aws

개발자 가이드

Amazon DocumentDB



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

Amazon DocumentDB: 개발자 가이드

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

Amazon의 상표 및 트레이드 드레스는 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 Amazon과 제휴 관계이거나 관련이 있거나 후원 관계와 관계없이 해당 소유자의 자산입니다.

Table of Contents

Amazon DocumentDB란 무엇인가 1
개요 1
클러스터 3
인스턴스 4
리전 및 가용 영역 6
리전 6
가용 영역 7
요금 9
무료 평가판 9
모니터링 9
인터페이스 10
AWS Management Console 10
AWS CLI 10
MongoDB 드라이버 10
다음 단계 11
작동 방법 11
Amazon DocumentDB 엔드포인트 13
TLS 지원 16
Amazon DocumentDB 스토리지 16
Amazon DocumentDB 복제 17
Amazon DocumentDB 신뢰성 18
읽기 기본 설정 옵션 19
TTL 삭제 23
청구 가능 리소스 23
문서 데이터베이스란? 26
사용 사례 27
문서 이해 28
문서 작업 33
시작 안내서 45
사전 조건 45
1단계: 클러스터 생성 47
2단계: 클러스터에 연결 49
3단계: 데이터 삽입 및 쿼리 51
4단계: 탐색 53

사용 빠른 시작 AWS CloudFormation 54
사전 조건 54
필요한 IAM 권한 55
Amazon EC2 키 페어 58
Amazon DocumentDB AWS CloudFormation 스택 시작 58
Amazon DocumentDB 클러스터 액세스 63
종료 방지 및 삭제 방지 63
MongoDB 호환성 64
MongoDB 5.0 호환성 64
Amazon DocumentDB 5.0의 새로운 기능 64
Amazon DocumentDB 5.0 시작하기 65
Amazon DocumentDB 5.0으로 업그레이드 또는 마이그레이션 65
기능적 차이 66
MongoDB 4.0 호환성 67
Amazon DocumentDB 4.0 기능 67
Amazon DocumentDB 4.0 시작하기 69
Amazon DocumentDB 4.0으로 업그레이드 또는 마이그레이션 69
기능적 차이 69
트랜잭션 71
요구 사항 71
모범 사례 71
제한 사항 72
모니터링 및 진단 73
트랜잭션 격리 수준 73
사용 사례 74
다중 문 트랜잭션 74
다중 컬렉션 트랜잭션 76
콜백 API의 트랜잭션 API 예제 77
코어 API의 트랜잭션 API 예제 77
지원되는 명령 111
지원되지 않는 기능 112
세션 112
인과 관계 일관성 113
재시도 가능한 쓰기 114
트랜잭션 오류 114
모범 사례 115

기본 운영 지침 115
인스턴스 크기 조정 116
인덱스 작업 118
인덱스 작성 118
인덱스 선택성 118
인덱스가 데이터 작성에 미치는 영향 118
누락된 인덱스 식별 119
사용되지 않는 인덱스 식별 119
보안 모범 사례 119
비용 최적화 120
지표를 통해 성능 문제 식별 120
성능 지표 보기 121
CloudWatch 경보 설정 121
성능 지표 평가 121
CloudWatch 지표를 사용한 Amazon DocumentDB 인스턴스 사용량 평가 123
쿼리 튜닝 124
TTL 및 시계열 워크로드 124
마이그레이션 125
클러스터 파라미터 그룹 작업 125
집계 파이프라인 쿼리 125
batchInsert 및 batchUpdate 126
MongoDB와 기능적 차이 127
Amazon DocumentDB의 기능적 이점 127
암시적 트랜잭션 127
업데이트된 기능적 차이 128
배열 인덱싱 129
다중 키 인덱스 130
문자열의 null 문자 130
역할 기반 액세스 제어 131
$regex 인덱싱 131
중첩된 문서에 대한 프로젝션 131
MongoDB와 기능적 차이 132
$vectorSearch 연산자 132
OpCountersCommand 133
관리자 데이터베이스 및 컬렉션 133
cursormaxTimeMS 133

explain() 133
인덱스 빌드 133
경로에 빈 키를 사용하여 조회 134
MongoDB API, 작업 및 데이터 형식 134
mongodump및 mongorestore 유틸리티 134
결과 주문 134
재시도 가능한 쓰기 135
희소 인덱스 135
$all 표현식 내에서 $elemMatch 사용 136
$ne, $nin, $nor, $not, $exists, 및 $elemMatch 인덱싱 136
필드 이름의 달러($) 및 점(.) 137
$lookup 137
$natural 및 역 정렬 141
지원되는 MongoDB API, 작업 및 데이터 형식 142
데이터베이스 명령 142
관리 명령 143
집계 144
인증 145
진단 명령 145
쿼리 및 쓰기 작업 146
역할 관리 명령 147
세션 명령 148
사용자 관리 149
샤딩 명령 150
쿼리 및 프로젝션 연산자 151
배열 연산자 152
Bitwise 연산자 152
설명 연산자 153
비교 연산자 153
요소 연산자 153
평가 쿼리 연산자 154
논리 연산자 154
프로젝션 연산자 154
업데이트 연산자 155
배열 연산자 155
Bitwise 연산자 156

필드 연산자 156
업데이트 한정자 156
지리 공간 157
지오메트리 지정자 157
쿼리 선택기 158
커서 메서드 158
집계 파이프라인 연산자 160
누적기 식 161
산술 연산자 162
배열 연산자 163
부울 연산 164
비교 연산자 165
조건식 연산자 165
데이터 유형 연산자 165
데이터 크기 연산자 166
날짜 연산자 166
리터럴 연산자 167
병합 연산자 167
자연 연산자 168
집합 연산자 168
단계 연산자 169
문자열 연산자 170
시스템 변수 172
텍스트 검색 연산자 172
유형 변환 연산자 172
변수 연산자 173
기타 연산자 173
데이터 타입 174
인덱스 및 인덱스 속성 175
인덱스 175
인덱스 속성 176
생성형 AI 177
SageMaker Canvas 177
SageMaker AI Canvas를 사용하여 코드 없는 ML 모델을 빌드하는 방법 177
SageMaker AI 도메인 및 사용자 프로필 구성 178
Amazon DocumentDB 및 SageMaker AI Canvas에 대한 IAM 액세스 권한 구성 178

SageMaker AI Canvas에 대한 데이터베이스 사용자 및 역할 생성 178
사용 가능한 리전 179
벡터 검색 179
벡터 삽입 180
벡터 인덱스 생성 181
인덱스 정의 가져오기 184
벡터 쿼리 185
특성 및 제한 사항 188
모범 사례 190
Amazon DocumentDB로 마이그레이션 191
마이그레이션 도구 191
AWS Database Migration Service 191
명령줄 유틸리티 192
Discovery 192
계획: Amazon DocumentDB 클러스터 요구 사항 196
마이그레이션 접근 방식 199
오프라인 199
온라인 200
하이브리드 201
마이그레이션 원본 203
마이그레이션 연결 204
테스트 207
마이그레이션 계획 테스트 고려 사항 208
성능 테스트 211
장애 조치 테스트 211
추가 리소스 211
마이그레이션 플레이북 212
마이그레이션 프로세스 212
추가 리소스 216
Amazon DocumentDB 엔진 버전 업그레이드 217
MVU 사전 조건 및 제한 사항 217
인플레이스 주요 버전 업그레이드 모범 사례 219
복제된 클러스터를 사용하여 인플레이스 주요 버전 업그레이드를 테스트합니다. 219
인플레이스 주요 버전 업그레이드를 하기 전에 219
인플레이스 주요 버전 업그레이드를 하는 동안 221
인플레이스 주요 버전 업그레이드를 한 후 222

인플레이스 주요 버전 업그레이드 수행 223
Amazon DocumentDB 3.6/4.0에서 업그레이드된 Amazon DocumentDB 5.0 클러스터와 새 Amazon DocumentDB 5.0 클러스터 간의 차이점 226
인플레이스 주요 버전 업그레이드 문제 해결 227
를 사용하여 업그레이드 AWS DMS 228
1단계: 변경 스트림 활성화 228
2단계: 변경 스트림 보존 기간 수정 229
3단계: 인덱스 마이그레이션 229
4단계: AWS DMS 복제 인스턴스 생성 230
5단계: AWS DMS 소스 엔드포인트 생성 233
6단계: AWS DMS 대상 엔드포인트 생성 235
7단계: 마이그레이션 작업 생성 및 실행 237
8단계: 애플리케이션 엔드포인트를 대상 Amazon DocumentDB 클러스터로 변경 239
보안 240
를 사용한 암호 관리 AWS Secrets Manager 241
Amazon DocumentDB와의 Secrets Manager 통합에 대한 제한 사항 241
를 사용한 기본 사용자 암호 관리 개요 AWS Secrets Manager 242
에서 기본 사용자 암호의 Amazon DocumentDB 관리 적용 AWS Secrets Manager 242
Secrets Manager를 사용하여 클러스터의 기본 사용자 암호 관리 243
데이터 보호 243
클라이언트 측 필드 레벨 암호화 244
저장 데이터 암호화 252
전송 중 데이터 암호화 257
키 관리 267
ID 및 액세스 관리 267
대상 268
ID를 통한 인증 269
정책을 사용하여 액세스 관리 272
Amazon DocumentDB에서 IAM을 사용하는 방법 274
자격 증명 기반 정책 예제 281
문제 해결 284
Amazon DocumentDB 리소스에 대한 액세스 권한 관리 286
자격 증명 기반 정책(IAM 정책) 사용 290
AWS Amazon DocumentDB에 대한 관리형 정책 294
Amazon DocumentDB API 권한 참조 312
IAM 자격 증명을 사용한 인증 320

IAM 사용자 및 역할 시작하기 321
컴퓨팅 유형 구성 324
인증 요청 모니터링 325
IAM 인증 사용 326
IAM을 지원하는 드라이버 326
IAM 자격 증명 인증 FAQ 326
Amazon DocumentDB 사용자 관리 327
기본 및 serviceadmin 사용자 327
추가 사용자 생성 328
자동으로 암호 교체 330
역할 기반 액세스 제어 330
RBAC 개념 331
RBAC 내장된 역할 시작하기 333
RBAC 사용자 정의 역할 시작하기 337
Amazon DocumentDB에 사용자로 연결 341
공통 명령 343
기능적 차이 348
Limits 348
역할 기반 액세스 제어를 사용한 데이터베이스 액세스 348
로깅 및 모니터링 357
인증서 업데이트 358
애플리케이션 및 Amazon DocumentDB 클러스터 업데이트 358
자동 서버 인증서 교체 362
문제 해결 363
FAQ 363
인증서 업데이트 — GovCloud 369
애플리케이션 및 Amazon DocumentDB 클러스터 업데이트 358
문제 해결 363
FAQ 363
규정 준수 확인 381
복원성 382
인프라 보안 383
VPC 엔드포인트(AWS PrivateLink) 383
VPC 엔드포인트에 대한 고려 사항 384
리전 가용성 385
Amazon DocumentDB API에 대한 인터페이스 VPC 엔드포인트 생성 386

Amazon DocumentDB API에 대한 VPC 엔드포인트 정책 생성 386
보안 모범 사례 387
이벤트 감사 388
지원되는 이벤트 389
감사 활성화 393
감사 비활성화 399
이벤트에 액세스 402
DML 이벤트 필터링 402
백업 및 복원 408
백업 및 복원: 개념 409
백업 스토리지 사용량 파악 410
데이터 덤프, 복원, 가져오기 및 내보내기 412
mongodump 412
mongorestore 413
mongoexport 413
mongoimport 414
자습서 415
클러스터 스냅샷 고려 사항 417
백업 스토리지 418
백업 기간 418
백업 보관 기간 420
클러스터 스냅샷 암호화 복사 420
자동 스냅샷과 수동 스냅샷 비교 421
수동 클러스터 스냅샷 생성 422
클러스터 스냅샷 복사 426
공유 스냅샷 복사 426
에서 스냅샷 복사 AWS 리전 427
제한 사항 427
암호화 처리 427
파라미터 그룹 고려 사항 428
클러스터 스냅샷 복사 428
클러스터 스냅샷 공유 435
암호화된 스냅샷 공유 435
스냅샷 공유 438
클러스터 스냅샷에서 복원 442
특정 시점으로 복원 449

클러스터 스냅샷 삭제 455
Amazon DocumentDB 관리 457
운영 작업 개요 457
Amazon DocumentDB 클러스터에 복제본 추가 458
클러스터 및 인스턴스 설명 459
클러스터 스냅샷 생성 461
스냅샷에서 복원 462
클러스터에서 인스턴스 제거 463
클러스터 삭제 463
글로벌 클러스터 464
글로벌 클러스터란 무엇입니까? 464
글로벌 클러스터는 어떻게 유용할까요? 464
글로벌 클러스터의 현재 한계는 무엇입니까? 465
빠른 시작 설명서 466
글로벌 클러스터 관리 480
글로벌 클러스터 연결 487
글로벌 클러스터 모니터링 488
재해 복구 489
클러스터 관리 504
클러스터 이해 505
클러스터 설정 507
클러스터 스토리지 구성 510
클러스터 상태 확인 513
클러스터 수명 주기 515
클러스터 조정 555
클러스터에 대한 볼륨 복제 558
클러스터의 내결함성 이해 570
인스턴스 관리 571
인스턴스 상태 확인 572
인스턴스 수명 주기 572
인스턴스 클래스 관리 595
NVMe 지원 인스턴스 605
서브넷 그룹 관리 607
서브넷 그룹 생성 608
서브넷 그룹 설명 614
서브넷 그룹 수정 617

서브넷 그룹 삭제 620
고가용성 및 복제 621
읽기 확장 622
높은 가용성 622
복제본 추가 623
장애 조치 624
복제 지연 628
인덱스 관리 629
Amazon DocumentDB 인덱스 생성 629
Amazon DocumentDB 인덱스 유지 관리 635
문서 압축 관리 635
문서 압축 관리 636
문서 압축 모니터링 638
이벤트 관리 639
이벤트 범주 보기 639
Amazon DocumentDB 이벤트 보기 642
리전 및 가용 영역 선택 644
리전 가용성 645
클러스터 파라미터 그룹 관리 647
클러스터 파라미터 그룹 설명 648
클러스터 파라미터 그룹 생성 654
클러스터 파라미터 그룹 수정 656
사용자 지정된 클러스터 파라미터 그룹을 사용하도록 클러스터 수정 661
클러스터 파라미터 그룹 복사 663
클러스터 파라미터 그룹 재설정 665
클러스터 파라미터 그룹 삭제 668
클러스터 파라미터 참조 671
엔드포인트 이해 686
클러스터 엔드포인트 찾기 686
인스턴스의 엔드포인트 찾기 689
엔드포인트에 연결 692
Amazon DocumentDB ARN 이해 694
ARN 생성 694
ARN 찾기 697
리소스에 태그 지정 700
리소스 태그 개요 700

태그 제약 조건 ... 701
태그 추가 또는 업데이트 ... 701
태그 나열 ... 703
태그 제거 ... 704
Amazon DocumentDB 유지 관리 ... 706
엔진 패치 알림 ... 706
보류 중인 유지 관리 보기 ... 708
엔진 업데이트 ... 709
사용자 시작 업데이트 ... 713
유지 관리 기간 관리 ... 714
운영 체제 업데이트 ... 716
서비스 연결 역할 이해 ... 719
서비스 연결 역할 권한 ... 720
서비스 연결 역할 만들기 ... 722
서비스 연결 역할 수정 ... 722
서비스 연결 역할 삭제 ... 722
Amazon DocumentDB 서비스 연결 역할에 대해 지원되는 리전 ... 723
변경 스트림 사용 ... 723
지원되는 연산자 ... 724
결제 ... 724
제한 사항 ... 725
변경 스트림 활성화 ... 726
Python에서 변경 스트림 사용 ... 727
전체 문서 조회 ... 730
변경 스트림 재개 ... 731
`startAtOperationTime`로 재개 ... 732
`postBatchResumeToken`로 재개 ... 734
트랜잭션 ... 735
로그 보존 기간 수정 ... 736
보조 인스턴스의 스트림 변경 ... 739
변경 스트림과 AWS Lambda 함께 사용 ... 740
제한 사항 ... 741
탄력적 클러스터 사용 ... 742
탄력적 클러스터 사용 사례 ... 742
사용자 프로필 ... 743
콘텐츠 관리 및 이력 기록 ... 743

탄력적 클러스터의 장점 743
AWS 서비스 통합 743
리전 및 버전 사용 가능 여부 744
탄력적 클러스터의 리전 가용성 744
버전 가용성 745
제한 사항 745
탄력적 클러스터 관리 745
쿼리 및 쓰기 작업 745
컬렉션 및 인덱스 관리 746
관리 및 진단 746
옵트인 기능 746
작동 방법 747
Amazon DocumentDB 탄력적 클러스터 샤딩 747
탄력적 클러스터 마이그레이션 751
탄력적 클러스터 규모 조정 751
탄력적 클러스터 안정성 751
탄력적 클러스터 스토리지 및 가용성 751
Amazon DocumentDB 4.0과 탄력적 클러스터 간의 기능적 차이 751
시작 753
사전 조건 754
1단계: 엘라스틱 클러스터 생성 755
2단계: 엘라스틱 클러스터에 연결 761
3단계: 수집 샤딩, 데이터 삽입 및 쿼리 762
4단계: 탐색 765
모범 사례 766
샤드 키 선택 766
연결 관리 766
샤딩되지 않은 컬렉션 767
탄력적 클러스터 크기 조정 767
탄력적 클러스터 모니터링 767
탄력적 클러스터 관리 768
엘라스틱 클러스터 구성 수정 768
엘라스틱 클러스터 모니터링 772
엘라스틱 클러스터 삭제 775
엘라스틱 클러스터 스냅샷 관리 778
탄력적 클러스터 중지 및 시작 791

탄력적 클러스터 유지 관리 ... 796
유휴 시(저장된) 데이터 암호화 ... 803
Amazon DocumentDB 탄력적 클러스터가에서 권한 부여를 사용하는 방법 AWS KMS ... 805
고객 관리형 키 생성 ... 806
Amazon DocumentDB 탄력적 클러스터에 대한 암호화 키 모니터링 ... 807
자세히 알아보기 ... 812
서비스 연결 역할 ... 813
탄력적 클러스터에 대한 서비스 연결 역할 권한 ... 813
Amazon DocumentDB 모니터링 ... 817
클러스터 상태 모니터링 ... 818
클러스터 상태 값 ... 818
클러스터 상태 모니터링 ... 820
인스턴스 상태 모니터링 ... 822
인스턴스 상태 값 ... 823
AWS Management Console 또는를 사용하여 인스턴스 상태 모니터링 AWS CLI ... 825
인스턴스 상태 값 ... 827
를 사용하여 인스턴스 상태 모니터링 AWS Management Console ... 827
Amazon DocumentDB 권장 사항 보기 ... 829
이벤트 구독 ... 831
이벤트 구독 ... 832
구독 관리 ... 835
범주 및 메시지 ... 839
CloudWatch를 사용하여 Amazon DocumentDB 모니터링 ... 842
Amazon DocumentDB 지표 ... 842
CloudWatch 데이터 보기 ... 854
Amazon DocumentDB 차원 ... 860
Opcounter 지표 모니터링 ... 861
데이터베이스 연결 모니터링 ... 861
CloudTrail로 Amazon DocumentDB API 호출 로깅 ... 861
CloudTrail의 Amazon DocumentDB 정보 ... 861
작업 프로파일링 ... 862
지원되는 연산자 ... 863
제한 사항 ... 864
프로파일러 활성화 ... 864
프로파일러 비활성화 ... 868
프로파일러 로그 내보내기 비활성화 ... 869

프로파일러 로그 액세스 872
공통 쿼리 872
성능 개선 도우미로 모니터링 873
성능 개선 도우미 개념 874
성능 개선 도우미 활성화 및 비활성화 877
성능 개선 도우미에 대한 액세스 정책 구성 880
성능 개선 도우미 대시보드를 사용한 지표 분석 885
성능 개선 도우미 API를 사용하여 지표 검색 904
성능 개선 도우미를 위한 Amazon CloudWatch 지표 918
카운터 지표에 대한 성능 개선 도우미 920
OpenSearch 통합 922
Amazon OpenSearch Service의 대상입니다. 922
1단계: Amazon OpenSearch Service 도메인 또는 OpenSearch 서버리스 컬렉션 생성 923
2단계: Amazon DocumentDB 클러스터에서 변경 스트림 활성화 923
3단계: Amazon S3 버킷 및 대상 도메인 또는 컬렉션에 쓸 수 있는 권한을 가진 파이프라인 역할 설정 923
4단계: 파이프라인 역할에 필요한 권한을 추가하여 X-ENI 생성 924
5단계: 파이프라인 생성 925
제한 사항 925
Amazon DocumentDB로 개발 927
DocumentDB에 연결 927
프로그래밍 방식으로 연결 927
Amazon VCP 외부에서 연결 955
복제본 세트로 연결 956
Studio 3T를 사용하여 연결 961
DataGrip을 사용하여 연결 972
Amazon EC2를 사용하여 연결 980
JDBC 드라이버를 사용하여 연결 1016
ODBC 드라이버를 사용하여 연결 1032
DocumentDB를 사용한 프로그램 1048
DocumentDB Java 프로그래밍 가이드 1048
JSON 스키마 검증 사용 1078
할당량 및 제한 1089
지원되는 인스턴스 유형 1089
지원되는 리전 1091
리전별 할당량 1092

집계 한도 1095
클러스터 한도 1095
인스턴스 제한 1097
명명 제약 조건 1099
TTL 제약 조건 1100
탄력적 클러스터 제한 1101
탄력적 클러스터 샤드 한도 1102
샤드당 탄력적 클러스터 CPU, 메모리, 연결 및 커서 제한 1102
쿼리 1104
문서 쿼리 1104
모든 문서 검색 1105
필드 값 매칭 1105
내장 문서 1105
포함된 문서의 필드 값 1105
배열 매칭 1106
배열의 값 매칭 1106
연산자 사용 1106
쿼리 계획 1107
쿼리 계획 1107
쿼리 계획 캐시 1109
결과의 설명 1109
스캔 및 필터 단계 1110
인덱스 교집합 1111
인덱스 병합 1111
다중 인덱스 교집합/병합 1112
복합 인덱스 1113
정렬 단계 1113
그룹 단계 1114
지리 공간 데이터 1114
개요 1
지리공간 데이터 인덱싱 및 저장 1114
지리 공간 데이터 쿼리 1116
제한 사항 1120
부분 인덱스 1120
부분 인덱스 생성 1120
지원되는 연산자 1120

부분 인덱스를 사용한 쿼리 1121
부분 인덱스 기능 1122
부분 인덱스 제한 사항 1126
텍스트 검색 1127
지원되는 기능 1127
Amazon DocumentDB 텍스트 인덱스 사용 1128
MongoDB와의 차이점 1133
모범 사례 및 지침 1133
제한 사항 1134
문제 해결 1135
연결 문제 1135
Amazon DocumentDB 엔드포인트에 연결할 수 없습니다. 1135
Amazon DocumentDB 인스턴스 연결 테스트 1140
유효하지 않은 엔드포인트에 연결 1141
연결 수에 영향을 미치는 드라이버 구성 1142
인덱스 생성 1142
인덱스 빌드 실패 1142
백그라운드 인덱스 빌드 지연 문제 및 실패 1142
성능 및 리소스 사용률 1143
통계 보기, 삽입, 업데이트, 삭제 1144
캐시 성능 분석 1145
장시간 실행 중이거나 차단된 쿼리를 찾아서 종료합니다 1146
쿼리 계획을 보고 쿼리를 최적화합니다 1148
엘라스틱 클러스터에서 쿼리 계획을 확인하려면 어떻게 해야 하나요? 1150
인스턴스에서 실행 중인 작업을 모두 나열합니다 1152
쿼리 진행 상황 알아보기 1155
시스템이 갑자기 느리게 실행되는 이유 알아보기 1157
CPU 사용률이 높은 원인 알아보기 1159
인스턴스에서 열린 커서를 찾으세요. 1160
현재 Amazon DocumentDB 엔진 버전 보기 1160
인덱스 사용량을 분석하고 사용하지 않는 인덱스를 식별 1160
누락된 인덱스를 식별 1163
유용한 쿼리 요약 1164
리소스 관리 API 참조 1166
작업 1166
Amazon DocumentDB (with MongoDB compatibility) 1169

Amazon DocumentDB Elastic Clusters 1354
데이터 타입 1425
Amazon DocumentDB (with MongoDB compatibility) 1427
Amazon DocumentDB Elastic Clusters 1503
일반적인 오류 1521
공통 파라미터 1523
릴리스 정보 1526
2025년 6월 18일 1528
새로운 특성 1528
2025년 5월 8일 1529
새로운 특성 1529
2025년 4월 2일 1529
새로운 특성 1529
버그 수정 및 기타 변경 1529
2025년 3월 24일 1529
새로운 특성 1529
2025년 2월 6일 1530
새로운 특성 1530
2025년 1월 28일 1530
새로운 특성 1530
2025년 1월 15일 1530
새로운 특성 1530
버그 수정 및 기타 변경 1530
2024년 12월 18일 1531
새로운 특성 1531
2024년 11월 12일 1531
새로운 특성 1531
2024년 11월 6일 1531
새로운 특성 1531
2024년 11월 1일 1532
새로운 특성 1532
2024년 10월 22일 1532
새로운 특성 1532
2024년 9월 18일 1532
새로운 특성 1532
2024년 9월 17일 1532

새로운 특성 1532
버그 수정 및 기타 변경 1533
2024년 8월 22일 1533
새로운 특성 1533
2024년 8월 20일 1533
새로운 특성 1533
2024년 8월 8일 1534
새로운 특성 1534
2024년 7월 23일 1534
새로운 특성 1534
버그 수정 및 기타 변경 1535
2024년 7월 22일 1535
새로운 특성 1535
버그 수정 및 기타 변경 1536
2024년 7월 9일 1536
새로운 특성 1536
2024년 7월 8일 1536
새로운 특성 1536
2024년 6월 25일 1537
새로운 특성 1537
2024년 5월 29일 1537
새로운 특성 1537
2024년 4월 3일 1537
새로운 특성 1537
버그 수정 및 기타 변경 1538
2024년 2월 22일 1538
새로운 특성 1538
2024년 1월 30일 1539
새로운 특성 1539
2024년 1월 10일 1539
새로운 특성 1539
버그 수정 및 기타 변경 1540
2023년 12월 20일 1540
기타 변경사항 1540
2023년 12월 13일 1541
새로운 특성 1541

2023년 11월 29일 1541
새로운 특성 1541
2023년 11월 21일 1541
새로운 특성 1541
2023년 11월 17일 1541
새로운 특성 1541
버그 수정 및 기타 변경 1541
2023년 11월 6일 1542
새로운 특성 1542
버그 수정 및 기타 변경 1542
2023년 9월 25일 1542
새로운 특성 1542
2023년 9월 20일 1542
새로운 기능 1542
2023년 9월 15일 1543
새로운 특성 1543
2023년 9월 11일 1543
새로운 특성 1543
2023년 8월 3일 1543
새로운 기능 1543
2023년 7월 13일 1544
새로운 특성 1544
버그 수정 및 기타 변경 1544
2023년 6월 7일 1544
버그 수정 및 기타 변경 1544
2023년 5월 10일 1545
버그 수정 및 기타 변경 1545
2023년 4월 4일 1545
버그 수정 및 기타 변경 1545
2023년 3월 22일 1545
새로운 특성 1545
2023년 3월 1일 1546
새로운 특성 1546
2023년 2월 27일 1546
버그 수정 및 기타 변경 1546
2023년 2월 2일 1547

버그 수정 및 기타 변경 1547
2022년 11월 30일 1547
새로운 특성 1547
2022년 8월 9일 1547
새로운 특성 1547
버그 수정 및 기타 변경 1548
2022년 7월 25일 1548
새로운 특성 1548
2022년 6월 27일 1548
새로운 특성 1548
2022년 4월 29일 1548
새로운 특성 1548
2022년 4월 7일 1549
새로운 특성 1549
2022년 3월 16일 1549
새로운 특성 1549
2022년 2월 8일 1549
새로운 특성 1549
2022년 1월 24일 1549
새로운 특성 1549
2022년 1월 21일 1550
새로운 특성 1550
2021년 10월 25일 1550
새로운 특성 1550
버그 수정 및 기타 변경 1551
2021년 6월 24일 1551
새로운 특성 1551
2021년 5월 4일 1551
새로운 특성 1551
버그 수정 및 기타 변경 1552
2021년 1월 15일 1552
새로운 특성 1552
2020년 11월 9일 1553
새로운 특성 1553
버그 수정 및 기타 변경 1554
2020년 10월 30일 1555

새로운 특성 1555
버그 수정 및 기타 변경 1555
2020년 9월 22일 1555
새로운 특성 1555
버그 수정 및 기타 변경 1556
2020년 7월 10일 1556
새로운 특성 1556
버그 수정 및 기타 변경 1556
2020년 6월 30일 1556
새로운 특성 1556
버그 수정 및 기타 변경 1557
문서 기록 1558
..... mdlxix

Amazon DocumentDB(MongoDB 호환)란 무엇인가

Amazon DocumentDB(MongoDB와 호환됨)는 빠르고 안정적이며 완전 관리형 데이터베이스 서비스입니다. Amazon DocumentDB를 사용하면 클라우드에서 MongoDB 호환 데이터베이스를 쉽게 설치, 운영 및 규모를 조정할 수 있습니다. Amazon DocumentDB를 사용하면 MongoDB에서 사용하는 것과 동일한 애플리케이션 코드를 실행하고 동일한 드라이버와 도구를 사용할 수 있습니다.

Amazon DocumentDB를 사용하기 전에 작동 방법에 설명된 개념과 기능을 검토해야 합니다. 그 이후에는 시작 안내서의 단계를 완료합니다.

주제

- Amazon DocumentDB 개요
- 클러스터
- 인스턴스
- 리전 및 가용 영역
- Amazon DocumentDB 요금
- 모니터링
- 인터페이스
- 다음 단계
- Amazon DocumentDB: 작동 방식
- 문서 데이터베이스란?

Amazon DocumentDB 개요

다음은 Amazon DocumentDB의 일부 고급 기능입니다.

- Amazon DocumentDB는 인스턴스 기반 클러스터와 탄력적 클러스터라는 두 가지 유형의 클러스터를 지원합니다. 탄력적 클러스터는 초당 수백만 읽기/쓰기 및 페타바이트의 스토리지 용량을 갖춘 워크로드를 지원합니다. 탄력적 클러스터에 대한 자세한 내용은 Amazon DocumentDB 탄력적 클러스터 사용을 참조하세요. 아래 내용은 Amazon DocumentDB 인스턴스 기반 클러스터를 참조합니다.
- Amazon DocumentDB는 데이터베이스 스토리지가 증가함에 따라 스토리지 볼륨의 크기를 자동으로 증가시킵니다. 스토리지 볼륨은 10GB, 최대 128TiB까지 증가합니다. 향후 증가를 처리하기 위해 클러스터에 추가 스토리지를 프로비저닝할 필요가 없습니다.

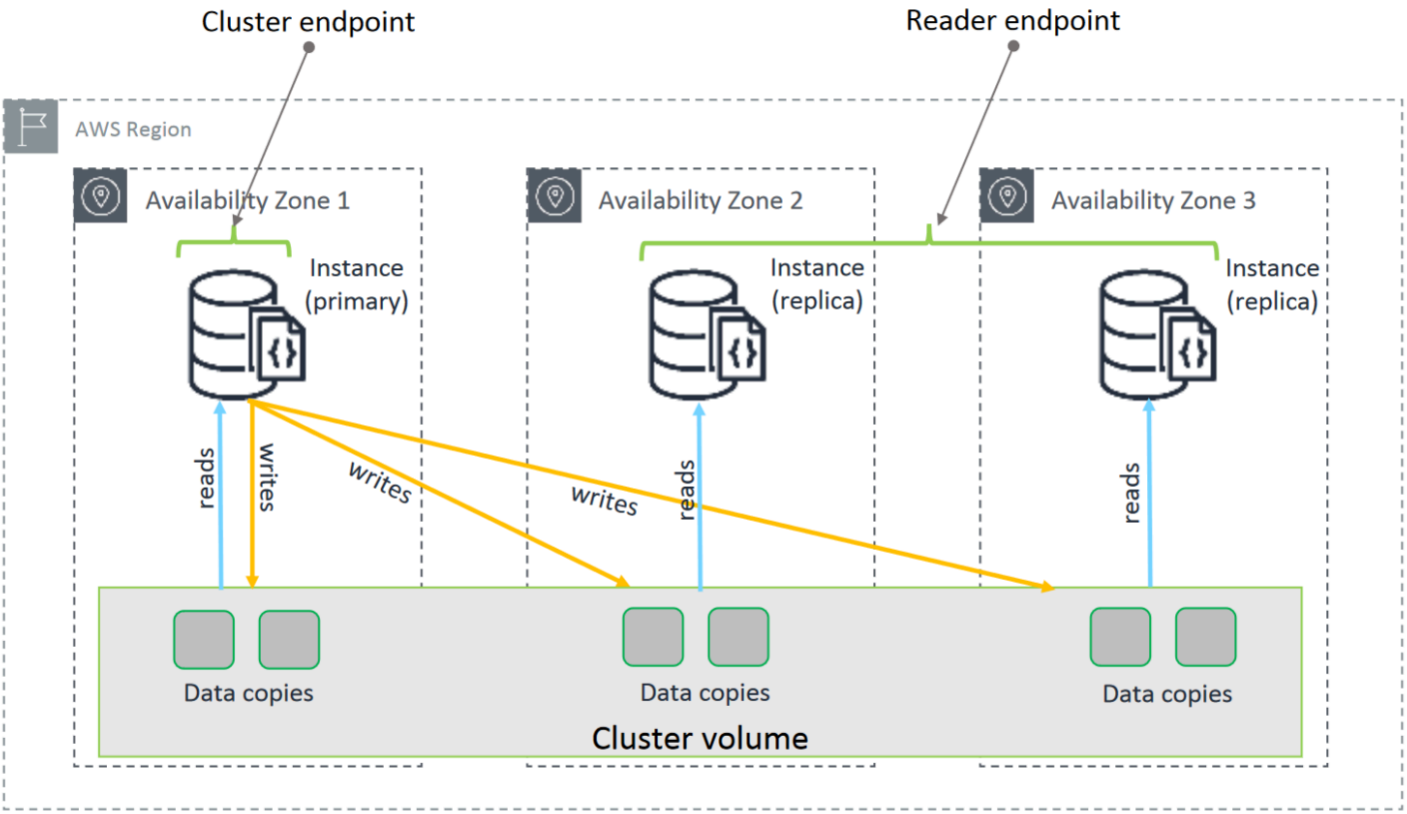
- Amazon DocumentDB를 사용하면 최대 15개의 복제본 인스턴스를 생성하여 읽기 처리량을 높여 높은 볼륨의 애플리케이션 요청을 지원할 수 있습니다. Amazon DocumentDB 복제본은 동일한 기본 스토리지를 공유하므로 비용이 절감되고 복제본 노드에서 쓰기를 수행할 필요가 없습니다. 이 기능을 사용하면 읽기 요청을 처리할 수 있는 처리 능력이 향상되고 복제본 지연 시간이 최대 한 자릿수 밀리초로 단축됩니다. 스토리지 볼륨 크기에 관계없이 몇 분 안에 복제본을 추가할 수 있습니다. 또한 Amazon DocumentDB는 리더 엔드포인트를 제공하므로 애플리케이션이 복제본이 추가 및 제거될 때 추적할 필요 없이 연결할 수 있습니다.
- Amazon DocumentDB를 사용하면 각 인스턴스의 계산 및 메모리 리소스를 위 또는 아래로 확장할 수 있습니다. 컴퓨팅 조정 작업은 일반적으로 몇 분이면 완료됩니다.
- Amazon DocumentDB는 Amazon Virtual Private Cloud(Amazon VPC)에서 실행되므로 데이터베이스를 자체 가상 네트워크에서 격리할 수 있습니다. 또한 클러스터에 대한 네트워크 액세스를 제어하도록 방화벽 설정을 구성할 수 있습니다.
- Amazon DocumentDB는 지속적으로 클러스터 상태를 모니터링합니다. 인스턴스 장애 발생 시 Amazon DocumentDB는 인스턴스 및 관련 프로세스를 자동으로 재시작합니다. Amazon DocumentDB는 데이터베이스 다시 실행 로그의 충돌 복구 재생이 필요하지 않으므로 재시작 시간이 크게 단축됩니다. 또한 Amazon DocumentDB는 데이터베이스 프로세스에서 데이터베이스 캐시를 분리하여 인스턴스를 다시 시작해도 캐시가 계속 유지되도록 합니다.
- 인스턴스 장애 시에서는 Amazon DocumentDB는 다른 가용 영역에서 작성하는 최대 15개의 Amazon DocumentDB 복제본 중 하나로 장애 조치를 자동화합니다. 복제본이 프로비저닝되지 않은 상태에서 오류가 발생하면 Amazon DocumentDB는 자동으로 새 Amazon DocumentDB 인스턴스를 작성하려고 합니다.
- Amazon DocumentDB의 백업 기능은 클러스터의 시점 복구를 가능하게 합니다. 이 기능을 통해 클러스터를 보존 기간 중 어느 시점(초)으로나 복원할 수 있습니다(마지막 5분까지 가능). 자동 백업 보존 기간은 최대 35일까지 구성할 수 있습니다. 자동 백업은 99.9999999% 의 내구성을 제공하도록 설계된 Amazon Simple Simple S3 Service(Amazon Simple S3)에 저장됩니다. Amazon DocumentDB 백업은 자동, 증분 및 지속적이며 클러스터 성능에 영향을 주지 않습니다.
- Amazon DocumentDB를 사용하면 AWS Key Management Service ()를 통해 생성하고 제어하는 키를 사용하여 데이터베이스를 암호화할 수 있습니다AWS KMS. Amazon DocumentDB 암호화와 함께 실행되는 데이터베이스 클러스터에서는 기본 저장소에 저장된 정지 상태의 데이터가 암호화됩니다. 동일한 클러스터에 있는 자동화된 백업, 스냅샷 및 복제본도 암호화됩니다.
- Amazon DocumentDB는 연방정 위험 및 인증 관리 프로그램(FedRAMP)에 따라 권한이 부여됩니다. 여기에는 AWS GovCloud(미국) 리전에 대한 FedRAMP High 권한 부여와 AWS 미국 동부/서부 리전에 대한 FedRAMP Moderate 권한이 있습니다. AWS 및 규정 준수 노력에 대한 자세한 내용은 AWS 규정 준수 프로그램 제공 범위 내 서비스를 참조하세요.

AWS 서비스를 처음 사용하는 경우 다음 리소스를 사용하여 자세히 알아보세요.

- AWS 는 컴퓨팅, 데이터베이스, 스토리지, 분석 및 기타 기능을 위한 서비스를 제공합니다. 모든 AWS 서비스에 대한 개요는 Amazon Web Services를 사용한 클라우드 컴퓨팅을 참조하세요.
- AWS 는 여러 데이터베이스 서비스를 제공합니다. 환경에 가장 적합한 서비스에 대한 지침은 AWS의 데이터베이스를 참조하십시오.

클러스터

클러스터는 0~16개의 인스턴스와 해당 인스턴스의 데이터를 관리하는 클러스터 스토리지 볼륨으로 구성됩니다. 모든 쓰기는 기본 인스턴스를 통해 수행됩니다. 모든 인스턴스(기본 및 복제본)는 읽기를 지원합니다. 클러스터의 데이터는 클러스터 볼륨에 저장되며 복사본은 세 개의 다른 가용 영역에 있습니다.



Amazon DocumentDB 5.0 인스턴스 기반 클러스터는 데이터베이스 클러스터에 대해 Amazon DocumentDB 표준 및 Amazon DocumentDB I/O 최적화라는 두 가지 스토리지 구성을 지원합니다. 자세한 내용은 Amazon DocumentDB 클러스터 스토리지 구성을 참조하세요.

인스턴스

Amazon DocumentDB 인스턴스는 클라우드에 있는 격리된 데이터베이스 환경입니다. 인스턴스에는 사용자가 만든 여러 개의 데이터베이스가 포함될 수 있습니다. AWS Management Console 또는를 사용하여 인스턴스를 생성하고 수정할 수 있습니다 AWS CLI.

인스턴스의 계산 및 메모리 용량은 인스턴스 클래스에 따라 결정됩니다. 사용자의 요구 사항에 가장 잘 맞는 인스턴스를 선택할 수 있습니다. 시간이 지나면서 요구 사항이 바뀌면 다른 인스턴스 클래스를 선택할 수 있습니다. 인스턴스 클래스 사양은 인스턴스 클래스 사양을 참조하십시오.

Amazon DocumentDB 인스턴스는 Amazon VPC 환경에서만 실행됩니다. Amazon VPC를 사용하면 가상 네트워킹 환경을 제어할 수 있습니다. 자신의 IP 주소 범위를 선택하고 서브넷을 생성하며 라우팅 및 액세스 제어 목록(ACL)을 구성할 수 있습니다.

Amazon DocumentDB 인스턴스를 작성하려면 먼저 인스턴스를 포함할 클러스터를 작성해야 합니다.

모든 리전에서 모든 인스턴스 클래스가 지원되지는 않습니다. 다음 표에는 각 리전에서 지원하는 인스턴스 클래스가 나와 있습니다.

> **Note**
>
> 각 인스턴스 클래스에서 Amazon DocumentDB가 지원하는 인스턴스 유형의 전체 목록은 섹션을 참조하세요인스턴스 클래스 사양.

리전별 지원되는 인스턴스 클래스

리전	R6GD	R6G	R5	R4	T4G	T3
미국 동부(오하이오)	지원	지원	지원	지원	지원	지원
미국 동부(버지니아 북부)	지원	지원	지원	지원	지원	지원
미국 서부(오리건)	지원	지원	지원	지원	지원	지원
아프리카(케이프타운)		지원	지원		지원	지원

리전	R6GD	R6G	R5	R4	T4G	T3
남아메리카(상파울루)	지원	지원	지원		지원	지원
아시아 태평양(홍콩)		지원	지원		지원	지원
아시아 태평양(하이데라바드)			지원			지원
아시아 태평양(뭄바이)	지원	지원	지원		지원	지원
아시아 태평양(서울)	지원	지원	지원		지원	지원
아시아 태평양(시드니)	지원	지원	지원		지원	지원
아시아 태평양(싱가포르)	지원	지원	지원		지원	지원
아시아 태평양(도쿄)	지원	지원	지원		지원	지원
캐나다(중부)	지원	지원	지원		지원	지원
유럽(프랑크푸르트)	지원	지원	지원		지원	지원
유럽(아일랜드)	지원	지원	지원	지원	지원	지원
유럽(런던)	지원	지원	지원		지원	지원
유럽(밀라노)		지원	지원		지원	지원
유럽(파리)	지원	지원	지원		지원	지원

리전	R6GD	R6G	R5	R4	T4G	T3
유럽(스페인)		지원	지원		지원	지원
유럽(스톡홀름)		지원	지원		지원	지원
중동(UAE)		지원	지원		지원	지원
중국(베이징)	지원	지원	지원		지원	지원
중국(닝샤)		지원	지원		지원	지원
이스라엘(텔아비브)		지원	지원		지원	지원
AWS GovCloud(미국 서부)	지원	지원	지원		지원	지원
AWS GovCloud(미국 동부)	지원	지원	지원		지원	지원

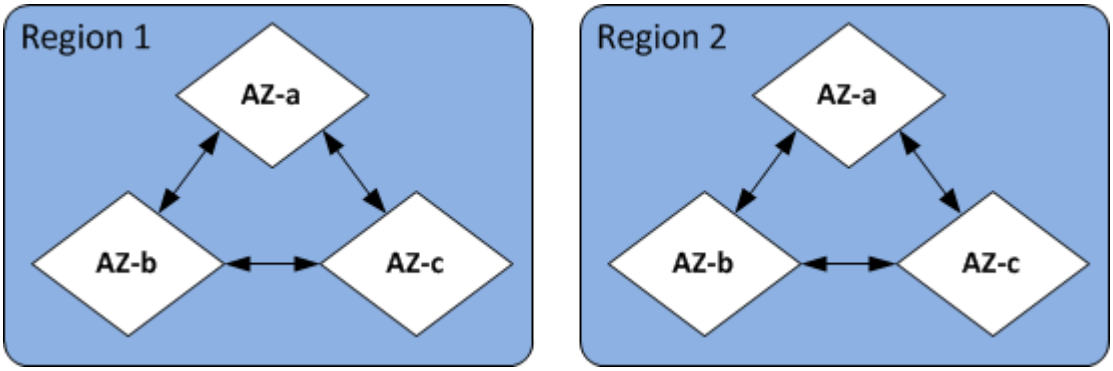
리전 및 가용 영역

리전 및 가용 영역은 클러스터 및 인스턴스의 물리적 위치를 정의합니다.

리전

AWS 클라우드 컴퓨팅 리소스는 전 세계 여러 지역(예: 북미, 유럽 또는 아시아)의 고가용성 데이터 센터 시설에 있습니다. 각 데이터 센터 위치를 리전이라고 합니다.

각 AWS 리전은 다른 AWS 리전과 완전히 격리되도록 설계되었습니다. 각 리전 내에는 가용 영역이 여러 개 있습니다. 서로 다른 가용 영역에서 노드를 시작하면 가능한 최고 수준의 내결함성을 갖출 수 있습니다. 다음 다이어그램은 AWS 리전 및 가용 영역의 작동 방식을 전체적으로 보여줍니다.



가용 영역

각 AWS 리전에는 가용 영역이라는 고유한 위치가 여러 개 포함되어 있습니다. 각 가용 영역은 다른 가용 영역의 장애로부터 격리되고 같은 리전의 다른 가용 영역에 경제적이고 지연 시간이 낮은 네트워크 연결을 제공하도록 엔지니어링됩니다. 여러 가용 영역에서 특정 클러스터에 대한 인스턴스를 시작하면 가용 영역에 드물게라도 장애가 발생할 경우 애플리케이션을 보호할 수 있습니다.

Amazon DocumentDB 아키텍처는 스토리지와 컴퓨팅을 분리합니다. 스토리지 계층의 경우 Amazon DocumentDB는 3개의 AWS 가용 영역에 걸쳐 6개의 데이터 사본을 복제합니다. 예를 들어, 두 개의 가용 영역만 지원하는 영역에서 Amazon DocumentDB 클러스터를 시작하는 경우 데이터 스토리지는 세 개의 가용 영역에 걸쳐 6가지 방식으로 복제되지만 컴퓨팅 인스턴스는 두 개의 가용 영역에서만 사용할 수 있습니다.

다음 표에는 지정된에서 클러스터의 컴퓨팅 인스턴스를 프로비저닝하는 AWS 리전 데 사용할 수 있는 가용 영역 수가 나열되어 있습니다.

리전 이름	지역	가용 영역(컴퓨팅)
미국 동부(오하이오)	`us-east-2`	3
미국 동부(버지니아 북부)	`us-east-1`	6
미국 서부(오리건)	`us-west-2`	4
아프리카(케이프타운)	`af-south-1`	3
남아메리카(상파울루)	`sa-east-1`	3
아시아 태평양(홍콩)	`ap-east-1`	3

리전 이름	지역	가용 영역(컴퓨팅)
아시아 태평양(하이데라바드)	ap-south-2	3
아시아 태평양(뭄바이)	ap-south-1	3
아시아 태평양(서울)	ap-northeast-2	4
아시아 태평양(싱가포르)	ap-southeast-1	3
아시아 태평양(시드니)	ap-southeast-2	3
아시아 태평양(도쿄)	ap-northeast-1	3
캐나다(중부)	ca-central-1	3
중국(베이징) 리전	cn-north-1	3
중국(닝샤)	cn-northwest-1	3
유럽(프랑크푸르트)	eu-central-1	3
유럽(아일랜드)	eu-west-1	3
유럽(런던)	eu-west-2	3
유럽(밀라노)	eu-south-1	3
유럽(파리)	eu-west-3	3
유럽(스페인)	eu-south-2	3
유럽(스톡홀름)	eu-north-1	3
중동(UAE)	me-central-1	3
이스라엘(텔아비브)	il-central-1	3

리전 이름	지역	가용 영역(컴퓨팅)
AWS GovCloud(미국 서부)	us-gov-west-1	3
AWS GovCloud(미국 동부)	us-gov-east-1	3

Amazon DocumentDB 요금

Amazon DocumentDB 클러스터는 다음 구성 요소를 기준으로 청구됩니다.

- 인스턴스 시간(시간당)—인스턴스의 인스턴스 클래스(예: db.r5.xlarge)를 기준으로 합니다. 요금은 시간 단위로 고시되지만, 청구서는 초 단위로 계산되고 시간을 10진수 형식으로 표시합니다. Amazon DocumentDB 사용량은 최소 10분을 기준으로 1초 단위로 청구됩니다. 자세한 내용은 인스턴스 클래스 관리 단원을 참조하십시오.
- I/O 요청(매월 100만 건당 요청) — 청구 주기 내에 수행하는 총 스토리지 I/O 요청 수.
- 백업 저장소(배월 단위 GiB) - 백업 저장소는 자동화된 데이터베이스 백업 및 사용 중인 데이터베이스 스냅샷과 관련된 저장소입니다. 백업 보존 기간을 연장하거나 추가 데이터베이스 스냅샷을 찍으면 데이터베이스가 사용하는 백업 스토리지가 증가합니다. 백업 스토리지는 GB-월 단위로 측정되며 초당으로 적용되지 않습니다. 자세한 내용은 Amazon DocumentDB에서 백업 및 복원 단원을 참조하십시오.
- 데이터 전송(GB당) - 인터넷 또는 기타 AWS 리전에서 인스턴스 안팎으로 데이터를 전송합니다.

자세한 내용은 Amazon DocumentDB 요금을 참조하십시오.

무료 평가판

1개월 무료 평가판을 사용하여 Amazon DocumentDB를 무료로 사용해 볼 수 있습니다. 자세한 내용은 Amazon DocumentDB 요금의 무료 평가판을 참조하거나 Amazon DocumentDB 무료 평가판 FAQ를 참조하십시오.

모니터링

인스턴스의 성능과 상태를 추적할 수 있는 여러 가지 방법이 있습니다. 무료 Amazon CloudWatch 서비스를 사용하여 인스턴스의 성능과 상태를 모니터링할 수 있습니다. 성능 차트는 Amazon DocumentDB

콘솔에서 찾을 수 있습니다. 인스턴스, 스냅샷, 파라미터 그룹 또는 보안 그룹에서 변경 사항이 발생할 때 알려줄 Amazon DocumentDB 이벤트에 가입할 수 있습니다.

자세한 내용은 다음 자료를 참조하세요.

- CloudWatch를 사용하여 Amazon DocumentDB 모니터링
- 를 사용하여 Amazon DocumentDB API 호출 로깅 AWS CloudTrail

인터페이스

AWS Management Console 및를 포함하여 Amazon DocumentDB와 상호 작용하는 방법에는 여러 가지가 있습니다 AWS CLI.

AWS Management Console

AWS Management Console 는 간단한 웹 기반 사용자 인터페이스입니다. 콘솔에서 프로그래밍 없이 클러스터 및 인스턴스를 관리할 수 있습니다. Amazon DocumentDB 콘솔에 액세스하려면에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.

AWS CLI

AWS Command Line Interface (AWS CLI)를 사용하여 Amazon DocumentDB 클러스터 및 인스턴스를 관리할 수 있습니다. 최소한의 구성으로 원하는 터미널 프로그램에서 Amazon DocumentDB 콘솔에서 제공하는 모든 기능을 사용할 수 있습니다.

- 를 설치하려면 명령줄 인터페이스 설치를 AWS CLI참조하세요. AWS
- Amazon DocumentDB AWS CLI 용 사용을 시작하려면 AWS Amazon DocumentDB용 명령줄 인터페이스 참조를 참조하세요.

MongoDB 드라이버

Amazon DocumentDB 클러스터에서 애플리케이션을 개발하고 작성하는 경우, MongoDB 드라이버를 Amazon DocumentDB와 함께 사용할 수도 있습니다. 자세한 내용은 TLS가 활성화된 상태에서 연결 또는의 MongoDB 셸 탭을 참조하세요TLS가 비활성화된 상태에서 연결.

다음 단계

앞 섹션에서는 Amazon DocumentDB가 제공하는 기본 인프라 구성 요소에 대해 소개했습니다. 다음으로 무엇을 해야 할까요? 환경에 따라 시작하려면 다음 주제 중 하나를 참조하십시오.

- 를 사용하여 클러스터와 인스턴스를 생성하여 Amazon DocumentDB를 시작합니다 AWS CloudFormation 를 사용한 Amazon DocumentDB 빠른 시작 AWS CloudFormation.
- 시작 안내서의 지침에 따라 클러스터 및 인스턴스를 생성하여 Amazon DocumentDB를 시작하십시오.
- Amazon DocumentDB 엘라스틱 클러스터 시작하기의 지침에 따라 탄력적 클러스터를 생성하여 Amazon DocumentDB를 시작하십시오.
- Amazon DocumentDB로 마이그레이션의 지침을 사용하여 MongoDB 구현을 Amazon DocumentDB로 마이그레이션하기

Amazon DocumentDB: 작동 방식

Amazon DocumentDB(MongoDB와 호환됨)는 완전 관리형의 MongoDB와 호환됨 데이터베이스 서비스입니다. Amazon DocumentDB를 사용하면 MongoDB에서 사용하는 것과 동일한 애플리케이션 코드를 실행하고 동일한 드라이버와 도구를 사용할 수 있습니다. Amazon DocumentDB는 MongoDB 3.6, 4.0, 5.0과 호환됩니다.

주제

- Amazon DocumentDB 엔드포인트
- TLS 지원
- Amazon DocumentDB 스토리지
- Amazon DocumentDB 복제
- Amazon DocumentDB 신뢰성
- 읽기 기본 설정 옵션
- TTL 삭제
- 청구 가능 리소스

Amazon DocumentDB를 사용할 때는 클러스터를 생성하는 것부터 시작합니다. 클러스터는 0개 이상의 데이터베이스 인스턴스 및 이 인스턴스의 데이터를 관리하는 클러스터 볼륨으로 구성됩니다.

Amazon DocumentDB 클러스터 볼륨은 여러 가용 영역에 걸쳐 있는 가상 데이터베이스 스토리지 볼륨입니다. 각 가용 영역마다 클러스터 데이터 복사본이 있습니다.

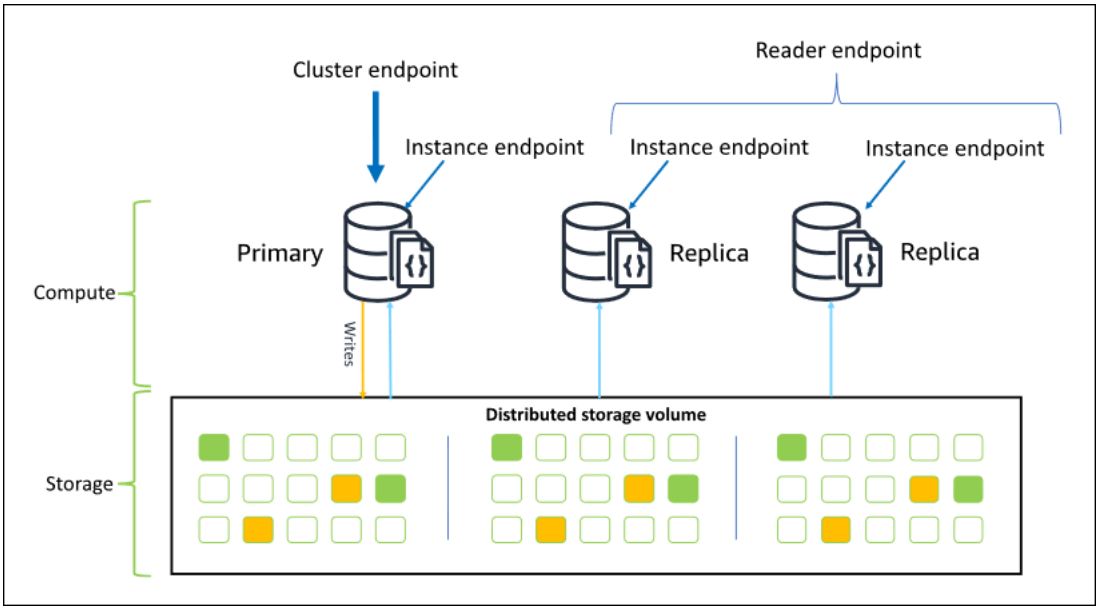
Amazon DocumentDB 클러스터는 다음 두 가지 구성 요소로 구성됩니다.

- 클러스터 볼륨—클라우드 네이티브 스토리지 서비스를 사용하여 3개의 가용 영역에 걸쳐 6가지 방식으로 데이터를 복제하여 내구성이 뛰어나고 가용성이 뛰어난 스토리지를 제공합니다. Amazon DocumentDB 클러스터는 하나의 클러스터 볼륨을 가지고 있으며, 최대 128TiB의 데이터를 저장할 수 있습니다.
- 인스턴스—데이터베이스에 대한 처리 능력을 제공하고, 데이터를 클러스터 스토리지 볼륨에 쓰고, 데이터를 읽어냅니다. Amazon DocumentDB 클러스터에는 0~16개의 인스턴스가 있을 수 있습니다.

인스턴스는 두 가지 역할 중 하나를 수행합니다.

- 기본 인스턴스—읽기 및 쓰기 작업을 지원하고 클러스터 볼륨에 대한 모든 데이터 수정을 수행합니다. 각 Amazon DocumentDB 클러스터에는 기본 인스턴스가 하나씩 있습니다.
- 복제본 인스턴스 - 읽기 작업만 지원합니다. Amazon DocumentDB 클러스터는 기본 인스턴스 외에 최대 15개의 복제본을 가질 수 있습니다. 복제본이 여러 개 있으면 읽기 워크로드를 분산시킬 수 있습니다. 또한 복제본을 별도의 가용 영역에 두어 클러스터 가용성을 높일 수 있습니다.

다음 다이어그램은 Amazon DocumentDB 클러스터의 클러스터 볼륨, 기본 인스턴스 및 복제본 간의 관계를 보여줍니다:



클러스터 인스턴스의 인스턴스 클래스가 같을 필요는 없으며 이 인스턴스를 원하는 대로 프로비저닝하고 종료할 수 있습니다. 이 아키텍처를 사용하여 클러스터 컴퓨팅 파워를 스토리지와 별개로 확장할 수 있습니다.

애플리케이션이 기본 인스턴스에 데이터를 쓸 때 기본 인스턴스는 클러스터 볼륨에 내구성 있는 쓰기를 실행합니다. 그런 다음 각 활성 복제본에 해당 쓰기 상태(데이터 제외)를 복제합니다. Amazon DocumentDB 복제본은 쓰기 처리에 관여하지 않으므로 Amazon DocumentDB 복제본은 읽기 확장에 유리합니다. Amazon DocumentDB 복제본에서의 읽기 작업은 최종적으로 복제본 지연 시간을 최소화하면서 일관되게 이루어지는데, 이는 기본 인스턴스가 데이터를 쓴 후 보통 100밀리초 미만입니다. 복제본에서 읽기는 기본 인스턴스에 쓰여진 순서대로 읽힙니다. 복제본 지연 시간은 데이터 변경 비율에 따라 달라지며 쓰기 작업 기간 집중적으로 일어나는 기간에는 복제본 지연 시간이 늘어날 수 있습니다. 자세한 내용은 Amazon DocumentDB 지표의 `ReplicationLag` 지표를 참조하십시오.

Amazon DocumentDB 엔드포인트

Amazon DocumentDB는 다양한 사용 사례를 제공하기 위해 여러 연결 옵션을 제공합니다. Amazon DocumentDB 클러스터의 인스턴스에 연결하려면 인스턴스의 엔드포인트를 지정합니다. 엔드포인트란 콜론으로 구분된 호스트 주소와 포트 번호입니다.

리더 엔드포인트 또는 인스턴스 엔드포인트에 연결하는 특정 사용 사례가 없는 경우 클러스터 엔드포인트를 사용하고 복제본 세트 모드(Amazon DocumentDB에 복제본 세트로 연결 참조)에서 클러스터에 연결하는 것이 좋습니다. 요청을 복제본으로 라우팅하려면 애플리케이션의 읽기 일관성 요구 사항을 충족하면서 읽기 확장성을 최대화하는 드라이버 읽기 기본 설정을 선택합니다. `secondaryPreferred` 읽기 기본 설정은 복제본 읽기를 활성화하고, 더 많은 작업을 수행할 수 있도록 기본 인스턴스를 비웁니다.

Amazon DocumentDB 클러스터에서 사용할 수 있는 엔드포인트는 다음과 같습니다.

클러스터 엔드포인트

클러스터 엔드포인트는 클러스터의 현재 기본 인스턴스에 연결됩니다. 읽기 및 쓰기 작업에 클러스터 엔드포인트를 사용할 수 있습니다. Amazon DocumentDB 클러스터에는 클러스터 엔드포인트가 하나만 있습니다.

클러스터 엔드포인트는 클러스터에 대한 읽기 및 쓰기 연결에 장애 조치 지원을 제공합니다. 클러스터의 현재 기본 인스턴스가 실패하고 클러스터에 적어도 하나의 활성 읽기 전용 복제본이 있으면 클러스터 엔드포인트가 연결 요청을 새 기본 인스턴스에 자동으로 리디렉션합니다. Amazon DocumentDB 클러스터에 연결할 때는 클러스터 엔트포인트를 사용하여 복제본 설정 모드로 클러스터에 연결하는 것이 좋습니다(Amazon DocumentDB에 복제본 세트로 연결 참조).

다음은 Amazon DocumentDB 클러스터 엔드포인트의 예제입니다.

```
sample-cluster.cluster-123456789012.us-east-1.docdb.amazonaws.com:27017
```

다음은 이 클러스터 엔드포인트를 사용하는 연결 문자열 예제입니다.

```
mongodb://username:password@sample-cluster.cluster-123456789012.us-
east-1.docdb.amazonaws.com:27017
```

클러스터 엔드포인트 찾기에 대한 자세한 내용은 클러스터 엔드포인트 찾기 단원을 참조하십시오.

리더 엔드포인트

리더 엔드포인트는 클러스터에 있는 사용 가능한 모든 복제본 간에 읽기 전용 연결을 로드 밸런싱합니다. `replicaSet` 모드를 통해 연결하는 경우 클러스터 리더 엔드포인트가 클러스터 엔드포인트로 작동합니다. 즉, 연결 문자열에서 복제본 세트 파라미터는 `&replicaSet=rs0`입니다. 이 경우 기본에서 쓰기 작업을 수행할 수 있습니다. 그러나 `directConnection=true`를 지정하는 클러스터에 연결하는 경우 리더 엔드포인트에 대한 연결을 통해 쓰기 작업을 수행하려고 하면 오류가 발생합니다. Amazon DocumentDB 클러스터에는 정확히 하나의 리더 엔트포인트가 있습니다.

클러스터에 기본 인스턴스 하나만 있는 경우 리더 엔드포인트가 기본 인스턴스에 연결합니다. 복제본 인스턴스를 Amazon DocumentDB 클러스터에 추가할 때, 리더 엔드포인트는 복제본이 활성화된 후 새 복제본에 대한 읽기 전용 연결을 엽니다.

다음은 Amazon DocumentDB 클러스터에 대한 리더 엔드포인트의 예제입니다.

```
sample-cluster.cluster-ro-123456789012.us-east-1.docdb.amazonaws.com:27017
```

다음은 리더 엔드포인트를 사용하는 연결 문자열 예제입니다.

```
mongodb://username:password@sample-cluster.cluster-ro-123456789012.us-
east-1.docdb.amazonaws.com:27017
```

리더 엔드포인트는 읽기 요청이 아닌 읽기 전용 연결을 로드 밸런싱합니다. 일부 리더 엔드포인트 연결이 다른 연결보다 매우 많이 사용되는 경우 읽기 요청이 클러스터의 인스턴스 간에 균등하게 밸런싱되지 않을 수 있습니다. 클러스터 엔드포인트에 복제본 세트로 연결하고 secondaryPreferred 읽기 기본 설정 옵션을 사용하여 요청을 분산하는 것이 좋습니다.

클러스터 엔드포인트 찾기에 대한 자세한 내용은 클러스터 엔드포인트 찾기 단원을 참조하십시오.

인스턴스 엔드포인트

인스턴스 엔드포인트는 클러스터에 있는 특정 인스턴스에 연결됩니다. 읽기 및 쓰기 작업에 현재 기본 인스턴스의 인스턴스 엔드포인트를 사용할 수 있습니다.니다. 하지만 읽기 전용 복제본의 인스턴스 엔드포인트에 대한 쓰기 작업을 수행하려는 시도로 인해 오류가 발생합니다. Amazon DocumentDB 클러스터에는 활성 인스턴스당 하나의 인스턴스 엔드포인트가 있습니다.

클러스터 엔드포인트 또는 리더 엔드포인트가 부적합한 시나리오에서는 인스턴스 엔드포인트가 특정 인스턴스에 대한 연결을 직접 제어합니다. 예제 사용 사례에서는 주기적인 읽기 전용 분석 워크로드를 프로비저닝합니다. 일반 복제본 인스턴스보다 큰 인스턴스를 프로비저닝하고, 인스턴스 엔드포인트를 사용하여 더 큰 새 인스턴스에 직접 연결하며, 분석 쿼리를 실행한 후 인스턴스를 종료할 수 있습니다. 인스턴스 엔드포인트를 사용하면 분석 트래픽이 다른 클러스터 인스턴스에 영향을 주지 않습니다.

다음은 Amazon DocumentDB 클러스터의 단일 인스턴스에 대한 인스턴스 엔드포인트의 예입니다.

```
sample-instance.123456789012.us-east-1.docdb.amazonaws.com:27017
```

다음은 이 인스턴스 엔드포인트를 사용하는 연결 문자열 예제입니다.

```
mongodb://username:password@sample-instance.123456789012.us-
east-1.docdb.amazonaws.com:27017
```

> **Note**
>
> 장애 조치 이벤트로 인해 기본 또는 복제본으로서 인스턴스의 역할이 변경될 수 있습니다. 애플리케이션에서 특정 인스턴스 엔드포인트가 기본 인스턴스라고 가정해서는 안 됩니다. 프로덕션 애플리케이션의 경우 인스턴스 엔드포인트에 연결하지 않는 것이 좋습니다. 대신 클러스터 엔드포인트를 사용하고 복제본 세트 모드(Amazon DocumentDB에 복제본 세트로 연결 참조)에서 클러스터에 연결하는 것이 좋습니다. 인스턴스 장애 조치 우선 순위의 고급 제어에 대한 자세한 내용은 Amazon DocumentDB 클러스터 내결함성에 대한 이해 단원을 참조하십시오.

클러스터 엔드포인트 찾기에 대한 자세한 내용은 인스턴스의 엔드포인트 찾기 단원을 참조하십시오.

복제본 세트 모드

복제본 세트 이름 `rs0`을 지정하여 복제본 세트 모드에서 Amazon DocumentDB 클러스터 엔드포인트에 연결할 수 있습니다. 복제본 세트 모드에서 연결하면 Read Concern(읽기 문제), Write Concern(쓰

기 문제) 및 Read Preference(읽기 기본 설정) 옵션을 지정할 수 있습니다. 자세한 내용은 읽기 정합성 단원을 참조하십시오.

다음은 복제본 세트 모드에서 연결하는 예제 연결 문자열입니다.

```
mongodb://username:password@sample-cluster.cluster-123456789012.us-
east-1.docdb.amazonaws.com:27017/?replicaSet=rs0
```

복제본 세트 모드로 연결하면 Amazon DocumentDB 클러스터가 복제본 세트로 드라이버와 클라이언트에 나타납니다. Amazon DocumentDB 클러스터에서 추가 및 제거된 인스턴스는 복제본 집합 구성에 자동으로 반영됩니다.

각 Amazon DocumentDB 클러스터는 기본 이름이 `rs0`인 단일 복제본 세트로 구성됩니다. 복제본 세트 이름은 수정할 수 없습니다.

일반적인 용도에서는 복제본 세트 모드에서 클러스터 엔드포인트에 연결하는 것이 좋은 방법입니다.

> **Note**
>
> Amazon DocumentDB 클러스터의 모든 인스턴스는 동일한 TCP 포트에서 연결을 수신합니다.

TLS 지원

전송 계층 보안(TLS)을 사용하여 Amazon DocumentDB에 연결하는 방법에 대한 자세한 내용은 전송 중 데이터 암호화을 참조하십시오.

Amazon DocumentDB 스토리지

Amazon DocumentDB 데이터는 솔리드 스테이트 드라이브(SSD)를 사용하는 단일 가상 볼륨인 클러스터 볼륨에 저장됩니다. 클러스터 볼륨은 데이터 복사본 6개로 구성되며, 데이터 복사본은 단일 AWS 리전에서 여러 가용 영역에 걸쳐 자동으로 복제됩니다. 이 복제를 통해 데이터의 내구성을 높이고 데이터 손실 가능성을 줄일 수 있습니다. 또한 다른 가용 영역에 데이터 복사본이 이미 있어 장애 조치가 이루어지는 동안 클러스터 가용성이 높아집니다. 이러한 복사본은 계속해서 Amazon DocumentDB 클러스터의 인스턴스에 데이터 요청을 제공할 수 있습니다.

데이터 스토리지 요금이 청구되는 방법

Amazon DocumentDB는 데이터의 양이 증가함에 따라 클러스터 볼륨의 크기가 자동으로 증가합니다. Amazon DocumentDB 클러스터 볼륨은 최대 128TiB까지 증가할 수 있지만, 사용자는

Amazon DocumentDB 클러스터 볼륨에서 사용하는 공간에 대해서만 비용이 청구됩니다. Amazon DocumentDB 4.0부터는 컬렉션이나 인덱스를 삭제하는 등의 방법으로 데이터가 제거되면 비슷한 양만큼 할당된 전체 공간이 감소합니다. 따라서 더 이상 필요하지 않은 컬렉션, 인덱스 및 데이터베이스를 삭제하여 스토리지 요금을 절감할 수 있습니다. Amazon DocumentDB 버전 3.6에서 클러스터 볼륨은 데이터를 제거할 때 확보된 공간을 재사용할 수 있지만 볼륨 자체의 크기는 감소하지 않습니다. 따라서 버전 3.6에서는 여유 공간이 재사용되더라도 컬렉션 또는 인덱스를 삭제할 때 스토리지가 변경되지 않을 수 있습니다.

> **Note**
>
> Amazon DocumentDB 3.6의 스토리지 비용은 스토리지 "하이 워터 마크"(Amazon DocumentDB 클러스터에 할당된 최대 양)를 기반으로 합니다. 대량의 임시 정보를 생성하거나 불필요한 오래된 데이터를 제거하기 전에 대량의 새 데이터를 로드하는 ETL 관행을 피함으로써 비용을 관리할 수 있습니다. Amazon DocumentDB 클러스터에서 데이터를 제거하면 많은 양의 공간이 할당되지만 사용되지 않는 경우 하이 워터 마크를 재설정하려면 논리적 데이터 덤프를 수행하고 `mongodump` 또는 `mongorestore`와 같은 도구를 사용하여 새 클러스터로 복원해야 합니다. 스냅샷을 생성하고 복원하면 기본 스토리지의 물리적 레이아웃이 복원된 스냅샷에서 동일하게 유지되기 때문에 할당된 스토리지가 감소하지 않습니다.

> **Note**
>
> `mongodump`와 `mongorestore` 같은 유틸리티를 사용하면 스토리지 볼륨에 읽고 쓰는 데이터의 크기에 따라 I/O 요금이 발생합니다.

Amazon DocumentDB 데이터 스토리지 및 I/O 요금에 대한 자세한 내용은 Amazon DocumentDB(MongoDB와 호환 가능) 요금 및 요금 FAQ를 참조하십시오.

Amazon DocumentDB 복제

Amazon DocumentDB 클러스터에서 각 복제본 인스턴스는 독립적 엔트포인트를 보여줍니다. 이 복제본 엔드포인트는 클러스터 볼륨의 데이터에 대한 읽기 전용 액세스를 제공하며 복제된 여러 인스턴스에서 데이터의 읽기 워크로드를 확장하도록 해줍니다. 또한 Amazon DocumentDB 클러스터의 데이터 읽기 성능을 개선하고 데이터 가용성을 높이는 데도 도움이 됩니다. Amazon DocumentDB 복제본은 장애 조치 대상이기도 하며 Amazon DocumentDB 클러스터의 기본 인스턴스에 장애가 발생할 경우 신속하게 승격됩니다.

Amazon DocumentDB 신뢰성

Amazon DocumentDB는 안정성, 내구성 및 내결함성을 고려하여 설계되었습니다. (가용성을 높이려면 Amazon DocumentDB 클러스터가 서로 다른 가용 영역에 여러 개의 복제본 인스턴스가 있도록 구성해야 합니다.) Amazon DocumentDB에는 신뢰할 수 있는 데이터베이스 솔루션으로 만들어 주는 몇 가지 자동 기능이 포함되어 있습니다.

스토리지 자동 복구

Amazon DocumentDB는 데이터의 여러 복사본을 3개의 가용 영역에 유지하므로 스토리지 장애로 인해 데이터가 손실될 가능성을 크게 줄입니다. Amazon DocumentDB는 클러스터 볼륨에서 장애를 자동으로 감지합니다. 클러스터 볼륨의 세그먼트에 장애가 발생하면 Amazon DocumentDB는 즉시 세그먼트를 복구합니다. 이때 클러스터 볼륨을 구성하는 나머지 볼륨의 데이터를 사용하여 복구된 세그먼트의 데이터도 이용 가능합니다. 따라서 Amazon DocumentDB는 데이터 손실을 방지하고 인스턴스 장애를 복구하기 위해 시점 복원을 수행할 필요성을 줄입니다.

유지 가능한 캐시 워밍

Amazon DocumentDB는 페이지 캐시가 데이터베이스와 독립적으로 생존할 수 있도록 데이터베이스와 별도의 프로세스로 자신의 페이지 캐시를 관리합니다. 데이터베이스 장애가 발생한 경우에는 페이지 캐시가 메모리에 남습니다. 이렇게 하면 데이터베이스를 다시 시작할 때 버퍼 풀이 가장 최근의 상태로 워밍업됩니다.

충돌 복구

Amazon DocumentDB는 거의 순간적으로 충돌을 복구하고 애플리케이션 데이터를 계속해서 제공하도록 설계되었습니다. Amazon DocumentDB는 병렬 스레드에서 비동기식으로 충돌 복구를 수행하므로, 충돌 후 거의 즉시 데이터베이스가 열리고 사용할 수 있습니다.

리소스 거버넌스

Amazon DocumentDB는 상태 확인과 같은 서비스의 중요한 프로세스를 실행하는 데 필요한 리소스를 보호합니다. 이를 위해 인스턴스에 높은 메모리 압력이 발생하는 경우 Amazon DocumentDB는 요청을 제한합니다. 따라서 일부 작업은 메모리 사용량이 줄어들 때까지 대기열에 들어갈 수 있습니다. 메모리 부족 현상이 계속되면 대기 중인 작업 시간이 초과될 수 있습니다. 다음의 `LowMemThrottleQueueDepth`, `LowMemThrottleMaxQueueDepth`, `LowMemNumOperationsThrottled`, `LowMemNumOperationsTimedOut` CloudWatch 측정치를 사용하여 메모리 부족으로 인한 서비스 제한 작업 여부를 모니터링할 수 있습니다. 자세한 내용은 CloudWatch를 사용하는 Amazon DocumentDB 모니터링을 참조하세요. LowMem CloudWatch 지표

로 인해 인스턴스의 메모리 부족 현상이 지속되면 인스턴스를 확장하여 워크로드에 사용할 추가 메모리를 제공하는 것이 좋습니다.

읽기 기본 설정 옵션

Amazon DocumentDB는 클라우드 네이티브 공유 스토리지 서비스를 사용하여 세 개의 가용 영역에 걸쳐 데이터를 6회 복제하여 높은 수준의 내구성을 제공합니다. Amazon DocumentDB는 내구성을 확보하기 위해 데이터를 여러 인스턴스에 복제하지 않습니다. 인스턴스가 1개이건 15개이건 클러스터 데이터의 내구성이 유지됩니다.

주제

- 쓰기 내구성
- 읽기 격리
- 읽기 정합성
- 높은 가용성
- 읽기 확장

쓰기 내구성

Amazon DocumentDB는 고유한, 배포된, 내결함성, 자가 복구 스토리지 시스템을 사용합니다. 이 시스템은 3개의 AWS 가용 영역에 6개의 데이터 복사본(V=6)을 복제하여 고가용성과 내구성을 제공합니다. Amazon DocumentDB는 데이터를 작성할 때 클라이언트에게 쓰기를 승인하기 전에 모든 쓰기가 다수의 노드에 지속적으로 기록되도록 보장합니다. 3노드 MongoDB 복제본 세트를 실행하는 경우 쓰기 문제 `{w:3, j:true}`을 사용하면 Amazon DocumentDB와 비교할 때 최상의 구성을 얻을 수 있습니다.

Amazon DocumentDB 클러스터에 대한 쓰기는 클러스터의 라이터 인스턴스에서 처리해야 합니다. 리더에 쓰기를 시도하면 오류가 발생합니다. Amazon DocumentDB 기본 인스턴스에서 승인된 쓰기는 내구성이 뛰어나며 롤백할 수 없습니다. Amazon DocumentDB는 기본적으로 내구성이 뛰어나며 내구성이 없는 쓰기 옵션을 지원하지 않습니다. 사용자가 내구성 수준(즉, 쓰기 문제)을 수정할 수 없습니다. Amazon DocumentDB는 w=아무거나를 무시하며 사실상 w: 3 및 j: true입니다. 줄일 수는 없습니다.

Amazon DocumentDB 아키텍처에서 스토리지와 컴퓨팅이 분리되어 있기 때문에 단일 인스턴스를 가진 클러스터는 내구성이 매우 높습니다. 내구성은 스토리지 계층에서 처리됩니다. 따라서 단일 인스턴스와 3개의 인스턴스가 있는 하나의 Amazon DocumentDB 클러스터는 동일한 수준의 내구성을 달성합니다. 데이터의 뛰어난 내구성을 유지하면서 클러스터를 특정 사용 사례로 구성할 수 있습니다.

Amazon DocumentDB 클러스터에 대한 쓰기는 단일 문서 내에서 원자성을 갖습니다.

Amazon DocumentDB는 `wtimeout` 옵션을 지원하지 않으며 값이 지정된 경우 오류를 반환하지 않습니다. 기본 Amazon DocumentDB 인스턴스에 대한 쓰기는 무기한 차단되지 않도록 보장됩니다.

읽기 격리

Amazon DocumentDB 인스턴스에서 읽은 데이터는 쿼리가 시작되기 전에만 지속되는 데이터를 반환합니다. 읽기는 쿼리 실행이 시작된 후 수정한 데이터를 반환하지 않으며 어떠한 경우에도 더티 읽기는 불가능합니다.

읽기 정합성

Amazon DocumentDB 클러스터에서 읽은 데이터는 내구성이 뛰어나며 롤백되지 않습니다. 요청 또는 연결에 대한 읽기 환경설정을 지정하여 Amazon DocumentDB 읽기에 대한 읽기 일관성을 수정할 수 있습니다. Amazon DocumentDB는 내구성이 떨어지는 읽기 옵션을 지원하지 않습니다.

Amazon DocumentDB 클러스터의 기본 인스턴스로부터의 읽기는 정상적인 동작 조건에서 강력하게 일관되고 쓰기 후 읽기 일관성을 갖습니다. 쓰기와 그 다음 읽기 사이에서 장애 조치 이벤트가 발생할 경우 시스템이 일시적으로 강력히 일관되지 않은 읽기를 반환할 수 있습니다. 읽기 전용 복제본에서 모든 읽기는 최종적으로 일관되며 같은 순서로 데이터를 반환할 뿐 아니라 복제 지연 시간이 종종 100ms 미만입니다.

Amazon DocumentDB 읽기 기본 설정

Amazon DocumentDB는 복제본 설정 모드에서 클러스터 엔드포인트에서 데이터를 읽을 때에만 읽기 환경설정 옵션을 설정할 수 있도록 지원합니다. 읽기 환경설정 옵션을 설정하면 MongoDB 클라이언트 또는 드라이버가 읽기 요청을 Amazon DocumentDB 클러스터의 인스턴스로 라우트하는 방법에 영향을 미칩니다. 특정 쿼리에 대해 또는 MongoDB 드라이버의 일반 옵션으로 읽기 기본 설정 옵션을 설정할 수 있습니다. 읽기 기본 설정 옵션을 설정하는 방법은 클라이언트 또는 드라이버 설명서를 참조하십시오.

클라이언트 또는 드라이버가 복제본 설정 모드에서 Amazon DocumentDB 클러스터 엔드포인트에 연결하지 않으면 읽기 환경설정을 지정한 결과가 정의되지 않습니다.

Amazon DocumentDB는 태그 세트를 읽기 환경설정으로 설정하는 것을 지원하지 않습니다.

지원되는 읽기 기본 설정 옵션

- **`primary`**—`primary` 읽기 기본 설정을 지정하면 모든 읽기가 클러스터의 기본 인스턴스로 라우팅되도록 할 수 있습니다. 기본 인스턴스를 사용할 수 없으면 읽기 작업이 실패합니다. `primary` 읽기 기본 설정은 쓰기 후 읽기 일관성을 생성하며, 고가용성과 읽기 확장성보다 쓰기 후 읽기 일관성이 우선시되는 사용 사례에 적합합니다.

다음 예제에서는 primary 읽기 기본 설정을 지정합니다.

```
db.example.find().readPref('primary')
```

- **primaryPreferred**—일반 작동 시 기본 인스턴스에 대한 primaryPreferred 읽기 기본 설정 경로를 지정합니다. 기본 장애 조치가 있을 경우 클라이언트가 요청을 복제본으로 라우팅합니다. primaryPreferred 읽기 기본 설정은 일반적인 작업 중 쓰기 후 읽기 일관성을 생성하고, 장애 조치 이벤트 중 최종적 일관된 읽기를 생성합니다. primaryPreferred 읽기 기본 설정은 읽기 확장성보다 쓰기 후 읽기 일관성이 우선시되지만 고가용성도 필요한 사용 사례에 적합합니다.

  다음 예제에서는 primaryPreferred 읽기 기본 설정을 지정합니다.

```
db.example.find().readPref('primaryPreferred')
```

- **secondary**— secondary 읽기 기본 설정을 지정하면 읽기가 기본 인스턴스가 아닌 복제본으로만 라우팅됩니다. 클러스터에 복제본 인스턴스가 없으면 읽기 요청이 실패합니다. secondary 읽기 기본 설정은 최종적 일관된 읽기를 생성하며 고가용성과 쓰기 후 읽기 일관성보다 기본 인스턴스 쓰기 처리량이 우선시되는 사용 사례에 적합합니다.

  다음 예제에서는 secondary 읽기 기본 설정을 지정합니다.

```
db.example.find().readPref('secondary')
```

- **secondaryPreferred**—secondaryPreferred 읽기 환경설정을 지정하면 읽기는 하나 이상의 복제본이 활성화될 때 읽기 복제본으로 라우팅됩니다. 클러스터에 활성 복제본 인스턴스가 없으면 읽기 요청이 기본 인스턴스로 라우팅됩니다. secondaryPreferred 읽기 기본 설정은 읽기 전용 복제본에서 읽기를 처리할 때 최종적 일관된 읽기를 생성합니다. 장애 조치 이벤트를 제외하고 기본 인스턴스에서 읽기를 처리할 때 쓰기 후 읽기 일관성이 생깁니다. secondaryPreferred 읽기 기본 설정은 쓰기 후 읽기 일관성보다 읽기 확장성 및 고가용성이 우선시되는 사용 사례에 적합합니다.

  다음 예제에서는 secondaryPreferred 읽기 기본 설정을 지정합니다.

```
db.example.find().readPref('secondaryPreferred')
```

- **`nearest`**—클라이언트와 Amazon DocumentDB 클러스터의 모든 인스턴스 간에 측정된 지연 시간만을 기준으로 `nearest` 읽기 기본 설정 경로를 지정합니다. `nearest` 읽기 기본 설정은 읽기 전용 복제본에서 읽기를 처리할 때 최종적 일관된 읽기를 생성합니다. 장애 조치 이벤트를 제외하고 기본 인스턴스에서 읽기를 처리할 때 쓰기 후 읽기 일관성이 생깁니다. `nearest` 읽기 기본 설정은 쓰기 후 읽기 일관성 및 읽기 확장성보다 가장 낮은 읽기 지연 시간 및 고가용성 달성이 우선시되는 사용 사례에 적합합니다.

  다음 예제에서는 `nearest` 읽기 기본 설정을 지정합니다.

```
db.example.find().readPref('nearest')
```

높은 가용성

Amazon DocumentDB는 복제본을 기본 인스턴스의 장애 조치 대상으로 사용하여 가용성이 높은 클러스터 구성을 지원합니다. 기본 인스턴스가 실패하면 Amazon DocumentDB 복제본이 새 기본 인스턴스로 승격되며, 기본 인스턴스에 대한 읽기 및 쓰기 요청이 예외로 실패하는 짧은 중단이 발생합니다.

Amazon DocumentDB 클러스터에 복제본이 없는 경우, 기본 인스턴스는 실패 중에 다시 작성됩니다. 그러나 Amazon DocumentDB 복제본을 승격하는 것은 기본 인스턴스를 다시 만드는 것보다 훨씬 빠릅니다. 따라서 장애 조치 대상으로 하나 이상의 Amazon DocumentDB 복제본을 작성하는 것이 좋습니다.

장애 조치 대상으로 사용할 복제본은 기본 인스턴스와 동일한 인스턴스 클래스에 속해야 하며 기본 인스턴스와는 다른 가용 영역에 프로비저닝되어야 합니다. 장애 조치 대상으로 사용할 복제본을 제어할 수 있습니다.니다. 고가용성을 위해 Amazon DocumentDB를 구성하는 모범 사례는 Amazon DocumentDB 클러스터 내결함성에 대한 이해을 참조하십시오.

읽기 확장

Amazon DocumentDB 복제본은 읽기 규모 조정에 적합합니다. 클러스터 볼륨에서 이루어지는 읽기 작업 전용으로 사용됩니다. 즉 복제본이 쓰기는 처리하지 않습니다. 데이터 복제는 클러스터 볼륨 내에서 일어나며 인스턴스 간에는 일어나지 않습니다. 따라서 각 복제본의 리소스는 쿼리 처리에만 사용되지 데이터의 복제 및 쓰기에 사용되지 않습니다.

애플리케이션에 읽기 용량이 더 필요하면 클러스터에 신속하게 복제본을 추가할 수 있습니다(보통 10분 미만). 읽기 용량 요구 사항이 줄어들면 불필요한 복제본을 제거할 수 있습니다. Amazon DocumentDB 복제본에서는 필요한 읽기 용량에 대해서만 비용을 지불합니다.

Amazon DocumentDB는 읽기 환경설정 옵션을 사용하여 클라이언트 측 읽기 스케일링을 지원합니다. 자세한 내용은 Amazon DocumentDB 읽기 기본 설정을 참조하세요.

TTL 삭제

백그라운드 프로세스를 통해 생성된 TTL 인덱스 영역에서의 삭제는 특정 기간 내에 삭제된다고 보장할 수 없으며 가능한 한 빠른 시간 내에 수행됩니다. 인스턴스 크기, 인스턴스 리소스 사용률, 문서 크기 및 전체 처리량과 같은 요소가 TTL 삭제 소요 시간에 영향을 줄 수 있습니다.

TTL 모니터가 문서를 삭제하면 삭제할 때마다 IO 비용이 발생하므로 청구 금액이 증가합니다. 처리량과 TTL 삭제율이 증가하면 IO 사용량 증가로 인해 청구서가 증가할 것으로 예상해야 합니다.

기존 컬렉션에 TTL 인덱스를 생성할 때는 인덱스를 생성하기 전에 만료된 문서를 모두 삭제해야 합니다. 현재 TTL 구현은 컬렉션에서 소량의 문서를 삭제하는 데 최적화되어 있는데, 이는 컬렉션에서 처음부터 TTL을 활성화한 경우 일반적으로 발생하며, 한 번에 많은 문서를 삭제해야 하는 경우 IOPS가 필요 이상으로 높아질 수 있습니다.

문서를 삭제하기 위해 TTL 인덱스를 작성하지 않으려면, 대신 시간을 기준으로 문서를 컬렉션으로 분할하고, 문서가 더 이상 필요하지 않을 때 해당 컬렉션을 삭제할 수 있습니다. 예를 들어, 일주일에 한 개의 컬렉션을 만들어 IO 비용을 들이지 않고도 삭제할 수 있습니다. 이 방법은 TTL 인덱스를 사용하는 것보다 훨씬 더 비용 효율적일 수 있습니다.

청구 가능 리소스

청구 가능한 Amazon DocumentDB 리소스 식별

Amazon DocumentDB는 완전 관리형 데이터베이스 서비스로서 인스턴스, 스토리지, I/O, 백업 및 데이터 전송 비용을 청구합니다. 자세한 내용은 Amazon DocumentDB(MongoDB와 호환됨) 요금을 참조하십시오.

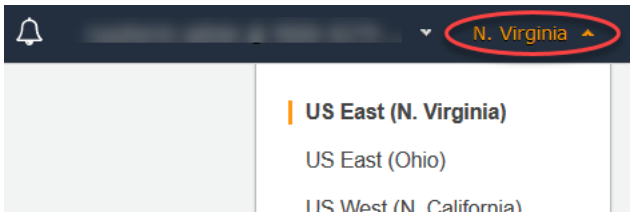
계정에서 청구 가능한 리소스를 검색하고 리소스를 잠재적으로 삭제하려면 AWS Management Console 또는를 사용할 수 있습니다 AWS CLI.

사용 AWS Management Console

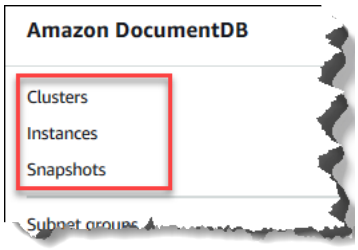
를 사용하면 지정된에 대해 프로비저닝한 Amazon DocumentDB 클러스터, 인스턴스 및 스냅샷을 검색할 AWS Management Console수 있습니다 AWS 리전.

클러스터, 인스턴스 및 스냅샷을 검색하려면

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 기본 리전이 아닌 리전에서 청구 가능한 리소스를 검색하려면 화면의 오른쪽 상단 모서리에서 검색할 AWS 리전 을 선택합니다.



3. 탐색 창에서 검색하려는 청구 대상 리소스의 유형을 선택합니다. 클러스터, 인스턴스 또는 스냅샷.



4. 해당 리전에 프로비저닝된 모든 클러스터, 인스턴스 또는 스냅샷이 오른쪽 창에 나열됩니다. 클러스터, 인스턴스 및 스냅샷에 요금이 청구됩니다.

사용 AWS CLI

를 사용하면 지정된에 프로비저닝한 Amazon DocumentDB 클러스터, 인스턴스 및 스냅샷을 검색할 AWS CLI수 있습니다 AWS 리전.

클러스터 및 인스턴스를 검색하려면

다음 코드는 지정된 리전의 모든 클러스터 및 인스턴스를 나열합니다. 기본 리전에서 클러스터 및 인스턴스를 검색하려면 `--region` 파라미터를 생략할 수 있습니다.

Example

Linux, macOS, Unix의 경우:

```
aws docdb describe-db-clusters \
    --region us-east-1 \
    --query 'DBClusters[?Engine==`docdb`]' | \
```

```
       grep -e "DBClusterIdentifier" -e "DBInstanceIdentifier"
```

Windows의 경우:

```
aws docdb describe-db-clusters ^
    --region us-east-1 ^
    --query 'DBClusters[?Engine==`docdb`]' | ^
       grep -e "DBClusterIdentifier" -e "DBInstanceIdentifier"
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
"DBClusterIdentifier": "docdb-2019-01-09-23-55-38",
        "DBInstanceIdentifier": "docdb-2019-01-09-23-55-38",
        "DBInstanceIdentifier": "docdb-2019-01-09-23-55-382",
"DBClusterIdentifier": "sample-cluster",
"DBClusterIdentifier": "sample-cluster2",
```

스냅샷을 검색하려면

다음 코드는 지정된 리전의 모든 스냅샷을 나열합니다. 기본 리전에서 스냅샷을 검색하려면 `--region` 파라미터를 생략할 수 있습니다.

Linux, macOS, Unix의 경우:

```
aws docdb describe-db-cluster-snapshots \
  --region us-east-1 \
  --query 'DBClusterSnapshots[?Engine==`docdb`].
[DBClusterSnapshotIdentifier,SnapshotType]'
```

Windows의 경우:

```
aws docdb describe-db-cluster-snapshots ^
  --region us-east-1 ^
  --query 'DBClusterSnapshots[?Engine==`docdb`].
[DBClusterSnapshotIdentifier,SnapshotType]'
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
[
    [
```

```
        "rds:docdb-2019-01-09-23-55-38-2019-02-13-00-06",
        "automated"
    ],
    [
        "test-snap",
        "manual"
    ]
]
```

manual 스냅샷만 삭제하면 됩니다. Automated 스냅샷은 클러스터를 삭제할 때 함께 삭제됩니다.

불필요한 청구 대상 리소스 삭제

클러스터를 삭제하려면 먼저 해당 클러스터의 인스턴스를 모두 삭제해야 합니다.

- 인스턴스를 삭제하려면 Amazon DocumentDB 인스턴스 삭제 단원을 참조하십시오.

> **Important**
>
> 클러스터에서 인스턴스를 삭제하더라도 해당 클러스터와 연결된 스토리지 및 백업에 대해서는 계속 요금이 발생합니다. 모든 요금을 중단하려면 클러스터와 수동 스냅샷도 삭제해야 합니다.

- 클러스터를 삭제하려면 Amazon DocumentDB 클러스터 삭제 단원을 참조하십시오.
- 수동 스냅샷 삭제는 클러스터 스냅샷 삭제 단원을 참조하십시오.

문서 데이터베이스란?

일부 개발자는 정규화된 행과 열의 측면에서 데이터 모델을 생각하지 않습니다. 일반적으로, 애플리케이션 계층 내에서 데이터가 JSON 문서로서 나타나게 되는데, 그 이유는 개발자들이 자신의 데이터 모델을 문서로 생각하는 것이 보다 직관적이기 때문입니다.

애플리케이션 코드에서 사용하는 것과 동일한 문서 모델 형식을 이용함으로써 데이터베이스에서 데이터를 지속해서 사용할 수 있으므로 문서 데이터베이스의 대중성은 높아져 왔습니다. 문서 데이터베이스는 유연하고 신속한 개발을 위해 강력하고 직관적인 API를 제공합니다.

주제

- 문서 데이터베이스 사용 사례
- 문서 이해

- 문서 작업

문서 데이터베이스 사용 사례

사용 사례는 문서 데이터베이스 또는 데이터를 관리하기 위한 일부 다른 유형의 데이터베이스가 필요한지 여부를 결정합니다. 문서 데이터베이스는 빠르고 반복적인 개발을 위해 유연한 스키마가 필요한 워크로드에 유용합니다. 다음은 문서 데이터베이스가 중요한 이점을 제공하는 일부 사용 사례의 예입니다.

주제

- 사용자 프로필
- 실시간 빅 데이터
- 콘텐츠 관리

사용자 프로필

문서 데이터베이스에는 유연한 스키마가 있으므로 속성과 데이터 값이 다른 문서를 저장할 수 있습니다. 문서 데이터베이스는 서로 다른 사용자가 다양한 유형의 정보를 제공하는 온라인 프로필에 대한 실용적인 솔루션입니다. 문서 데이터베이스에서는 각 사용자에 고유한 속성만 저장할 수 있어 각 사용자의 프로필을 효율적으로 저장할 수 있습니다.

사용자가 프로필에서 정보를 추가하거나 제거하도록 선택했다고 가정합니다. 이 경우 해당 문서를 최근에 추가된 속성과 데이터가 포함되거나 새로 생략된 속성과 데이터가 생략된 업데이트된 버전으로 쉽게 바꿀 수 있습니다. 문서 데이터베이스는 이 수준의 특성 및 유동성을 쉽게 관리합니다.

실시간 빅 데이터

과거에는 운영 데이터베이스와 분석 데이터베이스가 각각 운영 환경과 업무/보고 환경 등 서로 다른 환경에서 유지 관리되어 운영 데이터에서 정보를 추출하는 데 어려움이 있었습니다. 실시간으로 운영 정보를 추출할 수 있다는 것은 경쟁이 치열한 비즈니스 환경에서 중요합니다. 문서 데이터베이스를 사용하면 비즈니스에서는 모든 소스에서 운영 데이터를 저장하고 관리할 수 있으며 동시에 분석을 위해 데이터를 선택한 BI 엔진에 전달할 수 있습니다. 두 개의 환경이 필요하지 않습니다.

콘텐츠 관리

콘텐츠를 효과적으로 관리하려면 다양한 소스에서 콘텐츠를 수집 및 집계한 다음 고객에게 제공할 수 있어야 합니다. 문서 데이터베이스는 유연한 스키마로 인해 모든 유형의 데이터 수집과 저장에 적합합

니다. 이 데이터베이스를 사용하여 이미지, 설명, 비디오 등 사용자 생성 콘텐츠를 비롯한 새 유형의 콘텐츠를 생성하고 통합할 수 있습니다.

문서 이해

문서 데이터베이스는 관계형 데이터베이스에서처럼 각각 고유하고 고정된 구조를 가진 여러 테이블의 데이터를 정규화하는 대신 반정형 데이터를 문서로 저장하는 데 사용됩니다. 문서 데이터베이스에 저장된 문서는 중첩된 키-값 페어를 사용하여 문서의 구조 또는 스키마를 제공합니다. 그러나, 동일한 문서 데이터베이스에 다른 유형의 문서를 저장할 수 있으므로 다른 형식의 유사한 데이터를 처리하기 위한 요구 사항을 충족할 수 있습니다. 예를 들어, 각 문서는 자체적으로 설명되므로 문서 데이터베이스의 예제 문서 주제에 설명된 온라인 상점을 위한 JSON 인코딩 문서를 동일한 문서 데이터베이스에 저장할 수 있습니다.

주제

- SQL 대 비관계형 용어
- 단순 문서
- 내장 문서
- 문서 데이터베이스의 예제 문서
- 문서 데이터베이스에서 정규화 이해

SQL 대 비관계형 용어

다음 표에서는 문서 데이터베이스(MongoDB)에서 사용하는 용어를 SQL 데이터베이스에서 사용하는 용어와 비교합니다.

SQL	MongoDB
표	수집
열	문서
열	필드
프라이머리 키	ObjectId
인덱스	인덱스
보기	보기

SQL	MongoDB
중첩된 테이블 또는 객체	포함 문서
Array	Array

단순 문서

문서 데이터베이스의 모든 문서는 자체적으로 설명되어 있습니다. 이 설명서에서는 다른 인코딩 수단을 사용할 수 있는 경우에도 JSON과 유사한 형식의 문서를 사용합니다.

단순 문서에는 문서 내에서 모두 동일한 수준인 하나 이상의 필드가 있습니다. 다음 예에서 `SSN`, `LName`, `FName`, `DOB`, `Street`, `City`, `State-Province`, `PostalCode` 및 `Country` 필드는 문서 내에서 모두 형제 요소입니다.

```
{
   "SSN": "123-45-6789",
   "LName": "Rivera",
   "FName": "Martha",
   "DOB": "1992-11-16",
   "Street": "125 Main St.",
   "City": "Anytown",
   "State-Province": "WA",
   "PostalCode": "98117",
   "Country": "USA"
}
```

단순 문서에서 정보를 구성할 때 각 필드는 개별적으로 관리됩니다. 개인의 주소를 가져오려면 `Street`, `City`, `State-Province`, `PostalCode` 및 `Country`를 개별 데이터 항목으로 가져와야 합니다.

내장 문서

복잡한 문서는 문서 내에 내장 문서를 생성하여 데이터를 구성합니다. 내장 문서는 데이터 그룹화 및 개별 데이터 항목 중 제공된 사례에서 더 효율적인 방식으로 관리하는 데 도움이 됩니다. 이전 예제를 사용하면 기본 문서에 `Address` 문서를 포함할 수 있습니다. 이렇게 하면 다음과 같은 문서 구조가 발생합니다.

```
{
```

```
    "SSN": "123-45-6789",
    "LName": "Rivera",
    "FName": "Martha",
    "DOB": "1992-11-16",
    "Address":
    {
        "Street": "125 Main St.",
        "City": "Anytown",
        "State-Province": "WA",
        "PostalCode": "98117",
        "Country": "USA"
    }
}
```

이제 개별 필드("SSN":), 내장 문서("Address":) 또는 내장 문서의 멤버("Address": {"Street":})로 문서의 데이터에 액세스할 수 있습니다.

문서 데이터베이스의 예제 문서

이전에 설명한 대로 문서 데이터베이스의 각 문서는 자체적으로 설명되므로 문서 데이터베이스 내의 문서 구조는 서로 다를 수 있습니다. 다음의 두 문서(책 및 정기 간행물용)는 구조적으로 서로 다릅니다. 하지만 두 문서 모두 동일한 문서 데이터베이스에 있을 수 있습니다.

다음은 샘플 책 문서입니다.

```
{
    "_id" : "9876543210123",
    "Type": "book",
    "ISBN": "987-6-543-21012-3",
    "Author":
    {
        "LName":"Roe",
        "MI": "T",
        "FName": "Richard"
    },
    "Title": "Understanding Document Databases"
}
```

다음은 두 개의 기사가 있는 샘플 정기 간행물 문서입니다.

```
{
```

```
    "_id" : "0123456789012",
    "Publication": "Programming Today",
    "Issue":
    {
        "Volume": "14",
        "Number": "09"
    },
    "Articles" : [
        {
            "Title": "Is a Document Database Your Best Solution?",
            "Author":
            {
                "LName": "Major",
                "FName": "Mary"
            }
        },
        {
            "Title": "Databases for Online Solutions",
            "Author":
            {
                "LName": "Stiles",
                "FName": "John"
            }
        }
    ],
    "Type": "periodical"
}
```

이 두 문서의 구조를 비교합니다. 관계형 데이터베이스에서는 별도의 "정기 간행물" 및 "책" 테이블이나 "출판", "발행", "기사", "MI" 등 사용하지 않은 필드가 null 값인 단일 테이블이 필요합니다. 문서 데이터베이스는 각 문서가 자체 구조를 정의하는 반구조화이므로 이러한 두 문서는 null 필드가 없는 동일한 문서 데이터베이스에서 동시에 존재할 수 있습니다. 문서 데이터베이스는 희소 데이터를 처리하는 데 적합합니다.

문서 데이터베이스를 대상으로 개발하면 빠르고 반복적으로 개발할 수 있습니다. 이는 전체 모음에 대한 스키마를 변경하지 않고 문서의 데이터 구조를 동적으로 변경할 수 있기 때문입니다. 문서 데이터베이스는 신속한 개발 및 동적으로 변화하는 환경에 매우 적합합니다.

문서 데이터베이스에서 정규화 이해

문서 데이터베이스는 정규화되지 않습니다. 한 문서에서 발견된 데이터가 다른 문서에서 반복될 수 있습니다. 게다가 문서 간에 일부 데이터 차이가 있을 수 있습니다. 예를 들어, 온라인 상점에서 구매하고

구매에 대한 모든 세부 정보가 단일 문서에 저장되는 시나리오를 고려하십시오. 문서는 다음 JSON 문서와 같을 수 있습니다.

```
{
    "DateTime": "2018-08-15T12:13:10Z",
    "LName" : "Santos",
    "FName" : "Paul",
    "Cart" : [
        {
            "ItemId" : "9876543210123",
            "Description" : "Understanding Document Databases",
            "Price" : "29.95"
        },
        {
            "ItemId" : "0123456789012",
            "Description" : "Programming Today",
            "Issue": {
                "Volume": "14",
                "Number": "09"
            },
            "Price" : "8.95"
        },
        {
            "ItemId": "234567890-K",
            "Description": "Gel Pen (black)",
            "Price": "2.49"
        }
    ],
    "PaymentMethod" :
    {
        "Issuer" : "MasterCard",
        "Number" : "1234-5678-9012-3456"
    },
    "ShopperId" : "1234567890"
}
```

이러한 모든 정보는 거래 모음에서 문서로 저장됩니다. 나중에 품목 하나를 구매하지 않은 것을 알게 되었습니다. 따라서 다시 동일한 상점에 로그인하여 다시 구매했습니다. 이러한 구매도 거래 모음에서 다른 문서로 저장됩니다.

```
{
    "DateTime": "2018-08-15T14:49:00Z",
    "LName" : "Santos",
```

```
    "FName" : "Paul",
    "Cart" : [
        {
            "ItemId" : "2109876543210",
            "Description" : "Document Databases for Fun and Profit",
            "Price" : "45.95"
        }
    ],
    "PaymentMethod" :
    {
        "Issuer" : "Visa",
        "Number" : "0987-6543-2109-8765"
    },
    "ShopperId" : "1234567890"
}
```

이 두 문서, 즉 이름과 구매자 ID(같은 신용카드를 사용한 경우에는 신용카드 정보)가 중복된다는 점에 주목하세요. 그러나 스토리지가 저렴하고, 각 문서가 조인할 필요가 없는 단순 키-값 쿼리를 사용하여 빠르게 검색할 수 있는 단일 거래를 완전히 기록하기 때문에 괜찮습니다.

또한 신용 카드 정보라는 두 문서 간에도 명백한 차이가 있습니다. 각각의 구매에 대해 다른 신용카드를 사용한 것 같으므로 이러한 분명한 차이는 유일합니다. 각 문서는 문서화된 거래에 대해 정확합니다.

문서 작업

문서 데이터베이스로서 Amazon DocumentDB는 JSON 데이터를 쉽게 저장, 쿼리 및 인덱싱할 수 있습니다. Amazon DocumentDB에서 컬렉션은 모든 문서에 적용되는 단일 스키마가 없다는 점을 제외하면 관계형 데이터베이스의 테이블과 유사합니다. 모음을 통해 유사한 문서를 그룹화하여 모두 동일한 데이터베이스에 유지할 수 있으며, 구조가 동일하지 않아도 됩니다.

이전 단원의 예제 문서를 사용할 경우 `reading_material` 및 `office_supplies`에 대한 모음이 있을 수 있습니다. 소프트웨어에서는 문서가 속할 모음을 결정합니다.

다음 예제에서는 MongoDB API를 사용하여 문서를 추가, 쿼리, 업데이트 및 삭제하는 방법을 보여줍니다.

주제

- 문서 추가
- 문서 쿼리

- 문서 업데이트
- 문서 삭제

문서 추가

Amazon DocumentDB에서는 컬렉션에 문서를 처음 추가할 때 데이터베이스가 생성됩니다. 이 예제에서는 클러스터에 연결할 때 기본 데이터베이스인 `test` 데이터베이스에 `example`이라는 컬렉션을 만듭니다. 첫 번째 문서가 삽입될 때 컬렉션이 암시적으로 만들어지기 때문에 컬렉션 이름에 대한 오류 검사가 수행되지 않습니다. 따라서 컬렉션 이름에 오타(예: `example` 대신 `eexample`)가 생성되고 의도한 컬렉션이 아닌 `eexample` 컬렉션에 문서가 추가됩니다. 오류 확인은 애플리케이션을 통해 처리되어야 합니다.

다음 예제에서는 MongoDB API를 사용하여 문서를 추가합니다.

주제

- 단일 문서 추가
- 여러 문서 추가

단일 문서 추가

모음에 단일 문서를 추가하려면 모음에 추가할 문서에 `insertOne( {} )` 작업을 사용합니다.

```
db.example.insertOne(
    {
        "Item": "Ruler",
        "Colors": ["Red","Green","Blue","Clear","Yellow"],
        "Inventory": {
            "OnHand": 47,
            "MinOnHand": 40
        },
        "UnitPrice": 0.89
    }
)
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
    "acknowledged" : true,
    "insertedId" : ObjectId("5bedafbcf65ff161707de24f")
```

```
}
```

여러 문서 추가

모음에 여러 문서를 추가하려면 모음에 추가할 문서 목록에 `insertMany( [{},...,{}] )` 작업을 사용합니다. 이 특정 목록의 문서에 다른 스키마가 있는 경우에도 모두 동일한 모음에 추가할 수 있습니다.

```
db.example.insertMany(
    [
        {
            "Item": "Pen",
            "Colors": ["Red","Green","Blue","Black"],
            "Inventory": {
                "OnHand": 244,
                "MinOnHand": 72
            }
        },
        {
            "Item": "Poster Paint",
            "Colors": ["Red","Green","Blue","Black","White"],
            "Inventory": {
                "OnHand": 47,
                "MinOnHand": 50
            }
        },
        {
            "Item": "Spray Paint",
            "Colors": ["Black","Red","Green","Blue"],
            "Inventory": {
                "OnHand": 47,
                "MinOnHand": 50,
                "OrderQnty": 36
            }
        }
    ]
)
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
    "acknowledged" : true,
```

```
    "insertedIds" : [
            ObjectId("5bedb07941ca8d9198f5934c"),
            ObjectId("5bedb07941ca8d9198f5934d"),
            ObjectId("5bedb07941ca8d9198f5934e")
    ]
}
```

문서 쿼리

이따금 판매하는 물품을 고객이 보고 구매할 수 있도록 온라인 상점의 재고를 조회해야 할 수도 있습니다. 컬렉션을 쿼리하는 것은 컬렉션의 모든 문서에 대한 것이든 특정 기준을 충족하는 문서에만 대한 것이든 관계없이 상대적으로 간단합니다.

문서를 쿼리하려면 `find()` 작업을 사용합니다. `find()` 명령에는 반환할 문서 선택 시 사용할 기준을 정의하는 단일 문서 파라미터가 있습니다. `find()`의 출력은 줄 바꿈이 없이 한 줄로 된 텍스트 형식의 문서입니다. 쉽게 읽기 위해 출력 문서의 형식을 지정하려면 `find().pretty()`를 사용하십시오. 이 주제의 모든 예제는 `.pretty()`를 사용하여 출력 형식을 지정합니다.

앞선 두 가지 연습 `insertOne()` 및 `insertMany()`에서 `example` 컬렉션에 삽입한 4개의 문서를 사용합니다.

주제

- 컬렉션의 모든 문서 검색
- 피드 값과 일치하는 문서 검색
- 내장 문서와 일치하는 문서 검색
- 내장 문서의 필드 값과 일치하는 문서 검색
- 배열이 일치하는 문서 검색
- 배열의 값과 일치하는 문서 검색
- 연산자를 사용하여 문서 검색

컬렉션의 모든 문서 검색

모음의 모든 문서를 검색하려면 빈 쿼리 문서로 `find()` 작업을 사용하십시오.

다음 쿼리는 `example` 모음의 모든 문서를 반환합니다.

```
db.example.find( {} ).pretty()
```

피드 값과 일치하는 문서 검색

필드 및 값과 일치하는 모든 문서를 검색하려면 일치하는 필드 및 값을 식별하는 쿼리 문서로 `find()` 작업을 사용합니다.

이전 문서를 사용하면 이 쿼리는 "Item" 필드가 "Pen"과 동일한 모든 문서를 반환합니다.

```
db.example.find( { "Item": "Pen" } ).pretty()
```

내장 문서와 일치하는 문서 검색

내장 문서와 일치하는 모든 문서를 찾으려면 내장 문서 이름과 내장 문서의 모든 필드 및 값을 지정하는 쿼리 문서와 함께 `find()` 작업을 사용합니다.

내장 문서를 일치시킬 때 문서의 내장 문서는 쿼리에 있는 것과 동일한 이름을 가져야 합니다. 또한, 내장 문서의 필드 및 값은 쿼리와 일치해야 합니다.

다음 쿼리는 "Poster Paint" 문서만 반환합니다. "Pen"에는 "`OnHand`" 및 "`MinOnHand`"에 대한 다른 값이 있으며, "Spray Paint"에는 쿼리 문서보다 필드 하나(`OrderQnty`)가 더 있기 때문입니다.

```
db.example.find({"Inventory": {
    "OnHand": 47,
    "MinOnHand": 50 } } ).pretty()
```

내장 문서의 필드 값과 일치하는 문서 검색

내장 문서와 일치하는 모든 문서를 찾으려면 내장 문서 이름과 내장 문서의 모든 필드 및 값을 지정하는 쿼리 문서와 함께 `find()` 작업을 사용합니다.

이전 문서에서 다음 쿼리는 "점 표기법"을 사용하여 내장 문서와 관심 있는 필드를 지정합니다. 다른 필드가 내장 문서에 표시될 수 있는지 여부와 무관하게 이와 일치하는 모든 문서가 반환됩니다. "Poster Paint" 및 "Spray Paint"가 지정된 필드 및 값과 일치하므로 쿼리가 "Poster Paint" 및 "Spray Paint"를 반환합니다.

```
db.example.find({"Inventory.OnHand": 47, "Inventory.MinOnHand": 50 }).pretty()
```

배열이 일치하는 문서 검색

배열이 일치하는 모든 문서를 찾으려면 관심 있는 배열 이름과 해당 배열의 모든 값을 포함하여 `find()` 작업을 사용합니다. 쿼리가 배열 값이 동일하면서 쿼리와 동일한 순서인 해당 이름을 가진 배열을 포함한 모든 문서를 반환합니다.

"Poster Paint"에는 추가 색상(White)이 있고 "Spray Paint"에는 색상이 다른 순서로 있으므로 다음 쿼리는 "Pen"만을 반환합니다.

```
db.example.find( { "Colors": ["Red","Green","Blue","Black"] } ).pretty()
```

배열의 값과 일치하는 문서 검색

특정 배열 값을 보유한 모든 문서를 찾으려면 관심 있는 배열 이름과 값을 포함하여 `find()` 작업을 사용합니다.

```
db.example.find( { "Colors": "Red" } ).pretty()
```

각각 `Colors`라는 배열과 배열 내에 "Red" 값이 있으므로 이전 작업은 세 문서 모두를 반환합니다. "`White`" 값을 지정한 경우 쿼리는 "Poster Paint"만 반환합니다.

연산자를 사용하여 문서 검색

다음 쿼리는 "`Inventory.OnHand`" 값이 50 미만인 모든 문서를 반환합니다.

```
db.example.find(
        { "Inventory.OnHand": { $lt: 50 } } )
```

지원되는 쿼리 연산자 목록은 쿼리 및 프로젝션 연산자 섹션을 참조하십시오.

문서 업데이트

일반적으로 문서는 정적이 아니며 애플리케이션 워크플로의 일부로 업데이트됩니다. 다음은 문서를 업데이트할 수 있는 몇 가지 방법을 보여주는 예입니다.

기존 문서를 업데이트하려면 `update()` 작업을 사용합니다. `update()` 작업에는 두 개의 문서 파라미터가 있습니다. 첫 번째 문서는 업데이트할 문서를 식별합니다. 두 번째 문서는 업데이트를 지정합니다.

기존 필드를 업데이트할 때 (해당 필드가 단순 필드이든, 배열이든, 포함된 문서이든) 필드 이름과 해당 값을 지정합니다. 작업 종료 시 이전 문서의 필드가 새 필드와 값으로 교체된 것과 같습니다.

주제

- 기존 필드의 값 업데이트
- 새 필드 추가

- 내장 문서 교체
- 내장 문서에 새 필드 삽입
- 문서에서 필드 제거
- 여러 문서에서 필드 제거

기존 필드의 값 업데이트

다음 업데이트 작업에 대해 추가한 4개의 문서를 사용합니다.

```
{
    "Item": "Ruler",
    "Colors": ["Red","Green","Blue","Clear","Yellow"],
    "Inventory": {
        "OnHand": 47,
        "MinOnHand": 40
    },
    "UnitPrice": 0.89
},
{
    "Item": "Pen",
    "Colors": ["Red","Green","Blue","Black"],
    "Inventory": {
        "OnHand": 244,
        "MinOnHand": 72
    }
},
{
    "Item": "Poster Paint",
    "Colors": ["Red","Green","Blue","Black","White"],
    "Inventory": {
        "OnHand": 47,
        "MinOnHand": 50
    }
},
{
    "Item": "Spray Paint",
    "Colors": ["Black","Red","Green","Blue"],
    "Inventory": {
        "OnHand": 47,
        "MinOnHand": 50,
        "OrderQnty": 36
    }
```

```
}
```

단순 필드 업데이트

단순 필드를 업데이트하려면 $set와 함께 update()를 사용하여 필드 이름과 새 값을 지정합니다. 다음 예제에서는 Item을 "Pen"에서 "Gel Pen"으로 바꿉니다.

```
db.example.update(
    { "Item" : "Pen" },
    { $set: { "Item": "Gel Pen" } }
)
```

이 작업의 결과는 다음과 같습니다.

```
{
    "Item": "Gel Pen",
    "Colors": ["Red","Green","Blue","Black"],
    "Inventory": {
        "OnHand": 244,
        "MinOnHand": 72
    }
}
```

배열 업데이트

다음 예제에서는 색상 목록에서 Orange를 포함하고 White를 제외한 새 배열로 기존 배열을 교체합니다. 새 색상 목록은 update() 작업에서 지정된 순서입니다.

```
db.example.update(
    { "Item" : "Poster Paint" },
    { $set: { "Colors": ["Red","Green","Blue","Orange","Black"] } }
)
```

이 작업의 결과는 다음과 같습니다.

```
{
    "Item": "Poster Paint",
    "Colors": ["Red","Green","Blue","Orange","Black"],
    "Inventory": {
        "OnHand": 47,
        "MinOnHand": 50
```

```
    }
}
```

새 필드 추가

하나 이상의 새 필드를 추가하여 문서를 수정하려면 삽입할 문서 및 $set 연산자를 사용하여 삽입할 새 필드와 값을 식별하는 쿼리 문서에 update() 작업을 사용합니다.

다음 예제에서는 Spray Paint 문서에 UnitPrice 필드와 3.99 값을 추가합니다. 3.99라는 값은 숫자이지 문자열이 아닙니다.

```
db.example.update(
    { "Item": "Spray Paint" },
    { $set: { "UnitPrice": 3.99 } }
)
```

이 작업의 결과는 다음과 같습니다(JSON 형식).

```
{
    "Item": "Spray Paint",
    "Colors": ["Black","Red","Green","Blue"],
    "Inventory": {
        "OnHand": 47,
        "MinOnHand": 50,
        "OrderQnty": 36
    },
    "UnitPrice": 3.99
}
```

내장 문서 교체

내장 문서를 교체함으로써 문서를 수정하려면 내장 문서 및 $set 연산자를 사용한 새 필드와 값을 식별하는 문서에 update() 작업을 사용합니다.

다음과 같은 문서가 있다고 가정합니다.

```
db.example.insert({
    "DocName": "Document 1",
    "Date": {
        "Year": 1987,
        "Month": 4,
```

```
        "Day": 18
    }
})
```

내장 문서 교체

다음 예제에서는 현재 Date 문서를 새로운 문서로 교체합니다. 새 문서에는 Month 및 Day 필드만 있고, Year 필드는 제거되었습니다.

```
db.example.update(
    { "DocName" : "Document 1" },
    { $set: { "Date": { "Month": 4, "Day": 18 } } }
)
```

이 작업의 결과는 다음과 같습니다.

```
{
    "DocName": "Document 1",
    "Date": {
        "Month": 4,
        "Day": 18
    }
}
```

내장 문서에 새 필드 삽입

내장 문서에 필드를 추가하려면

내장 문서에 하나 이상의 새 필드를 추가하여 문서를 수정하려면 내장 문서와 내장 문서 지정을 위한 "점 표기법" 및 $set 연산자를 사용하여 삽입할 새 필드와 값을 식별하는 문서에 update() 작업을 사용합니다.

다음과 같은 문서가 있을 때 다음 코드는 "점 표기법"을 사용하여 Year 및 DoW 필드를 내장 Date 문서에 삽입하고, Words를 상위 문서에 삽입합니다.

```
{
    "DocName": "Document 1",
    "Date": {
        "Month": 4,
        "Day": 18
    }
```

```
}
```

```
db.example.update(
    { "DocName" : "Document 1" },
    { $set: { "Date.Year": 1987,
              "Date.DoW": "Saturday",
              "Words": 2482 } }
)
```

이 작업의 결과는 다음과 같습니다.

```
{
    "DocName": "Document 1",
    "Date": {
        "Month": 4,
        "Day": 18,
        "Year": 1987,
        "DoW": "Saturday"
    },
    "Words": 2482
}
```

문서에서 필드 제거

문서에서 필드를 제거하여 문서를 수정하려면 쿼리 문서에 필드를 제거할 문서를 식별하는 update() 작업을 사용하고, 제거할 필드를 지정하는 $unset 연산자를 사용합니다.

다음 예제에서는 이전 문서에서 Words 필드를 제거합니다.

```
db.example.update(
    { "DocName" : "Document 1" },
    { $unset: { Words:1 } }
)
```

이 작업의 결과는 다음과 같습니다.

```
{
    "DocName": "Document 1",
    "Date": {
        "Month": 4,
        "Day": 18,
```

```
        "Year": 1987,
        "DoW": "Saturday"
    }
}
```

여러 문서에서 필드 제거

여러 문서에서 필드를 제거하여 문서를 수정하려면 `$unset` 연산자와 `multi` 옵션 세트를 `true`로 설정하여 `update()` 작업을 사용합니다.

다음 예제에서는 예제 컬렉션의 모든 문서에서 `Inventory` 필드를 제거합니다. 문서에 `Inventory` 필드가 없으면 해당 문서에 어떤 작업도 수행되지 않습니다. `multi: true`가 생략되면 기준을 충족하는 첫 번째 문서에만 작업이 수행됩니다.

```
db.example.update(
    {},
    { $unset: { Inventory:1 } },
    { multi: true }
)
```

문서 삭제

데이터베이스에서 문서를 제거하려면 `remove()` 작업을 사용하여 제거할 문서를 지정합니다. 다음 코드는 `example` 모음에서 "Gel Pen"을 제거합니다.

```
db.example.remove( { "Item": "Gel Pen" } )
```

데이터베이스에서 모든 문서를 제거하려면 빈 쿼리와 함께 `remove()` 작업을 사용합니다.

```
db.example.remove( { } )
```

Amazon DocumentDB로 시작하기

Amazon DocumentDB에 연결하고 이를 통해 시작하는 방법에는 여러 가지가 있습니다. 이 가이드는 사용자가 강력한 문서 데이터베이스를 사용하여 시작할 수 있는 가장 빠르고 간단하며 쉬운 방법입니다. 이 가이드에서는 AWS CloudShell를 사용하여에서 직접 Amazon DocumentDB 클러스터를 연결하고 쿼리합니다 AWS Management Console. AWS 프리 티어 자격이 있는 신규 고객은 Amazon DocumentDB 및 CloudShell을 무료로 사용할 수 있습니다. AWS CloudShell 환경 또는 Amazon DocumentDB 클러스터가 프리 티어 이외의 리소스를 사용하는 경우 해당 리소스에 대한 일반 AWS 요금이 부과됩니다. 이 안내서를 통해 5분 이내에 Amazon DocumentDB를 시작할 수 있습니다.

> **Note**
>
> 이 가이드의 지침은 특히 Amazon DocumentDB 및를 사용할 AWS CloudShell 수 있는 Amazon DocumentDB 인스턴스 기반 클러스터를 생성하고 연결하는 데 사용됩니다.
>
> - Amazon DocumentDB 탄력성 클러스터를 생성하고 연결하려면 Amazon DocumentDB 엘라스틱 클러스터 시작하기을 참조하십시오.
> - AWS 중국 리전에 있는 경우 섹션을 참조하세요Amazon EC2 자동 연결.

주제

- 사전 조건
- 1단계: 클러스터 생성
- 2단계: 클러스터에 연결
- 3단계: 데이터 삽입 및 쿼리
- 4단계: 탐색

사전 조건

첫 번째 Amazon DocumentDB 클러스터를 작성하기 전에 다음 작업을 수행해야 합니다:

Amazon Web Services(AWS) 계정 만들기

Amazon DocumentDB 사용을 시작하기 전에 Amazon Web Services(AWS) 계정이 있어야 합니다. AWS 계정은 무료입니다. 사용하는 서비스 및 리소스에 대해서만 비용을 지불하는 것입니다.

이 없는 경우 다음 단계를 AWS 계정완료하여 생성합니다.

에 가입하려면 AWS 계정

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.

   에 가입하면 AWS 계정AWS 계정 루트 사용자이 생성됩니다. 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.

필요한 AWS Identity and Access Management (IAM) 권한을 설정합니다.

클러스터, 인스턴스 및 클러스터 파라미터 그룹과 같은 Amazon DocumentDB 리소스를 관리하기 위한 액세스에는가 요청을 인증하는 데 사용할 AWS 수 있는 자격 증명이 필요합니다. 자세한 내용은 Amazon DocumentDB의 ID 및 액세스 관리 단원을 참조하십시오.

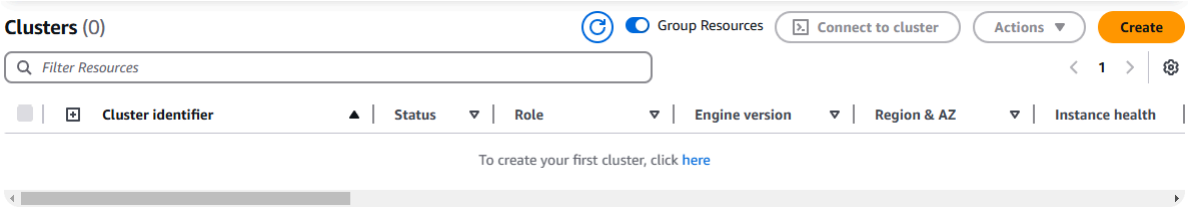
1. 의 검색 창에 IAM을 AWS Management Console입력하고 나타나는 드롭다운 메뉴에서 IAM을 선택합니다.
2. IAM 콘솔에 들어가면 탐색 창에서 사용자를 선택합니다.
3. 사용자 이름을 선택합니다.
4. Add permissions(권한 추가)를 클릭합니다.
5. 정책 직접 연결을 선택합니다.
6. 검색 창에 `AmazonDocDBFullAccess`을 입력하고 검색 결과에 표시되면 선택합니다.
7. 다음을 클릭합니다.
8. Add permissions(권한 추가)를 클릭합니다.

> **Note**
>
> AWS 계정에는 각 리전의 기본 VPC가 포함됩니다. Amazon VPC를 사용하기로 선택한 경우 Amazon VPC 사용 설명서의 Amazon VPC 생성 주제의 단계를 완료합니다.

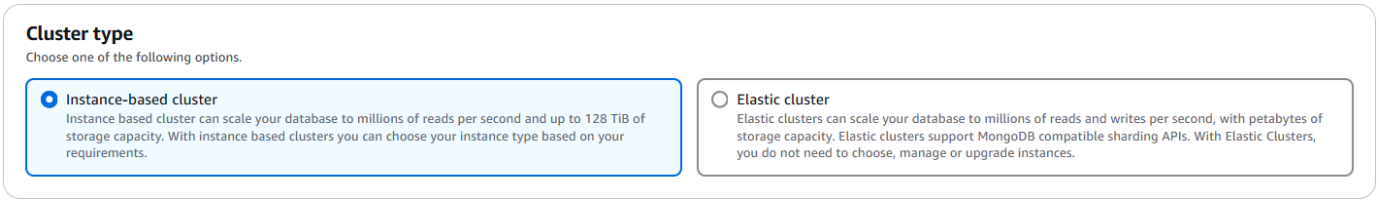
1단계: 클러스터 생성

이 단계에서 Amazon DocumentDB 클러스터를 생성합니다.

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. Amazon DocumentDB 관리 콘솔의 클러스터에서 생성을 선택합니다.



3. Amazon DocumentDB 클러스터 생성 페이지의 클러스터 유형 섹션에서 인스턴스 기반 클러스터(기본 옵션)를 선택합니다.

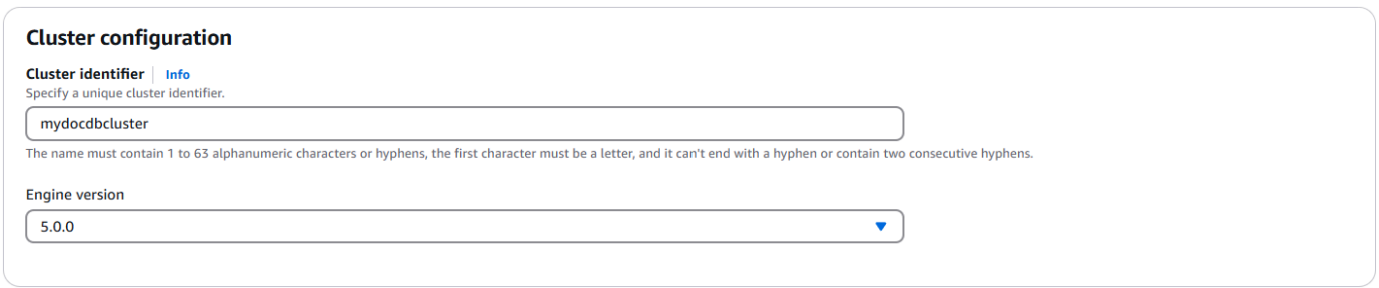


> **Note**
>
> 이 범주의 다른 옵션은 탄력적 클러스터입니다. Amazon DocumentDB 탄력적 클러스터에 대한 자세한 내용은 섹션을 참조하세요. Amazon DocumentDB 탄력적 클러스터 사용

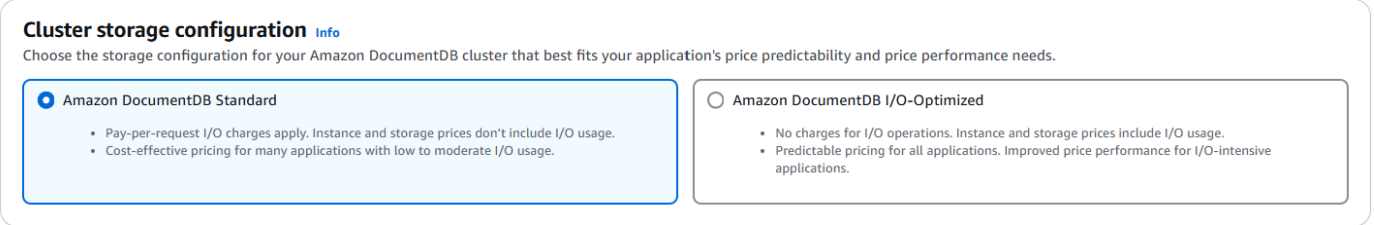
4. 클러스터 구성 섹션에서 다음을 수행합니다.

   a. 클러스터 식별자 에 **mydocdbcluster**와 같은 고유한 이름을 입력합니다. 콘솔은 모든 클러스터 이름을 입력 방식에 관계없이 소문자로 변경합니다.

   b. 엔진 버전의 경우 5.0.0을 선택합니다.



5. 클러스터 스토리지 구성 섹션에서 Amazon DocumentDB 표준(기본 옵션입니다)을 선택합니다.



> **Note**
>
> 이 범주의 다른 옵션은 Amazon DocumentDB I/O 최적화입니다. 두 옵션에 대한 자세한 내용은 섹션을 참조하세요. Amazon DocumentDB 클러스터 스토리지 구성

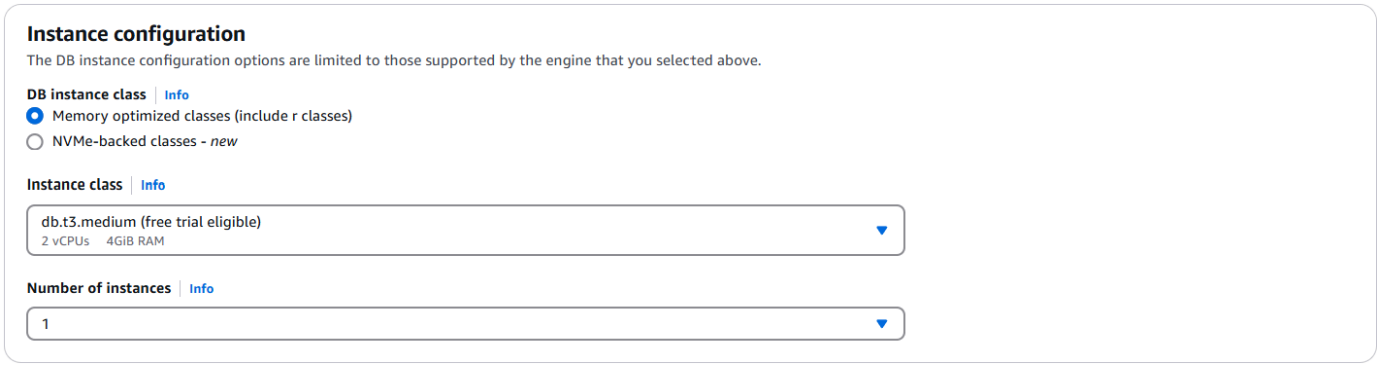
6. 인스턴스 구성 섹션에서:

   a. DB 인스턴스 클래스에서 메모리 최적화 클래스(r 클래스 포함)(기본값)를 선택합니다.

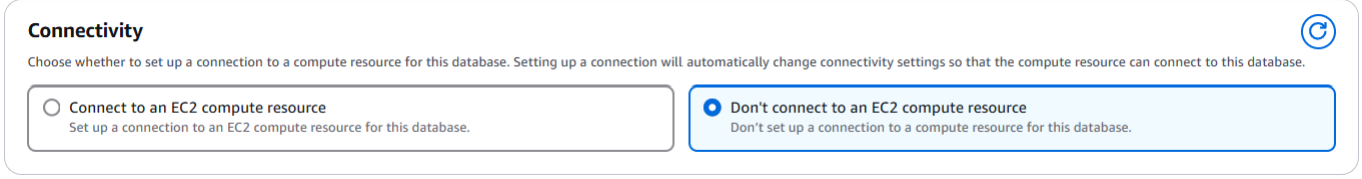
      다른 인스턴스 옵션은 NVMe 지원 클래스입니다. 자세한 내용은 NVMe 지원 인스턴스를 참조하세요.

   b. 인스턴스 클래스에서 db.t3.medium을 선택합니다. 무료 AWS 평가판을 사용할 수 있습니다.

   c. 인스턴스 수에 1 인스턴스를 선택합니다. 인스턴스 하나를 선택하면 비용을 최소화하는 데 도움이 됩니다. 프로덕션 시스템인 경우 고가용성을 위해 세 개의 인스턴스를 프로비저닝하는 것이 좋습니다.

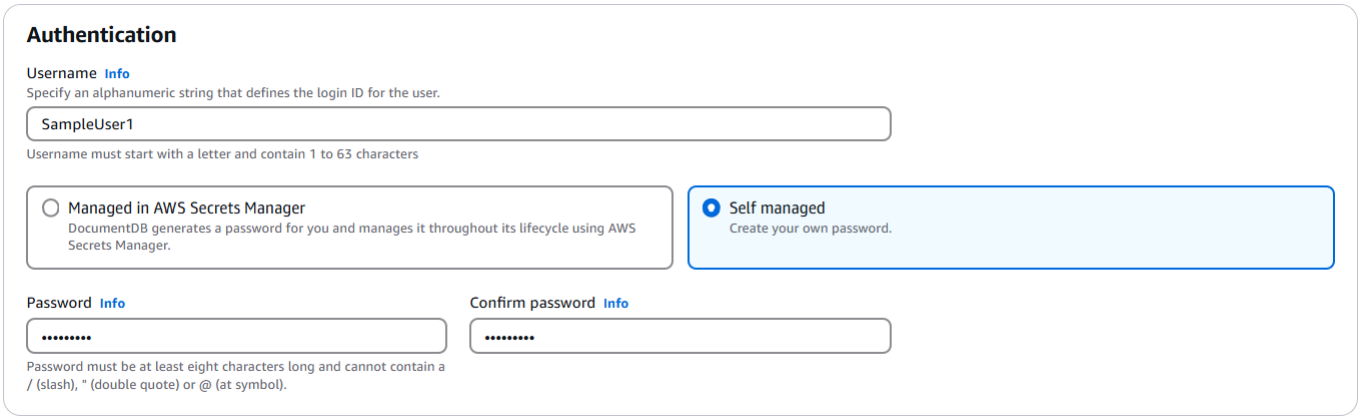


7. 연결 섹션에서 EC2 컴퓨팅 리소스에 연결하지 않음의 기본 설정을 그대로 둡니다.



8. 인증 섹션에서 기본 사용자의 사용자 이름을 입력한 다음 자체 관리를 선택합니다. 암호를 입력한 다음 확인합니다.

   대신 관리형을 AWS Secrets Manager 선택한 경우 자세한 내용은 Amazon DocumentDB 및를 사용한 암호 관리 AWS Secrets Manager 섹션을 참조하세요.



9. 다른 모든 옵션을 기본값으로 두고 클러스터 생성을 선택합니다.

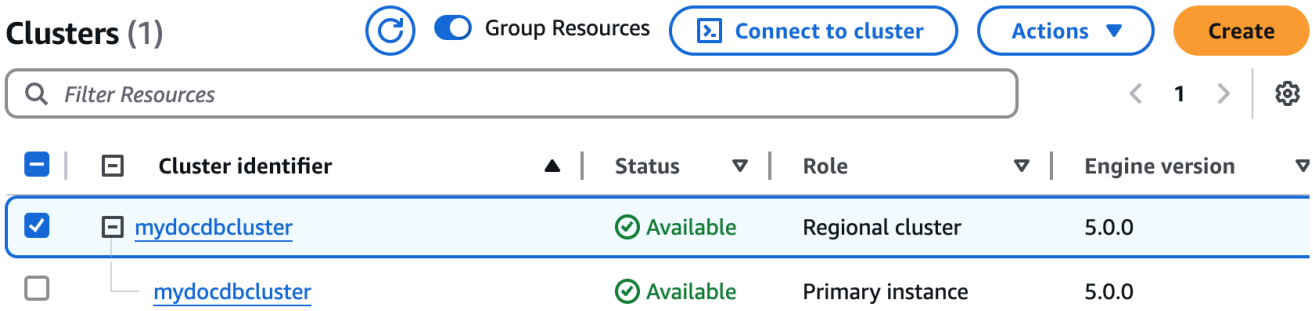
Amazon DocumentDB는 이제 클러스터를 프로비저닝하고 있으며, 완료하는 데 몇 분 정도 걸릴 수 있습니다.

> **Note**
>
> 클러스터 상태 값에 대한 자세한 내용은 Amazon DocumentDB 모니터링의 클러스터 상태 값(을)를 참조하십시오.

2단계: 클러스터에 연결

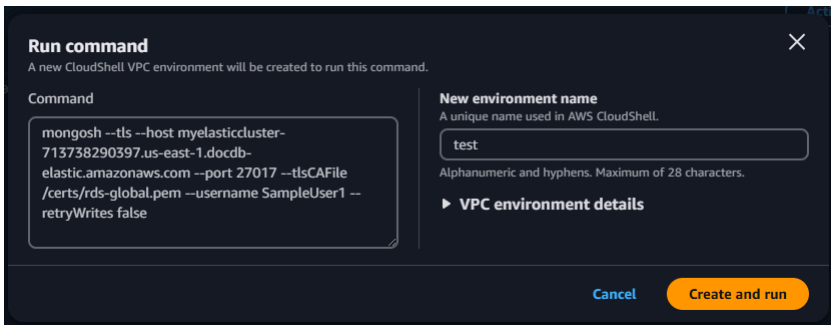
를 사용하여 Amazon DocumentDB 클러스터에 연결합니다 AWS CloudShell.

1. Amazon DocumentDB 관리 콘솔의 클러스터에서 생성한 클러스터를 찾습니다. 클러스터 옆의 확인란을 클릭하여 클러스터를 선택합니다.



2. 클러스터에 연결을 클릭합니다(작업 드롭다운 메뉴 옆). 이 버튼은 클러스터 옆의 확인란을 클릭하고 리전 클러스터와 기본 인스턴스(들)의 상태가 모두 사용 가능으로 표시된 후에만 활성화됩니다. CloudShell 명령 실행 화면이 나타납니다.

3. 새 환경 이름 필드에 "test"와 같은 고유한 이름을 입력하고 생성 및 실행을 클릭합니다. VPC 환경 세부 정보는 Amazon DocumentDB 데이터베이스에 대해 자동으로 구성됩니다.



4. 메시지가 표시되면 1단계: Amazon DocumentDB 클러스터 생성(하위 7단계)에서 생성한 암호를 입력합니다.



암호를 입력하고 프롬프트`rs0 [direct: primary] <env-name>>`가 되면 Amazon DocumentDB 클러스터에 성공적으로 연결된 것입니다.

> **Note**
>
> 흐름 문제 해결에 대한 자세한 내용은 Amazon DocumentDB 문제 해결을 참조하십시오.

3단계: 데이터 삽입 및 쿼리

이제 클러스터에 연결되었으므로 몇 가지 쿼리를 실행하여 문서 데이터베이스 사용에 익숙해질 수 있습니다.

1. 단일 문서를 삽입하려면 다음을 입력합니다:

   ```
   db.collection.insertOne({"hello":"DocumentDB"})
   ```

   출력은 다음과 같습니다:

   ```
   {
     acknowledged: true,
     insertedId: ObjectId('673657216bdf6258466b128c')
   }
   ```

2. `findOne()` 명령으로 작성한 문서를 읽을 수 있습니다 (단일 문서만 반환하기 때문). 다음 내용을 입력합니다.

   ```
   db.collection.findOne()
   ```

   다음과 같은 출력내용을 얻게 됩니다.

   ```
   { "_id" : ObjectId("5e401fe56056fda7321fbd67"), "hello" : "DocumentDB" }
   ```

3. 쿼리를 몇 개 더 수행하려면 게임 프로필 사용 사례를 고려해 보세요. 먼저 `profiles`이라는 제목의 컬렉션에 몇 개의 항목을 삽입합니다. 다음 내용을 입력합니다.

   ```
   db.profiles.insertMany([{ _id: 1, name: 'Matt', status: 'active', level: 12, score:
    202 },
         { _id: 2, name: 'Frank', status: 'inactive', level: 2, score: 9 },
         { _id: 3, name: 'Karen', status: 'active', level: 7, score: 87 },
         { _id: 4, name: 'Katie', status: 'active', level: 3, score: 27 }
   ])
   ```

다음과 같은 출력내용을 얻게 됩니다.

```
{ acknowledged: true, insertedIds: { '0': 1, '1': 2, '2': 3, '3': 4 } }
```

4. find() 명령을 사용하여 프로필 컬렉션의 모든 문서를 반환합니다. 다음 내용을 입력합니다.

```
db.profiles.find()
```

3단계에서 입력한 데이터와 일치하는 출력이 표시됩니다.

5. 필터를 사용하여 단일 문서에 대한 쿼리를 사용하세요. 다음을 입력합니다:

```
db.profiles.find({name: "Katie"})
```

다음과 같은 출력내용을 얻게 됩니다.

```
{ "_id" : 4, "name" : "Katie", "status": "active", "level": 3, "score":27}
```

6. 이제 프로필을 찾아 findAndModify 명령을 사용하여 해당 프로필을 수정해 보겠습니다. 다음 코드를 사용하여 Matt 사용자에게 추가로 10점을 부여합니다.

```
db.profiles.findAndModify({
        query: { name: "Matt", status: "active"},
        update: { $inc: { score: 10 } }
    })
```

다음과 같은 결과가 출력됩니다 (참고로 그의 점수는 아직 오르지 않았습니다):

```
{
    [{_id : 1, name : 'Matt', status: 'active', level: 12, score: 202}]
```

7. 다음 쿼리를 통해 그의 점수가 변경되었는지 확인할 수 있습니다:

```
db.profiles.find({name: "Matt"})
```

출력은 다음과 같습니다.

```
{ "_id" : 1, "name" : "Matt", "status" : "active", "level" : 12, "score" : 212 }
```

4단계: 탐색

축하합니다! Amazon DocumentDB 인스턴스 기반 클러스터에 대한 시작하기 안내서를 성공적으로 완료했습니다.

다음 단계? 몇 가지 인기 있는 기능을 통해 이 데이터베이스를 최대한 활용하는 방법을 알아보십시오:

- Amazon DocumentDB 관리
- 스케일링
- 백업 및 복원

> **Note**
>
> 이 시작하기 연습에서 생성한 클러스터는 삭제하지 않는 한 비용이 계속 발생합니다. 자세한 지침은 Amazon DocumentDB 클러스터 삭제를 참조하십시오.

를 사용한 Amazon DocumentDB 빠른 시작 AWS CloudFormation

이 섹션에는 AWS CloudFormation을 사용하여 Amazon DocumentDB(MongoDB 호환)를 신속하게 시작하는 데 도움이 되는 단계 및 기타 정보가 포함되어 있습니다. Amazon DocumentDB 엔드포인트에 관한 일반 정보는 Amazon DocumentDB(MongoDB 호환)란 무엇인가를 참조하세요.

이 지침은 AWS CloudFormation 템플릿을 사용하여 기본 Amazon VPC에서 클러스터와 인스턴스를 생성합니다. 이러한 리소스를 사용자가 직접 생성하는 방법에 관한 지침은 Amazon DocumentDB로 시작하기(을)를 참조하십시오.

> **Important**
>
> 이 템플릿으로 생성된 AWS CloudFormation 스택은 Amazon DocumentDB(예: 클러스터 및 인스턴스) 및 Amazon Elastic Compute Cloud(예: 서브넷 그룹)의 리소스를 포함하여 여러 리소스를 생성합니다.
>
> 이 리소스 중 일부는 프리 티어 리소스가 아닙니다. 가격 정보는 Amazon DocumentDB 가격 및 Amazon EC2 가격를 참조하십시오. 이 단원을 완료한 후에는 스택을 삭제하여 요금이 발생하지 않게 할 수 있습니다.

이 AWS CloudFormation 스택은 자습서 목적으로만 사용됩니다. 운영 환경에서 이 템플릿을 사용하는 경우 더 엄격한 IAM 정책 및 보안을 사용하는 것이 좋습니다. 리소스 보안에 대한 자세한 내용은 Amazon VPC 시큐리티 및 Amazon EC2 네트워크 및 시큐리티를 참조하십시오.

주제

- 사전 조건
- Amazon DocumentDB AWS CloudFormation 스택 시작
- Amazon DocumentDB 클러스터 액세스
- 종료 방지 및 삭제 방지

사전 조건

Amazon DocumentDB 클러스터를 작성하기 전에 다음 항목이 있어야 합니다.

- 기본 Amazon VPC

- 필요한 IAM 권한

필요한 IAM 권한

다음 권한을 통해 AWS CloudFormation 스택에 리소스를 생성할 수 있습니다:

AWS 관리형 정책

- `AWSCloudFormationReadOnlyAccess`
- `AmazonDocDBFullAccess`

추가 IAM 권한

다음 정책은이 AWS CloudFormation 스택을 생성하고 삭제하는 데 필요한 추가 권한을 간략하게 설명합니다.

다음 예제에서는 각 *### ## ## ##*자를 리소스 정보로 바꿉니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DocDBPermissions",
            "Effect": "Allow",
            "Action": [
                "rds:CreateDBCluster",
                "rds:DeleteDBCluster",
                "rds:ModifyDBCluster",
                "rds:DescribeDBClusters",
                "rds:CreateDBInstance",
                "rds:DeleteDBInstance",
                "rds:ModifyDBInstance",
                "rds:DescribeDBInstances",
                "rds:CreateDBSubnetGroup",
                "rds:DeleteDBSecurityGroup",
                "rds:DescribeDBSubnetGroups"
            ],
            "Resource": [
                "arn:aws:rds:{AWS_REGION}:{AWS_ACCOUNT_ID}:cluster:*",
                "arn:aws:rds:{AWS_REGION}:{AWS_ACCOUNT_ID}:db:*",
                "arn:aws:rds:{AWS_REGION}:{AWS_ACCOUNT_ID}:subgrp:*"
            ]
```

```
        },
        {
            "Sid": "EC2NetworkingPermissions",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateSecurityGroup",
                "ec2:DeleteSecurityGroup",
                "ec2:DescribeSecurityGroups",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:RevokeSecurityGroupIngress",
                "ec2:DescribeVpcs",
                "ec2:DescribeSubnets"
            ],
            "Resource": [
                "arn:aws:ec2:{AWS_REGION}:{AWS_ACCOUNT_ID}:security-group/*",
                "arn:aws:ec2:{AWS_REGION}:{AWS_ACCOUNT_ID}:vpc/*",
                "arn:aws:ec2:{AWS_REGION}:{AWS_ACCOUNT_ID}:subnet/*"
            ]
        },
        {
            "Sid": "EC2DescribePermissions",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeAvailabilityZones"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CloudWatchLogsPermissions",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData",
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:{AWS_REGION}:{AWS_ACCOUNT_ID}:log-group:/aws/docdb/*",
                "arn:aws:logs:{AWS_REGION}:{AWS_ACCOUNT_ID}:log-group:/aws/docdb/*:log-
stream:*"
            ]
        },
        {
```

```
            "Sid": "KMSPermissions",
            "Effect": "Allow",
            "Action": [
                "kms:CreateKey",
                "kms:DescribeKey",
                "kms:EnableKey",
                "kms:ListKeys",
                "kms:PutKeyPolicy"
            ],
            "Resource": "arn:aws:kms:{AWS_REGION}:{AWS_ACCOUNT_ID}:key/*"
        },
        {
            "Sid": "KMSEncryption",
            "Effect": "Allow",
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:GenerateDataKey"
            ],
            "Resource": "arn:aws:kms:{AWS_REGION}:{AWS_ACCOUNT_ID}:key/*",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": [
                        "rds.{AWS_REGION}.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Sid": "IAMServiceLinkedRole",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::{AWS_ACCOUNT_ID}:role/aws-service-role/
rds.amazonaws.com/AWSServiceRoleForRDS",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "rds.amazonaws.com"
                }
            }
        }
    ]
}
```

Amazon EC2 키 페어

AWS CloudFormation 스택을 생성할 리전에서 키 페어(및 PEM 파일)를 사용할 수 있어야 합니다. 키 페어를 만들어야 할 경우, Amazon EC2 사용 설명서의 Amazon EC2를 사용하여 키 페어 만들기를 참조하세요.

Amazon DocumentDB AWS CloudFormation 스택 시작

이 섹션에서는 Amazon DocumentDB AWS CloudFormation 스택을 시작하고 구성하는 방법에 대해 설명합니다.

1. 에서에 로그인 AWS Management Console 합니다https://console.aws.amazon.com/.
2. 다음 표는 각 AWS 리전에 대한 Amazon DocumentDB 스택 템플릿을 나열합니다. 스택을 시작하려는 에 대해 스택 시작을 선택합니다. AWS 리전

리전	템플릿 보기	Designer에서 보기	시작
미국 동부(오하이오)	템플릿 보기	Designer에서 보기	Launch Stack
미국 동부(버지니아 북부)	템플릿 보기	Designer에서 보기	Launch Stack
미국 서부(오레곤)	템플릿 보기	Designer에서 보기	Launch Stack
아시아 태평양(뭄바이)	템플릿 보기	Designer에서 보기	Launch Stack
아시아 태평양(서울)	템플릿 보기	Designer에서 보기	Launch Stack
아시아 태평양(싱가포르)	템플릿 보기	Designer에서 보기	Launch Stack
아시아 태평양(시드니)	템플릿 보기	Designer에서 보기	Launch Stack
아시아 태평양(도쿄)	템플릿 보기	Designer에서 보기	Launch Stack

리전	템플릿 보기	Designer에서 보기	시작
캐나다(중부)	템플릿 보기	Designer에서 보기	Launch Stack
유럽(프랑크푸르트)	템플릿 보기	Designer에서 보기	Launch Stack
유럽(아일랜드)	템플릿 보기	Designer에서 보기	Launch Stack
유럽(런던)	템플릿 보기	Designer에서 보기	Launch Stack
유럽(파리)	템플릿 보기	Designer에서 보기	Launch Stack

3. 스택 생성 — 선택한 Amazon DocumentDB 템플릿을 설명합니다. 모든 스택은 스택에 포함하려는 AWS 리소스에 대한 구성이 포함된 JSON 또는 YAML 파일인 템플릿을 기반으로 합니다. 위에 제공된 템플릿에서 스택을 시작하도록 선택했으므로 선택한에 대한 Amazon DocumentDB 스택을 생성하도록 템플릿이 이미 구성되어 AWS 리전 있습니다.

   AWS CloudFormation 스택을 시작하면 Amazon DocumentDB 클러스터에 대한 삭제 보호가 기본적으로 비활성화됩니다. 클러스터에 대해 삭제 방지를 활성화하려면 다음 단계를 완료하십시오. 그렇지 않으면 다음을 선택하여 다음 단계를 계속하십시오.

   Amazon DocumentDB 클러스터에 대해 삭제 보호를 사용하려면:

   1. 스택 생성 페이지의 오른쪽 하단에서 Designer에서 보기를 선택합니다.
   2. 콘솔의 결과 AWS CloudFormation Designer 페이지에서 통합 JSON 및 YAML 편집기를 사용하여 템플릿을 수정합니다. 다음과 같이 `Resources` 섹션으로 스크롤하여 `DeletionProtection`을 포함하도록 수정합니다. AWS CloudFormation Designer 사용에 대한 자세한 내용은 AWS CloudFormation Designer란 무엇입니까?를 참조하세요.

      JSON:

```
"Resources": {
    "DBCluster": {
        "Type": "AWS::DocDB::DBCluster",
        "DeletionPolicy": "Delete",
        "Properties": {
            "DBClusterIdentifier": {
```

```
                "Ref": "DBClusterName"
            },
            "MasterUsername": {
                "Ref": "MasterUser"
            },
            "MasterUserPassword": {
                "Ref": "MasterPassword"
            },
            "DeletionProtection": "true"
        }
    },
```

YAML:

```
Resources:
  DBCluster:
    Type: 'AWS::DocDB::DBCluster'
    DeletionPolicy: Delete
    Properties:
      DBClusterIdentifier: !Ref DBClusterName
      MasterUsername: !Ref MasterUser
      MasterUserPassword: !Ref MasterPassword
      DeletionProtection: 'true'
```

3. 페이지 왼쪽 위에서 스택 만들기(



)

를 선택하여 변경 사항을 저장하고 이러한 변경 사항이 활성화된 스택을 만듭니다.

4. 변경 사항을 저장하면 스택 만들기 페이지로 리디렉션됩니다.
5. 다음을 선택하여 계속 진행합니다.

4. 스택 상세 내역 지정 — 템플릿의 스택 이름과 매개 변수를 입력합니다. 파라미터는 템플릿에서 정의되며, 이를 통해 스택을 생성하거나 업데이트 할 때 사용자 지정 값을 입력할 수 있습니다.
   - 스택 이름에서 스택의 이름을 입력하거나 제공된 이름을 그대로 사용합니다. 스택 이름에는 문자(A—Z 및 a—z), 숫자(0-9) 및 대시(—)를 포함할 수 있습니다.
   - 파라미터에서 다음 세부 정보를 입력합니다:
     - DBC클러스터 이름 — Amazon DocumentDB 클러스터의 이름을 입력하거나 제공된 이름을 사용합니다.

       클러스터 명명 제약 조건:

- 길이는 [1-63]개의 글자, 숫자 또는 하이픈입니다.
- 첫 번째 문자는 글자이어야 합니다.
- 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다.
- 리전별로 별로 Amazon RDS, Neptune 및 Amazon DocumentDB의 모든 클러스터 AWS 계정에 대해 고유해야 합니다.

- DBInstanceClass — 드롭다운 목록에서 Amazon DocumentDB 클러스터의 인스턴스 클래스를 선택합니다.
- DBInstanceName — Amazon DocumentDB 인스턴스의 이름을 입력하거나 제공된 이름을 사용합니다.

  인스턴스 명명 제약 조건:

  - 길이는 [1-63] 문자, 숫자 또는 하이픈입니다.
  - 첫 번째 문자는 글자이어야 합니다.
  - 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다.
  - 리전별로 별로 Amazon RDS, Neptune 및 Amazon DocumentDB의 모든 인스턴스 AWS 계정에 대해 고유해야 합니다.
- MasterPassword — 데이터베이스 관리자 계정 암호입니다.
- MasterUser — 데이터베이스 관리자 계정 사용자 이름입니다. MasterUser는 문자로 시작해야 하며 영숫자만 포함할 수 있습니다.

다음을 선택하여 변경 사항을 저장하고 계속합니다.

5. 스택옵션 구성 — 스택의 태그, 권한 및 추가 옵션을 구성합니다.
   - 태그 — 스택의 리소스에 적용할 태그(키-값) 쌍을 지정합니다. 각 스택에 대해 최대 50개의 고유 태그를 추가할 수 있습니다.
   - 권한 — 선택 사항. IAM 역할을 선택하여가 스택에서 리소스를 생성, 수정 또는 삭제 AWS CloudFormation 하는 방법을 명시적으로 정의합니다. 역할을 선택하지 않으면는 사용자 자격 증명을 기반으로 권한을 AWS CloudFormation 사용합니다. 서비스 역할을 지정하기 전에 역할을 전달할 수 있는 권한이 있는지 확인합니다(`iam:PassRole`). `iam:PassRole` 권한은 사용 가능한 역할을 지정합니다.

     > **Note**
     >
     > 서비스 역할을 지정하면는 AWS CloudFormation 항상 해당 스택에서 수행되는 모든 작업에 대해 해당 역할을 사용합니다. 이 스택에서 작업을 수행할 수 있는 권한이 있는 다른 사용자는 이 역할을 전달할 수 있는 권한이 없더라도 이 역할을 사용할 수 있

습니다. 역할에 사용자에게 불필요한 권한이 포함되어 있는 경우 사용자의 권한을 실수로 에스컬레이션할 수 있습니다. 역할이 최소 권한을 부여하는지 확인합니다.

- 고급 옵션 — 다음 고급 옵션을 설정할 수 있습니다:
  - 스택 정책 — 선택 사항. 스택 업데이트 중 의도치 않게 업데이트되지 않도록 하려는 리소스를 정의합니다. 기본적으로 스택 업데이트 중에는 모든 리소스가 업데이트될 수 있습니다.

    스택 정책을 바로 JSON으로 입력하거나 스택 정책이 포함된 JSON 파일을 업로드할 수 있습니다. 자세한 내용은 스택 리소스에 대한 업데이트 방지를 참조하십시오.
  - 롤백 구성 — 선택 사항. 스택 AWS CloudFormation 을 생성하고 업데이트할 때 모니터링할 에 대한 CloudWatch Logs 경보를 지정합니다. 작업이 경보 임계값을 위반하면는 경보 임계값을 AWS CloudFormation 롤백합니다.
  - 알림 옵션 — 선택 사항. Simple Notification System(SNS)에 대한 주제를 지정합니다.
  - 스택 생성 옵션 — 선택 사항. 다음과 같은 옵션을 지정할 수 있습니다:
    - 실패 시 롤백 — 스택 생성이 실패한 경우 스택을 롤백해야 하는지 여부입니다.
    - 제한시간 —스택 생성 시간이 초과되기 전까지의 시간(분)입니다.
    - 종료 보호 — 스택이 실수로 삭제되는 것을 방지합니다.

      > **Note**
      >
      > AWS CloudFormation 종료 방지는 삭제 방지라는 Amazon DocumentDB 개념과 다릅니다. 자세한 내용은 종료 방지 및 삭제 방지 단원을 참조하십시오.

다음을 선택하여 계속 진행합니다.

6. 리뷰 <stack-name> — 스택 템플릿, 세부 정보 및 구성 옵션을 검토합니다. 페이지 하단에서 빠른 생성 링크를 열어 이와 동일한 기본 구성을 가진 스택을 만들 수도 있습니다.

   - 생성을 선택하여 스택을 생성합니다.
   - 또는 변경 세트 만들기를 선택할 수 있습니다. 변경 세트는 스택을 만들기 전에 이 스택을 구성하는 방법에 대한 미리 보기입니다. 이렇게 하면 변경 세트를 실행하기 전에 다양한 구성을 검사할 수 있습니다.

Amazon DocumentDB 클러스터 액세스

AWS CloudFormation 스택이 완료되면 Amazon EC2 인스턴스를 사용하여 Amazon DocumentDB 클러스터에 연결할 수 있습니다. SSH를 사용하여 Amazon EC2 인스턴스에 연결하는 방법에 대한 자세한 내용은 Amazon EC2 사용자 설명서의 Linux 인스턴스에 연결을 참조하십시오.

연결된 후에는 Amazon DocumentDB 사용에 대한 내용이 포함된 다음 섹션을 참조하십시오.

- 3단계: 데이터 삽입 및 쿼리
- Amazon DocumentDB 리소스 관리
- Amazon DocumentDB 모니터링

종료 방지 및 삭제 방지

삭제 보호 및 종료 보호를 가능하게 하는 Amazon DocumentDB 모범 사례입니다. CloudFormation 종료 보호는 Amazon DocumentDB 삭제 보호 기능과는 확연히 다른 기능입니다.

- 종료 보호 — CloudFormation 스택에 대한 종료 보호 기능을 사용하여 스택이 실수로 삭제되는 것을 방지할 수 있습니다. 종료료 보호 기능이 활성화된 스택을 삭제하려고 시도하면 삭제가 실패하고 스택은 변함없이 그대로 유지됩니다. CloudFormation을 사용하여 스택을 만들 때 기본적으로 종료 보호 기능이 비활성화됩니다. 스택을 생성할 때 종료 보호 기능을 활성화할 수 있습니다. 자세한 내용은 AWS CloudFormation 스택 옵션 설정을 참조하세요.
- 삭제 보호 — 또한 Amazon DocumentDB는 클러스터에 대해 삭제 보호를 실행하는 기능을 제공합니다. 사용자가 삭제 보호가 설정된 Amazon DocumentDB 클러스터를 삭제하려고 할 경우 삭제가 실패하고 클러스터는 변경되지 않습니다. 삭제 방지 기능을 활성화하면 Amazon DocumentDB AWS Management Console, AWS CLI및 CloudFormation에서 실수로 삭제되는 것을 방지합니다. Amazon DocumentDB 클러스터에 대한 삭제 방지 사용 및 사용 안 함에 대한 자세한 내용은 삭제 방지을 참조하십시오.

Amazon DocumentDB의 MongoDB 호환성

Amazon DocumentDB는 MongoDB 4.0과 MongoDB 5.0을 포함한 MongoDB 호환을 지원합니다. MongoDB 호환은 MongoDB 데이터베이스에서 현재 이미 사용하고 있는 애플리케이션, 드라이버 및 도구의 대다수를 거의 변경하지 않고 Amazon DocumentDB에서 사용할 수 있음을 의미합니다. 이 섹션에서는 MongoDB와의 Amazon DocumentDB 호환성에 대해 알아야 할 모든 것을 설명합니다. 여기에는 새로운 기능과 특성, 시작, 마이그레이션 경로 및 기능 차이 등이 포함됩니다.

주제

- MongoDB 5.0 호환성
- MongoDB 4.0 호환성

MongoDB 5.0 호환성

주제

- Amazon DocumentDB 5.0의 새로운 기능
- Amazon DocumentDB 5.0 시작하기
- Amazon DocumentDB 5.0으로 업그레이드 또는 마이그레이션
- 기능적 차이

Amazon DocumentDB 5.0의 새로운 기능

Amazon DocumentDB 5.0은 스토리지 제한 및 클라이언트 측 필드 레벨 암호화를 포함하는 새로운 기능과 특성을 소개합니다. 아래 요약은 Amazon DocumentDB 5.0에서 도입된 몇 가지 주요 특성을 소개합니다. 새로운 기능의 전체 목록을 보려면 릴리스 정보 섹션을 참조하세요.

- 인스턴스 기반의 모든 Amazon DocumentDB 클러스터 및 샤드 기반의 탄력적 클러스터에 대해 128TiB로 스토리지 제한을 높였습니다.
- Amazon DocumentDB 5.0 Engine Version 3.0.775) 소개
  - MongoDB 5.0 API 드라이버 지원
  - 클라이언트 측 FLE(Field Level Encryption)를 지원합니다. 이제 Amazon DocumentDB 클러스터에 데이터를 쓰기 전에 클라이언트 측에서 필드를 암호화할 수 있습니다. 자세한 내용은 클라이언트 측 필드 수준 암호화를 참조하세요.

- 새 집계 연산자:`$dateAdd`, `$dateSubtract`
- 연산자를 사용한 `$elemMatch` 인덱스 지원. 따라서 쿼리가 `$elemMatch`이면 인덱스 스캔이 발생합니다.

Amazon DocumentDB는 모든 MongoDB 5.0 기능을 지원하지는 않습니다. Amazon DocumentDB 5.0을 구축할 당시에는 고객이 가장 많이 구축하라고 요청한 기능과 특성을 다시 연구했습니다. 앞으로도 고객들이 구축을 요청하는 내용을 바탕으로 MongoDB 5.0 기능을 추가할 예정입니다. 지원되는 API의 최신 목록은 Amazon DocumentDB에서 지원되는 MongoDB API, 작업 및 데이터 형식 섹션을 참조하세요.

Amazon DocumentDB 5.0 시작하기

Amazon DocumentDB 5.0을 시작하려면 시작 가이드를 참조하세요. AWS Management Console 또는 AWS SDK AWS CLI또는를 사용하여 새 Amazon DocumentDB 5.0 클러스터를 생성할 수 있습니다 AWS CloudFormation. Amazon DocumentDB에 연결할 경우 MongoDB 5.0 이상과 호환되는 MongoDB 드라이버 또는 유틸리티를 사용해야 합니다.

> **Note**
>
> AWS SDK AWS CLI또는를 사용하는 경우 AWS CloudFormation엔진 버전은 기본적으로 5.0.0으로 설정됩니다. 새 Amazon DocumentDB 4.0 클러스터 또는 `engineVersion = 3.6.0`를 생성하거나 새 Amazon DocumentDB 3.6 클러스터를 생성하려면 `engineVersion = 4.0.0` 파라미터를 명시적으로 지정해야 합니다. 지정된 Amazon DocumentDB 클러스터의 경우를 사용하여 AWS CLI 를 호출`describe-db-clusters`하거나 Amazon DocumentDB 관리 콘솔을 사용하여 특정 클러스터의 엔진 버전 번호를 볼 수 있습니다.

Amazon DocumentDB 5.0은 클러스터에 대해 `r6g` 및 `t4.medium` 인스턴스 유형과 같은 Amazon EC2 Graviton2 프로세서를 지원하며 모든 지원 리전에서 사용할 수 있습니다. 요금에 대한 자세한 내용은 Amazon DocumentDB(MongoDB 호환) 요금을 참조하세요.

Amazon DocumentDB 5.0으로 업그레이드 또는 마이그레이션

AWS DMS 또는 `mongodump`, `mongorestore`, `mongoimport`, `mongoexport`와 같은 유틸리티를 사용하여 MongoDB 3.6 또는 MongoDB 4.0에서 Amazon DocumentDB 5.0으로 마이그레이션할 수 있습니다. 마이그레이션 방법에 대한 지침은 를 사용하여 Amazon DocumentDB 클러스터 업그레이드 AWS Database Migration Service 섹션을 참조하세요.

기능적 차이

Amazon DocumentDB 4.0 및 5.0 간의 기능적 차이

Amazon DocumentDB 5.0 릴리스에서는 Amazon DocumentDB 4.0과 Amazon DocumentDB 5.0 간에 기능적 차이가 있습니다.

- 이제 백업 기본 제공 역할이 `serverStatus`를 지원합니다. 작업 - 백업 역할을 가진 개발자 및 응용 프로그램은 Amazon DocumentDB 클러스터의 상태에 대한 통계를 수집할 수 있습니다.
- `replSetGetConfig` 출력에서 `SecondaryDelaySecs` 필드가 `slaveDelay`를 대체합니다.
- hello 명령이 `isMaster`를 대체함 - hello는 Amazon DocumentDB 클러스터의 역할을 설명하는 문서를 반환합니다.
- Amazon DocumentDB 5.0은 이제 첫 번째 중첩 수준에서 `$elemMatch` 연산자를 사용한 인덱스 스캔을 지원합니다. 쿼리 전용 필터에 한 레벨의 `$elemMatch` 필터가 있는 경우 인덱스 스캔이 지원되지만 중첩된 `$elemMatch` 쿼리가 포함된 경우에는 필터가 지원되지 않습니다.

  예를 들어, Amazon DocumentDB 5.0에서 `$elemMatch` 연산자를 중첩 수준에 포함하면 Amazon DocumentDB 4.0에서처럼 값이 반환되지 않습니다.

```
db.foo.insert(
[
    {a: {b: 5}},
    {a: {b: [5]}},
    {a: {b: [3, 7]}},
    {a: [{b: 5}]},
    {a: [{b: 3}, {b: 7}]},
    {a: [{b: [5]}]},
    {a: [{b: [3, 7]}]},
    {a: [[{b: 5}]]},
    {a: [[{b: 3}, {b: 7}]]},
    {a: [[{b: [5]}]]},
    {a: [[{b: [3, 7]}]]}
]);

// DocumentDB 5.0
> db.foo.find({a: {$elemMatch: {b: {$elemMatch: {$lt: 6, $gt: 4}}}}}, {_id: 0})
{ "a" : [ { "b" : [ 5 ] } ] }

// DocumentDB 4.0
> db.foo.find({a: {$elemMatch: {b: {$elemMatch: {$lt: 6, $gt: 4}}}}}, {_id: 0})
{ "a" : [ { "b" : [ 5 ] } ] }
```

```
{ "a" : [ [ { "b" : [ 5 ] } ] ] }
```

- Amazon DocumentDB 4.0의 "$" 프로젝션은 모든 필드가 포함된 모든 문서를 반환합니다. Amazon DocumentDB 5.0에서는 "$" 프로젝션이 있는 find 명령은 "$" 프로젝션과 일치하는 필드만 포함된 쿼리 매개 변수와 일치하는 문서를 반환합니다.
- Amazon DocumentDB 5.0에서 `$regex` 및 `$options` 쿼리 매개 변수가 있는 find 명령은 "`$regex` 및 `$options` 모두에서 옵션을 설정할 수 없습니다"라는 오류를 반환합니다.
- Amazon DocumentDB 5.0에서 `$indexOfCP`는 이제 다음과 같은 경우에 "-1"을 반환합니다.
  - 문자열 표현식에서 하위 문자열을 찾을 수 없음, 또는
  - 시작이 끝보다 큰 숫자임, 또는
  - 시작이 문자열의 바이트 길이보다 큰 숫자임.
- Amazon DocumentDB 4.0에서 `$indexOfCP`는 시작 위치가 끝 또는 문자열의 바이트 길이보다 큰 경우 "0"을 반환합니다.
- Amazon DocumentDB 5.0에서 `_id fields`의 프로젝션 연산(예: `{"_id.nestedField" : 1}`)은 프로젝션 필드만 포함하는 문서를 반환합니다. 단, Amazon DocumentDB 4.0에서는 중첩 필드 프로젝션 명령이 문서를 필터링하지 않습니다.

MongoDB 4.0 호환성

주제

- Amazon DocumentDB 4.0 특성
- Amazon DocumentDB 4.0 시작하기
- Amazon DocumentDB 4.0으로 업그레이드 또는 마이그레이션
- 기능적 차이

Amazon DocumentDB 4.0 특성

Amazon DocumentDB 4.0에는 ACID 트랜잭션과 스트림 변경을 위한 개선 사항을 포함한 많은 새로운 기능과 특성이 도입되었습니다. 아래 요약은 Amazon DocumentDB 4.0에서 소개된 주요 기능 중 일부 특성을 보여줍니다. 기능의 전체 목록을 보려면 릴리스 정보 섹션을 참조하세요.

- ACID 트랜잭션: Amazon DocumentDB는 이제 여러 문서, 명령문, 컬렉션 및 데이터베이스에서 트랜잭션을 수행하는 기능을 지원합니다. 트랜잭션은 Amazon DocumentDB 클러스터 내의 하나 이상의 문서에서 원자적이고 일관적이며 격리되고 내구성이 뛰어난(ACID) 작업을 수행할 수 있도록 하

여 애플리케이션 개발을 단순화합니다. 자세한 내용은 Amazon DocumentDB의 트랜잭션 섹션을 참조하십시오.

- 변경 스트림: 이제 클러스터 수준(`client.watch()` 또는 `mongo.watch()`)과 데이터베이스(`db.watch()`)에서 변경 스트림을 열 수 있고, 변경 스트림 커서를 열도록 `startAtOperationTime`을 지정할 수 있으며, 마지막으로 변경 스트림 보존 기간을 7일(이전에는 24시간)까지 연장할 수 있습니다. 자세한 내용은 Amazon DocumentDB에서 변경 스트림 사용 단원을 참조하십시오.
- AWS Database Migration Service (AWS DMS): 이제 AWS DMS 를 사용하여 MongoDB 4.0 워크로드를 Amazon DocumentDB로 마이그레이션할 수 있습니다. AWS DMS 이제는 Amazon DocumentDB 3.6과 4.0 간의 업그레이드를 수행하기 위해 MongoDB 4.0 소스, Amazon DocumentDB 4.0 대상 및 Amazon DocumentDB 3.6 소스를 지원합니다. Amazon DocumentDB 자세한 내용은 AWS DMS 설명서를 참조하세요.
- 성능 및 인덱싱: 이제 `$lookup`을 통한 인덱스를 활용하고, 하나의 필드 또는 하나의 필드와 `_id` 필드(컬렉션에서 읽을 필요 없이 인덱스에서 직접 제공 가능(가려진 쿼리))를 포함하는 프로젝션이 있는 쿼리를 찾을 수 있으며, `findAndModify`와 함께 `hint()`를 실행할 수 있고, `$addToSet`에 대한 성능 최적화 및 전체 인덱스 크기를 줄일 수 있는 개선 사항이 제공됩니다. 자세한 내용은 릴리스 정보 섹션을 참조하십시오.
- 연산자: Amazon DocumentDB 4.0은 이제 다양한 다음과 같은 새로운 집계 연산자를 지원합니다: `$ifNull`, `$replaceRoot`, `$setIsSubset`, `$setIntersection`, `$setUnion`, `$setEquals`. Amazon DocumentDB에서 지원되는 MongoDB API, 작업 및 데이터 형식에서 지원하는 모든 MongoDB API, 작업 및 데이터 유형을 볼 수 있습니다.
- 역할 기반 액세스 제어(RBAC): 이제 `ListCollection` 및 `ListDatabase` 명령을 모두 사용하여 `authorizedCollections` 및 `authorizedDatabases` 파라미터를 선택적으로 사용하여 사용자가 `listCollections` 및 `listDatabase` 역할을 필요로 하지 않고 액세스 권한이 있는 컬렉션 및 데이터베이스를 나열할 수 있습니다. 또한 `KillCursor` 역할 필요 없이 자신의 커서를 종료할 수 있습니다.

Amazon DocumentDB는 일부 MongoDB 4.0 기능을 지원하지 않습니다. Amazon DocumentDB 4.0을 구축할 때 고객이 가장 많이 구축하도록 요구했던 기능과 특성을 거꾸로 수행했습니다. 앞으로도 고객들이 구축을 요청하는 내용을 바탕으로 MongoDB 4.0 기능을 추가적으로 추가할 예정입니다. 예를 들어, Amazon DocumentDB 4.0은 현재 MongoDB 4.0에 도입된 유형 변환 연산자나 문자열 연산자를 지원하지 않습니다. 지원되는 API의 최신 목록은 Amazon DocumentDB에서 지원되는 MongoDB API, 작업 및 데이터 형식 섹션을 참조하십시오.

Amazon DocumentDB 4.0 시작하기

Amazon DocumentDB 4.0을 시작하려면 시작 가이드를 참조하세요. AWS Management Console 또는 AWS SDK AWS CLI또는를 사용하여 새 Amazon DocumentDB 4.0 클러스터를 생성할 수 있습니다 AWS CloudFormation. Amazon DocumentDB에 연결할 경우 MongoDB 4.0 이상과 호환되는 MongoDB 드라이버 또는 유틸리티를 사용해야 합니다.

> **Note**
>
> AWS SDK AWS CLI또는를 사용하는 경우 AWS CloudFormation엔진 버전은 기본적으로 5.0.0으로 설정됩니다. 새 Amazon DocumentDB 4.0 클러스터 또는 `engineVersion = 3.6.0`를 생성하거나 새 Amazon DocumentDB 3.6 클러스터를 생성하려면 `engineVersion = 4.0.0` 파라미터를 명시적으로 지정해야 합니다. 지정된 Amazon DocumentDB 클러스터의 경우를 사용하여 AWS CLI 를 호출`describe-db-clusters`하거나 Amazon DocumentDB 관리 콘솔을 사용하여 특정 클러스터의 엔진 버전 번호를 볼 수 있습니다.

Amazon DocumentDB 4.0은 클러스터에 대해 `r5`, `r6g`, `t3.medium` 및 `t4g.medium` 인스턴스 유형을 지원하며 지원되는 모든 영역에서 사용할 수 있습니다. Amazon DocumentDB 4.0 사용에 대한 추가 요금은 없습니다. 요금에 대한 자세한 내용은 Amazon DocumentDB(MongoDB 호환) 요금을 참조하십시오.

Amazon DocumentDB 4.0으로 업그레이드 또는 마이그레이션

AWS DMS 또는 `mongodump`, `mongorestore`, `mongoimport`, `mongoexport` 등의 유틸리티를 사용하여 MongoDB 3.6 또는 MongoDB 4.0에서 Amazon DocumentDB 4.0으로 마이그레이션할 수 있습니다. 마찬가지로 동일한 도구를 사용하여 Amazon DocumentDB 3.6에서 Amazon DocumentDB 4.0으로 업그레이드할 수 있습니다. 마이그레이션 방법에 대한 지침은 를 사용하여 Amazon DocumentDB 클러스터 업그레이드 AWS Database Migration Service 섹션을 참조하십시오.

기능적 차이

Amazon DocumentDB 3.6 및 4.0 간의 기능적 차이

Amazon DocumentDB 4.0의 릴리스에서는 Amazon DocumentDB 3.6 과 Amazon DocumentDB 4.0 간에 기능적 차이가 있습니다:

- 중첩 문서에 대한 프로젝션: Amazon DocumentDB 3.6은 프로젝션을 적용할 때 중첩된 문서의 첫 번째 필드를 고려합니다. 그러나 Amazon DocumentDB 4.0은 하위 문서를 구문 분석하고 각 하위 문서

에도 프로젝션을 적용합니다. 예를 들어"a.b.c": 1, 프로젝션이 다음과 같으면 두 버전의 동작은 동일합니다. 하지만 프로젝션이 프로젝션인 {a:{b:{c:1}}} 경우 Amazon DocumentDB 3.6은 프로젝션을 'a'에만 적용하고 'b' 또는 'c'에는 적용하지 않습니다.

- **minKey**, **maxKey**에 대한 동작: Amazon DocumentDB 4.0에서 {x:{$gt:MaxKey}}에 대한 동작은 아무것도 반환하지 않으며, {x:{$lt:MaxKey}}에 대한 동작은 모든 것을 반환합니다.
- 문서 비교 차이: 하위 문서(예: {"_id" :1, "a" :{"b":1}}의 b)에서 서로 다른 유형(double, int, long)의 수치를 비교함으로써, 이제 수치 데이터 유형과 문서의 각 수준에 걸쳐 일관된 출력이 제공됩니다.

Amazon DocumentDB 4.0 및 MongoDB 4.0 간의 기능적 차이

다음은 Amazon DocumentDB 4.0 및 MongoDB 4.0 간의 기능적 차이입니다.

- 경로에서 빈 키로 조회: 컬렉션에 배열 내에 빈 컨테이너가 있는 문서가 포함되어 있고(예:{"x" : [{ "" : 10 }, { "b" : 20 }]}) 쿼리에 사용된 키가 빈 문자열로 끝나는 경우(예:x.), Amazon DocumentDB는 배열의 모든 문서를 순회하므로 해당 문서를 반환하지만 MongoDB는 해당 문서를 반환하지 않습니다.
- 경로에서 **$setOnInsert** 및 **$** 함께: $setOnInsert 필드 연산자는 Amazon DocumentDB의 경로에서 $와 함께 작동하지 않으며, 이는 MongoDB 4.0과도 일치합니다.

Amazon DocumentDB의 트랜잭션

Amazon DocumentDB(MongoDB 호환)은 이제 트랜잭션을 포함한 MongoDB 4.0 호환성을 지원합니다. 여러 문서, 문, 컬렉션 및 데이터베이스에서 트랜잭션을 수행할 수 있습니다. 트랜잭션은 Amazon DocumentDB 클러스터 내의 하나 이상의 문서에서 원자적이고 일관적이며 격리되고 내구성이 뛰어난(ACID) 작업을 수행할 수 있도록 하여 애플리케이션 개발을 단순화합니다. 트랜잭션의 일반적인 사용 사례로는 재무 처리, 주문 이행 및 관리, 멀티플레이어 게임 구축 등이 있습니다.

트랜잭션을 활성화하는 데 추가 비용이 들지 않습니다. 트랜잭션의 일부로 사용한 읽기 및 쓰기 IO에 대한 비용만 지불하면 됩니다.

주제

- 요구 사항
- 모범 사례
- 제한 사항
- 모니터링 및 진단
- 트랜잭션 격리 수준
- 사용 사례
- 지원되는 명령
- 지원되지 않는 기능
- 세션
- 트랜잭션 오류

요구 사항

트랜잭션을 사용하려면 다음과 같은 요구 사항을 충족해야 합니다.

- Amazon DocumentDB 4.0 엔진을 사용하고 있어야 합니다.
- MongoDB 4.0 이상과 호환되는 드라이버를 사용해야 합니다.

모범 사례

다음은 Amazon DocumentDB를 통한 트랜잭션을 최대한 활용할 수 있는 몇 가지 모범 사례입니다.

- 트랜잭션이 완료된 후에는 항상 트랜잭션을 커밋하거나 중단하십시오. 트랜잭션을 불완전한 상태로 두면 데이터베이스 리소스가 묶이고 쓰기 충돌이 발생할 수 있습니다.
- 필요한 명령 수를 최소한으로 줄여 트랜잭션을 유지하는 것이 좋습니다. 여러 개의 작은 트랜잭션으로 나눌 수 있는 여러 개의 명령문이 포함된 트랜잭션이 있는 경우, 시간 초과가 발생할 가능성을 줄이기 위해 여러 개의 작은 트랜잭션으로 나누는 것이 좋습니다. 항상 오래 실행되는 읽기가 아닌 짧은 트랜잭션을 생성하는 것을 목표로 하십시오.

제한 사항

- Amazon DocumentDB는 트랜잭션 내에서 커서를 지원하지 않습니다.
- Amazon DocumentDB는 트랜잭션에서 새 컬렉션을 생성할 수 없으며 존재하지 않는 컬렉션에 대해 쿼리/업데이트할 수 없습니다.
- 문서 수준 쓰기 잠금에는 사용자가 구성할 수 없는 1분의 제한 시간이 적용됩니다.
- Amazon DocumentDB에서는 재시도 가능한 쓰기, 재시도 가능한 커밋 및 재시도 가능한 중단 명령이 지원되지 않습니다. mongo 쉘(mongosh 제외)을 사용하는 경우 코드 문자열에 `retryWrites=false` 명령을 포함시키지 마십시오. 기본적으로 재시도 가능한 쓰기는 비활성화되어 있습니다. `retryWrites=false`를 포함하면 일반 읽기 명령에서 오류가 발생할 수 있습니다.
- 각 Amazon DocumentDB 인스턴스에는 인스턴스에서 동시에 열리는 동시 트랜잭션 수에 대한 상한 제한이 있습니다. 제한에 대한 내용은 인스턴스 제한을 참조하십시오.
- 특정 트랜잭션의 트랜잭션 로그 크기는 32MB 미만이어야 합니다.
- Amazon DocumentDB는 트랜잭션 내에서 `count()`을 지원하지만 모든 드라이버가 이 기능을 지원하는 것은 아닙니다. 대안은 개수 쿼리를 클라이언트 측의 집계 쿼리로 변환하는 `countDocuments()` API를 사용하는 것입니다.
- 트랜잭션의 실행 제한은 1분이고 세션의 제한 시간은 30분입니다. 트랜잭션 제한 시간이 초과되면 트랜잭션은 중단되며 기존 트랜잭션에 대해 세션 내에서 후속 명령을 실행하면 다음 오류가 발생합니다.

```
WriteCommandError({
"ok" : 0,
"operationTime" : Timestamp(1603491424, 627726),
"code" : 251,
"errmsg" : "Given transaction number 0 does not match any in-progress transactions."
}
```

모니터링 및 진단

Amazon DocumentDB 4.0의 트랜잭션 지원과 함께, 트랜잭션을 모니터링하는 데 도움이 되는 추가 CloudWatch 지표가 추가되었습니다.

새로운 CloudWatch 지표

- `DatabaseTransactions`: 1분 동안 수행된 열린 트랜잭션 수입니다.
- `DatabaseTransactionsAborted`: 1분 동안 수행된 중단된 트랜잭션 수입니다.
- `DatabaseTransactionsMax`: 1분 동안 수행된 열린 최대 트랜잭션 수입니다.
- `TransactionsAborted`: 1분 동안 인스턴스에 수행된 중단된 트랜잭션 수입니다.
- `TransactionsCommitted`: 1분 동안 인스턴스에 수행된 커밋된 트랜잭션 수입니다.
- `TransactionsOpen`: 1분 동안 인스턴스에 수행된 열린 트랜잭션 수입니다.
- `TransactionsOpenMax`: 1분 동안 인스턴스에 수행된 열린 최대 트랜잭션 수입니다.
- `TransactionsStarted`: 1분 동안 인스턴스에 수행된 시작된 트랜잭션 수입니다.

> **Note**
>
> Amazon DocumentDB에 대한 더 많은 CloudWatch 지표를 보려면 CloudWatch를 사용하여 Amazon DocumentDB 모니터링로 이동하십시오.

또한 두 가지 `currentOp` `lsid`, `transactionThreadId` 모두에 새 필드가 추가되었고 , “`idle transaction`” 및 `serverStatus` 트랜잭션에 대한 새 상태: `currentActive`, `currentInactive`, `currentOpen`, `totalAborted`, `totalCommitted` 및 `totalStarted`가 추가되었습니다.

트랜잭션 격리 수준

트랜잭션을 시작할 때 아래 예제`writeConcern`와 같이 `readConcern` 및를 모두 지정할 수 있습니다.

```
mySession.startTransaction({readConcern: {level: 'snapshot'}, writeConcern: {w: 'majority'}});
```

`readConcern`의 경우, Amazon DocumentDB는 기본적으로 스냅샷 격리를 지원합니다. 로컬, 사용 가능 또는 과반수의 `readConcern`이 지정된 경우 Amazon DocumentDB는 `readConcern` 레벨을 스냅

샷으로 업그레이드합니다. Amazon DocumentDB는 선형화 가능한 `readConcern`을 지원하지 않으므로 이러한 읽기 문제를 지정하면 오류가 발생합니다.

`writeConcern`의 경우, Amazon DocumentDB는 기본적으로 과반수를 지원하며 데이터 사본 4개가 AZ에 걸쳐 유지되면 쓰기 쿼럼이 달성됩니다. 더 낮은 `writeConcern`을 지정하면 Amazon DocumentDB가 `writeConcern`를 과반수로 업그레이드합니다. 또한 모든 Amazon DocumentDB 쓰기는 저널링되며 저널링을 비활성화할 수 없습니다.

사용 사례

이 섹션에서는 다중 문과 다중 컬렉션이라는 두 가지 트랜잭션 사용 사례를 살펴보겠습니다.

다중 문 트랜잭션

Amazon DocumentDB 트랜잭션은 다중 문이므로 명시적인 커밋 또는 롤백을 통해 여러 문에 걸친 트랜잭션을 작성할 수 있습니다. `insert`, `update`, `delete` 및 `findAndModify` 작업을 단일 원자 연산으로 그룹화할 수 있습니다.

다중 문 트랜잭션의 일반적인 사용 사례는 직불 크레딧 트랜잭션입니다. 예를 들어, 친구에게 옷을 산 돈을 빚지고 있습니다. 따라서 귀하의 계좌에서 500달러를 인출(출금)하고 친구의 계좌에 500달러를 입금(예치금)해야 합니다. 이 작업을 수행하려면 단일 트랜잭션 내에서 출금 및 입금 작업을 모두 수행하여 원자성을 보장해야 합니다. 이렇게 하면 사용자 계정에서 500달러가 차감되지만 친구의 계좌에는 입금되지 않는 상황을 방지할 수 있습니다. 이 사용 사례는 다음과 같습니다.

```
// *** Transfer $500 from Alice to Bob inside a transaction: Success Scenario***
// Setup bank account for Alice and Bob. Each have $1000 in their account

var databaseName = "bank";
var collectionName = "account";
var amountToTransfer = 500;

var session = db.getMongo().startSession({causalConsistency: false});
var bankDB = session.getDatabase(databaseName);
var accountColl = bankDB[collectionName];
accountColl.drop();

accountColl.insert({name: "Alice", balance: 1000});
accountColl.insert({name: "Bob", balance: 1000});

session.startTransaction();
```

```
// deduct $500 from Alice's account
var aliceBalance = accountColl.find({"name": "Alice"}).next().balance;
var newAliceBalance = aliceBalance - amountToTransfer;
accountColl.update({"name": "Alice"},{"$set": {"balance": newAliceBalance}});
var findAliceBalance = accountColl.find({"name": "Alice"}).next().balance;

// add $500 to Bob's account
var bobBalance = accountColl.find({"name": "Bob"}).next().balance;
var newBobBalance = bobBalance + amountToTransfer;
accountColl.update({"name": "Bob"},{"$set": {"balance": newBobBalance}});
var findBobBalance = accountColl.find({"name": "Bob"}).next().balance;

session.commitTransaction();

accountColl.find();

// *** Transfer $500 from Alice to Bob inside a transaction: Failure Scenario***

// Setup bank account for Alice and Bob. Each have $1000 in their account
var databaseName = "bank";
var collectionName = "account";
var amountToTransfer = 500;

var session = db.getMongo().startSession({causalConsistency: false});
var bankDB = session.getDatabase(databaseName);
var accountColl = bankDB[collectionName];
accountColl.drop();

accountColl.insert({name: "Alice", balance: 1000});
accountColl.insert({name: "Bob", balance: 1000});

session.startTransaction();

// deduct $500 from Alice's account
var aliceBalance = accountColl.find({"name": "Alice"}).next().balance;
var newAliceBalance = aliceBalance - amountToTransfer;
accountColl.update({"name": "Alice"},{"$set": {"balance": newAliceBalance}});
var findAliceBalance = accountColl.find({"name": "Alice"}).next().balance;

session.abortTransaction();
```

다중 컬렉션 트랜잭션

또한 당사의 트랜잭션은 다중 컬렉션이므로 단일 트랜잭션 내에서 여러 컬렉션에 걸쳐 여러 작업을 수행하는 데 사용될 수 있습니다. 이를 통해 일관된 데이터 보기를 제공하고 데이터 무결성을 유지할 수 있습니다. 명령을 한 번의 <>로 커밋하면 트랜잭션은 모두 실행되거나 아예 실행되지 않습니다. 즉, 트랜잭션은 모두 성공하거나 모두 실패합니다.

다음은 다중 문 트랜잭션에 예제의 동일한 시나리오와 데이터를 사용하는 다중 컬렉션 트랜잭션의 예입니다.

```
// *** Transfer $500 from Alice to Bob inside a transaction: Success Scenario***

// Setup bank account for Alice and Bob. Each have $1000 in their account
var amountToTransfer = 500;
var collectionName = "account";

var session = db.getMongo().startSession({causalConsistency: false});
var accountCollInBankA = session.getDatabase("bankA")[collectionName];
var accountCollInBankB = session.getDatabase("bankB")[collectionName];

accountCollInBankA.drop();
accountCollInBankB.drop();

accountCollInBankA.insert({name: "Alice", balance: 1000});
accountCollInBankB.insert({name: "Bob", balance: 1000});

session.startTransaction();

// deduct $500 from Alice's account
var aliceBalance = accountCollInBankA.find({"name": "Alice"}).next().balance;
var newAliceBalance = aliceBalance - amountToTransfer;
accountCollInBankA.update({"name": "Alice"},{"$set": {"balance": newAliceBalance}});
var findAliceBalance = accountCollInBankA.find({"name": "Alice"}).next().balance;

// add $500 to Bob's account
var bobBalance = accountCollInBankB.find({"name": "Bob"}).next().balance;
var newBobBalance = bobBalance + amountToTransfer;
accountCollInBankB.update({"name": "Bob"},{"$set": {"balance": newBobBalance}});
var findBobBalance = accountCollInBankB.find({"name": "Bob"}).next().balance;

session.commitTransaction();
```

```
accountCollInBankA.find(); // Alice holds $500 in bankA
accountCollInBankB.find(); // Bob holds $1500 in bankB

// *** Transfer $500 from Alice to Bob inside a transaction: Failure Scenario***

// Setup bank account for Alice and Bob. Each have $1000 in their account
var collectionName = "account";
var amountToTransfer = 500;

var session = db.getMongo().startSession({causalConsistency: false});
var accountCollInBankA = session.getDatabase("bankA")[collectionName];
var accountCollInBankB = session.getDatabase("bankB")[collectionName];

accountCollInBankA.drop();
accountCollInBankB.drop();

accountCollInBankA.insert({name: "Alice", balance: 1000});
accountCollInBankB.insert({name: "Bob", balance: 1000});

session.startTransaction();

// deduct $500 from Alice's account
var aliceBalance = accountCollInBankA.find({"name": "Alice"}).next().balance;
var newAliceBalance = aliceBalance - amountToTransfer;
accountCollInBankA.update({"name": "Alice"},{"$set": {"balance": newAliceBalance}});
var findAliceBalance = accountCollInBankA.find({"name": "Alice"}).next().balance;

// add $500 to Bob's account
var bobBalance = accountCollInBankB.find({"name": "Bob"}).next().balance;
var newBobBalance = bobBalance + amountToTransfer;
accountCollInBankB.update({"name": "Bob"},{"$set": {"balance": newBobBalance}});
var findBobBalance = accountCollInBankB.find({"name": "Bob"}).next().balance;

session.abortTransaction();

accountCollInBankA.find(); // Alice holds $1000 in bankA
accountCollInBankB.find(); // Bob holds $1000 in bankB
```

콜백 API의 트랜잭션 API 예제

콜백 API는 4.2+ 드라이버에만 사용할 수 있습니다.

Javascript

다음 코드는 Amazon DocumentDB 트랜잭션 API를 자바스크립트와 함께 활용하는 방법을 보여줍니다.

```
// *** Transfer $500 from Alice to Bob inside a transaction: Success ***
// Setup bank account for Alice and Bob. Each have $1000 in their account
var databaseName = "bank";
var collectionName = "account";
var amountToTransfer = 500;

var session = db.getMongo().startSession({causalConsistency: false});
var bankDB = session.getDatabase(databaseName);
var accountColl = bankDB[collectionName];
accountColl.drop();

accountColl.insert({name: "Alice", balance: 1000});
accountColl.insert({name: "Bob", balance: 1000});

session.startTransaction();

// deduct $500 from Alice's account
var aliceBalance = accountColl.find({"name": "Alice"}).next().balance;
assert(aliceBalance >= amountToTransfer);
var newAliceBalance = aliceBalance - amountToTransfer;
accountColl.update({"name": "Alice"},{"$set": {"balance": newAliceBalance}});
var findAliceBalance = accountColl.find({"name": "Alice"}).next().balance;
assert.eq(newAliceBalance, findAliceBalance);

// add $500 to Bob's account
var bobBalance = accountColl.find({"name": "Bob"}).next().balance;
var newBobBalance = bobBalance + amountToTransfer;
accountColl.update({"name": "Bob"},{"$set": {"balance": newBobBalance}});
var findBobBalance = accountColl.find({"name": "Bob"}).next().balance;
assert.eq(newBobBalance, findBobBalance);

session.commitTransaction();

accountColl.find();
```

Node.js

다음 코드는 Amazon DocumentDB 트랜잭션 API를 Node.js와 함께 활용하는 방법을 보여줍니다.

```
// Node.js callback API:

const bankDB = await mongoclient.db("bank");
var accountColl = await bankDB.createCollection("account");
var amountToTransfer = 500;

const session = mongoclient.startSession({causalConsistency: false});
await accountColl.drop();

await accountColl.insertOne({name: "Alice", balance: 1000}, { session });
await accountColl.insertOne({name: "Bob", balance: 1000}, { session });

const transactionOptions = {
    readConcern: { level: 'snapshot' },
    writeConcern: { w: 'majority' }
    };

// deduct $500 from Alice's account
var aliceBalance = await accountColl.findOne({name: "Alice"}, {session});
assert(aliceBalance.balance >= amountToTransfer);
var newAliceBalance = aliceBalance - amountToTransfer;
session.startTransaction(transactionOptions);
await accountColl.updateOne({name: "Alice"}, {$set: {balance: newAliceBalance}},
 {session });
await session.commitTransaction();
aliceBalance = await accountColl.findOne({name: "Alice"}, {session});
assert(newAliceBalance == aliceBalance.balance);

// add $500 to Bob's account
var bobBalance = await accountColl.findOne({name: "Bob"}, {session});
var newBobBalance = bobBalance.balance + amountToTransfer;
session.startTransaction(transactionOptions);
await accountColl.updateOne({name: "Bob"}, {$set: {balance: newBobBalance}},
 {session });
await session.commitTransaction();
bobBalance = await accountColl.findOne({name: "Bob"}, {session});
assert(newBobBalance == bobBalance.balance);
```

C#

다음 코드는 Amazon DocumentDB 트랜잭션 API를 C#와 함께 활용하는 방법을 보여줍니다.

```
// C# Callback API
```

```
var dbName = "bank";
var collName = "account";
var amountToTransfer = 500;

using (var session = client.StartSession(new ClientSessionOptions{CausalConsistency
 = false}))
{
    var bankDB = client.GetDatabase(dbName);
    var accountColl = bankDB.GetCollection<BsonDocument>(collName);
    bankDB.DropCollection(collName);
    accountColl.InsertOne(session, new BsonDocument { {"name", "Alice"}, {"balance",
 1000 } });
    accountColl.InsertOne(session, new BsonDocument { {"name", "Bob"}, {"balance",
 1000 } });

    // start transaction
    var transactionOptions = new TransactionOptions(
            readConcern: ReadConcern.Snapshot,
            writeConcern: WriteConcern.WMajority);
    var result = session.WithTransaction(
        (sess, cancellationtoken) =>
        {
            // deduct $500 from Alice's account
            var aliceBalance = accountColl.Find(sess,
 Builders<BsonDocument>.Filter.Eq("name",
 "Alice")).FirstOrDefault().GetValue("balance");
            Debug.Assert(aliceBalance >= amountToTransfer);
            var newAliceBalance = aliceBalance.AsInt32 - amountToTransfer;
            accountColl.UpdateOne(sess, Builders<BsonDocument>.Filter.Eq("name",
 "Alice"),
                                      Builders<BsonDocument>.Update.Set("balance",
 newAliceBalance));
            aliceBalance = accountColl.Find(sess,
 Builders<BsonDocument>.Filter.Eq("name",
 "Alice")).FirstOrDefault().GetValue("balance");
            Debug.Assert(aliceBalance == newAliceBalance);

            // add $500 from Bob's account
            var bobBalance = accountColl.Find(sess,
 Builders<BsonDocument>.Filter.Eq("name",
 "Bob")).FirstOrDefault().GetValue("balance");
            var newBobBalance = bobBalance.AsInt32 + amountToTransfer;
```

```
            accountColl.UpdateOne(sess, Builders<BsonDocument>.Filter.Eq("name",
 "Bob"),
                                        Builders<BsonDocument>.Update.Set("balance",
 newBobBalance));
            bobBalance = accountColl.Find(sess,
 Builders<BsonDocument>.Filter.Eq("name",
 "Bob")).FirstOrDefault().GetValue("balance");
            Debug.Assert(bobBalance == newBobBalance);

            return "Transaction committed";
        }, transactionOptions);
  // check values outside of transaction
    var aliceNewBalance = accountColl.Find(Builders<BsonDocument>.Filter.Eq("name",
 "Alice")).FirstOrDefault().GetValue("balance");
    var bobNewBalance = accountColl.Find(Builders<BsonDocument>.Filter.Eq("name",
 "Bob")).FirstOrDefault().GetValue("balance");
    Debug.Assert(aliceNewBalance ==  500);
    Debug.Assert(bobNewBalance ==  1500);
}
```

Ruby

다음 코드는 Amazon DocumentDB 트랜잭션 API를 Ruby와 함께 활용하는 방법을 보여줍니다.

```
// Ruby Callback API

dbName = "bank"
collName = "account"
amountToTransfer = 500

session = client.start_session(:causal_consistency=> false)
bankDB = Mongo::Database.new(client, dbName)
accountColl = bankDB[collName]
accountColl.drop()

accountColl.insert_one({"name"=>"Alice", "balance"=>1000})
accountColl.insert_one({"name"=>"Bob", "balance"=>1000})

    # start transaction
    session.with_transaction(read_concern: {level: :snapshot}, write_concern:
 {w: :majority}) do
        # deduct $500 from Alice's account
        aliceBalance = accountColl.find({"name"=>"Alice"}, :session=>
 session).first['balance']
```

```
        assert aliceBalance >= amountToTransfer
        newAliceBalance = aliceBalance - amountToTransfer
        accountColl.update_one({"name"=>"Alice"}, { "$set" =>
 {"balance"=>newAliceBalance} }, :session=> session)
        aliceBalance = accountColl.find({"name"=>>"Alice"}, :session=>
 session).first['balance']
        assert_equal(newAliceBalance, aliceBalance)

        # add $500 from Bob's account
        bobBalance = accountColl.find({"name"=>"Bob"}, :session=>
 session).first['balance']
        newBobBalance = bobBalance + amountToTransfer
        accountColl.update_one({"name"=>"Bob"}, { "$set" =>
 {"balance"=>newBobBalance} }, :session=> session)
        bobBalance = accountColl.find({"name"=>"Bob"}, :session=>
 session).first['balance']
        assert_equal(newBobBalance, bobBalance)
    end

   # check results outside of transaction
    aliceBalance = accountColl.find({"name"=>"Alice"}).first['balance']
    bobBalance = accountColl.find({"name"=>"Bob"}).first['balance']
    assert_equal(aliceBalance, 500)
    assert_equal(bobBalance, 1500)

session.end_session
```

Go

다음 코드는 Amazon DocumentDB 트랜잭션 API를 Go와 함께 활용하는 방법을 보여줍니다.

```
// Go - Callback API
type Account struct {
    Name string
    Balance  int
}

ctx := context.TODO()

dbName := "bank"
collName := "account"
amountToTransfer := 500

session, err := client.StartSession(options.Session().SetCausalConsistency(false))
```

```
assert.NilError(t, err)
defer session.EndSession(ctx)

bankDB := client.Database(dbName)
accountColl := bankDB.Collection(collName)
accountColl.Drop(ctx)

_, err = accountColl.InsertOne(ctx, bson.M{"name" : "Alice", "balance":1000})
_, err = accountColl.InsertOne(ctx, bson.M{"name" : "Bob", "balance":1000})

transactionOptions := options.Transaction().SetReadConcern(readconcern.Snapshot()).

 SetWriteConcern(writeconcern.New(writeconcern.WMajority()))
_, err = session.WithTransaction(ctx, func(sessionCtx mongo.SessionContext)
 (interface{}, error) {
    var result Account
    // deduct $500 from Alice's account
    err = accountColl.FindOne(sessionCtx, bson.M{"name": "Alice"}).Decode(&result)
    aliceBalance := result.Balance
    newAliceBalance := aliceBalance - amountToTransfer
    _, err = accountColl.UpdateOne(sessionCtx, bson.M{"name": "Alice"},
 bson.M{"$set": bson.M{"balance": newAliceBalance}})
    err = accountColl.FindOne(sessionCtx, bson.M{"name": "Alice"}).Decode(&result)
    aliceBalance = result.Balance
    assert.Equal(t, aliceBalance, newAliceBalance)

    // add $500 to Bob's account
    err = accountColl.FindOne(sessionCtx, bson.M{"name": "Bob"}).Decode(&result)
    bobBalance := result.Balance
    newBobBalance := bobBalance + amountToTransfer
    _, err = accountColl.UpdateOne(sessionCtx, bson.M{"name": "Bob"}, bson.M{"$set":
 bson.M{"balance": newBobBalance}})
    err = accountColl.FindOne(sessionCtx, bson.M{"name": "Bob"}).Decode(&result)
    bobBalance = result.Balance
    assert.Equal(t, bobBalance, newBobBalance)

    if err != nil {
        return nil, err
    }
    return "transaction committed", err
}, transactionOptions)

// check results outside of transaction
var result Account
```

```
err = accountColl.FindOne(ctx, bson.M{"name": "Alice"}).Decode(&result)
aliceNewBalance := result.Balance
err = accountColl.FindOne(ctx, bson.M{"name": "Bob"}).Decode(&result)
bobNewBalance := result.Balance
assert.Equal(t, aliceNewBalance, 500)
assert.Equal(t, bobNewBalance, 1500)
```

Java

다음 코드는 Amazon DocumentDB 트랜잭션 API를 Java와 함께 활용하는 방법을 보여줍니다.

```
// Java (sync) - Callback API
MongoDatabase bankDB = mongoClient.getDatabase("bank");
MongoCollection accountColl = bankDB.getCollection("account");
accountColl.drop();
int amountToTransfer = 500;

// add sample data
accountColl.insertOne(new Document("name", "Alice").append("balance", 1000));
accountColl.insertOne(new Document("name", "Bob").append("balance", 1000));

TransactionOptions txnOptions = TransactionOptions.builder()
        .readConcern(ReadConcern.SNAPSHOT)
        .writeConcern(WriteConcern.MAJORITY)
        .build();
ClientSessionOptions sessionOptions =
 ClientSessionOptions.builder().causallyConsistent(false).build();
try ( ClientSession clientSession = mongoClient.startSession(sessionOptions) ) {
    clientSession.withTransaction(new TransactionBody<Void>() {
        @Override
        public Void execute() {
            // deduct $500 from Alice's account
            List<Document> documentList = new ArrayList<>();
            accountColl.find(clientSession, new Document("name",
 "Alice")).into(documentList);
            int aliceBalance = (int) documentList.get(0).get("balance");
            int newAliceBalance = aliceBalance - amountToTransfer;

            accountColl.updateOne(clientSession, new Document("name", "Alice"), new
 Document("$set", new Document("balance", newAliceBalance)));

            // check Alice's new balance
            documentList = new ArrayList<>();
```

```
            accountColl.find(clientSession, new Document("name",
 "Alice")).into(documentList);
            int updatedBalance = (int) documentList.get(0).get("balance");
            Assert.assertEquals(updatedBalance, newAliceBalance);

            // add $500 to Bob's account
            documentList = new ArrayList<>();
            accountColl.find(clientSession, new Document("name",
 "Bob")).into(documentList);
            int bobBalance = (int) documentList.get(0).get("balance");
            int newBobBalance = bobBalance + amountToTransfer;

            accountColl.updateOne(clientSession, new Document("name", "Bob"), new
 Document("$set", new Document("balance", newBobBalance)));

            // check Bob's new balance
            documentList = new ArrayList<>();
            accountColl.find(clientSession, new Document("name",
 "Bob")).into(documentList);
            updatedBalance = (int) documentList.get(0).get("balance");
            Assert.assertEquals(updatedBalance, newBobBalance);

            return null;
        }
    }, txnOptions);
}
```

C

다음 코드는 Amazon DocumentDB 트랜잭션 API를 C와 함께 활용하는 방법을 보여줍니다.

```
// Sample Code for C with Callback

#include <bson.h>
#include <mongoc.h>
#include <stdio.h>
#include <string.h>
#include <assert.h>

typedef struct {
    int64_t balance;
    bson_t *account;
    bson_t *opts;
    mongoc_collection_t *collection;
```

```
} ctx_t;

bool callback_session (mongoc_client_session_t *session, void *ctx, bson_t **reply,
 bson_error_t *error)
{
    bool r = true;
    ctx_t *data = (ctx_t *) ctx;
    bson_t local_reply;
    bson_t *selector = data->account;
    bson_t *update = BCON_NEW ("$set", "{", "balance", BCON_INT64 (data->balance),
 "}");

    mongoc_collection_update_one (data->collection, selector, update, data->opts,
 &local_reply, error);

    *reply = bson_copy (&local_reply);
    bson_destroy (&local_reply);
    bson_destroy (update);
    return r;
}

void test_callback_money_transfer(mongoc_client_t* client, mongoc_collection_t*
 collection, int amount_to_transfer){

    bson_t reply;
    bool r = true;
    const bson_t *doc;
    bson_iter_t iter;
    ctx_t alice_ctx;
    ctx_t bob_ctx;
    bson_error_t error;

    // find query
    bson_t *alice_query = bson_new ();
    BSON_APPEND_UTF8(alice_query, "name", "Alice");

    bson_t *bob_query = bson_new ();
    BSON_APPEND_UTF8(bob_query, "name", "Bob");

    // create session
    // set causal consistency to false
    mongoc_session_opt_t *session_opts = mongoc_session_opts_new ();
    mongoc_session_opts_set_causal_consistency (session_opts, false);
    // start the session
```

```
   mongoc_client_session_t *client_session = mongoc_client_start_session (client,
session_opts, &error);

   // add session to options
   bson_t *opts = bson_new();
   mongoc_client_session_append (client_session, opts, &error);

   // deduct 500 from Alice
   // find account balance of Alice
   mongoc_cursor_t *cursor = mongoc_collection_find_with_opts (collection,
alice_query, NULL, NULL);
   mongoc_cursor_next (cursor, &doc);
   bson_iter_init (&iter, doc);
   bson_iter_find (&iter, "balance");
   int64_t alice_balance = (bson_iter_value (&iter))->value.v_int64;
   assert(alice_balance >= amount_to_transfer);
   int64_t new_alice_balance = alice_balance - amount_to_transfer;

   // set variables which will be used by callback function
   alice_ctx.collection = collection;
   alice_ctx.opts = opts;
   alice_ctx.balance = new_alice_balance;
   alice_ctx.account = alice_query;

   // callback
   r = mongoc_client_session_with_transaction (client_session, &callback_session,
NULL, &alice_ctx, &reply, &error);
   assert(r);

   // find account balance of Alice after transaction
   cursor = mongoc_collection_find_with_opts (collection, alice_query, NULL, NULL);
   mongoc_cursor_next (cursor, &doc);
   bson_iter_init (&iter, doc);
   bson_iter_find (&iter, "balance");
   alice_balance = (bson_iter_value (&iter))->value.v_int64;
   assert(alice_balance == new_alice_balance);
   assert(alice_balance == 500);

       // add 500 to bob's balance
   // find account balance of Bob
   cursor = mongoc_collection_find_with_opts (collection, bob_query, NULL, NULL);
   mongoc_cursor_next (cursor, &doc);
   bson_iter_init (&iter, doc);
   bson_iter_find (&iter, "balance");
```

```
    int64_t bob_balance = (bson_iter_value (&iter))->value.v_int64;
    int64_t new_bob_balance = bob_balance + amount_to_transfer;

    bob_ctx.collection = collection;
    bob_ctx.opts = opts;
    bob_ctx.balance = new_bob_balance;
    bob_ctx.account = bob_query;

    // set read & write concern
    mongoc_read_concern_t *read_concern = mongoc_read_concern_new ();
    mongoc_write_concern_t *write_concern = mongoc_write_concern_new ();
    mongoc_transaction_opt_t *txn_opts = mongoc_transaction_opts_new ();

    mongoc_write_concern_set_w(write_concern, MONGOC_WRITE_CONCERN_W_MAJORITY);
    mongoc_read_concern_set_level(read_concern, MONGOC_READ_CONCERN_LEVEL_SNAPSHOT);
    mongoc_transaction_opts_set_write_concern (txn_opts, write_concern);
    mongoc_transaction_opts_set_read_concern (txn_opts, read_concern);

    // callback
    r = mongoc_client_session_with_transaction (client_session, &callback_session,
 txn_opts, &bob_ctx, &reply, &error);
    assert(r);

 // find account balance of Bob after transaction
    cursor = mongoc_collection_find_with_opts (collection, bob_query, NULL, NULL);
    mongoc_cursor_next (cursor, &doc);
    bson_iter_init (&iter, doc);
    bson_iter_find (&iter, "balance");
    bob_balance = (bson_iter_value (&iter))->value.v_int64;
    assert(bob_balance == new_bob_balance);
    assert(bob_balance == 1500);

    // cleanup
    bson_destroy(alice_query);
    bson_destroy(bob_query);
    mongoc_client_session_destroy(client_session);
    bson_destroy(opts);
    mongoc_transaction_opts_destroy(txn_opts);
    mongoc_read_concern_destroy(read_concern);
    mongoc_write_concern_destroy(write_concern);
    mongoc_cursor_destroy(cursor);
    bson_destroy(doc);
}
int main(int argc, char* argv[]) {
```

```
    mongoc_init ();
    mongoc_client_t* client = mongoc_client_new (<connection uri>);
    bson_error_t error;

    // connect to bank db
    mongoc_database_t *database = mongoc_client_get_database (client, "bank");
    // access account collection
    mongoc_collection_t* collection = mongoc_client_get_collection(client, "bank",
 "account");
    // set amount to transfer
    int64_t amount_to_transfer = 500;
    // delete the collection if already existing
    mongoc_collection_drop(collection, &error);

    // open Alice account
    bson_t *alice_account = bson_new ();
    BSON_APPEND_UTF8(alice_account, "name", "Alice");
    BSON_APPEND_INT64(alice_account, "balance", 1000);

    // open Bob account
    bson_t *bob_account = bson_new ();
    BSON_APPEND_UTF8(bob_account, "name", "Bob");
    BSON_APPEND_INT64(bob_account, "balance", 1000);

    bool r = true;

    r = mongoc_collection_insert_one(collection, alice_account, NULL, NULL, &error);
    if (!r) {printf("Error encountered:%s", error.message);}
    r = mongoc_collection_insert_one(collection, bob_account, NULL, NULL, &error);
    if (!r) {printf("Error encountered:%s", error.message);}

    test_callback_money_transfer(client, collection, amount_to_transfer);

}
```

Python

다음 코드는 Amazon DocumentDB 트랜잭션 API를 Python과 함께 활용하는 방법을 보여줍니다.

```
// Sample Python code with callback api

import pymongo

def callback(session, balance, query):
```

```
    collection.update_one(query, {'$set': {"balance": balance}}, session=session)

client = pymongo.MongoClient(<connection uri>)
rc_snapshot = pymongo.read_concern.ReadConcern('snapshot')
wc_majority = pymongo.write_concern.WriteConcern('majority')

# To start, drop and create an account collection and insert balances for both Alice
 and Bob
collection = client.get_database("bank").get_collection("account")
collection.drop()
collection.insert_one({"_id": 1, "name": "Alice", "balance": 1000})
collection.insert_one({"_id": 2, "name": "Bob", "balance": 1000})

amount_to_transfer = 500

# deduct 500 from Alice's account
alice_balance = collection.find_one({"name": "Alice"}).get("balance")
assert alice_balance >= amount_to_transfer
new_alice_balance = alice_balance - amount_to_transfer

with client.start_session({'causalConsistency':False}) as session:
    session.with_transaction(lambda s: callback(s, new_alice_balance, {"name":
 "Alice"}), read_concern=rc_snapshot, write_concern=wc_majority)

updated_alice_balance = collection.find_one({"name": "Alice"}).get("balance")
assert updated_alice_balance == new_alice_balance

# add 500 to Bob's account
bob_balance = collection.find_one({"name": "Bob"}).get("balance")
assert bob_balance >= amount_to_transfer
new_bob_balance = bob_balance + amount_to_transfer

with client.start_session({'causalConsistency':False}) as session:
    session.with_transaction(lambda s: callback(s, new_bob_balance, {"name":
 "Bob"}), read_concern=rc_snapshot, write_concern=wc_majority)

updated_bob_balance = collection.find_one({"name": "Bob"}).get("balance")
assert updated_bob_balance == new_bob_balance
```

코어 API의 트랜잭션 API 예제

Javascript

다음 코드는 Amazon DocumentDB 트랜잭션 API를 자바스크립트와 함께 활용하는 방법을 보여줍니다.

```
// *** Transfer $500 from Alice to Bob inside a transaction: Success ***
// Setup bank account for Alice and Bob. Each have $1000 in their account
var databaseName = "bank";
var collectionName = "account";
var amountToTransfer = 500;

var session = db.getMongo().startSession({causalConsistency: false});
var bankDB = session.getDatabase(databaseName);
var accountColl = bankDB[collectionName];
accountColl.drop();

accountColl.insert({name: "Alice", balance: 1000});
accountColl.insert({name: "Bob", balance: 1000});

session.startTransaction();

// deduct $500 from Alice's account
var aliceBalance = accountColl.find({"name": "Alice"}).next().balance;
assert(aliceBalance >= amountToTransfer);
var newAliceBalance = aliceBalance - amountToTransfer;
accountColl.update({"name": "Alice"},{"$set": {"balance": newAliceBalance}});
var findAliceBalance = accountColl.find({"name": "Alice"}).next().balance;
assert.eq(newAliceBalance, findAliceBalance);

// add $500 to Bob's account
var bobBalance = accountColl.find({"name": "Bob"}).next().balance;
var newBobBalance = bobBalance + amountToTransfer;
accountColl.update({"name": "Bob"},{"$set": {"balance": newBobBalance}});
var findBobBalance = accountColl.find({"name": "Bob"}).next().balance;
assert.eq(newBobBalance, findBobBalance);

session.commitTransaction();

accountColl.find();
```

C#

다음 코드는 Amazon DocumentDB 트랜잭션 API를 C#와 함께 활용하는 방법을 보여줍니다.

```
// C# Core API

public void TransferMoneyWithRetry(IMongoCollection<bSondocument> accountColl,
 IClientSessionHandle session)
{
    var amountToTransfer = 500;

    // start transaction
   var transactionOptions = new TransactionOptions(
                readConcern: ReadConcern.Snapshot,
                writeConcern: WriteConcern.WMajority);
    session.StartTransaction(transactionOptions);
   try
    {
        // deduct $500 from Alice's account
        var aliceBalance = accountColl.Find(session,
 Builders<bSondocument>.Filter.Eq("name",
 "Alice")).FirstOrDefault().GetValue("balance");
       Debug.Assert(aliceBalance >= amountToTransfer);
       var newAliceBalance = aliceBalance.AsInt32 - amountToTransfer;
       accountColl.UpdateOne(session, Builders<bSondocument>.Filter.Eq("name",
 "Alice"),
                                Builders<bSondocument>.Update.Set("balance",
 newAliceBalance));
       aliceBalance = accountColl.Find(session,
 Builders<bSondocument>.Filter.Eq("name",
 "Alice")).FirstOrDefault().GetValue("balance");
       Debug.Assert(aliceBalance == newAliceBalance);

       // add $500 from Bob's account
       var bobBalance = accountColl.Find(session,
 Builders<bSondocument>.Filter.Eq("name",
 "Bob")).FirstOrDefault().GetValue("balance");
       var newBobBalance = bobBalance.AsInt32 + amountToTransfer;
       accountColl.UpdateOne(session, Builders<bSondocument>.Filter.Eq("name",
 "Bob"),
                                Builders<bSondocument>.Update.Set("balance",
 newBobBalance));
```

```
        bobBalance = accountColl.Find(session,
 Builders<bSondocument>.Filter.Eq("name",
 "Bob")).FirstOrDefault().GetValue("balance");
        Debug.Assert(bobBalance == newBobBalance);

    }
    catch (Exception e)
    {
        session.AbortTransaction();
        throw;
    }

    session.CommitTransaction();
 }

}
public void DoTransactionWithRetry(MongoClient client)
{
    var dbName = "bank";
    var collName = "account";
    using (var session = client.StartSession(new
 ClientSessionOptions{CausalConsistency = false}))
    {
        try
        {
            var bankDB = client.GetDatabase(dbName);
            var accountColl = bankDB.GetCollection<bSondocument>(collName);
            bankDB.DropCollection(collName);
            accountColl.InsertOne(session, new BsonDocument { {"name", "Alice"},
 {"balance", 1000 } });
            accountColl.InsertOne(session, new BsonDocument { {"name", "Bob"},
 {"balance", 1000 } });

            while(true) {
                try
                {
                        TransferMoneyWithRetry(accountColl, session);
                        break;
                }
                catch (MongoException e)
                {
                    if(e.HasErrorLabel("TransientTransactionError"))
                    {
                        continue;
```

```
                    }
                    else
                    {
                        throw;
                    }
                }
            }

            // check values outside of transaction
            var aliceNewBalance =
 accountColl.Find(Builders<bSondocument>.Filter.Eq("name",
 "Alice")).FirstOrDefault().GetValue("balance");
            var bobNewBalance =
 accountColl.Find(Builders<bSondocument>.Filter.Eq("name",
 "Bob")).FirstOrDefault().GetValue("balance");
            Debug.Assert(aliceNewBalance ==  500);
            Debug.Assert(bobNewBalance ==  1500);
        }
        catch (Exception e)
        {
            Console.WriteLine("Error running transaction: " + e.Message);
        }
    }
}
```

Ruby

다음 코드는 Amazon DocumentDB 트랜잭션 API를 Ruby와 함께 활용하는 방법을 보여줍니다.

```
# Ruby Core API

def transfer_money_w_retry(session, accountColl)
    amountToTransfer = 500

    session.start_transaction(read_concern: {level: :snapshot}, write_concern:
 {w: :majority})
    # deduct $500 from Alice's account
    aliceBalance = accountColl.find({"name"=>"Alice"}, :session=>
 session).first['balance']
    assert aliceBalance >= amountToTransfer
    newAliceBalance = aliceBalance - amountToTransfer
    accountColl.update_one({"name"=>"Alice"}, { "$set" =>
 {"balance"=>newAliceBalance} }, :session=> session)
```

```
    aliceBalance = accountColl.find({"name"=>"Alice"}, :session=>
 session).first['balance']
    assert_equal(newAliceBalance, aliceBalance)

    # add $500 to Bob's account
    bobBalance = accountColl.find({"name"=>"Bob"}, :session=>
 session).first['balance']
    newBobBalance = bobBalance + amountToTransfer
    accountColl.update_one({"name"=>"Bob"}, { "$set" =>
 {"balance"=>newBobBalance} }, :session=> session)
    bobBalance = accountColl.find({"name"=>"Bob"}, :session=>
 session).first['balance']
    assert_equal(newBobBalance, bobBalance)

    session.commit_transaction

end

def do_txn_w_retry(client)
     dbName = "bank"
    collName = "account"

    session = client.start_session(:causal_consistency=> false)
    bankDB = Mongo::Database.new(client, dbName)
    accountColl = bankDB[collName]
    accountColl.drop()

    accountColl.insert_one({"name"=>"Alice", "balance"=>1000})
    accountColl.insert_one({"name"=>"Bob", "balance"=>1000})

    begin
        transferMoneyWithRetry(session, accountColl)
        puts "transaction committed"
    rescue Mongo::Error => e
        if e.label?('TransientTransactionError')
            retry
        else
            puts "transaction failed"
            raise
        end
    end

    # check results outside of transaction
    aliceBalance = accountColl.find({"name"=>"Alice"}).first['balance']
```

```
    bobBalance = accountColl.find({"name"=>"Bob"}).first['balance']
    assert_equal(aliceBalance, 500)
    assert_equal(bobBalance, 1500)

end
```

Go

다음 코드는 Amazon DocumentDB 트랜잭션 API를 Go와 함께 활용하는 방법을 보여줍니다.

```
// Go - Core API
type Account struct {
    Name string
    Balance  int
}

func transferMoneyWithRetry(sessionContext mongo.SessionContext, accountColl
 *mongo.Collection, t *testing.T) error {
    amountToTransfer := 500

    transactionOptions :=
 options.Transaction().SetReadConcern(readconcern.Snapshot()).

 SetWriteConcern(writeconcern.New(writeconcern.WMajority()))
    if err := sessionContext.StartTransaction(transactionOptions); err != nil {
        panic(err)
    }

    var result Account
    // deduct $500 from Alice's account
    err := accountColl.FindOne(sessionContext, bson.M{"name":
 "Alice"}).Decode(&result)
    aliceBalance := result.Balance
    newAliceBalance := aliceBalance - amountToTransfer
    _, err = accountColl.UpdateOne(sessionContext, bson.M{"name": "Alice"},
 bson.M{"$set": bson.M{"balance": newAliceBalance}})
    if err != nil {
        sessionContext.AbortTransaction(sessionContext)
    }
    err = accountColl.FindOne(sessionContext, bson.M{"name":
 "Alice"}).Decode(&result)
    aliceBalance = result.Balance
    assert.Equal(t, aliceBalance, newAliceBalance)
```

```
    // add $500 to Bob's account
    err = accountColl.FindOne(sessionContext, bson.M{"name": "Bob"}).Decode(&result)
    bobBalance := result.Balance
    newBobBalance := bobBalance + amountToTransfer
    _, err = accountColl.UpdateOne(sessionContext, bson.M{"name": "Bob"},
 bson.M{"$set": bson.M{"balance": newBobBalance}})
    if err != nil {
        sessionContext.AbortTransaction(sessionContext)
    }
    err = accountColl.FindOne(sessionContext, bson.M{"name": "Bob"}).Decode(&result)
    bobBalance = result.Balance
    assert.Equal(t, bobBalance, newBobBalance)

    err = sessionContext.CommitTransaction(sessionContext)
    return err
}

func doTransactionWithRetry(t *testing.T) {
    ctx := context.TODO()

    dbName := "bank"
    collName := "account"
    bankDB := client.Database(dbName)
    accountColl := bankDB.Collection(collName)

    client.UseSessionWithOptions(ctx, options.Session().SetCausalConsistency(false),
 func(sessionContext mongo.SessionContext) error {
        accountColl.Drop(ctx)
        accountColl.InsertOne(sessionContext, bson.M{"name" : "Alice",
 "balance":1000})
        accountColl.InsertOne(sessionContext, bson.M{"name" : "Bob",
 "balance":1000})
        for {
            err := transferMoneyWithRetry(sessionContext, accountColl, t)
            if err == nil {
                println("transaction committed")
                return nil
            }
            if mongoErr := err.(mongo.CommandError);
 mongoErr.HasErrorLabel("TransientTransactionError") {
                continue
            }
            println("transaction failed")
            return err
```

```
        }
    })

    // check results outside of transaction
    var result Account
    accountColl.FindOne(ctx, bson.M{"name": "Alice"}).Decode(&result)
    aliceBalance := result.Balance
    assert.Equal(t, aliceBalance, 500)
    accountColl.FindOne(ctx, bson.M{"name": "Bob"}).Decode(&result)
    bobBalance := result.Balance
    assert.Equal(t, bobBalance, 1500)
}
```

Java

다음 코드는 Amazon DocumentDB 트랜잭션 API를 Java와 함께 활용하는 방법을 보여줍니다.

```
// Java (sync) - Core API

public void transferMoneyWithRetry() {
   // connect to server
    MongoClientURI mongoURI = new MongoClientURI(uri);
    MongoClient mongoClient = new MongoClient(mongoURI);

    MongoDatabase bankDB = mongoClient.getDatabase("bank");
    MongoCollection accountColl = bankDB.getCollection("account");
    accountColl.drop();

   // insert some sample data
    accountColl.insertOne(new Document("name", "Alice").append("balance", 1000));
    accountColl.insertOne(new Document("name", "Bob").append("balance", 1000));

    while (true) {
        try {
            doTransferMoneyWithRetry(accountColl, mongoClient);
            break;
        } catch (MongoException e) {
            if (e.hasErrorLabel(MongoException.TRANSIENT_TRANSACTION_ERROR_LABEL)) {
                continue;
            } else {
                throw e;
            }
        }
    }
```

```
}

public void doTransferMoneyWithRetry(MongoCollection accountColl, MongoClient
 mongoClient) {
    int amountToTransfer = 500;

   TransactionOptions txnOptions = TransactionOptions.builder()
      .readConcern(ReadConcern.SNAPSHOT)
      .writeConcern(WriteConcern.MAJORITY)
      .build();
    ClientSessionOptions sessionOptions =
 ClientSessionOptions.builder().causallyConsistent(false).build();
    try ( ClientSession clientSession = mongoClient.startSession(sessionOptions) ) {
        clientSession.startTransaction(txnOptions);

        // deduct $500 from Alice's account
        List<Document> documentList = new ArrayList<>();
        accountColl.find(clientSession, new Document("name",
 "Alice")).into(documentList);
        int aliceBalance = (int) documentList.get(0).get("balance");
        Assert.assertTrue(aliceBalance >= amountToTransfer);
        int newAliceBalance = aliceBalance - amountToTransfer;
        accountColl.updateOne(clientSession, new Document("name", "Alice"), new
 Document("$set", new Document("balance", newAliceBalance)));

        // check Alice's new balance
        documentList = new ArrayList<>();
        accountColl.find(clientSession, new Document("name",
 "Alice")).into(documentList);
        int updatedBalance = (int) documentList.get(0).get("balance");
        Assert.assertEquals(updatedBalance, newAliceBalance);

        // add $500 to Bob's account
        documentList = new ArrayList<>();
        accountColl.find(clientSession, new Document("name",
 "Bob")).into(documentList);
        int bobBalance = (int) documentList.get(0).get("balance");
        int newBobBalance = bobBalance + amountToTransfer;
        accountColl.updateOne(clientSession, new Document("name", "Bob"), new
 Document("$set", new Document("balance", newBobBalance)));

        // check Bob's new balance
        documentList = new ArrayList<>();
```

```
        accountColl.find(clientSession, new Document("name",
 "Bob")).into(documentList);
        updatedBalance = (int) documentList.get(0).get("balance");
        Assert.assertEquals(updatedBalance, newBobBalance);

        // commit transaction
        clientSession.commitTransaction();
    }
}
// Java (async) -- Core API
public void transferMoneyWithRetry() {
    // connect to the server
    MongoClient mongoClient = MongoClients.create(uri);

    MongoDatabase bankDB = mongoClient.getDatabase("bank");
    MongoCollection accountColl = bankDB.getCollection("account");
    SubscriberLatchWrapper<Void> dropCallback = new SubscriberLatchWrapper<>();
    mongoClient.getDatabase("bank").drop().subscribe(dropCallback);
    dropCallback.await();

    // insert some sample data
    SubscriberLatchWrapper<InsertOneResult> insertionCallback = new
 SubscriberLatchWrapper<>();
    accountColl.insertOne(new Document("name", "Alice").append("balance",
 1000)).subscribe(insertionCallback);
    insertionCallback.await();

    insertionCallback = new SubscriberLatchWrapper<>();
    accountColl.insertOne(new Document("name", "Bob").append("balance",
 1000)).subscribe(insertionCallback);;
    insertionCallback.await();

    while (true) {
        try {
            doTransferMoneyWithRetry(accountColl, mongoClient);
            break;
        } catch (MongoException e) {
            if (e.hasErrorLabel(MongoException.TRANSIENT_TRANSACTION_ERROR_LABEL)) {
                continue;
            } else {
                throw e;
            }
        }
    }
```

```
}

public void doTransferMoneyWithRetry(MongoCollection accountColl, MongoClient
 mongoClient) {
    int amountToTransfer = 500;

    // start the transaction
    TransactionOptions txnOptions = TransactionOptions.builder()
            .readConcern(ReadConcern.SNAPSHOT)
            .writeConcern(WriteConcern.MAJORITY)
            .build();
    ClientSessionOptions sessionOptions =
 ClientSessionOptions.builder().causallyConsistent(false).build();

    SubscriberLatchWrapper<ClientSession> sessionCallback = new
 SubscriberLatchWrapper<>();
    mongoClient.startSession(sessionOptions).subscribe(sessionCallback);
    ClientSession session = sessionCallback.get().get(0);
    session.startTransaction(txnOptions);

    // deduct $500 from Alice's account
    SubscriberLatchWrapper<Document> findCallback = new SubscriberLatchWrapper<>();
    accountColl.find(session, new Document("name",
 "Alice")).first().subscribe(findCallback);
    Document documentFound = findCallback.get().get(0);
    int aliceBalance = (int) documentFound.get("balance");
    int newAliceBalance = aliceBalance - amountToTransfer;

    SubscriberLatchWrapper<UpdateResult> updateCallback = new
 SubscriberLatchWrapper<>();
    accountColl.updateOne(session, new Document("name",
 "Alice"), new Document("$set", new Document("balance",
 newAliceBalance))).subscribe(updateCallback);
    updateCallback.await();

    // check Alice's new balance
    findCallback = new SubscriberLatchWrapper<>();
    accountColl.find(session, new Document("name",
 "Alice")).first().subscribe(findCallback);
    documentFound = findCallback.get().get(0);
    int updatedBalance = (int) documentFound.get("balance");
    Assert.assertEquals(updatedBalance, newAliceBalance);

    // add $500 to Bob's account
```

```
    findCallback = new SubscriberLatchWrapper<>();
    accountColl.find(session, new Document("name",
 "Bob")).first().subscribe(findCallback);
    documentFound = findCallback.get().get(0);
    int bobBalance = (int) documentFound.get("balance");
    int newBobBalance = bobBalance + amountToTransfer;

    updateCallback = new SubscriberLatchWrapper<>();
    accountColl.updateOne(session, new Document("name", "Bob"), new Document("$set",
 new Document("balance", newBobBalance))).subscribe(updateCallback);
    updateCallback.await();

    // check Bob's new balance
    findCallback = new SubscriberLatchWrapper<>();
    accountColl.find(session, new Document("name",
 "Bob")).first().subscribe(findCallback);
    documentFound = findCallback.get().get(0);
    updatedBalance = (int) documentFound.get("balance");
    Assert.assertEquals(updatedBalance, newBobBalance);

    // commit the transaction
    SubscriberLatchWrapper<Void> transactionCallback = new
 SubscriberLatchWrapper<>();
    session.commitTransaction().subscribe(transactionCallback);
    transactionCallback.await();
}

public class SubscriberLatchWrapper<T> implements Subscriber<T> {

    /**
     * A Subscriber that stores the publishers results and provides a latch so can
 block on completion.
     *
     * @param <T> The publishers result type
     */
    private final List<T> received;
    private final List<RuntimeException> errors;
    private final CountDownLatch latch;
    private volatile Subscription subscription;
    private volatile boolean completed;

    /**
     * Construct an instance
     */
```

```
    public SubscriberLatchWrapper() {
        this.received = new ArrayList<>();
        this.errors = new ArrayList<>();
        this.latch = new CountDownLatch(1);
    }

    @Override
    public void onSubscribe(final Subscription s) {
        subscription = s;
        subscription.request(Integer.MAX_VALUE);
    }

    @Override
    public void onNext(final T t) {
        received.add(t);
    }

    @Override
    public void onError(final Throwable t) {
        if (t instanceof RuntimeException) {
            errors.add((RuntimeException) t);
        } else {
            errors.add(new RuntimeException("Unexpected exception", t));
        }
        onComplete();
    }

    @Override
    public void onComplete() {
        completed = true;
        subscription.cancel();
        latch.countDown();
    }

    /**
     * Get received elements
     *
     * @return the list of received elements
     */
    public List<T> getReceived() {
        return received;
    }

    /**
```

```
     * Get received elements.
     *
     * @return the list of receive elements
     */
    public List<T> get() {
        return await().getReceived();
    }

    /**
     * Await completion or error
     *
     * @return this
     */
    public SubscriberLatchWrapper<T> await() {
        subscription.request(Integer.MAX_VALUE);
        try {
            if (!latch.await(300, TimeUnit.SECONDS)) {
                throw new MongoTimeoutException("Publisher onComplete timed out for
 300 seconds");
            }
        } catch (InterruptedException e) {
            throw new MongoInterruptedException("Interrupted waiting for
 observeration", e);
        }
        if (!errors.isEmpty()) {
            throw errors.get(0);
        }
        return this;
    }

    public boolean getCompleted() {
        return this.completed;
    }

    public void close() {
        subscription.cancel();
        received.clear();
    }
}
```

C

다음 코드는 Amazon DocumentDB 트랜잭션 API를 C와 함께 활용하는 방법을 보여줍니다.

```
// Sample C code with core session

bool core_session(mongoc_client_session_t *client_session, mongoc_collection_t*
 collection, bson_t *selector, int64_t balance){
    bool r = true;
    bson_error_t error;
    bson_t *opts = bson_new();
    bson_t *update = BCON_NEW ("$set", "{", "balance", BCON_INT64 (balance), "}");

    // set read & write concern
    mongoc_read_concern_t *read_concern = mongoc_read_concern_new ();
    mongoc_write_concern_t *write_concern = mongoc_write_concern_new ();
    mongoc_transaction_opt_t *txn_opts = mongoc_transaction_opts_new ();

    mongoc_write_concern_set_w(write_concern, MONGOC_WRITE_CONCERN_W_MAJORITY);
    mongoc_read_concern_set_level(read_concern, MONGOC_READ_CONCERN_LEVEL_SNAPSHOT);
    mongoc_transaction_opts_set_write_concern (txn_opts, write_concern);
    mongoc_transaction_opts_set_read_concern (txn_opts, read_concern);

    mongoc_client_session_start_transaction (client_session, txn_opts, &error);
    mongoc_client_session_append (client_session, opts, &error);

    r = mongoc_collection_update_one (collection, selector, update, opts, NULL,
 &error);

    mongoc_client_session_commit_transaction (client_session, NULL, &error);
    bson_destroy (opts);
    mongoc_transaction_opts_destroy(txn_opts);
    mongoc_read_concern_destroy(read_concern);
    mongoc_write_concern_destroy(write_concern);
    bson_destroy (update);
    return r;
}

void test_core_money_transfer(mongoc_client_t* client, mongoc_collection_t*
 collection, int amount_to_transfer){

    bson_t reply;
    bool r = true;
    const bson_t *doc;
    bson_iter_t iter;
    bson_error_t error;
```

```
    // find query
    bson_t *alice_query = bson_new ();
    BSON_APPEND_UTF8(alice_query, "name", "Alice");

    bson_t *bob_query = bson_new ();
    BSON_APPEND_UTF8(bob_query, "name", "Bob");

    // create session
    // set causal consistency to false
    mongoc_session_opt_t *session_opts = mongoc_session_opts_new ();
    mongoc_session_opts_set_causal_consistency (session_opts, false);
    // start the session
    mongoc_client_session_t *client_session = mongoc_client_start_session (client,
session_opts, &error);

    // add session to options
    bson_t *opts = bson_new();
    mongoc_client_session_append (client_session, opts, &error);

    // deduct 500 from Alice
    // find account balance of Alice
    mongoc_cursor_t *cursor = mongoc_collection_find_with_opts (collection,
alice_query, NULL, NULL);
    mongoc_cursor_next (cursor, &doc);
    bson_iter_init (&iter, doc);
    bson_iter_find (&iter, "balance");
    int64_t alice_balance = (bson_iter_value (&iter))->value.v_int64;
    assert(alice_balance >= amount_to_transfer);
    int64_t new_alice_balance = alice_balance - amount_to_transfer;

    // core
    r = core_session (client_session, collection, alice_query, new_alice_balance);
    assert(r);

    // find account balance of Alice after transaction
    cursor = mongoc_collection_find_with_opts (collection, alice_query, NULL, NULL);
    mongoc_cursor_next (cursor, &doc);
    bson_iter_init (&iter, doc);
    bson_iter_find (&iter, "balance");
    alice_balance = (bson_iter_value (&iter))->value.v_int64;
    assert(alice_balance == new_alice_balance);
    assert(alice_balance == 500);

    // add 500 to Bob's balance
```

```
    // find account balance of Bob
    cursor = mongoc_collection_find_with_opts (collection, bob_query, NULL, NULL);
    mongoc_cursor_next (cursor, &doc);
    bson_iter_init (&iter, doc);
    bson_iter_find (&iter, "balance");
    int64_t bob_balance = (bson_iter_value (&iter))->value.v_int64;
    int64_t new_bob_balance = bob_balance + amount_to_transfer;

    //core
    r = core_session (client_session, collection, bob_query, new_bob_balance);
    assert(r);

    // find account balance of Bob after transaction
    cursor = mongoc_collection_find_with_opts (collection, bob_query, NULL, NULL);
    mongoc_cursor_next (cursor, &doc);
    bson_iter_init (&iter, doc);
    bson_iter_find (&iter, "balance");
    bob_balance = (bson_iter_value (&iter))->value.v_int64;
    assert(bob_balance == new_bob_balance);
    assert(bob_balance == 1500);

    // cleanup
    bson_destroy(alice_query);
    bson_destroy(bob_query);
    mongoc_client_session_destroy(client_session);
    bson_destroy(opts);
    mongoc_cursor_destroy(cursor);
    bson_destroy(doc);
}

int main(int argc, char* argv[]) {
    mongoc_init ();
    mongoc_client_t* client = mongoc_client_new (<connection uri>);
    bson_error_t error;

    // connect to bank db
    mongoc_database_t *database = mongoc_client_get_database (client, "bank");
    // access account collection
    mongoc_collection_t* collection = mongoc_client_get_collection(client, "bank",
 "account");
    // set amount to transfer
    int64_t amount_to_transfer = 500;
    // delete the collection if already existing
    mongoc_collection_drop(collection, &error);
```

```
    // open Alice account
    bson_t *alice_account = bson_new ();
    BSON_APPEND_UTF8(alice_account, "name", "Alice");
    BSON_APPEND_INT64(alice_account, "balance", 1000);

    // open Bob account
    bson_t *bob_account = bson_new ();
    BSON_APPEND_UTF8(bob_account, "name", "Bob");
    BSON_APPEND_INT64(bob_account, "balance", 1000);

    bool r = true;

    r = mongoc_collection_insert_one(collection, alice_account, NULL, NULL, &error);
    if (!r) {printf("Error encountered:%s", error.message);}
    r = mongoc_collection_insert_one(collection, bob_account, NULL, NULL, &error);
    if (!r) {printf("Error encountered:%s", error.message);}

    test_core_money_transfer(client, collection, amount_to_transfer);

}
```

Scala

다음 코드는 Amazon DocumentDB 트랜잭션 API를 Scala와 함께 활용하는 방법을 보여줍니다.

```
// Scala Core API
def transferMoneyWithRetry(sessionObservable: SingleObservable[ClientSession] ,
 database: MongoDatabase ): Unit = {
    val accountColl = database.getCollection("account")
    var amountToTransfer = 500

    var transactionObservable: Observable[ClientSession] =
 sessionObservable.map(clientSession => {
    clientSession.startTransaction()

    // deduct $500 from Alice's account
    var aliceBalance = accountColl.find(clientSession, Document("name" ->
 "Alice")).await().head.getInteger("balance")
    assert(aliceBalance >= amountToTransfer)
    var newAliceBalance = aliceBalance - amountToTransfer
    accountColl.updateOne(clientSession, Document("name" -> "Alice"),
 Document("$set" -> Document("balance" -> newAliceBalance))).await()
```

```
    aliceBalance = accountColl.find(clientSession, Document("name" ->
 "Alice")).await().head.getInteger("balance")
    assert(aliceBalance == newAliceBalance)

    // add $500 to Bob's account
    var bobBalance = accountColl.find(clientSession, Document("name" ->
 "Bob")).await().head.getInteger("balance")
    var newBobBalance = bobBalance + amountToTransfer
    accountColl.updateOne(clientSession, Document("name" -> "Bob"), Document("$set"
 -> Document("balance" -> newBobBalance))).await()
    bobBalance = accountColl.find(clientSession, Document("name" ->
 "Bob")).await().head.getInteger("balance")
    assert(bobBalance == newBobBalance)

    clientSession
    })

    transactionObservable.flatMap(clientSession =>
 clientSession.commitTransaction()).await()
}

def doTransactionWithRetry(): Unit = {
    val client: MongoClient = MongoClientWrapper.getMongoClient()
    val database: MongoDatabase = client.getDatabase("bank")
    val accountColl = database.getCollection("account")
    accountColl.drop().await()

    val sessionOptions =
 ClientSessionOptions.builder().causallyConsistent(false).build()
    var  sessionObservable: SingleObservable[ClientSession] =
 client.startSession(sessionOptions)
    accountColl.insertOne(Document("name" -> "Alice", "balance" -> 1000)).await()
    accountColl.insertOne(Document("name" -> "Bob", "balance" -> 1000)).await()

    var retry = true
    while (retry) {
        try {
        transferMoneyWithRetry(sessionObservable, database)
        println("transaction committed")
        retry = false
        }
        catch {
        case e: MongoException if
 e.hasErrorLabel(MongoException.TRANSIENT_TRANSACTION_ERROR_LABEL) => {
```

```
            println("retrying transaction")
        }
        case other: Throwable => {
            println("transaction failed")
            retry = false
            throw other

        }
        }
    }

    // check results outside of transaction
    assert(accountColl.find(Document("name" ->
 "Alice")).results().head.getInteger("balance") == 500)
    assert(accountColl.find(Document("name" ->
 "Bob")).results().head.getInteger("balance") == 1500)

    accountColl.drop().await()

}
```

Python

다음 코드는 Amazon DocumentDB 트랜잭션 API를 Python과 함께 활용하는 방법을 보여줍니다.

```
// Sample Python code with Core api

import pymongo

client = pymongo.MongoClient(<connection_string>)
rc_snapshot = pymongo.read_concern.ReadConcern('snapshot')
wc_majority = pymongo.write_concern.WriteConcern('majority')

# To start, drop and create an account collection and insert balances for both Alice
 and Bob
collection = client.get_database("bank").get_collection("account")
collection.drop()
collection.insert_one({"_id": 1, "name": "Alice", "balance": 1000})
collection.insert_one({"_id": 2, "name": "Bob", "balance": 1000})

amount_to_transfer = 500

# deduct 500 from Alice's account
alice_balance = collection.find_one({"name": "Alice"}).get("balance")
```

```
assert alice_balance >= amount_to_transfer
new_alice_balance = alice_balance - amount_to_transfer

with client.start_session({'causalConsistency':False}) as session:
    session.start_transaction(read_concern=rc_snapshot, write_concern=wc_majority)
    collection.update_one({"name": "Alice"}, {'$set': {"balance":
 new_alice_balance}}, session=session)
    session.commit_transaction()

updated_alice_balance = collection.find_one({"name": "Alice"}).get("balance")
assert updated_alice_balance == new_alice_balance

# add 500 to Bob's account
bob_balance = collection.find_one({"name": "Bob"}).get("balance")
assert bob_balance >= amount_to_transfer
new_bob_balance = bob_balance + amount_to_transfer

with client.start_session({'causalConsistency':False}) as session:
    session.start_transaction(read_concern=rc_snapshot, write_concern=wc_majority)
    collection.update_one({"name": "Bob"}, {'$set': {"balance": new_bob_balance}},
 session=session)
    session.commit_transaction()

updated_bob_balance = collection.find_one({"name": "Bob"}).get("balance")
assert updated_bob_balance == new_bob_balance
```

지원되는 명령

Command	지원
abortTransaction	예
commitTransaction	예
endSessions	예
killSession	예
killAllSession	예
killAllSessionsByPattern	아니요

Command	지원
refreshSessions	아니요
startSession	예

지원되지 않는 기능

메서드	단계 또는 명령
db.collection.aggregate()	$collStats $currentOp $indexStats $listSessions $out
db.collection.count() db.collection.countDocuments()	$where $near $nearSphere
db.collection.insert()	기존 컬렉션에 대해 실행되지 않는 경우 insert은 지원되지 않습니다. 이 메서드는 기존 컬렉션을 대상으로 하는 경우 지원됩니다.

세션

MongoDB 세션은 재시도 가능한 쓰기, 인과 관계 일관성, 트랜잭션을 지원하고 데이터베이스 전반의 작업을 관리하는 데 사용되는 프레임워크입니다. 세션이 생성되면 클라이언트에서 lsid(논리적 세션 식별자)를 생성하고 서버로 명령을 보낼 때 해당 세션 내의 모든 작업에 태그를 지정하는 데 사용됩니다.

Amazon DocumentDB는 세션을 사용하여 트랜잭션을 활성화하는 것을 지원하지만 인과 관계 일관성이나 재시도 가능한 쓰기는 지원하지 않습니다.

Amazon DocumentDB 내에서 트랜잭션을 사용하는 경우 `session.startTransaction()` API를 사용하여 세션 내에서 트랜잭션이 시작되며 세션은 한 번에 하나의 트랜잭션만 지원합니다. 마찬가지로, 트랜잭션은 커밋(`session.commitTransaction()`) 또는 중단(`session.abortTransaction()`) API를 사용하여 완료됩니다.

인과 관계 일관성

인과 관계 일관성을 통해 클라이언트는 단일 클라이언트 세션 내에서 읽기 후 쓰기 일관성, 단일 원자 읽기/쓰기 및 읽기 후 쓰기가 수행되며 이러한 보장은 기본 인스턴스뿐만 아니라 클러스터의 모든 인스턴스에 적용됩니다. Amazon DocumentDB는 인과 관계의 일관성을 지원하지 않으므로 다음 명령문을 실행하면 오류가 발생합니다.

```
var mySession = db.getMongo().startSession();
var mySessionObject = mySession.getDatabase('test').getCollection('account');

mySessionObject.updateOne({"_id": 2}, {"$inc": {"balance": 400}});
//Result:{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

mySessionObject.find()
//Error: error: {
//         "ok" : 0,
//         "code" : 303,
//         "errmsg" : "Feature not supported: 'causal consistency'",
//         "operationTime" : Timestamp(1603461817, 493214)
//}

mySession.endSession()
```

세션 내에서 인과 관계 일관성을 비활성화할 수 있습니다. 단, 이렇게 하면 세션 프레임워크를 활용할 수 있지만 읽기에 대한 인과 관계 일관성을 보장할 수는 없습니다. Amazon DocumentDB를 사용할 경우 기본 인스턴스에서의 읽기는 쓰기 후 읽기 일관성이 유지되고 복제본 인스턴스에서의 읽기도 최종적으로 일관됩니다. 트랜잭션은 세션을 활용하는 주요 사용 사례입니다.

```
var mySession = db.getMongo().startSession({causalConsistency: false});
var mySessionObject = mySession.getDatabase('test').getCollection('account');

mySessionObject.updateOne({"_id": 2}, {"$inc": {"balance": 400}});
//Result:{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
```

```
mySessionObject.find()
//{ "_id" : 1, "name" : "Bob", "balance" : 100 }
//{ "_id" : 2, "name" : "Alice", "balance" : 1700 }
```

재시도 가능한 쓰기

재시도 가능한 쓰기는 네트워크 오류가 발생하거나 클라이언트가 기본 데이터를 찾을 수 없는 경우 클라이언트가 쓰기 작업을 한 번 재시도하는 기능입니다. Amazon DocumentDB에서는 재시도 가능한 쓰기가 지원되지 않으므로 이를 비활성화해야 합니다. 연결 문자열의 명령(`retryWrites=false`)을 사용하여 비활성화할 수 있습니다.

> **Note**
>
> mongo 쉘(mongosh 제외)을 사용하는 경우 코드 문자열에 `retryWrites=false` 명령을 포함시키지 마십시오. 기본적으로 재시도 가능한 쓰기는 비활성화되어 있습니다. `retryWrites=false`를 포함하면 일반 읽기 명령에서 오류가 발생할 수 있습니다.

트랜잭션 오류

트랜잭션을 사용할 때 트랜잭션 번호가 진행 중인 트랜잭션과 일치하지 않는다는 오류가 발생할 수 있는 시나리오가 있습니다.

이 오류는 최소한 두 가지 상황에서 발생할 수 있습니다.

- 1분 트랜잭션 제한 시간 후입니다.
- 인스턴스를 다시 시작한 후(패칭, 충돌 복구 등으로 인해) 트랜잭션이 성공적으로 커밋된 경우에도 이 오류를 수신할 수 있습니다. 인스턴스를 다시 시작하는 동안 데이터베이스는 성공적으로 완료된 트랜잭션과 중단된 트랜잭션 간의 차이를 알 수 없습니다. 즉, 트랜잭션 완료 상태가 모호합니다.

이 오류를 처리하는 가장 좋은 방법은 트랜잭션 업데이트를 멱등적으로 만드는 것입니다. 예를 들어 증가/감소 작업 대신 `$set` 뮤테이터를 사용하는 것입니다. 아래 내용을 참조하십시오.

```
{ "ok" : 0,
"operationTime" : Timestamp(1603938167, 1),
"code" : 251,
"errmsg" : "Given transaction number 1 does not match any in-progress transactions."
}
```

Amazon DocumentDB 모범 사례

Amazon DocumentDB(MongoDB 호환)를 사용하여 작업하는 모범 사례를 알아보세요. 이 섹션은 새로운 모범 사례가 확인되는 대로 지속적으로 업데이트됩니다.

주제

- 기본 운영 지침
- 인스턴스 크기 조정
- 인덱스 작업
- 보안 모범 사례
- 비용 최적화
- 지표를 통해 성능 문제 식별
- TTL 및 시계열 워크로드
- 마이그레이션
- 클러스터 파라미터 그룹 작업
- 집계 파이프라인 쿼리
- batchInsert 및 batchUpdate

기본 운영 지침

다음은 Amazon DocumentDB로 작업할 때 모든 사용자가 따라야 하는 기본 운영 지침입니다. Amazon DocumentDB 서비스 수준 계약에 다음 지침을 따르도록 명시되어 있습니다.

- 두 개 이상의 Amazon DocumentDB 인스턴스로 구성된 클러스터를 두 개의 AWS 가용 영역에 배포합니다. 프로덕션 워크로드의 경우 3개의 가용 영역에 3개 이상의 Amazon DocumentDB 인스턴스로 구성된 클러스터를 배포하는 것이 좋습니다.
- 명시된 서비스 제한 내에서 서비스를 사용하세요. 자세한 내용은 Amazon DocumentDB 할당량 및 제한 단원을 참조하십시오.
- 메모리, CPU, 연결 및 스토리지 사용량을 모니터링합니다. 시스템 성능 및 가용성을 유지 관리하기 위해, 사용 패턴이 변경되거나 배포 용량에 도달할 때 알리도록 Amazon CloudWatch를 설정합니다.
- 용량 한도에 도달할 경우 인스턴스를 확장합니다. 애플리케이션의 예상치 못한 수요 증가를 수용할 수 있는 충분한 컴퓨팅 리소스(예: RAM, CPU)를 사용하여 인스턴스를 프로비저닝해야 합니다.
- 목표 복구 시점에 맞춰 백업 보존 기간을 설정합니다.

- 클러스터의 장애 조치를 테스트하여 사용 사례 절차에 소요되는 시간을 파악하세요. 자세한 내용은 Amazon DocumentDB 장애 조치 단원을 참조하십시오.
- 복제본 집합 모드(Amazon DocumentDB에 복제본 세트로 연결 참조)에서 클러스터 엔드포인트(Amazon DocumentDB 엔드포인트 참조)를 사용하여 Amazon DocumentDB 클러스터에 연결함으로써 장애 조치가 애플리케이션에 미치는 영향을 최소화합니다.
- 애플리케이션의 읽기 일관성 요구 사항을 충족하면서 읽기 확장성을 최대화하는 드라이버 읽기 기본 설정을 선택합니다. `secondaryPreferred` 읽기 기본 설정은 복제본 읽기를 활성화하고, 더 많은 작업을 수행할 수 있도록 기본 인스턴스를 비웁니다. 자세한 내용은 읽기 기본 설정 옵션 단원을 참조하십시오.
- 네트워크 및 데이터베이스 오류가 발생하는 경우 복원력을 발휘하도록 애플리케이션을 설계합니다. 일시적인 오류와 지속적인 오류를 구별하기 위해 드라이버의 오류 메커니즘을 사용합니다. 적절한 경우 지수 백오프 메커니즘을 사용하여 일시적인 오류를 다시 시도합니다. 재시도 논리를 구현할 때 애플리케이션에서 데이터 일관성을 고려해야 합니다.
- 모든 프로덕션 클러스터 또는 중요한 데이터가 있는 클러스터에 대해 클러스터 삭제 보호를 활성화합니다. Amazon DocumentDB 클러스터를 삭제하기 전에 최종 스냅샷을 생성합니다. 를 사용하여 리소스를 배포하는 경우 종료 방지를 AWS CloudFormation활성화합니다. 자세한 내용은 종료 방지 및 삭제 방지 단원을 참조하십시오.
- Amazon DocumentDB 클러스터를 생성할 때 `--engine-version`는 기본적으로 최신 메이저 엔진 버전으로 설정되는 선택적 파라미터입니다. 현재 메이저 엔진 버전은 5.0.0입니다. 새 메이저 엔진 버전이 릴리스되면의 기본 엔진 버전이 마지막 메이저 엔진 버전을 반영하도록 `--engine-version` 업데이트됩니다. 따라서 프로덕션 워크로드, 특히 스크립팅, 자동화 또는 AWS CloudFormation 템플릿에 의존하는 워크로드의 경우를 의도한 메이저 버전`--engine-version`에 명시적으로 지정하는 것이 좋습니다.

인스턴스 크기 조정

Amazon DocumentDB에서 인스턴스 크기를 선택할 때 가장 중요한 요소 중 하나는 캐시에 사용할 RAM 용량입니다. Amazon DocumentDB는 RAM의 3분의 1을 자체 서비스를 위해 예약합니다. 즉, 캐시에 사용할 수 있는 인스턴스 RAM은 3분의 2뿐입니다. 작업 집합(즉, 데이터 및 인덱스)을 메모리에 저장할 수 있는 충분한 RAM이 있는 인스턴스 유형을 선택하는 것이 Amazon DocumentDB 성능면에서 가장 좋습니다. 인스턴스 크기를 적절히 조정하면 전반적인 성능을 최적화하고 잠재적으로 I/O 비용을 최소화하는 데 도움이 됩니다.

애플리케이션의 작업 집합이 메모리에 적합한지 확인하려면 부하가 걸린 클러스터의 각 인스턴스별로 Amazon CloudWatch를 사용하여 `BufferCacheHitRatio`를 모니터링합니다.

BufferCacheHitRatio CloudWatch 지표는 스토리지 볼륨 대비 인스턴스의 메모리 캐시에서 제공되는 데이터 및 인덱스의 비율을 측정합니다. 일반적으로 작업 집합 메모리에서 데이터를 읽는 것이 저장소 볼륨에서 읽는 것보다 빠르고 비용 효율적이므로 BufferCacheHitRatio의 값은 가능한 한 높아야 합니다. BufferCacheHitRatio를 가능한 한 100%에 가깝게 유지하는 것이 바람직하지만 달성 가능한 최상의 가치는 애플리케이션의 액세스 패턴과 성능 요구 사항에 따라 달라집니다. BufferCacheHitRatio를 가능한 한 최대로 유지하려면 인덱스와 작업 데이터 세트가 메모리에 맞을 수 있을 만큼 충분한 RAM으로 클러스터의 인스턴스를 프로비저닝하는 것이 좋습니다.

인덱스가 메모리에 맞지 않으면 BufferCacheHitRatio가 더 낮게 표시될 수 있습니다. 디스크에서 계속 읽으면 추가 I/O 비용이 발생하며 메모리에서 읽는 것보다 좋은 성능을 얻을 수 없습니다. BufferCacheHitRatio 비율이 예상보다 낮으면 클러스터가 메모리의 작업 집합 데이터에 맞게 더 많은 RAM을 제공할 수 있도록 인스턴스 크기를 늘립니다. 인스턴스 클래스를 확장하여 BufferCacheHitRatio가 급격히 증가하면 애플리케이션의 작업 집합이 메모리에 맞은 것입니다. 규모 조정 작업 후 BufferCacheHitRatio가 더 이상 대폭 증가하지 않을 때까지 계속 확장합니다. 인스턴스의 지표 모니터링에 대한 자세한 내용은 Amazon DocumentDB 지표 섹션을 참조하세요.

워크로드 및 지연 시간 요구 사항에 따라 애플리케이션이 정상 상태 사용 중에는 더 높은 BufferCacheHitRatio 값을 가질 수 있지만 전체 컬렉션을 스캔해야 하는 분석 쿼리가 인스턴스에서 실행되므로 주기적으로 BufferCacheHitRatio 딥을 수행할 수 있습니다. BufferCacheHitRatio의 이러한 주기적 딥은 스토리지 볼륨에서 작업 집합 데이터를 버퍼 캐시로 다시 채워야 하는 후속 쿼리에 대해 더 높은 지연 시간으로 나타날 수 있습니다. 프로덕션에 워크로드를 배포하기 전에 성능 특성 및 **BufferCacheHitRatio**를 이해하려면 먼저 대표적인 프로덕션 워크로드를 사용하여 사전 프로덕션 환경에서 워크로드를 테스트하는 것이 좋습니다.

BufferCacheHitRatio는 인스턴스별 지표이므로 기본 인스턴스와 복제본 인스턴스 간에 읽기가 분산되는 방식에 따라 동일한 클러스터 내의 여러 인스턴스가 서로 다른 BufferCacheHitRatio 값을 가질 수 있습니다. 작업 워크로드가 분석 쿼리를 실행한 후 작업 집합 캐시를 다시 채울 때 발생하는 대기 시간 증가를 주기적으로 처리할 수 없는 경우 일반 작업 워크로드의 버퍼 캐시를 분석 질의의 버퍼 캐시와 분리해야 합니다. 운영 쿼리를 기본 인스턴스로, 분석 쿼리는 복제본 인스턴스로만 전송하여 BufferCacheHitRatio를 완전히 격리할 수 있습니다. 또한 일정 비율의 일반 쿼리가 해당 복제본에서 실행되고 영향을 받을 수 있음을 이해하고 분석 쿼리를 특정 복제본 인스턴스로 전송하여 부분 격리를 수행할 수도 있습니다.

적합한 BufferCacheHitRatio 값은 사용 사례 및 애플리케이션 요구 사항에 따라 다릅니다. 이 지표에는 최고 또는 최소 값이 하나도 없습니다. 비용 및 성능 측면에서 일시적으로 낮은 BufferCacheHitRatio의 절충이 허용되는지 여부만 결정할 수 있습니다.

인덱스 작업

인덱스 작성

Amazon DocumentDB로 데이터를 가져올 때, 대규모 데이터 세트를 가져오기 전에 인덱스를 생성해야 합니다. Amazon DocumentDB 인덱스 도구를 사용하여 실행 중인 MongoDB 인스턴스 또는 mongodump 디렉터리에서 인덱스를 추출하고, Amazon DocumentDB 클러스터에서 해당 인덱스를 생성할 수 있습니다. 마이그레이션에 대한 자세한 지침은 Amazon DocumentDB로 마이그레이션 섹션을 참조하세요.

인덱스 선택성

중복 값 수가 컬렉션에 있는 총 문서 수의 1% 미만인 필드로 인덱스 생성을 제한하는 것이 좋습니다. 예를 들어 컬렉션에 100,000개의 문서가 포함되어 있는 경우, 동일한 값이 1000회 이하로 나타나는 필드에만 인덱스를 생성하세요.

고유 값이 많은 인덱스를 선택하면(즉, 카디널리티가 높음) 필터 작업에서 적은 수의 문서를 반환하므로 인덱스 스캔 중에 우수한 성능을 얻을 수 있습니다. 높은 카디널리티 인덱스의 예는 같음 조건자가 최대 하나의 문서를 반환하도록 보장하는 고유 인덱스입니다. 낮은 카디널리티의 예로는 부울 필드에 대한 인덱스와 요일에 대한 인덱스가 있습니다. 성능 저하로 인해 낮은 카디널리티 인덱스는 데이터베이스의 쿼리 최적화 프로그램에 의해 선택될 가능성이 낮습니다. 동시에 낮은 카디널리티 인덱스는 디스크 스페이스 및 I/O와 같은 리소스를 계속 사용합니다. 일반적으로 일반적인 값 빈도가 전체 컬렉션 크기의 1% 이하인 필드의 인덱스를 대상으로 해야 합니다.

또한 일반적으로 필터로 사용되는 필드에만 인덱스를 생성하고, 사용하지 않는 인덱스를 정기적으로 찾는 것이 좋습니다. 자세한 내용은 인덱스 사용량을 분석하고 사용하지 않는 인덱스를 식별하려면 어떻게 해야 하나요? 단원을 참조하십시오.

인덱스가 데이터 작성에 미치는 영향

인덱스는 컬렉션의 모든 문서를 스캔할 필요가 없으므로 쿼리 성능을 향상할 수 있지만 이러한 개선 시에 다른 불리함이 따를 수 있습니다. 컬렉션의 각 인덱스에 대해 문서가 삽입, 업데이트 또는 삭제될 때마다 데이터베이스는 컬렉션을 업데이트하고 컬렉션의 각 인덱스에 필드를 기록해야 합니다. 예를 들어 컬렉션에 9개의 인덱스가 있는 경우 데이터베이스는 클라이언트에 작업을 승인하기 전에 10개의 쓰기를 수행해야 합니다. 따라서 인덱스가 추가될 때마다 쓰기 지연 시간, I/O가 추가되고 전체 활용도가 높은 스토리지가 증가합니다.

모든 작업 집합 메모리를 유지하려면 클러스터 인스턴스의 크기를 적절하게 조정해야 합니다. 이에 따라 스토리지 볼륨에서 인덱스 페이지를 지속적으로 읽을 필요가 없으므로 성능에 부정적인 영향을 미치고 I/O 비용이 증가합니다. 자세한 내용은 인스턴스 크기 조정 단원을 참조하십시오.

최상의 성능을 얻으려면 일반 쿼리의 성능을 향상하는 데 필요한 인덱스만 추가하여 컬렉션의 인덱스 수를 최소화합니다. 워크로드는 다양하지만 컬렉션당 인덱스 수를 5 이하로 유지하는 것이 좋습니다.

누락된 인덱스 식별

누락된 인덱스를 식별하고 제거하는 작업을 정기적으로 수행하는 것이 좋습니다. 자세한 내용은 누락된 인덱스는 어떻게 식별하나요? 섹션을 참조하세요.

사용되지 않는 인덱스 식별

사용되지 않는 인덱스를 식별하고 제거하는 작업을 정기적으로 수행하는 것이 좋습니다. 자세한 내용은 인덱스 사용량을 분석하고 사용하지 않는 인덱스를 식별하려면 어떻게 해야 하나요? 섹션을 참조하세요.

보안 모범 사례

보안 모범 사례의 경우 AWS Identity and Access Management (IAM) 계정을 사용하여 Amazon DocumentDB API 작업, 특히 Amazon DocumentDB 리소스를 생성, 수정 또는 삭제하는 작업에 대한 액세스를 제어해야 합니다. 이러한 리소스로는 클러스터, 보안 그룹 및 파라미터 그룹을 들 수 있습니다. 또한 IAM을 사용하여 클러스터 복원 백업과 같은 일반적인 관리 작업을 수행하는 작업을 제어해야 합니다. IAM 역할을 생성할 때 최소 권한 원칙을 사용합니다.

- 역할 기반 액세스 제어와 함께 최소 권한을 적용합니다.
- Amazon DocumentDB 리소스를 관리하는 각 사용자에게 개별 IAM 계정을 할당합니다. AWS 계정 루트 사용자를 사용하여 Amazon DocumentDB 리소스를 관리하지 마세요. 자신을 포함한 모든 사람을 위한 IAM 사용자를 생성합니다.
- 각 IAM 사용자에게 각자의 임무를 수행하는 데 필요한 최소 권한 집합을 부여합니다.
- IAM 그룹을 사용해 여러 사용자에 대한 권한을 효과적으로 관리합니다. IAM에 대한 자세한 내용은 IAM 사용 설명서를 참조하세요. IAM 모범 사례에 대한 자세한 내용은 IAM 모범 사례를 참조하세요.
- IAM 자격 증명을 정기적으로 교체합니다.
- Amazon DocumentDB의 AWS 보안 암호를 자동으로 교체하도록 Secrets Manager를 구성합니다. 자세한 내용은 AWS Secrets Manager 사용 설명서의 Secrets Manager 보안 암호 교체 및 Amazon DocumentDB용 보안 암호 교체를 참조하세요. AWS

- 각 Amazon DocumentDB 사용자에게 각자의 임무를 수행하는 데 필요한 최소 권한 집합을 부여합니다. 자세한 내용은 역할 기반 액세스 제어를 사용한 데이터베이스 액세스 단원을 참조하십시오.
- 전송 계층 보안(TLS)을 사용하여 전송 중 데이터를 암호화하고 저장 데이터를 AWS KMS 암호화합니다.

비용 최적화

다음 모범 사례는 Amazon DocumentDB를 사용할 때 비용을 관리하고 최소화하는 데 도움이 될 수 있습니다. 요금 정보는 Amazon DocumentDB(MongoDB 호환) 요금 및 Amazon DocumentDB(MongoDB 호환) FAQ를 참조하세요.

- 해당 월 예상 청구액의 50% 및 75% 임계값으로 결제 알림을 생성합니다. 결제 알림 생성에 대한 자세한 내용은 결제 경보 생성을 참조하세요.
- Amazon DocumentDB의 아키텍처는 스토리지와 컴퓨팅을 분리하므로 단일 인스턴스 클러스터도 매우 내구성이 높습니다. 클러스터 스토리지 볼륨은 3개의 가용 영역에 걸쳐 6방향으로 데이터를 복제하므로, 클러스터에 있는 인스턴스 수와 상관없이 매우 높은 내구성을 제공합니다. 일반적인 프로덕션 클러스터에는 고가용성을 제공하기 위해 3개 이상의 인스턴스가 있습니다. 그러나 고가용성이 필요하지 않은 경우 단일 인스턴스 개발 클러스터를 사용하여 비용을 최적화할 수 있습니다.
- 개발 및 테스트 시나리오의 경우 더 이상 필요하지 않을 때 클러스터를 중지하고 개발이 다시 시작되면 클러스터를 시작합니다. 자세한 내용은 Amazon DocumentDB 클러스터 중지 및 시작 단원을 참조하십시오.
- 데이터가 쓰기, 읽기 및 삭제될 때 TTL과 스트림 변경은 모두 I/O를 발생합니다. 이러한 기능을 활성화했지만 애플리케이션에서 사용하지 않는 경우 이 기능을 비활성화하면 비용이 절감될 수 있습니다.

지표를 통해 성능 문제 식별

주제

- 성능 지표 보기
- CloudWatch 경보 설정
- 성능 지표 평가
- CloudWatch 지표를 사용한 Amazon DocumentDB 인스턴스 사용량 평가
- 쿼리 튜닝

리소스 부족이나 기타 일반적인 병목 현상으로 인해 발생하는 성능 문제를 식별하기 위해 Amazon DocumentDB 클러스터에서 사용 가능한 지표를 모니터링할 수 있습니다.

성능 지표 보기

다양한 기간 동안의 평균, 최대, 최소 측정값을 보려면 성능 지표를 정기적으로 모니터링합니다. 이렇게 하면 성능이 저하된 시점을 식별하는 데 도움이 됩니다. 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. 새 클러스터를 설정하고 일반 워크로드로 실행한 후 서로 다른 간격(예: 1시간, 24시간, 1주, 2주)으로 모든 성능 지표의 평균값, 최댓값 및 최솟값을 파악하세요. 이렇게 하면 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.

AWS Management Console 또는를 사용하여 성능 지표를 볼 수 있습니다 AWS CLI. 자세한 내용은 CloudWatch 데이터 보기 단원을 참조하십시오.

CloudWatch 경보 설정

CloudWatch 경보 생성에 대한 자세한 내용을 알아보려면 Amazon CloudWatch 사용 설명서의 Amazon CloudWatch 경보 사용을 참조하세요.

성능 지표 평가

인스턴스에는 여러 지표 범주가 있습니다. 허용되는 값을 결정하는 방법은 지표에 따라 다릅니다.

CPU

- CPU 사용률 – 사용된 컴퓨터 처리 용량의 백분율입니다.

메모리

- 여유 메모리 - 인스턴스에서 사용 가능한 RAM을 나타냅니다.
- 스왑 사용량 - 인스턴스에서 스왑 스페이스를 얼마나 많이 사용했는지를 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.
- 디스크 대기열 깊이 - 디스크에 쓰거나 디스크에서 읽기를 기다리는 I/O 작업 수입니다.

네트워크 트래픽

- 네트워크 수신 처리량, 네트워크 전송 처리량 - 인스턴스에 대한 수신 또는 전송 네트워크 트래픽 속도(초당 메가바이트)입니다.

데이터베이스 연결

- DB 연결 – 인스턴스에 연결된 클라이언트 세션의 수입니다.

일반적으로 성능 지표에 허용되는 값은 기준이 무엇인지 그리고 애플리케이션 무엇을 수행하는지에 따라 다릅니다. 기준과의 일관된 차이 또는 추세를 조사하세요.

특정 지표 유형에 대한 권장 사항 및 조언은 다음과 같습니다.

- CPU 사용량이 많음 - CPU 사용량이 많을 경우 해당 애플리케이션의 목표(처리량 또는 동시성 등)와 일치하고 예상되는 결과라면 문제가 되지 않을 수 있습니다. CPU 소비가 지속적으로 80%를 초과하는 경우 인스턴스 확장을 고려하세요.
- 높은 RAM 소비 - `FreeableMemory` 메트릭이 총 인스턴스 메모리의 10% 이하로 자주 떨어지는 경우 인스턴스 확장을 고려하세요. DocumentDB 인스턴스의 메모리 사용량이 높을 때 어떤 일이 발생하는지에 대한 자세한 내용은 Amazon DocumentDB 리소스 거버넌스를 참조하세요.
- 스왑 사용량 - 이 메트릭은 0 또는 그 근처에 있어야 합니다. 스왑 사용량이 많은 경우 인스턴스 확장을 고려하세요.
- 네트워크 트래픽 - 네트워크 트래픽의 경우 시스템 관리자에게 문의하여 해당 도메인 네트워크 및 인터넷 연결의 예상 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. 인스턴스에 대한 최적의 사용자 연결 수는 해당 인스턴스 클래스와 수행하는 작업의 복잡성에 따라 다릅니다. 성능 지표와 관련하여 문제가 있을 경우 성능을 향상

하기 위해 제일 먼저 할 수 있는 것은, 가장 많이 사용되고 가장 비용이 높은 쿼리를 튜닝하여 시스템 리소스에 대한 부하를 줄일 수 있는지 확인하는 것입니다.

쿼리를 튜닝해도 문제가 지속된다면 문제와 관련된 리소스(CPU, RAM, 디스크 스페이스, 네트워크 대역폭, I/O 용량)가 많은 Amazon DocumentDB 인스턴스 클래스로 업그레이드하는 것이 좋습니다.

CloudWatch 지표를 사용한 Amazon DocumentDB 인스턴스 사용량 평가

CloudWatch 지표를 사용하여 인스턴스 처리량(throughput)을 관찰하고 인스턴스 클래스가 애플리케이션에 제공하는 리소스가 충분한지 확인할 수 있습니다. 인스턴스 클래스 한도에 대한 자세한 내용은 인스턴스 제한에서 인스턴스 클래스의 사양을 찾아 네트워크 성능을 확인하세요.

인스턴스 사용량이 인스턴스 클래스 한도에 가까워지면 성능이 느려질 수 있습니다. CloudWatch 지표를 통해 이러한 상황을 확인할 수 있으므로 더 큰 인스턴스 클래스로 수동 스케일 업할 계획을 세울 수 있습니다.

다음 CloudWatch 지표 값을 조합하여 인스턴스 클래스 한도에 가까워졌는지 확인하세요.

- NetworkThroughput – Amazon DocumentDB 클러스터의 각 인스턴스에 대해 클라이언트가 수신 및 전송한 네트워크 처리량(throughput)입니다. 이 처리량(throughput) 값에서 클러스터의 인스턴스와 클러스터 볼륨 간 네트워크 트래픽은 제외됩니다.
- StorageNetworkThroughput - Amazon DocumentDB 클러스터의 각 인스턴스가 수신하여 Amazon DocumentDB 클러스터 스토리지 볼륨으로 전송하는 네트워크 처리량입니다.

NetworkThroughput을 StorageNetworkThroughput에 추가하여 Amazon DocumentDB 클러스터의 각 인스턴스가 Amazon DocumentDB에서 수신 및 전송한 네트워크 처리량(throughput)을 확인하세요. 인스턴스에 대한 인스턴스 클래스 한도는 이 두 지표를 합한 값보다 커야 합니다.

다음 지표를 사용하여 전송 및 수신 시 클라이언트 애플리케이션의 네트워크 트래픽에 대한 추가 세부 정보를 검토할 수 있습니다.

- NetworkReceiveThroughput – Amazon DocumentDB 클러스터의 각 인스턴스가 클라이언트로부터 수신한 네트워크 처리량(throughput)입니다. 이 처리량에서 클러스터의 인스턴스와 클러스터 스토리지 볼륨 간 네트워크 트래픽은 제외됩니다.
- NetworkTransmitThroughput – Amazon DocumentDB 클러스터의 각 인스턴스가 클라이언트로 전송한 네트워크 처리량(throughput)입니다. 이 처리량에서 클러스터의 인스턴스와 클러스터 스토리지 볼륨 간 네트워크 트래픽은 제외됩니다.

- StorageNetworkThroughput - 클러스터의 각 인스턴스에 의해 Amazon DocumentDB 클러스터 스토리지 볼륨에서 수신한 네트워크 처리량(throughput)입니다.
- StorageNetworkTransmitThroughput - 클러스터의 각 인스턴스에 의해 Amazon DocumentDB 클러스터 스토리지 볼륨에서 수신한 네트워크 처리량(throughput)입니다.

이러한 지표를 모두 더해 네트워크 사용량을 인스턴스 클래스 한도와 비교하는 방법을 평가합니다. 인스턴스 클래스 한도는 이러한 결합 지표를 합한 값보다 커야 합니다.

인스턴스의 네트워크 한도와 CPU 사용률은 상호적인 관계가 있습니다. 네트워크 처리량(throughput)이 증가하면 CPU 사용률도 증가합니다. CPU 및 네트워크 사용량을 모니터링하면 리소스가 소진되는 방식 및 이유에 대한 정보를 얻을 수 있습니다.

네트워크 사용을 최소화하려면 다음을 고려하면 됩니다.

- 규모가 더 큰 인스턴스 클래스를 사용합니다.
- 쓰기 요청을 일괄적으로 나누어 전체 트랜잭션을 줄입니다.
- 읽기 전용 워크로드를 읽기 전용 인스턴스로 지정합니다.
- 사용하지 않는 인덱스를 삭제합니다.

쿼리 튜닝

클러스터 성능을 향상하는 제일 좋은 방법 중 하나는 일반적으로 가장 많이 사용하는 쿼리와 리소스를 가장 많이 사용하는 쿼리를 튜닝하여 실행 비용을 낮추는 것입니다.

프로파일러(Amazon DocumentDB 작업 프로파일링 참조)를 사용하여 클러스터에서 수행된 작업의 실행 시간 및 세부 정보를 기록할 수 있습니다. 프로파일러는 개별 쿼리 성능 및 전체 클러스터 성능을 개선하기 위해 클러스터에서 가장 느린 작업을 모니터링하는 경우에 유용합니다.

`explain` 명령을 사용하여 특정 쿼리에 대한 쿼리 계획을 분석하는 방법을 배울 수도 있습니다. 쿼리 성능을 높이기 위해 이 정보를 사용하여 쿼리나 기본 컬렉션을 수정할 수 있습니다(예: 인덱스 추가).

TTL 및 시계열 워크로드

TTL 인덱스 만료로 인한 문서 삭제가 최선의 방법입니다. 문서는 특정 기간 내에 삭제될 수 없습니다. 인스턴스 크기, 인스턴스 리소스 사용률, 문서 크기, 전체 처리량, 인덱스 수, 메모리에 인덱스 및 작업 집합이 맞는지 여부와 같은 요소는 모두 만료된 문서가 TTL 프로세스에 의해 삭제되는 시점에 영향을 줄 있습니다.

TTL 모니터가 문서를 삭제하면 삭제할 때마다 IO 비용이 발생하므로 청구 금액이 증가합니다. 처리량과 TTL 삭제율이 증가하면 I/O 사용량 증가로 인해 청구 금액이 더 증가하기 마련입니다. 하지만 TTL 인덱스를 생성하여 문서를 삭제하지 않고 시간을 기준으로 문서를 컬렉션으로 분류하고 더 이상 필요하지 않을 때 해당 컬렉션을 삭제하면 IO 비용이 발생하지 않습니다. 이 방법은 TTL 인덱스를 사용하는 것보다 훨씬 더 비용 효율적일 수 있습니다.

시계열 워크로드의 경우, 롤링 컬렉션이 데이터를 삭제하는 더 좋은 방법일 수 있고 I/O 사용량이 적기 때문에, TTL 인덱스 대신 롤링 컬렉션을 만드는 것이 좋을 수 있습니다. 대용량 컬렉션(특히 1TB를 초과하는 컬렉션)이 있거나 TTL 삭제 I/O 비용이 우려되는 경우 시간을 기준으로 문서를 컬렉션으로 분할하고 문서가 더 이상 필요하지 않을 때 컬렉션을 삭제하는 것이 좋습니다. 데이터 수집 속도에 따라 하루에 한 번 또는 일주일에 한 번씩 모음을 만들 수 있습니다. 요구 사항은 애플리케이션에 따라 다르지만 몇 개의 큰 컬렉션보다는 더 작은 컬렉션을 사용하는 것이 좋습니다. 이러한 컬렉션을 삭제하는 데는 IO 비용이 발생하지 않으므로 TTL 인덱스를 사용하는 것보다 더 빠르고 비용 효율적일 수 있습니다.

마이그레이션

데이터를 Amazon DocumentDB로 마이그레이션할 때, 데이터를 마이그레이션하기 전에 먼저 Amazon DocumentDB에서 인덱스를 생성하는 것이 가장 좋습니다. 인덱스를 먼저 생성하면 전체 시간을 줄이고 마이그레이션 속도를 높일 수 있습니다. 이렇게 하려면 Amazon DocumentDB 인덱스 도구를 사용하면 됩니다. 마이그레이션에 대한 자세한 내용은 Amazon DocumentDB 마이그레이션 안내서를 참조하세요.

또한 프로덕션 데이터베이스를 마이그레이션하기 전에 기능, 성능, 운영 및 비용을 고려하여 Amazon DocumentDB에서 애플리케이션을 완전히 테스트하는 것이 좋습니다.

클러스터 파라미터 그룹 작업

클러스터 파라미터 그룹 변경 내용을 프로덕션 클러스터에 적용하기 전에 테스트 클러스터에 적용해 보는 것이 좋습니다. 클러스터 백업에 대한 자세한 내용은 Amazon DocumentDB에서 백업 및 복원 섹션을 참조하세요.

집계 파이프라인 쿼리

여러 단계로 집계 파이프라인 쿼리를 생성하고 쿼리에 있는 데이터의 하위 집합만 평가하는 경우 `$match` 단계를 첫 번째 단계 또는 파이프라인의 시작으로 사용합니다. `$match` 첫 번째를 사용하면 집계 파이프라인 쿼리의 후속 단계에서 처리해야 하는 문서 수가 줄어들어 쿼리 성능이 향상됩니다.

batchInsert 및 batchUpdate

높은 비율의 동시 batchInsert 및/또는 batchUpdate 작업을 수행할 때 기본 인스턴스에서 FreeableMemory(CloudWatch Metric)의 양이 0이 되는 경우 일괄 삽입 또는 업데이트 워크로드의 동시성을 줄이거나, 워크로드의 동시성을 줄일 수 없는 경우 FreeableMemory의 양을 늘리려면 인스턴스 크기를 늘립니다.

기능적 차이: Amazon DocumentDB 및 MongoDB

다음은 Amazon DocumentDB(MongoDB 호환)과 MongoDB 간의 기능적 차이입니다.

주제

- Amazon DocumentDB의 기능적 이점
- 업데이트된 기능적 차이
- MongoDB와 기능적 차이

Amazon DocumentDB의 기능적 이점

암시적 트랜잭션

Amazon DocumentDB에서 모든 CRUD 문(`findAndModify`, `update`, `insert`, `delete`)은 여러 문서를 수정하는 작업의 경우에도 원자성과 일관성을 보장합니다. Amazon DocumentDB 4.0이 출시되면서, 이제 다중 명령문 및 다중 컬렉션 작업에 대한 ACID 속성을 제공하는 명시적 트랜잭션이 지원됩니다. Amazon DocumentDB에서의 트랜잭션 사용에 대한 자세한 내용은 Amazon DocumentDB의 트랜잭션을 참조하세요.

다음은 Amazon DocumentDB에서 원자성 동작과 일관성 동작을 모두 충족하는 여러 문서를 수정하는 작업의 예입니다.

```
db.miles.update(
    { "credit_card": { $eq: true } },
    { $mul: { "flight_miles.$[]": NumberInt(2) } },
    { multi: true }
)
```

```
db.miles.updateMany(
    { "credit_card": { $eq: true } },
    { $mul: { "flight_miles.$[]": NumberInt(2) } }
)
```

```
db.runCommand({
```

```
  update: "miles",
  updates: [
    {
      q: { "credit_card": { $eq: true } },
      u: { $mul: { "flight_miles.$[]": NumberInt(2) } },
      multi: true
    }
  ]
})
```

```
db.products.deleteMany({
  "cost": { $gt: 30.00 }
})
```

```
db.runCommand({
  delete: "products",
  deletes: [{ q: { "cost": { $gt: 30.00 } }, limit: 0 }]
})
```

updateMany 및 deleteMany와 같은 대량 작업을 구성하는 개별 작업은 원자성이지만 전체 대량 작업은 원자성이 아닙니다. 예를 들어, 개별 삽입 작업이 오류 없이 성공적으로 실행되면 전체 insertMany 작업이 원자성입니다. insertMany 작업에 오류가 발생하면 insertMany 작업 내의 각 개별 삽입 명령문이 원자성 작업으로 실행됩니다. insertMany, updateMany및 deleteMany 작업에 ACID 속성이 필요한 경우 트랜잭션을 사용하는 것이 좋습니다.

업데이트된 기능적 차이

Amazon DocumentDB에서는 고객이 빌드하도록 요청한 기능에서 거꾸로 작업함으로써 MongoDB와의 호환성을 지속적으로 개선하고 있습니다. 이 섹션에는 고객이 더 쉽게 마이그레이션하고 애플리케이션을 빌드할 수 있도록 Amazon DocumentDB에서 제거한 기능적 차이가 포함되어 있습니다.

주제

- 배열 인덱싱
- 다중 키 인덱스
- 문자열의 null 문자

- 역할 기반 액세스 제어
- $regex 인덱싱
- 중첩된 문서에 대한 프로젝션

배열 인덱싱

2020년 4월 23일 현재 Amazon DocumentDB에서는 2,048바이트보다 큰 배열을 인덱싱하는 기능을 지원합니다. 배열의 개별 키에 대한 제한은 여전히 MongoDB와 일치하는 2,048바이트로 유지됩니다.

새 인덱스를 생성하는 경우, 향상된 기능을 활용하기 위해 아무런 작업도 필요하지 않습니다. 기존 인덱스가 있는 경우, 인덱스를 삭제한 다음, 다시 생성하여 향상된 기능을 활용할 수 있습니다. 향상된 기능을 갖춘 현재 인덱스 버전은 `"v" : 3`입니다.

> **Note**
>
> 프로덕션 클러스터의 경우 인덱스를 삭제하면 애플리케이션 성능에 영향을 줄 수 있습니다. 프로덕션 시스템을 변경할 때는 먼저 테스트하고 주의해서 진행하는 것이 좋습니다. 또한 인덱스를 다시 생성하는 데 걸리는 시간은 컬렉션의 전체 데이터 크기의 함수입니다.

다음 명령을 사용하여 인덱스의 버전을 쿼리할 수 있습니다.

```
db.collection.getIndexes()
```

이 작업의 출력은 다음과 같이 표시됩니다. 이 출력에서 인덱스의 버전은 가장 최신 인덱스 버전인 `"v" : 3`입니다,

```
[
    {
        "v" : 3,
        "key" : {
        "_id" : 1
        },
        "name" : "_id_",
        "ns" : "test.test"
    }
]
```

다중 키 인덱스

2020년 4월 23일 현재 Amazon DocumentDB에서는 동일한 배열에 여러 키가 있는 복합 인덱스를 만드는 기능을 지원합니다.

새 인덱스를 생성하는 경우, 향상된 기능을 활용하기 위해 아무런 작업도 필요하지 않습니다. 기존 인덱스가 있는 경우, 인덱스를 삭제한 다음, 다시 생성하여 향상된 기능을 활용할 수 있습니다. 향상된 기능을 갖춘 현재 인덱스 버전은 `"v" : 3`입니다.

> **Note**
>
> 프로덕션 클러스터의 경우 인덱스를 삭제하면 애플리케이션 성능에 영향을 줄 수 있습니다. 프로덕션 시스템을 변경할 때는 먼저 테스트하고 주의해서 진행하는 것이 좋습니다. 또한 인덱스를 다시 생성하는 데 걸리는 시간은 컬렉션의 전체 데이터 크기의 함수입니다.

다음 명령을 사용하여 인덱스의 버전을 쿼리할 수 있습니다.

```
db.collection.getIndexes()
```

이 작업의 출력은 다음과 같이 표시됩니다. 이 출력에서 인덱스의 버전은 가장 최신 인덱스 버전인 `"v" : 3`입니다,

```
[
    {
        "v" : 3,
        "key" : {
            "_id" : 1
        },
        "name" : "_id_",
        "ns" : "test.test"
    }
]
```

문자열의 null 문자

2020년 6월 22일부터 Amazon DocumentDB는 문자열에서 null 문자(`'\0'`)를 지원합니다.

역할 기반 액세스 제어

2020년 3월 26일부터 Amazon DocumentDB는 기본 제공 역할에 대한 RBAC(역할 기반 액세스 제어)를 지원합니다. 자세한 내용은 역할 기반 액세스 제어을 참조하세요.

$regex 인덱싱

2020년 6월 22일부터 Amazon DocumentDB는 $regex 연산자가 인덱스를 활용하는 기능을 지원합니다.

$regex 연산자와 함께 인덱스를 활용하려면 hint() 명령을 사용해야 합니다. hint()를 사용할 때 $regex를 적용할 필드의 이름을 지정해야 합니다. 예를 들어 product 필드에 인덱스 이름이 p_1인 인덱스가 있으면 db.foo.find({product: /^x.*/}).hint({product:1})은 p_1 인덱스를 사용하지만 db.foo.find({product: /^x.*/}).hint("p_1")은 인덱스를 사용하지 않습니다. explain() 명령을 사용하거나 느린 쿼리를 로깅하는 프로파일러를 사용하여 인덱스가 선택되었는지 확인할 수 있습니다. 예: db.foo.find({product: /^x.*/}).hint("p_1").explain().

> **Note**
>
> hint() 메서드는 한 번에 하나의 인덱스에만 사용할 수 있습니다.

$regex 쿼리에 대한 인덱스 사용은 접두사를 사용하고 i, m 또는 정규식 옵션을 지정하지 않는 o 정규식 쿼리에 최적화됩니다.

$regex와 함께 인덱스를 사용하는 경우 중복 값 수가 컬렉션에 있는 총 문서 수의 1% 미만인 매우 선택적인 필드에 인덱스를 생성하는 것이 좋습니다. 예를 들어 컬렉션에 100,000개의 문서가 포함되어 있는 경우, 동일한 값이 1000회 이하로 나타나는 필드에만 인덱스를 생성하세요.

중첩된 문서에 대한 프로젝션

버전 3.6의 Amazon DocumentDB와 MongoDB 간에는 $project 연산자와의 기능적 차이가 있습니다. 이 차이는 Amazon DocumentDB 4.0에서는 해결되었지만 Amazon DocumentDB 3.6에서는 지원되지 않습니다.

Amazon DocumentDB 3.6은 프로젝션을 적용할 때 중첩 문서의 첫 번째 필드만 고려하는 반면, MongoDB 3.6은 하위 문서를 구문 분석하여 각 하위 문서에도 프로젝션을 적용합니다.

예를 들어, 프로젝션이 "a.b.c": 1이면, Amazon DocumentDB와 MongoDB 모두에서 동작이 예상대로 작동합니다. 그러나 프로젝션이 {a:{b:{c:1}}}이면, Amazon DocumentDB 3.6은 프로젝션을

a에만 적용하고 b 또는 c에는 적용하지 않습니다. Amazon DocumentDB 4.0에서는 프로젝션 `{a:{b:{c:1}}}`, a, b, 및 c에 적용됩니다.

MongoDB와 기능적 차이

주제

- $vectorSearch 연산자
- OpCountersCommand
- 관리자 데이터베이스 및 컬렉션
- cursormaxTimeMS
- explain()
- 인덱스 빌드
- 경로에 빈 키를 사용하여 조회
- MongoDB API, 작업 및 데이터 형식
- mongodump및 mongorestore 유틸리티
- 결과 주문
- 재시도 가능한 쓰기
- 희소 인덱스
- $all 표현식 내에서 $elemMatch 사용
- $ne, $nin, $nor, $not, $exists, 및 $elemMatch 인덱싱
- 필드 이름의 달러($) 및 점(.)
- $lookup
- $natural 및 역 정렬

`$vectorSearch` 연산자

Amazon DocumentDB는 독립 연산자로 `$vectorSearch` 항목을 지원하지 않습니다. 대신 `$search` 연산자 내에서 `vectorSearch` 항목을 지원합니다. 자세한 내용은 Amazon DocumentDB에 대한 벡터 검색 단원을 참조하십시오.

OpCountersCommand

Amazon DocumentDB의 OpCountersCommand 동작은 다음과 같이 MongoDB의 opcounters.command에서 벗어납니다.

- MongoDB opcounters.command 개수는 삽입, 업데이트 및 삭제를 제외한 모든 명령을 계산하는 반면, Amazon DocumentDB의 OpCountersCommand는 해당 find 명령도 제외합니다.
- Amazon DocumentDB는 내부 명령을 OpCountersCommand로 계산합니다.

관리자 데이터베이스 및 컬렉션

Amazon DocumentDB는 각각 관리자 또는 로컬 데이터베이스, MongoDB system.* 또는 startup_log 컬렉션을 지원하지 않습니다.

cursormaxTimeMS

Amazon DocumentDB에서 cursor.maxTimeMS은 각 getMore 요청에 대해 카운터를 재설정합니다. 따라서 3000MS maxTimeMS을 지정하면 쿼리에 2800MS가 걸리고 이후의 각 getMore 요청에는 300MS가 걸리므로, 커서가 타임아웃되지 않습니다. 쿼리 또는 개별 getMore 요청 중 단일 작업에 지정된 maxTimeMS보다 많은 시간이 걸리는 경우에만 커서가 시간 초과됩니다. 또한, 커서 실행 시간을 확인하는 스위퍼는 5분 단위로 실행됩니다.

explain()

Amazon DocumentDB는 분산된 내결함성 자체 복구 스토리지 시스템을 활용하는 특수 빌드된 데이터베이스 엔진에서 MongoDB 3.6, 4.0 및 5.0 APIs를 에뮬레이션합니다. 그 결과, 쿼리 계획과 explain()의 출력은 Amazon DocumentDB와 MongoDB 간에 다를 수 있습니다. 쿼리 계획을 제어하려는 고객은 $hint 연산자를 사용하여 기본 인덱스를 선택할 수 있습니다.

인덱스 빌드

Amazon DocumentDB는 특정 시간에 모음에서 발생하는 배경 인덱스 빌드를 하나만 허용합니다. 포그라운드 또는 백그라운드 둘 다에서. 인덱스 빌드가 현재 진행 중일 때 같은 컬렉션에서 createIndex() 또는 dropIndex()와 같은 작업이 발생하면 새로 시도한 작업이 실패합니다.

기본적으로 Amazon DocumentDB 및 MongoDB 버전 4.0의 인덱스 빌드는 백그라운드에서 실행됩니다. MongoDB 버전 4.2 이상에서는 createIndexes 또는 해당 쉘 도우미 createIndex() 및 createIndexes()에 지정된 경우, 백그라운드 인덱스 빌드 옵션을 무시합니다.

TTL(Time to Live) 인덱스는 배경 인덱스 빌드가 완료된 이후에 문서 만료를 시작합니다.

경로에 빈 키를 사용하여 조회

경로의 일부로 빈 문자열이 포함된 키(예: `x.`, `x..b`)를 찾고, 객체의 배열 내에 빈 문자열 키 경로가 있는 경우(예: `{"x" : [ { "" : 10 }, { "b" : 20 } ]}`), Amazon DocumentDB는 MongoDB에서 동일한 검색을 실행한 경우와 다른 결과를 반환합니다.

MongoDB에서, 배열 내 빈 키 경로 조회는 빈 문자열 키가 경로 조회의 끝에 있지 않을 때 예상대로 작동합니다. 하지만 경로 조회 끝에 빈 문자열 키가 있으면 배열을 들여다보지 않습니다.

하지만 Amazon DocumentDB에서는 `getArrayIndexFromKeyString`은 빈 문자열을 `0`로 변환하므로 배열 내의 첫 번째 요소만 읽습니다. 따라서 문자열 키 조회는 배열 인덱스 검색으로 취급됩니다.

MongoDB API, 작업 및 데이터 형식

Amazon DocumentDB는 MongoDB 3.6 및 5.0 API와 호환됩니다. 지원되는 기능의 최신 목록을 보려면 Amazon DocumentDB에서 지원되는 MongoDB API, 작업 및 데이터 형식 섹션을 참조하세요.

`mongodump`및 `mongorestore` 유틸리티

Amazon DocumentDB는 관리자 데이터베이스를 지원하지 않으며, 따라서 `mongodump` 또는 `mongorestore` 유틸리티를 사용할 때 관리자 데이터베이스를 덤프하거나 복원하지 않습니다. Amazon DocumentDB에서 `mongorestore`을 사용하여 새 데이터베이스를 생성할 경우 복원 작업 이외에 사용자 역할을 다시 생성해야 합니다.

> **Note**
>
> Amazon DocumentDB의 경우 버전 100.6.1까지의 MongoDB 데이터베이스 도구를 사용하는 것이 좋습니다. 여기에서 MongoDB 데이터베이스 도구 다운로드에 액세스할 수 있습니다.

결과 주문

Amazon DocumentDB는 결과 집합의 암시적 결과 정렬 순서를 보장하지 않습니다. 결과 집합의 순서를 보장하려면 `sort()`를 사용하여 정렬 순서를 명시적으로 지정합니다.

다음 예에서는 재고 수집의 항목을 재고 필드를 기준으로 내림차순으로 정렬합니다.

```
db.inventory.find().sort({ stock: -1 })
```

$sort 집계 단계를 사용할 경우 해당 단계가 집계 파이프라인의 마지막 $sort 단계가 아니면 정렬 순서가 보존되지 않습니다. $sort 집계 단계를 $group집계 단계와 함께 사용하는 경우 $sort 집계 단계는 $first 및 $last 누적기에만 적용됩니다. Amazon DocumentDB 4.0에서는 이전 $sort 단계의 정렬 순서를 준수하기 위한 지원이 $push에 추가되었습니다.

재시도 가능한 쓰기

MongoDB 4.2 호환 드라이버부터 재시도 가능한 쓰기가 기본적으로 활성화됩니다. 그러나 Amazon DocumentDB는 현재 재시도 가능한 쓰기는 지원하지 않습니다. 기능상의 차이점은 다음과 유사한 오류 메시지로 나타납니다.

```
{"ok":0,"errmsg":"Unrecognized field: 'txnNumber'","code":9,"name":"MongoError"}
```

연결 문자열(예: MongoClient("mongodb://my.mongodb.cluster/db?retryWrites=false")) 또는 MongoClient 생성자의 키워드 인수(예:)를 통해 재시도 가능한 쓰기를 비활성화할 수 있습니다MongoClient("mongodb://my.mongodb.cluster/db", retryWrites=False).

다음은 연결 문자열에서 재시도 가능한 쓰기를 비활성화하는 Python 예제입니다.

```
client =
 pymongo.MongoClient('mongodb://
<username>:<password>@docdb-2019-03-17-16-49-12.cluster-ccuszbx3pn5e.us-
east-1.docdb.amazonaws.com:27017/?
replicaSet=rs0',w='majority',j=True,retryWrites=False)
```

희소 인덱스

쿼리에서 생성된 희소 인덱스를 사용하려면 인덱스를 포함하는 필드에서 $exists 절을 사용해야 합니다. 를 생략하면 $exists Amazon DocumentDB는 희소 인덱스를 사용하지 않습니다.

다음은 예입니다.

```
db.inventory.count({ "stock": { $exists: true }})
```

스파스 다중 키 인덱스의 경우 Amazon DocumentDB는 문서 조회 결과로 값 집합이 생성되고 인덱싱된 필드의 하위 집합만 누락된 경우 고유 키 제약 조건을 지원하지 않습니다. 예를 들어,

"a.c"가 인덱스에 저장되므로 "a" : [{ "b" : 2 }, { "c" : 1 }]의 입력이 주어지면 createIndex({"a.b" : 1 }, { unique : true, sparse :true })는 지원되지 않습니다.

$all 표현식 내에서 $elemMatch 사용

Amazon DocumentDB는 현재 $all 표현식 내에서 $elemMatch 연산자 사용을 지원하지 않습니다. 차선책으로 다음과 같이 $elemMatch와 함께 $and 연산자를 사용할 수 있습니다.

원래 작업:

```
db.col.find({
  qty: {
    $all: [
      { "$elemMatch": { part: "xyz", qty: { $lt: 11 } } },
      { "$elemMatch": { num: 40, size: "XL" } }
    ]
  }
})
```

업데이트된 작업:

```
db.col.find({
  $and: [
    { qty: { "$elemMatch": { part: "xyz", qty: { $lt: 11 } } } },
    { qty: { "$elemMatch": { qty: 40, size: "XL" } } }
  ]
})
```

$ne, $nin, $nor, $not, $exists, 및 $elemMatch 인덱싱

Amazon DocumentDB는 현재 $ne, $nin, $nor, $not $exists 및 $distinct 연산자와 함께 인덱스를 사용하는 기능을 지원하지 않습니다. 따라서 이러한 연산자를 사용하면 수집 스캔이 발생합니다. 이러한 연산자 중 하나를 사용하기 전에 필터 또는 일치를 수행하면 스캔해야 하는 데이터의 양이 줄어들어 성능이 향상될 수 있습니다.

Amazon DocumentDB는 Amazon DocumentDB 5.0 및 엘라스틱 클러스터에서 $elemMatch 운영자를 통한 인덱스 스캔에 대한 지원을 추가했습니다. 쿼리 전용 필터에 한 레벨의 $elemMatch 필터가 있는 경우 인덱스 스캔이 지원되지만 중첩된 $elemMatch 쿼리가 포함된 경우에는 필터가 지원되지 않습니다.

`$elemMatch`Amazon DocumentDB 5.0의 인덱스 스캔을 지원하는 쿼리 셰이프:

```
db.foo.find( { "a": {$elemMatch: { "b": "xyz", "c": "abc"} } })
```

`$elemMatch`Amazon DocumentDB 5.0의 인덱스 스캔을 지원하지 않는 쿼리 셰이프:

```
db.foo.find( { "a": {$elemMatch: { "b": {$elemMatch: { "d": "xyz", "e": "abc"} }} } })
```

필드 이름의 달러($) 및 점(.)

Amazon DocumentDB는 중첩된 객체에서 $in, $nin 및 $all의 Dollar($) 접두사 필드 쿼리를 지원하지 않습니다. 예를 들어 Amazon DocumentDB에서는 다음 쿼리가 유효하지 않습니다.

```
coll.find({"field": {"$all": [{ "$a": 1 }]}})
```

$lookup

Amazon DocumentDB는 균등 일치(예: 왼쪽 외부 조인)를 수행할 수 있는 기능을 지원하고 상관 관계가 없는 하위 쿼리를 지원하지만, 상관 관계가 있는 하위 쿼리를 지원하지 않습니다.

$lookup과 인덱스를 활용하면

이제 `$lookup` 스테이지 연산자를 사용하여 인덱스를 활용할 수 있습니다. 사용 사례에 따라 성능을 최적화하는 데 사용할 수 있는 여러 인덱싱 알고리즘이 있습니다. 이 섹션에서는 `$lookup`을 위한 다양한 인덱싱 알고리즘을 설명하고 워크로드에 가장 적합한 알고리즘을 선택하는 데 도움이 됩니다.

기본적으로 Amazon DocumentDB는 `allowDiskUse:false`을 사용할 때는 해시 알고리즘을 활용하고, `allowDiskUse:true`를 사용할 때는 정렬 병합을 사용합니다.

> **Note**
>
> 이 `allowDiskUse` 옵션은 현재 `find` 명령에 지원되지 않습니다. 옵션은 집계의 일부로만 지원됩니다. 메모리 제한을 초과할 수 있는 대규모 쿼리를 처리하려면 `allowDiskUse:true` 항목과 함께 집계 프레임워크를 사용하는 것이 좋습니다.

일부 사용 사례의 경우 쿼리 최적화 프로그램이 다른 알고리즘을 사용하도록 강제하는 것이 바람직할 수 있습니다. `$lookup` 집계 연산자가 이용할 수 있는 다양한 인덱싱 알고리즘은 다음과 같습니다.

- 중첩 루프: 외부 컬렉션이 1GB 미만이고 외부 컬렉션의 필드에 인덱스가 있는 경우, 일반적으로 중첩 루프 계획이 워크로드에 유용합니다. 중첩 루프 알고리즘을 사용하는 경우 계획 설명에는 단계가 NESTED_LOOP_LOOKUP로 표시됩니다.
- 정렬 병합: 외부 컬렉션에 조회에 사용되는 필드에 대한 색인이 없고 작업 데이터셋이 메모리에 맞지 않는 경우 일반적으로 정렬 병합 계획이 워크로드에 유용합니다. 정렬 병합 알고리즘을 사용하는 경우 설명 계획에는 단계가 SORT_LOOKUP로 표시됩니다.
- 해시: 해시 계획은 일반적으로 외래 컬렉션이 < 1GB이고 작업 데이터 세트가 메모리에 맞는 경우 워크로드에 유용합니다. 해시 알고리즘을 사용하는 경우 설명 계획에는 단계가 HASH_LOOKUP로 표시됩니다.

쿼리에서를 사용하여 $lookup 연산자에 사용 중인 인덱싱 알고리즘을 식별할 수 explain 있습니다. 아래는 하나의 예제입니다.

```
db.localCollection.explain().aggregate(
   [
      {
         $lookup:
            {
               from: "foreignCollection",
               localField: "a",
               foreignField: "b",
               as: "joined"
            }
      }
   ]
)

output
{
    "queryPlanner" : {
        "plannerVersion" : 1,
        "namespace" : "test.localCollection",
        "winningPlan" : {
            "stage" : "SUBSCAN",
            "inputStage" : {
                "stage" : "SORT_AGGREGATE",
                "inputStage" : {
                    "stage" : "SORT",
                    "inputStage" : {
```

```
                        "stage" : "NESTED_LOOP_LOOKUP",
                        "inputStages" : [
                            {
                                "stage" : "COLLSCAN"
                            },
                            {
                                "stage" : "FETCH",
                                "inputStage" : {
                                    "stage" : "COLLSCAN"
                                }
                            }
                        ]
                    }
                }
            }
        }
    },
    "serverInfo" : {
        "host" : "devbox-test",
        "port" : 27317,
        "version" : "3.6.0"
    },
    "ok" : 1
}
```

explain() 메서드를 사용하는 대신 프로파일러를 사용하여 $lookup 연산자 사용 시 사용 중인 알고리즘을 검토할 수 있습니다. 이 프로파일에 대한 자세한 정보는 Amazon DocumentDB 작업 프로파일링 섹션을 참조하세요.

planHint 사용

쿼리 최적화 프로그램에서 $lookup에 다른 인덱싱 알고리즘을 사용하도록 하려면 planHint를 사용할 수 있습니다. 이렇게 하려면 집계 단계 옵션의 설명을 사용하여 다른 계획을 적용하세요. 다음은 주석 구문의 예시입니다.

```
comment : {
    comment :  "<string>",
    lookupStage : { planHint : "SORT" | "HASH" | "NESTED_LOOP" }
}
```

다음은 planHint을 사용하여 쿼리 최적화 프로그램이 HASH 인덱싱 알고리즘을 사용하도록 강제하는 예입니다.

```
db.foo.aggregate(
   [
      {
         $lookup:
            {
               from: "foo",
               localField: "_id",
               foreignField: "_id",
               as: "joined"
            },
      }
   ]
),
{
   comment : "{ \"lookupStage\" : { \"planHint\": \"HASH\" }}"
```

워크로드에 가장 적합한 알고리즘을 테스트하려면 explain 메서드의 executionStats 파라미터를 사용하여 $lookup 스테이지의 실행 시간을 측정하고 인덱싱 알고리즘을 수정할 수 있습니다(즉, HASH/SORT/NESTED_LOOP).

다음 예제는 SORT 알고리즘을 사용하여 $lookup 스테이지의 실행 시간을 측정하는 데 executionStats을 사용하는 방법을 보여줍니다.

```
db.foo.explain("executionStats").aggregate(
   [
      {
         $lookup:
            {
               from: "foo",
               localField: "_id",
               foreignField: "_id",
               as: "joined"
            },
      }
   ]
),
{
```

```
    comment : "{ \"lookupStage\" : { \"planHint\": \"SORT\" }}"
```

$natural 및 역 정렬

Amazon DocumentDB는 $natural 포워드 컬렉션 스캔만 지원합니다. 역수집 스캔({$natural: -1})은 로 이어집니다MongoServerError.

Amazon DocumentDB에서 지원되는 MongoDB API, 작업 및 데이터 형식

Amazon DocumentDB(MongoDB 호환)은 MongoDB 워크로드를 지원하는 빠르고, 확장 가능하며, 가용성이 높은 완전관리형 문서 데이터베이스 서비스입니다. Amazon DocumentDB는 MongoDB 3.6 및 5.0 API와 호환됩니다. 이 단원에서는 지원되는 기능에 대해 설명합니다. MongoDB API 및 드라이버 사용에 대한 지원은 MongoDB 커뮤니티 포럼을 참조하십시오. Amazon DocumentDB 서비스 사용에 대한 지원은 해당 AWS 지원 팀에 문의하십시오. Amazon DocumentDB와 MongoDB 간의 기능적 차이는 기능적 차이: Amazon DocumentDB 및 MongoDB(을)를 참조하세요.

내부 전용이거나 완전 관리형 서비스에 해당되지 않는 MongoDB 명령 및 연산자는 지원되지 않으며, 지원 기능 목록에도 포함되지 않습니다.

출시 이후 50개 이상의 기능을 더 추가했으며 고객이 원하는 기능을 제공하기 위해 앞으로도 계속 노력할 것입니다. 최신 출시에 대한 자세한 내용은 Amazon DocumentDB 공지 사항을 참조하십시오.

지원되지 않는 기능 중 구축을 원하는 기능이 있는 경우, 계정 ID, 요청된 기능, 사용 사례를 포함한 이메일을 Amazon DocumentDB 서비스 팀에 보내 알려주십시오.

주제

- 데이터베이스 명령
- 쿼리 및 프로젝션 연산자
- 업데이트 연산자
- 지리 공간
- 커서 메서드
- 집계 파이프라인 연산자
- 데이터 타입
- 인덱스 및 인덱스 속성

데이터베이스 명령

주제

- 관리 명령

- 집계
- 인증
- 진단 명령
- 쿼리 및 쓰기 작업
- 역할 관리 명령
- 세션 명령
- 사용자 관리
- 샤딩 명령

관리 명령

Command	3.6	4.0	5.0	엘라스틱 클러스터
제한 컬렉션	아니요	아니요	아니요	아니요
cloneCollectionAsCapped	아니요	아니요	아니요	아니요
collMod	부분	부분	부분	부분
collMod: expireAfterSeconds	예	예	예	예
convertToCapped	아니요	아니요	아니요	아니요
copydb	아니요	아니요	아니요	아니요
create	예	예	예	예
createView	아니요	아니요	아니요	아니요
createIndexes	예	예	예	예
currentOp	예	예	예	예

Command	3.6	4.0	5.0	엘라스틱 클러스터
drop	예	예	예	예
dropDatabase	예	예	예	예
dropIndexes	예	예	예	예
filemd5	아니요	아니요	아니요	아니요
getAuditConfig	아니요	예	예	아니요
killCursors	예	예	예	예
killOp	예	예	예	예
listCollections*	예	예	예	예
listDatabases	예	예	예	예
listIndexes	예	예	예	예
reIndex	아니요	아니요	예	아니요
renameCollection	예	예	예	아니요
setAuditConfig	아니요	예	예	아니요

* 필터 옵션의 `type` 키는 지원되지 않습니다.

집계

Command	3.6	4.0	5.0	엘라스틱 클러스터
aggregate	예	예	예	예
count	예	예	예	예

Command	3.6	4.0	5.0	엘라스틱 클러스터
distinct	예	예	예	예
mapReduce	아니요	아니요	아니요	아니요

인증

Command	3.6	4.0	5.0	엘라스틱 클러스터
authenticate	예	예	예	예
로그아웃	예	예	예	예

진단 명령

Command	3.6	4.0	5.0	엘라스틱 클러스터
buildInfo	예	예	예	예
collStats	예	예	예	예
connPoolStats	아니요	아니요	아니요	아니요
connectionStatus	예	예	예	예
dataSize	예	예	예	예
dbHash	아니요	아니요	아니요	아니요
dbStats	예	예	예	예
explain	예	예	예	예

Command	3.6	4.0	5.0	엘라스틱 클러스터
explain: executionStats	예	예	예	예
기능	아니요	아니요	아니요	아니요
hostInfo	예	예	예	예
listCommands	예	예	예	예
profiler	예	예	예	아니요
serverStatus	예	예	예	예
top	예	예	예	예

쿼리 및 쓰기 작업

Command	3.6	4.0	5.0	엘라스틱 클러스터
Change streams	예	예	예	아니요
delete	예	예	예	예
find	예	예	예	예
findAndModify	예	예	예	예
getLastError	아니요	아니요	아니요	아니요
getMore	예	예	예	예
getPrevError	아니요	아니요	아니요	아니요
GridFS	예	예	예	아니요
삽입	예	예	예	예

Command	3.6	4.0	5.0	엘라스틱 클러스터
parallelCollectionScan	아니요	아니요	아니요	아니요
resetError	아니요	아니요	아니요	아니요
업데이트	예	예	예	예
ReplaceOne	예	예	예	예

역할 관리 명령

C	3	4	5	엘 라 스 틱 클 러 스 터
c e	예	예	예	아 니 요
c le ta	예	예	예	아 니a 요
c	예	예	예	아 니 요

C	3	4	5	엘라스틱 클러스터
g s	예	예	예	아니요
r e e	예	예	예	아니요
r v c	예	예	예	아니요
r	예	예	예	아니요
u e	예	예	예	아니요

세션 명령

Command	3.6	4.0	5.0	엘라스틱 클러스터
abortTransaction	아니요	예	예	아니요

Command	3.6	4.0	5.0	엘라스틱 클러스터
commitTransaction	아니요	예	예	아니요
endSessions	아니요	아니요	아니요	아니요
killAllSessions	아니요	예	예	아니요
killAllSessionsByPattern	아니요	아니요	아니요	아니요
killSessions	아니요	예	예	아니요
refreshSessions	아니요	아니요	아니요	아니요
startSession	아니요	예	예	아니요

사용자 관리

Command	3.6	4.0	5.0	엘라스틱 클러스터
createUser	예	예	예	예
dropAllUsersFromDatabase	예	예	예	예
dropUser	예	예	예	예
grantRolesToUser	예	예	예	예
revokeRolesFromUser	예	예	예	예
updateUser	예	예	예	예

Command	3.6	4.0	5.0	엘라스틱 클러스터
usersInfo	예	예	예	예

샤딩 명령

Command	엘라스틱 클러스터
abortReshardCollection	아니요
addShard	아니요
addShardToZone	아니요
balancerCollectionStatus	아니요
balancerStart	아니요
balancerStatus	아니요
balancerStop	아니요
checkShardingIndex	아니요
clearJumboFlag	아니요
cleanupOrphaned	아니요
cleanupReshardCollection	아니요
commitReshardCollection	아니요
enableSharding	예
flushRouterConfig	아니요
getShardMap	아니요
getShardVersion	아니요

Command	엘라스틱 클러스터
isdbgrid	아니요
listShards	아니요
medianKey	아니요
moveChunk	아니요
movePrimary	아니요
mergeChunks	아니요
refineCollectionShardKey	아니요
removeShard	아니요
removeShardFromZone	아니요
reshardCollection	아니요
setAllowMigrations	아니요
setShardVersion	아니요
shardCollection	예
shardingState	아니요
split	아니요
splitVector	아니요
unsetSharding	아니요
updateZoneKeyRange	아니요

쿼리 및 프로젝션 연산자

주제

- 배열 연산자
- Bitwise 연산자
- 설명 연산자
- 비교 연산자
- 요소 연산자
- 평가 쿼리 연산자
- 논리 연산자
- 프로젝션 연산자

배열 연산자

Command	3.6	4.0	5.0	엘라스틱 클러스터
$all	예	예	예	예
$elemMatch	예	예	예	예
$size	예	예	예	예

Bitwise 연산자

Command	3.6	4.0	5.0	엘라스틱 클러스터
$bitsAllSet	예	예	예	예
$bitsAnySet	예	예	예	예
$bitsAllClear	예	예	예	예
$bitsAnyClear	예	예	예	예

설명 연산자

Command	3.6	4.0	5.0	엘라스틱 클러스터
$comment	예	예	예	예

비교 연산자

Command	3.6	4.0	5.0	엘라스틱 클러스터
$eq	예	예	예	예
$gt	예	예	예	예
$gte	예	예	예	예
$in	예	예	예	예
$lt	예	예	예	예
$lte	예	예	예	예
$ne	예	예	예	예
$nin	예	예	예	예

요소 연산자

Command	3.6	4.0	5.0	엘라스틱 클러스터
$exists	예	예	예	예
$type	예	예	예	예

평가 쿼리 연산자

Command	3.6	4.0	5.0	엘라스틱 클러스터
$expr	아니요	예	예	아니요
$jsonSchema	아니요	예	예	아니요
$mod	예	예	예	예
$regex	예	예	예	예
$text	아니요	아니요	예	아니요
$where	아니요	아니요	아니요	아니요

논리 연산자

Command	3.6	4.0	5.0	엘라스틱 클러스터
$and	예	예	예	예
$nor	예	예	예	예
$not	예	예	예	예
$or	예	예	예	예

프로젝션 연산자

Command	3.6	4.0	5.0	엘라스틱 클러스터
$	예	예	예	예

Command	3.6	4.0	5.0	엘라스틱 클러스터
$elemMatch	예	예	예	예
$meta	아니요	아니요	예	아니요
$slice	예	예	예	예

업데이트 연산자

주제

- 배열 연산자
- Bitwise 연산자
- 필드 연산자
- 업데이트 한정자

배열 연산자

Command	3.6	4.0	5.0	엘라스틱 클러스터
$	예	예	예	예
$[]	예	예	예	예
$[<identifier>]	예	예	예	예
$addToSet	예	예	예	예
$pop	예	예	예	예
$pullAll	예	예	예	예
$pull	예	예	예	예
$push	예	예	예	예

Bitwise 연산자

Command	3.6	4.0	5.0	엘라스틱 클러스터
$bit	예	예	예	예

필드 연산자

연산자	3.6	4.0	5.0	엘라스틱 클러스터
$currentDate	예	예	예	예
$inc	예	예	예	예
$max	예	예	예	예
$min	예	예	예	예
$mul	예	예	예	예
$rename	예	예	예	예
$set	예	예	예	예
$setOnInsert	예	예	예	예
$unset	예	예	예	예

업데이트 한정자

연산자	3.6	4.0	5.0	엘라스틱 클러스터
$each	예	예	예	예

연산자	3.6	4.0	5.0	엘라스틱 클러스터
$position	예	예	예	예
$slice	예	예	예	예
$sort	예	예	예	예

지리 공간

지오메트리 지정자

쿼리 선택기	3.6	4.0	5.0	엘라스틱 클러스터
$box	아니요	아니요	아니요	아니요
$center	아니요	아니요	아니요	아니요
$centerSphere	아니요	아니요	아니요	아니요
$geometry	예	예	예	예
$maxDistance	예	예	예	예
$minDistance	예	예	예	예
$nearSphere	예	예	예	예
$polygon	아니요	아니요	아니요	아니요
$uniqueDocs	아니요	아니요	아니요	아니요

쿼리 선택기

Command	3.6	4.0	5.0	엘라스틱 클러스터
$geoIntersects	예	예	예	예
$geoWithin	예	예	예	예
$near	아니요	아니요	아니요	아니요
$nearSphere	예	예	예	예
$polygon	아니요	아니요	아니요	아니요
$uniqueDocs	아니요	아니요	아니요	아니요

커서 메서드

Command	3.6	4.0	5.0	엘라스틱 클러스터
cursor.batchSize()	예	예	예	예
cursor.close()	예	예	예	예
cursor.collation()	아니요	아니요	아니요	아니요
cursor.comment()	예	예	예	예
cursor.count()	예	예	예	예
cursor.explain()	예	예	예	아니요
cursor.forEach()	예	예	예	예
cursor.hasNext()	예	예	예	예

Command	3.6	4.0	5.0	엘라스틱 클러스터
cursor.hint()	예	예	예	예*
cursor.isClosed()	예	예	예	예
cursor.is Exhausted()	예	예	예	아니요
cursor.itcount()	예	예	예	아니요
cursor.limit()	예	예	예	아니요
cursor.map()	예	예	예	아니요
cursor.max()	아니요	아니요	아니요	아니요
cursor.ma xScan()	예	예	예	아니요
cursor.ma xTimeMS()	예	예	예	아니요
cursor.min()	아니요	아니요	아니요	아니요
cursor.next()	예	예	예	예
cursor.no CursorTimeout()	아니요	아니요	아니요	아니요
cursor.ob jsLeftInBatch()	예	예	예	아니요
cursor.pretty()	예	예	예	아니요
cursor.re adConcern()	예	예	예	아니요
cursor.readPref()	예	예	예	아니요

Command	3.6	4.0	5.0	엘라스틱 클러스터
cursor.returnKey()	아니요	아니요	아니요	아니요
cursor.showRecordId()	아니요	아니요	아니요	아니요
cursor.size()	예	예	예	아니요
cursor.skip()	예	예	예	아니요
cursor.sort()	예	예	예	아니요
cursor.tailable()	아니요	아니요	아니요	아니요
cursor.toArray()	예	예	예	아니요

* `hint` 인덱스는 인덱스 표현식과 함께 지원됩니다. 예를 들어 `db.foo.find().hint({x:1})`입니다.

집계 파이프라인 연산자

주제

- 누적기 식
- 산술 연산자
- 배열 연산자
- 부울 연산
- 비교 연산자
- 조건식 연산자
- 데이터 유형 연산자
- 데이터 크기 연산자
- 날짜 연산자
- 리터럴 연산자

- 병합 연산자
- 자연 연산자
- 집합 연산자
- 단계 연산자
- 문자열 연산자
- 시스템 변수
- 텍스트 검색 연산자
- 유형 변환 연산자
- 변수 연산자
- 기타 연산자

누적기 식

표현식	3.6	4.0	5.0	엘라스틱 클러스터
$accumulator	-	-	아니요	아니요
$addToSet	예	예	예	예
$avg	예	예	예	예
$count	-	-	아니요	아니요
$covariancePop	아니요	아니요	아니요	아니요
$covarianceSamp	아니요	아니요	아니요	아니요
$denseRank	아니요	아니요	아니요	아니요
$파생	아니요	아니요	아니요	아니요
$documentNumber	아니요	아니요	아니요	아니요
$expMovingAvg	아니요	아니요	아니요	아니요

표현식	3.6	4.0	5.0	엘라스틱 클러스터
$first	예	예	예	예
$integral	아니요	아니요	아니요	아니요
$last	예	예	예	예
$max	예	예	예	예
$min	예	예	예	예
$push	예	예	예	예
$순위	아니요	아니요	아니요	아니요
$shift	아니요	아니요	아니요	아니요
$stdDevPop	아니요	아니요	아니요	아니요
$stdDevSamp	아니요	아니요	아니요	아니요
$sum	예	예	예	예

산술 연산자

Command	3.6	4.0	5.0	엘라스틱 클러스터
$abs	예	예	예	예
$add	예	예	예	예
$ceil	아니요	예	예	예
$divide	예	예	예	예
$exp	아니요	예	예	예

Command	3.6	4.0	5.0	엘라스틱 클러스터
$floor	아니요	예	예	예
$ln	아니요	예	예	예
$log	아니요	예	예	예
$log10	아니요	예	예	예
$mod	예	예	예	예
$multiply	예	예	예	예
$pow	아니요	아니요	아니요	아니요
$round	-	-	아니요	아니요
$sqrt	아니요	예	예	예
$subtract	예	예	예	예
$trunc	아니요	아니요	아니요	아니요

배열 연산자

Command	3.6	4.0	5.0	엘라스틱 클러스터
$arrayElemAt	예	예	예	예
$arrayToObject	예	예	예	예
$concatArrays	예	예	예	예
$filter	예	예	예	예
$first	-	-	아니요	아니요

Command	3.6	4.0	5.0	엘라스틱 클러스터
$in	예	예	예	예
$indexOfArray	예	예	예	예
$isArray	예	예	예	예
$last	-	-	아니요	아니요
$objectToArray	예	예	예	예
$range	예	예	예	예
$reverseArray	예	예	예	예
$reduce	예	예	예	예
$size	예	예	예	예
$slice	예	예	예	예
$zip	예	예	예	예

부울 연산

Command	3.6	4.0	5.0	엘라스틱 클러스터
$and	예	예	예	예
$not	예	예	예	예
$or	예	예	예	예

비교 연산자

Command	3.6	4.0	5.0	엘라스틱 클러스터
$cmp	예	예	예	예
$eq	예	예	예	예
$gt	예	예	예	예
$gte	예	예	예	예
$lt	예	예	예	예
$lte	예	예	예	예
$ne	예	예	예	예

조건식 연산자

Command	3.6	4.0	5.0	엘라스틱 클러스터
$cond	예	예	예	예
$ifNull	예	예	예	예
$switch	아니요	예	예	아니요

데이터 유형 연산자

Command	3.6	4.0	5.0	엘라스틱 클러스터
$type	예	예	예	예

데이터 크기 연산자

Command	3.6	4.0	5.0	엘라스틱 클러스터
$binarySize	-	-	아니요	아니요
$bsonSize	-	-	아니요	아니요

날짜 연산자

Command	3.6	4.0	5.0	엘라스틱 클러스터
$dateAdd	아니요	아니요	예	예
$dateDiff	-	-	아니요	아니요
$dateFromParts	아니요	아니요	아니요	아니요
$dateFromString	예	예	예	예
$dateSubtract	아니요	아니요	예	예
$dateToParts	아니요	아니요	아니요	아니요
$dateToString	예	예	예	예
$dateTrunc	-	-	아니요	아니요
$dayOfMonth	예	예	예	예
$dayOfWeek	예	예	예	예
$dayOfYear	예	예	예	예
$hour	예	예	예	예
$isoDayOfWeek	예	예	예	예

Command	3.6	4.0	5.0	엘라스틱 클러스터
$isoWeek	예	예	예	예
$isoWeekYear	예	예	예	예
$millisecond	예	예	예	예
$minute	예	예	예	예
$month	예	예	예	예
$second	예	예	예	예
$week	예	예	예	예
$year	예	예	예	예

리터럴 연산자

Command	3.6	4.0	5.0	엘라스틱 클러스터
$literal	예	예	예	예

병합 연산자

Command	3.6	4.0	5.0	엘라스틱 클러스터
$mergeObjects	예	예	예	예

자연 연산자

Command	3.6	4.0	5.0	엘라스틱 클러스터
$natural	예	예	예	예

집합 연산자

Command	3.6	4.0	5.0	엘라스틱 클러스터
$allElementsTrue	아니요	예	예	예
$anyElementTrue	아니요	예	예	예
$setDifference	아니요	예	예	예
$setEquals	예	예	예	예
$setIntersection	예	예	예	예
$setIsSubset	예	예	예	예
$setUnion	예	예	예	예
$setWindowFields	아니요	아니요	아니요	아니요

단계 연산자

Command	3.6	4.0	5.0	엘라스틱 클러스터
$addFields	예	예	예	예
$bucket	아니요	아니요	아니요	아니요
$bucketAuto	아니요	아니요	아니요	아니요
$collStats	아니요	아니요	아니요	아니요
$count	예	예	예	예
$currentOp	예	예	예	예
$facet	아니요	아니요	아니요	아니요
$geoNear	예	예	예	예
$graphLookup	아니요	아니요	아니요	아니요
$group	예	예	예	예
$indexStats	예	예	예	예
$limit	예	예	예	예
$listLocalSessions	아니요	아니요	아니요	아니요
$listSessions	아니요	아니요	아니요	아니요
$lookup	예	예	예	예
$match	예	예	예	예
$merge	-	-	아니요	아니요
$out	예	예	예	아니요

Command	3.6	4.0	5.0	엘라스틱 클러스터
$planCacheStats	-	-	아니요	아니요
$project	예	예	예	예
$redact	예	예	예	예
$replaceRoot	예	예	예	예
$sample	예	예	예	예
$set	-	-	아니요	아니요
$setWindowFields	-	-	아니요	아니요
$skip	예	예	예	예
$sort	예	예	예	예
$sortByCount	아니요	아니요	아니요	아니요
$unionWith	-	-	아니요	아니요
$unset	-	-	아니요	아니요
$unwind	예	예	예	예

문자열 연산자

Command	3.6	4.0	5.0	엘라스틱 클러스터
$concat	예	예	예	예
$indexOfBytes	예	예	예	예
$indexOfCP	예	예	예	예

Command	3.6	4.0	5.0	엘라스틱 클러스터
$ltrim	아니요	아니요	아니요	아니요
$regexFind	-	-	예	아니요
$regexFindAll	-	-	아니요	아니요
$regexMatch	-	-	예	아니요
$replaceAll	-	-	아니요	아니요
$replaceOne	-	-	아니요	아니요
$rtrim	아니요	아니요	아니요	아니요
$split	예	예	예	예
$strcasecmp	예	예	예	예
$strLenBytes	예	예	예	예
$strLenCP	예	예	예	예
$substr	예	예	예	예
$substrBytes	예	예	예	예
$substrCP	예	예	예	예
$toLower	예	예	예	예
$toUpper	예	예	예	예
$trim	아니요	아니요	아니요	아니요

시스템 변수

Command	3.6	4.0	5.0	엘라스틱 클러스터
$$CURRENT	아니요	아니요	아니요	아니요
$$DESCEND	예	예	예	예
$$KEEP	예	예	예	예
$$PRUNE	예	예	예	예
$$REMOVE	아니요	아니요	아니요	아니요
$$ROOT	예	예	예	예

텍스트 검색 연산자

Command	3.6	4.0	5.0	엘라스틱 클러스터
$meta	아니요	아니요	예	아니요
$search	아니요	아니요	예	아니요

유형 변환 연산자

Command	3.6	4.0	5.0	엘라스틱 클러스터
$convert	아니요	예	예	예
$isNumber	-	-	아니요	아니요
$toBool	아니요	예	예	예

Command	3.6	4.0	5.0	엘라스틱 클러스터
$toDate	아니요	예	예	예
$toDecimal	아니요	예	예	예
$toDouble	아니요	예	예	예
$toInt	아니요	예	예	예
$toLong	아니요	예	예	예
$toObjectId	아니요	예	예	예
$toString	아니요	예	예	예

변수 연산자

Command	3.6	4.0	5.0	엘라스틱 클러스터
$let	예	예	예	예
$map	예	예	예	예

기타 연산자

Command	3.6	4.0	5.0	엘라스틱 클러스터
$getField	-	-	아니요	아니요
$rand	-	-	아니요	아니요
$sampleRate	-	-	아니요	아니요

데이터 타입

Command	3.6	4.0	5.0	엘라스틱 클러스터
32비트 정수(int)	예	예	예	예
64비트 정수(int)	예	예	예	예
Array	예	예	예	예
이진 데이터	예	예	예	예
불	예	예	예	예
날짜	예	예	예	예
DBPointer	아니요	아니요	아니요	아니요
DBRefs	아니요	아니요	아니요	아니요
Decimal128	예	예	예	예
배정밀도 실수	예	예	예	예
JavaScript	아니요	아니요	아니요	아니요
JavaScript(범위 포함)	아니요	아니요	아니요	아니요
MaxKey	예	예	예	예
MinKey	예	예	예	예
Null	예	예	예	예
객체	예	예	예	예
ObjectId	예	예	예	예
정규식	예	예	예	예

Command	3.6	4.0	5.0	엘라스틱 클러스터
String	예	예	예	예
Symbol	아니요	아니요	아니요	아니요
Timestamp	예	예	예	예
정의되지 않음	아니요	아니요	아니요	아니요

인덱스 및 인덱스 속성

주제

- 인덱스
- 인덱스 속성

인덱스

Command	3.6	4.0	5.0	엘라스틱 클러스터
2dsphere	예	예	예	예
2d 인덱스	아니요	아니요	아니요	아니요
복합 인덱스	예	예	예	예
해시된 인덱스	아니요	아니요	아니요	아니요
Multikey 인덱스	예	예	예	예
단일 필드 인덱스	예	예	예	예
텍스트 인덱스	아니요	아니요	예	아니요
와일드카드	아니요	아니요	아니요	아니요

인덱스 속성

Command	3.6	4.0	5.0	엘라스틱 클러스터
배경	예	예	예	예
대소문자 구분 안 함	아니요	아니요	아니요	아니요
숨김	아니요	아니요	아니요	아니요
부분	아니요	아니요	예	아니요
희소	예	예	예	예
텍스트	아니요	아니요	예	아니요
TTL	예	예	예	예
고유	예	예	예	예
벡터	아니요	아니요	예	아니요

Amazon DocumentDB 생성형 인공 지능

Amazon DocumentDB는 기계 학습(ML) 및 생성형 인공 지능(AI) 모델이 Amazon DocumentDB에 저장된 데이터를 실시간으로 사용할 수 있는 기능을 제공합니다. 고객은 더 이상 별도의 인프라를 관리하고, 다른 서비스와 연결하기 위한 코드를 작성하고, 기본 데이터베이스에서 데이터를 복제하는 데 시간을 소비할 필요가 없습니다.

인공 지능과가 AI 요구 사항을 AWS 지원하는 방법에 대한 자세한 내용은이 "What-is" 문서를 참조하세요.

주제

- Amazon SageMaker AI Canvas를 사용한 노코드 기계 학습
- Amazon DocumentDB에 대한 벡터 검색

Amazon SageMaker AI Canvas를 사용한 노코드 기계 학습

Amazon SageMaker AI Canvas를 사용하면 한 줄의 코드를 작성하지 않고도 자체 AI/ML 모델을 구축할 수 있습니다. 회귀 및 예측과 같은 일반적인 사용 사례를 위한 ML 모델을 구축하고 Amazon Bedrock에서 파운데이션 모델(FM) 액세스하고 평가할 수 있습니다. 콘텐츠 생성, 텍스트 추출 및 텍스트 요약을 위해 Amazon SageMaker AI JumpStart에서 퍼블릭 FMs에 액세스하여 생성형 AI 솔루션을 지원할 수도 있습니다.

SageMaker AI Canvas를 사용하여 코드 없는 ML 모델을 빌드하는 방법

Amazon DocumentDB는 이제 Amazon SageMaker AI Canvas와 통합되어 Amazon DocumentDB에 저장된 데이터와 함께 노코드 기계 학습(ML)을 활성화합니다. 이제 회귀 및 예측 요구 사항을 위한 ML 모델을 구축하고 단일 코드 줄을 작성하지 않고도 Amazon DocumentDB에 저장된 데이터를 사용하여 콘텐츠 요약 및 생성을 위한 파운데이션 모델을 사용할 수 있습니다.

SageMaker AI Canvas는 Amazon DocumentDB 고객이 AI/ML 전문 지식 없이 예측을 생성하거나 한 줄의 코드를 작성할 수 있는 시각적 인터페이스를 제공합니다. 이제 고객은에서 SageMaker AI Canvas 워크스페이스를 시작하고 데이터 준비 및 모델 훈련을 위해 Amazon DocumentDB 데이터를 AWS Management Console가져오고 조인할 수 있습니다. 이제 SageMaker AI Canvas에서 Amazon DocumentDB의 데이터를 사용하여 모델을 구축 및 보강하여 고객 이탈을 예측하고, 사기를 감지하고, 유지 관리 실패를 예측하고, 비즈니스 지표를 예측하고, 콘텐츠를 생성할 수 있습니다. 이제 고객은 SageMaker AI Canvas와 QuickSight의 기본 통합을 사용하여 팀 간에 ML 기반 인사이트를 게시하고 공유할 수 있습니다. SageMaker AI Canvas의 데이터 수집 파이프라인은 기본적으로 Amazon

DocumentDB 보조 인스턴스에서 실행되므로 애플리케이션 및 SageMaker AI Canvas 수집 워크로드의 성능이 저하되지 않습니다.

Amazon DocumentDB 고객은 새 Amazon DocumentDB 코드 없는 ML 콘솔 페이지로 이동하여 새 SageMaker AI Canvas 워크스페이스 또는 사용 가능한 SageMaker AI Canvas 워크스페이스에 연결하여 SageMaker AI Canvas를 시작할 수 있습니다.

SageMaker AI 도메인 및 사용자 프로필 구성

VPC 전용 모드에서 실행 중인 SageMaker AI 도메인에서 Amazon DocumentDB 클러스터에 연결할 수 있습니다. VPC에서 SageMaker AI 도메인을 시작하면 SageMaker AI Studio 및 Canvas 환경에서 데이터 흐름을 제어할 수 있습니다. 이를 통해 인터넷 액세스를 제한하고, 표준 AWS 네트워킹 및 보안 기능을 사용하여 트래픽을 모니터링 및 검사하고, VPC 엔드포인트를 통해 다른 AWS 리소스에 연결할 수 있습니다. Amazon SageMaker AI Canvas 시작하기 및 Amazon SageMaker AI 개발자 안내서에 있는 인터넷 액세스 없이 VPC에서 Amazon SageMaker AI Canvas 구성을 참조하여 Amazon DocumentDB 클러스터에 연결할 SageMaker AI 도메인을 생성하세요. Amazon SageMaker

Amazon DocumentDB 및 SageMaker AI Canvas에 대한 IAM 액세스 권한 구성

연결된 역할 및 자격 증명에 `AmazonDocDBConsoleFullAccess`가 연결된 Amazon DocumentDB 사용자는 AWS Management Console에 액세스할 수 있습니다. 앞서 언급한 역할 또는 자격 증명에 다음 작업을 추가하여 Amazon SageMaker AI Canvas를 사용한 노코드 기계 학습에 대한 액세스를 제공합니다.

```
"sagemaker:CreatePresignedDomainUrl",
"sagemaker:DescribeDomain",
"sagemaker:ListDomains",
"sagemaker:ListUserProfiles"
```

SageMaker AI Canvas에 대한 데이터베이스 사용자 및 역할 생성

Amazon DocumentDB의 역할 기반 액세스 제어(RBAC)를 사용하여 데이터베이스에서 사용자가 수행할 수 있는 작업에 대한 액세스를 제한할 수 있습니다. RBAC는 사용자에게 하나 이상의 역할을 부여하여 작동합니다. 이러한 역할은 사용자가 데이터베이스 리소스에서 수행할 수 있는 작업을 결정합니다.

Canvas 사용자는 사용자 이름과 암호 자격 증명을 사용하여 Amazon DocumentDB 데이터베이스에 연결합니다. Amazon DocumentDB RBAC 기능을 사용하여 특정 데이터베이스에 대한 읽기 액세스 권한이 있는 Canvas 사용자의 데이터베이스 사용자/역할을 생성할 수 있습니다.

예를 들어 `createUser`를 사용합니다.

```
db.createUser({
user: "canvas_user",
pwd: "<insert-password>",
roles: [{role: "read", db: "sample-database-1"}]
})
```

이렇게 하면 `sample-database-1` 데이터베이스에 대한 읽기 권한이 있는 `canvas_user`가 생성됩니다. Canvas 분석가는 이 자격 증명을 사용하여 Amazon DocumentDB 클러스터의 데이터에 액세스할 수 있습니다. 자세한 내용은 역할 기반 액세스 제어를 사용한 데이터베이스 액세스 섹션을 참조하세요.

사용 가능한 리전

노코드 통합은 Amazon DocumentDB와 Amazon SageMaker AI Canvas가 모두 지원되는 리전에서 사용할 수 있습니다. 리전은 다음과 같습니다.

- us-east-1(버지니아 북부)
- us-east-2(오하이오)
- us-west-2(오리건)
- ap-northeast-1(도쿄)
- ap-northeast-2(서울)
- ap-south-1(뭄바이)
- ap-southeast-1(싱가포르)
- ap-southeast-2(시드니)
- eu-central-1(프랑크푸르트)
- eu-west-1(아일랜드)

최신 리전 가용성은 Amazon SageMaker AI 개발자 안내서의 Amazon SageMaker AI Canvas를 참조하세요. Amazon SageMaker

Amazon DocumentDB에 대한 벡터 검색

벡터 검색은 기계 학습에서 거리 또는 유사성 지표를 사용하여 벡터 표현을 비교하여 지정된 데이터 포인트와 유사한 데이터 포인트를 찾는 데 사용되는 방법입니다. 벡터 공간에 두 벡터가 가까울수록 기본

항목이 더 유사한 것으로 간주됩니다. 이 기법은 데이터의 의미 체계적 의미를 캡처하는 데 도움이 됩니다. 이 접근 방식은 추천 시스템, 자연어 처리 및 이미지 인식과 같은 다양한 애플리케이션에 유용합니다.

Amazon DocumentDB에 대한 벡터 검색은 JSON 기반 문서 데이터베이스의 유연성과 풍부한 쿼리 기능을 벡터 검색 기능과 결합합니다. 기존 Amazon DocumentDB 데이터 또는 유연한 문서 데이터 구조를 사용하여 의미 검색 경험, 제품 권장 사항, 개인화, 챗봇, 사기 탐지 및 이상 탐지와 같은 기계 학습 및 생성형 AI 사용 사례를 구축하려는 경우 Amazon DocumentDB에 대한 벡터 검색이 이상적인 선택입니다. 벡터 검색은 Amazon DocumentDB 5.0 인스턴스 기반 클러스터에서 사용할 수 있습니다.

주제

- 벡터 삽입
- 벡터 인덱스 생성
- 인덱스 정의 가져오기
- 벡터 쿼리
- 특성 및 제한 사항
- 모범 사례

벡터 삽입

Amazon DocumentDB 데이터베이스에 벡터를 삽입하려면 기존 삽입 방법을 사용할 수 있습니다.

예제

다음 예제에서는 테스트 데이터베이스 내에 5개의 문서 모음이 생성됩니다. 각 문서에는 제품 이름과 해당 벡터 임베딩의 두 필드가 포함되어 있습니다.

```
db.collection.insertMany([
  {"product_name": "Product A", "vectorEmbedding": [0.2, 0.5, 0.8]},
  {"product_name": "Product B", "vectorEmbedding": [0.7, 0.3, 0.9]},
  {"product_name": "Product C", "vectorEmbedding": [0.1, 0.2, 0.5]},
  {"product_name": "Product D", "vectorEmbedding": [0.9, 0.6, 0.4]},
  {"product_name": "Product E", "vectorEmbedding": [0.4, 0.7, 0.2]}
]);
```

벡터 인덱스 생성

Amazon DocumentDB는 계층적 탐색 가능 스몰 월드(HNSW) 인덱싱과 플랫 압축을 사용한 역 파일(IVFFlat) 인덱싱 방법을 모두 지원합니다. IVFFlat 인덱스는 벡터를 목록으로 분리한 다음 쿼리 벡터에 가장 가까운 해당 목록의 선택된 하위 집합을 검색합니다. 반면 HNSW 인덱스는 벡터 데이터를 다층 그래프로 구성합니다. HNSW는 IVFFlat 에 비해 구축 시간이 느리지만 쿼리 성능 및 리콜을 개선합니다. IVFFlat과 달리 HNSW에는 훈련 단계가 포함되어 있지 않으므로 초기 데이터 로드 없이 인덱스를 생성할 수 있습니다. 대부분의 사용 사례에서는 벡터 검색에 HNSW 인덱스 유형을 사용하는 것이 좋습니다.

벡터 인덱스를 생성하지 않으면 Amazon DocumentDB가 정확히 가장 가까운 이웃 검색을 수행하여 완벽한 호출을 보장합니다. 그러나 프로덕션 시나리오에서는 속도가 중요합니다. 속도 향상을 위해 일부 리콜을 처리할 수 있는 벡터 인덱스를 사용하는 것이 좋습니다. 벡터 인덱스를 추가하면 쿼리 결과가 달라질 수 있다는 점에 유의해야 합니다.

템플릿

다음 `createIndex` 또는 `runCommand` 템플릿을 사용하여 벡터 필드에 벡터 인덱스를 빌드할 수 있습니다.

Using createIndex

mongosh 및 Java와 같은 특정 드라이버에서 `createIndex`의`vectorOptions` 파라미터를 사용하면 오류가 발생할 수 있습니다. 이러한 경우 `runCommand`를 사용하는 것이 좋습니다.

```
db.collection.createIndex(
  { "<vectorField>": "vector" },
  { "name": "<indexName>",
    "vectorOptions": {
      "type": " <hnsw> | <ivfflat> ",
      "dimensions": <number_of_dimensions>,
      "similarity": " <euclidean> | <cosine> | <dotProduct> ",
      "lists": <number_of_lists> [applicable for IVFFlat],
      "m": <max number of connections> [applicable for HNSW],
      "efConstruction": <size of the dynamic list for index build> [applicable for
 HNSW]
    }
  }
);
```

Using runCommand

mongosh 및 Java와 같은 특정 드라이버에서 `createIndex`의`vectorOptions` 파라미터를 사용하면 오류가 발생할 수 있습니다. 이러한 경우 `runCommand`를 사용하는 것이 좋습니다.

```
db.runCommand(
  { "createIndexes": "<collection>",
  "indexes": [{
      key: { "<vectorField>": "vector" },
      vectorOptions: {
          type: " <hnsw> | <ivfflat> ",
          dimensions: <number of dimensions>,
          similarity: " <euclidean> | <cosine> | <dotProduct> ",
          lists: <number_of_lists> [applicable for IVFFlat],
          m: <max number of connections> [applicable for HNSW],
          efConstruction: <size of the dynamic list for index build> [applicable for
 HNSW]
          },
      name: "myIndex"
      }]
  }
);
```

파라미터	요구 사항	데이터 유형	설명	값
name	선택 사항	문자열	인덱스의 이름을 지정합니다.	영숫자
type	선택 사항		인덱스의 유형을 지정합니다.	지원: hnsw 또는 ivfflat 기본값: HNSW(엔진 패치 3.0.4574 이상)
dimensions	필수	정수	벡터 데이터의 차원 수를 지정합니다.	최대 2,000개의 차원.

파라미터	요구 사항	데이터 유형	설명	값
similarity	필수	문자열	유사성 계산에 사용되는 거리 지표를 지정합니다.	• **euclidean** • **cosine** • **dotProduct**
lists	IVFFlat에 필요	정수	IVFFlat 인덱스가 벡터 데이터를 그룹화하는 데 사용하는 클러스터 수를 지정합니다. 권장 설정은 최대 100만 개 문서의 경우 문서 수/1000, 그리고 100만 개가 넘는 문서의 경우 `sqrt(# of documents)` 입니다.	최소: 1 최대: 특성 및 제한 사항 아래의 인스턴스 유형별 목록 표를 참조하세요.
m	선택 사항	정수	HNSW 인덱스의 최대 연결 수를 지정합니다.	기본값: 16 범위 [2, 100]
efConstruction	선택 사항	정수	HNSW 인덱스에 대한 그래프를 구성할 동적 후보 목록의 크기를 지정합니다. `efConstruction` 은 (2 * m)보다 크거나 같아야 합니다.	기본값: 64 범위 [4, 1000]

검색의 정확도/리콜, 빌드 시간 및 성능에 영향을 미치므로 IVFFlat `lists`, `m` 및 HNSW `efConstruction`과 같은 하위 파라미터의 값을 적절하게 설정하는 것이 중요합니다. 목록 값이 높을수록 각 목록의 벡터 수가 줄어들어 리전이 작아지므로 쿼리 속도가 증가합니다. 그러나 리전 크기가 작을수록 더 많은 리콜 오류가 발생하여 정확도가 낮아질 수 있습니다. HNSW의 경우 `m` 및 `efConstruction` 값을 늘리면 정확도가 높아지지만 인덱스 구축 시간과 크기도 증가합니다. 다음 예를 참조하세요.

예시

HNSW

```
db.collection.createIndex(
  { "vectorEmbedding": "vector" },
  { "name": "myIndex",
    "vectorOptions": {
      "type": "hnsw",
      "dimensions": 3,
      "similarity": "euclidean",
      "m": 16,
      "efConstruction": 64
    }
  }
);
```

IVFFlat

```
db.collection.createIndex(
  { "vectorEmbedding": "vector" },
  { "name": "myIndex",
    "vectorOptions": {
      "type": "ivfflat",
      "dimensions": 3,
      "similarity": "euclidean",
      "lists":1
    }
  }
)
```

인덱스 정의 가져오기

다음 `getIndexes` 명령을 사용하여 벡터 인덱스를 포함한 인덱스의 세부 정보를 볼 수 있습니다.

예제

```
db.collection.getIndexes()
```

출력 예

```
[
 {
  "v" : 4,
  "key" : {
   "_id" : 1
  },
  "name" : "_id_",
  "ns" : "test.collection"
 },
 {
  "v" : 4,
  "key" : {
   "vectorEmbedding" : "vector"
  },
  "name" : "myIndex",
  "vectorOptions" : {
   "type" : "ivfflat",
   "dimensions" : 3,
   "similarity" : "euclidean",
   "lists" : 1
  },
  "ns" : "test.collection"
 }
]
```

벡터 쿼리

벡터 쿼리 템플릿

다음 템플릿을 사용하여 벡터를 쿼리합니다.

```
db.collection.aggregate([
  {
    $search: {
      "vectorSearch": {
        "vector": <query vector>,
```

```
        "path": "<vectorField>",
        "similarity": "<distance metric>",
        "k": <number of results>,
        "probes":<number of probes> [applicable for IVFFlat],
        "efSearch":<size of the dynamic list during search> [applicable for HNSW]
      }
    }
  }
]);
```

파라미터	요구 사항	유형	설명	값
vectorSearch	필수	연산자	$search 명령 내에서 벡터를 쿼리하는 데 사용됩니다.	
vector	필수	array	유사한 벡터를 찾는 데 사용할 쿼리 벡터를 나타냅니다.	
path	필수	문자열	벡터 필드의 이름을 정의합니다.	
k	필수	정수	검색에서 반환하는 결과의 수를 지정합니다.	
similarity	필수	문자열	유사성 계산에 사용되는 거리 지표를 지정합니다.	• **euclidean** • **cosine** • **dotProduct**
probes	선택 사항	정수	벡터 검색에서 검사할 클러스터 수입니다. 값이 높을수록 속도 비용으로 더 나은 리콜을 제공합니다.	기본값: 1

파라미터	요구 사항	유형	설명	값
			정확히 가장 가까운 이웃 검색을 위한 목록 수로 설정할 수 있습니다(실행 계획기가 인덱스를 사용하지 않는 시점). 미세 조정을 시작하기 위한 권장 설정은 `sqrt(# of lists)`입니다.	
`efSearch`	선택 사항	정수	검색 중에 HNSW 인덱스가 사용하는 동적 후보 목록의 크기를 지정합니다. `efSearch` 값이 높을수록 속도 비용으로 더 나은 리콜을 제공합니다.	기본값: 40 범위 [1, 1000]

원하는 성능과 정확도를 달성하려면 `efSearch` (HNSW) 또는 `probes` (IVFFlat)의 값을 미세 조정하는 것이 중요합니다. 다음 예제 작업을 참조하세요.

HNSW

```
db.collection.aggregate([
  {
    $search: {
      "vectorSearch": {
        "vector": [0.2, 0.5, 0.8],
        "path": "vectorEmbedding",
        "similarity": "euclidean",
        "k": 2,
```

```
        "efSearch": 40
      }
    }
  }
]);
```

IVFFlat

```
db.collection.aggregate([
  {
    $search: {
      "vectorSearch": {
        "vector": [0.2, 0.5, 0.8],
        "path": "vectorEmbedding",
        "similarity": "euclidean",
        "k": 2,
        "probes": 1
      }
    }
  }
]);
```

출력 예

이 작업의 출력은 다음과 같이 표시됩니다.

```
{ "_id" : ObjectId("653d835ff96bee02cad7323c"), "product_name" : "Product A",
 "vectorEmbedding" : [ 0.2, 0.5, 0.8 ] }
{ "_id" : ObjectId("653d835ff96bee02cad7323e"), "product_name" : "Product C",
 "vectorEmbedding" : [ 0.1, 0.2, 0.5 ] }
```

특성 및 제한 사항

버전 호환성

- Amazon DocumentDB에 대한 벡터 검색은 Amazon DocumentDB 5.0 인스턴스 기반 클러스터에서만 사용할 수 있습니다.

벡터

- Amazon DocumentDB는 최대 2,000개의 차원으로 벡터를 인덱싱할 수 있습니다. 하지만 인덱스 없이 최대 16,000개의 차원을 저장할 수 있습니다.

인덱스

- IVFFlat 인덱스 생성의 경우 목록 파라미터의 권장 설정은 최대 100만 개의 문서의 경우 문서 수/1000 및 100만 개가 넘는 문서의 경우 `sqrt(# of documents)`입니다. 작동 메모리 제한으로 인해 Amazon DocumentDB는 차원 수에 따라 목록 파라미터의 특정 최대값을 지원합니다. 참고로 다음 표에서는 500, 1,000 및 2,000 차원의 벡터에 대한 목록 파라미터의 최대값을 제공합니다.

인스턴스 유형	500개 차원이 있는 목록	1,000개 차원이 있는 목록	2,000개 차원이 있는 목록
t3.med	372	257	150
r5.l	915	741	511
r5.xl	1,393	1,196	901
r5.2xl	5,460	5,230	4,788
r5.4xl	7,842	7,599	7,138
r5.8xl	11,220	10,974	10,498
r5.12xl	13,774	13,526	13,044
r5.16xl	15,943	15,694	15,208
r5.24xl	19,585	19,335	18,845

- `compound`, `sparse` 또는 `partial`과 같은 다른 인덱스 옵션은 벡터 인덱스에서 지원되지 않습니다.
- 병렬 인덱스 빌드는 HNSW 인덱스에 지원되지 않습니다. IVFFlat 인덱스에만 지원됩니다.

벡터 쿼리

- 벡터 검색 쿼리의 경우 최적의 결과를 위해 probes 또는 efSearch와 같은 파라미터를 미세 조정하는 것이 중요합니다. probes 또는 efSearch 파라미터의 값이 높을수록 재호출이 높고 속도가 낮아집니다. 프로브 파라미터 미세 조정을 시작하기 위한 권장 설정은 sqrt(# of lists)입니다.

모범 사례

Amazon DocumentDB의 벡터 검색을 사용한 작업 모범 사례에 대해서 알아봅니다. 이 섹션은 새로운 모범 사례가 확인되는 대로 지속적으로 업데이트됩니다.

- 플랫 압축(IVFFlat) 인덱스 생성이 포함된 반전된 파일에는 유사성을 기반으로 데이터 포인트를 클러스터링하고 구성하는 작업이 포함됩니다. 따라서 인덱스가 더 효과적이려면 인덱스를 생성하기 전에 최소한 일부 데이터를 로드하는 것이 좋습니다.
- 벡터 검색 쿼리의 경우 최적의 결과를 위해 probes 또는 efSearch와 같은 파라미터를 미세 조정하는 것이 중요합니다. probes 또는 efSearch 파라미터의 값이 높을수록 재호출이 높고 속도가 낮아집니다. probes 파라미터 미세 조정을 시작하기 위한 권장 설정은 sqrt(lists)입니다.

리소스

- 새 블로그 게시물 내용 벡터 검색
- 의미 검색 코드 샘플
- Amazon DocumentDB 벡터 검색 코드 샘플

Amazon DocumentDB로 마이그레이션

Amazon DocumentDB(MongoDB 호환)는 MongoDB API와 호환되는 완전 관리형 데이터베이스 서비스입니다. 이 섹션에서 설명하는 프로세스를 사용하여 온프레미스 또는 Amazon Elastic Compute Cloud(Amazon EC2)에서 실행 중인 MongoDB 데이터베이스에서 Amazon DocumentDB로 데이터를 마이그레이션할 수 있습니다.

주제

- 마이그레이션 도구
- Discovery
- 계획: Amazon DocumentDB 클러스터 요구 사항
- 마이그레이션 접근 방식
- 마이그레이션 원본
- 마이그레이션 연결
- 테스트
- 성능 테스트
- 장애 조치 테스트
- 추가 리소스
- 마이그레이션 플레이북: MongoDB에서 Amazon DocumentDB로

마이그레이션 도구

Amazon DocumentDB로 마이그레이션하기 위해 대부분의 고객이 사용하는 두 가지 기본 도구는 AWS Database Migration Service (AWS DMS)와 `mongodump` 및 `mongorestore` 같은 명령줄 유틸리티입니다. 그리고 이러한 옵션 중 하나는 전체 시간을 줄이고 마이그레이션 속도를 높일 수 있으므로 마이그레이션을 시작하기 전에 먼저 Amazon DocumentDB에 인덱스를 작성하는 것이 좋습니다. 이렇게 하려면 Amazon DocumentDB 인덱스 도구를 사용하면 됩니다.

AWS Database Migration Service

AWS Database Migration Service (AWS DMS)는 관계형 데이터베이스와 비관계형 데이터베이스를 Amazon DocumentDB로 쉽게 마이그레이션할 수 있는 클라우드 서비스입니다. AWS DMS 를 사용하여 온프레미스 또는 EC2에서 호스팅되는 데이터베이스에서 Amazon DocumentDB로 데이터를 마이

그레이션할 수 있습니다. AWS DMS를 사용하면 일회성 마이그레이션을 수행하거나 지속적인 변경 사항을 복제하여 소스와 대상을 동기화할 수 있습니다.

를 사용하여 Amazon DocumentDB AWS DMS 로 마이그레이션하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- MongoDB를의 소스로 사용 AWS DMS
- Amazon DocumentDB를의 대상으로 사용 AWS Database Migration Service
- 안내서: MongoDB에서 Amazon DocumentDB로 마이그레이션

명령줄 유틸리티

데이터를 Amazon DocumentDB로(에서) 마이그레이션하기 위한 공통 유틸리티에는 `mongodump`, `mongorestore`, `mongoexport` 및 `mongoimport`가 있습니다. 일반적으로 `mongodump` 및 `mongorestore`는 데이터베이스에서 이진 형식으로 데이터를 덤프하고 복원할 때 가장 효율적인 유틸리티입니다. 이것은 일반적으로 최선의 옵션이며 논리적 내보내기에 비해 데이터 크기가 작아집니다. `mongoexport` 및 `mongoimport`는 JSON 또는 CSV 같은 논리적 형식으로 데이터를 내보내고 가져오려는 경우 유용합니다. 이 데이터는 사람이 읽을 수 있지만 일반적으로 `mongodump`/`mongorestore`보다 느리고 데이터 크기가 커지기 때문입니다.

아래 마이그레이션 접근 방식 섹션에서는 사용 사례 AWS DMS 및 요구 사항에 따라 및 명령줄 유틸리티를 사용하는 것이 가장 좋은 시기에 대해 설명합니다.

Discovery

각 MongoDB 배포에 대해 다음 두 가지 데이터 세트를 식별하고 기록해야 합니다. 아키텍처 세부 정보 및 운영 특성. 이 정보는 적절한 마이그레이션 접근 방식 및 클러스터 크기를 선택하는 데 도움이 됩니다.

아키텍처 세부 정보

- 이름

  이 배포를 추적하기 위한 고유한 이름을 선택합니다.

- 버전

배포가 실행하는 MongoDB의 버전을 기록합니다. 버전을 확인하려면 mongo 쉘을 사용하여 복제본 집합 멤버에 연결하고 `db.version()` 작업을 실행합니다.

- 유형

  배포가 독립 실행형 mongo 인스턴스, 복제본 집합 또는 샤드 있는 클러스터인지 기록합니다.

- 회원

  각 클러스터, 복제본 집합 또는 독립 실행형 멤버의 호스트 이름, 주소 및 포트를 기록합니다.

  클러스터링된 배포의 경우 mongo 쉘을 사용하여 mongo 호스트에 연결하고 `sh.status()` 작업을 실행하면 샤드 멤버를 찾을 수 있습니다.

  복제본 집합의 경우 mongo 쉘을 사용하여 복제본 집합 멤버에 연결하고 `rs.status()` 작업을 실행하면 멤버를 가져올 수 있습니다.

- Oplog 크기

  복제본 집합 또는 샤드 있는 클러스터의 경우 각 복제본 집합 멤버의 oplog 크기를 기록합니다. 멤버의 oplog 크기를 확인하려면 mongo 쉘을 사용하여 복제본 집합 멤버에 연결하고 `ps.printReplicationInfo()` 작업을 실행합니다.

- 복제본 집합 멤버 우선 순위

  복제본 집합 또는 샤드 있는 클러스터의 경우 각 복제본 집합 멤버의 우선 순위를 기록합니다. 복제본 집합 멤버 우선 순위를 확인하려면 mongo 쉘을 사용하여 복제본 집합 멤버에 연결하고 `rs.conf()` 작업을 실행합니다. 우선 순위가 `priority` 키 값으로 표시됩니다.

- TLS/SSL 사용

각 노드에서 전송 중 암호화로 TLS(전송 계층 보안)/SSL(보안 소켓 계층)이 사용되는지 여부를 기록합니다.

운영 특성

- 데이터베이스 특성

  각 모음에 대해 다음 정보를 기록합니다.

  - 명칭
  - 데이터 크기
  - 모음 개수

  데이터베이스 통계를 확인하려면 mongo 쉘을 사용하여 데이터베이스에 연결하고 `db.runCommand({dbstats: 1})` 명령을 실행합니다.

- 모음 통계

  각 모음에 대해 다음 정보를 기록합니다.

  - 네임스페이스
  - 데이터 크기
  - 인덱스 개수
  - 모음 제한이 있는지 여부

- 인덱스 통계

  각 모음에 대해 다음 인덱스 정보를 기록합니다.

  - 네임스페이스
  - ID
  - 크기
  - 키
  - TTL

- 희소
- 배경

인덱스 정보를 확인하려면 mongo 쉘을 사용하여 데이터베이스에 연결하고 `db.collection.getIndexes()` 명령을 실행합니다.

- Opcounters

이 정보는 현재 MongoDB 워크로드 패턴(읽기 중심, 쓰기 중심 또는 균형적)을 이해하는 데 도움이 됩니다. 또한 초기 Amazon DocumentDB 인스턴스 선택에 대한 지침을 제공합니다.

다음은 모니터링 기간에 수집할 주요 정보입니다(단위: 개수/초).

- 쿼리
- 삽입
- 업데이트
- 삭제

이 정보는 `db.serverStatus()` 명령의 출력을 시간 그래프로 작성하여 얻을 수 있습니다. 또한 mongostat 도구를 사용하여 이들 통계의 순간 값을 가져올 수 있습니다. 하지만 이 옵션을 사용할 경우 피크 로드가 아닌 사용 기간에 마이그레이션을 계획하는 위험이 따릅니다.

- 네트워크 통계

이 정보는 현재 MongoDB 워크로드 패턴(읽기 중심, 쓰기 중심 또는 균형적)을 이해하는 데 도움이 됩니다. 또한 초기 Amazon DocumentDB 인스턴스 선택에 대한 지침을 제공합니다.

다음은 모니터링 기간에 수집할 주요 정보입니다(단위: 개수/초).

- 연결

- 수신 네트워크 바이트
- 송신 네트워크 바이트

이 정보는 `db.serverStatus()` 명령의 출력을 시간 그래프로 작성하여 얻을 수 있습니다. 또한 mongostat 도구를 사용하여 이들 통계의 순간 값을 가져올 수 있습니다. 하지만 이 옵션을 사용할 경우 피크 로드가 아닌 사용 기간에 마이그레이션을 계획하는 위험이 따릅니다.

계획: Amazon DocumentDB 클러스터 요구 사항

마이그레이션이 성공하려면 Amazon DocumentDB 클러스터의 구성과 애플리케이션이 클러스터에 액세스하는 방법을 모두 신중하게 고려해야 합니다. 클러스터 요구 사항을 결정할 때 다음 각 차원을 고려하세요.

- 가용성

  Amazon DocumentDB는 복제본 인스턴스 배포를 통해 고가용성을 제공하며, 이러한 인스턴스는 장애 조치라고 하는 프로세스에서 기본 인스턴스로 승격될 수 있습니다. 복제본 인스턴스를 다른 가용 영역에 배포함으로써 한층 높은 수준의 가용성을 확보할 수 있습니다.

  다음 표에서는 특정 가용성 목표를 충족하기 위한 Amazon DocumentDB 배포 구성에 대한 지침을 제공합니다.

가용성 목표	인스턴스 합계	복제본	가용 영역
99%	1	0	1
99.9%	2	1	2
99.99%	3	2	3

전체 시스템 안정성이 데이터베이스뿐 아니라 모든 구성 요소를 고려해야 합니다. 전체 시스템 안정성 요구 사항을 충족하기 위한 모범 사례 및 권장 사항은 AWS Well-Architected 신뢰성 기반 백서를 참조하세요.

- 성능

  Amazon DocumentDB 인스턴스에서는 클러스터의 스토리지 볼륨에서 읽고 쓸 수 있습니다. 클러스터 인스턴스는 다양한 메모리 및 vCPU 용량으로 구성된 다수의 유형으로 제공되며, 이러한 용량은 클러스터 읽기 및 쓰기 성능에 영향을 미칩니다. 발견 단계에서 수집한 정보를 사용하여 워크로드 성능 요구 사항을 지원할 수 있는 인스턴스 유형을 선택합니다. 지원되는 인스턴스 유형의 목록은 인스턴스 클래스 관리 섹션을 참조하세요.

  Amazon DocumentDB 클러스터의 인스턴스 유형을 선택할 때 워크로드 성능 요구 사항의 다음 측면을 고려하세요.

  - vCPU - 더 많은 연결 수가 필요한 아키텍처에는 vCPU가 더 많은 인스턴스가 유리할 수 있습니다.

  - 메모리 - 가능할 경우, 작업 데이터 세트를 메모리에 유지하면 성능이 극대화됩니다. 시작 가이드라인은 인스턴스 메모리의 3분의 1을 Amazon DocumentDB 엔진에 예약하고, 2분의 3을 작업 데이터 세트에 남겨두는 것입니다.

  - 연결—최소 최적 연결 수는 Amazon DocumentDB 인스턴스 vCPU당 8개입니다. Amazon DocumentDB 인스턴스 연결 제한은 훨씬 더 크지만 vCPU당 8개를 초과하면 추가 연결의 성능 이점이 감소합니다.

  - 네트워크 - 클라이언트 또는 연결 수가 많은 워크로드는 삽입 및 검색된 데이터에 필요한 전체 네트워크 성능을 고려해야 합니다. 대량 작업은 네트워크 리소스를 보다 효율적으로 활용할 수 있습니다.

  - 삽입 성능 - 단일 문서 삽입은 일반적으로 Amazon DocumentDB에 데이터를 삽입하는 가장 느린 방법입니다. 대량 삽입 작업은 단일 삽입보다 속도가 현격하게 빠를 수 있습니다.

- 읽기 성능 - 작업 메모리에서 읽기는 스토리지 볼륨에서 읽기보다 항상 빠릅니다. 그러므로 작업 집합을 메모리에 유지하도록 인스턴스 메모리 크기를 최적화하는 것이 이상적입니다.

  기본 인스턴스로부터 읽기를 처리하는 이외에, Amazon DocumentDB 클러스터는 복제본 집합으로 자동 구성됩니다. 그러므로 MongoDB 드라이버에서 읽기 기본 설정을 지정하여 읽기 전용 쿼리를 읽기 전용 복제본으로 라우팅할 수 있습니다. 복제본을 추가하여 읽기 트래픽을 확장함으로써 기본 인스턴스의 전체 로드를 줄일 수 있습니다.

  한 클러스터에서 다른 인스턴스 유형의 Amazon DocumentDB 복제본을 배포할 수 있습니다. 더 큰 인스턴스 유형의 복제본이 임시 분석 트래픽을 처리하도록 설정하는 것이 하나의 사용 사례가 될 수 있습니다. 혼합된 인스턴스 유형의 집합을 배포하는 경우 각 인스턴스에 대한 장애 조치 우선 순위를 구성해야 합니다. 이를 통해 장애 조치 이벤트가 항상 쓰기 로드를 감당하기에 충분한 크기의 복제본을 승격할 수 있습니다.

- 복구

  Amazon DocumentDB 는 데이터가 기록될 때마다 지속적으로 데이터를 백업합니다. 이는 백업 보존 기간이라고 하는 1~35일의 구성 가능한 기간 내에 시점 복구(PITR) 기능을 제공합니다. 기본 백업 보존 기간은 1일입니다. 또한 Amazon DocumentDB는 스토리지 볼륨의 일별 스냅샷을 자동으로 생성하며, 이 스냅샷도 구성된 백업 보존 기간 동안 보존됩니다.

  백업 보존 기간 이후에 스냅샷을 보존하려면 AWS Management Console 및 AWS Command Line Interface ()를 사용하여 언제든지 수동 스냅샷을 시작할 수도 있습니다AWS CLI. 자세한 내용은 Amazon DocumentDB에서 백업 및 복원 단원을 참조하십시오.

  마이그레이션을 계획할 때 다음을 고려하세요.

  - 목표 복구 시점(RPO)를 충족하는 1~35일의 백업 보존 기간을 선택합니다.
  - 수동 스냅샷이 필요한지, 필요할 경우 어떤 간격으로 필요한지 결정합니다.

마이그레이션 접근 방식

데이터를 Amazon DocumentDB로 마이그레이션하는 세 가지 기본 접근 방식이 있습니다.

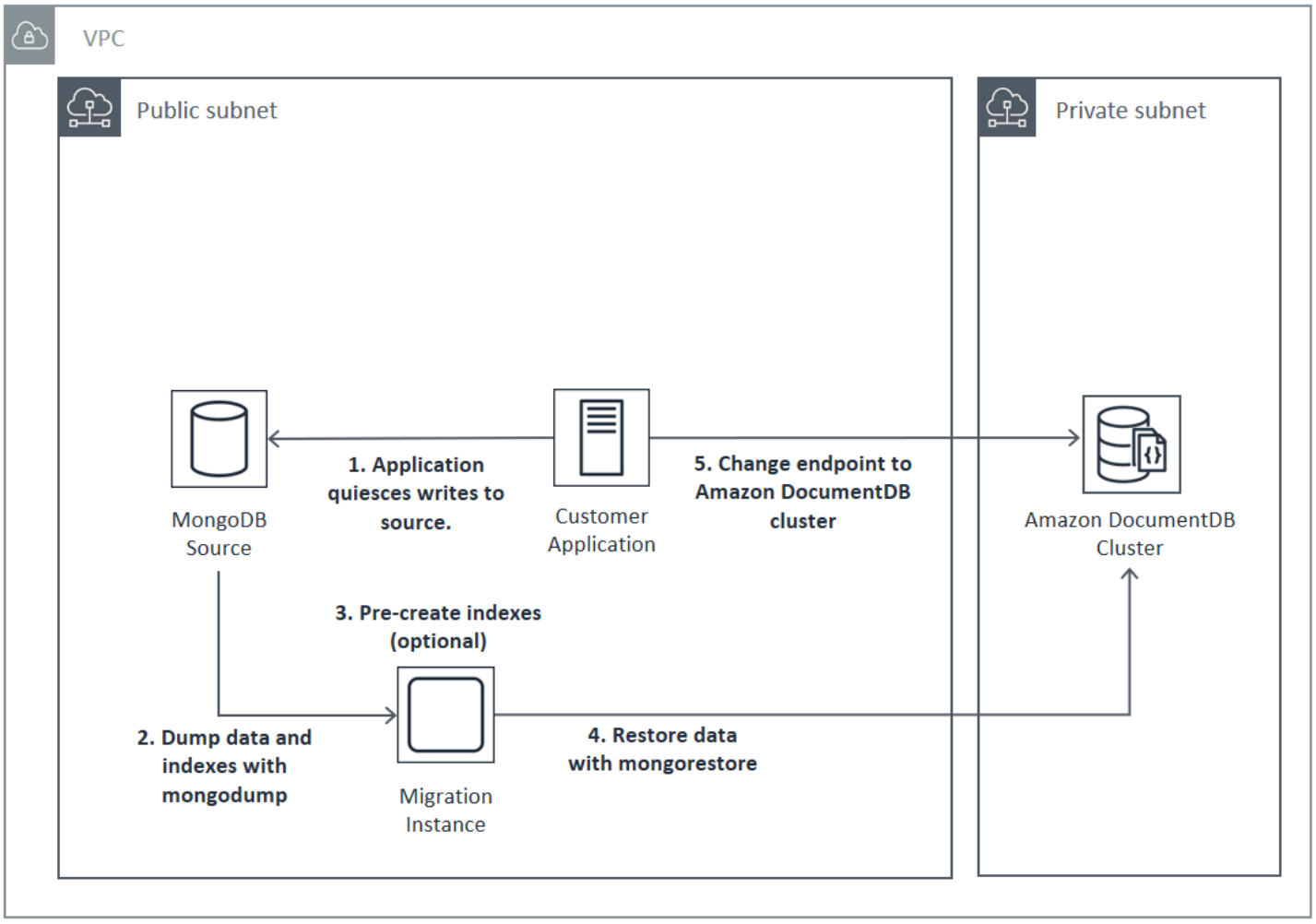
> **Note**
>
> Amazon DocumentDB에서 언제든 인덱스를 생성할 수 있지만 대규모 데이터셋을 가져오기 전에 인덱스를 생성하는 속도가 전반적으로 더 빠릅니다. 가장 좋은 방법은 아래의 각 접근 방법에 대해 마이그레이션을 수행하기 전에 먼저 Amazon DocumentDB에서 인덱스를 생성하는 것입니다. 이렇게 하려면 Amazon DocumentDB 인덱스 도구를 사용하면 됩니다.

오프라인

오프라인 접근 방식은 `mongodump` 및 `mongorestore` 도구를 사용하여 소스 MongoDB 배포에서 Amazon DocumentDB 클러스터로 데이터를 마이그레이션할 수 있습니다. 오프라인은 가장 간편한 마이그레이션 접근 방식이지만 클러스터에 대해 가장 많은 가동 중지 시간이 발생합니다.

오프라인 마이그레이션에 대한 기본 프로세스는 다음과 같습니다.

1. MongoDB 소스에 대한 쓰기를 중단합니다.
2. 소스 MongoDB 배포에서 컬렉션 데이터 및 인덱스를 덤프합니다.
3. 탄력적 클러스터로 마이그레이션하는 경우 `sh.shardCollection()` 명령을 사용하여 샤딩된 컬렉션을 생성합니다. 인스턴스 기반 클러스터로 마이그레이션하는 경우 다음 단계로 건너뜁니다.
4. Amazon DocumentDB 클러스터에 인덱스를 복원합니다.
5. 컬렉션 데이터를 Amazon DocumentDB 클러스터에 복원합니다.
6. 애플리케이션 엔드포인트를 변경하여 Amazon DocumentDB 클러스터에 씁니다.



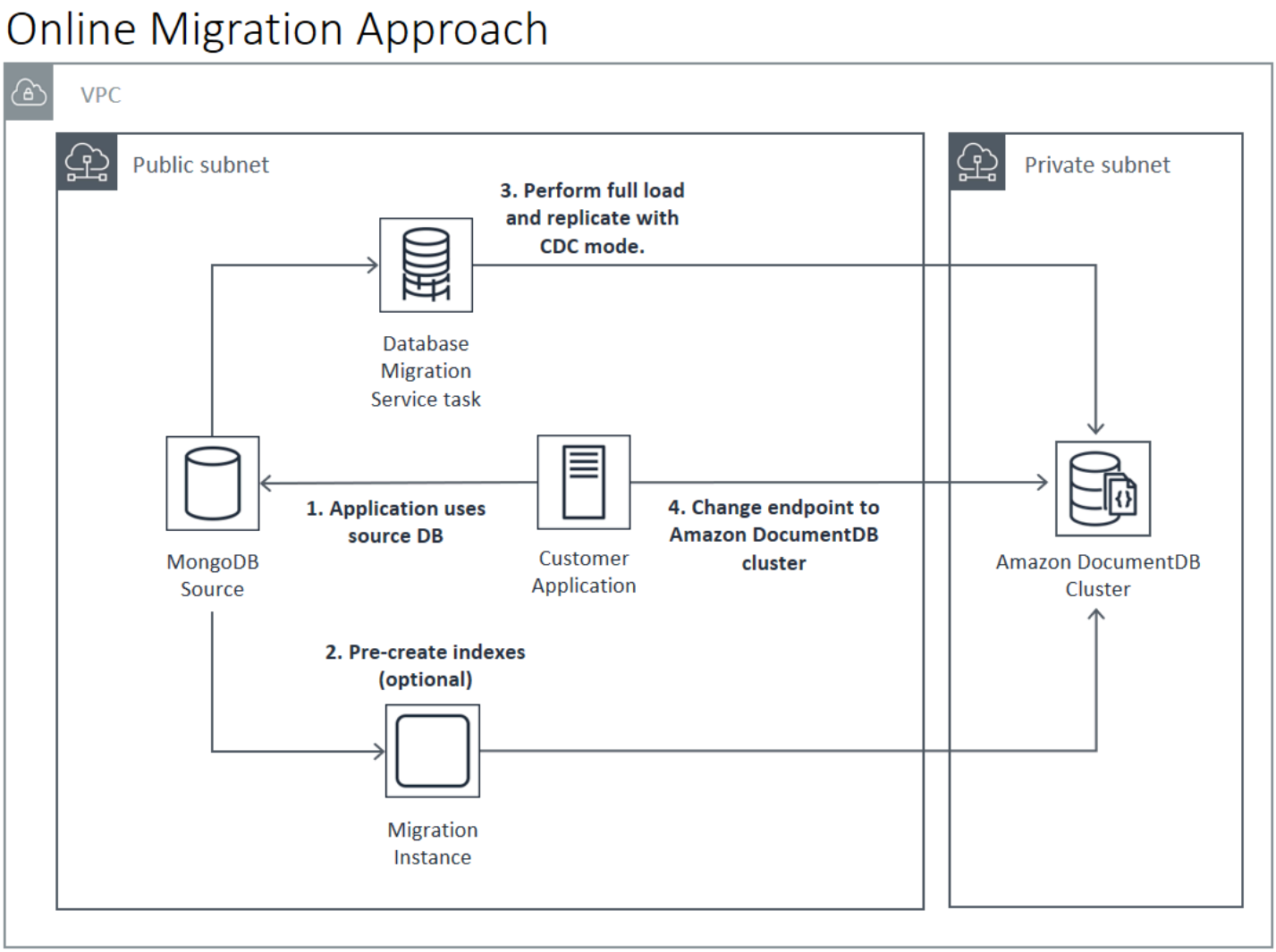
온라인

온라인 접근 방식은 AWS Database Migration Service (AWS DMS)를 사용합니다. 원본 MongoDB 배포로부터 Amazon DocumentDB 클러스터로의 데이터 전체 로드를 수행합니다. 그런 다음 변경 데이터 캡처(CDC) 모드로 전환하여 변경 사항을 복제합니다. 온라인 접근 방식은 클러스터 가동 중단을 최소화하지만 세 가지 방법 중 가장 느립니다.

온라인 마이그레이션에 대한 기본 프로세스는 다음과 같습니다.

1. 애플리케이션은 일반적으로 소스 DB를 사용합니다.
2. 탄력적 클러스터로 마이그레이션하는 경우 `sh.shardCollection()` 명령을 사용하여 샤딩된 컬렉션을 생성합니다. 인스턴스 기반 클러스터로 마이그레이션하는 경우 다음 단계로 건너뜁니다.
3. Amazon DocumentDB 클러스터에서 인덱스를 미리 생성합니다.

4. AWS DMS 태스크를 생성하여 전체 로드를 수행한 다음 소스 MongoDB 배포에서 Amazon DocumentDB 클러스터로 CDC를 활성화합니다.
5. AWS DMS 태스크가 전체 로드를 완료하고 Amazon DocumentDB에 대한 변경 사항을 복제한 후 애플리케이션의 엔드포인트를 Amazon DocumentDB 클러스터로 전환합니다.



를 사용하여 마이그레이션 AWS DMS 하는 방법에 대한 자세한 내용은 AWS Database Migration Service 사용 설명서의 의 대상으로 Amazon DocumentDB 사용 AWS Database Migration Service 및 관련 자습서를 참조하세요.

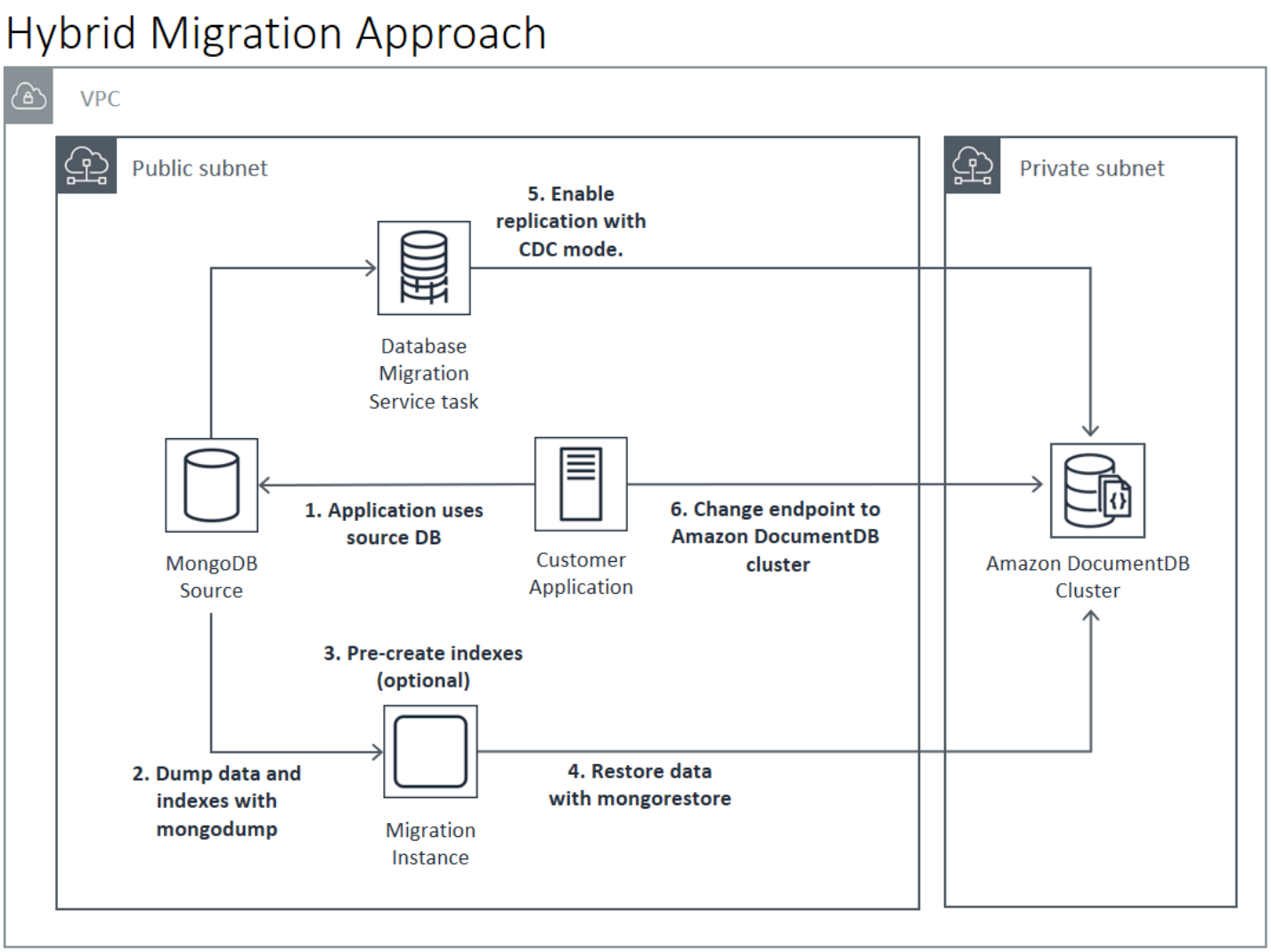
하이브리드

하이브리드 접근 방식은 `mongodump` 및 `mongorestore` 도구를 사용하여 소스 MongoDB 배포에서 Amazon DocumentDB 클러스터로 데이터를 마이그레이션할 수 있습니다. 그런 다음 CDC AWS DMS

모드에서를 사용하여 변경 사항을 복제합니다. 하이브리드 접근 방식은 마이그레이션 속도와 가동 중지 시간의 균형을 맞추지만, 세 가지 접근 방식 중 가장 복잡합니다.

하이브리드 마이그레이션에 대한 기본 프로세스는 다음과 같습니다.

1. 애플리케이션은 일반적으로 소스 MongoDB DB 배포를 사용합니다.
2. 소스 MongoDB 배포에서 컬렉션 데이터 및 인덱스를 덤프합니다.
3. Amazon DocumentDB 클러스터에 인덱스를 복원합니다.
4. 탄력적 클러스터로 마이그레이션하는 경우 `sh.shardCollection()` 명령을 사용하여 샤딩된 컬렉션을 생성합니다. 인스턴스 기반 클러스터로 마이그레이션하는 경우 다음 단계로 건너뜁니다.
5. 컬렉션 데이터를 Amazon DocumentDB 클러스터에 복원합니다.
6. 원본 MongoDB 배포에서 Amazon DocumentDB 클러스터로의 CDC를 활성화하는 AWS DMS 작업을 생성합니다.
7. AWS DMS 작업이 허용 기간 내에 변경 사항을 복제하는 경우 애플리케이션 엔드포인트를 변경하여 Amazon DocumentDB 클러스터에 기록합니다.



어떤 마이그레이션 접근 방식을 선택하든 데이터를 마이그레이션하기 전에 Amazon DocumentDB 클러스터에서 미리 인덱스를 생성하는 것이 가장 효율적입니다. 왜냐하면 Amazon DocumentDB 인덱스는 병렬로 삽입되는 데이터이지만, 기존 데이터에 대한 인덱스 생성은 단일 스레드 작업이기 때문입니다.

AWS DMS 는 인덱스(데이터만)를 마이그레이션하지 않으므로 인덱스를 두 번째로 생성하지 않으려면 추가 단계가 필요하지 않습니다.

마이그레이션 원본

MongoDB 원본이 독립 실행형 mongo 프로세스일 때 온라인 또는 하이브리드 마이그레이션 접근 방식을 사용하려는 경우 먼저 CDC 원본으로 사용할 oplog가 생성되도록 독립 실행형 mongo를 복제본 집합으로 변환합니다.

MongoDB 복제본 집합 또는 샤드 있는 클러스터로부터 마이그레이션하는 경우 각 복제본 집합 또는 샤드에 대해 마이그레이션 원본으로 사용할 묶이거나 숨겨진 보조 복제본 집합 또는 샤드를 생성할 것을 고려하세요. 데이터 덤프를 수행할 경우 작업 집합 데이터가 메모리를 벗어나게 되어 프로덕션 인스턴스에서의 성능에 악영향을 미칠 수 있습니다. 프로덕션 데이터를 처리하지 않는 노드로부터 마이그레이션함으로써 이 위험을 완화할 수 있습니다.

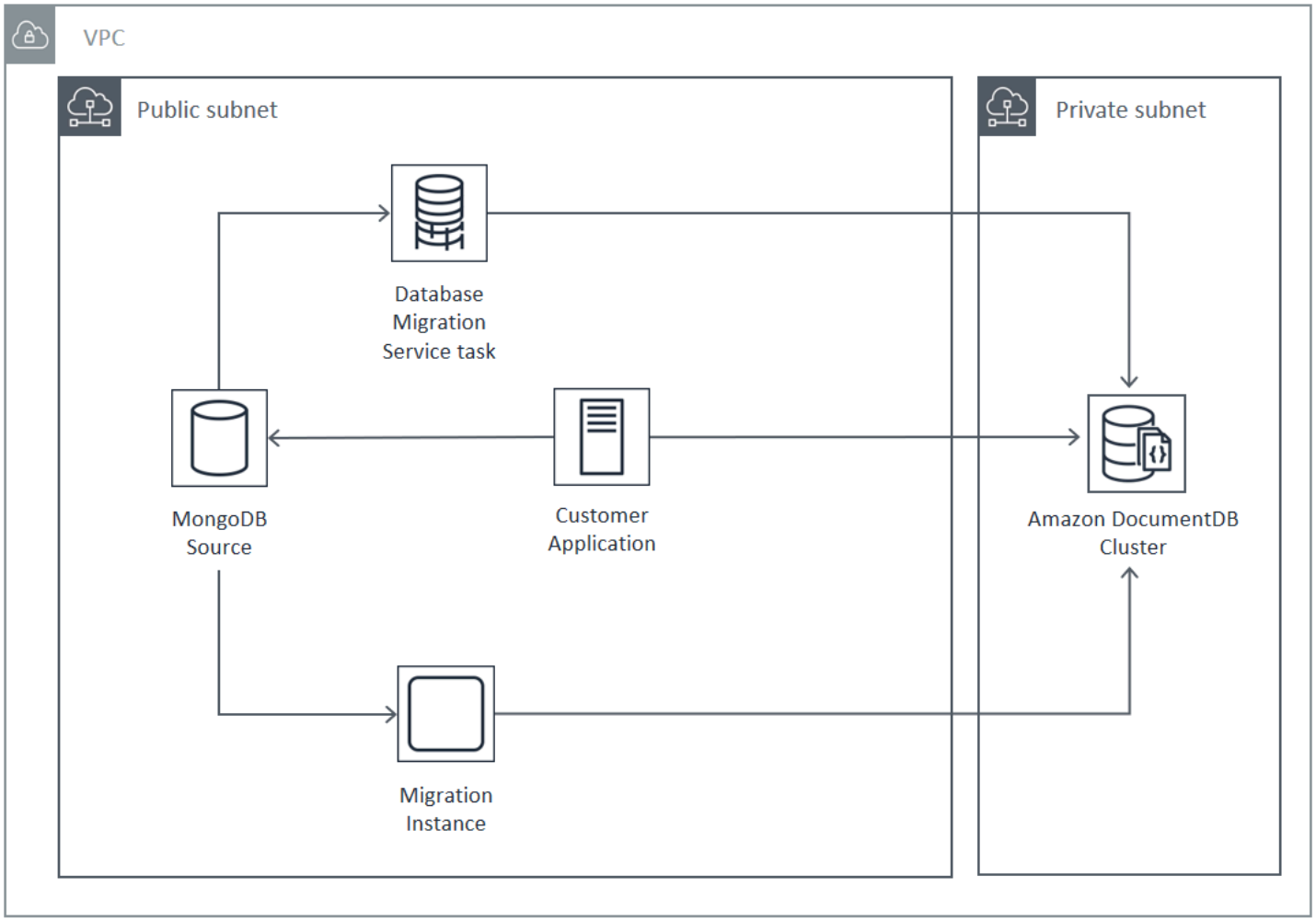
마이그레이션 원본 버전

소스 MongoDB 데이터베이스 버전이 대상 Amazon DocumentDB 클러스터의 호환성 버전과 다른 경우, 성공적인 마이그레이션을 위해 다른 준비 단계를 수행해야 할 수 있습니다. 가장 일반적인 두 가지 요구 사항은 소스 MongoDB 설치를 마이그레이션을 위해 지원되는 버전(MongoDB 버전 3.0 이상)으로 업그레이드해야 하는 것과 대상 Amazon DocumentDB 버전을 지원하도록 애플리케이션 드라이버를 업그레이드해야 하는 것입니다.

마이그레이션에 이러한 요구 사항이 있을 경우 마이그레이션 계획에 이러한 단계를 포함시켜 드라이버 변경 사항을 업그레이드 및 테스트하도록 하세요.

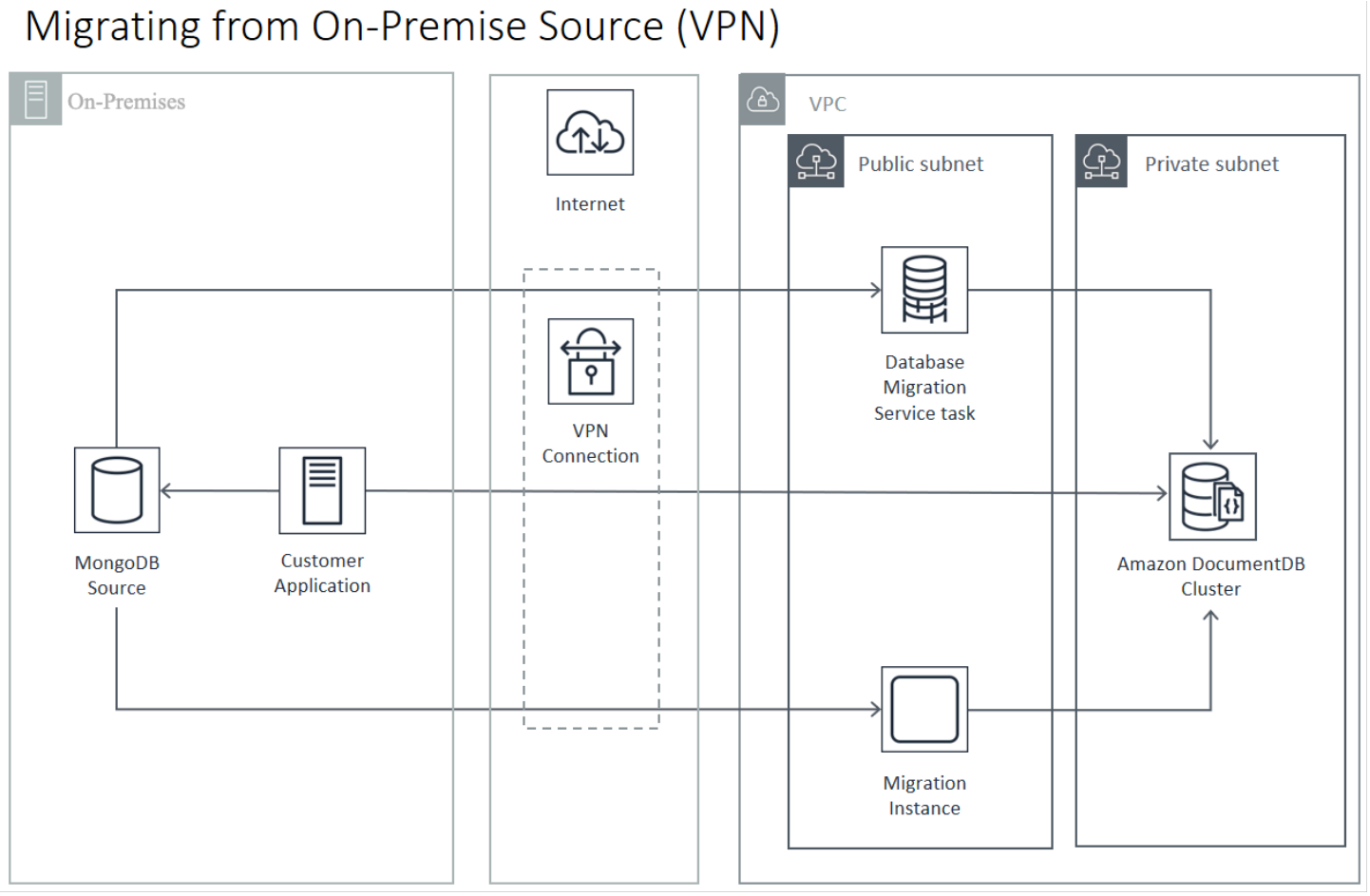
마이그레이션 연결

데이터 센터에서 실행 중인 소스 MongoDB 배포 또는 Amazon EC2 인스턴스에서 실행 중인 MongoDB 배포에서 Amazon DocumentDB로 마이그레이션할 수 있습니다. EC2에서 실행되는 MongoDB로부터 마이그레이션은 간단하며 보안 그룹 및 서브넷만 올바로 설정하면 됩니다.

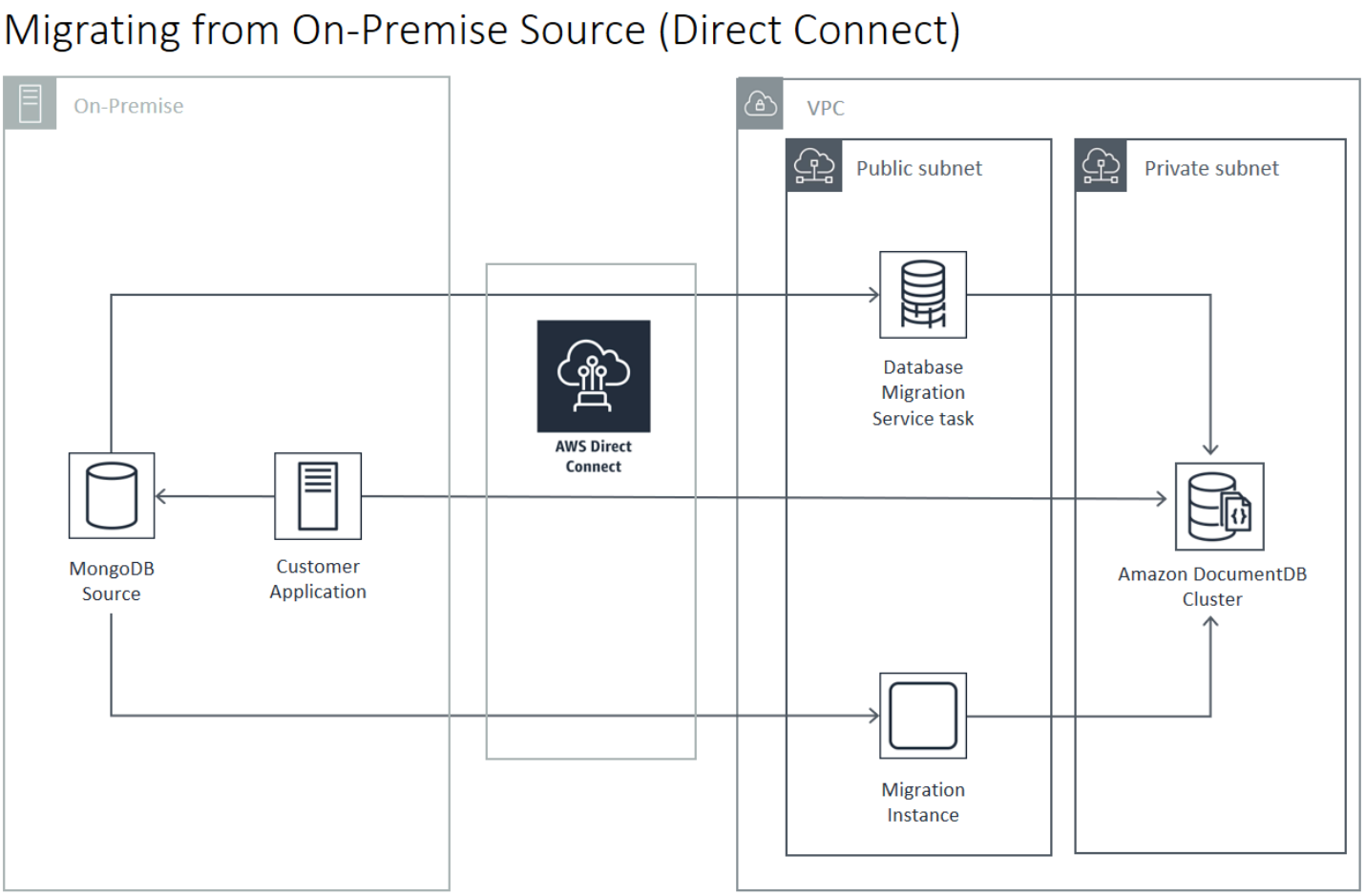


온프레미스 데이터베이스로부터 마이그레이션하려면 MongoDB 배포와 가상 프라이빗 클라우드(VPC) 간 연결이 필요합니다. 이를 위해 Virtual Private Network(VPN) 연결을 사용하거나 AWS Direct Connect 서비스를 사용할 수 있습니다. 인터넷을 통해 VPC로 마이그레이션할 수 있지만, 이 연결 방법은 보안 측면에서 가장 바람직하지 않습니다.

다음 다이어그램은 VPN 연결을 통해 온프레미스 소스에서 Amazon DocumentDB로 마이그레이션하는 것을 보여줍니다.



다음은 AWS Direct Connect을 사용한 온프레미스 소스에서 Amazon DocumentDB로의 마이그레이션입니다.



온라인 및 하이브리드 마이그레이션 접근 방식의 경우 AWS DMS 인스턴스를 사용해야 하며, 이 인스턴스는 Amazon VPC의 Amazon EC2에서 실행되어야 합니다. 모든 접근 방식에서 마이그레이션 서버가 `mongodump` 및 `mongorestore`를 실행해야 합니다. 일반적으로 Amazon DocumentDB 클러스터가 시작된 VPC의 Amazon EC2 인스턴스에서 마이그레이션 서버를 실행하는 것이 더 간편합니다. Amazon DocumentDB 클러스터 연결이 매우 간단해지기 때문입니다.

테스트

다음은 마이그레이션 전 테스트의 목표입니다.

- 선택한 접근 방식이 원하는 마이그레이션 결과를 달성하는지 확인합니다.
- 인스턴스 유형 및 읽기 기본 설정 선택이 애플리케이션 성능 요구 사항을 충족하는지 확인합니다.
- 장애 조치 시 애플리케이션 동작을 확인합니다.

마이그레이션 계획 테스트 고려 사항

Amazon DocumentDB 마이그레이션 계획을 테스트할 때 다음을 고려하세요.

주제

- 인덱스 복원
- 데이터 덤핑
- 데이터 복구
- Oplog 크기 설정
- AWS Database Migration Service 구성
- 샤드 있는 클러스터로부터 마이그레이션

인덱스 복원

기본적으로 mongorestore가 덤프된 모음의 인덱스를 생성하지만, 데이터가 복원된 이후에 생성합니다. 데이터가 클러스터에 복원되기 전에 Amazon DocumentDB에서 인덱스를 생성하는 것이 전반적으로 더 빠릅니다. 이는 데이터 로드 도중 인덱싱 작업이 병행되기 때문입니다.

인덱스를 사전 생성하기로 선택한 경우 `--noIndexRestore` 옵션을 지정하면 mongorestore을 사용하여 데이터를 복원할 때 인덱스 생성 단계를 건너뛸 수 있습니다.

데이터 덤핑

mongodump 도구는 원본 MongoDB 배포에서 데이터를 덤핑하는 기본 방법입니다. 마이그레이션 인스턴스에서 사용 가능한 리소스에 따라 `--numParallelCollections` 옵션을 사용하여 덤핑되는 병렬 연결 수를 기본 4개에서 더 늘려 mongodump 속도를 높일 수 있습니다.

데이터 복구

`mongorestore` 도구는 덤핑된 데이터를 Amazon DocumentDB 인스턴스로 복원하는 기본 방법입니다. 복원 시 `--numInsertionWorkersPerCollection` 옵션을 사용하여 각 모음의 작업자 수를 늘려 복원 성능을 개선할 수 있습니다. Amazon DocumentDB 클러스터 기본 인스턴스 내 vCPU 1개당 작업자 1개로 시작하는 것이 좋습니다.

Amazon DocumentDB는 현재 `mongorestore` 도구의 `--oplogReplay` 옵션을 지원하지 않습니다.

기본적으로 `mongorestore`는 삽입 오류를 건너뛰고 복원 프로세스를 계속합니다. 이 오류는 지원하지 않는 데이터를 Amazon DocumentDB 인스턴스로 복원할 경우 발생할 수 있습니다. 예를 들어 null 문자열이 있는 키 또는 값을 포함하는 문서가 있을 경우 이런 현상이 생길 수 있습니다. 어떤 복원 오류

가 발생하더라도 mongorestore 작업이 완전히 실패하도록 하려면 --stopOnError 옵션을 사용하세요.

Oplog 크기 설정

MongoDB 작업 로그(oplog)는 데이터베이스에 대한 모든 데이터 수정을 포함하는 제한 모음입니다. 복제본 집합 또는 샤드 번호에서 db.printReplicationInfo() 작업을 실행하여 포함된 oplog의 크기 및 시간 범위를 확인할 수 있습니다.

온라인 또는 하이브리드 접근 방식을 사용하는 경우 각 복제본 세트 또는 샤드의 옵로그가 전체 데이터 마이그레이션 프로세스 기간(mongodump 또는 AWS DMS 작업 전체 로드) 동안 이루어진 모든 변경 사항과 합리적인 버퍼를 포함할 수 있을 만큼 충분히 큰지 확인합니다. 자세한 내용은 MongoDB 설명서의 Oplog 크기 확인 섹션을 참조하세요. mongodump 또는 mongorestore 프로세스 또는 AWS DMS 전체 로드 작업의 첫 번째 테스트 실행에 소요된 시간을 기록하여 필요한 최소 oplog 크기를 결정합니다.

AWS Database Migration Service 구성

AWS Database Migration Service 사용자 가이드에서는 MongoDB 원본 데이터를 Amazon DocumentDB 클러스터로 마이그레이션하는 데 필요한 구성 요소 및 단계를 설명합니다. 다음은를 사용하여 온라인 또는 하이브리드 마이그레이션을 수행하는 기본 프로세스입니다 AWS DMS .

를 사용하여 마이그레이션을 수행하려면 AWS DMS

1. MongoDB 원본 엔드포인트를 생성합니다. 자세한 내용은 AWS DMS에서 MongoDB를 원본으로 사용 섹션을 참조하세요.
2. Amazon DocumentDB 대상 엔드포인트를 생성합니다. 자세한 내용은 AWS DMS 엔드포인트를 사용한 작업 섹션을 참조하세요.

   대상 엔드포인트를 탄력적 클러스터로 구성하는 경우, 기존 Amazon DocumentDBS SL 인증서는 탄력적 클러스터에서 작동하지 않으므로 다음 단계를 따라 엔드포인트에 새 SSL 인증서를 연결해야 합니다.

   a. https://www.amazontrust.com/repository/SFSRootCAG2.pem을 방문하여 콘텐츠를 "SFSrootCag2.pem" 파일로 저장하세요. 이 파일은 이후 단계에서 가져올 인증서 파일입니다.

   b. 엘라스틱 클러스터 엔드포인트를 생성할 때 엔드포인트 구성에서 새 CA 인증서 추가를 선택합니다.

   - 인증서 식별자에 SFSRootCAG2.pem을 입력합니다.

- 인증서 파일 가져오기에서 파일 선택을 선택하고 이전에 다운로드한 SFSRootCAG2.pem 파일로 이동합니다. 파일을 선택하고 엽니다. 인증서 가져오기를 선택한 다음 인증서 선택 드롭다운에서 SFSRootCAG2.pem 항목을 선택합니다.

3. AWS DMS 복제 인스턴스를 하나 이상 생성합니다. 자세한 내용은 AWS DMS 복제 인스턴스 작업을 참조하세요.
4. AWS DMS 복제 작업을 하나 이상 생성합니다. 자세한 내용은 AWS DMS 작업을 사용한 작업 섹션을 참조하세요.

   온라인 마이그레이션의 경우 마이그레이션 작업이 기존 데이터의 마이그레이션 및 지속적인 변경 복제 마이그레이션 유형을 사용합니다.

   하이브리드 마이그레이션의 경우 마이그레이션 작업이 데이터 변경 사항만 복제 마이그레이션 유형을 사용합니다. mongodump 작업의 덤프 시간에 맞춰 CDC 시작 시간을 선택할 수 있습니다. MongoDB oplog는 멱동성(idempotent)을 갖습니다. 변경 사항 누락을 방지하려면 mongodump 종료 시간과 CDC 시작 시간을 몇 분 동안 중첩시키는 것이 좋습니다.

샤드 있는 클러스터로부터 마이그레이션

MongoDB 공유 클러스터에서 Amazon DocumentDB 인스턴스로 데이터를 마이그레이션하는 프로세스는 기본적으로 여러 복제본 세트 마이그레이션을 병렬로 수행하는 프로세스입니다. 샤드 있는 클러스터 마이그레이션을 테스트할 때 중요한 고려 사항은 일부 샤드가 다른 샤드보다 빈번히 사용된다는 것입니다. 이러한 상황 때문에 데이터 마이그레이션 소요 시간이 달라집니다. 계획 및 테스트 시 각 샤드의 oplog 요구 사항을 평가해야 합니다.

다음은 샤드 있는 클러스터를 마이그레이션할 때 고려할 몇 가지 구성 문제입니다.

- mongodump를 실행하거나 AWS DMS 마이그레이션 작업을 시작하기 전에 샤드 있는 클러스터 밸런서를 비활성화하고 프로세스 내 마이그레이션이 모두 완료될 때까지 기다려야 합니다. 자세한 내용은 MongoDB 설명서의 밸런서 비활성화 섹션을 참조하세요.
- 를 사용하여 데이터를 복제 AWS DMS 하는 경우 마이그레이션 작업을 실행하기 전에 각 샤드에서 cleanupOrphaned 명령을 실행합니다. 이 명령을 실행하지 않을 경우 작업이 중복 문서 ID 때문에 실패할 수 있습니다. 이 명령은 성능에 영향을 미칠 수 있습니다. 자세한 내용은 MongoDB 설명서에서cleanupOrphaned를 참조하세요.
- mongodump 도구를 사용하여 데이터를 덤핑하는 경우 샤드당 1개의 mongodump 프로세스를 실행해야 합니다. 가장 시간 효율적인 접근 방식은 여러 마이그레이션 서버를 사용하여 덤프 성능을 극대화하는 것입니다.

- AWS Database Migration Service 를 사용하여 데이터를 복제하는 경우 각 샤드에 대한 소스 엔드포인트를 생성해야 합니다. 또한 마이그레이션하는 각 샤드에 대해 하나 이상의 마이그레이션 작업을 실행합니다. 가장 시간 효율적인 접근 방식은 여러 복제 인스턴스를 사용하여 마이그레이션 성능을 극대화하는 것입니다.

성능 테스트

성공적으로 데이터를 테스트 Amazon DocumentDB 클러스터로 마이그레이션했으면 클러스터에 대해 테스트 워크로드를 실행합니다. 성능이 MongoDB 소스 배포의 현재 처리량을 충족하거나 초과하는지 Amazon CloudWatch 지표를 통해 확인합니다.

다음과 같은 주요 Amazon DocumentDB 지표를 확인하세요.

- 네트워크 처리량
- 쓰기 처리량
- 읽기 처리량
- 복제본 지연 시간

자세한 내용은 Amazon DocumentDB 모니터링 단원을 참조하십시오.

장애 조치 테스트

Amazon DocumentDB 장애 조치 이벤트 중 애플리케이션의 동작이 사용자의 가용성 요구 사항을 충족하는지 확인합니다. 콘솔에서 Amazon DocumentDB 클러스터의 수동 장애 조치를 시작하려면 클러스터 페이지의 작업 메뉴에서 장애 조치 작업을 선택합니다.

또한 AWS CLI에서 `failover-db-cluster` 작업을 실행하여 장애 조치를 시작할 수도 있습니다. 자세한 내용은 AWS CLI 참조의 `failover-db-cluster` Amazon DocumentDB 섹션에서 섹션을 참조하세요.

추가 리소스

AWS Database Migration Service 사용 설명서의 다음 항목을 참조하세요.

- Amazon DocumentDB를의 대상으로 사용 AWS Database Migration Service
- 안내서: MongoDB에서 Amazon DocumentDB로 마이그레이션

마이그레이션 플레이북: MongoDB에서 Amazon DocumentDB로

이 마이그레이션 플레이북은 MongoDB 데이터베이스에서 Amazon DocumentDB로 마이그레이션하는 데 도움이 되는 리소스와 단계를 제공합니다.

마이그레이션 프로세스

다음은 MongoDB 데이터베이스에서 Amazon DocumentDB 로 데이터를 마이그레이션하는 데 일반적으로 수반되는 상위 수준의 단계입니다.

주제

- 1단계: 호환성 및 기능적 차이점
- 2단계: 개념 증명
- 3단계: 데이터 마이그레이션
- 4단계: 데이터 검증
- 5단계: 애플리케이션 전환

1단계: 호환성 및 기능적 차이점

Amazon DocumentDB는 Apache 2.0 오픈 소스 MongoDB 3.6, 4.0 및 5.0 API와 상호작용합니다. 따라서 변경 사항이 거의 없거나 전혀 없는 동일한 MongoDB 드라이버, 애플리케이션 및 도구를 Amazon DocumentDB와 함께 사용할 수 있습니다.

첫 번째 단계는 애플리케이션이 MongoDB 데이터베이스에서 사용하는 연산자와 인덱스 간의 호환성과 Amazon DocumentDB에서의 가용성을 확인하고 이들 간의 기능적 차이를 이해하는 것입니다.

연산자 호환성

Amazon DocumentDB 호환성 도구*를 사용하여 애플리케이션이 쿼리에서 지원되지 않는 연산자를 사용하는지 쉽게 확인할 수 있습니다. 이 도구는 MongoDB 데이터베이스 서버 로그 파일 또는 애플리케이션 소스 코드를 스캔하여 지원되지 않는 연산자의 보고서를 제공할 수 있습니다. 지원되지 않는 연산자의 사용을 발견한 경우 지원되지 않는 연산자를 처리하도록 애플리케이션을 수정해야 합니다.

설정에 사용되는 MongoDB 연산자와 지원되는 Amazon DocumentDB 연산자 간의 호환성을 확인하려면 다음을 실행합니다.

```
git clone https://github.com/awslabs/amazon-documentdb-tools.git
cd amazon-documentdb-tools/compat-tool/
```

```
python3 compat.py --version <Amazon DocumentDB version> --directory <mongodb logfiles/
source code>
```

자세한 내용은 Amazon DocumentDB에서 지원되는 MongoDB API, 작업 및 데이터 형식 단원을 참조하십시오.

* AWS에서 공식적으로 지원되지 않습니다.

인덱스 호환성

Amazon DocumentDB 인덱스 도구*를 사용하여 Amazon DocumentDB에서 지원되지 않는 인덱스 유형을 사용하고 있는지 확인할 수 있습니다. 이 도구에서 인덱스 정의를 읽으려면 소스 데이터베이스에 연결해야 합니다.

이렇게 하려면 먼저 `--dump-indexes` 옵션을 사용하여 인덱스 정의를 디렉터리에 덤프해야 합니다. 그런 다음 `--show-issues` 옵션을 사용해 도구를 실행하여 호환되지 않는 인덱스를 찾을 수 있는 디렉터리를 제공합니다.

인덱스 내보내기:

```
git clone https://github.com/awslabs/amazon-documentdb-tools.git
sudo pip install -r amazon-documentdb-tools/index-tool/requirements.txt
mkdir <directory to dump index definitions>
python3 migrationtools/documentdb_index_tool.py --dump-indexes --dir <directory> --uri
 <source-mongodb-uri>
```

호환되지 않는 인덱스 확인:

```
python3 migrationtools/documentdb_index_tool.py --show-issues --dir <dumped-index-
definitions-directory>
```

지원되지 않는 인덱스 유형의 사용을 발견한 경우 호환되지 않는 인덱스 없이 작업하거나 계속하도록 애플리케이션 또는 데이터 모델을 수정해야 합니다.

Amazon DocumentDB 에서 지원되는 인덱스 유형 및 속성에 대한 자세한 내용은 인덱스 및 인덱스 속성 및 Amazon DocumentDB 에서 인덱싱하는 방법을 참조하세요.

* AWS에서 공식적으로 지원되지 않습니다.

기능적 차이

차이점에 익숙해지려면 MongoDB와 기능적 차이 항목을 검토합니다.

2단계: 개념 증명

Amazon DocumentDB에서 애플리케이션 또는 일반 테스트 제품군을 실행하여 개념 증명을 수행해 기능과 성능을 테스트합니다. 테스트를 수행하려면 Amazon DocumentDB 클러스터에 데이터를 채워야 할 수 있습니다. 예를 들어 `mongodump` 및 `mongorestore` 도구를 사용하여 원본 MongoDB에서 데이터를 복사할 수 있습니다.

기능 테스트

Amazon DocumentDB 클러스터를 생성하고(Amazon DocumentDB 클러스터 생성 참조) 애플리케이션 또는 기능 테스트 제품군을 실행하여 모든 애플리케이션 워크플로가 Amazon DocumentDB에서 원활하게 계속 작동하는지 확인합니다.

성능 테스트

프로덕션 워크로드와 유사한 워크로드로 Amazon DocumentDB에서 실행되는 애플리케이션 또는 성능 테스트 제품군에서 성능 테스트를 실행하여 설정이 지연 시간 요구 사항을 충족하는지 확인합니다. 성능을 위해 워크로드를 미세 조정하거나 Amazon DocumentDB 클러스터를 필요에 따라 규모를 조정합니다. 자세한 내용은 성능 및 리소스 사용률 문제 해결 및 Amazon DocumentDB 클러스터 규모 조정 섹션을 참조하세요.

최적의 성능을 위해 적절한 인스턴스 유형으로 Amazon DocumentDB 클러스터의 크기를 조정하는 것이 중요합니다. 자세한 내용은 인스턴스 크기 조정 모범 사례를 참조하세요.

장애 조치 테스트

애플리케이션이 Amazon DocumentDB 프라이머리 노드 재부팅, 프라이머리 노드 장애 조치 또는 다중 노드 클러스터의 기본 노드 삭제에 어떻게 응답하는지, 복제본 노드가 재부팅되거나 제거되는지 관찰할 수 있습니다. 이렇게 하면 애플리케이션이 이러한 상황에 복원력을 갖추고 있는지 확인하는 데 도움이 됩니다. 자세한 내용은 장애 조치 테스트 단원을 참조하십시오.

애플리케이션이 허용해야 하는 예외 사항과 이를 효율적으로 처리하는 방법을 이해하려면 Amazon DocumentDB를 사용하여 복원력 있는 애플리케이션 구축하기를 참조하세요.

> **Note**
>
> Amazon DocumentDB에서 워크로드를 테스트하는 다른 방법은 없습니다.

3단계: 데이터 마이그레이션

개념 증명이 성공하면 데이터를 Amazon DocumentDB로 마이그레이션합니다. 대부분의 고객은 온라인 또는 오프라인 마이그레이션 접근 방식을 사용하여 데이터를 마이그레이션합니다.

온라인 마이그레이션

온라인 마이그레이션 방법을 사용하면 몇 기가바이트에서 여러 테라바이트에 이르는 원본 데이터베이스의 데이터를 가동 중지 시간이 거의 없는 Amazon DocumentDB로 마이그레이션할 수 있습니다. 자세한 내용은 AWS Database Migration Service (AWS DMS)를 참조하세요.

MongoDB 데이터베이스에서 마이그레이션하는 경우를 사용하여 전체 로드를 수행하고 진행 중인 변경 사항을 복제 AWS DMS 할 수 있습니다.

단계별 프로세스는 온라인 메서드를 이용하여 Amazon DocumentDB로 마이그레이션을 참조하세요.

추가 정보는 AWS Database Migration Service 사용 설명서의 Amazon DocumentDB를 대상으로 사용 AWS Database Migration Service 섹션에서 확인할 수 있습니다.

참고할 사항 AWS DMS:

- 세분화:를 사용하여 멀티 테라바이트 데이터베이스를 마이그레이션 AWS DMS할 때 DMS의 전체 로드는 기본적으로 컬렉션당 단일 스레드이므로 기본 설정으로 속도가 느려져 마이그레이션 시간이 길어질 수 있습니다. 대규모 데이터베이스 마이그레이션에 대한 전체 로드 속도를 높이려면 AWS DMS의 분할 기능을 사용할 수 있습니다.

  에서 분할을 사용하는 방법에 대한 자세한 내용은에서 자동 분할 사용을 AWS DMS참조하세요. AWS DMS
- DMS 인스턴스 유형: 데이터 마이그레이션 속도를 높이려면 올바른 DMS 인스턴스를 선택해야 합니다.

오프라인 마이그레이션

오프라인 마이그레이션은 데이터베이스를 Amazon DocumentDB로 이동하는 가장 간단한 접근 방식입니다. 이 접근 방식은 주로 POC와 마이그레이션 중에 쓰기 가동 중지 시간이 발생할 수 있는 워크로드에 사용됩니다.

단계별 프로세스는 오프라인 메서드를 사용하여 MongoDB에서 Amazon DocumentDB로 마이그레이션을 참조하세요.

4단계: 데이터 검증

데이터가 성공적으로 마이그레이션되면 데이터의 정확성을 검증하여 확신을 얻습니다. AWS DMS 마이그레이션 작업 콘솔에서 마이그레이션된 데이터 지표를 찾을 수 있습니다. 자세한 내용은 마이그레이션된 데이터 검증. 섹션을 참조하세요.

Amazon DocumentDB DataDiffer Tool*을 사용하여 소스 컬렉션과 대상 컬렉션 간의 데이터 일관성을 검증할 수도 있습니다.

* AWS에서 공식적으로 지원되지 않습니다.

5단계: 애플리케이션 전환

여기에는 Amazon DocumentDB 클러스터를 사용하도록 애플리케이션의 데이터베이스 연결 문자열을 변경하는 작업이 포함됩니다.

Amazon DocumentDB에 연결하는 방법에 대한 자세한 내용은 Amazon DocumentDB에 복제본 세트로 연결을 참조하세요.

온라인 마이그레이션

전체 데이터 로드가 완료되면 AWS DMS 는 계속 진행하여 소스에서 Amazon DocumentDB로 진행 중인 변경 사항을 복제합니다. 변경 사항이 포착되고 데이터 검증 확인이 완료되면 Amazon DocumentDB 로 전환 작업을 수행할 수 있습니다.

오프라인 마이그레이션

전체 데이터 로드 및 데이터 검증 검사를 완료하면 Amazon DocumentDB로 전환 작업을 수행할 수 있습니다.

추가 리소스

다음은 마이그레이션에 도움이 될 수 있는 몇 가지 추가 리소스입니다.

- 동영상: Amazon DocumentDB로 마이그레이션하는 모범 사례
- 동영상: Amazon DocumentDB 관찰성 및 모니터링 시작하기
- 추가 유틸리티: Amazon DocumentDB 도구*
- 마이그레이션 개발자 안내서: Amazon DocumentDB로 마이그레이션

* AWS에서 공식적으로 지원되지 않습니다.

Amazon DocumentDB 인플레이스 주요 버전 업그레이드

Amazon DocumentDB는 일반적으로 광범위한 테스트를 거친 후에만 데이터베이스 엔진의 새 버전을 만듭니다. Amazon DocumentDB 클러스터를 새 버전으로 업그레이드할 방법과 시기를 선택할 수 있습니다.

현재 Amazon DocumentDB는 Amazon DocumentDB 3.6, 4.0, 5.0의 세 가지 주요 버전을 지원합니다. 클러스터의 엔드포인트, 스토리지 및 태그를 동일하게 유지하면서 데이터베이스의 MVU(인플레이스 주요 버전 업그레이드)를 수행할 수 있으며 수정 없이 애플리케이션을 계속 사용할 수 있습니다. 이 기능은 Amazon DocumentDB 5.0을 사용할 수 있는 모든 지역에서 무료로 사용할 수 있습니다.

> **Important**
>
> 인플레이스 주요 버전 업그레이드 중에는 Amazon DocumentDB 클러스터를 사용할 수 없게 되며 클러스터가 여러 번 재부팅됩니다. 업그레이드를 시작한 후에는 클러스터에 연결, 읽기 또는 쓰기를 하지 마십시오. 업그레이드 가동 중지 시간은 컬렉션, 인덱스, 데이터베이스 및 인스턴스의 수에 따라 클러스터마다 다를 수 있습니다. 유지 관리 기간이나 사용률이 낮은 시간에 업그레이드를 수행하는 것이 좋습니다. 클러스터를 업그레이드한 후에는 클러스터를 이전 버전으로 다운그레이드할 수 없지만 업그레이드 전 스냅샷을 새 클러스터로 복원하도록 선택할 수 있습니다.

주제

- MVU 사전 조건 및 제한 사항
- 인플레이스 주요 버전 업그레이드 모범 사례
- 인플레이스 주요 버전 업그레이드 수행
- Amazon DocumentDB 3.6/4.0에서 업그레이드된 Amazon DocumentDB 5.0 클러스터와 새 Amazon DocumentDB 5.0 클러스터 간의 차이점
- 인플레이스 주요 버전 업그레이드 문제 해결

MVU 사전 조건 및 제한 사항

업그레이드를 수행하기 전에 이해하고 조치를 취해야 할 수 있는 인플레이스 주요 버전 업그레이드의 사전 요구 사항 및 제한 사항은 다음과 같습니다.

- 인스턴스 유형 - Amazon DocumentDB 4.0/5.0은 r4.* 인스턴스를 지원하지 않습니다. 인플레이스 주요 버전 업그레이드를 진행하려면 r4.* 인스턴스를 r5.* 인스턴스로 수정하세요. 자세한 정보는 Amazon DocumentDB 인스턴스 수정을 참조하세요. Amazon DocumentDB 엔진 버전을 기반으로 지원되는 인스턴스에 대해서는 리전별 지원되는 인스턴스 클래스을 참조하세요.
- 인스턴스 OS 패치 - 인플레이스 주요 버전 업그레이드를 진행하려면 최신 운영 체제(OS) 패치가 필요합니다. 현재 위치 업그레이드를 진행하기 전에 보류 중인 OS 유지 관리 작업을 인스턴스에 적용하세요. 자세한 내용은 Amazon DocumentDB 운영 체제 업데이트 단원을 참조하십시오.

> **Note**
>
> 보류 중인 클러스터 수준 엔진 패치가 있는 경우 인스턴스 OS 패치가 보이지 않는 경우도 있습니다. 인스턴스 OS 패치를 적용하고 이후 인플레이스 주요 버전 업그레이드를 진행하기 전에, 클러스터 수준 엔진 패치를 적용해야 할 수 있습니다. 클러스터의 엔진 버전에 대한 패치 업데이트 수행을(를) 참조하세요.

- Amazon DocumentDB 5.0을 사용할 수 있는 모든 지역에서 인플레이스 주요 버전 업그레이드가 가능합니다.
- Amazon DocumentDB 4.0을 대상 버전으로 사용하는 경우 인플레이스 주요 버전 업그레이드는 지원되지 않습니다.
- Amazon DocumentDB 4.0부터는 사용자 이름의 "."가 지원되지 않습니다. Amazon DocumentDB 3.6에서 5.0으로 업그레이드하고 "."가 포함된 사용자 이름이 있는 경우, 인플레이스 MVU를 실행하기 전에 "." 없이 사용자 이름을 다시 생성하세요.
- 인플레이스 주요 버전 업그레이드는 Amazon DocumentDB 글로벌 클러스터 및 엘라스틱 클러스터에서 지원되지 않습니다.

> **Note**
>
> 글로벌 클러스터를 업그레이드하려면 글로벌 클러스터에서 보조 클러스터를 삭제하고, 기본 클러스터를 리전 클러스터로 변환하고, 리전(기본) 클러스터에서 인플레이스 주요 버전 업그레이드를 수행한 다음, 이전과 동일한 엔드포인트를 유지하기 위해 동일한 이름을 사용하는 보조 클러스터를 추가하여 글로벌 클러스터를 다시 생성하세요. 업그레이드된 기본 클러스터에서 새로 추가된 보조 클러스터에 데이터를 복제하는 동안에는 I/O 요금이 발생한다는 점에 유의하세요. 삭제하기 전에 글로벌 클러스터에서 보조 클러스터를 제거하는 방법에 대한 자세한 단계는 Amazon DocumentDB 글로벌 클러스터에서 클러스터 분리을 참조하세요.

- 인덱스 수가 많고(3,000 초과) 성능 버스트 가능 인스턴스(예: t3.medium 또는 t4g.medium)에서 작업하는 경우, 기본 인스턴스를 더 큰 인스턴스(예: 최소 r5.xlarge)로 스케일 업하여 인플레이스 주요

버전 업그레이드를 수행할 수 있도록 해야 합니다. 인플레이스 주요 버전 업그레이드가 완료되면 인스턴스 크기를 축소할 수 있습니다. 인플레이스 주요 버전 업그레이드를 위해 db.t3 및 db.t4g 인스턴스 유형에서 지원되는 최대 인덱스 수는 아래 표를 참조하세요.

Instance	인플레이스 MVU에 지원되는 최대 인덱스
db.t4g.medium	3천
db.t3.medium	1만

인플레이스 주요 버전 업그레이드 모범 사례

주제

- 복제된 클러스터를 사용하여 인플레이스 주요 버전 업그레이드를 테스트합니다.
- 인플레이스 주요 버전 업그레이드를 하기 전에
- 인플레이스 주요 버전 업그레이드를 하는 동안
- 인플레이스 주요 버전 업그레이드를 한 후

복제된 클러스터를 사용하여 인플레이스 주요 버전 업그레이드를 테스트합니다.

1. 인플레이스 주요 버전 업그레이드를 테스트하려면 빠른 복제 기능을 사용하여 대상 클러스터의 클론을 생성하는 것이 좋습니다. 클러스터의 데이터를 수정하지 않는 한, 복제된 볼륨에서 인플레이스 주요 버전 업그레이드를 테스트하는 데 스토리지 비용이 발생하지 않습니다. 복제에 대한 자세한 내용은 Amazon DocumentDB 클러스터에 대한 볼륨 복제 섹션을 참조하세요.
2. 인플레이스 주요 버전 업그레이드를 완료하는 데 걸리는 시간을 보다 현실적으로 예측하려면 복제된 클러스터의 인스턴스 수를 대상 클러스터와 일치시키세요.
3. 새로 업그레이드된 Amazon DocumentDB 5.0 클러스터에서 기능적 차이가 있는지 완전히 테스트하여 모든 것이 예상대로 작동하는지 확인하는 것이 좋습니다.

인플레이스 주요 버전 업그레이드를 하기 전에

1. 버전 호환 클러스터 파라미터 그룹을 준비합니다.

새 엔진 버전에는 Amazon DocumentDB 기본 클러스터 파라미터 그룹을 사용하거나 새 엔진 버전을 위한 사용자 지정 클러스터 파라미터 그룹을 생성합니다.

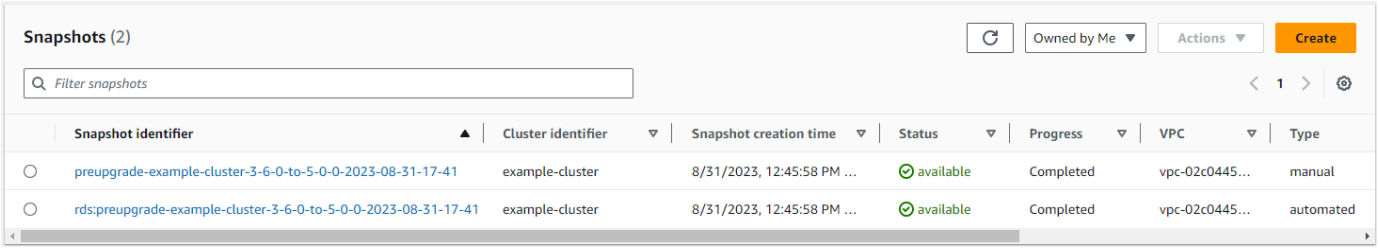
업그레이드 요청의 일부로 Amazon DocumentDB 클러스터 파라미터 그룹을 연결하는 경우, 인플레이스 주요 버전 업그레이드가 클러스터를 재부팅하여 새 파라미터 그룹을 적용합니다.

2. 사전 요구 사항 및 제한 섹션에 설명된 대로 인플레이스 주요 버전 업그레이드의 사전 요구 사항을 충족했는지 확인하세요.
3. 수동 스냅샷을 생성하는 방법

업그레이드 프로세스는 업그레이드 중에 DB 클러스터의 스냅샷을 생성합니다. 업그레이드 프로세스 전에 직접 수동 스냅샷을 생성하는 것이 좋습니다. 수동 클러스터 스냅샷 생성을(를) 참조하세요.

> **Note**
>
> 업그레이드 프로세스에서 생성된 자동 스냅샷은 인플레이스 주요 버전 업그레이드가 완료된 후에도 자동으로 삭제되지 않습니다. 이 스냅샷은 보존 기간 내에 있는 한 요금이 부과되지 않습니다. 클러스터의 성공적인 업그레이드가 확인되면 이 스냅샷을 삭제하도록 선택할 수 있습니다.

스냅샷의 이름은 `preupgrade-<name>-<version>-<timestamp>`로 지정됩니다.



4. 클러스터의 인플레이스 주요 버전 업그레이드를 이미 예약했는지 확인하세요.

클러스터를 수정하고 다음 유지 관리 창에서 적용하도록 선택한 경우, 인플레이스 주요 버전 업그레이드 일정은 콘솔에 표시되지 않지만 CLI에서 볼 수 있습니다. `describe-db-clusters` 명령을 실행하여 인플레이스 메이저 버전 업그레이드가 이미 예약되어 있는지 확인할 수 있습니다.

```
aws docdb describe-db-cluster \
  --region us-east-1 \
  --db-cluster-identifier mydocdbcluster
```

위 예제에서 각 ### ## ## ##자를 클러스터 정보로 바꿉니다.

명령은 다음 출력을 반환합니다.

```
"PendingModifiedValues": {
    "EngineVersion": "5.0.0"
},
```

5. 하위 환경에서 볼륨 클론을 사용하여 여러 번의 시험 실행을 수행하여 실행 계획 및 기능적 차이에 대해 클러스터 사후 주요 버전 업그레이드를 테스트하세요. 인플레이스 주요 버전 업그레이드 실행 시간을 더 잘 예측하려면 동일한 수와 크기의 인스턴스로 복제하는 것이 좋습니다. 자세한 내용은 Amazon DocumentDB 클러스터에 대한 볼륨 복제 단원을 참조하십시오.
6. 이전 단계가 성공적이면 프로덕션 클러스터에서 인플레이스 주요 버전 업그레이드를 진행하세요.

인플레이스 주요 버전 업그레이드를 하는 동안

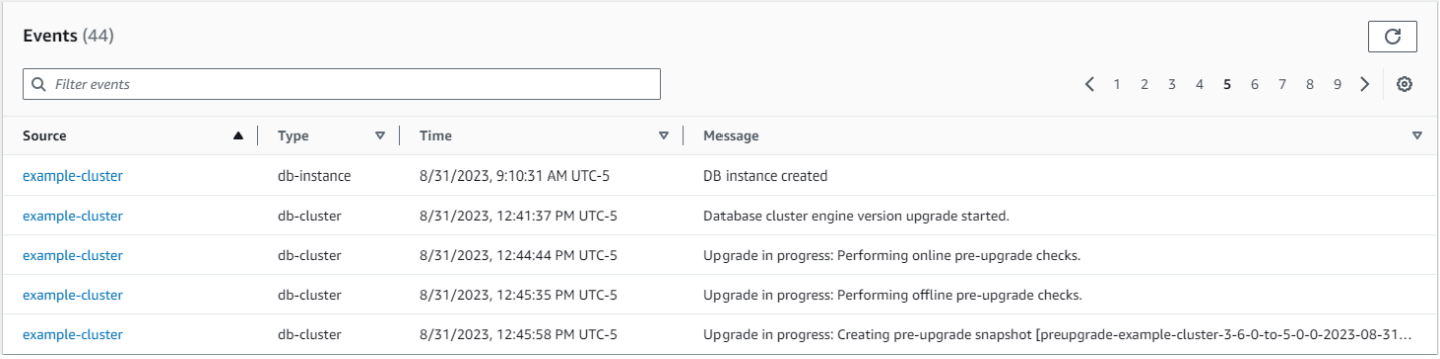
클러스터 유지 관리 이벤트를 구독하면 인플레이스 주요 버전 업그레이드 진행 상황을 모니터링할 수 있습니다. 업그레이드가 완료되면 '데이터베이스 클러스터 주요 버전이 업그레이드되었습니다.' 이벤트가 수신됩니다. 이 이벤트와 업그레이드 중에 발생하는 기타 이벤트는 Amazon DocumentDB 콘솔의 클러스터 세부 정보 페이지에 있는 '이벤트 및 태그' 섹션에 표시됩니다. 그러면 클러스터 상태가 '업그레이드'에서 '사용 가능'으로 변경됩니다.

CLI에서 `aws docdb create-event-subscription`을 실행하여 이벤트를 생성하고 `aws docdb describe-events`를 실행하여 진행 상황을 모니터링할 수 있습니다. 또한 위 이벤트에 대한 이벤트 알림을 Amazon SNS에 이메일, 푸시 메시지 및 기타 방법을 통해 알림을 받을 대상으로 설정할 수 있습니다. 자세한 내용은 Amazon DocumentDB 이벤트 구독 단원을 참조하십시오.

인플레이스 주요 버전 업그레이드는 업그레이드 중에 다음과 같은 이벤트를 생성합니다.

- 업그레이드 진행 중: 업그레이드 전 스냅샷 생성 [preupgrade-<cluster-name>-<timestamp>]
- 업그레이드 진행 중: 볼륨 복제.
- 업그레이드 진행 중: 라이터 업그레이드.
- 업그레이드 진행 중: 리더 업그레이드.
- 데이터베이스 클러스터 엔진 주요 버전이 업그레이드되었습니다.

이벤트는 이벤트 페이지 아래 콘솔에서도 볼 수 있습니다.



에서 describe-events 명령을 실행하여 진행 상황을 추적할 AWS CLI수 있습니다.

```
aws docdb describe-events
  --source-identifier mydocdbcluster
  --source-type db-cluster
```

위 예제에서 각 ### ## ## ##자를 클러스터 정보로 바꿉니다.

명령은 다음 출력을 반환합니다.

```
{
    "Events": [
        {
            "SourceIdentifier": "mydocdbcluster",
            "SourceType": "db-cluster",
            "Message": "Database cluster engine version upgrade started.",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2023-07-11T23:20:32.444000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:xxxx:cluster:mycluster"
        }
    ]
}
```

인플레이스 주요 버전 업그레이드를 한 후

Amazon DocumentDB 3.6의 경우 클러스터에 태그를 추가하여 클러스터가 새로 생성된 Amazon DocumentDB 5.0 클러스터와 달리 Amazon DocumentDB 3.6에서 Amazon DocumentDB 5.0으로 업그레이드되었음을 구별하세요. 업그레이드된 Amazon DocumentDB 5.0 클러스터와 새 Amazon DocumentDB 5.0 클러스터 간의 차이점에 대한 섹션을 참조하세요.

업그레이드 후 상태로 복원해야 하는 경우 인플레이스 MVU가 완료된 후 수동 스냅샷을 생성합니다. 인플레이스 주요 버전 업그레이드가 완료되는 즉시 자동 스냅샷 프로세스가 재개됩니다. 이 수동 스냅샷은 보존 기간 내에 있는 한 요금이 부과되지 않습니다.

Amazon DocumentDB 5.0과 관련된 새로운 기능(예: 클라이언트 측 필드 레벨 암호화)을 사용하려면 드라이버 버전을 MongoDB 5.0 API 버전으로 업그레이드하는 것이 좋습니다. 자세한 내용은 Amazon DocumentDB 5.0의 새로운 기능 섹션의 Amazon DocumentDB 5.0 기능 목록을 참조하세요.

> **Important**
>
> 인플레이스 주요 버전 업그레이드(MVU)를 수행한 직후 Amazon DocumentDB 5.0 클러스터는 데이터베이스 엔진의 쿼리 실행 계획 최적화에 기반해 인덱스 메타데이터를 다시 채웁니다. 인덱스 메타데이터 재계산 프로세스가 완료되면 Amazon DocumentDB 클러스터에서 예상되는 쿼리 성능이 재개됩니다. 일반적으로 이 프로세스는 몇 분 내에 완료되지만 클러스터의 인덱스 수에 따라 최대 2시간 동안 지속될 수 있습니다. 인플레이스 MVU 후 라이터 인스턴스를 즉시 재부팅, 장애 조치 또는 스케일 업/다운하면 클러스터의 인덱스 메타데이터 계산 프로세스가 중단될 수 있습니다. 인플레이스 MVU가 완료되면 Amazon DocumentDB 5.0 클러스터에서 예상 쿼리 성능을 관찰한 후 이러한 변경 조치를 수행하는 것이 좋습니다.
>
> 또한 현재 위치 MVU가 완료되면 사용 가능한 변경 스트림 데이터가 지난 3시간으로 제한됩니다.
>
> 이 임시 성능 저하가 현재 위치 MVU 후 2시간 이상 지속되는 경우 AWS 지원팀에 문의하십시오.

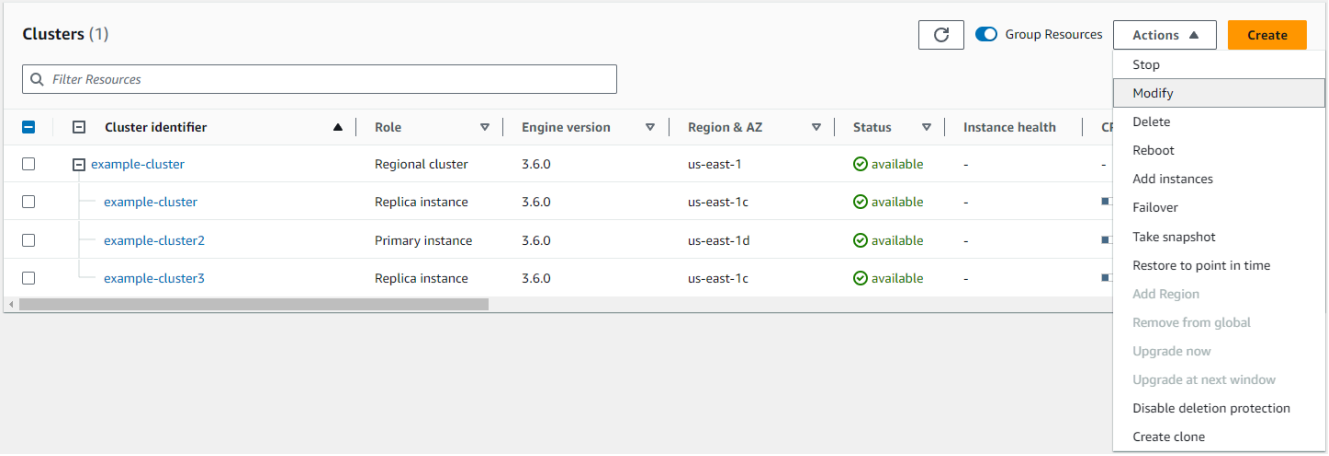
업그레이드된 Amazon DocumentDB 5.0 클러스터 전체를 테스트하여 모든 것이 예상대로 작동하는지 확인하세요.

인플레이스 주요 버전 업그레이드 수행

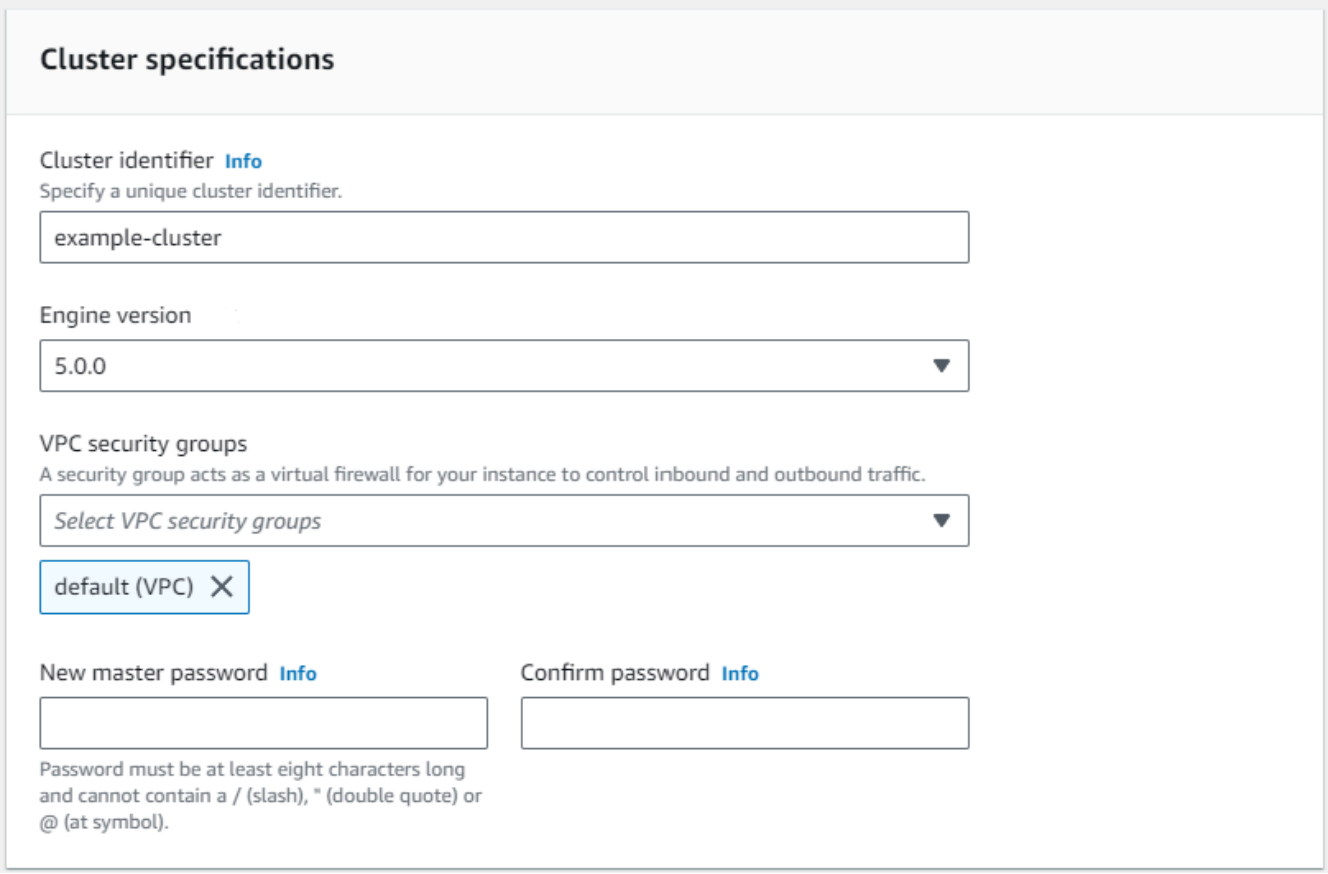
Using the AWS Management Console

AWS Management Console를 사용해 인플레이스 주요 버전 업그레이드를 수행하려면

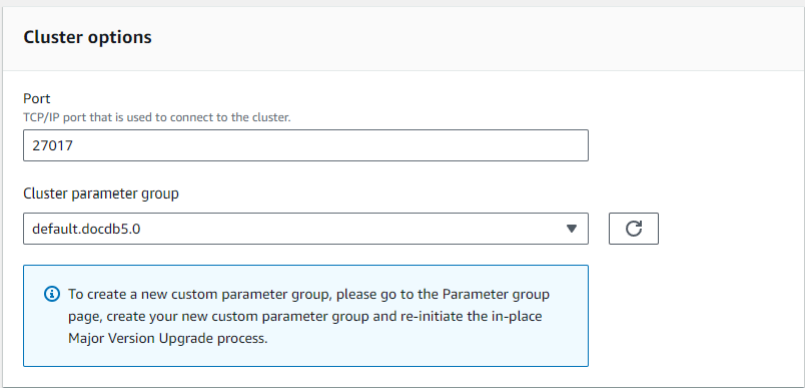
1. AWS Management Console에 로그인하고 Amazon DocumentDB 콘솔을 엽니다.
2. 클러스터 테이블에서 원본 클러스터를 선택하고 작업을 클릭한 다음 수정을 클릭합니다.



3. 클러스터 사양 섹션의 클러스터 수정 대화 상자의 엔진 버전 드롭다운 메뉴에서 대상 데이터베이스 버전(5.0.0)을 선택합니다.

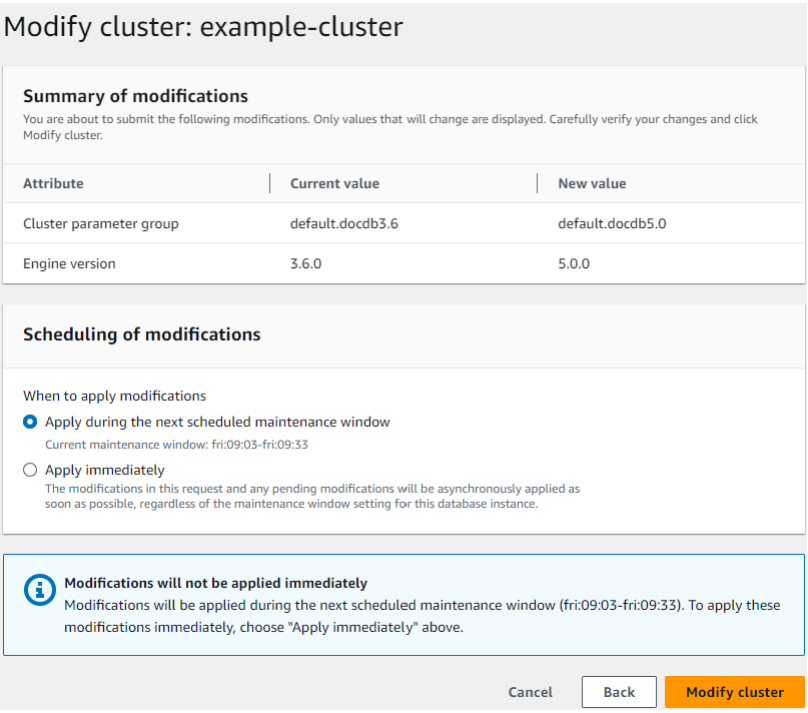


4. 클러스터 옵션 섹션에서 적절한 클러스터 파라미터 그룹(default.docdb5.0) 또는 사용자 지정 생성 파라미터 그룹을 선택합니다.

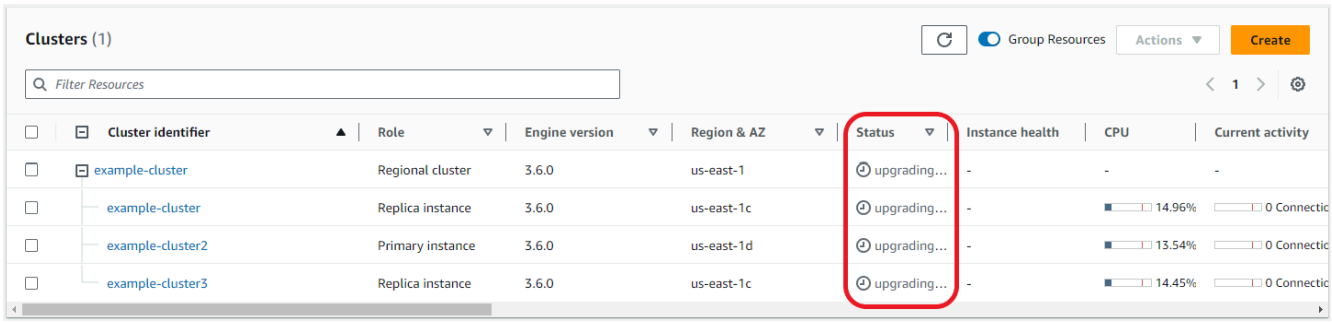


5. 완료되면 아래로 스크롤하여 계속을 선택합니다.
6. 수정 일정 예약 섹션에서 원하는 일정 계획(즉시 적용 또는 다음 유지 관리 창에서 적용)을 선택합니다.

   그런 다음 클러스터 수정을 선택합니다.



7. 클러스터 테이블에서 업그레이드 중인 클러스터의 상태를 기록해 둡니다.



Using the AWS CLI

원하는 엔진 버전 옵션 및 `allow-major-version-upgrade` 플래그 세트와 함께 `modify-db-cluster` 명령을 사용합니다.

```
aws docdb modify-db-cluster \
  --db-cluster-identifier mydocdbcluster \
  --allow-major-version-upgrade \
  --engine-version 5.0.0 \
  --apply-immediately \
  --cluster-parameter-group mydocdbparametergroup \
  --region us-east-1
```

위 예제에서 각 *### ## ## ##*자를 클러스터 정보로 바꿉니다.

Amazon DocumentDB 3.6/4.0에서 업그레이드된 Amazon DocumentDB 5.0 클러스터와 새 Amazon DocumentDB 5.0 클러스터 간의 차이점

- 인플레이스 주요 버전 업그레이드는 업그레이드된 클러스터의 원래 인덱스를 유지합니다. Amazon DocumentDB 5.0을 사용하면 특히 카디널리티가 낮은 인덱스의 경우 인덱스 유지 관리 및 가비지 수집 프로세스의 전반적인 효율성을 높였습니다. 일반적으로 MVU가 성공적으로 완료된 후 reindex 명령을 사용하여 인덱스를 다시 생성하는 것이 좋습니다. 인덱스를 재생성하는 것은 요구 사항이 아니며 추가 I/O가 필요합니다. 자세한 내용은 단원을 참조하십시오를 사용한 인덱스 유지 관리 `reIndex`.
- 여러 숫자 데이터 형식에 대한 하위 문서 비교:
  - 클러스터가 Amazon DocumentDB 3.6에서 마이그레이션되는 경우, 클러스터는 Amazon DocumentDB 3.6 하위 문서 비교 동작을 상속합니다. 함수의 차이는 하위 문서의 숫자 유형(예:

Long, Double, Decimal128)으로 제한됩니다. 예를 들어, Amazon DocumentDB 3.6에서는 `{a: {b: 1}}`는 `{a: {b: {NumberLong(1)}}`과 같지 않지만, Amazon DocumentDB 4.0 이후에서는 동일하다고 비교됩니다.

- 이 하위 문서 비교 동작은 인플레이스 주요 버전 업그레이드를 사용하여 버전 3.6에서 업그레이드된 Amazon DocumentDB 3.6 및 Amazon DocumentDB 5.0 클러스터에서만 존재합니다. 새로 생성된 Amazon DocumentDB 5.0 클러스터에는 적용되지 않습니다.

> **Note**
>
> Amazon DocumentDB 3.6/4.0과 Amazon DocumentDB 5.0 간의 기능적 차이 목록은 Amazon DocumentDB의 MongoDB 호환성을 참조하세요.

인플레이스 주요 버전 업그레이드 문제 해결

- 장애가 발생할 경우 인플레이스 주요 버전 업그레이드는 업그레이드 롤백을 시도하여 업그레이드가 시작되기 전 클러스터의 마지막 작동 상태를 가정합니다. 롤백에 성공하면 '데이터베이스 클러스터가 업그레이드할 수 없는 상태에 있습니다. DocumentDB 클러스터는 주요 버전 업그레이드를 성공적으로 완료할 수 없는 상태입니다.'라는 이벤트가 생성됩니다. 이때 AWS 지원 팀에 문의하여 버전 업그레이드 문제를 해결하고 다시 시도해야 합니다. 이전처럼 워크로드를 계속 사용할 수 있습니다. 업그레이드가 예상보다 오래 걸리는 다른 드문 시나리오에서는 AWS 지원 팀에 문의하여 도움을 받으세요.
- 인플레이스 MVU가 성공적으로 완료되면 인덱스 메타데이터 새로 고침 프로세스가 실행되는 동안 업그레이드된 클러스터가 잠시 동안 일시적인 성능 저하와 높은 CPU 사용률을 보일 수 있습니다. 2시간 이상 성능 저하가 계속 발생하는 경우 AWS 지원팀에 문의하십시오.

를 사용하여 Amazon DocumentDB 클러스터 업그레이드 AWS Database Migration Service

> **Important**
>
> Amazon DocumentDB는 MongoDB와 동일한 지원 수명 주기를 따르지 않으며 MongoDB의 수명 종료 일정은 Amazon DocumentDB에 적용되지 않습니다. Amazon DocumentDB 3.6의 수명 종료에 대한 계획은 현재 없으며, 기존 MongoDB 3.6 드라이버, 애플리케이션 및 도구는 Amazon DocumentDB에서 계속 사용할 수 있습니다.

를 사용하여 가동 중지 시간을 최소화하면서 Amazon DocumentDB 클러스터를 상위 버전으로 업그레이드할 수 있습니다 AWS DMS. AWS DMS 는 이전 Amazon DocumentDB 버전, 관계형 데이터베이스 및 비관계형 데이터베이스에서 대상 Amazon DocumentDB 클러스터로 쉽게 마이그레이션할 수 있는 완전 관리형 서비스입니다.

주제

- 1단계: 변경 스트림 활성화
- 2단계: 변경 스트림 보존 기간 수정
- 3단계: 인덱스 마이그레이션
- 4단계: AWS DMS 복제 인스턴스 생성
- 5단계: AWS DMS 소스 엔드포인트 생성
- 6단계: AWS DMS 대상 엔드포인트 생성
- 7단계: 마이그레이션 작업 생성 및 실행
- 8단계: 애플리케이션 엔드포인트를 대상 Amazon DocumentDB 클러스터로 변경

1단계: 변경 스트림 활성화

가동 중지 시간 마이그레이션을 최소화하려면 클러스터의 변경 스트림에 대한 액세스 권한이 AWS DMS 필요합니다. Amazon DocumentDB 변경 스트림 기능은 클러스터의 컬렉션과 데이터베이스 내에서 시간순으로 발생하는 업데이트 이벤트 시퀀스를 제공합니다. 변경 스트림에서를 읽으면 AWS DMS 가 변경 데이터 캡처(CDC)를 수행하고 대상 Amazon DocumentDB 클러스터에 증분 업데이트를 적용할 수 있습니다.

특정 데이터베이스의 모든 컬렉션에 대해 변경 스트림을 활성화하려면 mongo 쉘을 사용하여 Amazon DocumentDB 클러스터를 인증하고 다음 명령을 실행합니다.

```
db.adminCommand({modifyChangeStreams: 1,
    database: "db_name",
    collection: "",
    enable: true});
```

2단계: 변경 스트림 보존 기간 수정

그런 다음 변경 스트림에서 변경 이벤트를 보존하려는 기간에 따라 변경 스트림 보존 기간을 수정합니다. 예를 들어를 사용한 Amazon DocumentDB 클러스터 마이그레이션 AWS DMS 에 12시간이 걸릴 것으로 예상되는 경우 변경 스트림 보존을 12시간보다 큰 값으로 설정해야 합니다. Amazon DocumentDB 클러스터의 기본 보존 기간은 3시간입니다. AWS Management Console 또는를 사용하여 Amazon DocumentDB 클러스터의 변경 스트림 로그 보존 기간을 1시간에서 7일 사이로 수정할 수 있습니다 AWS CLI. 자세한 내용은 변경 스트림 로그 보존 기간 수정을 참조하세요.

3단계: 인덱스 마이그레이션

소스 Amazon DocumentDB 클러스터에 있는 것과 동일한 인덱스를 대상 Amazon DocumentDB 클러스터에 생성합니다. 는 데이터 마이그레이션을 AWS DMS 처리하지만 인덱스는 마이그레이션하지 않습니다. 인덱스를 마이그레이션하려면 Amazon DocumentDB 인덱스 도구를 사용하여 소스 Amazon DocumentDB 클러스터에서 인덱스를 내보냅니다. Amazon DocumentDB 도구 GitHub 리포지토리의 복제본을 생성하고 README.md의 지침을 따르며 도구를 다운로드할 수 있습니다. Amazon EC2 인스턴스 또는 Amazon DocumentDB 클러스터와 동일한 Amazon VPC에서 실행되는 AWS Cloud9 환경에서 도구를 실행할 수 있습니다.

다음은 자신의 정보를 각각의 *### ## ## ###*로 변경하는 예제입니다.

다음 코드는 소스 Amazon DocumentDB 클러스터의 인덱스를 덤프합니다.

```
python migrationtools/documentdb_index_tool.py --dump-indexes
--uri mongodb://sample-user:user-password@sample-source-cluster.node.us-
east 1.docdb.amazonaws.com:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false'
--dir ~/index.js/

2020-02-11 21:51:23,245: Successfully authenticated to database: admin2020-02-11
 21:46:50,432: Successfully connected to instance docdb-40-xx.cluster-xxxxxxxx.us-
east-1.docdb.amazonaws.com:27017
```

```
2020-02-11 21:46:50,432: Retrieving indexes from server...2020-02-11 21:46:50,440:
 Completed writing index metadata to local folder: /home/ec2-user/index.js/
```

인덱스를 성공적으로 내보내기 하면 대상 Amazon DocumentDB 클러스터에서 해당 인덱스를 복원합니다. 이전 단계에서 내보낸 인덱스를 복원하려면 Amazon DocumentDB 인데스 도구를 사용하세요. 다음 명령은 지정된 디렉터리에서 대상 Amazon DocumentDB 클러스터의 인덱스를 복원합니다.

```
python migrationtools/documentdb_index_tool.py --restore-indexes
--uri mongodb://sample-user:user-password@sample-destination-
cluster.node.us-east 1.docdb.amazonaws.com:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false'
--dir ~/index.js/

2020-02-11 21:51:23,245: Successfully authenticated to database: admin2020-02-11
 21:51:23,245: Successfully connected to instance docdb-50-xx.cluster-xxxxxxxx.us-
east-1.docdb.amazonaws.com:27017
2020-02-11 21:51:23,264: testdb.coll: added index: _id
```

인덱스가 제대로 복원되었는지 확인하려면 mongo 쉘을 사용하여 대상 Amazon DocumentDB 클러스터에 연결하고 해당 컬렉션의 인덱스를 나열하세요. 다음 코드를 확인합니다.

```
mongo --ssl
--host docdb-xx-xx.cluster-xxxxxxxx.us-east-1.docdb.amazonaws.com:27017
--sslCAFile rds-ca-2019-root.pem --username documentdb --password documentdb

db.coll.getIndexes()
```

4단계: AWS DMS 복제 인스턴스 생성

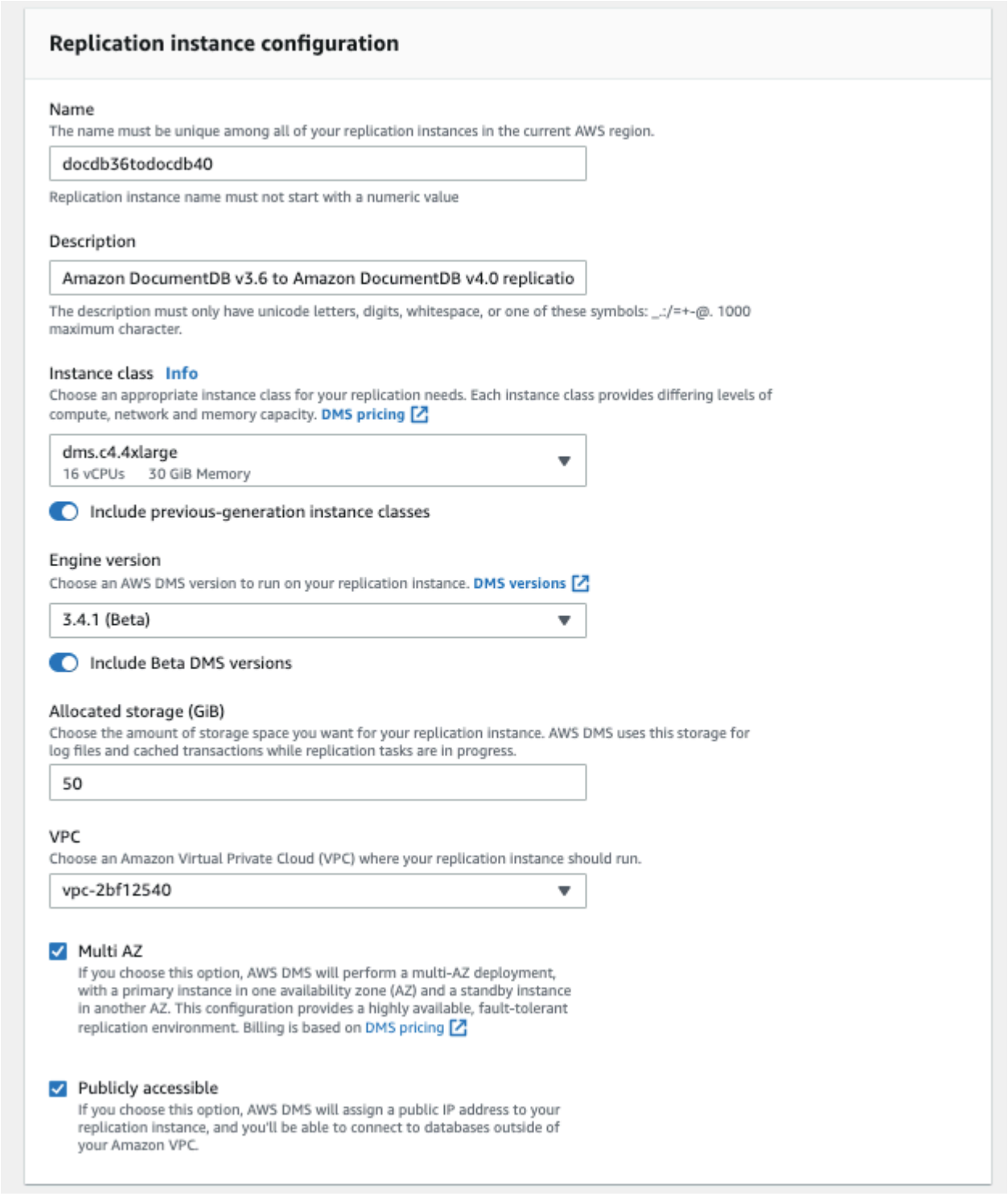
AWS DMS 복제 인스턴스는 소스 Amazon DocumentDB 클러스터에서 데이터를 연결하고 읽은 다음 대상 Amazon DocumentDB 클러스터에서 씁니다. AWS DMS 복제 인스턴스는 대량 로드 및 CDC 작업을 모두 수행할 수 있습니다. 이 절차 대다수는 메모리에서 진행됩니다. 그렇지만, 대규모 작업은 디스크에서 일부 버퍼링이 필요할 수 있습니다. 캐시된 트랜잭션과 로그 파일도 디스크에 기록됩니다. 데이터가 마이그레이션되면 복제 인스턴스는 소스와 대상이 동기화되어 있는지 확인하기 위해 변경 이벤트도 스트리밍합니다.

AWS DMS 복제 인스턴스를 생성하려면:

1. AWS DMS 콘솔을 엽니다.

2. 탐색 창에서 복제 인스턴스를 선택합니다.
3. 복제 인스턴스 생성을 선택하고 다음 정보를 입력합니다.

   - 이름에 원하는 이름을 입력합니다. 예: `docdb36todocdb40`.
   - 설명에 선택에 대한 설명을 입력합니다. listitem의 경우 Amazon DocumentDB 3.6에서 Amazon DocumentDB 4.0 복제 인스턴스입니다.
   - 인스턴스 클래스에서 필요에 따라 크기를 선택합니다.
   - 엔진 버전에서 `3.4.1.` 항목을 선택합니다.
   - Amazon VPC에서 원본 및 대상 Amazon DocumentDB 클러스터를 포함하는 Amazon VPC를 선택합니다.
   - 할당된 스토리지(GiB)에서 기본값인 50GiB를 사용합니다. 쓰기 처리량 워크로드가 높은 경우 워크로드에 맞게 이 값을 높입니다.
   - 다중 AZ의 경우 고가용성 및 장애 조치 지원이 필요한 경우 예를 선택합니다.
   - 퍼블릭 액세스 가능에서 이 옵션을 활성화합니다.



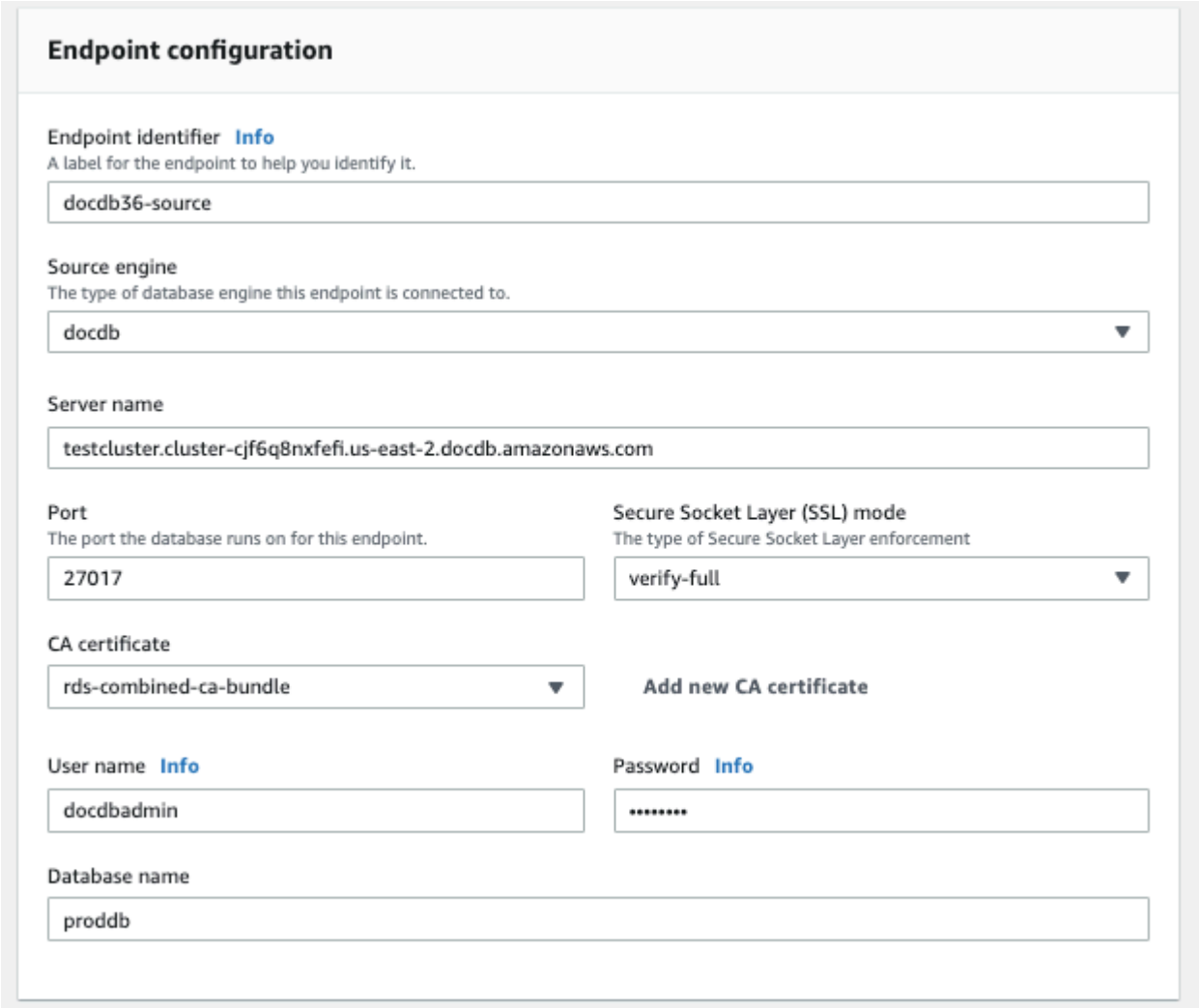
4. 복제 인스턴스 생성을 선택합니다.

5단계: AWS DMS 소스 엔드포인트 생성

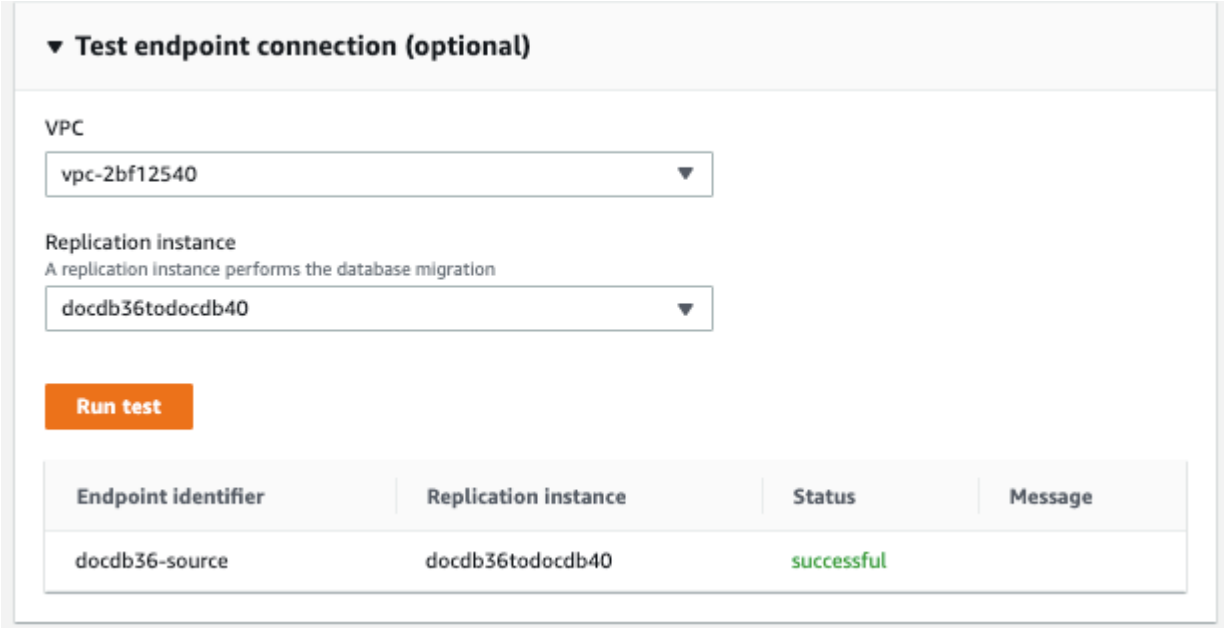
소스 엔드포인트는 소스 Amazon DocumentDB 클러스터에 사용됩니다.

소스 엔드포인트를 생성하려면

1. AWS DMS 콘솔을 엽니다.
2. 탐색 창에서 엔드포인트를 선택합니다.
3. `Create endpoint` 항목을 선택하고 다음 정보를 입력합니다.
   - 엔드포인트 유형에서 소스를 선택합니다.
   - >엔드포인트 식별자에 기억하기 쉬운 이름(예: `docdb-source`)을 입력합니다.
   - Source engine에서 `docdb` 항목을 선택합니다.
   - 서버 이름에 소스 Amazon DocumentDB 클러스터의 DNS 이름을 입력합니다.
   - 포트에 소스 Amazon DocumentDB 클러스터의 포트 번호를 입력합니다.
   - SSL 모드에서 `verify-full` 항목을 선택합니다.
   - CA 인증서의 경우 새 CA 인증서 추가를 선택합니다. 새 CA 인증서 를 다운로드하여 TLS 연결 번들을 생성합니다. 인증서 식별자에 `rds-combined-ca-bundle` 항목을 입력합니다. 인증서 파일 가져오기에서 파일 선택을 선택하고 이전에 다운로드한 `.pem` 파일로 이동합니다. 파일을 선택하고 엽니다. 인증서 가져오기를 선택한 다음 인증서 선택 드롭다운에서 `rds-combined-ca-bundle` 항목을 선택합니다.
   - 서버 이름에 원본 Amazon DocumentDB 클러스터의 기본 사용자 이름을 입력합니다.
   - 암호에 원본 Amazon DocumentDB 클러스터의 기본 사용자 암호를 입력합니다.
   - 데이터베이스 이름에 업그레이드하려는 데이터베이스 이름을 입력합니다.



4. 연결을 테스트하여 성공적으로 설정되었는지 확인합니다.



5. 엔드포인트 생성을 선택합니다.

> **Note**
>
> AWS DMS 는 한 번에 하나의 데이터베이스만 마이그레이션할 수 있습니다.

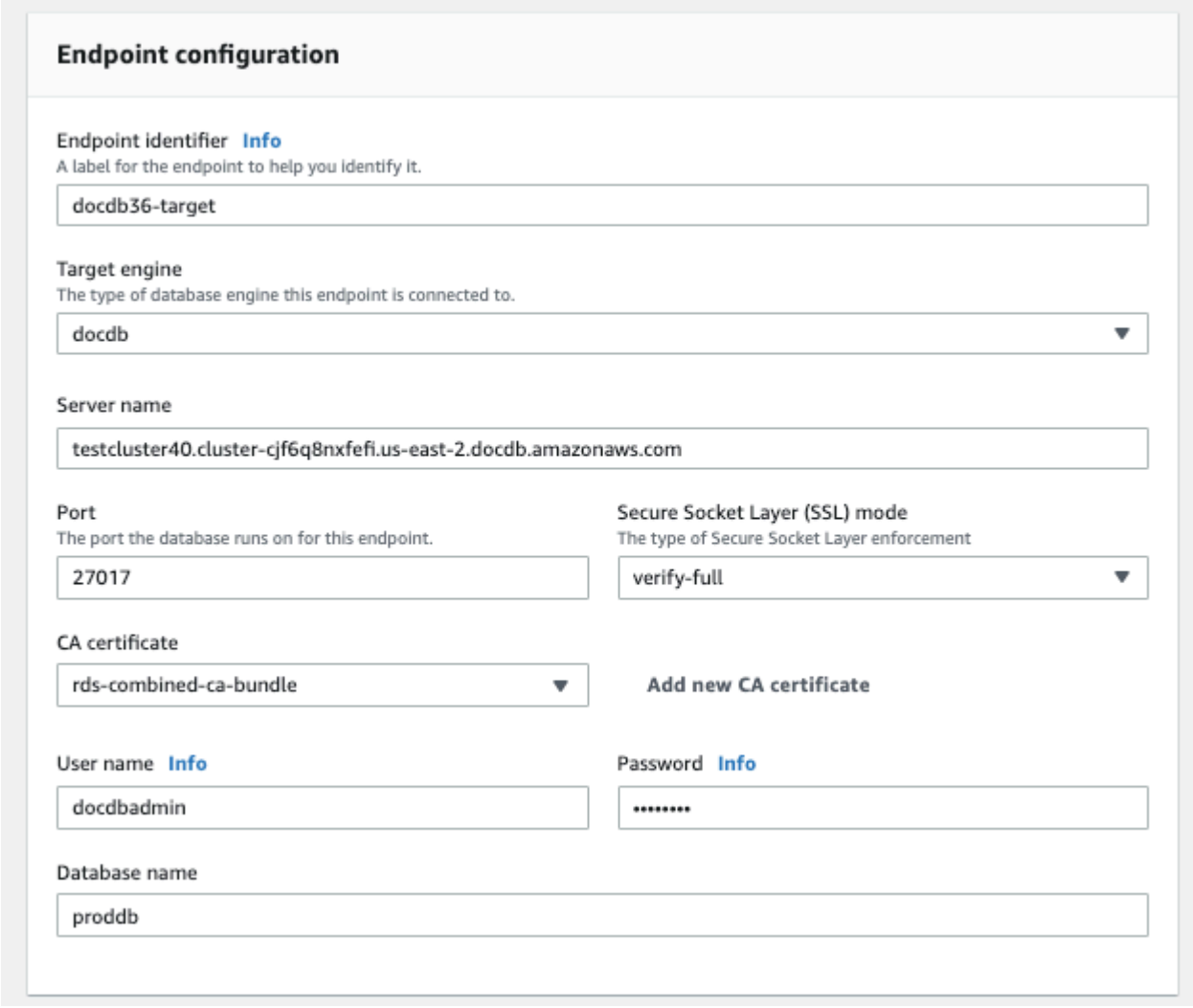
6단계: AWS DMS 대상 엔드포인트 생성

대상 엔드포인트는 대상 Amazon DocumentDB 클러스터에 사용됩니다.

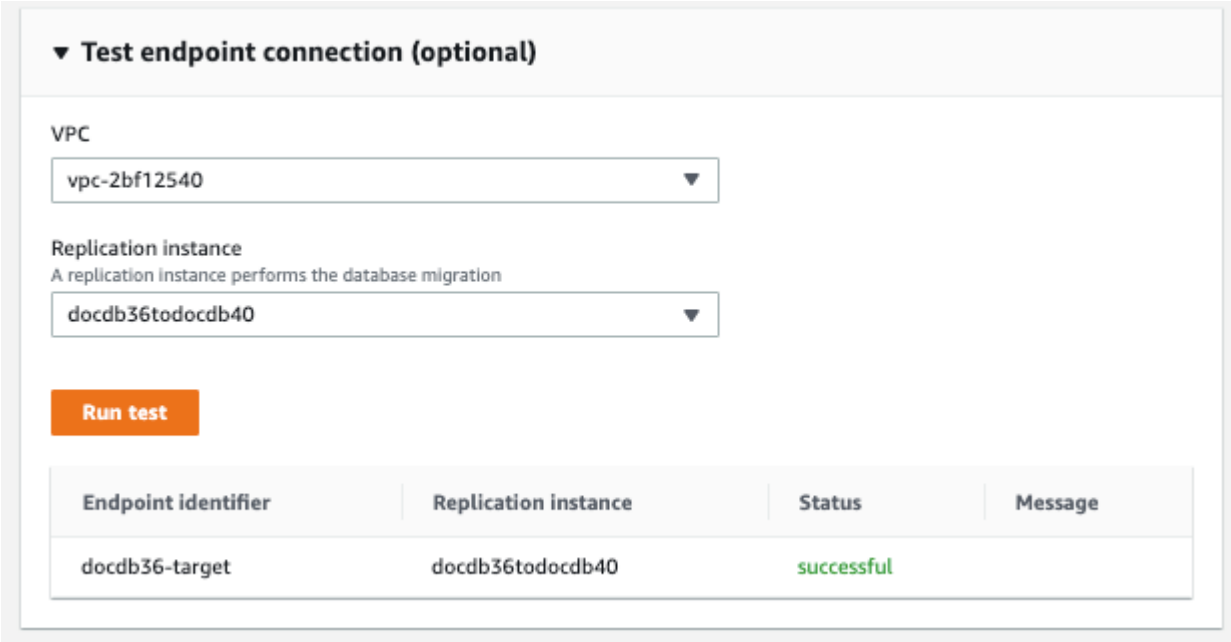
대상 엔드포인트를 생성하려면:

1. AWS DMS 콘솔을 엽니다.
2. 탐색 창에서 엔드포인트를 선택합니다.
3. 엔드포인트 생성을 선택하고 다음 정보를 입력합니다.
   - 엔드포인트 유형에서 대상을 선택합니다.
   - 엔드포인트 식별자에 기억하기 쉬운 이름(예: `docdb-target`)을 입력합니다.
   - Source engine에서 `docdb` 항목을 선택합니다.
   - 서버 이름에 대상 Amazon DocumentDB 클러스터의 DNS 이름을 입력합니다.
   - 포트에 대상 Amazon DocumentDB 클러스터의 포트 번호를 입력합니다.

- SSL 모드에서 `verify-full` 항목을 선택합니다.
- CA 인증서의 경우, 인증서 선택 드롭다운에서 기존 `rds-combined-ca-bundle` 인증서를 선택합니다.
- 사용자 이름에 대상 Amazon DocumentDB 클러스터의 기본 사용자 이름을 입력합니다.
- 암호에 대상 Amazon DocumentDB 클러스터의 기본 암호를 입력합니다.
- 데이터베이스 이름에 소스 엔드포인트를 설정하는 데 사용한 것과 동일한 데이터베이스 이름을 입력합니다.



4. 연결을 테스트하여 성공적으로 설정되었는지 확인합니다.



5. 엔드포인트 생성을 선택합니다.

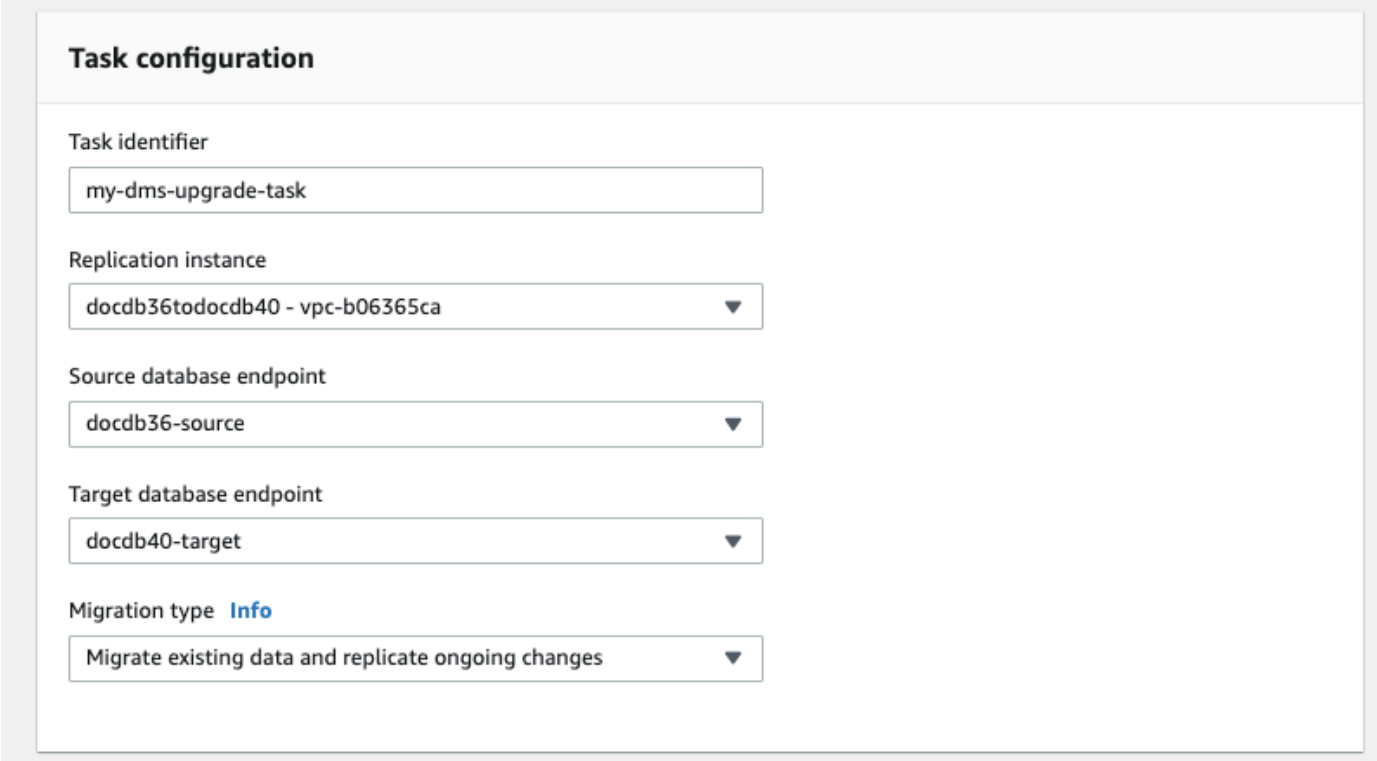
7단계: 마이그레이션 작업 생성 및 실행

AWS DMS 작업이 복제 인스턴스를 소스 및 대상 인스턴스와 바인딩합니다. 마이그레이션 작업을 생성할 때 소스 엔드포인트, 대상 엔드포인트, 복제 인스턴스 및 원하는 마이그레이션 설정을 지정합니다. AWS DMS 작업은 기존 데이터 마이그레이션, 기존 데이터 마이그레이션, 진행 중인 변경 사항 복제 또는 데이터 변경 사항 복제 등 세 가지 마이그레이션 유형으로 생성할 수 있습니다. 이 연습의 목적은 가동 중지 시간을 최소화하면서 Amazon DocumentDB 클러스터를 업그레이드하는 것이므로 이 단계에서는 옵션을 사용하여 기존 데이터를 마이그레이션하고 지속적인 변경 사항을 복제합니다. 이 옵션을 사용하면 기존 데이터를 마이그레이션하는 동안 변경 사항을 AWS DMS 캡처합니다.는 대량 데이터가 로드된 후에도 변경 사항을 AWS DMS 계속 캡처하고 적용합니다. 결국 원본과 대상 데이터베이스는 동기화되어 가동 중지가 최소화된 마이그레이션이 가능합니다.

다음은 가동 중지 시간 최소화 마이그레이션을 위한 마이그레이션 작업을 생성하는 단계입니다.

1. AWS DMS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스 마이그레이션 작업을 선택합니다.
3. 데이터베이스 마이그레이션 작업 생성을 선택하고 작업 구성 섹션에 다음 정보를 입력합니다.
   - 태스크 식별자에 기억하기 쉬운 이름(예: `my-dms-upgrade-task`)을 입력합니다.

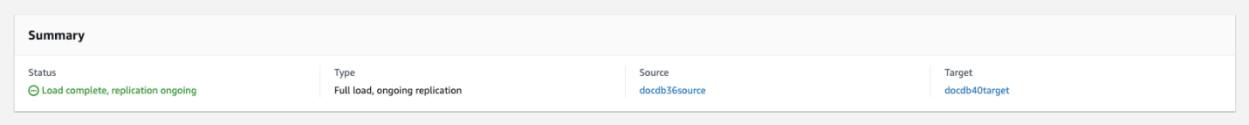
- 설명형 Amazon 리소스 이름(ARN)에 사용자 친화적 이름을 입력하여 기본 DMS ARN을 재정의합니다.
- 복제 인스턴스에서 4단계: AWS DMS 복제 인스턴스 생성에서 생성한 복제 인스턴스를 선택합니다.
- 소스 데이터베이스 엔드포인트에서에서 생성한 소스 엔드포인트를 선택합니다5단계: AWS DMS 소스 엔드포인트 생성.
- 대상 데이터베이스 엔드포인트에서에서 생성한 대상 엔드포인트를 선택합니다6단계: AWS DMS 대상 엔드포인트 생성.
- 마이그레이션 유형에서 마이그레이션 및 복제를 선택합니다.



4. 작업 설정 섹션에 다음 정보를 입력합니다.

- 대상 테이블 준비 모드 섹션에서 아무 작업도 수행하지 않음을 선택합니다. 이렇게 하면 3단계에서 생성된 인덱스가 삭제되지 않습니다.
- 작업 로그 하위 섹션에서 CloudWatch 로그 켜기를 선택합니다.
- 마이그레이션 작업 시작 구성에서 생성 시 자동을 선택합니다. 이를 통해 마이그레이션 작업이 생성되면 자동으로 시작됩니다.
- 데이터베이스 마이그레이션 작업 생성을 선택합니다.

AWS DMS 이제가 소스 Amazon DocumentDB 클러스터에서 대상 Amazon DocumentDB 클러스터로 데이터 마이그레이션을 시작합니다. 작업 상태가 시작 중에서 실행 중으로 바뀝니다. AWS DMS 콘솔에서 작업을 선택하여 진행 상황을 모니터링할 수 있습니다. 몇 분/몇 시간 후(마이그레이션 크기에 따라 다름) 상태가 로드 완료, 복제 진행 중으로 변경됩니다. 즉, AWS DMS 는 소스 Amazon DocumentDB 클러스터를 대상 Amazon DocumentDB 클러스터로 전체 로드 마이그레이션을 완료했으며 이제 변경 이벤트를 복제하고 있습니다.



결국 원본과 대상이 동기화됩니다. 컬렉션에서 `count()` 작업을 실행하여 모든 변경 이벤트가 마이그레이션되었는지 확인하여 동기화 여부를 확인할 수 있습니다.

8단계: 애플리케이션 엔드포인트를 대상 Amazon DocumentDB 클러스터로 변경

전체 로드가 완료되고 CDC 프로세스가 계속 복제되면 애플리케이션의 데이터베이스 연결 엔드포인트를 원본 Amazon DocumentDB 클러스터에서 대상 Amazon DocumentDB 클러스터로 변경할 준비가 된 것입니다.

Amazon DocumentDB의 보안

의 클라우드 보안 AWS 이 최우선 순위입니다. AWS 고객은 보안에 가장 민감한 조직의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 이점을 누릴 수 있습니다.

보안은 AWS 와 사용자 간의 공동 책임입니다. 이 설명서는 Amazon DocumentDB를 사용할 때 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 공동 책임 모델은 이 사항을 클라우드 내 보안 및 클라우드의 보안으로 설명합니다.

- 클라우드 보안 - AWS 는 AWS 클라우드에서 AWS 서비스를 실행하는 인프라를 보호할 책임이 있습니다.는 안전하게 사용할 수 있는 서비스 AWS 도 제공합니다. 서드 파티 감사원은 정기적으로 AWS 규정 준수 프로그램의 일환으로 보안 효과를 테스트하고 검증합니다. Amazon DocumentDB(MongoDB 호환)에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 규정 준수 프로그램에 대한 범위의AWS 서비스를 참조하십시오.
- 클라우드의 보안 - 사용자의 책임은 사용하는 AWS 서비스에 따라 결정됩니다. 또한 데이터의 민감도, 조직의 요건 및 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

Amazon DocumentDB는 연방정 위험 및 인증 관리 프로그램(FedRAMP)에 따라 권한이 부여됩니다. AWS GovCloud(미국) 리전에 대한 FedRAMP High 권한 부여와 AWS 미국 동부/서부 리전에 대한 FedRAMP Moderate 권한이 있습니다. AWS 및 규정 준수 노력에 대한 자세한 내용은 AWS 규정 준수 프로그램 제공 범위 내 서비스를 참조하세요.

> **Note**
>
> 이 장은 인스턴스 기반 클러스터와 탄력적 클러스터에 모두 적용됩니다. 자세한 내용은 아래 주제를 참조하십시오.

또한 Amazon DocumentDB 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법도 알아봅니다. 다음 주제에서는 보안 및 규정 준수 목표를 충족하도록 Amazon DocumentDB를 구성하는 방법을 보여 줍니다.

주제

- Amazon DocumentDB 및를 사용한 암호 관리 AWS Secrets Manager
- Amazon DocumentDB의 데이터 보호
- Amazon DocumentDB의 ID 및 액세스 관리

- IAM 자격 증명을 사용한 인증
- Amazon DocumentDB 사용자 관리
- 역할 기반 액세스 제어를 사용한 데이터베이스 액세스
- Amazon DocumentDB의 로깅 및 모니터링
- Amazon DocumentDB TLS 인증서 업데이트
- Amazon DocumentDB TLS 인증서 업데이트 — GovCloud
- Amazon DocumentDB에서의 규정 준수 검증
- Amazon DocumentDB의 복원성
- Amazon DocumentDB의 인프라 보안
- Amazon DocumentDB API 및 인터페이스 VPC 엔드포인트(AWS PrivateLink)
- Amazon DocumentDB에 대한 보안 모범 사례
- Amazon DocumentDB 이벤트 감사

Amazon DocumentDB 및를 사용한 암호 관리 AWS Secrets Manager

Amazon DocumentDB는 Secrets Manager와 통합되어 클러스터의 기본 사용자 암호를 관리합니다.

주제

- Amazon DocumentDB와의 Secrets Manager 통합에 대한 제한 사항
- 를 사용한 기본 사용자 암호 관리 개요 AWS Secrets Manager
- 에서 기본 사용자 암호의 Amazon DocumentDB 관리 적용 AWS Secrets Manager
- Secrets Manager를 사용하여 클러스터의 기본 사용자 암호 관리

Amazon DocumentDB와의 Secrets Manager 통합에 대한 제한 사항

Secrets Manager를 사용한 기본 사용자 암호 관리는 다음 기능에서 지원되지 않습니다.

- Amazon DocumentDB 글로벌 데이터베이스의 일부인 클러스터
- Amazon DocumentDB 리전 간 읽기 전용 복제본

를 사용한 기본 사용자 암호 관리 개요 AWS Secrets Manager

를 사용하면 데이터베이스 암호를 포함하여 코드의 하드 코딩된 보안 인증을 Secrets Manager에 대한 API 호출로 대체하여 프로그래밍 방식으로 보안 암호를 검색할 AWS Secrets Manager수 있습니다. Secrets Manager에 대한 자세한 내용은 AWS Secrets Manager 사용 설명서를 참조하세요.

Secrets Manager에 데이터베이스 보안 암호를 저장하면 AWS 계정에 요금이 발생합니다. 요금에 대한 자세한 정보는 AWS Secrets Manager 요금을 참조하세요.

다음 작업 중 하나를 수행할 때 Amazon DocumentDB가 Amazon DocumentDB 클러스터의 Secrets Manager에서 기본 사용자 암호를 관리하도록 지정할 수 있습니다.

- 클러스터 생성
- 클러스터 수정

Amazon DocumentDB가 Secrets Manager에서 기본 사용자 암호를 관리하도록 지정하면 Amazon DocumentDB가 암호를 생성하고 Secrets Manager에 저장합니다. 보안 암호와 직접 상호 작용하여 기본 사용자의 자격 증명을 검색할 수 있습니다. 고객 관리 키를 지정하여 암호를 암호화하거나 Secrets Manager에서 제공하는 KMS 키를 사용할 수도 있습니다.

Amazon DocumentDB는 보안 암호에 대한 설정을 관리하고 기본적으로 7일마다 보안 암호를 교체합니다. 교체 일정 같은 일부 설정을 수정할 수 있습니다. Secrets Manager에서 보안 암호를 관리하는 클러스터를 삭제하면 보안 암호와 관련 메타데이터도 삭제됩니다.

보안 암호의 자격 증명을 사용하여 클러스터에 연결하려면 Secrets Manager에서 보안 암호를 검색할 수 있습니다. 자세한 내용은 AWS Secrets Manager 사용 설명서의 에서 보안 암호 가져오기 AWS Secrets Manager 및 AWS Secrets Manager 보안 암호의 자격 증명과 함께 JDBC를 사용하여 SQL 데이터베이스에 연결을 참조하세요.

에서 기본 사용자 암호의 Amazon DocumentDB 관리 적용 AWS Secrets Manager

IAM 조건 키를 사용하여의 기본 사용자 암호에 대한 Amazon DocumentDB 관리를 적용할 수 있습니다 AWS Secrets Manager. 다음 정책은 Secrets Manager의 Amazon DocumentDB에서 기본 사용자 암호를 관리하지 않는 한 사용자가 인스턴스 또는 클러스터를 생성하거나 복원하도록 허용하지 않습니다.

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "rds:CreateDBCluster"
            ],
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "rds:ManageMasterUserPassword": false
                }
            }
        }
    ]
}
```

Secrets Manager를 사용하여 클러스터의 기본 사용자 암호 관리

다음 작업을 수행할 때 Secrets Manager에서 기본 사용자 암호의 Amazon DocumentDB 관리를 구성할 수 있습니다.

- Amazon DocumentDB 클러스터 생성
- Amazon DocumentDB 클러스터 수정

Amazon DocumentDB 콘솔 또는를 사용하여 이러한 작업을 수행할 수 AWS CLI 있습니다.

Amazon DocumentDB의 데이터 보호

AWS 공동 책임 모델 Amazon DocumentDB(MongoDB 호환)의 데이터 보호에 적용됩니다. 이 모델에 설명된 대로 AWS 는 모든를 실행하는 글로벌 인프라를 보호할 책임이 있습니다 AWS 클라우드. 사용자는 인프라에서 호스팅되는 콘텐츠를 관리해야 합니다. 사용하는 AWS 서비스 의 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.

데이터 보호를 위해 자격 증명을 보호하고 AWS 계정 AWS IAM Identity Center 또는 AWS Identity and Access Management (IAM)를 사용하여 개별 사용자를 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 통신합니다. TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다 AWS CloudTrail. CloudTrail 추적을 사용하여 AWS 활동을 캡처하는 방법에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail 추적 작업을 참조하세요.
- AWS 암호화 솔루션과 내부의 모든 기본 보안 제어를 사용합니다 AWS 서비스.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용하세요.
- 명령줄 인터페이스 또는 API를 AWS 통해에 액세스할 때 FIPS 140-3 검증 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 DocumentDB 또는 기타 AWS 서비스 에서 콘솔, API AWS CLI또는 AWS SDKs를 사용하여 작업하는 경우가 포함됩니다. 이름에 사용되는 태그 또는 자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 보안 인증 정보를 URL에 포함해서는 안 됩니다.

주제

- 클라이언트 측 필드 레벨 암호화
- Amazon DocumentDB 저장 데이터 암호화
- 전송 중 데이터 암호화
- 키 관리

클라이언트 측 필드 레벨 암호화

Amazon DocumentDB 클라이언트 측 필드 레벨 암호화(FLE)를 사용하면 Amazon DocumentDB 클러스터로 전송하기 전에 클라이언트 애플리케이션의 민감한 데이터를 암호화할 수 있습니다. 민감한 데이터는 클러스터에서 저장 및 처리될 때 암호화된 상태로 유지되며 검색 시 클라이언트 애플리케이션에서 복호화됩니다.

주제

- 시작

- 클라이언트 측 FLE의 쿼리
- 제한 사항

시작

Amazon DocumentDB에서 클라이언트측 FLE의 초기 구성은 암호화 키 생성, 애플리케이션에 역할 연결, 애플리케이션 구성, 암호화 옵션을 통한 CRUD 작업 정의 등을 포함하는 4단계 프로세스입니다.

주제

- 1단계: 암호화 키 생성하기
- 2단계: 역할을 애플리케이션에 연결하기
- 3단계: 애플리케이션 구성
- 4단계: CRUD 작업 정의하기
- 예: 클라이언트 측 필드 수준 암호화 구성 파일

1단계: 암호화 키 생성하기

를 사용하여 민감한 데이터 필드를 암호화 및 복호화하는 데 사용되는 대칭 키를 AWS Key Management Service생성하고 필요한 IAM 사용 권한을 제공합니다.는 데이터 키(DKs)를 암호화하는 데 사용되는 고객 키(CK)를 AWS KMS 저장합니다. 보안 태세를 강화하려면 고객 키를 KMS에 저장하는 것이 좋습니다. 데이터 키는 Amazon DocumentDB 컬렉션에 저장되는 보조 키로, 문서를 Amazon DocumentDB에 저장하기 전에 민감한 필드를 암호화하는 데 필요합니다. 고객 키는 데이터 키를 암호화하고, 데이터 키는 다시 데이터를 암호화하고 복호화합니다. 글로벌 클러스터를 사용하는 경우 여러 리전의 다양한 서비스 역할이 사용할 수 있는 다중 리전 키를 만들 수 있습니다.

키를 생성하는 방법을 포함하여 AWS Key Management Service에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서를 참조하세요.

2단계: 역할을 애플리케이션에 연결하기

적절한 AWS KMS 권한을 가진 IAM 정책을 생성합니다. 이 정책은 연결되는 IAM 자격 증명이 리소스 필드에 지정된 KMS 키를 암호화하고 해독하도록 허용합니다. 애플리케이션은 인증할이 IAM 역할을 수임합니다 AWS KMS.

정책은 다음과 비슷할 것입니다.

```
{ "Effect": "Allow",
"Action": ["kms:Decrypt", "kms:Encrypt"],
```

```
"Resource": "Customer Key ARN"
}
```

3단계: 애플리케이션 구성

이제에서 고객 키를 정의하고 IAM 역할을 AWS KMS 생성하여 고객 키에 액세스할 수 있는 적절한 IAM 권한을 제공했습니다. 필수 패키지를 가져옵니다.

```
import boto3
import json
import base64
from pymongo import MongoClient
from pymongo.encryption import (Algorithm,
                                ClientEncryption)
```

```
# create a session object:
my_session = boto3.session.Session()

# get access_key and secret_key programmatically using get_frozen_credentials() method:
 current_credentials = my_session.get_credentials().get_frozen_credentials()
```

1. KMS 공급자 유형으로 'aws'를 지정하고 이전 단계에서 검색한 계정 자격 증명을 입력합니다.

   ```
   provider = "aws"
   kms_providers = {
       provider: {
           "accessKeyId": current_credentials.access_key,
           "secretAccessKey": current_credentials.secret_key
       }
   }
   ```

2. 데이터 키를 암호화하는 데 사용되는 고객 키를 지정합니다.

   ```
   customer_key = {
   “region”: “AWS region of the customer_key”,
       “key”: “customer_key ARN”
   }

   key_vault_namespace = "encryption.dataKeys"

   key_alt_name = 'TEST_DATA_KEY'
   ```

3. MongoClient 개체를 구성합니다.

```
client = MongoClient(connection_string)

coll = client.test.coll
coll.drop()

client_encryption = ClientEncryption(
    kms_providers, # pass in the kms_providers variable from the previous step
    key_vault_namespace = key_vault_namespace,
    client,
    coll.codec_options
)
```

4. 데이터 키를 생성합니다.

```
data_key_id = client_encryption.create_data_key(provider,
    customer_key,
    key_alt_name = [key_alt_name])
```

5. 기존 데이터 키를 검색합니다.

```
data_key = DataKey("aws",
    master_key = customer_key)
key_id = data_key["_id"]
data_key_id = client[key_vault_namespace].find_one({"_id": key_id})
```

4단계: CRUD 작업 정의하기

암호화 옵션을 사용하여 CRUD 작업을 정의합니다.

1. 단일 문서를 작성/읽기/삭제할 컬렉션을 정의합니다.

```
coll = client.gameinfo.users
```

2. 명시적 암호화 - 필드를 암호화하고 다음을 삽입합니다.

> **Note**
>
> “key_id” 또는 “key_alt_name” 중 정확히 하나를 입력해야 합니다.

```
encrypted_first_name = client_encryption.encrypt(
    "Jane",
    Algorithm.AEAD_AES_256_CBC_HMAC_SHA_512_Deterministic,
    key_alt_name=data_key_id
)
encrypted_last_name = client_encryption.encrypt(
    "Doe",
    Algorithm.AEAD_AES_256_CBC_HMAC_SHA_512_Deterministic,
    key_alt_name=data_key_id
)
encrypted_dob = client_encryption.encrypt(
    "1990-01-01",
    Algorithm.AEAD_AES_256_CBC_HMAC_SHA_512_Random,
    key_alt_name=data_key_id
)

coll.insert_one(
    {"gamerTag": "jane_doe90",
    "firstName": encrypted_first_name,
    "lastName": encrypted_last_name,
    "dateOfBirth":encrypted_dob,
    "Favorite_games":["Halo","Age of Empires 2","Medal of Honor"]
})
```

예: 클라이언트 측 필드 수준 암호화 구성 파일

다음은 자신의 정보를 각각의 *### ## ## ###*로 변경하는 예제입니다.

```
# import python packages:
import boto3
import json
import base64
from pymongo import MongoClient
from pymongo.encryption import (Algorithm,
                                ClientEncryption)

def main():

    # create a session object:
    my_session = boto3.session.Session()
```

```
    # get aws_region from session object:
    aws_region = my_session.region_name

    # get access_key and secret_key programmatically using get_frozen_credentials()
 method:
    current_credentials = my_session.get_credentials().get_frozen_credentials()
    provider = "aws"

    # define the kms_providers which is later used to create the Data Key:
    kms_providers = {
        provider: {
            "accessKeyId": current_credentials.access_key,
            "secretAccessKey": current_credentials.secret_key
        }
    }

    # enter the kms key ARN. Replace the example ARN value.
    kms_arn = "arn:aws:kms:us-east-1:123456789:key/abcd-efgh-ijkl-mnop"
    customer_key = {
        "region": aws_region,
        "key":kms_arn
    }

    # secrets manager is used to strore and retrieve user credentials for connecting to
 an Amazon DocumentDB cluster.
    # retrieve the secret using the secret name. Replace the example secret key.
    secret_name = "/dev/secretKey"
    docdb_credentials = json.loads(my_session.client(service_name = 'secretsmanager',
 region_name = "us-east-1").get_secret_value(SecretId = secret_name)['SecretString'])

    connection_params = '/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false'
    conn_str = 'mongodb://' + docdb_credentials["username"] + ':' +
 docdb_credentials["password"] + '@' + docdb_credentials["host"] + ':' +
 str(docdb_credentials["port"]) + connection_params
    client = MongoClient(conn_str)

    coll = client.test.coll
    coll.drop()

    # store the encryption data keys in a key vault collection (having naming
 convention as db.collection):
    key_vault_namespace = "encryption.dataKeys"
```

```
    key_vault_db_name, key_vault_coll_name = key_vault_namespace.split(".", 1)

    # set up the key vault (key_vault_namespace) for this example:
    key_vault = client[key_vault_db_name][key_vault_coll_name]
    key_vault.drop()
    key_vault.create_index("keyAltNames", unique=True)

    client_encryption = ClientEncryption(
        kms_providers,
        key_vault_namespace,
        client,
        coll.codec_options)

    # create a new data key for the encrypted field:
    data_key_id = client_encryption.create_data_key(provider, master_key=customer_key,
key_alt_names=["some_key_alt_name"], key_material = None)

    # explicitly encrypt a field:
    encrypted_first_name = client_encryption.encrypt(
    "Jane",
    Algorithm.AEAD_AES_256_CBC_HMAC_SHA_512_Deterministic,
    key_id=data_key_id
    )
    coll.insert_one(
    {"gamerTag": "jane_doe90",
    "firstName": encrypted_first_name
    })
    doc = coll.find_one()
    print('Encrypted document: %s' % (doc,))

    # explicitly decrypt the field:
    doc["encryptedField"] = client_encryption.decrypt(doc["encryptedField"])
    print('Decrypted document: %s' % (doc,))

    # cleanup resources:
    client_encryption.close()
    client.close()

    if __name__ == "__main__":
        main()
```

클라이언트 측 FLE의 쿼리

Amazon DocumentDB는 클라이언트 측 FLE를 통한 포인트 동등 쿼리를 지원합니다. 불평등 및 비교 쿼리는 부정확한 결과를 반환할 수 있습니다. 암호 해독된 값에 대해 동일한 작업을 실행할 때와 비교할 때 읽기 및 쓰기 작업에서 예상치 못한 동작이 발생하거나 잘못된 동작이 발생할 수 있습니다.

예를 들어 게이머스코어가 500보다 큰 문서의 필터를 쿼리하려면

```
db.users.find( {
    "gamerscore" : { $gt : 500 }
})
```

클라이언트는 명시적 암호화 방법을 사용하여 쿼리 값을 암호화합니다.

```
encrypted_gamerscore_filter = client_encryption.encrypt(
    500,
        Algorithm.AEAD_AES_256_CBC_HMAC_SHA_512_Deterministic,
        key_alt_name=data_key_id
        )

db.users.find( {
    "gamerscore" : { $gt : encrypted_gamerscore_filter }
} )
```

찾기 작업에서 Amazon DocumentDB는 불평등보다 큰 값 검사를 사용하여 암호화된 값 500을 각 문서에 저장된 암호화된 필드 값과 비교합니다. 복호화된 데이터와 값을 사용하여 찾기 작업의 불평등 검사를 수행하면 결과가 성공적으로 생성되더라도 다른 결과가 반환될 수 있습니다.

제한 사항

Amazon DocumentDB 클라이언트 측 필드 레벨 암호화에 다음과 같은 제한 사항이 적용됩니다.

- Amazon DocumentDB는 포인트 동등 쿼리만 지원합니다. 불평등 및 비교 쿼리는 부정확한 결과를 반환할 수 있습니다. 암호 해독된 값에 대해 동일한 작업을 실행할 때와 비교할 때 읽기 및 쓰기 작업에서 예상치 못한 동작이 발생하거나 잘못된 동작이 발생할 수 있습니다. 게이머스코어가 500보다 큰 문서의 필터를 쿼리하기 위함입니다.

  ```
  db.users.find( {
      "gamerscore" : { $gt : 500 }
      })
  ```

클라이언트는 명시적 암호화 방법을 사용하여 쿼리 값을 암호화합니다.

```
encrypted_gamerscore_filter = client_encryption.encrypt(
    500,
    Algorithm.AEAD_AES_256_CBC_HMAC_SHA_512_Deterministic,
    key_alt_name=data_key_id
)

db.users.find({
    "gamerscore" : { $gt : encrypted_gamerscore_filter }
})
```

찾기 작업에서 Amazon DocumentDB는 불평등보다 큰 값 검사를 사용하여 암호화된 값 500을 각 문서에 저장된 암호화된 필드 값과 비교합니다. 복호화된 데이터와 값을 사용하여 찾기 작업의 불평등 검사를 수행하면 결과가 성공적으로 생성되더라도 다른 결과가 반환될 수 있습니다.

- Amazon DocumentDB는 몽고 쉘의 명시적인 클라이언트 측 FLE를 지원하지 않습니다. 하지만 이 기능은 지원되는 모든 드라이버에서 사용할 수 있습니다.

Amazon DocumentDB 저장 데이터 암호화

> **Note**
>
> AWS KMS 는 고객 마스터 키(CMK)라는 용어를 AWS KMS key 및 KMS 키로 대체합니다. 단, 개념은 바뀌지 않았습니다. 주요 변경 사항을 방지하기 위해 AWS KMS 는이 용어의 일부 변형을 유지합니다.

클러스터를 생성할 때 스토리지 암호화 옵션을 지정하여 Amazon DocumentDB 클러스터의 저장 데이터를 암호화합니다. 스토리지 암호화는 클러스터 전반에서 활성화되고 기본 인스턴스와 복제본을 포함한 모든 인스턴스에 적용됩니다. 또한 클러스터의 스토리지 볼륨, 데이터, 인덱스, 로그, 자동 백업 및 스냅샷에도 적용됩니다.

Amazon DocumentDB는 256비트 고급 암호화 표준(AES-256)을 사용하여 AWS Key Management Service ()에 저장된 암호화 키를 사용하여 데이터를 암호화합니다AWS KMS. 정지 상태의 암호화를 사용하도록 설정한 상태에서 Amazon DocumentDB 클러스터를 사용할 때는 응용프로그램 로직이나 클라이언트 연결을 수정할 필요가 없습니다. Amazon DocumentDB는 성능에 미치는 영향을 최소화하면서 데이터의 암호화 및 복호화를 투명하게 처리합니다.

Amazon DocumentDB는와 통합 AWS KMS 되며 봉투 암호화라는 메서드를 사용하여 데이터를 보호합니다. Amazon DocumentDB 클러스터가 로 암호화되면 AWS KMS Amazon DocumentDB는 KMS 키를 사용하여 스토리지 볼륨을 암호화하는 암호 텍스트 데이터 키를 생성 AWS KMS 하도록 요청합니다. 사이퍼텍스트 키는 사용자가 정의한 KMS 키를 사용하여 암호화되며, 암호화된 데이터 및 저장 메타데이터와 함께 저장됩니다. Amazon DocumentDB는 암호화된 데이터에 액세스해야 할 때 KMS 키를 사용하여 사이퍼텍스트 데이터 키를 복호화 AWS KMS 하도록 요청하고 일반 텍스트 데이터 키를 메모리에 캐시하여 스토리지 볼륨의 데이터를 효율적으로 암호화하고 복호화합니다.

Amazon DocumentDB의 스토리지 암호화 기능은 지원되는 모든 인스턴스 크기와 Amazon DocumentDB를 사용할 수 AWS 리전 있는 모든에서 사용할 수 있습니다.

Amazon DocumentDB 클러스터에 대해 중지 상태의 암호화 사용

또는 AWS Command Line Interface ()를 사용하여 클러스터를 프로비저닝할 때 Amazon DocumentDB 클러스터에서 저장 시 암호화를 활성화 AWS Management Console 하거나 비활성화할 수 있습니다 AWS CLI. 콘솔을 사용하여 만든 클러스터는 기본적으로 유휴 데이터 암호화가 활성화되어 있습니다. 를 사용하여 생성하는 클러스터 AWS CLI 에는 기본적으로 유휴 시 암호화가 비활성화되어 있습니다. 따라서 `--storage-encrypted` 파라미터를 사용하여 유휴 데이터 암호화를 명시적으로 활성화해야 합니다. 두 경우 모두 클러스터를 만든 후에는 유휴 데이터 암호화 옵션을 변경할 수 없습니다.

Amazon DocumentDB는 AWS KMS 를 사용하여 암호화 키를 검색 및 관리하고 이러한 키를 사용하는 방법을 제어하는 정책을 정의합니다. AWS KMS 키 식별자를 지정하지 않으면 Amazon DocumentDB는 기본 AWS 관리형 서비스 KMS 키를 사용합니다. Amazon DocumentDB는의 각 AWS 리전 에 대해 별도의 KMS 키를 생성합니다 AWS 계정. 자세한 내용은 AWS Key Management Service 개념을 참조하세요.

자체 KMS 키 생성을 시작하려면 AWS Key Management Service 개발자 안내서의 시작하기를 참조하세요.

> **Important**
>
> Amazon DocumentDB는 대칭 암호화 KMS 키만 지원하므로 클러스터를 암호화하려면 대칭 암호화 KMS 키를 사용해야 합니다. 비대칭 KMS 키를 사용하여 Amazon DocumentDB 클러스터의 데이터를 암호화하지 마세요. 자세한 내용은, AWS Key Management Service 개발자 가이드에서 비대칭 키 AWS KMS를 참조하세요 .

예를 들어, 키에 대한 액세스가 취소된 경우 Amazon DocumentDB가 더 이상 클러스터의 암호화 키에 액세스할 수 없는 경우 암호화된 클러스터는 터미널 상태가 됩니다. 이러한 경우에는 백업 파일에서만 클러스터를 복원할 수 있습니다. Amazon DocumentDB의 경우 항상 1일 동안 백업이 실행됩니다.

또한 암호화된 Amazon DocumentDB 클러스터에 대한 키를 비활성화하면 해당 클러스터에 대한 읽기 및 쓰기 액세스 권한이 손실됩니다. Amazon DocumentDB는 접근 권한이 없는 키로 암호화된 클러스터를 만나면 클러스터를 터미널 상태로 만듭니다. 이러한 상태에서는 클러스터를 더 이상 사용하지 못하기 때문에 데이터베이스의 현재 상태를 복구할 수 없습니다. 클러스터를 복원하려면 Amazon DocumentDB의 암호화 키에 대한 액세스를 다시 실행한 다음 백업에서 클러스터를 복원해야 합니다.

> **Important**
>
> 암호화된 클러스터의 KMS 키는 이미 만든 후에는 변경할 수 없습니다. 암호화된 클러스터를 생성하기 전에 암호화 키 요구 사항을 결정해야 합니다.

Using the AWS Management Console

클러스터를 생성할 때 유휴 데이터 암호화 옵션을 지정합니다. AWS Management Console을 사용하여 클러스터를 생성할 때 기본적으로 유휴 데이터 암호화가 활성화됩니다. 클러스터를 생성한 후에는 변경할 수 없습니다.

클러스터를 생성할 때 유휴 데이터 암호화 옵션을 지정하려면

1. 시작하기 섹션에 설명된 대로 Amazon DocumentDB 클러스터를 생성합니다 하지만 6단계에서 클러스터 생성을 선택하지 마세요.
2. 인증 섹션 아래에서 고급 설정 표시를 선택합니다.
3. 유휴 상태에서 암호화 섹션으로 아래로 스크롤합니다.
4. 유휴 상태 암호화에 사용할 옵션을 선택합니다. 어떤 옵션을 선택하든 클러스터를 생성한 후에는 변경할 수 없습니다.
   - 이 클러스터에서 저장 데이터를 암호화하려면 암호화 활성화를 선택합니다.
   - 이 클러스터에서 저장 데이터를 암호화하지 않으려면 암호화 비활성화를 선택합니다.
5. 원하는 프라이머리 키를 선택합니다. Amazon DocumentDB는 AWS Key Management Service (AWS KMS)를 사용하여 암호화 키를 검색 및 관리하고 이러한 키를 사용하는 방법을 제어하는 정책을 정의합니다. AWS KMS 키 식별자를 지정하지 않으면 Amazon DocumentDB는 기본 AWS 관리형 서비스 KMS 키를 사용합니다. 자세한 내용은 AWS Key Management Service 개념을 참조하세요.

> **Note**
>
> 암호화된 클러스터를 생성한 후에는 해당 클러스터의 KMS를 변경할 수 없습니다. 암호화된 클러스터를 생성하기 전에 암호화 키 요구 사항을 결정해야 합니다.

6. 필요에 따라 다른 섹션을 완료하고 클러스터를 생성합니다.

Using the AWS CLI

를 사용하여 Amazon DocumentDB 클러스터를 암호화하려면 `create-db-cluster` 명령을 AWS CLI실행하고 `--storage-encrypted` 옵션을 지정합니다. 를 사용하여 생성된 Amazon DocumentDB 클러스터는 기본적으로 스토리지 암호화를 활성화하지 AWS CLI 않습니다.

다음 예제에서는 스토리지 암호화를 사용하도록 설정된 Amazon DocumentDB 클러스터를 만듭니다.

다음 예제에서는 각 *### ## ## ##*자를 클러스터 정보로 바꿉니다.

Example

Linux, macOS, Unix의 경우:

```
aws docdb create-db-cluster \
  --db-cluster-identifier mydocdbcluster \
  --port 27017 \
  --engine docdb \
  --master-username SampleUser1 \
  --master-user-password primaryPassword \
  --storage-encrypted
```

Windows의 경우:

```
aws docdb create-db-cluster ^
  --db-cluster-identifier SampleUser1 ^
  --port 27017 ^
  --engine docdb ^
  --master-username SampleUser1 ^
  --master-user-password primaryPassword ^
  --storage-encrypted
```

암호화된 Amazon DocumentDB 클러스터를 생성할 때 다음 예제와 같이 AWS KMS 키 식별자를 지정할 수 있습니다.

Example

Linux, macOS, Unix의 경우:

```
aws docdb create-db-cluster \
  --db-cluster-identifier SampleUser1 \
  --port 27017 \
  --engine docdb \
  --master-username primaryUsername \
  --master-user-password yourPrimaryPassword \
  --storage-encrypted \
  --kms-key-id key-arn-or-alias
```

Windows의 경우:

```
aws docdb create-db-cluster ^
  --db-cluster-identifier SampleUser1 ^
  --port 27017 ^
  --engine docdb ^
  --master-username SampleUser1 ^
  --master-user-password primaryPassword ^
  --storage-encrypted ^
  --kms-key-id key-arn-or-alias
```

> **Note**
>
> 암호화된 클러스터를 생성한 후에는 해당 클러스터의 KMS를 변경할 수 없습니다. 암호화된 클러스터를 생성하기 전에 암호화 키 요구 사항을 결정해야 합니다.

Amazon DocumentDB 암호화된 클러스터의 제한 사항

Amazon DocumentDB 암호화된 클러스터에는 다음과 같은 제한이 있습니다.

- Amazon DocumentDB 클러스터에 대해 중지된 암호화는 클러스터가 생성된 후가 아니라 생성된 시점에만 사용 가능 또는 불가능으로 설정할 수 있습니다. 그러나 암호화되지 않은 클러스터의 스냅샷을 생성한 다음 암호화 중지 옵션 지정을 통해 암호화되지 않은 스냅샷을 새 클러스터로 복원하여 암호화되지 않은 클러스터의 암호화된 복사본을 생성할 수 있습니다.

자세한 정보는 다음의 주제를 참조하세요.

- 수동 클러스터 스냅샷 생성
- 클러스터 스냅샷에서 복원
- Amazon DocumentDB 클러스터 스냅샷 복사

- 스토리지 암호화를 사용하도록 설정된 Amazon DocumentDB 클러스터는 암호화를 사용하지 않도록 수정할 수 없습니다.
- Amazon DocumentDB 클러스터의 모든 인스턴스, 자동 백업, 스냅샷 및 인덱스는 동일한 KMS로 암호화됩니다.

전송 중 데이터 암호화

전송 계층 보안(TLS)을 사용하여 애플리케이션과 Amazon DocumentDB 클러스터 간의 연결을 암호화할 수 있습니다. 기본적으로 전송 중인 암호화는 새로 생성된 Amazon DocumentDB 클러스터에 대해 활성화됩니다. 클러스터를 생성할 때 또는 나중에 선택적으로 비활성화할 수 있습니다. 전송 중 데이터 암호화가 활성화된 경우 클러스터에 연결하려면 TLS를 사용하는 보안 연결이 필요합니다. TLS를 사용하여 Amazon DocumentDB에 연결하는 방법에 대한 자세한 내용은 Amazon DocumentDB에 프로그래밍 방식으로 연결를 참조하세요.

Amazon DocumentDB 클러스터 TLS 설정 관리

Amazon DocumentDB 클러스터에 대한 전송 중 데이터 암호화는 클러스터 파라미터 그룹의 TLS 파라미터를 통해 관리됩니다. AWS Management Console 또는 AWS Command Line Interface ()를 사용하여 Amazon DocumentDB 클러스터 TLS 설정을 관리할 수 있습니다AWS CLI. 현재 TLS 설정을 확인하고 수정하는 방법은 다음 섹션을 참조하세요.

Using the AWS Management Console

다음 단계에 따라 매개 변수 그룹 식별, TLS 값 확인, 필요한 수정 등 콘솔을 사용하여 TLS 암호화에 대한 관리 작업을 수행합니다.

> **Note**
>
> 클러스터를 생성할 때 다르게 지정하지 않으면 클러스터는 기본 클러스터 파라미터 그룹을 이용해 생성됩니다. `default` 클러스터 파라미터 그룹의 파라미터는 수정할 수 없습니다(예: `tls` 활성화/비활성화). 클러스터가 `default` 클러스터 파라미터 그룹을 사용하는 경우 해당 클러스터를 수정하여 기본이 아닌 클러스터 파라미터 그룹을 사용해야 합니다. 먼저

사용자 지정 클러스터 파라미터 그룹을 생성해야 할 수도 있습니다. 자세한 내용은 Amazon DocumentDB 클러스터 파라미터 그룹 생성 단원을 참조하십시오.

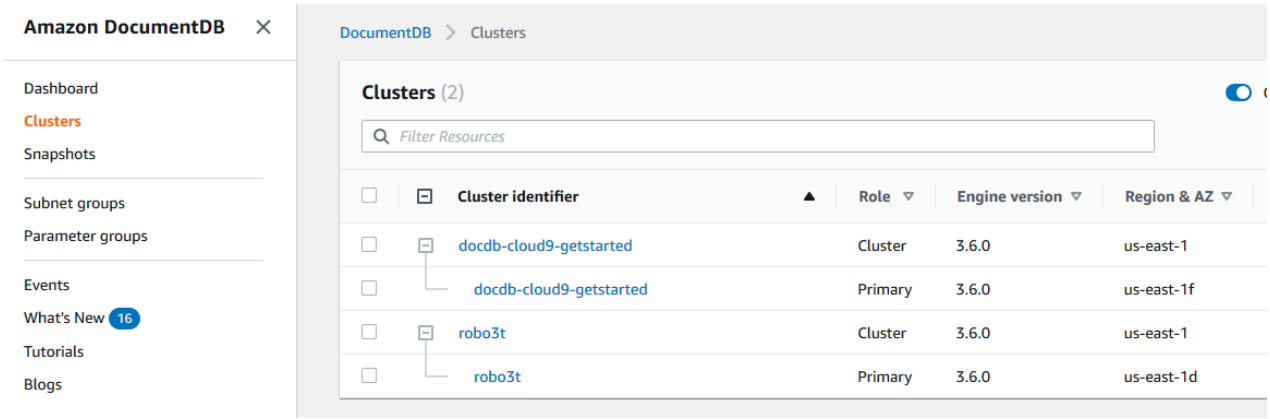
1. 클러스터에서 사용 중인 클러스터 매개 변수 그룹을 확인합니다.

   a. https://console.aws.amazon.com/docdb에서 Amazon DocumentDB 콘솔을 엽니다.

   b. 탐색 창에서 클러스터를 선택합니다.

   > **Tip**
   >
   > 화면 왼쪽에 탐색 창이 표시되지 않으면 페이지 왼쪽 상단 모서리에서 메뉴 아이콘
   > (≡)
   > 을 선택합니다.

   c. 참고로 클러스터 탐색 상자의 클러스터 식별자 열에는 클러스터와 인스턴스가 모두 표시됩니다. 인스턴스는 클러스터 아래에 나열됩니다. 참조는 아래 스크린샷을 참조하세요.



   d. 통합할 클러스터를 선택합니다.

   e. 설정 탭을 선택하여 클러스터 디테일 아래를 스크롤 다운하여 클러스터 파라미터 그룹을 위치시키세요. 클러스터 파라미터 그룹의 이름을 적어 둡니다.

      클러스터의 파라미터 그룹 이름이 `default`(예: `default.docdb3.6`)인 경우 사용자 지정 클러스터 파라미터 그룹을 생성해야 하며 계속하기 전에 이를 클러스터의 파라미터 그룹으로 지정해야 합니다. 자세한 내용은 다음을 참조하세요:

      1. Amazon DocumentDB 클러스터 파라미터 그룹 생성 — 사용할 수 있는 사용자 지정 클러스터 매개 변수 그룹이 없는 경우 생성합니다.

2. Amazon DocumentDB 클러스터 수정 — 사용자 지정 클러스터 매개 변수 그룹을 사용하도록 클러스터를 수정합니다.

2. **`tls`** 클러스터 파라미터의 현재 값을 확인합니다.

   a. https://console.aws.amazon.com/docdb에서 Amazon DocumentDB 콘솔로 이동합니다.

   b. 탐색 창에서 파라미터 그룹을 선택합니다.

   c. 클러스터 파라미터 그룹 목록에서 원하는 클러스터 파라미터 그룹의 이름을 선택합니다.

   d. 클러스터 파라미터 섹션을 찾습니다. 클러스터 파라미터 목록에서 `tls` 클러스터 파라미터 행을 찾습니다. 이때 다음 네 개의 열이 중요합니다:

      - 클러스터 파라미터 이름 — 클러스터 매개 변수의 이름입니다. TLS를 관리하는 경우 `tls` 클러스터 파라미터를 살펴보게 됩니다.
      - 값 — 각 클러스터 매개 변수의 현재 값입니다.
      - 허용된 값 — 클러스터 매개 변수에 적용할 수 있는 값 목록입니다.
      - 응용 유형 — 정적 또는 동적 타입. 정적 클러스터 파라미터에 대한 변경 사항은 인스턴스를 재부팅할 때에만 적용할 수 있습니다. 동적 클러스터 파라미터에 대한 변경 사항은 즉시 또는 인스턴스를 재부팅할 때 적용할 수 있습니다.

3. **`tls`** 클러스터 파라미터의 값을 수정합니다.

   `tls`의 값이 잘못된 경우 이 클러스터 파라미터 그룹에 대한 값을 수정합니다. `tls` 클러스터 파라미터의 값을 변경하려면, 다음 단계에 따라 이전 섹션으로부터 계속합니다.

   a. 클러스터 파라미터 이름의 왼쪽에 있는 버튼(`tls`)을 선택합니다.

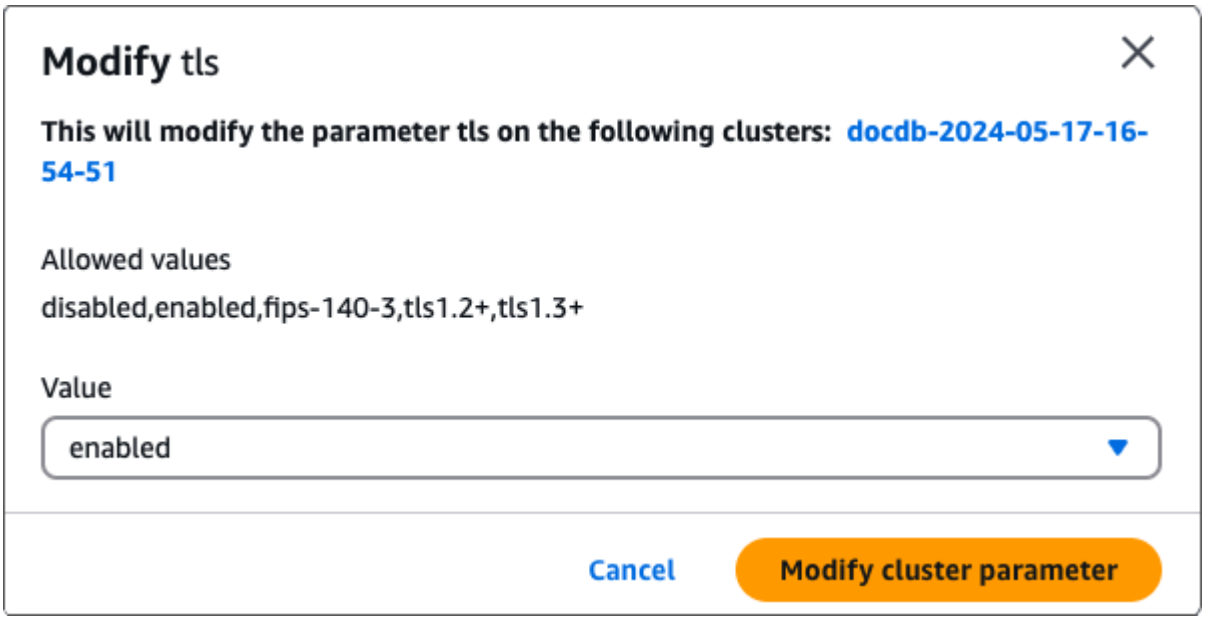
   b. 편집을 선택합니다.

   c. `tls`의 값을 변경하려면 **`tls`** 수정 대화 상자의 드롭다운 목록에서 클러스터 파라미터로 사용할 값을 선택합니다.

      유효한 값은 다음과 같습니다:

      - 비활성화 — TLS 비활성화
      - enabled - TLS 버전 1.0~1.3을 활성화합니다.
      - fips-140-3 — FIPS를 사용하여 TLS를 활성화합니다. 클러스터는 연방 정보 처리 표준(FIPS) 간행물 140-3의 요구 사항에 따라 보안 연결만 허용합니다. 이는 다음 지역의 Amazon DocumentDB 5.0(엔진 버전 3.0.3727) 클러스터에서만 지원됩니다: ca-central-1, us-west-2, us-east-1, us-east-2, us-gov-east-1, us-gov-west-1.

- tls1.2+ - TLS 버전 1.2 이상을 활성화합니다. 이는 Amazon DocumentDB 4.0(엔진 버전 2.0.10980) 및 Amazon DocumentDB(엔진 버전 3.0.11051)부터만 지원됩니다.
- tls1.3+ - TLS 버전 1.3 이상을 활성화합니다. 이는 Amazon DocumentDB 4.0(엔진 버전 2.0.10980) 및 Amazon DocumentDB(엔진 버전 3.0.11051)부터만 지원됩니다.



d. 클러스터 파라미터 수정을 선택합니다. 재부팅할 때 변경 사항이 각 클러스터 인스턴스에 적용됩니다.

4. Amazon DocumentDB 인스턴스를 재부팅합니다.

   클러스터의 각 인스턴스를 재부팅하여 변경 사항이 클러스터의 모든 인스턴스에 적용되도록 합니다.

   a. https://console.aws.amazon.com/docdb에서 Amazon DocumentDB 콘솔로 이동합니다.
   b. 탐색 창에서 인스턴스를 선택합니다.
   c. 재부팅할 인스턴스를 지정하려면 인스턴스 목록에서 인스턴스를 찾은 다음 이름 왼쪽에 있는 버튼을 선택합니다.
   d. 작업을 선택한 후 재부팅을 선택합니다. 재부팅을 선택하여 재부팅할지를 확인합니다.

Using the AWS CLI

파라미터 그룹 식별, TLS 값 확인, 필요한 수정 등 AWS CLI을 사용하여 TLS 암호화에 대한 관리 작업을 수행하려면 다음 단계를 수행합니다.

> **Note**
>
> 클러스터를 생성할 때 다르게 지정하지 않으면 클러스터는 기본 클러스터 파라미터 그룹을 이용해 생성됩니다. `default` 클러스터 파라미터 그룹의 파라미터는 수정할 수 없습니다(예: `tls` 활성화/비활성화). 클러스터가 `default` 클러스터 파라미터 그룹을 사용하는 경우 해당 클러스터를 수정하여 기본이 아닌 클러스터 파라미터 그룹을 사용해야 합니다. 먼저 사용자 지정 클러스터 파라미터 그룹을 생성해야 할 수도 있습니다. 자세한 내용은 Amazon DocumentDB 클러스터 파라미터 그룹 생성 단원을 참조하십시오.

1. 클러스터에서 사용 중인 클러스터 파라미터 그룹 확인.

   다음 옵션을 사용하여 `describe-db-clusters` 명령을 실행합니다.

   - `--db-cluster-identifier`
   - `--query`

   다음 예제에서는 각 *### ## ## ##*자를 클러스터 정보로 바꿉니다.

   ```
   aws docdb describe-db-clusters \
     --db-cluster-identifier mydocdbcluster \
     --query 'DBClusters[*].[DBClusterIdentifier,DBClusterParameterGroup]'
   ```

   이 작업의 출력은 다음과 같습니다(JSON 형식).

   ```
   [
       [
           "mydocdbcluster",
           "myparametergroup"
       ]
   ]
   ```

   클러스터의 파라미터 그룹 이름이 `default`(예: `default.docdb3.6`)인 경우 사용자 지정 클러스터 파라미터 그룹이 있어야 하며 계속하기 전에 이를 클러스터의 파라미터 그룹으로 지정해야 합니다. 자세한 정보는 다음 주제를 참조하세요:

   1. Amazon DocumentDB 클러스터 파라미터 그룹 생성 — 사용할 수 있는 사용자 지정 클러스터 파라미터 그룹이 없다면 만들어야 합니다.

2. Amazon DocumentDB 클러스터 수정 — 사용자 지정 클러스터 매개 변수 그룹을 사용하도록 클러스터를 수정합니다.

2. **`tls`** 클러스터 파라미터의 현재 값을 확인합니다.

   이 클러스터 파라미터 그룹에 대한 자세한 내용을 보려면 다음 옵션을 사용하여 `describe-db-cluster-parameters` 명령을 실행합니다.

   - `--db-cluster-parameter-group-name`
   - `--query`

     출력을 `ParameterName`, `ParameterValueAllowedValues`, 및 관심 필드로만 제한합니다`ApplyType`.

   다음 예제에서는 각 *`### ## ## ##`*자를 클러스터 정보로 바꿉니다.

```
aws docdb describe-db-cluster-parameters \
  --db-cluster-parameter-group-name myparametergroup \
  --query 'Parameters[*].[ParameterName,ParameterValue,AllowedValues,ApplyType]'
```

   이 작업의 출력은 다음과 같습니다(JSON 형식).

```
[
    [
        "audit_logs",
        "disabled",
        "enabled,disabled",
        "dynamic"
    ],
    [
        "tls",
        "disabled",
        "disabled,enabled,fips-140-3,tls1.2+,tls1.3+",
        "static"
    ],
    [
        "ttl_monitor",
        "enabled",
        "disabled,enabled",
        "dynamic"
    ]
```

```
]
```

3. **tls** 클러스터 파라미터의 값을 수정합니다.

   tls의 값이 잘못된 경우 이 클러스터 파라미터 그룹에 대한 값을 수정합니다. tls 클러스터 파라미터의 값을 변경하려면 다음 옵션을 사용하여 modify-db-cluster-parameter-group 명령을 실행합니다.

   - --db-cluster-parameter-group-name — 선택 사항. 수정할 클러스터 파라미터 그룹의 이름입니다. default.* 클러스터 파라미터 그룹은 지정할 수 없습니다.
   - --parameters - 필수입니다. 클러스터 파라미터 그룹에서 수정할 파라미터의 목록입니다.
     - ParameterName - 필수입니다. 수정할 클러스터 파라미터의 이름입니다.
     - ParameterValue - 필수입니다. 이 클러스터 파라미터의 새 값입니다. 클러스터 파라미터의 AllowedValues 중 하나이어야 합니다.
       - enabled - 클러스터는 TLS 버전 1.0~1.3을 사용한 보안 연결을 허용합니다.
       - disabled — 클러스터는 TLS를 사용한 보안 연결을 허용하지 않습니다.
       - fips-140-3 — 클러스터는 연방 정보 처리 표준(FIPS) 간행물 140-3의 요구 사항에 따라 보안 연결만 허용합니다. 이는 다음 지역의 Amazon DocumentDB 5.0(엔진 버전 3.0.3727) 클러스터(ca-central-1, us-west-2, us-east-1, us-east-2, us-gov-east-1, us-gov-west-1)에서만 지원됩니다.
       - tls1.2+ - 클러스터는 TLS 버전 1.2 이상을 사용한 보안 연결을 허용합니다. 이는 Amazon DocumentDB 4.0(엔진 버전 2.0.10980) 및 Amazon DocumentDB 5.0(엔진 버전 3.0.11051)부터만 지원됩니다.
       - tls1.3+ - 클러스터는 TLS 버전 1.3 이상을 사용한 보안 연결을 허용합니다. 이는 Amazon DocumentDB 4.0(엔진 버전 2.0.10980) 및 Amazon DocumentDB 5.0(엔진 버전 3.0.11051)부터만 지원됩니다.
     - ApplyMethod — 이 수정이 적용되지 않는 경우. tle 같은 정적 클러스터 파라미터의 경우 이 값이 pending-reboot이어야 합니다.
       - pending-reboot — 인스턴스가 재부팅된 후에만 변경 사항이 인스턴스에 적용됩니다. 이 변경 사항이 클러스터의 모든 인스턴스에 적용되려면 각 클러스터 인스턴스를 개별적으로 재부팅해야 합니다.

   다음 예제에서 각 ### ## ## ##자를 클러스터 정보로 바꿉니다.

   다음 코드는 를 비활성화하여 각 인스턴스가 재부팅될 때 변경 사항을 tls적용합니다.

```
aws docdb modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=tls,ParameterValue=disabled,ApplyMethod=pending-
reboot"
```

다음 코드는 tls (버전 1.0~1.3)가 각 인스턴스가 재부팅될 때 변경 사항을 적용할 수 있도록 합니다.

```
aws docdb modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=tls,ParameterValue=enabled,ApplyMethod=pending-
reboot"
```

다음 코드는를 사용하여 TLS를 활성화하여 각 인스턴스가 재부팅될 때 변경 사항을 fips-140-3적용합니다.

```
aws docdb modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup2 \
  --parameters "ParameterName=tls,ParameterValue=fips-140-3,ApplyMethod=pending-
reboot"
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
    "DBClusterParameterGroupName": "myparametergroup"
}
```

4. Amazon DocumentDB 인스턴스를 재부팅합니다.

   클러스터의 각 인스턴스를 재부팅하여 변경 사항이 클러스터의 모든 인스턴스에 적용되도록 합니다. Amazon DocumentDB 인스턴스를 재부팅하려면 다음 옵션을 사용하여 reboot-db-instance 명령을 실행합니다.

   - --db-instance-identifier

   다음 코드는 mydocdbinstance 인스턴스를 재부팅합니다.

   다음 예제에서는 각 ### ## ## ##자를 클러스터 정보로 바꿉니다.

Example

Linux, macOS, Unix의 경우:

```
aws docdb reboot-db-instance \
  --db-instance-identifier mydocdbinstance
```

Windows의 경우:

```
aws docdb reboot-db-instance ^
  --db-instance-identifier mydocdbinstance
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
    "DBInstance": {
        "AutoMinorVersionUpgrade": true,
        "PubliclyAccessible": false,
        "PreferredMaintenanceWindow": "fri:09:32-fri:10:02",
        "PendingModifiedValues": {},
        "DBInstanceStatus": "rebooting",
        "DBSubnetGroup": {
            "Subnets": [
                {
                    "SubnetStatus": "Active",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1a"
                    },
                    "SubnetIdentifier": "subnet-4e26d263"
                },
                {
                    "SubnetStatus": "Active",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1c"
                    },
                    "SubnetIdentifier": "subnet-afc329f4"
                },
                {
                    "SubnetStatus": "Active",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1e"
```

```
                    },
                    "SubnetIdentifier": "subnet-b3806e8f"
                },
                {
                    "SubnetStatus": "Active",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1d"
                    },
                    "SubnetIdentifier": "subnet-53ab3636"
                },
                {
                    "SubnetStatus": "Active",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1b"
                    },
                    "SubnetIdentifier": "subnet-991cb8d0"
                },
                {
                    "SubnetStatus": "Active",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1f"
                    },
                    "SubnetIdentifier": "subnet-29ab1025"
                }
            ],
            "SubnetGroupStatus": "Complete",
            "DBSubnetGroupDescription": "default",
            "VpcId": "vpc-91280df6",
            "DBSubnetGroupName": "default"
        },
        "PromotionTier": 2,
        "DBInstanceClass": "db.r5.4xlarge",
        "InstanceCreateTime": "2018-11-05T23:10:49.905Z",
        "PreferredBackupWindow": "00:00-00:30",
        "KmsKeyId": "arn:aws:kms:us-east-1:012345678901:key/0961325d-a50b-44d4-
b6a0-a177d5ff730b",
        "StorageEncrypted": true,
        "VpcSecurityGroups": [
            {
                "Status": "active",
                "VpcSecurityGroupId": "sg-77186e0d"
            }
        ],
        "EngineVersion": "3.6.0",
```

```
        "DbiResourceId": "db-SAMPLERESOURCEID",
        "DBInstanceIdentifier": "mydocdbinstance",
        "Engine": "docdb",
        "AvailabilityZone": "us-east-1a",
        "DBInstanceArn": "arn:aws:rds:us-east-1:012345678901:db:sample-cluster-
instance-00",
        "BackupRetentionPeriod": 1,
        "Endpoint": {
            "Address": "mydocdbinstance.corcjozrlsfc.us-
east-1.docdb.amazonaws.com",
            "Port": 27017,
            "HostedZoneId": "Z2R2ITUGPM61AM"
        },
        "DBClusterIdentifier": "mydocdbcluster"
    }
}
```

인스턴스가 재부팅되는 데 몇 분 정도 걸릴 수 있습니다. 사용 가능 상태인 경우에만 인스턴스를 사용할 수 있습니다. 콘솔 또는 AWS CLI를 사용하여 인스턴스의 상태를 모니터링할 수 있습니다. 자세한 내용은 Amazon DocumentDB 인스턴스 상태 모니터링 단원을 참조하십시오.

키 관리

Amazon DocumentDB는 AWS Key Management Service (AWS KMS)를 사용하여 암호화 키를 검색하고 관리합니다. AWS KMS 는 안전하고 가용성이 높은 하드웨어와 소프트웨어를 결합하여 클라우드에 맞게 조정된 키 관리 시스템을 제공합니다. 를 사용하면 암호화 키를 생성하고 이러한 키를 사용할 AWS KMS수 있는 방법을 제어하는 정책을 정의할 수 있습니다.는를 AWS KMS 지원 AWS CloudTrail 하므로 키 사용량을 감사하여 키가 적절하게 사용되고 있는지 확인할 수 있습니다.

AWS KMS 키를 Amazon DocumentDB 및 Amazon Simple Storage Service(Amazon S3), Amazon Relational Database Service(RDS), Amazon Elastic Block Store(Amazon EBS) 및 Amazon Redshift 와 같은 지원되는 AWS 서비스와 함께 사용할 수 있습니다. 가 지원하는 서비스 목록은 AWS Key Management Service 개발자 안내서의 AWS 서비스 사용 방법을 AWS KMS AWS KMS참조하세요. 에 대한 자세한 내용은 란 무엇입니까 AWS Key Management Service?를 AWS KMS참조하세요.

Amazon DocumentDB의 ID 및 액세스 관리

AWS Identity and Access Management (IAM)는 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어할 수 AWS 서비스 있도록 도와주는 입니다. IAM 관리자는 어떤 사용자가 Amazon DocumentDB 리소스를 사용할 수 있도록 인증(로그인)되고 권한 부여(권한 있음)될 수 있는지 제어합니다. IAM은 추가 비용 없이 사용할 수 AWS 서비스 있는 입니다.

주제

- 대상
- ID를 통한 인증
- 정책을 사용하여 액세스 관리
- Amazon DocumentDB에서 IAM을 사용하는 방법
- Amazon DocumentDB의 ID 기반 정책 예
- Amazon DocumentDB ID 및 액세스 문제 해결
- Amazon DocumentDB 리소스에 대한 액세스 권한 관리
- Amazon DocumentDB에 대한 ID 기반 정책(IAM 정책) 사용
- AWS Amazon DocumentDB에 대한 관리형 정책
- Amazon DocumentDB API 권한: 작업, 리소스 및 조건 참조

대상

사용 방법 AWS Identity and Access Management (IAM)은 Amazon DocumentDB에서 수행하는 작업에 따라 다릅니다.

서비스 사용자 – Amazon DocumentDB 서비스를 사용하여 작업을 수행하는 경우 필요한 보안 인증과 권한을 관리자가 제공합니다. 다른 Amazon DocumentDB 기능을 사용하여 작업을 수행한다면 추가 권한이 필요할 수 있습니다. 액세스 권한 관리 방법을 이해하면 관리자에게 올바른 권한을 요청하는 데 도움이 됩니다. Amazon DocumentDB의 기능에 액세스할 수 없다면 Amazon DocumentDB ID 및 액세스 문제 해결 단원을 참조하십시오.

서비스 관리자 – 회사에서 Amazon DocumentDB 리소스를 책임지고 있다면 Amazon DocumentDB에 대한 완전한 액세스 권한이 있을 것입니다. 서비스 관리자는 서비스 사용자가 액세스해야 하는 Amazon DocumentDB 기능과 리소스를 결정합니다. 그런 다음 IAM 관리자에게 요청을 제출하여 서비스 사용자의 권한을 변경해야 합니다. 이 페이지의 정보를 검토하여 IAM의 기본 개념을 이해하세요. 회사가 Amazon DocumentDB에서 IAM을 사용하는 방법에 대해 자세히 알아보려면 Amazon DocumentDB에서 IAM을 사용하는 방법 단원을 참조하십시오.

IAM 관리자 - IAM 관리자라면 Amazon DocumentDB에 대한 액세스 관리 정책 작성 방법을 자세히 알고 싶을 수도 있습니다. IAM에서 사용할 수 있는 Amazon DocumentDB ID 기반 정책의 예를 보려면 Amazon DocumentDB의 ID 기반 정책 예 단원을 참조하십시오.

ID를 통한 인증

인증은 AWS 자격 증명으로에 로그인하는 방법입니다. IAM 사용자 또는 AWS 계정 루트 사용자 IAM 역할을 수임하여 로 인증(로그인 AWS)되어야 합니다.

자격 증명 소스를 통해 제공된 자격 증명을 사용하여 페더레이션 자격 증명 AWS 으로에 로그인할 수 있습니다. AWS IAM Identity Center (IAM Identity Center) 사용자, 회사의 Single Sign-On 인증 및 Google 또는 Facebook 자격 증명은 페더레이션 자격 증명의 예입니다. 페더레이션형 ID로 로그인할 때 관리자가 이전에 IAM 역할을 사용하여 ID 페더레이션을 설정했습니다. 페더레이션을 사용하여 AWS 에 액세스하면 간접적으로 역할을 수임하게 됩니다.

사용자 유형에 따라 AWS Management Console 또는 AWS 액세스 포털에 로그인할 수 있습니다. 에 로그인하는 방법에 대한 자세한 내용은 AWS 로그인 사용 설명서의에 로그인하는 방법을 AWS참조하세요. AWS 계정

AWS 프로그래밍 방식으로에 액세스하는 경우는 자격 증명을 사용하여 요청에 암호화 방식으로 서명할 수 있는 소프트웨어 개발 키트(SDK)와 명령줄 인터페이스(CLI)를 AWS 제공합니다. AWS 도구를 사용하지 않는 경우 직접 요청에 서명해야 합니다. 권장 방법을 사용하여 요청에 직접 서명하는 자세한 방법은 IAM 사용 설명서에서 API 요청용AWS Signature Version 4를 참조하세요.

사용하는 인증 방법에 상관없이 추가 보안 정보를 제공해야 할 수도 있습니다. 예를 들어는 다중 인증(MFA)을 사용하여 계정의 보안을 강화할 것을 AWS 권장합니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서에서 다중 인증 및 IAM 사용 설명서에서 IAM의AWS 다중 인증을 참조하세요.

AWS 계정 루트 사용자

를 생성할 때 계정의 모든 AWS 서비스 및 리소스에 대한 완전한 액세스 권한이 있는 하나의 로그인 자격 증명으로 AWS 계정시작합니다. 이 자격 증명을 테 AWS 계정 루트 사용자라고 하며 계정을 생성하는 데 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 인간 사용자가 자격 증명 공급자와의 페더레이션을 사용하여 임시 자격 증명을 사용하여 AWS 서비스 에 액세스하도록 요구하는 것입니다.

페더레이션 자격 증명은 엔터프라이즈 사용자 디렉터리, 웹 자격 증명 공급자, AWS Directory Service, Identity Center 디렉터리 또는 자격 증명 소스를 통해 제공된 자격 증명을 사용하여 AWS 서비스 에 액세스하는 모든 사용자의 사용자입니다. 페더레이션 자격 증명에 액세스할 때 역할을 AWS 계정수임하고 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 및 애플리케이션에서 사용할 수 있도록 자체 자격 증명 소스의 사용자 AWS 계정 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 한 사람 또는 애플리케이션에 대한 특정 권한이 AWS 계정 있는 내 자격 증명입니다. 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 역할

IAM 역할은 특정 권한이 AWS 계정 있는 내 자격 증명입니다. IAM 사용자와 유사하지만, 특정 개인과 연결되지 않습니다. 에서 IAM 역할을 일시적으로 수임하려면 사용자에서 IAM 역할(콘솔)로 전환할 AWS Management Console수 있습니다. https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-console.html 또는 AWS API 작업을 호출하거나 사용자 지정 URL을 AWS

CLI 사용하여 역할을 수임할 수 있습니다. 역할 사용 방법에 대한 자세한 내용은 IAM 사용 설명서의 역할 수임 방법을 참조하세요.

임시 보안 인증이 있는 IAM 역할은 다음과 같은 상황에서 유용합니다.

- 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.
- 임시 IAM 사용자 권한 - IAM 사용자 또는 역할은 IAM 역할을 수임하여 특정 작업에 대한 다양한 권한을 임시로 받을 수 있습니다.
- 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 에서는 (역할을 프록시로 사용하는 대신) 정책을 리소스에 직접 연결할 AWS 서비스 수 있습니다. 교차 계정 액세스에 대한 역할과 리소스 기반 정책의 차이점을 알아보려면 IAM 사용 설명서의 IAM의 교차 계정 리소스 액세스를 참조하세요.
- 교차 서비스 액세스 - 일부는 다른에서 기능을 AWS 서비스 사용합니다 AWS 서비스. 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(FAS) - IAM 사용자 또는 역할을 사용하여에서 작업을 수행하는 경우 AWS보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우, 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는를 호출하는 보안 주체의 권한을 다운스트림 서비스에 AWS 서비스 대한 요청과 AWS 서비스함께 사용합니다. FAS 요청은 서비스가 다른 AWS 서비스 또는 리소스와의 상호 작용을 완료해야 하는 요청을 수신할 때만 수행됩니다. 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 역할은에 연결된 서비스 역할의 한 유형입니다 AWS 서비스. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은에 표시 AWS 계정 되며 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

- Amazon EC2에서 실행되는 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 AWS 역할을 할당하고 모든 애플리케이션에서 사용할 수 있도록 하려면 인스턴스에 연결된 인스턴스 프로파일을 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.

정책을 사용하여 액세스 관리

정책을 AWS 생성하고 자격 증명 또는 리소스에 연결하여 AWS 의 액세스를 제어합니다. 정책은 자격 증명 또는 리소스와 연결된 AWS 경우 권한을 정의하는의 객체입니다.는 보안 주체(사용자, 루트 사용자 또는 역할 세션)가 요청할 때 이러한 정책을 AWS 평가합니다. 정책에서 권한은 요청이 허용되거나 거부되는 지를 결정합니다. 대부분의 정책은에 JSON 문서 AWS 로 저장됩니다. JSON 정책 문서의 구조와 콘텐츠에 대한 자세한 정보는 IAM 사용 설명서의 JSON 정책 개요를 참조하세요.

관리자는 AWS JSON 정책을 사용하여 누가 무엇을 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

기본적으로, 사용자 및 역할에는 어떠한 권한도 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 수임할 수 있습니다.

IAM 정책은 작업을 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 API에서 역할 정보를 가져올 수 있습니다 AWS .

ID 기반 정책

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.

ID 기반 정책은 인라인 정책 또는 관리형 정책으로 한층 더 분류할 수 있습니다. 인라인 정책은 단일 사용자, 그룹 또는 역할에 직접 포함됩니다. 관리형 정책은의 여러 사용자, 그룹 및 역할에 연결할 수 있는 독립 실행형 정책입니다 AWS 계정. 관리형 정책에는 AWS 관리형 정책 및 고객 관리형 정책이 포함됩니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 관리형 정책 및 인라인 정책 중에서 선택을 참조하세요.

리소스 기반 정책

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 또는가 포함될 수 있습니다 AWS 서비스.

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. 리소스 기반 정책에서는 IAM의 AWS 관리형 정책을 사용할 수 없습니다.

액세스 제어 목록(ACL)

액세스 제어 목록(ACL)은 어떤 보안 주체(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 S3 AWS WAF및 Amazon VPC는 ACLs. ACL에 관한 자세한 내용은 Amazon Simple Storage Service 개발자 가이드의 액세스 제어 목록(ACL) 개요를 참조하세요.

기타 정책 타입

AWS 는 덜 일반적인 추가 정책 유형을 지원합니다. 이러한 정책 타입은 더 일반적인 정책 유형에 따라 사용자에게 부여되는 최대 권한을 설정할 수 있습니다.

- 권한 경계 – 권한 경계는 ID 기반 정책에 따라 IAM 엔터티(IAM 사용자 또는 역할)에 부여할 수 있는 최대 권한을 설정하는 고급 기능입니다. 개체에 대한 권한 경계를 설정할 수 있습니다. 그 결과로 얻는 권한은 객체의 ID 기반 정책과 그 권한 경계의 교집합입니다. `Principal` 필드에서 사용자나 역할을 지정하는 리소스 기반 정책은 권한 경계를 통해 제한되지 않습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 권한 경계에 대한 자세한 정보는 IAM 사용 설명서의 IAM 엔티티에 대한 권한 경계를 참조하세요.
- 서비스 제어 정책(SCPs) - SCPs는의 조직 또는 조직 단위(OU)에 대한 최대 권한을 지정하는 JSON 정책입니다 AWS Organizations. AWS Organizations 는 비즈니스가 소유 AWS 계정 한 여러를 그룹화하고 중앙에서 관리하기 위한 서비스입니다. 조직에서 모든 기능을 활성화할 경우, 서비스 제어 정책(SCP)을 임의의 또는 모든 계정에 적용할 수 있습니다. SCP는 각각을 포함하여 멤버 계정의 엔터티에 대한 권한을 제한합니다 AWS 계정 루트 사용자. 조직 및 SCP에 대한 자세한 내용은 AWS Organizations 사용 설명서에서 Service control policies을 참조하세요.

- 리소스 제어 정책(RCP) - RCP는 소유한 각 리소스에 연결된 IAM 정책을 업데이트하지 않고 계정의 리소스에 대해 사용 가능한 최대 권한을 설정하는 데 사용할 수 있는 JSON 정책입니다. RCP는 멤버 계정의 리소스에 대한 권한을 제한하며 조직에 속하는지 여부에 AWS 계정 루트 사용자관계없이를 포함한 자격 증명에 대한 유효 권한에 영향을 미칠 수 있습니다. RCP를 AWS 서비스 지원하는 목록을 포함하여 조직 및 RCPs에 대한 자세한 내용은 AWS Organizations 사용 설명서의 리소스 제어 정책(RCPs)을 참조하세요.
- 세션 정책 – 세션 정책은 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 결과적으로 얻는 세션의 권한은 사용자 또는 역할의 ID 기반 정책의 교차와 세션 정책입니다. 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. 가 여러 정책 유형이 관련될 때 요청을 허용할지 여부를 AWS 결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하세요.

Amazon DocumentDB에서 IAM을 사용하는 방법

IAM을 사용하여 Amazon DocumentDB에 대한 액세스를 관리하기 전에 Amazon DocumentDB에서 사용할 수 있는 IAM 기능에 대해 알아봅니다.

Amazon DocumentDB에서 사용할 수 있는 IAM 기능

IAM 기능	인스턴스 기반 클러스터	엘라스틱 클러스터
ID 기반 정책	예	예
리소스 기반 정책	아니요	아니요
정책 작업	예	예
정책 리소스	예	예
정책 조건 키(서비스별)	예	예
ACLs	아니요	아니요

IAM 기능	인스턴스 기반 클러스터	엘라스틱 클러스터
ABAC(정책 내 태그)	부분	예
임시 보안 인증	예	예
보안 주체 권한	예	예
서비스 역할	예	예
서비스 연결 역할	아니요	예

Amazon DocumentDB 및 기타 AWS 서비스가 대부분의 IAM 기능과 작동하는 방식을 전체적으로 알아보려면 IAM 사용 설명서의 AWS IAM으로 작업하는 서비스를 참조하세요.

Amazon DocumentDB의 ID 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. ID 기반 정책에서는 위탁자가 연결된 사용자 또는 역할에 적용되므로 위탁자를 지정할 수 없습니다. JSON 정책에서 사용하는 모든 요소에 대해 알아보려면 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하십시오.

Amazon DocumentDB의 ID 기반 정책 예

Amazon DocumentDB ID 기반 정책 예제를 보려면 Amazon DocumentDB의 ID 기반 정책 예 단원을 참조하십시오.

Amazon DocumentDB 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 관리자

는 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 또는가 포함될 수 있습니다 AWS 서비스.

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 다른 경우 신뢰할 수 AWS 계정있는 계정의 IAM 관리자는 보안 주체 엔터티(사용자 또는 역할)에게 리소스에 액세스할 수 있는 권한도 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 교차 계정 리소스 액세스를 참조하세요.

Amazon DocumentDB의 정책 작업

정책 작업 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇을 액세스할 수 있는지 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. 정책 작업은 일반적으로 연결된 AWS API 작업과 이름이 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.

> **Note**
>
> 일부 관리 기능의 경우 Amazon DocumentDB는 Amazon Relational Database Service(RDS)와 공유되는 운영 기술을 사용합니다.
> RDS 작업의 목록을 보려면 서비스 인증 참조의 Amazon Relational Database Service에서 정의한 작업을 참조하십시오.
> Amazon DocumentDB 탄력적 클러스터에 대한 정책 작업을 보려면 서비스 승인 참조의 Amazon DocumentDB 탄력적 클러스터에서 정의한 작업을 참조하세요.

Amazon DocumentDB의 정책 작업은 작업 앞에 다음 접두사를 사용합니다.

```
aws
```

단일 문에서 여러 작업을 지정하려면 다음과 같이 쉼표로 구분합니다.

```
"Action": [
      "aws:action1",
      "aws:action2"
         ]
```

Amazon DocumentDB ID 기반 정책 예제를 보려면 Amazon DocumentDB의 ID 기반 정책 예 단원을 참조하십시오.

Amazon DocumentDB용 정책 리소스

정책 리소스 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇을 액세스할 수 있는지 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 `Resource`또는 `NotResource`요소가 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": "*"
```

> **Note**
>
> 일부 관리 기능의 경우 Amazon DocumentDB는 Amazon Relational Database Service(RDS)와 공유되는 운영 기술을 사용합니다.
> RDS 리소스 유형 및 해당 ARN 목록을 보려면 서비스 승인 참조에서 Amazon Relational Database Service에서 정의한 리소스를 참조하십시오. 각 리소스의 ARN을 지정할 수 있는 작업을 알아보려면 Amazon Relational Database Service에서 정의한 작업을 참조하십시오.
> Amazon DocumentDB 엘라스틱 클러스터에 대한 리소스 유형을 보려면 서비스 승인 참조의 Amazon DocumentDB 엘라스틱 클러스터에서 정의한 리소스 유형을 참조하십시오.

Amazon DocumentDB ID 기반 정책 예제를 보려면 Amazon DocumentDB의 ID 기반 정책 예 단원을 참조하십시오.

Amazon DocumentDB의 정책 조건 키

서비스별 정책 조건 키 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇을 액세스할 수 있는지 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, AWS 는 논리적 `AND` 작업을 사용하여 평가합니다. 단일 조건 키에 여러 값을 지정하는 경우는 논리적 `OR` 작업을 사용하여 조건을 AWS 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.

AWS 는 전역 조건 키와 서비스별 조건 키를 지원합니다. 모든 AWS 전역 조건 키를 보려면 IAM 사용 설명서의 AWS 전역 조건 컨텍스트 키를 참조하세요.

> **Note**
>
> 일부 관리 기능의 경우 Amazon DocumentDB는 Amazon Relational Database Service(RDS)와 공유되는 운영 기술을 사용합니다.
> RDS 조건 키 목록을 보려면 서비스 승인 참조의 Amazon Relational Database Service에 대한 조건 키를 참조하십시오. 조건 키를 사용할 수 있는 작업과 리소스를 알아보려면 Amazon Relational Database Service에서 정의한 작업을 참조하십시오.
> Amazon DocumentDB 엘라스틱 클러스터에 대한 조건 키를 보려면 서비스 승인 참조의 Amazon DocumentDB 엘라스틱 클러스터에 대한 조건 키를 참조하십시오.

Amazon DocumentDB ID 기반 정책 예제를 보려면 Amazon DocumentDB의 ID 기반 정책 예 단원을 참조하십시오.

Amazon DocumentDB의 ACL

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 DocumentDB의 ABAC

> **Note**
>
> ABAC는 인스턴스 기반 클러스터에서는 일부만 지원되지만 엘라스틱 클러스터에서는 지원됩니다.

속성 기반 액세스 제어(ABAC)는 속성에 근거하여 권한을 정의하는 권한 부여 전략입니다. 에서는 AWS이러한 속성을 태그라고 합니다. IAM 엔터티(사용자 또는 역할)와 많은 AWS 리소스에 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 DocumentDB에서 임시 보안 인증 정보 사용

임시 자격 증명 지원: 예

임시 자격 증명을 사용하여 로그인할 때 작동하지 AWS 서비스 않는 경우도 있습니다. 임시 자격 증명으로 AWS 서비스 작업하는을 비롯한 자세한 내용은 AWS 서비스 IAM 사용 설명서의 IAM으로 작업하는를 참조하세요.

사용자 이름과 암호를 제외한 방법을 AWS Management Console 사용하여에 로그인하는 경우 임시 자격 증명을 사용합니다. 예를 들어 회사의 SSO(Single Sign-On) 링크를 AWS 사용하여에 액세스하면 해당 프로세스가 임시 자격 증명을 자동으로 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 생성할 수 있습니다. 그런 다음 이러한 임시 자격 증명을 사용하여 장기 액세스 키를 사용하는 대신 동적으로 임시 자격 증명을 생성하는 access AWS. AWS recommends에 액세스할 수 있습니다. 자세한 내용은 IAM의 임시 보안 인증 정보를 참조하십시오.

Amazon DocumentDB에 대한 교차 서비스 보안 주체 권한

전달 액세스 세션(FAS) 지원: 예

IAM 사용자 또는 역할을 사용하여에서 작업을 수행하는 경우 AWS보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우, 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는를 호출하는 보안 주체의 권한을 다운스트림 서비스에 AWS 서비스 대한 요청과 AWS 서비스함께 사용합니다. FAS 요청은 서비스가 다른 AWS 서비스 또는 리소스와의 상호 작용을 완료해야 하는 요청을 수신할 때만 수행됩니다. 이 경우, 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.

Amazon DocumentDB의 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 DocumentDB 기능이 중단될 수 있습니다. Amazon DocumentDB에서 관련 지침을 제공하는 경우에만 서비스 역할을 편집합니다.

Amazon DocumentDB의 서비스 연결 역할

> **Note**
>
> 서비스 연결 역할은 인스턴스 기반 클러스터에서는 지원되지 않지만 엘라스틱 클러스터에서는 지원됩니다.

서비스 연결 역할은에 연결된 서비스 역할의 한 유형입니다 AWS 서비스. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은에 표시 AWS 계정 되며 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

서비스 연결 역할 생성 또는 관리에 대한 자세한 내용은 IAM으로 작업하는AWS 서비스를 참조하세요. 서비스 연결 역할 열에서 `Yes`이(가) 포함된 서비스를 테이블에서 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예 링크를 선택합니다.

Amazon DocumentDB의 ID 기반 정책 예

기본적으로 사용자 및 역할은 Amazon DocumentDB 리소스를 생성하거나 수정할 수 있는 권한이 없습니다. 또한 AWS Management Console, AWS Command Line Interface (AWS CLI) 또는 AWS API를 사용하여 작업을 수행할 수 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 맡을 수 있습니다.

이러한 예제 JSON 정책 문서를 사용하여 IAM 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성(콘솔)을 참조하세요.

각 리소스 유형에 대한 ARN 형식을 비롯하여 Amazon DocumentDB에서 정의되는 작업 및 리소스 유형에 대한 자세한 내용은 서비스 승인 참조의 Amazon Relational Database Service에 사용되는 작업, 리소스 및 조건 키를 참조하십시오.

> **Note**
>
> 일부 관리 기능의 경우 Amazon DocumentDB는 Amazon Relational Database Service(RDS)와 공유되는 운영 기술을 사용합니다.
> Amazon DocumentDB 탄력적 클러스터에 대한 정책 작업은 서비스 승인 참조의 Amazon DocumentDB 탄력적 클러스터에 대한 작업, 리소스 및 조건 키를 참조하세요.

주제

- 정책 모범 사례
- Amazon DocumentDB 콘솔 사용
- 사용자가 자신의 고유한 권한을 볼 수 있도록 허용

정책 모범 사례

ID 기반 정책에 따라 계정에서 사용자가 Amazon DocumentDB 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 정책을 시작하고 최소 권한으로 전환 - 사용자 및 워크로드에 권한 부여를 시작하려면 많은 일반적인 사용 사례에 대한 권한을 부여하는 AWS 관리형 정책을 사용합니다. 에서 사용할 수 있습니다 AWS 계정. 사용 사례에 맞는 AWS 고객 관리형 정책을 정의하여 권한을 추가로 줄이는 것이 좋습니다. 자세한 정보는 IAM 사용 설명서의 AWS 관리형 정책 또는 AWS 직무에 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. 조건을 사용하여 AWS 서비스와 같은 특정를 통해 사용되는 서비스 작업에 대한 액세스 권한을 부여할 수도 있습니다 AWS CloudFormation. 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 -에서 IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안을 위해 MFA를 AWS 계정켭니다. 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 DocumentDB 콘솔 사용

Amazon DocumentDB(MongoDB 호환) 콘솔에 액세스하려면 최소한의 권한 세트가 있어야 합니다. 이러한 권한을 통해의 Amazon DocumentDB 리소스에 대한 세부 정보를 나열하고 볼 수 있어야 합니다 AWS 계정. 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 API에만 호출하는 사용자에 대해 최소 콘솔 권한을 허용할 필요는 없습니다. 대신 수행하려는 API 작업과 일치하는 작업에만 액세스할 수 있도록 합니다.

사용자와 역할이 Amazon DocumentDB 콘솔을 계속 사용할 수 있도록 하려면 Amazon DocumentDB *ConsoleAccess* 또는 *ReadOnly* AWS 관리형 정책도 엔터티에 연결합니다. 자세한 내용은 IAM 사용 설명서의 사용자에게 권한 추가를 참조하십시오.

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 방식으로이 작업을 완료할 수 있는 권한이 포함되어 있습니다.

```
{
    "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 DocumentDB ID 및 액세스 문제 해결

다음 정보를 사용하여 Amazon DocumentDB 및 IAM으로 작업할 때 발생할 수 있는 일반적인 문제를 진단하고 수정할 수 있습니다.

주제

- Amazon DocumentDB에서 작업을 수행할 권한이 없음
- iam:PassRole을 수행하도록 인증되지 않음
- 내 외부의 사람이 내 Amazon DocumentDB 리소스 AWS 계정 에 액세스하도록 허용하고 싶습니다.

Amazon DocumentDB에서 작업을 수행할 권한이 없음

작업을 수행할 권한이 없다는 오류가 수신되면, 작업을 수행할 수 있도록 정책을 업데이트해야 합니다.

다음의 예제 오류는 `mateojackson` IAM 사용자가 콘솔을 사용하여 가상 *`my-example-widget`* 리소스에 대한 세부 정보를 보려고 하지만 가상 `aws:`*`GetWidget`* 권한이 없을 때 발생합니다.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 aws:GetWidget on resource: my-example-widget
```

이 경우, `aws:`*`GetWidget`* 작업을 사용하여 *`my-example-widget`* 리소스에 액세스할 수 있도록 `mateojackson` 사용자 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

iam:PassRole을 수행하도록 인증되지 않음

`iam:PassRole` 작업을 수행할 수 있는 권한이 없다는 오류가 수신되면 Amazon DocumentDB에 역할을 전달할 수 있도록 정책을 업데이트해야 합니다.

일부 AWS 서비스 에서는 새 서비스 역할 또는 서비스 연결 역할을 생성하는 대신 기존 역할을 해당 서비스에 전달할 수 있습니다. 이렇게 하려면 사용자가 서비스에 역할을 전달할 수 있는 권한을 가지고 있어야 합니다.

다음 예제 오류는 `marymajor`라는 IAM 사용자가 콘솔을 사용하여 Amazon DocumentDB에서 작업을 수행하려고 하는 경우에 발생합니다. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 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, Mary가 `iam:PassRole` 작업을 수행할 수 있도록 Mary의 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

내 외부의 사람이 내 Amazon DocumentDB 리소스 AWS 계정 에 액세스하도록 허용하고 싶습니다.

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 DocumentDB에서 이러한 기능을 지원하는지 여부를 알아보려면 Amazon DocumentDB에서 IAM을 사용하는 방법 단원을 참조하십시오.
- 소유 AWS 계정 한의 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 소유 AWS 계정 한 다른의 IAM 사용자에게 액세스 권한 제공을 참조하세요.
- 타사에 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 타사 AWS 계정 소유에 대한 액세스 권한 제공을 AWS 계정참조하세요.
- ID 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 외부에서 인증된 사용자에게 액세스 권한 제공(ID 페더레이션)을 참조하세요.

- 크로스 계정 액세스에 대한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명서의 IAM의 크로스 계정 리소스 액세스를 참조하세요.

Amazon DocumentDB 리소스에 대한 액세스 권한 관리

모든 AWS 리소스는에서 소유하며 AWS 계정리소스 생성 또는 액세스 권한은 권한 정책에 의해 관리됩니다. 계정 관리자는 IAM 자격 증명(즉, 사용자, 그룹 및 역할)에 권한 정책을 연결할 수 있으며 일부 서비스(예: AWS Lambda)는 리소스에 권한 정책 연결을 지원합니다.

> **Note**
>
> 계정 관리자(또는 관리자 사용자)는 관리자 권한이 있는 사용자입니다. 자세한 내용은 IAM 사용 설명서의 IAM 모범 사례 단원을 참조하십시오.

주제

- Amazon DocumentDB 리소스 및 운영
- 리소스 소유권 이해
- 리소스 액세스 관리
- 정책 요소 지정: 작업, 효과, 리소스, 보안 주체
- 정책에서 조건 지정

Amazon DocumentDB 리소스 및 운영

Amazon DocumentDB에서 기본 리소스는 클러스터입니다. Amazon DocumentDB는 기본 리소스와 함께 사용할 수 있는 다른 리소스(예: 인스턴스, 파라미터 그룹, 이벤트 구독 등)를 지원합니다. 이 리소스를 하위 리소스라고 합니다.

다음 표에 나와 있는 것처럼 이러한 리소스와 하위 리소스에는 고유한 Amazon 리소스 이름(ARN)이 연결되어 있습니다.

리소스 유형	ARN 형식
클러스터	`arn:aws:rds:` *`region`*`:`*`account-id`* `:cluster:` *`db-cluster-name`*

리소스 유형	ARN 형식
클러스터 파라미터 그룹	arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*
클러스터 스냅샷	arn:aws:rds: *region*:*account-id* :cluster-snapshot: *cluster-snapshot-name*
Instance	arn:aws:rds: *region*:*account-id* :db:*db-instance-name*
보안 그룹	arn:aws:rds: *region*:*account-id* :secgrp:*security-group-name*
서브넷 그룹	arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*

Amazon DocumentDB은 Amazon DocumentDB 리소스를 처리하기 위한 작업을 제공합니다. 사용 가능한 작업 목록은 작업 단원을 참조하십시오.

리소스 소유권 이해

리소스 소유자는 리소스를 AWS 계정 생성한 입니다. 즉, 리소스 소유자는 리소스를 생성하는 요청을 인증하는 보안 주체 엔터티(루트 계정, IAM 사용자 또는 IAM 역할) AWS 계정 의 입니다. 다음의 예제에서는 이러한 작동 방법을 설명합니다.

- 의 루트 계정 자격 증명을 사용하여 인스턴스와 같은 Amazon DocumentDB 리소스를 AWS 계정 생성하는 경우 AWS 계정 는 Amazon DocumentDB 리소스의 소유자입니다.
- 에서 IAM 사용자를 생성하고 해당 사용자에게 Amazon DocumentDB 리소스를 생성할 수 있는 권한을 AWS 계정 부여하는 경우 사용자는 Amazon DocumentDB 리소스를 생성할 수 있습니다. 그러나 사용자가 속한 AWS 계정는 Amazon DocumentDB 리소스를 소유합니다.
- Amazon DocumentDB 리소스를 생성할 수 있는 권한이 AWS 계정 있는에서 IAM 역할을 생성하는 경우 역할을 수임할 수 있는 사람은 누구나 Amazon DocumentDB 리소스를 생성할 수 있습니다. 역할 AWS 계정이 속한이 Amazon DocumentDB 리소스를 소유합니다.

리소스 액세스 관리

권한 정책은 누가 무엇에 액세스 할 수 있는지를 나타냅니다. 다음 섹션에서는 권한 정책을 만드는 데 사용 가능한 옵션에 대해 설명합니다.

> **Note**
>
> 이 섹션에서는 Amazon DocumentDB의 맥락에서 IAM을 사용하는 방법에 대해 설명합니다. IAM 서비스에 대한 자세한 정보는 다루지 않습니다. IAM 설명서 전체 내용은 IAM 사용 설명서의 IAM이란 무엇입니까? 단원을 참조하십시오. IAM 정책 구문과 설명에 대한 자세한 내용은 IAM 사용 설명서의 AWSIAM 정책 참조를 참조하십시오.

IAM ID에 연결된 정책을 ID 기반 정책(IAM 정책)이라고 합니다. 리소스에 연결된 정책을 리소스 기반 정책이라고 합니다. Amazon DocumentDB는 ID 기반 정책(IAM 정책)만 지원합니다.

주제

- 자격 증명 기반 정책(IAM 정책)
- 리소스 기반 정책

자격 증명 기반 정책(IAM 정책)

정책을 IAM 보안 인증에 연결할 수 있습니다. 예를 들면, 다음을 수행할 수 있습니다:

- 계정 내 사용자 또는 그룹에 권한 정책 연결 - 계정 관리자는 특정 사용자에 연결된 권한 정책을 사용하여 해당 사용자에게 Amazon DocumentDB 리소스(예: 인스턴스) 생성 권한을 부여할 수 있습니다.
- 역할에 권한 정책 연결(교차 계정 권한 부여) – ID 기반 권한 정책을 IAM 역할에 연결하여 교차 계정 권한을 부여할 수 있습니다. 예를 들어 관리자는 다음과 같이 다른 AWS 계정 또는 AWS 서비스에 교차 계정 권한을 부여하는 역할을 생성할 수 있습니다.
  1. 계정 A 관리자는 IAM 역할을 생성하고 계정 A의 리소스에 대한 권한을 부여하는 역할에 권한 정책을 연결합니다.
  2. 계정 A 관리자는 계정 B를 역할을 수임할 보안 주체로 식별하는 역할에 신뢰 정책을 연결합니다.
  3. 그런 다음 계정 B 관리자는 계정 B의 모든 사용자에게 역할을 수임할 권한을 위임할 수 있습니다. 이렇게 하면 계정 B의 사용자가 계정 A의 리소스를 생성하거나 액세스할 수 있습니다. 역할을 수임할 권한을 AWS 서비스에 부여하려는 경우 신뢰 정책의 보안 주체가 AWS 서비스 보안 주체가 될 수도 있습니다.

  IAM을 사용하여 권한을 위임하는 방법에 대한 자세한 설명은 IAM 사용자 가이드의 액세스 관리 섹션을 참조하세요.

다음은 ID가 123456789012인 사용자에게 AWS 계정계정에 대한 인스턴스를 생성하도록 허용하는 정책 예제입니다. 새로운 인스턴스는 default로 시작하는 옵션 그룹 및 파라미터 그룹과 default 서브넷 그룹을 사용해야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCreateDBInstanceOnly",
            "Effect": "Allow",
            "Action": [
                "rds:CreateDBInstance"
            ],
            "Resource": [
                "arn:aws:rds:*:123456789012:db:test*",
                "arn:aws:rds:*:123456789012:pg:cluster-pg:default*",
                "arn:aws:rds:*:123456789012:subgrp:default"
            ]
        }
    ]
}
```

Amazon DocumentDB에서 ID 기반 정책 사용에 대한 자세한 내용은 Amazon DocumentDB에 대한 ID 기반 정책(IAM 정책) 사용 단원을 참조하십시오. 사용자, 그룹, 역할 및 권한에 대한 자세한 내용은 IAM 사용 설명서의 ID(사용자, 그룹 및 역할)을 참조하십시오.

리소스 기반 정책

Amazon Simple Storage Service(S3)와 같은 다른 서비스도 리소스 기반 권한 정책을 지원합니다. 예를 들어 Amazon S3 버킷에 정책을 연결하여 해당 버킷에 대한 액세스 권한을 관리할 수 있습니다. Amazon DocumentDB는 리소스 기반 액세스 정책을 지원하지 않습니다.

정책 요소 지정: 작업, 효과, 리소스, 보안 주체

각 Amazon DocumentDB 리소스(Amazon DocumentDB 리소스 및 운영 참조)에서 이 서비스는 API 작업을 정의합니다. 자세한 내용은 작업을 참조하십시오. 이러한 API 작업에 대한 권한을 부여하기 위해 Amazon DocumentDB에서는 정책에서 지정할 수 있는 작업을 정의합니다. API 작업을 실시하려면 둘 이상의 작업에 대한 권한이 필요할 수 있습니다.

다음은 기본 정책 요소입니다.

- 리소스 – 정책에서 Amazon 리소스 이름(ARN)을 사용하여 정책을 적용할 리소스를 식별합니다.
- 조치 – 조치 키워드를 사용하여 허용 또는 거부할 리소스 작업을 식별합니다. 예를 들어 `rds:DescribeDBInstances` 권한은 사용자에게 `DescribeDBInstances` 작업 수행을 허용합니다.
- 결과 – 사용자가 특정 작업을 요청하는 경우의 결과를 지정합니다. 이는 허용 또는 거부 중에 하나가 될 수 있습니다. 명시적으로 리소스에 대한 액세스 권한을 부여(허용)하지 않는 경우, 액세스는 묵시적으로 거부됩니다. 다른 정책에서 액세스 권한을 부여하는 경우라도 사용자가 해당 리소스에 액세스할 수 없도록 하기 위해 리소스에 대한 권한을 명시적으로 거부할 수도 있습니다.
- 보안 주체 – ID 기반 정책(IAM 정책)에서 정책이 연결되는 사용자는 암시적인 보안 주체입니다. 리소스 기반 정책의 경우, 사용자, 계정, 서비스 또는 권한의 수신자인 기타 개체를 지정합니다(리소스 기반 정책에만 해당). Amazon DocumentDB는 리소스 기반 액세스 정책을 지원하지 않습니다.

IAM 정책 구문과 설명에 대한 자세한 내용은 IAM 사용 설명서의 AWS IAM 정책 참조를 참조하십시오.

모든 Amazon DocumentDB API 작업과 해당 작업이 적용되는 리소스를 보여주는 표는 Amazon DocumentDB API 권한: 작업, 리소스 및 조건 참조 단원을 참조하십시오.

정책에서 조건 지정

권한을 부여할 때 IAM 정책 언어를 사용하여 정책이 적용되는 조건을 지정할 수 있습니다. 예를 들어, 특정 날짜 이후에만 정책을 적용할 수 있습니다. 정책 언어에서의 조건 지정에 관한 자세한 내용은 IAM 사용 설명서의 조건을 참조하십시오.

조건을 표시하려면 미리 정의된 조건 키를 사용합니다. Amazon DocumentDB에는 IAM 정책에 사용할 수 있는 서비스별 컨텍스트 키가 없습니다. 모든 서비스에 사용할 수 있는 글로벌 컨텍스트 키의 목록은 IAM 사용 설명서의 조건에 사용 가능한 키를 참조하십시오.

Amazon DocumentDB에 대한 ID 기반 정책(IAM 정책) 사용

Important

일부 관리 기능의 경우 Amazon DocumentDB는 Amazon RDS와 공유되는 운영 기술을 사용합니다. Amazon DocumentDB 콘솔 AWS CLI및 API 호출은 Amazon RDS API에 대한 호출로 로깅됩니다.

Amazon DocumentDB 리소스에 대한 액세스 관리를 위해 제공되는 기본 개념과 옵션 설명에 대한 소개 주제 부분을 먼저 읽어 보는 것이 좋습니다. 자세한 내용은 Amazon DocumentDB 리소스에 대한 액세스 권한 관리 단원을 참조하십시오.

이 항목에서는 계정 관리자가 IAM ID(사용자, 그룹, 역할)에 권한 정책을 연결할 수 있는 ID 기반 정책의 예를 제공합니다.

다음은 IAM 정책 예제입니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCreateDBInstanceOnly",
            "Effect": "Allow",
            "Action": [
                "rds:CreateDBInstance"
            ],
            "Resource": [
                "arn:aws:rds:*:123456789012:db:test*",
                "arn:aws:rds:*:123456789012:pg:cluster-pg:default*",
                "arn:aws:rds:*:123456789012:subgrp:default"
            ]
        }
    ]
}
```

정책은 다음 IAM 사용자 권한을 지정하는 단일 명령문을 포함합니다.

- 이 정책은 IAM 사용자가 CreateDBInstance 작업을 사용하여 인스턴스를 생성할 수 있도록 허용합니다(이는 create-db-instance AWS CLI 작업 및 에도 적용됨 AWS Management Console).
- `Resource` 요소는 사용자가 리소스 위치에서 또는 리소스를 사용하여 작업을 수행할 수 있도록 지정합니다. Amazon 리소스 이름(ARN)을 사용하여 리소스를 지정합니다. 이 ARN에는 리소스가 속한 서비스의 이름(`rds`), AWS 리전 (는`*`이 예제의 모든 리전을 나타냄), 사용자 계정 번호(`123456789012`는이 예제의 사용자 ID) 및 리소스 유형이 포함됩니다.

  위의 예제에서 `Resource` 요소는 사용자 리소스에 대해 다음과 같은 정책 제약 조건을 지정합니다.

  - 새 인스턴스의 인스턴스 식별자는 `test`로 시작해야 합니다(예: `testCustomerData1`, `test-region2-data`).
  - 새로운 인스턴스의 클러스터 파라미터 그룹은 `default`로 시작해야 합니다.
  - 새로운 인스턴스의 서브넷 그룹은 `default` 서브넷 그룹이 되어야 합니다.

ID 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 연결하면 역할의 신뢰 정책에서 식별된 보안 주체가 권한을 얻습니다.

모든 Amazon DocumentDB API 작업과 해당 작업이 적용되는 리소스를 보여주는 표는 Amazon DocumentDB API 권한: 작업, 리소스 및 조건 참조 단원을 참조하십시오.

Amazon DocumentDB 콘솔 사용에 필요한 권한

Amazon DocumentDB 콘솔에서 작업하려면 최소한의 권한이 사용자에게 필요합니다. 이러한 권한을 통해 사용자는에 대한 Amazon DocumentDB 리소스를 설명하고 Amazon EC2 보안 및 네트워크 정보를 포함한 기타 관련 정보를 제공할 AWS 계정 수 있습니다.

최소 필수 권한보다 더 제한적인 IAM 정책을 만들면 콘솔은 해당 IAM 정책에 연결된 사용자에 대해 의도대로 작동하지 않습니다. 이 사용자가 Amazon DocumentDB 콘솔을 사용할 수 있도록 하려면 `AmazonDocDBConsoleFullAccess` 관리형 정책을 사용자에게 연결합니다(AWS Amazon DocumentDB에 대한 관리형 정책 참조).

AWS CLI 또는 Amazon DocumentDB API에만 호출하는 사용자에 대해 최소 콘솔 권한을 허용할 필요는 없습니다.

고객 관리형 정책 예

이 섹션에서는 다양한 Amazon DocumentDB 작업에 대한 권한을 부여하는 사용자 정책의 예를 제공합니다. 이러한 정책은 Amazon DocumentDB API 작업, AWS SDKs 또는를 사용할 때 작동합니다 AWS CLI. 콘솔을 사용하는 경우 Amazon DocumentDB 콘솔 사용에 필요한 권한의 설명과 같이 콘솔에 특정한 추가 권한을 부여해야 합니다.

일부 관리 기능의 경우 Amazon DocumentDB는 Amazon Relational Database Service(RDS) 및 Amazon Neptune과 공유되는 운영 기술을 사용합니다.

> **Note**
>
> 모든 예에서는 미국 동부(버지니아 북부) 리전(`us-east-1`)을 사용하며 가상의 계정 ID를 포함합니다.

예시

- 예제 1: 사용자가 모든 Amazon DocumentDB 리소스에서 Describe 작업을 수행할 수 있도록 허용

- 예제 2: 사용자의 인스턴스 삭제 방지
- 예 3: 스토리지 암호화가 활성화되지 않은 경우 사용자가 클러스터를 생성하지 못하도록 차단

예제 1: 사용자가 모든 Amazon DocumentDB 리소스에서 Describe 작업을 수행할 수 있도록 허용

다음 권한 정책은 사용자에게 `Describe`로 시작하는 모든 작업을 실행할 수 있는 권한을 부여합니다. 이 작업은 인스턴스와 같은 Amazon DocumentDB 리소스에 대한 정보를 보여 줍니다. `Resource` 요소에 와일드카드 문자(*)가 있으면 계정이 소유한 모든 Amazon DocumentDB 리소스에 작업이 허용됩니다.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowRDSDescribe",
         "Effect":"Allow",
         "Action":"rds:Describe*",
         "Resource":"*"
      }
   ]
}
```

예제 2: 사용자의 인스턴스 삭제 방지

다음 권한 정책은 사용자의 특정 인스턴스 삭제를 방지하는 권한을 부여합니다. 예를 들어, 관리자가 아닌 모든 사용자에 대해 프로덕션 인스턴스를 삭제할 수 있는 권한을 거부해야 할 수 있습니다.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"DenyDelete1",
         "Effect":"Deny",
         "Action":"rds:DeleteDBInstance",
         "Resource":"arn:aws:rds:us-east-1:123456789012:db:my-db-instance"
      }
   ]
}
```

예 3: 스토리지 암호화가 활성화되지 않은 경우 사용자가 클러스터를 생성하지 못하도록 차단

다음 권한 정책은 스토리지 암호화가 활성화되지 않은 경우 Amazon DocumentDB 클러스터를 생성할 수 있는 사용자의 권한을 거부합니다.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "PreventUnencryptedDocumentDB",
         "Effect": "Deny",
         "Action": "RDS:CreateDBCluster",
         "Condition": {
         "Bool": {
         "rds:StorageEncrypted": "false"
      },
         "StringEquals": {
         "rds:DatabaseEngine": "docdb"
         }
      },
      "Resource": "*"
      }
   ]
}
```

AWS Amazon DocumentDB에 대한 관리형 정책

사용자, 그룹 및 역할에 권한을 추가하려면 직접 정책을 작성하는 것보다 AWS 관리형 정책을 사용하는 것이 더 쉽습니다. 팀에 필요한 권한만 제공하는 IAM 고객 관리형 정책을 생성하기 위해서는 시간과 전문 지식이 필요합니다. 빠르게 시작하려면 AWS 관리형 정책을 사용할 수 있습니다. 이러한 정책은 일반적인 사용 사례를 다루며 AWS 계정에서 사용할 수 있습니다. AWS 관리형 정책에 대한 자세한 내용은 AWS Identity and Access Management 사용 설명서의 AWS 관리형 정책을 참조하세요.

AWS 서비스는 AWS 관리형 정책을 유지 관리하고 업데이트합니다. AWS 관리형 정책에서는 권한을 변경할 수 없습니다. 서비스는 때때로 AWS 관리형 정책에 추가 권한을 추가하여 새 기능을 지원합니다. 이 유형의 업데이트는 정책이 연결된 모든 ID(사용자, 그룹 및 역할)에 적용됩니다. 서비스는 새 기능이 시작되거나 새 작업을 사용할 수 있게 되면 AWS 관리형 정책을 업데이트할 가능성이 높습니다. 서비스는 AWS 관리형 정책에서 권한을 제거하지 않으므로 정책 업데이트가 기존 권한을 손상시키지 않습니다.

또한 여러 서비스에 걸쳐 있는 직무에 대한 관리형 정책을 AWS 지원합니다. 예를 들어 `ViewOnlyAccess` AWS 관리형 정책은 많은 AWS 서비스 및 리소스에 대한 읽기 전용 액세스를 제공합니다. 서비스가 새 기능을 시작하면는 새 작업 및 리소스에 대한 읽기 전용 권한을 AWS 추가합니다. 직무 정책의 목록과 설명은 AWS ID 및 액세스 관리 사용 설명서의 직무에 관한AWS 관리형 정책을 참조하십시오.

계정의 사용자에게 연결할 수 있는 다음 AWS 관리형 정책은 Amazon DocumentDB에만 해당됩니다.

- AmazonDocDBFullAccess - 루트 AWS 계정의 모든 Amazon DocumentDB 리소스에 대한 전체 액세스 권한을 부여합니다.
- AmazonDocDBReadOnlyAccess - 루트 AWS 계정의 모든 Amazon DocumentDB 리소스에 대한 읽기 전용 액세스 권한을 부여합니다.
- AmazonDocDBConsoleFullAccess - AWS Management Console를 사용하여 Amazon DocumentDB 및 Amazon DocumentDB 엘라스틱 클러스터 리소스를 관리할 수 있는 전체 액세스를 부여합니다.
- AmazonDocDBElasticReadOnlyAccess - 루트 AWS 계정의 모든 Amazon DocumentDB 탄력적 클러스터 리소스에 대한 읽기 전용 액세스 권한을 부여합니다.
- AmazonDocDBElasticFullAccess - 루트 AWS 계정의 모든 Amazon DocumentDB 탄력적 클러스터 리소스에 대한 전체 액세스 권한을 부여합니다.

AmazonDocDBFullAccess

이 정책은 모든 Amazon DocumentDB 작업에 대한 전체 액세스 권한을 허용하는 관리 권한을 보안 주체에게 부여합니다. 해당 정책의 권한은 다음과 같이 그룹화됩니다.

- Amazon DocumentDB 권한은 모든 Amazon DocumentDB 작업을 허용합니다.
- 이 정책의 일부 Amazon EC2 권한은 API 요청에서 전달된 리소스를 검증하는 데 필요합니다. 이는 Amazon DocumentDB가 클러스터에서 리소스를 성공적으로 사용할 수 있도록 하기 위한 것입니다. 이 정책의 나머지 Amazon EC2 권한은 Amazon DocumentDB가 클러스터에 연결하는 데 필요한 AWS 리소스를 생성하도록 허용합니다.
- Amazon DocumentDB 권한은 API 호출 중에 요청에서 전달된 리소스를 검증하는 데 사용됩니다. Amazon DocumentDB가 전달된 키를 Amazon DocumentDB 클러스터에서 사용할 수 있도록 하기 위해 필요합니다.
- CloudWatch Logs는 Amazon DocumentDB가 로그 전송 대상에 도달할 수 있는지, 브로커 로그 사용에 유효한지 확인하는 데 필요합니다.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "rds:AddRoleToDBCluster",
                "rds:AddSourceIdentifierToSubscription",
                "rds:AddTagsToResource",
                "rds:ApplyPendingMaintenanceAction",
                "rds:CopyDBClusterParameterGroup",
                "rds:CopyDBClusterSnapshot",
                "rds:CopyDBParameterGroup",
                "rds:CreateDBCluster",
                "rds:CreateDBClusterParameterGroup",
                "rds:CreateDBClusterSnapshot",
                "rds:CreateDBInstance",
                "rds:CreateDBParameterGroup",
                "rds:CreateDBSubnetGroup",
                "rds:CreateEventSubscription",
                "rds:DeleteDBCluster",
                "rds:DeleteDBClusterParameterGroup",
                "rds:DeleteDBClusterSnapshot",
                "rds:DeleteDBInstance",
                "rds:DeleteDBParameterGroup",
                "rds:DeleteDBSubnetGroup",
                "rds:DeleteEventSubscription",
                "rds:DescribeAccountAttributes",
                "rds:DescribeCertificates",
                "rds:DescribeDBClusterParameterGroups",
                "rds:DescribeDBClusterParameters",
                "rds:DescribeDBClusterSnapshotAttributes",
                "rds:DescribeDBClusterSnapshots",
                "rds:DescribeDBClusters",
                "rds:DescribeDBEngineVersions",
                "rds:DescribeDBInstances",
                "rds:DescribeDBLogFiles",
                "rds:DescribeDBParameterGroups",
                "rds:DescribeDBParameters",
                "rds:DescribeDBSecurityGroups",
                "rds:DescribeDBSubnetGroups",
                "rds:DescribeEngineDefaultClusterParameters",
                "rds:DescribeEngineDefaultParameters",
                "rds:DescribeEventCategories",
                "rds:DescribeEventSubscriptions",
                "rds:DescribeEvents",
```

```
                "rds:DescribeOptionGroups",
                "rds:DescribeOrderableDBInstanceOptions",
                "rds:DescribePendingMaintenanceActions",
                "rds:DescribeValidDBInstanceModifications",
                "rds:DownloadDBLogFilePortion",
                "rds:FailoverDBCluster",
                "rds:ListTagsForResource",
                "rds:ModifyDBCluster",
                "rds:ModifyDBClusterParameterGroup",
                "rds:ModifyDBClusterSnapshotAttribute",
                "rds:ModifyDBInstance",
                "rds:ModifyDBParameterGroup",
                "rds:ModifyDBSubnetGroup",
                "rds:ModifyEventSubscription",
                "rds:PromoteReadReplicaDBCluster",
                "rds:RebootDBInstance",
                "rds:RemoveRoleFromDBCluster",
                "rds:RemoveSourceIdentifierFromSubscription",
                "rds:RemoveTagsFromResource",
                "rds:ResetDBClusterParameterGroup",
                "rds:ResetDBParameterGroup",
                "rds:RestoreDBClusterFromSnapshot",
                "rds:RestoreDBClusterToPointInTime"
            ],
            "Effect": "Allow",
            "Resource": [
                "*"
            ]
        },
        {
            "Action": [
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics",
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeVpcs",
                "kms:ListAliases",
                "kms:ListKeyPolicies",
                "kms:ListKeys",
                "kms:ListRetirableGrants",
                "logs:DescribeLogStreams",
```

```
                "logs:GetLogEvents",
                "sns:ListSubscriptions",
                "sns:ListTopics",
                "sns:Publish"
            ],
            "Effect": "Allow",
            "Resource": [
                "*"
            ]
        },
        {
            "Action": "iam:CreateServiceLinkedRole",
            "Effect": "Allow",
            "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
AWSServiceRoleForRDS",
            "Condition": {
                "StringLike": {
                    "iam:AWS ServiceName": "rds.amazonaws.com"
                }
            }
        }
    ]
}
```

AmazonDocDBReadOnlyAccess

이 정책은 Amazon DocumentDB의 정보를 볼 수 있는 읽기 전용 권한을 사용자에게 부여합니다. 이 정책이 첨부된 보안 주체는 기존 리소스를 업데이트하거나 삭제할 수 없으며 새 Amazon DocumentDB 리소스를 생성할 수도 없습니다. 예를 들어 이러한 권한이 있는 주체는 자신의 계정과 연결된 클러스터 및 구성 목록을 볼 수 있지만 클러스터의 구성이나 설정을 변경할 수는 없습니다. 해당 정책의 권한은 다음과 같이 그룹화됩니다.

- Amazon DocumentDB 권한을 사용하면 Amazon DocumentDB 리소스를 나열하고, 설명하고, 그에 대한 정보를 얻을 수 있습니다.
- Amazon EC2 권한은 클러스터와 연결된 Amazon VPC, 서브넷, 보안 그룹, ENI를 설명하는 데 사용됩니다.
- Amazon DocumentDB 권한은 클러스터와 연결된 키를 설명하는 데 사용됩니다.

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Action": [
                "rds:DescribeAccountAttributes",
                "rds:DescribeCertificates",
                "rds:DescribeDBClusterParameterGroups",
                "rds:DescribeDBClusterParameters",
                "rds:DescribeDBClusterSnapshotAttributes",
                "rds:DescribeDBClusterSnapshots",
                "rds:DescribeDBClusters",
                "rds:DescribeDBEngineVersions",
                "rds:DescribeDBInstances",
                "rds:DescribeDBLogFiles",
                "rds:DescribeDBParameterGroups",
                "rds:DescribeDBParameters",
                "rds:DescribeDBSubnetGroups",
                "rds:DescribeEventCategories",
                "rds:DescribeEventSubscriptions",
                "rds:DescribeEvents",
                "rds:DescribeOrderableDBInstanceOptions",
                "rds:DescribePendingMaintenanceActions",
                "rds:DownloadDBLogFilePortion",
                "rds:ListTagsForResource"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeInternetGateways",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeVpcs"
```

```
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "kms:ListKeys",
                "kms:ListRetirableGrants",
                "kms:ListAliases",
                "kms:ListKeyPolicies"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "logs:DescribeLogStreams",
                "logs:GetLogEvents"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:logs:*:*:log-group:/aws/rds/*:log-stream:*",
                "arn:aws:logs:*:*:log-group:/aws/docdb/*:log-stream:*"
            ]
        }
    ]
}
```

AmazonDocDBConsoleFullAccess

다음을 위해를 사용하여 Amazon DocumentDB 리소스를 관리할 AWS Management Console 수 있는 전체 액세스 권한을 부여합니다.

- 모든 Amazon DocumentDB 및 Amazon DocumentDB 클러스터 작업을 허용할 수 있는 Amazon DocumentDB 권한
- 이 정책의 일부 Amazon EC2 권한은 API 요청에서 전달된 리소스를 검증하는 데 필요합니다. 이는 Amazon DocumentDB가 클러스터에서 리소스를 성공적으로 프로비저닝하고 유지할 수 있도록 하기 위한 것입니다. 이 정책의 나머지 Amazon EC2 권한은 Amazon DocumentDB가 VPCEndpoint와 같은 클러스터에 연결하는 데 필요한 AWS 리소스를 생성하도록 허용합니다.

- AWS KMS 권한은에 대한 API 호출 중에 요청에서 전달된 리소스를 검증 AWS KMS 하는 데 사용됩니다. 이는 Amazon DocumentDB가 전달된 키를 사용해 Amazon DocumentDB 엘라스틱 클러스터에서 저장 데이터를 암호화 및 암호 해독할 수 있도록 하기 위해 필요합니다.
- CloudWatch Logs는 Amazon DocumentDB가 로그 전송 대상에 도달할 수 있는지, 로그 사용 감사 및 프로파일링에 유효한지 확인하는 데 필요합니다.
- 주어진 암호를 검증하고 이를 사용하여 Amazon DocumentDB 엘라스틱 클러스터의 관리자를 설정하려면 Secrets Manager 권한이 필요합니다.
- Amazon DocumentDB 클러스터 관리 작업에는 Amazon RDS 권한이 필요합니다. 일부 관리 기능의 경우 Amazon DocumentDB는 Amazon RDS와 공유되는 운영 기술을 사용합니다.
- SNS 권한은 보안 주체에게 Amazon Simple Notification Service(SNS) 구독 및 주제를 허용하고 Amazon SNS 메시지를 게시하도록 허용합니다.
- 지표 및 로그 게시에 필요한 서비스 연결 역할을 생성하려면 IAM 권한이 필요합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DocdbSids",
            "Effect": "Allow",
            "Action": [
                "docdb-elastic:CreateCluster",
                "docdb-elastic:UpdateCluster",
                "docdb-elastic:GetCluster",
                "docdb-elastic:DeleteCluster",
                "docdb-elastic:ListClusters",
                "docdb-elastic:CreateClusterSnapshot",
                "docdb-elastic:GetClusterSnapshot",
                "docdb-elastic:DeleteClusterSnapshot",
                "docdb-elastic:ListClusterSnapshots",
                "docdb-elastic:RestoreClusterFromSnapshot",
                "docdb-elastic:TagResource",
                "docdb-elastic:UntagResource",
                "docdb-elastic:ListTagsForResource",
                "docdb-elastic:CopyClusterSnapshot",
                "docdb-elastic:StartCluster",
                "docdb-elastic:StopCluster",
                "docdb-elastic:GetPendingMaintenanceAction",
                "docdb-elastic:ListPendingMaintenanceActions",
                "docdb-elastic:ApplyPendingMaintenanceAction",
```

```
                "rds:AddRoleToDBCluster",
                "rds:AddSourceIdentifierToSubscription",
                "rds:AddTagsToResource",
                "rds:ApplyPendingMaintenanceAction",
                "rds:CopyDBClusterParameterGroup",
                "rds:CopyDBClusterSnapshot",
                "rds:CopyDBParameterGroup",
                "rds:CreateDBCluster",
                "rds:CreateDBClusterParameterGroup",
                "rds:CreateDBClusterSnapshot",
                "rds:CreateDBInstance",
                "rds:CreateDBParameterGroup",
                "rds:CreateDBSubnetGroup",
                "rds:CreateEventSubscription",
                "rds:CreateGlobalCluster",
                "rds:DeleteDBCluster",
                "rds:DeleteDBClusterParameterGroup",
                "rds:DeleteDBClusterSnapshot",
                "rds:DeleteDBInstance",
                "rds:DeleteDBParameterGroup",
                "rds:DeleteDBSubnetGroup",
                "rds:DeleteEventSubscription",
                "rds:DeleteGlobalCluster",
                "rds:DescribeAccountAttributes",
                "rds:DescribeCertificates",
                "rds:DescribeDBClusterParameterGroups",
                "rds:DescribeDBClusterParameters",
                "rds:DescribeDBClusterSnapshotAttributes",
                "rds:DescribeDBClusterSnapshots",
                "rds:DescribeDBClusters",
                "rds:DescribeDBEngineVersions",
                "rds:DescribeDBInstances",
                "rds:DescribeDBLogFiles",
                "rds:DescribeDBParameterGroups",
                "rds:DescribeDBParameters",
                "rds:DescribeDBSecurityGroups",
                "rds:DescribeDBSubnetGroups",
                "rds:DescribeEngineDefaultClusterParameters",
                "rds:DescribeEngineDefaultParameters",
                "rds:DescribeEventCategories",
                "rds:DescribeEventSubscriptions",
                "rds:DescribeEvents",
                "rds:DescribeGlobalClusters",
                "rds:DescribeOptionGroups",
```

```
                "rds:DescribeOrderableDBInstanceOptions",
                "rds:DescribePendingMaintenanceActions",
                "rds:DescribeValidDBInstanceModifications",
                "rds:DownloadDBLogFilePortion",
                "rds:FailoverDBCluster",
                "rds:ListTagsForResource",
                "rds:ModifyDBCluster",
                "rds:ModifyDBClusterParameterGroup",
                "rds:ModifyDBClusterSnapshotAttribute",
                "rds:ModifyDBInstance",
                "rds:ModifyDBParameterGroup",
                "rds:ModifyDBSubnetGroup",
                "rds:ModifyEventSubscription",
                "rds:ModifyGlobalCluster",
                "rds:PromoteReadReplicaDBCluster",
                "rds:RebootDBInstance",
                "rds:RemoveFromGlobalCluster",
                "rds:RemoveRoleFromDBCluster",
                "rds:RemoveSourceIdentifierFromSubscription",
                "rds:RemoveTagsFromResource",
                "rds:ResetDBClusterParameterGroup",
                "rds:ResetDBParameterGroup",
                "rds:RestoreDBClusterFromSnapshot",
                "rds:RestoreDBClusterToPointInTime"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "DependencySids",
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "cloudwatch:GetMetricData",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics",
                "ec2:AllocateAddress",
                "ec2:AssignIpv6Addresses",
                "ec2:AssignPrivateIpAddresses",
                "ec2:AssociateAddress",
                "ec2:AssociateRouteTable",
                "ec2:AssociateSubnetCidrBlock",
                "ec2:AssociateVpcCidrBlock",
```

```
                "ec2:AttachInternetGateway",
                "ec2:AttachNetworkInterface",
                "ec2:CreateCustomerGateway",
                "ec2:CreateDefaultSubnet",
                "ec2:CreateDefaultVpc",
                "ec2:CreateInternetGateway",
                "ec2:CreateNatGateway",
                "ec2:CreateNetworkInterface",
                "ec2:CreateRoute",
                "ec2:CreateRouteTable",
                "ec2:CreateSecurityGroup",
                "ec2:CreateSubnet",
                "ec2:CreateVpc",
                "ec2:CreateVpcEndpoint",
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeAddresses",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeCustomerGateways",
                "ec2:DescribeInstances",
                "ec2:DescribeNatGateways",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribePrefixLists",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroupReferences",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeVpcEndpoints",
                "ec2:DescribeVpcs",
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:ModifySubnetAttribute",
                "ec2:ModifyVpcAttribute",
                "ec2:ModifyVpcEndpoint",
                "kms:DescribeKey",
                "kms:ListAliases",
                "kms:ListKeyPolicies",
                "kms:ListKeys",
                "kms:ListRetirableGrants",
                "logs:DescribeLogStreams",
                "logs:GetLogEvents",
                "sns:ListSubscriptions",
                "sns:ListTopics",
                "sns:Publish"
            ],
```

```
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "DocdbSLRSid",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
AWSServiceRoleForRDS",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "rds.amazonaws.com"
                }
            }
        },
        {
            "Sid": "DocdbElasticSLRSid",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/docdb-
elastic.amazonaws.com/AWSServiceRoleForDocDB-Elastic",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "docdb-elastic.amazonaws.com"
                }
            }
        }
    ]
}
```

AmazonDocDBElasticReadOnlyAccess

이 정책은 Amazon DocumentDB의 엘라스틱 클러스터 정보를 볼 수 있는 읽기 전용 권한을 사용자에게 부여합니다. 이 정책이 첨부된 보안 주체는 기존 리소스를 업데이트하거나 삭제할 수 없으며 새 Amazon DocumentDB 리소스를 생성할 수도 없습니다. 예를 들어 이러한 권한이 있는 주체는 자신의 계정과 연결된 클러스터 및 구성 목록을 볼 수 있지만 클러스터의 구성이나 설정을 변경할 수는 없습니다. 해당 정책의 권한은 다음과 같이 그룹화됩니다.

- Amazon DocumentDB 엘라스틱 클러스터 권한을 사용하면 Amazon DocumentDB 엘라스틱 클러스터 리소스를 나열하고, 설명하고, 그에 대한 정보를 얻을 수 있습니다.
- CloudWatch 권한은 서비스 지표를 확인하는 데 사용됩니다.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "docdb-elastic:ListClusters",
            "docdb-elastic:GetCluster",
            "docdb-elastic:ListClusterSnapshots",
            "docdb-elastic:GetClusterSnapshot",
            "docdb-elastic:ListTagsForResource"
         ],
         "Resource": "*"
      },
      {
         "Effect": "Allow",
         "Action": [
            "cloudwatch:GetMetricData",
            "cloudwatch:ListMetrics",
            "cloudwatch:GetMetricStatistics"
         ],
         "Resource": "*"
      }
   ]
}
```

AmazonDocDBElasticFullAccess

이 정책은 모든 Amazon DocumentDB 엘라스틱 클러스터의 Amazon DocumentDB 작업에 대한 전체 액세스 권한을 허용하는 관리 권한을 보안 주체에게 부여합니다.

이 정책은 조건 내에서 AWS 태그(https://docs.aws.amazon.com/tag-editor/latest/userguide/tagging.html)를 사용하여 리소스에 대한 액세스 범위를 지정합니다. 암호를 사용하는 경우 태그 키 `DocDBElasticFullAccess`과 태그 값으로 태그를 지정해야 합니다. 고객 관리형 키를 사용하는 경우 태그 키 `DocDBElasticFullAccess`과 태그 값으로 태그를 지정해야 합니다.

해당 정책의 권한은 다음과 같이 그룹화됩니다.

- Amazon DocumentDB 엘라스틱 클러스터 권한은 모든 Amazon DocumentDB 작업을 허용합니다.
- 이 정책의 일부 Amazon EC2 권한은 API 요청에서 전달된 리소스를 검증하는 데 필요합니다. 이는 Amazon DocumentDB가 클러스터에서 리소스를 성공적으로 프로비저닝하고 유지할 수 있도록 하

기 위한 것입니다. 이 정책의 나머지 Amazon EC2 권한을 통해 Amazon DocumentDB는 VPC 엔드포인트와 같은 클러스터에 연결하는 데 필요한 AWS 리소스를 생성할 수 있습니다.

- AWS KMS Amazon DocumentDB가 전달된 키를 사용하여 Amazon DocumentDB 탄력적 클러스터 내에서 저장 데이터를 암호화하고 복호화하려면 권한이 필요합니다.

> **Note**
>
> 고객 관리형 키에는 키 `DocDBElasticFullAccess`과 태그 값이 있는 태그가 있어야 합니다.

- 주어진 암호를 검증하고 이를 사용하여 Amazon DocumentDB 엘라스틱 클러스터의 관리자를 설정하려면 SecretsManager 권한이 필요합니다.

> **Note**
>
> 사용되는 암호에는 키 `DocDBElasticFullAccess`과 태그 값이 있는 태그가 있어야 합니다.

- 지표 및 로그 게시에 필요한 서비스 연결 역할을 생성하려면 IAM 권한이 필요합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DocdbElasticSid",
            "Effect": "Allow",
            "Action": [
                "docdb-elastic:CreateCluster",
                "docdb-elastic:UpdateCluster",
                "docdb-elastic:GetCluster",
                "docdb-elastic:DeleteCluster",
                "docdb-elastic:ListClusters",
                "docdb-elastic:CreateClusterSnapshot",
                "docdb-elastic:GetClusterSnapshot",
                "docdb-elastic:DeleteClusterSnapshot",
                "docdb-elastic:ListClusterSnapshots",
                "docdb-elastic:RestoreClusterFromSnapshot",
                "docdb-elastic:TagResource",
                "docdb-elastic:UntagResource",
                "docdb-elastic:ListTagsForResource",
```

```
                "docdb-elastic:CopyClusterSnapshot",
                "docdb-elastic:StartCluster",
                "docdb-elastic:StopCluster",
                "docdb-elastic:GetPendingMaintenanceAction",
                "docdb-elastic:ListPendingMaintenanceActions",
                "docdb-elastic:ApplyPendingMaintenanceAction"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "EC2Sid",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateVpcEndpoint",
                "ec2:DescribeVpcEndpoints",
                "ec2:DeleteVpcEndpoints",
                "ec2:ModifyVpcEndpoint",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ec2:DescribeAvailabilityZones",
                "secretsmanager:ListSecrets"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaFirst": "docdb-elastic.amazonaws.com"
                }
            }
        },
        {
            "Sid": "KMSSid",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey"
            ],
            "Resource": "*",
```

```
            "Condition": {
                "StringLike": {
                    "kms:ViaService": [
                        "docdb-elastic.*.amazonaws.com"
                    ],
                    "aws:ResourceTag/DocDBElasticFullAccess": "*"
                }
            }
        },
        {
            "Sid": "KMSGrantSid",
            "Effect": "Allow",
            "Action": [
                "kms:CreateGrant"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "aws:ResourceTag/DocDBElasticFullAccess": "*",
                    "kms:ViaService": [
                        "docdb-elastic.*.amazonaws.com"
                    ]
                },
                "Bool": {
                    "kms:GrantIsForAWSResource": true
                }
            }
        },
        {
            "Sid": "SecretManagerSid",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:ListSecretVersionIds",
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:GetResourcePolicy"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "secretsmanager:ResourceTag/DocDBElasticFullAccess": "*"
                },
                "StringEquals": {
                    "aws:CalledViaFirst": "docdb-elastic.amazonaws.com"
```

```
                }
            }
        },
        {
            "Sid": "CloudwatchSid",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData",
                "cloudwatch:ListMetrics",
                "cloudwatch:GetMetricStatistics"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "SLRSid",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/docdb-
elastic.amazonaws.com/AWSServiceRoleForDocDB-Elastic",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "docdb-elastic.amazonaws.com"
                }
            }
        }
    ]
}
```

AmazonDocDB-ElasticServiceRolePolicy

AWS Identity and Access Management 엔터AmazonDocDBElasticServiceRolePolicy티에 연결할 수 없습니다. 이 정책은 Amazon DocumentDB에 사용자를 대신하여 작업을 수행할 수 있도록 하는 서비스 연결 역할에 연결됩니다. 자세한 내용은 탄력적 클러스터에서 서비스 연결 역할 단원을 참조하십시오.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "cloudwatch:PutMetricData"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": [
                        "AWS/DocDB-Elastic"
                    ]
                }
            }
        }
    ]
}
```

AWS 관리형 정책에 대한 Amazon DocumentDB 업데이트

변경 사항	설명	날짜
AmazonDocDBElasticFullAccess, AmazonDocDBConsoleFullAccess - 변경	보류 중인 유지 관리 작업을 추가하도록 정책이 업데이트되었습니다.	2/11/2025
AmazonDocDBElasticFullAccess, AmazonDocDBConsoleFullAccess - 변경	시작/중지 클러스터를 추가하고 클러스터 스냅샷 작업을 복사하도록 정책이 업데이트되었습니다.	2/21/2024
AmazonDocDBElasticReadOnlyAccess, AmazonDocDBElasticFullAccess - 변경	`cloudwatch:GetMetricData` 작업을 추가하기 위해 정책이 업데이트되었습니다.	6/21/2023
AmazonDocDBElasticReadOnlyAccess - 새 정책	Amazon DocumentDB 탄력적 클러스터에 대한 새로운 관리형 정책입니다.	6/8/2023
AmazonDocDBElasticFullAccess - 새 정책	Amazon DocumentDB 탄력적 클러스터에 대한 새로운 관리형 정책입니다.	6/5/2023

변경 사항	설명	날짜
AmazonDocDB-ElasticServiceRolePolicy - 새 정책	Amazon DocumentDB는 Amazon DocumentDB 탄력적 클러스터에 대한 새로운 AWS ServiceRoleForDocDB 탄력적 서비스 연결 역할을 생성합니다. Amazon DocumentDB	11/30/2022
AmazonDocDBConsoleFullAccess - 변경 사항	Amazon DocumentDB 글로벌 및 탄력적 클러스터 권한을 추가하도록 정책이 업데이트되었습니다.	11/30/2022
AmazonDocDBConsoleFullAccess, AmazonDocDBFullAccess, AmazonDocDBReadOnlyAccess - 새 정책	서비스 시작.	1/19/2017

Amazon DocumentDB API 권한: 작업, 리소스 및 조건 참조

IAM ID에 연결할 수 있는 Amazon DocumentDB에 대한 ID 기반 정책(IAM 정책) 사용 및 쓰기 권한 정책(ID 기반 정책)을 설정할 때 다음 단원을 참조하십시오.

다음 목록에는 각 Amazon DocumentDB API 작업이 각각 나열되어 있습니다. 목록에는 작업을 수행할 수 있는 권한을 부여할 수 있는 해당 작업, 권한을 부여할 수 있는 AWS 리소스, 세분화된 액세스 제어를 위해 포함할 수 있는 조건 키가 포함되어 있습니다. 정책의 `Action` 필드에서 작업을 지정하고, `Resource` 필드에서 리소스 값을 지정하고, `Condition` 필드에서 조건을 지정합니다. 조건에 대한 자세한 내용은 정책에서 조건 지정 섹션을 참조하세요.

Amazon DocumentDB 정책에서 AWS전체 조건 키를 사용하여 조건을 표시할 수 있습니다. AWS전체 키의 전체 목록은 IAM 사용 설명서의 사용 가능한 키를 참조하세요.

IAM 정책은 IAM 정책 시뮬레이터로 테스트할 수 있습니다. Amazon DocumentDB AWS 작업을 포함하여 각 작업에 필요한 리소스 및 파라미터 목록을 자동으로 제공합니다. 또한 IAM 정책 시뮬레이터 정책은 사용자가 지정하는 작업마다 필요한 권한을 결정합니다. IAM 정책 테스트에 대한 자세한 내용은 IAM 사용 설명서의 IAM 정책 시뮬레이터로 IAM 정책 테스트 단원을 참조하십시오.

> **Note**
>
> 작업을 지정하려면 `rds:` 접두사 다음에 API 작업 명칭을 사용합니다(예: `rds:CreateDBInstance`).

다음 표에는 Amazon RDS API 작업을 비롯한 관련 작업과 리소스, 그리고 조건 키가 나와 있습니다.

주제

- 리소스 수준 권한을 지원하는 Amazon DocumentDB 작업
- 리소스 수준 권한을 지원하지 않는 Amazon DocumentDB 작업

리소스 수준 권한을 지원하는 Amazon DocumentDB 작업

리소스 수준 권한은 사용자가 작업을 수행할 수 있는 리소스를 지정하는 기능을 제공합니다. Amazon DocumentDB는 리소스 수준 권한을 부분적으로 지원합니다. 즉, 필요 조건을 지정하거나 사용 가능한 특정 리소스를 지정하여 사용자가 특정 Amazon DocumentDB 작업 사용할 수 있는지를 제어할 수 있습니다. 예를 들어 특정 인스턴스로 제한된 변경 권한을 사용자에게 부여할 수 있습니다.

다음 표에는 Amazon DocumentDB API 작업을 비롯한 관련 작업과 리소스, 그리고 조건 키가 나와 있습니다.

> **Note**
>
> 일부 관리 기능의 경우 Amazon DocumentDB는 Amazon RDS와 공유되는 운영 기술을 사용합니다. Amazon DocumentDB 작업 및 권한에 대한 자세한 내용은 서비스 승인 참조에서 Amazon RDS에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

Amazon DocumentDB API 작업 및 작업	리소스	조건 키
AddTagsToResource `rds:AddTagsToResource`	Instance `arn:aws:rds:` *`region`*`:`*`account-id`*` :db:`*`db-instance-name`*	`rds:db-tag`
	서브넷 그룹	`rds:subgrp-tag`

Amazon DocumentDB API 작업 및 작업	리소스	조건 키
	arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	
ApplyPendingMaintenanceAction rds:ApplyPendingMaintenanceAction	Instance arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:db-tag
CopyDBClusterSnapshot rds:CopyDBClusterSnapshot	클러스터 스냅샷 arn:aws:rds: *region*:*account-id* :cluster-snapshot: *cluster-snapshot-name*	rds:cluster-snapshot-tag
CreateDBCluster rds:CreateDBCluster	클러스터 arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-name*	rds:cluster-tag
	클러스터 파라미터 그룹 arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag
	서브넷 그룹 arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag

Amazon DocumentDB API 작업 및 작업	리소스	조건 키
CreateDBClusterParameterGroup rds:CreateDBClusterParameterGroup	클러스터 파라미터 그룹 arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag
CreateDBClusterSnapshot rds:CreateDBClusterSnapshot	클러스터 arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-name*	rds:cluster-tag
	클러스터 스냅샷 arn:aws:rds: *region*:*account-id* :cluster-snapshot: *cluster-snapshot-name*	rds:cluster-snapshot-tag
CreateDBInstance rds:CreateDBInstance	Instance arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:DatabaseClass rds:db-tag
	클러스터 arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-name*	rds:cluster-tag
CreateDBSubnetGroup rds:CreateDBSubnetGroup	서브넷 그룹 arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag

Amazon DocumentDB API 작업 및 작업	리소스	조건 키
DeleteDBInstance rds:DeleteDBInstance	Instance arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:db-tag
DeleteDBSubnetGroup rds:DeleteDBSubnetGroup	서브넷 그룹 arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag
DescribeDBClusterParameterGroups rds:DescribeDBClusterParameterGroups	클러스터 파라미터 그룹 arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag
DescribeDBClusterParameters rds:DescribeDBClusterParameters	클러스터 파라미터 그룹 arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag
DescribeDBClusters rds:DescribeDBClusters	클러스터 arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-instance-name*	rds:cluster-tag

Amazon DocumentDB API 작업 및 작업	리소스	조건 키
DescribeDBClusterSnapshotAttributes rds:DescribeDBClusterSnapshotAttributes	클러스터 스냅샷 arn:aws:rds: *region*:*account-id* :cluster-snapshot: *cluster-snapshot-name*	rds:cluster-snapshot-tag
DescribeDBSubnetGroups rds:DescribeDBSubnetGroups	서브넷 그룹 arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag
DescribePendingMaintenanceActions rds:DescribePendingMaintenanceActions	Instance arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:DatabaseClass rds:db-tag
FailoverDBCluster rds:FailoverDBCluster	클러스터 arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-instance-name*	rds:cluster-tag
ListTagsForResource rds:ListTagsForResource	Instance arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:db-tag

Amazon DocumentDB API 작업 및 작업	리소스	조건 키
	서브넷 그룹 arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag
ModifyDBCluster rds:ModifyDBCluster	클러스터 arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-name*	rds:cluster-tag
	클러스터 파라미터 그룹 arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag
ModifyDBClusterParameterGroup rds:ModifyDBClusterParameterGroup	클러스터 파라미터 그룹 arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag
ModifyDBClusterSnapshotAttribute rds:ModifyDBClusterSnapshotAttribute	클러스터 스냅샷 arn:aws:rds: *region*:*account-id* :cluster-snapshot: *cluster-snapshot-name*	rds:cluster-snapshot-tag

Amazon DocumentDB API 작업 및 작업	리소스	조건 키
ModifyDBInstance rds:ModifyDBInstance	Instance arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:DatabaseClass rds:db-tag
RebootDBInstance rds:RebootDBInstance	Instance arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:db-tag
RemoveTagsFromResource rds:RemoveTagsFromResource	Instance arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:db-tag
	서브넷 그룹 arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag
ResetDBClusterParameterGroup rds:ResetDBClusterParameterGroup	클러스터 파라미터 그룹 arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag
RestoreDBClusterFromSnapshot rds:RestoreDBClusterFromSnapshot	클러스터 arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-instance-name*	rds:cluster-tag

Amazon DocumentDB API 작업 및 작업	리소스	조건 키
	클러스터 스냅샷 arn:aws:rds: *region*:*account-id* :cluster-snapshot: *cluster-snapshot-name*	rds:cluster-snapshot-tag
RestoreDBClusterToPointInTime rds:RestoreDBClusterToPointInTime	클러스터 arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-instance-name*	rds:cluster-tag
	서브넷 그룹 arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag

리소스 수준 권한을 지원하지 않는 Amazon DocumentDB 작업

작업 사용 권한의 부여 여부를 결정하는 IAM 정책에서는 모든 Amazon DocumentDB 작업을 사용할 수 있습니다. 하지만 모든 Amazon DocumentDB 작업이 리소스 수준 권한을 지원하는 것은 아닙니다. 여기에서 리소스 수준 권한이란 작업이 가능한 리소스를 지정할 수 있는 권한을 말합니다. 현재 다음 Amazon DocumentDB API 작업은 리소스 수준 권한을 지원하지 않습니다. 따라서 IAM 정책에서 이러한 작업을 사용하려면 `Resource` 요소에 * 와일드카드를 사용하여 해당 작업 리소스를 모두 사용할 수 있는 권한을 사용자에게 부여해야 합니다.

- `rds:DescribeDBClusterSnapshots`
- `rds:DescribeDBInstances`

IAM 자격 증명을 사용한 인증

Amazon DocumentDB 사용자 및 애플리케이션은 IAM 사용자 및 역할을 사용하여 Amazon DocumentDB 클러스터에 인증할 수 있습니다. Amazon DocumentDB IAM 인증은 비밀번호가 없는 인

증 방법입니다. 또한 클라이언트 애플리케이션은 IAM 역할/사용자를 사용할 때 Amazon DocumentDB 클러스터에 비밀번호 암호를 전송하지 않습니다. 대신 클라이언트 연결은 임시 보안 토큰을 AWS STS 사용하여 인증됩니다. 이제 관리자가 아닌 사용자 및 애플리케이션은 다른 Amazon DocumentDB 클러스터 및 기타 AWS 서비스에 연결할 때 동일한 IAM 자격 증명 ARN을 사용할 수 있습니다.

암호 기반 인증과 IAM 인증을 모두 사용하여 Amazon DocumentDB 클러스터에 대한 사용자 및 애플리케이션을 인증하도록 선택할 수도 있습니다. IAM 인증은 Amazon DocumentDB 인스턴스 기반 클러스터 버전 5.0에서만 사용할 수 있습니다. Amazon DocumentDB 기본 사용자는 IAM 자격 증명 ARN을 사용한 IAM 인증이 지원되지 않습니다.

> **Note**
>
> 기본 사용자는 기존 비밀번호 기반 인증을 통해서만 인증할 수 있습니다.

주제

- IAM 사용자 및 역할을 사용한 인증 시작하기
- AWS IAM을 사용하여 Amazon DocumentDB에 인증하도록 AWS 컴퓨팅 유형 구성
- IAM 인증 요청 모니터링
- IAM 인증 사용
- IAM을 지원하는 드라이버
- IAM 자격 증명 인증 FAQ

IAM 사용자 및 역할을 사용한 인증 시작하기

IAM 자격 증명이 있는 Amazon DocumentDB 사용자 및 역할은 `$external` 데이터베이스에서 생성 및 관리됩니다.

사용자 생성

기본 사용자로 연결한 다음 IAM 사용자 및 역할을 생성합니다.

```
use $external;
db.createUser(
    {
        user: "arn:aws:iam::123456789123:user/iamuser",
        mechanisms: ["MONGODB-AWS"],
```

```
        roles: [ { role: "readWrite", db: "readWriteDB" } ]
    }
);
```

또는 IAM 역할을 사용하여 Amazon DocumentDB 사용자를 추가합니다.

```
use $external;
db.createUser(
    {
        user: "arn:aws:iam::123456789123:role/iamrole",
        mechanisms: ["MONGODB-AWS"],
        roles: [ { role: "readWrite", db: "readWriteDB" } ]
    }
);
```

IAM 사용자 또는 역할 수정

기존 IAM 사용자를 수정합니다.

```
use $external;
db.updateUser(
  "arn:aws:iam::123456789123:user/iamuser",
  {
    roles: [ { role: "read", db: "readDB" } ]
  }
);
```

기존 IAM 역할을 수정합니다.

```
use $external;
db.updateUser(
  "arn:aws:iam::123456789123:role/iamrole",
  {
    roles: [ { role: "read", db: "readDB" } ]
  }
);
```

IAM 사용자로부터 역할을 부여하거나 취소하려면

```
use $external;
db.grantRolesToUser(
```

```
  "arn:aws:iam::123456789123:user/iamuser",
  [ { db: "admin", role: "readWriteAnyDatabase" } ]
);
```

```
use $external;
db.revokeRolesFromUser(
  "arn:aws:iam::123456789123:user/iamuser",
  [ { db: "admin", role: "readWriteAnyDatabase" } ]
);
```

IAM 역할에서 역할을 부여하거나 취소하려면

```
use $external;
db.grantRolesToUser(
  "arn:aws:iam::123456789123:user/iamrole",
  [ { db: "admin", role: "readWriteAnyDatabase" } ]
);
```

```
use $external;
db.revokeRolesFromUser(
  "arn:aws:iam::123456789123:user/iamrole",
  [ { db: "admin", role: "readWriteAnyDatabase" } ]
);
```

IAM 사용자 또는 역할 삭제

기존 IAM 사용자를 삭제하려면

```
use $external;
db.dropUser("arn:aws:iam::123456789123:user/iamuser");
```

기존 IAM 역할을 삭제하려면

```
use $external;
db.dropUser("arn:aws:iam::123456789123:role/iamrole");
```

AWS IAM을 사용하여 인증하도록 연결 URI 구성

AWS IAM을 사용하여 인증하려면 다음 URI 파라미터를 `authSource`로 `$external` 사용하고 `authMechanism`로 사용합니다`MONGODB-AWS`. IAM 사용자를 사용하는 경우 사용자 이름과 암호 필드

가 각각 액세스 키와 보안 키로 바뀝니다. IAM 역할을 수임하는 경우 현재 환경에 연결됩니다(예: AWS Lambda 함수, Amazon EC2 인스턴스). MONGODB-AWS 메커니즘을 사용하여 인증할 때 자격 증명을 구체적으로 전달할 필요가 없습니다. MONGODB-AWS 인증 메커니즘을 지원하는 MongoDB 드라이버를 사용하는 경우 드라이버는 컴퓨팅 인스턴스(예: Amazon EC2, Lambda 함수 등)에서 IAM 역할 자격 증명을 검색할 수도 있습니다. 다음 예제에서는 mongo 쉘을 사용하여 (IAM 사용자의 것) 액세스 키와 비밀 키를 수동으로 전달하여 MONGODB-AWS를 사용하여 인증하는 예제로 Amazon DocumentDB에 대한 인증을 보여줍니다.

다음 예제에서는 Python 코드를 사용하여 자격 증명을 명시적으로 전달하지 않고(환경에 연결된 IAM 역할 사용) MONGODB-AWS를 사용하여 인증하여 Amazon DocumentDB 에 대한 인증을 보여줍니다.

```
##Create a MongoDB client, open a connection to Amazon DocumentDB using an IAM role
    client = pymongo.MongoClient('mongodb://<DocDBEndpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false&authSource=
%24external&authMechanism=MONGODB-AWS')
```

다음 예제에서는 mongo 쉘을 사용하여 액세스 키 및 보안 키(IAM 사용자의 것)를 수동으로 전달하여 MONGODB-AWS 메커니즘을 사용하여 인증하는 것으로 Amazon DocumentDB에 대한 인증을 보여줍니다.

```
$ mongo 'mongodb://<access_key>:<secret_key>@<cluster_endpoint>:<db_port>/test?
authSource=%24external&authMechanism=MONGODB-AWS'
```

다음 예제에서는 mongo 쉘을 사용하여 자격 증명을 명시적으로 전달하지 않고(환경에 연결된 IAM 역할 사용) MONGODB-AWS를 사용하여 인증하여 Amazon DocumentDB 에 대한 인증을 보여줍니다.

```
$ mongo 'mongodb://<cluster_endpoint>:<db_port>/test?authSource=
%24external&authMechanism=MONGODB-AWS'
```

AWS IAM을 사용하여 Amazon DocumentDB에 인증하도록 AWS 컴퓨팅 유형 구성

Amazon EC2/AWS Lambda/AWS Fargate사용

Amazon EC2는 다음 환경 변수를 사용합니다. EC2 인스턴스에 연결된 IAM 역할 또는 Lambda 함수 또는 Amazon ECS 태스크와 연결된 실행 IAM 역할이 있는 경우 이러한 변수가 자동으로 채워지고 드라이버는 환경에서 이러한 값을 가져올 수 있습니다.

```
AWS_ACCESS_KEY_ID
AWS_SECRET_ACCESS_KEY
AWS_SESSION_TOKEN
```

환경 변수에 대한 자세한 내용은 AWS Lambda 개발자 안내서의 Lambda 환경 변수 사용을 참조하세요.

Amazon EKS 사용

Amazon Elastic Kubernetes Service(Amazon EKS) 포드에 역할을 할당하면 다음 두 가지 환경 변수가 자동으로 설정됩니다.

```
AWS_WEB_IDENTITY_TOKEN_FILE - path of web identity token file
AWS_ROLE_ARN - Name of IAM role to connect with
```

이러한 변수의 도움을 받아에 대한 AWS SDK 호출을 사용하여 코드에서 역할을 수동으로 수임합니다`AssumeRoleWithWebIdentity`.

- `ProviderID` 파라미터를 생략합니다.
- `AWS_WEB_IDENTITY_TOKEN_FILE` 환경 변수에 설명된 파일에서 `WebIdentityToken` 파라미터 값을 찾습니다.

Amazon EKS에 관한 자세한 내용은 Amazon EKS 사용 설명서의 Amazon EKS란을 참조하세요.

IAM 인증 요청 모니터링

Amazon DocumentDB 감사 사용

Amazon CloudWatch 의 감사 로그 폴더로 이동하여 다른 검색 패턴을 사용하여 IAM 인증에 대한 로그를 가져옵니다. 예를 들어 `{ $.param.mechanism = "MONGODB-AWS" }`를 '모든 로그 스트림 검색'의 검색 패턴으로 사용합니다.

감사에서 지원되는 이벤트에 대한 자세한 내용은 Amazon DocumentDB 이벤트 감사 섹션을 참조하세요.

Amazon CloudWatch 지표 사용

`StsGetCallerIdentityCalls`:이 지표는 리전화된 AWS Security Token Service (AWS STS) 엔드포인트에 대해 Amazon DocumentDB 인스턴스가 수행하는 `GetCallerIdentity` 호출 수를 보여줍

니다. 데이터베이스 인스턴스가 STS `GetCallerIdentity`를 호출해야 하는 이유에 대한 `MONGODB-AWS` 인증 사양을 참조하세요.

IAM 인증 사용

자체 데이터베이스에서 사용자 이름과 암호를 관리하지 않으려면 IAM 인증을 사용할 수 있습니다. IAM 인증은 Amazon DocumentDB 인스턴스 기반 클러스터 버전 5.0에서만 사용할 수 있습니다.

IAM 인증은 STS 서비스에 의존합니다. 연결에 IAM 인증을 사용하고 STS 스로틀링 예외를 가져올 때 연결 속도를 낮출 수 있는지를 평가하는 것이 좋습니다.

IAM 할당량은 IAM 사용 설명서의 IAM 및 AWS STS 할당량을 참조하세요.

IAM을 지원하는 드라이버

Amazon DocumentDB 5.0 및 `MONGODB-AWS` 인증 메커니즘을 지원하는 드라이버는 Amazon DocumentDB 에서 IAM 인증 구현을 사용해야 합니다.

> **Important**
>
> 버전 6.13.1보다 오래된 Node.js 드라이버에는 알려진 제한 사항이 있으며, 현재 Amazon DocumentDB에 대한 IAM 자격 증명 인증에서 지원되지 않습니다. Node.js 드라이버를 사용하는 Node.js 드라이버 및 도구(예: mongosh)는 Node.js 드라이버 버전 6.13.1 이상을 사용하도록 업그레이드해야 합니다.

IAM 자격 증명 인증 FAQ

참조할 수 있는 샘플이 있나요?

샘플 사용 사례 및 구성은 다음 페이지를 참조하세요.

- 인간 사용자가 IAM 사용자 및 IAM 역할을 사용하여 Amazon DocumentDB에 인증하는 방법
- IAM 역할을 사용하여 Amazon DocumentDB에 대한 비밀번호 없는 인증

Python 드라이버를 사용하는 동안 “pymongo.errors.ConfigurationError: MONGODB-AWS authentication requires pymongo-auth-aws” 오류가 발생합니다. 이 문제를 해결하려면 어떻게 해야 하나요?

IAM 인증과 함께 Python 드라이버를 설치하는 동안 다음 문을 사용해야 합니다.

```
pip install 'pymongo[aws]'
```

그러면 IAM 인증이 작동하는 데 필요한 추가 AWS 종속성이 설치됩니다.

IAM 역할 임시 자격 증명이 만료되면 연결이 끊어집니까?

아니요. 임시 IAM 자격 증명은 연결 및 인증을 설정하는 데만 사용됩니다. 그런 다음 Amazon DocumentDB 클러스터에서 모든 추가 권한이 발생합니다. IAM 자격 증명이 교체/만료되더라도 연결이 끊기거나 경과되지 않습니다.

Amazon DocumentDB 사용자 관리

Amazon DocumentDB에서 사용자는 암호와 함께 클러스터에 대해 인증합니다. 각 클러스터에는 클러스터 생성 중에 설정되는 기본 로그인 자격 증명이 있습니다.

> **Note**
>
> 2020년 3월 26일 이전에 생성된 모든 새 사용자에게는 `dbAdminAnyDatabase`, `readWriteAnyDatabase` 및 `clusterAdmin` 역할이 부여되었습니다. 모든 사용자를 재평가하고 필요에 따라 역할을 수정하여 클러스터의 모든 사용자에 대해 최소 권한을 적용하는 것이 좋습니다.
> 자세한 내용은 역할 기반 액세스 제어를 사용한 데이터베이스 액세스 단원을 참조하십시오.

기본 및 `serviceadmin` 사용자

새로 생성된 Amazon DocumentDB 클러스터에는 기본 사용자와 `serviceadmin` 사용자의 두 사용자가 있습니다.

기본 사용자는 관리 작업을 수행하고 역할을 가진 추가 사용자를 생성할 수 있는 권한이 있는 단일 사용자입니다. Amazon DocumentDB 클러스터에 처음 연결하는 경우 기본 로그인 자격 증명을 사용하여 인증해야 합니다. 기본 사용자는 클러스터가 생성될 때 Amazon DocumentDB 클러스터에 대한 이러한 관리 권한을 받고 `root`의 역할이 부여됩니다.

클러스터가 생성될 때 `serviceadmin` 사용자는 암시적으로 생성됩니다. 모든 Amazon DocumentDB 클러스터에는 클러스터 AWS 를 관리할 수 있는 기능을 제공하는 `serviceadmin` 사용자가 있습니다. `serviceadmin`으로 로그인하거나, 그 암호를 삭제, 변경, 이름을 바꾸거나, 또는 그 권한을 변경할 수 없습니다. 이렇게 하려고 하면 오류가 발생합니다.

> **Note**
>
> Amazon DocumentDB 클러스터의 기본 및 `serviceadmin` 사용자는 삭제할 수 없으며 기본 사용자에 대한 `root`의 역할은 취소할 수 없습니다.
> 기본 사용자 암호를 잊어버린 경우 AWS Management Console 또는를 사용하여 재설정할 수 있습니다 AWS CLI.

추가 사용자 생성

기본 사용자(또는 `createUser` 역할이 있는 사용자)로 연결한 후 아래와 같이 새 사용자를 생성할 수 있습니다.

```
db.createUser(
    {
        user: "sample-user-1",
        pwd: "password123",
        roles:
            [{"db":"admin", "role":"dbAdminAnyDatabase" }]
    }
)
```

사용자 세부 정보를 보려면 다음과 같이 `show users` 명령을 사용할 수 있습니다. `dropUser` 명령을 사용하여 사용자를 추가로 제거할 수 있습니다. 자세한 내용은 공통 명령 단원을 참조하십시오.

```
show users
{
    "_id" : "serviceadmin",
    "user" : "serviceadmin",
    "db" : "admin",
    "roles" : [
     {
            "role" : "root",
            "db" : "admin"
        }
    ]
},

{
```

```
    "_id" : "myPrimaryUser",
    "user" : "myPrimaryUser",
    "db" : "admin",
    "roles" : [
     {
            "role" : "root",
            "db" : "admin"
        }
    ]
},

{
    "_id" : "sample-user-1",
    "user" : "sample-user-1",
    "db" : "admin",
    "roles" : [
     {
            "role" : "dbAdminAnyDatabase",
            "db" : "admin"
     }
    ]
}
```

이 예에서 새 사용자 `sample-user-1`는 `admin` 데이터베이스에 기인합니다. 신규 사용자의 경우 항상 그렇습니다. Amazon DocumentDB에는 `authenticationDatabase`라는 개념이 없으므로 모든 인증은 `admin` 데이터베이스 컨텍스트에서 수행됩니다.

사용자를 만들 때 역할 지정 시 db 필드를 생략하면 Amazon DocumentDB에서 연결이 실행되는 데이터베이스에 역할의 속성을 암시적으로 지정합니다. 예를 들어, `sample-database` 데이터베이스에 대해 연결이 실행되고 다음 명령을 실행하면 `sample-user-2` 사용자가 `admin` 데이터베이스에 생성되고 `sample-database` 데이터베이스에 대한 `readWrite` 권한이 부여됩니다.

```
db.createUser(
    {
        user: "sample-user-2",
        pwd: "password123",
        roles:
            ["readWrite"]
    }
)
```

모든 데이터베이스에서 범위가 지정된 역할(예: `readInAnyDatabase`)을 가진 사용자를 생성하려면 사용자를 생성할 때 `admin` 데이터베이스 컨텍스트에 있거나 사용자를 생성할 때 역할에 대한 데이터베이스를 명백하게 명시해야 합니다.

데이터베이스의 컨텍스트를 전환하려면 다음 명령을 사용할 수 있습니다.

```
use admin
```

역할 기반 액세스 제어 및 클러스터의 사용자 사이에 최소 권한 적용에 대한 자세한 내용은 역할 기반 액세스 제어를 사용한 데이터베이스 액세스 섹션을 참조하십시오.

Amazon DocumentDB의 암호 자동 교체

를 사용하면 코드의 하드 코딩된 자격 증명(암호 포함)을 Secrets Manager에 대한 API 호출로 대체하여 프로그래밍 방식으로 보안 암호를 검색할 AWS Secrets Manager수 있습니다. 이렇게 하면 보안 암호가 해당 위치에 있지 않기 때문에 여러분의 코드를 검사하는 누군가에 의해 보안 암호가 손상되지 않도록 방지할 수 있습니다. 또한 사용자가 지정한 일정에 따라 Secrets Manager가 자동으로 보안 암호를 교체하도록 구성할 수 있습니다. 따라서 단기 보안 암호로 장기 보안 암호를 교체할 수 있어 손상 위험이 크게 줄어듭니다.

Secrets Manager를 사용하면 Secrets Manager가 제공하는 AWS Lambda 함수를 사용하여 Amazon DocumentDB 암호(즉, 암호)를 자동으로 교체할 수 있습니다.

Amazon DocumentDB AWS Secrets Manager 와의 네이티브 통합에 대한 자세한 내용은 다음을 참조하세요.

- 블로그: AWS Secrets Manager에서 Amazon DocumentDB 및 Amazon Redshift의 자격 증명을 교체하는 방법
- AWS Secrets Manager란 무엇입니까?
- 비밀 AWS Secrets Manager 로테이트
- Secrets Manager의 Amazon DocumentDB 보안 인증 정보

역할 기반 액세스 제어를 사용한 데이터베이스 액세스

Amazon DocumentDB (MongoDB 호환)의 역할 기반 액세스 제어 (RBAC)를 사용하여 데이터베이스에서 사용자가 수행할 수 있는 작업에 대한 액세스를 제한할 수 있습니다. RBAC는 사용자에게 하나 이상의 역할을 부여하여 작동합니다. 이러한 역할은 사용자가 데이터베이스 리소스에서 수행할 수 있는 작업을 결정합니다. Amazon DocumentDB는 현재, `read`, `readWrite`, `readAnyDatabase`,

clusterAdmin 등 데이터베이스 수준에서 범위가 지정된 기본 제공 역할과, 요구 사항에 따라 특정 작업으로 범위를 지정할 수 있는 사용자 정의 역할과 컬렉션과 같은 세분화된 리소스를 모두 지원합니다.

RBAC의 일반적인 사용 사례에는 클러스터의 데이터베이스에 대한 읽기 전용 액세스 권한을 가진 사용자를 생성하여 최소 권한을 적용하는 것과 단일 사용자가 클러스터의 지정된 데이터베이스에 액세스할 수 있도록 하는 다중 테넌트 애플리케이션 디자인이 포함됩니다.

> **Note**
>
> 2020년 3월 26일 이전에 생성된 모든 새 사용자에게는 dbAdminAnyDatabase, readWriteAnyDatabase 및 clusterAdmin 역할이 부여되었습니다. 기존 사용자를 모두 재평가하고 필요에 따라 역할을 수정하여 클러스터에 대한 최소 권한을 적용하는 것이 좋습니다.

주제

- RBAC 개념
- RBAC 내장된 역할 시작하기
- RBAC 사용자 정의 역할 시작하기
- Amazon DocumentDB에 사용자로 연결
- 공통 명령
- 기능적 차이
- Limits
- 역할 기반 액세스 제어를 사용한 데이터베이스 액세스

RBAC 개념

다음은 역할 기반 액세스 제어와 관련된 중요한 용어와 개념입니다. Amazon DocumentDB 사용자에 관한 자세한 내용은 Amazon DocumentDB 사용자 관리 단원을 참조하세요.

- 사용자 - 데이터베이스에 대해 인증하고 작업을 수행할 수 있는 개별 엔터티입니다.
- 암호 - 사용자를 인증하는 데 사용되는 암호입니다.
- 역할 - 사용자에게 하나 이상의 데이터베이스에서 작업을 수행할 수 있는 권한을 부여합니다.
- 관리 데이터베이스 - 사용자가 저장되고 권한이 부여되는 데이터베이스입니다.

- 데이터베이스(**db**) - 문서를 저장하기 위한 컬렉션을 포함하는 클러스터 내의 네임스페이스입니다.

다음 명령은 sample-user라는 사용자를 생성합니다.

```
db.createUser({user: "sample-user", pwd: "abc123", roles: [{role: "read", db: "sample-database"}]})
```

이 예시에서는 다음이 적용됩니다.

- user: "sample-user" - 사용자 이름을 나타냅니다.
- pwd: "abc123" - 사용자 암호를 나타냅니다.
- role: "read", "db: "sample-database" - sample-user 사용자에게 sample-database에서 읽기 권한이 있음을 나타냅니다.



다음 예제에서는 db.getUser(sample-user)를 사용하여 sample-user 사용자를 가져온 후 출력을 보여 줍니다. 이 예제에서 sample-user 사용자는 admin 데이터베이스에 상주하지만 sample-database데이터베이스에 대한 읽기 역할이 있습니다.



사용자를 만들 때 역할 지정 시 db 필드를 생략하면 Amazon DocumentDB에서 연결이 실행되는 데이터베이스에 역할의 속성을 암시적으로 지정합니다. 예를 들어, sample-database 데이터베이스에

대해 연결이 실행되고 다음 명령을 실행하면 sample-user 사용자가 admin 데이터베이스에 생성되고 sample-database 데이터베이스에 대한 readWrite 권한이 부여됩니다.

```
db.createUser({user: "sample-user", pwd: "abc123", roles: ["readWrite"]})
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
{
   "user":"sample-user",
   "roles":[
      {
         "db":"sample-database",
         "role":"readWrite"
      }
   ]
}
```

모든 데이터베이스에서 범위가 지정된 역할(예: readAnyDatabase)을 가진 사용자를 생성하려면 사용자를 생성할 때 admin 데이터베이스 컨텍스트에 있거나 사용자를 생성할 때 역할에 대한 데이터베이스를 명백하게 명시해야 합니다. admin 데이터베이스에 대해 명령을 실행하려면 use admin 명령을 사용할 수 있습니다. 자세한 내용은 공통 명령 단원을 참조하십시오.

RBAC 내장된 역할 시작하기

역할 기반 액세스 제어를 시작하는 데 도움이 되도록 이 단원에서는 서로 다른 작업 기능을 가진 세 명의 사용자에 대한 역할을 생성하여 최소 권한을 적용하는 예제 시나리오를 설명합니다.

- user1은 클러스터의 모든 데이터베이스를 보고 액세스할 수 있어야 하는 새로운 관리자입니다.
- user2는 동일한 클러스터에서 하나의 데이터베이스 sample-database-1에만 액세스해야 하는 신입 직원입니다.
- user3은 동일한 클러스터에서 이전에 액세스할 수 없었던 다른 데이터베이스 sample-database-2를 보고 액세스해야 하는 기존 직원입니다.

나중에 user1과 user2 모두 회사를 떠나서 그들의 액세스를 취소해야 합니다.

사용자를 생성하고 역할을 부여하려면 클러스터에 인증하는 사용자에게 createUser 및 grantRole에 대한 작업을 수행할 수 있는 연결된 역할이 있어야 합니다. 예를 들어 admin 및 userAdminAnyDatabase 역할은 모두 이러한 능력을 부여할 수 있습니다. 각 역할의 작업에 대한 자세한 내용은 역할 기반 액세스 제어를 사용한 데이터베이스 액세스 단원을 참조하십시오.

> **Note**
>
> Amazon DocumentDB에서 모든 사용자 및 역할 작업(예: `create`, `get`, `drop`, `grant`, `revoke`, `admin` 등)은 `admin` 데이터베이스에 대해 명령을 실행하는지 여부에 관계없이 데이터베이스에서 암시적으로 수행됩니다.

먼저 클러스터에 현재 사용자 및 역할이 무엇인지 이해하려면 다음 예제와 같이 `show users` 명령을 실행할 수 있습니다. 두 명의 사용자, 즉 `serviceadmin`과 클러스터의 기본 사용자가 표시됩니다. 이 두 사용자는 항상 존재하며 삭제할 수 없습니다. 자세한 내용은 Amazon DocumentDB 사용자 관리 단원을 참조하십시오.

```
show users
```

`user1`의 경우 다음 명령을 사용하여 전체 클러스터의 모든 데이터베이스에 대해 읽기 및 쓰기 액세스 권한이 있는 역할을 만듭니다.

```
db.createUser({user: "user1", pwd: "abc123", roles: [{role: "readWriteAnyDatabase", db:
 "admin"}]})
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
{
   "user":"user1",
   "roles":[
      {
         "role":"readWriteAnyDatabase",
         "db":"admin"
      }
   ]
}
```

`user2`의 경우 다음 명령을 사용하여 데이터베이스 `sample-database-1`에 대한 읽기 전용 액세스 권한이 있는 역할을 만듭니다.

```
db.createUser({user: "user2", pwd: "abc123", roles: [{role: "read", db: "sample-
database-1"}]})
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
{
   "user":"user2",
   "roles":[
      {
         "role":"read",
         "db":"sample-database-1"
      }
   ]
}
```

user3이 기존 사용자인 시나리오를 시뮬레이션하려면 먼저 사용자 user3를 만든 다음 user3에 새 역할을 할당합니다.

```
db.createUser({user: "user3", pwd: "abc123", roles: [{role: "readWrite", db: "sample-
database-1"}]})
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
{
   "user":"user3",
   "roles":[
      {
         "role":"readWrite",
         "db":"sample-database-1"
      }
   ]
}
```

이제 사용자 user3이 생성되었으므로 user3에 read, sample-database-2 역할을 할당합니다.

```
db.grantRolesToUser("user3", [{role: "read", db: "sample-database-2"}])
```

마지막으로, user1과 user2 모두 회사를 떠나고 클러스터에 대한 그들의 액세스가 취소되어야 합니다. 다음과 같이 사용자를 삭제하여 이 작업을 수행할 수 있습니다.

```
db.dropUser("user1")
db.dropUser("user2")
```

모든 사용자에게 적절한 역할이 있는지 확인하려면 다음 명령을 사용하여 모든 사용자를 나열할 수 있습니다.

```
show users
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
{
   "_id":"serviceadmin",
   "user":"serviceadmin",
   "db":"admin",
   "roles":[
      {
         "db":"admin",
         "role":"root"
      }
   ]
}
{
   "_id":"master-user",
   "user":"master-user",
   "db":"admin",
   "roles":[
      {
         "db":"admin",
         "role":"root"
      }
   ]
}
{
   "_id":"user3",
   "user":"user3",
   "db":"admin",
   "roles":[
      {
         "db":"sample-database-2",
         "role":"read"
      },
      {
         "db":"sample-database-1",
         "role":"readWrite"
      }
   ]
}
```

RBAC 사용자 정의 역할 시작하기

사용자 정의 역할을 시작하는 데 도움이 되도록 이 단원에서는 서로 다른 작업 기능을 가진 세 명의 사용자에 대한 역할을 생성하여 최소 권한을 적용하는 예제 시나리오를 설명합니다.

이 예에서 이하의 내용이 모두 적용됩니다.

- user1은 클러스터의 모든 데이터베이스를 보고 액세스할 수 있어야 하는 새로운 관리자입니다.
- user2은 동일한 클러스터에서 하나의 데이터베이스 sample-database-1에만 '찾기' 작업이 필요한 신입 직원입니다.
- user3은 동일한 클러스터에서 이전에 액세스할 수 없었던 다른 데이터베이스 sample-database-2 내의 특정 컬렉션, col2를 보고 액세스해야 하는 기존 직원입니다.
- user1의 경우 다음 명령을 사용하여 전체 클러스터의 모든 데이터베이스에 대해 읽기 및 쓰기 액세스 권한이 있는 역할을 만듭니다.

```
db.createUser(
 {
     user: "user1", pwd: "abc123",
     roles: [{role: "readWriteAnyDatabase", db: "admin"}]
 }
 )
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
{
   "user":"user1",
   "roles":[
      {
         "role":"readWriteAnyDatabase",
         "db":"admin"
      }
   ]
}
```

user2의 경우, 다음 명령을 사용하여 데이터베이스 sample-database-1의 모든 컬렉션에 대해 '찾기' 권한이 있는 역할을 생성합니다. 이 역할을 사용하면 연결된 모든 사용자가 찾기 쿼리만 실행할 수 있다는 점에 유의하세요.

```
db.createRole(
```

```
{
    role: "findRole",
    privileges: [
    {
        resource: {db: "sample-database-1", collection: ""}, actions: ["find"]
    }],
    roles: []
}
)
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
{
   "role":"findRole",
   "privileges":[
      {
         "resource":{
            "db":"sample-database-1",
            "collection":""
         },
         "actions":[
            "find"
         ]
      }
   ],
   "roles":[

   ]
}
```

그런 다음 사용자(user2)를 만들고 최근에 만든 역할을 findRole 사용자에게 연결합니다.

```
db.createUser(
{
    user: "user2",
    pwd: "abc123",
    roles: []
})

 db.grantRolesToUser("user2",["findRole"])
```

user3이 기존 사용자인 시나리오를 시뮬레이션하려면 먼저 사용자 user3를 만든 후, 다음 단계에서 user3에 새 역할을 할당할 collectionRole이라는 새 역할을 생성합니다.

이제 새 역할을 user3에 할당할 수 있습니다. 이 새 역할을 통해 user3이 sample-database-2 내의 하나의 특정 컬렉션 col2에 액세스를 삽입, 업데이트, 삭제하고 액세스 권한을 찾을 수 있습니다.

```
db.createUser(
{
    user: "user3",
    pwd: "abc123",
    roles: []
})

db.createRole(
{
    role: "collectionRole",
    privileges: [
    {
        resource: {db: "sample-database-2", collection: "col2"}, actions: ["find",
 "update", "insert", "remove"]
    }],
    roles: []
}
)
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
{
   "role":"collectionRole",
   "privileges":[
      {
         "resource":{
            "db":"sample-database-2",
            "collection":"col2"
         },
         "actions":[
            "find",
            "update",
            "insert",
            "remove"
         ]
      }
   ],
```

```
   "roles":[

   ]
}
```

이제 사용자 user3이 생성되었으므로 user3에 역할 collectionFind을 부여합니다.

```
db.grantRolesToUser("user3",["collectionRole"])
```

마지막으로, user1과 user2 모두 회사를 떠나고 클러스터에 대한 그들의 액세스가 취소되어야 합니다. 다음과 같이 사용자를 삭제하여 이 작업을 수행할 수 있습니다.

```
db.dropUser("user1")
db.dropUser("user2")
```

모든 사용자에게 적절한 역할이 있는지 확인하려면 다음 명령을 사용하여 모든 사용자를 나열할 수 있습니다.

```
show users
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
{
   "_id":"serviceadmin",
   "user":"serviceadmin",
   "db":"admin",
   "roles":[
      {
         "db":"admin",
         "role":"root"
      }
   ]
}
{
   "_id":"master-user",
   "user":"master-user",
   "db":"admin",
   "roles":[
      {
         "db":"admin",
         "role":"root"
```

```
      }
   ]
}
{
   "_id":"user3",
   "user":"user3",
   "db":"admin",
   "roles":[
      {
         "db":"admin",
         "role":"collectionRole"
      }
   ]
}
```

Amazon DocumentDB에 사용자로 연결

Amazon DocumentDB 클러스터에 연결할 때 특정 데이터베이스의 컨텍스트에서 연결합니다. 기본적으로 연결 문자열에 데이터베이스를 지정하지 않으면 `test` 데이터베이스의 컨텍스트에서 클러스터에 자동으로 연결됩니다. `test` 데이터베이스의 컬렉션에 대해 `insert` 및 `find` 같은 모든 컬렉션 레벨 명령이 발행됩니다.

컨텍스트에 있는 또는, 달리 말해, 명령이 발행되는 데이터베이스를 보려면 다음과 같이 mongo 쉘에서 db 명령을 사용하십시오.

쿼리:

```
db
```

출력:

```
test
```

기본 연결이 `test` 데이터베이스 컨텍스트에 있을 수 있지만 해당 연결과 연결된 사용자가 `test` 데이터베이스에서 작업을 수행할 수 있는 권한이 있는 것은 아닙니다. 앞의 예제 시나리오에서 `sample-database-1` 데이터베이스에 대한 `readWrite` 역할이 있는 사용자 `user3`으로 인증하는 경우 연결의 기본 컨텍스트는 `test` 데이터베이스입니다. 그러나 `test` 데이터베이스의 컬렉션에 문서를 삽입하려고 하면 권한 부여 실패 오류 메시지가 나타납니다. 아래와 같이 해당 사용자가 해당 데이터베이스에서 해당 명령을 수행할 권한이 없기 때문입니다.

쿼리:

```
db
```

출력:

```
test
```

쿼리:

```
db.col.insert({x:1})
```

출력:

```
WriteCommandError({ "ok" : 0, "code" : 13, "errmsg" : "Authorization failure" })
```

sample-database-1 데이터베이스에 대한 연결 컨텍스트를 변경하면 사용자가 그렇게 할 수 있는 권한을 가진 컬렉션에 쓸 수 있습니다.

쿼리:

```
use sample-database-1
```

출력:

```
switched to db sample-database-1
```

쿼리:

```
db.col.insert({x:1})
```

출력:

```
WriteResult({ "nInserted" : 1})
```

특정 사용자를 사용하여 클러스터에 인증하는 경우 연결 문자열에서 데이터베이스를 지정할 수도 있습니다. 이렇게 하면 사용자가 admin 데이터베이스에 인증된 후 use 명령을 수행할 필요가 없습니다.

다음 연결 문자열은 admin 데이터베이스에 대해 사용자를 인증하지만 연결 컨텍스트는 sample-database-1 데이터베이스에 대한 것입니다.

```
mongo "mongodb://user3:abc123@sample-cluster.node.us-east-1.docdb.amazonaws.com:27017/
sample-database-2"
```

공통 명령

이 단원에서는 Amazon DocumentDB에서 역할 기반 액세스 제어를 사용하는 공통 명령의 예를 제공합니다. 사용자 및 역할을 만들고 수정하려면 admin 데이터베이스의 컨텍스트에 있어야 합니다. use admin 명령을 사용하여 admin 데이터베이스로 전환할 수 있습니다.

> **Note**
>
> 사용자 및 역할에 대한 수정은 admin 데이터베이스에서 암시적으로 발생합니다. 모든 데이터베이스에서 범위가 지정된 역할(예: readAnyDatabase)을 가진 사용자를 생성하려면 사용자를 생성할 때 admin 데이터베이스(즉 use admin) 컨텍스트에 있거나 사용자를 생성할 때 역할에 대한 데이터베이스를 명백하게 명시해야 합니다(이 단원의 예제 2 참조).

예제 1: 데이터베이스foo에 대한 read 역할을 가진 사용자를 만듭니다.

```
db.createUser({user: "readInFooBar", pwd: "abc123", roles: [{role: "read", db:
 "foo"}]})
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
{
   "user":"readInFooBar",
   "roles":[
      {
         "role":"read",
         "db":"foo"
      }
   ]
}
```

예제 2: 모든 데이터베이스에서 읽기 권한이 있는 사용자를 만듭니다.

```
db.createUser({user: "readAllDBs", pwd: "abc123", roles: [{role: "readAnyDatabase", db:
 "admin"}]})
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
{
   "user":"readAllDBs",
   "roles":[
      {
         "role":"readAnyDatabase",
         "db":"admin"
      }
   ]
}
```

예제 3: 새 데이터베이스의 기존 사용자에게 read 역할을 부여합니다.

```
db.grantRolesToUser("readInFooBar", [{role: "read", db: "bar"}])
```

예제 4: 사용자의 역할을 업데이트합니다.

```
db.updateUser("readInFooBar", {roles: [{role: "read", db: "foo"}, {role: "read", db:
 "baz"}]})
```

예제 5: 사용자의 데이터베이스에 대한 액세스 권한을 취소합니다.

```
db.revokeRolesFromUser("readInFooBar", [{role: "read", db: "baz"}])
```

예제 6: 기본 제공 역할을 설명합니다.

```
db.getRole("read", {showPrivileges:true})
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
{
   "role":"read",
   "db":"sample-database-1",
   "isBuiltin":true,
   "roles":[

   ],
   "inheritedRoles":[

   ],
```

```
   "privileges":[
      {
         "resource":{
            "db":"sample-database-1",
            "collection":""
         },
         "actions":[
            "changeStream",
            "collStats",
            "dbStats",
            "find",
            "killCursors",
            "listCollections",
            "listIndexes"
         ]
      }
   ],
   "inheritedPrivileges":[
      {
         "resource":{
            "db":"sample-database-1",
            "collection":""
         },
         "actions":[
            "changeStream",
            "collStats",
            "dbStats",
            "find",
            "killCursors",
            "listCollections",
            "listIndexes"
         ]
      }
   }
```

예제 7: 클러스터에서 사용자를 삭제합니다.

```
db.dropUser("readInFooBar")
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
true
```

예제 8: 특정 컬렉션에 대한 읽기 및 쓰기 권한이 있는 역할 생성합니다.

```
db.createRole(
{
    role: "collectionRole",
    privileges: [
    {
        resource: {db: "sample-database-2", collection: "col2"}, actions: ["find",
 "update", "insert", "remove"]
    }],
    roles: []
}
)
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
{
   "role":"collectionRole",
   "privileges":[
      {
         "resource":{
            "db":"sample-database-2",
            "collection":"col2"
         },
         "actions":[
            "find",
            "update",
            "insert",
            "remove"
         ]
      }
   ],
   "roles":[

   ]
}
```

예제 9: 사용자를 생성하고 사용자 정의 역할을 할당합니다.

```
db.createUser(
{
    user: "user3",
    pwd: "abc123",
```

```
    roles: []
})

db.grantRolesToUser("user3",["collectionRole"])
```

예제 10: 사용자 정의 역할에 추가 권한을 부여합니다.

```
db.grantPrivilegesToRole(
  "collectionRole",
  [
    {
      resource: { db: "sample-database-1", collection: "col1" },
      actions: ["find", "update", "insert", "remove"]
    }
  ]
)
```

예제 11: 사용자 정의 역할에서 권한을 제거합니다.

```
db.revokePrivilegesFromRole(
  "collectionRole",
  [
    {
      resource: { db: "sample-database-1", collection: "col2" },
      actions: ["find", "update", "insert", "remove"]
    }
  ]
)
```

예제 12: 기존 사용자 정의 역할을 업데이트합니다.

```
db.updateRole(
  "collectionRole",
  {
    privileges: [
    {
        resource: {db: "sample-database-3", collection: "sample-collection-3"},
 actions: ["find", "update", "insert", "remove"]
    }],
    roles: []
  }
)
```

기능적 차이

Amazon DocumentDB에서, 사용자 및 역할 정의는 admin 데이터베이스에 저장되고 사용자는 admin 데이터베이스에 대해 인증됩니다. 이 기능은 MongoDB Community Edition과 다르지만 MongoDB Atlas와 일치합니다.

Amazon DocumentDB는 또한 클러스터의 컬렉션 내에서 발생하는 변경 이벤트 시퀀스를 시간 순서대로 제공하는 변경 스트림을 지원합니다. `listChangeStreams` 작업은 클러스터 수준에서(즉, 모든 데이터베이스에 걸쳐) 적용되고 `modifyChangeStreams` 작업은 데이터베이스 수준 및 클러스터 수준에서 적용됩니다.

Limits

다음 표에는 Amazon DocumentDB의 역할 기반 액세스 제어에 대한 제한이 포함되어 있습니다.

설명	Limit
클러스터당 사용자 수	1000
사용자와 연결된 역할 수	1000
사용자 정의 역할 수	100
권한과 관련된 리소스 수	100

역할 기반 액세스 제어를 사용한 데이터베이스 액세스

역할 기반 액세스 제어를 사용하면 사용자를 생성하고 사용자에게 하나 이상의 역할을 부여하여 사용자가 데이터베이스 또는 클러스터에서 수행할 수 있는 작업을 결정할 수 있습니다.

다음은 Amazon DocumentDB에서 현재 지원되는 기본 제공 역할 목록입니다.

> **Note**
>
> Amazon DocumentDB 4.0 및 5.0에서는, `ListCollection` 및 `ListDatabase` 명령을 사용하면 선택적으로 `authorizedCollections` 및 `authorizedDatabases` 매개변수를 사용하여 사용자가 각각 `listCollections` 및 `listDatabase` 역할을 요구하지 않고도 액세스

권한이 있는 컬렉션과 데이터베이스를 나열할 수 있습니다. 또한, 이제 사용자는 `KillCursor` 역할이 없이도 자신의 커서를 죽일 수도 있습니다.

Database user

역할 이름	설명	작업

`read`	사용자에게 지정된 데이터베이스에 대한 읽기 액세스 권한을 부여합니다.	changeStreams
		`collStats`
		`dbStats`
		`find`
		`killCursors`
		`listIndexes`
		`listCollections`

`readWrite`	사용자에게 지정된 데이터베이스에 대한 읽기 및 쓰기 액세스 권한을 부여합니다.	`read` 권한의 모든 작업.
		`createCollection`
		`dropCollection`
		`createIndex`
		`dropIndex`
		`insert`
		`killCursors`
		`listIndexes`

역할 이름	설명	작업
		listCollections remove update

Cluster user

역할 이름	설명	작업
readAnyDatabase	사용자에게 클러스터의 모든 데이터베이스에 대한 읽기 권한을 부여합니다.	read 권한의 모든 작업. listChangeStreams listDatabases
readWriteAnyDatabase	사용자에게 클러스터의 모든 데이터베이스에 대한 읽기 및 쓰기 액세스 권한을 부여합니다.	readWrite 권한의 모든 작업. listChangeStreams listDatabases
userAdminAnyDatabase	사용자에게 지정된 데이터베이스에 대해 사용자가 갖는 역할 또는 권한을 할당하고 수정할 수 있는 기능을 부여합니다.	changeCustomData changePassword createUser dropRole dropUser

역할 이름	설명	작업
		grantRole listDatabases revokeRole viewRole viewUser
dbAdminAnyDatabase	사용자에게 지정된 데이터베이스에서 데이터베이스 관리 역할을 수행할 수 있는 기능을 부여합니다.	dbAdmin 권한의 모든 작업. dropCollection listDatabases listChangeStreams modifyChangeStreams

Superuser

역할 이름	설명	작업
root	사용자에게 readWriteAnyDatabase , dbAdminAnyDatabase , userAdminAnyDatabase , clusterAdmin , restore및 backup 역할이 결합된 모든 역할의 리소스 및 작업에 대한 액세스 권한을 부여합니다.	readWriteAnyDatabase , dbAdminAnyDatabase , userAdminAnyDatabase , clusterAdmin , restore 및 backup의 모든 작업

Database administrator

역할 이름	설명	작업
dbAdmin	사용자에게 지정된 데이터베이스에서 관리 작업을 수행할 수 있는 기능을 부여합니다.	bypassDocumentValidation collMod collStats createCollection createIndex dropCollection dropDatabase dropIndex dbStats find killCursors listIndexes listCollections modifyChangeStreams
dbOwner	사용자에게 dbAdmin 및 readWrite 역할을 결합하여 지정된 데이터베이스에서 모든 관리 작업을 수행할 수 있는 기능을 부여합니다.	dbAdmin 및 readWrite 의 모든 작업.

Cluster administrator

역할 이름	설명	작업
clusterAdmin	clusterManager , clusterMonitor 및 hostManager 역할을 결합하여 사용자에게 가장 큰 클러스터 관리 액세스 권한을 부여합니다.	clusterManager , clusterMonitor 및 hostManager 의 모든 작업. listChangeStreams dropDatabase modifyChangeStreams
clusterManager	사용자에게 지정된 클러스터에서 관리 및 모니터링 작업을 수행할 수 있는 기능을 부여합니다.	listChangeStreams listSessions modifyChangeStreams replSetGetConfig
clusterMonitor	사용자에게 모니터링 도구에 대한 읽기 전용 액세스 기능을 부여합니다.	collStats dbStats find getParameter hostInfo indexStats killCursors

역할 이름	설명	작업
		listChangeStreams listCollections listDatabases listIndexes listSessions replSetGetConfig serverStatus top
hostManager	사용자에게 서버를 모니터링하고 관리할 수 있는 기능을 부여합니다.	auditConfigure killCursors killAnyCursor killAnySession killop

Backup administrator

역할 이름	설명	작업
backup	사용자에게 데이터를 백업하는 데 필요한 액세스 권한을 부여합니다.	getParameter insert find

역할 이름	설명	작업

		`listChangeStreams`
		`listCollections`
		`listDatabases`
		`listIndexes`
		`update`

역할 이름	설명	작업
restore	사용자에게 데이터를 복원하는 데 필요한 액세스 권한을 부여합니다.	bypassDocumentValidation changeCustomData changePassword collMod createCollection createIndex createUser dropCollection dropRole dropUser getParameter grantRole find insert listCollections modifyChangeStreams revokeRole

역할 이름	설명	작업

		`remove`
		
		`viewRole`
		
		`viewUser`
		
		`update`

Amazon DocumentDB의 로깅 및 모니터링

Amazon DocumentDB(MongoDB 호환)은 모니터링을 통해 Amazon DocumentDB 클러스터와 인스턴스의 상태와 성능을 평가할 수 있는 Amazon CloudWatch 지표를 다양하게 제공합니다. Amazon DocumentDB 콘솔, , Amazon CloudWatch 콘솔, CloudWatch API 등 다양한 도구를 사용하여 AWS CLI Amazon DocumentDB 지표를 볼 수 있습니다. Amazon CloudWatch 모니터링에 대한 자세한 내용은 Amazon DocumentDB 모니터링 섹션을 참조하세요.

Amazon CloudWatch 지표 외에도 프로파일러를 사용하여 클러스터에서 수행된 작업의 실행 시간 및 세부 정보를 로깅할 수 있습니다. 프로파일러는 개별 쿼리 성능 및 전체 클러스터 성능을 개선하기 위해 클러스터에서 가장 느린 작업을 모니터링하는 경우에 유용합니다. 사용 설정하면 작업이 Amazon CloudWatch Logs 에 기록되며, CloudWatch Insight를 사용하여 Amazon DocumentDB 프로파일링 데이터를 분석, 모니터링 및 보관할 수 있습니다. 자세한 내용은 Amazon DocumentDB 작업 프로파일링 단원을 참조하십시오.

또한 Amazon DocumentDB는 Amazon DocumentDB(MongoDB 호환)에서 사용자, 역할 또는 서비스가 수행한 작업에 대한 레코드를 제공하는 AWS 서비스인 AWS CloudTrail과 통합됩니다. CloudTrail은 Amazon DocumentDB의 호출 및 Amazon DocumentDB Amazon DocumentDB SDK에 대한 코드 호출을 포함하여 Amazon DocumentDB AWS Management Console 에 대한 모든 AWS CLI API 호출을 이벤트로 캡처합니다. 자세한 내용은 를 사용하여 Amazon DocumentDB API 호출 로깅 AWS CloudTrail 단원을 참조하십시오.

Amazon DocumentDB에서는 클러스터에서 수행된 이벤트를 감사할 수 있습니다. 기록되는 이벤트의 예제로는 성공 또는 실패한 인증 시도, 데이터베이스에서 모음 삭제 또는 인덱스 생성이 있습니다. 기본적으로 Amazon DocumentDB에서 감사는 비활성화되므로 사용자가 이 기능을 선택해야 합니다. 자세한 내용은 Amazon DocumentDB 이벤트 감사 단원을 참조하십시오.

Amazon DocumentDB TLS 인증서 업데이트

주제

- 애플리케이션 및 Amazon DocumentDB 클러스터 업데이트
- 자동 서버 인증서 교체
- 문제 해결
- FAQ

Amazon DocumentDB 클러스터에 대한 인증 기관(CA) 인증서는 2024년 8월에 업데이트되었습니다. 전송 계층 보안(TLS)이 활성화(기본 설정)된 Amazon DocumentDB 클러스터를 사용하고 클라이언트 애플리케이션 및 서버 인증서를 교체하지 않은 경우, 애플리케이션과 Amazon DocumentDB 클러스터 간의 연결 문제를 완화하려면 다음 단계를 수행해야 합니다.

- 1단계: 새 CA 인증서 다운로드 및 애플리케이션 업데이트
- 2단계: 서버 인증서 업데이트

CA 및 서버 인증서는 Amazon DocumentDB에 대한 표준 유지 관리 및 보안 모범 사례의 일부로 업데이트되었습니다. 클라이언트 애플리케이션은 새 CA 인증서를 신뢰할 수 있는 스토어에 추가해야 하며, 이 만료 날짜 이전에 새 CA 인증서를 사용하도록 기존 Amazon DocumentDB 인스턴스를 업데이트해야 합니다.

애플리케이션 및 Amazon DocumentDB 클러스터 업데이트

이 섹션의 단계에 따라 애플리케이션의 CA 인증서 번들(1단계)과 클러스터의 서버 인증서(2단계)를 업데이트합니다. 프로덕션 환경에 변경 사항을 적용하기 전에 개발 또는 스테이징 환경에서 이러한 단계를 테스트하는 것이 좋습니다.

> **Note**
>
> Amazon DocumentDB 클러스터가 있는 각 AWS 리전 에서 1단계와 2단계를 완료해야 합니다.

1단계: 새 CA 인증서 다운로드 및 애플리케이션 업데이트

새 CA 인증서를 다운로드하고 응용 프로그램을 업데이트하여 새 CA 인증서를 사용하여 Amazon DocumentDB에 TLS 연결을 만듭니다. https://truststore.pki.rds.amazonaws.com/global/global-

bundle.pem에서 새 CA 인증서 번들을 다운로드합니다. 그러면 `global-bundle.pem`라는 파일이 다운로드됩니다.

> **Note**
>
> 이전 CA 인증서(`rds-ca-2019-root.pem`)와 새 CA 인증서(`rds-ca-rsa2048-g1`, `rds-ca-rsa4096-g1`, `rds-ca-ecc384-g1`)가 모두 포함된 키 스토어에 액세스하는 경우 키 스토어에서 `global-bundle` 항목을 선택하는지 확인하세요.

```
wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem
```

그런 다음 새 인증서 번들을 사용하도록 애플리케이션을 업데이트합니다. 새 CA 번들에는 이전 CA 인증서(rds-ca-2019)와 새 CA 인증서(rds-ca-rsa2048-g1, rds-ca-rsa4096-g1, rds-ca-ecc384-g1)가 모두 포함됩니다. 새 CA 번들에 두 CA 인증서가 모두 포함되어 있으므로 애플리케이션과 클러스터를 두 단계로 업데이트할 수 있습니다.

Java 애플리케이션의 경우 새 CA 인증서로 새 신뢰 스토어를 생성해야 합니다. 지침은 TLS가 활성화된 상태에서 연결 주제의 Java 탭을 참조하세요.

애플리케이션에서 최신 CA 인증서 번들을 사용하고 있는지 확인하려면 최신 CA 번들을 사용하고 있는지 어떻게 확인할 수 있습니까? 섹션을 참조하세요. 애플리케이션에서 최신 CA 인증서 번들을 이미 사용하고 있는 경우 2단계로 건너뛸 수 있습니다.

애플리케이션에 CA 번들을 사용하는 경우의 예는 전송 중 데이터 암호화 및 TLS가 활성화된 상태에서 연결 섹션을 참조하세요.

> **Note**
>
> 현재 MongoDB Go Driver 1.2.1은 `sslcertificateauthorityfile`에서 하나의 CA 서버 인증서만 허용합니다. TLS가 활성화될 때 Go를 사용하여 Amazon DocumentDB에 연결하는 방법은 TLS가 활성화된 상태에서 연결 섹션을 참조하세요.

2단계: 서버 인증서 업데이트

응용프로그램이 새 CA 번들을 사용하도록 업데이트되면, 다음 단계는 Amazon DocumentDB 클러스터의 각 인스턴스를 수정하여 서버 인증서를 업데이트하는 것입니다. 새 서버 인증서를 사용하도록 인스턴스를 수정하려면 다음 지침을 참조하세요.

Amazon DocumentDB는 DB 인스턴스의 DB 서버 인증서에 서명할 수 있는 다음 CA를 제공합니다.

- rds-ca-ecc384-g1—ECC 384 프라이빗 키 알고리즘과 SHA384 서명 알고리즘을 갖춘 인증 기관을 사용합니다. 이 CA는 자동 서버 인증서 교체를 지원합니다. 이는 현재 Amazon DocumentDB 4.0 및 5.0에서만 지원됩니다.
- rds-ca-rsa2048-g1 - 대부분의 AWS 리전에서 RSA 2048 프라이빗 키 알고리즘 및 SHA256 서명 알고리즘이 있는 인증 기관을 사용합니다. 이 CA는 자동 서버 인증서 교체를 지원합니다.
- rds-ca-rsa4096-g1—RSA 4096 프라이빗 키 알고리즘과 SHA256 서명 알고리즘을 갖춘 인증 기관을 사용합니다. 이 CA는 자동 서버 인증서 교체를 지원합니다.

> **Note**
>
> 를 사용하는 경우 describe-certificates를 사용하여 위에 나열된 인증 기관의 유효성을 확인할 AWS CLI수 있습니다.

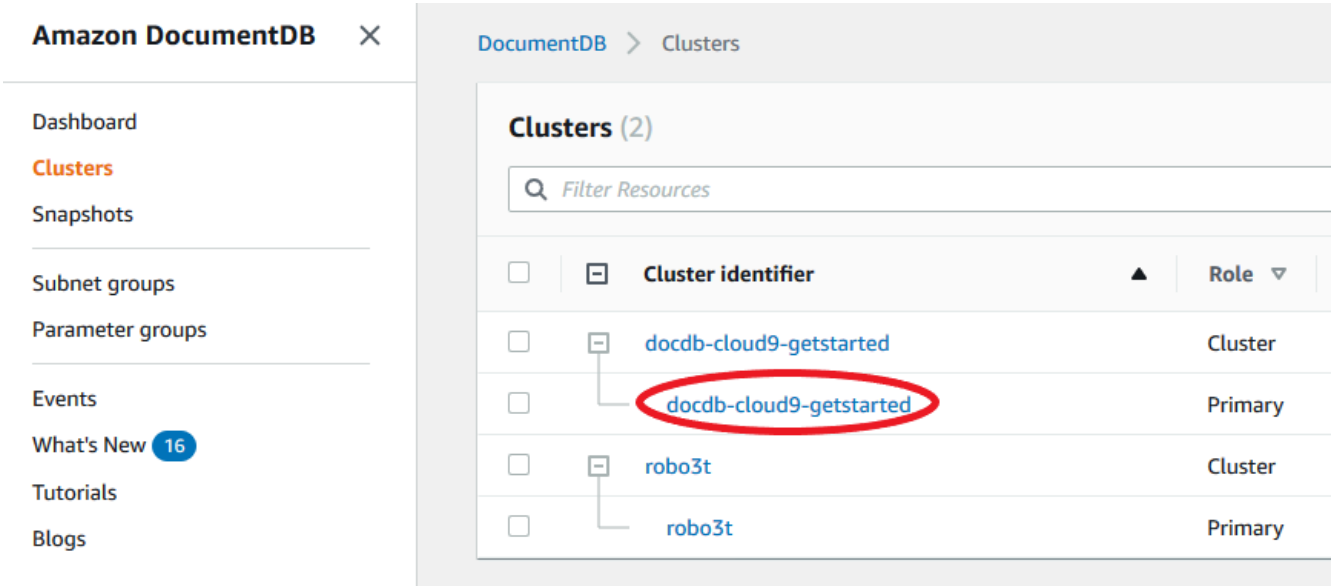
이러한 CA 인증서는 지역 및 글로벌 인증서 번들에 포함되어 있습니다. rds-ca-rsa2048-g1, rds-ca-rsa4096-g1, rds-ca-ecc384-g1 CA를 데이터베이스에 사용하면 Amazon DocumentDB가 데이터베이스에서 DB 서버 인증서를 관리합니다. Amazon DocumentDB는 DB 서버 인증서를 만료되기 전에 자동으로 교체합니다.

> **Note**
>
> 클러스터가 다음 엔진 패치 버전에서 실행 중인 경우 Amazon DocumentDB는 인증서 교체를 위해 재부팅할 필요가 없습니다.
>
> - Amazon DocumentDB 3.6: 1.0.208662 이상
> - Amazon DocumentDB 4.0: 2.0.10179 이상
> - Amazon DocumentDB 5.0: 3.0.4780 이상
>
> 현재 Amazon DocumentDB 엔진 패치 버전을 확인하려면 `db.runCommand({getEngineVersion: 1})`의 명령을 실행합니다. 서버 인증서를 업데이트하기 전에 1단계 항목을 완료했는지 확인합니다.

Using the AWS Management Console

AWS Management Console을 사용하여 기존 Amazon DocumentDB 인스턴스의 이전 서버 인증서를 식별하고 교체하기 위해 다음 단계를 완료하세요.

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb://https://https://https://://https://httpsAmazon DocumentDBhttps://://https://https://https://://https://https://https://https://://
2. 화면 오른쪽 상단의 리전 목록에서 클러스터가 있는 AWS 리전 를 선택합니다.
3. 콘솔 왼쪽의 탐색 창에서 클러스터를 선택합니다.
4. 이전 서버 인증서 (`rds-ca-2019`) 에 아직 남아 있는 인스턴스를 식별해야 할 수도 있습니다. 클러스터 테이블 맨 오른쪽에 있는 인증 기관 열에서 이 작업을 수행할 수 있습니다.
5. 클러스터 테이블의 맨 왼쪽에 클러스터 식별자 열이 보일 것입니다. 인스턴스는 아래 스크린샷과 비슷하게 클러스터 아래에 나열됩니다.



6. 관심 있는 인스턴스의 왼쪽에 있는 박스를 선택합니다.
7. 작업을 선택한 다음 수정을 선택합니다.
8. 인증 기관에서 이 인스턴스에 대한 새 서버 인증서(`rds-ca-rsa2048-g1`)를 선택합니다.
9. 다음 페이지에서 변경 사항 요약을 볼 수 있습니다. 연결이 중단되지 않도록 인스턴스를 수정하기 전에 애플리케이션이 최신 인증서 CA 번들을 사용하고 있는지 확인하는 추가 알림이 있습니다.
10. 다음 유지 관리 기간 중에 수정 사항을 적용하거나 즉시 적용하도록 선택할 수 있습니다. 서버 인증서를 즉시 수정하려는 경우 즉시 적용 옵션을 사용합니다.

11. Modify instance(인스턴스 수정)를 선택하여 업데이트를 완료합니다.

Using the AWS CLI

AWS CLI을 사용하여 기존 Amazon DocumentDB 인스턴스의 이전 서버 인증서를 식별하고 교체하기 위해 다음 단계를 완료하세요.

1. 인스턴스를 즉시 수정하려면 클러스터의 각 인스턴스에 대해 다음 명령을 실행합니다.

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier>
 --ca-certificate-identifier rds-ca-rsa2048-g1 --apply-immediately
```

2. 클러스터의 다음 유지 관리 기간 동안 새 CA 인증서를 사용하도록 클러스터의 인스턴스를 수정하려면 클러스터의 각 인스턴스에 대해 다음 명령을 실행합니다.

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier>
 --ca-certificate-identifier rds-ca-rsa2048-g1 --no-apply-immediately
```

자동 서버 인증서 교체

Amazon DocumentDB는 자동 서버 인증서 교체를 지원합니다. 서버 인증서는 각 클러스터 인스턴스에 발급된 리프 인증서입니다. 루트 CA 인증서와 달리 서버 인증서의 유효 기간은 짧으며(12개월) Amazon DocumentDB는 사용자의 조치 없이 자동으로 교체를 처리합니다. Amazon DocumentDB는이 자동 교체에 동일한 루트 CA를 사용하므로 새 CA 번들을 다운로드할 필요가 없습니다.

> **Important**
>
> Amazon DocumentDB 클러스터에 연결할 때는 각 서버 인증서를 직접 신뢰하는 대신 루트 CA 번들을 신뢰하는 것이 좋습니다. 이렇게 하면 서버 인증서가 교체된 후 연결 오류가 방지됩니다. TLS가 활성화된 상태에서 연결을(를) 참조하세요.

Amazon DocumentDB는 원하는 유지 관리 기간에서 서버 인증서의 수명이 짧을 때 서버 인증서를 교체하려고 시도합니다. 새 서버 인증서는 12개월 동안 유효합니다.

`describe-db-engine-versions` 명령을 사용하고 `SupportsCertificateRotationWithoutRestart` 플래그를 검사하여 엔진 버전이 다시 시작하지 않고 인증서 교체를 지원하는지 확인합니다.

> **Note**
>
> Amazon DocumentDB는 클러스터가 다음 엔진 패치 버전에서 실행 중인 경우 재시작 없이 서버 인증서 교체를 지원합니다.
>
> - Amazon DocumentDB 3.6: 1.0.208662 이상
> - Amazon DocumentDB 4.0: 2.0.10179 이상
> - Amazon DocumentDB 5.0: 3.0.4780 이상
>
> 다음 명령을 실행하여 현재 Amazon DocumentDB 엔진 패치 버전을 확인할 수 있습니다 `db.runCommand({getEngineVersion: 1})`.
> 이전 엔진 패치 버전을 사용하는 경우 Amazon DocumentDB는 서버 인증서를 교체하고 원하는 유지 관리 기간에 데이터베이스 재시작 이벤트를 예약합니다.

문제 해결

인증서 교체의 일부로 클러스터에 연결하는 데 문제가 있는 경우 다음을 권장합니다.

- 클라이언트가 최신 인증서 번들을 사용하고 있는지 확인합니다. 최신 CA 번들을 사용하고 있는지 어떻게 확인할 수 있습니까?을(를) 참조하세요.
- 인스턴스가 최신 인증서를 사용하고 있는지 확인합니다. 내 Amazon DocumentDB 인스턴스가 이전/새 서버 인증서를 사용하고 있는지 어떻게 알 수 있습니까?을(를) 참조하세요.
- 애플리케이션에서 최신 인증서 CA를 사용하고 있는지 확인합니다. Java 및 Go와 같은 일부 드라이버는 인증서 번들에서 신뢰 저장소로 여러 인증서를 가져오려면 추가 코드가 필요합니다. TLS를 사용하여 Amazon DocumentDB에 연결하는 방법에 대한 자세한 내용은 Amazon DocumentDB에 프로그래밍 방식으로 연결을 참조하세요.
- 고객 지원 센터에 문의.. 질문이나 문제가 있으면 지원에 연락하세요.

FAQ

다음은 TLS 인증서에 대한 몇 가지 일반적인 질문에 대한 답변입니다.

질문이나 문제가 있는 경우 어떻게 해야 합니까?

질문이나 문제가 있으면 지원에 연락하세요.

Amazon DocumentDB 클러스터에 연결하는 데 TLS를 사용했는지 어떻게 알 수 있습니까?

클러스터의 클러스터 파라미터 그룹에 대한 `tls` 파라미터를 확인하여 클러스터에서 TLS를 사용하고 있는지 확인할 수 있습니다. `tls` 파라미터가 `enabled`로 설정된 경우 TLS 인증서를 사용하여 클러스터에 연결되어 있는 것입니다. 자세한 내용은 Amazon DocumentDB 클러스터 파라미터 그룹 관리 단원을 참조하십시오.

CA와 서버 인증서를 업데이트하는 이유는 무엇입니까?

Amazon DocumentDB CA 및 서버 인증서는 Amazon DocumentDB에 대한 표준 유지 관리 및 보안 모범 사례의 일부로 업데이트되고 있습니다.

만료일까지 조치를 취하지 않으면 어떻게 됩니까?

TLS를 사용하여 만료된 CA 인증서로 Amazon DocumentDB 클러스터에 연결하는 경우 TLS를 통해 연결하는 애플리케이션은 더 이상 Amazon DocumentDB 클러스터와 통신할 수 없습니다.

Amazon DocumentDB는 만료가 이뤄질 때까지 데이터베이스 인증서를 자동으로 교체하지 않습니다. 만료일 이전 또는 이후에 새 CA 인증서를 사용하려면 애플리케이션과 클러스터를 업데이트해야 합니다.

내 Amazon DocumentDB 인스턴스가 이전/새 서버 인증서를 사용하고 있는지 어떻게 알 수 있습니까?

여전히 이전 서버 인증서를 사용하는 Amazon DocumentDB 인스턴스를 식별하려면 Amazon DocumentDB AWS Management Console 또는를 사용할 수 있습니다 AWS CLI.

사용 AWS Management Console

이전 인증서를 사용하는 클러스터의 인스턴스를 식별하려면

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb://https://httpsAmazon DocumentDBhttps:///:///https:///:///https:///:///https:///:///https:///:///:///https:///:///:///:///https://https://

2. 화면 오른쪽 상단의 리전 목록에서 인스턴스가 있는 AWS 리전 를 선택합니다.
3. 콘솔 왼쪽의 탐색 창에서 클러스터를 선택합니다.
4. 인증 기관 열(테이블의 맨 오른쪽 근처)에는 아직 이전 서버 인증서(`rds-ca-2019`)에 있는 인스턴스, 그리고 새 서버 인증서(`rds-ca-rsa2048-g1`)에 있는 인스턴스가 모두 표시됩니다.

사용 AWS CLI

이전 서버 인증서를 사용하는 클러스터의 인스턴스를 식별하려면 `describe-db-clusters` 명령을 다음과 함께 사용합니다.

```
aws docdb describe-db-instances \
    --filters Name=engine,Values=docdb \
    --query 'DBInstances[*].
{CertificateVersion:CACertificateIdentifier,InstanceID:DBInstanceIdentifier}'
```

Amazon DocumentDB 클러스터에서 개별 인스턴스를 수정하여 서버 인증서를 업데이트하려면 어떻게 해야 합니까?

특정 클러스터의 모든 인스턴스에 대한 서버 인증서를 동시에 업데이트하는 것이 좋습니다. 클러스터의 인스턴스를 수정하기 위해 콘솔 또는 AWS CLI를 사용할 수 있습니다.

> **Note**
>
> 서버 인증서를 업데이트하기 전에 1단계를 완료했는지 확인합니다.

사용 AWS Management Console

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb://https://httpsAmazon DocumentDBhttps://://https://://https://://https://://https://://https://://://https://://://://https://https://
2. 화면 오른쪽 상단의 리전 목록에서 클러스터가 있는 AWS 리전 를 선택합니다.
3. 콘솔 왼쪽의 탐색 창에서 클러스터를 선택합니다.
4. 인증 기관 열(테이블의 맨 오른쪽 근처)에는 아직 이전 서버 인증서(`rds-ca-2019`)에 있는 인스턴스가 표시됩니다.
5. 클러스터 테이블의 클러스터 식별자에서 수정할 인스턴스를 선택합니다.
6. 작업을 선택한 다음 수정을 선택합니다.
7. 인증 기관에서 이 인스턴스에 대한 새 서버 인증서(`rds-ca-rsa2048-g1`)를 선택합니다.
8. 다음 페이지에서 변경 사항 요약을 볼 수 있습니다. 연결이 중단되지 않도록 인스턴스를 수정하기 전에 애플리케이션이 최신 인증서 CA 번들을 사용하고 있는지 확인하는 추가 알림이 있습니다.
9. 다음 유지 관리 기간 중에 수정 사항을 적용하거나 즉시 적용하도록 선택할 수 있습니다.

10. Modify instance(인스턴스 수정)를 선택하여 업데이트를 완료합니다.

사용 AWS CLI

AWS CLI을 사용하여 기존 Amazon DocumentDB 인스턴스의 이전 서버 인증서를 식별하고 교체하기 위해 다음 단계를 완료하세요.

1. 인스턴스를 즉시 수정하려면 클러스터의 각 인스턴스에 대해 다음 명령을 실행합니다.

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier> --
ca-certificate-identifier rds-ca-rsa2048-g1 --apply-immediately
```

2. 클러스터의 다음 유지 관리 기간 동안 새 CA 인증서를 사용하도록 클러스터의 인스턴스를 수정하려면 클러스터의 각 인스턴스에 대해 다음 명령을 실행합니다.

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier> --
ca-certificate-identifier rds-ca-rsa2048-g1 --no-apply-immediately
```

기존 클러스터에 새 인스턴스를 추가하면 어떻게 됩니까?

생성된 모든 새 인스턴스는 이전 서버 인증서를 사용하며, 이전 CA 인증서를 사용하는 TLS 연결이 필요합니다. 2024년 1월 25일 이후에 생성된 모든 새 Amazon DocumentDB 인스턴스는 기본적으로 새 인증서 rds-ca-rsa2048-g1을 사용합니다.

클러스터에 인스턴스 대체 또는 장애 조치가 있는 경우 어떻게 됩니까?

클러스터에 인스턴스 대체가 있는 경우, 생성된 새 인스턴스는 인스턴스가 이전에 사용했던 것과 동일한 서버 인증서를 계속 사용합니다. 모든 인스턴스에 대한 서버 인증서를 동시에 업데이트하는 것이 좋습니다. 클러스터에 장애 조치가 발생하면 새로운 기본 서버 인증서가 사용됩니다.

TLS를 사용하여 클러스터에 연결하지 않는 경우에도 각 인스턴스를 업데이트해야 합니까?

TLS를 활성화하는 것이 좋습니다. TLS를 활성화하지 않은 경우에도 TLS를 사용하여 향후 클러스터에 연결할 계획인 경우 Amazon DocumentDB 인스턴스에서 인증서를 교체하는 것이 좋습니다. Amazon DocumentDB 클러스터 연결에 TLS를 사용할 계획이 없다면, 별도로 필요한 작업이 없습니다.

TLS를 사용하여 클러스터에 연결하지 않고 나중에 연결할 계획이라면 어떻게 해야 합니까?

2024년 1월 이전에 클러스터를 생성한 경우 이전 섹션의 1단계 및 2단계를 수행하여 애플리케이션에서 업데이트된 CA 번들을 사용하고 있는지 및 각 Amazon DocumentDB 인스턴스가 최신 서버 인증서를 사용 중인지 확인합니다. 2024년 1월 25일 이후 클러스터를 생성한 경우, 클러스터에 이미 최신 서버 인증서(rds-ca-rsa2048-g1)가 있습니다. 애플리케이션이 최신 CA 번들을 사용하고 있는지 확인하려면 TLS를 사용하여 클러스터에 연결하지 않는 경우에도 각 인스턴스를 업데이트해야 합니까? 단원을 참조하십시오.

2024년 8월 이후로 기한을 연장할 수 있습니까?

애플리케이션이 TLS를 통해 연결되는 경우, 기한을 연장할 수 없습니다.

최신 CA 번들을 사용하고 있는지 어떻게 확인할 수 있습니까?

최신 번들인지 확인하려면 다음 명령을 사용하세요. 이 명령을 실행하려면 Java가 설치되어 있어야 하고 Java 도구가 쉘의 PATH 변수에 있어야 합니다. 자세한 내용은 Java 사용 섹션을 참조하세요.

macOS 및 Amazon Linux

```
keytool -printcert -v -file global-bundle.pem
```

Windows

```
keytool -printcert -v -file global-bundle.p7b
```

CA 번들 이름에 "RDS"가 표시되는 이유는 무엇입니까?

인증서 관리와 같은 일부 관리 기능의 경우 Amazon DocumentDB는 Amazon Relational Database Service(RDS)와 공유되는 운영 기술을 사용합니다.

새 인증서는 언제 만료됩니까?

새 서버 인증서는 (일반적으로) 다음과 같이 만료됩니다.

- rds-ca-rsa2048-g1—2061년 만료
- rds-ca-rsa2048-g1—2021년 만료
- rds-ca-ecc384-g1—2121년 만료

인증서가 만료되기 전에 조치를 취하지 않으면 어떤 종류의 오류가 발생하나요?

오류 메시지는 드라이버에 따라 달라집니다. 일반적으로 '인증서가 만료되었습니다'라는 문자열이 포함된 인증서 검증 오류가 표시됩니다.

새 서버 인증서를 적용한 경우 이전 서버 인증서로 되돌릴 수 있습니까?

인스턴스를 이전 서버 인증서로 되돌려야 하는 경우 클러스터의 모든 인스턴스에 대해 수행하는 것이 좋습니다. AWS Management Console 또는를 사용하여 클러스터의 각 인스턴스에 대한 서버 인증서를 되돌릴 수 있습니다 AWS CLI.

사용 AWS Management Console

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb://https://https://https://httpsAmazon DocumentDB://https://://https://://https://://https://https://https://https://https://
2. 화면 오른쪽 상단의 리전 목록에서 클러스터가 있는 AWS 리전 를 선택합니다.
3. 콘솔 왼쪽의 탐색 창에서 클러스터를 선택합니다.
4. 클러스터 테이블의 클러스터 식별자에서 수정할 인스턴스를 선택합니다. 작업을 선택한 후 수정을 선택합니다.
5. 인증 기관에서 이전 서버 인증서(`rds-ca-2019`)를 선택할 수 있습니다.
6. 수정 사항의 요약을 보려면 계속을 선택합니다.
7. 이 결과 페이지에서 수정 사항이 다음 유지 관리 기간에 적용되도록 예약하거나 수정 사항을 즉시 적용하도록 선택할 수 있습니다. 선택한 다음 Modify instance(인스턴스 수정)를 선택합니다.

> **Note**
>
> 수정 사항을 즉시 적용하기로 선택하면 보류 중 수정 사항 대기열에 있는 변경 사항까지 모두 적용됩니다. 보류 중 수정 사항 중 하나라도 가동 중지를 필요로 하는 경우 이 옵션을 선택하면 예기치 못한 가동 중지가 발생할 수 있습니다.

사용 AWS CLI

```
aws docdb modify-db-instance --db-instance-identifier <db_instance_name> ca-
certificate-identifier rds-ca-2019 <--apply-immediately | --no-apply-immediately>
```

`--no-apply-immediately`를 선택하면, 클러스터의 다음 유지 관리 기간 중에 변경 사항이 적용됩니다.

스냅샷이나 특정 시점으로 복구를 통해 복원하는 경우 새 서버 인증서가 사용됩니까?

2024년 8월 이후 스냅샷을 복원하거나 시점 복원을 수행하면 생성된 새 클러스터에서 새 CA 인증서가 사용됩니다.

Mac OS X Catalina에서 내 Amazon DocumentDB 클러스터에 직접 연결하는 데 문제가 있으면 어떻게 됩니까?

Mac OS는 신뢰할 수 있는 인증서에 대한 요구 사항을 업데이트했습니다. 신뢰할 수 있는 인증서로 인정받으려면, 이제 유효 기간이 397일 이하여야 합니다(https://support.apple.com/en-us/HT211025 참조).

> **Note**
>
> 이 제한은 최신 버전의 Mac OS에서 적용됩니다.

Amazon DocumentDB 인스턴스 인증서는 4년 이상 유효하며 이는 Mac OS의 최대 유효 기간보다 더 깁니다. Mac OS를 실행하는 컴퓨터에서 Amazon DocumentDB 클러스터에 직접 연결하려면 TLS 연결을 만들 때 유효하지 않은 인증서를 허용해야 합니다. 이 경우 유효하지 않은 인증서는 유효 기간이 397일보다 길다는 것을 의미합니다. Amazon DocumentDB 클러스터에 연결할 때 유효하지 않은 인증서를 허용하기 전에 어떤 위험이 따를 수 있는지 이해해야 합니다.

를 사용하여 Mac OS에서 Amazon DocumentDB 클러스터에 연결하려면 `tlsAllowInvalidCertificates` 파라미터를 AWS CLI사용합니다.

```
mongo --tls --host <hostname> --username <username> --password <password> --port 27017
 --tlsAllowInvalidCertificates
```

Amazon DocumentDB TLS 인증서 업데이트 — GovCloud

주제

- 애플리케이션 및 Amazon DocumentDB 클러스터 업데이트
- 문제 해결
- FAQ

> **Note**
>
> 이 정보는 GovCloud(미국 서부) 및 GovCloud(미국 동부) 지역의 사용자에게 적용됩니다.

Amazon DocumentDB(MongoDB 호환) 클러스터의 CA(인증 기관) 인증서는 2022년 5월 18일부터 업데이트됩니다. 전송 계층 보안(TLS)을 사용하도록 설정된 상태에서 Amazon DocumentDB 클러스터를 사용하는 경우(기본 설정), 그리고 클라이언트 응용프로그램 및 서버 인증서를 회전하지 않은 경우, 응용프로그램과 Amazon DocumentDB 클러스터 간의 연결 문제를 완화하려면 다음 단계가 필요합니다.

- 1단계: 새 CA 인증서 다운로드 및 애플리케이션 업데이트
- 2단계: 서버 인증서 업데이트

CA 및 서버 인증서는 Amazon DocumentDB 표준 유지 관리 및 보안 모범 사례의 일부로 업데이트되었습니다. 이전 CA 인증서는 2022년 5월 18일에 만료됩니다. 클라이언트 응용프로그램은 신뢰 저장소에 새 CA 인증서를 추가해야 하며, 기존 Amazon DocumentDB 인스턴스는 이 만료일 이전에 새 CA 인증서를 사용하도록 업데이트해야 합니다.

애플리케이션 및 Amazon DocumentDB 클러스터 업데이트

이 섹션의 단계에 따라 애플리케이션의 CA 인증서 번들(1단계)과 클러스터의 서버 인증서(2단계)를 업데이트합니다. 프로덕션 환경에 변경 사항을 적용하기 전에 개발 또는 스테이징 환경에서 이러한 단계를 테스트하는 것이 좋습니다.

> **Note**
>
> Amazon DocumentDB 클러스터가 있는 각 AWS 리전 에서 1단계와 2단계를 완료해야 합니다.

1단계: 새 CA 인증서 다운로드 및 애플리케이션 업데이트

새 CA 인증서를 다운로드하고 응용 프로그램을 업데이트하여 새 CA 인증서를 사용하여 특정 리전의 Amazon DocumentDB에 TLS 연결을 만듭니다.

- GovCloud(미국 서부)의 경우 https://truststore.pki.us-gov-west-1.rds.amazonaws.com/us-gov-west-1/us-gov-west-1-bundle.pem에서 새 CA 인증서 번들을 다운로드합니다. 그러면 `us-gov-west-1-bundle.pem`라는 파일이 다운로드됩니다.

- GovCloud(미국 동부)의 경우 https://truststore.pki.us-gov-west-1.rds.amazonaws.com/us-gov-east-1/us-gov-east-1-bundle.pem에서 새 CA 인증서 번들을 다운로드합니다. 그러면 `us-gov-east-1-bundle.pem`라는 파일이 다운로드됩니다.

> **Note**
>
> 이전 CA 인증서(`rds-ca-2017-root.pem`)와 새 CA 인증서(`rds-ca-rsa2048-g1.pem`, `rds-ca-rsa4096-g1.pem` 또는 `rds-ca-ecc384-g1.pem`)가 모두 포함된 키 스토어에 액세스하는 경우 키 스토어에서 선택한 인증서를 선택하는지 확인하세요. 각 인증서에 대한 자세한 내용은 아래 2단계를 참조하세요.

```
wget https://truststore.pki.us-gov-west-1.rds.amazonaws.com/us-gov-west-1/us-gov-west-1-bundle.pem
```

```
wget https://truststore.pki.us-gov-west-1.rds.amazonaws.com/us-gov-east-1/us-gov-east-1-bundle.pem
```

그런 다음 새 인증서 번들을 사용하도록 애플리케이션을 업데이트합니다. 새 CA 번들에는 이전 CA 인증서와 새 CA 인증서(`rds-ca-rsa2048-g1.pem`, `rds-ca-rsa4096-g1.pem` 또는 `rds-ca-ecc384-g1.pem`)가 모두 포함되어 있습니다. 새 CA 번들에 두 CA 인증서가 모두 포함되어 있으므로 애플리케이션과 클러스터를 두 단계로 업데이트할 수 있습니다.

2021년 12월 21일 이후에 CA 인증서 번들을 다운로드하면 새 CA 인증서 번들이 사용됩니다. 애플리케이션에서 최신 CA 인증서 번들을 사용하고 있는지 확인하려면 최신 CA 번들을 사용하고 있는지 어떻게 확인할 수 있습니까? 섹션을 참조하세요. 애플리케이션에서 최신 CA 인증서 번들을 이미 사용하고 있는 경우 2단계로 건너뛸 수 있습니다.

애플리케이션에 CA 번들을 사용하는 경우의 예는 전송 중 데이터 암호화 및 TLS가 활성화된 상태에서 연결 섹션을 참조하세요.

> **Note**
>
> 현재 MongoDB Go Driver 1.2.1은 `sslcertificateauthorityfile`에서 하나의 CA 서버 인증서만 허용합니다. TLS가 활성화될 때 Go를 사용하여 Amazon DocumentDB에 연결하는 방법은 TLS가 활성화된 상태에서 연결 섹션을 참조하세요.

2단계: 서버 인증서 업데이트

응용프로그램이 새 CA 번들을 사용하도록 업데이트되면, 다음 단계는 Amazon DocumentDB 클러스터의 각 인스턴스를 수정하여 서버 인증서를 업데이트하는 것입니다. 새 서버 인증서를 사용하도록 인스턴스를 수정하려면 다음 지침을 참조하세요.

Amazon DocumentDB는 DB 인스턴스의 DB 서버 인증서에 서명할 수 있는 다음 CA를 제공합니다.

- rds-ca-ecc384-g1—ECC 384 프라이빗 키 알고리즘과 SHA384 서명 알고리즘을 갖춘 인증 기관을 사용합니다. 이 CA는 자동 서버 인증서 교체를 지원합니다. 이는 현재 Amazon DocumentDB 4.0 및 5.0에서만 지원됩니다.
- rds-ca-rsa2048-g1 - 대부분의 AWS 리전에서 RSA 2048 프라이빗 키 알고리즘 및 SHA256 서명 알고리즘과 함께 인증 기관을 사용합니다. 이 CA는 자동 서버 인증서 교체를 지원합니다.
- rds-ca-rsa4096-g1—RSA 4096 프라이빗 키 알고리즘과 SHA256 서명 알고리즘을 갖춘 인증 기관을 사용합니다. 이 CA는 자동 서버 인증서 교체를 지원합니다.

> **Note**
>
> 를 사용하는 경우 describe-certificates를 사용하여 위에 나열된 인증 기관의 유효성을 확인할 AWS CLI수 있습니다.

> **Note**
>
> Amazon DocumentDB 4.0 및 5.0 인스턴스는 재부팅할 필요가 없습니다.
> Amazon DocumentDB 3.6 인스턴스를 업데이트하려면 재부팅이 필요하며 이로 인해 서비스 중단이 발생할 수 있습니다. 서버 인증서를 업데이트하기 전에 1단계를 완료했는지 확인합니다.

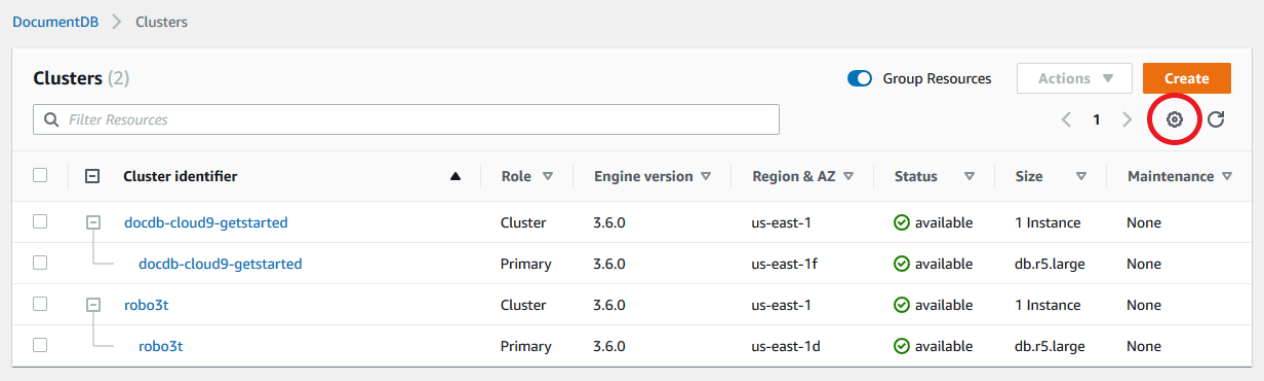
Using the AWS Management Console

AWS Management Console을 사용하여 기존 Amazon DocumentDB 인스턴스의 이전 서버 인증서를 식별하고 교체하기 위해 다음 단계를 완료하세요.

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 화면 오른쪽 상단의 리전 목록에서 클러스터가 있는 AWS 리전 를 선택합니다.

3. 콘솔 왼쪽의 탐색 창에서 클러스터를 선택합니다.
4. 이전 서버 인증서(rds-ca-2017)에 아직 남아 있는 인스턴스를 식별해야 할 수도 있습니다. 기본적으로 숨겨져 있는 인증 기관 열에서 이 작업을 수행할 수 있습니다. 인증 기관 열을 표시하려면 다음을 수행합니다.

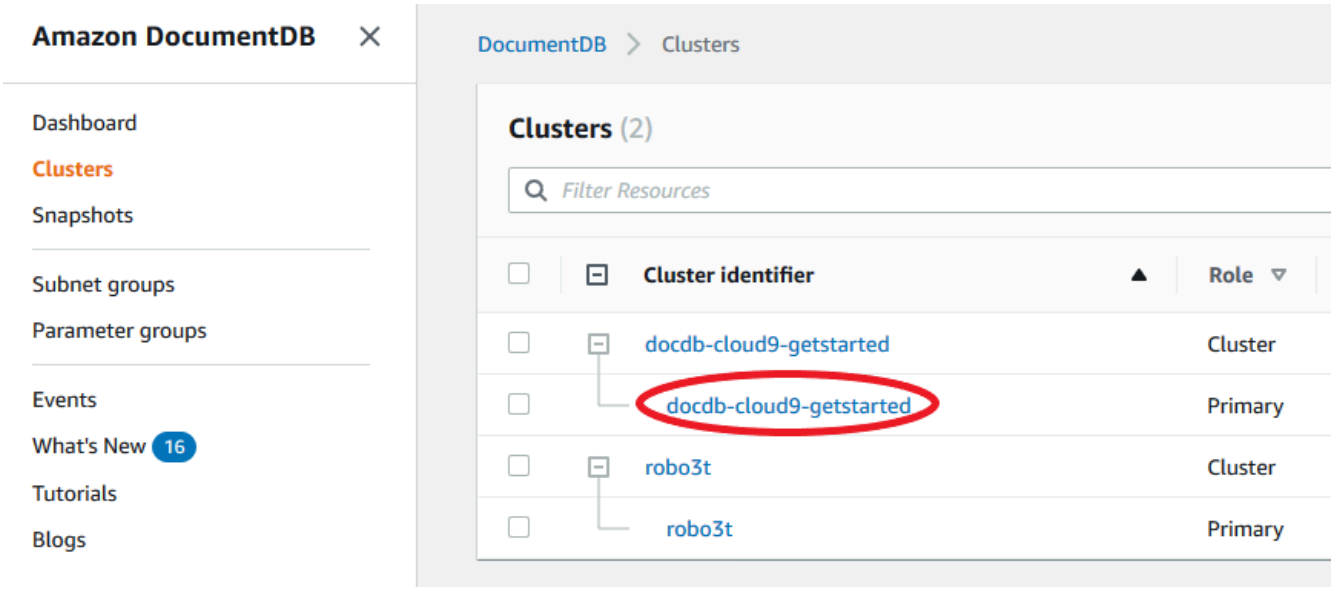
   a. 설정 아이콘을 선택합니다.



   b. 표시되는 열 목록에서 인증 기관 열을 선택합니다.

   c. 그런 다음 확인을 선택해 변경 사항을 저장합니다.

5. 클러스터 탐색 상자에 클러스터 식별자열이 표시됩니다. 인스턴스는 아래 스크린샷과 마찬가지로 클러스터 아래에 나열됩니다.



6. 관심 있는 인스턴스의 왼쪽에 있는 박스를 선택합니다.
7. 작업을 선택한 다음 수정을 선택합니다.
8. 인증 기관에서 이 인스턴스에 대한 새 서버 인증서(rds-ca-rsa2048-g1,rds-ca-rsa4096-g1, 또는 rds-ca-ecc384-g1)를 선택합니다.

9. 다음 페이지에서 변경 사항 요약을 볼 수 있습니다. 연결이 중단되지 않도록 인스턴스를 수정하기 전에 애플리케이션이 최신 인증서 CA 번들을 사용하고 있는지 확인하는 추가 알림이 있습니다.
10. 다음 유지 관리 기간 중에 수정 사항을 적용하거나 즉시 적용하도록 선택할 수 있습니다. 서버 인증서를 즉시 수정하려는 경우 즉시 적용 옵션을 사용합니다.
11. Modify instance(인스턴스 수정)를 선택하여 업데이트를 완료합니다.

Using the AWS CLI

AWS CLI을 사용하여 기존 Amazon DocumentDB 인스턴스의 이전 서버 인증서를 식별하고 교체하기 위해 다음 단계를 완료하세요.

1. 인스턴스를 즉시 수정하려면 클러스터의 각 인스턴스에 대해 다음 명령을 실행합니다. 인증서 중에서 `rds-ca-rsa2048-g1`,`rds-ca-rsa4096-g1` 또는 `rds-ca-ecc384-g1` 중 하나를 수행합니다.

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier>
 --ca-certificate-identifier rds-ca-rsa4096-g1 --apply-immediately
```

2. 클러스터의 다음 유지 관리 기간 동안 새 CA 인증서를 사용하도록 클러스터의 인스턴스를 수정하려면 클러스터의 각 인스턴스에 대해 다음 명령을 실행합니다. 인증서 중에서 `rds-ca-rsa2048-g1`,`rds-ca-rsa4096-g1` 또는 `rds-ca-ecc384-g1` 중 하나를 수행합니다.

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier>
 --ca-certificate-identifier rds-ca-rsa4096-g1 --no-apply-immediately
```

문제 해결

인증서 교체의 일부로 클러스터에 연결하는 데 문제가 있는 경우 다음을 권장합니다.

- 인스턴스를 재부팅합니다. 새 인증서를 교체하려면 각 인스턴스를 재부팅해야 합니다. 새 인증서를 하나 이상의 인스턴스에 적용했지만 재부팅하지 않은 경우 인스턴스를 재부팅하여 새 인증서를 적용합니다. 자세한 내용은 Amazon DocumentDB 인스턴스 재부팅 단원을 참조하십시오.
- 클라이언트가 최신 인증서 번들을 사용하고 있는지 확인합니다. 최신 CA 번들을 사용하고 있는지 어떻게 확인할 수 있습니까?을 참조하세요.
- 인스턴스가 최신 인증서를 사용하고 있는지 확인합니다. 내 Amazon DocumentDB 인스턴스가 이전/새 서버 인증서를 사용하고 있는지 어떻게 알 수 있습니까?을 참조하세요.

- 애플리케이션에서 최신 인증서 CA를 사용하고 있는지 확인합니다. Java 및 Go와 같은 일부 드라이버는 인증서 번들에서 신뢰 저장소로 여러 인증서를 가져오려면 추가 코드가 필요합니다. TLS를 사용하여 Amazon DocumentDB에 연결하는 방법에 대한 자세한 내용은 Amazon DocumentDB에 프로그래밍 방식으로 연결을 참조하세요.
- 고객 지원 센터에 문의.. 질문이나 문제가 있으면 지원에 연락하세요.

FAQ

다음은 TLS 인증서에 대한 몇 가지 일반적인 질문에 대한 답변입니다.

질문이나 문제가 있는 경우 어떻게 해야 합니까?

질문이나 문제가 있으면 지원에 연락하세요.

Amazon DocumentDB 클러스터에 연결하는 데 TLS를 사용했는지 어떻게 알 수 있습니까?

클러스터의 클러스터 파라미터 그룹에 대한 `tls` 파라미터를 확인하여 클러스터에서 TLS를 사용하고 있는지 확인할 수 있습니다. `tls` 파라미터가 `enabled`로 설정된 경우 TLS 인증서를 사용하여 클러스터에 연결되어 있는 것입니다. 자세한 내용은 Amazon DocumentDB 클러스터 파라미터 그룹 관리 단원을 참조하십시오.

CA와 서버 인증서를 업데이트하는 이유는 무엇입니까?

Amazon DocumentDB CA 및 서버 인증서는 Amazon DocumentDB 표준 유지 관리 및 보안 모범 사례의 일환으로 업데이트되고 있습니다. 현재 CA 및 서버 인증서는 2022년 5월 18일 수요일에 만료됩니다.

만료일까지 조치를 취하지 않으면 어떻게 됩니까?

TLS를 사용하여 Amazon DocumentDB 클러스터에 연결하고 2022년 5월 18일까지 인증서를 변경하지 않으면 TLS를 통해 연결하는 응용프로그램은 더 이상 Amazon DocumentDB 클러스터와 통신할 수 없습니다.

Amazon DocumentDB는 만료가 이뤄질 때까지 데이터베이스 인증서를 자동으로 교체하지 않을 것입니다. 만료일 이전 또는 이후에 새 CA 인증서를 사용하려면 애플리케이션과 클러스터를 업데이트해야 합니다.

내 Amazon DocumentDB 인스턴스가 이전/새 서버 인증서를 사용하고 있는지 어떻게 알 수 있습니까?

여전히 이전 서버 인증서를 사용하는 Amazon DocumentDB 인스턴스를 식별하려면 Amazon DocumentDB AWS Management Console 또는를 사용할 수 있습니다 AWS CLI.

사용 AWS Management Console

이전 인증서를 사용하는 클러스터의 인스턴스를 식별하려면

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 화면 오른쪽 상단의 리전 목록에서 인스턴스가 있는 AWS 리전 를 선택합니다.
3. 콘솔 왼쪽의 탐색 창에서 인스턴스를 선택합니다.
4. 인증 기관 열(기본적으로 숨겨짐)에는 아직 이전 서버 인증서(`rds-ca-2017`) 및 새 서버 인증서(`rds-ca-rsa2048-g1`, `rds-ca-rsa4096-g1`, 또는 `rds-ca-ecc384-g1`)에 있는 인스턴스가 표시됩니다. 인증 기관 열을 표시하려면 다음을 수행합니다.
   a. 설정 아이콘을 선택합니다.
   b. 표시되는 열 목록에서 인증 기관 열을 선택합니다.
   c. 그런 다음 확인을 선택해 변경 사항을 저장합니다.

사용 AWS CLI

이전 서버 인증서를 사용하는 클러스터의 인스턴스를 식별하려면 `describe-db-clusters` 명령을 다음과 함께 사용합니다.

```
aws docdb describe-db-instances \
    --filters Name=engine,Values=docdb \
    --query 'DBInstances[*].
{CertificateVersion:CACertificateIdentifier,InstanceID:DBInstanceIdentifier}'
```

Amazon DocumentDB 클러스터에서 개별 인스턴스를 수정하여 서버 인증서를 업데이트하려면 어떻게 해야 합니까?

특정 클러스터의 모든 인스턴스에 대한 서버 인증서를 동시에 업데이트하는 것이 좋습니다. 클러스터의 인스턴스를 수정하기 위해 콘솔 또는 AWS CLI를 사용할 수 있습니다.

> **Note**
>
> 인스턴스를 업데이트하려면 재부팅이 필요하며 이로 인해 서비스 중단이 발생할 수 있습니다. 서버 인증서를 업데이트하기 전에 1단계를 완료했는지 확인합니다.

사용 AWS Management Console

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 화면 오른쪽 상단의 리전 목록에서 클러스터가 있는 AWS 리전 를 선택합니다.
3. 콘솔 왼쪽의 탐색 창에서 인스턴스를 선택합니다.
4. 인증 기관 열(기본적으로 숨겨짐)에는 아직 이전 서버 인증서에 있는 인스턴스(`rds-ca-2017`)가 표시됩니다. 인증 기관 열을 표시하려면 다음을 수행합니다.
   a. 설정 아이콘을 선택합니다.
   b. 표시되는 열 목록에서 인증 기관 열을 선택합니다.
   c. 그런 다음 확인을 선택해 변경 사항을 저장합니다.
5. 수정할 인스턴스를 선택합니다.
6. 작업을 선택한 다음 수정을 선택합니다.
7. 인증 기관에서 이 인스턴스에 대한 새 서버 인증서(`rds-ca-rsa2048-g1`,`rds-ca-rsa4096-g1`, 또는 `rds-ca-ecc384-g1`) 중 하나를 선택합니다.
8. 다음 페이지에서 변경 사항 요약을 볼 수 있습니다. 연결이 중단되지 않도록 인스턴스를 수정하기 전에 애플리케이션이 최신 인증서 CA 번들을 사용하고 있는지 확인하는 추가 알림이 있습니다.
9. 다음 유지 관리 기간 중에 수정 사항을 적용하거나 즉시 적용하도록 선택할 수 있습니다.
10. Modify instance(인스턴스 수정)를 선택하여 업데이트를 완료합니다.

사용 AWS CLI

AWS CLI을 사용하여 기존 Amazon DocumentDB 인스턴스의 이전 서버 인증서를 식별하고 교체하기 위해 다음 단계를 완료하세요.

1. 인스턴스를 즉시 수정하려면 클러스터의 각 인스턴스에 대해 다음 명령을 실행합니다.

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier> --
ca-certificate-identifier rds-ca-rsa4096-g1 --apply-immediately
```

2. 클러스터의 다음 유지 관리 기간 동안 새 CA 인증서를 사용하도록 클러스터의 인스턴스를 수정하려면 클러스터의 각 인스턴스에 대해 다음 명령을 실행합니다.

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier> --
ca-certificate-identifier rds-ca-rsa4096-g1 --no-apply-immediately
```

기존 클러스터에 새 인스턴스를 추가하면 어떻게 됩니까?

생성된 모든 새 인스턴스는 이전 서버 인증서를 사용하며, 이전 CA 인증서를 사용하는 TLS 연결이 필요합니다. 2022년 3월 21일 이후에 생성된 모든 새 Amazon DocumentDB 인스턴스는 기본적으로 새 인증서를 사용합니다.

클러스터에 인스턴스 대체 또는 장애 조치가 있는 경우 어떻게 됩니까?

클러스터에 인스턴스 대체가 있는 경우, 생성된 새 인스턴스는 인스턴스가 이전에 사용했던 것과 동일한 서버 인증서를 계속 사용합니다. 모든 인스턴스에 대한 서버 인증서를 동시에 업데이트하는 것이 좋습니다. 클러스터에 장애 조치가 발생하면 새로운 기본 서버 인증서가 사용됩니다.

TLS를 사용하여 클러스터에 연결하지 않는 경우에도 각 인스턴스를 업데이트해야 합니까?

Amazon DocumentDB 클러스터 연결에 TLS를 사용하지 않으면, 별도로 필요한 작업이 없습니다.

TLS를 사용하여 클러스터에 연결하지 않고 나중에 연결할 계획이라면 어떻게 해야 합니까?

2022년 3월 21일 이전에 클러스터를 생성한 경우, 이전 섹션의 1단계 및 2단계를 수행하여 응용프로그램이 업데이트된 CA 번들을 사용하고 있고 각 Amazon DocumentDB 인스턴스가 최신 서버 인증서를 사용하고 있는지 확인합니다. 2022년 3월 21일 이후에 클러스터를 생성한 경우 클러스터에 이미 최신 서버 인증서가 있습니다. 애플리케이션이 최신 CA 번들을 사용하고 있는지 확인하려면 TLS를 사용하여 클러스터에 연결하지 않는 경우에도 각 인스턴스를 업데이트해야 합니까? 단원을 참조하십시오.

2022년 3월 18일 이후로 기한을 연장할 수 있습니까?

애플리케이션이 TLS를 통해 연결되는 경우, 2022년 5월 18일 이후로 기한을 연장할 수 없습니다.

최신 CA 번들을 사용하고 있는지 어떻게 확인할 수 있습니까?

호환성을 위해 이전 CA 번들 파일과 새 CA 번들 파일의 이름이 모두 `us-gov-west-1-bundle.pem`으로 지정됩니다. 또한 `openssl` 또는 `keytool` 같은 도구를 사용하여 CA 번들을 검사할 수도 있습니다.

CA 번들 이름에 "RDS"가 표시되는 이유는 무엇입니까?

인증서 관리와 같은 일부 관리 기능의 경우 Amazon DocumentDB는 Amazon Relational Database Service(RDS)와 공유되는 운영 기술을 사용합니다.

새 인증서는 언제 만료됩니까?

새 서버 인증서는 (일반적으로) 다음과 같이 만료됩니다.

- rds-ca-rsa2048-g1—2061년 만료
- rds-ca-rsa2048-g1—2021년 만료
- rds-ca-ecc384-g1—2121년 만료

인증서가 만료되기 전에 조치를 취하지 않으면 어떤 종류의 오류가 발생하나요?

오류 메시지는 드라이버에 따라 달라집니다. 일반적으로 '인증서가 만료되었습니다'라는 문자열이 포함된 인증서 검증 오류가 표시됩니다.

새 서버 인증서를 적용한 경우 이전 서버 인증서로 되돌릴 수 있습니까?

인스턴스를 이전 서버 인증서로 되돌려야 하는 경우 클러스터의 모든 인스턴스에 대해 수행하는 것이 좋습니다. AWS Management Console 또는를 사용하여 클러스터의 각 인스턴스에 대한 서버 인증서를 되돌릴 수 있습니다 AWS CLI.

사용 AWS Management Console

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 화면 오른쪽 상단의 리전 목록에서 클러스터가 있는 AWS 리전 를 선택합니다.
3. 콘솔 왼쪽의 탐색 창에서 인스턴스를 선택합니다.
4. 수정할 인스턴스를 선택합니다. 작업을 선택한 후 수정을 선택합니다.

5. 인증 기관에서 이전 서버 인증서(rds-ca-2017)를 선택할 수 있습니다.
6. 수정 사항의 요약을 보려면 계속을 선택합니다.
7. 이 결과 페이지에서 수정 사항이 다음 유지 관리 기간에 적용되도록 예약하거나 수정 사항을 즉시 적용하도록 선택할 수 있습니다. 선택한 다음 Modify instance(인스턴스 수정)를 선택합니다.

> **Note**
>
> 수정 사항을 즉시 적용하기로 선택하면 보류 중 수정 사항 대기열에 있는 변경 사항까지 모두 적용됩니다. 보류 중 수정 사항 중 하나라도 가동 중지를 필요로 하는 경우 이 옵션을 선택하면 예기치 못한 가동 중지가 발생할 수 있습니다.

사용 AWS CLI

```
aws docdb modify-db-instance --db-instance-identifier <db_instance_name> ca-
certificate-identifier rds-ca-2017 <--apply-immediately | --no-apply-immediately>
```

`--no-apply-immediately`를 선택하면, 클러스터의 다음 유지 관리 기간 중에 변경 사항이 적용됩니다.

스냅샷이나 특정 시점으로 복구를 통해 복원하는 경우 새 서버 인증서가 사용됩니까?

2022년 3월 21일 이후에 스냅샷을 복원하거나 시점 복원을 수행하면 생성된 새 클러스터에서 새 CA 인증서가 사용됩니다.

Mac OS X Catalina에서 내 Amazon DocumentDB 클러스터에 직접 연결하는 데 문제가 있으면 어떻게 해야 하나요?

Mac OS X Catalina는 신뢰할 수 있는 인증서에 대한 요구 사항을 업데이트했습니다. 신뢰할 수 있는 인증서로 인정받으려면, 이제 유효 기간이 825일 이하여야 합니다 (https://support.apple.com/en-us/HT210176 참조). Amazon DocumentDB 인스턴스 인증서는 4년 이상 유효하며 이는 Mac OS X의 최대 유효기간보다 더 깁니다. Mac OS X Catalina를 실행하는 컴퓨터에서 Amazon DocumentDB 클러스터에 직접 연결하려면 TLS 연결을 만들 때 유효하지 않은 인증서를 허용해야 합니다. 이 경우 유효하지 않은 인증서는 유효 기간이 825일보다 길다는 것을 의미합니다. Amazon DocumentDB 클러스터에 연결할 때 유효하지 않은 인증서를 허용하기 전, 어떤 리스크가 따를 수 있는지 이해해야 합니다.

를 사용하여 OS X Catalina에서 Amazon DocumentDB 클러스터에 연결하려면 `tlsAllowInvalidCertificates` 파라미터를 AWS CLI사용합니다.

```
mongo --tls --host <hostname> --username <username> --password <password> --port 27017
 --tlsAllowInvalidCertificates
```

Amazon DocumentDB에서의 규정 준수 검증

서드 파티 감사자는 다음을 포함하는 여러 AWS 규정 준수 프로그램의 일환으로 Amazon DocumentDB의 보안 및 규정 준수를 평가합니다.

- SOC(시스템 및 조직 제어) 1, 2 및 3 자세한 내용은 SOC를 참조하세요.
- 연방정부의 위험 및 인증 관리 프로그램(FedRAMP). 자세한 내용은 규정 준수 프로그램 제공 범위 내AWS 서비스를 참조하세요.
- PCI DSS(지불 카드 산업 데이터 보안 표준) 자세한 내용은 PCI DSS를 참조하세요.
- ISO 9001, 27001, 27017 및 27018. 자세한 내용은 ISO 인증을 참조하세요.
- HIPAA(미국 건강 보험 양도 및 책임에 관한 법) BAA(비즈니스 제휴 계약) 자세한 내용은 HIPAA 규정 준수 섹션을 참조하세요.

AWS 는 규정 준수 프로그램 제공 범위 내 AWS 서비스에서 특정 AWS 규정 준수 프로그램 범위 내에서 자주 업데이트되는 서비스 목록을 제공합니다.

를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다 AWS Artifact. 자세한 내용은 AWS 아티팩트에서 보고서 다운로드를 참조하세요.

AWS 규정 준수 프로그램에 대한 자세한 내용은 AWS 규정 준수 프로그램을 참조하세요.

Amazon DocumentDB 사용 시 귀하의 규정 준수 책임은 데이터의 민감도, 조직의 규정 준수 목표, 관련 법률과 규정에 따라 결정됩니다. Amazon DocumentDB를 사용할 때 HIPAA 또는 PCI와 같은 표준을 준수해야 하는 경우는 다음과 같은 도움이 되는 리소스를 AWS 제공합니다.

- AWS 규정 준수 리소스 - 업계 및 위치에 적용될 수 있는 워크북 및 가이드 모음입니다.
- 보안 및 규정 준수 빠른 시작 안내서 - 아키텍처 고려 사항을 설명하고 보안 및 규정 준수 중심 기준 환경을 배포하기 위한 단계를 제공하는 배포 안내서입니다 AWS.
- AWS Config - 리소스 구성이 내부 관행, 업계 지침 및 규정을 얼마나 잘 준수하는지 평가하는 서비스입니다.
- AWS Security Hub - 보안 업계 표준 및 모범 사례 준수를 확인하는 데 도움이 AWS 되는 내 보안 상태에 대한 포괄적인 보기입니다.

- HIPAA 보안 및 규정 준수를 위한 설계 백서 - 기업이를 사용하여 HIPAA 준수 애플리케이션을 AWS 생성하는 방법을 설명하는 백서입니다.

Amazon DocumentDB의 복원성

AWS 글로벌 인프라는 AWS 리전 및 가용 영역을 중심으로 구축됩니다.는 지연 시간이 짧고 처리량이 높으며 중복성이 높은 네트워킹과 연결된 물리적으로 분리되고 격리된 여러 가용 영역을 AWS 리전 제공합니다. 가용 영역을 사용하면 중단 없이 가용 영역 간에 자동으로 장애 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 복수 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

Amazon DocumentDB 클러스터는 최소 2개 이상의 가용 영역에 서브넷이 2개 이상인 Amazon VPC에서만 생성할 수 있습니다. 두 개 이상의 가용 영역에 클러스터 인스턴스를 배포하면, 가용 영역에 장애가 발생할 경우에도 Amazon DocumentDB를 통해 클러스터에서 인스턴스를 사용할 수 있습니다. Amazon DocumentDB 클러스터의 클러스터 볼륨은 항상 세 개의 가용 영역에 배포되므로 데이터 손실의 가능성을 최소화하고 스토리지의 내구성을 높일 수 있습니다.

AWS 리전 및 가용 영역에 대한 자세한 내용은 AWS 글로벌 인프라를 참조하세요.

AWS 글로벌 인프라 외에도 Amazon DocumentDB는 데이터 복원력 및 백업 요구 사항을 지원하는 데 도움이 되는 몇 가지 기능을 제공합니다.

내결함성 및 자가 복구 스토리지

스토리지 볼륨의 각 10GB 부분은 세 개의 가용 영역에 걸쳐 여섯 가지 방식으로 복제됩니다. Amazon DocumentDB는 데이터베이스 쓰기 가용성에 영향을 주지 않고 최대 2개의 데이터 사본 손실을 투명하게 처리하고 읽기 가용성에 영향을 주지 않고 최대 3개의 데이터 사본 손실을 투명하게 처리하는 내결함성 스토리지를 사용합니다. 또한 Amazon DocumentDB 스토리지는 자가 복구 기능을 제공하므로 데이터 블록과 디스크를 지속적으로 스캔하여 오류가 있는지 확인하고 자동으로 교체합니다.

수동 백업 및 복원

Amazon DocumentDB는 장기 보존 및 복구를 위해 클러스터의 전체 백업을 생성할 수 있는 기능을 제공합니다. 자세한 내용은 Amazon DocumentDB에서 백업 및 복원 단원을 참조하십시오.

시점 복구

특정 시점으로 복구를 사용하면 우발적인 쓰기 또는 삭제 작업으로부터 Amazon DocumentDB 클러스터를 보호할 수 있습니다. 특정 시점으로 복구를 설정해 두면 온디맨드 백업의 생성, 유지 관리, 예약을 걱정할 필요가 없습니다. 자세한 내용은 특정 시점으로 복원 단원을 참조하십시오.

Amazon DocumentDB의 인프라 보안

관리형 서비스인 Amazon DocumentDB는 AWS 글로벌 네트워크 보안으로 보호됩니다. AWS 보안 서비스 및가 인프라를 AWS 보호하는 방법에 대한 자세한 내용은 AWS 클라우드 보안을 참조하세요. 인프라 보안 모범 사례를 사용하여 AWS 환경을 설계하려면 Security Pillar AWS Well-Architected Framework의 인프라 보호를 참조하세요.

AWS 게시된 API 호출을 사용하여 네트워크를 통해 Amazon DocumentDB에 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 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.

어떤 네트워크 위치에서든 이러한 API 작업을 호출할 수 있습니다. Amazon DocumentDB 정책을 사용하여 특정 Amazon Virtual Private Cloud(Amazon VPC) 엔드포인트 또는 특정 VPC에서 액세스를 제어할 수도 있습니다. 이렇게 하면 네트워크 내의 특정 VPC에서만 지정된 Amazon DocumentDB 리소스에 대한 AWS 네트워크 액세스가 효과적으로 격리됩니다.

> **Note**
>
> Amazon DocumentDB는 리소스 기반 액세스 정책을 지원하지 않습니다.

Amazon DocumentDB API 및 인터페이스 VPC 엔드포인트(AWS PrivateLink)

> **Note**
>
> Amazon DocumentDB 탄력적 클러스터는 AWS PrivateLink VPC 엔드포인트를 지원하지 않습니다.

인터페이스 VPC 엔드포인트를 생성하여 VPC와 Amazon DocumentDB API 엔드포인트 간에 프라이빗 연결을 설정할 수 있습니다. 인터페이스 엔드포인트는 로 구동됩니다 AWS PrivateLink.

Amazon DocumentDB 인스턴스 기반 클러스터에는 인터페이스 VPC 엔드포인트 연결이 필요하지 않지만를 AWS PrivateLink 사용하면 인터넷 게이트웨이, NAT 디바이스, VPN 연결 또는 AWS Direct Connect 연결 없이 Amazon DocumentDB API 작업에 비공개로 액세스할 수 있습니다. VPC의 Amazon DocumentDB 인스턴스는 데이터베이스 인스턴스 및 데이터베이스 클러스터를 시작, 수정 또는 종료하기 위해 Amazon DocumentDB API 엔드포인트와 통신하는 데 퍼블릭 IP 주소가 필요하지 않습니다. 또한 Amazon DocumentDB 인스턴스가 사용 가능한 Amazon DocumentDB API 작업을 사용하기 위해 퍼블릭 IP 주소가 필요하지 않습니다. VPC와 Amazon DocumentDB 간의 트래픽은 Amazon 네트워크를 벗어나지 않습니다.

각 인터페이스 엔드포인트는 서브넷에서 하나 이상의 탄력적 네트워크 인터페이스로 표현됩니다. 자세한 내용은 Amazon EC2 사용 설명서의 탄력적 네트워크 인터페이스를 참조하세요.

VPC 엔드포인트에 대한 자세한 내용은 Amazon Virtual Private Cloud(AWS PrivateLink) 사용 설명서의 인터페이스 VPC 엔드포인트를 AWS 서비스 사용하여 액세스를 참조하세요. Amazon DocumentDB 작업에 대한 자세한 내용은 Amazon DocumentDB 클러스터, 인스턴스 및 리소스 관리 API 참조 항목을 참조하세요.

주제

- VPC 엔드포인트에 대한 고려 사항
- 리전 가용성
- Amazon DocumentDB API에 대한 인터페이스 VPC 엔드포인트 생성
- Amazon DocumentDB API에 대한 VPC 엔드포인트 정책 생성

VPC 엔드포인트에 대한 고려 사항

Amazon DocumentDB API 엔드포인트에 대한 인터페이스 VPC 엔드포인트를 설정하기 전에 Amazon Virtual Private Cloud(AWS PrivateLink) 사용 설명서의 인터페이스 엔드포인트 사전 조건을 검토해야 합니다.

Amazon DocumentDB 리소스 관리와 관련된 모든 Amazon DocumentDB API 작업은 AWS PrivateLink를 사용하여 VPC에서 수행할 수 있습니다.

VPC 엔드포인트 정책은 Amazon DocumentDB API 엔드포인트에 대해 지원됩니다. 기본적으로, 엔드포인트를 통해 Amazon DocumentDB API 작업에 대한 전체 액세스가 허용됩니다. 자세한 내용은

Amazon 버추얼 프라이빗 클라우드(AWS PrivateLink) 사용 설명서의 엔드포인트 정책을 사용한 VPC 엔드포인트에 대한 액세스 제어를 참조하세요.

리전 가용성

Amazon DocumentDB API는 현재 AWS 리전다음과 같은 VPC 엔드포인트를 지원합니다.

- 미국 동부(오하이오)
- 미국 동부(버지니아 북부)
- 미국 서부(오레곤)
- 아프리카(케이프타운)
- 아시아 태평양(홍콩)
- 아시아 태평양(뭄바이)
- 아시아 태평양(하이데라바드)
- 아시아 태평양(오사카)
- 아시아 태평양(서울)
- 아시아 태평양(싱가포르)
- 아시아 태평양(시드니)
- 아시아 태평양(도쿄)
- 캐나다(중부)
- 중국(베이징)
- 중국(닝샤)
- 유럽(프랑크푸르트)
- 유럽(아일랜드)
- 유럽(런던)
- 유럽(파리)
- 유럽(스페인)
- 유럽(밀라노)
- 중동(UAE)
- 남아메리카(상파울루)
- AWS GovCloud(미국 동부)
- AWS GovCloud(미국 서부)

Amazon DocumentDB API에 대한 인터페이스 VPC 엔드포인트 생성

Amazon VPC 콘솔 또는 ()를 사용하여 Amazon DocumentDB API에 대한 VPC 엔드포인트를 생성할 수 있습니다AWS CLI. AWS Command Line Interface 자세한 내용은 Amazon Virtual Private Cloud(AWS PrivateLink) 사용 설명서의 인터페이스 VPC 엔드포인트를 AWS 서비스 사용하여에 액세스를 참조하세요.

서비스 이름 `com.amazonaws.region.rds`을 사용하여 Amazon DocumentDB API에 대한 VPC 엔드포인트를 생성합니다.

중국을 제외하고 엔드포인트 AWS 리전 에 대해 프라이빗 DNS를 활성화한 경우 rds.us-east-1.amazonaws.com AWS 리전같은에 대한 기본 DNS 이름을 사용하여 VPC 엔드포인트를 사용하여 Amazon DocumentDB에 API 요청을 할 수 있습니다. 중국(베이징) 및 중국(닝샤)의 경우 각각 rds-api.cn-north-1.amazonaws.com.cn 및 사용하여 VPC 엔드포인트로 API 요청을 AWS 리전수행할 수 rds-api.cn-northwest-1.amazonaws.com.cn.

자세한 내용은 Amazon Virtual Private Cloud(AWS PrivateLink) 사용 설명서의 인터페이스 VPC 엔드포인트를 AWS 서비스 사용하여에 액세스를 참조하세요.

Amazon DocumentDB API에 대한 VPC 엔드포인트 정책 생성

Amazon DocumentDB API에 대한 액세스를 제어하는 VPC 엔드포인트에 엔드포인트 정책을 연결할 수 있습니다. 이 정책은 다음 정보를 지정합니다.

- 작업을 수행할 수 있는 위탁자.
- 수행할 수 있는 작업.
- 작업을 수행할 수 있는 리소스.

자세한 내용은 Amazon 버추얼 프라이빗 클라우드(AWS PrivateLink) 사용 설명서의 엔드포인트 정책을 사용한 VPC 엔드포인트에 대한 액세스 제어를 참조하세요.

예제: Amazon DocumentDB API 작업에 대한 VPC 엔드포인트 정책

다음은 Amazon DocumentDB에 대한 엔드포인트 정책의 예입니다. 이 정책은 엔드포인트에 연결될 때 모든 리소스의 모든 보안 주체에 대한 액세스 권한을 나열된 Amazon DocumentDB API 작업에 부여합니다.

```
{
"Statement":[
      {
```

```
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "docdb:CreateDBInstance",
            "docdb:ModifyDBInstance",
            "docdb:CreateDBSnapshot"
         ],
         "Resource":"*"
      }
   ]
}
```

예: 지정된 AWS 계정의 모든 액세스를 거부하는 VPC 엔드포인트 정책

다음 VPC 엔드포인트 정책은 엔드포인트를 사용하여 리소스에 대한 모든 액세스를 AWS 계정 123456789012에 거부합니다. 이 정책은 다른 계정의 모든 작업을 허용합니다.

```
{
  "Statement": [
    {
      "Action": "*",
      "Effect": "Allow",
      "Resource": "*",
      "Principal": "*"
    },
    {
      "Action": "*",
      "Effect": "Deny",
      "Resource": "*",
      "Principal": { "AWS": [ "123456789012" ] }
     }
   ]
}
```

Amazon DocumentDB에 대한 보안 모범 사례

보안 모범 사례의 경우 AWS Identity and Access Management (IAM) 계정을 사용하여 Amazon DocumentDB API 작업, 특히 Amazon DocumentDB 리소스를 생성, 수정 또는 삭제하는 작업에 대한 액세스를 제어해야 합니다. 이러한 리소스로는 클러스터, 보안 그룹 및 파라미터 그룹을 들 수 있습니다. 또한 IAM을 사용하여 클러스터 복원 백업과 같은 일반적인 관리 작업을 수행하는 작업을 제어해야 합니다. IAM 역할을 생성할 때 최소 권한 원칙을 사용합니다.

- 역할 기반 액세스 제어와 함께 최소 권한을 적용합니다.
- Amazon DocumentDB 리소스를 관리하는 각 사용자에게 개별 IAM 계정을 할당합니다. AWS 계정 루트 사용자를 사용하여 Amazon DocumentDB 리소스를 관리하지 마세요. 자신을 포함한 모든 사람을 위한 IAM 사용자를 생성합니다.
- 각 사용자에게 각자의 임무를 수행하는 데 필요한 최소 권한 집합을 부여합니다.
- IAM 그룹을 사용해 여러 사용자에 대한 권한을 효과적으로 관리합니다. IAM에 대한 자세한 내용은 IAM 사용 설명서를 참조하세요. IAM 모범 사례에 대한 자세한 내용은 IAM 모범 사례를 참조하세요.
- IAM 자격 증명을 정기적으로 교체합니다.
- Amazon DocumentDB의 AWS 보안 암호를 자동으로 교체하도록 Secrets Manager를 구성합니다. 자세한 내용은 AWS Secrets Manager 사용 설명서의 Secrets Manager 보안 암호 교체 및 Amazon DocumentDB 보안 암호 교체를 참조하세요. AWS
- TLS(전송 계층 보안) 및 비활성화되어 있는 암호화를 사용하여 데이터를 암호화합니다.

Amazon DocumentDB 이벤트 감사

Amazon DocumentDB(MongoDB 호환)에서는 클러스터에서 수행된 이벤트를 감사할 수 있습니다. 기록되는 이벤트의 예제로는 성공 또는 실패한 인증 시도, 데이터베이스에서 모음 삭제 또는 인덱스 생성이 있습니다. 기본적으로 Amazon DocumentDB에서 감사는 비활성화되어 있으므로 사용자가 이 기능을 선택해야 합니다.

감사가 활성화되면 Amazon DocumentDB가 데이터 정의 언어(DDL), 데이터 조작 언어(DML), 인증, 권한 부여 및 사용자 관리 이벤트를 Amazon CloudWatch Logs에 기록합니다. 감사가 활성화되면 Amazon DocumentDB가 클러스터의 감사 기록(JSON 문서)을 Amazon CloudWatch Logs로 내보냅니다. Amazon CloudWatch Logs를 사용하여 Amazon DocumentDB 감사 이벤트를 분석, 모니터링 및 보관할 수 있습니다.

Amazon DocumentDB가 감사 활성화에 대해 추가 요금을 부과하지는 않지만 CloudWatch Logs 사용에 따른 표준 요금이 부과됩니다. CloudWatch Logs 요금에 대한 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

Amazon DocumentDB 감사 기능은 로 모니터링되는 서비스 리소스 사용량과는 분명히 다릅니다 AWS CloudTrail. CloudTrail은 AWS Command Line Interface (AWS CLI) 또는 클러스터, 인스턴스, 파라미터 그룹 및 스냅샷과 같은 리소스 AWS Management Console 에서 수행되는 작업을 기록합니다. CloudTrail을 사용한 리소스에 대한 감사는 기본적으로 활성화되어 있으며 비활성화할 수 없습니다. Amazon DocumentDB 감사 기능은 옵트인 기능입니다. 데이터베이스, 컬렉션, 인덱스 및 사용자와 같은 객체의 클러스터 내에서 발생하는 작업을 기록합니다.

주제

- 지원되는 이벤트
- 감사 활성화
- 감사 비활성화
- 감사 이벤트 액세스
- DML 감사 이벤트 필터링

지원되는 이벤트

Amazon DocumentDB 감사는 다음 이벤트 카테고리를 지원합니다.

- 데이터 정의 언어 (DDL) - 데이터베이스 관리 작업, 연결, 사용자 관리 및 권한 부여를 포함합니다.
- 데이터 조작 언어 읽기 이벤트(DML 읽기) - `find()` 및 다양한 집계 연산자, 산술 연산자, 부울 연산자 및 기타 읽기 쿼리 연산자를 포함합니다.
- 데이터 조작 언어 쓰기 이벤트(DML 쓰기) - `insert()`, `update()`, `delete()`, 및 `bulkWrite()` 연산자를 포함합니다

이벤트 유형은 다음과 같습니다.

이벤트 유형	범주	설명
`authCheck`	권한 부여	결과 코드 0: 성공
		결과 코드 13: 권한 없이 작업을 수행하려는 시도.
`authenticate`	연결	새로운 연결에서 성공 또는 실패한 인증 시도.
`auditConfigure`	DDL	감사 필터 구성.
`createDatabase`	DDL	새 데이터베이스 생성.

이벤트 유형	범주	설명
createCollection	DDL	데이터베이스 내에서 새 모음 생성.
createIndex	DDL	모음 내에서 새 인덱스 생성.
dropCollection	DDL	데이터베이스에서 모음 삭제.
dropDatabase	DDL	데이터베이스 삭제.
dropIndex	DDL	모음에서 인덱스 삭제.
modifyChangeStreams	DDL	변경 스트림 생성.
renameCollection	DDL	데이터베이스 내에서 컬렉션 이름을 변경.
createRole	관리 역할	역할 생성
dropAllRolesFromDatabase	관리 역할	데이터베이스에서 모든 역할 삭제.
dropRole	관리 역할	역할 삭제.
grantPrivilegesToRole	관리 역할	역할에 권한 부여.
grantRolesToRole	관리 역할	사용자 정의 역할에 역할 부여.
revokePrivilegesFromRole	관리 역할	역할의 권한 취소.
revokeRolesFromRole	관리 역할	사용자 정의 역할에서 역할 취소.
updateRole	관리 역할	역할 업데이트.
createUser	사용자 관리	새 사용자 생성.

이벤트 유형	범주	설명
dropAllUsersFromDatabase	사용자 관리	데이터베이스에서 모든 사용자 삭제.
dropUser	사용자 관리	기존 사용자 삭제.
grantRolesToUser	사용자 관리	사용자에게 역할 부여.
revokeRolesFromUser	사용자 관리	사용자의 역할 취소.
updateUser	UserManagement	기존 사용자 업데이트.
insert	DML 쓰기	컬렉션에 1개 또는 여러 문서를 삽입합니다.
delete	DML 쓰기	컬렉션에서 1개 또는 여러 문서를 삭제합니다.
update	DML 쓰기	컬렉션에 있는 기존 문서 또는 여러 문서를 수정합니다.
bulkWrite	DML 쓰기	실행 순서를 제어하여 여러 쓰기 작업을 수행합니다.
setAuditConfig	DML 쓰기	DML 감사를 위한 새 필터를 설정합니다.
count	DML 읽기	컬렉션 또는 뷰의 find() 쿼리와 일치하는 문서 수를 반환합니다.
countDocuments	DML 읽기	컬렉션 또는 뷰의 쿼리와 일치하는 문서 수를 반환합니다.

이벤트 유형	범주	설명
find	DML 읽기	컬렉션 또는 뷰에서 문서를 선택하고 선택한 문서에 커서를 반환합니다.
getAuditConfig	DML 읽기	DML 감사를 위해 현재 필터를 검색합니다.
findAndModify	DML 읽기 및 DML 쓰기	1개의 문서를 수정 및 반환합니다.
findOneAndDelete	DML 읽기 및 DML 쓰기	필터 및 정렬 기준에 따라 단일 문서를 삭제하고 삭제된 문서를 반환합니다.
findOneAndReplace	DML 읽기 및 DML 쓰기	지정된 필터를 기반으로 단일 문서를 대체합니다.
findOneAndUpdate	DML 읽기 및 DML 쓰기	필터 및 정렬 기준에 따라 단일 문서를 업데이트합니다.
aggregate	DML 읽기 및 DML 쓰기	집계 파이프라인의 API를 지원합니다.
distinct	DML 읽기	단일 컬렉션 또는 뷰에서 지정된 필드의 고유 값을 찾아 결과를 배열로 반환합니다.

> **Note**
>
> DML 이벤트 문서 매개 변수 필드의 값에는 1KB 크기 제한이 있습니다. Amazon DocumentDB는 값이 1KB를 초과하는 경우 값을 잘라냅니다.

> **Note**
>
> TTL 삭제 이벤트는 현재 감사되지 않습니다.

감사 활성화

클러스터에서 감사를 활성화하는 절차는 두 단계로 이루어져 있습니다. 두 단계를 모두 완료하지 않으면 감사 로그가 CloudWatch Logs로 전송되지 않습니다.

1단계. audit_logs 클러스터 파라미터 활성화

감사를 활성화하려면 파라미터 그룹에서 `audit_logs` 파라미터를 수정해야 합니다. `audit_logs`는 로깅할 이벤트의 쉼표로 구분된 목록입니다. 이벤트는 모두 소문자로 지정해야 하며 목록 항목 사이에 공백이 없어야 합니다.

이 파라미터 그룹에 대해 다음 값을 지정할 수 있습니다.

값	설명	
`ddl`	이를 설정하면 createDatabase, dropDatabase, createCollection, dropCollection, createIndex, dropIndex, authCheck, authenticate, createUser, dropUser, grantRolesToUser, revokeRolesFromUser, updateUser,	

값	설명	
	dropAllUsersFromDatabase 등과 같은 DDL 이벤트에 대한 감사가 활성화됩니다	
`dml_read`	이를 설정하면 find, sort count, distinct, group, projecta, unwind, geoNear, geoIntersects, geoWithin 및 기타 MongoDB 읽기 쿼리 연산자와 같은 DML 읽기 이벤트에 대한 감사가 활성화됩니다.	
`dml_write`	이를 설정하면 insert(), update(), delete() 및 bulkWrite()와 같은 DML 쓰기 이벤트에 대한 감사가 활성화됩니다.	
`all`	이를 설정하면 쿼리 읽기, 쿼리 쓰기, 데이터베이스 작업 및 관리자 작업과 같은 데이터베이스 이벤트를 감사할 수 있습니다.	
`none`	이렇게 설정하면 감사가 비활성화됩니다.	

값	설명	
enabled(레거시)	이는 'ddl'과 동일한 레거시 파라미터 설정입니다. 이를 설정하면 ccreateDatabase, dropDatabase, createCollection, dropCollection, createIndex, dropIndex, authCheck, authenticate, createUser, dropUser, grantRolesToUser, revokeRolesFromUser, updateUser, dropAllUsersFromDatabase 등과 같은 DDL 이벤트에 대한 감사가 활성화됩니다 이것은 레거시 설정이므로 이 설정을 사용하지 않는 것이 좋습니다.	
disabled (레거시)	이는 'none'과 동일한 레거시 파라미터 설정입니다. 이것은 레거시 설정이므로 이 설정을 사용하지 않는 것이 좋습니다.	

> **Note**
>
> audit_logs 클러스터 매개변수의 기본값은 none(레거시 'disabled')입니다.

위에서 언급한 값을 조합하여 사용할 수도 있습니다.

값	설명	
`ddl, dml_read`	이를 설정하면 DDL 이벤트 및 DML 읽기 이벤트를 감사할 수 있습니다.	
`ddl, dml_write`	이를 설정하면 DDL 이벤트 및 DML 읽기를 감사할 수 있음	
`dml_read, dml_write`	이를 설정하면 모든 DDL 이벤트를 감사할 수 있음	

> **Note**
>
> 기본 파라미터 그룹은 수정할 수 없습니다.

자세한 내용은 다음 자료를 참조하세요.

- Amazon DocumentDB 클러스터 파라미터 그룹 생성

  사용자 지정 파라미터 그룹을 생성한 후 `audit_logs` 파라미터 값을 `all`로 변경하여 파라미터 그룹을 수정합니다.
- Amazon DocumentDB 클러스터 파라미터 그룹 수정

2단계. Amazon CloudWatch Logs 내보내기 활성화

`audit_logs` 클러스터 파라미터의 값을 `enabled`, `ddl`, `dml_read`, `dml_write`로 설정한 경우 이와 동시에 Amazon DocumentDB가 Amazon CloudWatch로 로그를 내보내도록 활성화해야 합니다. 이들 단계를 하나라도 생략하면 감사 로그가 CloudWatch로 전송되지 않습니다.

클러스터를 생성하거나, 특정 시점 복원을 수행하거나, 스냅샷을 복원할 때 다음 단계에 따라 CloudWatch Logs를 활성화할 수 있습니다.

Using the AWS Management Console

콘솔을 사용하여 CloudWatch로 로그를 내보낼 수 있도록 Amazon DocumentDB를 활성화하려면 다음 주제를 참조하세요.

- 클러스터를 생성하는 경우, 를 사용하여 클러스터 및 기본 인스턴스 생성 AWS Management Console에서 클러스터 생성: 추가 구성(5단계, 로그 내보내기)을 참조하세요.
- 기존 클러스터를 수정하는 경우 — Amazon DocumentDB 클러스터 수정
- 클러스터 스냅샷 복원을 수행하는 경우 — 클러스터 스냅샷에서 복원
- 특정 시점으로 복구를 수행하는 경우 — 특정 시점으로 복원

Using the AWS CLI

새 클러스터를 생성할 때 감사 로그를 활성화하려면

다음 코드는 클러스터 `sample-cluster`을 생성하고 CloudWatch 감사 로그를 활성화합니다.

Example

Linux, macOS, Unix의 경우:

```
aws docdb create-db-cluster \
    --db-cluster-identifier sample-cluster \
    --port 27017 \
    --engine docdb \
    --master-username master-username \
    --master-user-password password \
    --db-subnet-group-name default \
    --enable-cloudwatch-logs-exports audit
```

Windows의 경우:

```
aws docdb create-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --port 27017 ^
    --engine docdb ^
    --master-username master-username ^
    --master-user-password password ^
```

```
    --db-subnet-group-name default ^
    --enable-cloudwatch-logs-exports audit
```

기존 클러스터를 수정할 때 감사 로그를 활성화하려면

다음 코드는 클러스터 `sample-cluster`을 수정하고 CloudWatch 감사 로그를 활성화합니다.

Example

Linux, macOS, Unix의 경우:

```
aws docdb modify-db-cluster \
   --db-cluster-identifier sample-cluster \
   --cloudwatch-logs-export-configuration '{"EnableLogTypes":["audit"]}'
```

Windows의 경우:

```
aws docdb modify-db-cluster ^
   --db-cluster-identifier sample-cluster ^
   --cloudwatch-logs-export-configuration '{"EnableLogTypes":["audit"]}'
```

이러한 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
    "DBCluster": {
        "HostedZoneId": "ZNKXH85TT8WVW",
        "StorageEncrypted": false,
        "DBClusterParameterGroup": "default.docdb4.0",
        "MasterUsername": "<user-name>",
        "BackupRetentionPeriod": 1,
        "Port": 27017,
        "VpcSecurityGroups": [
            {
                "Status": "active",
                "VpcSecurityGroupId": "sg-77186e0d"
            }
        ],
        "DBClusterArn": "arn:aws:rds:us-east-1:900083794985:cluster:sample-cluster",
        "Status": "creating",
        "Engine": "docdb",
        "EngineVersion": "4.0.0",
        "MultiAZ": false,
```

```
        "AvailabilityZones": [
            "us-east-1a",
            "us-east-1c",
            "us-east-1f"
        ],
        "DBSubnetGroup": "default",
        "DBClusterMembers": [],
        "ReaderEndpoint": "sample-cluster.cluster-ro-corcjozrlsfc.us-
east-1.docdb.amazonaws.com",
        "EnabledCloudwatchLogsExports": [
            "audit"
        ],
        "PreferredMaintenanceWindow": "wed:03:08-wed:03:38",
        "AssociatedRoles": [],
        "ClusterCreateTime": "2019-02-13T16:35:04.756Z",
        "DbClusterResourceId": "cluster-YOS52CUXGDTNKDQ7DH72I4LED4",
        "Endpoint": "sample-cluster.cluster-corcjozrlsfc.us-
east-1.docdb.amazonaws.com",
        "PreferredBackupWindow": "07:16-07:46",
        "DBClusterIdentifier": "sample-cluster"
    }
}
```

감사 비활성화

CloudWatch Logs 내보내기를 비활성화하고 `audit_logs` 파라미터를 비활성화하여 감사를 비활성화할 수 있습니다.

CloudWatch Logs 내보내기 비활성화

AWS Management Console 또는를 사용하여 감사 로그 내보내기를 비활성화할 수 있습니다 AWS CLI.

Using the AWS Management Console

다음 절차에서는를 사용하여 CloudWatch AWS Management Console 로 로그를 내보내는 Amazon DocumentDB를 비활성화합니다.

감사 로그를 비활성화하려면

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.

2. 탐색 창에서 클러스터를 선택합니다. 그런 다음 로그 내보내기를 비활성화할 클러스터의 이름 왼쪽에 있는 버튼을 선택합니다.
3. 작업을 선택한 후 수정을 선택합니다.
4. 로그 내보내기 섹션으로 아래로 스크롤하여 비활성을 선택합니다.
5. Continue(계속)을 선택합니다.
6. 변경 사항을 검토하고 이 변경 사항을 클러스터에 적용할 시기를 선택합니다.
   - 예약된 다음 유지 관리 기간에 적용
   - 즉시 적용
7. 클러스터 수정을 선택합니다.

Using the AWS CLI

다음 코드는 클러스터 `sample-cluster`을 수정하고 CloudWatch 감사 로그를 비활성화합니다.

Example

Linux, macOS, Unix의 경우:

```
aws docdb modify-db-cluster \
   --db-cluster-identifier sample-cluster \
   --cloudwatch-logs-export-configuration '{"DisableLogTypes":["audit"]}'
```

Windows의 경우:

```
aws docdb modify-db-cluster ^
   --db-cluster-identifier sample-cluster ^
   --cloudwatch-logs-export-configuration '{"DisableLogTypes":["audit"]}'
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
    "DBCluster": {
        "DBClusterParameterGroup": "default.docdb4.0",
        "HostedZoneId": "ZNKXH85TT8WVW",
        "MasterUsername": "<user-name>",
        "Status": "available",
        "Engine": "docdb",
```

```
        "Port": 27017,
        "AvailabilityZones": [
            "us-east-1a",
            "us-east-1c",
            "us-east-1f"
        ],
        "EarliestRestorableTime": "2019-02-13T16:35:50.387Z",
        "DBSubnetGroup": "default",
        "LatestRestorableTime": "2019-02-13T16:35:50.387Z",
        "DBClusterArn": "arn:aws:rds:us-east-1:900083794985:cluster:sample-
cluster2",
        "Endpoint": "sample-cluster2.cluster-corcjozrlsfc.us-
east-1.docdb.amazonaws.com",
        "ReaderEndpoint": "sample-cluster2.cluster-ro-corcjozrlsfc.us-
east-1.docdb.amazonaws.com",
        "BackupRetentionPeriod": 1,
        "EngineVersion": "4.0.0",
        "MultiAZ": false,
        "ClusterCreateTime": "2019-02-13T16:35:04.756Z",
        "DBClusterIdentifier": "sample-cluster2",
        "AssociatedRoles": [],
        "PreferredBackupWindow": "07:16-07:46",
        "DbClusterResourceId": "cluster-YOS52CUXGDTNKDQ7DH72I4LED4",
        "StorageEncrypted": false,
        "PreferredMaintenanceWindow": "wed:03:08-wed:03:38",
        "DBClusterMembers": [],
        "VpcSecurityGroups": [
            {
                "Status": "active",
                "VpcSecurityGroupId": "sg-77186e0d"
            }
        ]
    }
}
```

audit_logs 파라미터 비활성화

클러스터에 대한 `audit_logs` 파라미터를 비활성화하려면 `audit_logs` 파라미터 값이 `disabled`인 파라미터 그룹을 사용하도록 클러스터를 수정하면 됩니다. 또는 클러스터의 파라미터 그룹에서 `audit_logs` 파라미터 값을 `disabled`로 수정하면 됩니다.

자세한 정보는 다음의 주제를 참조하세요.

- Amazon DocumentDB 클러스터 수정
- Amazon DocumentDB 클러스터 파라미터 그룹 수정

감사 이벤트 액세스

다음 단계에 따라 Amazon CloudWatch에서 감사 이벤트에 액세스합니다.

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 사용자가 Amazon DocumentDB 클러스터와 동일한 리전에 있어야 합니다.
3. 탐색 창에서 로그를 선택합니다.
4. 클러스터에 대한 감사 로그를 찾으려면 목록에서 **/aws/docdb/*yourClusterName*/audit**을 찾아 선택합니다.

   각 인스턴스에 대한 감사 이벤트는 해당 인스턴스 이름 아래에서 찾을 수 있습니다.

DML 감사 이벤트 필터링

DML 감사 필터링 시작하기

DML 감사 이벤트는 Amazon CloudWatch에 기록되기 전에 필터링할 수 있습니다. 이 기능을 사용하려면 감사 로그 및 DML 로깅을 활성화해야 합니다. Amazon DocumentDB는 `atype`, `command`, `user`, `namespace` 및 `auditAuthorizationSuccess`에서 필터링을 지원합니다.

> **Note**
>
> DDL 이벤트는 필터링되지 않습니다.

`db.adminCommand( { command } )` 작업에서 `setAuditConfig`, `filter`및 `auditAuthorizationSuccess` 파라미터를 사용해 감사 필터를 지정하여 언제든지 감사 필터링을 활성화할 수 있습니다.

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter:
         {
```

```
            //filter conditions
         },
      auditAuthorizationSuccess: true | false
   }
)
```

다음 명령을 실행하여 감사 필터 설정을 검색할 수도 있습니다.

```
db.admin.runCommand( { getAuditConfig: 1})
```

보안 요구 사항

권한이 있는 auditConfigure 작업이 있는 데이터베이스 사용자/역할만 DML 감사 필터를 설정하거나 나열할 때 admindb에 대한 명령을 실행할 수 있습니다. [clusterAdmin, hostManager, root]에서 기본 제공 역할 중 하나를 사용하거나 auditConfigure 권한이 있는 사용자 지정 역할을 생성할 수 있습니다. 다음은 auditConfigure 권한과 함께 기존 역할을 사용하는 예제와 사용자 지정 역할을 사용하는 예제입니다.

역할이 내장된 사용자:

```
use admin
db.createUser(
  {
    user: "myClusterAdmin",
    pwd: "password123",
    roles: [ { role: "clusterAdmin", db: "admin" } ]
  }
)
```

사용자 지정 역할이 있는 사용자:

```
use admin
db.createRole(
   {
     role: "myRole",
     privileges: [
       { resource: { cluster: true }, actions: [ "auditConfigure" ] }
     ],
     roles: []
   }
)
```

```
db.createUser(
  {
    user: "myUser",
    pwd: "myPassword",
    roles: [ { role: "myRole", db: "admin" } ]
  }
)
```

사용 사례 필터링

예: 명령을 기준으로 이벤트 필터링

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {
        "$and": [
         {
            "param.command":
               {
                  $in: [ "find","count", "insert", "delete", "update",
 "findandmodify" ]
               }
         }
         ]
      },
      auditAuthorizationSuccess: true
   }
)
```

예: 사용자 이름으로 이벤트 필터링

이 예제에서는 사용자 "myUser"만 로깅됩니다.

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {
      "$and": [
         {
            "param.user":
               {
                  $in: [ "myUser" ]
```

```
                }
          }
          ]},
       auditAuthorizationSuccess: true})
```

예제: **atype** 항목을 사용해 필터링

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {atype: "authCheck"},
      auditAuthorizationSuccess: true
   })
```

> **Note**
>
> 모든 DML 로그의 authCheck 항목은 atype입니다. DDL의 경우만 atype 항목이 다릅니다. filter에서 authCheck 이외의 값을 입력하면 CloudWatch 에서 DML 로그가 생성되지 않습니다.

예: 연산자가 조인한 여러 필터를 사용하여 필터링

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {
      "$and": [
         {
            "param.command":
               {
                  $in: [ "find","count", "insert", "delete", "update",
 "findandmodify" ]
               }
         }
         ],
       "$nor": [
         {
            "param.command":
               {
                  $in: ["count", "insert", "delete", "update", "findandmodify" ]
```

```
                }
          }]
       },
      auditAuthorizationSuccess: true})
```

> **Note**
>
> 최상위 수준에서는 $and, $or, 및 $nor 항목만 지원됩니다. 다른 연산자는 지원되지 않으며 오류가 발생합니다.

예: **auditAuthorizationSuccess**의 이벤트를 기준으로 필터링

이 필터에서는 권한 부여를 성공적으로 통과한 모든 명령이 로그되지 않습니다.

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {},
      auditAuthorizationSuccess: false
   }
)
```

예: **$in** 및 **$nin** 조건을 사용한 필터링

$in 및 $nin에서 둘 다 사용하는 경우 조건 간에 암시적인 'and'가 존재하므로 명령이 로그되지 않습니다. 이 예제에서는 정규식이 find 명령을 차단하므로 아무 것도 로그되지 않습니다.

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {
      "$and": [
         {
            atype: "authCheck",
            "param.command":
               {
                  $in: [ "find", "insert", "delete", "update", "findandmodify" ],
                  $nin: ["count", "insert", "delete", "update", "findandmodify" ],
                  $not: /^^find.*/
               }
```

```
        },
        ],
       "$or": [
        {
           "param.command":
              {
                 $nin: ["count", "insert", "delete", "update", "findandmodify" ]
              }
        }]
       },
      auditAuthorizationSuccess: true})
```

예제: **namespace** 항목을 사용해 필터링

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {
      "$and": [
         {
            "param.ns":
               {
                  $in: [ "test.foo" ]
               }
         }
         ]},
      auditAuthorizationSuccess: true})
```

예제: 기본 필터로 재설정

기본값으로 재설정하면 모든 DML 감사 이벤트가 로그됩니다. 필터링을 기본값으로 재설정하려면 다음 명령을 실행합니다.

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {},
      auditAuthorizationSuccess: true
   }
)
```

Amazon DocumentDB에서 백업 및 복원

Amazon DocumentDB(MongoDB 호환)는 1~35일 동안 Amazon Simple Storage Service(S3)에 데이터를 지속적으로 백업하므로 백업 보존 기간 내에 어떤 시점으로든 신속하게 복구가 가능합니다. 또한 Amazon DocumentDB는 이러한 연속 백업 프로세스의 일환으로 데이터의 스냅샷을 자동으로 생성합니다.

> **Note**
>
> 이는 서비스 관리형 Amazon S3 버킷이며 백업 파일에 액세스할 수 없습니다. 백업을 직접 관리하려면 데이터 덤프, 복원, 가져오기 및 내보내기에 대한 지침을 따르세요.

백업 보존 기간 이후에도 클러스터 데이터의 수동 스냅샷을 생성하여 백업 데이터를 보관할 수 있습니다. 백업 프로세스는 클러스터의 성능에 영향을 주지 않습니다.

이 섹션에서는 Amazon DocumentDB의 백업 기능에 대한 사용 사례를 설명하고 Amazon DocumentDB 클러스터에 대한 백업을 관리하는 방법을 보여 줍니다.

주제

- 백업 및 복원: 개념
- 백업 스토리지 사용량 파악
- 데이터 덤프, 복원, 가져오기 및 내보내기
- 클러스터 스냅샷 고려 사항
- 자동 스냅샷과 수동 스냅샷 비교
- 수동 클러스터 스냅샷 생성
- Amazon DocumentDB 클러스터 스냅샷 복사
- Amazon DocumentDB 클러스터 스냅샷 공유
- 클러스터 스냅샷에서 복원
- 특정 시점으로 복원
- 클러스터 스냅샷 삭제

백업 및 복원: 개념

명사	설명	API(동사)

백업 보관 기간	특정 시점으로 복원을 수행할 수 있는 1~35일 사이의 기간입니다.	`create-db-cluster`
		`modify-db-cluster`
		`restore-db-cluster-to-point-in-time`

Amazon DocumentDB 스토리지 볼륨	세 개의 가용 영역에 걸쳐 여섯 가지 방식으로 데이터를 복제하는 가용성과 내구성이 뛰어난 스토리지 볼륨입니다. Amazon DocumentDB 클러스터는 클러스터의 인스턴스 수에 관계없이 내구성이 우수합니다.	`create-db-cluster` `delete-db-cluster`
백업 기간	자동 스냅샷이 생성	`create-db-cluster`

명사	설명	API(동사)

	되는 기간(일)입니다.	`describe-db-cluster`
		
		`modify-db-cluster`

자동 스냅샷	Amazon DocumentDB에서 지속적 백업 프로세스에 의해 자동으로 생성되고 클러스터 전체를 백업하는 일별 스냅샷입니다.	`restore-db-cluster-from-snapshot`
		
		`describe-db-cluster-snapshot-attributes`
		
		`describe-db-cluster-snapshots`

수동 스냅샷	백업 보존 기간 이후에 클러스터의 전체 백업을 보관하기 위해 수동으로 생성하는 스냅샷입니다.	`create-db-cluster-snapshot`
		
		`copy-db-cluster-snapshot`
		
		`delete-db-cluster-snapshot`
		
		`describe-db-cluster-snapshot-attributes`
		
		`describe-db-cluster-snapshots`
		
		`modify-db-cluster-snapshot-attribute`

백업 스토리지 사용량 파악

Amazon DocumentDB 백업 스토리지는 백업 보존 기간 내 연속 백업과 보존 기간이 경과된 수동 스냅샷으로 구성됩니다. 백업 스토리지 사용량을 제어하려면 백업 보존 간격을 줄이거나 더 이상 필요하지 않은 경우 이전 수동 스냅샷을 제거하거나, 또는 두 가지 방법을 모두 사용할 수 있습니다. Amazon DocumentDB 백업에 대한 일반적인 정보는 Amazon DocumentDB에서 백업 및 복원 섹션을 참조하세

요. Amazon DocumentDB 백업 스토리지에 대한 요금 정보는 Amazon DocumentDB 요금 웹 페이지를 참조하세요.

비용을 제어하기 위해 연속 백업과 수동 스냅샷(보존 기간 이후에도 계속 유지됨)에 사용되는 스토리지의 양을 모니터링할 수 있습니다. 그런 다음 백업 보존 간격을 줄이거나 더 이상 필요하지 않은 경우 수동 스냅샷을 제거할 수 있습니다.

다음과 같이 Amazon CloudWatch 지표인 `TotalBackupStorageBilled`, `SnapshotStorageUsed` 및 `BackupRetentionPeriodStorageUsed`를 사용하여 Amazon DocumentDB 백업에 사용되는 스토리지의 양을 검토 및 모니터링할 수 있습니다.

- `BackupRetentionPeriodStorageUsed`는 현재 연속 백업을 저장하는 데 사용되는 백업 스토리지 양을 나타냅니다. 이 지표 값은 보존 기간 중에 발생하는 변경의 양과 클러스터 볼륨의 크기에 따라 달라집니다. 하지만, 결제 목적상, 이 지표는 보존 기간 동안 누적 클러스터 볼륨 크기를 초과하지 않습니다. 예를 들어 클러스터 크기가 100GiB이고 보존 기간이 2일이면 `BackRetentionPeriodStorageUsed`에 대한 최대 값은 200GiB(100GiB+100GiB)입니다.

- `SnapshotStorageUsed`는 백업 보존 기간 이후에 수동 스냅샷을 저장하는 데 사용되는 백업 스토리지의 양을 나타냅니다. 보존 기간 내에서 생성된 수동 스냅샷은 백업 스토리지 계산에 포함되지 않습니다. 마찬가지로, 자동 스냅샷도 백업 스토리지 계산에 포함되지 않습니다. 각 스냅샷의 크기는 스냅샷을 생성할 당시의 클러스터 볼륨 크기입니다. `SnapshotStorageUsed` 값은 유지하는 스냅샷 수와 각 스냅샷의 크기에 따라 달라집니다. 예를 들어 보존 기간 이후에 계속 유지되는 스냅샷이 하나 있고 해당 스냅샷을 생성할 당시의 클러스터 볼륨 크기가 100GiB라고 가정해 보겠습니다. `SnapshotStorageUsed`의 양은 100GiB입니다.

- `TotalBackupStorageBilled`는 `BackupRetentionPeriodStorageUsed` 및 `SnapshotStorageUsed`의 합계 - 당일의 클러스터 볼륨 크기에 해당하는 사용 가능한 백업 스토리지의 양을 나타냅니다. 예를 들어 클러스터 크기가 100GiB이고 보존 기간이 하루이며 해당 보존 기간 이후에도 계속 유지되는 스냅샷이 한 개 있는 경우, `TotalBackupStorageBilled`는 100GiB(100GiB+100GiB-100GiB)입니다.

- 이러한 지표는 각 Amazon DocumentDB 클러스터에 대해 독립적으로 계산됩니다.

CloudWatch 콘솔을 통해 CloudWatch 지표를 사용하여 Amazon DocumentDB 클러스터를 모니터링하고 보고서를 작성할 수 있습니다. CloudWatch 지표를 사용하는 방법에 대한 자세한 내용은 Amazon DocumentDB 모니터링 섹션을 참조하세요.

데이터 덤프, 복원, 가져오기 및 내보내기

`mongodump`, `mongorestore`, `mongoexport` 및 `mongoimport` 유틸리티를 사용하여 Amazon DocumentDB 클러스터 내부 및 외부로 데이터를 이동할 수 있습니다. 이 섹션에서는 이러한 각 도구의 목적과 성능을 높일 수 있는 구성에 대해 설명합니다.

주제

- mongodump
- mongorestore
- mongoexport
- mongoimport
- 자습서

mongodump

`mongodump` 유틸리티는 MongoDB 데이터베이스의 이진(BSON) 백업을 생성합니다. `mongodump` 도구는 데이터를 이진 형식으로 저장하여 실현되는 크기 효율성 때문에 Amazon DocumentDB 클러스터로 복원하려는 경우 소스 MongoDB 배포에서 데이터를 덤프하는 권장 방법입니다.

명령을 수행하는 데 사용하는 인스턴스 또는 시스템에서 사용할 수 있는 리소스에 따라 `--numParallelCollections` 옵션을 사용하여 기본 1에서 덤프된 병렬 컬렉션 수를 `mongodump` 늘려 속도를 높일 수 있습니다. 일반적으로 Amazon DocumentDB 클러스터의 기본 인스턴스에서 vCPU당 하나의 작업자로 시작하는 것이 좋습니다.

> **Note**
>
> Amazon DocumentDB의 경우 버전 100.6.1까지의 MongoDB 데이터베이스 도구를 사용하는 것이 좋습니다. 여기에서 MongoDB 데이터베이스 도구 다운로드에 액세스할 수 있습니다.

사용 예

다음은 Amazon DocumentDB cluster 클러스터 `sample-cluster`에서 mongodump 유틸리티를 사용하는 예입니다.

```
mongodump --ssl \
    --host="sample-cluster.node.us-east-1.docdb.amazonaws.com:27017" \
    --collection=sample-collection \
    --db=sample-database \
    --out=sample-output-file \
    --numParallelCollections 4  \
    --username=sample-user \
    --password=abc0123 \
    --sslCAFile global-bundle.pem
```

mongorestore

mongorestore 유틸리티를 사용하면 mongodump 유틸리티로 생성한 데이터베이스의 이진(BSON) 백업을 복원할 수 있습니다. 복원 시 `--numInsertionWorkersPerCollection` 옵션을 사용해 각 컬렉션의 작업자 수(기본값 1)를 늘려 복원 성능을 개선할 수 있습니다. 일반적으로 Amazon DocumentDB 클러스터의 기본 인스턴스에서 vCPU당 하나의 작업자로 시작하는 것이 좋습니다.

사용 예

다음은 Amazon DocumentDB cluster 클러스터 `sample-cluster`에서 mongorestore 유틸리티를 사용하는 예입니다.

```
mongorestore --ssl \
    --host="sample-cluster.node.us-east-1.docdb.amazonaws.com:27017" \
    --username=sample-user \
    --password=abc0123 \
    --sslCAFile global-bundle.pem <fileToBeRestored>
```

mongoexport

mongoexport 도구는 Amazon DocumentDB의 데이터를 JSON, CSV 또는 TSV 파일 형식으로 내보냅니다. mongoexport 도구는 사람 또는 기계가 읽어야 하는 데이터를 내보내는 권장 방법입니다.

> **Note**
>
> mongoexport는 병렬 내보내기를 직접 지원하지 않습니다. 그러나 다양한 컬렉션에 대해 여러 mongoexport 작업을 동시에 실행하여 성능을 높일 수 있습니다.

사용 예

다음은 Amazon DocumentDB cluster 클러스터 sample-cluster에서 mongoexport 도구를 사용하는 예입니다.

```
mongoexport --ssl \
    --host="sample-cluster.node.us-east-1.docdb.amazonaws.com:27017" \
    --collection=sample-collection \
    --db=sample-database \
    --out=sample-output-file \
    --username=sample-user \
    --password=abc0123 \
    --sslCAFile global-bundle.pem
```

mongoimport

mongoimport 도구는 JSON, CSV 또는 TSV 파일의 내용을 Amazon DocumentDB 클러스터로 가져옵니다. --numInsertionWorkers 파라미터를 사용하여 가져오기 작업을 병렬화하고 속도를 높일 수 있습니다(기본값은 1).

사용 예

다음은 Amazon DocumentDB cluster 클러스터 sample-cluster에서 mongoimport 도구를 사용하는 예입니다.

```
mongoimport --ssl \
    --host="sample-cluster.node.us-east-1.docdb.amazonaws.com:27017" \
    --collection=sample-collection \
    --db=sample-database \
    --file=<yourFile> \
    --numInsertionWorkers 4 \
    --username=sample-user \
    --password=abc0123 \
    --sslCAFile global-bundle.pem
```

자습서

다음 자습서에서는 mongodump, mongorestore, mongoexport 및 mongoimport 유틸리티를 사용하여 Amazon DocumentDB 클러스터 내/외부로 데이터를 이동하는 방법에 대해 설명합니다.

1. 사전 조건 — 시작하기 전에 Amazon DocumentDB 클러스터가 프로비저닝되고 클러스터와 동일한 VPC의 Amazon EC2 인스턴스에 액세스할 수 있는지 확인하세요. 자세한 내용은 Amazon EC2를 사용하여 연결 단원을 참조하십시오.

   mongo 유틸리티 도구를 사용하려면 다음과 같이 mongodb-org-tools 패키지가 EC2 인스턴스에 설치되어 있어야 합니다.

   ```
   sudo yum install mongodb-org-tools-4.0.18
   ```

   Amazon DocumentDB는 기본적으로 전송 계층 보안(TLS) 암호화를 사용하므로 mongo 쉘을 사용하여 연결하려면 다음과 같이 Amazon RDS 인증 기관(CA) 파일도 다운로드해야 합니다.

   ```
   wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem
   ```

2. 샘플 데이터 다운로드 — 이 자습서에서는 식당에 대한 정보가 포함된 샘플 데이터를 다운로드합니다.

   ```
   wget https://raw.githubusercontent.com/ozlerhakan/mongodb-json-files/master/
   datasets/restaurant.json
   ```

3. 샘플 데이터를 Amazon DocumentDB로 가져오기 — 데이터가 논리적 JSON 형식이므로 mongoimport유틸리티를 사용하여 데이터를 Amazon DocumentDB 클러스터로 가져옵니다.

   ```
   mongoimport --ssl \
       --host="tutorialCluster.amazonaws.com:27017" \
       --collection=restaurants \
       --db=business \
       --file=restaurant.json \
       --numInsertionWorkers 4 \
       --username=<yourUsername> \
       --password=<yourPassword> \
       --sslCAFile global-bundle.pem
   ```

4. **mongodump**로 데이터 덤프 — 이제 Amazon DocumentDB 클러스터에 데이터가 있으므로 mongodump 유틸리티를 사용하여 해당 데이터의 이진 덤프를 가져올 수 있습니다.

```
mongodump --ssl \
    --host="tutorialCluster.us-east-1.docdb.amazonaws.com:27017"\
    --collection=restaurants \
    --db=business \
    --out=restaurantDump.bson \
    --numParallelCollections 4 \
    --username=<yourUsername> \
    --password=<yourPassword> \
    --sslCAFile global-bundle.pem
```

5. **restaurants** 컬렉션 삭제 — business 데이터베이스에서 restaurants 컬렉션을 복원하기 전에 먼저 다음과 같이 해당 데이터베이스에 이미 있는 컬렉션을 삭제해야 합니다.

```
use business
```

```
db.restaurants.drop()
```

6. **mongorestore**로 데이터 복원 — 3단계에서 데이터의 이진 덤프가 생성되었으므로 이제 mongorestore 유틸리티를 사용하여 데이터를 Amazon DocumentDB 클러스터로 복원할 수 있습니다.

```
mongorestore --ssl \
    --host="tutorialCluster.us-east-1.docdb.amazonaws.com:27017" \
    --numParallelCollections 4 \
    --username=<yourUsername> \
    --password=<yourPassword> \
    --sslCAFile global-bundle.pem restaurantDump.bson
```

7. **mongoexport**를 사용하여 데이터 내보내기 — 1단계에서 가져온 파일과 같은 JSON 파일 형식으로 클러스터에서 데이터를 내보내어 자습서를 완료합니다.

```
mongoexport --ssl \
    --host="tutorialCluster.node.us-east-1.docdb.amazonaws.com:27017" \
    --collection=restaurants \
    --db=business \
    --out=restaurant2.json \
    --username=<yourUsername> \
    --password=<yourPassword> \
    --sslCAFile global-bundle.pem
```

8. 검증 — 다음 명령을 사용하여 5단계의 출력이 1단계와 동일한 결과를 내는지 검증할 수 있습니다.

```
wc -l restaurant.json
```

이 명령의 출력:

```
2548 restaurant.json
```

```
wc -l restaurant2.json
```

이 명령의 출력:

```
2548 restaurant2.json
```

클러스터 스냅샷 고려 사항

Amazon DocumentDB는 클러스터 백업 기간 동안 매일 클러스터의 자동 스냅샷을 생성합니다. Amazon DocumentDB는 사용자가 지정한 백업 보존 기간에 따라 클러스터의 자동 스냅샷을 저장합니다. 필요할 경우 백업 보존 기간 중 어느 특정 시점으로든 클러스터를 복구할 수 있습니다. 동일한 리전에서 동일 클러스터에 대한 복사 작업을 수행하고 있는 동안에는 자동 스냅샷이 생성되지 않습니다.

주제

- 백업 스토리지
- 백업 기간
- 백업 보관 기간
- 클러스터 스냅샷 암호화 복사

자동 클러스터 스냅샷 이외에 클러스터 스냅샷을 수동으로 생성할 수도 있습니다. 자동 스냅샷과 수동 스냅샷을 모두 복사할 수 있습니다. 자세한 내용은 수동 클러스터 스냅샷 생성 및 Amazon DocumentDB 클러스터 스냅샷 복사 섹션을 참조하세요.

> **Note**
>
> 자동 스냅샷을 생성하려면 클러스터가 사용 가능한 상태에 있어야 합니다.

Amazon DocumentDB 자동화된 클러스터 스냅샷은 공유할 수 없습니다. 이 문제를 해결하려면 자동화된 스냅샷을 복사하여 수동 스냅샷을 만든 다음 그 복사본을 공유하면 됩니다. 스냅샷 복사에 대한 자세한 내용은 Amazon DocumentDB 클러스터 스냅샷 복사 섹션을 참조하세요. 스냅샷에서 클러스터 복원에 대한 자세한 내용은 클러스터 스냅샷에서 복원 섹션을 참조하세요.

백업 스토리지

각에 대한 Amazon DocumentDB 백업 스토리지 AWS 리전 는 백업 보존 기간에 필요한 백업 스토리지로 구성되며, 여기에는 해당 리전의 자동 및 수동 클러스터 스냅샷이 포함됩니다. 기본 백업 보존 기간은 1일입니다. 백업 스토리지에 대한 요금 정보는 Amazon DocumentDB 요금 웹 페이지를 참조하세요.

클러스터를 삭제하면 클러스터의 모든 자동 스냅샷이 삭제되고 복구할 수 없습니다. 그러나 클러스터를 삭제할 때 수동 스냅샷은 삭제되지 않습니다. 클러스터를 삭제하기 전에 Amazon DocumentDB에서 최종 스냅샷(수동 스냅샷)이 생성되도록 선택한 경우, 그 최종 스냅샷을 사용하여 클러스터를 복구할 수 있습니다.

스냅샷 및 스토리지에 대한 자세한 내용은 백업 스토리지 사용량 파악 섹션을 참조하세요.

백업 기간

자동 스냅샷은 기본 백업 기간 동안 매일 생성됩니다. 스냅샷 시간이 백업 기간에 할당된 시간보다 오래 걸릴 경우 백업 프로세스는 백업 기간이 종료한 후에도 완료 시까지 계속 실행됩니다. 백업 기간은 해당 클러스터에 대한 주간 유지 보수 기간과 겹칠 수 없습니다.

클러스터를 생성할 때 원하는 백업 기간을 지정하지 않으면 Amazon DocumentDB이 기본 30분 백업 기간을 할당합니다. 클러스터의 리전과 연결된 8시간 블록 시간 중에서 이 시간이 임의로 선택됩니다. 클러스터를 수정하여 기본 백업 기간을 변경할 수 있습니다. 자세한 내용은 Amazon DocumentDB 클러스터 수정 단원을 참조하십시오.

리전 이름	지역	UTC 시간 블록
미국 동부(오하이오)	us-east-2	03:00~11:00
미국 동부(버지니아 북부)	us-east-1	03:00~11:00
미국 서부(오리건)	us-west-2	06:00~14:00

리전 이름	지역	UTC 시간 블록
아프리카(케이프타운)	af-south-1	03:00~11:00
아시아 태평양(홍콩)	ap-east-1	06:00~14:00
아시아 태평양(하이데라바드)	ap-south-2	06:30-14:30
아시아 태평양(뭄바이)	ap-south-1	06:00~14:00
아시아 태평양(서울)	ap-northeast-2	13:00~21:00
아시아 태평양(싱가포르)	ap-southeast-1	14:00–22:00
아시아 태평양(시드니)	ap-southeast-2	12:00–20:00
아시아 태평양(도쿄)	ap-northeast-1	13:00~21:00
캐나다(중부)	ca-central-1	03:00~11:00
중국(베이징)	cn-north-1	06:00~14:00
중국(닝샤)	cn-northwest-1	06:00~14:00
유럽(프랑크푸르트)	eu-central-1	21:00~05:00
유럽(아일랜드)	eu-west-1	22:00~06:00
유럽(런던)	eu-west-2	22:00~06:00
유럽(밀라노)	eu-south-1	02:00~10:00
유럽(파리)	eu-west-3	23:59~07:29
유럽(스페인)	eu-south-2	02:00~10:00
유럽(스톡홀름)	eu-north-1	04:00~12:00
중동(UAE)	me-central-1	05:00~13:00
남아메리카(상파울루)	sa-east-1	00:00~08:00

리전 이름	지역	UTC 시간 블록
이스라엘(텔아비브)	il-central-1	04:00-12:00
AWS GovCloud(미국 동부)	us-gov-east-1	17:00–01:00
AWS GovCloud(미국 서부)	us-gov-west-1	06:00~14:00

백업 보관 기간

백업 보존 기간은 자동 백업이 자동으로 삭제되기 이전에 보관되는 일 수입니다. Amazon DocumentDB는 1~35일의 백업 보존 기간을 지원합니다.

클러스터를 생성하면서 백업 보존 기간을 설정할 수 있습니다. 백업 보존 기간을 명시적으로 설정하지 않은 경우 기본 백업 보존 기간 1일이 클러스터에 할당됩니다. 클러스터를 생성한 후 AWS Management Console 또는를 사용하여 클러스터를 수정하여 백업 보존 기간을 수정할 수 있습니다 AWS CLI. 자세한 내용은 Amazon DocumentDB 클러스터 수정 단원을 참조하십시오.

클러스터 스냅샷 암호화 복사

클러스터 및 스냅샷 암호화는 KMS 암호화 키를 기반으로 합니다. KMS 키 ID는 Amazon 리소스 이름(ARN), KMS 키 식별자 또는 KMS 암호화 키에 대한 KMS 키 별칭입니다.

다음과 같은 지침과 제한 사항이 적용됩니다.

- 스냅샷을 생성할 때 클러스터에서 암호화가 추론됩니다. 클러스터가 암호화된 경우 클러스터의 스냅샷은 동일한 KMS 키로 암호화됩니다. 클러스터가 암호화되지 않으면 스냅샷은 암호화되지 않습니다.
- Amazon Web Services 계정에서 암호화된 클러스터 스냅샷을 복사하는 경우, `KmsKeyId` 값을 지정하여 새 KMS 암호화 키로 사본을 암호화할 수 있습니다. `KmsKeyId` 값을 지정하지 않으면 DB 클러스터 스냅샷의 사본을 원본 DB 스냅샷과 동일한 KMS 키로 암호화합니다.
- 다른 Amazon Web Services 계정에서 공유한 암호화된 DB 클러스터 스냅샷을 복사하는 경우, `KmsKeyId` 값을 지정해야 합니다.
- 암호화된 클러스터 스냅샷을 다른 Amazon Web Services 지역에 `KmsKeyId` 복사하려면 대상 지역의 클러스터 스냅샷 사본을 암호화하는 데 사용할 KMS 키 ID를 설정합니다. KMS 암호화 키는 해당 키를 만든 Amazon Web Services 리전에 고유하며, 한 Amazon Web Services 리전의 암호화 키를 다른 Amazon Web Services 리전에서 사용할 수는 없습니다.

- 암호화되지 않은 클러스터 스냅샷의 복사하고 `KmsKeyId` 파라미터에 대한 값을 지정하려고 시도하면 오류가 반환됩니다.

자동 스냅샷과 수동 스냅샷 비교

다음은 Amazon DocumentDB(MongoDB 호환) 자동 스냅샷 및 수동 스냅샷의 주요 특성입니다.

Amazon DocumentDB 자동 스냅샷의 주요 특성은 다음과 같습니다.

- 자동 스냅샷 이름 지정 — 자동 스냅샷 이름은 `rds:<cluster-name>-yyyy-mm-dd-hh-mm` 패턴을 따릅니다. 여기서 `yyyy-mm-dd-hh-mm`은 스냅샷이 생성된 날짜 및 시간을 나타냅니다.
- 일정에 따라 자동으로 생성됨 — 클러스터를 생성하거나 수정할 때 백업 보존 기간을 1~35일 사이의 정수 값으로 설정할 수 있습니다. 기본적으로 새 클러스터의 백업 보존 기간은 1일입니다. 백업 보존 기간은 자동 스냅샷이 자동으로 삭제되기 이전에 보관되는 일 수를 정의합니다. Amazon DocumentDB 클러스터에 대한 자동 백업을 비활성화할 수 없습니다.

  백업 보존 기간 설정 이외에 자동 스냅샷이 생성되는 기간(일)을 나타내는 백업 기간을 설정할 수도 있습니다.
- 자동 스냅샷 삭제 — 자동 스냅샷 클러스터를 삭제하면 자동 스냅샷이 삭제됩니다. 자동 스냅샷을 수동으로 삭제할 수 없습니다.
- 증분식 — 백업 보존 기간 동안 데이터베이스 업데이트가 증분식 변경 기록 방식으로 기록됩니다.
- 자동 스냅샷에서 복원 — AWS Management Console 또는 AWS CLI를 사용하여 자동 스냅샷에서 복원할 수 있습니다. 를 사용하여 스냅샷에서 복원 AWS CLI할 때는 클러스터를 사용할 수 있게 된 후 인스턴스를 별도로 추가해야 합니다.
- 공유 — Amazon DocumentDB의 자동 클러스터 스냅샷은 공유할 수 없습니다. 이 문제를 해결하려면 자동화된 스냅샷을 복사하여 수동 스냅샷을 만든 다음 그 복사본을 공유하면 됩니다. 스냅샷 복사에 대한 자세한 내용은 Amazon DocumentDB 클러스터 스냅샷 복사 섹션을 참조하세요. 스냅샷에서 클러스터 복원에 대한 자세한 내용은 클러스터 스냅샷에서 복원 섹션을 참조하세요.
- 백업 보존 기간 내의 어떤 시점으로든 복원 가능 — 데이터베이스 업데이트가 증분식으로 기록되므로 백업 보존 기간 내의 어떤 시점으로든 복원할 수 있습니다.

  를 사용하여 자동 스냅샷 또는 point-in-time으로 복원에서 복원 AWS CLI하는 경우 클러스터를 사용할 수 있게 된 후 인스턴스를 별도로 추가해야 합니다.

Amazon DocumentDB 수동 스냅샷의 주요 특성은 다음과 같습니다.

- 온디맨드 생성 - Amazon DocumentDB 수동 스냅샷은 Amazon DocumentDB 관리 콘솔 또는를 사용하여 온디맨드로 생성됩니다 AWS CLI.
- 수동 스냅샷 삭제 — 수동 Amazon DocumentDB 스냅샷은 콘솔 또는 AWS CLI를 사용하여 명시적으로 삭제한 경우에만 삭제됩니다. 클러스터를 삭제해도 수동 스냅샷은 삭제되지 않습니다.
- 전체 백업 — 수동 스냅샷을 생성하면 클러스터 데이터의 전체 백업이 생성된 후 저장됩니다.
- 수동 스냅샷 이름 지정 — 수동 스냅샷 이름을 지정합니다. Amazon DocumentDB는 이름에 `datetime` 스탬프를 추가하지 않으므로 이름에 포함시키려면 해당 정보를 추가해야 합니다.
- 수동 스냅샷에서 복원 — 콘솔 또는 AWS CLI를 사용하여 수동 스냅샷에서 복원할 수 있습니다. 를 사용하여 스냅샷에서 복원 AWS CLI할 때는 클러스터를 사용할 수 있게 된 후 인스턴스를 별도로 추가해야 합니다.
- Service Quotas - 당 최대 100개의 수동 스냅샷으로 제한됩니다 AWS 리전.
- 공유 — 수동 클러스터 스냅샷을 공유할 수 있고, 권한 있는 AWS 계정에서 이를 복사할 수 있습니다. 암호화되거나 암호화되지 않은 수동 스냅샷을 공유할 수 있습니다. 스냅샷 복사에 대한 자세한 내용은 Amazon DocumentDB 클러스터 스냅샷 복사 섹션을 참조하세요.
- 수동 스냅샷 생성 시점으로 복원 — 수동 스냅샷에서 복원하면 수동 스냅샷이 생성된 시점으로 복원됩니다.

  를 사용하여 스냅샷에서 복원 AWS CLI할 때는 클러스터를 사용할 수 있게 된 후 인스턴스를 별도로 추가해야 합니다.

수동 클러스터 스냅샷 생성

AWS Management Console 또는를 사용하여 수동 스냅샷을 생성할 수 있습니다 AWS CLI. 스냅샷을 생성하는 데 걸리는 시간은 데이터베이스 크기에 따라 다릅니다. 스냅샷을 생성하는 경우 다음을 수행해야 합니다.

1. 백업할 클러스터를 식별합니다.
2. 스냅샷의 이름을 지정합니다. 나중에 이 스냅샷에서 복원할 수 있습니다.

Using the AWS Management Console

를 사용하여 수동 스냅샷을 생성하려면 아래 방법 중 하나를 따를 AWS Management Console수 있습니다.

1. 방법 1:

   1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
   2. 탐색 창에서 스냅샷를 선택합니다.

      > **Tip**
      >
      > 화면 왼쪽에 탐색 창이 표시되지 않으면 페이지 왼쪽 상단 모서리에서 메뉴 아이콘 (≡) 을 선택하세요.

   3. 스냅샷 페이지에서 생성을 선택합니다.
   4. 클러스터 스냅샷 생성 페이지에서:

      a. 클러스터 식별자 — 클러스터의 드롭다운 목록에서 스냅샷을 생성할 클러스터를 선택합니다.

      b. 스냅샷 식별자 — 스냅샷의 이름을 입력합니다.

      스냅샷 명명 제약 조건:

      - 길이는 [1~255]글자, 숫자 또는 하이픈입니다.
      - 첫 번째 문자는 글자이어야 합니다.
      - 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다.
      - AWS 계정, 리전별로 모든 클러스터 (Amazon RDS, Amazon Neptune 및 Amazon DocumentDB)에 대해 고유해야 합니다.

      c. 생성(Create)을 선택합니다.

2. 방법 2:

   1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
   2. 탐색 창에서 클러스터를 선택합니다.

> **Tip**
>
> 화면 왼쪽에 탐색 창이 표시되지 않으면 페이지 왼쪽 상단 모서리에서 메뉴 아이콘 (≡) 을 선택하세요.

3. 클러스터 페이지에서 스냅샷을 생성하려는 클러스터의 왼쪽에 있는 버튼을 선택합니다.
4. 작업 메뉴에서 스냅샷 만들기을 선택합니다.
5. 클러스터 스냅샷 생성 페이지에서:
   a. 스냅샷 식별자 — 스냅샷의 이름을 입력합니다.

      스냅샷 명명 제약 조건:

      - 길이는 [1~63] 글자, 숫자 또는 하이픈입니다.
      - 첫 번째 문자는 글자이어야 합니다.
      - 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다.
      - AWS 계정, 리전별로 모든 클러스터 (Amazon RDS, Amazon Neptune 및 Amazon DocumentDB)에 대해 고유해야 합니다.

   b. 생성(Create)을 선택합니다.

Using the AWS CLI

를 사용하여 클러스터 스냅샷을 생성하려면 다음 파라미터와 함께 `create-db-cluster-snapshot` 작업을 AWS CLI사용합니다.

파라미터

- **`--db-cluster-identifier`** - 필수입니다. 스냅샷을 생성할 클러스터의 이름입니다. 이 클러스터가 있고 사용 가능해야 합니다.
- **`--db-cluster-snapshot-identifier`** - 필수입니다. 생성하려는 수동 스냅샷의 이름입니다.

다음 예제에서는 `sample-cluster`라는 클러스터에 대해 `sample-cluster-snapshot`이라는 스냅샷을 만듭니다.

Linux, macOS, Unix의 경우:

```
aws docdb create-db-cluster-snapshot \
    --db-cluster-identifier sample-cluster \
    --db-cluster-snapshot-identifier sample-cluster-snapshot
```

Windows의 경우:

```
aws docdb create-db-cluster-snapshot ^
    --db-cluster-identifier sample-cluster ^
    --db-cluster-snapshot-identifier sample-cluster-snapshot
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
{
    "DBClusterSnapshot": {
        "AvailabilityZones": [
            "us-east-1a",
            "us-east-1b",
            "us-east-1c"
        ],
        "DBClusterSnapshotIdentifier": "sample-cluster-snapshot",
        "DBClusterIdentifier": "sample-cluster",
        "SnapshotCreateTime": "2020-04-24T04:59:08.475Z",
        "Engine": "docdb",
        "Status": "creating",
        "Port": 0,
        "VpcId": "vpc-abc0123",
        "ClusterCreateTime": "2020-01-10T22:13:38.261Z",
        "MasterUsername": "master-user",
        "EngineVersion": "4.0.0",
        "SnapshotType": "manual",
        "PercentProgress": 0,
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:us-east-1:<accountID>:key/sample-key",
        "DBClusterSnapshotArn": "arn:aws:rds:us-east-1:<accountID>:cluster-
snapshot:sample-cluster-snapshot"
    }
}
```

Amazon DocumentDB 클러스터 스냅샷 복사

Amazon DocumentDB에서는 동일한 내 AWS 리전 또는 동일한 계정 AWS 리전 내 다른에 수동 및 자동 스냅샷을 복사할 수 있습니다. 동일한 AWS 계정 에서 다른이 소유한 스냅샷을 공유할 수도 있습니다 AWS 리전. 그러나 클러스터 스냅샷은 한 번에 전체 AWS 리전 및 AWS 계정 에 복사할 수 없습니다. 이러한 작업은 개별적으로 수행해야 합니다.

복사 대신 수동 스냅샷을 다른와 공유할 수도 있습니다 AWS 계정. 자세한 내용은 Amazon DocumentDB 클러스터 스냅샷 공유 단원을 참조하십시오.

> **Note**
>
> Amazon DocumentDB은 유지하는 백업 및 스냅샷 데이터의 양과 유지 기간에 따라 요금을 청구합니다. Amazon DocumentDB 백업 및 스냅샷과 연결된 스토리지에 대한 자세한 내용은 백업 스토리지 사용량 파악 섹션을 참조하세요. Amazon DocumentDB 스토리지에 대한 요금 정보는 Amazon DocumentDB 요금 섹션을 참조하세요.

주제

- 공유 스냅샷 복사
- 에서 스냅샷 복사 AWS 리전
- 제한 사항
- 암호화 처리
- 파라미터 그룹 고려 사항
- 클러스터 스냅샷 복사

공유 스냅샷 복사

다른 사용자가 공유한 스냅샷을 복사할 수 있습니다 AWS 계정. 다른에서 공유된 암호화된 스냅샷을 복사하는 경우 스냅샷을 암호화하는 데 사용된 AWS KMS 암호화 키에 액세스할 수 AWS 계정있어야 합니다.

스냅샷 암호화 여부에 AWS 리전관계없이 동일한 에서만 공유 스냅샷을 복사할 수 있습니다. 자세한 내용은 암호화 처리 단원을 참조하십시오.

에서 스냅샷 복사 AWS 리전

소스 스냅샷의와 AWS 리전 다른에 스냅샷을 복사하는 경우 AWS 리전각 복사본은 전체 스냅샷입니다. 전체 스냅샷 복사에는 Amazon DocumentDB 클러스터 복원에 필요한 모든 데이터 및 메타데이터가 포함됩니다.

AWS 리전 관련 및 복사할 데이터의 양에 따라 리전 간 스냅샷 복사를 완료하는 데 몇 시간이 걸릴 수 있습니다. 경우에 따라 지정된 소스 AWS 리전으로부터 대량의 크로스 리전 스냅샷 복사 요청이 있을 수 있습니다. 이러한 경우 Amazon DocumentDB는 진행 중인 일부 복사본이 완료될 때까지 해당 소스의 새 리전 간 복사 요청을 대기열 AWS 리전 에 넣을 수 있습니다. 대기열에 있는 복사 요청에 대한 진행 정보는 표시되지 않습니다. 복사가 시작되면 진행 정보가 표시됩니다.

제한 사항

다음은 스냅샷을 복사할 때 적용되는 몇몇 제한 사항입니다.

- 대상 스냅샷이 사용 가능해지기 전에 원본 스냅샷을 삭제할 경우 스냅샷 복사가 실패할 수 있습니다. 원본 스냅샷을 삭제하기 전에 대상 스냅샷의 상태가 `AVAILABLE`인지 확인하세요.
- 계정당 단일 대상 리전으로 최대 5개의 스냅샷 복사 요청이 진행될 수 있습니다.
- 관련된 리전과 복사할 데이터 양에 따라 리전 간 스냅샷 복사를 완료하는 데 몇 시간이 걸릴 수 있습니다. 자세한 내용은 에서 스냅샷 복사 AWS 리전 단원을 참조하십시오.

암호화 처리

AWS KMS 암호화 키를 사용하여 암호화된 스냅샷을 복사할 수 있습니다. 암호화된 스냅샷을 복사할 경우 스냅샷의 사본도 암호화해야 합니다. 동일한 내에서 암호화된 스냅샷을 복사 AWS 리전하는 경우 원본 스냅샷과 동일한 암호화 키로 복사본을 AWS KMS 암호화하거나 다른 AWS KMS 암호화 키를 지정할 수 있습니다. 리전 간에 암호화된 스냅샷을 복사하는 경우 키는 리전별로 다르므로 원본 스냅샷에 사용된 것과 동일한 AWS KMS 암호화 AWS KMS 키를 복사에 사용할 수 없습니다. 대신 대상 AWS 리전 n에 유효한 AWS KMS 키를 지정해야 합니다.

소스 스냅샷은 복사 프로세스 전체에서 암호화를 유지합니다. 자세한 내용은 Amazon DocumentDB의 데이터 보호 단원을 참조하십시오.

> **Note**
>
> Amazon DocumentDB 클러스터 스냅샷의 경우 암호화되지 않은 DB 클러스터 스냅샷은 스냅샷을 복사할 때 암호화할 수 없습니다.

파라미터 그룹 고려 사항

리전 간에 스냅샷을 복사하는 경우 복사에는 원래 Amazon DocumentDB 클러스터에서 사용된 파라미터 그룹이 포함되지 않습니다. 스냅샷을 복원하여 새 클러스터를 생성하면 해당 클러스터 AWS 리전는 생성된에 대한 기본 파라미터 그룹을 가져옵니다. 새 클러스터에 원본과 같은 파라미터를 지정하려면 다음을 수행해야 합니다.

1. 대상에서 원래 클러스터와 동일한 설정으로 Amazon DocumentDB 클러스터 파라미터 그룹을 AWS 리전생성합니다. Amazon DocumentDB 새에 이미 있는 경우 해당를 사용할 AWS 리전수 있습니다.
2. 대상에서 스냅샷을 복원한 후 새 Amazon DocumentDB 클러스터를 AWS 리전수정하고 이전 단계에서 새 파라미터 그룹 또는 기존 파라미터 그룹을 추가합니다. 자세한 내용은 Amazon DocumentDB 클러스터 수정 단원을 참조하십시오.

클러스터 스냅샷 복사

다음과 AWS CLI같이 AWS Management Console 또는를 사용하여 Amazon DocumentDB 클러스터를 복사할 수 있습니다.

Using the AWS Management Console

를 사용하여 클러스터 스냅샷의 복사본을 만들려면 다음 단계를 AWS Management Console완료합니다. 이 절차는 동일한 리전 또는 여러 리전에서 암호화된 클러스터 스냅샷 AWS 리전 또는 암호화되지 않은 클러스터 스냅샷을 복사하는 데 사용됩니다.

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 스냅샷을 선택한 다음 복사하려는 스냅샷 왼쪽에 있는 버튼을 선택합니다.

   > **Tip**
   > 화면 왼쪽에 탐색 창이 표시되지 않으면 페이지 왼쪽 상단 모서리에서 메뉴 아이콘 (≡) 을 선택하세요.

3. 작업 메뉴에서 복사를 선택합니다.
4. 표시되는 클러스터 스냅샷 복사본 만들기 페이지에서 설정 섹션을 완료하세요.

a. 대상 지역 — 선택 사항. 클러스터 스냅샷을 다른에 복사하려면 대상 리전 AWS 리전 에서 해당 스냅샷을 AWS 리전선택합니다.

b. 새 스냅샷 식별자 — 새 스냅샷의 이름을 입력합니다.

대상 스냅샷 명명 제약 조건:

- 기본 스냅샷의 이름일 수 없습니다.
- 길이는 [1~63]글자, 숫자 또는 하이픈입니다.
- 첫 번째 문자는 글자이어야 합니다.
- 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다.
- 리전별로 Amazon RDS, Neptune 및 Amazon DocumentDB의 모든 클러스터 AWS 계정에 대해 고유해야 합니다.

c. 태그 복사 — 소스 스냅샷에 있는 태그를 스냅샷 복사본으로 복사하려면 태그 복사를 선택합니다.

5. 유휴 상태에서 암호화 섹션을 완료합니다.

a. 유휴 상태에서 암호화 — 암호화된 스냅샷에서 암호화되지 않은 복사본을 생성할 수 없으므로 스냅샷이 암호화되면 이 옵션을 사용할 수 없습니다. 스냅샷이 암호화된 경우 저장 시 암호화 중에 AWS KMS key 사용되는를 변경할 수 있습니다.

암호화된 스냅샷 복사에 대한 자세한 내용은 클러스터 스냅샷 암호화 복사 섹션을 참조하세요.

저장된 암호화에 대한 자세한 내용은 Amazon DocumentDB 저장 데이터 암호화 섹션을 참조하세요.

b. AWS KMS 키 - 드롭다운 목록에서 다음 중 하나를 선택합니다.

- (기본값) aws/rds - 계정 번호와 AWS KMS 키 ID가이 옵션 다음에 나열됩니다.
- <some-key-name> — 키를 생성하면 해당 키가 나열되고 이를 선택할 수 있습니다.
- 키 ARN 입력 — ARN 상자에 AWS KMS 키의 Amazon 리소스 이름(ARN)을 입력합니다. ARN 형식은 `arn:aws:kms:<region>:<accountID>:key/<key-id>` 입니다.

6. 선택한 스냅샷을 복사하려면 스냅샷 복사를 선택합니다. 선택한 스냅샷을 복사하지 않으려면 취소를 선택합니다.

Using the AWS CLI

클러스터 스냅샷을 복사하려면 AWS CLI다음 파라미터와 함께 `copy-db-cluster-snapshot` 작업을 사용합니다. 스냅샷을 다른에 복사하는 경우 스냅샷을 복사 AWS 리전 할에서 명령을 AWS 리전실행합니다.

- **`--source-db-cluster-snapshot-identifier`** — 필수입니다. 복사할 클러스터 스냅샷의 식별자입니다. 클러스터 스냅샷이 있어야 하며 사용 가능한 상태여야 합니다. 스냅샷을 다른에 복사하는 경우 AWS 리전이 식별자는 소스의 ARN 형식이어야 합니다 AWS 리전. 이 파라미터는 대소문자를 구분하지 않습니다.
- **`--target-db-cluster-snapshot-identifier`** — 필수입니다. 소스 클러스터 스냅샷에서 생성할 새 클러스터 스냅샷의 식별자입니다. 이 파라미터는 대소문자를 구분하지 않습니다.

  대상 스냅샷 명명 제약 조건:

  - 기본 스냅샷의 이름일 수 없습니다.
  - 길이는 [1~63]글자, 숫자 또는 하이픈입니다.
  - 첫 번째 문자는 글자이어야 합니다.
  - 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다.
  - 리전별로 Amazon RDS, Neptune 및 Amazon DocumentDB의 모든 클러스터 AWS 계정에 대해 고유해야 합니다.
- **`--source-region`** - 스냅샷을 다른에 복사하는 경우 암호화된 클러스터 스냅샷을 복사 AWS 리전 할를 AWS 리전지정합니다.

  스냅샷을 다른 AWS 리전 리전에 복사하려고 하며 `--source-region`을 지정하지 않은 경우, 대신 `pre-signed-url` 옵션을 지정해야 합니다. `pre-signed-url` 값은 AWS 리전 클러스터 스냅샷이 복사되는 소스에서 `CopyDBClusterSnapshot` 작업을 호출하기 위한 서명 버전 4 서명 요청이 포함된 URL이어야 합니다. `pre-signed-url`에 대한 자세한 내용은 CopyDbClusterSnapshot을 참조하세요.
- **`--kms-key-id`** — DB 클러스터 스냅샷의 복사본을 암호화하는 데 사용할 키의 KMS 키 식별자입니다.

  암호화된 클러스터 스냅샷을 다른에 복사하는 경우 AWS 리전이 파라미터가 필요합니다. 대상에 대한 KMS 키를 지정해야 합니다 AWS 리전.

  동일한에서 암호화된 클러스터 스냅샷을 복사하는 경우 AWS 리전 AWS KMS 키 파라미터는 선택 사항입니다. 클러스터 스냅샷의 사본은 소스 클러스터 스냅샷과 동일한 AWS KMS 키로 암호

화됩니다. 복사본 AWS KMS 을 암호화하는 데 사용할 새 암호화 키를 지정하려면이 파라미터를 사용하면 됩니다.

- **`--copy-tags`** - 선택 사항입니다. 복사할 태그 및 값입니다.

진행 중인 복사 작업을 취소하려면 대상 클러스터 스냅샷이 복사 상태에 있는 동안 `--target-db-cluster-snapshot-identifier` 또는 `TargetDBClusterSnapshotIdentifier`로 식별된 해당 DB 클러스터 스냅샷을 삭제합니다.

Example

예 1: 암호화되지 않은 스냅샷을 동일한 리전에 복사

다음 AWS CLI 예시에서는 소스 스냅샷 AWS 리전 과 동일한 `sample-cluster-snapshot-copy`에 `sample-cluster-snapshot` 라는의 복사본을 생성합니다. 복사가 완료되면 원래 스냅샷의 모든 태그가 스냅샷 사본에 복사됩니다.

Linux, macOS 또는 Unix의 경우:

```
aws docdb copy-db-cluster-snapshot \
    --source-db-cluster-snapshot-identifier sample-cluster-snapshot \
    --target-db-cluster-snapshot-identifier sample-cluster-snapshot-copy \
    --copy-tags
```

Windows의 경우:

```
aws docdb copy-db-cluster-snapshot ^
    --source-db-cluster-snapshot-identifier sample-cluster-snapshot ^
    --target-db-cluster-snapshot-identifier sample-cluster-snapshot-copy ^
    --copy-tags
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
{
    "DBClusterSnapshot": {
        "AvailabilityZones": [
            "us-east-1a",
            "us-east-1b",
            "us-east-1c"
        ],
```

```
        "DBClusterSnapshotIdentifier": "sample-cluster-snapshot-copy",
        "DBClusterIdentifier": "sample-cluster",
        "SnapshotCreateTime": "2020-03-27T08:40:24.805Z",
        "Engine": "docdb",
        "Status": "copying",
        "Port": 0,
        "VpcId": "vpc-abcd0123",
        "ClusterCreateTime": "2020-01-10T22:13:38.261Z",
        "MasterUsername": "master-user",
        "EngineVersion": "4.0.0",
        "SnapshotType": "manual",
        "PercentProgress": 0,
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:us-east-1:111122223333:key/sample-key-id",
        "DBClusterSnapshotArn": "arn:aws:rds:us-east-1:111122223333:cluster-
snapshot:sample-cluster-snapshot-copy",
        "SourceDBClusterSnapshotArn": "arn:aws:rds:us-east-1:111122223333:cluster-
snapshot:sample-cluster-snapshot"
    }
}
```

Example

예제 2: 암호화되지 않은 스냅샷 복사 AWS 리전

다음 AWS CLI 예제에서는 ARN이 `sample-cluster-snapshot`있는의 복사본을 생성합니다`arn:aws:rds:us-east-1:123456789012:cluster-snapshot:sample-cluster-snapshot`. 이 복사본의 이름은 `sample-cluster-snapshot-copy` 이며 명령이 실행되는 AWS 리전 에 있습니다.

Linux, macOS, Unix의 경우:

```
aws docdb copy-db-cluster-snapshot \
    --source-db-cluster-snapshot-identifier arn:aws:rds:us-
east-1:123456789012:cluster-snapshot:sample-cluster-snapshot \
    --target-db-cluster-snapshot-identifier sample-cluster-snapshot-copy
```

Windows의 경우:

```
aws docdb copy-db-cluster-snapshot ^
    --source-db-cluster-snapshot-identifier arn:aws:rds:us-
east-1:123456789012:cluster-snapshot:sample-cluster-snapshot ^
```

```
    --target-db-cluster-snapshot-identifier sample-cluster-snapshot-copy
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
{
    "DBClusterSnapshot": {
        "AvailabilityZones": [
            "us-east-1a",
            "us-east-1b",
            "us-east-1c"
        ],
        "DBClusterSnapshotIdentifier": "sample-cluster-snapshot-copy",
        "DBClusterIdentifier": "sample-cluster",
        "SnapshotCreateTime": "2020-04-29T16:45:51.239Z",
        "Engine": "docdb",
        "AllocatedStorage": 0,
        "Status": "copying",
        "Port": 0,
        "VpcId": "vpc-abc0123",
        "ClusterCreateTime": "2020-04-28T16:43:00.294Z",
        "MasterUsername": "master-user",
        "EngineVersion": "4.0.0",
        "LicenseModel": "docdb",
        "SnapshotType": "manual",
        "PercentProgress": 0,
        "StorageEncrypted": false,
        "DBClusterSnapshotArn": "arn:aws:rds:us-east-1:111122223333:cluster-
snapshot:sample-cluster-snapshot-copy",
        "SourceDBClusterSnapshotArn": "arn:aws:rds:us-east-1:111122223333:cluster-
snapshot:sample-cluster-snapshot",
    }
}
```

Example

예제 3:에 암호화된 스냅샷 복사 AWS 리전

다음 AWS CLI 예제에서는 us-west-2 리전`sample-cluster-snapshot`에서 us-east-1 리전으로의 사본을 생성합니다. 이 명령은 us-east-1 리전에서 호출됩니다.

Linux, macOS, Unix의 경우:

```
aws docdb copy-db-cluster-snapshot \
```

```
    --source-db-cluster-snapshot-identifier arn:aws:rds:us-
west-2:123456789012:cluster-snapshot:sample-cluster-snapshot \
    --target-db-cluster-snapshot-identifier sample-cluster-snapshot-copy \
    --source-region us-west-2 \
    --kms-key-id sample-us-east-1-key
```

Windows의 경우:

```
aws docdb copy-db-cluster-snapshot ^
    --source-db-cluster-snapshot-identifier arn:aws:rds:us-
west-2:123456789012:cluster-snapshot:sample-cluster-snapshot ^
    --target-db-cluster-snapshot-identifier sample-cluster-snapshot-copy ^
    --source-region us-west-2 ^
    --kms-key-id sample-us-east-1-key
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
{
    "DBClusterSnapshot": {
        "AvailabilityZones": [],
        "DBClusterSnapshotIdentifier": "sample-cluster-snapshot-copy",
        "DBClusterIdentifier": "ayhu-xrsc-test-ap-southeast-1-small-cluster-kms",
        "SnapshotCreateTime": "2020-04-29T16:45:53.159Z",
        "Engine": "docdb",
        "AllocatedStorage": 0,
        "Status": "copying",
        "Port": 0,
        "ClusterCreateTime": "2020-04-28T16:43:07.129Z",
        "MasterUsername": "chimera",
        "EngineVersion": "4.0.0",
        "LicenseModel": "docdb",
        "SnapshotType": "manual",
        "PercentProgress": 0,
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:us-east-1:111122223333:key/sample-key-id",
        "DBClusterSnapshotArn": "arn:aws:rds:us-east-1:111122223333:cluster-
snapshot:sample-cluster-snapshot-copy",
        "SourceDBClusterSnapshotArn": "arn:aws:rds:us-west-2:111122223333:cluster-
snapshot:sample-cluster-snapshot",
    }
}
```

> **Note**
>
> 암호화된 스냅샷 복사에 대한 자세한 내용은 클러스터 스냅샷 암호화 복사 섹션을 참조하세요.
> 저장된 암호화에 대한 자세한 내용은 Amazon DocumentDB 저장 데이터 암호화 섹션을 참조하세요.

Amazon DocumentDB 클러스터 스냅샷 공유

Amazon DocumentDB를 사용하면 다음과 같은 방법으로 수동 클러스터 스냅샷을 공유할 수 있습니다.

- 암호화 여부에 관계없이 수동 클러스터 스냅샷을 공유하면 권한 있는 AWS 계정이 스냅샷을 복사할 수 있습니다.
- 수동 클러스터 스냅샷을 암호화하든 암호화되지 않든 공유하면 권한 있는 AWS 계정이 스냅샷의 복사본을 만들고 복원하는 대신 스냅샷에서 클러스터를 직접 복원할 수 있습니다.

> **Note**
>
> 자동 클러스터 스냅샷을 공유하려면 자동 스냅샷을 복사하여 수동 클러스터 스냅샷을 생성한 다음 해당 복사본을 공유합니다. 이 프로세스는 AWS 백업 생성 리소스에도 적용됩니다.

수동 스냅샷을 최대 20개의 다른 스냅샷과 공유할 수 있습니다 AWS 계정. 암호화되지 않은 수동 스냅샷을 퍼블릭으로도 공유할 수 있습니다. 그러면 모든 계정에서 해당 스냅샷을 사용할 수 있습니다. 스냅샷을 퍼블릭으로 공유할 경우, 어떤 퍼블릭 스냅샷에도 비공개 정보가 포함되지 않도록 유의하세요.

수동 스냅샷을 다른 AWS 계정와 공유하고 AWS CLI 또는 Amazon DocumentDB API를 사용하여 공유 스냅샷에서 클러스터를 복원하는 경우 공유 스냅샷의 Amazon 리소스 이름(ARN)을 스냅샷 식별자로 지정해야 합니다.

암호화된 스냅샷 공유

다음 제한은 암호화된 스냅샷 공유에 적용됩니다.

- 암호화된 스냅샷을 퍼블릭으로는 공유할 수 없습니다.
- 스냅샷을 공유한 계정의 기본 AWS KMS 암호화 키를 사용하여 암호화된 스냅샷은 공유할 수 없습니다.

다음 단계에 따라 암호화된 스냅샷을 공유합니다.

1. 스냅샷을 암호화하는 데 사용된 AWS Key Management Service (AWS KMS) 암호화 키를 스냅샷에 액세스할 수 있는 계정과 공유합니다.

   키 정책에 다른 AWS 계정을 추가하여 다른 계정과 AWS KMS 암호화 AWS KMS 키를 공유할 수 있습니다. 키 정책 업데이트에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 AWS KMS에서 키 정책 사용을 참조하세요. 키 정책 생성의 예는 이번 주제 후반부의 암호화된 스냅샷을 복사할 수 있도록 IAM 정책 생성 섹션을 참조하세요.
2. 아래와 AWS CLI같이를 사용하여 암호화된 스냅샷을 다른 계정과 공유합니다. ???

AWS KMS 암호화 키에 대한 액세스 허용

다른 사용자가 계정에서 공유된 암호화된 스냅샷을 복사 AWS 계정 하려면 스냅샷을 공유하는 계정이 스냅샷을 암호화한 AWS KMS 키에 액세스할 수 있어야 합니다. 키에 대한 다른 계정 액세스를 허용하려면 AWS KMS 키 정책의 보안 주체로 공유하려는 계정의 ARN으로 키의 AWS KMS 키 정책을 AWS KMS 업데이트합니다. 그런 다음 `kms:CreateGrant` 작업을 허용합니다.

AWS KMS 암호화 키에 대한 액세스 권한을 계정에 부여한 후 암호화된 스냅샷을 복사하려면 해당 계정에 아직 AWS Identity and Access Management (IAM) 사용자가 없는 경우 해당 계정을 생성해야 합니다. 또한 해당 계정은 사용자가 AWS KMS 키를 사용하여 암호화된 스냅샷을 복사할 수 있도록 허용하는 IAM 정책을 해당 IAM 사용자에게 연결해야 합니다. 계정은 IAM 사용자여야 하며 AWS KMS 보안 제한으로 인해 루트 AWS 계정 자격 증명일 수 없습니다.

다음 키 정책 예제에서 사용자 123451234512은 AWS KMS 암호화 키의 소유자입니다. 사용자 123456789012는 키를 공유 중인 계정입니다. 이 업데이트된 키 정책은 계정에 AWS KMS 키에 대한 액세스 권한을 부여합니다. 사용자 123456789012의 루트 AWS 계정 자격 증명에 대한 ARN을 정책의 보안 주체로 포함하고 `kms:CreateGrant` 작업을 허용하여이 작업을 수행합니다.

```
{
    "Id": "key-policy-1",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow use of the key",
            "Effect": "Allow",
            "Principal": {"AWS": [
                "arn:aws:iam::123451234512:user/KeyUser",
                "arn:aws:iam::123456789012:root"
            ]},
```

```
            "Action": [
                "kms:CreateGrant",
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey"
            ],
            "Resource": "*"},
            {
            "Sid": "Allow attachment of persistent resources",
            "Effect": "Allow",
            "Principal": {"AWS": [
                "arn:aws:iam::123451234512:user/KeyUser",
                "arn:aws:iam::123456789012:root"
            ]},
            "Action": [
                "kms:CreateGrant",
                "kms:ListGrants",
                "kms:RevokeGrant"
            ],
            "Resource": "*",
            "Condition": {"Bool": {"kms:GrantIsForAWSResource": true}}
        }
    ]
}
```

암호화된 스냅샷을 복사할 수 있도록 IAM 정책 생성

외부 AWS 계정 가 AWS KMS 키에 액세스할 수 있는 경우 해당 계정의 소유자는 계정에 대해 생성된 IAM 사용자가 해당 AWS KMS 키로 암호화된 암호화된 스냅샷을 복사하도록 허용하는 정책을 생성할 수 있습니다.

다음 예제에서는 AWS 계정 123456789012용 IAM 사용자에게 연결할 수 있는 정책을 보여줍니다. 이 정책을 통해 IAM 사용자는 us-west-2 리전의 AWS KMS 키로 암호화된 계정 123451234512c989c1dd-a3f2-4a5d-8d96-e793d082ab26에서 공유 스냅샷을 복사할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "AllowUseOfTheKey",
            "Effect": "Allow",
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey",
                "kms:CreateGrant",
                "kms:RetireGrant"
            ],
            "Resource": ["arn:aws:kms:us-west-2:123451234512:key/c989c1dd-
a3f2-4a5d-8d96-e793d082ab26"]
        },
        {
            "Sid": "AllowAttachmentOfPersistentResources",
            "Effect": "Allow",
            "Action": [
                "kms:CreateGrant",
                "kms:ListGrants",
                "kms:RevokeGrant"
            ],
            "Resource": ["arn:aws:kms:us-west-2:123451234512:key/c989c1dd-
a3f2-4a5d-8d96-e793d082ab26"],
            "Condition": {
                "Bool": {
                    "kms:GrantIsForAWSResource": true
                }
            }
        }
    ]
}
```

키 정책 업데이트에 관한 세부 정보는 AWS Key Management Service 개발자 안내서의 키 정책 AWS KMS을 참조하세요.

스냅샷 공유

AWS Management Console 또는 AWS CLI를 사용하여 Amazon DocumentDB 수동 클러스터 스냅샷(또는 자동 스냅샷의 사본)을 공유할 수 있습니다.

Using the AWS Management Console

를 사용하여 스냅샷을 공유하려면 다음 단계를 AWS Management Console완료하세요.

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 [Snapshots]를 선택합니다.
3. 공유할 수동 스냅샷을 선택합니다.
4. 작업 드롭다운 메뉴에서 공유를 선택합니다.
5. DB 스냅샷 가시성을 위해 다음 옵션 중 하나를 선택합니다.

   - 소스가 암호화되지 않은 경우 모든 AWS 계정이 수동 스냅샷에서 클러스터를 복원하도록 허용하려면 퍼블릭을 선택합니다. 또는 수동 스냅샷에서 클러스터를 복원하도록 지정한 AWS 계정만 허용하려면 프라이빗을 선택합니다.

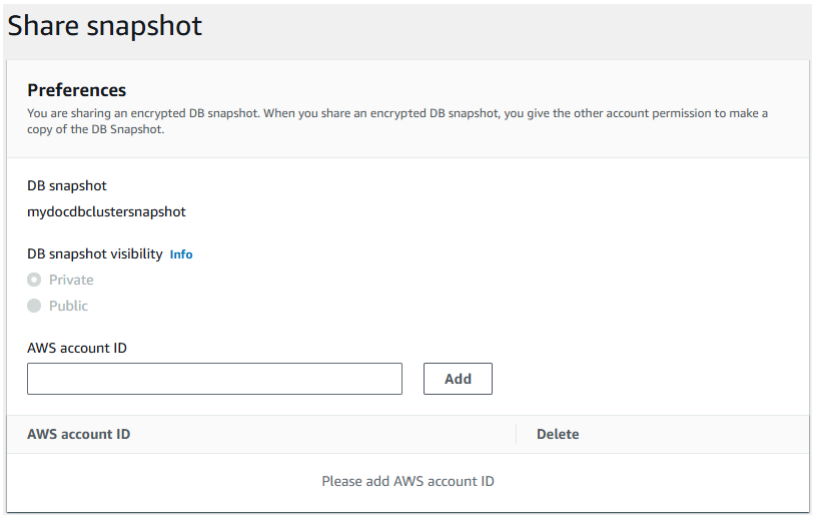
     > **Warning**
     >
     > DB 스냅샷 가시성을 퍼블릭으로 설정하면 모든 AWS 계정이 수동 스냅샷에서 클러스터를 복원하고 데이터에 액세스할 수 있습니다. 프라이빗 정보가 포함된 수동 클러스터 스냅샷을 퍼블릭으로 공유하지 마세요.

   - 원본 DB 클러스터가 암호화되어 있는 경우 암호화된 스냅샷을 퍼블릭으로 공유할 수 없으므로 DB snapshot visibility(DB 스냅샷 가시성)는 Private(프라이빗)으로 설정됩니다.

     > **Note**
     >
     > 기본값으로 암호화된 스냅샷은 공유할 AWS KMS key 수 없습니다.

6. AWS 계정 ID에가 수동 스냅샷에서 클러스터를 복원하도록 허용할 AWS 계정의 계정 식별자를 입력한 다음 추가를 선택합니다. 최대 20개의 AWS 계정까지 추가 계정 식별자를 포함하도록 반복합니다 AWS .

   허용된 AWS 계정 목록에 계정 식별자를 추가할 때 오류가 발생하면 잘못된 AWS 계정 식별자 오른쪽에 있는 삭제를 선택하여 목록에서 삭제할 수 있습니다.



7. 가 수동 스냅샷을 복원하도록 허용하려는 모든 AWS 계정의 식별자를 추가한 후 저장을 선택하여 변경 사항을 저장합니다.

Using the AWS CLI

를 사용하여 스냅샷을 공유하려면 Amazon DocumentDB `modify-db-snapshot-attribute` 작업을 AWS CLI사용합니다. `--values-to-add` 파라미터를 사용하여 수동 스냅샷을 복원할 권한이 AWS 계정 있는의 IDs 목록을 추가합니다.

다음 예제에서는 123451234512 및 1234567890120이라는 두 AWS 계정 식별자가 라는 스냅샷을 복원하도록 허용합니다`manual-snapshot1`. 또한 스냅샷이 프라이빗으로 표시되도록 `all` 속성 값을 지웁니다.

Linux, macOS, Unix의 경우:

```
aws docdb modify-db-cluster-snapshot-attribute \
    --db-cluster-snapshot-identifier sample-cluster-snapshot \
    --attribute-name restore \
    --values-to-add '["123451234512","123456789012"]'
```

Windows의 경우:

```
aws docdb modify-db-cluster-snapshot-attribute ^
    --db-cluster-snapshot-identifier sample-cluster-snapshot ^
    --attribute-name restore ^
    --values-to-add '["123451234512","123456789012"]'
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
{
    "DBClusterSnapshotAttributesResult": {
        "DBClusterSnapshotIdentifier": "sample-cluster-snapshot",
        "DBClusterSnapshotAttributes": [
            {
                "AttributeName": "restore",
                "AttributeValues": [
                    "123451234512",
                    "123456789012"
                ]
            }
        ]
    }
}
```

목록에서 AWS 계정 식별자를 제거하려면 `--values-to-remove` 파라미터를 사용합니다. 다음 예제에서는 AWS 계정 ID 123456789012가 스냅샷을 복원하지 못하도록 합니다.

Linux, macOS, Unix의 경우:

```
aws docdb modify-db-cluster-snapshot-attribute \
    --db-cluster-snapshot-identifier sample-cluster-snapshot \
    --attribute-name restore \
    --values-to-remove '["123456789012"]'
```

Windows의 경우:

```
aws docdb modify-db-cluster-snapshot-attribute ^
    --db-cluster-snapshot-identifier sample-cluster-snapshot ^
    --attribute-name restore ^
    --values-to-remove '["123456789012"]'
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
{
    "DBClusterSnapshotAttributesResult": {
        "DBClusterSnapshotIdentifier": "sample-cluster-snapshot",
        "DBClusterSnapshotAttributes": [
            {
                "AttributeName": "restore",
                "AttributeValues": [
                    "123451234512"
```

```
                ]
            }
        ]
    }
}
```

클러스터 스냅샷에서 복원

Amazon DocumentDB(MongoDB 호환)는 스토리지 볼륨의 클러스터 스냅샷을 만듭니다. 클러스터 스냅샷에서 복원하여 새로운 클러스터를 생성할 수 있습니다. 클러스터를 복원할 때 복원 원본으로 사용할 클러스터 스냅샷의 이름을 입력한 다음, 이 복원 작업에서 생성되는 새 클러스터의 이름을 입력합니다. 복원 시 새 클러스터가 생성되므로 스냅샷에서 기존 클러스터로의 복원은 불가능합니다.

클러스터 스냅샷에서 클러스터를 복원하는 경우:

- 이 작업에서는 클러스터만 복원하고 해당 클러스터에 대한 인스턴스는 복원하지 않습니다. `--db-cluster-identifier`에 복원된 클러스터의 식별자를 지정하여 복원된 클러스터의 인스턴스를 생성하려면 `create-db-instance` 작업을 호출해야 합니다. 클러스터가 사용 가능할 때에만 인스턴스를 생성할 수 있습니다.
- 암호화된 스냅샷은 암호화되지 않은 클러스터로 복원할 수 없습니다. 그러나 AWS KMS 키를 지정하여 암호화되지 않은 스냅샷을 암호화된 클러스터로 복원할 수 있습니다.
- 암호화된 스냅샷에서 클러스터를 복원하려면 AWS KMS 키에 액세스할 수 있어야 합니다.

> **Note**
>
> 3.6 클러스터를 4.0 클러스터로 복원할 수는 없지만 한 클러스터 버전에서 다른 클러스터 버전으로 마이그레이션할 수는 있습니다. 자세한 내용은 Amazon DocumentDB로 마이그레이션 섹션을 참조하세요.

Using the AWS Management Console

다음 절차에서는 Amazon DocumentDB 관리 콘솔을 사용하여 클러스터 스냅샷에서 Amazon DocumentDB 클러스터를 복원하는 방법을 보여줍니다.

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.

2. 탐색 창에서 스냅샷을 선택한 다음 클러스터 복원에 사용하려는 스냅샷 왼쪽에 있는 버튼을 선택합니다.

> **Tip**
>
> 화면 왼쪽에 탐색 창이 표시되지 않으면 페이지 왼쪽 상단 모서리에서 메뉴 아이콘 (≡)을 선택하세요.

3. 작업 메뉴에서 복원을 선택합니다.
4. 스냅샷 복원 페이지에서 구성 섹션을 완료합니다.

   a. 클러스터 식별자 — 새 클러스터의 이름입니다. Amazon DocumentDB 제공 이름을 수락하거나 원하는 이름을 입력할 수 있습니다. Amazon DocumentDBsupplied 제공 이름은 `docdb-` 및 UTC 타임스탬프 형식입니다(예: `docdb-yyyy-mm-dd-hh-mm-ss`).

   b. 인스턴스 클래스 — 새 클러스터의 인스턴스 클래스입니다. 기본 인스턴스 클래스를 수락하거나 드롭다운 목록에서 인스턴스 클래스를 선택할 수 있습니다.

   c. 인스턴스 수 — 이 클러스터를 사용하여 생성할 인스턴스 수입니다. 기본값인 3개의 인스턴스(기본 읽기/쓰기 1개 및 읽기 전용 복제본 2개)를 수락하거나 드롭다운 목록에서 인스턴스 수를 선택할 수 있습니다.

5. 클러스터 스토리지 구성을 위해서는 스토리지 옵션을 선택합니다.

> **Note**
>
> Amazon DocumentDB I/O 최적화 스토리지 구성은 Amazon DocumentDB 5.0 엔진 버전에서만 사용할 수 있습니다.

6. 클러스터 구성이 마음에 들면 클러스터 복원을 선택하고 클러스터가 복원되는 동안 기다립니다.
7. 기본값이 아닌 Amazon VPC 또는 보안 그룹 지정과 같은 일부 구성을 변경하려면 페이지 왼쪽 하단에서 고급 설정 표시를 선택한 후 다음 단계를 계속 진행합니다.

   a. 네트워크 설정 섹션을 완료합니다.

   - 가상 프라이빗 클라우드(VPC) — 현재 VPC를 수락하거나 드롭다운 목록에서 VPC를 선택합니다.

- 서브넷 그룹 — `default` 서브넷 그룹을 수락하거나 드롭다운 목록에서 하나를 선택합니다.
- VPC 보안 그룹 — `default (VPC)` 보안 그룹을 수락하거나 목록에서 하나를 선택합니다.

b. 클러스터 옵션 섹션을 작성합니다.

- 데이터베이스 포트 — 기본 포트인 27017을 수락하거나, 위쪽 화살표 또는 아래쪽 화살표를 사용하여 애플리케이션 연결에 사용하려는 포트를 설정합니다.

c. 암호화 섹션을 작성합니다.

- 저장된 데이터 암호화 — 스냅샷이 암호화되면 이 옵션은 사용할 수 없습니다. 스냅샷이 암호화되지 않으면 다음에서 한 개를 선택할 수 있습니다.
  - 모든 클러스터 데이터를 암호화하려면 Enable encryption-at-rest(유휴상태 암호화 활성)을 선택합니다. 이 옵션을 선택하면 KMS 키를 지정해야 합니다.
  - 클러스터 데이터를 암호화하지 않으려면 Disable encryption-at-rest(유휴상태 암호화 비활성)을 선택합니다. 이 옵션을 선택하면 암호화 섹션이 완료됩니다.
- AWS KMS 키 - 드롭다운 목록에서 다음 중 하나를 선택합니다.
  - (기본값) aws/rds - 계정 번호와 AWS KMS 키 ID가이 옵션 다음에 나열됩니다.
  - 고객 관리형 키 -이 옵션은 AWS Identity and Access Management (IAM) 콘솔에서 IAM 암호화 키를 생성한 경우에만 사용할 수 있습니다. 이 키를 선택하여 클러스터를 암호화할 수 있습니다.
  - 키 ARN 입력 - ARN 상자에 AWS KMS 키의 Amazon 리소스 이름(ARN)을 입력합니다. ARN 형식은 `arn:aws:kms:<region>:<accountID>:key/<key-id>`입니다.

d. 로그 내보내기 섹션을 완료합니다.

- CloudWatch에 게시할 로그 유형 선택 — 다음 중 하나를 선택합니다.
  - 활성화됨 — 클러스터가 DDL 로깅을 Amazon CloudWatch Logs로 내보낼 수 있도록 합니다.
  - 비활성화됨 — 클러스터가 DDL 로그를 Amazon CloudWatch Logs로 내보내는 것을 방지합니다. 비활성이 기본값입니다.
- IAM 역할 — 목록에서 RDS 서비스 연결 역할을 선택합니다.

e. 태그 섹션을 완료합니다.

- 태그 추가 — 키 상자에 클러스터의 태그 이름을 입력합니다. 값 상자에 태그 값을 입력합니다(선택 사항). 태그는 AWS Identity and Access Management (IAM) 정책과 함께

사용하여 Amazon DocumentDB 리소스에 대한 액세스를 관리하고 리소스에 적용할 수 있는 작업을 제어합니다.

f. 삭제 방지 섹션을 완료합니다.

- 삭제 방지 활성화 — 클러스터가 실수로 삭제되지 않도록 보호합니다. 이 옵션이 활성화되면 클러스터를 삭제할 수 없습니다.

8. 클러스터 복원을 선택합니다.

Using the AWS CLI

를 사용하여 스냅샷에서 클러스터를 복원하려면 다음 파라미터와 함께 `restore-db-cluster-from-snapshot` 작업을 AWS CLI사용합니다. 자세한 내용은 RestoreDBClusterFromSnapshot 단원을 참조하십시오.

- **`--db-cluster-identifier`** - 필수입니다. 작업에서 생성되는 클러스터 이름 이 이름으로 된 클러스터는 이 작업 이전에 존재할 수 없습니다.

  클러스터 명명 제약 조건:

  - 길이는 [1-63]개의 글자, 숫자 또는 하이픈입니다.
  - 첫 번째 문자는 글자이어야 합니다.
  - 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다.
  - 리전별로 Amazon RDS, Neptune 및 Amazon DocumentDB의 모든 클러스터 AWS 계정에 대해 고유해야 합니다.
- **`--snapshot-identifier`** — 필수입니다. 복원하는 데 사용되는 스냅샷의 이름입니다. 이 이름으로 된 스냅샷이 있어야 하며 사용 가능한 상태여야 합니다.
- **`--engine`** — 필수입니다. `docdb`여야 합니다.
- **`--storage-type standard | iopt1`** – 선택 사항. 기본값: `standard`.
- **`--kms-key-id`** - 선택 사항입니다. 암호화된 스냅샷을 복원하거나 암호화되지 않은 스냅샷에서 복원할 때 클러스터를 암호화할 때 사용할 AWS KMS 키 식별자의 ARN입니다. 키 ID를 AWS KMS 제공하면 스냅샷이 암호화되었는지 여부에 관계없이 복원된 클러스터가 AWS KMS 키로 암호화됩니다.

  `--kms-key-id` 형식은 `arn:aws:kms:<region>:<accountID>:key/<key-id>`입니다. `--kms-key-id` 파라미터 값을 지정하지 않으면 다음과 같이 진행됩니다.

  - 의 스냅샷`--snapshot-identifier`이 암호화된 경우 복원된 클러스터는 스냅샷을 암호화하는 데 사용된 것과 동일한 AWS KMS 키를 사용하여 암호화됩니다.

- `--snapshot-identifier` 에 스냅샷이 암호화되지 않으면 복원된 클러스터는 암호화되지 않습니다.

Linux, macOS, Unix의 경우:

```
aws docdb restore-db-cluster-from-snapshot \
    --db-cluster-identifier sample-cluster-restore \
    --snapshot-identifier sample-cluster-snapshot \
    --engine docdb \
    --kms-key-id arn:aws:kms:us-east-1:123456789012:key/SAMPLE-KMS-KEY-ID
```

Windows의 경우:

```
aws docdb restore-db-cluster-from-snapshot ^
    --db-cluster-identifier sample-cluster-restore ^
    --snapshot-identifier sample-cluster-snapshot ^
    --engine docdb ^
    --kms-key-id arn:aws:kms:us-east-1:123456789012:key/SAMPLE-KMS-KEY-ID
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
{
    "DBCluster": {
        "AvailabilityZones": [
            "us-east-1c",
            "us-east-1b",
            "us-east-1a"
        ],
        "BackupRetentionPeriod": 1,
        "DBClusterIdentifier": "sample-cluster-restore",
        "DBClusterParameterGroup": "default.docdb4.0",
        "DBSubnetGroup": "default",
        "Status": "creating",
        "Endpoint": "sample-cluster-restore.cluster-node.us-
east-1.docdb.amazonaws.com",
        "ReaderEndpoint": "sample-cluster-restore.cluster-node.us-
east-1.docdb.amazonaws.com",
        "MultiAZ": false,
        "Engine": "docdb",
        "EngineVersion": "4.0.0",
        "Port": 27017,
```

```
        "MasterUsername": "<master-user>",
        "PreferredBackupWindow": "02:00-02:30",
        "PreferredMaintenanceWindow": "tue:09:50-tue:10:20",
        "DBClusterMembers": [],
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-abcdefgh",
                "Status": "active"
            }
        ],
        "HostedZoneId": "ABCDEFGHIJKLM",
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:us-east-1:<accountID>:key/<sample-key-id>",
        "DbClusterResourceId": "cluster-ABCDEFGHIJKLMNOPQRSTUVWXYZ",
        "DBClusterArn": "arn:aws:rds:us-east-1:<accountID>:cluster:sample-cluster-
restore",
        "AssociatedRoles": [],
        "ClusterCreateTime": "2020-04-01T01:43:40.871Z",
        "DeletionProtection": true
    }
}
```

클러스터 상태가 사용 가능으로 전환되면, 클러스터에 적용할 인스턴스를 최소 한 개 생성합니다.

Linux, macOS, Unix의 경우:

```
aws docdb create-db-instance \
    --db-cluster-identifier sample-cluster-restore  \
    --db-instance-identifier sample-cluster-restore-instance \
    --availability-zone us-east-1b \
    --promotion-tier 2 \
    --db-instance-class db.r5.large \
    --engine docdb
```

Windows의 경우:

```
aws docdb create-db-instance ^
    --db-cluster-identifier sample-cluster-restore  ^
    --db-instance-identifier sample-cluster-restore-instance ^
    --availability-zone us-east-1b ^
    --promotion-tier 2 ^
    --db-instance-class db.r5.large ^
    --engine docdb
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
{
    "DBInstance": {
        "DBInstanceIdentifier": "sample-cluster-restore-instance",
        "DBInstanceClass": "db.r5.large",
        "Engine": "docdb",
        "DBInstanceStatus": "creating",
        "PreferredBackupWindow": "02:00-02:30",
        "BackupRetentionPeriod": 1,
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-abcdefgh",
                "Status": "active"
            }
        ],
        "AvailabilityZone": "us-west-2b",
        "DBSubnetGroup": {
            "DBSubnetGroupName": "default",
            "DBSubnetGroupDescription": "default",
            "VpcId": "vpc-6242c31a",
            "SubnetGroupStatus": "Complete",
            "Subnets": [
                {
                    "SubnetIdentifier": "subnet-abcdefgh",
                    "SubnetAvailabilityZone": {
                        "Name": "us-west-2a"
                    },
                    "SubnetStatus": "Active"
                },
                {
                    ...
                }
            ]
        },
        "PreferredMaintenanceWindow": "fri:09:43-fri:10:13",
        "PendingModifiedValues": {},
        "EngineVersion": "4.0.0",
        "AutoMinorVersionUpgrade": true,
        "PubliclyAccessible": false,
        "DBClusterIdentifier": "sample-cluster-restore",
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:us-east-1:<accountID>:key/<sample-key-id>",
        "DbiResourceId": "db-ABCDEFGHIJKLMNOPQRSTUVWXYZ",
```

```
        "CACertificateIdentifier": "rds-ca-2019",
        "PromotionTier": 2,
        "DBInstanceArn": "arn:aws:rds:us-east-1:<accountID>:db:sample-cluster-
restore-instance"
    }
}
```

특정 시점으로 복원

AWS Management Console 또는 AWS Command Line Interface ()를 사용하여 클러스터의 백업 보존 기간 내에 있는 시점으로 클러스터를 복원할 수 있습니다AWS CLI.

> **Note**
>
> 3.6 클러스터를 4.0 클러스터로 특정 시점으로 복원할 수는 없지만 한 클러스터 버전에서 다른 클러스터 버전으로 마이그레이션할 수는 있습니다. 자세한 내용은 Amazon DocumentDB로 마이그레이션 섹션을 참조하세요.

클러스터를 특정 시점으로 복원할 경우 다음에 유의하세요.

- 새 클러스터는 기본 파라미터 그룹으로 생성된다는 점만 제외하고, 소스 클러스터와 동일한 구성으로 생성됩니다. 새 클러스터의 파라미터 그룹을 원본 클러스터의 파라미터 그룹으로 설정하려면 클러스터가 사용 가능한 상태로 된 이후에 클러스터를 수정합니다. 클러스터를 수정하는 방법에 대한 자세한 정보는 Amazon DocumentDB 클러스터 수정을 참조하세요.

Using the AWS Management Console

AWS Management Console에서 다음을 완료하여 보존 기간 중 특정 시점으로 클러스터를 복원할 수 있습니다.

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 클러스터를 선택합니다. 클러스터 목록에서 복원할 클러스터 왼쪽에 있는 버튼을 선택합니다.

> **Tip**
>
> 화면 왼쪽에 탐색 창이 표시되지 않으면 페이지 왼쪽 상단 모서리에서 메뉴 아이콘 (≡) 을 선택하세요.

3. 작업 메뉴에서 특정 시점으로 복구를 선택합니다.
4. 복원 시간 섹션을 완료합니다. 여기서는 복원할 날짜 및 시간을 지정합니다.

   a. 복원 날짜 — 가장 빠른 복원 시간과 최근 복원 시간 사이의 날짜를 선택하거나 입력합니다.

   b. 복원 시간 — 가장 빠른 복원 시간과 최근 복원 시간 사이의 시간, 분, 초를 선택하거나 입력합니다.
5. 구성 섹션을 완료합니다.

   a. 클러스터 식별자 — 기본 식별자를 수락하거나 원하는 식별자를 입력합니다.

   클러스터 명명 제약 조건:

   - 길이는 [1-63]개의 글자, 숫자 또는 하이픈입니다.
   - 첫 번째 문자는 글자이어야 합니다.
   - 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다.
   - 리전별로 Amazon RDS, Neptune 및 Amazon DocumentDB의 모든 클러스터 AWS 계정에 대해 고유해야 합니다.

   b. 인스턴스 클래스 — 목록에서 클러스터 인스턴스에 사용할 인스턴스 클래스를 선택합니다.

   c. 인스턴스 수 — 드롭다운 목록에서 클러스터 복원 시 생성할 인스턴스 수를 선택합니다.
6. 클러스터 스토리지 구성을 위해서는 스토리지 옵션을 선택합니다.

> **Note**
>
> Amazon DocumentDB I/O 최적화 스토리지 구성은 Amazon DocumentDB 5.0 엔진 버전에서만 사용할 수 있습니다.

7. 선택 사항. 네트워크 설정 및 클러스터 옵션을 구성하고 로그 내보내기를 활성화하려면 고급 설정 표시를 선택하고 다음 섹션을 완료합니다. 그렇지 않으면 다음 단계로 계속 진행합니다.

- 네트워크 설정
  1. 가상 프라이빗 클라우드(VPC)) — 목록에서 이 클러스터에 사용할 VPC를 선택합니다.
  2. 서브넷 그룹 — 목록에서 이 클러스터에 사용할 서브넷 그룹을 선택합니다.
  3. VPC 보안 그룹 — 목록에서 이 클러스터에 사용할 VPC 보안 그룹을 선택합니다.

- 클러스터 옵션
  1. 포트 — 기본 포트(27017)를 수락하거나 위쪽 화살표 또는 아래쪽 화살표를 사용하여 이 클러스터와 통신할 포트를 설정합니다.

- 로그 내보내기
  1. 감사 로그 — 감사 로그를 Amazon CloudWatch Logs로 내보내기를 활성화하려면 이 옵션을 선택합니다. 이 옵션을 선택하는 경우, 클러스터의 사용자 지정 파라미터 그룹에서 `audit_logs`를 활성화해야 합니다. 자세한 내용은 Amazon DocumentDB 이벤트 감사 단원을 참조하십시오.
  2. 프로파일러 로그 — 작업 프로파일러 Amazon CloudWatch Logs를 로 내보내려면 이 옵션을 선택합니다. 이 옵션을 선택하는 경우, 클러스터의 사용자 지정 파라미터 그룹에서 다음 파라미터도 수정해야 합니다.
     - `profiler` — `enabled`으로 설정합니다.
     - `profiler_threshold_ms` — 작업 프로파일링에 대한 임계값을 설정하려면 값 `[0-INT_MAX]`로 설정합니다.
     - `profiler_sampling_rate` — 프로파일링할 느린 작업 비율을 설정하려면 값 `[0.0-1.0]`으로 설정합니다.

     자세한 내용은 Amazon DocumentDB 작업 프로파일링 단원을 참조하십시오.
  3. 프로파일러 로그 — 프로파일러 로그를 Amazon CloudWatch로 내보내기
  4. IAM 역할 — 목록에서 RDS 서비스 연결 역할을 선택합니다.
- 태그
  1. 태그 추가 — 키 상자에 클러스터의 태그 이름을 입력합니다. 값 상자에 태그 값을 입력합니다(선택 사항). AWS Identity and Access Management (IAM) 정책과 함께 이러한 태그를 사용하여 Amazon DocumentDB 리소스에 대한 액세스를 관리하고 해당 리소스에 적용 가능한 작업을 제어할 수 있습니다.

삭제 방지

1. 삭제 방지 활성화 — 클러스터가 실수로 삭제되지 않도록 보호합니다. 이 옵션이 활성화되면 클러스터를 삭제할 수 없습니다.

8. 클러스터를 복원하려면 클러스터 생성을 선택합니다. 또는 취소를 선택하여 작업을 취소할 수 있습니다.

Using the AWS CLI

스냅샷의 백업 보존 기간을 사용하여 클러스터를 특정 시점으로 복원하려면 `restore-db-cluster-to-point-in-time` 작업을 다음 파라미터와 함께 사용합니다.

- **`--db-cluster-identifier`** - 필수입니다. 생성할 새 클러스터의 이름입니다. 이 클러스터는 작업 이전에 존재할 수 없습니다. 파라미터 값은 다음 제약 조건을 충족해야 합니다.

  클러스터 명명 제약 조건:

  - 길이는 [1-63]개의 글자, 숫자 또는 하이픈입니다.
  - 첫 번째 문자는 글자이어야 합니다.
  - 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다.
  - 리전별로 Amazon RDS, Neptune 및 Amazon DocumentDB의 모든 클러스터 AWS 계정에 대해 고유해야 합니다.

- **`--restore-to-time`** — 클러스터를 복원할 UTC 날짜 및 시간입니다. 예를 들어 `2018-06-07T23:45:00Z`입니다.

  시간 제약:

  - 클러스터에 대해 복원 가능한 최신 시간보다 이전이어야 합니다.
  - `--use-latest-restorable-time` 파라미터를 제공하지 않은 경우에 지정해야 합니다.
  - `--use-latest-restorable-time` 파라미터가 `true`인 경우에는 지정할 수 없습니다.
  - `--restore-type` 파라미터 값이 `copy-on-write`인 경우에는 지정할 수 없습니다.

- **`--source-db-cluster-identifier`** — 복원할 소스 클러스터의 이름입니다. 이 클러스터가 있고 사용 가능해야 합니다.

- **`--use-latest-restorable-time`** 또는 **`--no-use-latest-restorable-time`** — 복원 가능한 최신 백업 시간으로 복원할지 여부를 나타냅니다. `--restore-to-time` 파라미터를 제공한 경우에는 지정할 수 없습니다.

- **`--storage-type standard | iopt1`** – 선택 사항. 기본값: `standard`.

AWS CLI 작업은 해당 클러스터의 인스턴스가 아닌 클러스터restore-db-cluster-to-point-in-time만 복원합니다. --db-cluster-identifier에 복원된 클러스터의 식별자를 지정하여 복원된 클러스터의 인스턴스를 생성하려면 create-db-instance 작업을 간접적으로 호출해야 합니다. restore-db-cluster-to-point-in-time 작업이 완료되고 복원된 클러스터를 사용 가능할 경우에만 인스턴스를 생성할 수 있습니다.

Example

다음 예에서는 sample-cluster-snapshot 스냅샷에서 복원 가능한 최신 시간으로 sample-cluster-restored를 생성합니다.

Linux, macOS, Unix의 경우:

```
aws docdb restore-db-cluster-to-point-in-time \
    --db-cluster-identifier sample-cluster-restored \
    --source-db-cluster-identifier sample-cluster-snapshot \
    --use-latest-restorable-time
```

Windows의 경우:

```
aws docdb restore-db-cluster-to-point-in-time ^
    --db-cluster-identifier sample-cluster-restored ^
    --source-db-cluster-identifier sample-cluster-snapshot ^
    --use-latest-restorable-time
```

Example

다음 예에서는 sample-cluster-snapshot 스냅샷에서 sample-cluster 백업 보존 기간 내에 있는 2018년 12월 11일 03:15(UTC)로 sample-cluster-restored를 생성합니다.

Linux, macOS, Unix의 경우:

```
aws docdb restore-db-cluster-to-point-in-time \
    --db-cluster-identifier sample-cluster-restore \
    --source-db-cluster-identifier sample-cluster \
    --restore-to-time 2020-05-12T03:15:00Z
```

Windows의 경우:

```
aws docdb restore-db-cluster-to-point-in-time ^
```

```
    --db-cluster-identifier sample-cluster-restore ^
    --source-db-cluster-identifier sample-cluster ^
    --restore-to-time 2020-05-12T03:15:00Z
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
{
    "DBCluster": {
        "AvailabilityZones": [
            "us-east-1c",
            "us-west-2b",
            "us-west-2a"
        ],
        "BackupRetentionPeriod": 1,
        "DBClusterIdentifier": "sample-cluster-restored",
        "DBClusterParameterGroup": "sample-parameter-group",
        "DBSubnetGroup": "default",
        "Status": "creating",
        "Endpoint": "sample-cluster-restored.node.us-east-1.docdb.amazonaws.com",
        "ReaderEndpoint": "sample-cluster-restored.node.us-
east-1.docdb.amazonaws.com",
        "MultiAZ": false,
        "Engine": "docdb",
        "EngineVersion": "4.0.0",
        "Port": 27017,
        "MasterUsername": "master-user",
        "PreferredBackupWindow": "02:00-02:30",
        "PreferredMaintenanceWindow": "tue:09:50-tue:10:20",
        "DBClusterMembers": [],
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-abc0123",
                "Status": "active"
            }
        ],
        "HostedZoneId": "ABCDEFGHIJKLM",
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:us-east-1:<accountID^>:key/sample-key",
        "DbClusterResourceId": "cluster-ABCDEFGHIJKLMNOPQRSTUVWXYZ",
        "DBClusterArn": "arn:aws:rds:us-east-1:<accountID>:cluster:sample-cluster-
restored",
        "AssociatedRoles": [],
        "ClusterCreateTime": "2020-04-24T20:14:36.713Z",
```

```
        "DeletionProtection": false
    }
}
```

클러스터 스냅샷 삭제

수동 스냅샷은 AWS Management Console 또는를 사용하여 수동으로 삭제할 때만 삭제되는 전체 백업입니다 AWS CLI. 자동 스냅샷은 스냅샷의 보존 기간이 만료되거나 스냅샷의 클러스터를 삭제하는 경우에만 삭제되므로 수동으로 삭제할 수 없습니다.

Using the AWS Management Console

를 사용하여 수동 클러스터 스냅샷을 삭제하려면 다음 단계를 AWS Management Console완료합니다.

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 스냅샷를 선택합니다.

   > **Tip**
   >
   > 화면 왼쪽에 탐색 창이 표시되지 않으면 페이지 왼쪽 상단 모서리에서 메뉴 아이콘 (≡)을 선택하세요.

3. 스냅샷 목록에서 삭제할 스냅샷 왼쪽에 있는 버튼을 선택합니다. 스냅샷은 수동 유형이어야 합니다.

   1. 유형 열 또는 아래에서 `manual` 또는`automatic`로 나열되어 있는지 확인하여 스냅샷 유형이 수동인지 확인할 수 있습니다.
4. 작업 메뉴에서 삭제를 선택합니다. 삭제 옵션을 사용할 수 없는 경우 자동 스냅샷을 선택했을 수 있습니다.
5. 삭제 확인 화면에서 스냅샷을 삭제하려면 삭제를 선택합니다. 스냅샷을 유지하려면 취소를 선택합니다.

Using the AWS CLI

Amazon DocumentDB 수동 클러스터 스냅샷은 AWS CLI를 사용하여 수동으로 삭제할 수 있는 전체 백업입니다. 자동 스냅샷을 수동으로 삭제할 수 없습니다.

를 사용하여 수동 클러스터 스냅샷을 삭제하려면 다음 파라미터와 함께 `delete-db-cluster-snapshot` 작업을 AWS CLI사용합니다.

파라미터

- **`--db-cluster-snapshot-identifier`** — 필수입니다. 삭제할 수동 스냅샷의 이름입니다.

다음 예에서는 클러스터 스냅샷 `sample-cluster-snapshot`을 삭제합니다.

Linux, macOS, Unix의 경우:

```
aws docdb delete-db-cluster-snapshot \
    --db-cluster-snapshot-identifier sample-cluster-snapshot
```

Windows의 경우:

```
aws docdb delete-db-cluster-snapshot ^
    --db-cluster-snapshot-identifier sample-cluster-snapshot
```

이 작업의 출력에는 삭제한 클러스터 스냅샷의 세부 정보가 나열됩니다.

Amazon DocumentDB 리소스 관리

이 단원에서는 Amazon DocumentDB(MongoDB 호환) 구현을 관리하기 위한 여러 가지 구성 요소와 관련 작업을 설명합니다.

주제

- Amazon DocumentDB 운영 작업 개요
- Amazon DocumentDB 글로벌 클러스터의 개요
- Amazon DocumentDB 클러스터 관리
- Amazon DocumentDB 인스턴스 관리
- Amazon DocumentDB 서브넷 그룹 관리
- Amazon DocumentDB 고가용성 및 복제
- Amazon DocumentDB 인덱스 관리
- 컬렉션 수준 문서 압축 관리
- Amazon DocumentDB 이벤트 관리
- 리전 및 가용 영역 선택
- Amazon DocumentDB 클러스터 파라미터 그룹 관리
- Amazon DocumentDB 엔드포인트에 대한 이해
- Amazon DocumentDB Amazon 리소스 이름(ARN) 이해
- Amazon DocumentDB 리소스 태그 지정
- Amazon DocumentDB 유지 관리
- 서비스 연결 역할 이해
- Amazon DocumentDB에서 변경 스트림 사용
- 변경 스트림과 AWS Lambda 함께 사용

Amazon DocumentDB 운영 작업 개요

이 섹션에서는 Amazon DocumentDB 클러스터의 운영 작업 개요 및 AWS CLI를 사용하여 이러한 작업을 수행하는 방법을 소개합니다.

주제

- Amazon DocumentDB 클러스터에 복제본 추가

- 클러스터 및 인스턴스 설명
- 클러스터 스냅샷 생성
- 스냅샷에서 복원
- 클러스터에서 인스턴스 제거
- 클러스터 삭제

Amazon DocumentDB 클러스터에 복제본 추가

Amazon DocumentDB 클러스터에 대한 기본 인스턴스를 만든 후에 1개 이상의 복제본을 추가할 수 있습니다. 복제본은 두 가지 용도로 사용되는 읽기 전용 인스턴스입니다.

- 확장성 — 동시 액세스가 필요한 클라이언트가 많은 경우 읽기 규모 조정을 위해 복제본을 더 추가할 수 있습니다.
- 고가용성 — 기본 인스턴스에 장애가 발생하면 Amazon DocumentDB에서는 복제본 인스턴스로 자동으로 장애 조치하고 이를 새로운 기본 인스턴스로 지정합니다. 복제본이 실패하면 실패한 노드가 복구될 때까지 클러스터의 다른 인스턴스가 계속 요청을 수행할 수 있습니다.

각 Amazon DocumentDB 클러스터에서 최대 15개의 복제본을 지원할 수 있습니다.

> **Note**
>
> 내결함성을 최대화하려면 별도의 가용 영역에 복제본을 배포해야 합니다. 이렇게 하면 전체 가용 영역을 사용할 수 없게 되는 경우에도 Amazon DocumentDB 클러스터가 계속 작동할 수 있습니다.

다음 AWS CLI 예제에서는 새 복제본을 추가하는 방법을 보여줍니다. `--availability-zone` 파라미터는 복제본을 지정된 가용 영역에 배치합니다.

```
aws docdb create-db-instance \
    --db-instance-identifier sample-instance \
    --db-cluster-identifier sample-cluster \
    --engine docdb \
    --db-instance-class db.r5.large \
    --availability-zone us-east-1a
```

클러스터 및 인스턴스 설명

다음 AWS CLI 예제에서는 리전의 모든 Amazon DocumentDB 클러스터를 나열합니다. 클러스터 및 인스턴스 수명주기 관리와 같은 특정 관리 기능의 경우 Amazon DocumentDB는 Amazon RDS와 공유하는 운영 기술을 활용합니다. `filterName=engine,Values=docdb` 필터 파라미터는 Amazon DocumentDB 클러스터만 반환합니다.

클러스터 설명 및 수정에 대한 자세한 내용은 Amazon DocumentDB 클러스터 수명 주기 섹션을 참조하십시오.

```
aws docdb describe-db-clusters --filter Name=engine,Values=docdb
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
{
    "DBClusters": [
        {
            "AvailabilityZones": [
                "us-east-1c",
                "us-east-1b",
                "us-east-1a"
            ],
            "BackupRetentionPeriod": 1,
            "DBClusterIdentifier": "sample-cluster-1",
            "DBClusterParameterGroup": "sample-parameter-group",
            "DBSubnetGroup": "default",
            "Status": "available",
            ...
        },
        {
            "AvailabilityZones": [
                "us-east-1c",
                "us-east-1b",
                "us-east-1a"
            ],
            "BackupRetentionPeriod": 1,
            "DBClusterIdentifier": "sample-cluster-2",
            "DBClusterParameterGroup": "sample-parameter-group",
            "DBSubnetGroup": "default",
            "Status": "available",
            ...
        },
```

```
        {
            "AvailabilityZones": [
                "us-east-1c",
                "us-east-1b",
                "us-east-1a"
            ],
            "BackupRetentionPeriod": 1,
            "DBClusterIdentifier": "sample-cluster-3",
            "DBClusterParameterGroup": "sample-parameter-group",
            "DBSubnetGroup": "default",
            "Status": "available",
            ...
        }
    ]
}
```

다음 AWS CLI 예제에서는 Amazon DocumentDB 클러스터의 인스턴스를 나열합니다. 클러스터 설명 및 수정에 대한 자세한 내용은 Amazon DocumentDB 인스턴스 수명 주기 섹션을 참조하십시오.

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].[DBClusterMembers]'
```

출력은 아래와 같습니다. 이 출력에는 두 개의 인스턴스가 있습니다. 기본 인스턴스는 `sample-instance-1`(`"IsClusterWriter": true`)입니다. 또한 복제본 인스턴스 `sample-instance2`(`"IsClusterWriter: false"`)도 있습니다.

```
[
    [
        [
            {
                "DBInstanceIdentifier": "sample-instance-1",
                "IsClusterWriter": true,
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1
            },
            {
                "DBInstanceIdentifier": "sample-cluster-2",
                "IsClusterWriter": false,
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1
            }
```

```
        ]
    ]
]
```

클러스터 스냅샷 생성

클러스터 스냅샷은Amazon DocumentDB 클러스터의 데이터를 완벽하게 백업합니다. 스냅샷을 만들 때 Amazon DocumentDB는 클러스터 볼륨에서 직접 데이터를 읽습니다. 따라서 클러스터에서 해당 인스턴스가 실행 중이 아니어도 스냅샷을 생성할 수 있습니다. 클러스터 스냅샷을 생성하는 데 걸리는 시간은 클러스터 볼륨의 크기에 따라 다릅니다.

Amazon DocumentDB는 자동 백업을 지원합니다. 자동 백업은 선호하는 백업 윈도우(하루 중 30분) 동안 매일 수행됩니다. 다음 AWS CLI 예제에서는 클러스터의 백업 기간을 보는 방법을 보여줍니다.

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].PreferredBackupWindow'
```

출력에는 백업 창(UTC)이 표시됩니다.

```
[
    "00:18-00:48"
]
```

Amazon DocumentDB 클러스터를 생성할 때 백업 기간을 정의할 수 있습니다. 다음 예제와 같이 백업 기간을 변경할 수도 있습니다. 백업 기간을 정의하지 않으면 Amazon DocumentDB에서 자동으로 클러스터에 할당합니다.

```
aws docdb modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --preferred-backup-window "02:00-02:30"
```

자동 백업 외에도 언제든지 수동으로 클러스터 스냅샷을 만들 수 있습니다. 이때 백업할 클러스터를 지정한 다음 나중에 복원할 수 있도록 스냅샷에 대해 고유한 이름을 지정합니다.

다음 AWS CLI 예제에서는 데이터의 스냅샷을 생성하는 방법을 보여줍니다.

```
aws docdb create-db-cluster-snapshot \
    --db-cluster-identifier sample-cluster \
```

```
    --db-cluster-snapshot-identifier sample-cluster-snapshot
```

스냅샷에서 복원

클러스터 스냅샷을 새 Amazon DocumentDB 클러스터로 복원할 수 있습니다. 이렇게 하려면 스냅샷의 이름과 새 클러스터의 이름을 제공합니다. 스냅샷에서 기존 클러스터로 복원할 수 없습니다. 대신 복원할 때 Amazon DocumentDB에서 새 클러스터를 만든 다음 스냅샷 데이터로 채웁니다.

다음 예제에서는 `sample-cluster` 클러스터의 모든 스냅샷을 보여 줍니다.

```
aws docdb describe-db-cluster-snapshots \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusterSnapshots[*].[DBClusterSnapshotIdentifier,SnapshotType,Status]'
```

출력은 다음과 같습니다. 수동 스냅샷은 수동으로 생성한 스냅샷이지만 자동 스냅샷은 클러스터 백업 기간 중 Amazon DocumentDB에 의해 생성됩니다.

```
[
        "sample-cluster-snapshot",
        "manual",
        "available"
    ],
    [
        "rds:sample-cluster",
        "automated",
        "available"
    ]
]
```

다음 예제에서는 스냅샷에서 Amazon DocumentDB 클러스터를 복원하는 방법을 보여 줍니다.

```
aws docdb restore-db-cluster-from-snapshot \
    --engine docdb \
    --db-cluster-identifier new-sample-cluster \
    --snapshot-identifier sample-cluster-snapshot
```

새 클러스터에 연결된 인스턴스가 없으므로 클러스터와 상호 작용하려면 인스턴스를 클러스터에 추가해야 합니다.

```
aws docdb create-db-instance \
```

```
    --db-instance-identifier new-sample-instance \
    --db-instance-class db.r5.large \
    --engine docdb \
    --db-cluster-identifier new-sample-cluster
```

다음 AWS CLI 작업을 사용하여 클러스터 및 인스턴스 생성 진행 상황을 모니터링할 수 있습니다. 클러스터 및 인스턴스 상태를 사용할 수 있으면 새 클러스터의 엔드포인트에 연결하여 데이터에 액세스할 수 있습니다.

```
aws docdb describe-db-clusters \
    --db-cluster-identifier new-sample-cluster  \
    --query 'DBClusters[*].[Status,Endpoint]'
```

```
aws docdb describe-db-instances \
    --db-instance-identifier new-sample-instance \
    --query 'DBInstances[*].[DBInstanceStatus]'
```

클러스터에서 인스턴스 제거

Amazon DocumentDB에서는 모든 데이터를 클러스터 볼륨에 저장합니다. 모든 인스턴스를 클러스터에서 제거하는 경우에도 데이터가 클러스터 볼륨에 유지됩니다. 데이터에 다시 액세스해야 하는 경우 언제든지 클러스터에 인스턴스를 추가하고 중단한 지점을 선택할 수 있습니다.

다음 예제에서는 Amazon DocumentDB 클러스터에서 인스턴스를 제거하는 방법을 보여줍니다.

```
aws docdb delete-db-instance \
    --db-instance-identifier sample-instance
```

클러스터 삭제

Amazon DocumentDB 클러스터를 삭제하려면 먼저 해당 인스턴스를 모두 제거해야 합니다. 다음 AWS CLI 예제에서는 클러스터의 인스턴스에 대한 정보를 반환합니다. 이 작업으로 인스턴스 식별자가 반환되면 각 인스턴스를 삭제해야 합니다. 자세한 내용은 클러스터에서 인스턴스 제거 단원을 참조하십시오.

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].DBClusterMembers[*].DBInstanceIdentifier'
```

더 이상 인스턴스가 남아 있지 않으면 클러스터를 삭제할 수 있습니다. 이때 다음 옵션 중 하나를 선택해야 합니다.

- 최종 스냅샷 생성 — 나중에 해당 데이터로 새 인스턴스를 다시 만들 수 있도록 스냅샷의 모든 클러스터 데이터를 캡처합니다. 다음 예에서는 이 작업을 수행하는 방법을 보여줍니다.

```
aws docdb delete-db-cluster \
    --db-cluster-identifier sample-cluster \
    --final-db-snapshot-identifier sample-cluster-snapshot
```

- 최종 스냅샷 건너뛰기 — 모든 클러스터 데이터를 영구적으로 삭제합니다. 이 작업은 되돌릴 수 없습니다. 다음 예에서는 이 작업을 수행하는 방법을 보여줍니다.

```
aws docdb delete-db-cluster \
    --db-cluster-identifier sample-cluster \
    --skip-final-snapshot
```

Amazon DocumentDB 글로벌 클러스터의 개요

글로벌 클러스터란 무엇입니까?

글로벌 클러스터는 1개의 기본 영역과 최대 5개의 읽기 전용 보조 영역으로 구성됩니다. 쓰기 작업을 기본 영역의 기본 클러스터에 직접 발급하고 Amazon DocumentDB는 전용 인프라를 사용하여 자동으로 보조 영역에 데이터를 복제합니다. 지연 시간은 일반적으로 1초 미만입니다.

글로벌 클러스터는 어떻게 유용할까요?

- 리전 범위 운영 중단에서 복구 - 리전 전체의 운영 중단 시 보조 클러스터 중 하나를 몇 분 이내에 기본 클러스터로 승격할 수 있으며, 일반적인 Recovery Time Objective(RTO)는 1분 이내입니다. Recovery Point Objective(RPO)는 일반적으로 초 단위로 측정되지만, 장애 발생 시 네트워크를 통한 스토리지 복제 지연에 따라 달라집니다.
- 로컬 대기 시간을 가진 글로벌 읽기 - 전 세계에 사무실이 있는 경우 글로벌 클러스터를 사용하여 주요 정보 소스를 기본 영역에서 업데이트할 수 있습니다. 다른 리전에 있는 사무실은 로컬 대기 시간을 사용하여 해당 리전의 정보에 액세스할 수 있습니다.
- 확장성 보조 클러스터 - 보조 영역에 읽기 전용 인스턴스 확장성을 더 추가하여 보조 클러스터를 확장하고 규모를 조정할 수 있습니다. 보조 클러스터는 읽기 전용이므로 단일 클러스터에 대한 일반적인 제한인 15개가 아닌 최대 16개의 읽기 전용 복제본 인스턴스를 지원할 수 있습니다.

- 일차적 클러스터에서 이차 클러스터로 빠른 복제 - 글로벌 클러스터에서 수행하는 복제는 기본 데이터베이스 클러스터에 대한 성능 영향이 거의 없습니다. DB 인스턴스의 리소스는 애플리케이션 읽기/쓰기 워크로드 처리 전용입니다.

글로벌 클러스터의 현재 한계는 무엇입니까?

- Amazon DocumentDB v3.6에서는 글로벌 클러스터가 지원되지 않습니다.
- t3, t4g 및 r4 인스턴스 유형에서는 글로벌 클러스터가 지원되지 않습니다.
- 글로벌 클러스터는 남아메리카(상파울루), 유럽(밀라노), 중국(베이징), 중국(닝샤) 리전에서는 사용할 수 없습니다.
- 리전이 다른 엔진 버전에 있는 경우 전환 및 글로벌 장애 조치가 지원되지 않습니다. 엔진 버전 불일치가 있는 경우 수동 장애 조치가 지원됩니다.
- 기본 클러스터만 쓰기 작업을 수행합니다. 쓰기 작업을 수행하는 클라이언트는 기본 클러스터의 DB 클러스터 엔드포인트에 연결합니다.
- 클러스터에는 5개의 보조 리전 및 1개의 기본 리전이 있을 수 있습니다.
- 보조 클러스터는 중지할 수 없습니다. 기본 클러스터에 보조 클러스터가 연결되어 있는 경우 기본 클러스터를 중지할 수 없습니다. 보조 클러스터가 없는 리전 클러스터만 중지할 수 있습니다.
- 보조 클러스터에 연결된 복제본은 특정 상황에서 다시 시작될 수 있습니다. 기본 영역의 인스턴스가 재시작되거나 장애 조치가 이뤄지는 경우, 보조 영역의 복제본도 재시작됩니다. 그러면 모든 복제본이 주 데이터베이스 클러스터의 작성자 인스턴스와 다시 동기화될 때까지 클러스터를 사용할 수 없습니다. 이는 예상된 동작입니다. 기본 클러스터를 변경하기 전에 글로벌 클러스터에 미치는 영향을 이해해야 합니다.
- 보조 클러스터에서는 변경 스트림을 사용할 수 없습니다.

주제

- 빠른 시작 설명서: 글로벌 클러스터
- Amazon DocumentDB 글로벌 클러스터 관리
- Amazon DocumentDB 글로벌 클러스터에 연결
- Amazon DocumentDB 글로벌 클러스터 모니터링
- 재해 복구 및 Amazon DocumentDB 글로벌 클러스터

빠른 시작 설명서: 글로벌 클러스터

주제

- 구성
- Amazon DocumentDB 글로벌 클러스터 생성
- Amazon DocumentDB 글로벌 클러스터 AWS 리전 에 추가
- Amazon DocumentDB 글로벌 클러스터에 스냅샷 사용

구성

Amazon DocumentDB 글로벌 클러스터는 최소 2개에 걸쳐 있습니다 AWS 리전. 기본 리전은 기본(라이터) 인스턴스 하나와 최대 15개의 복제본 인스턴스가 있는 클러스터를 지원하는 반면, 보조 리전은 최대 16개의 복제본 인스턴스로 구성된 읽기 전용 클러스터를 실행합니다. 글로벌 클러스터에는 최대 5개의 보조 리전이 있을 수 있습니다. 이 표에는 글로벌 클러스터에서 허용되는 최대 클러스터, 인스턴스 및 복제본이 나열됩니다.

설명	기본 AWS 리전	보조 AWS 리전
클러스터	1	5(최대)
라이터 인스턴스	1	0
클러스터당 읽기 전용 인스턴스(Amazon DocumentDB 복제본)	15(최대)	16(최대)
읽기 전용 인스턴스(최대 허용, 지정된 보조 리전 수)	15 - s	s = 총 보조 수 AWS 리전

클러스터의 특정 요구 사항은 다음과 같습니다.

- 데이터베이스 인스턴스 클래스 요구 사항 - `db.r5` 및 `db.r6g` 인스턴스 클래스만 사용할 수 있습니다.
- AWS 리전 요구 사항 - 기본 클러스터는 한 리전에 있어야 하며, 하나 이상의 보조 클러스터는 동일한 계정의 다른 리전에 있어야 합니다. 최대 5개의 보조(읽기 전용) 클러스터를 생성할 수 있으며 각 클러스터는 다른 리전에 있어야 합니다. 즉, 두 클러스터가 동일한 리전에 있을 수 없습니다.

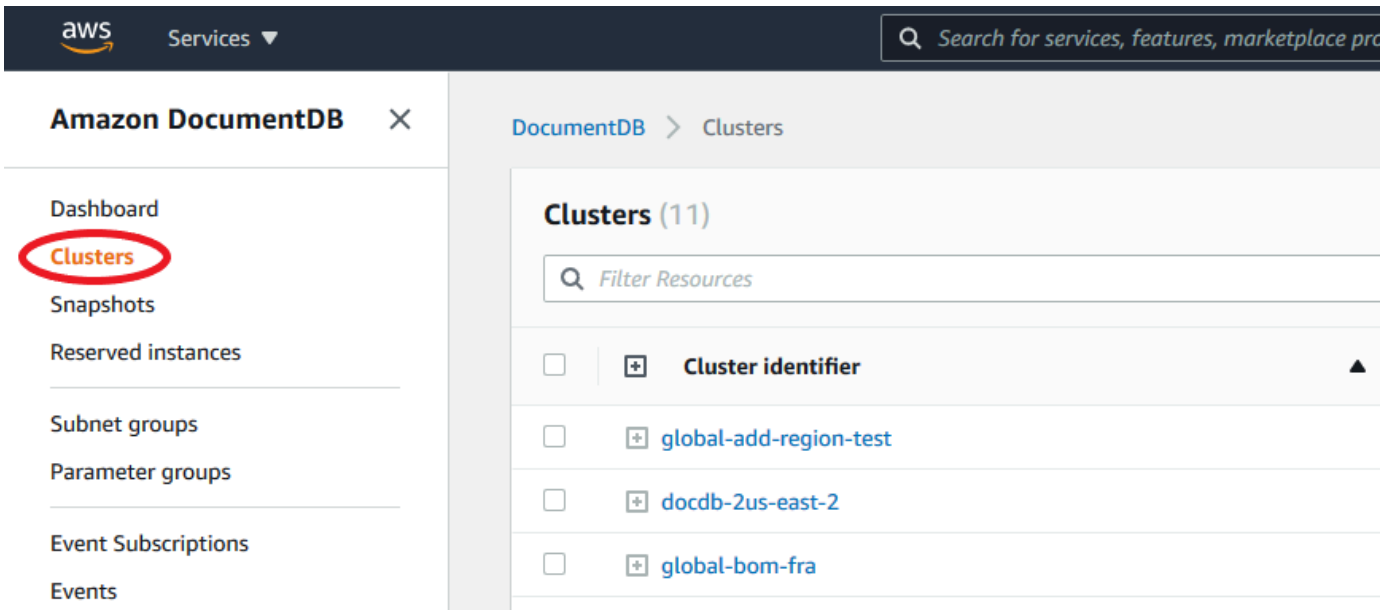
- 이름 지정 요구 사항 - 각 클러스터에 대해 선택한 이름은 모든 리전에서 고유해야 합니다. 다른 리전에 있더라도 다른 클러스터에 동일한 이름을 사용할 수 없습니다.

Amazon DocumentDB 글로벌 클러스터 생성

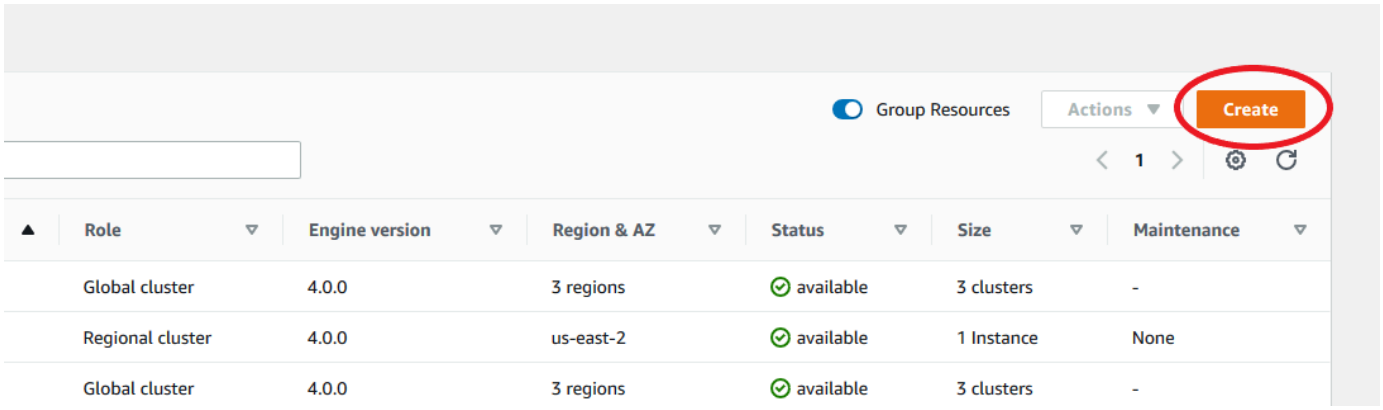
첫 번째 글로벌 클러스터를 구축할 준비가 되셨나요? 이 섹션에서는 AWS CLI 다음 지침에 따라 AWS Management Console 또는를 사용하여 새 데이터베이스 클러스터 및 인스턴스를 사용하여 새로운 글로벌 클러스터를 생성하는 방법을 설명합니다.

사용 AWS Management Console

1. 에서 Amazon DocumentDB로 AWS Management Console이동합니다.
2. Amazon DocumentDB 콘솔로 이동하면 클러스터를 선택합니다.

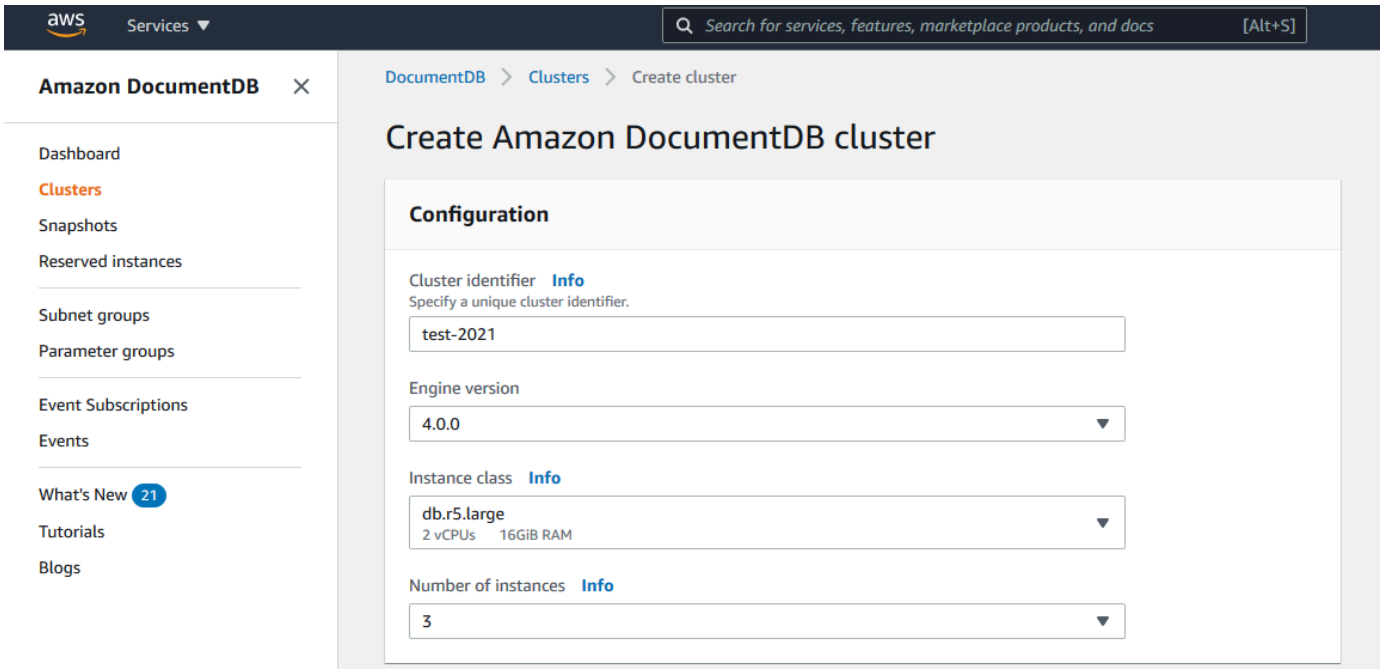


3. 생성(Create)을 선택합니다.

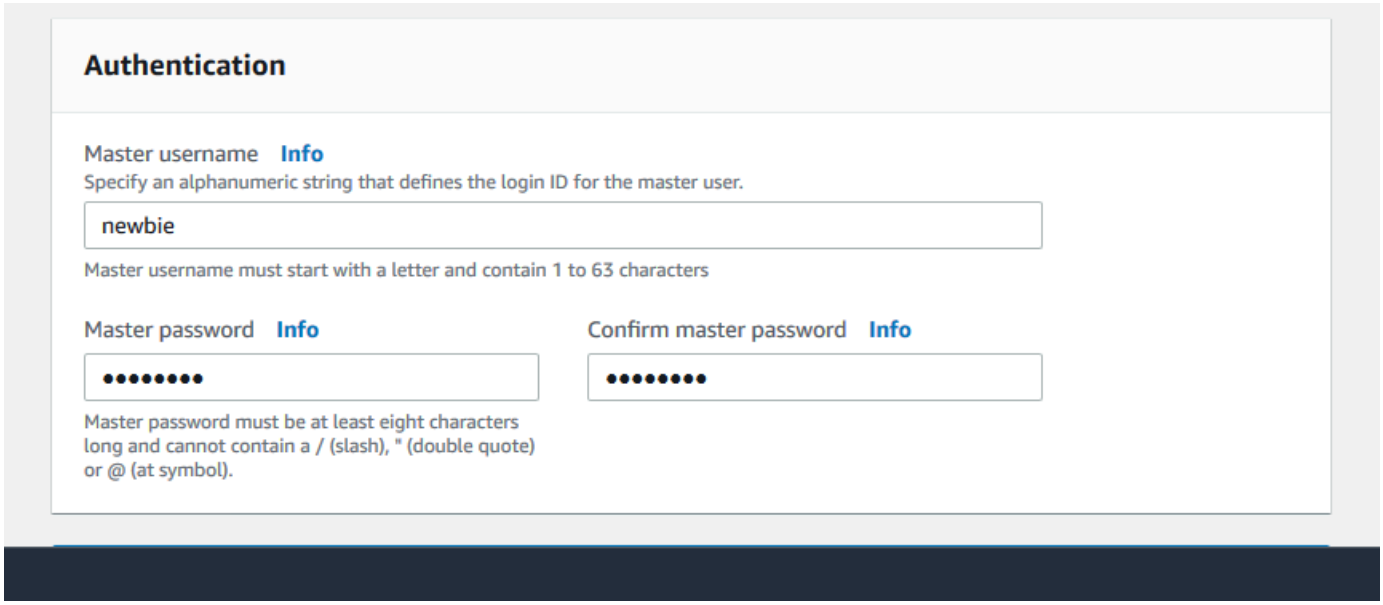


4. Amazon DocumentDB 클러스터 생성 양식의 구성 섹션을 적절히 작성하세요.

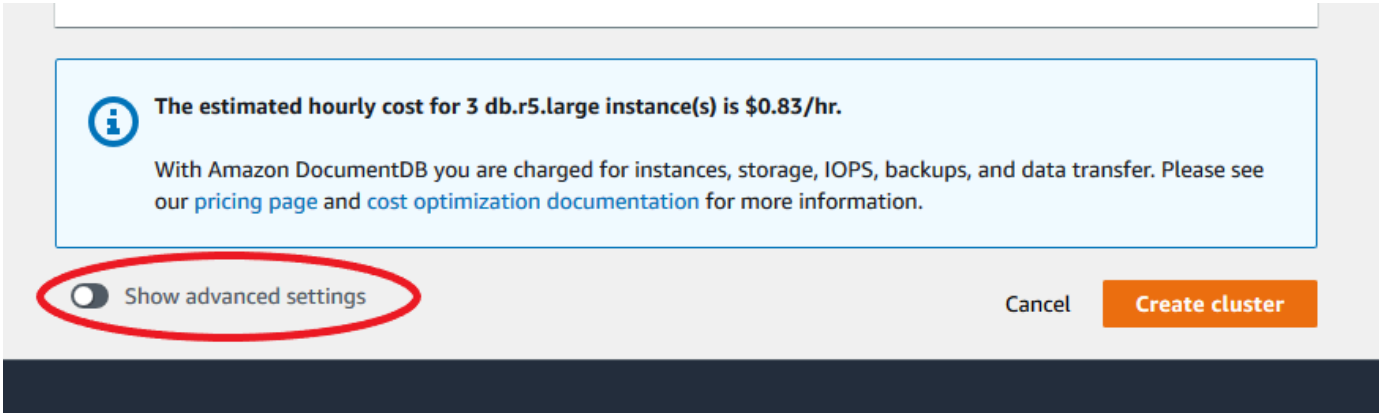
- 클러스터 식별자 - 이 인스턴스의 고유 식별자를 입력하거나 Amazon DocumentDB가 클러스터 식별자를 기반으로 인스턴스 식별자를 제공하도록 허용할 수 있습니다.
- 엔진 버전: 4.0.0 선택
- 인스턴스 클래스: db.r5.large를 선택합니다.
- 인스턴스 수에 3을 선택합니다.



5. 인증 섹션에서 마스터 사용자 이름과 마스터 비밀번호를 입력합니다.

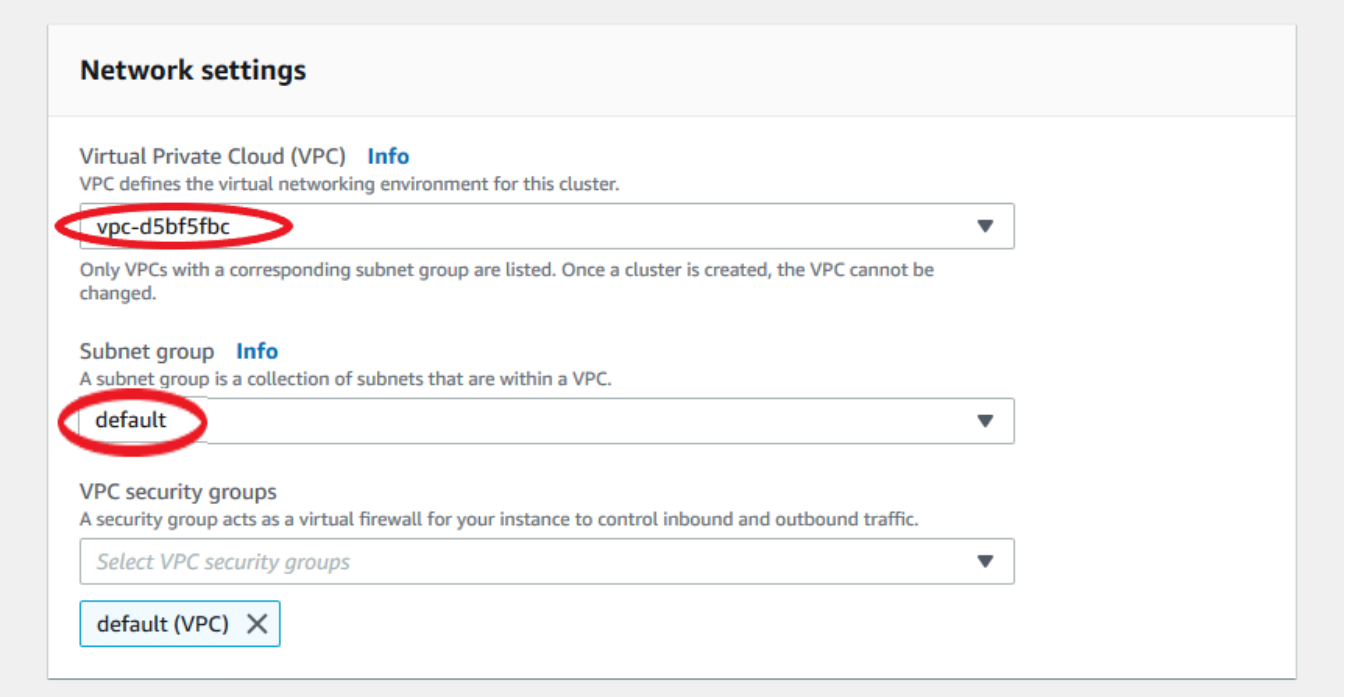


6. 고급 설정 표시를 선택합니다.

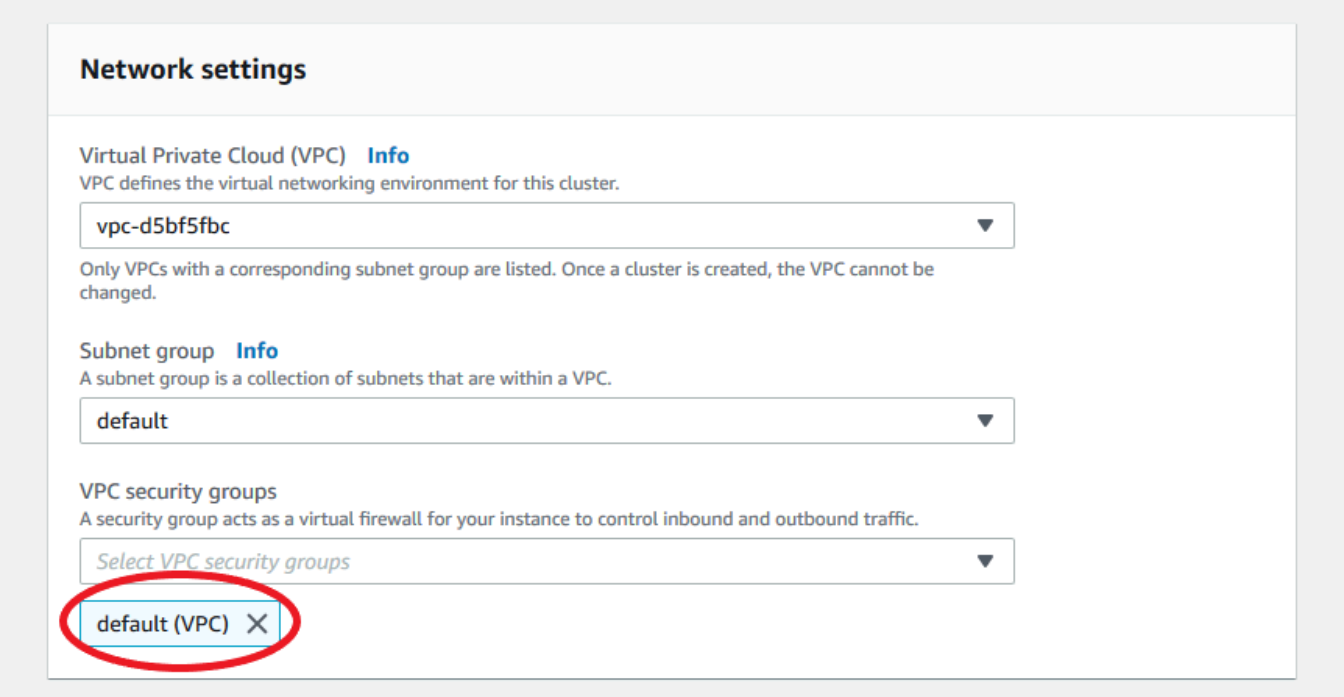


7. 네트워크 설정 섹션에서 다음을 수행합니다.

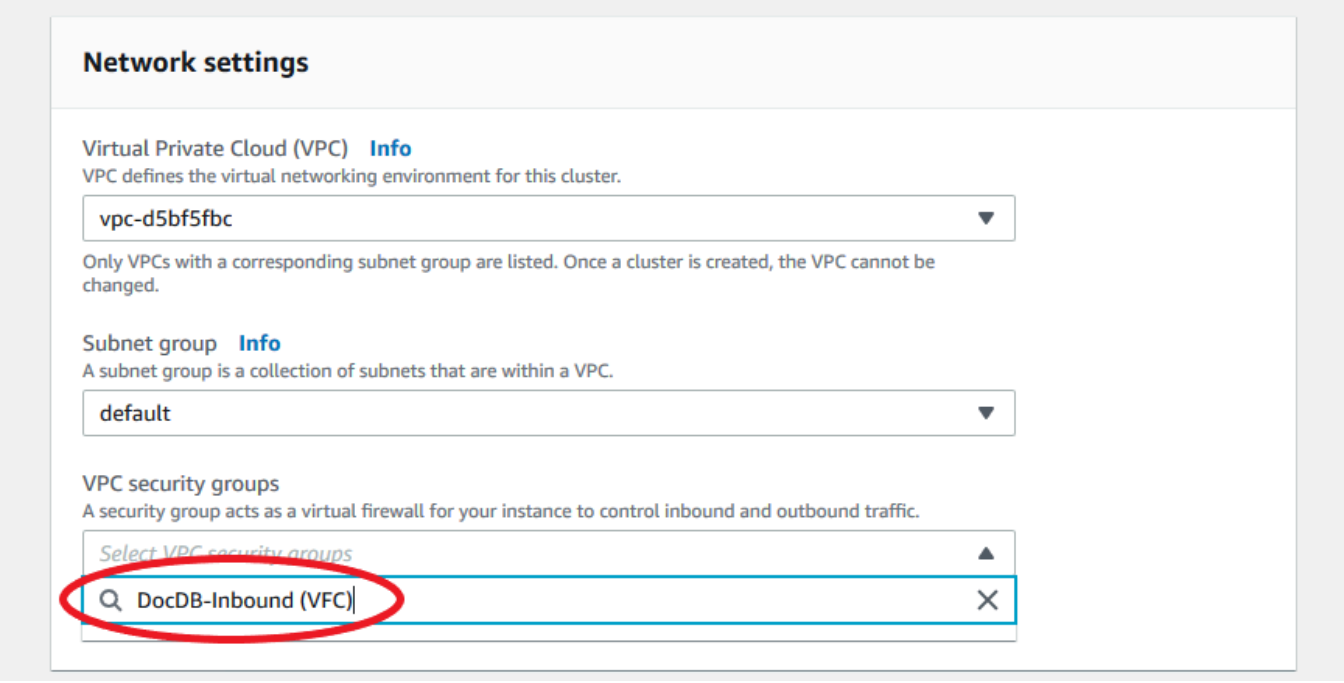
   - Virtual Private Cloud(VPC) 및 서브넷 그룹의 기본 옵션을 유지합니다.



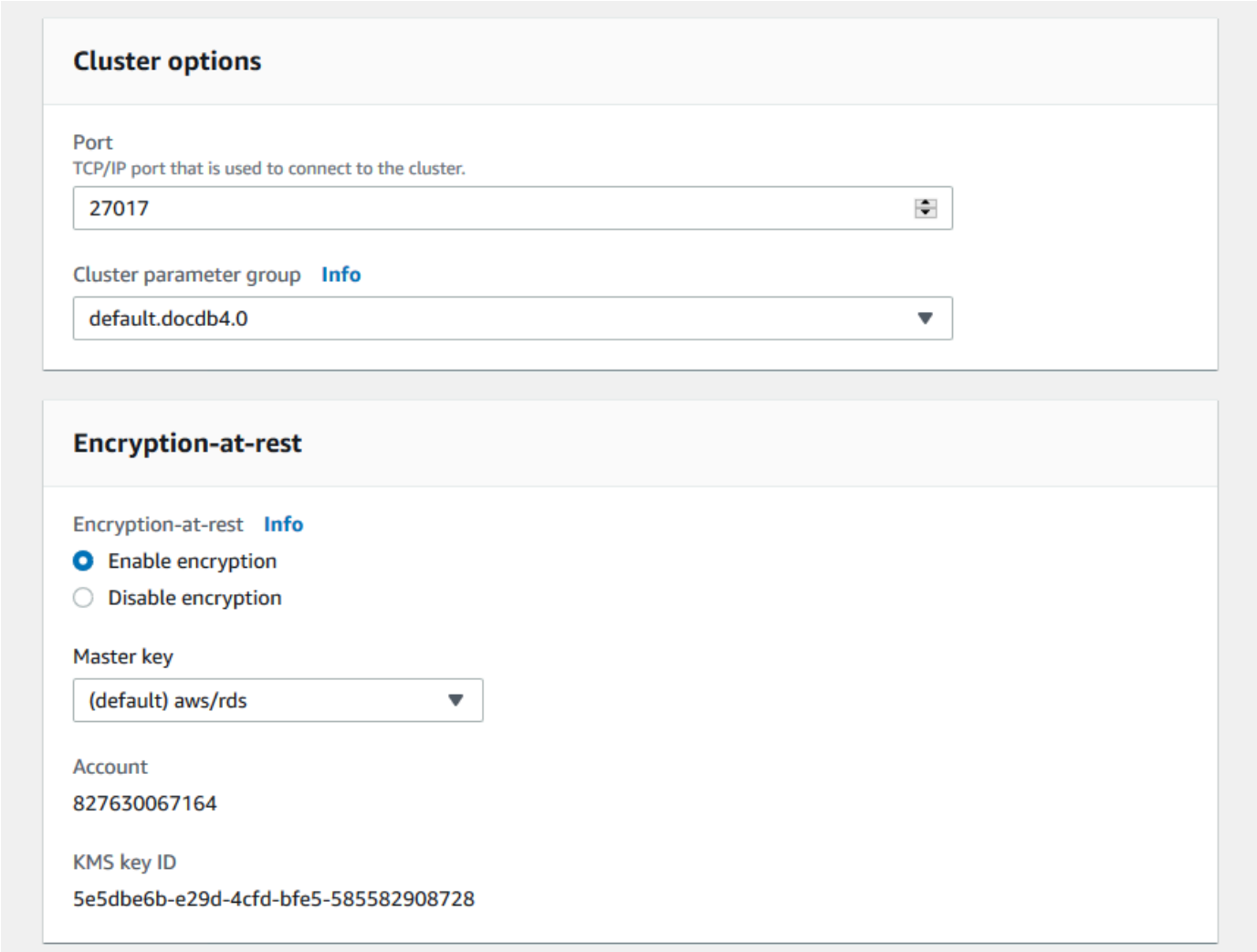
   - VPC 보안 그룹의 경우 기본값(VPC)이 이미 추가되어 있어야 합니다.



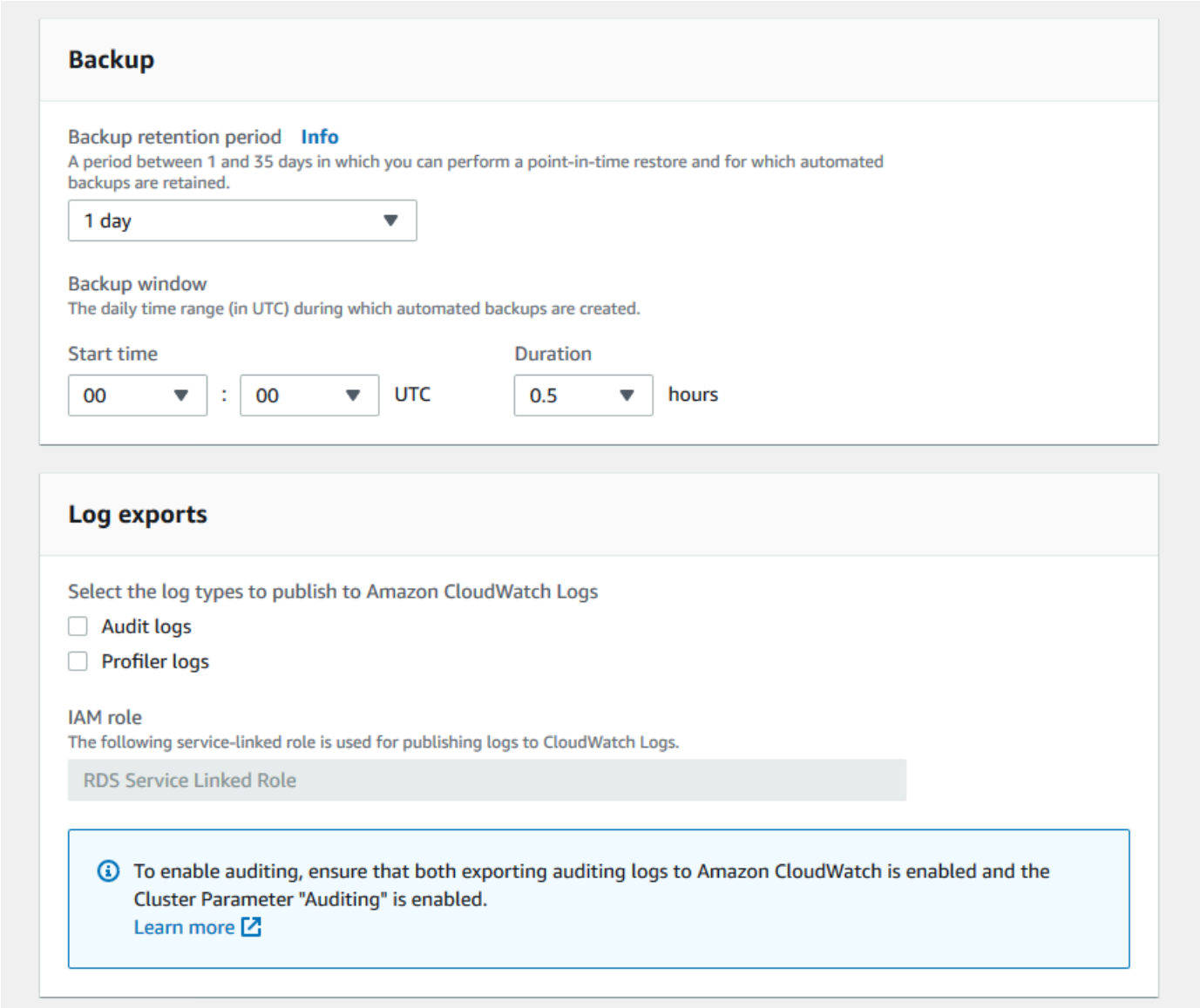
- VPC 보안 그룹 필드에 DocDB를 입력하고 DocDB-Inbound(VFC)를 선택합니다.



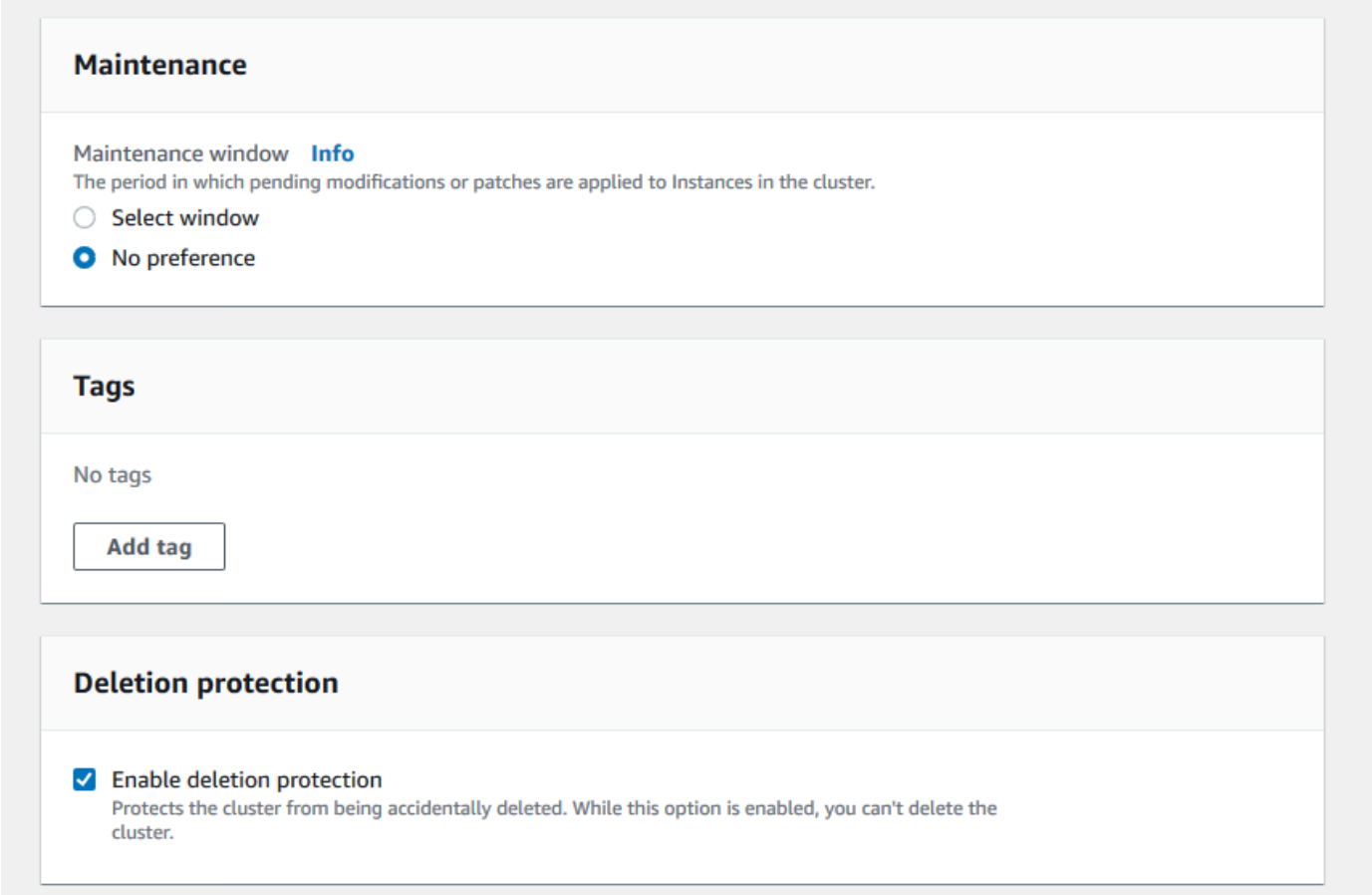
8. 클러스터 옵션 및 Encryption-at-rest 경우 기본 선택 항목을 그대로 둡니다.



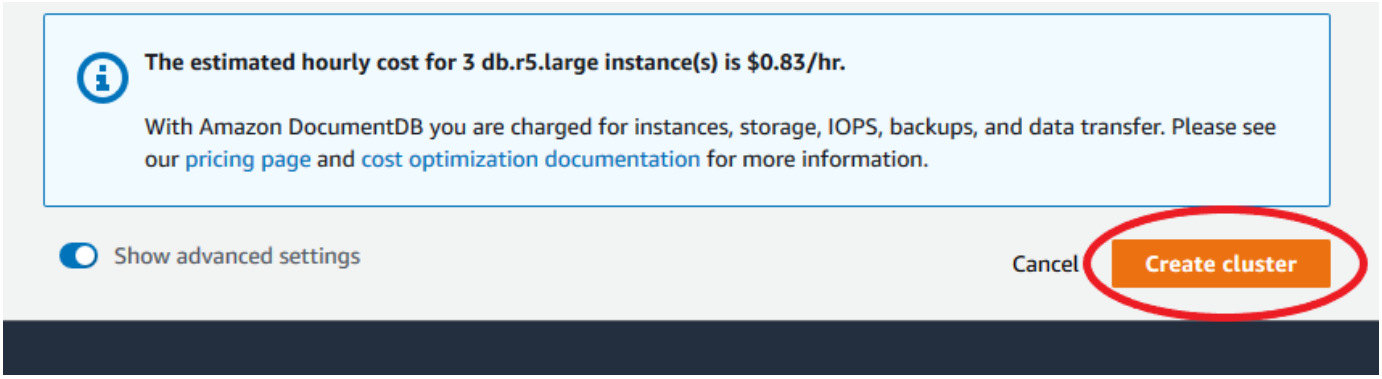
9. 백업 및 로그 내보내기의 경우 기본 선택 항목을 그대로 둡니다.



10. 유지 관리, 태그 및 삭제 보호의 경우 기본 선택 항목을 그대로 둡니다.



11. 이제 클러스터 생성 버튼을 클릭합니다.



사용 AWS CLI

Amazon DocumentDB 리전 클러스터를 생성하려면 create-global-cluster AWS CLI를 호출합니다. 다음 AWS CLI 명령은 라는 Amazon DocumentDB 클러스터를 생성합니다`global-cluster-id`. 삭제 방지에 대한 자세한 내용은 Amazon DocumentDB 클러스터 삭제을(를) 참조하세요.

또한 `--engine-version`은 기본적으로 최신 주요 엔진 버전으로 설정되는 선택적 파라미터입니다. 현재 메이저 엔진 버전은 `5.0.0`입니다. 새 메이저 엔진 버전이 릴리스되면의 기본 엔진 버전이 마지막 메이저 엔진 버전을 반영하도록 `--engine-version` 업데이트됩니다. 따라서 프로덕션 워크로드, 특히 스크립팅, 자동화 또는 AWS CloudFormation 템플릿에 의존하는 워크로드의 경우를 의도한 메이저 버전`--engine-version`에 명시적으로 지정하는 것이 좋습니다.

`db-subnet-group-name` 또는를 지정`vpc-security-group-id`하지 않으면 Amazon DocumentDB는 지정된 리전에 대해 기본 서브넷 그룹과 Amazon VPC 보안 그룹을 사용합니다.

다음 예제에서는 자신의 정보로 각각의 *### ## ## ###*를 바꿉니다.

Linux, macOS 또는 Unix의 경우:

```
aws docdb create-db-cluster \
      --global-cluster-identifier global-cluster-id \
      --source-db-cluster-identifier arn:aws:rds:us-east-1:111122223333:cluster-id
```

Windows의 경우:

```
aws docdb create-db-cluster ^
      --global-cluster-identifier global-cluster-id ^
      --source-db-cluster-identifier arn:aws:rds:us-east-1:111122223333:cluster-id
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
    "DBCluster": {
        "StorageEncrypted": false,
        "DBClusterMembers": [],
        "Engine": "docdb",
        "DeletionProtection" : "enabled",
        "ClusterCreateTime": "2018-11-26T17:15:19.885Z",
        "DBSubnetGroup": "default",
        "EngineVersion": "4.0.0",
        "MasterUsername": "masteruser",
        "BackupRetentionPeriod": 1,
        "DBClusterArn": "arn:aws:rds:us-east-1:123456789012:cluster:cluster-id",
        "DBClusterIdentifier": "cluster-id",
        "MultiAZ": false,
        "DBClusterParameterGroup": "default.docdb4.0",
```

```
        "PreferredBackupWindow": "09:12-09:42",
        "DbClusterResourceId": "cluster-KQSGI4MHU4NTDDRVNLNTU7XVAY",
        "PreferredMaintenanceWindow": "tue:04:17-tue:04:47",
        "Port": 27017,
        "Status": "creating",
        "ReaderEndpoint": "cluster-id.cluster-ro-sfcrlcjcoroz.us-
east-1.docdb.amazonaws.com",
        "AssociatedRoles": [],
        "HostedZoneId": "ZNKXTT8WH85VW",
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-77186e0d",
                "Status": "active"
            }
        ],
        "AvailabilityZones": [
            "us-east-1a",
            "us-east-1c",
            "us-east-1e"
        ],
        "Endpoint": "cluster-id.cluster-sfcrlcjcoroz.us-east-1.docdb.amazonaws.com"
    }
}
```

클러스터를 생성하는 데 몇 분 정도 걸립니다. AWS Management Console 또는를 사용하여 클러스터의 상태를 AWS CLI 모니터링할 수 있습니다. 자세한 내용은 Amazon DocumentDB 클러스터 상태 모니터링 단원을 참조하십시오.

> **Important**
>
> AWS CLI 를 사용하여 Amazon DocumentDB 리전 클러스터를 생성하면 인스턴스가 생성되지 않습니다. 따라서 기본 인스턴스와 필요한 복제 인스턴스를 명시적으로 생성해야 합니다. 콘솔 또는를 사용하여 인스턴스 AWS CLI 를 생성할 수 있습니다. 자세한 내용은 Amazon DocumentDB API 참조에서 클러스터에 Amazon DocumentDB 인스턴스 추가 및 CreateDBCluster을 참조하세요.

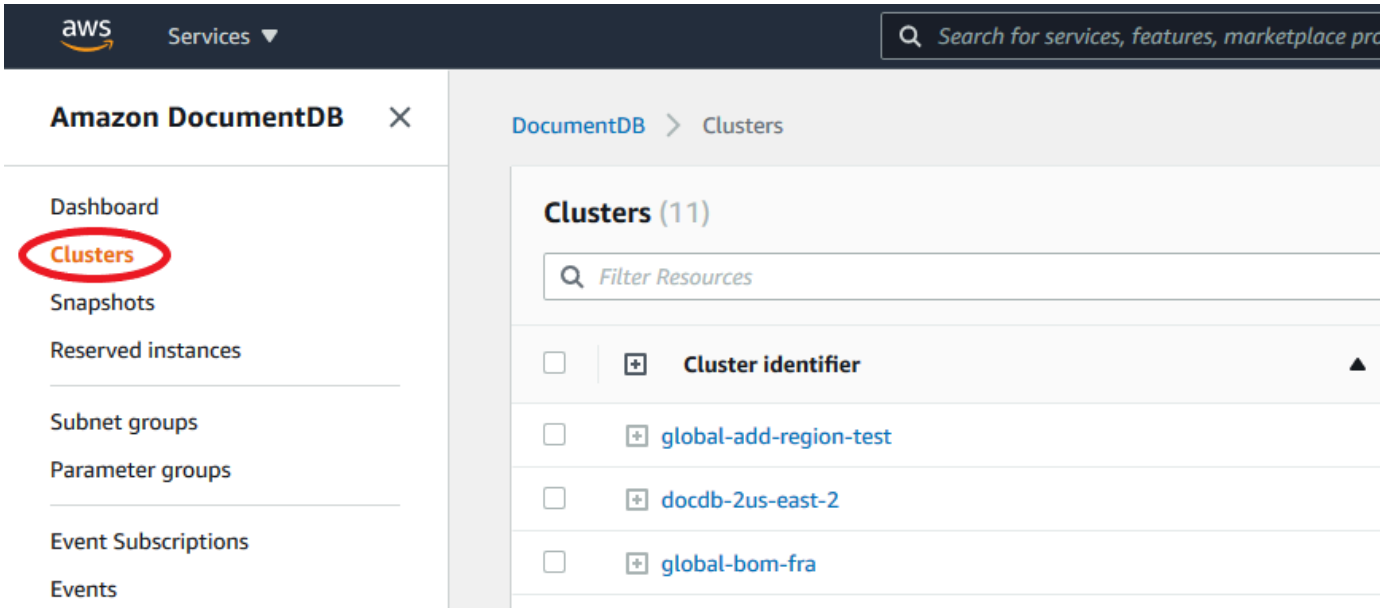
리전 클러스터를 사용할 수 있게 되면 다음 지침에 따라 다른 리전에 보조 클러스터를 추가할 수 있습니다Amazon DocumentDB 글로벌 클러스터 AWS 리전 에 추가. 리전을 추가하면 리전 클러스터가 기본 클러스터가 되고 선택한 리전에 새 보조 클러스터가 생깁니다.

Amazon DocumentDB 글로벌 클러스터 AWS 리전 에 추가

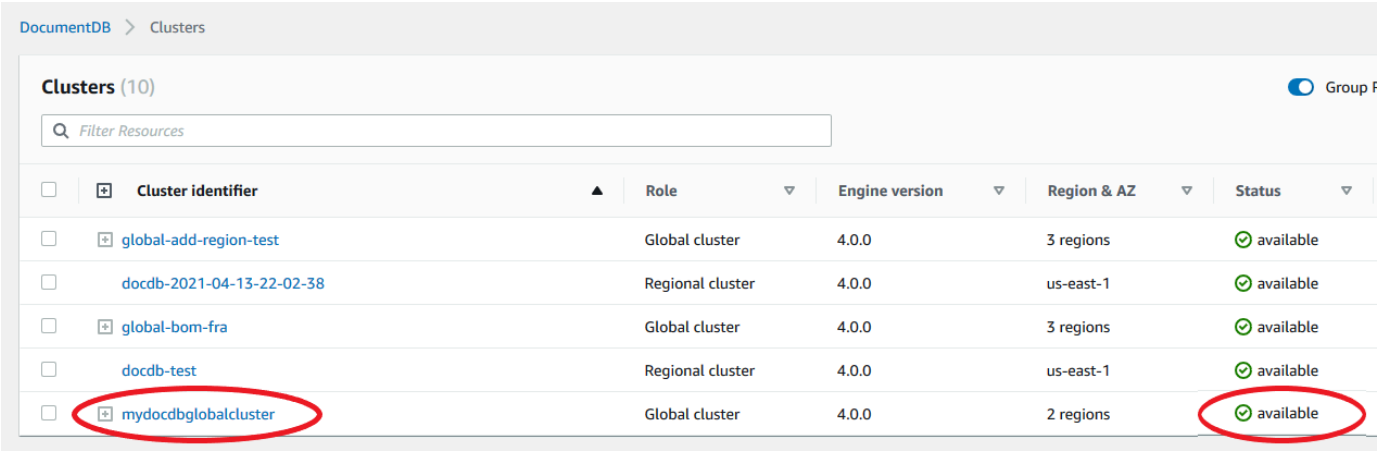
글로벌 클러스터에는 기본 클러스터와 다른 리전에 하나 이상의 보조 클러스터가 필요하며 최대 5개의 보조 클러스터를 추가할 수 있습니다. 추가하는 각 보조 클러스터에 대해 기본 클러스터에 허용되는 복제본 수를 하나씩 줄여야 한다는 점에 유의하세요. 예를 들어 글로벌 클러스터에 보조 리전이 5개 있는 경우 기본 클러스터에는 10개(15개가 아님)의 복제본만 있을 수 있습니다. 자세한 내용은 Amazon DocumentDB 글로벌 클러스터의 구성 요구 사항을 참조하세요.

사용 AWS Management Console

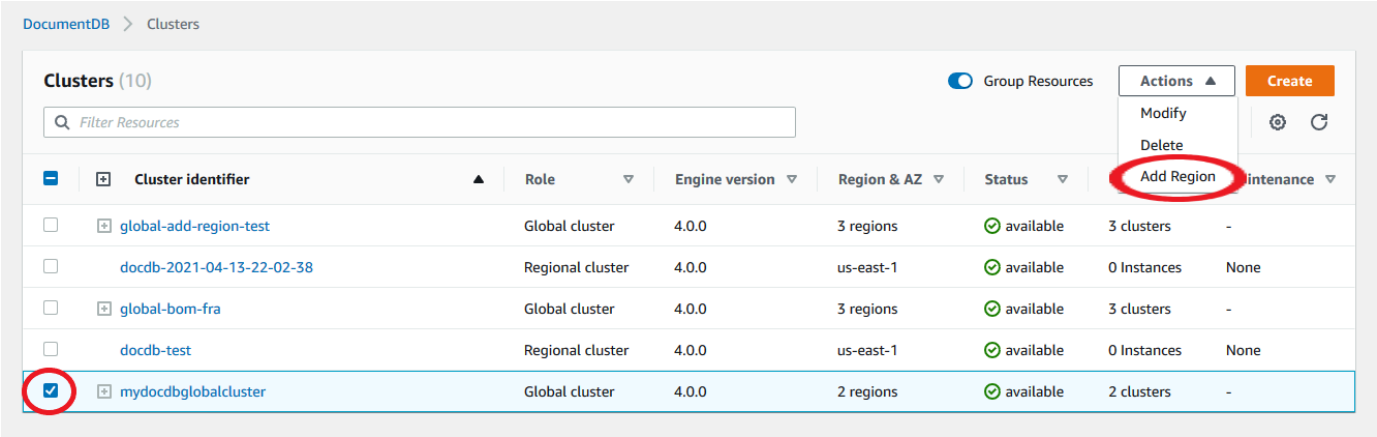
1. 에 로그인 AWS Management Console 하고 Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 클러스터를 선택합니다.



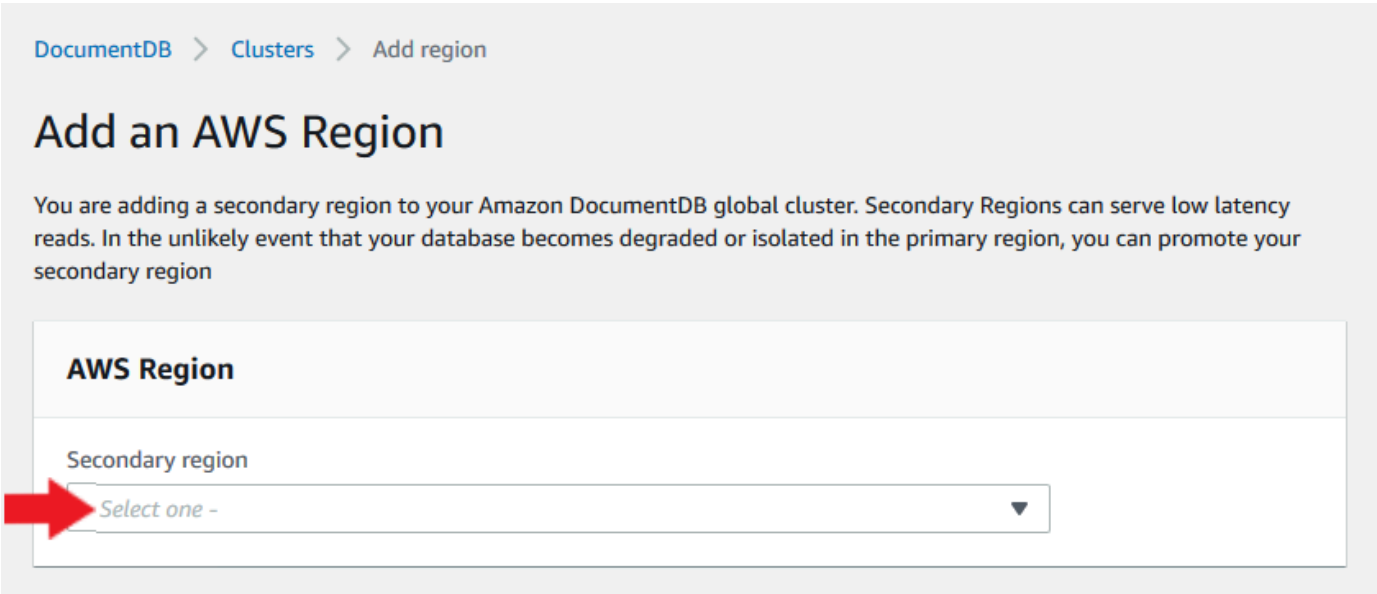
3. 보조 클러스터를 추가할 클러스터를 선택합니다. 클러스터가 `Available`인지 확인합니다.



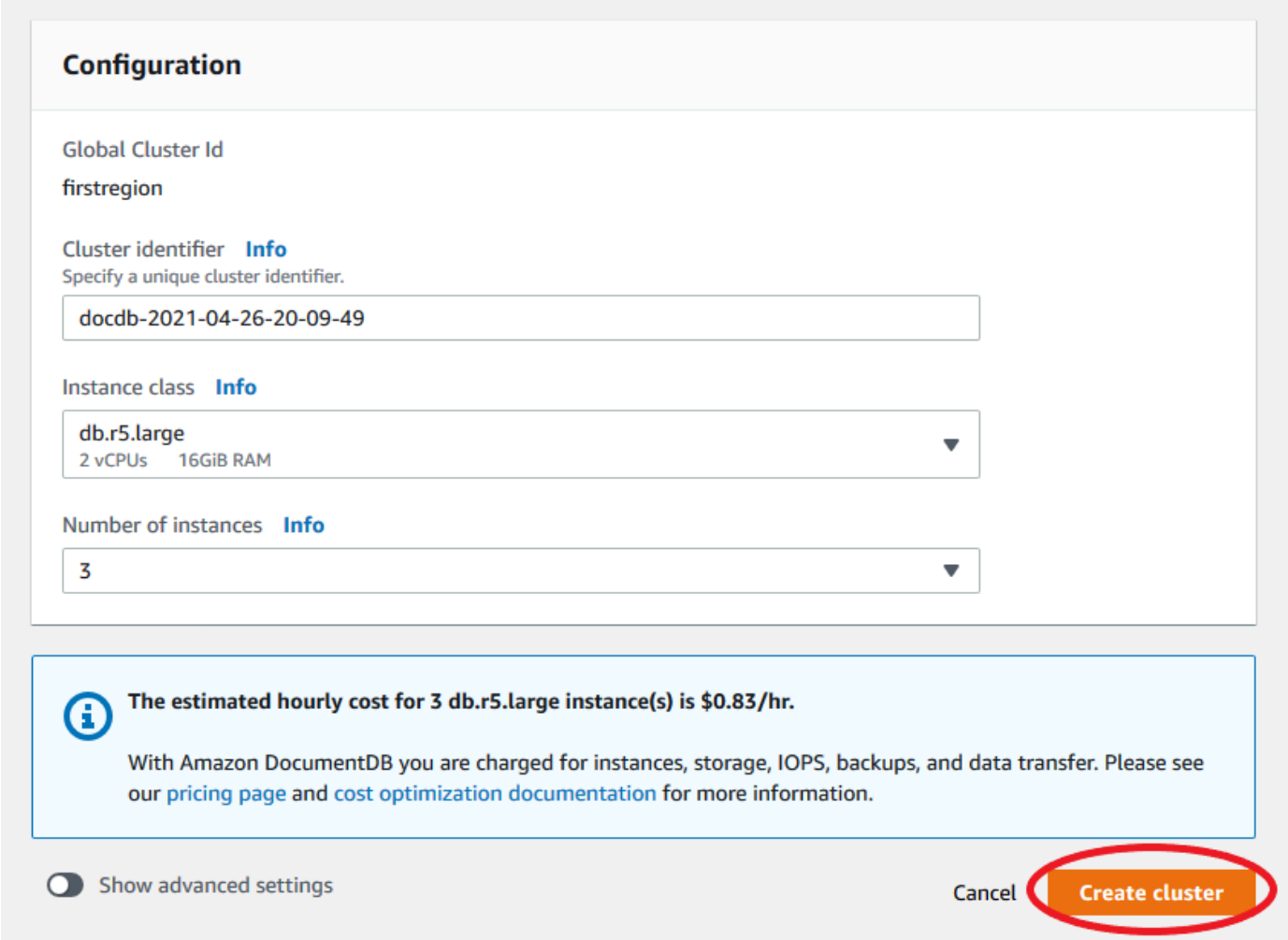
4. 작업의 드롭다운 목록을 선택한 다음 리전 추가를 선택합니다.



5. 추가 AWS 리전 페이지에서 보조 리전을 선택합니다. 동일한 글로벌 클러스터에 대한 보조 클러스터가 이미 있는 리전은 선택할 수 없습니다. 또한 기본 클러스터와 동일한 리전일 수 없습니다. 추가하는 첫 번째 리전인 경우 선택한 글로벌 클러스터 식별자도 지정해야 합니다.



6. 새 리전에서 보조 클러스터의 나머지 필드를 작성한 다음 클러스터 생성을 선택합니다. 리전 추가를 완료하면의 클러스터 목록에서 리전을 볼 수 있습니다 AWS Management Console.



사용 AWS CLI

- `create-db-cluster` CLI 명령을 글로벌 클러스터의 이름 (`--global-cluster-identifier`)과 함께 사용합니다. 기타 명령 파라미터에서 다음을 수행합니다.
  - 에서 기본 리전 AWS 리전 과 다른을 `--region`선택합니다.
  - `--engine` 및 `--engine-version` 파라미터의 구체적인 값을 선택합니다.
  - 암호화된 클러스터의 경우 기본를 암호화를 `--source-region` 위한 AWS 리전 로 지정합니다.

다음 예제에서는 새 Amazon DocumentDB 클러스터를 생성하여 글로벌 클러스터에 읽기 전용 보조 클러스터로 연결합니다. 마지막 단계에서 인스턴스가 새 클러스터에 추가됩니다.

다음 예제에서는 자신의 정보로 각각의 *### ## ## ###*를 바꿉니다.

Linux, macOS 또는 Unix의 경우:

```
aws docdb --region secondary-region-id \
  create-db-cluster \
    --db-cluster-identifier cluster-id \
    --global-cluster-identifier global-cluster-id \
    --engine-version version \
    --engine docdb

aws docdb --region secondary-region-id \
  create-db-instance \
    --db-cluster-identifier cluster-id \
    --global-cluster-identifier global-cluster-id \
    --engine-version version \
    --engine docdb
```

Windows의 경우:

```
aws docdb --region secondary-region-id ^
  create-db-cluster ^
    --db-cluster-identifier cluster-id ^
    --global-cluster-identifier global-cluster-id ^
    --engine-version version ^
    --engine docdb

aws docdb --region secondary-region-id ^
  create-db-instance ^
    --db-cluster-identifier cluster-id ^
    --global-cluster-identifier global-cluster-id ^
    --engine-version version ^
    --engine docdb
```

Amazon DocumentDB 글로벌 클러스터에 스냅샷 사용

Amazon DocumentDB 클러스터의 스냅샷을 복원하여 글로벌 클러스터의 시작점으로 사용할 수 있습니다. 이렇게 하려면 스냅샷을 복원하고 새 클러스터를 생성해야 합니다. 이는 글로벌 클러스터의 기본 클러스터 역할을 합니다. 그런 다음 복원된 클러스터에 다른 리전을 추가하여 글로벌 클러스터로 변환할 수 있습니다.

Amazon DocumentDB 글로벌 클러스터 관리

글로벌 클러스터를 구성하는 개별 클러스터에서 대부분의 관리 작업을 수행할 수 있습니다. 콘솔의 데이터베이스 페이지에서 관련 리소스 그룹화를 선택하면, 연결된 글로벌 클러스터 아래에 그룹화된 기본 클러스터와 보조 클러스터를 볼 수 있습니다.

글로벌 클러스터의 구성 탭에는 클러스터가 실행 중인 AWS 리전 위치, 버전 및 글로벌 클러스터 식별자가 표시됩니다.

주제

- Amazon DocumentDB 글로벌 클러스터 수정
- 파라미터 Amazon DocumentDB 글로벌 클러스터 수정
- Amazon DocumentDB 글로벌 클러스터에서 클러스터 분리
- Amazon DocumentDB 글로벌 클러스터에서 클러스터 제거
- 보조 리전에 헤드리스 Amazon DocumentDB 클러스터 생성

Amazon DocumentDB 글로벌 클러스터 수정

의 클러스터 페이지에는 모든 글로벌 클러스터가 AWS Management Console 나열되며, 각 클러스터의 기본 클러스터와 보조 클러스터가 표시됩니다. 글로벌 클러스터에는 고유한 구성 설정이 있습니다. 특히, 기본 및 보조 클러스터와 관련된 리전이 있습니다.

글로벌 클러스터를 변경하면 변경 사항을 취소 할 수 있는 기회가 제공됩니다.

계속을 선택하면 변경 사항이 승인됩니다.

파라미터 Amazon DocumentDB 글로벌 클러스터 수정

글로벌 클러스터 내에서 각 클러스터에 대해 클러스터 파라미터 그룹을 별도로 구성할 수 있습니다. 대부분의 파라미터는 다른 종류의 Amazon DocumentDB 클러스터와 동일하게 작동합니다. 전역 데이터베이스의 모든 클러스터 간에 설정을 일관성 있게 유지하는 것이 좋습니다. 이렇게 하면 보조 클러스터를 기본 클러스터로 승격하는 경우에 예상치 못한 동작 변경을 방지할 수 있습니다.

예를 들면 다른 클러스터가 기본 클러스터로 대신 사용되는 경우 일관되지 않은 동작을 방지하려면 시간대와 문자 세트에 대해 동일한 설정을 사용합니다.

Amazon DocumentDB 글로벌 클러스터에서 클러스터 분리

글로벌 클러스터에서 클러스터를 제거해야 하는 상황은 여러 가지가 있습니다. 예를 들어, 기본 클러스터가 성능이 저하되거나 격리된 경우 글로벌 클러스터에서 클러스터를 제거할 수 있습니다. 그런 다음, 프로비저닝된 독립형 클러스터가 되어, 새로운 글로벌 클러스터를 생성하는 데 사용할 수 있습니다. 자세한 내용은 Amazon DocumentDB 글로벌 클러스터에 대한 수동 장애 조치 수행을 참조하십시오.

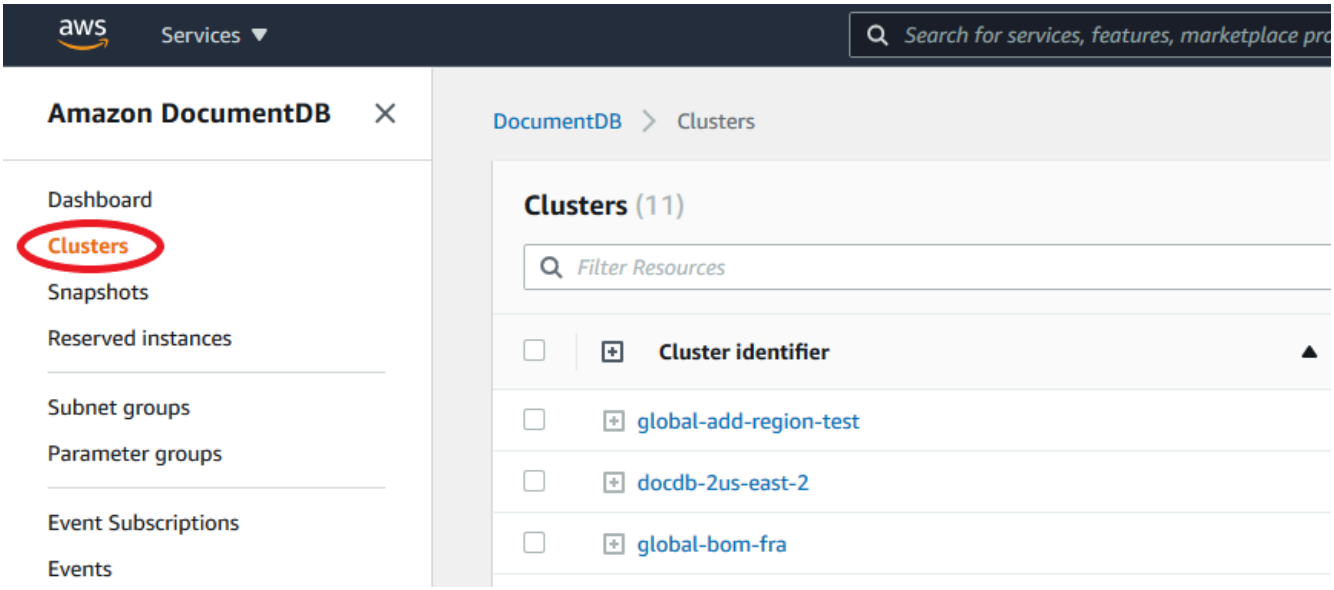
더 이상 필요하지 않은 글로벌 클러스터를 삭제하고자 하기 때문에 클러스터를 제거하고 싶을 수도 있습니다. 연결된 모든 클러스터를 분리하고 기본 클러스터를 마지막으로 남겨둘 때까지는 글로벌 클러스터를 삭제할 수 없습니다. 자세한 내용은 Amazon DocumentDB 글로벌 클러스터에서 클러스터 제거 단원을 참조하십시오.

> **Note**
>
> 클러스터가 글로벌 클러스터에서 분리되면, 더 이상 기본 클러스터와 동기화되지 않습니다. 완전한 읽기/쓰기 기능을 갖춘 프로비저닝된 독립형 클러스터가 됩니다. 또한 Amazon DocumentDB 콘솔에 더이상 표시되지 않습니다. 콘솔에서 클러스터가 위치한 지역을 선택한 경우에만 볼 수 있습니다.

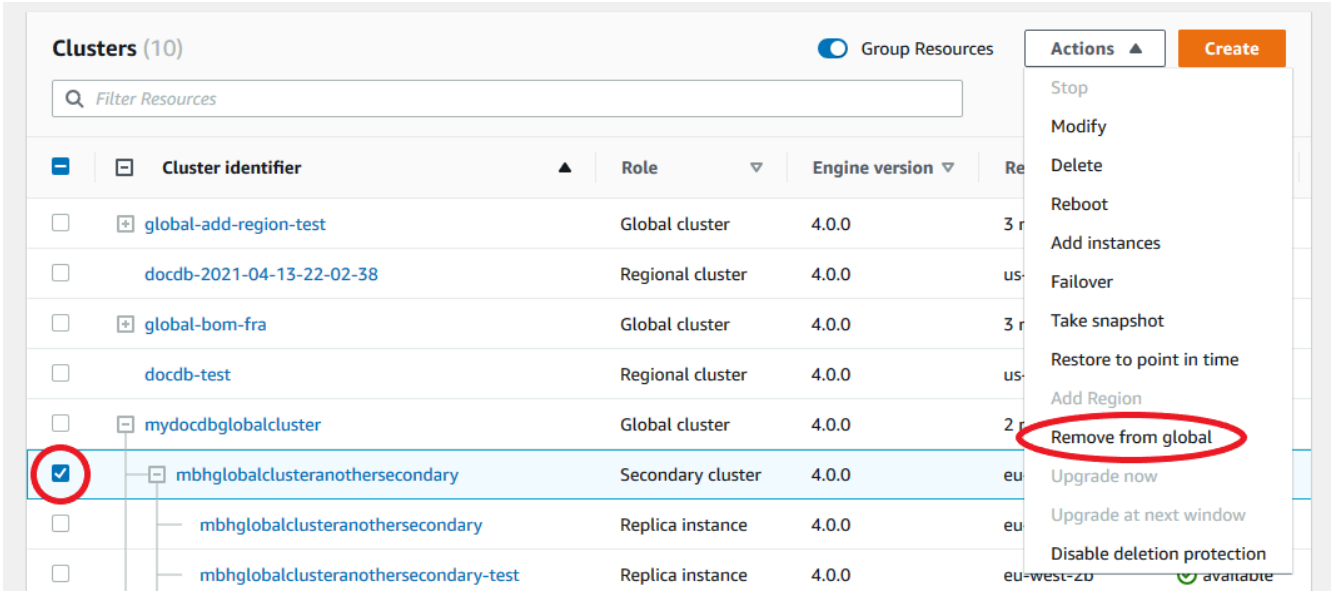
AWS Management Console AWS CLI, 또는 RDS API를 사용하여 글로벌 클러스터에서 클러스터를 제거할 수 있습니다.

Using the AWS Management Console

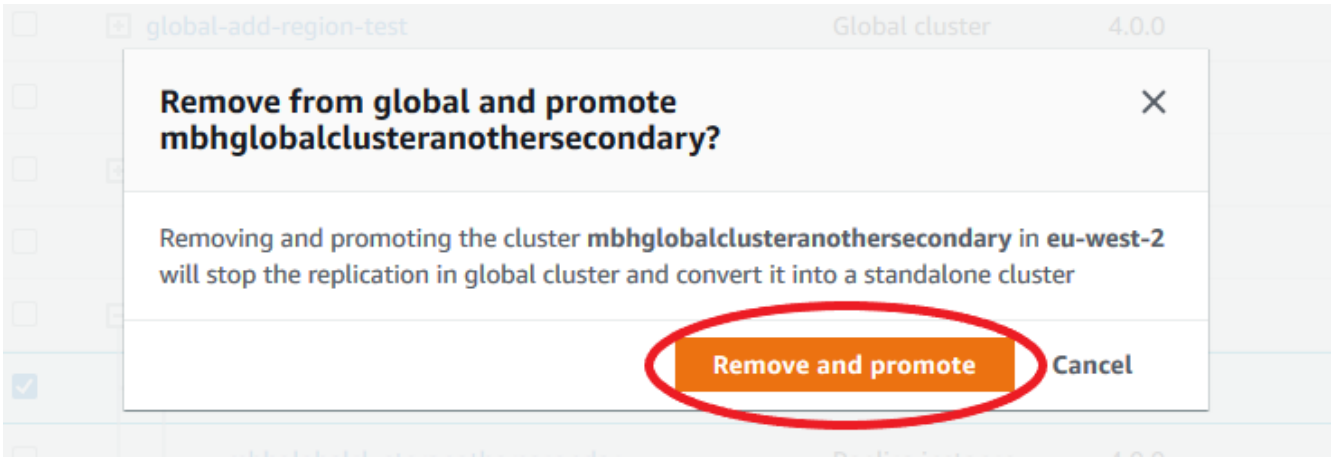
1. 에 로그인 AWS Management Console 하고 Amazon DocumentDB 콘솔로 이동합니다.
2. 왼쪽 측면의 탐색 창에서 클러스터를 선택합니다.



3. 모든 보조 클러스터를 볼 수 있도록 글로벌 클러스터를 확장하세요. 제거하려는 보조 클러스터를 선택합니다. 작업을 선택하고 드롭다운 메뉴에서 글로벌에서 제거를 선택합니다.



4. 보조 클러스터를 글로벌 클러스터에서 분리할지 확인하는 메시지가 표시됩니다. 글로벌 클러스터에서 클러스터를 제거하려면 제거 및 승격을 선택합니다.



클러스터는 더 이상 보조 클러스터로 사용되지 않으며 더 이상 기본 클러스터와 동기화되지 않습니다. 완전한 읽기/쓰기 기능을 갖춘 독립형 클러스터입니다.

보조 클러스터를 모두 제거하거나 삭제한 후 동일한 방식으로 기본 클러스터를 제거할 수 있습니다. 모든 보조 클러스터를 제거해야 기본 클러스터를 글로벌 클러스터에서 분리 또는 제거할 수 있습니다. 글로벌 클러스터는 0 리전 및 AZ가 있는 클러스터 목록에 남아 있을 수 있습니다. 이 글로벌 클러스터를 더 이상 사용하지 않을 경우 삭제할 수 있습니다.

Using the AWS CLI

글로벌 클러스터에서 클러스터를 제거하려면 `remove-from-global-cluster` CLI 명령 및 다음 파라미터를 실행합니다.

- `--global-cluster-identifier` - 글로벌 클러스터의 이름(식별자)입니다.
- `--db-cluster-identifier` - 글로벌 클러스터에서 제거할 각 클러스터의 이름입니다.

다음 명령은 글로벌 클러스터에서 보조 클러스터를 제거한 후 기본 클러스터를 제거합니다.

Linux, macOS 또는 Unix의 경우:

```
aws docdb --region secondary_region \
  remove-from-global-cluster \
    --db-cluster-identifier secondary_cluster_ARN \
    --global-cluster-identifier global_cluster_id

aws docdb --region primary_region \
  remove-from-global-cluster \
    --db-cluster-identifier primary_cluster_ARN \
```

```
    --global-cluster-identifier global_cluster_id
```

글로벌 클러스터의 각 보조 리전에 대해 `remove-from-global-cluster --db-cluster-identifier secondary_cluster_ARN` 명령을 반복합니다.

Windows의 경우:

```
aws docdb --region secondary_region ^
  remove-from-global-cluster ^
    --db-cluster-identifier secondary_cluster_ARN ^
    --global-cluster-identifier global_cluster_id

aws docdb --region primary_region ^
  remove-from-global-cluster ^
    --db-cluster-identifier primary_cluster_ARN ^
    --global-cluster-identifier global_cluster_id
```

글로벌 클러스터의 각 보조 리전에 대해 `remove-from-global-cluster --db-cluster-identifier secondary_cluster_ARN` 명령을 반복합니다.

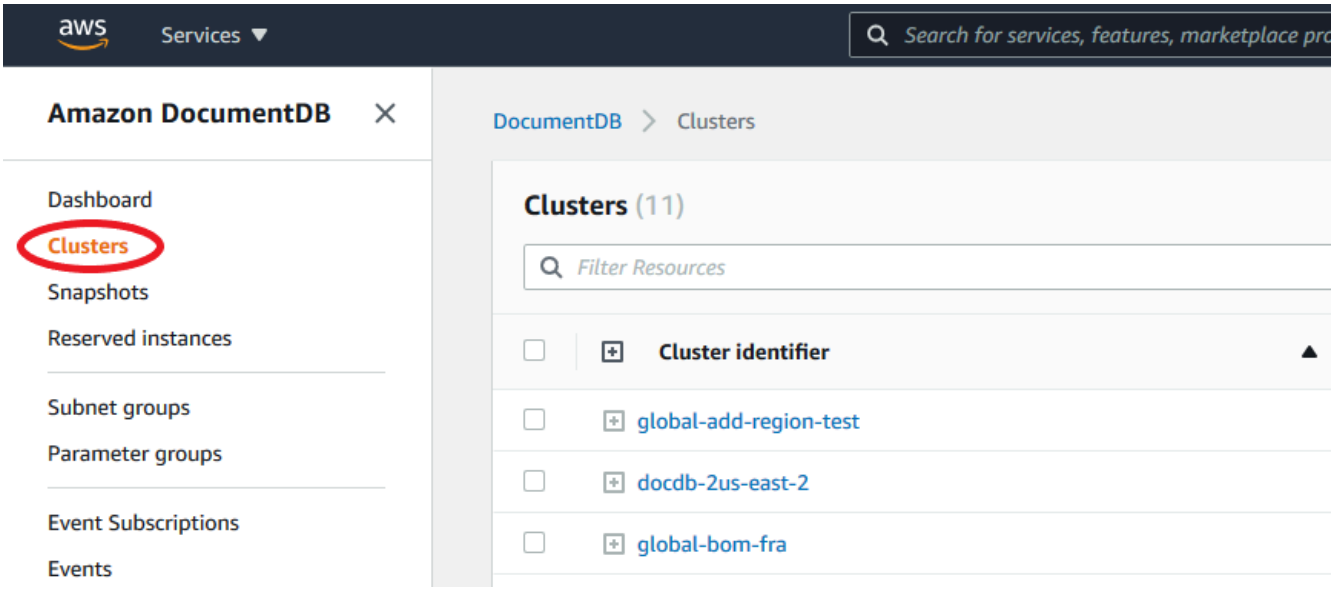
Amazon DocumentDB 글로벌 클러스터에서 클러스터 제거

글로벌 클러스터를 삭제하려면 다음을 수행합니다.

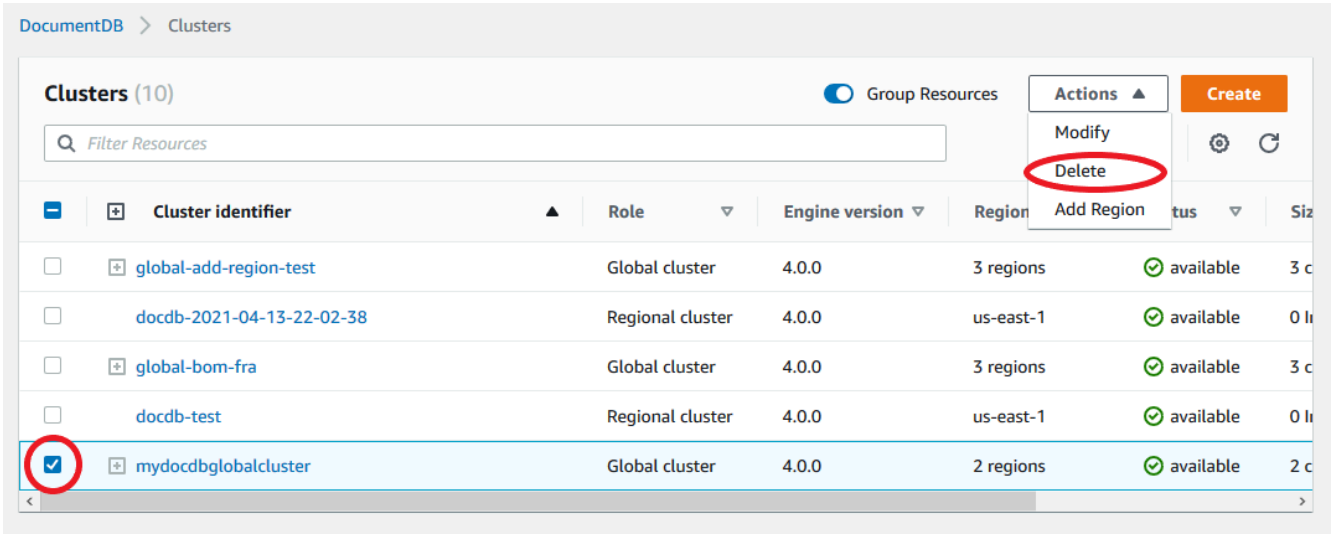
- 글로벌 클러스터에서 모든 보조 클러스터를 제거합니다. 각 클러스터는 독립형 클러스터가 됩니다. 이전 섹션인를 참조하세요Amazon DocumentDB 글로벌 클러스터에서 클러스터 분리.
- 각 독립형 클러스터에서 모든 복제본을 삭제합니다.
- 글로벌 클러스터에서 기본 클러스터를 제거합니다. 이 클러스터는 독립형 클러스터가 됩니다.
- 기본 클러스터에서 먼저 모든 복제본을 삭제한 다음 기본 인스턴스를 삭제합니다. 새로 독립 실행형 클러스터에서 기본 인스턴스를 삭제하면 일반적으로 클러스터와 글로벌 클러스터 모두가 제거됩니다.

Using the AWS Management Console

1. 에 로그인 AWS Management Console 하고 Amazon DocumentDB 콘솔로 이동합니다.
2. 클러스터를 선택하고 삭제할 글로벌 클러스터를 찾습니다.



3. 글로벌 클러스터를 선택한 상태에서 작업 메뉴에서 삭제를 선택합니다.



글로벌 클러스터에서 다른 모든 클러스터가 제거되었는지 확인합니다. 글로벌 클러스터에는 0 리전, AZ와 0 클러스터 크기가 표시되어야 합니다. 글로벌 클러스터에 클러스터가 포함되어 있으면 삭제할 수 없습니다. 먼저 이전 단계인의 지침을 따라야 합니다Amazon DocumentDB 글로벌 클러스터에서 클러스터 분리.

Using the AWS CLI

글로벌 클러스터를 삭제하려면 다음 예제와 같이 AWS 리전 의 이름과 글로벌 클러스터 식별자로 `delete-global-cluster` CLI 명령을 실행합니다.

Linux, macOS, Unix의 경우:

```
aws docdb --region primary_region delete-global-cluster \
   --global-cluster-identifier global_cluster_id
```

Windows의 경우:

```
aws docdb --region primary_region delete-global-cluster ^
   --global-cluster-identifier global_cluster_id
```

보조 리전에 헤드리스 Amazon DocumentDB 클러스터 생성

Amazon DocumentDB 글로벌 클러스터에는 기본 클러스터 AWS 리전 와 다른에 하나 이상의 보조 클러스터가 필요하지만 보조 클러스터에 헤드리스 구성을 사용할 수 있습니다. 헤드리스 세컨더리 Amazon DocumentDB 클러스터는 인스턴스가 없는 클러스터입니다. 이 유형의 구성은 글로벌 클러스터에 대한 비용을 줄일 수 있습니다. Amazon DocumentDB 클러스터에서는 컴퓨팅과 스토리지가 분리되어 있습니다. 인스턴스가 없으면 컴퓨팅 요금이 청구되지 않고 스토리지 요금만 청구됩니다. 올바르게 설정되면, 헤드리스 보조 스토리지 볼륨이 기본 클러스터와 동기화 상태를 유지합니다.

Amazon DocumentDB 글로벌 클러스터를 생성할 때 보통 때처럼 보조 클러스터를 추가합니다. 그러나 프라이머리 클러스터가 보조 클러스터로 복제되기 시작한 후에는 보조 클러스터에서 읽기 전용 DB 인스턴스를 삭제합니다. 이 보조 클러스터에는 더 이상 인스턴스가 없기 때문에 “헤드리스”로 간주됩니다. 그러나 스토리지 볼륨은 기본 Amazon DocumentDB 클러스터와 동기화 상태를 유지합니다.

> **Important**
>
> 15분 이상 지역 전반의 장애를 견딜 수 있는 고객에게만 헤드리스 클러스터를 사용하는 것이 좋습니다. 헤드리스 보조 클러스터로 지역 전반의 장애를 복구하려면 장애 조치 후 사용자가 새 인스턴스를 만들어야 하기 때문입니다. 새 인스턴스를 사용할 수 있게 되려면 약 10~15분 정도 걸릴 수 있습니다.

글로벌 클러스터에 헤드리스 보조 클러스터를 추가하는 방법

1. 에 로그인 AWS Management Console 하고 Amazon DocumentDB 콘솔을 엽니다.
2. 왼쪽 측면의 탐색 창에서 클러스터를 선택합니다.
3. 보조 클러스터가 필요한 글로벌 클러스터를 선택합니다. 기본 클러스터가 `Available`인지 확인합니다.
4. 작업에서 리전 추가를 선택합니다.

5. 리전 추가 페이지에서 보조 리전을 선택합니다.

   > **Note**
   >
   > 동일한 글로벌 클러스터에 대해 보조 클러스터가 이미 있는 리전을 선택할 수 없습니다. 또한 이는 기본 클러스터와 동일한 리전이 될 수 없습니다.

6. 새 리전에서 보조 클러스터에 대한 나머지 필드를 완료합니다. 이 옵션은 클러스터 인스턴스와 동일한 구성 옵션입니다.
7. 리전을 추가합니다. 글로벌 클러스터에 리전 추가를 완료하면 AWS Management Console의 `Clusters`에서 글로벌 클러스터의 목록에서 볼 수 있습니다.
8. AWS Management Console 또는를 사용하여 계속하기 전에 보조 클러스터와 해당 리더 인스턴스의 상태를 확인합니다 AWS CLI. 다음은 AWS CLI을 사용하는 경우의 샘플 명령입니다.

   ```
   $ aws docdb describe-db-clusters --db-cluster-identifier secondary-cluster-id --
   query '*[].[Status]' --output text
   ```

   새로 추가된 보조 클러스터의 상태가 생성에서 사용 가능으로 변경되려면 몇 분 정도 걸릴 수 있습니다. 클러스터를 사용할 수 있게 되면 리더 인스턴스를 삭제할 수 있습니다.
9. 보조 클러스터에서 리더 인스턴스를 선택한 후 삭제를 선택합니다.
10. 리더 인스턴스를 삭제한 후 보조 클러스터는 글로벌 클러스터의 일부로 남습니다. 데이터와 연결되지 않아야 합니다.

> **Note**
>
> 이러한 중단이 발생할 경우 이 헤드리스 세컨더리 Amazon DocumentDB 클러스터를 사용하여 기본 리전의 계획되지 않은 중단으로부터 Amazon DocumentDB 글로벌 클러스터를 수동으로 복구할 수 있습니다.

Amazon DocumentDB 글로벌 클러스터에 연결

글로벌 클러스터에 연결하는 방법은 클러스터에 쓸 것인지 클러스터에서 읽을 것인지에 따라 달라집니다:

- 읽기 전용 요청 또는 쿼리의 경우 자신의 AWS 리전에 있는 Aurora 클러스터 리더 엔드포인트에 연결합니다.

- 데이터 조작 언어(DML) 또는 데이터 정의 언어(DDL) 설명문을 실행하려면 기본 클러스터용 클러스터 엔드포인트에 연결합니다. 이 엔드포인트는 애플리케이션 AWS 리전 과 다른에 있을 수 있습니다.

콘솔에서 글로벌 클러스터를 볼 때 해당 모든 클러스터와 연결된 모든 범용 엔드포인트를 볼 수 있습니다.

글로벌 클러스터에 연결하는 방법은 데이터베이스에 쓸 것인지 데이터베이스에서 읽을 것인지에 따라 달라집니다. 기본 영역에서 제공할 DDL, DML 및 읽기 작업의 경우 기본 클러스터에 연결해야 합니다. 읽기 환경설정이 `secondaryPreferred=true`인 클러스터 엔드포인트를 복제본 세트 모드로 사용하여 기본 클러스터에 연결하는 것이 좋습니다. 그러면 쓰기 트래픽이 기본 클러스터의 작성자 인스턴스로 라우팅되고 읽기 트래픽이 기본 클러스터의 복제본 인스턴스로 라우팅됩니다.

리전 간 읽기 전용 트래픽의 경우 보조 클러스터 중 하나에 연결해야 합니다. 복제본 세트 모드에서 클러스터 엔드포인트를 사용하여 보조 클러스터에 연결하는 것이 좋습니다. 모든 인스턴스는 읽기 전용 복제본 인스턴스이므로 읽기 환경설정을 지정할 필요가 없습니다. 대기 시간을 최소화하려면 해당 리전 또는 가장 가까운 리전에 있는 판독기 엔드포인트를 선택합니다.

Amazon DocumentDB 글로벌 클러스터 모니터링

Amazon DocumentDB(MongoDB 호환)는 CloudWatch와 통합되므로 클러스터에 대한 운영 지표를 수집하고 분석할 수 있습니다. CloudWatch 콘솔, Amazon DocumentDB 콘솔, AWS Command Line Interface (AWS CLI) 또는 CloudWatch API를 사용하여 이러한 지표를 모니터링할 수 있습니다.

글로벌 클러스터를 모니터링하려면 다음 CloudWatch 지표를 사용합니다.

지표	설명
`GlobalClusterReplicatedWriteIO`	5분 간격으로 AWS 리전보고된 기본의 클러스터 볼륨에서 보조 AWS 리전 의 클러스터 볼륨으로 복제된 청구된 쓰기 I/O 작업의 평균 수입니다. 각 보조 리전에 복제되는 `ReplicatedWriteIOs` 횟수는 기본 리전에서 `VolumeWriteIOPs` 수행한 리전 내 복제 수와 동일합니다.

지표	설명
GlobalClusterDataTransferBytes	기본 클러스터에서 보조 클러스터로 전송 AWS 리전 된 데이터의 양은 바이트 단위로 AWS 리전측정됩니다.
GlobalClusterReplicationLag	기본 클러스터에서 보조 클러스터의 로 변경 이벤트를 복제 AWS 리전 할 때 밀리초 단위의 지연 시간 AWS 리전

이러한 지표를 보는 방법에 대한 자세한 내용은 CloudWatch 데이터 보기를 참조하십시오.

재해 복구 및 Amazon DocumentDB 글로벌 클러스터

주제

- Amazon DocumentDB 글로벌 클러스터에 대한 관리형 장애 조치 수행
- Amazon DocumentDB 글로벌 클러스터에 대한 수동 장애 조치 수행
- Amazon DocumentDB 글로벌 클러스터에 대한 전환 수행
- 글로벌 클러스터 전환 또는 장애 조치 차단 해제

글로벌 클러스터를 활용하여 리전 장애와 같은 재해로부터 신속하게 복구할 수 있습니다. 재해 복구는 일반적으로 RTO 및 RPO 값을 사용하여 측정됩니다.

- (Recovery Time Objective(RTO) – 재해 발생 후 시스템이 정상 작동 상태로 돌아가는 데 걸리는 시간입니다. 즉 RTO는 가동 중지 시간을 측정합니다. 글로벌 클러스터의 경우 몇 분 안에 RTO를 수행합니다.
- Recovery Point Objective(RPO) – 손실될 수 있는 데이터의 양입니다(시간으로 측정). 글로벌 클러스터의 경우 일반적으로 RPO는 초 단위로 측정됩니다.
- 계획되지 않은 중단에서 복구하려면 글로벌 클러스터의 보조 클러스터 중 하나에 대해 리전 간 장애 조치를 수행할 수 있습니다. 글로벌 클러스터에 여러 개의 보조 리전이 있다면 기본으로 승격하려는 모든 보조 리전의 연결을 해제해야 합니다. 그런 다음 보조 영역 중 하나를 새 기본 영역 AWS 리전으로 승격합니다. 마지막으로 다른 보조 영역 각각에 새 클러스터를 생성하고 해당 클러스터를 글로벌 클러스터에 연결합니다.

Amazon DocumentDB 글로벌 클러스터에 대한 관리형 장애 조치 수행

이 접근 방식은 실제 리전별 재해 이벤트 또는 전체 서비스 수준 중단이 발생하는 경우 비즈니스 연속성을 유지하기 위해 개발되었습니다.

관리형 장애 조치 중에, Amazon DocumentDB 글로벌 클러스터의 기존 복제 토폴로지가 유지되는 동안 기본 클러스터는 선택한 보조 리전으로 장애 조치됩니다. 선택한 보조 클러스터에서는 읽기 전용 노드 중 하나가 전체 라이터 상태로 승격됩니다. 이 단계를 통해 클러스터는 기본 클러스터의 역할을 맡게 됩니다. 클러스터가 새 역할을 맡는 동안에는 데이터베이스를 일시적으로 사용할 수 없습니다. 이 보조 클러스터가 새 기본 클러스터가 되면 기존 기본 클러스터에서 선택한 보조 클러스터로 복제되지 않은 데이터는 누락될 수 있습니다. 이전 기본 볼륨은 스냅샷에 복제되지 않은 데이터가 보존되도록 새 기본 볼륨과 동기화하기 전에 스냅샷을 찍으려고 최선의 노력을 기울입니다.

> **Note**
>
> 기본 클러스터와 모든 보조 클러스터의 엔진 버전이 동일한 경우에만 Amazon DocumentDB 글로벌 클러스터에서 관리형 리전 간 클러스터 장애 조치를 수행할 수 있습니다. 엔진 버전이 호환되지 않는 경우 Amazon DocumentDB 글로벌 클러스터에 대한 수동 장애 조치 수행에 나온 단계에 따라 수동으로 장애 조치를 수행할 수 있습니다.
> 리전의 엔진 버전이 일치하지 않으면 장애 조치가 차단됩니다. 보류 중인 업그레이드가 있는지 확인하고 이를 적용하여 모든 리전의 엔진 버전이 일치하고 글로벌 클러스터 장애 조치가 차단 해제되었는지 확인하세요. 자세한 내용은 글로벌 클러스터 전환 또는 장애 조치 차단 해제 단원을 참조하십시오.

데이터 손실을 최소화하려면 이 기능을 사용하기 전에 다음을 수행하는 것이 좋습니다.

- 애플리케이션을 오프라인으로 전환하여 쓰기가 Amazon DocumentDB 글로벌 클러스터의 기본 클러스터로 전송되는 경우를 방지합니다.
- 모든 Amazon DocumentDB 보조 클러스터의 지연 시간을 확인합니다. 복제 지연이 가장 적게 소요되는 보조 리전을 선택하면 현재 장애가 발생한 기본 리전에서 데이터 손실을 최소화할 수 있습니다. Amazon CloudWatch에서 `GlobalClusterReplicationLag` 지표를 확인하여 글로벌 클러스터의 모든 Amazon DocumentDB 보조 클러스터에 대한 지연 시간을 확인합니다. 이 지표를 통해 보조 클러스터가 기본 DB 클러스터에 비해 얼마나 뒤처져 있는지(밀리초 단위) 알 수 있습니다.

  Amazon DocumentDB용 CloudWatch 지표에 대한 자세한 내용은 Amazon DocumentDB 지표 섹션을 참조하세요.

관리형 장애 조치 중에, 선택한 보조 DB 클러스터가 기본 클러스터처럼 새 역할로 승격됩니다. 하지만 기본 클러스터의 다양한 구성 옵션은 상속되지 않습니다. 구성이 일치하지 않으면 성능 문제, 워크로드 비호환성, 기타 비정상적인 동작이 발생할 수 있습니다. 이러한 문제를 방지하려면, 다음에 대해 Amazon DocumentDB 글로벌 클러스터 간의 차이점을 해결하는 것이 좋습니다.

- 필요한 경우 새 기본에 대한 Amazon DocumentDB 클러스터 파라미터 그룹 구성 — Amazon DocumentDB 글로벌 클러스터의 각 클러스터에 대해 Amazon DocumentDB 클러스터 파라미터 그룹을 개별적으로 구성할 수 있습니다. 즉, 보조 클러스터가 기본 클러스터 역할을 맡도록 승격되면 보조 클러스터의 파라미터 그룹이 기본 클러스터와 다르게 구성될 수 있습니다. 그런 경우, 승격된 보조 클러스터의 파라미터 그룹을 기본 클러스터의 설정에 맞게 수정합니다. 자세한 방법은 Amazon DocumentDB 클러스터 파라미터 그룹 수정을 참조하세요.
- Amazon CloudWatch Events 및 경보 등의 모니터링 도구 및 옵션 구성 — 승격된 클러스터를 글로벌 클러스터에 필요한 것과 동일한 로깅 기능, 경보 등으로 구성합니다. 파라미터 그룹과 마찬가지로, 이러한 기능에 대한 구성은 장애 조치 프로세스 중에 기본 클러스터에서 상속되지 않습니다. 복제 지연과 같은 일부 CloudWatch 지표는 보조 리전에서만 사용할 수 있습니다. 따라서 장애 조치로 인해 해당 지표를 보고 경보를 설정하는 방법이 달라짐에 따라 사전 정의된 대시보드를 변경해야 할 수도 있습니다. Amazon DocumentDB 클러스터 및 모니터링에 대한 자세한 내용은 Amazon DocumentDB 모니터링 섹션을 참조하세요.

일반적으로 선택한 보조 클러스터는 1분 이내에 기본 역할을 수임합니다. 새 기본 리전의 라이터 노드를 사용할 수 있게 되면 애플리케이션을 노드에 연결하고 워크로드를 재개할 수 있습니다. Amazon DocumentDB에서 새 기본 클러스터가 승격되고 난 후, 모든 추가 보조 리전 클러스터가 자동으로 재구축됩니다.

Amazon DocumentDB 글로벌 클러스터는 비동기식 복제를 사용하므로 보조 리전마다 복제 지연 시간이 다를 수 있습니다. Amazon DocumentDB는 이러한 보조 리전을 재구축하여 새로운 기본 리전 클러스터와 정확히 동일한 시점의 데이터를 확보하도록 지원합니다. 전체 재구축 작업에 소요되는 시간은 스토리지 볼륨의 크기와 리전 간 거리에 따라 몇 분에서 몇 시간이 걸릴 수 있습니다. 보조 리전 클러스터의 재구축이 새 기본 리전에서 완료되면 읽기 권한으로 액세스할 수 있습니다. 새 기본 라이터가 승격되어 사용 가능해지면 새 기본 리전의 클러스터에서 Amazon DocumentDB 글로벌 클러스터의 읽기 및 쓰기 작업을 처리할 수 있습니다.

글로벌 클러스터의 기존 토폴로지를 복원하기 위해 Amazon DocumentDB는 기존 기본 리전의 가용성을 모니터링합니다. 해당 리전이 정상이고 다시 사용 가능한 상태가 되면 Amazon DocumentDB는 자동으로 글로벌 클러스터에 보조 리전으로 다시 추가합니다. Amazon DocumentDB는 장애가 발생하는 시점에 기존 스토리지 볼륨의 스냅샷을 만들고자 시도한 후, 기존 기본 리전에 새 스토리지 볼륨을 생성합니다. 이렇게 하면 누락된 데이터를 복구하는 데 사용할 수 있습니다. 이 작업이 성공하면 Amazon

DocumentDB는 AWS Management Console의 스냅샷 섹션에 'rds:docdb-unplanned-global-failover-name-of-old-primary-DB-cluster-timestamp'라는 이름으로 이 스냅샷을 배치합니다. 또한 이 스냅샷은 `DescribeDBClusterSnapshots` API 작업에서 반환된 정보에 나열되어 있으니 해당 위치에서 확인할 수 있습니다.

> **Note**
>
> 기존 스토리지 볼륨의 스냅샷은 시스템 스냅샷으로, 기존 기본 클러스터에 구성된 백업 보존 기간이 적용됩니다. 보존 기간 이후에도 이 스냅샷을 보존하려면 스냅샷을 복사하여 수동 스냅샷으로 저장할 수 있습니다. 요금 등 스냅샷 복사에 대해 자세히 알아보려면 클러스터 스냅샷 복사 단원을 참조하세요.

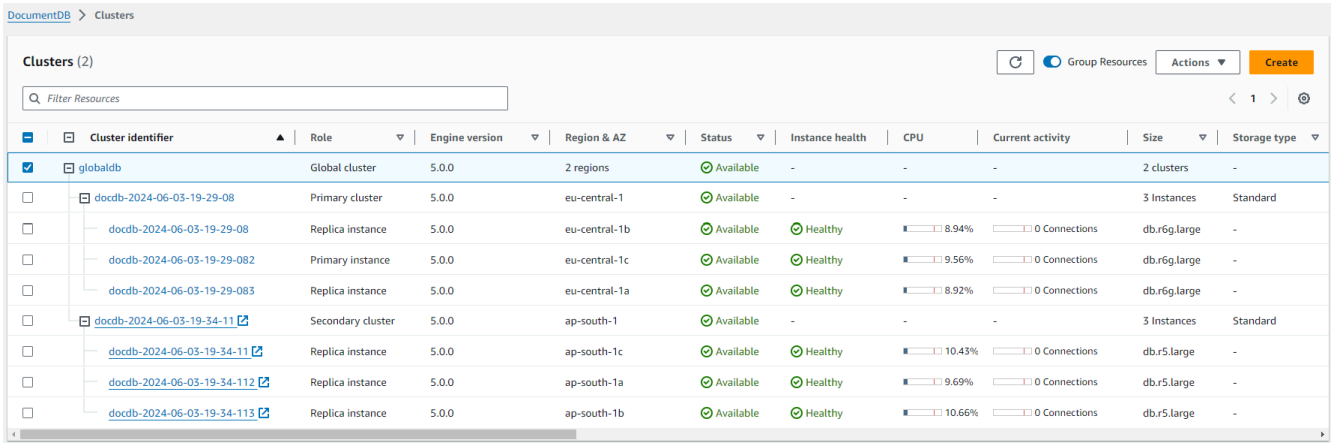
기존 토폴로지가 복원된 후 비즈니스 및 워크로드에 가장 적합한 시점에 전환 작업을 수행하여 글로벌 클러스터를 기존의 기본 리전으로 장애 복구할 수 있습니다. 이렇게 하려면 Amazon DocumentDB 글로벌 클러스터에 대한 전환 수행 단원의 절차를 따르세요.

AWS CLI, 또는 Amazon DocumentDB API를 사용하여 Amazon DocumentDB 글로벌 클러스터 AWS Management Console를 장애 조치할 수 있습니다.

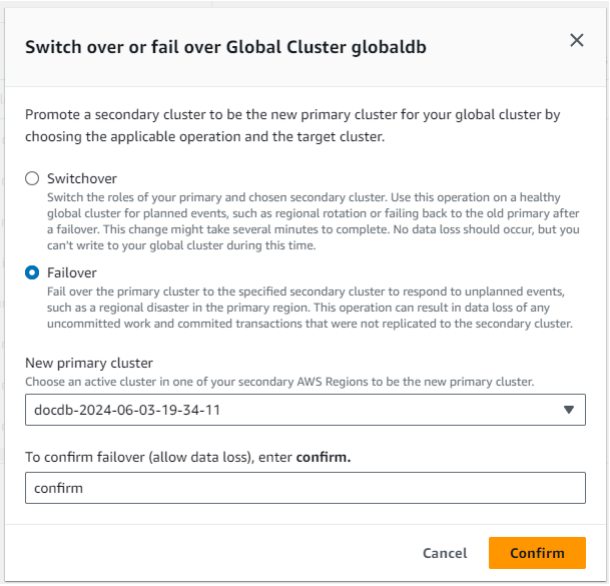
Using the AWS Management Console

Amazon DocumentDB 글로벌 클러스터에서 관리형 장애 조치 수행

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb://https://https://https://://https://://https://://httpsAmazon DocumentDB://https://://https://://https://https://https://https://://
2. 탐색 창에서 클러스터를 선택합니다.
3. 장애 조치하려는 Amazon DocumentDB 글로벌 클러스터를 찾아 선택합니다.

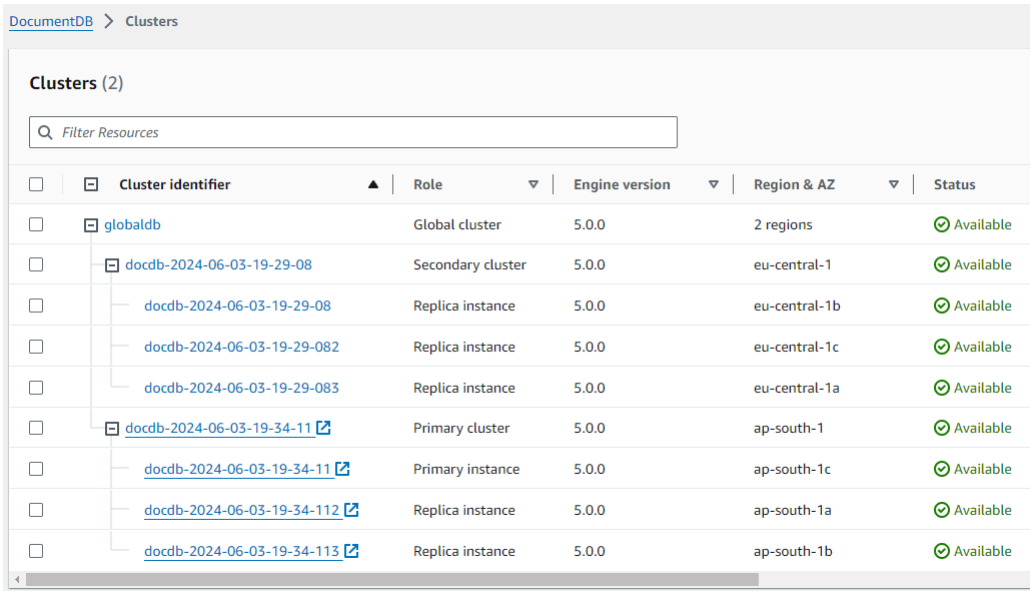


4. 작업 메뉴에서 전환 또는 장애 조치를 선택합니다.
5. 나타나는 대화 상자에서 장애 조치를 선택한 다음 새 기본 클러스터 필드 드롭다운 목록에서 보조 클러스터를 선택합니다.



6. 마지막 필드에 '확인'을 입력합니다. 그 다음 확인을 선택합니다.

   기본 클러스터의 상태가 '장애 조치 중'으로 변경됩니다. 이 조건은 약 1분이 소요됩니다. 이 시간 동안 새 기본 클러스터의 상태는 '수정 중...'으로 표시됩니다. 새 기본이 승격되면 '사용 가능'이 표시되고 읽기 및 쓰기 트랜잭션을 제공할 수 있습니다. 이전 기본을 포함한 보조 리전은 새 기본과 재동기화되는 동안 '재동기화 중...'을 표시합니다. 새 기본과 마찬가지로 상태가 '사용 가능'으로 변경된 후에만 트랜잭션을 제공할 수 있습니다.
7. 완료되면 원래 기본 클러스터가 보조 클러스터가 됩니다. 선택한 보조 클러스터가 기본 클러스터가 됩니다.



Using the AWS CLI

Amazon DocumentDB 글로벌 클러스터에서 관리형 장애 조치 수행

`failover-global-cluster` CLI 명령을 실행하여 Amazon DocumentDB 글로벌 클러스터를 장애 조치합니다. 명령을 사용하여 다음 옵션의 값을 전달합니다.

- `--region`
- `--global-cluster-identifier`
- `--target-db-cluster-identifier`
- `--allow-data-loss`

다음 예제에서는 각 *### ## ## ##*자를 클러스터 정보로 바꿉니다.

Linux, macOS, Unix의 경우:

```
aws docdb failover-global-cluster \
   --region region_of_selected_secondary \
   --global-cluster-identifier global_cluster_id \
   --target-db-cluster-identifier arn_of_secondary_to_promote \
   --allow-data-loss
```

Windows의 경우:

```
aws docdb failover-global-cluster ^
   --region region_of_selected_secondary ^
   --global-cluster-identifier global_cluster_id ^
   --target-db-cluster-identifier arn_of_secondary_to_promote ^
   --allow-data-loss
```

Amazon DocumentDB 글로벌 클러스터에 대한 수동 장애 조치 수행

한의 전체 클러스터를 사용할 수 없게 AWS 리전 되면 글로벌 클러스터의 다른 클러스터를 읽기/쓰기 기능으로 승격할 수 있습니다.

다른 AWS 리전 의 클러스터가 기본 클러스터로 더 적합한 경우 글로벌 클러스터 장애 조치 메커니즘을 수동으로 활성화할 수 있습니다. 예를 들어 보조 클러스터 중 하나의 용량을 늘린 후 이 클러스터를 기본 클러스터로 승격할 수 있습니다. 또는 간의 활동 균형이 변경될 AWS 리전 수 있으므로 기본 클러스터를 다른 로 전환하면 쓰기 작업의 지연 시간이 줄어들 AWS 리전 수 있습니다.

다음 절차에서는 Amazon DocumentDB 글로벌 클러스터의 보조 클러스터 중 하나를 승격하기 위해 수행할 작업을 설명합니다.

보조 클러스터를 승격하려면:

1. 중단 AWS 리전 시의 기본 클러스터에 대한 DML 문 및 기타 쓰기 작업 실행을 중지합니다.
2. 보조 클러스터에서 새 기본 클러스터로 사용할 클러스터를 식별 AWS 리전 합니다. 글로벌 클러스터 AWS 리전 에 보조 클러스터가 두 개(또는 그 이상) 있는 경우 지연 시간이 가장 짧은 보조 클러스터를 선택합니다.
3. 선택한 보조 클러스터를 글로벌 클러스터에서 분리합니다.

   글로벌 클러스터에서 보조 클러스터를 제거하면 기본 클러스터에서 이 보조 클러스터로의 복제가 즉시 중지되고 전체 읽기/쓰기 기능을 갖춘 독립 실행형 프로비저닝 클러스터로 승격됩니다. 정전이 발생한 리전의 주 클러스터와 관련된 다른 보조 클러스터는 여전히 사용할 수 있으며 애플리케이션의 호출을 수락할 수 있습니다. 클러스터는 리소스도 소비합니다. 글로벌 클러스터를 다시 생성하는 중이므로 분할 브레인 및 기타 문제를 방지하려면 다음 단계에서 새 글로벌 클러스터를 생성하기 전에 다른 보조 클러스터를 제거하십시오.

   분리 단계에 대한 자세한 내용은 Amazon DocumentDB 글로벌 클러스터에서 클러스터 분리 섹션을 참조하십시오.
4. 이 클러스터는 다음 단계에서 영역을 추가하기 시작하면 새 글로벌 클러스터의 기본 클러스터가 됩니다.

5. 클러스터에 AWS 리전 를 추가합니다. 이렇게 하면 기본 클러스터에서 보조 클러스터로의 복제 프로세스가 시작됩니다.
6. 필요에 AWS 리전 따라를 추가하여 애플리케이션을 지원하는 데 필요한 토폴로지를 다시 생성합니다. 글로벌 클러스터에 있는 클러스터 간의 데이터 불일치(분할 뇌 문제)를 방지하기 위해 이러한 변경 전, 변경 중 및 변경 후에 애플리케이션 쓰기가 올바른 클러스터로 전송되는지 확인합니다.
7. 운영 중단이 해결되고 원본 AWS 리전 을 다시 기본 클러스터로 할당할 준비가 되면 동일한 단계를 반대로 수행합니다.
8. 글로벌 데이터베이스에서 보조 클러스터 중 하나를 제거합니다. 이렇게 하면 읽기/쓰기 트래픽을 처리할 수 있습니다.
9. 모든 쓰기 트래픽을 원래 AWS 리전의 기본 클러스터로 리디렉션합니다.
10. AWS 리전 를 추가하여 이전 AWS 리전 과 동일한에 하나 이상의 보조 클러스터를 설정합니다.

Amazon DocumentDB 글로벌 클러스터는 AWS SDKs를 사용하여 관리할 수 있으므로 재해 복구 및 비즈니스 연속성 계획 사용 사례에 대한 글로벌 클러스터 장애 조치 프로세스를 자동화하는 솔루션을 생성할 수 있습니다. 이러한 솔루션 중 하나는 Apache 2.0 라이센스에 따라 고객이 사용할 수 있도록 제공되며 여기에 있는 툴 저장소에서 액세스할 수 있습니다. 이 솔루션은 엔드포인트 관리를 위해 Amazon Route 53를 활용하고 적절한 이벤트를 기반으로 트리거할 수 있는 AWS Lambda 함수를 제공합니다.

Amazon DocumentDB 글로벌 클러스터에 대한 전환 수행

전환을 사용하면 기본 클러스터의 리전을 정기적으로 변경할 수 있습니다. 이 접근 방식은 운영 유지 관리 및 기타 계획된 운영 절차 등 제어된 시나리오를 대상으로 개발되었습니다.

전환은 일반적으로 3가지의 경우에 사용됩니다.

- 특정 산업에서 필요로 하는 '리전별 순환' 요구 사항을 살펴봅니다. 예를 들어 금융 서비스 규정에 따라 재해 복구 절차가 정기적으로 실행되도록 보장하려면 Tier-0 시스템을 몇 개월 동안 다른 리전으로 전환해야 할 수 있습니다.
- 여러 리전의 'follow-the-sun' 애플리케이션을 살펴봅니다. 각기 다른 시간대 전반에서 업무 시간을 기준으로 여러 리전별로 지연 시간이 짧은 쓰기 기능을 제공하고자 하는 기업을 예로 들 수 있습니다.
- 데이터 손실 제로의 방법이며, 장애 조치 후 기존의 기본 리전으로 페일백하는 데 유용합니다.

> **Note**
>
> 전환은 정상 Amazon DocumentDB 글로벌 클러스터에서 사용하도록 설계되었습니다. 예상치 못한 중단이 발생한 상태에서 복구하려면 Amazon DocumentDB 글로벌 클러스터에 대한 수동 장애 조치 수행 단원에 나온 적절한 절차를 따르세요.
> 전환을 수행하려면 모든 보조 리전이 기본 리전과 정확히 동일한 엔진 버전을 실행해야 합니다. 리전의 엔진 버전이 일치하지 않으면 전환이 차단됩니다. 보류 중인 업그레이드가 있는지 확인하고 이를 적용하여 모든 리전의 엔진 버전이 일치하고 글로벌 클러스터 전환이 차단 해제되었는지 확인하세요. 자세한 내용은 글로벌 클러스터 전환 또는 장애 조치 차단 해제 단원을 참조하십시오.

전환 중에, Amazon DocumentDB에서는 글로벌 클러스터의 기존 복제 토폴로지가 유지되는 동안 기본 클러스터를 선택한 보조 리전으로 전환합니다. Amazon DocumentDB는 모든 보조 리전 클러스터가 기본 리전 클러스터와 완전히 동기화될 때까지 기다린 다음, 전환 프로세스를 시작합니다. 그러면 기본 리전의 DB 클러스터가 읽기 전용 상태가 되고, 선택한 보조 클러스터는 읽기 전용 노드 중 하나를 전체 라이터 상태로 승격시킵니다. 이 노드를 라이터로 승격시키면 보조 클러스터가 기본 클러스터의 역할을 맡을 수 있습니다. 프로세스 시작 시 모든 보조 클러스터가 기본 클러스터와 동기화되었으므로, 새로운 기본 클러스터는 데이터 손실 없이 Amazon DocumentDB 글로벌 클러스터에 대한 작업을 계속합니다. 기본 클러스터와 선택한 보조 클러스터가 새 역할을 맡으므로 데이터베이스를 잠시 사용할 수 없습니다.

애플리케이션 가용성을 최적화하려면 이 기능을 사용하기 전에 다음 작업을 수행하는 것이 좋습니다.

- 사용량이 적은 시간이나 기본 클러스터에 대한 쓰기가 최소인 시간에 이 작업을 수행합니다.
- 애플리케이션을 오프라인으로 전환하여 쓰기가 Amazon DocumentDB 글로벌 클러스터의 기본 클러스터로 전송되는 경우를 방지합니다.
- Amazon CloudWatch에서 `GlobalClusterReplicationLag` 지표를 확인하여 글로벌 클러스터의 모든 Amazon DocumentDB 보조 클러스터에 대한 지연 시간을 확인합니다. 이 지표는 보조 DB 클러스터가 기본 DB 클러스터에 비해 얼마나 뒤처져 있는지(밀리초 단위)를 보여 줍니다. 이 값은 Amazon DocumentDB가 전환을 완료하는 데 걸리는 시간에 직접적으로 비례합니다. 따라서 지연 값이 클수록 전환 시간이 더 오래 걸립니다.

  Amazon DocumentDB용 CloudWatch 지표에 대한 자세한 내용은 Amazon DocumentDB 지표 섹션을 참조하세요.

전환 중에, 선택한 보조 DB 클러스터가 기본 DB 클러스터처럼 새 역할로 승격됩니다. 하지만 기본 DB 클러스터의 다양한 구성 옵션은 상속되지 않습니다. 구성이 일치하지 않으면 성능 문제, 워크로드 비호환성, 기타 비정상적인 동작이 발생할 수 있습니다. 이러한 문제를 방지하려면, 다음에 대해 Amazon DocumentDB 글로벌 클러스터 간의 차이점을 해결하는 것이 좋습니다.

- 필요한 경우 새 기본에 대한 Amazon DocumentDB 클러스터 파라미터 그룹 구성 — Amazon DocumentDB 글로벌 클러스터의 각 클러스터에 대해 Amazon DocumentDB 클러스터 파라미터 그룹을 개별적으로 구성할 수 있습니다. 즉, 보조 DB 클러스터가 기본 클러스터 역할을 맡도록 승격되면 보조 클러스터의 파라미터 그룹이 기본 클러스터와 다르게 구성될 수 있습니다. 그런 경우, 승격된 보조 DB 클러스터의 파라미터 그룹을 기본 클러스터의 설정에 맞게 수정합니다. 자세한 방법은 Amazon DocumentDB 클러스터 파라미터 그룹 관리 단원을 참조하십시오.
- Amazon CloudWatch Events 및 경보 등의 모니터링 도구 및 옵션 구성 — 승격된 클러스터를 글로벌 클러스터에 필요한 것과 동일한 로깅 기능, 경보 등으로 구성합니다. 파라미터 그룹과 마찬가지로, 이러한 기능에 대한 구성은 전환 프로세스 중에 기본 클러스터에서 상속되지 않습니다. 복제 지연과 같은 일부 CloudWatch 지표는 기본 리전에서만 사용할 수 있습니다. 따라서 전환으로 인해 해당 지표를 보고 경보를 설정하는 방법이 달라짐에 따라 사전 정의된 대시보드를 변경해야 할 수도 있습니다. 자세한 내용은 Amazon DocumentDB 모니터링 단원을 참조하십시오.

> **Note**
>
> 일반적으로 역할 전환에는 최대 몇 분이 걸릴 수 있습니다.

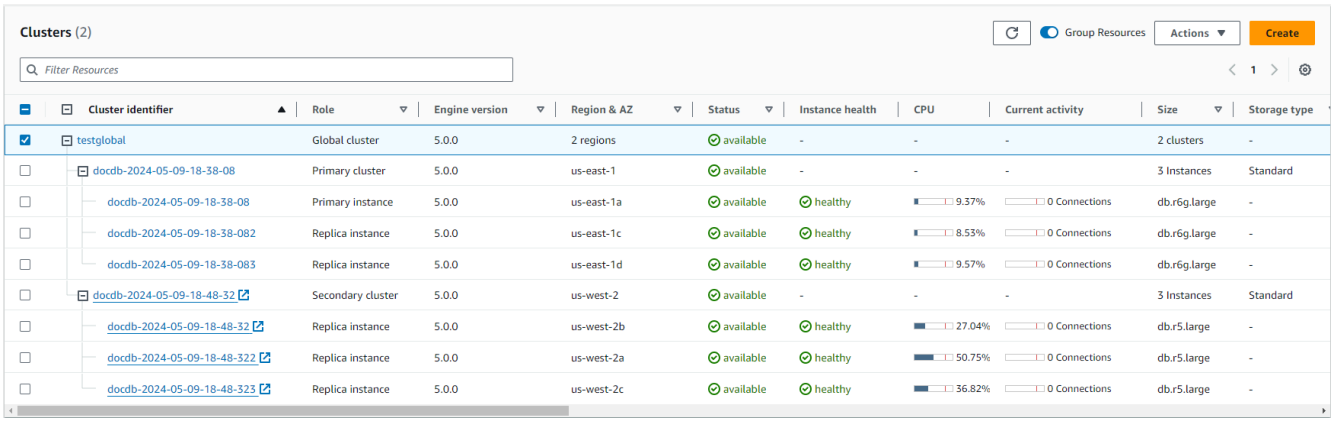
전환 프로세스가 완료되면 승격된 Amazon DocumentDB 클러스터에서 글로벌 클러스터에 대한 쓰기 작업을 처리할 수 있습니다.

AWS Management Console 또는 AWS CLI를 사용하여 Amazon DocumentDB 글로벌 클러스터를 전환할 수 있습니다.

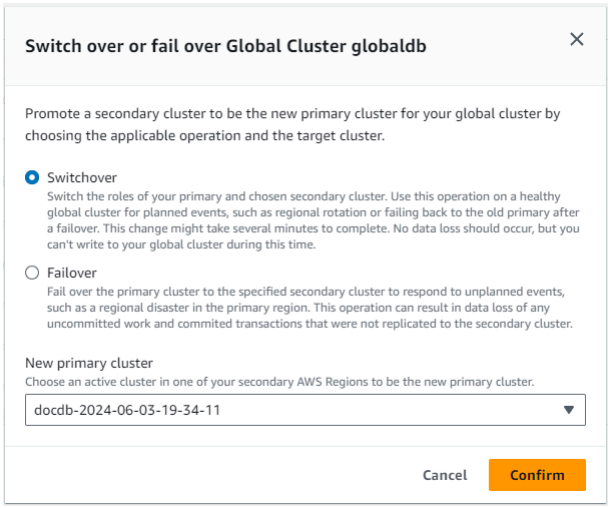
Using the AWS Management Console

Amazon DocumentDB 글로벌 클러스터에서 전환 수행

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb://https://https://https://://https://://https://://https://://https://AmazonDocumentDB://https://://https://://https://https://https://https://://
2. 탐색 창에서 클러스터를 선택합니다.
3. 전환하려는 Amazon DocumentDB 글로벌 클러스터를 찾아 선택합니다.

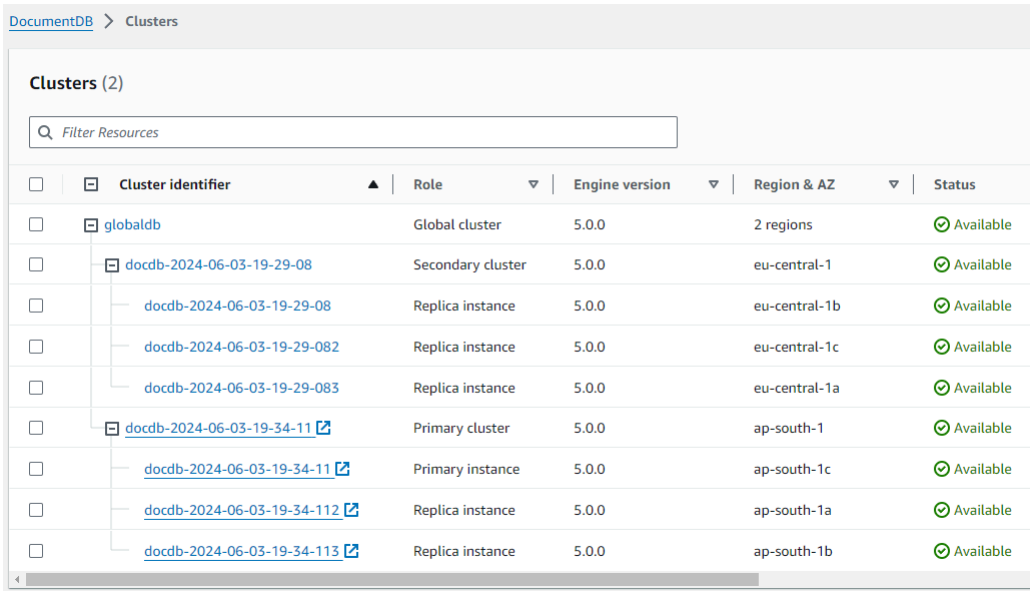


4. 작업 메뉴에서 전환 또는 장애 조치를 선택합니다.
5. 나타나는 대화 상자에서 전환을 선택한 다음 새 기본 클러스터 필드 드롭다운 목록에서 보조 클러스터를 선택합니다.



6. 확인을 선택합니다.

   기본 클러스터의 상태가 '전환'으로 변경됩니다. 이 조건은 약 3분이 소요됩니다. 이 시간 동안 모든 리전 클러스터의 상태는 '수정 중...'으로 표시됩니다. 리전이 동기화되고 새 프라이머리가 승격되면 모든 상태 필드에 대해 '사용 가능'이 표시되고 트랜잭션을 제공할 수 있습니다.
7. 완료되면 원래 기본 클러스터가 보조 클러스터가 됩니다. 선택한 보조 클러스터가 기본 클러스터가 됩니다.



Using the AWS CLI

Amazon DocumentDB 글로벌 클러스터에서 전환 수행

`switchover-global-cluster` CLI 명령을 실행하여 Amazon DocumentDB 글로벌 클러스터로 전환합니다. 명령을 사용하여 다음 옵션의 값을 전달합니다.

- `--region`
- `--global-cluster-identifier`
- `--target-db-cluster-identifier`

다음 예제에서는 각 *### ## ## ##*자를 클러스터 정보로 바꿉니다.

Linux, macOS, Unix의 경우:

```
aws docdb switchover-global-cluster \
   --region region_of_primary \
   --global-cluster-identifier global_cluster_id \
   --target-db-cluster-identifier arn_of_secondary_to_promote
```

Windows의 경우:

```
aws docdb switchover-global-cluster ^
   --region region_of_primary ^
```

```
    --global-cluster-identifier global_cluster_id ^
    --target-db-cluster-identifier arn_of_secondary_to_promote
```

글로벌 클러스터 전환 또는 장애 조치 차단 해제

글로벌 클러스터의 모든 리전 클러스터가 동일한 엔진 버전에 있지 않은 경우 글로벌 클러스터 전환 및 장애 조치가 차단됩니다. 버전이 일치하지 않는 경우 전환 또는 장애 조치를 호출할 때 응답으로이 오류가 표시될 수 있습니다. 지정된 대상 DB 클러스터가 소스 DB 클러스터와 다른 패치 수준의 엔진 버전을 실행하고 있습니다. 글로벌 클러스터를 정상 상태로 유지하기 위해 최신 업데이트를 실행하려면 최신 엔진 버전을 정기적으로 적용하는 것이 좋습니다.

이 오류를 해결하려면 먼저 모든 보조 리전을 업데이트한 다음 보류 중인 유지 관리 작업 항목을 적용하여 기본 리전을 동일한 엔진 버전으로 업데이트하십시오. 보류 중인 유지 관리 작업 항목을 보고 필요한 변경 사항을 적용하여 문제를 해결하려면 다음 탭 중 하나의 지침을 수행합니다.

Using the AWS Management Console

글로벌 클러스터 전환 또는 장애 조치를 차단 해제하려면 클러스터에 대해 보류 중인 유지 관리 작업이 있는지 확인하고 적용해야 합니다. 다음 단계에 따라 유지 관리 작업을 보고 적용합니다.

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb:// https://https://httpsAmazon DocumentDBhttps://https://://https://https://://https://://https://:// https://://://://://https://https
2. 탐색 창에서 클러스터를 선택합니다.
3. 클러스터 테이블의 클러스터 식별자 열에서 글로벌 클러스터를 찾습니다. 글로벌 클러스터에서 지정된 글로벌 클러스터의 각 보조 클러스터와 기본 클러스터를 기록하고 각각에 대해 다음 단계를 수행합니다.
4. 각 보조 클러스터에 대해:
    a. 클러스터에 업데이트를 사용할 수 있는 경우 유지 관리 열에 사용 가능, 필수 또는 다음 창으로 표시됩니다.
    b. 작업을 수행하려면 클러스터를 선택하여 세부 정보를 표시한 다음 유지 관리 및 백업을 선택합니다. 그러면 대기 중인 유지 관리 항목이 표시됩니다.
    c. 설명에서 "새 유지 관리 업데이트를 사용할 수 있습니다"라고 표시되면 해당 업데이트를 선택한 다음 지금 적용을 선택합니다.
5. 기본 클러스터의 경우:

a. 클러스터에 업데이트를 사용할 수 있는 경우 유지 관리 열에 사용 가능, 필수 또는 다음 창으로 표시됩니다.

b. 작업을 수행하려면 클러스터를 선택하여 세부 정보를 표시한 다음 유지 관리 및 백업을 선택합니다. 그러면 대기 중인 유지 관리 항목이 표시됩니다.

c. 설명에서 "새 유지 관리 업데이트를 사용할 수 있습니다"라고 표시되면 해당 업데이트를 선택한 다음 지금 적용을 선택합니다.

Using the AWS CLI

글로벌 클러스터 전환 또는 장애 조치를 차단 해제하려면 클러스터에 대해 보류 중인 유지 관리 작업이 있는지 확인하고 적용해야 합니다. 다음 단계에 따라 보조 클러스터에서 먼저 유지 관리 작업을 보고 적용한 다음 글로벌 클러스터의 기본 클러스터에서 유지 관리 작업을 보고 적용합니다.

1. 각 보조 리전의 리전 클러스터에서 먼저 다음을 실행한 다음 기본 리전 리전 클러스터에서 다음을 실행합니다.
2. `--resource-identifier` 옵션과 함께 describe-pending-maintenance-actions CLI 명령을 실행하여 Amazon DocumentDB 리전 클러스터에 사용할 수 있는 유지 관리 작업이 있는지 확인합니다.

   다음 예제에서는 각 *### ## ## ##*자를 클러스터 정보로 바꿉니다.

   Linux, macOS, Unix의 경우:

   ```
   aws docdb describe-pending-maintenance-action \
      --resource-identifier arn:aws:rds:us-
   east-1:001234567890:cluster:docdb-2025-03-27-19-21-15
   ```

   Windows의 경우:

   ```
   aws docdb describe-pending-maintenance-action ^
      --resource-identifier arn:aws:rds:us-
   east-1:001234567890:cluster:docdb-2025-03-27-19-21-15
   ```

   결과는 다음과 비슷합니다.

   ```
   {
       "PendingMaintenanceActions": [
           {
   ```

```
            "ResourceIdentifier": "arn:aws:rds:us-
east-1:001234567890:cluster:docdb-2025-03-27-19-21-15",
            "PendingMaintenanceActionDetails": [
                {
                    "Action": "system-update",
                    "CurrentApplyDate": "2025-04-11T03:01:00Z",
                    "Description": "db-version-upgrade",
                    "ForcedApplyDate": "2025-06-18T03:01:00Z",
                    "AutoAppliedAfterDate": "2025-05-11T03:01:00Z"
                    "OptInStatus": "pending"
                }
            ]
        }
    ]
}
```

3. 유지 관리 작업이 필요한 경우 다음 옵션을 사용하여 apply-pending-maintenance-action CLI 명령을 실행합니다.

   - `--resource-identifier`
   - `--apply-action`
   - `--opt-in-type`
   - `--region`

   다음 예제에서는 각 *### ## ## ##*자를 클러스터 정보로 바꿉니다.

   Linux, macOS, Unix의 경우:

```
aws docdb apply-pending-maintenance-action \
   --resource-identifier arn:aws:rds:us-
east-1:001234567890:cluster:docdb-2025-03-27-19-21-15 \
   --apply-action system-update \
   --opt-in-type immediate \
   --region us-east-1
```

   Windows의 경우:

```
aws docdb apply-pending-maintenance-action ^
   --resource-identifier arn:aws:rds:us-
east-1:001234567890:cluster:docdb-2025-03-27-19-21-15 ^
```

```
    --apply-action system-update ^
    --opt-in-type immediate ^
    --region us-east-1
```

4. 유지 관리 작업이 완료되면 describe-pending-maintenance-actions 명령을 다시 실행하여 클러스터에 보류 중인 다른 작업이 없는지 확인합니다.

   원하는 결과는 다음과 같습니다.

```
{
    "PendingMaintenanceActions": []
}
```

Using the Amazon DocumentDB API

글로벌 클러스터 전환 또는 장애 조치를 차단 해제하려면 클러스터에 대해 보류 중인 유지 관리 작업이 있는지 확인하고 적용해야 합니다. 다음 APIs를 사용하여 유지 관리 작업을 보고 적용합니다.

1. 각 보조 리전의 리전 클러스터에서 먼저 다음을 실행한 다음 기본 리전 리전 클러스터에서 다음을 실행합니다.
2. PendingMaintenanceAction API를 호출하여 Amazon DocumentDB 글로벌 클러스터에 사용할 수 있는 유지 관리 작업이 있는지 확인합니다.
3. ApplyPendingMaintenanceAction API를 호출하여 변경 사항을 적용합니다.

Amazon DocumentDB 클러스터 관리

Amazon DocumentDB 클러스터를 관리하려면 적절한 Amazon DocumentDB 컨트롤 플레인 권한을 가진 IAM 정책이 있어야 합니다. 이러한 권한은 클러스터와 인스턴스를 생성, 수정 및 삭제하도록 허용합니다. `AmazonDocDBFullAccess` 정책은 Amazon DocumentDB 클러스터를 관리하는 데 필요한 모든 권한을 제공합니다.

다음 주제에서는 Amazon DocumentDB 클러스터로 작업할 때 클러스터 생성, 삭제, 수정, 연결, 보기 등과 같은 다양한 작업을 수행하는 방법을 보여줍니다.

주제

- 클러스터 이해
- Amazon DocumentDB 클러스터 설정
- Amazon DocumentDB 클러스터 스토리지 구성

- 클러스터 상태 확인
- Amazon DocumentDB 클러스터 수명 주기
- Amazon DocumentDB 클러스터 규모 조정
- Amazon DocumentDB 클러스터에 대한 볼륨 복제
- Amazon DocumentDB 클러스터 내결함성에 대한 이해

클러스터 이해

Amazon DocumentDB에서는 컴퓨팅과 스토리지를 구분하고, 데이터 복제 및 백업을 클러스터 볼륨으로 오프로드합니다. 클러스터 볼륨은 가용 영역 3곳에 6가지 방식으로 데이터를 복제하는 내구성이 뛰어나고 안정적인 고가용성 스토리지 계층을 제공합니다. 복제본은 높은 데이터 가용성을 확보하고 읽기 확장이 가능합니다. 각 클러스터를 최대 15개의 복제본으로 스케일 업할 수 있습니다.

명사	설명	API 작업(동사)

클러스터	하나 이상의 인스턴스와 이 인스턴스의 데이터를 관리하는 클러스터 스토리지 볼륨으로 구성됩니다.	`create-db-cluster`
		`delete-db-cluster`
		`describe-db-clusters`
		`modify-db-cluster`

Instance	클러스터 스토리지 볼륨에 대한 데이터 읽기 및 쓰기는 인스턴스를 통해 수행됩니다. 지정된 클러스터에는 기본 인스턴스와 복제본의 2가지 인스턴스가 있습니다. 클러스터는 항상 기본 인스턴스 하나와 0~15개의 복제본을 가질 수 있습니다.	`create-db-instance`
		`delete-db-instance`
		`describe-db-instances`
		`modify-db-instance`
		`describe-orderable-db-instance-options`
		`reboot-db-instance`

클러스터 볼륨	3개의 가용 영역을 모두 아우르는 가상 데이터베이스 스토리	N/A

명사	설명	API 작업(동사)
	지 볼륨으로서, 각 가용 영역에는 클러스터 데이터의 사본이 2개 있습니다.	
기본 인스턴스	읽기 및 쓰기 작업을 모두 지원하고, 클러스터 볼륨의 모든 데이터 수정을 실행합니다. 클러스터마다 기본 인스턴스가 하나씩 있습니다.	N/A
복제본 인스턴스	읽기 연산만 지원합니다. 각 Amazon DocumentDB 클러스터는 기본 인스턴스에 더해 최대 15개의 복제 인스턴스를 포함할 수 있습니다. 복제본이 여러 개 있으면 읽기 워크로드가 분산됩니다. 복제본을 별도의 가용 영역에 두어 데이터베이스 가용성을 높일 수도 있습니다.	N/A
클러스터 엔드포인트	클러스터의 현재 기본 인스턴스에 연결되는 Amazon DocumentDB 클러스터 엔드포인트입니다. 각 Amazon DocumentDB 클러스터에는 클러스터 엔드포인트 하나와 기본 인스턴스 하나가 있습니다.	N/A

명사	설명	API 작업(동사)
리더 엔드포인트	클러스터에서 사용할 수 있는 복제본 중 하나에 연결되는 Amazon DocumentDB 클러스터 엔드포인트입니다. 각 Amazon DocumentDB 클러스터에는 리더 엔드포인트가 1개씩 있습니다. 복제본이 하나 이상이면 리더 엔드포인트는 각 연결 요청을 Amazon DocumentDB 복제본 중 하나로 전달합니다.	N/A
인스턴스 엔드포인트	특정 인스턴스에 연결되는 Amazon DocumentDB 클러스터의 인스턴스 엔드포인트입니다. 클러스터의 인스턴스에는 인스턴스 유형에 상관없이 각각 고유한 인스턴스 엔드포인트가 있습니다.	N/A

Amazon DocumentDB 클러스터 설정

클러스터를 생성 또는 수정할 때는 생성 후 수정할 수 없는 파라미터와 수정 가능한 파라미터를 이해하는 것이 중요합니다. 다음 표에는 한 클러스터에 적용되는 모든 설정 또는 파라미터가 나와 있습니다. 표에 나와 있듯이 일부 파라미터는 수정할 수 있고 다른 파라미터는 수정할 수 없습니다.

> **Note**
>
> 이러한 설정을 Amazon DocumentDB 클러스터 파라미터 그룹 및 해당 파라미터와 혼동해서는 안 됩니다. 클러스터 파라미터 그룹에 대한 자세한 내용은 Amazon DocumentDB 클러스터 파라미터 그룹 관리 섹션을 참조하세요.

파라미터	수정 가능	참고
DBClusterIdentifier	예	명명 제약 조건: • 길이는 [1~63] 글자, 숫자 또는 하이픈입니다. • 첫 번째 문자는 글자이어야 합니다. • 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다. • 리전별로 Amazon RDS, Amazon Neptune 및 Amazon DocumentDB의 모든 클러스터 AWS 계정에 대해 고유해야 합니다.
Engine	아니요	docdb여야 합니다.
BackupRetentionPeriod	예	1~35일 사이여야 합니다.
DBClusterParameterGroupName	예	명명 제약 조건: • [1-255]자 길이의 영숫자 문자입니다. • 첫 번째 문자는 글자이어야 합니다. • 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다.
DBSubnetGroupName	아니요	클러스터가 생성된 이후에는 클러스터의 서브넷을 수정할 수 없습니다.
EngineVersion	아니요	값은 `5.0.0` (기본값)`4.0.0`, 또는 `3.6.0`일 수 있습니다.
KmsKeyId	아니요	클러스터를 암호화하기로 선택한 경우 클러스터를 암호화하는 데 사용한 AWS KMS 키를 변경할 수 없습니다.
MasterUsername	아니요	클러스터가 생성된 이후에는 `MasterUsername` 을 수정할 수 없습니다. 명명 제약 조건:

파라미터	수정 가능	참고
		• 길이는 [1-63]자의 영숫자 문자입니다. • 첫 번째 문자는 글자이어야 합니다. • 데이터베이스 엔진에 포함된 단어는 사용할 수 없습니다.

MasterUserPassword	예	제약 조건:
		• 길이는 [8-100]자의 인쇄 가능한 ASCII 문자입니다.
		• 다음을 제외한 인쇄 가능한 ASCII 문자를 사용할 수 있습니다.
		  • **/**(슬래시)
		  • **"**(큰 따옴표)
		  • **@**(at 기호)

Port	예	포트 번호는 클러스터의 모든 인스턴스에 적용됩니다.
PreferredBackupWindow	예	
PreferredMaintenanceWindow	예	
StorageEncrypted	아니요	클러스터를 암호화하도록 선택한 경우 암호를 해제할 수 없습니다.

파라미터	수정 가능	참고
StorageType	예	DB 클러스터의 스토리지 유형: 표준 (standard) 또는 I/O 최적화(iopt1). 기본값: standard 이 파라미터는 CreateDBCluster 및 ModifyDBCluster 항목을 사용하여 구성할 수 있습니다. 자세한 내용은 Amazon DocumentDB 클러스터 스토리지 구성 단원을 참조하십시오.
Tags	예	
VpcSecurityGroupIds	아니요	클러스터를 만든 후에는 클러스터가 상주하는 VPC를 수정할 수 없습니다.

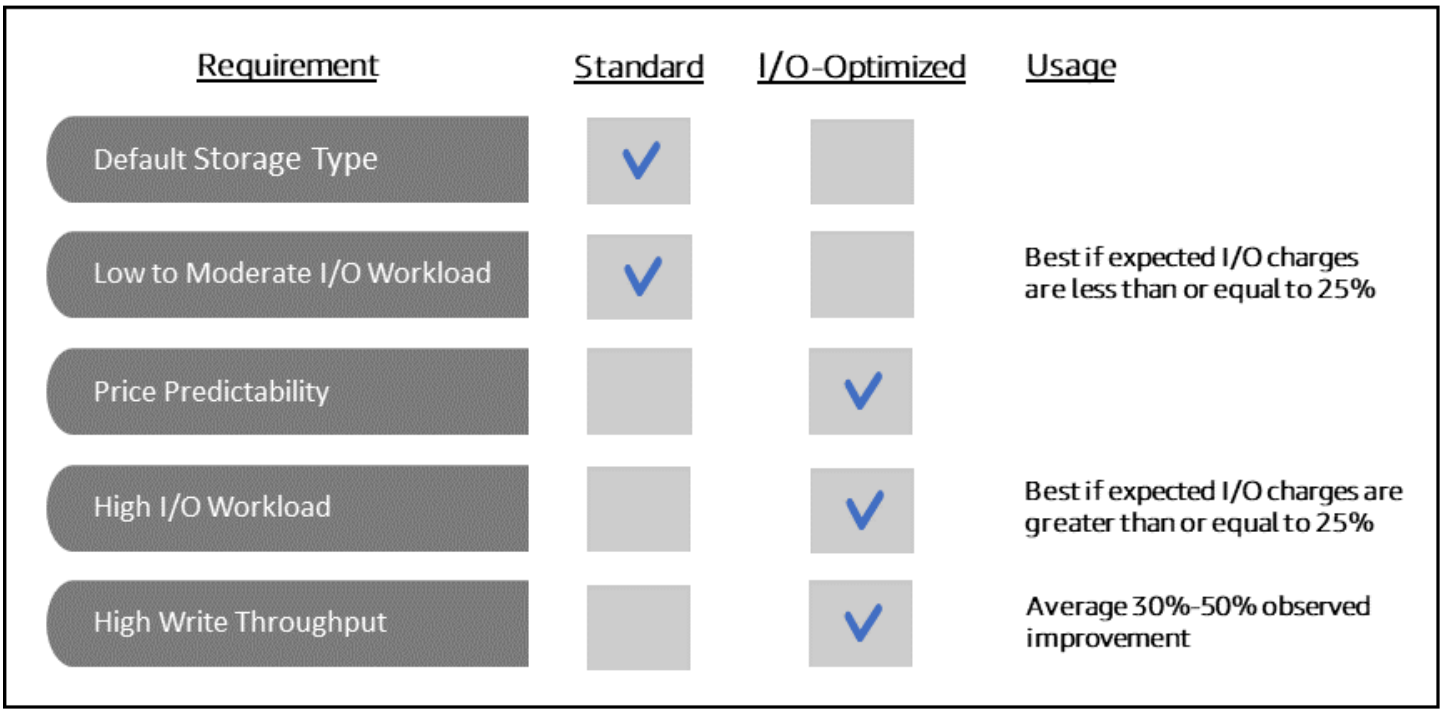
Amazon DocumentDB 클러스터 스토리지 구성

Amazon DocumentDB 5.0부터 인스턴스 기반 클러스터는 두 가지 스토리지 구성 유형을 지원합니다.

- Amazon DocumentDB 표준 스토리지: I/O 소비가 낮거나 중간 수준인 고객을 위해 설계되었습니다. I/O 비용이 총 Amazon DocumentDB 클러스터의 25% 미만일 것으로 예상되는 경우 이러한 선택이 적합할 수 있습니다. Amazon DocumentDB 표준 스토리지 구성을 사용하면 인스턴스 및 스토리지 요금 외에도 요청당 지불 I/O 기준으로 요금이 청구됩니다. 즉, 사용량에 따라 주기별로 청구가 달라질 수 있습니다. 이 구성은 애플리케이션의 변화하는 I/O 수요에 맞게 조정됩니다.
- Amazon DocumentDB I/O 최적화 스토리지: 가격 예측 가능성을 우선시하거나 I/O 집약적 애플리케이션을 사용하는 고객을 위해 설계되었습니다. I/O 최적화 구성은 I/O 집약적 워크로드를 사용하는 고객에게 향상된 성능, 향상된 처리량 및 감소된 지연 시간을 제공합니다. I/O 비용이 총 Amazon DocumentDB 클러스터 비용의 25%를 초과할 것으로 예상되는 경우 이 옵션은 향상된 가격 성능을 제공합니다. Amazon DocumentDB I/O 최적화 스토리지 구성을 사용하면 I/O 작업에 따른 요금이 부과되지 않으므로 각 결제 주기마다 비용을 예측 가능하도록 보장해 줍니다. 구성은 성능을 개선하면서 비용을 안정화합니다.

30일마다 한 번씩 기존 데이터베이스 클러스터를 Amazon DocumentDB I/O 최적화 스토리지로 전환할 수 있습니다. 언제든지 Amazon DocumentDB 표준 스토리지로 다시 전환할 수 있습니다. 스토리지 구성을 I/O 최적화로 수정하는 다음 날짜는 `describe-db-clusters` 명령을 사용하거나 클러스터 구성 페이지의 AWS Management Console 를 AWS CLI 통해 추적할 수 있습니다.

Amazon DocumentDB I/O 최적화 구성을 포함한 새 데이터베이스 클러스터를 생성하거나 AWS Management Console에서 몇 번의 클릭, AWS Command Line Interface (AWS CLI)에서 단일 파라미터 변경 또는 AWS SDK를 통해 기존 데이터베이스 클러스터를 변환할 수 있습니다. 스토리지 구성을 수정하는 동안 또는 수정한 후에 필요한 인스턴스의 가동 중지 시간이나 재부팅은 없습니다.

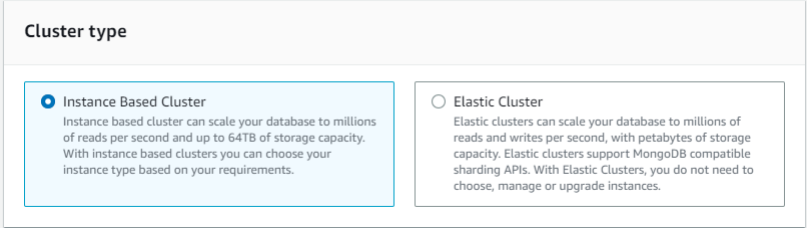


I/O 최적화 클러스터 생성

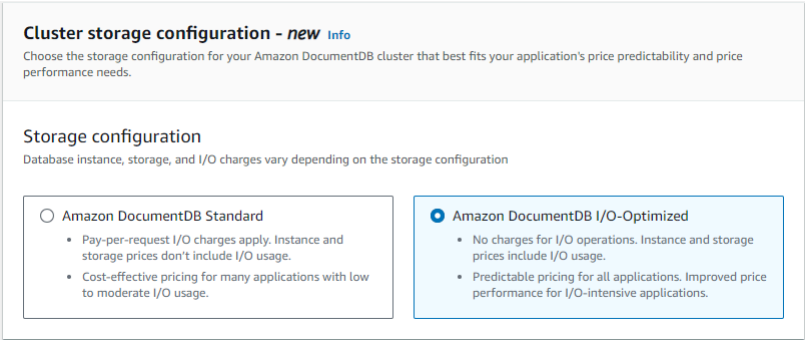
Using the AWS Management Console

AWS Management Console다음을 사용하여 I/O 최적화 클러스터를 생성하거나 수정하려면

1. Amazon DocumentDB 관리 콘솔의 클러스터 에서 클러스터 생성 또는 선택을 선택하고 작업을 선택한 다음 수정을 선택합니다.
2. 새 클러스터를 생성하는 경우 클러스터 유형 섹션에서 인스턴스 기반 클러스터를 선택해야 합니다(기본 옵션).



3. 구성 섹션의 클러스터 스토리지 구성에서 Amazon DocumentDB I/O 최적화를 선택합니다.



4. 클러스터 생성 또는 수정을 완료하고 클러스터 생성 또는 클러스터 수정을 선택합니다.

   전체 클러스터 생성 프로세스는 를 사용하여 클러스터 및 기본 인스턴스 생성 AWS Management Console 항목을 참조하세요.

   전체 클러스터 수정 프로세스는 Amazon DocumentDB 클러스터 수정 항목을 참조하세요.

Using the AWS CLI

를 사용하여 I/O 최적화 클러스터를 생성하려면 AWS CLI:

다음 예에서는 자신의 정보로 각각의 *### ## ## ###*를 바꿉니다.

Linux, macOS, Unix의 경우:

```
aws docdb create-db-cluster \
      --db-cluster-identifier sample-cluster \
      --engine docdb \
      --engine-version 5.0.0 \
      --storage-type iopt1 \
      --deletion-protection \
      --master-username username \
      --master-user-password password
```

Windows의 경우:

```
aws docdb create-db-cluster ^
      --db-cluster-identifier sample-cluster ^
      --engine docdb ^
      --engine-version 5.0.0 ^
      --storage-type iopt1 ^
      --deletion-protection ^
      --master-username username ^
      --master-user-password password
```

스토리지 구성을 결정하기 위한 비용 분석

Amazon DocumentDB를 사용하면 보유한 모든 데이터베이스 클러스터에 대한 스토리지 구성을 유연하게 선택할 수 있습니다. 표준 및 I/O 최적화 간에 클러스터를 올바르게 할당하기 위해 Amazon DocumentDB 비용을 클러스터별로 추적할 수 있습니다. 이렇게 하려면 기존 클러스터에 태그를 추가하고, AWS 결제 및 비용 관리 대시보드에서 비용 할당 태그 지정을 활성화하고, AWS Cost Explorer Service에서 지정된 클러스터에 대한 비용을 분석할 수 있습니다. 비용 분석에 대한 자세한 내용은 블로그 비용 할당 태그 사용을 참조하세요.

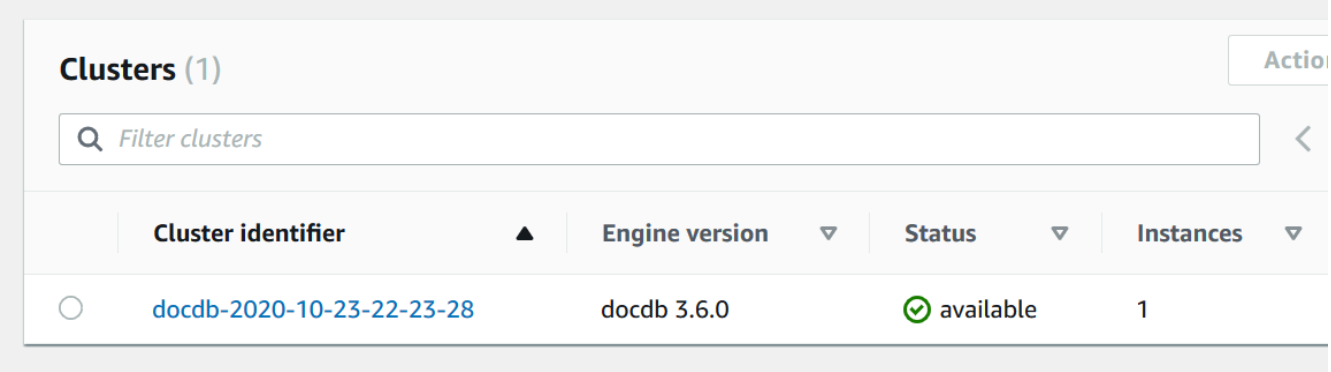
클러스터 상태 확인

AWS Management Console 또는를 사용하여 클러스터의 상태를 확인할 수 있습니다 AWS CLI.

Using the AWS Management Console

다음 절차에 따라를 사용하여 Amazon DocumentDB 클러스터의 상태를 확인합니다. AWS Management Console

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 클러스터를 선택합니다.
3. 클러스터 식별자 열에서 원하는 클러스터의 이름을 찾습니다. 그런 다음 클러스터의 상태를 찾으려면 아래와 같이 상태 열에 대한 해당 행을 읽습니다.



Using the AWS CLI

`describe-db-clusters` 작업에 따라 AWS CLI를 사용하여 Amazon DocumentDB 클러스터의 상태를 확인합니다.

다음 코드는 `sample-cluster` 클러스터의 상태를 찾습니다.

Linux, macOS 또는 Unix의 경우:

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster  \
    --query 'DBClusters[*].[DBClusterIdentifier,Status]'
```

Windows의 경우:

```
aws docdb describe-db-clusters ^
    --db-cluster-identifier sample-cluster  ^
    --query 'DBClusters[*].[DBClusterIdentifier,Status]'
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
[
    [
        "sample-cluster",
        "available"
    ]
]
```

Amazon DocumentDB 클러스터 수명 주기

Amazon DocumentDB 클러스터의 수명 주기는 클러스터 생성, 설명, 수정, 삭제로 구성되어 있습니다. 이 섹션에서는 이러한 프로세스를 완료하는 방법에 관해 설명합니다.

주제

- Amazon DocumentDB 클러스터 생성
- Amazon DocumentDB 클러스터에 대한 설명
- Amazon DocumentDB 클러스터 수정
- 대기 중인 유지 관리 확인
- 클러스터의 엔진 버전에 대한 패치 업데이트 수행
- Amazon DocumentDB 클러스터 중지 및 시작
- Amazon DocumentDB 클러스터 삭제

Amazon DocumentDB 클러스터 생성

Amazon DocumentDB 클러스터는 인스턴스와 해당 클러스터의 데이터를 나타내는 클러스터 볼륨으로 구성됩니다. 클러스터 볼륨은 3개의 가용 영역에서 단일 가상 볼륨으로 6개의 방법으로 복제됩니다. 클러스터에는 기본 인스턴스와 옵션으로 최대 15개의 복제 인스턴스가 포함됩니다.

다음 섹션에서는 AWS Management Console 또는를 사용하여 Amazon DocumentDB 클러스터를 생성하는 방법을 보여줍니다 AWS CLI. 그런 다음 해당 클러스터에 대한 복제 인스턴스를 추가할 수 있습니다. 콘솔을 사용하여 Amazon DocumentDB 클러스터를 생성하면 기본 인스턴스가 자동으로 동시에 생성됩니다. AWS CLI 를 사용하여 Amazon DocumentDB 클러스터를 생성하는 경우 클러스터의 상태를 사용할 수 있게 되면 해당 클러스터의 기본 인스턴스를 생성해야 합니다.

사전 조건

다음은 Amazon DocumentDB 클러스터를 생성할 때 필요한 사전 조건입니다.

이 없는 경우 다음 단계를 AWS 계정완료하여 생성합니다.

에 가입하려면 AWS 계정

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.

에 가입하면 AWS 계정AWS 계정 루트 사용자가 생성됩니다. 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.

VPC 사전 조건

Amazon DocumentDB 클러스터는 Amazon Virtual Private Cloud(Amazon VPC)에서만 생성할 수 있습니다. Amazon DocumentDB 클러스터에서 Amazon VPC를 사용하려면 2개 이상의 가용 영역마다 VPC에 서브넷이 1개 이상 있어야 합니다. 가용 영역에 클러스터 인스턴스를 배포하면 어쩌다가 가용 영역에 장애가 발생해도 클러스터에서 인스턴스를 사용할 수 있습니다.

서브넷 사전 조건

Amazon DocumentDB 클러스터를 생성하는 경우 VPC와 VPC 내 해당 서브넷 그룹을 선택해야 클러스터를 시작할 수 있습니다. 서브넷은 인스턴스를 시작하기 위해 사용할 가용 영역과 해당 가용 영역 내 IP 범위를 결정합니다. 설명을 위해 서브넷과 가용 영역이라는 용어가 혼용됩니다. 서브넷 그룹은 이름이 지정된 서브넷의 집합(또는 가용 영역)입니다. 서브넷 그룹을 통해 Amazon DocumentDB 인스턴스를 시작하는 데 사용할 가용 영역을 지정할 수 있습니다. 예를 들어, 인스턴스가 3개인 클러스터에서 고가용성을 위해 각 인스턴스를 별도의 가용 영역에 프로비저닝하는 것이 좋습니다. 이렇게 하면 가용 영역 하나가 작동이 안 되는 경우 인스턴스 하나만 영향을 받게 됩니다.

Amazon DocumentDB 인스턴스는 현재 최대 3개의 가용 영역에 프로비저닝할 수 있습니다. 서브넷 그룹에 서브넷이 3개 이상 있는 경우에도 Amazon DocumentDB 클러스터 생성에는 그 중 3개만 사용할 수 있습니다. 따라서 서브넷 그룹을 생성할 때 인스턴스 배포에 사용하려는 서브넷을 3개만 선택하는 것이 좋습니다. 미국 동부(버지니아 북부)에서 서브넷 그룹에는 6개의 서브넷(또는 가용 영역)이 있을 수 있습니다. 그러나 Amazon DocumentDB 클러스터가 프로비저닝되면 Amazon DocumentDB에서는 가용 영역 중 3개를 선택하여 인스턴스 프로비저닝에 사용합니다.

예를 들어, 클러스터를 생성할 때 Amazon DocumentDB에서 가용 영역{1A, 1B 및 1C}를 선택한다고 가정하겠습니다. 가용 영역{1D}에 인스턴스를 생성하려고 하면 API 직접 호출이 실패합니다. 하지만 특정 가용 영역을 지정하지 않고 인스턴스를 생성하기로 선택하면 Amazon DocumentDB가 사용자를 대신하여 가용 영역을 선택합니다. Amazon DocumentDB는 알고리즘을 사용하여 가용 영역 전체에 걸쳐 인스턴스의 부하를 분산하므로 고가용성을 확보할 수 있습니다. 예를 들어, 인스턴스 3개가 프로비

저닝되면 이러한 인스턴스가 기본적으로 가용 영역 1개에 모두 프로비저닝되지 않고 가용 영역 3개에 프로비저닝됩니다.

권장 사항:

- 특별한 사유가 없으면 항상 서브넷 그룹 서브넷을 3개 생성합니다. 이렇게 하면 인스턴스가 3개 이상인 클러스터의 경우 인스턴스가 가용 영역 3개에 프로비저닝되므로 더 높은 가용성을 달성할 수 있습니다.
- 고가용성을 달성하려면 항상 여러 가용 영역에 인스턴스를 분산시킵니다. 가용 영역 1개에 클러스터의 모든 인스턴스를 배치하지 마세요.
- 장애 조치 이벤트는 언제든지 발생할 수 있으므로 기본 인스턴스나 복제본 인스턴스가 항상 특정 가용 영역에 있다고 가정해서는 안 됩니다.

추가 사전 조건

다음은 Amazon DocumentDB 클러스터를 생성할 때 필요한 기타 사전 요구 사항입니다.

- AWS Identity and Access Management (IAM) 자격 증명을 AWS 사용하여에 연결하는 경우 IAM 계정에 Amazon DocumentDB 작업을 수행하는 데 필요한 권한을 부여하는 IAM 정책이 있어야 합니다.

  IAM 계정을 사용하여 Amazon DocumentDB 콘솔에 액세스하는 경우 먼저 IAM 계정으로 AWS Management Console 에 로그인해야 합니다. https://console.aws.amazon.com/docdb에서 Amazon DocumentDB 콘솔로 이동합니다.
- 클러스터의 구성 파라미터를 사용자 지정하려면 클러스터 파라미터 그룹과 파라미터 그룹을 필요한 파라미터 설정으로 지정해야 합니다. 클러스터 파라미터 그룹 또는 파라미터 그룹의 생성 또는 수정에 대한 자세한 내용은 Amazon DocumentDB 클러스터 파라미터 그룹 관리 섹션을 참조하세요.
- 클러스터에 지정할 TCP/IP 포트 번호를 결정해야 합니다. 일부 기업에서는 방화벽이 Amazon DocumentDB에 대한 기본 포트 연결을 차단하는 경우도 있습니다. 이처럼 기업 방화벽이 기본 포트를 차단할 경우 클러스터에 다른 포트를 선택해야 합니다. 클러스터의 인스턴스는 모두 동일한 포트를 사용합니다.

를 사용하여 클러스터 및 기본 인스턴스 생성 AWS Management Console

다음 절차에서는 콘솔을 사용하여 인스턴스가 1개 이상인 Amazon DocumentDB 클러스터를 시작하는 방법을 설명합니다.

클러스터 생성: 기본 설정 사용

를 사용하여 기본 설정을 사용하여 인스턴스가 있는 클러스터를 생성하려면 AWS Management Console

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 미국 동부(버지니아 북부) 리전 AWS 리전 이외의에서 클러스터를 생성하려면 콘솔의 오른쪽 상단 섹션에 있는 목록에서 리전을 선택합니다.
3. 탐색 창에서 클러스터를 선택한 다음 생성을 선택합니다.

   > **Tip**
   >
   > 화면 왼쪽에 탐색 창이 표시되지 않으면 페이지 왼쪽 상단 모서리에서 메뉴 아이콘 (≡) 을 선택합니다.

4. Amazon DocumentDB 클러스터 생성 페이지에서 구성 창을 작성합니다.

   a. 클러스터 식별자 - Amazon DocumentDB 제공 이름을 수락하거나 **`sample-cluster`**와 같은 클러스터 이름을 입력합니다.

      클러스터 명명 제약 조건:

      - 길이는 [1-63]개의 글자, 숫자 또는 하이픈입니다.
      - 첫 번째 문자는 글자이어야 합니다.
      - 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다.
      - 리전별로 Amazon RDS, Neptune 및 Amazon DocumentDB의 모든 클러스터 AWS 계정에 대해 고유해야 합니다.

   b. 엔진 버전 - 기본 엔진 버전인 5.0.0을 수락하거나 선택적으로 4.0.0 또는 3.6.0을 선택합니다.

   c. 인스턴스 클래스 - 기본값 `db.r5.large`를 수락하거나 목록에서 원하는 인스턴스 클래스를 선택합니다.

   d. 인스턴스 수 - 목록에서 이 클러스터로 생성하려는 인스턴스 수를 선택합니다. 첫 번째 인스턴스는 기본 인스턴스이며, 그 외 다른 모든 인스턴스는 읽기 전용 복제본 인스턴스입니다. 나중에 필요하면 인스턴트를 추가하거나 삭제할 수 있습니다. 기본적으로 Amazon DocumentDB 클러스터는 세 개의 인스턴트를 시작합니다(기본 한 개, 복제본 두 개).

5. 클러스터 스토리지 구성 섹션을 완료합니다.

Amazon DocumentDB Standard(기본값) 또는 Amazon DocumentDB I/O 최적화를 선택합니다. 자세한 내용은 Amazon DocumentDB 클러스터 스토리지 구성 단원을 참조하십시오.

6. 인증 창을 작성합니다.

   a. 사용자 이름 - 기본 사용자의 이름을 입력합니다. 클러스터에 로그인하려면 기본 사용자 이름을 사용해야 합니다.

      기본 사용자 이름 지정 시 제약 조건:

      - 길이는 [1-63]자의 영숫자 문자입니다.
      - 첫 번째 문자는 글자이어야 합니다.
      - 데이터베이스 엔진에 포함된 단어는 사용할 수 없습니다.

   b. 다음 암호 옵션 중 하나를 선택합니다.

      - 관리형 AWS Secrets Manager- 기본 사용자 암호를 자동으로 관리 AWS Secrets Manager 하려면이 옵션을 선택합니다.

        이 옵션을 선택하는 경우 자체 키를 생성하거나 Secrets Manager가 생성하는 키를 사용하여 KMS 키를 구성합니다.

      - 자체 관리형 - 기본 사용자 암호를 자체 관리하려면이 옵션을 선택합니다. 이 옵션을 선택하는 경우 기본 사용자의 암호를 입력한 다음 확인합니다. 클러스터에 로그인하려면 기본 사용자의 암호를 사용해야 합니다.

        암호 제약:

        - 길이는 [8-100]자의 인쇄 가능한 ASCII 문자입니다.
        - 다음을 제외한 인쇄 가능한 ASCII 문자를 사용할 수 있습니다.
          - **/**(슬래시)
          - **"**(큰 따옴표)
          - **@**(at 기호)

7. 화면 하단에서 다음 중 하나를 선택합니다.

   - 지금 클러스터를 생성하려면 클러스터 생성을 선택합니다.
   - 클러스터를 생성하지 않으려면 않으려면 취소를 선택합니다.
   - 생성 전 클러스터를 추가로 구성하려면 추가 구성 표시를 선택한 후 클러스터 생성: 추가 구성에서 계속 진행합니다.

추가 구성 섹션에 포함된 구성은 다음과 같습니다.

- 네트워크 설정 - 기본값은 `default` VPC 보안 그룹의 사용입니다.
- 클러스터 옵션 - 기본값은 포트 27017과 기본 파라미터 그룹을 사용하는 것입니다.
- 암호화 - 기본값은 `(default) aws/rds` 키를 사용한 암호화 활성화입니다.

> **Important**
>
> 클러스터가 암호화되면 암호화를 해제할 수 없습니다.

- 백업 - 기본값은 백업을 1일 동안 유지하며 Amazon DocumentDB에서 백업 기간을 선택하는 것입니다.
- 로그 내보내기 - 기본값은 감사 로그를 CloudWatch 로그로 내보내지 않는 것입니다.
- 유지 관리 - 기본값은 Amazon DocumentDB에서 유지 관리 기간을 선택하는 것입니다.
- 삭제 방지 - 실수로 인한 삭제로부터 클러스터를 보호합니다. 콘솔을 사용해 생성된 클러스터의 기본값이 활성화됩니다.

지금 기본 설정을 수락해도 나중에 클러스터를 수정하여 대부분을 변경할 수 있습니다.

8. 클러스터의 보안 그룹에 대해 인바운드 연결을 활성화합니다.

   클러스터의 기본 설정을 변경하지 않은 경우, 특정 리전의 기본 VPC에 대해 기본 보안 그룹을 사용하는 클러스터를 생성합니다. Amazon DocumentDB에 연결하려면 클러스터의 보안 그룹에 대해 포트 27017(또는 사용자가 선택한 포트)에서 인바운드 연결을 활성화해야 합니다.

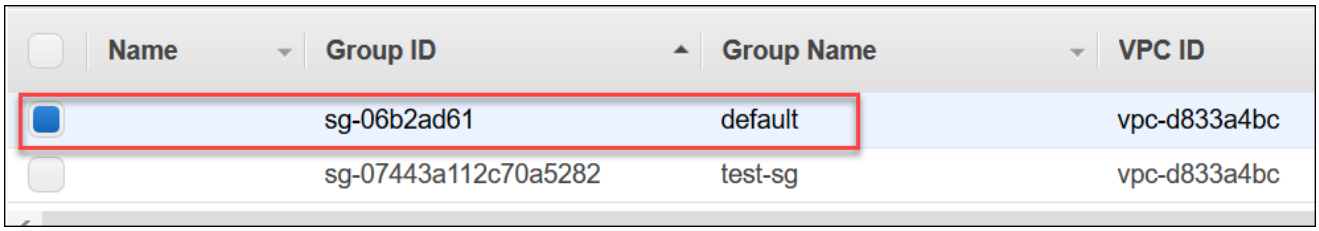
   클러스터의 보안 그룹에 대해 인바운드 연결을 추가하려면

   a. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/ec2/ Amazon EC2 콘솔을 엽니다.

   b. 기본 창의 리소스 섹션에서 보안 그룹을 선택합니다.



c. 보안 그룹 목록에서 클러스터 생성 시 사용한 보안 그룹을 찾고(기본값 보안 그룹일 확률이 높음) 보안 그룹 이름 왼쪽의 상자를 선택합니다.



d. 작업 메뉴에서 인바운드 규칙 편집을 선택한 다음 규칙 제약을 선택 또는 입력합니다.

   i. 유형 - 목록에서 네트워크 트래픽에 개방할 프로토콜을 선택합니다.

   ii. 프로토콜 - 목록에서 프로토콜 유형을 선택합니다.

   iii. 포트 범위 - 사용자 지정 규칙에 대해 포트 번호나 포트 범위를 입력합니다. 포트 번호 또는 범위에는 클러스터를 생성할 때 지정한 포트가 포함됩니다(기본값: 27017).

   iv. 소스 - 인스턴스에 도달 가능한 트래픽을 지정합니다. 목록에서 트래픽 소스를 선택합니다. 사용자 지정을 선택하는 경우, 단일 IP 주소나 IP 주소 범위를 CIDR 표기법으로 지정합니다(예: 203.0.113.5/32).

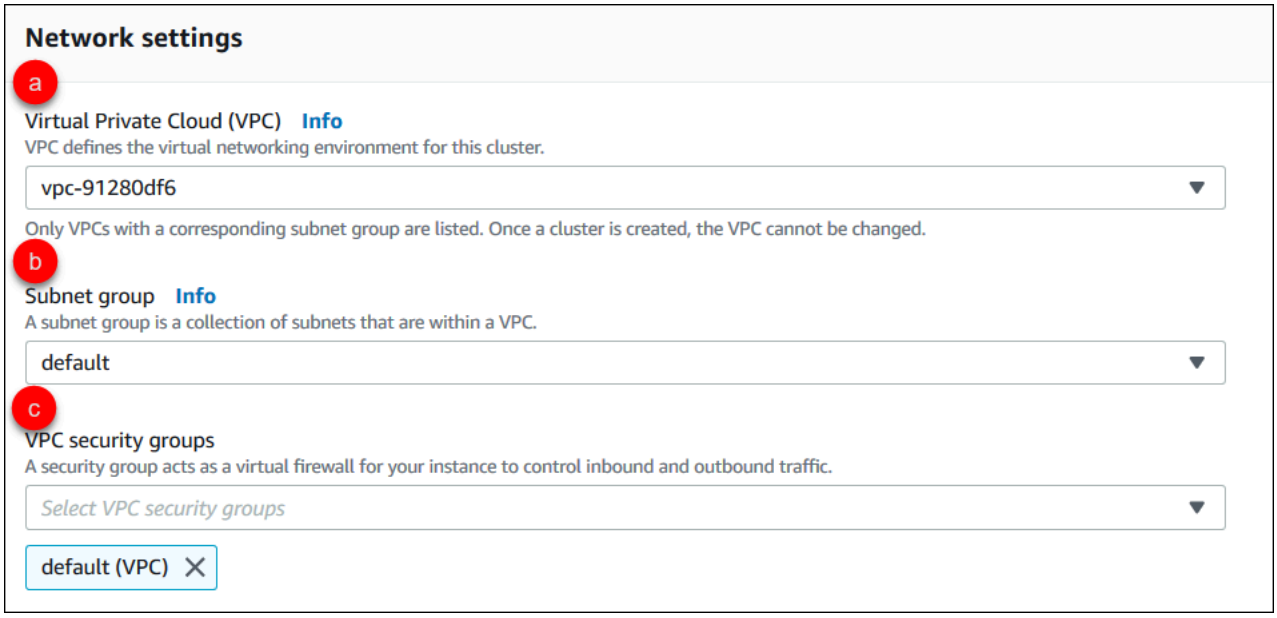
   v. 설명 - 이 규칙에 대한 설명을 입력합니다.

   vi. 규칙 만들기가 끝나면 저장을 선택합니다.

클러스터 생성: 추가 구성

클러스터에 대한 기본 설정을 수락하면 다음 단계를 건너뛰고 클러스터 생성을 선택할 수 있습니다.

1. 네트워크 설정 창을 작성합니다.



a. 가상 프라이빗 클라우드(VPC) - 목록에서 이 클러스터를 시작할 Amazon VPC를 선택합니다.

b. 서브넷 그룹 - 목록에서 이 클러스터에 사용할 서브넷 그룹을 선택합니다.

c. VPC 보안 그룹 - 목록에서 이 클러스터에 사용할 VPC 보안 그룹을 선택합니다.

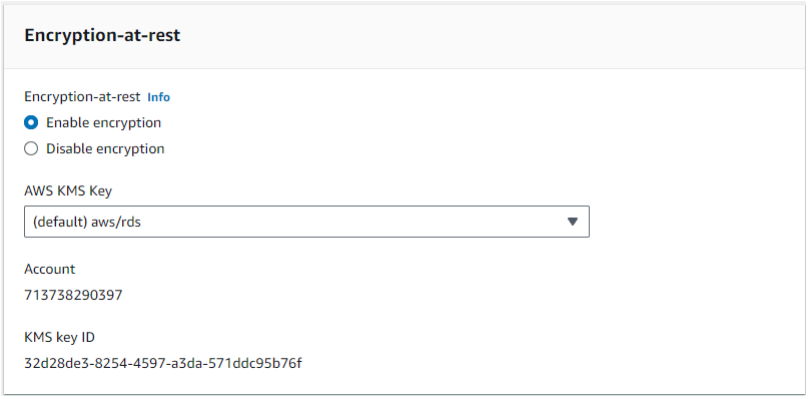
2. 클러스터 옵션 창을 작성합니다.



a. 데이터베이스 포트 - 위, 아래 화살표를 사용해 애플리케이션과 인스턴스 연결에 사용할 TCP/IP 포트를 설정합니다.

b. 클러스터 파라미터 그룹 - 파라미터 그룹 목록에서 이 클러스터에 사용할 클러스터 파라미터 그룹을 선택합니다.

3. 암호화 창을 작성합니다.



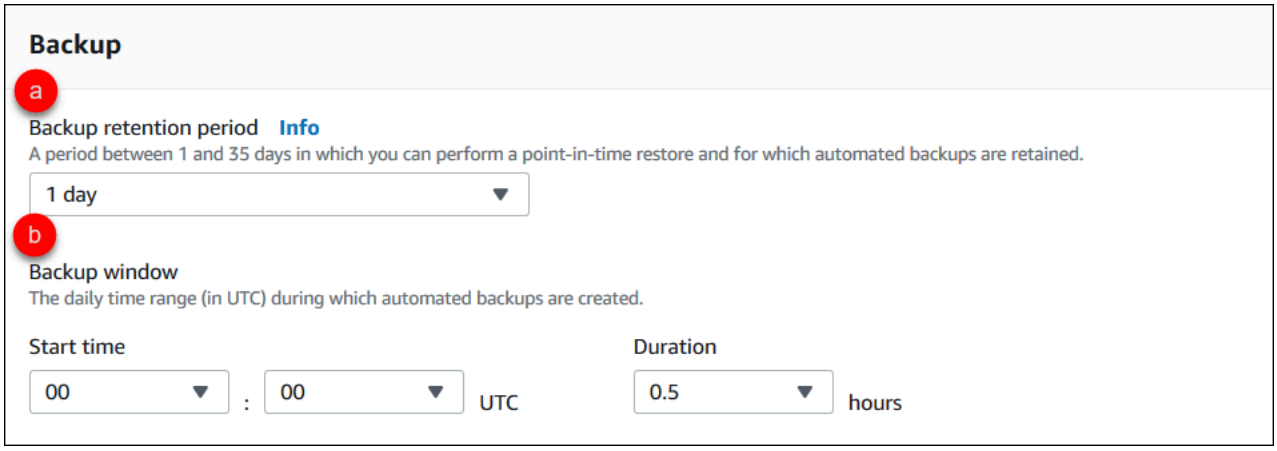
a. 유휴 암호화 - 다음 중 하나를 선택합니다.

- 암호화 활성 - 기본 설정입니다. 모든 저장 데이터는 암호화됩니다. 데이터 암호화를 선택하면 해당 작업을 실행 취소할 수 없습니다.
- 암호화 비활성 - 데이터가 암호화되지 않습니다.

b. AWS KMS 키 - 데이터를 암호화하는 경우에만 사용할 수 있습니다. 목록에서 이 클러스터에 있는 데이터 암호화에 사용할 키를 선택합니다. 기본값은 `(default) aws/rds`입니다.

키 ARN 입력을 선택하면 해당 키의 Amazon 리소스 이름(ARN)을 입력해야 합니다.

4. 백업 창을 작성합니다.



a. 백업 보존 기간 - 목록에서 이 클러스터의 자동 백업을 삭제하지 않고 유지하는 일 수를 선택합니다.

b. 백업 기간 - Amazon DocumentDB에서 이 클러스터를 백업하는 일별 시간과 기간을 설정합니다.

i. 시작 시간 - 첫 번째 목록에서 자동 백업 시작 시간(UTC)을 선택합니다. 두 번째 목록에서 자동 백업 시작 시간의 분을 선택합니다.

ii. 기간 - 목록에서 자동 백업 생성에 할당된 시간 수를 선택합니다.

5. CloudWatch Logs로 내보내려는 로그 유형을 선택하여 로그 내보내기 창을 완성하세요.

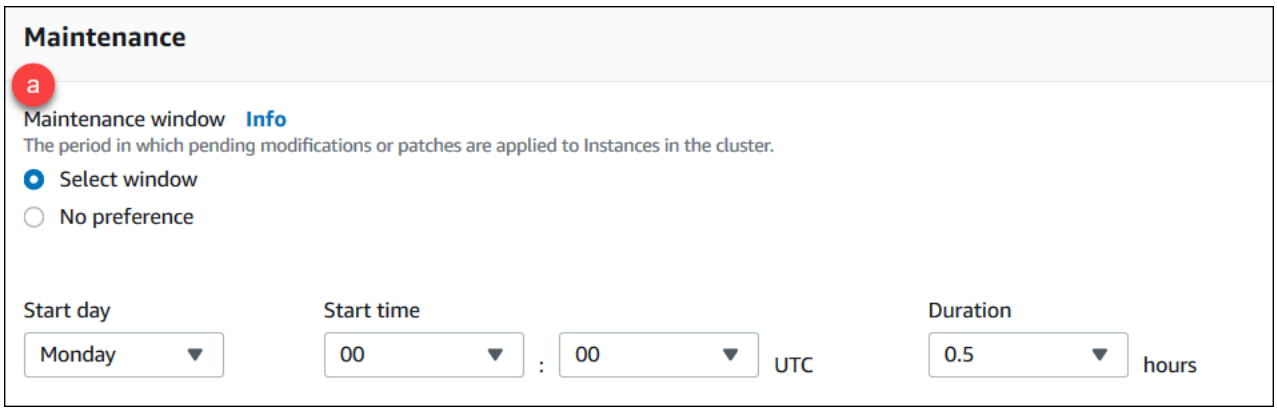


- 감사 로그 - Amazon CloudWatch Logs에 감사 로그 내보내기를 활성화하려면 이 옵션을 선택합니다. 감사 로그를 선택하는 경우 클러스터의 사용자 지정 파라미터 그룹에서 `audit_logs`를 활성화해야 합니다. 자세한 내용은 Amazon DocumentDB 이벤트 감사 섹션을 참조하십시오.
- 프로파일러 로그 - 작업 프로파일러 로그를 Amazon CloudWatch Logs로 내보내려면 이 옵션을 선택합니다. 프로파일러 로그를 선택하는 경우 클러스터의 사용자 지정 파라미터 그룹에서 다음 파라미터도 수정해야 합니다.

  - `profiler` - `enabled`로 설정합니다.
  - `profiler_threshold_ms` - 작업 프로파일링에 대한 임계값을 설정하려면 값 `[0-INT_MAX]`로 설정합니다.
  - `profiler_sampling_rate` - 프로파일링할 느린 작업 비율을 설정하려면 값 `[0.0-1.0]`으로 설정합니다.

  자세한 내용은 Amazon DocumentDB 작업 프로파일링 섹션을 참조하십시오.

6. 유지 관리 창을 작성합니다.



- 다음 중 하나를 선택합니다.

  - 선택 창 - 클러스터에 대한 유지 관리를 수행할 Amazon DocumentDB에 대한 요일, UTC 시작 시간 및 기간을 지정할 수 있습니다.

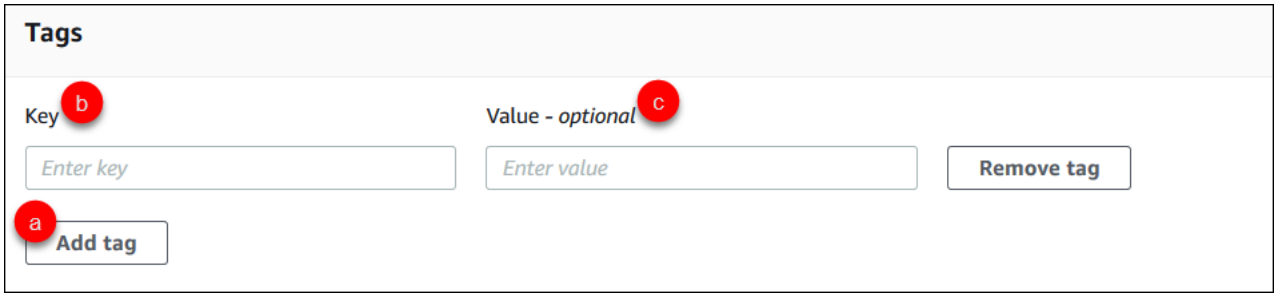
a. 시작일 - 목록에서 클러스터 유지 관리를 시작할 요일을 선택합니다.

b. 시작 시간 - 목록에서 유지 관리를 시작할 시간과 분(UTC)을 선택합니다.

c. 기간 - 목록에서 클러스터 유지 관리에 할당할 시간의 양을 선택합니다. 유지 관리 작업을 특정 시간에 끝내지 못하면 특정 시간이 지나도 완료될 때까지 유지 관리 프로세스가 계속 진행됩니다.

- 기본 설정 없음 - Amazon DocumentDB에서 유지 관리를 수행할 요일, 시작 시간 및 기간을 선택합니다.

7. 이 클러스터에 하나 이상의 태그를 추가하려면 태그 창을 작성합니다.



클러스터에 추가하려는 각 태그에 대해 다음 단계를 반복합니다. 클러스터 하나에 최대 10개까지 추가할 수 있습니다.

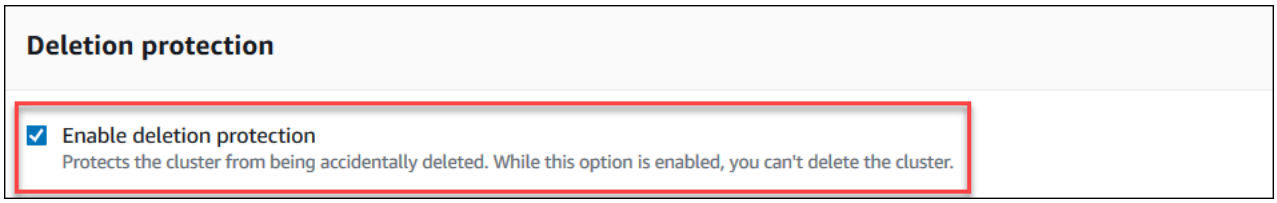
a. 태그 추가를 선택합니다.

b. 태그의 키를 입력합니다.

c. 선택적으로 태그의 값을 입력합니다.

태그를 제거하려면 태그 제거를 선택합니다.

8. 콘솔을 사용하여 클러스터를 생성할 때 기본적으로 삭제 방지가 활성화됩니다. 삭제 방지를 비활성화하려면 삭제 방지 활성화를 선택 취소합니다. 삭제 방지를 활성화하면 클러스터가 삭제되지 않도록 방지합니다. 삭제 방지된 클러스터를 삭제하려면 먼저 클러스터를 수정하여 삭제 방지를 비활성화해야 합니다.



삭제 방지에 대한 자세한 내용은 Amazon DocumentDB 클러스터 삭제 섹션을 참조하세요.

9. 클러스터를 생성하려면 클러스터 생성을 선택합니다. 그렇지 않은 경우 취소를 선택합니다.

를 사용하여 클러스터 생성 AWS CLI

다음 절차에서는를 사용하여 Amazon DocumentDB 클러스터를 AWS CLI 시작하고 Amazon DocumentDB 복제본을 생성하는 방법을 설명합니다.

파라미터

- **`--db-cluster-identifier`** - 필수입니다. 이 클러스터를 식별하는 소문자 문자열입니다.

  클러스터 명명 제약 조건:

  - 길이는 [1-63] 글자, 숫자 또는 하이픈입니다.
  - 첫 번째 문자는 글자이어야 합니다.
  - 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다.
  - 리전별로 AWS 계정당 모든 클러스터(Amazon RDS, Amazon Neptune 및 Amazon DocumentDB)에 대해 고유해야 합니다.

- **`--engine`** - 필수입니다. **`docdb`**여야 합니다.
- **`--deletion-protection | --no-deletion-protection`** - 선택 사항. 삭제 방지를 활성화하면 클러스터가 삭제되지 않도록 방지합니다. 를 사용할 때 기본 설정은 삭제 방지 AWS CLI기능을 비활성화하는 것입니다.

  삭제 방지에 대한 자세한 내용은 Amazon DocumentDB 클러스터 삭제 섹션을 참조하세요.
- **`--storage-type standard | iopt1`** - 선택 사항. 기본값: **`standard`**. 클러스터 스토리지 구성입니다. 유효한 값은 `standard` (표준) 또는 `iopt1` (I/O 최적화)입니다.
- **`--master-username`** - 필수입니다. 사용자 인증에 사용되는 사용자 이름입니다.

  마스터 사용자 명명 제약:

  - 길이는 [1-63]자의 영숫자 문자입니다.
  - 첫 번째 문자는 글자이어야 합니다.
  - 데이터베이스 엔진에 포함된 단어는 사용할 수 없습니다.

- **`--master-user-password`** - 선택 사항. 사용자 인증에 사용되는 사용자 암호입니다.

  마스터 암호 제약

  - 길이는 [8-100]자의 인쇄 가능한 ASCII 문자입니다.
  - 다음을 제외한 인쇄 가능한 ASCII 문자를 사용할 수 있습니다.
    - **/**(슬래시)
    - **"**(큰 따옴표)
    - **@**(at 기호)

- **`--manage-master-user-password`** - 선택 사항. Amazon DocumentDB는 마스터 사용자 암호를 생성하고 Secrets Manager에서 수명 주기 동안 관리합니다.

추가 파라미터는 CreateDBCluster 섹션을 참조하세요.

를 사용하여 Amazon DocumentDB 클러스터를 시작하려면 AWS CLI

Amazon DocumentDB 클러스터를 생성하려면를 호출합니다`create-db-cluster` AWS CLI. 다음 AWS CLI 명령은 삭제 방지가 활성화된 라는 Amazon DocumentDB 클러스터`sample-cluster`를 생성합니다. 삭제 방지에 대한 자세한 내용은 Amazon DocumentDB 클러스터 삭제을(를) 참조하세요.

또한 `--engine-version`은 기본적으로 최신 주요 엔진 버전으로 설정되는 선택적 매개변수입니다. 현재 메이저 엔진 버전은 5.0.0입니다. 새 메이저 엔진 버전이 릴리스되면의 기본 엔진 버전이 마지막 메이저 엔진 버전을 반영하도록 `--engine-version` 업데이트됩니다. 따라서 프로덕션 워크로드, 특히 스크립팅, 자동화 또는 AWS CloudFormation 템플릿에 의존하는 워크로드의 경우를 의도한 메이저 버전`--engine-version`에 명시적으로 지정하는 것이 좋습니다.

> **Note**
>
> `db-subnet-group-name` 또는 `vpc-security-group-id`이(가) 지정되지 않은 경우 Amazon DocumentDB는 해당 리전에 대해 기본 서브넷 그룹과 Amazon VPC 보안 그룹을 사용합니다.

Linux, macOS, Unix의 경우:

```
aws docdb create-db-cluster \
      --db-cluster-identifier sample-cluster \
      --engine docdb \
      --engine-version 5.0.0 \
      --deletion-protection \
      --master-username masteruser \
```

```
      --master-user-password password
```

Windows의 경우:

```
aws docdb create-db-cluster ^
      --db-cluster-identifier sample-cluster ^
      --engine docdb ^
      --engine-version 5.0.0 ^
      --deletion-protection ^
      --master-username masteruser ^
      --master-user-password password
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
    "DBCluster": {
        "StorageEncrypted": false,
        "DBClusterMembers": [],
        "Engine": "docdb",
        "DeletionProtection" : "enabled",
        "ClusterCreateTime": "2018-11-26T17:15:19.885Z",
        "DBSubnetGroup": "default",
        "EngineVersion": "5.0.0",
        "MasterUsername": "masteruser",
        "BackupRetentionPeriod": 1,
        "DBClusterArn": "arn:aws:rds:us-east-1:123456789012:cluster:sample-cluster",
        "DBClusterIdentifier": "sample-cluster",
        "MultiAZ": false,
        "DBClusterParameterGroup": "default.docdb5.0",
        "PreferredBackupWindow": "09:12-09:42",
        "DbClusterResourceId": "cluster-KQSGI4MHU4NTDDRVNLNTU7XVAY",
        "PreferredMaintenanceWindow": "tue:04:17-tue:04:47",
        "Port": 27017,
        "Status": "creating",
        "ReaderEndpoint": "sample-cluster.cluster-ro-sfcrlcjcoroz.us-
east-1.docdb.amazonaws.com",
        "AssociatedRoles": [],
        "HostedZoneId": "ZNKXTT8WH85VW",
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-77186e0d",
                "Status": "active"
            }
```

```
        ],
        "AvailabilityZones": [
            "us-east-1a",
            "us-east-1c",
            "us-east-1e"
        ],
        "Endpoint": "sample-cluster.cluster-sfcrlcjcoroz.us-east-1.docdb.amazonaws.com"
    }
}
```

클러스터를 생성하는 데 몇 분 정도 걸립니다. AWS Management Console 또는를 사용하여 클러스터의 상태를 AWS CLI 모니터링할 수 있습니다. 자세한 내용은 Amazon DocumentDB 클러스터 상태 모니터링 단원을 참조하십시오.

> **Important**
>
> AWS CLI 를 사용하여 Amazon DocumentDB 클러스터를 생성하면 인스턴스가 생성되지 않습니다. 따라서 기본 인스턴스와 필요한 복제 인스턴스를 명시적으로 생성해야 합니다. 콘솔 또는를 사용하여 인스턴스 AWS CLI 를 생성할 수 있습니다. 자세한 내용은 클러스터에 Amazon DocumentDB 인스턴스 추가 단원을 참조하십시오.

자세한 내용은 Amazon DocumentDB API 참조의 `CreateDBCluster` 섹션을 참조하세요.

Amazon DocumentDB 클러스터에 대한 설명

Amazon DocumentDB Management Console 또는 AWS CLI 를 사용하여 Amazon DocumentDB 클러스터와 관련된 연결 엔드포인트, 보안 그룹, VPCs 및 파라미터 그룹과 같은 세부 정보를 볼 수 있습니다.

자세한 내용은 다음 자료를 참조하세요.

- Amazon DocumentDB 클러스터 상태 모니터링
- 클러스터 엔드포인트 찾기

Using the AWS Management Console

다음 절차에 따라 콘솔을 사용하여 지정된 Amazon DocumentDB 클러스터의 세부 정보를 확인합니다.

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 클러스터를 선택합니다.

   > **Tip**
   >
   > 화면 왼쪽에 탐색 창이 표시되지 않으면 페이지 왼쪽 상단 모서리에서 메뉴 아이콘 (≡) 을 선택합니다.

3. 클러스터 목록에서 세부 정보를 보려는 클러스터의 이름을 선택합니다. 클러스터에 대한 정보는 다음과 같은 그룹으로 구성됩니다.

   - 요약 - 엔진 버전, 클러스터 상태, 보류 중인 유지 관리 및 파라미터 그룹의 상태를 포함하여 클러스터에 대한 일반 정보입니다.
   - 연결 및 보안 - 연결 섹션에는 mongo 쉘 또는 애플리케이션을 사용하여 이 클러스터에 연결할 연결 엔드포인트가 나열됩니다. 보안 그룹 섹션에는 이 클러스터와 연결된 보안 그룹과 해당 VPC ID 및 설명이 나열됩니다.
   - 구성 - 클러스터 세부 정보 섹션에는 클러스터의 Amazon 리소스 이름(ARN), 엔드포인트 및 파라미터 그룹을 포함하여 클러스터에 대한 세부 정보가 나열됩니다. 또한 클러스터의 백업 정보, 유지 관리 세부 정보, 보안 및 네트워크 설정이 나열됩니다. 클러스터 인스턴스 섹션에는 각 인스턴스의 역할 및 클러스터 파라미터 그룹 상태와 함께 이 클러스터에 속하는 인스턴스가 나열됩니다.
   - 모니터링 - 이 클러스터의 Amazon CloudWatch Logs 측정치. 자세한 내용은 CloudWatch를 사용하여 Amazon DocumentDB 모니터링 섹션을 참조하십시오.
   - 이벤트 및 태그 - 최근 이벤트 섹션에는 이 클러스터의 최근 이벤트가 나열됩니다. Amazon DocumentDB에서는 클러스터, 인스턴스, 스냅샷, 보안 그룹 및 클러스터 파라미터 그룹과 관련된 이벤트 레코드를 유지합니다. 이 정보에는 각 이벤트와 연결된 날짜, 시간 및 메시지가 포함됩니다. 태그 섹션에는 이 클러스터에 연결된 태그가 나열됩니다.

Using the AWS CLI

를 사용하여 Amazon DocumentDB 클러스터의 세부 정보를 보려면 아래 예제와 같이 `describe-db-clusters` 명령을 AWS CLI사용합니다. 자세한 내용은 Amazon DocumentDB 리소스 관리 API 참조에서 `DescribeDBClusters` 섹션을 참조하세요.

> **Note**
>
> 클러스터 및 인스턴스 수명주기 관리와 같은 특정 관리 기능의 경우 Amazon DocumentDB는 Amazon RDS와 공유하는 운영 기술을 활용합니다. `filterName=engine,Values=docdb` 필터 파라미터는 Amazon DocumentDB 클러스터만 반환합니다.

Example

예 1: 모든 Amazon DocumentDB 클러스터를 나열합니다.

다음 AWS CLI 코드는 리전의 모든 Amazon DocumentDB 클러스터에 대한 세부 정보를 나열합니다.

```
aws docdb describe-db-clusters --filter Name=engine,Values=docdb
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
{
    "DBClusters": [
        {
            "AvailabilityZones": [
                "us-east-1c",
                "us-east-1b",
                "us-east-1a"
            ],
            "BackupRetentionPeriod": 1,
            "DBClusterIdentifier": "sample-cluster-1",
            "DBClusterParameterGroup": "sample-parameter-group",
            "DBSubnetGroup": "default",
            "Status": "available",
            ...
        },
        {
            "AvailabilityZones": [
                "us-east-1c",
                "us-east-1b",
                "us-east-1a"
            ],
            "BackupRetentionPeriod": 1,
            "DBClusterIdentifier": "sample-cluster-2",
```

```
            "DBClusterParameterGroup": "sample-parameter-group",
            "DBSubnetGroup": "default",
            "Status": "available",
            ...
        },
        {
            "AvailabilityZones": [
                "us-east-1c",
                "us-east-1b",
                "us-east-1a"
            ],
            "BackupRetentionPeriod": 1,
            "DBClusterIdentifier": "sample-cluster-3",
            "DBClusterParameterGroup": "sample-parameter-group",
            "DBSubnetGroup": "default",
            "Status": "available",
            ...
        }
    ]
}
```

Example

예 2: 지정된 Amazon DocumentDB 클러스터에 대한 모든 세부 정보 나열

다음 AWS CLI 코드는 클러스터에 대한 세부 정보를 나열합니다`sample-cluster`.

Linux, macOS, Unix의 경우:

```
aws docdb describe-db-clusters \
   --filter Name=engine,Values=docdb \
   --db-cluster-identifier sample-cluster
```

Windows의 경우:

```
aws docdb describe-db-clusters ^
   --filter Name=engine,Values=docdb ^
   --db-cluster-identifier sample-cluster
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
{
    "DBClusters": [
```

```
        {
            "AllocatedStorage": 1,
            "AvailabilityZones": [
                "us-east-1c",
                "us-east-1a",
                "us-east-1d"
            ],
            "BackupRetentionPeriod": 2,
            "DBClusterIdentifier": "sample-cluster",
            "DBClusterParameterGroup": "sample-parameter-group",
            "DBSubnetGroup": "default",
            "Status": "available",
            "EarliestRestorableTime": "2023-11-07T22:34:08.148000+00:00",
            "Endpoint": "sample-cluster.node.us-east-1.amazon.com",
            "ReaderEndpoint": "sample-cluster.node.us-east-1.amazon.com",
            "MultiAZ": false,
            "Engine": "docdb",
            "EngineVersion": "5.0.0",
            "LatestRestorableTime": "2023-11-10T07:21:16.772000+00:00",
            "Port": 27017,
            "MasterUsername": "chimeraAdmin",
            "PreferredBackupWindow": "22:22-22:52",
            "PreferredMaintenanceWindow": "sun:03:01-sun:03:31",
            "ReadReplicaIdentifiers": [],
            "DBClusterMembers": [
                {
                    "DBInstanceIdentifier": "sample-instance-1",
                    "IsClusterWriter": true,
                    "DBClusterParameterGroupStatus": "in-sync",
                    "PromotionTier": 1
                },
                {
                    "DBInstanceIdentifier": "sample-instance-2",
                    "IsClusterWriter": true,
                    "DBClusterParameterGroupStatus": "in-sync",
                    "PromotionTier": 1
                },

            ],
            "VpcSecurityGroups": [
                {
                    "VpcSecurityGroupId": "sg-9084c2ec",
                    "Status": "active"
                }
```

```
            ],
            "HostedZoneId": "Z06853723JYKYBXTJ49RB",
            "StorageEncrypted": false,
            "DbClusterResourceId": "cluster-T4LGLANHVAPGQYYULWUDKLVQL4",
            "DBClusterArn": "arn:aws:rds:us-east-1:123456789012:cluster:sample-
cluster",
            "AssociatedRoles": [],
            "IAMDatabaseAuthenticationEnabled": false,
            "ClusterCreateTime": "2023-11-06T18:05:41.568000+00:00",
            "EngineMode": "provisioned",
            "DeletionProtection": false,
            "HttpEndpointEnabled": false,
            "CopyTagsToSnapshot": false,
            "CrossAccountClone": false,
            "DomainMemberships": [],
            "TagList": [],
            "StorageType": "iopt1",
            "AutoMinorVersionUpgrade": false,
            "NetworkType": "IPV4",
            "IOOptimizedNextAllowedModificationTime":
 "2023-12-07T18:05:41.580000+00:00"
        }
    ]
}
```

Example

예 3: Amazon DocumentDB 클러스터에 대한 특정 세부 정보 나열

를 사용하여 클러스터 세부 정보의 하위 집합을 나열하려면 `describe-db-clusters` 작업을 나열할 클러스터 멤버를 `--query` 지정하는를 AWS CLI추가합니다. `--db-cluster-identifier` 파라미터는 세부 정보를 표시할 특정 클러스터의 식별자입니다. 쿼리에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서의 `--query` 옵션을 사용하여 출력을 필터링하는 방법을 참조하세요.

다음 예제에서는 Amazon DocumentDB 클러스터의 인스턴스를 나열합니다.

Linux, macOS, Unix의 경우:

```
aws docdb describe-db-clusters \
    --filter Name=engine,Values=docdb \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].[DBClusterMembers]'
```

Windows의 경우:

```
aws docdb describe-db-clusters ^
    --filter Name=engine,Values=docdb ^
    --db-cluster-identifier sample-cluster ^
    --query 'DBClusters[*].[DBClusterMembers]'
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
[
    [
        [
            {
                "DBInstanceIdentifier": "sample-instance-1",
                "IsClusterWriter": true,
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1
            },
            {
                "DBInstanceIdentifier": "sample-instance-2",
                "IsClusterWriter": false,
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1
            }
        ]
    ]
]
```

Amazon DocumentDB 클러스터 수정

클러스터를 수정하려면 클러스터가 사용 가능 상태에 있어야 합니다. 중지된 클러스터는 수정할 수 없습니다. 클러스터가 중지된 경우, 먼저 클러스터를 시작하고 클러스터가 사용 가능하게 될 때까지 기다린 다음 원하는 대로 수정합니다. 자세한 내용은 Amazon DocumentDB 클러스터 중지 및 시작 섹션을 참조하십시오.

Using the AWS Management Console

다음 절차에 따라 콘솔을 사용하여 특정 Amazon DocumentDB 클러스터를 수정합니다.

Amazon DocumentDB 클러스터를 수정하려면

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 클러스터를 선택합니다.

   > **Tip**
   >
   > 화면 왼쪽에 탐색 창이 표시되지 않으면 페이지 왼쪽 상단 모서리에서 메뉴 아이콘 (≡) 을 선택합니다.

3. 클러스터 이름 왼쪽에 있는 버튼을 선택하여 수정할 클러스터를 지정합니다.
4. 작업을 선택한 후 수정을 선택합니다.
5. 클러스터 수정: <cluster-name> 창에서 원하는 항목을 변경합니다. 다음 영역에서 변경할 수 있습니다.
   - 클러스터 사양 - 클러스터의 이름, 보안 그룹 및 자격 증명 관리입니다.
   - 클러스터 스토리지 구성 - 클러스터의 데이터 스토리지 모드입니다. 표준 구성과 I/O 최적화 구성 중에서 선택합니다.
   - 클러스터 옵션 - 클러스터의 포트 및 파라미터 그룹.
   - 백업 - 클러스터의 백업 보존 기간 및 백업 기간.
   - 로그 내보내기 - 감사 또는 프로파일러 로그 내보내기 활성화 또는 비활성화.
   - 유지 관리 - 클러스터의 유지 관리 기간 설정.
   - 삭제 방지 - 클러스터에서 삭제 방지 활성화 또는 비활성화. 삭제 방지 기능은 기본적으로 활성화됩니다.
6. 작업을 마쳤으면 계속을 선택하여 변경 사항의 요약을 확인합니다.
7. 변경 사항이 만족스러우면 클러스터 수정을 선택하여 클러스터를 수정할 수 있습니다. 또는 뒤로 또는 취소를 선택하여 각각 변경 사항을 편집하거나 취소할 수 있습니다.

변경 사항을 적용하는 데 몇 분 정도 걸립니다. 사용 가능 상태인 경우에만 클러스터를 사용할 수 있습니다. 콘솔 또는 AWS CLI를 사용하여 클러스터의 상태를 모니터링할 수 있습니다. 자세한 내용은 Amazon DocumentDB 클러스터 상태 모니터링 단원을 참조하십시오.

Using the AWS CLI

`modify-db-cluster` 작업에 따라 AWS CLI를 사용하여 지정된 클러스터를 수정합니다. 자세한 내용은 Amazon DocumentDB API 참조의 `ModifyDBCluster` 섹션을 참조하세요.

파라미터

- **`--db-cluster-identifier`** - 필수입니다. 수정하려는 Amazon DocumentDB 클러스터의 식별자입니다.
- **`--backup-retention-period`** - 선택 사항. 자동 백업이 보관되는 일수입니다. 유효한 값은 1–35입니다.
- **`--storage-type`** - 선택 사항. 클러스터 스토리지 구성입니다. 유효한 값은 `standard` (표준) 또는 `iopt1` (I/O 최적화)입니다.
- **`--db-cluster-parameter-group-name`** - 선택 사항. 클러스터에 사용할 클러스터 파라미터 그룹의 이름입니다.
- **`--manage-master-user-password`** - 선택 사항. Amazon DocumentDB는 마스터 사용자 암호를 생성하고 Secrets Manager에서 수명 주기 동안 관리합니다.
- **`--rotate-master-user-password`** - 선택 사항. Secrets Manager는 기존 보안 암호에 대한 새 보안 암호 버전을 생성합니다. 보안 암호의 새 버전에는 새 기본 사용자 암호가 포함되어 있습니다. Amazon DocumentDB는 클러스터의 기본 사용자 암호를 새 보안 암호 버전의 암호와 일치하도록 변경합니다.

  기본 암호를 교체할 때 **`--apply-immediately`** 옵션을 지정해야 합니다.
- **`--master-user-password`** - 선택 사항. 기본 데이터베이스 사용자의 새 암호입니다.

  암호 제약:
  - 길이는 [8-100]자의 인쇄 가능한 ASCII 문자입니다.
  - 다음을 제외한 인쇄 가능한 ASCII 문자를 사용할 수 있습니다.
    - **`/`**(슬래시)
    - **`"`**(큰 따옴표)
    - **`@`**(at 기호)
- **`--new-db-cluster-identifier`** - 선택 사항. 클러스터의 이름을 변경할 때 클러스터의 새 클러스터 식별자입니다. 이 값은 소문자 문자열로 저장됩니다.

  명명 제약 조건:
  - 길이는 [1~63] 글자, 숫자 또는 하이픈입니다.

  - 첫 번째 문자는 글자이어야 합니다.
  - 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다.
  - 리전별로 Amazon RDS, Amazon Neptune 및 Amazon DocumentDB의 모든 클러스터 AWS 계정에 대해 고유해야 합니다.
- **`--preferred-backup-window`** - 선택 사항. 자동 백업이 생성되는 일별 시간 범위를 UTC(협정 세계시)로 표시합니다.
  - 형식: `hh24:mm-hh24:mm`
- **`--preferred-maintenance-window`** - 선택 사항. 시스템 유지 관리를 실행할 수 있는 주 단위 시간 범위(UTC)입니다.
  - 형식: `ddd:hh24:mm-ddd:hh24:mm`
  - 유효한 요일: `Sun`, `Mon`, `Tue`, `Wed`, `Thu`, `Fri`, `Sat`
- **`--deletion-protection`** 또는 **`--no-deletion-protection`** - 선택 사항. 이 클러스터에서 삭제 방지를 활성화해야 하는지 여부. 삭제 방지는 삭제 방지를 비활성화하도록 클러스터를 수정할 때까지 실수로 클러스터가 삭제되는 것을 방지합니다. 자세한 내용은 Amazon DocumentDB 클러스터 삭제 섹션을 참조하십시오.
- **`--apply-immediately`** 또는 **`--no-apply-immediately`** - `--apply-immediately`를 사용하여 즉시 변경합니다. `--no-apply-immediately`를 사용하여 클러스터의 다음 유지 관리 기간에 변경합니다.

Example

다음 코드는 클러스터 `sample-cluster`에 대한 백업 보존 기간을 변경합니다.

Linux, macOS, Unix의 경우:

```
aws docdb modify-db-cluster \
       --db-cluster-identifier sample-cluster \
       --apply-immediately \
       --backup-retention-period 7
```

Windows의 경우:

```
aws docdb modify-db-cluster ^
       --db-cluster-identifier sample-cluster ^
       --apply-immediately ^
       --backup-retention-period 7
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
{
    "DBCluster": {
        "BackupRetentionPeriod": 7,
        "DbClusterResourceId": "cluster-VDP53QEWST7YHM36TTXOPJT5YE",
        "Status": "available",
        "DBClusterMembers": [
            {
                "PromotionTier": 1,
                "DBClusterParameterGroupStatus": "in-sync",
                "DBInstanceIdentifier": "sample-cluster-instance",
                "IsClusterWriter": true
            }
        ],
        "ReadReplicaIdentifiers": [],
        "AvailabilityZones": [
            "us-east-1b",
            "us-east-1c",
            "us-east-1a"
        ],
        "ReaderEndpoint": "sample-cluster.cluster-ro-ctevjxdlur57.us-
east-1.rds.amazonaws.com",
        "DBClusterArn": "arn:aws:rds:us-east-1:123456789012:cluster:sample-cluster",
        "PreferredMaintenanceWindow": "sat:09:51-sat:10:21",
        "EarliestRestorableTime": "2018-06-17T00:06:19.374Z",
        "StorageEncrypted": false,
        "MultiAZ": false,
        "AssociatedRoles": [],
        "MasterUsername": "<your-master-user-name>",
        "DBClusterIdentifier": "sample-cluster",
        "VpcSecurityGroups": [
            {
                "Status": "active",
                "VpcSecurityGroupId": "sg-77186e0d"
            }
        ],
        "HostedZoneId": "Z2SUY0A1719RZT",
        "LatestRestorableTime": "2018-06-18T21:17:05.737Z",
        "AllocatedStorage": 1,
        "Port": 27017,
        "Engine": "docdb",
        "DBClusterParameterGroup": "default.docdb3.4",
```

```
        "Endpoint": "sample-cluster.cluster-ctevjxdlur57.us-
east-1.rds.amazonaws.com",
        "DBSubnetGroup": "default",
        "PreferredBackupWindow": "00:00-00:30",
        "EngineVersion": "3.4",
        "ClusterCreateTime": "2018-06-06T19:25:47.991Z",
        "IAMDatabaseAuthenticationEnabled": false
    }
}
```

변경 사항을 적용하는 데 몇 분 정도 걸립니다. 사용 가능 상태인 경우에만 클러스터를 사용할 수 있습니다. 콘솔 또는 AWS CLI를 사용하여 클러스터의 상태를 모니터링할 수 있습니다. 자세한 내용은 Amazon DocumentDB 클러스터 상태 모니터링 단원을 참조하십시오.

대기 중인 유지 관리 확인

대기 중인 클러스터 유지 관리가 있는지 확인하여 최신 Amazon DocumentDB 엔진 버전을 사용 중인지 확인할 수 있습니다.

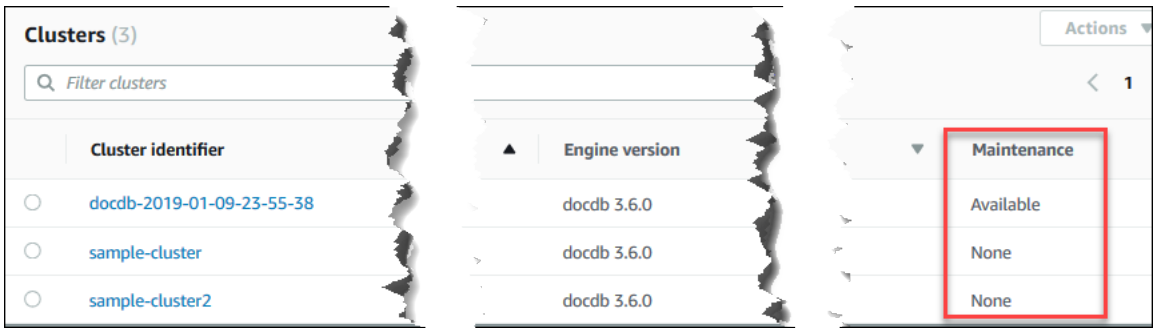
Using the AWS Management Console

AWS Management Console 를 사용하여 클러스터에 보류 중인 유지 관리가 있는지 확인할 수 있습니다.

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 클러스터를 선택합니다.

   > **Tip**
   > 화면 왼쪽에 탐색 창이 표시되지 않으면 페이지 왼쪽 상단 모서리에서 메뉴 아이콘 (≡) 을 선택합니다.

3. 유지 관리 열을 찾아 클러스터에 대기 중인 유지 관리가 있는지 확인합니다.



없음은 클러스터가 최신 엔진 버전을 실행 중임을 나타냅니다. 사용 가능은 클러스터에 대기 중인 유지 관리가 있음을 나타냅니다. 즉, 엔진 업그레이드가 필요할 수도 있습니다.

4. 클러스터에 대기 중인 유지 관리가 있을 경우 클러스터의 엔진 버전에 대한 패치 업데이트 수행의 단계를 진행합니다.

Using the AWS CLI

AWS CLI 를 사용하여 다음 파라미터와 함께 `describe-pending-maintenance-actions` 작업을 사용하여 클러스터에 최신 엔진 버전이 있는지 확인할 수 있습니다.

파라미터

- **`--resource-identifier`** - 선택 사항. 리소스(클러스터)의 ARN입니다. 이 파라미터를 생략하면 모든 클러스터의 대기 중인 유지 관리 작업이 나열됩니다.
- **`--region`** - 선택 사항. 이 작업을 실행하려는 AWS 리전입니다. 예: `us-east-1`.

Example

Linux, macOS, Unix의 경우:

```
aws docdb describe-pending-maintenance-actions \
   --resource-identifier arn:aws:rds:us-east-1:123456789012:cluster:sample-cluster \
   --region us-east-1
```

Windows의 경우:

```
aws docdb describe-pending-maintenance-actions ^
   --resource-identifier arn:aws:rds:us-east-1:123456789012:cluster:sample-cluster ^
   --region us-east-1
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
{
    "PendingMaintenanceActions": [
        {
            "ResourceIdentifier": "arn:aws:rds:us-
east-1:123456789012:cluster:sample-cluster",
            "PendingMaintenanceActionDetails": [
                {
                    "Description": "New feature",
                    "Action": "db-upgrade",
                    "ForcedApplyDate": "2019-02-25T21:46:00Z",
                    "AutoAppliedAfterDate": "2019-02-25T07:41:00Z",
                    "CurrentApplyDate": "2019-02-25T07:41:00Z"
                }
            ]
        }
    ]
}
```

클러스터에 대기 중인 유지 관리가 있을 경우 클러스터의 엔진 버전에 대한 패치 업데이트 수행의 단계를 진행합니다.

클러스터의 엔진 버전에 대한 패치 업데이트 수행

이 섹션에서는 AWS Management Console 또는를 사용하여 패치 업데이트를 배포하는 방법을 설명합니다 AWS CLI. 패치 업데이트는 동일한 엔진 버전 내의 업데이트입니다(예: 3.6 엔진 버전을 최신 3.6 엔진 버전으로 업데이트). 즉시 업데이트하거나 클러스터의 다음 유지 관리 기간에 업그레이드할 수 있습니다. 엔진 업데이트가 필요한지 확인하려면 대기 중인 유지 관리 확인 섹션을 참조하세요. 업데이트를 적용할 때 클러스터에 약간의 가동 중지 시간이 발생할 수 있다는 점에 유의하세요.

> **Note**
>
> 메이저 엔진 버전에서 다른 버전으로 업그레이드(예: 3.6에서 5.0으로)하려는 경우 Amazon DocumentDB 인플레이스 주요 버전 업그레이드 또는 를 사용하여 Amazon DocumentDB 클러스터 업그레이드 AWS Database Migration Service을(를) 참조하세요. 전체 메이저 버전 업그레이드는 docdb 5.0만 대상 엔진 버전으로 지원합니다.

클러스터의 엔진 버전에 대한 최신 패치 업데이트를 받으려면 다음과 같은 두 가지 구성 요구 사항이 있습니다.

- 클러스터의 상태가 사용 가능이어야 합니다.
- 클러스터가 이전 엔진 버전을 실행하고 있어야 합니다.

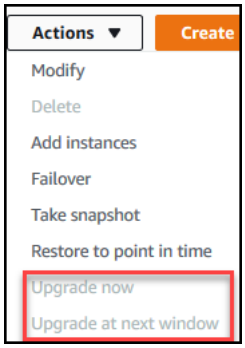
Using the AWS Management Console

다음 절차는 콘솔을 사용하여 클러스터의 엔진 버전에 패치 업데이트를 적용하는 것입니다. 즉시 업데이트하거나 클러스터의 다음 유지 관리 기간 중 업그레이드할 수 있습니다.

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 클러스터를 선택합니다. 클러스터 목록에서 업그레이드할 클러스터 왼쪽에 있는 버튼을 선택합니다. 클러스터의 상태가 사용 가능이어야 합니다.

   > **Tip**
   >
   > 화면 왼쪽에 탐색 창이 표시되지 않으면 페이지 왼쪽 상단 모서리에서 메뉴 아이콘 (≡) 을 선택합니다.

3. 작업 메뉴에서 다음 옵션 중 하나를 선택합니다. 이러한 메뉴 옵션은 선택한 클러스터가 최신 엔진 버전을 실행하고 있지 않은 경우에만 선택할 수 있습니다.



   - 지금 업그레이드 - 업그레이드 프로세스를 즉시 시작합니다. 클러스터가 최신 엔진 버전으로 업그레이드하는 동안에는 클러스터가 잠시 오프라인 상태가 됩니다.
   - 다음에 업그레이드 - 클러스터의 다음 유지 관리 기간 중에 업그레이드 프로세스를 시작합니다. 클러스터가 최신 엔진 버전으로 업그레이드되는 동안에는 클러스터가 잠시 오프라인 상태가 됩니다.

4. 확인 창이 열리면 다음 중 하나를 선택합니다.
   - 업그레이드 - 이전 단계에서 선택한 일정에 따라 클러스터를 최신 엔진 버전으로 업그레이드하려면 선택합니다.
   - 취소 - 클러스터의 엔진 업그레이드를 취소하고 클러스터의 현재 엔진 버전으로 계속하려면 선택합니다.

Using the AWS CLI

다음 파라미터와 함께 AWS CLI 및 `apply-pending-maintenance-action` 작업을 사용하여 클러스터에 패치 업데이트를 적용할 수 있습니다.

파라미터

- **`--resource-identifier`** - 필수입니다. 업그레이드할 Amazon DocumentDB 클러스터의 ARN입니다.
- **`--apply-action`** - 필수입니다. 다음과 같은 값이 허용됩니다. 클러스터 엔진 버전을 업그레이드하려면 `db-upgrade`를 사용합니다.
  - **`db-upgrade`**
  - **`system-update`**
- **`--opt-in-type`** - 필수입니다. 다음과 같은 값이 허용됩니다.
  - `immediate` - 유지 관리 작업을 즉시 적용합니다.
  - `next-maintenance` - 다음 유지 관리 기간 중에 유지 관리 작업을 적용합니다.
  - `undo-opt-in` - 기존 `next-maintenance` 옵트인 요청을 취소합니다.

Example

다음은 `sample-cluster`의 엔진 버전을 버전 4.0.0으로 패치 업데이트하는 예입니다.

Linux, macOS, Unix의 경우:

```
aws docdb apply-pending-maintenance-action \
   --resource-identifier arn:aws:rds:us-east-1:123456789012\:cluster:sample-cluster
 \
   --apply-action db-upgrade \
   --opt-in-type immediate
```

Windows의 경우:

```
aws docdb apply-pending-maintenance-action ^
   --resource-identifier arn:aws:rds:us-east-1:123456789012:cluster:sample-cluster ^
   --apply-action db-upgrade ^
   --opt-in-type immediate
```

이 작업의 출력은 다음과 같습니다.

```
{
    "ResourcePendingMaintenanceActions": {
        "ResourceIdentifier": "arn:aws:rds:us-
east-1:444455556666:cluster:docdb-2019-01-09-23-55-38",
        "PendingMaintenanceActionDetails": [
            {
                "CurrentApplyDate": "2019-02-20T20:57:06.904Z",
                "Description": "Bug fixes",
                "ForcedApplyDate": "2019-02-25T21:46:00Z",
                "OptInStatus": "immediate",
                "Action": "db-upgrade",
                "AutoAppliedAfterDate": "2019-02-25T07:41:00Z"
            }
        ]
    }
}
```

Amazon DocumentDB 클러스터 중지 및 시작

Amazon DocumentDB 클러스터를 중지하고 시작하면 개발 및 테스트 환경 비용을 관리하는 데 도움이 됩니다. 필요하지 않을 경우 Amazon DocumentDB를 사용할 때마다 클러스터 및 인스턴스를 생성하고 삭제하는 대신 클러스터의 모든 인스턴스를 일시적으로 중지할 수 있습니다. 그런 다음 테스트를 재개할 때 다시 시작할 수 있습니다.

주제

- 클러스터의 중지 및 시작 개요
- 중지된 클러스터에서 수행할 수 있는 작업

클러스터의 중지 및 시작 개요

Amazon DocumentDB 클러스터가 필요하지 않은 기간에는 이 클러스터의 모든 인스턴스를 한번에 중지할 수 있습니다. 그런 다음 사용해야 할 때는 언제든지 클러스터를 다시 시작할 수 있습니다. 시작 및 중지를 사용하면 개발, 테스트 또는 연속 가용성을 필요로 하지 않는 유사한 활동에 사용되는 클러스터의 설정 및 해제 프로세스가 간소화됩니다. 클러스터에 있는 인스턴스 수에 관계없이 단일 작업 AWS CLI 으로 AWS Management Console 또는를 사용하여 클러스터를 중지하고 시작할 수 있습니다.

클러스터가 중지되는 동안 클러스터 스토리지 볼륨은 변경되지 않습니다. 지정된 보존 기간 내에는 스토리지, 수동 스냅샷 및 자동 백업 스토리지에 대한 비용만 청구됩니다. 인스턴스 시간에 대해서는 요금이 부과되지 않습니다 Amazon DocumentDB는 필요한 유지 관리 업데이트에 뒤처지지 않도록 7일 후에 클러스터를 자동으로 시작합니다. 7일 후에 클러스터가 시작되면 클러스터의 인스턴스 비용이 다시 청구됩니다. 클러스터가 중지된 동안에는 쿼리하려면 인스턴스가 사용 가능한 상태여야 하므로 스토리지 볼륨을 쿼리할 수 없습니다.

Amazon DocumentDB 클러스터가 중지되면 클러스터도 인스턴스도 어떤 식으로든 수정할 수 없습니다. 여기에는 인스턴스 추가 또는 제거, 클러스터 삭제도 포함됩니다.

Using the AWS Management Console

다음 절차에서는 사용 가능한 상태인 하나 이상의 인스턴스가 있는 클러스터를 중지하거나, 중지된 클러스터를 시작하는 방법을 보여 줍니다.

Amazon DocumentDB 클러스터 중지 및 시작 방법

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 클러스터를 선택합니다.

   > **Tip**
   >
   > 화면 왼쪽에 탐색 창이 표시되지 않으면 페이지 왼쪽 상단 모서리에서 메뉴 아이콘 (≡) 을 선택합니다.

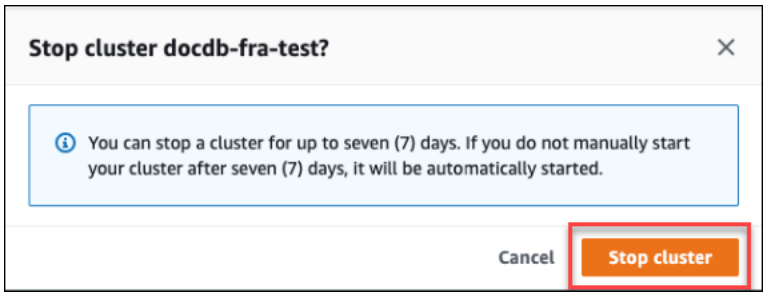
3. 클러스터 목록에서 중지하거나 시작하려는 클러스터 이름의 왼쪽에 있는 버튼을 선택합니다.
4. 작업을 선택한 다음 클러스터에서 수행하려는 작업을 선택합니다.
   - 클러스터를 중지하고 다음 작업을 수행합니다.

     a. 중지를 선택합니다.

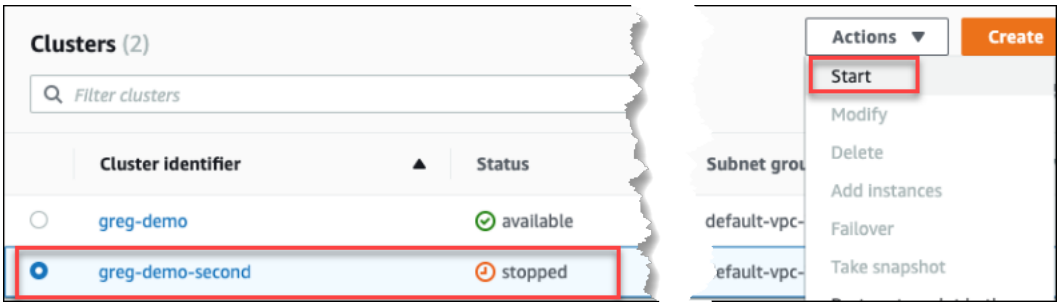


중지 작업은 장애 조치 메커니즘 활성화를 피하기 위해 복제본 인스턴스를 먼저 중지한 후 기본 인스턴스를 중지합니다.

b. 확인 대화 상자에서 클러스터 중지를 선택하여 클러스터 중지를 확인하거나, 클러스터를 계속 실행하려면 취소를 선택합니다.



- 클러스터를 시작하려면 클러스터가 중지된 상태에서 시작을 선택합니다.



5. 클러스터 및 인스턴스 상태를 모니터링합니다. 클러스터를 시작한 경우 클러스터 및 해당 인스턴스를 사용 가능할 때 클러스터 사용을 다시 시작할 수 있습니다. 자세한 내용은 클러스터 상태 확인 섹션을 참조하십시오.



Using the AWS CLI

다음 코드 예제에서는 사용 가능한 상태인 하나 이상의 인스턴스가 있는 클러스터를 중지하거나, 중지된 클러스터를 시작하는 방법을 보여 줍니다.

를 사용하여 사용 가능한 인스턴스가 하나 이상인 클러스터를 중지하려면 `stop-db-cluster` 작업을 AWS CLI사용합니다. 중지된 클러스터를 시작하려면 `start-db-cluster` 작업을 사용합니다. 두 작업은 `--db-cluster-identifier` 파라미터를 사용합니다.

파라미터:

- **`--db-cluster-identifier`** - 필수입니다. 중지하거나 시작할 클러스터의 이름.

Example -를 사용하여 클러스터를 중지하려면 AWS CLI

다음 코드는 클러스터 `sample-cluster`를 중지합니다. 클러스터에는 사용 가능한 상태인 인스턴스가 하나 이상 있어야 합니다.

Linux, macOS, Unix의 경우:

```
aws docdb stop-db-cluster \
   --db-cluster-identifier sample-cluster
```

Windows의 경우:

```
aws docdb stop-db-cluster ^
   --db-cluster-identifier sample-cluster
```

Example -를 사용하여 클러스터를 시작하려면 AWS CLI

다음 코드는 클러스터 `sample-cluster`를 시작합니다. 클러스터는 현재 중지된 상태여야 합니다.

Linux, macOS, Unix의 경우:

```
aws docdb start-db-cluster \
   --db-cluster-identifier sample-cluster
```

Windows의 경우:

```
aws docdb start-db-cluster ^
   --db-cluster-identifier sample-cluster
```

중지된 클러스터에서 수행할 수 있는 작업

Amazon DocumentDB 클러스터가 중지되어 있는 동안, 지정된 자동 백업 보존 기간 내에서 원하는 시점으로 특정 시점 복원을 수행할 수 있습니다. 특정 시점으로 복원에 대한 세부 정보는 특정 시점으로 복원 섹션을 참조하세요.

클러스터가 중지되는 동안 Amazon DocumentDB 클러스터의 구성 또는 해당 인스턴스를 수정할 수 없습니다. 또한 클러스터에서 인스턴스를 추가 또는 제거할 수 없으며, 연결된 인스턴스가 있는 경우에도 클러스터를 삭제할 수 없습니다. 이러한 관리 작업을 수행하기 전에 클러스터를 시작해야 합니다.

Amazon DocumentDB는 다시 시작한 후에만 중지된 클러스터에 예약 유지 관리를 적용합니다. 7일 후에 Amazon DocumentDB는 중지된 클러스터를 자동으로 시작하므로 유지 관리 상태에서 너무 늦어지지 않습니다. 클러스터가 다시 시작되면 클러스터의 인스턴스 비용이 다시 청구됩니다.

클러스터가 중지되는 동안 Amazon DocumentDB는 자동화된 백업을 수행하지 않으며 백업 보존 기간도 연장하지 않습니다.

Amazon DocumentDB 클러스터 삭제

AWS Management Console 또는를 사용하여 Amazon DocumentDB 클러스터를 삭제할 수 있습니다 AWS CLI. 클러스터를 삭제하려면 클러스터가 사용 가능한 상태여야 하며 클러스터와 연결된 인스턴스가 없어야 합니다. 클러스터가 중지된 경우, 먼저 클러스터를 시작하고 클러스터가 사용 가능하게 될 때까지 기다린 다음 클러스터를 삭제합니다. 자세한 내용은 Amazon DocumentDB 클러스터 중지 및 시작 섹션을 참조하십시오.

삭제 방지

실수로 인한 삭제로부터 클러스터를 보호하려면 삭제 방지를 활성화할 수 있습니다. 콘솔을 사용하여 클러스터를 생성할 때 기본적으로 삭제 방지가 활성화됩니다. 하지만 AWS CLI를 사용하여 클러스터를 생성하는 경우 삭제 방지가 기본적으로 비활성화됩니다.

Amazon DocumentDB는 콘솔 또는 AWS CLI중 어느 곳에서 삭제 작업을 수행하든 간에 클러스터에 대한 삭제 방지를 강제 시행합니다. 삭제 방지가 활성화되어 있으면 클러스터를 삭제할 수 없습니다. 삭제 방지가 활성화된 클러스터를 삭제하려면 먼저 클러스터를 수정하고 삭제 방지를 비활성화해야 합니다.

삭제 방지가 활성화된 클러스터가 있는 콘솔을 사용할 경우 클러스터의 마지막 인스턴스를 삭제하면 클러스터도 삭제되므로 해당 인스턴스를 삭제할 수 없습니다. AWS CLI를 사용하여 삭제 방지된 클러스터의 마지막 인스턴스를 삭제할 수 있습니다. 하지만 클러스터 자체는 여전히 존재하고 데이터는 보

존됩니다. 클러스터의 새 인스턴스를 생성하여 데이터에 액세스할 수 있습니다. 삭제 방지 활성화 및 비활성화에 대한 자세한 내용은 다음을 참조하세요.

- Amazon DocumentDB 클러스터 생성
- Amazon DocumentDB 클러스터 수정

Using the AWS Management Console

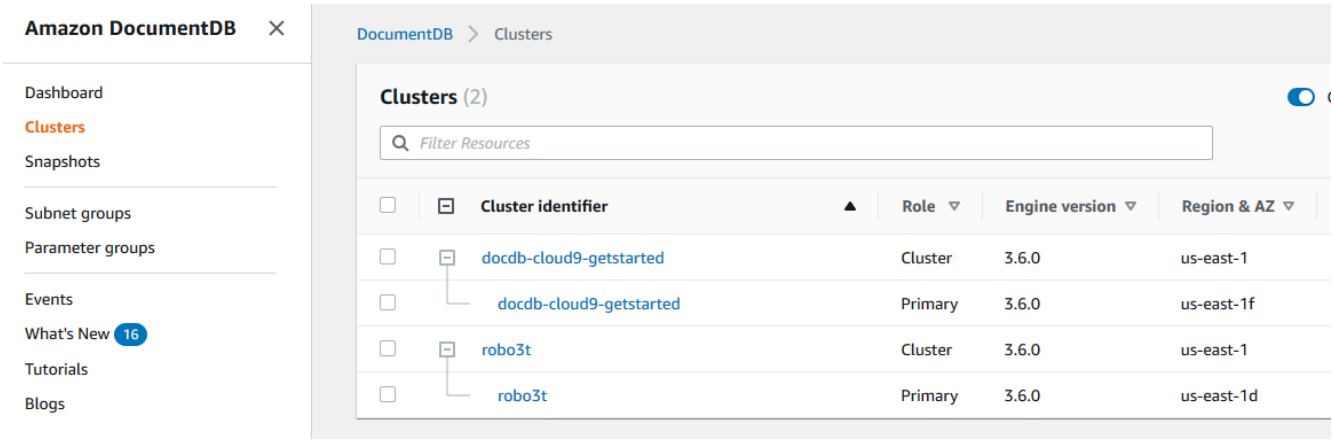
를 사용하여 클러스터를 삭제하려면 AWS Management Console삭제 방지 기능을 비활성화해야 합니다.

클러스터에 대해 삭제 방지를 활성화했는지 여부를 확인하려면 다음과 같이 하세요.

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 클러스터를 선택합니다.

> **Tip**
> 화면 왼쪽에 탐색 창이 표시되지 않으면 페이지 왼쪽 상단 모서리에서 메뉴 아이콘 (≡) 을 선택합니다.

3. 클러스터 탐색 상자에서 클러스터 식별자 열에는 클러스터와 인스턴스가 모두 표시됩니다. 인스턴스는 아래 스크린샷과 마찬가지로 클러스터 아래에 나열됩니다.



4. 클러스터의 이름을 선택하고 구성 탭을 선택합니다. 클러스터 세부 정보 섹션에서 삭제 보호를 찾습니다. 삭제 방지가 활성화된 경우 클러스터를 수정하여 삭제 방지를 비활성화합니다. 클러스터 수정에 대한 자세한 내용은 Amazon DocumentDB 클러스터 수정 섹션을 참조하세요.

삭제 방지가 비활성화된 후에는 클러스터를 삭제할 수 있습니다.

클러스터를 삭제하려면 다음과 같이 하세요.

1. 탐색 창에서 클러스터를 선택합니다.
2. 클러스터 식별자 열에서 아래에 나열된 인스턴스를 확인하여 클러스터에 인스턴스가 있는지 확인합니다. 클러스터를 삭제하려면 모든 인스턴스를 삭제해야 합니다. 자세한 내용은 Amazon DocumentDB 인스턴스 삭제 섹션을 참조하십시오.
3. 클러스터에 인스턴스가 있는지 여부에 따라 다음 단계 중 하나를 수행합니다.
   - 클러스터에 인스턴스가 없는 경우 클러스터 이름 왼쪽에 있는 단추를 선택하고 작업을 선택합니다. 드롭다운 메뉴에서 삭제를 선택합니다. <cluster-name> 삭제 대화 상자를 완료한 다음 삭제를 선택합니다.
   - 클러스터에 인스턴스가 하나 이상 있는 경우 다음을 수행합니다.
     a. 탐색 창에서 클러스터를 선택합니다.
     b. 클러스터 이름 왼쪽에 있는 확인란을 선택하여 클러스터의 각 인스턴스를 삭제합니다. 작업을 선택한 후 삭제를 선택합니다. <cluster-name> 삭제 대화 상자를 완료한 다음 삭제를 선택합니다.

        마지막 인스턴스를 삭제하면 클러스터도 삭제됩니다. 인스턴스 삭제에 대한 자세한 내용은 섹션을 참조하세요Amazon DocumentDB 인스턴스 삭제.

클러스터를 삭제하는 데 몇 분 정도 걸립니다. 클러스터 상태를 모니터링하려면 Amazon DocumentDB 클러스터 상태 모니터링 섹션을 참조하세요.

Using the AWS CLI

인스턴스가 연결되어 있는 클러스터를 삭제할 수 없습니다. 클러스터와 연결된 인스턴스를 확인하려면 `describe-db-clusters` 명령을 실행하고 클러스터의 모든 인스턴스를 삭제합니다. 그런 다음 필요한 경우 클러스터에서 삭제 방지를 비활성화하고 마지막으로 클러스터를 삭제합니다.

1. 먼저 클러스터의 모든 인스턴스를 삭제합니다.

   삭제해야 할 인스턴스를 결정하려면 다음 명령을 실행합니다.

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
```

```
    --query 'DBClusters[*].
[DBClusterIdentifier,DBClusterMembers[*].DBInstanceIdentifier]'
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
[
    [
        "sample-cluster",
        [
            "sample-instance-1",
            "sample-instance-2"
        ]
    ]
]
```

삭제하려는 클러스터에 인스턴스가 있는 경우 아래와 같이 인스턴스를 삭제합니다.

```
aws docdb delete-db-instance \
    --db-instance-identifier sample-instance
```

2. 다음으로 삭제 방지를 비활성화합니다.

AWS CLI 를 사용하여 클러스터의 모든 인스턴스를 삭제해도 클러스터는 삭제되지 않습니다. 클러스터를 삭제해야 하더라도 삭제 방지가 비활성화된 경우에만 삭제할 수 있습니다.

클러스터에 삭제 방지가 활성화되었는지 확인하려면 다음 명령을 실행합니다.

> **Tip**
>
> 모든 Amazon DocumentDB 클러스터의 삭제 방지 상태를 보려면 `--db-cluster-identifier` 파라미터를 생략합니다.

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].[DBClusterIdentifier,DeletionProtection]'
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
[
    [
        "sample-cluster",
        "true"
    ]
]
```

클러스터에 삭제 방지가 활성화된 경우 클러스터를 수정하여 삭제 방지를 비활성화합니다. 클러스터에서 삭제 방지를 비활성화하려면 다음 명령을 실행합니다.

```
aws docdb modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --no-deletion-protection \
    --apply-immediately
```

3. 마지막으로 클러스터를 삭제합니다.

   삭제 방지가 비활성화된 후에는 클러스터를 삭제할 수 있습니다. 클러스터를 삭제하려면 다음 파라미터와 함께 `delete-db-cluster` 작업을 사용합니다.

   - **`--db-cluster-identifier`** - 필수입니다. 삭제하려는 클러스터의 식별자입니다.
   - **`--final-db-snapshot-identifier`** - 선택 사항. 최종 스냅샷을 원하는 경우 최종 스냅샷의 이름과 함께 이 파라미터를 포함해야 합니다. `--final-db-snapshot-identifier` 또는 `--skip-final-snapshot`를 포함시켜야 합니다.

     명명 제약 조건:

     - 길이는 [1~63] 글자, 숫자 또는 하이픈입니다.
     - 첫 번째 문자는 글자이어야 합니다.
     - 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다.
     - 리전별로 Amazon RDS, Amazon Neptune 및 Amazon DocumentDB의 모든 클러스터 AWS 계정에 대해 고유해야 합니다.
   - **`--skip-final-snapshot`** - 선택 사항. 클러스터를 삭제하기 전에 최종 스냅샷을 생성하지 않으려는 경우에만 이 파라미터를 사용하세요. 기본 설정은 최종 스냅샷을 생성하는 것입니다. `--final-db-snapshot-identifier` 또는 `--skip-final-snapshot`를 포함시켜야 합니다.

다음 AWS CLI 코드는 최종 스냅샷이 `sample-cluster` 있는 클러스터를 삭제합니다. 클러스터와 연결된 인스턴스가 있거나 삭제 방지가 활성화된 경우 작업이 실패합니다.

Example

Linux, macOS, Unix의 경우:

```
aws docdb delete-db-cluster \
    --db-cluster-identifier sample-cluster \
    --final-db-snapshot-identifier sample-cluster-final-snapshot
```

Windows의 경우:

```
aws docdb delete-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --final-db-snapshot-identifier sample-cluster-final-snapshot
```

Example

다음 AWS CLI 코드는 최종 스냅샷을 생성`sample-cluster`하지 않고 클러스터를 삭제합니다.

Linux, macOS, Unix의 경우:

```
aws docdb delete-db-cluster \
    --db-cluster-identifier sample-cluster \
    --skip-final-snapshot
```

Windows의 경우:

```
aws docdb delete-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --skip-final-snapshot
```

`delete-db-cluster` 연산은 삭제하려는 클러스터를 출력합니다.

클러스터를 삭제하는 데 몇 분 정도 걸립니다. 클러스터 상태를 모니터링하려면 클러스터 상태 모니터링 섹션을 참조하세요.

Amazon DocumentDB 클러스터 규모 조정

Amazon DocumentDB를 사용하면 필요에 따라 클러스터의 스토리지와 컴퓨팅을 조정할 수 있습니다. 이 섹션에서는 스토리지 조정, 인스턴스 조정 및 읽기 조정을 사용하여 Amazon DocumentDB 클러스터 및 인스턴스의 성능 및 확장을 관리하는 방법을 설명합니다.

주제

- 스토리지 조정
- 인스턴스 조정
- 읽기 확장
- 쓰기 조정

스토리지 조정

Amazon DocumentDB 스토리지는 클러스터 볼륨에 저장된 데이터에 따라 자동 조정됩니다. 데이터가 증가하면 클러스터 볼륨 스토리지도 최대 128TiB까지 10GiB씩 확장됩니다.

인스턴스 조정

필요에 따라 클러스터의 인스턴스마다 인스턴스 클래스를 수정하여 Amazon DocumentDB 클러스터의 크기를 조정할 수 있습니다. Amazon DocumentDB는 Amazon DocumentDB에 최적화된 여러 인스턴스 클래스를 지원합니다.

자세한 내용은 Amazon DocumentDB 인스턴스 수정 섹션을 참조하십시오.

읽기 확장

클러스터에서 Amazon DocumentDB 복제본을 최대 15개까지 생성하여 Amazon DocumentDB 클러스터에 대한 읽기 확장이 가능합니다. 각 Amazon DocumentDB 복제본은 복제본 지연을 최소화하여 클러스터 볼륨에서 동일한 데이터를 반환합니다. 일반적으로 이 지연 시간은 기본 인스턴스가 업데이트를 적용한 후 100밀리초 미만입니다. 읽기 트래픽이 증가하면 Amazon DocumentDB 복제본을 추가 생성하여 직접 연결함으로써 클러스터의 읽기 부하를 분산시키는 것도 가능합니다. Amazon DocumentDB 복제본의 인스턴스 클래스가 기본 인스턴스의 DB 인스턴스 클래스와 같을 필요는 없습니다.

자세한 내용은 클러스터에 Amazon DocumentDB 인스턴스 추가 섹션을 참조하십시오.

Amazon DocumentDB로 스케일을 읽으려면 드라이버의 내장된 읽기 기본 설정 기능을 사용하여 복제본 세트로 클러스터에 연결하고 읽기를 복제본 인스턴스에 배포하는 것이 좋습니다. 자세한 내용은 Amazon DocumentDB에 복제본 세트로 연결 섹션을 참조하세요.

쓰기 조정

클러스터의 기본 인스턴스 크기를 늘려 Amazon DocumentDB 클러스터의 쓰기 용량을 확장할 수 있습니다. 이 섹션에서는 필요에 따라 클러스터의 기본 인스턴스를 확장하는 두 가지 방법을 설명합니다. 첫 번째 옵션은 애플리케이션 영향을 최소화하려고 하지만 완료하는 데 더 많은 단계가 필요합니다. 두 번째 옵션은 단계가 적어 더 간단하지만 애플리케이션에 더 많은 잠재적 영향을 미칠 수 있다는 단점이 있습니다.

애플리케이션에 따라 둘 중에서 더 적합한 방법을 선택할 수 있습니다. 사용 가능한 인스턴스 크기 및 비용에 대한 자세한 내용은 Amazon DocumentDB 요금 페이지를 참조하세요.

1. 고가용성 및 성능 최적화 - 복제본 집합 모드로 클러스터에 연결하는 경우(권장) 기본 인스턴스를 확장할 때 다음 프로세스를 사용하여 애플리케이션에 미치는 영향을 최소화할 수 있습니다. 이 방법을 사용하면 클러스터가 필요한 고가용성 수준 이상으로 유지되며 읽기 조정 대상이 현재 위치에서 업데이트되지 않고 인스턴스로 클러스터에 추가되므로 영향이 최소화됩니다.

   a. 클러스터에 더 큰 인스턴스 유형의 복제본을 하나 이상 추가합니다(??? 참조). 모든 복제본은 기본 인스턴스와 동일하거나 더 큰 인스턴스 유형인 것이 좋습니다. 이렇게 하면 더 작은 인스턴스 유형으로 장애 조치되어 쓰기 성능이 의도치 않게 저하되는 것을 방지할 수 있습니다. 대부분의 고객의 경우 클러스터의 인스턴스 수를 일시적으로 두 배로 늘린 다음 확장이 완료된 후 더 작은 복제본을 제거해야 합니다.

   b. 모든 새 복제본의 장애 조치 계층을 우선 순위 0으로 설정하여 더 작은 인스턴스 유형의 복제본이 가장 높은 장애 조치 우선 순위를 갖도록 합니다. 자세한 내용은 ??? 섹션을 참조하십시오.

   c. 수동 장애 조치를 시작합니다. 그러면 새 복제본 중 하나가 기본 인스턴스로 승격됩니다. 자세한 내용은 ??? 섹션을 참조하십시오.

      > **Note**
      >
      > 이렇게 하면 클러스터에 약 30초의 가동 중지 시간이 발생합니다. 이에 맞춰 계획을 세우세요.

   d. 새 기본 인스턴스보다 작은 인스턴스 유형의 모든 복제본을 클러스터에서 제거합니다.

e. 모든 인스턴스의 장애 조치 계층을 동일한 우선 순위로 다시 설정합니다(일반적으로 다시 1로 설정됨).

예를 들어 현재 3개의 `r5.large` 인스턴스(하나의 기본 인스턴스와 두 개의 복제본)가 포함된 클러스터가 있고 `r5.xlarge` 인스턴스 유형으로 확장하려는 경우, 먼저 클러스터에 `r5.xlarge` 복제본 인스턴스 3개를 추가하고 새 `r5.xlarge` 복제본의 장애 조치 계층을 0으로 설정한 다음 수동 장애 조치를 시작합니다(애플리케이션에 30초 정도의 가동 중지 시간이 발생함). 장애 조치가 완료된 후 클러스터에서 3개의 `r5.large` 인스턴스를 모두 제거하면 클러스터가 `r5.xlarge` 인스턴스로 확장됩니다.

비용을 최적화할 수 있도록 Amazon DocumentDB 인스턴스 요금은 인스턴스 생성, 수정 또는 삭제 같은 청구 가능한 상태 변경에 따라 1초 단위로 청구되며 최소 10분의 요금이 부과됩니다. 자세한 내용은 모범 사례 설명서의 비용 최적화 섹션을 참조하세요.

2. 단순성을 위한 최적화 - 이 접근 방식은 단순성을 위해 최적화됩니다. 클러스터를 확장하거나 축소하지는 않지만 일시적으로 읽기 용량을 줄일 수 있습니다.

   복제본의 인스턴스 클래스를 변경하면 해당 인스턴스가 몇 초에서 30초 미만으로 짧은 기간 동안 요청을 처리하지 못할 수 있습니다. 복제본 세트 모드(권장)로 클러스터에 연결하는 경우 조정 작업 중에 읽기 용량이 복제본 1개만큼 줄어듭니다(예: 3노드 클러스터의 경우 66%, 4노드 클러스터의 경우 75% 용량 등).

   a. 클러스터의 복제본 인스턴스 중 하나를 확장합니다. 자세한 내용은 인스턴스 클래스 관리 섹션을 참조하십시오.

   b. 인스턴스를 사용할 수 있을 때까지 기다리세요(Amazon DocumentDB 인스턴스 상태 모니터링 참조).

      > **Note**
      >
      > 이렇게 하면 클러스터에 약 30초의 가동 중지 시간이 발생합니다. 이에 맞춰 계획을 세우세요.

   c. 모든 복제본 인스턴스가 하나씩 확장될 때까지 1단계와 2단계를 계속 실행합니다.

   d. 수동 장애 조치 시작. 이렇게 하면 복제본 중 한 개가 기본 인스턴스로 승격됩니다. 자세한 내용은 Amazon DocumentDB 장애 조치 섹션을 참조하십시오.

> **Note**
>
> 이로 인해 클러스터에 최대 30초의 가동 중지 시간이 발생하지만 대개 그보다 시간이 덜 걸립니다. 이에 맞춰 계획을 세우세요.

e. 이전의 기본(지금은 복제본) 인스턴스를 확장하세요.

Amazon DocumentDB 클러스터에 대한 볼륨 복제

Amazon DocumentDB 복제를 사용하면 동일한 Amazon DocumentDB 클러스터 볼륨을 사용하고 원본과 동일한 데이터를 갖는 새 클러스터를 생성할 수 있습니다. 이 프로세스는 빠르고 비용 효율적으로 진행되도록 설계되었습니다. 연결된 데이터 볼륨이 있는 새 클러스터를 복제본이라고 합니다. 복제본 생성은 스냅샷 복원과 같은 다른 기술을 사용하여 데이터를 물리적으로 복사하는 것보다 빠르고 공간 효율적입니다.

Amazon DocumentDB는 프로비저닝된 Amazon DocumentDB 클러스터에서 Amazon DocumentDB 프로비저닝된 복제본을 생성할 수 있도록 지원합니다. 소스와 다른 배포 구성을 사용하여 복제를 생성하면 소스 Amazon DocumentDB 엔진의 최신 버전을 사용하여 복제본이 생성됩니다.

Amazon DocumentDB 클러스터에서 복제본을 생성하면 소스 Amazon DocumentDB 클러스터를 소유한 것과 동일한 계정인 AWS 계정에 복제본이 생성됩니다.

주제

- Amazon DocumentDB 복제 개요
- Amazon DocumentDB 복제의 제한
- Amazon DocumentDB 복제의 작동 방식
- Amazon DocumentDB 복제본 생성

Amazon DocumentDB 복제 개요

Amazon DocumentDB는 기록 중 복사(Copy-on-Write) 프로토콜을 사용하여 복제본을 생성합니다. 이 메커니즘은 최소한의 추가 공간을 사용하여 초기 복제를 만듭니다. 복제가 처음 생성되면 Amazon DocumentDB는 소스 DB 클러스터와 새로운(복제된) Amazon DocumentDB 클러스터에서 사용하는 데이터의 단일 복사본을 유지합니다. 소스 Amazon DocumentDB 클러스터 또는 Amazon DocumentDB 클러스터 복제가 Amazon DocumentDB 스토리지 볼륨의 데이터를 변경한 경우에만

추가 스토리지가 할당됩니다. 기록 중 복사(Copy-on-Write) 프로토콜에 대한 자세한 내용은 Amazon DocumentDB 복제의 작동 방식 섹션을 참조하세요.

Amazon DocumentDB 복제 작업은 데이터 손상 위험 없이 프로덕션 데이터를 사용하여 테스트 환경을 신속하게 설정하는 데 특히 유용합니다. 다음과 같은 여러 유형의 애플리케이션에 복제본을 사용할 수 있습니다.

- 잠재적 변경 사항(예: 스키마 변경 및 파라미터 그룹 변경)을 실험하여 모든 영향을 평가합니다.
- 데이터 내보내기 또는 복제본에서 분석 쿼리 실행과 같은 워크로드 집약적인 작업을 수행하는 경우
- 개발, 테스트 또는 기타 용도로 프로덕션 DB 클러스터의 복사본을 생성합니다.

동일한 Amazon DocumentDB 클러스터에서 둘 이상의 복제본을 생성할 수 있습니다. 다른 복제본에서 여러 복제본을 생성할 수도 있습니다.

Amazon DocumentDB 복제본을 생성한 후 Amazon DocumentDB 인스턴스를 원본 Amazon DocumentDB 클러스터와 다르게 구성할 수 있습니다. 예를 들어 소스 프로덕션 Amazon DocumentDB 클러스터와 동일한 고가용성 요구 사항을 충족하기 위해 개발 목적으로 복제본이 필요하지 않을 수 있습니다. 이 경우 Amazon DocumentDB 클러스터에서 사용하는 여러 DB 인스턴스가 아닌 단일 Amazon DocumentDB 인스턴스로 복제본을 구성할 수 있습니다.

복제본을 테스트, 개발 또는 다른 용도로 사용한 후 삭제할 수 있습니다.

Amazon DocumentDB 복제의 제한

Amazon DocumentDB 복제는 다음과 같은 제한 사항이 있습니다.

- AWS 리전에서 허용하는 최대 DB 클러스터 개수까지 원하는 만큼 복제본을 생성할 수 있습니다. 그러나 15개의 복제본을 생성한 후에는 다음 복제가 전체 복제본이 됩니다. 시점 복구와 같은 복제 작업이 수행됩니다.
- 소스 Amazon DocumentDB 클러스터와 다른 AWS 리전에서는 복제본을 생성할 수 없습니다.
- DB 인스턴스가 없는 Amazon DocumentDB 클러스터에서 복제본을 생성할 수 없습니다. 하나 이상의 DB 인스턴스가 있는 Amazon DocumentDB 클러스터만 복제할 수 있습니다.
- 복제본을 Amazon DocumentDB 클러스터의 가상 프라이빗 클라우드(VPC)와 다른 가상 프라이빗 클라우드(VPC)에 생성할 수 있습니다. 이렇게 하면 VPC의 서브넷을 동일한 가용 영역에 매핑해야 합니다.

Amazon DocumentDB 복제의 작동 방식

Amazon DocumentDB 복제는 Amazon DocumentDB 클러스터의 스토리지 계층에서 작동합니다. 이는 Amazon DocumentDB 스토리지 볼륨을 지원하는 기본 내구 미디어 측면에서 빠르고 공간 효율적인 기록 중 복사(copy-on-write) 프로토콜을 사용합니다. Amazon DocumentDB 클러스터 볼륨에 대한 자세한 내용은 Amazon DocumentDB 클러스터 관리에서 확인할 수 있습니다.

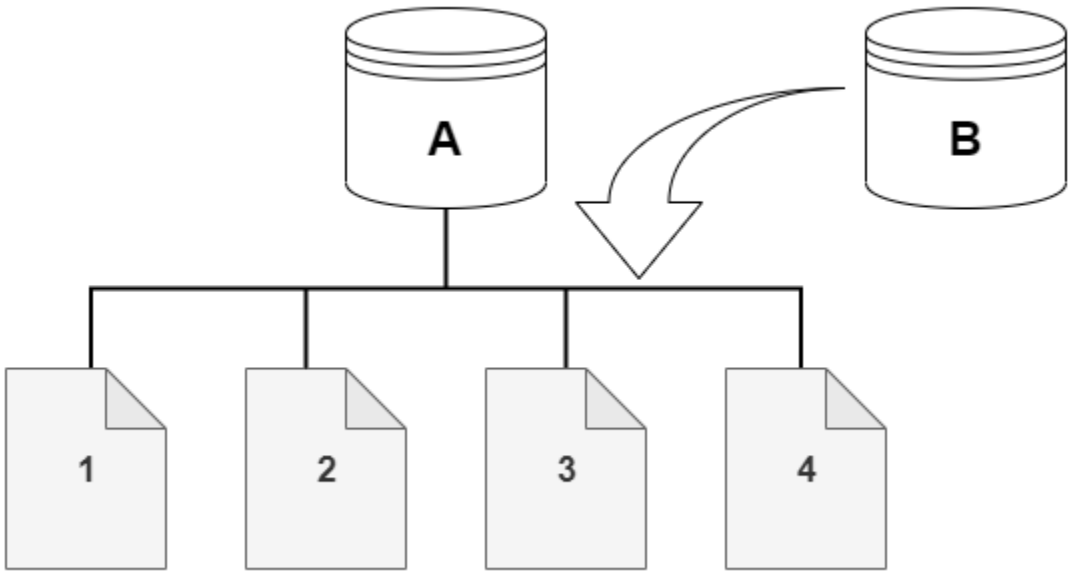
주제

- 기록 중 복사(copy-on-write) 이해
- 원본 클러스터 볼륨 삭제

기록 중 복사(copy-on-write) 이해

Amazon DocumentDB 클러스터는 기본 Amazon DocumentDB 스토리지 볼륨의 페이지에 데이터를 저장합니다.

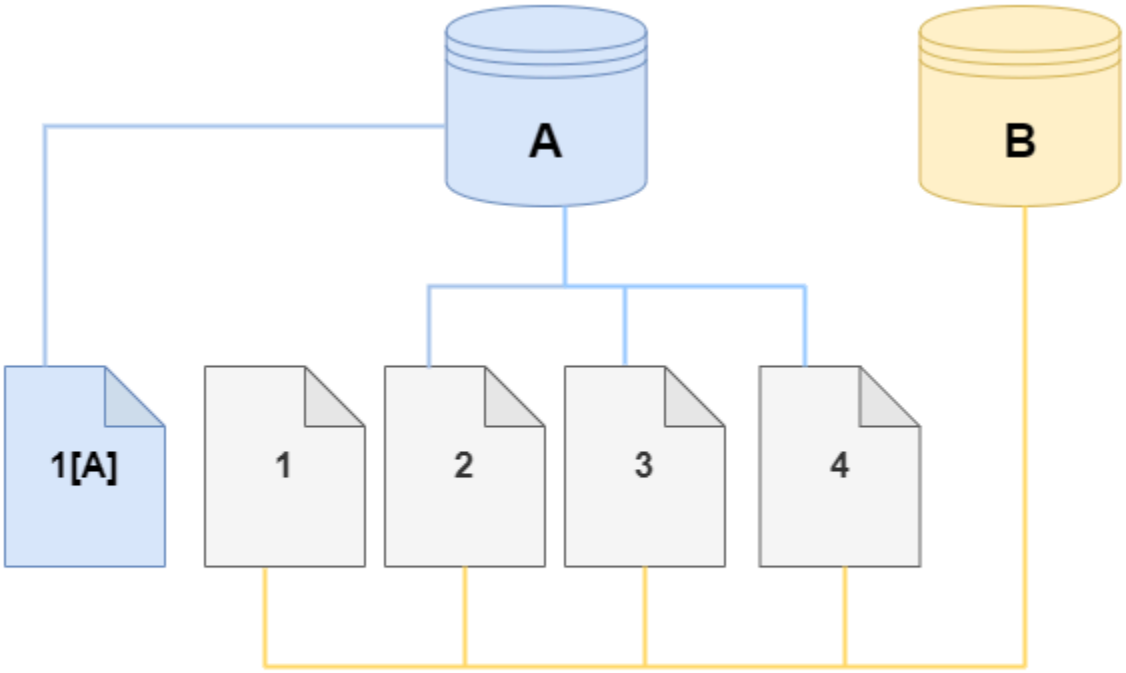
예를 들어, 다음 다이어그램에서 데이터 페이지(1, 2, 3, 4)가 네 개인 Amazon DocumentDB 클러스터(A)를 찾을 수 있습니다. 복제본 B가 Amazon DocumentDB 클러스터에서 생성된다고 가정해 보겠습니다. 복제본이 생성되면 데이터가 복사되지 않습니다. 대신 복제본은 소스 Amazon DocumentDB 클러스터와 동일한 페이지 집합을 가리킵니다.



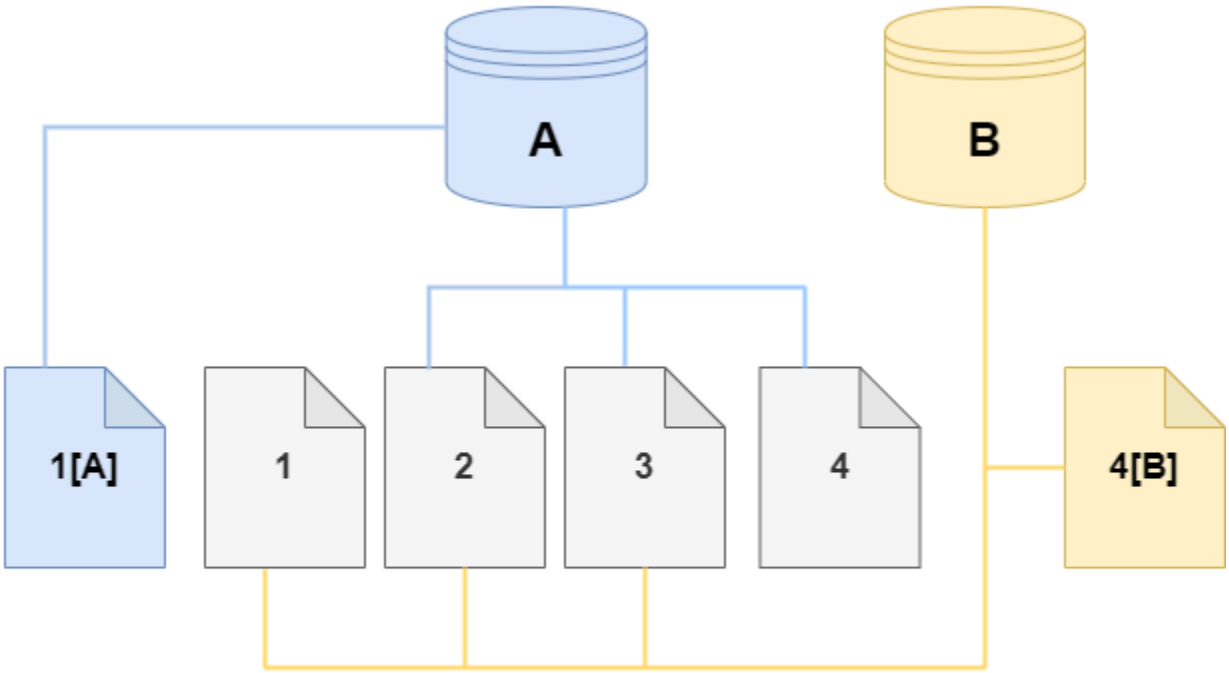
복제본이 생성되면 일반적으로 추가 스토리지가 필요하지 않습니다. 기록 중 복사(copy-on-write) 프로토콜은 물리적 스토리지 미디어에서 소스 세그먼트와 동일한 세그먼트를 사용합니다. 소스 세그먼트

의 용량이 전체 복제본 세그먼트에 충분하지 않은 경우에만 추가 스토리지가 필요합니다. 이 경우 소스 세그먼트는 다른 물리적 디바이스로 복사됩니다.

다음 다이어그램에서는 앞에서와 같이 동일한 클러스터 A와 해당 복제본 B를 사용하여 작동하는 기록 중 복사(copy-on-write) 프로토콜의 예를 찾을 수 있습니다. Amazon DocumentDB 클러스터(A)를 변경하여 페이지 1에 보관된 데이터가 변경된다고 가정해 보겠습니다. Amazon DocumentDB는 원본 페이지 1에 기록하는 대신 새 페이지 1[A]을 생성합니다. 클러스터(A)에 대한 Amazon DocumentDB 클러스터 볼륨은 이제 페이지 1[A], 2, 3, 4를 가리키고 복제본(B)은 여전히 원본 페이지를 참조합니다.



복제본에서 스토리지 볼륨의 페이지 4가 변경됩니다. Amazon DocumentDB는 원본 페이지 4에 기록하는 대신 새 페이지 4[B]를 생성합니다. 이제 복제본은 페이지 1, 2, 3 및 페이지 4[B]를 가리키고 클러스터(A)는 계속해서 1[A], 2, 3 및 4를 가리킵니다.



시간이 경과하여 소스 Amazon DocumentDB 클러스터 볼륨과 복제본 모두가 변경될 경우 변경 사항을 캡처하고 저장하기 위해 점점 더 많은 스토리지가 필요합니다.

원본 클러스터 볼륨 삭제

하나 이상의 복제본이 연결된 소스 클러스터 볼륨을 삭제해도 복제본은 영향을 받지 않습니다. 복제본은 이전에 원본 클러스터 볼륨이 소유하던 페이지를 계속 가리킵니다.

Amazon DocumentDB 복제본 생성

소스 Amazon DocumentDB 클러스터와 동일한 AWS 계정에 복제본을 생성할 수 있습니다. 이렇게 하려면 AWS Management Console 또는 AWS CLI 및 다음 절차를 사용할 수 있습니다.

Amazon DocumentDB 복제를 사용하면 프로비저닝된 Amazon DocumentDB 클러스터에서 프로비저닝된 Amazon DocumentDB 클러스터 복제본을 생성할 수 있습니다.

Using the AWS Management Console

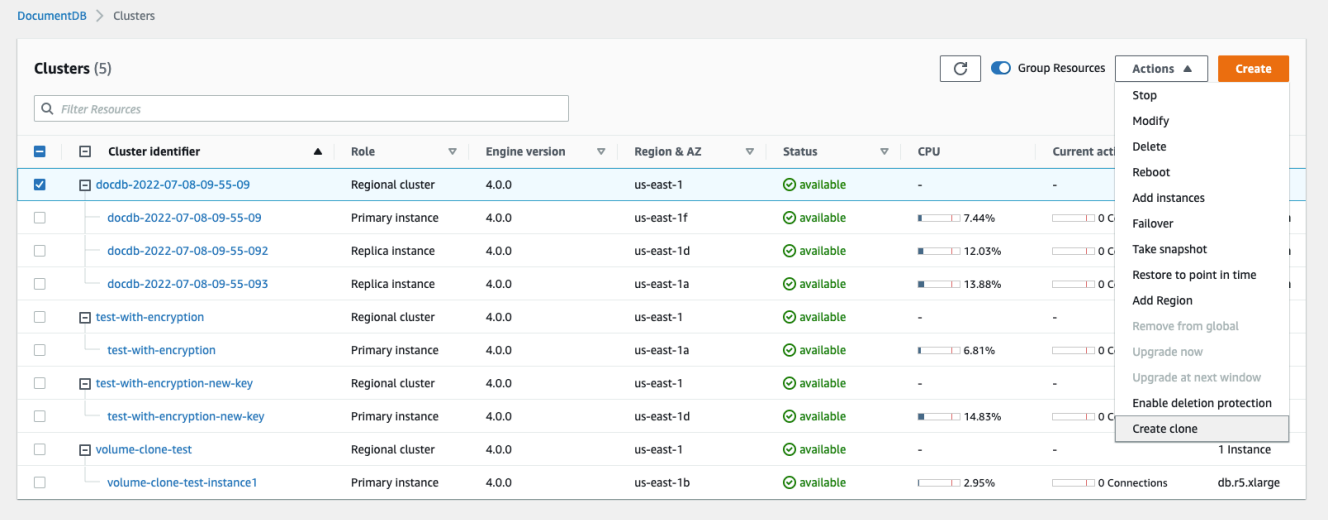
다음 절차에서는 AWS Management Console을 사용하여 Amazon DocumentDB 클러스터를 복제하는 방법에 대해 설명합니다.

Amazon DocumentDB 인스턴스 하나가 있는 Amazon DocumentDB 클러스터에서 AWS Management Console 결과를 사용하여 복제본을 생성합니다.

이 지침은 복제본을 생성하는 계정과 동일한 AWS 계정이 소유한 DB 클러스터에 적용됩니다. DB 클러스터는 Amazon DocumentDB에서 교차 AWS 계정 복제가 지원되지 않으므로 동일한 계정이 소유해야 합니다.

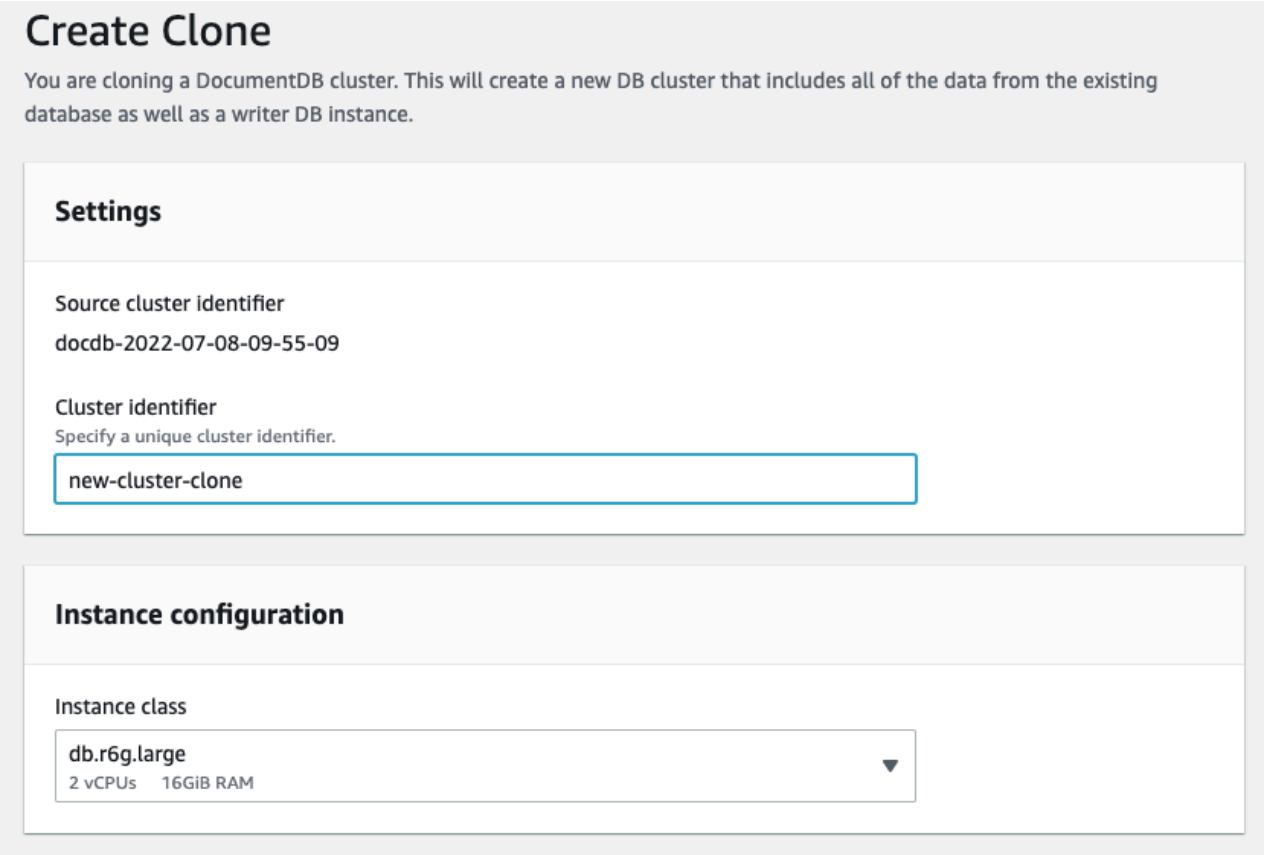
를 사용하여 AWS 계정이 소유한 DB 클러스터의 복제본을 생성하려면 AWS Management Console

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 클러스터를 선택합니다.
3. 목록에서 Amazon DocumentDB 클러스터를 선택하고 작업에서 복제본 생성을 선택합니다.



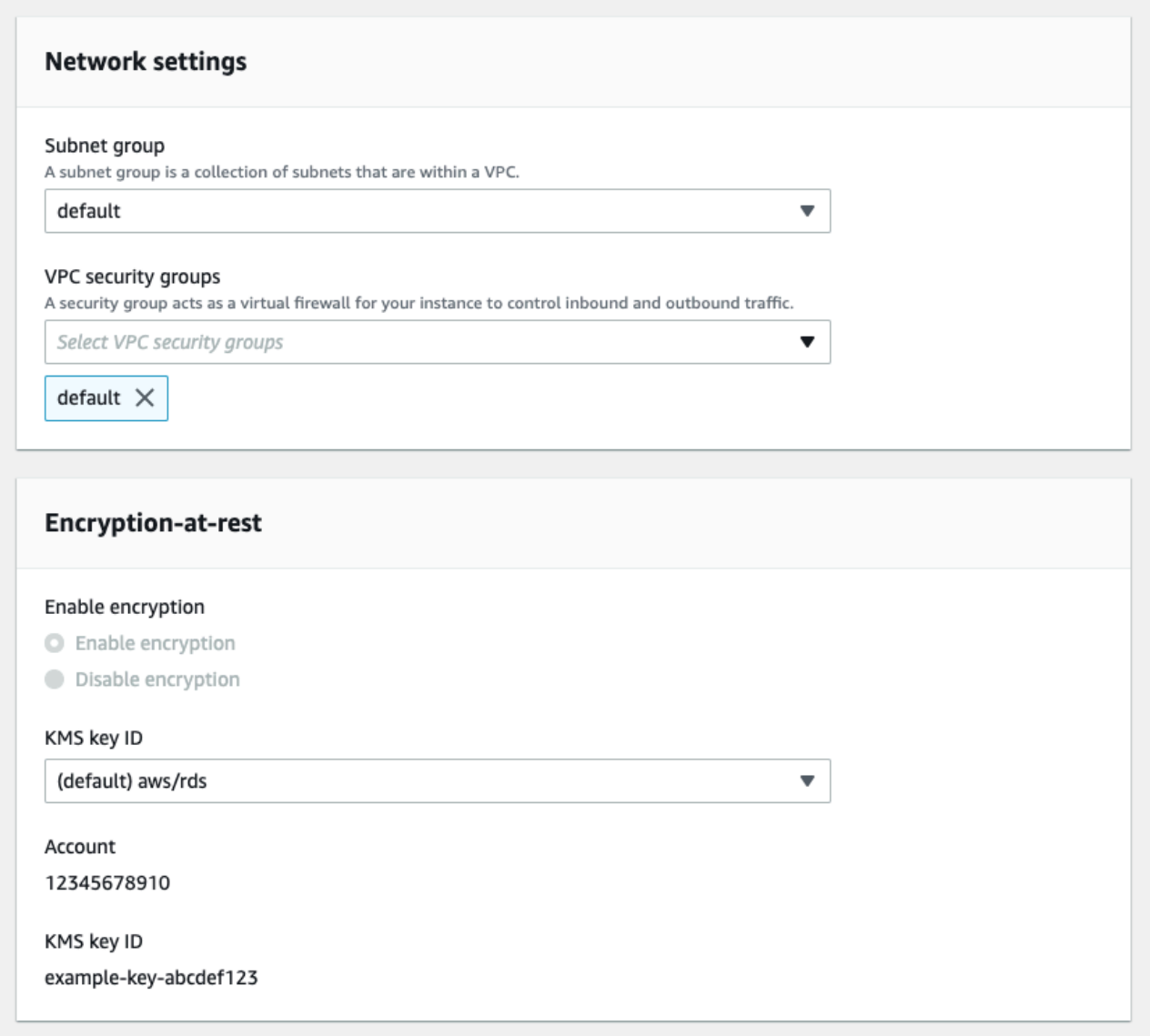
클러스터 식별자, 인스턴스 클래스, 그리고 Amazon DocumentDB 클러스터 복제본에 대한 기타 옵션을 구성할 수 있는 복제본 생성 페이지가 열립니다.

4. 설정 섹션에서 다음을 수행합니다.
   a. 클러스터 식별자에 복제된 Amazon DocumentDB 클러스터에 부여할 이름을 입력합니다.
   b. 인스턴스 구성의 경우 복제된 Amazon DocumentDB 클러스터에 적합한 인스턴스 클래스를 선택합니다.

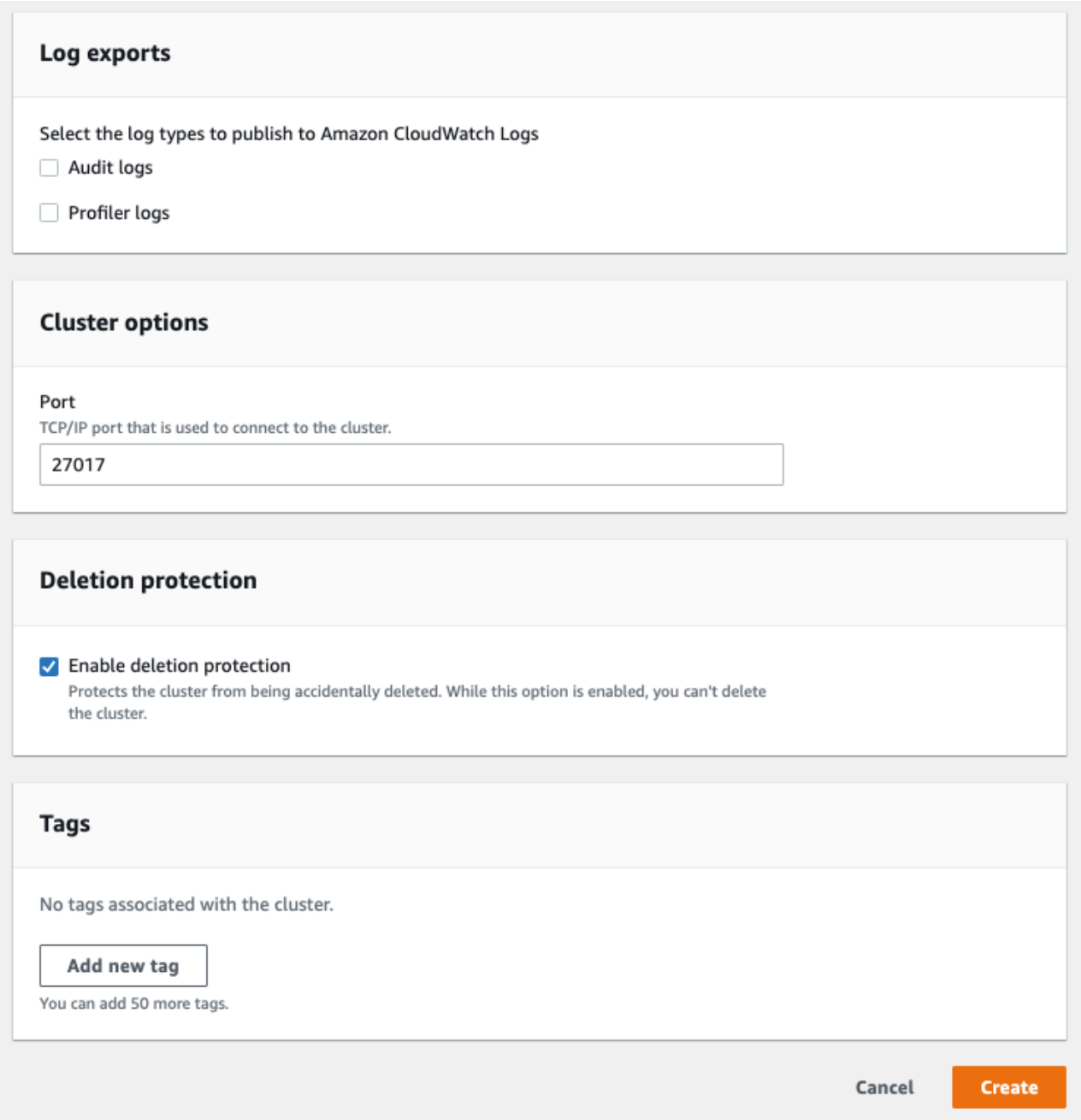


c. 네트워크 설정에서 사용 사례에 맞는 서브넷 그룹과 관련 VPC 보안 그룹을 선택합니다.

d. 저장 시 암호화의 경우 소스 클러스터(복제 중인 클러스터) 에 암호화가 활성화되어 있는 경우 복제된 클러스터에도 암호화가 활성화되어 있어야 합니다. 이 시나리오가 사실인 경우 암호화 사용 옵션은 회색으로 표시(사용 안 함)되지만 암호화 사용 옵션은 선택됩니다. 반대로 소스 클러스터에 암호화가 활성화되어 있지 않은 경우 암호화 활성화 옵션을 사용할 수 있으며 암호화를 활성화하거나 비활성화하도록 선택할 수 있습니다.



e. 내보낼 로그 유형(선택 사항)을 선택하고, 클러스터에 연결하는 데 사용되는 특정 포트를 입력하고, 클러스터를 실수로 삭제하지 않도록 보호(기본적으로 활성화됨)하여 새 클러스터 복제본 구성을 완료합니다.



f. Amazon DocumentDB 클러스터 복제본에 대한 모든 설정을 입력합니다. Amazon DocumentDB 클러스터 및 인스턴스 설정에 대한 자세한 내용은 Amazon DocumentDB 클러스터 관리 섹션에서 참조하세요.

5. 복제본 생성을 선택하여 선택한 Amazon DocumentDB 클러스터의 Amazon DocumentDB 복제본을 시작합니다.

복제본이 생성되면 복제본은 콘솔의 데이터베이스 섹션에 다른 Amazon DocumentDB 클러스터와 함께 나열되고 현재 상태가 표시됩니다. 상태가 사용 가능이면 복제본을 사용할 준비가 된 것입니다.

Using the AWS CLI

AWS CLI 를 사용하여 Amazon DocumentDB 클러스터를 복제하려면 몇 가지 단계가 필요합니다.

사용하는 `restore-db-cluster-to-point-in-time` AWS CLI 명령은 0개의 Amazon DocumentDB 인스턴스가 있는 빈 Amazon DocumentDB 클러스터를 생성합니다. 즉, 명령은 Amazon DocumentDB 클러스터만 복원하고, 해당 클러스터의 DB 인스턴스는 복원하지 않습니다. 복제본이 사용 가능한 후에 별도로 이 작업을 수행합니다. 프로세스의 두 단계는 다음과 같습니다.

1. restore-db-cluster-to-point-in-time CLI 명령을 사용하여 복제본을 생성합니다. 이 명령과 함께 사용하는 파라미터는 생성 중인 빈 Amazon DocumentDB 클러스터(복제본)의 용량 유형 및 기타 세부 정보를 제어합니다.
2. create-db-instance CLI 명령을 사용하여 복제본에 대한 Amazon DocumentDB 인스턴스를 생성하여 복원된 Amazon DocumentDB 클러스터에서 Amazon DocumentDB 인스턴스를 다시 생성합니다.

다음 명령은 AWS CLI 가 AWS 리전을 기본값으로 설정했다고 가정합니다. 이 방식을 사용하면 각 명령에서 `--region` 이름을 전달하지 않아도 됩니다. 자세한 내용은 AWS CLI구성 섹션을 참조하세요. 다음 각 CLI 명령에서 `--region`을 지정할 수도 있습니다.

복제본 생성

`restore-db-cluster-to-point-in-time` CLI 명령으로 전달하는 특정 파라미터는 다양합니다. 전달하는 항목은 생성하려는 복제본 유형에 따라 다릅니다.

다음 절차를 사용하여 프로비저닝된 Amazon DocumentDB 클러스터에서 프로비저닝된 Amazon DocumentDB 복제본을 생성할 수 있습니다.

소스 Amazon DocumentDB 클러스터와 동일한 엔진 모드의 복제본을 생성하려면

- `restore-db-cluster-to-point-in-time` CLI 명령을 사용하여 다음 파라미터에 대한 값을 지정합니다.
  - `--db-cluster-identifier` - 복제본에 대해 의미 있는 이름을 선택합니다. 복제본의 이름을 지정할 때 restore-db-cluster-to-point-in-time CLI 명령을 사용합니다.

- `--restore-type` – `copy-on-write`을(를) 사용하여 소스 DB 클러스터의 복제본을 생성합니다. 이 파라미터가 없으면 `restore-db-cluster-to-point-in-time`은 복제본을 생성하는 대신 Amazon DocumentDB 클러스터를 복원합니다. `restore-type` 기본값은 `full-copy`입니다.
- `--source-db-cluster-identifier` - 복제할 소스 Amazon DocumentDB 클러스터의 이름을 사용합니다.
- `--use-latest-restorable-time` - 이 값은 복제본에 대해 복원 가능한 최신 볼륨 데이터를 가리킵니다. 이 매개변수는 필수 `restore-type copy-on-write` 매개변수이지만 `restore-to-time parameter`를 함께 사용할 수는 없습니다.

다음 예제에서는 `my-source-cluster`라는 이름의 클러스터에서 `my-clone`라는 이름의 복제본을 생성합니다.

Linux, macOS 또는 Unix의 경우:

```
aws docdb restore-db-cluster-to-point-in-time \
    --source-db-cluster-identifier my-source-cluster \
    --db-cluster-identifier my-clone \
    --restore-type copy-on-write \
    --use-latest-restorable-time
```

Windows의 경우:

```
aws docdb restore-db-cluster-to-point-in-time ^
    --source-db-cluster-identifier my-source-cluster ^
    --db-cluster-identifier my-clone ^
    --restore-type copy-on-write ^
    --use-latest-restorable-time
```

이 명령은 복제본의 세부 사항을 포함하는 JSON 객체를 반환합니다. 복제본에 대한 DB 인스턴스를 생성하기 전에 복제된 DB 클러스터를 사용할 수 있는지 확인합니다. 자세한 내용은 아래의 상태 확인 및 복제본 세부 정보 가져오기를 참조하세요.

상태 확인 및 복제본 세부 정보 가져오기

다음 명령을 사용하여 새로 생성된 빈 DB 클러스터의 상태를 확인할 수 있습니다.

```
$ aws docdb describe-db-clusters --db-cluster-identifier my-clone --query '*[].
[Status]' --output text
```

또는 다음 AWS CLI 쿼리를 사용하여 복제본의 DB 인스턴스를 생성하는 데 필요한 상태 및 기타 값을 얻을 수 있습니다.

Linux, macOS, Unix의 경우:

```
aws docdb describe-db-clusters --db-cluster-identifier my-clone \
  --query '*[].{Status:Status,Engine:Engine,EngineVersion:EngineVersion}'
```

Windows의 경우:

```
aws docdb describe-db-clusters --db-cluster-identifier my-clone ^
  --query "*[].{Status:Status,Engine:Engine,EngineVersion:EngineVersion}"
```

이 쿼리는 다음과 비슷한 출력을 반환합니다.

```
[
  {
        "Status": "available",
        "Engine": "docdb",
        "EngineVersion": "4.0.0",
    }
]
```

복제본을 위한 Amazon DocumentDB 인스턴스 생성

create-db-instance CLI 명령을 사용하여 복제본에 대한 DB 인스턴스를 생성하세요.

`--db-instance-class` 파라미터는 프로비저닝된 Amazon DocumentDB 클러스터에만 사용됩니다.

Linux, macOS, Unix의 경우:

```
aws docdb create-db-instance \
    --db-instance-identifier my-new-db \
    --db-cluster-identifier my-clone \
    --db-instance-class  db.r5.4xlarge \
    --engine docdb
```

Windows의 경우:

```
aws docdb create-db-instance ^
```

```
    --db-instance-identifier my-new-db ^
    --db-cluster-identifier my-clone ^
    --db-instance-class  db.r5.4xlarge ^
    --engine docdb
```

복제본에 사용할 파라미터

다음 표에는 `restore-db-cluster-to-point-in-time`에서 Amazon DocumentDB 클러스터를 복제하는 데 사용되는 다양한 파라미터가 요약되어 있습니다.

파라미터	설명
---source-db-cluster-identifier	복제할 소스 Amazon DocumentDB 클러스터의 이름을 사용합니다.
--db-cluster-identifier	복제본에 대해 의미 있는 이름을 선택합니다. `restore-db-cluster-to-point-in-time` 명령으로 복제본의 이름을 지정합니다. 그런 다음 이 이름을 `create-db-instance` 명령으로 전달합니다.
--restore-type	`copy-on-write` 를 `--restore-type` 로 지정하여 소스 Amazon DocumentDB 클러스터를 복원하는 대신 소스 DB 클러스터의 복제본을 생성합니다.
--use-latest-restorable-time	이 값은 복제본에 대해 복원 가능한 최신 볼륨 데이터를 가리킵니다.

Amazon DocumentDB 클러스터 내결함성에 대한 이해

Amazon DocumentDB 클러스터는 내결함성을 고려하여 설계되었습니다. 각 클러스터의 볼륨은 단일의 여러 가용 영역에 걸쳐 있으며 AWS 리전각 가용 영역에는 클러스터의 볼륨 데이터 사본이 포함됩니다. 이 기능은 가용 영역 한 곳에서 결함이 발생하더라도 클러스터가 잠시 서비스가 중단될 뿐 전혀 데이터 손실 없이 결함을 견딜 수 있음을 의미합니다.

클러스터의 기본 인스턴스에 결함이 발생하면 Amazon DocumentDB가 다음 두 가지 방법 중 하나를 사용하여 자동으로 새 기본 인스턴스로 장애 조치를 수행합니다.

- 기존 Amazon DocumentDB 복제본을 각 복제본의 프로모션 티어 설정에 따라 선택된 새 기본 인스턴스로 승격한 다음 이전 기본 인스턴스의 대체 인스턴스를 생성합니다. 복제본 인스턴스로의 페일

오버는 보통 30초도 걸리지 않습니다. 이 기간 동안에는 읽기 및 쓰기 작업이 잠시 중단될 수 있습니다. 클러스터의 가용성을 높이려면 최소 하나 이상의 Amazon DocumentDB 복제본을 둘 이상의 서로 다른 가용 영역에서 생성하는 것이 좋습니다.

- 새로운 기본 인스턴스를 생성합니다. 이는 클러스터에 복제 인스턴스가 없는 경우에만 발생하며 완료하는 데 몇 분 정도 걸릴 수 있습니다.

클러스터에 Amazon DocumentDB 복제본이 하나 이상인 경우에는 장애가 발생하더라도 Amazon DocumentDB 복제본이 기본 인스턴스로 승격됩니다. 이 실패 이벤트로 인해 예외적으로 실패하는 읽기 및 쓰기 작업 동안 짧은 중단이 발생합니다. 하지만, 일반적인 서비스 복구 시간은 120초 미만이지만 대부분 60초 미만에 복원됩니다. 클러스터의 가용성을 높이려면 최소 하나 이상의 Amazon DocumentDB 복제본을 둘 이상의 서로 다른 가용 영역에서 생성하는 것이 좋습니다.

각 복제본에 우선 순위를 지정하여 장애 이후 기본 인스턴스로 승격할 Amazon DocumentDB 복제본 순서를 사용자 지정할 수 있습니다. 우선 순위 범위는 가장 높은 값인 0부터 가장 낮은 값인 15까지입니다. 기본 인스턴스에 결함이 발생하면 우선 순위가 가장 높은 Amazon DocumentDB 복제본을 새로운 기본 인스턴스로 승격시킵니다. Amazon DocumentDB 복제본의 우선 순위는 언제든지 수정할 수 있습니다. 우선 순위 수정으로 인해 장애 조치가 트리거되지는 않습니다. `modify-db-instance` 연산을 `--promotion-tier` 파라미터와 함께 사용할 수 있습니다. 인스턴스의 장애 조치 우선 순위를 사용자 지정하는 방법에 대한 자세한 내용은 Amazon DocumentDB 장애 조치 섹션을 참조하세요.

둘 이상의 Amazon DocumentDB 복제본이 동일한 우선 순위를 공유하여 승격 계층을 만들 수도 있습니다. 둘 이상의 Amazon DocumentDB 복제본이 동일한 우선 순위를 공유하면 크기가 가장 큰 복제본이 기본 복제본으로 승격됩니다. 둘 이상의 Amazon DocumentDB 복제본이 동일한 우선 순위와 크기를 공유하면 동일한 승격 티어에서 임의의 복제본이 승격됩니다.

클러스터에 Amazon DocumentDB 복제본이 포함되어 있지 않으면 기본 인스턴스가 실패 이벤트 중에 다시 생성됩니다. 이 실패 이벤트로 인해 예외적으로 실패하는 읽기 및 쓰기 작업 동안 중단이 발생합니다. 새로운 기본 인스턴스가 생성도면 서비스도 복구되지만 보통 10분 미만의 시간이 걸립니다. Amazon DocumentDB 복제본을 기본 인스턴스로 승격시키는 것이 기본 인스턴스를 새로 생성하는 것보다 훨씬 빠릅니다.

Amazon DocumentDB 인스턴스 관리

다음 항목에서는 Amazon DocumentDB 인스턴스를 관리하는 데 도움이 되는 정보를 제공합니다. 인스턴스 클래스 및 상태, 인스턴스를 생성, 삭제, 수정하는 방법에 대한 세부 정보가 포함됩니다.

주제

- 인스턴스 상태 확인
- Amazon DocumentDB 인스턴스 수명 주기
- 인스턴스 클래스 관리
- NVMe 지원 인스턴스

인스턴스 상태 확인

유효한 인스턴스 상태, 해당 의미, 인스턴스 상태 확인 방법을 알아보려면 Amazon DocumentDB 인스턴스 상태 모니터링 섹션을 참조하세요.

Amazon DocumentDB 인스턴스 수명 주기

Amazon DocumentDB 인스턴스의 수명 주기에는 인스턴스 생성, 수정, 유지 및 업그레이드, 백업 및 복원 수행, 재부팅 및 삭제가 포함됩니다. 이 섹션에서는 이러한 프로세스를 완료하는 방법에 관해 설명합니다.

주제

- 클러스터에 Amazon DocumentDB 인스턴스 추가
- Amazon DocumentDB 인스턴스 설명
- Amazon DocumentDB 인스턴스 수정
- Amazon DocumentDB 인스턴스 재부팅
- Amazon DocumentDB 인스턴스 삭제

AWS Management Console 또는를 사용하여 새 Amazon DocumentDB 인스턴스를 생성할 수 있습니다 AWS CLI. 인스턴스를 클러스터에 추가하려면 클러스터가 사용 가능 상태에 있어야 합니다. 중지된 클러스터에는 인스턴스를 추가할 수 없습니다. 클러스터가 중지된 경우, 먼저 클러스터를 시작하고 클러스터가 사용 가능하게 될 때까지 기다린 다음 인스턴스를 추가합니다. 자세한 내용은 Amazon DocumentDB 클러스터 중지 및 시작 섹션을 참조하십시오.

> **Note**
>
> 콘솔을 사용하여 Amazon DocumentDB 클러스터를 작성하면 인스턴스가 동시에 자동으로 생성됩니다. 추가 인스턴스를 생성하려면 다음 절차 중 하나를 사용하세요.

클러스터에 Amazon DocumentDB 인스턴스 추가

Using the AWS Management Console

다음 절차에 따라 콘솔을 사용하여 클러스터에 대한 Amazon DocumentDB 인스턴스를 생성할 수 있습니다.

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 클러스터를 선택합니다.

   > **Tip**
   > 화면 왼쪽에 탐색 창이 표시되지 않으면 페이지 왼쪽 상단 모서리에서 메뉴 아이콘 (≡) 을 선택합니다.

3. 인스턴스를 추가할 클러스터를 선택하려면 클러스터의 이름 왼쪽에 있는 버튼을 선택합니다.
4. 작업을 선택한 다음 인스턴스 추가를 선택합니다.
5. 다음 항목에 인스턴스 추가: <cluster-name> 페이지에서 클러스터에 추가하려는 각 인스턴스에 대해 다음 단계를 반복 실행합니다. 최대 15개까지 가능합니다.

   a. 인스턴스 식별자 - 이 인스턴스의 고유 식별자를 입력하거나 Amazon DocumentDB가 클러스터 식별자를 기반으로 인스턴스 식별자를 제공하도록 허용할 수 있습니다.

      인스턴스 명명 제약 조건:

      - 길이는 [1-63] 문자, 숫자 또는 하이픈입니다.
      - 첫 번째 문자는 글자이어야 합니다.
      - 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다.
      - 리전별로 Amazon RDS, Neptune 및 Amazon DocumentDB의 모든 인스턴스 AWS 계정에 대해 고유해야 합니다.

   b. 인스턴스 클래스 - 드롭다운 목록에서 이 인스턴스에 대해 원하는 인스턴스 유형을 선택합니다.

   c. 승격 계층 - 드롭다운 목록에서 인스턴스의 승격 계층을 선택하거나 선호 없음을 선택하여 Amazon DocumentDB가 인스턴스의 승격 계층을 설정할 수 있도록 합니다. 번호가 낮을수록 우선 순위가 높습니다. 자세한 내용은 장애 조치 대상 제어 섹션을 참조하십시오.

d. 더 많은 인스턴스를 추가하려면 추가 인스턴스 추가를 선택하고 a, b, c 단계를 반복합니다.

6. 작업을 완료합니다.

   - 클러스터에 인스턴스를 추가하려면 생성을 선택합니다.
   - 작업을 취소하려면 취소를 선택합니다.

인스턴스를 생성하는 데 몇 분 정도 걸립니다. 콘솔 또는를 사용하여 인스턴스의 상태를 AWS CLI 볼 수 있습니다. 자세한 내용은 인스턴스 상태 모니터링 단원을 참조하십시오.

Using the AWS CLI

다음 파라미터와 함께 `create-db-instance` AWS CLI 작업을 사용하여 클러스터의 기본 인스턴스를 생성합니다.

- **`--db-instance-class`** - 필수입니다. 인스턴스의 컴퓨팅 및 메모리 용량(예: `db.m4.large`) 입니다. 모든 AWS 리전에서 모든 인스턴스 클래스를 사용할 수 있는 것은 아닙니다.
- **`--db-instance-identifier`** - 필수입니다. 인스턴스를 식별하는 문자열입니다.

  인스턴스 명명 제약:

  - 길이는 [1-63] 문자, 숫자 또는 하이픈입니다.
  - 첫 번째 문자는 글자이어야 합니다.
  - 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다.
  - 리전별로 Amazon RDS, Neptune 및 Amazon DocumentDB의 모든 인스턴스 AWS 계정에 대해 고유해야 합니다.
- **`--engine`** - 필수입니다. `docdb`여야 합니다.
- **`--availability-zone`** - 선택 사항. 이 인스턴스를 생성할 가용 영역입니다. 다른 가용 영역에서 인스턴스를 찾아서 내결함성을 높이려면 이 파라미터를 사용합니다. 자세한 내용은 Amazon DocumentDB 고가용성 및 복제 섹션을 참조하십시오.
- **`--promotion-tier`** - 선택 사항. 이 인스턴스의 장애 조치 우선 순위 계층입니다. 0~15 사이여야 하며, 숫자가 작을수록 우선 순위가 높습니다. 자세한 내용은 장애 조치 대상 제어 섹션을 참조하십시오.

1. 먼저 인스턴스를 생성할 수 있는 가용 영역을 결정합니다.

인스턴스를 생성하기 전에 사용 가능 영역을 지정하려면 다음 명령을 실행하여 Amazon DocumentDB 클러스터에 사용할 수 있는 가능 영역을 결정합니다.

Linux, macOS, Unix의 경우:

```
aws docdb describe-db-clusters \
       --query 'DBClusters[*].[DBClusterIdentifier,AvailabilityZones[*]]'
```

Windows의 경우:

```
aws docdb describe-db-clusters ^
       --query 'DBClusters[*].[DBClusterIdentifier,AvailabilityZones[*]]'
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
[
    [
        "sample-cluster",
        [
            "us-east-1c",
            "us-east-1b",
            "us-east-1a"
        ]
    ]
]
```

2. 둘째, 리전에서 생성할 수 있는 인스턴스 클래스를 확인합니다.

   리전에서 사용 가능한 인스턴스 클래스를 확인하려면 다음 명령을 실행합니다. 출력에서 Amazon DocumentDB 클러스터에 추가할 인스턴스의 인스턴스 클래스를 선택합니다.

   Linux, macOS, Unix의 경우:

   ```
   aws docdb describe-orderable-db-instance-options \
           --engine docdb \
           --query 'OrderableDBInstanceOptions[*].DBInstanceClass'
   ```

   Windows의 경우:

```
aws docdb describe-orderable-db-instance-options ^
        --engine docdb ^
        --query 'OrderableDBInstanceOptions[*].DBInstanceClass'
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
[
    "db.r5.16xlarge",
    "db.r5.2xlarge",
    "db.r5.4xlarge",
    "db.r5.8xlarge",
    "db.r5.large",
    "db.r5.xlarge"
]
```

3. 마지막으로, Amazon DocumentDB 클러스터에 인스턴스를 추가합니다.

   인스턴스를 Amazon DocumentDB 클러스터에 추가하려면 다음 명령을 실행합니다..

   Linux, macOS, Unix의 경우:

```
aws docdb create-db-instance \
       --db-cluster-identifier sample-cluster \
       --db-instance-identifier sample-instance-2 \
       --availability-zone us-east-1b \
       --promotion-tier 2 \
       --db-instance-class db.r5.xlarge \
       --engine docdb
```

   Windows의 경우:

```
aws docdb create-db-instance ^
       --db-cluster-identifier sample-cluster ^
       --db-instance-identifier sample-instance-2 ^
       --availability-zone us-east-1b ^
       --promotion-tier 2 ^
       --db-instance-class db.r5.xlarge ^
       --engine docdb
```

   이 작업의 출력은 다음과 같이 표시됩니다.

```
{
    "DBInstance": {
        "DBInstanceIdentifier": "sample-instance-2",
        "DBInstanceClass": "db.r5.xlarge",
        "Engine": "docdb",
        "DBInstanceStatus": "creating",
        "PreferredBackupWindow": "02:00-02:30",
        "BackupRetentionPeriod": 1,
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-abcd0123",
                "Status": "active"
            }
        ],
        "AvailabilityZone": "us-east-1b",
        "DBSubnetGroup": {
            "DBSubnetGroupName": "default",
            "DBSubnetGroupDescription": "default",
            "VpcId": "vpc-6242c31a",
            "SubnetGroupStatus": "Complete",
            "Subnets": [
                {
                    "SubnetIdentifier": "subnet-abcd0123",
                    "SubnetAvailabilityZone": {
                        "Name": "us-west-2a"
                    },
                    "SubnetStatus": "Active"
                },
                {
                    "SubnetIdentifier": "subnet-wxyz0123",
                    "SubnetAvailabilityZone": {
                        "Name": "us-west-2b"
                    },
                    "SubnetStatus": "Active"
                }
            ]
        },
        "PreferredMaintenanceWindow": "sun:11:35-sun:12:05",
        "PendingModifiedValues": {},
        "EngineVersion": "3.6.0",
        "AutoMinorVersionUpgrade": true,
        "PubliclyAccessible": false,
        "DBClusterIdentifier": "sample-cluster",
```

```
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:us-east-1:<accountID>:key/sample-key",
        "DbiResourceId": "db-ABCDEFGHIJKLMNOPQRSTUVWXYZ",
        "CACertificateIdentifier": "rds-ca-2019",
        "PromotionTier": 2,
        "DBInstanceArn": "arn:aws:rds:us-east-1:<accountID>:db:sample-instance-2"
    }
}
```

인스턴스를 생성하는 데 몇 분 정도 걸립니다. 콘솔 또는를 사용하여 인스턴스의 상태를 AWS CLI 볼 수 있습니다. 자세한 내용은 Amazon DocumentDB 인스턴스 상태 모니터링 단원을 참조하십시오.

Amazon DocumentDB 인스턴스 설명

Amazon DocumentDB Management Console 또는 AWS CLI 을 사용하여 연결 엔드포인트, 보안 그룹 VPC, 인증 기관 및 Amazon DocumentDB 인스턴스와 관련된 매개 변수 그룹과 같은 세부 정보를 볼 수 있습니다.

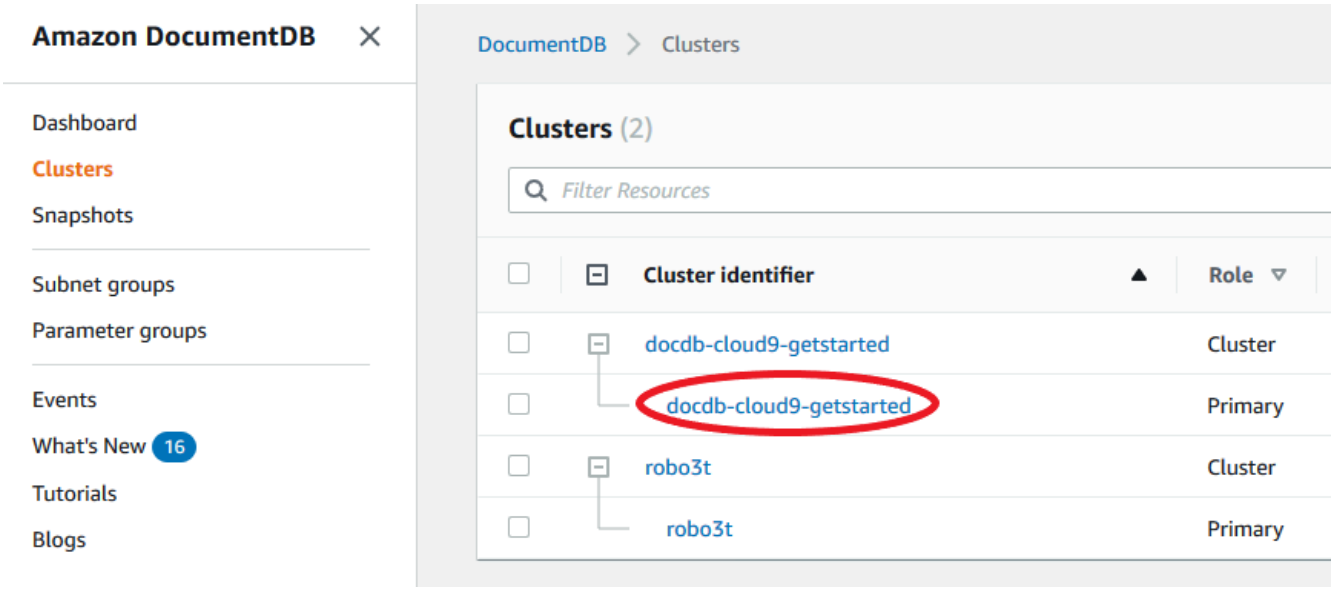
Using the AWS Management Console

AWS Management Console를 사용하여 인스턴스의 세부 정보를 보려면 아래 단계를 따르세요.

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 클러스터를 선택합니다.

   > **Tip**
   > 화면 왼쪽에 탐색 창이 표시되지 않으면 페이지 왼쪽 상단 모서리에서 메뉴 아이콘 (≡) 을 선택합니다.

3. 클러스터 탐색 상자에 클러스터 식별자 열이 표시됩니다. 인스턴스는 아래 스크린샷과 마찬가지로 클러스터 아래에 나열됩니다.



4. 인스턴스 목록에서 세부 정보를 볼 인스턴스의 이름을 선택합니다. 인스턴스에 대한 정보는 다음과 같은 그룹으로 구성됩니다.

   - 요약 - 엔진 버전, 클래스, 상태 및 보류 중인 유지 보수를 포함한 인스턴스에 대한 일반 정보입니다.
   - 연결 및 보안 - 연결 섹션에는 mongo 쉘 또는 애플리케이션을 사용하여 이 인스턴스에 연결할 연결 엔드포인트가 나열됩니다. 보안 그룹 섹션에는 이 인스턴스와 연결된 보안 그룹과 해당 VPC ID 및 설명이 나열됩니다.
   - 구성 - 세부 정보 섹션에는 인스턴스의 Amazon 리소스 이름(ARN), 엔드포인트, 역할, 클래스 및 인증 기관을 포함하여 인스턴스의 구성 및 상태가 나열됩니다. 또한 인스턴스의 보안 및 네트워크 설정과 백업 정보가 나열됩니다. 클러스터 세부 정보 섹션에는 이 인스턴스가 속한 클러스터의 세부 정보가 나열됩니다. 클러스터 인스턴스 섹션에는 클러스터에 속한 모든 인스턴스가 각 인스턴스의 역할 및 클러스터 파라미터 그룹 상태와 함께 나열됩니다.

     > **Note**
     >
     > 클러스터 디테일 헤더 옆에 있는 수정을 선택하여 인스턴스와 연결된 클러스터를 수정할 수 있습니다. 자세한 내용은 Amazon DocumentDB 클러스터 수정 섹션을 참조하십시오.

   - 모니터링 - 이 인스턴스에 대한 CloudWatch Logs 지표. 자세한 내용은 CloudWatch를 사용하여 Amazon DocumentDB 모니터링 섹션을 참조하십시오.
   - 이벤트 및 태그 — 최근 이벤트 섹션에는 이 인스턴스의 최근 이벤트가 나열됩니다. Amazon DocumentDB는 클러스터, 인스턴스, 스냅샷, 보안 그룹 및 클러스터 매개 변수 그룹과 관

련된 이벤트를 기록합니다. 이 정보에는 각 이벤트와 연결된 날짜, 시간 및 메시지가 포함됩니다. 태그 섹션에는 이 클러스터에 연결된 태그가 나열됩니다. 자세한 내용은 Amazon DocumentDB 리소스 태그 지정 단원을 참조하십시오.

Using the AWS CLI

를 사용하여 Amazon DocumentDB 인스턴스의 세부 정보를 보려면 아래 예제와 같이 `describe-db-clusters` 명령을 AWS CLI사용합니다. 자세한 내용은 Amazon DocumentDB 리소스 관리 API 참조에서 `DescribeDBInstances` 섹션을 참조하세요.

> **Note**
>
> 클러스터 및 인스턴스 수명주기 관리와 같은 특정 관리 기능의 경우 Amazon DocumentDB는 Amazon RDS와 공유하는 운영 기술을 활용합니다. `filterName=engine,Values=docdb` 필터 파라미터는 Amazon DocumentDB 클러스터만 반환합니다.

1. 모든 Amazon DocumentDB 인스턴스를 나열합니다.

   다음 AWS CLI 코드는 리전의 모든 Amazon DocumentDB 인스턴스에 대한 세부 정보를 나열합니다.

   Linux, macOS, Unix의 경우:

   ```
   aws docdb describe-db-instances \
       --filter Name=engine,Values=docdb
   ```

   Windows의 경우:

   ```
   aws docdb describe-db-instances \
       --filter Name=engine,Values=docdb
   ```

2. 지정된 Amazon DocumentDB 인스턴스에 대한 모든 세부 정보 나열

   다음 코드는 `sample-cluster-instance`의 세부 정보를 나열합니다. 인스턴스 이름과 함께 `--db-instance-identifier` 파라미터를 포함하면 해당 특정 인스턴스에 대한 정보로 출력이 제한됩니다.

Linux, macOS, Unix의 경우:

```
aws docdb describe-db-instances \
    --db-instance-identifier sample-cluster-instance
```

Windows의 경우:

```
aws docdb describe-db-instances \
    --db-instance-identifier sample-cluster-instance
```

이 작업의 출력은 다음과 같습니다.

```
{
    "DBInstances": [
        {
            "DbiResourceId": "db-BJKKB54PIDV5QFKGVRX5T3S6GM",
            "DBInstanceArn": "arn:aws:rds:us-east-1:012345678901:db:sample-
cluster-instance-00",
            "VpcSecurityGroups": [
                {
                    "VpcSecurityGroupId": "sg-77186e0d",
                    "Status": "active"
                }
            ],
            "DBInstanceClass": "db.r5.large",
            "DBInstanceStatus": "creating",
            "AutoMinorVersionUpgrade": true,
            "PreferredMaintenanceWindow": "fri:09:32-fri:10:02",
            "BackupRetentionPeriod": 1,
            "StorageEncrypted": true,
            "DBClusterIdentifier": "sample-cluster",
            "EngineVersion": "3.6.0",
            "AvailabilityZone": "us-east-1a",
            "Engine": "docdb",
            "PromotionTier": 2,
            "DBInstanceIdentifier": "sample-cluster-instance",
            "PreferredBackupWindow": "00:00-00:30",
            "PubliclyAccessible": false,
            "DBSubnetGroup": {
                "DBSubnetGroupName": "default",
                "Subnets": [
```

```
                {
                    "SubnetIdentifier": "subnet-4e26d263",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1a"
                    },
                    "SubnetStatus": "Active"
                },
                {
                    "SubnetIdentifier": "subnet-afc329f4",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1c"
                    },
                    "SubnetStatus": "Active"
                },
                {
                    "SubnetIdentifier": "subnet-b3806e8f",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1e"
                    },
                    "SubnetStatus": "Active"
                },
                {
                    "SubnetIdentifier": "subnet-53ab3636",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1d"
                    },
                    "SubnetStatus": "Active"
                },
                {
                    "SubnetIdentifier": "subnet-991cb8d0",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1b"
                    },
                    "SubnetStatus": "Active"
                },
                {
                    "SubnetIdentifier": "subnet-29ab1025",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1f"
                    },
                    "SubnetStatus": "Active"
                }
            ],
            "VpcId": "vpc-91280df6",
```

```
                "DBSubnetGroupDescription": "default",
                "SubnetGroupStatus": "Complete"
            },
            "PendingModifiedValues": {},
            "KmsKeyId": "arn:aws:kms:us-east-1:012345678901:key/0961325d-
a50b-44d4-b6a0-a177d5ff730b"
        }
    ]
}
```

Amazon DocumentDB 인스턴스 수정

AWS Management Console 또는를 사용하여 Amazon DocumentDB 인스턴스를 수정할 수 있습니다 AWS CLI. 인스턴스를 수정하려면 인스턴스가 사용 가능 상태에 있어야 합니다. 중지된 인스턴스는 수정할 수 없습니다. 클러스터가 중지된 경우, 먼저 클러스터를 시작하고 인스턴스가 사용 가능하게 될 때까지 기다린 다음 원하는 대로 수정합니다. 자세한 내용은 Amazon DocumentDB 클러스터 중지 및 시작 섹션을 참조하십시오.

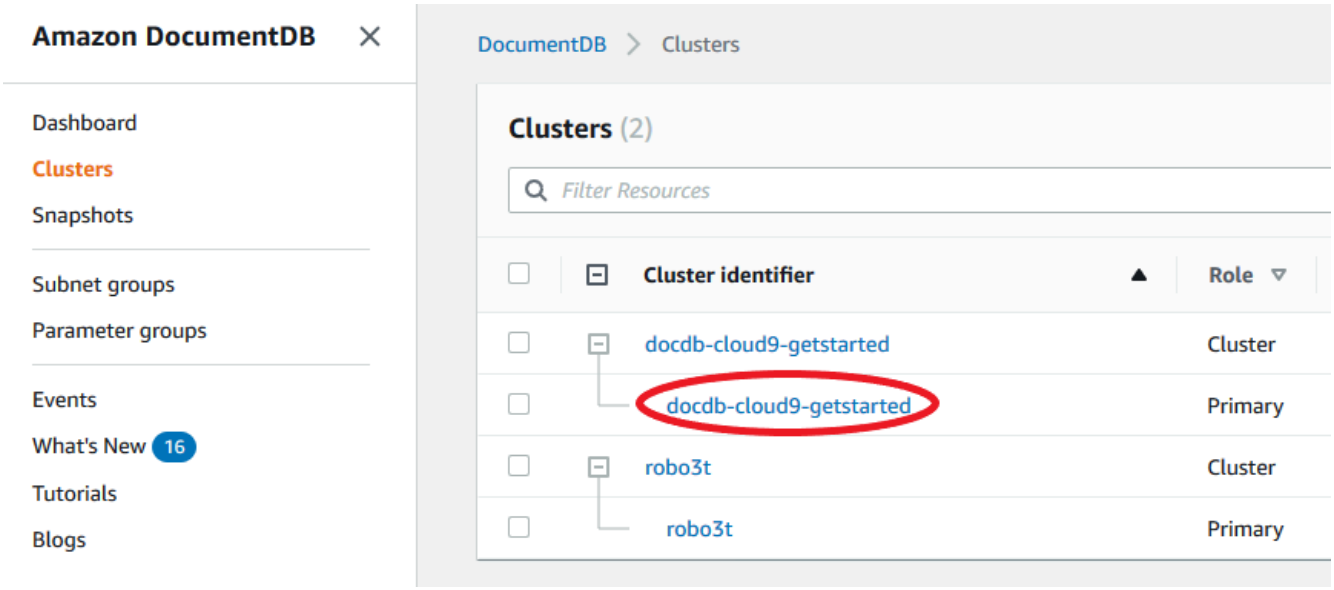
Using the AWS Management Console

콘솔을 사용하여 특정 Amazon DocumentDB 인스턴스를 수정하려면 다음 단계를 완료합니다.

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 클러스터를 선택합니다.

   > **Tip**
   > 화면 왼쪽에 탐색 창이 표시되지 않으면 페이지 왼쪽 상단 모서리에서 메뉴 아이콘 (≡) 을 선택합니다.

3. 클러스터 탐색 상자에 클러스터 식별자 열이 표시됩니다. 인스턴스는 아래 스크린샷과 마찬가지로 클러스터 아래에 나열됩니다.



4. 수정할 인스턴스의 왼쪽에 있는 상자를 선택합니다.
5. 작업을 선택한 후 수정을 선택합니다.
6. 인스턴스 수정: <instance-name> 창에서 원하는 항목을 변경합니다. 다음과 같이 변경할 수 있습니다:

   - 인스턴스 규격 - 인스턴스 식별자 및 클래스. 인스턴스 식별자 명명 제약 조건:
     - 인스턴스 식별자 - 현재 리전 AWS 계정 에서가 소유한 모든 인스턴스에 대해 고유한 이름을 입력합니다. 인스턴스 식별자는 [1-63]개의 영숫자 또는 하이픈을 포함해야 하며 첫 번째 문자로 문자가 있어야 하며 하이픈으로 끝나거나 연속적으로 두 개의 하이픈을 포함할 수 없습니다.
     - 인스턴스 클래스 — 드롭다운 메뉴에서 Amazon DocumentDB 인스턴스의 인스턴스 클래스를 선택합니다. 자세한 내용은 인스턴스 클래스 관리 섹션을 참조하십시오.
   - 인증서 권한 - 인스턴스의 서버 인증서입니다. 자세한 내용은 Amazon DocumentDB TLS 인증서 업데이트 섹션을 참조하십시오.
   - 장애 조치 - 장애 조치 중에 가장 높은 승격 계층을 가진 인스턴스가 기본으로 승격됩니다. 자세한 내용은 Amazon DocumentDB 장애 조치 섹션을 참조하십시오.
   - 관리 - 보류 중인 수정 또는 패치가 클러스터의 인스턴스에 적용되는 유지 관리 창입니다.
7. 작업을 마쳤으면 계속을 선택하여 변경 사항의 요약을 확인합니다.
8. 변경 사항을 확인한 후 즉시 적용하거나 수정 일정 아래의 다음 유지 관리 기간 중에 적용할 수 있습니다. 인스턴스 수정를 선택하여 변경 사항을 저장합니다. 또는 취소를 선택하여 변경 사항을 취소할 수 있습니다.

변경 사항을 적용하는 데 몇 분 정도 걸립니다. 사용 가능 상태인 경우에만 인스턴스를 사용할 수 있습니다. 콘솔 또는 AWS CLI를 사용하여 인스턴스의 상태를 모니터링할 수 있습니다. 자세한 내용은 Amazon DocumentDB 인스턴스 상태 모니터링 단원을 참조하십시오.

Using the AWS CLI

를 사용하여 특정 Amazon DocumentDB 인스턴스를 수정하려면 다음 파라미터`modify-db-instance`와 함께를 AWS CLI사용합니다. 자세한 내용은 ModifyDBInstance을 참조하세요. 다음 코드는 `sample-instance` 인스턴스에 대한 인스턴스 클래스를 `db.r5.large`로 수정합니다.

파라미터

- **`--db-instance-identifier`** - 필수입니다. 수정할 인스턴스의 식별자입니다.
- **`--db-instance-class`** - 선택 사항. 인스턴스의 새 컴퓨팅 및 메모리 용량입니다 (예: `db.r5.large`). 모든 AWS 리전에서 모든 인스턴스 클래스를 사용할 수 있는 것은 아닙니다. 인스턴스 클래스를 수정하면 변경 중에 중단이 발생합니다. 이 요청에 대해 `ApplyImmediately`이 true로 지정되지 않은 경우 다음 유지 관리 기간 동안 변경 사항이 적용됩니다.
- **`--apply-immediately`** 또는 **`--no-apply-immediately`** - 선택 사항. 이 수정을 즉시 적용할지 다음 유지 관리 기간까지 대기할지 여부를 지정합니다. 이 파라미터를 생략하면 다음 유지 관리 기간 중에 수정이 수행됩니다.

Example

Linux, macOS, Unix의 경우:

```
aws docdb modify-db-instance \
       --db-instance-identifier sample-instance \
       --db-instance-class db.r5.large \
       --apply-immediately
```

Windows의 경우:

```
aws docdb modify-db-instance ^
       --db-instance-identifier sample-instance ^
       --db-instance-class db.r5.large ^
       --apply-immediately
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
{
    "DBInstances": [
        {
            "DBInstanceIdentifier": "sample-instance-1",
            "DBInstanceClass": "db.r5.large",
            "Engine": "docdb",
            "DBInstanceStatus": "modifying",
            "Endpoint": {
                "Address": "sample-instance-1.node.us-east-1.docdb.amazonaws.com",
                "Port": 27017,
                "HostedZoneId": "ABCDEFGHIJKLM"
            },
            "InstanceCreateTime": "2020-01-10T22:18:55.921Z",
            "PreferredBackupWindow": "02:00-02:30",
            "BackupRetentionPeriod": 1,
            "VpcSecurityGroups": [
                {
                    "VpcSecurityGroupId": "sg-abcd0123",
                    "Status": "active"
                }
            ],
            "AvailabilityZone": "us-east-1a",
            "DBSubnetGroup": {
                "DBSubnetGroupName": "default",
                "DBSubnetGroupDescription": "default",
                "VpcId": "vpc-abcd0123",
                "SubnetGroupStatus": "Complete",
                "Subnets": [
                    {
                        "SubnetIdentifier": "subnet-abcd0123",
                        "SubnetAvailabilityZone": {
                            "Name": "us-east-1a"
                        },
                        "SubnetStatus": "Active"
                    },
                    {
                        "SubnetIdentifier": "subnet-abcd0123",
                        "SubnetAvailabilityZone": {
                            "Name": "us-east-1b"
                        },
                        "SubnetStatus": "Active"
                    }
                ]
```

```
            },
            "PreferredMaintenanceWindow": "sun:10:57-sun:11:27",
            "PendingModifiedValues": {
                "DBInstanceClass": "db.r5.large"
            },
            "EngineVersion": "3.6.0",
            "AutoMinorVersionUpgrade": true,
            "PubliclyAccessible": false,
            "DBClusterIdentifier": "sample-cluster",
            "StorageEncrypted": true,
            "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/wJalrXUtnFEMI/
K7MDENG/bPxRfiCYEXAMPLEKEY",
            "DbiResourceId": "db-ABCDEFGHIJKLMNOPQRSTUVWXYZ",
            "CACertificateIdentifier": "rds-ca-2019",
            "PromotionTier": 1,
            "DBInstanceArn": "arn:aws:rds:us-east-1:123456789012:db:sample-
instance-1",
            "EnabledCloudwatchLogsExports": [
                "profiler"
            ]
        }
    ]
}
```

수정 사항을 적용하는 데 몇 분 정도 걸립니다. 사용 가능 상태인 경우에만 인스턴스를 사용할 수 있습니다. AWS Management Console 또는를 사용하여 인스턴스의 상태를 모니터링할 수 있습니다 AWS CLI. 자세한 내용은 Amazon DocumentDB 인스턴스 상태 모니터링 단원을 참조하십시오.

Amazon DocumentDB 인스턴스 재부팅

보통 유지보수의 이유로 Amazon DocumentDB 인스턴스를 재부팅해야 하는 경우가 있습니다. 특정 항목을 변경한 경우(예: 클러스터와 연결된 클러스터 파라미터 그룹 변경) 클러스터의 인스턴스를 재부팅해야 변경 내용이 적용됩니다. AWS Management Console 또는를 사용하여 지정된 인스턴스를 재부팅할 수 있습니다 AWS CLI.

인스턴스를 재부팅하면 데이터베이스 엔진 서비스가 재시작됩니다. 재부팅하면 DB 인스턴스 상태가 `rebooting`으로 설정되면서 잠시 중단됩니다. 재부팅이 완료되면 Amazon DocumentDB 이벤트가 생성됩니다.

인스턴스를 재부팅해도 장애 조치가 발생하지 않습니다. Amazon DocumentDB 클러스터를 장애 조치하려면 AWS Management Console 또는 AWS CLI 작업을 사용합니다`failover-db-cluster`. 자세한 내용은 Amazon DocumentDB 장애 조치 단원을 참조하십시오.

사용 가능 상태가 아닌 경우 인스턴스를 재부팅할 수 없습니다. 이전에 수정을 요청했거나 유지 관리 기간 작업 등 여러 가지 이유로 데이터베이스를 사용할 수 없는 경우가 있습니다. 인스턴스 상태에 대한 자세한 내용은 Amazon DocumentDB 인스턴스 상태 모니터링 섹션을 참조하세요.

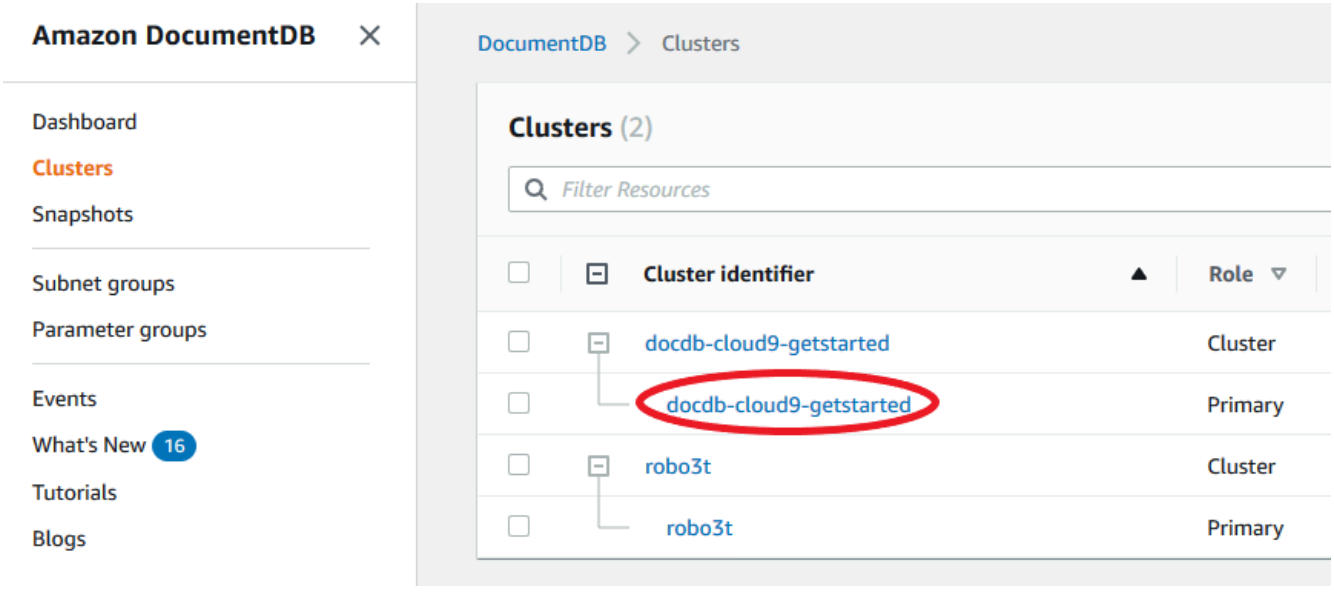
Using the AWS Management Console

다음 절차는 콘솔을 사용하여 지정한 인스턴스를 재부팅합니다.

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 클러스터를 선택합니다.

> **Tip**
>
> 화면 왼쪽에 탐색 창이 표시되지 않으면 페이지 왼쪽 상단 모서리에서 메뉴 아이콘 (≡) 을 선택합니다.

3. 클러스터 탐색 상자에 클러스터 식별자 열이 표시됩니다. 인스턴스는 아래 스크린샷과 마찬가지로 클러스터 아래에 나열됩니다.



4. 재부팅할 인스턴스의 왼쪽에 있는 확인란을 선택합니다.

5. 작업을 선택하고 재부팅을 선택한 다음 재부팅을 선택하여 재부팅을 확인합니다.

인스턴스가 재부팅되는 데 몇 분 정도 걸릴 수 있습니다. 사용 가능 상태인 경우에만 인스턴스를 사용할 수 있습니다. 콘솔 또는 AWS CLI를 사용하여 인스턴스의 상태를 모니터링할 수 있습니다. 자세한 내용은 Amazon DocumentDB 인스턴스 상태 모니터링 단원을 참조하십시오.

Using the AWS CLI

Amazon DocumentDB 인스턴스를 재부팅하려면 `--db-instance-identifier` 파라미터와 함께 `reboot-db-instance` 작업을 사용합니다. 이 파라미터는 재부팅할 인스턴스의 식별자를 지정합니다.

다음 코드는 `sample-instance` 인스턴스를 재부팅합니다.

Example

Linux, macOS, Unix의 경우:

```
aws docdb reboot-db-instance \
       --db-instance-identifier sample-instance
```

Windows의 경우:

```
aws docdb reboot-db-instance ^
       --db-instance-identifier sample-instance
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
{
    "DBInstance": {
        "DBInstanceIdentifier": "sample-instance",
        "DBInstanceClass": "db.r5.large",
        "Engine": "docdb",
        "DBInstanceStatus": "rebooting",
        "Endpoint": {
            "Address": "sample-instance.node.us-east-1.docdb.amazonaws.com",
            "Port": 27017,
            "HostedZoneId": "ABCDEFGHIJKLM"
        },
        "InstanceCreateTime": "2020-03-27T08:05:56.314Z",
        "PreferredBackupWindow": "02:00-02:30",
        "BackupRetentionPeriod": 1,
```

```
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-abcd0123",
                "Status": "active"
            }
        ],
        "AvailabilityZone": "us-east-1c",
        "DBSubnetGroup": {
            "DBSubnetGroupName": "default",
            "DBSubnetGroupDescription": "default",
            "VpcId": "vpc-abcd0123",
            "SubnetGroupStatus": "Complete",
            "Subnets": [
                {
                    "SubnetIdentifier": "subnet-abcd0123",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1a"
                    },
                    "SubnetStatus": "Active"
                },
                {
                    "SubnetIdentifier": "subnet-wxyz0123",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1b"
                    },
                    "SubnetStatus": "Active"
                }
            ]
        },
        "PreferredMaintenanceWindow": "sun:06:53-sun:07:23",
        "PendingModifiedValues": {},
        "EngineVersion": "3.6.0",
        "AutoMinorVersionUpgrade": true,
        "PubliclyAccessible": false,
        "DBClusterIdentifier": "sample-cluster",
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:us-east-1:<accountID>:key/sample-key",
        "DbiResourceId": "db-ABCDEFGHIJKLMNOPQRSTUVWXYZ",
        "CACertificateIdentifier": "rds-ca-2019",
        "PromotionTier": 1,
        "DBInstanceArn": "arn:aws:rds:us-east-1:<accountID>:db:sample-instance",
        "EnabledCloudwatchLogsExports": [
            "profiler"
        ]
```

```
    }
}
```

인스턴스가 재부팅되는 데 몇 분 정도 걸릴 수 있습니다. 사용 가능 상태인 경우에만 인스턴스를 사용할 수 있습니다. 콘솔 또는 AWS CLI를 사용하여 인스턴스의 상태를 모니터링할 수 있습니다. 자세한 내용은 Amazon DocumentDB 인스턴스 상태 모니터링 단원을 참조하십시오.

Amazon DocumentDB 인스턴스 삭제

AWS Management Console 또는를 사용하여 Amazon DocumentDB 인스턴스를 삭제할 수 있습니다 AWS CLI. 인스턴스를 삭제하려면 인스턴스가 사용 가능 상태에 있어야 합니다. 중지된 인스턴스는 삭제할 수 없습니다. 인스턴스가 포함된 Amazon DocumentDB 클러스터가 중지된 경우 먼저 클러스터를 시작하고 인스턴스가 사용 가능해질 때까지 기다렸다가 인스턴스를 삭제합니다. 자세한 내용은 Amazon DocumentDB 클러스터 중지 및 시작 섹션을 참조하십시오.

> **Note**
>
> Amazon DocumentDB에서는 모든 데이터를 클러스터 볼륨에 저장합니다. 모든 인스턴스를 클러스터에서 제거하는 경우에도 데이터가 클러스터 볼륨에 유지됩니다. 데이터에 다시 액세스해야 하는 경우 언제든지 클러스터에 인스턴스를 추가하고 중단한 지점을 선택할 수 있습니다.

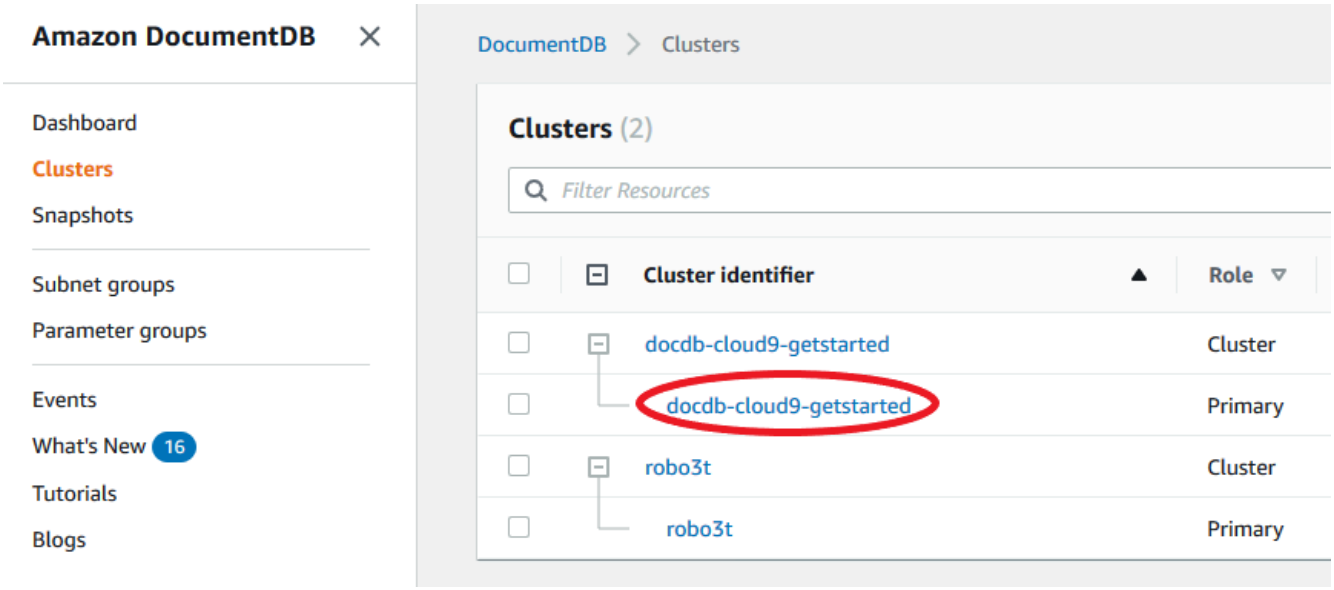
Using the AWS Management Console

다음 절차에서는 콘솔을 사용하여 지정된 Amazon DocumentDB 인스턴스를 삭제합니다.

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 클러스터를 선택합니다.

   > **Tip**
   >
   > 화면 왼쪽에 탐색 창이 표시되지 않으면 페이지 왼쪽 상단 모서리에서 메뉴 아이콘 (≡) 을 선택합니다.

3. 클러스터 탐색 상자에 클러스터 식별자 열이 표시됩니다. 인스턴스는 아래 스크린샷과 마찬가지로 클러스터 아래에 나열됩니다.



4. 삭제할 인스턴스의 왼쪽에 있는 확인란을 선택합니다.
5. 작업을 선택한 후 삭제를 선택합니다.

   1. 클러스터의 마지막 인스턴스를 삭제하는 경우:
      - 최종 클러스터 스냅샷을 생성하시겠습니까? — 클러스터를 삭제하기 전에 최종 스냅샷을 생성하려면 예를 선택합니다. 그렇지 않은 경우 아니요를 선택합니다.
      - 최종 스냅샷 이름 - 최종 스냅샷을 생성하도록 선택한 경우 생성된 새 클러스터 스냅샷의 클러스터 스냅샷 식별자를 입력합니다.
      - <instance-name> 인스턴스를 삭제하시겠습니까? — 삭제 확인을 위해 전체 클러스터 삭제라는 문구를 필드에 입력합니다.
   2. 클러스터의 마지막 인스턴스를 삭제하지 않는 경우:
      - <instance-name> 인스턴스를 삭제하시겠습니까? — 삭제 라는 문구를 필드에 입력하여 삭제를 확인합니다.
6. 삭제를 선택하여 인스턴스를 삭제합니다.

인스턴스를 삭제하는 데 몇 분 정도 걸립니다. 인스턴스 상태를 모니터링하려면 Amazon DocumentDB 인스턴스 상태 모니터링 섹션을 참조하십시오.

Using the AWS CLI

다음 절차는 AWS CLI을 사용하여 Amazon DocumentDB 인스턴스를 삭제합니다.

1. 먼저 Amazon DocumentDB 클러스터에 몇 개의 인스턴스가 있는지 확인합니다:

클러스터에 있는 인스턴스 수를 확인하려면 다음과 같이 `describe-db-clusters` 명령을 실행합니다.

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].
[DBClusterIdentifier,DBClusterMembers[*].DBInstanceIdentifier]'
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
[
    [
        "sample-cluster",
        [
            "sample-instance-1",
            "sample-instance-2"
        ]
    ]
]
```

2. Amazon DocumentDB 클러스터에 인스턴스가 두 개 이상 있는 경우:

   지정된 Amazon DocumentDB 인스턴스를 삭제하려면 아래와 같이 `delete-db-instance` 명령을 `--db-instance-identifier` 파라미터와 함께 사용합니다. 인스턴스를 삭제하는 데 몇 분 정도 걸립니다. 인스턴스 상태를 모니터링하려면 Amazon DocumentDB 인스턴스 상태 모니터링 섹션을 참조하십시오.

   ```
   aws docdb delete-db-instance \
          --db-instance-identifier sample-instance-2
   ```

   이 작업의 출력은 다음과 같이 표시됩니다.

   ```
   {
       "DBInstance": {
           "DBInstanceIdentifier": "sample-instance-2",
           "DBInstanceClass": "db.r5.large",
           "Engine": "docdb",
           "DBInstanceStatus": "deleting",
           "Endpoint": {
               "Address": "sample-instance-2.node.us-east-1.docdb.amazonaws.com",
   ```

```
            "Port": 27017,
            "HostedZoneId": "ABCDEFGHIJKLM"
        },
        "InstanceCreateTime": "2020-03-27T08:05:56.314Z",
        "PreferredBackupWindow": "02:00-02:30",
        "BackupRetentionPeriod": 1,
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-abcd0123",
                "Status": "active"
            }
        ],
        "AvailabilityZone": "us-east-1c",
        "DBSubnetGroup": {
            "DBSubnetGroupName": "default",
            "DBSubnetGroupDescription": "default",
            "VpcId": "vpc-6242c31a",
            "SubnetGroupStatus": "Complete",
            "Subnets": [
                {
                    "SubnetIdentifier": "subnet-abcd0123",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1a"
                    },
                    "SubnetStatus": "Active"
                },
                {
                    "SubnetIdentifier": "subnet-wxyz0123",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1b"
                    },
                    "SubnetStatus": "Active"
                }
            ]
        },
        "PreferredMaintenanceWindow": "sun:06:53-sun:07:23",
        "PendingModifiedValues": {},
        "EngineVersion": "3.6.0",
        "AutoMinorVersionUpgrade": true,
        "PubliclyAccessible": false,
        "DBClusterIdentifier": "sample-cluster",
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:us-east-1:<accountID>:key/sample-key",
        "DbiResourceId": "db-ABCDEFGHIJKLMNOPQRSTUVWXYZ",
```

```
        "CACertificateIdentifier": "rds-ca-2019",
        "PromotionTier": 1,
        "DBInstanceArn": "arn:aws:rds:us-east-1:<accountID>:db:sample-instance-2",
        "EnabledCloudwatchLogsExports": [
            "profiler"
        ]
    }
}
```

3. 삭제할 인스턴스가 Amazon DocumentDB 클러스터의 마지막 인스턴스인 경우:

   Amazon DocumentDB 클러스터의 마지막 인스턴스를 삭제하는 경우 해당 클러스터와 해당 클러스터에 연결된 자동 스냅샷 및 연속 백업도 삭제합니다.

   클러스터의 마지막 인스턴스를 삭제하려면 클러스터를 삭제하고 선택적으로 최종 스냅샷을 생성할 수 있습니다. 자세한 내용은 Amazon DocumentDB 클러스터 삭제 섹션을 참조하십시오.

삭제 방지

또한 Amazon DocumentDB 클러스터의 마지막 인스턴스를 삭제하면 해당 클러스터와 관련된 자동 스냅샷 및 연속 백업뿐만 아니라 클러스터도 삭제됩니다. Amazon DocumentDB는 AWS Management Console 또는를 사용하여 삭제 작업을 수행하든 관계없이 클러스터에 대한 삭제 방지를 적용합니다 AWS CLI. 삭제 방지가 활성화되어 있으면 클러스터를 삭제할 수 없습니다.

삭제 방지가 활성화된 클러스터를 삭제하려면 먼저 클러스터를 수정하고 삭제 방지를 비활성화해야 합니다. 자세한 내용은 Amazon DocumentDB 클러스터 삭제 섹션을 참조하십시오.

인스턴스 클래스 관리

인스턴스 클래스는 Amazon DocumentDB(MongoDB 호환) 인스턴스의 컴퓨팅 및 메모리 용량을 결정합니다. 필요한 인스턴스 클래스는 DB 인스턴스의 처리력 및 메모리 요구 사항에 따라 다릅니다.

Amazon DocumentDB는 R4, R5, R6G, T3 및 T4G 인스턴스 클래스 제품군을 지원합니다. 이들 클래스는 메모리 집약적 애플리케이션에 최적화된 최신 세대 인스턴스 클래스입니다. 이들 클래스의 사양은 인스턴스 클래스 사양 섹션을 참조하세요.

주제

- 인스턴스 클래스 결정
- 인스턴스의 클래스 변경

- 리전별 지원되는 인스턴스 클래스
- 인스턴스 클래스 사양

인스턴스 클래스 결정

인스턴스의 클래스를 확인하려면 AWS Management Console 또는 `describe-db-instances` AWS CLI 작업을 사용할 수 있습니다.

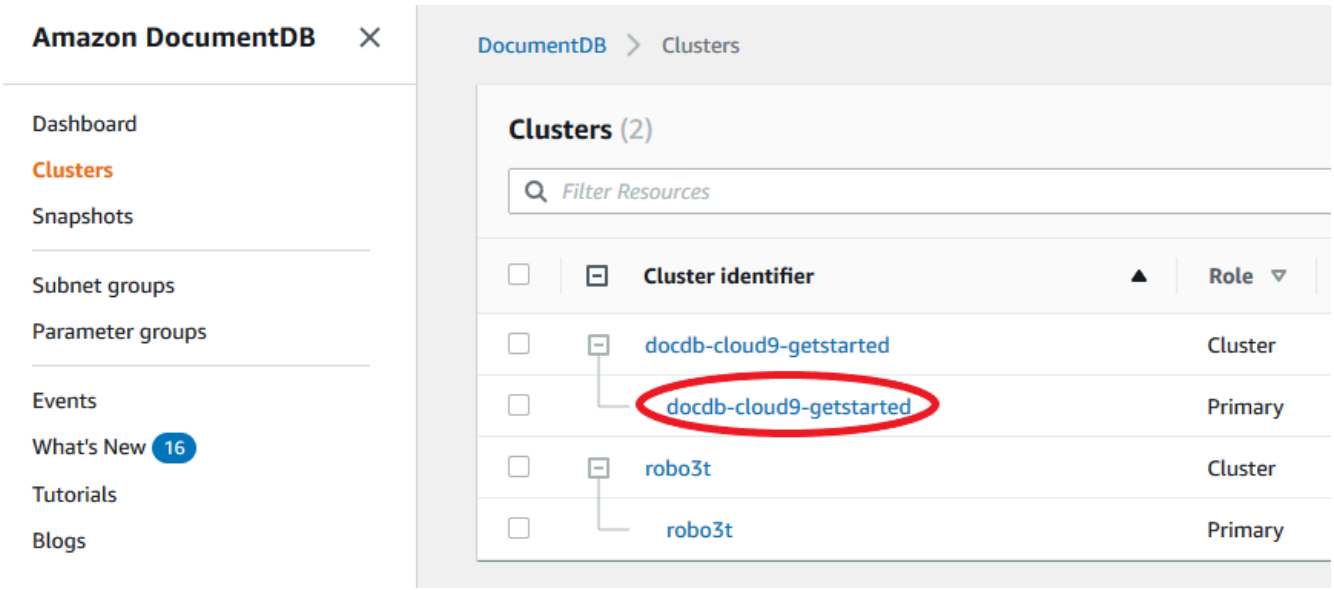
Using the AWS Management Console

클러스터 인스턴스의 인스턴스 클래스를 결정하려면 콘솔에서 다음 단계를 완료하세요.

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 클러스터를 선택하여 관심 있는 인스턴스를 찾습니다.

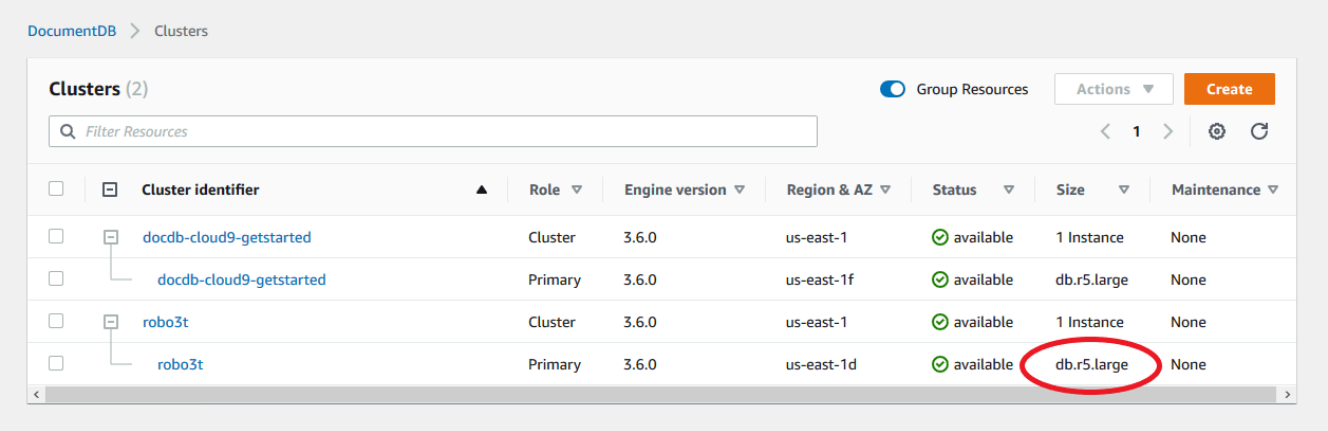
> **Tip**
>
> 화면 왼쪽에 탐색 창이 표시되지 않으면 페이지 왼쪽 상단 모서리에서 메뉴 아이콘 (≡) 을 선택합니다.

3. 클러스터 탐색 상자에 클러스터 식별자 열이 표시됩니다. 인스턴스는 아래 스크린샷과 마찬가지로 클러스터 아래에 나열됩니다.



4. 인스턴스 목록에서 클러스터를 확장하여 원하는 인스턴스를 찾으세요. 원하는 인스턴스를 찾습니다. 그런 다음 인스턴스 행의 크기 열을 확인하여 인스턴스 클래스를 확인합니다.

   다음 이미지에서는 인스턴스 `robo3t`의 인스턴스 클래스가 `db.r5.4xlarge`입니다.



Using the AWS CLI

를 사용하여 인스턴스의 클래스를 확인하려면 다음 파라미터와 함께 `describe-db-instances` 작업을 AWS CLI사용합니다.

- **`--db-instance-identifier`** - 선택 사항입니다. 인스턴스 클래스를 찾을 인스턴스를 지정합니다. 이 매개 변수를 생략한 경우 `describe-db-instances`는 최대 100개의 인스턴스에 대한 설명을 반환합니다.
- **`--query`** – 선택 사항. 결과에 포함할 인스턴스 멤버를 지정합니다. 이 매개 변수를 생략하면 모든 인스턴스 멤버가 반환됩니다.

Example

다음 예에서는 모든 인스턴스에 대한 인스턴스 `sample-instance-1` 이름과 클래스를 찾습니다.

Linux, macOS, Unix의 경우:

```
aws docdb describe-db-instances \
    --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceClass]' \
    --db-instance-identifier sample-instance-1
```

Windows의 경우:

```
aws docdb describe-db-instances ^
```

```
    --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceClass]' ^
    --db-instance-identifier sample-instance-1
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
[
    [
        "sample-instance-1",
        "db.r5.large"
    ]
```

Example

다음 예에서는 최대 100개의 Amazon DocumentDB 인스턴스에 대한 인스턴스 이름과 클래스를 찾습니다.

Linux, macOS, Unix의 경우:

```
aws docdb describe-db-instances \
    --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceClass]' \
    --filter Name=engine,Values=docdb
```

Windows의 경우:

```
aws docdb describe-db-instances ^
    --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceClass]' ^
    --filter Name=engine,Values=docdb
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
[
    [
        "sample-instance-1",
        "db.r5.large"
    ],
    [
        "sample-instance-2",
        "db.r5.large"
    ],
    [
        "sample-instance-3",
```

```
        "db.r5.4xlarge"
    ],
    [
        "sample-instance-4",
        "db.r5.4xlarge"
    ]
]
```

자세한 내용은 Amazon DocumentDB 인스턴스 설명 단원을 참조하십시오.

인스턴스의 클래스 변경

AWS Management Console 또는를 사용하여 인스턴스의 인스턴스 클래스를 변경할 수 있습니다 AWS CLI. 자세한 내용은 Amazon DocumentDB 인스턴스 수정 단원을 참조하십시오.

리전별 지원되는 인스턴스 클래스

Amazon DocumentDB는 다음 인스턴스 클래스를 지원합니다.

- R6G- Arm-based AWS Graviton2 프로세서로 구동되는 메모리 최적화 인스턴스의 최신 세대는 5% 더 저렴한 비용으로 R5 인스턴스보다 최대 30% 더 나은 성능을 제공합니다.
- R6GD- 임시 데이터를 위한 로컬 비휘발성 메모리 익스프레스(NVMe) 기반 솔리드 스테이트 드라이브(SSD) 스토리지가 있는 메모리 최적화 R6G 인스턴스입니다.
- R5 - 동일한 인스턴스 비용으로 R4 인스턴스보다 최대 100% 더 나은 성능을 제공하는 메모리 최적화 인스턴스입니다.
- R4 - 이전 세대의 메모리 최적화 인스턴스입니다.
- T4G- 기준 수준의 CPU 성능을 제공하는 Arm-based AWS Graviton2 프로세서로 구동되는 최신 세대 저비용 버스트 가능 범용 인스턴스 유형으로, T3 인스턴스보다 최대 35% 더 나은 가격 성능을 제공하며 일시적인 사용량 급증이 발생하는 중간 정도의 CPU 사용량이 있는 애플리케이션을 실행하는 데 적합합니다.
- T3 - 필요한 기간 동안 언제든지 CPU 사용량을 버스트할 수 있는 기능과 함께 기본 수준의 CPU 성능을 제공하는 저비용 버스터블 범용 인스턴스 유형입니다.

각 인스턴스 클래스의 세부 사양은 인스턴스 클래스 사양 섹션을 참조하세요.

특정 인스턴스 클래스가 특정 리전에서 지원되거나 지원되지 않을 수 있습니다. 다음 표에는 각 리전에서 Amazon DocumentDB가 지원하는 인스턴스 클래스가 나와 있습니다.

리전별 지원되는 인스턴스 클래스

리전	R6GD	R6G	R5	R4	T4G	T3
미국 동부(오하이오)	지원	지원	지원	지원	지원	지원
미국 동부(버지니아 북부)	지원	지원	지원	지원	지원	지원
미국 서부(오리건)	지원	지원	지원	지원	지원	지원
아프리카(케이프타운)		지원	지원		지원	지원
남아메리카(상파울루)	지원	지원	지원		지원	지원
아시아 태평양(홍콩)		지원	지원		지원	지원
아시아 태평양(하이데라바드)			지원			지원
아시아 태평양(뭄바이)	지원	지원	지원		지원	지원
아시아 태평양(서울)	지원	지원	지원		지원	지원
아시아 태평양(시드니)	지원	지원	지원		지원	지원
아시아 태평양(싱가포르)	지원	지원	지원		지원	지원
아시아 태평양(도쿄)	지원	지원	지원		지원	지원

리전	R6GD	R6G	R5	R4	T4G	T3
캐나다(중부)	지원	지원	지원		지원	지원
유럽(프랑크푸르트)	지원	지원	지원		지원	지원
유럽(아일랜드)	지원	지원	지원	지원	지원	지원
유럽(런던)	지원	지원	지원		지원	지원
유럽(밀라노)		지원	지원		지원	지원
유럽(파리)	지원	지원	지원		지원	지원
유럽(스페인)		지원	지원		지원	지원
유럽(스톡홀름)		지원	지원		지원	지원
중동(UAE)		지원	지원		지원	지원
중국(베이징)	지원	지원	지원		지원	지원
중국(닝샤)		지원	지원		지원	지원
이스라엘(텔아비브)		지원	지원		지원	지원
AWS GovCloud(미국 서부)	지원	지원	지원		지원	지원
AWS GovCloud(미국 동부)	지원	지원	지원		지원	지원

인스턴스 클래스 사양

다음 표에는 각 클래스에서 지원되는 인스턴스 유형을 포함하여 Amazon DocumentDB 인스턴스 클래스에 대한 세부 정보가 나와 있습니다. 테이블 아래에서 각 테이블 열에 대한 설명을 찾아볼 수 있습니다.

인스턴스 클래스	vCPU[1]	메모리 (GiB)[2]	NVMe SSD 계층형 캐시(GiB)[3]	최대 온도 스토리지 (GiB)[4]	기준/버스트 대역폭 (Gbps)[5]	지원 엔진[6]
R6G — Graviton2 기반의 현재 세대 메모리 최적화 인스턴스 클래스						
db.r6g.large	2	16	-	32	0.75/10	4.0.0 및 5.0.0
db.r6g.xlarge	4	32	-	63	1.25/10	4.0.0 및 5.0.0
db.r6g.2xlarge	8	64	-	126	2.5/10	4.0.0 및 5.0.0
db.r6g.4xlarge	16	128	-	252	5.0/10	4.0.0 및 5.0.0
db.r6g.8xlarge	32	256	-	504	12	4.0.0 및 5.0.0
db.r6g.12xlarge	48	384	-	756	20	4.0.0 및 5.0.0
db.r6g.16xlarge	64	512	-	1008	25	4.0.0 및 5.0.0
R6GD - Graviton2 기반 현재 세대 NVMe 지원 인스턴스 클래스						
db.r6gd.xlarge	4	32	173	64	1.25/10	5.0.0 전용

인스턴스 클래스	vCPU[1]	메모리 (GiB)[2]	NVMe SSD 계층형 캐시(GiB)[3]	최대 온도 스토리지 (GiB)[4]	기준/버스트 대역폭 (Gbps)[5]	지원 엔진[6]
db.r6gd.2xlarge	8	64	346	128	2.5/10	5.0.0 전용
db.r6gd.4xlarge	16	128	694	256	5.0/10	5.0.0 전용
db.r6gd.8xlarge	32	256	1388	512	12	5.0.0 전용
db.r6gd.12xlarge	48	384	2082	768	20	5.0.0 전용
db.r6gd.16xlarge	64	512	2776	1024	25	5.0.0 전용
R5 – 이전 세대 메모리 최적화 인스턴스 클래스						
db.r5.large	2	16	-	31	0.75/10	3.6.0, 4.0.0 및 5.0.0
db.r5.xlarge	4	32	-	62	1.25/10	3.6.0, 4.0.0 및 5.0.0
db.r5.2xlarge	8	64	-	124	2.5/10	3.6.0, 4.0.0 및 5.0.0
db.r5.4xlarge	16	128	-	249	5.0/10	3.6.0, 4.0.0 및 5.0.0
db.r5.8xlarge	32	256	-	504	10	3.6.0, 4.0.0 및 5.0.0
db.r5.12xlarge	48	384	-	748	12	3.6.0, 4.0.0 및 5.0.0

인스턴스 클래스	vCPU[1]	메모리 (GiB)[2]	NVMe SSD 계층형 캐시(GiB)[3]	최대 온도 스토리지 (GiB)[4]	기준/버스트 대역폭 (Gbps)[5]	지원 엔진[6]
db.r5.16xlarge	64	512	-	1008	20	3.6.0, 4.0.0 및 5.0.0
db.r5.24xlarge	96	768	-	1500	25	3.6.0, 4.0.0 및 5.0.0
R4 – 이전 세대 메모리 최적화 인스턴스 클래스						
db.r4.large	2	15.25	-	30	0.75/10	3.6.0만
db.r4.xlarge	4	30.5	-	60	1.25/10	3.6.0만
db.r4.2xlarge	8	61	-	120	2.5/10	3.6.0만
db.r4.4xlarge	16	122	-	240	5.0/10	3.6.0만
db.r4.8xlarge	32	244	-	480	10	3.6.0만
db.r4.16xlarge	64	488	-	960	25	3.6.0만
T4G – Graviton2 기반 최신 세대 성능 버스트 기능이 있는 인스턴스 클래스						
db.t4g.medium	2	4	-	8.13	0.256/5	4.0.0 및 5.0.0
T3 – 이전 세대 성능 버스트 기능이 있는 인스턴스 클래스						
db.t3.medium	2	4	-	7.5	0.256/5	3.6.0, 4.0.0 및 5.0.0

인스턴스 클래스	vCPU[1]	메모리 (GiB)[2]	NVMe SSD 계층형 캐시(GiB)[3]	최대 온도 스토리지 (GiB)[4]	기준/버스트 대역폭 (Gbps)[5]	지원 엔진[6]

1. vCPU - 가상 중앙 처리 장치(CPU)의 수입니다. 가상 CPU는 인스턴스 클래스를 비교하는 데 사용할 수 있는 용량을 가진 디바이스입니다. 특정 프로세서를 구매하거나 임차해 몇 개월 또는 몇 년간 사용하는 것이 아니라, 시간 단위로 용량을 임대합니다. 기본 하드웨어 용량에 관계없이 일정 CPU 용량을 제공하는 것이 목표입니다.
2. 메모리(GiB) - 인스턴스에 할당되는 RAM(기가바이트)입니다. 메모리와 vCPU 간 일정한 비율이 존재하는 경우가 많다는 점에 유의하세요.
3. NVMe SSD 계층형 캐시 - 임시 데이터를 저장하기 위한 확장 캐시로 할당된 SSD 볼륨의 공간으로, 기가바이트 단위로 측정됩니다. 이 캐시는 NVMe 지원 인스턴스에서만 사용할 수 있습니다.
4. 최대 임시 스토리지(GiB) - 비영구 임시 파일 스토리지를 위해 인스턴스에 할당된 기가바이트 단위로 측정된 공간입니다. NVMe 지원 인스턴스의 경우이 스토리지는 NVMe 기반 SSD 볼륨에서 호스팅됩니다. 다른 모든 인스턴스에서는 Amazon Elastic Block Store(EBS)에서 호스팅됩니다.
5. 기준/버스트 대역폭(Gbps) - 버스트 대역폭은 초당 기가비트 단위의 최대 대역폭을 나타냅니다. 예상 처리량을 초당 기가바이트 단위로 얻으려면 8로 나눕니다. 크기가 4xlarge 이하인 인스턴스에는 기준 대역폭이 있습니다. 추가 요구 사항을 충족하려면 네트워크 I/O 크레딧 메커니즘을 통해 기준 대역폭을 초과하여 확장할 수 있습니다. 인스턴스는 인스턴스 크기에 따라 일반적으로 5분에서 60분까지 제한된 시간 동안 버스트 대역폭을 사용할 수 있습니다.
6. 지원 엔진 - 인스턴스 클래스를 지원하는 Amazon DocumentDB 엔진.

NVMe 지원 인스턴스

NVMe 지원 인스턴스는 일반 인스턴스의 메모리를 초과하는 대규모 데이터 세트가 있는 애플리케이션에 대해 최대 7배 더 빠른 쿼리 성능을 제공합니다. 이러한 인스턴스는 r6gd 인스턴스에서 사용할 수 있는 로컬 비휘발성 메모리 익스프레스(NVMe) 기반 솔리드 스테이트 드라이브(SSD) 스토리지를 활용하여 임시 데이터를 저장하고 네트워크 기반 스토리지 액세스를 줄이며 읽기 지연 시간과 처리량을 개선합니다.

로컬 SSD 공간은 두 섹션으로 나뉩니다.

- 계층형 캐시 - 로컬 SSD의 약 73%가 데이터베이스 캐시로 할당되므로 시스템이 기본 메모리만 저장할 때보다 최대 5배 더 많은 데이터베이스 페이지를 저장할 수 있습니다. 로컬 SSD는 2계층 캐시 역할을 하지만 기존 인 메모리 버퍼 캐시는 1계층 캐시로 유지됩니다. 쿼리는 버퍼 캐시와 SSD 캐시 모두에서 누락이 발생한 경우에만 외부 스토리지에 액세스합니다.

- 임시 스토리지 - 나머지 27%는 비영구 임시 파일 스토리지용으로 예약되어 있으며, 정렬 또는 인덱스 빌드와 같은 리소스 집약적인 작업과 관련된 복잡한 쿼리에 사용됩니다. 일반적인 경우 임시 공간은 Amazon Elastic Block Store(EBS) 볼륨에 상주합니다. SSD의 로컬 호스팅 임시 스토리지는 정렬과 관련된 쿼리 지연 시간을 최대 2배까지 줄이고 인덱스 빌드와 같은 리소스 집약적인 작업을 가속화합니다.

지원되는 NVMe 지원 인스턴스의 유형 및 사양에 대한 세부 정보는에서 확인할 수 있습니다인스턴스 클래스 사양.

주제

- NVMe 지원 인스턴스의 권장 사용 사례
- Amazon DocumentDB에서 NVMe 지원 인스턴스 사용
- NVMe 지원 인스턴스 모니터링

NVMe 지원 인스턴스의 권장 사용 사례

다음 시나리오에서는 NVMe 지원 인스턴스를 사용하는 것이 좋습니다.

- 읽기 중심 워크로드 - 워크로드가 읽기 집약적이고 데이터 세트가 버퍼 캐시보다 크며 `ReadIOPS` 지표가 낮`BufferCacheHitRatio`거나 높은 경우 NVMe 지원 인스턴스는 성능 이점을 제공할 수 있습니다.
- 업데이트가 많은 워크로드 - 워크로드가 업데이트 집약적이고 네트워크 스토리지의 읽기 지연 시간으로 인해 가비지 수집을 따라잡을 수 없는 경우 NVMe 지원 인스턴스가 문제를 완화하는 데 도움이 될 수 있습니다.

NVMe 지원 인스턴스는 다음과 같은 다양한 사용 사례에 도움이 될 수 있습니다.

- 인터넷 규모 애플리케이션 - 엄격한 성능의 서비스 수준 계약(SLAs)을 사용하는 결제 처리, 결제 및 전자 상거래와 같은 애플리케이션은 NVMe 지원 인스턴스의 성능 이점을 활용할 수 있습니다.
- 실시간 보고 대시보드 - 지표/데이터 수집을 위해 수백 개의 쿼리를 실행하는 대시보드는 NVMe 지원 인스턴스의 짧은 지연 시간과 높은 처리량의 이점을 누릴 수 있습니다.
- 생성형 AI 애플리케이션 - 벡터 검색을 사용하여 수백만 개의 벡터 임베딩에서 정확하거나 가장 가까운 이웃을 찾는 애플리케이션은 NVMe 지원 인스턴스의 고성능을 활용할 수 있습니다.

Amazon DocumentDB에서 NVMe 지원 인스턴스 사용

Amazon DocumentDB의 NVMe 지원 인스턴스를 사용하려면:

- Amazon DocumentDB 클러스터를 생성하고 NVMe 지원 인스턴스 클래스 중 하나를 추가합니다. 자세한 내용은 Amazon DocumentDB 클러스터 생성 단원을 참조하십시오.
- 또는 NVMe 지원 인스턴스 클래스 중 하나를 사용하도록 기존 Amazon DocumentDB 클러스터를 수정합니다. 자세한 내용은 Amazon DocumentDB 클러스터 수정 단원을 참조하십시오.

여러 AWS 리전에서 NVMe 지원 인스턴스의 가용성을 확인하려면 섹션을 참조하세요리전별 지원되는 인스턴스 클래스.

NVMe 지원 인스턴스에서 일반 인스턴스로 다시 전환하려면 Amazon DocumentDB 인스턴스의 데이터베이스 인스턴스 클래스를 NVMe 스토리지가 없는 유사한 인스턴스 클래스로 수정합니다. 예를 들어 현재 인스턴스 클래스가 'db.r6gd.4xlarge'인 경우 'db.r6g.4xlarge'를 선택하여 다시 전환합니다. 자세한 내용은 Amazon DocumentDB 클러스터 수정 단원을 참조하십시오.

NVMe 지원 인스턴스 모니터링

Amazon CloudWatch에서 사용할 수 있는 일반 인스턴스 지표 외에도 NVMe 기반 인스턴스는 NVMe 기반 SSD 스토리지, IOPS 및 처리량과 관련된 추가 지표를 내보냅니다.

```
NVMeStorageCacheHitRatio
FreeNVMeStorage
ReadIOPSNVMeStorage
ReadLatencyNVMeStorage
ReadThroughputNVMeStorage
WriteIOPSNVMeStorage
WriteLatencyNVMeStorage
WriteThroughputNVMeStorage
```

이러한 지표에 대한 자세한 내용은 섹션을 참조하세요. NVMe 지원 인스턴스 지표

Amazon DocumentDB 서브넷 그룹 관리

Virtual Private Cloud(VPC)는 전용 가상 네트워크입니다 AWS 계정. AWS 클라우드의 다른 가상 네트워크와 논리적으로 격리되어 있습니다. Amazon DocumentDB 클러스터와 같은 AWS 리소스를 Amazon VPC로 시작할 수 있습니다. IP 주소 범위와 VPC 범위를 설정하고 서브넷을 추가하고 보안 그룹을 연결한 다음 라우팅 테이블을 구성합니다.

서브넷은 사용자의 Amazon VPC의 IP 주소 범위입니다. 리소스를 지정된 서브넷 AWS 으로 시작할 수 있습니다. 인터넷에 연결해야 할 리소스에 대한 퍼블릭 서브넷을 사용합니다. 인터넷에 연결하지 않을 리소스에 대한 프라이빗 서브넷을 사용합니다. 퍼블릭 및 프라이빗 서브넷에 대한 자세한 내용은 Amazon Virtual 프라이빗 클라우드 사용 설명서에서 VPC 및 서브넷 기본 사항을 참조하세요.

DB 서브넷 그룹은 사용자가 VPC에서 만든 다음 클러스터에 대해 지정하는 서브넷의 모음입니다. 서브넷 그룹에서는 클러스터를 생성할 때 특정 VPC를 지정할 수 있습니다. `default` 서브넷 그룹을 사용하면 해당 그룹은 VPC 내의 모든 서브넷을 포괄합니다.

각 DB 서브넷 그룹은 지정된 리전에서 두 개 이상의 가용 영역에 서브넷이 있어야 합니다. VPC에서 DB 클러스터를 만들 때, DB 서브넷 그룹을 선택해야 합니다. Amazon DocumentDB는 해당 DB 서브넷 그룹과 사용자가 선호하는 가용 영역을 사용하여 해당 서브넷 내에서 클러스터와 연결할 서브넷과 IP 주소를 선택합니다. 기본 인스턴스가 실패할 경우 Amazon DocumentDB가 해당 복제본 인스턴스를 새 기본 인스턴스로 승격할 수 있습니다. 이때 이전의 기본 인스턴스가 위치했던 서브넷의 IP 주소를 사용하여 새 복제본 인스턴스를 생성할 수 있습니다.

Amazon DocumentDB가 VPC에 인스턴스를 생성할 때는 DB 서브넷 그룹에서 선택한 IP 주소를 사용하여 클러스터에 네트워크 인터페이스를 할당합니다. 기본 IP 주소가 장애 조치 중에 변경될 수 있기 때문에 반드시 DNS 이름을 사용하는 것이 좋습니다. 자세한 내용은 Amazon DocumentDB 엔드포인트 섹션을 참조하십시오.

VPC 및 서브넷 생성에 대한 자세한 내용은 Amazon Virtual 프라이빗 클라우드 사용 설명서의 VPC 및 서브넷으로 작업하기를 참조하세요.

주제

- Amazon DocumentDB 서브넷 그룹 생성
- Amazon DocumentDB 서브넷 그룹 설명
- Amazon DocumentDB 서브넷 그룹 수정
- Amazon DocumentDB 서브넷 그룹 삭제

Amazon DocumentDB 서브넷 그룹 생성

Amazon DocumentDB 클러스터를 생성하는 경우 Amazon VPC와 Amazon VPC 내 해당 서브넷 그룹을 선택해야 클러스터를 시작할 수 있습니다. 서브넷은 인스턴스를 시작하기 위해 사용할 가용 영역과 해당 가용 영역 내 IP 범위를 결정합니다.

서브넷 그룹은 Amazon DocumentDB 인스턴스를 시작하는 데 사용하고자 하는 가용 영역을 지정할 수 있는 명명된 서브넷(AZ) 집합입니다. 예를 들어, 인스턴스가 3개인 클러스터에서 각 인스턴스를 별도

의 AZ에 프로비저닝하여 고가용성을 최적화하는 것이 좋습니다. 따라서 단일 AZ가 실패하면 단일 인스턴스에만 영향을 미칩니다.

현재 Amazon DocumentDB 인스턴스는 최대 3개의 AZ로 프로비저닝할 수 있습니다. 서브넷 그룹에 서브넷이 3개 이상 있는 경우에도 Amazon DocumentDB 클러스터 생성에는 그 중 3개만 사용할 수만 있습니다. 따라서 서브넷 그룹을 생성할 때 인스턴스를 배포하려는 서브넷을 3개만 선택하는 것이 좋습니다.

예: 클러스터가 생성되고 Amazon DocumentDB가 AZ{1A, 1B 및 1C}를 선택합니다. AZ{1D}에서 인스턴스를 생성하려고 하면 API 직접 호출이 실패합니다. 하지만 특정 AZ를 지정하지 않고 인스턴스를 생성하기로 선택하면 Amazon DocumentDB가 사용자를 대신하여 AZ를 선택합니다. Amazon DocumentDB는 알고리즘을 사용하여 여러 AZ에 걸쳐 인스턴스의 부하를 분산하므로 고가용성을 확보할 수 있습니다. 기본적으로 세 개의 인스턴스가 프로비저닝되면 세 개의 AZ에 프로비저닝되고 모든 인스턴스가 단일 AZ에 프로비저닝되지 않습니다.

모범 사례

- 특별한 사유가 없으면 항상 서브넷 그룹 서브넷을 3개 생성합니다. 이렇게 하면 인스턴스가 세 AZ 간에 프로비저닝되므로 인스턴스가 세 개 이상인 클러스터가 더 높은 가용성을 획득할 수 있습니다.
- 항상 여러 AZ에 걸쳐 인스턴스를 분산하여 고가용성을 실현합니다. 단일 AZ에 클러스터의 모든 인스턴스를 배치하지 마십시오.
- 장애 조치 이벤트는 언제든지 발생할 수 있으므로 기본 인스턴스 또는 복제본 인스턴스가 항상 특정 AZ에 있다고 가정해서는 안 됩니다.

서브넷 그룹 생성 방법

AWS Management Console 또는를 사용하여 Amazon DocumentDB 서브넷 그룹을 AWS CLI 생성할 수 있습니다.

Using the AWS Management Console

다음 단계에 따라 Amazon DocumentDB 서브넷 그룹을 만듭니다.

Amazon DocumentDB 서브넷 그룹 생성

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 서브넷 그룹을 선택한 다음 생성을 선택합니다.

> **Tip**
>
> 화면 왼쪽에 탐색 창이 표시되지 않으면 페이지 왼쪽 상단 모서리에서 메뉴 아이콘 (≡) 을 선택합니다.

3. 서브넷 그룹 생성 페이지에서:

   a. 서브넷 그룹 세부 정보 섹션에서:

      i. 이름 — 서브넷 그룹에 대해 의미 있는 이름을 입력합니다.

      ii. 설명 - 서브넷 그룹에 대한 설명을 입력합니다:

   b. 서브넷 추가 섹션에서:

      i. VPC — 목록에서 이 서브넷 그룹에 대한 VPC를 선택합니다.

      ii. 다음 중 하나를 수행합니다.

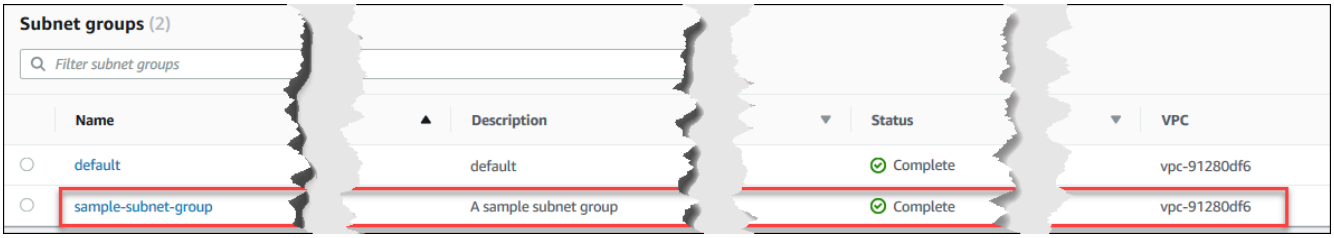
      - 선택된 VPC에 서브넷을 모두 포함하려면 이 VPC와 관련된 모든 서브넷 추가를 선택합니다.
      - 이 서브넷 그룹에 대한 서브넷을 지정하려면, 서브넷을 포함하려는 각 가용 영역에 대해 다음을 수행합니다. 2개 이상의 가용 영역을 포함해야 합니다.

        A. 가용 영역 - 목록에서 가용 영역을 선택합니다.

        B. 서브넷 — 목록에서 이 서브넷 그룹에 대해 선택된 가용 영역의 서브넷을 선택합니다.

        C. 서브넷 추가를 선택합니다.

4. 생성(Create)을 선택합니다. 새로 생성된 서브넷 그룹은 다른 서브넷 그룹과 함께 나열됩니다.



Using the AWS CLI

를 사용하여 서브넷 그룹을 생성하려면 먼저 사용 가능한 서브넷을 결정 AWS CLI해야 합니다. 다음 AWS CLI 작업을 실행하여 가용 영역과 해당 서브넷을 나열합니다.

파라미터:

- **--db-subnet-group** - 선택 사항. 특정 서브넷 그룹을 지정하면 가용 영역과 해당 그룹의 서브넷이 나열됩니다. 이 파라미터를 생략하면 모든 서브넷 그룹의 가용 영역과 서브넷이 나열됩니다. `default` 서브넷 그룹을 지정하면 모든 VPC의 서브넷이 나열됩니다.

Example

Linux, macOS, Unix의 경우:

```
aws docdb describe-db-subnet-groups \
    --db-subnet-group-name default \
    --query 'DBSubnetGroups[*].[DBSubnetGroupName,Subnets[*].
[SubnetAvailabilityZone.Name,SubnetIdentifier]]'
```

Windows의 경우:

```
aws docdb describe-db-subnet-groups ^
    --db-subnet-group-name default ^
    --query 'DBSubnetGroups[*].[DBSubnetGroupName,Subnets[*].
[SubnetAvailabilityZone.Name,SubnetIdentifier]]'
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
[
    [
        "default",
        [
            [
                "us-east-1a",
                "subnet-4e26d263"
            ],
            [
                "us-east-1c",
                "subnet-afc329f4"
            ],
            [
                "us-east-1e",
                "subnet-b3806e8f"
            ],
            [
                "us-east-1d",
```

```
                "subnet-53ab3636"
            ],
            [
                "us-east-1b",
                "subnet-991cb8d0"
            ],
            [
                "us-east-1f",
                "subnet-29ab1025"
            ]
        ]
    ]
]
```

이전 작업의 출력을 사용하여 새 서브넷 그룹을 만들 수 있습니다. 새 서브넷 그룹에는 최소 2개 가용 영역의 서브넷이 포함되어야 합니다.

파라미터:

- **--db-subnet-group-name** - 필수입니다. 이 서브넷 그룹의 이름입니다.
- **--db-subnet-group-description** - 필수입니다. 이 서브넷 그룹에 대한 설명입니다.
- **--subnet-ids** - 필수입니다. 이 서브넷 그룹에 포함하는 서브넷의 목록입니다. 예시: subnet-53ab3636.
- --태그 - 선택사항. 이 서브넷 그룹에 연결할 태그(키/값 페어) 목록입니다.

다음 코드는 subnet-4e26d263, subnet-afc329f4, subnet-b3806e8f의 3개 서브넷으로 구성된 서브넷 그룹 sample-subnet-group을 만듭니다.

Linux, macOS, Unix의 경우:

```
aws docdb create-db-subnet-group \
    --db-subnet-group-name sample-subnet-group \
    --db-subnet-group-description "A sample subnet group" \
    --subnet-ids subnet-4e26d263 subnet-afc329f4 subnet-b3806e8f \
    --tags Key=tag1,Value=One Key=tag2,Value=2
```

Windows의 경우:

```
aws docdb create-db-subnet-group ^
    --db-subnet-group-name sample-subnet-group ^
```

```
    --db-subnet-group-description "A sample subnet group" ^
    --subnet-ids subnet-4e26d263 subnet-afc329f4 subnet-b3806e8f ^
    --tags Key=tag1,Value=One Key=tag2,Value=2
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
    "DBSubnetGroup": {
        "DBSubnetGroupDescription": "A sample subnet group",
        "DBSubnetGroupName": "sample-subnet-group",
        "Subnets": [
            {
                "SubnetAvailabilityZone": {
                    "Name": "us-east-1a"
                },
                "SubnetIdentifier": "subnet-4e26d263",
                "SubnetStatus": "Active"
            },
            {
                "SubnetAvailabilityZone": {
                    "Name": "us-east-1c"
                },
                "SubnetIdentifier": "subnet-afc329f4",
                "SubnetStatus": "Active"
            },
            {
                "SubnetAvailabilityZone": {
                    "Name": "us-east-1e"
                },
                "SubnetIdentifier": "subnet-b3806e8f",
                "SubnetStatus": "Active"
            }
        ],
        "VpcId": "vpc-91280df6",
        "DBSubnetGroupArn": "arn:aws:rds:us-east-1:123SAMPLE012:subgrp:sample-
subnet-group",
        "SubnetGroupStatus": "Complete"
    }
}
```

Amazon DocumentDB 서브넷 그룹 설명

AWS Management Console 또는를 사용하여 Amazon DocumentDB 서브넷 그룹의 세부 정보를 AWS CLI 가져올 수 있습니다.

Using the AWS Management Console

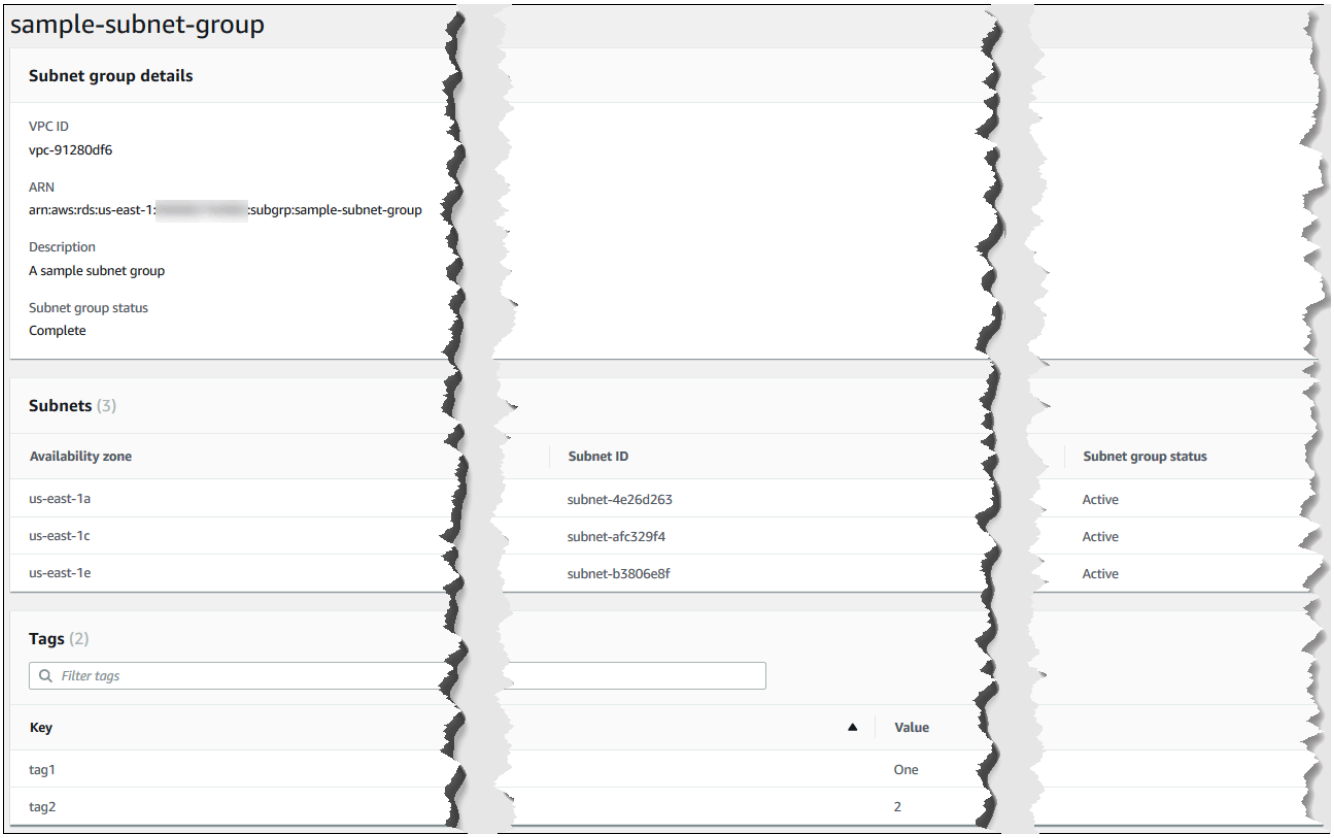
다음 절차에서는 Amazon DocumentDB 서브넷 그룹의 세부 정보 확인 방법을 볼 수 있습니다.

서브넷 그룹의 세부 정보를 찾으려면

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 서브넷 그룹을 선택합니다.

   > **Tip**
   > 화면 왼쪽에 탐색 창이 표시되지 않으면 페이지 왼쪽 상단 모서리에서 메뉴 아이콘 (≡)을 선택합니다.

3. 서브넷 그룹의 세부 정보를 확인하려면 해당 서브넷 그룹의 이름을 선택합니다.



Using the AWS CLI

Amazon DocumentDB 서브넷 그룹의 세부 정보를 찾으려면 다음 파라미터와 함께 `describe-db-subnet-groups` 작업을 사용합니다.

파라미터

- `--db-subnet=group-name` - 선택 사항. 포함된 경우, 명명된 서브넷 그룹의 세부 정보가 나열됩니다. 생략된 경우, 최대 100개 서브넷 그룹의 세부 정보가 나열됩니다.

Example

다음 코드는 Amazon DocumentDB 서브넷 그룹 생성 섹션에 생성한 `sample-subnet-group` 서브넷 그룹의 세부 정보를 나열합니다.

Linux, macOS, Unix의 경우:

```
aws docdb describe-db-subnet-groups \
    --db-subnet-group-name sample-subnet-group
```

Windows의 경우:

```
aws docdb describe-db-subnet-groups ^
    --db-subnet-group-name sample-subnet-group
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
    "DBSubnetGroup": {
        "DBSubnetGroupArn": "arn:aws:rds:us-east-1:123SAMPLE012:subgrp:sample-
subnet-group",
        "VpcId": "vpc-91280df6",
        "SubnetGroupStatus": "Complete",
        "DBSubnetGroupName": "sample-subnet-group",
        "Subnets": [
            {
                "SubnetAvailabilityZone": {
                    "Name": "us-east-1a"
                },
                "SubnetStatus": "Active",
                "SubnetIdentifier": "subnet-4e26d263"
            },
            {
                "SubnetAvailabilityZone": {
                    "Name": "us-east-1c"
                },
                "SubnetStatus": "Active",
                "SubnetIdentifier": "subnet-afc329f4"
            },
            {
                "SubnetAvailabilityZone": {
                    "Name": "us-east-1e"
                },
                "SubnetStatus": "Active",
                "SubnetIdentifier": "subnet-b3806e8f"
            }
        ],
        "DBSubnetGroupDescription": "A sample subnet group"
    }
}
```

Amazon DocumentDB 서브넷 그룹 수정

AWS Management Console 또는를 사용하여 서브넷 그룹의 설명을 AWS CLI 수정하거나 Amazon DocumentDB 서브넷 그룹에서 서브넷을 추가하거나 제거할 수 있습니다. 그러나 `default` 서브넷 그룹은 수정할 수 없습니다.

Using the AWS Management Console

AWS Management Console 를 사용하여 서브넷 그룹의 설명을 변경하거나 서브넷을 추가 및 제거할 수 있습니다. 완료 시 서브넷 그룹과 연결된 가용 영역은 2개 이상이어야 합니다.

서브넷 그룹을 수정하려면

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 서브넷 그룹을 선택합니다. 그런 다음 해당 서브넷 그룹 이름의 왼쪽에 있는 버튼을 선택합니다. `default` 서브넷 그룹은 수정할 수 없습니다.

   > **Tip**
   > 화면 왼쪽에 탐색 창이 표시되지 않으면 페이지 왼쪽 상단 모서리에서 메뉴 아이콘 (≡) 을 선택합니다.

3. 작업을 선택한 후 수정을 선택합니다.
4. 설명- 서브넷 그룹의 설명을 변경하려면 새 설명을 입력합니다.
5. 서브넷 그룹과 연결된 서브넷을 변경하려면 서브넷 추가 섹션에서 다음 중 하나 이상을 수행합니다.
   - 이 서브넷 그룹에서 서브넷을 모두 제거하려면 모두 제거를 선택합니다.
   - 이 서브넷 그룹에서 특정 서브넷을 제거하려면 제거하려는 각 서브넷에 대해 제거를 선택합니다.
   - 이 VPC에 연결된 서브넷을 모두 추가하려면 이 VPC와 관련된 모든 서브넷 추가를 선택합니다.
   - 이 서브넷 그룹에 특정 서브넷을 추가하려면 서브넷을 추가하려는 각 가용 영역에 대해 다음을 수행합니다.

     a. 가용 영역 - 목록에서 새 가용 영역을 선택합니다.

b. 서브넷 — 목록에서 이 서브넷 그룹에 대해 선택된 가용 영역의 서브넷을 선택합니다.

c. 서브넷 추가를 선택합니다.

6. 확인 대화 상자에서:

- 서브넷 그룹에 해당 수정 사항을 적용하려면 수정을 선택합니다.
- 서브넷 그룹을 변경 없이 유지하려면 취소를 선택합니다.

Using the AWS CLI

AWS CLI 를 사용하여 서브넷 그룹의 설명을 변경하거나 서브넷을 추가 및 제거할 수 있습니다. 완료 시 서브넷 그룹과 연결된 가용 영역은 2개 이상이어야 합니다. `default` 서브넷 그룹은 수정할 수 없습니다.

파라미터:

- `--db-subnet-group-name` - 필수입니다. 수정하는 Amazon DocumentDB 서브넷 그룹의 이름입니다.
- `--subnet-ids` - 필수입니다. 이 수정 사항의 적용이 완료된 후 서브넷 그룹에 포함하려는 모든 서브넷의 목록입니다.

> **Important**
>
> 현재 서브넷 그룹에 속해 있지만 이 목록에는 포함되지 않은 모든 서브넷은 서브넷 그룹에서 제거됩니다. 현재 서브넷 그룹에 포함된 모든 서브넷을 유지하려면 이 목록에 포함시켜야 합니다.

- `--db-subnet-group-description` - 선택 사항. 서브넷 그룹에 대한 설명입니다.

Example

다음 코드는 설명을 수정하고 기존 서브넷을 `subnet-991cb8d0`, `subnet-53ab3636`, `subnet-29ab1025` 서브넷으로 대체합니다.

Linux, macOS, Unix의 경우:

```
aws docdb modify-db-subnet-group \
    --db-subnet-group-name sample-subnet-group \
    --subnet-ids subnet-991cb8d0 subnet-53ab3636 subnet-29ab1025 \
```

```
    --db-subnet-group-description "Modified subnet group"
```

Windows의 경우:

```
aws docdb modify-db-subnet-group ^
    --db-subnet-group-name sample-subnet-group ^
    --subnet-ids subnet-991cb8d0 subnet-53ab3636 subnet-29ab1025 ^
    --db-subnet-group-description "Modified subnet group"
```

이 작업의 출력은 다음과 같습니다(JSON 형식). Amazon DocumentDB 서브넷 그룹 생성 섹션에 생성된 것과 동일한 서브넷 그룹입니다. 그러나 서브넷 그룹의 서브넷은 `modify-db-subnet-group` 작업에 나열된 서브넷으로 대체됩니다.

```
{
    "DBSubnetGroup": {
        "DBSubnetGroupArn": "arn:aws:rds:us-east-1:123SAMPLE012:subgrp:sample-
subnet-group",
        "DBSubnetGroupDescription": "Modified subnet group",
        "SubnetGroupStatus": "Complete",
        "Subnets": [
            {
                "SubnetAvailabilityZone": {
                    "Name": "us-east-1d"
                },
                "SubnetStatus": "Active",
                "SubnetIdentifier": "subnet-53ab3636"
            },
            {
                "SubnetAvailabilityZone": {
                    "Name": "us-east-1b"
                },
                "SubnetStatus": "Active",
                "SubnetIdentifier": "subnet-991cb8d0"
            },
            {
                "SubnetAvailabilityZone": {
                    "Name": "us-east-1f"
                },
                "SubnetStatus": "Active",
                "SubnetIdentifier": "subnet-29ab1025"
            }
        ],
```

```
        "VpcId": "vpc-91280df6",
        "DBSubnetGroupName": "sample-subnet-group"
    }
}
```

Amazon DocumentDB 서브넷 그룹 삭제

AWS Management Console 또는를 사용하여 Amazon DocumentDB 서브넷 그룹을 AWS CLI 삭제할 수 있습니다. 그러나 `default` 서브넷 그룹은 삭제할 수 없습니다.

Using the AWS Management Console

AWS Management Console 를 사용하여 서브넷 그룹을 삭제할 수 있습니다. `default` 서브넷 그룹은 삭제할 수 없습니다.

서브넷 그룹을 삭제하려면

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 서브넷 그룹을 선택합니다. 그런 다음 해당 서브넷 그룹 이름의 왼쪽에 있는 버튼을 선택합니다. `default` 서브넷 그룹은 삭제할 수 없습니다.

   > **Tip**
   > 화면 왼쪽에 탐색 창이 표시되지 않으면 페이지 왼쪽 상단 모서리에서 메뉴 아이콘 (≡) 을 선택합니다.

3. 작업을 선택한 후 삭제를 선택합니다.
4. 확인 대화 상자에서:
   - 서브넷 그룹을 삭제하려면 삭제를 선택합니다.
   - 서브넷 그룹을 유지하려면 취소를 선택합니다.

Using the AWS CLI

를 사용하여 Amazon DocumentDB 서브넷 그룹을 삭제하려면 다음 파라미터와 함께 `delete-db-subnet-group` 작업을 AWS CLI사용합니다.

파라미터

- `--db-subnet-group-name` - 필수입니다. 삭제할 Amazon DocumentDB 서브넷 그룹의 이름입니다. `default` 서브넷 그룹은 삭제할 수 없습니다.

Example

다음 코드를 사용하여 `sample-subnet-group`을 삭제할 수 있습니다.

Linux, macOS, Unix의 경우:

```
aws docdb delete-db-subnet-group \
    --db-subnet-group-name sample-subnet-group
```

Windows의 경우:

```
aws docdb delete-db-subnet-group ^
    --db-subnet-group-name sample-subnet-group
```

이 작업은 출력을 생성하지 않습니다.

Amazon DocumentDB 고가용성 및 복제

복제본 인스턴스를 사용하여 Amazon DocumentDB(MongoDB 호환)에서 고가용성 및 읽기 규모 조정을 구현할 수 있습니다. 단일 Amazon DocumentDB 클러스터는 1개의 기본 인스턴스와 최대 15개의 복제본 인스턴스를 지원합니다. 이러한 인스턴스는 해당 클러스터의 리전 내의 가용 영역에 두루 분산할 수 있습니다. 기본 인스턴스는 읽기 및 쓰기 트래픽을 허용하며, 복제본 인스턴스는 읽기 요청만 허용합니다.

클러스터 볼륨은 클러스터에 대한 여러 개의 데이터 사본으로 구성됩니다. 그러나 Amazon DocumentDB 클러스터의 기본 인스턴스 및 복제본에는 클러스터 볼륨 데이터가 단 하나의 논리 볼륨으로 표시됩니다. 복제본 인스턴스는 최종적으로 일관됩니다. 복제본 인스턴스 복제본 지연 시간을 최소화하여 쿼리 결과를 반환합니다. 이 지연 시간은 일반적으로 기본 인스턴스가 업데이트를 적용한 후 100밀리초 미만입니다. 데이터베이스 변경률에 따라 달라집니다. 즉, 데이터베이스의 쓰기 연산이 많은 기간에는 복제 지연 시간이 증가할 수 있습니다.

읽기 확장

Amazon DocumentDB 복제본은 클러스터 볼륨의 읽기 작업에 전적으로 사용되므로 읽기 규모 조정에 유용합니다. 쓰기 연산은 기본 인스턴스에서 관리합니다. 클러스터 볼륨은 클러스터에 있는 모든 인스턴스에 공유됩니다. 따라서 각 Amazon DocumentDB 복제본에 대한 데이터 사본을 복제하여 보관할 필요가 없습니다.

높은 가용성

Amazon DocumentDB 클러스터를 만들면 서브넷 그룹 내 가용 영역의 수에 따라(2개 이상이어야 함) Amazon DocumentDB가 가용 영역에서 인스턴스를 프로비저닝합니다. 클러스터에서 인스턴스를 생성할 때 Amazon DocumentDB가 서브넷 그룹의 가용 영역에 인스턴스를 자동으로 배포하여 클러스터의 균형을 잡습니다. 또한 이 작업을 통해 모든 인스턴스가 동일한 가용 영역에 위치하는 경우를 방지할 수 있습니다.

예제

이해를 돕기 위해 세 개의 가용 영역(AZ1, AZ2 및 AZ3)으로 구성된 서브넷 그룹이 있는 클러스터를 만든다고 가정해 보겠습니다.

클러스터의 첫 번째 인스턴스가 만들어지면 이는 기본 인스턴스로서 세 가용 영역 중 하나에 위치합니다. 이 예에서는 AZ1입니다. 두 번째로 만드는 인스턴스는 복제 인스턴스로서 나머지 두 가용 영역 중 하나, 즉 AZ2에 위치합니다. 세 번째 인스턴스는 복제 인스턴스이며 나머지 한 가용 영역, AZ3에 위치합니다. 추가로 인스턴스를 만들 경우 가용 영역에 배포하여 클러스터의 균형을 달성할 수 있습니다.

기본 인스턴스(AZ1)에 장애가 발생하면 장애 조치가 트리거되고 기존의 복제 인스턴스 중 하나가 기본 인스턴스로 승격됩니다. 기존의 기본 인스턴스가 복구되면 이는 프로비저닝되었던 동일 가용 영역(AZ1)에서 복제 인스턴스로 지정됩니다. 3개 인스턴스 클러스터를 프로비저닝하면 Amazon DocumentDB는 해당 3개 인스턴스 클러스터를 계속 보존합니다. Amazon DocumentDB는 수동 개입 없이 인스턴스 장애 감지, 장애 조치 및 복구를 자동으로 처리합니다.

Amazon DocumentDB의 장애 조치로 인스턴스를 복구하는 경우, 복구된 인스턴스는 처음 프로비저닝된 가용 영역에 그대로 위치합니다. 그러나 인스턴스의 역할은 기본에서 복제본으로 변경될 수 있습니다. 이렇게 함으로써 여러 번의 장애 조치로 모든 인스턴스가 동일한 가용 영역에 모이는 경우를 방지할 수 있습니다.

Amazon DocumentDB 복제본을 장애 조치 대상으로 지정할 수 있습니다. 즉 기본 인스턴스에 장애가 발생하면 지정된 Amazon DocumentDB 복제본이나 티어의 복제본이 기본 인스턴스로 승격됩니다. 기본 인스턴스에 대한 읽기/쓰기 요청이 예외로 인해 장애가 발생하는 동안에는 시스템이 짧게 중단됩니

다. Amazon DocumentDB 클러스터에 Amazon DocumentDB 복제본이 없는 경우에는 기본 인스턴스 장애 시 해당 인스턴스가 다시 생성됩니다. Amazon DocumentDB 복제본을 승격시키는 것이 기본 인스턴스를 다시 생성하는 것보다 훨씬 빠릅니다.

고가용성 시나리오에서는 Amazon DocumentDB 복제본을 1개 이상 생성하는 것이 좋습니다. 이러한 복제본은 인스턴스 클래스가 기본 인스턴스의 클래스와 동일해야 하고, 가용 영역이 Amazon DocumentDB 클러스터의 가용 영역과 달라야 합니다.

자세한 내용은 다음 자료를 참조하세요.

- Amazon DocumentDB 클러스터 내결함성에 대한 이해
- Amazon DocumentDB 장애 조치
  - 장애 조치 대상 제어

글로벌 클러스터를 통한 고가용성

여러에서 고가용성을 위해 Amazon DocumentDB 글로벌 클러스터를 설정할 AWS 리전수 있습니다. 각 글로벌 클러스터는 여러 리전에 걸쳐 있으므로 지연 시간이 짧은 글로벌 읽기 및의 중단으로 인한 재해 복구가 가능합니다 AWS 리전. Amazon DocumentDB는 기본 리전에서 각 보조 영역으로 모든 데이터 및 업데이트 복제를 자동으로 처리합니다.

복제본 추가

클러스터에 추가된 첫 번째 인스턴스는 기본 인스턴스입니다. 첫 번째 인스턴스 이후 추가된 모든 인스턴스는 복제본 인스턴스입니다. 클러스터에는 기본 인스턴스 외에 최대 15개의 복제본 인스턴스가 있을 수 있습니다.

를 사용하여 클러스터를 생성하면 AWS Management Console기본 인스턴스가 동시에 자동으로 생성됩니다. 클러스터 및 기본 인스턴스를 생성하는 것과 동시에 복제본을 생성하려면 Create replica in different zone(다른 영역에 복제본 만들기)을 선택합니다. 자세한 내용은 Amazon DocumentDB 클러스터 생성의 4.d 단계를 참조하십시오. Amazon DocumentDB 클러스터에 복제본을 추가하려면 클러스터에 Amazon DocumentDB 인스턴스 추가 단원을 참조하십시오.

를 사용하여 클러스터 AWS CLI 를 생성할 때 기본 인스턴스와 복제본 인스턴스를 명시적으로 생성해야 합니다. 자세한 내용은 다음 주제의 "사용 AWS CLI" 섹션을 참조하세요.

- Amazon DocumentDB 클러스터 생성
- 클러스터에 Amazon DocumentDB 인스턴스 추가

Amazon DocumentDB 장애 조치

특정 유형의 계획된 유지 관리 또는 예상치 못한 기본 노드 또는 가용 영역 장애의 경우와 같은 특정 경우에 Amazon DocumentDB(MongoDB 호환)는 장애를 감지하고 기본 노드를 대체합니다. 장애 조치를 실시하는 동안 쓰기 시간은 최소화됩니다. 이러한 경우는 기본 노드의 역할이 새 기본 노드를 생성하여 프로비저닝하는 대신 읽기 전용 복제본 중 하나에 장애 조치를 실행하기 때문에 발생합니다. 이 장애 탐지 및 복제본 승격을 통해 승격이 완료되는 즉시 새 기본 노드에 작성을 재개할 수 있습니다.

장애 조치가 작동하려면 클러스터에 최소 두 개의 인스턴스(기본 인스턴스 및 적어도 한 개의 복제본 인스턴스)가 있어야 합니다.

> **Note**
>
> 이 주제는 원본 Amazon DocumentDB 인스턴스 기반 클러스터에만 적용됩니다. 탄력적 또는 글로벌 클러스터에는 적용되지 않습니다.

장애 조치 대상 제어

Amazon DocumentDB에서는 장애 조치가 발생할 때 기본 인스턴스로 승격시킬 복제본 인스턴스를 제어하기 위한 수단으로 장애 조치 티어를 제공합니다.

장애 조치 티어

각 복제본 인스턴스는 장애 조치 티어(0–15)와 연결됩니다. 유지 관리 또는 예상치 못한 하드웨어 장애로 인해 장애 조치가 발생하면 기본 인스턴스는 우선 순위가 가장 높은 복제본(가장 낮은 번호의 티어)으로 장애 조치됩니다. 여러 복제본의 우선 순위 티어가 동일한 경우 이전의 기본 인스턴스의 크기와 가장 가까운 티어의 복제본에 기본 인스턴스가 장애 조치됩니다.

선택한 복제본 그룹에 대한 장애 조치 티어를 0(가장 높은 우선 순위)으로 설정하여 장애 조치에서 해당 그룹의 복제본 중 하나를 승격시키도록 할 수 있습니다. 우선 순위가 낮은 티어(높은 번호)를 이러한 복제본에 할당하여 장애 조치 시 특정 복제본이 기본 인스턴스로 승격되지 않도록 효과적으로 차단할 수 있습니다. 이는 애플리케이션에서 특정 복제본의 사용량이 높고 복제본 중 하나에 장애 조치하면 중요 애플리케이션에 부정적인 영향을 미치는 경우에 유용합니다.

인스턴스를 생성할 때 또는 나중에 수정하여 인스턴스의 장애 조치 티어를 설정할 수 있습니다. 인스턴스를 수정하여 인스턴스 장애 조치를 설정하면 장애 조치가 트리거되지 않습니다. 자세한 정보는 다음 주제를 참조하세요.

- 클러스터에 Amazon DocumentDB 인스턴스 추가

- Amazon DocumentDB 인스턴스 수정

장애 조치를 수동으로 시작할 때 두 가지 방법, 즉 앞에서 설명한 장애 조치 티어와 `--target-db-instance-identifier` 파라미터를 사용하여 기본 인스턴스로 승격되는 복제본 인스턴스를 제어할 수 있습니다.

`--target-db-instance-identifier`

테스트를 위해 `failover-db-cluster` 연산을 사용하여 장애 조치 이벤트를 강제할 수 있습니다. `--target-db-instance-identifier` 파라미터를 사용하여 기본 인스턴스로 승격할 복제본을 지정합니다. `--target-db-instance-identifier` 파라미터 사용이 장애 조치 우선 순위 티어에 우선합니다. `--target-db-instance-identifier` 파라미터를 지정하지 않은 경우 기본 장애 조치가 장애 조치 우선 순위 티어를 따릅니다.

장애 조치 중에 발생하는 일

장애 조치는 Amazon DocumentDB가 자동적으로 처리하므로 관리자의 개입 없이 애플리케이션에서 최대한 신속하게 데이터베이스 작업을 재개할 수 있습니다.

- 장애 조치 시 동일하거나 다른 가용 영역에 Amazon DocumentDB 복제본 인스턴스가 있는 경우 Amazon DocumentDB는 인스턴스의 정식 이름 레코드(CNAME)를 플립하여 정상 복제본을 가리키고, 이는 다시 새 기본 복제본으로 승격됩니다. 일반적으로 장애 조치는 처음부터 끝까지 30초 이내에 완료됩니다.
- Amazon DocumentDB 복제본 인스턴스(예: 단일 인스턴스 클러스터)가 없는 경우 Amazon DocumentDB는 원래 인스턴스와 동일한 가용 영역에 새 인스턴스 생성을 시도합니다. 이와 같은 원래 인스턴스 대체가 최선의 방법이며, 가용 영역에 크게 영향을 주는 문제가 발생하는 경우 등에는 성공하지 못할 수도 있습니다.

데이터베이스 연결이 끊어지는 경우 애플리케이션에서 연결을 다시 시도해야 합니다.

장애 조치 테스트

클러스터에 대한 장애 조치는 클러스터에 있는 Amazon DocumentDB 복제본(읽기 전용 인스턴스) 중 하나를 기본 인스턴스(클러스터 라이터)로 승격시킵니다.

기본 인스턴스가 실패하면 Amazon DocumentDB 복제본이 존재하는 경우 Amazon DocumentDB 복제본으로 Amazon DocumentDB가 자동적으로 장애 조치됩니다. 테스트를 위해 기본 인스턴스의 실패를 시뮬레이션하려는 경우 장애 조치를 강제할 수 있습니다. 클러스터의 각 인스턴스에는 자체적인 엔드

포인트 주소가 있습니다. 장애 조치가 완료되면 엔드포인트 주소를 사용하도록 기존 연결을 정리한 후 다시 설정해야 합니다.

장애 조치를 강제하려면 `failover-db-cluster` 연산을 해당 파라미터와 함께 사용합니다.

- `--db-cluster-identifier` - 필수입니다. 장애 조치할 클러스터의 이름입니다.
- `--target-db-instance-identifier`—선택 사항. 기본 인스턴스로 승격시킬 인스턴스의 이름입니다.

Example

다음 작업은 `sample-cluster` 클러스터의 장애 조치를 강제로 실행합니다. 새 기본 인스턴스를 만들 인스턴스를 지정하지 않아 Amazon DocumentDB는 장애 조치 티어 우선 순위에 따라 인스턴스를 선택합니다.

Linux, macOS, Unix의 경우:

```
aws docdb failover-db-cluster \
   --db-cluster-identifier sample-cluster
```

Windows의 경우:

```
aws docdb failover-db-cluster ^
   --db-cluster-identifier sample-cluster
```

다음 작업은 `sample-cluster-instance`를 기본 역할로 승격하도록 지정하여 `sample-cluster` 클러스터의 장애 조치를 강제로 실행합니다. 출력의 `"IsClusterWriter": true`를 참고하세요.

Linux, macOS, Unix의 경우:

```
aws docdb failover-db-cluster \
   --db-cluster-identifier sample-cluster \
   --target-db-instance-identifier sample-cluster-instance
```

Windows의 경우:

```
aws docdb failover-db-cluster ^
   --db-cluster-identifier sample-cluster ^
   --target-db-instance-identifier sample-cluster-instance
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
    "DBCluster": {
        "HostedZoneId": "Z2SUY0A1719RZT",
        "Port": 27017,
        "EngineVersion": "3.6.0",
        "PreferredMaintenanceWindow": "thu:04:05-thu:04:35",
        "BackupRetentionPeriod": 1,
        "ClusterCreateTime": "2018-06-28T18:53:29.455Z",
        "AssociatedRoles": [],
        "DBSubnetGroup": "default",
        "MasterUsername": "master-user",
        "Engine": "docdb",
        "ReadReplicaIdentifiers": [],
        "EarliestRestorableTime": "2018-08-21T00:04:10.546Z",
        "DBClusterIdentifier": "sample-cluster",
        "ReaderEndpoint": "sample-cluster.node.us-east-1.docdb.amazonaws.com",
        "DBClusterMembers": [
            {
                "DBInstanceIdentifier": "sample-cluster-instance",
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1,
                "IsClusterWriter": true
            },
            {
                "DBInstanceIdentifier": "sample-cluster-instance-00",
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1,
                "IsClusterWriter": false
            },
            {
                "DBInstanceIdentifier": "sample-cluster-instance-01",
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1,
                "IsClusterWriter": false
            }
        ],
        "AvailabilityZones": [
            "us-east-1b",
            "us-east-1c",
            "us-east-1a"
        ],
        "DBClusterParameterGroup": "default.docdb3.6",
```

```
        "Endpoint": "sample-cluster.node.us-east-1.docdb.amazonaws.com",
        "IAMDatabaseAuthenticationEnabled": false,
        "AllocatedStorage": 1,
        "LatestRestorableTime": "2018-08-22T21:57:33.904Z",
        "PreferredBackupWindow": "00:00-00:30",
        "StorageEncrypted": false,
        "MultiAZ": true,
        "Status": "available",
        "DBClusterArn": "arn:aws:rds:us-east-1:123456789012:cluster:sample-cluster",
        "VpcSecurityGroups": [
            {
                "Status": "active",
                "VpcSecurityGroupId": "sg-12345678"
            }
        ],
        "DbClusterResourceId": "cluster-ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    }
}
```

복제 지연

복제 지연은 일반적으로 50밀리초 이하입니다. 복제 지연이 증가하는 가장 일반적인 이유는 다음과 같습니다.

- 기본 복제본의 쓰기 속도가 높으면 읽기 전용 복제본이 기본 복제본보다 뒤쳐집니다.
- 장기 실행 쿼리(예: 대규모 순차 스캔, 집계 쿼리)와 수신되는 쓰기 복제 간의 읽기 전용 복제본 경합.
- 읽기 전용 복제본에 대한 동시 쿼리 수가 매우 많습니다.

복제 지연을 최소화하려면 다음 문제 해결 방법을 시도해 보십시오.

- 쓰기 속도가 높거나 CPU 사용률이 높으면 클러스터의 인스턴스를 스케일 업하는 것이 좋습니다.
- 읽기 전용 복제본에서 쿼리가 오래 실행되고 쿼리 대상 문서가 매우 자주 업데이트되는 경우, 장기 실행 쿼리를 변경하거나 기본/쓰기 복제본에 대해 실행하여 읽기 전용 복제본에서 충돌이 발생하지 않도록 하는 것이 좋습니다.
- 동시 쿼리 수가 매우 많거나 읽기 전용 복제본에서만 CPU 사용률이 높은 경우, 읽기 전용 복제본의 수를 스케일 아웃하여 워크로드를 분산하는 것도 또 다른 방법입니다.
- 복제 지연은 높은 쓰기 처리량과 오래 실행되는 쿼리의 결과이므로 DbClusterReplicalagMaximum CW 측정치를 느린 쿼리 로거 및 `WriteThroughput`/`WriteIOPS` 지표와 함께 활용하여 복제 지연 문제를 해결하는 것이 좋습니다.

일반적으로 클러스터 장애 조치로 인해 성능이 저하되지 않도록 모든 복제본의 인스턴스 유형을 동일하게 사용하는 것이 좋습니다.

스케일 업과 스케일 아웃 중 하나를 선택하는 경우(예: 작은 인스턴스 6개 vs. 대형 인스턴스 3개), DB 인스턴스당 버퍼 캐시가 더 커지므로 일반적으로 스케일 아웃 전에 먼저 스케일 업(대형 인스턴스) 하는 것이 좋습니다.

애플리케이션 기능에 영향을 미치기 전에, 복제본 인스턴스의 데이터가 얼마나 뒤처질 수 있는지(또는 "부실"하게 될 수 있는지) 사전에 복제 지연 경보를 설정하고 해당 임계값의 상한선을 설정해야 합니다. 일반적으로 일시적인 워크로드로 인해 경보가 울리기 전에 여러 데이터 포인트에 대해 복제 지연 임계값을 초과하는 것이 좋습니다.

> **Note**
>
> 또한 10초를 초과하는 복제 지연에 대해서는 다른 경보를 설정하는 것이 좋습니다. 여러 데이터 포인트에 대해 이 임계값을 초과할 경우 인스턴스를 스케일 업하거나 기본 인스턴스의 쓰기 처리량을 줄이는 것이 좋습니다.

Amazon DocumentDB 인덱스 관리

주제

- Amazon DocumentDB 인덱스 생성
- Amazon DocumentDB 인덱스 유지 관리

Amazon DocumentDB 인덱스 생성

Amazon DocumentDB에서 인덱스를 구축하려면 다음과 같이 다양한 결정을 내려야 합니다:

- 얼마나 빨리 완료해야 할까요?
- 빌드가 진행되는 동안 컬렉션에 액세스할 수 없나요?
- 빌드에 할당할 수 있는 인스턴스 컴퓨팅 파워는 얼마나 됩니까?
- 어떤 유형의 인덱스를 만들어야 하나요?

이 섹션에서는 이러한 질문에 답하는 데 도움이 되며, 인스턴스 기반 클러스터 컬렉션에서 Amazon DocumentDB 인덱스를 생성하는 데 필요한 명령과 모니터링 예제를 제공합니다.

지침

다음 지침에는 새 인덱스를 생성할 때의 기본 제한 및 구성 장단점이 포함되어 있습니다:

- Amazon DocumentDB 버전 지원 - 모든 Amazon DocumentDB 버전에서 단일작업자 인덱싱이 지원되지만, 다중 작업자 인덱싱은 Amazon DocumentDB 버전 4.0 및 5.0에서만 지원됩니다.
- 성능 트레이드오프 - 인덱스 생성 프로세스의 작업자 수를 늘리면 Amazon DocumentDB 데이터베이스의 기본 인스턴스에서 CPU 사용률과 읽기 IO가 증가합니다. 새 인덱스를 생성하는 데 필요한 리소스는 실행 중인 워크로드에 사용할 수 없습니다.
- Elastic 클러스터 - Amazon DocumentDB 엘라스틱 클러스터에서는 병렬 인덱싱이 지원되지 않습니다.
- 최대 작업자 - 구성할 수 있는 최대 작업자 수는 데이터베이스 클러스터의 기본 인스턴스 크기에 따라 다릅니다. 데이터베이스 클러스터의 기본 인스턴스에 있는 총 vCPU 수의 절반입니다. 예를 들어 vCPU가 64개 있는 db.r6g.16xlarge 인스턴스에서 최대 32개의 작업자를 실행할 수 있습니다.

  > **Note**
  >
  > 2xlarge 이하 인스턴스 클래스에서는 병렬 작업자가 지원되지 않습니다.

- 최소 작업자 - 구성할 수 있는 최소 작업자 수는 1개입니다. 인스턴스 기반 클러스터의 인덱스 생성에 대한 기본 설정은 작업자 2명입니다. 하지만 “작업자 스레드” 옵션을 사용하면 작업자 수를 1개로 줄일 수 있습니다. 이렇게 하면 단일 작업자로 프로세스가 실행됩니다.
- 인덱스 압축 - Amazon DocumentDB는 인덱스 압축을 지원하지 않습니다. 저장된 데이터 및 인덱스의 데이터 크기는 다른 옵션을 사용할 때보다 클 수 있습니다.
- 여러 컬렉션 인덱싱 - 데이터베이스 클러스터의 기본 인스턴스에 있는 vCPU의 절반을 구성된 작업자가 여러 컬렉션에서 인덱스를 생성하는 데 사용할 수 있습니다.
- 인덱스 유형 - Amazon DocumentDB에서 지원되는 인덱스 유형에 대한 전체 설명은 이 블로그 게시물을 참조하십시오.

시작

컬렉션에서 인덱스 생성을 시작하려면 `createIndexes` 명령을 사용합니다. 기본적으로 명령어는 인덱스 생성 프로세스의 속도를 두 배 증가시키는 두 개의 병렬 작업자를 실행합니다.

예를 들어, 다음 명령 프로세스는 문서의 “user_name” 필드에 대한 색인을 만들고 색인 처리 속도를 작업자 4개로 늘리는 방법을 보여줍니다:

1. 클러스터에서 두 개의 병렬 워커를 사용하여 인덱스를 생성합니다.

```
db.runCommand({"createIndexes":"test","indexes":[{"key": {"user_name":1},
 "name":"username_idx"}]})
```

2. 인덱스 생성 프로세스의 속도를 최적화하려면 db.runCommand createIndexes 메소드에서 '작업자 스레드' 옵션("workers":<number>)을 사용하여 워커 수를 지정할 수 있습니다.

   프로세스 속도를 병렬 작업자 4명으로 늘리십시오:

```
db.runCommand({"createIndexes":"test","indexes":[{"key": {"user_name":1},
 "name":"username_idx", "workers":4}]})
```

> **Note**
>
> 작업자 수가 많을수록 인덱스 생성 진행 속도가 빨라집니다. 하지만 작업자 수가 늘어날수록 기본 인스턴스의 vCPU 및 읽기 IO에 가해지는 부하도 증가합니다. 다른 워크로드를 저하시키지 않으면서 증가된 부담을 감당할 수 있을 만큼 클러스터를 충분히 프로비저닝해야 합니다.

인덱싱 진행 상태

인덱스 생성 프로세스는 컬렉션을 초기화하고, 스캔하고, 키를 정렬하고, 마지막으로 인덱스 빌더를 통해 키를 삽입하는 방식으로 진행됩니다. 프로세스를 포그라운드에서 실행할 경우 최대 6단계, 백그라운드에서 실행할 경우 최대 9단계로 구성됩니다. 단계별로 완료율, 스캔된 저장 블록의 총 수, 정렬된 키 및 삽입된 키와 같은 상태 메트릭을 볼 수 있습니다.

mongo 쉘의 db.currentOp() 명령을 사용하여 인덱싱 프로세스의 진행 상황을 모니터링할 수 있습니다. 마지막 단계를 100% 완료하면 모든 색인이 성공적으로 생성되었음을 알 수 있습니다.

```
db.currentOp({"command.createIndexes": { $exists : true } })
```

> **Note**
>
> 인덱싱 진행 상태 보기는 Amazon DocumentDB 5.0에서만 지원됩니다.

인덱스 빌드 유형

인덱스 빌드의 네 가지 유형은 다음과 같습니다.

- 포그라운드 - 포그라운드 인덱스 빌드는 인덱스가 생성될 때까지 다른 모든 데이터베이스 작업을 차단합니다. Amazon DocumentDB 포그라운드 빌드는 5단계로 구성되어 있습니다.
- 포그라운드(고유) - 단일 문서(고유) 포그라운드 인덱스 빌드는 일반 포그라운드 빌드와 같은 다른 데이터베이스 작업을 차단합니다. 기본 포그라운드 빌드와 달리 고유 빌드는 추가 단계 (정렬 키 2) 를 사용하여 중복 키를 찾습니다. 포그라운드 (고유) 빌드는 6단계로 구성되어 있습니다.
- 백그라운드 - 백그라운드 인덱스 빌드를 사용하면 인덱스가 생성되는 동안 다른 데이터베이스 작업을 포그라운드에서 실행할 수 있습니다. Amazon DocumentDB 백그라운드 빌드는 8단계로 구성되어 있습니다.
- 백그라운드(고유) - 단일 문서(고유) 백그라운드 인덱스 빌드를 사용하면 인덱스가 생성되는 동안 다른 데이터베이스 작업을 포그라운드에서 실행할 수 있습니다. 기본 백그라운드 빌드와 달리 고유 빌드는 추가 단계 (정렬 키 2) 를 사용하여 중복 키를 찾습니다. 백그라운드 (고유) 빌드는 9단계로 구성되어 있습니다.

인덱스 빌드 단계

단계	포그라운드	포그라운드 (고유)	배경	배경 (고유)
초기화 중	1	1	1	1
빌딩 인덱스: 초기화 중	2	2	2	2
빌드 인덱스: 컬렉션 스캔	3	3	3	3
빌딩 인덱스: 정렬 키 1	4	4	4	4
빌딩 인덱스: 정렬 키 2		5		5
빌딩 인덱스: 키 삽입	5	6	5	6

단계	포그라운드	포그라운드 (고유)	배경	배경 (고유)
검증: 인덱스 스캔			6	7
검증: 튜플 정렬			7	8
검증: 컬렉션 스캔			8	9

- 초기화 중 - createIndex가 인덱스 빌더를 준비하고 있습니다. 이 단계는 매우 간단해야 합니다.
- 인덱스 작성: 초기화 중 - 인덱스 빌더가 인덱스 생성을 준비 중입니다. 이 단계는 매우 간단해야 합니다.
- 인덱스 작성: 컬렉션 스캔 - 인덱스 빌더가 인덱스 키를 수집하기 위해 컬렉션 스캔을 수행합니다. 측정 단위는 “블록”입니다.

> **Note**
>
> 인덱스 빌드에 대해 둘 이상의 작업자가 구성된 경우 이 단계에 표시됩니다. 인덱스 빌드 프로세스 중에 여러 작업자를 사용하는 단계는 “스캐닝 컬렉션” 단계뿐입니다. 다른 모든 단계에는 작업자 한 명만 표시됩니다.

- 인덱스 작성: 정렬 키 1 - 인덱스 빌더가 수집된 인덱스 키를 정렬합니다. 측정 단위는 “키”입니다.
- 인덱스 작성: 정렬 키 2 - 인덱스 작성기가 죽은 튜플에 해당하는 수집된 인덱스 키를 정렬하고 있습니다. 이 단계는 고유한 인덱스 구축에만 존재합니다. 측정 단위는 “키”입니다.
- 인덱스 작성: 키 삽입 - 인덱스 빌더가 새 인덱스에 인덱스 키를 삽입합니다. 측정 단위는 “키”입니다.
- 유효성 검증: 인덱스 스캔 - createIndex는 인덱스를 스캔하여 검증이 필요한 키를 찾는 것입니다. 측정 단위는 “블록”입니다.
- 유효성 검증: 튜플 정렬 - createIndex는 인덱스 스캔 단계의 출력을 정렬합니다.
- 유효성 검증: 컬렉션 스캔 - createIndex는 컬렉션을 스캔하여 이전 두 단계에서 찾은 인덱스 키의 유효성을 검사합니다. 측정 단위는 “블록”입니다.

인덱스 빌드 출력 예제

아래 출력 예제 (포그라운드 인덱스 빌드) 에는 인덱스 생성 상태가 표시됩니다. “msg” 필드는 빌드의 단계와 완료율을 표시하여 빌드 진행 상황을 요약합니다. “작업자” 필드는 인덱스 빌드의 해당 단계에서 사용된 작업자 수를 나타냅니다. “progress” 필드에는 완료율을 계산하는 데 사용된 실제 수치가 표시됩니다.

> **Note**
>
> Amazon DocumentDB 버전 4.0에서는 “currentIndexBuildName”, “메시지” 및 “진행 상황” 필드가 지원되지 않습니다.

```
{
    "inprog" : [{
    …
        "command": {
            "createIndexes": "test",
            "indexes": [{
                "v": 2,
                "key": {
                    "user_name": 1
                },
                "name": "user_name_1"
            }],
            "lsid": {
                "id": UUID(“094d0fba-8f41-4373-82c3-7c4c7b5ff13b”)
            },
            "$db": "test"
        },
        "currentIndexBuildName": user_name_1,
        "msg": "Index Build: building index number_1, stage 6/6 building index:
 656860/1003520 (keys) 65%",
        "workers": 1,
        "progress": {
            "done": 656861,
            "total": 1003520
        },
    …
    ],

    "ok" : 1
```

```
}
```

Amazon DocumentDB 인덱스 유지 관리

를 사용한 인덱스 유지 관리 **reIndex**

reIndex는 인덱스를 재구축하는 데 사용되는 명령입니다. 일반적으로 인덱스가 손상되거나 비효율적일 때 사용됩니다. 시간이 지남에 따라 많은 업데이트, 삽입 또는 삭제로 인해 인덱스에 미사용 스페이스가 누적되어 성능이 저하될 수 있습니다. 리인덱싱은 사용하지 않는 스페이스를 제거하고 인덱스의 효율성을 복원하는 데 도움이 됩니다.

reIndex 지침

- reIndex는 현재 Amazon DocumentDB 5.0에서만 지원됩니다.
- Amazon DocumentDB는 백그라운드에서 단일 인덱스 reindex를 지원하므로 여러 작업자가 사용할 수 있습니다. 이전 인덱스는 reIndex 프로세스가 실행 중일 때 쿼리에서 사용할 수 있습니다.
- Amazon DocumentDB는 currentOp를 통한 인덱싱 진행률 보고서를 지원합니다. 인덱스 생성 중에 인덱스 빌드 단계 보기와 유사한 인덱스 구축 단계를 볼 수 있습니다. 유일한 차이점은 고유하든 아니든 reIndex에는 항상 8단계가 있다는 것입니다. '빌딩 인덱스: 정렬 키 2' 단계는 없습니다.
- reIndex는 createIndexes, dropIndexes, collMod 및 renameCollection과 같은 동일한 컬렉션에서 인덱스 관련 명령을 제외한 모든 명령과 동시에 실행할 수 있습니다.
- reIndex는 현재 텍스트, 지리 공간, 벡터 및 부분 인덱스에는 지원되지 않습니다.

reIndex 구축

다음 명령을 사용하여 인덱스를 재구축합니다.

```
db.runCommand({ reIndex: "collection-name", index: "index-name"})
```

필요에 따라 재구축 프로세스에 할당된 작업자 수를 제어할 수도 있습니다.

```
db.runCommand({ reIndex: "collection-name", index: "index-name", workers: number })
```

컬렉션 수준 문서 압축 관리

Amazon DocumentDB 컬렉션 수준 문서 압축을 사용하면 컬렉션에 있는 문서를 압축하여 스토리지 및 IO 비용을 줄일 수 있습니다. 압축된 문서의 스토리지 크기 및 압축 상태와 같은 압축 지표를 통해 스토

리지 이득을 측정하여 필요에 따라 컬렉션 수준에서 문서 압축을 활성화하고 압축 지표를 확인할 수 있습니다. Amazon DocumentDB는 LZ4 압축 알고리즘을 사용하여 문서를 압축합니다.

Amazon DocumentDB는 버전 5.0부터 문서 압축을 지원합니다. 다음 내용은 컬렉션 수준 문서 압축 함수입니다.

- 기본 동작 - 클러스터의 새 컬렉션에 대한 기본 압축 설정은 `default_collection_compression`의 클러스터 파라미터에 의해 결정됩니다. 이 파라미터는 기본적으로 "비활성화"되어 있습니다.
- 기존 컬렉션 압축 - `collMod` 명령을 사용하여 기존 컬렉션의 압축 설정을 변경할 수 있습니다.
- 압축 임계값 변경 - 기본 압축 임계값은 2KB 입니다. 이 값은 `createCollection` 명령을 사용하여 새 컬렉션에 대해 지정하고 `collMod` 명령을 사용하여 기존 컬렉션에서 변경할 수 있습니다.

> **Note**
>
> Amazon DocumentDB 문서 압축은 Amazon DocumentDB 버전 3.6 및 4.0에서 지원되지 않습니다.

주제

- 문서 압축 관리
- 문서 압축 모니터링

문서 압축 관리

컬렉션에서 문서 압축 활성화

다음과 같이 `db.createCollection()` 메서드를 사용하여 onAmazon DocumentDB 컬렉션을 생성하는 동안 문서 압축을 활성화합니다.

```
db.createCollection( sample_collection,{
    storageEngine : {
        documentDB: {
            compression:{enable: <true | false>}
        }
    }
})
```

클러스터에서 문서 압축 활성화

default_collection_compression의 클러스터 파라미터를 "활성화됨"으로 설정하여 클러스터 수준의 모든 새 컬렉션에 대해 문서 압축을 기본적으로 활성화할 수 있습니다. 이 파라미터가 "활성화됨"으로 설정되면 클러스터에서 새로 생성된 컬렉션에는 기본적으로 2KB의 압축 임계값으로 압축이 활성화됩니다.

기존 컬렉션 압축

collMod 작업을 사용하고 다음 storageEngine 구성을 지정하여 기존 컬렉션에 대한 압축 설정을 수정할 수도 있습니다. 이 명령을 사용한 변경 사항은 새로 삽입된/업데이트된 문서에만 적용되며 이전에 삽입된 문서에 대한 압축은 변경되지 않습니다.

```
db.runCommand({
    collMod: "orders",
    storageEngine: {
        documentDB: {compression: {enable: <true | false>} }
    }
})
```

압축 임계값 설정

기본적으로 압축된 컬렉션의 압축 임계값은 2032바이트입니다. 압축이 활성화된 새 컬렉션을 생성할 때 createCollection 명령에서 이 임계값을 설정할 수 있습니다.

```
db.createCollection( sample_collection, {
    storageEngine : {
        documentDB: {
            compression: {
                enable: true,
                threshold: <128 - 8000>
            }
        }
    }
})
```

collMod 작업을 사용하고 다음 storageEngine 구성을 지정하여 기존 압축 컬렉션에 대한 압축 임계값을 수정할 수도 있습니다.

```
db.runCommand({
    collMod: "orders",
```

```
    storageEngine: {
        documentDB: {
            compression: {
                enable: true,
                threshold: <128 - 8000>
            }
        }
    }
})
```

압축 임계값은 128~8,000바이트의 값으로만 설정할 수 있다는 점을 참조해 주세요. 또한 압축 임계값을 지정할 때 enable 옵션을 "true"로 설정해야 합니다.

문서 압축 모니터링

컬렉션이 압축되었는지 확인하고 다음과 같이 압축 비율을 계산할 수 있습니다.

mongo 쉘에서 db.printCollectionStats() 또는 db.collection.stats() 명령을 실행하여 압축 통계를 봅니다. 출력은 문서 압축의 스토리지 이득을 분석하기 위해 비교할 수 있는 기존 크기와 압축 크기를 보여줍니다. 이 예제에서는 "sample_collection"이라는 컬렉션에 대한 통계가 아래에 나와 있습니다. 1024*1024의 스케일링 인수는 아래에서 size 및 storageSize 값을 MB 단위로 출력하는 데 사용됩니다.

```
db.sample_collection.stats(1024*1024)
```

다음은 위의 명령에 대한 출력 예제입니다.

```
{
    "ns" : "test.sample_collection",
    "count" : 1000000,
    "size" : 3906.3,
    "avgObjSize" : 4096,
    "storageSize" : 1953.1,
    compression:{"enabled" : true,"threshold" : 2032},
    ...
}
```

- size - 문서 컬렉션의 원래 크기입니다.
- avgObjSize - 압축 전 평균 문서 크기를 소수점 첫째 자리까지 반올림한 값입니다. 측정 단위는 바이트입니다.

- storageSize - 압축 후 컬렉션의 스토리지 크기입니다. 측정 단위는 바이트입니다.
- 활성화됨 - 압축이 활성화 또는 비활성화되었는지 여부를 나타냅니다.

실제 압축 비율을 계산하려면 수집 크기를 스토리지 크기(크기/storageSize)로 나눕니다. 위의 예제에서 계산은 3906.3/1953.1이며, 이는 2:1 압축 비율로 변환됩니다.

Amazon DocumentDB 이벤트 관리

Amazon DocumentDB(MongoDB 호환)는 클러스터, 인스턴스, 스냅샷, 보안 그룹 및 클러스터 매개 변수 그룹과 관련된 이벤트 기록을 보관합니다. 여기에는 이벤트 날짜 및 시간, 이벤트의 원본 이름 및 유형, 이벤트 관련 메시지 등의 정보가 포함됩니다.

> **Important**
>
> 특정 관리 기능의 경우, Amazon DocumentDB는 Amazon RDS 및 Amazon Neptune과 공유하는 운영 기술을 사용한다. 영역 제한, 즉 영역 수준에서 관리되는 제한은 Amazon DocumentDB, Amazon RDS 및 Amazon Neptune 간에 공유됩니다. 자세한 내용은 리전별 할당량 단원을 참조하십시오.

주제

- Amazon DocumentDB 이벤트 범주 보기
- Amazon DocumentDB 이벤트 보기

Amazon DocumentDB 이벤트 범주 보기

각 Amazon DocumentDB 리소스 유형에는 관련될 수 있는 특정 유형의 이벤트가 있습니다. 작업을 사용하여 AWS CLI `describe-event-categories` 이벤트 유형과 Amazon DocumentDB 리소스 유형 간의 매핑을 볼 수 있습니다.

파라미터

- **`--source-type`**—선택 사항. `--source-type` 파라미터를 사용하여 특정 소스 유형의 이벤트 범주를 봅니다. 다음은 허용되는 값입니다:
  - `db-cluster`
  - `db-instance`

- db-parameter-group
- db-security-group
- db-cluster-snapshot

- **--filters**—선택 사항. Amazon DocumentDB에 대한 이벤트 범주를 보려면 필터--filter Name=engine,Values=docdb를 사용합니다.

Example

다음 코드는 클러스터와 연결된 이벤트 범주를 나열합니다.

Linux, macOS, Unix의 경우:

```
aws docdb describe-event-categories \
    --filter Name=engine,Values=docdb \
    --source-type db-cluster
```

Windows의 경우:

```
aws docdb describe-event-categories ^
    --filter Name=engine,Values=docdb ^
    --source-type db-cluster
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
    "EventCategoriesMapList": [
        {
            "EventCategories": [
                "notification",
                "failure",
                "maintenance",
                "failover"
            ],
            "SourceType": "db-cluster"
        }
    ]
}
```

다음 코드는 각 Amazon DocumentDB 원본 유형과 관련된 이벤트 범주를 나열합니다.

```
aws docdb describe-event-categories
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
    "EventCategoriesMapList": [
        {
            "SourceType": "db-instance",
            "EventCategories": [
                "notification",
                "failure",
                "creation",
                "maintenance",
                "deletion",
                "recovery",
                "restoration",
                "configuration change",
                "read replica",
                "backtrack",
                "low storage",
                "backup",
                "availability",
                "failover"
            ]
        },
        {
            "SourceType": "db-security-group",
            "EventCategories": [
                "configuration change",
                "failure"
            ]
        },
        {
            "SourceType": "db-parameter-group",
            "EventCategories": [
                "configuration change"
            ]
        },
        {
            "SourceType": "db-cluster",
            "EventCategories": [
                "notification",
                "failure",
```

```
                "maintenance",
                "failover"
            ]
        },
        {
            "SourceType": "db-cluster-snapshot",
            "EventCategories": [
                "backup"
            ]
        }
    ]
}
```

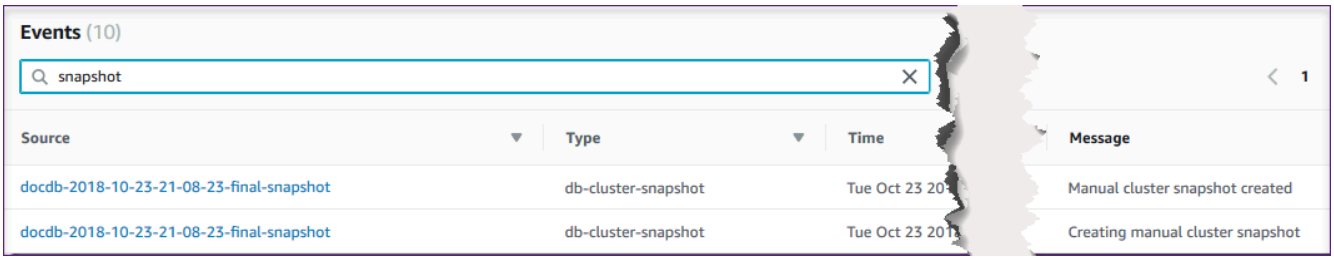
Amazon DocumentDB 이벤트 보기

지난 24시간 동안의 이벤트를 보여주는 Amazon DocumentDB 콘솔을 통해 Amazon DocumentDB 리소스에 대한 이벤트를 검색할 수 있습니다. 설명이벤트 AWS CLI 명령 또는 설명 이벤트 Amazon DocumentDB API 작업을 사용하여 Amazon DocumentDB 리소스에 대한 이벤트를 검색할 수도 있습니다. AWS CLI 또는 Amazon DocumentDB API를 사용하여 이벤트를 보는 경우 최대 지난 14일 동안의 이벤트를 검색할 수 있습니다.

Using the AWS Management Console

지난 24시간 동안 발생한 모든 Amazon DocumentDB 인스턴스 이벤트를 보려면

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 이벤트를 선택합니다. 사용 가능한 이벤트가 목록에 표시됩니다.
3. 필터 목록을 사용하여 유형별로 이벤트를 필터링합니다. 텍스트 상자에 단어를 입력하여 결과를 추가로 필터링할 수 있습니다. 예를 들어, 다음 스크린샷은 스냅샷 이벤트에 대한 모든 Amazon DocumentDB 이벤트를 필터링하는 것을 보여줍니다.



Using the AWS CLI

지난 7일 동안의 모든 Amazon DocumentDB 인스턴스 이벤트를 보려면 다음과 같이 하십시오

`--duration` 파라미터가 `10080` (10,080분)로 설정된 상태에서 설명 이벤트 AWS CLI 작업을 실행하여 지난 7일 동안의 모든 Amazon DocumentDB 인스턴스 이벤트를 볼 수 있습니다.

```
aws docdb describe-events --duration 10080
```

Amazon DocumentDB 이벤트 필터링

특정 Amazon DocumentDB 이벤트를 보려면 다음 매개변수를 사용하여 `describe-events` 작업을 수행합니다.

파라미터

- **`--filter`**—반환된 값을 Amazon DocumentDB 이벤트로 제한하는 데 필요합니다. Amazon DocumentDB의 모든 이벤트를 **`Name=engine,Values=docdb`** 필터링하는 데만 사용합니다.
- **`--source-identifier`**—선택 사항. 반환되는 이벤트에 대한 이벤트 소스의 식별자입니다. 생략하면 모든 소스의 이벤트가 결과에 포함됩니다.
- **`--source-type`**—선택 사항, 다만 `--source-identifier`이 제공되지 않을 경우, 필수 사항. `--source-identifier`를 제공한 경우 `--source-type`은 `--source-identifier` 유형에 동의해야 합니다. 다음은 허용되는 값입니다:
  - `db-cluster`
  - `db-instance`
  - `db-parameter-group`
  - `db-security-group`
  - `db-cluster-snapshot`

다음 예제에서는 모든 Amazon DocumentDB 이벤트를 나열합니다.

```
aws docdb describe-events --filters Name=engine,Values=docdb
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
    "Events": [
        {
```

```
            "SourceArn": "arn:aws:rds:us-east-1:123SAMPLE012:db:sample-cluster-
instance3",
            "Message": "instance created",
            "SourceType": "db-instance",
            "Date": "2018-12-11T21:17:40.023Z",
            "SourceIdentifier": "sample-cluster-instance3",
            "EventCategories": [
                "creation"
            ]
        },
        {
            "SourceArn": "arn:aws:rds:us-
east-1:123SAMPLE012:db:docdb-2018-12-11-21-08-23",
            "Message": "instance shutdown",
            "SourceType": "db-instance",
            "Date": "2018-12-11T21:25:01.245Z",
            "SourceIdentifier": "docdb-2018-12-11-21-08-23",
            "EventCategories": [
                "availability"
            ]
        },
        {
            "SourceArn": "arn:aws:rds:us-
east-1:123SAMPLE012:db:docdb-2018-12-11-21-08-23",
            "Message": "instance restarted",
            "SourceType": "db-instance",
            "Date": "2018-12-11T21:25:11.441Z",
            "SourceIdentifier": "docdb-2018-12-11-21-08-23",
            "EventCategories": [
                "availability"
            ]
        }
    ]
}
```

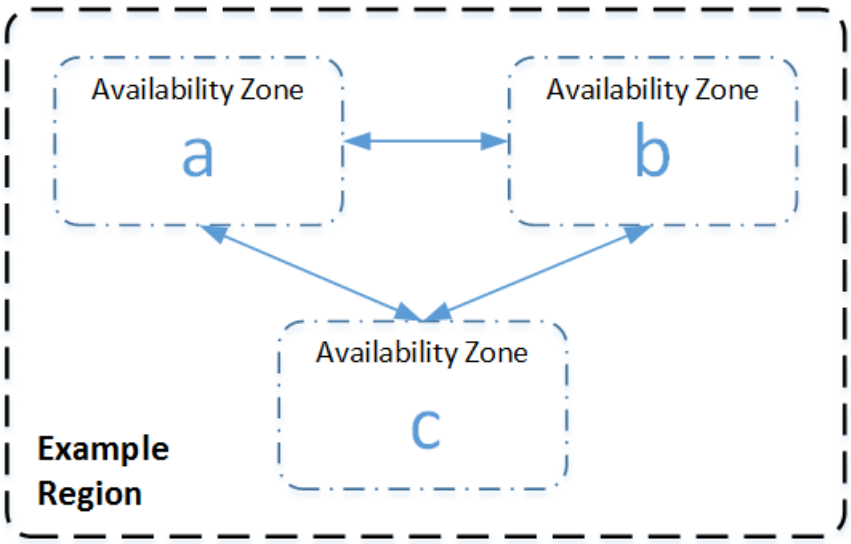
자세한 내용은 Amazon DocumentDB 이벤트 감사 단원을 참조하십시오.

리전 및 가용 영역 선택

Amazon 클라우드 컴퓨팅 리소스는 전 세계 여러 위치에서 호스팅됩니다. 이러한 위치는 AWS 리전 및 가용 영역으로 구성됩니다. 각 AWS 리전은 개별 지리 영역입니다. 각 리전은 가용 영역이라고 알려진 격리된 위치를 여러 개 가지고 있습니다. Amazon DocumentDB를 사용하면 인스턴스와 같은 리소스와

데이터를 여러 위치에 배치할 수 있습니다. 특별히 복제하지 않는 AWS 리전 한 리소스는 간에 복제되지 않습니다.

Amazon은 고급의 고가용성 데이터 센터를 운영하고 있습니다. 드물기는 하지만 동일한 위치에 있는 인스턴스의 가용성에 영향을 미치는 장애가 발생할 수도 있습니다. 그런 장애의 영향을 받는 단일한 위치에서 모든 인스턴스를 호스팅하는 경우에는 모든 인스턴스가 사용이 불가능해질 수 있습니다. 다음 다이어그램은 세 개의 가용 영역이 AWS 리전 있는를 보여줍니다.



각 리전은 서로 독립적이라는 점에 유의하십시오. 사용자의 모든 Amazon DocumentDB 활동(인스턴스 또는 사용할 수 있는 인스턴스 목록 생성 등)은 현재 기본 AWS 리전리전에서만 실행됩니다. `EC2_REGION` 환경 변수를 설정하여 콘솔의 기본 리전을 변경할 수 있습니다. 또는 AWS CLI에서 `--region` 파라미터를 사용하여 재정의할 수 있습니다. 자세한 내용은 구성 AWS Command Line Interface, 특히 환경 변수 및 명령줄 옵션에 대한 섹션을 참조하세요.

Amazon DocumentDB 콘솔을 사용하여 클러스터를 생성하고 다른 가용 영역에서 복제본을 생성하도록 선택할 경우 Amazon DocumentDB는 두 개의 인스턴스를 생성합니다. 가용 영역 하나에서 기본 인스턴스를 생성하고 다른 가용 영역에서 복제본 인스턴스를 생성합니다. 클러스터 볼륨은 항상 세 개의 가용 영역에 걸쳐 복제됩니다.

특정에서 Amazon DocumentDB 인스턴스를 생성하거나 사용하려면 해당 리전 서비스 엔드포인트를 AWS 리전사용합니다.

리전 가용성

Amazon DocumentDB는 다음 AWS 리전에서 사용할 수 있습니다.

Amazon DocumentDB에서 지원하는 지역

리전 이름	지역	가용 영역(컴퓨팅)
미국 동부(오하이오)	us-east-2	3
미국 동부(버지니아 북부)	us-east-1	6
미국 서부(오리건)	us-west-2	4
아프리카(케이프타운)	af-south-1	3
남아메리카(상파울루)	sa-east-1	3
아시아 태평양(홍콩)	ap-east-1	3
아시아 태평양(하이데라바드)	ap-south-2	3
아시아 태평양(뭄바이)	ap-south-1	3
아시아 태평양(서울)	ap-northeast-2	4
아시아 태평양(싱가포르)	ap-southeast-1	3
아시아 태평양(시드니)	ap-southeast-2	3
아시아 태평양(도쿄)	ap-northeast-1	3
캐나다(중부)	ca-central-1	3
중국(베이징) 리전	cn-north-1	3
중국(닝샤)	cn-northwest-1	3
유럽(프랑크푸르트)	eu-central-1	3
유럽(아일랜드)	eu-west-1	3
유럽(런던)	eu-west-2	3

리전 이름	지역	가용 영역(컴퓨팅)
유럽(밀라노)	eu-south-1	3
유럽(파리)	eu-west-3	3
유럽(스페인)	eu-south-2	3
유럽(스톡홀름)	eu-north-1	3
중동(UAE)	me-central-1	3
이스라엘(텔아비브)	il-central-1	3
AWS GovCloud(미국 서부)	us-gov-west-1	3
AWS GovCloud(미국 동부)	us-gov-east-1	3

기본적으로 Amazon DocumentDB 클러스터의 시간대는 협정 세계시(UTC)입니다.

특정 리전의 클러스터 및 인스턴스용 연결 엔드포인트를 찾는 방법에 대한 자세한 내용은 Amazon DocumentDB 엔드포인트에 대한 이해 섹션을 참조하십시오.

Amazon DocumentDB 클러스터 파라미터 그룹 관리

클러스터 파라미터 그룹에서 파라미터를 사용하여 Amazon DocumentDB 엔진 구성을 관리할 수 있습니다. 클러스터 파라미터 그룹은 Amazon DocumentDB 클러스터의 파라미터를 보다 쉽게 관리할 수 있는 Amazon DocumentDB 구성 값 모음입니다. 클러스터 파라미터 그룹은 클러스터의 모든 인스턴스에 적용되는 엔진 구성 값의 컨테이너 역할을 합니다.

이 단원에서는 클러스터 파라미터 그룹을 생성, 확인 및 수정하는 방법에 대해 설명합니다. 또한 지정된 클러스터와 연결된 클러스터 파라미터 그룹을 확인하는 방법을 보여줍니다.

주제

- Amazon DocumentDB 클러스터 파라미터 그룹 설명
- Amazon DocumentDB 클러스터 파라미터 그룹 생성
- Amazon DocumentDB 클러스터 파라미터 그룹 수정

- 사용자 지정 클러스터 파라미터 그룹을 사용하도록 Amazon DocumentDB 클러스터 수정
- Amazon DocumentDB 클러스터 파라미터 그룹 복사
- Amazon DocumentDB 클러스터 파라미터 그룹 재설정
- Amazon DocumentDB 클러스터 파라미터 그룹 삭제
- Amazon DocumentDB 클러스터 파라미터 참조

Amazon DocumentDB 클러스터 파라미터 그룹 설명

새 지역에서 첫 번째 Amazon DocumentDB 클러스터를 생성하거나 새 엔진을 사용할 때 `default` 클러스터 파라미터 그룹이 자동으로 생성됩니다. 동일한 지역에 생성되고 엔진 버전이 동일한 후속 클러스터는 `default` 클러스터 파라미터 그룹을 사용하여 생성됩니다.

주제

- Amazon DocumentDB 클러스터 파라미터 그룹의 세부 정보 설명
- Amazon DocumentDB 클러스터의 파라미터 그룹 확인

Amazon DocumentDB 클러스터 파라미터 그룹의 세부 정보 설명

지정된 클러스터 파라미터 그룹의 세부 정보를 설명하려면 AWS Management Console 또는 AWS Command Line Interface ()를 사용하여 다음 단계를 완료합니다AWS CLI.

Using the AWS Management Console

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 파라미터 그룹을 선택합니다.

   > **Tip**
   > 화면 왼쪽에 탐색 창이 표시되지 않으면 페이지 왼쪽 상단 모서리에서 메뉴 아이콘 (≡) 을 선택하십시오.

3. 클러스터 파라미터 그룹 창에서 세부 정보를 보려는 파라미터 그룹의 이름을 선택합니다.
4. 결과 페이지에는 파라미터 그룹의 파라미터, 최근 활동 및 태그가 표시됩니다.

- 클러스터 파라미터에서 파라미터의 이름, 현재 값, 허용되는 값, 파라미터의 수정 가능 여부, 적용 유형, 데이터 유형 및 설명을 볼 수 있습니다. 파라미터를 선택한 다음 클러스터 파라미터 섹션에서 편집을 선택하여 개별 파라미터를 수정할 수 있습니다. 자세한 내용은 Amazon DocumentDB 클러스터 파라미터 수정(을)를 참조하세요.
- 최근 이벤트에서 이 파라미터 그룹에 대한 가장 최근 이벤트를 볼 수 있습니다. 이 섹션의 검색 창을 사용하여 이러한 이벤트를 필터링할 수 있습니다. 자세한 내용은 Amazon DocumentDB 이벤트 관리(을)를 참조하세요.
- 태그에서 이 클러스터 파라미터 그룹에 있는 태그를 볼 수 있습니다. 태그 섹션에서 편집을 선택하여 태그를 추가하거나 제거할 수 있습니다. 자세한 내용은 Amazon DocumentDB 리소스 태그 지정(을)를 참조하세요.

Using the AWS CLI

`describe-db-cluster-parameter-groups` AWS CLI 명령을 사용하여 단일 클러스터 파라미터 그룹 또는 Amazon DocumentDB에 대해 보유한 모든 클러스터 파라미터 그룹의 Amazon 리소스 이름(ARN), 패밀리, 설명 및 이름을 볼 수 있습니다. 명령을 사용하여 단일 클러스터 파라미터 그룹 내에서 파라미터와 해당 세부 정보를 `describe-db-cluster-parameters` AWS CLI 볼 수도 있습니다.

- **`--describe-db-cluster-parameter-groups`** - 모든 클러스터 파라미터 그룹 및 세부 정보 목록을 확인합니다.
  - **`--db-cluster-parameter-group-name`** - 선택 사항. 설명하려는 클러스터 파라미터 그룹의 이름입니다. 이 파라미터를 생략할 경우 모든 클러스터 파라미터 그룹을 설명합니다.
- **`--describe-db-cluster-parameters`** - 파라미터 그룹 내의 모든 파라미터와 해당 값을 나열합니다.
  - **`--db-cluster-parameter-group name`** - 필수입니다. 설명하려는 클러스터 파라미터 그룹의 이름입니다.

Example

다음 코드에는 최대 100개의 클러스터 파라미터 그룹과 해당 ARN, 패밀리, 설명 및 이름이 나열되어 있습니다.

```
aws docdb describe-db-cluster-parameter-groups
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
          "DBClusterParameterGroups": [
              {
                  "DBClusterParameterGroupArn": "arn:aws:rds:us-
east-1:012345678912:cluster-pg:default.docdb4.0",
                  "DBParameterGroupFamily": "docdb4.0",
                  "Description": "Default cluster parameter group for docdb4.0",
                  "DBClusterParameterGroupName": "default.docdb4.0"
              },
              {
                  "DBClusterParameterGroupArn": "arn:aws:rds:us-
east-1:012345678912:cluster-pg:sample-parameter-group",
                  "DBParameterGroupFamily": "docdb4.0",
                  "Description": "Custom docdb4.0 parameter group",
                  "DBClusterParameterGroupName": "sample-parameter-group"
              }
          ]
}
```

Example

다음 코드에는 sample-parameter-group에 대한 ARN, 패밀리, 설명 및 이름이 나열되어 있습니다.

Linux, macOS 또는 Unix의 경우:

```
aws docdb describe-db-cluster-parameter-groups \
          --db-cluster-parameter-group-name sample-parameter-group
```

Windows의 경우:

```
aws docdb describe-db-cluster-parameter-groups ^
          --db-cluster-parameter-group-name sample-parameter-group
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
          "DBClusterParameterGroups": [
              {
                  "DBClusterParameterGroupArn": "arn:aws:rds:us-
east-1:123456789012:cluster-pg:sample-parameter-group",
                  "Description": "Custom docdb4.0 parameter group",
```

```
                "DBParameterGroupFamily": "docdb4.0",
                "DBClusterParameterGroupName": "sample-parameter-group"
            }
        ]
}
```

Example

다음 코드에는 sample-parameter-group의 파라미터 값이 나열되어 있습니다.

Linux, macOS 또는 Unix의 경우:

```
aws docdb describe-db-cluster-parameters \
    --db-cluster-parameter-group-name sample-parameter-group
```

Windows의 경우:

```
aws docdb describe-db-cluster-parameters ^
    --db-cluster-parameter-group-name sample-parameter-group
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
   "Parameters": [
         {
            "ParameterName": "audit_logs",
            "ParameterValue": "disabled",
            "Description": "Enables auditing on cluster.",
            "Source": "system",
            "ApplyType": "dynamic",
            "DataType": "string",
            "AllowedValues": "enabled,disabled",
            "IsModifiable": true,
            "ApplyMethod": "pending-reboot"
         },
         {
            "ParameterName": "change_stream_log_retention_duration",
            "ParameterValue": "17777",
```

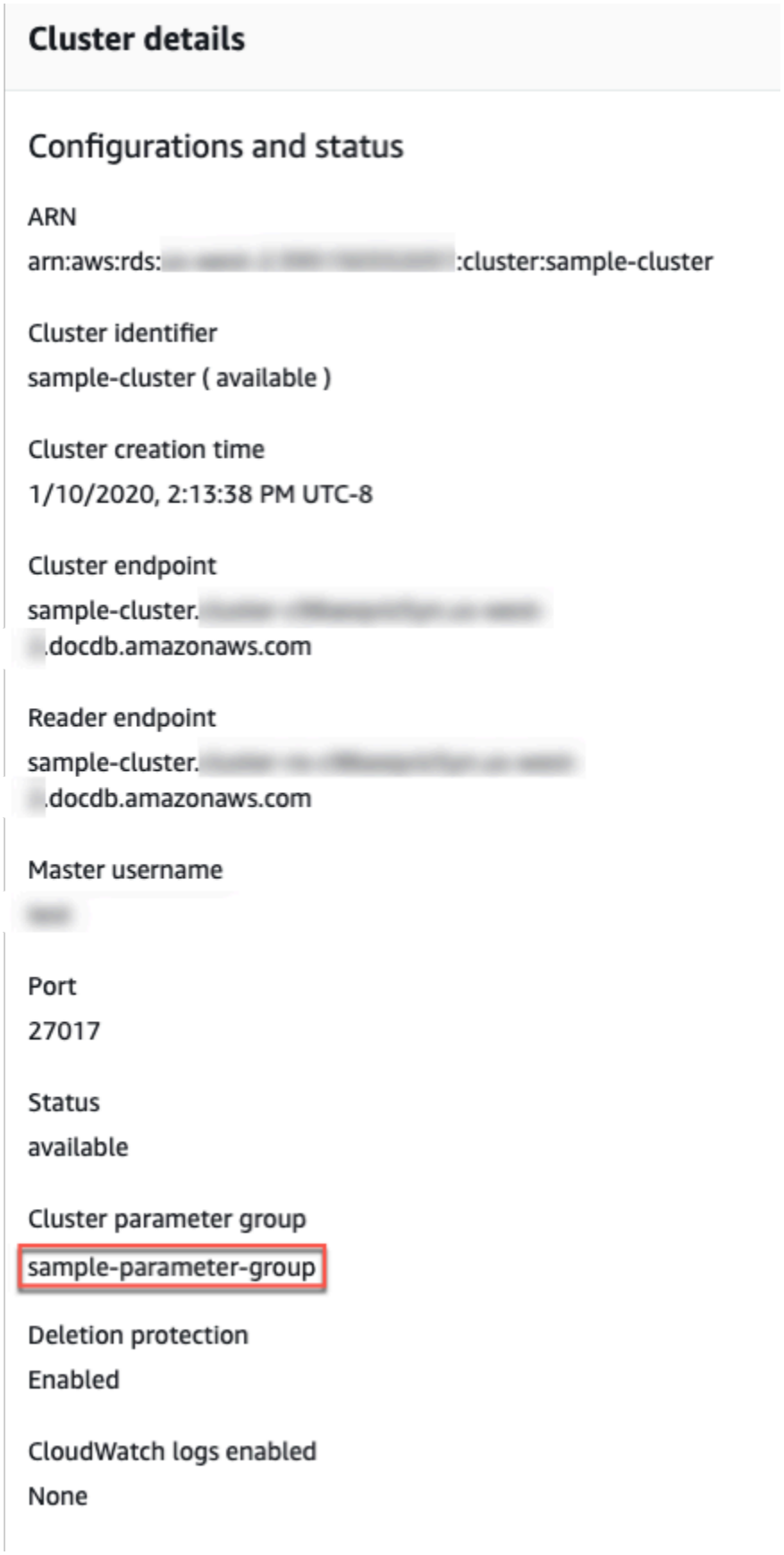
```
            "Description": "Duration of time in seconds that the change stream log
 is retained and can be consumed.",
            "Source": "user",
            "ApplyType": "dynamic",
            "DataType": "integer",
            "AllowedValues": "3600-86400",
            "IsModifiable": true,
            "ApplyMethod": "pending-reboot"
         }
   ]
}
```

Amazon DocumentDB 클러스터의 파라미터 그룹 확인

특정 클러스터와 연결된 파라미터 그룹을 확인하려면 AWS Management Console 또는를 사용하여 다음 단계를 완료합니다 AWS CLI.

Using the AWS Management Console

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 좌측 탐색 창에서 클러스터를 선택합니다.
3. 클러스터 목록에서 원하는 클러스터 이름을 선택합니다.
4. 결과 페이지에는 선택한 클러스터의 세부 정보가 표시됩니다. 클러스터 세부 정보까지 아래로 스크롤합니다. 해당 섹션의 하단에 있는 클러스터 파라미터 그룹 아래에서 파라미터 그룹의 이름을 찾습니다.


Cluster details
Configurations and status
ARN
arn:aws:rds: :cluster:sample-cluster
Cluster identifier
sample-cluster (available)
Cluster creation time
1/10/2020, 2:13:38 PM UTC-8
Cluster endpoint
sample-cluster.
.docdb.amazonaws.com
Reader endpoint
sample-cluster.
.docdb.amazonaws.com
Master username
Port
27017
Status
available
Cluster parameter group
sample-parameter-group
Deletion protection
Enabled
CloudWatch logs enabled
None

Using the AWS CLI

다음 AWS CLI 코드는 클러스터를 관리하는 파라미터 그룹을 결정합니다`sample-cluster`.

```
aws docdb describe-db-clusters \
      --db-cluster-identifier sample-cluster \
      --query 'DBClusters[*].[DBClusterIdentifier,DBClusterParameterGroup]'
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
[
       [
           "sample-cluster",
           "sample-parameter-group"
       ]
]
```

Amazon DocumentDB 클러스터 파라미터 그룹 생성

`default.docdb5.0`, `default.docdb4.0` 또는 `default.docdb3.6` 등의 기본 클러스터 파라미터 그룹은 새 엔진 버전을 사용하여 새 리전에 클러스터를 생성할 때 생성됩니다. 이 리전에 생성되고 엔진 버전이 동일한 후속 클러스터는 `default` 클러스터 파라미터 그룹을 상속합니다. 한 번 생성된 `default` 파라미터 그룹은 삭제하거나 이름을 바꿀 수 없습니다. 선호하는 파라미터 값을 사용하여 사용자 지정 파라미터 그룹을 생성하고 Amazon DocumentDB 클러스터에 연결하여 클러스터 인스턴스의 엔진 동작을 수정할 수 있습니다.

다음 절차에서는 사용자 지정 클러스터 파라미터 그룹을 생성하는 방법을 안내합니다. 그런 다음 해당 파라미터 그룹 내에서 파라미터를 수정할 수 있습니다.

> **Note**
>
> 클러스터 파라미터 그룹을 생성한 후 해당 특정 파라미터 그룹을 사용하려면 5분 이상 기다려야 합니다. 그러면 클러스터 파라미터 그룹이 새 클러스터에 사용되기 전에 Amazon DocumentDB에서 `create` 작업을 완전히 마칠 수 있습니다. AWS Management Console 또는 `describe-db-cluster-parameter-groups` AWS CLI 작업을 사용하여 클러스터 파라미터 그룹이 생성되었는지 확인할 수 있습니다. 자세한 내용은 Amazon DocumentDB 클러스터 파라미터 그룹 설명 단원을 참조하십시오.

Using the AWS Management Console

클러스터 파라미터 그룹을 생성하려면

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 파라미터 그룹을 선택합니다.

   > **Tip**
   >
   > 화면 왼쪽에 탐색 창이 표시되지 않으면 페이지 왼쪽 상단 모서리에서 메뉴 아이콘 (≡) 을 선택하십시오.

3. 클러스터 파라미터 그룹 창에서 생성을 선택합니다.
4. 클러스터 파라미터 그룹 만들기 창에서 다음을 입력합니다.

   a. 그룹 이름 - 클러스터 파라미터 그룹의 이름을 입력합니다. 예: `sample-parameter-group`. 클러스터 파라미터 그룹에는 다음과 같은 명명 제약 조건이 있습니다.

      - [1-255]자 길이의 영숫자 문자입니다.
      - 첫 번째 문자는 글자이어야 합니다.
      - 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다.

   b. 설명 - 이 클러스터 파라미터 그룹에 대한 설명을 제공합니다.

5. 클러스터 파라미터 그룹을 생성하려면 생성을 선택합니다. 작업을 취소하려면 취소를 선택합니다.
6. 생성을 선택하면 클러스터 파라미터 그룹이 성공적으로 생성되었는지 확인하기 위해 페이지 상단에 다음 텍스트가 나타납니다.

```
Successfully created cluster parameter group 'sample-parameter-group'.
```

Using the AWS CLI

Amazon DocumentDB 4.0 클러스터에 AWS CLI `create-db-cluster-parameter-group` 대한 새 클러스터 파라미터 그룹을 생성하려면 다음 파라미터와 함께 작업을 사용합니다.

- **--db-cluster-parameter-group-name** - 사용자 지정 클러스터 파라미터 그룹의 이름입니다. 예: `sample-parameter-group`.
- **--db-cluster-parameter-group-family** - 사용자 지정 클러스터 파라미터 그룹의 템플릿으로 사용되는 클러스터 파라미터 그룹 패밀리입니다. 현재 이 값은 `docdb4.0`이어야 합니다.
- **--description** - 이 클러스터 파라미터 그룹에 대해 사용자가 제공한 설명입니다. 다음 예에는 "`Custom docdb4.0 parameter group`"이 사용됩니다.

Linux, macOS 또는 Unix의 경우:

Example

```
aws docdb create-db-cluster-parameter-group \
 --db-cluster-parameter-group-name sample-parameter-group \
 --db-parameter-group-family docdb4.0 \
 --description "Custom docdb4.0 parameter group"
```

Windows의 경우:

```
aws docdb create-db-cluster-parameter-group ^
 --db-cluster-parameter-group-name sample-parameter-group ^
 --db-parameter-group-family docdb4.0 ^
 --description "Custom docdb4.0 parameter group"
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
    "DBClusterParameterGroup": {
        "DBClusterParameterGroupName": "sample-parameter-group",
        "DBParameterGroupFamily": "docdb4.0",
        "Description": "Custom docdb4.0 parameter group",
        "DBClusterParameterGroupArn": "sample-parameter-group-arn"
    }
}
```

Amazon DocumentDB 클러스터 파라미터 그룹 수정

이 단원에서는 사용자 지정 Amazon DocumentDB 파라미터 그룹을 수정하는 방법에 대해 설명합니다. Amazon DocumentDB에서는 새 리전에 새 엔진 버전으로 클러스터를 처음 생성할 때 생성되는

default 클러스터 파라미터 그룹을 수정할 수 없습니다. Amazon DocumentDB 클러스터가 기본 클러스터 파라미터 그룹을 사용하고 있는 경우, 이 그룹의 값을 수정하려면 먼저 새 파라미터 그룹 생성을 하거나 기존 파라미터 그룹 복사를 하여 수정한 다음 수정된 파라미터 그룹을 클러스터에 적용해야 합니다.

사용자 지정 클러스터 파라미터 그룹을 수정하려면 다음 단계를 완료합니다. 수정 작업이 전파되는 데 시간이 걸릴 수 있습니다. 수정된 클러스터 파라미터 그룹을 클러스터에 연결하기 전에 사용할 수 있을 때까지 기다리십시오. AWS Management Console 또는 AWS CLI describe-db-cluster-parameters 작업을 사용하여 클러스터 파라미터 그룹이 수정되었는지 확인할 수 있습니다. 자세한 내용은 클러스터 파라미터 그룹 설명 단원을 참조하십시오.

Using the AWS Management Console

사용자 지정 Amazon DocumentDB 파라미터 그룹을 수정하려면 다음 단계를 수행합니다. default 파라미터 그룹은 수정할 수 없습니다. default 파라미터 그룹의 값을 수정하려면 기본 클러스터 파라미터 그룹을 복사하여 수정한 다음 수정된 파라미터 그룹을 클러스터에 적용하면 됩니다. 클러스터에 파라미터 그룹을 적용하는 방법에 대한 자세한 내용은 Amazon DocumentDB 클러스터 수정(을)를 참조하세요.

사용자 지정 클러스터 파라미터 그룹을 수정하려면

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 콘솔 왼쪽의 탐색 창에서 파라미터 그룹을 선택합니다. 파라미터 그룹 목록에서 수정하고자 하는 파라미터 그룹의 이름을 선택합니다.

   > **Tip**
   > 화면 왼쪽에 탐색 창이 표시되지 않으면 페이지 왼쪽 상단 모서리에서 메뉴 아이콘 (≡) 을 선택하십시오.

3. 수정하고자 하는 파라미터 그룹의 각 파라미터에 대해 다음을 수행합니다.

   a. 수정하려는 파라미터를 찾은 다음, 수정 가능 열에 true로 나열되어 있는지 확인하여 해당 파라미터가 수정 가능한지 확인합니다.

   b. 수정할 수 있는 경우 해당 파라미터를 선택하고 콘솔 페이지의 오른쪽 상단에서 편집을 선택합니다.

c. 수정 **<parameter-name>** 대화 상자에서 원하는 대로 변경합니다. 그런 다음 Modify cluster parameter(클러스터 파라미터 수정)를 선택하거나 취소를 선택하여 변경 사항을 무시합니다.

Using the AWS CLI

AWS CLI를 사용하여 사용자 지정 Amazon DocumentDB 클러스터 파라미터 그룹에서 수정 가능한 파라미터의 `ParameterValue`, `Description` 또는 `ApplyMethod`를 수정할 수 있습니다. 기본 클러스터 파라미터 그룹을 직접 수정할 수는 없습니다.

사용자 지정 클러스터 파라미터 그룹의 파라미터를 수정하려면 다음 파라미터와 함께 `modify-db-cluster-parameter-group` 작업을 사용합니다.

- **`--db-cluster-parameter-group-name`** - 필수입니다. 수정하려는 클러스터 파라미터 그룹의 이름입니다.
- **`--parameters`** - 필수입니다. 수정 중인 파라미터입니다. Amazon DocumentDB 클러스터의 모든 인스턴스에 적용되는 파라미터 목록은 Amazon DocumentDB 클러스터 파라미터 참조(을)를 참조하세요. 각 파라미터 요소는 다음을 포함해야 합니다.
  - **`ParameterName`** - 수정 중인 파라미터의 이름입니다.
  - **`ParameterValue`** - 이 파라미터의 새 값입니다.
  - **`ApplyMethod`** - 파라미터에 대한 변경 사항을 적용할 방법입니다. 허용된 값은 `immediate` 및 `pending-reboot`입니다.

> **Note**
>
> `static`의 `ApplyType` 파라미터에는 `pending-reboot`의 `ApplyMethod`이 있어야 합니다.

Example - 파라미터 값 수정

이 예에서는 `sample-parameter-group`의 파라미터 값을 나열하고 `tls` 파라미터를 수정합니다. 그런 다음 5분 후에 다시 `sample-parameter-group`의 파라미터 값을 나열하여 변경된 파라미터 값을 확인합니다.

1. `sample-parameter-group`의 파라미터 및 해당 값을 나열합니다.

   Linux, macOS 또는 Unix의 경우:

```
aws docdb describe-db-cluster-parameters \
       --db-cluster-parameter-group-name sample-parameter-group
```

Windows의 경우:

```
aws docdb describe-db-cluster-parameters ^
       --db-cluster-parameter-group-name sample-parameter-group
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
       "Parameters": [
           {
               "Source": "system",
               "ApplyType": "static",
               "AllowedValues": "disabled,enabled",
               "ParameterValue": "enabled",
               "ApplyMethod": "pending-reboot",
               "DataType": "string",
               "ParameterName": "tls",
               "IsModifiable": true,
               "Description": "Config to enable/disable TLS"
           },
           {
               "Source": "user",
               "ApplyType": "dynamic",
               "AllowedValues": "disabled,enabled",
               "ParameterValue": "enabled",
               "ApplyMethod": "pending-reboot",
               "DataType": "string",
               "ParameterName": "ttl_monitor",
               "IsModifiable": true,
               "Description": "Enables TTL Monitoring"
           }
       ]
}
```

2. 해당 값이 `disabled`가 되도록 `tls` 파라미터를 수정합니다.

   `ApplyType`은 `static`이므로 `ApplyMethod`를 수정할 수 없습니다.

Linux, macOS 또는 Unix의 경우:

```
aws docdb modify-db-cluster-parameter-group \
       --db-cluster-parameter-group-name sample-parameter-group \
       --parameters
 "ParameterName"=tls,"ParameterValue"=disabled,"ApplyMethod"=pending-reboot
```

Windows의 경우:

```
aws docdb modify-db-cluster-parameter-group ^
       --db-cluster-parameter-group-name sample-parameter-group ^
       --parameters
 "ParameterName"=tls,"ParameterValue"=disabled,"ApplyMethod"=pending-reboot
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
       "DBClusterParameterGroupName": "sample-parameter-group"
   }
```

3. 적어도 5분을 기다립니다.
4. `sample-parameter-group`의 파라미터 값을 나열하여 `tls` 파라미터가 수정되었는지 확인합니다.

Linux, macOS 또는 Unix의 경우:

```
aws docdb describe-db-cluster-parameters \
       --db-cluster-parameter-group-name sample-parameter-group
```

Windows의 경우:

```
aws docdb describe-db-cluster-parameters ^
       --db-cluster-parameter-group-name sample-parameter-group
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
       "Parameters": [
           {
```

```
                "ParameterValue": "false",
                "ParameterName": "enable_audit_logs",
                "ApplyType": "dynamic",
                "DataType": "string",
                "Description": "Enables auditing on cluster.",
                "AllowedValues": "true,false",
                "Source": "system",
                "IsModifiable": true,
                "ApplyMethod": "pending-reboot"
            },
            {
                "ParameterValue": "disabled",
                "ParameterName": "tls",
                "ApplyType": "static",
                "DataType": "string",
                "Description": "Config to enable/disable TLS",
                "AllowedValues": "disabled,enabled",
                "Source": "system",
                "IsModifiable": true,
                "ApplyMethod": "pending-reboot"
            }
        ]
}
```

사용자 지정 클러스터 파라미터 그룹을 사용하도록 Amazon DocumentDB 클러스터 수정

Amazon DocumentDB 클러스터를 생성하면 해당 클러스터에 대해 `default.docdb4.0` 파라미터 그룹이 자동으로 생성됩니다. `default` 클러스터 파라미터 그룹은 수정할 수 없습니다. 대신 Amazon DocumentDB 클러스터를 수정하여 새로 사용자 지정된 파라미터 그룹을 연결할 수 있습니다.

이 섹션에서는 AWS Management Console 및 AWS Command Line Interface ()를 사용하여 사용자 지정 클러스터 파라미터 그룹을 사용하도록 기존 Amazon DocumentDB 클러스터를 수정하는 방법을 설명합니다AWS CLI.

Using the AWS Management Console

기본이 아닌 새로운 클러스터 파라미터 그룹을 사용하도록 Amazon DocumentDB 클러스터를 수정하려면

1. 시작하기 전에 Amazon DocumentDB 클러스터와 클러스터 파라미터 그룹을 생성했는지 확인합니다. 자세한 내용은 Amazon DocumentDB 클러스터 생성 및 Amazon DocumentDB 클러스터 파라미터 그룹 생성(을)를 참조하세요.
2. 클러스터 파라미터 그룹을 생성한 후 https://console.aws.amazon.com/docdb에서 Amazon DocumentDB 콘솔을 엽니다. 탐색 창에서 클러스터를 선택하여 새 파라미터 그룹을 클러스터에 추가합니다.
3. 파라미터 그룹을 연결할 클러스터를 선택합니다. 작업을 선택한 다음 수정을 선택하여 클러스터를 수정합니다.
4. 클러스터 옵션에서 클러스터를 연결할 새 파라미터 그룹을 선택합니다.
5. 수정 사항의 요약을 보려면 계속을 선택합니다.
6. 변경 사항을 확인한 후 즉시 적용하거나 수정 일정 아래의 다음 유지 관리 기간 중에 적용할 수 있습니다.
7. 클러스터를 새 파라미터 그룹으로 업데이트하려면 클러스터 수정을 선택합니다.

Using the AWS CLI

시작하기 전에 Amazon DocumentDB 클러스터와 클러스터 파라미터 그룹을 생성했는지 확인합니다. 작업을 사용하여 Amazon DocumentDB 클러스터를 생성할 수 있습니다. AWS CLI `create-db-cluster` `create-db-cluster-parameter-group` 작업을 사용하여 클러스터 파라미터 그룹을 생성할 수 있습니다 AWS CLI .

새 클러스터 파라미터 그룹을 클러스터에 추가하려면 다음 파라미터와 함께 `modify-db-cluster` 작업을 사용합니다 AWS CLI .

- --db-cluster-identifier - 클러스터의 이름(예: `sample-cluster`)입니다.
- --db-cluster-parameter-group-name - 클러스터를 연결할 파라미터 그룹의 이름(예: `sample-parameter-group`)입니다.

Example

```
aws docdb modify-db-cluster \
```

```
    --db-cluster-identifier sample-cluster \
    --db-cluster-parameter-group-name sample-parameter-group
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
"DBCluster": {
    "AvailabilityZones": [
            "us-west-2c",
            "us-west-2b",
            "us-west-2a"
    ],
    "BackupRetentionPeriod": 1,
    "DBClusterIdentifier": "sample-cluster",
    "DBClusterParameterGroup": "sample-parameter-group",
    "DBSubnetGroup": "default",
   ...
}
```

Amazon DocumentDB 클러스터 파라미터 그룹 복사

AWS Management Console 또는 AWS Command Line Interface ()를 사용하여 Amazon DocumentDB에서 클러스터 파라미터 그룹의 사본을 만들 수 있습니다AWS CLI.

Using the AWS Management Console

다음 절차에서는 기존 클러스터 파라미터 그룹의 복사본을 만들어 새 클러스터 파라미터 그룹을 만드는 방법을 안내합니다.

클러스터 파라미터 그룹을 복사하려면

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 파라미터 그룹을 선택합니다.
3. 클러스터 파라미터 그룹 창에서 복사하려는 클러스터 파라미터 그룹의 이름을 선택합니다.
4. 작업을 선택한 후 복사를 선택하여 해당 파라미터 그룹을 복사합니다.
5. 복사 옵션에서 새 클러스터 파라미터 그룹의 이름과 설명을 입력합니다. 그런 다음 복사를 선택하여 변경 사항을 저장합니다.

Using the AWS CLI

클러스터 파라미터 그룹을 복사하려면 다음 파라미터와 함께 `copy-db-cluster-parameter-group` 작업을 사용합니다.

- **`--source-db-cluster-parameter-group-identifier`** - 필수입니다. 복사하려는 클러스터 파라미터 그룹의 의 이름 또는 Amazon 리소스 이름(ARN)입니다.

  소스 및 대상 클러스터 파라미터 그룹이 동일한 경우 AWS 리전식별자는 이름 또는 ARN일 수 있습니다.

  소스 및 대상 클러스터 파라미터 그룹이 다른 경우 AWS 리전식별자는 ARN이어야 합니다.
- **`--target-db-cluster-parameter-group-identifier`** - 필수입니다. 클러스터 파라미터 그룹 복사본의 이름 또는 ARN입니다.

  제약 조건:
  - null이거나, 비워 두거나, 공백을 입력할 수 없습니다.
  - 1~255자의 문자, 숫자 또는 하이픈을 포함해야 합니다.
  - 첫 번째 문자는 글자이어야 합니다.
  - 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다.
- **`--target-db-cluster-parameter-group-description`** - 필수입니다. 사용자가 제공한 클러스터 파라미터 그룹 복사본에 대한 설명입니다.

Example

다음은 `sample-parameter-group`의 복사본을 만들어 이름을 `sample-parameter-group-copy`로 지정하는 코드입니다.

Linux, macOS 또는 Unix의 경우:

```
aws docdb copy-db-cluster-parameter-group \
    --source-db-cluster-parameter-group-identifier sample-parameter-group \
    --target-db-cluster-parameter-group-identifier sample-parameter-group-copy \
    --target-db-cluster-parameter-group-description "Copy of sample-parameter-group"
```

Windows의 경우:

```
aws docdb copy-db-cluster-parameter-group ^
```

```
    --source-db-cluster-parameter-group-identifier sample-parameter-group ^
    --target-db-cluster-parameter-group-identifier sample-parameter-group-copy ^
    --target-db-cluster-parameter-group-description "Copy of sample-parameter-group"
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
    "DBClusterParameterGroup": {
        "DBClusterParameterGroupArn": "arn:aws:rds:us-east-1:123456789012:cluster-
pg:sample-parameter-group-copy",
        "DBClusterParameterGroupName": "sample-parameter-group-copy",
        "DBParameterGroupFamily": "docdb4.0",
        "Description": "Copy of sample-parameter-group"
    }
}
```

Amazon DocumentDB 클러스터 파라미터 그룹 재설정

AWS Management Console 또는 AWS Command Line Interface (AWS CLI)를 사용하여 클러스터 파라미터 그룹을 재설정하여 Amazon DocumentDB 클러스터 파라미터 그룹의 파라미터 값의 일부 또는 전부를 기본값으로 재설정할 수 있습니다.

Using the AWS Management Console

클러스터 파라미터 그룹의 파라미터 값 중 일부 또는 전부를 기본값으로 재설정하려면 다음 단계를 수행하십시오.

클러스터 파라미터 그룹의 파라미터 값을 재설정하려면

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 콘솔 왼쪽의 탐색 창에서 파라미터 그룹을 선택합니다.
3. 클러스터 파라미터 그룹 창에서 재설정하려는 클러스터 파라미터 그룹의 이름을 선택합니다.
4. 작업을 선택한 후 재설정을 선택하여 해당 파라미터 그룹을 재설정합니다.
5. 결과 클러스터 파라미터 그룹 재설정 확인 페이지에서 해당 파라미터 그룹에 대한 모든 클러스터 파라미터를 기본값으로 재설정할지 확인합니다. 그런 다음 재설정을 선택하여 파라미터 그룹을 재설정합니다. 취소를 선택하여 변경 사항을 취소할 수도 있습니다.

Using the AWS CLI

클러스터 파라미터 그룹의 파라미터 값 중 일부 또는 전부를 기본값으로 재설정하려면 다음 파라미터와 함께 `reset-db-cluster-parameter-group` 작업을 사용합니다.

- **`--db-cluster-parameter-group-name`** - 필수입니다. 재설정할 클러스터 파라미터 그룹의 이름입니다.
- **`--parameters`** - 선택 사항. 기본값으로 재설정하려는 클러스터 파라미터 그룹의 `ParameterName` 및 `ApplyMethod` 목록입니다. 다음 인스턴스 재시작 또는 `reboot-db-instance` 요청 시 적용하려면 정적 파라미터를 `pending-reboot`로 설정해야 합니다. 업데이트된 정적 파라미터를 적용할 클러스터의 모든 인스턴스에 대해 `reboot-db-instance`를 호출해야 합니다.

  이 파라미터와 `--reset-all-parameters`는 상호 배타적이므로 둘 중 하나를 선택할 수 있지만 둘 다 선택할 수는 없습니다.
- **`--reset-all-parameters`** 또는 **`--no-reset-all-parameters`** - 선택 사항. 모든 파라미터(`--reset-all-parameters`)를 기본값으로 재설정할지 또는 일부 파라미터(`--no-reset-all-parameters`)를 기본값으로 재설정할지를 지정합니다. `--reset-all-parameters` 파라미터와 `--parameters`는 상호 배타적이므로 둘 중 하나를 선택할 수 있지만 둘 다 선택할 수는 없습니다.

  전체 그룹을 재설정하면 동적 파라미터가 즉시 업데이트됩니다. 다음 인스턴스 재시작 시 또는 `reboot-db-instance` 요청 시 적용하려면 정적 파라미터를 `pending-reboot`로 설정합니다. 업데이트된 정적 파라미터를 적용할 클러스터의 모든 인스턴스에 대해 `reboot-db-instance`를 호출해야 합니다.

Example

예제 1: 모든 파라미터를 기본값으로 재설정합니다.

다음 코드는 클러스터 파라미터 그룹 `sample-parameter-group`의 모든 파라미터를 기본값으로 재설정합니다.

Linux, macOS 또는 Unix의 경우:

```
aws docdb reset-db-cluster-parameter-group \
       --db-cluster-parameter-group-name sample-parameter-group \
       --reset-all-parameters
```

Windows의 경우:

```
aws docdb reset-db-cluster-parameter-group ^
       --db-cluster-parameter-group-name sample-parameter-group ^
       --reset-all-parameters
```

예제 2: 지정된 파라미터를 기본값으로 재설정합니다.

다음 코드는 클러스터 파라미터 그룹 `sample-parameter-group`의 `tls` 파라미터를 기본값으로 재설정합니다.

Linux, macOS 또는 Unix의 경우:

```
aws docdb reset-db-cluster-parameter-group \
       --db-cluster-parameter-group-name sample-parameter-group \
       --no-reset-all-parameters \
       --parameters ParameterName=tls,ApplyMethod=pending-reboot
```

Windows의 경우:

```
aws docdb reset-db-cluster-parameter-group ^
       --db-cluster-parameter-group-name sample-parameter-group ^
       --no-reset-all-parameters ^
       --parameters ParameterName=tls,ApplyMethod=pending-reboot
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
       "DBClusterParameterGroupName": "sample-parameter-group"
   }
```

클러스터 인스턴스 재부팅

정적 파라미터 값을 변경하기 전에 클러스터 인스턴스를 재부팅해야 합니다. 업데이트된 정적 파라미터를 적용할 클러스터의 각 인스턴스를 재부팅합니다.

Linux, macOS 또는 Unix의 경우:

```
aws docdb reboot-db-instance \
       --db-instance-identifier sample-cluster-instance
```

Windows의 경우:

```
aws docdb reboot-db-instance ^
       --db-instance-identifier sample-cluster-instance
```

Amazon DocumentDB 클러스터 파라미터 그룹 삭제

AWS Management Console 또는 AWS Command Line Interface ()를 사용하여 사용자 지정 Amazon DocumentDB 클러스터 파라미터 그룹을 삭제할 수 있습니다AWS CLI. `default.docdb4.0` 클러스터 파라미터 그룹은 삭제할 수 없습니다.

Using the AWS Management Console

클러스터 파라미터 그룹을 삭제하려면

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 파라미터 그룹을 선택합니다.

   > **Tip**
   > 화면 왼쪽에 탐색 창이 표시되지 않으면 페이지 왼쪽 상단 모서리에서 메뉴 아이콘 (≡) 을 선택하십시오.

3. 파라미터 그룹 창에서 삭제하려는 클러스터 파라미터 그룹 왼쪽에 있는 라디오 버튼을 선택합니다.
4. 작업을 선택한 후 삭제를 선택합니다.
5. 삭제 확인 창에서 삭제를 선택하여 클러스터 파라미터 그룹을 삭제합니다. 클러스터 파라미터 그룹을 유지하려면 취소를 선택합니다.

Using the AWS CLI

클러스터 파라미터 그룹을 삭제하려면 다음 파라미터와 함께 `delete-db-cluster-parameter-group` 작업을 사용합니다.

- **`--db-cluster-parameter-group-name`** - 필수입니다. 삭제할 클러스터 파라미터 그룹의 이름입니다. 기존 클러스터 파라미터 그룹이어야 합니다. *`default.docdb4.0`* 클러스터 파라미터 그룹은 삭제할 수 없습니다.

Example - 클러스터 파라미터 그룹 삭제

다음 예제에서는 클러스터 파라미터 그룹을 삭제하기 위한 3단계를 안내합니다.

1. 삭제하려는 클러스터 파라미터 그룹의 이름 찾기
2. 지정된 클러스터 파라미터 그룹 삭제
3. 클러스터 파라미터 그룹이 삭제되었는지 확인

1. 삭제하려는 클러스터 파라미터 그룹의 이름을 찾습니다.

다음 코드는 모든 클러스터 파라미터 그룹의 이름을 나열합니다.

Linux, macOS 또는 Unix의 경우:

```
aws docdb describe-db-cluster-parameter-groups \
       --query 'DBClusterParameterGroups[*].[DBClusterParameterGroupName]'
```

Windows의 경우:

```
aws docdb describe-db-cluster-parameter-groups ^
       --query 'DBClusterParameterGroups[*].[DBClusterParameterGroupName]'
```

이전 작업의 출력에는 다음과 유사한 클러스터 파라미터 그룹의 이름이 나열됩니다(JSON 형식).

```
[
       [
           "default.docdb4.0"
       ],
       [
           "sample-parameter-group"
       ],
       [
           "sample-parameter-group-copy"
       ]
   ]
```

2. 클러스터 파라미터 그룹을 삭제합니다.

다음 코드는 `sample-parameter-group-copy` 클러스터 파라미터 그룹을 삭제합니다.

Linux, macOS 또는 Unix의 경우:

```
aws docdb delete-db-cluster-parameter-group \
       --db-cluster-parameter-group-name sample-parameter-group-copy
```

Windows의 경우:

```
aws docdb delete-db-cluster-parameter-group ^
       --db-cluster-parameter-group-name sample-parameter-group-copy
```

이 작업에는 출력이 없습니다.

3. 지정된 클러스터 파라미터 그룹이 삭제되었는지 확인합니다.

다음 코드는 이름이 변경된 모든 클러스터 파라미터 그룹의 이름을 나열합니다.

Linux, macOS 또는 Unix의 경우:

```
aws docdb describe-db-cluster-parameter-groups \
       --query 'DBClusterParameterGroups[*].[DBClusterParameterGroupName]'
```

Windows의 경우:

```
aws docdb describe-db-cluster-parameter-groups ^
       --query 'DBClusterParameterGroups[*].[DBClusterParameterGroupName]'
```

이전 작업의 출력에는 다음과 유사한 클러스터 파라미터 그룹이 나열됩니다(JSON 형식). 방금 삭제한 클러스터 파라미터 그룹이 목록에 없어야 합니다.

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
[
       [
           "default.docdb4.0"
       ],
       [
           "sample-parameter-group"
       ]
   ]
```

Amazon DocumentDB 클러스터 파라미터 참조

동적 파라미터를 변경하고 클러스터 파라미터 그룹을 저장하면 즉시 적용 설정과 관계없이 변경 사항이 바로 적용됩니다. 고정 파라미터를 변경하고 클러스터 파라미터 그룹을 저장하면 인스턴스를 수동으로 재부팅한 후에 파라미터 변경 사항이 적용됩니다. Amazon DocumentDB 콘솔을 사용하거나 명시적으로 `reboot-db-instance`를 호출하여 인스턴스를 재부팅할 수 있습니다.

다음 표에는 Amazon DocumentDB 클러스터의 모든 인스턴스에 적용되는 파라미터가 나와 있습니다.

Amazon DocumentDB 클러스터 수준 파라미터

파라미터	기본 값	유효값	수정 가능	적용 유형	데이터 형식	설명
`audit_logs`	disabled	enabled, disabled, ddl, dml_read, dml_write, all, none	예	동적	String	Amazon CloudWatch 감사 로그 활성화 여부를 정의합니다. • **`enabled`** — CloudWatch 감사 로그가 활성화되었습니다. • **`disabled`** — CloudWatch 감사 로그가 비활성화되었습니다. • **`ddl`** — DDL 이

파라미터	기본 값	유효값	수정 가능	적용 유형	데이터 형식	설명
						벤트 감사가 활성화되었습니다. • **dml_read** - DML 읽기 이벤트에 대한 감사가 활성화됩니다. • **dml_write** — DML 쓰기 이벤트 감사가 활성화되었습니다. • **all** — 모든 데이터베이스 이벤트에 대한 감사가 활성화되었습니다. • **none** — 감사가 비활성화

파라미터	기본 값	유효값	수정 가능	적용 유형	데이터 형식	설명
						되었습니다.
change_stream_log_retention_duration	10800	3600-604800	예	동적	Integer	변경 스트림 로그가 보관되고 소비될 수 있는 시간(초)을 정의합니다.
default_collection_compression	disabled	enabled, disabled	예	동적	String	클러스터의 새 컬렉션에 대한 기본 압축 설정을 정의합니다. • **enabled** — 압축은 기본적으로 활성화되어 있습니다. • **disabled** — 압축은 기본적으로 비활성화되어 있습니다.

파라미터	기본 값	유효값	수정 가능	적용 유형	데이터 형식	설명
`profiler`	disabled	enabled, disabled	예	동적	String	느린 작업에 대해 프로파일링을 활성화합니다. • **`enabled`** — 고객이 정의한 임계값(예: 100ms)보다 오래 걸리는 작업은 Amazon CloudWatch Logs에 기록됩니다. • **`disabled`** — 느린 작업은 CloudWatch Logs에 기록되지 않습니다.

파라미터	기본 값	유효값	수정 가능	적용 유형	데이터 형식	설명
profiler_sampling_rate	1.0	0.0-1.0	예	동적	Float	로깅된 작업의 샘플링 비율을 정의합니다.
profiler_threshold_ms	100	50-2147483646	예	동적	Integer	profiler에 대한 임계값을 정의합니다. • profiler_threshold_ms 보다 큰 모든 작업은 CloudWatch Logs에 기록됩니다.

파라미터	기본 값	유효값	수정 가능	적용 유형	데이터 형식	설명
`tls`	enabled	활성화됨, 비활성화됨, fips-140-3, tls1.2+, tls1.3+	예	정적	String	TLS(전송 계층 보안) 연결이 필요한지 여부를 정의합니다. • **enabled** — 연결하려면 TLS 연결이 필요합니다. • **disabled** — 연결하는 데 TLS 연결을 사용할 수 없습니다. • **fips-140-3** — 연결하려면 FIPS(연방 정보 처리 표준) 특성을 사용한 TLS 연결이 필요합니다. FIPS 간행물

파라미터	기본 값	유효값	수정 가능	적용 유형	데이터 형식	설명
						140-3에 따라 클러스터는 보안 연결만 허용합니다. 이는 다음 지역의 Amazon DocumentDB 5.0(엔진 버전 3.0.3727) 클러스터에서만 지원됩니다: ca-central-1, us-west-2, us-east-1, us-east-2, us-gov-east-1, us-gov-west-1. • **tls1.2+** - TLS 버전 1.2 이상을 사용하는

파라미터	기본 값	유효값	수정 가능	적용 유형	데이터 형식	설명
						TLS 연결은 연결에 필요합니다. 이는 Amazon DocumentDB 4.0(엔진 버전 2.0.10980) 및 Amazon DocumentDB 5.0(엔진 버전 3.0.11051)부터만 지원됩니다. • **tls1.3+** - TLS 버전 1.3 이상을 사용하는 TLS 연결은 연결에 필요합니다. 이는 Amazon DocumentDB 4.0(엔진 버전 2.0.10980

파라미터	기본 값	유효값	수정 가능	적용 유형	데이터 형식	설명
						) 및 Amazon DocumentDB 5.0(엔진 버전 3.0.11051)부터만 지원됩니다.
`ttl_monitor`	enabled	enabled, disabled	예	동적	String	Time To Live(TTL) 모니터링이 클러스터에 대해 활성화되었는지 여부를 정의합니다. • **`enabled`** — TTL 모니터링이 활성화되었습니다. • **`disabled`** — TTL 모니터링이 비활성화되었습니다.

Amazon DocumentDB 클러스터 파라미터 수정

Amazon DocumentDB에서 클러스터 파라미터 그룹은 클러스터에서 만드는 모든 인스턴스에 적용되는 파라미터로 구성됩니다. 사용자 지정 클러스터 파라미터 그룹의 경우 언제든지 파라미터 값을 수정하거나 모든 파라미터 값을 사용자가 생성하는 파라미터 그룹에 대한 기본값으로 재설정할 수 있습니다. 이 단원에서는 Amazon DocumentDB 클러스터 파라미터 그룹을 구성하는 파라미터 및 해당 값을 확인하는 방법과 이러한 값을 변경하거나 업데이트하는 방법에 대해 설명합니다.

파라미터는 동적이거나 정적일 수 있습니다. 동적 파라미터를 변경하고 클러스터 파라미터 그룹을 저장하면 `Apply Immediately` 설정과 관계없이 변경 사항이 바로 적용됩니다. 고정 파라미터를 변경하고 클러스터 파라미터 그룹을 저장하면 인스턴스를 수동으로 재부팅한 후에만 파라미터 변경 사항이 적용됩니다.

Amazon DocumentDB 클러스터 파라미터 그룹의 파라미터 보기

AWS Management Console 또는를 사용하여 Amazon DocumentDB 클러스터의 파라미터와 해당 값을 볼 수 있습니다 AWS CLI.

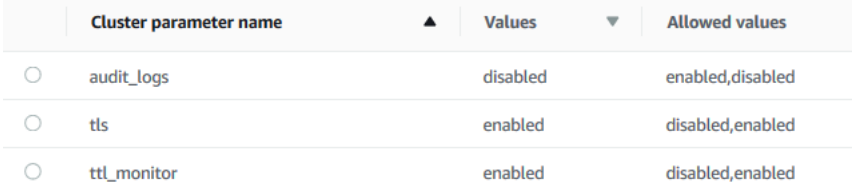
Using the AWS Management Console

클러스터 파라미터 그룹의 세부 정보를 보려면

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 파라미터 그룹을 선택합니다.

   > **Tip**
   >
   > 화면 왼쪽에 탐색 창이 표시되지 않으면 페이지 왼쪽 상단 모서리에서 메뉴 아이콘 (≡)을 선택하십시오.

3. 파라미터 그룹 창에서 세부 정보를 보려는 클러스터 파라미터 그룹의 이름을 선택합니다.
4. 결과 페이지에는 각 파라미터에 대한 값(파라미터 이름, 현재 값, 허용된 값, 파라미터 수정 가능 여부, 적용 유형, 데이터 형식 및 설명)이 표시됩니다.



Cluster parameter name	Values	Allowed values
audit_logs	disabled	enabled,disabled
tls	enabled	disabled,enabled
ttl_monitor	enabled	disabled,enabled

Using the AWS CLI

클러스터 파라미터 그룹의 파라미터 및 해당 값을 보려면 다음 파라미터와 함께 `describe-db-cluster-parameters` 작업을 사용합니다.

- **`--db-cluster-parameter-group-name`** - 필수입니다. 자세한 파라미터 목록을 원하는 클러스터 파라미터 그룹의 이름입니다.
- **`--source`** - 선택 사항. 제공된 경우 특정 소스에 대한 파라미터만 반환합니다. 파라미터 소스는 `engine-default`, `system` 또는 `user`가 될 수 있습니다.

Example

다음 코드에는 `custom3-6-param-grp` 파라미터 그룹의 모든 파라미터 및 해당 값이 나열되어 있습니다. 파라미터 그룹에 대한 세부 정보를 보려면 `--query` 줄을 생략합니다. 모든 파라미터 그룹에 대한 정보를 보려면 `--db-cluster-parameter-group-name` 줄을 생략합니다.

Linux, macOS 또는 Unix의 경우:

```
aws docdb describe-db-cluster-parameters \
   --db-cluster-parameter-group-name custom3-6-param-grp \
   --query 'Parameters[*].[ParameterName,ParameterValue]'
```

Windows의 경우:

```
aws docdb describe-db-cluster-parameters ^
   --db-cluster-parameter-group-name custom3-6-param-grp ^
   --query 'Parameters[*].[ParameterName,ParameterValue]'
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
[
    [
        "audit_logs",
        "disabled"
    ],
    [
        "tls",
        "enabled"
    ],
    [
        "ttl_monitor",
```

```
        "enabled"
    ]
]
```

Amazon DocumentDB 클러스터 파라미터 그룹의 파라미터 수정

AWS Management Console 또는를 사용하여 파라미터 그룹의 파라미터를 수정할 수 있습니다 AWS CLI.

Using the AWS Management Console

클러스터 파라미터 그룹의 파라미터를 업데이트하려면

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 파라미터 그룹을 선택합니다.

   > **Tip**
   > 화면 왼쪽에 탐색 창이 표시되지 않으면 페이지 왼쪽 상단 모서리에서 메뉴 아이콘 (≡) 을 선택하십시오.

3. 파라미터 그룹 창에서 파라미터를 업데이트하려는 클러스터 파라미터 그룹을 선택합니다.
4. 결과 페이지에는 이 클러스터 파라미터 그룹에 대한 파라미터 및 해당 세부 정보가 표시됩니다. 업데이트할 파라미터를 선택합니다.
5. 페이지 오른쪽 상단에서 편집을 선택하여 파라미터의 값을 변경합니다. 클러스터 파라미터 유형에 대한 자세한 내용은 Amazon DocumentDB 클러스터 파라미터 참조(을)를 참조하세요.
6. 변경한 다음 Modify cluster parameter(클러스터 파라미터 수정)를 선택하여 변경 사항을 저장합니다. 변경 사항을 취소하려면 취소를 선택합니다.

Using the AWS CLI

클러스터 파라미터 그룹의 파라미터를 수정하려면 다음 파라미터와 함께 `modify-db-cluster-parameter-group` 작업을 사용합니다.

- **`--db-cluster-parameter-group-name`** - 필수입니다. 수정하려는 클러스터 파라미터 그룹의 이름입니다.

- **--parameters** - 필수입니다. 수정할 파라미터입니다. 각 파라미터 요소는 다음을 포함해야 합니다.
  - **ParameterName** - 수정 중인 파라미터의 이름입니다.
  - **ParameterValue** - 이 파라미터의 새 값입니다.
  - **ApplyMethod** - 파라미터에 대한 변경 사항을 적용할 방법입니다. 허용된 값은 `immediate` 및 `pending-reboot`입니다.

> **Note**
>
> `static`의 `ApplyType` 파라미터에는 `pending-reboot`의 `ApplyMethod`이 있어야 합니다.

클러스터 파라미터 그룹의 파라미터 값을 변경하려면(AWS CLI)

다음 예에서는 `tls` 파라미터를 변경합니다.

1. **sample-parameter-group**의 파라미터 및 해당 값을 나열합니다.

   Linux, macOS 또는 Unix의 경우:

```
aws docdb describe-db-cluster-parameters \
    --db-cluster-parameter-group-name sample-parameter-group
```

   Windows의 경우:

```
aws docdb describe-db-cluster-parameters ^
    --db-cluster-parameter-group-name sample-parameter-group
```

   이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
    "Parameters": [
        {
            "Source": "system",
            "ApplyType": "static",
            "AllowedValues": "disabled,enabled",
            "ParameterValue": "enabled",
            "ApplyMethod": "pending-reboot",
```

```
            "DataType": "string",
            "ParameterName": "tls",
            "IsModifiable": true,
            "Description": "Config to enable/disable TLS"
        },
        {
            "Source": "user",
            "ApplyType": "dynamic",
            "AllowedValues": "disabled,enabled",
            "ParameterValue": "enabled",
            "ApplyMethod": "pending-reboot",
            "DataType": "string",
            "ParameterName": "ttl_monitor",
            "IsModifiable": true,
            "Description": "Enables TTL Monitoring"
        }
    ]
}
```

2. 값이 **disabled**이 되도록 **tls** 파라미터를 수정합니다. ApplyType은 static이므로 ApplyMethod를 수정할 수 없습니다.

   Linux, macOS 또는 Unix의 경우:

```
aws docdb modify-db-cluster-parameter-group \
    --db-cluster-parameter-group-name sample-parameter-group \
    --parameters
 "ParameterName"=tls,ParameterValue=disabled,ApplyMethod=pending-reboot"
```

   Windows의 경우:

```
aws docdb modify-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name sample-parameter-group ^
    --parameters "ParameterName=tls,ParameterValue=disabled,ApplyMethod=pending-
reboot"
```

   이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
    "DBClusterParameterGroupName": "sample-parameter-group"
}
```

3. 적어도 5분을 기다립니다.
4. **sample-parameter-group**의 파라미터 값을 나열합니다.

   Linux, macOS 또는 Unix의 경우:

```
aws docdb describe-db-cluster-parameters \
    --db-cluster-parameter-group-name sample-parameter-group
```

   Windows의 경우:

```
aws docdb describe-db-cluster-parameters ^
    --db-cluster-parameter-group-name sample-parameter-group
```

   이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
    "Parameters": [
        {
            "ParameterName": "audit_logs",
            "ParameterValue": "disabled",
            "Description": "Enables auditing on cluster.",
            "Source": "system",
            "ApplyType": "dynamic",
            "DataType": "string",
            "AllowedValues": "enabled,disabled",
            "IsModifiable": true,
            "ApplyMethod": "pending-reboot"
        },
        {
            "ParameterName": "tls",
            "ParameterValue": "disabled",
            "Description": "Config to enable/disable TLS",
            "Source": "user",
            "ApplyType": "static",
            "DataType": "string",
            "AllowedValues": "disabled,enabled",
            "IsModifiable": true,
            "ApplyMethod": "pending-reboot"
        }
    ]
}
```

Amazon DocumentDB 엔드포인트에 대한 이해

Amazon DocumentDB(MongoDB와 호환됨) 엔드포인트를 사용하여 클러스터 또는 인스턴스에 연결할 수 있습니다. Amazon DocumentDB에는 각각 고유한 용도가 있는 세 가지 유형의 엔드포인트가 있습니다.

주제

- 클러스터 엔드포인트 찾기
- 인스턴스의 엔드포인트 찾기
- 엔드포인트에 연결

클러스터 엔드포인트

클러스터 엔드포인트는 클러스터의 현재 기본 인스턴스에 연결되는 Amazon DocumentDB 클러스터의 엔드포인트입니다. 각 Amazon DocumentDB 클러스터 엔드포인트 하나와 기본 인스턴스 하나가 있습니다. 장애 조치의 경우 클러스터 엔드포인트가 새로운 기본 인스턴스에 다시 매핑됩니다.

리더 엔드포인트

리더 엔드포인트는 해당 클러스터에서 사용할 수 있는 복제본 중 하나에 연결되는 Amazon DocumentDB 클러스터의 엔드포인트입니다. 각 Amazon DocumentDB 클러스터에는 리더 엔드포인트가 1개씩 있습니다. 복제본이 둘 이상이면 리더 엔드포인트는 각 연결 요청을 Amazon DocumentDB 복제본 중 하나로 전달합니다.

인스턴스 엔드포인트

인스턴스 엔드포인트는 특정 인스턴스에 연결되는 엔드포인트입니다. 기본 인스턴스인지 복제본 인스턴스인지 여부에 상관없이 클러스터의 각 인스턴스에는 고유한 인스턴스 엔드포인트가 있습니다. 애플리케이션에서 인스턴스 엔드포인트를 사용하지 않는 것이 좋습니다. 왜냐하면 장애 조치를 실행하는 경우 역할을 변경할 수 있으므로 애플리케이션에서 코드 변경을 요구할 수 있기 때문입니다.

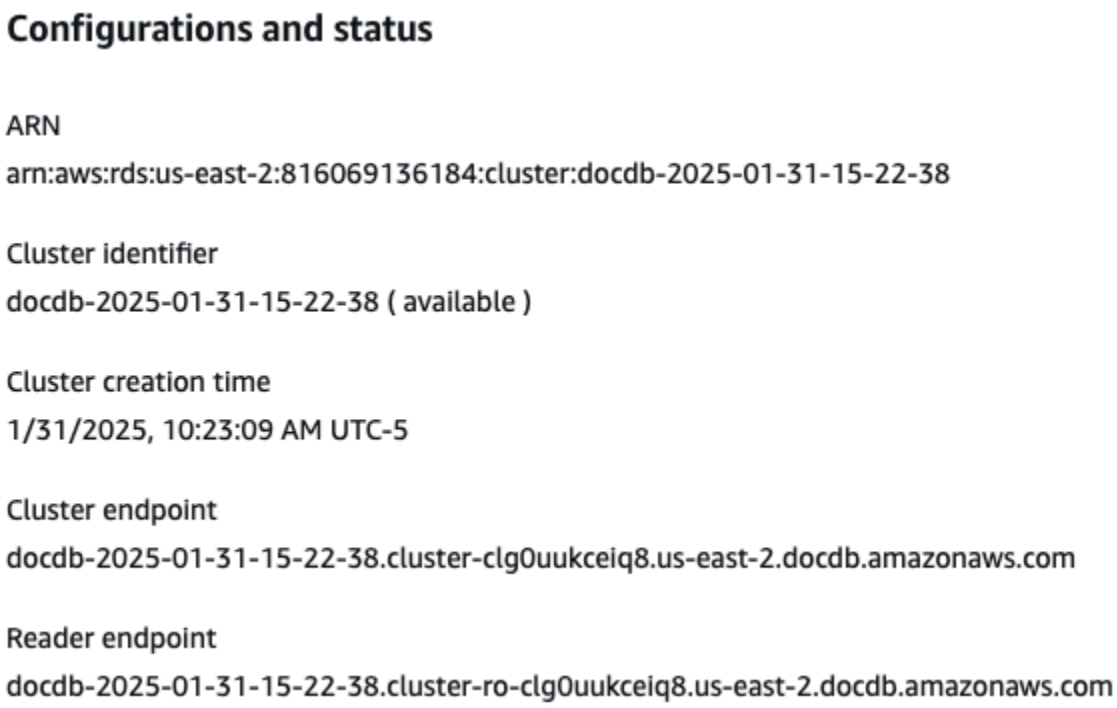
클러스터 엔드포인트 찾기

Amazon DocumentDB 콘솔 또는 AWS CLI를 사용하여 클러스터의 클러스터 엔드포인트와 리더 엔드포인트를 찾을 수 있습니다.

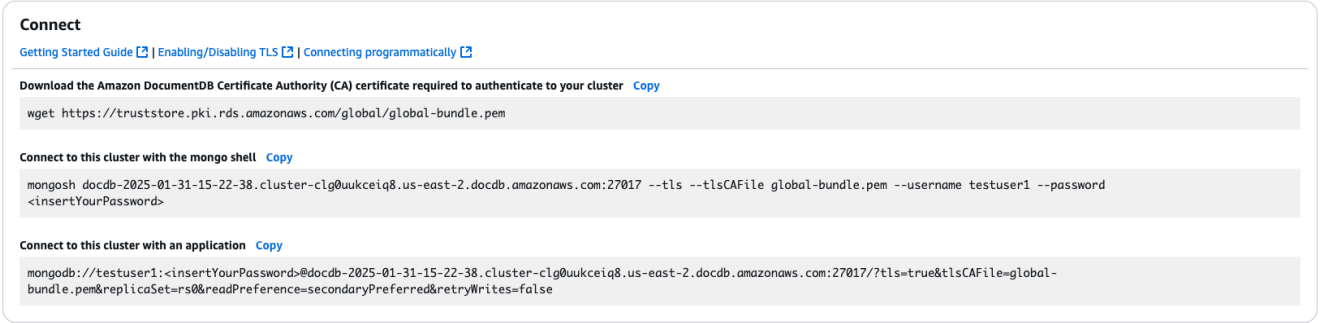
Using the AWS Management Console

콘솔을 사용하여 클러스터의 엔드포인트를 찾으려면:

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 클러스터를 선택합니다.
3. 클러스터 목록에서 원하는 클러스터 이름을 선택합니다.
4. 클러스터 세부 정보 페이지에서 구성 탭을 선택합니다. 구성 및 상태 섹션에서 클러스터 엔드포인트와 리더 엔드포인트를 찾을 수 있습니다.



5. 이 클러스터에 연결하려면 연결 및 보안 탭을 선택합니다. mongo 쉘의 연결 문자열과 애플리케이션 코드에서 클러스터에 연결하는 데 사용할 수 있는 연결 문자열을 찾습니다.



Using the AWS CLI

를 사용하여 클러스터의 클러스터 및 리더 엔드포인트를 찾으려면 이러한 파라미터로 `describe-db-clusters` 명령을 AWS CLI실행합니다.

파라미터

- **`--db-cluster-identifier`**—선택 사항. 엔드포인트를 반환하려는 클러스터를 지정합니다. 생략한 경우 최대 100개의 클러스터에 대한 엔드포인트를 반환합니다.
- **`--query`**—선택 사항. 표시할 필드를 지정합니다. 엔드포인트를 찾기 위해 볼 데이터의 양을 줄여서 도와줍니다. 생략하면 클러스터에 대한 모든 정보가 반환됩니다.
- **`--region`**—선택 사항. `--region` 파라미터를 사용하여 명령을 적용할 리전을 지정합니다. 생략하면 기본 리전이 사용됩니다.

Example

다음 예에서는 `sample-cluster`에 대한 `DBClusterIdentifier`, 엔드포인트(클러스터 엔드포인트) 및 `ReaderEndpoint`를 반환합니다.

Linux, macOS, Unix의 경우:

```
aws docdb describe-db-clusters \
   --region us-east-1 \
   --db-cluster-identifier sample-cluster \
   --query 'DBClusters[*].[DBClusterIdentifier,Port,Endpoint,ReaderEndpoint]'
```

Windows의 경우:

```
aws docdb describe-db-clusters ^
   --region us-east-1 ^
   --db-cluster-identifier sample-cluster ^
   --query 'DBClusters[*].[DBClusterIdentifier,Port,Endpoint,ReaderEndpoint]'
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
[
  [
     "sample-cluster",
```

```
    27017,
    "sample-cluster.cluster-corlsfccjozr.us-east-1.docdb.amazonaws.com",
    "sample-cluster.cluster-ro-corlsfccjozr.us-east-1.docdb.amazonaws.com"
  ]
]
```

클러스터 엔드포인트가 있으므로 이제 mongo 또는 mongodb를 사용하여 클러스터에 연결할 수 있습니다. 자세한 내용은 엔드포인트에 연결 단원을 참조하십시오.

인스턴스의 엔드포인트 찾기

Amazon DocumentDB 콘솔 또는 AWS CLI을 사용하여 인스턴스의 엔드포인트를 찾을 수 있습니다.

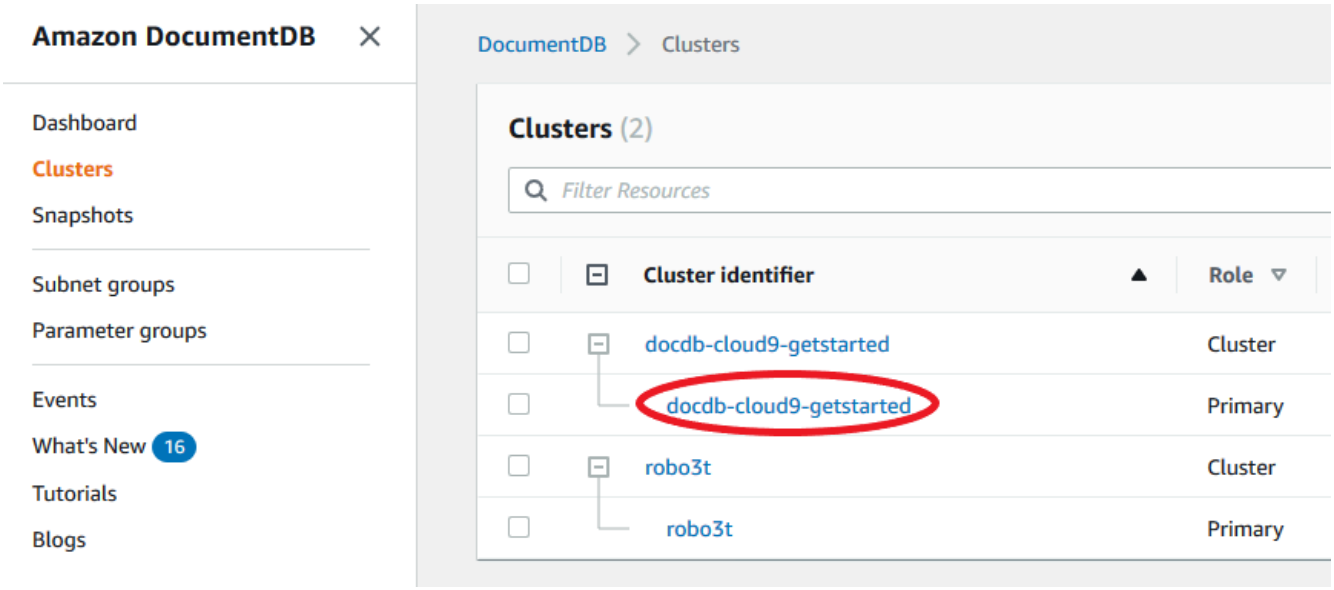
Using the AWS Management Console

콘솔을 사용하여 인스턴스의 엔드포인트를 찾으려면

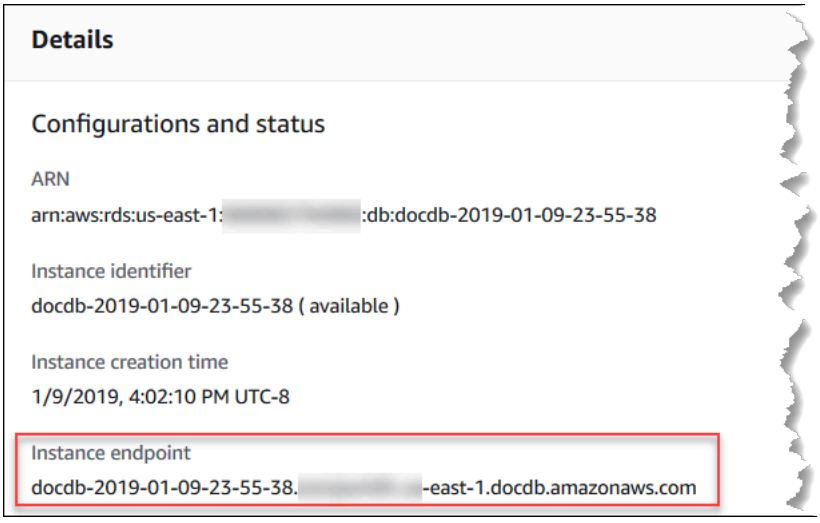
1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 클러스터를 선택합니다.

   > **Tip**
   >
   > 화면 왼쪽에 탐색 창이 표시되지 않으면 페이지 왼쪽 상단 모서리에서 메뉴 아이콘 (≡)을 선택합니다.

3. 클러스터 탐색 상자에 클러스터 식별자 열이 표시됩니다. 인스턴스는 아래 스크린샷과 마찬가지로 클러스터 아래에 나열됩니다.



4. 관심 있는 인스턴스의 왼쪽에 있는 확인란을 체크합니다.
5. 아래로 스크롤하여 세부 정보 섹션으로 이동한 다음 인스턴스 엔드포인트를 찾습니다.



6. 이 인스턴스에 연결하려면 위로 스크롤하여 연결 섹션으로 이동합니다. mongo 쉘에 대한 연결 문자열과 애플리케이션 코드에서 인스턴스에 연결하는 데 사용할 수 있는 연결 문자열을 찾습니다.



Using the AWS CLI

를 사용하여 인스턴스 엔드포인트를 찾으려면 다음 인수를 사용하여 다음 명령을 AWS CLI실행합니다.

인수

- **`--db-instance-identifier`**—선택 사항. 엔드포인트를 반환하려는 인스턴스를 지정합니다. 생략한 경우 최대 100개의 인스턴스에 대한 엔드포인트를 반환합니다.
- **`--query`**—선택 사항. 표시할 필드를 지정합니다. 엔드포인트를 찾기 위해 볼 데이터의 양을 줄여서 도와줍니다. 생략하면 인스턴스에 대한 모든 정보가 반환됩니다. `Endpoint` 필드에는 세 개의 멤버가 있습니다. 따라서 다음 예제와 같이 쿼리에 나열하면 세 멤버가 모두 반환됩니다. `Endpoint` 멤버 중 일부에만 관심이 있는 경우 두 번째 예제와 같이 쿼리의 `Endpoint`를 원하는 멤버로 교체하세요.
- **`--region`**—선택 사항. `--region` 파라미터를 사용하여 명령을 적용할 리전을 지정합니다. 생략하면 기본 리전이 사용됩니다.

Example

Linux, macOS, Unix의 경우:

```
aws docdb describe-db-instances \
    --region us-east-1 \
    --db-instance-identifier sample-cluster-instance \
    --query 'DBInstances[*].[DBInstanceIdentifier,Endpoint]'
```

Windows의 경우:

```
aws docdb describe-db-instances ^
    --region us-east-1 ^
    --db-instance-identifier sample-cluster-instance ^
    --query 'DBInstances[*].[DBInstanceIdentifier,Endpoint]'
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
[
    [
        "sample-cluster-instance",
        {
            "Port": 27017,
```

```
            "Address": "sample-cluster-instance.corcjozrlsfc.us-
east-1.docdb.amazonaws.com",
            "HostedZoneId": "Z2R2ITUGPM61AM"
        }
    ]
]
```

엔드포인트의 HostedZoneId를 제거하기 위해 출력을 줄이고 Endpoint.Port 및 Endpoint.Address를 지정하여 쿼리를 수정할 수 있습니다.

Linux, macOS, Unix의 경우:

```
aws docdb describe-db-instances \
    --region us-east-1 \
    --db-instance-identifier sample-cluster-instance \
    --query 'DBInstances[*].[DBInstanceIdentifier,Endpoint.Port,Endpoint.Address]'
```

Windows의 경우:

```
aws docdb describe-db-instances ^
    --region us-east-1 ^
    --db-instance-identifier sample-cluster-instance ^
    --query 'DBInstances[*].[DBInstanceIdentifier,Endpoint.Port,Endpoint.Address]'
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
[
    [
        "sample-cluster-instance",
        27017,
        "sample-cluster-instance.corcjozrlsfc.us-east-1.docdb.amazonaws.com"
    ]
]
```

인스턴스 엔드포인트가 있으므로 이제 mongo 또는 mongodb를 사용하여 인스턴스에 연결할 수 있습니다. 자세한 내용은 엔드포인트에 연결 단원을 참조하십시오.

엔드포인트에 연결

엔드포인트(클러스터 또는 인스턴스)가 있는 경우 mongo 쉘 또는 연결 문자열을 사용하여 엔드포인트에 연결할 수 있습니다.

mongo 쉘을 사용하여 연결

다음 구조를 사용하면 mongo 쉘을 사용하여 클러스터 또는 인스턴스에 연결하는 데 필요한 문자열을 작성할 수 있습니다.

```
mongo \
    --ssl \
    --host Endpoint:Port \
    --sslCAFile global-bundle.pem \
    --username UserName \
    --password Password
```

mongo 쉘 예

클러스터에 연결:

```
mongo \
    --ssl \
    --host sample-cluster.corcjozrlsfc.us-east-1.docdb.amazonaws.com:27017 \
    --sslCAFile global-bundle.pem \
    --username UserName \
    --password Password
```

인스턴스에 연결:

```
mongo \
    --ssl \
    --host sample-cluster-instance.corcjozrlsfc.us-east-1.docdb.amazonaws.com:27017 \
    --sslCAFile global-bundle.pem \
    --username UserName \
    --password Password
```

연결 문자열을 사용하여 연결

다음 구조를 사용하면 클러스터 또는 인스턴스에 연결하는 데 필요한 연결 문자열을 작성할 수 있습니다.

```
mongodb://UserName:Password@endpoint:port?replicaSet=rs0&ssl_ca_certs=global-
bundle.pem
```

연결 문자열 예

클러스터에 연결:

```
mongodb://UserName:Password@sample-cluster.cluster-corlsfccjozr.us-
east-1.docdb.amazonaws.com:27017?replicaSet=rs0&ssl_ca_certs=global-bundle.pem
```

인스턴스에 연결:

```
mongodb://UserName:Password@sample-cluster-instance.cluster-corlsfccjozr.us-
east-1.docdb.amazonaws.com:27017?replicaSet=rs0&ssl_ca_certs=global-bundle.pem
```

Amazon DocumentDB Amazon 리소스 이름(ARN) 이해

에서 생성하는 리소스 AWS 는 각각 Amazon 리소스 이름(ARN)으로 고유하게 식별됩니다. 특정 Amazon DocumentDB(MongoDB 호환) 작업의 경우 ARN을 지정하여 Amazon DocumentDB 리소스를 고유하게 식별해야 합니다. 예를 들어, 리소스에 태그를 추가할 경우 리소스의 ARN을 제공해야 합니다.

주제

- Amazon DocumentDB 리소스에 대한 ARN 구성
- Amazon DocumentDB 리소스 ARN 찾기

Amazon DocumentDB 리소스에 대한 ARN 구성

다음 구문을 사용하여 Amazon DocumentDB 리소스에 대한 ARN을 구성할 수 있습니다. Amazon DocumentDB는 Amazon Relational Database Service(RDS) ARNs의 형식을 공유합니다. Amazon DocumentDB ARN에는 `rds`가 포함되지만 `docdb`는 포함되지 않습니다.

`arn:aws:rds:region:account_number:resource_type:resource_id`

리전 이름	지역	가용 영역(컴퓨팅)
미국 동부(오하이오)	`us-east-2`	3
미국 동부(버지니아 북부)	`us-east-1`	6

리전 이름	지역	가용 영역(컴퓨팅)
미국 서부(오리건)	us-west-2	4
아프리카(케이프타운)	af-south-1	3
남아메리카(상파울루)	sa-east-1	3
아시아 태평양(홍콩)	ap-east-1	3
아시아 태평양(하이데라바드)	ap-south-2	3
아시아 태평양(뭄바이)	ap-south-1	3
아시아 태평양(서울)	ap-northeast-2	4
아시아 태평양(싱가포르)	ap-southeast-1	3
아시아 태평양(시드니)	ap-southeast-2	3
아시아 태평양(도쿄)	ap-northeast-1	3
캐나다(중부)	ca-central-1	3
중국(베이징) 리전	cn-north-1	3
중국(닝샤)	cn-northwest-1	3
유럽(프랑크푸르트)	eu-central-1	3
유럽(아일랜드)	eu-west-1	3
유럽(런던)	eu-west-2	3
유럽(밀라노)	eu-south-1	3
유럽(파리)	eu-west-3	3
유럽(스페인)	eu-south-2	3

리전 이름	지역	가용 영역(컴퓨팅)
유럽(스톡홀름)	`eu-north-1`	3
중동(UAE)	`me-central-1`	3
이스라엘(텔아비브)	`il-central-1`	3
AWS GovCloud(미국 서부)	`us-gov-west-1`	3
AWS GovCloud(미국 동부)	`us-gov-east-1`	3

> **Note**
>
> Amazon DocumentDB 아키텍처는 스토리지와 컴퓨팅을 분리합니다. 스토리지 계층의 경우 Amazon DocumentDB는 3개의 AWS 가용 영역(AZs. 위의 표에 나열된 AZ는 컴퓨팅 인스턴스 프로비저닝을 위해 특정 리전에서 사용 가능한 AZ의 수입니다. 예를 들어 ap-northeast-1에서 Amazon DocumentDB 클러스터를 시작하는 경우 스토리지는 3개의 AZ에 걸쳐 6가지 방식으로 복제되지만 컴퓨팅 인스턴스는 2개의 AZ에서만 사용할 수 있습니다.

다음 표는 특정 Amazon DocumentDB 리소스에 대한 ARN을 구성할 때 사용해야 하는 형식을 보여줍니다. Amazon DocumentDB는 Amazon RDS ARNs의 형식을 공유합니다. Amazon DocumentDB ARN에는 `rds`가 포함되지만 `docdb`는 포함되지 않습니다.

리소스 유형	ARN 형식/예
인스턴스(db)	`arn:aws:rds:` *`region`*`:`*`account_number`* `:db:`*`resource_id`* `arn:aws:rds:us-east-1:` *`1234567890`* `:db:`*`sample-db-instance`*
클러스터(cluster)	`arn:aws:rds:` *`region`*`:`*`account_number`* `:cluster:` *`resource_id`*

리소스 유형	ARN 형식/예
	`arn:aws:rds:us-east-1:` *1234567890* `:cluster:` *sample-db-cluster*
클러스터 파라미터 그룹 (cluster-pg)	`arn:aws:rds:` *region*`:`*account_number* `:cluster-pg:` *resource_id* `arn:aws:rds:us-east-1:` *1234567890* `:cluster-pg:` *sample-db-cluster-parameter-group*
보안 그룹(secgrp)	`arn:aws:rds:` *region*`:`*account_number* `:secgrp:`*resource_id* `arn:aws:rds:us-east-1:` *1234567890* `:secgrp:`*sample-public-secgrp*
클러스터 스냅샷(cluster-snapshot)	`arn:aws:rds:` *region*`:`*account_number* `:cluster-snapshot:` *resource_id* `arn:aws:rds:us-east-1:` *1234567890* `:cluster-snapshot:` *sample-db-cluster-snapshot*
서브넷 그룹(subgrp)	`arn:aws:rds:` *region*`:`*account_number* `:subgrp:`*resource_id* `arn:aws:rds:us-east-1:` *1234567890* `:subgrp:`*sample-subnet-10*

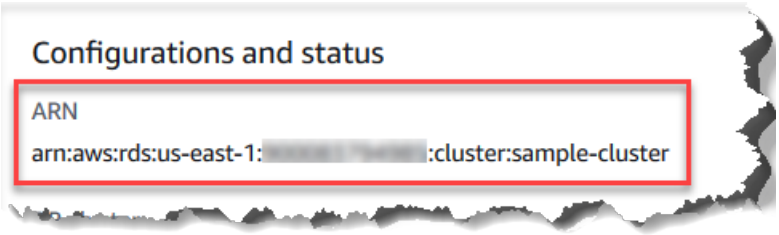
Amazon DocumentDB 리소스 ARN 찾기

AWS Management Console 또는를 사용하여 Amazon DocumentDB 리소스의 ARN을 찾을 수 있습니다 AWS CLI.

Using the AWS Management Console

콘솔을 사용해 ARN을 찾으려면 ARN을 보려는 리소스를 탐색하여 해당 리소스의 세부 정보를 확인합니다.

예를 들어 클러스터 세부 정보 페이지에서 구성 탭을 선택하여 클러스터의 ARN을 가져올 수 있습니다. 다음 스크린샷과 같이 구성 및 상태 섹션에서 ARN을 찾을 수 있습니다.



Using the AWS CLI

특정 Amazon DocumentDB 리소스에 AWS CLI 대해를 사용하여 ARN을 가져오려면 해당 리소스에 대한 `describe` 작업을 사용합니다. 다음 표에는 각 AWS CLI 작업과 작업과 함께 ARN을 가져오는 데 사용되는 ARN 속성이 나와 있습니다.

AWS CLI 명령	ARN 속성
`describe-db-instances`	`DBInstanceArn`
`describe-db-clusters`	`DBClusterArn`
`describe-db-parameter-groups`	`DBParameterGroupArn`
`describe-db-cluster-parameter-groups`	`DBClusterParameterGroupArn`
`describe-db-security-groups`	`DBSecurityGroupArn`
`describe-db-snapshots`	`DBSnapshotArn`
`describe-db-cluster-snapshots`	`DBClusterSnapshotArn`
`describe-db-subnet-groups`	`DBSubnetGroupArn`

Example - 클러스터의 ARN 찾기

다음 AWS CLI 작업은 클러스터의 ARN을 찾습니다`sample-cluster`.

Linux, macOS, Unix의 경우:

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].DBClusterArn'
```

Windows의 경우:

```
aws docdb describe-db-clusters ^
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].DBClusterArn'
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
[
    "arn:aws:rds:us-east-1:123456789012:cluster:sample-cluster"
]
```

Example - 여러 파라미터 그룹의 ARN 찾기

Linux, macOS, Unix의 경우:

```
aws docdb describe-db-cluster-parameter-groups \
   --query 'DBClusterParameterGroups[*].DBClusterParameterGroupArn'
```

Windows의 경우:

```
aws docdb describe-db-cluster-parameter-groups ^
   --query 'DBClusterParameterGroups[*].DBClusterParameterGroupArn'
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
[
    "arn:aws:rds:us-east-1:123456789012:cluster-pg:custom3-6-param-grp",
    "arn:aws:rds:us-east-1:123456789012:cluster-pg:default.aurora5.6",
    "arn:aws:rds:us-east-1:123456789012:cluster-pg:default.docdb3.6"
```

```
]
```

Amazon DocumentDB 리소스 태그 지정

Amazon DocumentDB(MongoDB 호환) 태그를 사용하여 Amazon DocumentDB 리소스에 메타데이터를 추가할 수 있습니다. 이러한 태그는 AWS Identity and Access Management (IAM) 정책과 함께 사용하여 Amazon DocumentDB 리소스에 대한 액세스를 관리하고 리소스에 적용할 수 있는 작업을 제어할 수 있습니다. 또한 비슷하게 태그가 지정된 리소스에 대한 비용을 그룹화하여 이러한 태그로 비용을 추적할 수 있습니다.

다음 Amazon DocumentDB 리소스에 태그를 지정할 수 있습니다.

- 클러스터
- 인스턴스
- 스냅샷
- 클러스터 스냅샷
- 파라미터 그룹
- 클러스터 파라미터 그룹
- 보안 그룹
- 서브넷 그룹

Amazon DocumentDB 리소스 태그 개요

Amazon DocumentDB 태그는 Amazon DocumentDB 리소스를 정의하고 연결하는 이름-값 페어입니다. 이 이름을 키라고 합니다. 키 값을 제공하는 것은 선택 사항입니다. 태그를 사용하여 임의의 정보를 Amazon DocumentDB 리소스에 할당할 수 있습니다. 범주 정의 등에 태그 키를 사용할 수 있으며 태그 값은 해당 범주의 항목일 수 있습니다. 예를 들어, 태그 키를 `project`로 정의하고 태그 값을 `Salix`로 정의할 수 있는데, 이는 Amazon DocumentDB 리소스가 Salix 프로젝트에 할당되었음을 나타냅니다. 태그를 사용하여 `environment=test` 또는 `environment=production`와 같은 키를 사용해 Amazon DocumentDB 리소스를 테스트나 생산에 사용되도록 지정할 수도 있습니다. 일관된 태그 키 세트를 사용하여 Amazon DocumentDB 리소스와 연관된 메타데이터를 보다 쉽게 추적하는 것이 좋습니다.

태그를 사용하여 자체 비용 구조를 반영하도록 AWS 청구서를 구성할 수 있습니다. 이렇게 하려면 가입하여 태그 키 값이 포함된 AWS 계정 청구서를 가져옵니다. 그런 다음 같은 태그 키 값을 가진 리소스

에 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 청구 및 비용 관리 사용 설명서의 비용 할당 태그 사용을 참조하십시오.

각 Amazon DocumentDB 리소스에는 해당 리소스에 할당된 모든 태그가 포함된 태그 세트가 있습니다. 태그 세트는 최대 10개의 태그를 포함하거나 비어 있을 수 있습니다. 리소스의 기존 태그와 동일한 키를 가진 Amazon DocumentDB 리소스에 태그를 추가하면 새 값이 이전 값을 덮어씁니다.

AWS 는 태그에 의미 체계적 의미를 적용하지 않습니다. 태그는 엄격하게 문자열로 해석됩니다. Amazon DocumentDB는 리소스를 생성할 때 사용하는 설정에 따라 인스턴스 또는 다른 Amazon DocumentDB 리소스에 태그를 설정할 수 있습니다. 예를 들어, Amazon DocumentDB는 인스턴스가 생산용 또는 테스트용임을 나타내는 태그를 추가할 수 있습니다.

스냅샷에 태그를 추가할 수 있지만 이 그룹화는 청구서에 반영되지 않습니다.

AWS Management Console 또는를 사용하여 Amazon DocumentDB 리소스에 태그를 AWS CLI 추가, 나열 및 삭제할 수 있습니다. 를 사용할 때는 작업할 리소스의 Amazon 리소스 이름(ARN)을 제공해야 AWS CLI합니다. Amazon DocumentDB ARN에 대한 자세한 내용은 Amazon DocumentDB Amazon 리소스 이름(ARN) 이해 섹션을 참조하십시오.

태그 제약 조건

Amazon DocumentDB 태그에는 다음 제약 조건이 적용됩니다.

- 리소스당 최대 태그 수 - 10개
- 최대 키 길이 - 유니코드 128자
- 최대 값 길이 - 유니코드 256자
- 키 및 값에 유효한 문자 - UTF-8 문자 세트의 대문자와 소문자, 숫자, 공백 및 `_ . : / = + -`와 `@` 문자(Java regex: `"^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$"`)
- 태그 키와 값은 대소문자를 구분합니다.
- 태그 키 또는 값에 `aws:` 접두사는 사용할 수 없습니다. 이 접두사는 AWS전용입니다.

Amazon DocumentDB 리소스의 태그 추가 및 업데이트

AWS Management Console 또는를 사용하여 리소스에 최대 10개의 태그를 추가할 수 있습니다 AWS CLI.

Using the AWS Management Console

리소스에 태그를 추가하는 프로세스는 태그를 추가하는 리소스에 관계없이 비슷합니다. 이 예에서는 클러스터에 태그를 추가합니다.

콘솔을 사용하여 클러스터에 태그를 추가하거나 업데이트하려면 다음을 수행합니다.

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 클러스터를 선택합니다.
3. 태그를 추가할 클러스터 이름을 선택합니다.
4. 태그 섹션으로 스크롤을 내린 다음 편집을 선택합니다.
5. 이 리소스에 추가할 각 태그에 대해 다음을 수행합니다.
   a. 새 태그를 추가하려면 키 상자의 태그 이름에 입력합니다. 태그 값을 변경하려면 키 열에서 태그 이름을 찾습니다.
   b. 태그에 새 값 또는 업데이트된 값을 제공하려면 값 상자에 태그에 대한 값을 입력합니다.
   c. 태그를 더 추가하려는 경우 추가를 선택합니다. 추가하지 않고 완료하려면 저장을 선택합니다.

Using the AWS CLI

리소스에 태그를 추가하는 프로세스는 태그를 추가하는 리소스에 관계없이 비슷합니다. 이 예에서는 클러스터에 세 개의 태그를 추가합니다. 두 번째 태그인 `key2`에는 값이 없습니다.

이러한 파라미터와 `add-tags-to-resource` 함께 AWS CLI 작업을 사용합니다.

파라미터

- **`--resource-name`** - 태그를 추가할 Amazon DocumentDB 리소스의 ARN입니다.
- **`--tags`** - 이 리소스에 추가할 태그(키-값 페어)를 `Key=`*`key-name`*`,Value=`*`tag-value`* 형식으로 나열합니다.

Example

Linux, macOS, Unix의 경우:

```
aws docdb add-tags-to-resource \
```

```
    --resource-name arn:aws:rds:us-east-1:1234567890:cluster:sample-cluster \
    --tags Key=key1,Value=value1 Key=key2 Key=key3,Value=value3
```

Windows의 경우:

```
aws docdb add-tags-to-resource ^
    --resource-name arn:aws:rds:us-east-1:1234567890:cluster:sample-cluster \
    --tags Key=key1,Value=value1 Key=key2 Key=key3,Value=value3
```

`add-tags-to-resource` 작업은 출력을 생성하지 않습니다. 작업 결과를 보려면 `list-tags-for-resource` 작업을 사용합니다.

Amazon DocumentDB 리소스에 태그 나열

AWS Management Console 또는를 사용하여 Amazon DocumentDB 리소스의 태그 목록을 AWS CLI 가져올 수 있습니다.

Using the AWS Management Console

리소스에서 태그를 나열하는 프로세스는 태그를 추가하는 리소스에 관계없이 비슷합니다. 이 예에서는 클러스터에 태그를 나열합니다.

콘솔을 사용하여 클러스터에서 태그를 나열하려면

1. https://console.aws.amazon.com/docdb에서 Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 클러스터를 선택합니다.
3. 태그를 나열할 클러스터 이름을 선택합니다.
4. 이 리소스의 태그 목록을 보려면 아래로 스크롤하여 태그 섹션으로 이동합니다.

Using the AWS CLI

리소스에서 태그를 나열하는 프로세스는 태그를 나열하는 리소스에 관계없이 비슷합니다. 이 예에서는 클러스터에 태그를 나열합니다.

이러한 파라미터와 `list-tags-for-resource` 함께 AWS CLI 작업을 사용합니다.

파라미터

- **`--resource-name`** - 필수입니다. 태그를 나열할 Amazon DocumentDB 리소스의 ARN입니다.

Example

Linux, macOS, Unix의 경우:

```
aws docdb list-tags-for-resource \
    --resource-name arn:aws:rds:us-east-1:1234567890:cluster:sample-cluster
```

Windows의 경우:

```
aws docdb list-tags-for-resource ^
    --resource-name arn:aws:rds:us-east-1:1234567890:cluster:sample-cluster
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
    "TagList": [
        {
            "Key": "key1",
            "Value": "value1"
        },
        {
            "Key": "key2",
            "Value": ""
        },
        {
            "Key": "key3",
            "Value": "value3"
        }
    ]
}
```

Amazon DocumentDB 리소스에서 태그 제거

AWS Management Console 또는를 사용하여 Amazon DocumentDB 리소스에서 태그를 AWS CLI 제거할 수 있습니다.

Using the AWS Management Console

리소스에서 태그를 제거하는 프로세스는 태그를 추가하는 리소스에 관계없이 비슷합니다. 이 예에서는 클러스터에서 태그를 제거합니다.

콘솔을 사용하여 클러스터에서 태그를 제거하려면

1. https://console.aws.amazon.com/docdb에서 Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 클러스터를 선택합니다.
3. 태그를 제거할 클러스터 이름을 선택합니다.
4. 태그 섹션으로 스크롤을 내린 다음 편집을 선택합니다.
5. 이 리소스에서 모든 태그를 제거하려면 모두 제거를 선택합니다. 그렇지 않으면, 이 리소스에서 제거할 각 태그에 대해 다음을 수행합니다.
   a. 키 열에서 태그 이름을 찾습니다.
   b. 태그 키와 동일한 행에서 제거를 선택합니다.
   c. 완료하였으면 저장을 선택합니다.

Using the AWS CLI

리소스에서 태그를 제거하는 프로세스는 태그를 제거하는 리소스에 관계없이 비슷합니다. 이 예에서는 클러스터에서 태그를 제거합니다.

이러한 파라미터와 `remove-tags-from-resource` 함께 AWS CLI 작업을 사용합니다.

- **`--resource-name`** - 필수입니다. 태그를 제거할 Amazon DocumentDB 리소스의 ARN입니다.
- **`--tag-keys`** - 필수입니다. 이 리소스에서 제거할 태그 키의 목록입니다.

Example

Linux, macOS, Unix의 경우:

```
aws docdb remove-tags-from-resource \
    --resource-name arn:aws:rds:us-east-1:1234567890:cluster:sample-cluster \
    --tag-keys key1 key3
```

Windows의 경우:

```
aws docdb remove-tags-from-resource ^
    --resource-name arn:aws:rds:us-east-1:1234567890:cluster:sample-cluster \
    --tag-keys key1 key3
```

`removed-tags-from-resource` 작업은 출력을 생성하지 않습니다. 작업 결과를 보려면 `list-tags-for-resource` 작업을 사용합니다.

Amazon DocumentDB 유지 관리

Amazon DocumentDB는 Amazon DocumentDB 리소스를 정기적으로 유지 관리합니다. 이러한 유지 관리에는 대개 데이터베이스 엔진(클러스터 유지 관리) 또는 인스턴스의 기본 운영 체제(OS)(인스턴스 유지 관리)에 대한 업데이트가 포함됩니다. 데이터베이스 엔진 업데이트는 필수 패치이며 데이터베이스 엔진에 대한 보안 수정, 버그 수정 및 개선 사항을 포함합니다. 대부분의 운영 체제 패치는 선택 사항이지만 한동안 패치를 적용하지 않으면 보안 태세를 유지하기 위해 패치가 필요할 수 있고 자동으로 적용될 수 있습니다. 따라서 운영 체제 업데이트는 사용 가능한 즉시 Amazon DocumentDB 인스턴스에 적용하는 것이 좋습니다.

데이터베이스 엔진 패치를 사용하려면 Amazon DocumentDB 클러스터를 잠시 오프라인으로 전환해야 합니다. 사용 가능하게 되면 Amazon DocumentDB 클러스터의 예정된 유지 관리 기간 동안 이러한 패치가 자동으로 적용되도록 예약됩니다.

클러스터 및 인스턴스 유지 관리 모두 해당 유지 관리 기간이 있습니다. 즉시 적용되지 않도록 선택한 클러스터 및 인스턴스 수정 사항은 유지 관리 기간에도 또한 적용됩니다. 기본적으로 클러스터를 생성할 때 Amazon DocumentDB가 클러스터 및 각 개별 인스턴스에 대해 유지 관리 기간을 지정합니다. 클러스터 또는 인스턴스를 생성할 때 유지 관리 기간을 선택할 수 있습니다. 또한 언제라도 비즈니스 일정 또는 절차에 맞춰 유지 관리 기간을 수정할 수 있습니다. 일반적으로 애플리케이션에 대 유지 관리의 영향을 최소화하는 유지 관리 기간을 서택하는 것이 좋습니다(예: 야간 또는 주말).

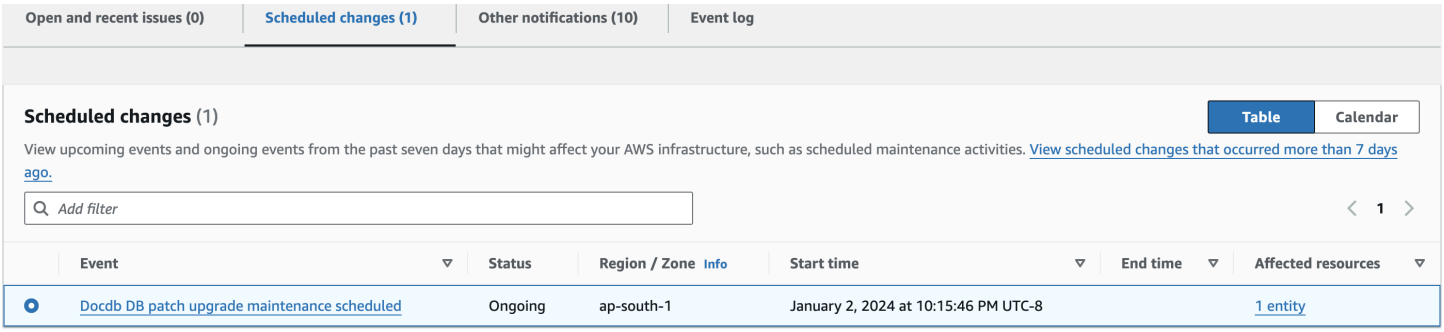
주제

- Amazon DocumentDB 엔진 패치에 대한 알림
- 보류 중인 Amazon DocumentDB 유지 관리 작업 보기
- Amazon DocumentDB 엔진 업데이트
- 사용자 시작 업데이트
- Amazon DocumentDB 유지 관리 기간 관리
- Amazon DocumentDB 운영 체제 업데이트

Amazon DocumentDB 엔진 패치에 대한 알림

AWS 콘솔의 AWS Health Dashboard (AHD) 및 이메일을 통해 상태 이벤트를 통해 필요한 데이터베이스 엔진 패치에 대한 유지 관리 알림을 받게 됩니다. 특정 AWS 리전에서 Amazon DocumentDB 엔

진 유지 관리 패치를 사용할 수 있게 되면 해당 리전의 영향을 받는 모든 Amazon DocumentDB 사용자 계정에 패치의 영향을 받는 각 Amazon DocumentDB 버전에 대한 AHD 및 이메일 알림이 전송됩니다. AWS 콘솔에서 AHD의 예약된 변경 사항 섹션에서 이러한 알림을 볼 수 있습니다. 알림에는 패치 가용성 타이밍, 자동 적용 일정, 영향을 받는 클러스터 목록 및 릴리스 정보에 대한 세부 정보가 포함됩니다. 이 알림은 이메일을 통해 AWS 계정의 루트 사용자 이메일 주소로도 전송됩니다.



이 알림을 받으면 예약된 자동 적용 날짜 이전에 이러한 엔진 패치를 Amazon DocumentDB 클러스터에 자체 적용하도록 선택할 수 있습니다. 또는 예정된 유지 관리 기간 동안 엔진 패치가 자동으로 적용될 때까지 기다릴 수 있습니다(기본 옵션).

> **Note**
>
> 새 엔진 패치 버전이 있는 새 Amazon DocumentDB 엔진 패치가 릴리스될 때까지 AHD의 알림 상태가 '진행 중'으로 설정됩니다.
> Amazon DocumentDB 클러스터에 엔진 패치가 적용되면 클러스터의 엔진 패치 버전이 업데이트되어 알림의 버전을 반영합니다. `db.runCommand({getEngineVersion: 1})` 명령을 실행하여 이 업데이트를 확인할 수 있습니다.

AWS Health 또한는 이벤트를 사용하여 확장 가능한 이벤트 기반 애플리케이션을 빌드하는 AWS Lambda Amazon EventBridge와 통합되며 Amazon Simple Queue Service(SQS) 등을 포함한 20개 이상의 대상과 통합됩니다. 엔진 패치를 사용할 수 있게 되기 전에 `AWS_DOCDB_DB_PATCH_UPGRADE_MAINTENANCE_SCHEDULED` 이벤트 코드를 사용하여 Amazon EventBridge를 설정할 수 있습니다. 이벤트 정보에 응답하고 이벤트 정보 캡처, 추가 이벤트 시작,에 대한 푸시 알림과 같은 추가 채널을 통한 알림 전송 AWS Console Mobile Application, 리전에서 Amazon DocumentDB 엔진 패치를 사용할 수 있게 될 때 수정 또는 기타 작업 수행과 같은 작업을 자동으로 수행하도록 EventBridge를 설정할 수 있습니다.

Amazon DocumentDB가 엔진 패치를 취소하는 드문 시나리오에서는 AHD 알림과 취소에 대해 알리는 이메일을 받게 됩니다. 따라서 `AWS_DOCDB_DB_PATCH_UPGRADE_MAINTENANCE_CANCELLED` 이

벤트 코드를 사용하여 이 이벤트에 응답하도록 Amazon EventBridge를 설정할 수 있습니다. Amazon EventBridge 규칙 사용에 대해 자세히 알아보려면 Amazon EventBridge 사용 설명서를 참조하세요.

보류 중인 Amazon DocumentDB 유지 관리 작업 보기

AWS Management Console 또는를 사용하여 클러스터에 유지 관리 업데이트를 사용할 수 있는지 확인할 수 있습니다 AWS CLI.

업데이트가 있을 경우에는 다음 중 한 가지를 선택할 수 있습니다.

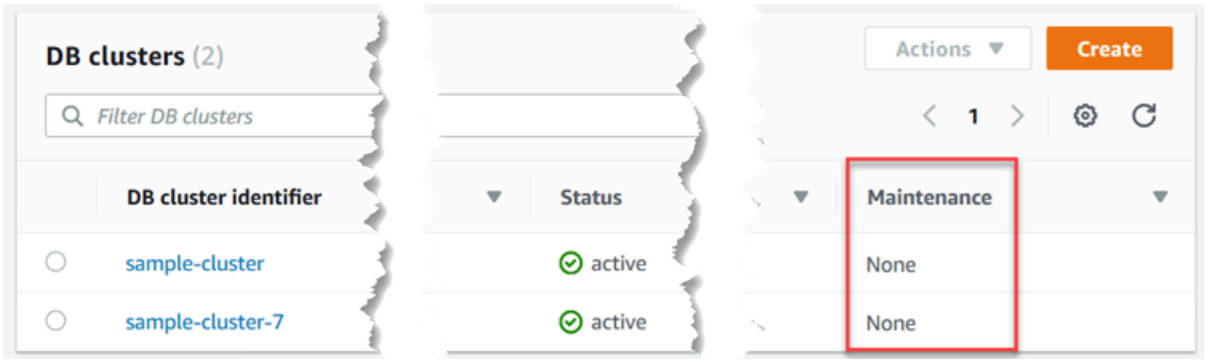
- 현재 다음 유지 관리 기간에 예약된 유지 관리 작업을 연기합니다(OS 패치만 해당).
- 유지 관리 작업을 즉시 적용합니다.
- 다음 유지 관리 기간 중 시작할 유지 관리 작업을 예약합니다.

> **Note**
>
> 조치를 취하지 않으면 엔진 패치와 같은 필수 유지 관리 조치가 예정된 유지 관리 기간에 자동으로 적용됩니다.

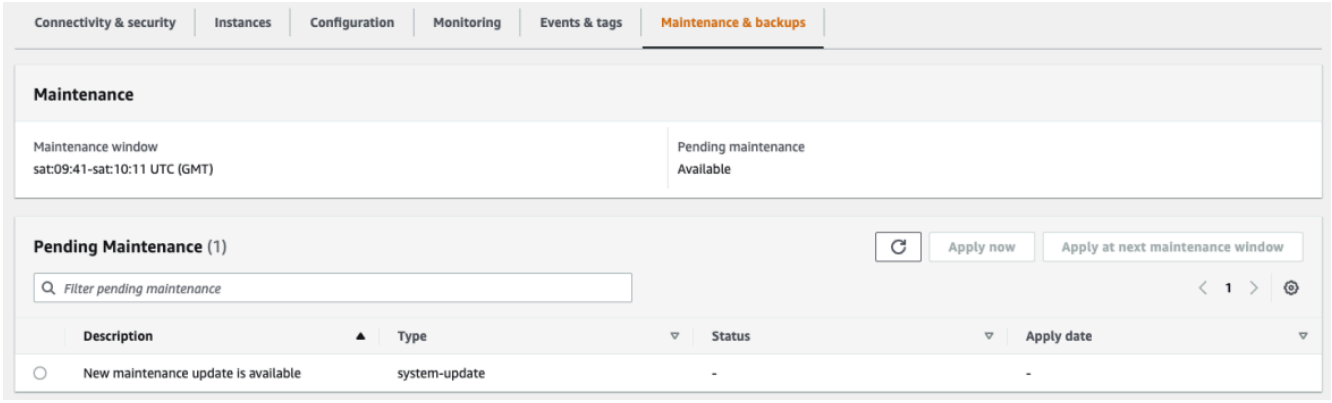
유지 관리 기간에 따라 대기 중인 작업의 시작 시기가 결정되지만 이러한 작업의 전체 실행 시간이 제한되지는 않습니다.

Using the AWS Management Console

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 클러스터를 선택합니다.
3. 업데이트가 있는 경우 다음과 같이 Amazon DocumentDB 콘솔의 클러스터에 대한 유지 관리 열에 이용 가능, 필수 또는 다음 창의 단어로 표시됩니다.



4. 조치를 취하려면 클러스터를 선택하여 세부 정보를 표시한 후 유지 관리 및 백업을 선택하십시오. 그러면 대기 중인 유지 관리 항목이 표시됩니다.



Using the AWS CLI

다음 AWS CLI 작업을 사용하여 보류 중인 유지 관리 작업을 확인합니다. 출력에는 대기 중인 유지 관리 작업이 없습니다.

```
aws docdb describe-pending-maintenance-actions
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
    "PendingMaintenanceActions": []
}
```

Amazon DocumentDB 엔진 업데이트

Amazon DocumentDB를 사용하여 유지 관리 작업을 적용하는 시기를 선택할 수 있습니다. AWS Management Console 또는를 사용하여 Amazon DocumentDB가 업데이트를 적용하는 시기를 결정할 수 있습니다 AWS CLI.

이번 주제에서 설명하는 절차에 따라 클러스터를 즉시 업그레이드하거나, 업드레이드 일정을 예약합니다.

Using the AWS Management Console

콘솔을 사용하여 Amazon DocumentDB 클러스터에 대한 업데이트를 관리할 수 있습니다.

클러스터에 대한 업데이트를 관리하려면

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 클러스터를 선택합니다.
3. 클러스터 목록에서 유지 관리 작업을 적용할 클러스터의 이름 옆에 있는 버튼을 선택합니다.
4. 작업 메뉴에서 다음 중 하나를 선택합니다.
   - 지금 업그레이드: 대기 중인 유지 관리 작업을 즉시 수행합니다.
   - 다음에 업그레이드: 클러스터의 다음 유지 관리 기간 중에 대기 중인 유지 관리 작업을 수행합니다.

   또는 클러스터 유지 관리 및 백업 탭의 보류 중인 유지 관리 섹션에서 지금 적용 또는 다음 유지 관리 기간에 적용을 클릭할 수 있습니다(이전 섹션의 사용 AWS Management Console 참조).

   > **Note**
   > 대기 중인 유지 관리 작업이 없는 경우 위의 두 옵션이 모두 비활성화됩니다.

Using the AWS CLI

보류 중인 업데이트를 클러스터에 적용하려면 `apply-pending-maintenance-action` AWS CLI 작업을 사용합니다.

파라미터

- **`--resource-identifier`** - 대기 중인 유지 관리 작업이 적용되는 Amazon DocumentDB Amazon 리소스 이름(ARN)입니다.
- **`--apply-action`** - 이 리소스에 적용할 대기 중인 유지 관리 작업입니다.

  유효한 값: `system-update` 및 `db-upgrade`
- **`--opt-in-type`** - 옵트인 요청의 유형을 지정하거나 옵트인 요청을 실행 취소하는 값입니다. `immediate` 유형의 옵트인 요청은 실행 취소할 수 없습니다.

  유효한 값:
  - `immediate` - 유지 관리 작업을 즉시 적용합니다.

- `next-maintenance` - 리소스에 대한 다음 유지 관리 기간 중에 유지 관리 작업을 적용합니다.
- `undo-opt-in` - 기존 `next-maintenance` 옵트인 요청을 취소합니다.

Example

Linux, macOS, Unix의 경우:

```
aws docdb apply-pending-maintenance-action \
    --resource-identifier arn:aws:rds:us-east-1:123456789012:db:docdb \
    --apply-action system-update \
    --opt-in-type immediate
```

Windows의 경우:

```
aws docdb apply-pending-maintenance-action ^
    --resource-identifier arn:aws:rds:us-east-1:123456789012:db:docdb ^
    --apply-action system-update ^
    --opt-in-type immediate
```

보류 중인 업데이트가 하나 이상 있는 리소스 목록을 반환하려면 `describe-pending-maintenance-actions` AWS CLI 명령을 사용합니다.

Example

Linux, macOS, Unix의 경우:

```
aws docdb describe-pending-maintenance-actions \
    --resource-identifier arn:aws:rds:us-east-1:001234567890:db:docdb
```

Windows의 경우:

```
aws docdb describe-pending-maintenance-actions ^
    --resource-identifier arn:aws:rds:us-east-1:001234567890:db:docdb
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
```

```
    "PendingMaintenanceActions": [
        {
            "ResourceIdentifier": "arn:aws:rds:us-
east-1:001234567890:cluster:sample-cluster",
            "PendingMaintenanceActionDetails": [
                {
                    "Action": "system-update",
                    "CurrentApplyDate": "2019-01-11T03:01:00Z",
                    "Description": "db-version-upgrade",
                    "ForcedApplyDate": "2019-01-18T03:01:00Z",
                    "AutoAppliedAfterDate": "2019-01-11T03:01:00Z"
                }
            ]
        }
    ]
}
```

`describe-pending-maintenance-actions` AWS CLI 작업의 `--filters` 파라미터를 지정하여 클러스터의 리소스 목록을 반환할 수도 있습니다. `--filters` 작업의 형식은 `Name=`*`filter-name`*`,Values=`*`resource-id`*`,...`입니다.

필터의 `Name` 파라미터에 대해 허용되는 값은 `db-cluster-id`입니다. 이 값은 클러스터 식별자 또는 ARN 목록을 허용합니다. 반환되는 목록에는 이러한 식별자 또는 ARN으로 식별된 클러스터에 대해 보류 중인 유지 관리 작업만 포함됩니다.

다음 예에서는 `sample-cluster1` 및 `sample-cluster2` 클러스터에 대해 보류 중인 유지 관리 작업을 반환합니다.

Example

Linux, macOS, Unix의 경우:

```
aws docdb describe-pending-maintenance-actions \
   --filters Name=db-cluster-id,Values=sample-cluster1,sample-cluster2
```

Windows의 경우:

```
aws docdb describe-pending-maintenance-actions ^
   --filters Name=db-cluster-id,Values=sample-cluster1,sample-cluster2
```

적용 날짜

각 유지 관리 작업에는 대기 중인 유지 관리 작업을 설명할 때 찾을 수 있는 적용 날짜가 있습니다. 에서 보류 중인 유지 관리 작업의 출력을 읽으면 AWS CLI세 가지 날짜가 나열됩니다. 이러한 날짜 값은 유지 관리가 선택 사항인 null 경우입니다. 해당 유지 관리 작업이 예약되거나 적용되면 값이 채워집니다.

- **CurrentApplyDate** - 유지 관리 작업이 즉시 또는 다음 유지 관리 기간에 적용되는 날짜입니다.
- **ForcedApplyDate** - 유지 관리가 사용자의 유지 관리 기간과 상관없이 자동으로 적용되는 날짜입니다.
- **AutoAppliedAfterDate** - 유지 관리가 클러스터의 유지 관리 기간 동안 이 날짜 이후에 적용됩니다.

사용자 시작 업데이트

Amazon DocumentDB 사용자가 클러스터 또는 인스턴스에 대한 업데이트를 시작할 수 있습니다. 예를 들어 사용자가 인스턴스의 클래스를 메모리 용량이 다른 클래스로 수정하거나 클러스터의 파라미터 그룹을 수정할 수 있습니다. Amazon DocumentDB는 이러한 변경 사항을 Amazon DocumentDB에서 시작한 업데이트와 다르게 봅니다. 클러스터 또는 인스턴스 수정에 대한 자세한 정보는 다음 단원 중 하나를 참조하십시오.

- Amazon DocumentDB 클러스터 수정
- Amazon DocumentDB 인스턴스 수정

대기 중인 사용자 시작 변경 사항의 목록을 보려면 다음 명령을 실행합니다.

Example

인스턴스에서 대기 중인 사용자 시작 변경 사항을 보려면

Linux, macOS, Unix의 경우:

```
aws docdb describe-db-instances \
    --query 'DBInstances[*].
[DBClusterIdentifier,DBInstanceIdentifier,PendingModifiedValues]'
```

Windows의 경우:

```
aws docdb describe-db-instances ^
```

```
    --query 'DBInstances[*].
[DBClusterIdentifier,DBInstanceIdentifier,PendingModifiedValues]'
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

이 경우, `sample-cluster-instance`는 `db.r5.xlarge` 인스턴스 클래스에 대기 중인 변경 사항이 있고, `sample-cluster-instance-2`는 대기 중인 변경 사항이 없습니다.

```
[
    [
        "sample-cluster",
        "sample-cluster-instance",
        {
            "DBInstanceClass": "db.r5.xlarge"
        }
    ],
    [
        "sample-cluster",
        "sample-cluster-instance-2",
        {}
    ]
]
```

Amazon DocumentDB 유지 관리 기간 관리

각 인스턴스 및 클러스터에는 대기 중인 변경 사항이 적용되는 주 단위 유지 관리 기간이 있습니다. 유지 관리 기간은 요청 또는 필요에 따라 수정하거나 소프트웨어 패치를 적용하는 시기를 조정할 수 있는 기간입니다. 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.

리전별로 8시간 블록 시간 중에서 30분 유지 관리 기간이 임의로 선택됩니다. 인스턴스 또는 클러스터를 생성할 때 기본 유지 관리 기간을 지정하지 않으면 Amazon DocumentDB에서 임의로 선택한 요일에 30분 유지 관리 기간을 배정합니다.

다음 표는 기본 유지 관리 기간을 할당하는 각 리전의 시간 블록 목록입니다.

리전 이름	지역	UTC 시간 블록
미국 동부(오하이오)	us-east-2	03:00~11:00

리전 이름	지역	UTC 시간 블록
미국 동부(버지니아 북부)	us-east-1	03:00~11:00
미국 서부(오리건)	us-west-2	06:00~14:00
아프리카(케이프타운)	af-south-1	03:00~11:00
아시아 태평양(홍콩)	ap-east-1	06:00~14:00
아시아 태평양(하이데라바드)	ap-south-2	06:30-14:30
아시아 태평양(뭄바이)	ap-south-1	06:00~14:00
아시아 태평양(서울)	ap-northeast-2	13:00~21:00
아시아 태평양(싱가포르)	ap-southeast-1	14:00–22:00
아시아 태평양(시드니)	ap-southeast-2	12:00–20:00
아시아 태평양(도쿄)	ap-northeast-1	13:00~21:00
캐나다(중부)	ca-central-1	03:00~11:00
중국(베이징)	cn-north-1	06:00~14:00
중국(닝샤)	cn-northwest-1	06:00~14:00
유럽(프랑크푸르트)	eu-central-1	21:00~05:00
유럽(아일랜드)	eu-west-1	22:00~06:00
유럽(런던)	eu-west-2	22:00~06:00
유럽(밀라노)	eu-south-1	02:00~10:00
유럽(파리)	eu-west-3	23:59~07:29
유럽(스페인)	eu-south-2	02:00~10:00
유럽(스톡홀름)	eu-north-1	04:00~12:00

리전 이름	지역	UTC 시간 블록
중동(UAE)	me-central-1	05:00~13:00
남아메리카(상파울루)	sa-east-1	00:00~08:00
이스라엘(텔아비브)	il-central-1	04:00-12:00
AWS GovCloud(미국 동부)	us-gov-east-1	17:00–01:00
AWS GovCloud(미국 서부)	us-gov-west-1	06:00~14:00

Amazon DocumentDB 유지 관리 윈도우 변경

유지 관리 기간은 사용률이 가장 낮은 시간에 할당되어야 하므로 수시로 변경되어야 할 수 있습니다. 시스템 변경 사항(스토리지 조정 작업 또는 인스턴스 클래스 변경 등)을 적용 중이고 가동 중단이 필요한 경우에만 인 이 기간 동안 클러스터 또는 인스턴스를 사용할 수 없습니다. 그런 다음 필수 변경 사항을 적용하는 데 필요한 최소 시간 동안만 사용이 불가능합니다.

데이터베이스 엔진 업그레이드를 위해 Amazon DocumentDB는 개별 인스턴스가 아닌 클러스터에 대한 기본 유지 관리 기간을 사용합니다.

유지 관리 기간 변경하기

- 클러스터: Amazon DocumentDB 클러스터 수정 단원을 참조하십시오.
- 인스턴스: Amazon DocumentDB 인스턴스 수정 단원을 참조하십시오.

Amazon DocumentDB 운영 체제 업데이트

Amazon DocumentDB 클러스터의 인스턴스에는 때때로 운영 체제 업데이트가 필요합니다. Amazon DocumentDB는 운영 체제를 최신 버전으로 업그레이드하여 데이터베이스 성능과 고객의 전반적인 보안 태세를 개선합니다. 운영 체제 업데이트는 Amazon DocumentDB 인스턴스의 클러스터 엔진 버전이나 인스턴스 클래스를 변경하지 않습니다.

클러스터의 가용성을 최대화하기 위해서 클러스터의 리더 인스턴스를 먼저 업데이트한 다음 라이터 인스턴스를 업데이트하는 것이 좋습니다. 장애 조치 시 다운타임이 발생할 수 있으므로 리더 인스턴스와 라이터 인스턴스를 동시에 업데이트하지 않는 것이 좋습니다.

Amazon DocumentDB에 대한 대부분의 운영 체제 업데이트는 선택 사항이며 이를 적용할 설정된 날짜가 없습니다. 그러나 한동안 이러한 업데이트를 적용하지 않으면 결국 인스턴스의 유지 관리 기간 동안 업데이트가 필요해지고 자동으로 적용될 수 있습니다. 이는 데이터베이스의 보안 태세를 유지하는 데 도움이 됩니다. 예기치 않은 가동 중지 시간을 방지하려면 Amazon DocumentDB 인스턴스를 사용할 수 있게 되는 즉시 운영 체제 업데이트를 적용하고 비즈니스 요구 사항에 따라 편한 시간에 인스턴스 유지 관리 기간을 설정하는 것이 좋습니다.

새로운 선택적 업데이트가 제공될 때 알림을 받으려면 보안 패치 이벤트 범주에서 RDS-EVENT-0230 구독을 신청하면 됩니다. Amazon DocumentDB 이벤트 구독에 대한 자세한 내용은 Amazon DocumentDB 이벤트 구독을 참조하십시오.

클러스터나 인스턴스 유지 관리 수행 시에는 해당 인스턴스가 기본 인스턴스일 경우 장애 조치를 실시합니다. 가용성을 높이려면 Amazon DocumentDB 클러스터에 두 개 이상의 인스턴스를 사용하는 것이 좋습니다. 자세한 내용은 Amazon DocumentDB 장애 조치 단원을 참조하십시오.

> **Note**
>
> 일부 관리 기능의 경우 Amazon DocumentDB는 Amazon Relational Database Service(RDS)와 공유되는 운영 기술을 사용합니다.

> **Important**
>
> Amazon DocumentDB 인스턴스는 운영 체제 업그레이드 중에 오프라인 상태가 됩니다. 다중 인스턴스 클러스터를 사용하여 클러스터 가동 중지 시간을 최소화할 수 있습니다. 다중 인스턴스 클러스터가 없는 경우 이 유지 관리를 수행하기 위해 보조 인스턴스를 추가한 다음 유지 관리가 완료되면 추가 리더 인스턴스를 삭제하여 임시로 생성하도록 선택할 수 있습니다(보조 인스턴스에 대한 정규 요금이 적용됩니다).

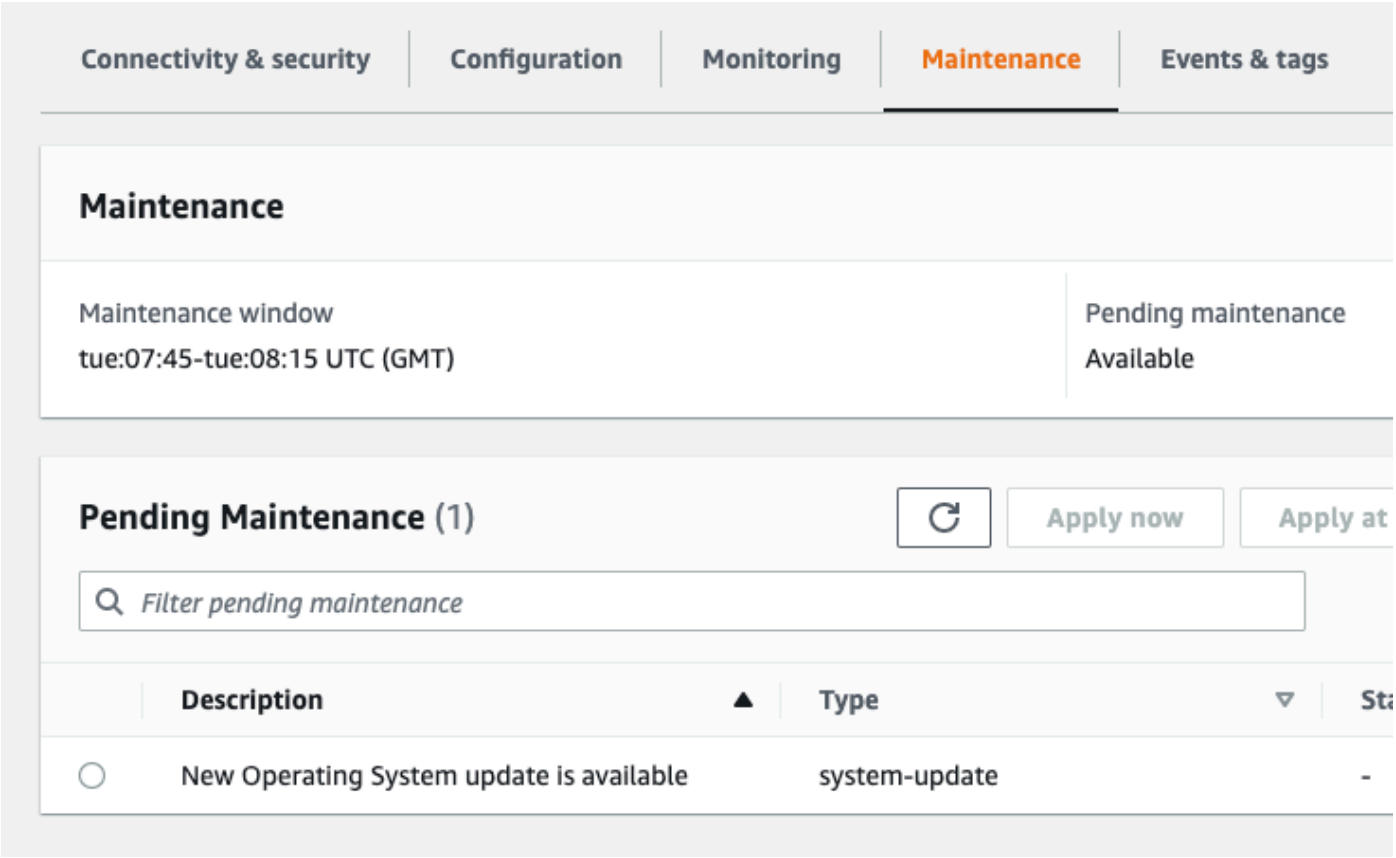
> **Note**
>
> 여러 규정 준수 의무를 충족하려면 모든 선택 및 필수 업데이트를 적용하여 최신 상태를 유지해야 할 수 있습니다. 유지 관리 기간 동안 Amazon DocumentDB에서 제공하는 모든 업데이트를 정기적으로 적용하는 것이 좋습니다.

AWS Management Console 또는를 사용하여 업데이트를 사용할 수 있는지 AWS CLI 확인할 수 있습니다.

Using the AWS Management Console

AWS Management Console을 사용하여 업데이트를 사용할 수 있는지 확인하려면

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 클러스터를 선택한 후 해당 인스턴스를 선택합니다.
3. 유지 관리를 선택합니다.
4. 대기 중인 유지 관리 섹션에서 운영 체제 업데이트를 찾습니다.



운영 체제 업데이트를 선택하고 대기 중 유지 관리 섹션에서 지금 적용 또는 다음 유지 관리 창에 적용을 클릭할 수 있습니다. 유지 관리 값이 다음 창인 경우 작업에서 Defer 업그레이드를 선택하여 유지 관리 항목을 보류하십시오. 유지 관리 작업이 이미 시작된 경우에는 보류할 수 없습니다.

또는 탐색 창에서 클러스터를 클릭하고 작업 메뉴에서 지금 적용 또는 다음 유지 관리 기간에 적용을 선택하여 클러스터 목록에서 인스턴스를 선택할 수 있습니다.

Using the AWS CLI

를 사용하여 업데이트를 사용할 수 있는지 확인하려면 `describe-pending-maintenance-actions` 명령을 AWS CLI호출합니다.

```
aws docdb describe-pending-maintenance-actions
```

```
{
  "ResourceIdentifier": "arn:aws:docdb:us-east-1:123456789012:db:mydb2",
  "PendingMaintenanceActionDetails": [
    {
      "Action": "system-update",
      "Description": "New Operating System update is available"
    }
  ]
}
```

운영 체제 업데이트는 Amazon DocumentDB 엔진 버전 및 인스턴스 클래스에 따라 다릅니다. 따라서 Amazon DocumentDB 인스턴스는 서로 다른 시점에 업데이트를 받거나 이를 요구합니다. 엔진 버전 및 인스턴스 클래스에 따라 인스턴스에 운영 체제 업데이트가 지원되는 경우 업데이트가 콘솔에 표시됩니다. 명령을 실행 AWS CLI `describe-pending-maintenance-actions`하거나 `DescribePendingMaintenanceActions` API 작업을 호출하여 볼 수도 있습니다.

Amazon DocumentDB 엔진의 최신 클러스터 패치 릴리스를 실행하지 않는 경우 운영 체제 업데이트가 사용 가능한 유지 관리로 나열되지 않을 수 있습니다. 운영 체제 업데이트를 보고 관리하려면 먼저 최신 엔진 패치 버전으로 업그레이드해야 합니다.

서비스 연결 역할 이해

Amazon DocumentDB(MongoDB 호환)는 AWS Identity and Access Management (IAM) 서비스 연결 역할을 사용합니다. 서비스 연결 역할은 Amazon DocumentDB에 직접 연결된 고유한 유형의 IAM 역할입니다. 서비스 연결 역할은 Amazon DocumentDB에서 사전 정의하며 서비스가 사용자를 대신하여 다른 AWS 서비스를 호출하는 데 필요한 모든 권한을 포함합니다.

필요한 권한을 수동으로 추가할 필요가 없으므로 서비스 연결 역할은 Amazon DocumentDB를 더 쉽게 사용할 수 있습니다. Amazon DocumentDB에서 서비스 연결 역할의 권한을 정의하므로 다르게 정의

되지 않은 한, Amazon DocumentDB만 해당 역할을 수임할 수 있습니다. 정의된 권한에는 신뢰 정책과 권한 정책이 포함되며 이 권한 정책은 다른 IAM 엔티티에 연결할 수 없습니다.

먼저 역할의 관련 리소스를 삭제해야만 역할을 삭제할 수 있습니다. 이렇게 하면 리소스에 대한 액세스 권한을 실수로 삭제할 수 없기 때문에 Amazon DocumentDB 리소스가 보호됩니다.

서비스 연결 역할을 지원하는 기타 서비스에 대한 자세한 내용은 IAM으로 작업하는AWS 서비스를 참조하고 서비스 연결 역할 열에 예가 있는 서비스를 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 링크가 있는 예를 선택합니다.

Amazon DocumentDB 서비스 연결 역할 권한

Amazon DocumentDB(MongoDB 호환)는 AWSServiceRoleForRDS라는 서비스 연결 역할을 사용하여 Amazon DocumentDB가 클러스터를 대신하여 AWS 서비스를 호출하도록 허용합니다.

AWSServiceRoleForRDS 서비스 연결 역할은 역할을 수임하기 위해 다음 서비스를 신뢰합니다.

- `docdb.amazonaws.com`

역할 권한 정책은 Amazon DocumentDB가 지정된 리소스에서 다음 작업을 완료하도록 허용합니다.

- `ec2`에 대한 작업:
  - `AssignPrivateIpAddresses`
  - `AuthorizeSecurityGroupIngress`
  - `CreateNetworkInterface`
  - `CreateSecurityGroup`
  - `DeleteNetworkInterface`
  - `DeleteSecurityGroup`
  - `DescribeAvailabilityZones`
  - `DescribeInternetGateways`
  - `DescribeSecurityGroups`
  - `DescribeSubnets`
  - `DescribeVpcAttribute`
  - `DescribeVpcs`
  - `ModifyNetworkInterfaceAttribute`
  - `RevokeSecurityGroupIngress`

  - UnassignPrivateIpAddresses
- sns에 대한 작업:
  - ListTopic
  - Publish
- cloudwatch에 대한 작업:
  - PutMetricData
  - GetMetricData
  - CreateLogStream
  - PullLogEvents
  - DescribeLogStreams
  - CreateLogGroup

> **Note**
>
> IAM 엔터티(사용자, 그룹, 역할 등)가 서비스 링크 역할을 생성하고 편집하거나 삭제할 수 있도록 권한을 구성할 수 있습니다. 다음과 같은 오류 메시지가 표시될 수 있습니다.
> 리소스를 만들 수 없습니다. 서비스 연결 역할을 생성할 권한이 있는지 확인하십시오. 그렇지 않은 경우 기다렸다가 나중에 다시 시도하십시오.
> 이러한 오류가 표시되면 다음 권한이 활성화되어 있는지 확인합니다.
>
> ```
> {
>     "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 권한 섹션을 참조하세요.

Amazon DocumentDB 서비스 연결 역할 생성

서비스 링크 역할은 수동으로 생성할 필요가 없습니다. 클러스터를 생성할 때 Amazon DocumentDB에서는 서비스 연결 역할을 생성합니다.

이 서비스 연결 역할을 삭제한 다음 다시 생성해야 하는 경우 동일한 프로세스를 사용하여 계정에서 역할을 다시 생성할 수 있습니다. 클러스터를 생성할 때 Amazon DocumentDB에서는 서비스 연결 역할을 다시 생성합니다.

Amazon DocumentDB 서비스 연결 역할 수정

Amazon DocumentDB에서는 AWSServiceRoleForRDS 서비스 연결 역할을 수정할 수 없습니다. 서비스 링크 역할을 생성한 후에는 다양한 개체가 역할을 참조할 수 있기 때문에 역할 이름을 변경할 수 없습니다. 그러나 IAM을 사용하여 역할의 설명을 수정할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 편집을 참조하세요.

Amazon DocumentDB 서비스 연결 역할 삭제

서비스 연결 역할이 필요한 기능 또는 서비스가 더 이상 필요 없는 경우에는 해당 역할을 삭제하는 것이 좋습니다. 따라서 적극적으로 모니터링하거나 유지하지 않는 미사용 엔터티가 없도록 합니다. 그러나 서비스 연결 역할을 삭제하려면 먼저 모든 클러스터를 삭제해야 합니다.

Amazon DocumentDB 서비스 연결 역할 정리

IAM을 사용하여 서비스 연결 역할을 삭제하기 전에 먼저 역할에 활성 세션이 없는지 확인하고 역할에서 사용되는 리소스를 모두 제거해야 합니다.

콘솔을 사용하여 서비스 연결 역할에 활성 세션이 있는지 확인하려면 다음을 수행합니다.

1. 에 로그인 AWS Management Console 하고에서 IAM 콘솔을 엽니다https://console.aws.amazon.com/iam/.
2. IAM 콘솔의 탐색 창에서 역할을 선택한 다음 AWSServiceRoleForRDS 역할의 이름(확인란 아님)을 선택합니다.
3. 선택한 역할의 요약 페이지에서 Access Advisor 탭을 선택합니다.
4. Access Advisor 탭에서 서비스 연결 역할의 최근 활동을 검토합니다.

> **Note**
>
> Amazon DocumentDB에서 AWSServiceRoleForRDS 역할을 사용하는지 잘 모를 경우에는 역할을 삭제할 수 있습니다. 서비스에서 역할을 사용하는 경우에는 삭제가 안 되어 역할이 사용 중인 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 인스턴스 및 클러스터를 삭제해야 합니다. 인스턴스 및 클러스터 삭제에 대한 자세한 내용은 다음 주제를 참조하십시오.

- Amazon DocumentDB 인스턴스 삭제
- Amazon DocumentDB 클러스터 삭제

Amazon DocumentDB 서비스 연결 역할에 대해 지원되는 리전

Amazon DocumentDB는 서비스가 제공되는 모든 리전에서 서비스 연결 역할 사용을 지원합니다. 자세한 내용은 https://docs.aws.amazon.com/documentdb/latest/developerguide/regions-and-azs.html#regions-and-azs-availability 단원을 참조하십시오.

Amazon DocumentDB에서 변경 스트림 사용

Amazon DocumentDB(MongoDB 호환)의 변경 스트림 기능은 클러스터의 컬렉션 내에서 시간순으로 발생하는 변경 이벤트 시퀀스를 제공합니다. 변경 스트림에서 이벤트를 읽어 다음을 비롯한 다양한 사용 사례를 구현할 수 있습니다.

- 변경 알림
- Amazon OpenSearch Service(OpenSearch Service)로 전체 텍스트 검색
- Amazon Redshift로 분석

애플리케이션에서는 변경 스트림을 사용하여 개별 컬렉션의 데이터 변경 사항을 구독할 수 있습니다. 변경 스트림 이벤트는 클러스터에서 발생하는 순서대로 정렬되며 이벤트가 기록된 후 3시간(기본값) 동안 저장됩니다. 파라미터를 사용하여 보존 기간을 최대 7일까지 연장할 수 있습니다.

change_stream_log_retention_duration 변경 스트림 보존 기간을 수정하려면 변경 스트림 로그 보존 기간 수정을 참조하세요.

주제

- 지원되는 연산자
- 결제
- 제한 사항
- 변경 스트림 활성화
- 예: Python에서 변경 스트림 사용
- 전체 문서 조회
- 변경 스트림 재개
- startAtOperationTime에서 변경 스트림 재개
- postBatchResumeToken에서 변경 스트림 재개
- 변경 스트림 내 트랜잭션
- 변경 스트림 로그 보존 기간 수정
- 보조 인스턴스에서 변경 스트림 사용

지원되는 연산자

Amazon DocumentDB는 변경 스트림에 대해 다음 작업을 지원합니다.

- MongoDB db.collection.watch(), db.watch() 및 client.watch()API에서 지원되는 모든 변경 이벤트.
- 업데이트를 위한 전체 문서 조회.
- 집계 단계: $match$project,$redact, 및 $addFields와 $replaceRoot.
- 이력서 토큰에서 변경 스트림 재개
- startAtOperation 항목을 사용하여 타임스탬프에서 변경 스트림 재개(Amazon DocumentDB 4.0+에 해당)

결제

Amazon DocumentDB 변경 스트림 기능은 기본적으로 비활성화되어 있으며 이 기능이 활성화될 때까지 추가 요금이 발생하지 않습니다. 클러스터에서 변경 스트림을 사용하면 추가 읽기 및 쓰기 IO와 스

토리지 비용이 발생합니다. `modifyChangeStreams` API 작업을 사용하여 클러스터에 대해 이 기능을 활성화할 수 있습니다. 요금에 대한 자세한 내용은 Amazon DocumentDB 요금을 참조하세요.

제한 사항

Amazon DocumentDB에서 변경 스트림에는 다음과 같은 제한 사항이 있습니다.

- Amazon DocumentDB 3.6. 및 Amazon DocumentDB 4.0에서 변경 스트림은 Amazon DocumentDB 클러스터의 기본 인스턴스에 대한 연결에서만 열 수 있습니다. Amazon DocumentDB 3.6. 및 Amazon DocumentDB 4.0에서는 복제본 인스턴스의 변경 스트림에서 읽기 기능이 지원되지 않습니다. `watch()` API 작업을 호출할 때 모든 읽기가 기본 인스턴스에 대해 수행되도록 `primary` 읽기 기본 설정을 지정해야 합니다(예제 섹션 참조).
- Amazon DocumentDB 5.0에서는 전역 클러스터를 포함하여 기본 인스턴스와 보조 인스턴스 모두에서 변경 스트림을 열 수 있습니다. 보조 읽기 기본 설정을 지정하여 변경 스트림을 보조 인스턴스로 리디렉션할 수 있습니다. 추가 모범 사례 및 제한 사항은 보조 인스턴스에서 변경 스트림 사용 항목을 참조하세요.
- 모음의 변경 스트림에 작성된 이벤트는 최대 7일 동안 사용할 수 있습니다(기본값은 3시간). 변경 스트림 데이터는 새 변경 사항이 발생하지 않은 경우에도 로그 보존 기간 후에 삭제됩니다.
- 모음에서 `updateMany` 또는 `deleteMany` 같은 장기 실행 쓰기 작업을 수행하는 경우, 장기 실행 쓰기 작업이 완료될 때까지 변경 스트림 이벤트의 쓰기를 일시적으로 중단할 수 있습니다.
- Amazon DocumentDB는 MongoDB 작업 로그(`oplog`)를 지원하지 않습니다.
- Amazon DocumentDB를 사용하면 지정된 컬렉션에서 변경 스트림을 명시적으로 활성화해야 합니다.
- 변경 스트림 이벤트의 총 크기(변경 데이터 및 요청된 경우 전체 문서 포함)가 16 `MB`보다 크면 클라이언트가 변경 스트림에서 읽기에 실패합니다.
- Amazon DocumentDB v3.6과 `client.watch()` 및 `db.watch()` 항목을 사용할 때는 현재 Ruby 드라이버가 지원되지 않습니다.
- 필드의 업데이트된 값이 이전 값과 동일한 경우 변경 스트림의 `updateDescription` 명령 출력은 Amazon DocumentDB와 MongoDB에서 다릅니다.
  - 제공된 필드가 `$set` 명령에 지정되어 있고 대상 값이 이미 소스 값과 동일한 경우 Amazon DocumentDB는 `updateDescription` 출력의 필드를 반환하지 않습니다.
  - MongoDB는 지정된 값이 현재 값과 같더라도 출력의 필드를 반환합니다.

변경 스트림 활성화

해당 데이터베이스 내의 모든 모음 또는 선택한 모음에 대해 Amazon DocumentDB 변경 스트림을 활성화할 수 있습니다. 다음은 mongo 쉘을 사용하여 다른 사용 사례에 변경 스트림을 활성화하는 방법의 예입니다. 데이터베이스 및 모음 이름을 지정할 때 빈 문자열은 와일드카드로 취급됩니다.

```
//Enable change streams for the collection "foo" in database "bar"
db.adminCommand({modifyChangeStreams: 1,
    database: "bar",
    collection: "foo",
    enable: true});
```

```
//Disable change streams on collection "foo" in database "bar"
db.adminCommand({modifyChangeStreams: 1,
    database: "bar",
    collection: "foo",
    enable: false});
```

```
//Enable change streams for all collections in database "bar"
db.adminCommand({modifyChangeStreams: 1,
    database: "bar",
    collection: "",
    enable: true});
```

```
//Enable change streams for all collections in all databases in a cluster
db.adminCommand({modifyChangeStreams: 1,
    database: "",
    collection: "",
    enable: true});
```

다음과 같은 경우 컬렉션에 대해 변경 스트림이 활성화됩니다.

- 데이터베이스와 컬렉션 모두 명시적으로 활성화되어 있습니다.
- 컬렉션을 포함하는 데이터베이스가 활성화되어 있습니다.
- 모든 데이터베이스가 활성화되어 있습니다.

상위 데이터베이스에도 변경 스트림이 활성화되어 있거나 클러스터의 모든 데이터베이스가 활성화되어 있는 경우 데이터베이스에서 컬렉션을 삭제해도 해당 컬렉션에 대한 변경 스트림이 비활성화되지

않습니다. 삭제된 컬렉션과 동일한 이름으로 새 컬렉션이 생성되면 해당 컬렉션에 대해 변경 스트림이 활성화됩니다.

`$listChangeStreams` 집계 파이프라인 단계를 사용하여 클러스터의 활성화된 모든 변경 스트림을 나열할 수 있습니다. Amazon DocumentDB에서 지원하는 모든 집계 단계는 추가 처리를 위해 파이프라인에서 사용할 수 있습니다. 이전에 활성화된 컬렉션이 비활성화된 경우 `$listChangeStreams` 출력에 나타나지 않습니다.

```
//List all databases and collections with change streams enabled
cursor = new DBCommandCursor(db,
    db.runCommand(
        {aggregate: 1,
        pipeline: [{$listChangeStreams: 1}],
        cursor:{}}));
```

```
//List of all databases and collections with change streams enabled
{ "database" : "test", "collection" : "foo" }
{ "database" : "bar", "collection" : "" }
{ "database" : "", "collection" : "" }
```

```
//Determine if the database “bar” or collection “bar.foo” have change streams enabled
cursor = new DBCommandCursor(db,
  db.runCommand(
      {aggregate: 1,
       pipeline: [{$listChangeStreams: 1},
                  {$match: {$or: [{database: "bar", collection: "foo"},
                                  {database: "bar", collection: ""},
                                  {database: "", collection: ""}]}}
                ],
      cursor:{}}));
```

예: Python에서 변경 스트림 사용

다음은 컬렉션 수준의 Python에서 Amazon DocumentDB 변경 스트림을 사용하는 예입니다.

```
import os
import sys
from pymongo import MongoClient, ReadPreference

username = "DocumentDBusername"
password = <Insert your password>
```

```
clusterendpoint = "DocumentDBClusterEndpoint”
client = MongoClient(clusterendpoint, username=username, password=password, tls='true',
 tlsCAFile='global-bundle.pem')

db = client['bar']

#While ‘Primary’ is the default read preference, here we give an example of
#how to specify the required read preference when reading the change streams
coll = db.get_collection('foo', read_preference=ReadPreference.PRIMARY)
#Create a stream object
stream = coll.watch()
#Write a new document to the collection to generate a change event
coll.insert_one({'x': 1})
#Read the next change event from the stream (if any)
print(stream.try_next())

"""
Expected Output:
{'_id': {'_data': '015daf94f600000002010000000200009025'},
'clusterTime': Timestamp(1571788022, 2),
'documentKey': {'_id': ObjectId('5daf94f6ea258751778163d6')},
'fullDocument': {'_id': ObjectId('5daf94f6ea258751778163d6'), 'x': 1},
'ns': {'coll': 'foo', 'db': 'bar'},
'operationType': 'insert'}
"""

#A subsequent attempt to read the next change event returns nothing, as there are no
 new changes
print(stream.try_next())

"""
Expected Output:
None
"""

#Generate a new change event by updating a document
result = coll.update_one({'x': 1}, {'$set': {'x': 2}})
print(stream.try_next())

"""
Expected Output:
{'_id': {'_data': '015daf99d400000001010000000100009025'},
'clusterTime': Timestamp(1571789268, 1),
```

```
'documentKey': {'_id': ObjectId('5daf9502ea258751778163d7')},
'ns': {'coll': 'foo', 'db': 'bar'},
'operationType': 'update',
'updateDescription': {'removedFields': [], 'updatedFields': {'x': 2}}}
"""
```

다음은 데이터베이스 수준의 Python에서 Amazon DocumentDB 변경 스트림을 사용하는 예입니다.

```
import os
import sys
from pymongo import MongoClient

username = "DocumentDBusername"
password = <Insert your password>
clusterendpoint = "DocumentDBClusterEndpoint”
client = MongoClient(clusterendpoint, username=username, password=password, tls='true',
 tlsCAFile='global-bundle.pem')

db = client['bar']
#Create a stream object
stream = db.watch()
coll = db.get_collection('foo')
#Write a new document to the collection foo to generate a change event
coll.insert_one({'x': 1})

#Read the next change event from the stream (if any)
print(stream.try_next())

"""
Expected Output:
{'_id': {'_data': '015daf94f600000002010000000200009025'},
'clusterTime': Timestamp(1571788022, 2),
'documentKey': {'_id': ObjectId('5daf94f6ea258751778163d6')},
'fullDocument': {'_id': ObjectId('5daf94f6ea258751778163d6'), 'x': 1},
'ns': {'coll': 'foo', 'db': 'bar'},
'operationType': 'insert'}
"""
#A subsequent attempt to read the next change event returns nothing, as there are no
 new changes
print(stream.try_next())

"""
Expected Output:
```

```
None
"""

coll = db.get_collection('foo1')

#Write a new document to another collection to generate a change event
coll.insert_one({'x': 1})
print(stream.try_next())

"""
Expected Output: Since the change stream cursor was the database level you can see
 change events from different collections in the same database
{'_id': {'_data': '015daf94f600000002010000000200009025'},
'clusterTime': Timestamp(1571788022, 2),
'documentKey': {'_id': ObjectId('5daf94f6ea258751778163d6')},
'fullDocument': {'_id': ObjectId('5daf94f6ea258751778163d6'), 'x': 1},
'ns': {'coll': 'foo1', 'db': 'bar'},
'operationType': 'insert'}
"""
```

전체 문서 조회

변경 사항 업데이트 이벤트에는 전체 문서가 포함되지 않으며 변경된 내용만 포함됩니다. 사용 사례에 업데이트의 영향을 받는 전체 문서가 필요한 경우, 스트림을 열 때 전체 문서 조회를 활성화할 수 있습니다.

변경 스트림 업데이트 이벤트에 대한 `fullDocument` 문서는 문서 조회 시 업데이트된 문서의 최신 버전을 나타냅니다. 업데이트 작업과 `fullDocument` 조회 사이에 변경 사항이 발생한 경우 `fullDocument` 문서가 업데이트 당시의 문서 상태를 나타내지 않을 수 있습니다.

업데이트 조회가 활성화된 스트림 객체를 생성하려면 다음 예제를 사용하세요.

```
stream = coll.watch(full_document='updateLookup')

#Generate a new change event by updating a document
result = coll.update_one({'x': 2}, {'$set': {'x': 3}})

stream.try_next()
```

스트림 객체의 결과는 다음과 비슷합니다.

```
{'_id': {'_data': '015daf9b7c000000010100000000100009025'},
```

```
'clusterTime': Timestamp(1571789692, 1),
'documentKey': {'_id': ObjectId('5daf9502ea258751778163d7')},
'fullDocument': {'_id': ObjectId('5daf9502ea258751778163d7'), 'x': 3},
'ns': {'coll': 'foo', 'db': 'bar'},
'operationType': 'update',
'updateDescription': {'removedFields': [], 'updatedFields': {'x': 3}}}
```

변경 스트림 재개

마지막으로 검색된 변경 이벤트 문서의 _id 필드와 동일한 다시 시작 토큰을 사용하여 나중에 변경 스트림을 다시 시작할 수 있습니다.

```
import os
import sys
from pymongo import MongoClient

username = "DocumentDBusername"
password = <Insert your password>
clusterendpoint = "DocumentDBClusterEndpoint”
client = MongoClient(clusterendpoint, username=username, password=password, tls='true',
 tlsCAFile='global-bundle.pem', retryWrites='false')

db = client['bar']
coll = db.get_collection('foo')
#Create a stream object
stream = db.watch()
coll.update_one({'x': 1}, {'$set': {'x': 4}})
event = stream.try_next()
token = event['_id']
print(token)

"""
Output: This is the resume token that we will later us to resume the change stream
{'_data': '015daf9c5b0000000101000000100009025'}
"""
#Python provides a nice shortcut for getting a stream’s resume token
print(stream.resume_token)

"""
Output
{'_data': '015daf9c5b0000000101000000100009025'}
"""
#Generate a new change event by updating a document
```

```
result = coll.update_one({'x': 4}, {'$set': {'x': 5}})
#Generate another change event by inserting a document
result = coll.insert_one({'y': 5})
#Open a stream starting after the selected resume token
stream = db.watch(full_document='updateLookup', resume_after=token)
#Our first change event is the update with the specified _id
print(stream.try_next())

"""
#Output: Since we are resuming the change stream from the resume token, we will see all
 events after the first update operation. In our case, the change stream will resume
 from the update operation {x:5}

{'_id': {'_data': '015f7e8f0c0000000601000000006000fe038'},
'operationType': 'update',
'clusterTime': Timestamp(1602129676, 6),
'ns': {'db': 'bar', 'coll': 'foo'},
'documentKey': {'_id': ObjectId('5f7e8f0ac423bafbfd9adba2')},
'fullDocument': {'_id': ObjectId('5f7e8f0ac423bafbfd9adba2'), 'x': 5},
'updateDescription': {'updatedFields': {'x': 5}, 'removedFields': []}}
"""
#Followed by the insert
print(stream.try_next())

"""
#Output:
{'_id': {'_data': '015f7e8f0c0000000701000000007000fe038'},
'operationType': 'insert',
'clusterTime': Timestamp(1602129676, 7),
'ns': {'db': 'bar', 'coll': 'foo'},
'documentKey': {'_id': ObjectId('5f7e8f0cbf8c233ed577eb94')},
'fullDocument': {'_id': ObjectId('5f7e8f0cbf8c233ed577eb94'), 'y': 5}}
"""
```

`startAtOperationTime`에서 변경 스트림 재개

`startAtOperationTime`을 사용하여 특정 타임스탬프에서 나중에 변경 스트림을 재개할 수 있습니다.

> **Note**
>
> `startAtOperationTime`을 사용하는 기능은 Amazon DocumentDB 4.0 이상에서 사용할 수 있습니다. `startAtOperationTime`을 사용하는 경우, 변경 스트림 커서는 지정된 타

임스탬프 시점 또는 이후에 발생한 변경 사항만 반환합니다. `startAtOperationTime` 및 `resumeAfter` 명령은 상호 배타적이므로 함께 사용할 수 없습니다.

```
import os
import sys
from pymongo import MongoClient

username = "DocumentDBusername"
password = <Insert your password>
clusterendpoint = "DocumentDBClusterEndpoint”
client = MongoClient(clusterendpoint, username=username, password=password, tls='true',
 tlsCAFile='rds-root-ca-2020.pem',retryWrites='false')
db = client['bar']
coll = db.get_collection('foo')
#Create a stream object
stream = db.watch()
coll.update_one({'x': 1}, {'$set': {'x': 4}})
event = stream.try_next()
timestamp = event['clusterTime']
print(timestamp)
"""
Output
Timestamp(1602129114, 4)
"""
#Generate a new change event by updating a document
result = coll.update_one({'x': 4}, {'$set': {'x': 5}})
result = coll.insert_one({'y': 5})
#Generate another change event by inserting a document
#Open a stream starting after specified time stamp

stream = db.watch(start_at_operation_time=timestamp)
print(stream.try_next())

"""
#Output: Since we are resuming the change stream at the time stamp of our first update
 operation (x:4), the change stream cursor will point to that event
{'_id': {'_data': '015f7e941a0000000301000000003000fe038'},
'operationType': 'update',
'clusterTime': Timestamp(1602130970, 3),
'ns': {'db': 'bar', 'coll': 'foo'},
'documentKey': {'_id': ObjectId('5f7e9417c423bafbfd9adbb1')},
'updateDescription': {'updatedFields': {'x': 4}, 'removedFields': []}}
```

```
"""

print(stream.try_next())
"""
#Output: The second event will be the subsequent update operation (x:5)
{'_id': {'_data': '015f7e9502000000050100000005000fe038'},
'operationType': 'update',
'clusterTime': Timestamp(1602131202, 5),
'ns': {'db': 'bar', 'coll': 'foo'},
'documentKey': {'_id': ObjectId('5f7e94ffc423bafbfd9adbb2')},
'updateDescription': {'updatedFields': {'x': 5}, 'removedFields': []}}
"""

print(stream.try_next())

"""
#Output: And finally the last event will be the insert operation (y:5)
{'_id': {'_data': '015f7e9502000000060100000006000fe038'},
'operationType': 'insert',
'clusterTime': Timestamp(1602131202, 6),
'ns': {'db': 'bar', 'coll': 'foo'},
'documentKey': {'_id': ObjectId('5f7e95025c4a569e0f6dde92')},
'fullDocument': {'_id': ObjectId('5f7e95025c4a569e0f6dde92'), 'y': 5}}
"""
```

postBatchResumeToken에서 변경 스트림 재개

Amazon DocumentDB 변경 스트림은 이제 라는 추가 필드를 반환합니다`postBatchResumeToken`. 이 필드는 `$changestream` 명령 및 `getMore` 명령에서 반환됩니다.

Python의 `$changestream` 명령 예제:

```
db.command({"aggregate": "sales", "pipeline": [{ "$changeStream": {}}], "cursor":
 {"batchSize": 1}})
```

예상 결과:

```
cursor" : {
   "firstBatch" : [ ],
   "postBatchResumeToken" : {"_data" : "0167c8cbe60000000004"},
   "id" : NumberLong("9660788144470"),
   "ns" : "test.sales"
```

```
}
```

Python의 getMore 명령 예제:

```
db.command({"getMore": NumberLong(<cursor id>), "collection": "sales", "batchSize":
 1 })
```

예상 결과

```
cursor" : {
   "nextBatch" : [ ],
   "postBatchResumeToken" : {"_data" : "0167c8cbe60000000004"},
   "id" : NumberLong("9660788144470"),
   "ns" : "test.sales"
}
```

재개 토큰 사용 방식과 마찬가지로 postBatchResumeToken 필드를 사용하여 resumeAfter 필드에서 새 변경 스트림 커서를 열 수 있습니다.

선택한 이후에 시작하는 스트림을 엽니다postBatchResumeToken.

```
post_batch_resume_token = output['cursor']['postBatchResumeToken']
stream = db.watch(full_document='updateLookup', resume_after=post_batch_resume_token)
```

실제 이벤트를 반영하는 작업 로그(oplog) 항목에 항상 해당하는 일반 재개 토큰과 달리는 변경 스트림이 서버에서 스캔한 oplog 항목에 postBatchResumeToken 해당하며, 이는 반드시 일치하는 변경 사항은 아닙니다.

이전 일반 재개 토큰으로 재개를 시도하면 데이터베이스가 지정된 타임스탬프와 현재 시간 사이의 모든 oplog 항목을 강제로 스캔합니다. 이렇게 하면 짧은 시간 동안 각 하위 쿼리 스캔을 통해 내부적으로 많은 쿼리가 생성될 수 있습니다. 이로 인해 CPU 사용량이 급증하고 데이터베이스 성능이 저하됩니다. 마지막 로 재개하면 일치하지 않는 oplog 항목의 스캔이 postBatchResumeToken 건너뜁니다.

변경 스트림 내 트랜잭션

변경 스트림 이벤트에는 커밋되지 않았거나 중단된 트랜잭션의 이벤트는 포함되지 않습니다. 예를 들어 하나의 INSERT 작업과 하나의 UPDATE 작업으로 트랜잭션을 시작하고 INSERT 작업이 성공하지만 UPDATE 작업이 실패하면 트랜잭션이 롤백됩니다. 이 트랜잭션이 롤백되었으므로 변경 스트림에는 이 트랜잭션에 대한 이벤트가 포함되지 않습니다.

변경 스트림 로그 보존 기간 수정

AWS Management Console 또는를 사용하여 변경 스트림 로그 보존 기간을 1시간에서 7일로 수정할 수 있습니다 AWS CLI.

Using the AWS Management Console

변경 스트림 로그 보존 기간을 수정하려면

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 파라미터 그룹을 선택합니다.

   > **Tip**
   > 화면 왼쪽에 탐색 창이 표시되지 않으면 페이지 왼쪽 상단 모서리에서 메뉴 아이콘 (≡) 을 선택합니다.

3. 파라미터 그룹 창에서 클러스터와 연결된 클러스터 파라미터 그룹을 선택합니다. 클러스터와 연결된 클러스터 파라미터 그룹을 식별하려면 Amazon DocumentDB 클러스터의 파라미터 그룹 확인 단원을 참조하십시오.
4. 결과 페이지에는 이 클러스터 파라미터 그룹에 대한 파라미터 및 해당 세부 정보가 표시됩니다. `change_stream_log_retention_duration` 파라미터를 선택합니다.
5. 페이지 오른쪽 상단에서 편집을 선택하여 파라미터의 값을 변경합니다. `change_stream_log_retention_duration` 파라미터는 1시간~7일 사이로 수정할 수 있습니다.
6. 변경한 다음 클러스터 파라미터 수정을 선택하여 변경 사항을 저장합니다. 변경 사항을 취소하려면 취소를 선택합니다.

Using the AWS CLI

클러스터 파라미터 그룹의 `change_stream_log_retention_duration` 파라미터를 수정하려면 다음 파라미터와 함께 `modify-db-cluster-parameter-group` 작업을 사용합니다.

- **`--db-cluster-parameter-group-name`** - 필수입니다. 수정하려는 클러스터 파라미터 그룹의 이름입니다. 클러스터와 연결된 클러스터 파라미터 그룹을 식별하려면 Amazon DocumentDB 클러스터의 파라미터 그룹 확인 단원을 참조하십시오.

- **--parameters** - 필수입니다. 수정 중인 파라미터입니다. 각 파라미터 요소는 다음을 포함해야 합니다.
  - **ParameterName** — 수정 중인 파라미터의 이름입니다. 이 경우에는 change_stream_log_retention_duration입니다.
  - **ParameterValue** — 이 파라미터의 새 값입니다.
  - **ApplyMethod** - 파라미터에 대한 변경 사항을 적용할 방법입니다. 허용된 값은 immediate 및 pending-reboot입니다.

> **Note**
>
> static의 ApplyType 파라미터에는 pending-reboot의 ApplyMethod이 있어야 합니다.

1. 파라미터 change_stream_log_retention_duration의 값을 변경하려면 다음 명령을 실행하고 parameter-value를 파라미터를 수정할 값으로 바꿉니다.

   Linux, macOS, Unix의 경우:

```
aws docdb modify-db-cluster-parameter-group \
    --db-cluster-parameter-group-name sample-parameter-group \
    --parameters
 "ParameterName=change_stream_log_retention_duration,ParameterValue=<parameter-
value>,ApplyMethod=immediate"
```

   Windows의 경우:

```
aws docdb modify-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name sample-parameter-group ^
    --parameters
 "ParameterName=change_stream_log_retention_duration,ParameterValue=<parameter-
value>,ApplyMethod=immediate"
```

   이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
    "DBClusterParameterGroupName": "sample-parameter-group"
}
```

2. 적어도 5분을 기다립니다.
3. `sample-parameter-group`의 파라미터 값을 나열하여 변경 사항을 확인합니다.

   Linux, macOS, Unix의 경우:

```
aws docdb describe-db-cluster-parameters \
    --db-cluster-parameter-group-name sample-parameter-group
```

   Windows의 경우:

```
aws docdb describe-db-cluster-parameters ^
    --db-cluster-parameter-group-name sample-parameter-group
```

   이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
    "Parameters": [
        {
            "ParameterName": "audit_logs",
            "ParameterValue": "disabled",
            "Description": "Enables auditing on cluster.",
            "Source": "system",
            "ApplyType": "dynamic",
            "DataType": "string",
            "AllowedValues": "enabled,disabled",
            "IsModifiable": true,
            "ApplyMethod": "pending-reboot"
        },
        {
            "ParameterName": "change_stream_log_retention_duration",
            "ParameterValue": "12345",
            "Description": "Duration of time in seconds that the change stream
 log is retained and can be consumed.",
            "Source": "user",
            "ApplyType": "dynamic",
            "DataType": "integer",
            "AllowedValues": "3600-86400",
            "IsModifiable": true,
            "ApplyMethod": "immediate"
        }
    ]
```

```
}
```

> **Note**
>
> 변경 스트림 로그 보존은 로그 크기가 (>) 51,200MB보다 커질 때까지 구성된 `change_stream_log_retention_duration` 값보다 오래된 로그를 삭제하지 않습니다.

보조 인스턴스에서 변경 스트림 사용

보조 인스턴스에서 변경 스트림 사용을 시작하려면 `readPreference` 항목을 보조 인스턴스로 사용하여 변경 스트림 커서를 엽니다.

변경 스트림 커서를 열어 클러스터 또는 데이터베이스의 특정 컬렉션 또는 모든 컬렉션에 대한 변경 이벤트를 감시할 수 있습니다. Amazon DocumentDB 인스턴스에서 변경 스트림 커서를 열고 라이터 인스턴스와 리더 인스턴스 모두에서 변경 스트림 문서를 가져올 수 있습니다. 라이터 및 리더 인스턴스에서 열린 다양한 변경 스트림 커서에서 변경 스트림 토큰(예: `resumeToken` 또는 `startOperationTime`)을 공유할 수 있습니다.

예제

```
import os
import sys
from pymongo import MongoClient, ReadPreference

username = "DocumentDBusername"
password = <Your password>

clusterendpoint = "DocumentDBClusterEndpoint"

client = MongoClient(clusterendpoint, username=username, password=password, tls='true',
 tlsCAFile='global-bundle.pem')

db = client['bar']

# Make sure to use SECONDARY to redirect cursor reads from secondary instances
coll = db.get_collection('foo', read_preference=ReadPreference.SECONDARY)

# Create a stream object on RO. The token needs to generated from PRIMARY.
stream = coll.watch(resumeAfter=token)
```

```
for event in stream:
   print(event)
```

보조 인스턴스의 변경 스트림에 대한 지침 및 제한 사항

- 스트림 변경 이벤트는 기본 인스턴스에서 보조 인스턴스로 복제해야 합니다. Amazon CloudWatch를 사용하여 `DBInstanceReplicaLag` 지표를 모니터링할 수도 있습니다.
- 보조 인스턴스의 타임스탬프가 항상 기본 인스턴스와 동기화되지는 않을 수 있습니다. 이 경우 보조 인스턴스 타임스탬프가 지연될 수 있어 따라잡을 수 있습니다. 모범 사례로 `startAtOperationTime` 또는 `resumeToken`를 사용하여 보조 인스턴스에서 감시를 시작하는 것이 좋습니다.
- 문서 크기가 크고, `fullDocumentLookup` 작업을 수행 중이며, 기본 인스턴스에 동시 쓰기 워크로드가 많은 경우 기본 인스턴스에 비해 보조 인스턴스의 처리량이 낮은 상황을 경험할 수도 있습니다. 가장 좋은 방법은 보조 인스턴스에서 버퍼 캐시 적중률을 모니터링하고 버퍼 캐시 적중률이 높은지 확인하는 것입니다.

변경 스트림과 AWS Lambda 함께 사용

Amazon DocumentDB는와 통합되어 AWS Lambda 있으므로 Lambda 함수를 사용하여 변경 스트림의 레코드를 처리할 수 있습니다. Lambda 이벤트 소스 매핑은 Lambda를 직접 호출하지 않는 Amazon DocumentDB 이벤트를 처리하기 위해 Lambda 함수를 호출하는 데 사용할 수 있는 리소스입니다. Amazon DocumentDB 변경 스트림을 이벤트 소스로 사용하면 데이터 변경에 응답하는 이벤트 기반 애플리케이션을 구축할 수 있습니다. 예를 들어 Lambda 함수를 사용하여 새 문서를 처리하거나, 기존 문서에 대한 업데이트를 추적하거나, 삭제된 문서를 로그할 수 있습니다.

이벤트 소스 매핑을 구성하여 Amazon DocumentDB 변경 스트림의 레코드를 Lambda 함수로 전송하도록 할 수 있습니다. 이벤트는 한 번에 하나씩 전송하거나 효율성을 높이기 위해 일괄 처리할 수 있으며 순서대로 처리됩니다. 이벤트 소스 매핑의 배치 동작은 특정 시간대(0 - 300초) 또는 배치 레코드 수(최대 10,000개 레코드 제한)를 기준으로 구성할 수 있습니다. 여러 개의 이벤트 소스 매핑을 생성하여 동일한 데이터를 여러 개의 Lambda 함수로 처리하거나 여러 스트림에서 구별되는 항목을 단일 함수로 처리할 수 있습니다.

함수가 오류를 반환하면 Lambda는 처리가 성공할 때까지 배치를 재시도합니다. 변경 스트림의 이벤트가 만료된 경우 Lambda는 이벤트 소스 매핑을 비활성화합니다. 이 경우 새 이벤트 소스 매핑을 생성하고 원하는 시작 위치로 구성할 수 있습니다. Lambda 이벤트 소스 매핑은 폴러의 분산 특성으로 인해 이벤트를 한 번 이상 처리합니다. 결과적으로 Lambda 함수는 드물게 중복 이벤트를 수신할 수 있습니

다. 중복 이벤트와 관련된 문제를 방지하려면 AWS Lambda 함수 작업 모범 사례를 따르고 자발성 함수를 구축합니다. 자세한 내용은 AWS Lambda 개발자 안내서의 Amazon DocumentDB AWS Lambda console 에서 사용을 참조하세요.

성능 모범 사례에 따라 Lambda 함수는 수명이 짧아야 합니다. 불필요한 처리 지연을 방지하기 위해 복잡한 로직도 실행하지 않아야 합니다. 특히 고속 스트림의 경우 장기 실행 중인 동기식 Lambda보다 비동기식 사후 처리 단계 함수 워크플로를 트리거하는 것이 좋습니다. 에 대한 자세한 내용은 AWS Lambda 개발자 안내서를 AWS Lambda참조하세요.

제한 사항

다음은 Amazon DocumentDB 및 작업 시 고려해야 할 제한 사항입니다. AWS Lambda

- AWS Lambda 는 현재 Amazon DocumentDB 4.0 및 5.0에서만 지원됩니다.
- AWS Lambda 는 현재 탄력적 클러스터 또는 글로벌 클러스터에서 지원되지 않습니다.
- AWS Lambda 페이로드 크기는 6MB를 초과할 수 없습니다. Lambda 배치 크기에 대한 자세한 내용은 AWS Lambda 개발자 가이드의 Lambda 이벤트 소스 매핑 섹션의 "배치 동작"을 참조하세요.

Amazon DocumentDB 탄력적 클러스터 사용

Amazon DocumentDB 탄력적 클러스터는 초당 수백만 읽기/쓰기 및 페타바이트의 스토리지 용량을 갖춘 워크로드를 지원합니다. 또한 탄력적 클러스터는 인스턴스를 선택, 관리 또는 업그레이드할 필요가 없으므로 개발자가 Amazon DocumentDB와 상호 작용하는 방식을 단순화합니다.

Amazon DocumentDB 탄력적 클러스터는 다음과 같이 생성되었습니다.

- 풍부한 쿼리 기능과 MongoDB API 호환성을 통해 사실상 무한한 규모를 제공하는 데이터베이스를 찾는 고객에게 솔루션을 제공합니다.
- 고객에게 더 높은 연결 제한을 제공하고 패치로 인한 가동 중지 시간을 줄입니다.
- JSON 워크로드를 위한 클라우드 네이티브, 엘라스틱 및 최고 등급의 아키텍처에 지속적으로 투자하세요.

주제

- 탄력적 클러스터 사용 사례
- 탄력적 클러스터의 장점
- 탄력적 클러스터 리전 및 버전 가용성
- 제한 사항
- Amazon DocumentDB 탄력적 클러스터: 작동 방식
- Amazon DocumentDB 엘라스틱 클러스터 시작하기
- Amazon DocumentDB 탄력적 클러스터 모범 사례
- Amazon DocumentDB 탄력적 클러스터 관리
- Amazon DocumentDB 탄력적 클러스터에 대한 저장 데이터 암호화
- 탄력적 클러스터에서 서비스 연결 역할

탄력적 클러스터 사용 사례

문서 데이터베이스는 빠르고 반복적인 개발을 위해 유연한 스키마가 필요한 워크로드에 유용합니다. Amazon DocumentDB 사용 사례의 예는 문서 데이터베이스 사용 사례을 참조하세요.

다음은 탄력적 클러스터가 중요한 이점을 제공하는 일부 사용 사례의 예입니다.

사용자 프로필

문서 데이터베이스에는 유연한 스키마가 있으므로 규모에서 속성과 데이터 값이 다른 문서를 저장할 수 있습니다. 탄력적 클러스터는 서로 다른 사용자가 다양한 유형의 정보를 제공하는 온라인 프로필에 대한 실용적인 솔루션입니다. 내 애플리케이션이 수억 개의 사용자 프로필을 지원한다고 가정해 보겠습니다. 탄력적 클러스터를 사용하여 이러한 애플리케이션을 지원하실 수 있습니다. 사용자 프로필에 대한 수백만 건의 쓰기 및 읽기를 지원하도록 규모를 확장 및 축소할 수 있기 때문입니다. 피크 이외의 시간에도 확장하여 비용을 절감할 수 있습니다.

콘텐츠 관리 및 이력 기록

콘텐츠를 효과적으로 관리하려면 다양한 소스에서 콘텐츠를 수집 및 집계한 다음 고객에게 제공할 수 있어야 합니다. 문서 데이터베이스는 유연한 스키마로 인해 모든 유형의 데이터 수집과 저장에 적합합니다. 이미지, 코멘트 및 비디오와 같은 사용자 생성 콘텐츠를 포함하여 새로운 유형의 콘텐츠를 생성하고 통합하는 데 사용할 수 있습니다. 시간이 지남에 따라 데이터베이스에 더 많은 스토리지가 필요할 수 있습니다. 엘라스틱 클러스트를 사용하면 더 많은 스토리지 볼륨에 데이터를 분산하여 단일 클러스터에 페타바이트 규모의 데이터를 저장할 수 있습니다.

탄력적 클러스터의 장점

AWS 서비스 통합

Amazon DocumentDB 탄력적 클러스터는 Amazon DocumentDB와 동일한 방식으로 다른 AWS 서비스와 통합됩니다.

- 마이그레이션 - AWS Database Migration Service (DMS)를 사용하여 MongoDB 및 기타 관계형 데이터베이스에서 Amazon DocumentDB 탄력적 클러스터로 마이그레이션할 수 있습니다.
- 모니터링 - Amazon CloudWatch를 사용하여 탄력적 클러스터의 상태와 성능을 모니터링할 수 있습니다.
- 보안 - AWS Identity and Access Management (IAM)을 통해 인증 및 권한 부여를 설정하여 탄력적 클러스터 생성 및 업데이트와 같은 탄력적 클러스터 리소스를 관리하고 보안 VPC 전용 연결에 Amazon VPC를 사용할 수 있습니다. 탄력적 클러스터에 대한 IAM 인증 로그인은 지원되지 않습니다.
- 데이터 관리 - AWS Glue 를 사용하여 Amazon S3, Amazon Redshift 및 Amazon OpenSearch Service와 같은 다른 AWS 서비스에서 데이터를 가져오고 내보낼 수 있습니다.

탄력적 클러스터 리전 및 버전 가용성

탄력적 클러스터의 리전 가용성

다음 표에는 Amazon DocumentDB 탄력적 클러스터를 현재 사용할 수 있는 AWS 리전과 각 리전의 엔드포인트가 나와 있습니다.

지역명	지역	가용 영역
미국 동부(버지니아 북부)	us-east-1	5
미국 동부(오하이오)	us-east-2	3
미국 서부(오리건)	us-west-2	3
아시아 태평양(홍콩)	ap-east-1	3
아시아 태평양(뭄바이)	ap-south-1	3
아시아 태평양(서울)	ap-northeast-2	3
아시아 태평양(싱가포르)	ap-southeast-1	3
아시아 태평양(시드니)	ap-southeast-2	3
아시아 태평양(도쿄)	ap-northeast-1	3
캐나다(중부)	ca-central-1	3
남아메리카(상파울루)	sa-east-1	3
유럽(프랑크푸르트)	eu-central-1	3
유럽(아일랜드)	eu-west-1	3
유럽(런던)	eu-west-2	3
유럽(밀라노)	eu-south-1	3
유럽(파리)	eu-west-3	3

버전 가용성

탄력적 클러스터는 MongoDB 5.0과 호환되는 와이어 프로토콜을 지원합니다. Amazon DocumentDB 4.0 인스턴스 기반 클러스터와 탄력적 클러스터 간의 차이점은 섹션을 참조하세요Amazon DocumentDB 4.0과 탄력적 클러스터 간의 기능적 차이.

제한 사항

탄력적 클러스터 관리

다음 클러스터 관리 기능 및 기능은이 릴리스에서 지원되지 않습니다.

- IAM 사용자 및 역할을 사용하여 데이터베이스에 인증할 수 있는 기능
- 글로벌 클러스터를 생성할 수 있는 기능.
- 기존 Amazon DocumentDB 이벤트와 이벤트 구독
- 레인지 샤딩
- 기존 컬렉션 샤드
- 멀티필드 샤드 키
- 샤드 키 변경하기
- 특정 시점 복원
- 복제
- 성능 개선 도우미

> **Note**
>
> 탄력적 클러스터 제한에 대한 자세한 내용은 Amazon DocumentDB 할당량 및 제한을 참조하세요.

쿼리 및 쓰기 작업

이 릴리스에서는 다음 쿼리 및 쓰기 작업 명령과 기능이 지원되지 않습니다.

- 규모 조정 작업 중의 DDL 명령

- 프로파일러
- 파라미터 그룹
- AWS Config
- AWS Backup

컬렉션 및 인덱스 관리

이 릴리스에서는 다음 컬렉션 및 인덱스 관리 기능이 지원되지 않습니다.

- 고유 인덱스
- 부분 인덱스
- 텍스트 인덱스
- 벡터 인덱스
- 문서 압축

관리 및 진단

이 릴리스에서는 다음 관리 및 진단 명령과 기능이 지원되지 않습니다.

- AWS Secrets Manager
- 역할 기반 액세스 제어(RBAC) 사용자 정의 역할.
- 연결 시 0의 쓰기 문제는 지원되지 않습니다.
- 기존 탄력적 클러스터에 현재 할당되지 않은 VPC에 속하는 서브넷을 변경.

옵트인 기능

이 릴리스에서는 다음 Amazon DocumentDB 옵트인 기능이 지원되지 않습니다.

- ACID 트랜잭션
- DDL/DML 감사
- Change streams
- 세션 명령

Amazon DocumentDB 탄력적 클러스터: 작동 방식

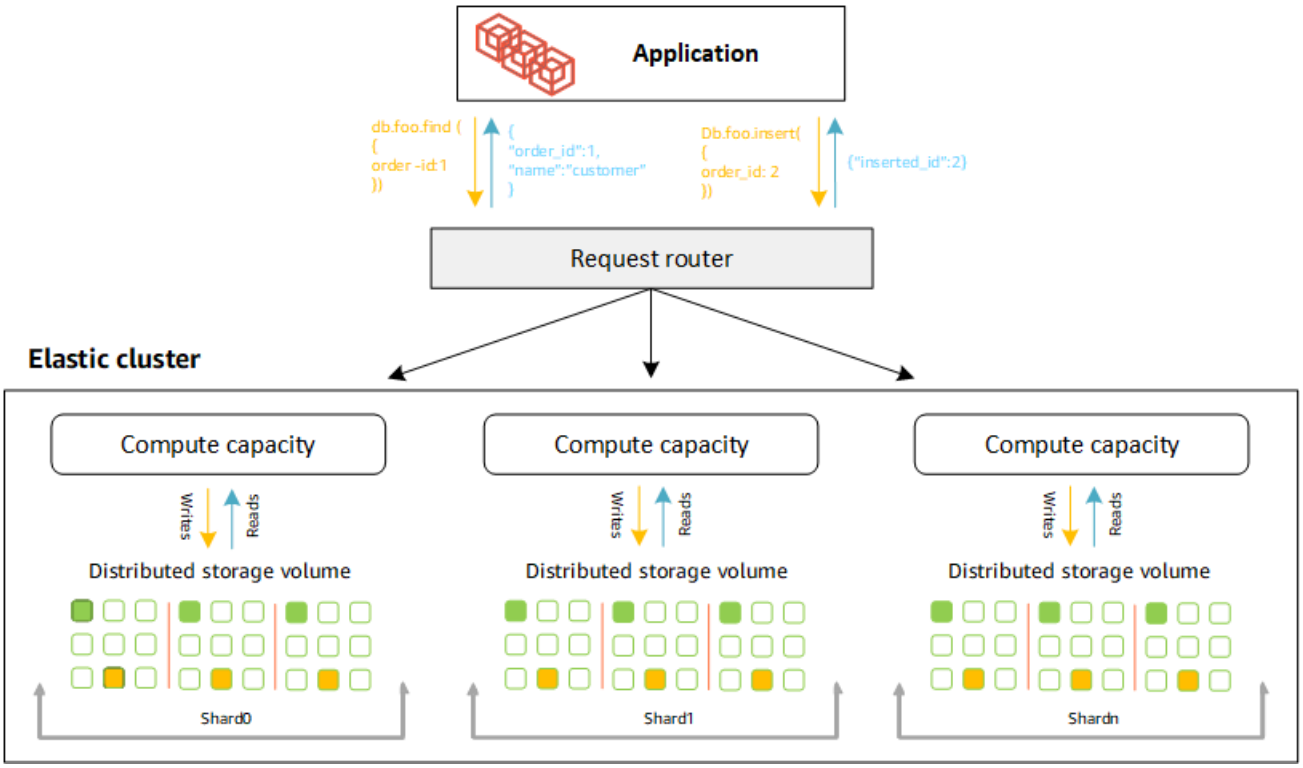
이 섹션의 주제에서는 Amazon DocumentDB 탄력적 클러스터를 지원하는 메커니즘 및 기능에 대한 정보를 제공합니다.

주제

- Amazon DocumentDB 탄력적 클러스터 샤딩
- 탄력적 클러스터 마이그레이션
- 탄력적 클러스터 규모 조정
- 탄력적 클러스터 안정성
- 탄력적 클러스터 스토리지 및 가용성
- Amazon DocumentDB 4.0과 탄력적 클러스터 간의 기능적 차이

Amazon DocumentDB 탄력적 클러스터 샤딩

Amazon DocumentDB 탄력적 클러스터는 해시 기반 샤딩을 사용하여 분산 스토리지 시스템 전체에서 데이터를 분할합니다. 파티셔닝이라고도 하는 샤딩은 대규모 데이터 세트를 여러 노드에 걸쳐 작은 데이터 세트로 분할하므로 수직 규모 조정 한도 이상으로 데이터베이스를 규모 조정할 수 있습니다. 탄력적 클러스터는 Amazon DocumentDB에서 컴퓨팅 및 스토리지의 분리 또는 '분리'를 사용하므로 서로 독립적으로 확장할 수 있습니다. 탄력적 클러스터는 컴퓨팅 노드 간에 작은 데이터 청크를 이동하여 컬렉션을 다시 분할하는 대신 분산 스토리지 시스템 내에서 데이터를 효율적으로 복사합니다.



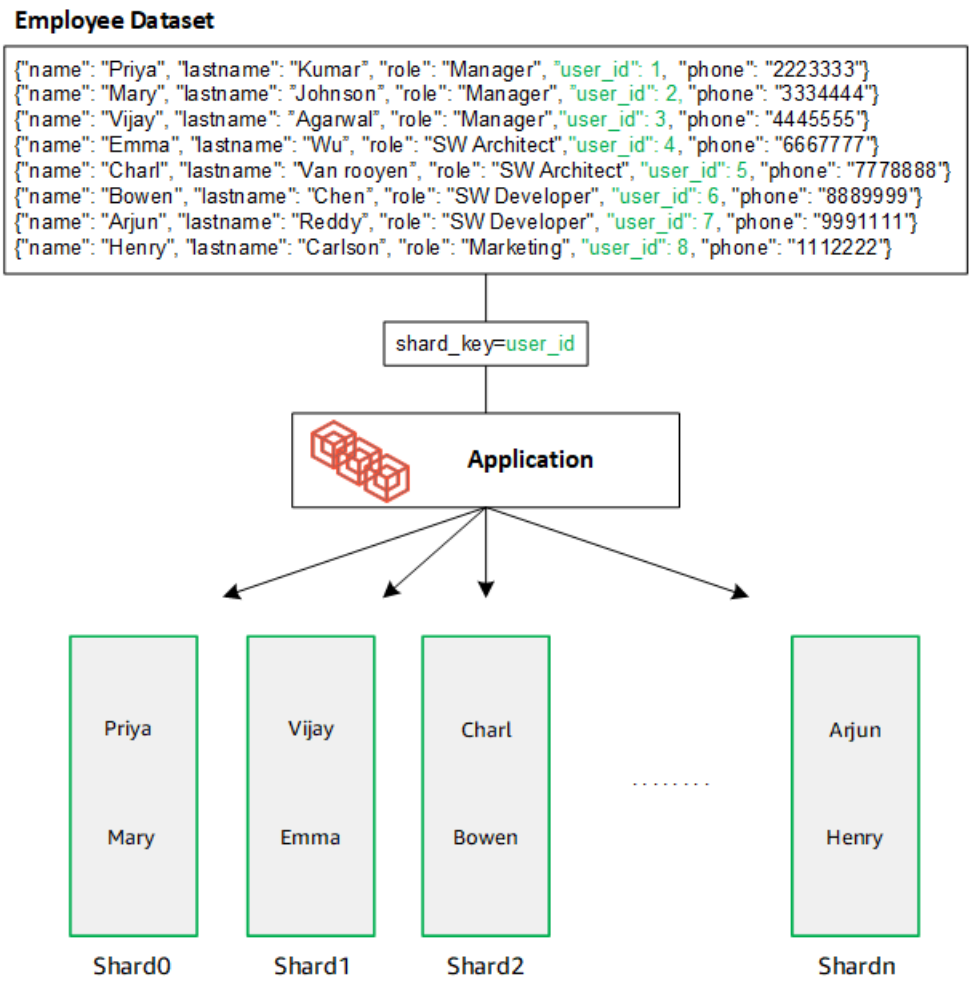
샤드 정의

샤드 명명법의 정의:

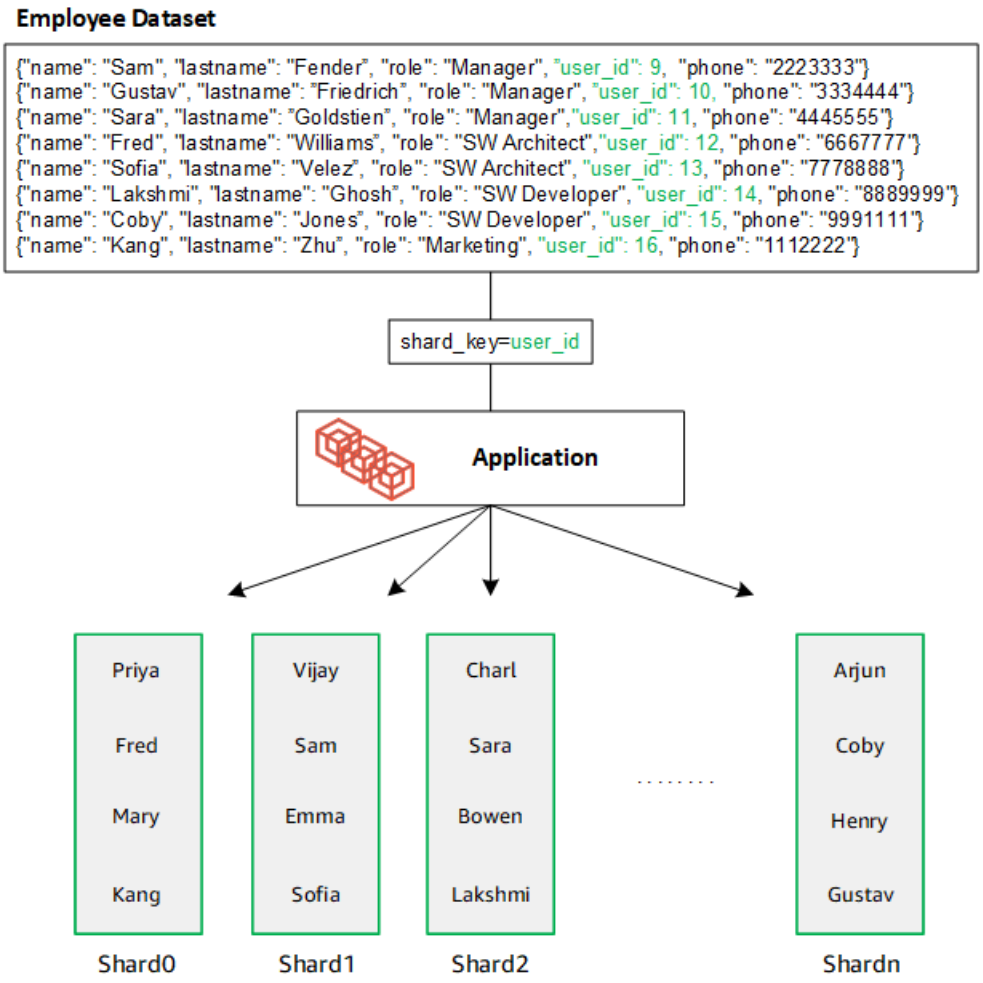
- 샤드 - 샤드는 탄력적 클러스터를 위한 컴퓨팅을 제공합니다. 단일 라이터 인스턴스와 0~15개의 읽기 전용 복제본이 있습니다. 기본적으로 샤드에는 라이터와 단일 읽기 전용 복제본이라는 두 개의 인스턴스가 있습니다. 최대 32개의 샤드를 구성할 수 있으며 각 샤드 인스턴스는 최대 64개의 vCPUs.
- 샤드 키 - 샤드 키는 탄력적 클러스터가 읽기 및 쓰기 트래픽을 일치하는 샤드에 분배하는 데 사용하는 샤드 컬렉션의 JSON 문서의 필수 필드입니다.
- 샤딩된 컬렉션 - 샤딩된 컬렉션은 데이터가 데이터 파티션의 탄력적 클러스터에 분산되는 컬렉션입니다.
- 파티션 - 파티션은 샤딩된 데이터의 논리적 부분입니다. 샤딩된 컬렉션을 만들면 샤드 키를 기반으로 데이터가 각 샤드 내의 파티션으로 자동 구성됩니다. 각 샤드에는 여러 개의 파티션이 있습니다.

구성된 샤드에 데이터 배포

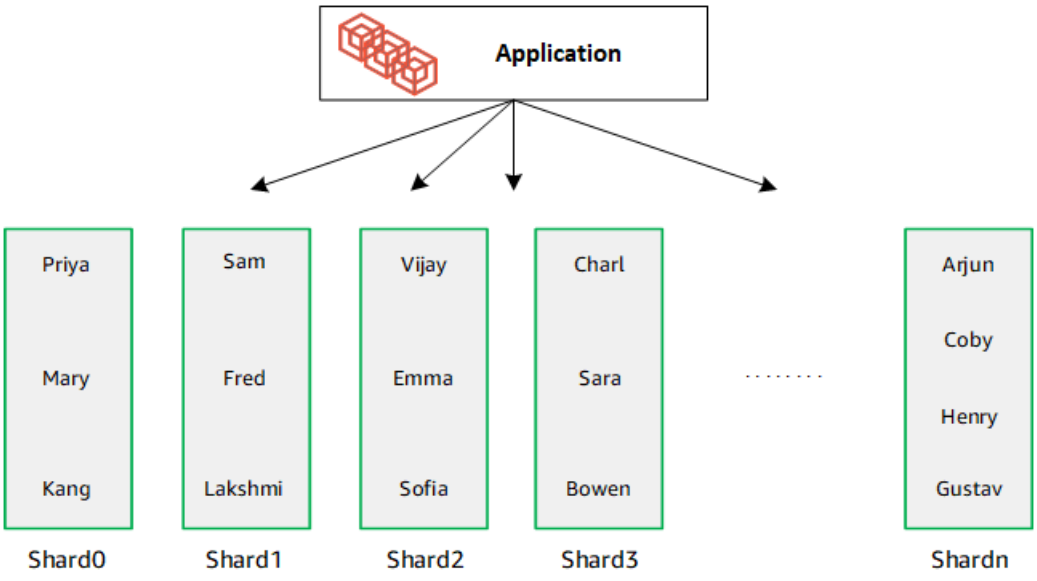
고유한 값이 많은 샤드 키를 생성하세요. 좋은 샤드 키는 데이터를 기본 분할된 데이터베이스로 균등하게 분할하여 워크로드에 최상의 처리량과 성능을 제공합니다. 다음은 "user_id"라는 샤드 키를 사용하는 직원 이름 데이터를 예로 들 수 있습니다.



DocumentDB는 해시 샤딩을 사용하여 데이터를 기본 샤드로 분할합니다. 추가 데이터는 다음과 같은 방식으로 삽입 및 배포됩니다.



샤드를 추가하여 데이터베이스를 규모 조정하면 Amazon DocumentDB가 데이터를 자동으로 재배포합니다.



탄력적 클러스터 마이그레이션

Amazon DocumentDB는 MongoDB 샤딩 데이터를 탄력적 클러스터로 마이그레이션하는 것을 지원합니다. 오프라인, 온라인 및 하이브리드 마이그레이션 방법이 지원됩니다. 자세한 내용은 Amazon DocumentDB로 마이그레이션 단원을 참조하십시오.

탄력적 클러스터 규모 조정

Amazon DocumentDB 탄력적 클러스터는 탄력적 클러스터의 샤드 수(스케일 아웃)와 각 샤드에 적용된 vCPUs 수(스케일 업)를 늘릴 수 있는 기능을 제공합니다. 또한 필요에 따라 샤드 수와 컴퓨팅 용량(vCPU)을 줄일 수 있습니다.

규모 조정 모범 사례는 탄력적 클러스터 크기 조정을 참조하세요.

> **Note**
>
> 클러스터 수준 규모 조정도 가능합니다. 자세한 내용은 Amazon DocumentDB 클러스터 규모 조정 단원을 참조하십시오.

탄력적 클러스터 안정성

Amazon DocumentDB는 신뢰성, 내구성 및 내결함성을 제공하도록 설계되었습니다. 가용성을 개선하기 위해 탄력적 클러스터는 서로 다른 가용 영역에 배치된 샤드당 두 개의 노드를 배포합니다. Amazon DocumentDB에는 신뢰할 수 있는 데이터베이스 솔루션으로 만들어 주는 몇 가지 자동 기능이 포함되어 있습니다. 자세한 내용은 Amazon DocumentDB 신뢰성 단원을 참조하십시오.

탄력적 클러스터 스토리지 및 가용성

Amazon DocumentDB 데이터는 SSD(Solid State Drive)를 사용하는 단일 가상 볼륨인 클러스터 볼륨에 저장됩니다. 클러스터 볼륨은 단일 AWS 리전의 여러 가용 영역에 자동으로 복제되는 6개의 데이터 복사본으로 구성됩니다. 이 복제를 통해 데이터의 내구성을 높이고 데이터 손실 가능성을 줄일 수 있습니다. 또한 다른 가용 영역에 데이터 복사본이 이미 있어 장애 조치가 이루어지는 동안 클러스터 가용성이 높아집니다. 스토리지, 고가용성 및 복제에 대한 자세한 내용은 섹션을 참조하세요Amazon DocumentDB: 작동 방식.

Amazon DocumentDB 4.0과 탄력적 클러스터 간의 기능적 차이

Amazon DocumentDB 4.0과 탄력적 클러스터 간에는 다음과 같은 기능적 차이가 있습니다.

- top 및 collStats의 결과는 샤드로 분할됩니다. 샤딩된 컬렉션의 경우 데이터는 여러 파티션에 분산되고 collStats 보고서는 파티션collScans에서 집계됩니다.
- 클러스터 샤드 수가 변경되면 샤딩된 컬렉션의 컬렉션 top 통계와 collStats 샤딩된 컬렉션의 컬렉션 통계가 재설정됩니다.
- 이제 백업 빌트인 역할이 지원됩니다. serverStatus 작업 - 백업 역할을 가진 개발자 및 응용프로그램은 Amazon DocumentDB 클러스터의 상태에 대한 통계를 수집할 수 있습니다.
- replSetGetConfig 출력에서 SecondaryDelaySecs 필드는 slaveDelay를 대체합니다.
- hello 명령은 isMaster를 대체합니다 - hello는 탄력적 클러스터의 역할을 설명하는 문서를 반환합니다.
- 탄력적 클러스터의 $elemMatch 연산자는 배열의 첫 번째 중첩 수준에 있는 문서만 일치시킵니다. Amazon DocumentDB 4.0에서는 작업자가 모든 수준을 순회한 후 일치하는 문서를 반환합니다. 예시:

```
db.foo.insert(
[
    {a: {b: 5}},
    {a: {b: [5]}},
    {a: {b: [3, 7]}},
    {a: [{b: 5}]},
    {a: [{b: 3}, {b: 7}]},
    {a: [{b: [5]}]},
    {a: [{b: [3, 7]}]},
    {a: [[{b: 5}]]},
    {a: [[{b: 3}, {b: 7}]]},
    {a: [[{b: [5]}]]},
    {a: [[{b: [3, 7]}]]}
]);
// Elastic clusters
> db.foo.find({a: {$elemMatch: {b: {$elemMatch: {$lt: 6, $gt: 4}}}}}, {_id: 0})
{ "a" : [ { "b" : [ 5 ] } ] }

// Docdb 4.0: traverse more than one level deep
> db.foo.find({a: {$elemMatch: {b: {$elemMatch: {$lt: 6, $gt: 4}}}}}, {_id: 0})
{ "a" : [ { "b" : [ 5 ] } ] }
{ "a" : [ [ { "b" : [ 5 ] } ] ] }
```

- Amazon DocumentDB 4.0의 "$" 프로젝션은 모든 필드가 포함된 모든 문서를 반환합니다. 탄력적 클러스터의 경우 "$" 프로젝션을 사용하는 `find` 명령은 "$" 프로젝션과 일치하는 필드만 포함하는 쿼리 파라미터와 일치하는 문서를 반환합니다.
- 엘라스틱 클러스터에서 `$regex` 및 `$options` 쿼리 파라미터가 있는 `find` 명령은 "$regex 및 $options 모두에서 옵션을 설정할 수 없음"이라는 오류를 반환합니다.

- 탄력적 클러스터를 사용하면 `$indexOfCP`은 이제 다음과 같은 경우 "-1"을 반환합니다.
  - `string expression`에서 하위 문자열을 찾을 수 없거나
  - `start`이 `end`보다 큰 숫자이거나
  - `start`이 문자열의 바이트 길이보다 큰 수인 경우입니다.

  Amazon DocumentDB 4.0에서 `$indexOfCP`는 `start` 위치가 `end`이나 문자열의 바이트 길이보다 크면 "0"을 반환합니다.
- 탄력적 클러스터를 사용하면 `_id fields`와 같은의 프로젝션 작업이 프로젝션된 필드만 포함하는 문서를 `{"_id.nestedField" : 1}`반환합니다. 한편 Amazon DocumentDB 4.0에서는 중첩된 필드 프로젝션 명령이 문서를 필터링하지 않습니다.

Amazon DocumentDB 엘라스틱 클러스터 시작하기

이 시작하기 섹션에서는 첫 번째 엘라스틱 클러스터를 생성하고 쿼리하는 방법을 안내합니다.

Amazon DocumentDB에 연결하고 이를 통해 시작하는 방법에는 여러 가지가 있습니다. 다음 절차는 사용자가 강력한 문서 데이터베이스를 사용하여 시작할 수 있는 가장 빠르고 간단하며 쉬운 방법입니다. 이 가이드에서는 AWS CloudShell를 사용하여에서 직접 Amazon DocumentDB 클러스터를 연결하고 쿼리합니다 AWS Management Console. AWS 프리 티어 자격이 있는 신규 고객은 Amazon DocumentDB 및 CloudShell을 무료로 사용할 수 있습니다. AWS CloudShell 환경 또는 Amazon DocumentDB 탄력적 클러스터가 프리 티어 이외의 리소스를 사용하는 경우 해당 리소스에 대한 일반 AWS 요금이 부과됩니다. 이 안내서를 통해 5분 이내에 Amazon DocumentDB를 시작할 수 있습니다.

주제

- 사전 조건
- 1단계: 엘라스틱 클러스터 생성
- 2단계: 엘라스틱 클러스터에 연결
- 3단계: 수집 샤딩, 데이터 삽입 및 쿼리
- 4단계: 탐색

사전 조건

첫 번째 Amazon DocumentDB 클러스터를 작성하기 전에 다음 작업을 수행해야 합니다:

Amazon Web Services(AWS) 계정 만들기

Amazon DocumentDB 사용을 시작하기 전에 Amazon Web Services(AWS) 계정이 있어야 합니다. AWS 계정은 무료입니다. 사용하는 서비스 및 리소스에 대해서만 비용을 지불하는 것입니다.

이 없는 경우 다음 단계를 AWS 계정완료하여 생성합니다.

에 가입하려면 AWS 계정

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.

   에 가입하면 AWS 계정AWS 계정 루트 사용자이 생성됩니다. 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.

필요한 AWS Identity and Access Management (IAM) 권한을 설정합니다.

클러스터, 인스턴스 및 클러스터 파라미터 그룹과 같은 Amazon DocumentDB 리소스를 관리하기 위한 액세스에는가 요청을 인증하는 데 사용할 AWS 수 있는 자격 증명이 필요합니다. 자세한 내용은 Amazon DocumentDB의 ID 및 액세스 관리 단원을 참조하십시오.

1. 의 검색 창에 IAM을 AWS Management Console입력하고 드롭다운 메뉴에서 IAM을 선택합니다.
2. IAM 콘솔에 들어가면 탐색 창에서 사용자를 선택합니다.
3. 사용자 이름을 선택합니다.
4. Add permissions(권한 추가)를 클릭합니다.
5. 정책 직접 연결을 선택합니다.
6. 검색 창에 `AmazonDocDBElasticFullAccess`을 입력하고 검색 결과에 표시되면 선택합니다.

7. 다음을 클릭합니다.
8. Add permissions(권한 추가)를 클릭합니다.

> **Note**
>
> AWS 계정에는 각 리전에 기본 VPC가 포함되어 있습니다. Amazon VPC를 사용하기로 선택한 경우 Amazon VPC 사용 설명서의 Amazon VPC 생성 주제의 단계를 완료합니다.

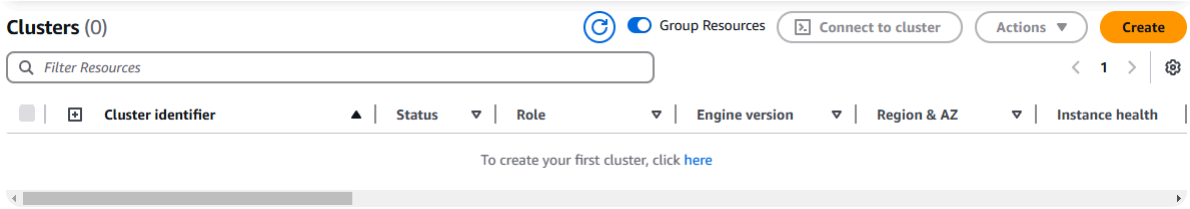
1단계: 엘라스틱 클러스터 생성

이 섹션에서는 AWS CLI 다음 지침에 따라 AWS Management Console 또는를 사용하여 새로운 탄력적 클러스터를 생성하는 방법을 설명합니다.

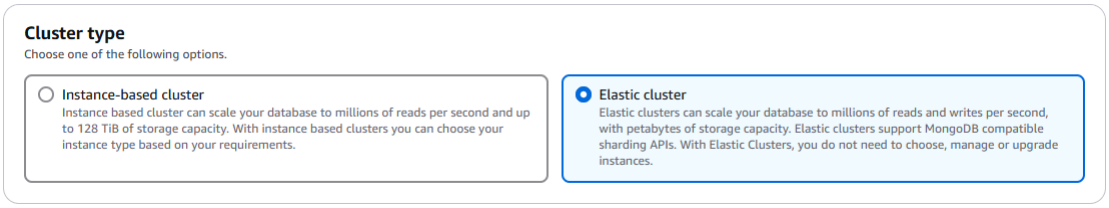
Using the AWS Management Console

AWS Management Console을 사용하여 엘라스틱 클러스터 구성을 생성하려면:

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. Amazon DocumentDB 관리 콘솔의 클러스터에서 생성을 선택합니다.



3. Amazon DocumentDB 클러스터 생성 페이지의 클러스터 유형 섹션에서 탄력적 클러스터를 선택합니다.



4. 구성 섹션에서 다음을 구성합니다.

   a. 클러스터 이름 필드에 고유한 클러스터 식별자를 입력합니다(필드 아래의 이름 지정 요구 사항 준수).

b. 샤드 수 필드에서, 클러스터에 포함하려는 샤드 수를 입력합니다. 클러스터당 최대 샤드 수는 32개입니다.

> **Note**
>
> 각 샤드마다 두 개의 노드가 배포됩니다. 두 노드의 샤드 용량은 동일합니다.

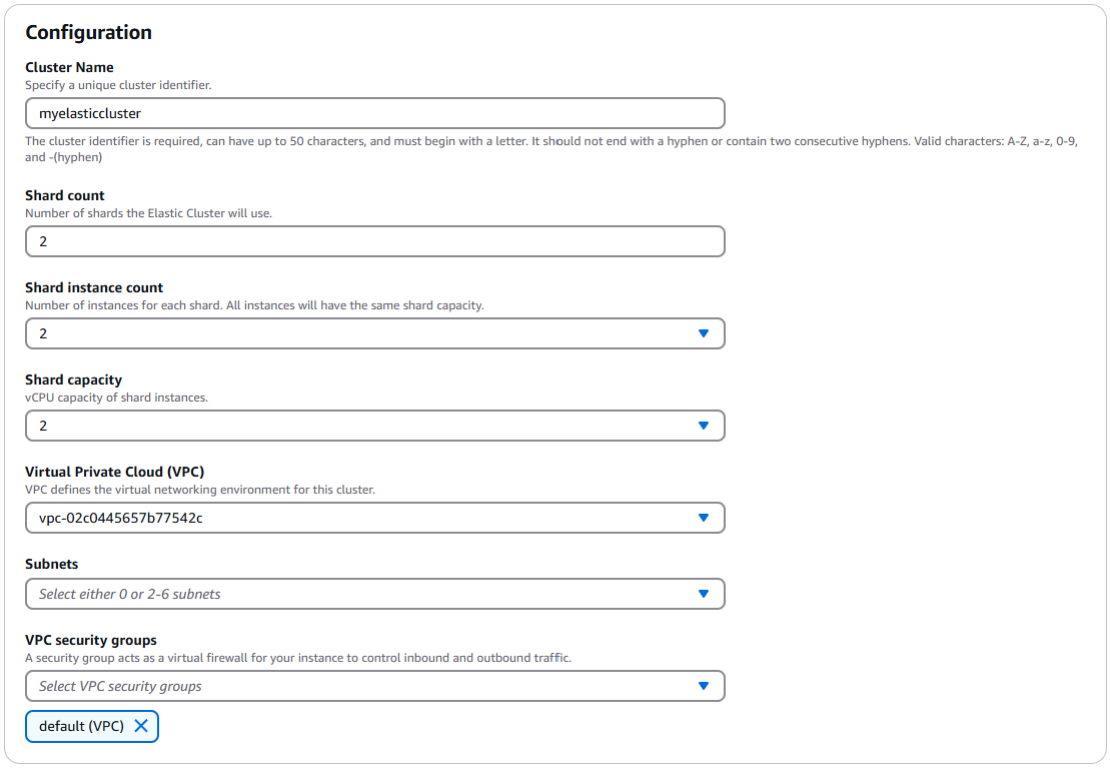
c. 샤드 인스턴스 수 필드에서 각 샤드와 연결할 복제본 인스턴스 수를 선택합니다. 최대 샤드 인스턴스 수는 16개이며 1씩 증가합니다. 모든 복제본 인스턴스는 다음 필드에 정의된 것과 동일한 샤드 용량을 갖습니다. 테스트를 위해 기본값인 2면 충분합니다.

> **Note**
>
> 복제본 인스턴스의 수는 탄력적 클러스터의 모든 샤드에 적용됩니다. 샤드 인스턴스 수의 값이 1이면 쓰기 인스턴스가 하나라는 의미이며, 추가 인스턴스는 읽기에 사용할 수 있고 가용성을 개선하는 복제본입니다. 테스트를 위해 기본값인 2면 충분합니다.

d. 샤드 용량 필드에서 각 샤드 인스턴스에 연결할 가상 CPU(vCPU) 수를 선택합니다. 샤드 인스턴스당 최대 vCPU 수는 64개입니다. 허용되는 값은 2, 4, 8, 16, 32, 64입니다. 테스트를 위해 기본값인 2면 충분합니다.

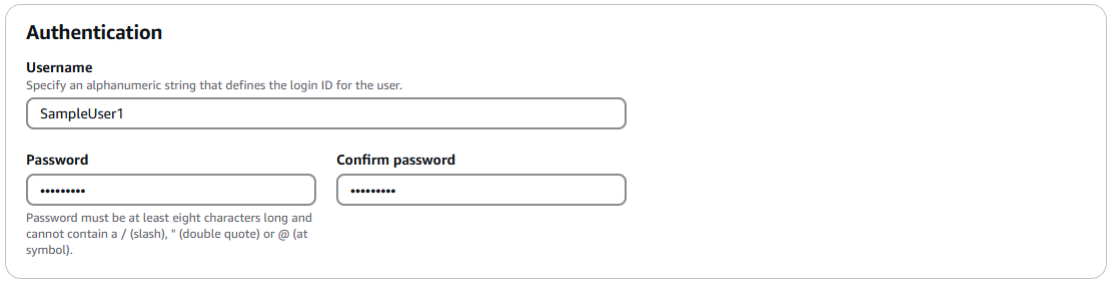
e. Virtual Private Cloud(VPC) 필드의 드롭다운 목록에서 VPC를 선택합니다.

f. 서브넷 및 VPC 보안 그룹의 경우, 기본값을 사용하거나 원하는 서브넷 3개와 VPC 보안 그룹 최대 3개(최소 1개)를 선택할 수 있습니다.



5. 인증 섹션에서 사용자 이름 필드에 기본 사용자의 로그인 이름을 식별하는 문자열을 입력합니다.

   암호 필드에 지침을 준수하는 고유한 암호를 입력한 다음 확인합니다.



6. 암호화 섹션에서 기본 설정(기본 키)을 유지합니다.

   선택적으로 생성한 AWS KMS key ARN을 입력할 수 있습니다. 자세한 내용은 Amazon DocumentDB 탄력적 클러스터에 대한 저장 데이터 암호화 단원을 참조하십시오.

> **Important**
>
> 엘라스틱 클러스터에 대해 암호화를 활성화해야 합니다.

7. 백업 섹션에서 백업 요구 사항에 따라 필드를 편집합니다. 테스트 목적으로 기본 설정을 유지할 수 있습니다.



   a. 백업 보존 기간 - 목록에서 이 클러스터의 자동 백업을 삭제하지 않고 유지하는 일 수를 선택합니다.

   b. 백업 기간 - Amazon DocumentDB에서 이 클러스터를 백업하는 일별 시간과 기간을 설정합니다.

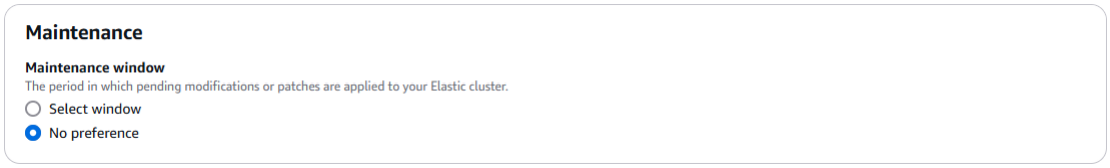
      i. 백업이 생성되는 시간과 기간을 구성하려면 창 선택을 선택합니다.

         시작 시간 - 첫 번째 목록에서 자동 백업 시작 시간(UTC)을 선택합니다. 두 번째 목록에서 자동 백업 시작 시간의 분을 선택합니다.

         기간 - 목록에서 자동 백업 생성에 할당된 시간 수를 선택합니다.

      ii. Amazon DocumentDB가 백업이 생성되는 시간과 기간을 선택하도록 하려면 기본 설정 없음을 선택합니다.

8. 유지 관리 섹션에서 클러스터에 수정 사항 또는 패치를 적용할 날짜, 시간, 기간을 선택합니다. 테스트 목적으로 기본 설정을 유지할 수 있습니다.



9. 클러스터 생성을 선택합니다.

이제 엘라스틱 클러스터를 프로비저닝하는 중입니다. 이 과정은 완료하는 데 최대 수분 소요될 수 있습니다. 탄력적 클러스터 상태가 클러스터 목록에 사용 가능으로 표시되면 클러스터에 연결할 수 있습니다.

Using the AWS CLI

를 사용하여 탄력적 클러스터를 생성하려면 다음 파라미터와 함께 `create-cluster` 작업을 AWS CLI사용합니다.

- --cluster-name - 필수입니다. 생성시 입력했거나 마지막으로 수정한 엘라스틱 스케일 클러스터의 현재 이름.
- --shard-capacity - 필수입니다. 각 샤드에 할당된 vCPU 개수입니다. 최대값은 64입니다. 허용되는 값은 2, 4, 8, 16, 32, 64입니다.
- --shard-count - 필수입니다. 클러스터에 할당된 샤드 개수입니다. 최대 값은 32입니다.
- --shard-instance-count - 선택 사항. 이 클러스터의 모든 샤드에 적용되는 복제본 인스턴스 수입니다. 최대 값은 16입니다.
- --admin-user-name - 필수입니다. 관리자 사용자와 연결된 사용자 이름입니다.
- --admin-user-password - 필수입니다. 관리자와 연결된 암호입니다.
- --auth-type - 필수입니다. 엘라스틱 클러스터에 액세스하는 데 사용되는 암호를 가져올 위치를 결정하는 데 사용되는 인증 유형입니다. 유효한 형식은 `PLAIN_TEXT` 또는 `SECRET_ARN`입니다.
- --vpc-security-group-ids - 선택 사항. 이 클러스터와 연결할 EC2 VPC 보안 그룹 목록을 구성합니다.
- --preferred-maintenance-window - 선택 사항. 시스템 유지보수가 수행될 수 있는 주간 시간 범위를 UTC(Universal Coordinated Time)로 구성합니다.

  형식은 `ddd:hh24:mi-ddd:hh24:mi`입니다. 유효한 요일(ddd): 월, 화, 수, 목, 금, 토, 일

  기본값은 각 Amazon Web Services Region(아마존 웹 서비스 리전)의 8시간 블록에서 임의로 선택된 30분 창으로, 주 마다 임의의 요일에 발생합니다.

  최소 30분의 시간.
- --kms-key-id - 선택 사항. 암호화된 클러스터의 KMS 키 식별자를 구성하세요.

  KMS 키 식별자는 AWS KMS 암호화 키의 Amazon 리소스 이름(ARN)입니다. 새 클러스터를 암호화하는 데 사용되는 KMS 암호화 키를 소유한 동일한 Amazon Web Services 계정을 사용하여 클러스터를 생성하는 경우, KMS 암호화 키에 대한 ARN 대신 KMS 키 별칭을 사용할 수 있습니다.

  암호화 키가 KmsKeyId에 지정되어 있지 않고 `StorageEncrypted` 파라미터가 참이면 Amazon DocumentDB는 기본 암호화 키를 사용합니다.
- --preferred-backup-window - 선택 사항. 자동 백업이 생성되는 일일 선호 시간 범위입니다. 기본값은 각각에 대해 8시간 블록 시간 중에서 임의로 선택한 30분 기간입니다 AWS 리전.
- --backup-retention-period - 선택 사항. 자동 백업이 보관되는 일수입니다. 기본값은 1입니다.
- --storage-encrypted - 선택 사항. 클러스터의 암호화 여부를 구성합니다.

`--no-storage-encrypted`은 클러스터가 암호화되지 않도록 지정합니다.

- --subnet-ids - 선택 사항. 네트워크 서브넷 ID를 구성합니다.

다음 예제에서는 자신의 정보로 각각의 *### ## ## ###*를 바꿉니다.

> **Note**
>
> 다음 예에는 특정 KMS 키 생성이 포함됩니다. 기본 KMS 키를 사용하려면 `--kms-key-id` 파라미터를 포함하지 마세요.

Linux, macOS 또는 Unix의 경우:

```
aws docdb-elastic create-cluster \
     --cluster-name sample-cluster-123 \
     --shard-capacity 8 \
     --shard-count 4 \
     --shard-instance-count 3 \
     --auth-type PLAIN_TEXT \
     --admin-user-name testadmin \
     --admin-user-password testPassword \
     --vpc-security-group-ids ec-65f40350 \
     --kms-key-id arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2 \
     --subnet-ids subnet-9253c6a3, subnet-9f1b5af9 \
     --preferred-backup-window 18:00-18:30 \
     --backup-retention-period 7
```

Windows의 경우:

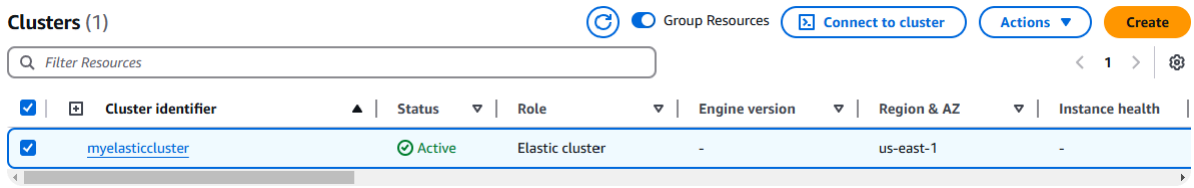
```
aws docdb-elastic create-cluster ^
     --cluster-name sample-cluster-123 ^
     --shard-capacity 8 ^
     --shard-count 4 ^
     --shard-instance-count 3 ^
     --auth-type PLAIN_TEXT ^
     --admin-user-name testadmin ^
     --admin-user-password testPassword ^
     --vpc-security-group-ids ec-65f40350 ^
```

```
    --kms-key-id arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2 ^
    --subnet-ids subnet-9253c6a3, subnet-9f1b5af9 \
    --preferred-backup-window 18:00-18:30 \
    --backup-retention-period 7
```

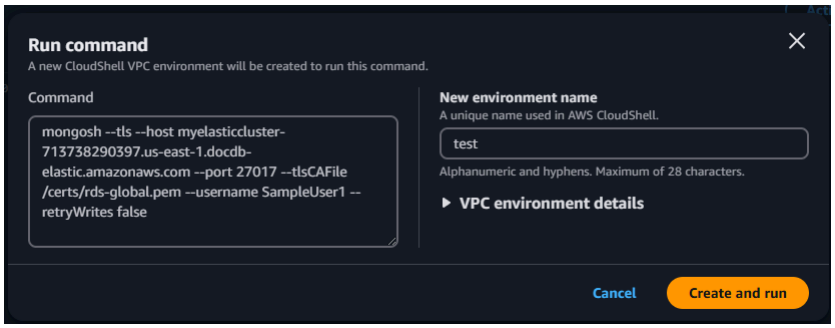
2단계: 엘라스틱 클러스터에 연결

를 사용하여 Amazon DocumentDB 탄력적 클러스터에 연결합니다 AWS CloudShell.

1. Amazon DocumentDB 관리 콘솔의 클러스터에서 생성한 탄력적 클러스터를 찾습니다. 클러스터 옆의 확인란을 클릭하여 클러스터를 선택합니다.



2. 클러스터에 연결을 클릭합니다(작업 드롭다운 메뉴 옆에 있음. 이 버튼은 클러스터 옆의 확인란을 클릭한 후에만 활성화되며 클러스터의 상태는 사용 가능으로 표시됩니다. CloudShell 명령 실행 화면이 나타납니다.
3. 새 환경 이름 필드에 "테스트"와 같은 고유한 이름을 입력하고 생성 및 실행을 클릭합니다. VPC 환경 세부 정보는 Amazon DocumentDB 데이터베이스에 대해 자동으로 구성됩니다.



4. 메시지가 표시되면 1단계: Amazon DocumentDB 탄력적 클러스터 생성(하위 5단계)에서 생성한 암호를 입력합니다.



암호를 입력하고 프롬프트`direct: mongos] <env-name>>`가 되면 Amazon DocumentDB 클러스터에 성공적으로 연결된 것입니다.

> **Note**
>
> 흐름 문제 해결에 대한 자세한 내용은 Amazon DocumentDB 문제 해결을 참조하십시오.

3단계: 수집 샤딩, 데이터 삽입 및 쿼리

엘라스틱 클러스터는 Amazon DocumentDB에서 샤딩에 대한 지원을 추가합니다. 이제 클러스터에 연결되었으므로 클러스터를 샤딩, 데이터를 삽입하고 몇 가지 쿼리를 실행할 수 있습니다.

1. 컬렉션을 샤드하려면 다음을 입력합니다.

```
sh.shardCollection("db.Employee1" , { "Employeeid" : "hashed" })
```

2. 단일 문서를 삽입하려면 다음을 입력합니다:

```
db.Employee1.insertOne({"Employeeid":1, "Name":"Joe", "LastName": "Bruin", "level":
 1 })
```

   다음 결과가 표시됩니다:

```
WriteResult({ "nInserted" : 1 })
```

3. 작성한 문서를 읽으려면 `findOne()` 명령을 입력합니다(단일 문서만 반환함).

```
db.Employee1.findOne()
```

다음 결과가 표시됩니다:

```
{
"_id" : ObjectId("61f344e0594fe1a1685a8151"),
"EmployeeID" : 1,
"Name" : "Joe",
"LastName" : "Bruin",
"level" : 1
}
```

4. 쿼리를 몇 개 더 수행하려면 게임 프로필 사용 사례를 고려하세요. 먼저 "직원"이라는 제목이 붙은 컬렉션에 몇 개의 항목을 삽입합니다. 다음을 입력합니다.

```
db.profiles.insertMany([ { "_id": 1, "name": "Matt", "status": "active", "level":
 12, "score": 202 },
     { "_id": 2, "name": "Frank", "status": "inactive", "level": 2, "score": 9 },
     { "_id": 3, "name": "Karen", "status": "active", "level": 7, "score": 87 },
     { "_id": 4, "name": "Katie", "status": "active", "level": 3, "score": 27 }
])
```

다음 결과가 표시됩니다:

```
{ acknowledged: true,
     insertedIds: {
        '0': ObjectId('679d02cd6b5a0581be78bcbd'),
        '1': ObjectId('679d02cd6b5a0581be78bcbe'),
        '2': ObjectId('679d02cd6b5a0581be78bcbf'),
        '3': ObjectId('679d02cd6b5a0581be78bcc0')
    }
}
```

5. find() 명령을 사용하여 프로필 컬렉션의 모든 문서를 반환합니다.

```
db.Employee.find()
```

4단계에서 입력한 데이터가 표시됩니다.

6. 단일 문서를 쿼리하려면 필터(예: "Katie")를 포함하세요. 다음을 입력합니다.

```
db.Employee.find({name: "Katie"})
```

다음 결과가 표시됩니다:

```
[

   {

     _id: ObjectId('679d02cd6b5a0581be78bcc0'),

     Employeeid: 4,

     name: 'Katie',

     lastname: 'Schaper',

     level: 3

   }

]
```

7. 프로필을 찾아 수정하려면 `findAndModify` 명령을 입력합니다. 이 예에서는 직원 "Matt"에게 더 높은 레벨인 "14"가 지정되었습니다.

```
db.Employee.findAndModify({
    query: { "Employeeid" : 1, "name" : "Matt"},
    update: { "Employeeid" : 1, "name" : "Matt", "lastname" : "Winkle", "level" :
 14 }
})
```

다음 결과가 표시됩니다(레벨은 아직 변경되지 않음).

```
{

   _id: ObjectId('679d02cd6b5a0581be78bcbd'),

   Employeeid: 1,

   name: 'Matt',

   lastname: 'Winkle',
```

```
   level: 12

}
```

8. 레벨 증가를 확인하려면 다음 쿼리를 입력합니다:

```
db.Employee.find({name: "Matt"})
```

다음 결과가 표시됩니다:

```
[
   {

   _id: ObjectId('679d02cd6b5a0581be78bcbd'),

   Employeeid: 1,

   name: 'Matt',

   lastname: 'Winkle',

   level: 14

   }
]
```

4단계: 탐색

축하합니다! Amazon DocumentDB 탄력적 클러스터에 대한 시작하기 절차를 성공적으로 완료했습니다.

다음 단계? 몇 가지 인기 있는 기능을 통해 이 데이터베이스를 최대한 활용하는 방법을 알아보십시오:

- Amazon DocumentDB 탄력적 클러스터 모범 사례
- Amazon DocumentDB 탄력적 클러스터 관리

> **Note**
>
> 이 시작 절차에서 생성한 탄력적 클러스터는 삭제하지 않는 한 비용이 계속 발생합니다. 지침은 엘라스틱 클러스터 삭제 섹션을 참조하세요.

Amazon DocumentDB 탄력적 클러스터 모범 사례

Amazon DocumentDB 탄력적 클러스터 작업 모범 사례에 대해서 알아봅니다. 모든 인스턴스 기반 Amazon DocumentDB 클러스터 모범 사례는 탄력적 클러스터에도 적용됩니다. 이 섹션은 새로운 모범 사례가 확인되는 대로 지속적으로 업데이트됩니다.

주제

- 샤드 키 선택
- 연결 관리
- 샤딩되지 않은 컬렉션
- 탄력적 클러스터 크기 조정
- 탄력적 클러스터 모니터링

샤드 키 선택

다음 목록은 분할 키를 만들기 위한 지침을 설명합니다.

- 균등하게 분산된 해시 키를 사용하여 클러스터의 모든 샤드에 데이터를 분산합니다(핫키는 제외).
- 분산형 수집 쿼리를 방지하려면 모든 읽기/업데이트/삭제 요청에 샤드 키를 사용하세요.
- 읽기/업데이트/삭제 작업을 수행할 때는 분할된 키가 중첩되지 않도록 하세요.
- 일괄 작업을 수행할 때 모든 샤드가 병렬로 실행되고 지연 시간을 개선할 수 있도록 `ordered`를 false로 설정하세요.

연결 관리

다음 목록은 데이터베이스 연결 관리에 대한 지침을 설명합니다.

- 연결 수와 새 연결이 열리고 닫히는 빈도를 모니터링하세요.

- 애플리케이션 구성의 모든 서브넷에 연결을 분산하세요. 클러스터가 여러 서브넷으로 구성되어 있지만 서브넷의 일부만 사용하는 경우 최대 연결 수에 병목 현상이 발생할 수 있습니다.

샤딩되지 않은 컬렉션

다음은 샤딩되지 않은 컬렉션에 대한 가이드라인을 설명합니다.

- 샤딩되지 않은 컬렉션을 사용할 때는 부하를 분산하기 위해 활용도가 높은 샤딩되지 않은 컬렉션을 여러 데이터베이스에 보관해 보세요. Amazon DocumentDB 탄력적 클러스터는 데이터베이스를 여러 샤드에 배치하고 샤딩되지 않은 동일한 데이터베이스에 대한 샤딩되지 않은 컬렉션을 동일한 샤드에 배치합니다.

탄력적 클러스터 크기 조정

다음 목록은 탄력적 클러스터를 규모 조정하기 위한 지침을 설명합니다.

- 조정 작업으로 인해 잠시 동안 데이터베이스 및 네트워크 오류가 간헐적으로 발생할 수 있습니다. 가능하면 사용량이 많은 시간대에는 규모를 조정하지 마세요. 유지 관리 기간 중에 크기 조정을 시도해 보세요.
- 컴퓨팅 성능을 높이기 위해 샤드 용량을 늘리거나 줄이는 것(샤드당 vCPU 수 변경)이 샤드 수를 늘리거나 줄이는 것보다 더 빠르고 간헐적으로 발생하는 데이터베이스 및 네트워크 오류의 지속 시간이 짧기 때문에 선호됩니다.
- 용량 증가를 예상할 때는 샤드 용량을 규모 조정하는 대신 샤드 수를 늘리는 것이 좋습니다. 이를 통해 빠르게 규모 조정해야 하는 시나리오에서 샤드 용량을 늘려 클러스터를 규모 조정할 수 있습니다.
- 클라이언트측 재시도 정책을 모니터링하고 지수 백오프 및 지터로 재시도하여 규모 조정 중에 오류가 발생할 경우 데이터베이스 과부하를 방지하세요.

탄력적 클러스터 모니터링

다음 목록은 탄력적 클러스터를 모니터링하기 위한 지침을 설명합니다.

- 샤드당 지표의 피크 대 평균 비율을 추적하여 고르지 않은 트래픽(핫키/핫스팟이 있음)을 유도하고 있는지 확인하세요. 피크 대 평균 비율을 추적하는 주요 지표는 다음과 같습니다.
  - `PrimaryInstanceCPUUtilization`
    - 이는 샤드별 수준에서 모니터링할 수 있습니다.
    - 클러스터 수준에서 평균 p99 스큐를 모니터링할 수 있습니다.

- `PrimaryInstanceFreeableMemory`
  - 이는 샤드별 수준에서 모니터링할 수 있습니다.
  - 클러스터 수준에서 평균 p99 스큐를 모니터링할 수 있습니다.
- `DatabaseCursorsMax`
  - 샤드별 수준에서 모니터링하여 스큐를 확인해야 합니다.
- `Documents-Inserted/Updated/Returned/Deleted`
  - 샤드별 수준에서 모니터링하여 스큐를 확인해야 합니다.

Amazon DocumentDB 탄력적 클러스터 관리

Amazon DocumentDB 엘라스틱 클러스터를 관리하려면 적절한 Amazon DocumentDB 컨트롤 플레인 권한이 있는 IAM 정책이 있어야 합니다. 이러한 권한은 클러스터를 생성, 수정 및 삭제하도록 허용합니다. Amazon DocumentDBFullAccess 정책은 Amazon DocumentDB 엘라스틱 클러스터를 관리하는 데 필요한 모든 권한을 제공합니다.

다음 항목에서는 Amazon DocumentDB 엘라스틱 클러스터로 작업할 때 다양한 작업을 수행하는 방법을 보여줍니다.

주제

- 엘라스틱 클러스터 구성 수정
- 엘라스틱 클러스터 모니터링
- 엘라스틱 클러스터 삭제
- 엘라스틱 클러스터 스냅샷 관리
- Amazon DocumentDB 탄력적 클러스터 중지 및 시작
- Amazon DocumentDB 탄력적 클러스터 유지 관리

엘라스틱 클러스터 구성 수정

이 섹션에서는 다음 지침과 AWS CLI 함께 AWS Management Console 또는를 사용하여 탄력적 클러스터를 수정하는 방법을 설명합니다.

클러스터를 수정하는 주된 용도는 샤드 카운트 및/또는 샤드 컴퓨팅 용량을 늘리거나 줄임으로써 샤드를 확장하는 것입니다.

Using the AWS Management Console

를 사용하여 탄력적 클러스터 구성을 수정하려면 AWS Management Console:

1. AWS Management Console에 로그인하고 Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 클러스터를 선택합니다.

   > **Tip**
   >
   > 화면 왼쪽에 탐색 창이 보이지 않으면 탐색 창의 왼쪽 상단 모서리에 있는 메뉴 아이콘을 선택합니다.

3. 클러스터 식별자 열에서 수정할 클러스터의 이름을 선택합니다.
4. 수정을 선택합니다.
5. 변경하려는 필드를 편집한 다음 클러스터 수정을 선택합니다.

Configuration

Cluster identifier

SampleCluster

Shard count

Number of shards the Elastic Cluster will use.

2

Shard instance count

Number of instances for each shard. All instances will have the same shard capacity.

2

Shard capacity

vCPU capacity of each shard.

2

Maintenance

Maintenance window

The period in which pending modifications or patches are applied to your Elastic cluster.

- ○ Select window
- ◉ No preference

Authentication

Username

SampleUser

New password

Confirm new password

Password must be at least eight characters long and cannot contain a / (slash), " (double quote) or @ (at symbol).

Network settings

Subnets

Select either 0 or 2-6 subnets

subnet-0b2962f92a0f5a8fb ✕ subnet-08c6d849efd4dfe96 ✕

VPC security groups

> **Note**
>
> 또는 클러스터 페이지로 이동하여 클러스터 옆의 확인란을 선택한 다음 작업, 수정을 선택하여 클러스터 수정 대화 상자에 액세스할 수 있습니다.

Using the AWS CLI

를 사용하여 탄력적 클러스터 구성을 수정하려면 다음 파라미터와 함께 `update-cluster` 작업을 AWS CLI사용합니다.

- **`--cluster-arn`** - 필수입니다. 수정하려는 클러스터의 ARN 식별자입니다.
- **`--shard-capacity`** - 선택 사항. 각 샤드에 할당된 vCPU 개수입니다. 최대값은 64입니다. 허용되는 값은 2, 4, 8, 16, 32, 64입니다.
- **`--shard-count`** - 선택 사항. 클러스터에 할당된 샤드 개수입니다. 최대 값은 32입니다.
- **`--shard-instance`**-수 - 선택 사항입니다. 이 클러스터의 모든 샤드에 적용되는 복제본 인스턴스 수입니다. 최대 값은 16입니다.
- **`--auth-type`** - 선택 사항. 엘라스틱 클러스터에 액세스하는 데 사용되는 암호를 가져올 위치를 결정하는 데 사용되는 인증 유형입니다. 유효한 형식은 `PLAIN_TEXT` 또는 `SECRET_ARN`입니다.
- **`--admin-user-password`** - 선택 사항. 관리자 사용자와 연결된 암호입니다.
- **`--vpc-security-group-ids`** - 선택 사항. 클러스터와 연결할 Amazon EC2 및 Amazon Virtual Private Cloud(Amazon VPC) 보안 그룹의 목록을 구성합니다.
- **`--preferred-maintenance-window`** - 선택 사항. 시스템 유지보수가 수행될 수 있는 주간 시간 범위를 UTC(Universal Coordinated Time)로 구성합니다

  형식은 `ddd:hh24:mi-ddd:hh24:mi`입니다. 유효한 요일(ddd): 월, 화, 수, 목, 금, 토, 일

  기본값은 각 Amazon Web Services Region(아마존 웹 서비스 리전)의 8시간 블록에서 임의로 선택된 30분 창으로, 주 마다 임의의 요일에 발생합니다.

  최소 30분의 시간.
- **`--subnet-ids`** - 선택 사항. 네트워크 서브넷 ID를 구성합니다.

다음 예제에서는 자신의 정보로 각각의 *`### ## ## ###`*를 바꿉니다.

Linux, macOS 또는 Unix의 경우:

```
aws docdb-elastic update-cluster \
    --cluster-arn arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2 \
    --shard-capacity 8 \
    --shard-count 4 \
    --shard-instance-count 3 \
    --admin-user-password testPassword \
    --vpc-security-group-ids ec-65f40350 \
    --subnet-ids subnet-9253c6a3, subnet-9f1b5af9
```

Windows의 경우:

```
aws docdb-elastic update-cluster ^
    --cluster-arn arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2 ^
    --shard-capacity 8 ^
    --shard-count 4 ^
    --shard-instance-count 3 ^
    --admin-user-password testPassword ^
    --vpc-security-group-ids ec-65f40350 ^
    --subnet-ids subnet-9253c6a3, subnet-9f1b5af9
```

수정 후 엘라스틱 클러스터의 상태를 모니터링하려면 엘라스틱 클러스터 모니터링을 참조하세요.

엘라스틱 클러스터 모니터링

이 섹션에서는 다음 지침에 따라 AWS Management Console 또는를 사용하여 탄력 AWS CLI 적 클러스터를 모니터링하는 방법을 설명합니다.

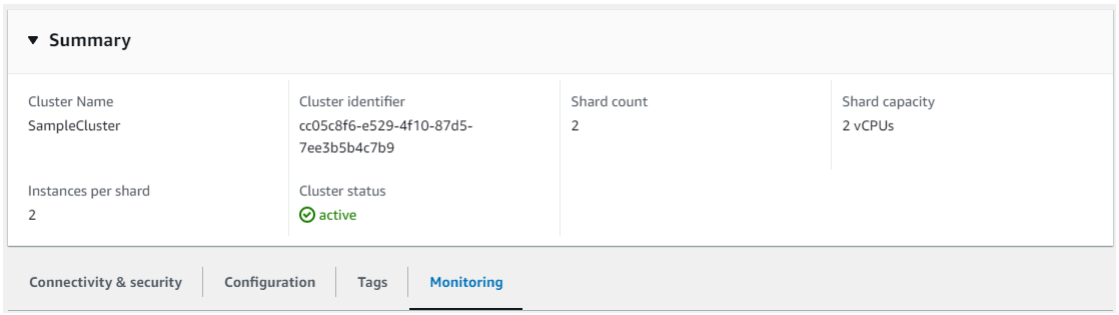
Using the AWS Management Console

를 사용하여 탄력적 클러스터 구성을 모니터링하려면 AWS Management Console:

1. AWS Management Console에 로그인하고 Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 클러스터를 선택합니다.

> **Tip**
>
> 화면 왼쪽에 탐색 창이 보이지 않으면 탐색 창의 왼쪽 상단 모서리에 있는 메뉴 아이콘을 선택합니다.

3. 클러스터 식별자 열에서 모니터링할 클러스터의 이름을 선택합니다.
4. 모니터링 탭을 선택합니다.



Amazon CloudWatch의 여러 차트가 다음과 같은 모니터링 카테고리에 대해 표시됩니다.

- 리소스 사용률
- 처리량
- 운영
- 시스템

를 통해 Amazon CloudWatch에 액세스 AWS Management Console 하여 탄력적 클러스터에 대한 자체 모니터링 환경을 설정할 수도 있습니다.

Using the AWS CLI

를 사용하여 특정 탄력적 클러스터 구성을 모니터링하려면 다음 파라미터와 함께 `get-cluster` 작업을 AWS CLI사용합니다.

- **`--cluster-arn`** - 필수입니다. 정보를 원하는 클러스터의 ARN 식별자입니다.

다음 예제에서는 자신의 정보로 각각의 *### ## ## ###*를 바꿉니다.

Linux, macOS 또는 Unix의 경우:

```
aws docdb-elastic get-cluster \
    --cluster-arn arn:aws:docdb-elastic:us-west-2:123456789012:cluster:/68ffcdf8-e3af-40a3-91e4-24736f2dacc9
```

Windows의 경우:

```
aws docdb-elastic get-cluster ^
    --cluster-arn arn:aws:docdb:-elastic:us-west-2:123456789012:cluster:/68ffcdf8-e3af-40a3-91e4-24736f2dacc9
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
"cluster": {
        ...
        "clusterArn": "arn:aws:docdb-elastic:us-west-2:123456789012:cluster:/68ffcdf8-e3af-40a3-91e4-24736f2dacc9",
        "clusterEndpoint": "stretch-11-477568257630.us-east-1.docdb-elastic.amazonaws.com",
        "readerEndpoint": "stretch-11-477568257630-ro.us-east-1.docdb-elastic.amazonaws.com",
        "clusterName": "stretch-11",
        "shardCapacity": 2,
        "shardCount": 3,
        "shardInstanceCount: 5,
        "status": "ACTIVE",
        ...
 }
```

자세한 내용은 Amazon DocumentDB 리소스 관리 API 참조에서 `DescribeClusterSnapshot` 섹션을 참조하세요.

를 사용하여 모든 탄력적 클러스터의 세부 정보를 보려면 다음 파라미터와 함께 `list-clusters` 작업을 AWS CLI사용합니다.

- **`--next-token`** - 선택 사항. 출력 항목 수(`--max-results`)가 기본 API 호출에서 반환하는 전체 항목 수보다 적을 경우 사용자가 다음 항목 세트를 검색하기 위해 후속 명령에 전달할 수 있도록 출력에 `NextToken`이 포함됩니다.
- **`--max-results`** - 선택 사항. 명령의 출력에서 반환되는 항목의 총 수입니다. 지정된 `max-results` 값보다 레코드 수가 많으면 페이지 매김 토큰(`next-token`)을 응답에 포함시켜 나머지 결과를 검색할 수 있도록 합니다.
  - 기본값: 100
  - 최소: 20 최대: 100

다음 예제에서는 자신의 정보로 각각의 *### ## ## ###*를 바꿉니다.

Linux, macOS 또는 Unix의 경우:

```
aws docdb-elastic list-clusters \
    --next-token eyJNYXJrZXIiOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAxfQ== \
    --max-results 2
```

Windows의 경우:

```
aws docdb-elastic list-clusters ^
    --next-token eyJNYXJrZXIiOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAxfQ== ^
    --max-results 2
```

이 작업의 출력은 다음과 같습니다.

```
{
   "Clusters": [
      {
         "ClusterIdentifier":"mycluster-1",
         "ClusterArn":"arn:aws:docdb:us-west-2:123456789012:sharded-cluster:sample-
cluster"
         "Status":"available",
         "ClusterEndpoint":"sample-cluster.sharded-cluster-corcjozrlsfc.us-
west-2.docdb.amazonaws.com"
       }
       {
         "ClusterIdentifier":"mycluster-2",
         "ClusterArn":"arn:aws:docdb:us-west-2:987654321098:sharded-cluster:sample-
cluster"
         "Status":"available",
         "ClusterEndpoint":"sample-cluster2.sharded-cluster-corcjozrlsfc.us-
west-2.docdb.amazonaws.com"
       }
   ]
}
```

엘라스틱 클러스터 삭제

이 섹션에서는 다음 지침과 AWS CLI 함께 AWS Management Console 또는를 사용하여 탄력적 클러스터를 삭제하는 방법을 설명합니다.

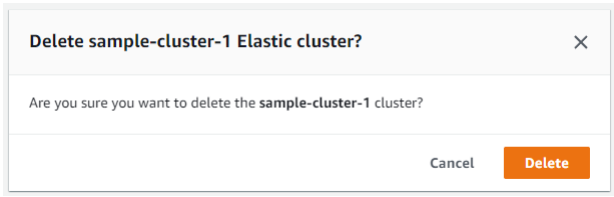
Using the AWS Management Console

AWS Management Console을 사용하여 엘라스틱 클러스터 구성을 삭제하려면:

1. AWS Management Console에 로그인하고 Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 클러스터를 선택합니다.

> **Tip**
>
> 화면 왼쪽에 탐색 창이 보이지 않으면 탐색 창의 왼쪽 상단 모서리에 있는 메뉴 아이콘을 선택합니다.

3. 클러스터 목록 테이블에서 삭제할 클러스터 이름의 왼쪽에 있는 확인란을 선택한 다음 작업을 선택합니다. 드롭다운 메뉴에서 삭제를 선택합니다.
4. "클러스터 이름" 엘라스틱 클러스터를 삭제하시겠습니까? 대화 상자에서 삭제를 선택합니다.



클러스터를 삭제하는 데 몇 분 정도 걸립니다. 클러스터 상태를 모니터링하려면 Amazon DocumentDB 클러스터 상태 모니터링을 참조하십시오.

Using the AWS CLI

를 사용하여 탄력적 클러스터를 삭제하려면 다음 파라미터와 함께 `delete-cluster` 작업을 AWS CLI사용합니다.

- **`--cluster-arn`** - 필수입니다. 삭제하려는 클러스터의 ARN 식별자입니다.
- **`--no-skip-final-backup`** - 선택 사항. 최종 백업을 원하는 경우, 최종 백업의 이름과 함께 이 파라미터를 포함해야 합니다. `--final-backup-identifier` 또는 `--skip-final-backup`를 포함시켜야 합니다.
- **`--skip-final-backup`** - 선택 사항. 클러스터를 삭제하기 전에 최종 백업을 수행하지 않을 경우에만 이 파라미터를 사용합니다. 기본 설정은 최종 스냅샷을 생성하는 것입니다.

다음 AWS CLI 코드 예제에서는 ARN이 arn:aws:docdb:us-west-2:123456789012:sharded-cluster:sample-cluster이고 최종 백업이 있는 클러스터를 삭제합니다.

다음 예제에서는 자신의 정보로 각각의 *### ## ## ###*를 바꿉니다.

Linux, macOS, Unix의 경우:

```
aws docdb-elastic delete-cluster \
    --cluster-arn arn:aws:docdb:us-west-2:123456789012:sharded-cluster:sample-
cluster \
    --no-skip-final-backup \
    --final-backup-identifier finalArnBU-arn:aws:docdb:us-
west-2:123456789012:sharded-cluster:sample-cluster
```

Windows의 경우:

```
aws docdb-elastic delete-cluster ^
    --cluster-arn arn:aws:docdb:us-west-2:123456789012:sharded-cluster:sample-
cluster ^
    --no-skip-final-backup ^
    --final-backup-identifier finalArnBU-arn:aws:docdb:us-
west-2:123456789012:sharded-cluster:sample-cluster
```

다음 AWS CLI 코드 예제에서는 최종 백업을 수행하지 않고 ARN이 arn:aws:docdb:us-west-2:123456789012:sharded-cluster:sample-cluster인 클러스터를 삭제합니다.

다음 예제에서는 자신의 정보로 각각의 *### ## ## ###*를 바꿉니다.

Linux, macOS 또는 Unix의 경우:

```
aws docdb-elastic delete-cluster \
    --cluster-arn arn:aws:docdb:us-west-2:123456789012:sharded-cluster:sample-
cluster \
    --skip-final-backup \
```

Windows의 경우:

```
aws docdb-elastic delete-cluster ^
    --cluster-arn arn:aws:docdb:us-west-2:123456789012:sharded-cluster:sample-
cluster ^
    --skip-final-backup ^
```

`delete-cluster` 작업의 출력은 삭제할 클러스터의 표시입니다.

클러스터를 삭제하는 데 몇 분 정도 걸립니다. 클러스터 상태를 모니터링하려면 Amazon DocumentDB 클러스터 상태 모니터링을 참조하십시오.

엘라스틱 클러스터 스냅샷 관리

수동 스냅샷은 엘라스틱 클러스터를 생성한 후에 만들 수 있습니다. 자동 백업은 탄력적 클러스터 스냅샷이 생성되는 즉시 생성됩니다.

> **Note**
>
> 수동 스냅샷을 생성하려면 엘라스틱 클러스터가 `Available` 상태에 있어야 합니다.

이 섹션에서는 엘라스틱 클러스터 스냅샷을 생성하고, 보고, 복원하고, 삭제하는 방법을 설명합니다.

다음 항목에서는 Amazon DocumentDB 엘라스틱 클러스터 스냅샷으로 작업할 때 다양한 작업을 수행하는 방법을 보여줍니다.

주제

- 수동 클러스터 스냅샷 생성
- 엘라스틱 클러스터 스냅샷 보기
- 스냅샷에서 엘라스틱 클러스터 복원
- 탄력적 클러스터 스냅샷 보기
- 탄성 클러스터 스냅샷 삭제
- 탄력적 클러스터 스냅샷 자동 백업 관리

수동 클러스터 스냅샷 생성

이 섹션에서는 다음 지침과 AWS CLI 함께 AWS Management Console 또는를 사용하여 수동 탄력적 클러스터 스냅샷을 생성하는 방법을 설명합니다.

Using the AWS Management Console

AWS Management Console을 사용하여 수동 엘라스틱 클러스터 스냅샷을 만들려면:

1. AWS Management Console에 로그인하고 Amazon DocumentDB 콘솔을 엽니다.

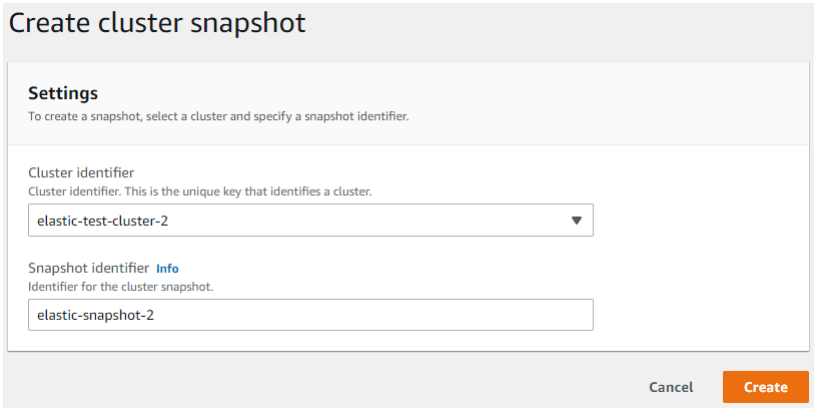
2. 탐색 창에서 스냅샷을 선택합니다.

> **Tip**
>
> 화면 왼쪽에 탐색 창이 보이지 않으면 탐색 창의 왼쪽 상단 모서리에 있는 메뉴 아이콘을 선택합니다.

3. 스냅샷 페이지에서 생성을 선택합니다.
4. 클러스터 스냅샷 생성 페이지의 클러스터 식별자 필드에 있는 드롭다운 목록에서 탄력적 클러스터를 선택합니다.

   스냅샷 식별자 필드에 엘라스틱 클러스터의 고유 식별자를 입력합니다.

   생성(Create)을 선택합니다.



> **Note**
>
> 또는 클러스터 페이지로 이동하여 클러스터 옆의 확인란을 선택한 다음 작업, 스냅샷 찍기를 선택하여 클러스터 스냅샷 생성 대화 상자에 액세스할 수 있습니다.

이제 엘라스틱 클러스터 스냅샷이 프로비저닝 중입니다. 이 과정은 완료하는 데 최대 수 분 소요될 수 있습니다. 스냅샷 목록에 상태가 `Available`로 표시될 때 스냅샷을 보고 복원할 수 있습니다.

Using the AWS CLI

를 사용하여 수동 탄력적 클러스터 스냅샷을 생성하려면 다음 파라미터와 함께 `create-cluster-snapshot` 작업을 AWS CLI사용합니다.

- **`--snapshot-name`** - 필수입니다. 생성할 클러스터 스냅샷의 이름입니다.

- **`--cluster-arn`** - 필수입니다. 스냅샷을 생성하고자 하는 클러스터의 ARN 식별자입니다.

다음 예제에서는 자신의 정보로 각각의 *### ## ## ###*를 바꿉니다.

Linux, macOS 또는 Unix의 경우:

```
aws docdb-elastic create-cluster-snapshot \
    --snapshot-name sample-snapshot-1 \
    --cluster-arn arn:aws:docdb:us-west-2:123456789012:sharded-cluster:sample-
cluster
```

Windows의 경우:

```
aws docdb-elastic create-cluster-snapshot ^
    --snapshot-name sample-snapshot-1 ^
    --cluster-arn arn:aws:docdb:us-west-2:123456789012:sharded-cluster:sample-
cluster
```

엘라스틱 클러스터 스냅샷 보기

이 섹션에서는 다음 지침과 AWS CLI 함께 AWS Management Console 또는를 사용하여 탄력적 클러스터 스냅샷 정보를 보는 방법을 설명합니다.

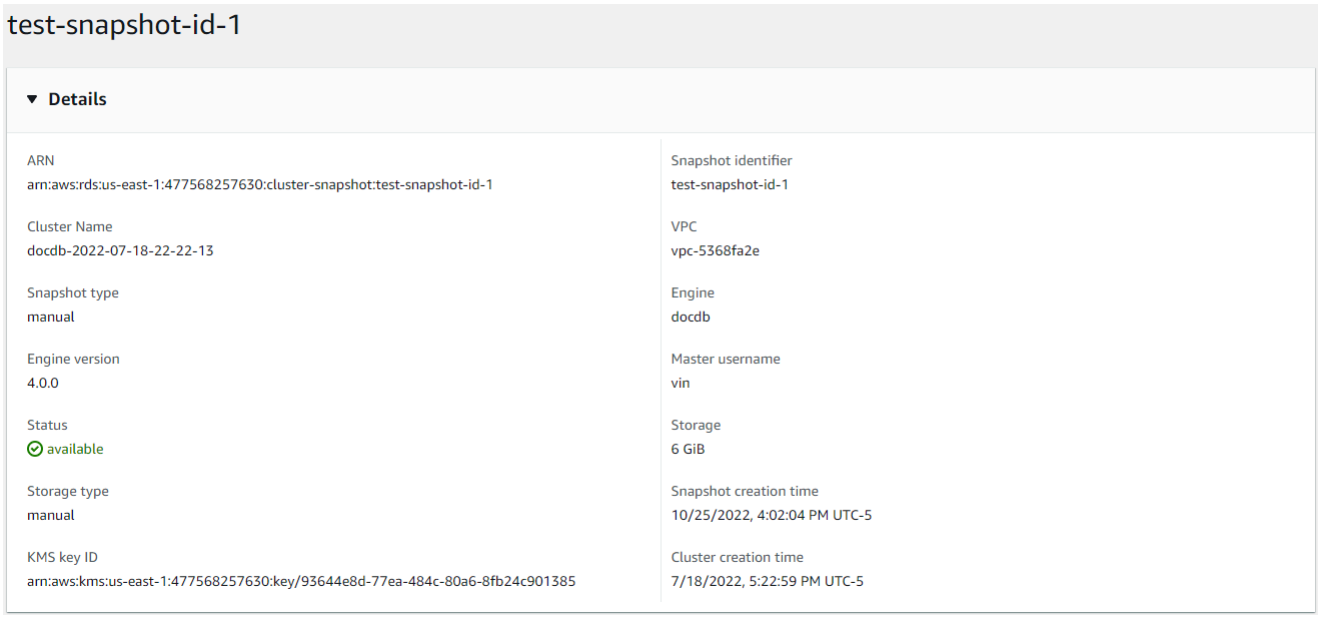
Using the AWS Management Console

를 사용하여 특정 탄력적 클러스터 스냅샷에 대한 정보를 보려면 AWS Management Console:

1. AWS Management Console에 로그인하고 Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 스냅샷을 선택합니다.

   > **Tip**
   >
   > 화면 왼쪽에 탐색 창이 보이지 않으면 탐색 창의 왼쪽 상단 모서리에 있는 메뉴 아이콘을 선택합니다.

3. 스냅샷 페이지에서 스냅샷 식별자 열에 있는 이름을 클릭하여 목록에서 스냅샷을 선택합니다.
4. 세부 정보에서 스냅샷 정보를 확인하세요.



Using the AWS CLI

를 사용하여 특정 탄력적 클러스터 스냅샷에 대한 정보를 보려면 다음 파라미터와 함께 `get-cluster-snapshot` 작업을 AWS CLI사용합니다.

- **`--snapshot-arn`** - 필수입니다. 정보를 원하는 스냅샷의 ARN 식별자입니다.

다음 예제에서는 자신의 정보로 각각의 *### ## ## ###*를 바꿉니다.

Linux, macOS 또는 Unix의 경우:

```
aws docdb-elastic get-cluster-snapshot \
    --snapshot-arn sampleResourceName
```

Windows의 경우:

```
aws docdb-elastic get-cluster-snapshot ^
    --snapshot-arn sampleResourceName
```

를 사용하여 특정 탄력적 클러스터 스냅샷에 대한 정보를 보려면 다음 파라미터와 함께 `get-cluster-snapshot` 작업을 AWS CLI사용합니다.

- **`--snapshot-arn`** - 필수입니다. 정보를 원하는 스냅샷의 ARN 식별자입니다.

다음 예제에서는 자신의 정보로 각각의 *### ## ## ###*를 바꿉니다.

Linux, macOS 또는 Unix의 경우:

```
aws docdb-elastic get-cluster-snapshot \
    --snapshot-arn sampleResourceName
```

Windows의 경우:

```
aws docdb-elastic get-cluster-snapshot ^
    --snapshot-arn sampleResourceName
```

를 사용하여 모든 탄력적 클러스터 스냅샷에 대한 정보를 보려면 다음 파라미터와 함께 `list-cluster-snapshots` 작업을 AWS CLI사용합니다.

- **`--snapshot-type`** - 선택 사항. 반환되는 클러스터 스냅샷의 유형입니다. 다음 값 중 하나를 지정할 수 있습니다.
  - `automated` - Amazon DocumentDB가 AWS 계정에 대해 자동으로 생성한 모든 클러스터 스냅샷을 반환합니다.
  - `manual` - AWS 계정에 대해 수동으로 생성한 모든 클러스터 스냅샷을 반환합니다.
  - `shared` - AWS 계정에 공유된 모든 수동 클러스터 스냅샷을 반환합니다.
  - `public` - 퍼블릭으로 표시된 모든 클러스터 스냅샷을 모두 반환합니다.
- **`--next-token`** - 선택 사항. 이전의 요청에서 제공된 선택적 페이지 매김 토큰입니다. 이 파라미터를 지정한 경우, 이 토큰 이후부터 `max-results`에 지정된 값까지의 레코드만 응답에 포함됩니다.
- **`--max-results`** - 선택 사항. 응답에 포함되는 최대 결과 수입니다. 지정된 `max-results` 값보다 레코드 수가 많으면 페이지 매김 토큰(`next-token`)을 응답에 포함시켜 나머지 결과를 검색할 수 있도록 합니다.
  - 기본값: 100
  - 최소: 20 최대: 100

다음 예제에서는 자신의 정보로 각각의 *### ## ## ###*를 바꿉니다.

Linux, macOS 또는 Unix의 경우:

```
aws docdb-elastic list-cluster-snapshots \
    --snapshot-type  value \
    --next-token  value \
    --max-results 50
```

Windows의 경우:

```
aws docdb-elastic list-cluster-snapshots ^
    --snapshot-type  value ^
    --next-token  value ^
    --max-results 50
```

스냅샷에서 엘라스틱 클러스터 복원

이 섹션에서는 다음 지침과 AWS CLI 함께 AWS Management Console 또는를 사용하여 스냅샷에서 탄력적 클러스터를 복원하는 방법을 설명합니다.

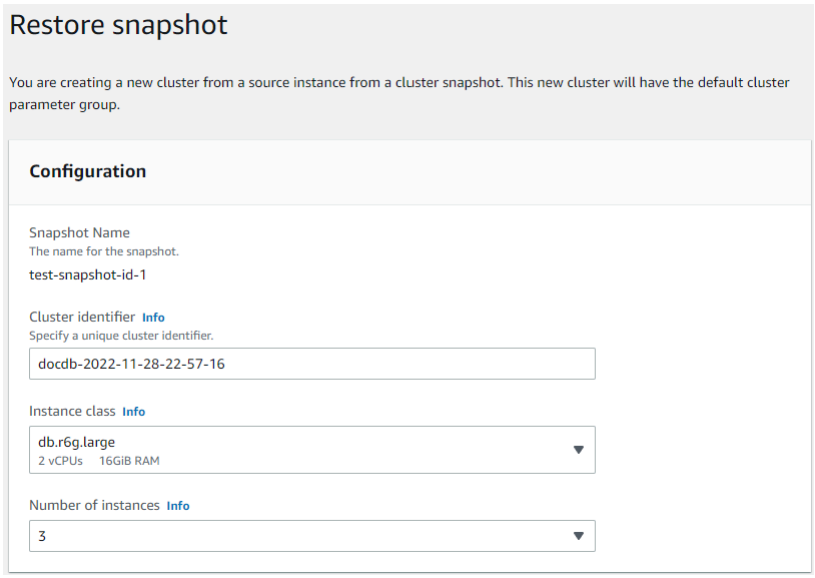
Using the AWS Management Console

AWS Management Console를 사용하여 스냅샷에서 엘라스틱 클러스터 복원:

1. AWS Management Console에 로그인하고 Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 스냅샷을 선택합니다.

   > **Tip**
   >
   > 화면 왼쪽에 탐색 창이 보이지 않으면 탐색 창의 왼쪽 상단 모서리에 있는 메뉴 아이콘을 선택합니다.

3. 스냅샷 식별자 열에서 클러스터 복원에 사용하려는 스냅샷 왼쪽에 있는 버튼을 선택합니다.
4. 작업을 선택한 다음, 복원을 선택합니다.



5. 스냅샷 복원 페이지의 클러스터 식별자 필드에 새 클러스터의 이름을 입력합니다.

> **Note**
>
> 수동 스냅샷 복원의 경우 새 클러스터를 생성해야 합니다.

6. 가상 프라이빗 클라우드(VPC) 필드에서 드롭다운 목록 중 VPC를 선택합니다.
7. 서브넷 및 VPC 보안 그룹의 경우 기본값을 사용하거나 원하는 서브넷 3개와 VPC 보안 그룹 최대 3개 (최소 1개) 를 선택할 수 있습니다.
8. 클러스터 구성이 마음에 들면 클러스터 복원을 선택하고 클러스터가 복원되는 동안 기다립니다.

Using the AWS CLI

를 사용하여 스냅샷에서 탄력적 클러스터를 복원하려면 다음 파라미터와 함께 `restore-cluster-from-snapshot` 작업을 AWS CLI사용합니다.

- **`--cluster-name`** - 필수입니다. 생성시 입력했거나 마지막으로 수정한 엘라스틱 클러스터의 현재 이름.
- **`--snapshot-arn`** - 필수입니다. 클러스터를 복원하는 데 사용되는 스냅샷의 ARN 식별자입니다.
- **`--vpc-security-group-ids`** - 선택 사항. 클러스터와 연결할 Amazon EC2 및 and Amazon Virtual Private Cloud(Amazon VPC) 보안 그룹의 목록을 구성합니다.
- **`--kms-key-id`** - 선택 사항. 암호화된 클러스터의 KMS 키 식별자를 구성하세요.

KMS 키 식별자는 AWS KMS 암호화 키의 Amazon 리소스 이름(ARN)입니다. 새 클러스터를 암호화하는 데 사용되는 KMS 암호화 키를 소유한 동일한 Amazon Web Services 계정을 사용하여 클러스터를 생성하는 경우, KMS 암호화 키에 대한 ARN 대신 KMS 키 별칭을 사용할 수 있습니다.

암호화 키가 KmsKeyId에 지정되어 있지 않고 `StorageEncrypted` 파라미터가 참이면 Amazon DocumentDB는 기본 암호화 키를 사용합니다.

- **`--subnet-ids`** - 선택 사항. 네트워크 서브넷 ID.

다음 예제에서는 자신의 정보로 각각의 *### ## ## ###*를 바꿉니다.

Linux, macOS, Unix의 경우:

```
aws docdb-elastic restore-cluster-from-snapshot \
     --cluster-name elastic-sample-cluster \
     --snapshot-arn sampleResourceName \
     --vpc-security-group-ids value ec-65f40350 \
     --kms-key-id arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2 \
     --subnet-ids subnet-9253c6a3, subnet-9f1b5af9
```

Windows의 경우:

```
aws docdb-elastic restore-cluster-from-snapshot ^
     --cluster-name elastic-sample-cluster ^
     --snapshot-arn sampleResourceName ^
     --vpc-security-group-ids value ec-65f40350 ^
     --kms-key-id arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2 ^
     --subnet-ids subnet-9253c6a3, subnet-9f1b5af9
```

탄력적 클러스터 스냅샷 보기

Amazon DocumentDB에서는 동일한 리전 및 동일한 계정 내의 수동 및 자동 탄력적 클러스터 스냅샷을 복사할 수 있습니다. 이 섹션에서는 AWS Management Console 또는를 사용하여 탄력적 클러스터 스냅샷을 복사하는 방법을 설명합니다 AWS CLI.

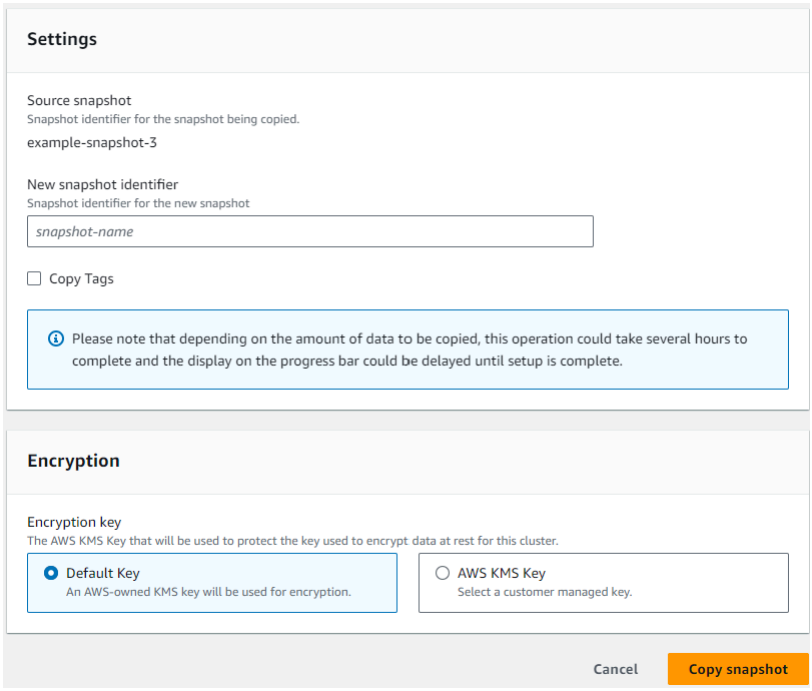
Using the AWS Management Console

를 사용하여 탄력적 클러스터 스냅샷을 복사하려면 AWS Management Console:

1. AWS Management Console에 로그인하고 Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 스냅샷을 선택합니다.

> **Tip**
>
> 화면 왼쪽에 탐색 창이 보이지 않으면 탐색 창의 왼쪽 상단 모서리에 있는 메뉴 아이콘을 선택합니다.

3. 스냅샷 식별자 열에서 복사하려는 스냅샷 왼쪽에 있는 버튼을 선택합니다.
4. 작업, 복사 순으로 선택합니다.



5. 새 스냅샷 식별자의 경우 새 스냅샷의 이름을 입력합니다.
6. 태그 복사에서 소스 탄력적 클러스터 스냅샷의 모든 태그를 대상 탄력적 클러스터 스냅샷으로 복사하려면 확인란을 선택합니다.
7. 암호화에서 기본 AWS KMS 키 또는 원하는 KMS 키를 선택합니다. 두 번째 옵션을 사용하면 이미 생성한 기존 KMS 키를 선택하거나 새 키를 생성할 수 있습니다.
8. 완료되면 스냅샷 복사를 선택합니다.

Using the AWS CLI

를 사용하여 탄력적 클러스터 스냅샷을 복사하려면 다음 파라미터와 함께 `copy-cluster-snapshot` 작업을 AWS CLI사용합니다.

- **`--source-db-cluster-snapshot-identifier`** - 필수입니다. 복사되는 기존 탄력적 클러스터 스냅샷의 식별자. 탄력적 클러스터 스냅샷이 있어야 하며 사용 가능한 상태여야 합니다. 이 파라미터는 대소문자를 구분하지 않습니다.
- **`--target-db-cluster-snapshot-identifier`** - 필수입니다. 기존의 탄력적 클러스터 스냅샷에서 생성할 새 탄력적 클러스터 스냅샷의 식별자입니다. 이 파라미터는 대소문자를 구분하지 않습니다.

  대상 스냅샷 명명 제약 조건:

  - 기본 스냅샷의 이름일 수 없습니다.
  - 길이는 [1~63]글자, 숫자 또는 하이픈입니다.
  - 첫 번째 문자는 글자이어야 합니다.
  - 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다.

다음 예제에서는 자신의 정보로 각각의 *### ## ## ###*를 바꿉니다.

Linux, macOS 또는 Unix의 경우:

```
aws docdb-elastic copy-cluster-snapshot \
     --source-cluster-snapshot-arn <sample ARN> \
     --target-cluster-snapshot-name my-target-copied-snapshot
```

Windows의 경우:

```
aws docdb-elastic copy-cluster-snapshot ^
     --source-cluster-snapshot-arn <sample ARN> ^
     --target-cluster-snapshot-name my-target-copied-snapshot
```

탄성 클러스터 스냅샷 삭제

이 섹션에서는 AWS Management Console 또는를 사용하여 탄력적 클러스터 스냅샷을 삭제하는 방법을 설명합니다 AWS CLI.

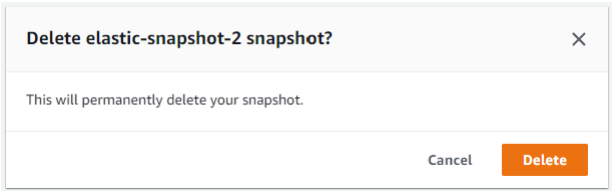
Using the AWS Management Console

AWS Management Console를 사용하여 스냅샷에서 엘라스틱 클러스터 복원:

1. AWS Management Console에 로그인하고 Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 스냅샷을 선택합니다.

> **Tip**
>
> 화면 왼쪽에 탐색 창이 보이지 않으면 탐색 창의 왼쪽 상단 모서리에 있는 메뉴 아이콘을 선택합니다.

3. 스냅샷 식별자 열에서 클러스터 복원에 사용하려는 스냅샷 왼쪽에 있는 버튼을 선택합니다.
4. 작업을 선택한 다음 삭제를 선택합니다.



5. "스냅샷 이름" 스냅샷 삭제 대화 상자에서 삭제를 선택합니다.

Using the AWS CLI

를 사용하여 탄력적 클러스터 스냅샷을 삭제하려면 다음 파라미터와 함께 `delete-cluster-snapshot` 작업을 AWS CLI사용합니다.

- **`--snapshot-arn`** - 필수입니다. 클러스터를 복원하는 데 사용되는 스냅샷의 ARN 식별자입니다.

다음 예제에서는 자신의 정보로 각각의 *`### ## ## ###`*를 바꿉니다.

Linux, macOS, Unix의 경우:

```
aws docdb-elastic delete-cluster-snapshot \
     --snapshot-arn sampleResourceName
```

Windows의 경우:

```
aws docdb-elastic delete-cluster-snapshot ^
```

```
    --snapshot-arn sampleResourceName
```

탄력적 클러스터 스냅샷 자동 백업 관리

Amazon DocumentDB는 탄력적 클러스터의 일일 스냅샷을 생성합니다. 새 탄력적 클러스터 또는 기존 탄력적 클러스터 스냅샷 구성에서 기본 백업 기간과 백업 보존 기간을 지정할 수 있습니다. 이 섹션에서는 AWS Management Console 또는를 사용하여 탄력적 클러스터 스냅샷에서 자동 백업 파라미터를 설정하는 방법을 설명합니다 AWS CLI.

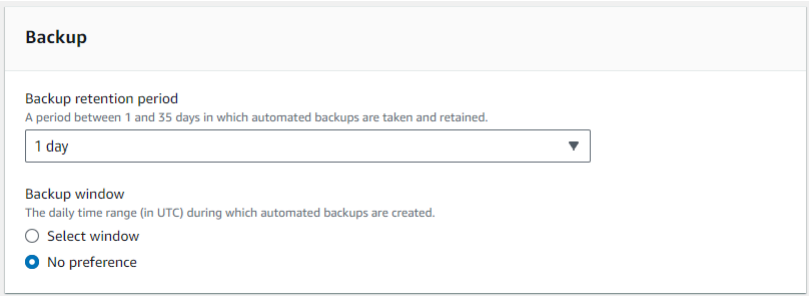
Using the AWS Management Console

를 사용하여 새 탄력적 클러스터 스냅샷에 대한 자동 백업을 설정하려면 AWS Management Console:

1. AWS Management Console에 로그인하고 Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 클러스터를 선택합니다.

> **Tip**
>
> 화면 왼쪽에 탐색 창이 보이지 않으면 탐색 창의 왼쪽 상단 모서리에 있는 메뉴 아이콘을 선택합니다.

3. 클러스터 식별자 열에서 백업 설정을 변경할 클러스터 왼쪽의 버튼을 선택합니다.
4. 작업(Actions), 수정(Modify)을 선택합니다.
5. 백업 섹션에서 백업 요구 사항에 따라 필드를 편집합니다.



   a. 백업 보존 기간 - 목록에서 이 클러스터의 자동 백업을 삭제하지 않고 유지하는 일 수를 선택합니다.

   b. 백업 기간 - Amazon DocumentDB에서 이 클러스터를 백업하는 일별 시간과 기간을 설정합니다.

i. 백업이 생성되는 시간과 기간을 구성하려면 창 선택을 선택합니다.

시작 시간 - 첫 번째 목록에서 자동 백업 시작 시간(UTC)을 선택합니다. 두 번째 목록에서 자동 백업 시작 시간의 분을 선택합니다.

기간 - 목록에서 자동 백업 생성에 할당된 시간 수를 선택합니다.

ii. Amazon DocumentDB가 백업이 생성되는 시간과 기간을 선택하도록 하려면 기본 설정 없음을 선택합니다.

6. 완료되면 클러스터 수정을 선택합니다.

Using the AWS CLI

를 사용하여 새 탄력적 클러스터 스냅샷에 대한 자동 백업을 설정하려면 다음 파라미터와 함께 `create-cluster-snapshot` 작업을 AWS CLI사용합니다.

- **`--preferred-backup-window`** - 선택 사항. 자동 백업이 생성되는 일일 선호 시간 범위입니다. 기본값은 각각에 대해 8시간 블록 시간 중에서 임의로 선택한 30분 기간입니다 AWS 리전.

  제약 조건:

  - `hh24:mi-hh24:mi` 형식이어야 합니다.
  - 협정 세계시(UTC)여야 합니다.
  - 원하는 유지 관리 기간과 충돌하지 않아야 합니다.
  - 30분 이상이어야 합니다.
- **`--backup-retention-period`** - 선택 사항. 자동 백업이 보관되는 일수입니다. 기본값은 1입니다.

  제약 조건:

  - 최소 1 이상의 값을 지정해야 합니다.
  - 범위는 1~35여야 합니다.

> **Note**
>
> 자동 백업은 클러스터가 '활성' 상태인 경우에만 수행됩니다.

> **Note**
>
> `aws docdb-elastic update-cluster` 명령을 사용하여 기존 탄력적 클러스터의 `preferred-backup-window` 및 `backup-retention-period` 파라미터를 수정할 수도 있습니다.

다음은 자신의 정보를 각각의 *### ## ## ###*로 변경하는 예제입니다.

다음 `create-cluster` 예제에서는 자동 백업을 위한 보존 기간이 *7*일이고 기본 백업 기간이 *18:00~18:30 UTC*인 Amazon DocumentDB 탄력적 클러스터 *## ####*를 생성합니다.

Linux, macOS, Unix의 경우:

```
aws docdb-elastic create-cluster \
     --cluster-name sample-cluster \
     --shard-capacity 2 \
     --shard-count 2 \
     --admin-user-name SampleAdmin \
     --auth-type PLAIN_TEXT \
     --admin-user-password SamplePass123! \
     --preferred-backup-window 18:00-18:30 \
     --backup-retention-period 7
```

Windows의 경우:

```
aws docdb-elastic create-cluster ^
     --cluster-name sample-cluster ^
     --shard-capacity 2 ^
     --shard-count 2 ^
     --admin-user-name SampleAdmin ^
     --auth-type PLAIN_TEXT ^
     --admin-user-password SamplePass123! ^
     --preferred-backup-window 18:00-18:30 ^
     --backup-retention-period 7
```

Amazon DocumentDB 탄력적 클러스터 중지 및 시작

Amazon DocumentDB 탄력적 클러스터를 중지하고 시작하면 개발 및 테스트 환경 비용을 관리하는 데 도움이 됩니다. 필요하지 않을 경우 Amazon DocumentDB를 사용할 때마다 탄력적 클러스터를 생성하

고 삭제하는 대신 클러스터를 일시적으로 중지할 수 있습니다. 그런 다음 테스트를 재개할 때 다시 시작할 수 있습니다.

주제

- 탄력적 DB 클러스터의 중지 및 시작 개요
- 중지된 탄력적 클러스터에서 수행할 수 있는 작업

탄력적 DB 클러스터의 중지 및 시작 개요

Amazon DocumentDB 탄력적 클러스터가 필요하지 않은 기간에는 이 클러스터를 중지할 수 있습니다. 그런 다음 사용해야 할 때는 언제든지 클러스터를 다시 시작할 수 있습니다. 시작 및 중지를 사용하면 개발, 테스트 또는 연속 가용성을 필요로 하지 않는 유사한 활동에 사용되는 탄력적 클러스터의 설정 및 해제 프로세스가 간소화됩니다. 단일 작업으로 AWS Management Console 또는를 사용하여 탄력 AWS CLI 적 클러스터를 중지하고 시작할 수 있습니다.

탄력적 클러스터가 중지되는 동안 클러스터 스토리지 볼륨은 변경되지 않습니다. 지정된 보존 기간 내에는 스토리지, 수동 스냅샷 및 자동 백업 스토리지에 대한 비용만 청구됩니다. Amazon DocumentDB는 필요한 유지 관리 업데이트에 뒤처지지 않도록 7일 후에 탄력적 클러스터를 자동으로 시작합니다. 7일 후에 클러스터가 시작되면 탄력적 클러스터의 이용 비용이 다시 청구됩니다. 클러스터가 중지된 동안에는 쿼리하려면 클러스터가 사용 가능한 상태여야 하므로 스토리지 볼륨을 쿼리할 수 없습니다.

Amazon DocumentDB 탄력적 클러스터가 중지되면 클러스터는 어떤 식으로든 수정할 수 없습니다. 여기에는 클러스터 삭제가 포함됩니다.

Using the AWS Management Console

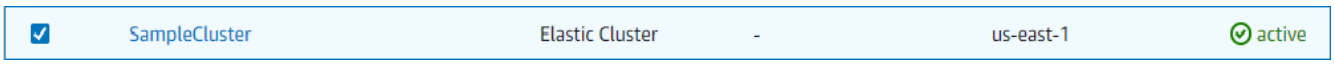
다음 절차에서는 사용 가능한 상태인 탄력적 클러스터를 중지하거나, 중지된 탄력적 클러스터를 시작하는 방법을 보여 줍니다.

Amazon DocumentDB 탄력적 클러스터 중지 및 시작 방법

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb:// https://https://https://httpsAmazon DocumentDB://https://https://://https://https://://https://:// https://://://://https://://://://
2. 탐색 창에서 클러스터를 선택합니다.

> **Tip**
>
> 화면 왼쪽에 탐색 창이 표시되지 않으면 페이지 왼쪽 상단 모서리에서 메뉴 아이콘 (≡) 을 선택합니다.

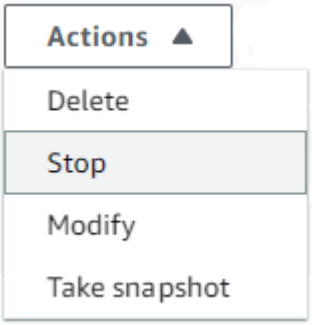
3. 클러스터 목록에서 중지하거나 시작하려는 클러스터 이름의 왼쪽에 있는 버튼을 선택합니다.



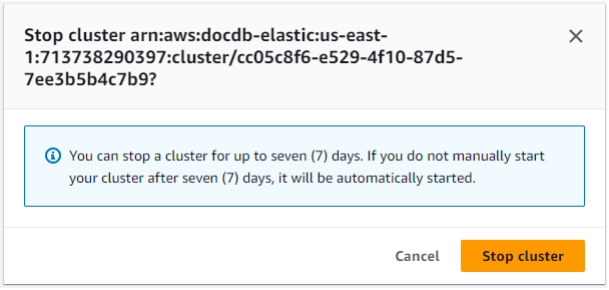
4. 작업을 선택한 다음 클러스터에서 수행하려는 작업을 선택합니다.

   - 클러스터를 중지하고 다음 작업을 수행합니다.

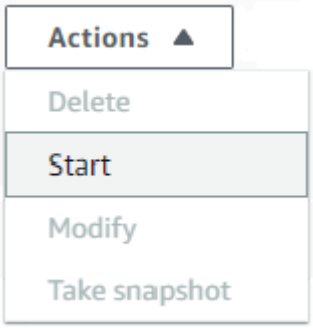
     a. 중지를 선택합니다.



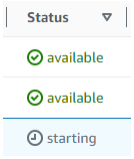
     b. 확인 대화 상자에서 클러스터 중지를 선택하여 탄력적 클러스터 중지를 확인하거나, 클러스터를 계속 실행하려면 취소를 선택합니다.



   - 클러스터를 시작하려면 클러스터가 중지된 상태에서 시작을 선택합니다.



5. Elastic 클러스터의 상태를 모니터링합니다. 클러스터를 시작한 경우 클러스터를 사용 가능할 때 클러스터 사용을 다시 시작할 수 있습니다. 자세한 내용은 클러스터 상태 확인 단원을 참조하십시오.



Using the AWS CLI

다음 절차에서는 활성 상태이거나 사용 가능한 상태인 탄력적 클러스터를 중지하거나, 중지된 탄력적 클러스터를 시작하는 방법을 보여 줍니다.

를 사용하여 탄력적 클러스터를 중지하려면 `stop-cluster` 작업을 AWS CLI사용합니다. 중지된 클러스터를 시작하려면 `start-cluster` 작업을 사용합니다. 두 작업은 `--cluster-arn` 파라미터를 사용합니다.

파라미터:

- **`--cluster-arn`** - 필수입니다. 중단하거나 시작하려는 탄력적 클러스터의 ARN 식별자입니다.

Example -를 사용하여 탄력적 클러스터를 중지하려면 AWS CLI

다음은 자신의 정보를 각각의 *### ## ## ###*로 변경하는 예제입니다.

다음 코드는 ARN이 `arn:aws:docdb-elastic:us-east-1:477568257630:cluster/b9f1d489-6c3e-4764-bb42-da62ceb7bda2`인 탄력적 클러스터를 중지합니다.

> **Note**
>
> 탄력적 클러스터가 활성 상태거나 사용 가능 상태여야 합니다.

Linux, macOS, Unix의 경우:

```
aws docdb-elastic stop-cluster \
   --cluster-arn arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2
```

Windows의 경우:

```
aws docdb-elastic stop-cluster ^
   --cluster-arn arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2
```

Example -를 사용하여 탄력적 클러스터를 시작하려면 AWS CLI

다음은 자신의 정보를 각각의 *### ## ## ###*로 변경하는 예제입니다.

다음 코드는 ARN이 `arn:aws:docdb-elastic:us-east-1:477568257630:cluster/b9f1d489-6c3e-4764-bb42-da62ceb7bda2`인 탄력적 클러스터를 시작합니다.

> **Note**
>
> 탄력적 클러스터는 현재 중지된 상태여야 합니다.

Linux, macOS, Unix의 경우:

```
aws docdb-elastic start-cluster \
   --cluster-arn arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2
```

Windows의 경우:

```
aws docdb-elastic start-cluster ^
   --cluster-arn arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2
```

중지된 탄력적 클러스터에서 수행할 수 있는 작업

클러스터가 중지되는 동안 Amazon DocumentDB 탄력적 클러스터의 구성을 수정할 수 없습니다. 이러한 관리 작업을 수행하기 전에 클러스터를 시작해야 합니다.

Amazon DocumentDB는 다시 시작한 후에만 중지된 탄력적 클러스터에 예약 유지 관리를 적용합니다. 7일 후에 Amazon DocumentDB는 중지된 탄력적 클러스터를 자동으로 시작하므로 유지 관리 상태에서 너무 늦어지지 않습니다. 탄력적 클러스터가 다시 시작되면 클러스터의 샤드 비용이 다시 청구됩니다.

탄력적 클러스터가 중지되는 동안 Amazon DocumentDB는 자동화된 백업을 수행하지 않으며 백업 보존 기간도 연장하지 않습니다.

Amazon DocumentDB 탄력적 클러스터 유지 관리

주제

- 보류 중인 탄력적 클러스터 유지 관리 작업 보기
- 탄력적 클러스터 엔진 업데이트
- 탄력적 클러스터 운영 체제 업데이트

Amazon DocumentDB는 주기적으로 Amazon DocumentDB 탄력적 클러스터 리소스에 대한 유지 관리를 수행합니다. 유지 관리에는 데이터베이스 엔진(탄력적 클러스터 유지 관리) 또는 탄력적 클러스터의 기본 운영 체제(OS 업데이트)에 대한 업데이트가 포함되는 경우가 많습니다. 데이터베이스 엔진 업데이트는 필수 패치이며 데이터베이스 엔진에 대한 보안 수정, 버그 수정 및 개선 사항을 포함합니다. 대부분의 운영 체제 패치는 선택 사항이지만 한동안 패치를 적용하지 않으면 보안 태세를 유지하기 위해 패치가 필요할 수 있고 자동으로 적용될 수 있습니다. 따라서 Amazon DocumentDB 탄력적 클러스터를 사용할 수 있게 되는 즉시 운영 체제 업데이트를 적용하는 것이 좋습니다.

데이터베이스 엔진 패치를 사용하려면 Amazon DocumentDB 탄력적 클러스터를 잠시 오프라인으로 전환해야 합니다. 사용 가능한 경우 Amazon DocumentDB 탄력적 클러스터의 예정된 유지 관리 기간 동안 이러한 패치가 자동으로 적용되도록 예약됩니다.

탄력적 클러스터에는 고유한 유지 관리 기간이 있습니다. 즉시 적용하지 않도록 선택한 탄력적 클러스터 수정 사항은 유지 관리 기간 중에 적용됩니다. 기본적으로 탄력적 클러스터를 생성하면 Amazon DocumentDB는 탄력적 클러스터에 유지 관리 기간을 할당합니다. 탄력적 클러스터를 생성할 때 유지 관리 기간을 선택할 수 있습니다. 또한 언제라도 비즈니스 일정 또는 절차에 맞춰 유지 관리 기간을 수정할 수 있습니다. 일반적으로 애플리케이션에 대 유지 관리의 영향을 최소화하는 유지 관리 기간을 서택하는 것이 좋습니다(예: 야간 또는 주말).

보류 중인 탄력적 클러스터 유지 관리 작업 보기

를 사용하여 엘라스틱 클러스터에 유지 관리 업데이트를 사용할 수 있는지 확인할 수 있습니다 AWS CLI.

업데이트가 있을 경우에는 다음 중 한 가지를 선택할 수 있습니다.

- 현재 다음 유지 관리 기간에 예약된 유지 관리 작업을 연기합니다(OS 패치만 해당).
- 유지 관리 작업을 즉시 적용합니다.
- 다음 유지 관리 기간 중 시작할 유지 관리 작업을 예약합니다.
- 선택한 적용 기간 중에 시작하도록 유지 관리 작업을 예약합니다.

유지 관리 기간에 따라 대기 중인 작업의 시작 시기가 결정되지만 이러한 작업의 전체 실행 시간이 제한되지는 않습니다.

다음 AWS CLI 작업을 사용하여 보류 중인 유지 관리 작업을 확인합니다. 보류 중인 유지 관리 작업을 모두 나열합니다.

```
aws docdb-elastic list-pending-maintenance-actions
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
'ResourcePendingMaintenanceActions': [
    {
        'ResourceArn': 'string-arn',
        'PendingMaintenanceActionDetails': [
            {
                'Action': 'ENGINE_UPDATE',
                'AutoAppliedAfterDate': 'string',
                'ForcedApplyDate': 'string',
                'OptInStatus': 'string',
                'CurrentApplyDate': 'string',
                'Description': 'string'
            },
        ]
    },
],
'NextToken': 'string'
}
```

지정된에서 보류 중인 유지 관리 작업(있는 경우)을 가져옵니다`resourceArn`.

```
aws docdb-elastic get-pending-maintenance-action --resource-arn string-arn
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
    'ResourcePendingMaintenanceAction': {
        'ResourceArn': 'string-arn',
        'PendingMaintenanceActionDetails': [
            {
                'Action': 'ENGINE_UPDATE',
                'AutoAppliedAfterDate': 'string',
                'ForcedApplyDate': 'string',
                'OptInStatus': 'string',
                'CurrentApplyDate': 'string',
                'Description': 'string'
            }
        ]
    }
}
```

파라미터:

- ResourceArn- 보류 중인 유지 관리 작업이 적용되는 리소스의 Amazon DocumentDB Amazon 리소스 이름(ARN)입니다.
- Action- 보류 중인 유지 관리 작업이 리소스에 적용됩니다.

  유효한 값:

  - ENGINE_UPDATE
  - ENGINE_UPGRADE
  - SECURITY_UPDATE
  - OS_UPDATE
  - MASTER_USER_PASSWORD_UPDATE
- AutoAppliedAfterDate-이 날짜 이후의 첫 번째 유지 관리 기간입니다.이 경우 NEXT_MAINTENANCE OPT_IN는 무시됩니다.
- ForcedApplyDate- 유지 관리 기간에 관계없이 적용됩니다.이 경우 IMMEDIATE OPT_IN는 무시됩니다.
- OptInStatus - 옵트인 요청의 유형을 지정하거나 옵트인 요청을 실행 취소하는 값입니다. IMMEDIATE 유형의 옵트인 요청은 실행 취소할 수 없습니다.

  유효한 값:

- IMMEDIATE - 유지 관리 작업을 즉시 적용합니다.
- NEXT_MAINTENANCE - 리소스에 대한 다음 유지 관리 기간 중에 유지 관리 작업을 적용합니다.
- APPLY_ON- 리소스의 다음 유지 관리 기간에 관계없이 지정된 적용 날짜에 유지 관리 작업을 적용합니다.
- UNDO_OPT_IN- 기존 NEXT_MAINTENANCE 또는 APPLY_ON옵트인 요청을 취소합니다.

- CurrentApplyDateopt-in-type 인 경우 표시됩니다APPLY_ON.
- Description- 유지 관리 작업에 대한 옵션 설명입니다.

탄력적 클러스터 엔진 업데이트

Amazon DocumentDB를 사용하여 유지 관리 작업을 적용하는 시기를 선택할 수 있습니다. 를 사용하여 Amazon DocumentDB가 업데이트를 적용하는 시기를 결정할 수 있습니다 AWS CLI.

보류 중인 유지 관리 작업 적용:

```
aws docdb-elastic apply-pending-maintenance-action
--resource-arn string-arn
--apply-action string-enum
--opt-in-type string-enum
[--apply-on string-date-range]
```

파라미터:

- **--resource-arn**- 보류 중인 유지 관리 작업이 적용되는 리소스의 Amazon DocumentDB Amazon 리소스 이름(ARN)입니다.
- **--apply-action** - 이 리소스에 적용할 대기 중인 유지 관리 작업입니다.

  유효한 값:

  - ENGINE_UPDATE
  - ENGINE_UPGRADE
  - SECURITY_UPDATE
  - OS_UPDATE
  - MASTER_USER_PASSWORD_UPDATE
- **--opt-in-type** - 옵트인 요청의 유형을 지정하거나 옵트인 요청을 실행 취소하는 값입니다. IMMEDIATE 유형의 옵트인 요청은 실행 취소할 수 없습니다.

유효한 값:

- IMMEDIATE - 유지 관리 작업을 즉시 적용합니다.
- NEXT_MAINTENANCE - 리소스에 대한 다음 유지 관리 기간 중에 유지 관리 작업을 적용합니다.
- APPLY_ON- 리소스의 다음 유지 관리 기간에 관계없이 지정된 적용 날짜에 유지 관리 작업을 적용합니다.
- UNDO_OPT_IN- 기존 NEXT_MAINTENANCE 또는 APPLY_ON옵트인 요청을 취소합니다.

- **[--apply-on]**opt-in-type 인 경우 필수입니다APPLY_ON. 형식: yyyy/MM/dd HH:mm-yyyy/MM/dd HH:mm (이 옵션은 UTC 시간을 사용합니다. 시작 시간은 최소 30분에서 최대 14일 또는 보류 중인 작업에 대한 강제 적용 날짜 중 먼저 도래하는 날짜부터 향후 언제든지 가능합니다. 시작부터 종료까지의 기간은 최소 30분에서 최대 8시간입니다.)

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
 'ResourcePendingMaintenanceAction': {
        'ResourceArn': 'string-arn',
        'PendingMaintenanceActionDetails': [
            {
                'Action': 'SECURITY_UPDATE',
                'AutoAppliedAfterDate': 'string',
                'ForcedApplyDate': 'string',
                'OptInStatus': 'IMMEDIATE',
                'CurrentApplyDate': 'string',
                'Description': 'string'
            },
        ]
 }
}
```

파라미터:

- ResourceArn- 보류 중인 유지 관리 작업이 적용되는 리소스의 Amazon DocumentDB Amazon 리소스 이름(ARN)입니다.
- Action- 보류 중인 유지 관리 작업이 리소스에 적용됩니다.

  유효한 값:

  - ENGINE_UPDATE

  - ENGINE_UPGRADE
  - SECURITY_UPDATE
  - OS_UPDATE
  - MASTER_USER_PASSWORD_UPDATE
- AutoAppliedAfterDate-이 날짜 이후의 첫 번째 유지 관리 기간입니다.이 경우 NEXT_MAINTENANCE OPT_IN는 무시됩니다.
- ForcedApplyDate- 유지 관리 기간에 관계없이 적용됩니다.이 경우 IMMEDIATE OPT_IN는 무시됩니다.
- OptInStatus - 옵트인 요청의 유형을 지정하거나 옵트인 요청을 실행 취소하는 값입니다. IMMEDIATE 유형의 옵트인 요청은 실행 취소할 수 없습니다.

  유효한 값:

  - IMMEDIATE - 유지 관리 작업을 즉시 적용합니다.
  - NEXT_MAINTENANCE - 리소스에 대한 다음 유지 관리 기간 중에 유지 관리 작업을 적용합니다.
  - APPLY_ON- 리소스의 다음 유지 관리 기간에 관계없이 지정된 적용 날짜에 유지 관리 작업을 적용합니다.
  - UNDO_OPT_IN- 기존 NEXT_MAINTENANCE 또는 APPLY_ON옵트인 요청을 취소합니다.
- CurrentApplyDateopt-in-type 인 경우 표시됩니다APPLY_ON.
- Description- 유지 관리 작업에 대한 옵션 설명입니다.

적용 날짜

각 유지 관리 작업에는 대기 중인 유지 관리 작업을 설명할 때 찾을 수 있는 적용 날짜가 있습니다. 에서 보류 중인 유지 관리 작업의 출력을 읽으면 세 AWS CLI가지 날짜가 나열됩니다.

- CurrentApplyDate - 유지 관리 작업이 즉시 또는 다음 유지 관리 기간에 적용되는 날짜입니다. 유지 관리가 선택 사항인 경우이 값은 null일 수 있습니다.
- ForcedApplyDate - 유지 관리가 사용자의 유지 관리 기간과 상관없이 자동으로 적용되는 날짜입니다.
- AutoAppliedAfterDate - 유지 관리가 클러스터의 유지 관리 기간 동안 이 날짜 이후에 적용됩니다.

사용자가 생성한 유지 관리 작업

Amazon DocumentDBelastic 클러스터 사용자는 클러스터 구성에 대한 업데이트를 시작할 수 있습니다.

클러스터 기본 암호 업데이트

```
aws docdb-elastic update-cluster
--cluster-arn string-arn
[--admin-user-password string]
[--auth-type string-enum]
[--apply-method string-enum]
[--apply-on string-date-range]
#... other parameters of the API that follow here are not relevant for this
 configuration
```

파라미터:

- **`--cluster-arn`**- 유지 관리 작업이 적용될 리소스의 Amazon DocumentDB Amazon 리소스 이름(ARN)입니다.
- **`[--admin-user-password]`**- 관리자 사용자와 연결된 암호입니다.
- **`[--auth-type]`**- 탄력적 클러스터에 액세스하는 데 사용되는 암호를 가져올 위치를 결정하는 데 사용되는 인증 유형입니다. 유효한 형식은 `PLAIN_TEXT` 및 `SECRET_ARN`입니다.
- **`[--apply-method]`**- 적용할 메서드 유형을 지정하는 값입니다. 허용되는 값은 `IMMEDIATE` 및 `APPLY_ON`입니다. 기본값은 `IMMEDIATE`입니다.
- **`[--apply-on]`**-이 `apply-method`인 경우 필수입니다`APPLY_ON`. 형식: `yyyy/MM/dd HH:mm-yyyy/MM/dd HH:mm` (이 옵션은 UTC 시간을 사용합니다. 시작 시간은 최소 30분에서 최대 14일 사이의 향후 언제든지 가능합니다. 시작부터 종료까지의 기간은 최소 30분에서 최대 8시간입니다.)

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
 'ResourcePendingMaintenanceAction': {
        'ResourceArn': 'string-arn',
        'PendingMaintenanceActionDetails': [
            {
                'Action': 'MASTER_USER_PASSWORD_UPDATE',
                'OptInStatus': 'APPLY_ON',
                'CurrentApplyDate': 'string',
                'Description': 'string'
```

```
            },
        ]
 }
}
```

Amazon DocumentDB 유지 관리 윈도우 변경

유지 관리 기간은 사용률이 가장 낮은 시간에 할당되어야 하므로 수시로 변경되어야 할 수 있습니다. 이 기간 동안에는 시스템 변경(예: 스케일 스토리지 작업 변경)이 적용되고 중단이 필요한 경우에만 탄력적 클러스터를 사용할 수 없습니다. 필요한 변경을 수행하는 데 필요한 최소 시간 동안만 사용할 수 없습니다.

기본값은 각 Amazon Web Services Region(아마존 웹 서비스 리전)의 8시간 블록에서 임의로 선택된 30분 창으로, 주 마다 임의의 요일에 발생합니다.

유지 관리 기간을 변경하려면 섹션을 참조하세요엘라스틱 클러스터 구성 수정.

탄력적 클러스터 운영 체제 업데이트

Amazon DocumentDB 탄력적 클러스터는 때때로 운영 체제 업데이트가 필요합니다. Amazon DocumentDB는 운영 체제를 최신 버전으로 업그레이드하여 데이터베이스 성능과 고객의 전반적인 보안 태세를 개선합니다. 운영 체제 업데이트는 Amazon DocumentDB 탄력적 클러스터의 클러스터 엔진 버전을 변경하지 않습니다.

Amazon DocumentDB 탄력적 클러스터에 대한 대부분의 운영 체제 업데이트는 선택 사항이며 이를 적용할 설정된 날짜가 없습니다. 그러나 이러한 업데이트를 한동안 적용하지 않으면 결국 클러스터 유지 관리 기간 동안 필요해지고 자동으로 적용될 수 있습니다. 이는 데이터베이스의 보안 태세를 유지하는 데 도움이 됩니다. 예기치 않은 가동 중지를 방지하려면 운영 체제 업데이트가 제공되는 즉시 Amazon DocumentDB 탄력적 클러스터에 적용하고 비즈니스 요구 사항에 따라 편한 시간에 클러스터 유지 관리 기간을 설정하는 것이 좋습니다.

Amazon DocumentDB 탄력적 클러스터에 대한 저장 데이터 암호화

다음 주제는 Amazon DocumentDB 탄력적 클러스터에 대한 AWS Key Management Service 암호화 키를 학습, 생성 및 모니터링하는 데 도움이 됩니다.

주제

- Amazon DocumentDB 탄력적 클러스터가에서 권한 부여를 사용하는 방법 AWS KMS
- 고객 관리형 키 생성
- Amazon DocumentDB 탄력적 클러스터에 대한 암호화 키 모니터링

- 자세히 알아보기

Amazon DocumentDB 탄력적 클러스터는 키 관리를 위해 AWS Key Management Service (AWS KMS)와 자동으로 통합되며 봉투 암호화라는 메서드를 사용하여 데이터를 보호합니다. 봉투 암호화에 대한 자세한 내용은 AWS Key Management Service 개발자 가이드의 봉투 암호화를 참조하십시오.

AWS KMS key 는 키의 논리적 표현입니다. KMS 키에는 키 ID, 생성 날짜, 설명 및 키 상태와 같은 메타데이터가 포함됩니다. 또한 KMS 키에는 데이터를 암호화 및 복호화하는 데 사용되는 키 재료도 포함됩니다. KMS 키에 대한 자세한 내용은 AWS Key Management Service 개발자 가이드의 AWS KMS keys 섹션을 참조하십시오.

Amazon DocumentDB 탄력적 클러스터는 두 가지 유형의 키를 사용한 암호화를 지원합니다:

- AWS 소유 키 - Amazon DocumentDB 탄력적 클러스터는 기본적으로 이러한 키를 사용하여 개인 식별 데이터를 자동으로 암호화합니다. AWS소유 키를 보거나, 관리하거나, 사용하거나, 사용을 감사할 수 없습니다. 하지만 데이터를 암호화하는 키를 보호하기 위해 어떤 작업을 수행하거나 어떤 프로그램을 변경할 필요가 없습니다. 자세한 내용은 AWS Key Management Service 개발자 안내서의 AWS 소유 키를 참조하세요.
- 고객 관리형 키 - AWS KMS keys 생성, 소유 및 관리하는 대칭입니다. 이 암호화 계층을 완전히 제어할 수 있으므로 다음과 같은 작업을 수행할 수 있습니다.
  - 키 정책 수립 및 유지
  - IAM 정책 및 권한 부여 수립 및 유지
  - 키 정책 활성화 및 비활성화
  - 키 암호화 자료 교체
  - 태그 추가
  - 키 별칭 만들기
  - 삭제를 위한 스케줄 키

  자세한 내용은 AWS Key Management Service 개발자 가이드의 고객 관리형 키를 참조하십시오.

> **Important**
>
> Amazon DocumentDB는 대칭 암호화 KMS 키만 지원하므로 클러스터를 암호화하려면 대칭 암호화 KMS 키를 사용해야 합니다. 비대칭 KMS 키를 사용하여 Amazon DocumentDB 탄력적 클러스터의 데이터를 암호화하지 마십시오. 자세한 내용은 AWS Key Management Service 개발자 가이드의 AWS KMS의 비대칭 키를 참조하십시오.

예를 들어, 키에 대한 액세스가 취소된 경우 Amazon DocumentDB가 더 이상 클러스터의 암호화 키에 액세스할 수 없는 경우 암호화된 클러스터는 터미널 상태가 됩니다. 이러한 경우에는 백업 파일에서만 클러스터를 복원할 수 있습니다. Amazon DocumentDB의 경우 항상 1일 동안 백업이 실행됩니다. 또한 암호화된 Amazon DocumentDB 클러스터에 대한 키를 비활성화하면 해당 클러스터에 대한 읽기 및 쓰기 액세스 권한이 손실됩니다. Amazon DocumentDB는 접근 권한이 없는 키로 암호화된 클러스터를 만나면 클러스터를 터미널 상태로 만듭니다. 이러한 상태에서는 클러스터를 더 이상 사용하지 못하기 때문에 데이터베이스의 현재 상태를 복구할 수 없습니다. 클러스터를 복원하려면 Amazon DocumentDB의 암호화 키에 대한 액세스를 다시 실행한 다음 백업에서 클러스터를 복원해야 합니다.

Important

암호화된 클러스터의 KMS 키는 이미 생성한 후에는 변경할 수 없습니다. 암호화된 탄력적 클러스터를 생성하기 전에 암호화 키 요구 사항을 결정해야 합니다.

Amazon DocumentDB 탄력적 클러스터가에서 권한 부여를 사용하는 방법 AWS KMS

Amazon DocumentDB 탄력적 클러스터는 고객 관리형 키를 사용하려면 권한이 필요합니다.

고객 관리형 키로 암호화된 클러스터를 생성하면 Amazon DocumentDB 탄력적 클러스터는 `CreateGrant` 요청을 전송하여 사용자를 대신하여 권한을 생성합니다 AWS KMS. 의 권한 부여 AWS KMS 는 Amazon DocumentDB 탄력적 클러스터에 고객 계정의 KMS 키에 대한 액세스 권한을 부여하는 데 사용됩니다.

Amazon DocumentDB 탄력적 클러스터는 다음 내부 작업에 고객 관리형 키를 사용하려면 권한이 필요합니다:

- 트래커 또는 지오펜스 컬렉션을 생성할 때 입력한 대칭 고객 관리형 KMS 키 ID가 유효한지 AWS KMS 확인하려면에 `DescribeKey` 요청을 보냅니다.
- AWS KMS 에 `GenerateDataKey` 요청을 보내 고객 관리형 키로 암호화된 데이터 키를 생성합니다.
- 데이터를 암호화하는 AWS KMS 데 사용할 수 있도록에 암호화된 데이터 키를 복호화하라는 `Decrypt` 요청을 보냅니다.

- 언제든지 권한 부여에 대한 액세스 권한을 취소하거나 고객 관리형 키에 대한 서비스 액세스를 제거할 수 있습니다. 이렇게 하면 Amazon DocumentDB 탄력적 클러스터는 고객 관리형 키에 의해 암호화된 데이터에 액세스할 수 없게 되어 해당 데이터에 의존하는 작업에 영향을 미칩니다.

고객 관리형 키 생성

AWS Management Console 또는 AWS KMS API를 사용하여 대칭 고객 관리형 키를 생성할 수 있습니다.

대칭 고객 관리형 키

AWS Key Management Service 개발자 가이드의 대칭형 고객 관리형 키 생성 단계를 따르십시오.

키 정책

키 정책에서는 고객 관리형 키에 대한 액세스를 제어합니다. 모든 고객 관리형 키에는 키를 사용할 수 있는 사람과 키를 사용하는 방법을 결정하는 문장이 포함된 정확히 하나의 키 정책이 있어야 합니다. 고객 관리형 키를 만들 때 키 정책을 지정할 수 있습니다. 자세한 내용은 AWS Key Management Service 개발자 가이드의 AWS Key Management Service 개요에 있는 KMS 키 액세스 정보를 참조하십시오.

고객 관리형 키를 Amazon DocumentDB 탄력적 클러스터 리소스와 함께 사용하려면 키 정책에서 다음과 같은 API 작업을 허용해야 합니다:

- `kms:CreateGrant` - 고객 관리형 키에 권한 부여를 추가합니다. 지정된 KMS 키에 대한 제어 액세스 권한을 부여합니다. 이 키를 통해 Amazon Location Service가 필요로 하는 작업에 액세스할 수 있습니다. 권한 부여에 대한 자세한 내용은 AWS Key Management Service 개발자 가이드의 AWS KMS의 권한 부여를 참조하십시오.
- `kms:DescribeKey` - Docdb Elastic이 키를 검증할 수 있도록 고객 관리형 키 세부 정보를 제공합니다.
- `kms:Decrypt` – Docdb Elastic이 저장된 암호화된 데이터 키를 사용하여 암호화된 데이터에 액세스할 수 있습니다.
- `kms:GenerateDataKey`— 데이터 키가 암호화에 즉시 사용되지 않기 때문에 Docdb Elastic이 암호화된 데이터 키를 생성하고 저장할 수 있습니다.

자세한 내용은 AWS Key Management Service 개발자 안내서의 키 정책의 AWS 서비스에 대한 권한 및 키 액세스 문제 해결을 참조하세요.

IAM 정책을 통한 고객 관리형 키 액세스 제한

KMS 키 정책 외에도 IAM 정책에서 KMS 키 권한을 제한할 수도 있습니다.

다양한 방법으로 IAM 정책을 더 엄격하게 설정할 수 있습니다. 예를 들어 고객 관리형 키를 Amazon DocumentDB 탄력적 클러스터에서 시작되는 요청에만 사용할 수 있도록 하려면 `kms:ViaService` 조건 키를 `docdb-elastic.<region-name>.amazonaws.com` 값과 함께 사용할 수 있습니다.

자세한 내용은 AWS Key Management Service 개발자 가이드의 다른 계정의 사용자가 CMK를 사용하도록 허용을 참조하십시오.

Amazon DocumentDB 탄력적 클러스터에 대한 암호화 키 모니터링

Docdb Elastic 리소스와 함께 AWS KMS key 고객 관리형 키를 사용하는 경우 AWS CloudTrail 또는 Amazon CloudWatch Logs를 사용하여 Docdb Elastic이 보내는 요청을 추적할 수 있습니다 AWS KMS.

다음 예제는 `CreateGrant`, `Decrypt`, `GenerateDataKeyWithoutPlainText`및가 Amazon DocumentDB 탄력적 클러스터가 고객 관리형 키`DescribeKey`로 암호화된 데이터에 액세스하기 위해 호출하는 AWS KMS key 작업을 모니터링하는 AWS CloudTrail 이벤트입니다.

CreateGrant

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE",
                "arn": "arn:aws:iam::111122223333:assumed-role/Admin/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Sampleuser01"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2023-05-09T23:04:20Z",
                "mfaAuthenticated": "false"
```

```
            }
        },
        "invokedBy": "docdb-elastic.amazonaws.com"
    },
    "eventTime": "2023-05-09T23:55:48Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "docdb-elastic.amazonaws.com",
    "userAgent": "docdb-elastic.amazonaws.com",
    "requestParameters": {
        "retiringPrincipal": "docdb-elastic.us-east-1.amazonaws.com",
        "granteePrincipal": "docdb-elastic.us-east-1.amazonaws.com",
        "operations": [
            "Decrypt",
            "Encrypt",
            "GenerateDataKey",
            "GenerateDataKeyWithoutPlaintext",
            "ReEncryptFrom",
            "ReEncryptTo",
            "CreateGrant",
            "RetireGrant",
            "DescribeKey"
        ],
        "keyId": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
    "responseElements": {
        "grantId":
 "0ab0ac0d0b000f00ea00cc0a0e00fc00bce000c000f0000000c0bc0a0000aaafSAMPLE",
        "keyId": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": false,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
```

```
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

GenerateDataKey

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE",
                "arn": "arn:aws:iam::111122223333:assumed-role/Admin/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Sampleuser01"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2023-05-10T18:02:59Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "docdb-elastic.amazonaws.com"
    },
    "eventTime": "2023-05-10T18:03:25Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "docdb-elastic.amazonaws.com",
    "userAgent": "docdb-elastic.amazonaws.com",
    "requestParameters": {
        "keySpec": "AES_256",
        "keyId": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
```

```
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

Decrypt

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE",
                "arn": "arn:aws:iam::111122223333:assumed-role/Admin/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Sampleuser01"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2023-05-10T18:05:49Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "docdb-elastic.amazonaws.com"
```

```
    },
    "eventTime": "2023-05-10T18:06:19Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "docdb-elastic.amazonaws.com",
    "userAgent": "docdb-elastic.amazonaws.com",
    "requestParameters": {
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

DescribeKey

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE",
                "arn": "arn:aws:iam::111122223333:assumed-role/Admin/Sampleuser01",
```

```
                "accountId": "111122223333",
                "userName": "Sampleuser01"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2023-05-09T23:04:20Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "docdb-elastic.amazonaws.com"
    },
    "eventTime": "2023-05-09T23:55:48Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DescribeKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "docdb-elastic.amazonaws.com",
    "userAgent": "docdb-elastic.amazonaws.com",
    "requestParameters": {
        "keyId": "alias/SampleKmsKey"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

자세히 알아보기

다음 리소스에서 저장 데이터 암호화에 대한 추가 정보를 확인할 수 있습니다:

- AWS KMS 개념에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 AWS Key Management Service 기본 개념을 참조하세요.
- AWS KMS 보안에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의에 대한 보안 모범 사례를 AWS Key Management Service 참조하세요.

탄력적 클러스터에서 서비스 연결 역할

Amazon DocumentDB 탄력적 클러스터는 AWS Identity and Access Management (IAM) 서비스 연결 역할을 사용합니다. 서비스 연결 역할은 Amazon DocumentDB 탄력적 클러스터에 직접 연결되는 고유한 유형의 IAM 역할입니다. 서비스 연결 역할은 Amazon DocumentDB 탄력적 클러스터에 의해 사전 정의되며 서비스가 사용자를 대신하여 다른 AWS 서비스를 호출하는 데 필요한 모든 권한을 포함합니다.

서비스 연결 역할을 사용하면 필요한 권한을 수동으로 추가할 필요가 없으므로 Amazon DocumentDB 탄력적 클러스터를 보다 쉽게 사용할 수 있습니다. Amazon DocumentDB 탄력적 클러스터는 서비스 연결 역할의 권한을 정의하며, 달리 정의되지 않는 한 Amazon DocumentDB 탄력적 클러스터만 역할을 맡을 수 있습니다. 정의된 권한에는 신뢰 정책과 권한 정책이 포함되며 이 권한 정책은 다른 IAM 엔티티에 연결할 수 없습니다. 먼저 역할의 관련 리소스를 삭제해야만 역할을 삭제할 수 있습니다. 이렇게 하면 리소스에 대한 액세스 권한을 실수로 제거할 수 없기 때문에 Amazon DocumentDB 탄력적 클러스터 리소스를 보호할 수 있습니다.

서비스 연결 역할을 지원하는 다른 서비스에 대한 내용은 IAM과 함께 작동하는AWS 서비스를 참조하고 서비스 연결 역할 열에서 예로 표시된 서비스를 찾으십시오. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예 링크를 선택합니다.

탄력적 클러스터에 대한 서비스 연결 역할 권한

Amazon DocumentDB 탄력적 클러스터는 라는 서비스 연결 역할을 사용하여 Amazon DocumentDB 탄력적 클러스터가 클러스터를 대신하여 AWS 서비스를 호출할 수 `AWS ServiceRoleForDocDB-Elastic` 있도록 허용합니다.

이 서비스 연결 역할에는 계정에서 운영할 수 있는 권한을 부여하는 `AmazonDocDB-ElasticServiceRolePolicy`라는 권한 정책이 연결되어 있습니다. 역할 권한 정책을 통해 Amazon DocumentDB 탄력적 클러스터는 지정된 리소스에 대해 다음 작업을 완료할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": [
                        "AWS/DocDB-Elastic"
                    ]
                }
            }
        }
    ]
}
```

> **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 생성할 수 없습니다. 서비스 연결 역할을 생성할 권한이 있는지 확인합니다. 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/docdb-elastic.amazonaws.com/
AWSServiceRoleForDocDB-Elastic",
    "Condition": {
"StringLike": {
"iam:AWSServiceName":"docdb-elastic.amazonaws.com"
        }
    }
}
```

자세한 내용은 AWS ID 및 액세스 관리 사용 설명서의 서비스 연결 역할 권한을 참조하십시오.

Amazon DocumentDB 탄력적 클러스터를 위한 서비스 연결 역할 생성

서비스 링크 역할은 수동으로 생성할 필요가 없습니다. DB 인스턴스를 생성할 때 Amazon DocumentDB 탄력적 클러스터는 서비스 연결 역할을 생성합니다.

Amazon DocumentDB 탄력적 클러스터에 대한 서비스 연결 역할 편집

Amazon DocumentDB 탄력적 클러스터에서는 `AWS ServiceRoleForDocDB-Elastic` 서비스 연결 역할을 편집할 수 없습니다. 서비스 링크 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 AWS ID 및 액세스 관리 사용 설명서의 서비스 연결 역할 편집을 참조하십시오.

Amazon DocumentDB 탄력적 클러스터의 서비스 연결 역할 삭제

서비스 연결 역할이 필요한 기능 또는 서비스가 더 이상 필요 없는 경우에는 해당 역할을 삭제할 것을 권합니다. 그렇게 하면 적극적으로 모니터링하거나 유지 관리하지 않은 미사용 엔터티가 없습니다. 그러나 서비스 연결 역할을 삭제하려면 먼저 모든 클러스터를 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 로그인하고 IAM 콘솔을 엽니다.
2. IAM 콘솔의 탐색 창에서 역할을 선택합니다. 그런 다음 `AWS ServiceRoleForDocDB-Elastic` 역할의 이름(확인란 아님)을 선택합니다.
3. 선택한 역할의 요약 페이지에서 Access Advisor 탭을 선택합니다.

> **Note**
>
> Amazon DocumentDB 탄력적 클러스터가 `AWS ServiceRoleForDocDB-Elastic` 역할을 사용하고 있는지 확신할 수 없으면 역할을 삭제할 수 있습니다. 서비스가 역할을 사용하는 경우 삭제가 실패하고 역할이 사용 중인 AWS 리전 를 볼 수 있습니다. 역할이 사용 중인 경우에는 세션이 종료될 때까지 기다렸다가 역할을 삭제해야 합니다. 서비스 연결 역할에 대한 세션은 취소할 수 없습니다.
>
> `AWS ServiceRoleForDocDB-Elastic` 역할을 제거하려면 먼저 모든 클러스터를 삭제해야 합니다.

모든 클러스터 삭제

Amazon DocumentDB 콘솔에서 클러스터를 삭제하려면

1. AWS Management Console에 로그인하고 Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 클러스터를 선택합니다.
3. 삭제할 클러스터를 선택합니다.
4. 작업에 대해 삭제를 선택합니다.
5. 최종 스냅샷을 생성할까요?를 묻는 메시지가 나타나면 예 또는 아니오를 선택합니다.
6. 이전 단계에서 예를 선택한 경우 최종 스냅샷 이름에 최종 스냅샷 이름을 입력합니다.
7. Delete(삭제)를 선택합니다.

> **Note**
>
> IAM 콘솔, IAM CLI 또는 IAM API를 사용하여 `AWS ServiceRoleForDocDB-Elastic` 서비스 연결 역할을 삭제할 수 있습니다. 자세한 내용은 AWS ID 및 액세스 관리 사용 설명서의 서비스 연결 역할 삭제를 참조하십시오.

Amazon DocumentDB 모니터링

AWS 서비스를 모니터링하는 것은 시스템을 건강하게 유지하고 최적의 상태로 작동하기 위한 중요한 부분입니다. 오류나 성능 저하가 발생했을 때 이를 쉽게 디버깅하고 수정할 수 있도록 AWS 솔루션의 모든 부분으로부터 모니터링 데이터를 수집하는 것이 좋습니다. AWS 솔루션 모니터링을 시작하기 전에 다음 질문에 대한 답변을 고려하고 공식화하는 것이 좋습니다.

- 모니터링의 목표
- 모니터링할 리소스는 무엇입니까?
- 이 리소스를 얼마나 자주 모니터링합니까?
- 사용할 모니터링 도구
- 누가 모니터링을 담당합니까?
- 문제가 발생한 경우 알릴 대상은 누구이며 어떤 의미가 있습니까?

현재 성능 패턴을 이해하고 성능 이상을 식별하고 문제를 해결할 수 있는 방법을 공식화하려면 다양한 시간대 및 다양한 로드 조건에 대한 기준 성능 지표를 설정해야 합니다. AWS 솔루션을 모니터링할 때 나중에 참조하고 기준을 설정하기 위해 기록 모니터링 데이터를 저장하는 것이 좋습니다.

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 사용량이 많을 경우 해당 애플리케이션의 목표(처리량 또는 동시성)와 일치하고 예상되는 결과라면 문제가 되지 않을 수 있습니다.
- 스토리지 볼륨 사용량 - 총 스토리지 볼륨 공간의 85% 이상이 계속 사용될 경우 스토리지 사용량(`VolumeBytesUsed`)을 검사합니다. 스토리지 볼륨에서 데이터를 삭제할 수 있는지 또는 다른 시스템에 데이터를 아카이브하여 공간을 확보할 수 있는지 결정합니다. 자세한 정보는 Amazon DocumentDB 스토리지 및 Amazon DocumentDB 할당량 및 제한(을)를 참조하세요.
- 네트워크 트래픽 - 네트워크 트래픽의 경우 시스템 관리자에게 문의하여 해당 도메인 네트워크 및 인터넷 연결의 예상 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. 인스턴스에 대한 최적의 사용자 연결 수는 해당 인스턴스 클래스와, 수행하는 작업의 복잡성에 따라 다릅니다.

- IOPS 지표 - OPS 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.

Amazon DocumentDB(MongoDB 호환)은 모니터링을 통해 Amazon DocumentDB 클러스터와 인스턴스의 상태와 성능을 평가할 수 있는 Amazon CloudWatch 지표를 다양하게 제공합니다. Amazon DocumentDB 콘솔, CloudWatch API AWS CLI, 성능 개선 도우미 등 다양한 도구를 사용하여 Amazon DocumentDB 지표를 볼 수 있습니다.

주제

- Amazon DocumentDB 클러스터 상태 모니터링
- Amazon DocumentDB 인스턴스 상태 모니터링
- Amazon DocumentDB 권장 사항 보기
- Amazon DocumentDB 이벤트 구독 사용
- CloudWatch를 사용하여 Amazon DocumentDB 모니터링
- 를 사용하여 Amazon DocumentDB API 호출 로깅 AWS CloudTrail
- Amazon DocumentDB 작업 프로파일링
- 성능 개선 도우미로 모니터링

Amazon DocumentDB 클러스터 상태 모니터링

클러스터 상태는 클러스터의 상태를 나타냅니다. Amazon DocumentDB 콘솔 또는 명령을 사용하여 클러스터의 상태를 볼 수 있습니다 AWS CLI `describe-db-clusters`.

주제

- 클러스터 상태 값
- 클러스터 상태 모니터링

클러스터 상태 값

다음 표에는 클러스터 상태에 유효한 값이 나열되어 있습니다.

클러스터 상태	설명
active	클러스터가 활성 상태입니다. 이 상태는 엘라스틱 클러스터에만 적용됩니다.
available	클러스터에 문제가 없으며 사용할 수 있습니다. 이 상태는 인스턴스 기반 클러스터에만 적용됩니다.
backing-up	현재 클러스터를 백업 중입니다.
creating	클러스터를 생성 중입니다. 생성 도중에는 액세스할 수 없습니다.
deleting	클러스터를 삭제 중입니다. 삭제 도중에는 액세스할 수 없습니다.
failing-over	기본 인스턴스에서 Amazon DocumentDB 복제본에 대한 장애 조치가 수행 중입니다.
inaccessible-encryption-credentials	클러스터를 암호화하거나 복호화하는 데 사용되는 AWS KMS 키에는 액세스할 수 없습니다.
maintenance	클러스터에 유지 관리 업데이트를 적용 중입니다. 이 상태는 Amazon DocumentDB가 한참 전에 미리 예약하는 클러스터 수준의 유지 관리에 사용됩니다.

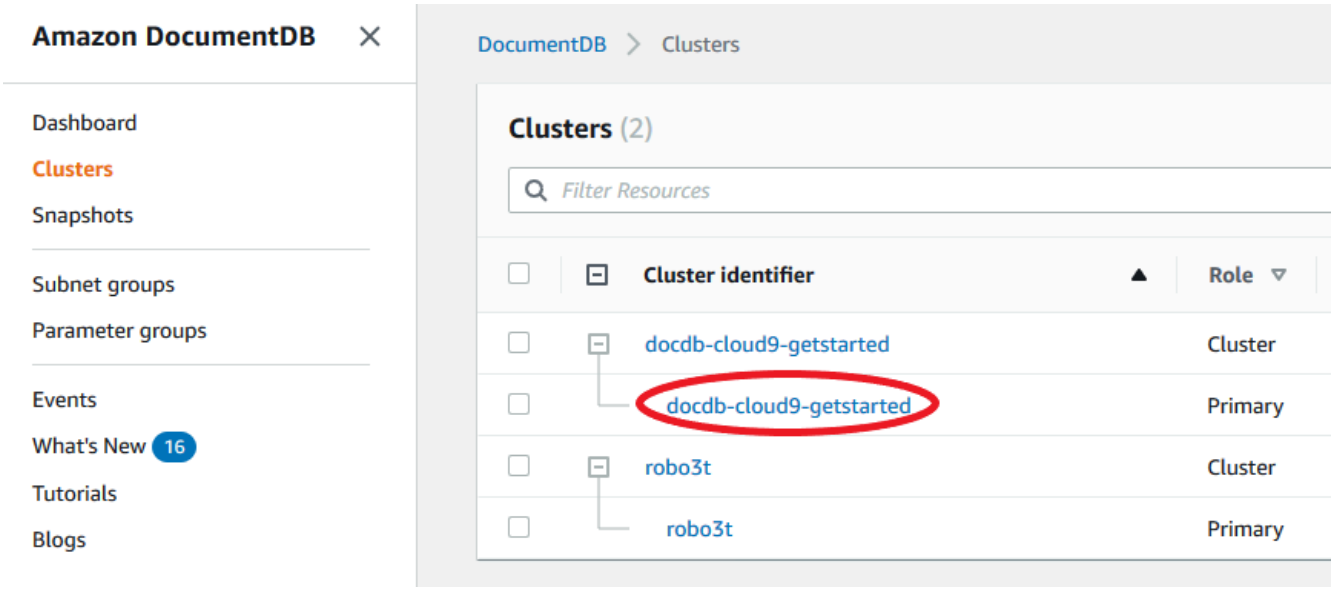
클러스터 상태	설명
`migrating`	클러스터 스냅샷이 클러스터로 복원 중입니다.
`migration-failed`	마이그레이션이 실패했습니다.
`modifying`	고객의 클러스터 수정 요청으로 인해 클러스터를 수정 중입니다.
`renaming`	고객의 이름 바꾸기 요청으로 인해 클러스터 이름을 바꾸는 중입니다.
`resetting-master-credentials`	고객의 재설정 요청으로 인해 클러스터 사용자 자격 증명을 재설정하는 중입니다.
`upgrading`	클러스터 엔진 버전이 현재 업그레이드 중입니다.

클러스터 상태 모니터링

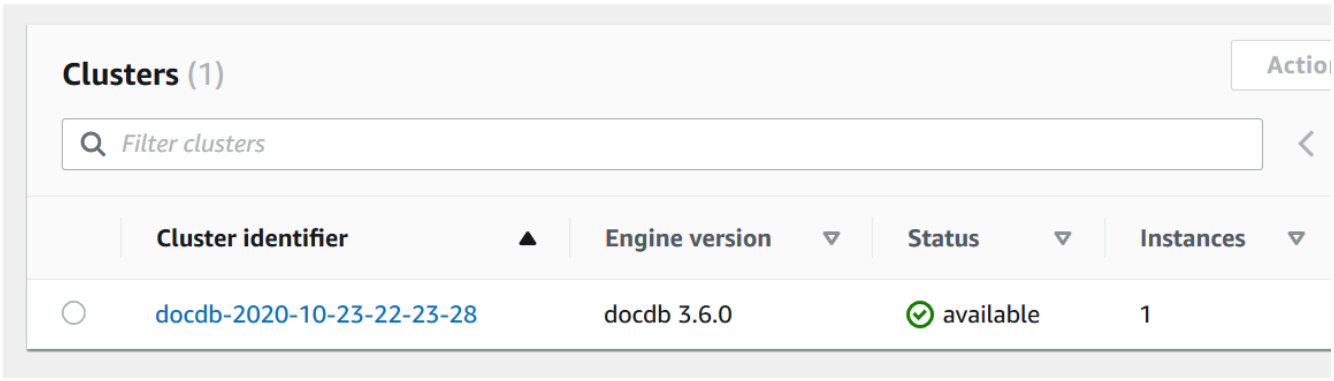
Using the AWS Management Console

AWS Management Console 를 사용하여 클러스터의 상태를 확인할 때 다음 절차를 사용합니다.

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 클러스터(Clusters)를 선택합니다.
3. 클러스터 탐색 상자에 클러스터 식별자 열이 표시됩니다. 인스턴스는 아래 스크린샷과 마찬가지로 클러스터 아래에 나열됩니다.



4. 클러스스터 식별자 열에서 관심 있는 인스턴스의 이름을 찾습니다. 그런 다음, 인스턴스의 상태를 찾으려면 아래 그림과 같이 상태 열에 대한 해당 행을 읽습니다.



Using the AWS CLI

AWS CLI 를 사용하여 클러스터의 상태를 확인할 때는 `describe-db-clusters` 작업을 사용합니다. 다음 코드는 `sample-cluster` 클러스터의 상태를 찾습니다.

Linux, macOS 또는 Unix의 경우:

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster  \
    --query 'DBClusters[*].[DBClusterIdentifier,Status]'
```

Windows의 경우:

```
aws docdb describe-db-clusters ^
```

```
    --db-cluster-identifier sample-cluster  ^
    --query 'DBClusters[*].[DBClusterIdentifier,Status]'
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
[
    [
        "sample-cluster",
        "available"
    ]
]
```

Amazon DocumentDB 인스턴스 상태 모니터링

Amazon DocumentDB는 데이터베이스에 구성된 각 인스턴스의 현재 상태에 대한 정보를 제공합니다.

Amazon DocumentDB 인스턴스의 상태를 확인할 수 있는 세 가지 유형이 있습니다.

- 인스턴스 상태:이 상태는의 클러스터 테이블 상태 열에 표시 AWS Management Console 되며 인스턴스의 현재 수명 주기 조건을 보여줍니다. 상태 열에 표시된 값은 `DescribeDBCluster` API 응답의 `Status` 필드에서 파생됩니다.
- 인스턴스 상태:이 상태는의 클러스터 테이블의 인스턴스 상태 열에 표시 AWS Management Console 되며 데이터 관리 및 검색을 담당하는 구성 요소인 데이터베이스 엔진이 실행 중인지 여부를 보여줍니다. 인스턴스 상태 열에 표시된 값은 Amazon CloudWatch `EngineUptime` 시스템 지표를 기반으로 합니다.
- 유지 관리 상태:이 상태는의 클러스터 테이블 유지 관리 열에 표시 AWS Management Console 되며 인스턴스에 적용해야 하는 유지 관리 이벤트의 상태를 나타냅니다. 유지 관리 상태는 다른 인스턴스 상태와는 독립적이며 `PendingMaintenanceAction` API에서 파생됩니다. 유지 관리 상태에 대한 자세한 내용은 Amazon DocumentDB 유지 관리를 참조하세요.

주제

- 인스턴스 상태 값
- AWS Management Console 또는를 사용하여 인스턴스 상태 모니터링 AWS CLI
- 인스턴스 상태 값
- 를 사용하여 인스턴스 상태 모니터링 AWS Management Console

인스턴스 상태 값

다음 표에는 인스턴스에 대해 가능한 상태 값과 각 상태별 요금이 표시되어 있습니다. 인스턴스와 스토리지에 요금이 청구되는지, 스토리지에만 청구되는지, 혹은 청구되지 않는지 표시됩니다. 모든 인스턴스 상태의 백업 사용에는 항상 청구됩니다.

인스턴스 상태	청구	설명
`available`	청구	인스턴스에 문제가 없으며 사용할 수 있습니다.
`backing-up`	청구	현재 인스턴스를 백업 중입니다.
`configuring-log-exports`	청구	Amazon CloudWatch Logs에 로그 파일을 게시하는 기능이 이 인스턴스에 대해 활성화 또는 비활성화 상태입니다.
`creating`	미청구	인스턴스를 생성 중입니다. 생성 중인 인스턴스에는 액세스할 수 없습니다.
`deleting`	미청구	인스턴스를 삭제 중입니다.
`failed`	미청구	인스턴스 오류가 발생하여 Amazon DocumentDB에서 이를 복구하지 못했습니다. 데이터를 복구하려면 인스턴스의 복원 가능한 가장 최근 시간에 대한 시점 복원을 수행합니다.
`inaccessible-encryption-credentials`	미청구	인스턴스를 암호화하거나 복호화하는 데 사용되는 AWS KMS 키에 액세스할 수 없습니다.
`incompatible-network`	미청구	Amazon DocumentDB에서 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 DocumentDB에서 인스턴스에 대한 IP 주소를 받을 수 없는 경우 이러한 상태가 발생할 수 있습니다.

인스턴스 상태	청구	설명
maintenance	청구	Amazon DocumentDB는 인스턴스에 유지 관리 업데이트를 적용합니다. 이 상태는 Amazon DocumentDB가 한참 전에 미리 예약하는 인스턴스 수준의 유지 관리에 사용됩니다. 현재 이 상태를 통해 고객에게 유지 관리 작업을 추가로 알릴 방법을 평가 중입니다.
modifying	청구	인스턴스 수정 요청으로 인해 인스턴스를 수정 중입니다.
rebooting	청구	인스턴스 재부팅이 필요한 요청 또는 Amazon DocumentDB 프로세스로 인해 인스턴스를 재부팅 중입니다.
renaming	청구	이름 바꾸기 요청으로 인해 인스턴스 이름을 바꾸는 중입니다.
resetting-master-credentials	청구	재설정 요청으로 인해 인스턴스 사용자 자격 증명을 재설정하는 중입니다.
restore-error	청구	특정 시점으로 복원을 시도하거나 스냅샷에서 복원을 시도할 때 인스턴스에서 오류가 발생했습니다.
starting	요금 부과 스토리지	인스턴스를 시작하는 중입니다.
stopped	요금 부과 스토리지	인스턴스가 중지되었습니다.
stopping	요금 부과 스토리지	인스턴스를 중지 중입니다.

인스턴스 상태	청구	설명
`storage-full`	청구	인스턴스에 할당된 스토리지 용량이 거의 다 찼습니다. 즉시 해결해야 하는 중요한 상태이므로 인스턴스를 수정하여 스토리지를 확장합니다. 스토리지 공간이 부족해지면 경고하도록 Amazon CloudWatch 경보를 설정해 이러한 상황이 발생하지 않도록 할 수 있습니다.

AWS Management Console 또는를 사용하여 인스턴스 상태 모니터링 AWS CLI

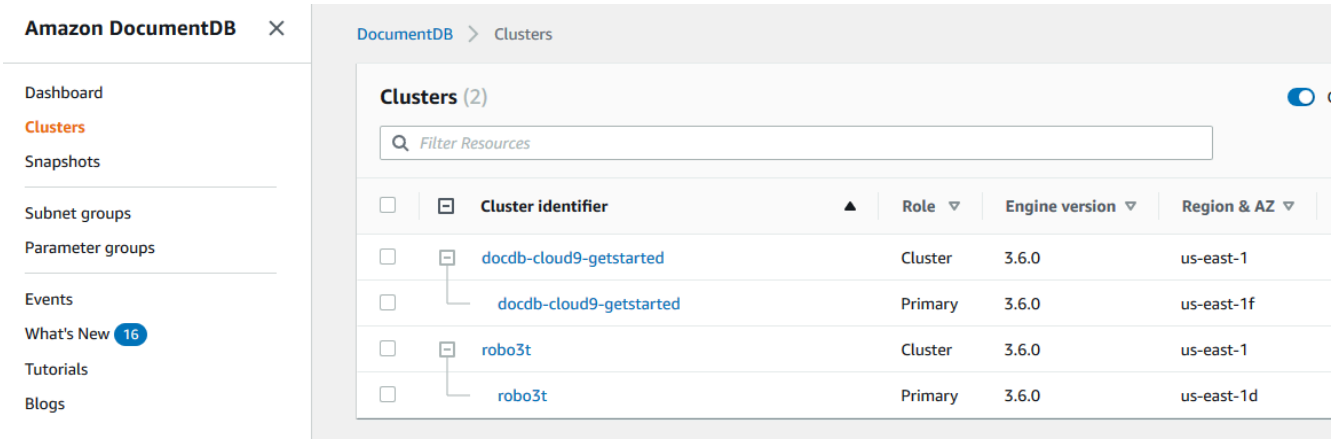
AWS Management Console 또는를 사용하여 인스턴스의 상태를 AWS CLI 모니터링합니다.

Using the AWS Management Console

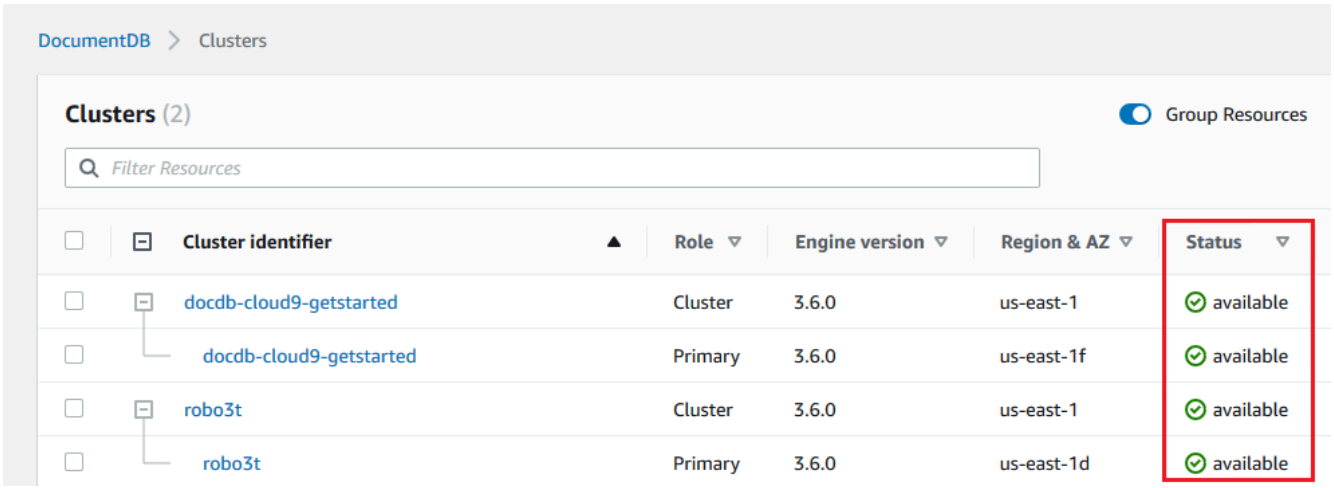
AWS Management Console 를 사용하여 클러스터의 상태를 확인할 때 다음 절차를 사용합니다.

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 클러스터(Clusters)를 선택합니다.

> **Note**
>
> 참고로 클러스터 탐색 상자의 클러스터 식별자 열에는 클러스터와 인스턴스가 모두 표시됩니다. 아래 이미지와 유사한 클러스터 아래에 인스턴스가 나열됩니다.



3. 관심 있는 인스턴스의 이름을 찾으십시오. 그런 다음 인스턴스의 상태를 찾으려면 다음 그림과 같이 상태 열에 대한 해당 행을 읽습니다.



Using the AWS CLI

AWS CLI 를 사용하여 클러스터의 상태를 확인할 때는 `describe-db-instances` 작업을 사용합니다. 다음 코드는 `sample-cluster-instance-01` 인스턴스의 상태를 찾습니다.

Linux, macOS 또는 Unix의 경우:

```
aws docdb describe-db-instances \
          --db-instance-identifier sample-cluster-instance-01  \
          --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceStatus]'
```

Windows의 경우:

```
aws docdb describe-db-instances ^
          --db-instance-identifier sample-cluster-instance-01  ^
          --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceStatus]'
```

이 작업의 출력은 다음과 같습니다.

```
[
          [
              "sample-cluster-instance-01",
              "available"
          ]
      ]
```

인스턴스 상태 값

다음 표는 인스턴스를 위한 잠재적인 상태 값을 나열하고 있습니다. 의 클러스터 테이블에 있는 인스턴스 상태 열은 데이터 저장 AWS Management Console, 관리 및 검색을 담당하는 구성 요소인 데이터베이스 엔진이 정상적으로 작동하는지 여부를 보여줍니다. 또한 이 열은 CloudWatch에서 사용할 수 있는 `EngineUptime` 시스템 지표에 각 인스턴스의 상태가 표시되는지 여부도 나타냅니다.

인스턴스 상태	설명
정상	데이터베이스 엔진이 Amazon DocumentDB 인스턴스에서 실행 중입니다.
비정상	데이터베이스 엔진이 실행 중이 아니거나 1분이 채 지나지 않아 다시 시작되었습니다.

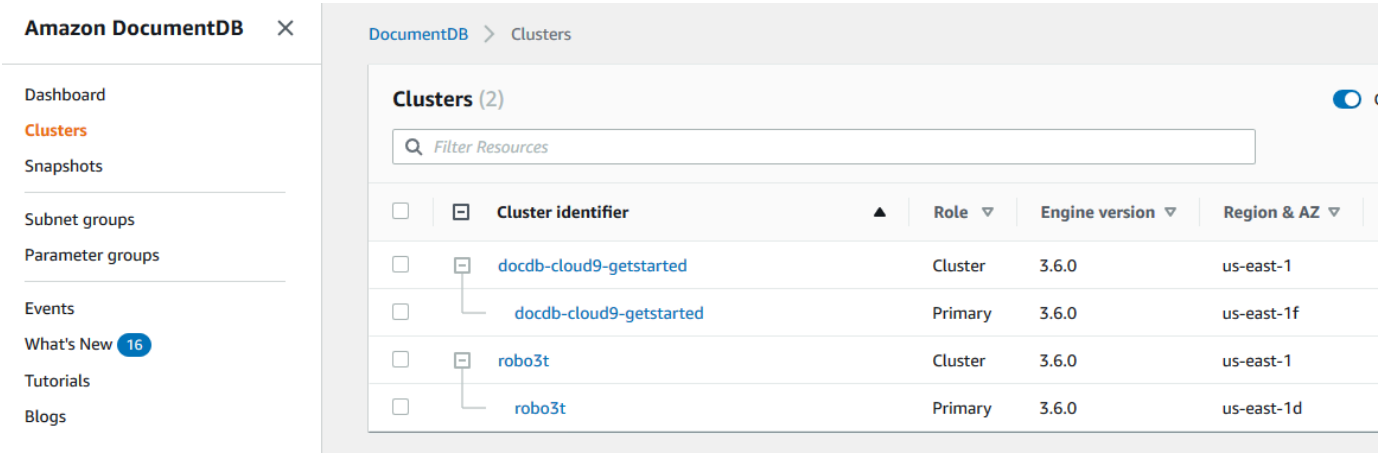
를 사용하여 인스턴스 상태 모니터링 AWS Management Console

AWS Management Console 를 사용하여 인스턴스의 상태를 모니터링합니다.

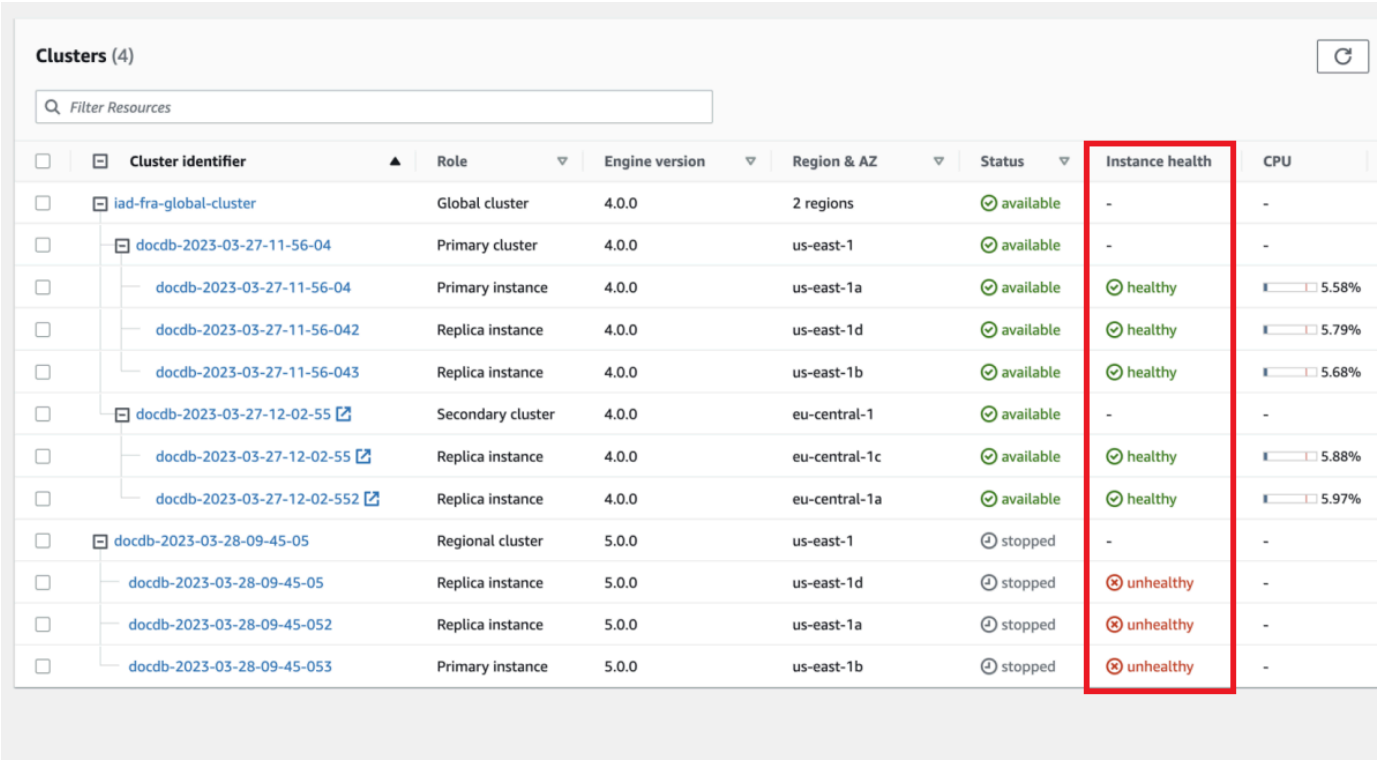
AWS Management Console를 사용할 때 다음 단계에 따라 인스턴스의 상태를 파악합니다.

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 클러스터를 선택합니다.

> **Note**
>
> 참고로 클러스터 탐색 상자의 클러스터 식별자 열에는 클러스터와 인스턴스가 모두 표시됩니다. 아래 이미지와 유사한 클러스터 아래에 인스턴스가 나열됩니다.



3. 관심 있는 인스턴스의 이름을 찾으십시오. 그런 다음 인스턴스의 상태를 찾으려면 다음 그림과 같이 인스턴스 상태 열에 대한 해당 행을 읽습니다.



> **Note**
>
> 인스턴스 상태 폴링은 60초마다 발생하며 CloudWatch `EngineUptime` 시스템 지표를 기반으로 합니다. 인스턴스 상태 열의 값은 자동으로 업데이트됩니다.

Amazon DocumentDB 권장 사항 보기

Amazon DocumentDB는 인스턴스 및 클러스터와 같은 데이터베이스 리소스에 대한 자동화된 권장 사항 목록을 제공합니다. 이러한 권장 사항은 클러스터와 인스턴스 구성 분석을 통해 모범 사례 지침을 제공합니다.

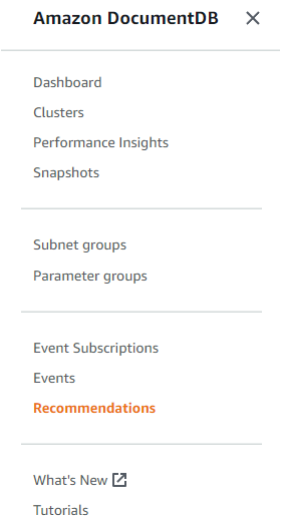
이러한 권장 사항의 예는 다음을 참조하세요.

유형	설명	권장 사항	추가 정보
1개의 인스턴스	클러스터에는 하나의 인스턴스만 포함됩니다.	성능 및 가용성: 동일한 인스턴스 클래스가 있는 또 다른 인스턴스를 다른 가용 영역에 추가하는 것이 좋습니다.	Amazon DocumentDB 고가용성 및 복제

Amazon DocumentDB는 리소스가 생성되거나 수정될 때 리소스에 대해 권장 사항을 생성합니다. Amazon DocumentDB는 리소스를 주기적으로 스캔하고 권장 사항을 생성합니다.

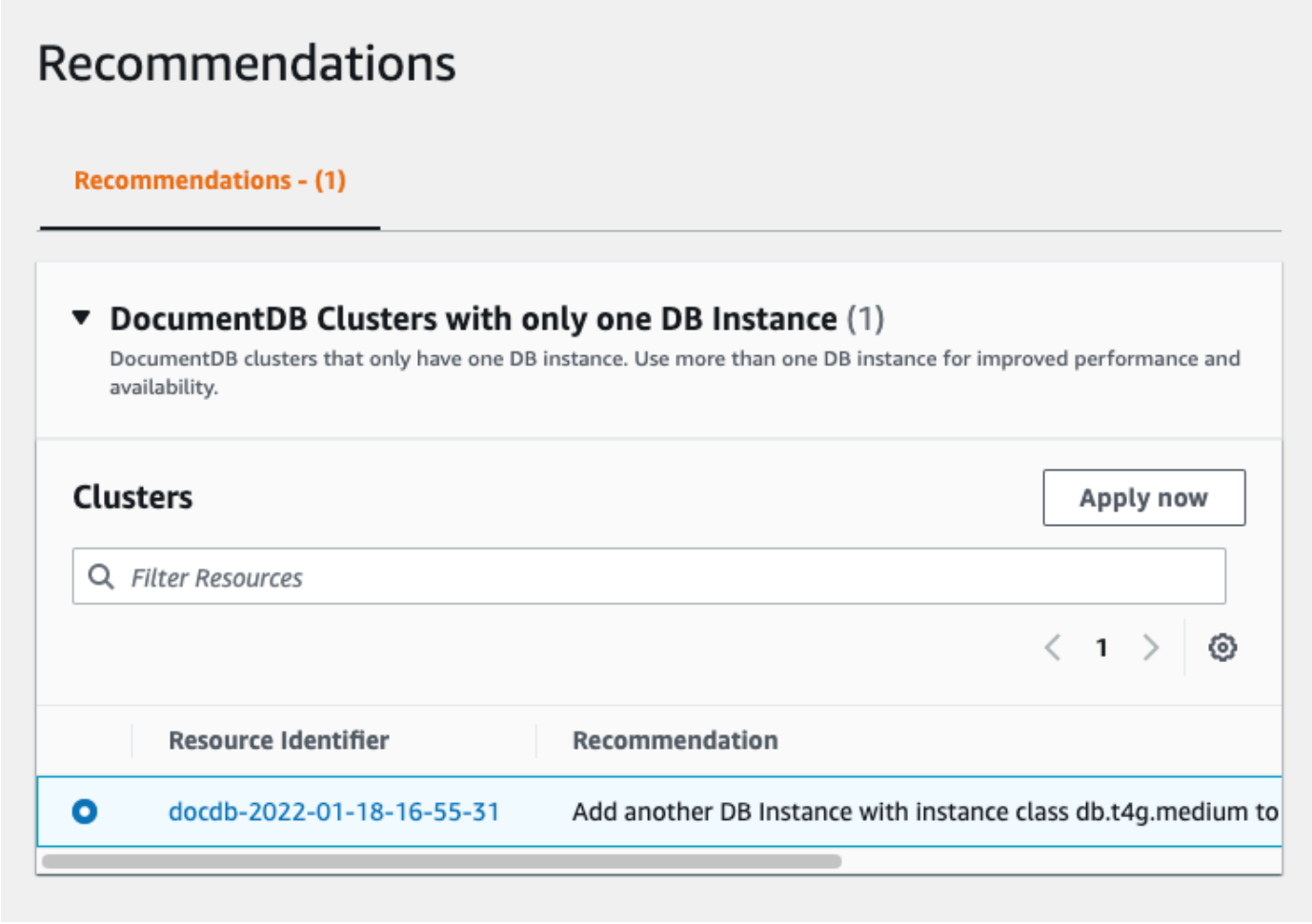
Amazon DocumentDB 권장 사항을 확인하고 이에 대한 조치를 취하려면

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 권장 사항을 선택합니다.



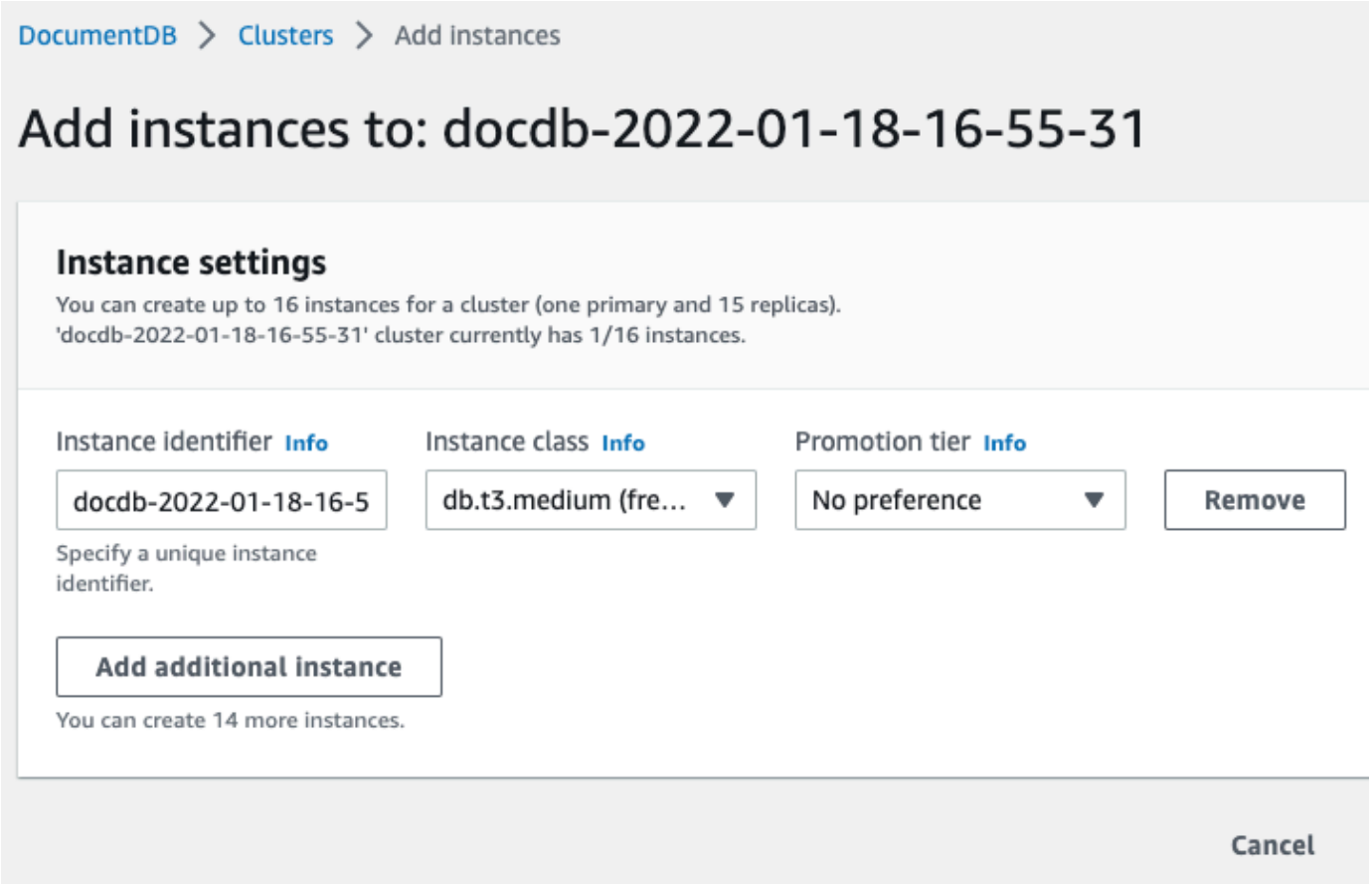
3. 권장 사항 대화 상자에서 관심 섹션을 펼치고 권장 작업을 선택합니다.

   아래 예제에서 권장 작업은 인스턴스가 하나뿐인 Amazon DocumentDB 클러스터에 적용됩니다. 성능과 가용성을 개선하려면 다른 인스턴스를 추가하는 것이 좋습니다.



4. 지금 적용을 클릭합니다.

   이 예시에서는 인스턴스 추가 대화상자가 나타납니다.



5. 새 인스턴스의 설정을 수정하고 생성을 클릭합니다.

Amazon DocumentDB 이벤트 구독 사용

Amazon DocumentDB는 Amazon Simple Notification Service(SNS)를 사용하여 Amazon DocumentDB 이벤트 발생 시 알림을 제공합니다. 이러한 알림은 이메일 AWS 리전, 문자 메시지 또는 HTTP 엔드포인트 호출과 같이에 대해 Amazon SNS에서 지원하는 모든 형식일 수 있습니다.

Amazon DocumentDB는 이러한 이벤트를 가입할 수 있는 카테고리로 그룹화하여 해당 카테고리의 이벤트가 발생할 때 알림을 받을 수 있도록 합니다. 구독 가능한 이벤트 범주로는 DB 인스턴스, DB 클러스터, DB 스냅샷, DB 클러스터 스냅샷, DB 파라미터 그룹, DB 보안 그룹 등이 있습니다. 예를 들어 임의의 DB 인스턴스에 대한 Backup 카테고리를 구독할 경우 백업 관련 이벤트가 발생하여 DB 인스턴스에 영향을 끼칠 때마다 알림 메시지가 수신됩니다. 또한 이벤트 알림 메시지 구독이 변경되어도 알림 메시지가 수신됩니다.

이벤트는 클러스터와 인스턴스 수준에서 모두 발생하므로 클러스터나 인스턴스에 가입하면 이벤트를 수신할 수 있습니다.

이벤트 알림 메시지는 구독 생성 시 입력한 주소로 보내집니다. 모든 이벤트 알림을 받는 구독과 프로덕션 인스턴스에 대한 중요한 이벤트만 포함하는 다른 구독 등 여러 가지 구독을 생성할 수 있습니다. 구독을 삭제하지 않고도 알림을 쉽게 끌 수 있습니다. 이렇게 하려면 Amazon DocumentDB 콘솔에서 활성화 라디오 버튼을 아니오로 설정하세요.

> **Important**
>
> Amazon DocumentDB는 이벤트 스트림에서 전송되는 이벤트 순서를 보장하지 않습니다. 이벤트 순서는 변경될 수 있습니다.

Amazon DocumentDB는 Amazon SNS 주제의 Amazon 리소스 이름(ARN)을 사용하여 각 구독을 식별합니다. 구독을 작성할 때 Amazon DocumentDB 콘솔이 ARN을 작성합니다.

Amazon DocumentDB 이벤트 구독에 대한 청구는 Amazon SNS를 통해 이루어집니다. Amazon SNS 요금은 이벤트 알림 사용 시 적용됩니다. 자세한 내용은 Amazon Simple Notification Service Pricing을 참조하세요. 아마존 Amazon SNS 요금 외에 이벤트 구독료는 Amazon DocumentDB에서 청구하지 않습니다.

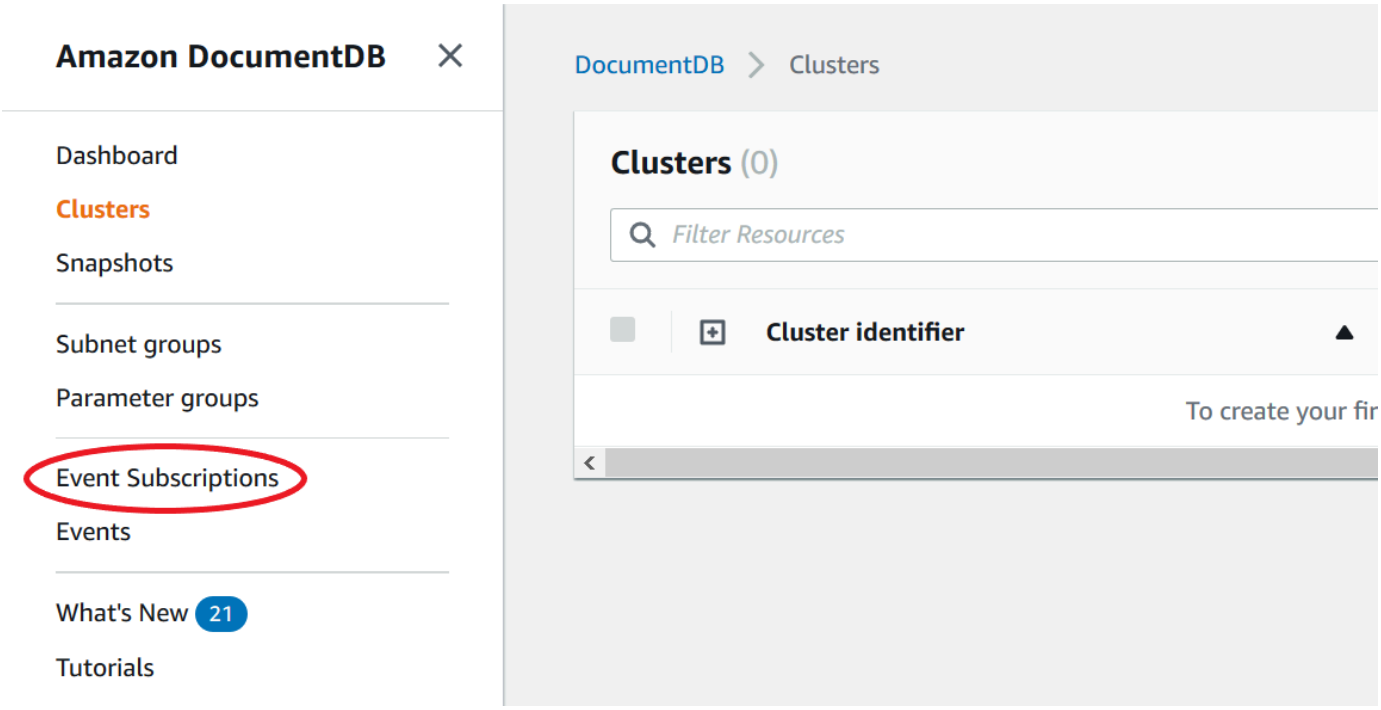
주제

- Amazon DocumentDB 이벤트 구독
- Amazon DocumentDB 이벤트 알림 구독 관리
- Amazon DocumentDB 이벤트 범주 및 메시지

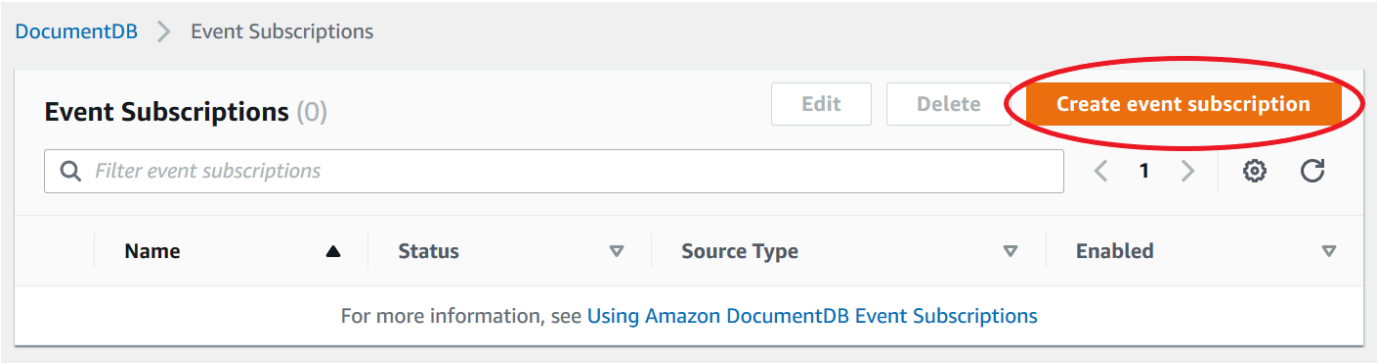
Amazon DocumentDB 이벤트 구독

Amazon DocumentDB 콘솔을 사용하여 다음과 같이 이벤트 구독을 구독할 수 있습니다:

1. 에서에 로그인 AWS Management Console 합니다https://console.aws.amazon.com/docdb.
2. 탐색 창에서 [이벤트 구독]을 선택합니다.

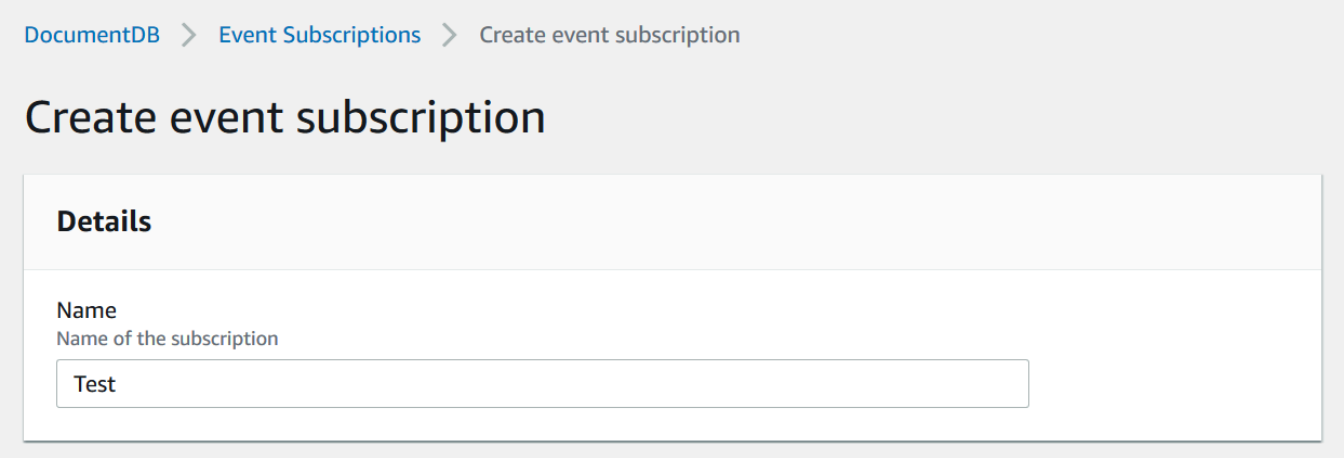


3. 이벤트 구독 창에서 이벤트 구독 생성을 선택합니다.

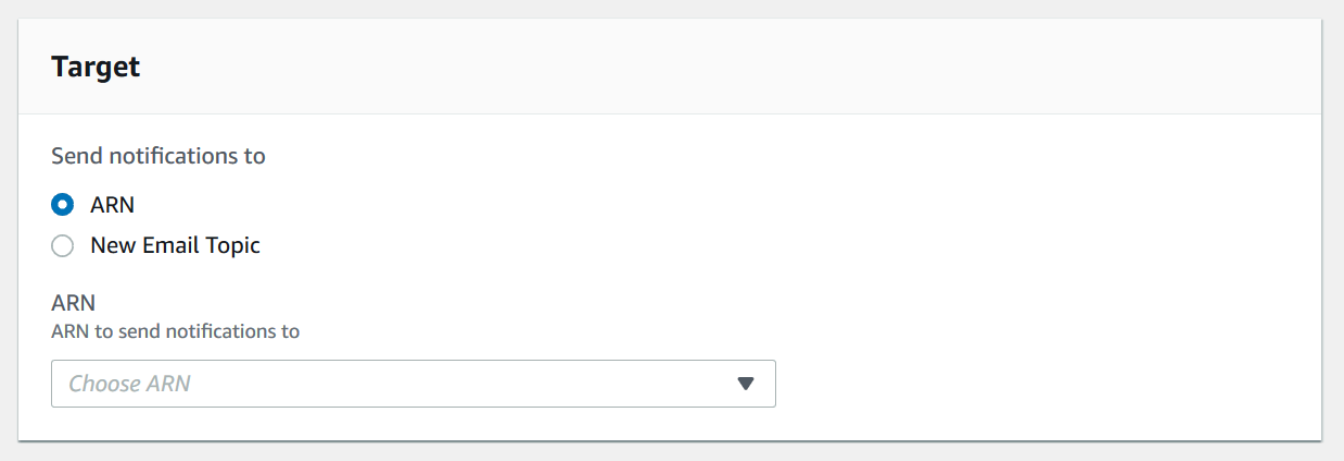


4. [Create event subscription] 대화 상자에서 다음과 같이 실행합니다.

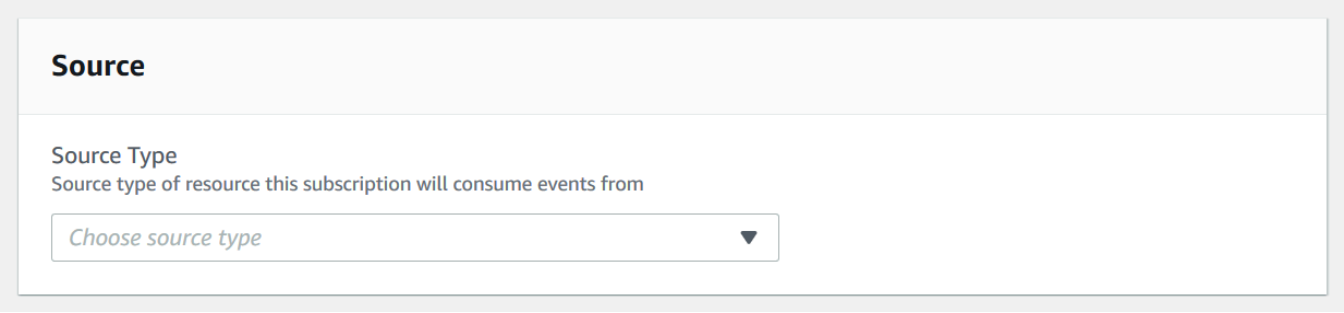
   - 이름에서 이벤트 알림 구독 이름을 입력합니다.



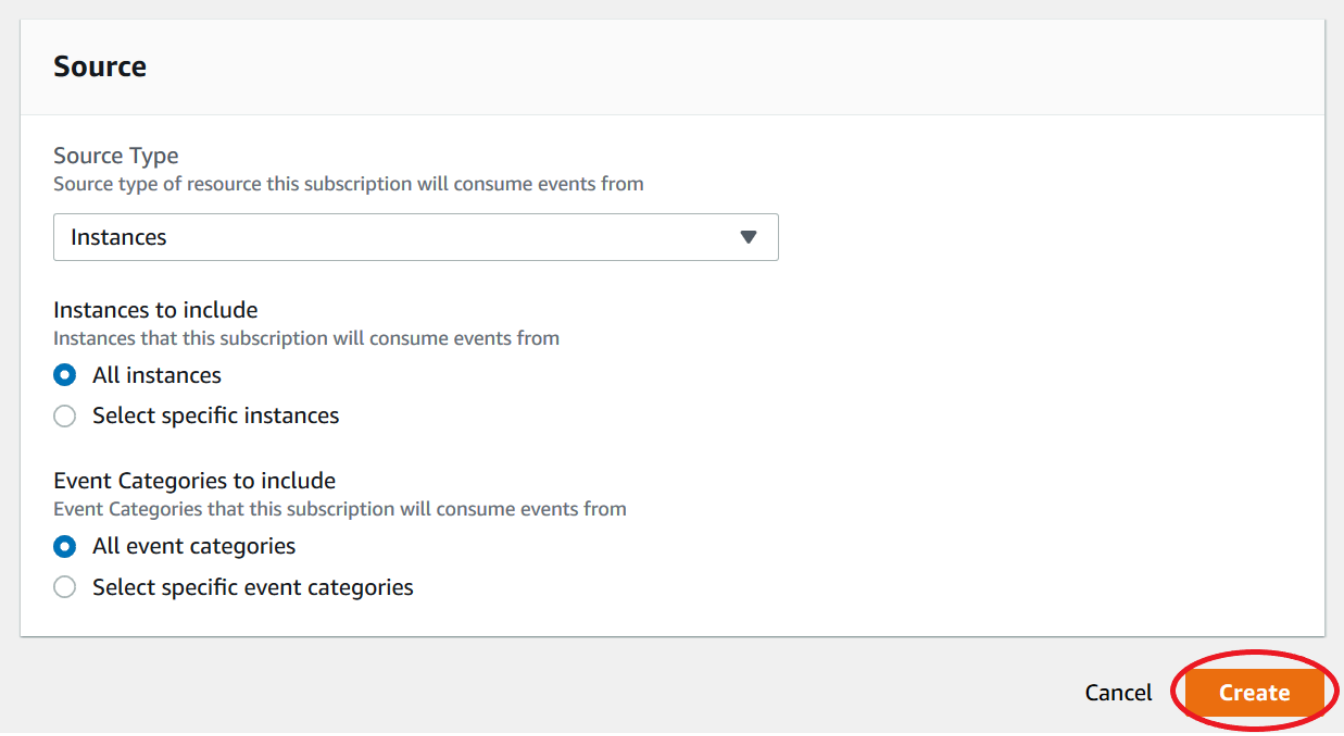
- 대상에서, 알림을 전송할 위치를 선택합니다. 기존의 ARN을 선택하거나 새 이메일 토픽을 선택하여 주제의 이름과 수신인 목록을 입력할 수 있습니다.



- 소스에서 소스 유형을 선택합니다. 선택한 원본 유형에 따라 이벤트 범주와 이벤트 알림을 수신할 원본을 선택합니다.



- 생성(Create)을 선택합니다.

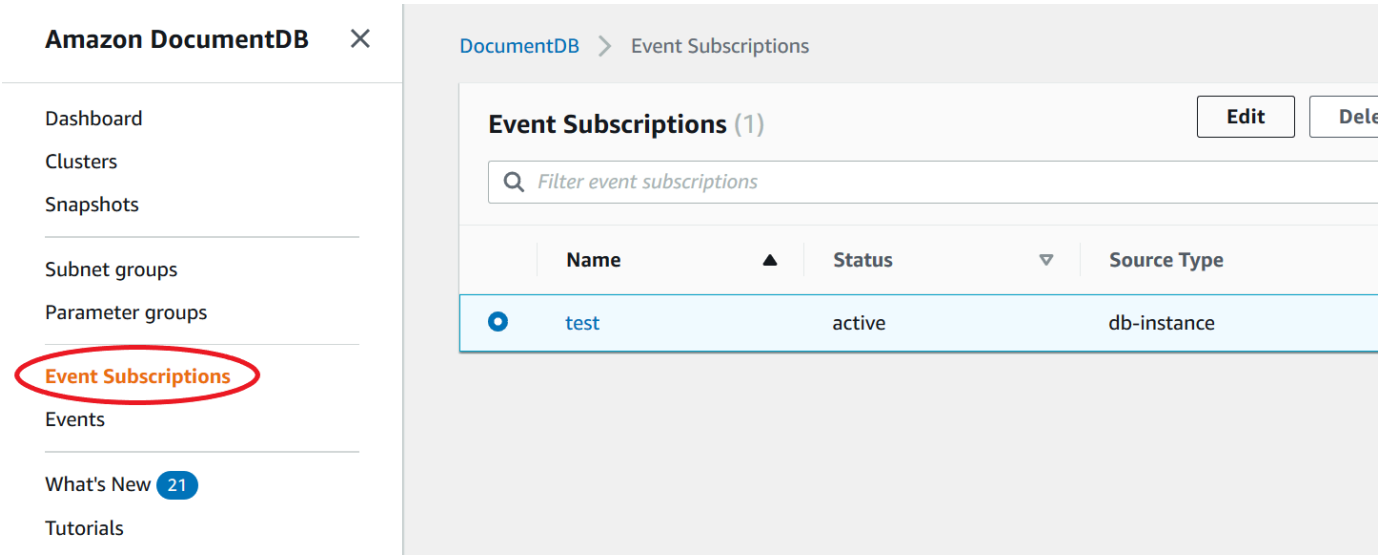


Amazon DocumentDB 이벤트 알림 구독 관리

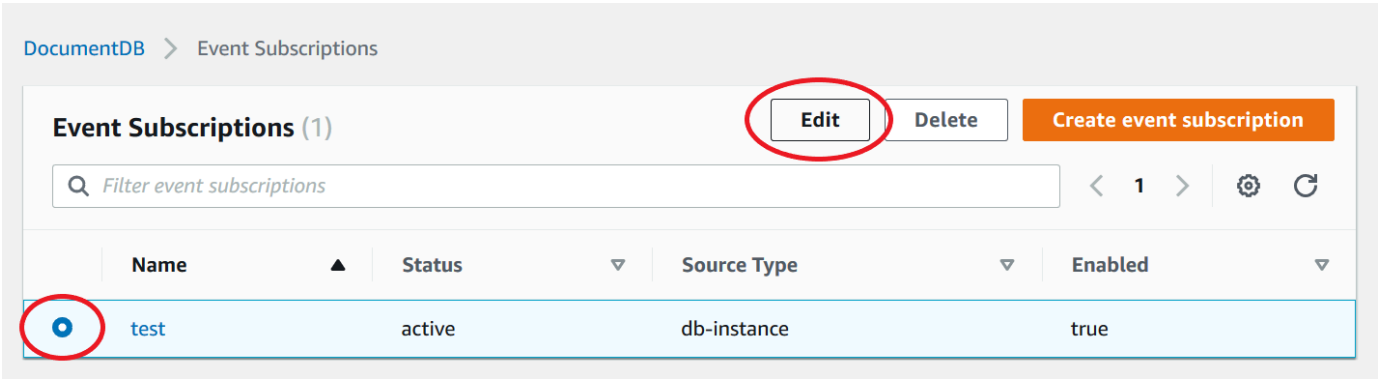
Amazon DocumentDB 콘솔의 탐색 창에서 이벤트 구독 을 선택하면 구독 범주와 현재 구독 목록을 볼 수 있습니다. 특정 구독을 수정하거나 삭제할 수도 있습니다.

현재 Amazon DocumentDB 이벤트 알림 구독을 수정하려면 다음과 같이 하세요

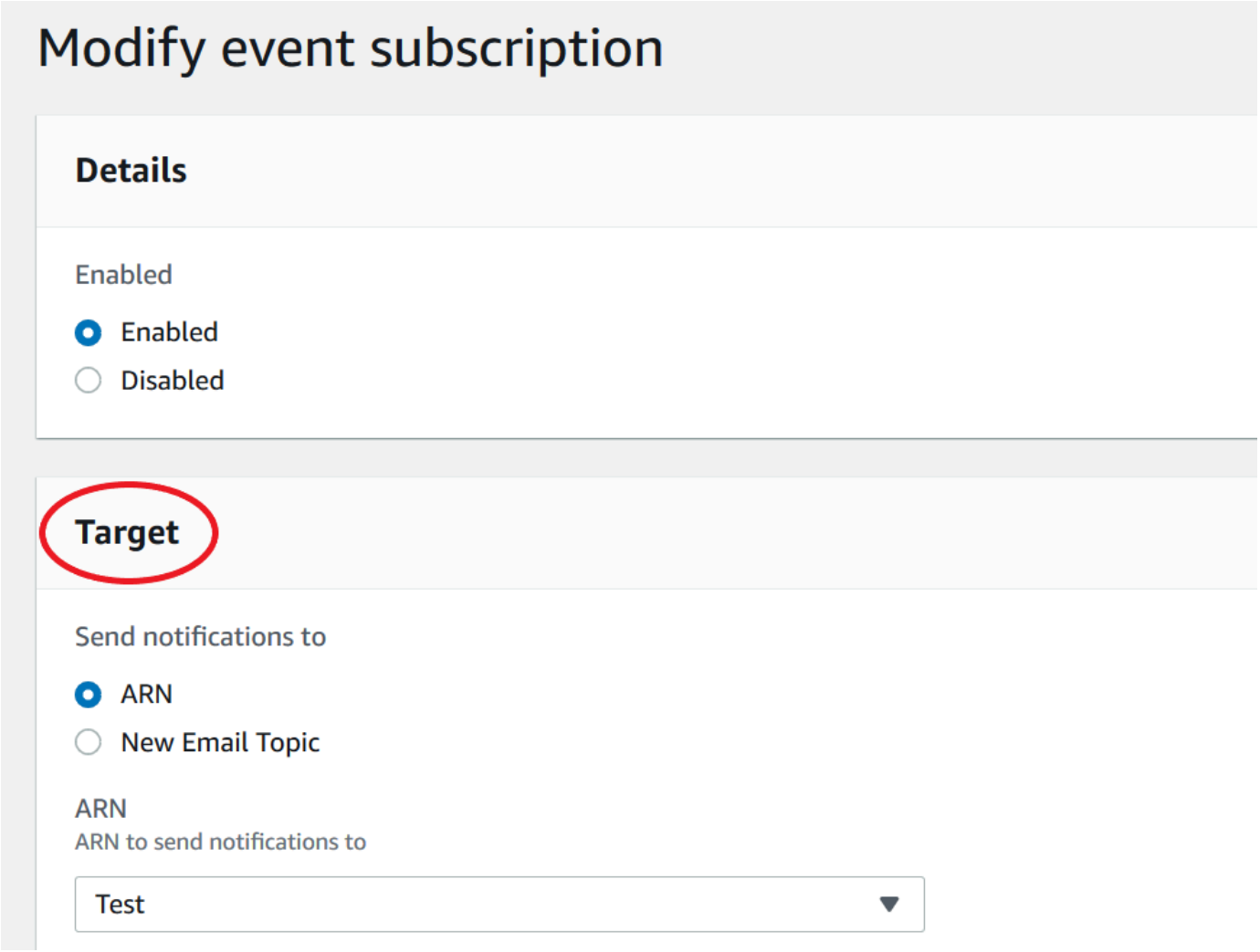
1. 에서에 로그인 AWS Management Console 합니다https://console.aws.amazon.com/docdb.
2. 탐색 창에서 [Event subscriptions]를 선택합니다. [Event subscriptions] 창에 이벤트 알림 구독이 모두 표시됩니다.



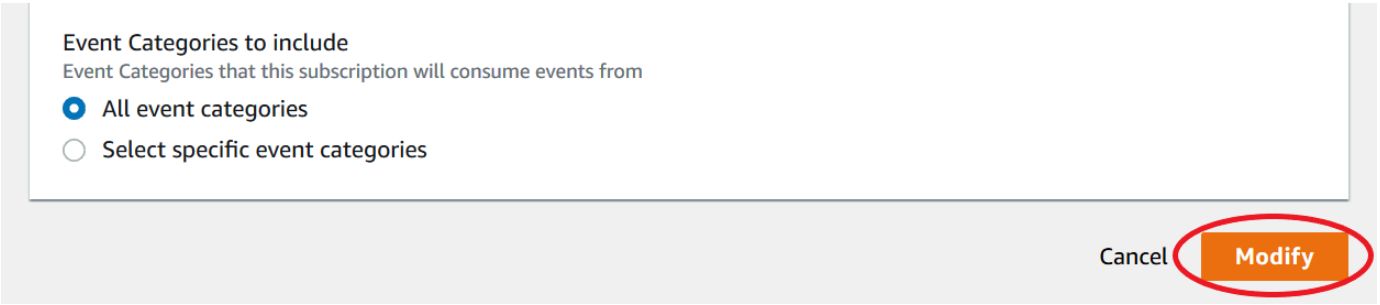
3. [Event subscriptions] 창에서 수정할 구독을 선택한 다음 [Edit]를 선택합니다.



4. 대상 또는 원본 섹션에서 구독을 변경합니다. 소스 섹션에서 소스 식별자를 선택하거나 선택 취소하여 소스 식별자를 추가하거나 제거할 수 있습니다.



5. 수정을 선택합니다. Amazon DocumentDB 콘솔은 구독이 수정되고 있음을 나타냅니다.

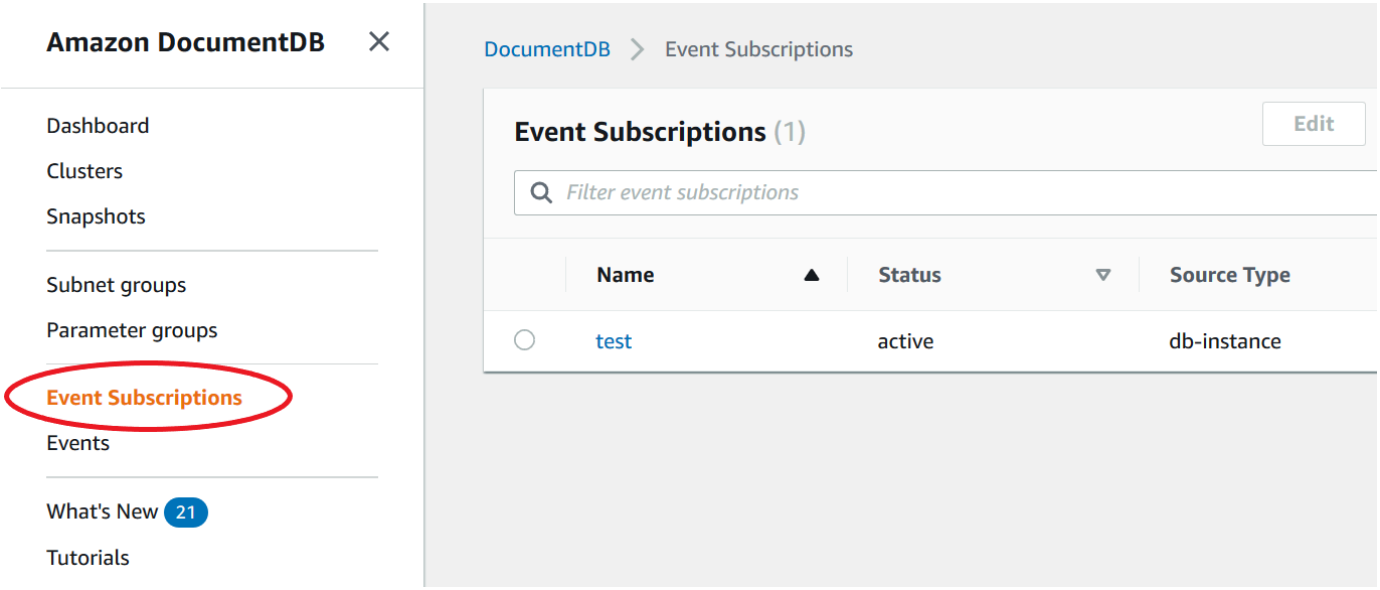


Amazon DocumentDB 이벤트 알림 구독 삭제

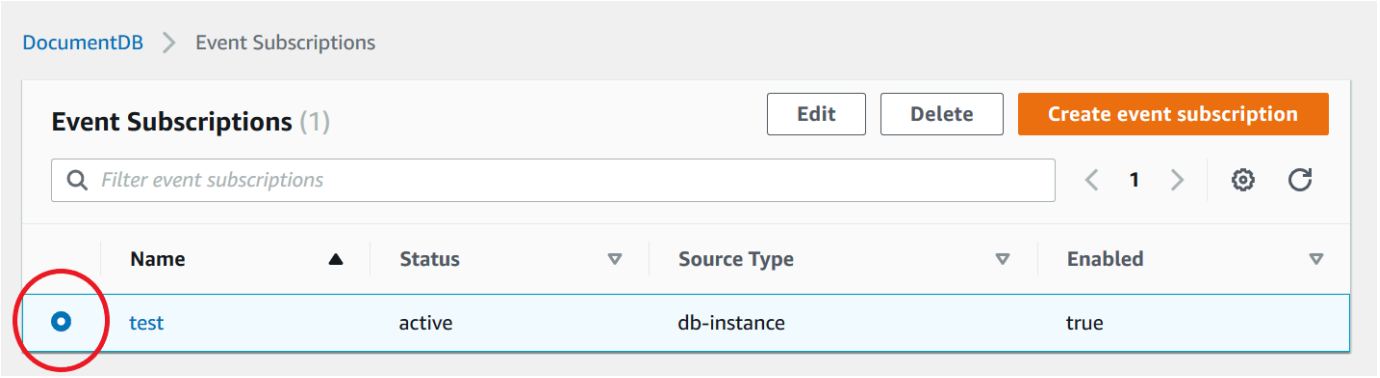
필요 없는 구독은 삭제할 수 있습니다. 그러면 해당 주제의 모든 구독자에게는 구독 시 지정한 이벤트 알림 메시지가 발송되지 않습니다.

1. 에서에 로그인 AWS Management Console 합니다https://console.aws.amazon.com/docdb.

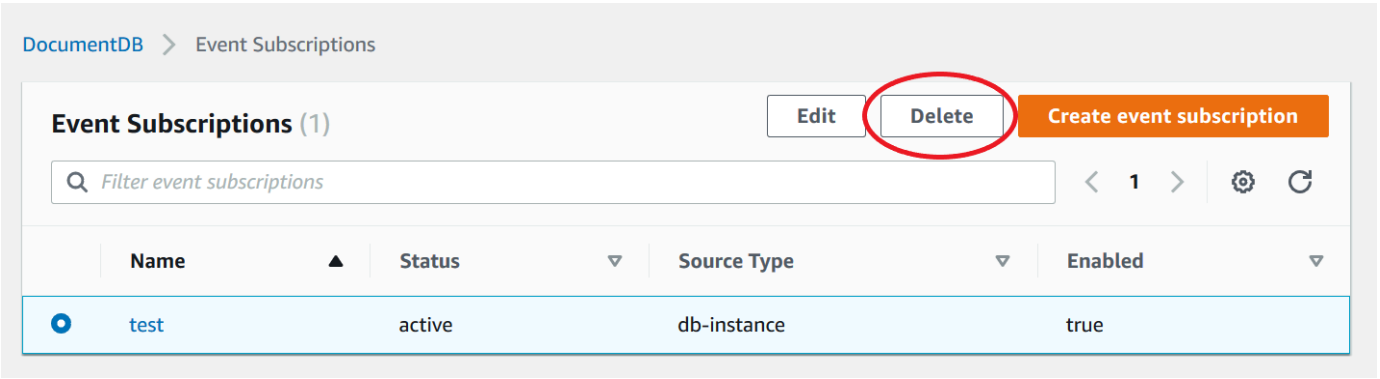
2. 탐색 창에서 [이벤트 구독]을 선택합니다.



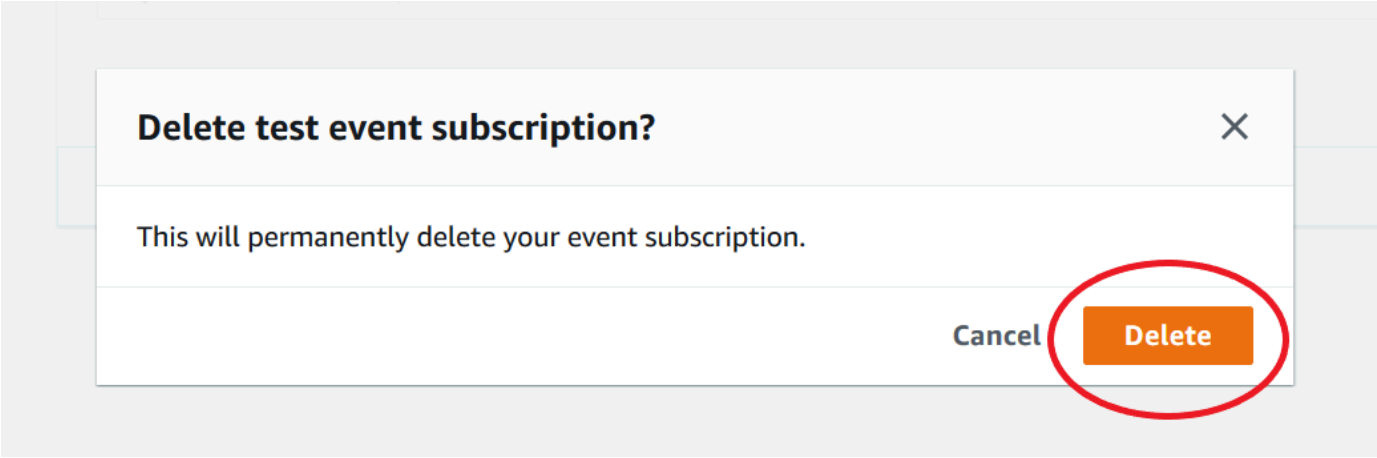
3. 이벤트 구독 패널에서 삭제할 구독을 선택합니다.



4. Delete(삭제)를 선택합니다.



5. 이 알림을 영구적으로 삭제할지 묻는 팝업 창이 나타납니다. Delete(삭제)를 선택합니다.



Amazon DocumentDB 이벤트 범주 및 메시지

Amazon DocumentDB는 콘솔을 사용하여 가입할 수 있는 카테고리에서 상당한 수의 이벤트를 생성합니다. 각 범주는 인스턴스, 클러스터, 스냅샷 또는 파라미터 그룹일 수 있는 소스 유형에 적용됩니다.

> **Note**
>
> Amazon DocumentDB는 기존의 Amazon RDS 이벤트 정의와 ID를 사용합니다.

인스턴스에서 발생한 Amazon DocumentDB 이벤트

범주	설명
가용성	인스턴스 재시작
가용성	인스턴스 셧다운.
구성 변경	인스턴스 클래스에 수정을 적용하는 중입니다.
구성 변경	인스턴스 클래스에 수정을 적용했습니다.
구성 변경	기본 자격 증명을 재설정합니다.
생성	인스턴스가 생성되었습니다.
삭제	인스턴스가 삭제되었습니다.

범주	설명
실패	호환되지 않는 구성 또는 기본 스토리지 문제로 인해 인스턴스가 실패했습니다. 인스턴스에 대한 시점 복원을 시작합니다.
알림	인스턴스 멈춤.
알림	인스턴스 시작.
알림	인스턴스가 중지된 최대 허용 시간을 초과하여 시작되고 있습니다.
복구	인스턴스 복구가 시작되었습니다. 복구 시간은 복구할 데이터 용량에 따라 달라집니다.
복구	인스턴스 복구가 완료되었습니다.
보안 패치	인스턴스에 대해 운영 체제 업데이트를 사용할 수 있습니다. 업데이트 적용에 대한 자세한 내용은 Amazon DocumentDB 관리를 참조하세요.

클러스터에서 발생하는 Amazon DocumentDB 이벤트

범주	설명
생성	클러스터가 생성됨.
삭제	클러스터가 삭제되었습니다.
장애 조치	이전 프라이머리를 다시 추진합니다.
장애 조치	인스턴스로의 장애 조치가 완료되었습니다.
장애 조치	DB 인스턴스에 대한 장애 조치 시작: %초
장애 조치	DB 인스턴스에 대한 동일한 AZ 장애 조치 시작: %초

범주	설명
장애 조치	DB 인스턴스에 대한 동일한 AZ 장애 조치 시작: %초
유지 관리	클러스터가 패치되었습니다.
유지 관리	데이터베이스 클러스터를 업그레이드할 수 없는 상태: %초
알림	클러스터가 중지되었습니다.
알림	클러스터가 시작되었습니다.
알림	클러스터 중지에 실패했습니다.
알림	클러스터가 중지되는 최대 허용 시간을 초과하여 시작되고 있습니다.
알림	클러스터 이름이 %초에서 %초로 변경되었습니다.

클러스터 스냅샷에서 발생한 Amazon DocumentDB 이벤트

다음 표는 Amazon DocumentDB 클러스터 스냅샷이 원본 유형인 경우 이벤트 범주와 이벤트 목록을 보여줍니다.

범주	설명
백업	수동 클러스터 스냅샷을 생성하는 중입니다.
백업	수동 클러스터 스냅샷이 생성되었습니다.
백업	자동화된 클러스터 스냅샷을 생성하는 중입니다.
백업	자동화된 클러스터 스냅샷이 생성되었습니다.

파라미터 그룹에서 발생하는 Amazon DocumentDB 이벤트

다음 표는 파라미터 그룹이 소스 유형일 때 이벤트 카테고리와 이벤트 목록을 나타냅니다.

범주	설명
구성 변경	적용 방법 %초를 사용하여 매개 변수 %초를 %초로 업데이트했습니다

CloudWatch를 사용하여 Amazon DocumentDB 모니터링

Amazon DocumentDB(MongoDB 호환)는 Amazon CloudWatch와 통합되므로 클러스터의 운영 지표를 수집하고 분석할 수 있습니다. CloudWatch 콘솔, Amazon DocumentDB 콘솔, AWS Command Line Interface (AWS CLI) 또는 CloudWatch API를 사용하여 이러한 지표를 모니터링할 수 있습니다.

또한 CloudWatch를 사용하면 지표 값이 지정한 임계값을 위반할 경우 알림을 받을 수 있도록 알람을 설정할 수 있습니다. 위반이 발생할 경우 수정 조치를 취하기 위해 Amazon CloudWatch Events를 설정할 수도 있습니다. CloudWatch 및 알람 사용에 대한 자세한 내용은 Amazon CloudWatch 설명서를 참조하세요.

주제

- Amazon DocumentDB 지표
- CloudWatch 데이터 보기
- Amazon DocumentDB 차원
- Opcounter 지표 모니터링
- 데이터베이스 연결 모니터링

Amazon DocumentDB 지표

Amazon DocumentDB 클러스터 및 인스턴스의 상태 및 성능을 모니터링하려면 Amazon DocumentDB 콘솔에서 다음 지표를 볼 수 있습니다.

> **Note**
>
> 다음 표의 지표는 인스턴스 기반 클러스터와 탄력적 클러스터 모두에 적용됩니다.

주제

- 리소스 사용률 지표
- 지연 시간 지표
- NVMe 지원 인스턴스 지표
- 작업 지표
- 처리량 지표
- 시스템 지표
- T3 인스턴스 지표

리소스 사용률 지표

지표	설명
`BackupRetentionPeriodStorageUsed`	Amazon DocumentDB의 보존 기간 내에서 point-in-time 복원 기능을 지원하는 데 사용되는 바이트 단위의 총 백업 스토리지 양입니다. `TotalBackupStorageBilled` 지표를 통해 보고되는 총계에 포함됩니다. 각 Amazon DocumentDB 클러스터마다 개별적으로 계산됩니다.
`ChangeStreamLogSize`	클러스터에서 변경 스트림 로그를 저장하는 데 사용하는 스토리지의 양(MB)입니다. 이 값은 클러스터의 전체 스토리지 하위 집합(`VolumeBytesUsed`)이며 클러스터 비용에 영향을 줍니다. 스토리지 요금 정보는 Amazon DocumentDB 제품 페이지를 참조하세요. 변경 스트림 로그 크기는 클러스터에서 발생하

지표	설명
	는 변경 양과 변경 스트림 장기 보존 기간의 함수입니다. 변경 스트림에 대한 자세한 내용은 Amazon DocumentDB에서 변경 스트림 사용 섹션을 참조하세요.
CPUUtilization	인스턴스의 CPU 사용률
DatabaseConnections	1분 간격으로 인스턴스에서 열린 연결 수입니다.
DatabaseConnectionsMax	1분 동안 인스턴스에서 열린 데이터베이스 연결의 최대 수입니다.
DatabaseCursors	1분 간격으로 인스턴스에서 열린 커서 수입니다.
DatabaseCursorsMax	1분 동안 인스턴스에서 열린 최대 커서 수입니다.
DatabaseCursorsTimedOut	1분 동안 시간 초과된 커서 수입니다.
FreeableMemory	사용 가능한 RAM 크기(바이트).
FreeLocalStorage	이 지표는 각 인스턴스에서 임시 테이블 및 로그로 사용할 수 있는 스토리지 크기를 보고합니다. 이 값은 인스턴스 클래스에 따라 달라집니다. 인스턴스 클래스를 큰 것으로 선택하면 인스턴스의 여유 스토리지 공간을 늘릴 수 있습니다.

지표	설명
LowMemThrottleQueueDepth	사용 가능한 메모리가 부족하여 1분 간격으로 제한된 요청의 대기열 깊이입니다.
LowMemThrottleMaxQueueDepth	1분 동안 사용 가능한 메모리가 부족하여 제한되는 요청의 최대 대기열 깊이입니다.
LowMemNumOperationsThrottled	1분 동안 사용 가능한 메모리 부족으로 인해 제한되는 요청 수입니다.
SnapshotStorageUsed	백업 보존 기간 외부의 지정된 Amazon DocumentDB 클러스터에 대한 모든 스냅샷에서 사용된 바이트 단위의 총 백업 스토리지 양입니다. TotalBackupStorageBilled 지표를 통해 보고되는 총계에 포함됩니다. 각 Amazon DocumentDB 클러스터마다 개별적으로 계산됩니다.
SwapUsage	인스턴스에서 사용된 스왑 공간 크기

지표	설명
TotalBackupStorageBilled	지정된 Amazon DocumentDB 클러스터에 대해 청구되는 바이트 단위의 총 백업 스토리지 양입니다. BackupRetentionPeriodStorageUsed 및 SnapshotStorageUsed 지표로 측정되는 백업 스토리지를 포함합니다. 각 Amazon DocumentDB 클러스터마다 개별적으로 계산됩니다.
TransactionsOpen	1분 간격으로 인스턴스에서 열린 트랜잭션 수입니다.
TransactionsOpenMax	1분 동안 인스턴스에서 열린 최대 트랜잭션 수입니다.
VolumeBytesUsed	클러스터가 사용하는 스토리지의 양(바이트)입니다. 이 값은 클러스터의 비용에 영향을 미칩니다. 요금 정보는 Amazon DocumentDB 제품 페이지를 참조하세요.

지연 시간 지표

지표	설명
DBClusterReplicaLagMaximum	클러스터의 기본 인스턴스와 각 Amazon DocumentDB 인스턴스 간의 최대 지연 시간(밀리초)입니다.

지표	설명	
DBClusterReplicaLagMinimum	기본 인스턴스와 클러스터의 각 복제본 인스턴스 사이에 발생하는 최소 지연 시간(밀리초)	
DBInstanceReplicaLag	기본 인스턴스에서 복제본 인스턴스로 업데이트를 복제할 때의 지연 시간(밀리초)	
ReadLatency	디스크 I/O 연산당 평균 처리 시간입니다.	
WriteLatency	디스크 I/O 연산당 평균 처리 시간(밀리초)	

NVMe 지원 인스턴스 지표

지표	설명	
NVMeStorageCacheHitRatio	계층형 캐시에서 제공하는 요청의 백분율입니다.	
FreeNVMeStorage	사용 가능한 임시 NVMe 스토리지 크기입니다.	
ReadIOPSNVMeStorage	임시 NVMe 스토리지에 대한 평균 디스크 읽기 I/O 작업 수입니다.	
ReadLatencyNVMeStorage	임시 NVMe 스토리지의 디스크 읽기 I/O 작업당 소요된 평균 시간입니다.	
ReadThroughputNVMeStorage	임시 NVMe 스토리지의 초당 디스크에서 읽은 평균 바이트 수입니다.	

지표	설명
WriteIOPSNVMeStorage	임시 NVMe 스토리지에 대한 평균 디스크 쓰기 I/O 작업 수입니다.
WriteLatencyNVMeStorage	임시 NVMe 스토리지의 디스크 쓰기 I/O 작업당 소요된 평균 시간입니다.
WriteThroughputNVMeStorage	임시 NVMe 스토리지에 디스크에 쓰여진 초당 평균 바이트 수입니다.

작업 지표

지표	설명
DocumentsDeleted	1분 동안 삭제된 문서 수입니다.
DocumentsInserted	1분 동안 삽입된 문서 수입니다.
DocumentsReturned	1분 동안 반환된 문서 수입니다.
DocumentsUpdated	1분 동안 업데이트된 문서 수입니다.
OpcountersCommand	1분 동안 실행된 명령 수입니다.
OpcountersDelete	1분 동안 실행된 삭제 작업 수입니다.
OpcountersGetmore	1분 동안 발급된 getmore 수입니다.

지표	설명
OpcountersInsert	1분 동안 실행된 삽입 작업 수입니다.
OpcountersQuery	1분 동안 실행된 쿼리 수입니다.
OpcountersUpdate	1분 동안 실행된 업데이트 작업 수입니다.
TransactionsStarted	1분 동안 인스턴스에서 시작된 트랜잭션 수입니다.
TransactionsCommitted	1분 동안 인스턴스에 커밋된 트랜잭션 수입니다.
TransactionsAborted	1분 동안 인스턴스에서 중단된 트랜잭션 수입니다.
TTLDeletedDocuments	1분 동안 TTLMonitor에서 삭제한 문서 수입니다.

처리량 지표

지표	설명
NetworkReceiveThroughput	클러스터의 인스턴스 하나가 클라이언트에서 수신하는 네트워크 처리량(bps) 이 처리량에서 클러스터의 인스턴스와 클러스터 볼륨 간 네트워크 트래픽은 제외됩니다.
NetworkThroughput	Amazon DocumentDB 클러스터의 각 인스턴스에서 클라이언트에서 수신 및 클라이언트로 전송된 네트워크 처리량(초

지표	설명
	당 바이트)입니다. 이 처리량에서 클러스터의 인스턴스와 클러스터 볼륨 간 네트워크 트래픽은 제외됩니다.
NetworkTransmitThroughput	클러스터의 인스턴스 하나가 클라이언트로 전송하는 네트워크 처리량(bps). 이 처리량에서 클러스터의 인스턴스와 클러스터 볼륨 간 네트워크 트래픽은 제외됩니다.
ReadIOPS	초당 평균 디스크 읽기 I/O 연산 수 Amazon DocumentDB는 읽기 및 쓰기 IOPS를 1분 간격으로 별도로 보고합니다.
ReadThroughput	초당 디스크에서 읽은 평균 바이트 수입니다.
StorageNetworkReceiveThroughput	클러스터의 각 인스턴스에 의해 Amazon DocumentDB 클러스터 스토리지 볼륨에서 수신한 네트워크 처리량(초당 바이트)입니다.
StorageNetworkTransmitThroughput	클러스터의 각 인스턴스에 의해 Amazon DocumentDB 클러스터 스토리지 볼륨에서 전송한 네트워크 처리량(초당 바이트)입니다.

지표	설명
StorageNetworkThroughput	Amazon DocumentDB 클러스터의 각 인스턴스에 의해 Amazon DocumentDB 클러스터 스토리지 볼륨에서 수신 및 전송한 네트워크 처리량(초당 바이트)입니다.
VolumeReadIOPs	5분마다 보고되는 클러스터 볼륨에서 요금이 청구된 읽기 I/O 작업의 평균 수. 요금이 청구된 읽기 작업은 클러스터 볼륨 수준에서 계산되며, 클러스터의 모든 인스턴스에 대해 집계된 후 5분 간격으로 보고됩니다. 이 값은 5분 동안의 읽기 작업 측정치 값을 사용하여 계산됩니다. 요금 부과된 읽기 작업 측정치의 값을 300로 나눠서 초당 요금 부과된 읽기 작업의 양을 확인할 수 있습니다. 예를 들어, VolumeReadIOPs 이 13,686을 반환하면 초당 청구된 읽기 작업은 45입니다(13,686 / 300 = 45.62). 버퍼 캐시에 없는 데이터베이스 페이지를 요청하는 쿼리에 대해서는 요금 부과된 읽기 작업이 발생하므로 스토리지에서 로드해야 합니다. 쿼리 결과를 스토리지에서 읽은 후 버퍼 캐시로 로드하면 요금 부과된 작업이 급증할 수 있습니다.

지표	설명	
VolumeWriteIOPs	5분마다 보고되는 클러스터 볼륨에서 요금이 청구된 쓰기 I/O 작업의 평균 수. 요금이 청구된 쓰기 작업은 클러스터 볼륨 수준에서 계산되며, 클러스터의 모든 인스턴스에 대해 집계된 후 5분 간격으로 보고됩니다. 이 값은 5분 동안의 쓰기 작업 측정치 값을 사용하여 계산됩니다. 요금이 청구된 쓰기 작업 측정치의 값을 300초로 나눠서 초당 요금이 청구된 쓰기 작업의 양을 확인할 수 있습니다. 예를 들어 VolumeWriteIOPs 이 13,686을 반환하면 초당 청구된 쓰기 작업은 45입니다(13,686 / 300 = 45.62). 참고로 VolumeReadIOPs 및 VolumeWriteIOPs 지표는 DocumentDB 스토리지 계층에서 계산되며 여기에는 기본 및 복제 인스턴스에서 수행한 IO가 포함됩니다. 데이터는 20-30분마다 집계된 후 5분 간격으로 보고되므로 일정 기간 동안 지표에 대해 동일한 데이터 지점이 생성됩니다. 1분 간격으로 삽입 작업과 상호 연관시킬 지표를 찾으려면 인스턴스 수준 WriteiOps 측정치를 사용할 수 있습니다. 지표는 Amazon DocumentDB 기본 인스턴스의	

지표	설명
	모니터링 탭에서 사용할 수 있습니다.
WriteIOPS	초당 평균 디스크 쓰기 I/O 연산 수 클러스터 수준에서 사용할 경우 WriteIOPs 은 클러스터의 모든 인스턴스에서 평가됩니다. 1분의 간격을 두고 읽기 IOPS와 쓰기 IOPS를 따로 보고합니다.
WriteThroughput	초당 디스크에 쓴 평균 바이트 수.

시스템 지표

지표	설명
BufferCacheHitRatio	버퍼 캐시에서 처리하는 요청 비율입니다.
DiskQueueDepth	디스크에 쓰거나 디스크에서 읽기를 기다리는 I/O 작업 수입니다.
EngineUptime	인스턴스 실행 시간(초)
IndexBufferCacheHitRatio	버퍼 캐시에서 제공되는 인덱스 요청의 백분율입니다. 인덱스, 컬렉션 또는 데이터베이스를 삭제한 직후 지표에 대해 100%를 초과하는 스파이크가 표시될 수 있습니다. 이는 60초 후에 자동으로 수정됩니다. 향후 패치 업데이트에서 이러한

지표	설명	
	제한 사항이 수정될 예정입니다.	

T3 인스턴스 지표

지표	설명	
`CPUCreditUsage`	측정 기간 동안 사용된 CPU 크레딧 수.	
`CPUCreditBalance`	인스턴스에서 발생한 CPU 크레딧 수. CPU에서 버스트가 발생하고 CPU 크레딧이 획득 속도보다 빠르게 소비될 때 크레딧 밸런스가 고갈됩니다.	
`CPUSurplusCreditBalance`	CPUCreditBalance 값이 0일 때 CPU 성능을 유지하기 위해 사용된 잉여 CPU 크레딧의 수입니다.	
`CPUSurplusCreditsCharged`	잉여 CPU 크레딧 수가 24시간 동안 얻을 수 있는 최대 CPU 크레딧 수를 초과하므로 추가 요금이 발생합니다. 자세한 내용은 CPU 크레딧 모니터링을 참조하세요.	

CloudWatch 데이터 보기

CloudWatch 콘솔, Amazon DocumentDB 콘솔, AWS Command Line Interface (AWS CLI) 또는 CloudWatch API를 사용하여 Amazon CloudWatch 데이터를 볼 수 있습니다. Amazon DocumentDB

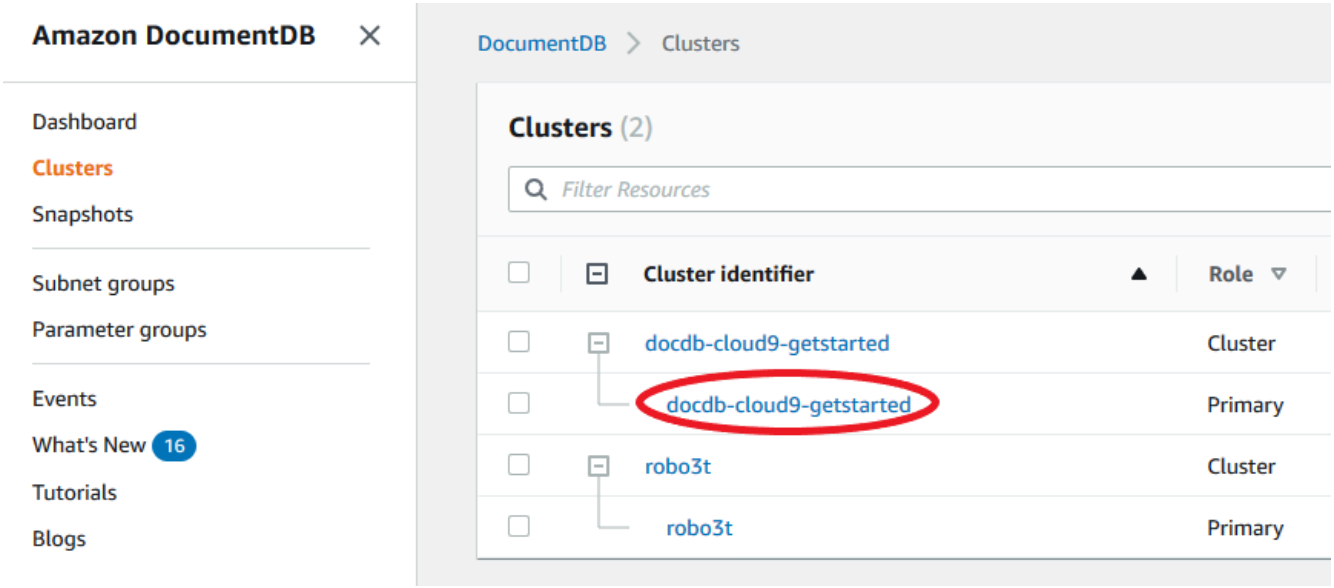
Using the AWS Management Console

Amazon DocumentDB 관리 콘솔을 사용하여 CloudWatch 지표를 보려면 다음 단계를 완료합니다.

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 클러스터를 선택합니다.

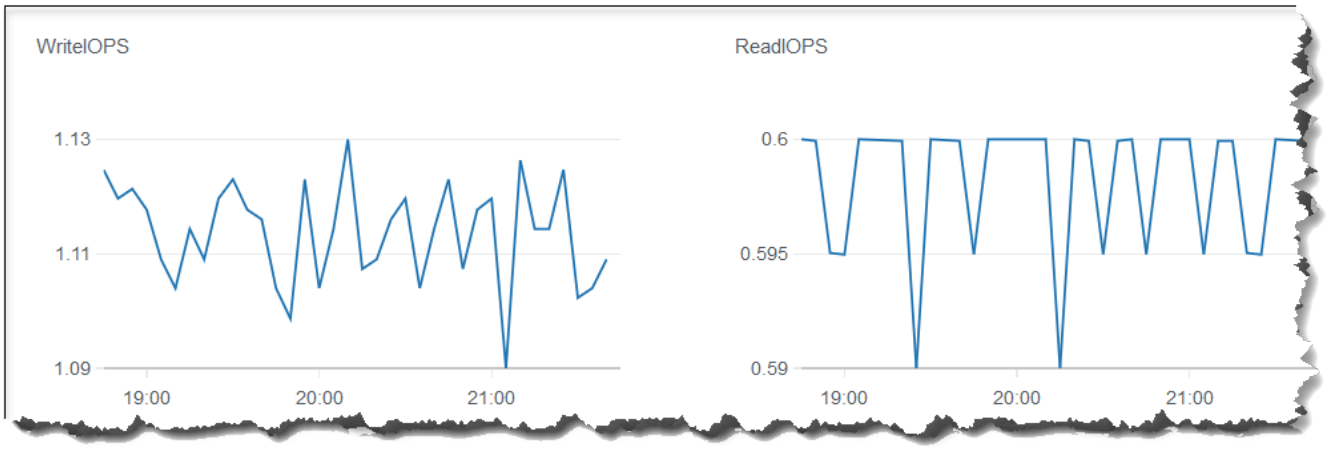
   > **Tip**
   >
   > 화면 왼쪽에 탐색 창이 표시되지 않으면 페이지 왼쪽 상단 모서리에서 메뉴 아이콘 (≡) 을 선택합니다.

3. 클러스터 탐색 상자에 클러스터 식별자 열이 표시됩니다. 인스턴스는 아래 스크린샷과 마찬가지로 클러스터 아래에 나열됩니다.



4. 인스턴스 목록에서 지표를 사용할 인스턴스의 이름을 선택합니다.
5. 결과 인스턴스 요약 페이지에서 모니터링 탭을 선택하여 Amazon DocumentDB 인스턴스의 지표를 그래픽으로 표시합니다. 각 지표에 대한 그래픽이 생성되기 전에 CloudWatch 그래프를 채우는 데 몇 분 정도 걸릴 수 있습니다.

   다음 이미지는 Amazon DocumentDB 콘솔, `WriteIOPS` 및 `ReadIOPS`에 있는 두 개의 CloudWatch 지표를 그래픽으로 표현한 것입니다.



Using the CloudWatch Management Console

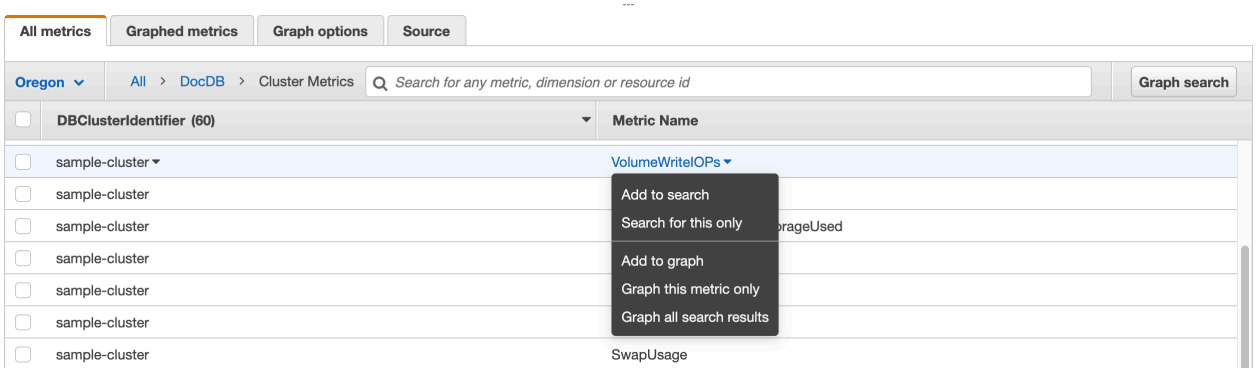
CloudWatch 관리 콘솔을 사용하여 CloudWatch 지표를 보려면 다음 단계를 수행합니다.

1. 에 로그인 AWS Management Console하고에서 Amazon DocumentDB 콘솔을 엽니다https://console.aws.amazon.com/cloudwatch.
2. 탐색 창에서 지표(Metrics)를 선택합니다. 그런 다음 서비스 이름 목록에서 DocDB를 선택합니다.
3. 지표 차원(예: 클러스터 지표)을 선택합니다.
4. 모든 지표 탭은 DocDB에서 해당 차원의 모든 지표를 표시합니다.

   a. 테이블을 정렬하려면 열 머리글을 사용합니다.

   b. 측정치를 그래프로 표시하려면 측정치 옆에 있는 확인란을 선택합니다. 모든 지표를 선택하려면 테이블의 머리글 행에 있는 확인란을 선택합니다.

   c. 지표를 기준으로 필터링하려면 지표 이름 위로 마우스를 가져가 지표 이름 옆의 드롭다운 화살표를 선택합니다. 그런 다음 아래 이미지와 같이 검색에 추가를 선택합니다.



Using the AWS CLI

Amazon DocumentDB용 CloudWatch 데이터를 보려면 다음 파라미터를 사용하여 CloudWatch `get-metric-statistics` 작업을 수행합니다.

파라미터

- **`--namespace`** - 필수입니다. CloudWatch 지표를 원하는 서비스 네임스페이스. Amazon DocumentDB의 경우, 이것은 `AWS/DocDB`이어야 합니다.
- **`--metric-name`** - 필수입니다. 데이터를 원하는 지표의 이름.
- **`--start-time`** - 필수입니다. 반환할 첫 번째 데이터 포인트를 결정하는 타임스탬프입니다.

  지정된 값이 포함되므로, 지정된 타임스탬프를 가진 데이터 포인트가 결과에 포함됩니다. 타임스탬프는 ISO 8601 UTC 형식(예: 2016-10-03T23:00:00Z)이어야 합니다.
- **`--end-time`** - 필수입니다. 반환할 마지막 데이터 포인트를 결정하는 타임스탬프입니다.

  지정된 값이 포함되므로, 지정된 타임스탬프를 가진 데이터 포인트가 결과에 포함됩니다. 타임스탬프는 ISO 8601 UTC 형식(예: 2016-10-03T23:00:00Z)이어야 합니다.
- **`--period`** - 필수입니다. 반환되는 데이터 포인트의 세부 수준(초)입니다. 일반 분해능 지표의 경우 기간은 최소 1분(60초)이고 60의 배수여야 합니다. 1분 미만의 간격으로 수집되는 고분해능 지표의 경우 기간은 1, 5, 10, 30, 60 또는 60의 배수입니다.
- **`--dimensions`** - 선택 사항. 지표에 여러 차원이 포함된 경우 각 차원에 대한 값을 포함해야 합니다. CloudWatch는 각각의 고유한 차원의 조합을 별도의 지표로 처리합니다. 특정 차원 조합이 게시되지 않은 경우 해당 조합에 대한 통계를 검색할 수 없습니다. 지표 생성 시 사용된 것과 동일한 차원을 지정해야 합니다.
- **`--statistics`** - 선택 사항. 백분위수 이외의 지표 통계입니다. 백분위수 통계의 경우 `ExtendedStatistics` 섹션을 참조하세요. `GetMetricStatistics`를 호출할 경우 `Statistics` 또는 `ExtendedStatistics` 중 하나만 지정해야 합니다.

  허용되는 값:

  - `SampleCount`
  - `Average`
  - `Sum`
  - `Minimum`
  - `Maximum`

- **--extended-statistics** - 선택 사항. percentile 통계입니다. p0.0~p100 사이의 값을 지정합니다. GetMetricStatistics를 호출할 경우 Statistics 또는 ExtendedStatistics 중 하나만 지정해야 합니다.
- **--unit** - 선택 사항. 주어진 지표의 단위입니다. 지표가 여러 단위로 보고될 수 있습니다. 단위 결과가 반환 중인 모든 단위로 제공되지 않습니다. 지표가 보고되지 않는 단위만 지정한 경우 호출 결과는 null입니다.

  가능한 값은 다음과 같습니다.

  - Seconds
  - Microseconds
  - Milliseconds
  - Bytes
  - Kilobytes
  - Megabytes
  - Gigabytes
  - Terabytes
  - Bits
  - Kilobytes
  - Megabits
  - Gigabits
  - Terabits
  - Percent
  - Count
  - Bytes/Second
  - Kilobytes/Second
  - Megabytes/Second
  - Gigabytes/Second
  - Terabytes/Second
  - Bits/Second
  - Kilobits/Second
  - Megabits/Second

- Gigabits/Second
- Terabits/Second
- Count/Second
- None

Example

다음 예에서는 60초마다 샘플을 생성하여 2시간 기간에 대한 최대 CPUUtilization을 찾습니다.

Linux, macOS, Unix의 경우:

```
aws cloudwatch get-metric-statistics \
       --namespace AWS/DocDB \
       --dimensions \
           Name=DBInstanceIdentifier,Value=docdb-2019-01-09-23-55-38 \
       --metric-name CPUUtilization \
       --start-time 2019-02-11T05:00:00Z \
       --end-time 2019-02-11T07:00:00Z \
       --period 60 \
       --statistics Maximum
```

Windows의 경우:

```
aws cloudwatch get-metric-statistics ^
       --namespace AWS/DocDB ^
       --dimensions ^
           Name=DBInstanceIdentifier,Value=docdb-2019-01-09-23-55-38 ^
       --metric-name CPUUtilization ^
       --start-time 2019-02-11T05:00:00Z ^
       --end-time 2019-02-11T07:00:00Z ^
       --period 60 ^
       --statistics Maximum
```

이 작업의 출력은 다음과 같습니다.

```
{
       "Label": "CPUUtilization",
       "Datapoints": [
           {
```

```
            "Unit": "Percent",
            "Maximum": 4.49152542374361,
            "Timestamp": "2019-02-11T05:51:00Z"
        },
        {
            "Unit": "Percent",
            "Maximum": 4.25000000000485,
            "Timestamp": "2019-02-11T06:44:00Z"
        },

        ********* some output omitted for brevity *********

        {
            "Unit": "Percent",
            "Maximum": 4.33333333331878,
            "Timestamp": "2019-02-11T06:07:00Z"
        }
    ]
}
```

Amazon DocumentDB 차원

Amazon DocumentDB에 대한 지표는 계정 또는 작업에 대한 값으로 지정됩니다. CloudWatch 콘솔을 사용하여 다음 표의 차원 중 하나로 필터링된 Amazon DocumentDB 데이터를 검색할 수 있습니다.

차원	설명
`DBClusterIdentifier`	특정 Amazon DocumentDB 클러스터에 대해 요청하는 데이터를 필터링합니다.
`DBClusterIdentifier, Role`	특정 Amazon DocumentDB 클러스터에 대해 요청하는 데이터를 필터링하여 인스턴스 역할별 지표(쓰기/읽기)를 집계합니다. 예를 들어 클러스터에 속하는 모든 READER 인스턴스에 대한 지표를 집계할 수 있습니다.
`DBInstanceIdentifier`	특정 데이터베이스 인스턴스에 대해 요청하는 데이터를 필터링합니다.

Opcounter 지표 모니터링

유휴 클러스터의 경우 Opcounter 지표는 값이 0이 아닙니다 (보통 ~50). 이는 Amazon DocumentDB가 주기적인 상태 점검, 내부 작업 및 지표 수집 작업을 수행하기 때문입니다.

데이터베이스 연결 모니터링

`db.runCommand( { serverStatus: 1 })`과 같은 데이터베이스 엔진 명령을 사용하여 연결 수를 볼 때 CloudWatch를 통해 `DatabaseConnections`에 표시된 것보다 최대 10개 더 많은 연결을 볼 수 있습니다. 이것은 Amazon DocumentDB가 `DatabaseConnections`에 설명되지 않는 주기적인 상태 검사 및 지표 수집 작업을 수행하기 때문에 발생합니다. `DatabaseConnections`는 고객 시작 연결만 나타냅니다.

를 사용하여 Amazon DocumentDB API 호출 로깅 AWS CloudTrail

Amazon DocumentDB(MongoDB 호환)는 Amazon DocumentDB(MongoDB 호환)에서 사용자, 역할 또는 서비스가 수행한 작업에 대한 레코드를 AWS CloudTrail제공하는 AWS 서비스와 통합됩니다. CloudTrail은 Amazon DocumentDB 콘솔의 호출 및 Amazon DocumentDB SDK에 대한 코드 호출을 포함하여 Amazon DocumentDB에 대한 모든 AWS CLI API 호출을 이벤트로 캡처합니다. 추적을 생성하는 경우 Amazon DocumentDB에 대한 이벤트를 포함하여 CloudTrail 이벤트를 Amazon S3 버킷에 지속적으로 전달할 수 있습니다. 추적을 구성하지 않은 경우에도 CloudTrail 콘솔의 이벤트 기록에서 최신 이벤트를 볼 수 있습니다. CloudTrail에서 수집한 정보를 사용하여 Amazon DocumentDB(MongoDB 호환)에 대한 요청, 요청이 이루어진 IP 주소, 요청자, 요청이 이루어진 시기 및 기타 세부 정보를 확인할 수 있습니다.

> **Important**
>
> 일부 관리 기능의 경우 Amazon DocumentDB는 Amazon Relational Database Service(RDS)와 공유되는 운영 기술을 사용합니다. Amazon DocumentDB 콘솔 AWS CLI및 API 호출은 Amazon RDS API에 대한 호출로 로깅됩니다.

자세한 내용은 AWS CloudTrail 사용 설명서를 AWS CloudTrail참조하세요.

CloudTrail의 Amazon DocumentDB 정보

CloudTrail은 계정을 생성할 AWS 계정 때에서 활성화됩니다. Amazon DocumentDB(MongoDB 호환)에서 활동이 발생하면 해당 활동이 이벤트 기록의 다른 AWS 서비스 이벤트와 함께 CloudTrail 이벤트

에 기록됩니다. 에서 최근 이벤트를 보고 검색하고 다운로드할 수 있습니다 AWS 계정. 자세한 정보는 CloudTrail 이벤트 기록을 사용하여 이벤트 보기를 참조하세요.

Amazon DocumentDB(MongoDB 호환)에 대한 이벤트를 AWS 계정포함하여에서 이벤트를 지속적으로 기록하려면 추적을 생성합니다. CloudTrail은 추적을 사용하여 Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 콘솔에서 추적을 생성하면 기본적으로 모든 AWS 리전에 추적이 적용됩니다. 추적은 AWS 파티션에 있는 모든 리전의 이벤트를 로깅하고 지정된 Amazon S3 버킷으로 로그 파일을 전송합니다. 또한 CloudTrail 로그에서 수집된 이벤트 데이터를 추가로 분석하고 조치를 취하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 내용은AWS CloudTrail 사용 설명서에서 다음 주제를 참조하세요.

- 트레일 생성 개요
- CloudTrail 지원 서비스 및 통합
- CloudTrail에서 Amazon SNS 알림 구성
- 여러 리전에서 CloudTrail 로그 파일 수신
- 여러 계정에서 CloudTrail 로그 파일 수신

모든 이벤트 및 로그 항목에는 요청을 생성한 사용자에 대한 정보가 들어 있습니다. 보안 인증 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청을 루트로 했는지 아니면 사용자 자격 증명으로 했는지 여부
- 역할 또는 페더레이션 사용자에 대한 임시 보안 인증을 사용하여 요청이 생성되었는지 여부.
- 요청이 다른 AWS 서비스에 의해 이루어졌는지 여부.

자세한 설명은 CloudTrail userIdentity 요소를 참조하세요.

Amazon DocumentDB 작업 프로파일링

Amazon DocumentDB(MongoDB 호환)를 사용하여 클러스터에서 수행된 작업의 실행 시간 및 세부 정보를 기록할 수 있습니다. 프로파일러는 개별 쿼리 성능 및 전체 클러스터 성능을 개선하기 위해 클러스터에서 가장 느린 작업을 모니터링하는 경우에 유용합니다.

프로파일러 특성은 기본적으로 비활성화됩니다. 이 옵션을 사용하도록 설정하면 프로파일러 로그 고객 정의 임계값(예: 100ms)보다 오래 걸리는 작업을 Amazon CloudWatch Logs에 기록합니다. 로깅되는 세부 정보에는 프로파일링된 명령, 시간, 계획 요약 및 클라이언트 메타데이터가 포함됩니다. 작업

이 CloudWatch 로그에 기록된 후 CloudWatch 로그 인사이트를 사용하여 Amazon DocumentDB 프로파일링 데이터를 분석, 모니터링 및 보관할 수 있습니다. 공통적인 쿼리는 공통 쿼리 섹션에 제공됩니다.

활성화되면 프로파일러는 클러스터에 있는 추가 리소스를 사용합니다. 높은 임계값(예: 500ms)으로 시작하여 값을 점차적으로 낮추면서 느린 작업을 식별하는 것이 좋습니다. 임계값 50ms로 시작하면 처리량이 많은 애플리케이션의 경우 클러스터에서 성능 문제가 발생할 수 있습니다. 프로파일러는 클러스터 수준에서 사용 가능하며 클러스터의 모든 인스턴스와 데이터베이스에서 작동합니다. Amazon DocumentDB 로그는 작업을 최상의 방법으로 Amazon CloudWatch Logs에 기록합니다.

Amazon DocumentDB는 프로파일러를 활성화하기 위해 추가 요금을 부과하지 않지만 CloudWatch 로그 사용에 대한 표준 요금이 부과됩니다. CloudWatch 로그 요금에 대한 자세한 내용은 Amazon CloudWatch 요금을 참조하십시오.

주제

- 지원되는 연산자
- 제한 사항
- Amazon DocumentDB 프로파일러 활성화
- Amazon DocumentDB 프로파일러 비활성화
- 프로파일러 로그 내보내기 비활성화
- Amazon DocumentDB 프로파일러 로그에 액세스
- 공통 쿼리

지원되는 연산자

Amazon DocumentDB 프로파일러는 다음 작업을 지원합니다:

- `aggregate`
- `count`
- `delete`
- `distinct`
- `find`(OP_QUERY 및 명령)
- `findAndModify`
- `insert`

- `update`

제한 사항

슬로우 쿼리 프로파일러는 쿼리의 전체 결과 집합이 한 배치에 들어갈 수 있고 결과 집합이 16MB(최대 BSON 크기) 미만인 경우에만 프로파일러 로그를 내보낼 수 있습니다. 16MB 이상의 결과 세트는 자동으로 여러 배치로 분할됩니다.

대부분의 드라이버나 쉘은 기본 배치 크기를 작게 설정할 수 있습니다. 쿼리의 일부로 배치 크기를 지정할 수 있습니다. 슬로우 쿼리 로그를 캡처하기 위해 예상 결과 집합의 크기를 초과하는 배치 크기를 권장합니다. 결과 집합 크기가 확실하지 않거나 크기가 다른 경우 배치 크기를 큰 수(예: 100k)로 설정할 수도 있습니다.

그러나 더 큰 배치 크기를 사용하면 응답이 클라이언트로 전송되기 전에 데이터베이스에서 더 많은 결과를 검색해야 합니다. 일부 쿼리의 경우 결과를 얻기 전에 지연이 더 길어질 수 있습니다. 전체 결과 집합을 소비할 계획이 없는 경우, 더 많은 I/O를 사용하여 쿼리를 처리하고 결과를 버릴 가능성이 있습니다.

Amazon DocumentDB 프로파일러 활성화

클러스터에서 프로파일러를 활성화하는 작업은 3단계 프로세스입니다. 모든 단계를 완료하거나 프로파일링 로그가 CloudWatch Logs로 전송되지 않도록 합니다. 프로파일러는 클러스터 수준에서 설정되며 클러스터의 모든 데이터베이스와 인스턴스에 대해 수행됩니다.

클러스터에서 프로파일러를 활성화하려면

1. 기본 클러스터 파라미터 그룹을 수정할 수 없으므로, 사용 가능한 사용자 지정 클러스터 파라미터 그룹이 있는지 확인하십시오. 자세한 내용은 Amazon DocumentDB 클러스터 파라미터 그룹 생성 섹션을 참조하십시오.
2. 사용 가능한 사용자 지정 클러스터 파라미터 그룹을 사용하여 `profiler`, `profiler_threshold_ms` 및 `profiler_sampling_rate` 파라미터를 수정합니다. 자세한 내용은 Amazon DocumentDB 클러스터 파라미터 그룹 수정 섹션을 참조하십시오.
3. 사용자 지정 클러스터 매개 변수 그룹을 사용하고 `profiler` CloudWatch Logs로 로그 내보내기를 사용하도록 클러스터를 생성하거나 수정합니다.

다음 섹션에서는 AWS Management Console 및 AWS Command Line Interface ()를 사용하여 이러한 단계를 구현하는 방법을 보여줍니다AWS CLI.

Using the AWS Management Console

1. 시작하기 전에 Amazon DocumentDB 클러스터와 사용자 정의 클러스터 파라미터 그룹이 없는 경우, 이를 생성합니다. 자세한 내용은 Amazon DocumentDB 클러스터 파라미터 그룹 생성 및 Amazon DocumentDB 클러스터 생성 섹션을 참조하십시오.
2. 사용 가능한 사용자 지정 클러스터 파라미터 그룹을 사용하여 다음 파라미터를 수정합니다. 자세한 내용은 Amazon DocumentDB 클러스터 파라미터 그룹 수정 섹션을 참조하십시오.

   - `profiler` - 쿼리 프로파일링을 활성화하거나 비활성화합니다. 허용된 값은 `enabled` 및 `disabled`입니다. 기본값은 `disabled`입니다. 프로파일링을 활성화하려면 값을 `enabled`로 설정합니다.
   - `profiler_threshold_ms` - `profiler`를 `enabled`으로 설정하면 `profiler_threshold_ms`보다 더 오래 걸리는 모든 명령이 CloudWatch 로그에 기록됩니다. 허용된 값은 `[50-INT_MAX]`입니다. 기본값은 `100`입니다.
   - `profiler_sampling_rate` - 프로파일링하거나 로깅해야 하는 느린 작업의 조각입니다. 허용된 값은 `[0.0-1.0]`입니다. 기본값은 `1.0`입니다.
3. 사용자 지정 클러스터 매개 변수 그룹을 사용하도록 클러스터를 수정하고 Amazon CloudWatch에 게시자 프로파일러 로그 내보내기를 설정합니다.

   a. 탐색 창에서 클러스터를 선택하여 사용자 지정 파라미터 그룹을 클러스터에 추가합니다.

   b. 파라미터 그룹을 연결할 클러스터의 이름 왼쪽에 있는 버튼을 선택합니다. 작업, 수정 순으로 선택하여 클러스터를 수정합니다.

   c. 클러스터 옵션의 위 단계에서 사용자 지정 파라미터 그룹을 선택하여 클러스터에 추가합니다.

   d. 로그 내보내기에서 Amazon CloudWatch에 게시자로 등록할 프로파일 로그를 선택합니다.

   e. 수정 사항의 요약을 보려면 계속을 선택합니다.

   f. 변경 사항을 확인한 후 즉시 적용하거나 수정 일정 아래의 다음 유지 관리 기간 중에 적용할 수 있습니다.

   g. 클러스터를 새 파라미터 그룹으로 업데이트하려면 클러스터 수정을 선택합니다.

Using the AWS CLI

다음 절차는 `sample-cluster` 클러스터에 지원되는 모든 작업에 대해 프로파일러를 활성화합니다.

1. 시작하기 전에 다음 명령을 실행한 후 이름에 default가 없고 docdb3.6이 파라미터 그룹 패밀리인 클러스터 파라미터 그룹의 출력을 검토하여 사용 가능한 사용자 지정 클러스터 파라미터 그룹이 있는지 확인합니다. 기본이 아닌 클러스터 파라미터 그룹이 없는 경우 Amazon DocumentDB 클러스터 파라미터 그룹 생성 단원을 참조하십시오.

```
aws docdb describe-db-cluster-parameter-groups \
    --query 'DBClusterParameterGroups[*].
[DBClusterParameterGroupName,DBParameterGroupFamily]'
```

다음 출력에서는 sample-parameter-group 만 두 기준을 모두 충족합니다.

```
[
       [
           "default.docdb3.6",
           "docdb3.6"
       ],
       [
           "sample-parameter-group",
           "docdb3.6"
       ]
]
```

2. 사용자 지정 클러스터 파라미터 그룹을 사용하여 다음 파라미터를 수정합니다:

   - profiler - 쿼리 프로파일링을 활성화하거나 비활성화합니다. 허용된 값은 enabled 및 disabled입니다. 기본값은 disabled입니다. 프로파일링을 활성화하려면 값을 enabled로 설정합니다.
   - profiler_threshold_ms - profiler를 enabled으로 설정하면 profiler_threshold_ms보다 더 오래 걸리는 모든 명령이 CloudWatch에 기록됩니다. 허용된 값은 [50-INT_MAX]입니다. 기본값은 100입니다.
   - profiler_sampling_rate - 프로파일링하거나 로깅해야 하는 느린 작업의 조각입니다. 허용된 값은 [0.0-1.0]입니다. 기본값은 1.0입니다.

```
aws docdb modify-db-cluster-parameter-group \
    --db-cluster-parameter-group-name sample-parameter-group \
    --parameters
 ParameterName=profiler,ParameterValue=enabled,ApplyMethod=immediate \

 ParameterName=profiler_threshold_ms,ParameterValue=100,ApplyMethod=immediate \
```

```
  ParameterName=profiler_sampling_rate,ParameterValue=0.5,ApplyMethod=immediate
```

3. 이전 단계의 sample-parameter-group 사용자 지정 클러스터 파라미터 그룹을 사용하고 파라미터 --enable-cloudwatch-logs-exports를 profiler으로 설정하도록 Amazon DocumentDB 클러스터를 수정합니다.

   다음 코드는 이전 단계에서 sample-parameter-group를 사용하도록 클러스터 sample-cluster을 수정하고 사용하도록 설정된 CloudWatch 로그 내보내기에 profiler을 추가합니다.

```
aws docdb modify-db-cluster \
       --db-cluster-identifier sample-cluster \
       --db-cluster-parameter-group-name sample-parameter-group \
       --cloudwatch-logs-export-configuration '{"EnableLogTypes":["profiler"]}'
```

   이 작업의 출력은 다음과 같이 표시됩니다.

```
{
    "DBCluster": {
        "AvailabilityZones": [
            "us-east-1c",
            "us-east-1b",
            "us-east-1a"
        ],
        "BackupRetentionPeriod": 1,
        "DBClusterIdentifier": "sample-cluster",
        "DBClusterParameterGroup": "sample-parameter-group",
        "DBSubnetGroup": "default",
        "Status": "available",
        "EarliestRestorableTime": "2020-04-07T02:05:12.479Z",
        "Endpoint": "sample-cluster.node.us-east-1.docdb.amazonaws.com",
        "ReaderEndpoint": "sample-cluster.node.us-east-1.docdb.amazonaws.com",
        "MultiAZ": false,
        "Engine": "docdb",
        "EngineVersion": "3.6.0",
        "LatestRestorableTime": "2020-04-08T22:08:59.317Z",
        "Port": 27017,
        "MasterUsername": "test",
        "PreferredBackupWindow": "02:00-02:30",
        "PreferredMaintenanceWindow": "tue:09:50-tue:10:20",
```

```
        "DBClusterMembers": [
            {
                "DBInstanceIdentifier": "sample-instance-1",
                "IsClusterWriter": true,
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1
            },
            {
                "DBInstanceIdentifier": "sample-instance-2",
                "IsClusterWriter": true,
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1
            }
        ],
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-abcd0123",
                "Status": "active"
            }
        ],
        "HostedZoneId": "ABCDEFGHIJKLM",
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:us-east-1:<accountID>:key/sample-key",
        "DbClusterResourceId": "cluster-ABCDEFGHIJKLMNOPQRSTUVWXYZ",
        "DBClusterArn": "arn:aws:rds:us-east-1:<accountID>:cluster:sample-
cluster",
        "AssociatedRoles": [],
        "ClusterCreateTime": "2020-01-10T22:13:38.261Z",
        "EnabledCloudwatchLogsExports": [
            "profiler"
        ],
        "DeletionProtection": true
    }
}
```

Amazon DocumentDB 프로파일러 비활성화

프로파일러를 실행 중지하려면 `profiler` 매개 변수와 `profiler` 로그를 CloudWatch Logs로 내보내기를 모두 실행 중지해야 합니다.

프로파일러 비활성화

다음과 AWS CLI같이 AWS Management Console 또는를 사용하여 `profiler` 파라미터를 비활성화할 수 있습니다.

Using the AWS Management Console

다음 절차에서는를 사용하여 Amazon DocumentDB AWS Management Console 를 비활성화합니다`profiler`.

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 파라미터 그룹을 선택합니다. 그런 다음 프로파일러를 비활성화하려는 클러스터 파라미터 그룹의 이름을 선택합니다.
3. 그 결과 표시되는 클러스터 파라미터 페이지에서 `profiler` 파라미터 왼쪽에 있는 버튼을 선택하고 편집을 선택합니다.
4. 프로파일러 수정 대화 상자의 목록에서 `disabled`를 선택합니다.
5. 클러스터 파라미터 수정을 선택합니다.

Using the AWS CLI

AWS CLI를 사용하여 클러스터에서 `profiler`를 비활성화하려면 다음과 같이 클러스터를 수정합니다.

```
aws docdb modify-db-cluster-parameter-group \
    --db-cluster-parameter-group-name sample-parameter-group \
    --parameters
 ParameterName=profiler,ParameterValue=disabled,ApplyMethod=immediate
```

프로파일러 로그 내보내기 비활성화

다음과 AWS CLI같이 AWS Management Console 또는를 사용하여 CloudWatch Logs로 `profiler` 로그 내보내기를 비활성화할 수 있습니다.

Using the AWS Management Console

다음 절차에서는를 사용하여 CloudWatch AWS Management Console 로 로그를 내보내는 Amazon DocumentDB를 비활성화합니다.

1. https://console.aws.amazon.com/docdb에서 Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 클러스터를 선택합니다. 로그 내보내기를 비활성화할 클러스터의 이름 왼쪽에 있는 버튼을 선택합니다.
3. 작업 메뉴에서 수정을 선택합니다.
4. 로그 내보내기 섹션까지 아래로 스크롤하여 프로필러 로그를 선택 취소합니다.
5. Continue(계속)을 선택합니다.
6. 변경 사항을 검토하고, 언제 이 변경 사항을 클러스터에 적용할지를 선택합니다:
   - 예약된 다음 유지 관리 기간에 적용
   - 즉시 적용
7. 클러스터 수정을 선택합니다.

Using the AWS CLI

다음 코드는 클러스터`sample-cluster` 를 수정하고 CloudWatch 프로파일러 로그를 비활성화합니다.

Example

Linux, macOS, Unix의 경우:

```
aws docdb modify-db-cluster \
   --db-cluster-identifier sample-cluster \
   --cloudwatch-logs-export-configuration '{"DisableLogTypes":["profiler"]}'
```

Windows의 경우:

```
aws docdb modify-db-cluster ^
   --db-cluster-identifier sample-cluster ^
   --cloudwatch-logs-export-configuration '{"DisableLogTypes":["profiler"]}'
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
{
    "DBCluster": {
        "AvailabilityZones": [
            "us-east-1c",
            "us-east-1b",
            "us-east-1a"
```

```
        ],
        "BackupRetentionPeriod": 1,
        "DBClusterIdentifier": "sample-cluster",
        "DBClusterParameterGroup": "sample-parameter-group",
        "DBSubnetGroup": "default",
        "Status": "available",
        "EarliestRestorableTime": "2020-04-08T02:05:17.266Z",
        "Endpoint": "sample-cluster.node.us-east-1.docdb.amazonaws.com",
        "ReaderEndpoint": "sample-cluster.node.us-east-1.docdb.amazonaws.com",
        "MultiAZ": false,
        "Engine": "docdb",
        "EngineVersion": "3.6.0",
        "LatestRestorableTime": "2020-04-09T05:14:44.356Z",
        "Port": 27017,
        "MasterUsername": "test",
        "PreferredBackupWindow": "02:00-02:30",
        "PreferredMaintenanceWindow": "tue:09:50-tue:10:20",
        "DBClusterMembers": [
            {
                "DBInstanceIdentifier": "sample-instance-1",
                "IsClusterWriter": true,
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1
            },
            {
                "DBInstanceIdentifier": "sample-instance-2",
                "IsClusterWriter": true,
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1
            }
        ],
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-abcd0123",
                "Status": "active"
            }
        ],
        "HostedZoneId": "ABCDEFGHIJKLM",
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:us-east-1:<accountID>:key/sample-key",
        "DbClusterResourceId": "cluster-ABCDEFGHIJKLMNOPQRSTUVWXYZ",
        "DBClusterArn": "arn:aws:rds:us-east-1:<accountID>:cluster:sample-cluster",
        "AssociatedRoles": [],
        "ClusterCreateTime": "2020-01-10T22:13:38.261Z",
```

```
        "DeletionProtection": true
    }
}
```

Amazon DocumentDB 프로파일러 로그에 액세스

Amazon CloudWatch에서 프로필 로그에 액세스하려면 다음 단계를 수행합니다.

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 사용자가 Amazon DocumentDB 클러스터와 동일한 리전에 있어야 합니다.
3. 탐색 창에서 로그를 선택합니다.
4. 클러스터에 대한 프로파일러 로그를 찾으려면, 목록에서 `/aws/docdb/`*`yourClusterName`*`/profiler`을 선택합니다.

   각 인스턴스에 대한 프로파일 로그는 해당 인스턴스 이름 아래에서 사용할 수 있습니다.

공통 쿼리

다음은 프로파일링된 명령을 분석하는 데 사용할 수 있는 몇 가지 공통적인 쿼리입니다. CloudWatch Logs Insights에 대한 자세한 내용은 CloudWatch Logs Insights로 로그 데이터 분석 및 샘플 쿼리를 참조하십시오.

지정된 모음에서 가장 느린 10개 작업 가져오기

```
filter ns="test.foo" | sort millis desc | limit 10
```

모음에서 60ms보다 오래 걸린 모든 업데이트 작업 가져오기

```
filter millis > 60 and op = "update"
```

지난 달에 가장 느린 10개 작업 가져오기

```
sort millis desc | limit 10
```

COLLSCAN 계획 요약을 사용하여 모든 쿼리 가져오기

```
filter planSummary="COLLSCAN"
```

성능 개선 도우미로 모니터링

성능 개선 도우미는 기존 Amazon DocumentDB 모니터링 특성을 확장한 것으로서 데이터베이스 성능을 표시하여 성능 문제를 분석하는 데 효과적입니다. 성능 개선 도우미 대시보드를 사용하면 데이터베이스 로드를 시각화하고 대기, 쿼리문, 호스트 또는 애플리케이션별로 로드를 필터링할 수 있습니다.

> **Note**
>
> 성능개선 도우미는 Amazon DocumentDB 3.6, 4.0 및 5.0 인스턴스 기반 클러스터에서만 사용할 수 있습니다.

어떻게 유용할까요?

- 데이터베이스 성능 가시화 — 언제 어디에서 데이터베이스의 로드가 있는지 파악하기 위해 로드를 시각화합니다
- 데이터베이스 로드의 원인 파악 — 어떤 쿼리, 호스트, 애플리케이션이 인스턴스 로드에 영향을 미치는지 확인합니다
- 데이터베이스에 로드가 있을 때 판단 — 성능 개선 도우미 대시보드를 확대하여 특정 이벤트에 초점을 맞추거나 축소하여 더 큰 시간에 걸쳐 추세를 확인할 수 있습니다
- 데이터베이스 로드에 대한 알림 — CloudWatch에서 새로운 데이터베이스 로드 메트릭에 자동으로 액세스하여 다른 Amazon DocumentDB 메트릭과 함께 DB 로드 메트릭을 모니터링하고 해당 메트릭에 대한 알림을 설정할 수 있습니다

Amazon DocumentDB 성능 개선 도우미의 한계는 무엇입니까?

- AWS GovCloud(미국 동부) 및 AWS GovCloud(미국 서부) 리전의 성능 개선 도우미를 사용할 수 없음
- Amazon DocumentDB용 성능 개선 도우미는 최대 7일 간의 성능 데이터를 유지합니다
- 1,024바이트보다 긴 쿼리는 성능 개선 도우미에 집계되지 않습니다.

주제

- 성능 개선 도우미 개념
- 성능 개선 도우미 활성화 및 비활성화
- 성능 개선 도우미에 대한 액세스 정책 구성
- 성능 개선 도우미 대시보드를 사용한 지표 분석
- 성능 개선 도우미 API를 사용하여 지표 검색
- 성능 개선 도우미를 위한 Amazon CloudWatch 지표
- 카운터 지표에 대한 성능 개선 도우미

성능 개선 도우미 개념

주제

- 평균 활성 세션
- Dimensions
- 최대 vCPU

평균 활성 세션

데이터베이스 로드(DB 로드)는 데이터베이스의 활동 수준을 측정합니다. 성능 개선 도우미의 핵심 지표는 DB Load이며, 1초 간격으로 수집됩니다. DBLoad 메트릭의 단위는 Amazon DocumentDB 인스턴스에 대한 AAS(평균 활성 세션 수)입니다.

활성 세션은 Amazon DocumentDB 인스턴스에 작업을 제출하고 응답을 기다리는 연결입니다. 예를 들어 Amazon DocumentDB 인스턴스에 쿼리를 제출하면 인스턴스가 쿼리를 처리하는 동안 데이터베이스 세션이 활성화됩니다.

평균 활성 세션을 구하기 위해 성능 개선 도우미는 쿼리를 동시에 실행하는 세션 수를 샘플링합니다. AAS는 총 세션 수를 총 샘플 수로 나눈 값입니다. 다음 표는 실행 중인 쿼리의 연속된 5개 샘플을 보여줍니다.

샘플	쿼리를 실행 중인 세션 수	AAS	계산
1	2	2	2개 세션/1개 샘플
2	0	1	2개 세션/2개 샘플

샘플	쿼리를 실행 중인 세션 수	AAS	계산
3	4	2	6개 세션/3개 샘플
4	0	1.5	6개 세션/4개 샘플
5	4	2	10개 세션/5개 샘플

앞의 예제에서 1-5의 시간 간격에 대한 DB 로드는 2 AAS입니다. DB 로드의 증가는 평균적으로 데이터베이스에서 더 많은 세션이 실행되고 있음을 의미합니다.

Dimensions

DB Load 지표는 '차원'이라는 하위 구성 요소로 구분할 수 있기 때문에 다른 시계열 지표와 다릅니다. 차원을 DB Load 지표의 다양한 특성에 대한 범주로 생각할 수 있습니다. 성능 문제를 진단할 때 가장 유용한 차원은 대기 상태 및 상위 쿼리입니다.

대기 상태

대기 상태는 쿼리 문이 계속 실행되려면 특정 이벤트가 발생할 때까지 쿼리 문이 기다리도록 합니다. 예를 들어 잠긴 리소스의 잠금이 해제될 때까지 쿼리 문이 대기할 수 있습니다. DB Load와 대기 상태와 결합하여 전체 세션 상태를 이해할 수 있습니다. Amazon DocumentDB의 다양한 대기 상태는 다음과 같습니다:

Amazon DocumentDB 대기 상태	대기 상태 설명
래치	래치 대기 상태는 세션이 버퍼 풀의 페이징을 기다리고 있을 때 발생합니다. 버퍼 풀에서 자주 호출되는 것은 시스템에서 처리 중인 대규모 쿼리, 수집 검색이 빈번할 때 또는 버퍼 풀이 너무 작아서 작업 세트를 처리할 수 없을 때 더 자주 발생할 수 있습니다.
CPU	CPU 대기 상태는 세션이 CPU에서 대기할 때 발생합니다.
CollectionLock	CollectionLock 대기 상태는 세션이 컬렉션에 대한 잠금을 획득하기 위해 대기할 때 발생합니다.

Amazon DocumentDB 대기 상태	대기 상태 설명
	이러한 이벤트는 컬렉션에 DDL 작업이 있을 때 발생합니다.
DocumentLock	DocumentLock 대기 상태는 세션이 문서에 대한 잠금 획득을 대기 중일 때 발생합니다. 동일한 문서에 대한 동시 쓰기 횟수가 많으면 해당 문서의 DocumentLock 대기 상태가 늘어나는데 기여하게 됩니다.
SystemLock	SystemLock 대기 상태는 세션이 시스템에서 대기할 때 발생합니다. 이 문제는 시스템에서 자주 실행되는 쿼리, 오래 실행되는 트랜잭션 또는 높은 동시성이 있을 때 발생할 수 있습니다.
IO	IO 대기 상태는 IO에서 세션이 완료될 때 발생합니다.
BufferLock	BufferLock 대기 상태는 세션이 버퍼의 공유 페이지에서 잠금을 획득하기 위해 대기할 때 발생합니다. 다른 프로세스가 요청된 페이지에서 열린 커서를 유지하고 있는 경우 BufferLock 대기 상태가 길어질 수 있습니다.
LowMemThrottle	Amazon DocumentDB 인스턴스의 과도한 메모리 부족으로 인해 세션이 대기 중일 때 LowMemThrottle 대기 상태가 발생합니다. 이 상태가 오래 지속되면 인스턴스를 확장하여 추가 메모리를 제공하는 것을 고려합니다. 자세한 내용은 리소스 거버너를 참조하세요.
BackgroundActivity	BackgroundActivity 대기 상태는 세션이 내부 시스템 프로세스에서 대기할 때 발생합니다.

Amazon DocumentDB 대기 상태	대기 상태 설명
기타	기타 대기 상태는 내부 대기 상태입니다. 이 상태가 오래 지속되면 이 쿼리를 종료하는 것을 고려하십시오. 자세한 내용은 긴 실행 중이거나 차단된 쿼리를 어떻게 찾고 종료하나요?를 참조하십시오.

상위 쿼리

대기 상태는 병목 현상을 보여주는 반면 상위 쿼리는 DB 로드에 가장 큰 기여를 하는 쿼리를 보여줍니다. 예를 들어 현재 데이터베이스에서 여러 쿼리가 실행 중이더라도 단일 쿼리가 DB 로드의 99%를 소비할 수 있습니다. 이 경우 로드가 높으면 쿼리에 문제가 있음을 나타낼 수 있습니다.

최대 vCPU

대시보드에서 데이터베이스 로드 차트는 세션 정보를 수집, 집계 및 표시합니다. 활성 세션이 최대 CPU를 초과하는지 확인하려면 최대 vCPU 선과의 관계를 확인합니다. 최대 vCPU 값은 Amazon DocumentDB 인스턴스의 vCPU(가상 CPU) 코어 수에 따라 결정됩니다.

DB 로드가 최대 vCPU 선을 상회하는 경우가 잦아지고 CPU가 기본 대기 상태라면 CPU에서 과부하가 발생한 것입니다. 이 경우 인스턴스에 대한 연결을 조절하거나 CPU 로드가 큰 쿼리를 조정하거나 더 큰 인스턴스 클래스를 고려할 수 있습니다. 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.

성능 개선 도우미 활성화 및 비활성화

성능 개선 도우미를 사용하려면 DB 인스턴스에서 이 기능을 활성화합니다. 필요한 경우 나중에 이를 비활성화할 수 있습니다. 성능 개선 도우미를 활성화하거나 비활성화해도 가동 중지, 재부팅 또는 장애 조치가 발생하지 않습니다.

성능 개선 도우미 에이전트는 DB 호스트에서 제한된 CPU 및 메모리를 사용합니다. DB 로드가 높을 경우 에이전트는 데이터 수집 빈도를 줄여 성능에 미치는 영향을 제한합니다.

클러스터 생성 시 성능 개선 도우미 활성화

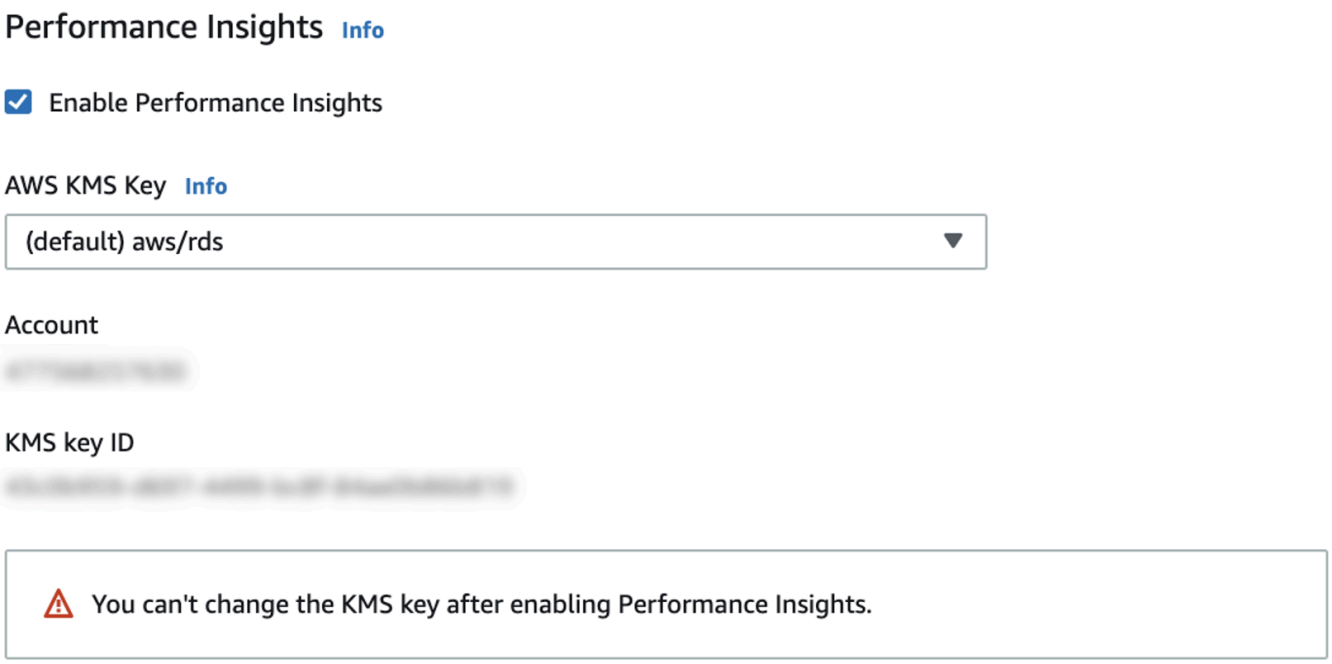
콘솔에서 새 DB 인스턴스를 생성하거나 수정할 때 성능 개선 도우미를 활성화하거나 비활성화할 수 있습니다.

사용 AWS Management Console

콘솔에서 Amazon DocumentDB 클러스터를 작성할 때 성능 개선 도우미를 활성화할 수 있습니다. 새 Amazon DocumentDB 클러스터를 작성할 때 성능 개선 도우미 섹션에서 성능 개선 도우미 활성화를 선택하여 성능 개선 도우미를 활성화합니다.

콘솔 지침

1. 클러스터를 만드는 방법에 대한 지침은 Amazon DocumentDB 클러스터 생성을 참조하세요.
2. 성능 개선 도우미 섹션에서 성능 개선 도우미 활성화를 선택합니다.



> **Note**
>
> 성능 개선 도우미 데이터 보존 기간은 7일입니다.

AWS KMS 키 - AWS KMS 키를 지정합니다. 성능 개선 도우미는 AWS KMS 키를 사용하여 잠재적으로 민감한 모든 데이터를 암호화합니다. 데이터는 암호화된 상태로 전송 및 저장됩니다. 자세한 내용은 성능 개선 도우미에 대한 AWS AWS KMS 정책 구성을 참조하세요.

인스턴스 수정 시 활성화 및 비활성화

콘솔 또는 AWS CLI을 사용하여 성능 개선 도우미를 활성화하도록 DB 인스턴스를 수정할 수 있습니다.

Using the AWS Management Console

콘솔 지침

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 클러스터를 선택하십시오.
3. DB 인스턴스를 선택하고 수정을 선택합니다.
4. 성능 개선 도우미 섹션에서 성능 개선 도우미 활성화 또는 성능 개선 도우미 비활성화를 선택합니다.

   > **Note**
   >
   > 성능 개선 도우미 활성화를 선택하면 AWS AWS KMS 키를 지정할 수 있습니다. 성능 개선 도우미는 AWS KMS 키를 사용하여 잠재적으로 민감한 모든 데이터를 암호화합니다. 데이터는 암호화된 상태로 전송 및 저장됩니다. 자세한 내용은 정지 상태의 Amazon DocumentDB Data 암호화를 참조하십시오.

5. Continue(계속)를 선택합니다.
6. 수정 스케줄링에 대해 즉시 적용을 선택합니다. 다음 예약된 유지 관리 기간 동안 적용을 선택하면 인스턴스는 이 설정을 무시하고 즉시 성능 개선 도우미를 활성화합니다.
7. Modify Instance(인스턴스 수정)를 선택합니다.

Using the AWS CLI

`create-db-instance` 또는 `modify-db-instance` AWS AWS CLI 명령을 사용할 때를 지정하여 성능 개선 도우미를 활성화`--enable-performance-insights`하거나를 지정하여 비활성화할 수 있습니다`--no-enable-performance-insights`.

다음 절차에서는 AWS AWS CLI를 사용하여 DB 인스턴스의 성능 개선 도우미를 활성화 또는 비활성화하는 방법을 설명합니다.

AWS AWS CLI 지침

modify-db-instance AWS AWS CLI 명령을 호출하고 다음 값을 제공합니다.

- --db-instance-identifer — DB 인스턴스 이름.
- 활성화하려면 --enable-performance-insights 또는 비활성화하려면 --no-enable-performance-insights

Example

다음 예제에서는 sample-db-instance에 대한 성능 개선 도우미를 활성화합니다:

For Linux, macOS, or Unix:

```
aws docdb modify-db-instance \
    --db-instance-identifier sample-db-instance \
    --enable-performance-insights
```

For Windows:

```
aws docdb modify-db-instance ^
    --db-instance-identifier sample-db-instance ^
    --enable-performance-insights
```

성능 개선 도우미에 대한 액세스 정책 구성

성능 개선 도우미에 액세스하려면 AWS Identity and Access Management (IAM)의 적절한 권한이 있어야 합니다. 액세스 권한 부여 옵션은 다음과 같습니다:

- 권한 세트 또는 역할에 AmazonRDSPerformanceInsightsReadOnly 관리형 정책을 연결합니다.
- 사용자 지정 IAM 정책을 생성하고 권한 세트 또는 역할에 연결합니다.

또한 성능 개선 도우미를 활성화할 때 고객 관리형 키를 지정한 경우 계정의 사용자에게 KMS 키에 대한 kms:Decrypt 및 kms:GenerateDataKey 권한이 있는지 확인합니다.

> **Note**
>
> AWS KMS 키 및 보안 그룹 관리를 통한 encryption-at-rest를 위해 Amazon DocumentDB는 Amazon RDS와 공유되는 운영 기술을 활용합니다.

AmazonRDSPerformanceInsightsReadOnly 정책을 IAM 보안 주체에 연결

AmazonRDSPerformanceInsightsReadOnly는 Amazon DocumentDB 성능 개선 도우미 API의 모든 읽기 전용 작업에 대한 액세스 권한을 부여하는 AWS관리형 정책입니다. 현재 이 API의 모든 작업은 읽기 전용입니다. AmazonRDSPerformanceInsightsReadOnly를 권한 세트 또는 역할에 연결하면 수신자는 성능 개선 도우미를 다른 콘솔 기능과 함께 사용할 수 있습니다.

성능 개선 도우미를 위한 사용자 지정 IAM 정책 만들기

AmazonRDSPerformanceInsightsReadOnly 정책이 없는 사용자의 경우, 사용자 관리형 IAM 정책을 생성 또는 수정하여 성능 개선 도우미에 대한 액세스 권한을 부여할 수 있습니다. 정책을 권한 세트 또는 역할에 연결할 때 수신자는 성능 개선 도우미을 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 탭을 선택합니다.
5. 다음 텍스트를 복사하여 붙여넣습니다. *us-east-1*을 AWS 리전 이름으로 바꾸고 *111122223333*을 고객 계정 번호로 바꿉니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "rds:DescribeDBInstances",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "rds:DescribeDBClusters",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "pi:DescribeDimensionKeys",
            "Resource": "arn:aws:pi:us-east-1:111122223333:metrics/rds/*"
```

```
        },
        {
            "Effect": "Allow",
            "Action": "pi:GetDimensionKeyDetails",
            "Resource": "arn:aws:pi:us-east-1:111122223333:metrics/rds/*"
        },
        {
            "Effect": "Allow",
            "Action": "pi:GetResourceMetadata",
            "Resource": "arn:aws:pi:us-east-1:111122223333:metrics/rds/*"
        },
        {
            "Effect": "Allow",
            "Action": "pi:GetResourceMetrics",
            "Resource": "arn:aws:pi:us-east-1:111122223333:metrics/rds/*"
        },
        {
            "Effect": "Allow",
            "Action": "pi:ListAvailableResourceDimensions",
            "Resource": "arn:aws:pi:us-east-1:111122223333:metrics/rds/*"
        },
        {
            "Effect": "Allow",
            "Action": "pi:ListAvailableResourceMetrics",
            "Resource": "arn:aws:pi:us-east-1:111122223333:metrics/rds/*"
        }
    ]
}
```

6. 정책 검토를 선택합니다.
7. 정책의 이름과 설명(선택 사항)을 지정한 다음 정책 검토를 선택합니다.

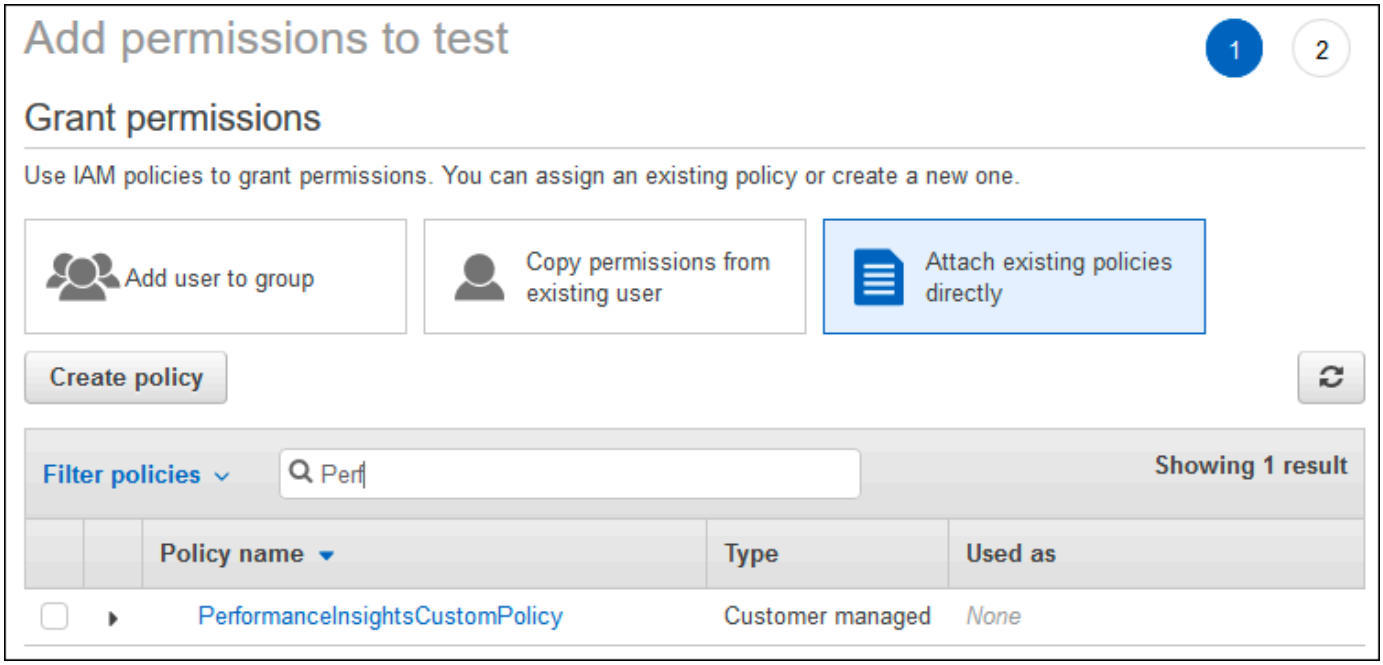
이제 정책을 권한 세트 또는 역할에 연결할 수 있습니다. 다음 절차에서는 이 목적으로 사용할 수 있는 사용자가 이미 있다고 가정합니다.

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 도우미을 사용하려면 사용자 정의 정책 외에 Amazon DocumentDB에 액세스할 수 있어야 합니다. 예를 들어 AmazonDocDBReadOnlyAccess 사전 정의 정책은 Amazon DocDB에 대한 읽기 전용 액세스를 제공합니다. 자세한 내용은 정책을 사용한 액세스 권리를 참조하세요.

4. 요약 페이지에서 권한 추가를 선택합니다.
5. 기존 정책 직접 첨부를 선택합니다. 검색에 다음과 같이 정책 이름의 첫 문자 몇 개를 입력합니다.



6. 정책을 선택하고 다음: 검토를 선택합니다.
7. 권한 추가를 선택합니다.

성능 개선 도우미에 대한 AWS KMS 정책 구성

성능 개선 도우미는 AWS KMS key 를 사용하여 민감한 데이터를 암호화합니다. API 또는 콘솔을 통해 성능 개선 도우미를 활성화하면 다음 옵션이 있습니다:

- 기본값을 선택합니다 AWS 관리형 키.

  Amazon DocumentDB는 새 DB 인스턴스에 AWS 관리형 키 를 사용합니다. Amazon DocumentDB는 AWS 계정에 AWS 관리형 키 대한를 생성합니다. AWS 계정에는 AWS 리전마다 Amazon DocumentDB에 AWS 관리형 키 대해 다른이 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 KMS에서 키 정책 사용을 참조하세요.

Example

다음 샘플 키 정책은 KMS 키 정책에 문을 추가하는 방법을 보여 줍니다. 이러한 문을 통해 성능 개선 도우미에 액세스할 수 있습니다. 사용 방법에 따라 몇 가지 제한 사항을 변경할 AWS KMS수 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/DocumentDB 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 DocDB instance.
            //The value is a DbiResourceId.
           "kms:EncryptionContext:aws:rds:db-id": "db-AAAAABBBBBCCCCDDDDDEEEEE"
        }
    }
}
```

성능 개선 도우미 대시보드를 사용한 지표 분석

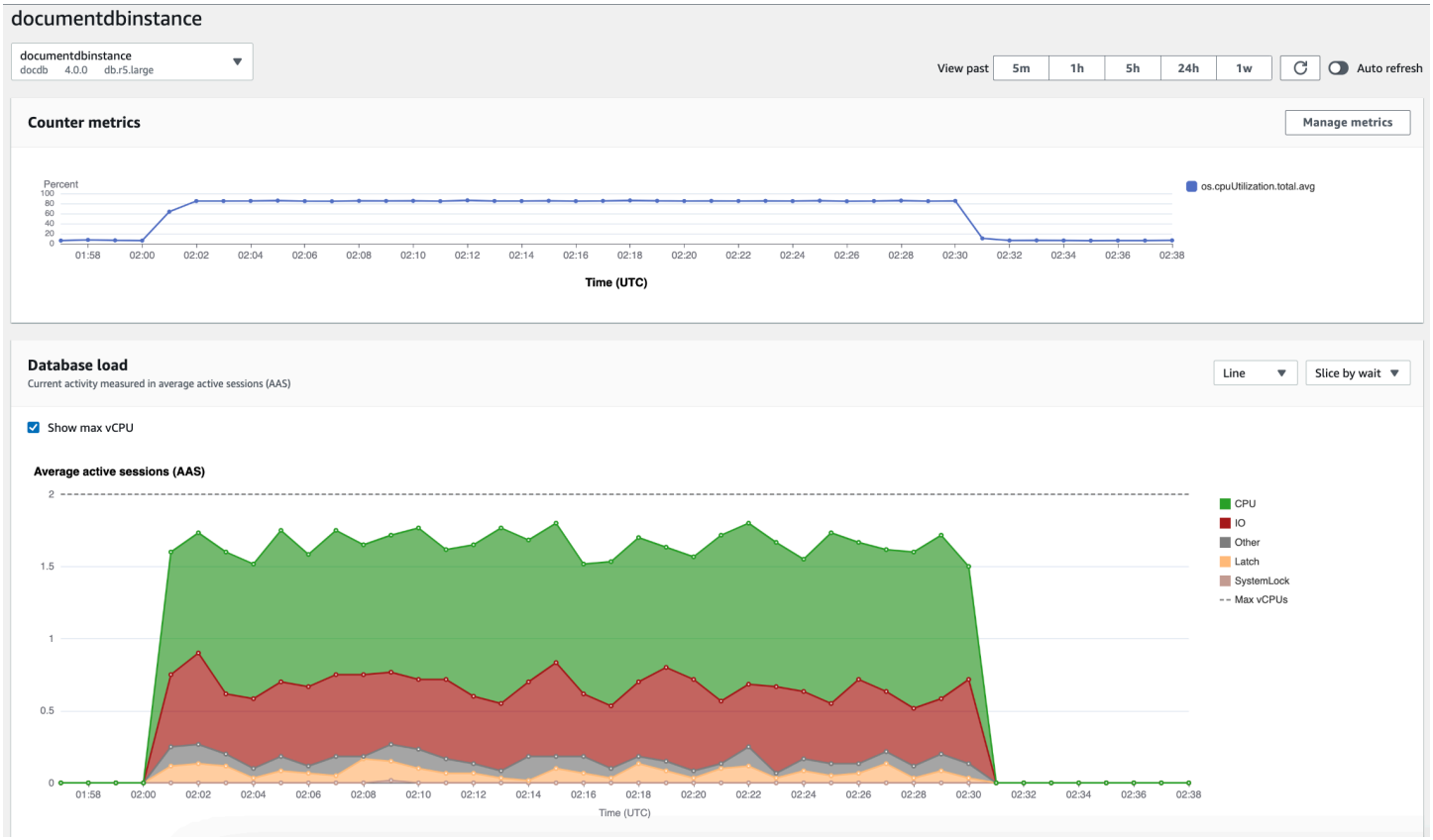
성능 개선 도우미 대시보드에는 성능 문제를 분석하여 해결할 수 있는 데이터베이스 성능 정보가 포함됩니다. 기본 대시보드 페이지에서 데이터베이스 로드 (DB load)에 대한 정보를 볼 수 있습니다. 대기 상태나 쿼리와 같은 차원별로 DB 로드를 "슬라이스"할 수 있습니다.

주제

- 성능 개선 도우미 대시보드 개요
- 성능 개선 도우미 대시보드 열기
- 대기 상태별 데이터베이스 로드 분석
- 상위 쿼리 탭 개요
- 데이터베이스 로드 차트에서 확대

성능 개선 도우미 대시보드 개요

대시보드는 성능 개선 도우미와 상호 작용하는 가장 간편한 방법입니다. 다음 예제는 Amazon DocumentDB 인스턴스의 대시보드를 보여줍니다. 성능 개선 도우미 대시보드는 기본적으로 마지막 1시간 동안 수집된 데이터를 표시합니다.



대시보드는 다음과 같은 부분으로 나뉩니다:

1. 카운터 지표 – 특정 성능 카운터 지표의 데이터를 보여줍니다.
2. 데이터베이스 로드 – 최대 vCPU 선으로 표시된 바와 같이 DB 로드와 DB 인스턴스 용량을 비교하는 방식을 표시합니다.
3. 상위 차원 – DB 로드에 기여하는 상위 차원을 보여줍니다. 이러한 차원에는 `waits`, `queries`, `hosts`, `databases`, 및 `applications`가 포함됩니다.

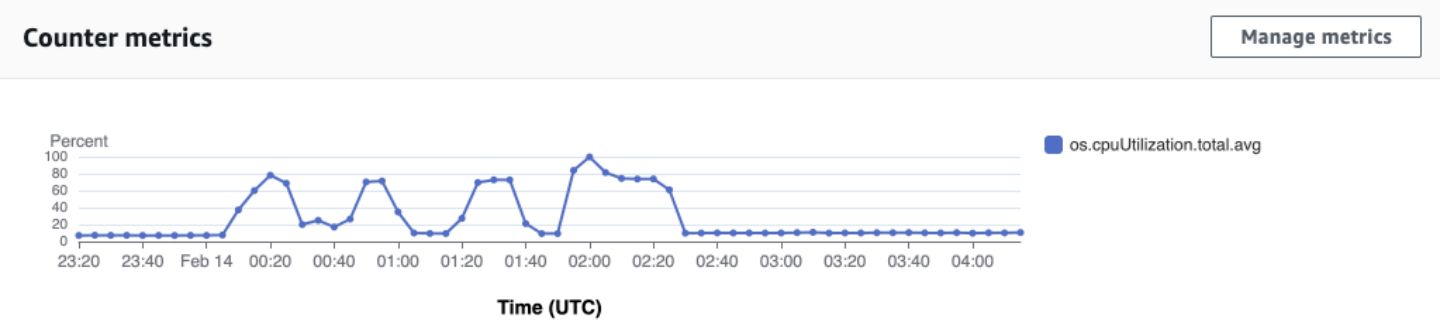
주제

- 카운터 지표 차트
- 데이터베이스 로드 차트
- 상위 측정기준 테이블

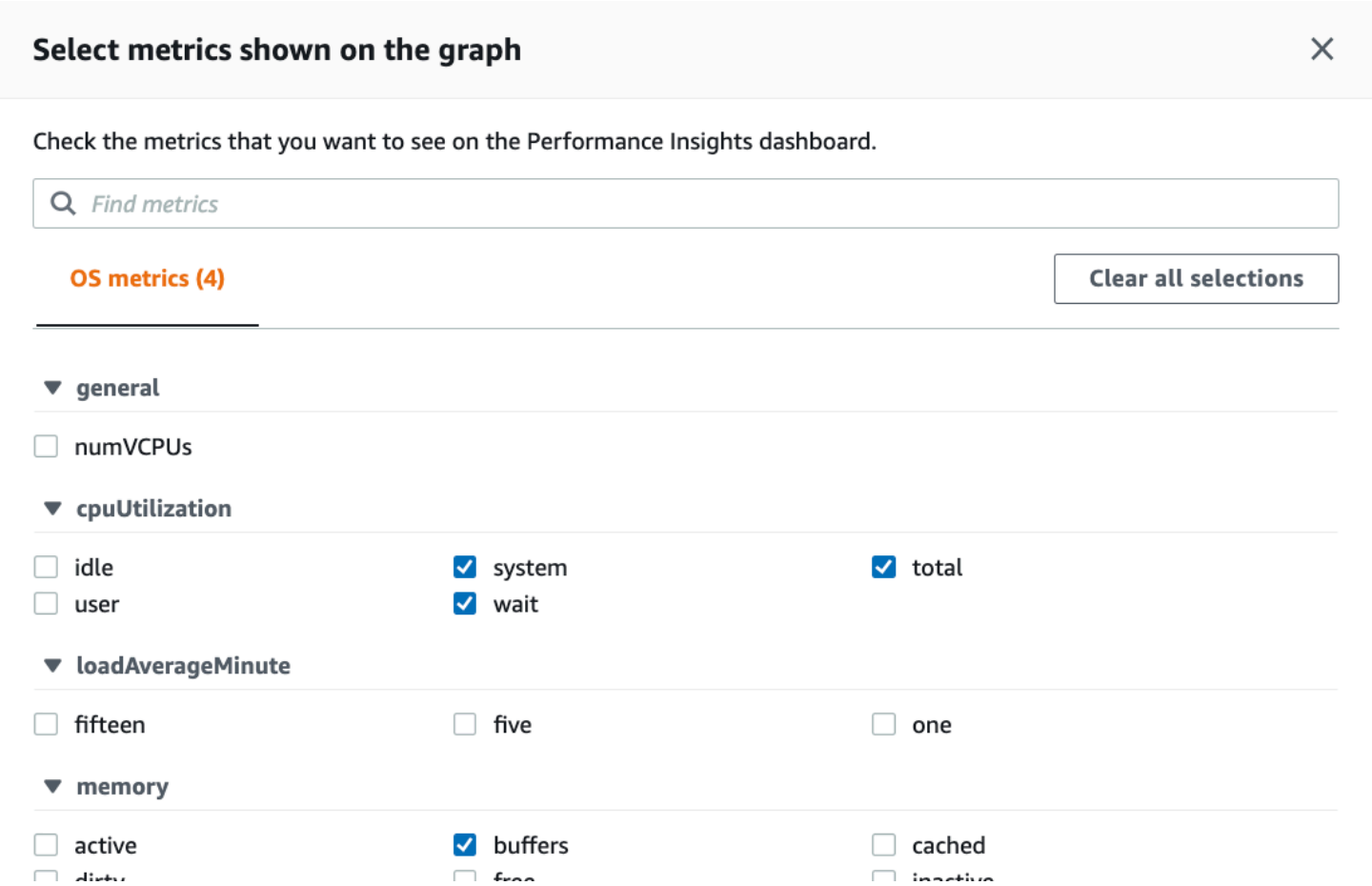
카운터 지표 차트

계수기 지표를 통해 성능 개선 도우미 대시보드에 최대 10개의 추가 그래프가 포함되도록 사용자 지정할 수 있습니다. 이 그래프는 수십 개의 운영 체제 지표를 보여줍니다. 이 정보와 데이터베이스 로드를 연관지으면 성능 문제를 식별하고 분석하는 데 도움이 됩니다.

카운터 지표 차트에는 성능 카운터의 데이터가 표시됩니다.



지표 관리를 선택하여 성능 카운터를 변경합니다. 다음 스크린샷과 같이 여러 OS 메트릭을 선택할 수 있습니다. 지표에 대한 세부 정보를 보려면 지표 이름 위에 마우스 포인터를 놓습니다.



데이터베이스 로드 차트

데이터베이스 로드 차트는 최대 vCPU 선으로 표시된 바와 같이 데이터베이스 활동과 인스턴스 용량을 비교하는 방식을 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 이미지는 Amazon DocumentDB 인스턴스의 치수를 보여줍니다.

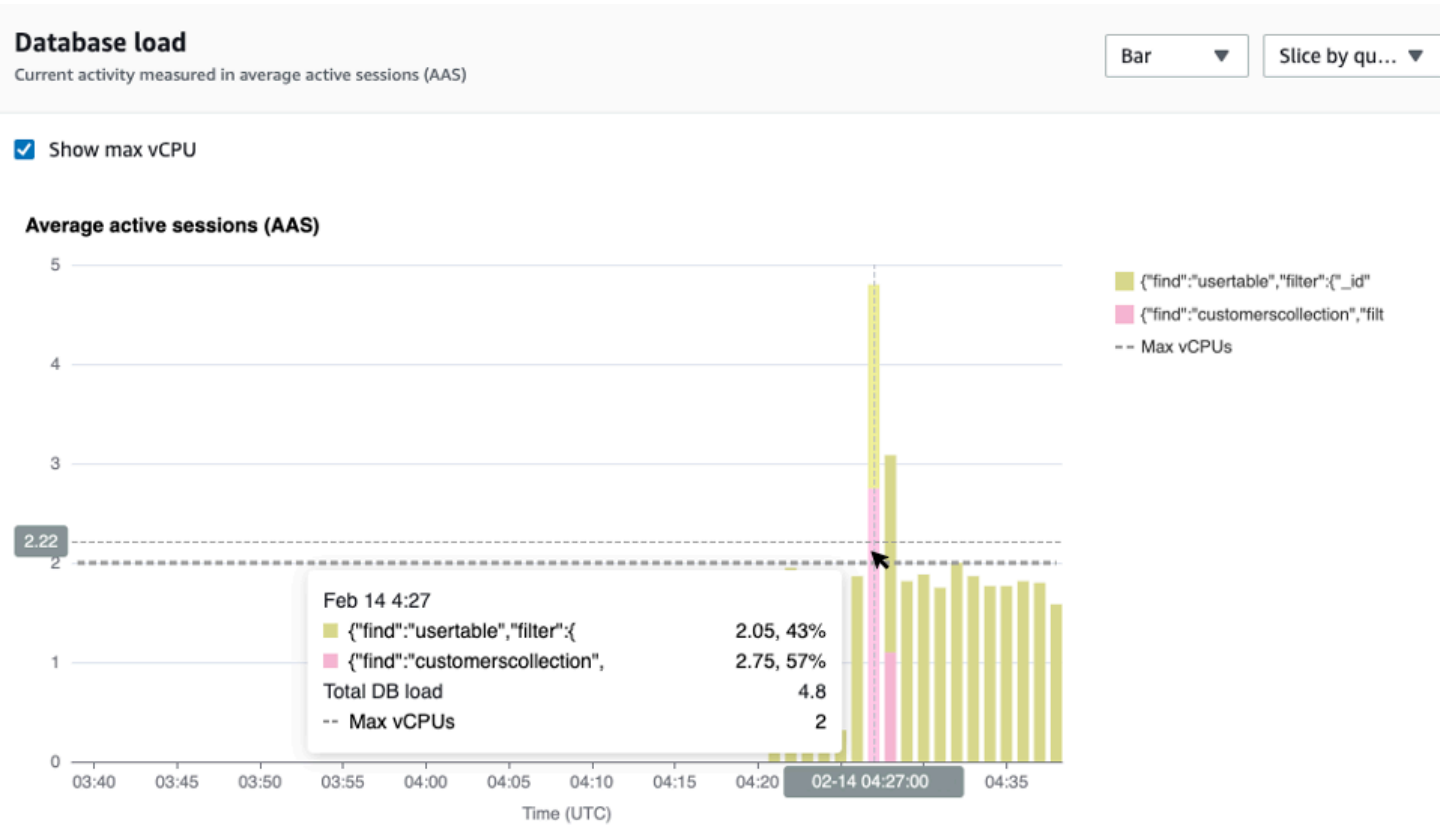


차원 항목에 대한 DB 로드 세부 정보

차원 내의 DB 로드 항목에 대한 세부 정보를 보려면 항목 이름 위로 마우스를 가져갑니다. 다음 이미지는 쿼리문에 대한 세부 정보를 보여줍니다.



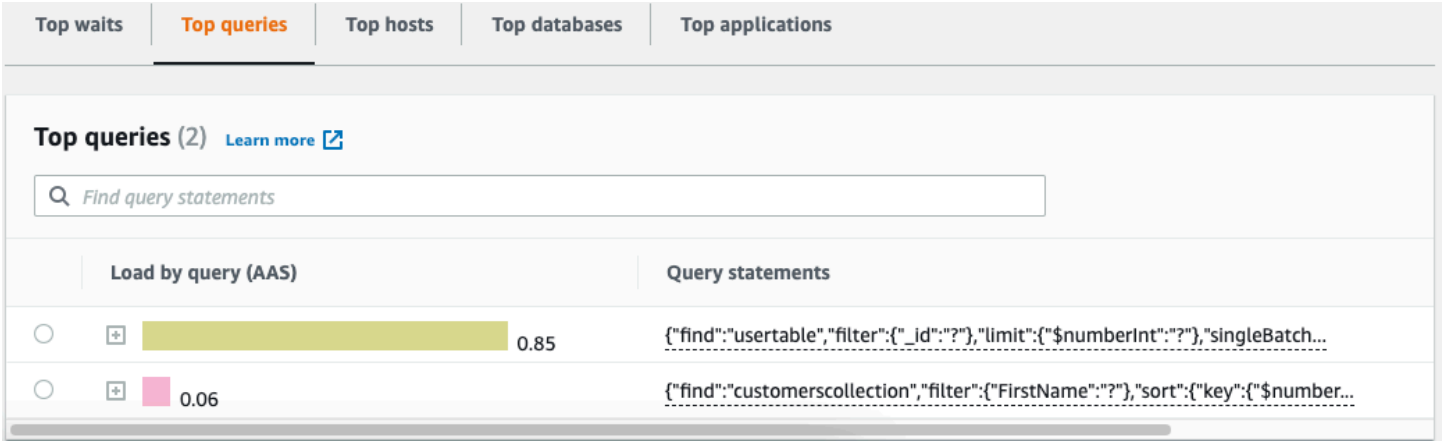
범례에서 선택한 기간의 항목에 대한 세부 정보를 보려면 해당 항목 위에 마우스 포인터를 놓습니다.



상위 측정기준 테이블

상위 측정기준 테이블은 DB 로드를 다른 차원으로 슬라이스합니다. 차원은 DB 로드의 다양한 특성에 대한 카테고리 또는 "슬라이스"입니다. 측정기준이 쿼리인 경우 상위 쿼리(Top queries)에서는 DB 로드에 가장 많이 기여하는 쿼리 문을 보여줍니다.

다음 차원 탭 중 하나를 선택합니다.



다음 표는 각 탭에 대한 간략한 설명을 제공합니다.

탭 명	설명
상위 대기	대기 이벤트가 백엔드가 대기 중인 이

필드명

이벤트

현재 실행 중인 쿼리문

연결된 클라이언트의 호스트 IP 및 포트

테이블 이름

데이터베이스의 이름

사용자 및 데이터베이스 연결

연결된 데이터베이스 사용자의 이름

필드	설명
세션	현재 애플리케이션에 연결된 애플리케이션의 이름

상위 쿼리(Top queries) 탭을 사용하여 쿼리를 분석하는 방법을 알아보려면 상위 쿼리 탭 개요 섹션을 참조하세요.

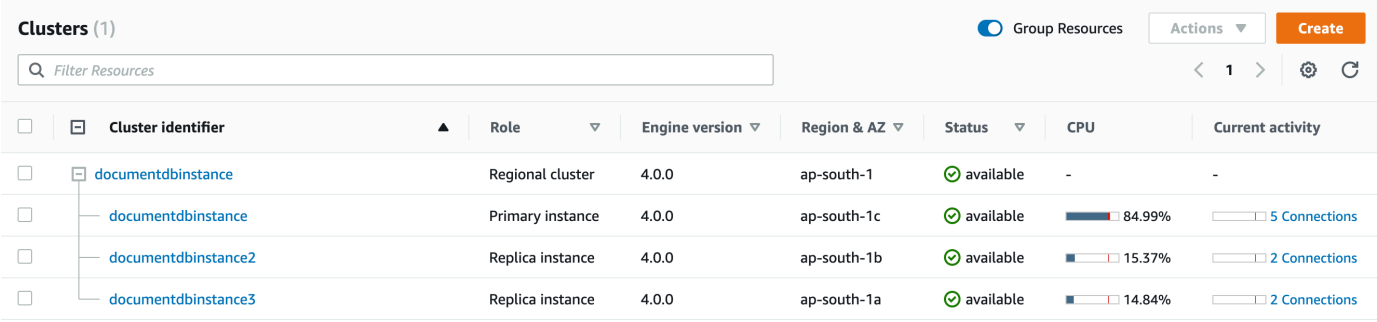
성능 개선 도우미 대시보드 열기

AWS 관리 콘솔에서 성능 개선 도우미 대시보드를 보려면 다음 단계를 사용합니다.

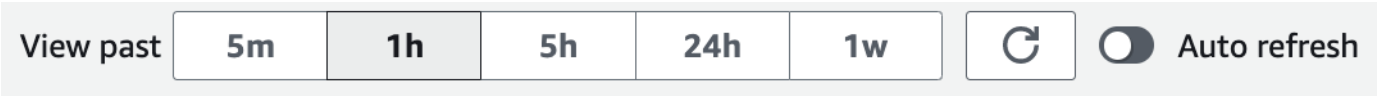
1. https://console.aws.amazon.com/docdb/ 에서 성능 개선 도우미 콘솔을 엽니다.
2. DB 인스턴스를 선택합니다. Amazon DocumentDB 인스턴스에 대한 성능 개선 도우미 대시보드가 표시됩니다.

   성능 개선 도우미가 사용 가능한 Amazon DocumentDB 인스턴스의 경우, 인스턴스 목록에서 세션 항목을 선택하여 대시보드에 도달할 수도 있습니다. 현재 활동에서 세션 항목은 지난 5분 동안 평균 활동 세션의 데이터베이스 로드를 보여 줍니다. 로드가 막대 모양으로 표시됩니다. 막대가 비어

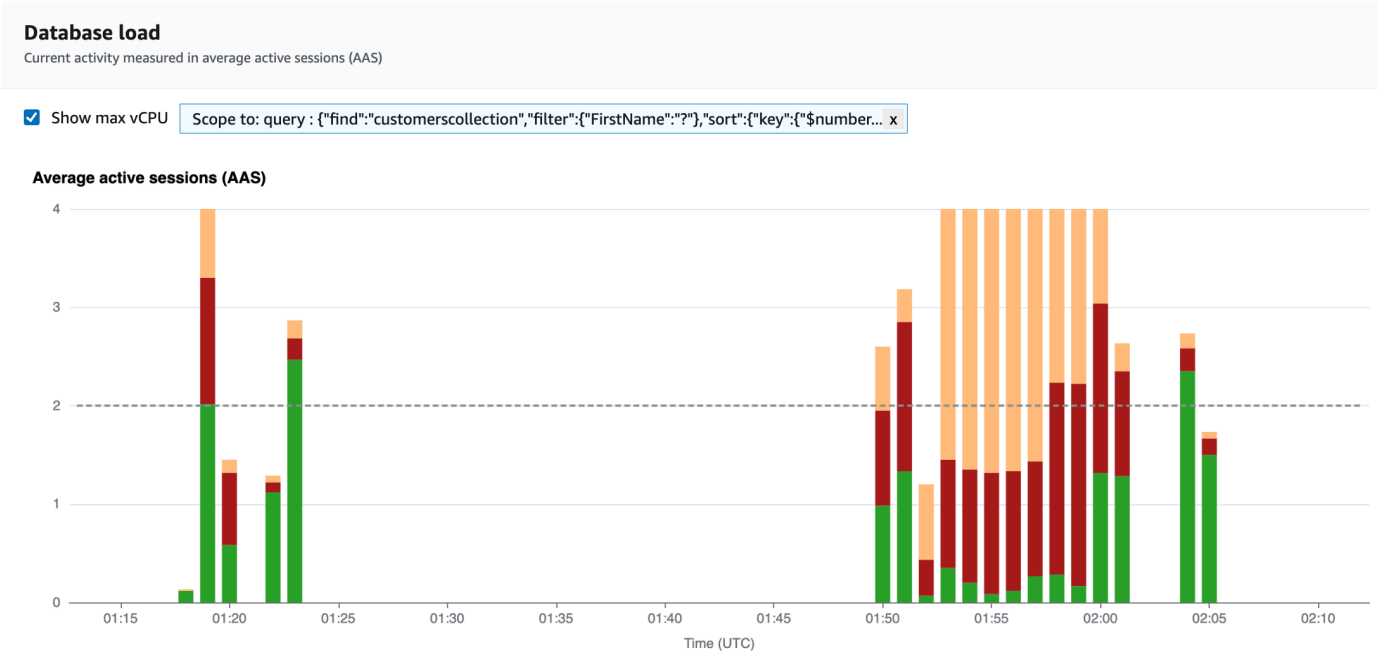
있으면 DB 인스턴스가 유휴 상태임을 뜻합니다. 로드가 증가하면 막대가 파란색으로 채워집니다. 로드에서 인스턴스 클래스의 가상 CPU(vCPU) 수를 전달하면 막대가 빨간색으로 바뀌고 병목 가능성을 나타냅니다.



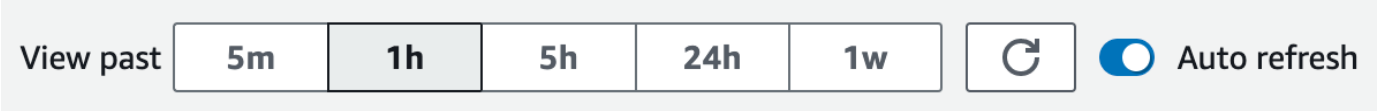
3. (선택 사항) 오른쪽 상단의 버튼을 선택하여 다른 시간 간격을 선택합니다. 예를 들어 간격을 1시간으로 변경하려면 1h를 선택합니다.



다음 스크린샷에서 DB 로드 간격은 1시간입니다.



4. 데이터를 자동으로 새로 고치려면 자동 새로 고침을 활성화합니다.



성능 개선 도우미 대시보드는 새 데이터로 자동으로 고쳐집니다. 새로 고침 속도는 표시되는 데이터의 양에 따라 다릅니다:

- 5분은 5초마다 새로 고칩니다.
- 1시간은 1분마다 새로 고칩니다.
- 5시간은 1분마다 새로 고칩니다.
- 24시간은 5분마다 새로 고칩니다.
- 1주는 1시간마다 새로 고칩니다.

대기 상태별 데이터베이스 로드 분석

데이터베이스 로드(DB 로드) 차트에 병목현상이 나타나면 어디에서 로드가 발생하는지 알 수 있습니다. 이렇게 하려면 데이터베이스 로드 차트 아래의 상위 로드 항목 테이블을 살펴보세요. 특정 항목(예: 쿼리 또는 응용프로그램)을 선택하여 해당 항목을 드릴다운하고 세부 정보를 확인합니다.

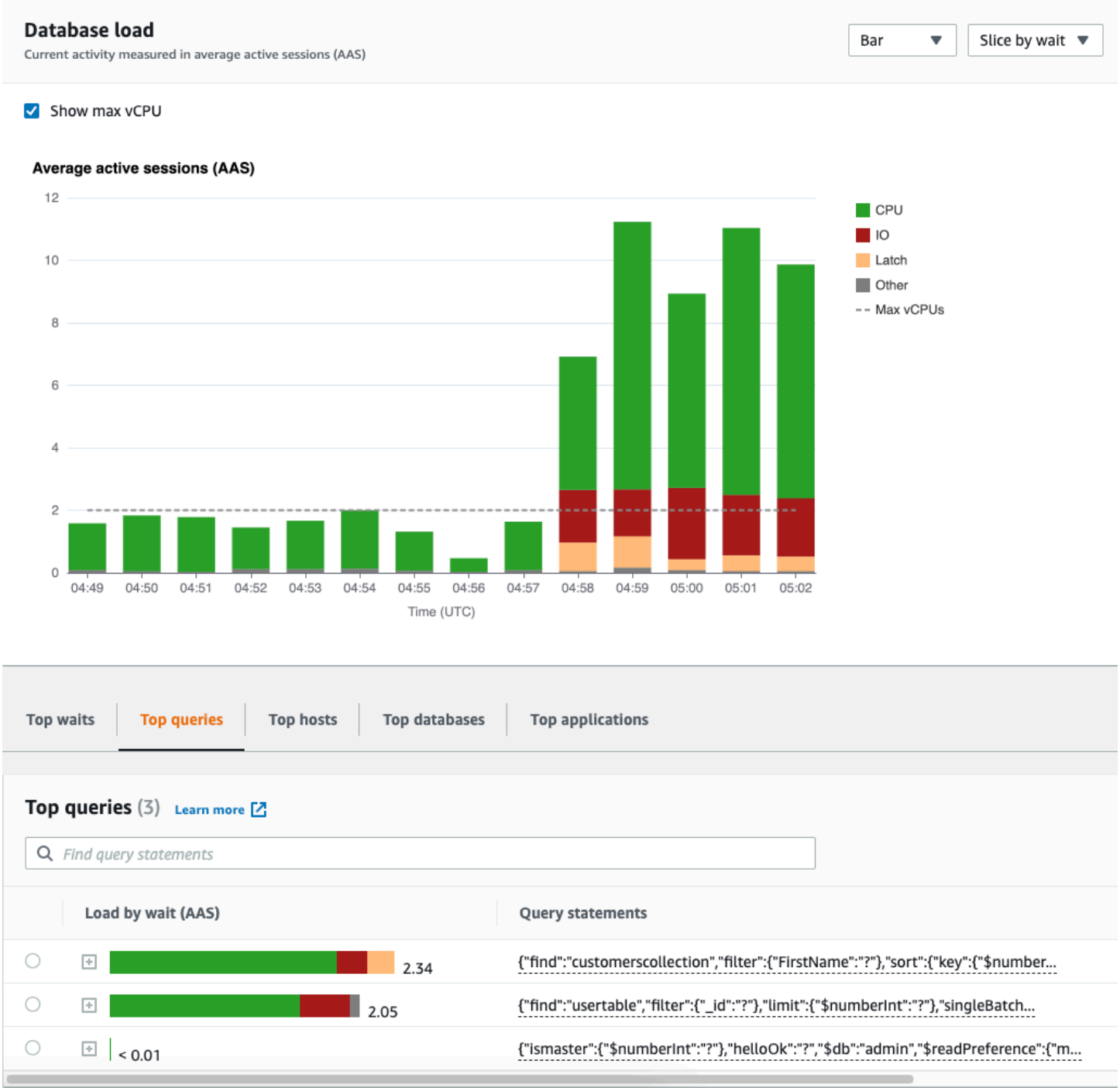
대기 및 상위 쿼리로 그룹화된 DB 로드는 일반적으로 성능 문제에 대한 가장 많은 통찰력을 제공합니다. 대기 상태를 기준으로 구분된 DB 로드는 데이터베이스의 리소스 또는 동시성 병목 현상 유무를 표시합니다. 이 경우, 상위 로드 항목 테이블의 상위 쿼리 탭에는 해당 로드를 구동하는 쿼리가 표시됩니다.

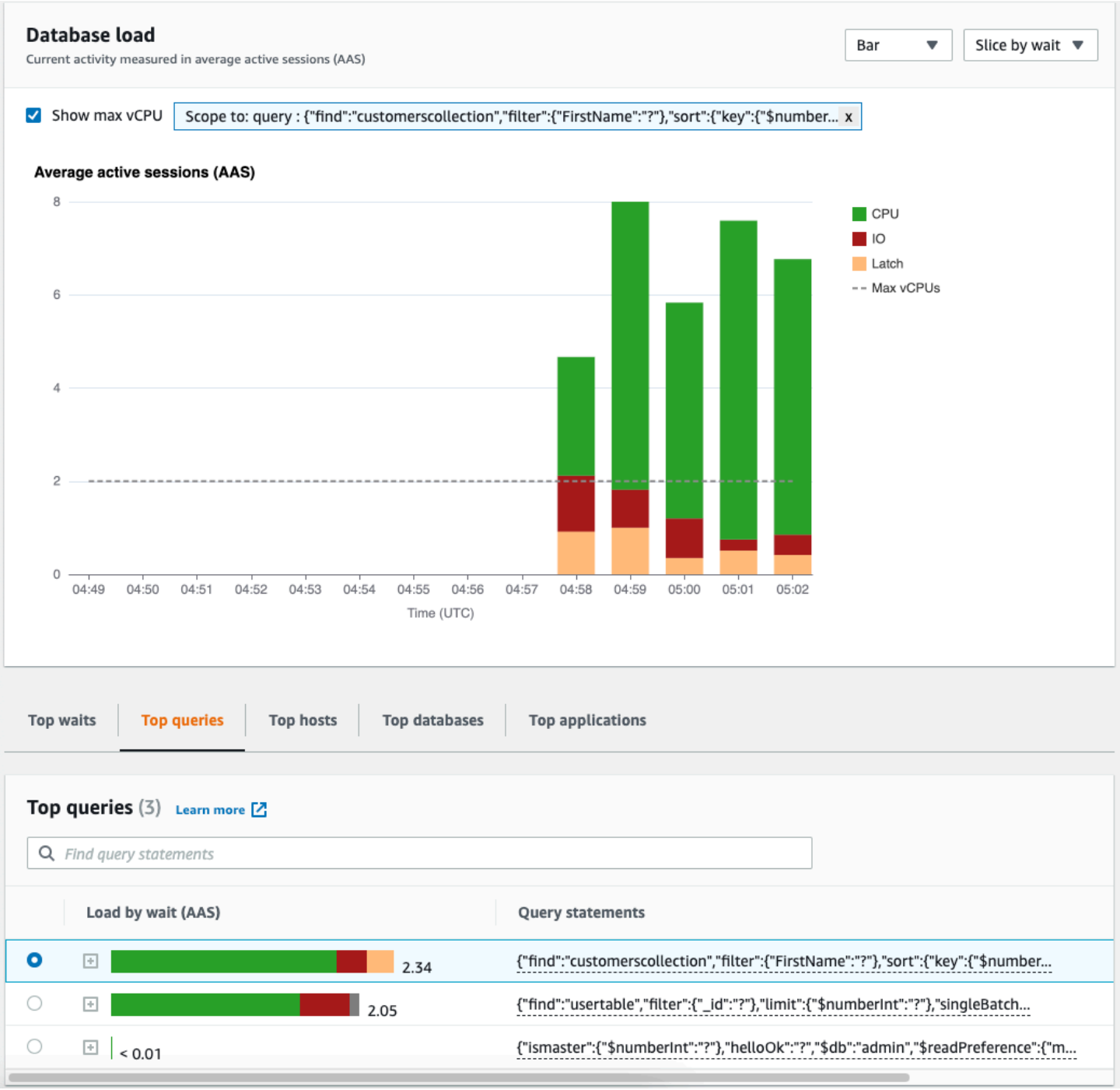
성능 문제를 진단하는 일반 워크플로우는 다음과 같습니다:

1. 데이터베이스 로드 차트를 보면서 데이터베이스 로드가 최대 CPU 선을 상회하는지 확인합니다.
2. 상회하는 경우가 있으면 데이터베이스 로드 차트를 보면서 원인이 되는 대기 상태를 식별합니다.
3. 상위 로드 항목 테이블의 상위 쿼리 탭이 해당 대기 상태에 가장 크게 기여하는 쿼리를 확인하여 로드를 유발하는 다이제스트 쿼리를 식별합니다. 이러한 내용들을 대기별 로드(AAS) 열로 식별할 수 있습니다.
4. 상위 쿼리 탭에서 이러한 다이제스트 쿼리 중 하나를 선택하여 해당 쿼리를 확장하고 해당 쿼리가 구성된 하위 쿼리를 확인합니다.

또한 상위 호스트 또는 상위 애플리케이션을 각각 선택하면 어떤 호스트나 애플리케이션이 가장 많은 로드를 주고 있는지 확인할 수 있습니다. 애플리케이션 이름은 Amazon DocumentDB 인스턴스에 대한 연결 문자열에 지정되어 있습니다. Unknown은 애플리케이션 필드가 지정되지 않았음을 나타냅니다.

예를 들어 다음 대시보드에서 CPU 대기 시간은 대부분의 DB 로드를 차지합니다. 상위 쿼리에서 상위 쿼리를 선택하면 데이터베이스 로드 차트의 범위가 선택 쿼리로 인해 발생한 가장 많은 로드를 집중적으로 표시합니다.





상위 쿼리 탭 개요

기본적으로 상위 쿼리 탭에는 DB 로드에 가장 큰 기여를 하는 쿼리가 표시됩니다. 쿼리 텍스트를 분석하여 쿼리를 조정하는 데 도움이 될 수 있습니다.

주제

- 쿼리 다이제스트

- 대기별 로드(AAS)
- 자세한 쿼리 정보 보기
- 명령문 쿼리 텍스트에 액세스
- 명령문 쿼리 텍스트 보기 및 다운로드

쿼리 다이제스트

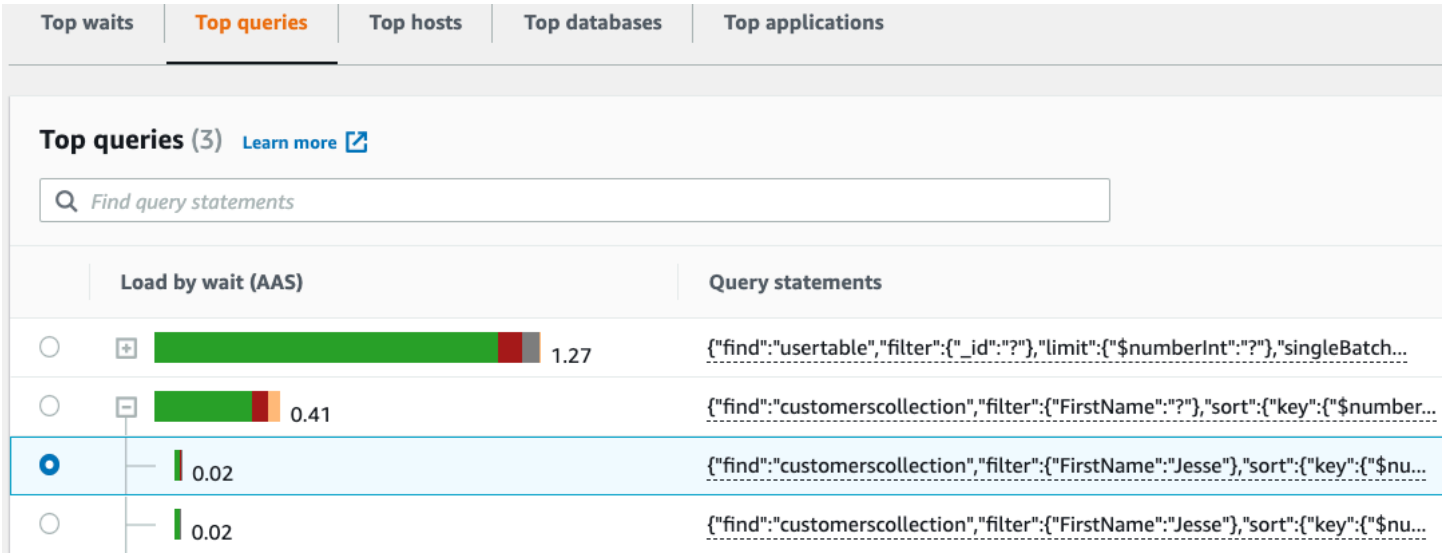
쿼리 다이제스트는 구조적으로 유사하지만 다른 리터럴 값을 가질 수 있는 여러 실제 쿼리의 합성입니다. 다이제스트는 하드 코딩된 값을 물음표로 바꿉니다. 예를 들어 쿼리 다이제스트는 다음과 같습니다:

```
{"find":"customerscollection","filter":{"FirstName":"?"},"sort":{"key":
{"$numberInt":"?"}},"limit":{"$numberInt":"?"}}
```

이 다이제스트에는 다음 하위 쿼리가 포함될 수 있습니다:

```
{"find":"customerscollection","filter":{"FirstName":"Karrie"},"sort":{"key":
{"$numberInt":"1"}},"limit":{"$numberInt":"3"}}
{"find":"customerscollection","filter":{"FirstName":"Met"},"sort":{"key":
{"$numberInt":"1"}},"limit":{"$numberInt":"3"}}
{"find":"customerscollection","filter":{"FirstName":"Rashin"},"sort":{"key":
{"$numberInt":"1"}},"limit":{"$numberInt":"3"}}
```

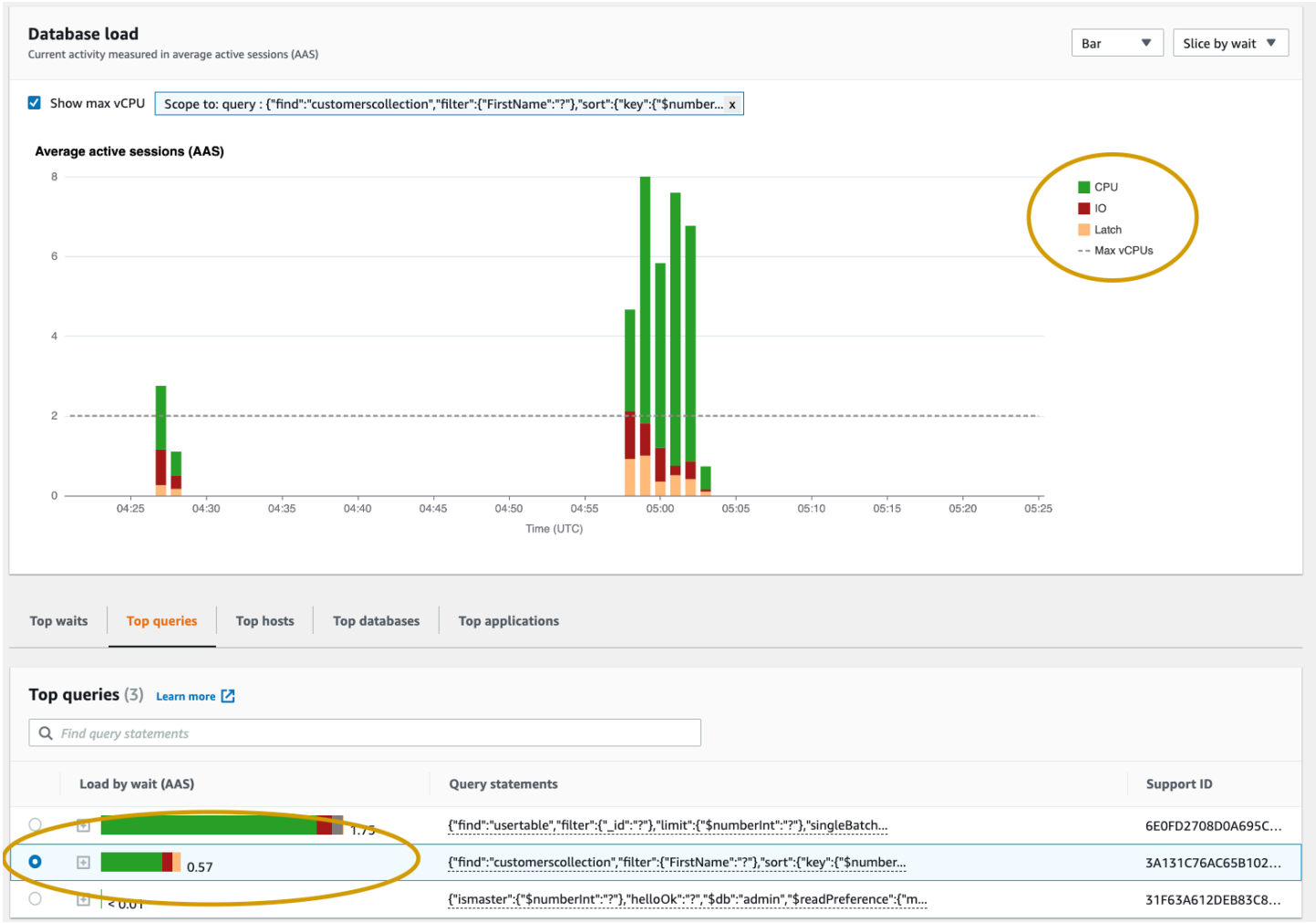
요약에서 문자 그대로의 쿼리 문을 보려면 쿼리를 선택한 다음 더하기 기호(+)를 선택합니다. 다음 스크린샷에서 선택한 쿼리는 다이제스트입니다.



> **Note**
>
> 쿼리 요약은 유사한 쿼리 문을 그룹화하지만 중요한 정보는 수정하지 않습니다.

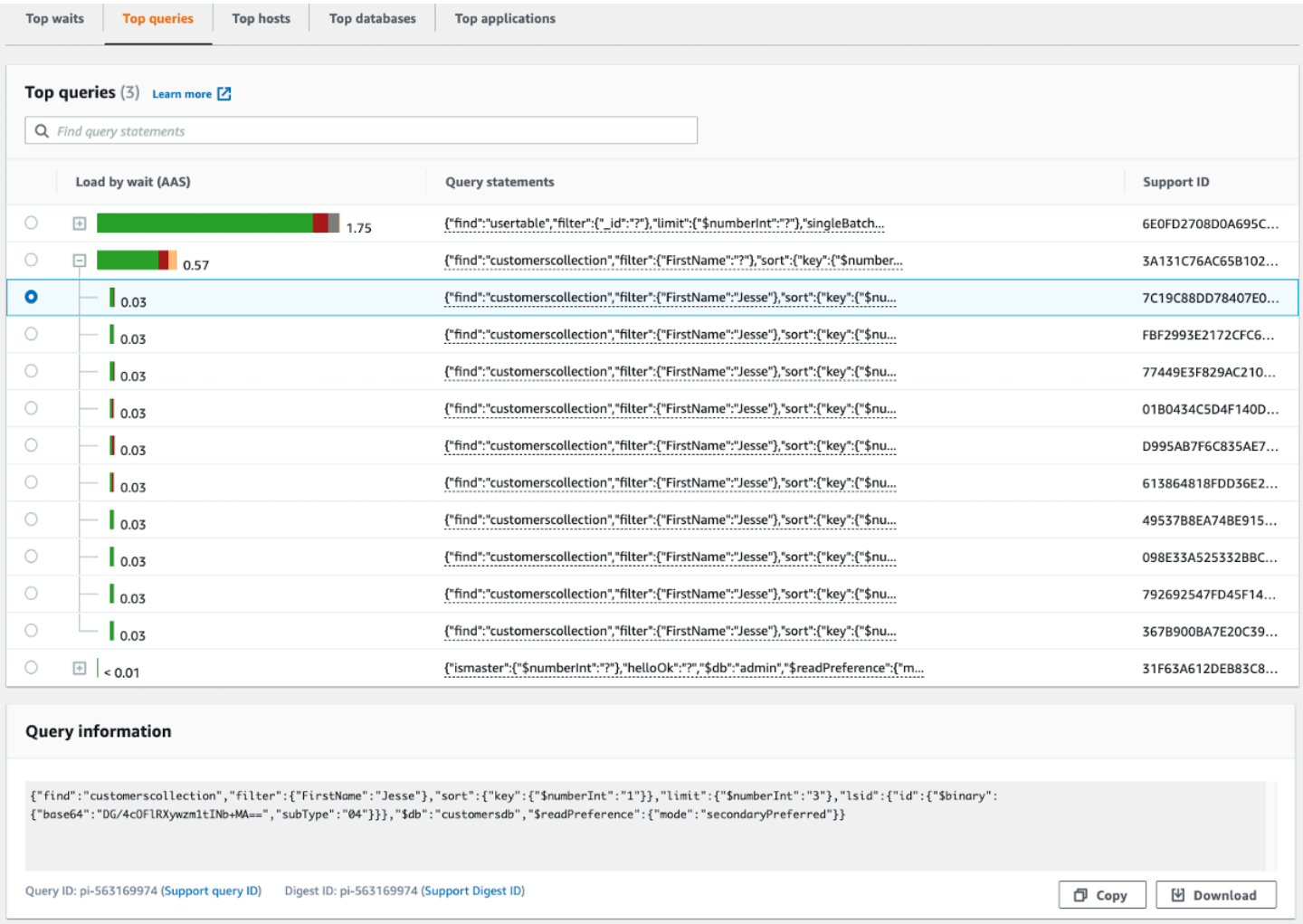
대기별 로드(AAS)

상위 쿼리에서 대기별 로드(AAS) 열은 각 상위 로드 항목과 관련된 데이터베이스 로드의 백분율을 나타냅니다. 이 열에는 현재 DB 로드 차트에서 어떤 그룹화 기준을 선택하든 그 기준에 따라 해당 항목의 로드가 반영됩니다. 예를 들어 DB 로드차트를 대기 상태별로 그룹화할 수 있습니다. 이 경우 대기별 DB 로드 막대는 쿼리가 영향을 미치는 대기 상태의 정도를 크기, 세그먼트 및 컬러 코드로 표시합니다. 또한 선택한 쿼리에 영향을 미치는 대기 상태를 표시합니다.



자세한 쿼리 정보 보기

상위 쿼리 표에서, 다이제스트 문을 열어 해당 정보를 볼 수 있습니다. 맨 아래 창에 정보가 나타납니다.



다음과 같은 식별자(ID) 유형은 쿼리 문과 연결되어 있습니다.

1. 쿼리 ID 지원 – 쿼리 ID의 해시 값입니다. 이 값은 AWS Support로 작업할 때 쿼리 ID를 참조하기 위한 것입니다. AWS Support는 실제 쿼리 IDs 및 쿼리 텍스트에 액세스할 수 없습니다.
2. 다이제스트 ID 지원 – 다이제스트 ID의 해시 값입니다. 이 값은 AWS Support로 작업할 때 다이제스트 ID를 참조하기 위한 것입니다. AWS Support는 실제 다이제스트 IDs 및 쿼리 텍스트에 액세스할 수 없습니다.

명령문 쿼리 텍스트에 액세스

기본적으로 상위 쿼리 테이블의 각 행은 각 쿼리문에 대해 500바이트의 쿼리 텍스트를 보여줍니다. 다이제스트 문이 500바이트 이상인 경우 성능 개선 도우미 대시보드에서 해당 문을 열어 더 많은 텍스트를 볼 수 있습니다. 이 경우 표시되는 쿼리의 최대 길이는 1KB입니다. 하위 SQL 문을 보는 경우 다운로드를 선택할 수도 있습니다.

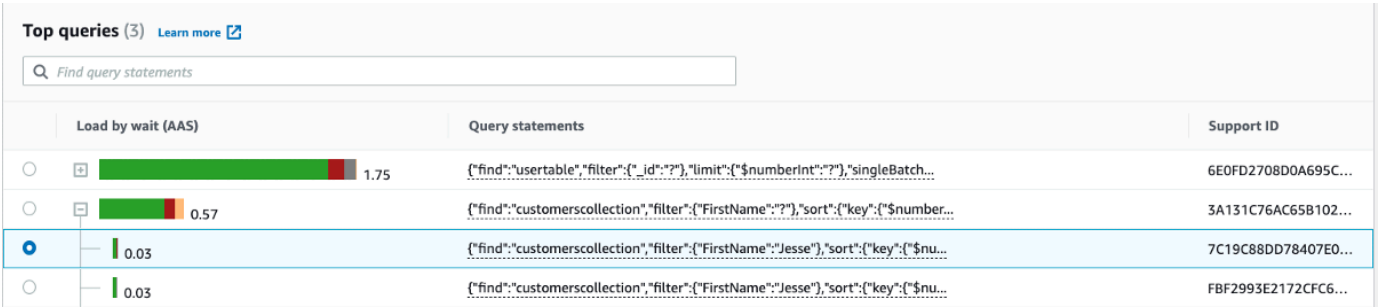
명령문 쿼리 텍스트 보기 및 다운로드

성능 개선 도우미 대시보드에서 쿼리 텍스트를 보거나 다운로드할 수 있습니다.

성능 개선 도우미 대시보드에서 더 많은 쿼리 텍스트를 보려면 다음과 같이 하십시오

1. https://console.aws.amazon.com/docdb/ 에서 Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 성능 개선 도우미을 선택합니다.
3. DB 인스턴스를 선택합니다. 선택한 DB 인스턴스에 대한 성능 개선 도우미 대시보드가 표시됩니다.

   500바이트 이상의 텍스트가 있는 SQL 문은 다음 이미지와 유사합니다:



4. 쿼리 정보 섹션을 조사하여 쿼리 텍스트를 더 확인합니다.



성능 개선 도우미 대시보드는 전체 쿼리문마다 최대 1KB까지 표시할 수 있습니다.

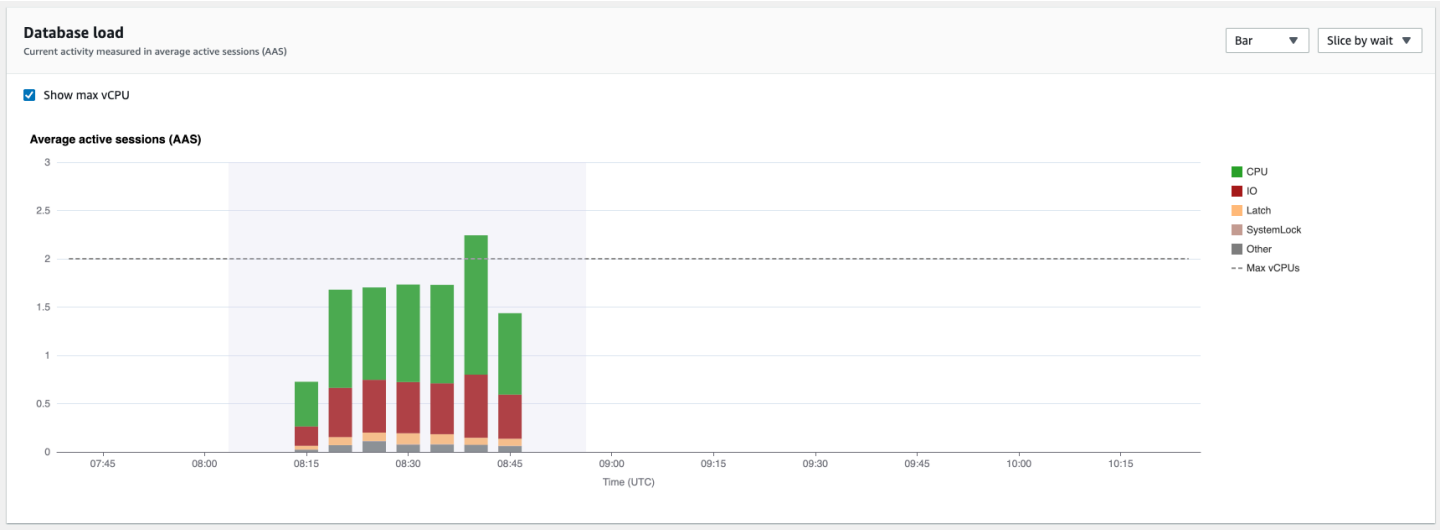
> **Note**
>
> 쿼리문을 복사하거나 다운로드하려면 팝업 차단을 비활성화합니다.

데이터베이스 로드 차트에서 확대

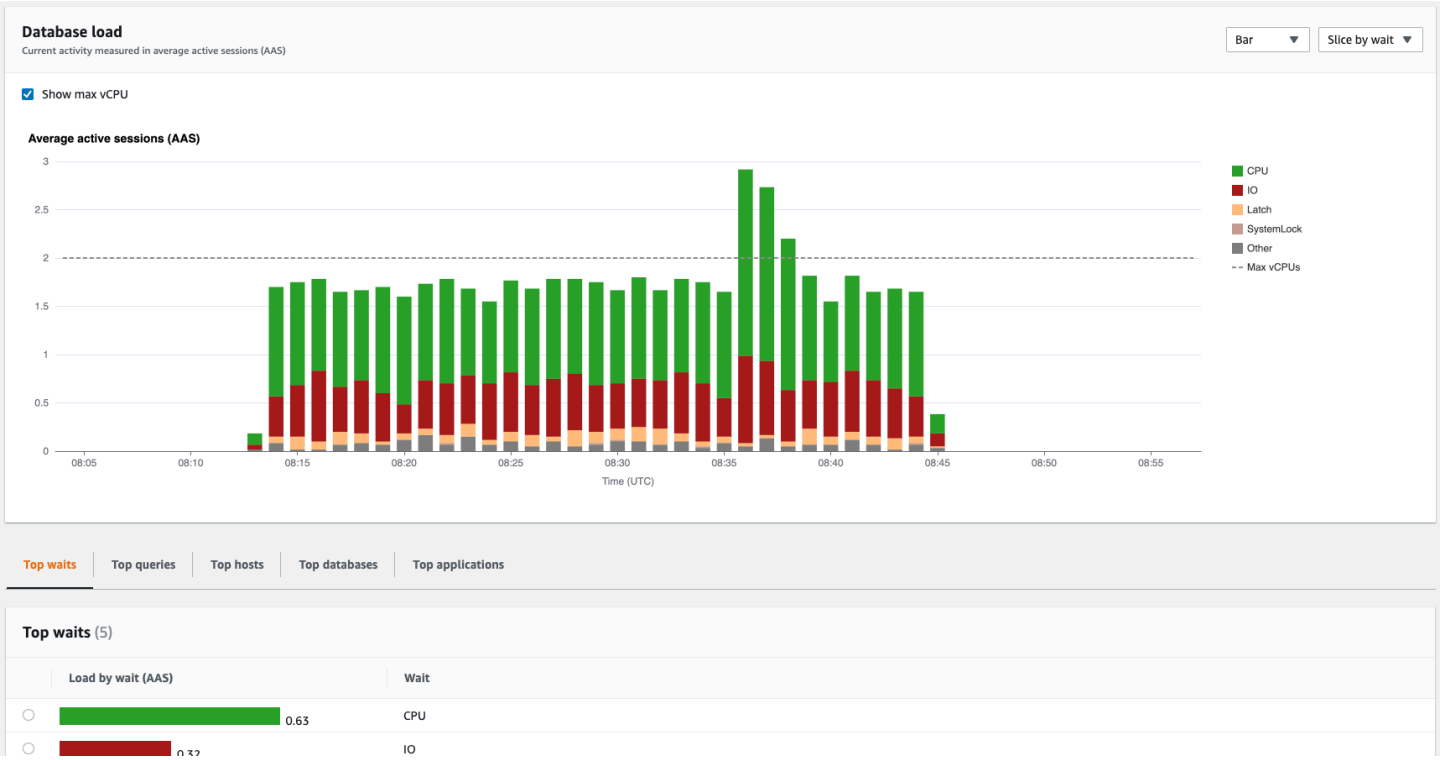
성능 개선 도우미 사용자 인터페이스의 다른 특성을 사용해 성능 데이터를 분석할 수 있습니다.

클릭하여 끌어 확대

성능 개선 도우미 인터페이스에서 로드 차트의 작은 부분을 선택하여 확대해 자세히 볼 수 있습니다.



로드 차트의 한 부분을 확대하려면 시작 시간을 선택하고 원하는 기간 끝까지 끕니다. 이렇게 하면 선택한 영역이 강조 표시됩니다. 마우스를 놓으면 로드 차트가 선택한 영역을 확대하고 상위 항목 테이블이 다시 계산됩니다.



성능 개선 도우미 API를 사용하여 지표 검색

성능 개선 도우미를 활성화하면 API에서 인스턴스 성능에 대한 가시성을 제공합니다. Amazon CloudWatch Logs는 AWS 서비스에 대한 벤딩 모니터링 지표에 대한 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 로드를 분해합니다(예: `Query`, `Wait-state`, `Application` 또는 `Host`).

Amazon DocumentDB 성능 개선 도우미는 데이터베이스 성능을 분석하고 문제를 해결할 수 있도록 Amazon DocumentDB 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 DocumentDB 인스턴스 중 하나에서 성능 개선 도우미를 활성화합니다. 성능 개선 도우미 활성화에 대한 자세한 내용은 성능 개선 도우미 활성화 및 비활성화(을)를 참조하십시오. 성능 개선 도우미 API에 대한 자세한 내용은 성능 개선 도우미 API 참조를 참조하십시오.

성능 개선 도우미 API에서는 다음과 같은 작업을 제공합니다.

성능 개선 도우미 작업	AWS CLI 명령	설명
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 쿼리 ID를 지정하고 차원 세부 정보를 사용할 수 있는 경

성능 개선 도우미 작업	AWS CLI 명령	설명
		우 GetDimensionKeyDetails 는 이 ID에 연결된 차원 db.query.statement 의 전체 텍스트를 검색합니다. GetResourceMetrics 및 DescribeDimensionKeys 는 대용량 쿼리 문 텍스트 검색을 지원하지 않으므로 이 작업을 유용하게 사용할 수 있습니다.
GetResourceMetadata	aws pi get-resource-metadata	다양한 기능에 대한 특성을 검색합니다. 예를 들어 메타데이터는 특정 DB 인스턴스에서 특성이 켜지거나 꺼짐을 나타낼 수 있습니다.
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.

주제

- AWS CLI 성능 개선 도우미용

- 시계열 지표 조회
- AWS CLI 성능 개선 도우미 예제

AWS CLI 성능 개선 도우미용

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.

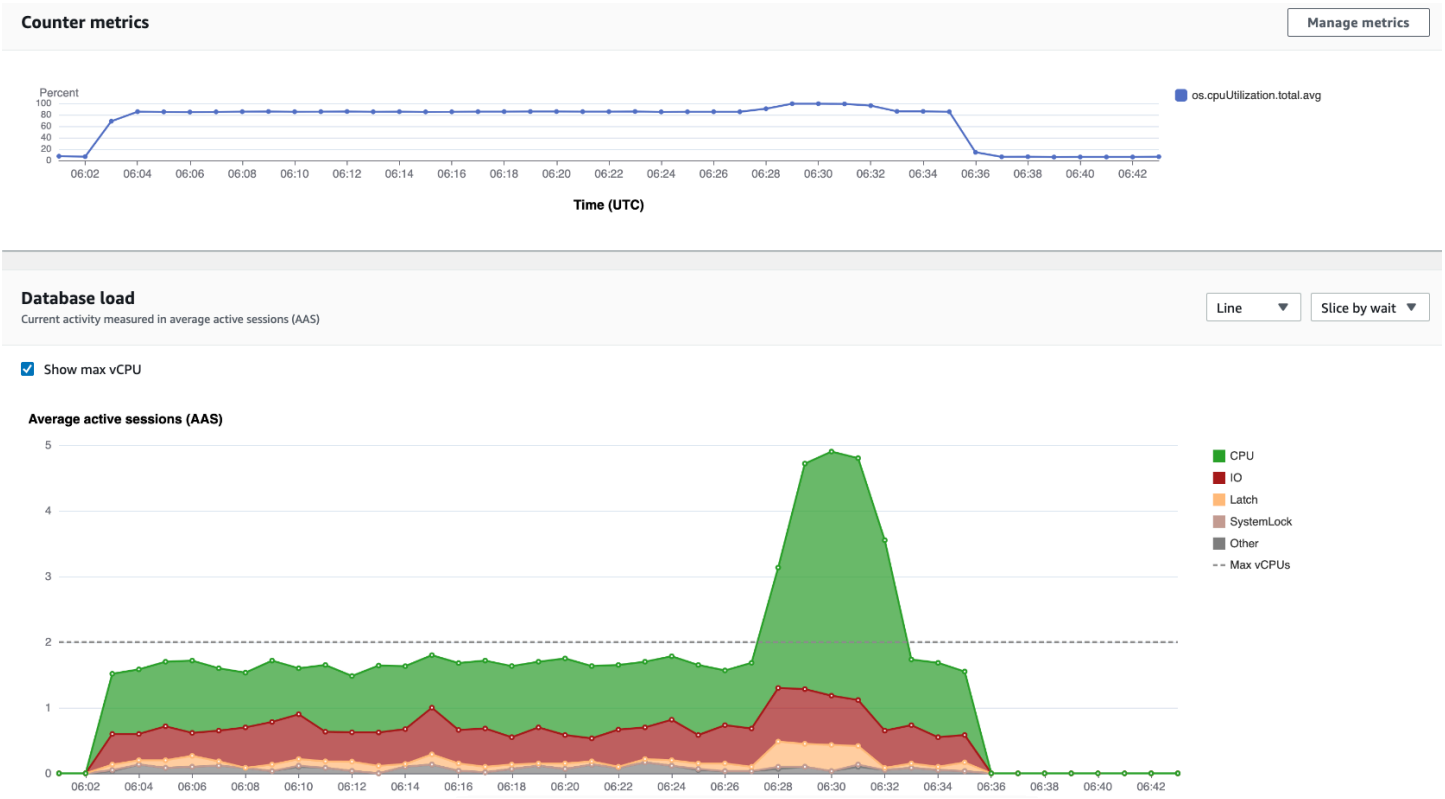
```
aws pi help
```

가 AWS CLI 설치되어 있지 않은 경우 설치에 대한 자세한 내용은 사용 설명서의 AWS 명령줄 인터페이스 설치를 참조하세요. AWS CLI

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 들어는 다음 이미지와 같이를 AWS Management Console `GetResourceMetrics` 사용하여 카운터 지표 차트와 데이터베이스 로드 차트를 채웁니다.



GetResourceMetrics에서 반환하는 지표는 db.load를 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.load는 db.load를 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 카운터 지표에 대한 성능 개선 도우미를 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 섹션을 참조하세요get-resource-metrics.

`--metric-queries` 옵션의 경우 결과를 얻고자 하는 쿼리를 한 개 이상 지정하십시오. 각 쿼리는 필수인 `Metric`과 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"
     ...}
```

AWS CLI 성능 개선 도우미 예제

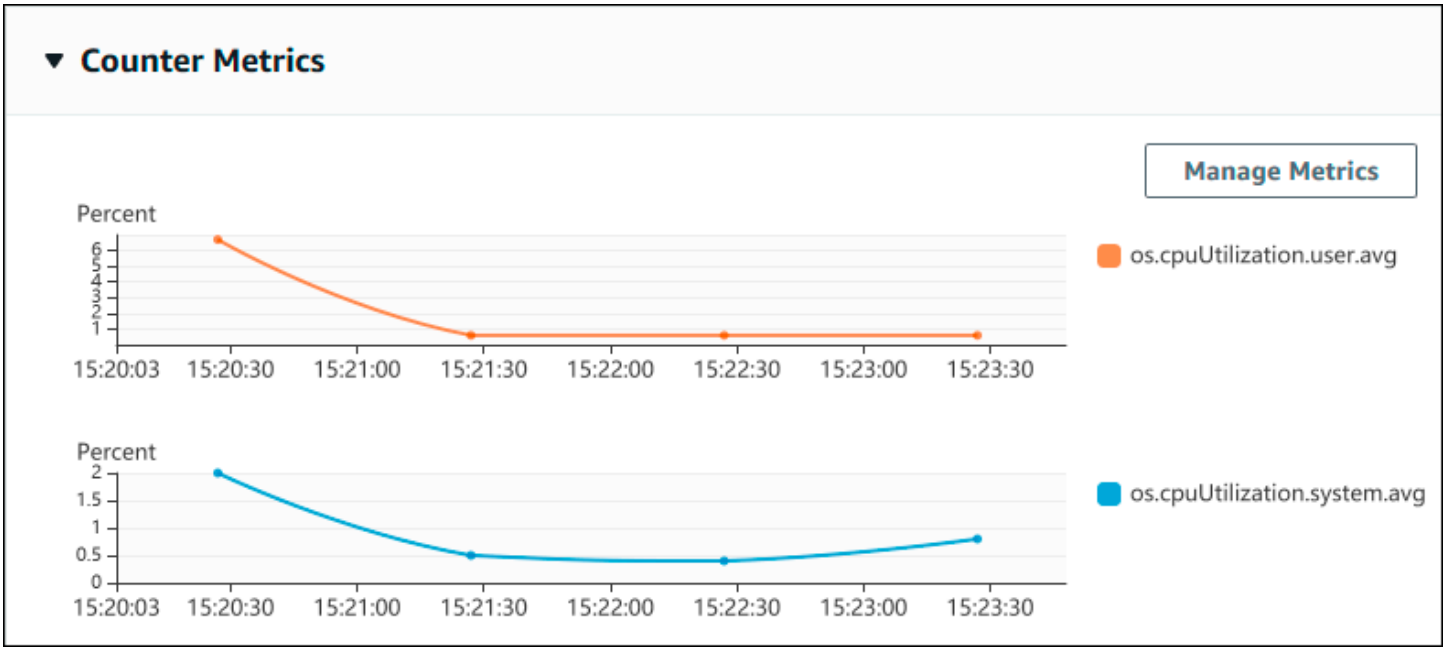
다음 예제에서는 성능 개선 도우미 AWS CLI 용를 사용하는 방법을 보여줍니다.

주제

- 카운터 지표 검색
- 상위 대기 상태에 대한 DB 로드 평균 검색
- 상위 쿼리에 대한 DB 평균 로드 검색
- 쿼리로 필터링된 DB 로드 평균 검색

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 Console 이 사용하는 것과 동일한 데이터를 수집하는 방법을 보여줍니다.

Linux, macOS 또는 Unix의 경우는 다음과 같습니다:

```
aws pi get-resource-metrics \
   --service-type DOCDB \
   --identifier db-ID \
   --start-time 2022-03-13T8:00:00Z \
   --end-time   2022-03-13T9:00:00Z \
   --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 DOCDB ^
   --identifier db-ID ^
   --start-time 2022-03-13T8:00:00Z ^
   --end-time   2022-03-13T9:00:00Z ^
   --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.

Linux, macOS 또는 Unix의 경우는 다음과 같습니다:

```
aws pi get-resource-metrics \
   --service-type DOCDB \
   --identifier db-ID \
   --start-time 2022-03-13T8:00:00Z \
   --end-time   2022-03-13T9:00:00Z \
   --period-in-seconds 60 \
   --metric-queries file://query.json
```

Windows의 경우:

```
aws pi get-resource-metrics ^
   --service-type DOCDB ^
   --identifier db-ID ^
   --start-time 2022-03-13T8:00:00Z ^
   --end-time   2022-03-13T9:00:00Z ^
   --period-in-seconds 60 ^
   --metric-queries file://query.json
```

앞의 예에서는 옵션에 다음 값을 지정합니다:

- `--service-type` – Amazon DocumentDB 용 `DOCDB`
- `--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.

```
{
    "AlignedStartTime": "2022-03-13T08:00:00+00:00",
    "AlignedEndTime": "2022-03-13T09:00:00+00:00",
    "Identifier": "db-NQF3TTMFQ3GTOKIMJODMC3KQQ4",
    "MetricList": [
        {
            "Key": {
                "Metric": "os.cpuUtilization.user.avg"
            },
            "DataPoints": [
                {
                    "Timestamp": "2022-03-13T08:01:00+00:00", //Minute1
                    "Value": 3.6
                },
                {
                    "Timestamp": "2022-03-13T08:02:00+00:00", //Minute2
                    "Value": 2.6
                },
                //.... 60 datapoints for the os.cpuUtilization.user.avg metric
        {
            "Key": {
                "Metric": "os.cpuUtilization.idle.avg"
            },
            "DataPoints": [
                {
                    "Timestamp": "2022-03-13T08:01:00+00:00",
                    "Value": 92.7
                },
                {
                    "Timestamp": "2022-03-13T08:02:00+00:00",
                    "Value": 93.7
```

```
                },
                //.... 60 datapoints for the os.cpuUtilization.user.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 상태에 대한 DB 로드 평균 검색

다음 예제는가 스택 영역 선 그래프를 생성하는 데 AWS Management Console 사용하는 것과 동일한 쿼리입니다. 이 예제에서는 로드를 상위 7개 대기 상태에 따라 나눈 마지막 시간 동안 db.load.avg을 검색합니다. 명령은 카운터 지표 검색의 명령과 동일합니다. 그러나 query.json 파일의 컨텐츠는 다음과 같습니다.

```
[
    {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.wait_state", "Limit": 7 }
    }
]
```

다음 명령을 실행합니다.

Linux, macOS 또는 Unix의 경우는 다음과 같습니다:

```
aws pi get-resource-metrics \
   --service-type DOCDB \
   --identifier db-ID \
   --start-time 2022-03-13T8:00:00Z \
   --end-time   2022-03-13T9:00:00Z \
   --period-in-seconds 60 \
```

```
    --metric-queries file://query.json
```

Windows의 경우:

```
aws pi get-resource-metrics ^
   --service-type DOCDB ^
   --identifier db-ID ^
   --start-time 2022-03-13T8:00:00Z ^
   --end-time   2022-03-13T9:00:00Z ^
   --period-in-seconds 60 ^
   --metric-queries file://query.json
```

예제에서는 상위 7개 대기 상태 중 db.load.avg과 GroupBy의 메트릭을 지정합니다. 이 예의 유효 값에 대한 자세한 내용은 성능 개선 도우미 API 참조의 DimensionGroup을 참조하십시오.

응답은 다음과 비슷합니다.

```
{
    "AlignedStartTime": "2022-04-04T06:00:00+00:00",
    "AlignedEndTime": "2022-04-04T06:15:00+00:00",
    "Identifier": "db-NQF3TTMFQ3GTOKIMJODMC3KQQ4",
    "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": "2022-04-04T06:01:00+00:00",//Minute1
                    "Value": 0.0
                },
                {
                    "Timestamp": "2022-04-04T06:02:00+00:00",//Minute2
                    "Value": 0.0
                },
                //... 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_state.name": "CPU"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": "2022-04-04T06:01:00+00:00",//Minute1
                    "Value": 0.0
                },
                {
                    "Timestamp": "2022-04-04T06:02:00+00:00",//Minute2
                    "Value": 0.0
                },
                //... 60 datapoints for the CPU key
            ]
        },//... In total we have 3 key/datapoints entries, 1) total, 2-3) Top Wait
 States
    ] //end of MetricList
} //end of response
```

이 응답에는 MetricList에 항목이 3개 있습니다. 총 db.load.avg에 대해 하나의 엔트리가 있으며, 상위 3개 대기 상태 중 하나에 따라 나누어진 db.load.avg에 대해 각각 3개의 엔트리가 있습니다. 그룹화 차원이 있었기 때문에(첫 번째 예제와 달리) 메트릭의 각 그룹화에는 하나의 키가 있어야 합니다. 기본 카운터 지표 사용 사례처럼 각 지표에 키가 한 개만 있을 수는 없습니다.

상위 쿼리에 대한 DB 평균 로드 검색

다음 예에서는 상위 10개 쿼리 문을 기준으로 db.wait_state를 그룹화합니다. SQL 문에는 두 가지 그룹이 있습니다:

- db.query – {"find":"customers","filter":{"FirstName":"Jesse"},"sort": {"key":{"$numberInt":"1"}}}와 같은 전체 쿼리문
- db.query_tokenized – {"find":"customers","filter": {"FirstName":"?"},"sort":{"key":{"$numberInt":"?"}},"limit": {"$numberInt":"?"}}와 같은 토큰화된 쿼리문

데이터베이스 성능을 분석할 때는 파라미터에 의해서만 다른 쿼리문을 하나의 논리 항목으로 고려하는 것이 유용할 수 있습니다. 따라서 쿼리 시에는 db.query_tokenized를 사용할 수 있습니다. 그러

나 특히 explain()에 관심이 있는 경우에는 때때로 파라미터가 있는 전체 쿼리 문을 검토하는 것이 더 유용합니다. 토큰화된 쿼리와 전체 쿼리 사이에는 부모-자녀 관계가 있으며, 여러 전체 쿼리(자녀)가 동일한 토큰화된 쿼리(부모) 아래에 그룹화됩니다.

이 예의 명령은 상위 대기 상태에 대한 DB 로드 평균 검색의 명령과 유사합니다. 그러나 query.json 파일의 컨텐츠는 다음과 같습니다.

```
[
    {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.query_tokenized", "Limit": 10 }
    }
]
```

다음 예에는 db.query_tokenized가 사용됩니다.

Linux, macOS 또는 Unix의 경우는 다음과 같습니다:

```
aws pi get-resource-metrics \
   --service-type DOCDB \
   --identifier db-ID \
   --start-time 2022-03-13T8:00:00Z \
   --end-time   2022-03-13T9:00:00Z \
   --period-in-seconds 3600 \
   --metric-queries file://query.json
```

Windows의 경우:

```
aws pi get-resource-metrics ^
   --service-type DOCDB ^
   --identifier db-ID ^
   --start-time 2022-03-13T8:00:00Z ^
   --end-time   2022-03-13T9:00:00Z  ^
   --period-in-seconds 3600 ^
   --metric-queries file://query.json
```

이 예에서는 1시간 동안 쿼리를 실행하는데 1분은 초 단위로 구성됩니다.

예제에서는 상위 7개 대기 상태 중 db.load.avg과 GroupBy의 메트릭을 지정합니다. 이 예의 유효 값에 대한 자세한 내용은 성능 개선 도우미 API 참조의 DimensionGroup을 참조하십시오.

응답은 다음과 비슷합니다.

```
{
    "AlignedStartTime": "2022-04-04T06:00:00+00:00",
    "AlignedEndTime": "2022-04-04T06:15:00+00:00",
    "Identifier": "db-NQF3TTMFQ3GTOKIMJODMC3KQQ4",
    "MetricList": [
        {//A list of key/datapoints
            "Key": {
                "Metric": "db.load.avg"
            },
            "DataPoints": [
                //... 60 datapoints for the total db.load.avg key
                ]
        },
               {
            "Key": {//Next key are the top tokenized queries
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.query_tokenized.db_id": "pi-1064184600",
                    "db.query_tokenized.id": "77DE8364594EXAMPLE",
                    "db.query_tokenized.statement": "{\"find\":\"customers\",\"filter
\":{\"FirstName\":\"?\"},\"sort\":{\"key\":{\"$numberInt\":\"?\"}},\"limit\"
:{\"$numberInt\":\"?\"},\"$db\":\"myDB\",\"$readPreference\":{\"mode\":\"primary\"}}"
                }
            },
            "DataPoints": [
            //... 60 datapoints
            ]
        },
        // In total 11 entries, 10 Keys of top tokenized queries, 1 total key
    ] //End of MetricList
} //End of response
```

이 응답은 `MetricList`에 11개의 항목이 있는데 (전체 1개, 최상위 토큰화 SQL 10개) 각 항목에는 시간당 `DataPoints`가 24개입니다.

토큰화된 커리의 경우 각 차원 목록에 3개의 항목이 있습니다:

- `db.query_tokenized.statement` – 토큰화된 쿼리문.
- `db.query_tokenized.db_id` — 성능 개선 도우미에서 자동으로 생성하는 합성 ID입니다. 이 예에서는 `pi-1064184600` 합성 ID를 반환합니다.

- db.query_tokenized.id – 성능 개선 도우미 내부의 쿼리에 대한 ID입니다.

  에서 AWS Management Console이 ID를 지원 ID라고 합니다. ID는 AWS Support가 데이터베이스 관련 문제를 해결하는 데 도움이 되도록 검사할 수 있는 데이터이기 때문에이 이름이 지정됩니다.는 데이터의 보안 및 프라이버시를 매우 중요하게 AWS 생각하며 거의 모든 데이터는 로 암호화되어 저장됩니다 AWS KMS key. 따라서 내부에 있는 누구도이 데이터를 볼 AWS 수 없습니다. 앞의 예에서 tokenized.statement와 tokenized.db_id 모두 암호화되어 저장됩니다. 데이터베이스에 문제가 있는 경우 AWS Support ID를 참조하여 Support에서 도움을 받을 수 있습니다.

쿼리 시 Group에서 GroupBy을 지정하면 편리할 수 있습니다. 그러나 반환되는 데이터에 대한 더 세분화된 제어를 위해서는 차원 목록을 지정하십시오. 예를 들어 db.query_tokenized.statement만 필요한 경우에는 query.json file에 Dimensions 속성을 추가할 수 있습니다.

```
[
    {
        "Metric": "db.load.avg",
        "GroupBy": {
            "Group": "db.query_tokenized",
            "Dimensions":["db.query_tokenized.statement"],
            "Limit": 10
        }
    }
]
```

쿼리로 필터링된 DB 로드 평균 검색

이 예에서 해당되는 API 쿼리는 상위 쿼리에 대한 DB 평균 로드 검색의 명령과 유사합니다. 그러나 query.json 파일의 컨텐츠는 다음과 같습니다.

```
[
 {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.wait_state", "Limit": 5  },
        "Filter": { "db.query_tokenized.id": "AKIAIOSFODNN7EXAMPLE" }
    }
]
```

이 응답에서는 query.json 파일에 지정된 토큰화된 쿼리 AKIAIOSFODNN7EXAMPLE의 기여도에 따라 모든 값이 필터링됩니다. 키는 필터링된 쿼리에 영향을 준 상위 5개 대기 상태이기 때문에 필터가 없는 쿼리와 다른 순서를 따를 수도 있습니다.

성능 개선 도우미를 위한 Amazon CloudWatch 지표

성능 개선 도우미는 Amazon CloudWatch에 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	Amazon DocumentDB의 활성 세션 수입니다. 일반적으로 사용자는 활성 세션의 평균 개수에 대한 데이터를 원합니다. 성능 개선 도우미에서 이 데이터는 `db.load.avg` 로 쿼리됩니다.
DBLoadCPU	대기 상태 유형이 CPU인 활성 세션 수입니다. 성능 개선 도우미에서 이 데이터는 `db.load.avg` 로 쿼리되며 대기 상태 유형인 CPU를 기준으로 필터링됩니다.
DBLoadNonCPU	대기 상태 유형이 CPU가 아닌 활성 세션 수입니다.

> **Note**
>
> 이러한 메트릭은 DB 인스턴스에 로드가 있는 경우에만 CloudWatch에 게시됩니다.

CloudWatch 콘솔, AWS CLI또는 CloudWatch API를 사용하여 이러한 지표를 검사할 수 있습니다.

예를 들어, get-metric-statistics 명령을 실행하여 DBLoad 지표에 대한 통계를 가져올 수 있습니다.

```
aws cloudwatch get-metric-statistics \
    --region ap-south-1 \
    --namespace AWS/DocDB \
    --metric-name DBLoad  \
    --period 360 \
```

```
    --statistics Average \
    --start-time 2022-03-14T8:00:00Z \
    --end-time 2022-03-14T9:00:00Z \
    --dimensions Name=DBInstanceIdentifier,Value=documentdbinstance
```

이 예에서는 다음과 비슷한 출력이 생성됩니다.

```
{
    "Datapoints": [
        {
            "Timestamp": "2022-03-14T08:42:00Z",
            "Average": 1.0,
            "Unit": "None"
        },
        {
            "Timestamp": "2022-03-14T08:24:00Z",
            "Average": 2.0,
            "Unit": "None"
        },
        {
            "Timestamp": "2022-03-14T08:54:00Z",
            "Average": 6.0,
            "Unit": "None"
        },
        {
            "Timestamp": "2022-03-14T08:36:00Z",
            "Average": 5.7,
            "Unit": "None"
        },
        {
            "Timestamp": "2022-03-14T08:06:00Z",
            "Average": 4.0,
            "Unit": "None"
        },
        {
            "Timestamp": "2022-03-14T08:00:00Z",
            "Average": 5.2,
            "Unit": "None"
        }
    ],
    "Label": "DBLoad"
}
```

CloudWatch 콘솔에서 DB_PERF_INSIGHTS 메트릭 산술 함수를 사용하여 Amazon DocumentDB 성능 개선 도우미 카운터 메트릭을 쿼리할 수 있습니다. DB_PERF_INSIGHTS 함수에는 분 단위로 DBLoad 메트릭도 포함됩니다. 이러한 지표에 대해 CloudWatch 경보를 설정할 수 있습니다. 경보를 생성하는 방법에 대한 자세한 내용은 AWS 데이터베이스에서 성능 개선 도우미 카운터 지표에 대한 경보 생성을 참조하세요.

CloudWatch에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서의 Amazon CloudWatch란 무엇입니까?를 참조하세요.

카운터 지표에 대한 성능 개선 도우미

카운터 메트릭은 성능 개선 도우미 대시보드의 운영 체제 메트릭입니다. 이 정보와 데이터베이스 로드를 연관 지으면 성능 문제를 식별하고 분석하는 데 도움이 됩니다.

성능 개선 도우미 운영 체제 카운터

Amazon DocumentDB 성능 개선 도우미와 함께 사용할 수 있는 운영 체제 카운터는 다음과 같습니다.

카운터	유형	측정치
활성화	메모리	os.memory.active
버퍼	메모리	os.memory.buffers
캐시됨	메모리	os.memory.cached
더티	메모리	os.memory.dirty
사용 가능	메모리	os.memory.free
비활성	메모리	os.memory.inactive
매핑됨	메모리	os.memory.mapped
pageTables	메모리	os.memory.pageTables
슬래브	메모리	os.memory.slab
총합	메모리	os.memory.total
writeback	메모리	os.memory.writeback

카운터	유형	측정치
유휴	cpuUtilization	os.cpuUtilization.idle
시스템	cpuUtilization	os.cpuUtilization.system
총합	cpuUtilization	os.cpuUtilization.total
사용자	cpuUtilization	os.cpuUtilization.user
대기	cpuUtilization	os.cpuUtilization.wait
1	loadAverageMinute	os.loadAverageMinute.one
15	loadAverageMinute	os.loadAverageMinute.fifteen
5	loadAverageMinute	os.loadAverageMinute.five
캐시됨	스왑	os.swap.cached
사용 가능	스왑	os.swap.free
인	스왑	os.swap.in
아웃	스왑	os.swap.out
총합	스왑	os.swap.total
rx	네트워크	os.network.rx
tx	네트워크	os.network.tx
numVCPUs	일반	os.general.numVCPUs

Amazon OpenSearch Service와 제로 ETL 통합

주제

- Amazon OpenSearch Service의 대상입니다.
- 제한 사항

Amazon OpenSearch Service의 대상입니다.

Amazon DocumentDB와의 OpenSearch Service 통합을 통해 전체 로드를 스트리밍하고 데이터 이벤트를 OpenSearch 도메인으로 변경할 수 있습니다. 수집 인프라는 OpenSearch 수집 파이프라인으로 호스팅되며 Amazon DocumentDB 컬렉션에서 데이터를 지속적으로 스트리밍할 수 있는 대규모의 짧은 지연 시간 메커니즘을 제공합니다.

전체 로드 중에 제로 ETL 통합은 먼저 수집 파이프라인을 사용하여 과거 전체 로드 데이터를 OpenSearch로 추출합니다. 전체 로드 데이터가 수집되면 OpenSearch 수집 파이프라인은 Amazon DocumentDB 변경 스트림에서 데이터를 읽기 시작하고 결국 Amazon DocumentDB와 OpenSearch 간의 실시간에 가까운 데이터 일관성을 유지하는 것에 가까워집니다. OpenSearch는 문서를 인덱스에 저장합니다. Amazon DocumentDB 컬렉션에서 수신되는 데이터는 하나의 인덱스로 전송하거나 다른 인덱스로 분할할 수 있습니다. 수집 파이프라인은 Amazon DocumentDB 컬렉션의 모든 생성, 업데이트 및 삭제 이벤트를 OpenSearch 문서의 해당 생성, 업데이트 및 삭제로 동기화하여 두 데이터 시스템을 동기화된 상태로 유지합니다. 수집 파이프라인은 한 가지 컬렉션에서 데이터를 읽고 한 가지 인덱스에 쓰거나 한 가지 컬렉션에서 데이터를 읽고 여러 인덱스로 조건부로 라우팅하도록 구성할 수 있습니다.

수집 파이프라인은 다음을 사용하여 Amazon DocumentDB에서 Amazon OpenSearch Service로 데이터를 스트리밍하도록 구성할 수 있습니다.

- 전체 로드만
- 전체 로드 없이 Amazon DocumentDB에서 스트림 변경 이벤트 스트리밍
- 전체 로드 후 Amazon DocumentDB에서 스트림 변경

수집 파이프라인을 설정하려면 다음 단계를 수행합니다.

1단계: Amazon OpenSearch Service 도메인 또는 OpenSearch 서버리스 컬렉션 생성

데이터를 읽을 수 있는 적절한 권한이 있는 Amazon OpenSearch Service 컬렉션이 필요합니다. 모음을 생성하려면 Amazon OpenSearch Service 개발자 가이드의 Amazon OpenSearch Service 시작하기 또는 Amazon OpenSearch Serverless 시작하기를 참조하세요. Amazon OpenSearch Service 개발자 안내서의 Amazon OpenSearch Ingestion을 참조하여 컬렉션 또는 도메인에 쓰기 데이터에 액세스할 수 있는 올바른 권한이 있는 AIM 역할을 생성합니다.

2단계: Amazon DocumentDB 클러스터에서 변경 스트림 활성화

Amazon DocumentDB 클러스터의 필수 컬렉션에서 변경 스트림이 활성화되어 있는지 확인합니다. 자세한 내용은 Amazon DocumentDB에서 변경 스트림 사용 섹션을 참조하세요.

3단계: Amazon S3 버킷 및 대상 도메인 또는 컬렉션에 쓸 수 있는 권한을 가진 파이프라인 역할 설정

Amazon DocumentDB 컬렉션을 생성하고 변경 스트림을 활성화한 후 파이프라인 구성에 사용할 파이프라인 역할을 설정하고 역할에 다음 권한을 추가합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "allowReadAndWriteToS3ForExport",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:AbortMultipartUpload",
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": [
                "arn:aws:s3:::my-bucket/export/*"
            ]
        }
    ]
}
```

OpenSearch 파이프라인이 OpenSearch 도메인에 데이터를 쓰려면 도메인에 sts_role_arn 파이프라인 역할이 도메인에 액세스할 수 있도록 허용하는 도메인 수준 액세스 정책이 있어야 합니다. 다음 샘플 도메인 액세스 정책은 이전 단계에서 생성한 `pipeline-role`이라는 파이프라인 역할을 사용하여 `ingestion-domain`이라는 도메인에 데이터를 쓸 수 있도록 허용합니다.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::{your-account-id}:role/{pipeline-role}"
      },
      "Action": ["es:DescribeDomain", "es:ESHttp*"],
      "Resource": "arn:aws:es:{region}:{your-account-id}:domain/{domain-name}/*"
    }
  ]
}
```

4단계: 파이프라인 역할에 필요한 권한을 추가하여 X-ENI 생성

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AttachNetworkInterface",
                "ec2:CreateNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:DetachNetworkInterface",
                "ec2:DescribeNetworkInterfaces"
            ],
            "Resource": [
                "arn:aws:ec2:*:420497401461:network-interface/*",
                "arn:aws:ec2:*:420497401461:subnet/*",
                "arn:aws:ec2:*:420497401461:security-group/*"
            ]
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeDhcpOptions",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ec2:Describe*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [ "ec2:CreateTags" ],
            "Resource": "arn:aws:ec2:*:*:network-interface/*",
            "Condition": {
               "StringEquals": { "aws:RequestTag/OSISManaged": "true" }
            }
        }
    ]
}
```

5단계: 파이프라인 생성

Amazon DocumentDB를 원본으로 지정하는 OpenSearch 수집 파이프라인을 구성합니다. 이 샘플 파이프라인 구성은 변경 스트림 가져오기 메커니즘을 사용하는 것으로 가정합니다. 자세한 내용은 Amazon OpenSearch Service 개발자 안내서의 Amazon DocumentDB에서 OpenSearch 수집 파이프라인 사용을 참조하세요.

제한 사항

Amazon DocumentDB OpenSearch 통합에는 다음 제한 사항이 적용됩니다.

- 파이프라인당 하나의 Amazon DocumentDB 컬렉션만 원본으로 지원됩니다.
- 교차 리전 데이터 수집은 지원되지 않습니다. Amazon DocumentDB 클러스터와 OpenSearch 도메인은 동일한 AWS 리전에 있어야 합니다.
- 교차 계정 데이터 수집은 지원되지 않습니다. Amazon DocumentDB 클러스터와 OpenSearch 수집 파이프라인은 동일한 AWS 계정에 있어야 합니다.
- Amazon DocumentDB 탄력적 클러스터는 지원되지 않습니다. Amazon DocumentDB 인스턴스 기반 클러스터만 지원됩니다.

- Amazon DocumentDB 클러스터에 AWS secrets. AWS secrets를 사용하여 인증이 활성화되어 있는지 확인합니다. 인증 메커니즘은 유일하게 지원됩니다.
- 기존 파이프라인 구성은 다른 데이터베이스 및/또는 다른 컬렉션에서 데이터를 수집하도록 업데이트할 수 없습니다. 파이프라인의 데이터베이스 및/또는 컬렉션 이름을 업데이트하려면 새 파이프라인을 생성해야 합니다.

Amazon DocumentDB로 개발

이 섹션에서는 Amazon DocumentDB를 사용한 개발에 대해 설명합니다.

주제

- Amazon DocumentDB에 연결
- Amazon DocumentDB를 사용한 프로그램

Amazon DocumentDB에 연결

Amazon DocumentDB는 MongoDB 드라이버를 사용한 프로그래밍 방식 연결, SSH 터널링 또는 AWS VPN을 통해 VPC 외부에서 연결, Studio 3T 및 DataGrip과 같은 인기 있는 도구를 비롯한 여러 연결 옵션을 제공합니다. 이 서비스는 고가용성을 위해 복제본 세트 연결을 지원합니다. 분석 및 보고를 위해 JDBC 또는 ODBC 드라이버를 사용하여 연결할 수 있습니다. 동일한 VPC의 EC2 인스턴스는 DocumentDB 클러스터에 직접 연결할 수 있습니다. 모든 연결에는 TLS와 적절한 자격 AWS 증명/인증이 필요합니다.

주제

- Amazon DocumentDB에 프로그래밍 방식으로 연결
- Amazon VPC 외부에서 Amazon DocumentDB 클러스터에 연결
- Amazon DocumentDB에 복제본 세트로 연결
- Studio 3T에서 Amazon DocumentDB 클러스터에 연결
- DataGrip을 사용하여 Amazon DocumentDB에 연결
- Amazon EC2를 사용하여 연결
- Amazon DocumentDB JDBC 드라이버를 사용하여 연결합니다.
- Amazon DocumentDB ODBC 드라이버를 사용하여 연결

Amazon DocumentDB에 프로그래밍 방식으로 연결

이 섹션에는 다양한 언어를 사용하여 Amazon DocumentDB(MongoDB 호환)에 연결하는 방법을 보여주는 코드 예제가 포함되어 있습니다. 이들 예제는 전송 계층 보안(TLS)이 활성화 또는 비활성화된 클러스터에 연결하는지에 따라 두 가지로 구분됩니다. 기본적으로 Amazon DocumentDB 클러스터에서는 TLS가 활성화됩니다. 하지만 원할 경우 TLS를 비활성화할 수 있습니다. 자세한 내용은 전송 중 데이터 암호화 단원을 참조하십시오.

클러스터가 상주하는 VPC 외부에서 Amazon DocumentDB에 연결하려는 경우 Amazon VPC 외부에서 Amazon DocumentDB 클러스터에 연결 섹션을 참조하세요.

클러스터에 연결하기 전에 클러스터에서 TLS가 활성화되어 있는지 여부를 알아야 합니다. 다음 섹션에서는 AWS Management Console 또는 AWS CLI를 사용하여 클러스터의 `tls` 파라미터 값을 확인하는 방법을 설명합니다. 그런 다음 적절한 코드 예제를 찾아 적용할 수 있습니다.

주제

- tls 파라미터 값 확인
- TLS가 활성화된 상태에서 연결
- TLS가 비활성화된 상태에서 연결

`tls` 파라미터 값 확인

클러스터에 TLS가 활성화되어 있는지 확인하는 것은 AWS Management Console 또는를 사용하여 수행할 수 있는 2단계 프로세스입니다 AWS CLI.

1. 어느 파라미터 그룹이 클러스터를 관리하는지 결정합니다.

   Using the AWS Management Console

   1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb:// https://https://https://httpsAmazon DocumentDB://https://://https://://https://://https://https:// https://https://https://https://
   2. 좌측 탐색 창에서 클러스터를 선택합니다.
   3. 클러스터 목록에서 클러스터 이름을 선택합니다.
   4. 결과 페이지에는 선택한 클러스터의 세부 정보가 표시됩니다. [구성(Configuration)] 탭을 선택합니다. 구성 및 상태 섹션에서 클러스터 파라미터 그룹 아래에서 파라미터 그룹의 이름을 찾습니다.

   Using the AWS CLI

   다음 AWS CLI 코드는 클러스터를 관리하는 파라미터를 결정합니다. *`sample-cluster`*를 클러스터의 이름으로 바꾸어야 합니다.

   ```
   aws docdb describe-db-clusters \
       --db-cluster-identifier sample-cluster \
   ```

```
    --query 'DBClusters[*].[DBClusterIdentifier,DBClusterParameterGroup]'
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
[
       [
           "sample-cluster",
           "sample-parameter-group"
       ]
]
```

2. 클러스터의 파라미터 그룹에서 **tls** 파라미터의 값을 확인합니다.

Using the AWS Management Console

1. 탐색 창에서 파라미터 그룹을 선택합니다.
2. 클러스터 파라미터 그룹 창의 1d단계에서 클러스터 파라미터 그룹 이름을 선택합니다.
3. 결과 페이지에는 클러스터 파라미터 그룹의 파라미터가 표시됩니다. 여기서 `tls` 파라미터의 값을 볼 수 있습니다. 이 파라미터 수정에 대한 자세한 내용은 Amazon DocumentDB 클러스터 파라미터 그룹 수정 섹션을 참조하세요.

Using the AWS CLI

`describe-db-cluster-parameters` AWS CLI 명령을 사용하여 클러스터 파라미터 그룹의 파라미터 세부 정보를 볼 수 있습니다.

- **`--describe-db-cluster-parameters`** - 파라미터 그룹 내의 모든 파라미터와 해당 값을 나열합니다.
  - **`--db-cluster-parameter-group name`** - 필수입니다. 클러스터 파라미터 그룹의 이름입니다.

다음 예제에서는 각 *### ## ## ##*자를 클러스터 정보로 바꿉니다.

```
aws docdb describe-db-cluster-parameters \
    --db-cluster-parameter-group-name sample-parameter-group
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
{
        "Parameters": [
            {
                "ParameterName": "profiler_threshold_ms",
                "ParameterValue": "100",
                "Description": "Operations longer than profiler_threshold_ms
 will be logged",
                "Source": "system",
                "ApplyType": "dynamic",
                "DataType": "integer",
                "AllowedValues": "50-2147483646",
                "IsModifiable": true,
                "ApplyMethod": "pending-reboot"
            },
            {
                "ParameterName": "tls",
                "ParameterValue": "disabled",
                "Description": "Config to enable/disable TLS",
                "Source": "user",
                "ApplyType": "static",
                "DataType": "string",
                "AllowedValues": "disabled,enabled,fips-140-3",
                "IsModifiable": true,
                "ApplyMethod": "pending-reboot"
            }
        ]
}
```

> **Note**
>
> Amazon DocumentDB는 ca-central-1, us-west-2, us-east-1, us-east-2, us-gov-east-1, us-gov-west-1 리전에서 Amazon DocumentDB 5.0 (엔진 버전 3.0.3727) 클러스터부터 FIPS 140-3 엔드포인트를 지원합니다.

`tls` 파라미터의 값을 확인했으면 이어서 다음 섹션에 수록된 코드 예제 중 하나를 사용하여 클러스터에 연결합니다.

- TLS가 활성화된 상태에서 연결
- TLS가 비활성화된 상태에서 연결

TLS가 활성화된 상태에서 연결

TLS 지원 Amazon DocumentDB 클러스터에 프로그램 방식으로 연결하는 코드 예제를 보려면 사용하려는 언어에 해당하는 탭을 선택합니다.

전송 중인 데이터를 암호화하려면 다음 작업을 사용하여 이름이 `global-bundle.pem`인 Amazon DocumentDB에 대한 퍼블릭 키를 다운로드합니다.

```
wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem
```

애플리케이션이 Microsoft Windows에서 실행되어 PKCS7 파일이 필요한 경우에는 PKCS7 인증서 번들을 다운로드할 수 있습니다. 이 번들에는 https://truststore.pki.rds.amazonaws.com/global/global-bundle.p7b의 중간 인증서와 루트 인증서가 모두 포함되어 있습니다.

Python

다음 코드는 TLS가 활성화된 상태에서 Python을 사용하여 Amazon DocumentDB에 연결하는 방법을 보여줍니다.

다음 예제에서는 각 *### ## ## ##*자를 클러스터 정보로 바꿉니다.

```
import pymongo
import sys

##Create a MongoDB client, open a connection to Amazon DocumentDB as a replica set
 and specify the read preference as secondary preferred
client = pymongo.MongoClient('mongodb://sample-user:password@sample-
cluster.node.us-east-1.docdb.amazonaws.com:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

##Specify the database to be used
db = client.sample_database

##Specify the collection to be used
col = db.sample_collection

##Insert a single document
col.insert_one({'hello':'Amazon DocumentDB'})

##Find the document that was previously written
x = col.find_one({'hello':'Amazon DocumentDB'})
```

```
##Print the result to the screen
print(x)

##Close the connection
client.close()
```

Node.js

다음 코드는 TLS가 활성화된 상태에서 Node.js를 사용하여 Amazon DocumentDB에 연결하는 방법을 보여줍니다.

> **Important**
>
> 현재 Amazon DocumentDB에 대한 IAM 자격 증명 인증에서 지원되지 않는 버전 6.13.1 이전 Node.js 드라이버에는 알려진 제한 사항이 있습니다. Node.js 드라이버를 사용하는 Node.js 드라이버 및 도구(예: mongosh)는 Node.js 드라이버 버전 6.13.1 이상을 사용하도록 업그레이드해야 합니다.

다음 예제에서는 각 *### ## ## ##*자를 클러스터 정보로 바꿉니다.

```
var MongoClient = require('mongodb').MongoClient

//Create a MongoDB client, open a connection to DocDB; as a replica set,
//  and specify the read preference as secondary preferred

var client = MongoClient.connect(
'mongodb://sample-user:password@sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017/sample-database?
tls=true&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false',
{
  tlsCAFile: `global-bundle.pem` //Specify the DocDB; cert
},
function(err, client) {
    if(err)
        throw err;

    //Specify the database to be used
    db = client.db('sample-database');

    //Specify the collection to be used
    col = db.collection('sample-collection');
```

```
    //Insert a single document
    col.insertOne({'hello':'Amazon DocumentDB'}, function(err, result){
      //Find the document that was previously written
      col.findOne({'hello':'Amazon DocumentDB'}, function(err, result){
        //Print the result to the screen
        console.log(result);

        //Close the connection
        client.close()
      });
   });
});
```

PHP

다음 코드는 TLS가 활성화된 상태에서 PHP를 사용하여 Amazon DocumentDB에 연결하는 방법을 보여줍니다.

다음 예제에서는 각 *### ## ## ##*자를 클러스터 정보로 바꿉니다.

```
<?php
//Include Composer's autoloader
require 'vendor/autoload.php';

$TLS_DIR = "/home/ubuntu/global-bundle.pem";

//Create a MongoDB client and open connection to Amazon DocumentDB
$client = new MongoDB\Client("mongodb://sample-user:password@sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017/?retryWrites=false", ["tls" => "true", "tlsCAFile"
 => $TLS_DIR ]);

//Specify the database and collection to be used
$col = $client->sampledatabase->samplecollection;

//Insert a single document
$result = $col->insertOne( [ 'hello' => 'Amazon DocumentDB'] );

//Find the document that was previously written
$result = $col->findOne(array('hello' => 'Amazon DocumentDB'));

//Print the result to the screen
print_r($result);
```

```
?>
```

Go

다음 코드는 TLS가 활성화된 상태에서 Go를 사용하여 Amazon DocumentDB에 연결하는 방법을 보여줍니다.

> **Note**
>
> 버전 1.2.1부터 MongoDB Go Driver는 `sslcertificateauthorityfile`에서 발견된 첫 번째 CA 서버 인증서만을 사용합니다. 아래 예제 코드는 `sslcertificateauthorityfile`에서 찾은 모든 서버 인증서를 클라이언트 생성 중에 사용되는 사용자 지정 TLS 구성에 수동으로 추가하여 이 제한을 해결합니다.

다음 예제에서는 각 `### ## ## ##`자를 클러스터 정보로 바꿉니다.

```
package main

import (
 "context"
 "fmt"
 "log"
 "time"

 "go.mongodb.org/mongo-driver/bson"
 "go.mongodb.org/mongo-driver/mongo"
 "go.mongodb.org/mongo-driver/mongo/options"

 "io/ioutil"
 "crypto/tls"
 "crypto/x509"
 "errors"
)

const (
 // Path to the AWS CA file
 caFilePath = "global-bundle.pem"

 // Timeout operations after N seconds
 connectTimeout  = 5
 queryTimeout    = 30
```

```
 username        = "sample-user"
 password        = "password"
 clusterEndpoint = "sample-cluster.node.us-east-1.docdb.amazonaws.com:27017"

 // Which instances to read from
 readPreference = "secondaryPreferred"

 connectionStringTemplate = "mongodb://%s:%s@%s/sample-database?
tls=true&replicaSet=rs0&readpreference=%s"
)

func main() {

 connectionURI := fmt.Sprintf(connectionStringTemplate, username, password,
 clusterEndpoint, readPreference)

 tlsConfig, err := getCustomTLSConfig(caFilePath)
 if err != nil {
  log.Fatalf("Failed getting TLS configuration: %v", err)
 }

 client, err :=
 mongo.NewClient(options.Client().ApplyURI(connectionURI).SetTLSConfig(tlsConfig))
 if err != nil {
  log.Fatalf("Failed to create client: %v", err)
 }

 ctx, cancel := context.WithTimeout(context.Background(),
 connectTimeout*time.Second)
 defer cancel()

 err = client.Connect(ctx)
 if err != nil {
  log.Fatalf("Failed to connect to cluster: %v", err)
 }

 // Force a connection to verify our connection string
 err = client.Ping(ctx, nil)
 if err != nil {
  log.Fatalf("Failed to ping cluster: %v", err)
 }

 fmt.Println("Connected to DocumentDB!")
```

```
 collection := client.Database("sample-database").Collection("sample-collection")

 ctx, cancel = context.WithTimeout(context.Background(), queryTimeout*time.Second)
 defer cancel()

 res, err := collection.InsertOne(ctx, bson.M{"name": "pi", "value": 3.14159})
 if err != nil {
  log.Fatalf("Failed to insert document: %v", err)
 }

 id := res.InsertedID
 log.Printf("Inserted document ID: %s", id)

 ctx, cancel = context.WithTimeout(context.Background(), queryTimeout*time.Second)
 defer cancel()

 cur, err := collection.Find(ctx, bson.D{})

 if err != nil {
  log.Fatalf("Failed to run find query: %v", err)
 }
 defer cur.Close(ctx)

 for cur.Next(ctx) {
  var result bson.M
  err := cur.Decode(&result)
  log.Printf("Returned: %v", result)

  if err != nil {
   log.Fatal(err)
  }
 }

 if err := cur.Err(); err != nil {
  log.Fatal(err)
 }

}

func getCustomTLSConfig(caFile string) (*tls.Config, error) {
 tlsConfig := new(tls.Config)
 certs, err := ioutil.ReadFile(caFile)

 if err != nil {
```

```
 return tlsConfig, err
}

tlsConfig.RootCAs = x509.NewCertPool()
ok := tlsConfig.RootCAs.AppendCertsFromPEM(certs)

if !ok {
 return tlsConfig, errors.New("Failed parsing pem file")
}

return tlsConfig, nil
```

Java

Java 애플리케이션에서 TLS 지원 Amazon DocumentDB 클러스터에 연결할 때 프로그램은 AWS 제공 인증 기관(CA) 파일을 사용하여 연결을 검증해야 합니다. Amazon RDS CA 인증서를 사용하려면 다음을 수행합니다.

1. https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem 에서 Amazon RDS CA 파일을 다운로드하세요.
2. 다음 명령을 수행하여 파일에 포함된 CA 인증서로 트러스트 스토어를 생성합니다. *truststore-password*를 다른 것으로 변경해야 합니다. 이전 CA 인증서(rds-ca-2015-root.pem)와 새 CA 인증서(rds-ca-2019-root.pem)가 모두 포함된 트러스트 저장소에 액세스하는 경우 인증서 번들을 트러스트 저장소로 가져올 수 있습니다.

   다음은 Linux 운영 체제에서 트러스트 스토어로 인증서 번들을 가져오는 샘플 쉘 스크립트입니다. 다음 예에서는 자신의 정보로 각각의 *### ## ## ###*를 바꿉니다. 특히 예제 디렉터리 "*mydir*"가 스크립트에 있는 경우 이 작업을 위해 생성한 디렉터리로 교체합니다.

   ```
   mydir=/tmp/certs
   truststore=${mydir}/rds-truststore.jks
   storepassword=truststore-password

   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 ".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
```

다음은 macOS에서 트러스트 스토어로 인증서 번들을 가져오는 샘플 쉘 스크립트입니다.

```
mydir=/tmp/certs
truststore=${mydir}/rds-truststore.jks
storepassword=truststore-password

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
```

3. Amazon DocumentDB 클러스터에 연결하기 전에 애플리케이션에서 다음 시스템 속성을 설정하여 프로그램에서 keystore를 사용합니다.

```
javax.net.ssl.trustStore: truststore
javax.net.ssl.trustStorePassword: truststore-password;
```

4. 다음 코드는 TLS가 활성화된 상태에서 Java를 사용하여 Amazon DocumentDB에 연결하는 방법을 보여줍니다.

   다음 예제에서는 각 *### ## ## ##*자를 클러스터 정보로 바꿉니다.

```
package com.example.documentdb;

import com.mongodb.client.*;
import org.bson.Document;

public final class Test {
    private Test() {
    }
    public static void main(String[] args) {

        String template = "mongodb://%s:%s@%s/sample-database?
ssl=true&replicaSet=rs0&readpreference=%s";
        String username = "sample-user";
        String password = "password";
        String clusterEndpoint = "sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017";
        String readPreference = "secondaryPreferred";
        String connectionString = String.format(template, username, password,
 clusterEndpoint, readPreference);

        String truststore = "truststore";
        String truststorePassword = "truststore-password";
```

```
        System.setProperty("javax.net.ssl.trustStore", truststore);
        System.setProperty("javax.net.ssl.trustStorePassword",
 truststorePassword);

        MongoClient mongoClient = MongoClients.create(connectionString);

        MongoDatabase testDB = mongoClient.getDatabase("sample-database");
        MongoCollection<Document> numbersCollection =
 testDB.getCollection("sample-collection");

        Document doc = new Document("name", "pi").append("value", 3.14159);
        numbersCollection.insertOne(doc);

        MongoCursor<Document> cursor = numbersCollection.find().iterator();
        try {
            while (cursor.hasNext()) {
                System.out.println(cursor.next().toJson());
            }
        } finally {
            cursor.close();
        }

    }
}
```

C# / .NET

다음 코드는 TLS가 활성화된 상태에서 C# / .NET를 사용하여 Amazon DocumentDB에 연결하는 방법을 보여줍니다.

다음 예제에서는 각 `### ## ## ##`자를 클러스터 정보로 바꿉니다.

```
using System;
using System.Text;
using System.Linq;
using System.Collections.Generic;
using System.Security.Cryptography;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
using MongoDB.Driver;
using MongoDB.Bson;

namespace DocDB
```

```
{
    class Program
    {
        static void Main(string[] args)
        {
            string template = "mongodb://{0}:{1}@{2}/sampledatabase?
tls=true&replicaSet=rs0&readpreference={3}";
            string username = "sample-user";
            string password = "password";
            string readPreference = "secondaryPreferred";
            string clusterEndpoint="sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017";
            string connectionString = String.Format(template, username, password,
 clusterEndpoint, readPreference);

            string pathToCAFile = "<PATH/global-bundle.p7b_file>";

            // ADD CA certificate to local trust store
            // DO this once - Maybe when your service starts
            X509Store localTrustStore = new X509Store(StoreName.Root);
            X509Certificate2Collection certificateCollection = new
 X509Certificate2Collection();
            certificateCollection.Import(pathToCAFile);
            try
            {
                localTrustStore.Open(OpenFlags.ReadWrite);
                localTrustStore.AddRange(certificateCollection);
            }
            catch (Exception ex)
            {
                Console.WriteLine("Root certificate import failed: " + ex.Message);
                throw;
            }
            finally
            {
                localTrustStore.Close();
            }

            var settings = MongoClientSettings.FromUrl(new
 MongoUrl(connectionString));
            var client = new MongoClient(settings);

            var database = client.GetDatabase("sampledatabase");
```

```
            var collection =
 database.GetCollection<BsonDocument>("samplecollection");
            var docToInsert = new BsonDocument { { "pi", 3.14159 } };
            collection.InsertOne(docToInsert);
        }
    }
}
```

MongoDB Shell

다음 코드는 TLS가 활성화된 경우 최신 버전, mongosh 또는 이전 mongo 셸 버전을 사용하여 Amazon DocumentDB에 연결하고 쿼리하는 방법을 보여줍니다.

mongosh를 사용하여 Amazon DocumentDB에 연결

> **Important**
>
> 현재 Amazon DocumentDB에 대한 IAM 자격 증명 인증에서 지원되지 않는 버전 6.13.1 이전 Node.js 드라이버에는 알려진 제한 사항이 있습니다. Node.js 드라이버를 사용하는 Node.js 드라이버 및 도구(예: mongosh)는 Node.js 드라이버 버전 6.13.1 이상을 사용하도록 업그레이드해야 합니다.

다음 예제에서는 각 *### ## ## ##*자를 클러스터 정보로 바꿉니다.

```
mongosh --tls --host cluster-end-point:27017 --tlsCAFile  global-bundle.pem --
username sample-user --password password --retryWrites false
```

이전 mongo 셸 버전으로 Amazon DocumentDB에 연결

IAM을 사용하는 경우 이전 버전의 mongo 셸을 사용해야 합니다. 다음 명령 옵션 중 하나를 입력합니다.

```
mongo --ssl --host cluster-end-point:27017 --sslCAFile global-bundle.pem --
username sample-user --password password
```

4.2 이상의 버전을 사용하는 경우 다음 코드를 사용하여 연결하세요. Amazon DocumentDB에서는 재시도 가능한 쓰기가 지원되지 않습니다. mongo 쉘(mongosh 제외)을 사용하는 경우 코드 문자열에 `retryWrites=false` 명령을 포함시키지 마십시오. 기본적으로 재시도 가능한 쓰기는 비활성

화되어 있습니다. retryWrites=false를 포함하면 일반 읽기 명령에서 오류가 발생할 수 있습니다.

```
mongo --tls --host cluster-end-point:27017 --tlsCAFile global-bundle.pem --
username sample-user --password password
```

연결 테스트

1. 단일 문서를 삽입합니다.

   ```
   db.myTestCollection.insertOne({'hello':'Amazon DocumentDB'})
   ```

2. 이전에 삽입된 문서를 찾습니다.

   ```
   db.myTestCollection.find({'hello':'Amazon DocumentDB'})
   ```

R

다음 코드는 TLS가 활성화된 상태에서 mongolite(https://jeroen.github.io/mongolite/)를 사용하여 R을 Amazon DocumentDB에 연결하고 쿼리하는 방법을 보여줍니다.

다음 예제에서는 각 ### ## ## ##자를 클러스터 정보로 바꿉니다.

```
#Include the mongolite library.
library(mongolite)

mongourl <- paste("mongodb://sample-user:password@sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017/test2?ssl=true&",
          "readPreference=secondaryPreferred&replicaSet=rs0", sep="")

#Create a MongoDB client, open a connection to Amazon DocumentDB as a replica
#   set and specify the read preference as secondary preferred
client <-  mongo(url = mongourl, options = ssl_options(weak_cert_validation = F, ca
 ="<PATH/global-bundle.pem>"))

#Insert a single document
str <- c('{"hello" : "Amazon DocumentDB"}')
client$insert(str)

#Find the document that was previously written
client$find()
```

Ruby

다음 코드는 TLS가 활성화된 상태에서 Java를 사용하여 Ruby를 Amazon DocumentDB에 연결하는 방법을 보여줍니다.

다음 예제에서는 각 *### ## ## ##*자를 클러스터 정보로 바꿉니다.

```
require 'mongo'
require 'neatjson'
require 'json'
client_host = 'mongodb://sample-cluster.node.us-east-1.docdb.amazonaws.com:27017'
client_options = {
   database: 'test',
   replica_set: 'rs0',
   read: {:secondary_preferred => 1},
   user: 'sample-user',
   password: 'password',
   ssl: true,
   ssl_verify: true,
   ssl_ca_cert: 'PATH/global-bundle.pem',
   retry_writes: false
}


begin
   ##Create a MongoDB client, open a connection to Amazon DocumentDB as a
   ##   replica set and specify the read preference as secondary preferred
   client = Mongo::Client.new(client_host, client_options)

   ##Insert a single document
   x = client[:test].insert_one({"hello":"Amazon DocumentDB"})

   ##Find the document that was previously written
   result = client[:test].find()

   #Print the document
   result.each do |document|
      puts JSON.neat_generate(document)
   end
end

#Close the connection
client.close
```

TLS가 비활성화된 상태에서 연결

TLS가 비활성화된 Amazon DocumentDB 클러스터에 프로그래밍 방식으로 연결하기 위한 코드 예제를 보려면 사용할 언어에 해당하는 탭을 선택합니다.

Python

다음 코드는 TLS가 비활성화된 상태에서 Python을 사용하여 Amazon DocumentDB에 연결하는 방법을 보여줍니다.

다음 예제에서는 각 *### ## ## ##*자를 클러스터 정보로 바꿉니다.

```
## Create a MongoDB client, open a connection to Amazon DocumentDB as a replica set
 and specify the read preference as secondary preferred

import pymongo
import sys

client = pymongo.MongoClient('mongodb://sample-user:password@sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017/?
replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

##Specify the database to be used
db = client.sample_database

##Specify the collection to be used
col = db.sample_collection

##Insert a single document
col.insert_one({'hello':'Amazon DocumentDB'})

##Find the document that was previously written
x = col.find_one({'hello':'Amazon DocumentDB'})

##Print the result to the screen
print(x)

##Close the connection
client.close()
```

Node.js

다음 코드는 TLS가 비활성화된 상태에서 Node.js를 사용하여 Amazon DocumentDB에 연결하는 방법을 보여줍니다.

> **Important**
>
> 현재 Amazon DocumentDB에 대한 IAM 자격 증명 인증에서 지원되지 않는 버전 6.13.1 이전 Node.js 드라이버에는 알려진 제한 사항이 있습니다. Node.js 드라이버를 사용하는 Node.js 드라이버 및 도구(예: mongosh)는 Node.js 드라이버 버전 6.13.1 이상을 사용하도록 업그레이드해야 합니다.

다음 예제에서는 각 *### ## ## ##*자를 클러스터 정보로 바꿉니다.

```
var MongoClient = require('mongodb').MongoClient;

//Create a MongoDB client, open a connection to Amazon DocumentDB as a replica set,
//  and specify the read preference as secondary preferred
var client = MongoClient.connect(
'mongodb://sample-user:password@sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017/sample-database?
replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false',
{
  useNewUrlParser: true
},

function(err, client) {
    if(err)
        throw err;
    //Specify the database to be used
    db = client.db('sample-database');

    //Specify the collection to be used
    col = db.collection('sample-collection');

    //Insert a single document
    col.insertOne({'hello':'Amazon DocumentDB'}, function(err, result){
      //Find the document that was previously written
      col.findOne({'hello':'Amazon DocumentDB'}, function(err, result){
        //Print the result to the screen
        console.log(result);
```

```
        //Close the connection
        client.close()
      });
   });
});
```

PHP

다음 코드는 TLS가 비활성화된 상태에서 PHP를 사용하여 Amazon DocumentDB에 연결하는 방법을 보여줍니다.

다음 예제에서는 각 *### ## ## ##*자를 클러스터 정보로 바꿉니다.

```
<?php
//Include Composer's autoloader
require 'vendor/autoload.php';

//Create a MongoDB client and open connection to Amazon DocumentDB
$client = new MongoDB\Client("mongodb://sample-user:password@sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017/?retryWrites=false");

//Specify the database and collection to be used
$col = $client->sampledatabase->samplecollection;

//Insert a single document
$result = $col->insertOne( [ 'hello' => 'Amazon DocumentDB'] );

//Find the document that was previously written
$result = $col->findOne(array('hello' => 'Amazon DocumentDB'));

//Print the result to the screen
print_r($result);
?>
```

Go

다음 코드는 TLS가 비활성화된 상태에서 Go를 사용하여 Amazon DocumentDB에 연결하는 방법을 보여줍니다.

다음 예제에서는 각 *### ## ## ##*자를 클러스터 정보로 바꿉니다.

```
package main
```

```
import (
 "context"
 "fmt"
 "log"
 "time"

 "go.mongodb.org/mongo-driver/bson"
 "go.mongodb.org/mongo-driver/mongo"
 "go.mongodb.org/mongo-driver/mongo/options"
)

const (
 // Timeout operations after N seconds
 connectTimeout  = 5
 queryTimeout    = 30
 username        = "sample-user"
 password        = "password"
 clusterEndpoint = "sample-cluster.node.us-east-1.docdb.amazonaws.com:27017"

 // Which instances to read from
 readPreference           = "secondaryPreferred"
 connectionStringTemplate = "mongodb://%s:%s@%s/sample-database?
replicaSet=rs0&readpreference=%s"
)

func main() {

 connectionURI := fmt.Sprintf(connectionStringTemplate, username, password,
 clusterEndpoint, readPreference)

 client, err := mongo.NewClient(options.Client().ApplyURI(connectionURI))
 if err != nil {
  log.Fatalf("Failed to create client: %v", err)
 }

 ctx, cancel := context.WithTimeout(context.Background(),
 connectTimeout*time.Second)
 defer cancel()

 err = client.Connect(ctx)
 if err != nil {
  log.Fatalf("Failed to connect to cluster: %v", err)
 }
```

```
// Force a connection to verify our connection string
err = client.Ping(ctx, nil)
if err != nil {
 log.Fatalf("Failed to ping cluster: %v", err)
}

fmt.Println("Connected to DocumentDB!")

collection := client.Database("sample-database").Collection("sample-collection")

ctx, cancel = context.WithTimeout(context.Background(), queryTimeout*time.Second)
defer cancel()

res, err := collection.InsertOne(ctx, bson.M{"name": "pi", "value": 3.14159})
if err != nil {
 log.Fatalf("Failed to insert document: %v", err)
}

id := res.InsertedID
log.Printf("Inserted document ID: %s", id)

ctx, cancel = context.WithTimeout(context.Background(), queryTimeout*time.Second)
defer cancel()

cur, err := collection.Find(ctx, bson.D{})

if err != nil {
 log.Fatalf("Failed to run find query: %v", err)
}
defer cur.Close(ctx)

for cur.Next(ctx) {
 var result bson.M
 err := cur.Decode(&result)
 log.Printf("Returned: %v", result)

 if err != nil {
  log.Fatal(err)
 }
}

if err := cur.Err(); err != nil {
 log.Fatal(err)
```

```
 }

}
```

Java

다음 코드는 TLS가 비활성화된 상태에서 Java를 사용하여 Amazon DocumentDB에 연결하는 방법을 보여줍니다.

다음 예제에서는 각 *### ## ## ##*자를 클러스터 정보로 바꿉니다.

```
package com.example.documentdb;

import com.mongodb.MongoClient;
import com.mongodb.MongoClientURI;
import com.mongodb.ServerAddress;
import com.mongodb.MongoException;
import com.mongodb.client.MongoCursor;
import com.mongodb.client.MongoDatabase;
import com.mongodb.client.MongoCollection;
import org.bson.Document;


public final class Main {
    private Main() {
    }
    public static void main(String[] args) {

        String template = "mongodb://%s:%s@%s/sample-database?
replicaSet=rs0&readpreference=%s";
        String username = "sample-user";
        String password = "password";
        String clusterEndpoint = "sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017";
        String readPreference = "secondaryPreferred";
        String connectionString = String.format(template, username, password,
 clusterEndpoint, readPreference);

        MongoClientURI clientURI = new MongoClientURI(connectionString);
        MongoClient mongoClient = new MongoClient(clientURI);

        MongoDatabase testDB = mongoClient.getDatabase("sample-database");
        MongoCollection<Document> numbersCollection = testDB.getCollection("sample-
collection");
```

```
        Document doc = new Document("name", "pi").append("value", 3.14159);
        numbersCollection.insertOne(doc);

        MongoCursor<Document> cursor = numbersCollection.find().iterator();
        try {
            while (cursor.hasNext()) {
                System.out.println(cursor.next().toJson());
            }
        } finally {
            cursor.close();
        }

    }
}
```

C# / .NET

다음 코드는 TLS가 비활성화된 상태에서 C# / .NET를 사용하여 Amazon DocumentDB에 연결하는 방법을 보여줍니다.

다음 예제에서는 각 *### ## ## ##*자를 클러스터 정보로 바꿉니다.

```
using System;
using System.Text;
using System.Linq;
using System.Collections.Generic;
using System.Security.Cryptography;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
using MongoDB.Driver;
using MongoDB.Bson;

namespace CSharpSample
{
   class Program
    {
       static void Main(string[] args)
        {
           string template = "mongodb://{0}:{1}@{2}/sampledatabase?
replicaSet=rs0&readpreference={3}";
           string username = "sample-user";
           string password = "password";
```

```
            string clusterEndpoint = "sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017";
            string readPreference = "secondaryPreferred";
            string connectionString = String.Format(template, username, password,
 clusterEndpoint, readPreference);

            var settings = MongoClientSettings.FromUrl(new
 MongoUrl(connectionString));
            var client = new MongoClient(settings);

            var database = client.GetDatabase("sampledatabase");
            var collection =
 database.GetCollection<BsonDocument>("samplecollection");
            var docToInsert = new BsonDocument { { "pi", 3.14159 } };
             collection.InsertOne(docToInsert);
        }
    }
}
```

MongoDB Shell

다음 코드는 TLS가 비활성화된 경우 최신 버전, mongosh 또는 이전 mongo 셸 버전을 사용하여 Amazon DocumentDB에 연결하고 쿼리하는 방법을 보여줍니다.

mongosh를 사용하여 Amazon DocumentDB에 연결

> **Important**
>
> 현재 Amazon DocumentDB에 대한 IAM 자격 증명 인증에서 지원되지 않는 버전 6.13.1 이전 Node.js 드라이버에는 알려진 제한 사항이 있습니다. Node.js 드라이버를 사용하는 Node.js 드라이버 및 도구(예: mongosh)는 Node.js 드라이버 버전 6.13.1 이상을 사용하도록 업그레이드해야 합니다.

다음 예제에서는 각 *### ## ## ##*자를 클러스터 정보로 바꿉니다.

```
mongosh --host cluster-end-point:27017 --username sample-user --password password --
retryWrites false
```

이전 mongo 셸 버전으로 Amazon DocumentDB에 연결

IAM을 사용하는 경우 이전 버전의 mongo 셸을 사용해야 합니다. 다음 명령 옵션 중 하나를 입력합니다.

```
mongo --host cluster-end-point:27017 --username sample-user --password password
```

4.2 이상의 버전을 사용하는 경우 다음 코드를 사용하여 연결하세요. Amazon DocumentDB에서는 재시도 가능한 쓰기가 지원되지 않습니다. mongo 쉘(mongosh 제외)을 사용하는 경우 코드 문자열에 `retryWrites=false` 명령을 포함시키지 마십시오. 기본적으로 재시도 가능한 쓰기는 비활성화되어 있습니다. `retryWrites=false`를 포함하면 일반 읽기 명령에서 오류가 발생할 수 있습니다.

```
mongo --host cluster-end-point:27017 --username sample-user --password password
```

연결 테스트

1. 단일 문서를 삽입합니다.

   ```
   db.myTestCollection.insertOne({'hello':'Amazon DocumentDB'})
   ```

2. 이전에 삽입된 문서를 찾습니다.

   ```
   db.myTestCollection.find({'hello':'Amazon DocumentDB'})
   ```

R

다음 코드는 TLS가 비활성화된 상태에서 mongolite(https://jeroen.github.io/mongolite/)를 사용하여 R을 Amazon DocumentDB에 연결하는 방법을 보여줍니다.

다음 예제에서는 각 *### ## ## ##*자를 클러스터 정보로 바꿉니다.

```
#Include the mongolite library.
library(mongolite)

#Create a MongoDB client, open a connection to Amazon DocumentDB as a replica
#   set and specify the read preference as secondary preferred
client <- mongo(url = "mongodb://sample-user:password@sample-
cluster.node.us-east-1.docdb.amazonaws.com:27017/sample-database?
readPreference=secondaryPreferred&replicaSet=rs0")
```

```
##Insert a single document
str <- c('{"hello" : "Amazon DocumentDB"}')
client$insert(str)

##Find the document that was previously written
client$find()
```

Ruby

다음 코드는 TLS가 활성화된 상태에서 Ruby를 Amazon DocumentDB에 연결하는 방법을 보여줍니다.

다음 예제에서는 각 *### ## ## ##*자를 클러스터 정보로 바꿉니다.

```
require 'mongo'
require 'neatjson'
require 'json'
client_host = 'mongodb://sample-cluster.node.us-east-1.docdb.amazonaws.com:27017'
client_options = {
   database: 'test',
   replica_set: 'rs0',
   read: {:secondary_preferred => 1},
   user: 'sample-user',
   password: 'password',
   retry_writes: false
}

begin
   ##Create a MongoDB client, open a connection to Amazon DocumentDB as a
   ##   replica set and specify the read preference as secondary preferred
   client = Mongo::Client.new(client_host, client_options)

   ##Insert a single document
   x = client[:test].insert_one({"hello":"Amazon DocumentDB"})

   ##Find the document that was previously written
   result = client[:test].find()

   #Print the document
   result.each do |document|
      puts JSON.neat_generate(document)
   end
end
```

```
#Close the connection
client.close
```

Amazon VPC 외부에서 Amazon DocumentDB 클러스터에 연결

Amazon DocumentDB(MongoDB 호환) 클러스터는 Amazon Virtual Private Cloud (Amazon VPC) 내에 배포됩니다. Amazon EC2 인스턴스 또는 동일한 Amazon VPC에 배포된 다른 AWS 서비스에서 직접 액세스할 수 있습니다. 또한 EC2 인스턴스 또는 VPCs 피어링을 통해 동일한 AWS 리전 리전 또는 다른 리전의 다른 VPC에 있는 다른 AWS 서비스에서 Amazon DocumentDB에 액세스할 수 있습니다.

그러나 사용 사례에서 사용자 또는 사용자의 애플리케이션이 클러스터의 VPC 외부에서 Amazon DocumentDB 리소스에 액세스해야 한다고 가정합니다. 그럴 경우 SSH 터널링(포트 포워딩이라고도 함)을 사용하여 Amazon DocumentDB 리소스에 액세스할 수 있습니다.

SSH 터널링에 대한 자세한 내용은 이 주제의 범위를 벗어납니다. SSH 터널링에 대한 자세한 내용은 다음을 참조하세요.

- SSH 터널
- SSH 포트 전송 예, 특히 로컬 전송 섹션

SSH 터널을 만들려면 Amazon DocumentDB 클러스터와 동일한 Amazon VPC에서 실행 중인 Amazon EC2 인스턴스가 필요합니다. 클러스터와 동일한 VPC에서 기존 EC2 인스턴스를 사용하거나 새 인스턴스를 생성할 수 있습니다. 자세한 내용은 사용 중인 운영 체제에 해당하는 주제를 참조하세요.

- Amazon EC2 Linux 인스턴스 시작하기
- Amazon EC2 Windows 인스턴스 시작하기

일반적으로 다음 명령을 사용하여 EC2 인스턴스에 연결할 수 있습니다.

```
ssh -i "ec2Access.pem" ubuntu@ec2-34-229-221-164.compute-1.amazonaws.com
```

그 경우 로컬 컴퓨터에서 다음 명령을 사용하여 SSH 터널을 Amazon DocumentDB 클러스터 `sample-cluster.node.us-east-1.docdb.amazonaws.com`으로 설정할 수 있습니다. `-L` 플래그는 로컬 포트를 전달하는 데 사용됩니다. SSH 터널을 사용하는 경우 클러스터 엔드포인트를 사용하여 클러스터에 연결하고, 오류가 발생하므로 복제본 집합 모드(예: 연결 문자열에 `replicaSet=rs0` 지정)로 연결을 시도하지 않는 것이 좋습니다.

```
ssh -i "ec2Access.pem" -L 27017:sample-cluster.node.us-east-1.docdb.amazonaws.com:27017
 ubuntu@ec2-34-229-221-164.compute-1.amazonaws.com -N
```

SSH 터널이 생성된 이후에 `localhost:27017`에 대해 실행한 명령은 Amazon VPC에서 실행 중인 Amazon DocumentDB 클러스터 `sample-cluster`에 전달됩니다. Amazon DocumentDB 클러스터에서 전송 계층 보안 (TLS) 이 활성화된 경우 Amazon DocumentDB의 퍼블릭 키를 에서 다운로드해야 합니다. https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem 다음 작업은 이 파일을 다운로드합니다:

```
wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem
```

> **Note**
>
> 새 Amazon DocumentDB 클러스터에는 기본적으로 TLS가 사용 설정되어 있습니다. 그러나 비활성화는 가능합니다. 자세한 내용은 Amazon DocumentDB 클러스터 TLS 설정 관리 단원을 참조하십시오.

Amazon VPC 외부에서 Amazon DocumentDB 클러스터에 연결하려면 다음 명령을 사용하세요.

```
mongo --sslAllowInvalidHostnames --ssl --sslCAFile global-bundle.pem --username
 <yourUsername> --password <yourPassword>
```

Amazon DocumentDB에 복제본 세트로 연결

Amazon DocumentDB(MongoDB 호환)를 사용하여 개발할 때는 클러스터를 복제본 세트로 연결하고 드라이버의 내장된 읽기 환경설정 기능을 사용하여 복제본 인스턴스에 읽기를 배포하는 것이 좋습니다. 이 섹션에서는 이것이 의미하는 바에 대해 자세히 설명하고, Python용 SDK를 예로 들어 복제본 세트로 Amazon DocumentDB 클러스터에 연결하는 방법을 설명합니다.

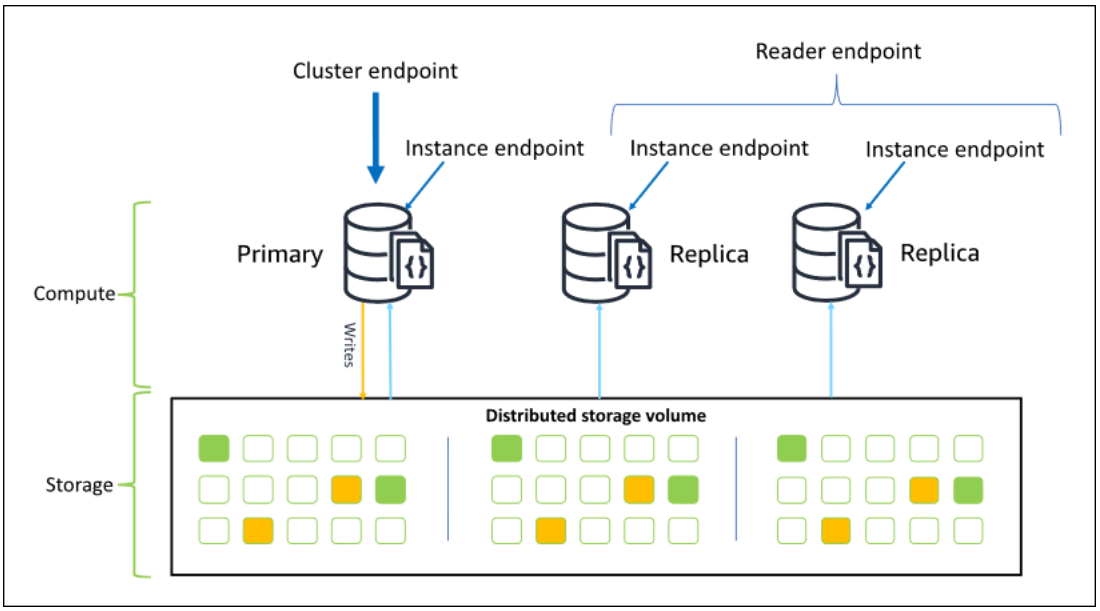
Amazon DocumentDB에는 클러스터에 연결하는 데 사용할 수 있는 세 가지 엔드포인트가 있습니다:

- 클러스터 엔드포인트
- 리더 엔드포인트
- 인스턴스 엔드포인트

대부분의 경우 Amazon DocumentDB에 연결할 때 클러스터 엔드포인트을 사용하는 것이 좋습니다. 이는 다음 다이어그램과 같이 클러스터의 기본 인스턴스를 가리키는 CNAME입니다.

SSH 터널을 사용하는 경우 클러스터 엔드포인트를 사용하여 클러스터에 연결하고, 오류가 발생하므로 복제본 집합 모드(예: 연결 문자열에 `replicaSet=rs0` 지정)로 연결을 시도하지 않는 것이 좋습니다.

> **Note**
>
> Amazon DocumentDB 엔드포인트에 대한 자세한 내용은 Amazon DocumentDB 엔드포인트 섹션을 참조하세요.



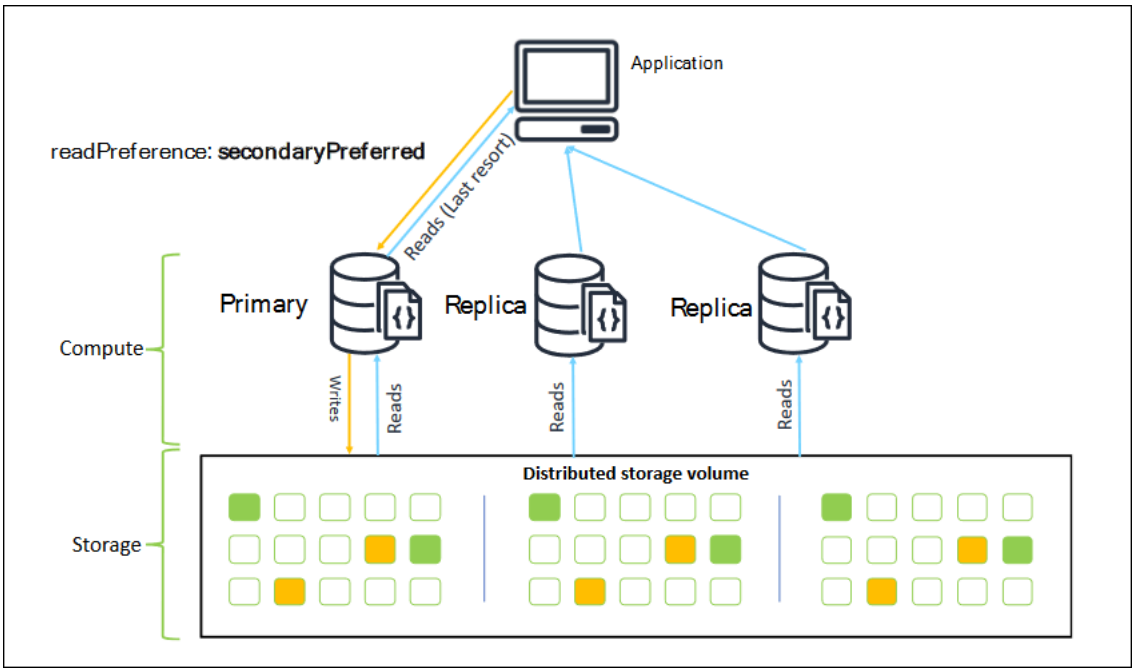
클러스터 엔드포인트를 사용하여 복제본 세트 모드에서 클러스터에 연결할 수 있습니다. 그런 다음 내장된 읽기 기본 설정 드라이버 기능을 사용할 수 있습니다. 다음 예제에서 `/?replicaSet=rs0`을 지정하면 복제본 세트로 연결하려는 SDK를 나타냅니다. `/?replicaSet=rs0'`을 생략하면 클라이언트는 모든 요청을 클러스터 엔드포인트, 즉 기본 인스턴스로 라우팅합니다.

```
## Create a MongoDB client, open a connection to Amazon DocumentDB as a
   ## replica set and specify the read preference as secondary preferred
   client = pymongo.MongoClient('mongodb://<user-name>:<password>@mycluster.node.us-
east-1.docdb.amazonaws.com:27017/?replicaSet=rs0')
```

복제본 세트로 연결하면 클러스터에서 인스턴스를 추가하거나 제거할 때를 포함하여 SDK가 클러스터 지형을 자동으로 검색할 수 있다는 장점이 있습니다. 그런 다음 읽기 요청을 복제본 인스턴스로 라우팅하여 클러스터를 보다 효율적으로 사용할 수 있습니다.

복제본 세트로 연결할 때 연결을 위해 `readPreference`를 지정할 수 있습니다. `secondaryPreferred`의 읽기 기본 설정을 지정하면 클라이언트는 읽기 쿼리를 복제본에 라우팅하고 쿼리를 기본 인스턴스에 씁니다(다음 다이어그램 참조). 이것은 클러스터 리소스를 더 잘 사용하는 것입니다. 자세한 내용은 읽기 기본 설정 옵션 섹션을 참조하십시오.

```
## Create a MongoDB client, open a connection to Amazon DocumentDB as a
   ##   replica set and specify the read preference as secondary preferred
client = pymongo.MongoClient('mongodb://<user-name>:<password>@mycluster.node.us-
east-1.docdb.amazonaws.com:27017/?replicaSet=rs0&readPreference=secondaryPreferred')
```



Amazon DocumentDB 복제본의 읽기는 결국 일치합니다. 기본 데이터에 기록된 순서와 동일한 순서로 데이터를 반환하며 50ms 미만의 복제 지연이 있는 경우가 많습니다. Amazon CloudWatch 지표 `DBInstanceReplicaLag`및 `DBClusterReplicaLagMaximum`를 사용하여 클러스터의 복제 지연을 모니터링할 수 있습니다. 자세한 내용은 CloudWatch를 사용하여 Amazon DocumentDB 모니터링 섹션을 참조하십시오.

기존의 단일 데이터베이스 아키텍처와 달리 Amazon DocumentDB는 스토리지와 컴퓨팅을 분리합니다. 이러한 최신 아키텍처를 고려할 때, 복제본 인스턴스에서 읽기 확장을 권장합니다. 복제본 인스턴

스에 대한 읽기는 기본 인스턴스에서 복제되는 쓰기를 차단하지 않습니다. 클러스터에 최대 15개의 읽기 전용 복제본 인스턴스를 추가하고 초당 수백만 건의 읽기로 스케일 아웃할 수 있습니다.

복제본 세트로 연결하고 복제본에 읽기를 배포할 때 얻을 수 있는 주요 이점은 클러스터에서 애플리케이션에 사용할 수 있는 전체 리소스가 늘어난다는 것입니다. 모범 사례로 복제본 세트로 연결하는 것을 권장합니다. 또한 다음 시나리오에서 가장 일반적으로 권장됩니다:

- 기본에서 거의 100% CPU를 사용하고 있습니다.
- 버퍼 캐시 적중률이 거의 0입니다.
- 개별 인스턴스의 연결 또는 커서 한계에 도달합니다.

클러스터 인스턴스 크기를 확장하는 것은 옵션이며 경우에 따라 클러스터를 확장하는 가장 좋은 방법일 수 있습니다. 그러나 이미 클러스터에 있는 복제본을 더 잘 사용하는 방법도 고려해야 합니다. 이를 통해 더 큰 인스턴스 유형을 사용하는 비용을 늘리지 않고도 규모를 조정할 수 있습니다. 또한 CloudWatch 알람을 사용하여 이러한 제한(즉, `CPUUtilization`, `DatabaseConnections`, 및 `BufferCacheHitRatio`)을 모니터링하고 경고하여 리소스가 많이 사용되는 시기를 파악하는 것이 좋습니다.

자세한 내용은 다음 주제를 참조하세요.

- Amazon DocumentDB 모범 사례
- Amazon DocumentDB 할당량 및 제한

클러스터 연결 사용

클러스터에서 모든 연결을 사용하는 시나리오를 고려하세요. 예를 들어, `r5.2xlarge` 인스턴스에는 4,500개의 연결(및 450개의 열려 있는 커서) 제한이 있습니다. 3-인스턴스 Amazon DocumentDB 클러스터를 생성하고 클러스터 끝점을 사용하여 기본 인스턴스에만 연결하는 경우 열린 연결 및 커서에 대한 클러스터 제한은 각각 4,500 및 450입니다. 컨테이너에서 작성하는 많은 작업자를 사용하는 애플리케이션을 작성하는 경우 이러한 제한에 도달할 수 있습니다. 컨테이너는 한 번에 많은 연결을 열고 클러스터를 포화시킵니다.

대신 복제본 세트로 Amazon DocumentDB 클러스터에 연결하고 읽기를 복제본 인스턴스에 분배할 수 있습니다. 그런 다음 클러스터에서 사용 가능한 연결 및 커서 수를 각각 13,500 및 1,350으로 세 배로 늘릴 수 있습니다. 클러스터에 인스턴스를 더 추가하면 읽기 워크로드에 대한 연결 및 커서 수가 증가합니다. 클러스터에 쓰기 위해 연결 수를 늘려야 하는 경우 인스턴스 크기를 늘리는 것이 좋습니다.

> **Note**
>
> large, xlarge 및 2xlarge 인스턴스에 대한 연결 수는 인스턴스 크기가 4,500개가 될 때까지 증가합니다. 4xlarge 인스턴스 이상의 경우 인스턴스당 최대 연결 수는 4,500개입니다. 인스턴스 유형별 제한에 대한 자세한 내용은 인스턴스 제한 섹션을 참조하세요.

일반적으로 secondary 읽기 환경 설정을 사용하여 클러스터에 연결하지 않는 것이 좋습니다. 클러스터에 복제본 인스턴스가 없으면 읽기가 실패하기 때문입니다. 예를 들어, 기본 복제본과 복제본이 하나인 두 개의 인스턴스 Amazon DocumentDB 클러스터가 있다고 가정합니다. 복제본에 문제가 있으면 secondary로 설정된 연결 풀의 읽기 요청은 실패합니다. secondaryPreferred는 클라이언트가 연결하기에 적합한 복제본 인스턴스를 찾을 수 없는 경우 기본 인스턴스로 돌아가서 읽을 수 있다는 장점이 있습니다.

다중 연결 풀

일부 시나리오에서는 애플리케이션의 읽기가 쓰기 후 읽기 일관성을 가져야 하며, 이는 Amazon DocumentDB의 기본 인스턴스에서만 제공될 수 있습니다. 이러한 시나리오에서는 두 개의 클라이언트 연결 풀을 작성할 수 있습니다. 하나는 쓰기용이고 다른 하나는 쓰기 후 읽기 일관성이 필요한 읽기용입니다. 이를 위한 코드는 다음과 같을 것입니다.

```
## Create a MongoDB client,
##    open a connection to Amazon DocumentDB as a replica set and specify the
 readPreference as primary
clientPrimary = pymongo.MongoClient('mongodb://<user-
name>:<password>@mycluster.node.us-east-1.docdb.amazonaws.com:27017/?
replicaSet=rs0&readPreference=primary')

## Create a MongoDB client,
##    open a connection to Amazon DocumentDB as a replica set and specify the
 readPreference as secondaryPreferred
secondaryPreferred = pymongo.MongoClient('mongodb://<user-
name>:<password>@mycluster.node.us-east-1.docdb.amazonaws.com:27017/?
replicaSet=rs0&readPreference=secondaryPreferred')
```

다른 옵션은 단일 연결 풀을 작성하고 주어진 컬렉션에 대한 읽기 환경 설정을 덮어쓰는 것입니다.

```
##Specify the collection and set the read preference level for that collection
col = db.review.with_options(read_preference=ReadPreference.SECONDARY_PREFERRED)
```

요약

클러스터의 리소스를 더 잘 사용하려면 복제본 세트 모드를 사용하여 클러스터에 연결하는 것이 좋습니다. 애플리케이션에 적합한 경우 읽기를 복제본 인스턴스에 분배하여 애플리케이션 읽기를 확장할 수 있습니다.

Studio 3T에서 Amazon DocumentDB 클러스터에 연결

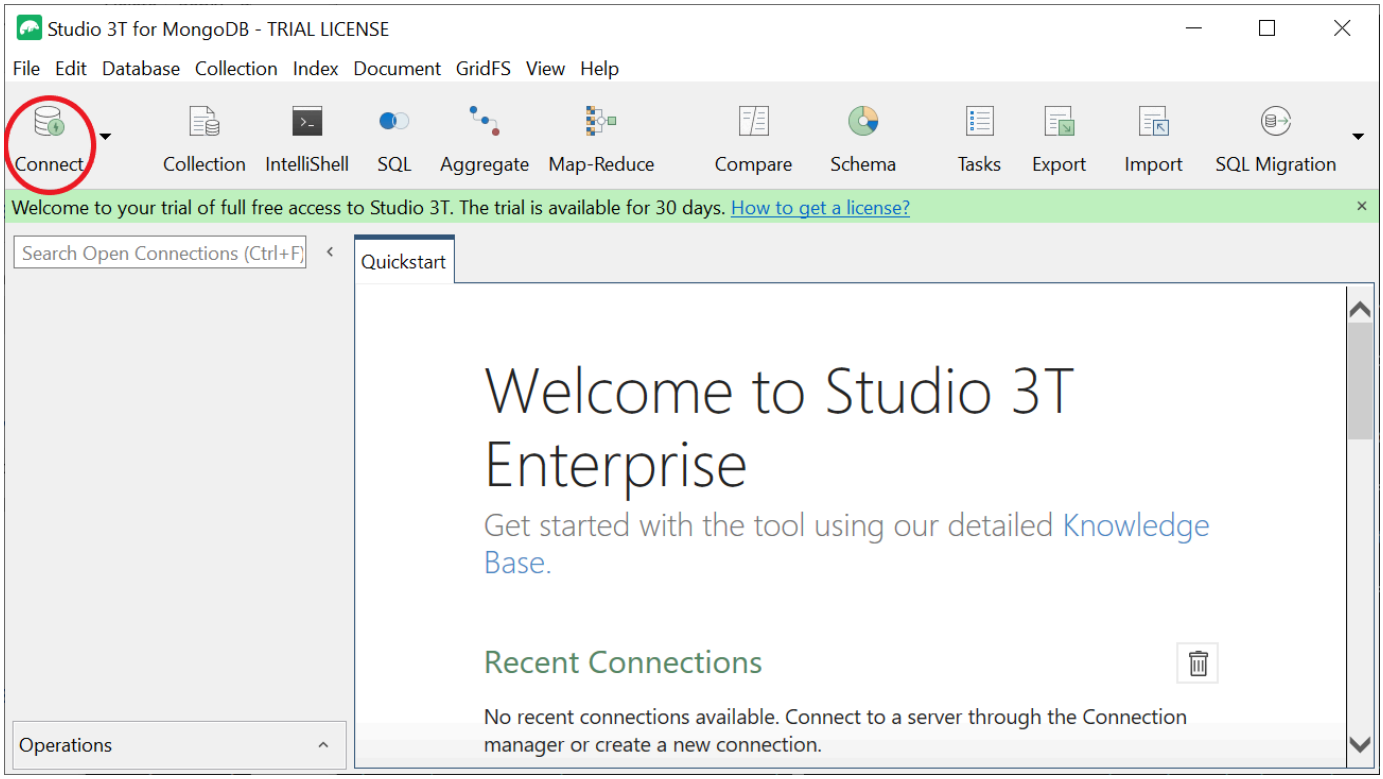
Studio 3T는 MongoDB를 사용하는 개발자와 데이터 엔지니어에게 널리 사용되는 GUI 및 IDE입니다. 데이터의 트리, 테이블 및 JSON 보기, CSV, JSON, SQL 및 BSON/Mongodump로 쉽게 가져오기/내보내기, 유연한 쿼리 옵션, 시각적 드래그 앤 드롭 UI, 자동 완성 기능이 있는 내장 몽고 쉘, 집계 파이프라인 편집기 및 SQL 쿼리 지원 등의 여러 가지 강력한 기능을 제공합니다.

사전 조건

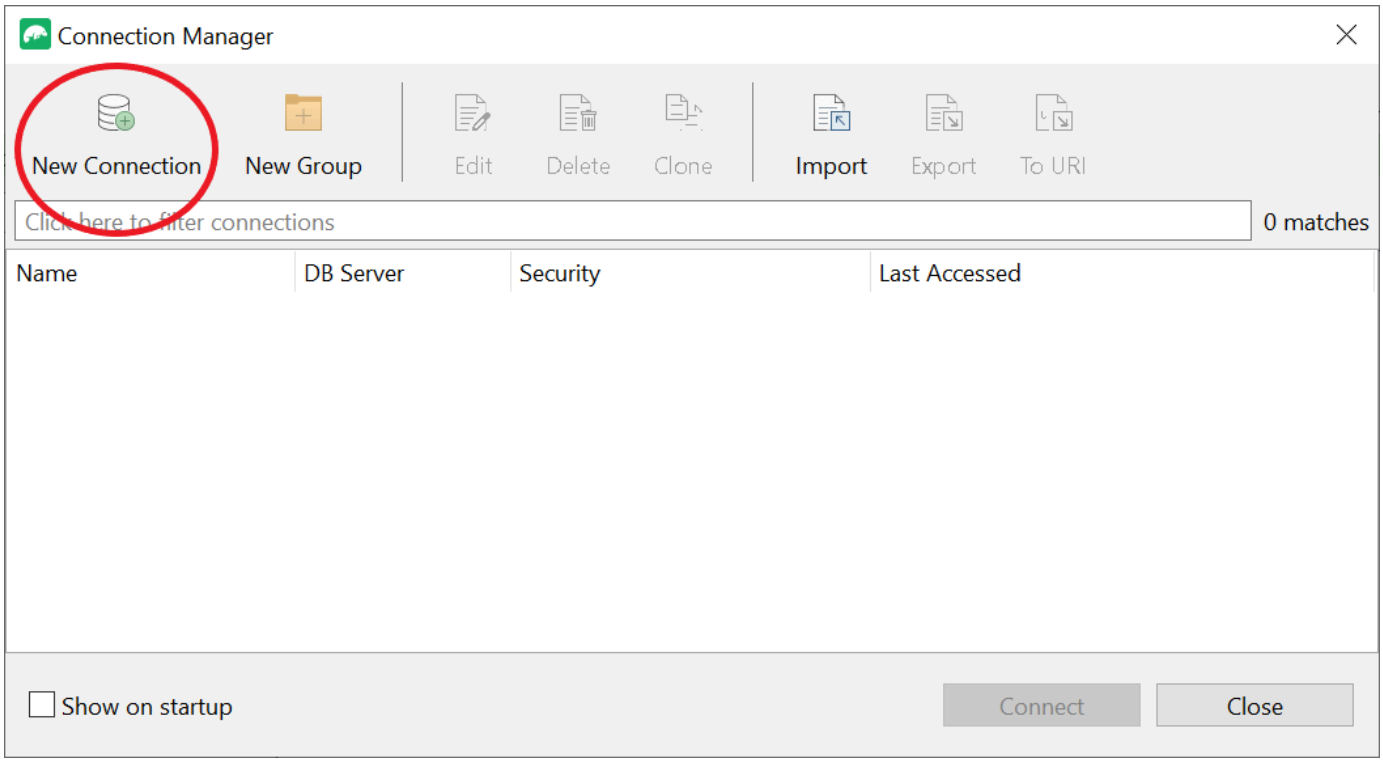
- Amazon DocumentDB에서 Amazon EC2를 배스천/점프 호스트로 아직 사용하고 있지 않은 경우, Amazon EC2와 연결하는 방법에 대한 지침을 따르세요.
- Studio 3T가 없는 경우, 다운로드하여 설치합니다.

Studio 3T와 연결

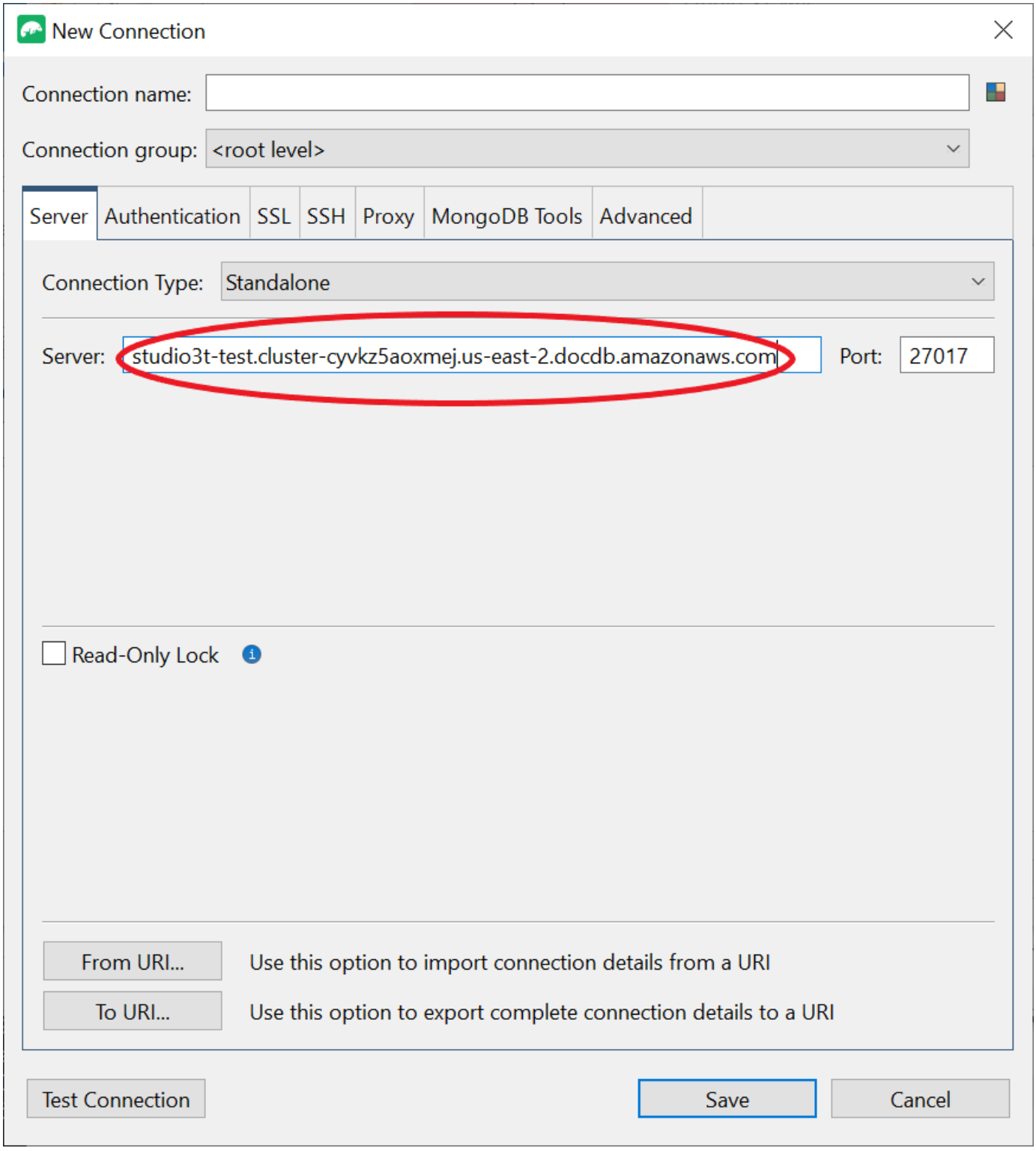
1. 도구 모음의 왼쪽 상단에서 연결을 선택합니다.



2. 도구 모음의 왼쪽 상단에서 새 연결을 선택합니다.

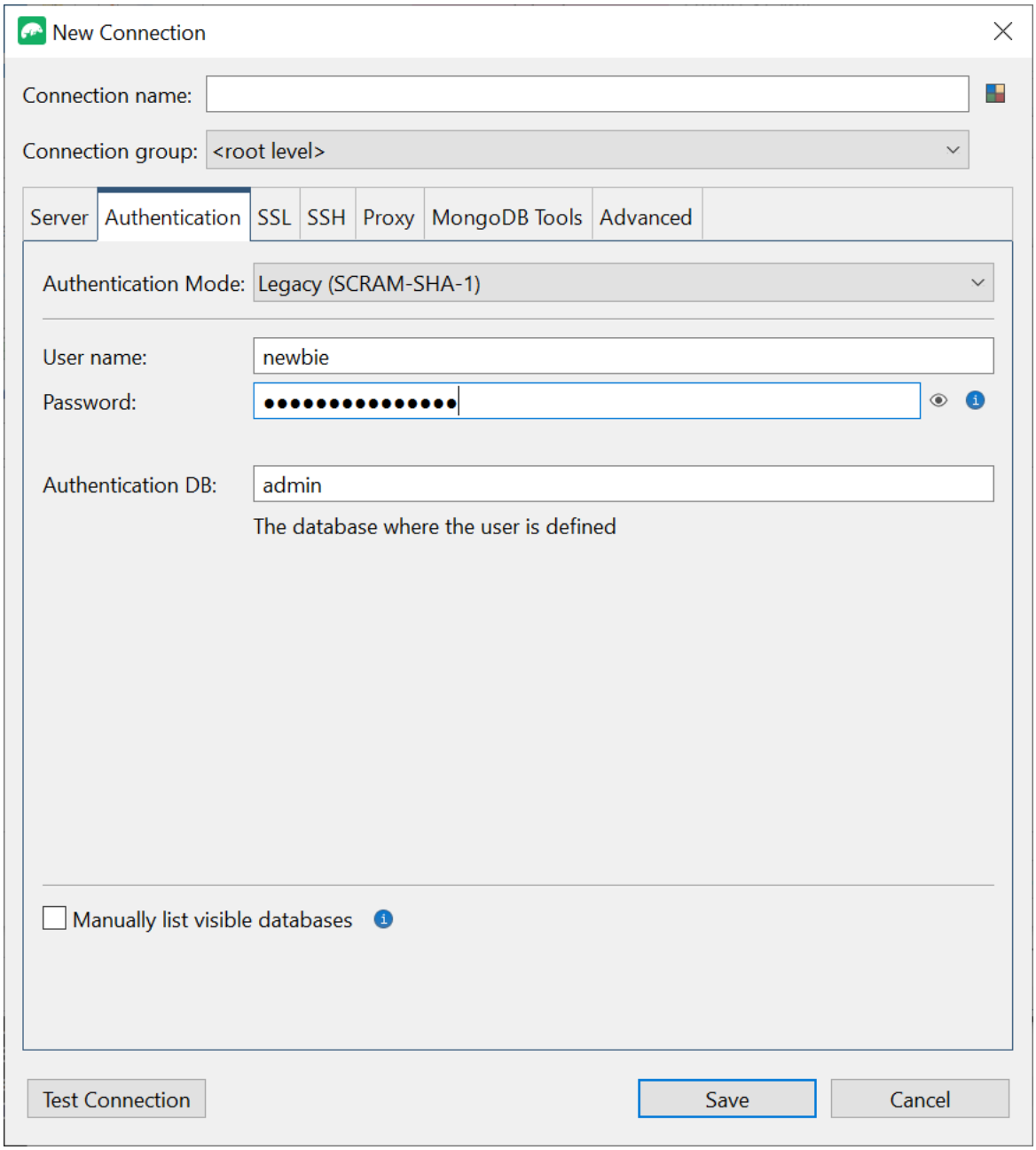


3. 서버 탭의 서버 필드에 클러스터 엔드포인트 정보를 입력합니다.

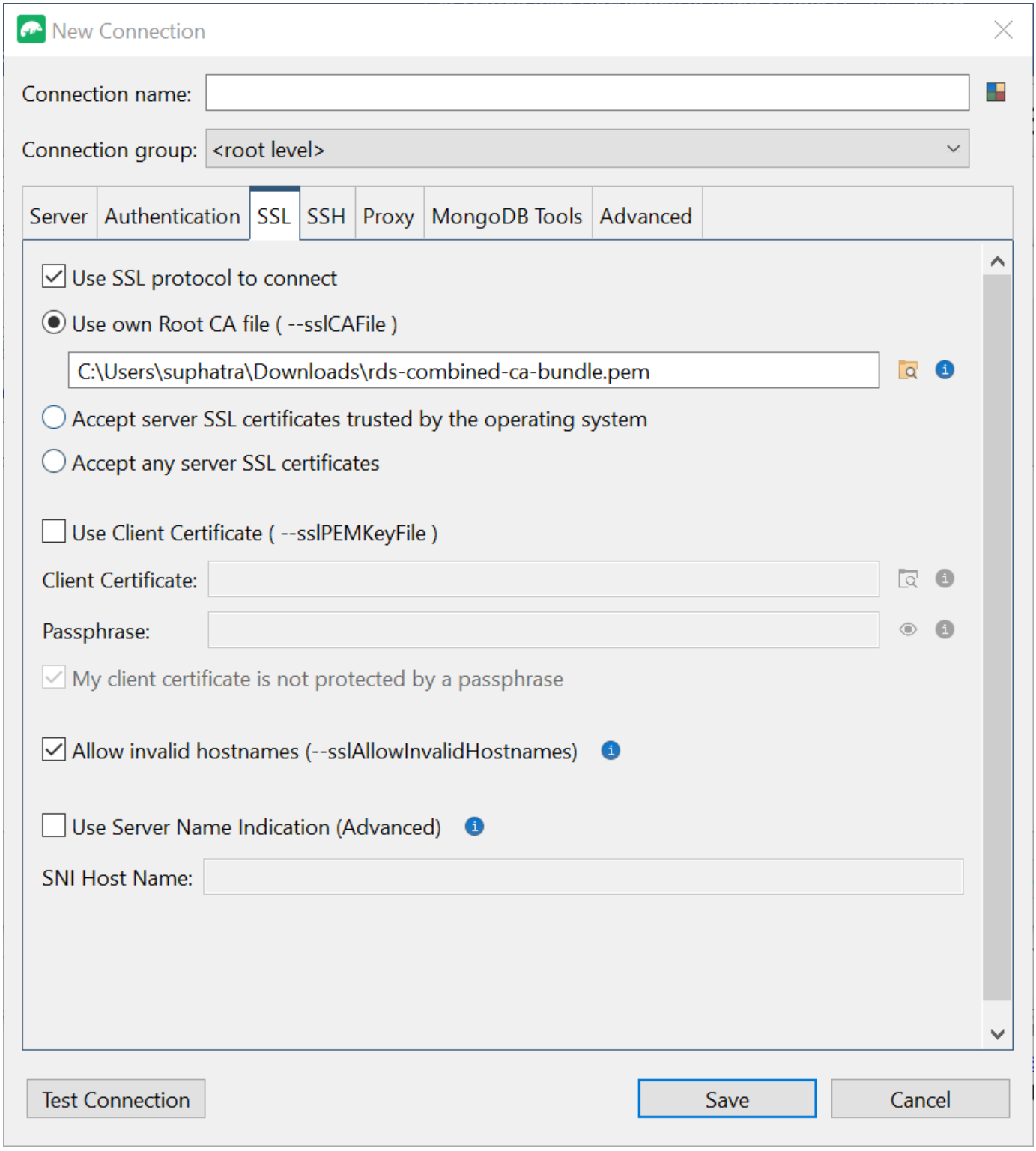


> **Note**
>
> 클러스터 엔드포인트를 찾을 수 없나요? 여기의 단계를 따르기만 하면 됩니다.

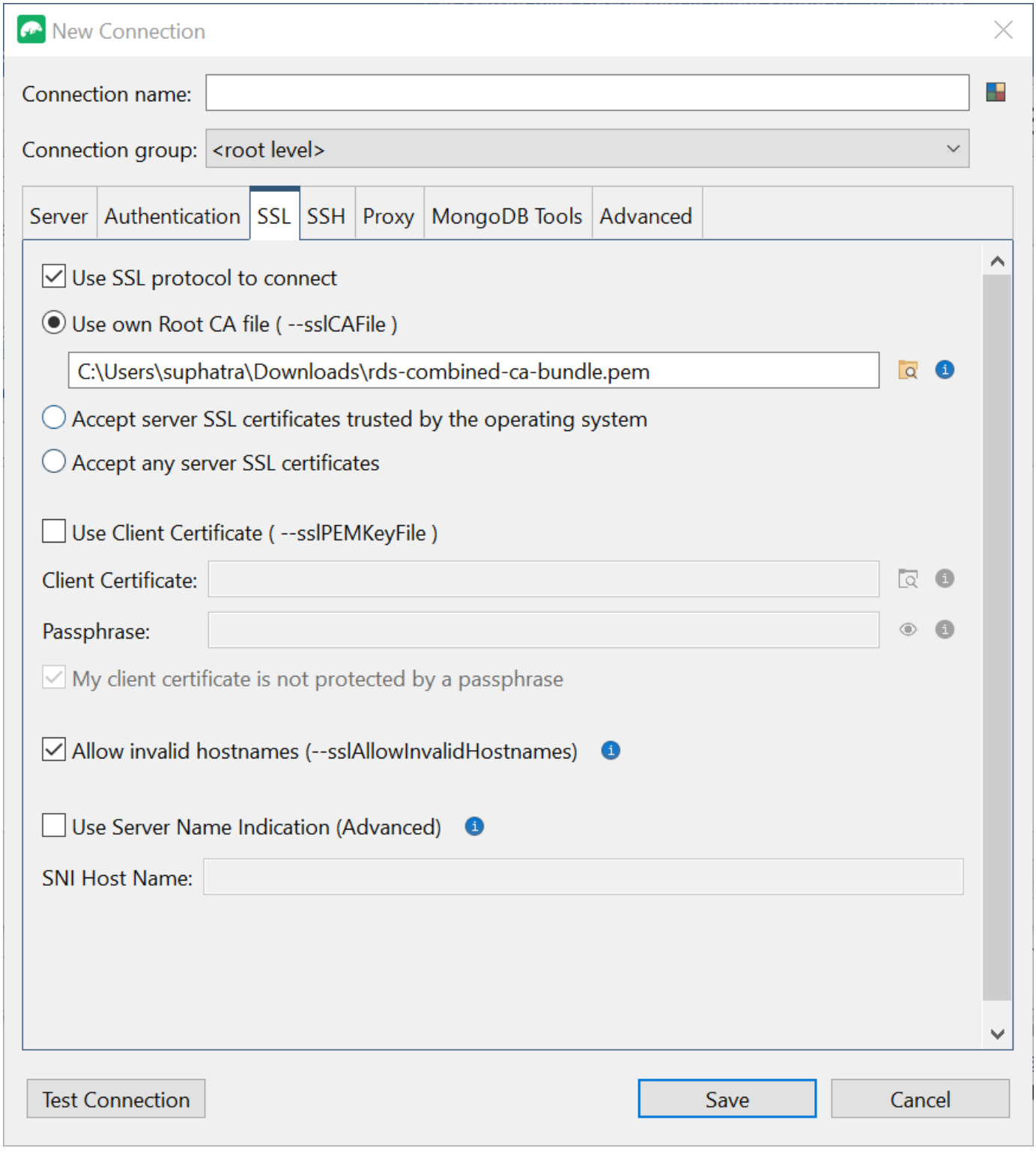
4. 인증 탭을 선택하고 인증 모드의 드롭다운 메뉴에서 레거시를 선택합니다.



5. 사용자 이름 및 암호 필드에 사용자 이름과 자격 증명을 입력합니다.
6. SSL 탭을 선택하고 SSL 프로토콜을 사용하여 연결 박스를 선택합니다.



7. 자체 루트 CA 파일 사용을 선택합니다. 그런 다음 Amazon DocumentDB 인증서를 추가합니다(DocumentDB 클러스터에서 SSL이 비활성화된 경우 이 단계를 건너뛸 수 있음). 잘못된 호스트 이름을 허용하려면 박스를 선택합니다.



Note

인증서가 없으세요? 다음 명령을 사용하여 다운로드할 수 있습니다.

```
wget https://truststore.pki.rds.amazonaws.com/global/global-
bundle.pem
```

8. Amazon VPC 외부의 클라이언트 머신에서 연결하는 경우 SSH 터널을 생성해야 합니다. SSH 탭에서 이 작업을 수행합니다.

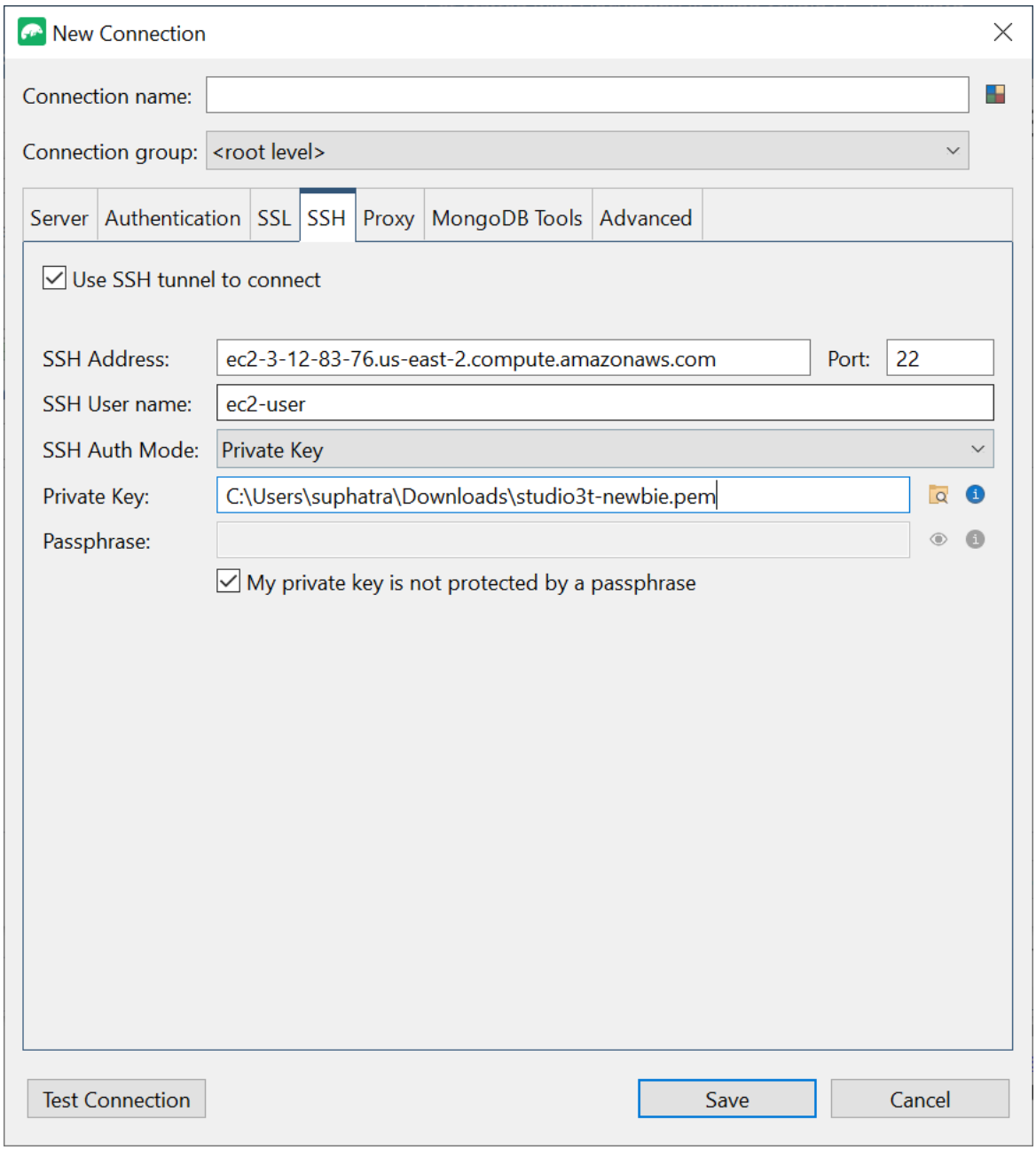
   a. SSH 터널 사용 박스를 선택하고 SSH 주소 필드에 SSH 주소를 입력합니다. 인스턴스 퍼블릭 DNS(IPV4)입니다. Amazon EC2 관리 콘솔에서 이 URL을 가져올 수 있습니다.

   b. 사용자 이름을 입력합니다. Amazon EC2 인스턴스의 사용자 이름입니다.

   c. SSH 인증 모드의 경우 프라이빗 키를 선택합니다. 프라이빗 키 필드에서 파일 찾기 아이콘을 선택하여 Amazon EC2 인스턴스의 프라이빗 키를 찾아 선택합니다. Amazon EC2 콘솔에서 인스턴스를 생성할 때 저장한 .pem 파일(키 페어)입니다.

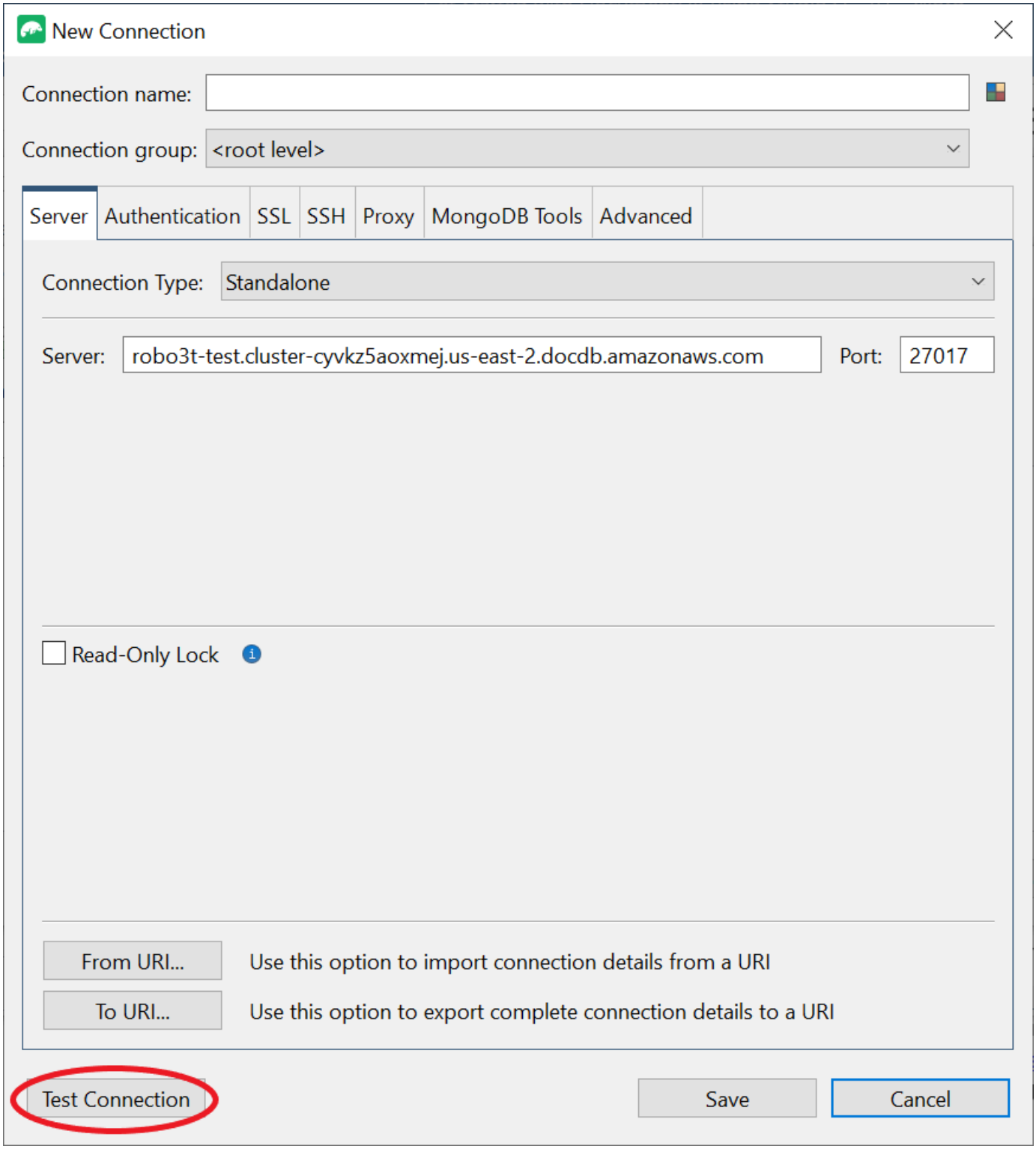
   d. Linux/macOS 클라이언트 머신을 사용하는 경우 다음 명령을 사용하여 프라이빗 키의 권한을 변경해야 할 수 있습니다.

      ```
      chmod 400 /fullPathToYourPemFile/<yourKey>.pem
      ```

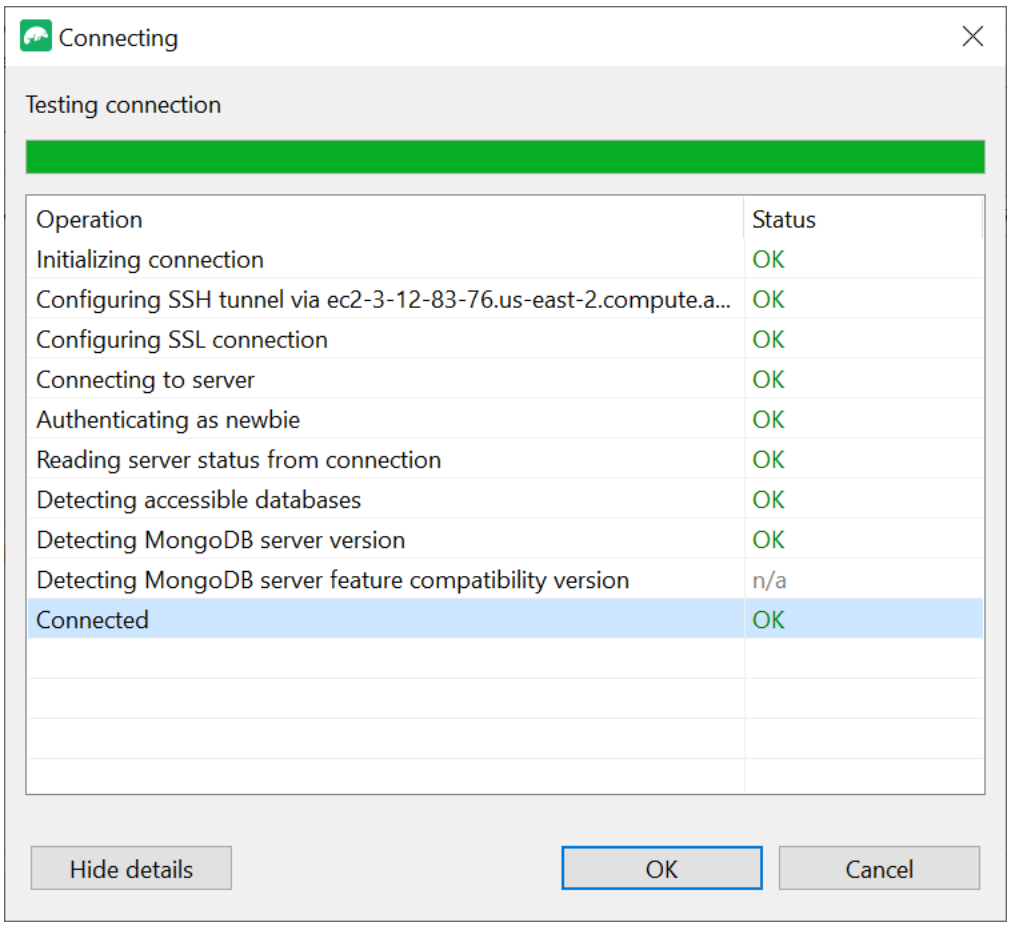


> **Note**
>
> Amazon EC2 인스턴스는 DocumentDB 클러스터와 동일한 Amazon VPC 및 보안 그룹에 있어야 합니다. Amazon EC2 관리 콘솔에서 SSH 주소, 사용자 이름 및 프라이빗 키를 가져올 수 있습니다.

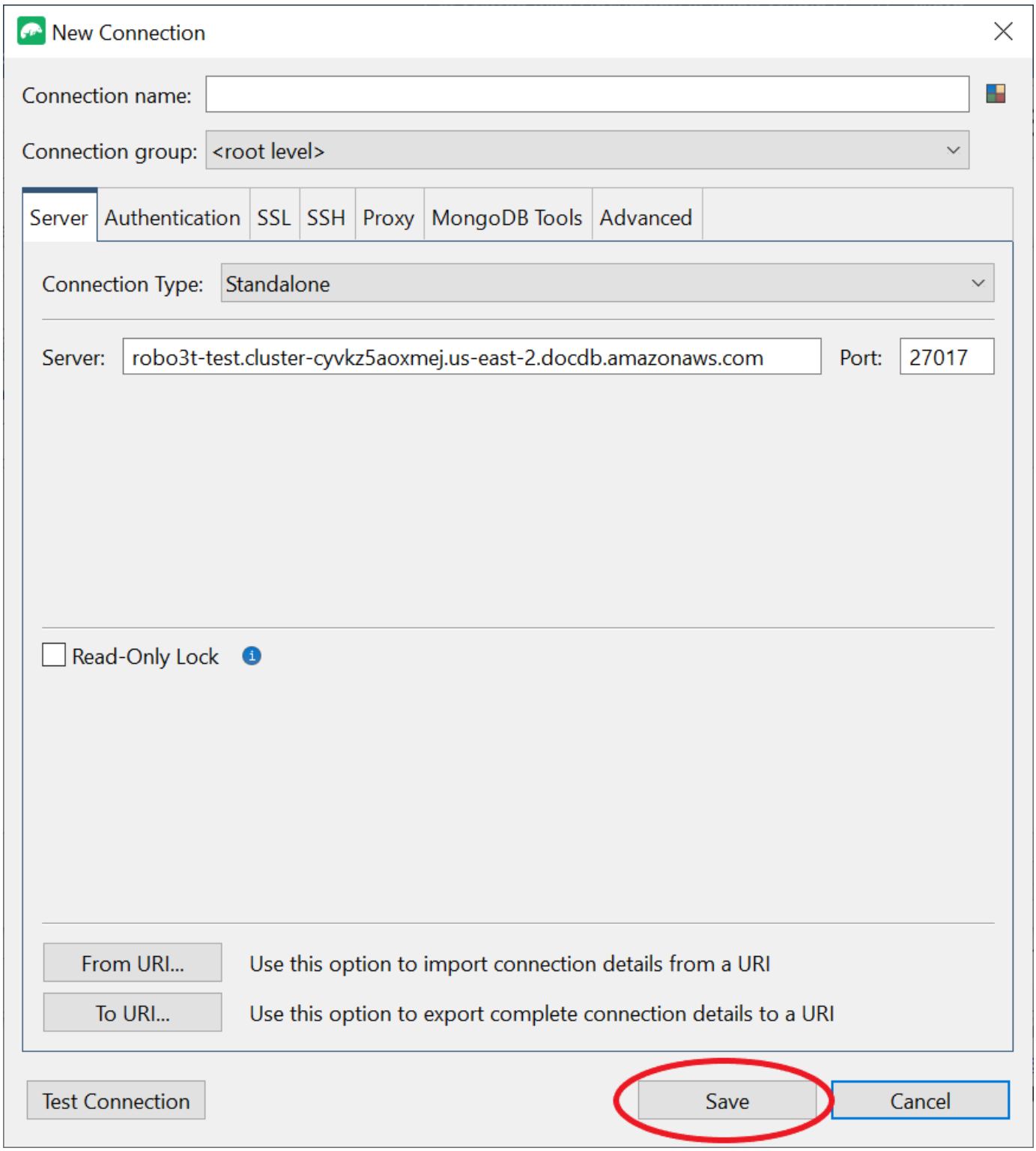
9. 이제 연결 테스트 버튼을 선택하여 구성을 테스트합니다.



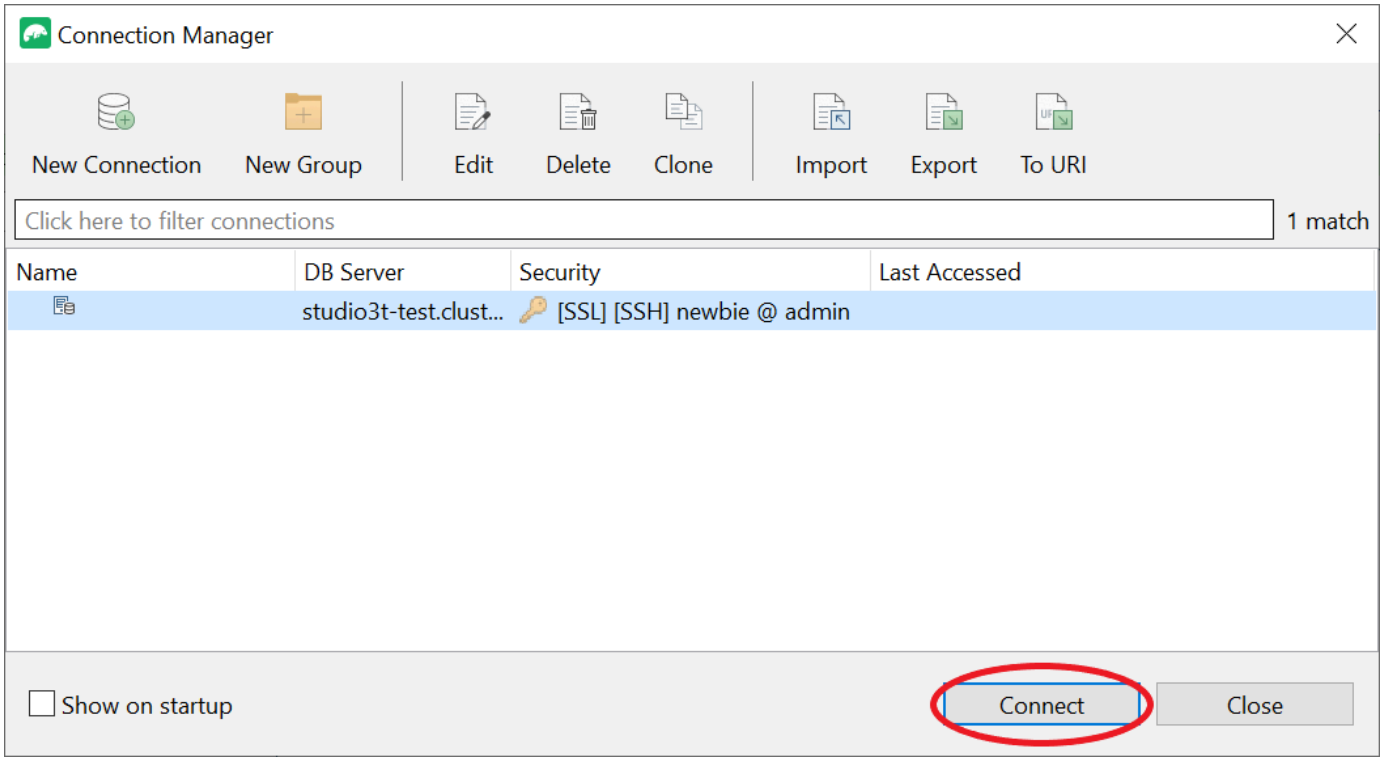
10. 진단 창에 녹색 막대가 로드되어 테스트가 성공했음을 나타냅니다. 이제 OK를 선택하여 진단 창을 닫습니다.



11. 나중에 사용할 수 있도록 연결을 저장하려면 저장을 선택합니다.



12. 이제 클러스터를 선택하고 연결을 선택합니다.



축하합니다! 이제 Studio 3T를 통해 Amazon DocumentDB 클러스터에 성공적으로 연결되었습니다.

DataGrip을 사용하여 Amazon DocumentDB에 연결

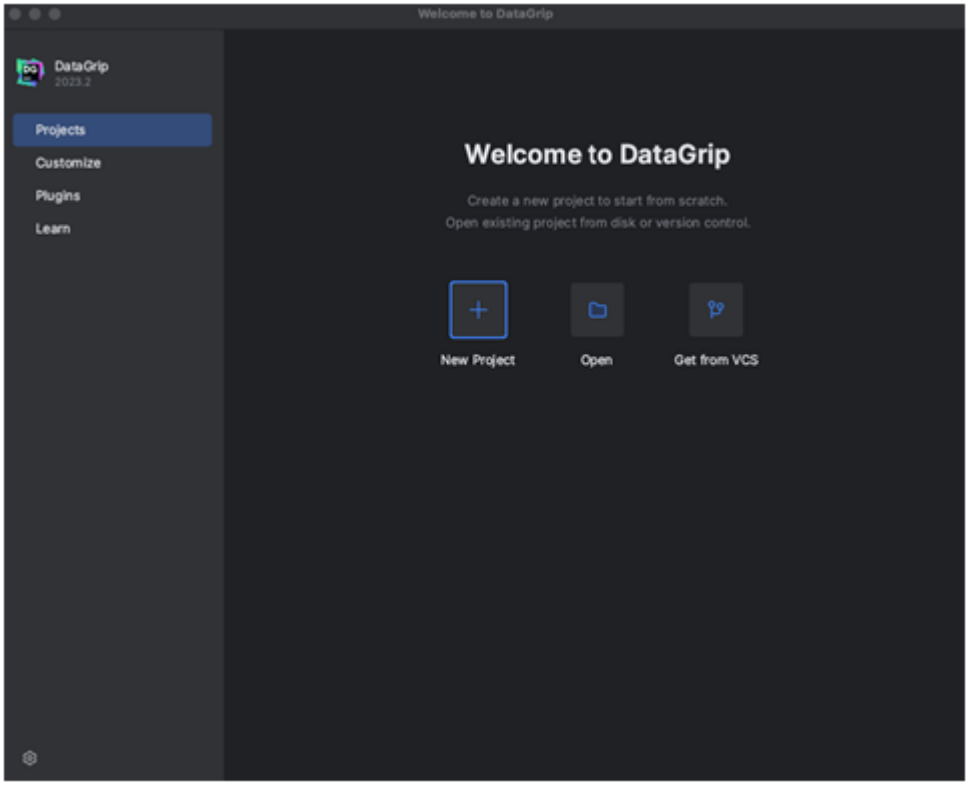
DataGrip은 Amazon DocumentDB를 비롯한 다양한 데이터베이스 시스템을 지원하는 강력한 통합 개발 환경(IDE)입니다. 이 섹션에서는 DataGrip을 사용하여 Amazon DocumentDB 클러스터에 연결하는 단계를 안내하며, 그래픽 인터페이스를 사용하여 데이터를 쉽게 관리하고 쿼리할 수 있습니다.

사전 조건

- 컴퓨터에 데이터그립 IDE가 설치되어 있습니다. JetBrains에서 다운로드할 수 있습니다.
- Amazon DocumentDB 클러스터와 동일한 VPC에서 실행 중인 Amazon EC2 인스턴스입니다. 이 인스턴스를 사용하여 로컬 시스템에서 Amazon DocumentDB 클러스터로 연결되는 보안 터널을 설정합니다. Amazon EC2를 사용하여 연결 방법을 알려면 다음 지침에 따르십시오.
- Amazon EC2 인스턴스, VPN 연결 또는 이미 보안 VPN을 사용하여 AWS 인프라에 액세스하는 경우의 대안입니다. 이 옵션을 선호하는 경우 지침에 따라를 사용하여 Amazon DocumentDB에 안전하게 액세스 AWS Client VPN합니다.

DataGrip을 사용한 연결

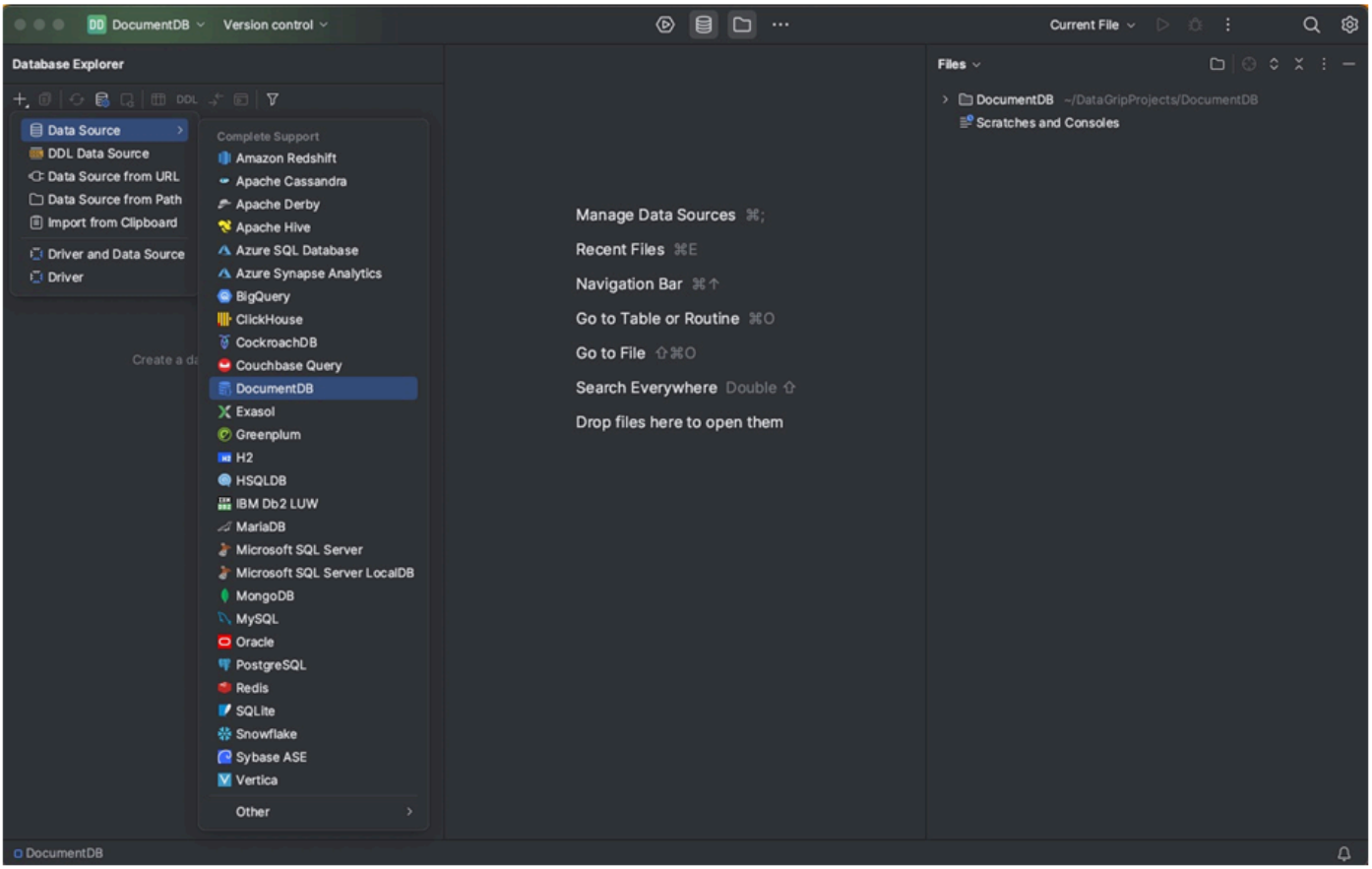
1. 컴퓨터에서 DataGrip을 시작하고 새 프로젝트를 생성합니다.



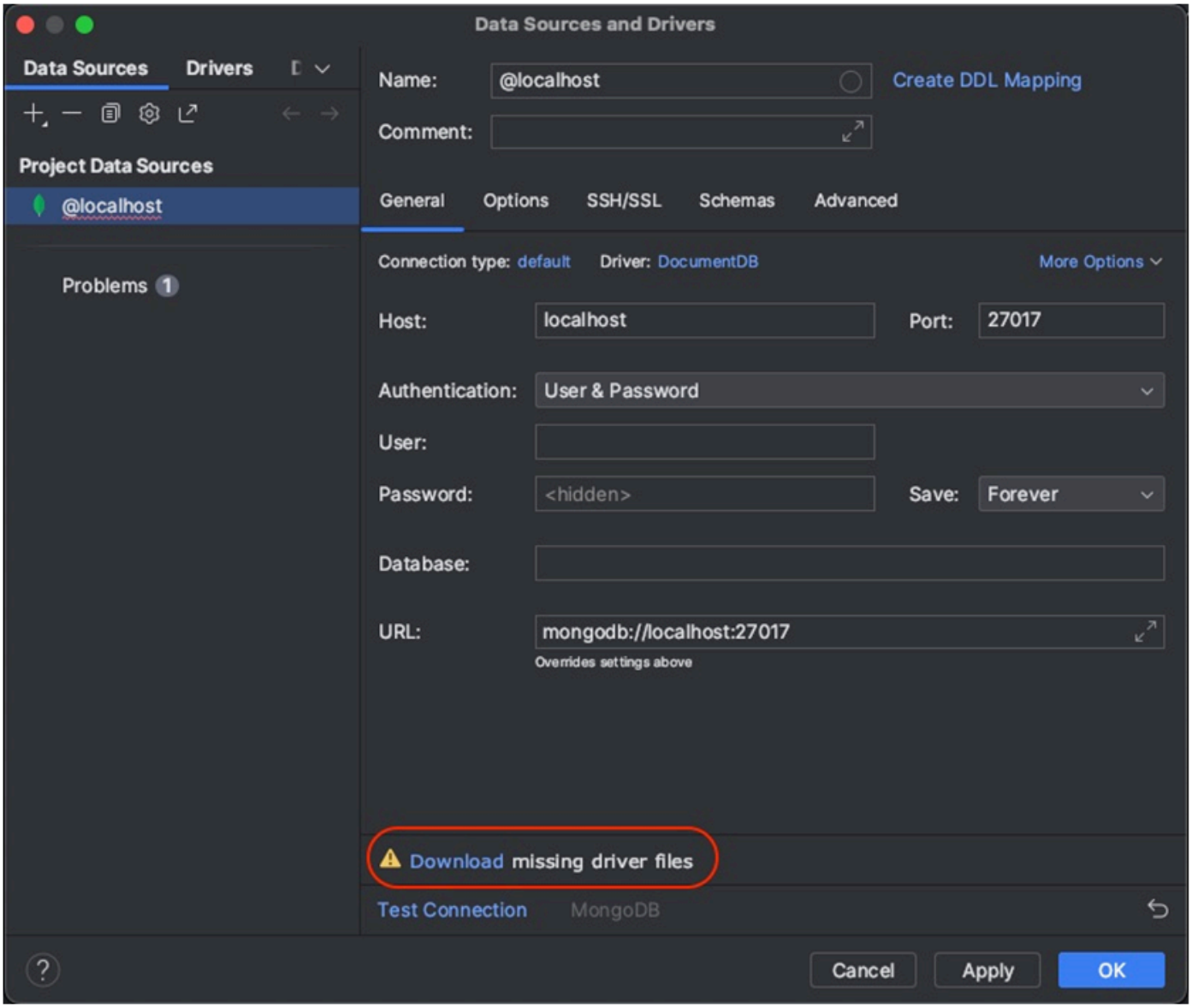
2. 다음 방법 중 하나로 새 데이터 원본을 추가합니다.

   a. 기본 메뉴에서 파일 — 새로 만들기 — 데이터 원본으로 이동한 다음 DocumentDB를 선택합니다.

   b. 데이터베이스 탐색기의 도구 모음에서 새 아이콘 (+) 을 클릭합니다. 데이터 소스로 이동하여 DocumentDB를 선택합니다.



3. 일반 탭의 데이터 소스 페이지에서 연결 설정 영역 하단에 누락된 드라이버 파일 다운로드 링크가 있는지 확인합니다. 이 링크를 클릭하면 데이터베이스와 상호 작용하는 데 필요한 드라이버를 다운로드할 수 있습니다. 직접 다운로드 링크는 JetBrains JDBC 드라이버를 참조하십시오.



4. 일반 탭에서 연결 세부 정보를 지정합니다.

   a. 호스트 필드에 Amazon DocumentDB 클러스터 엔드포인트를 지정합니다.

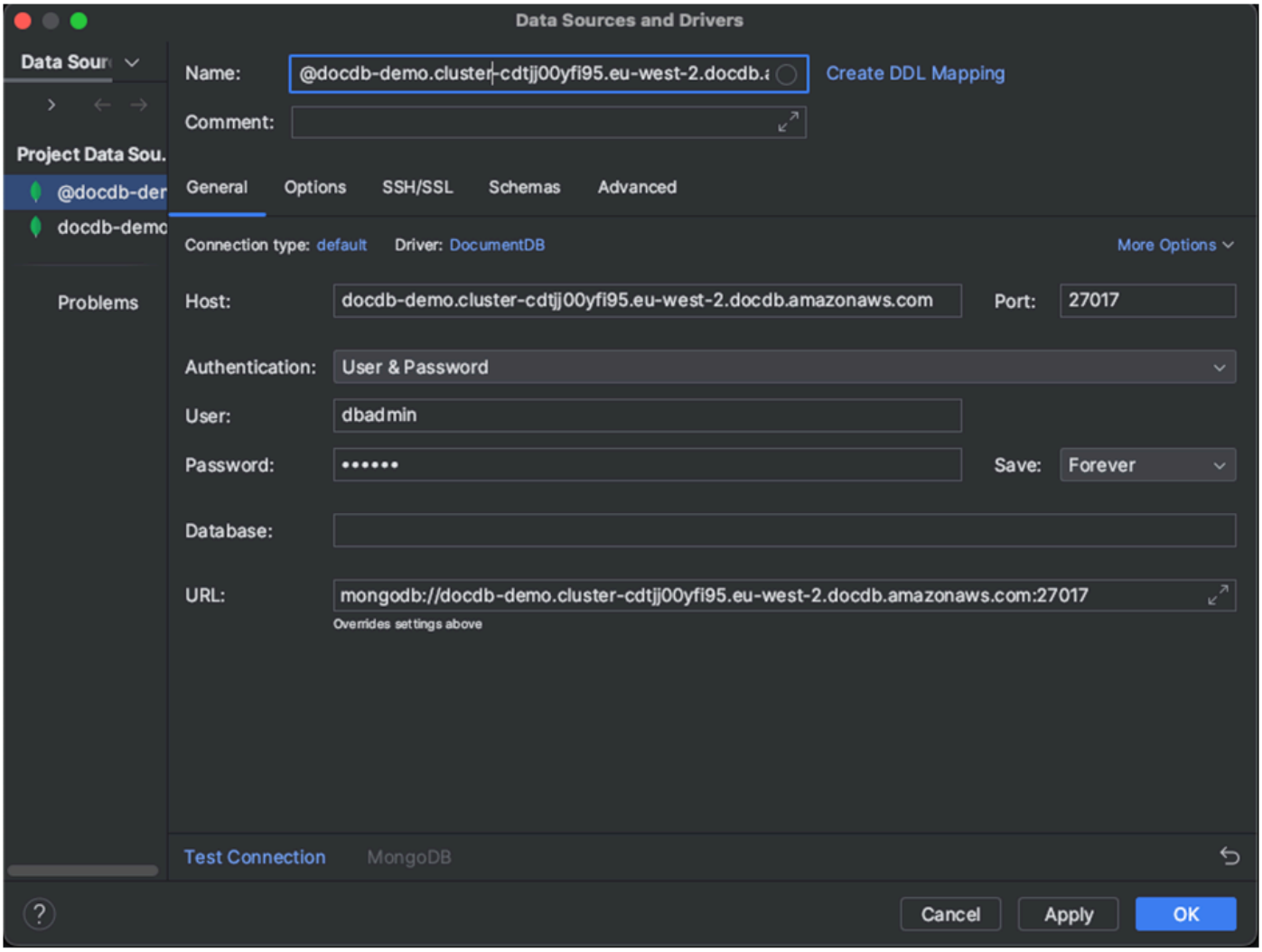
   b. 포트는 이미 27017로 설정되어 있습니다. 클러스터가 다른 포트에 배포된 경우 변경하십시오.

   c. 인증에서 사용자 및 암호를 선택합니다.

   d. 사용자 이름 및 암호 정보를 입력하십시오.

   e. 데이터베이스 필드는 선택 사항입니다. 연결할 데이터베이스를 지정할 수 있습니다.

   f. 위의 세부 정보를 추가하면 URL 필드가 자동으로 완성됩니다.



5. SSH/SSL 탭에서 SSH 터널 사용을 활성화한 다음 아이콘을 클릭하여 SSH 구성 대화 상자를 엽니다. 다음 정보를 입력합니다.

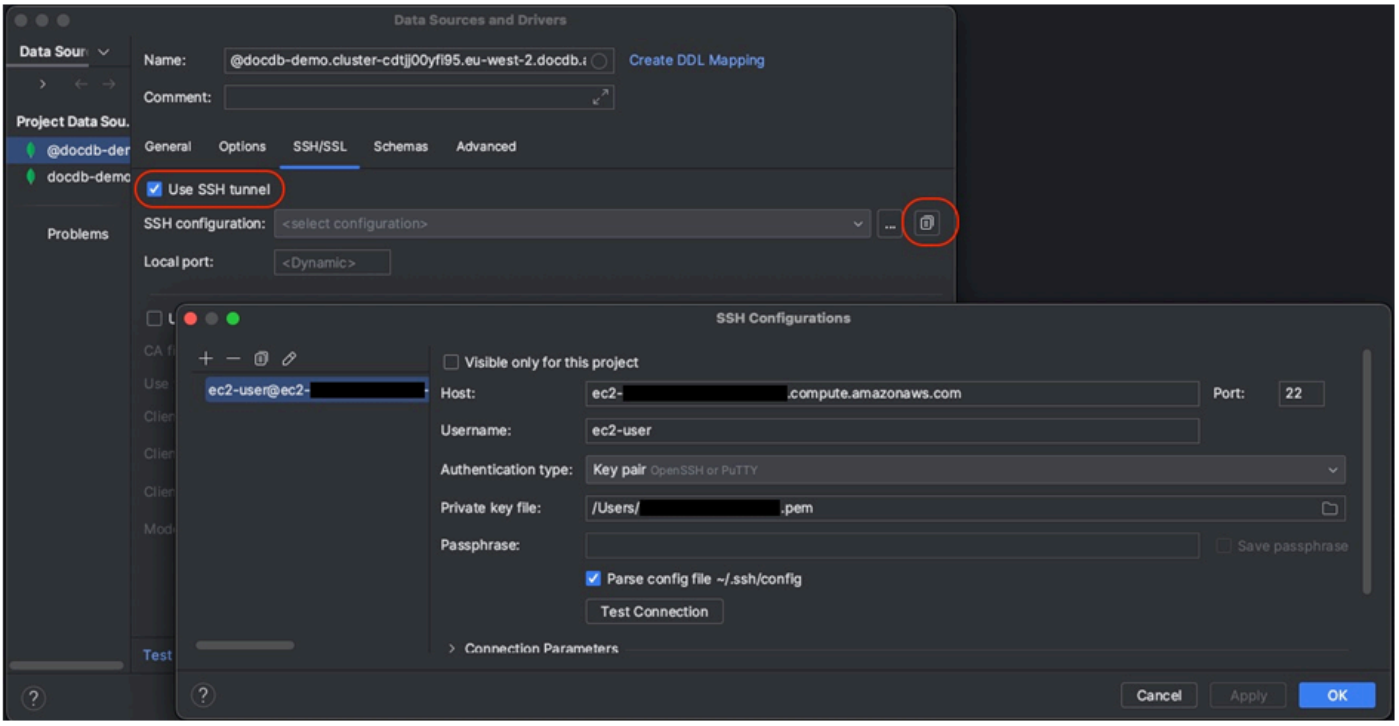
   a. 호스트 필드에 Amazon EC2 인스턴스의 호스트 이름을 입력합니다.

   b. Amazon EC2 인스턴스의 사용자 이름과 암호를 입력합니다.

   c. 인증 유형에서 키 페어를 선택합니다.

   d. 프라이빗 키 파일을 입력합니다.

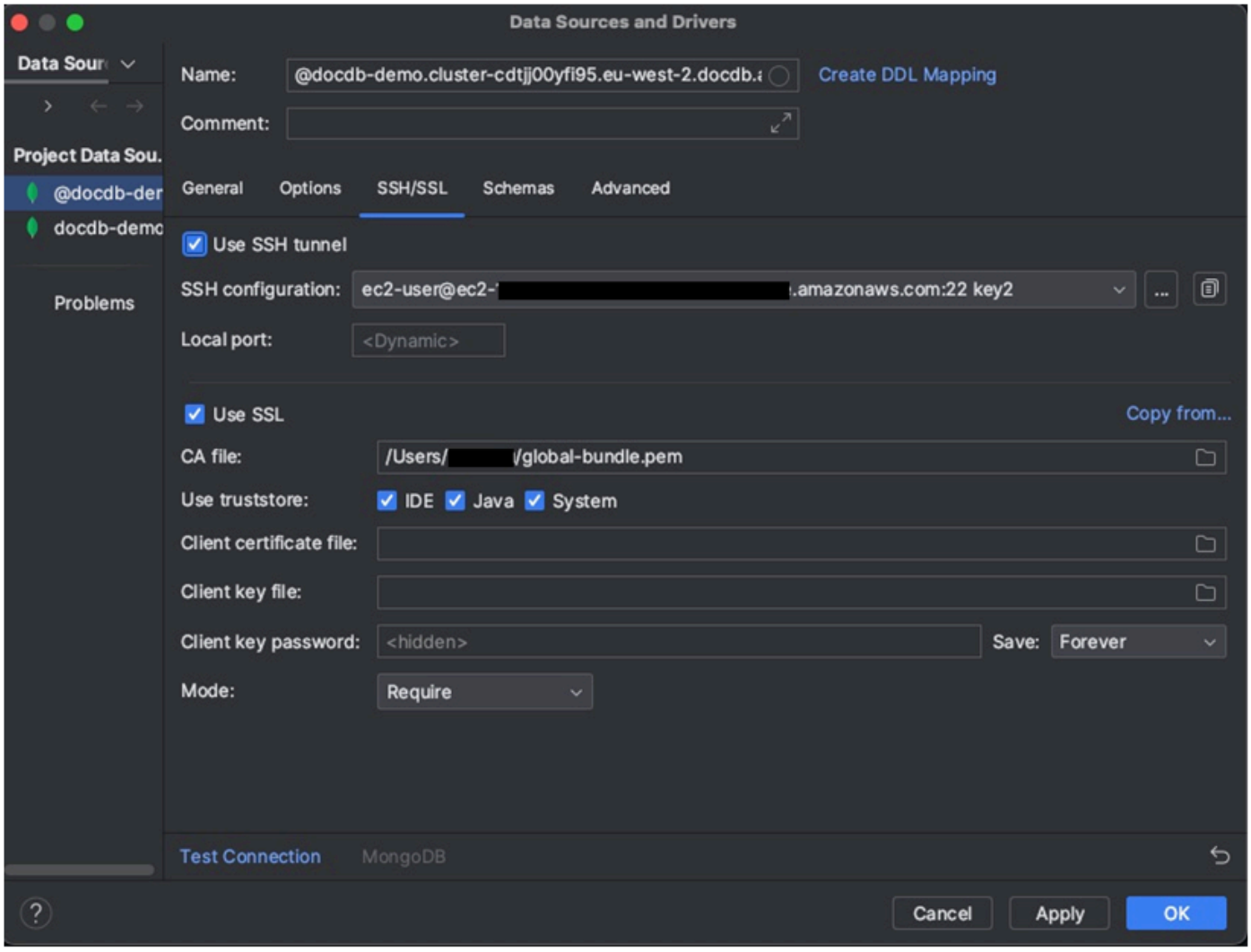
> **Note**
>
> VPN 옵션을 사용하는 경우 SSH 터널을 구성할 필요가 없습니다.



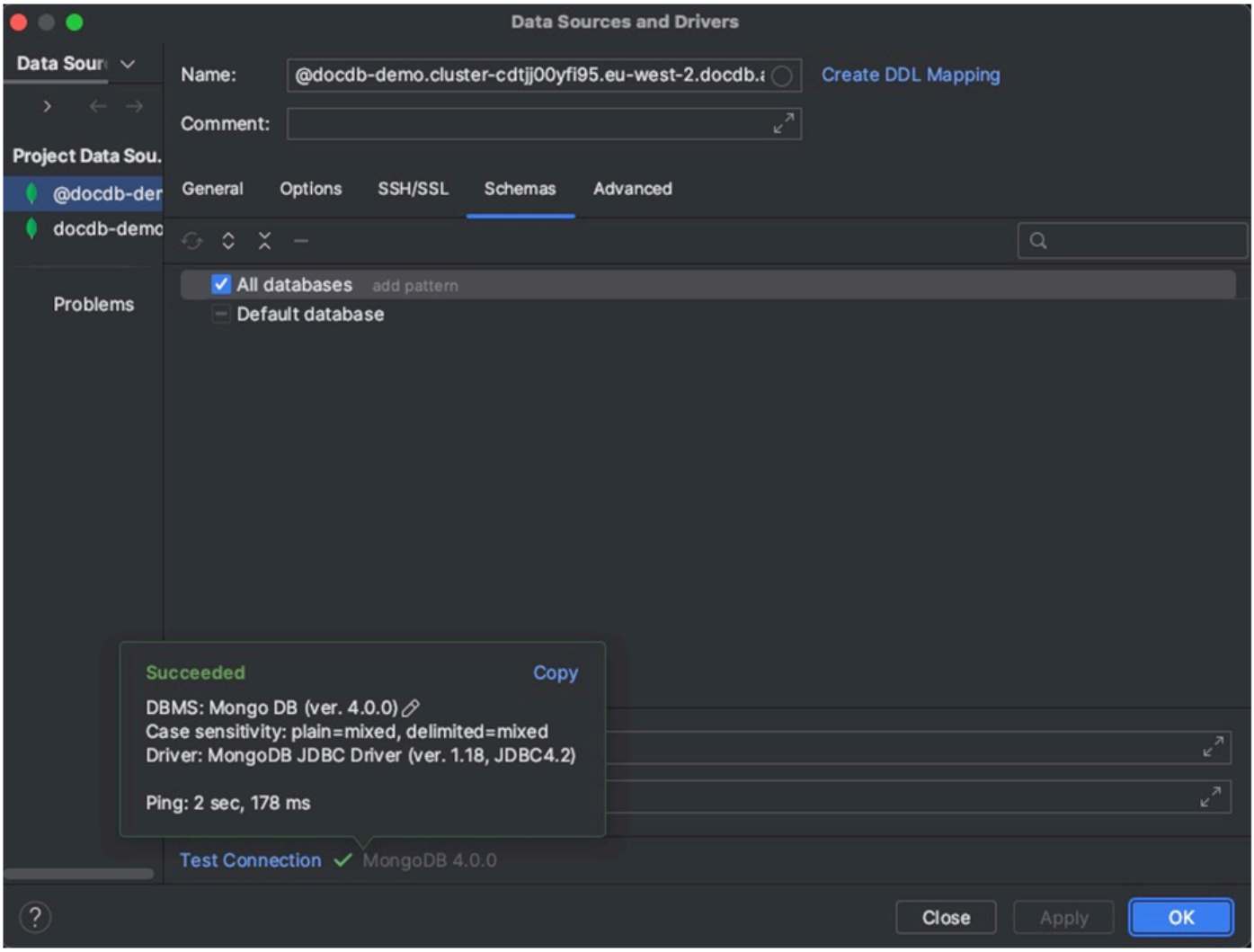
6. SSH/SSL 탭에서 SSL 사용을 활성화합니다. CA 파일 필드에 컴퓨터에 있는 `global-bundle.pem` 파일의 위치를 입력합니다. 모드의 경우 필수 옵션을 그대로 두십시오.

> **Note**
>
> 이 위치에서 또는 wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem 명령을 사용하여 인증서를 다운로드할 수 있습니다.

> **Note**
>
> Amazon DocumentDB 엘라스틱 클러스터에 연결하는 경우 CA 파일을 지정하지 않아도 됩니다. SSL 사용 옵션은 선택된 상태로 두고 다른 모든 옵션은 기본값으로 둡니다.



7. 스키마 탭에서 모든 데이터베이스를 선택하거나 스키마 패턴 필드에 “*: *” 필터를 입력합니다. 연결 테스트 링크를 클릭하여 연결을 테스트합니다.



8. 연결이 성공적으로 테스트되면 확인을 클릭하여 데이터 소스 구성을 저장합니다.

데이터그립 기능

DataGrip은 Amazon DocumentDB를 효율적으로 사용하는 데 도움이 되는 다양한 기능을 제공합니다.

- SQL 편집기 — DataGrip의 SQL 편집기를 사용하여 DocumentDB 컬렉션에서 SQL과 유사한 쿼리를 작성하고 실행할 수 있습니다.
- 비주얼 쿼리 빌더 - 비주얼 쿼리 빌더를 사용하면 SQL 코드를 작성하지 않고도 그래픽 방식으로 쿼리를 만들 수 있습니다.
- 스키마 관리 — 컬렉션 생성, 변경, 삭제 등 데이터베이스 스키마를 쉽게 관리할 수 있습니다.
- 데이터 시각화 — DataGrip에서 제공되는 다양한 시각화 도구를 사용하여 데이터를 보고 분석할 수 있습니다.

- 데이터 내보내기 및 가져오기 — DataGrip의 내보내기 및 가져오기 기능을 사용하여 Amazon DocumentDB와 다른 데이터베이스 간에 데이터를 전송합니다.

Amazon DocumentDB 및 기타 데이터베이스 시스템 사용에 대한 고급 기능 및 팁은 공식 DataGrip 설명서를 참조하십시오.

Amazon EC2를 사용하여 연결

이 섹션에서는 Amazon DocumentDB 클러스터와 Amazon EC2 간의 연결을 설정하고 Amazon EC2 인스턴스에서 Amazon DocumentDB 클러스터에 액세스하는 방법을 설명합니다.

EC2 연결을 구성하기 위한 두 가지 옵션이 있습니다.

- EC2 인스턴스를 Amazon DocumentDB 데이터베이스에 자동 연결 - EC2 콘솔의 자동 연결 기능을 사용하여 EC2 인스턴스와 새 또는 기존 Amazon DocumentDB 데이터베이스 간의 연결을 자동으로 구성합니다. 이 연결을 통해 트래픽이 EC2 인스턴스와 Amazon DocumentDB 데이터베이스 간에 이동할 수 있습니다. 이 옵션은 일반적으로 새 보안 그룹을 테스트하고 생성하는 데 사용됩니다.
- EC2 인스턴스를 Amazon DocumentDB 데이터베이스에 수동으로 연결 - 자동 연결 기능으로 생성된 구성을 재현하도록 보안 그룹을 수동으로 구성하고 할당하여 EC2 인스턴스와 Amazon DocumentDB 데이터베이스 간의 연결을 구성합니다. 이 옵션은 일반적으로 고급 설정을 변경하고 기존 보안 그룹을 사용하는 데 사용됩니다.

사전 조건

옵션과 관계 없이 첫 번째 Amazon DocumentDB 클러스터를 작성하기 전에 다음 작업을 수행해야 합니다:

Amazon Web Services(AWS) 계정 만들기

Amazon DocumentDB 사용을 시작하기 전에 Amazon Web Services(AWS) 계정이 있어야 합니다. AWS 계정은 무료입니다. 사용하는 서비스 및 리소스에 대해서만 비용을 지불하는 것입니다.

이 없는 경우 다음 단계를 AWS 계정완료하여 생성합니다.

에 가입하려면 AWS 계정

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.

에 가입하면 AWS 계정AWS 계정 루트 사용자이 생성됩니다. 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.

선택적으로 필요한 AWS Identity and Access Management (IAM) 권한을 설정합니다.

클러스터, 인스턴스 및 클러스터 파라미터 그룹과 같은 Amazon DocumentDB 리소스를 관리하려면가 요청을 인증하는 데 사용할 AWS 수 있는 자격 증명이 필요합니다. 자세한 내용은 Amazon DocumentDB의 ID 및 액세스 관리 단원을 참조하십시오.

1. 의 검색 창에 IAM을 AWS Management Console입력하고 나타나는 드롭다운 메뉴에서 IAM을 선택합니다.
2. IAM 콘솔에 들어가면 탐색 창에서 사용자를 선택합니다.
3. 사용자 이름을 선택합니다.
4. 권한 추가 버튼을 클릭합니다.
5. 기존 정책 직접 연결을 선택합니다.
6. 검색 창에 `AmazonDocDBFullAccess`을 입력하고 검색 결과에 표시되면 선택합니다.
7. 하단에서 다음: 검토라고 표시된 파란색 버튼을 클릭합니다.
8. 하단에서 권한 추가라고 표시된 파란색 버튼을 클릭합니다.

Amazon Virtual Private Cloud (Amazon VPC) 생성

AWS 리전 현재 위치에 따라 기본 VPC가 이미 생성되어 있거나 생성되어 있지 않을 수 있습니다. 기본 VPC가 없는 경우, Amazon VPC 사용 설명서의 Amazon VPC 시작하기에서 1단계를 완료하세요. 이 작업에는 5분 미만이 소요됩니다.

Amazon EC2 자동 연결

주제

- EC2 인스턴스를 새 Amazon DocumentDB 데이터베이스에 자동으로 연결
- EC2 인스턴스를 기존 Amazon DocumentDB 데이터베이스에 자동 연결

- EC2 인스턴스와의 자동 연결 개요
- 연결된 컴퓨팅 리소스 보기

EC2 인스턴스와 새 Amazon DocumentDB 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 인스턴스와Amazon DocumentDB 데이터베이스 간의 연결에 영향을 미칠 수 있습니다.

> **Note**
>
> AWS Management Console을 사용해야만 EC2 인스턴스와 Amazon DocumentDB 데이터베이스 간의 연결을 자동으로 설정할 수 있습니다. AWS CLI 또는 Amazon DocumentDB API를 사용하여 연결을 자동으로 설정할 수 없습니다.

EC2 인스턴스를 새 Amazon DocumentDB 데이터베이스에 자동으로 연결

다음 프로세스에서는 사전 조건 주제의 단계를 완료했다고 가정합니다.

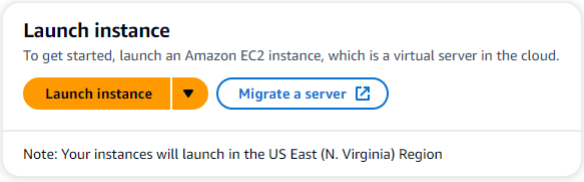
단계

- 1단계: Amazon EC2 인스턴스 생성
- 2단계: Amazon DocumentDB 클러스터 생성
- 3단계: Amazon EC2 인스턴스 구성
- 4단계: MongoDB 쉘 설치
- 5단계: Amazon DocumentDB TLS 관리
- 6단계: Amazon DocumentDB 클러스터에 연결
- 7단계: 데이터 삽입 및 쿼리
- 8단계: 살펴보기

1단계: Amazon EC2 인스턴스 생성

이 단계에서는 나중에 Amazon DocumentDB 클러스터를 프로비저닝하는 데 사용할 동일한 지역 및 Amazon VPC에 Amazon EC2 인스턴스를 생성합니다.

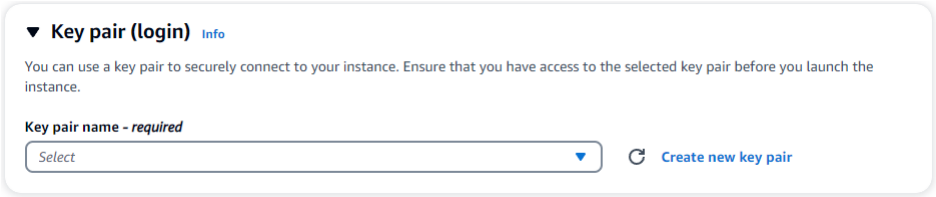
1. Amazon EC2 콘솔 대시보드에서 인스턴스 시작을 선택합니다.



2. 이름 및 태그 섹션에 있는 이름 필드에 이름 또는 식별자를 입력합니다.
3. Amazon Machine Image(AMI) 드롭다운 목록에서 Amazon Linux 2 AMI를 찾아 선택합니다.



4. 인스턴스 유형 드롭다운 목록에서 t3.micro를 찾아 선택합니다.
5. 키 페어(로그인) 섹션에서 기존 키 페어의 식별자를 입력하거나 새 키 페어 생성을 선택합니다.



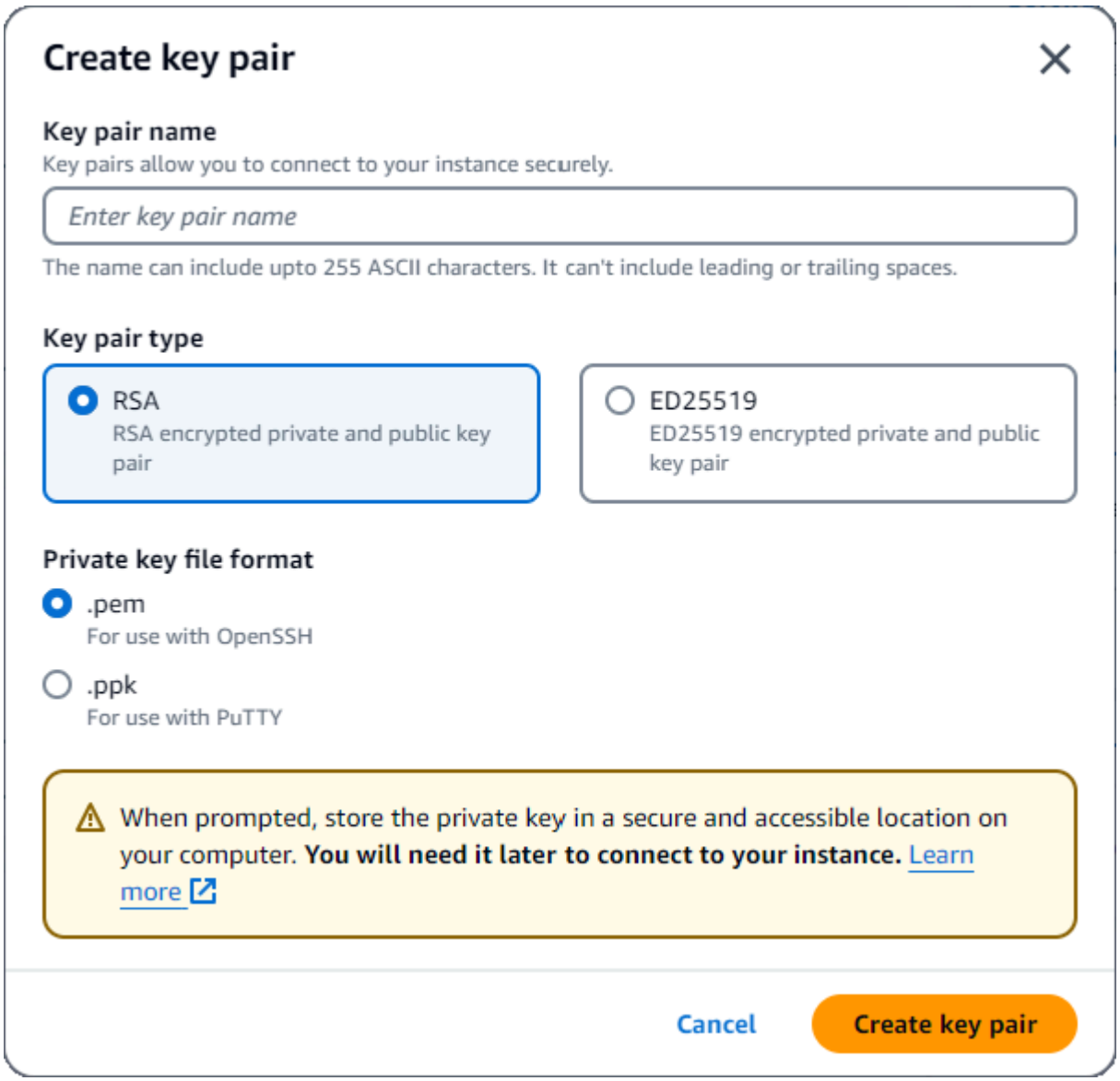
Amazon EC2 키 페어를 제공해야 합니다.

Amazon EC2 키 페어가 있는 경우:

a. 키 페어를 선택하고 목록에서 키 페어를 선택합니다.
b. Amazon EC2 인스턴스에 로그인하려면 프라이빗 키 파일(.pem 또는 .ppk 파일)이 있어야 합니다.

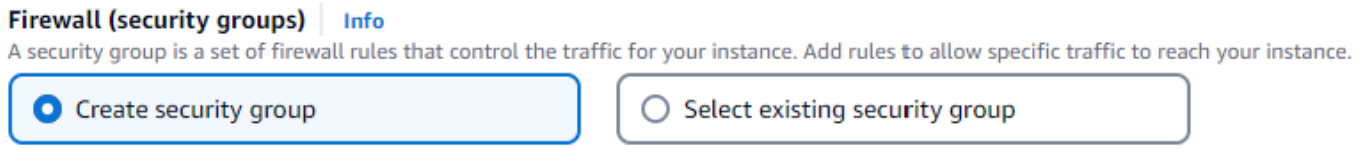
Amazon EC2 키 페어가 없는 경우:

a. 새 키 페어 생성을 선택하면 키 페어 생성 대화 상자가 나타납니다.
b. 키 페어 명칭 필드에 명칭을 입력합니다.
c. 키 페어 유형 및 프라이빗 키 파일 형식을 선택합니다.
d. 키 페어 생성(Create key pair)를 선택합니다.



> **Note**
>
> 보안을 위해 SSH 및 EC2 인스턴스에 대한 인터넷 연결 모두에 키 페어를 사용하는 것이 좋습니다.

6. 선택 사항: 네트워크 설정 섹션의 방화벽(보안 그룹)에서 보안 그룹 생성을 선택합니다.



보안 그룹 생성을 선택합니다(EC2 연결에 적용되는 모든 트래픽 허용 규칙 확인).

> **Note**
>
> 기존 보안 그룹을 사용하려면의 지침을 따르세요Amazon EC2 수동 연결.

7. 요약 섹션에서 EC2 구성을 검토하고 올바른 경우 인스턴스 시작을 선택합니다.

2단계: Amazon DocumentDB 클러스터 생성

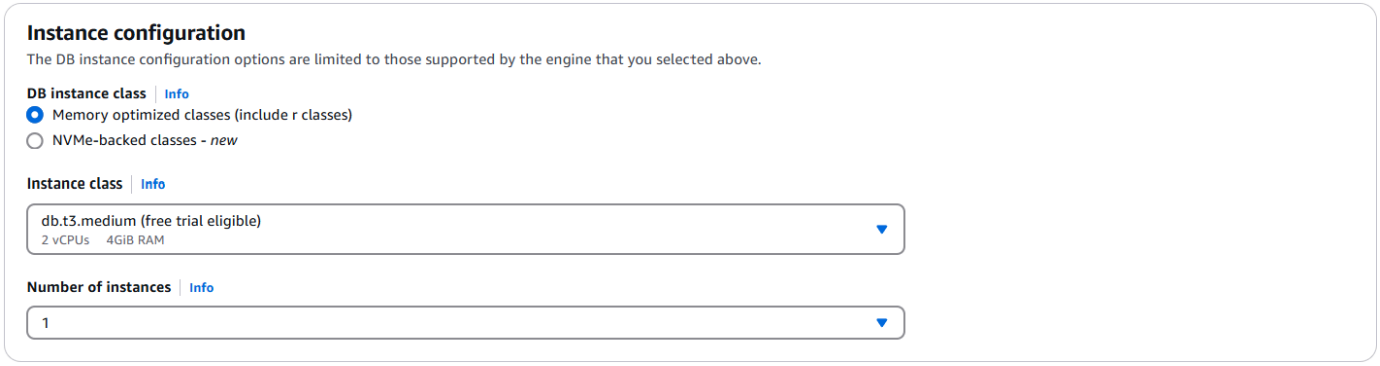
Amazon EC2 인스턴스가 프로비저닝되는 동안 Amazon DocumentDB 클러스터를 생성합니다.

1. Amazon DocumentDB 콘솔로 이동한 다음 탐색 창에서 클러스터를 선택합니다.
2. 생성(Create)을 선택합니다.
3. 클러스터 유형 설정을 인스턴스 기반 클러스터 의 기본값으로 둡니다.
4. 클러스터 구성의 클러스터 식별자에 고유한 이름을 입력합니다. 콘솔은 모든 클러스터 이름을 입력 방식에 관계없이 소문자로 변경합니다.

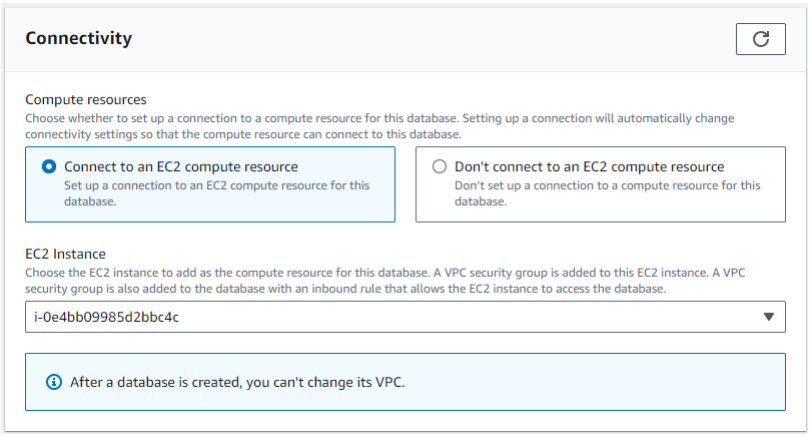
   엔진 버전을 기본값인 5.0.0으로 둡니다.
5. 클러스터 스토리지 구성의 경우 Amazon DocumentDB Standard의 기본 설정을 그대로 둡니다.
6. 인스턴스 구성에서:

   - DB 인스턴스 클래스에서 메모리 최적화 클래스(r 클래스 포함)(기본값)를 선택합니다.

     다른 인스턴스 옵션은 NVMe 지원 클래스입니다. 자세한 내용은 NVMe 지원 인스턴스를 참조하세요.
   - 인스턴스 클래스에서 필요에 맞는 인스턴스 유형을 선택합니다. 인스턴스 클래스에 대한 자세한 설명은 섹션을 참조하세요인스턴스 클래스 사양.
   - 인스턴스 수에서 요구 사항을 가장 잘 반영하는 숫자를 선택합니다. 숫자가 낮을수록 비용이 낮아지고 클러스터에서 관리할 수 있는 읽기/쓰기 볼륨이 낮아집니다.



7. 연결성에서 EC2 컴퓨팅 리소스에 연결을 선택합니다. 이것이 1단계에서 생성한 EC2 인스턴스입니다.



> **Note**
>
> EC2 컴퓨팅 리소스에 연결하면 Amazon DocumentDB 클러스터에 대한 EC2 컴퓨팅 리소스 연결을 위한 보안 그룹이 자동으로 생성됩니다. 클러스터 생성을 완료했고 새로 생성된 보안 그룹을 보려면 클러스터 목록으로 이동하여 클러스터의 식별자를 선택합니다. 연결 및 보안 탭에서 보안 그룹으로 이동하여 보안 그룹 이름(ID)에서 그룹을 찾습니다. 클러스터 이름이 포함되며와 비슷합니다`docdb-ec2-docdb-2023-12-11-21-33-41:i-0e4bb09985d2bbc4c (sg-0238e0b0bf0f73877)`.

8. 인증 섹션에서 기본 사용자의 사용자 이름을 입력한 다음 자체 관리를 선택합니다. 암호를 입력한 다음 확인합니다.

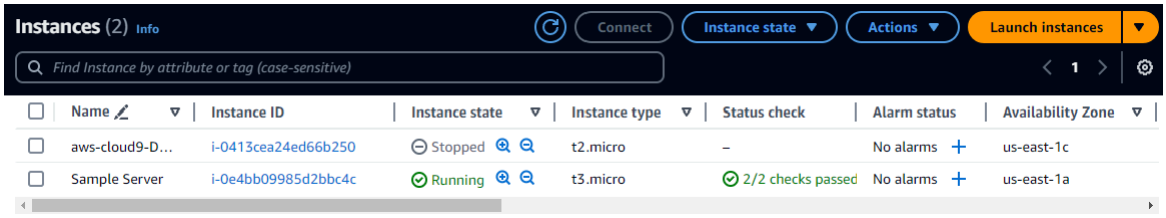
   대신 관리형을 AWS Secrets Manager 선택한 경우 자세한 내용은 Amazon DocumentDB 및를 사용한 암호 관리 AWS Secrets Manager 섹션을 참조하세요.

9. 클러스터 생성을 선택합니다.

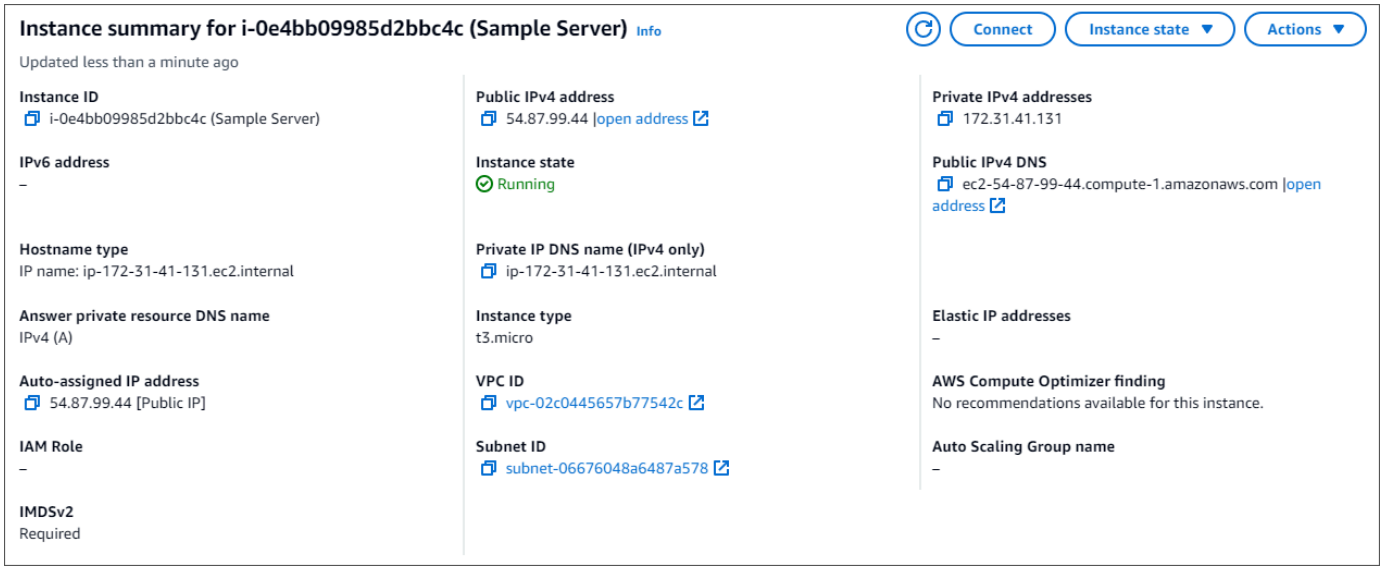
3단계: Amazon EC2 인스턴스 구성

mongo 쉘을 설치하려면 먼저 Amazon EC2 인스턴스에 연결해야 합니다. mongo 쉘을 설치하면 Amazon DocumentDB 클러스터에 연결하고 쿼리할 수 있습니다. 다음 단계를 완료합니다.

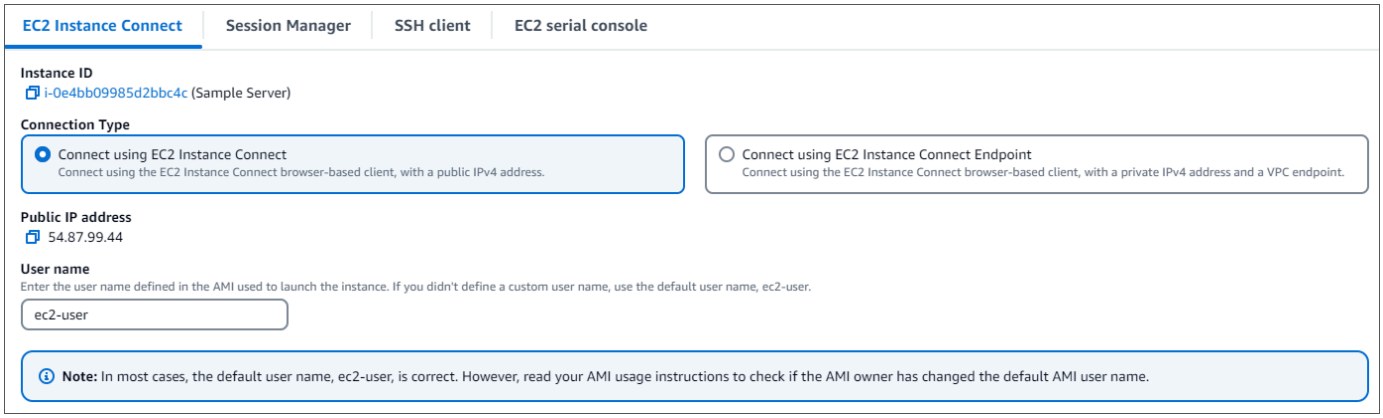
1. Amazon EC2 콘솔에서 인스턴스로 이동하여 방금 생성한 인스턴스가 실행 중인지 확인합니다. 그렇다면 인스턴스 ID를 클릭하여 인스턴스를 선택합니다.



2. 연결을 선택합니다.



3. 연결 방법에는 Amazon EC2 인스턴스 연결, 세션 관리자, SSH 클라이언트 또는 EC2 직렬 콘솔의 4가지 탭 옵션이 표시됩니다. 하나를 선택하고 지침을 따라야 합니다. 완료되면 연결을 선택합니다.



> **Note**
>
> 이 연습을 시작한 후 IP 주소가 변경되었거나 나중에 환경으로 돌아오는 경우 demoEC2 보안 그룹 인바운드 규칙을 업데이트하여 새 API 주소의 인바운드 트래픽을 활성화해야 합니다.

4단계: MongoDB 쉘 설치

이제 Amazon DocumentDB 클러스터를 연결하고 쿼리하는 데 사용하는 명령줄 유틸리티인 MongoDB 셸을 설치할 수 있습니다. Amazon DocumentDB 현재 MongoDB 쉘에는 최신 버전인 mongosh와 이전 버전인 mongo 쉘의 두 가지 버전이 있습니다.

> **Important**
>
> 현재 Amazon DocumentDB에 대한 IAM 자격 증명 인증에서 지원되지 않는 버전 6.13.1 이전의 Node.js 드라이버에는 알려진 제한 사항이 있습니다. Node.js 드라이버를 사용하는 Node.js 드라이버 및 도구(예: mongosh)는 Node.js 드라이버 버전 6.13.1 이상을 사용하도록 업그레이드해야 합니다.

아래 지침에 따라 운영 체제용 MongoDB 쉘을 설치합니다.

On Amazon Linux

Amazon Linux에 MongoDB 쉘을 설치하려면

IAM 인증을 사용하지 않고 최신 MongoDB 셸(mongosh)을 사용하여 Amazon DocumentDB 클러스터에 연결하려면 다음 단계를 따르세요.

1. 리포지토리 파일을 생성합니다. 생성한 EC2 인스턴스의 명령줄에서 다음 명령을 실행합니다.

```
echo -e "[mongodb-org-5.0] \nname=MongoDB Repository\nbaseurl=https://
repo.mongodb.org/yum/amazon/2023/mongodb-org/5.0/x86_64/\ngpgcheck=1 \nenabled=1
 \ngpgkey=https://pgp.mongodb.com/server-5.0.asc" | sudo tee /etc/yum.repos.d/
mongodb-org-5.0.repo
```

2. 완료되면 명령 프롬프트에서 다음 두 명령 옵션 중 하나를 사용하여 mongosh를 설치합니다.

   옵션 1 - Amazon EC2 구성 중에 기본 Amazon Linux 2023을 선택한 경우 다음 명령을 입력합니다.

```
sudo yum install -y mongodb-mongosh-shared-openssl3
```

   옵션 2 - Amazon EC2 구성 중에 Amazon Linux 2를 선택한 경우 다음 명령을 입력합니다.

```
sudo yum install -y mongodb-mongosh
```

IAM 인증을 사용하는 경우 이전 버전의 MongoDB 셸(5.0)을 사용하여 Amazon DocumentDB 클러스터에 연결해야 합니다. 다음 단계를 따르세요.

1. 리포지토리 파일을 생성합니다. 생성한 EC2 인스턴스의 명령줄에서 다음 명령을 실행합니다.

```
echo -e "[mongodb-org-5.0] \nname=MongoDB Repository\nbaseurl=https://
repo.mongodb.org/yum/amazon/2023/mongodb-org/5.0/x86_64/\ngpgcheck=1 \nenabled=1
 \ngpgkey=https://pgp.mongodb.com/server-5.0.asc" | sudo tee /etc/yum.repos.d/
mongodb-org-5.0.repo
```

2. 완료되면 명령 프롬프트에서 다음 명령 옵션을 사용하여 mongodb 5.0 셸을 설치합니다.

```
sudo yum install -y mongodb-org-shell
```

On Ubuntu

Ubuntu에 mongosh를 설치하려면

1. 패키지 관리 시스템에서 사용할 퍼블릭 키를 가져옵니다.

```
curl -fsSL https://pgp.mongodb.com/server-5.0.asc | sudo gpg --dearmor -o /usr/
share/keyrings/mongodb-server-5.0.gpg
```

2. Ubuntu 버전에 맞는 명령을 사용하여 MongoDB의 목록 파일 `mongodb-org-5.0.list`를 생성합니다.

```
echo "deb [ arch=amd64,arm64 signed-by=/usr/share/keyrings/mongodb-
server-5.0.gpg ] https://repo.mongodb.org/apt/ubuntu focal/mongodb-org/5.0
 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-5.0.list
```

3. 다음 명령을 사용하여 로컬 패키지 데이터베이스를 가져오고 업데이트합니다.

```
sudo apt-get update
```

4. mongosh를 설치합니다.

```
sudo apt-get install -y mongodb-mongosh
```

Ubuntu 시스템에서 이전 버전의 MongoDB를 설치하는 방법에 대한 자세한 내용은 Install MongoDB Community Edition on Ubuntu를 참조하세요.

On other operating systems

다른 운영 체제에서 mongo 쉘을 설치하려면 MongoDB 설명서의 MongoDB Community Edition 설치를 참조하세요.

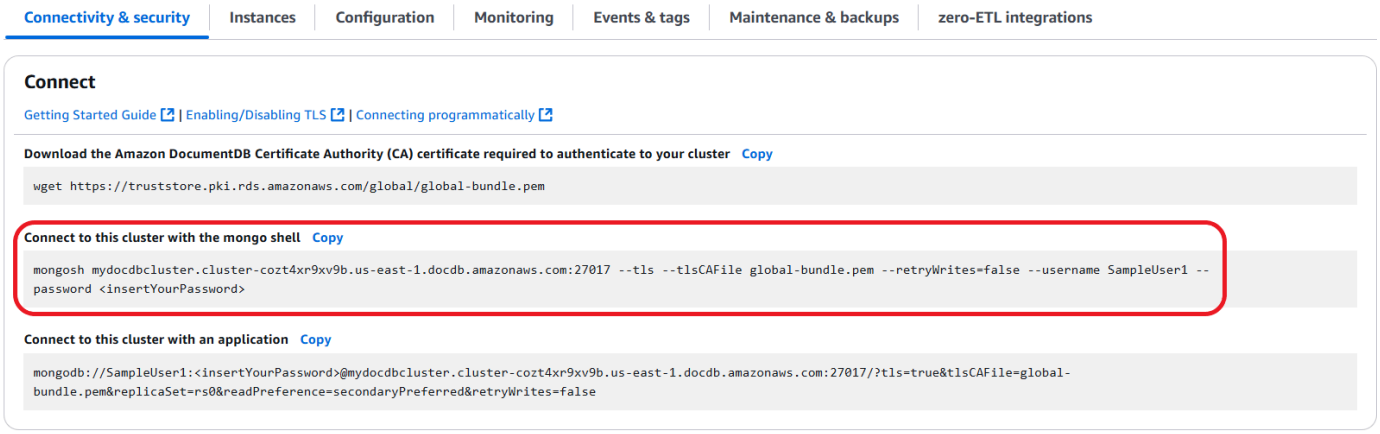
5단계: Amazon DocumentDB TLS 관리

다음 코드를 사용하여 Amazon DocumentDB용 CA 인증서를 다운로드하세요. `wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem`

> **Note**
>
> 전송 계층 보안 (TLS) 은 모든 새 Amazon DocumentDB 클러스터에 기본적으로 활성화됩니다. 자세한 내용은 Amazon DocumentDB 클러스터 TLS 설정을 참조하세요.

6단계: Amazon DocumentDB 클러스터에 연결

1. Amazon DocumentDB 콘솔의 클러스터에서 클러스터를 찾습니다. 해당 클러스터의 클러스터 식별자를 클릭하여 생성한 클러스터를 선택합니다.
2. 연결 및 보안 탭에서 연결 상자의 mongo 쉘을 사용하여 이 클러스터에 연결을 찾습니다.



   제공된 연결 문자열을 복사하여 터미널에 붙여넣습니다.

   해당 항목을 다음과 같이 변경합니다.

   a. 문자열에 올바른 사용자 이름이 있는지 확인합니다.
   b. 그런 다음 연결할 때 mongo 쉘에서 비밀번호를 입력하라는 메시지가 표시되도록 `<insertYourPassword>` 항목을 생략합니다.
   c. 선택 사항: IAM 인증을 사용하거나 이전 버전의 MongoDB 셸을 사용하는 경우 다음과 같이 연결 문자열을 수정합니다.

      ```
      mongo --ssl --host mydocdbcluster.cluster-cozt4xr9xv9b.us-east-1.docdb.amazonaws.com:27017 --sslCAFile global-bundle.pem --username SampleUser1 --password
      ```

      `mydocdbcluster.cluster-cozt4xr9xv9b.us-east-1`를 클러스터의 동일한 정보로 바꿉니다.

3. 이제 터미널에서 Enter를 누릅니다. 이제 비밀번호를 입력하라는 메시지가 표시됩니다. 암호를 입력합니다.
4. 암호를 입력하고 `rs0 [direct: primary] <env-name>>` 메시지가 표시되면 Amazon DocumentDB 클러스터에 성공적으로 연결된 것입니다.

연결에 문제가 있으신가요? Amazon DocumentDB 문제 해결을 참조하세요.

7단계: 데이터 삽입 및 쿼리

이제 클러스터에 연결되었으므로 몇 가지 쿼리를 실행하여 문서 데이터베이스 사용에 익숙해질 수 있습니다.

1. 단일 문서를 삽입하려면 다음을 입력합니다:

```
db.collection.insertOne({"hello":"DocumentDB"})
```

출력은 다음과 같습니다:

```
{
  acknowledged: true,
  insertedId: ObjectId('673657216bdf6258466b128c')
}
```

2. `findOne()` 명령으로 작성한 문서를 읽을 수 있습니다 (단일 문서만 반환하기 때문). 다음 내용을 입력합니다.

```
db.collection.findOne()
```

다음과 같은 출력내용을 얻게 됩니다.

```
{ "_id" : ObjectId("5e401fe56056fda7321fbd67"), "hello" : "DocumentDB" }
```

3. 쿼리를 몇 개 더 수행하려면 게임 프로필 사용 사례를 고려해 보세요. 먼저 `profiles`이라는 제목의 컬렉션에 몇 개의 항목을 삽입합니다. 다음 내용을 입력합니다.

```
db.profiles.insertMany([{ _id: 1, name: 'Matt', status: 'active', level: 12, score:
 202 },
      { _id: 2, name: 'Frank', status: 'inactive', level: 2, score: 9 },
      { _id: 3, name: 'Karen', status: 'active', level: 7, score: 87 },
      { _id: 4, name: 'Katie', status: 'active', level: 3, score: 27 }
])
```

다음과 같은 출력내용을 얻게 됩니다.

```
{ acknowledged: true, insertedIds: { '0': 1, '1': 2, '2': 3, '3': 4 } }
```

4. `find()` 명령을 사용하여 프로필 컬렉션의 모든 문서를 반환합니다. 다음 내용을 입력합니다.

```
db.profiles.find()
```

3단계에서 입력한 데이터와 일치하는 출력을 받게 됩니다.

5. 필터를 사용하여 단일 문서에 대한 쿼리를 사용하세요. 다음을 입력합니다:

```
db.profiles.find({name: "Katie"})
```

다음과 같은 출력내용을 얻게 됩니다.

```
{ "_id" : 4, "name" : "Katie", "status": "active", "level": 3, "score":27}
```

6. 이제 프로필을 찾아 `findAndModify` 명령을 사용하여 해당 프로필을 수정해 보겠습니다. 다음 코드를 사용하여 Matt 사용자에게 10포인트를 추가로 제공합니다.

```
db.profiles.findAndModify({
        query: { name: "Matt", status: "active"},
        update: { $inc: { score: 10 } }
    })
```

다음과 같은 결과가 출력됩니다 (참고로 그의 점수는 아직 오르지 않았습니다):

```
{
    [{_id : 1, name : 'Matt', status: 'active', level: 12, score: 202}]
```

7. 다음 쿼리를 통해 그의 점수가 변경되었는지 확인할 수 있습니다:

```
db.profiles.find({name: "Matt"})
```

출력은 다음과 같습니다.

```
{ "_id" : 1, "name" : "Matt", "status" : "active", "level" : 12, "score" : 212 }
```

8단계: 살펴보기

축하합니다! Amazon DocumentDB에 대한 퀵 스타트 가이드를 성공적으로 완료했습니다.

다음 단계? 몇 가지 인기 있는 기능을 갖춘 이 강력한 데이터베이스를 최대한 활용하는 방법을 알아보세요.

- Amazon DocumentDB 관리
- 스케일링
- 백업 및 복원

> **Note**
>
> 비용을 절감하려면 Amazon DocumentDB 클러스터를 중지하여 비용을 절감하거나 클러스터를 삭제할 수 있습니다. 기본적으로 30분 동안 사용하지 않으면 AWS Cloud9 환경이 기본 Amazon EC2 인스턴스를 중지합니다.

EC2 인스턴스를 기존 Amazon DocumentDB 데이터베이스에 자동 연결

다음 절차에서는 기존 Amazon DocumentDB 클러스터와 기존 Amazon EC2 인스턴스가 있다고 가정합니다.

Amazon DocumentDB 클러스터에 액세스하고 Amazon EC2 연결을 설정합니다.

1. Amazon DocumentDB 클러스터에 액세스합니다.

   a. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.

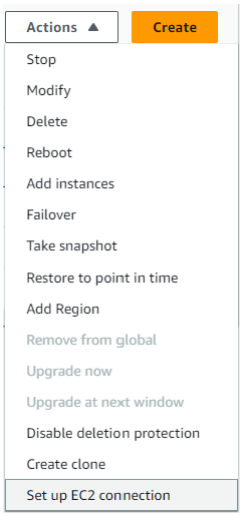
   b. 탐색 창에서 클러스터를 선택합니다.

   > **Tip**
   >
   > 화면 왼쪽에 탐색 창이 표시되지 않으면 페이지 왼쪽 상단 모서리에서 메뉴 아이콘 (≡) 을 선택합니다.

   c. 클러스터 이름 왼쪽에 있는 버튼을 선택하여 원하는 클러스터를 지정합니다.

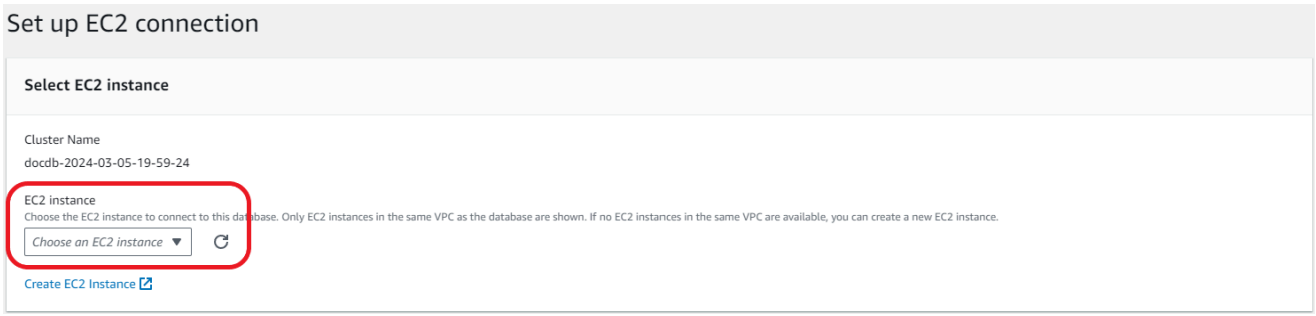
2. Amazon EC2 연결을 설정합니다.

   a. 작업을 선택하고 EC2 연결 설정을 선택합니다.



EC2 연결 설정 대화가 나타납니다.

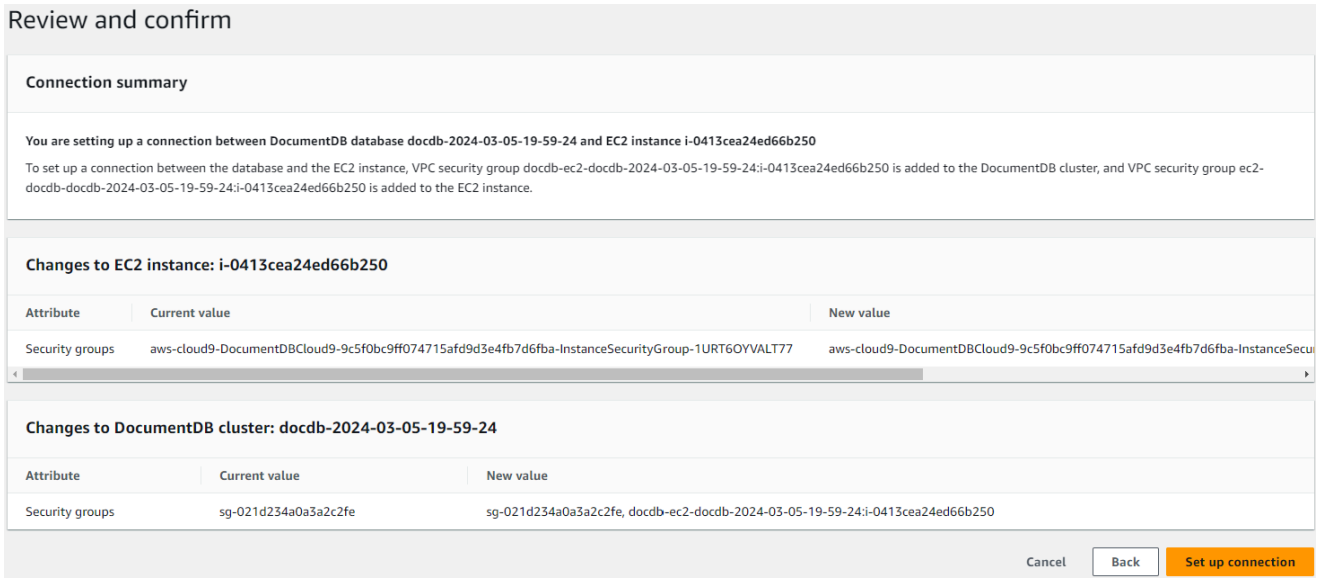
b. EC2 인스턴스 필드에서 클러스터에 연결하려는 EC2 인스턴스를 선택합니다.



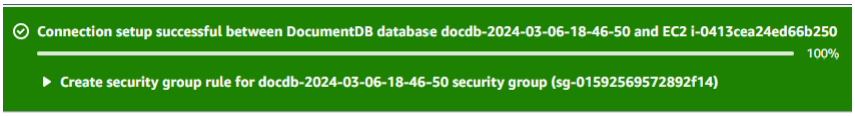
c. 계속을 선택합니다.

검토 및 확인 대화가 나타납니다.

d. 변경 사항이 올바른지 확인합니다. 그런 다음 연결 설정을 선택합니다.



성공하면 다음 확인 메시지가 나타납니다.



EC2 인스턴스와의 자동 연결 개요

EC2 인스턴스와 Amazon DocumentDB 데이터베이스 간의 연결을 설정하는 경우 Amazon DocumentDB는 EC2 인스턴스 및 Amazon DocumentDB 데이터베이스에 대한 VPC 보안 그룹을 자동으로 구성합니다.

다음은 EC2 인스턴스를 Amazon DocumentDB 데이터베이스와 연결하기 위한 요구 사항입니다.

- EC2 인스턴스는 Amazon DocumentDB 데이터베이스와 동일한 VPC가 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 Amazon DocumentDB 데이터베이스 및 EC2 인스턴스와 연결된 보안 그룹의 현재 구성을 기반으로 조치를 취합니다.

현재 Amazon DocumentDB 보안 그룹 구성	현재 EC2 보안 그룹 구성	Amazon DocumentDB 작업
이름이 `DocumentDB-ec2-n` 패턴과 일치하며 Amazon DocumentDB 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.	`DocumentDB-ec2-n` (n은 숫자를 나타냄) 패턴과 일치하는 이름을 가진 EC2 인스턴스와 연결된 보안 그룹이 하나 이상 있습니다. 해당 패턴과 일치하는 보안 그룹이 수정되지 않았습니다. 이 보안 그룹에는 Amazon DocumentDB 데이터베이스의 VPC 보안 그룹을 원본으로 하는 아웃바운드 규칙이 하나만 포함됩니다.	Amazon DocumentDB는 아무런 조치도 취하지 않습니다. EC2 인스턴스와 Amazon DocumentDB 데이터베이스 간의 연결이 이미 자동으로 구성되었습니다. EC2 인스턴스와 Amazon DocumentDB 데이터베이스 사이에 이미 연결이 존재하기 때문에 보안 그룹은 수정되지 않습니다.
다음 중 하나의 조건이 적용됩니다. • `DocumentDB-ec2-n` 패턴과 일치하는 이름을 가진 Amazon DocumentDB 데이터베이스와 연결된 보안 그룹이 없습니다. • 이름이 `DocumentDB-ec2-n` 패턴과 일치하며 Amazon DocumentDB와 연결된 보안 그룹이 하나 이상	다음 중 하나의 조건이 적용됩니다. • `ec2-DocumentDB-n` 패턴과 일치하는 이름을 가진 EC2 인스턴스와 연결된 보안 그룹이 없습니다. • `ec2-DocumentDB-n` 패턴과 일치하는 이름을 가진 EC2 인스턴스와 연결된 보안 그룹이 하나 이상 있습니다. 그러나 Amazon	Amazon DocumentDB 작업: 새 보안 그룹 생성

현재 Amazon DocumentDB 보안 그룹 구성	현재 EC2 보안 그룹 구성	Amazon DocumentDB 작업
있습니다. 그러나 Amazon DocumentDB는 EC2 인스턴스와의 연결에 이러한 보안 그룹을 사용할 수 없습니다. Amazon DocumentDB는 EC2 인스턴스의 VPC 보안 그룹을 소스로 하는 인바운드 규칙 하나가 포함되지 않은 보안 그룹을 사용할 수 없습니다. Amazon DocumentDB는 또한 수정된 보안 그룹을 사용할 수 없습니다. 수정 사항의 예로는 규칙 추가 또는 기존 규칙의 포트 변경이 있습니다.	DocumentDB는 Amazon DocumentDB 데이터베이스와의 연결에 이러한 보안 그룹을 사용할 수 없습니다. Amazon DocumentDB는 Amazon DocumentDB 데이터베이스의 VPC 보안 그룹을 원본으로 하는 아웃바운드 규칙 하나가 포함되지 않은 보안 그룹을 사용할 수 없습니다. Amazon DocumentDB는 또한 수정된 보안 그룹을 사용할 수 없습니다.	
이름이 `DocumentDB-ec2-n` 패턴과 일치하며 Amazon DocumentDB 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-DocumentDB-n` 패턴과 일치하는 이름을 가진 EC2 인스턴스와 연결된 보안 그룹이 하나 이상 있습니다. 그러나 Amazon DocumentDB는 Amazon DocumentDB 데이터베이스와의 연결에 이러한 보안 그룹을 사용할 수 없습니다. Amazon DocumentDB는 Amazon DocumentDB 데이터베이스의 VPC 보안 그룹을 원본으로 하는 아웃바운드 규칙 하나가 포함되지 않은 보안 그룹을 사용할 수 없습니다. Amazon DocumentDB는 또한 수정된 보안 그룹을 사용할 수 없습니다.	Amazon DocumentDB 작업: 새 보안 그룹 생성

현재 Amazon DocumentDB 보안 그룹 구성	현재 EC2 보안 그룹 구성	Amazon DocumentDB 작업
이름이 `DocumentDB-ec2-n` 패턴과 일치하며 Amazon DocumentDB 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 `DocumentDB-ec2-n` 패턴과 일치합니다. 수정되지 않았습니다. Amazon DocumentDB 데이터베이스의 VPC 보안 그룹을 원본으로 하는 아웃바운드 규칙이 하나만 포함됩니다.	Amazon DocumentDB 작업: EC2 보안 그룹 연결

현재 Amazon DocumentDB 보안 그룹 구성	현재 EC2 보안 그룹 구성	Amazon DocumentDB 작업
다음 중 하나의 조건이 적용됩니다. • `DocumentDB-ec2-n` 패턴과 일치하는 이름을 가진 Amazon DocumentDB 데이터베이스와 연결된 보안 그룹이 없습니다. • 이름이 `DocumentDB-ec2-n` 패턴과 일치하며 Amazon DocumentDB 데이터베이스와 연결된 보안 그룹이 하나 이상 있습니다. 그러나 Amazon DocumentDB는 EC2 인스턴스와의 연결에 이러한 보안 그룹을 사용할 수 없습니다. Amazon DocumentDB는 EC2 인스턴스의 VPC 보안 그룹을 소스로 하는 인바운드 규칙 하나가 포함되지 않은 보안 그룹을 사용할 수 없습니다. 또한 Amazon DocumentDB는 수정된 보안 그룹을 사용할 수 없습니다.	`DocumentDB-ec2-n` 패턴과 일치하는 이름을 가진 EC2 인스턴스와 연결된 보안 그룹이 하나 이상 있습니다. 해당 패턴과 일치하는 보안 그룹이 수정되지 않았습니다. 이 보안 그룹에는 Amazon DocumentDB 데이터베이스의 VPC 보안 그룹을 원본으로 하는 아웃바운드 규칙이 하나만 포함됩니다.	Amazon DocumentDB 작업: 새 보안 그룹 생성

Amazon DocumentDB 작업: 새 보안 그룹 생성

Amazon DocumentDB에서 다음 작업을 수행합니다.

- `DocumentDB-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 Amazon DocumentDB 데이터베이스와 연결되어 있으며 EC2 인스턴스가 Amazon DocumentDB 데이터베이스에 액세스하도록 허용합니다.

- `ec2-DocumentDB-n` 패턴과 일치하는 새 보안 그룹을 생성합니다. 이 보안 그룹에는 Amazon DocumentDB 데이터베이스의 VPC 보안 그룹을 원본으로 하는 아웃바운드 규칙이 있습니다. 이 보안 그룹은 EC2 인스턴스와 연결되어 있으며 EC2 인스턴스가 Amazon DocumentDB 데이터베이스에 트래픽을 보내도록 허용합니다.

Amazon DocumentDB 작업: EC2 보안 그룹 연결

Amazon DocumentDB는 유효한 기존 EC2 보안 그룹을 EC2 인스턴스와 연결합니다. 이 보안 그룹은 EC2 인스턴스가 Amazon DocumentDB 데이터베이스에 트래픽을 보내도록 허용합니다.

연결된 컴퓨팅 리소스 보기

AWS Management Console 를 사용하여 Amazon DocumentDB 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 DocumentDB 클러스터 생성 및 다중 AZ DB 클러스터 생성을 참조하세요.
- 기존 데이터베이스와 컴퓨팅 리소스 간의 연결을 설정할 수 있습니다. 자세한 내용은 Amazon EC2 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 `ec2-DocumentDB-n` 패턴(n이 숫자)과 일치합니다.
- 컴퓨팅 리소스와 연결된 보안 그룹에는 포트 범위가 Amazon DocumentDB 데이터베이스에서 사용하는 포트로 설정된 아웃바운드 규칙이 있습니다.
- 컴퓨팅 리소스와 연결된 보안 그룹에는 소스가 Amazon DocumentDB 데이터베이스에 연결된 보안 그룹으로 설정된 아웃바운드 규칙이 있습니다.
- Amazon DocumentDB 데이터베이스와 연결된 보안 그룹의 이름은 `DocumentDB-ec2-n` 패턴(n이 숫자)과 일치합니다.
- Amazon DocumentDB 데이터베이스와 연결된 보안 그룹에는 포트 범위가 Amazon DocumentDB 데이터베이스에서 사용하는 포트로 설정된 인바운드 규칙이 있습니다.

- Amazon DocumentDB 데이터베이스와 연결된 보안 그룹에는 원본이 컴퓨팅 리소스에 연결된 보안 그룹으로 설정된 인바운드 규칙이 있습니다.

Amazon DocumentDB 데이터베이스에 연결된 컴퓨팅 리소스 보기

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/docdb Amazon DocumentDB 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택한 다음 Amazon DocumentDB 데이터베이스의 이름을 선택합니다.
3. 연결 및 보안 탭의 연결된 컴퓨팅 리소스 섹션에서 컴퓨팅 리소스를 확인합니다.

Amazon EC2 수동 연결

주제

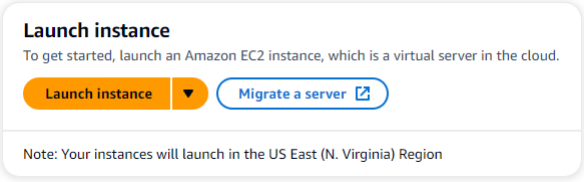
- 1단계: Amazon EC2 인스턴스 생성
- 2단계: 보안 그룹 만들기
- 3단계: Amazon DocumentDB 클러스터 생성
- 4단계: Amazon EC2 인스턴스 구성
- 5단계: MongoDB 쉘 설치
- 6단계: Amazon DocumentDB TLS 관리
- 7단계: Amazon DocumentDB 클러스터에 연결
- 8단계: 데이터 삽입 및 쿼리
- 9단계: 살펴보기

다음 단계에서는 사전 조건 주제의 단계를 완료했다고 가정합니다.

1단계: Amazon EC2 인스턴스 생성

이 단계에서는 나중에 Amazon DocumentDB 클러스터를 프로비저닝하는 데 사용할 동일한 지역 및 Amazon VPC에 Amazon EC2 인스턴스를 생성합니다.

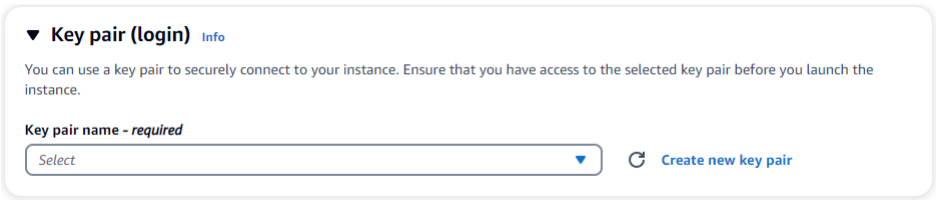
1. Amazon EC2 콘솔 대시보드에서 인스턴스 시작을 선택합니다.



2. 이름 및 태그 섹션에 있는 이름 필드에 이름 또는 식별자를 입력합니다.
3. Amazon Machine Image(AMI) 드롭다운 목록에서 Amazon Linux 2 AMI를 찾아 선택합니다.



4. 인스턴스 유형 드롭다운 목록에서 t3.micro를 찾아 선택합니다.
5. 키 페어(로그인) 섹션에서 기존 키 페어의 식별자를 입력하거나 새 키 페어 생성을 선택합니다.



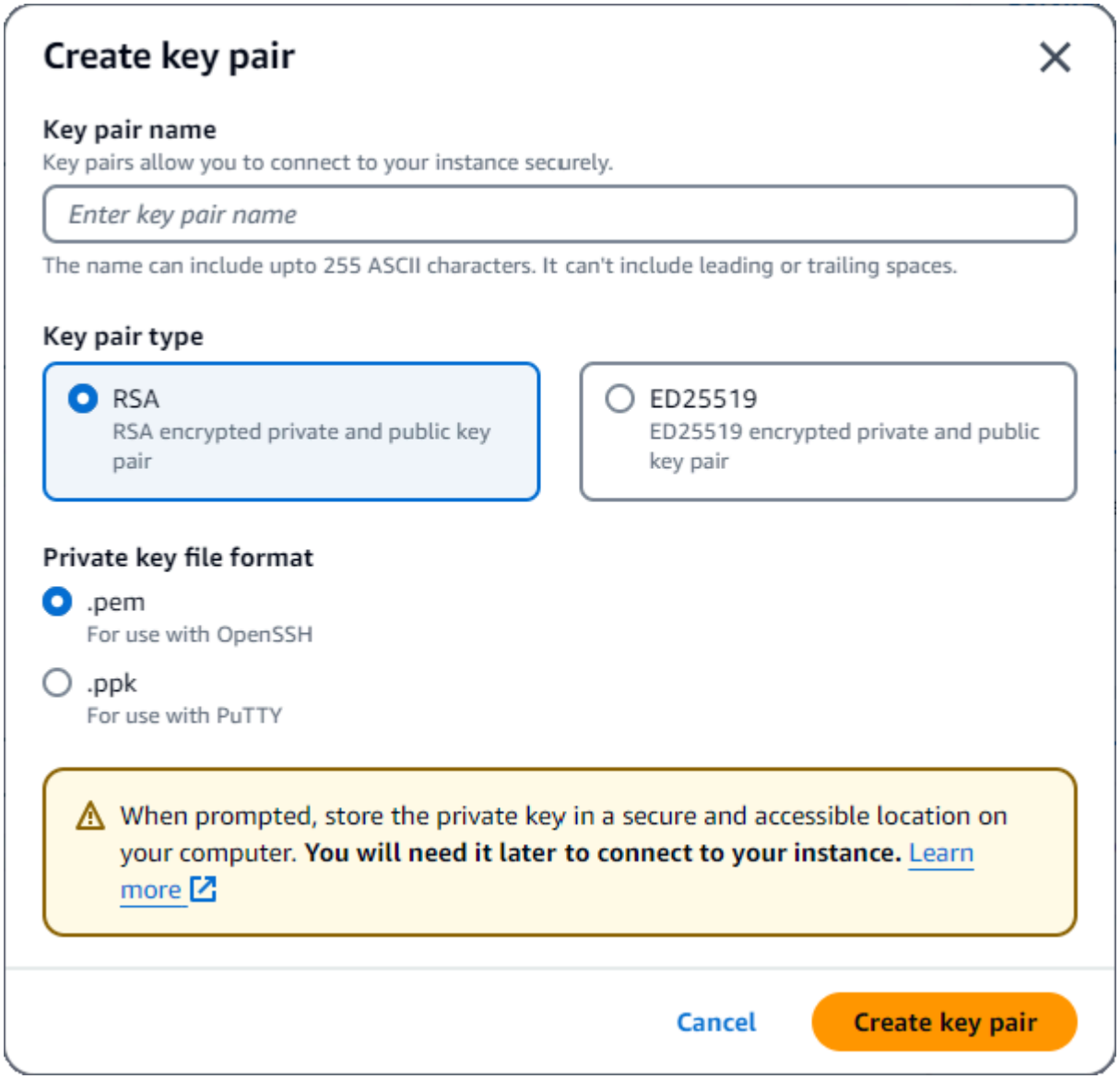
Amazon EC2 키 페어를 제공해야 합니다.

Amazon EC2 키 페어가 있는 경우:

a. 키 페어를 선택하고 목록에서 키 페어를 선택합니다.
b. Amazon EC2 인스턴스에 로그인하려면 프라이빗 키 파일(.pem 또는 .ppk 파일)이 있어야 합니다.

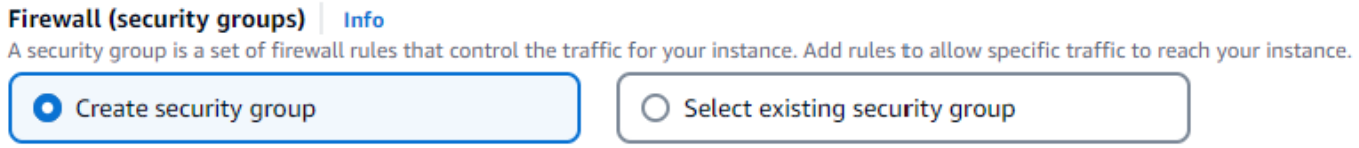
Amazon EC2 키 페어가 없는 경우:

a. 새 키 페어 생성을 선택하면 키 페어 생성 대화 상자가 나타납니다.
b. 키 페어 명칭 필드에 명칭을 입력합니다.
c. 키 페어 유형 및 프라이빗 키 파일 형식을 선택합니다.
d. 키 페어 생성(Create key pair)를 선택합니다.



> **Note**
>
> 보안을 위해 SSH 및 EC2 인스턴스에 대한 인터넷 연결 모두에 키 페어를 사용하는 것이 좋습니다.

6. 네트워크 설정 섹션의 방화벽(보안 그룹)에서 보안 그룹 생성 또는 기존 보안 그룹 선택을 선택합니다.



기존 보안 그룹을 선택하기로 선택한 경우 일반 보안 그룹 드롭다운 목록에서 하나를 선택합니다.

새 보안 그룹을 생성하도록 선택한 경우 다음을 수행합니다.

a. EC2 연결에 적용되는 모든 트래픽 허용 규칙을 확인합니다.

b. IP 필드에서 내 IP를 선택하거나 사용자 지정을 선택하여 CIDR 블록, 접두사 목록 또는 보안 그룹 목록에서 선택합니다. EC2 인스턴스는 EC2 인스턴스에 대한 IP 주소 액세스를 허용하므로 EC2 인스턴스가 격리된 네트워크에 있지 않는 한 Anywhere를 선택하지 않는 것이 좋습니다.

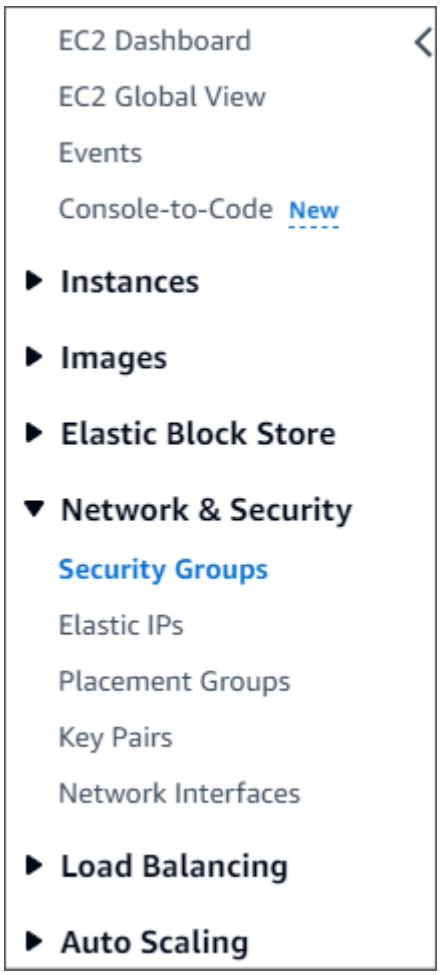


7. 요약 섹션에서 EC2 구성을 검토하고 올바른 경우 인스턴스 시작을 선택합니다.

2단계: 보안 그룹 만들기

이제 기본 Amazon VPC에 새 보안 그룹을 생성합니다. 보안 그룹을 demoDocDB 사용하면 Amazon EC2 인스턴스에서 포트 27017 (Amazon DocumentDB의 기본 포트) 을 통해 Amazon DocumentDB 클러스터에 연결할 수 있습니다.

1. Amazon EC2 관리 콘솔의 네트워크 및 보안에서 보안 그룹을 선택합니다.



2. 보안 그룹 생성을 선택합니다.



3. 기본 세부 정보 섹션에서 다음을 수행합니다.

   a. 보안 그룹 이름에 `demoDocDB`를 입력합니다.

   b. 설명에 설명을 입력합니다.

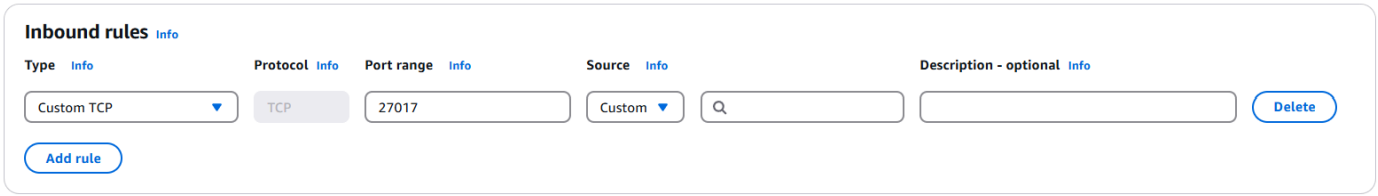
   c. VPC의 경우 기본 VPC 사용을 수락합니다.

4. 인바운드 규칙 섹션에서 규칙 추가를 선택합니다.

   a. 유형에서 사용자 지정 TCP 규칙(기본값)을 선택합니다.

   b. 포트 범위에 `27017`를 입력합니다.

   c. 소스(Source)에서 사용자 지정(Custom)을 선택합니다. 옆에 있는 필드에서 1단계에서 방금 생성한 보안 그룹을 검색합니다. Amazon EC2 콘솔에서 소스 이름을 자동으로 채우려면 브라우저를 새로 고쳐야 할 수 있습니다.



5. 다른 모든 기본값을 그대로 사용하고 보안 그룹 생성을 선택합니다.



3단계: Amazon DocumentDB 클러스터 생성

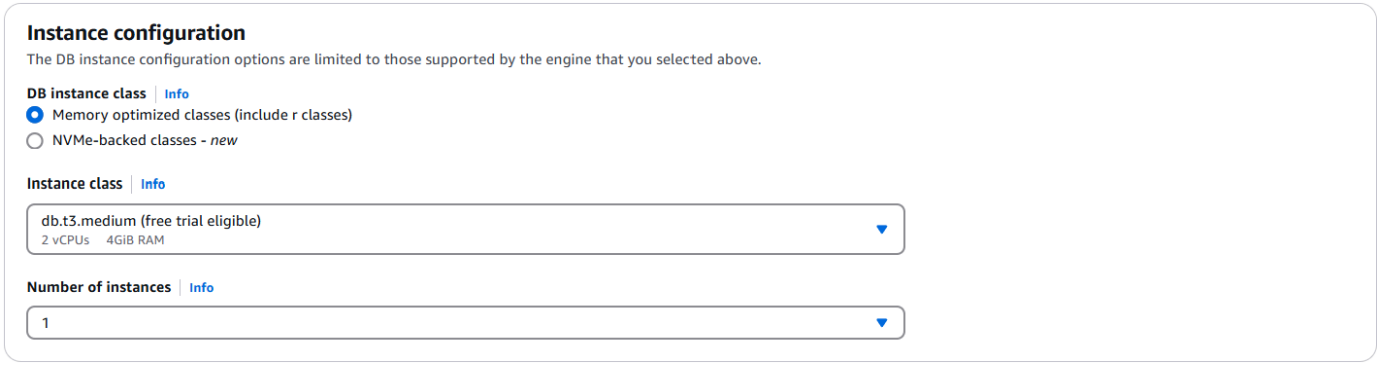
Amazon EC2 인스턴스가 프로비저닝되는 동안 Amazon DocumentDB 클러스터를 생성해야 합니다.

1. Amazon DocumentDB 콘솔로 이동한 다음 탐색 창에서 클러스터를 선택합니다.
2. 생성(Create)을 선택합니다.
3. 클러스터 유형 설정을 인스턴스 기반 클러스터 의 기본값으로 둡니다.
4. 클러스터 구성의 클러스터 식별자에 고유한 이름을 입력합니다. 콘솔은 모든 클러스터 이름을 입력 방식에 관계없이 소문자로 변경합니다.

   엔진 버전을 기본값인 5.0.0으로 둡니다.
5. 클러스터 스토리지 구성의 경우 Amazon DocumentDB Standard의 기본 설정을 그대로 둡니다.
6. 인스턴스 구성에서:

   - DB 인스턴스 클래스에서 메모리 최적화 클래스(r 클래스 포함)(기본값)를 선택합니다.

     다른 인스턴스 옵션은 NVMe 지원 클래스입니다. 자세한 내용은 NVMe 지원 인스턴스를 참조하세요.
   - 인스턴스 클래스에서 필요에 맞는 인스턴스 유형을 선택합니다. 인스턴스 클래스에 대한 자세한 설명은 섹션을 참조하세요인스턴스 클래스 사양.
   - 인스턴스 수에서 요구 사항을 가장 잘 반영하는 숫자를 선택합니다. 숫자가 낮을수록 비용이 낮아지고 클러스터에서 관리할 수 있는 읽기/쓰기 볼륨이 낮아집니다.



7. 연결의 경우 EC2 컴퓨팅 리소스에 연결 안 함을 기본 설정으로 둡니다.

> **Note**
>
> EC2 컴퓨팅 리소스에 연결하면 클러스터에 연결하기 위한 보안 그룹이 자동으로 생성됩니다. 이전 단계에서 이러한 보안 그룹을 수동으로 생성했으므로 두 번째 보안 그룹 세트를 생성하지 않도록 EC2 컴퓨팅 리소스에 연결하지 않음을 선택해야 합니다.

8. 인증 섹션에서 기본 사용자의 사용자 이름을 입력한 다음 자체 관리를 선택합니다. 암호를 입력한 다음 확인합니다.

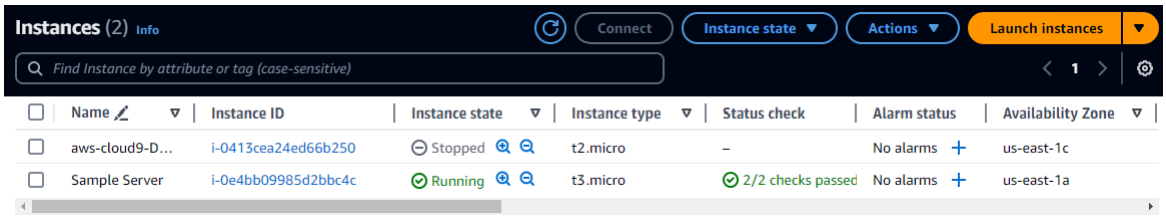
   대신 관리형을 AWS Secrets Manager 선택한 경우 자세한 내용은 Amazon DocumentDB 및를 사용한 암호 관리 AWS Secrets Manager 섹션을 참조하세요.

9. 클러스터 생성을 선택합니다.

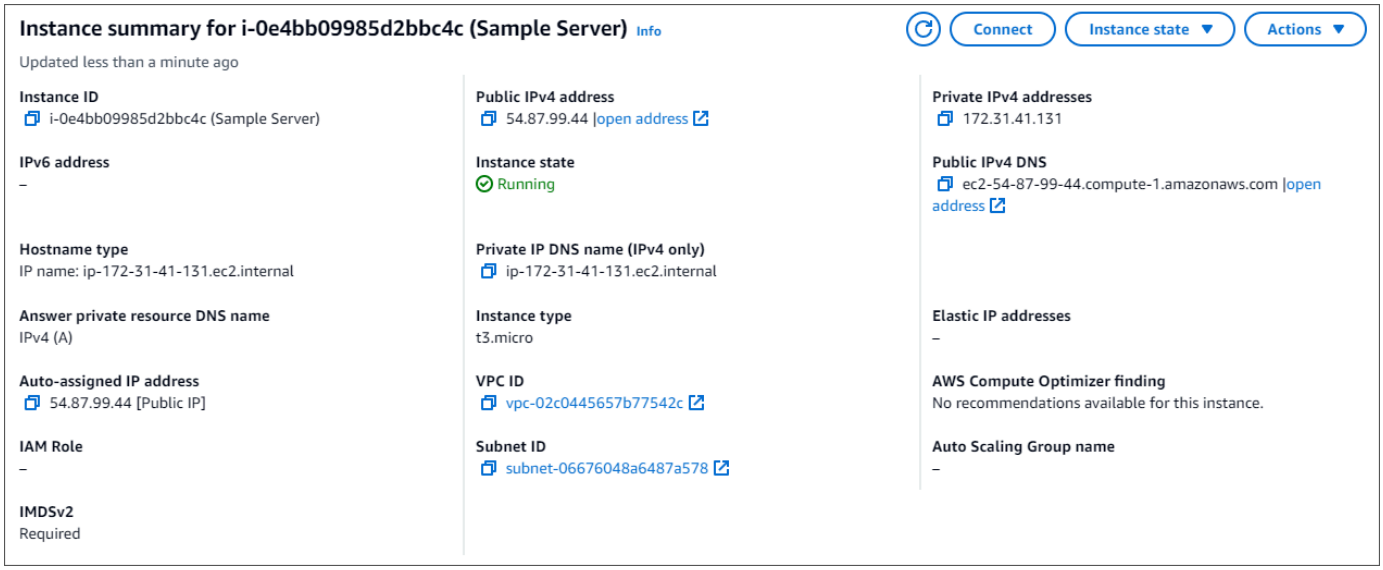
4단계: Amazon EC2 인스턴스 구성

Amazon EC2 인스턴스에 연결하면 MongoDB 셸을 설치할 수 있습니다. mongo 쉘을 설치하면 Amazon DocumentDB 클러스터에 연결하고 쿼리할 수 있습니다. 다음 단계를 완료합니다.

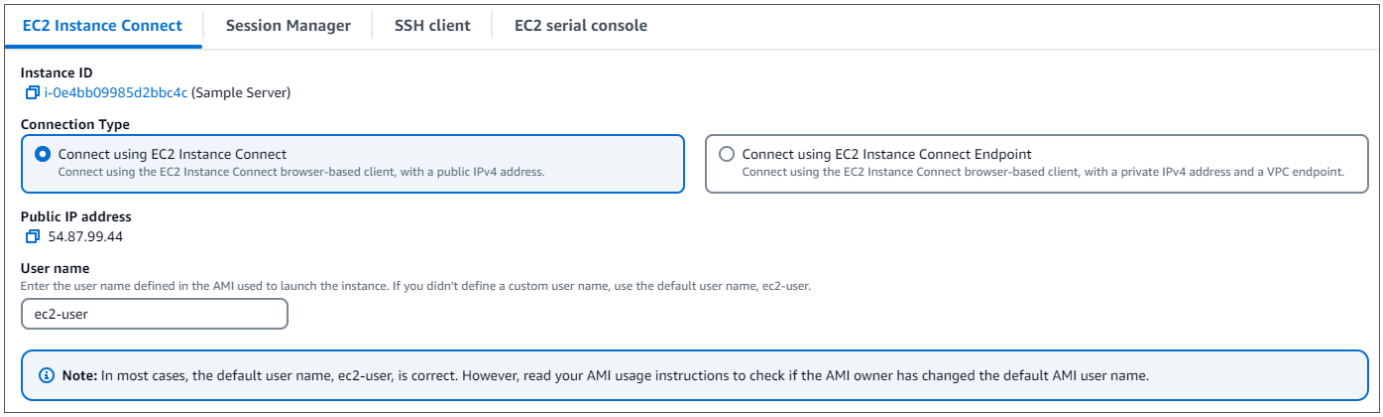
1. Amazon EC2 콘솔에서 인스턴스로 이동하여 방금 생성한 인스턴스가 실행 중인지 확인합니다. 그렇다면 인스턴스 ID를 클릭하여 인스턴스를 선택합니다.



2. 연결을 선택합니다.



3. 연결 방법에는 Amazon EC2 인스턴스 연결, 세션 관리자, SSH 클라이언트 또는 EC2 직렬 콘솔의 4가지 탭 옵션이 표시됩니다. 하나를 선택하고 지침을 따라야 합니다. 완료되면 연결을 선택합니다.



> **Note**
>
> 이 연습을 시작한 후 IP 주소가 변경되었거나 나중에 환경으로 돌아오는 경우 demoEC2 보안 그룹 인바운드 규칙을 업데이트하여 새 API 주소의 인바운드 트래픽을 활성화해야 합니다.

5단계: MongoDB 쉘 설치

이제 Amazon DocumentDB 클러스터를 연결하고 쿼리하는 데 사용하는 명령줄 유틸리티인 MongoDB 셸을 설치할 수 있습니다. Amazon DocumentDB 현재 MongoDB 쉘에는 최신 버전인 mongosh와 이전 버전인 mongo 쉘의 두 가지 버전이 있습니다.

> **Important**
>
> 현재 Amazon DocumentDB에 대한 IAM 자격 증명 인증에서 지원되지 않는 버전 6.13.1 이전의 Node.js 드라이버에는 알려진 제한 사항이 있습니다. Node.js 드라이버를 사용하는 Node.js 드라이버 및 도구(예: mongosh)는 Node.js 드라이버 버전 6.13.1 이상을 사용하도록 업그레이드해야 합니다.

아래 지침에 따라 운영 체제용 MongoDB 쉘을 설치합니다.

On Amazon Linux

Amazon Linux에 MongoDB 쉘을 설치하려면

IAM을 사용하지 않고 최신 MongoDB 셸(mongosh)을 사용하여 Amazon DocumentDB 클러스터에 연결하려면 다음 단계를 따르세요.

1. 리포지토리 파일을 생성합니다. 생성한 EC2 인스턴스의 명령줄에서 다음 명령을 실행합니다.

```
echo -e "[mongodb-org-5.0] \nname=MongoDB Repository\nbaseurl=https://
repo.mongodb.org/yum/amazon/2023/mongodb-org/5.0/x86_64/\ngpgcheck=1 \nenabled=1
 \ngpgkey=https://pgp.mongodb.com/server-5.0.asc" | sudo tee /etc/yum.repos.d/
mongodb-org-5.0.repo
```

2. 완료되면 명령 프롬프트에서 다음 두 명령 옵션 중 하나를 사용하여 mongosh를 설치합니다.

   옵션 1 - Amazon EC2 구성 중에 기본 Amazon Linux 2023을 선택한 경우 다음 명령을 입력합니다.

```
sudo yum install -y mongodb-mongosh-shared-openssl3
```

   옵션 2 - Amazon EC2 구성 중에 Amazon Linux 2를 선택한 경우 다음 명령을 입력합니다.

```
sudo yum install -y mongodb-mongosh
```

IAM을 사용하는 경우 이전 버전의 MongoDB 셸(5.0)을 사용하여 Amazon DocumentDB 클러스터에 연결해야 합니다. 다음 단계를 따르세요.

1. 리포지토리 파일을 생성합니다. 생성한 EC2 인스턴스의 명령줄에서 다음 명령을 실행합니다.

```
echo -e "[mongodb-org-5.0] \nname=MongoDB Repository\nbaseurl=https://
repo.mongodb.org/yum/amazon/2023/mongodb-org/5.0/x86_64/\ngpgcheck=1 \nenabled=1
 \ngpgkey=https://pgp.mongodb.com/server-5.0.asc" | sudo tee /etc/yum.repos.d/
mongodb-org-5.0.repo
```

2. 완료되면 명령 프롬프트에서 다음 명령 옵션을 사용하여 mongodb 5.0 셸을 설치합니다.

```
sudo yum install -y mongodb-org-shell
```

On Ubuntu

Ubuntu에 mongosh를 설치하려면

1. 패키지 관리 시스템에서 사용할 퍼블릭 키를 가져옵니다.

```
curl -fsSL https://pgp.mongodb.com/server-5.0.asc | sudo gpg --dearmor -o /usr/
share/keyrings/mongodb-server-5.0.gpg
```

2. Ubuntu 버전에 맞는 명령을 사용하여 MongoDB의 목록 파일 `mongodb-org-5.0.list`를 생성합니다.

```
echo "deb [ arch=amd64,arm64 signed-by=/usr/share/keyrings/mongodb-
server-5.0.gpg ] https://repo.mongodb.org/apt/ubuntu focal/mongodb-org/5.0
 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-5.0.list
```

3. 다음 명령을 사용하여 로컬 패키지 데이터베이스를 가져오고 업데이트합니다.

```
sudo apt-get update
```

4. mongosh를 설치합니다.

```
sudo apt-get install -y mongodb-mongosh
```

Ubuntu 시스템에서 이전 버전의 MongoDB를 설치하는 방법에 대한 자세한 내용은 Install MongoDB Community Edition on Ubuntu를 참조하세요.

On other operating systems

다른 운영 체제에서 mongo 쉘을 설치하려면 MongoDB 설명서의 MongoDB Community Edition 설치를 참조하세요.

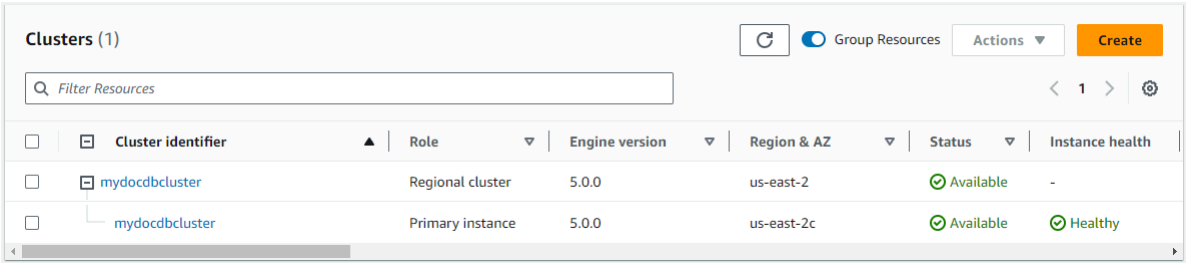
6단계: Amazon DocumentDB TLS 관리

다음 코드를 사용하여 Amazon DocumentDB용 CA 인증서를 다운로드하세요. `wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem`

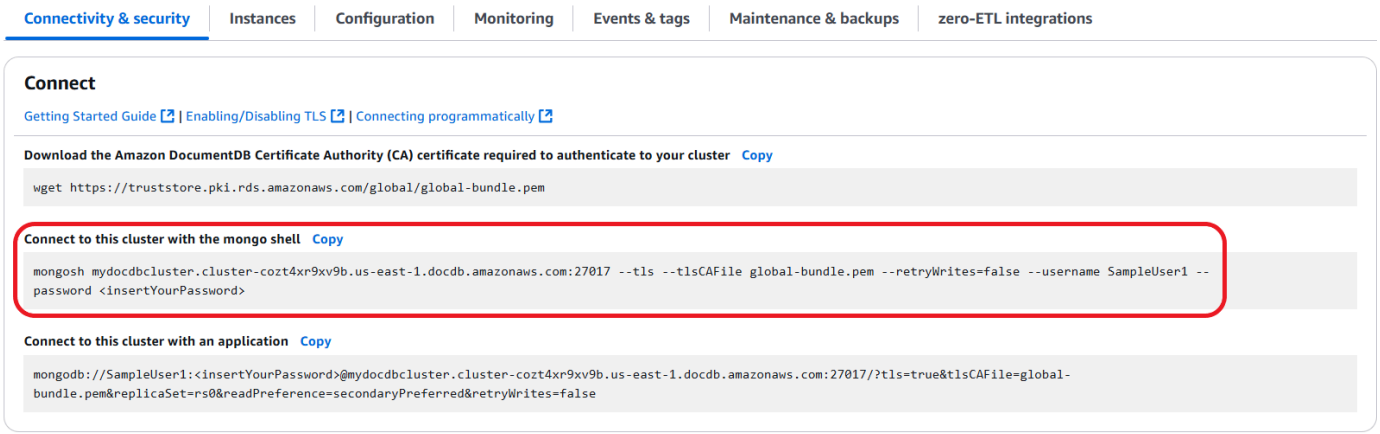
> **Note**
>
> 전송 계층 보안 (TLS) 은 모든 새 Amazon DocumentDB 클러스터에 기본적으로 활성화됩니다. 자세한 내용은 Amazon DocumentDB 클러스터 TLS 설정을 참조하세요.

7단계: Amazon DocumentDB 클러스터에 연결

1. Amazon DocumentDB 콘솔의 클러스터에서 클러스터를 찾습니다. 해당 클러스터의 클러스터 식별자를 클릭하여 생성한 클러스터를 선택합니다.



2. 연결 및 보안 탭에서 연결 상자의 mongo 쉘을 사용하여 이 클러스터에 연결을 찾습니다.



제공된 연결 문자열을 복사하여 터미널에 붙여넣습니다.

해당 항목을 다음과 같이 변경합니다.

a. 문자열에 올바른 사용자 이름이 있는지 확인합니다.

b. 그런 다음 연결할 때 mongo 쉘에서 비밀번호를 입력하라는 메시지가 표시되도록 <insertYourPassword> 항목을 생략합니다.

c. 선택 사항: IAM 인증을 사용하거나 이전 버전의 MongoDB 셸을 사용하는 경우 다음과 같이 연결 문자열을 수정합니다.

```
mongo --ssl --host mydocdbcluster.cluster-cozt4xr9xv9b.us-
east-1.docdb.amazonaws.com:27017 --sslCAFile global-bundle.pem --
username SampleUser1 --password
```

mydocdbcluster.cluster-cozt4xr9xv9b.us-east-1를 클러스터의 동일한 정보로 바꿉니다.

3. 이제 터미널에서 Enter를 누릅니다. 이제 비밀번호를 입력하라는 메시지가 표시됩니다. 암호를 입력합니다.

4. 암호를 입력하고 rs0 [direct: primary] <env-name>> 메시지가 표시되면 Amazon DocumentDB 클러스터에 성공적으로 연결된 것입니다.

연결에 문제가 있으신가요? Amazon DocumentDB 문제 해결을 참조하세요.

8단계: 데이터 삽입 및 쿼리

이제 클러스터에 연결되었으므로 몇 가지 쿼리를 실행하여 문서 데이터베이스 사용에 익숙해질 수 있습니다.

1. 단일 문서를 삽입하려면 다음을 입력합니다:

```
db.collection.insertOne({"hello":"DocumentDB"})
```

출력은 다음과 같습니다:

```
{
  acknowledged: true,
  insertedId: ObjectId('673657216bdf6258466b128c')
}
```

2. findOne() 명령으로 작성한 문서를 읽을 수 있습니다 (단일 문서만 반환하기 때문). 다음 내용을 입력합니다.

```
db.collection.findOne()
```

다음과 같은 출력내용을 얻게 됩니다.

```
{ "_id" : ObjectId("5e401fe56056fda7321fbd67"), "hello" : "DocumentDB" }
```

3. 쿼리를 몇 개 더 수행하려면 게임 프로필 사용 사례를 고려해 보세요. 먼저 profiles이라는 제목의 컬렉션에 몇 개의 항목을 삽입합니다. 다음 내용을 입력합니다.

```
db.profiles.insertMany([{ _id: 1, name: 'Matt', status: 'active', level: 12, score:
 202 },
      { _id: 2, name: 'Frank', status: 'inactive', level: 2, score: 9 },
      { _id: 3, name: 'Karen', status: 'active', level: 7, score: 87 },
      { _id: 4, name: 'Katie', status: 'active', level: 3, score: 27 }
])
```

다음과 같은 출력내용을 얻게 됩니다.

```
{ acknowledged: true, insertedIds: { '0': 1, '1': 2, '2': 3, '3': 4 } }
```

4. find() 명령을 사용하여 프로필 컬렉션의 모든 문서를 반환합니다. 다음 내용을 입력합니다.

```
db.profiles.find()
```

3단계에서 입력한 데이터와 일치하는 출력을 받게 됩니다.

5. 필터를 사용하여 단일 문서에 대한 쿼리를 사용하세요. 다음을 입력합니다:

```
db.profiles.find({name: "Katie"})
```

다음과 같은 출력내용을 얻게 됩니다.

```
{ "_id" : 4, "name" : "Katie", "status": "active", "level": 3, "score":27}
```

6. 이제 프로필을 찾아 findAndModify 명령을 사용하여 해당 프로필을 수정해 보겠습니다. 다음 코드를 사용하여 Matt 사용자에게 10포인트를 추가로 제공합니다.

```
db.profiles.findAndModify({
        query: { name: "Matt", status: "active"},
        update: { $inc: { score: 10 } }
    })
```

다음과 같은 결과가 출력됩니다 (참고로 그의 점수는 아직 오르지 않았습니다):

```
{
    [{_id : 1, name : 'Matt', status: 'active', level: 12, score: 202}]
```

7. 다음 쿼리를 통해 그의 점수가 변경되었는지 확인할 수 있습니다:

   ```
   db.profiles.find({name: "Matt"})
   ```

   출력은 다음과 같습니다.

   ```
   { "_id" : 1, "name" : "Matt", "status" : "active", "level" : 12, "score" : 212 }
   ```

9단계: 살펴보기

축하합니다! Amazon DocumentDB에 대한 퀵 스타트 가이드를 성공적으로 완료했습니다.

다음 단계? 몇 가지 인기 있는 기능을 갖춘 이 강력한 데이터베이스를 최대한 활용하는 방법을 알아보세요.

- Amazon DocumentDB 관리
- 스케일링
- 백업 및 복원

> **Note**
>
> 비용을 절감하려면 Amazon DocumentDB 클러스터를 중지하여 비용을 절감하거나 클러스터를 삭제할 수 있습니다. 기본적으로 30분 동안 사용하지 않으면 AWS Cloud9 환경이 기본 Amazon EC2 인스턴스를 중지합니다.

Amazon DocumentDB JDBC 드라이버를 사용하여 연결합니다.

Amazon DocumentDB용 JDBC 드라이버는 개발자를 위한 SQL 관계형 인터페이스를 제공하며, Tableau 및 DbVisualizer와 같은 BI 도구에서 연결할 수 있도록 합니다.

자세한 내용은 GitHub의 Amazon DocumentDB JDBC 드라이버 설명서를 참조하세요.

주제

- 시작
- 타블로 데스크톱에서 Amazon DocumentDB에 연결
- DB 비주얼라이저에서 Amazon DocumentDB에 연결
- JDBC 자동 스키마 생성
- SQL 지원 및 제한 사항
- 문제 해결

시작

1단계. Amazon DocumentDB 클러스터

Amazon DocumentDB 클러스터를 생성하지 않은 경우에는 Amazon DocumentDB 개발자 안내서의 시작하기 섹션에 있는 지침을 사용하여 클러스터를 만드세요.

> **Note**
>
> Amazon DocumentDB는 가상 프라이빗 클라우드(VPC) 전용 서비스입니다. 클러스터 VPC 외부의 로컬 시스템에서 연결하는 경우 Amazon EC2 인스턴스에 대한 SSH 연결을 생성해야 합니다. 이 경우 Connect with EC2의 지침을 사용하여 클러스터를 시작하세요. SSH 터널링에 대한 자세한 내용과 필요할 수 있는 시기에 대해서는 SSH 터널을 사용하여 Amazon DocumentDB에 연결을 참조하세요.

2단계. JRE 또는 JDK 설치

BI 애플리케이션에 따라 컴퓨터에 64비트 JRE 또는 JDK 설치 버전 8 이상이 설치되어 있는지 확인해야 할 수 있습니다. Java SE 런타임 환경 8은 여기에서 다운로드할 수 있습니다.

3단계. DocumentDB JDBC 드라이버 다운로드

여기에서 DocumentDB JDBC 드라이버를 다운로드하세요. 드라이버는 단일 JAR 파일 (예: documentdb-jdbc-1.0.0-all.jar) 로 패키지됩니다.

4단계. SSH 터널을 사용하여 Amazon DocumentDB에 연결

Amazon DocumentDB(MongoDB 호환) 클러스터는 Amazon Virtual Private Cloud (Amazon VPC) 내에 배포됩니다. Amazon EC2 인스턴스 또는 동일한 Amazon VPC에 배포된 다른 AWS 서비스에서 직접 액세스할 수 있습니다. 또한 EC2a 인스턴스 또는 VPCs 피어링을 통해 동일한 AWS 리전 또는 다른 리전의 다른 VPC에 있는 다른 AWS 서비스에서 Amazon DocumentDB에 액세스할 수 있습니다.

SSH 터널링(포트 포워딩이라고도 함)을 사용하여 클러스터의 VPC 외부에서 Amazon DocumentDB 리소스에 액세스할 수 있습니다. 이는 대부분의 사용자가 DocumentDB 클러스터와 동일한 VPC에 있는 VM에서 애플리케이션을 실행하지 않는 경우입니다.

SSH 터널을 만들려면 Amazon DocumentDB 클러스터와 동일한 Amazon VPC에서 실행 중인 Amazon EC2 인스턴스가 필요합니다. 클러스터와 동일한 VPC에서 기존 EC2 인스턴스를 사용하거나 새 인스턴스를 생성할 수 있습니다. 로컬 컴퓨터에서 다음 명령을 실행하여 Amazon DocumentDB 클러스터 `sample-cluster.node.us-east-1.docdb.amazonaws.com`에 대한 SSH 터널을 설정할 수 있습니다.

```
ssh -i "ec2Access.pem" -L 27017:sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017 ubuntu@ec2-34-229-221-164.compute-1.amazonaws.com -
N
```

-L 플래그는 로컬 포트를 전달할 때 사용됩니다. 이는 VPC 외부의 클라이언트에서 실행되는 모든 BI 도구에 연결하기 위한 전제 조건입니다. 위 단계를 실행한 후에는 선택한 BI 도구의 다음 단계로 넘어갈 수 있습니다.

SSH 터널링에 대한 자세한 내용은 SSH 터널을 사용하여 Amazon DocumentDB에 연결하는 방법에 대한 설명서를 참조하세요.

타블로 데스크톱에서 Amazon DocumentDB에 연결

주제

- Amazon DocumentDB JDBC 드라이버 추가

- 타블로를 사용하여 Amazon DocumentDB에 연결하기 - SSH 터널

Amazon DocumentDB JDBC 드라이버 추가

Tableau Desktop에서 Amazon DocumentDB에 연결하려면 Amazon DocumentDB JDBC 드라이버와 DocumentDB Tableau 커넥터를 다운로드하여 설치해야 합니다.

1. Amazon DocumentDB JDBC 드라이버 리포지토리에서 Amazon DocumentDB JDBC 드라이버 JAR 파일을 다운로드하여 운영 체제에 따라 다음 디렉터리 중 하나에 복사합니다.
   - Windows – `C:\Program Files\Tableau\Drivers`
   - macOS – `~/Library/Tableau/Drivers`
2. Tableau Exchange 웹 사이트에서 DocumentDB Tableau 커넥터(TACO 파일)를 다운로드하여 My Tableau Repository/Connectors 디렉터리에 복사합니다.
   - Windows – `C:\Users\[user]\Documents\My Tableau Repository\Connectors`
   - macOS – `/Users/[user]/Documents/My Tableau Repository/Connectors`

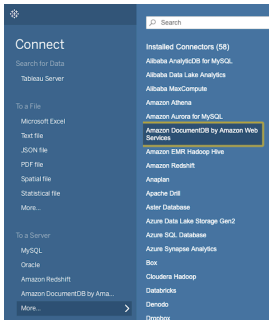
자세한 내용은 Tableau 설명서를 참조하세요.

> **Note**
>
> 최신 CA 인증서를 사용하는 경우 JDBC 드라이버를 v1.4.5(AWS ThisGitHub 리포지토리에서 사용 가능)로 업그레이드해야 합니다.

타블로를 사용하여 Amazon DocumentDB에 연결하기 - SSH 터널

DocumentDB 클러스터의 VPC 외부에 있는 클라이언트 컴퓨터에서 Tableau에 연결하려면 아래 단계를 수행하기 전에 SSH 터널을 설정해야 합니다.

1. Tableau Desktop 애플리케이션을 실행합니다.
2. 연결 > 서버로 > 기타로 이동합니다.
3. 설치된 커넥터에서 Amazon Web Services의 Amazon DocumentDB를 선택합니다.

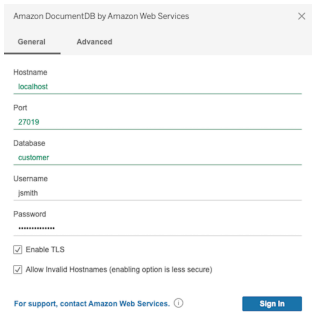


타블로를 사용하여 Amazon DocumentDB에 연결 - 외부 SSH 터널

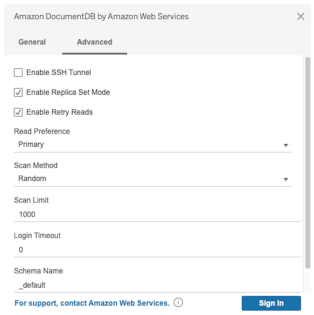
1. 필수 연결 파라미터 호스트 이름, 포트, 데이터베이스, 사용자 이름 및 암호를 입력합니다. 아래 예제의 연결 파라미터는 JDBC 연결 문자열과 동일합니다.

   ```
   jdbc:documentdb://localhost:27019/test?
   tls=true&tlsAllowInvalidHostnames=true&scanMethod=random&scanLimit=1000&logir
   ```

   성 컬렉션에서 사용자 이름 및 암호 파라미터가 별도로 전달됩니다. 연결 문자열 파라미터에 대한 자세한 내용은 Amazon DocumentDB JDBC 드라이버 github 설명서를 참조하세요.



2. (선택 사항) 고급 탭에서 고급 옵션을 찾을 수 있습니다.

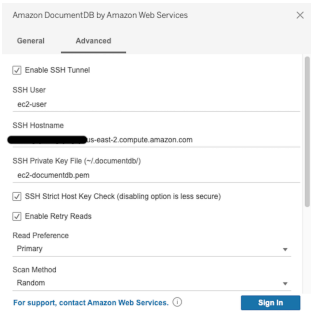


3. 로그인을 선택합니다.

타블로를 사용하여 Amazon DocumentDB에 연결 - 내부 SSH 터널

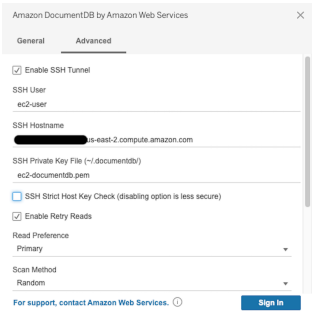
> **Note**
>
> 터미널을 사용하여 SSH 터널을 설정하지 않으려는 경우 Tableau GUI를 사용하여 JDBC 드라이버가 SSH 터널을 생성하는 데 기본적으로 사용할 EC2 인스턴스 세부 정보를 지정할 수 있습니다.

1. 고급 탭에서 SSH 터널 활성화 옵션을 선택하여 추가 속성을 검토하세요.



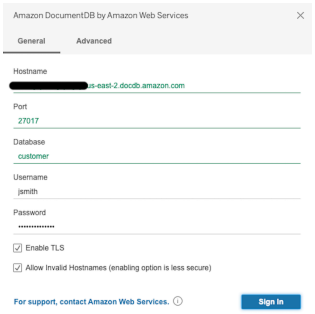
2. SSH 사용자, SSH 호스트 이름 및 SSH 개인 키 파일을 입력합니다.
3. (선택 사항) 알려진 호스트 파일에 대한 호스트 키 검사를 우회하는 SSH 엄격한 호스트 키 검사 옵션을 비활성화할 수 있습니다.

> **Note**
>
> 이 옵션을 비활성화하면 중간자 공격으로 이어질 수 있으므로 보안이 취약합니다.



4. 필수 파라미터 (호스트 이름, 포트, 데이터베이스, 사용자 이름 및 암호) 를 입력합니다.

> **Note**
>
> 내부 SSH 터널 옵션을 사용할 때는 로컬 호스트가 아닌 DocumentDB 클러스터 엔드포인트를 사용해야 합니다.



5. Sign In(로그인)을 선택합니다.

DB 비주얼라이저에서 Amazon DocumentDB에 연결

주제

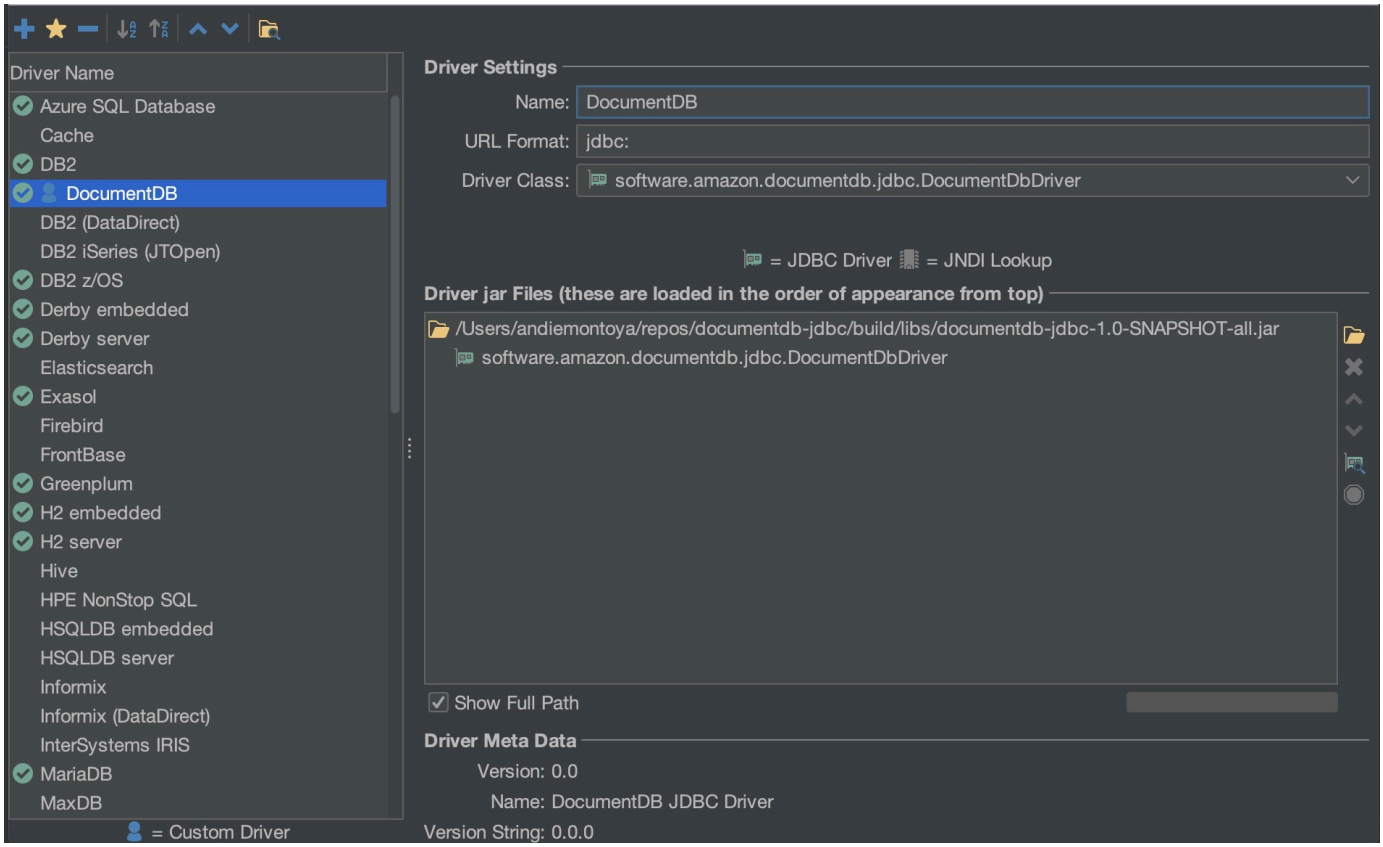
- Amazon DocumentDB JDBC 드라이버 추가
- DB 비주얼라이저를 사용하여 Amazon DocumentDB에 연결

Amazon DocumentDB JDBC 드라이버 추가

DbVisualizer에서 Amazon DocumentDB에 연결하려면 먼저 Amazon DocumentDB JDBC 드라이버를 가져와야 합니다.

1. DbVisualizer 애플리케이션을 시작하고 도구 > 드라이버 관리자... 메뉴 경로로 이동합니다.
2. +를 선택합니다 (또는 메뉴에서 드라이버 > 드라이버 생성을 선택).
3. 이름을 `DocumentDB`으로 설정합니다.
4. URL 형식을 다음으로 설정합니다. `jdbc:documentdb://<host>[:port]/<database>[?option=value[&option=value[...]]]`
5. 폴더 버튼을 선택한 다음 Amazon DocumentDB JDBC 드라이버 JAR 파일을 선택하고 열기 버튼을 선택합니다.

6. 드라이버 클래스 필드가 로 설정되어 있는지 확인하세요.
`software.amazon.documentdb.jdbc.DocumentDbDriver` DocumentDB의 드라이버 관리자 설정은 다음 예시와 같습니다.



7. 대화 상자를 닫습니다. Amazon DocumentDB JDBC 드라이버가 설치되고 바로 사용할 수 있습니다.

DB 비주얼라이저를 사용하여 Amazon DocumentDB에 연결

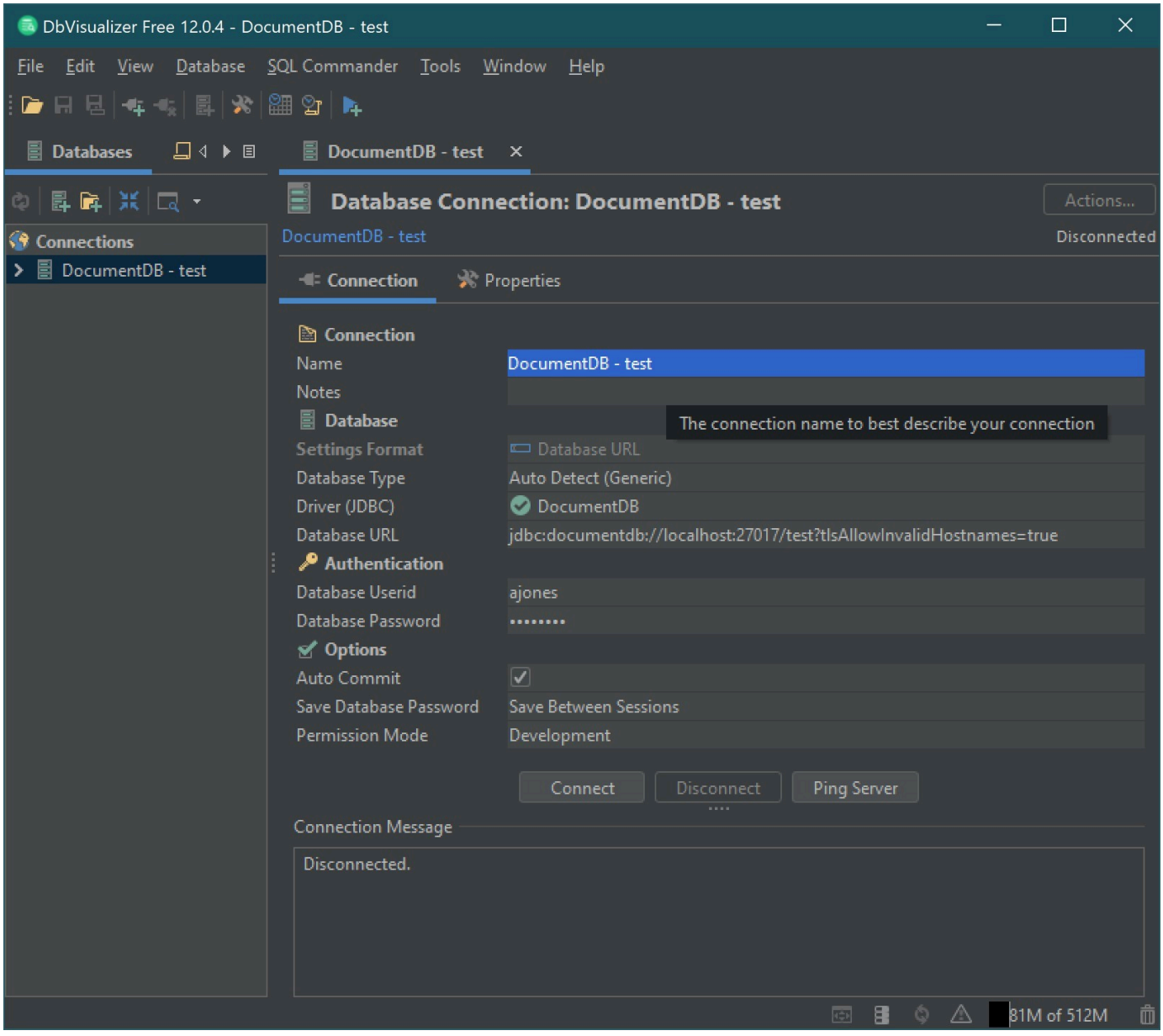
DB 비주얼라이저를 사용하여 Amazon DocumentDB에 연결

1. Amazon DocumentDB 클러스터의 VPC 외부에서 연결하는 경우 SSH 터널을 설정했는지 확인하세요.
2. 최상위 메뉴에서 데이터베이스 > 데이터베이스 연결 생성을 선택합니다.
3. 이름 필드에 설명이 포함된 이름을 입력합니다.
4. 드라이버 (JDBC) 를 이전 섹션에서 만든 DocumentDB 드라이버로 설정합니다.
5. 데이터베이스 URL을 JDBC 연결 문자열로 설정합니다.

예: `jdbc:documentdb://localhost:27017/database?tlsAllowInvalidHostnames=true`

6. 데이터베이스 사용자 ID를 Amazon DocumentDB 사용자 ID로 설정합니다.
7. 데이터베이스 암호를 사용자 ID에 해당하는 암호로 설정합니다.

데이터베이스 연결 대화 상자는 다음 대화 상자와 비슷합니다.



8. 연결을 선택합니다.

JDBC 자동 스키마 생성

Amazon DocumentDB는 문서 데이터베이스이므로 테이블과 스키마라는 개념이 없습니다. 하지만 Tableau와 같은 BI 도구에서는 연결하는 데이터베이스가 스키마를 제공할 것으로 예상합니다. 특히 JDBC 드라이버 연결에서 데이터베이스의 컬렉션에 대한 스키마를 가져와야 하는 경우 데이터베이스의 모든 컬렉션을 폴링합니다. 드라이버는 해당 컬렉션에 대한 스키마의 캐시된 버전이 이미 존재하는지 확인합니다. 캐시된 버전이 없는 경우 문서 컬렉션을 샘플링하고 다음 동작에 따라 스키마를 만듭니다.

주제

- 스키마 생성 제한
- 스캔 방법 옵션
- Amazon DocumentDB 데이터 유형
- 스칼라 문서 필드 매핑
- 객체 및 배열 데이터 유형 처리

스키마 생성 제한

DocumentDB JDBC 드라이버는 식별자 길이를 128자로 제한합니다. 스키마 생성기는 생성된 식별자(테이블 이름 및 열 이름)의 길이를 줄여 해당 한도에 맞는지 확인할 수 있습니다.

스캔 방법 옵션

연결 문자열 또는 데이터 소스 옵션을 사용하여 샘플링 동작을 수정할 수 있습니다.

- 스캔 방법= <option>
  - random - (기본값) - 샘플 문서가 무작위 순서로 반환됩니다.
  - IDForward - 샘플 문서가 ID 순으로 반환됩니다.
  - IDReverse - 샘플 문서가 ID의 역순으로 반환됩니다.
  - 모두 - 컬렉션의 모든 문서를 샘플링합니다.
- ScanLimit= <n>- 샘플링할 문서 수입니다. 값은 양의 정수여야 합니다. 기본값은 1000입니다. ScanMethod가 모두로 설정된 경우 이 옵션은 무시됩니다.

Amazon DocumentDB 데이터 유형

Amazon DocumentDB 서버는 다양한 MongoDB 데이터 유형을 지원합니다. 지원되는 데이터 유형 및 관련 JDBC 데이터 형식은 다음과 같습니다.

MongoDB 데이터 형식	DocumentDB에서 지원	JDBC 데이터 형식
이진 데이터	예	VARBINARY
불	예	BOOLEAN
배정밀도 실수	예	DOUBLE
32비트 정수	예	INTEGER
64비트 정수	예	BIGINT
String	예	VARCHAR
ObjectId	예	VARCHAR
날짜	예	TIMESTAMP
Null	예	VARCHAR
정규식	예	VARCHAR
Timestamp	예	VARCHAR
MinKey	예	VARCHAR
MaxKey	예	VARCHAR
객체	예	가상 테이블
Array	예	가상 테이블
Decimal128	아니요	DECIMAL
JavaScript	아니요	VARCHAR
JavaScript (범위 포함)	아니요	VARCHAR
정의되지 않음	아니요	VARCHAR
Symbol	아니요	VARCHAR

MongoDB 데이터 형식	DocumentDB에서 지원	JDBC 데이터 형식
DB 포인터 (4.0 이상)	아니요	VARCHAR

스칼라 문서 필드 매핑

컬렉션에서 문서 샘플을 스캔할 때 JDBC 드라이버는 컬렉션의 샘플을 나타내는 스키마를 하나 이상 만듭니다. 일반적으로 문서의 스칼라 필드는 테이블 스키마의 열에 매핑됩니다. 예를 들어 team이라는 컬렉션과 단일 `{ "_id" : "112233", "name" : "Alastair", "age": 25 }` 문서에서는 다음과 같이 스키마에 매핑됩니다.

테이블 이름	열 이름	데이터 형식	키
팀	team id	VARCHAR	PK
팀	name	VARCHAR	
팀	나이	INTEGER	

데이터 유형 충돌 촉진

샘플링된 문서를 스캔할 때 필드의 데이터 유형이 문서마다 일치하지 않을 수 있습니다. 이 경우 JDBC 드라이버는 JDBC 데이터 유형을 샘플 문서의 모든 데이터 유형에 적합한 공통 데이터 유형으로 승격시킵니다.

예:

```
{
"_id" : "112233",
"name" : "Alastair", "age" : 25
}

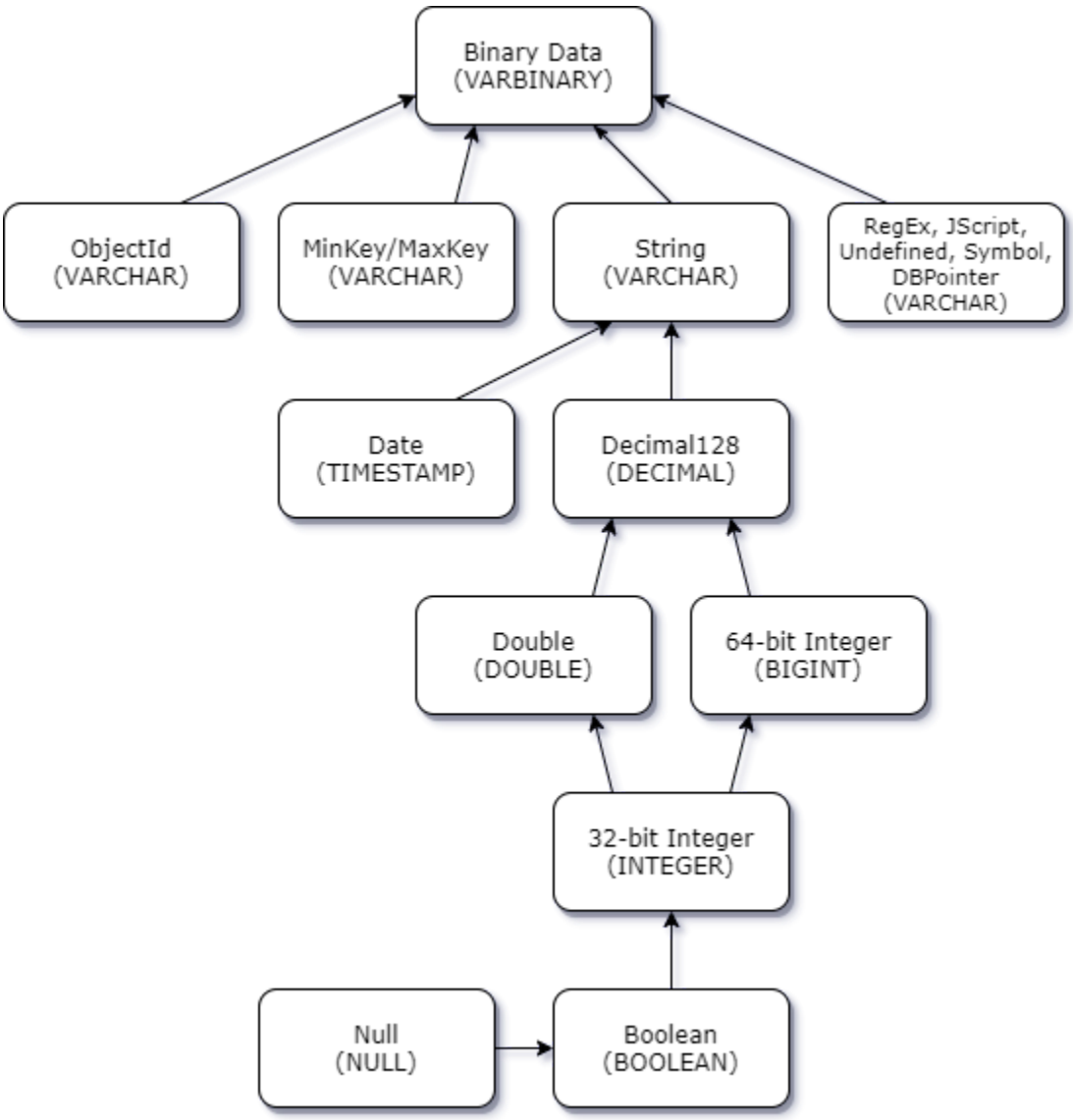
{
"_id" : "112244",
"name" : "Benjamin",
"age" : "32"
}
```

첫 번째 문서에서는 age 필드의 유형이 32비트 정수이고 두 번째 문서에서는 문자열입니다. 여기서 JDBC 드라이버는 JDBC 데이터 유형을 VARCHAR로 승격시켜 두 데이터 유형 중 하나를 발견했을 때 이를 처리합니다.

테이블 이름	열 이름	데이터 형식	키
팀	team id	VARCHAR	PK
팀	name	VARCHAR	
팀	나이	VARCHAR	

스칼라-스칼라 충돌 프로모션

다음 다이어그램은 스칼라-스칼라 데이터 유형 충돌을 해결하는 방법을 보여줍니다.



스칼라-복소수 형식 충돌 승격

스칼라-스칼라 형식 충돌과 마찬가지로, 서로 다른 문서에 있는 동일한 필드에서도 복소수 (배열 및 객체) 와 스칼라 (정수, 부울 등) 간에 데이터 유형이 충돌할 수 있습니다. 이러한 모든 충돌은 해당 필드에 대해 VARCHAR로 해결 (승격) 됩니다. 이 경우 배열 및 객체 데이터가 JSON 표현으로 반환됩니다.

임베디드 어레이 - 문자열 필드 충돌 예제:

```
{
   "_id":"112233",
   "name":"George Jackson",
   "subscriptions":[
      "Vogue",
```

```
      "People",
      "USA Today"
   ]
}
{
   "_id":"112244",
   "name":"Joan Starr",
   "subscriptions":1
}
```

위 예제는 customer2 테이블의 스키마에 매핑됩니다.

테이블 이름	열 이름	데이터 형식	키
고객 2	customer2 id	VARCHAR	PK
고객 2	name	VARCHAR	
고객 2	구독	VARCHAR	

and the customer1_subscriptions 가상 테이블:

테이블 이름	열 이름	데이터 형식	키
customer1_subscriptions	customer1 id	VARCHAR	PK/FK
customer1_subscriptions	subscriptions_index_lvl0	BIGINT	PK
customer1_subscriptions	값	VARCHAR	
customer_address	city	VARCHAR	
customer_address	리전	VARCHAR	
customer_address	country	VARCHAR	
customer_address	code	VARCHAR	

객체 및 배열 데이터 유형 처리

지금까지는 스칼라 데이터형이 매핑되는 방식만 설명했습니다. 객체 및 배열 데이터 유형은 (현재) 가상 테이블에 매핑되어 있습니다. JDBC 드라이버는 문서의 개체 또는 배열 필드를 나타내는 가상 테이블을 만듭니다. 매핑된 가상 테이블의 이름은 원본 컬렉션의 이름 뒤에 밑줄 문자("_")로 구분된 필드 이름을 연결합니다.

기본 테이블의 프라이머리 키("_id")는 새 가상 테이블에서 새 이름을 사용하며 연결된 기본 테이블에 외래 키로 제공됩니다.

내장된 배열 유형 필드의 경우 배열의 각 수준에서 배열에 대한 인덱스를 나타내는 인덱스 열이 생성됩니다.

내장된 객체 필드 예제

문서의 개체 필드의 경우 JDBC 드라이버가 가상 테이블에 매핑을 만듭니다.

```
{
   "Collection: customer",
   "_id":"112233",
   "name":"George Jackson",
   "address":{
      "address1":"123 Avenue Way",
      "address2":"Apt. 5",
      "city":"Hollywood",
      "region":"California",
      "country":"USA",
      "code":"90210"
   }
}
```

위의 예는 고객 테이블의 스키마에 매핑됩니다.

테이블 이름	열 이름	데이터 형식	키
customer	customer id	VARCHAR	PK
customer	name	VARCHAR	

및 the customer_address 가상 테이블:

테이블 이름	열 이름	데이터 형식	키
customer_address	customer id	VARCHAR	PK/FK
customer_address	address1	VARCHAR	
customer_address	address2	VARCHAR	
customer_address	city	VARCHAR	
customer_address	리전	VARCHAR	
customer_address	country	VARCHAR	
customer_address	code	VARCHAR	

임베디드 어레이 필드 예제

문서의 배열 필드의 경우 JDBC 드라이버가 가상 테이블로의 매핑도 생성합니다.

```
{
   "Collection: customer1",
   "_id":"112233",
   "name":"George Jackson",
   "subscriptions":[
      "Vogue",
      "People",
      "USA Today"
   ]
}
```

위 예제는 customer1 테이블의 스키마에 매핑됩니다.

테이블 이름	열 이름	데이터 형식	키
customer1	customer1 id	VARCHAR	PK
customer1	name	VARCHAR	

and the customer1_subscriptions 가상 테이블:

테이블 이름	열 이름	데이터 형식	키
customer1_subscriptions	customer1 id	VARCHAR	PK/FK
customer1_subscriptions	subscriptions_index_lvl0	BIGINT	PK
customer1_subscriptions	값	VARCHAR	
customer_address	city	VARCHAR	
customer_address	리전	VARCHAR	
customer_address	country	VARCHAR	
customer_address	code	VARCHAR	

SQL 지원 및 제한 사항

Amazon DocumentDB JDBC 드라이버는 SQL-92 하위 집합과 일부 일반 확장을 지원하는 읽기 전용 드라이버입니다. 자세한 내용은 SQL 제한 설명서 및 JDBC 제한 설명서를 참조하세요.

문제 해결

Amazon DocumentDB JDBC 드라이버 사용에 문제가 있는 경우 문제 해결 안내서를 참조하세요.

Amazon DocumentDB ODBC 드라이버를 사용하여 연결

Amazon DocumentDB용 ODBC 드라이버는 개발자를 위한 SQL 관계형 인터페이스를 제공하며 Power BI Desktop 및 Microsoft Excel과 같은 BI 도구에서 연결할 수 있도록 합니다.

자세한 내용은 GitHub의 Amazon DocumentDB ODBC 드라이버 설명서를 참조하세요.

주제

- 시작
- 윈도우에서 Amazon DocumentDB ODBC 드라이버 설정하기

- 마이크로소프트 엑셀에서 Amazon DocumentDB에 연결
- 마이크로소프트 파워 BI 데스크톱에서 Amazon DocumentDB에 연결
- 자동 스키마 생성
- SQL 지원 및 제한 사항
- 문제 해결

시작

1단계. Amazon DocumentDB 클러스터 생성

Amazon DocumentDB 클러스터가 아직 없는 경우 여러 가지 방법으로 시작할 수 있습니다.

> **Note**
>
> Amazon DocumentDB는 가상 프라이빗 클라우드(VPC) 전용 서비스입니다. 클러스터 VPC 외부의 로컬 시스템에서 연결하는 경우 Amazon EC2 인스턴스에 대한 SSH 연결을 생성해야 합니다. 이 경우 Connect with EC2의 지침을 사용하여 클러스터를 시작하세요. SSH 터널링에 대한 자세한 내용과 필요할 수 있는 시기에 대해서는 SSH 터널을 사용하여 Amazon DocumentDB에 연결을 참조하세요.

2단계. JRE 또는 JDK 설치

BI 애플리케이션에 따라 컴퓨터에 64비트 JRE 또는 JDK 설치 버전 8 이상이 설치되어 있는지 확인해야 할 수 있습니다. Java SE 런타임 환경 8은 여기에서 다운로드할 수 있습니다.

3단계. Amazon DocumentDB ODBC 드라이버 다운로드

여기에서 Amazon DocumentDB ODBC 드라이버를 다운로드하세요. 적절한 설치 프로그램(예: documentdb-odbc-1.0.0.msi)을 선택합니다. 설치 가이드를 따르세요.

4단계. SSH 터널을 사용하여 Amazon DocumentDB에 연결

Amazon DocumentDB 클러스터는 Amazon Virtual Private Cloud(Amazon VPC)에 배포됩니다. Amazon EC2 인스턴스 또는 동일한 Amazon VPC에 배포된 다른 AWS 서비스에서 직접 액세스할 수 있습니다. 또한 Amazon DocumentDB는 Amazon EC2 인스턴스 또는 VPCs 피어링을 통해 동일한 AWS 리전 또는 다른 리전의 다른 VPC에 있는 다른 AWS 서비스에서 액세스할 수 있습니다.

그러나 사용 사례에서 사용자 또는 사용자의 애플리케이션이 클러스터의 VPC 외부에서 Amazon DocumentDB 리소스에 액세스해야 한다고 가정합니다. 이는 대부분의 사용자가 Amazon

DocumentDB 클러스터와 동일한 VPC에 있는 VM에서 애플리케이션을 실행하지 않는 경우입니다. VPC 외부에서 연결하는 경우 SSH 터널링(포트 포워딩이라고도 함)을 사용하여 Amazon DocumentDB 리소스에 액세스할 수 있습니다.

SSH 터널을 만들려면 Amazon DocumentDB 클러스터와 동일한 Amazon VPC에서 실행 중인 Amazon EC2 인스턴스가 필요합니다. 클러스터와 동일한 VPC에서 기존 EC2 인스턴스를 사용하거나 새 인스턴스를 생성할 수 있습니다. 로컬 컴퓨터에서 다음 명령을 사용하여 SSH 터널을 Amazon DocumentDB 클러스터 `sample-cluster.node.us-east-1.docdb.amazonaws.com`으로 설정할 수 있습니다.

```
ssh -i "ec2Access.pem" -L 27017:sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017 ubuntu@ec2-34-229-221-164.compute-1.amazonaws.com -
N
```

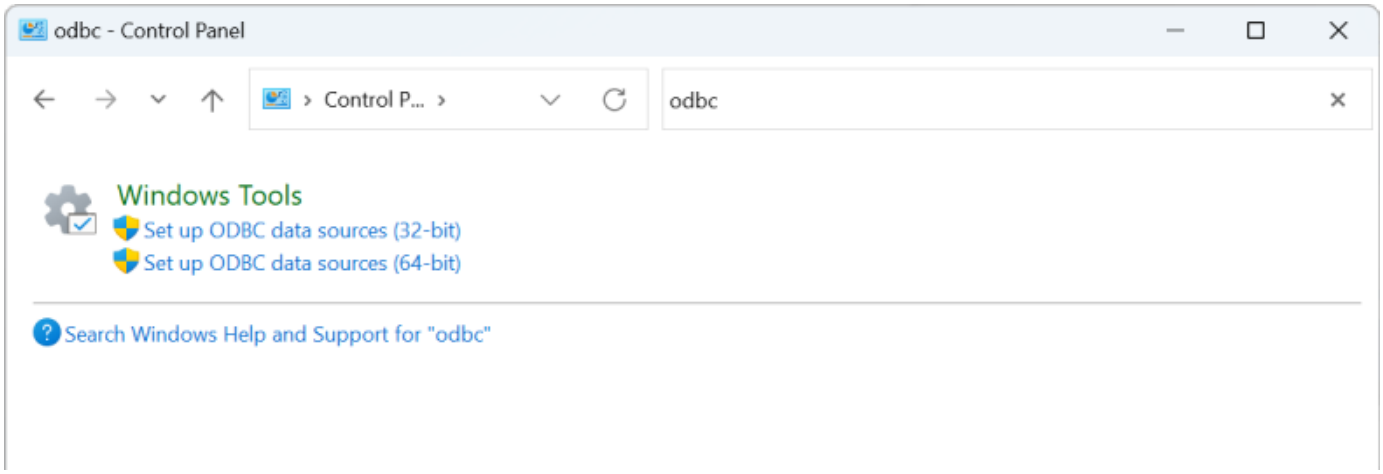
`-L` 플래그는 로컬 포트를 전달하는 데 사용됩니다. 이는 VPC 외부의 클라이언트에서 실행되는 모든 BI 도구에 연결하기 위한 전제 조건입니다. 위 단계를 실행한 후에는 선택한 BI 도구의 다음 단계로 넘어갈 수 있습니다.

SSH 터널링에 대한 자세한 내용은 SSH 터널을 사용하여 Amazon DocumentDB에 연결 설명서를 참조하세요.

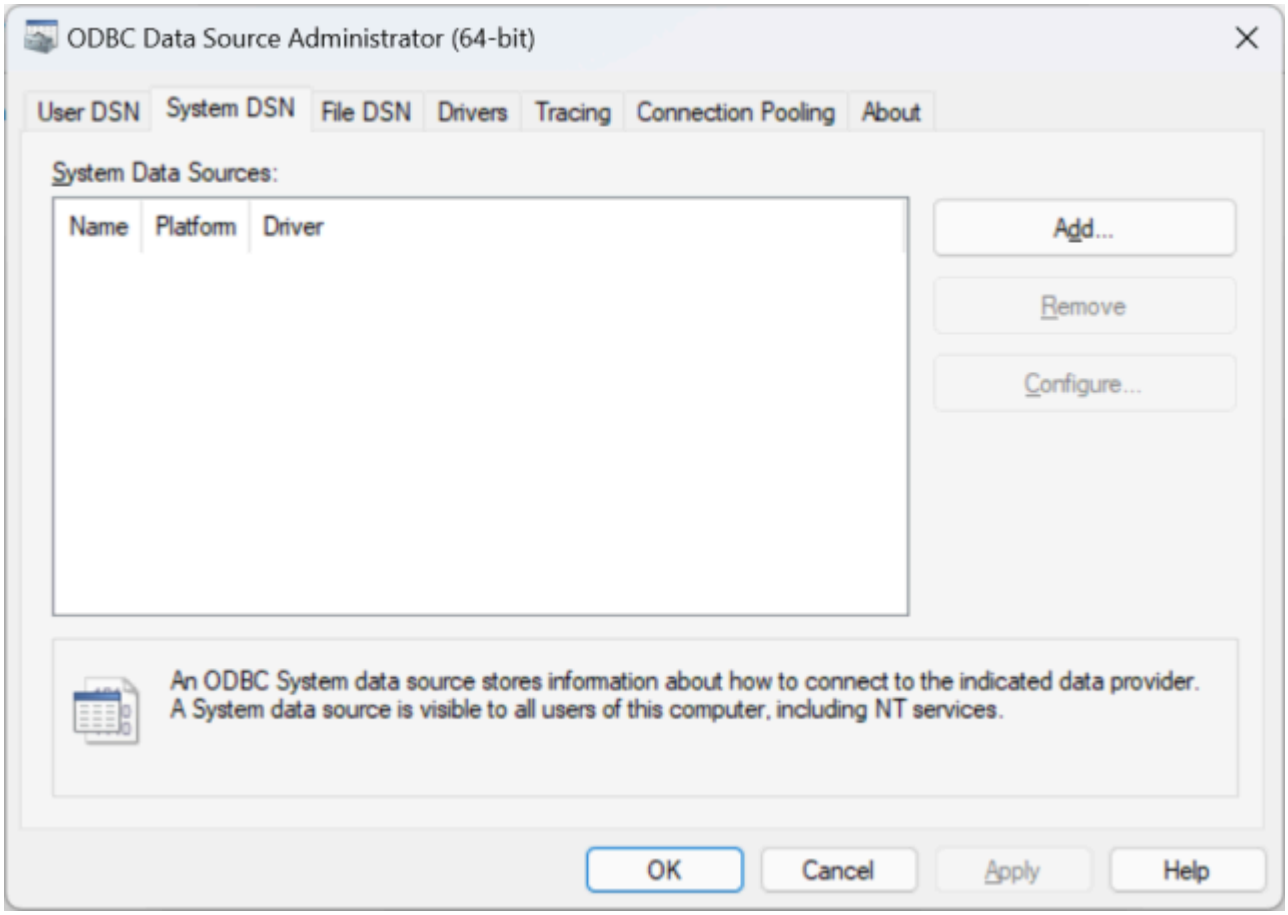
윈도우에서 Amazon DocumentDB ODBC 드라이버 설정하기

다음 절차에 따라 Windows에서 Amazon DocumentDB ODBC 드라이버를 설정합니다.

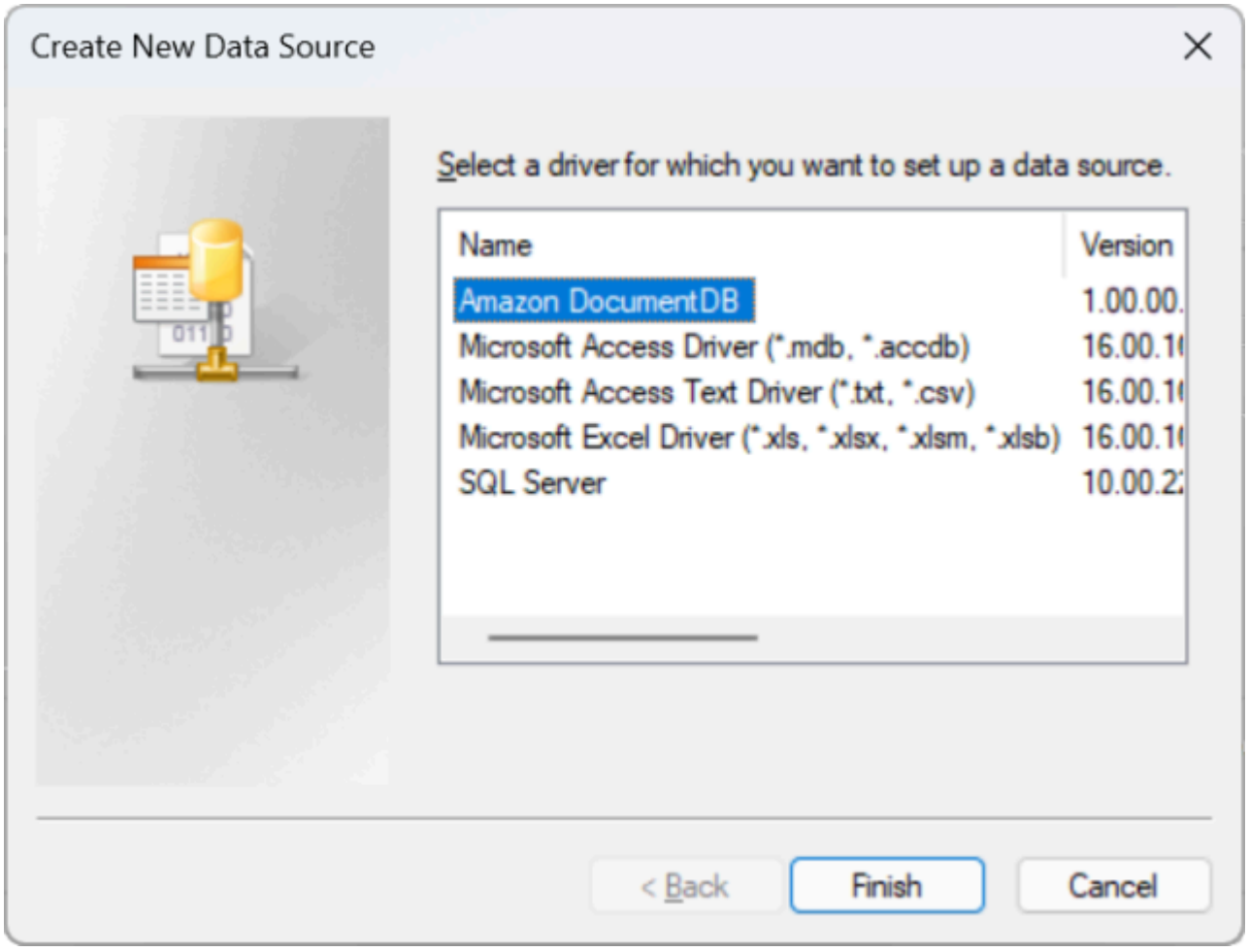
1. Windows에서 제어판을 열고 ODBC를 검색합니다 (또는 메뉴에서 Windows 도구 > ODBC 데이터 소스 (32비트) 또는 ODBC 데이터 소스 (64비트) 를 선택).



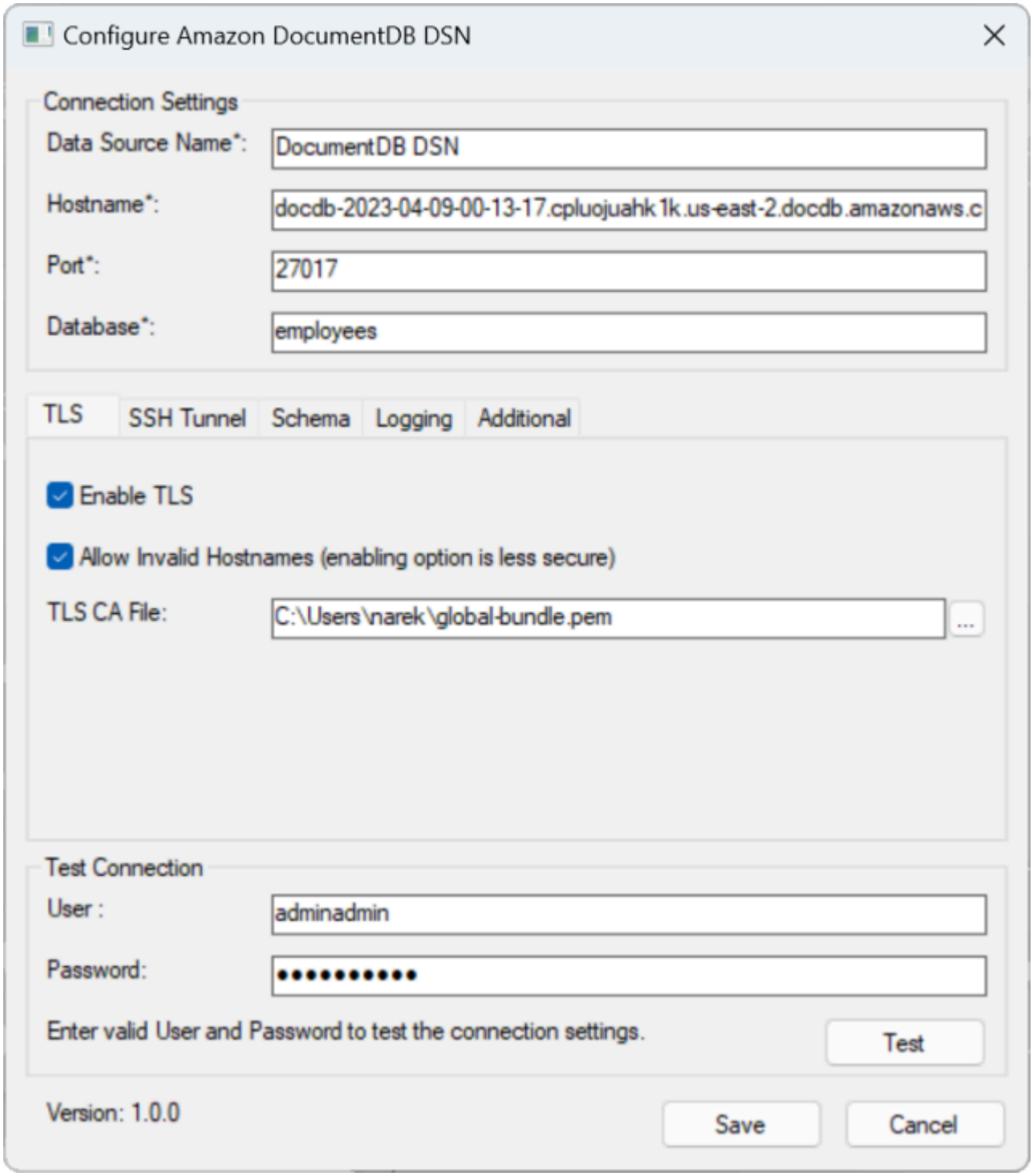
2. 적절한 ODBC 드라이버 데이터 원본 관리자 선택: 설치되어 있는 경우 32비트 버전을 선택하고, 그렇지 않으면 64비트 버전을 선택하세요.
3. 시스템 DSN 탭을 선택한 다음 추가... 를 클릭합니다. 새 DSN을 추가하려면



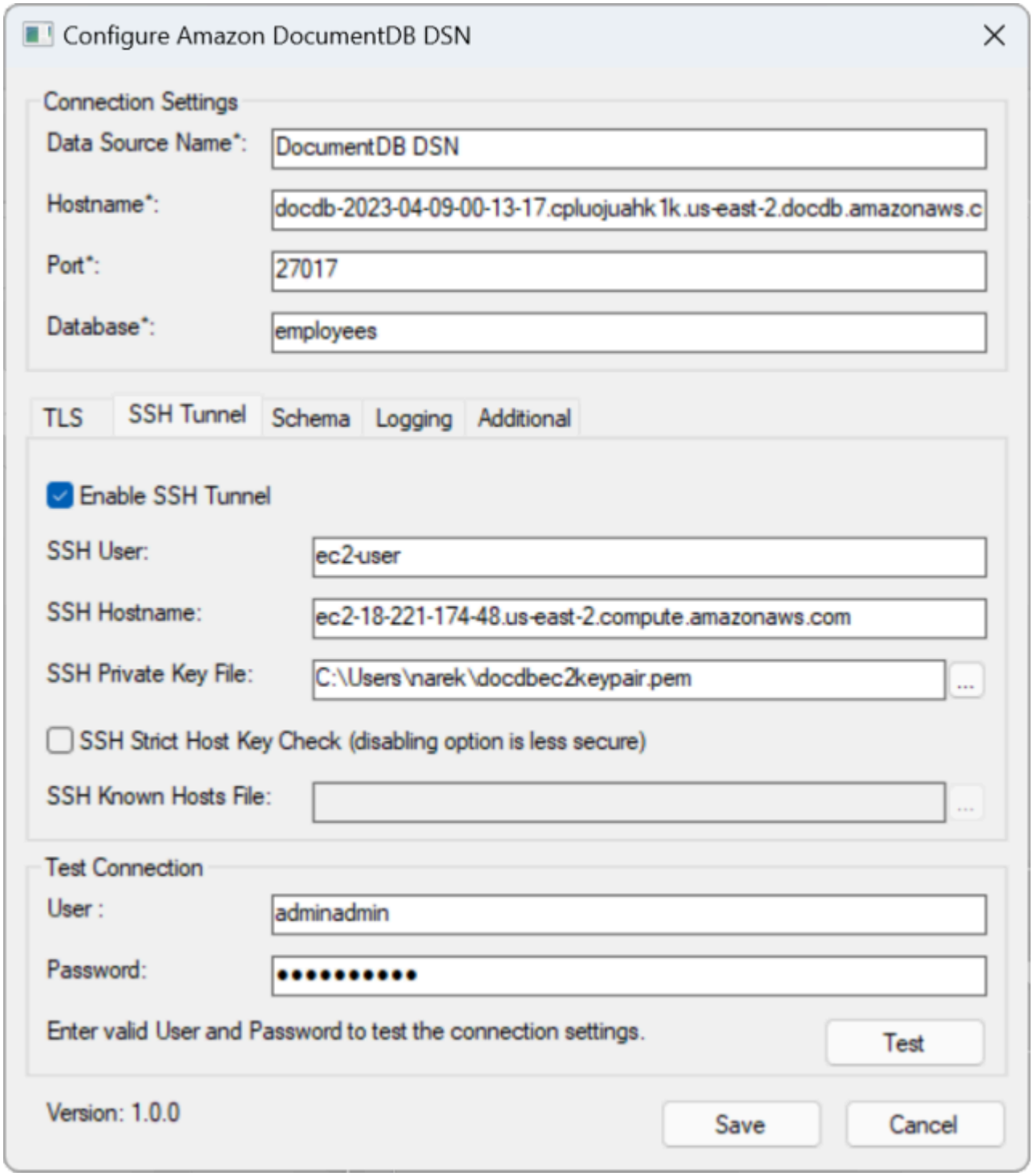
4. 데이터 소스 드라이버 목록에서 Amazon DocumentDB를 선택합니다.



5. Amazon DocumentDB DSN 구성 대화 상자에서 연결 설정, TLS 탭 및 연결 테스트 필드를 완료한 다음 저장을 클릭합니다.



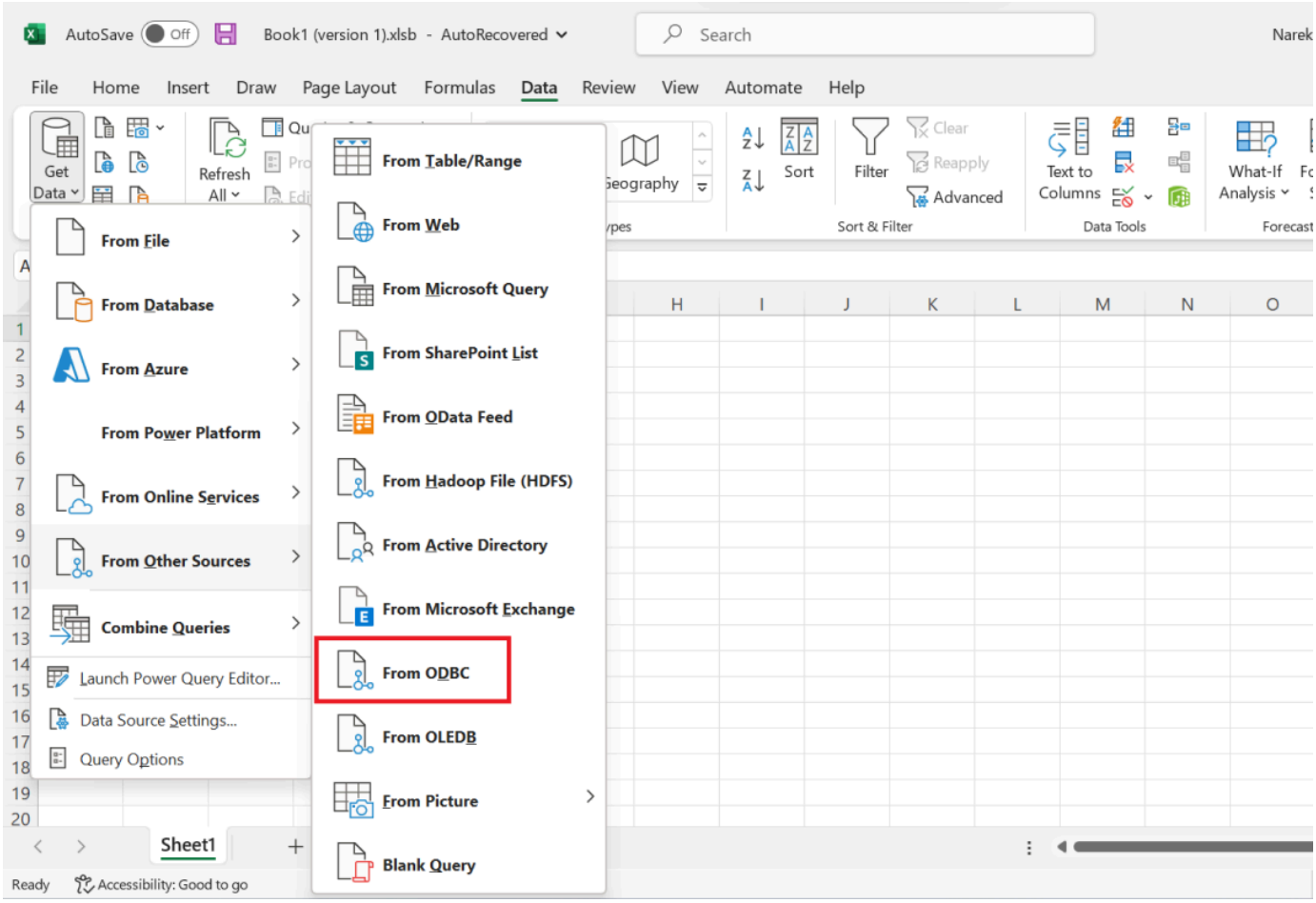
6. 선택한 EC2 인스턴스에 대한 SSH 터널링 방법에 따라 연결 세부 정보가 달라지므로 Windows 양식을 정확하게 작성해야 합니다. 여기에서 SSH 터널링 방법을 참조하세요. 각 속성에 대한 자세한 내용은 연결 문자열 구문 및 옵션을 참조하세요.



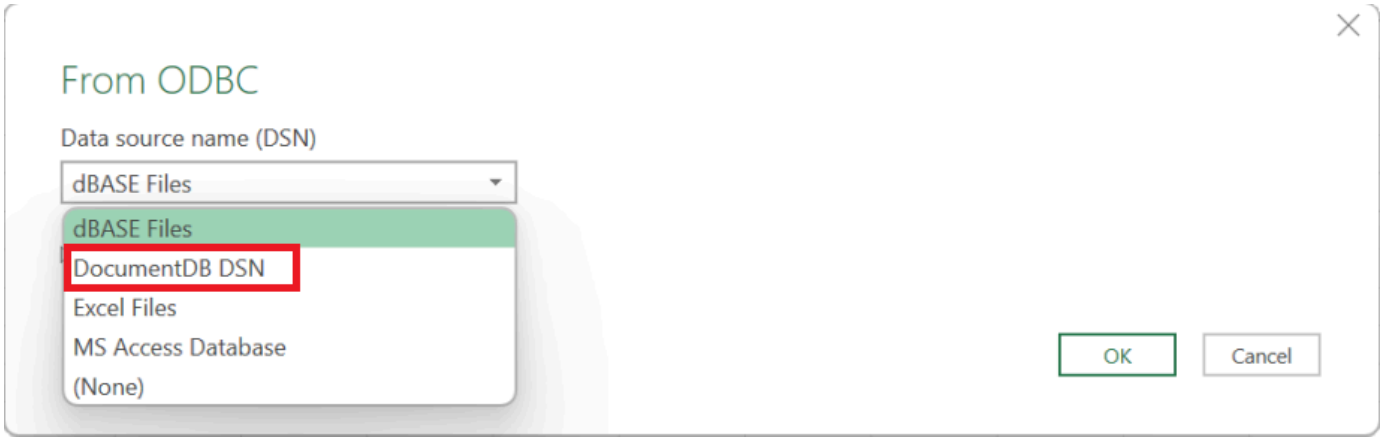
Windows에서 Amazon DocumentDB ODBC 드라이버를 구성하는 방법에 대한 자세한 내용을 보려면 여기를 클릭하세요.

마이크로소프트 엑셀에서 Amazon DocumentDB에 연결

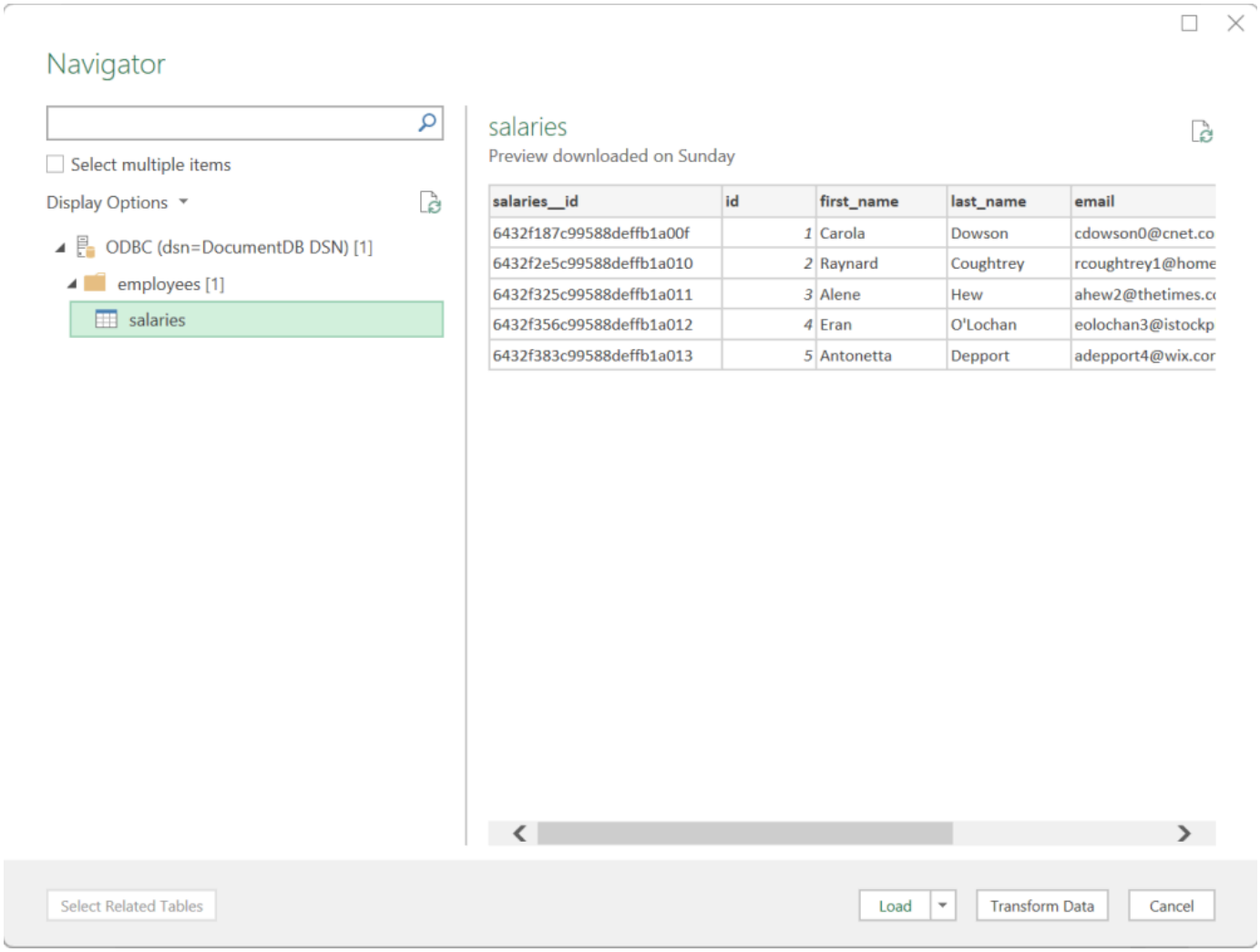
1. Amazon DocumentDB 드라이버가 올바르게 설치 및 구성되었는지 확인하세요. 자세한 내용은 Windows에서 ODBC 드라이버 설정을 참조하세요.
2. 마이크로소프트 엑셀을 실행합니다.
3. 데이터 > 데이터 가져오기 > 기타 소스에서로 이동합니다.
4. ODBC에서 선택:



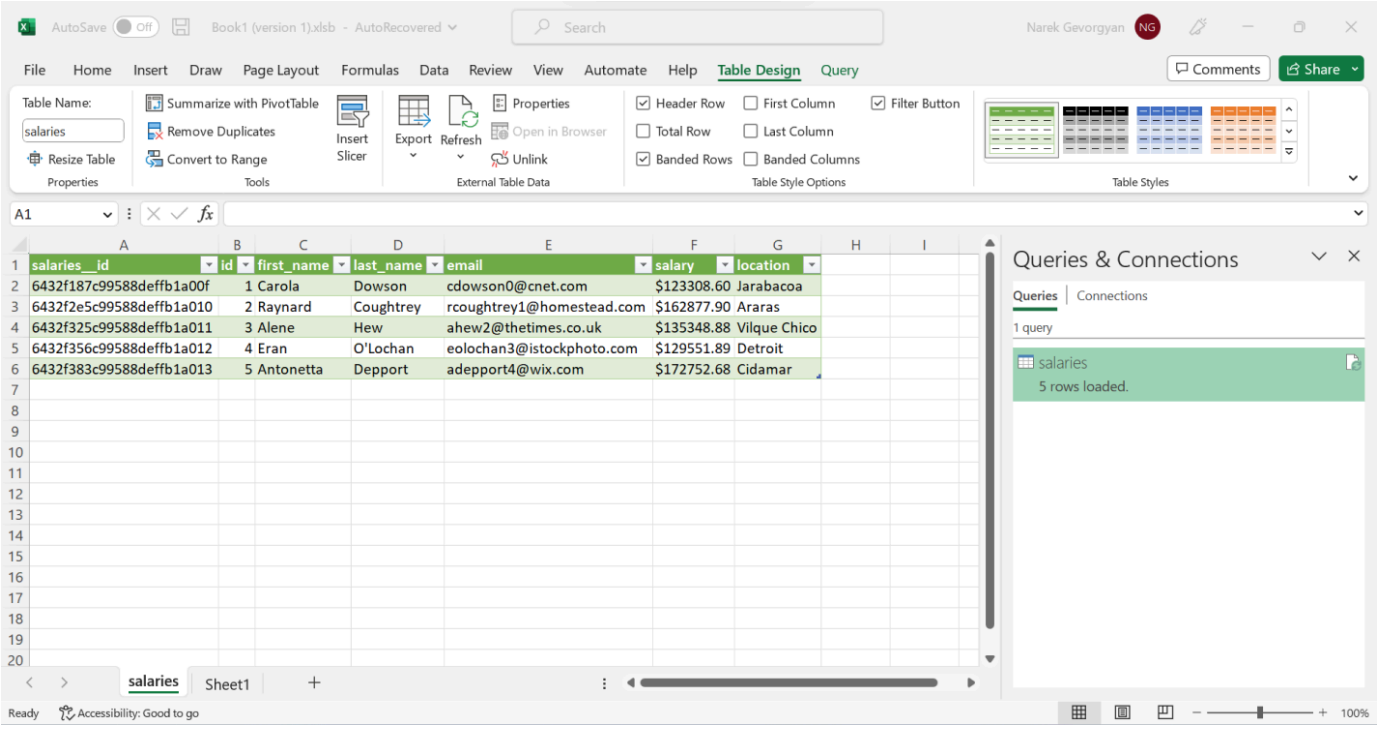
5. Amazon DocumentDB와 연결된 데이터 소스 이름 (DSN) 드롭다운 메뉴에서 데이터 소스를 선택합니다.



6. 데이터를 Excel에 로드할 컬렉션을 선택합니다.



7. 데이터를 Excel에 로드합니다.



마이크로소프트 파워 BI 데스크톱에서 Amazon DocumentDB에 연결

주제

- 사전 조건
- Microsoft Power BI 데스크톱 사용자 지정 커넥터 추가
- Amazon DocumentDB 사용자 지정 커넥터를 사용하여 연결
- 마이크로소프트 파워 BI 게이트웨이 구성

사전 조건

시작하기 전에 Amazon DocumentDB ODBC 드라이버가 올바르게 설치되어 있는지 확인하세요.

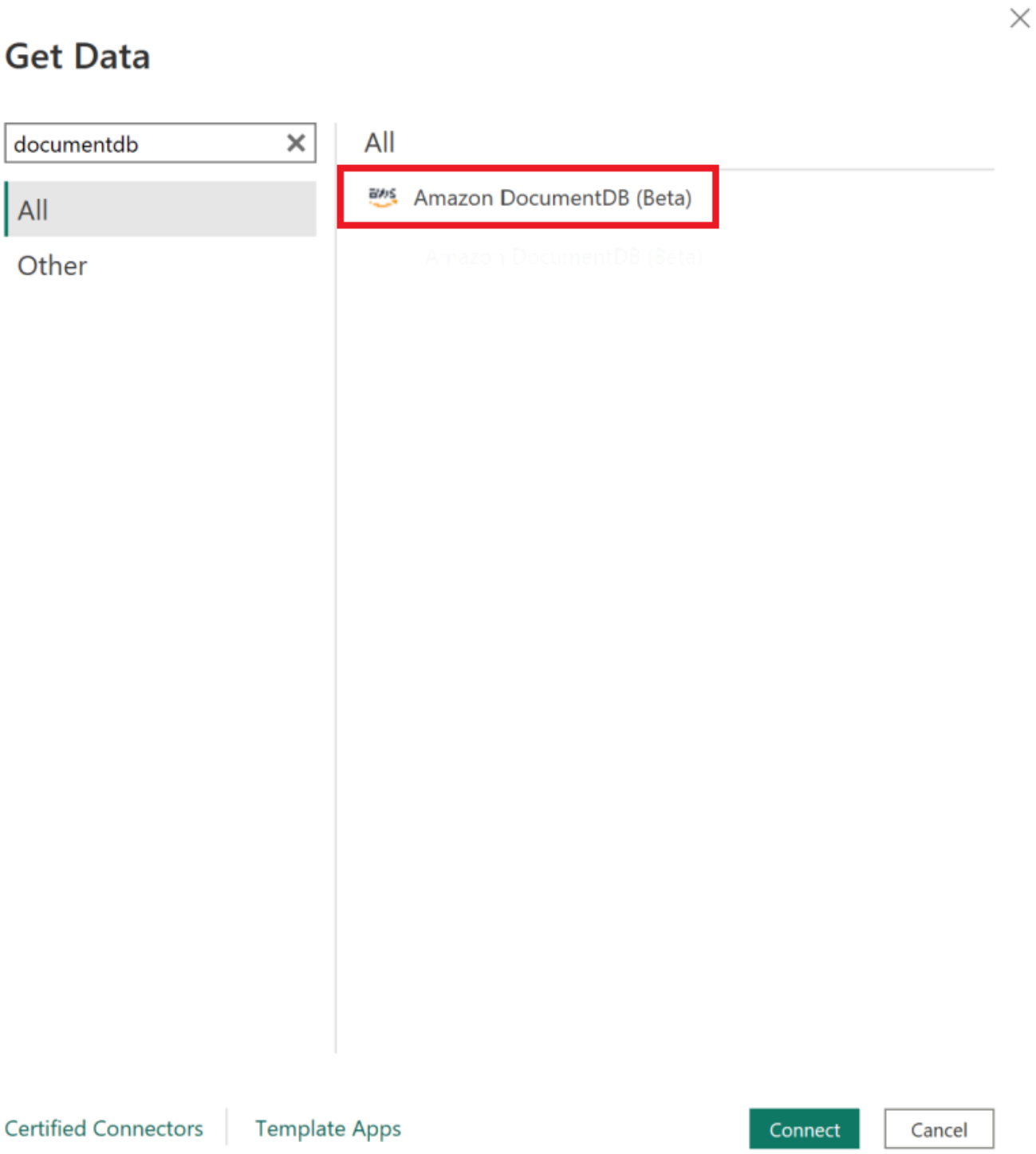
Microsoft Power BI 데스크톱 사용자 지정 커넥터 추가

`AmazonDocumentDBConnector.mez`파일을 `<User>\Documents\Power BI Desktop\Custom Connectors\` 폴더 (또는 OneDrive를 사용하는 `<User>\OneDrive\Documents\Power BI Desktop\Custom Connectors` 경우) 에 복사합니다. 이렇게 하면 Power BI에서 사용자 지정 커넥터에 액세스할 수 있습니다. Power BI 데스크톱에 대한 커넥터는 여기에서 구할 수 있습니다. Power BI Desktop을 다시 시작하여 커넥터가 로드되었는지 확인합니다.

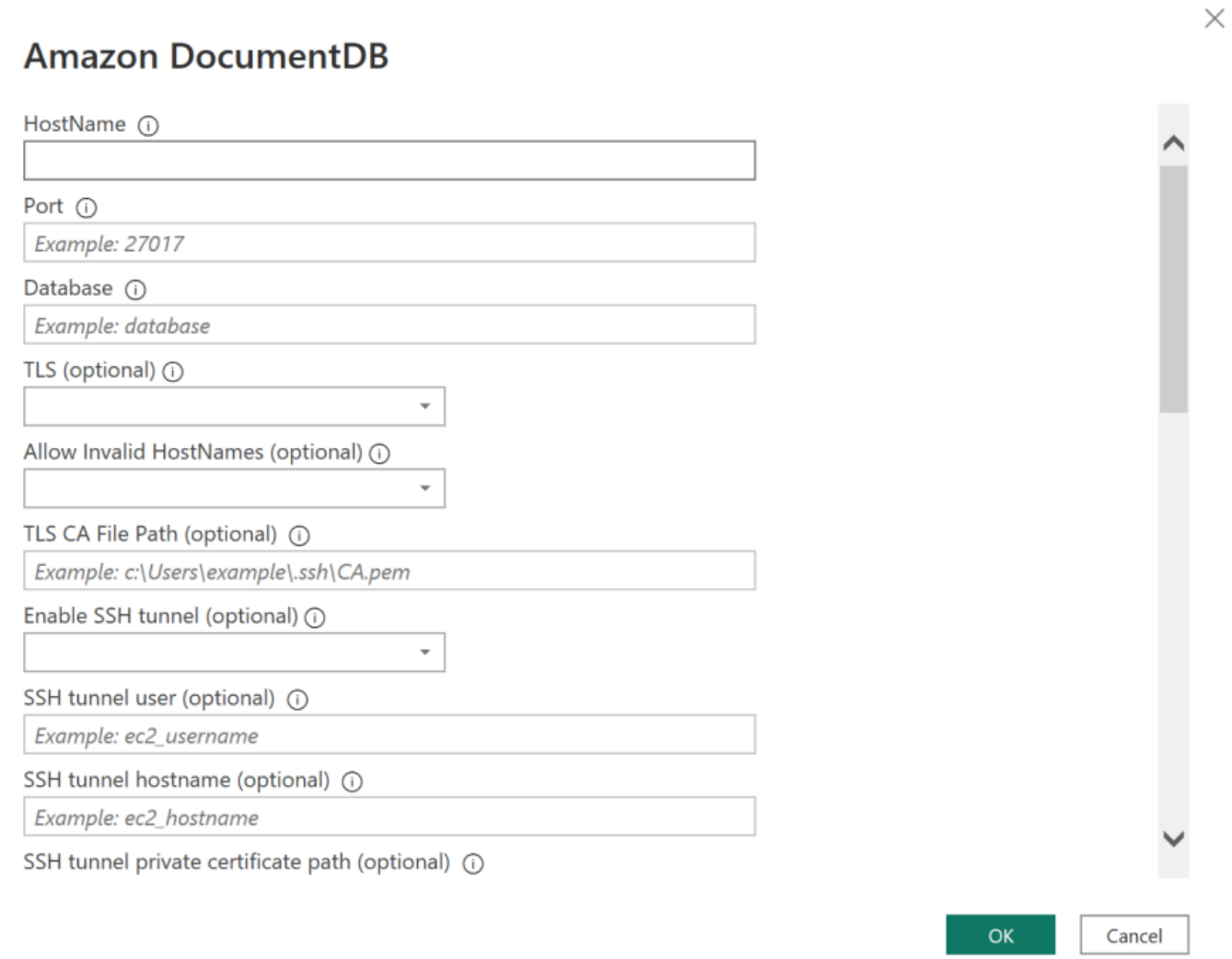
> **Note**
>
> 사용자 지정 커넥터는 인증을 위한 Amazon DocumentDB 사용자 이름 및 암호만 지원합니다.

Amazon DocumentDB 사용자 지정 커넥터를 사용하여 연결

1. 데이터 가져오기에서 Amazon DocumentDB (베타) 를 선택하고 연결을 클릭합니다. 타사 서비스 사용에 대한 경고가 표시되면 계속을 클릭하세요.



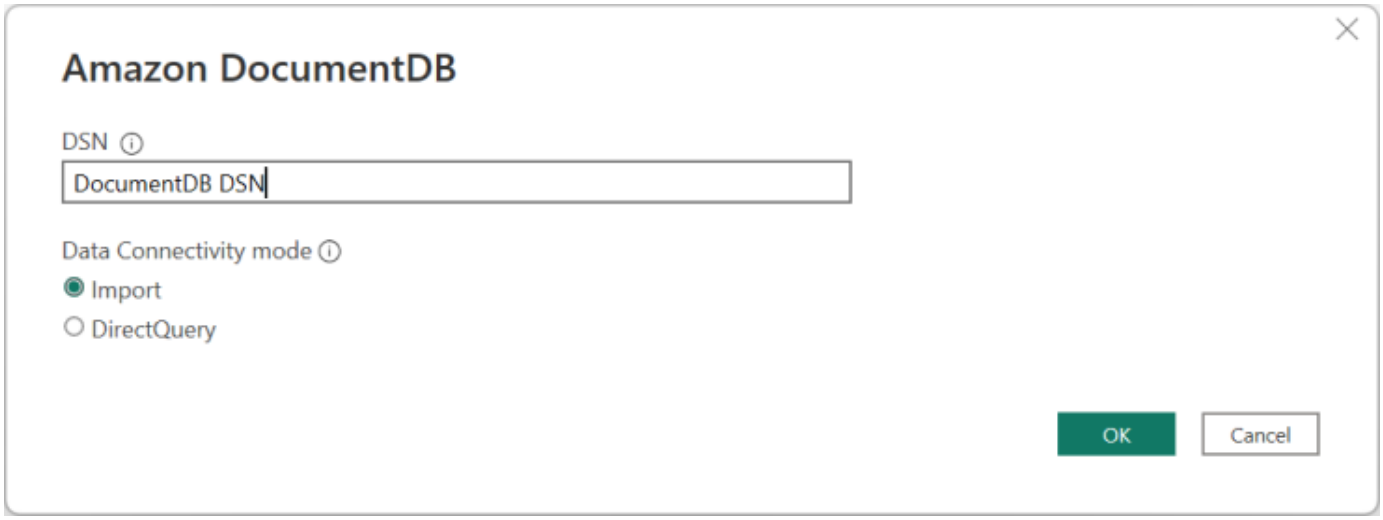
2. Amazon DocumentDB 클러스터에 연결하는 데 필요한 모든 정보를 입력한 다음 확인을 클릭합니다.



> **Note**
>
> DSN 설정에 필요한 정보를 이미 제공한 경우 ODBC 드라이버의 데이터 소스 이름 (DSN) 구성에 따라 SSH 연결 세부 정보 화면이 표시되지 않을 수 있습니다.

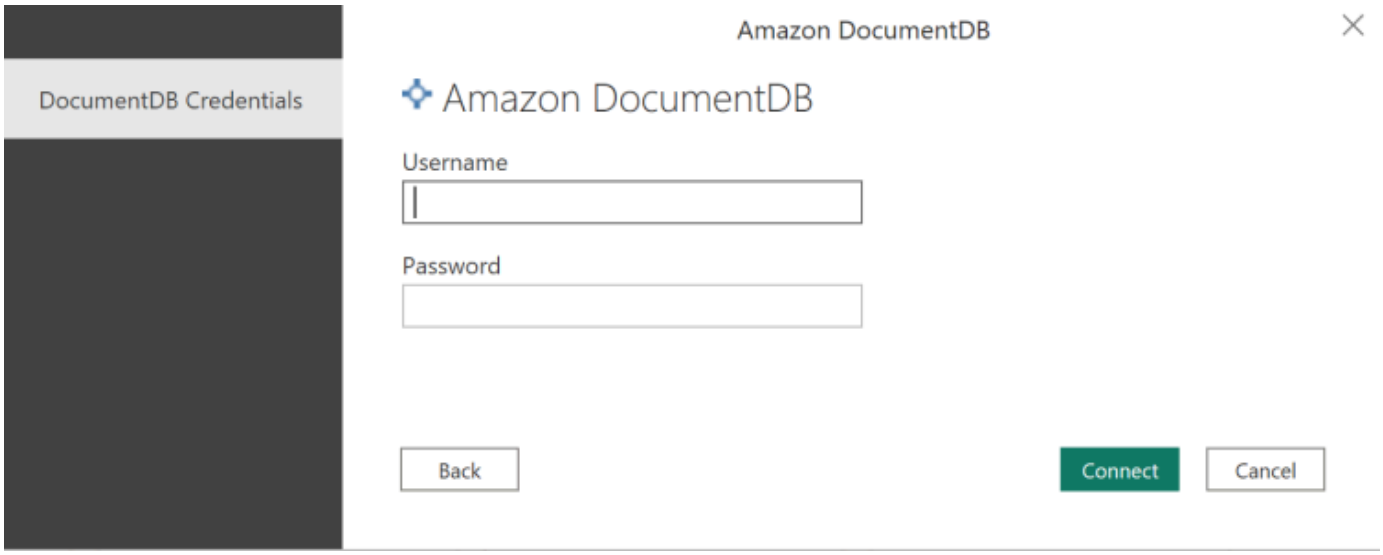
3. 데이터 연결 모드를 선택합니다.

   - 가져오기 - 모든 데이터를 로드하고 정보를 디스크에 저장합니다. 데이터 업데이트를 표시하려면 데이터를 새로 고치고 다시 로드해야 합니다.
   - 직접 쿼리 - 데이터를 로드하지는 않지만 데이터에 대한 실시간 쿼리를 수행합니다. 즉, 데이터 업데이트를 표시하기 위해 데이터를 새로 고치고 다시 로드할 필요가 없습니다.

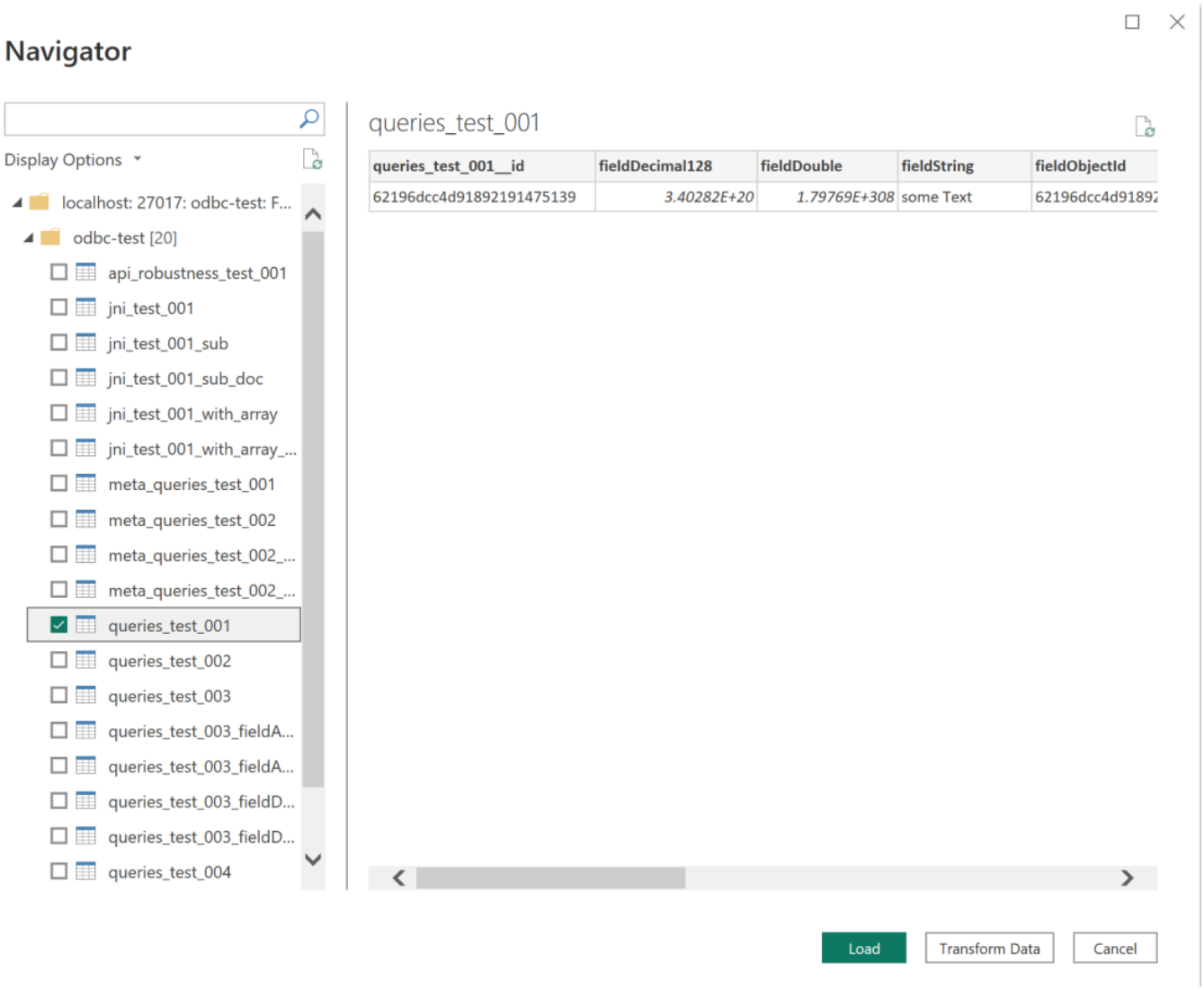


> **Note**
>
> 매우 큰 데이터셋을 사용하는 경우 모든 데이터를 가져오는 데 시간이 더 오래 걸릴 수 있습니다.

4. 이 데이터 원본에 처음 연결하는 경우 인증 유형을 선택하고 메시지가 표시되면 자격 증명을 입력합니다. 그런 다음 연결을 클릭합니다:



5. 네비게이터 대화 상자에서 원하는 데이터베이스 테이블을 선택한 다음 로드를 클릭하여 데이터를 로드하거나 데이터 변환을 클릭하여 데이터 변환을 계속합니다.



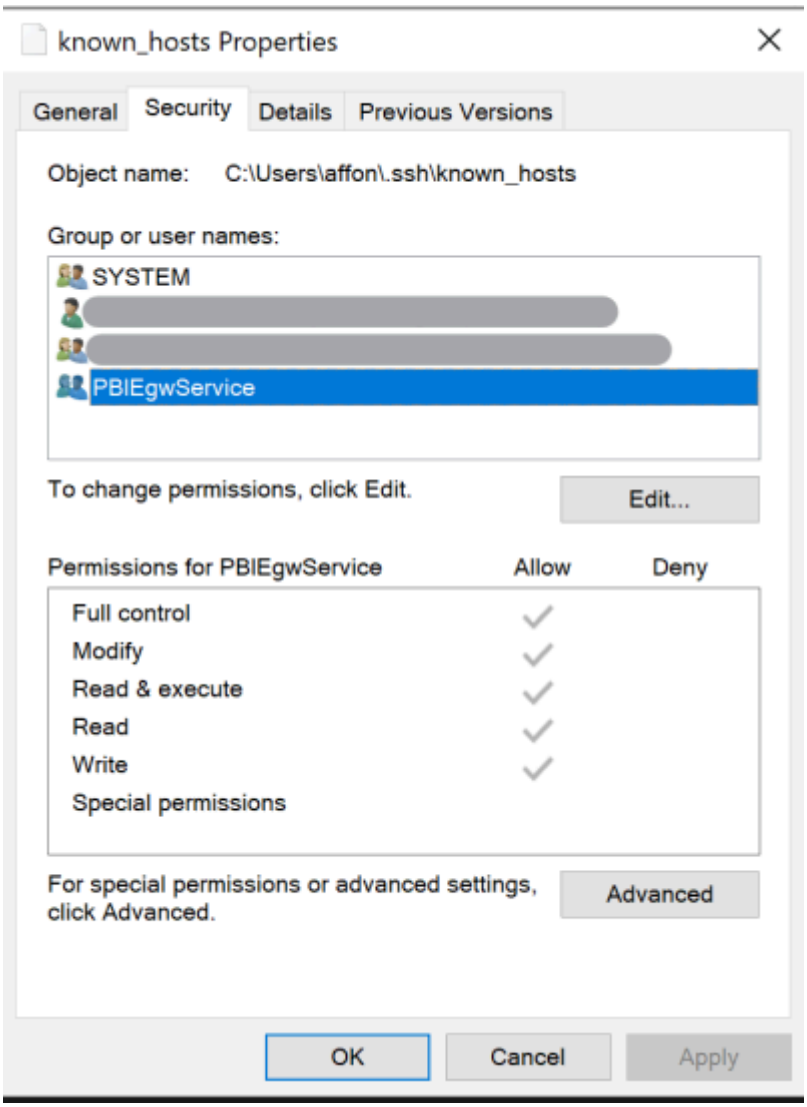
> **Note**
>
> 연결하면 데이터 소스 설정이 저장됩니다. 설정을 수정하려면 데이터 변환 > 데이터 소스 설정을 선택합니다.

마이크로소프트 파워 BI 게이트웨이 구성

사전 조건:

- 사용자 지정 커넥터가 Power BI 게이트웨이와 호환되는지 확인하세요.
- Power BI Gateway가 설치된 컴퓨터의 시스템 탭에 있는 ODBC 데이터 원본에 ODBC DSN이 생성되었는지 확인하세요.

내부 SSH 터널 기능을 사용하는 경우 파일은 known_hosts Power BI 서비스 계정이 액세스할 수 있는 위치에 있어야 합니다.



> **Note**
>
> 이는 인증 기관 (CA) 인증서 파일 (pem 파일) 과 같이 Amazon DocumentDB 클러스터에 대한 연결을 설정하는 데 필요한 모든 파일에도 적용됩니다.

자동 스키마 생성

ODBC 드라이버는 JNI (자바 네이티브 인터페이스) 를 통해 Amazon DocumentDB JDBC 드라이버를 활용하므로 자동 스키마 생성 기능이 JDBC 드라이버에서도 비슷하게 작동합니다. 자동 스키마 생성에

대한 자세한 내용은 JDBC 자동 스키마 생성을 참조하세요. 또한 ODBC 드라이버 아키텍처에 대해 자세히 알아보려면 여기를 클릭하세요.

SQL 지원 및 제한 사항

Amazon DocumentDB ODBC 드라이버는 SQL-92 하위 집합과 일부 일반 확장을 지원하는 읽기 전용 드라이버입니다. 자세한 내용은 ODBC 지원 및 제한 설명서를 참조하세요.

문제 해결

Amazon DocumentDB ODBC 드라이버 사용에 문제가 있는 경우 문제 해결 안내서를 참조하세요.

Amazon DocumentDB를 사용한 프로그램

서비스는 JSON 스키마 검증을 지원합니다.

주제

- Amazon DocumentDB Java 프로그래밍 가이드
- JSON 스키마 검증 사용

Amazon DocumentDB Java 프로그래밍 가이드

이 포괄적인 가이드는 MongoDB의 Java 드라이버를 사용하여 Amazon DocumentDB로 작업하기 위한 자세한 안내를 제공하며, 데이터베이스 운영 및 관리의 필수 측면을 다룹니다.

주제

- 소개
- 사전 조건
- 이 가이드의 데이터 모델
- MongoDB Java 드라이버를 사용하여 Amazon DocumentDB에 연결
- Java를 사용하여 Amazon DocumentDB에서 CRUD 작업 수행
- Java를 사용한 Amazon DocumentDB의 인덱스 관리
- Amazon DocumentDB 및 Java를 사용한 이벤트 기반 프로그래밍

소개

이 가이드는 MongoDB Java 드라이버를 사용하여 DocumentDB 클러스터에 대한 보안 연결을 설정하는 방법을 설명하는 연결로 시작합니다. 연결 문자열 구성 요소, SSL/TLS 구현, IAM 인증 및 연결 풀링을 비롯한 다양한 연결 옵션과 강력한 오류 처리 전략을 자세히 설명합니다.

CRUD(생성, 읽기, 업데이트, 삭제) 작업 섹션에서 가이드는 문서 조작을 철저히 다루며, 단일 작업과 대량 작업을 모두 사용하여 문서를 생성, 읽기, 업데이트 및 삭제하는 방법을 보여줍니다. 필터, 쿼리 및 다양한 작업 옵션의 사용에 대해 설명하고 오류 처리에 대한 모범 사례를 강조하고 신뢰성 향상을 위한 재시도 로직을 구현합니다. 또한 단일 필드, 복합, 희소 및 텍스트 인덱스를 포함한 다양한 인덱스 유형의 생성 및 유지 관리를 자세히 설명하는 인덱스 관리를 광범위하게 다룹니다. 적절한 인덱스 선택 및 `explain()` 함수 사용을 통해 쿼리 실행 계획을 분석하여 쿼리 성능을 최적화하는 방법을 설명합니다.

마지막 섹션에서는 Amazon DocumentDB의 변경 스트림을 사용한 이벤트 기반 프로그래밍에 중점을 두고 Java 애플리케이션에서 실시간 데이터 변경 모니터링을 구현하는 방법을 보여줍니다. 여기에는 변경 스트림 커서 구현, 연속 작업을 위한 재개 토큰 처리, 과거 데이터 처리를 위한 시간 기반 작업이 포함됩니다. 가이드 전체에서 실용적인 코드 예제와 모범 사례가 제공되므로 Amazon DocumentDB로 강력한 Java 애플리케이션을 구축할 때 매우 유용한 리소스가 됩니다.

사전 조건

시작하기 전에 다음이 있는지 확인합니다.

- 구성된 DocumentDB 클러스터가 있는 AWS 계정입니다. DocumentDB 클러스터 설정에 대한이 시작하기 블로그 게시물을 참조하세요.
- Java Development Kit(JDK)가 설치되었습니다(이 가이드에서는 Amazon Corretto 21을 사용할 예정입니다).
- 종속성 관리를 위한 Maven.

이 가이드의 데이터 모델

이 가이드의 모든 예제 코드는 “Restaurants” 컬렉션이 있는 “ProgGuideData” 테스트 데이터베이스에 대한 연결을 가정합니다. 이 가이드의 모든 샘플 코드는 레스토랑 목록 시스템 이하에서 작동합니다.이 시스템의 문서는 다음과 같습니다.

```
{
    "_id": "ab6ad8f119b5bca3efa2c7ae",
```

```
    "restaurantId": "REST-CRT9BL",
    "name": "Thai Curry Palace",
    "description": "Amazing Restaurant, must visit",
    "cuisine": "Thai",
    "address": {
        "street": "914 Park Street",
        "city": "Bryan",
        "state": "AL",
        "zipCode": "96865",
        "location": {
            "type": "Point",
            "coordinates": [-25.4619, 8.389]
        }
    },
    "contact": {
        "phone": "(669) 915-9056 x6657"
    },
    "rating": {
        "average": 3.4,
        "totalReviews": 275
    },
    "priceRange": "$",
    "menu": [{
        "category": "Appetizers",
        "items": [{
            "name": "Buffalo Chicken Wings",
            "price": 13.42
        }]
    }],
    "features": [
        "Private Dining"
    ],
    "isActive": false“ michelin”: {“
        star”: 3,
        “ranking_years”: 4
    }
}
```

CRUD, 인덱스 관리 및 이벤트 기반 프로그래밍을 보여주는 모든 코드 샘플은 MongoClient 객체 , dbClient MongoDatabase 객체 및 MongoCollection 객체 connectionDB가 있다고 가정합니다collection.

> **Note**
>
> 이 가이드의 모든 코드 예제는 MongoDB Java 드라이버 버전 5.3.0으로 테스트되었습니다.

MongoDB Java 드라이버를 사용하여 Amazon DocumentDB에 연결

이 섹션에서는 Java 드라이버를 사용하여 Amazon DocumentDB에 연결하기 위한 step-by-step 가이드를 제공합니다. 그러면 DocumentDB를 Java 애플리케이션에 통합하는 작업을 시작할 수 있습니다.

주제

- 1단계: 프로젝트 설정
- 2단계: 연결 문자열 생성
- 3단계: 연결 코드 작성
- 4단계: 연결 예외 처리
- 5단계: 코드 실행
- 연결 모범 사례

1단계: 프로젝트 설정

1. Maven을 사용하여 java 프로젝트를 생성합니다.

```
mvn archetype:generate -DgroupId=com.docdb.guide -DartifactId=my-docdb-project -
DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
```

2. MongoDB Java 드라이버를 'pom.xml' 파일의 프로젝트에 대한 종속성으로 추가합니다.

```
<dependency>
    <groupId>org.mongodb</groupId>
    <artifactId>mongodb-driver-sync</artifactId>
    <version>5.3.0</version>
</dependency>
```

2단계: 연결 문자열 생성

Amazon DocumentDB 연결 문자열은 애플리케이션과 DocumentDB 클러스터 간에 연결을 설정하는 데 필수적입니다. 이 문자열은 클러스터 엔드포인트, 포트, 인증 세부 정보 및 다양한 연결 옵션과 같은

중요한 정보를 캡슐화합니다. DocumentDB 연결 문자열을 빌드하려면 일반적으로 기본 형식으로 시작합니다.

```
"mongodb://username:password@cluster-endpoint:port/?[connection options]"
```

"사용자 이름" 및 "암호"를 실제 자격 증명으로 바꿔야 합니다. AWS Management Console 뿐만 아니라 에서도 클러스터의 엔드포인트와 포트 번호를 찾을 수 있습니다 AWS CLI. 클러스터의 클러스터 엔드포인트를 찾으클러스터 엔드포인트 찾기려면 섹션을 참조하세요. DocumentDB의 기본 포트는 27017입니다.

연결 문자열 예제

- 전송 중 암호화를 사용하여 DocumentDB에 연결하고 읽기 요청이 읽기 전용 복제본으로 이동하고 프라이머리에 쓰기를 수행하도록 합니다.

```
"mongodb://username:password@cluster-endpoint:27017/?tls=true&
   tlsCAFile=global-bundle.pem&
   readPreference=secondaryPreferred&
   retryWrites=false"
```

- IAM 인증을 사용하여 DocumentDB에 연결:

```
"mongodb://cluster-endpoint:27017/?tls=true&
   tlsCAFile=global-bundle.pem&
   readPreference=secondaryPreferred&
   retryWrites=false&
   authSource=%24external&
   authMechanism=MONGODB-AWS"
```

연결 문자열에 사용할 수 있는 다양한 옵션은 다음과 같습니다.

- TLS 인증서
- 읽기 전용 복제본에서 읽기
- 문제 작성 및 저널링
- RetryWrites
- IAM 인증.
- 연결 풀
- 연결 제한 시간 파라미터

TLS 인증서

tls=true|false -이 옵션은 전송 계층 보안(TLS)을 활성화하거나 비활성화합니다. 기본적으로 Amazon DocumentDB 클러스터에서 전송 중 암호화가 활성화되므로 클러스터 수준에서 TLS가 비활성화되지 않는 한이 옵션의 값은 여야 합니다`true`.

TLS를 사용하는 경우 코드는 DocumentDB 클러스터에 대한 연결을 생성할 때 SSL 인증서를 제공해야 합니다. 클러스터에 안전하게 연결하는 데 필요한 인증서를 다운로드합니다`global-bundle.pem`. `global-bundle.pem` 파일을 사용하는 방법에는 두 가지가 있습니다.

- 옵션 1 - `global-bundle.pem` 파일에서 모든 인증서를 추출하고 Java의 keytool을 사용하여 나중에 코드에서 사용할 수 있는 `.jks` 파일에 저장합니다. 이 작업을 수행하는 방법을 보여주는 스크립트는 TLS가 활성화된 상태에서 연결의 Java 탭을 참조하세요.
- 옵션 2 - 코드에 `global-bundle.pem` 파일을 동적으로 추가하고, 인 메모리 키 스토어를 빌드하고, `SSLContext`를 사용하여 연결의 일부로 인증서를 제공합니다.

읽기 전용 복제본에서 읽기

replicaSet=rs0&readPreference=secondaryPreferred -이 두 옵션을 지정하면 모든 읽기 요청이 읽기 전용 복제본으로 라우팅되고 쓰기 요청이 기본 인스턴스로 라우팅됩니다. 연결 문자열`replicaSet=rs0`에서를 사용하면 MongoDB 드라이버가 클러스터 토폴로지의 자동 업데이트된 보기를 유지하여 인스턴스가 추가되거나 제거될 때 애플리케이션이 현재 노드 구성의 가시성을 유지할 수 있습니다. 이러한 옵션을 제공하지 않거나를 지정하지 않으면 모든 읽기 및 쓰기가 기본 인스턴스로 `readPreference=primary` 전송됩니다. 에 대한 추가 옵션은 섹션을 `readPreference`참조하세요읽기 기본 설정 옵션.

문제 작성 및 저널링

쓰기 문제에 따라 쓰기 작업을 위해 데이터베이스에서 요청된 승인 수준이 결정됩니다. MongoDB 드라이버는 쓰기 문제 및 저널 파일을 조정할 수 있는 옵션을 제공합니다. Amazon DocumentDB는 쓰기 문제 및 저널을 설정할 것으로 예상하지 않으며 및 `w` `j` (`writeConcern` 및)에 대해 전송된 값을 무시합니다`journal`. DocumentDB는 항상 `writeConcern: majority` 및 `journal`:를 사용하여 데이터를 기록`true`하므로 클라이언트에 승인을 보내기 전에 대부분의 노드에 쓰기가 내구성 있게 기록됩니다.

RetryWrites

retryWrites=false - DocumentDB는 재시도 가능한 쓰기를 지원하지 않으므로이 속성은 항상 로 설정해야 합니다`false`.

IAM 인증.

authSource=%24external 및 **authMechanism=MONGODB-AWS** -이 두 파라미터는를 사용하여 인증하는 데 사용됩니다 AWS Identity and Access Management. IAM 인증은 현재 인스턴스 기반 클러스터 버전 5.0에서만 사용할 수 있습니다. 자세한 내용은 IAM 자격 증명을 사용한 인증 단원을 참조하십시오.

연결 풀

연결 풀링에 사용할 수 있는 옵션은 다음과 같습니다.

- **maxPoolSize** - 풀에서 생성할 수 있는 최대 연결 수를 설정합니다. 모든 연결이 사용 중이고 새 요청이 들어오면 연결을 사용할 수 있을 때까지 기다립니다. MongoDB Java 드라이버의 기본값은 100입니다.
- **minPoolSize** - 풀에서 항상 유지해야 하는 최소 연결 수를 나타냅니다. MongoDB Java 드라이버의 기본값은 0입니다.
- **maxIdleTimeMS** - 연결을 닫고 제거하기 전에 풀에서 연결을 유휴 상태로 유지할 수 있는 기간을 결정합니다. MongoDB Java 드라이버의 기본값은 100밀리초입니다.
- **waitQueueTimeoutMS** - 풀이 최대 크기일 때 스레드가 연결을 사용할 수 있을 때까지 기다려야 하는 시간을 구성합니다. 이 시간 내에 연결을 사용할 수 없는 경우 예외가 발생합니다. MongoDB Java 드라이버의 기본값은 120,000밀리초(2분)입니다.

연결 제한 시간 파라미터

제한 시간은 작업 또는 연결 시도가 실패로 간주되기까지 걸릴 수 있는 시간을 제한하는 메커니즘입니다. 무기한 대기를 방지하고 리소스 할당을 관리하는 데 사용할 수 있는 제한 시간 파라미터는 다음과 같습니다.

- **connectTimeoutMS** - 드라이버가 클러스터에 대한 연결을 설정하기 위해 대기하는 시간을 구성합니다. 기본값은 10,000밀리초(10초)입니다.
- **socketTimeoutMS** - 드라이버가 비쓰기 작업을 위해 서버의 응답을 기다리는 시간을 지정합니다. 기본값은 0입니다(제한 시간 없음 또는 무한).
- **serverSelectionTimeoutMS** - 드라이버가 클러스터에서 사용 가능한 서버를 찾기 위해 대기하는 시간을 지정합니다. 이 설정의 기본값은 30초이며 장애 조치 중에 새 기본 인스턴스를 선택하기에 충분합니다.

3단계: 연결 코드 작성

다음 코드 예제에서는 Amazon DocumentDB에 TLS를 연결하는 방법을 보여줍니다.

- Java KeyStore 및 SSLContext> 객체를 생성합니다.
- 또한 MongoClientSettings 객체에 전달하여 ConnectionString 객체를 생성합니다. TLS 연결을 만들려면 MongoClientSettings 객체를 사용하여 connectionstring 및를 바인딩해야 합니다sslcontext.
- 를 사용하면 MongoClient 객체를 MongoClients 가져옵니다.

```
public static MongoClient makeDbConnection(String dbName, String DbUserName, String
 DbPassword,
    String DbClusterEndPoint, String keyStorePass) throws Exception {
    MongoClient connectedClient;
    String connectionOptions = "?
replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false";
    String connectionUrl = "mongodb://" + DbUserName + ":" + DbPassword + "@" +
 DbClusterEndPoint + ":27017/" +
        dbName + connectionOptions;

    try {
        KeyStore trustStore = KeyStore.getInstance(KeyStore.getDefaultType());
        try (FileInputStream fis = new FileInputStream("src/main/resources/certs/
truststore.jks")) {
            trustStore.load(fis, keyStorePass.toCharArray());
            TrustManagerFactory tmf =
 TrustManagerFactory.getInstance(TrustManagerFactory.getDefaultAlgorithm());
            tmf.init(trustStore);

            SSLContext sslContext = SSLContext.getInstance("TLSv1.2");
            sslContext.init(null, tmf.getTrustManagers(), new SecureRandom());
            ConnectionString connectionString = new ConnectionString(connectionUrl);
            MongoClientSettings settings = MongoClientSettings.builder()
                .applyConnectionString(connectionString)
                .applyToSslSettings(builder - > {
                    builder.enabled(true);
                    builder.context(sslContext);
                })
                .build();
            connectedClient = MongoClients.create(settings);
        }
```

```
        return connectedClient;
    } catch (MongoException e5) {
        throw new RuntimeException(e5);
    } catch (Exception e) {
        throw new RuntimeException(e);
    }
}
```

4단계: 연결 예외 처리

Java 애플리케이션에서 DocumentDB로 작업할 때 강력하고 안정적인 데이터베이스 작업을 유지하려면 연결 예외를 처리하는 것이 중요합니다. 이러한 예외를 적절하게 관리하면 문제를 신속하게 진단하는 데 도움이 될 뿐만 아니라 애플리케이션이 임시 네트워크 중단 또는 서버 가용성을 정상적으로 처리할 수 있으므로 안정성과 사용자 경험이 향상됩니다. 연결 설정과 관련된 몇 가지 중요한 예외는 다음과 같습니다.

- **MongoException** - 일반적인 예외 및는 보다 구체적인 예외가 적용되지 않는 다양한 시나리오에서 발행될 수 있습니다. 이 예외는 모든 MongoDB 예외를 일반적으로 포착하기 때문에 다른 모든 특정 예외 이후에 처리되어야 합니다.
- **MongoTimeoutException** - 작업 시간이 초과될 때 발생합니다. 존재하지 않는 클러스터 엔드포인트를 쿼리하는 경우를 예로 들 수 있습니다.
- **MongoSocketException** - 네트워크 관련 문제에 대해 발급됩니다. 예를 들어 작업 중 갑작스러운 네트워크 연결 해제가 있습니다.
- **MongoSecurityException** - 인증이 실패할 때 발생합니다. 예를 들어 잘못된 자격 증명으로 연결합니다.
- **MongoConfigurationException** - 클라이언트 구성에 오류가 있을 때 발생합니다. 예를 들어 잘못된 연결 문자열을 사용합니다.

5단계: 코드 실행

다음 코드 샘플은 Amazon DocumentDB 연결을 생성하고 모든 데이터베이스를 인쇄합니다.

```
public static void TestConnection() {
    try (MongoClient mongoClient = makeDbConnection(DATABASE_NAME, DB_USER_NAME,
 DB_PASSWORD, DB_CLUSTER_ENDPOINT, KEYSTORE_PASSWORD)) {
        List < String > databases = mongoClient.listDatabaseNames().into(new ArrayList
 < > ());
        System.out.println("Databases: " + databases);
```

```
    } catch (MongoException e) {
        System.err.println("MongoDB error: " + e.getMessage());
        throw new RuntimeException(e);
    }
}
```

연결 모범 사례

다음은 MongoDB Java 드라이버를 사용하여 Amazon DocumentDB에 연결할 때 고려해야 할 모범 사례입니다.

- 클라이언트가 더 이상 리소스를 릴리스할 필요가 없는 `MongoClient` 경우 항상를 닫습니다.
- 예외를 적절하게 처리하고 적절한 오류 로깅을 구현합니다.
- 환경 변수 또는 AWS Secrets Manager 를 사용하여 사용자 이름 및 암호와 같은 민감한 정보를 저장합니다.

Java를 사용하여 Amazon DocumentDB에서 CRUD 작업 수행

이 섹션에서는 MongoDB Java 드라이버를 사용하여 Amazon DocumentDB에서 CRUD(생성, 읽기, 업데이트, 삭제) 작업을 수행하는 방법을 설명합니다.

주제

- DocumentDB 컬렉션에 문서 생성 및 삽입
- DocumentDB 컬렉션에서 데이터 읽기 및 검색
- DocumentDB 컬렉션의 기존 문서 업데이트
- DocumentDB 컬렉션에서 문서 제거
- 재시도 로직을 사용한 오류 처리

DocumentDB 컬렉션에 문서 생성 및 삽입

Amazon DocumentDB에 문서를 삽입하면 컬렉션에 새 데이터를 추가할 수 있습니다. 요구 사항과 작업 중인 데이터의 양에 따라 삽입을 수행하는 방법에는 여러 가지가 있습니다. 컬렉션에 개별 문서를 삽입하는 가장 기본적인 방법은 입니다`insertOne()`. 한 번에 여러 문서를 삽입하는 경우 단일 작업으로 문서 배열을 추가할 수 있는 `insertMany()` 메서드를 사용할 수 있습니다. DocumentDB 컬렉션에 많은 문서를 삽입하는 또 다른 방법은 입니다`bulkWrite()`. 이 가이드에서는 DocumentDB 컬렉션에서 문서를 생성하는 이러한 모든 방법을 설명합니다.

insertOne()

먼저 Amazon DocumentDBB 컬렉션에 개별 문서를 삽입하는 방법을 살펴보겠습니다. 단일 문서 삽입은 `insertOne()` 메서드를 사용하여 수행됩니다. 이 메서드는 삽입을 위해 BsonDocument를 가져와 새로 삽입된 문서의 `InsertOneResult` 객체 ID를 가져오는 데 사용할 수 있는 객체를 반환합니다. 아래 예제 코드는 컬렉션에 하나의 레스토랑 문서를 삽입하는 것을 보여줍니다.

```
Document article = new Document()
    .append("restaurantId", "REST-21G145")
    .append("name", "Future-proofed Intelligent Bronze Hat")
    .append("cuisine", "International")
    .append("rating", new Document()
        .append("average", 1.8)
        .append("totalReviews", 267))
    .append("features", Arrays.asList("Outdoor Seating", "Live Music"));

try {
    InsertOneResult result = collection.insertOne(article);
    System.out.println("Inserted document with the following id: " +
 result.getInsertedId());
} catch (MongoWriteException e) {
    // Handle duplicate key or other write errors
    System.err.println("Failed to insert document: " + e.getMessage());
    throw e;
} catch (MongoException e) {
    // Handle other MongoDB errors
    System.err.println("MongoDB error: " + e.getMessage());
    throw e;
}
```

를 사용할 때는 적절한 오류 처리를 포함해야 `insertOne()`합니다. 예를 들어 위 코드에서 "`restaurantId`"에는 고유한 인덱스가 있으므로이 코드를 다시 실행하면 `MongoWriteException`다음과 같은가 발생합니다.

```
Failed to insert document: Write operation error on server
 docdbCluster.docdb.amazonaws.com:27017.
Write error: WriteError{code=11000, message='E11000 duplicate key error collection:
 Restaurants index: restaurantId_1', details={}}.
```

insertMany()

컬렉션에 많은 문서를 삽입하는 데 사용되는 기본 방법은 insertMany() 및 입니다`bulkWrite()`.

메insertMany()서드는 단일 작업으로 여러 문서를 삽입하는 가장 간단한 방법입니다. 문서 목록을 받아 컬렉션에 삽입합니다. 이 방법은 서로 독립적이며 특별한 처리 또는 혼합 작업이 필요하지 않은 새 문서 배치를 삽입할 때 적합합니다. 다음 코드는 파일에서 JSON 문서를 읽고 컬렉션에 삽입하는 방법을 보여줍니다. insertMany() 함수는 삽입된 모든 문서의 IDs를 가져오는 데 사용할 수 있는 InsertManyResultInsertManyResult 객체를 반환합니다.

```
// Read JSON file content
String content = new String(Files.readAllBytes(Paths.get(jsonFileName)));
JSONArray jsonArray = new JSONArray(content);

// Convert JSON articles to Documents
List < Document > restaurants = new ArrayList < > ();
for (int i = 0; i < jsonArray.length(); i++) {
    JSONObject jsonObject = jsonArray.getJSONObject(i);
    Document doc = Document.parse(jsonObject.toString());
    restaurants.add(doc);
}
//insert documents in collection
InsertManyResult result = collection.insertMany(restaurants);

System.out.println("Count of inserted documents: " + result.getInsertedIds().size());
```

bulkWrite()

메bulkWrite()서드를 사용하면가 단일 배치에서 여러 쓰기 작업(삽입, 업데이트, 삭제)을 수행할 수 있습니다. 다른 문서를 업데이트하는 동안 일부 문서를 삽입하는 등 단일 배치에서 다양한 유형의 작업을 수행해야 하는 bulkWrite() 경우를 사용할 수 있습니다.는 순서가 지정된 배치 쓰기와 순서가 지정되지 않은 배치 쓰기의 두 가지 유형을 bulkWrite() 지원합니다.

- 순서가 지정된 작업 - (기본값) Amazon DocumentDB는 쓰기 작업을 순차적으로 처리하고 오류가 처음 발생할 때 중지합니다. 이는 이후 작업이 이전 작업에 의존하는 경우와 같이 작업 순서가 중요한 경우에 유용합니다. 그러나 정렬된 작업은 일반적으로 정렬되지 않은 작업보다 느립니다. 순서가 지정된 작업의 경우 첫 번째 오류 시 배치가 중지되어 일부 작업이 처리되지 않을 수 있는 경우를 해결해야 합니다.
- 순서가 지정되지 않은 작업 - Amazon DocumentDB가 데이터베이스에서 삽입을 단일 실행으로 처리할 수 있습니다. 한 문서에 오류가 발생하면 작업은 나머지 문서로 계속됩니다. 이는 대량의 데이터를 삽입할 때 특히 유용하며 중복 키로 인해 일부 문서가 실패할 수 있는 데이터 마이그레이션 또는 대량 가져오기와 같은 일부 오류를 허용할 수 있습니다. 순서가 지정되지 않은 작업의 경우 일부 작업은 성공하지만 다른 작업은 실패하는 부분 성공 시나리오를 해결해야 합니다.

bulkWrite() 메서드를 사용할 때는 몇 가지 필수 클래스가 필요합니다. 먼저 클래스WriteModel는 모든 쓰기 작업에 대한 기본 클래스 역할을 하며 , InsertOneModel, DeleteOneModel, UpdateManyModelUpdateOneModel와 같은 특정 구현을 통해 다양한 유형의 작업을 DeleteManyModel 처리합니다.

BulkWriteOptions 클래스는 순서가 지정된/순서가 지정되지 않은 실행 설정 또는 문서 검증 우회와 같은 대량 작업의 동작을 구성하는 데 필요합니다. BulkWriteResult 클래스는 삽입, 업데이트 및 삭제된 문서 수를 포함하여 실행 결과에 대한 자세한 정보를 제공합니다.

오류 처리의 경우 MongoBulkWriteException 클래스는 대량 작업 중 발생하는 모든 장애에 대한 정보를 포함하는 반면 BulkWriteError 클래스는 개별 작업 장애에 대한 특정 세부 정보를 제공하므로 매우 중요합니다. 다음 코드는 단일 bulkWrite() 메서드 호출 실행 내에서 문서 목록을 삽입하고 단일 문서를 업데이트 및 삭제하는 예를 보여줍니다. 또한 코드는 BulkWriteOptions 및 작업 방법과 bulkWrite() 작업의 BulkWriteResult적절한 오류 처리를 보여줍니다.

```
List < WriteModel < Document >> bulkOperations = new ArrayList < > ();
// get list of 10 documents representing 10 restaurants
List < Document > restaurantsToInsert = getSampleData();

for (Document doc: restaurantsToInsert) {
    bulkOperations.add(new InsertOneModel < > (doc));
}
// Update operation
bulkOperations.add(new UpdateOneModel < > (
    new Document("restaurantId", "REST-Y2E9H5"),
    new Document("", new Document("stats.likes", 20))
    .append("", new Document("rating.average", 4.5))));
// Delete operation
bulkOperations.add(new DeleteOneModel < > (new Document("restaurantId", "REST-
D2L431")));

// Perform bulkWrite operation
try {
    BulkWriteOptions options = new BulkWriteOptions()
        .ordered(false); // Allow unordered inserts

    BulkWriteResult result = collection.bulkWrite(bulkOperations, options);

    System.out.println("Inserted: " + result.getInsertedCount());
    System.out.println("Updated: " + result.getModifiedCount());
    System.out.println("Deleted: " + result.getDeletedCount());
} catch (MongoBulkWriteException e) {
```

```
    System.err.println("Bulk write error occurred: " + e.getMessage());
    // Log individual write errors
    for (BulkWriteError error: e.getWriteErrors()) {
        System.err.printf("Error at index %d: %s (Code: %d)%n", error.getIndex(),
 error.getMessage(),
            error.getCode());

        // Log the problematic document
        Document errorDoc = new Document(error.getDetails());
        if (errorDoc != null) {
            System.err.println("Problematic document: " + errorDoc);
        }
    }
} catch (Exception e) {
    System.err.println("Error during bulkWrite: " + e.getMessage());
}
```

재시도 가능한 쓰기

MongoDB와 달리 Amazon DocumentDB는 재시도 가능한 쓰기를 지원하지 않습니다. 따라서 특히 네트워크 문제 또는 임시 서비스 가용성을 처리하기 위해 애플리케이션에서 사용자 지정 재시도 로직을 구현해야 합니다. 일반적으로 잘 구현된 재시도 전략에는 재시도 간의 지연을 늘리고 총 재시도 횟수를 제한하는 것이 포함됩니다. 오류 처리로 재시도 로직을 빌드하는 코드 샘플은 재시도 로직을 사용한 오류 처리 아래를 참조하세요.

DocumentDB 컬렉션에서 데이터 읽기 및 검색

Amazon DocumentDB에서 문서를 쿼리하는 것은 데이터를 정확하게 검색하고 조작할 수 있는 몇 가지 주요 구성 요소를 중심으로 합니다. `find()` 메서드는 MongoDB Java 드라이버의 기본 쿼리 APIs. 결과를 필터링, 정렬 및 프로젝션하기 위한 다양한 옵션으로 복잡한 데이터를 검색할 수 있습니다. `find()` 메서드 외에도 `Filters` 및 `FindIterable`는 MongoDB Java 드라이버에서 쿼리 작업을 위한 구성 요소를 제공하는 두 가지 다른 기본 구성 요소입니다.

`Filters` 클래스는 쿼리 필터를 구성하기 위한 유창한 API를 제공하는 MongoDB Java 드라이버의 유틸리티 클래스입니다. 이 클래스는 다양한 쿼리 조건을 나타내는 `Bson` 객체의 인스턴스를 생성하는 정적 팩토리 메서드를 제공합니다. 가장 일반적으로 사용되는 메서드에는 `eq()`균등 비교, `gt()` `lt()` `gte()` `lte()` 숫자 비교, 여러 조건 `and()` `or()` 결합, `in()` `nin()` 배열 멤버십 테스트, 패턴 일치 `regex()` 등이 있습니다. 클래스는 유형 안전하도록 설계되었으며 원시 문서 기반 쿼리에 비해 더 나은 컴파일 시간 확인을 제공하므로 Java 애플리케이션에서 DocumentDB 쿼리를 구성하는 데 선호되는 접근 방식입니다. 오류 처리는 견고하며 잘못된 필터 구성에 대해 명확한 예외가 발생합니다.

FindIterable는 find() 메서드의 결과를 처리하도록 설계된 특수 인터페이스입니다. 쿼리 실행을 구체화하고 제어하는 다양한 방법을 제공하여 메서드 체인을 위한 유창한 API를 제공합니다. 인터페이스에는 반환된 문서 수 limit() 제한, skip() 페이지 매김, 결과 sort() 순서 지정, 특정 필드 projection() 선택, 인덱스 선택과 같은 필수 쿼리 수정 방법이 포함되어 hint() 있습니다. 의 배치, 건너뛰기 및 제한 작업은 데이터베이스에서 문서를 검색하고 처리하는 방법을 제어하는 데 도움이 되는 FindIterable 필수 페이지 매김 및 데이터 관리 도구입니다.

일괄 처리(batchSize)는 DocumentDB가 단일 네트워크 왕복으로 클라이언트에 반환하는 문서 수를 제어합니다. 배치 크기를 설정하면 DocumentDB는 일치하는 모든 문서를 한 번에 반환하지 않고 대신 지정된 배치 크기의 그룹으로 반환합니다.

건너뛰기를 사용하면 결과의 시작점을 오프셋하여 일치 항목을 반환하기 전에 지정된 수의 문서를 건너뛰도록 DocumentDB에 지시할 수 있습니다. 예를 들어 skip(20)는 처음 20개의 일치하는 문서를 우회합니다. 이는 결과의 후속 페이지를 검색하려는 페이지 매김 시나리오에서 일반적으로 사용됩니다.

제한은 쿼리에서 반환할 수 있는 총 문서 수를 제한합니다. 를 지정하면 데이터베이스에 일치하는 항목이 더 있더라도 limit(n) DocumentDB는 'n' 문서를 반환한 후 문서 반환을 중지합니다.

FindIterable는 Amazon DocumentDB에서 문서를 검색할 때 반복자와 커서 패턴을 모두 지원합니다. 를 FindIterable 반복자로 사용하면 문서를 느리게 로드할 수 있으며 애플리케이션에서 요청할 때만 문서를 가져올 수 있다는 이점이 있습니다. 반복기 사용의 또 다른 이점은 클러스터에 대한 연결을 유지할 책임이 없으므로 연결을 명시적으로 닫을 필요가 없다는 것입니다.

FindIterable는 Amazon DocumentDB 쿼리로 작업할 때 커서 패턴을 사용할 수 MongoCursor 있는 지원도 제공합니다. MongoCursor는 데이터베이스 작업 및 리소스 관리를 제어하는 MongoDB Java 드라이버별 구현입니다. AutoCloseable 인터페이스를 구현하여 try-with-resources 통해 명시적 리소스 관리를 가능하게 합니다.이 블록은 데이터베이스 연결을 올바르게 닫고 서버 리소스를 확보하는 데 매우 중요합니다. 기본적으로 커서는 10분 후에 시간 초과되며 DocumentDB는이 시간 초과 동작을 변경할 수 있는 옵션을 제공하지 않습니다. 배치 데이터로 작업할 때는 커서 시간이 초과되기 전에 다음 데이터 배치를 검색해야 합니다. 를 사용할 때 한 가지 주요 고려 MongoCursor 사항은 리소스 누수를 방지하기 위해 명시적으로 닫아야 한다는 것입니다.

이 단원에서는 , find() Filters 및에 대한 몇 가지 예제가 나와 있습니다FindIterable.

다음 코드 예제에서는를 사용하여 “restaurantId” 필드를 사용하여 단일 문서를 find() 검색하는 방법을 보여줍니다.

```
Document filter = new Document("restaurantId", "REST-21G145");
Document result = collection.find(filter).first();
```

를 사용하면 컴파일 시간 오류 검사를 개선할 Filters 수 있지만 java 드라이버를 사용하면 find() 메서드에서 직접 Bson 필터를 지정할 수도 있습니다. 다음 예제 코드는 Bson 문서를에 전달합니다find().

```
result = collection.find(new Document("$and", Arrays.asList(
    new Document("rating.totalReviews", new Document("$gt", 1000)),
    new Document("priceRange", "$$"))))
```

다음 예제 코드는에서 Filters 클래스를 사용하는 몇 가지 예를 보여줍니다. find()

```
FindIterable < Document > results;

// Exact match
results = collection.find(Filters.eq("name", "Thai Curry Palace"));

// Not equal
results = collection.find(Filters.ne("cuisine", "Thai"));

// find an element in an array
results = collection.find(Filters.in("features", Arrays.asList("Private Dining")));

// Greater than
results = collection.find(Filters.gt("rating.average", 3.5));

// Between (inclusive)
results = collection.find(Filters.and(
    Filters.gte("rating.totalReviews", 100),
    Filters.lte("rating.totalReviews", 200)));
// AND
results = collection.find(Filters.and(
    Filters.eq("cuisine", "Thai"),
    Filters.gt("rating.average", 4.5)));

// OR
results = collection.find(Filters.or(
    Filters.eq("cuisine", "Thai"),
    Filters.eq("cuisine", "American")));


// All document where the Field exists
results = collection.find(Filters.exists("michelin"));

// Regex
```

```
results = collection.find(Filters.regex("name", Pattern.compile("Curry",
 Pattern.CASE_INSENSITIVE)));

// Find all document where the array contain the list of value regardless of its order
results = collection.find(Filters.all("features", Arrays.asList("Private Dining",
 "Parking")));

// Array size
results = collection.find(Filters.size("features", 4));
```

다음 예제에서는 batchSize() FindIterable 객체에서 sort(), skip()limit(), 및 작업을 연결하는 방법을 보여줍니다. 이러한 작업이 제공되는 순서는 쿼리 성능에 영향을 미칩니다. 가장 좋은 방법은 이러한 작업의 순서가 , sort(), projection() skip() limit() 및 이어야 한다는 것입니다batchSize().

```
FindIterable < Document > results = collection.find(Filters.gt("rating.totalReviews",
 1000))
    // Sorting
    .sort(Sorts.orderBy(
        Sorts.descending("address.city"),
        Sorts.ascending("cuisine")))
    // Field selection
    .projection(Projections.fields(
        Projections.include("name", "cuisine", "priceRange"),
        Projections.excludeId()))

    // Pagination
    .skip(20)
    .limit(10)
    .batchSize(2);
```

다음 예제 코드는에서 반복자를 생성하는 방법을 보여줍니다FindIterable. Java의 forEach 구문을 사용하여 결과 집합을 통과합니다.

```
collection.find(Filters.eq("cuisine", "American")).forEach(doc ->
 System.out.println(doc.toJson()));
```

마지막 find() 코드 예제에서는 문서 검색에 cursor()를 사용하는 방법을 보여줍니다. 코드가 시도 블록을 종료할 때 커서가 닫히도록 시도 블록에 커서를 생성합니다.

```
try (MongoCursor < Document > cursor = collection.find(Filters.eq("cuisine",
 "American"))
    .batchSize(25)
    .cursor()) {
    while (cursor.hasNext()) {
        Document doc = cursor.next();
        System.out.println(doc.toJson());
    }
} // Cursor automatically closed
```

DocumentDB 컬렉션의 기존 문서 업데이트

Amazon DocumentDB는 기존 문서를 수정하고 존재하지 않을 때 새 문서를 삽입할 수 있는 유연하고 강력한 메커니즘을 제공합니다. MongoDB Java 드라이버는 `updateOne()` 단일 문서 업데이트, `updateMany()` 여러 문서 업데이트, `replaceOne()` 전체 문서 교체 등 여러 가지 업데이트 방법을 제공합니다. 이 세 가지 방법 외에도 `Updates`, `UpdateOptions`및 `UpdateResult`는 MongoDB Java 드라이버에서 업데이트 작업을 위한 구성 요소를 제공하는 다른 기본 구성 요소입니다.

MongoDB Java 드라이버의 `Updates` 클래스는 업데이트 연산자를 생성하기 위한 정적 팩토리 메서드를 제공하는 유틸리티 클래스입니다. 유형 안전 및 읽기 가능한 방식으로 업데이트 작업을 구성하기 위한 기본 빌더 역할을 합니다. `set()`, `unset()`및와 같은 기본 메서드를 `inc()` 사용하면 문서를 직접 수정할 수 있습니다. 여러 업데이트 작업을 원자적으로 실행하여 데이터 일관성을 보장할 수 있는 `Updates.combine()` 메서드를 사용하여 여러 작업을 결합하면이 클래스의 성능이 분명해집니다.

`UpdateOptions`는 문서 업데이트 작업에 필수적인 사용자 지정 기능을 제공하는 MongoDB의 Java 드라이버에 있는 강력한 구성 클래스입니다. 이 클래스의 두 가지 중요한 측면은 업데이트 작업에 대한 업서트 및 배열 필터 지원을 제공하는 것입니다. 를 통해 활성화된 업서트 기능을 `upsert(true)`사용하면 업데이트 작업 중에 일치하는 문서를 찾을 수 없는 경우 새 문서를 생성할 수 있습니다. `arrayFilters()`를 통해 업데이트 작업은 특정 기준을 충족하는 배열 요소를 정확하게 업데이트할 수 있습니다.

`UpdateResult` MongoDB의 Java 드라이버에서는 업데이트 작업의 결과를 자세히 설명하는 피드백 메커니즘을 제공합니다. 이 클래스는 업데이트 기준과 일치하는 문서 수(`matchedCount`), 실제로 수정된 문서 수(`modifiedCount`), 업서트된 문서에 대한 정보()의 세 가지 주요 지표를 캡슐화합니다`upsertedId`. 이러한 지표를 이해하는 것은 적절한 오류 처리, 업데이트 작업 확인 및 애플리케이션의 데이터 일관성 유지에 필수적입니다.

단일 문서 업데이트 및 교체

DocumentDB에서는 updateOne() 메서드를 사용하여 단일 문서를 업데이트할 수 있습니다. 이 메서드는 업데이트할 문서를 식별하기 위해 일반적으로 Filters 클래스에서 제공하는 필터 파라미터, 업데이트할 필드를 결정하는 Update 파라미터, 업데이트에 대한 다양한 옵션을 설정하는 선택적 UpdateOptions 파라미터를 사용합니다. updateOne() 메서드를 사용하면 선택 기준과 일치하는 첫 번째 문서만 업데이트됩니다. 다음 예제 코드는 문서 하나의 단일 필드를 업데이트합니다.

```
collection.updateOne(Filters.eq("restaurantId", "REST-Y2E9H5"),
    Updates.set("name", "Amazing Japanese sushi"));
```

한 문서에서 여러 필드를 업데이트하려면 다음 예제Update.combine()와 같이와 updateOne() 함께를 사용합니다. 이 예제에서는 문서의 배열에 항목을 추가하는 방법도 보여줍니다.

```
List<Bson> updates = new ArrayList<>();
// Basic field updates
updates.add(Updates.set("name", "Shanghai Best"));
// Array operations
updates.add(Updates.addEachToSet("features", Arrays.asList("Live Music")));
// Counter updates
updates.add(Updates.inc("rating.totalReviews", 10));
// Combine all updates
Bson combinedUpdates = Updates.combine(updates);
// Execute automic update with one call
collection.updateOne(Filters.eq("restaurantId","REST-1J83NH"), combinedUpdates);
```

다음 코드 예제에서는 데이터베이스에서 문서를 업데이트하는 방법을 보여줍니다. 지정된 문서가 없는 경우 작업은 자동으로 새 문서로 삽입합니다. 이 코드는 UpdateResult 객체를 통해 사용 가능한 지표를 사용하는 방법도 보여줍니다.

```
Bson filter = Filters.eq("restaurantId", "REST-0Y9GL0");
Bson update = Updates.set("cuisine", "Indian");
// Upsert operation
UpdateOptions options = new UpdateOptions().upsert(true);
UpdateResult result = collection.updateOne(filter, update, options);

if (result.getUpsertedId() != null) {
    System.out.println("Inserted document with _id: " + result.getUpsertedId());
} else {
     System.out.println("Updated " + result.getModifiedCount() + " document(s)");
}
```

다음 코드 예제에서는 개별 필드를 업데이트하는 대신 replaceOne() 메서드를 사용하여 기존 문서를 새 문서로 완전히 바꾸는 방법을 보여줍니다. replaceOne() 메서드는 원본의 _id 필드만 보존하여 전체 문서를 덮어씁니다. 여러 문서가 필터 기준과 일치하는 경우 처음 발견된 문서만 대체됩니다.

```
Document newDocument = new Document()
                .append("restaurantId", "REST-0Y9GL0")
                .append("name", "Bhiryani Adda")
                .append("cuisine", "Indian")
                .append("rating", new Document()
                        .append("average", 4.8)
                        .append("totalReviews", 267))
                .append("features", Arrays.asList("Outdoor Seating", "Live Music"));

UpdateResult result = collection.replaceOne(
                    Filters.eq("restaurantId", "REST-0Y9GL0"),
                    newDocument);
System.out.printf("Modified %d document%n", result.getModifiedCount());
```

여러 문서 업데이트

컬렉션에서 여러 문서를 동시에 업데이트하는 두 가지 방법이 있습니다. updateMany() 메서드를 사용하거나 bulkWrite() 메서드와 UpdateManyModel 함께를 사용할 수 있습니다. updateMany() 메서드는 필터 파라미터를 사용하여 업데이트할 문서를 선택하고, Update 파라미터를 사용하여 업데이트할 필드를 식별하고, 선택적 UpdateOptions 파라미터를 사용하여 업데이트 옵션을 지정합니다.

다음 예제 코드는 updateMany() 메서드의 사용을 보여줍니다.

```
Bson filter = Filters.and(
    Filters.in("features", Arrays.asList("Private Dining")),
    Filters.eq("cuisine", "Thai"));
UpdateResult result1 = collection.updateMany(filter, Updates.set("priceRange", "$$$"));
```

다음 예제 코드는 동일한 업데이트를 사용하는 bulkWrite() 메서드를 보여줍니다.

```
BulkWriteOptions options = new BulkWriteOptions().ordered(false);
List < WriteModel < Document >> updates = new ArrayList < > ();
Bson filter = Filters.and(
    Filters.in("features", Arrays.asList("Private Dining")),
    Filters.eq("cuisine", "Indian"));
Bson updateField = Updates.set("priceRange", "$$$");
updates.add(new UpdateManyModel < > (filter, updateField));
BulkWriteResult result = collection.bulkWrite(updates, options);
```

```
System.out.printf("Modified %d document%n", result.getModifiedCount());
```

DocumentDB 컬렉션에서 문서 제거

MongoDB Java 드라이버는 단일 문서를 `deleteOne()` 제거하고 특정 기준과 일치하는 여러 문서를 제거하기 `deleteMany()` 위해를 제공합니다. 업데이트와 마찬가지로 삭제 작업을 `bulkWrite()` 메서드와 함께 사용할 수도 있습니다. `deleteOne()` 및 모두 삭제된 문서 수를 포함하여 작업 결과에 대한 정보를 제공하는 `DeleteResult` 객체를 `deleteMany()` 반환합니다. 다음은를 사용하여 여러 문서를 `deleteMany()` 제거하는 예입니다.

```
Bson filter = Filters.and(
    Filters.eq("cuisine", "Thai"),
    Filters.lt("rating.totalReviews", 50));
DeleteResult result = collection.deleteMany(filter);
System.out.printf("Deleted %d document%n", result.getDeletedCount());
```

재시도 로직을 사용한 오류 처리

Amazon DocumentDB에 대한 강력한 오류 처리 전략은 오류 분류를 재시도 가능(예: 네트워크 제한 시간, 연결 문제) 및 재시도 불가능(예: 인증 실패, 잘못된 쿼리)으로 구현해야 합니다. 재시도해야 하는 오류로 인한 작업 실패의 경우 각 재시도와 최대 재시도 사이에 시간 지연을 구현해야 합니다. CRUD 작업은 `MongoException` 및 해당 하위 클래스를 포착하는 try-catch 블록에 있어야 합니다. 또한 운영 가시성을 위해 오류 모니터링 및 로깅을 포함해야 합니다. 다음은 재시도 오류 처리를 구현하는 방법을 보여주는 샘플 코드입니다.

```
int MAX_RETRIES = 3;
int INITIAL_DELAY_MS = 1000;
int retryCount = 0;

while (true) {
    try {
        crud_operation(); //perform crud that will throw MongoException or one of its
 subclass
        break;
    } catch (MongoException e) {
        if (retryCount < MAX_RETRIES) {
            retryCount++;
            long delayMs = INITIAL_DELAY_MS * (long) Math.pow(2, retryCount - 1);
            try {
                TimeUnit.MILLISECONDS.sleep(delayMs);
            } catch (InterruptedException t) {
```

```
                Thread.currentThread().interrupt();
                throw new RuntimeException("Retry interrupted", t);
            }
            continue;
        } else
            throw new RuntimeException("Crud operation failed", e);
    }
}
```

Java를 사용한 Amazon DocumentDB의 인덱스 관리

인덱스를 사용하면 Amazon DocumentDB 컬렉션에서 데이터를 효율적으로 검색할 수 있습니다. 인덱스가 없으면 DocumentDB는 컬렉션의 모든 문서를 스캔하여 지정된 쿼리를 충족하는 결과를 반환해야 합니다. 이 주제에서는 MongoDB Java 드라이버를 사용하여 인덱스를 생성, 삭제 및 나열하는 방법에 대한 정보를 제공합니다. 또한 쿼리에서 특정 인덱스가 사용되고 있는지 확인하는 방법과 특정 인덱스를 사용하도록 Amazon DocumentDB에 힌트를 제공하는 방법도 설명합니다.

주제

- Java를 사용하여 인덱스 생성
- 인덱스 삭제
- 인덱스 선택 결정 및 인덱스 힌트 제공

Amazon DocumentDB는 다양한 유형의 인덱스를 지원합니다. 지원되는 모든 인덱스에 대한 포괄적인 개요는이 블로그 게시물을 참조하세요.

Java를 사용하여 인덱스 생성

MongoDB Java 드라이버를 사용하여 Amazon DocumentDB에서 인덱스를 생성하는 두 가지 메커니즘은 단일 인덱스의 `createIndex()` 경우 메서드 또는 여러 인덱스의 경우 `createIndexes()` 메서드를 `runCommand()`통해 ~입니다. `createIndex()` 및 `createIndexes()` 메서드를 사용하는 한 가지 이유는 인덱스 생성과 관련된 특정 오류를 포착하여 오류 처리를 개선할 수 있기 때문입니다. 에서 이러한 방법을 사용하는 또 다른 이유는 MongDB Java 드라이버`runCommand()`가 인덱스 생성 및 조작을 위한 풍부한 지원 클래스 세트를 제공하기 때문입니다. 이러한 지원 클래스는 `createIndex()` 또는 `createIndexes()` 메서드를 사용하는 경우에만 사용할 수 있습니다. 지원 클래스에는 세 가지가 있습니다.

- **`Indexes`** -이 클래스는 다양한 유형의 인덱스를 생성하기 위한 정적 팩토리 메서드를 제공하는 유틸리티 클래스 역할을 합니다. 복잡한 인덱스 정의를 생성하는 프로세스를 간소화하며 일반적으로 다른 인덱스 관련 클래스와 함께 사용됩니다.

- **IndexModel** - 인덱스 키 정의와 해당 옵션을 모두 캡슐화하는 기본 클래스입니다. 인덱싱할 항목(키)과 인덱싱 방법(옵션)을 결합한 전체 인덱스 사양을 나타냅니다. 이 클래스는 `createIndexes()` 여러 인덱스를 동시에 생성할 때 특히 유용합니다. 메서드에 전달할 수 있는 인덱스 사양 모음을 정의할 수 있기 때문입니다.
- **IndexOptions** - 인덱스 동작을 사용자 지정하기 위한 다양한 방법을 제공하는 포괄적인 구성 클래스입니다. 여기에는 고유 인덱스, 희소 인덱스, 만료 시간(TTL) 및 부분 필터 표현식에 대한 설정이 포함됩니다. 메서드 체인을 통해 백그라운드 인덱스 구축 및 고유한 제약 조건과 같은 여러 옵션을 구성할 수 있습니다.

단일 인덱스 생성

이 예제에서는 백그라운드에서 `createIndex()` 메서드를 사용하여 단일 인덱스를 생성하는 방법을 보여줍니다. 배경 및 포그라운드 인덱스 생성을 이해하려면 섹션을 참조하세요인덱스 빌드 유형. 다음 코드 예제에서는 `IndexOptions`를 사용하여 백그라운드에서 이름이 “unique_restaurantId_idx”인 고유 인덱스를 생성합니다. 그러면이 `IndexOptions` 객체가 `createIndex()` 메서드로 전달됩니다.

```
collection.createIndex(
    Indexes.ascending("restaurantId"),
    new IndexOptions()
        .unique(true)
        .name("unique_restaurantId_idx")
        .background(true));
```

여러 인덱스 생성

이 예제에서는 `createIndexes()` 메서드를 사용하여 여러 인덱스를 생성합니다. 먼저 `IndexModel` 객체를 사용하여 각 인덱스에 대한 옵션을 빌드한 다음 `IndexModel` 객체 목록을 `createIndexes()` 메서드에 전달합니다. 다음 코드 예제에서는 `Indexes` 유틸리티 클래스를 사용하여 복합 인덱스를 생성하는 방법을 보여줍니다. 이 클래스는 오름차순 또는 내림차순 정렬 순서를 사용하여 인덱스를 생성할지 여부를 지정하는 데에도 사용됩니다. 여러 인덱스를 생성한 후 `listIndexes()` 메서드를 호출하여 인덱스 생성을 확인합니다.

```
// Single Field Index on cuisine
IndexModel singleIndex = new IndexModel(
    Indexes.ascending("cuisine"),
    new IndexOptions().name("cuisine_idx"));

// Compound Index
IndexModel compoundIndex = new IndexModel(
```

```
    Indexes.compoundIndex(
        Indexes.ascending("address.state"),
        Indexes.ascending("priceRange")),
    new IndexOptions().name("location_price_idx"));

// Build a list of IndexModel for the indexes
List < IndexModel > indexes = Arrays.asList(
    singleIndex,
    compoundIndex
);

collection.createIndexes(indexes);

// Verify created indexes
collection.listIndexes().forEach(index - > System.out.println("Created index: " +
 index.toJson()));
```

희소 인덱스 및 부분 인덱스 생성

이 예제에서는 각 인덱스 유형에 `IndexModel` 대해를 생성하여 희소 인덱스와 부분 인덱스를 생성합니다.

```
// Sparse Index Model, this will identify only those documents that have a
// michelin star rating
IndexModel sparseIndex = new IndexModel(
    Indexes.ascending("michelin.star"),
    new IndexOptions()
    .name("michelin_sparse_idx")
    .sparse(true));

// Partial Index Model where the restaurant is active and has a rating of 4 and above
IndexModel partialIndex = new IndexModel(
    Indexes.ascending("rating.average"),
    new IndexOptions()
    .name("high_rated_active_idx")
    .partialFilterExpression(
        Filters.and(
            Filters.eq("isActive", true),
            Filters.gte("rating.average", 4.0))));
```

텍스트 인덱스 생성

이 예제에서는 텍스트 인덱스를 생성하는 방법을 보여줍니다. 컬렉션에는 하나의 텍스트 인덱스만 허용되지만 하나의 텍스트 인덱스는 여러 필드를 포함하는 복합 인덱스일 수 있습니다. 텍스트 인덱스에서 여러 필드를 사용하는 경우 인덱스의 각 필드에 가중치를 할당할 수도 있습니다. 배열 필드의 텍스트 인덱스는 Amazon DocumentDB에서 지원되지 않으며 복합 텍스트 인덱스에 최대 30개의 필드를 사용할 수 있지만 세 개의 필드만 가중치를 할당할 수 있습니다.

```
IndexModel textIndex = new IndexModel(
    new Document()
    .append("name", "text")
    .append("description", "text")
    .append("cuisine", "text"),
    new IndexOptions()
    .name("restaurant_text_idx")
    .weights(new Document()
        .append("name", 10) // Restaurant name gets highest weight
        .append("description", 5) // Description get medium weight
        .append("cuisine", 2) // Cuisine type gets low weight
    ));

collection.createIndex(textIndex.getKeys(), textIndex.getOptions());
```

를 사용하여 인덱스 생성 **runCommand()**

Amazon DocumentDB는 병렬 인덱스 생성을 지원하여 인덱스를 생성하는 데 걸리는 시간을 줄입니다. 병렬 인덱싱은 여러 동시 작업자를 사용합니다. 인덱스 생성에 사용되는 기본 작업자는 2개입니다. 이 블로그 게시물은 병렬 인덱싱에 대한 심층적인 설명을 제공합니다. 현재 MongDB Java 드라이버는 `createIndex()` 또는를 사용할 때 작업자 옵션 지정을 지원하지 `createIndexes()` 않으므로 작업자를 지정하는 유일한 방법은를 통하는 것입니다`runCommand`. 다음 코드 예제에서는를 사용하여 작업자를 4`runCommand`개로 늘리는 인덱스를 생성하는 방법을 보여줍니다.

```
Document command = new Document("createIndexes", "Restaurants")
    .append("indexes", Arrays.asList(
        new Document("key", new Document("name", 1))
        .append("name", "restaurant_name_idx")
        .append("workers", 4) // Specify number of workers
    ));

Document commendResult = connectedDB.runCommand(command);
```

인덱스 삭제

MongoDB Java 드라이버는 인덱스를 삭제하는 여러 방법을 제공하여 다양한 시나리오와 기본 설정에 맞게 조정할 수 있습니다. 이름, 키 사양 또는 모든 인덱스를 한 번에 삭제할 수 있습니다. 컬렉션 객체에서 메서드 `dropIndex()` 및를 호출하여 인덱스를 삭제할 `dropIndexes()` 수 있습니다. 이름별로 인덱스를 삭제할 때는 특히 복합 또는 자동 생성된 인덱스의 경우 항상 직관적이지 않을 수 있는 올바른 인덱스 이름을 사용해야 합니다. 존재하지 않는 인덱스를 삭제하려고 하면가 발생합니다[MongoCommandException](). `default _id` 인덱스는 컬렉션 내에서 문서 고유성을 보장하기 때문에 삭제할 수 없습니다.

다음 코드 예제에서는 인덱스가 생성되는 필드 이름을 제공하거나 모든 인덱스를 삭제하여 인덱스를 삭제하는 방법을 보여줍니다.

```
String indexName = "unique_restaurantId_idx";
Document keys = new Document("cuisine", 1);
// Drop index by name
collection.dropIndex(indexName);

// Drop index by keys
collection.dropIndex(keys);

// Drop all indexes
collection.dropIndexes();
```

여러 키를 사용하여 인덱스를 삭제할 때는 지정된 모든 키가 포함된 복합 인덱스가 있고 키 순서가 올바른지 확인합니다. 위의 인덱스 생성 예제 코드는 "cuisine" 및 기능에 대한 복합 키를 보여줍니다. 해당 복합 키를 삭제하려고 하지만 순서가 생성에 사용된 순서가 아닌 경우 MongoCommnadException 오류가 다음과 같이 발생합니다.

```
Document keys = new Document("features", 1)
    .append("cuisine", 1);
try {
    // Drop index by keys
    collection.dropIndex(keys);
    System.out.println("Successfully dropped index with keys: " + keys.toJson());

} catch (MongoCommandException commErr) {
    System.out.println("Error dropping index: " + commErr.getErrorMessage());
    throw new RuntimeException("MongoCommandException was thrown while dropping index",
 commErr);
}
```

다음 오류가 표시됩니다.

```
Error dropping index: Cannot drop index: index not found.
Tests run: 3, Failures: 1, Errors: 0, Skipped: 0, Time elapsed: 0.819 sec <<< FAILURE!
com.amazon.docdb.guide.DocDBGuideTest.testindexGuide()  Time elapsed: 0.817 sec  <<<
 FAILURE!
org.opentest4j.AssertionFailedError: Unexpected exception thrown:
 java.lang.RuntimeException: MongoCommandException was thrown while dropping index
```

인덱스 선택 결정 및 인덱스 힌트 제공

Amazon DocumentDB의 설명 기능을 사용하는 것은 쿼리 성능과 인덱스 사용량을 이해하는 데 매우 중요합니다. 쿼리를 실행할 때 `explain()` 메서드를 추가하여 사용 중인 인덱스를 포함하여 쿼리 계획에 대한 자세한 정보를 얻을 수 있습니다. `explain()` 출력은 쿼리 실행 단계, 검사된 문서 수, 각 단계에 소요된 시간에 대한 인사이트를 제공합니다. 이 정보는 특정 인덱스가 효과적으로 사용되고 있는지 또는 쿼리가 다른 인덱스 구조의 이점을 누릴 수 있는지 식별하는 데 매우 중요합니다.

`explain()` 메서드는 `find()` 메서드와 함께 연결할 수 있습니다. `explain()` 메서드는 ExplainVerbosity에서 반환되는 세부 수준을 결정하는 선택적 열거형을 사용할 수 있습니다`explain()`. 현재 DocumentDB에서는 `EXECUTION_STATS` 및 `QUERY_PLANNER` 열거자만 지원됩니다. 다음 코드 예제에서는 특정 쿼리에 대한 쿼리 플래너를 가져오는 방법을 보여줍니다.

```
// Query we want to analyze
Document query = new Document()
    .append("cuisine", "Thai")
    .append("rating.average", new Document("$gte", 4.0));


Document allPlansExplain =
 collection.find(query).explain(ExplainVerbosity.QUERY_PLANNER);
System.out.println("All Plans Explain:\n" + allPlansExplain.toJson());
```

쿼리 플래너 세부 수준에 대해 다음 JSON 문서가 반환됩니다.

```
{
  "queryPlanner": {
    "plannerVersion": 1,
    "namespace": "ProgGuideData.Restaurants",
    "winningPlan": {
      "stage": "IXSCAN",
      "indexName": "cuisine_idx",
```

```
      "direction": "forward"
    }
  },
  "serverInfo": {
    "host": "guidecluster3",
    "port": 27017,
    "version": "5.0.0"
  },
  "ok": 1,
  "operationTime": {
    "$timestamp": {
      "t": 1739221668,
      "i": 1
    }
  }
}
```

Amazon DocumentDB가 특정 인덱스를 사용하도록 영향을 미치거나 강제하는 몇 가지 옵션이 있습니다. `hint()` 및 `hintString()` 메서드를 사용하면 쿼리에 사용할 인덱스를 명시적으로 지정하여 쿼리 최적화 프로그램의 기본 인덱스 선택 동작을 재정의할 수 있습니다. DocumentDB의 쿼리 옵티마이저는 일반적으로 인덱스 선택에 적합한 선택이지만 왜곡된 데이터를 처리하거나 인덱스 성능을 테스트하는 등 특정 인덱스를를 통해 강제 적용`hint()`하거나 유용할 `hintString()` 수 있는 시나리오가 있습니다.

다음 코드 예제에서는 위 코드에서 실행된 동일한 쿼리에 복합 인덱스 “cuisine_features_idx”를 강제로 사용합니다.

```
// Query we want to analyze
Document query = new Document()
    .append("cuisine", "Thai")
    .append("rating.average", new Document("$gte", 4.0));

List < Document > queryDocs = new ArrayList < > ();
collection.find(query).hintString("cuisine_features_idx").forEach(doc - >
 queryDocs.add(doc));
```

Amazon DocumentDB 및 Java를 사용한 이벤트 기반 프로그래밍

Amazon DocumentDB의 맥락에서 이벤트 기반 프로그래밍은 데이터베이스 변경이 후속 비즈니스 로직 및 프로세스를 트리거하는 기본 이벤트 생성기 역할을 하는 강력한 아키텍처 패턴을 나타냅니다. DocumentDB 컬렉션에 레코드가 삽입, 업데이트 또는 삭제되면 이러한 변경 사항은 다양한 다운스트

림 프로세스, 알림 또는 데이터 동기화 작업을 자동으로 시작하는 이벤트 역할을 합니다. 이 패턴은 여러 애플리케이션 또는 서비스가 데이터 변경에 실시간으로 대응해야 하는 최신 분산 시스템에서 특히 유용합니다. DocumentDB에서 이벤트 기반 프로그래밍을 구현하는 주요 메커니즘은 변경 스트림입니다.

> **Note**
>
> 이 안내서에서는 작업 중인 컬렉션에서 변경 스트림을 활성화했다고 가정합니다. 컬렉션에서 변경 스트림을 활성화하는 방법은 Amazon DocumentDB에서 변경 스트림 사용 섹션을 참조하세요.

Java 애플리케이션에서 변경 스트림 작업

MongoDB의 Java 드라이버에 있는 `watch()` 메서드는 Amazon DocumentDB의 실시간 데이터 변경 사항을 모니터링하는 기본 메커니즘입니다. `watch()` 메서드는 `MongoClient`, `MongoDatabase`및 `MongoCollection` 객체에 의해에서 호출될 수 있습니다.

`watch()` 메서`ChangeStreamIterable`드는 업데이트에 대한 전체 문서 조회, 신뢰성을 위한 재개 토큰 및 타임스탬프 제공, 변경 사항 필터링을 위한 파이프라인 집계 단계 등 다양한 구성 옵션을 지원하는의 인스턴스를 반환합니다.

`ChangeStreamIterable`는 코어 Java 인터페이스를 구현`Iterable`하며와 함께 사용할 수 있습니다`forEach()`. 를 사용하여 이벤트를 캡처하려면 변경된 이벤트를 `forEach()` 처리하는에 콜백 함수를 `forEach()`전달합니다. 다음 코드 조각은 컬렉션에서 변경 스트림을 열어 변경 이벤트 모니터링을 시작하는 방법을 보여줍니다.

```
ChangeStreamIterable < Document > iterator = collection.watch();
iterator.forEach(event - > {
    System.out.println("Received a change: " + event);
});
```

모든 변경 이벤트를 통과하는 또 다른 방법은 클러스터에 대한 연결을 유지하고 새 변경 이벤트가 발생할 때 지속적으로 수신하는 커서를 여는 것입니다. 변경 스트림 커서를 가져오려면 `ChangeStreamIterable` 객체의 `cursor()` 메서드를 사용합니다. 다음 코드 예제에서는 커서를 사용하여 변경 이벤트를 모니터링하는 방법을 보여줍니다.

```
try (MongoChangeStreamCursor < ChangeStreamDocument < Document >> cursor =
 collection.watch().cursor()) {
```

```
    System.out.println(cursor.tryNext());
}
```

가장 좋은 방법은 try-with-resource 문MongoChangeStreamCursor에서를 생성하거나 커서를 수동으로 닫는 것입니다. 에서 cursor() 메서드를 호출하면 ChangeStreamDocument 객체를 통해 생성된 MongoChangeStreamCursor이 ChangeStreamIterable 반환됩니다.

ChangeStreamDocument 클래스는 스트림의 개별 변경 이벤트를 나타내는 중요한 구성 요소입니다. 여기에는 작업 유형(삽입, 업데이트, 삭제, 교체), 문서 키, 네임스페이스 정보, 사용 가능한 경우 전체 문서 콘텐츠를 포함하여 각 수정에 대한 자세한 정보가 포함되어 있습니다. 클래스는 변경 유형을 getOperationType() 결정하고, 전체 문서 상태에 액세스하고, 수정된 문서를 getDocumentKey() 식별하는 등 변경 이벤트의 다양한 측면에 getFullDocument() 액세스하는 방법을 제공합니다.

ChangeStreamDocument 객체는 두 가지 중요한 정보인 재개 토큰과 변경 이벤트 시간을 제공합니다.

DocumentDB 변경 스트림에서 토큰 재개 및 시간 기반 작업은 연속성을 유지하고 과거 데이터 액세스를 관리하기 위한 중요한 메커니즘을 제공합니다. 재개 토큰은 각 변경 이벤트에 대해 생성된 고유 식별자로, 애플리케이션이 연결 해제 또는 실패 후 특정 지점에서 변경 스트림 처리를 다시 시작할 수 있도록 허용하는 북마크 역할을 합니다. 변경 스트림 커서가 생성되면 resumeAfter() 옵션을 통해 이전에 저장된 재개 토큰을 사용하여 처음부터 시작하거나 이벤트를 손실하는 대신 스트림이 중단된 위치에서 계속될 수 있습니다.

변경 스트림의 시간 기반 작업은 변경 이벤트 모니터링의 시작점을 관리하기 위한 다양한 접근 방식을 제공합니다. startAtOperationTime() 옵션을 사용하면 특정 타임스탬프에서 또는 그 이후에 발생한 변경 사항을 볼 수 있습니다. 이러한 시간 기반 기능은 과거 데이터 처리, point-in-time으로 복구 또는 시스템 간 동기화가 필요한 시나리오에서 특히 유용합니다.

다음 코드 예제에서는 삽입 문서와 연결된 이벤트를 검색하고 재개 토큰을 캡처한 다음 해당 토큰을 제공하여 삽입 이벤트 후 이벤트 모니터링을 시작합니다. 이벤트는 업데이트 이벤트와 연결되어 업데이트가 발생한 클러스터 시간을 가져오고 해당 타임스탬프를 추가 처리를 위한 시작점으로 사용합니다.

```
BsonDocument resumeToken;
BsonTimestamp resumeTime;

try (MongoChangeStreamCursor < ChangeStreamDocument < Document >> cursor =
 collection.watch().cursor()) {
    System.out.println("****************** Insert Document ******************");
```

```
    ChangeStreamDocument < Document > insertChange = cursor.tryNext();
    resumeToken = insertChange.getResumeToken();
    printJson(cursor.tryNext());
}
try (MongoChangeStreamCursor < ChangeStreamDocument < Document >> cursor =
 collection.watch()
    .resumeAfter(resumeToken)
    .cursor()) {
    System.out.println("****************** Update Document ******************");
    ChangeStreamDocument < Document > insertChange = cursor.tryNext();
    resumeTime = insertChange.getClusterTime();
    printJson(cursor.tryNext());
}
try (MongoChangeStreamCursor < ChangeStreamDocument < Document >> cursor =
 collection.watch()
    .startAtOperationTime(resumeTime)
    .cursor()) {
    System.out.println("****************** Delete Document ******************");
    printJson(cursor.tryNext());
  }
```

기본적으로 업데이트 변경 이벤트에는 전체 문서가 포함되지 않으며 수행된 변경 사항만 포함됩니다. 업데이트된 전체 문서에 액세스해야 하는 경우 `ChangeStreamIterable` 객체에서 `fullDocument()` 메서드를 호출할 수 있습니다. 업데이트 이벤트에 대해 전체 문서를 반환하도록 요청하면 스트림 변경 호출이 이루어진 시점에 존재하는 문서가 반환된다는 점에 유의하세요.

이 메서드는 열거형`FullDocument`을 파라미터로 사용합니다. 현재 Amazon DocumentDB는 DEFAULT 및 UPDATE_LOOKUP 값만 지원합니다. 다음 코드 조각은 변경 사항을 감시하기 시작할 때 업데이트 이벤트에 대한 전체 문서를 요청하는 방법을 보여줍니다.

```
try (MongoChangeStreamCursor < ChangeStreamDocument < Document >> cursor =
 collection.watch().fullDocument(FullDocument.UPDATE_LOOKUP).cursor())
```

JSON 스키마 검증 사용

`$jsonSchema`평가 쿼리 연산자를 사용하여, 컬렉션에 삽입되는 문서를 검증할 수 있습니다.

주제

- JSON 스키마 검증 생성 및 사용
- 지원되는 키워드

- bypassDocumentValidation
- 제한 사항

JSON 스키마 검증 생성 및 사용

스키마 검증을 통한 컬렉션 만들기

`createCollection`작업 및 검증 규칙을 사용하여 컬렉션을 만들 수 있습니다. 이러한 검증 규칙은 Amazon DocumentDB 문서를 삽입하거나 업데이트할 때 적용됩니다. 다음 코드 예제에서는 직원 컬렉션에 대한 검증 규칙을 보여줍니다:

```
db.createCollection("employees", {
   "validator": {
      "$jsonSchema": {
         "bsonType": "object",
         "title": "employee validation",
         "required": [ "name", "employeeId"],
         "properties": {
            "name": {
                  "bsonType": "object",
                  "properties": {
                     "firstName": {
                        "bsonType": ["string"]
                     },
                     "lastName": {
                        "bsonType": ["string"]
                     }
                  },
                  "additionalProperties" : false
            },
            "employeeId": {
               "bsonType": "string",
               "description": "Unique Identifier for employee"
            },
             "salary": {
               "bsonType": "double"
            },
            "age": {
               "bsonType": "number"
            }
         },
         "additionalProperties" : true
```

```
      }
   },
   "validationLevel": "strict", "validationAction": "error"
} )
```

유효한 문서 삽입

다음 예제는 위의 스키마 검증 규칙을 준수하는 문서를 삽입합니다:

```
db.employees.insert({"name" : { "firstName" : "Carol" , "lastName" : "Smith"},
 "employeeId": "c720a" , "salary": 1000.0 })
db.employees.insert({ "name" : { "firstName" : "William", "lastName" : "Taylor" },
 "employeeId" : "c721a", "age" : 24})
```

잘못된 문서 삽입

다음 예제에서는 위 스키마 유효성 검사 규칙을 준수하지 않는 문서를 삽입합니다. 이 예제에서 employeeId 값은 문자열이 아닙니다:

```
db.employees.insert({
    "name" : { "firstName" : "Carol" , "lastName" : "Smith"},
    "employeeId": 720 ,
    "salary": 1000.0
})
```

이 예제는 문서 내의 잘못된 구문을 보여줍니다.

컬렉션 수정

이 `collMod` 명령은 기존 컬렉션의 검증 규칙을 추가하거나 수정하는 데 사용됩니다. 다음 예제에서는 필수 필드 목록에 급여 필드를 추가합니다:

```
db.runCommand({"collMod" : "employees",
   "validator": {
      "$jsonSchema": {
         "bsonType": "object",
         "title": "employee validation",
         "required": [ "name", "employeeId", "salary"],
         "properties": {
            "name": {
                  "bsonType": "object",
                  "properties": {
```

```
                    "firstName": {
                       "bsonType": ["string"]
                    },
                    "lastName": {
                       "bsonType": ["string"]
                    }
                 },
                 "additionalProperties" : false
           },
           "employeeId": {
              "bsonType": "string",
              "description": "Unique Identifier for employee"
           },
            "salary": {
              "bsonType": "double"
           },
           "age": {
              "bsonType": "number"
           }
        },
        "additionalProperties" : true
     }
  }
} )
```

검증 규칙이 변경되기 전에 추가된 주소 지정 문서

검증 규칙이 변경되기 전에 컬렉션에 추가된 문서의 주소를 지정하려면 다음 `validationLevel` 수정자를 사용하십시오:

- 엄격: 모든 삽입 및 업데이트에 검증 규칙을 적용합니다.
- 보통: 기존의 유효한 문서에 검증 규칙을 적용합니다. 업데이트하는 동안 기존의 잘못된 문서는 확인되지 않습니다.

다음 예제에서는 “employees”라는 이름의 컬렉션에 대한 검증 규칙을 업데이트한 후 급여 필드가 필수입니다. 다음 문서 업데이트는 실패합니다:

```
db.runCommand({
    update: "employees",
    updates: [{
        q: { "employeeId": "c721a" },
        u: { age: 25 , salary : 1000},
```

```
        upsert: true }]
})
```

Amazon DocumentDB는 다음 출력을 반환합니다:

```
{
"n" : 0,
    "nModified" : 0,
    "writeErrors" : [
        {
"index" : 0,
            "code" : 121,
            "errmsg" : "Document failed validation"
        }
    ],
    "ok" : 1,
    "operationTime" : Timestamp(1234567890, 1)
}
```

검증 수준을 `moderate`로 업데이트하면 위 문서가 성공적으로 업데이트될 수 있습니다:

```
db.runCommand({
    "collMod" : "employees",
    validationLevel : "moderate"
})

db.runCommand({
    update: "employees",
    updates: [{
        q: { "employeeId": "c721a" },
        u: { age: 25 , salary : 1000},
        upsert: true }]
})
```

Amazon DocumentDB는 다음 출력을 반환합니다:

```
{
"n" : 1,
    "nModified" : 1,
    "ok" : 1,
    "operationTime" : Timestamp(1234567890, 1)
}
```

$jsonSchema를 사용하여 문서 검색

$jsonSchema연산자를 필터로 사용하여 JSON schema와 일치하는 문서를 쿼리할 수 있습니다. 이것은 필터 문서에 최상위 필드로 존재하거나 $and, $or, 및 $nor과 같은 쿼리 연산자와 함께 사용할 수 있는 최상위 연산자입니다. 다음 예제는 $JSONSchema를 개별 필터 및 기타 필터 연산자와 함께 사용하는 방법을 보여줍니다:

"직원" 컬렉션에 삽입된 문서:

```
{ "name" : { "firstName" : "Carol", "lastName" : "Smith" }, "employeeId" : "c720a",
 "salary" : 1000 }
{ "name" : { "firstName" : "Emily", "lastName" : "Brown" }, "employeeId" : "c720b",
 "age" : 25, "salary" : 1050.2 }
{ "name" : { "firstName" : "William", "lastName" : "Taylor" }, "employeeId" : "c721a",
 "age" : 24, "salary" : 1400.5 }
{ "name" : { "firstName" : "Jane", "lastName" : "Doe" }, "employeeId" : "c721a",
 "salary" : 1300 }
```

$jsonSchema연산자로만 필터링된 컬렉션:

```
db.employees.find({
       $jsonSchema: { required: ["age"] } })
```

Amazon DocumentDB는 다음 출력을 반환합니다:

```
{ "_id" : ObjectId("64e5f91c6218c620cf0e8f8b"), "name" : { "firstName" : "Emily",
 "lastName" : "Brown" }, "employeeId" : "c720b", "age" : 25, "salary" : 1050.2 }
{ "_id" : ObjectId("64e5f94e6218c620cf0e8f8c"), "name" : { "firstName" : "William",
 "lastName" : "Taylor" }, "employeeId" : "c721a", "age" : 24, "salary" : 1400.5 }
```

$jsonSchema연산자와 다른 연산자로 필터링된 컬렉션:

```
db.employees.find({
       $or: [{ $jsonSchema: { required: ["age", "name"]}},
            { salary: { $lte:1000}}]});
```

Amazon DocumentDB는 다음 출력을 반환합니다:

```
{ "_id" : ObjectId("64e5f8886218c620cf0e8f8a"), "name" : { "firstName" : "Carol",
 "lastName" : "Smith" }, "employeeId" : "c720a", "salary" : 1000 }
```

```
{ "_id" : ObjectId("64e5f91c6218c620cf0e8f8b"), "name" : { "firstName" : "Emily",
 "lastName" : "Brown" }, "employeeId" : "c720b", "age" : 25, "salary" : 1050.2 }
{ "_id" : ObjectId("64e5f94e6218c620cf0e8f8c"), "name" : { "firstName" : "William",
 "lastName" : "Taylor" }, "employeeId" : "c721a", "age" : 24, "salary" : 1400.5 }
```

$jsonSchema 연산자로 그리고 $match로 집합 필터에서 필터링된 집합:

```
db.employees.aggregate(
    [{ $match: {
        $jsonSchema: {
            required: ["name", "employeeId"],
            properties: {"salary" :{"bsonType": "double"}}
        }
       }
    }]
)
```

Amazon DocumentDB는 다음 출력을 반환합니다:

```
{
"_id" : ObjectId("64e5f8886218c620cf0e8f8a"),
 "name" : { "firstName" : "Carol", "lastName" : "Smith" },
"employeeId" : "c720a",
"salary" : 1000
}
{
"_id" : ObjectId("64e5f91c6218c620cf0e8f8b"),
"name" : { "firstName" : "Emily", "lastName" : "Brown" },
"employeeId" : "c720b",
"age" : 25,
"salary" : 1050.2
}
{
"_id" : ObjectId("64e5f94e6218c620cf0e8f8c"),
"name" : { "firstName" : "William", "lastName" : "Taylor" },
"employeeId" : "c721a",
"age" : 24,
"salary" : 1400.5
}
{
"_id" : ObjectId("64e5f9786218c620cf0e8f8d"),
"name" : { "firstName" : "Jane", "lastName" : "Doe" },
"employeeId" : "c721a",
```

```
"salary" : 1300
}
```

기존 검증 규칙 보기

컬렉션의 기존 검증 규칙을 보려면 다음을 사용하세요:

```
db.runCommand({
    listCollections: 1,
    filter: { name: 'employees' }
})
```

Amazon DocumentDB는 다음 출력을 반환합니다:

```
{
    "waitedMS" : NumberLong(0),
    "cursor" : {
        "firstBatch" : [
            {
                "name" : "employees",
                "type" : "collection",
                "options" : {
                    "autoIndexId" : true,
                    "capped" : false,
                    "validator" : {
                        "$jsonSchema" : {
                            "bsonType" : "object",
                            "title" : "employee validation",
                            "required" : [
                                "name",
                                "employeeId",
                                "salary"
                            ],
                            "properties" : {
                                "name" : {
                                    "bsonType" : "object",
                                    "properties" : {
                                        "firstName" : {
                                            "bsonType" : [
                                                "string"
                                            ]
                                        },
                                        "lastName" : {
```

```
                                        "bsonType" : [
                                            "string"
                                        ]
                                    }
                                },
                                "additionalProperties" : false
                            },
                            "employeeId" : {
                                "bsonType" : "string",
                                "description" : "Unique Identifier for employee"
                            },
                            "salary" : {
                                "bsonType" : "double"
                            },
                            "age" : {
                                "bsonType" : "number"
                            }
                        },
                        "additionalProperties" : true
                    }
                },
                "validationLevel" : "moderate",
                "validationAction" : "error"
            },
            "info" : {
                "readOnly" : false
            },
            "idIndex" : {
                "v" : 2,
                "key" : {
                    "_id" : 1
                },
                "name" : "_id_",
                "ns" : "test.employees"
            }
        }
    ],
    "id" : NumberLong(0),
    "ns" : "test.$cmd.listCollections"
},
"ok" : 1,
"operationTime" : Timestamp(1692788937, 1)
}
```

Amazon DocumentDB는 또한 $out 집계 단계에서 검증 규칙을 유지합니다.

지원되는 키워드

create및 collMod 명령에서 지원되는 필드는 다음과 같습니다.

- **Validator**— $jsonSchema 연산자를 지원합니다.
- **ValidationLevel**— off, strict, 및 moderate 값을 지원합니다.
- **ValidationAction**— error 값을 지원합니다.

$jsonSchema 연산자는 다음 키워드를 지원합니다:

- additionalItems
- additionalProperties
- allOf
- anyOf
- bsonType
- dependencies
- description
- enum
- exclusiveMaximum
- exclusiveMinimum
- items
- maximum
- minimum
- maxItems
- minItems
- maxLength
- minLength
- maxProperties
- minProperties
- multipleOf
- not

- oneOf
- pattern
- patternProperties
- properties
- required
- title
- type
- uniqueItems

bypassDocumentValidation

Amazon DocumentDB는 다음 명령 및 메서드에 대해 bypassDocumentValidation을 지원합니다.

- insert
- update
- findAndModify
- aggregate 명령 및 db.collection.aggregate() 메서드의 $out 단계

Amazon DocumentDB는 bypassDocumentValidation에 대해 다음 명령을 지원하지 않습니다.

- aggregate 명령 및 db.collection.aggregate() 메서드의 $merge
- mapReduce 명령 및 db.collection.mapReduce() 메서드
- applyOps 명령

제한 사항

$jsonSchema에는 다음과 같은 제한 사항이 적용됩니다.

- Amazon DocumentDB는 작업이 검증 규칙에 실패하면 “문서 검증에 실패했습니다.”라는 오류를 반환합니다.
- Amazon DocumentDB 탄력적 클러스터는 $jsonSchema를 지원하지 않습니다.

Amazon DocumentDB 할당량 및 제한

이 주제에서는 Amazon DocumentDB(MongoDB 호환)의 리소스 할당량, 제한 및 명명 제약 조건에 대해 설명합니다.

일부 관리 기능의 경우 Amazon DocumentDB는 Amazon Relational Database Service(RDS) 및 Amazon Neptune과 공유되는 운영 기술을 사용합니다.

주제

- 지원되는 인스턴스 유형
- 지원되는 리전
- 리전별 할당량
- 집계 한도
- 클러스터 한도
- 인스턴스 제한
- 명명 제약 조건
- TTL 제약 조건
- 탄력적 클러스터 제한
- 탄력적 클러스터 샤드 한도
- 샤드당 탄력적 클러스터 CPU, 메모리, 연결 및 커서 제한

지원되는 인스턴스 유형

Amazon DocumentDB는 온디맨드 인스턴스와 다음의 인스턴스 유형을 지원합니다.

- NVMe 지원:
  - R6GD 인스턴스 유형: `db.r6gd.xlarge`, `db.r6gd.2xlarge`, `db.r6gd.4xlarge`, `db.r6gd.8xlarge`, `db.r6gd.12xlarge`, `db.r6gd.16xlarge`.
- 메모리 최적화:
  - R6G 인스턴스 유형: `db.r6g.large`, `db.r6g.2xlarge`, `db.r6g.4xlarge`, `db.r6g.8xlarge`, `db.r6g.12xlarge`, `db.r6g.16xlarge`.
  - R5 인스턴스 유형: `db.r5.large`, `db.r5.2xlarge`, `db.r5.4xlarge`, `db.r5.8xlarge`, `db.r5.12xlarge`, `db.r5.16xlarge`, `db.r5.24xlarge`.

  - R4 인스턴스 유형: `db.r4.large`, `db.r4.2xlarge`, `db.r4.4xlarge`, `db.r4.8xlarge`, `db.r4.16xlarge`.
- 버스트 가능한 성능:
  - T4G 인스턴스 유형: `db.t4g.medium`.
  - T3 인스턴스 유형: `db.t3.medium`.

지원되는 인스턴스 유형 및 사양에 대한 자세한 내용은 인스턴스 클래스 사양 섹션을 참조하십시오.

지원되는 리전

Amazon DocumentDB는 다음 AWS 리전에서 사용할 수 있습니다.

리전 이름	지역	가용 영역(컴퓨팅)
미국 동부(오하이오)	us-east-2	3
미국 동부(버지니아 북부)	us-east-1	6
미국 서부(오리건)	us-west-2	4
아프리카(케이프타운)	af-south-1	3
남아메리카(상파울루)	sa-east-1	3
아시아 태평양(홍콩)	ap-east-1	3
아시아 태평양(하이데라바드)	ap-south-2	3
아시아 태평양(뭄바이)	ap-south-1	3
아시아 태평양(서울)	ap-northeast-2	4
아시아 태평양(싱가포르)	ap-southeast-1	3
아시아 태평양(시드니)	ap-southeast-2	3
아시아 태평양(도쿄)	ap-northeast-1	3
캐나다(중부)	ca-central-1	3
중국(베이징) 리전	cn-north-1	3
중국(닝샤)	cn-northwest-1	3
유럽(프랑크푸르트)	eu-central-1	3

리전 이름	지역	가용 영역(컴퓨팅)
유럽(아일랜드)	eu-west-1	3
유럽(런던)	eu-west-2	3
유럽(밀라노)	eu-south-1	3
유럽(파리)	eu-west-3	3
유럽(스페인)	eu-south-2	3
유럽(스톡홀름)	eu-north-1	3
중동(UAE)	me-central-1	3
이스라엘(텔아비브)	il-central-1	3
AWS GovCloud(미국 서부)	us-gov-west-1	3
AWS GovCloud(미국 동부)	us-gov-east-1	3

리전별 할당량

일부 관리 기능의 경우 Amazon DocumentDB는 Amazon Relational Database Service(RDS)와 공유되는 운영 기술을 사용합니다. 다음 표에는 Amazon DocumentDB와 Amazon RDS 사이에 공유하는 리전 제한이 나와 있습니다.

> **Note**
>
> 위에서 설명한 Amazon RDS 공유 기술은 Amazon DocumentDB 인스턴스 기반 클러스터에만 적용됩니다. Amazon DocumentDB 탄력적 클러스터는 Amazon RDS와 기술을 공유하지 않습니다.

다음 제한은 Amazon DocumentDB 인스턴스 기반 클러스터에 적용되며 리전별 AWS 계정당 적용됩니다.

리소스	AWS 기본 제한
클러스터	40
클러스터 파라미터 그룹	50
이벤트 구독	20
인스턴스	40
수동 클러스터 스냅샷 수	100
클러스터당 읽기 전용 복제본	15
서브넷 그룹 수	50
서브넷 그룹 1개당 서브넷 수	20
리소스당 태그	50
인스턴스당 VPC 보안 그룹	5

다음 제한은 Amazon DocumentDB 탄력적 클러스터에 적용되며 리전별 AWS 계정당 적용됩니다.

리소스	AWS 기본 제한
엘라스틱 클러스터	20
탄력적 클러스터 vCPU	1024
수동 탄력적 클러스터 스냅샷	20

할당량을 조정할 수 있는 경우 Service Quotas를 사용하여 할당량 증가를 요청할 수 있습니다. 일부 요청은 자동으로 해결되고 다른 요청은 제출됩니다 지원. 제출된 할당량 증가 요청의 상태를 추적할 수 있습니다 지원. Service Quotas 증가 요청은 우선 순위 지원을 받지 못합니다. 긴급한 요청이 있는 경우 지원에 문의하십시오. Service Quotas에 대한 자세한 내용은 Service Quotas는 무엇입니까?를 참조하십시오.

Amazon DocumentDB에 대한 할당량 증가를 요청하려면

1. https://console.aws.amazon.com/servicequotas에서 Service Quotas 콘솔을 열고 필요한 경우 로그인합니다.
2. 탐색 창에서 AWS 서비스를 선택합니다.
3. 목록에서 Amazon DocumentDB(MongoDB 호환) 또는 Amazon DocumentDB 탄력적 클러스터를 선택하거나 검색 필드에 어느 하나를 입력합니다.
4. 할당량을 조정할 수 있는 경우 해당 라디오 버튼 또는 이름을 선택한 다음 페이지 오른쪽 상단에 있는 할당량 증가 요청를 선택할 수 있습니다.
5. 할당량 값 변경에 새 값을 입력합니다. 새 값은 현재 값보다 커야 합니다.
6. 요청을 선택합니다. 요청이 해결되면 할당량에 대한 적용된 할당량 값이 새 값으로 설정됩니다.
7. 보류 중이거나 최근에 해결된 요청을 보려면 탐색 창에서 대시보드를 선택합니다. 보류 중인 요청의 경우 요청 상태를 선택하여 요청 접수증을 엽니다. 요청의 초기 상태는 `Pending`입니다. 상태가 로 변경되면 `Quota requested`사례 번호가 표시됩니다 지원. 이 케이스 번호를 선택하여 요청의 티켓을 엽니다.

집계 한도

다음 표에서는 Amazon DocumentDB의 집계 한도에 대해 설명합니다.

리소스	Limit
지원되는 스테이지의 최대 개수	500

클러스터 한도

다음 표에서는 Amazon DocumentDB 인스턴스 기반 클러스터 한도에 대해 설명합니다.

리소스	Limit
클러스터 크기(모든 컬렉션 및 인덱스의 합계)	128TiB
컬렉션 크기(모든 컬렉션의 합계는 클러스터 제한을 초과할 수 없음) – 인덱스 크기를 포함하지 않음	32TiB
클러스터당 모음 수	100,000건
클러스터당 데이터베이스 수	100,000건
데이터베이스 크기(모든 데이터베이스의 합계는 클러스터 제한을 초과할 수 없음)	128TiB
문서 중첩 깊이	200개 수준
문서 크기	16MiB
인덱스 키 크기	2,048바이트
모음당 인덱스 수	64
복합 인덱스의 키	32
단일 배치 명령 내 최대 읽기 수	100,000건
클러스터당 사용자 수	1000

인스턴스 제한

다음 표에서는 인스턴스당 Amazon DocumentDB 한도에 대해 설명합니다.

인스턴스 유형	인스턴스 메모리(GiB)	연결(모두)	커서 제한	트랜잭션 열기	연결(활성)
T3.medium	4	1000	30	50	102
T4G.medium	4	1000	30	50	102
R4.large	15.25	1700	450	N/A	1100
R4.xlarge	30.5	3400	450	N/A	2700
R4.2xlarge	61	6800	450	N/A	4500
R4.4xlarge	122	13600	725	N/A	4500
R4.8xlarge	288	27200	1450	N/A	4500
R4.16xlarge	488	30000	2900	N/A	4500
R5.large	16	3400	450	200	1100
R5.xlarge	32	7000	450	400	2700
R5.2xlarge	64	14200	450	800	4500
R5.4xlarge	128	28400	760	1600	4500
R5.8xlarge	256	60000	1520	3200	4500
R5.12xlarge	384	60000	2280	4800	4500
R5.16xlarge	512	60000	3040	6400	4500
R5.24xlarge	768	60000	4560	9600	4500
R6G.large*	16	3400	450	200	1100

인스턴스 유형	인스턴스 메모리(GiB)	연결(모두)	커서 제한	트랜잭션 열기	연결(활성)
R6G.xlarge*	32	7000	450	400	2700
R6G.2xlarge*	64	14200	450	800	4500
R6G.4xlarge*	128	28400	760	1600	4500
R6G.8xlarge*	256	60000	1520	3200	4500
R6G.12xlarge*	384	60000	2280	4800	4500
R6G.16xlarge*	512	60000	3040	6400	4500

* R6GD 포함

다음 CloudWatch 지표를 사용하여 인스턴스당 제한을 모니터링하고 경보를 생성할 수 있습니다. Amazon DocumentDB CloudWatch 지표에 관한 더 많은 내용은 CloudWatch를 사용하여 Amazon DocumentDB 모니터링를 참조하세요.

Limit	CloudWatch 지표
인스턴스 메모리	FreeableMemory
연결	DatabaseConnectionsMax
커서	DatabaseCursorsMax
트랜잭션	TransactionsOpenMax

명명 제약 조건

다음 표에서는 Amazon DocumentDB의 명명 제약 조건을 설명합니다.

리소스	기본 제한
클러스터 식별자	• 길이는 [1~63] 글자, 숫자 또는 하이픈입니다. • 첫 번째 문자는 글자이어야 합니다. • 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다. • 리전별로 AWS 계정당 모든 클러스터(Amazon RDS, Amazon Neptune 및 Amazon DocumentDB 전체)에 대해 고유해야 합니다.
컬렉션 이름: `<col>`	길이는 [1~57]자입니다.
데이터베이스 이름: `<db>`	길이는 [1~63]자입니다.
정규화된 컬렉션 이름: `<db>.<col>`	길이는 [3~120]자입니다.
정규화된 인덱스 이름: `<db>.<col>$<index>`	길이는 [6~377]자입니다.
인덱스 이름: `<col>$<index>`	길이는 [3~255]자입니다.
인스턴스 식별자	• 길이는 [1~63] 글자, 숫자 또는 하이픈입니다. • 첫 번째 문자는 글자이어야 합니다 • 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다 • 리전별로 AWS 계정당 모든 인스턴스(Amazon RDS, Amazon

리소스	기본 제한
	Neptune 및 Amazon DocumentDB)에 대해 고유해야 합니다.

기본 암호	• 길이는 [8-100]자의 인쇄 가능한 ASCII 문자입니다.
	• 다음을 제외한 인쇄 가능한 ASCII 문자를 사용할 수 있습니다.
	  • **/**(슬래시)
	  • **"**(큰 따옴표)
	  • **@**(at 기호)

기본 사용자 이름	• 길이는 [1-63]자의 영숫자 문자입니다. • 첫 번째 문자는 글자이어야 합니다. • 데이터베이스 엔진에 포함된 단어는 사용할 수 없습니다.
파라미터 그룹 이름	• [1-255]자 길이의 영숫자 문자입니다. • 첫 번째 문자는 글자이어야 합니다. • 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다.

TTL 제약 조건

TTL 인덱스에서의 삭제는 특정 기간 내에 삭제된다고 보장할 수 없으며 가능한 한 빠른 시간 내에 수행됩니다. 인스턴스 리소스 사용률, 문서 크기 및 전체 처리량과 같은 요소가 TTL 삭제 소요 시간에 영향을 줄 수 있습니다.

탄력적 클러스터 제한

다음 표에서는 Amazon DocumentDB 탄력적 클러스터의 최대 한도에 대해 설명합니다.

리소스	Limit
리전당 탄력적 클러스터	20
리전당 모든 탄력적 클러스터에 걸쳐 합산된 vCPU	1024
리전당 수동 클러스터 스냅샷	20
클러스터당 샤드 수	32
클러스터당 스토리지(샤드 키로 데이터가 균등하게 분산되는 경우)	4PiB
클러스터에 연결	300,000의 더 낮은 값 또는 샤드 수 x 샤드당 vCPU와 연결된 연결 한도
샤딩되지 않은 컬렉션 크기	32TiB
샤딩된 컬렉션 크기(샤드 키로 데이터가 균등하게 배포되는 경우)	1PB
클러스터당 데이터베이스 수	10,000
클러스터당 샤딩되지 않은 컬렉션	100,000건
클러스터당 샤딩된 컬렉션	1000
클러스터 당 사용자	100
단일 배치 명령 내 읽기	100,000건
모음당 인덱스 수	64
문서 중첩 깊이	100개 수준
문서 크기	16MB

리소스	Limit
인덱스 키 크기	2048 바이트
복합 인덱스의 키	32

탄력적 클러스터 샤드 한도

다음 표에서는 Amazon DocumentDB 탄력적 클러스터의 최대 샤드 제한에 대해 설명합니다.

리소스	Limit
샤드 인스턴스당 vCPU	64
샤드당 인스턴스 수	16
샤드당 스토리지	128TiB
샤드당 컬렉션별 스토리지	32TiB

샤드당 탄력적 클러스터 CPU, 메모리, 연결 및 커서 제한

다음 표에서는 Amazon DocumentDB 탄력적 클러스터 샤드의 최대 CPU, 메모리, 연결 및 커서 제한을 설명합니다.

샤드당 vCPU	인스턴스 메모리(GiB)	연결 제한	커서 제한
2	16	1700	450
4	32	3500	450
8	64	7100	450
16	128	14200	760
32	256	28400	1520
48	384	30000	2280

샤드당 vCPU	인스턴스 메모리(GiB)	연결 제한	커서 제한
64	512	30000	3040

Amazon DocumentDB에서 쿼리

이 섹션에서는 Amazon DocumentDB를 사용한 쿼리의 모든 측면을 설명합니다.

주제

- 문서 쿼리
- 쿼리 계획
- 결과의 설명
- Amazon DocumentDB에서 지리 공간 데이터 쿼리
- 부분 인덱스
- Amazon DocumentDB를 사용하여 텍스트 검색 수행

문서 쿼리

이따금 판매하는 물품을 고객이 보고 구매할 수 있도록 온라인 상점의 재고를 조회해야 할 수도 있습니다. 컬렉션을 쿼리하는 것은 컬렉션의 모든 문서에 대한 것이든 특정 기준을 충족하는 문서에만 대한 것이든 관계없이 상대적으로 간단합니다.

문서를 쿼리하려면 `find()` 작업을 사용합니다. `find()` 명령에는 반환할 문서 선택 시 사용할 기준을 정의하는 단일 문서 파라미터가 있습니다. `find()`의 출력은 줄 바꿈이 없이 한 줄로 된 텍스트 형식의 문서입니다. 쉽게 읽기 위해 출력 문서의 형식을 지정하려면 `find().pretty()`를 사용하십시오. 이 주제의 모든 예제는 `.pretty()`를 사용하여 출력 형식을 지정합니다.

다음 코드 샘플은 이전 두 연습에서 `example` 컬렉션에 삽입한 문서 4개를 사용합니다. `insertOne()` 및 `insertMany()`는 문서 작업의 문서 추가 섹션에 있습니다.

주제

- 컬렉션의 모든 문서 검색
- 피드 값과 일치하는 문서 검색
- 내장 문서와 일치하는 문서 검색
- 내장 문서의 필드 값과 일치하는 문서 검색
- 배열이 일치하는 문서 검색
- 배열의 값과 일치하는 문서 검색

- 연산자를 사용하여 문서 검색

컬렉션의 모든 문서 검색

모음의 모든 문서를 검색하려면 빈 쿼리 문서로 `find()` 작업을 사용하십시오.

다음 쿼리는 `example` 모음의 모든 문서를 반환합니다.

```
db.example.find( {} ).pretty()
```

피드 값과 일치하는 문서 검색

필드 및 값과 일치하는 모든 문서를 검색하려면 일치하는 필드 및 값을 식별하는 쿼리 문서로 `find()` 작업을 사용합니다.

이전 문서를 사용하면 이 쿼리는 "Item" 필드가 "Pen"과 동일한 모든 문서를 반환합니다.

```
db.example.find( { "Item": "Pen" } ).pretty()
```

내장 문서와 일치하는 문서 검색

내장 문서와 일치하는 모든 문서를 찾으려면 내장 문서 이름과 내장 문서의 모든 필드 및 값을 지정하는 쿼리 문서와 함께 `find()` 작업을 사용합니다.

내장 문서를 일치시킬 때 문서의 내장 문서는 쿼리에 있는 것과 동일한 이름을 가져야 합니다. 또한, 내장 문서의 필드 및 값은 쿼리와 일치해야 합니다.

다음 쿼리는 "Poster Paint" 문서만 반환합니다. "Pen"에는 "OnHand" 및 "MinOnHand"에 대한 다른 값이 있으며, "Spray Paint"에는 쿼리 문서보다 필드 하나(`OrderQnty`)가 더 있기 때문입니다.

```
db.example.find({"Inventory": {
    "OnHand": 47,
    "MinOnHand": 50 } } ).pretty()
```

내장 문서의 필드 값과 일치하는 문서 검색

내장 문서와 일치하는 모든 문서를 찾으려면 내장 문서 이름과 내장 문서의 모든 필드 및 값을 지정하는 쿼리 문서와 함께 `find()` 작업을 사용합니다.

이전 문서에서 다음 쿼리는 "점 표기법"을 사용하여 내장 문서와 관심 있는 필드를 지정합니다. 다른 필드가 내장 문서에 표시될 수 있는지 여부와 무관하게 이와 일치하는 모든 문서가 반환됩니다. "Poster Paint" 및 "Spray Paint"가 지정된 필드 및 값과 일치하므로 쿼리가 "Poster Paint" 및 "Spray Paint"를 반환합니다.

```
db.example.find({"Inventory.OnHand": 47, "Inventory.MinOnHand": 50 }).pretty()
```

배열이 일치하는 문서 검색

배열이 일치하는 모든 문서를 찾으려면 관심 있는 배열 이름과 해당 배열의 모든 값을 포함하여 `find()` 작업을 사용합니다. 쿼리가 배열 값이 동일하면서 쿼리와 동일한 순서인 해당 이름을 가진 배열을 포함한 모든 문서를 반환합니다.

"Poster Paint"에는 추가 색상(White)이 있고 "Spray Paint"에는 색상이 다른 순서로 있으므로 다음 쿼리는 "Pen"만을 반환합니다.

```
db.example.find( { "Colors": ["Red","Green","Blue","Black"] } ).pretty()
```

배열의 값과 일치하는 문서 검색

특정 배열 값을 보유한 모든 문서를 찾으려면 관심 있는 배열 이름과 값을 포함하여 `find()` 작업을 사용합니다.

```
db.example.find( { "Colors": "Red" } ).pretty()
```

각각 `Colors`라는 배열과 배열 내에 "`Red`" 값이 있으므로 이전 작업은 세 문서 모두를 반환합니다. "`White`" 값을 지정한 경우 쿼리는 "Poster Paint"만 반환합니다.

연산자를 사용하여 문서 검색

다음 쿼리는 "`Inventory.OnHand`" 값이 50 미만인 모든 문서를 반환합니다.

```
db.example.find(
        { "Inventory.OnHand": { $lt: 50 } } )
```

지원되는 쿼리 연산자 목록은 쿼리 및 프로젝션 연산자 섹션을 참조하십시오.

쿼리 계획

쿼리 계획에 대한 **executionStats**를 보려면 어떻게 해야 합니까?

쿼리가 예상보다 느리게 실행되는 이유를 결정할 때 쿼리 계획에 대한 executionStats가 무엇인지 이해하는 것이 유용할 수 있습니다. executionStats에서는 특정 단계에서 반환된 문서 수(nReturned), 각 단계에서 소요된 실행 시간(executionTimeMillisEstimate) 및 쿼리 계획을 생성하는 데 걸리는 시간(planningTimeMillis)을 제공합니다. 아래 쿼리 예제와 같이 가장 시간이 많이 걸리는 쿼리 단계를 결정하여 executionStats의 출력에서 최적화 작업에 집중할 수 있습니다. executionStats 파라미터는 현재 update 및 delete 명령을 지원하지 않습니다.

> **Note**
>
> Amazon DocumentDB는 분산, 내결함성, 자가 치유 스토리지 시스템을 활용하는 목적으로 만들어진 데이터베이스 엔진에서 MongoDB 3.6 API를 에뮬레이션합니다. 그 결과, 쿼리 계획과 explain()의 출력은 Amazon DocumentDB와 MongoDB 간에 다를 수 있습니다. 쿼리 계획을 제어하려는 고객은 $hint 연산자를 사용하여 기본 인덱스를 선택할 수 있습니다.

다음과 같이 explain() 명령에서 개선하려는 쿼리를 실행합니다.

```
db.runCommand({explain: {query document}}).
explain("executionStats").executionStats;
```

예는 다음과 같습니다.

```
db.fish.find({}).limit(2).explain("executionStats");
```

이 작업의 출력은 다음과 같이 표시됩니다.

```
{
    "queryPlanner" : {
        "plannerVersion" : 1,
        "namespace" : "test.fish",
        "winningPlan" : {
            "stage" : "SUBSCAN",
            "inputStage" : {
                "stage" : "LIMIT_SKIP",
```

```
                "inputStage" : {
                    "stage" : "COLLSCAN"
                }
            }
        }
    },
    "executionStats" : {
        "executionSuccess" : true,
        "executionTimeMillis" : "0.063",
        "planningTimeMillis" : "0.040",
        "executionStages" : {
            "stage" : "SUBSCAN",
            "nReturned" : "2",
            "executionTimeMillisEstimate" : "0.012",
            "inputStage" : {
                "stage" : "LIMIT_SKIP",
                "nReturned" : "2",
                "executionTimeMillisEstimate" : "0.005",
                "inputStage" : {
                    "stage" : "COLLSCAN",
                    "nReturned" : "2",
                    "executionTimeMillisEstimate" : "0.005"
                }
            }
        }
    },
    "serverInfo" : {
        "host" : "enginedemo",
        "port" : 27017,
        "version" : "3.6.0"
    },
    "ok" : 1
}
```

위의 쿼리에서 executionStats만 보는 데 관심이 있다면 다음 명령을 사용할 수 있습니다. 작은 컬렉션에서는 성능 이득이 무시할 수준인 경우 Amazon DocumentDB 쿼리 프로세서는 인덱스를 사용하지 않도록 결정할 수 있습니다.

```
db.fish.find({}).limit(2).explain("executionStats").executionStats;
```

쿼리 계획 캐시

성능을 최적화하고 계획 기간을 줄이기 위해 Amazon DocumentDB는 쿼리 계획을 내부적으로 캐싱합니다. 이렇게 하면 캐시된 계획을 사용하여 동일한 모양의 쿼리를 직접 실행할 수 있습니다.

하지만 이 캐싱으로 인해 동일한 쿼리에 대해 무작위 지연이 발생할 수 있습니다. 예를 들어 쿼리를 실행하는 데 보통 1초가 걸리는 경우 가끔 10초가 걸릴 수 있습니다. 이는 시간이 지나면서 reader 인스턴스가 다양한 형태의 쿼리를 캐시하여 메모리를 소비하기 때문입니다. 이러한 무작위 속도 저하가 발생하는 경우 메모리를 해제하기 위해 별도의 조치를 취하지 않아도 됩니다. 시스템에서 자동으로 메모리 사용량을 관리하고 메모리가 특정 임계값에 도달하면 자동으로 해제됩니다.

결과의 설명

쿼리 계획에 대한 정보를 반환하려는 경우 Amazon DocumentDB는 세부 정보 표시 모드 `queryPlanner`를 지원합니다. `explain` 결과는 옵티마이저가 선택한 쿼리 계획을 다음과 비슷한 형식으로 반환합니다.

```
{
   "queryPlanner" : {
      "plannerVersion" : <int>,
      "namespace" : <string>,
      "winningPlan" : {
         "stage" : <STAGE1>,
         ...
         "inputStage" : {
            "stage" : <STAGE2>,
            ...
            "inputStage" : {
               ...
            }
         }
      }
   }
}
```

다음 섹션에서는 일반적인 `explain` 결과를 정의합니다.

주제

- 스캔 및 필터 단계

- 인덱스 교집합
- 인덱스 병합
- 다중 인덱스 교집합/병합
- 복합 인덱스
- 정렬 단계
- 그룹 단계

스캔 및 필터 단계

옵티마이저는 다음 스캔 중 하나를 선택할 수 있습니다.

콜스캔

이 단계는 순차적 수집 스캔입니다.

```
{
    "stage" : "COLLSCAN"
}
```

ISSCAN

이 단계에서는 인덱스 키를 스캔합니다. 옵티마이저는 이 단계 내 문서를 검색할 수 있으며, 이 경우 나중에 FETCH 단계가 추가될 수 있습니다.

```
db.foo.find({"a": 1})
{
    "stage" : "IXSCAN",
    "direction" : "forward",
    "indexName" : <idx_name>
}
```

FETCH

옵티마이저가 IXSCAN 이외의 단계에서 문서를 검색한 경우 결과에는 FETCH 단계가 포함됩니다. 예를 들어, 위의 IXSCAN 쿼리는 FETCH와 IXSCAN 단계가 조합된 결과를 초래할 수 있습니다.

```
db.foo.find({"a": 1})
```

```
{
    "stage" : "FETCH",
    "inputStage" : {
        "stage" : "IXSCAN",
        "indexName" : <idx_name>
    }
}
```

IXONLYSCAN은 인덱스 키만 스캔합니다. 복합 인덱스를 생성해도 FETCH를 피할 수는 없습니다.

인덱스 교집합

아일랜드

Amazon DocumentDB에는 인덱스 교차를 활용할 수 있는 경우 IXScan의 InputStage 배열이 포함된 IXAND 스테이지가 포함될 수 있습니다. 예를 들어 다음과 같은 출력이 표시될 수 있습니다.

```
{
    "stage" : "FETCH",
    "inputStage" : {
        "stage" : "IXAND",
        "inputStages" : [
            {
                "stage" : "IXSCAN",
                "indexName" : "a_1"
            },
            {
                "stage" : "IXSCAN",
                "indexName" : "b_1"
            }
        ]
    }
}
```

인덱스 병합

익소르

인덱스 교차와 마찬가지로 Amazon DocumentDB에는 `$or` 연산자를 위한 `inputStages` 배열이 있는 `IXOR` 스테이지가 포함될 수 있습니다.

```
db.foo.find({"$or": [{"a": {"$gt": 2}}, {"b": {"$lt": 2}}]})
```

위 쿼리의 경우 설명 출력은 다음과 같을 수 있습니다.

```
{
    "stage" : "FETCH",
    "inputStage" : {
        "stage" : "IXOR",
        "inputStages" : [
            {
                "stage" : "IXSCAN",
                "indexName" : "a_1"
            },
            {
                "stage" : "IXSCAN",
                "indexName" : "b_1"
            }
        ]
    }
}
```

다중 인덱스 교집합/병합

Amazon DocumentDB는 여러 인덱스 교차 또는 유니온 단계를 함께 결합한 다음 결과를 가져올 수 있습니다. 예시:

```
{
    "stage" : "FETCH",
    "inputStage" : {
        "stage" : "IXOR",
        "inputStages" : [
            {
                "stage" : "IXSCAN",
                ...
            },
            {
                "stage" : "IXAND",
                "inputStages" : [
                    {
                        "stage" : "IXSCAN",
                        ...
```

```
                    },
                    {
                        "stage" : "IXSCAN",
                        ...
                    }
                ]
            }
        ]
    }
}
```

인덱스 교차 또는 유니온 스테이지의 사용은 인덱스 유형(스파스, 복합 등)의 영향을 받지 않습니다.

복합 인덱스

Amazon DocumentDB 복합 인덱스 사용은 인덱싱된 필드의 시작 부분에만 국한되지 않습니다. 접미사 부분이 있는 인덱스를 사용할 수 있지만 그다지 효율적이지 않을 수 있습니다.

예를 들어, `{ a: 1, b: -1 }`의 복합 인덱스는 아래 세 가지 쿼리를 모두 지원할 수 있습니다.

```
db.orders.find( { a: 1 } )

db.orders.find( { b: 1 } )

db.orders.find( { a: 1, b: 1 } )
```

정렬 단계

요청된 정렬 키에 인덱스가 있는 경우 Amazon DocumentDB는 인덱스를 사용하여 순서를 확인할 수 있습니다. 이 경우 결과에는 `SORT` 단계가 아닌 `IXSCAN` 단계가 포함됩니다. 옵티마이저가 일반 정렬을 선호하는 경우 다음과 같은 단계가 포함됩니다.

```
{
    "stage" : "SORT",
    "sortPattern" : {
        "a" : 1,
        "b" : -1
    }
}
```

그룹 단계

Amazon DocumentDB는 다음과 같은 두 가지 그룹 전략을 지원합니다.

- SORT_AGGREGATE: 온 디스크 정렬 집계.
- HASH_AGGREGATE: 인 메모리 해시 집계.

Amazon DocumentDB에서 지리 공간 데이터 쿼리

이 섹션에서는 Amazon DocumentDB를 사용하여 지리공간 데이터를 쿼리하는 방법을 다룹니다. 이 섹션을 읽고 나면 Amazon DocumentDB에서 지리공간 데이터를 저장, 쿼리 및 인덱싱하는 방법에 대한 답을 얻을 수 있을 것입니다.

주제

- 개요
- 지리공간 데이터 인덱싱 및 저장
- 지리 공간 데이터 쿼리
- 제한 사항

개요

지리정보의 일반적인 사용 사례에는 데이터로부터의 근접성 분석이 포함됩니다. 예: “시애틀에서 50마일 이내에 있는 모든 공항 찾기” 또는 “특정 위치에서 가장 가까운 레스토랑 찾기”. Amazon DocumentDB는 GeoJSON 사양을 사용하여 지리공간 데이터를 나타냅니다. GeoJSON은 좌표 공간에서 도형의 JSON 형식 지정을 위한 오픈 소스 사양입니다. GeoJSON 좌표는 경도와 위도를 모두 캡처하여 지구와 같은 구의 위치를 나타냅니다.

지리공간 데이터 인덱싱 및 저장

Amazon DocumentDB는 '포인트' GeoJSON 유형을 사용하여 지리공간 데이터를 저장합니다. 각 GeoJSON 문서(또는 하위 문서)는 일반적으로 다음 두 필드로 구성됩니다.

- type - 표시되는 도형으로, Amazon DocumentDB에 “좌표” 필드를 해석하는 방법을 알려줍니다. 현재 Amazon DocumentDB는 포인트만 지원합니다.
- 좌표 - 배열에서 객체로 표시되는 위도 및 경도 쌍 - [경도, 위도]

또한 Amazon DocumentDB는 2dsphere 인덱스를 사용하여 지리공간 데이터를 인덱싱합니다. Amazon DocumentDB는 인덱싱 포인트를 지원합니다. Amazon DocumentDB는 2dsphere 인덱싱을 통한 근접 쿼리를 지원합니다.

음식 배달 서비스를 위한 애플리케이션을 구축하는 시나리오를 가정해 보겠습니다. 다양한 레스토랑의 위도 및 경도 쌍을 Amazon DocumentDB에 저장하려고 합니다. 이렇게 하려면 먼저 지리공간 필드에 위도와 경도 쌍을 포함하는 색인을 생성하는 것이 좋습니다.

```
use restaurantsdb
db.usarestaurants.createIndex({location:"2dsphere"})
```

이 명령의 출력은 다음과 같이 표시됩니다:

```
{
 "createdCollectionAutomatically" : true,
 "numIndexesBefore" : 1,
 "numIndexesAfter" : 2,
 "ok" : 1
}
```

색인을 생성한 후에는 Amazon DocumentDB 컬렉션에 데이터 삽입을 시작할 수 있습니다.

```
db.usarestaurants.insert({
   "state":"Washington",
   "city":"Seattle",
   "name":"Thai Palace",
   "rating": 4.8,
   "location":{
      "type":"Point",
      "coordinates":[
         -122.3264,
         47.6009
      ]
   }
});

db.usarestaurants.insert({
   "state":"Washington",
   "city":"Seattle",
   "name":"Noodle House",
   "rating": 4.8,
   "location":{
```

```
      "type":"Point",
      "coordinates":[
        -122.3517,
         47.6159
      ]
   }
});

db.usarestaurants.insert({
   "state":"Washington",
   "city":"Seattle",
   "name":"Curry House",
   "rating": 4.8,
   "location":{
      "type":"Point",
      "coordinates":[
         -121.4517,
         47.6229
      ]
   }
});
```

지리 공간 데이터 쿼리

Amazon DocumentDB는 지리공간 데이터의 근접성, 포함 및 교차 쿼리를 지원합니다. 근접 쿼리의 좋은 예는 특정 거리 미만 및 다른 포인트(도시)와의 거리 초과 거리에 있는 모든 포인트(모든 공항)를 찾는 것입니다. 포함 쿼리의 좋은 예는 지정된 지역/폴리곤(뉴욕주)에 위치한 모든 포인트(모든 공항)을 찾는 것입니다. 교차로 쿼리의 좋은 예는 한 포인트(도시)와 교차하는 한 폴리곤(주)을 찾는 것입니다. 다음 지리공간 연산자를 사용하여 데이터에서 통찰력을 얻을 수 있습니다.

- **$nearSphere** - $nearSphere GeoJSON 포인트에서 가장 가까운 포인트부터 가장 먼 포인트까지 점을 찾을 수 있도록 지원하는 찾기 연산자입니다.
- **$geoNear** - $geoNear GeoJSON 포인트로부터의 거리 계산을 미터 단위로 지원하는 집계 연산자입니다.
- **$minDistance** - $minDistance는 중심점으로부터 지정된 최소 거리에 있는 문서를 필터링하는 데 $nearSphere 또는 $geoNear와 함께 사용되는 찾기 연산자입니다.
- **$maxDistance** - $maxDistance는 중심점으로부터 지정된 최대 거리에 있는 문서를 필터링하는 데 $nearSphere 또는 $geoNear와 함께 사용되는 찾기 연산자입니다.
- **$geoWithin** - $geoWithin 찾기 연산자로, 다각형과 같이 지정된 도형 내에 완전히 존재하는 지리공간 데이터가 포함된 문서를 찾는 데 사용할 수 있습니다.

- **$geoIntersects** - $geoIntersects 지리공간 데이터가 지정된 GeoJSON 객체와 교차하는 문서를 찾을 수 있도록 지원하는 찾기 연산자입니다.

> **Note**
>
> $geoNear 및 $nearSphere는 근접 쿼리에서 사용하는 GeoJSON 필드에는 2dsphere 인덱스가 필요합니다.

예시 1

이 예제에서는 주소(포인트)로부터 가장 가까운 거리를 기준으로 정렬된 모든 레스토랑(포인트)을 찾는 방법을 알아봅니다.

이러한 쿼리를 수행하려면 $geoNear를 사용하여 포인트 집합과 다른 포인트 간의 거리를 계산할 수 있습니다. distanceMultiplier를 추가하여 킬로미터 단위로 거리를 측정할 수도 있습니다.

```
db.usarestaurants.aggregate([
   {
      "$geoNear":{
         "near":{
            "type":"Point",
            "coordinates":[
               -122.3516,
               47.6156
            ]
         },
         "spherical":true,
         "distanceField":"DistanceKilometers",
         "distanceMultiplier":0.001
      }
   }
])
```

위 명령은 지정된 포인트로부터의 거리(가장 가까운 곳부터 가장 먼 곳까지)를 기준으로 정렬된 식당을 반환합니다. 이 명령의 출력은 다음과 같이 표시됩니다:

```
{ "_id" : ObjectId("611f3da985009a81ad38e74b"), "state" : "Washington", "city" :
 "Seattle", "name" : "Noodle House", "rating" : 4.8, "location" : { "type" : "Point",
 "coordinates" : [ -122.3517, 47.6159 ] }, "DistanceKilometers" : 0.03422834547294996 }
```

```
{ "_id" : ObjectId("611f3da185009a81ad38e74a"), "state" : "Washington", "city" :
 "Seattle", "name" : "Thai Palace", "rating" : 4.8, "location" : { "type" : "Point",
 "coordinates" : [ -122.3264, 47.6009 ] }, "DistanceKilometers" : 2.5009390081704277 }
{ "_id" : ObjectId("611f3dae85009a81ad38e74c"), "state" : "Washington", "city" :
 "Seattle", "name" : "Curry House", "rating" : 4.8, "location" : { "type" : "Point",
 "coordinates" : [ -121.4517, 47.6229 ] }, "DistanceKilometers" : 67.52845344856914 }
```

쿼리의 결과 수를 제한하려면 `limit` 또는 num 옵션을 사용합니다.

`limit`:

```
db.usarestaurants.aggregate([
   {
      "$geoNear":{
         "near":{
            "type":"Point",
            "coordinates":[
               -122.3516,
               47.6156
            ]
         },
         "spherical":true,
         "distanceField":"DistanceKilometers",
         "distanceMultiplier":0.001,
         "limit": 10
      }
   }
])
```

num:

```
db.usarestaurants.aggregate([
   {
      "$geoNear":{
         "near":{
            "type":"Point",
            "coordinates":[
               -122.3516,
               47.6156
            ]
         },
         "spherical":true,
         "distanceField":"DistanceKilometers",
```

```
         "distanceMultiplier":0.001,
         "num": 10
      }
   }
])
```

> **Note**
>
> $geoNear 스테이지는 반환할 최대 문서 수를 지정하는 limit 및 num 옵션을 지원합니다. $geoNear는 limit 또는 num 옵션이 지정되지 않은 경우 기본적으로 최대 100개의 문서를 반환합니다. 이 값은 있는 경우 $limit 스테이지의 값으로 재정의되고 값이 100보다 작습니다.

예시 2

이 예제에서는 특정 주소(포인트)에서 2km 이내에 있는 모든 레스토랑(포인트)을 찾는 방법을 알아봅니다. 이러한 쿼리를 수행하려면 GeoJSON $maxDistance 포인트에서 최소 $minDistance 및 최대 $nearSphere 범위 내에서 사용할 수 있습니다.

```
db.usarestaurants.find({
   "location":{
      "$nearSphere":{
         "$geometry":{
            "type":"Point",
            "coordinates":[
               -122.3516,
               47.6156
            ]
         },
         "$minDistance":1,
         "$maxDistance":2000
      }
   }
},
{
   "name":1
})
```

위 명령은 지정된 포인트로부터 최대 2km 거리에 있는 레스토랑을 반환합니다. 이 명령의 출력은 다음과 같이 표시됩니다:

```
{ "_id" : ObjectId("611f3da985009a81ad38e74b"), "name" : "Noodle House" }
```

제한 사항

Amazon DocumentDB는 폴리곤, 라인스트링, 멀티포인트, 멀티폴리곤, 멀티라인스트링 및 지오메트리 컬렉션의 쿼리 또는 인덱싱을 지원하지 않습니다.

부분 인덱스

부분 인덱스는 지정된 필터 기준을 충족하는 컬렉션의 문서를 색인합니다. 부분 인덱스 기능은 Amazon DocumentDB 5.0 인스턴스 기반 클러스터에서 지원됩니다.

주제

- 부분 인덱스 생성
- 지원되는 연산자
- 부분 인덱스를 사용한 쿼리
- 부분 인덱스 기능
- 부분 인덱스 제한 사항

부분 인덱스 생성

부분 인덱스를 생성하려면 `partialFilterExpression` 옵션과 함께 `createIndex()` 메서드를 사용합니다. 예를 들어, 다음 작업은 `isDelivered` 필드가 `OrderID` true인 문서를 색인하는 명령 컬렉션에 고유한 복합 인덱스를 생성합니다.

```
db.orders.createIndex(
  {"category": 1, "CustomerId": 1, "OrderId": 1},
  {"unique": true, "partialFilterExpression":
    {"$and": [
      {"OrderId": {"$exists": true}},
      {"isDelivered": {"$eq": false}}
    ]}
  }
)
```

지원되는 연산자

- $eq
- $exists
- $and(최상위 수준에서만)
- $gt/$gte/$lt/$lte(인덱스 스캔은 쿼리에 예측된 필터가 부분 필터 식과 정확히 일치하는 경우에만 사용됨)(제한 사항 참조)

부분 인덱스를 사용한 쿼리

부분 인덱스를 사용하면 다음 쿼리 패턴을 사용할 수 있습니다.

- 쿼리 조건자는 부분 인덱스 필터 식과 정확히 일치합니다.

```
db.orders.find({"$and": [
    {"OrderId": {"$exists": true}},
    {"isDelivered": {"$eq": false}}
  ]}).explain()
```

- 쿼리 필터의 예상 결과는 부분 필터의 논리적 하위 집합입니다.

```
db.orders.find({"$and": [
    {"OrderId": {"$exists": true}},
    {"isDelivered": {"$eq": false}},
    {"OrderAmount": {"$eq": "5"}}
  ]}).explain()
```

- 쿼리의 하위 조건자는 다른 인덱스와 함께 사용할 수 있습니다.

```
db.orders.createIndex({"anotherIndex":1})
db.orders.find({ "$or": [
      {"$and": [
        {"OrderId": {"$exists": true}},
        {"isDelivered": {"$eq": false}}
      ]},
      {"anotherIndex": {"$eq": 5}}
    ]
  }).explain()
```

> **Note**
>
> 쿼리 실행 계획기는 효율적인 경우 인덱스 스캔 대신 컬렉션 스캔을 사용하도록 선택할 수 있습니다. 이는 일반적으로 컬렉션의 많은 부분을 반환하는 매우 작은 컬렉션 또는 쿼리에 표시됩니다.

부분 인덱스 기능

부분 인덱스 나열

`getIndex` 작업을 사용하여 partialFilterExpression으로 부분 인덱스를 나열합니다. 예를 들어, 발행된 `getIndex` 작업은 키, 이름 및 partialfilterExpressions 필드가 포함된 부분 인덱스를 나열합니다.

```
db.orders.getIndexes()
```

이 예제는 다음 출력을 반환합니다.

```
[
  {
    "v" : 4,
    "key" : {
      "_id" : 1
    },
    "name" : "_id_",
    "ns" : "ecommerceApp.orders"
  },
  {
    "v" : 4,
    "unique" : true,
    "key" : {
      "category" : 1,
      "" : 1,
      "CustomerId" : 1,
      "OrderId" : 1
    },
    "name" : "category_1_CustID_1_OrderId_1",
    "ns" : "ecommerceApp.orders",
    "partialFilterExpression" : {
      "$and" : [
        {"OrderId": {"$exists": true}},
```

```
        {"isDelivered": {"$eq": false}}
      ]
    }
  }
]
```

동일한 키에 대한 여러 부분 필터 식: 순서

동일한 필드 조합(키: 순서)에 대해 서로 다른 부분 인덱스를 생성할 수 있습니다. 이러한 인덱스의 이름은 달라야 합니다.

```
db.orders.createIndex(
  {"OrderId":1},
  {
    name:"firstPartialIndex",
    partialFilterExpression:{"OrderId":{"$exists": true}}
  }
)
```

```
db.orders.createIndex(
  {"OrderId":1},
  {
    name:"secondPartialIndex",
    partialFilterExpression:{"OrderId":{"$gt": 1000}}
  }
)
```

getIndexes 작업을 실행하여 컬렉션의 모든 인덱스를 나열합니다.

```
db.orders.getIndexes()
```

이러한 예제는 다음 출력을 반환합니다.

```
[
  {
    "v" : 4,
    "key" : {
      "_id" : 1
    },
    "name" : "_id_",
    "ns" : "ecommerceApp.orders"
```

```
  },
  {
    "v" : 4,
    "key" : {
      "OrderId" : 1
    },
    "name" : "firstPartialIndex",
    "ns" : "ecommerceApp.orders",
    "partialFilterExpression" : {"OrderId":{"$exists": true}}
  },
  {
    "v" : 4,
    "key" : {
      "OrderId" : 1
    },
    "name" : "secondPartialIndex",
    "ns" : "ecommerceApp.orders",
    "partialFilterExpression" : {"OrderId":{"$gt": 1000}}
  }
]
```

> **Important**
>
> 인덱스 이름은 달라야 하며 이름으로만 삭제해야 합니다.

부분 및 TTL 속성이 있는 인덱스

인덱스 생성 중에 `partialFilterExpression` 및 `expireAfterSeconds` 옵션을 모두 지정하여 부분 및 TTL 속성을 가진 인덱스를 생성할 수도 있습니다. 이렇게 하면 컬렉션에서 이제 제거되는 문서를 더 잘 제어할 수 있습니다.

예를 들어 특정 기간 이후에 삭제할 문서를 식별하는 TTL 인덱스가 있을 수 있습니다. 이제 부분 인덱스 옵션을 사용하여 문서를 삭제하는 시기에 대한 추가 조건을 지정할 수 있습니다.

```
db.orders.createIndex(
    { "OrderTimestamp": 1 },
    {
        expireAfterSeconds: 3600 ,
        partialFilterExpression: { "isDelivered": { $eq: true } }
    }
)
```

이 예제는 다음 출력을 반환합니다.

```
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1,
        "operationTime" : Timestamp(1234567890, 1)
}
```

getIndexes 작업을 실행하여 컬렉션에 존재하는 인덱스를 나열합니다.

```
db.orders.getIndexes()
[
    {
        "v" : 4,
        "key" : {
            "_id" : 1
        },
        "name" : "_id_",
        "ns" : "test.orders"
    }
```

이 예제는 다음 출력을 반환합니다.

```
[
    {
        "v": 4,
        "key": {
            "_id": 1
        },
        "name": "_id_",
        "ns": "ecommerceApp.orders"
    },
    {
        "v": 4,
        "key": {
            "OrderTimestamp": 1
        },
        "name": "OrderTimestamp_1",
        "ns": "ecommerceApp.orders",
        "partialFilterExpression": {
            "isDelivered": {
```

```
                "$eq": true
            }
        },
        "expireAfterSeconds": 3600
    }
]
```

부분 인덱스 제한 사항

다음 제한 사항이 부분 인덱스 기능에 적용됩니다.

- Amazon DocumentDB의 불평등 쿼리는 쿼리 필터 예측이 `partialFilterExpression`과 정확히 일치하고 동일한 데이터 유형인 경우에만 부분 인덱스를 사용합니다.

  > **Note**
  >
  > 위 사례에 대해 IXSCAN을 강제로 적용하는 데도 `$hint`를 사용할 수 없습니다.

  다음 예제에서 `partialFilterExpression`은 `field1`에만 적용되고 `field2`에는 적용되지 않습니다.

  ```
  db.orders.createIndex(
    {"OrderAmount": 1},
    {"partialFilterExpression": { OrderAmount : {"$gt" : 5}}}
  )

  db.orders.find({OrderAmount : {"$gt" : 5}}) // Will use partial index
  db.orders.find({OrderAmount : {"$gt" : 6}}) // Will not use partial index
  db.orders.find({OrderAmount : {"$gt" : Decimal128(5.00)}}) // Will not use partial
   index
  ```

- 배열 연산자가 `partialFilterExpression`은 지원되지 않습니다. 다음 작업은 오류를 생성합니다.

  ```
  db.orders.createIndex(
    {"CustomerId":1},
    {'partialFilterExpression': {'OrderId': {'$eq': [1000, 1001, 1002]}}}
  )
  ```

- partialFilterExpression 필드에서는 다음 연산자가 지원되지 않습니다.

- `$all` (배열 연산자)
- `$mod` (배열 연산자)
- `$or`
- `$xor`
- `$not`
- `$nor`

- 필터 식과 필터의 데이터 유형은 동일해야 합니다.

Amazon DocumentDB를 사용하여 텍스트 검색 수행

Amazon DocumentDB의 기본 전체 텍스트 검색 기능을 사용하면 특수 목적 텍스트 인덱스를 사용하여 대규모 텍스트 데이터 세트에서 텍스트 검색을 수행할 수 있습니다. 이 섹션에서는 텍스트 인덱스 기능의 기능을 설명하고 Amazon DocumentDB에서 텍스트 인덱스를 생성하고 사용하는 방법에 대한 단계를 제공합니다. 텍스트 검색 제한 사항도 나열됩니다.

주제

- 지원되는 기능
- Amazon DocumentDB 텍스트 인덱스 사용
- MongoDB와의 차이점
- 모범 사례 및 지침
- 제한 사항

지원되는 기능

Amazon DocumentDB 텍스트 검색은 다음 MongoDB API 호환 기능을 지원합니다.

- 단일 필드에 텍스트 인덱스를 생성합니다.
- 둘 이상의 텍스트 필드가 포함된 복합 텍스트 인덱스를 생성합니다.
- 단일 단어 또는 다중 단어 검색을 수행합니다.
- 가중치를 사용하여 검색 결과를 제어합니다.
- 점수별로 검색 결과를 정렬합니다.
- 집계 파이프라인에서 텍스트 인덱스를 사용합니다.
- 정확한 구문을 검색합니다.

Amazon DocumentDB 텍스트 인덱스 사용

문자열 데이터가 포함된 필드에 텍스트 인덱스를 생성하려면 다음과 같이 문자열 '텍스트'를 지정합니다.

단일 필드 인덱스:

```
db.test.createIndex({"comments": "text"})
```

이 인덱스는 지정된 컬렉션의 '의견' 문자열 필드에서 텍스트 검색 쿼리를 지원합니다.

둘 이상의 문자열 필드에 복합 텍스트 인덱스 생성:

```
db.test.createIndex({"comments": "text", "title":"text"})
```

이 인덱스는 지정된 컬렉션의 '의견' 및 '제목' 문자열 필드에서 텍스트 검색 쿼리를 지원합니다. 복합 텍스트 인덱스를 생성할 때 최대 30개의 필드를 지정할 수 있습니다. 생성된 텍스트 검색 쿼리는 모든 인덱싱된 필드를 쿼리합니다.

> **Note**
>
> 각 컬렉션에는 하나의 텍스트 인덱스만 허용됩니다.

Amazon DocumentDB 컬렉션에 텍스트 인덱스 나열

아래 예제와 같이 컬렉션에서 `getIndexes()`를 사용하여 텍스트 인덱스를 포함한 인덱스를 식별하고 설명할 수 있습니다.

```
rs0:PRIMARY> db.test.getIndexes()
[
   {
      "v" : 4,
      "key" : {
         "_id" : 1
      },
      "name" : "_id_",
      "ns" : "test.test"
   },
   {
```

```
      "v" : 1,
      "key" : {
         "_fts" : "text",
         "_ftsx" : 1
      },
      "name" : "contents_text",
      "ns" : "test.test",
      "default_language" : "english",
      "weights" : {
         "comments" : 1
      },
      "textIndexVersion" : 1
   }
]
```

색인을 생성한 후에는 Amazon DocumentDB 컬렉션에 데이터 삽입을 시작합니다.

```
db.test.insertMany([{"_id": 1, "star_rating": 4, "comments": "apple is red"},
                    {"_id": 2, "star_rating": 5, "comments": "pie is delicious"},
                    {"_id": 3, "star_rating": 3, "comments": "apples, oranges - healthy
 fruit"},
                    {"_id": 4, "star_rating": 2, "comments": "bake the apple pie in the
 oven"},
                    {"_id": 5, "star_rating": 5, "comments": "interesting couch"},
                    {"_id": 6, "star_rating": 5, "comments": "interested in couch for
 sale, year 2022"}])
```

텍스트 검색 쿼리 실행

단일 단어 텍스트 검색 쿼리 실행

`$text` 및 `$search` 연산자를 사용하여 텍스트 검색을 수행해야 합니다. 다음 예제에서는 텍스트 인덱싱된 필드에 문자열 “apple” 또는 “apples”와 같이 다른 형식의 “apple”이 포함된 모든 문서를 반환합니다.

```
db.test.find({$text: {$search: "apple"}})
```

출력:

이 명령의 출력은 다음과 같이 표시됩니다.

```
{ "_id" : 1, "star_rating" : 4, "comments" : "apple is red" }
```

```
{ "_id" : 3, "star_rating" : 3, "comments" : "apples, oranges - healthy fruit" }
{ "_id" : 4, "star_rating" : 2, "comments" : "bake the apple pie in the oven" }
```

다중 단어 텍스트 검색 실행

Amazon DocumentDB 데이터에 대해 다중 단어 텍스트 검색을 수행할 수도 있습니다. 아래 명령은 “apple” 또는 “pie”가 포함된 텍스트 인덱싱 필드가 있는 문서를 반환합니다.

```
db.test.find({$text: {$search: "apple pie"}})
```

출력:

이 명령의 출력은 다음과 같이 표시됩니다.

```
{ "_id" : 1, "star_rating" : 4, "comments" : "apple is red" }
{ "_id" : 2, "star_rating" : 5, "comments" : "pie is delicious" }
{ "_id" : 3, "star_rating" : 3, "comments" : "apples, oranges - healthy fruit" }
{ "_id" : 4, "star_rating" : 2, "comments" : "bake the apple pie in the oven" }
```

다중 단어 구문 텍스트 검색 실행

다중 단어 구문 검색의 경우 다음 예제를 사용합니다.

```
db.test.find({$text: {$search: "\"apple pie\""}})
```

출력:

위 명령은 정확한 문구 “apple pie”가 포함된 텍스트 인덱싱 필드가 있는 문서를 반환합니다. 이 명령의 출력은 다음과 같이 표시됩니다.

```
{ "_id" : 4, "star_rating" : 2, "comments" : "bake the apple pie in the oven" }
```

필터로 텍스트 검색 실행

텍스트 검색을 다른 쿼리 연산자와 결합하여 추가 기준에 따라 결과를 필터링할 수도 있습니다.

```
db.test.find({$and: [{star_rating: 5}, {$text: {$search: "interest"}}]})
```

출력:

위 명령은 모든 형태의 'interest'와 5와 동일한 'star_rating'이 포함된 텍스트 인덱싱 필드가 있는 문서를 반환합니다. 이 명령의 출력은 다음과 같이 표시됩니다.

```
{ "_id" : 5, "star_rating" : 5, "comments" : "interesting couch" }
{ "_id" : 6, "star_rating" : 5, "comments" : "interested in couch for sale, year
 2022" }
```

텍스트 검색에서 반환되는 문서 수 한도

`limit`를 사용하여 반환되는 문서 수를 제한하도록 선택할 수 있습니다.

```
db.test.find({$and: [{star_rating: 5}, {$text: {$search: "couch"}}]}).limit(1)
```

출력:

위 명령은 필터를 충족하는 결과 하나를 반환합니다.

```
{ "_id" : 5, "star_rating" : 5, "comments" : "interesting couch" }
```

텍스트 점수로 결과 정렬

다음 예제에서는 텍스트 검색 결과를 텍스트 점수별로 정렬합니다.

```
db.test.find({$text: {$search: "apple"}}, {score: {$meta: "textScore"}}).sort({score:
 {$meta: "textScore"}})
```

출력:

위 명령은 "apple" 또는 "apple"이 포함된 텍스트 인덱싱 필드가 포함된 문서를 "apples"와 같은 다른 형식으로 반환하고 문서가 검색어와 얼마나 관련이 있는지에 따라 결과를 정렬합니다. 이 명령의 출력은 다음과 같이 표시됩니다.

```
{ "_id" : 1, "star_rating" : 4, "comments" : "apple is red", "score" :
 0.6079270860936958 }
{ "_id" : 3, "star_rating" : 3, "comments" : "apples, oranges - healthy fruit",
 "score" : 0.6079270860936958 }
{ "_id" : 4, "star_rating" : 2, "comments" : "bake the apple pie in the oven",
 "score" : 0.6079270860936958 }
```

`$text` 및 `$search`는 `aggregate`, `count`, `findAndModify`, `update` 및 `delete` 명령에도 지원됩니다.

집계 연산자

$match를 사용한 집계 파이프라인

```
db.test.aggregate(
   [{ $match: { $text: { $search: "apple pie" } } }]
)
```

출력:

위의 명령은 다음 결과를 반환합니다.

```
{ "_id" : 1, "star_rating" : 4, "comments" : "apple is red" }
{ "_id" : 3, "star_rating" : 3, "comments" : "apple - a healthy fruit" }
{ "_id" : 4, "star_rating" : 2, "comments" : "bake the apple pie in the oven" }
{ "_id" : 2, "star_rating" : 5, "comments" : "pie is delicious" }
```

다른 집계 연산자의 조합

```
db.test.aggregate(
   [
      { $match: { $text: { $search: "apple pie" } } },
      { $sort: { score: { $meta: "textScore" } } },
      { $project: { score: { $meta: "textScore" } } }
   ]
)
```

출력:

위의 명령은 다음 결과를 반환합니다.

```
{ "_id" : 4, "score" : 0.6079270860936958 }
{ "_id" : 1, "score" : 0.3039635430468479 }
{ "_id" : 2, "score" : 0.3039635430468479 }
{ "_id" : 3, "score" : 0.3039635430468479 }
```

텍스트 인덱스를 생성할 때 여러 필드 지정

복합 텍스트 인덱스에서 최대 3개의 필드에 가중치를 할당할 수 있습니다. 텍스트 인덱스의 필드에 할당된 기본 가중치는 일(1)입니다. 가중치는 선택적 파라미터이며 1~100,000 범위여야 합니다.

```
db.test.createIndex(
   {
     "firstname": "text",
     "lastname": "text",
     ...
   },
   {
     weights: {
       "firstname": 5,
       "lastname":10,
       ...
     },
     name: "name_text_index"
   }
 )
```

MongoDB와의 차이점

Amazon DocumentDB의 텍스트 인덱스 기능은 용어-주파수 알고리즘과 함께 반전된 인덱스를 사용합니다. 텍스트 인덱스는 기본적으로 희소입니다. 구문 분석 로직, 토큰화 구분 기호 등의 차이로 인해 MongoDB와 동일한 결과 세트가 동일한 데이터 세트 또는 쿼리 모양에 대해 반환되지 않을 수 있습니다.

Amazon DocumentDB 텍스트 인덱스와 MongoDB에는 다음과 같은 추가 차이점이 있습니다.

- 텍스트가 아닌 인덱스를 사용하는 복합 인덱스는 지원되지 않습니다.
- Amazon DocumentDB 텍스트 인덱스는 대/소문자를 구분하지 않습니다.
- 텍스트 인덱스에서는 영어만 지원됩니다.
- 배열(또는 다중 키) 필드의 텍스트 인덱싱은 지원되지 않습니다. 예를 들어 문서 {“a”:[“apple”, “pie”]}로 “a“에 텍스트 인덱스를 생성하면 실패합니다.
- 와일드카드 텍스트 인덱싱은 지원되지 않습니다.
- 고유한 텍스트 인덱스는 지원되지 않습니다.
- 용어 제외는 지원되지 않습니다.

모범 사례 및 지침

- 텍스트 점수별 정렬과 관련된 텍스트 검색 쿼리의 성능을 최적화하려면 데이터를 로드하기 전에 텍스트 인덱스를 생성하는 것이 좋습니다.

- 텍스트 인덱스에는 인덱싱된 데이터의 최적화된 내부 복사본을 위한 추가 스토리지가 필요합니다. 이로 인해 추가 비용이 발생합니다.

제한 사항

Amazon DocumentDB 에서 텍스트 검색에는 다음과 같은 제한이 있습니다.

- 텍스트 검색은 Amazon DocumentDB 5.0 인스턴스 기반 클러스터에서만 지원됩니다.
- 텍스트 인덱스는 어휘와 해당 위치 정보를 저장합니다. 단일 문서 내에서 모든 어휘와 위치 정보의 결합 크기는 1MB로 제한됩니다.

Amazon DocumentDB 문제 해결

다음 섹션에는 Amazon DocumentDB(MongoDB 호환) 사용 시 발생할 수 있는 문제를 해결하는 방법에 대한 정보가 나와 있습니다.

주제

- 연결 문제 해결
- 인덱스 생성 문제 해결
- 성능 및 리소스 사용률 문제 해결

연결 문제 해결

연결에 문제가 있으신가요? 다음에서는 몇 가지 일반적인 시나리오와 이에 대한 해결 방법에 대해 설명합니다.

주제

- Amazon DocumentDB 엔드포인트에 연결할 수 없습니다.
- Amazon DocumentDB 인스턴스 연결 테스트
- 유효하지 않은 엔드포인트에 연결
- 연결 수에 영향을 미치는 드라이버 구성

Amazon DocumentDB 엔드포인트에 연결할 수 없습니다.

다음은 Amazon DocumentDB에 연결하려고 할 때 표시될 수 있는 가장 일반적인 오류 메시지 중 하나입니다.

```
connecting to: mongodb://docdb-2018-11-08-21-47-27.cluster-ccuszbx3pn5e.us-east-
1.docdb.amazonaws.com:27017/
2018-11-14T14:33:46.451-0800 W NETWORK [thread1] Failed to connect to
172.31.91.193:27017 after 5000ms milliseconds, giving up.
2018-11-14T14:33:46.452-0800 E QUERY [thread1] Error: couldn't connect to server
docdb-2018-11-08-21-47-27.cluster-ccuszbx3pn5e.us-east-1.docdb.amazonaws.com:27017,
connection attempt failed :
connect@src/mongo/shell/mongo.js:237:13
@(connect):1:6
```

```
exception: connect failed
```

이 오류 메시지의 일반적인 의미는 클라이언트(이 예제에서는 mongo 쉘)가 Amazon DocumentDB 엔드포인트에 액세스할 수 없다는 것입니다. 다음과 같이 여러 가지 원인이 있을 수 있습니다.

주제

- 퍼블릭 엔드포인트로부터 연결
- 리전 간 연결
- 다른 Amazon VPC에서 연결
- 보안 그룹이 인바운드 연결을 차단
- Java Mongo 드라이버 읽기 기본 설정 문제

퍼블릭 엔드포인트로부터 연결

노트북 또는 로컬 개발 머신에서 직접 Amazon DocumentDB 클러스터에 연결하려고 시도하는 중입니다.

노트북 또는 로컬 개발 머신과 같은 퍼블릭 엔드포인트에서 직접 Amazon DocumentDB 클러스터에 연결하려고 시도하는 것은 실패하게 됩니다. Amazon DocumentDB는 Virtual Private Cloud(VPC) 전용이며 현재 퍼블릭 엔드포인트를 지원하지 않습니다. 따라서 VPC 외부의 노트북 또는 개발 환경에서 Amazon DocumentDB 클러스터에 직접 연결할 수 없습니다.

Amazon VPC 외부에서 Amazon DocumentDB 클러스터에 연결하려면 SSH 터널을 사용할 수 있습니다. 자세한 내용은 Amazon VPC 외부에서 Amazon DocumentDB 클러스터에 연결 섹션을 참조하세요. 또한, 개발 환경이 다른 Amazon VPC에 있을 경우에는 VPC 피어링을 사용하여 동일한 리전 또는 다른 리전의 다른 Amazon VPC에서 Amazon DocumentDB 클러스터에 연결할 수 있습니다.

리전 간 연결

또 다른 리전에 있는 Amazon DocumentDB 클러스터에 연결하려고 합니다.

클러스터 지역이 아닌 지역의 Amazon EC2 인스턴스에서 Amazon DocumentDB 클러스터에 연결하려고 시도할 경우(예: 미국 서부(오레곤) 지역(us-west-2)에서 미국 동부(버지니아 북부) 지역(us-east-1)의 클러스터에 연결하려고 하면 연결이 실패합니다.

Amazon DocumentDB 클러스터의 리전을 확인하려면 다음 명령을 실행합니다. 리전이 엔드포인트에 있습니다.

```
aws docdb describe-db-clusters \
   --db-cluster-identifier sample-cluster \
   --query 'DBClusters[*].Endpoint'
```

이 작업의 출력은 다음과 같습니다.

```
[
    "sample-cluster.node.us-east-1.docdb.amazonaws.com"
]
```

EC2 인스턴스의 리전을 확인하려면 다음 명령을 실행합니다.

```
 aws ec2 describe-instances \
     --query 'Reservations[*].Instances[*].Placement.AvailabilityZone'
```

이 작업의 출력은 다음과 같습니다.

```
[
    [
        "us-east-1a"
    ]
]
```

다른 Amazon VPC에서 연결

클러스터가 배포된 Amazon VPC가 아닌 다른 Amazon DocumentDB 클러스터에 연결하려고 합니다.

Amazon DocumentDB 클러스터와 Amazon EC2 인스턴스가 동일한에 AWS 리전있지만 동일한 Amazon VPC에는 없는 경우 두 Amazon VPC 간에 VPC 피어링이 활성화되지 않는 한 Amazon DocumentDB 클러스터에 직접 연결할 수 없습니다 VPCs.

Amazon DocumentDB 인스턴스의 Amazon VPC를 확인하려면 다음 명령을 실행합니다.

```
aws docdb describe-db-instances \
   --db-instance-identifier sample-instance \
   --query 'DBInstances[*].DBSubnetGroup.VpcId'
```

Amazon EC2 인스턴스의 Amazon VPC를 확인하려면 다음 명령을 실행합니다.

```
aws ec2 describe-instances \
```

```
   --query 'Reservations[*].Instances[*].VpcId'
```

보안 그룹이 인바운드 연결을 차단

Amazon DocumentDB 클러스터에 연결하려고 하는데 클러스터의 보안 그룹이 클러스터 포트(기본 포트: 27017)의 인바운드 연결을 허용하지 않습니다.

Amazon DocumentDB 클러스터와 Amazon EC2 인스턴스가 모두 동일한 리전 및 Amazon VPC에 있고 동일한 Amazon VPC 보안 그룹을 사용한다고 가정합니다. Amazon DocumentDB 클러스터에 연결할 수 없는 경우 가능성이 높은 원인은 클러스터에 대한 보안 그룹(예: 방화벽)이 Amazon DocumentDB 클러스터에 대해 선택한 포트(기본 포트: 27017)의 인바운드 연결을 허용하지 않기 때문입니다.

Amazon DocumentDB 클러스터의 포트를 확인하려면 다음 명령을 실행합니다.

```
aws docdb describe-db-clusters \
   --db-cluster-identifier sample-cluster \
   --query 'DBClusters[*].[DBClusterIdentifier,Port]'
```

클러스터에 대한 Amazon DocumentDB 보안 그룹을 가져오려면 다음 명령을 실행합니다.

```
aws docdb describe-db-clusters \
   --db-cluster-identifier sample-cluster \
   --query 'DBClusters[*].[VpcSecurityGroups[*],VpcSecurityGroupId]'
```

보안 그룹에 대한 인바운드 규칙을 확인하려면 Amazon EC2 설명서에서 다음 주제를 참조하십시오.

- Linux 인스턴스의 인바운드 트래픽 권한 부여
- Windows 인스턴스의 인바운드 트래픽 권한 부여

Java Mongo 드라이버 읽기 기본 설정 문제

클라이언트의 읽기 기본 설정이 적용되지 않으며 일부 클라이언트는 재부팅하지 않는 한 장애 조치 후 Amazon DocumentDB에 쓸 수 없습니다.

Java Mongo Driver 3.7.x에서 처음 발견된 이 문제는 클라이언트가 특히 `MongoClientSettings` 메서드를 사용하여 Amazon DocumentDB에 연결할 때, 특히 `applyToClusterSettings` 메서드를 체인으로 연결할 때 발생합니다. MongoClient 클러스터 설정은 `hosts()`,

requiredReplicaSetName(), mode()와(과) 같은 몇 가지 다른 방법을 사용하여 정의할 수 있습니다.

클라이언트가 hosts() 메서드에서 호스트를 하나만 지정하면 모드가 ClusterConnectionMode.SINGLE 대신 ClusterConnectionMode.MULTIPLE으로 설정됩니다. 이렇게 하면 클라이언트는 읽기 기본 설정을 무시하고 hosts()에서 구성된 서버에만 연결합니다. 따라서 아래와 같이 클라이언트 설정을 초기화하더라도 모든 읽기는 여전히 보조 서버 대신 기본 서버로 이동합니다.

```
final ServerAddress serverAddress0 = new ServerAddress("cluster-endpoint", 27317));
    final MongoCredential credential = MongoCredential.createCredential("xxx",
            "admin", "xxxx".toCharArray());
    final MongoClientSettings settings = MongoClientSettings.builder()
            .credential(credential)
            .readPreference(ReadPreference.secondaryPreferred())
            .retryWrites(false)
            .applyToSslSettings(builder -> builder
                    .enabled(false))
            .applyToClusterSettings(builder -> builder.hosts(
                            Arrays.asList(serverAddress0
                            ))
                    .requiredReplicaSetName("rs0"))
            .build();
    MongoClient mongoClient = MongoClients.create(settings);
```

장애 조치 사례

위의 클라이언트 연결 설정을 사용하면 클러스터 작성기 엔드포인트에 대한 장애 조치 및 지연된 DNS 레코드 업데이트가 발생하는 경우 클라이언트는 여전히 이전 작성기(이제는 장애 조치 후 리더)에 쓰기를 시도합니다. 이로 인해 서버 측 오류(기본 아님)가 발생하는데, 이 오류는 Java 드라이버에서 적절하게 처리되지 않습니다(이 내용은 아직 조사 중임). 따라서 예를 들어 애플리케이션 서버가 재부팅될 때까지 클라이언트는 잘못된 상태로 남아 있을 수 있습니다.

이에 대한 두 가지 해결 방법이 있습니다.

- 연결 문자열을 통해 Amazon DocumentDB에 연결하는 클라이언트는 읽기 기본 설정을 지정할 때 ClusterConnectionMode가 MULTIPLE로 설정되므로 이 문제가 발생하지 않습니다.

  ```
  MongoClientURI mongoClientURI = new MongoClientURI("mongodb://usr:pass:cluster-
  endpoint:27317/test?ssl=false&replicaSet=rs0&readpreference=secondaryPreferred");
  MongoClient mongoClient = MongoClients.create(mongoClientURI.getURI());
  ```

applyConnectionString 메서드와 함께 MongoClientSettings 빌더를 사용할 수도 있습니다.

```
final MongoClientSettings settings = MongoClientSettings.builder()
        .credential(credential)
        .applyConnectionString(new ConnectionString("usr:pass:cluster-endpoint:27317/
test?ssl=false&replicaSet=rs0&readpreference=secondaryPreferred"))
        .retryWrites(false)
        .applyToSslSettings(builder # builder
                .enabled(false))
        .build();
MongoClient mongoClient = MongoClients.create(settings);
```

- 명시적으로 ClusterConnectionMode을 MULTIPLE으로 설정합니다. 이는 applyToClusterSettings 및 hosts().size() == 1를 사용할 때만 필요합니다.

```
final ServerAddress serverAddress0 = new ServerAddress("cluster-endpoint", 27317));
final MongoCredential credential = MongoCredential.createCredential("xxx","admin",
 "xxxx".toCharArray());
final MongoClientSettings settings = MongoClientSettings.builder()
    .credential(credential)
    .readPreference(ReadPreference.secondaryPreferred())
    .retryWrites(false)
    .applyToSslSettings(builder # builder
    .enabled(false))
    .applyToClusterSettings(builder # builder
                .hosts(Arrays.asList(serverAddress0))
                .requiredReplicaSetName("rs0"))
                .mode(ClusterConnectionMode.MULTIPLE))
    .build();
MongoClient mongoClient = MongoClients.create(settings);
```

Amazon DocumentDB 인스턴스 연결 테스트

공통 Linux 또는 Windows 도구를 사용하여 클러스터에 대한 연결을 테스트할 수 있습니다.

Linux 또는 Unix 터미널에서 다음을 입력하여 연결을 테스트할 수 있습니다(cluster-endpoint를 해당 엔드포인트로 바꾸고 port를 인스턴스의 포트로 바꿈).

```
nc -zv cluster-endpoint port
```

다음은 샘플 작업 및 반환 값의 예입니다.

```
nc -zv docdbTest.d4c7nm7stsfc0.us-west-2.docdb.amazonaws.com 27017

Connection to docdbTest.d4c7nm7stsfc0.us-west-2.docdb.amazonaws.com 27017 port [tcp/*]
 succeeded!
```

유효하지 않은 엔드포인트에 연결

Amazon DocumentDB 클러스터에 연결하고 유효하지 않은 클러스터 엔드포인트를 사용할 때 다음과 유사한 오류가 표시됩니다.

```
mongo --ssl \
   --host sample-cluster.node.us-east-1.docdb.amazonaws.com:27017 \
   --sslCAFile global-bundle.pem \
   --username <user-name> \
   --password <password>
```

출력은 다음과 같습니다.

```
MongoDB shell version v3.6
connecting to: mongodb://sample-cluster.node.us-east-1.docdb.amazonaws.com:27017/
2018-11-14T17:21:18.516-0800 I NETWORK [thread1] getaddrinfo("sample-cluster.node.us-
east-1.docdb.amazonaws.com") failed:
nodename nor servname provided, or not known 2018-11-14T17:21:18.537-0800 E QUERY
 [thread1] Error: couldn't initialize
connection to host sample-cluster.node.us-east-1.docdb.amazonaws.com, address is
 invalid :
connect@src/mongo/shell/mongo.js:237:13@(connect):1:6
exception: connect failed
```

클러스터에 대해 유효한 엔드포인트를 가져오려면 다음 명령을 실행합니다.

```
aws docdb describe-db-clusters \
   --db-cluster-identifier sample-cluster \
   --query 'DBClusters[*].[Endpoint,Port]'
```

인스턴스에 대해 유효한 엔드포인트를 가져오려면 다음 명령을 실행합니다.

```
aws docdb describe-db-instances \
   --db-instance-identifier sample-instance \
```

```
    --query 'DBInstances[*].[Endpoint.Address,Endpoint.Port]'
```

자세한 내용은 Amazon DocumentDB 엔드포인트에 대한 이해 단원을 참조하십시오.

연결 수에 영향을 미치는 드라이버 구성

클라이언트 드라이버를 사용하여 Amazon DocumentDB 클러스터에 연결할 때는 `maxPoolSize` 구성 파라미터를 고려하는 것이 중요합니다. `maxPoolSize` 설정은 클라이언트 드라이버가 연결 풀에서 유지할 최대 연결 수를 결정합니다.

인덱스 생성 문제 해결

다음 주제에서는 인덱스 또는 배경 인덱스 빌드에 실패한 경우 취해야 할 조치를 설명합니다.

주제

- 인덱스 빌드 실패
- 백그라운드 인덱스 빌드 지연 문제 및 실패

인덱스 빌드 실패

Amazon DocumentDB는 인덱스 생성 프로세스의 일부로써 인스턴스의 로컬 스토리지를 활용합니다. FreeLocalStorage CloudWatch 지표(`CloudWatch -> Metrics -> DocDB -> Instance Metrics`)를 사용하여 이 디스크 사용량을 모니터링할 수 있습니다. 인덱스 빌드에 로컬 디스크 전체가 사용되고 실패할 경우 오류가 표시됩니다. 데이터를 Amazon DocumentDB로 마이그레이션할 경우 먼저 인덱스를 생성한 다음 데이터를 삽입하는 것이 좋습니다. 마이그레이션 전략 및 인덱스 생성에 대한 자세한 내용은 Amazon DocumentDB 설명서의 Amazon DocumentDB로 마이그레이션 섹션과 오프라인 방법을 사용한 MongoDB에서 Amazon DocumentDB로의 마이그레이션 블로그를 참조하십시오.

기존 클러스터에서 인덱스를 생성할 때 인덱스 빌드에 예상보다 시간이 오래 걸리거나 실패하는 경우, 인스턴스 크기를 늘려(규모 조정) 인덱스를 생성한 다음 다시 축소하는 것이 좋습니다. Amazon DocumentDB를 사용하면 AWS Management Console 또는를 사용하여 인스턴스 크기를 몇 분 만에 빠르게 조정할 수 있습니다 AWS CLI. 자세한 내용은 인스턴스 클래스 관리 단원을 참조하십시오. 초당 요금이 청구되는 경우, 초 단위로 계산하여 사용한 리소스에 대해서만 지불하면 됩니다.

백그라운드 인덱스 빌드 지연 문제 및 실패

Amazon DocumentDB의 백그라운드 인덱스 빌드는 인덱스 빌드가 시작되기 전에 시작된 기본 인스턴스의 모든 쿼리가 실행될 때까지 시작되지 않습니다. 쿼리가 오래 실행되는 경우 쿼리가 완료될 때까지

백그라운드 인덱스 빌드가 차단되므로 완료하는 데 예상보다 시간이 오래 걸릴 수 있습니다. 컬렉션이 비어 있는 경우에도 마찬가지입니다.

포그라운드 인덱스 빌드는 동일한 차단 동작을 나타내지 않습니다. 대신 포그라운드 인덱스 빌드는 인덱스 빌드가 완료될 때까지 컬렉션을 독점적으로 잠급니다. 따라서 빈 컬렉션에 인덱스를 만들고 장기 실행 쿼리가 차단되지 않도록 하려면 포그라운드 인덱스 빌드를 사용하는 것이 좋습니다.

> **Note**
>
> Amazon DocumentDB는 특정 시간에 모음에서 발생하는 배경 인덱스 빌드를 하나만 허용합니다. 배경 인덱스 빌드 중에 동일한 컬렉션에 `createIndex()` 또는 `dropIndex()`와 같은 DDL(Data Definition Language) 연산이 발생하면 배경 인덱스 빌드가 실패합니다.

성능 및 리소스 사용률 문제 해결

이 섹션에서는 Amazon DocumentDB 배포 시 일반적인 진단 문제에 대한 질문과 솔루션을 제공합니다. 제공된 예제는 mongo 쉘을 사용하며 개별 인스턴스에 적용됩니다. 인스턴스 엔드포인트를 찾으려면 Amazon DocumentDB 엔드포인트에 대한 이해을 참조하세요.

주제

- Mongo API를 통해 내 컬렉션에 대해 수행된 삽입, 업데이트 및 삭제 작업 수를 확인하려면 어떻게 해야 하나요?
- 캐시 성능을 분석하려면 어떻게 해야 하나요?
- 장시간 실행 중이거나 차단된 쿼리를 찾아서 종료하려면 어떻게 해야 하나요?
- 쿼리 계획을 보고 쿼리를 최적화하려면 어떻게 해야 하나요?
- 엘라스틱 클러스터에서 쿼리 계획을 확인하려면 어떻게 해야 하나요?
- 인스턴스에서 실행 중인 작업을 모두 나열하려면 어떻게 해야 하나요?
- 쿼리 진행 상황을 어떻게 알 수 있나요?
- 시스템이 갑자기 느리게 실행되는 이유를 어떻게 알 수 있나요?
- 하나 이상의 클러스터 인스턴스에서 높은 CPU 사용률의 원인을 어떻게 확인하나요?
- 인스턴스에서 열려 있는 커서를 어떻게 확인하나요?
- 현재 Amazon DocumentDB 엔진 버전은 어떻게 확인하나요?
- 인덱스 사용량을 분석하고 사용하지 않는 인덱스를 식별하려면 어떻게 해야 하나요?
- 누락된 인덱스는 어떻게 식별하나요?

- 유용한 쿼리 요약

Mongo API를 통해 내 컬렉션에 대해 수행된 삽입, 업데이트 및 삭제 작업 수를 확인하려면 어떻게 해야 하나요?

특정 컬렉션에서 수행된 삽입, 업데이트 및 삭제 작업 수를 보려면 해당 컬렉션에서 다음 명령을 실행합니다.

```
db.collection.stats()
```

명령에 대한 출력은 opCounters 필드에서 다음과 같습니다.

- numDocsIns - 이 컬렉션에 삽입된 문서 수입니다. 여기에는 insert 및 insertMany 명령을 사용하여 삽입한 문서와 업서트로 삽입한 문서가 포함됩니다.
- numDocsUpd - 이 컬렉션에서 업데이트된 문서 수입니다. 여기에는 update 및 findAndModify 명령을 사용하여 업데이트한 문서가 포함됩니다.
- numDocsDel - 이 컬렉션에서 삭제된 문서 수입니다. 여기에는 deleteOne, deleteMany, remove, 및 findAndModify 명령을 사용하여 삭제된 문서가 포함됩니다.
- lastReset - 이 카운터를 마지막으로 재설정한 시간입니다. 이 명령으로 제공된 통계는 클러스터를 시작/중지하거나 인스턴스를 스케일 업/다운할 때 재설정됩니다.

db.collection.stats()를 실행한 예제 출력은 다음과 같습니다.

```
{
    "ns" : "db.test",
    "count" : ...,
    "size" : ...,
    "avgObjSize" : ...,
    "storageSize" : ...,
    "capped" : false,
    "nindexes" : ...,
    "totalIndexSize" : ...,
    "indexSizes" : {
        "_id_" : ...,
        "x_1" : ...
    },
    "collScans" : ...,
    "idxScans" : ...,
```

```
    "opCounter" : {
        "numDocsIns" : ...,
        "numDocsUpd" : ...,
        "numDocsDel" : ...
    },
    "cacheStats" : {
        "collBlksHit" : ...,
        "collBlksRead" : ..,
        "collHitRatio" : ...,
        "idxBlksHit" : ...,
        "idxBlksRead" : ...,
        "idxHitRatio" : ...
    },
    "lastReset" : "2022-09-02 19:41:40.471473+00",
    "ok" : 1,
    "operationTime" : Timestamp(1662159707, 1)
}
```

이 stats 명령은 Mongo API를 통해 삽입, 업데이트 및 삭제 작업에 대한 컬렉션별 카운터를 볼 때 사용해야 합니다. 컬렉션별 작업 카운터를 보는 또 다른 방법은 DML 감사를 활성화하는 것입니다. 1분 간격 동안의 모든 컬렉션에 대한 삽입, 업데이트 및 삭제 작업 수는 CloudWatch를 사용하여 Amazon DocumentDB 모니터링에서 확인할 수 있습니다.

캐시 성능을 분석하려면 어떻게 해야 하나요?

캐시 성능을 분석하면 캐시와 비교하여 디스크에서 읽은 데이터의 양을 기반으로 데이터 검색 효율성과 시스템 성능에 대한 통찰력을 얻을 수 있습니다. 캐시 성능에 대한 통찰력을 제공하기 위해 캐시 적중 수(캐시에서 읽은 데이터) 및 캐시 실패(캐시에 없고 디스크에서 읽은 데이터) 에 대한 캐시 통계를 제공합니다. 특정 컬렉션에 대한 캐시 통계는 해당 컬렉션에서 다음 명령을 실행하여 확인할 수 있습니다.

```
db.collection.stats()
```

이 명령 출력의 `cacheStats` 필드 값은 컬렉션에 대한 캐시 통계는 물론 컬렉션에 생성된 인덱스의 전체 캐시 통계도 제공합니다. 이러한 통계는 다음과 같습니다.

- **`collBlksHit`** - 이 컬렉션에 대한 작업 중에 캐시에서 읽은 블록 수입니다.
- **`collBlksRead`** - 이 컬렉션에 대한 작업 중에 디스크에서 읽은 블록 수(캐시 누락)입니다.
- **`collHitRatio`** - 이 컬렉션의 캐시 적중률(`100 * [collBlksHit / (collBlksHit + collBlksRead)]`).

- **idxBlksHit** - 이 컬렉션에서 생성된 인덱스에 대해 캐시에서 읽은 블록 수입니다.
- **idxBlksRead** - 이 컬렉션에서 생성된 인덱스에 대해 디스크에서 읽은 블록 수(캐시 누락)입니다.
- **idxHitRatio** - 이 컬렉션에서 생성된 인덱스의 캐시 적중률(`100 * [idxBlksHit / (idxBlksHit + idxBlksRead)]`).
- **lastReset** - 이 통계가 마지막으로 재설정된 시간. `db.collection.stats()`에 의해 제공된 통계는 클러스터를 시작/중지하거나 인스턴스를 스케일 업/다운할 때 재설정됩니다.

`indexStats` 명령을 사용하여 각 인덱스의 `idxBlksHit` 및 `idxBlksRead` 필드의 분류를 확인할 수도 있습니다. 인덱스별 캐시 통계는 다음 명령을 실행하여 확인할 수 있습니다.

```
db.collection.aggregate([{$indexStats:{}}]).pretty()
```

각 인덱스에 대해 `cacheStats` 필드 아래에서 다음과 같은 캐시 통계를 찾을 수 있습니다.

- **blksHit** - 이 인덱스에 대해 캐시에서 읽은 블록 수입니다.
- **blksRead** - 이 인덱스에 대해 디스크에서 읽은 블록 수입니다.
- **blksHitRatio** - `100 * [blksHit / (blksHit + blksRead)]`에서 캐시 적중률을 소수점 네 자리로 반올림하여 계산합니다.

장시간 실행 중이거나 차단된 쿼리를 찾아서 종료하려면 어떻게 해야 하나요?

사용자 쿼리는 최적화되지 않은 쿼리 계획으로 인해 실행 속도가 느려지거나 리소스 경합으로 인해 차단될 수 있습니다.

최적이지 않은 쿼리 계획으로 인해 느려지고 오래 실행되는 쿼리나, 속도가 느려지거나 리소스 경합으로 인해 차단된 쿼리를 찾으려면 `currentOp` 명령을 사용하세요. 명령을 필터링하면 종료할 쿼리의 목록을 좁힐 수 있습니다. 쿼리를 종료하려면 장시간 실행 중인 쿼리와 연결된 `opid`가 필요합니다.

다음 쿼리는 `currentOp` 명령을 사용하여 차단되었거나 10초 이상 실행 중인 모든 쿼리를 나열합니다.

```
db.adminCommand({
    aggregate: 1,
    pipeline: [
```

```
        {$currentOp: {}},
        {$match:
            {$or: [
                {secs_running: {$gt: 10}},
                {WaitState: {$exists: true}}]}},
        {$project: {_id:0, opid: 1, secs_running: 1}}],
    cursor: {}
});
```

다음으로, 쿼리 결과를 좁혀 10초를 초과하여 실행 중인 쿼리의 opid를 찾아 종료할 수 있습니다.

10초를 초과하여 실행 중인 쿼리를 찾아 종료하려면

1. 쿼리의 opid를 찾습니다.

```
db.adminCommand({
    aggregate: 1,
    pipeline: [
        {$currentOp: {}},
        {$match:
            {$or:
                [{secs_running: {$gt: 10}},
                 {WaitState: {$exists: true}}]}}],
    cursor: {}
});
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
    "waitedMS" : NumberLong(0),
    "cursor" : {
        "firstBatch" : [
            {
                "opid" : 24646,
                "secs_running" : 12
            }
        ],
        "id" : NumberLong(0),
        "ns" : "admin.$cmd"
    },
    "ok" : 1
}
```

2. killOp 작업을 사용하여 쿼리를 종료합니다.

```
db.adminCommand({killOp: 1, op: 24646});
```

쿼리 계획을 보고 쿼리를 최적화하려면 어떻게 해야 하나요?

쿼리의 실행 속도가 느린 경우 이 상황은 쿼리 실행 시 모음 전체 검색하여 관련 문서를 선택해야 하기 때문에 발생했을 수 있습니다. 경우에 따라 적절한 인덱스를 생성하면 쿼리 실행 속도가 빨라질 수도 있습니다. 이 시나리오를 찾아내어 인덱스를 생성할 기준 필드를 결정하려면 explain 명령을 사용합니다.

> **Note**
>
> Amazon DocumentDB는 분산, 내결함성, 자가 치유 스토리지 시스템을 활용하는 목적으로 만들어진 목적별 데이터베이스 엔진에서 MongoDB 3.6 API를 에뮬레이션합니다. 그 결과, 쿼리 계획과 explain()의 출력은 Amazon DocumentDB와 MongoDB 간에 다를 수 있습니다. 쿼리 계획을 제어하려는 고객은 $hint 연산자를 사용하여 기본 인덱스를 선택할 수 있습니다.

다음과 같이 explain 명령에서 개선하려는 쿼리를 실행합니다.

```
db.runCommand({explain: {<query document>}})
```

예는 다음과 같습니다.

```
db.runCommand({explain:{
    aggregate: "sample-document",
    pipeline: [{$match: {x: {$eq: 1}}}],
    cursor: {batchSize: 1}}
});
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
    "queryPlanner" : {
        "plannerVersion" : 1,
        "namespace" : "db.test",
        "winningPlan" : {
```

```
            "stage" : "COLLSCAN"
        }
    },
    "serverInfo" : {
        "host" : "...",
        "port" : ...,
        "version" : "..."
    },
    "ok" : 1
}
```

위의 출력은 $match 단계에서 전체 모음을 검색하고 각 문서의 "x" 필드가 1인지 확인해야 함을 나타냅니다. 모음에 여러 문서가 있는 경우 모음 검색 및 전체 쿼리 성능이 매우 느려질 수 있습니다. 그러므로 explain 명령의 출력에 "COLLSCAN"이 있는 경우 이는 적정 인덱스를 생성하여 쿼리 성능을 향상시킬 수 있음을 나타냅니다.

이 예제에서 쿼리는 모든 문서에서 "x" 필드가 1인지 확인합니다. 따라서 "x" 필드에서 인덱스를 작성하면 쿼리가 전체 컬렉션 스캔을 피하고 인덱스를 사용하여 관련 문서를 더 빨리 반환할 수 있습니다.

"x" 필드에 인덱스를 생성한 후 explain 출력은 다음과 같습니다.

```
{
    "queryPlanner" : {
         "plannerVersion" : 1,
         "namespace" : "db.test",
         "winningPlan" : {
             "stage" : "IXSCAN",
             "indexName" : "x_1",
             "direction" : "forward"
         }
    },
    "serverInfo" : {
        "host" : "...",
        "port" : ...,
        "version" : "..."
    },
    "ok" : 1
}
```

그러므로 "x" 필드에 대한 인덱스를 생성하면 $match 단계에서 인덱스 검색을 통해 조건자 "x = 1"을 평가해야 하는 문서 수를 줄일 수 있습니다.

작은 컬렉션에서는 성능 이득이 무시할 수준인 경우 Amazon DocumentDB 쿼리 프로세서는 인덱스를 사용하지 않도록 결정할 수 있습니다.

엘라스틱 클러스터에서 쿼리 계획을 확인하려면 어떻게 해야 하나요?

엘라스틱 클러스터의 쿼리 계획을 검사하려면 `explain` 명령을 사용합니다. 다음은 샤딩된 컬렉션을 대상으로 하는 찾기 쿼리의 예제 `explain` 작업입니다.

```
db.runCommand(
   {
     explain: { find: "cities", filter: {"name": "Seoul"}}
   }
)
```

> **Note**
>
> Amazon DocumentDB는 특별히 구축된 목적별 데이터베이스 엔진에서 MongoDB를 에뮬레이션합니다. 그 결과, 쿼리 계획과 `explain()`의 출력은 Amazon DocumentDB와 MongoDB 간에 다를 수 있습니다. `$hint` 연산자를 사용하여 기본 인덱스를 선택하도록 하여 쿼리 계획을 제어할 수 있습니다.

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
  "queryPlanner" : {
    "elasticPlannerVersion" : 1,
    "winningPlan" : {
      "stage" : "SINGLE_SHARD",
      "shards" : [
        {
          "plannerVersion" : 1,
          "namespace" : "population.cities",
          "winningPlan" : {
            "stage" : "SHARD_MERGE",
            "shards" : [
              {
                "shardName" : "f2cf5cfd-fe9c-40ca-b4e5-298ca0d11111",
                "plannerVersion" : 1,
                "namespace" : "population.cities",
                "winningPlan" : {
```

```
                "stage" : "PARTITION_MERGE",
                "inputStages" : [
                  {
                    "stage" : "COLLSCAN",
                    "partitionCount" : 21
                  }
                ]
              }
            },
            {
              "shardName" : "8f3f80e2-f96c-446e-8e9d-aab8c7f22222",
              "plannerVersion" : 1,
              "namespace" : "population.cities",
              "winningPlan" : {
                "stage" : "PARTITION_MERGE",
                "inputStages" : [
                  {
                    "stage" : "COLLSCAN",
                    "partitionCount" : 21
                  }
                ]
              }
            },
            {
              "shardName" : "32c5a06f-1b2b-4af1-8849-d7c4a033333",
              "plannerVersion" : 1,
              "namespace" : "population.cities",
              "winningPlan" : {
                "stage" : "PARTITION_MERGE",
                "inputStages" : [
                  {
                    "stage" : "COLLSCAN",
                    "partitionCount" : 22
                  }
                ]
              }
            }
          ]
        },
        "shardName" : "32c5a06f-1b2b-4af1-8849-d7c4a0f3fb58"
      }
    ]
  }
},
```

```
  "serverInfo" : {
    "host" : "example-4788267630.us-east-1.docdb-elastic.amazonaws.com:27017",
    "version" : "5.0.0"
  },
  "ok" : 1,
  "operationTime" : Timestamp(1695097923, 1)
}
```

위 출력은 3개 샤드 클러스터의 find 쿼리에 대한 쿼리 계획을 보여줍니다. 각 샤드에는 입력 단계가 서로 다를 수 있는 데이터 파티션이 여러 개 있습니다. 이 예시에서는 결과가 각 샤드의 'PARTITION_MERGE' 단계에서 병합되기 전에 모든 파티션에서 'COLLSCAN'(컬렉션 스캔)이 실행됩니다. 그런 다음 샤드 전체의 결과가 'SHARD_MERGE' 단계에서 병합된 후 클라이언트로 다시 전송됩니다.

인스턴스에서 실행 중인 작업을 모두 나열하려면 어떻게 해야 하나요?

사용자 또는 기본 사용자로서, 진단 및 문제 해결을 위해 인스턴스에서 실행 중인 모든 현재 작업을 나열해야 하는 경우가 흔히 있습니다. (사용자 관리에 대한 자세한 내용은 Amazon DocumentDB 사용자 관리 섹션을 참조하세요.)

mongo 쉘을 사용하여 다음 쿼리를 통해 Amazon DocumentDB 인스턴스에서 실행 중인 모든 작업을 나열할 수 있습니다.

```
db.adminCommand({currentOp: 1, $all: 1});
```

이 쿼리는 현재 인스턴스에서 실행 중인 모든 사용자 쿼리 및 내부 시스템 작업의 전체 목록을 반환합니다.

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
    "inprog" : [
        {
            "desc" : "INTERNAL"
        },
        {
            "desc" : "TTLMonitor",
            "active" : false
        },
        {
            "client" : ...,
            "desc" : "Conn",
```

```
            "active" : true,
            "killPending" : false,
            "opid" : 195,
            "ns" : "admin.$cmd",
            "command" : {
                "currentOp" : 1,
                "$all" : 1
            },
            "op" : "command",
            "$db" : "admin",
            "secs_running" : 0,
            "microsecs_running" : NumberLong(68),
            "clientMetaData" : {
            "application" : {
                "name" : "MongoDB Shell"
            },
            "driver" : {
                ...
            },
            "os" : {
                ...
            }
        }
    },
    {
        "desc": "GARBAGE_COLLECTION",
        "garbageCollection": {
            "databaseName": "testdb",
            "collectionName": "testCollectionA"
        },
        "secs_running": 3,
        "microsecs_running": NumberLong(3123456)
    },
    {
        "desc": "GARBAGE_COLLECTION",
        "garbageCollection": {
            "databaseName": "testdb",
            "collectionName": "testCollectionB"
        },
        "secs_running": 4,
        "microsecs_running": NumberLong(4123456)
    }
],
"ok" : 1
```

```
}
```

"desc" 필드의 유효한 값은 다음과 같습니다.

- **INTERNAL** - 커서 정리 또는 기한 경과 사용자 정리 작업 같은 내부 시스템 작업입니다.
- **TTLMonitor** - 유지 시간(TTL) 모니터 스레드입니다. 실행 중 상태가 "active" 필드에 반영됩니다.
- **GARBAGE_COLLECTION** - 내부 가비지 수집기 스레드입니다.
- **CONN** - 사용자 쿼리입니다.
- **CURSOR** - 이 작업은 사용자가 'getMore' 명령을 호출하여 다음 일괄 결과를 가져오기를 기다리는 유휴 상태의 커서입니다. 이 상태에서는 커서가 메모리를 소비하지만 컴퓨팅은 소비하지 않습니다.

이전 출력에는 시스템에서 실행 중인 모든 사용자 쿼리도 나열됩니다. 각 사용자 쿼리는 데이터베이스 및 모음의 컨텍스트에서 실행되며 이들의 결합을 네임스페이스라고 합니다. 각 사용자 쿼리의 네임스페이스는 "ns" 필드에서 사용할 수 있습니다.

때로는 특정 네임스페이스에서 실행 중인 모든 사용자 쿼리를 나열해야 합니다. 따라서 "ns" 필드에서 이전 출력을 필터링해야 합니다. 필터링할 출력을 얻기 위한 예제 쿼리는 다음과 같습니다. 쿼리는 데이터베이스 "db" 및 "test" 컬렉션(즉, "db.test" 네임스페이스)에서 현재 실행 중인 모든 사용자 쿼리를 나열합니다.

```
db.adminCommand({aggregate: 1,
    pipeline: [{$currentOp: {allUsers: true, idleConnections: true}},
               {$match: {ns: {$eq: "db.test"}}}],
    cursor: {}
});
```

시스템의 기본 사용자는 모든 사용자의 쿼리와 및 모든 내부 시스템 작업을 볼 수 있습니다. 다른 모든 사용자는 각자의 쿼리만 볼 수 있습니다.

총 쿼리 수와 내부 시스템 작업 수가 기본 배치 커서 크기를 초과하는 경우 mongo 쉘에서 반복자 객체 'it'이 자동으로 생성되어 나머지 결과를 표시합니다. 모든 결과가 표시될 때까지 'it' 명령을 계속 실행합니다.

쿼리 진행 상황을 어떻게 알 수 있나요?

사용자 쿼리는 최적화되지 않은 쿼리 계획으로 인해 실행 속도가 느려지거나 리소스 경합으로 인해 차단될 수 있습니다. 이러한 쿼리를 디버깅하는 작업은 다단계 프로세스로 경우에 따라서는 동일 단계를 여러 번 실행해야 할 수도 있습니다.

디버깅의 첫 번째 단계는 장시간 실행 중이거나 차단된 모든 쿼리를 나열하는 것입니다. 다음 쿼리는 실행 시간이 10초를 넘었거나 리소스 대기 중인 모든 사용자 쿼리를 나열합니다.

```
db.adminCommand({aggregate: 1,
                 pipeline: [{$currentOp: {}},
                            {$match: {$or: [{secs_running: {$gt: 10}},
                                            {WaitState: {$exists: true}}]}},
                            {$project: {_id:0,
                                        opid: 1,
                                        secs_running: 1,
                                        WaitState: 1,
                                        blockedOn: 1,
                                        command: 1}}],
                 cursor: {}
                });
```

위의 쿼리를 주기적으로 반복 실행하여 쿼리 목록의 변경 여부를 확인하고 장시간 실행 중이거나 차단된 쿼리를 찾습니다.

해당 쿼리의 출력 문서에 WaitState 필드가 있는 경우 이는 쿼리 실행 속도가 느리거나 쿼리 실행이 차단된 이유가 리소스 경합 때문임을 나타냅니다. 리소스 경합은 I/O, 내부 시스템 작업 또는 다른 사용자 쿼리가 원인일 수 있습니다.

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
    "waitedMS" : NumberLong(0),
    "cursor" : {
        "firstBatch" : [
            {
                "opid" : 201,
                "command" : {
                    "aggregate" : ...
                },
                "secs_running" : 208,
```

```
                "WaitState" : "IO"
            }
        ],
        "id" : NumberLong(0),
        "ns" : "admin.$cmd"
    },
    "ok" : 1
}
```

동일 인스턴스에 대해 동시에 여러 모음에 대한 여러 쿼리가 있거나 쿼리가 실행 중인 데이터 세트에 비해 인스턴스 크기가 너무 작은 경우 IO가 병목 요인일 수 있습니다. 쿼리가 읽기 전용 쿼리인 경우 이전 상황에 대한 완화 방법은 여러 복제본에 대해 각 모음별로 쿼리를 분리하는 것입니다. 다양한 모음을 동시에 업데이트하거나 인스턴스 크기가 데이터 세트에 비해 너무 작은 경우 완화 방법은 인스턴스 크기를 스케일 업하는 것입니다.

리소스 경합이 다른 사용자 쿼리로 인해 발생하는 경우 출력 문서의 "blockedOn" 필드에 이 쿼리에 영향을 주는 쿼리의 "opid"가 있습니다. 모든 쿼리의 "WaitState" 및 "blockedOn" 필드 체인 뒤에 "opid"를 사용하여 체인의 헤드에서 쿼리를 찾습니다.

체인의 헤드에 있는 작업이 내부 작업이면 이 경우 유일한 완화 방법은 쿼리를 종료했다가 잠시 후에 다시 실행하는 방법뿐입니다.

아래에는 다른 작업이 소유하는 모음 잠금에서 찾기 쿼리가 차단된 샘플 출력이 나와 있습니다.

```
{
    "inprog" : [
        {
            "client" : "...",
            "desc" : "Conn",
            "active" : true,
            "killPending" : false,
            "opid" : 75,
            "ns" : "...",
            "command" : {
                "find" : "...",
                "filter" : {

                }
            },
            "op" : "query",
            "$db" : "test",
            "secs_running" : 9,
```

```
            "microsecs_running" : NumberLong(9449440),
            "threadId" : 24773,
            "clientMetaData" : {
                "application" : {
                   "name" : "MongoDB Shell"
                },
                "driver" : {
                    ...
                },
                "os" : {
                    ...
                }
            },
            "WaitState" : "CollectionLock",
            "blockedOn" : "INTERNAL"
        },
        {
            "desc" : "INTERNAL"
        },
        {
            "client" : "...",
            ...
            "command" : {
                "currentOp" : 1
            },
            ...
        }
    ],
    "ok" : 1
}
```

"WaitState"의 값이 "Latch", "SystemLock", "BufferLock", "BackgroundActivity" 또는 "Other"이면 리소스 경합의 원인은 내부 시스템 작업입니다. 이 상황이 장시간 지속되는 경우 유일한 완화 방법은 쿼리를 종료했다가 나중에 다시 실행하는 방법뿐입니다.

시스템이 갑자기 느리게 실행되는 이유를 어떻게 알 수 있나요?

다음은 시스템 속도가 느려지는 몇 가지 일반적인 이유입니다.

- 동시 쿼리 간 과도한 리소스 경합
- 시간이 지남에 따라 증가하는 활성 동시 쿼리 수
- "GARBAGE_COLLECTION" 같은 내부 시스템 작업

시간 경과에 따른 시스템 사용량을 모니터링하려면 다음 "currentOp" 쿼리를 주기적으로 실행하고 그 결과를 외부 저장소로 출력합니다. 이 쿼리는 시스템의 각 네임스페이스에서 쿼리 및 작업 수를 셉니다. 그 다음에는 시스템 사용량 결과를 분석하여 시스템에 대한 부하를 파악하고 그에 따른 적절한 조치를 취할 수 있습니다.

```
db.adminCommand({aggregate: 1,
                 pipeline: [{$currentOp: {allUsers: true, idleConnections: true}},
                            {$group: {_id: {desc: "$desc", ns: "$ns", WaitState:
 "$WaitState"}, count: {$sum: 1}}}],
                 cursor: {}
                });
```

이 쿼리는 각 네임스페이스에서 실행 중인 모든 쿼리, 모든 내부 시스템 작업, 그리고 네임스페이스별 고유한 대기 상태 수의 집계 값을 반환합니다.

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{
    "waitedMS" : NumberLong(0),
    "cursor" : {
        "firstBatch" : [
            {
                "_id" : {
                    "desc" : "Conn",
                    "ns" : "db.test",
                    "WaitState" : "CollectionLock"
                },
               "count" : 2
            },
            {
                "_id" : {
                    "desc" : "Conn",
                    "ns" : "admin.$cmd"
                },
                "count" : 1
            },
            {
                "_id" : {
                    "desc" : "TTLMonitor"
                },
                "count" : 1
            }
```

```
        ],
        "id" : NumberLong(0),
        "ns" : "admin.$cmd"
    },
    "ok" : 1
}
```

이전 출력에서, 모음 잠금에 대해 차단된 네임스페이스 "db.test"에 사용자 쿼리 2개가 있고, 네임스페이스 "admin.$cmd"의 쿼리 1개, 그리고 내부 "TTLMonitor" 작업 1개가 있습니다.

출력에 차단 대기 상태인 쿼리가 여러 개 있으면 장시간 실행 중이거나 차단된 쿼리를 찾아서 종료하려면 어떻게 해야 하나요? 섹션을 참조합니다.

하나 이상의 클러스터 인스턴스에서 높은 CPU 사용률의 원인을 어떻게 확인하나요?

다음 섹션은 인스턴스 CPU 사용률이 높은 원인을 식별하는 데 도움이 될 수 있습니다. 워크로드에 따라 결과가 달라질 수 있습니다.

- 인스턴스가 갑자기 느리게 실행되는 이유를 확인하려면 시스템이 갑자기 느리게 실행되는 이유를 어떻게 알 수 있나요? 섹션을 참조하세요.
- 특정 인스턴스에서 장기 실행 쿼리를 식별하고 종료하려면 장시간 실행 중이거나 차단된 쿼리를 찾아서 종료하려면 어떻게 해야 하나요? 섹션을 참조합니다.
- 쿼리가 진행 중인지 알아보려면 쿼리 진행 상황을 어떻게 알 수 있나요? 섹션을 참조하세요.
- 쿼리를 실행하는 데 시간이 오래 걸리는 이유를 확인하려면 쿼리 계획을 보고 쿼리를 최적화하려면 어떻게 해야 하나요? 섹션을 참조하세요.
- 시간의 경과에 따라 장기 실행 쿼리를 추적하려면 Amazon DocumentDB 작업 프로파일링 섹션을 참조합니다.

높은 인스턴스 CPU 사용률의 이유에 따라 다음 중 하나 이상을 수행하면 도움이 될 수 있습니다.

- 기본 인스턴스의 CPU 사용률이 높지만 복제본 인스턴스는 그렇지 않은 경우 클라이언트 읽기 기본 설정(예: `secondaryPreferred`)을 통해 복제본에 읽기 트래픽을 분산시키는 것을 고려하세요. 자세한 내용은 Amazon DocumentDB에 복제본 세트로 연결 단원을 참조하십시오.

  읽기에 복제본을 사용하면 기본 인스턴스가 더 많은 쓰기 트래픽을 처리할 수 있게 하여 클러스터 리소스를 더 잘 활용할 수 있습니다. 복제본에서의 읽기는 최종적으로 일관됩니다.

- 높은 CPU 사용률이 쓰기 워크로드의 결과인 경우 클러스터 인스턴스의 크기를 더 큰 인스턴스 유형으로 변경하면 워크로드를 처리하는 데 사용할 수 있는 CPU 코어 수가 증가합니다. 자세한 내용은 인스턴스 및 인스턴스 클래스 사양 섹션을 참조하세요.
- 모든 클러스터 인스턴스의 CPU 사용률이 높고 워크로드에서 읽기 전용 복제본을 사용하는 경우 클러스터에 복제본을 더 추가하면 읽기 트래픽에 사용할 수 있는 리소스가 늘어납니다. 자세한 내용은 클러스터에 Amazon DocumentDB 인스턴스 추가 단원을 참조하십시오.

인스턴스에서 열려 있는 커서를 어떻게 확인하나요?

Amazon DocumentDB 인스턴스에 연결되면 `db.runCommand("listCursors")` 명령을 사용하여 해당 인스턴스에서 열려 있는 커서를 나열할 수 있습니다. 인스턴스 유형에 따라 특정 Amazon DocumentDB 인스턴스에서 한 번에 최대 4,560개의 활성 커서를 열 수 있습니다. 일반적으로 커서는 인스턴스의 리소스를 사용하고 상한이 있으므로 더 이상 사용하지 않는 커서를 닫는 것이 좋습니다. 구체적인 한도는 Amazon DocumentDB 할당량 및 제한 섹션을 참조하세요.

```
db.runCommand("listCursors")
```

현재 Amazon DocumentDB 엔진 버전은 어떻게 확인하나요?

현재 Amazon DocumentDB 엔진 버전을 확인하려면 다음 명령을 실행합니다.

```
db.runCommand({getEngineVersion: 1})
```

이 작업의 출력은 다음과 같습니다(JSON 형식).

```
{ "engineVersion" : "2.x.x", "ok" : 1 }
```

> **Note**
>
> Amazon DocumentDB 3.6의 엔진 버전은 1.x.x이고 Amazon DocumentDB 4.0의 엔진 버전은 2.x.x입니다.

인덱스 사용량을 분석하고 사용하지 않는 인덱스를 식별하려면 어떻게 해야 하나요?

지정된 컬렉션에 대한 인덱스를 식별하려면 다음 명령을 실행합니다.

```
db.collection.getIndexes()
```

collStats 및 indexStats 명령을 사용하여 컬렉션에서 작업을 수행하는 동안 사용되는 인덱스의 양을 분석할 수 있습니다. 인덱스를 사용하여 수행한 총 스캔 수(인덱스 스캔)를 인덱스 없이 수행한 스캔 수(컬렉션 스캔)와 비교하여 보려면 다음 명령을 실행합니다.

```
db.collection.stats()
```

이 명령의 출력에는 다음과 같은 값이 포함됩니다.

- **idxScans**- 인덱스를 사용하여 이 컬렉션에 대해 수행한 스캔 횟수.
- **collScans**- 인덱스를 사용하지 않고 이 컬렉션에 대해 수행한 스캔 횟수. 이러한 스캔에서는 컬렉션에 있는 문서를 한 번에 하나씩 살펴봐야 했을 것입니다.
- **lastReset** - 이 카운터를 마지막으로 재설정한 시간. 이 명령으로 제공된 통계는 클러스터를 시작/중지하거나 인스턴스를 스케일 업/다운할 때 재설정됩니다.

각 인덱스의 사용량에 대한 분석은 다음 명령의 출력에서 확인할 수 있습니다. 인덱스를 유지 관리하는 데 사용되는 불필요한 컴퓨팅, 스토리지 및 I/O를 제거하므로 성능을 개선하고 비용을 절감하기 위해 사용되지 않는 인덱스를 정기적으로 식별하고 제거하는 것이 가장 좋습니다.

```
db.collection.aggregate([{$indexStats:{}}]).pretty()
```

이 명령의 출력은 컬렉션에 생성된 각 인덱스에 대해 다음 값을 제공합니다.

- **ops** - 인덱스를 사용한 작업 수. 작업 로드가 충분히 오랫동안 실행 중이고 작업 로드가 정상 상태에 있다고 확신하는 경우 ops 값이 0이면 인덱스가 전혀 사용되지 않음을 나타냅니다.
- **numDocsRead**- 이 인덱스를 사용하여 작업을 수행하는 동안 읽은 문서 수.
- **since** - Amazon DocumentDB에서 인덱스 사용량에 대한 통계를 수집하기 시작한 이후의 시간으로서, 일반적으로 마지막 데이터베이스 재시작 또는 유지 관리 작업 이후의 값.
- **size** - 이 인덱스의 크기(바이트).

다음은 위 명령을 실행한 출력 예제입니다.

```
{
    "name" : "_id_",
```

```
    "key" : {
        "_id" : 1
    },
    "host" : "example-host.com:12345",
    "size" : NumberLong(...),
    "accesses" : {
        "ops" : NumberLong(...),
        "docsRead" : NumberLong(...),
        "since" : ISODate("...")
    },
    "cacheStats" : {
        "blksRead" : NumberLong(...),
        "blksHit" : NumberLong(...),
        "hitRatio" : ...
    }
}
{
    "name" : "x_1",
    "key" : {
        "x" : 1
    },
    "host" : "example-host.com:12345",
    "size" : NumberLong(...),
    "accesses" : {
        "ops" : NumberLong(...),
        "docsRead" : NumberLong(...),
        "since" : ISODate("...")
    },
    "cacheStats" : {
        "blksRead" : NumberLong(...),
        "blksHit" : NumberLong(...),
        "hitRatio" : ...
    }
}
```

컬렉션의 전체 인덱스 크기를 확인하려면 다음 명령을 실행합니다.

```
db.collection.stats()
```

사용되지 않는 인덱스를 삭제하려면 다음 명령을 실행합니다.

```
db.collection.dropIndex("indexName")
```

누락된 인덱스는 어떻게 식별하나요?

Amazon DocumentDB 프로파일러를 사용하여 느린 쿼리를 로깅할 수 있습니다. 느린 쿼리 로그에 반복적으로 나타나는 쿼리는 해당 쿼리의 성능을 향상하기 위해 추가 인덱스가 필요함을 나타낼 수 있습니다.

하나 이상의 단계를 수행하는 단계가 하나 이상 있는 장기 실행 쿼리를 찾아 유용한 인덱스의 기회를 식별할 수 있습니다. 즉, 쿼리COLLSCAN에 대한 응답을 제공하려면 쿼리 단계가 컬렉션의 모든 문서를 읽어야 합니다.

다음 예제에서는 대규모 컬렉션에서 실행된 택시 타기 컬렉션에 대한 쿼리를 보여 줍니다.

```
db.rides.count({"fare.totalAmount":{$gt:10.0}}))
```

이 예제를 실행하려면 `fare.totalAmount` 필드에 인덱스가 없으므로 쿼리는 컬렉션 스캔(즉, 컬렉션의 모든 단일 문서 읽기)을 수행해야 했습니다. 이 쿼리에 대한 Amazon DocumentDB 프로파일러의 출력은 다음과 같습니다.

```
{
    ...
    "cursorExhausted": true,
    "nreturned": 0,
    "responseLength": 0,
    "protocol": "op_query",
    "millis": 300679,
    "planSummary": "COLLSCAN",
    "execStats": {
        "stage": "COLLSCAN",
        "nReturned": "0",
        "executionTimeMillisEstimate": "300678.042"
    },
    "client": "172.31.5.63:53878",
    "appName": "MongoDB Shell",
    "user": "example"
}
```

이 예에서 쿼리 속도를 높이기 위해 아래 그림과 같이 `fare.totalAmount` 색인을 작성하려고 합니다.

```
db.rides.createIndex( {"fare.totalAmount": 1}, {background: true} )
```

> **Note**
>
> 포그라운드에서 생성된 인덱스(인덱스를 만들 때 {background:true} 옵션이 제공되지 않은 경우)는 독점적 인 쓰기 잠금을 통해 인덱스 빌드가 완료될 때까지 애플리케이션이 컬렉션에 데이터를 쓰지 못하게 합니다. 프로덕션 클러스터에서 인덱스를 만들 때 이러한 잠재적인 영향을 고려해야 합니다. 인덱스를 만들 때 {background:true}로 설정하는 것이 좋습니다.

일반적으로 카디널리티가 높은 필드(예: 많은 수의 고유 값)에 인덱스를 만드는 경향이 있습니다. 카디널리티가 낮은 필드에 인덱스를 만들면 사용되지 않는 큰 인덱스가 발생할 수 있습니다. Amazon DocumentDB 쿼리 최적화 프로그램은 쿼리 계획을 만들 때 컬렉션의 전체 크기와 인덱스의 선택성을 고려합니다. 인덱스가 있는 경우에도 쿼리 프로세서가 COLLSCAN을 선택하는 경우가 있습니다. 이는 인덱스를 사용하는 경우 전체 컬렉션을 스캔하는 데 비해 성능 이점이 없을 것으로 쿼리 프로세서가 예측할 때 발생합니다. 특정 인덱스를 활용하기 위해 쿼리 프로세서를 강제하려면 아래 그림과 같이 hint() 연산자를 사용할 수 있습니다.

```
db.collection.find().hint("indexName")
```

유용한 쿼리 요약

다음 쿼리는 Amazon DocumentDB의 성능 및 리소스 사용률을 모니터링하는 데 유용할 수 있습니다.

- 다음 명령을 사용하여 작업 카운터, 캐시 통계, 액세스 통계, 크기 통계를 비롯한 특정 컬렉션에 대한 통계를 볼 수 있습니다.

  ```
  db.collection.stats()
  ```

- 다음 명령을 사용하여 인덱스 크기, 인덱스별 캐시 통계, 인덱스 사용 통계를 포함하여 컬렉션에 생성된 각 인덱스에 대한 통계를 볼 수 있습니다.

  ```
  db.collection.aggregate([{$indexStats:{}}]).pretty()
  ```

- 다음 쿼리를 사용하여 모든 활동을 나열하세요.

  ```
  db.adminCommand({currentOp: 1, $all: 1});
  ```

- 다음 코드는 오래 실행되거나 차단된 모든 쿼리를 나열합니다.

  ```
  db.adminCommand({aggregate: 1,
  ```

```
                pipeline: [{$currentOp: {}},
                           {$match: {$or: [{secs_running: {$gt: 10}},
                                           {WaitState: {$exists: true}}]}},
                           {$project: {_id:0,
                                       opid: 1,
                                       secs_running: 1,
                                       WaitState: 1,
                                       blockedOn: 1,
                                       command: 1}}],
                cursor: {}
               });
```

- 다음 코드는 쿼리를 종료합니다.

```
db.adminCommand({killOp: 1, op: <opid of running or blocked query>});
```

- 다음 코드를 사용하여 시스템 상태를 종합적으로 볼 수 있습니다.

```
db.adminCommand({aggregate: 1,
                 pipeline: [{$currentOp: {allUsers: true, idleConnections: true}},
                            {$group: {_id: {desc: "$desc", ns: "$ns", WaitState:
 "$WaitState"}, count: {$sum: 1}}}],
                 cursor: {}
                });
```

Amazon DocumentDB 클러스터, 인스턴스 및 리소스 관리 API 참조

이 섹션에서는 HTTP, AWS Command Line Interface ()AWS CLI또는 AWS SDK를 통해 액세스할 수 있는 Amazon DocumentDB(MongoDB 호환)의 클러스터, 인스턴스 및 리소스 관리 작업에 대해 설명합니다. 이러한 API를 사용하여 클러스터 및 인스턴스를 생성, 삭제 및 수정할 수 있습니다.

> **Important**
>
> 이 API는 클러스터, 인스턴스 및 관련 리소스를 관리하는 데만 사용됩니다. 실행 중인 Amazon DocumentDB 클러스터에 연결하는 방법에 관한 자세한 내용은 시작 안내서 섹션을 참조하세요.

주제

- 작업
- 데이터 타입
- 일반적인 오류
- 공통 파라미터

작업

다음 작업은 Amazon DocumentDB (with MongoDB compatibility)에서 지원됩니다.

- AddSourceIdentifierToSubscription
- AddTagsToResource
- ApplyPendingMaintenanceAction
- CopyDBClusterParameterGroup
- CopyDBClusterSnapshot
- CreateDBCluster
- CreateDBClusterParameterGroup
- CreateDBClusterSnapshot

- CreateDBInstance
- CreateDBSubnetGroup
- CreateEventSubscription
- CreateGlobalCluster
- DeleteDBCluster
- DeleteDBClusterParameterGroup
- DeleteDBClusterSnapshot
- DeleteDBInstance
- DeleteDBSubnetGroup
- DeleteEventSubscription
- DeleteGlobalCluster
- DescribeCertificates
- DescribeDBClusterParameterGroups
- DescribeDBClusterParameters
- DescribeDBClusters
- DescribeDBClusterSnapshotAttributes
- DescribeDBClusterSnapshots
- DescribeDBEngineVersions
- DescribeDBInstances
- DescribeDBSubnetGroups
- DescribeEngineDefaultClusterParameters
- DescribeEventCategories
- DescribeEvents
- DescribeEventSubscriptions
- DescribeGlobalClusters
- DescribeOrderableDBInstanceOptions
- DescribePendingMaintenanceActions
- FailoverDBCluster
- FailoverGlobalCluster
- ListTagsForResource

- ModifyDBCluster
- ModifyDBClusterParameterGroup
- ModifyDBClusterSnapshotAttribute
- ModifyDBInstance
- ModifyDBSubnetGroup
- ModifyEventSubscription
- ModifyGlobalCluster
- RebootDBInstance
- RemoveFromGlobalCluster
- RemoveSourceIdentifierFromSubscription
- RemoveTagsFromResource
- ResetDBClusterParameterGroup
- RestoreDBClusterFromSnapshot
- RestoreDBClusterToPointInTime
- StartDBCluster
- StopDBCluster
- SwitchoverGlobalCluster

Amazon DocumentDB Elastic Clusters는 다음 작업을 지원합니다.

- ApplyPendingMaintenanceAction
- CopyClusterSnapshot
- CreateCluster
- CreateClusterSnapshot
- DeleteCluster
- DeleteClusterSnapshot
- GetCluster
- GetClusterSnapshot
- GetPendingMaintenanceAction
- ListClusters
- ListClusterSnapshots

- ListPendingMaintenanceActions
- ListTagsForResource
- RestoreClusterFromSnapshot
- StartCluster
- StopCluster
- TagResource
- UntagResource
- UpdateCluster

Amazon DocumentDB (with MongoDB compatibility)

다음 작업은 Amazon DocumentDB (with MongoDB compatibility)에서 지원됩니다.

- AddSourceIdentifierToSubscription
- AddTagsToResource
- ApplyPendingMaintenanceAction
- CopyDBClusterParameterGroup
- CopyDBClusterSnapshot
- CreateDBCluster
- CreateDBClusterParameterGroup
- CreateDBClusterSnapshot
- CreateDBInstance
- CreateDBSubnetGroup
- CreateEventSubscription
- CreateGlobalCluster
- DeleteDBCluster
- DeleteDBClusterParameterGroup
- DeleteDBClusterSnapshot
- DeleteDBInstance
- DeleteDBSubnetGroup
- DeleteEventSubscription
- DeleteGlobalCluster

- DescribeCertificates
- DescribeDBClusterParameterGroups
- DescribeDBClusterParameters
- DescribeDBClusters
- DescribeDBClusterSnapshotAttributes
- DescribeDBClusterSnapshots
- DescribeDBEngineVersions
- DescribeDBInstances
- DescribeDBSubnetGroups
- DescribeEngineDefaultClusterParameters
- DescribeEventCategories
- DescribeEvents
- DescribeEventSubscriptions
- DescribeGlobalClusters
- DescribeOrderableDBInstanceOptions
- DescribePendingMaintenanceActions
- FailoverDBCluster
- FailoverGlobalCluster
- ListTagsForResource
- ModifyDBCluster
- ModifyDBClusterParameterGroup
- ModifyDBClusterSnapshotAttribute
- ModifyDBInstance
- ModifyDBSubnetGroup
- ModifyEventSubscription
- ModifyGlobalCluster
- RebootDBInstance
- RemoveFromGlobalCluster
- RemoveSourceIdentifierFromSubscription
- RemoveTagsFromResource

- ResetDBClusterParameterGroup
- RestoreDBClusterFromSnapshot
- RestoreDBClusterToPointInTime
- StartDBCluster
- StopDBCluster
- SwitchoverGlobalCluster

AddSourceIdentifierToSubscription

서비스: Amazon DocumentDB (with MongoDB compatibility)

기존의 이벤트 알림 구독에 소스 식별자를 추가합니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 관한 정보는 공통 파라미터를 참조하십시오.

SourceIdentifier

추가할 이벤트 소스의 식별자입니다.

- 소스 유형이 인스턴스라면 `DBInstanceIdentifier`을 입력해야 합니다.
- 소스 유형이 보안 그룹이라면 `DBSecurityGroupName`을 제공해야 합니다.
- 소스 유형이 파라미터 그룹이라면 `DBParameterGroupName`을 제공해야 합니다.
- 소스 유형이 스냅샷이라면 `DBSnapshotIdentifier`를 입력해야 합니다.

유형: 문자열

필수 항목 여부: 예

SubscriptionName

소스 식별자를 추가하려는 Amazon DocumentDB 이벤트 알림 구독의 이름입니다.

유형: 문자열

필수 항목 여부: 예

응답 요소

서비스에서 다음 요소를 반환합니다.

EventSubscription

구독한 이벤트에 대한 세부 정보.

타입: EventSubscription 객체

오류

모든 작업에서 공통적으로 발생하는 오류에 대한 자세한 내용은 일반적인 오류 단원을 참조하십시오.

SourceNotFound

요청한 소스를 찾을 수 없습니다.

HTTP 상태 코드: 404

SubscriptionNotFound

구독의 이름이 존재하지 않습니다.

HTTP 상태 코드: 404

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go v2용 SDK
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

AddTagsToResource

서비스: Amazon DocumentDB (with MongoDB compatibility)

메타데이터 태그를 Amazon DocumentDB 리소스에 추가합니다. 이러한 태그를 비용 할당 보고와 함께 사용하여 Amazon DocumentDB 리소스와 연결된 비용을 추적하거나 Amazon DocumentDB에 대한 AWS Identity and Access Management (IAM) 정책의 `Condition` 문에서 추적할 수 있습니다. Amazon DocumentDB

요청 파라미터

모든 작업에 공통으로 적용되는 파라미터에 대한 자세한 내용은 공통 파라미터를 참조하세요.

ResourceName

태그가 추가된 Amazon DocumentDB 리소스입니다. 이 값은 Amazon 리소스 이름입니다.

유형: 문자열

필수 항목 여부: 예

Tags.Tag.N

Amazon DocumentDB 리소스에 할당할 태그입니다.

타입: Tag객체 배열

필수 여부: 예

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하세요.

DBClusterNotFoundFault

`DBClusterIdentifier`은 기존 클러스터를 참조하지 않습니다.

HTTP 상태 코드: 404

DBInstanceNotFound

`DBInstanceIdentifier`는 기존 인스턴스를 참조하지 않습니다.

HTTP 상태 코드: 404

DBSnapshotNotFound

`DBSnapshotIdentifier`는 기존 스냅샷을 참조하지 않습니다.

HTTP 상태 코드: 404

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

ApplyPendingMaintenanceAction

서비스: Amazon DocumentDB (with MongoDB compatibility)

대기 중인 유지 관리 작업을 리소스(예: Amazon DocumentDB 인스턴스)에 적용합니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 내용은 공통 파라미터를 참조하십시오.

ApplyAction

이 리소스에 적용할 대기 중인 유지 관리 작업입니다.

유효값: `system-update`, `db-upgrade`

유형: 문자열

필수 항목 여부: 예

OptInType

옵트인 요청의 유형을 지정하거나 옵트인 요청을 실행 취소하는 값입니다. `immediate` 유형의 옵트인 요청은 실행 취소할 수 없습니다.

유효한 값:

- `immediate` - 유지 관리 작업을 즉시 적용합니다.
- `next-maintenance` - 리소스의 다음번 유지 관리 기간 중에 유지 관리 작업을 적용합니다.
- `undo-opt-in` - 기존의 `next-maintenance` 옵트인 요청을 모두 취소합니다.

유형: 문자열

필수 항목 여부: 예

ResourceIdentifier

대기 중인 유지 관리 작업이 적용되는 리소스의 Amazon 리소스 이름(ARN)입니다.

유형: 문자열

필수 항목 여부: 예

응답 요소

서비스에서 다음 요소를 반환합니다.

ResourcePendingMaintenanceActions

ApplyPendingMaintenanceAction의 출력을 나타냅니다.

타입: ResourcePendingMaintenanceActions 객체

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하세요.

InvalidDBClusterStateFault

클러스터가 유효한 상태가 아닙니다.

HTTP 상태 코드: 400

InvalidDBInstanceState

지정된 인스턴스가 사용 가능한 상태가 아닙니다.

HTTP 상태 코드: 400

ResourceNotFoundFault

지정된 리소스 ID를 찾을 수 없습니다.

HTTP 상태 코드: 404

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK

- AWS Python용 SDK
- AWS SDK for Ruby V3

CopyDBClusterParameterGroup

서비스: Amazon DocumentDB (with MongoDB compatibility)

지정된 클러스터 파라미터 그룹을 복사합니다.

요청 파라미터

모든 작업에 공통으로 적용되는 파라미터에 대한 자세한 내용은 공통 파라미터를 참조하세요.

SourceDBClusterParameterGroupIdentifier

소스 클러스터 파라미터 그룹의 식별자 또는 Amazon 리소스 이름(ARN)입니다.

제약 조건:

- 유효한 클러스터 파라미터 그룹을 지정해야 합니다.
- 소스 클러스터 파라미터 그룹이 복사본 AWS 리전 과 동일한에 있는 경우 유효한 파라미터 그룹 식별자, 예를 들어 `my-db-cluster-param-group`또는 유효한 ARN을 지정합니다.
- 소스 파라미터 그룹이 복사본 AWS 리전 과 다른에 있는 경우 유효한 클러스터 파라미터 그룹 ARN을 지정합니다. 예: `arn:aws:rds:us-east-1:123456789012:sample-cluster:sample-parameter-group`.

유형: 문자열

필수 항목 여부: 예

TargetDBClusterParameterGroupDescription

복사된 클러스터 파라미터 그룹에 대한 설명입니다.

유형: 문자열

필수 항목 여부: 예

TargetDBClusterParameterGroupIdentifier

복사된 클러스터 파라미터 그룹의 식별자입니다.

제약 조건:

- null이거나, 비워 두거나, 공백을 입력할 수 없습니다.
- 1~255자의 문자, 숫자 또는 하이픈으로 구성되어야 합니다.
- 첫 자는 문자여야 합니다.

- 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다.

예시: `my-cluster-param-group1`

유형: 문자열

필수 항목 여부: 예

Tags.Tag.N

파라미터 그룹에 할당할 태그입니다.

타입: Tag객체 배열

필수: 아니요

응답 요소

서비스에서 반환되는 요소는 다음과 같습니다.

DBClusterParameterGroup

클러스터 파라미터 그룹의 상세 정보입니다.

타입: DBClusterParameterGroup 객체

오류

모든 작업에서 발생하는 일반적인 오류에 대한 자세한 내용은 일반적인 오류 섹션을 참조하세요.

DBParameterGroupAlreadyExists

동일한 이름의 파라미터 그룹이 있습니다.

HTTP 상태 코드: 400

DBParameterGroupNotFound

`DBParameterGroupName`는 기존 파라미터 그룹을 참조하지 않습니다.

HTTP 상태 코드: 404

DBParameterGroupQuotaExceeded

이 요청으로 인해 허용된 파라미터 그룹 수를 초과하게 됩니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

CopyDBClusterSnapshot

서비스: Amazon DocumentDB (with MongoDB compatibility)

클러스터의 스냅샷을 복사합니다.

공유된 수동 클러스터 스냅샷에서 클러스터 스냅샷을 복사하려면 `SourceDBClusterSnapshotIdentifier`가 공유된 클러스터 스냅샷의 Amazon 리소스 이름(ARN)이어야 합니다. 동일한 AWS 리전에서는 암호화 여부와 관계없이 공유 DB 클러스터 스냅샷만 복사할 수 있습니다.

복사 작업이 진행 중일 때 이를 취소하려면 해당 클러스터 스냅샷이 복사 상태에 있는 동안 `TargetDBClusterSnapshotIdentifier`에 의해 식별된 대상 클러스터 스냅샷을 삭제합니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 내용은 공통 파라미터를 참조하십시오.

SourceDBClusterSnapshotIdentifier

복사할 클러스터 스냅샷의 식별자입니다. 이 파라미터는 대소문자를 구분하지 않습니다.

제약 조건:

- 사용 가능한 상태에서 유효한 시스템 스냅샷을 지정해야 합니다.
- 원본 스냅샷이 복사본 AWS 리전 과 동일한에 있는 경우 유효한 스냅샷 식별자를 지정합니다.
- 소스 스냅샷이 복사본 AWS 리전 과 다른에 있는 경우 유효한 클러스터 스냅샷 ARN을 지정합니다.

예시: `my-cluster-snapshot1`

유형: 문자열

필수 항목 여부: 예

TargetDBClusterSnapshotIdentifier

소스 클러스터 스냅샷에서 생성할 새 클러스터 스냅샷의 식별자입니다. 이 파라미터는 대소문자를 구분하지 않습니다.

제약 조건:

- 1~63자의 문자, 숫자 또는 하이픈으로 구성되어야 합니다.

- 첫 자는 문자여야 합니다.
- 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다.

예시: `my-cluster-snapshot2`

유형: 문자열

필수 항목 여부: 예

CopyTags

원본 클러스터 스냅샷의 모든 태그를 대상 클러스터 스냅샷으로 복사하려면 `true`이고, 그렇지 않으면 `false`입니다. 기본값은 `false`입니다.

유형: 부울

필수 항목 여부: 아니요

KmsKeyId

암호화된 클러스터 스냅샷의 AWS KMS 키 ID입니다. 키 ID는 AWS KMS Amazon 리소스 이름(ARN), AWS KMS 키 식별자 또는 AWS KMS 암호화 AWS KMS 키의 키 별칭입니다.

에서 암호화된 클러스터 스냅샷을 복사 AWS 계정하는 경우의 값을 지정`KmsKeyId`하여 새 암호화 AWS KMS 키로 복사본을 암호화할 수 있습니다. 에 값을 지정하지 않으면 클러스터 스냅샷의 `KmsKeyId`사본이 소스 클러스터 스냅샷과 동일한 AWS KMS 키로 암호화됩니다.

다른에서 공유되는 암호화된 클러스터 스냅샷을 복사 AWS 계정하는 경우에 대한 값을 지정해야 합니다`KmsKeyId`.

암호화된 클러스터 스냅샷을 다른에 복사하려면 대상 리전에서 클러스터 스냅샷의 복사본을 암호화하는 데 사용할 AWS KMS 키 ID로 `KmsKeyId`를 AWS 리전설정합니다. AWS KMS 암호화 키는 생성된에 고유 AWS 리전 하며 한 AWS 리전 에서 다른 로 암호화 키를 사용할 수 없습니다 AWS 리전.

암호화되지 않은 클러스터 스냅샷의 복사하고 `KmsKeyId` 파라미터에 대한 값을 지정하려고 시도하면 오류가 반환됩니다.

유형: 문자열

필수 항목 여부: 아니요

PreSignedUrl

복사할 소스 클러스터 스냅샷이 포함된의 API 작업에 대한CopyDBClusterSnapshot 서명 버전 4 서명 요청이 AWS 리전 포함된 URL입니다. 다른 AWS 리전의 클러스터 스냅샷을 복사할 때는 이 PreSignedUrl 파라미터를 사용해야 합니다.

AWS SDK 도구 또는를 사용하는 경우 PreSignedUrl 수동으로 지정하는 대신를 지정할 수 있습니다SourceRegion(또는 --source-region의 AWS CLI경우 AWS CLI). SourceRegion을 지정하면 소스 AWS 리전에서 실행할 수 있는 작업에 대한 유효한 요청인 미리 서명된 URL이 자동으로 생성됩니다.

미리 서명된 URL은 복사할 클러스터 스냅샷이 포함된 소스에서 실행할 수 AWS 리전 있는 CopyDBClusterSnapshot API 작업에 대한 유효한 요청이어야 합니다. 미리 서명된 URL 요청은 다음 파라미터 값을 포함해야 합니다.

- SourceRegion- 복사할 스냅샷이 포함된 지역의 ID.
- SourceDBClusterSnapshotIdentifier – 복사할 암호화된 클러스터 스냅샷의 식별자입니다. 이 식별자는 소스 AWS 리전용 Amazon 리소스 이름(ARN) 형식이어야 합니다. 예를 들어, 암호화된 클러스터 스냅샷을 us-east-1 AWS 리전에서 복사하는 경우, SourceDBClusterSnapshotIdentifier는 다음과 같이 보입니다: arn:aws:rds:us-east-1:12345678012:sample-cluster:sample-cluster-snapshot.
- TargetDBClusterSnapshotIdentifier – 생성할 새 클러스터 스냅샷의 식별자입니다. 이 파라미터는 대/소문자를 구분하지 않습니다.

유형: 문자열

필수 항목 여부: 아니요

Tags.Tag.N

새 클러스터 스냅샷에 할당할 태그입니다.

타입: Tag객체 배열

필수: 아니요

응답 요소

서비스에서 다음 요소를 반환합니다.

DBClusterSnapshot

클러스터 스냅샷에 대한 세부 정보입니다.

타입: DBClusterSnapshot 객체

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하세요.

DBClusterSnapshotAlreadyExistsFault

해당 식별자를 사용하는 클러스터 스냅샷이 이미 있습니다.

HTTP 상태 코드: 400

DBClusterSnapshotNotFoundFault

`DBClusterSnapshotIdentifier`은 기존 클러스터 스냅샷을 참조하지 않습니다.

HTTP 상태 코드: 404

InvalidDBClusterSnapshotStateFault

제공된 값은 유효한 클러스터 스냅샷 상태가 아닙니다.

HTTP 상태 코드: 400

InvalidDBClusterStateFault

클러스터가 유효한 상태가 아닙니다.

HTTP 상태 코드: 400

KMSKeyNotAccessibleFault

AWS KMS 키에 액세스할 때 오류가 발생했습니다.

HTTP 상태 코드: 400

SnapshotQuotaExceeded

요청으로 인해 허용된 스냅샷 수를 초과하게 됩니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go v2용 SDK
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

CreateDBCluster

서비스: Amazon DocumentDB (with MongoDB compatibility)

새 Amazon DocumentDB 클러스터를 생성합니다.

요청 파라미터

모든 작업에 공통으로 적용되는 파라미터에 대한 자세한 내용은 공통 파라미터를 참조하세요.

DBClusterIdentifier

클러스터 식별자입니다. 이 파라미터는 소문자 문자열로 저장됩니다.

제약 조건:

- 1~63자의 문자, 숫자 또는 하이픈으로 구성되어야 합니다.
- 첫 자는 문자여야 합니다.
- 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다.

예시: `my-cluster`

유형: 문자열

필수 항목 여부: 예

Engine

이 클러스터에 사용할 데이터베이스 엔진의 이름입니다.

유효값: `docdb`

유형: 문자열

필수 항목 여부: 예

AvailabilityZones.AvailabilityZone.N

클러스터의 인스턴스를 생성할 수 있는 Amazon EC2 가용 영역의 목록입니다.

유형: 문자열 배열

필수 여부: 아니요

BackupRetentionPeriod

자동 백업이 보관되는 일수입니다. 1 이상의 값을 지정해야 합니다.

기본값: 1

제약 조건:

- 1~35의 값이어야 합니다.

유형: 정수

필수 항목 여부: 아니요

DBClusterParameterGroupName

이 클러스터와 연결할 클러스터 파라미터 그룹의 이름입니다.

유형: 문자열

필수 항목 여부: 아니요

DBSubnetGroupName

이 클러스터와 연결할 서브넷 그룹입니다.

제약: 기존의 `DBSubnetGroup` 이름과 일치해야 합니다. 기본값이 아니어야 합니다.

예시: `mySubnetgroup`

유형: 문자열

필수 항목 여부: 아니요

DeletionProtection

이 클러스터를 삭제할 수 있는지 없는지를 지정합니다. `DeletionProtection`이 활성화되면 클러스터를 수정하고 `DeletionProtection` 를 비활성화하지 않는 한 클러스터를 삭제할 수 없습니다. `DeletionProtection`은 클러스터가 실수로 삭제되지 않도록 보호합니다.

유형: 부울

필수 항목 여부: 아니요

EnableCloudwatchLogsExports.member.N

Amazon CloudWatch Logs로 내보내기에 대해 활성화해야 하는 로그 유형의 목록입니다. 감사 로그 또는 프로파일러 로그를 활성화할 수 있습니다. 자세한 내용은 Amazon DocumentDB 이벤트 감사 및 Amazon DocumentDB 작업 프로파일링을 참조하십시오.

유형: 문자열 배열

필수 여부: 아니요

EngineVersion

사용할 데이터베이스 엔진의 버전 번호입니다. `--engine-version`은 기본적으로 최신 주요 엔진 버전으로 설정됩니다. 프로덕션 워크로드의 경우 이 파라미터를 의도한 주요 엔진 버전으로 명시적으로 선언하는 것이 좋습니다.

유형: 문자열

필수 항목 여부: 아니요

GlobalClusterIdentifier

새 글로벌 클러스터의 클러스터 식별자입니다.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 255.

패턴: `[A-Za-z][0-9A-Za-z-:._]*`

필수 여부: 아니요

KmsKeyId

암호화된 클러스터의 AWS KMS 키 식별자입니다.

AWS KMS 키 식별자는 AWS KMS 암호화 키의 Amazon 리소스 이름(ARN)입니다. 새 클러스터를 암호화하는 데 사용되는 AWS KMS 암호화 키를 소유 AWS 계정 한 동일한를 사용하여 클러스터를 생성하는 경우 AWS KMS 암호화 AWS KMS 키에 ARN 대신 키 별칭을 사용할 수 있습니다.

`KmsKeyId`에 암호화 키가 지정되어 있지 않은 경우:

- `StorageEncrypted` 파라미터가 `true`인 경우 Amazon DocumentDB는 기본 암호화 키를 사용합니다.

AWS KMS 는에 대한 기본 암호화 키를 생성합니다 AWS 계정. AWS 계정 에는 각각 다른 기본 암호화 키가 있습니다 AWS 리전.

유형: 문자열

필수 항목 여부: 아니요

ManageMasterUserPassword

Amazon Web Services Secrets Manager를 사용하여 마스터 사용자 암호를 관리할지 여부를 지정합니다.

제약:이 `MasterUserPassword` 지정된 경우 Amazon Web Services Secrets Manager를 사용하여 마스터 사용자 암호를 관리할 수 없습니다.

유형: 부울

필수 항목 여부: 아니요

MasterUsername

클러스터의 마스터 사용자 이름입니다.

제약 조건:

- 1~63자의 문자 또는 숫자여야 합니다.
- 첫 번째 자리는 문자여야 합니다.
- 선택한 데이터베이스 엔진의 예약어는 사용할 수 없습니다.

유형: 문자열

필수 항목 여부: 아니요

MasterUserPassword

마스터 데이터베이스 사용자의 암호입니다. 이 암호에는 슬래시(/), 큰따옴표(") 또는 '앳' 기호(@)를 제외한 인쇄 가능 ASCII 문자가 포함될 수 있습니다.

제약: 8~100자여야 합니다.

유형: 문자열

필수 항목 여부: 아니요

MasterUserSecretKmsKeyId

Amazon Web Services Secrets Manager에서 자동으로 생성되고 관리되는 보안 암호를 암호화하기 위한 Amazon Web Services KMS 키 식별자입니다. 이 설정은 DB 클러스터용 Amazon Web Services Secrets Manager의 Amazon DocumentDB에서 마스터 사용자 암호를 관리하는 경우에만 유효합니다.

Amazon Web Services KMS 키 식별자는 KMS 키의 키 ARN, 키 ID, 별칭 ARN 또는 별칭 이름입니다. 다른 Amazon Web Services 계정에서 KMS 키를 사용하려면 키 ARN 또는 별칭 ARN을 지정합니다.

`MasterUserSecretKmsKeyId`를 지정하지 않으면 `aws/secretsmanager` KMS 키가 보안 암호를 암호화하는 데 사용됩니다. 보안 암호가 다른 Amazon Web Services 계정에 있는 경우 `aws/secretsmanager` KMS 키를 사용하여 보안 암호를 암호화할 수 없으며 고객 관리형 KMS 키를 사용해야 합니다.

Amazon Web Services 계정에 대한 기본 KMS 키가 있습니다. Amazon Web Services 계정에는 Amazon Web Services 리전마다 다른 기본 KMS 키가 있습니다.

유형: 문자열

필수 항목 여부: 아니요

Port

클러스터의 인스턴스가 연결을 허용하는 포트 번호입니다.

유형: 정수

필수 항목 여부: 아니요

PreferredBackupWindow

`BackupRetentionPeriod` 파라미터를 사용하여 자동 백업을 활성화한 경우, 자동 백업이 생성되는 일일 시간 범위입니다.

기본값은 각각에 대해 8시간 블록 시간 중에서 임의로 선택한 30분 기간입니다 AWS 리전.

제약 조건:

- `hh24:mi-hh24:mi` 형식이어야 합니다.
- 협정 세계시(UTC)여야 합니다.
- 원하는 유지 관리 기간과 충돌하지 않아야 합니다.
- 30분 이상이어야 합니다.

유형: 문자열

필수 항목 여부: 아니요

PreferredMaintenanceWindow

시스템 유지 관리를 실행할 수 있는 주 단위 기간(UTC, 협정 세계시)입니다.

형식: `ddd:hh24:mi-ddd:hh24:mi`

기본값은 각에 대해 8시간 블록 시간 중에서 임의로 선택한 30분 기간으로 AWS 리전, 요일에 무작위로 발생합니다.

유효한 요일: 월, 화, 수, 목, 금, 토, 일

제약 조건: 최소 30분의 기간.

유형: 문자열

필수 항목 여부: 아니요

PreSignedUrl

현재 지원되지 않습니다.

유형: 문자열

필수 항목 여부: 아니요

StorageEncrypted

클러스터의 암호화 여부를 지정합니다.

유형: 부울

필수 항목 여부: 아니요

StorageType

DB 클러스터와 연결할 스토리지 유형입니다.

Amazon DocumentDB 클러스터의 스토리지 유형에 관한 자세한 내용은 Amazon DocumentDB 개발자 안내서의 클러스터 스토리지 구성을 참조하세요.

스토리지 유형에 유효한 값 - `standard | iopt1`

기본값은 `standard` 입니다.

> **Note**
>
> 스토리지 유형이 `iopt1`로 설정된 DocumentDB DB 클러스터를 생성하면 스토리지 유형이 응답에 반환됩니다. 스토리지 유형을 `standard`로 설정하면 스토리지 유형이 반환되지 않습니다.

유형: 문자열

필수 항목 여부: 아니요

Tags.Tag.N

클러스터에 할당할 태그입니다.

타입: Tag객체 배열

필수: 아니요

VpcSecurityGroupIds.VpcSecurityGroupId.N

이 클러스터와 연결할 EC2 VPC 보안 그룹 목록입니다.

유형: 문자열 배열

필수 여부: 아니요

응답 요소

서비스에서 다음 요소를 반환합니다.

DBCluster

클러스터에 관한 자세한 정보입니다.

타입: DBCluster 객체

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하세요.

DBClusterAlreadyExistsFault

해당 식별자를 사용하는 클러스터가 이미 있습니다.

HTTP 상태 코드: 400

DBClusterNotFoundFault

`DBClusterIdentifier`는 기존 클러스터를 참조하지 않습니다.

HTTP 상태 코드: 404

DBClusterParameterGroupNotFound

`DBClusterParameterGroupName`는 기존 클러스터 파라미터 그룹을 참조하지 않습니다.

HTTP 상태 코드: 404

DBClusterQuotaExceededFault

클러스터의 최대 허용 할당량에 도달했기 때문에 클러스터를 생성할 수 없습니다.

HTTP 상태 코드: 403

DBInstanceNotFound

`DBInstanceIdentifier`는 기존 인스턴스를 참조하지 않습니다.

HTTP 상태 코드: 404

DBSubnetGroupDoesNotCoverEnoughAZs

가용 영역이 하나뿐인 경우를 제외하고, 서브넷 그룹의 서브넷은 최소한 두 개의 가용 영역을 포함해야 합니다.

HTTP 상태 코드: 400

DBSubnetGroupNotFoundFault

`DBSubnetGroupName`은 기존 서브넷 그룹을 참조하지 않습니다.

HTTP 상태 코드: 404

GlobalClusterNotFoundFault

`GlobalClusterIdentifier`는 기존 글로벌 클러스터를 참조하지 않습니다.

HTTP 상태 코드: 404

InsufficientStorageClusterCapacity

현재의 작업에 사용할 스토리지가 부족합니다. 사용 가능한 스토리지가 더 많은 다른 가용 영역을 사용하도록 서브넷 그룹을 업데이트하여 이 오류를 해결할 수 있습니다.

HTTP 상태 코드: 400

InvalidDBClusterStateFault

클러스터가 유효한 상태가 아닙니다.

HTTP 상태 코드: 400

InvalidDBInstanceState

지정된 인스턴스가 사용 가능한 상태가 아닙니다.

HTTP 상태 코드: 400

InvalidDBSubnetGroupStateFault

서브넷 그룹이 사용 중이므로 삭제할 수 없습니다.

HTTP 상태 코드: 400

InvalidGlobalClusterStateFault

클러스터가 이 상태인 동안에는 요청된 작업을 수행할 수 없습니다.

HTTP 상태 코드: 400

InvalidSubnet

요청한 서브넷이 올바르지 않거나 공통 Virtual Private Cloud(VPC)에 모두 포함되지 않은 여러 서브넷이 요청되었습니다.

HTTP 상태 코드: 400

InvalidVPCNetworkStateFault

서브넷 그룹이 생성된 후에는 변경 사항으로 인해 모든 가용 영역에 적용되지 않습니다.

HTTP 상태 코드: 400

KMSKeyNotAccessibleFault

AWS KMS 키에 액세스할 때 오류가 발생했습니다.

HTTP 상태 코드: 400

StorageQuotaExceeded

요청으로 인해 모든 인스턴스에서 사용 가능한 스토리지 허용량을 초과하게 됩니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

CreateDBClusterParameterGroup

서비스: Amazon DocumentDB (with MongoDB compatibility)

새 클러스터 파라미터 그룹을 생성합니다.

클러스터 파라미터 그룹의 파라미터는 클러스터의 모든 인스턴스에 적용됩니다.

처음에 클러스터 파라미터 그룹은 클러스터의 인스턴스에서 사용하는 데이터베이스 엔진의 기본 파라미터로 생성됩니다. Amazon DocumentDB에서는 `default.docdb3.6` 클러스터 파라미터 그룹을 직접 수정할 수 없습니다. Amazon DocumentDB 클러스터가 기본 클러스터 파라미터 그룹을 사용하고 있고 이 그룹의 값을 수정하려면 먼저 새 파라미터 그룹을 생성하거나 기존 파라미터 그룹을 복사하여 수정한 다음 수정된 파라미터 그룹을 클러스터에 적용해야 합니다. 새 클러스터 파라미터 그룹 및 연결된 설정을 적용하려면 장애 조치 없이 클러스터의 인스턴스를 재부팅해야 합니다. 자세한 내용은 Amazon DocumentDB 클러스터 파라미터 그룹 수정을 참조하세요.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 내용은 공통 파라미터를 참조하십시오.

DBClusterParameterGroupName

클러스터 파라미터 그룹의 이름입니다.

제약 조건:

- 기존의 `DBClusterParameterGroup` 이름과 일치할 수 없습니다.

> **Note**
> 이 값은 소문자 문자열로 저장됩니다.

유형: 문자열

필수 항목 여부: 예

DBParameterGroupFamily

클러스터 파라미터 그룹 패밀리의 이름입니다.

유형: 문자열

필수 항목 여부: 예

Description

클러스터 파라미터 그룹에 대한 설명입니다.

유형: 문자열

필수 항목 여부: 예

Tags.Tag.N

클러스터 파라미터 그룹에 할당할 태그입니다.

타입: Tag객체 배열

필수: 아니요

응답 요소

서비스에서 반환되는 요소는 다음과 같습니다.

DBClusterParameterGroup

클러스터 파라미터 그룹의 상세 정보입니다.

타입: DBClusterParameterGroup 객체

오류

모든 작업에서 발생하는 일반적인 오류에 대한 자세한 내용은 일반적인 오류 섹션을 참조하세요.

DBParameterGroupAlreadyExists

동일한 이름의 파라미터 그룹이 있습니다.

HTTP 상태 코드: 400

DBParameterGroupQuotaExceeded

이 요청으로 인해 허용된 파라미터 그룹 수를 초과하게 됩니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

CreateDBClusterSnapshot

서비스: Amazon DocumentDB (with MongoDB compatibility)

클러스터의 스냅샷을 생성합니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 내용은 공통 파라미터를 참조하십시오.

DBClusterIdentifier

스냅샷을 만들 클러스터의 식별자입니다. 이 파라미터는 대소문자를 구분하지 않습니다.

제약 조건:

- 기존 DBCluster의 식별자와 일치해야 합니다.

예시: `my-cluster`

유형: 문자열

필수 항목 여부: 예

DBClusterSnapshotIdentifier

클러스터 스냅샷의 식별자입니다. 이 파라미터는 소문자 문자열로 저장됩니다.

제약 조건:

- 1~63자의 문자, 숫자 또는 하이픈으로 구성되어야 합니다.
- 첫 자는 문자여야 합니다.
- 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다.

예시: `my-cluster-snapshot1`

유형: 문자열

필수 항목 여부: 예

Tags.Tag.N

새 클러스터 스냅샷에 할당할 태그입니다.

타입: Tag객체 배열

필수: 아니요

응답 요소

서비스에서 다음 요소를 반환합니다.

DBClusterSnapshot

클러스터 스냅샷에 대한 세부 정보입니다.

타입: DBClusterSnapshot 객체

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하세요.

DBClusterNotFoundFault

`DBClusterIdentifier`은 기존 클러스터를 참조하지 않습니다.

HTTP 상태 코드: 404

DBClusterSnapshotAlreadyExistsFault

해당 식별자를 사용하는 클러스터 스냅샷이 이미 있습니다.

HTTP 상태 코드: 400

InvalidDBClusterSnapshotStateFault

제공된 값은 유효한 클러스터 스냅샷 상태가 아닙니다.

HTTP 상태 코드: 400

InvalidDBClusterStateFault

클러스터가 유효한 상태가 아닙니다.

HTTP 상태 코드: 400

SnapshotQuotaExceeded

요청으로 인해 허용된 스냅샷 수를 초과하게 됩니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

CreateDBInstance

서비스: Amazon DocumentDB (with MongoDB compatibility)

새 인스턴스를 생성합니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 내용은 공통 파라미터를 참조하십시오.

DBClusterIdentifier

인스턴스가 속하게 될 클러스터의 식별자입니다.

유형: 문자열

필수 항목 여부: 예

DBInstanceClass

인스턴스의 컴퓨팅 및 메모리 용량입니다(예: `db.r5.large`).

유형: 문자열

필수 항목 여부: 예

DBInstanceIdentifier

인스턴스 식별자입니다. 이 파라미터는 소문자 문자열로 저장됩니다.

제약 조건:

- 1~63자의 문자, 숫자 또는 하이픈으로 구성되어야 합니다.
- 첫 자는 문자여야 합니다.
- 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다.

예시: `mydbinstance`

유형: 문자열

필수 항목 여부: 예

Engine

이 인스턴스에서 사용되는 데이터베이스 엔진의 이름입니다.

유효한 값: `docdb`

유형: 문자열

필수 항목 여부: 예

AutoMinorVersionUpgrade

이 파라미터는 Amazon DocumentDB에는 적용되지 않습니다. Amazon DocumentDB는 값 세트에 관계없이 마이너 버전 업그레이드를 수행하지 않습니다.

기본값: `false`

유형: 부울

필수 항목 여부: 아니요

AvailabilityZone

인스턴스가 생성된 Amazon EC2 가용 영역입니다.

기본값: 엔드포인트의 시스템에서 선택한 무작위 가용 영역입니다 AWS 리전.

예시: `us-east-1d`

유형: 문자열

필수 항목 여부: 아니요

CACertificateIdentifier

DB 인스턴스의 서버 인증서에 사용할 CA 인증서 식별자입니다.

자세한 내용은 Amazon DocumentDB 개발자 안내서의 Amazon DocumentDB TLS 인증서 업데이트 및 전송 중 데이터 암호화를 참조하십시오.

유형: 문자열

필수 항목 여부: 아니요

CopyTagsToSnapshot

태그를 DB 인스턴스에서 DB 인스턴스의 스냅샷으로 복사할지 여부를 나타내는 값입니다. 태그는 기본적으로 복사되지 않습니다.

유형: 부울

필수 항목 여부: 아니요

EnablePerformanceInsights

DB 인스턴스에 Performance Insights를 활성화할지 여부를 나타내는 값입니다. 자세한 내용은 Amazon 성능 개선 도우미 사용을 참조하세요.

유형: 부울

필수 항목 여부: 아니요

PerformanceInsightsKMSKeyId

성능 개선 도우미 데이터의 암호화를 위한 AWS KMS 키 식별자입니다.

AWS KMS 키 식별자는 KMS 키의 키 ARN, 키 ID, 별칭 ARN 또는 별칭 이름입니다.

PerformanceInsightsKMSKeyId에 대한 값을 지정하지 않으면 Amazon DocumentDB에서는 기본 KMS 키를 사용합니다. Amazon Web Services 계정에 대한 기본 KMS 키가 있습니다. Amazon Web Services 계정에는 Amazon Web Services 리전마다 다른 기본 KMS 키가 있습니다.

유형: 문자열

필수 항목 여부: 아니요

PreferredMaintenanceWindow

시스템 유지 관리를 실행할 수 있는 주 단위의 시간 범위(UTC, 협정 세계시)입니다.

형식: `ddd:hh24:mi-ddd:hh24:mi`

기본값은 각에 대해 8시간 블록 시간 중에서 임의로 선택되는 30분 기간으로 AWS 리전, 요일에 무작위로 발생합니다.

유효한 요일: 월, 화, 수, 목, 금, 토, 일

제약 조건: 최소 30분의 기간.

유형: 문자열

필수 항목 여부: 아니요

PromotionTier

기존 기본 인스턴스에 결함이 발생한 후 Amazon DocumentDB 복제본을 기본 인스턴스로 승격할 순서를 지정하는 값

기본값: 1

유효한 값: 0~15

유형: 정수

필수 항목 여부: 아니요

Tags.Tag.N

인스턴스에 할당할 태그입니다. 하나의 인스턴스에 최대 10개의 태그를 할당할 수 있습니다.

타입: Tag객체 배열

필수: 아니요

응답 요소

서비스에서 다음 요소를 반환합니다.

DBInstance

인스턴스에 대한 자세한 정보입니다.

타입: DBInstance 객체

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하세요.

AuthorizationNotFound

지정한 CIDR IP 또는 Amazon EC2 보안 그룹에 대해 지정한 보안 그룹에 대한 권한이 없습니다.

Amazon DocumentDB는 또한 IAM을 사용하여 사용자를 대신하여 필요한 작업을 수행할 권한이 없을 수도 있습니다.

HTTP 상태 코드: 404

DBClusterNotFoundFault

`DBClusterIdentifier`은 기존 클러스터를 참조하지 않습니다.

HTTP 상태 코드: 404

DBInstanceAlreadyExists

해당 식별자를 사용하는 인스턴스가 이미 있습니다.

HTTP 상태 코드: 400

DBParameterGroupNotFound

`DBParameterGroupName`는 기존 파라미터 그룹을 참조하지 않습니다.

HTTP 상태 코드: 404

DBSecurityGroupNotFound

`DBSecurityGroupName`는 기존 보안 그룹을 참조하지 않습니다.

HTTP 상태 코드: 404

DBSubnetGroupDoesNotCoverEnoughAZs

가용 영역이 하나뿐인 경우를 제외하고, 서브넷 그룹의 서브넷은 최소한 두 개의 가용 영역을 포함해야 합니다.

HTTP 상태 코드: 400

DBSubnetGroupNotFoundFault

`DBSubnetGroupName`은 기존 서브넷 그룹을 참조하지 않습니다.

HTTP 상태 코드: 404

InstanceQuotaExceeded

요청으로 인해 허용된 인스턴스 수를 초과하게 됩니다.

HTTP 상태 코드: 400

InsufficientDBInstanceCapacity

지정한 인스턴스 클래스를 지정한 가용성 영역에서 사용할 수 없습니다.

HTTP 상태 코드: 400

InvalidDBClusterStateFault

클러스터가 유효한 상태가 아닙니다.

HTTP 상태 코드: 400

InvalidSubnet

요청한 서브넷이 올바르지 않거나 공통 Virtual Private Cloud(VPC)에 모두 포함되지 않은 여러 서브넷이 요청되었습니다.

HTTP 상태 코드: 400

InvalidVPCNetworkStateFault

서브넷 그룹이 생성된 후에는 변경 사항으로 인해 모든 가용 영역에 적용되지 않습니다.

HTTP 상태 코드: 400

KMSKeyNotAccessibleFault

AWS KMS 키에 액세스할 때 오류가 발생했습니다.

HTTP 상태 코드: 400

StorageQuotaExceeded

요청으로 인해 모든 인스턴스에서 사용 가능한 스토리지 허용량을 초과하게 됩니다.

HTTP 상태 코드: 400

StorageTypeNotSupported

지정된 `StorageType`을 이 DB 인스턴스와 연결할 수 없습니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go v2용 SDK
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin

- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

CreateDBSubnetGroup

서비스: Amazon DocumentDB (with MongoDB compatibility)

새 서브넷 그룹을 생성합니다. 서브넷 그룹은 AWS 리전의 두 개 이상의 가용성 영역에 하나 이상의 서브넷을 포함해야 합니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 내용은 공통 파라미터를 참조하십시오.

DBSubnetGroupDescription

서브넷 그룹에 대한 설명입니다.

유형: 문자열

필수 항목 여부: 예

DBSubnetGroupName

서브넷 그룹의 이름입니다. 이 값은 소문자 문자열로 저장됩니다.

제약: 255자 이하의 문자, 숫자, 마침표, 밑줄, 공백 또는 하이픈만 포함해야 합니다. 기본값이 아니어야 합니다.

예시: `mySubnetgroup`

유형: 문자열

필수 항목 여부: 예

SubnetIds.SubnetIdentifier.N

서브넷 그룹의 Amazon EC2 서브넷 ID입니다.

유형: 문자열 배열

필수 여부: 예

Tags.Tag.N

서브넷 그룹에 할당할 태그입니다.

타입: Tag객체 배열

필수: 아니요

응답 요소

서비스에서 다음 요소를 반환합니다.

DBSubnetGroup

서브넷 그룹에 대한 자세한 정보.

타입: DBSubnetGroup 객체

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하십시오.

DBSubnetGroupAlreadyExists

`DBSubnetGroupName`은 기존 서브넷 그룹에서 이미 사용하고 있습니다.

HTTP 상태 코드: 400

DBSubnetGroupDoesNotCoverEnoughAZs

가용 영역이 하나뿐인 경우를 제외하고 서브넷 그룹의 서브넷은 두 개 이상의 가용 영역을 포함해야 합니다.

HTTP 상태 코드: 400

DBSubnetGroupQuotaExceeded

요청 시 허용되는 서브넷 그룹의 수를 초과하게 됩니다.

HTTP 상태 코드: 400

DBSubnetQuotaExceededFault

요청 시 서브넷 그룹의 허용된 서브넷의 수를 초과하게 됩니다.

HTTP 상태 코드: 400

InvalidSubnet

요청한 서브넷이 올바르지 않거나 공통 Virtual Private Cloud(VPC)에 모두 포함되지 않은 여러 서브넷이 요청되었습니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

CreateEventSubscription

서비스: Amazon DocumentDB (with MongoDB compatibility)

Amazon DocumentDB 이벤트 알림 구독 생성 이 작업을 하려면 Amazon DocumentDB 콘솔, Amazon SNS 콘솔 또는 Amazon SNS API를 사용해 생성한 주제 Amazon 리소스 이름(ARN)이 필요합니다. Amazon SNS를 통해 ARN을 받으려면 Amazon SNS에서 주제를 생성하고 그 주제를 구독해야 합니다. ARN이 Amazon SNS 콘솔에 표시됩니다.

알림 메시지를 받고 싶은 소스 유형과 이벤트를 트리거링하는 소스(SourceType)를 지정할 수 있습니다. 또한 이벤트를 트리거하는 Amazon DocumentDB 소스 목록(SourceIds)을 제공하고 알림을 받고자 하는 이벤트의 이벤트 카테고리 목록(EventCategories)을 제공할 수 있습니다. 예를 들어, `SourceType = db-instance`, `SourceIds = mydbinstance1, mydbinstance2` 및 `EventCategories = Availability, Backup`을 지정할 수 있습니다.

`SourceType`및 `SourceIds` (예: `SourceType = db-instance` 및`SourceIdentifier = myDBInstance1`)를 모두 지정하면 지정된 소스의 모든 `db-instance` 이벤트에 대한 알림을 받게 됩니다. 하지만 `SourceType`을 지정하지만 `SourceIdentifier`를 지정하지 않으면 모든 Amazon DocumentDB 소스에 대한 소스 유형의 이벤트만 알림 메시지로 받게 됩니다. `SourceType` 또는 `SourceIdentifier`를 둘 다 지정하지 않으면 고객 계정에 속하는 모든 Amazon DocumentDB 소스에서 발생하는 이벤트의 알림을 받게 됩니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 자세한 내용은 공통 파라미터를 참조하십시오.

SnsTopicArn

이벤트 알림을 위해 생성한 SNS 주제의 Amazon 리소스 이름(ARN)입니다. Amazon SNS는 주제를 생성하고 구독할 때 ARN을 생성합니다.

유형: 문자열

필수 항목 여부: 예

SubscriptionName

구독의 이름.

제약: 이름은 255자 미만이어야 합니다.

유형: 문자열

필수 항목 여부: 예

Enabled

부울 값입니다. 구독을 활성화하려면 `true`로 설정하고, 구독을 생성만 하고 활성화하지 않으려면 `false`로 설정합니다.

유형: 부울

필수 항목 여부: 아니요

EventCategories.EventCategory.N

구독할 `SourceType`의 이벤트 범주 목록입니다.

유형: 문자열 배열

필수 여부: 아니요

SourceIds.SourceId.N

반환되는 이벤트에 대한 이벤트 소스 식별자 목록입니다. 지정하지 않으면 모든 소스가 응답에 포함됩니다. 식별자는 문자로 시작해야 하고, ASCII 문자, 숫자 및 하이픈만 포함할 수 있으며, 하이픈으로 끝나거나 하이픈을 연속으로 두 개 사용하면 안 됩니다.

제약 조건:

- `SourceIds`이 제공된 경우 `SourceType`도 제공해야 합니다.
- 소스 유형이 인스턴스라면 `DBInstanceIdentifier`을 제공해야 합니다.
- 소스 유형이 보안 그룹이라면 `DBSecurityGroupName`을 제공해야 합니다.
- 소스 유형이 파라미터 그룹이라면 `DBParameterGroupName`을 제공해야 합니다.
- 소스 유형이 스냅샷이라면 `DBSnapshotIdentifier`을 제공해야 합니다.

유형: 문자열 배열

필수 여부: 아니요

SourceType

이벤트가 발생하는 소스의 유형입니다. 예를 들어, 인스턴스에서 생성되는 이벤트에 대한 알림을 받으려면 이 파라미터를 `db-instance`로 설정합니다. 이 값을 지정하지 않으면 모든 이벤트가 반환됩니다.

유효한 값: `db-instance`, `db-cluster`, `db-parameter-group`, `db-security-group`, `db-cluster-snapshot`

유형: 문자열

필수 항목 여부: 아니요

Tags.Tag.N

이벤트 구독에 할당할 태그입니다.

타입: Tag객체 배열

필수: 아니요

응답 요소

서비스에서 다음 요소를 반환합니다.

EventSubscription

구독한 이벤트에 대한 세부 정보.

타입: EventSubscription 객체

오류

모든 작업에서 공통적으로 발생하는 오류에 대한 자세한 내용은 일반적인 오류 단원을 참조하십시오.

EventSubscriptionQuotaExceeded

최대 이벤트 구독 수에 도달했습니다.

HTTP 상태 코드: 400

SNSInvalidTopic

Amazon SNS에서 지정된 주제에 문제가 있다고 응답했습니다.

HTTP 상태 코드: 400

SNSNoAuthorization

SNS 주제 Amazon 리소스 이름(ARN) 에 게시할 권한이 없습니다.

HTTP 상태 코드: 400

SNSTopicArnNotFound

SNS 주제 Amazon 리소스 이름(ARN)이 존재하지 않습니다

HTTP 상태 코드: 404

SourceNotFound

요청한 소스를 찾을 수 없습니다.

HTTP 상태 코드: 404

SubscriptionAlreadyExist

입력한 구독 이름이 이미 존재합니다.

HTTP 상태 코드: 400

SubscriptionCategoryNotFound

제공된 범주가 존재하지 않습니다.

HTTP 상태 코드: 404

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

CreateGlobalCluster

서비스: Amazon DocumentDB (with MongoDB compatibility)

여러 개의 AWS 리전에 걸쳐 있는 Amazon DocumentDB 글로벌 클러스터를 생성합니다. 글로벌 클러스터에는 읽기-쓰기 기능이 있는 기본 클러스터 1개와 읽기 전용 보조 클러스터까지 포함되어 있습니다. 글로벌 클러스터는 워크로드 성능에 영향을 주지 않는 전용 인프라를 사용하여 지연 시간이 1초 미만인 리전 간에 스토리지 기반의 고속 복제를 사용합니다.

처음에 비어 있는 글로벌 클러스터를 생성한 다음 기본 클러스터와 보조 클러스터를 추가할 수 있습니다. 또는 생성 작업 중에 기존 클러스터를 지정할 수 있으며, 그러면 이 클러스터가 글로벌 클러스터의 기본 클러스터가 됩니다.

> **Note**
>
> 이 작업은 Amazon DocumentDB 클러스터에만 적용됩니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 내용은 공통 파라미터를 참조하십시오.

GlobalClusterIdentifier

새 글로벌 클러스터의 클러스터 식별자.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 255.

패턴: `[A-Za-z][0-9A-Za-z-:._]*`

필수 여부: 예

DatabaseName

영숫자 문자 최대 64자로 된 데이터베이스의 이름입니다. 이름을 입력하지 않으면 Amazon DocumentDB는 생성 중인 글로벌 클러스터에 데이터베이스를 생성하지 않습니다.

유형: 문자열

필수 항목 여부: 아니요

DeletionProtection

새 글로벌 클러스터에 대한 삭제 보호 설정. 삭제 방지 기능이 활성화되면 글로벌 클러스터가 삭제될 수 없습니다.

유형: 부울

필수 항목 여부: 아니요

Engine

이 클러스터에 사용할 데이터베이스 엔진의 이름.

유형: 문자열

필수 항목 여부: 아니요

EngineVersion

글로벌 클러스터의 엔진 버전.

유형: 문자열

필수 항목 여부: 아니요

SourceDBClusterIdentifier

글로벌 클러스터의 기본 클러스터로 사용할 Amazon 리소스 이름(ARN). 이 파라미터는 선택 사항입니다.

유형: 문자열

필수 항목 여부: 아니요

StorageEncrypted

새 글로벌 클러스터에 대한 스토리지 암호화 설정입니다.

유형: 부울

필수 항목 여부: 아니요

응답 요소

서비스에서 다음 요소를 반환합니다.

GlobalCluster

Amazon DocumentDB 글로벌 클러스터를 나타내는 데이터 유형.

타입: GlobalCluster 객체

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하세요.

DBClusterNotFoundFault

`DBClusterIdentifier`은 기존 클러스터를 참조하지 않습니다.

HTTP 상태 코드: 404

GlobalClusterAlreadyExistsFault

`GlobalClusterIdentifier`이 이미 존재합니다. 새 글로벌 클러스터 식별자(고유 이름)를 선택하여 새 글로벌 클러스터를 생성합니다.

HTTP 상태 코드: 400

GlobalClusterQuotaExceededFault

이 계정의 글로벌 클러스터 수가 이미 허용된 최대치에 도달했습니다.

HTTP 상태 코드: 400

InvalidDBClusterStateFault

클러스터가 유효한 상태가 아닙니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2

- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

DeleteDBCluster

서비스: Amazon DocumentDB (with MongoDB compatibility)

이전에 프로비저닝된 클러스터를 삭제합니다. 클러스터를 삭제하면 해당 클러스터의 자동 백업도 모두 삭제되며 복구할 수 없습니다. 지정한 클러스터의 수동 DB 클러스터 스냅샷은 삭제되지 않습니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 내용은 공통 파라미터를 참조하십시오.

DBClusterIdentifier

삭제할 클러스터의 클러스터 식별자. 이 파라미터는 대/소문자를 구분하지 않습니다.

제약 조건:

- 기존 `DBClusterIdentifier`과 일치해야 합니다.

유형: 문자열

필수 항목 여부: 예

FinalDBSnapshotIdentifier

`SkipFinalSnapshot`이 `false`로 설정된 경우 생성되는 새 클러스터 스냅샷의 클러스터 스냅샷 식별자.

> **Note**
>
> 이 파라미터를 지정하고 또한 `SkipFinalShapshot` 파라미터를 `true`로 설정하면 오류가 발생합니다.

제약 조건:

- 1~255개의 문자, 숫자 또는 하이픈을 사용해야 합니다.
- 첫 자는 문자여야 합니다.
- 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다.

유형: 문자열

필수 항목 여부: 아니요

SkipFinalSnapshot

클러스터를 삭제하기 전에 최종 클러스터 스냅샷을 생성할지 여부를 결정합니다. `true`로 지정된 경우 클러스터 스냅샷이 생성되지 않습니다. `false`로 지정된 경우 DB 클러스터가 삭제되기 전에 클러스터 스냅샷이 생성됩니다.

> **Note**
>
> `SkipFinalSnapshot`이 `false`이면 `FinalDBSnapshotIdentifier` 파라미터를 지정해야 합니다.

기본값: `false`

유형: 부울

필수 항목 여부: 아니요

응답 요소

서비스에서 다음 요소를 반환합니다.

DBCluster

클러스터에 관한 자세한 정보입니다.

타입: DBCluster 객체

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하세요.

DBClusterNotFoundFault

`DBClusterIdentifier`은 기존 클러스터를 참조하지 않습니다.

HTTP 상태 코드: 404

DBClusterSnapshotAlreadyExistsFault

해당 식별자를 사용하는 클러스터 스냅샷이 이미 있습니다.

HTTP 상태 코드: 400

InvalidDBClusterSnapshotStateFault

제공된 값은 유효한 클러스터 스냅샷 상태가 아닙니다.

HTTP 상태 코드: 400

InvalidDBClusterStateFault

클러스터가 유효한 상태가 아닙니다.

HTTP 상태 코드: 400

SnapshotQuotaExceeded

요청으로 인해 허용된 스냅샷 수를 초과하게 됩니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

DeleteDBClusterParameterGroup

서비스: Amazon DocumentDB (with MongoDB compatibility)

지정된 클러스터 파라미터 그룹을 삭제합니다. 삭제할 클러스터 파라미터 그룹은 어떤 클러스터와도 연결할 수 없습니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 내용은 공통 파라미터를 참조하십시오.

DBClusterParameterGroupName

클러스터 파라미터 그룹의 이름입니다.

제약 조건:

- 기존 클러스터 파라미터 그룹의 이름이어야 합니다.
- 기본 클러스터 파라미터 그룹은 삭제할 수 없습니다.
- 어떤 클러스터와도 연결할 수 없습니다.

유형: 문자열

필수 항목 여부: 예

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하십시오.

DBParameterGroupNotFound

`DBParameterGroupName`은 기존 파라미터 그룹을 참조하지 않습니다.

HTTP 상태 코드: 404

InvalidDBParameterGroupState

파라미터 그룹이 사용 중이거나 유효하지 않은 상태입니다. 파라미터 그룹을 삭제하려는 경우 파라미터 그룹이 이 상태일 때는 삭제할 수 없습니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

DeleteDBClusterSnapshot

서비스: Amazon DocumentDB (with MongoDB compatibility)

클러스터 스냅샷을 삭제합니다. 스냅샷을 복사 중인 경우, 복사 작업이 종료됩니다.

> **Note**
>
> 클러스터 스냅샷을 삭제하려면 `available` 상태여야 합니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 자세한 내용은 공통 파라미터를 참조하세요.

DBClusterSnapshotIdentifier

삭제할 클러스터 스냅샷의 식별자입니다.

제약: `available` 상태인 기존 클러스터 스냅샷의 이름이어야 합니다.

유형: 문자열

필수 항목 여부: 예

응답 요소

서비스에서 다음 요소를 반환합니다.

DBClusterSnapshot

클러스터 스냅샷에 대한 세부 정보입니다.

타입: DBClusterSnapshot 객체

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하십시오.

DBClusterSnapshotNotFoundFault

`DBClusterSnapshotIdentifier`은 기존 클러스터 스냅샷을 참조하지 않습니다.

HTTP 상태 코드: 404

InvalidDBClusterSnapshotStateFault

제공된 값은 유효한 클러스터 스냅샷 상태가 아닙니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go v2용 SDK
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

DeleteDBInstance

서비스: Amazon DocumentDB (with MongoDB compatibility)

이전에 프로비전된 인스턴스를 삭제합니다.

요청 파라미터

모든 작업에 공통으로 적용되는 파라미터에 대한 자세한 내용은 공통 파라미터를 참조하세요.

DBInstanceIdentifier

삭제할 인스턴스의 인스턴스 식별자입니다. 이 파라미터는 대/소문자를 구분하지 않습니다.

제약 조건:

- 기존 인스턴스의 이름과 일치해야 합니다.

유형: 문자열

필수 항목 여부: 예

응답 요소

서비스에서 다음 요소를 반환합니다.

DBInstance

인스턴스에 대한 자세한 정보입니다.

타입: DBInstance 객체

오류

모든 작업에 공동되는 오류에 대한 자세한 내용은 일반적인 오류을 참조하십시오.

DBInstanceNotFound

`DBInstanceIdentifier`는 기존 인스턴스를 참조하지 않습니다.

HTTP 상태 코드: 404

DBSnapshotAlreadyExists

`DBSnapshotIdentifier`는 기존 스냅샷에서 이미 사용 중입니다.

HTTP 상태 코드: 400

InvalidDBClusterStateFault

클러스터가 유효한 상태가 아닙니다.

HTTP 상태 코드: 400

InvalidDBInstanceState

지정된 인스턴스가 사용 가능한 상태가 아닙니다.

HTTP 상태 코드: 400

SnapshotQuotaExceeded

요청으로 인해 허용된 스냅샷 수를 초과하게 됩니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go v2용 SDK
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

DeleteDBSubnetGroup

서비스: Amazon DocumentDB (with MongoDB compatibility)

서브넷 그룹을 삭제합니다.

> **Note**
> 지정된 데이터베이스 서브넷 그룹은 어떤 DB 인스턴스와도 연결되어 있으면 안 됩니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 내용은 공통 파라미터를 참조하십시오.

DBSubnetGroupName

삭제할 데이터베이스 서브넷 그룹의 이름입니다.

> **Note**
> 기본 서브넷 그룹은 삭제할 수 없습니다.

제약 조건:

기존의 `DBSubnetGroup` 이름과 일치해야 합니다. 기본값이 아니어야 합니다.

예시: `mySubnetgroup`

유형: 문자열

필수 항목 여부: 예

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하십시오.

DBSubnetGroupNotFoundFault

`DBSubnetGroupName`은 기존 서브넷 그룹을 참조하지 않습니다.

HTTP 상태 코드: 404

InvalidDBSubnetGroupStateFault

서브넷 그룹이 사용 중이므로 삭제할 수 없습니다.

HTTP 상태 코드: 400

InvalidDBSubnetStateFault

서브넷이 사용 가능한 상태가 아닙니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

DeleteEventSubscription

서비스: Amazon DocumentDB (with MongoDB compatibility)

Amazon DocumentDB 이벤트 알림 구독을 삭제합니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 내용은 공통 파라미터를 참조하십시오.

SubscriptionName

삭제할 Amazon DocumentDB 이벤트 알림 구독의 이름.

유형: 문자열

필수 항목 여부: 예

응답 요소

서비스에서 다음 요소를 반환합니다.

EventSubscription

구독한 이벤트에 대한 세부 정보.

타입: EventSubscription 객체

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하십시오.

InvalidEventSubscriptionState

다른 사람이 구독을 수정하고 있을 수 있습니다. 몇 초 기다린 후 다시 시도하십시오.

HTTP 상태 코드: 400

SubscriptionNotFound

구독의 이름이 존재하지 않습니다.

HTTP 상태 코드: 404

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

DeleteGlobalCluster

서비스: Amazon DocumentDB (with MongoDB compatibility)

글로벌 클러스터를 삭제합니다. 글로벌 클러스터를 삭제하려면 먼저 기본 및 보조 클러스터를 분리하거나 삭제해야 합니다.

> **Note**
>
> 이 작업은 Amazon DocumentDB 클러스터에만 적용됩니다.

요청 파라미터

모든 작업에 공통으로 적용되는 파라미터에 대한 자세한 내용은 공통 파라미터를 참조하세요.

GlobalClusterIdentifier

삭제되는 글로벌 클러스터의 클러스터 식별자입니다.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 255.

패턴: `[A-Za-z][0-9A-Za-z-:._]*`

필수 여부: 예

응답 요소

서비스에서 다음 요소를 반환합니다.

GlobalCluster

Amazon DocumentDB 글로벌 클러스터를 나타내는 데이터 유형.

타입: GlobalCluster 객체

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하세요.

GlobalClusterNotFoundFault

`GlobalClusterIdentifier`는 기존 글로벌 클러스터를 참조하지 않습니다.

HTTP 상태 코드: 404

InvalidGlobalClusterStateFault

클러스터가 이 상태인 동안에는 요청된 작업을 수행할 수 없습니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

DescribeCertificates

서비스: Amazon DocumentDB (with MongoDB compatibility)

이에 대해 Amazon DocumentDB에서 제공한 인증 기관 (CA) 인증서 목록을 반환합니다. AWS 계정

요청 파라미터

모든 작업에 공통으로 적용되는 파라미터에 대한 자세한 내용은 공통 파라미터를 참조하세요.

CertificateIdentifier

사용자 제공 인증서 식별자. 이 파라미터를 지정하면 지정된 인증서에 대한 정보만 반환됩니다. 이 파라미터가 누락되면 최대 `MaxRecords` 인증서 목록이 반환됩니다. 이 파라미터는 대소문자를 구분하지 않습니다.

제약 조건

- 기존 것과 일치해야 합니다. `CertificateIdentifier`

유형: 문자열

필수 항목 여부: 아니요

Filters.Filter.N

현재 지원되지 않는 파라미터입니다.

타입: Filter객체 배열

필수: 아니요

Marker

이전의 `DescribeCertificates` 요청에서 제공된 선택적 페이지 매김 토큰입니다. 이 파라미터를 지정한 경우, 마커 이후부터 `MaxRecords`에 지정된 값까지의 레코드만 응답에 포함됩니다.

유형: 문자열

필수 항목 여부: 아니요

MaxRecords

응답에 포함되는 최대 레코드 수입니다. 지정된 `MaxRecords` 값보다 레코드 수가 많으면 마커라고 부르는 페이지 매김 토큰을 응답에 포함시켜 나머지 결과를 검색할 수 있도록 합니다.

기본값: 100

제약 조건:

- 최소: 20
- 최대: 100

유형: 정수

필수 항목 여부: 아니요

응답 요소

서비스에서 반환되는 요소는 다음과 같습니다.

Certificates.Certificate.N

이 AWS 계정에 대한 인증서 목록.

타입: Certificate객체 배열

Marker

검색된 레코드 수가 다음보다 큰 경우 제공되는 선택적 페이지 매김 토큰. `MaxRecords` 이 파라미터가 지정되면 마커가 목록의 다음 레코드를 지정합니다. 다음 호출 `DescribeCertificates` 결과에 `Marker` 의 값을 포함하면 인증서의 다음 페이지로 이동합니다.

유형: 문자열

오류

모든 작업에서 발생하는 일반적인 오류에 대한 자세한 내용은 일반적인 오류 섹션을 참조하십시오.

CertificateNotFound

`CertificateIdentifier`는 기존 인증서를 참조하지 않습니다.

HTTP 상태 코드: 404

참고

언어별 AWS SDKs

- AWS Command Line Interface

- AWS SDK for .NET
- AWS SDK for C++
- AWS Go v2용 SDK
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

DescribeDBClusterParameterGroups

서비스: Amazon DocumentDB (with MongoDB compatibility)

DBClusterParameterGroup 설명 목록을 반환합니다. DBClusterParameterGroupName 파라미터가 지정된 경우, 목록에는 지정된 클러스터 파라미터 그룹에 대한 설명만 포함됩니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 내용은 공통 파라미터를 참조하십시오.

DBClusterParameterGroupName

세부 정보를 반환할 특정 클러스터 파라미터 그룹의 이름.

제약 조건:

- 제공된 경우 기존 DBClusterParameterGroup의 이름과 일치해야 합니다.

유형: 문자열

필수 항목 여부: 아니요

Filters.Filter.N

현재 지원되지 않는 파라미터입니다.

타입: Filter객체 배열

필수: 아니요

Marker

이전의 요청에서 제공된 선택적 페이지 매김 토큰입니다. 이 파라미터를 지정한 경우, 마커 이후부터 MaxRecords에 지정된 값까지의 레코드만 응답에 포함됩니다.

유형: 문자열

필수 항목 여부: 아니요

MaxRecords

응답에 포함되는 최대 레코드 수입니다. 지정된 MaxRecords 값보다 레코드 수가 많으면 페이지 매김 토큰(마커)을 응답에 포함시켜 나머지 결과를 검색할 수 있도록 합니다.

기본값: 100

제약: 최소 20, 최대 100입니다.

유형: 정수

필수 항목 여부: 아니요

응답 요소

서비스에서 반환되는 요소는 다음과 같습니다.

DBClusterParameterGroups.DBClusterParameterGroup.N

클러스터 파라미터 그룹의 목록.

타입: DBClusterParameterGroup객체 배열

Marker

이전의 요청에서 제공된 선택적 페이지 매김 토큰입니다. 이 파라미터를 지정한 경우, 마커 이후부터 `MaxRecords`에 지정된 값까지의 레코드만 응답에 포함됩니다.

유형: 문자열

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하십시오.

DBParameterGroupNotFound

`DBParameterGroupName`은 기존 파라미터 그룹을 참조하지 않습니다.

HTTP 상태 코드: 404

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go v2용 SDK

- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

DescribeDBClusterParameters

서비스: Amazon DocumentDB (with MongoDB compatibility)

특정 클러스터 파라미터 그룹에 대한 세부 파라미터 목록을 반환합니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 내용은 공통 파라미터를 참조하십시오.

DBClusterParameterGroupName

파라미터의 세부 정보를 반환할 특정 클러스터 파라미터 그룹의 이름.

제약 조건:

- 제공된 경우 기존 `DBClusterParameterGroup`의 이름과 일치해야 합니다.

유형: 문자열

필수 항목 여부: 예

Filters.Filter.N

현재 지원되지 않는 파라미터입니다.

타입: Filter객체 배열

필수: 아니요

Marker

이전의 요청에서 제공된 선택적 페이지 매김 토큰입니다. 이 파라미터를 지정한 경우, 마커 이후부터 `MaxRecords`에 지정된 값까지의 레코드만 응답에 포함됩니다.

유형: 문자열

필수 항목 여부: 아니요

MaxRecords

응답에 포함되는 최대 레코드 수입니다. 지정된 `MaxRecords` 값보다 레코드 수가 많으면 페이지 매김 토큰(마커)을 응답에 포함시켜 나머지 결과를 검색할 수 있도록 합니다.

기본값: 100

제약: 최소 20, 최대 100입니다.

유형: 정수

필수 항목 여부: 아니요

Source

특정 소스의 파라미터만 반환됨을 나타내는 값입니다. 파라미터 소스는 `engine`, `service` 또는 `customer`가 될 수 있습니다.

유형: 문자열

필수 항목 여부: 아니요

응답 요소

서비스에서 반환되는 요소는 다음과 같습니다.

Marker

이전의 요청에서 제공된 선택적 페이지 매김 토큰입니다. 이 파라미터를 지정한 경우, 마커 이후부터 `MaxRecords`에 지정된 값까지의 레코드만 응답에 포함됩니다.

유형: 문자열

Parameters.Parameter.N

클러스터 파라미터 그룹의 파라미터 목록을 제공합니다.

타입: Parameter 객체 배열

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하십시오.

DBParameterGroupNotFound

`DBParameterGroupName`은 기존 파라미터 그룹을 참조하지 않습니다.

HTTP 상태 코드: 404

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

DescribeDBClusters

서비스: Amazon DocumentDB (with MongoDB compatibility)

프로비저닝된 Amazon DocumentDB 클러스터에 대한 정보를 반환합니다. 이 API 작업은 페이지 매김을 지원합니다. 클러스터 및 인스턴스 수명 주기 관리와 같은 특정 관리 기능의 경우 Amazon DocumentDB는 Amazon RDS 및 Amazon Neptune과 공유되는 운영 기술을 활용합니다. `filterName=engine,Values=docdb` 필터 파라미터를 사용하면 Amazon DocumentDB 클러스터만 반환할 수 있습니다.

요청 파라미터

모든 작업에 공통으로 적용되는 파라미터에 대한 내용은 공통 파라미터를 참조하십시오.

DBClusterIdentifier

사용자가 제공한 클러스터 식별자. 이 파라미터를 지정한 경우 바로 그 클러스터에서 온 정보만 반환됩니다. 이 파라미터는 대/소문자를 구분하지 않습니다.

제약 조건:

- 제공된 경우 기존 `DBClusterIdentifier`과 일치해야 합니다.

유형: 문자열

필수 항목 여부: 아니요

Filters.Filter.N

설명할 클러스터를 하나 이상 지정하는 필터.

지원되는 필터:

- `db-cluster-id` - 클러스터 식별자 및 클러스터의 Amazon 리소스 이름(ARN)을 사용할 수 있습니다. 결과 목록에는 이러한 ARN으로 식별된 클러스터에 대한 정보만 포함됩니다.

타입: Filter객체 배열

필수: 아니요

Marker

이전의 요청에서 제공된 선택적 페이지 매김 토큰입니다. 이 파라미터를 지정한 경우, 마커 이후부터 `MaxRecords`에 지정된 값까지의 레코드만 응답에 포함됩니다.

유형: 문자열

필수 항목 여부: 아니요

MaxRecords

응답에 포함되는 최대 레코드 수입니다. 지정된 `MaxRecords` 값보다 레코드 수가 많으면 페이지 매김 토큰(마커)을 응답에 포함시켜 나머지 결과를 검색할 수 있도록 합니다.

기본값: 100

제약: 최소 20, 최대 100입니다.

유형: 정수

필수 항목 여부: 아니요

응답 요소

서비스에서 반환되는 요소는 다음과 같습니다.

DBClusters.DBCluster.N

클러스터의 목록.

타입: DBCluster객체 배열

Marker

이전의 요청에서 제공된 선택적 페이지 매김 토큰입니다. 이 파라미터를 지정한 경우, 마커 이후부터 `MaxRecords`에 지정된 값까지의 레코드만 응답에 포함됩니다.

유형: 문자열

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하세요.

DBClusterNotFoundFault

`DBClusterIdentifier`은 기존 클러스터를 참조하지 않습니다.

HTTP 상태 코드: 404

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go v2용 SDK
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

DescribeDBClusterSnapshotAttributes

서비스: Amazon DocumentDB (with MongoDB compatibility)

수동 DB 클러스터 스냅샷의 클러스터 스냅샷 속성 이름 및 값 목록을 반환합니다.

스냅샷을 다른와 공유하면 AWS 계정는 수동 클러스터 스냅샷을 복사하거나 복원할 권한이 AWS 계정 있는의 `restore` 속성과 IDs 목록을 `DescribeDBClusterSnapshotAttributes` 반환합니다. `all`이 `restore` 특성에 대한 값 목록에 포함된 경우 수동 클러스터 스냅샷은 공용이며 모든 AWS 계정에 의해 복사 또는 복원될 수 있습니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 내용은 공통 파라미터를 참조하십시오.

DBClusterSnapshotIdentifier

특성을 설명하는 클러스터 스냅샷의 식별자입니다.

유형: 문자열

필수 항목 여부: 예

응답 요소

서비스에서 다음 요소를 반환합니다.

DBClusterSnapshotAttributesResult

클러스터 스냅샷과 관련된 특성에 대한 상세 정보입니다.

타입: DBClusterSnapshotAttributesResult 객체

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하십시오.

DBClusterSnapshotNotFoundFault

`DBClusterSnapshotIdentifier`은 기존 클러스터 스냅샷을 참조하지 않습니다.

HTTP 상태 코드: 404

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

DescribeDBClusterSnapshots

서비스: Amazon DocumentDB (with MongoDB compatibility)

클러스터 스냅샷에 대한 정보를 반환합니다. API 작업은 페이지 매김을 지원합니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 내용은 공통 파라미터를 참조하십시오.

DBClusterIdentifier

클러스터 스냅샷의 목록을 검색할 클러스터의 ID. 이 파라미터는 `DBClusterSnapshotIdentifier` 파라미터와 함께 사용할 수 없습니다. 이 파라미터는 대소문자를 구분하지 않습니다.

제약 조건:

- 제공된 경우 기존 `DBCluster`의 식별자와 일치해야 합니다.

유형: 문자열

필수 항목 여부: 아니요

DBClusterSnapshotIdentifier

설명할 특정 클러스터 스냅샷 식별자. 이 파라미터는 `DBClusterIdentifier` 파라미터와 함께 사용할 수 없습니다. 이 값은 소문자 문자열로 저장됩니다.

제약 조건:

- 제공된 경우 기존 `DBClusterSnapshot`의 식별자와 일치해야 합니다.
- 이 식별자가 자동화된 스냅샷의 식별자인 경우, `SnapshotType` 파라미터도 지정해야 합니다.

유형: 문자열

필수 항목 여부: 아니요

Filters.Filter.N

현재 지원되지 않는 파라미터입니다.

타입: Filter객체 배열

필수: 아니요

IncludePublic

퍼블릭인 수동 클러스터 스냅샷을 포함`true`하려면를 로 설정하고 AWS 계정, 그렇지 않으면에서 복사하거나 복원할 수 있습니다`false`. 기본값은 `false`입니다.

유형: 부울

필수 항목 여부: 아니요

IncludeShared

복사 또는 복원 권한이 AWS 계정 부여된 다른의 공유 수동 클러스터 스냅샷 AWS 계정 을 포함`true`하려면 로 설정하고, 그렇지 않으면 로 설정합니다`false`. 기본값은 `false`입니다.

유형: 부울

필수 항목 여부: 아니요

Marker

이전의 요청에서 제공된 선택적 페이지 매김 토큰입니다. 이 파라미터를 지정한 경우, 마커 이후부터 `MaxRecords`에 지정된 값까지의 레코드만 응답에 포함됩니다.

유형: 문자열

필수 항목 여부: 아니요

MaxRecords

응답에 포함되는 최대 레코드 수입니다. 지정된 `MaxRecords` 값보다 레코드 수가 많으면 페이지 매김 토큰(마커)을 응답에 포함시켜 나머지 결과를 검색할 수 있도록 합니다.

기본값: 100

제약: 최소 20, 최대 100입니다.

유형: 정수

필수 항목 여부: 아니요

SnapshotType

반환되는 클러스터 스냅샷의 유형. 다음 값 중 하나를 지정할 수 있습니다.

- `automated` - Amazon DocumentDB가 AWS 계정를 위해 자동으로 생성한 모든 클러스터 스냅샷을 반환합니다.
- `manual` - AWS 계정를 위해 수동으로 생성한 모든 클러스터 스냅샷을 반환합니다.

- shared - AWS 계정에 공유된 모든 수동 클러스터 스냅샷을 반환합니다.
- public - 퍼블릭으로 표시된 클러스터 스냅샷을 모두 반환합니다.

SnapshotType 값을 지정하지 않으면 자동 및 수동 클러스터 스냅샷이 모두 반환됩니다. IncludeShared 파라미터를 true로 설정하여 이러한 결과에 공유 클러스터 스냅샷을 포함시킬 수 있습니다. IncludePublic 파라미터를 true로 설정하여 이러한 결과에 퍼블릭 클러스터 스냅샷을 포함시킬 수 있습니다.

SnapshotType 값이 manual 또는 automated인 경우 IncludeShared 및 IncludePublic 파라미터는 적용되지 않습니다. SnapshotType이 shared로 설정된 경우 IncludePublic 파라미터는 적용되지 않습니다. SnapshotType이 public로 설정된 경우 IncludeShared 파라미터는 적용되지 않습니다.

유형: 문자열

필수 항목 여부: 아니요

응답 요소

서비스에서 반환되는 요소는 다음과 같습니다.

DBClusterSnapshots.DBClusterSnapshot.N

클러스터 스냅샷의 목록을 제공합니다.

타입: DBClusterSnapshot객체 배열

Marker

이전의 요청에서 제공된 선택적 페이지 매김 토큰입니다. 이 파라미터를 지정한 경우, 마커 이후부터 MaxRecords에 지정된 값까지의 레코드만 응답에 포함됩니다.

유형: 문자열

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하십시오.

DBClusterSnapshotNotFoundFault

DBClusterSnapshotIdentifier은 기존 클러스터 스냅샷을 참조하지 않습니다.

HTTP 상태 코드: 404

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

DescribeDBEngineVersions

서비스: Amazon DocumentDB (with MongoDB compatibility)

사용 가능한 엔진의 목록을 반환합니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 자세한 내용은 공통 파라미터를 참조하십시오.

DBParameterGroupFamily

세부 정보를 반환할 특정 파라미터 그룹 패밀리의 이름입니다.

제약 조건:

- 제공된 경우 기존 `DBParameterGroupFamily`과 일치해야 합니다.

유형: 문자열

필수 항목 여부: 아니요

DefaultOnly

지정된 엔진의 기본 버전 또는 지정된 엔진과 메이저 버전의 조합만 반환됨을 나타냅니다.

유형: 부울

필수 항목 여부: 아니요

Engine

반환할 데이터베이스 엔진입니다.

유형: 문자열

필수 항목 여부: 아니요

EngineVersion

반환할 데이터베이스 버전입니다.

예시: `3.6.0`

유형: 문자열

필수 항목 여부: 아니요

Filters.Filter.N

현재 지원되지 않는 파라미터입니다.

타입: Filter객체 배열

필수: 아니요

ListSupportedCharacterSets

이 파라미터를 지정했으며 요청한 엔진이 CreateDBInstance에 대해 CharacterSetName 파라미터를 지원하는 경우, 엔진 버전별로 지원되는 문자 세트 목록이 응답에 포함됩니다.

유형: 부울

필수 항목 여부: 아니요

ListSupportedTimezones

이 파라미터를 지정했으며 요청한 엔진이 CreateDBInstance에 대해 TimeZone 파라미터를 지원하는 경우, 엔진 버전별로 지원되는 시간대 목록이 응답에 포함됩니다.

유형: 부울

필수 항목 여부: 아니요

Marker

이전의 요청에서 제공된 선택적 페이지 매김 토큰입니다. 이 파라미터를 지정한 경우, 마커 이후부터 MaxRecords에 지정된 값까지의 레코드만 응답에 포함됩니다.

유형: 문자열

필수 항목 여부: 아니요

MaxRecords

응답에 포함되는 최대 레코드 수입니다. 지정된 MaxRecords 값보다 레코드 수가 많으면 페이지 매김 토큰(마커)을 응답에 포함시켜 나머지 결과를 검색할 수 있도록 합니다.

기본값: 100

제약: 최소 20, 최대 100입니다.

유형: 정수

필수 항목 여부: 아니요

응답 요소

서비스에서 반환되는 요소는 다음과 같습니다.

DBEngineVersions.DBEngineVersion.N

하나 이상의 엔진 버전 세부 정보.

타입: DBEngineVersion객체 배열

Marker

이전의 요청에서 제공된 선택적 페이지 매김 토큰입니다. 이 파라미터를 지정한 경우, 마커 이후부터 `MaxRecords`에 지정된 값까지의 레코드만 응답에 포함됩니다.

유형: 문자열

오류

모든 작업에서 발생하는 일반적인 오류에 대한 자세한 내용은 일반적인 오류를 참조하십시오.

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

DescribeDBInstances

서비스: Amazon DocumentDB (with MongoDB compatibility)

프로비저닝된 Amazon DocumentDB 인스턴스에 대한 정보를 반환합니다. 이 API는 페이지 매김을 지원합니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 내용은 공통 파라미터를 참조하십시오.

DBInstanceIdentifier

사용자가 제공한 인스턴스 식별자. 이 파라미터를 지정한 경우, 바로 그 인스턴스에서 온 정보만 반환됩니다. 이 파라미터는 대/소문자를 구분하지 않습니다.

제약 조건:

- 제공된 경우 기존 `DBInstance`의 식별자와 일치해야 합니다.

유형: 문자열

필수 항목 여부: 아니요

Filters.Filter.N

설명할 인스턴스를 하나 이상 지정하는 필터입니다.

지원되는 필터:

- `db-cluster-id` - 클러스터 식별자 및 클러스터 Amazon 리소스 이름(ARN)을 허용합니다. 결과 목록에는 이러한 ARN으로 식별된 클러스터와 연결되어 있는 인스턴스에 관한 정보만 포함됩니다.
- `db-instance-id` - 인스턴스 식별자 및 인스턴스 ARN을 허용합니다. 결과 목록에는 이러한 ARN으로 식별된 인스턴스에 대한 정보만 포함됩니다.

타입: Filter객체 배열

필수: 아니요

Marker

이전의 요청에서 제공된 선택적 페이지 매김 토큰입니다. 이 파라미터를 지정한 경우, 마커 이후부터 `MaxRecords`에 지정된 값까지의 레코드만 응답에 포함됩니다.

유형: 문자열

필수 항목 여부: 아니요

MaxRecords

응답에 포함되는 최대 레코드 수입니다. 지정된 MaxRecords 값보다 레코드 수가 많으면 페이지 매김 토큰(마커)을 응답에 포함시켜 나머지 결과를 검색할 수 있도록 합니다.

기본값: 100

제약: 최소 20, 최대 100입니다.

유형: 정수

필수 항목 여부: 아니요

응답 요소

서비스에서 반환되는 요소는 다음과 같습니다.

DBInstances.DBInstance.N

하나 이상의 인스턴스 세부 정보.

타입: DBInstance객체 배열

Marker

이전의 요청에서 제공된 선택적 페이지 매김 토큰입니다. 이 파라미터를 지정한 경우, 마커 이후부터 MaxRecords에 지정된 값까지의 레코드만 응답에 포함됩니다.

유형: 문자열

오류

모든 작업에 공동되는 오류에 대한 자세한 내용은 일반적인 오류을 참조하십시오.

DBInstanceNotFound

DBInstanceIdentifier는 기존 인스턴스를 참조하지 않습니다.

HTTP 상태 코드: 404

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

DescribeDBSubnetGroups

서비스: Amazon DocumentDB (with MongoDB compatibility)

DBSubnetGroup 설명 목록을 반환합니다. DBSubnetGroupName이 지정된 경우, 이 목록에는 지정된 DBSubnetGroup에 대한 설명만 포함됩니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 내용은 공통 파라미터를 참조하십시오.

DBSubnetGroupName

세부 정보를 반환할 서브넷 그룹의 이름입니다.

유형: 문자열

필수 항목 여부: 아니요

Filters.Filter.N

현재 지원되지 않는 파라미터입니다.

타입: Filter객체 배열

필수: 아니요

Marker

이전의 요청에서 제공된 선택적 페이지 매김 토큰입니다. 이 파라미터를 지정한 경우, 마커 이후부터 MaxRecords에 지정된 값까지의 레코드만 응답에 포함됩니다.

유형: 문자열

필수 항목 여부: 아니요

MaxRecords

응답에 포함되는 최대 레코드 수입니다. 지정된 MaxRecords 값보다 레코드 수가 많으면 페이지 매김 토큰(마커)을 응답에 포함시켜 나머지 결과를 검색할 수 있도록 합니다.

기본값: 100

제약: 최소 20, 최대 100입니다.

유형: 정수

필수 항목 여부: 아니요

응답 요소

서비스에서 반환되는 요소는 다음과 같습니다.

DBSubnetGroups.DBSubnetGroup.N

하나 이상의 서브넷 그룹에 대한 자세한 정보.

타입: DBSubnetGroup객체 배열

Marker

이전의 요청에서 제공된 선택적 페이지 매김 토큰입니다. 이 파라미터를 지정한 경우, 마커 이후부터 `MaxRecords`에 지정된 값까지의 레코드만 응답에 포함됩니다.

유형: 문자열

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하십시오.

DBSubnetGroupNotFoundFault

`DBSubnetGroupName`은 기존 서브넷 그룹을 참조하지 않습니다.

HTTP 상태 코드: 404

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin

- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

DescribeEngineDefaultClusterParameters

서비스: Amazon DocumentDB (with MongoDB compatibility)

클러스터 데이터베이스 엔진에 대한 기본 엔진 및 시스템 파라미터 정보를 반환합니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 자세한 내용은 범용 파라미터를 참조하십시오.

DBParameterGroupFamily

엔진 파라미터 정보를 반환할 클러스터 파라미터 그룹 패밀리의 이름입니다.

유형: 문자열

필수 항목 여부: 예

Filters.Filter.N

현재 지원되지 않는 파라미터입니다.

타입: Filter객체 배열

필수: 아니요

Marker

이전의 요청에서 제공된 선택적 페이지 매김 토큰입니다. 이 파라미터를 지정한 경우, 마커 이후부터 `MaxRecords`에 지정된 값까지의 레코드만 응답에 포함됩니다.

유형: 문자열

필수 항목 여부: 아니요

MaxRecords

응답에 포함되는 최대 레코드 수입니다. 지정된 `MaxRecords` 값보다 레코드 수가 많으면 페이지 매김 토큰(마커)을 응답에 포함시켜 나머지 결과를 검색할 수 있도록 합니다.

기본값: 100

제약: 최소 20, 최대 100입니다.

유형: 정수

필수 항목 여부: 아니요

응답 요소

서비스에서 반환되는 요소는 다음과 같습니다.

EngineDefaults

성공한 `DescribeEngineDefaultClusterParameters` 작업의 간접 호출 결과가 포함되어 있습니다.

타입: EngineDefaults 객체

오류

모든 작업에서 발생하는 일반적인 오류에 대한 자세한 내용은 일반적인 오류를 참조하십시오.

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

DescribeEventCategories

서비스: Amazon DocumentDB (with MongoDB compatibility)

모든 이벤트 소스 유형 또는 지정된 경우 지정된 소스 유형에 대한 범주 목록을 표시합니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 내용은 범용 파라미터를 참조하십시오.

Filters.Filter.N

현재 지원되지 않는 파라미터입니다.

타입: Filter객체 배열

필수: 아니요

SourceType

이벤트가 발생하는 소스의 유형입니다.

유효한 값: `db-instance`, `db-parameter-group`, `db-security-group`

유형: 문자열

필수 항목 여부: 아니요

응답 요소

서비스에서 반환되는 요소는 다음과 같습니다.

EventCategoriesMapList.EventCategoriesMap.N

이벤트 카테고리 맵의 목록입니다.

타입: EventCategoriesMap 객체 배열

오류

모든 작업에서 발생하는 일반적인 오류에 대한 자세한 내용은 일반적인 오류를 참조하십시오.

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

DescribeEvents

서비스: Amazon DocumentDB (with MongoDB compatibility)

지난 14일 동안의 인스턴스, 보안 그룹, 스냅샷 및 DB 파라미터 그룹과 관련된 이벤트를 반환합니다. 이름을 파라미터로 제공하여 특정 DB 인스턴스, 보안 그룹, 스냅샷 또는 파라미터 그룹과 관련된 이벤트를 얻을 수 있습니다. 기본적으로 지난 1시간 동안의 이벤트가 반환됩니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 관한 정보는 공통 파라미터를 참조하십시오.

Duration

이벤트를 검색할 시간(분)입니다.

기본값: 60

유형: 정수

필수 항목 여부: 아니요

EndTime

이벤트를 검색할 기간의 종료 시점을 ISO 8601 형식으로 지정합니다.

예: 2009-07-08T18:00Z

유형: 타임스탬프

필수 여부: 아니요

EventCategories.EventCategory.N

이벤트 알림 구독에서 알림을 트리거하는 이벤트 범주의 목록입니다.

유형: 문자열 배열

필수 여부: 아니요

Filters.Filter.N

현재 지원되지 않는 파라미터입니다.

타입: Filter객체 배열

필수: 아니요

Marker

이전의 요청에서 제공된 선택적 페이지 매김 토큰입니다. 이 파라미터를 지정한 경우, 마커 이후부터 MaxRecords에 지정된 값까지의 레코드만 응답에 포함됩니다.

유형: 문자열

필수 항목 여부: 아니요

MaxRecords

응답에 포함되는 최대 레코드 수입니다. 지정된 MaxRecords 값보다 레코드 수가 많으면 페이지 매김 토큰(마커)을 응답에 포함시켜 나머지 결과를 검색할 수 있도록 합니다.

기본값: 100

제약: 최소 20, 최대 100입니다.

유형: 정수

필수 항목 여부: 아니요

SourceIdentifier

반환되는 이벤트에 대한 이벤트 소스의 식별자입니다. 지정하지 않으면 모든 소스가 응답에 포함됩니다.

제약 조건:

- SourceIdentifier이 제공된 경우, SourceType도 함께 제공해야 합니다.
- 소스 유형이 DBInstance이라면 DBInstanceIdentifier을 입력해야 합니다.
- 소스 유형이 DBSecurityGroup이라면 DBSecurityGroupName을 입력해야 합니다.
- 소스 유형이 DBParameterGroup이라면 DBParameterGroupName을 입력해야 합니다.
- 소스 유형이 DBSnapshot이라면 DBSnapshotIdentifier을 입력해야 합니다.
- 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다.

유형: 문자열

필수 항목 여부: 아니요

SourceType

이벤트를 검색할 이벤트 소스입니다. 값을 지정하지 않으면 모든 이벤트가 반환됩니다.

타입: 문자열

유효 값: `db-instance | db-parameter-group | db-security-group | db-snapshot | db-cluster | db-cluster-snapshot`

필수 여부: 아니요

StartTime

이벤트를 검색할 기간의 시작 시점을 ISO 8601 형식으로 지정합니다.

예: 2009-07-08T18:00Z

유형: 타임스탬프

필수 여부: 아니요

응답 요소

서비스에서 반환되는 요소는 다음과 같습니다.

Events.Event.N

하나 이상의 이벤트 세부 정보.

타입: Event객체 배열

Marker

이전의 요청에서 제공된 선택적 페이지 매김 토큰입니다. 이 파라미터를 지정한 경우, 마커 이후부터 `MaxRecords`에 지정된 값까지의 레코드만 응답에 포함됩니다.

유형: 문자열

오류

모든 작업에서 발생하는 일반적인 오류에 대한 자세한 내용은 일반적인 오류를 참조하십시오.

참고

언어별 AWS SDKs

- AWS Command Line Interface

- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

DescribeEventSubscriptions

서비스: Amazon DocumentDB (with MongoDB compatibility)

고객 계정의 모든 구독 설명을 나열합니다. 구독에 대한 설명에는 SubscriptionName, SNSTopicARN, CustomerID, SourceType, SourceID, CreationTime, Status가 포함됩니다.

SubscriptionName을 지정하면 해당 구독의 설명이 나열됩니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 내용은 범용 파라미터를 참조하십시오.

Filters.Filter.N

현재 지원되지 않는 파라미터입니다.

타입: Filter객체 배열

필수: 아니요

Marker

이전의 요청에서 제공된 선택적 페이지 매김 토큰입니다. 이 파라미터를 지정한 경우, 마커 이후부터 MaxRecords에 지정된 값까지의 레코드만 응답에 포함됩니다.

유형: 문자열

필수 항목 여부: 아니요

MaxRecords

응답에 포함되는 최대 레코드 수입니다. 지정된 MaxRecords 값보다 레코드 수가 많으면 페이지 매김 토큰(마커)을 응답에 포함시켜 나머지 결과를 검색할 수 있도록 합니다.

기본값: 100

제약: 최소 20, 최대 100입니다.

유형: 정수

필수 항목 여부: 아니요

SubscriptionName

설명할 Amazon DocumentDB 이벤트 알림 구독의 이름.

유형: 문자열

필수 항목 여부: 아니요

응답 요소

서비스에서 반환되는 요소는 다음과 같습니다.

EventSubscriptionsList.EventSubscription.N

이벤트 구독의 목록

타입: EventSubscription객체 배열

Marker

이전의 요청에서 제공된 선택적 페이지 매김 토큰입니다. 이 파라미터를 지정한 경우, 마커 이후부터 `MaxRecords`에 지정된 값까지의 레코드만 응답에 포함됩니다.

유형: 문자열

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하세요.

SubscriptionNotFound

구독의 이름이 존재하지 않습니다.

HTTP 상태 코드: 404

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK

- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

DescribeGlobalClusters

서비스: Amazon DocumentDB (with MongoDB compatibility)

Amazon DocumentDB 글로벌 클러스터에 대한 정보를 반환합니다. 이 API는 페이지 매김을 지원합니다.

> **Note**
>
> 이 작업은 Amazon DocumentDB 클러스터에만 적용됩니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 내용은 범용 파라미터를 참조하십시오.

Filters.Filter.N

설명할 글로벌 DB 클러스터를 하나 이상 지정하는 필터입니다.

지원되는 필터: `db-cluster-id`는 클러스터 식별자 및 클러스터 Amazon 리소스 이름(ARN)을 허용합니다. 결과 목록에는 이러한 ARN으로 식별된 클러스터에 대한 정보만 포함됩니다.

타입: Filter객체 배열

필수: 아니요

GlobalClusterIdentifier

사용자가 제공한 클러스터 식별자입니다. 이 파라미터를 지정한 경우, 바로 그 클러스터에서 온 정보만 반환됩니다. 이 파라미터는 대/소문자를 구분하지 않습니다.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 255.

패턴: `[A-Za-z][0-9A-Za-z-:._]*`

필수 여부: 아니요

Marker

이전의 `DescribeGlobalClusters` 요청에서 제공된 선택적 페이지 매김 토큰입니다. 이 파라미터를 지정한 경우, 마커 이후부터 `MaxRecords`에 지정된 값까지의 레코드만 응답에 포함됩니다.

유형: 문자열

필수 항목 여부: 아니요

MaxRecords

응답에 포함되는 최대 레코드 수입니다. 지정된 `MaxRecords` 값보다 레코드 수가 많으면 나머지 결과를 검색할 수 있도록 마커라는 페이지 매김 토큰이 응답에 포함됩니다.

유형: 정수

필수 항목 여부: 아니요

응답 요소

서비스에서 반환되는 요소는 다음과 같습니다.

GlobalClusters.GlobalClusterMember.N

타입: GlobalCluster객체 배열

Marker

유형: 문자열

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하세요.

GlobalClusterNotFoundFault

`GlobalClusterIdentifier`는 기존 글로벌 클러스터를 참조하지 않습니다.

HTTP 상태 코드: 404

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET

- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

DescribeOrderableDBInstanceOptions

서비스: Amazon DocumentDB (with MongoDB compatibility)

지정된 엔진에 대해 명령 가능한 인스턴스 옵션의 목록을 반환합니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 내용은 공통 파라미터를 참조하십시오.

Engine

인스턴스 옵션을 검색할 엔진의 이름입니다.

유형: 문자열

필수 항목 여부: 예

DBInstanceClass

인스턴스 클래스 필터의 값입니다. 지정된 인스턴스 클래스에 맞고 사용 가능한 제품만 표시되게 하려면 이 파라미터를 지정하십시오.

유형: 문자열

필수 항목 여부: 아니요

EngineVersion

엔진 버전 필터의 값입니다. 지정된 엔진 버전에 맞고 사용 가능한 제품만 표시되게 하려면 이 파라미터를 지정하십시오.

유형: 문자열

필수 항목 여부: 아니요

Filters.Filter.N

현재 지원되지 않는 파라미터입니다.

타입: Filter객체 배열

필수: 아니요

LicenseModel

라이선스 모델 필터의 값입니다. 지정된 라이선스 모델에 맞고 사용 가능한 제품만 표시되게 하려면 이 파라미터를 지정하십시오.

유형: 문자열

필수 항목 여부: 아니요

Marker

이전의 요청에서 제공된 선택적 페이지 매김 토큰입니다. 이 파라미터를 지정한 경우, 마커 이후부터 `MaxRecords`에 지정된 값까지의 레코드만 응답에 포함됩니다.

유형: 문자열

필수 항목 여부: 아니요

MaxRecords

응답에 포함되는 최대 레코드 수입니다. 지정된 `MaxRecords` 값보다 레코드 수가 많으면 페이지 매김 토큰(마커)을 응답에 포함시켜 나머지 결과를 검색할 수 있도록 합니다.

기본값: 100

제약: 최소 20, 최대 100입니다.

유형: 정수

필수 항목 여부: 아니요

Vpc

Virtual Private Cloud(VPC) 필터 값입니다. 사용 가능한 VPC 또는 비VPC 제품만 표시되게 하려면 이 파라미터를 지정하십시오.

유형: 부울

필수 항목 여부: 아니요

응답 요소

서비스에서 반환되는 요소는 다음과 같습니다.

Marker

이전의 요청에서 제공된 선택적 페이지 매김 토큰입니다. 이 파라미터를 지정한 경우, 마커 이후부터 `MaxRecords`에 지정된 값까지의 레코드만 응답에 포함됩니다.

유형: 문자열

OrderableDBInstanceOptions.OrderableDBInstanceOption.N

주문 가능한 특정 인스턴스에 사용할 수 있는 옵션

타입: OrderableDBInstanceOption 객체 배열

오류

모든 작업에서 발생하는 일반적인 오류에 대한 자세한 내용은 일반적인 오류를 참조하십시오.

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

DescribePendingMaintenanceActions

서비스: Amazon DocumentDB (with MongoDB compatibility)

대기 중인 유지 관리 작업이 하나 이상 있는 리소스(예: 인스턴스)의 목록을 반환합니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 내용은 범용 파라미터를 참조하십시오.

Filters.Filter.N

대기 중인 유지 관리 작업을 반환할 리소스를 하나 이상 지정하는 필터입니다.

지원되는 필터:

- `db-cluster-id` - 클러스터 식별자 및 클러스터 Amazon 리소스 이름(ARN)을 허용합니다. 결과 목록에는 이러한 ARN으로 식별된 클러스터에 대해 대기 중인 유지 관리 작업만 포함됩니다.
- `db-instance-id` - 인스턴스 식별자 및 인스턴스 ARN을 허용합니다. 결과 목록에는 이러한 ARN으로 식별된 DB 인스턴스에 대해 대기 중인 유지 관리 작업만 포함됩니다.

타입: Filter객체 배열

필수: 아니요

Marker

이전의 요청에서 제공된 선택적 페이지 매김 토큰입니다. 이 파라미터를 지정한 경우, 마커 이후부터 `MaxRecords`에 지정된 값까지의 레코드만 응답에 포함됩니다.

유형: 문자열

필수 항목 여부: 아니요

MaxRecords

응답에 포함되는 최대 레코드 수입니다. 지정된 `MaxRecords` 값보다 레코드 수가 많으면 페이지 매김 토큰(마커)을 응답에 포함시켜 나머지 결과를 검색할 수 있도록 합니다.

기본값: 100

제약: 최소 20, 최대 100입니다.

유형: 정수

필수 항목 여부: 아니요

ResourceIdentifier

대기 중인 유지 관리 작업을 반환할 리소스의 ARN입니다.

유형: 문자열

필수 항목 여부: 아니요

응답 요소

서비스에서 반환되는 요소는 다음과 같습니다.

Marker

이전의 요청에서 제공된 선택적 페이지 매김 토큰입니다. 이 파라미터를 지정한 경우, 마커 이후부터 `MaxRecords`에 지정된 값까지의 레코드만 응답에 포함됩니다.

유형: 문자열

PendingMaintenanceActions.ResourcePendingMaintenanceActions.N

적용할 유지 관리 작업.

타입: ResourcePendingMaintenanceActions 객체 배열

오류

모든 작업에서 공통적인 오류에 대한 자세한 내용은 일반적인 오류을 참조하십시오.

ResourceNotFoundFault

지정된 리소스 ID를 찾을 수 없습니다.

HTTP 상태 코드: 404

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET

- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

FailoverDBCluster

서비스: Amazon DocumentDB (with MongoDB compatibility)

클러스터에 대한 장애 조치를 강제로 실행합니다.

클러스터에 대한 장애 조치에서 클러스터 내 Amazon DocumentDB 복제본 중 하나(읽기 전용 인스턴스)를 기본 인스턴스(클러스터 라이터)로 승격시킵니다.

Amazon DocumentDB는 기본 인스턴스에 장애가 발생할 경우 Amazon DocumentDB 복제본(있는 경우)에 자동으로 장애 조치됩니다. 테스트를 위해 기본 인스턴스의 실패를 시뮬레이션하려는 경우 장애 조치를 강제할 수 있습니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 정보는 범용파라미터를 참조하세요.

DBClusterIdentifier

장애 조치를 강제 실행할 클러스터 식별자입니다. 이 파라미터는 대소문자를 구분하지 않습니다.

제약 조건:

- 기존 `DBCluster`의 식별자와 일치해야 합니다.

유형: 문자열

필수 항목 여부: 아니요

TargetDBInstanceIdentifier

기본 인스턴스로 승격시킬 인스턴스의 이름입니다.

클러스터에서 Amazon DocumentDB 복제본의 인스턴스 식별자를 지정해야 합니다. 예를 들어 `mydbcluster-replica1`입니다.

유형: String

필수 항목 여부: 아니요

응답 요소

서비스에서 다음 요소를 반환합니다.

DBCluster

클러스터에 관한 자세한 정보입니다.

타입: DBCluster 객체

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하세요.

DBClusterNotFoundFault

`DBClusterIdentifier`은 기존 클러스터를 참조하지 않습니다.

HTTP 상태 코드: 404

InvalidDBClusterStateFault

DB 클러스터가 유효한 상태가 아닙니다.

HTTP 상태 코드: 400

InvalidDBInstanceState

지정된 인스턴스가 사용 가능한 상태가 아닙니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK

- AWS Python용 SDK
- AWS SDK for Ruby V3

FailoverGlobalCluster

서비스: Amazon DocumentDB (with MongoDB compatibility)

글로벌 클러스터를 장애 조치할 때 지정된 보조 DB 클러스터를 글로벌 클러스터의 기본 DB 클러스터로 승격합니다.

이 작업을 사용하여 기본 리전의 리전 재해와 같은 계획되지 않은 이벤트에 대응합니다. 장애 조치로 인해 장애 조치 이벤트가 발생하기 전에 선택한 보조 리전에 쓰기 트랜잭션 데이터가 복제되지 않으면 해당 데이터가 손실될 수 있습니다. 하지만 선택한 보조 DB 클러스터의 DB 인스턴스를 기본 라이터 DB 인스턴스로 승격시키는 복구 프로세스는 데이터가 트랜잭션 형태로 일관된 상태를 유지하도록 보장해 줍니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 내용은 공통 파라미터를 참조하십시오.

GlobalClusterIdentifier

이 작업을 적용할 Amazon DocumentDB 글로벌 클러스터의 식별자입니다. 식별자는 클러스터가 생성될 때 사용자가 할당한 고유 키입니다. 즉, 글로벌 클러스터의 이름입니다.

제약 조건:

- 기존 글로벌 클러스터의 식별자와 일치해야 합니다.
- 최소 길이는 1이고, 최대 길이는 255.

패턴: `[A-Za-z][0-9A-Za-z-:._]*`

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 255.

패턴: `[A-Za-z][0-9A-Za-z-:._]*`

필수 여부: 예

TargetDbClusterIdentifier

글로벌 클러스터의 기본으로 승격시키려는 보조 Amazon DocumentDB 클러스터의 식별자입니다. Amazon DocumentDB가 해당 리전에서 클러스터를 찾을 수 있도록 식별자에 Amazon 리소스 이름(ARN)을 사용합니다. AWS

제약 조건:

- 기존 보조 클러스터의 식별자와 일치해야 합니다.
- 최소 길이는 1이고, 최대 길이는 255.

패턴: `[A-Za-z][0-9A-Za-z-:._]*`

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 255.

패턴: `[A-Za-z][0-9A-Za-z-:._]*`

필수 여부: 예

AllowDataLoss

이 글로벌 클러스터 작업에 대한 데이터 손실을 허용할지 여부를 지정합니다. 데이터 손실을 허용하면 글로벌 장애 조치 작업이 트리거됩니다.

`AllowDataLoss`를 지정하지 않으면 글로벌 클러스터 작업이 기본적으로 전환으로 설정됩니다.

제약 조건:

- `Switchover` 파라미터와 함께 지정할 수 없습니다.

유형: 부울

필수 항목 여부: 아니요

Switchover

이 글로벌 데이터베이스 클러스터를 전환할지 여부를 지정합니다.

제약 조건:

- `AllowDataLoss` 파라미터와 함께 지정할 수 없습니다.

유형: 부울

필수 항목 여부: 아니요

응답 요소

서비스에서 다음 요소를 반환합니다.

GlobalCluster

Amazon DocumentDB 글로벌 클러스터를 나타내는 데이터 유형.

타입: GlobalCluster 객체

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하세요.

DBClusterNotFoundFault

`DBClusterIdentifier`은 기존 클러스터를 참조하지 않습니다.

HTTP 상태 코드: 404

GlobalClusterNotFoundFault

`GlobalClusterIdentifier`는 기존 글로벌 클러스터를 참조하지 않습니다.

HTTP 상태 코드: 404

InvalidDBClusterStateFault

DB 클러스터가 유효한 상태가 아닙니다.

HTTP 상태 코드: 400

InvalidGlobalClusterStateFault

클러스터가 이 상태인 동안에는 요청된 작업을 수행할 수 없습니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK

- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

ListTagsForResource

서비스: Amazon DocumentDB (with MongoDB compatibility)

Amazon DocumentDB 리소스의 모든 태그를 나열합니다.

요청 파라미터

모든 작업에서 공유하는 파라미터에 대한 내용은범용파라미터를 참조하세요.

ResourceName

목록으로 나열할 태그가 있는 Amazon DocumentDB 리소스입니다. 이 값은 Amazon 리소스 이름(ARN)입니다.

유형: 문자열

필수 항목 여부: 예

Filters.Filter.N

현재 지원되지 않는 파라미터입니다.

타입: Filter객체 배열

필수: 아니요

응답 요소

서비스에서 반환되는 요소는 다음과 같습니다.

TagList.tag.n

태그를 하나 이상 지정합니다.

타입: Tag 객체 배열

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하세요.

DBClusterNotFoundFault

`DBClusterIdentifier`은 기존 클러스터를 참조하지 않습니다.

HTTP 상태 코드: 404

DBInstanceNotFound

`DBInstanceIdentifier`는 기존 인스턴스를 참조하지 않습니다.

HTTP 상태 코드: 404

DBSnapshotNotFound

`DBSnapshotIdentifier`는 기존 스냅샷을 참조하지 않습니다.

HTTP 상태 코드: 404

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

ModifyDBCluster

서비스: Amazon DocumentDB (with MongoDB compatibility)

Amazon DocumentDB 클러스터에 대한 설정을 수정합니다. 요청에 이러한 데이터베이스 구성 파라미터와 새 값을 지정하여 하나 이상의 파라미터를 변경할 수 있습니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 내용은 공통 파라미터를 참조하십시오.

DBClusterIdentifier

수정 중인 클러스터의 클러스터 식별자입니다. 이 파라미터는 대소문자를 구분하지 않습니다.

제약 조건:

- 기존 DBCluster의 식별자와 일치해야 합니다.

유형: 문자열

필수 항목 여부: 예

AllowMajorVersionUpgrade

메이저 버전 업그레이드가 허용되는지 여부를 나타내는 값입니다.

제약 조건: DB 클러스터의 현재 버전과 다른 주 버전인 EngineVersion 파라미터 값을 지정하는 경우 주 버전 업그레이드를 허용해야 합니다.

유형: 부울

필수 항목 여부: 아니요

ApplyImmediately

클러스터의 PreferredMaintenanceWindow 설정과 관계없이, 이 요청의 변경 사항과 대기 중인 모든 변경 사항을 비동기적으로 최대한 빨리 적용할 것인지 여부를 지정하는 값입니다. 이 파라미터가 false로 설정되어 있으면 클러스터에 대한 변경 사항이 다음번 유지 관리 기간에 적용됩니다.

ApplyImmediately 파라미터는 NewDBClusterIdentifier 및 MasterUserPassword 값에만 영향을 줍니다. 이 파라미터 값을 false로 설정한 경우, NewDBClusterIdentifier 및 MasterUserPassword 값의 변경 사항이 다음번 유지 관리 기간에 적용됩니다. 그 밖의 모든 변경 사항은 ApplyImmediately 파라미터 값과 관계없이 즉시 적용됩니다.

기본값: `false`

유형: 부울

필수 항목 여부: 아니요

BackupRetentionPeriod

자동 백업이 보관되는 일수입니다. 1 이상의 값을 지정해야 합니다.

기본값: 1

제약 조건:

- 1~35의 값이어야 합니다.

유형: 정수

필수 항목 여부: 아니요

CloudwatchLogsExportConfiguration

특정 인스턴스 또는 클러스터에 대하여 Amazon CloudWatch Logs로 내보내기를 활성화할 로그 유형의 구성 설정입니다. `EnableLogTypes` 및 `DisableLogTypes` 배열이 CloudWatch Logs로 내보내는(또는 내보내지 않을) 로그를 결정합니다.

유형: CloudwatchLogsExportConfiguration객체

필수 여부: 아니요

DBClusterParameterGroupName

클러스터에 사용할 클러스터 파라미터 그룹의 이름입니다.

유형: 문자열

필수 항목 여부: 아니요

DeletionProtection

이 클러스터를 삭제할 수 있는지 없는지를 지정합니다. `DeletionProtection`이 활성화되면 클러스터를 수정하고 `DeletionProtection` 를 비활성화하지 않는 한 클러스터를 삭제할 수 없습니다. `DeletionProtection`은 클러스터가 실수로 삭제되지 않도록 보호합니다.

유형: 부울

필수 항목 여부: 아니요

EngineVersion

업그레이드할 데이터베이스 엔진의 버전 번호입니다. 이 파라미터를 변경해도 작동이 중단되지 않습니다. `ApplyImmediately`를 활성화하지 않은 한, 변경 사항은 다음번 유지 관리 기간에 적용됩니다.

Amazon DocumentDB에 대한 사용 가능한 모든 엔진 버전을 나열하려면 다음 명령을 사용합니다.

```
aws docdb describe-db-engine-versions --engine docdb --query
"DBEngineVersions[].EngineVersion"
```

유형: 문자열

필수 항목 여부: 아니요

ManageMasterUserPassword

Amazon Web Services Secrets Manager를 사용하여 마스터 사용자 암호를 관리할지 여부를 지정합니다. 클러스터가 Amazon Web Services Secrets Manager를 사용하여 마스터 사용자 암호를 관리하지 않는 경우이 관리를 켤 수 있습니다. 이 경우를 지정할 수 없습니다`MasterUserPassword`. 클러스터가 이미 Amazon Web Services Secrets Manager로 마스터 사용자 암호를 관리하고 있고 마스터 사용자 암호가 Amazon Web Services Secrets Manager로 관리되지 않도록 지정하는 경우를 지정해야 합니다`MasterUserPassword`. 이 경우 Amazon DocumentDB는 보안 암호를 삭제하고에서 지정한 마스터 사용자의 새 암호를 사용합니다`MasterUserPassword`.

유형: 부울

필수 항목 여부: 아니요

MasterUserPassword

마스터 데이터베이스 사용자의 암호입니다. 이 암호에는 슬래시(/), 큰따옴표(") 또는 '앳' 기호(@)를 제외한 인쇄 가능 ASCII 문자가 포함될 수 있습니다.

제약: 8~100자여야 합니다.

유형: 문자열

필수 항목 여부: 아니요

MasterUserSecretKmsKeyId

Amazon Web Services Secrets Manager에서 자동으로 생성되고 관리되는 보안 암호를 암호화하기 위한 Amazon Web Services KMS 키 식별자입니다.

이 설정은 다음 두 조건이 모두 충족되는 경우에만 유효합니다.

- 클러스터는 Amazon Web Services Secrets Manager에서 마스터 사용자 암호를 관리하지 않습니다. 클러스터가 이미 Amazon Web Services Secrets Manager에서 마스터 사용자 암호를 관리하는 경우 보안 암호를 암호화하는 데 사용되는 KMS 키를 변경할 수 없습니다.
- Amazon Web Services Secrets Manager에서 마스터 사용자 암호를 관리할 수 `ManageMasterUserPassword` 있습니다. 를 켜고 `MasterUserSecretKmsKeyId`를 지정`ManageMasterUserPassword`하지 않으면 `aws/secretsmanager` KMS 키가 보안 암호를 암호화하는 데 사용됩니다. 보안 암호가 다른 Amazon Web Services 계정에 있는 경우 `aws/secretsmanager` KMS 키를 사용하여 보안 암호를 암호화할 수 없으며 고객 관리형 KMS 키를 사용해야 합니다.

Amazon Web Services KMS 키 식별자는 KMS 키의 키 ARN, 키 ID, 별칭 ARN 또는 별칭 이름입니다. 다른 Amazon Web Services 계정에서 KMS 키를 사용하려면 키 ARN 또는 별칭 ARN을 지정합니다.

Amazon Web Services 계정에 대한 기본 KMS 키가 있습니다. Amazon Web Services 계정에는 Amazon Web Services 리전마다 다른 기본 KMS 키가 있습니다.

유형: 문자열

필수 항목 여부: 아니요

NewDBClusterIdentifier

클러스터의 이름을 변경할 때 클러스터의 새 클러스터 식별자입니다. 이 값은 소문자 문자열로 저장됩니다.

제약 조건:

- 1~63자의 문자, 숫자 또는 하이픈으로 구성되어야 합니다.
- 첫 자는 문자여야 합니다.
- 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다.

예시: `my-cluster2`

유형: 문자열

필수 항목 여부: 아니요

Port

클러스터에서 연결을 허용하는 포트 번호입니다.

제약 조건: 값이 1150~65535여야 합니다.

기본값: 원래의 클러스터와 동일한 포트입니다.

유형: 정수

필수 항목 여부: 아니요

PreferredBackupWindow

BackupRetentionPeriod 파라미터를 사용하여 자동 백업을 활성화한 경우, 자동 백업이 생성되는 일일 시간 범위입니다.

기본값은 각각에 대해 8시간 블록 시간 중에서 임의로 선택한 30분 기간입니다 AWS 리전.

제약 조건:

- hh24:mi-hh24:mi 형식이어야 합니다.
- 협정 세계시(UTC)여야 합니다.
- 원하는 유지 관리 기간과 충돌하지 않아야 합니다.
- 30분 이상이어야 합니다.

유형: 문자열

필수 항목 여부: 아니요

PreferredMaintenanceWindow

시스템 유지 관리를 실행할 수 있는 주 단위 기간(UTC, 협정 세계시)입니다.

형식: ddd:hh24:mi-ddd:hh24:mi

기본값은 각에 대해 8시간 블록 시간 중에서 임의로 선택되는 30분 기간으로 AWS 리전, 요일에 무작위로 발생합니다.

유효한 요일: 월, 화, 수, 목, 금, 토, 일

제약 조건: 최소 30분의 기간.

유형: 문자열

필수 항목 여부: 아니요

RotateMasterUserPassword

Amazon Web Services Secrets Manager에서 관리하는 보안 암호를 마스터 사용자 암호로 교체할지 여부를 지정합니다.

이 설정은 클러스터용 Amazon Web Services Secrets Manager의 Amazon DocumentDB에서 마스터 사용자 암호를 관리하는 경우에만 유효합니다. 보안 암호 값에는 업데이트된 암호가 포함됩니다.

제약: 마스터 사용자 암호를 교체할 때 변경 사항을 즉시 적용해야 합니다.

유형: 부울

필수 항목 여부: 아니요

StorageType

DB 클러스터와 연결할 스토리지 유형입니다.

Amazon DocumentDB 클러스터의 스토리지 유형에 관한 자세한 내용은 Amazon DocumentDB 개발자 안내서의 클러스터 스토리지 구성을 참조하세요.

스토리지 유형에 유효한 값 - `standard | iopt1`

기본값은 `standard` 입니다.

유형: 문자열

필수 항목 여부: 아니요

VpcSecurityGroupIds.VpcSecurityGroupId.N

클러스터가 속하게 될 Virtual Private Cloud(VPC) 보안 그룹 목록입니다.

유형: 문자열 배열

필수 여부: 아니요

응답 요소

서비스에서 다음 요소를 반환합니다.

DBCluster

클러스터에 관한 자세한 정보입니다.

타입: DBCluster 객체

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하세요.

DBClusterAlreadyExistsFault

해당 식별자를 사용하는 클러스터가 이미 있습니다.

HTTP 상태 코드: 400

DBClusterNotFoundFault

`DBClusterIdentifier`는 기존 클러스터를 참조하지 않습니다.

HTTP 상태 코드: 404

DBClusterParameterGroupNotFound

`DBClusterParameterGroupName`는 기존 클러스터 파라미터 그룹을 참조하지 않습니다.

HTTP 상태 코드: 404

DBSubnetGroupNotFoundFault

`DBSubnetGroupName`는 기존 서브넷 그룹을 참조하지 않습니다.

HTTP 상태 코드: 404

InvalidDBClusterStateFault

DB 클러스터가 유효한 상태가 아닙니다.

HTTP 상태 코드: 400

InvalidDBInstanceState

지정된 인스턴스가 사용 가능한 상태가 아닙니다.

HTTP 상태 코드: 400

InvalidDBSecurityGroupState

보안 그룹의 상태로 인해 삭제할 수 없습니다.

HTTP 상태 코드: 400

InvalidDBSubnetGroupStateFault

서브넷 그룹이 사용 중이므로 삭제할 수 없습니다.

HTTP 상태 코드: 400

InvalidSubnet

요청한 서브넷이 올바르지 않거나 공통 Virtual Private Cloud(VPC)에 모두 포함되지 않은 여러 서브넷이 요청되었습니다.

HTTP 상태 코드: 400

InvalidVPCNetworkStateFault

서브넷 그룹이 생성된 후에는 변경 사항으로 인해 모든 가용 영역에 적용되지 않습니다.

HTTP 상태 코드: 400

StorageQuotaExceeded

요청으로 인해 모든 인스턴스에서 사용 가능한 스토리지 허용량을 초과하게 됩니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go v2용 SDK
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

ModifyDBClusterParameterGroup

서비스: Amazon DocumentDB (with MongoDB compatibility)

DB 클러스터 파라미터 그룹의 파라미터를 수정합니다. 하나 이상의 파라미터를 수정하려면 `ParameterName`, `ParameterValue` 및 `ApplyMethod`의 목록을 제출하십시오. 요청 하나에서 최대 20개의 파라미터를 수정할 수 있습니다.

> **Note**
>
> 동적 파라미터의 변경 내용은 즉시 적용됩니다. 정적 매개변수를 변경한 경우 변경 내용을 적용하려면 재부팅 또는 유지 관리 기간이 필요합니다.

> **Important**
>
> 클러스터 파라미터 그룹을 생성한 후 해당 클러스터 파라미터 그룹을 기본 파라미터 그룹으로 사용하는 첫 번째 클러스터를 생성하기 전에 5분 이상 기다려야 합니다. 이렇게 하면 Amazon DocumentDB가 생성 작업을 완전히 마친 뒤에 해당 파라미터 그룹을 새 클러스터의 기본값으로 사용할 수 있습니다. 이 단계는 `character_set_database` 파라미터로 정의한 기본 데이터베이스의 문자 세트 등 클러스터의 기본 데이터베이스를 생성할 때 필요한 파라미터의 경우 특히 중요합니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 내용은 공통 파라미터를 참조하십시오.

DBClusterParameterGroupName

수정할 클러스터 파라미터 그룹의 이름입니다.

유형: 문자열

필수 항목 여부: 예

Parameters.Parameter.N

클러스터 파라미터 그룹에서 수정할 파라미터의 목록입니다.

타입: Parameter객체 배열

필수 여부: 예

응답 요소

서비스에서 다음 요소를 반환합니다.

DBClusterParameterGroupName

클러스터 파라미터 그룹의 이름입니다.

제약 조건:

- 1~255자의 문자 또는 숫자여야 합니다.
- 첫 자는 문자여야 합니다.
- 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다.

> **Note**
> 이 값은 소문자 문자열로 저장됩니다.

유형: 문자열

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하십시오.

DBParameterGroupNotFound

`DBParameterGroupName`은 기존 파라미터 그룹을 참조하지 않습니다.

HTTP 상태 코드: 404

InvalidDBParameterGroupState

파라미터 그룹이 사용 중이거나 유효하지 않은 상태입니다. 파라미터 그룹을 삭제하려는 경우 파라미터 그룹이 이 상태일 때는 삭제할 수 없습니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

ModifyDBClusterSnapshotAttribute

서비스: Amazon DocumentDB (with MongoDB compatibility)

속성 및 값을 수동 클러스터 스냅샷에 추가하거나, 수동 DB 클러스터 스냅샷에서 속성 및 값을 제거합니다.

수동 클러스터 스냅샷을 다른와 공유하려면를 `restore`로 AWS 계정지정`AttributeName`하고 `ValuesToAdd` 파라미터를 사용하여 수동 클러스터 스냅샷을 복원할 권한이 AWS 계정 있는의 IDs 목록을 추가합니다. 수동 클러스터 스냅샷을 모든 AWS 계정에서 복사하거나 복원할 수 있는 퍼블릭 스냅샷으로 만들려면 `all` 값을 사용하십시오. 모든 AWS 계정에는 제공하지 않을 비공개 정보가 포함된 수동 클러스터 스냅샷에는 `all` 값을 추가하지 마십시오. 수동 클러스터 스냅샷이 암호화된 경우 `ValuesToAdd` 파라미터에 대해 AWS 계정 IDs 목록을 지정해야만 공유할 수 있습니다. 이 경우에는 해당 파라미터 값으로 `all`을 사용할 수 없습니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 내용은 공통 파라미터를 참조하십시오.

AttributeName

수정할 클러스터 스냅샷 속성의 이름입니다.

다른이 수동 클러스터 스냅샷을 복사하거나 복원 AWS 계정 할 수 있는 권한을 관리하려면이 값을로 설정합니다`restore`.

유형: 문자열

필수 항목 여부: 예

DBClusterSnapshotIdentifier

속성을 수정할 클러스터 스냅샷의 식별자입니다.

유형: 문자열

필수 항목 여부: 예

ValuesToAdd.AttributeValue.N

`AttributeName`에 지정된 속성에 추가할 클러스터 스냅샷 속성의 목록입니다.

다른 사용자에게 수동 클러스터 스냅샷을 복사하거나 복원 AWS 계정 할 수 있는 권한을 부여하려면이 목록을 하나 이상의 AWS 계정 IDs를 포함하도록 설정합니다. 수동 클러스터 스냅샷을에서 복

원할 수 있도록 하려면 로 AWS 계정설정합니다`all`. 모든 AWS 계정에는 제공하지 않을 비공개 정보가 포함된 수동 클러스터 스냅샷에는 `all` 값을 추가하지 마십시오.

유형: 문자열 배열

필수 여부: 아니요

ValuesToRemove.AttributeValue.N

`AttributeName`에 지정된 속성에서 제거할 클러스터 스냅샷 속성의 목록입니다.

다른이 수동 클러스터 스냅샷을 복사하거나 복원 AWS 계정 할 수 있는 권한을 제거하려면이 목록을 하나 이상의 AWS 계정 식별자를 포함하도록 설정합니다. 클러스터 스냅샷을 복사하거나 복원 AWS 계정 할 수 있는 권한 부여를 제거하려면 `all` 로 설정합니다. `all`를 지정하더라도 `restore` 속성에 계정 ID AWS 계정 가 명시적으로 추가된는 수동 클러스터 스냅샷을 복사하거나 복원할 수 있습니다.

유형: 문자열 배열

필수 여부: 아니요

응답 요소

서비스에서 다음 요소를 반환합니다.

DBClusterSnapshotAttributesResult

클러스터 스냅샷과 관련된 특성에 대한 상세 정보입니다.

타입: DBClusterSnapshotAttributesResult 객체

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하십시오.

DBClusterSnapshotNotFoundFault

`DBClusterSnapshotIdentifier`은 기존 클러스터 스냅샷을 참조하지 않습니다.

HTTP 상태 코드: 404

InvalidDBClusterSnapshotStateFault

제공된 값은 유효한 클러스터 스냅샷 상태가 아닙니다.

HTTP 상태 코드: 400

SharedSnapshotQuotaExceeded

수동 DB 스냅샷을 공유할 수 있는 최대 계정 수를 초과했습니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

ModifyDBInstance

서비스: Amazon DocumentDB (with MongoDB compatibility)

인스턴스의 설정을 수정합니다. 요청에 이러한 데이터베이스 구성 파라미터와 새 값을 지정하여 하나 이상의 파라미터를 변경할 수 있습니다.

요청 파라미터

모든 작업에 공통으로 적용되는 파라미터에 대한 자세한 내용은 공통 파라미터를 참조하십시오.

DBInstanceIdentifier

인스턴스 식별자입니다. 이 값은 소문자 문자열로 저장됩니다.

제약 조건:

- 기존 `DBInstance`의 식별자와 일치해야 합니다.

유형: 문자열

필수 항목 여부: 예

ApplyImmediately

인스턴스의 `PreferredMaintenanceWindow` 설정과 관계없이, 이 요청의 수정 사항과 보류 중인 모든 수정 사항을 비동기적으로 최대한 빨리 적용할 것인지 여부를 지정합니다.

이 파라미터가 `false`로 설정되어 있으면 인스턴스에 대한 변경 사항이 다음번 유지 관리 기간에 적용됩니다. 일부 파라미터는 변경하면 중단이 발생하며, 다음번 재부팅 시 적용됩니다.

기본값: `false`

유형: 부울

필수 항목 여부: 아니요

AutoMinorVersionUpgrade

이 파라미터는 Amazon DocumentDB에는 적용되지 않습니다. Amazon DocumentDB는 값 세트에 관계없이 마이너 버전 업그레이드를 수행하지 않습니다.

유형: 부울

필수 항목 여부: 아니요

CACertificateIdentifier

인스턴스와 연결해야 하는 인증서를 나타냅니다.

유형: 문자열

필수 항목 여부: 아니요

CertificateRotationRestart

SSL/TLS 인증서를 교체할 때 DB 인스턴스를 다시 시작할지를 지정합니다.

기본적으로 SSL/TLS 인증서를 교체하면 DB 인스턴스가 다시 시작됩니다. DB 인스턴스를 다시 시작할 때까지 인증서가 업데이트되지 않습니다.

> **Important**
>
> SSL/TLS를 사용하여 DB 인스턴스에 연결하지 않는 경우에만 이 파라미터를 설정합니다.

SSL/TLS를 사용하여 DB 인스턴스에 연결하는 경우, Amazon DocumentDB 개발자 안내서의 Amazon DocumentDB TLS 인증서 업데이트 및 전송 중 데이터 암호화를 참조하세요.

유형: 부울

필수 항목 여부: 아니요

CopyTagsToSnapshot

모든 태그를 DB 인스턴스에서 DB 인스턴스의 스냅샷으로 복사할지 여부를 나타내는 값입니다. 태그는 기본적으로 복사되지 않습니다.

유형: 부울

필수 항목 여부: 아니요

DBInstanceClass

인스턴스의 컴퓨팅 및 메모리 용량입니다(예: `db.r5.large`). 모든 AWS 리전에서 모든 인스턴스 클래스를 사용할 수 있는 것은 아닙니다.

인스턴스 클래스를 수정하면 변경 중에 중단이 발생합니다. 이 요청에 대해 `ApplyImmediately`를 `true`로 지정하지 않은 한, 변경 사항은 다음번 유지 관리 기간에 적용됩니다.

기본값: 기존 설정을 사용합니다.

유형: 문자열

필수 항목 여부: 아니요

EnablePerformanceInsights

DB 인스턴스에 Performance Insights를 활성화할지 여부를 나타내는 값입니다. 자세한 내용은 Amazon 성능 개선 도우미 사용을 참조하세요.

유형: 부울

필수 항목 여부: 아니요

NewDBInstanceIdentifier

인스턴스의 이름을 변경할 때 인스턴스의 새로운 인스턴스 식별자입니다. 인스턴스 식별자를 변경할 때 `Apply Immediately`을 `true`로 설정하면 인스턴스 식별자가 즉시 재부팅됩니다. `Apply Immediately`을 `false`로 설정하면 다음 유지 관리 기간에 발생합니다. 이 값은 소문자 문자열로 저장됩니다.

제약 조건:

- 1~63자의 문자, 숫자 또는 하이픈으로 구성되어야 합니다.
- 첫 자는 문자여야 합니다.
- 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다.

예시: `mydbinstance`

유형: 문자열

필수 항목 여부: 아니요

PerformanceInsightsKMSKeyId

성능 개선 도우미 데이터의 암호화를 위한 AWS KMS 키 식별자입니다.

AWS KMS 키 식별자는 KMS 키의 키 ARN, 키 ID, 별칭 ARN 또는 별칭 이름입니다.

PerformanceInsightsKMSKeyId에 대한 값을 지정하지 않으면 Amazon DocumentDB에서는 기본 KMS 키를 사용합니다. Amazon Web Services 계정에 대한 기본 KMS 키가 있습니다. Amazon Web Services 계정에는 Amazon Web Services 리전마다 다른 기본 KMS 키가 있습니다.

유형: 문자열

필수 항목 여부: 아니요

PreferredMaintenanceWindow

중단으로 이어질 가능성이 있는 시스템 유지 관리를 실행할 수 있는 주 단위 기간(UTC)입니다. 다음 상황을 제외하고는 이 파라미터를 변경해도 중단되지 않으며, 변경 사항은 비동기적으로 최대한 빨리 적용됩니다. 재부팅해야 하는 작업이 대기 중이고 유지 관리 기간이 현재 시간을 포함하도록 변경된 경우, 이 파라미터를 변경하면 인스턴스가 재부팅됩니다. 이 기간을 현재 시간으로 옮기는 경우, 대기 중인 변경 사항이 적용될 수 있도록 현재 시간과 기간 종료 사이에 최소 30분의 시간을 두어야 합니다.

기본값: 기존 설정을 사용합니다.

형식: `ddd:hh24:mi-ddd:hh24:mi`

유효한 요일: 월, 화, 수, 목, 금, 토, 일

제약: 30분 이상이어야 합니다.

유형: 문자열

필수 항목 여부: 아니요

PromotionTier

기존 기본 인스턴스에 결함이 발생한 후 Amazon DocumentDB 복제본을 기본 인스턴스로 승격할 순서를 지정하는 값

기본값: 1

유효한 값: 0~15

유형: 정수

필수 항목 여부: 아니요

응답 요소

서비스에서 다음 요소를 반환합니다.

DBInstance

인스턴스에 대한 자세한 정보입니다.

타입: DBInstance 객체

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하세요.

AuthorizationNotFound

지정한 CIDR IP 또는 Amazon EC2 보안 그룹에 대해 지정한 보안 그룹에 대한 권한이 없습니다.

Amazon DocumentDB는 또한 IAM을 사용하여 사용자를 대신하여 필요한 작업을 수행할 권한이 없을 수도 있습니다.

HTTP 상태 코드: 404

CertificateNotFound

`CertificateIdentifier`는 기존 인증서를 참조하지 않습니다.

HTTP 상태 코드: 404

DBInstanceAlreadyExists

해당 식별자를 사용하는 인스턴스가 이미 있습니다.

HTTP 상태 코드: 400

DBInstanceNotFound

`DBInstanceIdentifier`는 기존 인스턴스를 참조하지 않습니다.

HTTP 상태 코드: 404

DBParameterGroupNotFound

`DBParameterGroupName`은 기존 파라미터 그룹을 참조하지 않습니다.

HTTP 상태 코드: 404

DBSecurityGroupNotFound

`DBSecurityGroupName`는 기존 보안 그룹을 참조하지 않습니다.

HTTP 상태 코드: 404

DBUpgradeDependencyFailure

종속 대상인 리소스를 수정할 수 없어 업그레이드에 실패했습니다.

HTTP 상태 코드: 400

InsufficientDBInstanceCapacity

지정한 인스턴스 클래스를 지정한 가용성 영역에서 사용할 수 없습니다.

HTTP 상태 코드: 400

InvalidDBInstanceState

지정된 인스턴스가 사용 가능한 상태가 아닙니다.

HTTP 상태 코드: 400

InvalidDBSecurityGroupState

보안 그룹의 상태로 인해 삭제할 수 없습니다.

HTTP 상태 코드: 400

InvalidVPCNetworkStateFault

서브넷 그룹이 생성된 후에는 변경 사항으로 인해 모든 가용 영역에 적용되지 않습니다.

HTTP 상태 코드: 400

StorageQuotaExceeded

요청으로 인해 모든 인스턴스에서 사용 가능한 스토리지 허용량을 초과하게 됩니다.

HTTP 상태 코드: 400

StorageTypeNotSupported

지정된 `StorageType`을 이 DB 인스턴스와 연결할 수 없습니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface

- AWS SDK for .NET
- AWS SDK for C++
- AWS Go v2용 SDK
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

ModifyDBSubnetGroup

서비스: Amazon DocumentDB (with MongoDB compatibility)

기존 서브넷 그룹을 수정합니다. 서브넷 그룹은 AWS 리전에서 최소 두 개의 가용 영역에 적어도 하나의 서브넷을 포함해야 합니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 내용은 공통 파라미터를 참조하십시오.

DBSubnetGroupName

서브넷 그룹의 이름입니다. 이 값은 소문자 문자열로 저장됩니다. 기본 서브넷 그룹은 수정할 수 없습니다.

제약: 기존의 `DBSubnetGroup` 이름과 일치해야 합니다. 기본값이 아니어야 합니다.

예시: `mySubnetgroup`

유형: 문자열

필수 항목 여부: 예

SubnetIds.SubnetIdentifier.N

서브넷 그룹의 Amazon EC2 서브넷 ID입니다.

유형: 문자열 배열

필수 여부: 예

DBSubnetGroupDescription

서브넷 그룹에 대한 설명입니다.

유형: 문자열

필수 항목 여부: 아니요

응답 요소

서비스에서 다음 요소를 반환합니다.

DBSubnetGroup

서브넷 그룹에 대한 자세한 정보.

타입: DBSubnetGroup 객체

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하세요.

DBSubnetGroupDoesNotCoverEnoughAZs

가용 영역이 하나뿐인 경우를 제외하고, 서브넷 그룹의 서브넷은 최소한 두 개의 가용 영역을 포함해야 합니다.

HTTP 상태 코드: 400

DBSubnetGroupNotFoundFault

`DBSubnetGroupName`은 기존 서브넷 그룹을 참조하지 않습니다.

HTTP 상태 코드: 404

DBSubnetQuotaExceededFault

이 요청으로 인해 서브넷 그룹에서 허용되는 서브넷 수를 초과하게 됩니다.

HTTP 상태 코드: 400

InvalidSubnet

요청한 서브넷이 올바르지 않거나 공통 Virtual Private Cloud(VPC)에 모두 포함되지 않은 여러 서브넷이 요청되었습니다.

HTTP 상태 코드: 400

SubnetAlreadyInUse

가용 영역에서 이미 사용 중인 서브넷입니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

ModifyEventSubscription

서비스: Amazon DocumentDB (with MongoDB compatibility)

기존 Amazon DocumentDB 이벤트 알림 구독을 수정합니다.

요청 파라미터

모든 작업에 공통으로 적용되는 파라미터에 대한 자세한 내용은 공통 파라미터를 참조하십시오.

SubscriptionName

Amazon DocumentDB 이벤트 알림 구독의 이름.

유형: 문자열

필수 항목 여부: 예

Enabled

부울 값입니다. 구독을 활성화하려면 `true`로 설정합니다.

유형: 부울

필수 항목 여부: 아니요

EventCategories.EventCategory.N

구독할 `SourceType`의 이벤트 범주 목록입니다.

유형: 문자열 배열

필수 여부: 아니요

SnsTopicArn

이벤트 알림을 위해 생성한 SNS 주제의 Amazon 리소스 이름(ARN)입니다. 주제를 만들고 구독하면 Amazon SNS에서 ARN이 생성됩니다.

유형: 문자열

필수 항목 여부: 아니요

SourceType

이벤트가 발생하는 소스의 유형입니다. 예를 들어, 인스턴스에서 생성되는 이벤트에 대한 알림을 받으려면 이 파라미터를 `db-instance`로 설정합니다. 이 값을 지정하지 않으면 모든 이벤트가 반환됩니다.

유효한 값: db-instance, db-parameter-group, db-security-group

유형: 문자열

필수 항목 여부: 아니요

응답 요소

서비스에서 다음 요소를 반환합니다.

EventSubscription

구독한 이벤트에 대한 세부 정보.

타입: EventSubscription 객체

오류

모든 작업에서 공통적으로 발생하는 오류에 대한 자세한 내용은 일반적인 오류 단원을 참조하십시오.

EventSubscriptionQuotaExceeded

최대 이벤트 구독 수에 도달했습니다.

HTTP 상태 코드: 400

SNSInvalidTopic

Amazon SNS에서 지정된 주제에 문제가 있다고 응답했습니다.

HTTP 상태 코드: 400

SNSNoAuthorization

SNS 주제 Amazon 리소스 이름(ARN) 에 게시할 권한이 없습니다.

HTTP 상태 코드: 400

SNSTopicArnNotFound

SNS 주제 Amazon 리소스 이름(ARN)이 존재하지 않습니다

HTTP 상태 코드: 404

SubscriptionCategoryNotFound

제공된 범주가 존재하지 않습니다.

HTTP 상태 코드: 404

SubscriptionNotFound

구독의 이름이 존재하지 않습니다.

HTTP 상태 코드: 404

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

ModifyGlobalCluster

서비스: Amazon DocumentDB (with MongoDB compatibility)

Amazon DocumentDB 글로벌 클러스터에 대한 설정을 수정합니다. 요청에서 이러한 파라미터와 새 값을 지정하여 하나 이상의 구성 파라미터 (예: 삭제 보호) 또는 글로벌 클러스터 식별자를 변경할 수 있습니다.

> **Note**
>
> 이 작업은 Amazon DocumentDB 클러스터에만 적용됩니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 내용은 공통 파라미터를 참조하십시오.

GlobalClusterIdentifier

수정 중인 글로벌 클러스터의 식별자입니다. 이 파라미터는 대/소문자를 구분하지 않습니다.

제약 조건:

- 기존 글로벌 클러스터의 식별자와 일치해야 합니다.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 255.

패턴: `[A-Za-z][0-9A-Za-z-:._]*`

필수 여부: 예

DeletionProtection

글로벌 클러스터의 삭제 방지 기능 활성화 여부를 나타냅니다. 삭제 방지 기능이 활성화되면 글로벌 클러스터가 삭제될 수 없습니다.

유형: 부울

필수 항목 여부: 아니요

NewGlobalClusterIdentifier

글로벌 클러스터를 수정할 때 표시되는 글로벌 클러스터의 새 식별자. 이 값은 소문자 문자열로 저장됩니다.

- 1~63자의 문자, 숫자 또는 하이픈으로 구성되어야 합니다.

  첫 번째 자리는 문자여야 합니다.

  하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다.

예시: `my-cluster2`

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 255.

패턴: `[A-Za-z][0-9A-Za-z-:._]*`

필수 여부: 아니요

응답 요소

서비스에서 다음 요소를 반환합니다.

GlobalCluster

Amazon DocumentDB 글로벌 클러스터를 나타내는 데이터 유형.

타입: GlobalCluster 객체

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하세요.

GlobalClusterNotFoundFault

`GlobalClusterIdentifier`는 기존 글로벌 클러스터를 참조하지 않습니다.

HTTP 상태 코드: 404

InvalidGlobalClusterStateFault

클러스터가 이 상태인 동안에는 요청된 작업을 수행할 수 없습니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

RebootDBInstance

서비스: Amazon DocumentDB (with MongoDB compatibility)

일반적으로 유지 관리를 이유로 인스턴스를 재부팅해야 할 수 있습니다. 예를 들어, 특정 내용을 변경하거나 인스턴스에 연결된 클러스터 파라미터 그룹을 변경하는 경우 인스턴스를 재부팅해야 변경 내용이 적용됩니다.

인스턴스를 재부팅하면 데이터베이스 엔진 서비스가 재시작됩니다. 인스턴스를 재부팅하면 인스턴스 상태가 재부팅으로 설정되면서 잠시 중단됩니다.

요청 파라미터

모든 작업에 공통으로 적용되는 파라미터에 대한 자세한 내용은 공통 파라미터를 참조하십시오.

DBInstanceIdentifier

인스턴스 식별자입니다. 이 파라미터는 소문자 문자열로 저장됩니다.

제약 조건:

- 기존 `DBInstance`의 식별자와 일치해야 합니다.

유형: 문자열

필수 항목 여부: 예

ForceFailover

`true`인 경우, Multi-AZ 장애 조치를 통해 재부팅이 이루어집니다.

제약: 인스턴스가 Multi-AZ용으로 구성되지 않았으면 `true`로 지정할 수 없습니다.

유형: 부울

필수 항목 여부: 아니요

응답 요소

서비스에서 다음 요소를 반환합니다.

DBInstance

인스턴스에 대한 자세한 정보입니다.

타입: DBInstance 객체

오류

모든 작업에 공동되는 오류에 대한 자세한 내용은 일반적인 오류을 참조하십시오.

DBInstanceNotFound

DBInstanceIdentifier는 기존 인스턴스를 참조하지 않습니다.

HTTP 상태 코드: 404

InvalidDBInstanceState

지정된 인스턴스는 사용 가능 상태가 아닙니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

RemoveFromGlobalCluster

서비스: Amazon DocumentDB (with MongoDB compatibility)

Amazon DocumentDB 보조 클러스터를 글로벌 클러스터에서 분리합니다. 클러스터는 읽기 전용이고 다른 지역의 기본 클러스터로부터 데이터를 수신하는 대신 읽기-쓰기 기능을 갖춘 독립 실행형 클러스터가 됩니다.

> **Note**
>
> 이 작업은 Amazon DocumentDB 클러스터에만 적용됩니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 내용은 공통 파라미터를 참조하십시오.

DbClusterIdentifier

Amazon DocumentDB 글로벌 클러스터에서 분리된 클러스터를 식별하는 Amazon 리소스 이름(ARN)입니다.

유형: 문자열

필수 항목 여부: 예

GlobalClusterIdentifier

Amazon DocumentDB 글로벌 클러스터에서 분리할 클러스터 식별자입니다.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 255.

패턴: `[A-Za-z][0-9A-Za-z-:._]*`

필수 여부: 예

응답 요소

서비스에서 다음 요소를 반환합니다.

GlobalCluster

Amazon DocumentDB 글로벌 클러스터를 나타내는 데이터 유형.

타입: GlobalCluster 객체

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하세요.

DBClusterNotFoundFault

`DBClusterIdentifier`은 기존 클러스터를 참조하지 않습니다.

HTTP 상태 코드: 404

GlobalClusterNotFoundFault

`GlobalClusterIdentifier`는 기존 글로벌 클러스터를 참조하지 않습니다.

HTTP 상태 코드: 404

InvalidGlobalClusterStateFault

클러스터가 이 상태인 동안에는 요청된 작업을 수행할 수 없습니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK

- AWS Python용 SDK
- AWS SDK for Ruby V3

RemoveSourceIdentifierFromSubscription

서비스: Amazon DocumentDB (with MongoDB compatibility)

기존의 Amazon DocumentDB 이벤트 알림 구독에서 소스 식별자를 제거합니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 자세한 내용은 공통 파라미터를 참조하십시오.

SourceIdentifier

인스턴스의 인스턴스 식별자 또는 보안 그룹의 이름 등 구독에서 제거할 소스 식별자입니다.

유형: 문자열

필수 항목 여부: 예

SubscriptionName

소스 식별자를 제거하려는 Amazon DocumentDB 이벤트 알림 구독의 이름입니다.

유형: 문자열

필수 항목 여부: 예

응답 요소

서비스에서 다음 요소를 반환합니다.

EventSubscription

구독한 이벤트에 대한 세부 정보.

타입: EventSubscription 객체

오류

모든 작업에서 공통적으로 발생하는 오류에 대한 자세한 내용은 일반적인 오류 단원을 참조하십시오.

SourceNotFound

요청한 소스를 찾을 수 없습니다.

HTTP 상태 코드: 404

SubscriptionNotFound

구독의 이름이 존재하지 않습니다.

HTTP 상태 코드: 404

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

RemoveTagsFromResource

서비스: Amazon DocumentDB (with MongoDB compatibility)

Amazon DocumentDB 리소스에서 메타데이터 태그를 제거합니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 자세한 내용은 공통 파라미터를 참조하십시오.

ResourceName

에서 태그를 제거한 Amazon DocumentDB 리소스입니다. 이 값은 Amazon 리소스 이름(ARN)입니다.

유형: 문자열

필수 항목 여부: 예

TagKeys.member.N

제거할 태그의 태그 키(이름)입니다.

유형: 문자열 배열

필수 여부: 예

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하세요.

DBClusterNotFoundFault

`DBClusterIdentifier`은 기존 클러스터를 참조하지 않습니다.

HTTP 상태 코드: 404

DBInstanceNotFound

`DBInstanceIdentifier`는 기존 인스턴스를 참조하지 않습니다.

HTTP 상태 코드: 404

DBSnapshotNotFound

`DBSnapshotIdentifier`는 기존 스냅샷을 참조하지 않습니다.

HTTP 상태 코드: 404

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

ResetDBClusterParameterGroup

서비스: Amazon DocumentDB (with MongoDB compatibility)

클러스터 파라미터 그룹의 파라미터를 기본값으로 수정합니다. 특정 파라미터를 재설정하려면 `ParameterName` 및 `ApplyMethod`의 목록을 제출해야 합니다. 클러스터 파라미터 그룹 전체를 재설정하려면 `DBClusterParameterGroupName` 및 `ResetAllParameters` 파라미터를 지정합니다.

전체 그룹을 재설정하면 동적 파라미터가 즉시 업데이트되고 정적 파라미터는 `pending-reboot`로 설정되어 다음 DB 인스턴스 재시작 시 적용됩니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 내용은 공통 파라미터를 참조하십시오.

DBClusterParameterGroupName

재설정할 클러스터 파라미터 그룹의 이름입니다.

유형: 문자열

필수 항목 여부: 예

Parameters.Parameter.N

기본값으로 재설정하려는 클러스터 파라미터 그룹의 파라미터 이름 목록입니다. `ResetAllParameters` 파라미터를 `true`로 설정한 경우 이 파라미터를 사용할 수 없습니다.

타입: Parameter객체 배열

필수: 아니요

ResetAllParameters

클러스터 파라미터 그룹의 모든 파라미터를 기본값으로 재설정하려면 이 값을 `true`로 설정하고, 그렇지 않으면 `false`로 설정합니다. `Parameters` 파라미터용으로 지정된 파라미터 이름 목록이 있는 경우에는 이 파라미터를 사용할 수 없습니다.

유형: 부울

필수 항목 여부: 아니요

응답 요소

서비스에서 다음 요소를 반환합니다.

DBClusterParameterGroupName

클러스터 파라미터 그룹의 이름입니다.

제약 조건:

- 1~255자의 문자 또는 숫자여야 합니다.
- 첫 자는 문자여야 합니다.
- 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다.

> **Note**
> 이 값은 소문자 문자열로 저장됩니다.

유형: 문자열

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하십시오.

DBParameterGroupNotFound

`DBParameterGroupName`은 기존 파라미터 그룹을 참조하지 않습니다.

HTTP 상태 코드: 404

InvalidDBParameterGroupState

파라미터 그룹이 사용 중이거나 유효하지 않은 상태입니다. 파라미터 그룹을 삭제하려는 경우 파라미터 그룹이 이 상태일 때는 삭제할 수 없습니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++

- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

RestoreDBClusterFromSnapshot

서비스: Amazon DocumentDB (with MongoDB compatibility)

스냅샷 또는 클러스터 스냅샷에서 새 클러스터를 생성합니다.

스냅샷을 지정하는 경우, 기본 구성과 기본 보안 그룹으로 원본 스냅샷에서 대상 DB 클러스터가 생성됩니다.

클러스터 스냅샷을 지정하는 경우, 원래의 원본 DB 클러스터와 동일한 구성으로 원본 클러스터의 복원 지점에서 대상 클러스터가 생성됩니다. 단, 새 클러스터가 기본 보안 그룹으로 생성된 경우는 예외입니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 내용은 공통 파라미터를 참조하십시오.

DBClusterIdentifier

스냅샷 또는 클러스터 스냅샷에서 생성할 클러스터의 이름입니다. 이 파라미터는 대/소문자를 구분하지 않습니다.

제약 조건:

- 1~63자의 문자, 숫자 또는 하이픈으로 구성되어야 합니다.
- 첫 자는 문자여야 합니다.
- 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다.

예시: `my-snapshot-id`

유형: 문자열

필수 항목 여부: 예

Engine

새 클러스터에 사용할 데이터베이스 엔진입니다.

기본값: 원본과 동일합니다.

제약: 원본의 엔진과 호환되어야 합니다.

유형: 문자열

필수 항목 여부: 예

SnapshotIdentifier

복원에 사용할 스냅샷 또는 클러스터 스냅샷의 식별자입니다.

이름 또는 Amazon 리소스 이름(ARN)을 사용하여 클러스터 스냅샷을 지정할 수 있습니다. 그러나 스냅샷을 지정할 때는 ARN만 사용해야 합니다.

제약 조건:

- 기존 스냅샷의 식별자와 일치해야 합니다.

유형: 문자열

필수 항목 여부: 예

AvailabilityZones.AvailabilityZone.N

복원된 DB 클러스터의 인스턴스를 생성할 수 있는 Amazon EC2 가용 영역 목록을 알려 줍니다.

유형: 문자열 배열

필수 여부: 아니요

DBClusterParameterGroupName

이 DB 클러스터와 연결할 DB 클러스터 파라미터 그룹의 이름입니다.

유형: 문자열입니다. 필수 항목 여부: 아니요

이 인수를 생략하면 기본 DB 클러스터 파라미터 그룹이 사용됩니다. 제공된 경우 기존 기본 DB 클러스터 파라미터 그룹의 이름과 일치해야 합니다. 문자열은 1~255개의 문자, 숫자 또는 하이픈으로 구성되어야 합니다. 첫 번째 문자는 문자이어야 하며 하이픈으로 끝나거나 두 개의 연속된 하이픈을 포함할 수 없습니다.

유형: 문자열

필수 항목 여부: 아니요

DBSubnetGroupName

새 클러스터에 사용할 서브넷 그룹의 이름입니다.

제약: 입력 시 기존의 `DBSubnetGroup` 이름과 일치해야 합니다.

예시: `mySubnetgroup`

유형: 문자열

필수 항목 여부: 아니요

DeletionProtection

이 클러스터를 삭제할 수 있는지 없는지를 지정합니다. DeletionProtection이 활성화되면 클러스터를 수정하고 DeletionProtection 를 비활성화하지 않는 한 클러스터를 삭제할 수 없습니다. DeletionProtection은 클러스터가 실수로 삭제되지 않도록 보호합니다.

유형: 부울

필수 항목 여부: 아니요

EnableCloudwatchLogsExports.member.N

Amazon CloudWatch Logs 로 내보내려면 사용 설정해야 하는 로그 유형 목록입니다.

유형: 문자열 배열

필수 여부: 아니요

EngineVersion

새 클러스터에 사용할 데이터베이스 엔진의 버전입니다.

유형: 문자열

필수 항목 여부: 아니요

KmsKeyId

DB 스냅샷 또는 클러스터 스냅샷에서 암호화된 클러스터를 복원할 때 사용할 AWS KMS 키 식별자입니다.

AWS KMS 키 식별자는 AWS KMS 암호화 키의 Amazon 리소스 이름(ARN)입니다. 새 클러스터를 암호화하는 데 사용되는 AWS KMS 암호화 키를 소유 AWS 계정 한 동일한 로 클러스터를 복원하는 경우 AWS KMS 암호화 AWS KMS 키에 ARN 대신 키 별칭을 사용할 수 있습니다.

KmsKeyId 파라미터 값을 지정하지 않으면 다음과 같이 진행됩니다.

- 의 스냅샷 또는 클러스터 스냅샷SnapshotIdentifier이 암호화된 경우 복원된 클러스터는 스냅샷 또는 클러스터 스냅샷을 암호화하는 데 사용된 AWS KMS 키를 사용하여 암호화됩니다.
- SnapshotIdentifier의 스냅샷 또는 클러스터 스냅샷이 암호화되어 있지 않으면 복원된 DB 클러스터는 암호화되지 않습니다.

유형: 문자열

필수 항목 여부: 아니요

Port

새 클러스터에서 연결을 허용하는 포트 번호입니다.

제약 조건: 값이 1150~65535여야 합니다.

기본값: 원래의 클러스터와 동일한 포트입니다.

유형: 정수

필수 항목 여부: 아니요

StorageType

DB 클러스터와 연결할 스토리지 유형입니다.

Amazon DocumentDB 클러스터의 스토리지 유형에 관한 자세한 내용은 Amazon DocumentDB 개발자 안내서의 클러스터 스토리지 구성을 참조하세요.

스토리지 유형에 유효한 값 - `standard | iopt1`

기본값은 `standard` 입니다.

유형: 문자열

필수 항목 여부: 아니요

Tags.Tag.N

복원된 클러스터에 할당할 태그입니다.

타입: Tag객체 배열

필수: 아니요

VpcSecurityGroupIds.VpcSecurityGroupId.N

새 클러스터가 속하게 될 Virtual Private Cloud(VPC) 보안 그룹 목록입니다.

유형: 문자열 배열

필수 여부: 아니요

응답 요소

서비스에서 다음 요소를 반환합니다.

DBCluster

클러스터에 관한 자세한 정보입니다.

타입: DBCluster 객체

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하세요.

DBClusterAlreadyExistsFault

해당 식별자를 사용하는 클러스터가 이미 있습니다.

HTTP 상태 코드: 400

DBClusterQuotaExceededFault

클러스터의 최대 허용 할당량에 도달했기 때문에 클러스터를 생성할 수 없습니다.

HTTP 상태 코드: 403

DBClusterSnapshotNotFoundFault

`DBClusterSnapshotIdentifier`는 기존 클러스터 스냅샷을 참조하지 않습니다.

HTTP 상태 코드: 404

DBSnapshotNotFound

`DBSnapshotIdentifier`는 기존 스냅샷을 참조하지 않습니다.

HTTP 상태 코드: 404

DBSubnetGroupNotFoundFault

`DBSubnetGroupName`는 기존 서브넷 그룹을 참조하지 않습니다.

HTTP 상태 코드: 404

DBSubnetGroupNotFoundFault

`DBSubnetGroupName`는 기존 서브넷 그룹을 참조하지 않습니다.

HTTP 상태 코드: 404

InsufficientDBClusterCapacityFault

현재의 작업을 하기에는 클러스터의 용량이 부족합니다.

HTTP 상태 코드: 403

InsufficientStorageClusterCapacity

현재의 작업에 사용할 스토리지가 부족합니다. 사용 가능한 스토리지가 더 많은 다른 가용 영역을 사용하도록 서브넷 그룹을 업데이트하여 이 오류를 해결할 수 있습니다.

HTTP 상태 코드: 400

InvalidDBClusterSnapshotStateFault

제공된 값은 유효한 클러스터 스냅샷 상태가 아닙니다.

HTTP 상태 코드: 400

InvalidDBSnapshotState

스냅샷의 상태로 인해 삭제할 수 없습니다.

HTTP 상태 코드: 400

InvalidRestoreFault

Virtual Private Cloud(VPC) 백업에서 VPC가 아닌 DB 인스턴스로 복원할 수 없습니다.

HTTP 상태 코드: 400

InvalidSubnet

요청한 서브넷이 올바르지 않거나 공통 Virtual Private Cloud(VPC)에 모두 포함되지 않은 여러 서브넷이 요청되었습니다.

HTTP 상태 코드: 400

InvalidVPCNetworkStateFault

서브넷 그룹이 생성된 후에는 변경 사항으로 인해 모든 가용 영역에 적용되지 않습니다.

HTTP 상태 코드: 400

KMSKeyNotAccessibleFault

AWS KMS 키에 액세스할 때 오류가 발생했습니다.

HTTP 상태 코드: 400

StorageQuotaExceeded

요청으로 인해 모든 인스턴스에서 사용 가능한 스토리지 허용량을 초과하게 됩니다.

HTTP 상태 코드: 400

StorageQuotaExceeded

요청으로 인해 모든 인스턴스에서 사용 가능한 스토리지 허용량을 초과하게 됩니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go v2용 SDK
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

RestoreDBClusterToPointInTime

서비스: Amazon DocumentDB (with MongoDB compatibility)

클러스터를 임의의 시점으로 복원합니다. 사용자는 `LatestRestorableTime` 이전의 최대 `BackupRetentionPeriod`일까지 원하는 시점으로 복원할 수 있습니다. 원래의 클러스터와 동일한 구성으로 원본 클러스터에서 대상 클러스터가 생성됩니다. 단, 새 클러스터가 기본 보안 그룹으로 생성된 경우는 예외입니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 관한 정보는 공통 파라미터를 참조하십시오.

DBClusterIdentifier

생성할 새 클러스터의 이름입니다.

제약 조건:

- 1~63자의 문자, 숫자 또는 하이픈으로 구성되어야 합니다.
- 첫 자는 문자여야 합니다.
- 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다.

유형: 문자열

필수 항목 여부: 예

SourceDBClusterIdentifier

복원할 소스 클러스터의 식별자입니다.

제약 조건:

- 기존 `DBCluster`의 식별자와 일치해야 합니다.

유형: 문자열

필수 항목 여부: 예

DBSubnetGroupName

새 클러스터에 사용할 서브넷 그룹 이름입니다.

제약: 입력 시 기존의 `DBSubnetGroup` 이름과 일치해야 합니다.

예시: `mySubnetgroup`

유형: 문자열

필수 항목 여부: 아니요

DeletionProtection

이 클러스터를 삭제할 수 있는지 없는지를 지정합니다. DeletionProtection이 활성화되면 클러스터를 수정하고 DeletionProtection 를 비활성화하지 않는 한 클러스터를 삭제할 수 없습니다. DeletionProtection은 클러스터가 실수로 삭제되지 않도록 보호합니다.

유형: 부울

필수 항목 여부: 아니요

EnableCloudwatchLogsExports.member.N

Amazon CloudWatch Logs 로 내보내려면 사용 설정해야 하는 로그 유형 목록입니다.

유형: 문자열 배열

필수 여부: 아니요

KmsKeyId

암호화된 클러스터에서 암호화된 클러스터를 복원할 때 사용할 AWS KMS 키 식별자입니다.

AWS KMS 키 식별자는 AWS KMS 암호화 키의 Amazon 리소스 이름(ARN)입니다. 새 클러스터를 암호화하는 데 사용되는 AWS KMS 암호화 키를 소유 AWS 계정 한 동일한 로 클러스터를 복원하는 경우 AWS KMS 암호화 AWS KMS 키에 ARN 대신 키 별칭을 사용할 수 있습니다.

새 클러스터로 복원하고 소스 클러스터를 암호화하는 데 사용되는 AWS KMS 키와 다른 AWS KMS 키를 사용하여 새 클러스터를 암호화할 수 있습니다. 새 DB 클러스터는 KmsKeyId 파라미터로 식별되는 AWS KMS 키로 암호화됩니다.

KmsKeyId 파라미터 값을 지정하지 않으면 다음과 같이 진행됩니다.

- 클러스터가 암호화된 경우 복원된 클러스터는 소스 클러스터를 암호화하는 데 사용된 AWS KMS 키를 사용하여 암호화됩니다.
- 클러스터가 암호화되어 있지 않으면 복원된 클러스터도 암호화되지 않습니다.

DBClusterIdentifier가 암호화되지 않은 DB 클러스터를 가리키는 경우, 복원 요청이 거부됩니다.

유형: 문자열

필수 항목 여부: 아니요

Port

새 클러스터에서 연결을 허용하는 포트 번호입니다.

제약 조건: 값이 1150~65535여야 합니다.

기본값: 엔진의 기본 포트입니다.

유형: 정수

필수 항목 여부: 아니요

RestoreToTime

클러스터를 복원할 날짜 및 시간입니다.

유효한 값: 협정 세계시(UTC) 형식의 시간.

제약 조건:

- 인스턴스의 최근 복원 가능 시간보다 이전이어야 합니다.
- UseLatestRestorableTime 파라미터를 제공하지 않은 경우에 지정해야 합니다.
- UseLatestRestorableTime 파라미터가 true인 경우에는 지정할 수 없습니다.
- RestoreType 파라미터가 copy-on-write인 경우에는 지정할 수 없습니다.

예시: 2015-03-07T23:45:00Z

유형: 타임스탬프

필수 여부: 아니요

RestoreType

수행할 복원의 유형입니다. 다음 값 중 하나를 지정할 수 있습니다.

- full-copy - 새 DB 클러스터가 소스 DB 클러스터의 전체 복사로서 복구됩니다.
- copy-on-write - 새 DB 클러스터가 소스 DB 클러스터의 복제로서 복구됩니다.

제약: 소스 DB 클러스터의 엔진 버전이 1.11 이하인 경우 copy-on-write를 지정할 수 없습니다.

RestoreType 값을 지정하지 않으면 새 DB 클러스터가 소스 DB 클러스터의 전체 복사로서 복구됩니다.

유형: 문자열

필수 항목 여부: 아니요

StorageType

DB 클러스터와 연결할 스토리지 유형입니다.

Amazon DocumentDB 클러스터의 스토리지 유형에 관한 자세한 내용은 Amazon DocumentDB 개발자 안내서의 클러스터 스토리지 구성을 참조하세요.

스토리지 유형에 유효한 값 - `standard | iopt1`

기본값은 `standard` 입니다.

유형: 문자열

필수 항목 여부: 아니요

Tags.Tag.N

복원된 클러스터에 할당할 태그입니다.

타입: Tag객체 배열

필수: 아니요

UseLatestRestorableTime

클러스터를 복원 가능한 마지막 백업 시간으로 복원하려면 이 값을 `true`로 설정하고, 그렇지 않으면 `false`로 설정합니다.

기본값: `false`

제약 조건: `RestoreToTime` 파라미터를 제공한 경우에는 지정할 수 없습니다.

유형: 부울

필수 항목 여부: 아니요

VpcSecurityGroupIds.VpcSecurityGroupId.N

새 클러스터가 속해 있는 VPC 보안 그룹의 목록입니다.

유형: 문자열 배열

필수 여부: 아니요

응답 요소

서비스에서 다음 요소를 반환합니다.

DBCluster

클러스터에 관한 자세한 정보입니다.

타입: DBCluster 객체

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하세요.

DBClusterAlreadyExistsFault

해당 식별자를 사용하는 클러스터가 이미 있습니다.

HTTP 상태 코드: 400

DBClusterNotFoundFault

`DBClusterIdentifier`는 기존 클러스터를 참조하지 않습니다.

HTTP 상태 코드: 404

DBClusterQuotaExceededFault

클러스터의 최대 허용 할당량에 도달했기 때문에 클러스터를 생성할 수 없습니다.

HTTP 상태 코드: 403

DBClusterSnapshotNotFoundFault

`DBClusterSnapshotIdentifier`는 기존 클러스터 스냅샷을 참조하지 않습니다.

HTTP 상태 코드: 404

DBSubnetGroupNotFoundFault

`DBSubnetGroupName`는 기존 서브넷 그룹을 참조하지 않습니다.

HTTP 상태 코드: 404

InsufficientDBClusterCapacityFault

현재의 작업을 하기에는 클러스터의 용량이 부족합니다.

HTTP 상태 코드: 403

InsufficientStorageClusterCapacity

현재의 작업에 사용할 스토리지가 부족합니다. 사용 가능한 스토리지가 더 많은 다른 가용 영역을 사용하도록 서브넷 그룹을 업데이트하여 이 오류를 해결할 수 있습니다.

HTTP 상태 코드: 400

InvalidDBClusterSnapshotStateFault

제공된 값은 유효한 클러스터 스냅샷 상태가 아닙니다.

HTTP 상태 코드: 400

InvalidDBClusterStateFault

클러스터가 유효한 상태가 아닙니다.

HTTP 상태 코드: 400

InvalidDBSnapshotState

스냅샷의 상태로 인해 삭제할 수 없습니다.

HTTP 상태 코드: 400

InvalidRestoreFault

Virtual Private Cloud(VPC) 백업에서 VPC가 아닌 DB 인스턴스로 복원할 수 없습니다.

HTTP 상태 코드: 400

InvalidSubnet

요청한 서브넷이 올바르지 않거나 공통 Virtual Private Cloud(VPC)에 모두 포함되지 않은 여러 서브넷이 요청되었습니다.

HTTP 상태 코드: 400

InvalidVPCNetworkStateFault

서브넷 그룹이 생성된 후에는 변경 사항으로 인해 모든 가용 영역에 적용되지 않습니다.

HTTP 상태 코드: 400

KMSKeyNotAccessibleFault

AWS KMS 키에 액세스할 때 오류가 발생했습니다.

HTTP 상태 코드: 400

StorageQuotaExceeded

요청으로 인해 모든 인스턴스에서 사용 가능한 스토리지 허용량을 초과하게 됩니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go v2용 SDK
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

StartDBCluster

서비스: Amazon DocumentDB (with MongoDB compatibility)

`DBClusterIdentifier`에서 지정한 중지된 클러스터를 다시 시작합니다. 자세한 내용은 Amazon DocumentDB 클러스터 중단 및 시작 섹션을 참조하십시오.

요청 파라미터

모든 작업에 공통되는 파라미터에 관한 정보는 범용 파라미터를 참조하십시오.

DBClusterIdentifier

재시작 클러스터의 식별자입니다. 예시: `docdb-2019-05-28-15-24-52`

유형: 문자열

필수 항목 여부: 예

응답 요소

서비스에서 다음 요소를 반환합니다.

DBCluster

클러스터에 관한 자세한 정보입니다.

타입: DBCluster 객체

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하세요.

DBClusterNotFoundFault

`DBClusterIdentifier`은 기존 클러스터를 참조하지 않습니다.

HTTP 상태 코드: 404

InvalidDBClusterStateFault

DB 클러스터가 유효한 상태가 아닙니다.

HTTP 상태 코드: 400

InvalidDBInstanceState

지정된 인스턴스가 사용 가능한 상태가 아닙니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go v2용 SDK
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

StopDBCluster

서비스: Amazon DocumentDB (with MongoDB compatibility)

`DBClusterIdentifier`에서 지정한 실행 중인 클러스터를 중지합니다. 클러스터가 사용 가능 상태여야 합니다. 자세한 내용은 Amazon DocumentDB 클러스터 중단 및 시작 섹션을 참조하십시오.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 내용은 공통 파라미터를 참조하십시오.

DBClusterIdentifier

중지할 클러스터의 식별자입니다. 예시: `docdb-2019-05-28-15-24-52`

유형: 문자열

필수 항목 여부: 예

응답 요소

서비스에서 다음 요소를 반환합니다.

DBCluster

클러스터에 관한 자세한 정보입니다.

타입: DBCluster 객체

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하세요.

DBClusterNotFoundFault

`DBClusterIdentifier`은 기존 클러스터를 참조하지 않습니다.

HTTP 상태 코드: 404

InvalidDBClusterStateFault

DB 클러스터가 유효한 상태가 아닙니다.

HTTP 상태 코드: 400

InvalidDBInstanceState

지정된 인스턴스가 사용 가능한 상태가 아닙니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

SwitchoverGlobalCluster

서비스: Amazon DocumentDB (with MongoDB compatibility)

지정된 보조 Amazon DocumentDB 클러스터를 글로벌 데이터베이스 클러스터의 새 기본 Amazon DocumentDB 클러스터로 전환합니다.

요청 파라미터

모든 작업에 공통되는 파라미터에 대한 내용은 공통 파라미터를 참조하십시오.

GlobalClusterIdentifier

전환할 Amazon DocumentDB 글로벌 데이터베이스 클러스터의 식별자입니다. 식별자는 클러스터가 생성될 때 사용자가 할당한 고유 키입니다. 즉, 글로벌 클러스터의 이름입니다. 이 파라미터는 대/소문자를 구분하지 않습니다.

제약 조건:

- 기존 글로벌 클러스터(Amazon DocumentDB 글로벌 데이터베이스)의 식별자와 일치해야 합니다.
- 최소 길이는 1이고, 최대 길이는 255.

패턴: `[A-Za-z][0-9A-Za-z-:._]*`

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 255.

패턴: `[A-Za-z][0-9A-Za-z-:._]*`

필수 여부: 예

TargetDbClusterIdentifier

글로벌 데이터베이스 클러스터의 새 기본으로 승격할 보조 Amazon DocumentDB 클러스터의 식별자입니다. Amazon DocumentDB가 해당 리전에서 클러스터를 찾을 수 있도록 식별자에 Amazon 리소스 이름(ARN)을 사용합니다. AWS

제약 조건:

- 기존 보조 클러스터의 식별자와 일치해야 합니다.
- 최소 길이는 1이고, 최대 길이는 255.

패턴: `[A-Za-z][0-9A-Za-z-:._]*`

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 255.

패턴: `[A-Za-z][0-9A-Za-z-:._]*`

필수 여부: 예

응답 요소

서비스에서 다음 요소를 반환합니다.

GlobalCluster

Amazon DocumentDB 글로벌 클러스터를 나타내는 데이터 유형.

타입: GlobalCluster 객체

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하세요.

DBClusterNotFoundFault

`DBClusterIdentifier`은 기존 클러스터를 참조하지 않습니다.

HTTP 상태 코드: 404

GlobalClusterNotFoundFault

`GlobalClusterIdentifier`는 기존 글로벌 클러스터를 참조하지 않습니다.

HTTP 상태 코드: 404

InvalidDBClusterStateFault

DB 클러스터가 유효한 상태가 아닙니다.

HTTP 상태 코드: 400

InvalidGlobalClusterStateFault

클러스터가 이 상태인 동안에는 요청된 작업을 수행할 수 없습니다.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

Amazon DocumentDB Elastic Clusters

Amazon DocumentDB Elastic Clusters는 다음 작업을 지원합니다.

- ApplyPendingMaintenanceAction
- CopyClusterSnapshot
- CreateCluster
- CreateClusterSnapshot
- DeleteCluster
- DeleteClusterSnapshot
- GetCluster
- GetClusterSnapshot
- GetPendingMaintenanceAction
- ListClusters
- ListClusterSnapshots
- ListPendingMaintenanceActions
- ListTagsForResource
- RestoreClusterFromSnapshot

- StartCluster
- StopCluster
- TagResource
- UntagResource
- UpdateCluster

ApplyPendingMaintenanceAction

서비스: Amazon DocumentDB Elastic Clusters

리소스에 적용할 보류 중인 유지 관리 작업의 유형입니다.

Request Syntax

```
POST /pending-action HTTP/1.1
Content-type: application/json

{
   "applyAction": "string",
   "applyOn": "string",
   "optInType": "string",
   "resourceArn": "string"
}
```

URI 요청 파라미터

요청은 URI 파라미터를 사용하지 않습니다.

요청 본문

요청은 JSON 형식으로 다음 데이터를 받습니다.

applyAction

리소스에 적용할 보류 중인 유지 관리 작업입니다.

유효 작업은 다음과 같습니다.

- ENGINE_UPDATE
- ENGINE_UPGRADE
- SECURITY_UPDATE
- OS_UPDATE
- MASTER_USER_PASSWORD_UPDATE

유형: 문자열

길이 제약 조건: 최소 길이는 1입니다. 최대 길이는 256.

필수 여부: 예

optInType

옵트인 요청의 유형을 지정하거나 옵트인 요청을 실행 취소하는 값입니다. IMMEDIATE 유형의 옵트인 요청은 실행 취소할 수 없습니다.

타입: 문자열

유효 값: IMMEDIATE | NEXT_MAINTENANCE | APPLY_ON | UNDO_OPT_IN

필수 사항 여부: 예

resourceArn

보류 중인 유지 관리 작업이 적용되는 리소스의 Amazon DocumentDB Amazon 리소스 이름(ARN)입니다.

유형: 문자열

길이 제약 조건: 최소 길이는 1입니다. 최대 길이는 256.

필수 여부: 예

applyOn

보류 중인 유지 관리 작업을 적용할 특정 날짜입니다. opt-in-type이 인 경우 필수입니다APPLY_ON. 형식: yyyy/MM/dd HH:mm-yyyy/MM/dd HH:mm

유형: 문자열

길이 제약 조건: 최소 길이는 1입니다. 최대 길이는 256.

필수 여부: 아니요

응답 구문

```
HTTP/1.1 200
Content-type: application/json

{
   "resourcePendingMaintenanceAction": {
      "pendingMaintenanceActionDetails": [
         {
            "action": "string",
```

```
            "autoAppliedAfterDate": "string",
            "currentApplyDate": "string",
            "description": "string",
            "forcedApplyDate": "string",
            "optInStatus": "string"
         }
      ],
      "resourceArn": "string"
   }
}
```

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 반송합니다.

다음 데이터는 서비스에 의해 JSON 형식으로 반환됩니다.

resourcePendingMaintenanceAction

적용 중인 보류 중인 유지 관리 작업의 출력입니다.

타입: ResourcePendingMaintenanceAction 객체

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하십시오.

AccessDeniedException

작업을 수행할 권한이 충분하지 않을 때 발생하는 예외.

HTTP 상태 코드: 403

ConflictException

액세스 충돌이 발생했습니다.

HTTP 상태 코드: 409

InternalServerException

내부 서버 오류가 발생했습니다.

HTTP 상태 코드: 500

ResourceNotFoundException

지정된 리소스를 찾을 수 없습니다.

HTTP 상태 코드: 404

ThrottlingException

요청 제한으로 인해 요청이 거부되면 ThrottlingException이 발생합니다.

HTTP 상태 코드: 429

ValidationException

유효성 검사 예외를 정의하는 구조.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go v2용 SDK
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

CopyClusterSnapshot

서비스: Amazon DocumentDB Elastic Clusters

탄력적 클러스터의 스냅샷을 복사합니다.

Request Syntax

```
POST /cluster-snapshot/snapshotArn/copy HTTP/1.1
Content-type: application/json

{
   "copyTags": boolean,
   "kmsKeyId": "string",
   "tags": {
      "string" : "string"
   },
   "targetSnapshotName": "string"
}
```

URI 요청 파라미터

요청은 다음 URI 파라미터를 사용합니다.

snapshotArn

탄력적 클러스터 스냅샷의 Amazon 리소스 이름(ARN) 식별자.

필수 여부: 예

요청 본문

요청은 JSON 형식으로 다음 데이터를 받습니다.

targetSnapshotName

소스 클러스터 스냅샷에서 생성할 새 탄력적 클러스터 스냅샷의 식별자입니다. 이 파라미터는 대소문자를 구분하지 않습니다.

제약 조건:

- 1~63자의 문자, 숫자 또는 하이픈으로 구성되어야 합니다.
- 첫 자는 문자여야 합니다.

- 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다.

예시: `elastic-cluster-snapshot-5`

유형: 문자열

길이 제약: 최소 길이 1. 최대 길이 63.

필수 여부: 예

copyTags

원본 클러스터 스냅샷의 모든 태그를 대상 탄력적 클러스터 스냅샷으로 복사하려면 `true`로 설정합니다. 기본값은 `false`입니다.

유형: 부울

필수 항목 여부: 아니요

kmsKeyId

암호화된 탄력적 클러스터 스냅샷의 AWS KMS 키 ID입니다. AWS KMS 키 ID는 Amazon 리소스 이름(ARN), AWS KMS 키 식별자 또는 AWS KMS 암호화 키의 AWS KMS 키 별칭입니다.

AWS 계정에서 암호화된 탄력적 클러스터 스냅샷을 복사하는 경우의 값을 지정`KmsKeyId`하여 새 AWS S KMS 암호화 키로 복사본을 암호화할 수 있습니다. `KmsKeyId` 값을 지정하지 않으면 탄력적 클러스터 스냅샷의 사본을 원본 탄력적 스냅샷과 동일한 `AWS` KMS 키로 암호화합니다.

암호화되지 않은 탄력적 클러스터 스냅샷의 복사하고 `KmsKeyId` 파라미터에 대한 값을 지정하려고 시도하면 오류가 반환됩니다.

유형: 문자열

필수 항목 여부: 아니요

tags

탄력적 클러스터 스냅샷에 할당할 태그입니다.

유형: 문자열 간 맵

키 길이 제약 조건: 최소 길이는 1. 최대 길이 128.

키 패턴: `(?!aws:)[a-zA-Z+-=._:/]+`

값 길이 제약 조건: 최소 길이는 0입니다. 최대 길이는 256입니다.

필수 여부: 아니요

응답 구문

```
HTTP/1.1 200
Content-type: application/json

{
   "snapshot": {
      "adminUserName": "string",
      "clusterArn": "string",
      "clusterCreationTime": "string",
      "kmsKeyId": "string",
      "snapshotArn": "string",
      "snapshotCreationTime": "string",
      "snapshotName": "string",
      "snapshotType": "string",
      "status": "string",
      "subnetIds": [ "string" ],
      "vpcSecurityGroupIds": [ "string" ]
   }
}
```

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 반송합니다.

다음 데이터는 서비스에 의해 JSON 형식으로 반환됩니다.

snapshot

특정 탄력적 클러스터 스냅샷에 대한 정보를 반환합니다.

타입: ClusterSnapshot 객체

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하십시오.

AccessDeniedException

작업을 수행할 권한이 충분하지 않을 때 발생하는 예외.

HTTP 상태 코드: 403

ConflictException

액세스 충돌이 발생했습니다.

HTTP 상태 코드: 409

InternalServerException

내부 서버 오류가 발생했습니다.

HTTP 상태 코드: 500

ResourceNotFoundException

지정된 리소스를 찾을 수 없습니다.

HTTP 상태 코드: 404

ServiceQuotaExceededException

작업에 대한 서비스 할당량을 초과했습니다.

HTTP 상태 코드: 402

ThrottlingException

요청 조절로 인해 요청이 거부되면 ThrottlingException이 발생합니다.

HTTP 상태 코드: 429

ValidationException

유효성 검사 예외를 정의하는 구조.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++

- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

CreateCluster

서비스: Amazon DocumentDB Elastic Clusters

Amazon DocumentDB 탄력적 클러스터를 새로 생성하고 클러스터 구조를 반환합니다.

Request Syntax

```
POST /cluster HTTP/1.1
Content-type: application/json

{
   "adminUserName": "string",
   "adminUserPassword": "string",
   "authType": "string",
   "backupRetentionPeriod": number,
   "clientToken": "string",
   "clusterName": "string",
   "kmsKeyId": "string",
   "preferredBackupWindow": "string",
   "preferredMaintenanceWindow": "string",
   "shardCapacity": number,
   "shardCount": number,
   "shardInstanceCount": number,
   "subnetIds": [ "string" ],
   "tags": {
      "string" : "string"
   },
   "vpcSecurityGroupIds": [ "string" ]
}
```

URI 요청 파라미터

요청은 URI 파라미터를 사용하지 않습니다.

요청 본문

요청은 JSON 형식으로 다음 데이터를 받습니다.

adminUserName

Amazon DocumentDB 탄력적 클러스터 관리자의 이름.

제약 조건:

- 1~63자의 문자 또는 숫자여야 합니다.
- 첫 번째 자리는 문자여야 합니다.
- 예약어가 될 수는 없습니다.

유형: 문자열

필수 항목 여부: 예

adminUserPassword

Amazon DocumentDB 탄력적 클러스터 관리자의 암호. 암호는 일체의 인쇄 가능한 ASCII 문자를 포함할 수 있습니다.

제약 조건:

- 8자~100자여야 합니다.
- 포워드 슬래시(/), 큰따옴표(“) 또는 “at” 기호(@)를 포함할 수 없습니다.

유형: 문자열

필수 항목 여부: 예

authType

탄력적 클러스터에 액세스하는 데 사용되는 암호를 가져올 위치를 결정하는 데 사용되는 인증 유형입니다. 유효한 형식은 `PLAIN_TEXT` 및 `SECRET_ARN`입니다.

타입: 문자열

유효 값: `PLAIN_TEXT | SECRET_ARN`

필수 사항 여부: 예

clusterName

새 탄력적 클러스터의 이름. 이 파라미터는 소문자 문자열로 저장됩니다.

제약 조건:

- 1~63자의 문자, 숫자 또는 하이픈으로 구성되어야 합니다.
- 첫 자는 문자여야 합니다.
- 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다.

예: `my-cluster`

유형: 문자열

필수 항목 여부: 예

shardCapacity

각 탄력적 클러스터 샤드에 할당된 vCPU 수 최대 값은 64입니다. 허용되는 값은 2, 4, 8, 16, 32, 64입니다.

유형: 정수

필수 여부: 예

shardCount

탄력적 클러스터에 할당된 샤드 수. 최대 값은 32입니다.

유형: 정수

필수 여부: 예

backupRetentionPeriod

자동 스냅샷이 보관되는 일수입니다.

유형: 정수

필수 항목 여부: 아니요

clientToken

탄력적 클러스터의 클라이언트 토큰.

유형: 문자열

필수 항목 여부: 아니요

kmsKeyId

새 탄력적 클러스터를 암호화하는 데 사용할 KMS 키 식별자.

KMS 키 식별자는 KMS 암호화 키의 Amazon 리소스 이름(ARN)입니다. 이 KMS 암호화 키를 소유하고 있는 동일한 Amazon 계정을 사용하여 클러스터를 생성하는 경우, ARN 대신 KMS 키 별칭을 KMS 암호화 키로 사용할 수 있습니다.

암호화 키가 지정되지 않은 경우 Amazon DocumentDB는 KMS가 사용자 계정에 대해 생성하는 기본 암호화 키를 사용합니다. 계정에는 Amazon 리전마다 다른 기본 암호화 키가 있습니다.

유형: 문자열

필수 항목 여부: 아니요

preferredBackupWindow

자동 백업이 활성화된 경우 자동 백업이 생성되는 일일 시간 범위이며, `backupRetentionPeriod` 속성에 의해 결정됩니다.

유형: 문자열

필수 항목 여부: 아니요

preferredMaintenanceWindow

시스템 유지 관리를 실행할 수 있는 주 단위 기간(UTC, 협정 세계시)입니다.

형식: `ddd:hh24:mi-ddd:hh24:mi`

기본값: 각에 대해 8시간 블록 시간 중에서 임의로 선택되는 30분 기간으로 AWS 리전, 요일에 무작위로 발생합니다.

유효한 요일: 월, 화, 수, 목, 금, 토, 일

제약 조건: 최소 30분의 기간.

유형: 문자열

필수 항목 여부: 아니요

shardInstanceCount

탄력적 클러스터의 모든 샤드에 적용되는 복제본 인스턴스 수입니다. `shardInstanceCount` 값이 1이면 라이터 인스턴스가 하나라는 의미이며, 추가 인스턴스는 읽기에 사용할 수 있고 가용성을 개선하는 복제본입니다.

유형: 정수

필수 항목 여부: 아니요

subnetIds

새 탄력적 클러스터의 Amazon EC2 서브넷 ID.

유형: 문자열 배열

필수 여부: 아니요

tags

새 탄력적 클러스터에 할당할 태그.

유형: 문자열 간 맵

키 길이 제약 조건: 최소 길이는 1. 최대 길이 128.

키 패턴: `(?!aws:)[a-zA-Z+-=._:/]+`

값 길이 제약 조건: 최소 길이는 0입니다. 최대 길이는 256입니다.

필수 여부: 아니요

vpcSecurityGroupIds

새 탄력적 클러스터에 연결할 EC2 VPC 보안 그룹의 목록.

유형: 문자열 배열

필수 항목 여부: 아니요

응답 구문

```
HTTP/1.1 200
Content-type: application/json

{
   "cluster": {
      "adminUserName": "string",
      "authType": "string",
      "backupRetentionPeriod": number,
      "clusterArn": "string",
      "clusterEndpoint": "string",
      "clusterName": "string",
      "createTime": "string",
      "kmsKeyId": "string",
      "preferredBackupWindow": "string",
      "preferredMaintenanceWindow": "string",
      "shardCapacity": number,
      "shardCount": number,
      "shardInstanceCount": number,
```

```
      "shards": [
         {
            "createTime": "string",
            "shardId": "string",
            "status": "string"
         }
      ],
      "status": "string",
      "subnetIds": [ "string" ],
      "vpcSecurityGroupIds": [ "string" ]
   }
}
```

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 반송합니다.

다음 데이터는 서비스에 의해 JSON 형식으로 반환됩니다.

cluster

생성된 새 탄력적 클러스터.

타입: Cluster 객체

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하십시오.

AccessDeniedException

작업을 수행할 권한이 충분하지 않을 때 발생하는 예외.

HTTP 상태 코드: 403

ConflictException

액세스 충돌이 발생했습니다.

HTTP 상태 코드: 409

InternalServerException

내부 서버 오류가 발생했습니다.

HTTP 상태 코드: 500

ServiceQuotaExceededException

작업에 대한 서비스 할당량을 초과했습니다.

HTTP 상태 코드: 402

ThrottlingException

요청 조절로 인해 요청이 거부되면 ThrottlingException이 발생합니다.

HTTP 상태 코드: 429

ValidationException

유효성 검사 예외를 정의하는 구조.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

CreateClusterSnapshot

서비스: Amazon DocumentDB Elastic Clusters

탄력적 클러스터의 스냅샷을 생성합니다.

Request Syntax

```
POST /cluster-snapshot HTTP/1.1
Content-type: application/json

{
   "clusterArn": "string",
   "snapshotName": "string",
   "tags": {
      "string" : "string"
   }
}
```

URI 요청 파라미터

요청은 URI 파라미터를 사용하지 않습니다.

요청 본문

요청은 JSON 형식으로 다음 데이터를 받습니다.

clusterArn

스냅샷을 생성하려는 탄력적 클러스터의 ARN 식별자.

유형: 문자열

필수 항목 여부: 예

snapshotName

새 탄력적 클러스터 스냅샷의 이름.

유형: 문자열

길이 제약: 최소 길이 1. 최대 길이 63.

필수 여부: 예

tags

새 탄력적 클러스터 스냅샷에 할당할 태그.

유형: 문자열 간 맵

키 길이 제약 조건: 최소 길이는 1. 최대 길이 128.

키 패턴: `(?!aws:)[a-zA-Z+-=._:/]+`

값 길이 제약 조건: 최소 길이는 0입니다. 최대 길이는 256입니다.

필수 여부: 아니요

응답 구문

```
HTTP/1.1 200
Content-type: application/json

{
   "snapshot": {
      "adminUserName": "string",
      "clusterArn": "string",
      "clusterCreationTime": "string",
      "kmsKeyId": "string",
      "snapshotArn": "string",
      "snapshotCreationTime": "string",
      "snapshotName": "string",
      "snapshotType": "string",
      "status": "string",
      "subnetIds": [ "string" ],
      "vpcSecurityGroupIds": [ "string" ]
   }
}
```

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 반송합니다.

다음 데이터는 서비스에 의해 JSON 형식으로 반환됩니다.

snapshot

새 탄력적 클러스터 스냅샷에 대한 정보를 반환합니다.

타입: ClusterSnapshot 객체

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하십시오.

AccessDeniedException

작업을 수행할 권한이 충분하지 않을 때 발생하는 예외.

HTTP 상태 코드: 403

ConflictException

액세스 충돌이 발생했습니다.

HTTP 상태 코드: 409

InternalServerException

내부 서버 오류가 발생했습니다.

HTTP 상태 코드: 500

ResourceNotFoundException

지정된 리소스를 찾을 수 없습니다.

HTTP 상태 코드: 404

ServiceQuotaExceededException

작업에 대한 서비스 할당량을 초과했습니다.

HTTP 상태 코드: 402

ThrottlingException

요청 조절로 인해 요청이 거부되면 ThrottlingException이 발생합니다.

HTTP 상태 코드: 429

ValidationException

유효성 검사 예외를 정의하는 구조.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

DeleteCluster

서비스: Amazon DocumentDB Elastic Clusters

탄력적 클러스터를 삭제합니다.

Request Syntax

```
DELETE /cluster/clusterArn HTTP/1.1
```

URI 요청 파라미터

요청은 다음 URI 파라미터를 사용합니다.

clusterArn

삭제하려는 탄력적 클러스터의 ARN 식별자.

필수 여부: 예

Request Body

해당 요청에는 본문이 없습니다.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "cluster": {
      "adminUserName": "string",
      "authType": "string",
      "backupRetentionPeriod": number,
      "clusterArn": "string",
      "clusterEndpoint": "string",
      "clusterName": "string",
      "createTime": "string",
      "kmsKeyId": "string",
      "preferredBackupWindow": "string",
      "preferredMaintenanceWindow": "string",
      "shardCapacity": number,
      "shardCount": number,
```

```
      "shardInstanceCount": number,
      "shards": [
         {
            "createTime": "string",
            "shardId": "string",
            "status": "string"
         }
      ],
      "status": "string",
      "subnetIds": [ "string" ],
      "vpcSecurityGroupIds": [ "string" ]
   }
}
```

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 반송합니다.

다음 데이터는 서비스에 의해 JSON 형식으로 반환됩니다.

cluster

새로 삭제된 탄력적 클러스터에 대한 정보를 반환합니다.

타입: Cluster 객체

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하십시오.

AccessDeniedException

작업을 수행할 권한이 충분하지 않을 때 발생하는 예외.

HTTP 상태 코드: 403

ConflictException

액세스 충돌이 발생했습니다.

HTTP 상태 코드: 409

InternalServerException

내부 서버 오류가 발생했습니다.

HTTP 상태 코드: 500

ResourceNotFoundException

지정된 리소스를 찾을 수 없습니다.

HTTP 상태 코드: 404

ThrottlingException

요청 제한으로 인해 요청이 거부되면 ThrottlingException이 발생합니다.

HTTP 상태 코드: 429

ValidationException

유효성 검사 예외를 정의하는 구조.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

DeleteClusterSnapshot

서비스: Amazon DocumentDB Elastic Clusters

Elastic 클러스터 스냅샷을 삭제합니다.

Request Syntax

```
DELETE /cluster-snapshot/snapshotArn HTTP/1.1
```

URI 요청 파라미터

요청은 다음 URI 파라미터를 사용합니다.

snapshotArn

삭제할 Elastic 클러스터 스냅샷의 ARN 식별자입니다.

필수 여부: 예

Request Body

해당 요청에는 본문이 없습니다.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "snapshot": {
      "adminUserName": "string",
      "clusterArn": "string",
      "clusterCreationTime": "string",
      "kmsKeyId": "string",
      "snapshotArn": "string",
      "snapshotCreationTime": "string",
      "snapshotName": "string",
      "snapshotType": "string",
      "status": "string",
      "subnetIds": [ "string" ],
      "vpcSecurityGroupIds": [ "string" ]
   }
```

```
}
```

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 반송합니다.

다음 데이터는 서비스에 의해 JSON 형식으로 반환됩니다.

snapshot

새로 삭제된 Elastic 클러스터 스냅샷에 대한 정보를 반환합니다.

타입: ClusterSnapshot 객체

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하십시오.

AccessDeniedException

작업을 수행할 권한이 충분하지 않을 때 발생하는 예외.

HTTP 상태 코드: 403

ConflictException

액세스 충돌이 발생했습니다.

HTTP 상태 코드: 409

InternalServerException

내부 서버 오류가 발생했습니다.

HTTP 상태 코드: 500

ResourceNotFoundException

지정된 리소스를 찾을 수 없습니다.

HTTP 상태 코드: 404

ThrottlingException

요청 제한으로 인해 요청이 거부되면 ThrottlingException이 발생합니다.

HTTP 상태 코드: 429

ValidationException

유효성 검사 예외를 정의하는 구조.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

GetCluster

서비스: Amazon DocumentDB Elastic Clusters

특정 탄력적 클러스터에 대한 정보를 반환합니다.

Request Syntax

```
GET /cluster/clusterArn HTTP/1.1
```

URI 요청 파라미터

요청은 다음 URI 파라미터를 사용합니다.

clusterArn

탄력적 클러스터의 ARN 식별자.

필수 여부: 예

Request Body

해당 요청에는 본문이 없습니다.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "cluster": {
      "adminUserName": "string",
      "authType": "string",
      "backupRetentionPeriod": number,
      "clusterArn": "string",
      "clusterEndpoint": "string",
      "clusterName": "string",
      "createTime": "string",
      "kmsKeyId": "string",
      "preferredBackupWindow": "string",
      "preferredMaintenanceWindow": "string",
      "shardCapacity": number,
      "shardCount": number,
```

```
      "shardInstanceCount": number,
      "shards": [
         {
            "createTime": "string",
            "shardId": "string",
            "status": "string"
         }
      ],
      "status": "string",
      "subnetIds": [ "string" ],
      "vpcSecurityGroupIds": [ "string" ]
   }
}
```

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 반송합니다.

다음 데이터는 서비스에 의해 JSON 형식으로 반환됩니다.

cluster

특정 탄력적 클러스터에 대한 정보를 반환합니다.

타입: Cluster 객체

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하십시오.

AccessDeniedException

작업을 수행할 권한이 충분하지 않을 때 발생하는 예외.

HTTP 상태 코드: 403

InternalServerException

내부 서버 오류가 발생했습니다.

HTTP 상태 코드: 500

ResourceNotFoundException

지정된 리소스를 찾을 수 없습니다.

HTTP 상태 코드: 404

ThrottlingException

요청 제한으로 인해 요청이 거부되면 ThrottlingException이 발생합니다.

HTTP 상태 코드: 429

ValidationException

유효성 검사 예외를 정의하는 구조.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

GetClusterSnapshot

서비스: Amazon DocumentDB Elastic Clusters

특정 탄력적 클러스터 스냅샷에 대한 정보를 반환합니다.

Request Syntax

```
GET /cluster-snapshot/snapshotArn HTTP/1.1
```

URI 요청 파라미터

요청은 다음 URI 파라미터를 사용합니다.

snapshotArn

탄력적 클러스터 스냅샷의 ARN 식별자.

필수 여부: 예

Request Body

해당 요청에는 본문이 없습니다.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "snapshot": {
      "adminUserName": "string",
      "clusterArn": "string",
      "clusterCreationTime": "string",
      "kmsKeyId": "string",
      "snapshotArn": "string",
      "snapshotCreationTime": "string",
      "snapshotName": "string",
      "snapshotType": "string",
      "status": "string",
      "subnetIds": [ "string" ],
      "vpcSecurityGroupIds": [ "string" ]
   }
```

```
}
```

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 반송합니다.

다음 데이터는 서비스에 의해 JSON 형식으로 반환됩니다.

snapshot

특정 탄력적 클러스터 스냅샷에 대한 정보를 반환합니다.

타입: ClusterSnapshot 객체

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하십시오.

AccessDeniedException

작업을 수행할 권한이 충분하지 않을 때 발생하는 예외.

HTTP 상태 코드: 403

InternalServerException

내부 서버 오류가 발생했습니다.

HTTP 상태 코드: 500

ResourceNotFoundException

지정된 리소스를 찾을 수 없습니다.

HTTP 상태 코드: 404

ThrottlingException

요청 제한으로 인해 요청이 거부되면 ThrottlingException이 발생합니다.

HTTP 상태 코드: 429

ValidationException

유효성 검사 예외를 정의하는 구조.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

GetPendingMaintenanceAction

서비스: Amazon DocumentDB Elastic Clusters

보류 중인 모든 유지 관리 작업을 검색합니다.

Request Syntax

```
GET /pending-action/resourceArn HTTP/1.1
```

URI 요청 파라미터

요청은 다음 URI 파라미터를 사용합니다.

resourceArn

특정 Amazon 리소스 이름(ARN)에 대해 보류 중인 유지 관리 작업을 검색합니다.

길이 제약: 최소 길이 1. 최대 길이는 256.

필수 여부: 예

Request Body

해당 요청에는 본문이 없습니다.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "resourcePendingMaintenanceAction": {
      "pendingMaintenanceActionDetails": [
         {
            "action": "string",
            "autoAppliedAfterDate": "string",
            "currentApplyDate": "string",
            "description": "string",
            "forcedApplyDate": "string",
            "optInStatus": "string"
         }
```

```
        ],
        "resourceArn": "string"
    }
}
```

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 반송합니다.

다음 데이터는 서비스에 의해 JSON 형식으로 반환됩니다.

resourcePendingMaintenanceAction

대기 중인 리소스 유지 관리 작업에 대한 정보를 제공합니다.

타입: ResourcePendingMaintenanceAction 객체

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하십시오.

AccessDeniedException

작업을 수행할 권한이 충분하지 않을 때 발생하는 예외.

HTTP 상태 코드: 403

ConflictException

액세스 충돌이 발생했습니다.

HTTP 상태 코드: 409

InternalServerException

내부 서버 오류가 발생했습니다.

HTTP 상태 코드: 500

ResourceNotFoundException

지정된 리소스를 찾을 수 없습니다.

HTTP 상태 코드: 404

ThrottlingException

요청 제한으로 인해 요청이 거부되면 ThrottlingException이 발생합니다.

HTTP 상태 코드: 429

ValidationException

유효성 검사 예외를 정의하는 구조.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

ListClusters

서비스: Amazon DocumentDB Elastic Clusters

프로비저닝된 Amazon DocumentDB 탄력적 클러스터에 대한 정보를 반환합니다.

Request Syntax

```
GET /clusters?maxResults=maxResults&nextToken=nextToken HTTP/1.1
```

URI 요청 파라미터

요청은 다음 URI 파라미터를 사용합니다.

maxResults

응답에 수신될 최대 탄력적 클러스터 스냅샷 결과의 수.

유효 범위: 최소값 1. 최댓값은 100입니다.

nextToken

이전의 요청에서 제공된 페이지 매김 토큰. 이 파라미터가 지정된 경우 응답에는 이 토큰을 넘어 `max-results`에 의해 지정된 값까지의 레코드만 포함됩니다.

응답에 더 이상 데이터가 없는 경우 `nextToken`은 반환되지 않습니다.

Request Body

해당 요청에는 본문이 없습니다.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "clusters": [
      {
         "clusterArn": "string",
         "clusterName": "string",
         "status": "string"
      }
```

```
    ],
    "nextToken": "string"
}
```

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 반송합니다.

다음 데이터는 서비스에 의해 JSON 형식으로 반환됩니다.

clusters

Amazon DocumentDB 탄력적 클러스터의 목록.

타입: ClusterInList객체 배열

nextToken

이전의 요청에서 제공된 페이지 매김 토큰. 이 파라미터가 지정된 경우 응답에는 이 토큰을 넘어 max-results에 의해 지정된 값까지의 레코드만 포함됩니다.

응답에 더 이상 데이터가 없는 경우 nextToken은 반환되지 않습니다.

유형: 문자열

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하십시오.

AccessDeniedException

작업을 수행할 권한이 충분하지 않을 때 발생하는 예외.

HTTP 상태 코드: 403

InternalServerException

내부 서버 오류가 발생했습니다.

HTTP 상태 코드: 500

ThrottlingException

요청 제한으로 인해 요청이 거부되면 ThrottlingException이 발생합니다.

HTTP 상태 코드: 429

ValidationException

유효성 검사 예외를 정의하는 구조.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

ListClusterSnapshots

서비스: Amazon DocumentDB Elastic Clusters

지정된 탄력적 클러스터의 스냅샷에 대한 정보를 반환합니다.

Request Syntax

```
GET /cluster-snapshots?
clusterArn=clusterArn&maxResults=maxResults&nextToken=nextToken&snapshotType=snapshotType
 HTTP/1.1
```

URI 요청 파라미터

요청은 다음 URI 파라미터를 사용합니다.

clusterArn

탄력적 클러스터의 ARN 식별자.

maxResults

응답에 수신될 최대 탄력적 클러스터 스냅샷 결과의 수.

유효한 범위: 최소값은 20입니다. 최댓값은 100입니다.

nextToken

이전의 요청에서 제공된 페이지 매김 토큰. 이 파라미터가 지정된 경우 응답에는 이 토큰을 넘어 `max-results`에 의해 지정된 값까지의 레코드만 포함됩니다.

응답에 더 이상 데이터가 없는 경우 `nextToken`은 반환되지 않습니다.

snapshotType

반환되는 클러스터 스냅샷의 유형. 다음 값 중 하나를 지정할 수 있습니다.

- `automated` - Amazon DocumentDB가 AWS 계정에 대해 자동으로 생성한 모든 클러스터 스냅샷을 반환합니다.
- `manual` - AWS 계정에 대해 수동으로 생성한 모든 클러스터 스냅샷을 반환합니다.

Request Body

해당 요청에는 본문이 없습니다.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "nextToken": "string",
   "snapshots": [
      {
         "clusterArn": "string",
         "snapshotArn": "string",
         "snapshotCreationTime": "string",
         "snapshotName": "string",
         "status": "string"
      }
   ]
}
```

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 반송합니다.

다음 데이터는 서비스에 의해 JSON 형식으로 반환됩니다.

nextToken

이전의 요청에서 제공된 페이지 매김 토큰. 이 파라미터가 지정된 경우 응답에는 이 토큰을 넘어 max-results에 의해 지정된 값까지의 레코드만 포함됩니다.

응답에 더 이상 데이터가 없는 경우 nextToken은 반환되지 않습니다.

유형: 문자열

snapshots

지정된 탄력적 클러스터의 스냅샷 목록.

타입: ClusterSnapshotInList 객체 배열

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하십시오.

AccessDeniedException

작업을 수행할 권한이 충분하지 않을 때 발생하는 예외.

HTTP 상태 코드: 403

InternalServerException

내부 서버 오류가 발생했습니다.

HTTP 상태 코드: 500

ThrottlingException

요청 제한으로 인해 요청이 거부되면 ThrottlingException이 발생합니다.

HTTP 상태 코드: 429

ValidationException

유효성 검사 예외를 정의하는 구조.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

ListPendingMaintenanceActions

서비스: Amazon DocumentDB Elastic Clusters

보류 중인 모든 유지 관리 작업 목록을 검색합니다.

Request Syntax

```
GET /pending-actions?maxResults=maxResults&nextToken=nextToken HTTP/1.1
```

URI 요청 파라미터

요청은 다음 URI 파라미터를 사용합니다.

maxResults

응답에 포함되는 최대 결과 수입니다. 지정된 maxResults 값보다 레코드 수가 많으면 페이지 매김 토큰(마커)을 응답에 포함시켜 나머지 결과를 검색할 수 있도록 합니다.

유효 범위: 최소값 1. 최댓값은 100입니다.

nextToken

이전의 요청에서 제공된 선택적 페이지 매김 토큰입니다. 이 파라미터를 지정한 경우, 마커 이후부터 maxResults에 지정된 값까지의 레코드만 응답에 포함됩니다.

Request Body

해당 요청에는 본문이 없습니다.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "nextToken": "string",
   "resourcePendingMaintenanceActions": [
      {
         "pendingMaintenanceActionDetails": [
            {
               "action": "string",
               "autoAppliedAfterDate": "string",
```

```
                "currentApplyDate": "string",
                "description": "string",
                "forcedApplyDate": "string",
                "optInStatus": "string"
            }
        ],
        "resourceArn": "string"
    }
  ]
}
```

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 반송합니다.

다음 데이터는 서비스에 의해 JSON 형식으로 반환됩니다.

resourcePendingMaintenanceActions

대기 중인 리소스 유지 관리 작업에 대한 정보를 제공합니다.

타입: ResourcePendingMaintenanceAction객체 배열

nextToken

이전의 요청에서 제공된 선택적 페이지 매김 토큰입니다. 이 파라미터가 표시되면 응답에는에서 지정한 값까지 마커 이외의 레코드만 포함됩니다maxResults.

유형: 문자열

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하십시오.

AccessDeniedException

작업을 수행할 권한이 충분하지 않을 때 발생하는 예외.

HTTP 상태 코드: 403

InternalServerException

내부 서버 오류가 발생했습니다.

HTTP 상태 코드: 500

ThrottlingException

요청 제한으로 인해 요청이 거부되면 ThrottlingException이 발생합니다.

HTTP 상태 코드: 429

ValidationException

유효성 검사 예외를 정의하는 구조.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

ListTagsForResource

서비스: Amazon DocumentDB Elastic Clusters

탄력적 클러스터 리소스의 모든 태그 나열하기

Request Syntax

```
GET /tags/resourceArn HTTP/1.1
```

URI 요청 파라미터

요청은 다음 URI 파라미터를 사용합니다.

resourceArn

탄력적 클러스터 리소스의 ARN 식별자.

길이 제약 조건: 최소 길이는 1입니다. 최대 길이는 1011입니다.

필수 여부: 예

Request Body

해당 요청에는 본문이 없습니다.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "tags": {
      "string" : "string"
   }
}
```

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 반송합니다.

다음 데이터는 서비스에 의해 JSON 형식으로 반환됩니다.

tags

지정된 탄력적 클러스터 리소스의 태그 목록.

유형: 문자열 간 맵

키 길이 제약 조건: 최소 길이는 1. 최대 길이 128.

키 패턴: `(?!aws:)[a-zA-Z+-=._:/]+`

값 길이 제약 조건: 최소 길이는 0입니다. 최대 길이는 256입니다.

오류

모든 작업에서 발생하는 일반적인 오류에 대한 자세한 내용은 일반적인 오류 섹션을 참조하세요.

InternalServerException

내부 서버 오류가 발생했습니다.

HTTP 상태 코드: 500

ResourceNotFoundException

지정된 리소스를 찾을 수 없습니다.

HTTP 상태 코드: 404

ThrottlingException

요청 제한으로 인해 요청이 거부되면 ThrottlingException이 발생합니다.

HTTP 상태 코드: 429

ValidationException

유효성 검사 예외를 정의하는 구조.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface

- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

RestoreClusterFromSnapshot

서비스: Amazon DocumentDB Elastic Clusters

스냅샷에서 탄력적 클러스터를 복원합니다.

Request Syntax

```
POST /cluster-snapshot/snapshotArn/restore HTTP/1.1
Content-type: application/json

{
   "clusterName": "string",
   "kmsKeyId": "string",
   "shardCapacity": number,
   "shardInstanceCount": number,
   "subnetIds": [ "string" ],
   "tags": {
      "string" : "string"
   },
   "vpcSecurityGroupIds": [ "string" ]
}
```

URI 요청 파라미터

요청은 다음 URI 파라미터를 사용합니다.

snapshotArn

탄력적 클러스터 스냅샷의 ARN 식별자.

필수 여부: 예

요청 본문

요청은 JSON 형식으로 다음 데이터를 받습니다.

clusterName

Elastic 클러스터의 이름입니다.

유형: 문자열

필수 항목 여부: 예

kmsKeyId

새 Amazon DocumentDB Elastic 클러스터 클러스터를 암호화하는 데 사용할 KMS 키 식별자입니다.

KMS 키 식별자는 KMS 암호화 키의 Amazon 리소스 이름(ARN)입니다. 이 KMS 암호화 키를 소유하고 있는 동일한 Amazon 계정을 사용하여 클러스터를 생성하는 경우, ARN 대신 KMS 키 별칭을 KMS 암호화 키로 사용할 수 있습니다.

여기에 암호화 키가 지정되지 않은 경우 Amazon DocumentDB는 KMS가 사용자 계정에 대해 생성하는 기본 암호화 키를 사용합니다. 계정에는 Amazon 리전마다 다른 기본 암호화 키가 있습니다.

유형: 문자열

필수 항목 여부: 아니요

shardCapacity

새로 복원된 탄력적 클러스터의 각 샤드 용량입니다.

유형: 정수

필수 항목 여부: 아니요

shardInstanceCount

탄력적 클러스터의 모든 샤드에 적용되는 복제본 인스턴스 수입니다. `shardInstanceCount` 값이 1이면 라이터 인스턴스가 하나라는 의미이며, 추가 인스턴스는 읽기에 사용할 수 있고 가용성을 개선하는 복제본입니다.

유형: 정수

필수 항목 여부: 아니요

subnetIds

탄력적 클러스터의 Amazon EC2 서브넷 ID입니다.

유형: 문자열 배열

필수 여부: 아니요

tags

키가 태그 이름이고 값이 키 값인 키-값 쌍의 배열 형태로 복원된 Elastic 클러스터에 할당될 태그 이름 목록입니다.

유형: 문자열 간 맵

키 길이 제약 조건: 최소 길이는 1. 최대 길이 128.

키 패턴: `(?!aws:)[a-zA-Z+-=._:/]+`

값 길이 제약 조건: 최소 길이는 0입니다. 최대 길이는 256입니다.

필수 여부: 아니요

vpcSecurityGroupIds

이 엘라스틱 클러스터와 연결할 EC2 VPC 보안 그룹 목록입니다.

유형: 문자열 배열

필수 항목 여부: 아니요

응답 구문

```
HTTP/1.1 200
Content-type: application/json

{
   "cluster": {
      "adminUserName": "string",
      "authType": "string",
      "backupRetentionPeriod": number,
      "clusterArn": "string",
      "clusterEndpoint": "string",
      "clusterName": "string",
      "createTime": "string",
      "kmsKeyId": "string",
      "preferredBackupWindow": "string",
      "preferredMaintenanceWindow": "string",
      "shardCapacity": number,
      "shardCount": number,
      "shardInstanceCount": number,
      "shards": [
         {
            "createTime": "string",
            "shardId": "string",
            "status": "string"
         }
```

```
        ],
        "status": "string",
        "subnetIds": [ "string" ],
        "vpcSecurityGroupIds": [ "string" ]
    }
}
```

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 반송합니다.

다음 데이터는 서비스에 의해 JSON 형식으로 반환됩니다.

cluster

복원된 Elastic 클러스터에 대한 정보를 반환합니다.

타입: Cluster 객체

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하십시오.

AccessDeniedException

작업을 수행할 권한이 충분하지 않을 때 발생하는 예외.

HTTP 상태 코드: 403

ConflictException

액세스 충돌이 발생했습니다.

HTTP 상태 코드: 409

InternalServerException

내부 서버 오류가 발생했습니다.

HTTP 상태 코드: 500

ResourceNotFoundException

지정된 리소스를 찾을 수 없습니다.

HTTP 상태 코드: 404

ServiceQuotaExceededException

작업에 대한 서비스 할당량을 초과했습니다.

HTTP 상태 코드: 402

ThrottlingException

요청 조절로 인해 요청이 거부되면 ThrottlingException이 발생합니다.

HTTP 상태 코드: 429

ValidationException

유효성 검사 예외를 정의하는 구조.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

StartCluster

서비스: Amazon DocumentDB Elastic Clusters

`clusterARN`에서 지정한 중지된 탄력적 클러스터를 다시 시작합니다.

Request Syntax

```
POST /cluster/clusterArn/start HTTP/1.1
```

URI 요청 파라미터

요청은 다음 URI 파라미터를 사용합니다.

clusterArn

탄력적 클러스터의 ARN 식별자.

필수 여부: 예

Request Body

해당 요청에는 본문이 없습니다.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "cluster": {
      "adminUserName": "string",
      "authType": "string",
      "backupRetentionPeriod": number,
      "clusterArn": "string",
      "clusterEndpoint": "string",
      "clusterName": "string",
      "createTime": "string",
      "kmsKeyId": "string",
      "preferredBackupWindow": "string",
      "preferredMaintenanceWindow": "string",
      "shardCapacity": number,
      "shardCount": number,
```

```
      "shardInstanceCount": number,
      "shards": [
         {
            "createTime": "string",
            "shardId": "string",
            "status": "string"
         }
      ],
      "status": "string",
      "subnetIds": [ "string" ],
      "vpcSecurityGroupIds": [ "string" ]
   }
}
```

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 반송합니다.

다음 데이터는 서비스에 의해 JSON 형식으로 반환됩니다.

cluster

특정 탄력적 클러스터에 대한 정보를 반환합니다.

타입: Cluster 객체

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하십시오.

AccessDeniedException

작업을 수행할 권한이 충분하지 않을 때 발생하는 예외.

HTTP 상태 코드: 403

InternalServerException

내부 서버 오류가 발생했습니다.

HTTP 상태 코드: 500

ResourceNotFoundException

지정된 리소스를 찾을 수 없습니다.

HTTP 상태 코드: 404

ThrottlingException

요청 제한으로 인해 요청이 거부되면 ThrottlingException이 발생합니다.

HTTP 상태 코드: 429

ValidationException

유효성 검사 예외를 정의하는 구조.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

StopCluster

서비스: Amazon DocumentDB Elastic Clusters

`clusterArn`에서 지정한 실행 중인 탄력적 클러스터를 중지합니다. 탄력적 클러스터가 사용 가능 상태여야 합니다.

Request Syntax

```
POST /cluster/clusterArn/stop HTTP/1.1
```

URI 요청 파라미터

요청은 다음 URI 파라미터를 사용합니다.

clusterArn

탄력적 클러스터의 ARN 식별자.

필수 여부: 예

Request Body

해당 요청에는 본문이 없습니다.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "cluster": {
      "adminUserName": "string",
      "authType": "string",
      "backupRetentionPeriod": number,
      "clusterArn": "string",
      "clusterEndpoint": "string",
      "clusterName": "string",
      "createTime": "string",
      "kmsKeyId": "string",
      "preferredBackupWindow": "string",
      "preferredMaintenanceWindow": "string",
```

```
      "shardCapacity": number,
      "shardCount": number,
      "shardInstanceCount": number,
      "shards": [
         {
            "createTime": "string",
            "shardId": "string",
            "status": "string"
         }
      ],
      "status": "string",
      "subnetIds": [ "string" ],
      "vpcSecurityGroupIds": [ "string" ]
   }
}
```

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 반송합니다.

다음 데이터는 서비스에 의해 JSON 형식으로 반환됩니다.

cluster

특정 탄력적 클러스터에 대한 정보를 반환합니다.

타입: Cluster 객체

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하십시오.

AccessDeniedException

작업을 수행할 권한이 충분하지 않을 때 발생하는 예외.

HTTP 상태 코드: 403

InternalServerException

내부 서버 오류가 발생했습니다.

HTTP 상태 코드: 500

ResourceNotFoundException

지정된 리소스를 찾을 수 없습니다.

HTTP 상태 코드: 404

ThrottlingException

요청 제한으로 인해 요청이 거부되면 ThrottlingException이 발생합니다.

HTTP 상태 코드: 429

ValidationException

유효성 검사 예외를 정의하는 구조.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go v2용 SDK
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

TagResource

서비스: Amazon DocumentDB Elastic Clusters

메타데이터 태그를 엘라스틱 클러스터 리소스에 추가합니다.

Request Syntax

```
POST /tags/resourceArn HTTP/1.1
Content-type: application/json

{
   "tags": {
      "string" : "string"
   }
}
```

URI 요청 파라미터

요청은 다음 URI 파라미터를 사용합니다.

resourceArn

탄력적 클러스터 리소스의 ARN 식별자.

길이 제약 조건: 최소 길이는 1입니다. 최대 길이는 1011입니다.

필수 여부: 예

요청 본문

요청은 JSON 형식으로 다음 데이터를 받습니다.

tags

엘라스틱 클러스터 리소스에 배정된 태그

유형: 문자열 간 맵

키 길이 제약 조건: 최소 길이는 1. 최대 길이 128.

키 패턴: `(?!aws:)[a-zA-Z+-=._:/]+`

값 길이 제약 조건: 최소 길이는 0입니다. 최대 길이는 256.

필수 여부: 예

응답 구문

```
HTTP/1.1 200
```

Response Elements

작업이 성공하면 서비스가 비어 있는 HTTP 본문과 함께 HTTP 200 응답을 다시 전송합니다.

오류

모든 작업에서 발생하는 일반적인 오류에 대한 자세한 내용은 일반적인 오류 섹션을 참조하세요.

InternalServerException

내부 서버 오류가 발생했습니다.

HTTP 상태 코드: 500

ResourceNotFoundException

지정된 리소스를 찾을 수 없습니다.

HTTP 상태 코드: 404

ThrottlingException

요청 제한으로 인해 요청이 거부되면 ThrottlingException이 발생합니다.

HTTP 상태 코드: 429

ValidationException

유효성 검사 예외를 정의하는 구조.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface

- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

UntagResource

서비스: Amazon DocumentDB Elastic Clusters

탄력적 클러스터 리소스에서 메타데이터 태그를 제거합니다.

Request Syntax

```
DELETE /tags/resourceArn?tagKeys=tagKeys HTTP/1.1
```

URI 요청 파라미터

요청은 다음 URI 파라미터를 사용합니다.

resourceArn

탄력적 클러스터 리소스의 ARN 식별자.

길이 제약 조건: 최소 길이는 1입니다. 최대 길이는 1011입니다.

필수 여부: 예

tagKeys

탄력적 클러스터 리소스에서 제거해야 할 태그 키.

배열 구성원: 최소수는 0개입니다. 최대수 50개.

길이 제약: 최소 길이 1. 최대 길이 128.

패턴: `(?!aws:)[a-zA-Z+-=._:/]+`

필수 여부: 예

Request Body

해당 요청에는 본문이 없습니다.

Response Syntax

```
HTTP/1.1 200
```

Response Elements

작업이 성공하면 서비스가 비어 있는 HTTP 본문과 함께 HTTP 200 응답을 다시 전송합니다.

오류

모든 작업에서 발생하는 일반적인 오류에 대한 자세한 내용은 일반적인 오류 섹션을 참조하세요.

InternalServerException

내부 서버 오류가 발생했습니다.

HTTP 상태 코드: 500

ResourceNotFoundException

지정된 리소스를 찾을 수 없습니다.

HTTP 상태 코드: 404

ThrottlingException

요청 제한으로 인해 요청이 거부되면 ThrottlingException이 발생합니다.

HTTP 상태 코드: 429

ValidationException

유효성 검사 예외를 정의하는 구조.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK

- AWS SDK for Ruby V3

UpdateCluster

서비스: Amazon DocumentDB Elastic Clusters

엘라스틱 클러스터를 수정합니다. 여기에는 관리자-사용자 이름/암호 업데이트, API 버전 업그레이드, 백업 기간 및 유지 관리 기간 설정이 포함됩니다.

Request Syntax

```
PUT /cluster/clusterArn HTTP/1.1
Content-type: application/json

{
   "adminUserPassword": "string",
   "authType": "string",
   "backupRetentionPeriod": number,
   "clientToken": "string",
   "preferredBackupWindow": "string",
   "preferredMaintenanceWindow": "string",
   "shardCapacity": number,
   "shardCount": number,
   "shardInstanceCount": number,
   "subnetIds": [ "string" ],
   "vpcSecurityGroupIds": [ "string" ]
}
```

URI 요청 파라미터

요청은 다음 URI 파라미터를 사용합니다.

clusterArn

탄력적 클러스터의 ARN 식별자.

필수 여부: 예

요청 본문

요청은 JSON 형식으로 다음 데이터를 받습니다.

adminUserPassword

엘라스틱 클러스터 관리자와 관련된 비밀번호. 이 암호에는 슬래시(/), 큰따옴표(") 또는 '앳' 기호(@)를 제외한 인쇄 가능 ASCII 문자가 포함될 수 있습니다.

제약: 8~100자여야 합니다.

유형: 문자열

필수 항목 여부: 아니요

authType

엘라스틱 클러스터에 액세스하는 데 사용되는 암호를 가져올 위치를 결정하는 데 사용되는 인증 유형입니다. 유효한 형식은 PLAIN_TEXT 및 SECRET_ARN입니다.

타입: 문자열

유효 값: PLAIN_TEXT | SECRET_ARN

필수 여부: 아니요

backupRetentionPeriod

자동 스냅샷이 보관되는 일수입니다.

유형: 정수

필수 항목 여부: 아니요

clientToken

탄력적 클러스터의 클라이언트 토큰.

유형: 문자열

필수 항목 여부: 아니요

preferredBackupWindow

자동 백업이 활성화된 경우 자동 백업이 생성되는 일일 시간 범위이며, backupRetentionPeriod 속성에 의해 결정됩니다.

유형: 문자열

필수 항목 여부: 아니요

preferredMaintenanceWindow

시스템 유지 관리를 실행할 수 있는 주 단위 기간(UTC, 협정 세계시)입니다.

형식: `ddd:hh24:mi-ddd:hh24:mi`

기본값: 각에 대해 8시간 블록 시간 중에서 임의로 선택되는 30분 기간으로 AWS 리전, 요일에 무작위로 발생합니다.

유효한 요일: 월, 화, 수, 목, 금, 토, 일

제약 조건: 최소 30분의 기간.

유형: 문자열

필수 항목 여부: 아니요

shardCapacity

각 탄력적 클러스터 샤드에 할당된 vCPU 수 최대 값은 64입니다. 허용되는 값은 2, 4, 8, 16, 32, 64입니다.

유형: 정수

필수 항목 여부: 아니요

shardCount

탄력적 클러스터에 할당된 샤드 수. 최대 값은 32입니다.

유형: 정수

필수 항목 여부: 아니요

shardInstanceCount

탄력적 클러스터의 모든 샤드에 적용되는 복제본 인스턴스 수입니다. `shardInstanceCount` 값이 1이면 라이터 인스턴스가 하나라는 의미이며, 추가 인스턴스는 읽기에 사용할 수 있고 가용성을 개선하는 복제본입니다.

유형: 정수

필수 항목 여부: 아니요

subnetIds

탄력적 클러스터의 Amazon EC2 서브넷 ID입니다.

유형: 문자열 배열

필수 여부: 아니요

vpcSecurityGroupIds

이 엘라스틱 클러스터와 연결할 EC2 VPC 보안 그룹 목록입니다.

유형: 문자열 배열

필수 항목 여부: 아니요

응답 구문

```
HTTP/1.1 200
Content-type: application/json

{
   "cluster": {
      "adminUserName": "string",
      "authType": "string",
      "backupRetentionPeriod": number,
      "clusterArn": "string",
      "clusterEndpoint": "string",
      "clusterName": "string",
      "createTime": "string",
      "kmsKeyId": "string",
      "preferredBackupWindow": "string",
      "preferredMaintenanceWindow": "string",
      "shardCapacity": number,
      "shardCount": number,
      "shardInstanceCount": number,
      "shards": [
         {
            "createTime": "string",
            "shardId": "string",
            "status": "string"
         }
      ],
      "status": "string",
      "subnetIds": [ "string" ],
      "vpcSecurityGroupIds": [ "string" ]
   }
```

```
}
```

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 반송합니다.

다음 데이터는 서비스에 의해 JSON 형식으로 반환됩니다.

cluster

업데이트된 Elastic 클러스터에 대한 정보를 반환합니다.

타입: Cluster 객체

오류

모든 작업에 공통되는 오류에 대한 내용은 일반적인 오류 섹션을 참조하십시오.

AccessDeniedException

작업을 수행할 권한이 충분하지 않을 때 발생하는 예외.

HTTP 상태 코드: 403

ConflictException

액세스 충돌이 발생했습니다.

HTTP 상태 코드: 409

InternalServerException

내부 서버 오류가 발생했습니다.

HTTP 상태 코드: 500

ResourceNotFoundException

지정된 리소스를 찾을 수 없습니다.

HTTP 상태 코드: 404

ThrottlingException

요청 제한으로 인해 요청이 거부되면 ThrottlingException이 발생합니다.

HTTP 상태 코드: 429

ValidationException

유효성 검사 예외를 정의하는 구조.

HTTP 상태 코드: 400

참고

언어별 AWS SDKs

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go용 SDK v2
- AWS Java V2용 SDK
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS PHP V3용 SDK
- AWS Python용 SDK
- AWS SDK for Ruby V3

데이터 타입

다음 데이터 형식은 Amazon DocumentDB (with MongoDB compatibility)에서 지원됩니다.

- AvailabilityZone
- Certificate
- CertificateDetails
- CloudwatchLogsExportConfiguration
- ClusterMasterUserSecret
- DBCluster
- DBClusterMember
- DBClusterParameterGroup

- DBClusterRole
- DBClusterSnapshot
- DBClusterSnapshotAttribute
- DBClusterSnapshotAttributesResult
- DBEngineVersion
- DBInstance
- DBInstanceStatusInfo
- DBSubnetGroup
- Endpoint
- EngineDefaults
- Event
- EventCategoriesMap
- EventSubscription
- Filter
- GlobalCluster
- GlobalClusterMember
- OrderableDBInstanceOption
- Parameter
- PendingCloudwatchLogsExports
- PendingMaintenanceAction
- PendingModifiedValues
- ResourcePendingMaintenanceActions
- Subnet
- Tag
- UpgradeTarget
- VpcSecurityGroupMembership

Amazon DocumentDB Elastic Clusters에서 지원되는 데이터 유형은 다음과 같습니다.

- Cluster
- ClusterInList

- ClusterSnapshot
- ClusterSnapshotInList
- PendingMaintenanceActionDetails
- ResourcePendingMaintenanceAction
- Shard
- ValidationExceptionField

Amazon DocumentDB (with MongoDB compatibility)

다음 데이터 형식은 Amazon DocumentDB (with MongoDB compatibility)에서 지원됩니다.

- AvailabilityZone
- Certificate
- CertificateDetails
- CloudwatchLogsExportConfiguration
- ClusterMasterUserSecret
- DBCluster
- DBClusterMember
- DBClusterParameterGroup
- DBClusterRole
- DBClusterSnapshot
- DBClusterSnapshotAttribute
- DBClusterSnapshotAttributesResult
- DBEngineVersion
- DBInstance
- DBInstanceStatusInfo
- DBSubnetGroup
- Endpoint
- EngineDefaults
- Event
- EventCategoriesMap
- EventSubscription

- Filter
- GlobalCluster
- GlobalClusterMember
- OrderableDBInstanceOption
- Parameter
- PendingCloudwatchLogsExports
- PendingMaintenanceAction
- PendingModifiedValues
- ResourcePendingMaintenanceActions
- Subnet
- Tag
- UpgradeTarget
- VpcSecurityGroupMembership

AvailabilityZone

서비스: Amazon DocumentDB (with MongoDB compatibility)

가용 영역과 관련된 정보입니다.

내용

> **Note**
>
> 필수 파라미터는 다음 목록에 설명되어 있습니다.

Name

가용 영역의 이름입니다.

유형: 문자열

필수 항목 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Certificate

서비스: Amazon DocumentDB (with MongoDB compatibility)

에 대한 인증 기관(CA) 인증서입니다 AWS 계정.

내용

> **Note**
>
> 필수 파라미터는 다음 목록에 설명되어 있습니다.

CertificateArn

인증서에 대한 Amazon 리소스 이름(ARN)입니다.

예제: `arn:aws:rds:us-east-1::cert:rds-ca-2019`

유형: 문자열

필수 항목 여부: 아니요

CertificateIdentifier

인증서를 식별하는 고유한 키입니다.

예제: `rds-ca-2019`

유형: 문자열

필수 항목 여부: 아니요

CertificateType

인증서의 유형입니다.

예제: `CA`

유형: 문자열

필수 항목 여부: 아니요

Thumbprint

인증서의 지문.

유형: 문자열

필수 항목 여부: 아니요

ValidFrom

인증서가 유효한 시작 날짜/시간입니다.

예제: `2019-07-31T17:57:09Z`

유형: 타임스탬프

필수 여부: 아니요

ValidTill

인증서가 더 이상 유효하지 않은 날짜-시간입니다.

예제: `2024-07-31T17:57:09Z`

유형: 타임스탬프

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS Java V2용 SDK
- AWS SDK for Ruby V3

CertificateDetails

서비스: Amazon DocumentDB (with MongoDB compatibility)

DB 인스턴스의 서버 인증서 세부 정보를 반환합니다.

자세한 내용은 Amazon DocumentDB 개발자 안내서의 Amazon DocumentDB TLS 인증서 업데이트 및 전송 중 데이터 암호화를 참조하십시오.

내용

> **Note**
>
> 필수 파라미터는 다음 목록에 설명되어 있습니다.

CAIdentifier

DB 인스턴스의 서버 인증서에 사용되는 CA 인증서의 CA 식별자입니다.

유형: 문자열

필수 항목 여부: 아니요

ValidTill

DB 인스턴스 서버 인증서의 만료 날짜.

유형: 타임스탬프

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

CloudwatchLogsExportConfiguration

서비스: Amazon DocumentDB (with MongoDB compatibility)

특정 인스턴스 또는 클러스터에 대하여 Amazon CloudWatch Logs로 내보내기를 활성화할 로그 유형의 구성 설정입니다.

`EnableLogTypes` 및 `DisableLogTypes` 배열이 CloudWatch Logs로 내보내는(또는 내보내지 않을) 로그를 결정합니다. 이러한 배열 내의 값은 사용 중인 엔진에 따라 달라집니다.

내용

> **Note**
>
> 필수 파라미터는 다음 목록에 설명되어 있습니다.

DisableLogTypes.member.N

비활성화할 로그 유형의 목록입니다.

유형: 문자열 배열

필수 여부: 아니요

EnableLogTypes.member.N

활성화할 로그 유형의 목록입니다.

유형: 문자열 배열

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ClusterMasterUserSecret

서비스: Amazon DocumentDB (with MongoDB compatibility)

마스터 사용자 암호에 대한 AWS Secrets Manager의 Amazon DocumentDB에서 관리하는 보안 암호를 포함합니다.

내용

> **Note**
> 필수 파라미터는 다음 표에 처음으로 설명되어 있습니다.

KmsKeyId

보안 암호를 암호화하는 데 사용되는 AWS KMS 키 식별자입니다.

유형: 문자열

필수 항목 여부: 아니요

SecretArn

보안 암호의 Amazon 리소스 이름(ARN)입니다.

유형: 문자열

필수 항목 여부: 아니요

SecretStatus

보안 암호의 상태입니다.

가능한 상태 값에는 다음이 포함됩니다.

- 생성 중 - 보안 암호가 생성 중입니다.
- 활성 - 보안 암호는 일반적인 사용 및 교체에 사용할 수 있습니다.
- 교체 중 - 보안 암호가 교체 중입니다.
- 손상됨 - 보안 암호를 사용하여 데이터베이스 자격 증명에 액세스할 수 있지만 교체할 수는 없습니다. 예를 들어 Amazon DocumentDB가 더 이상 보안 암호 또는 보안 암호에 대한 KMS 키에 액세스할 수 없도록 권한이 변경되는 경우 보안 암호는이 상태를 가질 수 있습니다.

암호가 이 상태인 경우 상태를 초래한 조건을 수정할 수 있습니다. 또는 인스턴스를 수정하여 데이터베이스 자격 증명의 자동 관리를 끈 다음 인스턴스를 다시 수정하여 데이터베이스 자격 증명의 자동 관리를 켭니다.

유형: 문자열

필수 항목 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS Java V2용 SDK
- AWS SDK for Ruby V3

DBCluster

서비스: Amazon DocumentDB (with MongoDB compatibility)

클러스터에 관한 자세한 정보.

내용

> **Note**
>
> 필수 파라미터는 다음 표에 처음으로 설명되어 있습니다.

AssociatedRoles.DBClusterRole.N

클러스터와 연결된 AWS Identity and Access Management (IAM) 역할 목록을 제공합니다. 클러스터와 연결된 (IAM) 역할은 클러스터가 사용자를 대신하여 다른 AWS 서비스에 액세스할 수 있는 권한을 부여합니다.

유형: DBClusterRole 객체 배열

필수 여부: 아니요

AvailabilityZones.AvailabilityZone.N

클러스터의 인스턴스를 생성할 수 있는 Amazon EC2 가용 영역의 목록을 제공합니다.

유형: 문자열 배열

필수 여부: 아니요

BackupRetentionPeriod

자동 스냅샷이 보관되는 일수를 지정합니다.

유형: 정수

필수 항목 여부: 아니요

CloneGroupId

DB 클러스터와 연결되어 있는 복제 그룹을 나타냅니다.

유형: 문자열

필수 항목 여부: 아니요

ClusterCreateTime

클러스터가 생성된 시간(협정 세계시(UTC))을 나타냅니다.

유형: 타임스탬프

필수 여부: 아니요

DBClusterArn

클러스터의 Amazon 리소스 이름(ARN).

유형: 문자열

필수 항목 여부: 아니요

DBClusterIdentifier

사용자가 제공한 클러스터 식별자가 포함되어 있습니다. 이 식별자는 클러스터를 식별하는 고유한 키입니다.

유형: 문자열

필수 항목 여부: 아니요

DBClusterMembers.DBClusterMember.N

클러스터를 구성하는 인스턴스의 목록을 제공합니다.

유형: DBClusterMember 객체 배열

필수 여부: 아니요

DBClusterParameterGroup

클러스터에 사용할 클러스터 파라미터 그룹의 이름을 지정합니다.

유형: 문자열

필수 항목 여부: 아니요

DbClusterResourceId

클러스터의 AWS 리전고유하고 변경 불가능한 식별자입니다. 이 식별자는 클러스터의 AWS KMS 키에 액세스할 때마다 AWS CloudTrail 로그 항목에서 찾을 수 있습니다.

유형: 문자열

필수 항목 여부: 아니요

DBSubnetGroup

이름, 설명 및 서브넷 그룹 내의 서브넷 등 클러스터와 연결된 서브넷 그룹에 대한 정보를 지정합니다.

유형: 문자열

필수 항목 여부: 아니요

DeletionProtection

이 클러스터를 삭제할 수 있는지 없는지를 지정합니다. `DeletionProtection`이 활성화되면 클러스터를 수정하고 `DeletionProtection` 를 비활성화하지 않는 한 클러스터를 삭제할 수 없습니다. `DeletionProtection`은 클러스터가 실수로 삭제되지 않도록 보호합니다.

유형: 부울

필수 항목 여부: 아니요

EarliestRestorableTime

특정 시점으로 복원을 사용하여 데이터베이스를 복원할 수 있는 가장 이른 시간.

유형: 타임스탬프

필수 여부: 아니요

EnabledCloudwatchLogsExports.member.N

이 클러스터에서 Amazon CloudWatch Logs로 내보내도록 구성된 로그 유형의 목록.

유형: 문자열 배열

필수 여부: 아니요

Endpoint

클러스터의 기본 인스턴스에 대한 연결 엔드포인트를 지정합니다.

유형: 문자열

필수 항목 여부: 아니요

Engine

이 클러스터에 사용할 데이터베이스 엔진의 이름을 제공합니다.

유형: 문자열

필수 항목 여부: 아니요

EngineVersion

데이터베이스 엔진의 버전을 나타냅니다.

유형: 문자열

필수 항목 여부: 아니요

HostedZoneId

호스팅 영역을 생성할 때 Amazon Route 53에서 할당하는 ID를 나타냅니다.

유형: 문자열

필수 항목 여부: 아니요

KmsKeyId

이 `StorageEncrypted`인 경우 암호화된 클러스터의 `true` AWS KMS 키 식별자입니다.

유형: 문자열

필수 항목 여부: 아니요

LatestRestorableTime

특정 시점으로 복원을 사용하여 데이터베이스를 복원할 수 있는 가장 늦은 시간을 지정합니다.

유형: 타임스탬프

필수 여부: 아니요

MasterUsername

클러스터의 마스터 사용자 이름이 포함되어 있습니다.

유형: 문자열

필수 항목 여부: 아니요

MasterUserSecret

마스터 사용자 암호에 대해 AWS Secrets Manager의 Amazon DocumentDB에서 관리하는 보안 암호입니다.

유형: ClusterMasterUserSecret 객체

필수 여부: 아니요

MultiAZ

클러스터에 여러 가용 영역의 인스턴스가 있는지 여부를 나타냅니다.

유형: 부울

필수 항목 여부: 아니요

PercentProgress

작업의 진행 상황을 백분율로 나타냅니다.

유형: 문자열

필수 항목 여부: 아니요

Port

데이터베이스 엔진이 수신 대기하는 포트를 지정합니다.

유형: 정수

필수 항목 여부: 아니요

PreferredBackupWindow

자동 백업이 활성화된 경우 자동 백업이 생성되는 일일 시간 범위를 나타내며, `BackupRetentionPeriod` 속성에 의해 결정됩니다.

유형: 문자열

필수 항목 여부: 아니요

PreferredMaintenanceWindow

시스템 유지 관리를 실행할 수 있는 주 단위 기간(UTC, 협정 세계시)을 지정합니다.

유형: 문자열

필수 항목 여부: 아니요

ReaderEndpoint

클러스터에 대한 리더 엔드포인트입니다. 클러스터의 리더 엔드포인트는 클러스터에서 사용 가능한 Amazon DocumentDB 복제본 간의 연결을 로드 밸런싱합니다. 클라이언트가 리더 엔드포인트에 대한 새로운 연결을 요청하면 Amazon DocumentDB는 클러스터 내의 Amazon DocumentDB 복제본 간에 연결 요청을 분배합니다. 이 기능은 클러스터 내의 여러 Amazon DocumentDB 복제본 간에 읽기 워크로드의 균형을 유지하는 데 도움이 됩니다.

장애 조치가 발생하고 사용자와 연결된 Amazon DocumentDB 복제본이 기본 인스턴스로 승격되면 연결이 끊어집니다. 읽기 워크로드를 클러스터 내의 다른 Amazon DocumentDB 복제본으로 계속 전송하려면 리더 엔드포인트에 다시 연결하면 됩니다.

유형: 문자열

필수 항목 여부: 아니요

ReadReplicaIdentifiers.ReadReplicaIdentifier.N

이 클러스터와 연결된 보조 클러스터의 식별자를 하나 이상 포함합니다.

유형: 문자열 배열

필수 여부: 아니요

ReplicationSourceIdentifier

이 클러스터가 보조 클러스터인 경우 소스 클러스터의 식별자를 포함합니다.

유형: 문자열

필수 항목 여부: 아니요

Status

이 클러스터의 현재 상태를 지정합니다.

유형: 문자열

필수 항목 여부: 아니요

StorageEncrypted

클러스터의 암호화 여부를 지정합니다.

유형: 부울

필수 항목 여부: 아니요

StorageType

클러스터와 연결된 스토리지 유형

Amazon DocumentDB 클러스터의 스토리지 유형에 관한 자세한 내용은 Amazon DocumentDB 개발자 안내서의 클러스터 스토리지 구성을 참조하세요.

스토리지 유형에 유효한 값 - `standard | iopt1`

기본값은 `standard` 입니다.

유형: 문자열

필수 항목 여부: 아니요

VpcSecurityGroups.VpcSecurityGroupMembership.N

클러스터가 속한 Virtual Private Cloud(VPC) 보안 그룹의 목록을 제공합니다.

유형: VpcSecurityGroupMembership 객체 배열

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS Java V2용 SDK
- AWS SDK for Ruby V3

DBClusterMember

서비스: Amazon DocumentDB (with MongoDB compatibility)

클러스터에 포함된 인스턴스에 관한 정보가 나와 있습니다.

내용

> **Note**
> 필수 파라미터는 다음 목록에 먼저 설명되어 있습니다.

DBClusterParameterGroupStatus

이 DB 클러스터 멤버에 사용할 클러스터 파라미터 그룹의 상태를 지정합니다.

유형: 문자열

필수 항목 여부: 아니요

DBInstanceIdentifier

이 클러스터 멤버의 인스턴스 식별자를 지정합니다.

유형: 문자열

필수 항목 여부: 아니요

IsClusterWriter

클러스터 멤버가 클러스터의 기본 인스턴스이면 `true`이고, 그렇지 않으면 `false`인 값입니다.

유형: 부울

필수 항목 여부: 아니요

PromotionTier

기존 기본 인스턴스에 결함이 발생한 후 Amazon DocumentDB 복제본을 기본 인스턴스로 승격할 순서를 지정하는 값

유형: 정수

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS Java V2용 SDK
- AWS SDK for Ruby V3

DBClusterParameterGroup

서비스: Amazon DocumentDB (with MongoDB compatibility)

클러스터 파라미터 그룹의 상세 정보입니다.

내용

> **Note**
> 다음 목록에는 필수 파라미터가 먼저 설명되어 있습니다.

DBClusterParameterGroupArn

클러스터 파라미터 그룹의 Amazon 리소스 이름(ARN)입니다.

유형: 문자열

필수 항목 여부: 아니요

DBClusterParameterGroupName

클러스터 파라미터 그룹의 이름을 제공합니다.

유형: 문자열

필수 항목 여부: 아니요

DBParameterGroupFamily

이 클러스터 파라미터 그룹과 호환되는 파라미터 그룹 패밀리의 이름을 제공합니다.

유형: 문자열

필수 항목 여부: 아니요

Description

이 클러스터 파라미터 그룹에 대한 고객 설명을 제공합니다.

유형: 문자열

필수 항목 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

DBClusterRole

서비스: Amazon DocumentDB (with MongoDB compatibility)

클러스터와 연결된 AWS Identity and Access Management (IAM) 역할을 설명합니다.

내용

> **Note**
> 다음 목록에는 필수 파라미터가 먼저 설명되어 있습니다.

RoleArn

DB 클러스터와 연결되어 있는 IAMrole의 Amazon 리소스 이름(ARN)입니다.

유형: 문자열

필수 항목 여부: 아니요

Status

IAMrole과 클러스터 간의 연결 상태를 설명합니다. `Status` 속성은 다음 값 중 하나를 반환합니다.

- `ACTIVE` - IAMrole ARN은 클러스터와 연결되며 사용자를 대신하여 다른 AWS 서비스에 액세스하는 데 사용할 수 있습니다.
- `PENDING` - IAMrole ARN을 클러스터와 연결하는 중입니다.
- `INVALID` - IAMrole ARN은 클러스터와 연결되어 있지만 클러스터는 사용자를 대신하여 다른 AWS 서비스에 액세스하기 위해 IAMrole을 수임할 수 없습니다.

유형: 문자열

필수 항목 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS Java V2용 SDK
- AWS Ruby V3용 SDK

DBClusterSnapshot

서비스: Amazon DocumentDB (with MongoDB compatibility)

클러스터 스냅샷에 대한 세부 정보입니다.

내용

> **Note**
> 다음 목록에는, 필수 파라미터가 먼저 설명되어 있습니다.

AvailabilityZones.AvailabilityZone.N

클러스터 스냅샷의 인스턴스를 복원할 수 있는 Amazon EC2 가용 영역 목록을 제공합니다.

유형: 문자열 배열

필수 항목 여부: 아니요

ClusterCreateTime

클러스터가 생성된 시간(협정 세계시(UTC))을 나타냅니다.

유형: 타임스탬프

필수 여부: 아니요

DBClusterIdentifier

이 클러스터 스냅샷을 생성한 클러스터의 클러스터 식별자를 지정합니다.

유형: 문자열

필수 항목 여부: 아니요

DBClusterSnapshotArn

클러스터의 Amazon 리소스 이름(ARN)입니다.

유형: 문자열

필수 항목 여부: 아니요

DBClusterSnapshotIdentifier

클러스터 스냅샷의 식별자를 지정합니다.

유형: 문자열

필수 항목 여부: 아니요

Engine

데이터베이스 엔진의 이름을 지정합니다.

유형: 문자열

필수 항목 여부: 아니요

EngineVersion

이 클러스터 스냅샷에 사용할 데이터베이스 엔진의 버전을 알려 줍니다.

유형: 문자열

필수 항목 여부: 아니요

KmsKeyId

이 `StorageEncrypted`인 경우 암호화된 클러스터 스냅샷의 `true` AWS KMS 키 식별자입니다.

유형: 문자열

필수 항목 여부: 아니요

MasterUsername

클러스터 스냅샷의 마스터 사용자 이름을 알려 줍니다.

유형: 문자열

필수 항목 여부: 아니요

PercentProgress

전송된 데이터의 추정 백분율을 나타냅니다.

유형: 정수

필수 항목 여부: 아니요

Port

스냅샷 생성 시점에 클러스터가 수신하던 포트를 지정합니다.

유형: 정수

필수 항목 여부: 아니요

SnapshotCreateTime

스냅샷이 생성된 시간을 나타냅니다(협정 세계시(UTC)).

유형: 타임스탬프

필수 여부: 아니요

SnapshotType

클러스터 스냅샷의 유형을 알려 줍니다.

유형: 문자열

필수 항목 여부: 아니요

SourceDBClusterSnapshotArn

클러스터 스냅샷을 소스 클러스터 스냅샷에서 복사한 경우 그 소스 클러스터 스냅샷의 ARN이고, 그렇지 않으면 null 값입니다.

유형: 문자열

필수 항목 여부: 아니요

Status

이 클러스터 스냅샷의 상태를 지정합니다.

유형: 문자열

필수 항목 여부: 아니요

StorageEncrypted

클러스터 스냅샷의 암호화 여부를 지정합니다.

유형: 부울

필수 항목 여부: 아니요

StorageType

클러스터 스냅샷과 연결된 스토리지 유형

Amazon DocumentDB 클러스터의 스토리지 유형에 관한 자세한 내용은 Amazon DocumentDB 개발자 안내서의 클러스터 스토리지 구성을 참조하세요.

스토리지 유형에 유효한 값 - `standard | iopt1`

기본값은 `standard` 입니다.

유형: 문자열

필수 항목 여부: 아니요

VpcId

클러스터 스냅샷과 연결된 Virtual Private Cloud(VPC) ID를 제공합니다.

유형: 문자열

필수 항목 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS Java V2용 SDK
- AWS SDK for Ruby V3

DBClusterSnapshotAttribute

서비스: Amazon DocumentDB (with MongoDB compatibility)

수동 클러스터 스냅샷 속성의 이름과 값이 포함되어 있습니다.

수동 클러스터 스냅샷 속성은 다른가 수동 클러스터 스냅샷을 복원할 수 AWS 계정 있는 권한을 부여하는 데 사용됩니다.

내용

> **Note**
>
> 다음 목록에는, 필수 파라미터가 먼저 설명되어 있습니다.

AttributeName

수동 클러스터 스냅샷 속성의 이름입니다.

라는 속성은 수동 클러스터 스냅샷을 복사하거나 복원할 수 AWS 계정 있는 권한이 있는 목록을 `restore` 나타냅니다.

유형: 문자열

필수 항목 여부: 아니요

AttributeValues.AttributeValue.N

수동 클러스터 스냅샷 속성의 값

`AttributeName` 필드가 로 설정된 경우 `restore`이 요소는 수동 클러스터 스냅샷을 복사하거나 복원할 권한이 AWS 계정 있는의 IDs 목록을 반환합니다. 값이 `all` 목록에 있는 경우 수동 클러스터 스냅샷은 퍼블릭이며 복사하거나 복원 AWS 계정 할 수 있습니다.

유형: 문자열 배열

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS Java V2용 SDK
- AWS SDK for Ruby V3

DBClusterSnapshotAttributesResult

서비스: Amazon DocumentDB (with MongoDB compatibility)

클러스터 스냅샷과 관련된 속성에 대한 세부 정보입니다.

내용

> **Note**
> 다음 목록에는, 필수 파라미터가 먼저 설명되어 있습니다.

DBClusterSnapshotAttributes.DBClusterSnapshotAttribute.N

클러스터 스냅샷의 속성과 값 목록입니다.

유형: DBClusterSnapshotAttribute 객체 배열

필수 여부: 아니요

DBClusterSnapshotIdentifier

해당 속성이 적용되는 클러스터 스냅샷의 식별자입니다.

유형: 문자열

필수 항목 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

DBEngineVersion

서비스: Amazon DocumentDB (with MongoDB compatibility)

엔진 버전에 대한 상세 정보입니다.

내용

> **Note**
> 다음 목록에는, 필수 파라미터가 먼저 설명되어 있습니다.

DBEngineDescription

데이터베이스 엔진에 대한 설명입니다.

유형: 문자열

필수 항목 여부: 아니요

DBEngineVersionDescription

데이터베이스 엔진 버전에 대한 설명입니다.

유형: 문자열

필수 항목 여부: 아니요

DBParameterGroupFamily

데이터베이스 엔진에 사용할 파라미터 그룹 패밀리의 이름입니다.

유형: 문자열

필수 항목 여부: 아니요

Engine

데이터베이스 엔진의 이름입니다.

유형: 문자열

필수 항목 여부: 아니요

EngineVersion

데이터베이스 엔진의 버전 번호입니다.

유형: 문자열

필수 항목 여부: 아니요

ExportableLogTypes.member.N

데이터베이스 엔진에서 Amazon CloudWatch Logs로 내보낼 수 있는 로그 유형입니다.

유형: 문자열 배열

필수 여부: 아니요

SupportedCACertificateIdentifiers.member.N

지원되는 CA 인증서 식별자 목록

자세한 내용은 Amazon DocumentDB 개발자 안내서의 Amazon DocumentDB TLS 인증서 업데이트 및 전송 중 데이터 암호화를 참조하십시오.

유형: 문자열 배열

필수 여부: 아니요

SupportsCertificateRotationWithoutRestart

엔진 버전이 DB 인스턴스를 재부팅하지 않고 서버 인증서 교체를 지원하는지를 나타냅니다.

유형: 부울

필수 항목 여부: 아니요

SupportsLogExportsToCloudwatchLogs

CloudWatch Logs로 내보내도록 `ExportableLogTypes`에 지정된 로그 유형을 해당 엔진 버전에서 지원하는지 여부를 나타내는 값입니다.

유형: 부울

필수 항목 여부: 아니요

ValidUpgradeTarget.UpgradeTarget.N

이 데이터베이스 엔진 버전을 업그레이드할 수 있는 엔진 버전의 목록입니다.

유형: UpgradeTarget 객체 배열

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

DBInstance

서비스: Amazon DocumentDB (with MongoDB compatibility)

인스턴스에 대한 자세한 정보입니다.

내용

> **Note**
>
> 다음 목록에는, 필수 파라미터가 먼저 설명되어 있습니다.

AutoMinorVersionUpgrade

적용되지 않습니다. 이 파라미터는 Amazon DocumentDB에는 적용되지 않습니다. Amazon DocumentDB는 값 세트에 관계없이 마이너 버전 업그레이드를 수행하지 않습니다.

유형: 부울

필수 항목 여부: 아니요

AvailabilityZone

인스턴스가 있는 가용 영역의 이름을 지정합니다.

유형: 문자열

필수 항목 여부: 아니요

BackupRetentionPeriod

자동 스냅샷이 보관되는 일수를 지정합니다.

유형: 정수

필수 항목 여부: 아니요

CACertificateIdentifier

이 DB 인스턴스의 CA 인증서 식별자입니다.

유형: 문자열

필수 항목 여부: 아니요

CertificateDetails

DB 인스턴스의 서버 인증서의 세부 정보입니다.

유형: CertificateDetails 객체

필수 여부: 아니요

CopyTagsToSnapshot

태그를 DB 인스턴스에서 DB 인스턴스의 스냅샷으로 복사할지 여부를 나타내는 값입니다. 태그는 기본적으로 복사되지 않습니다.

유형: 부울

필수 항목 여부: 아니요

DBClusterIdentifier

해당 인스턴스가 어떤 클러스터의 멤버인 경우, 그 인스턴스가 속해 있는 클러스터의 이름이 나와 있습니다.

유형: 문자열

필수 항목 여부: 아니요

DBInstanceArn

인스턴스의 Amazon 리소스 이름(ARN)입니다.

유형: 문자열

필수 항목 여부: 아니요

DBInstanceClass

인스턴스의 컴퓨팅 및 메모리 용량 클래스 이름을 포함합니다.

유형: 문자열

필수 항목 여부: 아니요

DBInstanceIdentifier

사용자가 제공한 데이터베이스 ID가 포함되어 있습니다. 이 ID는 인스턴스를 식별하는 고유한 키입니다.

유형: 문자열

필수 항목 여부: 아니요

DBInstanceStatus

이 데이터베이스의 현재 상태를 지정합니다.

유형: 문자열

필수 항목 여부: 아니요

DbiResourceId

인스턴스의 AWS 리전고유하고 변경 불가능한 식별자입니다. 이 식별자는 인스턴스의 AWS KMS 키에 액세스할 때마다 AWS CloudTrail 로그 항목에서 찾을 수 있습니다.

유형: 문자열

필수 항목 여부: 아니요

DBSubnetGroup

이름, 설명, 그리고 서브넷 그룹 내의 서브넷 등 인스턴스와 연결된 서브넷 그룹에 대한 정보를 지정합니다.

유형: DBSubnetGroup 객체

필수 여부: 아니요

EnabledCloudwatchLogsExports.member.N

이 인스턴스에서 CloudWatch Logs로 내보내도록 구성된 로그 유형의 목록입니다.

유형: 문자열 배열

필수 여부: 아니요

Endpoint

연결 엔드포인트를 지정합니다.

유형: Endpoint 객체

필수 여부: 아니요

Engine

이 인스턴스에 사용할 데이터베이스 엔진의 이름을 제공합니다.

유형: 문자열

필수 항목 여부: 아니요

EngineVersion

데이터베이스 엔진의 버전을 나타냅니다.

유형: 문자열

필수 항목 여부: 아니요

InstanceCreateTime

인스턴스를 생성한 날짜 및 시간입니다.

유형: 타임스탬프

필수 여부: 아니요

KmsKeyId

이 `StorageEncrypted`인 경우 암호화된 인스턴스의 `true` AWS KMS 키 식별자입니다.

유형: 문자열

필수 항목 여부: 아니요

LatestRestorableTime

특정 시점으로 복원을 사용하여 데이터베이스를 복원할 수 있는 가장 늦은 시간을 지정합니다.

유형: 타임스탬프

필수 여부: 아니요

PendingModifiedValues

인스턴스에 대한 변경 사항이 대기 중임을 나타냅니다. 이 요소는 변경 사항이 대기 중일 때만 포함됩니다. 구체적인 변경 사항은 하위 요소로 식별됩니다.

유형: PendingModifiedValues 객체

필수 여부: 아니요

PerformanceInsightsEnabled

DB 인스턴스에 대해 Amazon RDS Performance Insights가 활성 상태이면 `true`이고, 그렇지 않으면 `false`로 설정합니다.

유형: 부울

필수 항목 여부: 아니요

PerformanceInsightsKMSKeyId

성능 개선 도우미 데이터의 암호화를 위한 AWS KMS 키 식별자입니다. AWS KMS 키 ID는 Amazon 리소스 이름(ARN), AWS KMS 키 식별자 또는 AWS KMS 암호화 AWS KMS 키의 키 별칭입니다.

유형: 문자열

필수 항목 여부: 아니요

PreferredBackupWindow

자동 백업이 활성화된 경우 자동 백업이 생성되는 일일 시간 범위를 나타내며, `BackupRetentionPeriod` 속성에 의해 결정됩니다.

유형: 문자열

필수 항목 여부: 아니요

PreferredMaintenanceWindow

시스템 유지 관리를 실행할 수 있는 주 단위 기간(UTC, 협정 세계시)을 지정합니다.

유형: 문자열

필수 항목 여부: 아니요

PromotionTier

기존 기본 인스턴스에 결함이 발생한 후 Amazon DocumentDB 복제본을 기본 인스턴스로 승격할 순서를 지정하는 값

유형: 정수

필수 항목 여부: 아니요

PubliclyAccessible

지원하지 않음. Amazon DocumentDB는 현재 퍼블릭 엔드포인트를 지원하지 않습니다. `PubliclyAccessible` 값은 항상 `false`입니다.

유형: 부울

필수 항목 여부: 아니요

StatusInfos.DBInstanceStatusInfo.N

읽기 전용 복제본의 상태입니다. 인스턴스가 읽기 전용 복제본이 아닌 경우 비어 있습니다.

유형: DBInstanceStatusInfo 객체 배열

필수 여부: 아니요

StorageEncrypted

인스턴스의 암호화 여부를 지정합니다.

유형: 부울

필수 항목 여부: 아니요

VpcSecurityGroups.VpcSecurityGroupMembership.N

인스턴스가 속해 있는 VPC 보안 그룹 요소의 목록을 제공합니다.

유형: VpcSecurityGroupMembership 객체 배열

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS Java V2용 SDK
- AWS SDK for Ruby V3

DBInstanceStatusInfo

서비스: Amazon DocumentDB (with MongoDB compatibility)

인스턴스에 대한 상태 정보 목록을 제공합니다.

내용

> **Note**
>
> 다음 목록에서는 필수 파라미터가 먼저 설명되어 있습니다.

Message

인스턴스에 오류가 있는 경우, 오류에 대한 세부 정보입니다. 인스턴스가 오류 상태가 아니면 이 값은 비어 있습니다.

유형: 문자열

필수 항목 여부: 아니요

Normal

인스턴스가 정상 작동 중이면 `true`이고 인스턴스가 오류 상태이면 `false`인 부울 값입니다.

유형: 부울

필수 항목 여부: 아니요

Status

인스턴스의 상태입니다. 읽기 전용 `StatusType` 복제본의 경우 값은 `replicating`, error, `stopped` 또는 `terminated`일 수 있습니다.

유형: 문자열

필수 항목 여부: 아니요

StatusType

현재 이 값은 “`read replication`“입니다.

유형: 문자열

필수 항목 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

DBSubnetGroup

서비스: Amazon DocumentDB (with MongoDB compatibility)

서브넷 그룹에 대한 자세한 정보.

내용

> **Note**
> 다음 목록에서는 필수 파라미터가 먼저 설명되어 있습니다.

DBSubnetGroupArn

DB 서브넷 그룹의 Amazon 리소스 이름(ARN)입니다.

유형: 문자열

필수 항목 여부: 아니요

DBSubnetGroupDescription

서브넷 그룹에 대한 설명을 제공합니다.

유형: 문자열

필수 항목 여부: 아니요

DBSubnetGroupName

서브넷 그룹의 이름입니다.

유형: 문자열

필수 항목 여부: 아니요

SubnetGroupStatus

서브넷 그룹의 상태를 제공합니다.

유형: 문자열

필수 항목 여부: 아니요

Subnets.Subnet.N

서브넷 그룹에 있는 하나 이상의 서브넷에 대한 세부적인 정보입니다.

유형: Subnet 객체 배열

필수 여부: 아니요

VpcId

서브넷 그룹의 Virtual Private Cloud (VPC)의 ID를 제공합니다.

유형: 문자열

필수 항목 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Endpoint

서비스: Amazon DocumentDB (with MongoDB compatibility)

클러스터 또는 인스턴스에 액세스하기 위한 네트워크 정보. 클라이언트 프로그램이 이러한 Amazon DocumentDB 리소스에 액세스하려면 유효한 엔드포인트를 지정해야 합니다.

내용

> **Note**
>
> 다음 목록에는 필수 파라미터가 먼저 설명되어 있습니다.

Address

인스턴스의 DNS 주소를 지정합니다.

유형: 문자열

필수 항목 여부: 아니요

HostedZoneId

호스팅 영역을 생성할 때 Amazon Route 53에서 할당하는 ID를 나타냅니다.

유형: 문자열

필수 항목 여부: 아니요

Port

데이터베이스 엔진이 수신 대기하는 포트를 지정합니다.

유형: 정수

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2

- AWS SDK for Ruby V3

EngineDefaults

서비스: Amazon DocumentDB (with MongoDB compatibility)

`DescribeEngineDefaultClusterParameters` 작업의 성공적인 간접 호출의 결과가 포함되어 있습니다.

내용

> **Note**
>
> 다음 목록에서는 필수 파라미터에 대해 먼저 설명합니다.

DBParameterGroupFamily

엔진 파라미터 정보를 반환할 클러스터 파라미터 그룹 패밀리의 이름입니다.

유형: 문자열

필수 항목 여부: 아니요

Marker

이전의 요청에서 제공된 선택적 페이지 매김 토큰입니다. 이 파라미터를 지정한 경우, 마커 이후부터 `MaxRecords`에 지정된 값까지의 레코드만 응답에 포함됩니다.

유형: 문자열

필수 항목 여부: 아니요

Parameters.Parameter.N

특정 클러스터 파라미터 그룹 패밀리의 파라미터.

유형: Parameter 객체 배열

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++

- AWS SDK for Java V2
- AWS SDK for Ruby V3

Event

서비스: Amazon DocumentDB (with MongoDB compatibility)

이벤트에 대한 자세한 정보.

내용

> **Note**
> 다음 목록에서는 필수 파라미터에 대해 먼저 설명합니다.

Date

이벤트 날짜 및 시간을 지정합니다.

유형: 타임스탬프

필수 여부: 아니요

EventCategories.EventCategory.N

이 이벤트의 범주를 지정합니다.

유형: 문자열 배열

필수 여부: 아니요

Message

이 이벤트의 텍스트를 제공합니다.

유형: 문자열

필수 항목 여부: 아니요

SourceArn

이벤트의 Amazon 리소스 이름(ARN)입니다.

유형: 문자열

필수 항목 여부: 아니요

SourceIdentifier

이벤트 소스의 식별자를 제공합니다.

유형: 문자열

필수 항목 여부: 아니요

SourceType

이 이벤트의 소스 유형을 지정합니다.

타입: 문자열

유효 값: `db-instance | db-parameter-group | db-security-group | db-snapshot | db-cluster | db-cluster-snapshot`

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS Java V2용 SDK
- AWS SDK for Ruby V3

EventCategoriesMap

서비스: Amazon DocumentDB (with MongoDB compatibility)

하나 이상의 이벤트 범주 이름이 포함된 이벤트 소스 유형입니다.

내용

> **Note**
> 다음 목록에는 필수 파라미터가 먼저 설명되어 있습니다.

EventCategories.EventCategory.N

지정된 소스 유형의 이벤트 범주입니다.

유형: 문자열 배열

필수 여부: 아니요

SourceType

반환된 범주의 소스 유형입니다.

유형: 문자열

필수 항목 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

EventSubscription

서비스: Amazon DocumentDB (with MongoDB compatibility)

구독한 이벤트에 대한 세부 정보.

내용

> **Note**
> 필수 파라미터는 다음 표에 처음으로 설명되어 있습니다.

CustomerAwsId

Amazon DocumentDB 이벤트 알림 구독과 연결된 AWS 고객 계정입니다.

유형: 문자열

필수 항목 여부: 아니요

CustSubscriptionId

Amazon DocumentDB 이벤트 알림 구독 ID.

유형: 문자열

필수 항목 여부: 아니요

Enabled

구독의 활성화 여부를 나타내는 부울 값. `true`의 값은 구독이 활성화되었음을 나타냅니다.

유형: 부울

필수 항목 여부: 아니요

EventCategoriesList.EventCategory.N

Amazon DocumentDB 이벤트 알림 구독에 대한 이벤트 카테고리의 목록.

유형: 문자열 배열

필수 여부: 아니요

EventSubscriptionArn

이벤트 구독의 Amazon 리소스 이름(ARN)입니다.

유형: 문자열

필수 항목 여부: 아니요

SnsTopicArn

Amazon DocumentDB 이벤트 알림 구독의 주제 ARN.

유형: 문자열

필수 항목 여부: 아니요

SourceIdsList.SourceId.N

Amazon DocumentDB 이벤트 알림 구독의 소스 ID 목록.

유형: 문자열 배열

필수 여부: 아니요

SourceType

Amazon DocumentDB 이벤트 알림 구독의 소스 유형.

유형: 문자열

필수 항목 여부: 아니요

Status

Amazon DocumentDB 이벤트 알림 구독의 상태.

제약 조건:

`creating`, `modifying`, `deleting`, `active`, `no-permission`, `topic-not-exist` 중 하나가 될 수 있습니다.

`no-permission` 상태는 Amazon DocumentDB가 더 이상 SNS 주제에 게시할 수 있는 권한이 없음을 나타냅니다. `topic-not-exist` 상태는 구독이 생성된 후 주제가 삭제되었음을 나타냅니다.

유형: 문자열

필수 항목 여부: 아니요

SubscriptionCreationTime

Amazon DocumentDB 이벤트 알림 구독이 생성된 시간.

유형: 문자열

필수 항목 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS Java V2용 SDK
- AWS SDK for Ruby V3

Filter

서비스: Amazon DocumentDB (with MongoDB compatibility)

보다 구체적인 결과 목록을 반환하는 데 사용되는 이름이 지정된 필터 값 세트. 필터를 사용하여 ID와 같은 특정 기준에 따라 리소스 집합을 일치시킬 수 있습니다.

와일드카드는 필터에서 지원되지 않습니다.

내용

> **Note**
>
> 다음 목록에서 필요한 파라미터가 먼저 설명되어 있습니다.

Name

필터의 이름. 필터 이름은 대/소문자를 구분합니다.

유형: 문자열

필수 항목 여부: 예

Values.Value.N

하나 이상의 필터 값. 필터 값은 대소문자를 구분합니다.

유형: 문자열 배열

필수 여부: 예

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

GlobalCluster

서비스: Amazon DocumentDB (with MongoDB compatibility)

Amazon DocumentDB 글로벌 클러스터를 나타내는 데이터 유형.

내용

> **Note**
> 필수 파라미터는 다음 목록에 먼저 설명되어 있습니다.

DatabaseName

새 글로벌 클러스터 내의 기본 데이터베이스 이름.

유형: 문자열

필수 항목 여부: 아니요

DeletionProtection

새 글로벌 클러스터에 대한 삭제 방지 설정.

유형: 부울

필수 항목 여부: 아니요

Engine

글로벌 클러스터에서 사용하는 Amazon DocumentDB 데이터베이스 엔진.

유형: 문자열

필수 항목 여부: 아니요

EngineVersion

데이터베이스 엔진의 버전을 나타냅니다.

유형: 문자열

필수 항목 여부: 아니요

GlobalClusterArn

글로벌 클러스터의 Amazon 리소스 이름(ARN).

유형: 문자열

필수 항목 여부: 아니요

GlobalClusterIdentifier

사용자가 제공한 글로벌 클러스터 식별자가 포함되어 있습니다. 이 식별자는 글로벌 클러스터를 식별하는 고유한 키입니다.

유형: 문자열

길이 제약: 최소 길이는 1. 최대 길이는 255.

패턴: `[A-Za-z][0-9A-Za-z-:._]*`

Required: No

GlobalClusterMembers.GlobalClusterMember.N

글로벌 클러스터 내 보조 클러스터의 클러스터 ID 목록. 현재 한 항목으로 제한됨.

유형: GlobalClusterMember 객체 배열

필수 여부: 아니요

GlobalClusterResourceId

글로벌 데이터베이스 클러스터의 AWS 리전리전 고유하고 변경 불가능한 식별자입니다. 이 식별자는 클러스터의 AWS KMS 고객 마스터 키(CMK)에 액세스할 때마다 AWS CloudTrail 로그 항목에서 찾을 수 있습니다.

유형: 문자열

필수 항목 여부: 아니요

Status

이 글로벌 클러스터의 현재 상태를 지정합니다.

유형: 문자열

필수 항목 여부: 아니요

StorageEncrypted

글로벌 클러스터의 스토리지 암호화 설정.

유형: 부울

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

GlobalClusterMember

서비스: Amazon DocumentDB (with MongoDB compatibility)

Amazon DocumentDB 글로벌 클러스터와 관련된 모든 기본 및 보조 클러스터에 대한 정보가 포함된 데이터 구조입니다.

내용

> **Note**
>
> 다음 목록에는, 필수 파라미터가 먼저 설명되어 있습니다.

DBClusterArn

각 Amazon DocumentDB 클러스터의 Amazon 리소스 이름(ARN)

유형: 문자열

필수 항목 여부: 아니요

IsWriter

Amazon DocumentDB 클러스터가 해당 클러스터와 연결된 Amazon DocumentDB 글로벌 클러스터의 기본 클러스터인지 (즉, 읽기-쓰기 기능이 있는지) 여부를 지정합니다.

유형: 부울

필수 항목 여부: 아니요

Readers.member.N

Amazon DocumentDB 글로벌 클러스터와 연결된 각 읽기 전용 보조 클러스터의 Amazon 리소스 이름(ARN)입니다.

유형: 문자열 배열

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS Java V2용 SDK
- AWS SDK for Ruby V3

OrderableDBInstanceOption

서비스: Amazon DocumentDB (with MongoDB compatibility)

인스턴스에 사용할 수 있는 옵션.

내용

> **Note**
> 다음 목록에는, 필수 파라미터가 먼저 설명되어 있습니다.

AvailabilityZones.AvailabilityZone.N

인스턴스의 가용 영역 목록입니다.

유형: AvailabilityZone 객체 배열

필수 여부: 아니요

DBInstanceClass

인스턴스의 인스턴스 클래스입니다.

유형: 문자열

필수 항목 여부: 아니요

Engine

인스턴스의 엔진 유형입니다.

유형: 문자열

필수 항목 여부: 아니요

EngineVersion

인스턴스의 엔진 버전입니다.

유형: 문자열

필수 항목 여부: 아니요

LicenseModel

인스턴스의 라이선스 모델입니다.

유형: 문자열

필수 항목 여부: 아니요

StorageType

DB 클러스터와 연결할 스토리지 유형

유형: 문자열

필수 항목 여부: 아니요

Vpc

인스턴스가 Virtual Private Cloud(VPC)에 있는지 여부를 나타냅니다.

유형: 부울

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Parameter

서비스: Amazon DocumentDB (with MongoDB compatibility)

개별 파라미터에 대한 상세 정보

내용

> **Note**
> 다음 목록에는, 필수 파라미터가 먼저 설명되어 있습니다.

AllowedValues

파라미터의 유효한 값 범위를 지정합니다.

유형: 문자열

필수 항목 여부: 아니요

ApplyMethod

파라미터 업데이트의 적용 시점을 나타냅니다.

타입: 문자열

유효 값: `immediate | pending-reboot`

필수 여부: 아니요

ApplyType

엔진별 파라미터 유형을 지정합니다.

유형: 문자열

필수 항목 여부: 아니요

DataType

파라미터의 유효한 데이터 형식을 지정합니다.

유형: 문자열

필수 항목 여부: 아니요

Description

파라미터의 설명을 제공합니다.

유형: 문자열

필수 항목 여부: 아니요

IsModifiable

파라미터를 수정할 수 있는지 여부(`true`) 또는 (`false`)를 나타냅니다. 일부 파라미터에는 보안 또는 작동상의 의미가 있어 변경할 수 없습니다.

유형: 부울

필수 항목 여부: 아니요

MinimumEngineVersion

파라미터를 적용할 수 있는 가장 빠른 엔진 버전입니다.

유형: 문자열

필수 항목 여부: 아니요

ParameterName

파라미터의 이름을 지정합니다.

유형: 문자열

필수 항목 여부: 아니요

ParameterValue

파라미터의 값을 지정합니다.

유형: 문자열

필수 항목 여부: 아니요

Source

파라미터 값의 소스를 나타냅니다.

유형: 문자열

필수 항목 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

PendingCloudwatchLogsExports

서비스: Amazon DocumentDB (with MongoDB compatibility)

구성이 계속 보류 중인 로그 유형의 목록입니다. 이러한 로그 유형은 활성화되거나 비활성화되는 중입니다.

내용

> **Note**
>
> 다음 목록에는, 필수 파라미터가 먼저 설명되어 있습니다.

LogTypesToDisable.member.N

활성화되는 중인 로그 유형입니다. 활성화된 후에는 이러한 로그 유형은 Amazon CloudWatch Logs로 내보냅니다.

유형: 문자열 배열

필수 여부: 아니요

LogTypesToEnable.member.N

비활성화되는 중인 로그 유형입니다. 비활성화된 후에는 이러한 로그 유형은 CloudWatch Logs로 내보내지 않습니다.

유형: 문자열 배열

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

PendingMaintenanceAction

서비스: Amazon DocumentDB (with MongoDB compatibility)

대기 중인 리소스 유지 관리 작업에 대한 정보를 제공합니다.

내용

> **Note**
> 다음 목록에는 필요한 파라미터가 먼저 설명되어 있습니다.

Action

해당 리소스에 사용 가능한 대기 중 유지 관리 작업의 유형입니다.

유형: 문자열

필수 항목 여부: 아니요

AutoAppliedAfterDate

작업을 적용할 유지 관리 기간의 날짜입니다. 이 날짜 이후의 첫 번째 유지 관리 기간에 해당 리소스에 유지 관리 작업을 적용합니다. 이 날짜를 지정하면 모든 `next-maintenance` 옵트인 요청은 무시됩니다.

유형: 타임스탬프

필수 여부: 아니요

CurrentApplyDate

대기 중 유지 관리 작업을 리소스에 적용한 발효 날짜입니다.

유형: 타임스탬프

필수 여부: 아니요

Description

유지 관리 작업에 대한 세부 정보를 담은 설명입니다.

유형: 문자열

필수 항목 여부: 아니요

ForcedApplyDate

유지 관리 작업이 자동으로 적용되는 날짜입니다. 해당 리소스의 유지 관리 기간과 관계없이 이 날짜에 해당 리소스에 유지 관리 작업을 적용합니다. 이 날짜를 지정하면 모든 `immediate` 옵트인 요청은 무시됩니다.

유형: 타임스탬프

필수 여부: 아니요

OptInStatus

해당 리소스에 대해 수신된 옵트인 요청의 유형을 나타냅니다.

유형: 문자열

필수 항목 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

PendingModifiedValues

서비스: Amazon DocumentDB (with MongoDB compatibility)

인스턴스에 대한 하나 이상의 수정된 설정 이러한 수정된 설정이 요청되었지만 아직 적용되지 않았습니다.

내용

> **Note**
>
> 다음 목록에는, 필수 파라미터가 먼저 설명되어 있습니다.

AllocatedStorage

현재 적용 중이거나 적용할 예정인 인스턴스의 새로운 `AllocatedStorage` 크기가 포함되어 있습니다.

유형: 정수

필수 항목 여부: 아니요

BackupRetentionPeriod

자동 백업이 보관되는 대기 중 일수를 지정합니다.

유형: 정수

필수 항목 여부: 아니요

CACertificateIdentifier

DB 인스턴스의 인증 기관(CA) 인증서 식별자를 지정합니다.

유형: 문자열

필수 항목 여부: 아니요

DBInstanceClass

현재 적용 중이거나 적용할 예정인 인스턴스의 새로운 `DBInstanceClass`가 포함되어 있습니다.

유형: 문자열

필수 항목 여부: 아니요

DBInstanceIdentifier

현재 적용 중이거나 적용할 예정인 인스턴스의 새로운 `DBInstanceIdentifier`가 포함되어 있습니다.

유형: 문자열

필수 항목 여부: 아니요

DBSubnetGroupName

인스턴스의 새 서브넷 그룹입니다.

유형: 문자열

필수 항목 여부: 아니요

EngineVersion

데이터베이스 엔진의 버전을 나타냅니다.

유형: 문자열

필수 항목 여부: 아니요

Iops

현재 적용 중이거나 적용할 예정인 인스턴스의 새로운 프로비저닝된 IOPS 값을 지정합니다.

유형: 정수

필수 항목 여부: 아니요

LicenseModel

인스턴스의 라이선스 모델입니다.

유효한 값: `license-included`, `bring-your-own-license`, `general-public-license`

유형: 문자열

필수 항목 여부: 아니요

MasterUserPassword

대기 중이거나 현재 진행 중인 인스턴스의 마스터 보안 인증에 대한 변경 사항이 포함되어 있습니다.

유형: 문자열

필수 항목 여부: 아니요

MultiAZ

단일 AZ 인스턴스를 다중 AZ 배포로 변경할 것인지를 나타냅니다.

유형: 부울

필수 항목 여부: 아니요

PendingCloudwatchLogsExports

구성이 계속 보류 중인 로그 유형의 목록입니다. 이러한 로그 유형은 활성화되거나 비활성화되는 중입니다.

유형: PendingCloudwatchLogsExports 객체

필수 여부: 아니요

Port

인스턴스에 대한 대기 중 포트를 지정합니다.

유형: 정수

필수 항목 여부: 아니요

StorageType

인스턴스와 연결할 스토리지 유형을 지정합니다.

유형: 문자열

필수 항목 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ResourcePendingMaintenanceActions

서비스: Amazon DocumentDB (with MongoDB compatibility)

의 출력을 나타냅니다ApplyPendingMaintenanceAction.

내용

> **Note**
> 다음 목록에서는 필수 파라미터가 먼저 설명되어 있습니다.

PendingMaintenanceActionDetails.PendingMaintenanceAction.N

리소스에 대해 대기 중인 유지 관리 작업의 세부 정보를 담은 목록입니다.

유형: PendingMaintenanceAction 객체 배열

필수 여부: 아니요

ResourceIdentifier

보류 중인 유지 관리 작업이 있는 리소스의 Amazon 리소스 이름(ARN)입니다.

유형: 문자열

필수 항목 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Subnet

서비스: Amazon DocumentDB (with MongoDB compatibility)

서브넷에 대한 세부 정보입니다.

내용

> **Note**
> 다음 목록에서는 필수 파라미터가 먼저 설명되어 있습니다.

SubnetAvailabilityZone

서브넷에 대한 가용 영역을 사용합니다.

유형: AvailabilityZone 객체

필수 여부: 아니요

SubnetIdentifier

서브넷의 식별자를 지정합니다.

유형: 문자열

필수 항목 여부: 아니요

SubnetStatus

서브넷의 상태를 지정합니다.

유형: 문자열

필수 항목 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2

- AWS SDK for Ruby V3

Tag

서비스: Amazon DocumentDB (with MongoDB compatibility)

키-값 페어로 구성된 Amazon DocumentDB 리소스에 할당되는 메타데이터입니다.

내용

> **Note**
>
> 다음 목록에서는 필수 파라미터가 먼저 설명되어 있습니다.

Key

태그의 필수 이름입니다. 문자열 값은 길이가 1~128자(유니코드 문자)이며 "aws:" 또는 "rds:"로 시작할 수 없습니다. 문자열에는 유니코드 문자 집합, 숫자, 공백, '_', '.', '/', '=', '+', '-'(Java regex: "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$")만 포함될 수 있습니다.

유형: 문자열

필수 항목 여부: 아니요

Value

태그의 선택적 값입니다. 문자열 값은 길이가 1~256자(유니코드 문자)이며 "aws:" 또는 "rds:"로 시작할 수 없습니다. 문자열에는 유니코드 문자 집합, 숫자, 공백, '_', '.', '/', '=', '+', '-'(Java regex: "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$")만 포함될 수 있습니다.

유형: 문자열

필수 항목 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

UpgradeTarget

서비스: Amazon DocumentDB (with MongoDB compatibility)

인스턴스를 업그레이드할 수 있는 데이터베이스 엔진의 버전입니다.

내용

> **Note**
>
> 필수 파라미터는 다음 목록에 설명되어 있습니다.

AutoUpgrade

`AutoMinorVersionUpgrade`가 `true`로 설정되어 있는 소스 DB 인스턴스에 대상 버전을 적용할 것인지 여부를 나타내는 값입니다.

유형: 부울

필수 항목 여부: 아니요

Description

인스턴스를 업그레이드할 수 있는 데이터베이스 엔진의 버전입니다.

유형: 문자열

필수 항목 여부: 아니요

Engine

업그레이드 대상인 데이터베이스 엔진의 이름입니다.

유형: 문자열

필수 항목 여부: 아니요

EngineVersion

업그레이드 대상인 데이터베이스 엔진의 버전 번호입니다.

유형: 문자열

필수 항목 여부: 아니요

IsMajorVersionUpgrade

데이터베이스 엔진을 메이저 버전으로 업그레이드할 것인지 여부를 나타내는 값입니다.

유형: 부울

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

VpcSecurityGroupMembership

서비스: Amazon DocumentDB (with MongoDB compatibility)

Virtual Private Cloud (VPC) 보안 그룹 멤버십에 대한 쿼리의 응답 요소로 사용됩니다.

내용

> **Note**
>
> 다음 목록에서는 필수 파라미터가 먼저 설명되어 있습니다.

Status

VPC 보안 그룹의 상태입니다.

유형: 문자열

필수 항목 여부: 아니요

VpcSecurityGroupId

VPC 보안 그룹의 이름입니다.

유형: 문자열

필수 항목 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Amazon DocumentDB Elastic Clusters

Amazon DocumentDB Elastic Clusters에서 지원되는 데이터 유형은 다음과 같습니다.

- Cluster

- ClusterInList
- ClusterSnapshot
- ClusterSnapshotInList
- PendingMaintenanceActionDetails
- ResourcePendingMaintenanceAction
- Shard
- ValidationExceptionField

Cluster

서비스: Amazon DocumentDB Elastic Clusters

특정 탄력적 클러스터에 대한 정보를 반환합니다.

내용

> **Note**
> 다음 목록에는 필수 파라미터가 먼저 설명되어 있습니다.

adminUserName

탄성 클러스터 관리자의 이름입니다.

유형: 문자열

필수 항목 여부: 예

authType

탄력적 클러스터의 인증 유형입니다.

타입: 문자열

유효 값: `PLAIN_TEXT | SECRET_ARN`

필수 사항 여부: 예

clusterArn

탄력적 클러스터의 ARN 식별자.

유형: 문자열

필수 항목 여부: 예

clusterEndpoint

탄력적 클러스터에 연결하는 데 사용되는 URL입니다.

유형: 문자열

필수 항목 여부: 예

clusterName

Elastic 클러스터의 이름입니다.

유형: 문자열

필수 항목 여부: 예

createTime

탄력적 클러스터가 생성된 시간(협정 세계시(UTC)).

유형: 문자열

필수 항목 여부: 예

kmsKeyId

탄력적 클러스터를 암호화하는 데 사용할 KMS 키 식별자입니다.

유형: 문자열

필수 항목 여부: 예

preferredMaintenanceWindow

시스템 유지 관리를 실행할 수 있는 주 단위 기간(UTC, 협정 세계시)입니다.

형식: `ddd:hh24:mi-ddd:hh24:mi`

유형: 문자열

필수 항목 여부: 예

shardCapacity

각 탄력적 클러스터 샤드에 할당된 vCPU 수 최대 값은 64입니다. 허용되는 값은 2, 4, 8, 16, 32, 64입니다.

유형: 정수

필수 여부: 예

shardCount

탄력적 클러스터에 할당된 샤드 수. 최대 값은 32입니다.

유형: 정수

필수 여부: 예

status

Elastic 클러스터의 상태입니다.

타입: 문자열

유효 값: `CREATING | ACTIVE | DELETING | UPDATING | VPC_ENDPOINT_LIMIT_EXCEEDED | IP_ADDRESS_LIMIT_EXCEEDED | INVALID_SECURITY_GROUP_ID | INVALID_SUBNET_ID | INACCESSIBLE_ENCRYPTION_CREDS | INACCESSIBLE_SECRET_ARN | INACCESSIBLE_VPC_ENDPOINT | INCOMPATIBLE_NETWORK | MERGING | MODIFYING | SPLITTING | COPYING | STARTING | STOPPING | STOPPED | MAINTENANCE | INACCESSIBLE_ENCRYPTION_CREDENTIALS_RECOVERABLE`

필수 사항 여부: 예

subnetIds

탄력적 클러스터의 Amazon EC2 서브넷 ID입니다.

유형: 문자열 배열

필수 여부: 예

vpcSecurityGroupIds

탄력적 클러스터와 연결된 EC2 VPC 보안 그룹 목록입니다.

유형: 문자열 배열

필수 여부: 예

backupRetentionPeriod

자동 스냅샷이 보관되는 일수입니다.

유형: 정수

필수 항목 여부: 아니요

preferredBackupWindow

자동 백업이 활성화된 경우 자동 백업이 생성되는 일일 시간 범위이며, `backupRetentionPeriod` 속성에 의해 결정됩니다.

유형: 문자열

필수 항목 여부: 아니요

shardInstanceCount

클러스터의 모든 샤드에 적용되는 복제 인스턴스 수입니다. `shardInstanceCount` 값이 1이면 라이터 인스턴스가 하나라는 의미이며, 추가 인스턴스는 읽기에 사용할 수 있고 가용성을 개선하는 복제본입니다.

유형: 정수

필수 항목 여부: 아니요

shards

클러스터의 샤드 총 개수입니다.

유형: Shard 객체 배열

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ClusterInList

서비스: Amazon DocumentDB Elastic Clusters

Amazon DocumentDB Elastic Clusters.

내용

> **Note**
>
> 필수 파라미터는 다음 표에 처음으로 설명되어 있습니다.

clusterArn

탄력적 클러스터의 ARN 식별자.

유형: 문자열

필수 항목 여부: 예

clusterName

Elastic 클러스터의 이름입니다.

유형: 문자열

필수 항목 여부: 예

status

Elastic 클러스터의 상태입니다.

타입: 문자열

유효 값: `CREATING | ACTIVE | DELETING | UPDATING | VPC_ENDPOINT_LIMIT_EXCEEDED | IP_ADDRESS_LIMIT_EXCEEDED | INVALID_SECURITY_GROUP_ID | INVALID_SUBNET_ID | INACCESSIBLE_ENCRYPTION_CREDS | INACCESSIBLE_SECRET_ARN | INACCESSIBLE_VPC_ENDPOINT | INCOMPATIBLE_NETWORK | MERGING | MODIFYING | SPLITTING | COPYING | STARTING | STOPPING | STOPPED | MAINTENANCE | INACCESSIBLE_ENCRYPTION_CREDENTIALS_RECOVERABLE`

필수 여부: 예

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ClusterSnapshot

서비스: Amazon DocumentDB Elastic Clusters

특정 탄력적 클러스터 스냅샷에 대한 정보를 반환합니다.

내용

> **Note**
> 다음 목록에는 필수 파라미터가 먼저 설명되어 있습니다.

adminUserName

탄성 클러스터 관리자의 이름입니다.

유형: 문자열

필수 항목 여부: 예

clusterArn

탄력적 클러스터의 ARN 식별자.

유형: 문자열

필수 항목 여부: 예

clusterCreationTime

탄력적 클러스터가 생성된 시간(협정 세계시(UTC)).

유형: 문자열

필수 항목 여부: 예

kmsKeyId

KMS 키 식별자는 KMS 암호화 키의 Amazon 리소스 이름(ARN)입니다. 이 KMS 암호화 키를 소유하고 있는 동일한 Amazon 계정을 사용하여 클러스터를 생성하는 경우, ARN 대신 KMS 키 별칭을 KMS 암호화 키로 사용할 수 있습니다. 여기에 암호화 키가 지정되지 않은 경우 Amazon DocumentDB는 KMS가 사용자 계정에 대해 생성하는 기본 암호화 키를 사용합니다. 계정에는 Amazon 리전마다 다른 기본 암호화 키가 있습니다.

유형: 문자열

필수 항목 여부: 예

snapshotArn

탄력적 클러스터 스냅샷의 ARN 식별자.

유형: 문자열

필수 항목 여부: 예

snapshotCreationTime

탄력적 클러스터 스냅샷이 생성된 시간(협정 세계시(UTC)).

유형: 문자열

필수 항목 여부: 예

snapshotName

탄력적 클러스터 스냅샷의 이름.

유형: 문자열

필수 항목 여부: 예

status

탄력적 클러스터 스냅샷의 상태.

타입: 문자열

유효 값: `CREATING | ACTIVE | DELETING | UPDATING | VPC_ENDPOINT_LIMIT_EXCEEDED | IP_ADDRESS_LIMIT_EXCEEDED | INVALID_SECURITY_GROUP_ID | INVALID_SUBNET_ID | INACCESSIBLE_ENCRYPTION_CREDS | INACCESSIBLE_SECRET_ARN | INACCESSIBLE_VPC_ENDPOINT | INCOMPATIBLE_NETWORK | MERGING | MODIFYING | SPLITTING | COPYING | STARTING | STOPPING | STOPPED | MAINTENANCE | INACCESSIBLE_ENCRYPTION_CREDENTIALS_RECOVERABLE`

필수 사항 여부: 예

subnetIds

탄력적 클러스터의 Amazon EC2 서브넷 ID입니다.

유형: 문자열 배열

필수 여부: 예

vpcSecurityGroupIds

탄력적 클러스터에 연결할 EC2 VPC 보안 그룹의 목록.

유형: 문자열 배열

필수 여부: 예

snapshotType

반환되는 클러스터 스냅샷의 유형. 다음 값 중 하나를 지정할 수 있습니다.

- `automated` - Amazon DocumentDB가 AWS 계정에 대해 자동으로 생성한 모든 클러스터 스냅샷을 반환합니다.
- `manual` - AWS 계정에 대해 수동으로 생성한 모든 클러스터 스냅샷을 반환합니다.

타입: 문자열

유효 값: `MANUAL | AUTOMATED`

필수 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ClusterSnapshotInList

서비스: Amazon DocumentDB Elastic Clusters

탄력적 클러스터 스냅샷의 목록.

내용

> **Note**
> 필수 파라미터는 다음 표에 처음으로 설명되어 있습니다.

clusterArn

탄력적 클러스터의 ARN 식별자.

유형: 문자열

필수 항목 여부: 예

snapshotArn

탄력적 클러스터 스냅샷의 ARN 식별자.

유형: 문자열

필수 항목 여부: 예

snapshotCreationTime

탄력적 클러스터 스냅샷이 생성된 시간(협정 세계시(UTC)).

유형: 문자열

필수 항목 여부: 예

snapshotName

탄력적 클러스터 스냅샷의 이름.

유형: 문자열

필수 항목 여부: 예

status

탄력적 클러스터 스냅샷의 상태.

타입: 문자열

유효 값: `CREATING | ACTIVE | DELETING | UPDATING | VPC_ENDPOINT_LIMIT_EXCEEDED | IP_ADDRESS_LIMIT_EXCEEDED | INVALID_SECURITY_GROUP_ID | INVALID_SUBNET_ID | INACCESSIBLE_ENCRYPTION_CREDS | INACCESSIBLE_SECRET_ARN | INACCESSIBLE_VPC_ENDPOINT | INCOMPATIBLE_NETWORK | MERGING | MODIFYING | SPLITTING | COPYING | STARTING | STOPPING | STOPPED | MAINTENANCE | INACCESSIBLE_ENCRYPTION_CREDENTIALS_RECOVERABLE`

필수 여부: 예

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

PendingMaintenanceActionDetails

서비스: Amazon DocumentDB Elastic Clusters

보류 중인 유지 관리 작업의 세부 정보를 검색합니다.

내용

> **Note**
>
> 필수 파라미터는 다음 표에 처음으로 설명되어 있습니다.

action

보류 중인 유지 관리 작업의 특정 작업을 표시합니다.

유형: 문자열

필수 항목 여부: 예

autoAppliedAfterDate

작업이 적용될 때 유지 관리 기간의 날짜를 표시합니다. 이 날짜 이후의 첫 번째 유지 관리 기간에 해당 리소스에 유지 관리 작업을 적용합니다. 이 날짜를 지정하면 모든 `NEXT_MAINTENANCE optInType` 요청이 무시됩니다.

유형: 문자열

필수 항목 여부: 아니요

currentApplyDate

보류 중인 유지 관리 작업이 리소스에 적용되는 유효 날짜를 표시합니다.

유형: 문자열

필수 항목 여부: 아니요

description

유지 관리 작업에 대한 자세한 내용을 제공하는 설명을 표시합니다.

유형: 문자열

필수 항목 여부: 아니요

forcedApplyDate

유지 관리 작업이 자동으로 적용되는 날짜를 표시합니다. 해당 리소스의 유지 관리 기간과 관계없이 이 날짜에 해당 리소스에 유지 관리 작업을 적용합니다. 이 날짜를 지정하면 모든 `IMMEDIATE` `optInType` 요청이 무시됩니다.

유형: 문자열

필수 항목 여부: 아니요

optInStatus

리소스에 대해 수신된 `optInType` 요청 유형을 표시합니다.

유형: 문자열

필수 항목 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS Java V2용 SDK
- AWS SDK for Ruby V3

ResourcePendingMaintenanceAction

서비스: Amazon DocumentDB Elastic Clusters

대기 중인 리소스 유지 관리 작업에 대한 정보를 제공합니다.

내용

> **Note**
>
> 필수 파라미터는 다음 표에 처음으로 설명되어 있습니다.

pendingMaintenanceActionDetails

대기 중인 리소스 유지 관리 작업에 대한 정보를 제공합니다.

유형: PendingMaintenanceActionDetails 객체 배열

필수 여부: 아니요

resourceArn

보류 중인 유지 관리 작업이 적용되는 리소스의 Amazon DocumentDB Amazon 리소스 이름(ARN)입니다.

유형: 문자열

필수 항목 여부: 아니요

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Shard

서비스: Amazon DocumentDB Elastic Clusters

샤드의 이름입니다.

내용

> **Note**
>
> 필수 파라미터는 다음 표에 처음으로 설명되어 있습니다.

createTime

샤드가 생성된 시간(협정 세계시(UTC))입니다.

유형: 문자열

필수 항목 여부: 예

shardId

샤드의 ID입니다.

유형: 문자열

필수 항목 여부: 예

status

샤드의 현재 상태.

타입: 문자열

유효 값: `CREATING | ACTIVE | DELETING | UPDATING | VPC_ENDPOINT_LIMIT_EXCEEDED | IP_ADDRESS_LIMIT_EXCEEDED | INVALID_SECURITY_GROUP_ID | INVALID_SUBNET_ID | INACCESSIBLE_ENCRYPTION_CREDS | INACCESSIBLE_SECRET_ARN | INACCESSIBLE_VPC_ENDPOINT | INCOMPATIBLE_NETWORK | MERGING | MODIFYING | SPLITTING | COPYING | STARTING | STOPPING | STOPPED | MAINTENANCE | INACCESSIBLE_ENCRYPTION_CREDENTIALS_RECOVERABLE`

필수 여부: 예

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS Java V2용 SDK
- AWS SDK for Ruby V3

ValidationExceptionField

서비스: Amazon DocumentDB Elastic Clusters

지정된 유효성 검사 예외가 발생한 특정 필드.

내용

> **Note**
>
> 필수 파라미터는 다음 목록에 먼저 설명되어 있습니다.

message

이 필드의 유효성 검사 예외를 설명하는 오류 메시지.

유형: 문자열

필수 항목 여부: 예

name

유효성 검사 예외가 발생한 필드의 이름.

유형: 문자열

필수 항목 여부: 예

참고

언어별 AWS SDKs

- AWS SDK for C++
- AWS Java V2용 SDK
- AWS SDK for Ruby V3

일반적인 오류

이 섹션에서는 모든 AWS 서비스의 API 작업에 공통적인 오류를 나열합니다. 이 서비스의 API 작업에 대한 오류는 해당 API 작업 항목을 참조하세요.

AccessDeniedException

이 작업을 수행할 수 있는 충분한 액세스 권한이 없습니다.

HTTP 상태 코드: 400

IncompleteSignature

요청 서명이 AWS 표준을 준수하지 않습니다.

HTTP 상태 코드: 400

InternalFailure

알 수 없는 오류, 예외 또는 장애 때문에 요청 처리가 실패했습니다.

HTTP 상태 코드: 500

InvalidAction

요청된 동작 또는 작업이 유효하지 않습니다. 작업을 올바로 입력했는지 확인합니다.

HTTP 상태 코드: 400

InvalidClientTokenId

제공된 X.509 인증서 또는 AWS 액세스 키 ID가 레코드에 없습니다.

HTTP 상태 코드: 403

NotAuthorized

이 작업을 수행하려면 권한이 있어야 합니다.

HTTP 상태 코드: 400

OptInRequired

AWS 액세스 키 ID에는 서비스에 대한 구독이 필요합니다.

HTTP 상태 코드: 403

RequestExpired

요청이 요청상의 날짜 스탬프로부터 15분 이상, 또는 요청 만료 날짜(예: 미리 서명된 URL)로부터 15분 이상 경과한 후 서비스에 도달했거나, 요청상의 날짜 스탬프가 15분 이상 미래입니다.

HTTP 상태 코드: 400

ServiceUnavailable

서버의 일시적 장애로 인해 요청이 실패했습니다.

HTTP 상태 코드: 503

ThrottlingException

요청 제한 때문에 요청이 거부되었습니다.

HTTP 상태 코드: 400

ValidationError

입력이 AWS 서비스에서 지정한 제약 조건을 충족하지 못합니다.

HTTP 상태 코드: 400

공통 파라미터

다음 목록에는 모든 작업이 쿼리 문자열을 사용하여 Signature Version 4 요청에 서명하는 데 사용하는 파라미터가 포함되어 있습니다. 작업별 파라미터는 그 작업에 대한 항목에 나열되어 있습니다. 서명 버전 4에 대한 자세한 내용은 IAM 사용 설명서의 AWS API 요청 서명을 참조하세요.

Action

수행할 작업입니다.

타입: 문자열

필수 항목 여부: 예

Version

요청이 작성되는 API 버전으로 YYYY-MM-DD 형식으로 표시됩니다.

타입: 문자열

필수 항목 여부: 예

X-Amz-Algorithm

요청 서명을 생성하는 데 사용된 해시 알고리즘입니다.

조건: HTTP 권한 부여 헤더 대신 쿼리 문자열에 인증 정보를 포함하는 경우 이 파라미터를 지정합니다.

타입: 문자열

유효 값: `AWS4-HMAC-SHA256`

필수 항목 여부: 조건부

X-Amz-Credential

자격 증명 범위 값이며 액세스 키, 날짜, 대상으로 하는 리전, 요청하는 서비스 및 종료 문자열("aws4_request")이 포함된 문자열입니다. 값은 다음 형식으로 표시됩니다. access_key/YYYYMMDD/region/service/aws4_request.

자세한 내용은 IAM 사용 설명서의 서명된 AWS API 요청 생성을 참조하세요.

조건: HTTP 권한 부여 헤더 대신 쿼리 문자열에 인증 정보를 포함하는 경우 이 파라미터를 지정합니다.

타입: 문자열

필수 항목 여부: 조건부

X-Amz-Date

서명을 만드는 데 사용되는 날짜입니다. 형식은 ISO 8601 기본 형식(YYYYMMDD'T'HHMMSS'Z')이어야 합니다. 예를 들어 다음 날짜 시간은 유효한 X-Amz-Date 값: `20120325T120000Z`.

조건: X-Amz-Date는 모든 요청에서 옵션이지만 서명 요청에 사용되는 날짜보다 우선할 때 사용됩니다. 날짜 헤더가 ISO 8601 기본 형식으로 지정된 경우 X-Amz-Date가 필요하지 않습니다. X-Amz-Date를 사용하는 경우 항상 Date 헤더의 값을 재정의합니다. 자세한 내용은 IAM 사용 설명서의 AWS API 요청 서명 요소를 참조하세요.

유형: 문자열

필수 항목 여부: 조건부

X-Amz-Security-Token

AWS Security Token Service ()에 대한 호출을 통해 얻은 임시 보안 토큰입니다AWS STS. AWS STS의 임시 보안 인증 정보를 지원하는 서비스 목록은 IAM 사용 설명서의 IAM으로 작업하는AWS 서비스를 참조하세요.

조건:에서 임시 보안 자격 증명을 사용하는 경우 보안 토큰을 포함해야 AWS STS합니다.

유형: 문자열

필수 항목 여부: 조건부

X-Amz-Signature

서명할 문자열과 파생된 서명 키에서 계산된 16진수로 인코딩된 서명을 지정합니다.

조건: HTTP 권한 부여 헤더 대신 쿼리 문자열에 인증 정보를 포함하는 경우 이 파라미터를 지정합니다.

타입: 문자열

필수 항목 여부: 조건부

X-Amz-SignedHeaders

표준 요청의 일부로 포함된 모든 HTTP 헤더를 지정합니다. 서명된 헤더 지정에 대한 자세한 내용은 IAM 사용 설명서의 서명된 AWS API 요청 생성을 참조하세요.

조건: HTTP 권한 부여 헤더 대신 쿼리 문자열에 인증 정보를 포함하는 경우 이 파라미터를 지정합니다.

타입: 문자열

필수 항목 여부: 조건부

릴리스 정보

이 릴리스 노트에서는 릴리스 날짜별로 Amazon DocumentDB의 특성, 개선 및 버그 수정에 대해 설명합니다. 릴리스 노트에는 모든 Amazon DocumentDB 엔진 버전에 대한 업데이트가 발생하는 대로 포함됩니다.

현재 Amazon DocumentDB 엔진 패치 버전을 확인하려면 다음 명령을 실행합니다.

```
db.runCommand({getEngineVersion: 1})
```

클러스터가 최신 버전의 엔진을 사용하지 않는 경우, 엔진을 업그레이드하기 위해 보류 중인 유지 관리가 있을 수 있습니다. 자세한 내용은 개발자 가이드의 Amazon DocumentDB 유지 관리 섹션을 참조하세요.

데이터베이스 관련 새로운 소식 페이지에서 새로운 Amazon DocumentDB 기능을 필터링할 수 있습니다. 제품에서 Amazon DocumentDB를 선택합니다. 그런 다음 **elastic clusters** 또는 **vector search**와 같은 키워드를 사용하여 검색합니다.

주제

- 2025년 6월 18일
- 2025년 5월 8일
- 2025년 4월 2일
- 2025년 3월 24일
- 2025년 2월 6일
- 2025년 1월 28일
- 2025년 1월 15일
- 2024년 12월 18일
- 2024년 11월 12일
- 2024년 11월 6일
- 2024년 11월 1일
- 2024년 10월 22일
- 2024년 9월 18일
- 2024년 9월 17일

- 2024년 8월 22일
- 2024년 8월 20일
- 2024년 8월 8일
- 2024년 7월 23일
- 2024년 7월 22일
- 2024년 7월 9일
- 2024년 7월 8일
- 2024년 6월 25일
- 2024년 5월 29일
- 2024년 4월 3일
- 2024년 2월 22일
- 2024년 1월 30일
- 2024년 1월 10일
- 2023년 12월 20일
- 2023년 12월 13일
- 2023년 11월 29일
- 2023년 11월 21일
- 2023년 11월 17일
- 2023년 11월 6일
- 2023년 9월 25일
- 2023년 9월 20일
- 2023년 9월 15일
- 2023년 9월 11일
- 2023년 8월 3일
- 2023년 7월 13일
- 2023년 6월 7일
- 2023년 5월 10일
- 2023년 4월 4일

- 2023년 3월 22일
- 2023년 3월 1일
- 2023년 2월 27일
- 2023년 2월 2일
- 2022년 11월 30일
- 2022년 8월 9일
- 2022년 7월 25일
- 2022년 6월 27일
- 2022년 4월 29일
- 2022년 4월 7일
- 2022년 3월 16일
- 2022년 2월 8일
- 2022년 1월 24일
- 2022년 1월 21일
- 2021년 10월 25일
- 2021년 6월 24일
- 2021년 5월 4일
- 2021년 1월 15일
- 2020년 11월 9일
- 2020년 10월 30일
- 2020년 9월 22일
- 2020년 7월 10일
- 2020년 6월 30일

2025년 6월 18일

새로운 특성

이제 이스라엘(텔아비브) 리전에서 Amazon DocumentDB를 사용할 수 있습니다.

2025년 5월 8일

새로운 특성

이제 유럽(스톡홀름) 리전에서 Amazon DocumentDB를 사용할 수 있습니다. 자세한 내용은 이 블로그 게시물을 참조하십시오.

2025년 4월 2일

> **Note**
>
> 다음 Amazon DocumentDB 엔진 패치는 향후 몇 주 내에 모든 Amazon DocumentDB 리전에 전달되는 과정에 있습니다. 해당 리전에서이 엔진 패치를 사용할 수 있게 되면의 AWS 상태 대시보드(AHD)를 통해, AWS Management Console 그리고 AWS 계정의 루트 사용자 이메일 주소로 이메일을 통해 서비스 패치 알림을 받게 됩니다.
> 이 엔진 패치에는 다음과 같은 새로운 기능과 버그 수정이 포함되어 있습니다. 엔진 패치가 모든 리전에서 사용할 수 있게 되면 아래 목록은 관련 지원 문서와 함께 추가 기능 공지를 포함하도록 업데이트될 수 있습니다.

새로운 특성

Amazon DocumentDB 5.0(엔진 패치 버전 3.0.14228)

`collStats` 진단 명령 내에 미사용 스토리지 크기 및 백분율에 대한 지원이 추가되었습니다.

Amazon DocumentDB 4.0(엔진 패치 버전 2.0.11357)

`collStats` 진단 명령 내에 미사용 스토리지 크기 및 백분율에 대한 지원이 추가되었습니다.

버그 수정 및 기타 변경

배경 인덱스가 있는 벡터 인덱스 생성의 버그를 수정했습니다.

2025년 3월 24일

새로운 특성

Amazon DocumentDB 5.0(엔진 패치 버전 3.0.11800)

변경 스트림postBatchResumeToken에서에 대한 지원이 추가되었습니다.

자세한 내용은 postBatchResumeToken에서 변경 스트림 재개 단원을 참조하십시오.

2025년 2월 6일

새로운 특성

이제 Amazon DocumentDB가와 통합되었습니다 AWS Toolkit for Visual Studio Code. 자세한 내용은 이 새로운 기능 블로그 게시물을 참조하고 사용 AWS Toolkit for Visual Studio Code 설명서의 도구 키트에서 Amazon DocumentDB 작업을 참조하세요.

2025년 1월 28일

새로운 특성

원클릭 연결을 위해 이제 Amazon DocumentDB가 인스턴스 기반 및 탄력적 클러스터 AWS CloudShell 와 통합됩니다.

자세한 내용은 다음 중 일부 또는 전부를 참조하십시오.

- Amazon DocumentDB로 시작하기
- Amazon DocumentDB 엘라스틱 클러스터 시작하기
- 새로운 블로그 게시물
- 기술 방법 블로그 게시물

2025년 1월 15일

새로운 특성

Amazon DocumentDB 5.0(엔진 패치 버전 3.0.11800)

Amazon DocumentDB는 이제 라이터 인스턴스 재시작을 통해 읽기 가용성을 유지합니다. 이제 리더 인스턴스는 라이터 인스턴스를 다시 시작하는 동안 읽기 요청을 계속 처리합니다.

버그 수정 및 기타 변경

Amazon DocumentDB 5.0(엔진 패치 버전 3.0.11800)

- 대량 삽입의 특수 사례를 처리하는 `killOp` 명령을 수정했습니다.
- Amazon DocumentDB 인스턴스에서 네트워크 I/O 관련 메모리 사용량을 개선했습니다.
- `$text` 필터 쿼리에 대한 `count` 명령을 수정했습니다.

Amazon DocumentDB 4.0(엔진 패치 버전 2.0.11096)

- 대량 삽입의 특수 사례를 처리하는 `killOp` 명령을 수정했습니다.
- Amazon DocumentDB 인스턴스에서 네트워크 I/O 관련 메모리 사용량을 개선했습니다.

2024년 12월 18일

새로운 특성

Amazon DocumentDB 5.0(엔진 패치 버전 3.0.5721)

Amazon DocumentDB 인스턴스 기반 클러스터는 이제 NVMe 지원 R6GD 인스턴스를 지원합니다.

자세한 내용은 NVMe 지원 인스턴스 및 새로운 블로그 게시물을 참조하세요.

2024년 11월 12일

새로운 특성

Amazon DocumentDB 탄력적 클러스터는 이제 백그라운드 인덱스를 지원합니다.

2024년 11월 6일

새로운 특성

Amazon DocumentDB 5.0(엔진 패치 버전 3.0.11051)

- 최소 TLS 버전 `tls1.2+` 및 `tls` 클러스터 파라미터`tls1.3+`에 대한 지원이 추가되었습니다.
- 필드 이름에서 문자 달러($) 및 점(.)에 대한 지원이 활성화되었습니다. 기능적 차이점은 단원을 참조하십시오필드 이름의 달러($) 및 점(.).

Amazon DocumentDB 4.0(엔진 패치 버전 2.0.10980)

- 최소 TLS 버전 `tls1.2+` 및 `tls` 클러스터 파라미터`tls1.3+`에 대한 지원이 추가되었습니다.

2024년 11월 1일

새로운 특성

Amazon DocumentDB는 이제 탄력적 클러스터 유지 관리 작업을 지원합니다. 자세한 내용은 Amazon DocumentDB 탄력적 클러스터 유지 관리 단원을 참조하십시오.

2024년 10월 22일

새로운 특성

Amazon DocumentDB는 이제 스토리지 네트워크 처리량 지표 `StorageNetworkReceiveThroughput`, 및 `StorageNetworkTransmitThroughput`를 지원합니다`StorageNetworkThroughput`. 자세한 내용은 CloudWatch 지표를 사용한 Amazon DocumentDB 인스턴스 사용량 평가 단원을 참조하십시오.

2024년 9월 18일

새로운 특성

이제 아프리카(케이프타운) 리전에서 Amazon DocumentDB를 사용할 수 있습니다. 자세한 내용은 이 블로그 게시물을 참조하십시오.

이제 유럽(스페인) 지역에서 Amazon DocumentDB를 사용할 수 있습니다. 자세한 내용은 이 블로그 게시물을 참조하십시오.

2024년 9월 17일

새로운 특성

Amazon DocumentDB 5.0(엔진 패치 버전 3.0.10696)

이제 Amazon DocumentDB는 `runCommand`에서 `reIndex`를 사용하여 인덱스 재구축을 지원합니다. 자세한 내용은 를 사용한 인덱스 유지 관리 `reIndex` 단원을 참조하십시오.

> **Note**
>
> `reIndex`는 Amazon DocumentDB 5.0(엔진 패치 버전 3.0.10696 이상)에서만 지원됩니다.

버그 수정 및 기타 변경

Amazon DocumentDB 5.0(엔진 패치 버전 3.0.10696) 및 Amazon DocumentDB 4.0(엔진 패치 버전 2.0.10898)

- 이제 `$setOnInsert`는 업데이트 작업으로 인한 삽입 중에 `_id` 필드를 지원합니다.
- Amazon DocumentDB 5.0으로 메이저 버전을 업그레이드한 후 스토리지 공간을 회수하지 못하는 문제를 수정했습니다.

2024년 8월 22일

새로운 특성

Amazon DocumentDB 5.0(모든 엔진 패치 버전) 및 Amazon DocumentDB 4.0(엔진 패치 버전 2.0.5704)

Amazon DocumentDB 글로벌 클러스터는 이제 클러스터 전환 및 관리형 클러스터 장애 조치를 지원합니다. 자세한 내용은 Amazon DocumentDB 글로벌 클러스터에 대한 전환 수행 및 Amazon DocumentDB 글로벌 클러스터에 대한 관리형 장애 조치 수행 섹션을 참조하세요.

> **Note**
>
> 글로벌 클러스터 전환 및 장애 조치는 Amazon DocumentDB 4.0 및 5.0에서만 지원됩니다.

2024년 8월 20일

새로운 특성

Amazon DocumentDB 3.6(최소 엔진 패치 버전 1.0.208662)의 경우 TLS CA 인증서 업데이트 시 더 이상 시스템을 재부팅할 필요가 없습니다. 자세한 내용은 Amazon DocumentDB TLS 인증서 업데이트 단원을 참조하십시오.

2024년 8월 8일

새로운 특성

Amazon DocumentDB 탄력적 클러스터는 이제 아시아 태평양(홍콩), 캐나다(중부) 및 유럽(파리) 리전에서 사용할 수 있습니다. 자세한 내용은 탄력적 클러스터의 리전 가용성 단원을 참조하십시오.

2024년 7월 23일

새로운 특성

Amazon DocumentDB 5.0(엔진 패치 버전 3.0.8126) 및 Amazon DocumentDB 4.0(엔진 패치 버전 2.0.10709)

- 더 긴 인덱스 이름(최대 255자)에 대한 지원이 추가되었습니다. 자세한 내용은 명명 제약 조건 섹션을 참조하세요.
- 최대 연결 한도를 두 배로 늘렸습니다.

인스턴스 유형	원래 한도	신규 한도
t3.medium	500	1000
t4g.medium	500	1000
r5.large	1700	3400
r5.xlarge	3500	7000
r5.2xlarge	7100	14200
r5.4xlarge	14200	28400
r5.8xlarge	28400	60000
r5.12xlarge	30000	60000
r5.16xlarge	30000	60000
r5.24xlarge	30000	60000

인스턴스 유형	원래 한도	신규 한도
r6g.large	1700	3400
r6g.xlarge	3500	7000
r6g.2xlarge	7100	14200
r6g.4xlarge	14200	28400
r6g.8xlarge	28400	60000
r6g.12xlarge	30000	60000
r6g.16xlarge	30000	60000

자세한 내용은 인스턴스 제한 단원을 참조하십시오.

버그 수정 및 기타 변경

Amazon DocumentDB 5.0(엔진 패치 버전 3.0.8126)

리더의 변경 스트림에 대해 `CurrentTime` 및 `ResumeToken`을 동기화하도록 로직을 개선했습니다.

2024년 7월 22일

새로운 특성

Amazon DocumentDB 5.0(엔진 패치 버전 3.0.6742)

- DML 감사 필터링 지원이 추가되었습니다. 이제 모든 DML 쿼리를 로깅하는 대신 특정 요구 사항에 따라 DML 감사 로그를 필터링하도록 필터 조건을 설정할 수 있습니다. 자세한 내용은 DML 감사 이벤트 필터링 단원을 참조하십시오.
- 다음에 대한 문서 압축 지원이 추가되었습니다.
  - 최소 압축 임계값 설정
  - 기존 컬렉션에 대한 압축 활성화(새 문서에 적용 가능)
  - 클러스터 수준에서 기본 압축 설정 허용

자세한 내용은 컬렉션 수준 문서 압축 관리 단원을 참조하십시오.

- 리더 인스턴스에서 변경 스트림 소비에 대한 지원이 추가되었습니다. 자세한 내용은 보조 인스턴스에서 변경 스트림 사용 단원을 참조하십시오.

Amazon DocumentDB 4.0(엔진 패치 버전 2.0.10593)

- DML 감사 필터링 지원이 추가되었습니다. 이제 모든 DML 쿼리를 로깅하는 대신 특정 요구 사항에 따라 DML 감사 로그를 필터링하도록 필터 조건을 설정할 수 있습니다. 자세한 내용은 DML 감사 이벤트 필터링 단원을 참조하십시오.

Amazon DocumentDB 3.6(엔진 패치 버전 1.0.208662)

Amazon DocumentDB MVU의 db.r5.* 및 db.r6.* 인스턴스에 대한 인덱스 제한을 제거했습니다. 자세한 내용은 MVU 사전 조건 및 제한 사항 단원을 참조하십시오.

버그 수정 및 기타 변경

Amazon DocumentDB 3.6(엔진 패치 버전 1.0.208662)

Amazon DocumentDB는 이제 -NaN을 유효한 JSON 토큰으로 인식합니다.

2024년 7월 9일

새로운 특성

Amazon DocumentDB 4.0(최소 엔진 패치 버전 2.0.10179) 및 5.0(최소 엔진 패치 버전 3.0.4780)의 경우 TLS CA 인증서 업데이트에는 더 이상 시스템을 재부팅할 필요가 없습니다. 자세한 내용은 Amazon DocumentDB TLS 인증서 업데이트 단원을 참조하십시오.

2024년 7월 8일

새로운 특성

이제 유럽(밀라노) 지역에서 Amazon DocumentDB 탄력적 클러스터를 사용할 수 있습니다. 자세한 내용은 탄력적 클러스터의 리전 가용성 단원을 참조하십시오.

2024년 6월 25일

새로운 특성

AWS IAM ARNs 사용한 인증은 지원되는 모든 리전의 Amazon DocumentDB 인스턴스 기반 5.0 클러스터에서 사용할 수 있습니다. 자세한 내용은 IAM 자격 증명을 사용한 인증 단원을 참조하십시오.

2024년 5월 29일

새로운 특성

Amazon DocumentDB 5.0(엔진 패치 버전 3.0.6742)

- `$regexMatch` 및 `$regexFind` 연산자에 대한 지원이 추가되었습니다.
- 큰 정수를 처리할 때 감사 로그의 전체 정밀도를 보장하기 위한 지원이 추가되었습니다. 이제 감사 로그는 모든 숫자에 대해 정확한 숫자 표현을 유지하여 정밀도 손실을 방지합니다.

Amazon DocumentDB 4.0(엔진 패치 버전 2.0.10593)

- 큰 정수를 처리할 때 감사 로그의 전체 정밀도를 보장하기 위한 지원이 추가되었습니다. 이제 감사 로그는 모든 숫자에 대해 정확한 숫자 표현을 유지하여 정밀도 손실을 방지합니다.

2024년 4월 3일

중동(UAE) 리전에서 이제 Amazon DocumentDB를 사용할 수 있습니다. 자세한 내용은 이 블로그 게시물을 참조하십시오.

새로운 특성

Amazon DocumentDB 5.0(엔진 패치 버전 3.0.5721)

- `bypassDocumentValidation`에 대한 지원 및 `$jsonSchema`에 대한 세분화된 오류 메시지가 추가되었습니다. `bypassDocumentValidation`에 대한 자세한 정보는 bypassDocumentValidation 섹션을 참조하세요.
- `$expr`의 지원이 추가되었습니다.
- `$lookup`에 상관 없는 조인에 대한 지원이 추가되었습니다.

- `$out` 집계 단계에서 검증 규칙을 유지하기 위한 지원이 추가되었습니다.

Amazon DocumentDB 4.0(엔진 패치 버전 2.0.10392)

- `$jsonSchema`에 대한 `bypassDocumentValidation` 지원을 추가했습니다. `bypassDocumentValidation`에 대한 자세한 정보는 bypassDocumentValidation 섹션을 참조하세요.
- `$expr`의 지원이 추가되었습니다.
- `$lookup`에 상관 없는 조인에 대한 지원이 추가되었습니다.
- `$out` 집계 단계에서 검증 규칙을 유지하기 위한 지원이 추가되었습니다.

버그 수정 및 기타 변경

- mongo 쉘 버전 1.7 이상에서 `db.coll.stats()`를 호출할 때 발생하는 오류를 수정했습니다.
- 동일한 집계 파이프라인의 일부로 `$regex`가 포함된 변경 스트림 쿼리에 대한 메모리 누출 문제를 수정했습니다.

2024년 2월 22일

새로운 특성

Amazon DocumentDB 탄력적 클러스터

Amazon DocumentDB 탄력적 클러스터는 이제 다음 기능을 지원합니다.

- 읽기 가능한 보조 샤드 인스턴스 복제본 - 자세한 내용은 1단계: 엘라스틱 클러스터 생성의 5b단계를 참조하세요.
- 클러스터 시작/중지 - 자세한 내용은 Amazon DocumentDB 탄력적 클러스터 중지 및 시작 섹션을 참조하세요.
- 구성 가능한 샤드 인스턴스 - 자세한 내용은 1단계: 엘라스틱 클러스터 생성의 5b단계를 참조하세요.
- 스냅샷 자동 백업 - 자세한 내용은 탄력적 클러스터 스냅샷 자동 백업 관리 섹션을 참조하세요.
- 스냅샷 복사 - 자세한 내용은 탄력적 클러스터 스냅샷 보기 섹션을 참조하세요.

2024년 1월 30일

새로운 특성

Amazon DocumentDB 탄력적 클러스터

이제 Amazon DocumentDB 탄력적 클러스터를 다음 리전에서 사용할 수 있습니다.

- 아시아 태평양(뭄바이)
- 아시아 태평양(서울)
- 남아메리카(상파울루)
- 유럽(런던)

자세한 내용은 탄력적 클러스터 리전 및 버전 가용성 단원을 참조하십시오.

Amazon DocumentDB 글로벌 클러스터

이제 AWS GovCloud(미국 동부) 및 GovCloud(미국 AWS GovCloud 서부) AWS GovCloud (US) 리전 모두에서 글로벌 클러스터를 사용할 수 있습니다.

2024년 1월 10일

새로운 특성

Amazon DocumentDB 5.0(엔진 패치 버전 3.0.4574, 3.0.4780, 3.0.4960)

- HNSW 벡터 인덱스에 대한 지원이 추가되었습니다. 자세한 내용은 Amazon DocumentDB에 대한 벡터 검색 단원을 참조하십시오.
- 부분 인덱스에 대한 지원이 추가되었습니다. 자세한 내용은 부분 인덱스 단원을 참조하십시오.
- `currentOp` 명령 내 컬렉션에서 GC 런타임에 대한 지원이 추가되었습니다.
- Amazon DocumentDB의 기본 텍스트 검색에 대한 텍스트 인덱스 지원이 추가되었습니다. 자세한 내용은 Amazon DocumentDB를 사용하여 텍스트 검색 수행 단원을 참조하십시오.
- `$jsonSchema` 스키마 키워드 `type`, `allOf`, `oneOf`, `anyOf`, `not`, `maxItems`, `minItems`, `maxProperties`, `minProperties`, `pattern`, `patternProperties`, `multipleOf`, `dependencies` 및 `uniqueItems`에 대한 지원이 추가되었습니다.

  자세한 내용은 JSON 스키마 검증 사용을 참조하세요.

- 산술 연산자 `$ceil`, `$floor`, `$ln`, `$log`, `$log10`, `$sqrt` 및 `$exp`에 대한 지원이 추가되었습니다.

  자세한 내용은 산술 연산자을 참조하세요.
- 조건식 연산자 `$switch`에 대한 지원이 추가되었습니다.
- 병렬 `IVFFLAT` 벡터 인덱스 빌드에 대한 지원이 추가되었습니다. 개발자 안내서에서 병렬 `IVFFLAT` 벡터 인덱스 빌드 제한을 제거하여 설명서가 업데이트되었습니다.

Amazon DocumentDB 4.0(엔진 패치 버전 2.0.10124, 2.0.10179, 2.0.10221)

- `currentOp` 명령 내 컬렉션에서 GC 런타임에 대한 지원이 추가되었습니다.
- `$jsonSchema` 스키마 키워드 `type`, `allOf`, `oneOf`, `anyOf`, `not`, `maxItems`, `minItems`, `maxProperties`, `minProperties`, `pattern`, `patternProperties`, `multipleOf`, `dependencies` 및 `uniqueItems`에 대한 지원이 추가되었습니다.

  자세한 내용은 JSON 스키마 검증 사용을 참조하세요.
- 산술 연산자 `$ceil`, `$floor`, `$ln`, `$log`, `$log10`, `$sqrt` 및 `$exp`에 대한 지원이 추가되었습니다.

  자세한 내용은 산술 연산자을 참조하세요.
- 조건식 연산자 `$switch`에 대한 지원이 추가되었습니다.

버그 수정 및 기타 변경

- `db.runCommand("dbstats")` 호출 시 대/소문자를 구분하지 않는 기능이 추가되었습니다. 3.0.4960 또는 2.0.10221 이전 엔진 패치 버전의 Amazon DocumentDB 5.0 및 4.0 고객은 이러한 최신 엔진 패치를 적용해야 합니다.
- mongo 쉘 버전 1.7 이상에서 `db.coll.stats()`를 호출할 때 발생하는 오류를 수정했습니다. 개발자 안내서에서 mongo 쉘 `db.coll.stats()` 문제 해결 팁을 제거하여 설명서가 업데이트되었습니다.

2023년 12월 20일

기타 변경사항

Amazon DocumentDB 3.6 및 4.0에서 인플레이스 주요 버전 업그레이드에 대한 지원이 활성화되었습니다. 자세한 내용은 Amazon DocumentDB 인플레이스 주요 버전 업그레이드 단원을 참조하십시오.

2023년 12월 13일

새로운 특성

원클릭 EC2 연결에 대한 지원이 추가되었습니다. 자세한 내용은 Amazon EC2를 사용하여 연결 단원을 참조하십시오.

2023년 11월 29일

Amazon DocumentDB 5.0(엔진 패치 버전 3.0.3727)

새로운 특성

벡터 검색에 대한 지원이 추가되었습니다. 자세한 내용은 이 블로그 게시물과 Amazon DocumentDB 개발자 안내서의 Amazon DocumentDB에 대한 벡터 검색를 참고하시기 바랍니다.

2023년 11월 21일

Amazon DocumentDB 5.0(엔진 패치 버전 3.0.3727)

새로운 특성

I/O 최적화 스토리지에 대한 지원이 추가되었습니다. 자세한 내용은 Amazon DocumentDB 개발자 안내서의 Amazon DocumentDB 클러스터 스토리지 구성를 참조하세요.

SageMaker Canvas와 코드 없는 기계 학습을 위한 통합이 추가되었습니다. 자세한 내용은 Amazon DocumentDB 개발자 안내서의 Amazon SageMaker AI Canvas를 사용한 노코드 기계 학습를 참조하세요.

2023년 11월 17일

새로운 특성

이제 AWS GovCloud(미국 동부) 리전에서 Amazon DocumentDB를 사용할 수 있습니다. 자세한 내용은 이 블로그 게시물을 참조하십시오.

버그 수정 및 기타 변경

Amazon DocumentDB 3.6(엔진 패치 버전 1.0.208570)

사용자 정의 로컬 변수 이름은 이제 `$let` 및 `$filter`와 같은 프로젝션 연산자에 대해 "_"(밑줄)을 지원합니다.

2023년 11월 6일

Amazon DocumentDB 5.0(엔진 패치 버전 3.0.3727) 및 4.0(엔진 패치 버전 2.0.9876)

새로운 특성

- `$jsonSchema`스키마 키워드`maxLength`, `minLength`, `maximum`, `minimum`, `exclusiveMaximum`, `exclusiveMinimum`, `items`, 및 `additionalItems`에 대한 지원이 추가되었습니다.

  JSON 스키마 검증은 인스턴스 기반 클러스터에서만 지원된다는 점에 유의하세요.
- `$convert`집계 파이프라인 연산자 및 속기 파생 연산자 `$toBool`, `$toInt`, `$toLong`, `$toDouble`, `$toString`, `$toDecimal`, `$toObjectId` 및 `$toDate`에 대한 지원이 추가되었습니다.
- 집합 표현식 연산자`$setDifference`, `$anyElementTrue` 및 `$allElementTrue`에 대한 지원이 추가되었습니다.

버그 수정 및 기타 변경

`-NaN`에서 `NaN`로 변경 스트림 업데이트가 표시되지 않던 문제가 수정되었습니다.

2023년 9월 25일

새로운 특성

이제 아시아 태평양(홍콩) 리전에서 Amazon DocumentDB를 사용할 수 있습니다. 자세한 내용은 이 블로그 게시물을 참조하십시오.

2023년 9월 20일

새로운 기능

Amazon DocumentDB 3.6 및 4.0에서 인플레이스 주요 버전 업그레이드에 대한 지원이 추가되었습니다. 자세한 내용은 Amazon DocumentDB 인플레이스 주요 버전 업그레이드을 참조하세요.

2023년 9월 15일

새로운 특성

Amazon DocumentDB 5.0(엔진 패치 버전 3.0.3140) 및 4.0(엔진 패치 버전 2.0.9686)

인스턴스 기반 클러스터에서만 $jsonSchema 스키마 검사기에 대한 지원이 추가되었습니다. 자세한 내용은 JSON 스키마 검증 사용을 참조하세요.

2023년 9월 11일

새로운 특성

이제 아시아 태평양(하이데라바드) 리전에서 Amazon DocumentDB를 사용할 수 있습니다. 자세한 내용은 이 블로그 게시물을 참조하십시오.

2023년 8월 3일

새로운 기능

Amazon DocumentDB Elastic Clusters

- Amazon DocumentDB Elastic 클러스터는 이제 다음 작업을 지원합니다.
  - `top`
  - `collStats`
  - `hint`
  - `dataSize`

  지원되는 명령 및 작업의 전체 목록은 Amazon DocumentDB에서 지원되는 MongoDB API, 작업 및 데이터 형식을 참조하세요.
- 이제 TTL(Time to Live) 인덱스가 지원됩니다.
- 이제 인덱스 표현식에서도 인덱스 `hints`이 지원됩니다.

2023년 7월 13일

새로운 특성

Amazon DocumentDB 5.0(엔진 패치 버전 3.0.1948)

- 문서 압축에 대한 지원이 추가되었습니다.
- 병렬 인덱스 빌드에 대한 지원이 추가되었습니다.
- 인덱스 빌드 상태에 대한 지원이 추가되었습니다.

Amazon DocumentDB 4.0(엔진 패치 버전 2.0.9259)

- 병렬 인덱스 빌드에 대한 지원이 추가되었습니다.

버그 수정 및 기타 변경

Amazon DocumentDB 5.0(엔진 패치 버전 3.0.1948)

- 사용자가 시스템 컬렉션에 액세스할 수 없을 때 Amazon DocumentDB Elastic 클러스터의 `createCollection`에서 발생하는 인증 문제를 수정했습니다.
- 보조 지역 인스턴스가 동일한 기본 지역 인스턴스 이름을 사용할 수 없던 문제가 해결되었습니다.

Amazon DocumentDB 4.0(엔진 패치 버전 2.0.9259)

- 감사 로그에 내부 모니터링 쿼리를 추가하는 것을 중단했습니다.

2023년 6월 7일

버그 수정 및 기타 변경

Amazon DocumentDB 5.0

- 이제 Amazon DocumentDB 5.0에서 r5 및 t3.medium 인스턴스가 지원됩니다.
- `engineVersion` 옵션 기본값은 AWS SDK AWS CLI, 및 `5.0.0` 입니다 AWS CloudFormation.

2023년 5월 10일

버그 수정 및 기타 변경

Amazon DocumentDB 5.0(엔진 패치 버전 3.0.1361)

- createIndex 명령에서 `ignoreunknownindexoptions`에 대한 지원을 추가했습니다.
- 감사 로그에 내부 모니터링 쿼리를 추가하는 것을 중단했습니다.
- 사용자 정의 로컬 변수 이름은 이제 `$let` 및 `$filter`와 같은 프로젝션 연산자에 대해 "_"(밑줄)을 지원합니다.

2023년 4월 4일

버그 수정 및 기타 변경

Amazon DocumentDB 4.0(엔진 패치 버전 2.0.8934)

- 워크로드가 진행 중인 동안 DML 감사가 활성화된 경우 발생하는 문제가 해결되었습니다.
- 힌트가 포함된 집계 명령에 문자열 값이 전달될 때 발생하는 DML 감사 문제가 해결되었습니다.
- readwriteanydatabase 역할을 사용하는 사용자가 authorizedCollections와 nameOnly 옵션을 모두 true로 설정한 경우 `listCollections` 명령이 작동하지 않는 문제가 해결되었습니다.
- 필드 이름의 숫자 문자열을 제대로 구문 분석하기 위한 문제가 해결되었습니다.
- 장시간 실행되는 커서가 가비지 수집에 영향을 줄 경우, 해당 커서를 취소하세요.
- 사용자 정의 로컬 변수 이름은 이제 `$let` 및 `$filter`와 같은 프로젝션 연산자에 대해 "_"(밑줄)을 지원합니다.

2023년 3월 22일

새로운 특성

이제 Amazon DocumentDB Elastic 클러스터를 이제 아시아 태평양(싱가포르), 아시아 태평양(시드니) 및 아시아 태평양(도쿄) 지역에서 사용할 수 있습니다. 자세한 내용은 탄력적 클러스터 리전 및 버전 가용성 단원을 참조하십시오.

2023년 3월 1일

새로운 특성

Amazon DocumentDB 5.0(엔진 패치 버전 3.0.775)

- Amazon DocumentDB 5.0 출시
  - 몽고DB 5.0 호환성(MongoDB 5.0 API 드라이버 지원)
  - 클라이언트측 필드 레벨 암호화 FLE) 지원 이제 Amazon DocumentDB 클러스터에 데이터를 쓰기 전에 클라이언트 측에서 필드를 암호화할 수 있습니다. 자세한 내용은 클라이언트 측 필드 수준 암호화를 참조하세요.
  - 새 집계 연산자: `$dateAdd`, `$dateSubtract`
- 모든 인스턴스 기반 Amazon DocumentDB 클러스터와 샤드 기반 Elastic 클러스터의 스토리지 한도를 128TiB로 늘렸습니다.
- Amazon DocumentDB 5.0은 이제 첫 번째 중첩 수준에서 `$elemMatch`운영자를 사용한 인덱스 스캔을 지원합니다. 쿼리 전용 필터에 한 레벨의 `$elemMatch` 필터가 있는 경우 인덱스 스캔이 지원되고 중첩된 `$elemMatch` 쿼리는 인덱스 스캔을 지원하지 않습니다.

  인덱스 스캔을 지원하는 쿼리 셰이프:

  ```
  db.foo.find( { "a": {$elemMatch: { "b": "xyz", "c": "abc"} } })
  ```

  인덱스 스캔을 지원하지 않는 쿼리 셰이프:

  ```
  db.foo.find( { "a": {$elemMatch: { "b": {$elemMatch: { "d": "xyz", "e":
   "abc"} }} } })
  ```

2023년 2월 27일

버그 수정 및 기타 변경

Amazon DocumentDB 4.0

에 대한 지원이 추가되었습니다 AWS Lambda. 자세한 내용은 변경 스트림 AWS Lambda 과 함께 사용을 참조하세요.

2023년 2월 2일

버그 수정 및 기타 변경

Amazon DocumentDB 3.6(엔진 패치 버전 1.0.208432)

- 워크로드가 진행 중인 동안 DML 감사가 활성화된 경우 발생하는 문제가 해결되었습니다.
- 힌트가 포함된 집계 명령에 문자열 값이 전달될 때 발생하는 DML 감사 문제가 해결되었습니다.
- readwriteanydatabase 역할을 사용하는 사용자가 authorizedCollections와 nameOnly 옵션을 모두 true로 설정한 경우 `listCollections` 명령이 작동하지 않는 문제가 해결되었습니다.
- 필드 이름의 숫자 문자열을 제대로 구문 분석하기 위한 문제가 해결되었습니다.
- 장시간 실행되는 커서가 가비지 수집에 영향을 줄 경우, 해당 커서를 취소하세요.

2022년 11월 30일

새로운 특성

Amazon DocumentDB Elastic Clusters

Amazon DocumentDB Elastic 클러스터는 사용자가 MongoDB 샤딩 API를 활용하여 클러스터를 스케일 아웃할 수 있는 새로운 유형의 Amazon DocumentDB 클러스터입니다. 엘라스틱 클러스터는 데이터와 컴퓨팅을 여러 기본 컴퓨팅 인스턴스 및 볼륨에 분산하여 페타바이트의 스토리지 용량으로 거의 모든 수의 읽기 및 쓰기를 처리합니다. 자세한 내용은 Amazon DocumentDB 엘라스틱 클러스터 사용법을 참조하십시오.

2022년 8월 9일

새로운 특성

Amazon DocumentDB 3.6(엔진 패치 버전 1.0.208152) 및 4.0

- Decimal128 데이터 유형에 대한 지원이 추가되었습니다. Decimal128은 DocumentDB를 사용할 수 있는 모든 지역에서 지원되는 BSON 데이터 유형입니다.

  자세한 정보는 데이터 형식을 참조하십시오.

- Amazon CloudWatch Logs에 DML 쿼리 감사에 대한 지원이 추가되었습니다. 이제 Amazon DocumentDB는 데이터 조작 언어(DML) 이벤트 및 데이터 정의 언어(DDL) 이벤트를 Amazon CloudWatch Logs에 기록할 수 있습니다.

  자세한 내용은 이 블로그 게시물을 참조하십시오.

버그 수정 및 기타 변경

Amazon DocumentDB 3.6(엔진 패치 버전 1.0.208152) 및 4.0

- 이제 `changeOwnPassword` 권한이 있는 자체 비밀번호로 비밀번호를 변경할 수 있습니다.

2022년 7월 25일

새로운 특성

Amazon DocumentDB 4.0

이제 동일한 DocumentDB 클러스터 볼륨을 사용하고 원본 클러스터와 동일한 데이터를 갖는 클론을 생성할 수 있어 클러스터를 더 빠르게 생성할 수 있습니다. 자세한 내용은 Amazon DocumentDB 클러스터 관리를 참조하십시오.

2022년 6월 27일

새로운 특성

Amazon DocumentDB 4.0(엔진 패치 버전 2.0.7509)

Amazon DocumentDB는 사용 패턴에 따라 데이터베이스 크기를 동적으로 조정합니다. 데이터를 더 추가하면 스페이스가 최대 64 테비바이트(TiB)까지 늘어나고 데이터를 삭제하면 할당된 공간이 줄어듭니다.

2022년 4월 29일

새로운 특성

Amazon DocumentDB는 이제 중국(베이징) 지역에서 사용할 수 있습니다. 자세한 내용은 이 블로그 게시물을 참조하십시오.

2022년 4월 7일

새로운 특성

Amazon DocumentDB 3.6(엔진 패치 버전 1.0.207836 및 1.0.208015) 및 4.0(엔진 패치 버전 2.0.6142 및 2.0.6948)

Amazon DocumentDB Performance Insights는 현재 미리 보기 단계에 있습니다. 이제 추가 비용 없이 7일간의 성능 기록을 롤링 윈도우에 저장할 수 있습니다. 자세한 내용은 성능 인사이트를 사용한 모니터링을 참조하세요.

2022년 3월 16일

새로운 특성

이제 유럽(밀라노) 지역에서 Amazon DocumentDB를 사용할 수 있습니다. 자세한 내용은 이 블로그 게시물을 참조하십시오.

2022년 2월 8일

새로운 특성

Amazon DocumentDB R6g 및 T4g 인스턴스는 이제 아시아 태평양, 남미 및 유럽에서 사용할 수 있습니다. 자세한 내용은 이 블로그 게시물을 참조하십시오.

2022년 1월 24일

새로운 특성

Amazon DocumentDB 3.6(엔진 패치 버전 1.0.207684) 및 4.0(엔진 패치 버전 2.0.5170)

- Amazon DocumentDB는 이제 무료 평가판을 제공합니다. 자세한 내용은 Amazon DocumentDB 무료 평가판 페이지를 참조하십시오.
- 이제 지리 공간 쿼리에서 다음 API를 포함한 향상된 기능을 사용할 수 있습니다.
  - `$geoWithin`
  - `$geoIntersects`

- 다음 MongoDB 연산자에 대한 지원이 추가되었습니다.
  - `$mergeObjects`
  - `$reduce`

  자세한 내용은 Amazon DocumentDB에 지리 공간 데이터 쿼리를 참조하십시오.

2022년 1월 21일

새로운 특성

Amazon DocumentDB 4.0(엔진 패치 버전 2.0.5706)

- 이제 Amazon DocumentDB Graviton2(r6g.large, r6g.2xlarge, r6g.4xlarge, r6g.8xlarge, r6g.12xlarge, r6g.16xlarge, t4g.medium) 인스턴스가 지원됩니다.

Amazon DocumentDB 3.6(엔진 패치 버전 1.0.207781) 및 4.0(엔진 패치 버전 2.0.5706)

- 다음 MongoDB API에 대한 지원이 추가되었습니다.
  - `$reduce`
  - `$mergeObjects`
  - `$geoWithin`
  - `$geoIntersects`

2021년 10월 25일

새로운 특성

Amazon DocumentDB 3.6(엔진 패치 버전 1.0.207780) 및 4.0(엔진 패치 버전 2.0.5704)

- 다음 MongoDB API에 대한 지원이 추가되었습니다.
  - `$literal`
  - `$map`
  - `$$ROOT`
- 지리 공간 쿼리 기능 지원 자세한 내용은 이 블로그 게시물을 참조하십시오.
- 사용자 정의 역할을 통한 액세스 제어 지원 자세한 내용은 이 블로그 게시물을 참조하십시오.

- Amazon DocumentDB JDBC 드라이버를 사용하면 Tableau와 같은 BI 도구와 SQL Workbench와 같은 쿼리 도구에서 연결할 수 있습니다.

버그 수정 및 기타 변경

Amazon DocumentDB 3.6(엔진 패치 버전 1.0.207780) 및 4.0(엔진 패치 버전 2.0.5704)

- 다음과 같이 명시적인 `.sort()`와 `$natural`이 있을 때 올바르게 정렬되도록 `$natural` 버그 수정
- `$redact`에 사용할 변경 스트림에 대한 버그 수정
- 빈 배열과 함께 작동하기 위한 `$ifNull` 버그 수정
- 현재 로그인한 사용자를 삭제하거나 진행 중인 활동에 대한 해당 사용자의 권한이 취소될 때 과도한 리소스 소비/서버 충돌이 발생하는 버그 수정
- `listDatabase`의 버그 수정 및 `listCollection` 권한 확인
- 버그 수정 다중 키 요소에 대한 중복 제거 로직

2021년 6월 24일

새로운 특성

Amazon DocumentDB 3.6(엔진 패치 버전 1.0.207117) 및 4.0(엔진 패치 버전 2.0.3371)

- 이제 r5.8xlarge 및 r5.16xlarge 인스턴스가 지원됩니다. Amazon DocumentDB는 이제 r5.8xlarge 및 r5.16xlarge 인스턴스를 지원합니다 블로그 게시물에서 자세히 알아보십시오.
- 글로벌 클러스터는 이제 가장 가까운 Amazon DocumentDB 클러스터에서 읽기를 허용하여 지역 전반의 정전 발생 시 재해 복구를 제공하고 지연 시간이 짧은 글로벌 읽기를 가능하게 합니다. 글로벌 클러스터는 현재 남아메리카(상파울루), 유럽(밀라노), 중국(베이징) 및 중국(닝샤) 리전에서 지원되지 않습니다.

2021년 5월 4일

새로운 특성

이 블로그 게시물에서 모든 새로운 기능을 확인하십시오.

Amazon DocumentDB 3.6(엔진 패치 버전 1.0.207117) 및 4.0(엔진 패치 버전 2.0.3371)

- renameCollection
- $zip
- $indexOfArray
- $reverseArray
- $natural
- $hint업데이트 지원
- distinct의 인덱스 스캔

버그 수정 및 기타 변경

Amazon DocumentDB 3.6(엔진 패치 버전 1.0.207117) 및 4.0(엔진 패치 버전 2.0.3371)

- $in 쿼리의 메모리 사용량 감소
- 다중 키 인덱스의 메모리 누수가 수정되었습니다.
- $out에 대한 설명 계획 및 프로파일러 출력을 수정했습니다.
- 신뢰성을 개선하기 위해 내부 모니터링 시스템의 작업 제한 시간을 추가했습니다.
- 다중 키 인덱스에 전달되는 쿼리 조건자에 영향을 미치는 결함을 수정했습니다.

2021년 1월 15일

새로운 특성

Amazon DocumentDB 4.0(엔진 패치 버전 2.0.722)

- 없음

Amazon DocumentDB 3.6(엔진 패치 버전 1.0.206295)

- $lookup 집계 단계에서 인덱스를 사용할 수 있는 능력
- find() 프로젝션이 있는 쿼리는 인덱스의 방향에 따라 제공될 수 있습니다(대상 쿼리)
- findAndModify와 함께 hint() 가용성
- $addToSet 운영자를 위한 성능 최적화
- 전체 인덱스 크기를 줄이기 위한 개선

- 새 집계 연산자: `$ifNull`, `$replaceRoot`, `$setIsSubset`, `$setInstersection`, `$setUnion`, 및 `$setEquals`
- 또한 사용자는 `KillCursor` 역할 없이도 자신의 커서를 끝낼 수 있습니다.

2020년 11월 9일

새로운 특성

이 블로그 게시물에서 모든 새로운 기능을 확인하십시오.

Amazon DocumentDB 4.0(엔진 패치 버전 2.0.722)

- MongoDB 4.0 호환성
- ACID 트랜잭션
- `cluster(client.watch()`또는 `mongo.watch())` 및 데이터베이스 수준 `(db.watch())` 변경 스트림 지원
- `startAtOperationTime`을 사용하여 변경 스트림을 시작하거나 재개할 수 있습니다.
- 변경 스트림 보존 기간을 7일로 연장(이전에는 24시간)
- AWS DMS Amazon DocumentDB 4.0 대상
- CloudWatch 지표: `TransactionsOpen`, `TransactionsOpenMax`, `TransactionsAborted`, `TransactionsStarted` 및 `TransactionsCommitted`
- `currentOp`, `ServerStatus` 및 `profiler`의 트랜잭션을 위한 새 필드.
- `$lookup` 집계 단계에서 인덱스를 사용할 수 있는 능력
- `find()` 프로젝션이 있는 쿼리는 인덱스의 방향에 따라 제공될 수 있습니다(대상 쿼리)
- `findAndModify`와 함께 `hint()` 가용성
- `$addToSet` 운영자를 위한 성능 최적화
- 전체 인덱스 크기를 줄이기 위한 개선.
- 새 집계 연산자: `$ifNull`, `$replaceRoot`, `$setIsSubset`, `$setInstersection`, `$setUnion`, 및 `$setEquals`
- 이제 `ListCollection` 및 `ListDatabase` 명령을 사용하면 선택적으로 `authorizedCollections` 및 `authorizedDatabases` 매개변수를 사용하여 사용자가 각각 `listCollections` 및 `listDatabase` 역할을 요구하지 않고도 액세스 권한이 있는 컬렉션과 데이터베이스를 나열할 수 있습니다.
- 또한 사용자는 `KillCursor` 역할 없이도 자신의 커서를 끝낼 수 있습니다.

- 이제 하위 문서의 숫자 유형을 비교하는 것이 첫 번째 수준 문서의 숫자 유형을 비교하는 것과 동일합니다. Amazon DocumentDB 4.0에서의 동작은 이제 MongoDB와 호환됩니다.

Amazon DocumentDB 3.6(엔진 패치 버전 1.0.206295)

- 없음

버그 수정 및 기타 변경

Amazon DocumentDB 4.0(엔진 패치 버전 2.0.722)

- `$setOnInsert`위치 연산자를 사용할 때는 더 이상 업데이트를 허용하지 않습니다 `$`. Amazon DocumentDB 4.0에서의 동작은 이제 MongoDB와 호환됩니다.
- `$createCollection`의 문제를 해결하고 `autoIndexId`를 설정했습니다.
- 중첩된 문서에 대한 프로젝션
- 작업 메모리의 기본 설정을 인스턴스 메모리 크기에 맞게 규모를 조정하도록 변경했습니다.
- 가비지 수집 개선
- 경로에 빈 키를 사용한 조회, mongo와의 동작 차이
- 시간대 동작의 `dateToString` 버그 수정
- 정렬 순서를 준수하도록 `$push` (집계) 수정
- 집계 관련 `$currentOp` 버그 수정
- 보조의 `readPreference`에 발생하는 문제가 해결되었습니다.
- `$createIndex`이 명령이 실행된 것과 동일한 데이터베이스인지 확인하는 문제가 해결되었습니다.
- `minKey`, `maxKey` 조회 실패에 대한 일관되지 않은 동작이 수정되었습니다.
- `$size` 연산자가 복합 배열을 사용하지 않는 문제가 해결되었습니다.
- regex `$in` 부정 관련 문제가 해결되었습니다.
- 뷰에 대한 `$distinct` 명령 실행 문제가 해결되었습니다.
- 누락된 필드를 다르게 정렬하는 집계 및 찾기 명령 관련 문제가 해결되었습니다.
- 정규 표현식이 `$eq` 유형을 확인하지 않던 문제를 수정했습니다.
- 타임스탬프 서수 위치 동작의 `$currentDate` 버그 수정
- `$currentDate`에 대한 밀리초 단위 수정

Amazon DocumentDB 3.6(엔진 패치 버전 1.0.206295)

- 없음

2020년 10월 30일

새로운 특성

이 블로그 게시물에서 모든 새로운 기능을 확인하십시오.

Amazon DocumentDB 3.6(엔진 패치 버전 1.0.206295)

- 클러스터 수준 (`client.watch()` 또는 `mongo.watch()`) 및 데이터베이스 (`db.watch()`)에서 변경 스트림 커서를 여는 기능이 추가되었습니다.
- 변경 스트림 보존 기간을 7일로 연장(이전에는 24시간)

버그 수정 및 기타 변경

Amazon DocumentDB 3.6(엔진 패치 버전 1.0.206295)

- 다양한 일반 케이스 성능 개선
- 표적 보안 개선
- 복합 인덱스의 두 번째 필드에서 정렬을 건너뛰는 문제가 해결되었습니다.
- 다중 키 인덱스의 단일 필드에서 일반 인덱스가 같도록 활성화(복합 아님)
- 인증 경쟁 조건이 수정되었습니다.
- 가비지 수집 충돌이 간헐적으로 발생하던 문제가 해결되었습니다.
- RBAC 보안 개선
- `databaseConnectionsMax` 메트릭 추가
- `r5.24xlarge` 인스턴스의 특정 워크로드에 대한 성능 개선

2020년 9월 22일

새로운 특성

이 블로그 게시물에서 모든 새로운 기능을 확인하십시오.

Amazon DocumentDB 3.6(엔진 패치 버전 1.0.206295)

- `$out` 집계 단계
- 인스턴스당 최대 연결 및 커서 수를 10배까지 늘렸습니다.

버그 수정 및 기타 변경

Amazon DocumentDB 3.6(엔진 패치 버전 1.0.206295)

- 없음

2020년 7월 10일

새로운 특성

이 블로그 게시물에서 모든 새로운 기능을 확인하십시오.

Amazon DocumentDB 3.6(엔진 패치 버전 1.0.206295)

- 리전 간 스냅샷 복제

버그 수정 및 기타 변경

Amazon DocumentDB 3.6(엔진 패치 버전 1.0.206295)

- 없음

2020년 6월 30일

새로운 특성

이 블로그 게시물에서 모든 새로운 기능을 확인하십시오.

Amazon DocumentDB 3.6(엔진 패치 버전 1.0.206295)

- T3 medium 인스턴스

버그 수정 및 기타 변경

Amazon DocumentDB 3.6(엔진 패치 버전 1.0.206295)

- t3 인스턴스의 유휴 메모리 회수
- 인증 개선
- 향상된 SASL 인증 성능
- 가능한 최대 작업을 초과할 때 발생하는 `currentOp` 문제가 해결되었습니다.
- 대량 업데이트 및 삭제 관련 `killOps` 문제가 해결되었습니다.
- `$match`에서의 `$sample` 성능 개선
- 수정 단계의 $$ cond 사례의 지원이 수정되었습니다.
- 여러 가지 반복되는 크래시 근본 원인을 수정했습니다.
- I/O 및 지연 시간을 줄이기 위한 TTL 스위핑 개선
- `$unwind`에 대한 메모리 사용률 최적화
- 드롭 인덱스를 이용한 수집 통계 경쟁 조건을 수정했습니다.
- 동시 인덱스 빌드 중 경쟁 상태가 수정되었습니다.
- 인덱스의 `hash_search`에서 가끔 충돌이 발생하는 문제를 수정했습니다.

Amazon DocumentDB 개발자 안내서용 문서 기록

- API 버전: 2014-10-31
- 최종 설명서 업데이트: 2024년 7월 23일

다음 표에서는 Amazon DocumentDB 개발자 안내서의 이번 릴리스를 소개합니다.

변경 사항	설명	날짜
AWS 관리형 정책 업데이트 - 정책 변경	Amazon DocumentDB는 엘라스틱 클러스터에 대한 전체 액세스 정책을 업데이트합니다.	2025년 2월 11일
AWS 관리형 정책 업데이트 - 정책 변경	Amazon DocumentDB는 엘라스틱 클러스터에 대한 전체 액세스 정책을 업데이트합니다.	2024년 2월 21일
AWS 관리형 정책 업데이트 - 정책 변경	Amazon DocumentDB는 엘라스틱 클러스터에 대한 읽기 전용 및 전체 액세스 정책을 업데이트합니다.	2023년 6월 21일
AWS 관리형 정책 업데이트 - 새 정책	Amazon DocumentDB는 엘라스틱 클러스터에 대한 새로운 읽기 전용 정책을 도입합니다.	2023년 6월 8일
AWS 관리형 정책 업데이트 - 새 정책	Amazon DocumentDB는 엘라스틱 클러스터에 대한 새로운 액세스 정책을 도입합니다.	2023년 6월 5일
MongoDB 5.0 호환성	Amazon DocumentDB은 이제 MongoDB의 5.0 버전과 호환됩니다.	2023년 3월 1일
정책 업데이트	Amazon DocumentDB 엘라스틱 클러스터 기능을 지원하기 위해, AmazonDocDBConsole	2022년 11월 30일

	FullAccess 정책이 업데이트되고 AmazonDocDB-ElasticServiceRolePolicy 정책이 도입되었습니다.	
엘라스틱 클러스터	Amazon DocumentDB의 분산 스토리지 시스템에서 데이터의 해시 기반 파티셔닝(샤딩)을 지원하는 새로운 엘라스틱 클러스터 기능이 추가되었습니다.	2022년 11월 30일
글로벌 클러스터	글로벌 클러스터 사용 방법에 대한 설명서가 추가되었습니다.	2021년 6월 2일
이벤트 구독	이벤트 구독 설명서가 추가되었습니다.	2021년 3월 26일
버전 3.6 업그레이드	역할 기반 액세스 제어, 집계 연산자 및 성능에서 버전 3.6에 대한 개선 사항을 문서화했습니다.	2021년 1월 15일
MongoDB 4.0 호환성	Amazon DocumentDB은 이제 MongoDB의 4.0 버전과 호환됩니다.	2020년 11월 9일
시작 안내서	Amazon EC2 AWS Cloud9, Robo3T 또는 Studio3T를 사용하여 Amazon DocumentDB를 시작하기 위한 새로운 시작 안내서입니다. EC2, Robo3T	2020년 8월 15일
추가 가용 영역 지원	Amazon DocumentDB에는 아시아 태평양(서울) (ap-northeast-2)의 새로운 가용 영역이 추가되었습니다.	2020년 7월 14일

리전 간의 스냅샷 복사 지원이 추가되었습니다.	Amazon DocumentDB에 AWS 리전간의 클러스터 스냅샷을 복사하기 위한 지원이 추가되었습니다. 자세한 내용은 리전 간 스냅샷 복사를 참조하십시오.	2020년 7월 10일
T3 인스턴스 클래스에 대한 지원이 추가되었습니다.	Amazon DocumentDB를 지원하는 모든 리전에 T3 인스턴스 유형에 대한 지원이 추가되었습니다. 자세한 내용은 리전별 지원되는 인스턴스 클래스 및 인스턴스 클래스 사양을 참조하십시오.	2020년 6월 30일
에 대한 지원이 추가되었습니다 AWS GovCloud (US).	이제 AWS GovCloud (US) 리전(us-gov-west-1)에서 Amazon DocumentDB를 사용할 수 있습니다.	2020년 6월 29일
16개의 새로운 CloudWatch 지표가 추가되었습니다.	Amazon DocumentDB에 16개의 새로운 Amazon CloudWatch 지표에 대한 지원이 추가되었습니다. 자세한 내용은 CloudWatch를 사용하는 Amazon DocumentDB 모니터링을 참조하십시오.	2020년 6월 23일
null 문자 및 $regex 연산자에 대한 지원이 추가되었습니다.	Amazon DocumentDB에 문자열에서 null 문자에 대한 지원과 $regex에 대한 인덱스 사용 기능이 추가되었습니다. Amazon DocumentDB에 대해 지원되는 MongoDB API 및 집계 파이프라인 기능을 보려면 MongoDB와의 기능적 차이를 참조하십시오.	2020년 6월 22일

향상된 다중 키 인덱싱 기능에 대한 지원이 추가되었습니다.	Amazon DocumentDB에 2,048바이트보다 큰 배열 인덱싱을 포함한 다중 키로 개선된 다중 키 인덱스를 생성하는 기능 지원과 동일한 배열의 다중 키가 있는 복합 다중 키 인덱싱 기능에 대한 지원이 추가되었습니다. 자세한 내용은 MongoDB와의 기능 차이점을 참조하십시오.	2020년 4월 23일
Amazon DocumentDB AWS CloudFormation 스택 삭제 보호에 대한 지원이 추가되었습니다.	Amazon DocumentDB는 Amazon DocumentDB AWS CloudFormation 스택을 생성할 때 삭제 방지를 활성화하기 위한 지원을 추가했습니다.	2020년 4월 20일
역할 기반 액세스 제어에 대한 지원이 추가되었습니다.	Amazon DocumentDB에 기본 제공 역할을 사용하여 역할 기반 액세스를 제어하는 데 대한 지원이 추가되었습니다.	2020년 3월 26일
캐나다(중부) (ca-central-1)의 추가 가용 영역에 대한 지원이 추가되었습니다.	Amazon DocumentDB를 이제 R5 클래스 인스턴스와 3개의 가용 영역이 있는 캐나다(중부) 리전(ca-central-1)에서 사용할 수 있습니다.	2020년 3월 26일
두 가지 추가 MongoDB API에 대한 지원이 추가되었습니다.	Amazon DocumentDB에 `$dateFromString` 및 `executionStats` MongoDB API에 대한 지원이 추가되었습니다.	2020년 3월 23일

다섯 가지 추가 MongoDB API에 대한 지원이 추가되었습니다.	Amazon DocumentDB에 `$objectToArray` , `$arrayToObject` , `$slice`, `$mod` 및 `$range` MongoDB API에 대한 지원이 추가되었습니다.	2020년 2월 6일
캐나다(중부)에 대한 지원이 추가되었습니다.	Amazon DocumentDB를 이제 R5 클래스 인스턴스가 있는 캐나다(중부) 리전(ca-central-1)에서 사용할 수 있습니다.	2019년 12월 11일
ChangeStreamLogSize에 대한 지원이 추가되었습니다.	Amazon DocumentDB에 CloudWatch 지표에 대한 `ChangeStreamLogSize` 지원이 추가되었습니다.	2019년 11월 22일
유럽 (파리) 리전에 대한 지원이 추가되었습니다	Amazon DocumentDB를 이제 R5 클래스 인스턴스가 있는 유럽(파리) 리전(eu-west-3)에서 사용할 수 있습니다.	2019년 10월 30일
아시아 태평양(뭄바이) 리전에 대한 지원이 추가되었습니다	Amazon DocumentDB를 이제 R5 클래스 인스턴스가 있는 아시아 태평양(뭄바이) 리전(ap-south-1)에서 사용할 수 있습니다.	2019년 10월 17일
세 가지 추가 MongoDB API에 대한 지원이 추가되었습니다	Amazon DocumentDB에 `$addFields` , `$concatArrays` 및 `$lookup` MongoDB API에 대한 지원이 추가되었습니다.	2019년 10월 16일

아시아 태평양(싱가포르) 리전에 대한 지원이 추가되었습니다	Amazon DocumentDB를 이제 R5 클래스 인스턴스가 있는 아시아 태평양(싱가포르) (ap-southeast-1)에서 사용할 수 있습니다.	2019년 10월 14일
TLS 인증서 업데이트를 위한 새 문서가 추가되었습니다	새 CA 인증서를 사용하여 TLS 연결을 생성하기 위한 CA 인증서 업데이트 지침이 추가되었습니다.	2019년 10월 2일
인증서에 대한 API 지원이 추가되었습니다	Amazon DocumentDB는 인스턴스에 대한 새 인증서 데이터 유형입니다. 자세한 내용은 DBInstance를 참조하십시오.	2019년 10월 1일
쿼리 프로파일링 지원	Amazon DocumentDB에 클러스터의 인스턴스와 데이터베이스에서 지원되는 작업을 프로파일링할 수 있는 기능이 추가되었습니다.	2019년 8월 19일
아시아 태평양(도쿄)에 세 번째 AZ가 추가되었습니다	Amazon DocumentDB에 아시아 태평양(도쿄)의 컴퓨팅 인스턴스를 위한 세 번째 가용 영역(AZ)이 추가되었습니다.	2019년 8월 9일

추가 Mongo API 지원	`$in`, `$isoWeek`, `$isoWeekYear` , `$isoDayOfWeek` 및 `$dateToString` 집계 연산자와 `$addToSet` 집계 단계를 포함하는 추가 집계 파이프라인 기능에 대한 지원이 추가되었습니다. 또한 Amazon DocumentDB에 컬렉션 수준 진단을 위한 `top()` 명령에 대한 지원과 `collMod()` 명령을 사용하여 TTL 인덱스의 `expireAfterSeconds` 파라미터를 수정할 수 있는 기능이 추가되었습니다.	2019년 7월 31일
유럽(런던)에 대한 지원이 추가되었습니다	Amazon DocumentDB를 이제 R5 클래스 인스턴스가 있는 유럽(런던) 리전(eu-west-2)에서 사용할 수 있습니다.	2019년 7월 18일
코드 샘플이 추가됨	프로그래밍 방식으로 Amazon DocumentDB에 연결하기 위한 R 및 Ruby의 코드 예제가 추가되었습니다.	2019년 7월 17일
모범 사례가 추가됨	Amazon DocumentDB 비용 관리에 도움이 되는 모범 사례가 추가되었습니다.	2019년 7월 17일
클러스터의 중지 및 시작 지원	Amazon DocumentDB에 클러스터를 중지하고 시작하여 개발 및 테스트 환경 비용을 관리할 수 있는 지원 기능이 추가되었습니다.	2019년 7월 1일

클러스터 삭제 방지 지원	실수로 인한 삭제로부터 클러스터를 보호하기 위해, Amazon DocumentDB에 삭제 방지가 추가되었습니다. 자세한 내용은 Amazon DocumentDB 클러스터 생성, Amazon DocumentDB 클러스터 수정, Amazon DocumentDB 클러스터 삭제 및 DBCluster의 API 주제의 `DeletionProtection` 를 참조하십시오.	2019년 7월 1일
기능적 차이 업데이트	기능적 차이에 암시적 트랜잭션이 추가되었습니다.	2019년 6월 26일
기능적 차이 추가	Amazon DocumentDB에 스토리지 및 인덱스 압축 관련 참고 사항이 추가되었습니다.	2019년 6월 13일
추가 리전 지원	Amazon DocumentDB를 이제 R5 클래스 인스턴스가 있는 아시아 태평양(시드니) (ap-southeast-2)에서 사용할 수 있습니다.	2019년 6월 5일
추가 리전에서 지원되는 R5 인스턴스 클래스	미국 동부(오하이오), 미국 동부(버지니아 북부), 미국 서부(오레곤), EU(아일랜드) 등 4개의 추가 리전에 R5 인스턴스 클래스 지원이 추가되었습니다. 이 변경을 통해, R5 인스턴스는 Amazon DocumentDB를 지원하는 모든 지역에서 지원됩니다.	2019년 5월 17일

추가 리전 지원	2개의 추가 리전인 아시아 태평양(도쿄) (ap-northeast-1) 및 아시아 태평양(서울) (ap-northeast-2) 리전(R5 인스턴스 클래스 사용)에 대한 지원이 추가되었습니다. 자세한 내용은 리전별 지원되는 인스턴스 클래스 및 인스턴스 클래스 사양을 참조하십시오.	2019년 5월 8일
연결 코드 예제 추가	Java 및 C#의 Amazon DocumentDB 연결 코드 예제가 추가되었습니다.	2019년 4월 24일
추가 Mongo API 지원	7개 집계 문자열 연산자($indexOfBytes , $indexOfCP , $strLenBytes , $strLenCP , $toLower, $toUpper, $split), 9개 날짜-시간 연산자($dayOfYear , $dayOfMonth , $dayOfWeek , $year, $month, $hour, $minute, $second, $millisecond) 및 $sample 집계 파이프라인 단계에 대한 지원이 추가되었습니다.	2019년 4월 4일
연결 코드 예제 추가	Python, Node.js, PHP, Go의 Amazon DocumentDB 연결 코드 예제가 추가되었습니다.	2019년 3월 21일

프랑크푸르트 리전 및 R5 인스턴스에 대한 지원	R5 인스턴스 클래스를 사용하는 유럽(프랑크푸르트) 리전(eu-central-1)에 대한 지원이 추가되었습니다. 자세한 내용은 리전별 지원되는 인스턴스 클래스 및 인스턴스 클래스 사양을 참조하십시오.	2019년 3월 13일
집계 파이프라인 연산자 지원	새로운 집계 문자열 연산자($concat, $substr, $substrBytes , $substrCP , $strcasecmp), 배열 집계 연산자($size), 집계 그룹 누적기 연산자($push) 및 집계 단계($redact 및$indexStats) 에 대한 지원이 추가되었습니다. 또한 위치 배열 연산자($[] 및 $[<identifier>]) 및 hint()에 대한 지원도 추가되었습니다.	2019년 2월 28일
엔진 업그레이드	대기 중인 클러스터 수정 사항 확인 및 클러스터 엔진 버전 업그레이드에 대한 설명서가 추가되었습니다.	2019년 2월 15일
이벤트 감사	Amazon CloudWatch Logs를 사용한 데이터베이스 이벤트 감사에 대한 지원이 추가되었습니다.	2019년 2월 12일
빠른 시작	Amazon DocumentDB 사용을 쉽게 시작하는 데 도움이 되는 빠른 시작 주제가 추가되었습니다 AWS CloudFormation.	2019년 1월 11일

공개 릴리스	Amazon DocumentDB(MongoDB 호환)의 첫 공개 릴리스입니다. 이 릴리스에는 개발자 안내서 및 통합 리소스 관리 API 참조가 포함됩니다.	2019년 1월 9일

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.