aws

개발자 가이드

AWS Key Management Service



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

AWS Key Management Service: 개발자 가이드

Copyright © 2026 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 Amazon 계열사, 관련 업체 또는 Amazon의 지원 업체 여부에 상관없이 해당 소유자의 자산입니다.

Table of Contents

AWS Key Management Service 1
를 사용하는 이유는 무엇입니까 AWS KMS? 1
AWS KMS in AWS 리전 2
AWS KMS 요금 2
AWS KMS 서비스 수준 계약 2
액세스 AWS Key Management Service 3
AWS Management Console 3
AWS KMS 콘솔을 사용하는 데 필요한 권한 3
AWS Command Line Interface 4
AWS KMS REST API 4
AWS SDKs 4
AWS SDKs 작업 4
AWS Encryption SDK 6
AWS KMS 최종 일관성 6
하이브리드 포스트 양자 TLS 6
사용 방법 7
하이브리드 포스트 퀀텀 TLS 구성 8
자세히 알아보기 10
VPC 엔드포인트를 AWS KMS 통해에 연결 10
에 대한 VPC 엔드포인트 생성 AWS KMS 11
VPC 엔드포인트에 연결 12
VPC 엔드포인트를 사용하여 AWS KMS 리소스에 대한 액세스 제어 13
VPC 엔드포인트를 사용하는 AWS KMS 요청 로깅 17
듀얼 스택 엔드포인트 18
IPv6을 통해 사용할 수 없는 기능 18
개념 19
소개 19
설계 목표 20
AWS KMS keys 21
고객 관리형 키 23
AWS 관리형 키 24
AWS 소유 키 25
AWS KMS key 계층 구조 25
키 식별자(KeyId) 27

비대칭 키 ... 30
HMAC 키 ... 32
ML-DSA 키 ... 34
다중 리전 키 ... 34
가져온 키 구성 요소 ... 44
CloudHSM 키 저장소의 KMS 키 ... 50
외부 키 저장소의 KMS 키 ... 52
AWS KMS 암호화 필수 요소 ... 55
엔트로피 및 난수 생성 ... 55
대칭 키 작업(암호화만 해당) ... 55
비대칭 키 작업(암호화, 디지털 서명 및 서명 확인) ... 56
키 유도 함수 ... 56
AWS KMS 디지털 서명의 내부 사용 ... 57
봉투 암호화 ... 57
암호화 작업 ... 59
지원되는 알고리즘 ... 60
대칭 키 알고리즘 ... 60
비대칭 키 알고리즘 ... 61
KMS 키 액세스 및 권한 ... 63
KMS 키 정책 ... 63
KMS 키 권한 부여 ... 64
키 정책 ... 64
키 정책 생성 ... 65
기본 키 정책 ... 71
키 정책 보기 ... 86
키 정책 변경 ... 89
AWS 서비스에 대한 권한 ... 92
IAM 정책 ... 92
여러 IAM 보안 주체가 KMS 키에 액세스하도록 허용 ... 94
IAM 정책 모범 사례 ... 95
IAM 정책 설명에서 KMS 키 지정 ... 97
예제 ... 100
리소스 제어 정책 ... 106
권한 부여 ... 109
권한 부여 개념 ... 110
모범 사례 ... 116

권한 부여에 대한 액세스 제어 117
권한 부여 생성 118
권한 부여 보기 129
권한 부여 토큰 사용 131
권한 부여 사용 중지 및 취소 132
조건 키 134
AWS 전역 조건 키 134
AWS KMS 조건 키 137
AWS KMS 증명된 플랫폼의 조건 키 209
최소 권한 215
최소 권한 구현 216
ABAC(속성 기반 액세스 제어) 219
에 대한 ABAC 조건 키 AWS KMS 220
태그 또는 별칭? 222
에 대한 ABAC 문제 해결 AWS KMS 224
역할 기반 액세스 제어(RBAC) 228
크로스 계정 액세스 230
1단계: 로컬 계정에서 키 정책 문 추가 232
2단계: 외부 계정에서 IAM 정책 추가 235
에서 외부 KMS 키 사용 허용 AWS 서비스 237
다른 계정에서 KMS 키 사용 237
다중 리전 키에 대한 액세스 제어 237
다중 리전 키에 대한 인증 기본 사항 238
다중 리전 키 관리자 및 사용자 권한 부여 240
액세스 결정 244
키 정책 검사 244
IAM 정책 검사 247
권한 부여 검사 249
암호화 컨텍스트 251
암호화 컨텍스트 규칙 252
정책의 암호화 컨텍스트 252
권한 부여의 암호화 컨텍스트 253
암호화 컨텍스트 로깅 254
암호화 컨텍스트 저장 254
권한 테스트 254
DryRun이란 무엇인가요? 255

API를 통해 DryRun 지정 256
AWS KMS 권한 문제 해결 256
예제 1: 사용자의에서 KMS 키에 대한 액세스가 거부됨 AWS 계정 258
예제 2: 사용자가 다른에서 KMS 키를 사용할 권한이 있는 역할을 맡음 AWS 계정 260
용어집 263
Authentication 264
권한 부여 264
ID를 통한 인증 264
정책을 사용하여 액세스 관리 265
AWS KMS 리소스 267
KMS 키 생성 268
KMS 키를 생성하기 위한 사용 권한 270
생성할 KMS 키 유형 선택 270
대칭 암호화 KMS 키 생성 273
비대칭 KMS 키 생성 278
HMAC KMS 키 생성 284
다중 리전 프라이머리 키 생성 288
다중 리전 복제본 키 생성 293
1단계: 복제본 리전 선택 294
2단계: 복제본 키 생성 294
가져온 키 구성 요소가 있는 KMS 키 생성 300
키 구성 요소 가져오기 권한 301
가져온 키 구성 요소에 대한 요구 사항 302
1단계: 키 구성 요소 AWS KMS key 없이 생성 304
2단계: 래핑 퍼블릭 키 및 가져오기 토큰 다운로드 307
3단계: 키 구성 요소 암호화 316
4단계: 키 구성 요소 가져오기 325
키 스토어에서 KMS AWS CloudHSM 키 생성 331
CloudHSM 키 저장소에서 새로운 KMS 키 생성 332
외부 키 저장소에서 KMS 키 생성 338
외부 키 스토어의 KMS 키 요구 사항 339
외부 키 저장소에서 새로운 KMS 키 생성 340
키 식별 및 보기 348
키 ID 및 키 ARN 찾기 348
KMS 키 세부 정보 액세스 및 나열 350
다양한 키 유형 식별 359

비대칭 KMS 키 식별 ... 360
HMAC KMS 키 식별 ... 360
다중 리전 KMS 키 식별 ... 361
가져온 키 구성 요소로 KMS 키 식별 ... 362
키 스토어에서 KMS AWS CloudHSM 키 식별 ... 362
외부 키 저장소에서 KMS 키 식별 ... 363
콘솔 보기 사용자 지정 ... 364
KMS 키 정렬 및 필터링 ... 364
KMS 키 테이블 사용자 지정 ... 367
키 스토어에서 KMS 키 및 AWS CloudHSM 키 구성 요소 찾기 ... 368
키 스토어에서 KMS AWS CloudHSM 키 찾기 ... 369
키 스토어의 모든 AWS CloudHSM 키 찾기 ... 371
키의 KMS AWS CloudHSM 키 찾기 ... 372
KMS AWS CloudHSM 키의 키 찾기 ... 377
KMS 키 활성화 및 비활성화 ... 380
키 순환 ... 382
KMS 키를 교체하는 이유는 무엇인가요? ... 384
키 교체의 작동 방식 ... 384
자동 키 교체를 활성화하려면 ... 389
자동 키 교체 비활성화 ... 391
온디맨드 키 교체 수행 ... 392
온디맨드 키 교체 시작(콘솔) ... 393
온디맨드 키 교체(AWS KMS API) 시작 ... 394
교체 및 키 구성 요소 나열 ... 395
교체 및 키 구성 요소 나열(콘솔) ... 395
교체 및 키 구성 요소 나열(AWS KMS API) ... 396
수동으로 키 교체 ... 397
다중 리전 키 세트에서 프라이머리 키 변경 ... 400
기본 리전 업데이트 ... 401
키 삭제 ... 405
대기 기간에 대해 ... 406
특별 고려 사항 ... 406
키 삭제에 대한 액세스 제어 ... 409
키 관리자의 키 삭제 예약 및 취소 허용 ... 409
키 삭제 예약 ... 411
키 삭제 취소 ... 413

경보 만들기 ... 414
가져온 키 구성 요소 삭제 ... 416
데이터 키 생성 ... 419
데이터 키 생성 ... 419
데이터 키를 사용한 암호화 작업의 작동 방식 ... 420
데이터 키로 데이터 암호화 ... 420
데이터 키로 데이터 해독 ... 421
사용할 수 없는 KMS 키가 데이터 키에 미치는 영향 ... 422
데이터 키 페어 생성 ... 424
데이터 키 페어 생성 ... 424
데이터 키 페어를 사용하는 암호화 작업의 작동 방식 ... 425
데이터 키 페어로 데이터 암호화 ... 426
데이터 키 페어로 데이터 암호 해독 ... 426
데이터 키 페어를 사용하여 메시지 서명 ... 427
데이터 키 페어를 사용하여 서명 확인 ... 428
데이터 키 페어로 공유 보안 비밀 도출 ... 429
퍼블릭 키를 사용하여 오프라인 작업 수행 ... 430
퍼블릭 키 다운로드 시 특별 고려 사항 ... 430
퍼블릭 키 다운로드 ... 432
예제 오프라인 작업 ... 433
공유 보안 비밀을 오프라인으로 도출 ... 434
ML-DSA 키 페어를 사용한 오프라인 검증 ... 435
SM2 키 페어를 사용한 오프라인 인증(중국 리전 전용) ... 437
키 모니터링 ... 443
모니터링 도구 ... 444
자동 도구 ... 444
수동 도구 ... 444
를 사용하여 로깅 AWS CloudTrail ... 445
CloudTrail에서 AWS KMS 로그 항목 찾기 ... 446
추적에서 AWS KMS 이벤트 제외 ... 448
AWS KMS 로그 항목의 예 ... 449
CloudWatch를 사용하여 키 모니터링 ... 535
AWS KMS 지표 및 차원 ... 536
가져온 키 구성 요소의 만료 여부에 대한 CloudWatch 경보 생성 ... 544
외부 키 저장소에 대한 CloudWatch 경보 생성 ... 545
KMS 키의 과거 사용 결정 ... 549

KMS 키 권한 확인을 통해 사용 가능 범위 파악 550
KMS 키로 수행된 마지막 암호화 작업 검사 550
AWS CloudTrail 로그를 검사하여 과거 사용량 확인 553
Amazon EventBridge를 사용하여 키 모니터링 556
KMS CMK 로테이션 557
KMS가 가져온 키 자료 만료 557
KMS CMK 삭제 558
별칭 559
별칭의 작동 방식 559
별칭에 대한 액세스 제어 562
kms:CreateAlias 563
kms:ListAliases 564
kms:UpdateAlias 564
kms:DeleteAlias 566
별칭 권한 제한 567
별칭 생성 568
별칭 이름 및 별칭 ARN 찾기 570
별칭 업데이트 575
별칭 삭제 576
별칭을 사용하여 KMS 키에 대한 액세스 제어 577
kms:RequestAlias 579
kms:ResourceAliases 580
애플리케이션에서 별칭을 사용하는 방법 알아보기 581
AWS CloudTrail 로그에서 별칭 찾기 583
Tags 585
태그에 대한 액세스 제어 586
정책에서 태그 지정 권한 587
태그 지정 권한 제한 589
태그 추가 591
KMS 키 생성 중 태그 추가 591
기존 KMS 키에 태그 추가 592
태그 편집 593
태그 제거 594
태그 보기 595
태그를 사용하여 KMS 키에 대한 액세스 제어 597
키 저장소 601

올바른 키 스토어 선택 ... 601
AWS KMS 표준 키 스토어 ... 602
특성 ... 602
운영 ... 603
책임 모델 ... 603
AWS KMS 가져온 키 구성 요소가 있는 표준 키 스토어 ... 603
특성 ... 603
운영 ... 603
책임 모델 ... 603
AWS KMS 사용자 지정 키 스토어 ... 604
지원되는 기능 ... 604
AWS CloudHSM 키 스토어 ... 604
외부 키 스토어 ... 605
AWS CloudHSM 키 스토어 ... 606
AWS CloudHSM 키 스토어 개념 ... 609
AWS CloudHSM 키 스토어에 대한 액세스 제어 ... 611
AWS CloudHSM 키 스토어 생성 ... 613
AWS CloudHSM 키 스토어 보기 ... 619
AWS CloudHSM 키 스토어 설정 편집 ... 622
AWS CloudHSM 키 스토어 연결 ... 626
AWS CloudHSM 키 스토어 연결 해제 ... 629
AWS CloudHSM 키 스토어 삭제 ... 632
사용자 지정 키 스토어 문제 해결 ... 635
외부 키 스토어 ... 649
외부 키 스토어 개념 ... 653
외부 키 스토어 작동 방식 ... 660
외부 키 저장소에 대한 액세스 제어 ... 662
프록시 연결 옵션 선택 ... 667
외부 키 저장소 생성 ... 679
외부 키 저장소 속성 편집 ... 692
외부 키 저장소 보기 ... 698
외부 키 저장소 모니터링 ... 704
외부 키 저장소 연결 및 연결 해제 ... 715
외부 키 저장소 삭제 ... 724
외부 키 스토어 문제 해결 ... 726
보안 ... 749

데이터 보호 749
키 구성 요소 보호 750
데이터 암호화 751
인터네트워크 개인 정보 보호 752
ID 및 액세스 관리 753
AWS 관리형 정책 754
서비스 연결 역할 756
로깅 및 모니터링 762
규정 준수 확인 763
규정 준수 및 보안 문서 764
자세히 알아보기 764
복원력 765
리전별 격리 765
멀티 테넌트 디자인 766
의 복원력 모범 사례 AWS KMS 766
인프라 보안 766
물리적 호스트의 격리 768
할당량 769
리소스 할당량 769
AWS KMS keys: 100,000건 770
KMS 키당 별칭: 50 770
KMS 키당 권한 부여: 50,000 771
사용자 지정 키 스토어 리소스 할당량: 10 771
온디맨드 교체: 25 771
요청 할당량 771
각 AWS KMS API 작업에 대한 요청 할당량 772
요청 할당량 적용 779
암호화 작업에 대한 공유 할당량 780
자동으로 이루어지는 API 요청 781
교차 계정 요청 782
사용자 지정 키 스토어 요청 할당량 782
요청 제한 783
코드 예제 785
기본 사항 786
안녕하세요 AWS KMS 787
기본 사항 알아보기 791

작업 864
시나리오 1029
테이블 암호화 작업 1030
에서 암호화 증명 지원 AWS KMS 1032
에 대해 증명된 호출을 수행하는 방법 AWS KMS 1033
증명된 요청 모니터링 1034
Nitro 엔클레이브에 대한 요청 모니터링 1034
NitroTPM에 대한 요청 모니터링 1039
AWS 서비스 암호화 1045
Amazon Elastic Block Store(Amazon EBS) 1045
Amazon EBS 암호화 1046
KMS 키 및 데이터 키 사용 1046
Amazon EBS 암호화 컨텍스트 1047
Amazon EBS 오류 감지 1048
AWS CloudFormation 를 사용하여 암호화된 Amazon EBS 볼륨 생성 1048
Amazon EMR 1048
EMR 파일 시스템(EMRFS)에서 데이터 암호화 1049
클러스터 노드의 스토리지 볼륨에서 데이터 암호화 1052
암호화 컨텍스트 1053
Amazon Redshift 1053
Amazon Redshift 암호화 1054
암호화 컨텍스트 1055
레퍼런스 1056
키 상태 참조 1057
키 상태 및 KMS 키 유형 1057
키 상태 테이블 1058
키 유형 참조 1066
키 유형 표 1066
특수 기능 테이블 1073
키 사양 참조 1081
SYMMETRIC_DEFAULT 키 사양 1083
RSA 키 사양 1083
타원 곡선 키 사양 1087
HMAC KMS 키의 키 사양 1090
ML-DSA 키 사양 1091
SM2 키 사양(중국 리전 전용) 1091

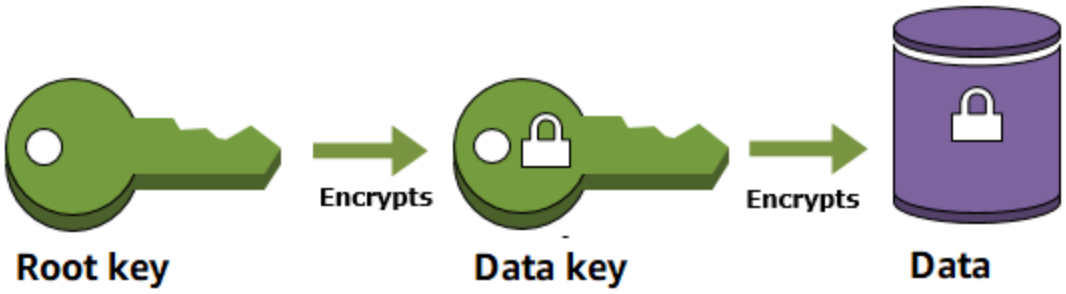
권한 참조 1092
열 설명 1137
AWS KMS 내부 작업 1139
도메인 및 도메인 상태 1139
내부 통신 보안 1143
다중 리전 키의 복제 프로세스 1146
내구성 보호 1146
문서 기록 1148
최신 업데이트 1148
이전 업데이트 1155
..... mclix

AWS Key Management Service

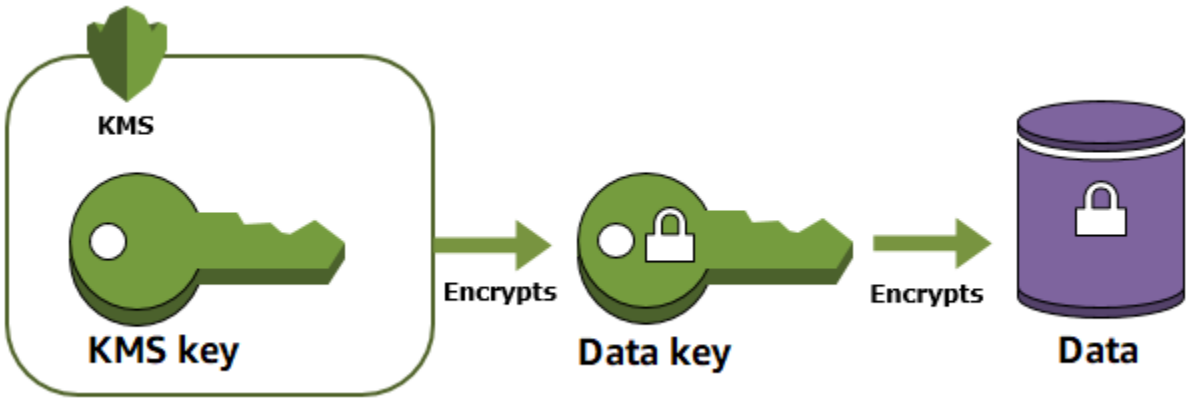
AWS Key Management Service (AWS KMS)는 데이터를 암호화하고 서명하는 데 사용되는 키를 쉽게 생성하고 제어할 수 있는 AWS 관리형 서비스입니다. 에서 AWS KMS keys 생성하는는 FIPS 140-3 보안 수준 3 검증 하드웨어 보안 모듈(HSM)로 보호 AWS KMS 됩니다. 암호화 AWS KMS 되지 않은 상태로 두지 않습니다. KMS 키를 사용하거나 관리하려면와 상호 작용합니다 AWS KMS.

를 사용하는 이유는 무엇입니까 AWS KMS?

데이터를 암호화할 때는 암호화 키를 보호해야 합니다. 키를 암호화하는 경우 암호화 키를 보호해야 합니다. 궁극적으로 데이터를 보호하는 계층 구조에서 최상위 암호화 키(루트 키 라고 함)를 보호해야 합니다. 여기에서가 AWS KMS 들어옵니다.



AWS KMS 는 루트 키를 보호합니다. KMS 키는 내에서만 생성, 관리, 사용 및 삭제됩니다 AWS KMS. KMS 키는 서비스를 암호화되지 않은 상태로 두지 않습니다. KMS 키를 사용하거나 관리하려면를 호출합니다 AWS KMS.



또한에서 키 정책을 생성하고 관리하여 신뢰할 수 있는 사용자만 KMS 키에 액세스할 수 AWS KMS있도록 할 수 있습니다.

AWS KMS in AWS 리전

AWS 리전 AWS KMS 가 지원되는는 AWS Key Management Service 엔드포인트 및 할당량에 나열됩니다. 가 지원하는에서 AWS KMS 기능이 지원되지 않는 경우 기능에 대한 주제에 리전 AWS 리전 AWS KMS 별 차이점이 설명되어 있습니다.

AWS KMS 요금

다른 AWS 제품과 마찬가지로를 사용할 때는 계약이나 최소 구매 AWS KMS 가 필요하지 않습니다. AWS KMS 요금에 대한 자세한 내용은 AWS Key Management Service 요금을 참조하세요.

AWS KMS 서비스 수준 계약

AWS Key Management Service 는 서비스 가용성 정책을 정의하는 서비스 수준 계약에 의해 지원됩니다.

액세스 AWS Key Management Service

다음과 같은 AWS KMS 방법으로를 사용할 수 있습니다.

주제

- AWS Management Console
- AWS Command Line Interface
- AWS KMS REST API
- AWS SDKs
- AWS SDK에서이 서비스 사용
- AWS Encryption SDK
- AWS KMS 최종 일관성
- 에서 하이브리드 포스트 양자 TLS 사용 AWS KMS
- VPC 엔드포인트를 AWS KMS 통해에 연결
- 듀얼 스택 엔드포인트 지원

AWS Management Console

콘솔은 AWS KMS 및 AWS 리소스를 관리하기 위한 웹 기반 사용자 인터페이스입니다. 에 가입한 경우 AWS 계정에 로그인 AWS Management Console 하고 AWS Management Console 홈 페이지에서를 선택하여 AWS KMS 콘솔 AWS KMS 에 액세스할 수 있습니다.

AWS KMS 콘솔을 사용하는 데 필요한 권한

AWS KMS 콘솔로 작업하려면 사용자가 자신의 AWS KMS 리소스로 작업할 수 있는 최소 권한 집합이 있어야 합니다 AWS 계정. 이러한 AWS KMS 권한 외에 IAM 사용자와 IAM 역할을 나열할 수 있는 권한도 가지고 있어야 합니다. 최소 필수 권한보다 더 제한적인 IAM 정책을 만들면 AWS KMS 콘솔은 해당 IAM 정책에 연결된 사용자에 대해 의도대로 작동하지 않습니다.

사용자에게 AWS KMS 콘솔에 대한 읽기 전용 액세스를 허용하기 위해 필요한 최소 권한은 사용자가 AWS KMS 콘솔에서 KMS 키를 볼 수 있도록 허용 섹션을 참조하십시오.

사용자가 AWS KMS 콘솔을 사용하여 KMS 키를 생성하고 관리할 수 있도록 하려면에 설명된 대로 AWSKeyManagementServicePowerUser 관리형 정책을 사용자에게 연결합니다AWS 에 대한 관리형 정책 AWS Key Management Service.

AWS SDK, AWS Command Line Interface 또는 AWS Tools for PowerShell를 통해 AWS KMS API로 작업하는 사용자에게는 최소 콘솔 권한을 허용할 필요가 없습니다. 그러나 API를 사용하려면 이러한 사용자에게 권한을 부여해야 합니다. 자세한 내용은 권한 참조 단원을 참조하십시오.

AWS Command Line Interface

AWS CLI 도구를 사용하여 시스템 명령줄에서 명령을 실행하거나 스크립트를 빌드하여 작업을 수행할 AWS (포함 AWS KMS) 수 있습니다.

를 AWS KMS 통한 사용에 대한 자세한 AWS CLI내용은 AWS CLI 명령 참조를 참조하세요.

AWS KMS REST API

의 아키텍처는 프로그래밍 언어 중립적이 되도록 AWS KMS 설계되었습니다. REST API는에 대한 HTTP 인터페이스입니다 AWS KMS. REST API를 통해 표준 HTTP 요청을 사용하여 키를 생성하고, 가져오며, 삭제할 수 있습니다.

사용에 대한 자세한 내용은 AWS Key Management Service API 참조를 AWS KMS REST API참조하세요.

AWS SDKs

AWS 는 일반적인 프로그래밍 언어 및 플랫폼(Java, JavaScript, C, Python 등)을 위한 라이브러리 및 샘플 코드로 구성된 SDKs(소프트웨어 개발 키트)를 제공합니다. AWS SDKs는 AWS KMS 및 AWS에 대한 프로그래밍 방식의 액세스를 생성하는 편리한 방법을 제공합니다 AWS KMS REST. 기본 AWS KMS REST API를 래핑하고 프로그래밍 작업을 간소화하는 AWS SDK 라이브러리를 AWS KMS 사용하여에 요청을 보낼 수 있습니다. 다운로드 및 설치 방법을 AWS SDKs포함하여에 대한 자세한 내용은 에 빌드하기 위한 도구를 AWS참조하세요.

는를 AWS KMS 통해를 사용하기에 좋은 시작점을 AWS SDKs AWS KMS 사용에 대한 코드 예제 제공합니다 AWS SDKs.

AWS SDK에서이 서비스 사용

AWS 소프트웨어 개발 키트(SDKs)는 널리 사용되는 많은 프로그래밍 언어에 사용할 수 있습니다. 각 SDK는 개발자가 선호하는 언어로 애플리케이션을 쉽게 구축할 수 있도록 하는 API, 코드 예제 및 설명서를 제공합니다.

SDK 설명서	코드 예제
AWS SDK for C++	AWS SDK for C++ 코드 예제
AWS CLI	AWS CLI 코드 예제
AWS SDK for Go	AWS SDK for Go 코드 예제
AWS SDK for Java	AWS SDK for Java 코드 예제
AWS SDK for JavaScript	AWS SDK for JavaScript 코드 예제
AWS SDK for Kotlin	AWS SDK for Kotlin 코드 예제
AWS SDK for .NET	AWS SDK for .NET 코드 예제
AWS SDK for PHP	AWS SDK for PHP 코드 예제
AWS Tools for PowerShell	AWS Tools for PowerShell 코드 예제
AWS Python v4용 SDK	AWS SDK for Python v4 코드 예제
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) 코드 예제
AWS SDK for Ruby	AWS SDK for Ruby 코드 예제
AWS SDK for Rust	AWS SDK for Rust 코드 예제
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP 코드 예제
AWS SDK for Swift	AWS SDK for Swift 코드 예제

예제 사용 가능 여부

필요한 예제를 찾을 수 없습니까? 이 페이지 하단의 피드백 제공 링크를 사용하여 코드 예제를 요청합니다.

AWS Encryption SDK

AWS Encryption SDK 는 애플리케이션에서 클라이언트 측 암호화를 구현하기 위한 도구입니다. KMS에 대한 전체 액세스를 제공하지는 않지만 대신 KMS 키를 참조하지 않고와 통합 AWS KMS되거나 독립형 SDK로 사용할 수 있습니다. 라이브러리는 Java, JavaScript, C, Python 및 기타 프로그래밍 언어에서 사용할 수 있습니다.

자세한 내용은 개발자 안내서AWS Encryption SDK를 참조하세요.

AWS KMS key 정책 및 IAM 정책

AWS KMS 최종 일관성

AWS KMS API는 시스템의 분산 특성으로 인해 최종 일관성 모델을 따릅니다. 따라서 AWS KMS 리소스 변경 사항이 실행하는 후속 명령에 즉시 표시되지 않을 수 있습니다.

AWS KMS API 호출을 수행할 때 변경 사항이 전체에 적용되기까지 잠시 지연될 수 있습니다 AWS KMS. 변경 사항이 시스템 전체에 전파되는 데에는 일반적으로 몇 초도 걸리지 않지만 경우에 따라 몇 분 정도 걸릴 수도 있습니다. 이 기간 동안 `NotFoundException` 또는 `InvalidStateException`과 같은 예상치 못한 오류가 발생할 수 있습니다. 예를 들어를 호출한 `GetParametersForImport` 직후를 호출`NotFoundException`하면가를 반환할 수 AWS KMS 있습니다`CreateKey`.

잠시 대기한 후 작업을 자동으로 재시도하도록 AWS KMS 클라이언트에서 재시도 전략을 구성하는 것이 좋습니다. 자세한 정보는 AWS SDK 및 도구 참조 설명서의 재시도 동작을 참조하세요.

권한 부여 관련 API 호출의 경우 권한 부여 토큰을 사용하여 잠재적인 지연을 방지하고 권한 부여에 권한을 즉시 사용할 수 있습니다. 자세한 내용은 결과적 일관성(권한 부여용)을 참조하세요.

에서 하이브리드 포스트 양자 TLS 사용 AWS KMS

(AWS KMS)에 전송하는 데이터는 전송 계층 보안 AWS Key Management Service (TLS) 연결에서 제공하는 암호화로 전송 중에 보호됩니다. AWS KMS 가 TLS 세션에 대해 지원하는 클래식 암호 제품군으로 인해 현재 기술로는 키 교환 메커니즘에 대한 무차별 암호 대입 공격이 불가능합니다. 그러나 대규모 양자 컴퓨팅이 실용적이 되면 TLS 키 교환 메커니즘에 사용되는 클래식 암호 제품군은 이러한 공격에 취약합니다.

2024년 8월에 NIST는 최초의 포스트 퀀텀 암호화 표준을 완료했습니다. 여기에는 NIST 웹 사이트의 FIPS 203에 지정된 Module-Lattice-Based Key-Encapsulation Mechanism(ML-KEM)이 포함됩니다.

AWS KMS 이제는 타원 곡선 Diffie-Hellman(ECDH)과 ML-KEM을 결합하는 TLS에 대한 하이브리드 포스트 양자 키 교환 옵션을 지원합니다. 이 하이브리드 키 교환은 오늘날의 클래식 암호 제품군의 강도를 유지하면서 향후 양자 지원 공격자에 대한 복원력을 추가합니다. 자세한 내용은 AWS 보안 블로그의 ML-KEM 포스트 양자 TLS now supported in AWS KMS, ACM 및 Secrets Manager를 참조하세요.

애플리케이션이 TLS 연결을 통해 전달되는 데이터의 장기 기밀성에 의존하는 경우 포스트 양자 암호화를 사용해야 합니다. 그렇지 않으면 공격자는 현재 클래식 암호 제품군으로 암호화된 데이터를 캡처하고 양자 컴퓨터를 사용할 수 있게 되면 암호를 해독할 수 있습니다. 이 전략을 지금 수집, 나중에 복호화라고 합니다.

에서 하이브리드 포스트 양자 TLS 사용 AWS KMS

모든 API 호출에 하이브리드 포스트 양자 TLS를 사용할 수 있습니다 AWS KMS. HTTP 클라이언트를 설정할 때 다음 정보에 유의하세요.

AWS KMS 엔드포인트

AWS KMS 는 FIPS 140-3 검증 엔드포인트를 포함한 모든 엔드포인트에서 하이브리드 포스트 양자 TLS를 사용할 수 AWS 리전 있는 모든에서 지원합니다. 각 AWS KMS 엔드포인트 목록은의 엔드포인트 및 할당량을 AWS 리전참조하세요Amazon Web Services 일반 참조. AWS Key Management Service FIPS 엔드포인트에 대한 자세한 내용은 Amazon Web Services 일반 참조의 FIPS 엔드포인트를 참조하세요.

지원되는 시스템

AWS SDKs AWS SDKs 및 도구 참조 안내서의 하이브리드 포스트 양자 TLS 활성화에 설명된 대로 하이브리드 포스트 양자 TLS를 지원합니다. Java 예제는 섹션을 참조하세요하이브리드 포스트 퀀텀 TLS 구성.

하이브리드 포스트 양자 TLS 확인

CloudTrail 로그 항목을 검사하여 특정 AWS KMS API 호출이 하이브리드 포스트 양자 TLS를 사용했는지 확인할 수 있습니다. 호출에 대한 로그 항목에서 `tlsDetails` 섹션을 찾습니다. `keyExchange` 필드는 사용된 키 교환 알고리즘의 이름을 지정합니다. 하이브리드 포스트 양자 연결의 경우이 필드는와 같은 하이브리드 알고리즘을 보여줍니다`X25519MLKEM768`. 로그 항목의 예는 섹션을 참조하세요포스트 양자 TLS 연결을 통해 표준 대칭 암호화 키로 복호화.

> **Note**
>
> 서비스가 AWS KMS 사용자를 대신하여를 호출할 때 AWS 로그 항목에 `tlsDetails` 필드가 표시되지 않습니다. CloudTrail 로그 항목의 필드와 필드가 있는 경우에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail 레코드 콘텐츠를 참조하세요.

성능

하이브리드 키 교환은 일부 TLS 핸드셰이크 메시지의 크기와 처리 시간을 약간 늘리지만 대부분의 워크로드에서 전반적인 성능에 미치는 영향은 감지할 수 없습니다. 이러한 메시지는 더 크기 때문에 네트워크의 여러 위치에서 AWS KMS API 호출을 테스트하는 것이 좋습니다.

요청이 수행하는 네트워크 경로에 따라 레거시 중간 호스트, 프록시 또는 심층 패킷 검사(DPI)가 있는 방화벽이 요청을 차단할 수 있습니다. 이는 TLS 핸드셰이크의 ClientHello 부분에 있는 새 키 교환 그룹 또는 더 큰 키 교환 메시지에서 발생할 수 있습니다. 이러한 문제를 해결하는 데 문제가 있는 경우 보안 팀 또는 IT 관리자와 협력하여 새 TLS 키 교환 그룹의 차단을 해제하세요.

전송 중 데이터 암호화

하이브리드 암호 제품군은 전송 중 암호화에만 사용됩니다. 클라이언트에서 AWS KMS 엔드포인트로 이동하는 동안 데이터를 보호합니다.는 이러한 암호 제품군을 사용하여 데이터를 암호화하지 AWS KMS 않습니다 AWS KMS keys.

대신 AWS KMS 가 KMS 키로 데이터를 암호화할 때 256비트 키를 사용한 대칭 암호화와 이미 양자 내성이 있는 Galois 카운터 모드의 고급 암호화 표준(AES-GCM) 알고리즘을 사용합니다. 256비트 AES-GCM 키로 생성된 암호화 텍스트에 대한 이론적인 미래의 대규모 양자 컴퓨팅 공격은 키의 효과적 보안을 128비트로 감소시킵니다. 자세한 내용은 ETSI 웹 사이트의 Quantum Safe Cryptography and Security 백서를 참조하세요. 이 보안 수준은 AWS KMS 사이퍼텍스트에 대한 무차별 대입 공격을 실행 불가능하게 만들기에 충분합니다.

하이브리드 포스트 퀀텀 TLS 구성

이 절차에서는 AWS 공통 런타임 HTTP 클라이언트에 대한 Maven 종속성을 추가합니다. 다음으로 포스트 양자 TLS를 선호하는 HTTP 클라이언트를 구성하세요. 그런 다음 HTTP AWS KMS 클라이언트를 사용하는 클라이언트를 생성합니다.

에서 하이브리드 포스트 양자 TLS를 구성하고 사용하는 전체 작업 예제를 보려면 GitHub의 aws-kms-pq-tls-example리포지토리를 AWS KMS참조하세요.

1. Maven 종속성에 AWS 공통 런타임 클라이언트를 추가합니다. 사용 가능한 최신 버전을 사용하는 것이 좋습니다.

   예를 들어이 문은 Maven 종속성에 AWS 공통 런타임 클라이언트 `2.30.22` 버전을 추가합니다. 버전 `2.30.22` 이상을 사용하여 ML-KEM을 활성화합니다.

   ```
   <dependency>
       <groupId>software.amazon.awssdk</groupId>
       <artifactId>aws-crt-client</artifactId>
       <version>2.30.22</version>
   </dependency>
   ```

2. 하이브리드 포스트 양자 암호 제품군을 활성화하려면 프로젝트에 AWS SDK for Java 2.x 를 추가하고 초기화합니다. 이어서 다음 예제와 같이 HTTP 클라이언트에서 하이브리드 포스트 양자 암호 제품군을 활성화합니다.

   이 코드는 `postQuantumTlsEnabled()` 메서드 파라미터를 사용하여 권장되는 하이브리드 양자 내성 암호 제품군인 ML-KEM이 포함된 ECDH를 선호하는 AWS 공용 런타임 HTTP 클라이언트를 구성합니다. 그런 다음 구성된 HTTP 클라이언트를 사용하여 AWS KMS 비동기 클라이언트의 인스턴스를 빌드합니다`KmsAsyncClient`. 이 코드가 완료된 후에 `KmsAsyncClient` 인스턴스의 모든 AWS KMS API 요청은 하이브리드 포스트 양자 TLS를 사용합니다.

   ```
   // Configure HTTP client
   SdkAsyncHttpClient awsCrtHttpClient = AwsCrtAsyncHttpClient.builder()
              .postQuantumTlsEnabled(true)
              .build();

   // Create the AWS KMS async client
   KmsAsyncClient kmsAsync = KmsAsyncClient.builder()
             .httpClient(awsCrtHttpClient)
             .build();
   ```

3. 하이브리드 포스트 양자 TLS를 사용하여 AWS KMS 호출을 테스트합니다.

   구성된 AWS KMS 클라이언트에서 AWS KMS API 작업을 호출하면 하이브리드 포스트 양자 TLS를 사용하여 AWS KMS 엔드포인트로 호출이 전송됩니다. 구성을 테스트하려면와 같은 AWS KMS API를 호출합니다`ListKeys`.

   ```
   ListKeysReponse keys = kmsAsync.listKeys().get();
   ```

호출에 하이브리드 포스트 양자 TLS가 사용되었는지 확인하려면에 설명된 대로 CloudTrail 로그 항목을 검사합니다하이브리드 포스트 양자 TLS 확인.

에서 포스트 양자 TLS에 대해 자세히 알아보기 AWS KMS

에서 하이브리드 포스트 양자 TLS를 사용하는 방법에 대한 자세한 내용은 다음 리소스를 AWS KMS참조하세요.

- 블로그 게시물 및 연구 논문에 대한 링크를 AWS포함하여의 포스트 양자 암호화에 대해 알아보려면 포스트 양자 암호화를 참조하세요.
- AWS KMS 는 하이브리드 포스트 양자 TLS 연결의 서버 측 AWS에 대해의 s2n-tls오픈 소스 TLS 구현인를 사용합니다. 에 대한 자세한 내용은 새로운 오픈 소스 TLS 구현s2n-tls인 소개 및 GitHub에서 사용을 s2n-tls s2n-tls참조하세요.
- AWS 공통 런타임 HTTP 클라이언트에 대한 자세한 내용은 AWS SDK for Java 2.x 개발자 안내서의 AWS CRT 기반 HTTP 클라이언트 구성을 참조하세요.
- 국립 표준 기술 연구소(NIST)의 포스트 양자 암호화 프로젝트에 대한 자세한 내용은 포스트 양자 암호화를 참조하세요.
- NIST 포스트 양자 암호화 표준화에 대한 자세한 내용을 확인하려면 포스트 양자 암호화 표준화를 참조하세요.

VPC 엔드포인트를 AWS KMS 통해에 연결

Virtual Private Cloud(VPC)의 프라이빗 인터페이스 엔드포인트를 AWS KMS 통해에 직접 연결할 수 있습니다. 인터페이스 VPC 엔드포인트를 사용하는 경우 VPC와 간의 통신 AWS KMS 은 전적으로 AWS 네트워크 내에서 수행됩니다.

AWS KMS 는 로 구동되는 Amazon Virtual Private Cloud(VPC) 엔드포인트를 지원합니다AWS PrivateLink. 각 VPC 엔드포인트는 하나 이상의 탄력적 네트워크 인터페이스(ENI) 및 VPC 서브넷의 프라이빗 IP 주소로 표현됩니다.

인터페이스 VPC 엔드포인트는 인터넷 게이트웨이, NAT 디바이스, VPN 연결 또는 AWS Direct Connect 연결 AWS KMS 없이 VPC를에 직접 연결합니다. VPC의 인스턴스는 통신에 퍼블릭 IP 주소가 필요하지 않습니다 AWS KMS.

리전

AWS KMS 는 AWS 리전 AWS KMS가 지원되는 모든에서 VPC 엔드포인트 및 VPC 엔드포인트 정책을 지원합니다.

AWS KMS VPC 엔드포인트에 대한 고려 사항

에 대한 인터페이스 VPC 엔드포인트를 설정하기 전에 AWS PrivateLink 가이드의 인터페이스 엔드포인트 속성 및 제한 주제를 AWS KMS검토하세요.

AWS KMS VPC 엔드포인트에 대한 지원에는 다음이 포함됩니다.

- VPC 엔드포인트를 사용하여 VPC에서 모든 AWS KMS API 작업을 호출할 수 있습니다.
- AWS KMS 리전 엔드포인트 또는 AWS KMS FIPS 엔드포인트에 연결하는 인터페이스 VPC 엔드포인트를 생성할 수 있습니다.
- AWS CloudTrail 로그를 사용하여 VPC 엔드포인트를 통해 KMS 키 사용을 감사할 수 있습니다. 자세한 내용은 VPC 엔드포인트를 사용하는 AWS KMS 요청 로깅을 참조하세요.

주제

- 에 대한 VPC 엔드포인트 생성 AWS KMS
- AWS KMS VPC 엔드포인트에 연결
- VPC 엔드포인트를 사용하여 AWS KMS 리소스에 대한 액세스 제어
- VPC 엔드포인트를 사용하는 AWS KMS 요청 로깅

에 대한 VPC 엔드포인트 생성 AWS KMS

Amazon VPC 콘솔 또는 Amazon VPC API를 AWS KMS 사용하여에 대한 VPC 엔드포인트를 생성할 수 있습니다. 다음 값 중 하나를 사용하여 인터페이스 엔드포인트를 생성하는 절차를 따릅니다.

- 에 대한 VPC 엔드포인트를 생성하려면 다음 서비스 이름을 AWS KMS사용합니다.

```
com.amazonaws.region.kms
```

예를 들어 미국 서부(오레곤) 리전(us-west-2)에서 서비스 이름은 다음과 같습니다.

```
com.amazonaws.us-west-2.kms
```

- AWS KMS FIPS 엔드포인트에 연결하는 VPC 엔드포인트를 생성하려면 다음 서비스 이름을 사용합니다.

```
com.amazonaws.region.kms-fips
```

예를 들어 미국 서부(오레곤) 리전(us-west-2)에서 서비스 이름은 다음과 같습니다.

```
com.amazonaws.us-west-2.kms-fips
```

VPC 엔드포인트를 더 쉽게 사용하려면 VPC 엔드포인트에 프라이빗 DNS 이름을 사용하도록 설정합니다. Enable DNS Name(DNS 이름 활성화) 옵션을 선택하면 표준 AWS KMS DNS 호스트 이름이 VPC 엔드포인트로 확인됩니다. 예를 들어 `https://kms.us-west-2.amazonaws.com`은 서비스 이름 `com.amazonaws.us-west-2.kms`에 연결된 VPC 엔드포인트로 확인됩니다.

이 옵션을 선택하면 VPC 엔드포인트를 더 쉽게 사용할 수 있습니다. AWS SDKs는 기본적으로 표준 AWS KMS DNS 호스트 이름을 AWS CLI 사용하므로 애플리케이션 및 명령에서 VPC 엔드포인트 URL을 지정할 필요가 없습니다.

자세한 내용은 AWS PrivateLink 가이드의 인터페이스 엔드포인트를 통해 서비스 액세스를 참조하세요.

AWS KMS VPC 엔드포인트에 연결

AWS SDK, AWS CLI또는를 사용하여 VPC 엔드포인트를 AWS KMS 통해에 연결할 수 있습니다 AWS Tools for PowerShell. VPC 엔드포인트를 지정하려면 해당 DNS 이름을 사용합니다.

예를 들어 이 list-keys 명령은 `endpoint-url` 파라미터를 사용해 VPC 엔드포인트를 지정합니다. 이러한 명령을 사용하려면 VPC 엔드포인트 ID 예제를 본인 계정의 ID로 바꿉니다.

```
$ aws kms list-keys --endpoint-url https://vpce-1234abcdf5678c90a-09p7654s-us-east-1a.ec2.us-east-1.vpce.amazonaws.com
```

필수 권한

VPC 엔드포인트를 사용하는 AWS KMS 요청이 성공하려면 보안 주체에 다음 두 소스의 권한이 필요합니다.

- 키 정책, IAM 정책 또는 권한 부여는 보안 주체에 리소스(KMS 키 또는 별칭)에 대한 작업을 호출할 수 있는 권한을 부여해야 합니다.
- VPC 엔드포인트 정책은 보안 주체에 엔드포인트를 사용하여 요청을 수행할 권한을 부여해야 합니다.

예를 들어 키 정책은 보안 주체에 특정 KMS 키에서 Decrypt를 호출할 권한을 부여합니다. 그러나 VPC 엔드포인트 정책에서 해당 보안 주체가 엔드포인트를 사용하여 해당 KMS 키에서 `Decrypt`를 호출하는 것을 허용하지 않을 수 있습니다.

또는 VPC 엔드포인트 정책에서 보안 주체가 엔드포인트를 사용하여 KMS 키에서 DisableKey를 호출할 수 있도록 합니다 그러나 보안 주체가 키 정책, IAM 정책 또는 권한 부여의 권한을 가지고 있지 않으면 요청이 실패합니다.

엔드포인트를 생성할 때 VPC 엔드포인트 정책을 생성할 수 있으며, 언제든지 VPC 엔드포인트 정책을 변경할 수 있습니다. VPC 관리 콘솔이나 CreateVpcEndpoint 또는 ModifyVpcEndpoint 작업을 사용합니다. AWS CloudFormation 템플릿을 사용하여 VPC 엔드포인트 정책을 생성하고 변경할 수도 있습니다. VPC 관리 콘솔 사용에 대한 도움말은 AWS PrivateLink 설명서의 인터페이스 엔드포인트 생성 및 인터페이스 엔드포인트 수정을 참조하세요.

프라이빗 호스트 이름

VPC 엔드포인트를 만들 때 프라이빗 호스트 이름을 사용하도록 설정한 경우 CLI 명령 또는 애플리케이션 구성에 VPC 엔드포인트 URL을 지정할 필요가 없습니다. 표준 AWS KMS DNS 호스트 이름이 VPC 엔드포인트로 확인됩니다. AWS CLI 및 SDKs는 기본적으로이 호스트 이름을 사용하므로 스크립트 및 애플리케이션에서 아무것도 변경하지 않고 VPC 엔드포인트를 사용하여 AWS KMS 리전 엔드포인트에 연결할 수 있습니다.

프라이빗 호스트 이름을 사용하려면 VPC의 `enableDnsHostnames` 및 `enableDnsSupport` 속성을 `true`로 설정해야 합니다. 이러한 속성을 설정하려면 ModifyVpcAttribute 작업을 사용합니다. 자세한 내용은 Amazon VPC 사용 설명서의 VPC에 대한 DNS 속성 보기 및 업데이트 섹션을 참조하세요.

VPC 엔드포인트를 사용하여 AWS KMS 리소스에 대한 액세스 제어

요청이 VPC에서 왔거나 VPC 엔드포인트를 사용하는 경우 AWS KMS 리소스 및 작업에 대한 액세스를 제어할 수 있습니다. 이렇게 하려면 키 정책 또는 IAM 정책에서 다음 전역 조건 키 중 하나를 사용합니다.

- `aws:sourceVpce` 조건 키를 사용해 VPC 엔드포인트를 기반으로 액세스 권한을 부여하거나 액세스를 제한합니다.
- `aws:sourceVpc` 조건 키를 사용해 프라이빗 엔드포인트를 호스팅하는 VPC를 기반으로 액세스를 부여하거나 제한합니다.

> **Note**
>
> VPC 엔드포인트를 기반으로 키 정책과 IAM 정책을 작성할 때 주의해야 합니다. 정책 설명에 특정 VPC 또는 VPC 엔드포인트의 요청이 필요한 경우 사용자를 대신하여 AWS KMS 리소스를 사용하는 통합 AWS 서비스의 요청이 실패할 수 있습니다. 도움말은 AWS KMS 권한이 있는 정책에서 VPC 엔드포인트 조건 사용를 참조하십시오.
> 또한 요청이 Amazon VPC 엔드포인트에서 이루어지는 경우 `aws:sourceIP` 조건 키는 유효하지 않습니다. 요청을 VPC 엔드포인트로 제한하려면 `aws:sourceVpce` 또는 `aws:sourceVpc` 조건 키를 사용합니다. 자세한 내용은AWS PrivateLink 가이드의 VPC 엔드포인트 및 VPC 엔드포인트 서비스에 대한 ID 및 액세스 관리 섹션을 참조하세요.

이러한 전역 조건 키를 사용하여 AWS KMS keys (KMS 키), 별칭 및 특정 리소스에 의존하지 않는 CreateKey와 같은 작업에 대한 액세스를 제어할 수 있습니다.

예를 들어 다음 샘플 키 정책은 요청이 지정된 VPC 엔드포인트를 사용하는 경우에만 사용자가 KMS 키로 일부 암호화 작업을 수행하도록 허용합니다. 사용자가에 요청하면 요청 AWS KMS의 VPC 엔드포인트 ID가 정책의 `aws:sourceVpce` 조건 키 값과 비교됩니다. 두 값이 일치하지 않는 경우 요청이 거부됩니다.

이와 같은 정책을 사용하려면 자리 표시자 AWS 계정 ID 및 VPC 엔드포인트 IDs 계정의 유효한 값으로 바꿉니다.

JSON

```
{
    "Id": "example-key-1",
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "EnableIAMpolicies",
            "Effect": "Allow",
            "Principal": {"AWS":["111122223333"]},
            "Action": ["kms:*"],
            "Resource": "*"
        },
        {
            "Sid": "Restrict usage to my VPC endpoint",
            "Effect": "Deny",
            "Principal": "*",
```

```
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*"
            ],
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                "aws:sourceVpce": "vpce-1234abcdf5678c90a"
                }
            }
        }

    ]
}
```

또한 aws:sourceVpc 조건 키를 사용하여 VPC 엔드포인트가 있는 VPC를 기반으로 KMS 키에 대한 액세스를 제한할 수 있습니다.

다음 샘플 키 정책은 KMS 키가 vpc-12345678에서 이루어진 경우에만 KMS 키를 관리하는 명령을 허용합니다. 또한 KMS 키를 사용하는 명령이 vpc-2b2b2b2b에서 이루어진 경우에만 암호화 작업에 사용할 수 있습니다. 애플리케이션이 하나의 VPC에서 실행 중이지만 관리 용도로 VPC를 하나 더 사용하는 경우, 이와 같은 정책을 사용할 수 있습니다.

이와 같은 정책을 사용하려면 자리 표시자 AWS 계정 ID 및 VPC 엔드포인트 IDs 계정의 유효한 값으로 바꿉니다.

JSON

```
{
    "Id": "example-key-2",
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAdministrativeActionsFromVPC",
            "Effect": "Allow",
            "Principal": {
                "AWS": "111122223333"
            },
            "Action": [
```

```
                "kms:Create*",
                "kms:Enable*",
                "kms:Put*",
                "kms:Update*",
                "kms:Revoke*",
                "kms:Disable*",
                "kms:Delete*",
                "kms:TagResource",
                "kms:UntagResource"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:sourceVpc": "vpc-12345678"
                }
            }
        },
        {
            "Sid": "AllowKeyUsageFromVPC2b2b2b2b",
            "Effect": "Allow",
            "Principal": {
                "AWS": "111122223333"
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:GenerateDataKey*"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:sourceVpc": "vpc-2b2b2b2b"
                }
            }
        },
        {
            "Sid": "AllowReadActionsEverywhere",
            "Effect": "Allow",
            "Principal": {
                "AWS": "111122223333"
            },
            "Action": [
                "kms:Describe*",
                "kms:List*",
```

```
                "kms:Get*"
            ],
            "Resource": "*"
        }
    ]
}
```

VPC 엔드포인트를 사용하는 AWS KMS 요청 로깅

AWS CloudTrail 는 VPC 엔드포인트를 사용하는 모든 작업을 기록합니다. 에 대한 요청이 VPC 엔드포인트를 AWS KMS 사용하는 경우 요청을 기록하는 AWS CloudTrail 로그 항목에 VPC 엔드포인트 ID가 나타납니다. 엔드포인트 ID를 사용하여 AWS KMS VPC 엔드포인트 사용을 감사할 수 있습니다.

그러나 CloudTrail 로그에는 다른 계정의 보안 주체가 요청한 작업이나 다른 계정의 KMS 키 및 별칭에 대한 AWS KMS 작업 요청은 포함되지 않습니다. 또한 VPC를 보호하기 위해 VPC 엔드포인트 정책에 의해 거부되었지만 그렇지 않았으면 허용되었을 요청은 AWS CloudTrail에 기록되지 않습니다.

예를 들어, 이 샘플 로그 항목은 VPC 엔드포인트를 사용한 GenerateDataKey 요청을 기록합니다. 로그 항목 끝에 `vpcEndpointId` 필드가 나타납니다.

```
{
  "eventVersion":"1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "accountId": "111122223333",
    "userName": "Alice"
  },
  "eventTime":"2018-01-16T05:46:57Z",
  "eventSource":"kms.amazonaws.com",
  "eventName":"GenerateDataKey",
  "awsRegion":"eu-west-1",
  "sourceIPAddress":"172.01.01.001",
  "userAgent":"aws-cli/1.14.23 Python/2.7.12 Linux/4.9.75-25.55.amzn1.x86_64
 botocore/1.8.27",
  "requestParameters":{
    "keyId":"1234abcd-12ab-34cd-56ef-1234567890ab",
    "numberOfBytes":128
  },
```

```
  "responseElements":null,
  "requestID":"a9fff0bf-fa80-11e7-a13c-afcabff2f04c",
  "eventID":"77274901-88bc-4e3f-9bb6-acf1c16f6a7c",
  "readOnly":true,
  "resources":[{
    "ARN":"arn:aws:kms:eu-
west-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "accountId":"111122223333",
    "type":"AWS::KMS::Key"
  }],
  "eventType":"AwsApiCall",
  "recipientAccountId":"111122223333",
  "vpcEndpointId": "vpce-1234abcdf5678c90a"
}
```

듀얼 스택 엔드포인트 지원

AWS KMS 는 IPv4 및 IPv6 클라이언트를 모두 지원하는 듀얼 스택 퍼블릭 엔드포인트를 제공합니다. 듀얼 스택 엔드포인트를 사용하면 클라이언트가 IPv4 또는 IPv6 주소를 AWS KMS 사용하여와 통신할 수 있습니다. AWS KMS 엔드포인트에 대한 자세한 내용은 AWS Key Management Service 엔드포인트 및 할당량을 참조하세요.

의 AWS KMS 듀얼 스택 퍼블릭 엔드포인트는 IPv4 및 IPv6 클라이언트를 모두 `https://kms.`*`your-region`*`.api.aws` 지원합니다. AWS KMS 는를 사용하여 가상 프라이빗 클라우드(VPC)에서 IPv4 및 IPv6를 통해 비공개로 액세스할 수도 있습니다 AWS PrivateLink. 에 대한 프라이빗 인터페이스 VPC 엔드포인트 생성에 대한 자세한 내용은 섹션을 AWS KMS참조하세요VPC 엔드포인트를 AWS KMS 통해에 연결.

VPC의 IPv6 주소 지정에 대한 자세한 내용은 Amazon Virtual Private Cloud 사용 설명서의 Amazon VPC의 작동 원리를 참조하세요. 듀얼 스택 모드로 VPC를 구성하는 방법에 대한 자세한 내용은 Amazon 가상 프라이빗 클라우드 사용 설명서의 VPC 및 서브넷에 대한 IP 주소 지정을 참조하세요.

IPv6을 통해 사용할 수 없는 기능

AWS KMS 는 IPv6를 통해 AWS CloudHSM 키 스토어 또는 외부 키 스토어와 통신할 수 없습니다. 이 제한으로 인해 IPv6를 통해 AWS KMS APIs 호출할 수 없습니다.

AWS KMS 개념

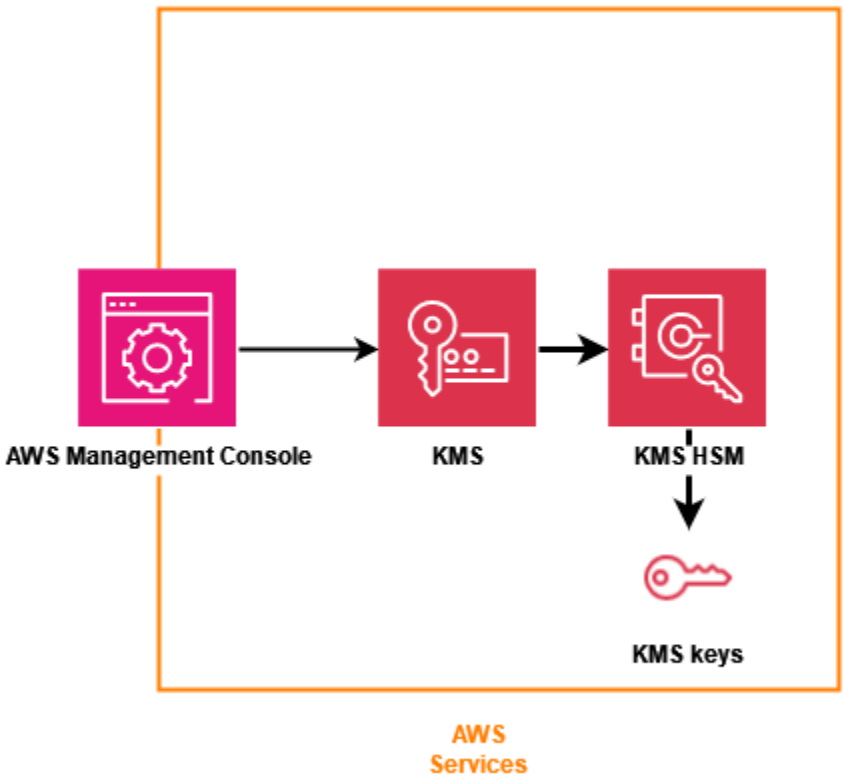
AWS Key Management Service (AWS KMS)에 사용되는 기본 용어 및 개념과 이러한 용어 및 개념이 함께 작동하여 데이터를 보호하는 방법을 알아봅니다.

소개 AWS KMS

AWS Key Management Service (AWS KMS)는 암호화 키를 생성 및 관리할 수 있는 웹 인터페이스를 제공하며 데이터를 보호하기 위한 암호화 서비스 공급자 역할을 합니다. AWS KMS 는 AWS 서비스와 통합된 기존 키 관리 서비스를 제공하여 중앙 집중식 관리 및 감사를 AWS통해 고객의 키를 전체적으로 일관되게 볼 수 있습니다.

AWS KMS 에는 FIPS 140-3 검증 하드웨어 보안 모듈(HSM)의 분산 플릿의 암호화 작업을 요청하는 AWS Management Console명령줄 인터페이스 및 RESTful API 작업을 통한 웹 인터페이스가 포함되어 있습니다.HSMs AWS KMS HSM은의 보안 및 확장성 요구 사항을 충족하는 전용 암호화 기능을 제공하도록 설계된 멀티칩 독립 실행형 하드웨어 암호화 어플라이언스입니다 AWS KMS. AWS KMS keys와 같이 사용자가 관리하는 키로 HSM 기반 암호화 계층 구조를 설정할 수 있습니다. 이러한 키는 HSM에서만 사용할 수 있으며 암호화 요청을 처리하는 데 필요한 시간 동안 메모리에서만 사용할 수 있습니다. KMS 키를 여러 개 생성할 수 있으며 각 키는 키 ID로 표시됩니다. 각 고객이 관리하는 AWS IAM 역할 및 계정에서만 고객 관리형 KMS 키를 생성, 삭제 또는 사용하여 데이터를 암호화, 복호화, 서명 또는 확인할 수 있습니다. 키에 연결된 정책을 생성하여 KMS 키를 관리하거나 사용하도록 허용되는 사용자에 대한 액세스 제어를 정의할 수 있습니다. 이러한 정책을 사용하면 각 API 작업을 위한 키의 애플리케이션별 용도를 정의할 수 있습니다.

또한 대부분의 AWS 서비스는 KMS 키를 사용하여 저장 데이터 암호화를 지원합니다. 이 기능을 통해 고객은 KMS 키에 액세스하는 방법과 시기를 제어하여 AWS 서비스가 암호화된 데이터에 액세스하는 방법과 시기를 제어할 수 있습니다.



AWS KMS 는 웹 경계 AWS KMS 호스트와 HSMs. 이러한 계층형 호스트의 그룹화는 AWS KMS 스택을 형성합니다. 에 대한 모든 요청은 전송 계층 보안 프로토콜(TLS)을 통해 이루어져야 하며 AWS KMS host. AWS KMS hosts에서 종료 AWS KMS 되어야 합니다.는 완벽한 전달 보안을 제공하는 ciphersuite가 있는 TLS만 허용합니다. AWS KMS 는 다른 모든 AWS API 작업에 사용할 수 있는 AWS Identity and Access Management (IAM)의 동일한 자격 증명 및 정책 메커니즘을 사용하여 요청을 인증하고 승인합니다.

AWS KMS 설계 목표

AWS KMS 는 다음 요구 사항을 충족하도록 설계되었습니다.

내구성

암호화 키의 내구성은에서 가장 내구성이 높은 서비스와 같도록 설계되었습니다 AWS. 단일 암호화 키를 사용하여 장기간 누적된 대량의 데이터를 암호화할 수 있습니다.

신뢰성

키 사용은 사용자가 정의하고 관리하는 액세스 제어 정책으로 보호됩니다. 일반 텍스트 KMS 키를 내보내는 메커니즘은 없습니다. 암호화 키의 기밀성은 매우 중요합니다. HSM에서 관리 작업을 수

행하려면 쿼럼 기반 액세스 제어에 대한 역할별 액세스 권한을 가진 여러 Amazon 직원이 필요합니다.

짧은 대기 시간과 높은 처리량

AWS KMS 는의 다른 서비스에서 사용하기에 적합한 지연 시간 및 처리량 수준에서 암호화 작업을 제공합니다 AWS.

독립적인 리전

AWS 는 서로 다른 리전에서 데이터 액세스를 제한해야 하는 고객에게 독립적인 리전을 제공합니다. 키 사용은 단일 AWS 리전으로 제한됩니다.

난수의 보안 소스

강력한 암호화는 실제로 예측할 수 없는 난수 생성에 의존하기 때문에는 고품질의 검증된 난수 소스를 AWS KMS 제공합니다.

감사

AWS KMS 는 AWS CloudTrail 로그에서 암호화 키의 사용 및 관리를 기록합니다. AWS CloudTrail 로그를 사용하여 사용자를 대신하여 AWS 서비스의 키 사용을 포함하여 암호화 키 사용을 검사할 수 있습니다.

이러한 목표를 달성하기 위해 AWS KMS 시스템에는 “도메인”을 관리하는 일련의 AWS KMS 연산자 및 서비스 호스트 연산자(총칭하여 “작업자”)가 포함됩니다. 도메인은 리전별로 정의된 AWS KMS 서버, HSMs 및 연산자 집합입니다. 각 AWS KMS 운영자에는 작업을 인증하는 데 사용되는 프라이빗 및 퍼블릭 키 페어가 포함된 하드웨어 토큰이 있습니다. HSM에는 HSM 상태 동기화를 보호하는 암호화 키를 설정하기 위한 추가 프라이빗 및 퍼블릭 키 페어가 있습니다.

AWS KMS keys

자체 암호화 애플리케이션에서 사용하기 위해 생성하고 관리하는 KMS 키는 고객 관리형 키라고 하는 유형입니다. 고객 관리형 키는 KMS 키를 사용하여 AWS 서비스가 사용자를 대신하여 저장하는 데이터를 암호화하는 서비스와 함께 사용할 수도 있습니다. 고객 관리형 키는 해당 키의 수명 주기와 사용을 완전히 제어하려는 고객에게 권장됩니다. 계정에 고객 관리형 키를 보유하는 데는 월별 비용이 발생합니다. 또한 요청 사용 및/또는 키 관리에는 사용 비용이 발생합니다. 자세한 내용은 AWS Key Management Service 요금을 참조하세요.

고객이 AWS 서비스에서 데이터를 암호화하기를 원하지만 키 관리의 오버헤드를 원하지 않고 키에 대한 비용을 지불하고 싶지 않은 경우가 있습니다. AWS 관리형 키는 계정에 존재하지만 특정 상황에서만 사용할 수 있는 KMS 키입니다. 특히 운영 중인 AWS 서비스의 컨텍스트에서만 사용할 수 있으며 키

가 있는 계정 내의 보안 주체만 사용할 수 있습니다. 이러한 키의 수명 주기 또는 권한은 관리할 수 없습니다. AWS 서비스에서 암호화 기능을 사용하면 'aws<service code>' 형식의 별칭을 사용할 수 AWS 관리형 키있습니다. 예를 들어 aws/ebs 키는 EBS 볼륨을 암호화하는 데만 사용할 수 있으며 키와 동일한 계정의 IAM 위탁자가 사용하는 볼륨에만 사용할 수 있습니다. 계정의 사용자만 계정의 리소스에 사용할 수 AWS 관리형 키 있도록 범위가 축소된를 생각해 보세요. 에서 암호화된 리소스를 다른 계정 AWS 관리형 키 과 공유할 수 없습니다. 계정에 AWS 관리형 키 가 무료로 존재하지만 키에 할당된 AWS 서비스에서이 키 유형을 사용하는 경우 요금이 부과됩니다.

AWS 관리형 키 는 2021년부터 새 AWS 서비스에 대해 더 이상 생성되지 않는 레거시 키 유형입니다. 대신 새로운(및 레거시) AWS 서비스는 기본적으로 라고 하는 AWS 소유 키를 사용하여 고객 데이터를 암호화합니다. AWS 소유 키 는 AWS 서비스에서 관리하는 계정에 있는 KMS 키이므로 서비스 운영자는 수명 주기 및 사용 권한을 관리할 수 있습니다. 를 사용하면 AWS 소유 키 AWS 서비스가 데이터를 투명하게 암호화하고 키 권한에 대해 걱정할 필요 없이 쉽게 교차 계정 또는 교차 리전에서 데이터를 공유할 수 있습니다. 더 쉽고, 더 자동화된 데이터 보호를 제공하는 기본 암호화 워크로드에 AWS 소유 키 를 사용합니다. 이러한 키는에서 소유 및 관리하므로 해당 키의 존재 또는 사용에 대한 요금이 부과 AWS되지 않으며, 정책을 변경할 수 없고, 이러한 키에 대한 활동을 감사할 수 없으며, 삭제할 수 없습니다. 제어가 중요한 경우 고객 관리형 키를 사용하고 편의성이 가장 중요한 경우 AWS 소유 키 를 사용합니다.

	고객 관리형 키	AWS 관리형 키	AWS 소유 키
키 정책	고객이 독점적으로 제어	서비스에서 제어, 고객이 볼 수 있음	데이터를 암호화하는 AWS 서비스에서만 독점적으로 제어되고 볼 수 있음
로깅	CloudTrail 고객 추적 또는 이벤트 데이터 저장소	CloudTrail 고객 추적 또는 이벤트 데이터 저장소	고객이 볼 수 없음
수명 주기 관리	고객이 교체, 삭제 및 리전 위치 관리	AWS KMS 는 교체(연간), 삭제 및 리전 위치를 관리합니다.	AWS 서비스 는 교체, 삭제 및 리전 위치를 관리합니다.
요금	키 존재에 대한 월별 요금(시간당 비례 배분 기준). 키 사용에 대해서도 요금이 부과됨	월별 수수료는 없지만 호출자에게 이러한 키의 API 사용에 대한 요금이 부과됨	고객에게 부과되는 요금 없음

사용자가 생성하는 KMS 키는 고객 관리형 키입니다. KMS 키를 사용하여 서비스 리소스를 암호화하는 AWS 서비스 는 종종 사용자를 위해 키를 만듭니다. AWS 계정에서가 AWS 서비스 생성하는 KMS 키는 입니다AWS 관리형 키. 서비스 계정에서를 AWS 서비스 생성하는 KMS 키는 입니다AWS 소유 키.

KMS 키의 유형	KMS 키 메타데이터 보기 가능	KMS 키 관리 가능	내에서 생성됨 AWS 계정	자동 회전	요금
고객 관리형 키	예	예	예	선택 사항.	월 요금(시간당 비례 배분) 사용량 기준 요금
AWS 관리형 키	예	아니요	예	필수 사항입니다. 매년(약 365일)	월 요금 없음 사용당 요금(일부는이 요금을 대신 AWS 서비스 지불)
AWS 소유 키	아니요	아니요	아니요	는 교체 전략을 AWS 서비스 관리합니다.	요금 없음

AWS 와 통합되는 서비스는 AWS KMS KMS 키에 대한 지원이 다릅니다. 일부 AWS 서비스는 기본적으로 AWS 소유 키 또는를 사용하여 데이터를 암호화합니다 AWS 관리형 키. 일부 AWS 서비스는 고객 관리형 키를 지원합니다. 다른 AWS 서비스는 모든 유형의 KMS 키를 지원하므로의 용이성 AWS 소유 키, 가시성 AWS 관리형 키또는 고객 관리형 키의 제어가 가능합니다. AWS 서비스가 제공하는 암호화 옵션에 대한 자세한 내용은 사용 설명서의 저장 시 암호화 주제 또는 서비스의 개발자 안내서를 참조하세요.

고객 관리형 키

사용자가 생성하는 KMS 키는 고객 관리형 키입니다. 고객 관리형 키는 AWS 계정 사용자가 생성, 소유 및 관리하는의 KMS 키입니다. 사용자는 키 정책, IAM 정책 및 권한 부여의 설정 및 유지 관리, 활성화

및 비활성화, 암호화 구성 요소 교체, 태그 추가, KMS 키를 가리키는 별칭 생성, KMS 키 삭제 예약 등을 포함해 이러한 KMS 키에 대한 완전한 제어 권한을 가집니다.

고객 관리형 키는 AWS KMS에 대한 AWS Management Console 의 고객 관리형 키 페이지에 나타납니다. 고객 관리형 키를 명확하게 식별하려면 DescribeKey 작업을 사용합니다. 고객 관리형 키에서는 `DescribeKey` 응답의 `KeyManager` 필드 값이 `CUSTOMER`입니다.

암호화 작업에서 사용자 관리형 키를 사용하고 AWS CloudTrail 로그에서 사용을 감사할 수 있습니다. 뿐만 아니라 AWS KMS에 통합된 다양한AWS 서비스 덕분에 사용자를 위해 저장 및 관리하는 데이터를 보호하도록 고객 관리형 키를 지정할 수 있습니다.

고객 관리형 키는 매달 요금이 발생하며, 프리 티어를 초과해서 사용한 만큼 요금이 부과됩니다. 계정 AWS KMS 의 할당량에 포함됩니다. 자세한 내용은 AWS Key Management Service 요금 및 할당량 섹션을 참조하세요.

AWS 관리형 키

AWS 관리형 키는 AWS 와 통합된 서비스가 AWS KMS 사용자를 대신하여 생성, 관리 및 사용하는 계정의 KMS 키입니다.

일부 AWS 서비스를 사용하면 AWS 관리형 키 또는 고객 관리형 키를 선택하여 해당 서비스의 리소스를 보호할 수 있습니다. 일반적으로 리소스를 보호하는 암호화 키를 제어해야 하는 경우가 아니면 AWS 관리형 키 를 선택하는 것이 좋습니다. 키 또는 키 정책을 만들거나 유지할 필요가 없으며 AWS 관리형 키에 대한 월 요금도 없습니다.

계정에서 를 보고 AWS 관리형 키, 키 정책을 보고, AWS CloudTrail 로그에서 사용을 감사할 수 있는 권한이 있습니다. 그러나의 속성을 변경하거나 AWS 관리형 키, 교체하거나, 키 정책을 변경하거나, 삭제를 예약할 수 없습니다. 또한 암호화 작업에 AWS 관리형 키 를 직접 사용할 수 없습니다. 암호화 작업을 생성하는 서비스는 사용자를 대신하여 암호화 작업을 사용합니다.

조직의 리소스 제어 정책은 AWS 관리형 키에 적용되지 않습니다.

AWS 관리형 키 는 AWS Management Console 의 AWS 관리형 키 페이지에 표시됩니다 AWS KMS. 와 `aws/`*`service-name`*같은 형식의 별칭 AWS 관리형 키 으로 식별할 수도 있습니다`aws/redshift`. 를 명확하게 식별하려면 DescribeKey 작업을 AWS 관리형 키사용합니다. AWS 관리형 키에서는 `DescribeKey` 응답의 `KeyManager` 필드 값이 `AWS`입니다.

모든 AWS 관리형 키 는 매년 자동으로 교체됩니다. 이 교체 일정은 변경할 수 없습니다.

> **Note**
>
> 2022년 5월에의 교체 일정을 3년마다(약 1,095일) AWS 관리형 키 에서 1년마다(약 365일)로 AWS KMS 변경했습니다.

에 대한 월별 요금은 없습니다 AWS 관리형 키. 프리 티어를 초과하는 사용 요금은 부과될 수 있지만 일부 AWS 서비스에서는 이러한 비용을 부담합니다. 자세한 내용은 서비스에 대한 사용 설명서 또는 개발자 안내서의 저장 시 암호화 주제를 참조하세요. 자세한 내용은 AWS Key Management Service 요금을 참조하세요.

AWS 관리형 키 는 계정의 각 리전에 있는 KMS 키 수에 대한 리소스 할당량에 포함되지 않습니다. 그러나 계정의 보안 주체 대신 사용되는 KMS 키는 요청 할당량에 포함됩니다. 자세한 내용은 할당량을 참조하세요.

AWS 소유 키

AWS 소유 키는 AWS 서비스가 여러에서 사용하기 위해 소유하고 관리하는 KMS 키 모음입니다 AWS 계정. AWS 소유 키 는에 없지만 AWS 계정 AWS 서비스는를 AWS 소유 키 사용하여 계정의 리소스를 보호할 수 있습니다.

일부 AWS 서비스를 사용하면 AWS 소유 키 또는 고객 관리형 키를 선택할 수 있습니다. 일반적으로 리소스를 보호하는 암호화 키를 감사하거나 제어해야 하는 경우가 아니면 AWS 소유 키 를 선택하는 것이 좋습니다. AWS 소유 키 는 완전히 무료(월별 요금 또는 사용 요금 없음)이며 계정AWS KMS 의 할당량에 포함되지 않으며 사용하기 쉽습니다. 키 또는 키 정책을 만들거나 유지하지 않아도 됩니다.

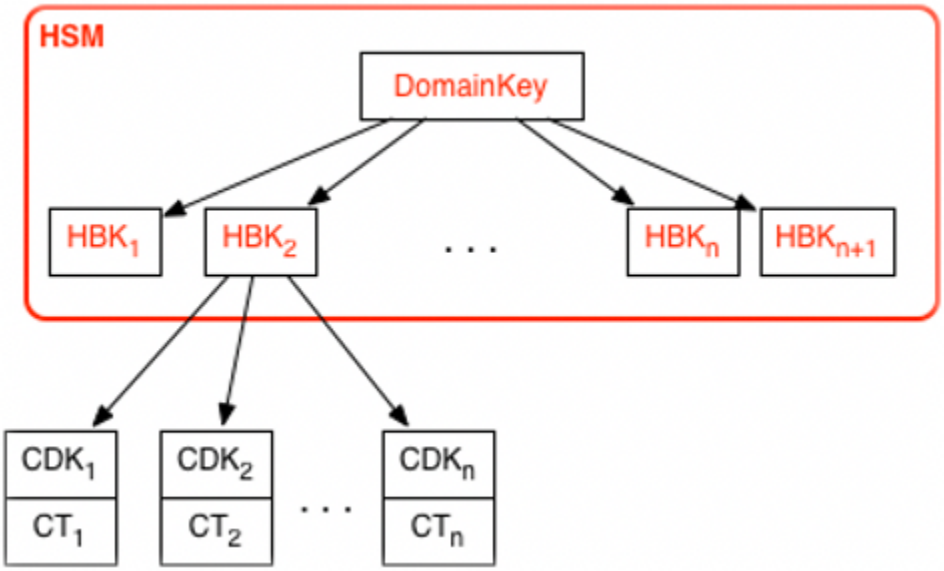
의 교체는 서비스마다 AWS 소유 키 다릅니다. 특정의 교체에 대한 자세한 내용은 서비스 사용 설명서 또는 개발자 안내서의 저장 시 암호화 주제를 AWS 소유 키참조하세요.

AWS KMS key 계층 구조

키 계층 구조는 최상위 논리적 키인 로 시작합니다 AWS KMS key. KMS 키는 최상위 키 구성 요소의 컨테이너를 나타내며, Amazon 리소스 이름(ARN)을 사용하여 AWS 서비스 네임스페이스에 고유하게 정의됩니다. ARN에는 고유하게 생성된 키 식별자인 키 ID가 포함되어 있습니다. KMS 키는 사용자가 시작한 요청을 기반으로 생성됩니다 AWS KMS. 수신 시는 KMS 키 컨테이너에 배치할 초기 HSM 백업 키(HBK) 생성을 AWS KMS 요청합니다. HBK는 도메인의 HSM에 생성되며 HSM에서 일반 텍스트로 내보낼 수 없도록 설계되었습니다. 대신 HBK는 HSM 관리 도메인 키로 암호화되어 내보내집니다. 이러한 내보낸 HBK를 내보낸 키 토큰(EKT)이라고 합니다.

EKT는 내구성이 높고 대기 시간이 짧은 스토리지로 내보내집니다. 예를 들어 논리적 KMS 키에 대한 ARN 수신한다고 가정합니다. 이는 키 계층 구조 또는 암호화 컨텍스트의 최상위를 나타냅니다. 계정 내에 여러 KMS 키를 생성하고 다른 AWS 명명된 리소스와 마찬가지로 KMS 키에 정책을 설정할 수 있습니다.

특정 KMS 키의 계층 구조 내에서 HBK는 KMS 키의 한 버전이라고 할 수 있습니다. KMS 키를 교체하려는 경우 AWS KMS새 HBK가 생성되고 KMS 키에 대한 활성 HBK로 KMS 키와 연결됩니다. 이전 HBK는 보존되며 이전에 보호된 데이터를 복호화하고 확인하는 데 사용할 수 있습니다. 단, 새 정보를 보호하는 데에는 활성 암호화 키만 사용할 수 있습니다.



AWS KMS 를 통해 KMS 키를 사용하여 정보를 직접 보호하도록 요청하거나 KMS 키로 보호되는 추가 HSM 생성 키를 요청할 수 있습니다. 이러한 키를 고객 데이터 키(CDK)라고 합니다. CDK는 암호화 텍스트(CT), 일반 텍스트 또는 두 가지 형식 모두로 암호화된 상태로 반환될 수 있습니다. KMS 키(고객 제공 데이터 또는 HSM 생성 키)로 암호화된 모든 객체는 직접 호출을 통해 HSM에서만 복호화할 수 있습니다 AWS KMS.

반환된 사이퍼텍스트 또는 해독된 페이로드는 내에 저장되지 않습니다 AWS KMS. 이 정보는 AWS KMS에 대한 TLS 연결을 통해 사용자에게 반환됩니다. 이는 사용자를 대신하여 AWS 서비스에서 수행한 호출에도 적용됩니다.

다음 표에는 키 계층 구조 및 특정 키 속성이 나와 있습니다.

Key(키)	설명	수명 주기
도메인 키	HSM 메모리에만 있는 256비트 AES-GCM 키로, KMS 키의 버전, 즉 HSM 백업 키를 래핑하는 데 사용됩니다.	매일 교체됨[1]
HSM 백업 키	256비트 대칭 키 또는 RSA 또는 타원 곡선 개인 키로, 고객 데이터 및 키를 보호하는 데 사용되며 도메인 키로 암호화됩니다. 하나 이상의 HSM 백업 키가 KMS 키를 구성하며 keyId로 표시됩니다.	매년 교체됨[2](선택적 구성)
추출된 암호화 키	HSM 메모리에만 있는 256비트 AES-GCM 키로, 고객 데이터 및 키를 암호화하는 데 사용됩니다. 각 암호화에 대해 HBK에서 추출됩니다.	암호화당 한 번 사용되며 복호화할 때 다시 생성됩니다.
고객 데이터 키	HSM에서 일반 텍스트 및 암호화 텍스트 형식으로 내보낸 사용자 정의 대칭 또는 비대칭 키입니다. HSM 백업 키로 암호화되며, TLS 채널을 통해 권한 있는 사용자에게 반환됩니다.	교체 및 애플리케이션에 의한 통제된 사용

[1] AWS KMS 도메인 관리 및 구성 작업을 고려하기 위해 때때로 도메인 키 교체를 최대 주 단위로 완화할 수 있습니다.

[2] 사용자를 대신하여 AWS KMS 에서 AWS 관리형 키 생성하고 관리하는 기본값은 매년 자동으로 교체됩니다.

키 식별자(KeyId)

키 식별자는 KMS 키의 이름과 같은 역할을 합니다. 콘솔에서 KMS 키를 인식하는 데 도움이 됩니다. 이를 사용하여 AWS KMS API 작업, 키 정책, IAM 정책 및 권한 부여에 사용할 KMS 키를 나타냅니다. 키 식별자 값은 KMS 키와 연결된 키 자료와 전혀 관련이 없습니다.

AWS KMS 는 여러 키 식별자를 정의합니다. KMS 키를 생성할 때는 KMS 키의 속성인 키 ARN 및 키 ID를 AWS KMS 생성합니다. 별칭을 생성할 때는 사용자가 정의한 별칭 이름을 기반으로 별칭 ARN을 AWS KMS 생성합니다. 및 AWS KMS API에서 키 AWS Management Console 및 별칭 식별자를 볼 수 있습니다.

AWS KMS 콘솔에서 키 ARN, 키 ID 또는 별칭 이름을 기준으로 KMS 키를 보고 필터링하고 키 ID 및 별칭 이름을 기준으로 정렬할 수 있습니다. 콘솔에서 키 식별자를 찾는 방법에 대한 도움말은 the section called “키 ID 및 키 ARN 찾기” 섹션을 참조하세요.

AWS KMS API에서 KMS 키를 식별하는 데 사용하는 파라미터의 이름은 `KeyId` 또는와 같은 `TargetKeyId` 또는 변형입니다`DestinationKeyId`. 그러나 이러한 파라미터의 값은 키 ID로 제한되지 않습니다. 일부는 유효한 키 식별자를 취할 수 있습니다. 각 파라미터의 값에 대한 자세한 내용은 AWS Key Management Service API 참조의 파라미터 설명을 참조하세요.

> **Note**
>
> AWS KMS API를 사용할 때는 사용하는 키 식별자에 주의해야 합니다. API에 따라 다른 키 식별자가 필요합니다. 일반적으로 작업에 가장 완전하고 실용적인 키 식별자를 사용하십시오.

AWS KMS 는 다음과 같은 키 식별자를 지원합니다.

키 ARN

키 ARN은 KMS 키의 Amazon 리소스 이름(ARN)입니다. KMS 키에 대한 고유한 정규화된 식별자입니다. 키 ARN에는 AWS 계정, 리전 및 키 ID가 포함됩니다. KMS 키의 키 ARN을 찾는 방법에 대한 도움말은 the section called “키 ID 및 키 ARN 찾기” 섹션을 참조하세요.

키 ARN의 형식은 다음과 같습니다.

```
arn:<partition>:kms:<region>:<account-id>:key/<key-id>
```

다음은 단일 리전 KMS 키의 키 ARN 예입니다.

```
arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab
```

다중 리전 키의 키 ARN의 *`key-id`* 요소는 `mrk-` 접두사로 시작합니다. 다음은 다중 리전 키에 대한 예제 키 ARN입니다.

```
arn:aws:kms:us-west-2:111122223333:key/mrk-1234abcd12ab34cd56ef1234567890ab
```

키 ID

키 ID는 계정 및 리전 내의 KMS 키를 고유하게 식별합니다. KMS 키의 키 ID를 찾는 방법에 대한 도움말은 the section called “키 ID 및 키 ARN 찾기” 섹션을 참조하세요.

다음은 단일 리전 KMS 키의 키 ID 예입니다.

```
1234abcd-12ab-34cd-56ef-1234567890ab
```

다중 리전 키의 키 ID는 `mrk-` 접두사로 시작합니다. 다음은 다중 리전 키에 대한 예제 키 ID입니다.

```
mrk-1234abcd12ab34cd56ef1234567890ab
```

별칭 ARN

별칭 ARN은 AWS KMS 별칭의 Amazon 리소스 이름(ARN)입니다. 별칭 및 해당 별칭이 나타내는 KMS 키에 대한 고유한 정규화된 식별자입니다. 별칭 ARN에는 AWS 계정, 리전 및 별칭 이름이 포함됩니다.

지정된 시간에 별칭 ARN은 하나의 특정 KMS 키를 식별합니다. 그러나 별칭과 연결된 KMS 키를 변경할 수 있으므로 별칭 ARN이 서로 다른 시간에 서로 다른 KMS 키를 식별할 수 있습니다. KMS 키의 별칭 ARN을 찾는 방법에 대한 도움말은 KMS 키의 별칭 이름 및 별칭 ARN 찾기 섹션을 참조하세요.

별칭 ARN의 형식은 다음과 같습니다.

```
arn:<partition>:kms:<region>:<account-id>:alias/<alias-name>
```

다음은 가상 `ExampleAlias`의 별칭 ARN입니다.

```
arn:aws:kms:us-west-2:111122223333:alias/ExampleAlias
```

별칭 이름

별칭 이름은 최대 256자의 문자열입니다. 계정 및 리전 내의 연결된 KMS 키를 고유하게 식별합니다. AWS KMS API에서 별칭 이름은 항상 로 시작합니다`alias/`. KMS 키의 별칭 이름을 찾는 방법에 대한 도움말은 KMS 키의 별칭 이름 및 별칭 ARN 찾기 섹션을 참조하세요.

별칭 이름의 형식은 다음과 같습니다.

```
alias/<alias-name>
```

예:

```
alias/ExampleAlias
```

별칭 이름의 aws/ 접두사는 AWS 관리형 키용으로 예약됩니다. 이 접두사를 가진 별칭은 만들 수 없습니다. 예를 들어 Amazon Simple Storage Service(Amazon S3) AWS 관리형 키 용의 별칭 이름은 다음과 같습니다.

```
alias/aws/s3
```

의 비대칭 키 AWS KMS

비대칭 KMS 키는 수학적으로 관련된 퍼블릭 키 및 프라이빗 키 페어를 나타냅니다. 퍼블릭 키는 신뢰할 수 없더라도 누구에게나 제공할 수 있지만 프라이빗 키는 비밀로 유지해야 합니다.

비대칭 KMS 키에서 프라이빗 키는에서 생성 AWS KMS 되며 암호화되지 않은 상태로 두 AWS KMS 지 않습니다. 프라이빗 키를 사용하려면를 호출해야 합니다 AWS KMS. AWS KMS API 작업을 호출 AWS KMS 하여 내에서 퍼블릭 키를 사용할 수 있습니다. 또는 퍼블릭 키를 다운로드하여 외부에서 사용할 수 있습니다 AWS KMS.

를 호출할 수 없는 사용자가 외부 AWS 에서 암호화해야 하는 사용 사례의 경우 AWS KMS비대칭 KMS 키를 선택하는 것이 좋습니다. 그러나 AWS 서비스에서 저장하거나 관리하는 데이터를 암호화하기 위해 KMS 키를 생성하는 경우 대칭 암호화 KMS 키를 사용합니다. AWS 와 통합된 서비스는 AWS KMS 대칭 암호화 KMS 키만 사용하여 데이터를 암호화합니다. 이러한 서비스는 비대칭 KMS 키를 사용한 암호화를 지원하지 않습니다.

로 4KB보다 큰 메시지에 서명 AWS KMS하는 경우 서명 AWS KMS 하기 전에 외부에서 메시지를 해시해야 합니다. AWS KMS 는 메시지 입력을 처리하기 RAW 위한 세 가지 MessageType 옵션을 제공합니다. 일반 텍스트 메시지의 경우(가 해시를 AWS KMS 수행하는 경우), 미리 해시된 메시지의 DIGEST 경우(가 해시 단계를 AWS KMS 건너뛰는 경우), EXTERNAL_MU 특히 입력이 64바이트 대표 μ 값인 ML-DSA KMS 키 사양의 경우 입니다. 4KB 제한을 초과하는 큰 메시지의 경우 메시지를 외부에서 해시하고 AWS KMS Sign 및 AWS KMS Verify 작업을 호출할 때 MessageType:DIGEST (또는 ML-DSA KMS 키MessageType:EXTERNAL_MU의 경우)를 사용합니다.

AWS KMS 는 여러 유형의 비대칭 KMS 키를 지원합니다.

RSA KMS 키

암호화 및 복호화 또는 서명 및 확인을 위한 RSA 키 페어가 있는 KMS 키(둘 다 아님).는 다양한 보안 요구 사항에 대해 여러 키 길이를 AWS KMS 지원합니다.

가 RSA KMS 키에 대해 AWS KMS 지원하는 암호화 및 서명 알고리즘에 대한 기술적 세부 정보는 RSA 키 사양을 참조하세요.

타원 곡선(ECC) KMS 키

서명 및 확인 또는 공유 보안 암호 도출(둘 다 아님)을 위한 타원 곡선 키 페어가 있는 KMS 키입니다.는 일반적으로 사용되는 여러 곡선을 AWS KMS 지원합니다.

가 ECC KMS 키에 대해 AWS KMS 지원하는 서명 알고리즘에 대한 기술적 세부 정보는 타원 곡선 키 사양을 참조하세요.

ML-DSA KMS 키

서명 및 검증을 위한 ML-DSA 키 페어가 있는 KMS 키입니다. ML-DSA는 양자 컴퓨팅으로 인해 발생할 보안 위협에 대응하기 위해 미국 국립표준기술원(NIST)이 개발한 양자 내성 암호 표준입니다. ML-DSA는 RSA 또는 타원 곡선 기반 디지털 서명 알고리즘에서 안전한 양자 내성 암호로 전환하려는 조직에 권장되는 디지털 서명 알고리즘입니다.

AWS KMS 는 다양한 보안 요구 사항에 대해 여러 키 길이를 지원합니다. 가 ML-DSA KMS 키에 대해 AWS KMS 지원하는 서명 알고리즘에 대한 기술적 세부 정보는 ML-DSA 키 사양을 참조하세요.

SM2 KMS 키 사양(중국 리전만 해당)

암호화 및 복호화, 서명 및 검증 또는 공유 비밀 도출(한 가지 Key usage 유형을 선택해야 함)을 위한 SM2 키 페어가 있는 KMS 키입니다.

가 SM2 KMS 키에 대해 AWS KMS 지원하는 암호화 및 서명 알고리즘에 대한 기술적 세부 정보는 SM2 키 사양을 참조하세요(중국 리전만 해당).

비대칭 키 구성 선택에 대한 도움말은 생성할 KMS 키 유형 선택 섹션을 참조하세요.

리전

비대칭 KMS 키 및 비대칭 데이터 키 페어는가 지원하는 모든에서 AWS KMS 지원 AWS 리전 됩니다.

자세히 알아보기

- 비대칭 KMS 키를 만들려면 비대칭 KMS 키 생성 섹션을 참조하세요.
- 다중 리전 비대칭 KMS 키를 만들려면 다중 리전 프라이머리 키 생성 섹션을 참조하세요.
- 비대칭 KMS 키를 사용하여 메시지에 서명하고 서명을 확인하는 방법을 알아보려면 AWS 보안 블로그의 새로운 비대칭 키 기능으로 디지털 서명 AWS KMS을 참조하세요.
- 비대칭 KMS 키 삭제에 대한 특별 고려 사항에 대한 자세한 내용은 Deleting asymmetric KMS keys 단원을 참조하세요.
- 비대칭 KMS 키를 식별하고 보려면 비대칭 KMS 키 식별 단원을 참조하세요.

의 HMAC 키 AWS KMS

해시 기반 메시지 인증 코드(HMAC) KMS 키는 AWS KMS내에서 HMAC를 생성 및 확인하는 데 사용하는 대칭 키입니다. 각 HMAC KMS 키와 관련된 고유한 키 구성 요소는 HMAC 알고리즘에 필요한 비밀 키를 제공합니다. `GenerateMac` 및 `VerifyMac` 작업과 함께 HMAC KMS 키를 사용하여 AWS KMS내에서 데이터 무결성 및 신뢰성을 확인할 수 있습니다.

HMAC 알고리즘은 암호화 해시 함수와 공유 비밀 키를 결합합니다. HMAC KMS 키의 키 구성 요소와 같은 메시지와 비밀 키를 가져와서 고정된 크기의 고유한 코드 또는 태그를 반환합니다. 메시지의 한 문자라도 변경되거나 비밀 키가 동일하지 않은 경우 결과 태그는 완전히 달라집니다. 비밀 키를 요구함으로써 HMAC는 신뢰성도 제공합니다. 비밀 키가 없다면 동일한 HMAC 태그를 생성할 수 없습니다. HMAC는 디지털 서명처럼 작동하지만 서명과 확인 모두에 단일 키를 사용하기 때문에 대칭 서명이라고도 합니다.

HMAC KMS 키와를 AWS KMS 사용하는 HMAC 알고리즘은 RFC 2104에 정의된 업계 표준을 준수합니다. AWS KMS GenerateMac 작업은 표준 HMAC 태그를 생성합니다. HMAC KMS 키는 FIPS 140-3 암호화 모듈 검증 프로그램(중국(베이징) 및 중국(닝샤) 리전 제외)에 따라 인증된 AWS KMS 하드웨어 보안 모듈에서 생성되며 암호화 AWS KMS 되지 않은 상태로 두지 않습니다. https://csrc.nist.gov/projects/cryptographic-module-validation-program/certificate/4884 HMAC KMS 키를 사용하려면 AWS KMS를 호출해야 합니다.

HMAC KMS 키를 사용하여 JSON 웹 토큰(JWT), 토큰화된 신용 카드 정보 또는 제출된 암호와 같은 메시지의 신뢰성을 결정할 수 있습니다. 특히 결정적 키가 필요한 애플리케이션에서 안전한 키 추출 함수(KDF)로 사용할 수도 있습니다.

HMAC KMS 키는 키 구성 요소가 키에 설정한 액세스 제어에 AWS KMS따라 완전히 생성되고 사용되므로 애플리케이션 소프트웨어의 HMACs에 비해 이점이 있습니다.

> **Tip**
>
> 모범 사례에서는 HMAC을 비롯한 모든 서명 메커니즘이 유효한 시간을 제한하는 것이 좋습니다. 이렇게 하면 행위자가 서명된 메시지를 사용하여 반복적으로나 메시지가 대체된 후 유효성을 설정하는 공격을 차단합니다. HMAC 태그에는 타임스탬프가 포함되지 않지만 토큰이나 메시지에 타임스탬프를 포함시켜 HMAC 새로 고침 시간을 감지할 수 있습니다.

지원되는 암호화 작업

HMAC KMS 키는 `GenerateMac` 및 `VerifyMac` 암호화 작업만을 지원합니다. HMAC KMS 키를 사용하여 데이터를 암호화하거나 메시지에 서명하거나 HMAC 작업에서 다른 유형의 KMS 키를 사용할 수 없습니다. `GenerateMac` 작업을 사용할 때 최대 4,096바이트의 메시지, HMAC KMS 키 및 HMAC 키 사양과 호환되는 MAC 알고리즘을 제공하고, `GenerateMac`이 HMAC 태그를 계산합니다. HMAC 태그를 확인하려면 HMAC 태그와 `GenerateMac`에서 원래 HMAC 태그를 계산하는 데 사용하는 동일한 메시지, HMAC KMS 키 및 MAC 알고리즘을 제공해야 합니다. `VerifyMac` 작업은 HMAC 태그를 계산하고 제공된 HMAC 태그와 동일한지 확인합니다. 입력 태그와 계산된 HMAC 태그가 동일하지 않으면 확인이 실패합니다.

HMAC KMS 키는 자동 키 교체를 지원하지 않으며 사용자 지정 키 스토어에서 HMAC KMS 키를 생성할 수 없습니다.

KMS 키를 생성하여 AWS 서비스의 데이터를 암호화하는 경우 대칭 암호화 키를 사용합니다. HMAC KMS 키를 사용할 수 없습니다.

리전

HMAC KMS 키는가 지원하는 모든에서 AWS KMS 지원 AWS 리전 됩니다.

자세히 알아보기

- HMAC KMS 키를 생성하려면 HMAC KMS 키 생성 단원을 참조하세요.
- 다중 리전 HMAC KMS 키를 생성하려면 의 다중 리전 키 AWS KMS 단원을 참조하세요.
- AWS KMS 콘솔이 HMAC KMS 키에 대해 설정하는 기본 키 정책의 차이를 검사하려면 섹션을 참조하세요the section called “키 사용자가 암호화 작업에 KMS 키를 사용하도록 허용”.
- HMAC KMS 키를 식별하고 보려면 HMAC KMS 키 식별 단원을 참조하세요.
- HMAC을 사용하여 JSON 웹 토큰을 생성하는 방법에 대한 자세한 내용은 AWS 보안 블로그의 AWS KMS내부에서 HMAC를 보호하는 방법을 참조하세요.

- 공식 AWS 팟캐스트에서 AWS Key Management Service를 위한 HMAC 소개 팟캐스트를 들어보세요.

의 ML-DSA 키 AWS KMS

AWS Key Management Service (AWS KMS)는 포스트 퀀텀 암호화 서명에 대해 ML-DSA(모듈 래티스 디지털 서명 알고리즘)를 지원합니다. 이 구현은 향후 양자 컴퓨팅 위협에 대비하기 위해 연방 정보 처리 표준(FIPS) 204 표준을 따릅니다. AWS KMS 는 FIPS 140-3 보안 수준 3 검증된 하드웨어 보안 모듈에서 모든 ML-DSA 키 및 서명 작업을 생성하고 보호합니다. 의 ML-DSA는 보안과 성능의 균형을 맞추기 위해 다양한 키 사양인 ML_DSA_44, ML_DSA_65 및 ML_DSA_87을 통해 세 가지 보안 수준을 AWS KMS 제공합니다.

AWS KMS 는 메시지 유형을 사용하여 최대 4KB의 RAW 메시지에 대해 비대칭 키 서명을 지원합니다. 더 큰 메시지의 경우, NIST FIPS 204 섹션 6.2에 정의된 대로 ML-DSA 서명에 사용되는 64바이트 메시지 표현 μ를 외부에서 컴퓨팅해야 합니다. AWS KMS 서명 작업의 EXTERNAL_MU 메시지 유형을 사용하여이 사전 처리된 64바이트 메시지를 지정합니다. 외부에서 컴퓨팅된 μ로 생성된 서명은 동일한 메시지와 프라이빗 키를 사용할 경우 RAW에서 생성된 서명과 동일합니다. 이 서명은 NIST FIPS 204 섹션 5.4의 “pre-hash” ML-DSA 또는 HashML-DSA와는 다릅니다.

ML-DSA 및 EXTERNAL_MU 메시지 유형 사용에 대한 자세한 내용은 ML-DSA 키 사양 페이지를 참조하세요.

ML-DSA와 EXTERNAL_MU 메시지 유형을 사용하는 예시는 ML-DSA 키 페어를 사용한 오프라인 검증 페이지를 참조하세요.

의 다중 리전 키 AWS KMS

AWS KMS 는 여러 리전에 동일한 키가 있는 것처럼 서로 교환하여 사용할 수 AWS 리전 있는 다른의 다중 리전 키를 지원합니다. AWS KMS keys 각 관련 다중 리전 키 세트는 키 구성 요소와 키 ID가 동일하므로 다시 암호화하거나 리전 간 호출을 수행하지 AWS 리전 않고도 한에서 데이터를 암호화 AWS 리전 하고 다른에서 복호화할 수 있습니다 AWS KMS.

모든 KMS 키와 마찬가지로 다중 리전 키는 암호화 AWS KMS 되지 않은 상태로 두지 않습니다. 암호화 또는 서명을 위한 대칭 또는 비대칭 다중 리전 키를 생성하고 HMAC 태그를 생성 및 확인하기 위한 HMAC 다중 리전 키를 생성하고 가져온 키 구성 요소 또는 AWS KMS 가 생성한 키 구성 요소가 있는 다중 리전 키를 생성할 수 있습니다. 별칭 및 태그 생성, 키 정책 및 권한 부여 설정, 선택적으로 활성화 및 비활성화를 포함하여 독립적으로 각 다중 리전 키를 관리해야 합니다. 단일 리전 키로 수행할 수 있는 모든 암호화 작업에서 다중 리전 키를 사용할 수 있습니다.

다중 리전 키는 많은 일반적인 데이터 보안 시나리오를 위한 유연하고 강력한 솔루션입니다.

재해 복구

백업 및 복구 아키텍처에서 다중 리전 키를 사용하면 AWS 리전 중단 발생 시에도 중단 없이 암호화된 데이터를 처리할 수 있습니다. 백업 리전에서 유지 관리되는 데이터는 백업 리전에서 해독할 수 있으며, 백업 리전에서 새로 암호화된 데이터는 해당 리전이 복원될 때 기본 리전에서 해독할 수 있습니다.

글로벌 데이터 관리

전 세계적으로 운영되는 비즈니스에는 AWS 리전에서 일관되게 사용할 수 있는 전 세계적으로 분산된 데이터가 필요합니다. 데이터가 상주하는 모든 리전에서 다중 리전 키를 만든 다음 교차 리전 호출의 지연 시간 또는 각 리전의 다른 키로 데이터를 다시 암호화하는 비용 없이 단일 리전 키인 것처럼 키를 사용할 수 있습니다.

분산 서명 애플리케이션

교차 리전 서명 기능이 필요한 애플리케이션은 다중 리전 비대칭 서명 키를 사용하여 다른 AWS 리전에서 일관되고 반복적으로 동일한 디지털 서명을 생성할 수 있습니다.

단일 글로벌 신뢰 스토어(단일 루트 CA(인증 기관) 및 루트 CA에서 서명한 리전의 중간 CA에 대해 인증서 체인을 사용하는 경우 다중 리전 키가 필요하지 않습니다. 그러나 시스템에서 애플리케이션 서명과 같은 중간 CA를 지원하지 않는 경우 다중 리전 키를 사용하여 리전 인증에 일관성을 유지할 수 있습니다.

여러 리전에 걸친 활성/활성 애플리케이션

일부 워크로드 및 애플리케이션은 여러 리전에 걸쳐 활성/활성 아키텍처에 있을 수 있습니다. 이러한 애플리케이션의 경우 다중 리전 키는 리전 경계를 넘어 이동할 수 있는 데이터에 대한 동시 암호화 및 해독 작업에 동일한 키 구성 요소를 제공하여 복잡성을 줄일 수 있습니다.

클라이언트 측 암호화 라이브러리(예: AWS Encryption SDK, AWS Database Encryption SDK 및 Amazon S3 클라이언트 측 암호화)와 함께 다중 리전 키를 사용할 수 있습니다

유휴 시 암호화 또는 디지털 서명을 위해 AWS 와 통합되는 대부분의 서비스는 AWS KMS 현재 다중 리전 키를 단일 리전 키인 것처럼 취급합니다. 예를 들어 Amazon S3 리전 간 복제는 두 리전의 KMS 키가 관련 다중 리전 키인 경우에도 대상 리전의 KMS 키로 객체 데이터를 암호화하는 데 사용되는 데이터 키를 복호화하고 다시 암호화합니다. 서비스가 암호화된 데이터를 복제하는 방법과 다중 리전 키를 다르게 처리하는지 여부를 이해하려면 서비스별 설명서를 참조하세요.

다중 리전 키는 글로벌 키가 아닙니다. 다중 리전 기본 키를 만든 다음 AWS 파티션 내에서 선택한 리전에 복제합니다. 그런 다음 각 리전에서 다중 리전 키를 독립적으로 관리합니다. AWS 는 AWS KMS 사용자를 대신하여 다중 리전 키를 자동으로 생성하거나 복제하지 않습니다. AWS 서비스AWS 관리형 키에서 계정에서 생성하는 KMS 키인는 항상 단일 리전 키입니다.

중국 리전에서는 다중 리전 키 기능을 사용해 중국 리전 파티션(aws-cn) 내에서 KMS 키를 복제할 수 있습니다. 예를 들어 중국(베이징) 리전에서 중국(닝샤) 리전으로, 또는 그 반대로 키를 복제할 수 있습니다. 한 중국 리전에서 다른 중국 리전으로 키를 복제하면 대상 리전 AWS Key Management Service 의를 사용하고 대상 리전에 적용되는 모든 계약 조건을 준수하는 데 동의하는 것입니다. 베이징 및 닝샤 리전에서 중국 리전 파티션 외부 AWS 의 리전으로 키를 복제할 수 없습니다. 마찬가지로 중국 리전 파티션 외부의 리전에서 베이징 또는 닝샤 리전으로 키를 복제할 수 없습니다.

기존 단일 리전 키는 다중 리전 키로 변환할 수 없습니다. 이러한 설계는 기존 단일 리전 키로 보호되는 모든 데이터가 동일한 데이터 상주 및 데이터 주권 속성을 유지하도록 보장합니다.

대부분의 데이터 보안 요구 사항에서 리전 리소스의 리전 격리 및 내결함성으로 인해 표준 AWS KMS 단일 리전 키가 가장 적합한 솔루션입니다. 그러나 여러 리전의 클라이언트 측 애플리케이션에서 데이터를 암호화하거나 서명해야 하는 경우 다중 리전 키가 해결책일 수 있습니다.

리전

다중 리전 키는가 지원하는 모든에서 지원 AWS 리전 됩니다 AWS KMS .

요금 및 할당량

관련된 다중 리전 키 세트의 모든 키는 가격 책정 및 할당량에 대해 하나의 KMS 키로 계산됩니다. AWS KMS 할당량은 계정의 각 리전에 대해 별도로 계산됩니다. 각 리전에서 다중 리전 키의 사용 및 관리는 해당 리전의 할당량에 포함됩니다.

지원되는 KMS 키 유형

다음 유형의 멀티 리전 KMS 키를 만들 수 있습니다.

- 대칭 암호화 KMS 키
- 비대칭 KMS 키
- HMAC KMS 키
- 가져온 키 구성 요소가 있는 KMS 키

사용자 지정 키 스토어에서는 다중 리전 키를 만들 수 없습니다.

자세히 알아보기

- 다중 리전 KMS 키에 대한 액세스를 제어하는 방법을 알아보려면 다중 리전 키에 대한 액세스 제어 단원을 참조하세요.
- 모든 유형의 다중 리전 기본 KMS 키를 생성하려면 다중 리전 프라이머리 키 생성 단원을 참조하세요.
- 다중 리전 복제본 KMS 키를 생성하려면 다중 리전 복제본 키 생성 단원을 참조하세요.
- 기본 리전을 업데이트하려면 다중 리전 키 세트에서 프라이머리 키 변경 단원을 참조하세요.
- 다중 리전 KMS 키를 식별하고 보려면 HMAC KMS 키 식별 단원을 참조하세요.
- 다중 리전 KMS 키 삭제에 대한 특별 고려 사항에 대한 자세한 내용은 Deleting multi-Region keys 단원을 참조하세요.

용어 및 개념

다중 리전 키에 사용되는 용어와 개념은 다음과 같습니다.

다중 리전 키

다중 리전 키는 다른 AWS 리전에서 동일한 키 ID와 키 구성 요소(및 다른공유 속성)를 가진 KMS 키 집합 중 하나입니다 각 다중 리전 키는 완벽하게 작동하는 KMS 키이며 관련 다중 리전 키와 완전히 독립적으로 사용할 수 있습니다. 모든 관련 다중 리전 키는 키 ID와 키 구성 요소가 동일하므로 상호 운용 가능합니다. 즉, 모든의 모든 관련 다중 리전 키는 다른 모든 관련 다중 리전 키로 암호화된 사이퍼텍스트를 해독 AWS 리전 할 수 있습니다.

KMS 키를 생성할 때 다중 리전 속성을 설정합니다. 기존 키에서는 다중 리전 속성을 변경할 수 없습니다. 단일 리전 키를 다중 리전 키로 변환하거나 다중 리전 키를 단일 리전 키로 변환할 수 없습니다. 기존 워크로드를 다중 리전 시나리오로 이동하려면 데이터를 다시 암호화하거나 새로운 다중 리전 키로 새 서명을 만들어야 합니다.

다중 리전 키는 대칭 또는 비대칭일 수 있으며 AWS KMS 키 구성 요소 또는 가져온 키 구성 요소를 사용할 수 있습니다. 사용자 지정 키 스토어에서는 다중 리전 키를 만들 수 없습니다.

관련된 다중 리전 키 세트에는 항상 정확히 하나의 기본 키가 있습니다. 다른 AWS 리전에서 해당 기본 키의 복제본 키를 생성할 수 있습니다. 기본 리전을 업데이트할 수도 있습니다. 즉, 기본 키를 복제본 키로 변경하고 지정된 복제본 키를 기본 키로 변경하는 것입니다. 그러나 각각에 기본 키 또는 복제본 키를 하나만 유지할 수 있습니다 AWS 리전. 모든 리전은 동일한 AWS 파티션에 있어야 합니다.

동일 또는 다른 AWS 리전에 여러 관련 다중 리전 키 세트가 있을 수 있습니다. 관련된 다중 리전 키는 상호 운용이 가능하지만 관련되지 않은 다중 리전 키는 상호 운용할 수 없습니다.

프라이머리 키

다중 리전 기본 키는 동일한 파티션 AWS 리전 의 다른에 복제할 수 있는 KMS 키입니다. 각 다중 리전 키 세트에는 기본 키가 하나만 있습니다.

기본 키와 복제본 키는 차이점은 다음과 같습니다.

- 기본 키만 복제될 수 있습니다.
- 기본 키는 키 구성 요소 및 키 ID를 포함하여 복제 키의 공유 속성에 대한 소스입니다.
- 기본 키에서만 자동 키 교체를 활성화, 비활성화할 수 있습니다.
- 언제든지 기본 키의 삭제를 예약할 수 있습니다. 그러나 AWS KMS 는 모든 복제본 키가 삭제될 때까지 기본 키를 삭제하지 않습니다.

기본 키와 복제본 키는 암호화 속성에서 다르지 않습니다. 기본 키와 해당 복제 키를 서로 바꿔서 사용할 수 있습니다.

따라서 기본 키를 복제할 필요는 없습니다. KMS 키와 마찬가지로 사용할 수 있으며 유용할 때 복제할 수 있습니다. 그러나 다중 리전 키는 단일 리전 키와 보안 속성이 다르기 때문에 다중 리전 키를 복제하려는 경우에만 다중 리전 키를 만드는 것이 좋습니다.

복제본 키

다중 리전 복제본 키는 프라이머리 키 및 관련 복제본 키와 키 ID 및 키 구성 요소가 동일하지만 다른 AWS 리전에 존재하는 KMS 키입니다.

복제 키는 자체 키 정책, 권한 부여, 별칭, 태그 및 기타 속성을 포함하는 완전한 기능을 갖춘 KMS 키입니다. 기본 키 또는 다른 키의 복사본이나 포인터가 아닙니다. 기본 키 및 모든 관련 복제본 키가 비활성화되어 있어도 복제본 키를 사용할 수 있습니다. 복제본 키를 기본 키로, 기본 키를 복제본 키로 변환할 수도 있습니다. 복제 키는 일단 생성되면 키 교체 및 기본 리전 업데이트에 대해서만 기본 키에 의존합니다.

기본 키와 복제 키는 암호화 속성에서 다르지 않습니다. 기본 키와 해당 복제 키를 서로 바꿔서 사용할 수 있습니다. 기본 키 또는 복제본 키로 암호화된 데이터는 동일한 키 또는 관련된 기본 키 또는 복제본 키로 해독할 수 있습니다.

복제

다중 리전 기본 키를 동일한 파티션 AWS 리전 의 다른에 복제할 수 있습니다. 이렇게 하면는 기본 키와 동일한 키 ID 및 기타 공유 속성을 사용하여 지정된 리전에서 다중 리전 복제본 키를 AWS KMS 생성합

니다. ??? ??? AWS_KMS 오리진이 있는 KMS 키의 경우는 키 구성 요소를 리전 경계를 통해 AWS KMS 안전하게 전송하고 내부에서 새 복제본 키와 연결합니다 AWS KMS. EXTERNAL 오리진이 있는 KMS 키의 경우 기본 리전 키로 가져온 것과 동일한 키 구성 요소를 개별적으로 리치 복제본 리전 키로 가져와야 합니다.

공유 속성

공유 속성은 복제본 키와 공유되는 다중 리전 기본 키의 속성입니다.는 기본 키와 동일한 공유 속성 값을 가진 복제본 키를 AWS KMS 생성합니다. 그런 다음 기본 키의 공유 속성 값을 복제 키와 주기적으로 동기화합니다. 복제본 키에는 이러한 속성을 설정할 수 없습니다.

다음은 다중 리전 키의 공유 속성입니다.

- 키 ID — (키 ARN의 Region 요소가 다릅니다.)
- 키 구성 요소 - 관련 다중 리전 키 세트의 기본 키와 복제본 키는 동일한 키 구성 요소를 공유합니다. 키 구성 요소가 AWS KMS (AWS_KMS 오리진)에 의해 생성되는 다중 리전 키의 경우는 복제본이 생성되거나 자동 또는 온디맨드 교체를 통해 새 키 구성 요소가 생성될 때 모든 키 구성 요소를 기본 복제본에서 각 복제본으로 AWS KMS 안전하게 전송합니다. 가져온 키 구성 요소(EXTERNAL 오리진)가 있는 다중 리전 키의 경우는 기본 키에서 키 구성 요소 식별자를 AWS KMS 동기화하지만 각 복제본 키로 키 구성 요소를 독립적으로 가져와야 합니다.
- 키 구성 요소 오리진
- 키 사양 및 암호화 알고리즘
- 키 사용
- 자동 키 교체 — 기본 키에서만 자동 키 교체를 활성화하거나 비활성화할 수 있습니다. 공유 키 구성 요소의 모든 버전을 사용하여 새 복제본 키가 생성됩니다. 자세한 내용은 Rotating multi-Region keys을 참조하세요.
- 온디맨드 교체 - 프라이머리 키에서만 온디맨드 교체를 수행할 수 있습니다. 키 구성 요소가 AWS KMS (AWS_KMS 오리진)에 의해 생성되는 다중 리전 키의 경우는 공유 키 구성 요소의 모든 버전으로 복제본 키를 AWS KMS 생성합니다. 가져온 키 구성 요소(EXTERNAL 오리진)가 있는 다중 리전 키의 경우는 키 구성 요소 ID와 키 구성 요소 설명을 기본 키에서 복제본 키로 AWS KMS 전파하지만 키 구성 요소는 전파하지 않습니다. 올바른 키 구성 요소를 각 복제본 키로 개별적으로 가져와야 합니다. 자세한 내용은 Rotating multi-Region keys을 참조하세요.

또한 관련 다중 리전 키의 기본 및 복제 지정을 공유 속성으로 생각할 수 있습니다. 새 복제본 키를 생성하거나 기본 키를 업데이트하면는 모든 관련 다중 리전 키에 대한 변경 사항을 AWS KMS 동기화합니다. 이러한 변경이 완료되면 관련된 모든 다중 리전 키가 기본 키와 복제 키를 정확하게 나열합니다.

다중 리전 키의 다른 모든 속성은 키 설명, 키 정책, 권한 부여, 활성화 및 비활성화된 키 상태, 별칭 및 태그를 포함한 독립적인 속성입니다. 관련된 모든 다중 리전 키에서 이러한 속성에 대해 동일한 값을 설정할 수 있지만 독립 속성의 값을 변경하면 AWS KMS 가 동기화하지 않습니다.

다중 리전 키의 공유 속성 동기화를 추적할 수 있습니다. AWS CloudTrail 로그에서 SynchronizeMultiRegionKey 이벤트를 찾습니다.

다중 리전 키에 대한 보안 고려 사항

AWS KMS 다중 리전 키는 필요한 경우에만 사용합니다. 다중 리전 키는 AWS 리전 간에 암호화된 데이터를 이동하거나 지역 간 액세스가 필요한 워크로드에 유연하고 확장 가능한 솔루션을 제공합니다. 여러 리전에서 보호된 데이터를 공유, 이동 또는 백업해야 하거나 다른 리전에서 작동하는 애플리케이션의 동일한 디지털 서명을 만들어야 하는 경우 다중 리전 키를 고려하십시오.

그러나 다중 지역 키를 만드는 프로세스는 AWS KMS내의 AWS 리전 경계를 넘어 키 구송 요소를 이동합니다. 다중 리전 키에 의해 생성된 암호문은 여러 지리적 위치에 있는 여러 관련 키에 의해 해독될 수 있습니다. 또한 지역적으로 격리된 서비스 및 리소스에도 상당한 이점이 있습니다. 각 AWS 리전 은 격리되어 있으며 다른 리전에 독립적입니다. 리전에서는 내결함성, 안정성 및 복원성을 지원하고 지연 시간을 줄일 수도 있습니다. 이를 통해 사용자는 가용 상태를 유지하며 다른 리전에서 중단되는 경우 영향을 받지 않는 중복 리소스를 생성할 수 있습니다. AWS KMS또한 에서는 모든 사이퍼텍스트를 하나의 키로만 복호화할 수 있습니다.

또한 리전 지역 키는 새로운 보안 고려 사항을 제기합니다.

- 다중 리전 키를 사용하면 액세스 제어 및 데이터 보안 정책 적용이 더욱 복잡해집니다. 격리된 여러 리전의 키에 대해 정책을 일관되게 감사해야 합니다. 또한 별도의 키에 의존하는 대신 정책을 사용하여 경계를 적용해야합니다.

  예를 들어, 한 리전의 급여 팀이 다른 지역의 급여 데이터를 읽을 수 없도록 데이터에 대한 정책 조건을 설정해야 합니다. 또한 한 리전의 다중 리전 키가 한 테넌트의 데이터를 보호하고 다른 리전의 관련 다중 리전 키가 다른 테넌트의 데이터를 보호하는 시나리오를 방지하려면 액세스 제어를 사용해야 합니다.
- 리전 간 키 감사 또한 더 복잡합니다. 다중 리전 키를 사용하면 여러 리전의 감사 활동을 검사하고 조정하여 보호된 데이터에 대한 주요 활동을 완벽하게 이해할 수 있어야 합니다.
- 데이터 상주 규정 준수는 더욱 복잡할 수 있습니다. 격리된 리전을 사용하면 데이터 상주 및 데이터 주권 준수를 보장할 수 있습니다. 지정된 리전의 KMS 키는 해당 리전의 민감한 데이터만 해독할 수 있습니다. 한 리전에서 암호화된 데이터는 완전히 보호되고 다른 리전에서는 액세스할 수 없게 됩니다.

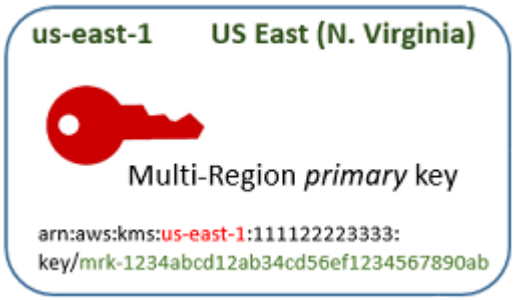
다중 리전 키를 사용하여 데이터 레지던시 및 데이터 주권을 확인하려면 액세스 정책을 구현하고 여러 리전에서 AWS CloudTrail 이벤트를 컴파일해야 합니다.

다중 리전 키에 대한 액세스 제어를 더 쉽게 관리할 수 있도록 다중 리전 키 복제 권한(kms:ReplicateKey) 복제 권한은 키 생성을 위한 표준 권한(kms:CreateKey)과 별개입니다. 또한는 다중 리전 키를 생성, 사용 또는 관리할 수 있는 권한을 허용하거나 거부`kms:MultiRegion`하는와 다중 리전 키를 복제할 수 있는 리전을 제한`kms:ReplicaRegion`하는를 포함하여 다중 리전 키에 대한 여러 정책 조건을 AWS KMS 지원합니다. 자세한 내용은 다중 리전 키에 대한 액세스 제어을 참조하세요.

다중 리전 키의 작동 방식

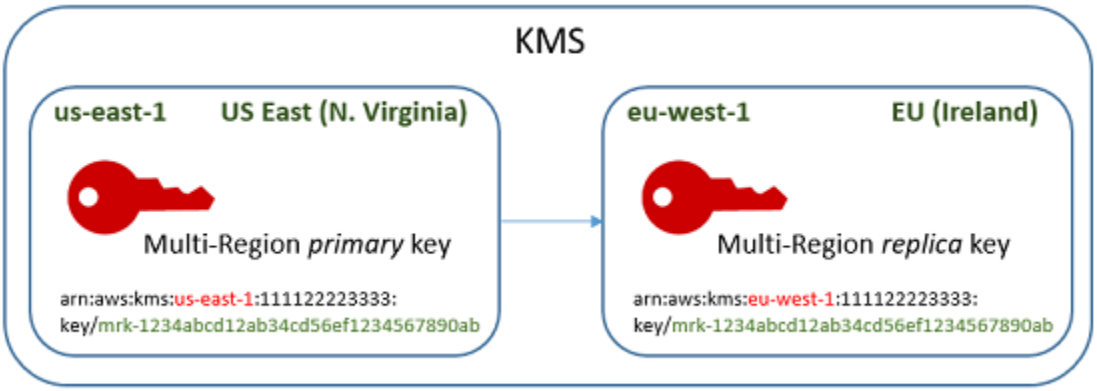
먼저 미국 동부(버지니아 북부)와 같이가 AWS 리전 AWS KMS 지원하는에서 대칭 또는 비대칭 다중 리전 기본 키를 생성합니다. 키를 만들 때만 키가 단일 리전인지 또는 다중 리전인지를 결정합니다. 나중에 이 속성을 변경할 수 없습니다. 다른 KMS 키와 마찬가지로 다중 리전 키에 대한 키 정책을 설정하고 권한 부여를 만들고 분류 및 권한 부여를 위한 별칭 및 태그를 추가할 수 있습니다. (이들은 다른 키와 공유되거나 동기화되지 않는 독립적인 속성입니다.) 암호화 또는 서명에 대한 암호화 작업에 다중 리전 기본 키를 사용할 수 있습니다.

AWS KMS 콘솔에서 또는 `MultiRegion` 파라미터가 로 설정된 CreateKey API를 사용하여 다중 리전 기본 키를 생성할 수 있습니다`true`. 다중 리전 키에는 `mrk-`로 시작하는 구분 키 ID가 있습니다. `mrk-` 접두사를 사용하여 프로그래밍 방식으로 MRK를 식별할 수 있습니다.



선택한 경우 다중 리전 기본 키를 유럽(아일랜드)과 같은 동일한 AWS 파티션 AWS 리전 에서 하나 이상의 다른에 복제할 수 있습니다. 이렇게 하면가 기본 키와 동일한 키 ID 및 기타 공유 속성을 사용하여 지정된 리전에서 복제본 키를 AWS KMS 생성합니다. ??? 그 결과 두 개의 관련 다중 리전 키(기본 키와 복제본 키)가 생성되며, 이 키를 서로 바꿔 사용할 수 있습니다..

AWS KMS 콘솔에서 또는 ReplicateKey API를 사용하여 다중 리전 복제본 키를 생성할 수 있습니다.



결과 다중 리전 복제본 키는 기본 키와 동일한 공유 속성을 가진 완전한 기능의 KMS 키입니다. 다른 모든 측면에서 자체 설명, 키 정책, 권한 부여, 별칭 및 태그가 있는 독립 KMS 키입니다. 다중 리전 키를 사용하거나 사용하지 않도록 설정해도 관련된 다중 리전 키에는 영향을 주지 않습니다. 기본 키와 복제 키를 암호화 작업에서 독립적으로 사용하거나 사용을 조정할 수 있습니다. 예를 들어 미국 동부(버지니아 북부) 리전의 기본 키를 사용하여 데이터를 암호화하고, 데이터를 유럽(아일랜드) 리전으로 이동하고, 복제 키를 사용하여 데이터를 해독할 수 있습니다.

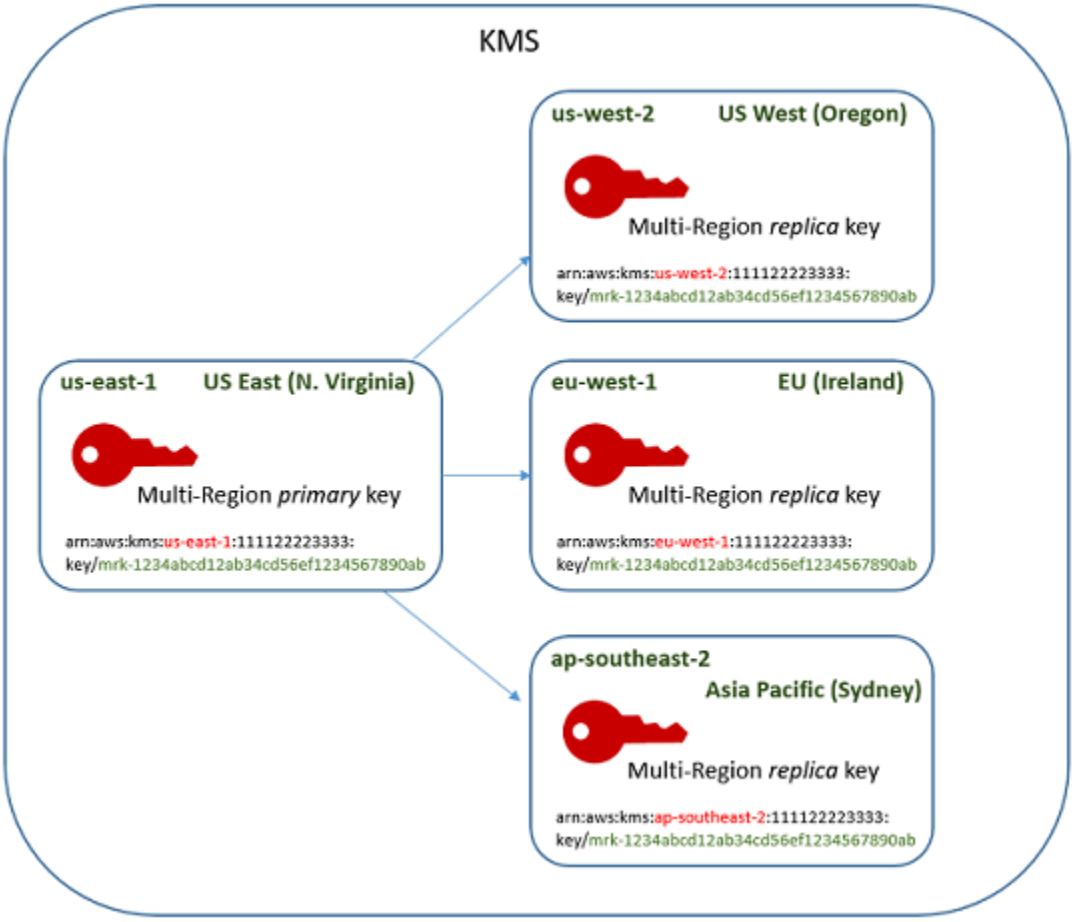
관련 다중 리전 키는 동일한 키 ID를 갖습니다. 키 ARN(Amazon 리소스 이름)은 리전 필드에서만 다릅니다. 예를 들어 다중 리전 기본 키 및 복제본 키에는 다음 예제 키 ARN이 있을 수 있습니다. 키 ID(키 ARN의 마지막 요소)는 동일합니다. 두 키 모두 다중 리전 키의 구분적 키 ID를 가지며, mrk-로 시작합니다.

```
Primary key: arn:aws:kms:us-
east-1:111122223333:key/mrk-1234abcd12ab34cd56ef12345678990ab
Replica key: arn:aws:kms:eu-
west-1:111122223333:key/mrk-1234abcd12ab34cd56ef12345678990ab
```

상호 운용성을 위해서는 동일한 키 ID가 필요합니다. 암호화할 때는 KMS 키의 키 ID를 사이퍼텍스트로 AWS KMS 바인딩하므로 사이퍼텍스트는 해당 KMS 키 또는 키 ID가 동일한 KMS 키로만 복호화할 수 있습니다. 또한 이 기능을 사용하면 관련 다중 리전 키를 쉽게 인식할 수 있으며 상호 교환 방식으로 쉽게 사용할 수 있습니다. 예를 들어 애플리케이션에서 이러한 키를 사용하는 경우 공유 키 ID로 관련 다중 리전 키를 참조할 수 있습니다. 그런 다음 필요한 경우 리전 또는 ARN을 지정하여 구분합니다.

데이터가 변경되면 기본 키를 미국 서부(오레곤) 및 아시아 태평양(시드니)과 같은 동일한 파티션 AWS 리전 의 다른에 복제할 수 있습니다. 결과는 다음 다이어그램과 같이 동일한 키 구성 요소 및 키 ID를 가진 4개의 관련 다중 리전 키입니다. 키를 독립적으로 관리합니다. 가져온 키 구성 요소가 있는 다중 리전 키의 경우 키 구성 요소를 각 관련 키로 개별적으로 가져와야 합니다. 독립적으로 또는 조정된 방식으로 사용할 수 있습니다. 예를 들어 아시아 태평양(시드니)의 복제본 키를 사용하여 데이터를 암호화

하고, 데이터를 미국 서부(오레곤)로 이동한 다음, 미국 서부(오레곤)의 복제 키를 사용하여 암호를 해독할 수 있습니다.



다중 리전 키에 대한 기타 고려 사항은 다음과 같습니다.

공유 속성 동기화 - 다중 리전 키의 공유 속성이 변경되면는 기본 키에서 모든 복제본 키로 변경 사항을 AWS KMS 자동으로 동기화합니다. 공유 속성의 동기화를 요청하거나 강제할 수 없습니다.는 모든 변경 사항을 AWS KMS 감지하고 동기화합니다. 그러나 CloudTrail 로그에서 SynchronizeMultiRegionKey 이벤트를 사용하여 동기화를 감사할 수 있습니다.

예를 들어 오리진이 있는 대칭 다중 리전 기본 키에서 자동 키 교체를 활성화하면는 해당 설정을 모든 복제본 키로 AWS_KMS AWS KMS 복사합니다. 키 구성 요소가 교체되면 모든 관련 다중 리전 키 간에 교체가 동기화되므로 동일한 현재 키 구성 요소를 가지며 모든 이전 버전의 키 구성 요소에 액세스할 수 있습니다. 새 복제 키를 만들면 관련된 모든 다중 리전 키와 동일한 현재 키 구성 요소를 가지며 이전 버전의 키 구성 요소에 액세스할 수 있습니다. 자세한 내용은 Rotating multi-Region keys을 참조하세요.

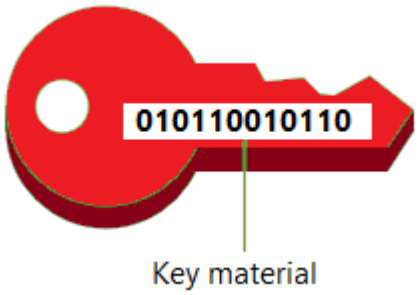
기본 키 변경 — 모든 다중 리전 키 세트에는 정확히 하나의 기본 키가 있어야 합니다. 기본 키는 복제할 수있는 유일한 키입니다. 또한 복제 키의 공유 속성의 소스이기도합니다. 하지만 기본 키를 복제본으로 변경하고 복제본 키 중 하나를 기본 키로 승격할 수 있습니다. 이를 수행하여 특정 리전에서 다중 리전 기본 키를 삭제하거나 프로젝트 관리자와 더 가까운 지역에서 기본 키를 찾을 수 있습니다. 자세한 내용은 다중 리전 키 세트에서 프라이머리 키 변경을 참조하세요.

다중 리전 키 삭제 - 모든 KMS 키와 마찬가지로에서 삭제하기 전에 다중 리전 키 AWS KMS 삭제를 예약해야 합니다. 키가 삭제 대기 중인 동안에는 어떠한 암호화 작업에도 해당 키를 사용할 수 없습니다. 그러나 AWS KMS 는 모든 복제본 키가 삭제될 때까지 다중 리전 기본 키를 삭제하지 않습니다. 자세한 내용은 Deleting multi-Region keys 섹션을 참조하십시오.

키의 AWS KMS 키 구성 요소 가져오기

제공하는 키 구성 요소를 사용하여 AWS KMS keys (KMS 키)를 생성할 수 있습니다.

KMS 키는 데이터 키를 논리적으로 표현한 것입니다. KMS 키의 메타데이터에는 암호화 작업을 수행하는 데 사용되는 키 구성 요소의 ID가 포함됩니다. KMS 키를 생성할 때 기본적으로는 해당 KMS 키의 키 구성 요소를 AWS KMS 생성합니다. 그러나 키 구성 요소 없이 KMS 키를 생성한 다음 자신의 키 구성 요소를 해당 KMS 키로 가져올 수 있습니다. 이 KMS 키는 종종 “자체 키 사용"(BYOK)이라고 합니다.



> **Note**
>
> AWS KMS 는 가져온 키 구성 요소가 있는 KMS 키로 암호화되었 AWS KMS더라도 외부에서 AWS KMS 대칭 암호화 KMS 키로 암호화된 사이퍼텍스트의 복호화를 지원하지 않습니다.는 이 작업에 필요한 사이퍼텍스트 형식을 게시하지 AWS KMS 않으며 형식이 예고 없이 변경될 수 있습니다.

가져온 키 구성 요소를 사용하는 경우가 해당 구성 요소의 사본을 사용하도록 허용하면서 키 구성 요소에 대한 책임은 계속 AWS KMS 사용자에게 있습니다. 이렇게 하기로 선택하는 경우는 다음과 같습니다.

- 요구 사항에 부합하는 엔트로피 소스를 사용하여 키 구성 요소를 생성했음을 입증하기 위해.
- AWS 서비스와 함께 자체 인프라의 키 구성 요소를 사용하고를 사용하여 AWS KMS 내에서 해당 키 구성 요소의 수명 주기를 관리합니다 AWS.
- 코드 서명용 키 AWS KMS, PKI 인증서 서명 및 인증서 고정 애플리케이션과 같이에서 기존의 잘 정립된 키를 사용하려면
- 에서 키 구성 요소의 만료 시간을 설정하고 AWS 수동으로 삭제하지만 나중에 다시 사용할 수 있도록 하려면 . 반면 키 삭제를 예약하려면 7 - 30일의 대기 기간이 필요하며, 이 기간이 지나면 삭제한 KMS 키를 복구할 수 없습니다.
- 키 구성 요소의 원본 사본을 소유하고 키 구성 요소의 전체 수명 주기 동안 추가 내구성 및 재해 복구를 AWS 위해 외부에 보관합니다.
- 비대칭 키 및 HMAC 키의 경우 가져오기는 내부 및 외부에서 작동하는 호환 가능하고 상호 운용 가능한 키를 생성합니다 AWS.

지원되는 KMS 키 유형

AWS KMS 는 다음 유형의 KMS 키에 대해 가져온 키 구성 요소를 지원합니다. 사용자 지정 키 스토어의 KMS 키에 키 구성 요소를 가져올 수 없습니다.

- 대칭 암호화 KMS 키
- 비대칭 KMS 키(ML-DSA 키 제외)
- HMAC KMS 키
- 지원되는 모든 유형의 다중 리전 키

리전

가져온 키 구성 요소는가 지원하는 모든에서 지원 AWS 리전 됩니다 AWS KMS .

중국 리전의 경우 대칭 암호화 KMS 키에 대한 주요 구성 요소 요구 사항은 다른 리전과 다릅니다. 자세한 내용은 3단계: 키 구성 요소 암호화을 참조하세요.

자세히 알아보기

- 가져온 키 구성 요소로 KMS 키를 생성하려면 가져온 키 구성 요소가 있는 KMS 키 생성 단원을 참조하세요.
- KMS 키의 가져온 키 구성 요소 만료일이 가까워지면 알려주는 알람을 만들려면 가져온 키 구성 요소의 만료 여부에 대한 CloudWatch 경보 생성 단원을 참조하세요.

- 키 구성 요소를 KMS 키로 다시 가져오려면 키 구성 요소 다시 가져오기 단원을 참조하세요.
- 새로운 키 구성 요소를 온디맨드 교체를 위해 KMS 키에 가져오려면 새 키 구성 요소 가져오기 및 온디맨드 키 교체 수행 페이지를 참조하세요.
- 가져온 키 요소가 있는 KMS 키를 식별하고 보려면 가져온 키 구성 요소로 KMS 키 식별 단원을 참조하세요.
- 가져온 키 구성 요소가 있는 KMS 키를 삭제하기 위한 특별 고려 사항에 대해 알아보려면 Deleting KMS keys with imported key material 단원을 참조하세요.

가져온 키 구성 요소에 대한 특별 고려 사항

키 구성 요소를 로 가져오기 전에 가져온 키 구성 요소의 다음 특성을 이해해야 AWS KMS합니다.

키 구성 요소를 생성합니다.

보안 요구 사항에 부합하는 임의 소스를 이용해 키 구성 요소를 생성해야 합니다.

가용성과 지속성에 대한 책임은 고객에게 있습니다.

AWS KMS 는 가져온 키 구성 요소를 고가용성으로 유지하도록 설계되었습니다. 그러나 AWS KMS 는 가져온 키 구성 요소의 내구성을가 AWS KMS 생성하는 키 구성 요소와 동일한 수준으로 유지하지 않습니다. 자세한 내용은 가져온 키 구성 요소 보호을 참조하세요.

키 구성 요소를 삭제할 수 있음

KMS 키에서 가져온 키 구성 요소를 삭제하여 즉시 KMS 키를 사용할 수 없게 만들 수 있습니다. 또한 키 구성 요소를 KMS 키로 가져올 때 키의 만료 여부를 결정하고 만료 시간을 설정할 수 있습니다. 만료 시간이 되면가 AWS KMS 키 구성 요소를 삭제합니다. 키 구성 요소가 없으면 어떠한 암호화 작업에도 KMS 키를 사용할 수 없습니다. 키를 복원하려면 같은 키 구성 요소를 다시 가져와야 합니다.

비대칭 및 HMAC 키의 키 구성 요소는 변경할 수 없습니다.

KMS 키로 키 구성 요소를 가져오면, KMS 키는 키 구성 요소와 영구적으로 연결됩니다. 동일한 키 구성 요소를 가져오되, 다른 키 구성 요소를 KMS 키로 가져올 수는 없습니다. 또한 가져온 키 구성 요소가 있ㄴ느 KMS 키에 대한 자동 키 교체를 활성화할 수 없습니다. 하지만 가져온 키 구성 요소가 있는 KMS 키를 수동 교체할 수 있습니다.

대칭 암호화 키에 대해 온디맨드 교체를 수행할 수 있습니다.

가져온 키 구성 요소가 있는 대칭 암호화 키는 온디맨드 교체를 지원합니다. 여러 키 구성 요소를 이러한 키에 가져와 온디맨드 교체를 통해 현재 키 구성 요소를 업데이트할 수 있습니다. 현재 키 구성

요소는 암호화와 복호화에 모두 사용되지만, 다른(현재가 아닌) 키 구성 요소는 복호화에만 사용할 수 있습니다.

키 구성 요소 오리진을 변경할 수 없음

가져온 키 구성 요소용으로 설계된 KMS 키에는 변경할 수 없는 오리진 EXTERNAL 값이 있습니다. 가져온 키 구성 요소의 KMS 키를를 포함한 다른 소스의 키 구성 요소를 사용하도록 변환할 수 없습니다 AWS KMS. 마찬가지로 AWS KMS 키 구성 요소가 있는 KMS 키를 가져온 키 구성 요소용으로 설계된 키로 변환할 수 없습니다.

키 구성 요소를 내보낼 수 없음

가져온 키 구성 요소는 내보낼 수 없습니다. 가져온 키 구성 요소는 어떤 형식으로도 반환할 수 AWS KMS 없습니다. 가져온 키 구성 요소의 사본을 외부 AWS, 가급적이면 하드웨어 보안 모듈(HSM)과 같은 키 관리자에 보관하여 키 구성 요소를 삭제하거나 만료되면 다시 가져올 수 있도록 해야 합니다.

가져온 키 구성 요소가 있는 다중 리전 키를 생성할 수 있음

가져온 키 구성 요소가 있는 다중 리전에는 가져온 키 구성 요소가 있는 KMS 키의 특징을 가지며, AWS 리전간에 상호 운용이 가능합니다. 가져온 키 구성 요소가 있는 다중 리전 키를 생성하려면 동일한 키 구성 요소를 기본 KMS 키와 각 복제본 키로 가져와야 합니다. 다중 리전 키의 키 구성 요소 가져오기에 대한 자세한 내용은 섹션을 참조하세요새 키 구성 요소 가져오기.

비대칭 키 및 HMAC 키는 이동 가능하며 상호 운용이 가능함

외부에서 비대칭 키 구성 요소와 HMAC 키 구성 요소를 사용하여 가져온 키 구성 요소가 동일한 AWS KMS 키와 상호 작용 AWS 할 수 있습니다.

알고리즘에 사용되는 KMS 키에 불가역적으로 바인딩되는 AWS KMS 대칭 사이퍼텍스트와 달리는 암호화, 서명 및 MAC 생성에 표준 HMAC 및 비대칭 형식을 AWS KMS 사용합니다. 따라서 키는 이동이 가능하며 기존 에스크로 키 시나리오를 지원합니다.

KMS 키에 키 구성 요소를 가져온 경우 외부에서 가져온 키 구성 요소를 사용하여 다음 작업을 AWS 수행할 수 있습니다.

- HMAC 키 - 가져온 키 구성 요소가 있는 HMAC KMS 키로 생성된 HMAC 태그를 확인할 수 있습니다. 가져온 키 구성 요소와 함께 HMAC KMS 키를 사용하여 외부의 키 구성 요소에 의해 생성된 HMAC 태그를 확인할 수도 있습니다 AWS.
- 비대칭 암호화 키 - 외부에서 프라이빗 비대칭 암호화 키를 사용하여 KMS 키로 암호화된 사이퍼텍스트를 해당 퍼블릭 키로 AWS 복호화할 수 있습니다. 비대칭 KMS 키를 사용하여 외부에서 생성된 비대칭 사이퍼텍스트를 복호화할 수도 있습니다 AWS.

- 비대칭 서명 키 - 가져온 키 구성 요소와 함께 비대칭 서명 KMS 키를 사용하여 외부에서 프라이빗 서명 키로 생성된 디지털 서명을 확인할 수 있습니다 AWS. 외부에서 비대칭 퍼블릭 서명 키를 사용하여 비대칭 KMS 키로 생성된 서명을 AWS 확인할 수도 있습니다.
- 비대칭 키 계약 키 - 가져온 키 구성 요소가 있는 비대칭 키 계약 KMS 키를 사용하여 AWS외부 피어와 공유된 비밀을 도출할 수 있습니다.

동일한 키 구성 요소를 동일한 AWS 리전의 다른 KMS 키로 가져오는 경우 이러한 키도 상호 운용할 수 있습니다. 서로 다른에서 상호 운용 가능한 KMS 키를 생성하려면 가져온 키 구성 요소가 있는 다중 리전 키를 AWS 리전생성합니다.

RSA 프라이빗 키

- AWS KMS 에서는 가져온 RSA 프라이빗 키에 FIPS 186-5, 섹션 A. 1.3에 설명된 테스트를 준수하는 주요 요소가 있어야 합니다. 다른 소프트웨어 또는 장치는 RSA 프라이빗 키의 이러한 소인수 검증에 대해 다른 알고리즘을 사용할 수 있습니다. 드문 경우지만 다른 알고리즘으로 검증된 키가 AWS KMS에서 허용되지 않을 수 있습니다.

대칭 암호화 키는 이동하거나 상호 운용할 수 없음

가 AWS KMS 생성하는 대칭 사이퍼텍스트는 이동하거나 상호 운용할 수 없습니다.는 이동성에 필요한 대칭 사이퍼텍스트 형식을 게시하지 AWS KMS 않으며, 형식은 예고 없이 변경될 수 있습니다.

- AWS KMS 는 가져온 키 구성 요소를 사용 AWS하더라도 외부에서 암호화하는 대칭 사이퍼텍스트를 해독할 수 없습니다.
- AWS KMS 는 사이퍼텍스트가 가져온 키 구성 요소가 있는 KMS 키로 암호화되었 AWS KMS더라도 외부의 AWS KMS 대칭 사이퍼텍스트 복호화를 지원하지 않습니다.
- 가져온 키 구성 요소가 동일한 KMS 키는 상호 운용이 불가능합니다. 각 KMS 키에 고유한 사이퍼텍스트를 AWS KMS 생성하는 대칭 사이퍼텍스트입니다. 이 사이퍼텍스트 형식은 데이터를 암호화한 KMS 키만 해독할 수 있도록 보장합니다.

또한 AWS Encryption SDK 또는 Amazon S3 클라이언트 측 암호화와 같은 AWS 도구를 사용하여 대칭 사이퍼텍스트를 AWS KMS 해독할 수 없습니다.

따라서 가져온 키 구성 요소가 있는 키를 사용하여 키 구성 요소에 조건부로 액세스할 수 있는 권한이 있는 타사가 외부의 특정 사이퍼텍스트를 복호화할 수 있는 키 에스크로 배열을 지원할 수 없습니다 AWS KMS. 키 에스크로를 지원하려면 AWS Encryption SDK를 사용하여 AWS KMS와 독립적인 키로 메시지를 암호화합니다.

가져온 키 구성 요소 보호

가져온 키 구성 요소는 전송 중 및 보관 중에 보호됩니다. 키 구성 요소를 가져오기 전에 FIPS 140-3 암호화 모듈 검증 프로그램에 따라 검증된 AWS KMS 하드웨어 보안 모듈(HSMs)합니다. 래핑 퍼블릭 키를 사용하여 키 구성 요소를 직접 암호화하거나 AES 대칭 키를 사용하여 키 구성 요소를 암호화한 다음 RSA 퍼블릭 키를 사용하여 AES 대칭 키를 암호화할 수 있습니다.

수신 시는 AWS KMS HSM의 해당 프라이빗 키로 키 구성 요소를 AWS KMS 해독하고 HSM의 휘발성 메모리에만 있는 AES 대칭 키로 다시 암호화합니다. 키 구성 요소는 HSM을 일반 텍스트로 두지 않습니다. 사용 중이고 AWS KMS HSMs.

가져온 키 구성 요소와 함께 KMS 키를 사용하는 것은 KMS 키에 설정한 액세스 제어 정책에 의해서만 결정됩니다. 또한 별칭과 태그를 사용하여 KMS 키에 대한 액세스를 식별하고 제어할 수 있습니다. AWS CloudTrail과 같은 서비스를 사용하여 키를 활성화 및 비활성화하고 확인하며 모니터링할 수 있습니다.

하지만 키 구성 요소의 유일한 안전 장치 복사본은 유지해야 합니다. 이러한 추가 제어 조치의 대가로 가져온 키 구성 요소의 내구성과 전체 가용성에 대한 책임은 사용자에게 있습니다. AWS KMS 는 가져온 키 구성 요소를 고가용성을 유지하도록 설계되었습니다. 그러나 AWS KMS 는 가져온 키 구성 요소의 내구성을가 AWS KMS 생성하는 키 구성 요소와 동일한 수준으로 유지하지 않습니다.

내구성의 이러한 차이는 다음과 같은 경우에 의미가 있습니다.

- 가져온 키 구성 요소의 만료 시간을 설정하면는 키 구성 요소가 만료된 후 AWS KMS 삭제합니다.는 KMS 키 또는 메타데이터를 삭제하지 AWS KMS 않습니다. 가져온 키 구성 요소의 만료일이 가까워지면 이를 알리는 Amazon CloudWatch 경보를 생성할 수 있습니다.

  가 KMS 키에 대해 AWS KMS 생성하는 키 구성 요소를 삭제할 수 없으며 AWS KMS 키 구성 요소가 만료되도록 설정할 수 없습니다.
- 가져온 키 구성 요소를 수동으로 삭제하면가 키 구성 요소를 AWS KMS 삭제하지만 KMS 키 또는 메타데이터는 삭제하지 않습니다. 반대로 키 삭제를 예약하려면 7~30일의 대기 기간이 필요하며, 그 후에는가 KMS 키, 메타데이터 및 키 구성 요소를 AWS KMS 영구적으로 삭제합니다.
- 드물지만에 영향을 미치는 특정 리전 전체의 장애 AWS KMS (예: 전체 전원 손실)가 발생하는 경우는 가져온 키 구성 요소를 자동으로 복원 AWS KMS 할 수 없습니다. 그러나 AWS KMS 는 KMS 키와 메타데이터를 복원할 수 있습니다.

사용자가 제어하는 시스템의 외부 AWS 에 가져온 키 구성 요소의 사본을 보관해야 합니다. 하드웨어 보안 모듈(HSM)과 같은 키 관리 시스템에 가져온 키 구성 요소의 내보내기 가능한 복사본을 저장하는 것이 좋습니다. 가장 좋은 방법은 내보낼 수 있는 키 구성 요소 사본과 함께 KMS 키 ARN과 AWS KMS

가 생성한 키 구성 요소 ID에 대한 참조를 저장하는 것입니다. 가져온 키 구성 요소가 삭제되거나 만료되면 동일한 키 구성 요소를 다시 가져올 때까지 관련 KMS 키를 사용할 수 없게 됩니다. 가져온 키 구성 요소가 영구적으로 손실되면 KMS 키로 암호화된 사이퍼텍스트를 복구할 수 없습니다.

> **Important**
>
> 대칭 암호화 키에는 여러 키 구성 요소가 연결될 수 있습니다. 이러한 키 구성 요소 중 하나를 삭제하거나 해당 키 구성 요소 중 하나가 만료되는 즉시 전체 KMS 키를 사용할 수 없게 됩니다(삭제되거나 만료되는 키 구성 요소가 `PENDING_ROTATION` 또는가 아닌 경우`PENDING_MULTI_REGION_IMPORT_AND_ROTATION`). 이러한 키와 연결된 만료되었거나 삭제된 키 구성 요소는 키가 암호화 작업에 사용 가능해지기 전에 반드시 다시 가져와야 합니다.

CloudHSM 키 저장소의 KMS 키

AWS CloudHSM 키 스토어 AWS KMS keys 에서의 삭제를 생성, 보기, 관리, 사용 및 예약할 수 있습니다. 사용하는 절차는 다른 KMS 키에 사용하는 절차와 매우 비슷합니다. 유일한 차이점은 KMS AWS CloudHSM 키를 생성할 때 키 스토어를 지정한다는 것입니다. 그런 다음 키 스토어와 연결된 클러스터의 KMS 키에 AWS CloudHSM 대해 추출할 수 없는 AWS CloudHSM 키 구성 요소를 AWS KMS 생성합니다. AWS CloudHSM 키 스토어에서 KMS 키를 사용하는 경우 암호화 작업은 클러스터의 HSMs에서 수행됩니다.

지원되는 기능

이 단원에서 설명하는 절차 외에도 AWS CloudHSM 키 스토어의 KMS 키를 사용하여 다음을 수행할 수 있습니다.

- 키 정책, IAM 정책 및 권한 부여를 사용해 KMS 키에 대한 액세스 권한을 부여합니다.
- KMS 키를 사용하거나 사용하지 않도록 설정합니다.
- 태그를 할당하고 별칭을 생성하고 ABAC(속성 기반 액세스 제어)를 사용하여 KMS 키에 대한 액세스 권한을 부여합니다.
- KMS 키를 이용하여 다음 암호화 작업을 수행합니다.
  - 암호화
  - Decrypt
  - GenerateDataKey
  - GenerateDataKeyWithoutPlaintext

- ReEncrypt

  비대칭 데이터 키 페어를 생성하는 작업인 GenerateDataKeyPair 및 GenerateDataKeyPairWithoutPlaintext는 사용자 지정 키 스토어에서 지원되지 않습니다.
- 고객 관리형 KMS 키를 지원하고 AWS KMS에 통합되는AWS 서비스와 함께 KMS 키를 사용합니다.
- AWS CloudTrail 로그와 Amazon CloudWatch 모니터링 도구에서 KMS 키 사용을 추적합니다.

지원되지 않는 기능

- AWS CloudHSM 키 스토어는 대칭 암호화 KMS 키만 지원합니다. AWS CloudHSM 키 스토어에서는 HMAC KMS 키, 비대칭 KMS 키 또는 비대칭 데이터 키 페어를 생성할 수 없습니다.
- 키 스토어의 KMS 키로 키 구성 요소를 가져올 수 AWS CloudHSM 없습니다.는 클러스터의 KMS 키에 대한 키 구성 요소를 AWS KMS 생성합니다. AWS CloudHSM
- 키 AWS CloudHSM 스토어에서 KMS 키에 대한 키 구성 요소의 자동 교체를 활성화하거나 비활성화할 수 없습니다.

키 스토어에서 KMS AWS CloudHSM 키 사용

요청에 KMS 키를 사용하는 경우 ID 또는 별칭으로 KMS 키를 식별합니다. AWS CloudHSM 키 스토어 또는 AWS CloudHSM 클러스터를 지정할 필요가 없습니다. 응답에는 모든 비대칭 암호화 KMS 키에서 반환된 동일한 필드가 포함되어 있습니다.

그러나 AWS CloudHSM 키 스토어에서 KMS 키를 사용하면 AWS CloudHSM 키 스토어와 연결된 AWS CloudHSM 클러스터 내에서 암호화 작업이 완전히 수행됩니다. 이 작업에서는 사용자가 선택한 KMS 키와 연결된 클러스터의 키 구성 요소를 사용합니다.

이것이 가능하려면 다음 조건이 충족되어야 합니다.

- KMS 키의 키 상태가 `Enabled`여야 합니다. 키 상태를 찾으려면 AWS KMS 콘솔의 Status(상태) 필드 또는 DescribeKey 응답의 `KeyState` 필드를 사용합니다.
- AWS CloudHSM 키 스토어는 AWS CloudHSM 클러스터에 연결되어야 합니다. AWS KMS 콘솔의 Status(상태) 또는 DescribeCustomKeyStores 응답의 `ConnectionState`는 `CONNECTED`여야 합니다.
- 사용자 지정 키 스토어와 연결된 AWS CloudHSM 클러스터에는 하나 이상의 활성 HSM이 포함되어야 합니다. 클러스터에서 활성 HSMs 수를 찾으려면 AWS KMS 콘솔, AWS CloudHSM 콘솔 또는 DescribeClusters 작업을 사용합니다.
- AWS CloudHSM 클러스터에는 KMS 키의 키 구성 요소가 포함되어야 합니다. 클러스터에서 키 구성 요소가 삭제된 경우나 키 구성 요소를 포함하지 않은 백업에서 HSM이 생성된 경우에는 암호화 작업이 실패합니다.

이러한 조건이 충족되지 않으면 암호화 작업이 실패하고가 `KMSInvalidStateException` 예외를 AWS KMS 반환합니다. 일반적으로 AWS CloudHSM 키 스토어를 다시 연결하기만 하면 됩니다. 추가적인 도움말은 오류가 발생하는 KMS 키를 수정하는 방법 섹션을 참조하십시오.

AWS CloudHSM 키 스토어에서 KMS 키를 사용하는 경우 각 AWS CloudHSM 키 스토어의 KMS 키는 암호화 작업에 대한 사용자 지정 키 스토어 요청 할당량을 공유한다는 점에 유의하세요. 할당량을 초과하면가를 AWS KMS 반환합니다`ThrottlingException`. AWS CloudHSM 키 스토어와 연결된 AWS CloudHSM 클러스터가 키 스토어와 관련이 없는 명령을 포함하여 많은 명령을 처리하는 경우 훨씬 더 낮은 속도로 `ThrottlingException`를 얻을 AWS CloudHSM 수 있습니다. 어떤 요청에 대해 `ThrottlingException`이 반환된 경우에는 요청 속도를 낮추고 명령을 다시 시도하십시오. 사용자 지정 키 스토어 요청 할당량에 대한 자세한 내용은 사용자 지정 키 스토어 요청 할당량 섹션을 참조하세요.

자세히 알아보기

- AWS CloudHSM 키 스토어에 대한 자세한 내용은 섹션을 참조하세요AWS CloudHSM 키 스토어.
- 키 스토어에서 KMS AWS CloudHSM 키를 생성하려면 섹션을 참조하세요키 스토어에서 KMS AWS CloudHSM 키 생성.
- 키 스토어에서 KMS AWS CloudHSM 키를 식별하고 보려면 섹션을 참조하세요키 스토어에서 KMS AWS CloudHSM 키 식별.
- 키 스토어에서 KMS 키와 AWS CloudHSM 키 구성 요소를 찾으려면 섹션을 참조하세요키 스토어에서 KMS 키 및 AWS CloudHSM 키 구성 요소 찾기.
- AWS CloudHSM 키 스토어에서 KMS 키를 삭제하기 위한 특별 고려 사항에 대해 알아보려면 AWS CloudHSM 키 스토어에서 KMS 키 삭제를 참조하세요.

외부 키 저장소의 KMS 키

외부 키 스토어에서 KMS 키를 생성, 확인, 관리, 사용 및 삭제하도록 예약하려면 다른 KMS 키에 사용하는 것과 매우 유사한 절차를 사용합니다. 그러나 외부 키 스토어에서 KMS 키를 생성할 때는 외부 키 스토어와 외부 키를 지정합니다. 외부 키 스토어에서 KMS 키를 사용하면 외부 키 관리자가 지정된 외부 키를 사용하여 암호화 및 복호화 작업을 수행합니다.

AWS KMS 는 외부 키 관리자에서 암호화 키를 생성, 보기, 업데이트 또는 삭제할 수 없습니다. AWS KMS 는 외부 키 관리자 또는 외부 키에 직접 액세스하지 않습니다. 암호화 작업에 대한 모든 요청은 외부 키 스토어 프록시에 의해 조정됩니다. 외부 키 스토어에서 KMS 키를 사용하려면 KMS 키를 호스팅하는 외부 키 스토어가 외부 키 스토어 프록시에 연결되어 있어야 합니다.

지원되는 기능

이 섹션에서 설명한 절차 외에도 외부 키 스토어에서 KMS 키를 통해 다음을 수행할 수 있습니다.

- 키 정책, IAM 정책 및 권한 부여를 사용하여 KMS 키에 대한 액세스를 제어합니다.
- KMS 키를 사용하거나 사용하지 않도록 설정합니다. 이러한 작업은 외부 키 관리자의 외부 키에 영향을 주지 않습니다.
- 태그를 할당하고 별칭을 생성하고 ABAC(속성 기반 액세스 제어)를 사용하여 KMS 키에 대한 액세스 권한을 부여합니다.
- KMS 키를 이용하여 다음 암호화 작업을 수행합니다.
  - 암호화
  - Decrypt
  - GenerateDataKey
  - GenerateDataKeyWithoutPlaintext
  - ReEncrypt

  비대칭 데이터 키 페어를 생성하는 작업인 GenerateDataKeyPair 및 GenerateDataKeyPairWithoutPlaintext는 사용자 지정 키 스토어에서 지원되지 않습니다.
- 고객 관리형 키를 지원하고 AWS KMS와 통합되는AWS 서비스와 함께 KMS 키를 사용합니다.

지원되지 않는 기능

- 외부 키 스토어는 대칭 암호화 KMS 키만 지원합니다. 외부 키 스토어에서 HMAC KMS 키 또는 비대칭 KMS 키를 생성할 수 없습니다.
- GenerateDataKeyPair와 GenerateDataKeyPairWithoutPlaintext는 외부 키 스토어의 KMS 키에서 지원되지 않습니다.
- AWS::KMS::Key CloudFormation 템플릿을 사용하여 외부 키 저장소를 생성하거나 외부 키 저장소에 KMS 키를 생성할 수 없습니다.
- 다중 리전 키는 외부 키 스토어에서 지원되지 않습니다.
- 가져온 키 구성 요소가 있는 KMS 키는 외부 키 스토어에서 지원되지 않습니다.
- 외부 키 스토어의 KMS 키에는 자동 키 교체가 지원되지 않습니다.

외부 키 저장소에서 KMS 키 사용

요청에서 KMS 키를 사용하는 경우 키 ID, 키 ARN, 별칭 또는 별칭 ARN으로 KMS 키를 식별합니다. 외부 키 스토어를 지정할 필요가 없습니다. 응답에는 모든 비대칭 암호화 KMS 키에서 반환된 동일한 필드가 포함되어 있습니다. 그러나 외부 키 스토어에서 KMS 키를 사용하면 외부 키 관리자가 KMS 키와 연결된 외부 키를 사용하여 암호화 및 복호화 작업을 수행합니다.

외부 키 스토어의 KMS 키로 암호화된 사이퍼텍스트가 최소한 표준 KMS 키로 암호화된 사이퍼텍스트만큼 안전한지 확인하기 위해는 이중 암호화를 AWS KMS 사용합니다. 데이터는 먼저 AWS KMS 키 구성 요소를 AWS KMS 사용하여에서 암호화됩니다. 그런 다음 KMS 키의 외부 키를 사용하여 외부 키 관리자에 의해 암호화됩니다. 이중 암호화된 암호화 텍스트를 복호화하기 위해 먼저 KMS 키의 외부 키를 사용하여 외부 키 관리자에 의해 사이퍼텍스트가 복호화됩니다. 그런 다음 KMS 키의 AWS KMS 키 구성 요소를 AWS KMS 사용하여에서 복호화됩니다.

이것이 가능하려면 다음 조건이 충족되어야 합니다.

- KMS 키의 키 상태가 `Enabled`여야 합니다. 키 상태를 찾으려면 AWS KMS 콘솔의 고객 관리형 키에 대한 Status(상태) 필드 또는 DescribeKey 응답의 `KeyState` 필드를 확인합니다.
- KMS 키를 호스팅하는 외부 키 스토어는 외부 키 스토어 프록시에 연결되어야 합니다. 즉, 외부 키 스토어의 연결 상태가 `CONNECTED`여야 합니다.

  연결 상태는 AWS KMS 콘솔의 외부 키 스토어 페이지 또는 DescribeCustomKeyStores 응답에서 볼 수 있습니다. 외부 키 스토어의 연결 상태는 AWS KMS 콘솔의 KMS 키에 대한 세부 정보 페이지에도 표시됩니다. 세부 정보 페이지에서 Cryptographic configuration(암호화 구성) 탭을 선택하고 Custom key store(사용자 지정 키 스토어) 섹션의 Connection state(연결 상태) 필드를 봅니다.

  연결 상태가 `DISCONNECTED`면 먼저 연결해야 합니다. 연결 상태가 `FAILED`면 문제를 해결하고 외부 키 스토어를 연결 해제한 다음 연결해야 합니다. 지침은 외부 키 저장소 연결 및 연결 해제 섹션을 참조하세요.
- 외부 키 스토어 프록시는 외부 키를 찾을 수 있어야 합니다.
- 외부 키가 활성화되고 암호화 및 복호화를 수행해야 합니다.

  외부 키의 상태는 KMS 키 활성화 및 비활성화를 포함하여 KMS 키의 키 상태 변경과 독립적이며 영향을 받지 않습니다. 마찬가지로 외부 키를 비활성화하거나 삭제해도 KMS 키의 키 상태는 변경되지 않지만 연결된 KMS 키를 사용하는 암호화 작업은 실패합니다.

이러한 조건이 충족되지 않으면 암호화 작업이 실패하고가 `KMSInvalidStateException` 예외를 AWS KMS 반환합니다. 외부 키 스토어를 다시 연결하거나 외부 키 관리자 도구를 사용하여 외부 키를 재구성 또는 복구해야 할 수 있습니다. 추가적인 도움말은 the section called “외부 키 스토어 문제 해결” 섹션을 참조하십시오.

외부 키 스토어에서 KMS 키를 사용할 때는 각 외부 키 스토어의 KMS 키가 암호화 작업에 대해 사용자 지정 키 스토어 요청 할당량을 공유한다는 점에 유의하세요. 할당량을 초과하면가를 AWS KMS 반환합니다`ThrottlingException`. 사용자 지정 키 스토어 요청 할당량에 대한 자세한 내용은 사용자 지정 키 스토어 요청 할당량 섹션을 참조하세요.

자세히 알아보기

- 외부 키 저장소에 대한 자세한 내용은 외부 키 스토어 단원을 참조하세요.
- 외부 키 저장소의 키 구성 요소에 대한 자세한 내용은 외부 키 단원을 참조하세요.
- 외부 키 저장소에서 KMS 키를 생성하려면 외부 키 저장소에서 KMS 키 생성 단원을 참조하세요.
- 외부 키 저장소에서 KMS 키를 식별하고 보려면 외부 키 저장소에서 KMS 키 식별 단원을 참조하세요.
- 외부 키 저장소에서 KMS 키를 삭제할 경우의 특별한 고려 사항에 대해 알아보려면 외부 키 저장소에서 KMS 키 삭제를 참조하세요.

AWS KMS 암호화 필수 요소

AWS KMS 는 암호화에 민감한 아키텍처를 기반으로 구축되어 표준이 발전함에 따라 승인된 알고리즘 또는 모드에서 다른 알고리즘 또는 모드로 원활하게 마이그레이션할 수 있습니다. 당사는 강력하고 널리 검증된 암호화 알고리즘을 선택하고 NIST 승인 옵션의 우선 순위를 지정하여 고객 데이터가 현재 위협과 새로운 위협 모두에서 보호되도록 합니다.

권장 및 허용되는 암호 알고리즘에 대한 자세한 내용은 지원되는 암호화 알고리즘 페이지를 참조하세요.

엔트로피 및 난수 생성

AWS KMS 키 생성은 AWS KMS HSMs에서 수행됩니다. HSM은 AES-256을 사용한 NIST SP800-90A Deterministic Random Bit Generator(DRBG) CTR_DRBG를 사용하는 하이브리드 난수 생성기를 구현합니다. 384비트의 엔트로피가 포함된 비결정적 임의 비트 생성기로 시드되고 추가 엔트로피가 업데이트되어 암호화 구성 요소를 호출 할 때마다 예측 저항을 제공합니다.

대칭 키 작업(암호화만 해당)

HSM에서 사용되는 모든 대칭 키 암호화 명령은 256비트 키를 사용한 Galois Counter Mode(GCM)의 고급 암호화 표준(AES)을 사용합니다. 복호화하기 위한 유사한 호출에는 역 함수가 사용됩니다.

AES-GCM은 인증된 암호화 체계입니다. 암호화 텍스트를 생성하기 위해 일반 텍스트를 암호화하는 것 외에 암호문과 인증이 필요한 추가 데이터(추가로 인증된 데이터 또는 AAD)를 통해 인증 태그를 계산합니다. 이 인증 태그는 데이터가 취합된 소스로부터 얻어진 것이며 암호화 텍스트와 AAD가 수정되지 않았음을 확인하는 데 유용합니다.

자주, 특히 데이터 키 암호화를 참조할 때 설명에 AAD를 포함하지 AWS 않습니다. 암호화할 구조가 암호화될 일반 텍스트와 보호할 일반 텍스트 AAD로 분할되는 이 같은 경우에는 주변 텍스트에 의해 암시됩니다.

AWS KMS 는에 의존하여 키 구성 요소를 생성하는 AWS KMS key 대신 키 구성 요소를 AWS KMS 로 가져올 수 있는 옵션을 제공합니다. 가져온이 키 구성 요소는 RSAES-OAEP를 사용하여 암호화하여 AWS KMS HSM으로 전송하는 동안 키를 보호할 수 있습니다. RSA 키 페어는 AWS KMS HSMs에서 생성됩니다. 가져온 키 구성 요소는 서비스에 저장되기 전에 AWS KMS HSM에서 복호화되고 AES-GCM에서 다시 암호화됩니다.

비대칭 키 작업(암호화, 디지털 서명 및 서명 확인)

AWS KMS 는 암호화, 디지털 서명 및 키 계약 작업 모두에 비대칭 키 작업 사용을 지원합니다. 암호화 및 암호 해독, 서명 및 서명 확인 또는 공유 비밀 도출에 사용할 수 있는 수학적으로 관련된 퍼블릭 키 및 프라이빗 키 페어를 사용합니다. 프라이빗 키는 암호화 AWS KMS 되지 않은 상태로 두지 않습니다. AWS KMS API 작업을 호출 AWS KMS 하여 내에서 퍼블릭 키를 사용하거나 퍼블릭 키를 다운로드하여 외부에서 사용할 수 있습니다 AWS KMS.

AWS KMS 는 다음과 같은 비대칭 암호를 지원합니다.

- RSA-OAEP(암호화용) 및 RSA-PSS와 RSA-PKCS-#1-v1_5(서명 및 확인용) - 다양한 보안 요구 사항에 맞게 2,048, 3,072, 4,096의 RSA 키 길이(비트 단위)를 지원합니다.
- 타원 곡선(ECC)- 서명 및 확인 또는 공유 비밀 도출에 사용되지만 둘 다에 사용되는 것은 아닙니다. ECC 곡선 지원: NIST P256, P384, P521, SECP 256k1, Ed25519.
- ML-DSA - 서명 및 검증에 사용됩니다. 지원되는 ML-DSA 키 사양은 ML_DSA_44, ML_DSA_65 및 ML_DSA_87입니다.
- SM2(중국 리전만 해당) - 암호화 및 복호화, 서명 및 확인 또는 공유 비밀 도출에 사용되지만 이러한 키 사용 중에서 하나를 선택해야 합니다. 암호화를 위해 SM2PKE를 지원하고 서명을 위해 SM2DSA를 지원합니다.

키 유도 함수

키 파생 함수는 초기 보안 암호 또는 키에서 추가 키를 파생하는 데 사용됩니다.는 키 파생 함수(KDF)를 AWS KMS 사용하여 아래의 모든 암호화에 대한 호출당 키를 파생합니다 AWS KMS key. 모든 KDF 작업에는 HMAC [FIPS197]과 SHA256 [FIPS180]을 사용한 카운터 모드의 KDF가 사용됩니다. 추출된 256비트 키는 고객 데이터 및 키를 암호화하거나 복호화하는 데 AES-GCM과 함께 사용됩니다.

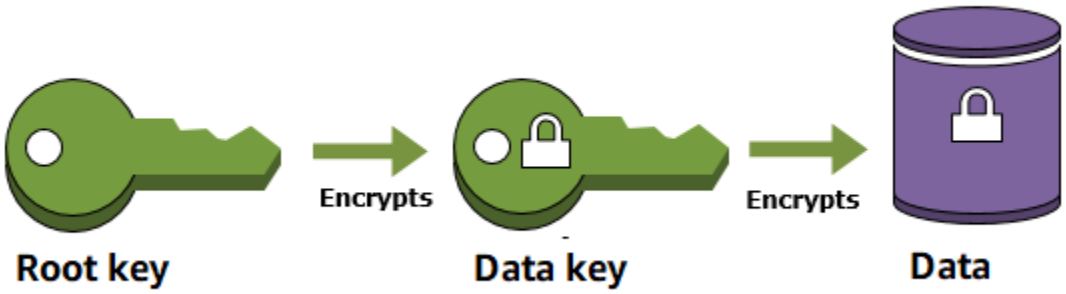
AWS KMS 디지털 서명의 내부 사용

디지털 서명은 AWS KMS 엔터티 간에 명령 및 통신을 인증하는 데에도 사용됩니다. 모든 서비스 엔터티에는 타원 곡선 디지털 서명 알고리즘(ECDSA) 키 페어가 있습니다. 이 키 페어는 Use of Elliptic Curve Cryptography(ECC) Algorithms in Cryptographic Message Syntax(CMS) 및 X9.62-2005: Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm(ECDSA)에 정의된 대로 ECDSA를 수행합니다. 엔터티는 SHA384라고 하는 Federal Information Processing Standards Publications, FIPS PUB 180-4에 정의된 보안 해시 알고리즘을 사용합니다. 모든 키는 secp384r1(NIST-P384) 곡선에 생성됩니다.

봉투 암호화

데이터를 암호화하면 데이터가 보호되지만 암호화 키를 보호해야 합니다. 암호화하는 것도 하나의 전략입니다. 봉투 암호화는 데이터 키로 일반 텍스트 데이터를 암호화한 후, 다른 키 아래에서 데이터 키를 암호화하는 방법입니다.

데이터 암호화 키를 다른 암호화 키로 암호화하고 해당 암호화 키를 다른 암호화 키로 암호화할 수도 있습니다. 그러나 궁극적으로는 키와 데이터를 해독할 수 있도록 하나의 키는 일반 텍스트로 남겨둬야 합니다. 이러한 최상위 일반 텍스트 키 암호화 키를 루트 키라고 합니다.



AWS KMS 를 사용하면 암호화 키를 안전하게 저장하고 관리하여 암호화 키를 보호할 수 있습니다. 로 알려진 AWS KMS에 저장된 루트 키는 AWS KMS FIPS 140-3 보안 수준 3 검증 하드웨어 보안 모듈을 암호화되지 않은 상태로 두지 AWS KMS keys않습니다. KMS 키를 사용하려면를 호출해야 합니다 AWS KMS.

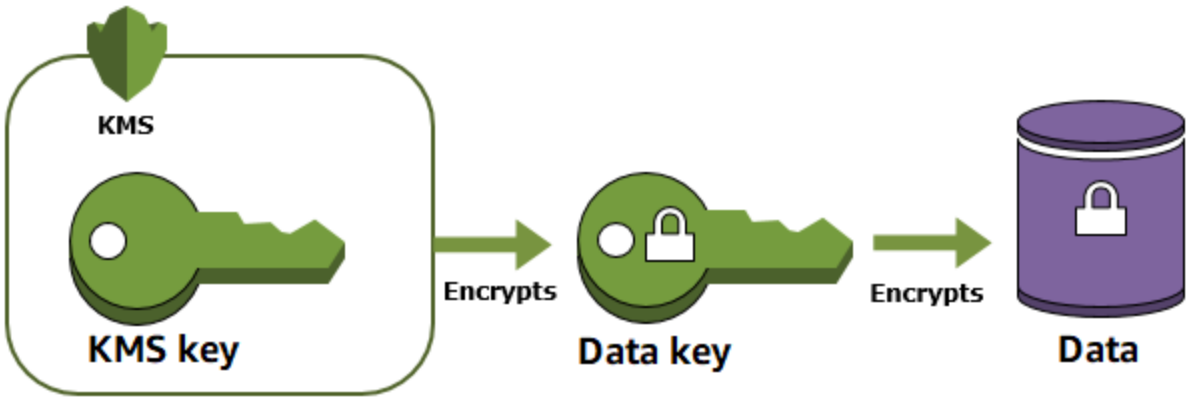
많은 암호화 시스템에 사용되는 기본 구조는 봉투 암호화입니다. 봉투 암호화는 2개 이상의 암호화 키를 사용하여 메시지를 보호합니다. 일반적으로 키 중 하나는 장기 정적 키 k에서 추출되며, 다른 하나는 메시지를 암호화하기 위해 생성되는 메시지별 키 msgKey입니다. 봉투는 ciphertext = Encrypt(msgKey, message)와 같이 메시지를 암호화하여 구성됩니다. 그런 다음 encKey = Encrypt(k,

msgKey)와 같이 메시지 키가 장기 정적 키로 암호화됩니다. 마지막으로, (encKey, ciphertext)라는 두 값이 단일 구조 또는 봉투 암호화 메시지로 패키징됩니다.

k에 대한 액세스 권한이 있는 수신자는 암호화된 키를 먼저 복호화한 다음 메시지를 복호화하여 봉투로 암호화된 메시지를 열 수 있습니다.

AWS KMS 는 이러한 장기 정적 키를 관리하고 데이터의 봉투 암호화 프로세스를 자동화하는 기능을 제공합니다.

AWS Encryption SDK는 AWS KMS 서비스 내에서 제공되는 암호화 기능 외에도 클라이언트 측 봉투 암호화 라이브러리를 제공합니다. 이 라이브러리를 사용하여 데이터 및 해당 데이터를 암호화하는 데 사용되는 암호화 키를 보호할 수 있습니다.



봉투 암호화는 여러 가지 장점을 제공합니다.

- 데이터 키 보호

  데이터 키를 암호화하면 해당 데이터 키의 보안이 암호화에 의해 근본적으로 보호되기 때문에 암호화한 데이터 키의 저장에 대해 고민할 필요가 없습니다. 암호화한 데이터 키를 암호화한 데이터와 함께 안전하게 저장할 수 있습니다.
- 여러 개의 키로 동일한 데이터 암호화

  암호화 작업, 특히 암호화되는 데이터가 대용량 객체일 경우 긴 시간이 걸릴 수 있습니다. 서로 다른 키로 원시 데이터를 여러 차례 다시 암호화하는 대신, 원시 데이터를 보호하는 데이터 키만 다시 암호화할 수 있습니다.
- 여러 알고리즘의 강점 결합

  일반적으로 대칭 키 알고리즘이 퍼블릭 키 알고리즘보다 빠르고 더 작은 암호화 텍스트를 생성합니다. 그러나 퍼블릭 키 알고리즘은 고유한 역할 구분을 제공하고 키 관리가 더 쉽습니다. 봉투 암호화는 각 전략의 장점을 하나로 결합하게 해 줍니다.

암호화 작업

에서 AWS KMS암호화 작업은 KMS 키를 사용하여 데이터를 보호하는 API 작업입니다. KMS 키는 내에 유지되므로 암호화 작업에서 KMS 키를 AWS KMS 사용하려면를 호출 AWS KMS해야 합니다.

KMS 키로 암호화 작업을 수행하려면 AWS SDKs, AWS Command Line Interface (AWS CLI) 또는를 사용합니다 AWS Tools for PowerShell. AWS KMS 콘솔에서 암호화 작업을 수행할 수 없습니다. 여러 프로그래밍 언어로 암호화 작업을 호출하는 예는 AWS SDKs AWS KMS 사용에 대한 코드 예제 섹션을 참조하세요.

다음 표에는 AWS KMS 암호화 작업이 나열되어 있습니다. 또한 작업에 사용되는 KMS 키의 키 유형 및 키 사용 요구 사항을 보여줍니다.

연산	키 유형	키 사용
Decrypt	대칭 또는 비대칭	ENCRYPT_DECRYPT
DeriveSharedSecret	비대칭	KEY_AGREEMENT
암호화	대칭 또는 비대칭	ENCRYPT_DECRYPT
GenerateDataKey	대칭	ENCRYPT_DECRYPT
GenerateDataKeyPair	대칭 [1] 사용자 지정 키 스토어의 KMS 키에서는 지원되지 않습니다.	ENCRYPT_DECRYPT
GenerateDataKeyPairWithoutPlaintext	대칭 [1] 사용자 지정 키 스토어의 KMS 키에서는 지원되지 않습니다.	ENCRYPT_DECRYPT
GenerateDataKeyWithoutPlaintext	대칭	ENCRYPT_DECRYPT
GenerateMac	HMAC	GENERATE_VERIFY_MAC

연산	키 유형	키 사용
GenerateRandom	해당 없음. 이 작업은 KMS 키를 사용하지 않습니다.	해당 사항 없음
ReEncrypt	대칭 또는 비대칭	ENCRYPT_DECRYPT
Sign	비대칭	SIGN_VERIFY
Verify	비대칭	SIGN_VERIFY
VerifyMac	HMAC	GENERATE_ VERIFY_MAC

[1] 대칭 암호화 KMS 키로 보호되는 비대칭 데이터 키 페어를 생성합니다.

암호화 작업과 관련한 권한에 대한 자세한 내용은 the section called “권한 참조”를 참조하세요.

모든 사용자에게 AWS KMS 응답성과 높은 기능을 제공하기 위해는 초당 호출되는 암호화 작업 수에 대한 할당량을 AWS KMS 설정합니다. 자세한 내용은 the section called “암호화 작업에 대한 공유 할당량” 섹션을 참조하십시오.

지원되는 암호화 알고리즘

AWS KMS 는 KMS 키에 대해 다음과 같은 암호화 알고리즘을 지원합니다. 사용할 수 있는 알고리즘은 KMS 키의 키 사양 및 키 사용에 따라 달라집니다. 각 키 사양 및 지원하는 알고리즘에 대한 자세한 설명은 섹션을 참조하세요키 사양 참조.

사용할 알고리즘과 시기에 대한 지침은 암호화 알고리즘 및 단원을 AWS 서비스 참조하십시오.

대칭 키 알고리즘

AWS KMS 는 대칭 KMS 키에 대해 다음 알고리즘을 지원합니다.

대칭 KMS 키에 지원되는 알고리즘

Algorithm	키 사용	키 사양
AES-256-GCM	암호화 및 해독	SYMMETRIC_DEFAULT

Algorithm	키 사용	키 사양
SM4-128(중국 리전만 해당)	암호화 및 해독	SYMMETRIC_DEFAULT
HMAC_SHA_224	HMAC 생성 및 확인	HMAC_224
HMAC_SHA_256	HMAC 생성 및 확인	HMAC_256
HMAC_SHA_384	HMAC 생성 및 확인	HMAC_384
HMAC_SHA_512	HMAC 생성 및 확인	HMAC_512

비대칭 키 알고리즘

각 비대칭 KMS 키에는 사용할 수 있는 알고리즘을 결정하는 단일 키 사용량이 있습니다.

비대칭 KMS 키에 지원되는 알고리즘

Algorithm	키 사용	키 사양
RSAES_OAEP_SHA_1, RSAES_OAEP_SHA_256	암호화 및 해독	RSA_2048, RSA_3072, RSA_4096
RSASSA_PSS_SHA_256, RSASSA_PSS_SHA_384, RSASSA_PSS_SHA_512, RSASSA_PKCS1_V1_5_SHA_256, RSASSA_PKCS1_V1_5_SHA_384, RSASSA_PKCS1_V1_5_SHA_512	서명 및 확인	RSA_2048, RSA_3072, RSA_4096
ECDSA_SHA_256	서명 및 확인	ECC_NIST_P256(secp256r1)
ECDSA_SHA_384	서명 및 확인	ECC_NIST_P384(secp384r1)
ECDSA_SHA_512	서명 및 확인	ECC_NIST_P521(secp521r1)

Algorithm	키 사용	키 사양
ECDSA_SHA_256	서명 및 확인	ECC_SECG_P256K1(secp256k1)
ED25519_SHA_512, ED25519_PH_SHA_512	서명 및 확인	ECC_NIST_EDWARDS25519(ed25519)
ML_DSA_SHAKE_256	서명 및 확인	ML_DSA_44, ML_DSA_65, ML_DSA_87
ECDH	키 계약	ECC_NIST_P256, ECC_NIST_P384, ECC_NIST_P521
SM2PKE(암호화), SM2DSA(서명), ECDH(키 계약)	암호화 및 복호화, 서명 및 확인 또는 키 계약	SM2(중국 리전 전용)

KMS 키 액세스 및 권한

를 사용하려면 AWS KMS가 요청을 인증하는 데 사용할 AWS 수 있는 자격 증명이 있어야 합니다. 자격 증명에는 AWS 리소스 AWS KMS keys 및 별칭에 액세스할 수 있는 권한이 포함되어야 합니다. 해당 권한이 명시적으로 제공되고 거부되지 않는 한 보안 AWS 주체는 KMS 키에 대한 권한이 없습니다. KMS 키를 사용하거나 관리하기 위한 묵시적 권한이나 자동 권한은 없습니다.

KMS 키에 대한 액세스를 제어하는 데 다음 정책 메커니즘을 사용할 수 있습니다.

- 키 정책 - 모든 KMS 키에는 키 정책이 있습니다. 키 정책은 KMS 키에 대한 액세스를 제어하는 기본적인 방법입니다. 키 정책만 사용하여 액세스를 제어할 수 있습니다. 즉 KMS 키에 대한 전체 액세스 범위가 단일 문서(키 정책)에 정의됩니다. 키 정책 사용에 대한 자세한 내용은 키 정책 섹션을 참조하세요.
- IAM 정책 – 키 정책과 함께 IAM 정책을 사용하여 KMS 키에 대한 액세스를 제어하는 권한을 부여할 수 있습니다. 이러한 방식으로 액세스를 제어하면 IAM에서 IAM 자격 증명에 대한 모든 권한을 관리할 수 있습니다. IAM 정책을 사용하여 KMS 키에 대한 액세스를 허용하려면 키 정책에서 명시적으로 허용해야 합니다. IAM 정책 사용에 대한 자세한 내용은 IAM 정책 섹션을 참조하세요.
- 권한 부여 – 키 정책 및 IAM 정책과 함께 권한 부여를 사용하여 KMS 키에 대한 액세스를 허용할 수 있습니다. 이와 같이 액세스를 제어하면 키 정책에서 KMS 키에 대한 액세스를 허용하고 ID가 다른 사람에게 액세스를 위임하도록 허용할 수 있습니다. 권한 부여 사용에 대한 자세한 내용은 의 권한 부여 AWS KMS를 참조하세요.

KMS 키 정책

AWS KMS 리소스에 대한 액세스를 관리하는 기본 방법은 정책을 사용하는 것입니다. 정책은 어떤 보안 주체가 어떤 리소스에 액세스 할 수 있는지를 설명하는 문서입니다. IAM 자격 증명에 연결된 정책을 자격 증명 기반 정책(또는 IAM 정책)이라고 하고, 다른 종류의 리소스에 연결된 정책을 리소스 정책이라고 합니다. KMS 키에 대한 AWS KMS 리소스 정책을 키 정책이라고 합니다.

모든 KMS 키에는 키 정책이 있습니다. 제공하지 않으면에서 자동으로 AWS KMS 생성합니다. 에서 AWS KMS 사용하는 기본 키 정책은 AWS KMS 콘솔에서 키를 생성하는지 아니면 AWS KMS API를 사용하는지에 따라 다릅니다. 조직의 최소 권한 요구 사항에 맞게 기본 키 정책을 편집하는 것이 좋습니다.

키와 IAM 보안 주체가 동일한 AWS 계정에 있는 경우 키 정책만 사용하여 액세스를 제어할 수 있습니다. 즉, KMS 키에 대한 전체 액세스 범위가 단일 문서(키 정책)에 정의되어 있습니다. 그러나 한 계정의

호출자가 다른 계정의 키에 액세스해야 하는 경우 키 정책만 사용하여 액세스 권한을 부여할 수 없습니다. 교차 계정 시나리오에서는 호출자가 API를 직접 호출할 수 있도록 명시적으로 허용하는 IAM 정책을 호출자의 사용자 또는 역할에 연결해야 합니다.

키 정책과 함께 IAM 정책을 사용하여 KMS 키에 대한 액세스를 제어하는 권한을 부여할 수 있습니다. IAM 정책을 사용하여 KMS 키에 대한 액세스를 제어하려면 키 정책에서 계정에 IAM 정책 사용 권한을 부여해야 합니다. IAM 정책을 활성화하는 키 정책 문을 지정하거나 키 정책에서 허용되는 위탁자를 명시적으로 지정할 수 있습니다.

정책을 작성할 때는 다음 작업을 수행할 수 있는 사용자를 제한하는 강력한 제어 기능이 있어야 합니다.

- IAM 및 KMS 키 정책 업데이트, 생성 및 삭제
- 사용자, 역할 및 그룹에서 IAM 정책 연결 및 분리
- KMS 키 정책 연결 및 KMS 키에서 키 정책 분리

KMS 키 권한 부여

IAM 및 키 정책 외에도는 권한 부여를 AWS KMS 지원합니다. 권한 부여는 키 정책을 수정하지 않고도 키에 액세스할 수 있는 IAM 보안 주체 또는 AWS 서비스 보안 주체를 관리할 수 있는 유연하고 강력한 방법을 제공합니다. 여기에는 일부 다른 작업 또는 리소스의 기간 동안에만 권한이 필요한 사용 사례가 포함될 수 있습니다. API를 호출하여 권한 부여를 생성하고 더 이상 필요하지 않을 때 권한 부여를 사용 중지하거나 취소합니다.

자세한 내용은 의 권한 부여 AWS KMS 단원을 참조하십시오.

의 키 정책 AWS KMS

키 정책은에 대한 리소스 정책입니다 AWS KMS key. KMS 키에 대한 액세스를 제어하는 기본 방법입니다. 모든 KMS 키에는 단일 키 정책이 요구되고, 키 정책에 따라 KMS 키의 사용 권한 보유자와 사용 방법이 결정됩니다. KMS 키에 대한 액세스는 IAM 정책 및 권한 부여를 사용하여 제어할 수도 있지만 모든 KMS 키에는 키 정책이 필요합니다.

계정 루트 사용자 또는 키 생성자를 포함한 보안 AWS 주체는 키 정책, IAM 정책 또는 권한 부여에서 명시적으로 허용되거나 거부되지 않는 한 KMS 키에 대한 권한이 없습니다.

키 정책에서 명시적으로 허용하는 경우를 제외하고, IAM 정책을 사용하여 KMS 키에 대한 액세스를 허용할 수는 없습니다. 키 정책의 권한이 없으면 IAM 정책으로 권한을 허용하더라도 효과가 없습니다.

(키 정책의 권한이 없어도 IAM 정책으로 KMS 키에 대한 권한 거부는 가능). 기본 키 정책은 IAM 정책을 활성화합니다. 키 정책에서 IAM 정책을 활성화하려면 에 대한 액세스 허용 AWS 계정 및 IAM 정책 활성화에 기재된 정책을 추가하시기 바랍니다

글로벌 IAM 정책과 달리 키 정책은 리전에 따라 다릅니다. 키 정책은 동일 리전의 KMS 키에 대한 액세스만 제어하고, 다른 리전의 KMS 키에는 영향을 주지 않습니다.

주제

- 키 정책 생성
- 기본 키 정책
- 키 정책 보기
- 키 정책 변경
- 키 정책의 AWS 서비스에 대한 권한

키 정책 생성

AWS KMS 콘솔에서 또는 CreateKey, ReplicateKey, PutKeyPolicy와 같은 AWS KMS API 작업을 사용하여 키 정책을 생성하고 관리할 수 있습니다.

AWS KMS 콘솔에서 KMS 키를 생성하면 콘솔에서 콘솔의 기본 키 정책을 기반으로 키 정책을 생성하는 단계를 안내합니다. `CreateKey` 또는 `ReplicateKey` API를 사용할 경우, 키 정책을 지정하지 않으면 이 API가 프로그래밍 방식으로 생성된 키에 대한 기본 키 정책을 적용합니다. `PutKeyPolicy` API를 사용하는 경우 키 정책을 지정해야 합니다.

각 정책 문서에는 하나 이상의 정책 문이 포함될 수 있습니다. 다음은 정책 문이 하나 포함된 유효한 키 정책 문서를 나타낸 예제입니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "DescribePolicyStatement",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:user/Alice"
      },
```

```
      "Action": "kms:DescribeKey",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "kms:KeySpec": "SYMMETRIC_DEFAULT"
        }
      }
    }
  ]
}
```

주제

- 키 정책 형식
- 키 정책의 요소
- 예제 키 정책

키 정책 형식

키 정책 문서는 다음 규칙을 준수해야 합니다.

- 최대 32KB(32,768바이트)
- 키 정책 문의 `Sid` 요소에는 공백이 포함될 수 있습니다. (IAM 정책 문서의 `Sid` 요소에는 공백이 금지됩니다.)

키 정책 문서에는 다음 문자만 포함될 수 있습니다.

- 인쇄 가능한 ASCII 문자
- Basic Latin 및 Latin-1 Supplement 문자 집합의 인쇄 가능한 문자
- 탭(`\u0009`), 줄 바꿈(`\u000A`) 및 캐리지 리턴(`\u000D`) 특수 문자

키 정책의 요소

키 정책 문서에는 다음 요소가 있어야 합니다.

버전

키 정책 문서 버전을 지정합니다. 버전을 `2012-10-17`(최신 버전)로 설정합니다.

명령

Statement - 정책 문을 포함합니다. 키 정책 문서에는 하나 이상의 문이 있어야 합니다.

각 키 정책 문은 최대 6개의 요소로 구성될 수 있습니다. `Effect`, `Principal`, `Action` 및 `Resource` 요소는 필수입니다.

Sid

(선택 사항) 문 식별자(`Sid`)로서 문 설명에 사용할 수 있는 임의 문자열입니다. 키 정책의 `Sid`에는 공백이 포함될 수 있습니다. (IAM 정책 `Sid` 요소에는 공백을 포함할 수 없습니다.)

Effect

(필수) 정책 문에서 권한을 허용할지 거부할지를 결정합니다. 유효한 값은 `Allow` 또는 `Deny`입니다. KMS 키에 대한 액세스 권한을 명시적으로 허용하지 않으면 액세스가 암시적으로 거부됩니다. KMS 키에 대한 액세스를 명시적으로 거부할 수도 있습니다. 다른 정책에서 액세스 권한을 허용하더라도 사용자가 액세스할 수 없도록 하려고 할 때 이러한 작업을 수행할 수 있습니다.

Principal

(필수) 보안 주체는 정책 설명에 지정된 권한을 갖는 자격 증명입니다. 키 정책에서 AWS 계정, IAM 사용자, IAM 역할 및 일부 AWS 서비스를 보안 주체로 지정할 수 있습니다. IAM 사용자 그룹은 어떤 정책 유형에서도 유효한 보안 주체가 아닙니다.

`"AWS": "*"`와 같은 별표 값은 모든 계정의 모든 AWS ID를 나타냅니다.

> **Important**
>
> 조건을 사용하여 키 정책을 제한하지 않는 한 권한을 허용하는 키 정책문에서 보안 주체를 별표(*)로 설정하지 마세요. 별표는 다른 정책 문이 명시적으로 거부하지 않는 한 모든 자격 증명에 KMS 키를 사용할 수 있는 AWS 계정 권한을 부여합니다. 다른의 사용자는 자신의 계정에 해당 권한이 있을 때마다 KMS 키를 사용할 AWS 계정 수 있습니다.

> **Note**
>
> IAM 모범 사례는 장기 보안 인증 정보가 있는 IAM 사용자의 사용을 장려하지 않습니다. 가능할 경우, 임시 보안 인증 정보를 제공하는 IAM 역할을 사용하세요. 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.

키 정책 문의 보안 주체가 `arn:aws:iam::111122223333:root"` 형식으로 표현된 AWS 계정 보안 주체인 경우 정책 문에서는 IAM 보안 주체에 권한을 부여하지 않습니다. 대신 IAM 정책을 AWS 계정 사용하여 키 정책에 지정된 권한을 위임할 수 있는 권한을에 부여합니다. (계정 식별자에 'root'를 사용하지만 `arn:aws:iam::111122223333:root"` 형식의 보안 주체는 AWS 계정 루트 사용자를 나타내지 않습니다. 계정 보안 주체는 계정과 계정 루트 사용자를 포함한 해당 관리자를 나타냅니다.)

보안 주체가 다른 보안 주체 AWS 계정 이거나 보안 주체인 경우 권한은 KMS 키 및 키 정책을 사용하여 리전에서 계정이 활성화된 경우에만 유효합니다. 기본적으로 사용하도록 설정되지 않은 리전('옵트인 리전')에 대한 자세한 내용은 AWS 일반 참조의 AWS 리전관리를 참조하세요.

다른 AWS 계정 또는 해당 보안 주체가 KMS 키를 사용하도록 허용하려면 키 정책 및 다른 계정의 IAM 정책에 권한을 제공해야 합니다. 자세한 내용은 다른 계정의 사용자가 KMS를 사용하도록 허용을 참조하세요.

Action

(필수) 허용하거나 거부할 API 작업을 지정합니다. 예를 들어 `kms:Encrypt` 작업은 AWS KMS 암호화 작업에 해당합니다. 정책 설명에 두 개 이상의 작업을 나열할 수 있습니다. 자세한 내용은 권한 참조 단원을 참조하십시오.

> **Note**
>
> 필수 `Action` 요소가 키 정책 문에서 누락된 경우 정책 문은 영향을 미치지 않습니다. `Action` 요소가 없는 키 정책 문은 KMS 키에 적용되지 않습니다.
> 키 정책 문에 해당 `Action` 요소가 없는 경우 AWS KMS 콘솔은 오류를 올바르게 보고하지만 정책 문이 효과적이지 않더라도 CreateKey 및 PutKeyPolicy APIs 성공합니다.

Resource

(필수) 키 정책에서 리소스 요소의 값은 `"*"`이고, "이 KMS 키"를 의미합니다. 별표(`"*"`)는 키 정책이 연결된 KMS 키를 식별합니다.

> **Note**
>
> 필수 `Resource` 요소가 키 정책 문에서 누락된 경우 정책 문은 영향을 미치지 않습니다. `Resource` 요소가 없는 키 정책 문은 KMS 키에 적용되지 않습니다.

키 정책 문에 해당 `Resource` 요소가 없는 경우 AWS KMS 콘솔은 오류를 올바르게 보고하지만 정책 문이 효과적이지 않더라도 CreateKey 및 PutKeyPolicy APIs 성공합니다.

조건

(선택 사항) 키 정책이 적용되기 위해 충족해야 하는 요구 사항을 지정합니다. 조건을 사용하여는 API 요청의 컨텍스트를 AWS 평가하여 정책 설명이 적용되는지 여부를 결정할 수 있습니다.

조건을 지정하려면 사전 정의된 조건 키를 사용합니다.는 AWS 전역 조건 키와 AWS KMS 조건 키를 AWS KMS 지원합니다. 속성 기반 액세스 제어(ABAC)를 지원하기 위해는 태그 및 별칭을 기반으로 KMS 키에 대한 액세스를 제어하는 조건 키를 AWS KMS 제공합니다. 자세한 내용은 에 대한 ABAC AWS KMS을 참조하세요.

조건의 형식:

```
"Condition": {"condition operator": {"condition key": "condition value"}}
```

즉, 다음과 같습니다.

```
"Condition": {"StringEquals": {"kms:CallerAccount": "111122223333"}}
```

AWS 정책 구문에 대한 자세한 내용은 AWS IAM 사용 설명서의 IAM 정책 참조를 참조하세요.

예제 키 정책

다음은 대칭 암호화 KMS 키에 대한 전체 키 정책을 나타낸 예제입니다. 이 장의 주요 정책 개념에 대해 읽으면서 참조용으로 사용할 수 있습니다. 이 키 정책은 앞의 기본 키 정책 섹션의 예제 정책 설명을 단일 키 정책에 결합하여 다음을 달성합니다.

- 예제 AWS 계정 111122223333에서 KMS 키에 대한 전체 액세스를 허용합니다. 계정과 해당 관리자(계정 루트 사용자(비상용) 포함)가 해당 계정의 IAM 정책을 사용하여 KMS 키에 대한 액세스를 허용하도록 허용합니다.
- `ExampleAdminRole` IAM 역할에서 KMS 키를 관리할 수 있도록 허용합니다.
- `ExampleUserRole` IAM 역할에서 KMS 키를 사용할 수 있도록 허용합니다.

JSON

```
{
    "Id": "key-consolepolicy",
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "EnableIAMUserPermissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Sid": "AllowKeyAdministratorsAccess",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/ExampleAdminRole"
            },
            "Action": [
                "kms:Create*",
                "kms:Describe*",
                "kms:Enable*",
                "kms:List*",
                "kms:Put*",
                "kms:Update*",
                "kms:Revoke*",
                "kms:Disable*",
                "kms:Get*",
                "kms:Delete*",
                "kms:TagResource",
                "kms:UntagResource",
                "kms:ScheduleKeyDeletion",
                "kms:CancelKeyDeletion",
                "kms:RotateKeyOnDemand"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowKeyUse",
            "Effect": "Allow",
```

```
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/ExampleUserRole"
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowAttachmentPersistentResources",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/ExampleUserRole"
            },
            "Action": [
                "kms:CreateGrant",
                "kms:ListGrants",
                "kms:RevokeGrant"
            ],
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "kms:GrantIsForAWSResource": "true"
                }
            }
        }
    ]
}
```

기본 키 정책

KMS 키를 생성할 때 새 KMS 키에 대한 키 정책을 지정할 수 있습니다. 제공하지 않으면에서 자동으로 AWS KMS 생성합니다. 에서 AWS KMS 사용하는 기본 키 정책은 AWS KMS 콘솔에서 키를 생성하는지 아니면 AWS KMS API를 사용하는지에 따라 다릅니다.

프로그래밍 방식으로 KMS 키를 생성하는 경우 기본 키 정책

AWS KMS API를 사용하여 프로그래밍 방식으로 KMS 키를 생성하고(AWS SDKs AWS Command Line Interface 또는 사용 포함AWS Tools for PowerShell) 키 정책을 지정하지 않으면가 매우 간단한 기본 키 정책을 AWS KMS 적용합니다. 이 기본 키 정책에는 KMS 키를 소유 AWS 계정 한에 IAM 정책을 사용하여 KMS 키의 모든 AWS KMS 작업에 대한 액세스를 허용할 수 있는 권한을 부여하는 정책 문이 하나 있습니다. 이 정책 설명에 대한 자세한 내용은 에 대한 액세스 허용 AWS 계정 및 IAM 정책 활성화 섹션을 참조하세요.

를 사용하여 KMS 키를 생성할 때의 기본 키 정책 AWS Management Console

를 사용하여 KMS 키를 생성 AWS Management Console하면 키 정책은에 대한 액세스를 AWS 계정 허용하고 IAM 정책을 활성화하는 정책 설명으로 시작됩니다. 그런 다음 콘솔은 키 관리자 문, 키 사용자 문 및 (대부분의 키 유형에 대해) 보안 주체가 다른 AWS 서비스와 함께 KMS 키를 사용하도록 허용하는 문을 추가합니다. AWS KMS 콘솔의 기능을 사용하여 IAM 사용자, IAM 역할, 키 관리자 AWS 계정 및 키 사용자(또는 둘 다)를 지정할 수 있습니다.

권한

- 에 대한 액세스 허용 AWS 계정 및 IAM 정책 활성화
- 키 관리자가 KMS 키를 관리하도록 허용
- 키 사용자가 KMS 키를 사용하도록 허용
  - 키 사용자가 암호화 작업에 KMS 키를 사용하도록 허용
  - 키 사용자가 AWS 서비스와 함께 KMS 키를 사용하도록 허용

에 대한 액세스 허용 AWS 계정 및 IAM 정책 활성화

다음 기본 키 정책 문은 매우 중요합니다.

- KMS 키를 소유 AWS 계정 한에 KMS 키에 대한 전체 액세스 권한을 부여합니다.

  다른 AWS 리소스 정책과 달리 AWS KMS 키 정책은 계정 또는 해당 자격 증명에 대한 권한을 자동으로 부여하지 않습니다. 계정 관리자에게 권한을 부여하려면 키 정책에 이와 같이 이 권한을 제공하는 명시적인 문이 포함되어야 합니다.
- 계정이 IAM 정책을 사용하여 키 정책뿐만 아니라 KMS 키에 대한 액세스를 허용할 수 있도록 합니다.

이 권한이 없으면 키에 대한 액세스를 허용하는 IAM 정책은 효과가 없지만 키에 대한 액세스를 거부하는 IAM 정책은 여전히 유효합니다.

- 이는 삭제할 수 없는 계정 루트 사용자를 포함한 계정 관리자에게 액세스 제어 권한을 부여하여 키를 관리할 수 없게 될 위험을 줄입니다.

다음 키 정책 문은 프로그래밍 방식으로 생성된 KMS 키에 대한 전체 기본 키 정책입니다. AWS KMS 콘솔에서 생성된 KMS 키에 대한 기본 키 정책의 첫 번째 정책 설명입니다.

```
{
  "Sid": "Enable IAM User Permissions",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:root"
   },
  "Action": "kms:*",
  "Resource": "*"
}
```

IAM 정책에서 KMS 키에 대한 액세스를 허용할 수 있습니다.

위에 표시된 키 정책 문은 키를 소유 AWS 계정 한에 IAM 정책 및 키 정책을 사용하여 KMS 키에 대한 모든 작업(`kms:*`)을 허용할 수 있는 권한을 부여합니다.

이 키 정책 문의 보안 주체는 계정 보안 주체이며 `arn:aws:iam::account-id:root` 형식의 ARN으로 표시됩니다. 계정 보안 주체는 AWS 계정과 해당 관리자를 나타냅니다.

키 정책 문의 보안 주체가 계정 보안 주체인 경우 정책 문에서는 IAM 보안 주체에 KMS 키를 사용할 권한을 부여하지 않습니다. 대신, 계정이 IAM 정책을 사용하여 정책 문에 지정된 권한을 위임할 수 있도록 허용합니다. 이 기본 키 정책 문은 계정이 IAM 정책을 사용하여 KMS 키에 대한 모든 작업(`kms:*`) 권한을 위임할 수 있도록 허용합니다.

는 KMS 키를 관리할 수 없게 될 위험을 줄입니다.

다른 AWS 리소스 정책과 달리 AWS KMS 키 정책은 계정 또는 해당 보안 주체에게 권한을 자동으로 부여하지 않습니다. 계정 보안 주체를 포함한 모든 보안 주체에게 권한을 부여하려면 권한을 명시적으로 제공하는 키 정책 문을 사용해야 합니다. 계정 보안 주체 또는 다른 보안 주체에게 KMS 키에 대한 액세스 권한을 부여할 필요는 없습니다. 그러나 계정 보안 주체에게 액세스 권한을 부여하면 키를 관리할 수 없게 되는 문제를 방지할 수 있습니다.

예를 들어 한 명의 사용자에게만 KMS 키에 대한 액세스 권한을 부여하는 키 정책을 만들었다고 가정해 보겠습니다. 그런 다음 해당 사용자를 삭제하면 키를 관리할 수 없게 되므로 AWS Support에 문의하여 KMS 키에 대한 액세스 권한을 다시 받아야 합니다.

위에 표시된 키 정책 문은 계정 위탁자에게 키를 제어할 수 있는 권한을 부여합니다. 계정 보안 주체는 계정 루트 사용자를 포함하여 AWS 계정 및 해당 관리자를 나타냅니다. 계정 루트 사용자는 AWS 계정을 삭제하지 않으면 삭제할 수 없는 유일한 보안 주체입니다. IAM 모범 사례에서는 긴급 상황을 제외하고는 계정 루트 사용자를 대신하여 조치를 취하는 것을 권장하지 않습니다. 하지만 KMS 키에 액세스할 수 있는 다른 모든 사용자 및 역할을 삭제할 경우 계정 루트 사용자 역할을 해야 할 수도 있습니다.

키 관리자가 KMS 키를 관리하도록 허용

콘솔이 생성한 기본 키 정책을 통해 해당 계정에서 IAM 사용자와 역할을 선택하고 이들을 키 관리자로 지정할 수 있습니다. 이 문은 키 관리자 문이라고 합니다. 키 관리자는 KMS 키를 관리할 수 있는 권한이 있지만 암호화 작업에 KMS 키를 사용할 수 있는 권한은 없습니다. 기본 보기 또는 정책 보기에서 KMS 키를 생성할 때 키 관리자 목록에 IAM 사용자와 역할을 추가할 수 있습니다.

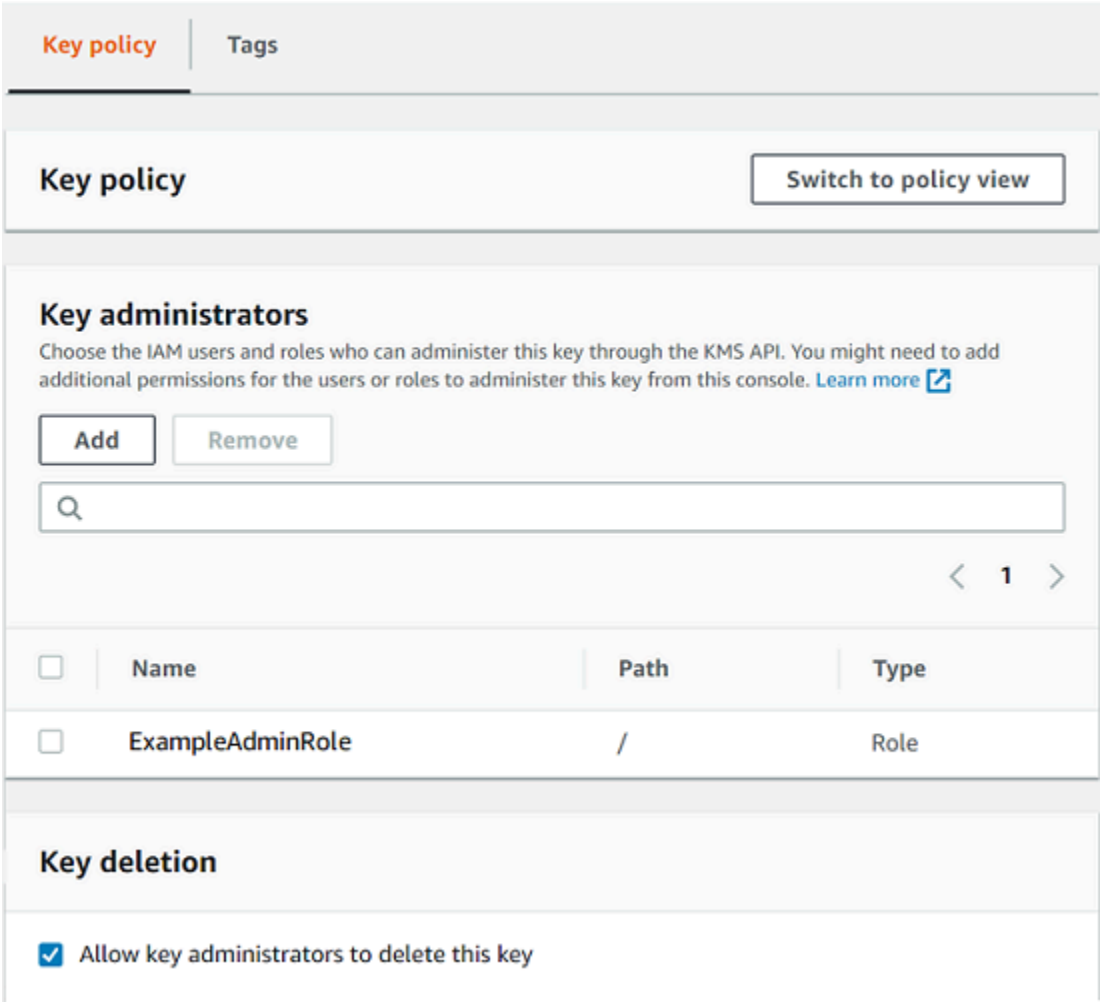
Warning

키 관리자는 키 정책을 변경하고 권한 부여를 생성할 수 있는 권한이 있으므로 자신과 다른 사용자에게이 정책에 지정되지 않은 AWS KMS 권한을 부여할 수 있습니다.
태그 및 별칭을 관리할 권한이 있는 보안 주체는 KMS 키에 대한 액세스를 제어할 수도 있습니다. 자세한 내용은 에 대한 ABAC AWS KMS을 참조하세요.

Note

IAM 모범 사례는 장기 보안 인증 정보가 있는 IAM 사용자의 사용을 장려하지 않습니다. 가능할 경우, 임시 보안 인증 정보를 제공하는 IAM 역할을 사용하세요. 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.

다음은 AWS KMS 콘솔의 기본 보기에서 키 관리자 문을 나타낸 예제입니다.



다음은 AWS KMS 콘솔의 기본 보기의 예제 키 관리자 문입니다. 이 키 관리자 문은 단일 리전 대칭 암호화 KMS 키를 위한 것입니다.

> **Note**
>
> AWS KMS 콘솔은 문 식별자 아래의 키 정책에 키 관리자를 추가합니다`"Allow access for Key Administrators"`. 이 문 식별자를 수정하면 문에 대해 콘솔이 업데이트 내용을 표시하는 방식에 영향을 미칠 수 있습니다.

```
{
  "Sid": "Allow access for Key Administrators",
  "Effect": "Allow",
  "Principal": {"AWS":"arn:aws:iam::111122223333:role/ExampleAdminRole"},
  "Action": [
    "kms:Create*",
```

```
    "kms:Describe*",
    "kms:Enable*",
    "kms:List*",
    "kms:Put*",
    "kms:Update*",
    "kms:Revoke*",
    "kms:Disable*",
    "kms:Get*",
    "kms:Delete*",
    "kms:TagResource",
    "kms:UntagResource",
    "kms:ScheduleKeyDeletion",
    "kms:CancelKeyDeletion",
    "kms:RotateKeyOnDemand"
  ],
  "Resource": "*"
}
```

가장 일반적인 KMS 키인 단일 리전 대칭 KMS 키에 대한 기본 키 관리자 문은 다음과 같은 권한을 허용합니다. 이러한 권한에 대한 자세한 내용은 AWS KMS 권한 섹션을 참조하세요.

AWS KMS 콘솔을 사용하여 KMS 키를 생성하면 콘솔은 키 관리자 문의 `Principal` 요소에 지정한 사용자와 역할을 추가합니다.

많은 경우 이러한 권한에 와일드카드 문자(*)가 포함되며, 지정된 동사로 시작하는 모든 권한이 허용됩니다. 따라서가 새 API 작업을 AWS KMS 추가하면 키 관리자가 해당 작업을 자동으로 사용할 수 있습니다. 새 작업을 포함하도록 키 정책을 업데이트할 필요는 없습니다. 키 관리자를 고정된 API 작업 집합으로 제한하려는 경우 키 정책을 변경할 수 있습니다.

`kms:Create*`

`kms:CreateAlias` 및 `kms:CreateGrant`를 허용합니다. (`kms:CreateKey` 권한은 IAM 정책에서만 유효)

`kms:Describe*`

`kms:DescribeKey`를 허용합니다. AWS Management Console에서 KMS 키의 키 세부 정보 페이지를 보려면 `kms:DescribeKey` 권한이 필요합니다.

`kms:Enable*`

`kms:EnableKey`를 허용합니다. 대칭 암호화 KMS 키의 경우 `kms:EnableKeyRotation`도 허용합니다.

kms:List*

kms:ListGrants, kms:ListKeyPolicies 및 kms:ListResourceTags를 허용합니다. (AWS Management Console에서 KMS 키를 보는 데 필요한 kms:ListAliases 및 kms:ListKeys 권한은 IAM 정책에서만 유효)

kms:Put*

kms:PutKeyPolicy를 허용합니다. 이 권한은 키 관리자가 이 KMS 키에 대한 키 정책을 변경하도록 허용합니다.

kms:Update*

kms:UpdateAlias 및 kms:UpdateKeyDescription을 허용합니다. 다중 리전 키의 경우 이 KMS 키에서 kms:UpdatePrimaryRegion을 허용합니다.

kms:Revoke*

키 관리자가 권한 부여에서 사용 중지 위탁자가 아니어도 권한 부여를 삭제할 수 있도록 kms:RevokeGrant를 허용합니다.

kms:Disable*

kms:DisableKey를 허용합니다. 대칭 암호화 KMS 키의 경우 kms:DisableKeyRotation도 허용합니다.

kms:Get*

kms:GetKeyPolicy 및 kms:GetKeyRotationStatus를 허용합니다. 가져온 키 구성 요소가 있는 KMS 키의 경우 kms:GetParametersForImport를 허용합니다. 비대칭 KMS 키의 경우 kms:GetPublicKey를 허용합니다. AWS Management Console에서 KMS 키의 키 정책을 보려면 kms:GetKeyPolicy 권한이 필요합니다.

kms:Delete*

kms:DeleteAlias를 허용합니다. 가져온 키 구성 요소가 있는 키의 경우 kms:DeleteImportedKeyMaterial을 허용합니다. kms:Delete* 권한은 키 관리자가 KMS 키(ScheduleKeyDeletion)를 삭제하도록 허용하지 않습니다.

kms:TagResource

키 관리자가 KMS 키에 태그를 추가할 수 있도록 하는 kms:TagResource를 허용합니다. 태그를 사용하여 KMS 키에 대한 액세스를 제어할 수도 있으므로 이 권한을 통해 관리자는 KMS 키에 대한 액세스를 허용하거나 거부할 수 있습니다. 자세한 내용은 에 대한 ABAC AWS KMS 섹션을 참조하세요.

kms:UntagResource

키 관리자가 KMS 키에 태그를 삭제할 수 있도록 하는 kms:UntagResource를 허용합니다. 태그를 사용하여 키에 대한 액세스를 제어할 수 있으므로 이 권한을 통해 관리자는 KMS 키에 대한 액세스를 허용하거나 거부할 수 있습니다. 자세한 내용은 에 대한 ABAC AWS KMS 섹션을 참조하세요.

kms:ScheduleKeyDeletion

키 관리자가 이 KMS 키를 삭제할 수 있도록 하는 kms:ScheduleKeyDeletion을 허용합니다. 이 권한을 삭제하려면 키 관리자가 이 키를 삭제하도록 허용(Allow key administrators to delete this key) 옵션을 삭제합니다.

kms:CancelKeyDeletion

키 관리자가 이 KMS 키 삭제를 취소할 수 있도록 하는 kms:CancelKeyDeletion를 허용합니다. 이 권한을 삭제하려면 키 관리자가 이 키를 삭제하도록 허용(Allow key administrators to delete this key) 옵션을 삭제합니다.

kms:RotateKeyOnDemand

키 관리자가 이 KMS 키에서 키 구성 요소의 온디맨드 교체를 수행할 수 있는 kms:RotateKeyOnDemand를 허용합니다.

AWS KMS 는 특수 목적 키를 생성할 때 기본 키 관리자 문에 다음 권한을 추가합니다.

kms:ImportKeyMaterial

kms:ImportKeyMaterial 권한은 키 관리자가 키 구성 요소를 KMS로 가져오도록 허용합니다. 이 권한은 키 구성 요소 없이 KMS 키를 생성하는 경우에만 키 정책에 포함됩니다.

kms:ReplicateKey

kms:ReplicateKey 권한을 통해 키 관리자는 다른 AWS 리전에서 다중 리전 기본 키의 복제본을 생성할 수 있습니다. 이 권한은 다중 리전 기본 또는 복제본 키를 생성하는 경우에만 키 정책에 포함됩니다.

kms:UpdatePrimaryRegion

kms:UpdatePrimaryRegion 권한은 키 관리자가 다중 리전 복제본 키를 다중 리전 기본 키로 변경하도록 허용합니다. 이 권한은 다중 리전 기본 또는 복제본 키를 생성하는 경우에만 키 정책에 포함됩니다.

키 사용자가 KMS 키를 사용하도록 허용

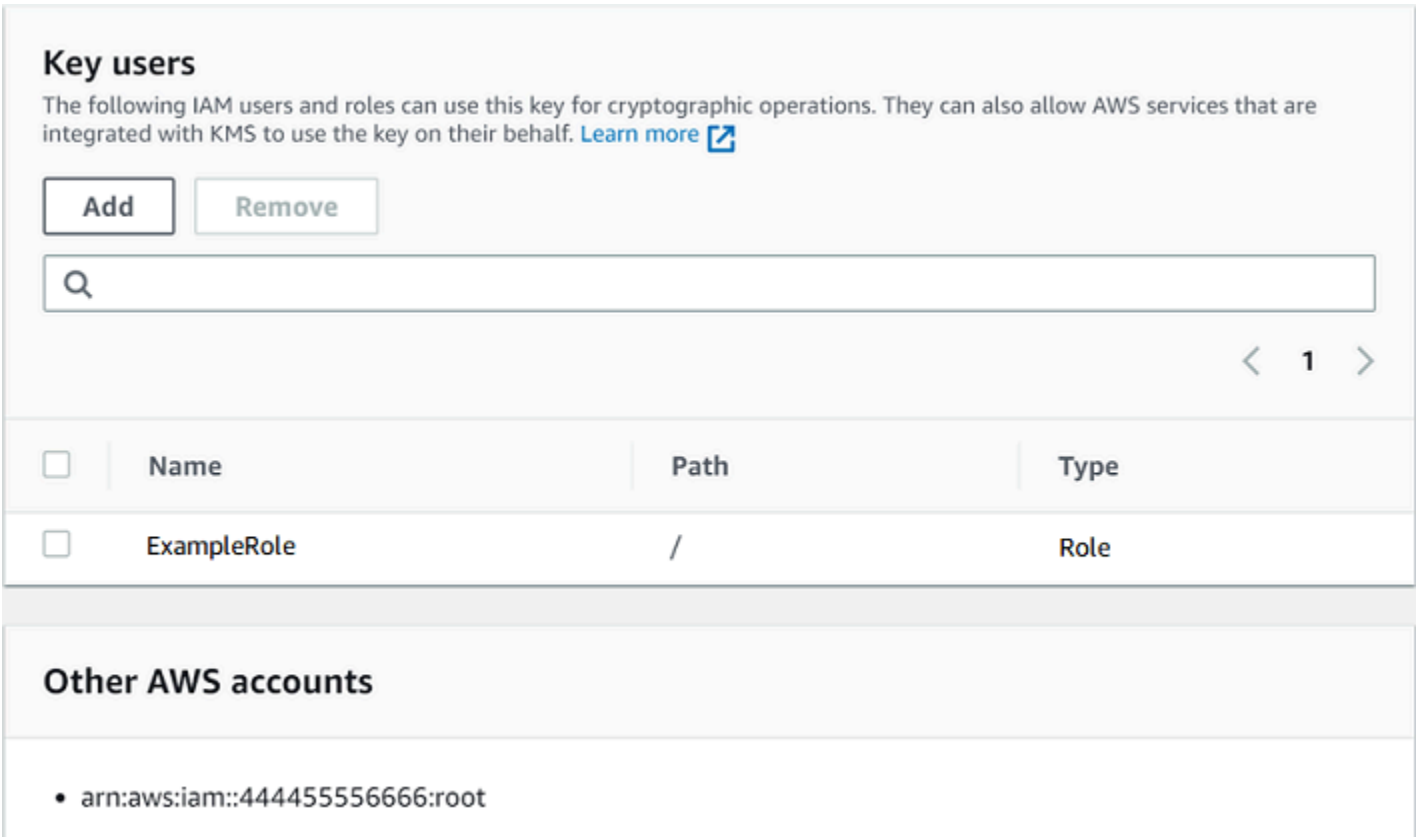
콘솔에서 KMS 키에 대해 생성하는 기본 키 정책을 사용하면 계정 및 외부에서 IAM 사용자 AWS 계정 및 IAM 역할을 선택하고 키 사용자를 만들 수 있습니다.

콘솔은 키 사용자에 대한 키 정책에 두 개의 정책 문을 추가합니다.

- KMS 직접 사용 — 첫 번째 키 정책 문은 키 사용자에게 해당 유형의 KMS 키에 대해 지원되는 모든 암호화 작업에 KMS 키를 직접 사용할 수 있는 권한을 부여합니다.
- 서비스와 함께 AWS KMS 키 사용 - 두 번째 정책 설명은 키 사용자에게와 통합된 AWS 서비스가 자신을 대신하여 KMS 키를 AWS KMS 사용하여 Amazon S3 버킷 및 Amazon DynamoDB 테이블과 같은 리소스를 보호할 수 있는 권한을 부여합니다.

KMS 키를 생성할 때 IAM 사용자, IAM 역할 및 기타 AWS 계정 를 키 사용자 목록에 추가할 수 있습니다. 다음 이미지에서 보듯이, 키 정책에 대한 콘솔의 기본 보기로 목록을 편집할 수도 있습니다. 키 정책의 기본 보기는 키 세부 정보 페이지에 있습니다. 다른의 사용자가 KMS 키를 사용하도록 허용하는 AWS 계정 방법에 대한 자세한 내용은 섹션을 참조하세요다른 계정의 사용자가 KMS를 사용하도록 허용.

> **Note**
>
> IAM 모범 사례는 장기 보안 인증 정보가 있는 IAM 사용자의 사용을 장려하지 않습니다. 가능할 경우, 임시 보안 인증 정보를 제공하는 IAM 역할을 사용하세요. 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.



단일 리전 대칭에 대한 기본 키 사용자 문은 다음과 같은 권한을 허용합니다. 각 권한에 대한 자세한 내용은 AWS KMS 권한 섹션을 참조하세요.

AWS KMS 콘솔을 사용하여 KMS 키를 생성하면 콘솔은 사용자가 지정한 사용자와 역할을 각 키 사용자 문의 `Principal` 요소에 추가합니다.

> **Note**
>
> AWS KMS 콘솔은 문 식별자 `"Allow use of the key"` 및 아래의 키 정책에 키 사용자를 추가합니다`"Allow attachment of persistent resources"`. 이러한 문 식별자들을 수정하면 문에 대해 콘솔이 업데이트 내용을 표시하는 방식에 영향을 미칠 수 있습니다.

```
{
  "Sid": "Allow use of the key",
  "Effect": "Allow",
  "Principal": {"AWS": [
    "arn:aws:iam::111122223333:role/ExampleRole",
    "arn:aws:iam::444455556666:root"
  ]},
```

```
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey"
  ],
  "Resource": "*"
},
{
  "Sid": "Allow attachment of persistent resources",
  "Effect": "Allow",
  "Principal": {"AWS": [
    "arn:aws:iam::111122223333:role/ExampleRole",
    "arn:aws:iam::444455556666:root"
  ]},
  "Action": [
    "kms:CreateGrant",
    "kms:ListGrants",
    "kms:RevokeGrant"
  ],
  "Resource": "*",
  "Condition": {"Bool": {"kms:GrantIsForAWSResource": true}}
}
```

키 사용자가 암호화 작업에 KMS 키를 사용하도록 허용

키 사용자는 KMS 키에서 지원되는 모든 암호화 작업에서 KMS 키를 직접 사용할 수 있는 권한이 있습니다. 또한 DescribeKey 작업을 사용하여 AWS KMS 콘솔에서 또는 AWS KMS API 작업을 사용하여 KMS 키에 대한 자세한 정보를 가져올 수 있습니다.

기본적으로 AWS KMS 콘솔은 다음 예제와 같은 키 사용자 문을 기본 키 정책에 추가합니다. 서로 다른 API 작업을 지원하므로 대칭 암호화 KMS 키, HMAC KMS 키, 퍼블릭 키 암호화를 위한 비대칭 KMS 키, 서명 및 확인을 위한 비대칭 KMS 키에 대한 정책 문의 작업은 약간 다릅니다.

대칭 암호화 KMS 키

콘솔은 대칭 암호화 KMS 키에 대한 키 정책에 다음 문을 추가합니다.

```
{
  "Sid": "Allow use of the key",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:role/ExampleKeyUserRole"},
```

```
  "Action": [
    "kms:Decrypt",
    "kms:DescribeKey",
    "kms:Encrypt",
    "kms:GenerateDataKey*",
    "kms:ReEncrypt*"
  ],
  "Resource": "*"
}
```

HMAC KMS 키

콘솔은 HMAC KMS 키에 대한 키 정책에 다음 문을 추가합니다.

```
{
  "Sid": "Allow use of the key",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:role/ExampleKeyUserRole"},
  "Action": [
    "kms:DescribeKey",
    "kms:GenerateMac",
    "kms:VerifyMac"
  ],
  "Resource": "*"
}
```

퍼블릭 키 암호화를 위한 비대칭 KMS 키

콘솔은 키 사용이 암호화 및 해독인 비대칭 KMS 키에 대한 키 정책에 다음 문을 추가합니다.

```
{
  "Sid": "Allow use of the key",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleKeyUserRole"
  },
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:DescribeKey",
    "kms:GetPublicKey"
  ],
```

```
  "Resource": "*"
}
```

서명 및 확인을 위한 비대칭 KMS 키

콘솔은 키 사용이 서명 및 확인인 비대칭 KMS 키에 대한 키 정책에 다음 문을 추가합니다.

```
{
  "Sid": "Allow use of the key",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:role/ExampleKeyUserRole"},
  "Action": [
    "kms:DescribeKey",
    "kms:GetPublicKey",
    "kms:Sign",
    "kms:Verify"
  ],
  "Resource": "*"
}
```

공유 보안 비밀 도출을 위한 비대칭 KMS 키

콘솔은 키 사용에 대한 키 계약을 사용하여 비대칭 KMS 키에 대한 키 정책에 다음 문을 추가합니다.

```
{
  "Sid": "Allow use of the key",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:role/ExampleKeyUserRole"},
  "Action": [
    "kms:DescribeKey",
    "kms:GetPublicKey",
    "kms:DeriveSharedSecret"
  ],
  "Resource": "*"
}
```

이러한 문의 작업은 키 사용자에게 다음 권한을 제공합니다.

kms:Encrypt

키 사용자가 이 KMS 키를 사용하여 데이터를 암호화하도록 허용합니다.

kms:Decrypt

키 사용자가 이 KMS 키를 사용하여 데이터를 해독하도록 허용합니다.

kms:DeriveSharedSecret

키 사용자가 이 KMS 키로 공유 비밀을 도출할 수 있습니다.

kms:DescribeKey

키 사용자가 식별자, 생성 날짜, 키 상태 등 이 KMS 키에 대한 세부 정보를 가져오도록 허용합니다. 또한 키 사용자가 AWS KMS 콘솔에서 KMS 키에 대한 세부 정보를 표시할 수 있습니다.

kms:GenerateDataKey*

키 사용자가 클라이언트 측 암호화 작업을 위해 대칭 데이터 키 또는 비대칭 데이터 키 페어를 요청하도록 허용합니다. 콘솔은 * 와일드카드 문자를 사용하여 다음 API 작업에 대한 권한을 나타냅니다. GenerateDataKey, GenerateDataKeyWithoutPlaintext, GenerateDataKeyPair, GenerateDataKeyPairWithoutPlaintext. 이러한 권한은 데이터 키를 암호화하는 대칭 KMS 키에만 유효합니다.

kms:GenerateMac

키 사용자가 HMAC KMS 키를 사용하여 HMAC 태그를 생성하도록 허용합니다.

kms:GetPublicKey

키 사용자가 비대칭 KMS 키의 퍼블릭 키를 다운로드하도록 허용합니다. 이 퍼블릭 키를 공유하는 당사자는 외부에서 데이터를 암호화할 수 있습니다 AWS KMS. 그러나, 그 암호문은 AWS KMS에서 Decrypt 작업을 호출해야만 해독할 수 있습니다.

kms:ReEncrypt*

키 사용자가 원래 이 KMS 키로 암호화한 데이터를 다시 암호화하거나, 이 KMS 키를 이용해 이전에 암호화한 데이터를 다시 암호화하도록 허용합니다. ReEncrypt 작업을 수행하려면 소스 및 대상 KMS 키 모두에 대한 액세스 권한이 필요합니다. 이를 위해 원본 KMS 키에 대한 `kms:ReEncryptFrom` 권한과 대상 KMS 키에 대한 `kms:ReEncryptTo` 권한을 허용할 수 있습니다. 그러나 단순화를 위해 콘솔에서는 두 KMS 키 모두에 대해 `kms:ReEncrypt*`를 허용합니다(* 와일드카드 문자 사용).

kms:Sign

키 사용자가 이 KMS 키를 사용하여 메시지에 서명하도록 허용합니다.

kms:Verify

키 사용자가 이 KMS 키를 사용하여 서명을 확인하도록 허용합니다.

kms:VerifyMac

키 사용자가 HMAC KMS 키를 사용하여 HMAC 태그를 확인하도록 허용합니다.

키 사용자가 AWS 서비스와 함께 KMS 키를 사용하도록 허용

또한 콘솔의 기본 키 정책은 키 사용자에게 권한 부여를 사용하는 AWS 서비스에서 데이터를 보호하는 데 필요한 권한 부여 권한을 부여합니다. AWS 서비스는 권한 부여를 사용하여 KMS 키를 사용할 수 있는 구체적이고 제한된 권한을 얻는 경우가 많습니다.

이 키 정책 문을 사용하면 키 사용자가 KMS 키에 대한 권한 부여를 생성, 확인 및 취소할 수 있지만 권한 부여 작업 요청이 AWS 와 통합된 서비스 AWS KMS에서 오는 경우에만 가능합니다. kms:GrantIsForAWSResource 정책 조건은 사용자가 이러한 권한 부여 작업을 직접 호출하도록 허용하지 않습니다. 키 사용자가 허용하면 AWS 서비스는 사용자를 대신하여 KMS 키를 사용하여 사용자의 데이터를 보호할 수 있는 권한 부여를 생성할 수 있습니다.

키 사용자는 통합 서비스와 함께 KMS 키를 사용하려면 이러한 권한 부여 권한이 필요하지만 이러한 권한으로는 충분하지 않습니다. 키 사용자는 통합 서비스를 사용할 수 있는 권한도 필요합니다. 와 통합되는 AWS 서비스에 대한 액세스 권한을 사용자에게 부여하는 방법에 대한 자세한 내용은 통합 서비스에 대한 설명서를 AWS KMS참조하세요.

```
{
  "Sid": "Allow attachment of persistent resources",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:role/ExampleKeyUserRole"},
  "Action": [
    "kms:CreateGrant",
    "kms:ListGrants",
    "kms:RevokeGrant"
  ],
  "Resource": "*",
  "Condition": {"Bool": {"kms:GrantIsForAWSResource": true}}
}
```

예를 들어 키 사용자는 다음과 같은 방법으로 KMS 키에서 이러한 권한을 사용할 수 있습니다.

- 이 KMS 키를 Amazon Elastic Block Store(Amazon EBS) 및 Amazon Elastic Compute Cloud(Amazon EC2)와 함께 사용하여 암호화된 EBS 볼륨을 EC2 인스턴스에 연결합니다. 키 사용자는 KMS 키를 사용할 수 있는 Amazon EC2 권한을 묵시적으로 부여해 암호화된 볼륨을 인스턴스

에 연결합니다. 자세한 내용은 Amazon Elastic Block Store(Amazon EBS)의 사용 방식 AWS KMS 단원을 참조하십시오.

- Amazon Redshift와 함께 이 KMS 키를 사용하여 암호화된 클러스터를 시작합니다. 키 사용자는 KMS 키를 사용할 수 있는 Amazon Redshift 권한을 묵시적으로 부여해 암호화된 클러스터를 시작하고 암호화된 스냅샷을 생성합니다. 자세한 내용은 Amazon Redshift의 사용 방식 AWS KMS 단원을 참조하십시오.
- 권한 부여를 사용해 해당 서비스로 암호화된 리소스를 생성, 관리 또는 사용하는 AWS KMS와 통합된 다른AWS 서비스와 함께 이 KMS 키를 사용합니다.

기본 키 정책은 키 사용자가 권한 부여를 사용하는 모든 통합 서비스에 권한 부여 권한을 위임하도록 허용합니다. 그러나 권한을 지정된 AWS 서비스로 제한하는 사용자 지정 키 정책을 생성할 수 있습니다. 자세한 내용은 kms:ViaService 조건 키를 참조하세요.

키 정책 보기

AWS KMS API에서 AWS KMS 콘솔 또는 GetKeyPolicy 작업을 사용하여 계정AWS 관리형 키의 AWS KMS 고객 관리형 키 또는에 대한 키 정책을 볼 수 있습니다. 이러한 기법을 사용하여 다른 AWS 계정에서 KMS 키의 키 정책을 볼 수 없습니다.

AWS KMS 키 정책에 대한 자세한 내용은 섹션을 참조하세요의 키 정책 AWS KMS. KMS 키에 액세스할 수 있는 사용자 및 역할을 확인하는 방법은 the section called “액세스 결정” 단원을 참조하십시오.

AWS KMS 콘솔 사용

권한이 부여된 사용자는 AWS Management Console의 키 정책(Key policy) 탭에서 AWS 관리형 키 또는 고객 관리형 키에 대한 키 정책을 볼 수 있습니다.

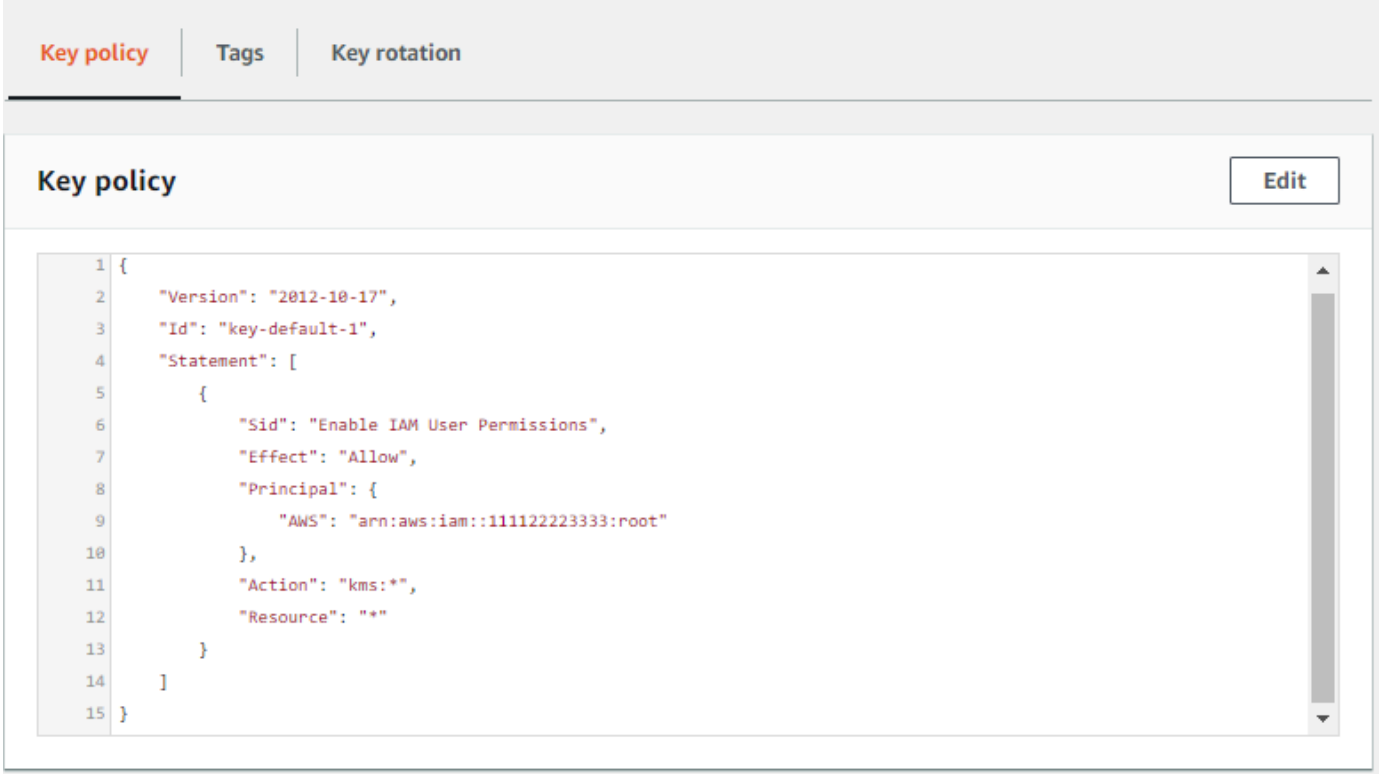
에서 KMS 키에 대한 키 정책을 보려면 kms:ListAliases, kms:DescribeKey 및 kms:GetKeyPolicy 권한이 있어야 AWS Management Console합니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 가 AWS 자동으로 생성하고 관리하는 계정의 키를 보려면 탐색 창에서 AWS 관리형 키를 선택합니다. 해당 계정에서 직접 생성하고 관리하는 키를 보려면 탐색 창에서 고객 관리형 키를 선택합니다.

4. KMS 키 목록에서 검사하려는 KMS 키의 별칭 또는 키 ID를 선택합니다.
5. 키 정책(Key policy) 탭을 선택합니다.

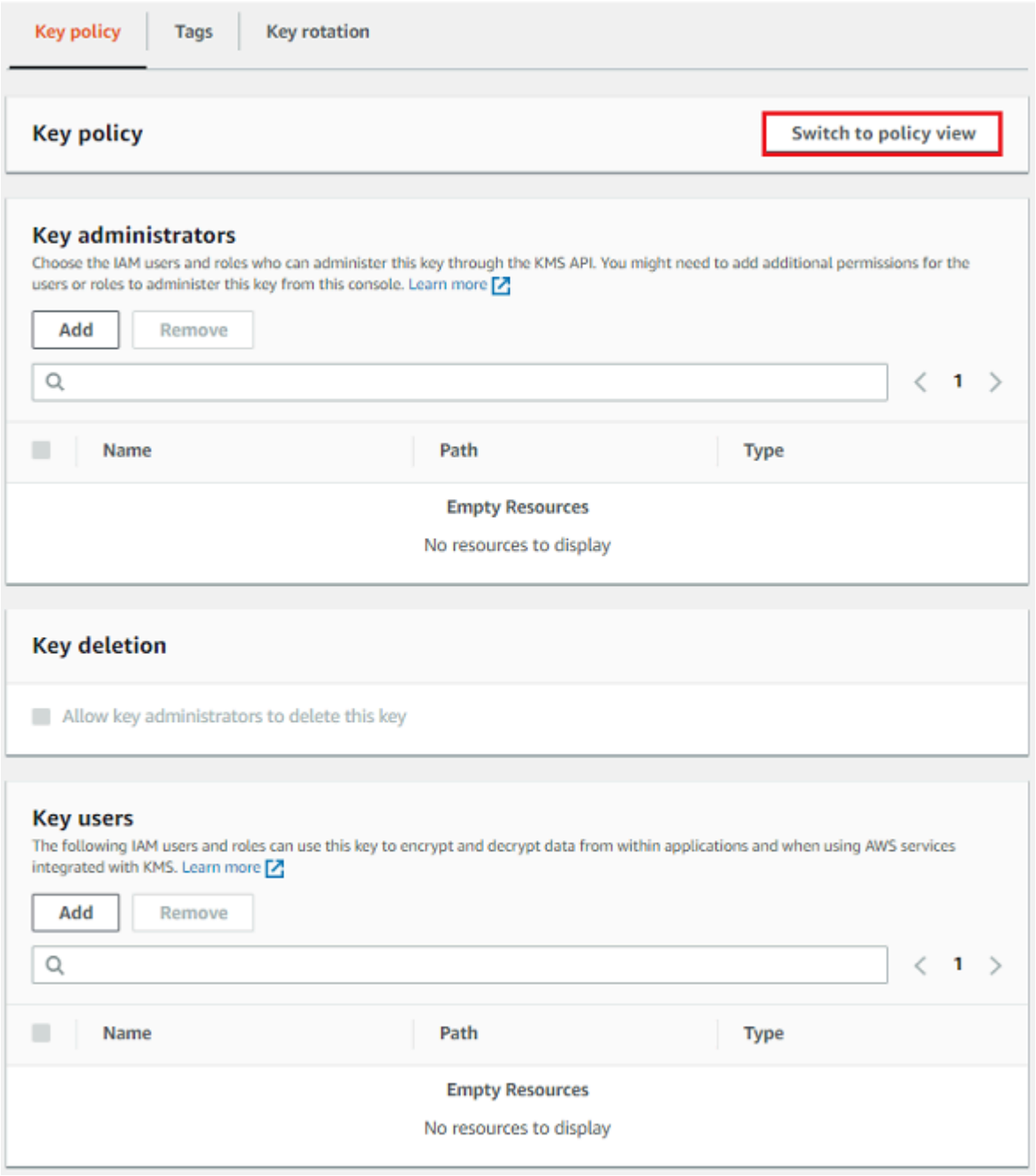
   키 정책(Key policy) 탭에서 키 정책 문서를 볼 수 있습니다. 다음은 정책 보기입니다. 키 정책 문에서 키 정책을 통해 KMS 키에 대한 액세스 권한이 부여된 보안 주체를 볼 수 있고, 이러한 보안 주체가 수행할 수 있는 작업을 볼 수 있습니다.

   다음 예제에서는 기본 키 정책에 대한 정책 보기를 보여 줍니다.



   또는에서 KMS 키를 생성한 경우 키 관리자 AWS Management Console, 키 삭제 및 키 사용자에 대한 섹션이 포함된 기본 보기가 표시됩니다. 키 정책 문서를 보려면 정책 보기로 전환(Switch to policy view)을 선택합니다.

   다음 예제에서는 기본 키 정책에 대한 기본 보기를 보여 줍니다.



AWS KMS API 사용

에서 KMS 키에 대한 키 정책을 가져오려면 API에서 AWS KMS GetKeyPolicy 작업을 AWS 계정사용합니다. 이 작업을 사용하여 다른 계정의 키 정책은 볼 수 없습니다.

다음 예제에서는 AWS Command Line Interface (AWS CLI)에서 get-key-policy 명령을 사용하지만 SDK를 AWS 사용하여이 요청을 할 수 있습니다.

default가 유일하게 유효한 값이더라도 PolicyName 파라미터가 필요합니다. 또한 이 명령은 더 쉽게 볼 수 있도록 JSON이 아닌 텍스트로 출력을 요청합니다.

이 명령을 실행하기 앞서 예제 키 ID를 유효한 키 ID로 바꿉니다.

```
$ aws kms get-key-policy --key-id 1234abcd-12ab-34cd-56ef-1234567890ab --policy-name
 default --output text
```

응답은 다음과 같이 기본 키 정책을 반환합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Id" : "key-consolepolicy-3",
  "Statement" : [ {
  "Sid" : "EnableIAMUserPermissions",
    "Effect" : "Allow",
    "Principal" : {
      "AWS" : "arn:aws:iam::111122223333:root"
    },
    "Action" : "kms:*",
    "Resource" : "*"
  } ]
}
```

키 정책 변경

AWS Management Console 또는 PutKeyPolicy 작업을 AWS 계정 사용하여에서 KMS 키에 대한 키 정책을 변경할 수 있습니다. 이러한 기법을 사용하여 다른 AWS 계정에서 KMS 키의 키 정책을 변경할 수 없습니다.

키 정책을 변경하는 경우 다음 사항을 주의해야 합니다.

- AWS 관리형 키 또는 고객 관리형 키의 키 정책은 볼 수 있지만 고객 관리형 키의 키 정책만 변경할 수 있습니다. 의 AWS 관리형 키 정책은 계정에서 KMS 키를 생성한 AWS 서비스에 의해 생성되고 관리됩니다. AWS 소유 키의 키 정책은 변경할 수 없습니다.

- 키 정책 AWS 계정 에서 IAM 사용자, IAM 역할 및를 추가하거나 제거하고 해당 보안 주체에 대해 허용되거나 거부되는 작업을 변경할 수 있습니다. 키 정책에서 보안 주체와 권한을 지정하는 방법에 대한 자세한 내용은 키 정책 단원을 참조하십시오.
- 키 정책에 IAM 그룹을 추가할 수 없지만 여러 명의 IAM 사용자 및 IAM 역할을 추가할 수 있습니다. 자세한 내용은 여러 IAM 보안 주체가 KMS 키에 액세스하도록 허용 단원을 참조하십시오.
- 키 정책에 외부 AWS 계정 를 추가하는 경우 외부 계정의 IAM 정책도 사용하여 해당 계정의 IAM 사용자, 그룹 또는 역할에 권한을 부여해야 합니다. 자세한 내용은 다른 계정의 사용자가 KMS를 사용하도록 허용 단원을 참조하십시오.
- 결과 키 정책 문서는 32KB(32,768바이트)를 초과할 수 없습니다.

키 정책 변경 방법

세 가지 다른 방법을 이용해 키 정책을 변경할 수 있으며 이들 방법에 대해서는 다음 섹션에서 설명합니다.

주제

- AWS Management Console 기본 보기 사용
- AWS Management Console 정책 보기 사용
- AWS KMS API 사용

AWS Management Console 기본 보기 사용

콘솔을 이용해 기본 보기라는 그래픽 인터페이스로 키 정책을 변경할 수 있습니다.

다음 절차가 콘솔에서 본 것과 일치하지 않는 경우, 이 키 정책이 콘솔에서 생성되지 않았거나 키 정책이 콘솔의 기본 보기가 지원하지 않는 방식으로 수정되었음을 뜻합니다. 이 경우 AWS Management Console 정책 보기 사용 또는 AWS KMS API 사용에 설명된 단계를 따릅니다.

1. AWS KMS 콘솔 사용에서 설명한 대로 고객 관리형 키 정책을 확인합니다. (의 키 정책은 변경할 수 없습니다 AWS 관리형 키.)
2. 변경하려는 내용을 결정합니다.
   - 키 관리자를 추가하거나 제거하고, 키 관리자에게 KMS 키 삭제를 허용하거나 금지하려면, 페이지의 키 관리자 섹션에 있는 컨트롤을 사용합니다. 키 관리자는 활성화 및 비활성화, 키 정책 설정 및 키 교체 활성화를 포함해 KMS 키를 관리합니다.

- 키 사용자를 추가 또는 제거하고 외부에서 KMS 키를 AWS 계정 사용하도록 허용하거나 허용하지 않으려면 페이지의 키 사용자 섹션에 있는 컨트롤을 사용합니다. 키 사용자는 암호화, 암호 해독, 재암호화 및 데이터 키 생성 같은 암호화 작업에서 KMS 키를 사용할 수 있습니다.

AWS Management Console 정책 보기 사용

콘솔을 이용해 콘솔의 정책 보기에서 키 정책 문서를 변경할 수 있습니다.

1. AWS KMS 콘솔 사용에서 설명한 대로 고객 관리형 키 정책을 확인합니다. (의 키 정책은 변경할 수 없습니다 AWS 관리형 키.)
2. 키 정책 섹션에서 정책 보기로 전환을 선택합니다.
3. 편집을 선택합니다.
4. 변경하려는 내용을 결정합니다.
   - 새 문을 추가하려면 새 문 추가를 선택하세요. 그런 다음 새 키 정책 문에 대해 문 작성 도구 패널에 나열된 옵션에서 작업, 위탁자, 조건을 선택하거나 정책 문 요소를 직접 입력할 수 있습니다.
   - 키 정책에서 문을 제거하려면 해당 문을 선택한 다음 제거를 선택하세요. 선택한 정책 문을 검토하고 삭제할 것인지 확인하세요. 해당 문을 제거하는 작업을 진행하지 않으려면 취소를 선택하세요.
   - 기존 키 정책 문을 수정하려면 해당 문을 선택하세요. 그런 다음 문 빌더 패널을 사용하여 수정하려는 특정 요소를 선택하거나 문을 수동으로 편집할 수 있습니다.
5. 변경 사항 저장을 선택합니다.

AWS KMS API 사용

PutKeyPolicy 작업을 사용하여에서 KMS 키의 키 정책을 변경할 수 있습니다 AWS 계정. 다른 AWS 계정의 KMS 키에는 이 API를 사용할 수 없습니다.

1. GetKeyPolicy 작업을 이용해 기존 키 정책 문서를 검색한 다음 이 정책 문서를 파일에 저장합니다. 다중 프로그래밍 언어로 작성된 샘플 코드는 AWS SDK 또는 CLI와 `GetKeyPolicy` 함께 사용 단원을 참조하십시오.
2. 원하는 텍스트 편집기에서 키 정책 문서를 열고 편집한 후 파일을 저장합니다.
3. PutKeyPolicy 작업을 이용해 KMS 키에 업데이트된 키 정책 문서를 적용합니다. 다중 프로그래밍 언어로 작성된 샘플 코드는 AWS SDK 또는 CLI와 `PutKeyPolicy` 함께 사용 단원을 참조하십시오.

한 KMS 키에서 다른 KMS 키로 키 정책을 복사하는 예는 AWS CLI 명령 참조의 GetKeyPolicy 예제를 참조하십시오.

키 정책의 AWS 서비스에 대한 권한

많은 AWS 서비스는 AWS KMS keys 를 사용하여 관리하는 리소스를 보호합니다. 서비스가 AWS 소유 키 또는 AWS 관리형 키를 사용하는 경우, 서비스가 이러한 KMS 키에 대한 키 정책을 설정하고 유지 관리합니다.

그러나 고객 관리형 키와 함께 AWS 서비스를 사용하는 경우, 사용자가 키 정책을 설정하고 유지 관리합니다. 이 키 정책은 사용자를 대신하여 리소스를 보호하는 데 필요한 최소 권한을 서비스에 허용해야 합니다. 서비스에 필요한 권한만 제공하는 최소 권한의 원칙을 따르는 것이 좋습니다. 서비스에 필요한 권한이 무엇인지 파악하고 AWS 전역 조건 키 및 AWS KMS 조건 키를 사용하여 권한을 구체화하면 이 작업을 효과적으로 수행할 수 있습니다.

고객 관리형 키에서 서비스에 필요한 권한을 찾으려면 서비스에 대한 암호화 문서를 참조하세요. 다음 목록에는 일부 서비스 문서에 대한 링크가 포함되어 있습니다.

- AWS CloudTrail 권한 - CloudTrail에 대한 AWS KMS 키 정책 구성
- Amazon Elastic Block Store 권한 - Amazon EC2 사용 설명서 및 Amazon EC2 사용 설명서
- AWS Lambda 권한 - Lambda의 저장 데이터 암호화
- Amazon Q 권한 - Amazon Q의 데이터 암호화
- Amazon Relational Database Service 권한 - AWS KMS 키 관리
- AWS Secrets Manager 권한 - KMS 키 사용 권한 허용
- Amazon Simple Queue Service 권한 - Amazon SQS 키 관리

에서 IAM 정책 사용 AWS KMS

IAM 정책을 키 정책, 권한 부여 및 VPC 엔드포인트 정책과 함께 사용하여 AWS KMS keys 에 대한 액세스를 제어할 수 있습니다 AWS KMS.

> **Note**
>
> IAM 정책을 사용하여 KMS 키에 대한 액세스를 제어하려면 KMS 키의 키 정책에서 계정에 IAM 정책 사용 권한을 부여해야 합니다. 특히 키 정책에는 IAM 정책을 활성화하는 정책 설명이 포함되어야 합니다.

이 섹션에서는 IAM 정책을 사용하여 AWS KMS 작업에 대한 액세스를 제어하는 방법을 설명합니다. IAM에 대한 일반적인 내용은 IAM 사용 설명서를 참조하십시오.

모든 KMS 키에는 키 정책이 있어야 합니다. IAM 정책은 선택 사항입니다. IAM 정책을 사용하여 KMS 키에 대한 액세스를 제어하려면 KMS 키의 키 정책에서 계정에 IAM 정책 사용 권한을 부여해야 합니다. 특히 키 정책에는 IAM 정책을 활성화하는 정책 설명이 포함되어야 합니다.

IAM 정책은 모든 AWS KMS 작업에 대한 액세스를 제어할 수 있습니다. 키 정책과 달리 IAM 정책은 여러 KMS 키에 대한 액세스를 제어하고 여러 관련 AWS 서비스의 작업에 대한 권한을 제공할 수 있습니다. 그러나 IAM 정책은 CreateKey와 같은 작업에 대한 액세스를 제어하는 데 특히 유용합니다. 여기에는 특정 KMS 키가 포함되어 있지 않기 때문에 키 정책으로는 제어할 수 없습니다.

Amazon Virtual Private Cloud(VPC) 엔드포인트를 AWS KMS 통해에 액세스하는 경우 VPC 엔드포인트 정책을 사용하여 엔드포인트를 사용할 때 AWS KMS 리소스에 대한 액세스를 제한할 수도 있습니다. 예를 들어 VPC 엔드포인트를 사용하는 경우의 보안 주체만 고객 관리형 키에 액세스 AWS 계정 하도록 허용할 수 있습니다. 자세한 내용은 VPC 엔드포인트 정책을 참조하세요.

JSON 정책 문서 작성 및 형식 지정에 대한 도움말은 IAM 사용 설명서의 IAM JSON 정책 참조 섹션을 확인하세요.

IAM 정책을 다음과 같이 사용할 수 있습니다.

- 페더레이션 또는 크로스 계정 권한의 역할에 권한 정책 연결 – IAM 역할에 IAM 정책을 연결하여 자격 증명 연동을 활성화하거나, 크로스 계정 권한을 허용하거나, EC2 인스턴스에서 실행되는 애플리케이션에 권한을 부여할 수 있습니다. IAM 역할의 다양한 사용 사례에 대한 자세한 내용은 IAM 사용 설명서의 IAM 역할을 참조하십시오.
- 사용자 또는 그룹에 권한 정책 연결 – 사용자 또는 사용자 그룹이 AWS KMS 작업을 호출하도록 허용하는 정책을 연결할 수 있습니다. 그러나 IAM 모범 사례에서는 가능한 경우 IAM 역할과 같은 임시 보안 인증 정보가 있는 ID를 사용할 것을 권장합니다.

다음 예제에서는 AWS KMS 권한이 있는 IAM 정책을 보여줍니다. 이 정책은 연결되는 IAM 자격 증명이 모든 KMS 키와 별칭을 나열하도록 허용합니다.

JSON

```
{
  "Version":"2012-10-17",
```

```
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "kms:ListKeys",
      "kms:ListAliases"
    ],
    "Resource": "*"
  }
}
```

모든 IAM 정책처럼 이 정책에는`Principal` 요소가 없습니다. IAM 정책을 IAM ID에 연결할 경우 해당 ID는 정책에 지정된 권한을 얻습니다.

모든 AWS KMS API 작업과 해당 작업이 적용되는 리소스를 보여주는 표는 섹션을 참조하세요권한 참조.

여러 IAM 보안 주체가 KMS 키에 액세스하도록 허용

키 정책에서 IAM 그룹은 유효한 보안 주체가 아닙니다. 여러 사용자 및 역할이 KMS 키에 액세스하도록 허용하려면 다음 중 한 방법을 사용합니다.

- IAM 역할을 키 정책의 보안 주체로 사용하세요. 필요한 경우 여러 명의 권한 있는 사용자가 해당 역할을 맡을 수 있습니다. 자세한 내용은 IAM 사용 설명서에서 IAM 역할을 참조하세요.

  키 정책에 여러 IAM 사용자를 나열할 수 있지만 이 방법을 따르는 경우에는 인증된 사용자 목록이 변경될 때마다 키 정책을 업데이트하기 때문에 권장되지 않습니다. 또한 IAM 모범 사례는 장기 보안 인증 정보가 있는 IAM 사용자의 사용을 장려하지 않습니다. 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.
- IAM 정책을 사용하여 IAM 그룹에 권한을 부여하세요. 이 작업을 수행하기 위해서는 키 정책에 IAM 정책이 KMS 키에 대한 액세스를 허용하도록 하는 문이 포함되어 있는지 확인하고, KMS 키에 대한 액세스를 허용하는 IAM 정책을 생성한 다음, 해당 정책을 권한이 있는 IAM 사용자가 포함된 IAM 그룹에 연결합니다. 이 방법을 사용하면 권한있는 사용자 목록이 변경되어도 정책을 변경할 필요가 없습니다. 해당 IAM 그룹에서 사용자를 제거하거나 추가하기만 하면 됩니다. 자세한 내용은 IAM 사용 설명서의 IAM 사용자 그룹을 참조하세요.

AWS KMS 키 정책과 IAM 정책이 함께 작동하는 방식에 대한 자세한 내용은 섹션을 참조하세요AWS KMS 권한 문제 해결.

IAM 정책 모범 사례

에 대한 액세스를 보호하는 AWS KMS keys 것은 모든 AWS 리소스의 보안에 매우 중요합니다. KMS 키는 AWS 계정에서 가장 중요한 리소스를 보호하는 데 사용됩니다. 따라서 KMS 키에 대한 액세스를 제어하는 키 정책, IAM 정책, 권한 부여 및 VPC 엔드포인트 정책을 설계하는 데 시간을 할애하세요.

KMS 키에 대한 액세스를 제어하는 IAM 정책 설명에서 최소 권한 원칙을 사용합니다. IAM 보안 주체에 사용하거나 관리해야 하는 KMS 키에만 필요한 권한만 부여합니다.

다음 모범 사례는 AWS KMS 키 및 별칭에 대한 액세스를 제어하는 IAM 정책에 적용됩니다. 일반적인 IAM 정책 모범 사례 지침은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.

키 정책 사용

가능하면 다른 AWS 계정에 있는 키를 포함해 여러 KMS 키에 적용할 수 있는 IAM 정책보다는 하나의 KMS 키에 영향을 주는 키 정책에 권한을 제공합니다. 이는 kms:PutKeyPolicy 및 kms:ScheduleKeyDeletion과 같은 민감한 권한뿐만 아니라 데이터 보호 방법을 결정하는 암호화 작업에도 특히 중요합니다.

CreateKey 권한 제한

필요한 보안 주체에게만 키 생성 권한(kms:CreateKey)을 부여합니다. KMS 키를 만드는 보안 주체도 키 정책을 설정하므로 자신이 만든 KMS 키를 사용하고 관리할 수 있는 권한을 자신과 다른 사용자에게 부여할 수 있습니다. 이 사용 권한을 할 때 정책 조건을 사용하여 제한하는 것이 좋습니다. 예를 들어, kms:KeySpec 조건을 사용하여 대칭 암호화 KMS 키에 대한 권한을 제한할 수 있습니다.

IAM 정책에서 KMS 키 지정

가장 좋은 방법은 정책 설명의 `Resource` 요소에 권한이 적용되는 각 KMS 키의 키 ARN을 지정하는 것입니다. 이 방법은 보안 주체에 필요한 KMS 키에 대한 사용 권한을 제한합니다. 예를 들어 이 `Resource` 요소는 보안 주체가 사용해야 하는 KMS 키만 나열합니다.

```
"Resource": [
    "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-ab0987654321"
]
```

KMS 키를 지정하는 것이 실용적이지 않은 경우와 같이 신뢰할 수 있는 AWS 계정 및 리전의 KMS 키에 대한 액세스를 제한하는 `Resource` 값을 사용합니다`arn:aws:kms:`*region*`:`*account*`:key/*`. 또는 AWS 계정와 같이 신뢰할 수 있는의 모든 리전(*)에서 KMS 키에 대한 액세스를 제한합니다`arn:aws:kms:*:`*account*`:key/*`.

키 ID, 별칭 이름 또는 별칭 ARN을 사용하여 IAM 정책의 `Resource` 필드에서 KMS 키를 나타낼 수 없습니다. 별칭 ARN을 지정하면 정책이 KMS 키가 아닌 별칭에 적용됩니다. 별칭에 대한 IAM 정책에 대한 자세한 내용은 별칭에 대한 액세스 제어 섹션을 참조하십시오.

IAM 정책에서 "Resource": "*" 사용 안 함

와일드카드 문자 (*)를 신중하게 사용하십시오. 키 정책에서 `Resource` 요소의 와일드카드 문자는 키 정책이 연결된 KMS 키를 나타냅니다. 그러나 IAM 정책에서는 `Resource` 요소(`"Resource": "*"`)의 와일드카드 문자만 보안 주체의 계정에 사용 권한이 AWS 계정 있는 모든의 모든 KMS 키에 권한을 적용합니다. 여기에는 다른의 KMS 키 AWS 계정와 보안 주체 계정의 KMS 키가 포함될 수 있습니다.

예를 들어 다른에서 KMS 키를 사용하려면 보안 주체 AWS 계정는 외부 계정에 있는 KMS 키의 키 정책 및 자체 계정에 있는 IAM 정책의 권한이 필요합니다. 임의의 계정이 KMS 키에 대한 AWS 계정 kms:Decrypt 권한을 부여했다고 가정합니다. 이 경우 역할에 모든 KMS 키(`"Resource": "*"`)에 대한 `kms:Decrypt` 권한을 부여하는 계정의 IAM 정책은 요구 사항의 IAM 부분을 충족합니다. 따라서 해당 역할을 수임할 수 있는 보안 주체는 이제 신뢰할 수 없는 계정의 KMS 키를 사용하여 암호문을 해독할 수 있습니다. 두 계정의 CloudTrail 로그에 해당 작업에 대한 항목이 표시됩니다.

특히 다음 API 작업을 허용하는 정책 설명에서 `"Resource": "*"`를 사용하지 마십시오. 이러한 작업은 다른의 KMS 키에서 호출할 수 있습니다 AWS 계정.

- DescribeKey
- GetKeyRotationStatus
- 암호화 작업(Encrypt, Decrypt, GenerateDataKey, GenerateDataKeyPair, GenerateDataKeyWithoutPlaintext, GenerateDataKeyPairWithoutPlaintext, GetPublicKey, ReEncrypt, Sign, Verify)
- CreateGrant, ListGrants, ListRetirableGrants, RetireGrant, RevokeGrant

"Resource": "*"를 사용하는 경우

IAM 정책에서 필요한 권한에 대해서만 `Resource` 요소의 와일드카드 문자를 사용하십시오. 다음 권한에만 `"Resource": "*"` 요소가 필요합니다.

- kms:CreateKey
- kms:GenerateRandom
- kms:ListAliases
- kms:ListKeys
- 사용자 지정 키 스토어 권한(예: kms:CreateCustomKeyStore 및 kms:ConnectCustomKeyStore)

> **Note**
>
> 별칭 작업에 대한 권한(kms:CreateAlias, kms:UpdateAlias, kms:DeleteAlias)은 별칭과 KMS 키에 연결되어야 합니다. IAM 정책에서 `"Resource": "*"`를 사용하여 별칭과 KMS 키를 나타내거나 `Resource` 요소에서 별칭과 KMS 키를 지정할 수 있습니다. 예제는 별칭에 대한 액세스 제어 섹션을 참조하십시오.

이 항목의 예에서는 KMS 키에 대한 IAM 정책을 설계하기 위한 추가 정보와 지침을 제공합니다. 모든 AWS 리소스에 대한 IAM 모범 사례는 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.

IAM 정책 설명에서 KMS 키 지정

IAM 정책을 사용하여 보안 주체가 KMS 키를 사용하거나 관리하도록 허용할 수 있습니다. KMS 키는 정책 설명의 `Resource` 요소에 지정됩니다.

- IAM 정책 설명에서 KMS 키를 지정하려면키 ARN을 사용해야 합니다. 키 ID, 별칭 이름 또는 별칭 ARN을 사용하여 IAM 정책 설명에서 KMS 키를 식별할 수 없습니다.

  예: `"Resource": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"`

  별칭을 기반으로 KMS 키에 대한 액세스를 제어하려면kms:RequestAlias 또는 kms:ResourceAliases 조건 키를 사용합니다. 자세한 내용은 에 대한 ABAC AWS KMS 섹션을 참조하십시오.

  CreateAlias, UpdateAlias 또는 DeleteAlias와 같은 별칭 작업에 대한 액세스를 제어하는 정책 설명에서만 별칭 ARN을 리소스로 사용합니다. 자세한 내용은 별칭에 대한 액세스 제어 섹션을 참조하세요.
- 계정 및 리전에서 여러 KMS 키를 지정하려면 키 ARN의 리전 또는 리소스 ID 위치에 와일드카드 문자(*)를 사용합니다.

  예를 들어 계정의 미국 서부(오레곤) 리전에 있는 모든 KMS 키를 지정하려면 `"Resource": "arn:aws:kms:us-west-2:111122223333:key/*"`를 사용합니다. 계정의 모든 리전에서 모든 KMS 키를 지정하려면 `"Resource": "arn:aws:kms:*:111122223333:key/*"`를 사용합니다.
- 모든 KMS 키를 표시하려면 와일드카드 문자(`"*"`)만 사용합니다. 특정 KMS 키, 즉 CreateKey, GenerateRandom, ListAliases 및 ListKeys를 사용하지 않는 작업에 이 형식을 사용합니다.

정책 설명을 작성할 때 모든 KMS 키에 대한 액세스 권한을 부여하는 대신 보안 주체가 사용해야 하는 KMS 키만 지정하는 것이 모범 사례입니다.

예를 들어 다음 IAM 정책 문은 위탁자가 정책 문의 `Resource` 요소에 나열된 KMS 키에 대해서만 DescribeKey, GenerateDataKey, Decrypt 작업을 직접적으로 호출하도록 허용합니다. 키 ARN으로 KMS 키를 지정하면 사용 권한이 지정된 KMS 키로만 제한됩니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "kms:DescribeKey",
      "kms:GenerateDataKey",
      "kms:Decrypt"
    ],
    "Resource": [
     "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
     "arn:aws:kms:us-
west-2:111122223333:key/01234abcd-12ab-34cd-56ef-1234567890ab"
    ]
  }
}
```

신뢰할 수 있는 특정의 모든 KMS 키에 권한을 적용하려면 리전 및 키 ID 위치에 와일드카드 문자(*)를 사용할 AWS 계정수 있습니다. 예를 들어 다음 정책 설명은 보안 주체가 두 개의 신뢰할 수 있는 예제 계정의 모든 KMS 키에 대해 지정된 작업을 호출하도록 허용합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "kms:DescribeKey",
      "kms:GenerateDataKey",
```

```
      "kms:GenerateDataKeyPair"
    ],
    "Resource": [
      "arn:aws:kms:*:111122223333:key/*",
      "arn:aws:kms:*:444455556666:key/*"
    ]
  }
}
```

Resource 요소에서 와일드카드 문자("*")만 사용할 수도 있습니다. 계정에 사용 권한이 있는 모든 KMS 키에 대한 액세스를 허용하므로 특정 KMS 키가 없는 작업 및 Deny 문에 주로 권장됩니다. 덜 민감한 읽기 전용 작업만 허용하는 정책 설명에서도 이 기능을 사용할 수 있습니다. AWS KMS 작업에 특정 KMS 키가 포함되어 있는지 확인하려면에 있는 테이블의 리소스 열에서 KMS 키 값을 찾습니다the section called “권한 참조”.

예를 들어 다음 정책 설명은Deny 효과를 사용하여 보안 주체가 모든 KMS 키에 대해 지정된 작업을 사용하지 못하도록 합니다. Resource 요소에 와일드카드 문자를 사용하여 모든 KMS 키를 나타냅니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Deny",
    "Action": [
      "kms:CreateKey",
      "kms:PutKeyPolicy",
      "kms:CreateGrant",
      "kms:ScheduleKeyDeletion"
    ],
    "Resource": "*"
  }
}
```

다음 정책 설명에서는 와일드카드 문자만 사용하여 모든 KMS 키를 나타냅니다. 그러나 덜 민감한 읽기 전용 작업과 특정 KMS 키에 적용되지 않는 작업만 허용합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "kms:CreateKey",
      "kms:ListKeys",
      "kms:ListAliases",
      "kms:ListResourceTags"
    ],
    "Resource": "*"
  }
}
```

IAM 정책 예시

이 섹션에서는 다양한 AWS KMS 작업에 대한 권한을 허용하는 IAM 정책의 예를 제공합니다.

> **Important**
>
> 다음 정책에 포함된 권한 중 일부는 KMS 키의 키 정책에서도 허용하는 경우에만 허용됩니다. 자세한 내용은 권한 참조 단원을 참조하십시오.

JSON 정책 문서 작성 및 형식 지정에 대한 도움말은 IAM 사용 설명서의 IAM JSON 정책 참조를 참조하십시오.

예제

- 사용자가 AWS KMS 콘솔에서 KMS 키를 볼 수 있도록 허용
- 사용자가 KMS 키를 생성할 수 있도록 허용
- 사용자가 특정의 KMS 키를 사용하여 암호화 및 복호화하도록 허용 AWS 계정
- 사용자가 특정 및 리전의 모든 KMS 키를 사용하여 암호화 AWS 계정 및 복호화할 수 있도록 허용
- 사용자가 특정 KMS 키로 암호화와 해독을 수행하도록 허용
- 사용자가 KMS 키를 비활성화하거나 삭제하지 못하도록 차단

사용자가 AWS KMS 콘솔에서 KMS 키를 볼 수 있도록 허용

다음 IAM 정책은 사용자가 AWS KMS 콘솔에 대한 읽기 전용 액세스를 허용합니다. 이러한 권한이 있는 사용자는의 모든 KMS 키를 볼 수 AWS 계정있지만 KMS 키를 생성하거나 변경할 수는 없습니다.

AWS 관리형 키 및 고객 관리형 키 페이지에서 KMS 키를 보려면, 키에 태그나 별칭이 없더라도 위탁자에게 kms:ListKeys, kms:ListAliases 및 tag:GetResources 권한이 필요합니다. 나머지 사용 권한, 특히 kms:DescribeKey는 KMS 키 세부 정보 페이지에서 선택적 KMS 키 테이블 열 및 데이터를 보는 데 필요합니다. iam:ListUsers 및 iam:ListRoles 권한은 오류 없이 기본 보기에 키 정책을 표시하는 데 필요합니다. 사용자 지정 키 스토어 페이지의 데이터와 사용자 지정 키 저장소의 KMS 키에 대한 세부 정보를 보려면 보안 주체에 kms:DescribeCustomKeyStores 권한도 필요합니다.

사용자의 콜솔 액세스를 특정 KMS 키로 제한하면 콘솔에서 보이지 않는 각 KMS 키에 대한 오류를 표시합니다.

이 정책에는 두 가지 정책 설명이 포함됩니다. 첫 번째 정책 설명의 `Resource` 요소는 예제 AWS 계정의 모든 리전에 있는 모든 KMS 키에 대해 지정된 권한을 허용합니다. 콘솔 뷰어는 AWS KMS 콘솔이 보안 주체의 계정에 KMS 키만 표시하므로 추가 액세스가 필요하지 않습니다. 다른에서 KMS 키를 볼 수 있는 권한이 있는 경우에도 마찬가지입니다 AWS 계정. 나머지 AWS KMS 및 IAM 권한은 특정 KMS 키에 적용되지 않으므로 `"Resource": "*"` 요소가 필요합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "ReadOnlyAccessForAllKMSKeysInAccount",
      "Effect": "Allow",
      "Action": [
        "kms:GetPublicKey",
        "kms:GetKeyRotationStatus",
        "kms:GetKeyPolicy",
        "kms:DescribeKey",
        "kms:ListKeyPolicies",
        "kms:ListResourceTags",
        "tag:GetResources"
      ],
      "Resource": "arn:aws:kms:*:111122223333:key/*"
    },
    {
```

```
      "Sid": "ReadOnlyAccessForOperationsWithNoKMSKey",
      "Effect": "Allow",
      "Action": [
        "kms:ListKeys",
        "kms:ListAliases",
        "iam:ListRoles",
        "iam:ListUsers"
      ],
      "Resource": "*"
    }
  ]
}
```

사용자가 KMS 키를 생성할 수 있도록 허용

다음 IAM 정책은 사용자가 모든 유형의 KMS 키를 만들 수 있도록 허용합니다. `CreateKey` 작업에서 특정 AWS KMS 리소스(KMS 키 또는 별칭)를 사용하지 않기 `*` 때문에 `Resource` 요소의 값은 입니다.

사용자를 특정 유형의 KMS 키로 제한하려면 kms:KeySpec, kms:KeyUsage, kms:KeyOrigin 조건 키를 사용합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "kms:CreateKey",
    "Resource": "*"
  }
}
```

키를 만드는 보안 주체에게는 몇 가지 관련 사용 권한이 필요할 수 있습니다.

- kms:PutKeyPolicy — `kms:CreateKey` 권한이 있는 보안 주체는 KMS 키에 대한 초기 키 정책을 설정할 수 있습니다. 그러나, `CreateKey` 호출자는 KMS 키 정책을 변경할 수 있는 kms:PutKeyPolicy 권한이 있어야 하고, 권장되지 않는 `CreateKey`의 `BypassPolicyLockoutSafetyCheck` 파라미터를 지정해야 합니다. `CreateKey` 호출자는 IAM 정책에서 KMS 키에 대한 `kms:PutKeyPolicy` 권한을 가져오거나 생성 중인 KMS 키의 키 정책에 이 권한을 포함할 수 있습니다.

- kms:TagResource — `CreateKey` 작업 중에 KMS 키에 태그를 추가하려면 `CreateKey` 호출자는 IAM 정책에서 kms:TagResource 권한이 있어야 합니다. 새 KMS 키의 키 정책에 이 권한을 포함시키는 것만으로는 충분하지 않습니다. 그러나, `CreateKey` 호출자가 초기 키 정책에 `kms:TagResource`를 포함하는 경우 KMS 키가 생성된 후 별도의 호출에서 태그를 추가할 수 있습니다.
- kms:CreateAlias - AWS KMS 콘솔에서 KMS 키를 생성하는 보안 주체는 KMS 키 및 별칭에 대해 kms:CreateAlias 권한이 있어야 합니다. (콘솔은 `CreateKey`, `CreateAlias`를 각각 한 번씩, 두 번 호출합니다.) IAM 정책에서 별칭 권한을 제공해야 합니다. 키 정책 또는 IAM 정책에서 KMS 키 권한을 제공할 수 있습니다. 자세한 내용은 별칭에 대한 액세스 제어 섹션을 참조하세요.

외에도 다음 IAM 정책은 `kms:CreateKey`의 모든 KMS 키에 대한 `kms:TagResource` 권한 AWS 계정 과 계정의 모든 별칭에 대한 `kms:CreateAlias` 권한을 제공합니다. 또한 IAM 정책에서만 제공할 수 있는 몇 가지 유용한 읽기 전용 권한도 포함되어 있습니다.

이 IAM 정책에는 키 정책에서 설정할 수 있는 권한 또는 기타 `kms:PutKeyPolicy` 권한이 포함되어 있지 않습니다. 하나의 KMS 키에만 적용되는 키 정책에 이러한 권한을 설정하는 것이 모범 사례입니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "IAMPermissionsForParticularKMSKeys",
      "Effect": "Allow",
      "Action": "kms:TagResource",
      "Resource": "arn:aws:kms:*:111122223333:key/*"
    },
    {
      "Sid": "IAMPermissionsForParticularAliases",
      "Effect": "Allow",
      "Action": "kms:CreateAlias",
      "Resource": "arn:aws:kms:*:111122223333:alias/*"
    },
    {
      "Sid": "IAMPermissionsForAllKMSKeys",
      "Effect": "Allow",
      "Action": [
```

```
        "kms:CreateKey",
        "kms:ListKeys",
        "kms:ListAliases"
      ],
      "Resource": "*"
    }
  ]
}
```

사용자가 특정의 KMS 키를 사용하여 암호화 및 복호화하도록 허용 AWS 계정

다음 IAM 정책은 사용자가 AWS 계정 111122223333의 모든 KMS 키로 데이터를 암호화하고 복호화할 수 있도록 허용합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "kms:Encrypt",
      "kms:Decrypt"
    ],
    "Resource": "arn:aws:kms:*:111122223333:key/*"
  }
}
```

사용자가 특정 및 리전의 모든 KMS 키를 사용하여 암호화 AWS 계정 및 복호화할 수 있도록 허용

다음 IAM 정책은 사용자가 미국 서부(오레곤) 리전의에 있는 AWS 계정 111122223333 모든 KMS 키로 데이터를 암호화하고 복호화할 수 있도록 허용합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
```

```
    "Effect": "Allow",
    "Action": [
      "kms:Encrypt",
      "kms:Decrypt"
    ],
    "Resource": [
      "arn:aws:kms:us-west-2:111122223333:key/*"
    ]
  }
}
```

사용자가 특정 KMS 키로 암호화와 해독을 수행하도록 허용

다음 IAM 정책은 사용자가 `Resource` 요소에 지정된 두 개의 KMS 키를 사용하여 데이터를 암호화하고 해독할 수 있도록 허용합니다. IAM 정책 설명에서 KMS 키를 지정할 때 KMS 키의 키 ARN을 사용해야 합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "kms:Encrypt",
      "kms:Decrypt"
    ],
    "Resource": [
      "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
      "arn:aws:kms:us-west-2:111122223333:key/01234abc-d12a-b34c-d56e-
f1234567890a'"
    ]
  }
}
```

사용자가 KMS 키를 비활성화하거나 삭제하지 못하도록 차단

다음 IAM 정책은 다른 IAM 정책이나 키 정책에 관련 권한이 있더라도 사용자가 KMS 키를 비활성화하거나 삭제하는 것을 금지합니다. 권한을 명시적으로 거부하는 정책이 동일한 권한을 명시적으로 허용

하는 정책을 비롯한 다른 모든 정책을 무시합니다. 자세한 내용은 AWS KMS 권한 문제 해결 단원을 참조하십시오.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Deny",
    "Action": [
      "kms:DisableKey",
      "kms:ScheduleKeyDeletion"
    ],
    "Resource": "*"
  }
}
```

의 리소스 제어 정책 AWS KMS

리소스 제어 정책(RCPs)은 조직의 AWS 리소스에 대한 예방 제어를 적용하는 데 사용할 수 있는 조직 정책의 한 유형입니다. RCPs 사용하면 AWS 리소스에 대한 외부 액세스를 대규모로 중앙에서 제한할 수 있습니다. RCP는 서비스 제어 정책(SCP)을 보완합니다. SCPs를 사용하여 조직의 IAM 역할 및 사용자에 대한 최대 권한을 중앙에서 설정할 수 있지만, RCPs를 사용하여 조직의 AWS 리소스에 대한 최대 권한을 중앙에서 설정할 수 있습니다.

조직의 고객 관리형 KMS 키에 대한 권한을 관리하기 위해 RCP를 사용할 수 있습니다. RCP만으로는 고객 관리형 KMS 키에 대한 권한을 부여하기에 충분하지 않습니다. RCP는 어떤 권한도 부여하지 않습니다. RCP는 권한 가드레일을 정의하거나 영향을 받는 계정의 리소스에 대해 주체가 수행할 수 있는 작업에 제한을 설정합니다. 관리자는 실제로 권한을 부여하기 위해 IAM 역할 또는 사용자에게 자격 증명 기반 정책을 연결하거나 키 정책을 연결해야 합니다.

> **Note**
>
> 조직의 리소스 제어 정책은 AWS 관리형 키에 적용되지 않습니다.
> AWS 관리형 키 는 AWS 서비스에서 사용자를 대신하여 생성, 관리 및 사용되므로 해당 권한을 변경하거나 관리할 수 없습니다.

자세히 알아보기

- RCP에 대한 일반적인 정보를 더 알아보려면 AWS Organizations 사용 설명서의 리소스 제어 정책을 참조하세요.
- RCP를 정의하는 방법(예제 포함)에 대한 자세한 내용은 AWS Organizations 사용 설명서의 RCP 구문을 참조하세요.

다음 예시는 조직 내에서 고객 관리형 키에 대한 외부 위탁자의 액세스를 방지하기 위해 RCP를 사용하는 방법을 보여줍니다. 이 정책은 예시일 뿐이며 고유한 비즈니스 및 보안 요구 사항에 맞게 조정해야 합니다. 예를 들어 비즈니스 파트너의 액세스를 허용하도록 정책을 사용자 지정해야 할 수 있습니다. 자세한 내용은 데이터 경계 정책 예제 리포지토리를 참조하세요.

> **Note**
>
> `Action` 권한은 `kms:RetireGrant` 요소가 와일드카드(*)로 지정된 경우에도 RCP에서 유효하지 않습니다.
>
> `kms:RetireGrant` 권한을 결정하는 방법에 대한 자세한 내용은 권한 부여 사용 중지 및 취소 페이지를 참조하세요.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "RCPEnforceIdentityPerimeter",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "kms:*",
            "Resource": "*",
            "Condition": {
                "StringNotEqualsIfExists": {
                    "aws:PrincipalOrgID": "my-org-id"
                },
                "Bool": {
                    "aws:PrincipalIsAWSService": "false"
                }
            }
```

```
        }
    ]
}
```

다음 예제 RCP에서는 요청이 조직에서 시작된 경우에만 AWS 서비스 보안 주체가 고객 관리형 KMS 키에 액세스할 수 있도록 요구합니다. 이 정책은 `aws:SourceAccount`이 있는 요청에만 해당 통제를 적용합니다. 이렇게 하면 `aws:SourceAccount`를 사용할 필요가 없는 서비스 통합이 영향을 받지 않습니다. 요청 컨텍스트에 `aws:SourceAccount`가 있는 경우 `Null` 조건이 `true`로 평가되어 `aws:SourceOrgID` 키가 적용됩니다.

혼동된 대리자 문제에 관한 자세한 내용은 IAM 사용 설명서의 혼동된 대리자 문제를 참조하세요.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "RCPEnforceConfusedDeputyProtection",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "kms:*",
            "Resource": "*",
            "Condition": {
                "StringNotEqualsIfExists": {
                    "aws:SourceOrgID": "my-org-id"
                },
                "Bool": {
                    "aws:PrincipalIsAWSService": "true"
                },
                "Null": {
                    "aws:SourceAccount": "false"
                }
            }
        }
    ]
}
```

의 권한 부여 AWS KMS

권한 부여는 보안 AWS 주체 또는 AWS 서비스 보안 주체가 암호화 작업에서 KMS 키를 사용할 수 있도록 허용하는 정책 도구입니다. 또한 KMS 키(DescribeKey)를 보고 권한 부여를 생성 및 관리할 수 있습니다. KMS 키에 대한 액세스 권한을 부여할 때 키 정책 및 IAM 정책과 함께 권한 부여가 고려됩니다. 권한 부여는 키 정책이나 IAM 정책을 변경하지 않고 권한을 생성하고 삭제할 수 있기 때문에 임시 권한에 자주 사용됩니다.

권한 부여는 저장 데이터를 암호화 AWS KMS 하기 위해와 통합되는 AWS 서비스에서 일반적으로 사용됩니다. 서비스는 계정의 사용자를 대신하여 권한 부여를 만들고, 권한을 사용하고, 작업이 완료되는 즉시 권한 부여를 폐기합니다. AWS 서비스에서 권한 부여를 사용하는 방법에 대한 자세한 내용은 서비스 사용 설명서 또는 개발자 안내서의 저장 시 암호화 주제를 참조하세요.

권한 부여는 매우 유연하고 유용한 액세스 제어 메커니즘입니다. KMS 키에 대한 권한 부여를 생성하면 권한 부여에 지정된 모든 조건이 충족되는 경우 권한 부여를 통해 피부여자 보안 주체가 KMS 키에 대해 지정된 권한 부여 작업을 호출할 수 있습니다.

- 각 권한 부여를 통해 정확히 하나의 KMS 키에 액세스할 수 있습니다. 다른 AWS 계정에서 KMS 키에 대한 권한 부여를 생성할 수 있습니다.
- 권한 부여는 KMS 키에 대한 액세스를 허용할 수 있지만 액세스를 거부할 수는 없습니다.
- 각 권한 부여에는 피부여자 보안 주체(IAM 자격 증명) 또는 피부여자 서비스 보안 주체(서비스 보안 주체) 중 하나일 수 있는 피부여자 하나가 AWS 있습니다. AWS GranteePrincipal 또는 GranteeServicePrincipal 중 하나만 지정해야 하지만 두 가지 모두 지정해서는 안 됩니다.
- 권한 부여는 권한 부여 작업만 허용할 수 있습니다. 권한 부여 작업은 권한 부여의 KMS 키로 지원되어야 합니다. 지원되지 않는 작업을 지정하면 CreateGrant 요청이 ValidationError 예외와 함께 실패합니다.
- 피부여자 보안 주체는 권한이 키 정책 또는 IAM 정책에서 제공되는 경우와 마찬가지로 권한 부여를 지정하지 않고 권한 부여가 부여하는 권한을 사용할 수 있습니다. 그러나 AWS KMS API는 최종 일관성 모델을 따르므로 권한 부여를 생성, 사용 중지 또는 취소할 때 변경 사항이 전체에 적용되기 전에 잠시 지연될 수 있습니다 AWS KMS. 권한 부여에서 권한을 즉시 사용하려면 권한 부여 토큰을 사용하십시오.
- 권한이 부여된 보안 주체는 권한 부여를 삭제(사용 중지 또는 취소)할 수 있습니다. 권한 부여를 삭제하면 권한 부여가 허용한 모든 권한이 제거됩니다. 권한 부여를 실행 취소하기 위해 추가하거나 제거할 정책을 파악할 필요가 없습니다.
- AWS KMS 는 각 KMS 키에 대한 권한 부여 수를 제한합니다. 자세한 내용은 KMS 키당 권한 부여: 50,000.

권한 부여를 생성할 때와 다른 사람에게 권한 부여 생성 권한을 부여할 때는 주의해야 합니다. 권한 부여를 생성할 수 있는 권한은 정책을 설정할 수 있는 kms:PutKeyPolicy 권한을 사용하여 정책을 설정할 수 있습니다.

- KMS 키(`kms:CreateGrant`)에 대한 권한 부여를 생성할 권한이 있는 사용자는 권한 부여를 사용하여 AWS 서비스를 포함한 사용자 및 역할이 KMS 키를 사용하도록 허용할 수 있습니다. 보안 주체는 자신의 자격 증명 AWS 계정 이거나 다른 계정 또는 조직의 자격 증명일 수 있습니다.
- 권한 부여는 AWS KMS 작업의 하위 집합만 허용할 수 있습니다. 권한 부여를 사용하여 보안 주체가 KMS 키를 보고, 암호화 작업에 사용하고, 권한 부여를 만들고 사용 중지하도록 허용할 수 있습니다. 자세한 내용은 권한 부여 작업 단원을 참조하세요. 권한 부여 제약 조건을 사용하여 권한 부여의 사용 권한을 제한할 수 있습니다.
- 보안 주체는 키 정책 또는 IAM 정책에서 권한 부여를 생성할 수 있는 권한을 가질 수 있습니다. 정책에서 `kms:CreateGrant` 권한을 받은 보안 주체는 KMS 키에 대한 권한 부여 작업에 대한 권한을 생성할 수 있습니다. 이러한 보안 주체는 키에 대해 부여한 권한이 필요하지 않습니다. 정책에서 `kms:CreateGrant` 권한을 허용하면 정책 조건을 사용하여 이 권한을 제한할 수 있습니다.
- 보안 주체는 권한 부여에서 권한 부여를 만들 수 있는 권한을 가져올 수도 있습니다. 이러한 위탁자는 정책에서 다른 권한이 있더라도 부여된 권한만 위임할 수 있습니다. 자세한 내용은 CreateGrant 권한 부여을 참조하세요.

권한 부여 개념

권한 부여를 효과적으로 사용하려면 AWS KMS 가 사용하는 용어와 개념을 이해해야 합니다.

권한 부여 제약

권한 부여의 권한을 제한하는 조건입니다.는 두 가지 유형의 권한 부여 제약 조건을 AWS KMS 지원합니다.

- 암호화 컨텍스트 제약 - 암호화 작업 요청에 포함된 암호화 컨텍스트를 기반으로 권한을 제한합니다. 이 제약 조건은 대칭 암호화 KMS 키에서만 작동합니다.
- SourceArn 제약 - 특정 AWS 리소스를 대신하여 이루어진 요청에 대한 권한 부여 권한을 제한합니다. 이렇게 하면 aws:SourceArn 전역 조건 키가 권한 부여에 효과적으로 배치됩니다. 이 제약 조건은 모든 유형의 KMS 키에서 작동합니다.

자세한 내용은 권한 부여 제약 사용을 참조하세요.

권한 부여 ID

KMS 키에 대한 권한 부여의 고유 식별자입니다. 키 식별자와 함께 권한 부여 ID를 사용하여 RetireGrant 또는 RevokeGrant 요청에서 권한 부여를 식별할 수 있습니다.

권한 부여 작업

권한 부여에서 허용할 수 있는 AWS KMS 작업입니다. 다른 작업을 지정하면 CreateGrant 요청이 `ValidationError` 예외와 함께 실패합니다. 권한 부여 토큰을 수락하는 작업이기도 합니다. 이러한 권한에 대한 자세한 내용은 AWS KMS 권한 섹션을 참조하세요.

이러한 권한 부여 작업은 실제로 작업을 사용할 수 있는 권한을 나타냅니다. 따라서 `ReEncrypt` 작업의 경우 `ReEncryptFrom`, `ReEncryptTo` 또는 둘 다 `ReEncrypt*`를 지정할 수 있습니다.

권한 부여 작업은 다음과 같습니다.

- 암호화 작업
  - Decrypt
  - DeriveSharedSecret
  - 암호화
  - GenerateDataKey
  - GenerateDataKeyPair
  - GenerateDataKeyPairWithoutPlaintext
  - GenerateDataKeyWithoutPlaintext
  - GenerateMac
  - ReEncryptFrom
  - ReEncryptTo
  - Sign
  - Verify
  - VerifyMac
- 기타 작업
  - CreateGrant
  - DescribeKey
  - GetPublicKey
  - RetireGrant

허용한 권한 부여 작업은 권한 부여의 KMS 키로 지원되어야 합니다. 지원되지 않는 작업을 지정하면 CreateGrant 요청이 `ValidationError` 예외와 함께 실패합니다. 예를 들어 대칭 암호화 KMS 키에 대한 권한 부여는 Sign, Verify, `GenerateMac` 또는 `VerifyMac` 작업을 허용하지 않습니다. 비대칭 KMS 키에 대한 권한 부여는 데이터 키 또는 데이터 키 페어를 생성하는 작업을 허용하지 않습니다.

권한 부여 토큰

AWS KMS API는 최종 일관성 모델을 따릅니다. 권한 부여를 생성할 때, 변경 사항이 AWS KMS전체에 적용되기까지 잠깐의 지연이 있을 수 있습니다. 변경 사항이 시스템 전체에 전파되는 데에는 일반적으로 몇 초도 걸리지 않지만 경우에 따라 몇 분 정도 걸릴 수도 있습니다. 시스템에 전체에 완전히 전파하기 전에 권한 부여를 사용하려고 하면 액세스 거부 오류가 발생할 수 있습니다. 권한 부여 토큰을 사용하면 권한 부여를 참조하고 즉시 권한 부여 권한을 사용할 수 있습니다.

권한 부여 토큰은 권한 부여를 나타내는 고유하고 암호화되지 않은 가변 길이 base64 인코딩 문자열입니다. 권한 부여 토큰을 사용하여 권한 부여 작업에서 권한 부여를 식별할 수 있습니다. 그러나 토큰 값은 해시 다이제스트이므로 권한 부여에 대한 세부 정보는 공개되지 않습니다.

권한 부여 토큰은 권한 부여가 AWS KMS전체에 완전히 전파될 때까지만 사용하도록 설계되었습니다. 그 후, 피부여자 보안 주체는 권한 부여 토큰이나 권한 부여에 대한 다른 증거를 제공하지 않고 권한 부여의 권한을 사용할 수 있습니다. 권한 부여 토큰은 언제든지 사용할 수 있지만 권한 부여가 최종적으로 일관되면는 권한 부여 토큰이 아닌 권한을 결정하는 데 권한 부여를 AWS KMS 사용합니다.

예를 들어 다음 명령은 GenerateDataKey 작업을 사용합니다. 권한 부여 토큰을 사용하여 지정된 KMS 키에서 `GenerateDataKey`를 호출할 수 있는 권한을 호출자(피부여자 보안 주체)에게 부여하는 권한 부여를 나타냅니다.

```
$ aws kms generate-data-key \
        --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
        --key-spec AES_256 \
        --grant-token $token
```

권한 부여 토큰을 사용하여 권한 부여를 관리하는 작업에서 권한 부여를 식별할 수도 있습니다. 예를 들어, 사용 중지 보안 주체는 RetireGrant 작업을 호출할 때 권한 부여 토큰을 사용할 수 있습니다.

```
$ aws kms retire-grant \
        --grant-token $token
```

`CreateGrant`는 권한 부여 토큰을 반환하는 유일한 작업입니다. CreateGrant AWS KMS 작업에 대한 다른 작업 또는 CloudTrail 로그 이벤트에서는 권한 부여 토큰을 가져올 수 없습니다. ListGrants 및 ListRetirableGrants 작업은 권한 부여 ID를 반환하지만 권한 부여 토큰은 아닙니다.

자세한 내용은 권한 부여 토큰 사용 섹션을 참조하세요.

피부여자 보안 주체

권한 부여에 지정된 권한을 가져오는 AWS 보안 주체(IAM 자격 증명)입니다.

피부여자 보안 주체는 AWS 계정 (루트), IAM 사용자, IAM 역할, 페더레이션 역할 또는 사용자 또는 수임된 역할 사용자를 포함한 모든 보안 AWS 주체일 수 있습니다. 피부여자 보안 주체는 KMS 키와 동일한 계정 또는 다른 계정에 있을 수 있습니다. 그러나 피부여자 보안 주체는 서비스 주체, IAM 그룹 또는 AWS 조직일 수 없습니다.

피부여자 보안 주체를 지정하려면 CreateGrant 요청에서 `GranteePrincipal` 파라미터를 사용합니다.

> **Note**
>
> IAM 모범 사례는 장기 보안 인증 정보가 있는 IAM 사용자의 사용을 장려하지 않습니다. 가능할 경우, 임시 보안 인증 정보를 제공하는 IAM 역할을 사용하세요. 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.

피부여자 서비스 보안 주체

권한 부여에 지정된 권한을 가져오는 AWS 서비스 보안 주체입니다. 피부여자 서비스 보안 주체를 지정하려면 CreateGrant 요청에서 `GranteeServicePrincipal` 파라미터를 사용합니다.

를 사용하여 권한 부여를 생성할 때 도 포함해야 `GranteeServicePrincipal`합니다`SourceArn` `grant constraint`. `SourceArn` 제약 조건은 지정된 AWS 리소스를 대신하여 요청이 이루어진 경우에만 서비스 보안 주체가 KMS 키를 사용할 수 있도록 합니다.

를 지정할 때 `RetiringPrincipal` 또는 도 지정`GranteeServicePrincipal`해야 합니다`RetiringServicePrincipal`.

> **Note**
>
> GranteePrincipal 또는 GranteeServicePrincipal 중 하나만 지정해야 하지만 두 가지 모두 지정해서는 안 됩니다.

권한 부여 사용 중지

권한 부여를 종료합니다. 권한 사용을 마치면 권한 부여를 사용 중지합니다.

권한 부여를 취소하거나 사용 중지하면 권한 부여가 삭제됩니다. 그러나 사용 중지는 권한 부여에 지정된 보안 주체에 의해 수행됩니다. 취소는 일반적으로 키 관리자가 수행합니다. 자세한 내용은 권한 부여 사용 중지 및 취소 섹션을 참조하세요.

사용 중지 보안 주체

권한 부여를 사용 중지할 수 있는 보안 주체입니다. 권한 부여에서 사용 중지 보안 주체를 지정할 수 있지만 필수는 아닙니다. 사용 중지 보안 주체는 IAM 사용자, IAM 역할 AWS 계정, 페더레이션 사용자, 수임된 역할 사용자를 포함한 모든 보안 AWS 주체일 수 있습니다. 사용 중지 보안 주체는 KMS 키와 동일한 계정에 있거나 다른 계정에 있을 수 있습니다.

사용 중지 보안 주체를 지정하려면 CreateGrant 요청에서 RetiringPrincipal 파라미터를 사용합니다.

> **Note**
>
> IAM 모범 사례는 장기 보안 인증 정보가 있는 IAM 사용자의 사용을 장려하지 않습니다. 가능할 경우, 임시 보안 인증 정보를 제공하는 IAM 역할을 사용하세요. 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.

서비스 보안 주체 사용 중지

권한 부여를 사용 중지할 수 있는 권한이 있는 AWS 서비스 보안 주체입니다. ??? 사용 중지 서비스 보안 주체를 지정하려면 CreateGrant 요청에서 RetiringServicePrincipal 파라미터를 사용합니다.

> **Note**
>
> RetiringPrincipal 또는 RetiringServicePrincipal을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.

권한 부여를 사용 중지할 수 있는 사용자에 대한 자세한 내용은 섹션을 참조하세요권한 부여 사용 중지 및 취소.

권한 부여 취소

권한 부여를 종료합니다. 권한 부여를 취소하면 권한 부여가 허용하는 권한을 적극적으로 거부할 수 있습니다.

권한 부여를 취소하거나 사용 중지하면 권한 부여가 삭제됩니다. 그러나 사용 중지는 권한 부여에 지정된 보안 주체에 의해 수행됩니다. 취소는 일반적으로 키 관리자가 수행합니다. 자세한 내용은 권한 부여 사용 중지 및 취소 섹션을 참조하세요.

(권한 부여의) 최종 일관성

AWS KMS API는 최종 일관성 모델을 따릅니다. 권한 부여를 생성, 사용 중지 또는 철회할 때, 변경 사항이 AWS KMS전체에 적용되기까지 잠깐의 지연이 있을 수 있습니다. 변경 사항이 시스템 전체에 전파되는 데에는 일반적으로 몇 초도 걸리지 않지만 경우에 따라 몇 분 정도 걸릴 수도 있습니다.

예기치 않은 오류가 발생하는 경우 이 짧은 지연을 의식하게 됩니다. 예를 들어 권한 부여가 전체적으로 알려지기 전에 새 권한 부여를 관리하거나 새 권한 부여에서 권한을 사용하려고 AWS KMS하면 액세스 거부 오류가 발생할 수 있습니다. 권한 부여를 사용 중지하거나 취소하는 경우 권한 부여가 완전히 삭제될 때까지 피부여자 보안 주체가 잠시 동안 해당 권한을 계속 사용할 수 있습니다. 일반적인 전략은 요청을 재시도하는 것이며, 일부 AWS SDKs 포함됩니다.

AWS KMS 에는 이러한 짧은 지연을 완화하는 기능이 있습니다.

- 새 권한 부여에서 권한을 즉시 사용하려면 권한 부여 토큰을 사용하십시오. 권한 부여 토큰을 사용하여 모든 권한 부여 작업에서 권한 부여를 권한 부여를 참조할 수 있습니다. 지침은 권한 부여 토큰 사용 섹션을 참조하세요.
- CreateGrant 작업에는 재시도 작업이 중복 부여를 생성하지 못하도록 하는 Name 파라미터가 있습니다.

> **Note**
>
> 권한 부여 토큰은 서비스의 모든 엔드포인트가 새 권한 부여 상태로 업데이트될 때까지 권한 부여의 유효성을 대체합니다. 대부분의 경우 최종 일관성은 5분 이내에 달성됩니다.

자세한 내용은 AWS KMS 결과적 일관성을 참조하세요.

AWS KMS 권한 부여 모범 사례

AWS KMS 는 권한 부여를 생성, 사용 및 관리할 때 다음 모범 사례를 권장합니다.

- 권한 부여의 권한을 피부여자에게 필요한 권한으로 제한합니다. 최소 권한 액세스 원칙을 사용합니다. 모든 권한 부여 파라미터에서 가능한 한 구체적이어야 합니다.
  - 피부여자에게 필요한 권한 부여 작업만 지정합니다.
  - IAM 역할과 같은 특정 피부여자 보안 주체 또는 특정 피부여자 서비스 보안 주체를 사용합니다.
  - 권한 부여 제약 조건을 사용하여 권한 부여를 추가로 제한합니다.
- 암호화 컨텍스트 권한 부여 제약 조건 을 사용하여 호출자가 의도한 목적으로 KMS 키를 사용하고 있는지 확인합니다. 요청에서 암호화 컨텍스트를 사용하여 데이터를 보호하는 방법에 대한 자세한 내용은 AWS 보안 블로그의 AWS Key Management Service 및 EncryptionContext를 사용하여 암호화된 데이터의 무결성을 보호하는 방법을 참조하세요.

> **Tip**
>
> 가능할 때마다 EncryptionContextEqual 제약 조건을 사용하십시오. EncryptionContextSubset 권한 부여 제약 조건은 올바르게 사용하기가 더 어렵습니다. 그것을 사용해야 하는 경우 문서를 주의 깊게 읽고 권한 부여 제약 조건을 테스트해 의도한대로 작동하는지 확인하십시오.

- 지원되는 AWS 서비스에 대한 권한 부여를 생성할 때 `SourceArn grant constraint`를 사용하여 권한 부여를 특정 리소스로 제한합니다.
- 중복 권한 부여에 유의하세요. 중복 권한 부여에는 권한 부여 이름을 제외하고 동일한 파라미터가 있습니다. 불필요한 중복으로 인해 KMS 키당 권한 부여 할당량. `CreateGrant` 요청을 재시도할 때 중복 권한 부여를 방지하려면 `Name` 파라미터를 사용하십시오. 중복 권한 부여를 검색하려면 ListGrants 작업을 사용합니다.

> **Note**
>
> 일부 AWS 서비스는 중복으로 보일 수 있는 다양한 리소스에 대한 권한 부여를 생성합니다. 이러한 권한 부여에는 서로 다른 리소스에 연결된 수명 주기가 있습니다. AWS 서비스에서 생성된 권한 부여를 삭제하면 중단될 수 있으며 추가 예방 조치가 필요합니다.

- 권한 부여는 자동으로 만료되지 않습니다. 권한이 더 이상 필요하지 않으면 즉시 권한 부여 사용을 중지 또는 취소합니다. 삭제되지 않은 권한 부여는 암호화된 리소스에 대한 보안 위험을 초래할 수 있습니다.

권한 부여에 대한 액세스 제어

주요 정책, IAM 정책 및 권한부여에서 권한 부여를 생성하고 관리하는 작업에 대한 액세스를 제어할 수 있습니다. 권한 부여에서 `CreateGrant` 권한을 받은 보안 주체는 더 제한된 권한 부여 권한을 갖습니다.

API 작업	키 정책 또는 IAM 정책	권한 부여
CreateGrant	✓	✓
ListGrants	✓	-
ListRetirableGrants	✓	-
권한 부여 사용 중지	(제한됨. 권한 부여 사용 중지 및 취소 참조)	✓
RevokeGrant	✓	-

키 정책 또는 IAM 정책을 사용하여 권한 부여를 생성하고 관리하는 작업에 대한 액세스를 제어하는 경우 다음 정책 조건 중 하나 이상을 사용하여 권한을 제한할 수 있습니다.는 다음 권한 부여 관련 조건 키를 모두 AWS KMS 지원합니다. 자세한 정보와 예제는 AWS KMS 조건 키 섹션을 참조하세요.

kms:GrantConstraintSourceArn

보안 주체가 권한 부여에 지정된 값의 제약 `SourceArn` 조건이 포함된 경우에만 권한 부여를 생성할 수 있도록 허용합니다.

kms:GrantConstraintType

권한 부여에 지정된 권한 부여 제약 조건이 포함된 경우에만 보안 주체가 권한 부여를 생성할 수 있습니다.

kms:GranteePrincipal

보안 주체가 지정된 피부여자 보안 주체에 대해서만 권한 부여를 생성하도록 허용합니다.

kms:GranteeServicePrincipal

권한 부여가 특정 피부여자 서비스 위탁자를 지정하는 경우에만 위탁자가 권한 부여를 생성하도록 허용합니다.

kms:GrantIsForAWSResource

와 통합된 AWS 서비스가 보안 주체를 대신하여 요청을 AWS KMS 보내는 `RevokeGrant` 경우에만 보안 주체가 `ListGrants`, 또는 `CreateGrant`를 호출하도록 허용합니다.

kms:GrantOperations

보안 주체가 권한 부여를 생성할 수 있도록 허용하지만 권한 부여를 지정된 작업으로 제한합니다.

kms:RetiringPrincipal

권한 부여가 특정 사용 중지 보안 주체를 지정하는 경우에만 보안 주체가 권한 부여를 생성하도록 허용합니다.

kms:RetiringServicePrincipal

권한 부여가 특정 사용 중지 서비스 보안 주체를 지정하는 경우에만 보안 주체가 권한 부여를 생성하도록 허용합니다.

권한 부여 생성

권한 부여를 생성하기 전에 권한 부여 사용자 정의 옵션에 대해 알아봅니다. 권한 부여 제약 조건을 사용하여 권한 부여의 사용 권한을 제한할 수 있습니다. 또한 `CreateGrant` 권한 부여에 대해 알아보십시오. 권한 부여에서 권한 부여를 생성할 수 있는 권한을 받은 보안 주체는 생성할 수 있는 권한이 제한됩니다.

주제

- 권한 부여 생성
- CreateGrant 권한 부여

권한 부여 생성

권한 부여를 생성하려면 CreateGrant 작업을 호출합니다. KMS 키, 허용된 권한 부여 작업 목록, 피부여자 보안 주체 또는 피부여자 서비스 보안 주체를 지정합니다.

피부여자 보안 주체를 지정할 때 선택적으로 사용 중지 보안 주체 또는 사용 중지 서비스 보안 주체를 지정하고 Constraints 파라미터를 사용하여 권한 부여 제약 조건을 정의할 수 있습니다. 피부여자 서비스 보안 주체를 지정할 때는 SourceArn 권한 부여 제약 조건을 포함해야 합니다. 또한 RetiringServicePrincipal 또는를 지정해야 합니다RetiringPrincipal.

권한 부여를 생성, 폐기 또는 취소할 때 AWS KMS전체에서 변경 내용을 사용할 수 있을 때까지 짧은 지연(보통 5분 미만)이 있을 수 있습니다. 자세한 내용은 결과적 일관성(권한 부여용)을 참조하세요.

예를 들어 다음 CreateGrant 명령은 keyUserRole 역할을 수임할 권한이 있는 사용자가 지정된 대칭 KMS 키에서 Decrypt 작업을 호출할 수 있도록 허용하는 권한 부여를 생성합니다. 이 권한 부여는 RetiringPrincipal 파라미터를 사용하여 권한 부여를 중단시킬 수 있는 보안 주체를 지정합니다. 또한 요청의 암호화 컨텍스트가 "Department": "IT"를 포함하는 경우에만 권한을 허용하는 권한 부여 제약 조건이 포함되어 있습니다.

```
$  aws kms create-grant \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --grantee-principal arn:aws:iam::111122223333:role/keyUserRole \
    --operations Decrypt \
    --retiring-principal arn:aws:iam::111122223333:role/adminRole \
    --constraints EncryptionContextSubset={Department=IT}
```

코드가 CreateGrant 작업을 다시 시도하거나 요청을 자동으로 다시 시도하는AWS SDK를 사용하는 경우 이름 파라미터 옵션을 사용하여 중복 권한 부여의 생성을 방지합니다. 가 이름을 포함하여 기존 권한 부여와 동일한 속성의 권한 부여 CreateGrant 요청을 AWS KMS 받으면 요청을 재시도로 인식하고 새 권한 부여를 생성하지 않습니다. Name 값을 사용하여 AWS KMS 작업에서 권한 부여를 식별할 수 없습니다.

> **Important**
>
> 권한 부여 이름에 기밀 또는 민감한 정보를 포함하지 마십시오. CloudTrail 로그 및 기타 출력에 일반 텍스트로 표시될 수 있습니다.

```
$ aws kms create-grant \
    --name IT-1234abcd-keyUserRole-decrypt \
```

```
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --grantee-principal arn:aws:iam::111122223333:role/keyUserRole \
    --operations Decrypt \
    --retiring-principal arn:aws:iam::111122223333:role/adminRole \
    --constraints EncryptionContextSubset={Department=IT}
```

여러 프로그래밍 언어로 권한 부여를 생성하는 방법을 보여주는 코드 예제는 AWS SDK 또는 CLI와 CreateGrant 함께 사용 단원을 참조하세요.

권한 부여 제약 사용

권한 부여 제약은 권한 부여를 통해 피부여자 보안 주체에게 부여하는 권한에 대한 조건을 설정합니다. 권한 부여 제약 조건은 키 정책 또는 IAM 정책에서 조건 키를 대신합니다.

> **Important**
>
> 이 필드에 기밀 또는 민감한 정보를 포함하지 마십시오. 이 필드는 CloudTrail 로그 및 기타 출력에 일반 텍스트로 표시될 수 있습니다.

AWS KMS 는 암호화 컨텍스트 제약 조건과 제약 조건이라는 두 가지 유형의 권한 부여 제약 SourceArn 조건을 지원합니다.

암호화 컨텍스트 제약 조건

암호화 컨텍스트 제약 조건은 암호화 작업에 대한 요청에서 암호화 컨텍스트에 대한 요구 사항을 설정합니다. 각 암호화 컨텍스트 권한 부여 제약 조건 값에는 최대 8개의 암호화 컨텍스트 페어가 포함될 수 있습니다. 각 제약 조건의 암호화 컨텍스트 값은 384자를 초과할 수 없습니다.

- 암호화 컨텍스트 제약 조건은 대칭 암호화 KMS 키에 대한 권한 부여에서만 유효합니다. 다른 KMS 키를 사용한 암호화 작업은 암호화 컨텍스트를 지원하지 않습니다.
- 암호화 컨텍스트 제약 조건은 DescribeKey 및 RetireGrant 작업에 대해 무시됩니다. DescribeKey 및 RetireGrant에는 암호화 컨텍스트 파라미터가 없지만 암호화 컨텍스트 제약 조건이 있는 권한 부여에 이러한 작업을 포함할 수 있습니다.
- CreateGrant 작업에 대한 권한 부여에서 암호화 컨텍스트 제약 조건을 사용할 수 있습니다. 암호화 컨텍스트 제약 조건에서는 CreateGrant 권한으로 생성된 모든 권한 부여에 동등하게 엄격하거나 더 엄격한 암호화 컨텍스트 제약 조건이 있어야 합니다.

AWS KMS 는 두 가지 암호화 컨텍스트 권한 부여 제약 조건을 지원합니다.

EncryptionContextEquals

`EncryptionContextEquals`를 사용하여 허용된 요청에 대한 정확한 암호화 컨텍스트를 지정합니다.

`EncryptionContextEquals`는 요청의 암호화 컨텍스트 페어가 권한 부여 제약 조건의 암호화 컨텍스트 페어에 대해 정확히 대소문자가 일치해야 합니다. 쌍은 순서에 관계없이 나타날 수 있지만 각 쌍의 키와 값은 다를 수 없습니다.

예를 들어, `EncryptionContextEquals` 권한 부여 제약 조건에 `"Department": "IT"` 암호화 컨텍스트 페어가 필요한 경우 권한 부여는 요청의 암호화 컨텍스트가 정확히 `"Department": "IT"`인 경우에만 지정된 유형의 요청을 허용합니다.

EncryptionContextSubset

`EncryptionContextSubset`을 사용하여 요청에 특정 암호화 컨텍스트 페어가 포함되도록 요구합니다.

`EncryptionContextSubset`는 요청에 권한 부여 제약 조건의 모든 암호화 컨텍스트 페어(정확한 대소문자 구분 일치)을 포함해야 하지만 요청에 추가 암호화 컨텍스트 페어가 있을 수도 있습니다. 쌍은 순서에 관계없이 나타날 수 있지만 각 쌍의 키와 값은 다를 수 없습니다.

예를 들어, `EncryptionContextSubset` 권한 부여 제약 조건에 `Department=IT` 암호화 컨텍스트 페어가 필요한 경우 권한 부여는 요청의 암호화 컨텍스트가 정확히 `"Department": "IT"`인 경우에 지정된 유형의 요청을 허용하거나 다른 암호화 컨텍스트 페어(예: `"Department": "IT","Purpose": "Test"`)와 함께 `"Department": "IT"`를 포함합니다.

대칭 암호화 KMS 키에 대한 권한 부여에서 암호화 컨텍스트 제약 조건을 지정하려면 CreateGrant 작업의 `Constraints` 파라미터를 사용합니다. 이 명령이 생성하는 권한 부여는 Decrypt 작업을 호출할 수 있는 `keyUserRole` 역할 권한을 맡을 권한이 있는 사용자에게 부여합니다. 그러나 이 권한은 `Decrypt` 요청의 암호화 컨텍스트가 `"Department": "IT"` 암호화 컨텍스트 페어인 경우에만 유효합니다.

```
$ aws kms create-grant \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --grantee-principal arn:aws:iam::111122223333:role/keyUserRole \
    --operations Decrypt \
    --retiring-principal arn:aws:iam::111122223333:role/adminRole \
    --constraints EncryptionContextEquals={Department=IT}
```

결과 권한 부여는 다음과 같습니다. keyUserRole 역할에 부여된 권한은 Decrypt 요청이 권한 부여 제약 조건에 지정된 동일한 암호화 컨텍스트 페어를 사용하는 경우에만 유효합니다. KMS 키에 대한 권한 부여를 확인하려면 ListGrants 작업을 사용합니다.

```
$ aws kms list-grants --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
{
    "Grants": [
        {
            "Name": "",
            "IssuingAccount": "arn:aws:iam::111122223333:root",
            "GrantId":
 "abcde1237f76e4ba7987489ac329fbfba6ad343d6f7075dbd1ef191f0120514a",
            "Operations": [
                "Decrypt"
            ],
            "GranteePrincipal": "arn:aws:iam::111122223333:role/keyUserRole",
            "Constraints": {
                "EncryptionContextEquals": {
                    "Department": "IT"
                }
            },
            "CreationDate": 1568565290.0,
            "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "RetiringPrincipal": "arn:aws:iam::111122223333:role/adminRole"
        }
    ]
}
```

EncryptionContextEquals 권한 부여 제약 조건을 충족하려면 Decrypt 작업에 대한 요청의 암호화 컨텍스트가 "Department": "IT" 페어여야 합니다. 피부여자 보안 주체의 다음과 같은 요청은 EncryptionContextEquals 권한 부여 제약 조건을 충족합니다.

```
$ aws kms decrypt \
    --key-id arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab\
    --ciphertext-blob fileb://encrypted_msg \
    --encryption-context Department=IT
```

권한 부여 제약 조건이 EncryptionContextSubset인 경우 요청의 암호화 컨텍스트 페어는 권한 부여 제약 조건의 암호화 컨텍스트 페어를 포함해야 하지만 요청은 다른 암호화 컨텍스트 페어

도 포함할 수 있습니다. 다음 권한 부여 제약 조건에서는 요청의 암호화 컨텍스트 페어 중 하나가 `"Deparment": "IT"`여야 합니다.

```
"Constraints": {
   "EncryptionContextSubset": {
       "Department": "IT"
   }
}
```

피부여자 보안 주체의 다음 요청은 이 예에서 `EncryptionContextEqual` 및 `EncryptionContextSubset` 권한 부여 제약 조건을 모두 충족합니다.

```
$ aws kms decrypt \
    --key-id arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab \
    --ciphertext-blob fileb://encrypted_msg \
    --encryption-context Department=IT
```

그러나 피부여자 주체의 다음과 같은 요청은 `EncryptionContextSubset` 권한 부여 제약 조건을 만족하지만 `EncryptionContextEquals` 권한 부여 제약 조건에는 실패합니다.

```
$ aws kms decrypt \
    --key-id arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab \
    --ciphertext-blob fileb://encrypted_msg \
    --encryption-context Department=IT,Purpose=Test
```

AWS 서비스는에서 KMS 키를 사용할 수 있는 권한을 부여하는 권한 부여에 암호화 컨텍스트 제약 조건을 사용하는 경우가 많습니다 AWS 계정. 예를 들어 Amazon DynamoDB는 다음과 같은 권한 부여를 사용하여 계정에서 DynamoDB용 AWS 관리형 키를 사용할 수 있는 권한을 얻습니다. 이 권한 부여의 `EncryptionContextSubset` 권한 부여 제약 조건은 권한 부여에 지정된 권한이 요청의 암호화 컨텍스트가 `"subscriberID": "111122223333"` 및 `"tableName": "Services"` 쌍을 포함하는 경우에만 유효하게 만듭니다. 이 권한 부여 제약 조건은 이 권한 부여는 DynamoDB가 AWS 계정에서 특정 테이블에 대해서만 지정된 KMS 키를 사용할 수 있게 허용함을 의미입니다.

이 출력을 가져오려면 계정의 DynamoDB AWS 관리형 키 용에서 ListGrants 작업을 실행합니다.

```
$ aws kms list-grants --key-id 0987dcba-09fe-87dc-65ba-ab0987654321
```

```
{
    "Grants": [
        {
            "Operations": [
                "Decrypt",
                "Encrypt",
                "GenerateDataKey",
                "ReEncryptFrom",
                "ReEncryptTo",
                "RetireGrant",
                "DescribeKey"
            ],
            "IssuingAccount": "arn:aws:iam::111122223333:root",
            "Constraints": {
                "EncryptionContextSubset": {
                    "aws:dynamodb:tableName": "Services",
                    "aws:dynamodb:subscriberId": "111122223333"
                }
            },
            "CreationDate": 1518567315.0,
            "KeyId": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321",
            "GranteePrincipal": "dynamodb.us-west-2.amazonaws.com",
            "RetiringPrincipal": "dynamodb.us-west-2.amazonaws.com",
            "Name": "8276b9a6-6cf0-46f1-b2f0-7993a7f8c89a",
            "GrantId":
 "1667b97d27cf748cf05b487217dd4179526c949d14fb3903858e25193253fe59"
        }
    ]
}
```

SourceArn 제약 조건

SourceArn

SourceArn 제약 조건은 Amazon AWS 리소스 이름(ARN)으로 식별되는 특정 리소스를 대신하여 이루어진 요청으로 권한 부여 권한을 제한합니다. SourceArn 권한 부여 제약 조건은 aws:SourceArn 전역 조건 키를 권한 부여에 효과적으로 배치하고 AWS 서비스에서 요청할 때만 평가됩니다.

권한 부여가를 지정할 때는 항상 제약 SourceArn 조건이 필요합니다GranteeServicePrincipal. 선택적으로와 함께 사용할 수 있습니다GranteePrincipal.

암호화 컨텍스트 제약 조건과 달리 SourceArn 제약 조건은 모든 유형의 KMS 키에 대한 권한 부여에서 지원됩니다.

권한 부여에 제약 SourceArn 조건을 지정하려면 CreateGrant 작업에서 Constraints 파라미터를 사용합니다. 다음 예제에서는 AWS 서비스 보안 주체가 지정된 DynamoDB 테이블을 대신하여 요청이 이루어진 경우에만 Encrypt 및 Decrypt 작업을 호출할 수 있도록 허용하는 권한 부여를 생성합니다. 권한 부여가를 지정할 때 SourceArn 제약 조건이 필요합니다GranteeServicePrincipal.

```
$ aws kms create-grant \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --grantee-service-principal service-name.amazonaws.com \
    --operations Encrypt Decrypt \
    --retiring-service-principal service-name.amazonaws.com
 \
    --constraints '{"SourceArn":"arn:aws:dynamodb:us-east-1:111122223333:table/
ExampleTable"}'
```

결과 권한 부여는 다음과 같습니다. 권한 부여의 권한은 요청에 권한 부여 제약 조건의 SourceArn 값과 일치하는 소스 ARN이 포함된 경우에만 유효합니다.

```
$ aws kms list-grants --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
{
    "Grants": [
        {
            "Name": "",
            "IssuingAccount": "arn:aws:iam::111122223333:root",
            "GrantId":
 "abcde1237f76e4ba7987489ac329fbfba6ad343d6f7075dbd1ef191f0120514a",
            "Operations": [
                "Encrypt",
                "Decrypt"
            ],
            "GranteeServicePrincipal": "service-name.amazonaws.com",
            "Constraints": {
                "SourceArn": "arn:aws:dynamodb:us-east-1:111122223333:table/
ExampleTable"
            },
            "CreationDate": 1718567315.0,
            "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "RetiringServicePrincipal": "service-name.amazonaws.com"
```

```
        }
    ]
}
```

여러 제약 조건을 단일 권한 부여로 결합할 수도 있습니다. 그러나 EncryptionContextEquals 또는 중 하나를 지정할 수 EncryptionContextSubset있지만 둘 다 지정할 수는 없습니다.

GranteeServicePrincipal를 지정하면 SourceArn 제약 조건이 필요하므로 여러 제약 조건이 있는 권한 부여를 생성하는 데 유효한 조합은 다음과 같습니다.

- SourceArn가 있는 EncryptionContextEquals
- EncryptionContextSubset가 있는 SourceArn

다음 예제에서는 GranteeServicePrincipal 및를 사용하여 권한 부여를 생성하고 EncryptionContextSubset 제약 조건과 SourceArn 제약 조건을 모두 RetiringServicePrincipal포함합니다.

```
$ aws kms create-grant \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --grantee-service-principal service-name.amazonaws.com \
    --retiring-service-principal service-name.amazonaws.com \
    --operations Encrypt Decrypt GenerateDataKey DescribeKey \
    --constraints '{"EncryptionContextSubset":
{"Department":"IT"},"SourceArn":"arn:aws:dynamodb:us-east-1:111122223333:table/
ExampleTable"}'
```

결과 권한 부여는 다음과 같습니다. 응답에는 GranteeServicePrincipal 및 RetiringServicePrincipal 필드가 포함되고 Constraints 필드에는 EncryptionContextSubset 및가 모두 포함됩니다SourceArn.

```
$ aws kms list-grants --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
{
    "Grants": [
        {
            "Name": "",
            "IssuingAccount": "arn:aws:iam::111122223333:root",
            "GrantId":
 "abcde1237f76e4ba7987489ac329fbfba6ad343d6f7075dbd1ef191f0120514a",
            "Operations": [
                "Encrypt",
                "Decrypt",
```

```
                "GenerateDataKey",
                "DescribeKey"
            ],
            "GranteeServicePrincipal": "service-name.amazonaws.com",
            "Constraints": {
                "EncryptionContextSubset": {
                    "Department": "IT"
                },
                "SourceArn": "arn:aws:dynamodb:us-east-1:111122223333:table/
ExampleTable"
            },
            "CreationDate": 1718567315.0,
            "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "RetiringServicePrincipal": "service-name.amazonaws.com"
        }
    ]
}
```

CreateGrant 권한 부여

권한 부여에는 CreateGrant 호출할 수 있는 권한이 포함될 수 있습니다. 그러나 피부여자 보안 주체가 정책이 아닌 부여에서 CreateGrant를 호출할 수 있는 권한을 얻으면 해당 권한이 제한됩니다.

- 피부여자 보안 주체는 상위 권한 부여의 일부 또는 모든 작업을 허용하는 부여만 생성할 수 있습니다.
- 생성하는 권한 부여의 권한 부여 제약 조건은 최소한 상위 권한 부여의 제약 조건만큼 엄격해야 합니다.

정책 조건에 따라 권한이 제한될 수 있지만 이러한 제한은 정책에서 CreateGrant 권한을 받는 보안 주체에게는 적용되지 않습니다.

예를 들면, GenerateDataKey, Decrypt 및 CreateGrant 작업을 호출할 수 있게 하는 권한 부여를 고려하십시오. CreateGrant 권한을 허용하는 권한을 상위 권한 부여라고 합니다.

```
# The original grant in a ListGrants response.
{
    "Grants": [
        {
            "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
```

```
            "CreationDate": 1572216195.0,
            "GrantId":
 "abcde1237f76e4ba7987489ac329fbfba6ad343d6f7075dbd1ef191f0120514a",
            "Operations": [
                "GenerateDataKey",
                "Decrypt",
                "CreateGrant
            ]
            "RetiringPrincipal": "arn:aws:iam::111122223333:role/adminRole",
            "Name": "",
            "IssuingAccount": "arn:aws:iam::111122223333:root",
            "GranteePrincipal": "arn:aws:iam::111122223333:role/keyUserRole",
            "Constraints": {
                "EncryptionContextSubset": {
                    "Department": "IT"
                }
            },
        }
    ]
}
```

피부여자 보안 주체인 exampleUser는 이 권한을 사용하여 `CreateGrant` 및 `Decrypt`와 같이 원래 권한 부여에 지정된 작업의 하위 집합을 포함하는 권한 부여를 생성할 수 있습니다. 하위 권한 부여는 다른 다른 작업(예: `ScheduleKeyDeletion` 또는 `ReEncrypt`)을 포함할 수 없습니다.

또한, 하위 권한 부여에 대한 권한 부여 제약 조건 은 상위 권한 부여만큼 제한적이거나 더 제한적이어야 합니다. 예를 들어, 하위 권한 부여는 상위 권한 부여의 `EncryptionContextSubset` 제한 조건에 쌍을 추가 할 수 있지만 이를 제거 할 수는 없습니다. 하위 권한 부여는 `EncryptionContextSubset` 제약 조건을 `EncryptionContextEquals` 제약 조건으로 변경할 수 있지만 그 반대는 아닙니다.

IAM 모범 사례는 장기 보안 인증 정보가 있는 IAM 사용자의 사용을 장려하지 않습니다. 가능할 경우, 임시 보안 인증 정보를 제공하는 IAM 역할을 사용하세요. 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.

예를 들어 피부여자 보안 주체는 상위 권한 부여에서 얻은 `CreateGrant` 권한을 사용하여 다음 하위 권한 부여를 생성할 수 있습니다. 하위 권한 부여의 작업은 상위 권한 부여 작업의 하위 집합이며 권한 부여 제약 조건은 더 제한적입니다.

```
# The child grant in a ListGrants response.
{
    "Grants": [
```

```
        {
            "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "CreationDate": 1572249600.0,
            "GrantId":
 "fedcba9999c1e2e9876abcde6e9d6c9b6a1987650000abcee009abcdef40183f",
            "Operations": [
                "CreateGrant"
                "Decrypt"
            ]
            "RetiringPrincipal": "arn:aws:iam::111122223333:user/exampleUser",
            "Name": "",
            "IssuingAccount": "arn:aws:iam::111122223333:root",
            "GranteePrincipal": "arn:aws:iam::111122223333:user/anotherUser",
            "Constraints": {
                "EncryptionContextEquals": {
                    "Department": "IT"
                }
            },
        }
    ]
}
```

하위 권한 부여 anotherUser의 피부여자 보안 주체는 CreateGrant 권한을 사용하여 권한 부여를 생성할 수 있습니다. 그러나, anotherUser가 생성하는 권한 부여는 상위 권한 부여 또는 하위 집합의 작업을 포함해야 하며 권한 부여 제약 조건은 동일하거나 더 엄격해야 합니다.

권한 부여 보기

권한 부여를 보려면 ListGrants 작업을 사용합니다. 권한 부여가 적용되는 KMS 키를 지정해야 합니다. 권한 부여 ID, 피부여자 보안 주체 또는 피부여자 서비스 보안 주체를 기준으로 권한 부여 목록을 필터링할 수도 있습니다. 더 많은 예시는 AWS SDK 또는 CLI와 ListGrants 함께 사용를 참조합니다.

특정 사용 중지 보안 주체 또는 사용 중지 서비스 보안 주체가 있는 AWS 계정 및 리전의 모든 권한 부여를 보려면 ListRetirableGrants를 사용합니다. 응답에는 각 권한 부여에 대한 세부 정보가 포함됩니다.

> **Note**
>
> ListGrants 응답의 GranteePrincipal 필드에는 일반적으로 권한 부여의 피부여자 보안 주체가 포함됩니다. 그러나 권한 부여의 피부여자 보안 주체가 AWS 서비스인 경우

GranteePrincipal 필드에는 여러 피부여자 보안 주체를 나타낼 수 있는 서비스 보안 주체가 포함됩니다.

> **Note**
>
> GranteeServicePrincipal 파라미터로 권한 부여가 생성되면 ListGrants 응답에는 대신 GranteeServicePrincipal 필드가 포함됩니다GranteePrincipal. 이렇게 하면 AWS 서비스 보안 주체에 GranteeServicePrincipal 대해 로 명시적으로 생성된 권한 부여가 GranteePrincipal 필드에 AWS 서비스가 표시되는 권한 부여와 구별됩니다.

예를 들어 다음 명령은 KMS 키에 대한 모든 권한 부여를 나열합니다.

```
$  aws kms list-grants --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
{
    "Grants": [
        {
            "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "CreationDate": 1572216195.0,
            "GrantId":
 "abcde1237f76e4ba7987489ac329fbfba6ad343d6f7075dbd1ef191f0120514a",
            "Constraints": {
                "EncryptionContextSubset": {
                    "Department": "IT"
                }
            },
            "RetiringPrincipal": "arn:aws:iam::111122223333:role/adminRole",
            "Name": "",
            "IssuingAccount": "arn:aws:iam::111122223333:root",
            "GranteePrincipal": "arn:aws:iam::111122223333:user/exampleUser",
            "Operations": [
                "Decrypt"
            ]
        }
    ]
}
```

다음 예제에서는 로 생성된 권한 부여에 대한 ListGrants 응답을 보여줍니다GranteeServicePrincipal. 응답에는 대신 GranteeServicePrincipal 필드가 포함GranteePrincipal되고 Constraints 필드에는 SourceArn 값이 포함됩니다.

```
$  aws kms list-grants --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
{
    "Grants": [
        {
            "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "CreationDate": 1718567315.0,
            "GrantId":
 "abcde1237f76e4ba7987489ac329fbfba6ad343d6f7075dbd1ef191f0120514a",
            "Constraints": {
                "EncryptionContextSubset": {
                    "Department": "IT"
                },
                "SourceArn": "arn:aws:dynamodb:us-east-1:111122223333:table/
ExampleTable"
            },
            "RetiringServicePrincipal": "service-name.amazonaws.com",
            "Name": "",
            "IssuingAccount": "arn:aws:iam::111122223333:root",
            "GranteeServicePrincipal": "service-name.amazonaws.com",
            "Operations": [
                "Encrypt",
                "Decrypt",
                "GenerateDataKey"
            ]
        }
    ]
}
```

권한 부여 토큰 사용

AWS KMS API는 최종 일관성 모델을 따릅니다. 권한 부여를 생성할 때 권한 부여가 즉시 유효하지 않을 수 있습니다. 변경 사항이 AWS KMS전체에 적용되기까지 잠깐의 지연이 있을 수 있습니다. 변경 사항이 시스템 전체에 전파되는 데에는 일반적으로 몇 초도 걸리지 않지만 경우에 따라 몇 분 정도 걸릴 수도 있습니다. 변경 사항이 시스템 전체에 완전히 전파되면 피부여자 보안 주체는 권한 부여 토큰이나 권한 부여에 대한 증거를 지정하지 않고 권한 부여의 권한을 사용할 수 있습니다. 그러나 너무 새 권한

부여로 인해 아직 모든 사람에게 알려지지 않은 경우 AccessDeniedException 오류와 함께 AWS KMS요청이 실패할 수 있습니다.

새 권한 부여에서 권한을 즉시 사용하려면 권한 부여에 대한 권한 부여 토큰을 사용하십시오. CreateGrant 작업으로 반환되는 권한 부여 토큰을 저장합니다. 그런 다음 AWS KMS 작업 요청에 권한 부여 토큰을 제출합니다. 권한 부여 토큰을 모든 AWS KMS 권한 부여 작업에 제출하고 동일한 요청에서 여러 권한 부여 토큰을 제출할 수 있습니다.

다음 예에서는 CreateGrant 작업을 사용하여 GenerateDataKey 및 Decrypt 작업을 허용하는 권한 부여를 생성합니다. CreateGrant에서 반환되는 권한 부여 토큰을 token 변수에 저장합니다. 그런 다음 GenerateDataKey 작업에 대한 호출에서 token 변수의 권한 부여 토큰을 사용합니다.

```
# Create a grant; save the grant token
$ token=$(aws kms create-grant \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --grantee-principal arn:aws:iam::111122223333:user/appUser \
    --retiring-principal arn:aws:iam::111122223333:user/acctAdmin \
    --operations GenerateDataKey Decrypt \
    --query GrantToken \
    --output text)

# Use the grant token in a request
$ aws kms generate-data-key \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --key-spec AES_256 \
    --grant-tokens $token
```

권한이 있는 위탁자는 권한 부여가 AWS KMS를 통해 제공되기 전에도 권한 부여 토큰을 사용하여 새로운 권한 부여를 사용 중지할 수도 있습니다. (RevokeGrant 작업은 권한 부여 토큰을 허용하지 않습니다.) 자세한 내용은 권한 부여 사용 중지 및 취소 섹션을 참조하세요.

```
# Retire the grant
$ aws kms retire-grant --grant-token $token
```

권한 부여 사용 중지 및 취소

권한 부여를 삭제하려면 부여를 사용 중지하거나 취소합니다.

RetireGrant 및 RevokeGrant 작업은 서로 매우 유사합니다. 두 작업 모두 권한부여를 삭제하므로 권한 부여로 허용하는 권한이 제거됩니다. 이러한 작업의 주요 차이점은 권한이 부여되는 방식입니다.

RevokeGrant

대부분의 AWS KMS 작업과 마찬가지로 `RevokeGrant` 작업에 대한 액세스는 키 정책 및 IAM 정책을 통해 제어됩니다. RevokeGrant API는 `kms:RevokeGrant` 권한이 있는 모든 보안 주체가 호출할 수 있습니다. 이 권한은 키 관리자에게 부여된 표준 권한에 포함됩니다. 일반적으로 관리자는 권한 부여로 허용되는 권한을 거부하기 위해 권한 부여를 취소합니다.

RetireGrant

권한 부여는 누가 권한 부여를 사용 중지할 수 있는지 결정합니다. 이 설계를 통해 키 정책 또는 IAM 정책을 변경하지 않고 권한 부여의 수명 주기를 제어할 수 있습니다. 일반적으로 권한 사용을 마치면 권한 부여를 사용 중지합니다.

권한 부여는 다음 중 하나에 의해 사용 중지될 수 있습니다.

- 권한 부여에 지정된 사용 중지 보안 주체 또는 사용 중지 서비스 보안 주체입니다.
- 권한 부여에 작업이 포함된 경우 피부여자 보안 주체 또는 피부여자 서비스 보안 주체입니다. `RetireGrant`
- 권한 부여가 생성된 AWS 계정 입니다.

IAM 정책에서 사용할 수 있는 `kms:RetireGrant` 권한이 있지만 유틸리티가 제한되어 있습니다. 권한 부여에 지정된 보안 주체는 `kms:RetireGrant` 권한 없이 권한 부여를 취소할 수 있습니다. `kms:RetireGrant` 권한만으로는 보안 주체가 권한 부여를 사용 중지할 수 없습니다. `kms:RetireGrant` 권한은 키 정책 또는 리소스 제어 정책에서 유효하지 않습니다.

- 권한 부여 사용 중지 권한을 거부하려면 IAM 정책에서 `kms:RetireGrant` 권한과 함께 Deny 작업을 사용할 수 있습니다.
- KMS 키를 소유 AWS 계정 한는 계정의 IAM 보안 주체에게 `kms:RetireGrant` 권한을 위임할 수 있습니다.
- 사용 중지된 보안 주체가 다른 경우 다른 계정의 AWS 계정관리자는 `kms:RetireGrant`를 사용하여 해당 계정의 IAM 보안 주체에게 권한 부여 사용 중지 권한을 위임할 수 있습니다.

AWS KMS API는 최종 일관성 모델을 따릅니다. 권한 부여를 생성, 사용 중지 또는 철회할 때, 변경 사항이 AWS KMS전체에 적용되기까지 잠깐의 지연이 있을 수 있습니다. 변경 사항이 시스템 전체에 전파되는 데에는 일반적으로 몇 초도 걸리지 않지만 경우에 따라 몇 분 정도 걸릴 수도 있습니다. 새 권한 부여를 즉시 삭제해야 하는 경우 전체 권한 부여 토큰을 AWS KMS사용하여 권한 부여를 사용 중지합니다. ??? 권한 부여 토큰을 사용하여 권한 부여를 취소할 수는 없습니다.

에 대한 조건 키 AWS KMS

키 정책 및 IAM 정책에서 AWS KMS 리소스에 대한 액세스를 제어하는 조건을 지정할 수 있습니다. 이 정책문은 조건이 true일 때만 유효합니다. 예를 들어, 특정 날짜 이후에만 정책문이 적용되기를 원할 수 있습니다. 또는 API 요청에 특정 값이 있는 경우에만 정책문이 액세스를 제어하기를 원할 수 있습니다.

조건을 지정하려면 IAM 조건 연산자와 함께 정책문의 `Condition` 요소에서 조건 키를 사용합니다. 일부 조건 키는 일반적으로에 적용되며 AWS, 다른 조건 키는에 고유합니다 AWS KMS.

조건 키 값은 AWS KMS 키 정책 및 IAM 정책에 대한 문자 및 인코딩 규칙을 준수해야 합니다. 키 정책 문서 규칙에 대한 자세한 내용은 키 정책 형식 섹션을 참조하세요. IAM 정책 문서 규칙에 대한 자세한 내용은 IAM 사용 설명서의 IAM 이름 요구 사항 섹션을 참조하세요.

주제

- AWS 전역 조건 키
- AWS KMS 조건 키
- AWS KMS 증명된 플랫폼의 조건 키

AWS 전역 조건 키

AWS 는 액세스 제어에 IAM을 사용하는 모든 AWS 서비스에 대한 정책 조건 키 세트인 전역 조건 키를 정의합니다.는 모든 전역 조건 키를 AWS KMS 지원합니다. AWS KMS 키 정책 및 IAM 정책에서 사용할 수 있습니다.

예를 들어 요청의 보안 주체가 조건 키 값에 Amazon 리소스 이름(ARN)으로 표시되는 경우에만 aws:PrincipalArn 전역 조건 키를 사용하여 AWS KMS key (KMS 키)에 대한 액세스를 허용할 수 있습니다. 에서 속성 기반 액세스 제어(ABAC)를 지원하려면 IAM 정책의 aws:ResourceTag/tag-key 전역 조건 키를 사용하여 특정 태그가 있는 KMS 키에 대한 액세스를 허용할 AWS KMS수 있습니다.

보안 주체가 AWS 서비스 보안 주체인 정책에서 AWS 서비스가 혼동된 대리자로 사용되지 않도록 또는 aws:SourceArn aws:SourceAccount 전역 조건 키를 사용할 수 있습니다. 자세한 내용은 `aws:SourceArn` 또는 `aws:SourceAccount` 조건 키 사용을 참조하세요.

사용 가능한 요청 유형을 포함한 AWS 전역 조건 키에 대한 자세한 내용은 IAM 사용 설명서의 AWS 전역 조건 컨텍스트 키를 참조하세요. IAM 정책에서 전역 조건 키를 사용하는 예는 IAM 사용 설명서의 요청에 대한 액세스 제어 및 태그 키 제어를 참조하세요.

다음 주제에서는 IP 주소 및 VPC 엔드포인트를 기반으로 조건 키를 사용하기 위한 특별한 지침을 제공합니다.

주제

- AWS KMS 권한이 있는 정책에서 IP 주소 조건 사용
- AWS KMS 권한이 있는 정책에서 VPC 엔드포인트 조건 사용
- IAM 및 AWS KMS 키 정책에서 IPv6 주소 사용

AWS KMS 권한이 있는 정책에서 IP 주소 조건 사용

AWS KMS 를 사용하여 통합 AWS 서비스에서 데이터를 보호할 수 있습니다. 그러나 액세스를 허용하거나 거부하는 동일한 정책 설명에서 IP 주소 조건 연산자 또는 `aws:SourceIp` 조건 키를 지정할 때는 주의해야 합니다 AWS KMS. 예를 들어 AWS의 정책은 소스 IP를 AWS 기반으로에 대한 액세스 거부 AWS 작업을 지정된 IP 범위의 요청으로 제한합니다.

다음 시나리오를 고려하십시오.

1. AWS: 소스 IP를 AWS 기반으로에 대한 액세스 거부에 표시된 것과 같은 정책을 IAM 자격 증명에 연결합니다. `aws:SourceIp` 조건 키의 값을 사용자 회사의 IP 주소 범위로 설정합니다. 이 IAM ID에게는 Amazon EBS, Amazon EC2 및 AWS KMS사용을 허용하는 다른 정책이 연결되어 있습니다.
2. ID가 암호화된 EBS 볼륨을 EC2 인스턴스에 연결하려고 합니다. 사용자가 모든 관련 서비스를 이용할 권한을 가지고 있음에도 인증 오류로 인해 이 작업은 실패합니다.

볼륨의 암호화된 데이터 키 복호화 AWS KMS 요청이 Amazon EC2 인프라와 연결된 IP 주소에서 오기 때문에 2단계는 실패합니다. 요청이 성공하려면 원래 사용자의 IP 주소에서 이루어져야 합니다. 1단계의 정책은 지정된 IP 주소 이외에서 이루어진 모든 요청을 명시적으로 거부하기 때문에, EBS 볼륨의 암호화된 데이터 키를 해독할 수 있는 권한이 Amazon EC2에 부여되지 않습니다.

또한 요청이 Amazon VPC 엔드포인트에서 이루어지는 경우 `aws:SourceIP` 조건 키는 유효하지 않습니다. AWS KMS VPC 엔드포인트를 포함해 VPC 엔드포인트로 요청을 제한하려면 `aws:SourceVpce` 또는 `aws:SourceVpc` 조건 키를 사용합니다. 자세한 정보는 Amazon VPC 사용 설명서의 VPC 엔드포인트 - 엔드포인트 사용 제어를 참조하세요.

AWS KMS 권한이 있는 정책에서 VPC 엔드포인트 조건 사용

는 AWS PrivateLink로 구동되는AWS KMS Amazon Virtual Private Cloud(VPC) 엔드포인트를 지원합니다. 요청이 VPC에서 오거나 VPC 엔드포인트를 사용할 때 키 정책 및 IAM 정책에서 다음과 같은 전역 조건 키를 사용하여 AWS KMS 리소스에 대한 액세스를 제어할 수 있습니다. 자세한 내용은 VPC 엔드포인트를 사용하여 AWS KMS 리소스에 대한 액세스 제어을 참조하세요.

- `aws:SourceVpc`는 지정된 VPC의 요청으로 액세스를 제한합니다.

- `aws:SourceVpce`는 지정된 VPC 엔드포인트의 요청으로 액세스를 제한합니다.

이러한 조건 키를 사용하여 KMS 키에 대한 액세스를 제어하는 경우가 사용자를 대신하여 사용하는 AWS 서비스에 AWS KMS 대한 액세스를 실수로 거부할 수 있습니다.

IP 주소 조건 키 예제와 같은 상황을 피하도록 주의하십시오. KMS 키에 대한 요청을 VPC 또는 VPC 엔드포인트로 제한하면 Amazon S3 또는 Amazon EBS와 같은 통합 서비스 AWS KMS 에서에 대한 호출이 실패할 수 있습니다. 원본 요청이 궁극적으로 VPC 또는 VPC 엔드포인트에서 시작된 경우에도 이러한 상황이 발생할 수 있습니다.

IAM 및 AWS KMS 키 정책에서 IPv6 주소 사용

IPv6를 AWS KMS 통해에 액세스하기 전에 IP 주소 제한이 포함된 키 및 IAM 정책이 IPv6 주소 범위를 포함하도록 업데이트되었는지 확인합니다. IP 기반 정책이 IPv6 주소를 처리하도록 업데이트되지 않은 경우, 클라이언트가 IPv6을 사용할 때 액세스 권한을 잘못 잃거나 얻을 수 있습니다. KMS 액세스 제어에 대한 일반적인 지침은 KMS 키 액세스 및 권한 페이지를 참조하세요. KMS 및 듀얼 스택 지원에 대해 알아보려면 듀얼 스택 엔드포인트 지원 페이지를 참조하세요.

> **Important**
>
> 이러한 문은 어떤 작업도 허용하지 않습니다. 이러한 문은 특정 작업을 허용하는 다른 문과 함께 사용하세요.

다음 문은 `192.0.2.*`의 IPv4 주소 범위에서 발생한 요청에 대해 모든 KMS 권한에 대한 액세스를 명시적으로 거부합니다. 이 범위 밖의 IP 주소는 KMS 권한이 명시적으로 거부되지 않습니다. 모든 IPv6 주소는 거부 범위 밖에 있으므로, 이 문은 IPv6 주소에 대해 KMS 권한을 명시적으로 거부하지 않습니다.

```
{
     "Sid": "DenyKMSPermissions",
     "Effect": "Deny",
    "Action": [
        "kms:*"
    ],
    "Resource": "*",
    "Condition": {
        "NotIpAddress": {
            "aws:SourceIp": [
```

```
                "192.0.2.0/24"
            ]
        }
    }
}
```

다음 예제와 같이 IPv4(Condition) 및 IPv6(192.0.2.0/24) 주소 범위를 모두 거부하도록 2001:db8:1234::/32 요소를 수정할 수 있습니다.

```
{
    "Sid": "DenyKMSPermissions",
    "Effect": "Deny",
    "Action": [
        "kms:*"
    ],
    "Resource": "*",
    "Condition": {
        "NotIpAddress": {
            "aws:SourceIp": [
                "192.0.2.0/24",
                "2001:db8:1234::/32"
            ]
        }
    }
}
```

AWS KMS 조건 키

AWS KMS 는 키 정책 및 IAM 정책에 사용할 수 있는 조건 키 세트를 제공합니다. 이러한 조건 키는에 고유합니다 AWS KMS. 예를 들어 대칭 암호화 KMS 키에 대한 액세스를 제어할 때 kms:EncryptionContext:*context-key* 조건 키를 사용해 특정한 암호화 컨텍스트를 요구할 수 있습니다.

API 작업 요청에 대한 조건

많은 AWS KMS 조건 키는 AWS KMS 작업 요청의 파라미터 값을 기반으로 KMS 키에 대한 액세스를 제어합니다. 예를 들어 IAM 정책에서 kms:KeySpec 조건 키를 사용하여 CreateKey 요청의 KeySpec 파라미터 값이 RSA_4096인 경우에만 CreateKey 작업의 사용을 허용할 수 있습니다.

이 유형의 조건은 파라미터의 기본값을 사용하는 경우와 같이 요청에 파라미터가 나타나지 않는 경우에도 작동합니다. 예를 들어 kms:KeySpec 조건 키를 사용하여 사용자가 KeySpec 파라미터 값이 기본

값 SYMMETRIC_DEFAULT인 경우에만 CreateKey 작업을 사용하도록 허용할 수 있습니다. 이 조건은 KeySpec 파라미터 값이 SYMMETRIC_DEFAULT인 요청과 KeySpec 파라미터가 없는 요청을 허용합니다.

API 작업에 사용되는 KMS 키에 대한 조건

일부 AWS KMS 조건 키는 작업에 사용되는 KMS 키의 속성을 기반으로 작업에 대한 액세스를 제어할 수 있습니다. 예를 들어 kms:KeyOrigin 조건을 사용하여 KMS 키의 Origin이 AWS_KMS인 경우에만 보안 주체가 KMS 키에서 GenerateDataKey를 호출하도록 할 수 있습니다. 이러한 방식으로 조건 키를 사용할 수 있는지 확인하려면 조건 키의 설명을 참조하십시오.

작업은 특정 KMS 키에 대해 권한이 부여된 KMS 키 리소스 작업이어야 합니다. KMS 키 리소스 작업을 식별하려면 작업 및 리소스 테이블에서, 해당 작업에 대한 Resources 열에서 KMS key 값을 찾습니다. ListKeys와 같이 특정 KMS 키 리소스에 대해 권한이 부여되지 않은 작업과 함께 이 유형의 조건 키를 사용하면 조건을 만족시킬 수 없으므로 권한이 적용되지 않습니다. ListKeys 작업 권한 부여와 관련된 KMS 키 리소스가 없고 KeySpec 속성이 없기 때문입니다.

다음 주제에서는 각 AWS KMS 조건 키를 설명하고 정책 구문을 보여주는 예제 정책 문을 포함합니다.

조건 키와 함께 집합 연산자 사용

정책 조건이 요청의 태그 세트와 정책의 태그 세트와 같은 두 값 세트를 비교하는 경우 세트를 비교하는 AWS 방법을 알려주어야 합니다. IAM은 이를 위해 두 개의 세트 연산자 ForAnyValue와 ForAllValues를 정의합니다. 세트 연산자는 세트 연산자를 필요로 하는 다중값 조건 키(multi-valued condition keys)에만 사용합니다. 단일 값 조건 키(single-valued condition keys)에는 집합 연산자를 사용하지 마세요. 항상 정책문을 프로덕션 환경에서 사용하기 전에 철저히 테스트합니다.

조건 키는 단일 값이거나 다중 값입니다. AWS KMS 조건 키가 단일 값인지 다중 값인지 확인하려면 조건 키 설명의 값 유형 열을 참조하세요.

- 단일 값 조건 키는 권한 부여 컨텍스트(요청 또는 리소스)에서 최대 하나의 값을 갖습니다. 예를 들어 각 API 호출은 하나에서만 시작될 수 있으므로 AWS 계정kms:CallerAccount는 단일 값 조건 키입니다. 단일 값 조건 키(single-valued condition keys)에는 집합 연산자를 사용하지 마세요.
- 다중 값 조건 키는 권한 부여 컨텍스트(요청 또는 리소스)에서 여러 값을 갖습니다. 예를 들어, 각 KMS 키에는 여러 개의 별칭이 있을 수 있으므로 kms:ResourceAliases에는 여러 개의 값이 있을 수 있습니다. 다중값 조건 키에는 집합 연산자가 필요합니다.

단일 값 조건 키와 다중값 조건 키의 차이는 정책 조건의 값 수가 아니라 권한 부여 컨텍스트의 값 수에 따라 달라집니다.

> **Warning**
>
> 단일 값 조건 키와 함께 집합 연산자를 사용하면 지나치게 허용 (또는 지나치게 제한적인) 정책 문을 만들 수 있습니다. 집합 연산자는 다중값 조건 키(multi-valued condition keys)에만 사용합니다.
>
> kms:EncryptionContext:context-key 또는 `aws:RequestTag/`*`tag-key`* 조건 키가 있는 `ForAllValues` 세트 연산자를 포함하는 정책을 생성하거나 업데이트하면가 다음 오류 메시지를 AWS KMS 반환합니다.
>
> `OverlyPermissiveCondition: Using the ForAllValues set operator with a single-valued condition key matches requests without the specified [encryption context or tag] or with an unspecified [encryption context or tag]. To fix, remove ForAllValues.`

`ForAnyValue` 및 `ForAllValues` 집합 연산자에 대한 자세한 내용은 IAM 사용 설명서의 여러 키 및 값 사용을 참조하세요. 단일 값 조건에서 `ForAllValues` 집합 연산자를 사용할 때의 위험에 대한 자세한 내용은 IAM 사용 설명서의 보안 경고 – 단일 값 키가 있는 ForAllValues 섹션을 참조하세요.

주제

- kms:BypassPolicyLockoutSafetyCheck
- kms:CallerAccount
- kms:CustomerMasterKeySpec(사용되지 않음)
- kms:CustomerMasterKeyUsage(사용되지 않음)
- kms:DataKeyPairSpec
- kms:EncryptionAlgorithm
- kms:EncryptionContext:context-key
- kms:EncryptionContextKeys
- kms:ExpirationModel
- kms:GrantConstraintType
- kms:GrantIsForAWSResource
- kms:GrantOperations
- kms:GranteePrincipal
- kms:KeyAgreementAlgorithm
- kms:KeyOrigin

- kms:KeySpec
- kms:KeyUsage
- kms:MacAlgorithm
- kms:MessageType
- kms:MultiRegion
- kms:MultiRegionKeyType
- kms:PrimaryRegion
- kms:ReEncryptOnSameKey
- kms:RequestAlias
- kms:ResourceAliases
- kms:ReplicaRegion
- kms:RetiringPrincipal
- kms:GrantConstraintSourceArn
- kms:GranteeServicePrincipal
- kms:RetiringServicePrincipal
- kms:RotationPeriodInDays
- kms:ScheduleKeyDeletionPendingWindowInDays
- kms:SigningAlgorithm
- kms:TrailingDaysWithoutKeyUsage
- kms:ValidTo
- kms:ViaService
- kms:WrappingAlgorithm
- kms:WrappingKeySpec

kms:BypassPolicyLockoutSafetyCheck

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
`kms:BypassPolicyLo`	부울	단일 값	`CreateKey`	IAM 정책만

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
ckoutSafetyCheck			PutKeyPolicy	키 정책 및 IAM 정책

kms:BypassPolicyLockoutSafetyCheck 조건 키는 요청의 BypassPolicyLockoutSafetyCheck 파라미터 값을 기반으로 CreateKey 및 PutKeyPolicy 작업에 대한 액세스를 제어합니다.

다음 예제 IAM 정책문은 CreateKey 요청의 BypassPolicyLockoutSafetyCheck 파라미터 값이 true.일 때 사용자의 KMS 키 생성 권한을 거부하여 사용자가 정책 잠금 안전 점검을 우회하지 못하도록 합니다.

```
{
  "Effect": "Deny",
  "Action": [
    "kms:CreateKey",
    "kms:PutKeyPolicy"
  ],
  "Resource": "*",
  "Condition": {
    "Bool": {
      "kms:BypassPolicyLockoutSafetyCheck": true
    }
  }
}
```

또한 IAM 정책 또는 키 정책의 kms:BypassPolicyLockoutSafetyCheck 조건 키를 사용하여 PutKeyPolicy 작업에 대한 액세스를 제어할 수도 있습니다. 키 정책에 들어 있는 다음 예제 정책문은 KMS 키의 정책을 변경할 때 사용자가 정책 잠금 안전 점검을 우회하지 못하도록 합니다.

명시적 Deny를 사용하는 대신에, 이 정책문에서는 Null 조건 연산자와 Allow를 함께 사용하여 요청에 BypassPolicyLockoutSafetyCheck 파라미터가 포함되지 않을 때만 액세스를 허용합니다. 이 파라미터를 사용하지 않는 경우 기본값은 false입니다. 이와 같이 강도가 조금 낮은 정책문은 극히 드문 경우지만 우회가 필요할 경우 재정의할 수 있습니다.

```
{
  "Effect": "Allow",
  "Action": "kms:PutKeyPolicy",
```

```
  "Resource": "*",
  "Condition": {
    "Null": {
      "kms:BypassPolicyLockoutSafetyCheck": true
    }
  }
}
```

참고 항목

- kms:KeySpec
- kms:KeyOrigin
- kms:KeyUsage

kms:CallerAccount

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
kms:CallerAccount	문자열	단일 값	KMS 키 리소스 작업 사용자 지정 키 스토어 작업	키 정책 및 IAM 정책

이 조건 키를 사용해 AWS 계정의 모든 ID(사용자와 역할)에 대한 액세스를 허용하거나 거부할 수 있습니다. 키 정책에서 Principal 요소를 이용해 정책문이 적용될 자격 증명을 지정합니다. Principal 요소에 대한 구문은 AWS 계정의 모든 자격 증명을 지정하는 방법을 제공하지 않지만, 그러나이 조건 키를 모든 AWS 자격 증명을 지정하는 Principal 요소와 결합하여이 효과를 달성할 수 있습니다.

이를 사용하여 모든 KMS 키 리소스 작업, 즉 특정 KMS 키를 사용하는 모든 AWS KMS 작업에 대한 액세스를 제어할 수 있습니다. KMS 키 리소스 작업을 식별하려면 작업 및 리소스 테이블에서, 해당 작업에 대해 Resources 열에서 KMS key 값을 찾습니다. 사용자 지정 키 스토어를 관리하는 작업에도 유효합니다.

예를 들어 다음 키 정책문은 kms:CallerAccount 조건 키를 사용하는 방법을 보여줍니다. 이 정책 설명은 AWS 관리형 키 Amazon EBS용의 키 정책에 있습니다. 모든 AWS 자격 증명을 지정하는

Principal 요소를 kms:CallerAccount 조건 키와 결합하여 AWS 계정 111122223333의 모든 자격 증명에 대한 액세스를 효과적으로 허용합니다. 여기에는 Amazon EBS를 통해 들어오는 요청만 허용하여 권한을 추가로 제한하는 추가 AWS KMS 조건 키(kms:ViaService)가 포함되어 있습니다. 자세한 내용은 kms:ViaService 단원을 참조하십시오.

```
{
  "Sid": "Allow access through EBS for all principals in the account that are
 authorized to use EBS",
  "Effect": "Allow",
  "Principal": {"AWS": "*"},
  "Condition": {
    "StringEquals": {
      "kms:CallerAccount": "111122223333",
      "kms:ViaService": "ec2.us-west-2.amazonaws.com"
    }
  },
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:CreateGrant",
    "kms:DescribeKey"
  ],
  "Resource": "*"
}
```

kms:CustomerMasterKeySpec(사용되지 않음)

kms:CustomerMasterKeySpec 조건 키는 더 이상 사용되지 않습니다. 대신 kms:KeySpec 조건 키를 사용합니다.

kms:CustomerMasterKeySpec 및 kms:KeySpec 조건 키는 같은 방식으로 작동하며, 이름만 다릅니다. kms:KeySpec을 사용할 것을 권장합니다. 그러나 변경 사항을 방지하기 위해는 두 조건 키를 모두 AWS KMS 지원합니다.

kms:CustomerMasterKeyUsage(사용되지 않음)

kms:CustomerMasterKeyUsage 조건 키는 더 이상 사용되지 않습니다. 대신 kms:KeyUsage 조건 키를 사용합니다.

kms:CustomerMasterKeyUsage 및 kms:KeyUsage 조건 키는 같은 방식으로 작동하며, 이름만 다릅니다. kms:KeyUsage을 사용할 것을 권장합니다. 그러나 변경 사항을 방지하기 위해는 두 조건 키를 모두 AWS KMS 지원합니다.

kms:DataKeyPairSpec

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
kms:DataKeyPairSpec	문자열	단일 값	GenerateDataKeyPair GenerateDataKeyPairWithoutPlaintext	키 정책 및 IAM 정책

이 조건 키를 사용하여 요청의 KeyPairSpec 파라미터 값을 기반으로 GenerateDataKeyPair 및 GenerateDataKeyPairWithoutPlaintext 작업에 대한 액세스를 제어할 수 있습니다. 예를 들어 사용자가 특정 유형의 데이터 키 페어만 생성하도록 허용할 수 있습니다.

다음 예제 키 정책 문은 kms:DataKeyPairSpec 조건 키를 사용하여 사용자가 KMS 키를 사용하여 RSA 데이터 키 페어만 생성하도록 허용합니다.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
  "Action": [
    "kms:GenerateDataKeyPair",
    "kms:GenerateDataKeyPairWithoutPlaintext"
  ],
  "Resource": "*",
  "Condition": {
    "StringLike": {
      "kms:DataKeyPairSpec": "RSA*"
    }
  }
```

```
}
```

참고 항목

- kms:KeySpec
- the section called “kms:EncryptionAlgorithm”
- the section called “kms:EncryptionContext:context-key”
- the section called “kms:EncryptionContextKeys”

kms:EncryptionAlgorithm

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형

`kms:EncryptionAlgorithm`	문자열	단일 값	`Decrypt`
			`Encrypt`
			`GenerateDataKey`
			`GenerateDataKeyPair`
			`GenerateDataKeyPairWithoutPlaintext`
			`GenerateDataKeyWithoutPlaintext`
			`ReEncrypt`
				키 정책 및 IAM 정책

`kms:EncryptionAlgorithm` 조건 키를 사용하여 작업에 사용되는 암호화 알고리즘을 기반으로 암호화 작업에 대한 액세스를 제어할 수 있습니다. Encrypt, Decrypt 및 ReEncrypt 작업의 경우 요청의 EncryptionAlgorithm 파라미터의 값에 따라 액세스를 제어합니다. 데이터 키 및 데이터 키 페어를 생성하는 작업의 경우 데이터 키를 암호화하는 데 사용되는 암호화 알고리즘을 기반으로 액세스를 제어합니다.

이 조건 키는 외부의 비대칭 KMS 키 페어에서 퍼블릭 키로 암호화하는 AWS KMS등 외부에서 수행되는 작업에 영향을 주지 않습니다 AWS KMS.

요청의 EncryptionAlgorithm 파라미터

사용자가 KMS 키에 특정 암호화 알고리즘만 사용하도록 하려면 `Deny` 효과 및 `StringNotEquals` 조건 연산자가 포함된 정책문을 사용합니다. 예를 들어, 다음 예제 키 정책문에서는 요청의 암호화 알고리즘이 `RSAES_OAEP_SHA_256`(RSA KMS 키와 함께 사용되는 비대칭 암호화 알고리즘)이 아닐 경우 `ExampleRole` 역할을 수임할 수 있는 보안 주체가 지정된 암호화 작업에서 이 KMS 키를 사용할 수 없습니다.

사용자가 특정 암호화 알고리즘을 사용하도록 허용하는 정책 문과 달리, 이와 같은 이중 부정을 가진 정책 문은 이 KMS 키의 다른 정책 및 권한 부여가 이 역할이 다른 암호화 알고리즘을 사용하도록 허용하는 것을 금지합니다. 이 키 정책문의 `Deny`는 `Allow` 효과가 있는 모든 키 정책 또는 IAM 정책에 우선하며 KMS 키 및 해당 보안 주체에 대한 모든 권한 부여에 우선합니다.

```
{
  "Sid": "Allow only one encryption algorithm with this asymmetric KMS key",
  "Effect": "Deny",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*"
  ],
  "Resource": "*",
  "Condition": {
    "StringNotEquals": {
      "kms:EncryptionAlgorithm": "RSAES_OAEP_SHA_256"
    }
  }
}
```

작업에 사용되는 암호화 알고리즘

또한 kms:EncryptionAlgorithm 조건 키를 사용하여 알고리즘이 요청에 지정되지 않은 경우에도 작업에 사용된 암호화 알고리즘을 기반으로 작업에 대한 액세스를 제어할 수 있습니다. 이를 통해 SYMMETRIC_DEFAULT 알고리즘을 요구하거나 금지할 수 있습니다. 이 알고리즘은 기본값이기 때문에 요청에 지정되지 않을 수 있습니다.

kms:EncryptionAlgorithm 조건 키를 사용하여 데이터 키 및 데이터 키 페어를 생성하는 작업에 대한 액세스도 제어할 수 있습니다. 이러한 작업은 대칭 암호화 KMS 키와 SYMMETRIC_DEFAULT 알고리즘만 사용합니다.

예를 들어 다음 IAM 정책은 보안 주체를 대칭 암호화로 제한합니다. 요청에 지정되거나 작업에 사용된 암호화 알고리즘이 SYMMETRIC_DEFAULT가 아닐 경우 예제 계정에서 암호화 작업을 위한 KMS 키에 대한 액세스를 모두 거부합니다. GenerateDataKey*를 포함하면 권한에 GenerateDataKey, GenerateDataKeyWithoutPlaintext, GenerateDataKeyPair 및 GenerateDataKeyPairWithoutPlaintext가 추가됩니다. 조건은 항상 대칭 암호화 알고리즘을 사용하기 때문에 이러한 작업에 영향을 주지 않습니다.

```
{
  "Sid": "AllowOnlySymmetricAlgorithm",
  "Effect": "Deny",
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*"
  ],
  "Resource": "arn:aws:kms:us-west-2:111122223333:key/*",
  "Condition": {
    "StringNotEquals": {
      "kms:EncryptionAlgorithm": "SYMMETRIC_DEFAULT"
    }
  }
}
```

참고 항목

- the section called “kms:MacAlgorithm”
- kms:SigningAlgorithm

kms:EncryptionContext:context-key

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형

`kms:EncryptionContext:` *`context-key`*	문자열	단일 값	`CreateGrant`
			`Encrypt`
			`Decrypt`
			`GenerateDataKey`
			`GenerateDataKeyPair`
			`GenerateDataKeyPairWithoutPlaintext`
			`GenerateDataKeyWithoutPlaintext`
			`ReEncrypt`
			`RetireGrant`
				키 정책 및 IAM 정책

`kms:EncryptionContext:`*`context-key`* 조건 키를 사용하여 암호화 작업 요청의 암호화 컨텍스트를 기반으로 대칭 암호화 KMS 키에 대한 액세스를 제어할 수 있습니다. 이 조건 키를 사용하여 암호화 컨텍스트 페어의 키와 값을 평가합니다. 키 또는 값에 관계없이 암호화 컨텍스트 키만 평가하거나 암호화 컨텍스트를 요구하려면 kms:EncryptionContextKeys 조건 키를 사용합니다.

> **Note**
>
> 조건 키 값은 키 정책 및 IAM 정책에 대한 문자 규칙을 준수해야 합니다. 암호화 컨텍스트에서 유효한 일부 문자는 정책에서 유효하지 않습니다. 이 조건 키를 사용하여 유효한 암호화 컨텍스트 값을 모두 표현하지 못할 수 있습니다. 키 정책 문서 규칙에 대한 자세한 내용은 키 정책 형식 섹션을 참조하세요. IAM 정책 문서 규칙에 대한 자세한 내용은 IAM 사용 설명서의 IAM 이름 요구 사항 섹션을 참조하세요.

비대칭 KMS 키 또는 HMAC KMS 키를 사용하여 암호화 작업에서 암호화 컨텍스트를 지정할 수 없습니다. 비대칭 알고리즘 및 MAC 알고리즘은 암호화 컨텍스트를 지원하지 않습니다.

kms:EncryptionContext:context-key 조건 키를 사용하려면 *context-key* 자리 표시자를 암호화 컨텍스트 키로 바꿉니다. *context-value* 자리 표시자를 암호화 컨텍스트 값으로 바꿉니다.

```
"kms:EncryptionContext:context-key": "context-value"
```

예를 들어, 다음 조건 키는 키가 AppName이고 값이 ExampleApp (AppName = ExampleApp)인 암호화 컨텍스트를 지정합니다.

```
"kms:EncryptionContext:AppName": "ExampleApp"
```

이것은 단일 값 조건 키입니다. 조건 키의 키는 특정 암호화 컨텍스트 키(context-key)입니다. 각 API 요청에 여러 암호화 컨텍스트 페어를 포함할 수 있지만 지정된 context-key가 있는 암호화 컨텍스트 쌍은 하나의 값만 가질 수 있습니다. 예를 들어, kms:EncryptionContext:Department 조건 키는 Department 키가 있는 암호화 컨텍스트 페어에만 적용되며 Department 키가 있는 지정된 암호화 컨텍스트 페어에는 하나의 값만 있을 수 있습니다.

kms:EncryptionContext:*context-key* 조건 키와 함께 집합 연산자를 사용하지 마십시오. Allow 작업, kms:EncryptionContext:*context-key* 조건 키 및 ForAllValues 집합 연산자를 사용하여 정책문을 생성하는 경우 조건은 암호화 컨텍스트가 없는 요청과 정책 조건에 지정되지 않은 암호화 컨텍스트 페어가 있는 요청을 허용합니다.

> **Warning**
>
> 단일 값 조건 키(single-valued condition keys)에는 ForAnyValue 또는 ForAllValues 집합 연산자를 사용하지 마세요. 이러한 집합 연산자는 필요한 값을 요구하지 않고 금지하려는 값을 허용하는 정책 조건을 만들 수 있습니다.

kms:EncryptionContext:context-key를 사용하여 ForAllValues 세트 연산자를 포함하는 정책을 생성하거나 업데이트하면가 다음 오류 메시지를 AWS KMS 반환합니다. `OverlyPermissiveCondition:EncryptionContext: Using the ForAllValues set operator with a single-valued condition key matches requests without the specified encryption context or with an unspecified encryption context. To fix, remove ForAllValues.`

특정 암호화 컨텍스트 페어를 요구하려면 StringEquals 연산자와 함께 kms:EncryptionContext:*context-key* 조건 키를 사용하세요.

다음 예제 키 정책문은 요청의 암호화 컨텍스트에 AppName:ExampleApp 페어가 포함된 경우에만 역할을 맡을 수 있는 보안 주체가 GenerateDataKey 요청에서 KMS 키를 사용할 수 있도록 허용합니다. 다른 암호화 컨텍스트 페어가 허용됩니다.

키 이름은 대/소문자를 구분하지 않습니다. 값의 대소문자 구분은 StringEquals와 같은 조건 연산자에 의해 결정됩니다. 자세한 내용은 암호화 컨텍스트 조건의 대/소문자 구분 섹션을 참조하세요.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": "kms:GenerateDataKey",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:EncryptionContext:AppName": "ExampleApp"
    }
  }
}
```

암호화 컨텍스트 페어를 요구하고 다른 모든 암호화 컨텍스트 페어를 금지하려면 정책문에서 kms:EncryptionContext:context-key와 kms:EncryptionContextKeys를 모두 사용하세요. 다음 키 정책문은 kms:EncryptionContext:AppName 조건을 사용하여 요청에 AppName=ExampleApp 암호화 컨텍스트 페어를 요구합니다. 또한 ForAllValues 집합 연산자와 함께 kms:EncryptionContextKeys 조건 키를 사용하여 AppName 암호화 컨텍스트 키만 허용합니다.

ForAllValues 집합 연산자는 요청의 암호화 컨텍스트 키를 AppName으로 제한합니다. ForAllValues 집합 연산자가 있는 kms:EncryptionContextKeys 조건이 정책문에서 단독으로

사용된 경우 이 집합 연산자는 암호화 컨텍스트가 없는 요청을 허용합니다. 그러나 요청에 암호화 컨텍스트가 없으면 kms:EncryptionContext:AppName 조건이 실패합니다. ForAllValues 집합 연산자에 대한 자세한 내용은 IAM 사용 설명서의 여러 키 및 값 사용을 참조하세요.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/KeyUsers"
  },
  "Action": "kms:GenerateDataKey",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:EncryptionContext:AppName": "ExampleApp"
    },
    "ForAllValues:StringEquals": {
      "kms:EncryptionContextKeys": [
        "AppName"
      ]
    }
  }
}
```

이 조건 키를 사용하여 특정 작업에 대한 KMS 키에 대한 액세스를 거부할 수도 있습니다. 다음 예제 키 정책문은 Deny 효과를 사용하여 요청의 암호화 컨텍스트에 Stage=Restricted 암호화 컨텍스트 페어가 포함된 경우 보안 주체가 KMS 키를 사용하는 것을 금지합니다. 이 조건은 Stage 키 및 Stage=Test와 같은 기타 값이 있는 암호화 컨텍스트 페어를 포함하여 다른 암호화 컨텍스트 페어로 요청을 허용합니다.

```
{
  "Effect": "Deny",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": "kms:GenerateDataKey",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:EncryptionContext:Stage": "Restricted"
    }
  }
```

```
}
```

여러 암호화 컨텍스트 페어 사용

여러 암호화 컨텍스트 페어를 요구하거나 금지할 수 있습니다. 여러 암호화 컨텍스트 페어 중 하나를 요구할 수도 있습니다. 이러한 조건을 해석하는 데 사용되는 논리에 대한 자세한 내용은 IAM 사용 설명서의 여러 키 또는 값으로 조건 만들기 섹션을 참조하세요.

> **Note**
>
> 이 항목의 이전 버전에서는 kms:EncryptionContext:context-key 조건 키와 함께 `ForAnyValue` 및 `ForAllValues` 집합 연산자를 사용한 정책문을 표시했습니다. 단일 값 조건 키와 함께 집합 연산자를 사용하면 암호화 컨텍스트가 없고 지정되지 않은 암호화 컨텍스트 페어가 없는 요청을 허용하는 정책이 생성될 수 있습니다.
> 예를 들어, `Allow` 효과가 있는 정책 조건, `ForAllValues` 집합 연산자 및 `"kms:EncryptionContext:Department": "IT"` 조건 키가 암호화 컨텍스트를 "Department=IT" 페어로 제한하지 않습니다. 암호화 컨텍스트가없는 요청과 지정되지 않은 암호화 컨텍스트 쌍을 사용하는 요청(예: `Stage=Restricted`)을 허용합니다.
> 정책을 검토하고 kms:EncryptionContext:context-key를 사용하여 모든 조건에서 집합 연산자를 제거하십시오. 이 형식의 정책을 생성하거나 업데이트하려는 시도는 `OverlyPermissiveCondition` 예외와 함께 실패합니다. 오류를 해결하려면 집합 연산자를 삭제합니다.

여러 암호화 컨텍스트 쌍을 요구하려면 동일한 조건의 쌍을 나열합니다. 다음 예제 키 정책문에는 두 개의 암호화 컨텍스트 쌍인 `Department=IT`와 `Project=Alpha`가 필요합니다. 조건이 다른 키(`kms:EncryptionContext:Department` 및 `kms:EncryptionContext:Project`)를 가지고 있기 때문에 AND 연산자에 의해 암시적으로 연결됩니다. 다른 암호화 컨텍스트 쌍은 허용되지만 필수는 아닙니다.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": "kms:Decrypt",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
```

```
      "kms:EncryptionContext:Department": "IT",
      "kms:EncryptionContext:Project": "Alpha"
    }
  }
}
```

하나의 암호화 컨텍스트 쌍 또는 다른 쌍을 요구하려면 각 조건 키를 별도의 정책문에 배치합니다. 다음 예제 키 정책은 `Department=IT` 또는 `Project=Alpha` 쌍이나 둘 다 사용할 수 있습니다. 다른 암호화 컨텍스트 쌍은 허용되지만 필수는 아닙니다.

```
{
 "Effect": "Allow",
 "Principal": {
  "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
 },
 "Action": "kms:GenerateDataKey",
 "Resource": "*",
 "Condition": {
  "StringEquals": {
   "kms:EncryptionContext:Department": "IT"
  }
 }
},
{
 "Effect": "Allow",
 "Principal": {
  "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
 },
 "Action": "kms:GenerateDataKey",
 "Resource": "*",
 "Condition": {
  "StringEquals": {
   "kms:EncryptionContext:Project": "Alpha"
  }
 }
}
```

특정 암호화 쌍을 요구하고 다른 모든 암호화 컨텍스트 쌍을 실행하려면 정책문에서 kms:EncryptionContext:context-key와 `kms:EncryptionContextKeys`를 모두 사용하십시오. 다음 키 정책문은 kms:EncryptionContext:context-key 컨디션을 사용하여 `Department=IT` 및 `Project=Alpha` 페어 모두가 있는 암호화 컨텍스트를 요구합니다. `ForAllValues` 집합 연산자와

함께 kms:EncryptionContextKeys 조건 키를 사용하여 Department 및 Project 암호화 컨텍스트 키만 허용합니다.

ForAllValues 집합 연산자는 요청의 암호화 컨텍스트 키를 Department 및 Project으로 제한합니다. 조건에 단독으로 사용된 경우 이 집합 연산자는 암호화 컨텍스트가 없는 요청을 허용하지만 이 구성에서는 이 조건의 kms:EncryptionContext:context-key가 실패합니다.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": "kms:GenerateDataKey",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:EncryptionContext:Department": "IT",
      "kms:EncryptionContext:Project": "Alpha"
    },
    "ForAllValues:StringEquals": {
      "kms:EncryptionContextKeys": [
        "Department",
        "Project"
      ]
    }
  }
}
```

여러 암호화 컨텍스트 쌍을 금지할 수도 있습니다. 다음 예제 키 정책문은 Deny 효과를 사용하여 요청의 암호화 컨텍스트에 Stage=Restricted 또는 Stage=Production 페어가 포함된 경우 보안 주체가 KMS 키를 사용하는 것을 금지합니다.

동일한 키(kms:EncryptionContext:Stage)에 대한 여러 값(Restricted 및 Production)은 암시적으로 OR에 의해 연결됩니다. 자세한 내용은 IAM 사용 설명서의 여러 키 또는 값이 있는 조건에 대한 평가 논리 섹션을 참조하세요.

```
{
  "Effect": "Deny",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": "kms:GenerateDataKey",
```

```
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:EncryptionContext:Stage": [
         "Restricted",
         "Production"
      ]
    }
  }
}
```

암호화 컨텍스트 조건의 대/소문자 구분

해독 작업에 지정된 암호화 컨텍스트는 암호화 작업에 지정된 암호화 컨텍스트에 대해 대소문자가 정확히 일치해야 합니다. 여러 쌍이 있는 암호화 컨텍스트에서 쌍의 순서만 다를 수 있습니다.

그러나 정책 조건에서 조건 키는 대소문자를 구분하지 않습니다. 조건 값의 대소문자 구분은 `StringEquals` 또는 `StringEqualsIgnoreCase`와 같이 사용하는 정책 조건 연산자에 의해 결정됩니다.

이와 같이 `kms:EncryptionContext:` 접두사와 *`context-key`* 대체로 구성된 조건 키는 대소문자를 구분하지 않습니다. 이 조건을 사용하는 정책은 조건 키의 두 요소의 대소문자를 검사하지 않습니다. 대소문자 구분, 즉 *`context-value`* 대체는 정책 조건 연산자에 의해 결정됩니다.

예를 들어 다음 정책 명령문은 대문자 사용에 관계없이 암호화 컨텍스트에 Appname키가 포함 된 경우 작업을 허용합니다. 예를 들어 다음 정책 명령문은 대문자 사용에 관계없이 암호화 컨텍스트에 `StringEquals` 키가 포함 된 경우 작업을 허용합니다ExampleApp.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": "kms:Decrypt",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:EncryptionContext:Appname": "ExampleApp"
    }
  }
}
```

대소문자를 구분하는 암호화 컨텍스트 키를 요구하려면 kms: EncryptionContextKeys 정책 조건을 대소문자를 구분하는 조건 연산자 (예: StringEquals)와 함께 사용하십시오. 이 정책 조건에서는 암호화 컨텍스트 키가 이 정책 조건의 값이기 때문에 대소문자 구분은 조건 연산자에 의해 결정됩니다.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": "kms:GenerateDataKey",
  "Resource": "*",
  "Condition": {
    "ForAnyValue:StringEquals": {
      "kms:EncryptionContextKeys": "AppName"
    }
  }
}
```

암호화 컨텍스트 키와 값 모두에 대해 대소문자가 구분된 평가가 필요하면 kms:EncryptionContextKeys 및 kms:EncryptionContext:context-key 정책 조건을 동일한 정책문에 함께 사용하십시오. 대소문자를 구분하는 조건 연산자(예: StringEquals)는 항상 조건의 값에 적용됩니다. 암호화 컨텍스트 키(예: AppName)는 kms:EncryptionContextKeys 조건의 값입니다. 암호화 컨텍스트 값(예: ExampleApp)은 kms:EncryptionContext:context-key 조건의 의 값입니다.

예를 들어, StringEquals 연산자는 대소문자를 구분하기 때문에 다음 예제 키 정책문에서는 암호화 컨텍스트 키와 암호화 컨텍스트 값 모두 대소문자를 구분합니다.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": "kms:GenerateDataKey",
  "Resource": "*",
  "Condition": {
    "ForAnyValue:StringEquals": {
      "kms:EncryptionContextKeys": "AppName"
    },
    "StringEquals": {
      "kms:EncryptionContext:AppName": "ExampleApp"
    }
  }
```

```
}
```

암호화 컨텍스트 조건에 변수 사용

암호화 컨텍스트 쌍의 키와 값은 단순 리터럴 문자열이어야 합니다. 정수나 객체, 또는 완전히 해결되지 않는 어떤 형식도 사용할 수 없습니다. 정수 또는 부동 소수점과 같은 다른 유형을 사용하는 경우는 이를 리터럴 문자열로 AWS KMS 해석합니다.

```
"encryptionContext": {
    "department": "10103.0"
}
```

그러나 `kms:EncryptionContext:context-key` 조건 키의 값은 IAM 정책 변수일 수 있습니다. 이러한 정책 변수는 런타임에 요청에 포함된 값을 기반으로 확인됩니다. 예를 들어 `aws:CurrentTime`은 요청의 시간으로 확인되고 `aws:username`은 호출자의 표시 이름으로 확인됩니다.

이러한 정책 변수를 사용하여 암호화 컨텍스트에서 매우 구체적인 정보(예: 호출자의 사용자 이름)를 요구하는 조건을 포함하는 정책 문을 생성할 수 있습니다. 변수를 포함하기 때문에 특정 역할을 수임할 수 있는 모든 사용자에 동일한 정책 문을 사용할 수 있습니다. 사용자마다 따로 정책 문을 작성할 필요가 없습니다.

예를 들어 모든 사용자가 동일한 KMS 키를 사용하여 데이터를 암호화 및 해독하는 역할을 수임할 수 있게 하려고 합니다. 그러나 모든 사용자가 직접 암호화한 데이터만 암호화를 해제하도록 허용하려고 합니다. 먼저 모든 요청에 키가 `user`이고 값이 다음과 같은 호출자의 AWS 사용자 이름인 암호화 컨텍스트를 AWS KMS 포함하도록 요구합니다.

```
"encryptionContext": {
    "user": "bob"
}
```

그런 다음 이 요구 사항을 시행하려면 다음 예제와 같은 정책 문을 사용할 수 있습니다. 이 정책문은 `TestTeam` 역할에 KMS 키를 사용하여 데이터를 암호화 및 해독할 권한을 부여합니다. 그러나 이 권한은 요청의 암호화 컨텍스트가 `"user": "<username>"` 쌍을 포함하는 경우에만 유효합니다. 사용자 이름을 표현하기 위해 조건이 `aws:username` 정책 변수를 사용합니다.

요청이 평가될 때 호출자의 사용자 이름이 조건의 변수를 대체합니다. 따라서 조건은 "bob"에 대해서는 암호화 컨텍스트 `"user": "bob"`을 요구하고 "alice"에 대해서는 `"user": "alice"`를 요구합니다.

```
{
  "Effect": "Allow",
```

```
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/TestTeam"
  },
  "Action": [
    "kms:Decrypt",
    "kms:Encrypt"
  ],
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:EncryptionContext:user": "${aws:username}"
    }
  }
}
```

IAM 정책 변수는 `kms:EncryptionContext:`*`context-key`* 조건 키의 값에만 사용할 수 있습니다. 키에는 변수를 사용할 수 없습니다.

또한 변수에 공급자별 컨텍스트 키를 사용할 수도 있습니다. 이러한 컨텍스트 키는 웹 자격 증명 페더레이션을 사용하여 AWS 에 로그인한 사용자를 고유하게 식별합니다.

다른 모든 변수와 마찬가지로 이러한 변수도 실제 암호화 컨텍스트가 아니라 `kms:EncryptionContext:`*`context-key`* 정책 조건에서만 사용할 수 있습니다. 또한 키가 아니라 조건의 값에만 사용할 수 있습니다.

예를 들어 다음 키 정책 문은 이전 것과 비슷합니다. 그러나 조건은 키가 `sub`이고 값이 Amazon Cognito 사용자 풀에 로그인한 사용자를 고유하게 식별하는 암호화 컨텍스트를 요구합니다. Amazon Cognito에서 사용자 및 역할을 식별하는 방법에 대한 자세한 내용은 Amazon Cognito 개발자 안내서의 IAM 역할을 참조하세요.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/TestTeam"
  },
  "Action": [
    "kms:Decrypt",
    "kms:Encrypt"
  ],
  "Resource": "*",
  "Condition": {
    "StringEquals": {
```

```
        "kms:EncryptionContext:sub": "${cognito-identity.amazonaws.com:sub}"
      }
    }
}
```

참고 항목

- the section called “kms:EncryptionContextKeys”
- the section called “kms:GrantConstraintType”

kms:EncryptionContextKeys

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
kms:EncryptionContextKeys	문자열(목록)	다중 값	CreateGrant Decrypt Encrypt GenerateDataKey GenerateDataKeyPair GenerateDataKeyPairWithoutPlaintext GenerateDataKeyWithoutPlaintext ReEncrypt	키 정책 및 IAM 정책

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
			RetireGrant	

kms:EncryptionContextKeys 조건 키를 사용하여 암호화 작업 요청의 암호화 컨텍스트를 기반으로 대칭 암호화 KMS 키에 대한 액세스를 제어할 수 있습니다. 이 조건 키를 사용하여 각 암호화 컨텍스트 페어의 키만 평가하십시오. 암호화 컨텍스트의 키와 값을 평가하려면 이 kms:EncryptionContext:*context-key* 조건 키를 사용합니다.

비대칭 KMS 키 또는 HMAC KMS 키를 사용하여 암호화 작업에서 암호화 컨텍스트를 지정할 수 없습니다. 비대칭 알고리즘 및 MAC 알고리즘은 암호화 컨텍스트를 지원하지 않습니다.

> **Note**
>
> 암호화 컨텍스트 키를 포함한 조건 키 값은 AWS KMS 키 정책에 대한 문자 및 인코딩 규칙을 준수해야 합니다. 이 조건 키를 사용하여 유효한 암호화 컨텍스트 키를 모두 표현하지 못할 수 있습니다. 키 정책 문서 규칙에 대한 자세한 내용은 키 정책 형식 섹션을 참조하세요. IAM 정책 문서 규칙에 대한 자세한 내용은 IAM 사용 설명서의 IAM 이름 요구 사항 섹션을 참조하세요.

이는 다중 값 조건 키입니다. 각 API 요청에서 여러 암호화 컨텍스트 페어를 지정할 수 있습니다. kms:EncryptionContextKeys는 요청의 암호화 컨텍스트 키를 정책의 암호화 컨텍스트 키 집합과 비교합니다. 이러한 집합이 비교되는 방법을 확인하려면 정책 조건에서 ForAnyValue 또는 ForAllValues 연산자를 제공해야 합니다. 집합 연산자에 대한 자세한 내용은 IAM 사용 설명서의 여러 키 및 값 사용을 참조하세요.

- ForAnyValue: 요청에 있는 하나 이상의 암호화 컨텍스트 키가 정책 조건의 암호화 컨텍스트 키와 일치해야 합니다. 다른 암호화 컨텍스트 키가 허용됩니다. 요청에 암호화 컨텍스트가 없는 경우 조건이 충족되지 않습니다.
- ForAllValues: 요청의 모든 암호화 컨텍스트 키는 정책 조건의 암호화 컨텍스트 키와 일치해야 합니다. 이 집합 연산자는 암호화 컨텍스트 키를 정책 조건의 컨텍스트 키로 제한합니다. 암호화 컨텍스트 키는 필요하지 않지만 지정되지 않은 암호화 컨텍스트 키는 금지됩니다.

다음 예제 키 정책 명령문에서 kms:EncryptionContextKeys 조건 키를 ForAnyValue 집합 연산자와 함께 사용합니다. 이 정책문은 지정된 작업에 대해 KMS 키를 사용할 수 있도록 허용하지만 값에 관계없이 요청의 암호화 컨텍스트 페어 중 하나 이상이 AppName 키를 포함하는 경우에만 가능합니다.

예를 들어, 이 키 정책문은 첫 번째 암호화 컨텍스트 페어가 조건을 충족하기 때문에 두 개의 암호화 컨텍스트 페어(AppName=Helper 및 Project=Alpha)가 있는 GenerateDataKey 요청을 허용합니다. Project=Alpha만 있거나 암호화 컨텍스트가 없는 요청은 실패합니다.

StringEquals 조건 연산은 대소문자를 구별하므로 정책문에는 암호화 컨텍스트 키의 철자와 대소 문자가 필요합니다. 그러나 StringEqualsIgnoreCase과 같은 키 대소문자를 무시하는 조건 연산자를 사용할 수 있습니다.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": [
    "kms:Encrypt",
    "kms:GenerateDataKey*"
  ],
  "Resource": "*",
  "Condition": {
    "ForAnyValue:StringEquals": {
      "kms:EncryptionContextKeys": "AppName"
    }
  }
}
```

또한 kms:EncryptionContextKeys 조건 키를 사용하여 KMS 키를 사용하는 암호화 작업에서 암호화 컨텍스트(모든 암호화 컨텍스트)를 요구할 수도 있습니다.

다음의 키 정책문 예제는 kms:EncryptionContextKeys 조건 키와 함께 를 사용하여 조건 키에서 API 요청에 존재할 때만(null 아님) CMK에 대한 액세스를 허용합니다. 이 조건은 암호화 컨텍스트의 키나 값을 확인하지 않습니다. 암호화 컨텍스트가 존재하는지 확인합니다.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": [
    "kms:Encrypt",
    "kms:GenerateDataKey*"
  ],
```

```
  "Resource": "*",
  "Condition": {
    "Null": {
      "kms:EncryptionContextKeys": false
    }
  }
}
```

다음 사항도 참조하세요.

- kms:EncryptionContext:context-key
- kms:GrantConstraintType

kms:ExpirationModel

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
kms:ExpirationModel	문자열	단일 값	ImportKeyMaterial	키 정책 및 IAM 정책

kms:ExpirationModel 조건 키는 요청의 ExpirationModel 파라미터 값을 기반으로 ImportKeyMaterial 작업에 대한 액세스를 제어합니다.

ExpirationModel은 가져온 키 자료가 만료되는지 여부를 결정하는 선택적 파라미터입니다. 유효한 값은 KEY_MATERIAL_EXPIRES 및 KEY_MATERIAL_DOES_NOT_EXPIRE이며, 기본값은 KEY_MATERIAL_EXPIRES입니다.

만료 날짜 및 시간은 ValidTo 파라미터의 값에 따라 결정됩니다. ValidTo 파라미터의 값이 ExpirationModel가 아닌 경우에는 KEY_MATERIAL_DOES_NOT_EXPIRE 파라미터가 필요합니다. 또한 kms:ValidTo 조건 키를 사용하여 특정 만료 날짜를 액세스 조건으로 요구할 수도 있습니다.

다음 예제 정책 문은 kms:ExpirationModel 조건 키를 사용하여 요청에 ExpirationModel 파라미터가 포함되고 해당 값이 KEY_MATERIAL_DOES_NOT_EXPIRE일 때만 사용자가 키 구성 요소를 KMS 키로 가져올 수 있도록 합니다.

```
{
  "Effect": "Allow",
```

```
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": "kms:ImportKeyMaterial",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:ExpirationModel": "KEY_MATERIAL_DOES_NOT_EXPIRE"
    }
  }
}
```

또한 kms:ExpirationModel 조건 키를 사용하여 사용자가 키 구성 요소가 만료된 경우에만 키 구성 요소를 가져오도록 할 수 있습니다. 다음다음 예제 키 정책 문은 Null 조건 연산자와 함께 kms:ExpirationModel 조건 키를 사용하여 요청에 ExpirationModel 파라미터가 없는 경우에만 사용자가 키 구성 요소를 가져올 수 있도록 합니다. ExpirationModel의 기본값은 KEY_MATERIAL_EXPIRES입니다.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": "kms:ImportKeyMaterial",
  "Resource": "*",
  "Condition": {
    "Null": {
      "kms:ExpirationModel": true
    }
  }
}
```

참고 항목

- kms:ValidTo
- kms:WrappingAlgorithm
- kms:WrappingKeySpec

kms:GrantConstraintType

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
`kms:GrantConstraintType`	문자열	단일 값	`CreateGrant` `RetireGrant`	키 정책 및 IAM 정책

이 조건 키를 사용하여 요청의 권한 부여 제약 조건 유형에 따라 CreateGrant 작업에 대한 액세스를 제어할 수 있습니다.

권한 부여를 생성하면 선택적으로 권한 부여 제약을 지정해 특정 암호화 컨텍스트가 있는 경우에만 권한 부여에 의해 허용된 작업을 허용할 수 있습니다. 권한 부여 제약은 `EncryptionContextEquals` 또는 `EncryptionContextSubset`, 두 유형 중 하나일 수 있습니다. 이 조건 키를 사용해 요청 제약에 어떤 유형이 포함되었는지 확인할 수 있습니다.

> **Note**
>
> `kms:GrantConstraintType` 조건 키는 암호화 컨텍스트 권한 부여 제약 조건(`EncryptionContextEquals` 및 `EncryptionContextSubset`)만 평가합니다. `SourceArn` 권한 부여 제약 조건에 따라 액세스를 제어하려면 kms:GrantConstraintSourceArn 조건 키를 사용합니다.

> **Important**
>
> 이 필드에 기밀 또는 민감한 정보를 포함하지 마십시오. 이 필드는 CloudTrail 로그 및 기타 출력에 일반 텍스트로 표시될 수 있습니다.

다음 예제 키 정책 문은 `kms:GrantConstraintType` 조건 키를 사용해 요청에 `EncryptionContextEquals` 권한 부여 제약이 포함된 경우에만 사용자가 권한 부여를 생성하도록 허용합니다. 이 예제는 키 정책의 정책 설명을 보여줍니다.

```
{
```

```
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": "kms:CreateGrant",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:GrantConstraintType": "EncryptionContextEquals"
    }
  }
}
```

참고 항목

- kms:EncryptionContext:context-key
- kms:EncryptionContextKeys
- kms:GrantConstraintSourceArn
- kms:GranteePrincipal
- kms:GranteeServicePrincipal
- kms:GrantIsForAWSResource
- kms:GrantOperations
- kms:RetiringPrincipal
- kms:RetiringServicePrincipal

kms:GrantIsForAWSResource

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
kms:GrantIsForAWSResource	부울	단일 값	CreateGrant ListGrants RevokeGrant	키 정책 및 IAM 정책

AWS KMS와 통합된AWS 서비스가 사용자를 대신하여 작업을 직접적으로 호출하는 경우에만 CreateGrant, ListGrants 또는 RevokeGrant 작업에 대한 권한을 허용하거나 거부합니다. 이 정책 조건은 사용자가 이러한 권한 부여 작업을 직접 호출하도록 허용하지 않습니다.

다음 예제 키 정책 문은 `kms:GrantIsForAWSResource` 조건 키를 사용합니다. Amazon EBS AWS KMS와 같이와 통합된 AWS 서비스는 지정된 보안 주체를 대신하여이 KMS 키에 대한 권한 부여를 생성할 수 있습니다.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
  "Action": "kms:CreateGrant",
  "Resource": "*",
  "Condition": {
    "Bool": {
      "kms:GrantIsForAWSResource": true
    }
  }
}
```

참고 항목

- kms:GrantConstraintSourceArn
- kms:GrantConstraintType
- kms:GranteePrincipal
- kms:GranteeServicePrincipal
- kms:GrantOperations
- kms:RetiringPrincipal
- kms:RetiringServicePrincipal

kms:GrantOperations

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
kms:GrantOperations	문자열	다중 값	CreateGrant	키 정책 및 IAM 정책

이 조건 키를 사용해 요청에서 권한 부여 작업을 기반으로 CreateGrant 작업에 대한 액세스를 제어할 수 있습니다. 예를 들어, 사용자가 암호화할 권한은 위임하지만 해독할 권한은 위임하지 않는 권한 부여를 생성하도록 허용할 수 있습니다. 권한 부여에 대한 자세한 내용은 권한 부여 사용을 참조하세요.

이는 다중 값 조건 키입니다. kms:GrantOperations는 CreateGrant 요청의 승인 작업 세트를 정책의 승인 작업 세트와 비교합니다. 이러한 집합이 비교되는 방법을 확인하려면 정책 조건에서 ForAnyValue 또는 ForAllValues 연산자를 제공해야 합니다. 집합 연산자에 대한 자세한 내용은 IAM 사용 설명서의 여러 키 및 값 사용을 참조하세요.

- ForAnyValue: 요청의 모든 권한 부여 작업은 정책 조건의 부여 작업과 일치해야 합니다. 기타 권한 부여 작업이 허용됩니다.
- ForAllValues: 요청의 모든 권한 부여 작업은 정책 조건의 부여 작업과 일치해야 합니다. 이 집합 연산자는 권한 부여 작업을 정책 조건에 지정된 작업으로 제한합니다. 어떠한 권한 부여 작업도 필요하지 않지만 지정되지 않은 권한 부여 작업은 금지합니다.

  ForAllValues는 요청에 권한 부여 작업이 없을 때 true를 반환하지만 CreateGrant는 허용하지 않습니다. 만약 Operations 파라미터가 없거나 null 값을 갖는 경우 CreateGrant 요청이 실패합니다.

다음 예제 키 정책 문은 kms:GrantOperations 조건 키를 사용하여 권한 부여 작업이 Encrypt, ReEncryptTo 또는 둘 다인 경우에만 사용자가 권한 부여를 생성할 수 있도록 합니다. 권한 부여에 다른 작업이 포함된 경우 CreateGrant 요청이 실패합니다.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
  "Action": "kms:CreateGrant",
```

```
  "Resource": "*",
  "Condition": {
    "ForAllValues:StringEquals": {
      "kms:GrantOperations": [
        "Encrypt",
        "ReEncryptTo"
      ]
    }
  }
}
```

정책 조건의 집합 연산자를 ForAnyValue로 변경하면 정책문에서 권한 부여의 권한 부여 작업 중 하나 이상이 Encrypt 또는 ReEncryptTo여야 하지만 Decrypt 또는 ReEncryptFrom과 같은 다른 권한 부여 작업은 허용합니다.

참고 항목

- kms:GrantConstraintSourceArn
- kms:GrantConstraintType
- kms:GrantIsForAWSResource
- kms:GranteePrincipal
- kms:GranteeServicePrincipal
- kms:RetiringPrincipal
- kms:RetiringServicePrincipal

kms:GranteePrincipal

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
kms:GranteePrincipal	문자열	단일 값	CreateGrant	IAM 및 키 정책

이 조건 키를 사용하여 요청의 GranteePrincipal 파라미터 값을 기반으로 CreateGrant 작업에 대한 액세스를 제어할 수 있습니다. 예를 들면 CreateGrant 요청의 피부여자 보안 주체가 조건문에 지정된 보안 주체와 일치할 때만 사용자에게 KMS 키 사용 권한을 생성하도록 허용할 수 있습니다.

권한 부여자 위탁자를 지정하려면 AWS 위탁자의 Amazon 리소스 이름(ARN)을 사용합니다. 유효한 보안 주체에는 IAM 사용자 AWS 계정, IAM 역할, 페더레이션 사용자 및 수임된 역할 사용자가 포함됩니다. 위탁자의 ARN 구문에 대한 도움말은 IAM 사용 설명서의 IAM ARN을 참조하세요.

다음 예제 키 정책 문은 `kms:GranteePrincipal` 조건 키를 사용하여 권한의 피부여자 보안 주체가 `LimitedAdminRole`일 때만 KMS 키 권한 부여를 생성합니다.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
  "Action": "kms:CreateGrant",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:GranteePrincipal": "arn:aws:iam::111122223333:role/LimitedAdminRole"
    }
  }
}
```

참고 항목

- kms:GrantConstraintSourceArn
- kms:GrantConstraintType
- kms:GrantIsForAWSResource
- kms:GrantOperations
- kms:GranteeServicePrincipal
- kms:RetiringPrincipal
- kms:RetiringServicePrincipal

kms:KeyAgreementAlgorithm

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
kms:KeyAgreementAlgorithm	문자열	단일 값	DeriveSharedSecret	키 정책 및 IAM 정책

kms:KeyAgreementAlgorithm 조건 키를 사용하면 요청의 KeyAgreementAlgorithm 파라미터 값을 기반으로 DeriveSharedSecret 작업에 대한 액세스를 제어할 수 있습니다. KeyAgreementAlgorithm의 유일한 유효 값은 ECDH입니다.

예를 들어 다음 키 정책 문은 kms:KeyAgreementAlgorithm 조건 키를 사용하여 KeyAgreementAlgorithm이 ECDH가 아닌 한 DeriveSharedSecret에 대한 모든 액세스를 거부합니다.

```
{
       "Effect": "Deny",
       "Principal": {
         "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
       },
       "Action": "kms:DeriveSharedSecret",
       "Resource": "*",
       "Condition": {
            "StringNotEquals": {
               "kms:KeyAgreementAlgorithm": "ECDH"
         }
       }
}
```

참고 항목

- the section called “kms:KeyUsage”

kms:KeyOrigin

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
kms:KeyOrigin	문자열	단일 값	CreateKey KMS 키 리소스 작업	IAM 정책 키 정책 및 IAM 정책

kms:KeyOrigin 조건 키는 작업에 의해 작성되거나 작업에 사용된 KMS 키의 Origin 속성 값에 따라 작업에 대한 액세스를 제어합니다. 그것은 리소스 조건 또는 요청 조건으로 작동합니다.

이 조건 키를 사용하여 요청의 Origin 파라미터 값을 기반으로 CreateKey 작업에 대한 액세스를 제어할 수 있습니다. Origin의 유효한 값은 AWS_KMS, AWS_CLOUDHSM, EXTERNAL_KEY_STORE 및 EXTERNAL입니다.

예를 들어 키 구성 요소가 AWS KMS (AWS_KMS)에서 생성되는 경우에만, CloudHSM 사용자 지정 키 스토어()와 연결된 AWS CloudHSM 클러스터에서 키 구성 요소가 생성되는 경우에만AWS_CLOUDHSM, 외부 키 스토어()에서 키 구성 요소가 생성되는 경우에만 또는 외부 소스(EXTERNAL_KEY_STORE)에서 키 구성 요소를 가져오는 경우에만 KMS 키를 생성할 수 있습니다EXTERNAL.

다음 예제 키 정책 문은 kms:KeyOrigin 조건 키를 사용하여가 키 구성 요소를 생성할 때만 KMS 키를 AWS KMS 생성합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
      },
      "Action": "kms:CreateKey",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
```

```
            "kms:KeyOrigin": "AWS_KMS"
          }
        }
      },
      {
        "Effect": "Allow",
        "Principal": {
          "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
        },
        "Action": [
          "kms:Encrypt",
          "kms:Decrypt",
          "kms:GenerateDataKey",
          "kms:GenerateDataKeyWithoutPlaintext",
          "kms:GenerateDataKeyPair",
          "kms:GenerateDataKeyPairWithoutPlaintext",
          "kms:ReEncrypt*"
        ],
        "Resource": "arn:aws:kms:us-west-2:111122223333:key/*",
        "Condition": {
          "StringEquals": {
            "kms:KeyOrigin": "AWS_CLOUDHSM"
          }
        }
      }
    ]
}
```

또한 `kms:KeyOrigin` 조건 키를 사용하여 작업에 사용되는 KMS 키의 `Origin` 속성을 기반으로 KMS 키를 사용 또는 관리하는 작업에 대한 액세스를 제어할 수도 있습니다. 작업은 특정 KMS 키에 대해 권한이 부여된 KMS 키 리소스 작업이어야 합니다. KMS 키 리소스 작업을 식별하려면 작업 및 리소스 테이블에서, 해당 작업에 대해 `Resources` 열에서 `KMS key` 값을 찾습니다.

예를 들어 다음 IAM 정책에서는 보안 주체가 지정된 KMS 키 리소스 작업을 수행하도록 허용하지만 계정의 사용자 지정 키 스토어에서 생성된 KMS 키에 대해서만 허용합니다.

```
{
  "Effect": "Allow",
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:GenerateDataKey",
```

```
    "kms:GenerateDataKeyWithoutPlaintext",
    "kms:GenerateDataKeyPair",
    "kms:GenerateDataKeyPairWithoutPlaintext",
    "kms:ReEncrypt*"
  ],
  "Resource": "arn:aws:kms:us-west-2:111122223333:key/*",
  "Condition": {
    "StringEquals": {
      "kms:KeyOrigin": "AWS_CLOUDHSM"
    }
  }
}
```

참고 항목

- kms:BypassPolicyLockoutSafetyCheck
- kms:KeySpec
- kms:KeyUsage

kms:KeySpec

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
kms:KeySpec	문자열	단일 값	CreateKey KMS 키 리소스 작업	IAM 정책 키 정책 및 IAM 정책

kms:KeySpec 조건 키는 작업에 의해 작성되거나 작업에 사용된 KMS 키의 KeySpec 속성 값에 따라 작업에 대한 액세스를 제어합니다.

IAM 정책에서 이 조건 키를 사용하여 CreateKey 요청의 KeySpec 파라미터 값을 기반으로 CreateKey 작업에 대한 액세스를 제어할 수 있습니다. 예를 들어 이 조건을 사용하여 사용자가 대칭 암호화 KMS 키만 생성하거나 HMAC KMS 키만 생성하도록 허용할 수 있습니다.

다음 예제 IAM 정책문은 kms:KeySpec 조건 키를 사용하여 보안 주체가 RSA 비대칭 KMS 키만을 생성하도록 허용합니다. 이 권한은 요청의 KeySpec이 RSA_로 시작하는 경우에만 유효합니다.

```
{
  "Effect": "Allow",
  "Action": "kms:CreateKey",
  "Resource": "*",
  "Condition": {
    "StringLike": {
      "kms:KeySpec": "RSA_*"
    }
  }
}
```

또한 kms:KeySpec 조건 키를 사용하여 작업에 사용되는 KMS 키의 KeySpec 속성을 기반으로 KMS 키를 사용 또는 관리하는 작업에 대한 액세스를 제어할 수도 있습니다. 작업은 특정 KMS 키에 대해 권한이 부여된 KMS 키 리소스 작업이어야 합니다. KMS 키 리소스 작업을 식별하려면 작업 및 리소스 테이블에서, 해당 작업에 대한 Resources 열에서 KMS key 값을 찾습니다.

예를 들어 다음 IAM 정책에서는 보안 주체가 지정된 KMS 키 리소스 작업을 수행하도록 허용하지만 계정의 대칭 암호화 KMS 키에 대해서만 허용합니다.

```
{
  "Effect": "Allow",
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:DescribeKey"
  ],
  "Resource": "arn:aws:kms:us-west-2:111122223333:key/*",
  "Condition": {
    "StringEquals": {
      "kms:KeySpec": "SYMMETRIC_DEFAULT"
    }
  }
}
```

참고 항목

- kms:BypassPolicyLockoutSafetyCheck
- kms:CustomerMasterKeySpec(사용되지 않음)
- kms:DataKeyPairSpec

- kms:KeyOrigin
- kms:KeyUsage

kms:KeyUsage

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
kms:KeyUsage	문자열	단일 값	CreateKey KMS 키 리소스 작업	IAM 정책 키 정책 및 IAM 정책

kms:KeyUsage 조건 키는 작업에 의해 작성되거나 작업에 사용된 KMS 키의 KeyUsage 속성 값에 따라 작업에 대한 액세스를 제어합니다.

이 조건 키를 사용하여 요청의 KeyUsage 파라미터 값을 기반으로 CreateKey 작업에 대한 액세스를 제어할 수 있습니다. KeyUsage의 유효한 값은 ENCRYPT_DECRYPT, SIGN_VERIFY, GENERATE_VERIFY_MAC 및 KEY_AGREEMENT입니다.

예를 들어, KeyUsage가 ENCRYPT_DECRYPT인 경우에만 사용자가 KMS 키를 생성하도록 허용하고 KeyUsage가 SIGN_VERIFY인 경우에는 사용자 권한을 거부할 수 있습니다.

다음 예제 IAM 정책 문은 kms:KeyUsage 조건 키를 사용하여 KeyUsage가 ENCRYPT_DECRYPT인 경우에만 사용자가 KMS 키를 생성하도록 허용합니다.

```
{
  "Effect": "Allow",
  "Action": "kms:CreateKey",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:KeyUsage": "ENCRYPT_DECRYPT"
    }
  }
}
```

또한 kms:KeyUsage 조건 키를 사용하여 작업에 사용되는 KMS 키의 KeyUsage 속성을 기반으로 KMS 키를 사용 또는 관리하는 작업에 대한 액세스를 제어할 수도 있습니다. 작업은 특정 KMS 키에 대

해 권한이 부여된 KMS 키 리소스 작업이어야 합니다. KMS 키 리소스 작업을 식별하려면 작업 및 리소스 테이블에서, 해당 작업에 대해 Resources 열에서 KMS key 값을 찾습니다.

예를 들어 다음 IAM 정책에서는 보안 주체가 지정된 KMS 키 리소스 작업을 수행하도록 허용하지만 계정에서 서명 및 확인에 사용되는 KMS 키에 대해서만 허용합니다.

```
{
  "Effect": "Allow",
  "Action": [
    "kms:CreateGrant",
    "kms:DescribeKey",
    "kms:GetPublicKey",
    "kms:ScheduleKeyDeletion"
  ],
  "Resource": "arn:aws:kms:us-west-2:111122223333:key/*",
  "Condition": {
    "StringEquals": {
      "kms:KeyUsage": "SIGN_VERIFY"
    }
  }
}
```

참고 항목

- kms:BypassPolicyLockoutSafetyCheck
- kms:CustomerMasterKeyUsage(사용되지 않음)
- kms:KeyOrigin
- kms:KeySpec

kms:MacAlgorithm

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
kms:MacAlgorithm	문자열	단일 값	GenerateMac VerifyMac	키 정책 및 IAM 정책

`kms:MacAlgorithm` 조건 키를 사용해 요청의 `MacAlgorithm` 파라미터 값을 기반으로 GenerateMac 및 VerifyMac 작업에 대한 액세스를 제어할 수 있습니다.

다음 예제 키 정책은 요청의 MAC 알고리즘이 `HMAC_SHA_384` 또는 `HMAC_SHA_512`인 경우에만 `testers` 역할을 수임할 수 있는 사용자가 HMAC KMS 키를 사용하여 HMAC 태그를 생성하고 확인하도록 허용합니다. 이 정책은 각각 고유한 조건을 가진 두 개의 별도의 정책 문을 사용합니다. 단일 조건 문에 MAC 알고리즘을 두 개 이상 지정하는 경우 조건에는 둘 중 하나가 아니라 두 알고리즘이 모두 필요합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/testers"
      },
      "Action": [
        "kms:GenerateMac",
        "kms:VerifyMac"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "kms:MacAlgorithm": "HMAC_SHA_384"
        }
      }
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/testers"
      },
      "Action": [
        "kms:GenerateMac",
        "kms:VerifyMac"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
```

```
          "kms:MacAlgorithm": "HMAC_SHA_512"
        }
      }
    }
  ]
}
```

참고 항목

- the section called “kms:EncryptionAlgorithm”
- kms:SigningAlgorithm

kms:MessageType

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
kms:MessageType	문자열	단일 값	Sign Verify	키 정책 및 IAM 정책

kms:MessageType 조건 키는 요청의 MessageType 파라미터 값을 기반으로 서명 및 확인 작업에 대한 액세스를 제어합니다. 에 유효한 값은 RAW, DIGEST 및 MessageType입니다EXTERNAL_MU.

예를 들어 다음 키 정책 문은 kms:MessageType 조건 키를 사용하여 사용자가 비대칭 KMS 키를 사용하여 메시지에 서명하도록 허용하지만 메시지 다이제스트는 허용하지 않습니다.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
  "Action": "kms:Sign",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:MessageType": "RAW"
    }
```

```
  }
}
```

참고 항목

- the section called “kms:SigningAlgorithm”

kms:MultiRegion

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
`kms:MultiRegion`	부울	단일 값	`CreateKey` KMS 키 리소스 작업	키 정책 및 IAM 정책

이 조건 키를 사용하여 단일 리전 키 또는 다중 리전 키에서만 작업을 허용할 수 있습니다. `kms:MultiRegion` 조건 키는 KMS 키의 `MultiRegion` 속성 값에 따라 KMS 키 AWS KMS 및 CreateKey 작업에 대한 액세스를 제어합니다. 유효 값은 `true`(다중 리전) 및 `false`(단일 리전)입니다. 모든 KMS 키에는 `MultiRegion` 속성입니다.

예를 들어 다음 IAM 정책문은 `kms:MultiRegion` 조건 키를 사용하여 보안 주체가 단일 리전 키를 생성하도록 허용합니다.

```
{
  "Effect": "Allow",
  "Action": "kms:CreateKey",
  "Resource": "*",
  "Condition": {
    "Bool": {
      "kms:MultiRegion": false
    }
  }
}
```

kms:MultiRegionKeyType

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
kms:MultiRegionKeyType	문자열	단일 값	CreateKey KMS 키 리소스 작업	키 정책 및 IAM 정책

이 조건 키를 사용하여 다중 리전 기본 키 또는 다중 리전 복제본 키에서만 작업을 허용할 수 있습니다. kms:MultiRegionKeyType 조건 키는 KMS 키의 MultiRegionKeyType 속성 AWS KMS 에 따라 KMS 키 및 CreateKey 작업에 대한 액세스를 제어합니다. 유효 값은 PRIMARY 및 REPLICA입니다. 다중 리전 키에만 MultiRegionKeyType 속성이 있습니다.

일반적으로 IAM 정책에서 kms:MultiRegionKeyType 조건 키를 사용하여 여러 KMS 키에 대한 액세스를 제어합니다. 그러나 지정된 다중 리전 키가 기본 또는 복제본으로 변경될 수 있으므로 특정 다중 리전 키가 기본 또는 복제본 키인 경우에만 작업을 허용하려면 키 정책에서 이 조건을 사용할 수 있습니다.

예를 들어 다음 IAM 정책 설명은 kms:MultiRegionKeyType 조건 키를 사용하여 보안 주체가 지정된 AWS 계정의 다중 리전 복제본 키에서만 키 삭제를 예약하고 취소할 수 있도록 합니다.

```
{
  "Effect": "Allow",
  "Action": [
    "kms:ScheduleKeyDeletion",
    "kms:CancelKeyDeletion"
  ],
  "Resource": "arn:aws:kms:*:111122223333:key/*",
  "Condition": {
    "StringEquals": {
      "kms:MultiRegionKeyType": "REPLICA"
    }
  }
}
```

모든 다중 지역 키에 대한 액세스를 허용하거나 거부하려면 kms:MultiRegionKeyType와 함께 두 값 또는 null 값을 모두 사용할 수 있습니다. 그러나, kms:MultiRegion 조건 키를 사용하는 것이 좋습니다.

kms:PrimaryRegion

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
kms:PrimaryRegion	문자열(목록)	단일 값	UpdatePrimaryRegion	키 정책 및 IAM 정책

이 조건 키를 사용하여 UpdatePrimaryRegion 작업에서 대상 리전을 제한할 수 있습니다. 다중 리전 기본 키를 호스팅 AWS 리전 할 수 있는 입니다.

kms:PrimaryRegion 조건 키는 PrimaryRegion 파라미터 변수의 값을 기반으로 UpdatePrimaryRegion 작업에 대한 액세스를 제어합니다. PrimaryRegion 파라미터는 기본으로 승격되는 다중 리전 복제본 키 AWS 리전 의를 지정합니다. 조건의 값은 또는와 같은 하나 이상의 AWS 리전 이름 ap-southeast-2또는 us-east-1와 같은 리전 이름 패턴입니다. eu-*

예를 들어 다음 키 정책 문은 kms:PrimaryRegion 조건 키를 사용하여 보안 주체가 다중 리전 키의 기본 리전을 지정된 4개의 리전 중 하나로 업데이트할 수 있습니다.

```
{
  "Effect": "Allow",
  "Action": "kms:UpdatePrimaryRegion",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/Developer"
  },
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:PrimaryRegion": [
         "us-east-1",
         "us-west-2",
         "eu-west-3",
         "ap-southeast-2"
      ]
    }
  }
```

```
}
```

kms:ReEncryptOnSameKey

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
kms:ReEncryptOnSameKey	부울	단일 값	ReEncrypt	키 정책 및 IAM 정책

이 조건 키를 사용해 요청이 원래 암호화에 사용된 것과 동일한 대상 KMS 키를 지정하는지 여부에 따라 ReEncrypt 작업에 대한 액세스를 제어할 수 있습니다.

예를 들어 다음 키 정책 문은 kms:ReEncryptOnSameKey 조건 키를 사용해 대상 KMS 키가 원래 암호화에 사용된 것과 동일한 경우에만 사용자가 다시 암호화하도록 허용합니다.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
  "Action": "kms:ReEncrypt*",
  "Resource": "*",
  "Condition": {
    "Bool": {
      "kms:ReEncryptOnSameKey": true
    }
  }
}
```

kms:RequestAlias

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
kms:RequestAlias	문자열(목록)	단일 값	암호화 작업 DescribeKey	키 정책 및 IAM 정책

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
			GetPublicKey	

이 조건 키를 사용하여 요청이 KMS 키를 식별하는 데 특정 별칭을 사용하는 경우에만 작업을 허용할 수 있습니다. 이 `kms:RequestAlias` 조건 키는 요청에서 해당 KMS 키를 식별하는 별칭을 기반으로 암호화 작업에 사용되는 KMS 키 `GetPublicKey` 또는 `DescribeKey`에 대한 액세스를 제어합니다. (이 정책 조건은 GenerateRandom 작업이 KMS 키 또는 별칭을 사용하지 않기 때문에 영향을 미치지 않습니다.)

이 조건은에서 속성 기반 액세스 제어(ABAC)를 지원 AWS KMS하므로 KMS 키의 태그 및 별칭을 기반으로 KMS 키에 대한 액세스를 제어할 수 있습니다. 정책 또는 권한 부여를 변경하지 않고 태그 및 별칭을 사용하여 KMS 키에 대한 액세스를 허용할 수 있습니다. 자세한 내용은 에 대한 ABAC AWS KMS 섹션을 참조하세요.

이 정책 조건에서 별칭을 지정하려면 별칭 이름(예: `alias/project-alpha`) 또는 별칭 이름 패턴(예: `alias/*test*`)을 사용합니다. 이 조건 키의 값에 별칭 ARN을 지정할 수 없습니다.

이 조건을 만족시키기 위해 요청의 `KeyId` 파라미터 값은 일치하는 별칭 이름 또는 별칭 ARN이어야 합니다. 요청이 다른 키 식별자를 사용하면 동일한 KMS 키를 식별하더라도 조건을 충족하지 못합니다.

예를 들어 다음 키 정책문은 보안 주체가 KMS 키에서 GenerateDataKey 작업을 호출하도록 허용합니다. 그러나 이는 요청의 `KeyId` 파라미터 값이 `alias/finance-key`이거나 해당 별칭 이름(예: `arn:aws:kms:us-west-2:111122223333:alias/finance-key`)이 있는 별칭 ARN인 경우에만 허용됩니다.

```
{
  "Sid": "Key policy using a request alias condition",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/developer"
  },
  "Action": "kms:GenerateDataKey",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:RequestAlias": "alias/finance-key"
    }
  }
```

```
}
```

이 조건 키를 사용하여 CreateAlias 또는 DeleteAlias와 같은 별칭 작업에 대한 액세스를 제어할 수 없습니다. 별칭 작업에 대한 액세스 제어와 관련된 자세한 내용은 별칭에 대한 액세스 제어 섹션을 참조하세요.

kms:ResourceAliases

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
`kms:ResourceAliases`	문자열(목록)	다중 값	KMS 키 리소스 작업	IAM 정책만

이 조건 키를 사용하여 KMS 키와 연결된 별칭을 기반으로 KMS 키에 대한 액세스를 제어합니다. 작업은 특정 KMS 키에 대해 권한이 부여된 KMS 키 리소스 작업이어야 합니다. KMS 키 리소스 작업을 식별하려면 작업 및 리소스 테이블에서, 해당 작업에 대한 `Resources` 열에서 `KMS key` 값을 찾습니다.

이 조건은 AWS KMS의 속성 기반 액세스 제어(ABAC)를 지원합니다. ABAC를 사용하면 KMS 키에 할당된 태그 및 KMS 키와 연결된 별칭을 기반으로 KMS 키에 대한 액세스를 제어할 수 있습니다. 정책 또는 권한 부여를 변경하지 않고 태그 및 별칭을 사용하여 KMS 키에 대한 액세스를 허용할 수 있습니다. 자세한 내용은 에 대한 ABAC AWS KMS을 참조하세요.

별칭은 AWS 계정 및 리전에서 고유해야 하지만,이 조건을 사용하면 동일한 리전에 있는 여러 KMS 키(`StringLike`비교 연산자 사용) 또는 각 계정마다 다른에 있는 여러 KMS 키에 대한 액세스를 제어할 AWS 리전 수 있습니다.

> **Note**
>
> kms:ResourceAliases 조건은 KMS 키가 KMS 키당 별칭 할당량을 준수하는 경우에만 유효합니다. KMS 키가 이 할당량을 초과하면 `kms:ResourceAliases` 조건에 따라 KMS 키를 사용할 권한이 있는 보안 주체는 KMS 키에 대한 액세스가 거부됩니다.

이 정책 조건에서 별칭을 지정하려면 별칭 이름(예: `alias/project-alpha`) 또는 별칭 이름 패턴(예: `alias/*test*`)을 사용합니다. 이 조건 키의 값에 별칭 ARN을 지정할 수 없습니다. 조건을 충족하려면 작업에 사용된 KMS 키에 지정된 별칭이 있어야 합니다. 작업 요청에서 KMS 키가 식별되는지 여부와 방법은 중요하지 않습니다.

KMS 키와 연결된 별칭 집합을 정책의 별칭 집합과 비교하는 다중값 조건 키입니다. 이러한 집합이 비교되는 방법을 확인하려면 정책 조건에서 `ForAnyValue` 또는 `ForAllValues` 연산자를 제공해야 합니다. 집합 연산자에 대한 자세한 내용은 IAM 사용 설명서의 여러 키 및 값 사용을 참조하세요.

- ForAnyValue: KMS 키와 연결된 별칭이 하나 이상 정책 조건의 별칭과 일치해야 합니다. 다른 별칭은 허용됩니다. KMS 키에 별칭이 없으면 조건이 충족되지 않습니다.
- ForAllValues: KMS 키와 연결된 모든 별칭은 정책의 별칭과 일치해야 합니다. 이 집합 연산자는 KMS 키와 연결된 별칭을 정책 조건의 별칭으로 제한합니다. 별칭은 필요하지 않지만 지정되지 않은 별칭을 금지합니다.

예를 들어 다음 IAM 정책 문은 보안 주체 AWS 계정 가 `finance-key`별칭과 연결된 지정된의 모든 KMS 키에 대해 GenerateDataKey 작업을 호출하도록 허용합니다. (영향을 받는 KMS 키의 키 정책도 보안 주체의 계정에서 이 작업에 사용할 수 있도록 허용해야 합니다.) KMS 키와 연결될 수 있는 여러 별칭 중 하나가`alias/finance-key`인 경우 조건이 충족되었음을 나타내기 위해 조건은 `ForAnyValue` 집합 연산자를 사용합니다.

`kms:ResourceAliases` 조건은 요청이 아닌 리소스를 기반으로 하기 때문에 요청이 키 ID 또는 키 ARN을 사용하여 KMS 키를 식별하는 경우에도 `finance-key` 별칭과 연결된 KMS 키에 대해 `GenerateDataKey` 직접 호출이 성공합니다.

```
{
  "Sid": "AliasBasedIAMPolicy",
  "Effect": "Allow",
  "Action": "kms:GenerateDataKey",
  "Resource": [
    "arn:aws:kms:*:111122223333:key/*",
    "arn:aws:kms:*:444455556666:key/*"
  ],
  "Condition": {
    "ForAnyValue:StringEquals": {
      "kms:ResourceAliases": "alias/finance-key"
    }
  }
}
```

다음 예제 IAM 정책 설명은 보안 주체가 KMS 키를 활성화 및 비활성화하도록 허용하지만 KMS 키의 모든 별칭에 "`Test`"가 포함된 경우에만 가능합니다. 이 정책 설명은 두 가지 조건을 사용합니다. `ForAllValues` 집합 연산자가 있는 조건에서는 KMS 키와 연결된 모든 별칭에 "Test"가 포함되어야 합니다. `ForAnyValue` 집합 연산자가 있는 조건에서는 KMS 키에 "Test"라는 별칭이 하나 이상 있어

야 합니다. ForAnyValue 조건이 없었다면 이 정책문은 보안 주체가 별칭이 없는 KMS 키를 사용하도록 허용했을 것입니다.

```
{
  "Sid": "AliasBasedIAMPolicy",
  "Effect": "Allow",
  "Action": [
    "kms:EnableKey",
    "kms:DisableKey"
  ],
  "Resource": "arn:aws:kms:*:111122223333:key/*",
  "Condition": {
    "ForAllValues:StringLike": {
      "kms:ResourceAliases": [
        "alias/*Test*"
      ]
    },
    "ForAnyValue:StringLike": {
      "kms:ResourceAliases": [
        "alias/*Test*"
      ]
    }
  }
}
```

kms:ReplicaRegion

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
kms:ReplicaRegion	문자열(목록)	단일 값	ReplicateKey	키 정책 및 IAM 정책

이 조건 키를 사용하여 보안 주체 AWS 리전 가 다중 리전 키를 복제할 수 있는를 제한할 수 있습니다. kms:ReplicaRegion 조건 키는 요청의 ReplicaRegion 파라미터 값을 기반으로 ReplicateKey 작업에 대한 액세스를 제어합니다. 이 파라미터는 새 복제본 키에 대한 AWS 리전 을 지정합니다.

조건의 값은 또는와 같은 하나 이상의 AWS 리전 이름 ap-southeast-2또는 us-east-1와 같은 이름 패턴입니다eu-*. 가 AWS 리전 AWS KMS 지원하는 이름 목록은 AWS Key Management Service 의 엔드포인트 및 할당량을 참조하세요 AWS 일반 참조.

예를 들어 다음 키 정책 문은 kms:ReplicaRegion조건 키를 사용하여 ReplicaRegion 파라미터의 값이 지정된 리전 중 하나인 경우에만 보안 주체가 ReplicateKey 작업을 호출하도록 허용합니다.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/Administrator"
  },
  "Action": "kms:ReplicateKey"
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:ReplicaRegion": [
         "us-east-1",
         "eu-west-3",
         "ap-southeast-2"
      ]
    }
  }
}
```

이 조건 키는 ReplicateKey 작업에 대한 액세스만 제어합니다. UpdatePrimaryRegion 작업에 대한 액세스를 제어하려면kms:PrimaryRegion 조건 키를 사용합니다.

kms:RetiringPrincipal

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
kms:RetiringPrincipal	문자열(목록)	단일 값	CreateGrant	키 정책 및 IAM 정책

이 조건 키를 사용하여 요청의 RetiringPrincipal 파라미터 값을 기반으로 CreateGrant 작업에 대한 액세스를 제어할 수 있습니다. 예를 들면 CreateGrant 요청의 RetiringPrincipal이 조건문의 RetiringPrincipal과 일치할 때만 사용자에게 KMS 키 사용 권한을 생성하도록 허용할 수 있습니다.

사용 중지 위탁자를 지정하려면 AWS 위탁자의 Amazon 리소스 이름(ARN)을 사용합니다. 유효한 보안 주체에는 IAM 사용자 AWS 계정, IAM 역할, 페더레이션 사용자 및 수임된 역할 사용자가 포함됩니다. 위탁자의 ARN 구문에 대한 도움말은 IAM 사용 설명서의 IAM ARN을 참조하세요.

다음 예제 키 정책 문을 사용하면 사용자가 KMS 키에 대한 권한을 생성할 수 있습니다. `kms:RetiringPrincipal` 조건 키는 권한을 `CreateGrant` 요청으로 제한하며, 사용 중지 위탁자는 `LimitedAdminRole`입니다.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
  "Action": "kms:CreateGrant",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:RetiringPrincipal": "arn:aws:iam::111122223333:role/LimitedAdminRole"
    }
  }
}
```

참고 항목

- kms:GrantConstraintSourceArn
- kms:GrantConstraintType
- kms:GranteePrincipal
- kms:GranteeServicePrincipal
- kms:GrantIsForAWSResource
- kms:GrantOperations
- kms:RetiringServicePrincipal

kms:GrantConstraintSourceArn

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
`kms:GrantConstraintSourceArn`	ARN	단일 값	`CreateGrant`	키 정책 및 IAM 정책

이 조건 키를 사용하여 요청의 제약 조건 파라미터에 있는 `SourceArn` 값을 기반으로 CreateGrant 작업에 대한 액세스를 제어할 수 있습니다. `SourceArn` 권한 부여 제약 조건은 aws:SourceArn 전역 조건 키를 권한 부여에 효과적으로 적용하는 것입니다.

다음 예제 키 정책 문은 `kms:GrantConstraintSourceArn` 조건 키를 사용하여 권한 부여의 `SourceArn` 제약 조건이 특정 DynamoDB 테이블 ARN 패턴과 일치하는 경우에만 KMS 키에 대한 권한 부여 생성을 허용합니다.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
  "Action": "kms:CreateGrant",
  "Resource": "*",
  "Condition": {
    "ArnLike": {
      "kms:GrantConstraintSourceArn": "arn:aws:dynamodb:us-east-1:111122223333:table/ExampleTable*"
    }
  }
}
```

참고 항목

- kms:GrantConstraintType
- kms:GranteePrincipal
- kms:GranteeServicePrincipal
- kms:GrantIsForAWSResource

- kms:GrantOperations
- kms:RetiringPrincipal
- kms:RetiringServicePrincipal

kms:GranteeServicePrincipal

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
kms:GranteeServicePrincipal	문자열	단일 값	CreateGrant	키 정책 및 IAM 정책

이 조건 키를 사용하여 요청의 GranteeServicePrincipal 파라미터 값을 기반으로 CreateGrant 작업에 대한 액세스를 제어할 수 있습니다. 값은 권한 부여의 피부여자로 지정된 AWS 서비스 보안 주체입니다(예: service-name.amazonaws.com).

다음 예제 키 정책 문은 kms:GranteeServicePrincipal 조건 키를 사용하여 권한 부여의 피부여자 서비스 보안 주체가 인 경우에만 KMS 키에 대한 권한 부여 생성을 허용합니다*service-name*.amazonaws.com.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
  "Action": "kms:CreateGrant",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:GranteeServicePrincipal": "service-name.amazonaws.com"
    }
  }
}
```

참고 항목

- kms:GrantConstraintSourceArn

- kms:GrantConstraintType
- kms:GranteePrincipal
- kms:GrantIsForAWSResource
- kms:GrantOperations
- kms:RetiringPrincipal
- kms:RetiringServicePrincipal

kms:RetiringServicePrincipal

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
`kms:RetiringServicePrincipal`	문자열	단일 값	`CreateGrant`	키 정책 및 IAM 정책

이 조건 키를 사용하여 요청의 RetiringServicePrincipal 파라미터 값을 기반으로 CreateGrant 작업에 대한 액세스를 제어할 수 있습니다. 값은 권한 부여의 사용 중지 AWS 보안 주체로 지정된 서비스 보안 주체입니다(예: `service-name.amazonaws.com`).

다음 예제 키 정책 문은 `kms:RetiringServicePrincipal` 조건 키를 사용하여 권한 부여의 사용 중지 서비스 보안 주체가 인 경우에만 KMS 키에 대한 권한 부여 생성을 허용합니다*`service-name`*`.amazonaws.com`.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
  "Action": "kms:CreateGrant",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:RetiringServicePrincipal": "service-name.amazonaws.com"
    }
  }
```

```
}
```

참고 항목

- kms:GrantConstraintSourceArn
- kms:GrantConstraintType
- kms:GranteePrincipal
- kms:GranteeServicePrincipal
- kms:GrantIsForAWSResource
- kms:GrantOperations
- kms:RetiringPrincipal

kms:RotationPeriodInDays

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
`kms:RotationPeriodInDays`	Numeric	단일 값	`EnableKeyRotation`	키 정책 및 IAM 정책

이 조건 키를 사용하여 위탁자가 EnableKeyRotation 요청의 `RotationPeriodInDays` 파라미터에 지정할 수 있는 값을 제한할 수 있습니다.

는 각 자동 키 교체 날짜 사이의 일수를 `RotationPeriodInDays` 지정합니다.를 AWS KMS 사용하면 90~2,560일의 교체 기간을 지정할 수 있지만 `kms:RotationPeriodInDays` 조건 키를 사용하여 유효한 범위 내에서 최소 교체 기간을 적용하는 등 교체 기간을 추가로 제한할 수 있습니다.

예를 들어 다음 키 정책 문은 `kms:RotationPeriodInDays` 조건 키를 사용하여 키 교체 기간이 180일 이하인 경우에는 위탁자가 키 교체를 활성화하지 못하도록 합니다.

```
{
  "Effect": "Deny",
  "Action": "kms:EnableKeyRotation",
  "Principal": "*",
  "Resource": "*",
```

```
  "Condition" : {
      "NumericLessThanEquals" : {
        "kms:RotationPeriodInDays" : "180"
      }
  }
}
```

kms:ScheduleKeyDeletionPendingWindowInDays

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
kms:ScheduleKeyDeletionPendingWindowInDays	Numeric	단일 값	ScheduleKeyDeletion	키 정책 및 IAM 정책

이 조건 키를 사용하여 보안 주체가 ScheduleKeyDeletion 요청의 PendingWindowInDays 파라미터에 지정할 수 있는 값을 제한할 수 있습니다.

AWS KMS 는 키를 삭제하기 전에 대기할 일수를 PendingWindowInDays 지정합니다. AWS KMS 를 사용하면 7~30일의 대기 기간을 지정할 수 있지만 조건 kms:ScheduleKeyDeletionPendingWindowInDays 키를 사용하여 유효한 범위 내에서 최소 대기 기간을 적용하는 등 대기 기간을 추가로 제한할 수 있습니다.

예를 들어 다음 키 정책 문은 kms:ScheduleKeyDeletionPendingWindowInDays 조건 키를 사용하여 대기 기간이 21일 이하인 경우 보안 주체가 키 삭제 일정을 잡을 수 없도록 합니다.

```
{
  "Effect": "Deny",
  "Action": "kms:ScheduleKeyDeletion",
  "Principal": "*",
  "Resource": "*",
  "Condition" : {
      "NumericLessThanEquals" : {
        "kms:ScheduleKeyDeletionPendingWindowInDays" : "21"
      }
  }
```

```
}
```

kms:SigningAlgorithm

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
kms:SigningAlgorithm	문자열	단일 값	Sign Verify	키 정책 및 IAM 정책

kms:SigningAlgorithm 조건 키를 사용해 요청의 SigningAlgorithm 파라미터 값을 기반으로 Sign 및 Verify 작업에 대한 액세스를 제어할 수 있습니다. 이 조건 키는 외부의 비대칭 KMS 키 페어에서 퍼블릭 키를 사용하여 서명을 확인하는 AWS KMS등 외부에서 수행되는 작업에 영향을 주지 않습니다 AWS KMS.

다음 예제 키 정책은 요청에 사용된 서명 알고리즘이 RSASSA_PSS_SHA512 같은 RSASSA_PSS 알고리즘인 경우에만 testers 역할을 수임할 수 있는 사용자가 KMS 키를 사용하여 메시지에 서명하도록 허용합니다.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/testers"
  },
  "Action": "kms:Sign",
  "Resource": "*",
  "Condition": {
    "StringLike": {
      "kms:SigningAlgorithm": "RSASSA_PSS*"
    }
  }
}
```

참고 항목

- kms:EncryptionAlgorithm
- the section called “kms:MacAlgorithm”
- the section called “kms:MessageType”

kms:TrailingDaysWithoutKeyUsage

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
kms:TrailingDaysWithoutKeyUsage	Numeric	단일 값	DisableKey ScheduleKeyDeletion	키 정책 및 IAM 정책

kms:TrailingDaysWithoutKeyUsage 조건 키는 마지막으로 성공한 암호화 작업 또는 KMS 키의 생성 날짜 또는 키가 사용된 적이 없는 TrackingStartDate 경우 계산된 KMS 키에 대한 암호화 작업이 없는 후행 일수를 나타냅니다. 추적 시작 날짜와 해당 동작이이 조건 키에 미치는 영향에 대한 자세한 내용은 섹션을 참조하세요사용량 추적 기간 이해. 키 정책 및 IAM 정책에서이 조건 키를 사용하여 ScheduleKeyDeletion 및 DisableKey 작업에 대한 액세스를 제어할 수 있습니다.

이 조건 키는 숫자로 된 단일 값 조건입니다. 후행 일수 값은 항상 내림됩니다. 예를 들어 키가 89.9일 전에 마지막으로 사용된 경우 값은 89입니다.

는 다음과 같이 계산kms:TrailingDaysWithoutKeyUsage됩니다.

- 추적TrackingStartDate이 시작된 이후 사용되지 않은 또는 이후에 생성된 키의 경우:

  공식: 현재 날짜 - KeyCreationDate

  예를 들어 현재 날짜가 2026년 4월 30일이고 키가 2026년 4월 20일에 생성된 경우 값은 10입니다.
- 추적이 시작된 이후 사용되지 TrackingStartDate 않은 이전에 생성된 키의 경우:

  공식: 현재 날짜 - TrackingStartDate

  예를 들어 현재 날짜가 2026년 4월 30일이고 TrackingStartDate가 2026년 4월 15일인 경우 키 생성 시기에 관계없이 값은 15입니다.
- 키 생성 시기에 관계없이 추적이 시작된 이후 사용된 키의 경우:

  공식: 현재 날짜 - Timestamp/ KeyLastUsage

다음 예제 키 정책 문은 지난 90일 이내에 키를 사용한 경우 또는 키에 기록된 사용량이 없고 90일 이내에 생성되었거나 90일 이내에 추적이 시작된 경우 ScheduleKeyDeletion 및 DisableKey 작업을

거부합니다. 이렇게 하면 활성 상태이거나 최근에 생성된 키가 실수로 삭제되거나 비활성화되는 것을 방지할 수 있습니다.

```
{
  "Effect": "Deny",
  "Action": [
    "kms:ScheduleKeyDeletion",
    "kms:DisableKey"
  ],
  "Principal": "*",
  "Resource": "*",
  "Condition": {
    "NumericLessThanEquals": {
      "kms:TrailingDaysWithoutKeyUsage": "90"
    }
  }
}
```

Warning

사용하지 않는 키를 삭제할 때 마지막 사용 정보에만 의존하지 마세요. 대신 먼저 키를 비활성화하고 비활성화된 상태에서 키를 사용하려는 시도를 나타내는 `DisabledException` 항목이 있는지 모니터링합니다 AWS CloudTrail . 이렇게 하면 잠재적 종속성과 워크로드 장애를 식별하는 데 도움이 됩니다.는 키에 대한 모든 API 호출의 신뢰할 수 있는 소스로 AWS CloudTrail 남아 있습니다.

참고 항목

- KMS 키로 수행된 마지막 암호화 작업 검사
- kms:ScheduleKeyDeletionPendingWindowInDays

kms:ValidTo

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
kms:ValidTo	타임스탬프	단일 값	ImportKeyMaterial	키 정책 및 IAM 정책

kms:ValidTo 조건 키는 가져온 키 자료가 만료되는 시기를 결정하는 요청의 ValidTo 파라미터 값을 기반으로 ImportKeyMaterial 작업에 대한 액세스를 제어합니다. 이 값은 Unix 시간으로 표현됩니다.

기본적으로 ValidTo 파라미터는 ImportKeyMaterial 요청에서 필요합니다. 하지만 ExpirationModel 파라미터의 값이 KEY_MATERIAL_DOES_NOT_EXPIRE이면 ValidTo 파라미터가 유효하지 않습니다. 또한 kms:ExpirationModel 조건 키를 사용하여 ExpirationModel 파라미터 또는 특정 파라미터 값을 요구할 수도 있습니다.

다음 예제 정책 문을 사용하면 키 재료를 KMS 키로 가져올 수 있습니다. kms:ValidTo 조건 키는 ValidTo 값이 1546257599.0 (2018년 12월 31일 11:59:59 PM)보다 작거나 같은 ImportKeyMaterial 요청에 대한 권한을 제한합니다.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
  "Action": "kms:ImportKeyMaterial",
  "Resource": "*",
  "Condition": {
    "NumericLessThanEquals": {
      "kms:ValidTo": "1546257599.0"
    }
  }
}
```

참고 항목

- kms:ExpirationModel
- kms:WrappingAlgorithm

- kms:WrappingKeySpec

kms:ViaService

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
kms:ViaService	문자열	단일 값	KMS 키 리소스 작업	키 정책 및 IAM 정책

kms:ViaService 조건 키는 KMS 키 사용을 지정된의 요청으로 제한합니다 AWS 서비스. 이 조건 키는 전달 액세스 세션에만 적용됩니다. 각 kms:ViaService 조건 키에 하나 이상의 서비스를 지정할 수 있습니다. 작업은 특정 KMS 키에 대해 권한이 부여된 KMS 키 리소스 작업이어야 합니다. KMS 키 리소스 작업을 식별하려면 작업 및 리소스 테이블에서, 해당 작업에 대해 Resources 열에서 KMS key 값을 찾습니다.

예를 들어 다음 키 정책 설명은 kms:ViaService 조건 키를 사용하여 ExampleRole를 대신하여 미국 서부(오레곤) 리전의 Amazon EC2 또는 Amazon RDS에서 요청이 오는 경우에만 고객 관리형 키를 지정된 작업에 사용할 수 있도록 허용합니다.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:CreateGrant",
    "kms:ListGrants",
    "kms:DescribeKey"
  ],
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:ViaService": [
        "ec2.us-west-2.amazonaws.com",
```

```
        "rds.us-west-2.amazonaws.com"
      ]
    }
  }
}
```

또한 kms:ViaService 조건 키를 사용하여 요청이 특정 서비스에서 이루어지는 경우 KMS 키를 사용할 권한을 거부할 수도 있습니다. 예를 들어 키 정책에서 다음 정책 문은 kms:ViaService 조건 키를 사용하여 요청이 AWS Lambda 에서 ExampleRole 대신 이루어지는 경우 고객 관리형 키를 Encrypt 작업에 대해 사용하는 것을 금지합니다.

```
{
  "Effect": "Deny",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
  "Action": [
    "kms:Encrypt"
  ],
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:ViaService": [
          "lambda.us-west-2.amazonaws.com"
      ]
    }
  }
}
```

Important

kms:ViaService 조건 키를 사용할 경우 서비스가 AWS 계정의 보안 주체 대신 요청을 생성합니다. 이러한 보안 주체는 다음 권한을 가져야 합니다.

- KMS 키를 사용할 수 있는 권한입니다. 보안 주체는 서비스가 보안 주체 대신 고객 관리형 키를 사용할 수 있도록 이러한 권한을 통합 서비스에 부여해야 합니다. 자세한 정보는 AWS 서비스와 함께 AWS KMS 암호화 사용 섹션을 참조하세요.
- 통합 서비스 사용 권한. 와 통합되는 AWS 서비스에 대한 액세스 권한을 사용자에게 부여하는 방법에 대한 자세한 내용은 통합 서비스에 대한 설명서를 AWS KMS참조하세요.

모든 AWS 관리형 키는 키 정책 문서에서 `kms:ViaService` 조건 키를 사용합니다. 이 조건은 KMS 키가 생성된 서비스에서 이루어지는 요청에만 KMS 키를 사용하도록 허용합니다. 에 대한 키 정책을 보려면 GetKeyPolicy 작업을 AWS 관리형 키사용합니다.

`kms:ViaService` 조건 키는 IAM 및 키 정책 설명에서 유효합니다. 사용자가 지정한 서비스가 AWS KMS와 통합되고 `kms:ViaService` 조건 키를 지원해야 합니다.

`kms:ViaService` 조건 키를 지원하는 서비스

다음 표에는 AWS KMS 와 통합되고 고객 관리형 키에서 `kms:ViaService` 조건 키 사용을 지원하는 AWS 서비스가 나열되어 있습니다.이 표의 서비스는 일부 리전에서 사용하지 못할 수 있습니다. 모든 AWS 파티션에서 AWS KMS ViaService 이름의 `.amazonaws.com` 접미사를 사용합니다.

> **Note**
>
> 이 테이블의 모든 데이터를 보려면 가로 또는 세로로 스크롤해야 할 수도 있습니다.

서비스 이름	AWS KMS ViaService 이름
Amazon AI Operations	`aiops.`*`AWS_region`*` .amazonaws.com`
AWS App Runner	`apprunner.`*`AWS_region`*` .amazonaws.com`
AWS AppFabric	`appfabric.`*`AWS_region`*` .amazonaws.com`
Amazon AppFlow	`appflow.`*`AWS_region`*` .amazonaws.com`
AWS Transform MGN	`mgn.`*`AWS_region`*` .amazonaws.com`
Amazon Athena	`athena.`*`AWS_region`*` .amazonaws.com`
AWS Audit Manager	`auditmanager.`*`AWS_region`*` .amazonaws.com`
Amazon Aurora	`rds.`*`AWS_region`*` .amazonaws.com`
AWS Backup	`backup.`*`AWS_region`*` .amazonaws.com`

서비스 이름	AWS KMS ViaService 이름
AWS Backup 게이트웨이	backup-gateway. *AWS_region* .amazonaws.com
Amazon Bedrock 모델 복사	bedrock.*AWS_region* .amazonaws.com
Amazon Chime SDK	chimevoiceconnector. *AWS_region* .amazonaws.com
AWS Clean Rooms ML	cleanrooms-ml. *AWS_region* .amazonaws.com
AWS CodeArtifact	codeartifact. *AWS_region* .amazonaws.com
Amazon CodeGuru Reviewer	codeguru-reviewer. *AWS_region* .amazonaws.com
Amazon Comprehend	comprehend. *AWS_region* .amazonaws.com
Amazon Connect Customer	connect.*AWS_region* .amazonaws.com
Amazon Connect Customer Customer Profiles	profile.*AWS_region* .amazonaws.com
Amazon Q in Connect	wisdom.*AWS_region* .amazonaws.com
AWS Database Migration Service (AWS DMS)	dms.*AWS_region* .amazonaws.com
AWS DeepRacer	deepracer. *AWS_region* .amazonaws.com
AWS Directory Service	directoryservice. *AWS_region* .amazonaws.com
Amazon DocumentDB	docdb-elastic. *AWS_region* .amazonaws.com

서비스 이름	AWS KMS ViaService 이름
Amazon DynamoDB	dynamodb. *AWS_region* .amazonaws.com
Amazon EC2 Systems Manager(SSM)	ssm.*AWS_region* .amazonaws.com
Amazon Elastic Block Store(Amazon EBS)	ec2.*AWS_region* .amazonaws.com (EBS 전용)
Amazon Elastic Container Registry (Amazon ECR)	ecr.*AWS_region* .amazonaws.com
Amazon Elastic File System(Amazon EFS)	elasticfilesystem. *AWS_region* .amazonaws.com
Amazon ElastiCache	조건 키 값에 ViaService 이름을 모두 포함합니다. • elasticache. *AWS_region* .amazonaws.com • dax.*AWS_region* .amazonaws.com
AWS Elemental MediaTailor	mediatailor. *AWS_region* .amazonaws.com
AWS 개체 해결	entityresolution. *AWS_region* .amazonaws.com
Amazon EventBridge	events.*AWS_region* .amazonaws.com
Amazon FinSpace	finspace. *AWS_region* .amazonaws.com
Amazon Forecast	forecast. *AWS_region* .amazonaws.com
Amazon FSx	fsx.*AWS_region* .amazonaws.com
AWS Glue	glue.*AWS_region* .amazonaws.com

서비스 이름	AWS KMS ViaService 이름
AWS Ground Station	groundstation. *AWS_region* .amazonaws.com
Amazon GuardDuty	malware-protection. *AWS_region* .amazonaws.com
AWS HealthLake	healthlake. *AWS_region* .amazonaws.com
AWS IoT SiteWise	iotsitewise. *AWS_region* .amazonaws.com
Amazon Kendra	kendra.*AWS_region* .amazonaws.com
Amazon Keyspaces(Apache Cassandra용)	cassandra. *AWS_region* .amazonaws.com
Amazon Kinesis	kinesis.*AWS_region* .amazonaws.com
Amazon Data Firehose	firehose. *AWS_region* .amazonaws.com
Amazon Kinesis Video Streams	kinesisvideo. *AWS_region* .amazonaws.com
AWS Lambda	lambda.*AWS_region* .amazonaws.com
Amazon Lex	lex.*AWS_region* .amazonaws.com
AWS License Manager	license-manager. *AWS_region* .amazonaws.com
Amazon Location Service	geo.*AWS_region* .amazonaws.com
Amazon Lookout for Equipment	lookoutequipment. *AWS_region* .amazonaws.com

서비스 이름	AWS KMS ViaService 이름
Amazon Lookout for Metrics	lookoutmetrics. *AWS_region* .amazonaws.com
Amazon Lookout for Vision	lookoutvision. *AWS_region* .amazonaws.com
Amazon Macie	macie.*AWS_region* .amazonaws.com
AWS Mainframe Modernization	m2.*AWS_region* .amazonaws.com
AWS Mainframe Modernization 애플리케이션 테스트	apptest.*AWS_region* .amazonaws.com
Amazon Managed Blockchain	managedblockchain. *AWS_region* .amazonaws.com
Amazon Managed Streaming for Apache Kafka(Amazon MSK)	kafka.*AWS_region* .amazonaws.com
Amazon Managed Workflows for Apache Airflow(MWAA)	airflow.*AWS_region* .amazonaws.com
Amazon MemoryDB	memorydb. *AWS_region* .amazonaws.com
Amazon Monitron	monitron. *AWS_region* .amazonaws.com
Amazon MQ	mq.*AWS_region* .amazonaws.com
Amazon Neptune	rds.*AWS_region* .amazonaws.com
Amazon Nimble Studio	nimble.*AWS_region* .amazonaws.com
AWS HealthOmics	omics.*AWS_region* .amazonaws.com
Amazon OpenSearch Service	es.*AWS_region* .amazonaws.com , aoss.*AWS_region* .amazonaws.com

서비스 이름	AWS KMS ViaService 이름
Amazon OpenSearch 사용자 지정 패키지	`custom-packages.` *`AWS_region`* `.amazonaws.com`
AWS Proton	`proton.`*`AWS_region`*` .amazonaws.com`
Amazon Quantum Ledger Database(QLDB)	`qldb.`*`AWS_region`*` .amazonaws.com`
Amazon RDS Performance Insights	`rds.`*`AWS_region`*` .amazonaws.com`
Amazon Redshift	`redshift.` *`AWS_region`* `.amazonaws.com`
Amazon Redshift 쿼리 편집기 V2	`sqlworkbench.` *`AWS_region`* `.amazonaws.com`
Amazon Redshift Serverless	`redshift-serverless.` *`AWS_region`* `.amazonaws.com`
Amazon Rekognition	`rekognition.` *`AWS_region`* `.amazonaws.com`
Amazon Relational Database Service(Amazon RDS)	`rds.`*`AWS_region`*` .amazonaws.com`
Amazon 복제 데이터 스토어	`ards.`*`AWS_region`*` .amazonaws.com`
Amazon SageMaker AI	`sagemaker.` *`AWS_region`* `.amazonaws.com`
AWS Secrets Manager	`secretsmanager.` *`AWS_region`* `.amazonaws.com`
Amazon Security Lake	`securitylake.` *`AWS_region`* `.amazonaws.com`
Amazon Simple Email Service(Amazon SES)	`ses.`*`AWS_region`*` .amazonaws.com`

서비스 이름	AWS KMS ViaService 이름
Amazon Simple Notification Service(Amazon SNS)	sns.*AWS_region* .amazonaws.com
Amazon Simple Queue Service(Amazon SQS)	sqs.*AWS_region* .amazonaws.com
Amazon Simple Storage Service(Amazon S3)	s3.*AWS_region* .amazonaws.com
Amazon S3 Tables	s3tables. *AWS_region* .amazonaws.com
AWS Snowball Edge	importexport. *AWS_region* .amazonaws.com
AWS Step Functions	states.*AWS_region* .amazonaws.com
AWS Storage Gateway	storagegateway. *AWS_region* .amazonaws.com
AWS Systems Manager Incident Manager	ssm-incidents. *AWS_region* .amazonaws.com
AWS Systems Manager Incident Manager 연락처	ssm-contacts. *AWS_region* .amazonaws.com
Amazon Timestream	timestream. *AWS_region* .amazonaws.com
Amazon Translate	translate. *AWS_region* .amazonaws.com
AWS Verified Access	verified-access. *AWS_region* .amazonaws.com
Amazon WorkMail	workmail. *AWS_region* .amazonaws.com
Amazon WorkSpaces	workspaces. *AWS_region* .amazonaws.com

서비스 이름	AWS KMS ViaService 이름
Amazon WorkSpaces Thin Client	`thinclient.` *`AWS_region`* `.amazonaws.com`
Amazon WorkSpaces Web	`workspaces-web.` *`AWS_region`* `.amazonaws.com`
AWS X-Ray	`xray.`*`AWS_region`* `.amazonaws.com`

kms:WrappingAlgorithm

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
`kms:WrappingAlgorithm`	문자열	단일 값	`GetParametersForImport`	키 정책 및 IAM 정책

이 조건 키는 요청의 WrappingAlgorithm 파라미터 값에 따라 GetParametersForImport 작업에 대한 액세스를 제어합니다. 이 조건을 사용하여 보안 주체가 가져오기 프로세스 중에 특정 알고리즘을 사용하여 키 자료를 암호화하도록 요구할 수 있습니다. 다른 래핑 알고리즘을 지정하면 필수 퍼블릭 키 및 가져오기 토큰에 대한 요청이 실패합니다.

다음 예제 키 정책문에서는 `kms:WrappingAlgorithm` 조건 키를 사용하여 `GetParametersForImport` 작업을 호출할 수 있는 사용자 권한을 부여하지만, `RSAES_OAEP_SHA_1` 래핑 알고리즘을 사용하지 못하도록 합니다. `GetParametersForImport` 요청의 `WrappingAlgorithm`이 `RSAES_OAEP_SHA_1`이면 작업이 실패합니다.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
  "Action": "kms:GetParametersForImport",
  "Resource": "*",
  "Condition": {
    "StringNotEquals": {
```

```
      "kms:WrappingAlgorithm": "RSAES_OAEP_SHA_1"
    }
  }
}
```

참고 항목

- kms:ExpirationModel
- kms:ValidTo
- kms:WrappingKeySpec

kms:WrappingKeySpec

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
kms:WrappingKeySpec	문자열	단일 값	GetParametersForImport	키 정책 및 IAM 정책

이 조건 키는 요청의 WrappingKeySpec 파라미터 값에 따라 GetParametersForImport 작업에 대한 액세스를 제어합니다. 가져 오기 프로세스 중에 공개 키의 특정 유형을 사용하는 사용자를 요구하는 이 조건을 사용할 수 있습니다. 요청에서 다른 키 유형을 지정하면 실패합니다.

WrappingKeySpec 파라미터 값에 사용할 수 있는 값은 RSA_2048뿐이므로 실제로 사용자에게 이 값을 실제로 사용하지 못하도록 할 경우 사용자가 GetParametersForImport 작업을 사용할 수 없게 됩니다.

다음 예제 정책 설명은 kms:WrappingAlgorithm 조건 키를 사용하여 요청의 WrappingKeySpec이 RSA_4096이 되도록 요구합니다.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
  "Action": "kms:GetParametersForImport",
```

```
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:WrappingKeySpec": "RSA_4096"
    }
  }
}
```

참고 항목

- kms:ExpirationModel
- kms:ValidTo
- kms:WrappingAlgorithm

AWS KMS 증명된 플랫폼의 조건 키

AWS KMS 는 AWS Nitro Enclaves 및 NitroTPM에 대한 암호화 증명을 지원하는 조건 키를 제공합니다. AWS Nitro Enclaves는 엔클레이브라는 격리된 컴퓨팅 환경을 생성하여 매우 민감한 데이터를 보호하고 처리할 수 있는 Amazon EC2 기능입니다. NitroTPM은 EC2 인스턴스로 유사한 증명 기능을 확장합니다.

서명된 증명 문서와 함께 Decrypt, DeriveSharedSecret, GenerateDataKey, GenerateDataKeyPair 또는 GenerateRandom API 작업을 직접적으로 호출하는 경우 이러한 API는 증명 문서의 퍼블릭 키로 응답의 평문을 암호화하고 평문 대신 사이퍼텍스트를 반환합니다.직접적으로 호출하다 이 사이퍼텍스트는 엔클레이브의 프라이빗 키를 사용해야만 해독할 수 있습니다. 자세한 내용은 에서 암호화 증명 지원 AWS KMS 단원을 참조하십시오.

> **Note**
>
> 키를 생성할 때 키 정책을 제공하지 않으면가 AWS KMS 키를 AWS 생성합니다. 이 기본 키 정책은 KMS 키를 소유한 AWS 계정 에 키에 대한 전체 액세스 권한을 부여하며 계정이 IAM 정책을 사용해 키에 대한 액세스를 허용할 수 있도록 합니다. 이 정책은 Decrypt와 같은 모든 작업을 허용합니다. AWS 에서는 KMS 키 정책에 최소 권한 원칙을 적용하는 것이 좋습니다. 또한 `kms:*`의 KMS 키 정책 작업을 `NotAction:kms:Decrypt`로 수정해 액세스를 제한할 수 있습니다.

다음 조건 키를 사용하면 서명된 증명 문서의 내용에 따라 이러한 작업에 대한 사용 권한을 제한할 수 있습니다. 작업을 허용하기 전에는 증명 문서를 이러한 AWS KMS 조건 키의 값과 AWS KMS 비교합니다.

Nitro Enclaves에 대한 조건 키

다음 조건 키는 Nitro Enclaves 증명에만 사용됩니다.

kms:RecipientAttestation:ImageSha384

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형

`kms:RecipientAttestation:ImageSha384`	문자열	단일 값	`Decrypt`
			`DeriveSharedSecret`
			`GenerateDataKey`
			`GenerateDataKeyPair`
			`GenerateRandom`
				키 정책 및 IAM 정책

`kms:RecipientAttestation:ImageSha384` 조건 키는 요청의 서명된 증명 문서의 이미지 다이제스트가 조건 키의 값과 일치하는 경우 KMS 키를 통해 `Decrypt`, `DeriveSharedSecret`, `GenerateDataKey`, `GenerateDataKeyPair` 및 `GenerateRandom`에 대한 액세스를 제어합니다. `ImageSha384` 값은 증명 문서의 PCR0에 해당합니다. 이 조건 키는 요청의 `Recipient` 파라미터가 AWS Nitro 엔클레이브에 대해 서명된 증명 문서를 지정하는 경우에만 유효합니다.

이 값은 Nitro 엔클레이브에 대한에 AWS KMS 대한 요청의 CloudTrail 이벤트에도 포함됩니다.

예를 들어 다음 키 정책 문은 `data-processing` 역할이 Decrypt, DeriveSharedSecret, GenerateDataKey, GenerateDataKeyPair 및 GenerateRandom 작업에 KMS 키를 사용하도록 허용합니다. `kms:RecipientAttestation:ImageSha384` 조건 키는 요청에 있는 증명 문서의 이미지 다이제스트 값(PCR0)이 조건의 이미지 다이제스트 값과 일치하는 경우에만 작업을 허용합니다. 이 조건

키는 요청의 Recipient 파라미터가 AWS Nitro 엔클레이브에 대해 서명된 증명 문서를 지정하는 경우에만 유효합니다.

AWS Nitro 엔클레이브의 유효한 증명 문서가 요청에 포함되지 않은 경우이 조건이 충족되지 않으므로 권한이 거부됩니다.

```
{
  "Sid" : "Enable enclave data processing",
  "Effect" : "Allow",
  "Principal" : {
    "AWS" : "arn:aws:iam::111122223333:role/data-processing"
  },
  "Action": [
    "kms:Decrypt",
    "kms:DeriveSharedSecret",
    "kms:GenerateDataKey",
    "kms:GenerateDataKeyPair",
    "kms:GenerateRandom"
  ],
  "Resource" : "*",
  "Condition": {
    "StringEqualsIgnoreCase": {
      "kms:RecipientAttestation:ImageSha384":
 "9fedcba8abcdef7abcdef6abcdef5abcdef4abcdef3abcdef2abcdef1abcdef1abcdef0abcdef1abcdef2abcdef3a
    }
  }
}
```

kms:RecipientAttestation:PCR<PCR_ID>

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
kms:RecipientAttestation:PCR<PCR_ID>	문자열	단일 값	Decrypt DeriveSharedSecret GenerateDataKey	키 정책 및 IAM 정책

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
			GenerateDataKeyPair GenerateRandom	

kms:RecipientAttestation:PCR<PCR_ID> 조건 키는 요청의 서명된 증명 문서의 플랫폼 구성 레지스터(PCR)가 조건 키의 PCR과 일치하는 경우에만 KMS 키를 통해 Decrypt, DeriveSharedSecret, GenerateDataKey, GenerateDataKeyPair 및 GenerateRandom에 대한 액세스를 제어합니다. 이 조건 키는 요청의 Recipient 파라미터가 AWS Nitro 엔클레이브에서 서명된 증명 문서를 지정하는 경우에만 유효합니다.

이 값은 Nitro 엔클레이브에 AWS KMS 대한 요청을 나타내는 CloudTrail 이벤트에도 포함됩니다.

PCR 값을 지정하려면 다음 형식을 사용합니다. PCR ID를 조건 키 이름에 연결합니다. 6개의 엔클레이브 측정값 중 하나를 식별하는 PCR ID 또는 특정 사용 사례에 맞게 정의한 사용자 지정 PCR ID를 지정할 수 있습니다. PCR 값은 최대 96바이트의 소문자 16진수 문자열이어야 합니다.

```
"kms:RecipientAttestation:PCRPCR_ID": "PCR_value"
```

예를 들어, 다음 조건 키는 PCR1의 특정 값을 지정합니다. 이 값은 엔클레이브 및 부트스트랩 프로세스에 사용되는 커널의 해시에 해당합니다.

```
kms:RecipientAttestation:PCR1:
 "abc1abcdef2abcdef3abcdef4abcdef5abcdef6abcdef7abcdef8abcdef9abcdef8abcdef7abcdef6abcdef5abcde
```

다음 예제 키 정책문은 data-processing 역할이 Decrypt 작업에 KMS 키를 사용하도록 허용합니다.

이 명령문의 kms:RecipientAttestation:PCR 조건 키는 요청의 서명된 증명 문서의 PCR1 값이 조건의 kms:RecipientAttestation:PCR1 값과 일치하는 경우에만 작업을 허용합니다. StringEqualsIgnoreCase 정책 연산자를 사용하여 PCR 값의 대소문자를 구분하지 않는 비교를 요구합니다.

요청에 증명 문서가 포함되어 있지 않으면 이 조건이 충족되지 않으므로 사용 권한이 거부됩니다.

```
{
```

```
  "Sid" : "Enable enclave data processing",
  "Effect" : "Allow",
  "Principal" : {
    "AWS" : "arn:aws:iam::111122223333:role/data-processing"
  },
  "Action": "kms:Decrypt",
  "Resource" : "*",
  "Condition": {
    "StringEqualsIgnoreCase": {
      "kms:RecipientAttestation:PCR1":
 "abc1de4f2dcf774f6e3b679f62e5f120065b2e408dcea327bd1c9dddaea6664e7af79355814748447674530 82c6f1
    }
  }
}
```

NitroTPM용 조건 키

다음 조건 키는 NitroTPM 증명에만 사용됩니다.

kms:RecipientAttestation:NitroTPMPCR<PCR_ID>

AWS KMS 조건 키	조건 유형	값 유형	API 작업	정책 유형
kms:RecipientAttestation:NitroTPMPCR<PCR_ID>	문자열	단일 값	Decrypt DeriveSharedSecret GenerateDataKey GenerateDataKeyPair GenerateRandom	키 정책 및 IAM 정책

kms:RecipientAttestation:NitroTPMPCR<PCR_ID> 조건 키는 요청의 서명된 증명 문서의 플랫폼 구성 레지스터(PCR)가 조건 키의 PCR과 일치하는 경우에만 KMS 키를 통해 Decrypt,

DeriveSharedSecret, GenerateDataKey, GenerateDataKeyPair 및 GenerateRandom에 대한 액세스를 제어합니다. 이 조건 키는 요청의 Recipient 파라미터가 NitroTPM의 서명된 증명 문서를 지정하는 경우에만 유효합니다.

이 값은 NitroTPM에 AWS KMS 대한 요청을 나타내는 CloudTrail 이벤트에도 포함됩니다.

PCR 값을 지정하려면 다음 형식을 사용합니다. PCR ID를 조건 키 이름에 연결합니다. PCR 값은 최대 96바이트의 소문자 16진수 문자열이어야 합니다.

```
"kms:RecipientAttestation:NitroTPMPCRPCR_ID": "PCR_value"
```

예를 들어 다음 조건 키는 PCR4에 대한 특정 값을 지정합니다.

```
kms:RecipientAttestation:NitroTPMPCR4:
 "abc1abcdef2abcdef3abcdef4abcdef5abcdef6abcdef7abcdef8abcdef9abcdef8abcdef7abcdef6abcdef5abcde
```

다음 예제 키 정책문은 data-processing 역할이 Decrypt 작업에 KMS 키를 사용하도록 허용합니다.

이 문의 kms:RecipientAttestation:NitroTPMPCR 조건 키는 요청의 서명된 증명 문서의 PCR4 값이 조건의 kms:RecipientAttestation:NitroTPMPCR4 값과 일치하는 경우에만 작업을 허용합니다. StringEqualsIgnoreCase 정책 연산자를 사용하여 PCR 값의 대소문자를 구분하지 않는 비교를 요구합니다.

요청에 증명 문서가 포함되어 있지 않으면 이 조건이 충족되지 않으므로 사용 권한이 거부됩니다.

```
{
  "Sid" : "Enable NitroTPM data processing",
  "Effect" : "Allow",
  "Principal" : {
    "AWS" : "arn:aws:iam::111122223333:role/data-processing"
  },
  "Action": "kms:Decrypt",
  "Resource" : "*",
  "Condition": {
    "StringEqualsIgnoreCase": {
      "kms:RecipientAttestation:NitroTPMPCR4":
 "abc1de4f2dcf774f6e3b679f62e5f120065b2e408dcea327bd1c9dddaea6664e7af7935581474844767453082c6f1
    }
  }
}
```

최소 권한

KMS 키는 민감한 정보를 보호하므로 최소 권한 액세스 원칙을 따르는 것이 좋습니다. 키 정책을 정의할 때 작업을 수행하는 데 필요한 최소 권한을 위임합니다. 추가 IAM 정책으로 권한을 더 제한하려는 경우에만 KMS 키 정책에 대한 모든 작업(`kms:*`)을 허용합니다. IAM 정책을 사용하여 권한을 관리하려는 경우 IAM 정책을 생성하여 IAM 위탁자에 연결할 수 있는 사용자를 제한하고 정책 변경을 모니터링합니다.

키 정책과 IAM 정책 모두에서 모든 작업(`kms:*`)을 허용하는 경우 위탁자는 KMS 키에 대한 관리 및 사용 권한을 모두 갖습니다. 보안 모범 사례로, 이러한 권한을 특정 위탁자에게만 위임하는 것이 좋습니다. 키 정책에서 위탁자의 이름을 명시적으로 지정하거나 IAM 정책이 연결되는 위탁자를 제한하여 이 작업을 수행할 수 있습니다. 조건 키를 사용하여 권한을 제한할 수도 있습니다. 예를 들어 `aws:PrincipalTag`를 사용하면 API 직접 호출을 하는 위탁자가 조건 규칙에 지정된 태그를 가진 경우 모든 작업을 허용하도록 설정할 수 있습니다.

정책 설명의 평가 방법을 이해하는 데 도움이 필요하면 IAM 사용 설명서의 정책 평가 로직을 AWS참조하세요. 정책을 작성하기 전에 이 주제를 검토하여 정책이 의도치 않은 영향(예: 접근 권한이 없어야 할 주체에게 접근 권한을 제공하는 경우)을 미칠 가능성을 줄이는 것이 좋습니다.

> **Tip**
>
> 비프로덕션 환경에서 애플리케이션을 테스트할 때는 IAM Access Analyzer를 사용하면 IAM 정책에 최소 권한을 적용하는 데 도움이 됩니다.

IAM 역할 대신 IAM 사용자를 사용하는 경우 장기 자격 증명의 취약성을 완화하기 위해 AWS 다중 인증(MFA)을 활성화하는 것이 좋습니다. MFA를 사용하여 다음을 수행할 수 있습니다.

- 사용자에게 키 삭제 예약과 같은 권한 있는 작업을 수행하기 전에 MFA를 사용하여 자격 증명을 검증하도록 요구합니다.
- 관리자 계정 암호 및 MFA 디바이스의 소유권을 개인 간에 분할하여 분할 권한 부여를 구현합니다.

자세히 알아보기

- AWS 직무에 대한 관리형 정책
- Techniques for writing least privilege IAM policies

최소 권한 구현

AWS 서비스에 KMS 키를 사용할 수 있는 권한을 부여할 때 서비스가 사용자를 대신하여 액세스해야 하는 리소스에 대해서만 권한이 유효한지 확인합니다. 이 최소 권한 전략은 AWS 서비스 간에 요청이 전달될 때 KMS 키의 무단 사용을 방지하는 데 도움이 됩니다.

최소 권한 전략을 구현하려면 AWS KMS 암호화 컨텍스트 조건 키와 글로벌 소스 ARN 또는 소스 계정 조건 키를 사용하는 것이 좋습니다.

암호화 컨텍스트 조건 키 사용

리소스를 사용할 AWS KMS 때 최소 권한 권한을 구현하는 가장 효과적인 방법은 보안 주체가 AWS KMS 암호화 작업을 호출할 수 있도록 허용하는 kms:EncryptionContext:context-key 또는 kms:EncryptionContextKeys 조건 키를 정책에 포함하는 것입니다. 이러한 조건 키는 리소스가 암호화될 때 암호문에 바인딩된 암호화 컨텍스트에 권한을 연결하기 때문에 특히 효과적입니다.

정책 설명의 작업이 CreateGrant이거나 GenerateDataKey 또는 Decrypt와 같은 작업과 같은 `EncryptionContext` 파라미터를 사용하는 AWS KMS 대칭 암호화 작업인 경우에만 암호화 컨텍스트 조건 키를 사용합니다. (지원되는 작업 목록은 kms:EncryptionContext:context-key 또는 kms:EncryptionContextKeys 섹션 참조) 이러한 조건 키를 사용하여 DescribeKey와 같은 다른 작업을 허용하는 경우 권한이 거부됩니다.

리소스를 암호화할 때 서비스에서 사용하는 암호화 컨텍스트로 값을 설정합니다. 이 정보는 일반적으로 서비스 문서의 보안 장에서 확인할 수 있습니다. 예를 들어 AWS Proton의 암호화 컨텍스트는 AWS Proton 리소스와 관련 템플릿을 식별합니다. AWS Secrets Manager 암호화 컨텍스트는 보안 암호와 해당 버전을 식별합니다. Amazon Location에 대한 암호화 컨텍스트는 추적기 또는 컬렉션을 식별합니다.

다음 예제 키 정책문은 Amazon Location Service에서 승인된 사용자를 대신하여 권한 부여를 생성하도록 허용합니다. 이 정책문은 kms:ViaService, kms:CallerAccount 및 `kms:EncryptionContext:`*`context-key`* 조건 키를 사용하여 권한을 제한해 특정 추적기 리소스에 권한을 연결합니다.

```
{
  "Sid": "Allow Amazon Location to create grants on behalf of authorized users",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/LocationTeam"
  },
  "Action": "kms:CreateGrant",
```

```
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:ViaService": "geo.us-west-2.amazonaws.com",
      "kms:CallerAccount": "111122223333",
      "kms:EncryptionContext:aws:geo:arn": "arn:aws:geo:us-west-2:111122223333:tracker/
SAMPLE-Tracker"
    }
  }
}
```

`aws:SourceArn` 또는 `aws:SourceAccount` 조건 키 사용

키 정책 문의 위탁자가 AWS 서비스 위탁자인 경우, `kms:EncryptionContext:`*context-key* 조건 키에 추가로 aws:SourceArn 또는 aws:SourceAccount 전역 조건 키를 사용하는 것이 좋습니다. ARN 및 계정 값은 요청이 다른 AWS 서비스 AWS KMS 에서 로 오는 경우에만 권한 부여 컨텍스트에 포함됩니다. 이러한 조건 조합은 최소 권한을 구현하고 잠재적 혼동된 대리자 시나리오를 방지합니다. 서비스 보안 주체는 일반적으로 키 정책에서 보안 주체로 사용되지 않지만와 같은 일부 AWS 서비스에서는 보안 주체가 AWS CloudTrail필요합니다.

`aws:SourceArn` 또는 `aws:SourceAccount` 전역 조건 키를 사용하려면 값을 Amazon 리소스 이름(ARN) 또는 암호화되는 리소스의 계정으로 설정합니다. 예를 들어, 추적을 암호화하기 위해 AWS CloudTrail 권한을 부여하는 키 정책문에서 `aws:SourceArn`의 값을 추적의 ARN으로 설정합니다. 가능하면 더 구체적인 `aws:SourceArn`을 사용합니다. 값을 ARN 또는 와일드카드 문자가 있는 ARN 패턴으로 설정합니다. 리소스의 ARN을 모르면 `aws:SourceAccount`를 대신 사용합니다.

> **Note**
>
> 리소스 ARN에 AWS KMS 키 정책에서 허용되지 않는 문자가 포함된 경우 `aws:SourceArn` 조건 키 값에 해당 리소스 ARN을 사용할 수 없습니다. 대신 `aws:SourceAccount` 조건 키를 사용합니다. 키 정책 문서 규칙에 대한 자세한 내용은 키 정책 형식 섹션을 참조하세요.

다음 예제 키 정책에서 권한을 가져오는 보안 주체는 AWS CloudTrail 서비스 주체인 `cloudtrail.amazonaws.com`입니다. 최소 권한을 구현하기 위해 이 정책은 `aws:SourceArn` 및 `kms:EncryptionContext:`*context-key* 조건 키를 사용합니다. 정책문은 CloudTrail이 KMS 키를 사용하여 추적을 암호화하는 데 사용하는 데이터 키를 생성하도록 허용합니다. `aws:SourceArn` 및 `kms:EncryptionContext:`*context-key* 조건은 독립적으로 평가됩니다. 지정된 작업에 KMS 키를 사용하기 위한 요청은 두 조건을 모두 충족해야 합니다.

서비스의 권한을 예제 계정(111122223333) 및 us-west-2 리전의 finance 추적으로 제한하기 위해, 이 정책문은 aws:SourceArn 조건 키를 특정 추적의 ARN에 대해 설정합니다. 조건문은 ArnEquals 연산자를 사용하여 일치하는 경우 ARN의 모든 요소가 독립적으로 평가되도록 합니다. 또한 예제에서는 kms:EncryptionContext:*context-key* 조건 키를 사용하여 권한을 특정 계정 및 리전의 추적으로 제한합니다.

이 키 정책을 사용하기 전에 예제 계정 ID, 리전 및 추적 이름을 계정의 유효한 값으로 바꿉니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCloudTrailToEncryptLogs",
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudtrail.amazonaws.com"
      },
      "Action": "kms:GenerateDataKey",
      "Resource": "*",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": [
            "arn:aws:cloudtrail:us-west-2:111122223333:trail/finance"
          ]
        },
        "StringLike": {
          "kms:EncryptionContext:aws:cloudtrail:arn": [
            "arn:aws:cloudtrail:*:111122223333:trail/*"
          ]
        }
      }
    }
  ]
}
```

에 대한 ABAC AWS KMS

속성 기반 액세스 제어(ABAC)는 속성을 기반으로 권한을 정의하는 권한 부여 전략입니다.는 KMS 키와 연결된 태그 및 별칭을 기반으로 고객 관리형 키에 대한 액세스를 제어할 수 있도록 하여 ABAC를 AWS KMS 지원합니다. 에서 ABAC를 활성화하는 태그 및 별칭 조건 키는 보안 주체가 정책을 편집하거나 권한 부여를 관리하지 않고 KMS 키를 사용하도록 권한을 부여하는 강력하고 유연한 방법을 AWS KMS 제공합니다. 그러나 보안 주체에 실수로 액세스가 허용되거나 거부되지 않도록 이러한 기능을 주의해서 사용해야 합니다.

ABAC를 사용하는 경우 태그 및 별칭 관리 권한이 이제 액세스 제어 권한이라는 점에 유의하십시오. 태그 또는 별칭에 따라 달라지는 정책을 배포하기 전에 모든 KMS 키의 기존 태그와 별칭을 알고 있어야 합니다. 별칭을 추가, 삭제 및 업데이트하거나 키에 태그를 지정하고 태그를 해제할 때 합리적인 예방 조치를 취해야 합니다. 태그와 별칭을 관리할 수 있는 권한을 필요한 보안 주체에게만 부여하고 관리할 수 있는 태그와 별칭을 제한합니다.

> **참고**
>
> 에 ABAC를 사용할 때는 보안 주체에게 태그와 AWS KMS별칭을 관리할 수 있는 권한을 부여하는 데 주의해야 합니다. 태그 또는 별칭을 변경하면 KMS 키에 대한 사용 권한을 허용하거나 거부할 수 있습니다. 키 정책을 변경하거나 권한 부여를 생성할 권한이 없는 키 관리자는 태그 또는 별칭을 관리할 권한이 있는 경우 KMS 키에 대한 액세스를 제어할 수 있습니다.
> 태그 및 별칭 변경으로 KMS 키 인증에 영향을 미치는 데 최대 5분이 소요될 수 있습니다. 최근 변경 사항은 권한 부여에 영향을 미치기 전에 API 작업에서 볼 수 있습니다.
> 별칭을 기반으로 KMS 키에 대한 액세스를 제어하려면 조건 키를 사용해야 합니다. 별칭을 사용하여 정책문의 `Resource` 요소에서 KMS 키를 나타낼 수 없습니다. 별칭이 `Resource` 요소에 나타나면 정책문이 연결된 KMS 키가 아니라 별칭에 적용됩니다.

자세히 알아보기

- 예제를 포함하여 ABAC AWS KMS 지원에 대한 자세한 내용은 별칭을 사용하여 KMS 키에 대한 액세스 제어 및 섹션을 참조하세요태그를 사용하여 KMS 키에 대한 액세스 제어.
- 태그를 사용하여 리소스에 대한 액세스를 AWS 제어하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 ABAC는 무엇입니까 AWS? 및 리소스 태그를 사용하여 리소스에 대한 액세스 제어를 참조하세요 AWS.

에 대한 ABAC 조건 키 AWS KMS

태그 및 별칭을 기반으로 KMS 키에 대한 액세스 권한을 부여하려면 키 정책 또는 IAM 정책에서 다음 조건 키를 사용합니다.

ABAC 조건 키	설명	정책 유형	AWS KMS 작업
aws:ResourceTag	KMS 키의 태그(키 및 값)가 정책의 태그 (키 및 값) 또는 태그 패턴과 일치합니다.	IAM 정책만	KMS 키 리소스 작업 [2]
aws:RequestTag/tag-key	요청의 태그(키 및 값)가 정책의 태그(키 및 값) 또는 태그 패턴과 일치합니다.	주요 정책 및 IAM 정책[1]	TagResource, UntagResource
aws:TagKeys	요청의 태그 키가 정책의 태그 키와 일치합니다.	주요 정책 및 IAM 정책[1]	TagResource, UntagResource
kms:ResourceAliases	KMS 키와 연결된 별칭은 정책의 별칭 또는 별칭 패턴과 일치합니다.	IAM 정책만	KMS 키 리소스 작업 [2]
kms:RequestAlias	요청의 KMS 키를 나타내는 별칭은 정책의 별칭 또는 별칭 패턴과 일치합니다.	주요 정책 및 IAM 정책[1]	암호화 작업, DescribeKey, GetPublicKey

[1]키 정책에서 사용할 수 있는 모든 조건 키는 IAM 정책에서도 사용할 수 있지만 키 정책에서 허용하는 경우에만 가능합니다.

[2]KMS 키 리소스 작업은 특정 KMS 키에 대해 권한이 부여된 작업입니다. KMS 키 리소스 작업을 식별하려면 AWS KMS 권한 테이블에서 작업의 `Resources` 열에서 KMS 키 값을 찾습니다.

예를 들어 이러한 조건 키를 사용하여 다음 정책을 생성할 수 있습니다.

- 특정 별칭 또는 별칭 패턴이 있는 KMS 키를 사용할 수 있는 권한을 허용하는 `kms:ResourceAliases`가 포함된 IAM 정책. 이는 태그에 의존하는 정책과 약간 다릅니다. 정책에서 별칭 패턴을 사용할 수 있지만 각 별칭은 AWS 계정 및 리전에서 고유해야 합니다. 이렇게 하면 정책 설명에 KMS 키의 키 ARN을 나열하지 않고 선택한 KMS 키 집합에 정책을 적용할 수 있습니다. 집합에서 KMS 키를 추가하거나 제거하려면 KMS 키의 별칭을 변경합니다.
- 보안 주체가 `Encrypt` 작업에서 KMS 키를 사용하도록 허용하지만 `Encrypt` 요청이 해당 별칭을 사용하여 KMS 키를 식별하는 경우에만 허용하는 `kms:RequestAlias`가 포함된 키 정책.
- 특정 태그 키 및 태그 값과 함께 KMS 키를 사용할 수 있는 권한을 거부하는 `aws:ResourceTag/`*`tag-key`*가 포함된 IAM 정책. 이렇게 하면 정책문에 KMS 키의 키 ARN을 나열하지 않고 선택한 KMS 키 집합에 정책을 적용할 수 있습니다. KMS 키를 집합에서 추가하거나 제거하려면 KMS 키에 태그를 지정하거나 태그를 해제합니다.
- 보안 주체가 `"Purpose"="Test"` KMS 키 태그만 삭제할 수 있도록 허용하는 `aws:RequestTag/`*`tag-key`*가 포함된 IAM 정책.
- `Restricted` 태그 키로 KMS 키에 태그를 지정하거나 태그를 해제할 수 있는 권한을 거부하는 `aws:TagKeys`가 포함된 IAM 정책.

ABAC는 액세스 관리를 유연하고 확장 가능하게 합니다. 예를 들어 `aws:ResourceTag/`*`tag-key`* 조건 키를 사용하여 KMS 키에 `Purpose=Test` 태그가 있는 경우에만 보안 주체가 지정된 작업에 KMS 키를 사용하도록 허용하는 IAM 정책을 생성할 수 있습니다. 이 정책은 AWS 계정의 모든 리전에 있는 모든 KMS 키에 적용됩니다.

사용자 또는 역할에 연결된 경우 다음 IAM 정책은 보안 주체가 지정된 작업에 대해 `Purpose=Test` 태그가 있는 모든 기존 KMS 키를 사용할 수 있도록 허용합니다. 새 KMS 키 또는 기존 KMS 키에 대한 액세스를 제공하기 위해 정책을 변경할 필요가 없습니다. `Purpose=Test` 태그를 KMS 키에 지정하기만 하면 됩니다. 마찬가지로 `Purpose=Test` 태그가 있는 KMS 키에서 이 액세스 권한을 제거하려면 태그를 편집하거나 삭제합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AliasBasedIAMPolicy",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
```

```
        "kms:Encrypt",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": "arn:aws:kms:*:111122223333:key/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Purpose": "Test"
        }
      }
    }
  ]
}
```

그러나 이 기능을 사용하는 경우 태그와 별칭을 관리할 때는 주의해야 합니다. 태그 또는 별칭을 추가, 변경 또는 삭제하면 실수로 KMS 키에 대한 액세스를 허용하거나 거부할 수 있습니다. 키 정책을 변경하거나 권한 부여를 생성할 권한이 없는 키 관리자는 태그 및 별칭을 관리할 권한이 있는 경우 KMS 키에 대한 액세스를 제어할 수 있습니다. 이 위험을 완화하기 위해 별칭과 태그 관리 권한을 제한하는 것이 좋습니다. 예를 들어, 선택된 보안 주체만 `Purpose=Test` 태그를 관리하도록 허용할 수 있습니다. 자세한 내용은 별칭을 사용하여 KMS 키에 대한 액세스 제어 및 태그를 사용하여 KMS 키에 대한 액세스 제어 섹션을 참조하세요.

태그 또는 별칭?

AWS KMS 는 태그 및 별칭이 있는 ABAC를 지원합니다. 두 옵션 모두 유연하고 확장 가능한 액세스 제어 전략을 제공하지만 서로 약간 다릅니다.

특정 사용 패턴에 따라 태그를 사용하거나 별칭을 AWS 사용하기로 결정할 수 있습니다. 예를 들어 대부분의 관리자에게 이미 태그 사용 권한을 부여한 경우 별칭을 기반으로 권한 부여 전략을 제어하는 것이 더 쉬울 수 있습니다. 또는 KMS 키당 별칭 할당량에 근접한 경우 태그 기반 권한 부여 전략을 선호할 수 있습니다.

일반적으로 다음과 같은 이점이 있습니다.

태그 기반 액세스 제어의 이점

- 다양한 유형의 AWS 리소스에 대해 동일한 권한 부여 메커니즘.

  동일한 태그 또는 태그 키를 사용하여 Amazon Relational Database Service(RDS) 클러스터, Amazon Elastic Block Store(Amazon EBS) 볼륨 및 KMS 키와 같은 여러 리소스 유형에 대한 액세스

를 제어할 수 있습니다. 이 기능을 사용하면 기존의 역할 기반 액세스 제어보다 유연한 여러 가지 권한 부여 모델을 사용할 수 있습니다.

- KMS 키 그룹에 대한 액세스 권한을 부여합니다.

  태그를 사용하여 동일한 AWS 계정 및 리전의 KMS 키 그룹에 대한 액세스를 관리할 수 있습니다. 선택한 KMS 키에 동일한 태그 또는 태그 키를 할당합니다. 그런 다음 태그 또는 태그 키를 기반으로 유지 관리가 간편한 정책 설명을 만듭니다. 인증 그룹에서 KMS 키를 추가하거나 제거하려면 태그를 추가하거나 제거하십시오. 정책을 편집할 필요가 없습니다.

별칭 기반 액세스 제어의 이점

- 별칭을 기반으로 암호화 작업에 대한 액세스 권한을 부여합니다.

  aws:RequestTag/tag-key를 비롯해 속성에 대한 대부분의 요청 기반 정책 조건은 속성을 추가, 편집 또는 삭제하는 작업에만 영향을 미칩니다. 그러나 kms:RequestAlias 조건 키는 요청에서 KMS 키를 식별하는 데 사용되는 별칭을 기반으로 암호화 작업에 대한 액세스를 제어합니다. 예를 들어 보안 주체에게 `Encrypt` 작업에서 KMS 키를 사용할 수 있는 권한을 부여할 수 있지만 `KeyId` 매개 변수의 값이 `alias/restricted-key-1`인 경우에만 가능합니다. 이 조건을 충족하려면 다음 사항이 모두 필요합니다.

  - KMS 키는 해당 별칭과 연결되어 있어야 합니다.
  - 요청은 별칭을 사용하여 KMS 키를 식별해야 합니다.
  - 보안 주체는 `kms:RequestAlias` 조건에 따라 KMS 키를 사용할 수 있는 권한이 있어야 합니다.

  이는 애플리케이션에서 일반적으로 별칭 이름이나 별칭 ARN을 사용하여 KMS 키를 참조하는 경우에 특히 유용합니다.

- 매우 제한된 권한을 제공합니다.

  별칭은 AWS 계정 및 리전에서 고유해야 합니다. 따라서 보안 주체에 별칭을 기반으로 KMS 키에 대한 액세스 권한을 부여하는 것이 태그를 기반으로 액세스 권한을 부여하는 것보다 훨씬 더 제한적일 수 있습니다. 별칭과 달리 동일한 계정 및 리전에 있는 여러 KMS 키에 태그를 할당할 수 있습니다. 원하는 경우 `alias/test*`와 같은 별칭 패턴을 사용하여 보안 주체에게 동일한 계정 및 리전의 KMS 키 그룹에 대한 액세스 권한을 부여할 수 있습니다. 그러나 특정 별칭에 대한 액세스를 허용하거나 거부하면 KMS 키에 대한 매우 엄격한 제어가 가능합니다.

에 대한 ABAC 문제 해결 AWS KMS

태그 및 별칭을 기반으로 KMS 키에 대한 액세스를 제어하는 것은 편리하고 강력합니다. 그러나 몇 가지 예측 가능한 오류가 발생하기 쉽습니다.

태그 변경으로 인해 액세스 변경

태그가 삭제되거나 해당 값이 변경되면 해당 태그만 기반으로 KMS 키에 액세스할 수 있는 보안 주체는 KMS 키에 대한 액세스가 거부됩니다. 거부 정책문에 포함된 태그가 KMS 키에 추가되는 경우에도 이 문제가 발생할 수 있습니다. KMS 키에 정책 관련 태그를 추가하면 KMS 키에 대한 액세스가 거부되어야 하는 보안 주체에 액세스가 허용될 수 있습니다.

예를 들어 보안 주체가 다음 예제 IAM 정책문에서 제공하는 권한과 같이 `Project=Alpha` 태그를 기반으로 KMS 키에 액세스할 수 있다고 가정합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "IAMPolicyWithResourceTag",
      "Effect": "Allow",
      "Action": [
        "kms:GenerateDataKeyWithoutPlaintext",
        "kms:Decrypt"
      ],
      "Resource": "arn:aws:kms:ap-southeast-1:111122223333:key/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Project": "Alpha"
        }
      }
    }
  ]
}
```

해당 KMS 키에서 태그가 삭제되거나 태그 값이 변경되면 보안 주체는 지정된 작업에 KMS 키를 사용할 수 있는 권한을 더 이상 갖지 않습니다. 이는 보안 주체가 고객 관리형 키를 사용하는 AWS 서비스에

서 데이터를 읽거나 쓰려고 할 때 명백해질 수 있습니다. 태그 변경을 추적하려면 CloudTrail 로그에서 TagResource 또는 UntagResource 항목을 검토합니다.

정책을 업데이트하지 않고 액세스를 복원하려면 KMS 키의 태그를 변경합니다. 이 조치는 AWS KMS 전반에 걸쳐 효력을 발휘하는 짧은 기간 외에는 최소한의 영향을 미칩니다. 이와 같은 오류를 방지하려면 태그 지정 및 태그 해제 권한을 필요한 보안 주체에게만 부여하고 관리해야 하는 태그로 태그 지정 권한을 제한하십시오. 태그를 변경하기 전에 정책을 검색하여 태그에 따라 달라지는 액세스를 감지하고 태그가 있는 모든 리전에서 KMS 키를 가져옵니다. 특정 태그가 변경되면 Amazon CloudWatch 경보를 생성하는 것을 고려할 수 있습니다.

별칭 변경으로 인한 액세스 변경

별칭이 삭제되거나 다른 KMS 키와 연결된 경우 해당 별칭만 기반으로 KMS 키에 액세스할 수 있는 보안 주체는 KMS 키에 대한 액세스가 거부됩니다. 이는 KMS 키와 연결된 별칭이 거부 정책문에 포함된 경우에도 발생할 수 있습니다. KMS 키에 정책 관련 별칭을 추가해도 KMS 키에 대한 액세스가 거부되어야 하는 보안 주체에 액세스가 허용될 수 있습니다.

예를 들어 다음 IAM 정책 문은 kms:ResourceAliases 조건 키를 사용하여 지정된 별칭을 가진 계정의 다른 리전에 있는 KMS 키에 대한 액세스를 허용합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AliasBasedIAMPolicy",
      "Effect": "Allow",
      "Action": [
        "kms:List*",
        "kms:Describe*",
        "kms:Decrypt"
      ],
      "Resource": "arn:aws:kms:*:111122223333:key/*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "kms:ResourceAliases": [
            "alias/ProjectAlpha",
            "alias/ProjectAlpha_Test",
            "alias/ProjectAlpha_Dev"
          ]
```

```
        }
      }
    }
  ]
}
```

별칭 변경을 추적하려면 CloudTrail 로그에서 CreateAlias,UpdateAlias 및 DeleteAlias 항목을 검토하십시오.

정책을 업데이트하지 않고 액세스를 복원하려면 KMS 키와 연결된 별칭을 변경합니다. 각 별칭은 계정 및 리전에서 하나의 KMS 키에만 연결될 수 있으므로 별칭 관리는 태그를 관리하는 것보다 조금 더 어렵습니다. 하나의 KMS 키에서 일부 보안 주체에 액세스 권한을 복원하면 다른 KMS 키에 대한 동일 또는 다른 보안 주체의 액세스가 거부될 수 있습니다.

이 오류를 방지하려면 별칭 관리 권한을 필요한 보안 주체에게만 부여하고 관리해야 하는 별칭으로 별칭 관리 권한을 제한합니다. 별칭을 업데이트하거나 삭제하기 전에 정책을 검색하여 별칭에 종속된 액세스를 검색하고 별칭과 연결된 모든 리전에서 KMS 키를 찾습니다.

별칭 할당량으로 인한 액세스 거부

kms:ResourceAliases 조건에 따라 KMS 키를 사용하도록 권한이 부여된 사용자는 KMS 키가 해당 계정 및 리전에 대한 KMS 키당 별칭의 기본 할당량을 초과하는 경우 `AccessDenied` 예외를 수신합니다.

액세스를 복원하려면 할당량을 준수하도록 KMS 키와 연결된 별칭을 삭제합니다. 또는 대체 메커니즘을 사용하여 사용자에게 KMS 키에 대한 액세스 권한을 부여합니다.

지연된 권한 부여 변경 사항

태그 및 별칭을 변경하면 KMS 키 인증에 영향을 미치는 데 최대 5분이 소요될 수 있습니다. 따라서 권한 부여에 영향을 미치기 전에 API 작업의 응답에 태그 또는 별칭 변경이 반영될 수 있습니다. 이 지연은 대부분의 AWS KMS 작업에 영향을 미치는 짧은 최종 일관성 지연보다 길 수 있습니다.

예를 들어 특정 보안 주체가 `"Purpose"="Test"` 태그가 있는 모든 KMS 키를 사용하도록 허용하는 IAM 정책이 있을 수 있습니다. 그런 다음 KMS 키에 `"Purpose"="Test"` 태그를 추가합니다. TagResource 작업이 완료되고 ListResourceTags 응답에서 태그가 KMS 키에 할당되었음을 확인하지만 보안 주체는 최대 5분 동안 KMS 키에 액세스하지 못할 수 있습니다.

오류를 방지하려면 이 예상 지연을 코드에 빌드하십시오.

별칭 업데이트로 인해 실패한 요청

별칭을 업데이트할 때 기존 별칭을 다른 KMS 키와 연결합니다.

별칭 이름 또는 별칭 ARN을 지정하는 Decrypt 및 ReEncrypt 요청은 이제 암호문을 암호화하지 않은 KMS 키와 별칭이 연결되기 때문에 실패할 수 있습니다. 이 상황은 일반적으로 `IncorrectKeyException` 또는 `NotFoundException`을 반환합니다. 또는 요청에 `KeyId` 또는 `DestinationKeyId` 파라미터가 없는 경우 호출자가 암호문을 암호화한 KMS 키에 더 이상 액세스할 수 없기 때문에 `AccessDenied` 예외와 함께 작업이 실패할 수 있습니다.

CreateAlias, UpdateAlias 및 DeleteAlias로그 항목에 대한 CloudTrail 로그를 확인하여 변경 사항을 추적할 수 있습니다. ListAliases 응답의 `LastUpdatedDate` 필드 값을 사용하여 변경 사항을 감지할 수도 있습니다.

예를 들어 다음 ListAliases 예제 응답은 `kms:ResourceAliases` 조건의 `ProjectAlpha_Test` 별칭이 업데이트되었음을 보여줍니다. 따라서 별칭을 기반으로 액세스 권한이 있는 보안 주체는 이전에 연결된 KMS 키에 대한 액세스 권한을 잃게 됩니다. 대신 새로 연결된 KMS 키에 액세스할 수 있습니다.

```
$ aws kms list-aliases --query 'Aliases[?starts_with(AliasName, `alias/ProjectAlpha`)]'

{
    "Aliases": [
        {
            "AliasName": "alias/ProjectAlpha_Test",
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/ProjectAlpha_Test",
            "TargetKeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
            "CreationDate": 1566518783.394,
            "LastUpdatedDate": 1605308931.903
        },
        {
            "AliasName": "alias/ProjectAlpha_Restricted",
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/
ProjectAlpha_Restricted",
            "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
            "CreationDate": 1553410800.010,
            "LastUpdatedDate": 1553410800.010
        }
    ]
}
```

이 변경에 대한 해결책은 간단하지 않습니다. 별칭을 다시 업데이트하여 원래 KMS 키와 연결할 수 있습니다. 그러나 작업을 수행하기 전에 해당 변경 사항이 현재 연결된 KMS 키에 미치는 영향을 고려해

야 합니다. 보안 주체가 암호화 작업에 후자의 KMS 키를 사용한 경우 보안 주체에 계속 액세스해야 할 수 있습니다. 이 경우 보안 주체가 두 KMS 키를 모두 사용할 수 있는 권한을 갖도록 정책을 업데이트할 수 있습니다.

이와 같은 오류는 다음과 같이 방지할 수 있습니다. 별칭을 업데이트하기 전에 별칭에 의존하는 액세스를 감지하는 정책을 검색하십시오. 그런 다음 별칭과 연결된 모든 리전의 KMS 키를 가져옵니다. 별칭 관리 권한을 필요한 보안 주체에게만 부여하고 관리해야 하는 별칭으로 별칭 관리 권한을 제한합니다.

에 대한 RBAC AWS KMS

역할 기반 액세스 제어(RBAC)는 사용자에게 직무 수행에 필요한 권한만 제공하는 권한 부여 전략입니다. AWS KMS 는 키 정책 내에서 키 사용에 대한 세분화된 권한을 지정하여 키에 대한 액세스를 제어할 수 있도록 함으로써 RBAC를 지원합니다. 키 정책은 키에 대한 액세스 권한을 부여하는 리소스, 작업, 효과, 위탁자 및 선택적 조건을 지정합니다.

에서 RBAC를 구현하려면 키 사용자와 키 관리자에 대한 권한을 분리하는 AWS KMS것이 좋습니다.

Key users

다음 키 정책 문 예제는 `ExampleUserRole` IAM 역할이 KMS 키를 사용하도록 허용합니다.

```
{
            "Sid": "Allow use of the key",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws: iam::111122223333:role/ExampleUserRole"
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey"
            ],
            "Resource": "*"
  }
```

실제 키 사용자에게는 이 예제의 사용자보다 더 적은 권한이 필요할 수 있습니다. 그런 경우에는 사용자에게 필요한 권한만 할당하세요. 다음 질문을 사용하여 권한을 더욱 구체화할 수 있습니다.

- 어떤 IAM 위탁자(역할 또는 사용자)가 키에 액세스해야 하나요?

- 각 위탁자가 키로 수행해야 하는 작업은 무엇인가요? 예를 들어 위탁자에게 암호화 및 서명 권한만 필요한가요?
- 사용자가 인간입니까 아니면 AWS 서비스입니까? 서비스인 AWS 경우 조건 키를 사용하여 키 사용을 특정 AWS 서비스로 제한할 수 있습니다.

Key administrators

다음 키 정책 예제에서는 `ExampleAdminRole` IAM 역할이 KMS 키를 관리할 수 있도록 허용합니다.

```
{
            "Sid": "Allow access for Key Administrators",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws: iam::111122223333:role/ExampleAdminRole"
            },
            "Action": [
                "kms:Create*",
                "kms:Describe*",
                "kms:Enable*",
                "kms:List*",
                "kms:Put*",
                "kms:Update*",
                "kms:Revoke*",
                "kms:Disable*",
                "kms:Get*",
                "kms:Delete*",
                "kms:TagResource",
                "kms:UntagResource",
                "kms:ScheduleKeyDeletion",
                "kms:CancelKeyDeletion"
            ],
            "Resource": "*"
    }
```

실제 키 관리자에게는 이 예제의 관리자보다 적은 권한이 필요할 수 있습니다. 그런 경우에는 키 관리자에게 필요한 권한만 할당하세요.

사용자에게 역할을 수행하는 데 필요한 권한만 부여하세요. 사용자의 권한은 키가 테스트 환경에서 사용되는지 또는 프로덕션 환경에서 사용되는지에 따라 달라질 수 있습니다. 특정 비프로덕션 환경에서

덜 제한적인 권한을 사용하는 경우에는 정책을 프로덕션으로 릴리스하기 전에 테스트하는 프로세스를 구현하세요.

자세히 알아보기

- IAM 자격 증명(사용자, 사용자 그룹 및 역할)
- 액세스 제어의 유형

다른 계정의 사용자가 KMS를 사용하도록 허용

다른의 사용자 또는 역할이 계정에서 KMS 키를 AWS 계정 사용하도록 허용할 수 있습니다. 교차 계정 액세스에는 KMS 키의 키 정책과 외부 사용자 계정의 IAM 정책에 대한 권한이 필요합니다.

교차 계정 권한은 다음 작업에만 적용됩니다.

- 암호화 작업
- CreateGrant
- DescribeKey
- GetKeyRotationStatus
- GetPublicKey
- ListGrants
- RetireGrant
- RevokeGrant

다른 계정의 사용자에게 다른 작업에 대한 권한을 부여하면 해당 권한은 영향을 주지 않습니다. 예를 들어 다른 계정의 보안 주체에 IAM 정책의 kms:ListKeys 권한을 부여하거나 키 정책의 KMS 키에 대한 kms:ScheduleKeyDeletion 권한을 부여하는 경우 리소스에서 해당 작업을 호출하려는 사용자의 시도는 여전히 실패합니다.

AWS KMS 작업에 다른 계정에서 KMS 키를 사용하는 방법에 대한 자세한 내용은 AWS KMS 권한 및 의 교차 계정 사용 열을 참조하세요다른 계정에서 KMS 키 사용. 또한 AWS Key Management Service API 참조서의 각 API 설명에 교차 계정 사용 섹션도 있습니다.

> **Warning**
>
> 보안 주체에 KMS 키를 사용할 수 있는 권한을 부여할 때는 주의해야 합니다. 가능하면 최소 권한 원칙을 따르십시오. 사용자에게 필요한 작업에만 필요한 KMS 키에만 액세스할 수 있도록 합니다.
> 또한 익숙하지 않은 KMS 키, 특히 다른 계정의 KMS 키를 사용할 때는 주의해야 합니다. 악의적인 사용자는 KMS 키를 사용하여 본인이나 계정에 대한 정보를 얻을 수 있는 권한을 부여할 수 있습니다.
> 정책을 사용하여 계정의 리소스를 보호하는 방법에 대한 자세한 내용은 IAM 정책 모범 사례 섹션을 참조하십시오.

다른 계정의 사용자 및 역할에게 KMS 키를 사용할 수 있는 권한을 부여하려면 다음 두 가지 유형의 정책을 사용해야 합니다.

- KMS 키의 키 정책은 외부 계정(또는 외부 계정의 사용자 및 역할)에게 KMS 키를 사용할 수 있는 권한을 부여해야 합니다. 키 정책은 KMS 키를 소유하는 계정에 있습니다.
- 외부 계정의 IAM 정책은 키 정책 권한을 해당 사용자 및 역할에 위임해야 합니다. 이러한 정책은 외부 계정에 설정되며 해당 계정의 사용자 및 역할에 권한을 부여합니다.

키 정책은 누가 KMS 키에 대한 액세스 권한을 가질 수 있는지를 결정합니다. IAM 정책은 누가 KMS 키에 대한 액세스 권한을 가질 수 있는지를 결정합니다. 키 정책과 IAM 정책만으로는 충분하지 않습니다. 둘 다 변경해야 합니다.

키 정책을 편집하려면 AWS Management Console 에서 정책 보기를 사용하거나 CreateKey 또는 PutKeyPolicy 작업을 사용할 수 있습니다.

IAM 정책 편집에 대한 도움말은 에서 IAM 정책 사용 AWS KMS 섹션을 참조하십시오.

키 정책과 IAM 정책이 함께 작동하여 다른 계정에서 KMS 키 사용을 허용하는 방법을 보여 주는 예는 예제 2: 사용자가 다른에서 KMS 키를 사용할 권한이 있는 역할을 맡음 AWS 계정 섹션을 참조하십시오.

AWS CloudTrail 로그의 KMS 키에서 결과 교차 계정 AWS KMS 작업을 볼 수 있습니다. 다른 계정의 KMS 키를 사용하는 작업은 호출자의 계정과 KMS 키 소유자 계정 모두에 기록됩니다.

주제

- 1단계: 로컬 계정에서 키 정책 문 추가

- 2단계: 외부 계정에서 IAM 정책 추가
- 에서 외부 KMS 키 사용 허용 AWS 서비스
- 다른 계정에서 KMS 키 사용

> **Note**
>
> 이 주제의 예제에서는 키 정책과 IAM 정책을 함께 사용하여 KMS 키에 대한 액세스를 제공하고 제한하는 방법을 보여 줍니다. 이러한 일반 예제는 KMS 키에 대해 특정에 AWS 서비스 필요한 권한을 나타내기 위한 것이 아닙니다. 에 AWS 서비스 필요한 권한에 대한 자세한 내용은 서비스 설명서의 암호화 주제를 참조하세요.

1단계: 로컬 계정에서 키 정책 문 추가

KMS 키에 대한 키 정책은 누가 KMS 키에 액세스할 수 있으며 어떤 작업을 수행할 수 있는지에 대한 주요 결정 요인입니다. 키 정책은 항상 KMS 키를 소유하는 계정에 있습니다. IAM 정책과 달리, 키 정책은 리소스를 지정하지 않습니다. 리소스는 키 정책과 연결된 KMS 키입니다. 교차 계정 권한을 제공할 때 KMS 키의 키 정책은 외부 계정(또는 외부 계정의 사용자 및 역할)에게 KMS 키를 사용할 수 있는 권한을 부여해야 합니다.

외부 계정에게 KMS 키를 사용할 수 있는 권한을 부여하려면 외부 계정을 지정하는 문을 키 정책에 추가합니다. 키 정책의 `Principal` 요소에 외부 계정의 Amazon 리소스 이름(ARN)을 입력합니다.

키 정책에서 외부 계정을 지정하면 외부 계정의 IAM 관리자는 IAM 정책을 사용하여 해당 권한을 외부 계정의 사용자 및 역할에게 위임할 수 있습니다. 또한 키 정책에 지정된 작업 중 어떤 작업을 사용자 및 역할이 수행할 수 있는지도 결정할 수 있습니다.

외부 계정 및 보안 주체에 부여된 사용 권한은 KMS 키와 키 정책을 호스팅하는 리전에서 외부 계정을 사용하도록 설정한 경우에만 유효합니다. 기본적으로 사용하도록 설정되지 않은 리전('옵트인 리전')에 대한 자세한 내용은 AWS 일반 참조의 AWS 리전관리를 참조하세요.

예를 들어 계정 444455556666이 계정 111122223333에서 대칭 암호화 KMS 키를 사용하도록 허용한다고 가정합니다. 이렇게 하려면 다음 예제와 같은 정책 문을 111122223333 계정의 KMS 키에 대한 키 정책에 추가합니다. 이 정책문은 대칭 암호화 KMS 키에 대한 암호화 작업에서 KMS 키를 사용할 수 있는 권한을 외부 계정 444455556666에 부여합니다.

> **Note**
>
> 다음 예는 KMS 키를 다른 계정과 공유하기 위한 샘플 키 정책을 나타냅니다. 예제 `Sid`, `Principal` 및 `Action` 값을 KMS 키의 용도에 맞는 유효한 값으로 바꾸세요.

```
{
    "Sid": "Allow an external account to use this KMS key",
    "Effect": "Allow",
    "Principal": {
        "AWS": [
            "arn:aws:iam::444455556666:root"
        ]
    },
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
    ],
    "Resource": "*"
}
```

외부 계정에게 권한을 부여하는 대신, 키 정책에서 특정 외부 사용자 및 역할을 지정할 수 있습니다. 하지만 이러한 사용자 및 역할은 외부 계정의 IAM 관리자가 적절한 IAM 정책을 해당 자격 증명에 연결할 때까지 KMS 키를 사용할 수 없습니다. IAM 정책은 키 정책에 지정된 외부 사용자 및 역할의 일부 또는 전부에게 권한을 부여할 수 있습니다. 그리고 이 정책은 키 정책에 지정된 작업의 일부 또는 전부를 허용할 수 있습니다.

키 정책에서 자격 증명을 지정하면 외부 계정의 IAM 관리자가 제공할 수 있는 권한이 제한됩니다. 하지만 이렇게 하면 두 계정의 정책 관리가 더 복잡해집니다. 예를 들어, 사용자 또는 역할을 추가해야 한다고 가정합니다. KMS 키를 소유하는 계정에서 키 정책에 해당 자격 증명을 추가하고 자격 증명의 계정에서 IAM 정책을 생성해야 합니다.

키 정책에서 특정 외부 사용자 또는 역할을 지정하려면 `Principal` 요소에 외부 계정의 사용자 또는 역할의 Amazon 리소스 이름(ARN)을 입력합니다.

예를 들어, 다음 예제 키 정책 문은 444455556666 계정의 ExampleRole이 111122223333 계정의 KMS 키를 사용하도록 허용합니다. 이 키 정책문은 대칭 암호화 KMS 키에 대한 암호화 작업에서 KMS 키를 사용할 수 있는 권한을 외부 계정 444455556666에 부여합니다.

> **Note**
>
> 다음 예는 KMS 키를 다른 계정과 공유하기 위한 샘플 키 정책을 나타냅니다. 예제 Sid, Principal 및 Action 값을 KMS 키의 용도에 맞는 유효한 값으로 바꾸세요.

```
{
    "Sid": "Allow an external account to use this KMS key",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::444455556666:role/ExampleRole"
    },
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
    ],
    "Resource": "*"
}
```

> **Note**
>
> 조건을 사용하여 키 정책을 제한하지 않는 한 권한을 허용하는 키 정책문에서 보안 주체를 별표(*)로 설정하지 마세요. 별표는 다른 정책 문이 명시적으로 거부하지 않는 한 모든 자격 증명에 KMS 키를 사용할 수 있는 AWS 계정 권한을 부여합니다. 다른의 사용자는 자신의 계정에 해당 권한이 있을 때마다 KMS 키를 사용할 AWS 계정 수 있습니다.

또한 외부 계정에게 어떤 권한을 부여할지도 결정해야 합니다. 예를 들어, 암호화 해제할 수 있지만 암호화할 수 없는 권한을 부여하거나, KMS 키를 볼 수 있지만 사용할 수 없는 권한을 부여해야 할 수 있습니다. KMS 키에 대한 권한 목록은 AWS KMS 권한 섹션을 참조하세요.

KMS 키를 생성하는 경우 키 정책 설정

CreateKey 작업을 사용하여 KMS 키를 생성하는 경우 `Policy` 파라미터를 사용하여 외부 계정 또는 외부 사용자 및 역할에게 KMS 키를 사용할 권한을 부여하는 키 정책을 지정할 수 있습니다.

에서 KMS 키를 생성할 때 해당 키 정책 AWS Management Console도 생성합니다. Key Administrators(키 관리자) 및 Key Users(키 사용자) 섹션에서 자격 증명을 선택하면 AWS KMS 는 해당 자격 증명에 대한 정책 문을 KMS 키의 키 정책에 추가합니다. 또한 Key Users(키 사용자) 섹션을 사용하여 외부 계정을 키 사용자로 추가할 수도 있습니다.

외부 계정의 계정 ID를 입력하면가 키 정책에 두 개의 문을 AWS KMS 추가합니다. 이 작업은 키 정책에만 영향을 미칩니다. 해당 사용자 및 역할에 이러한 권한의 일부 또는 전부를 부여하는 IAM 정책을 연결할 때까지 외부 계정의 사용자 및 역할은 KMS 키를 사용할 수 없습니다.

첫 번째 키 정책문은 외부 계정에게 암호화 작업에서 KMS 키를 사용할 수 있는 권한을 부여합니다. 두 번째 키 정책 문은 외부 계정이 KMS 키에 대한 권한 부여를 생성, 확인 및 취소할 수 있도록 허용합니다. 단, 요청이 AWS 와 통합된 서비스 AWS KMS에서 오는 경우에만 가능합니다. 이러한 권한을 통해 사용자 데이터를 암호화하는 다른 AWS 서비스에서 KMS 키를 사용할 수 있습니다. 이러한 권한은 AWS 서비스의 사용자 데이터를 암호화하는 KMS 키용으로 설계되었습니다.

2단계: 외부 계정에서 IAM 정책 추가

KMS 키를 소유하는 계정의 키 정책은 권한의 유효 범위를 설정합니다. 그러나 해당 권한을 위임하는 IAM 정책을 연결하거나 권한 부여를 사용하여 KMS 키에 대한 액세스를 관리할 때까지 외부 계정의 사용자 및 역할은 KMS 키를 사용할 수 없습니다. IAM 정책은 외부 계정에서 설정됩니다.

키 정책이 외부 계정에게 권한을 부여하는 경우 계정의 어떠한 사용자 또는 역할에든 IAM 정책을 연결할 수 있습니다. 키 정책이 지정된 사용자 또는 역할에게 권한을 부여하는 경우 IAM 정책은 지정된 사용자 및 역할의 일부 또는 전부에게만 해당 권한을 부여할 수 있습니다. IAM 정책이 기타 외부 사용자 또는 역할에게 KMS 키 액세스 권한을 부여하는 경우 해당 권한은 효과가 없습니다.

키 정책은 IAM 정책의 작업도 제한합니다. IAM 정책은 키 정책에 지정된 작업의 일부 또는 전부를 위임할 수 있습니다. IAM 정책이 키 정책에 지정되지 않은 작업을 나열하는 경우 해당 권한은 효과가 없습니다.

다음 예제 IAM 정책은 보안 주체가 111122223333 계정의 KMS 키를 암호화 작업에 사용하도록 허용합니다. 이 권한을 444455556666 계정의 사용자 및 역할에 부여하려면 444455556666 계정의 사용자 또는 역할에 정책을 연결합니다.

> **Note**
>
> 다음 예제는 KMS 키를 다른 계정과 공유하기 위한 샘플 IAM 정책을 나타냅니다. 예제 `Sid`, `Resource` 및 `Action` 값을 KMS 키의 용도에 맞는 유효한 값으로 바꾸세요.

```
{
    "Sid": "AllowUseOfKeyInAccount111122223333",
    "Effect": "Allow",
    "Action": [
      "kms:Encrypt",
      "kms:Decrypt",
      "kms:ReEncrypt*",
      "kms:GenerateDataKey*",
      "kms:DescribeKey"
    ],
    "Resource": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
}
```

이 정책에 대한 다음 세부 정보를 참고하세요.

- 키 정책과 달리, IAM 정책문에는 `Principal` 요소가 포함되지 않습니다. IAM 정책에서 보안 주체는 정책이 연결되는 자격 증명입니다.
- IAM 정책의 `Resource` 요소는 보안 주체가 사용할 수 있는 KMS 키를 식별합니다. KMS 키를 지정하려면 해당 키 ARN을 `Resource` 요소에 추가합니다.
- `Resource` 요소에서 KMS 키를 두 개 이상 지정할 수 있습니다. 하지만 `Resource` 요소에서 특정 KMS 키를 지정하지 않으면 의도한 것보다 많은 KMS 키에 대한 액세스 권한을 실수로 부여할 수 있습니다.
- 외부 사용자가 AWS KMS와 통합된AWS 서비스에 KMS 키를 사용하도록 허용하려면 키 정책 또는 IAM 정책에 권한을 추가해야 할 수 있습니다. 자세한 내용은 에서 외부 KMS 키 사용 허용 AWS 서비스 섹션을 참조하세요.

IAM 정책 작업에 대한 자세한 내용은 IAM 정책 섹션을 참조하십시오.

에서 외부 KMS 키 사용 허용 AWS 서비스

다른 계정의 사용자에게 AWS KMS와 통합된 서비스에 KMS 키를 사용할 권한을 부여할 수 있습니다. 예를 들어, 외부 계정의 사용자는 KMS 키를 사용하여 Amazon S3 버킷의 객체를 암호화하거나 객체가 AWS Secrets Manager에 저장하는 비밀을 암호화할 수 있습니다.

키 정책은 외부 사용자 또는 외부 사용자의 계정에게 KMS 키를 사용할 수 있는 권한을 부여해야 합니다. 추가로, 사용자에게 AWS 서비스를 사용할 수 있는 권한을 부여하는 IAM 정책을 자격 증명에 연결해야 합니다. 서비스에서 사용자에게 키 정책 또는 IAM 정책에 대한 추가 권한이 필요할 수도 있습니다. 에서 고객 관리형 키에 AWS 서비스 필요한 권한 목록은 서비스 사용 설명서 또는 개발자 안내서의 보안 장에서 데이터 보호 주제를 참조하세요.

다른 계정에서 KMS 키 사용

다른에서 KMS 키를 사용할 권한이 있는 경우 AWS Management Console, AWS 계정, AWS SDKs AWS CLI및에서 KMS 키를 사용할 수 있습니다 AWS Tools for PowerShell.

셸 명령 또는 API 요청에서 다른 계정의 KMS 키를 식별하려면 다음 키 식별자를 사용합니다.

- 암호화 작업, DescribeKey 및 GetPublicKey의 경우 KMS 키의 키 ARN 또는 별칭 ARN을 사용합니다.
- CreateGrant, GetKeyRotationStatus, ListGrants 및 RevokeGrant의 경우 KMS 키의 키 ARN을 사용합니다.

키 ID 또는 별칭 이름만 입력하면 KMS 키가 계정에 있다고 AWS 가정합니다.

사용 권한이 있더라도 AWS KMS 콘솔은 다른 계정에 KMS 키를 표시하지 않습니다. 또한 다른 AWS 서비스의 콘솔에 표시되는 KMS 키 목록에는 다른 계정의 KMS 키가 포함되어 있지 않습니다.

AWS 서비스 콘솔의 다른 계정에서 KMS 키를 지정하려면 KMS 키의 키 ARN 또는 별칭 ARN을 입력해야 합니다. 필요한 키 식별자는 서비스에 따라 다르며, 서비스 콘솔과 API 작업마다 다를 수 있습니다. 세부 정보는 서비스 문서를 참조하세요.

다중 리전 키에 대한 액세스 제어

다중 리전 키를 사용하면 규정 준수, 재해 복구 및 백업 시나리오에서 단일 리전 키를 사용할 수 있습니다. 그러나 다중 리전 키의 보안 속성은 단일 리전 키의 보안 속성과 크게 다르기 때문에 다중 리전 키의 생성, 관리 및 사용을 승인할 때는 주의를 기울이는 것이 좋습니다.

> **Note**
>
> `Resource` 필드에 와일드카드 문자가 있는 기존 IAM 정책문은 단일 리전 키와 다중 리전 키 모두에 적용됩니다. 단일 리전 KMS 키 또는 다중 리전 키로 제한하려면 kms:MultiRegion 조건 키를 사용합니다.

단일 리전으로 충분할 경우 권한 부여 도구를 사용하여 다중 리전 키를 생성 및 사용할 수 없게 할 수 있습니다. 보안 주체가 필요한 에만 다중 리전 키를 복제하도록 허용합니다 AWS 리전 . 다중 리전 키에 대한 권한을 필요로 하는 보안 주체와 해당 키가 필요한 작업에 대해서만 부여합니다.

키 정책, IAM 정책 및 권한 부여를 사용하여 IAM 보안 주체가에서 다중 리전 키를 관리하고 사용하도록 허용할 수 있습니다 AWS 계정. 각 다중 리전 키는 고유한 키 ARN 및 키 정책을 가진 독립적인 리소스입니다. 각 키에 대한 키 정책을 설정 및 유지 관리하고 신규 및 기존 IAM 정책이 권한 부여 전략을 구현하는지 확인해야 합니다.

다중 리전 키를 지원하려면 IAM 서비스 연결 역할을 AWS KMS 사용합니다. 이 역할은 AWS KMS 에게 공유 속성을 동기화하는 데 필요한 권한을 부여합니다. 자세한 내용은 다중 리전 키 AWS KMS 동기화 권한 부여 단원을 참조하십시오.

주제

- 다중 리전 키에 대한 인증 기본 사항
- 다중 리전 키 관리자 및 사용자 권한 부여

다중 리전 키에 대한 인증 기본 사항

다중 리전 키에 대한 주요 정책 및 IAM 정책을 설계할 때는 다음 원칙을 고려하십시오.

- 키 정책 — 각 다중 리전 키는 독립적인 KMS 키 리소스이며 자체 키 정책이 있습니다. 관련된 다중 리전 키 집합의 각 키에 동일하거나 다른 키 정책을 적용할 수 있습니다. 키 정책은 다중 리전 키의 공유 속성이 아닙니다.는 관련 다중 리전 키 간에 키 정책을 복사하거나 동기화하지 AWS KMS 않습니다.

  AWS KMS 콘솔에서 복제본 키를 생성하면 콘솔에 편의를 위해 기본 키의 현재 키 정책이 표시됩니다. 이 키 정책을 사용하거나 편집하거나 삭제하고 바꿀 수 있습니다. 그러나 기본 키 정책을 변경하지 않고 수락하더라도 AWS KMS 는 정책을 동기화하지 않습니다. 예를 들어, 기본 키의 키 정책을 변경하는 경우 복제 키의 키 정책은 동일하게 유지됩니다.

- 기본 키 정책 — CreateKey 및 ReplicateKey 작업을 사용하여 다중 리전 키를 생성하는 경우 요청에 키 정책을 지정하지 않는 한 기본 키 정책이 적용됩니다. 이는 단일 리전 키에 적용되는 기본 키 정책과 동일합니다.
- IAM 정책 — 모든 KMS 키와 마찬가지로 IAM 정책을 사용하여 키 정책에서 허용하는 경우에만 다중 리전 키에 대한 액세스를 제어할 수 있습니다. IAM 정책은 AWS 리전 기본적으로 모든에 적용됩니다. 그러나 aws:RequestedRegion과 같은 조건 키를 사용하여 특정 리전으로 권한을 제한할 수 있습니다.

  기본 및 복제본 키를 만들려면 보안 주체는 키가 생성된 리전에 적용되는 IAM 정책에 kms:CreateKey 권한이 있어야 합니다.
- 권한 부여 - AWS KMS 권한 부여는 리전별입니다. 각 권한 부여는 하나의 KMS 키에 대한 권한을 허용합니다. 권한 부여를 사용하여 다중 리전 기본 키 또는 복제본 키에 대한 사용 권한을 허용할 수 있습니다. 그러나 다중 리전 키와 관련된 경우에도 단일 권한 부여를 사용하여 여러 KMS 키에 대한 사용 권한을 허용할 수는 없습니다.
- 키 ARN — 각 다중 리전 키에는 고유 키 ARN이 있습니다. 관련된 다중 리전 키의 키 ARN은 파티션, 계정 및 키 ID는 같지만 리전은 다릅니다.

  특정 다중 리전 키에 IAM 정책문을 적용하려면 해당 키 ARN 또는 해당 리전을 포함하는 키 ARN 패턴을 사용합니다. IAM 정책문을 모든 관련 다중 리전 키에 적용하려면 다음 예시에서와 같이 ARN의 리전 요소에 와일드카드 문자(*)를 사용합니다.

```
{
  "Effect": "Allow",
  "Action": [
    "kms:Describe*",
    "kms:List*"
  ],
  "Resource": {
      "arn:aws:kms:*::111122223333:key/mrk-1234abcd12ab34cd56ef1234567890ab"
  }
}
```

  의 모든 다중 리전 키에 정책 설명을 적용하려면 kms:MultiRegion 정책 조건 또는 고유한 mrk- 접두사를 포함하는 키 ID 패턴을 사용할 AWS 계정수 있습니다.
- 서비스 연결 역할 — 다중 리전 기본 키를 만드는 보안 주체는 iam:CreateServiceLinkedRole 권한이 있어야 합니다.

관련 다중 리전 키의 공유 속성을 동기화하기 위해 AWS KMS 는 IAM 서비스 연결 역할을 수임합니다.는 다중 리전 기본 키를 AWS KMS 생성할 때마다 AWS 계정 에서 서비스 연결 역할을 생성합니다. (역할이 존재하는 경우 AWS KMS 는 유해한 영향을 미치지 않습니다.) 역할은 모든 리전에서 유효합니다. AWS KMS 가 서비스 연결 역할을 생성(또는 다시 생성)하도록 허용하려면 다중 리전 기본 키를 생성하는 보안 주체에 iam:CreateServiceLinkedRole 권한이 있어야 합니다.

다중 리전 키 관리자 및 사용자 권한 부여

다중 리전 키를 만들고 관리하는 보안 주체는 기본 및 복제본 리전에서 다음 사용 권한이 필요합니다.

- `kms:CreateKey`
- `kms:ReplicateKey`
- `kms:UpdatePrimaryRegion`
- `iam:CreateServiceLinkedRole`

기본 키 생성

다중 리전 기본 키를 생성하려면 보안 주체에게 기본 키의 리전에서 유효한 IAM 정책에 kms:CreateKey 및 iam:CreateServiceLinkedRole 권한이 필요합니다. 이러한 사용 권한이 있는 보안 주체는 사용자가 권한을 제한하지 않는 한 단일 리전 및 다중 리전 키를 만들 수 있습니다.

`iam:CreateServiceLinkedRole` 권한을 통해는 AWSServiceRoleForKeyManagementServiceMultiRegionKeys 역할을 AWS KMS 생성하여 관련 다중 리전 키의 공유 속성을 동기화할 수 있습니다.

예를 들어 이 IAM 정책은 위탁자가 다중 리전 키를 생성하고, 해당 키에 대한 정책을 연결하며, 다중 리전 키용 서비스 연결 역할을 생성하도록 허용합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement":{
      "Action": [
        "kms:CreateKey",
        "iam:CreateServiceLinkedRole"
```

```
        ],
        "Effect":"Allow",
        "Resource":"*"
    }
}
```

다중 리전 기본 키를 만들 수 있는 권한을 허용하거나 거부하려면 kms:MultiRegion 조건 키를 사용합니다. 유효 값은 `true`(다중 리전 키) 또는 `false`(단일 리전 키)입니다. 예를 들어 다음 IAM 정책문은 보안 주체가 다중 리전 키를 생성하지 못하도록 방지하기 위해 `kms:MultiRegion` 조건 키와 함께 `Deny` 작업을 사용합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement":{
      "Action":"kms:CreateKey",
      "Effect":"Deny",
      "Resource":"*",
      "Condition": {
          "Bool": {
            "kms:MultiRegion": true
          }
      }
  }
}
```

키 복제

다중 리전 복제본 키를 생성하려면 보안 주체는 다음 권한을 필요로 합니다.

- kms:ReplicateKey 권한(기본 키의 키 정책에서)
- kms:CreateKey 권한(복제본 키 리전에서 유효한 IAM 정책에서).

이러한 권한을 허용할 때는 주의해야 합니다. 보안 주체는 KMS 키와 사용 권한을 부여하는 키 정책을 만들 수 있습니다. `kms:ReplicateKey` 권한은 또한 AWS KMS내의 리전 경계를 넘어 키 구성 요소의 이전을 허용합니다.

다중 리전 키를 복제할 수 AWS 리전 있는를 제한하려면 kms:ReplicaRegion 조건 키를 사용합니다. 그것은 단지 kms:ReplicateKey 권한을 제한합니다. 그렇지 않으면 아무런 효과도 없습니다. 예를 들어 다음 키 정책은 보안 주체가 이 기본 키를 지정된 리전에서만 복제할 수 있도록 허용합니다.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/Administrator"
  },
  "Action": "kms:ReplicateKey",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:ReplicaRegion": [
         "us-east-1",
         "eu-west-3",
         "ap-southeast-2"
      ]
    }
  }
}
```

기본 리전 업데이트

인증된 보안 주체는 복제본 키를 기본 키로 변환하여 이전 기본 키를 복제본으로 변경할 수 있습니다. 이 작업을 기본 리전 업데이트라고 합니다. 기본 리전을 업데이트하려면 보안 주체에게 두 리전 모두에서 kms:UpdatePrimaryRegion 권한이 필요합니다. 키 정책 또는 IAM 정책에서 이러한 권한을 제공할 수 있습니다.

- kms:UpdatePrimaryRegion(기본 키에서). 이 권한은 기본 키 리전에서 유효해야 합니다.
- kms:UpdatePrimaryRegion(복제 키에서). 이 권한은 복제 키 리전에서 유효해야 합니다.

예를 들어 다음 키 정책은 관리자 역할을 수임할 수 있는 사용자에게 KMS 키의 기본 리전을 업데이트할 수 있는 권한을 부여합니다. 이 KMS 키는 이 작업에서 기본 키 또는 복제 키일 수 있습니다.

```
{
  "Effect": "Allow",
  "Resource": "*",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/Administrator"
  },
```

```
  "Action": "kms:UpdatePrimaryRegion"
}
```

기본 키를 호스팅할 수 AWS 리전 있는를 제한하려면 kms:PrimaryRegion 조건 키를 사용합니다. 예를 들어, 다음 IAM 정책 문은 새 기본 리전이 지정된 리전 중 하나인 AWS 계정경우에만 보안 주체가에서 다중 리전 키의 기본 리전을 업데이트하도록 허용합니다.

```
{
  "Effect": "Allow",
  "Action": "kms:UpdatePrimaryRegion",
  "Resource": {
      "arn:aws:kms:*:111122223333:key/*"
  },
  "Condition": {
    "StringEquals": {
      "kms:PrimaryRegion": [
         "us-west-2",
         "sa-east-1",
         "ap-southeast-1"
      ]
    }
  }
}
```

다중 리전 키 사용 및 관리

기본적으로 AWS 계정 및 리전에서 KMS 키를 사용하고 관리할 권한이 있는 보안 주체는 다중 리전 키를 사용하고 관리할 수 있는 권한도 있습니다. 그러나 kms:MultiRegion 조건 키를 사용하여 단일 리전 키만 허용하거나 다중 리전 키만 허용할 수 있습니다. 또는 kms:MultiRegionKeyType 조건 키를 사용하여 다중 리전 기본 키만 허용하거나 복제본 키만 허용합니다. 두 조건 키는 CreateKey 작업 및 기존 KMS 키를 사용하는 모든 작업(예: Encrypt 또는 EnableKey)에 대한 액세스를 제어합니다.

다음 예제 IAM 정책문은 `kms:MultiRegion` 조건 키를 사용하여 보안 주체가 다중 리전 키를 사용하거나 관리하지 못하도록 합니다.

```
{
  "Effect": "Deny",
  "Action": "kms:*",
  "Resource": "*",
  "Condition": {
    "Bool": "kms:MultiRegion": true
  }
```

```
}
```

이 예제 IAM 정책문은 kms:MultiRegionKeyType 조건을 사용하여 보안 주체가 다중 리전 복제본 키에 대해서만 키 삭제를 예약하고 취소할 수 있도록 허용합니다.

```
{
  "Effect": "Allow",
  "Action": [
    "kms:ScheduleKeyDeletion",
    "kms:CancelKeyDeletion"
  ],
  "Resource": {
      "arn:aws:kms:us-west-2:111122223333:key/*"
  },
  "Condition": {
    "StringEquals": "kms:MultiRegionKeyType": "REPLICA"
  }
}
```

에 대한 액세스 결정 AWS KMS keys

현재에 액세스할 수 있는 사람 또는 대상의 전체 범위를 확인하려면 KMS 키의 키 정책 AWS KMS key, KMS 키에 적용되는 모든 권한 부여 및 잠재적으로 모든 AWS Identity and Access Management (IAM) 정책을 검사해야 합니다. KMS 키의 가능한 사용 범위를 파악하거나 규정 또는 감사 요구 사항을 준수하기 위해 이 작업을 수행할 수 있습니다. 다음 주제는 현재 KMS 키에 액세스할 수 있는 AWS 보안 주체(ID)의 전체 목록을 생성하는 데 도움이 될 수 있습니다.

주제

- 키 정책 검사
- IAM 정책 검사
- 권한 부여 검사

키 정책 검사

키 정책은 KMS 키에 대한 액세스를 제어하는 기본적인 방법입니다. 모든 KMS 키에는 정확히 하나의 키 정책이 있습니다.

키 정책이 기본 키 정책으로 구성되거나 포함하는 경우 키 정책은 계정의 IAM 관리자가 IAM 정책을 사용하여 KMS 키에 대한 액세스를 제어하도록 허용합니다. 또한 키 정책이 다른 AWS 계정에 KMS 키

를 사용할 수 있는 권한을 부여하는 경우 외부 계정의 IAM 관리자는 IAM 정책을 사용하여 해당 권한을 위임할 수 있습니다. KMS 키에 액세스할 수 있는 보안 주체의 전체 목록을 결정하려면 IAM 정책을 검사하십시오.

계정AWS 관리형 키에서 AWS KMS 고객 관리형 키 또는의 키 정책을 보려면 AWS KMS API에서 AWS Management Console 또는 GetKeyPolicy 작업을 사용합니다. 키 정책을 보려면 KMS 키에 대한 `kms:GetKeyPolicy` 권한이 있어야 합니다. KMS 키에 대한 키 정책을 보는 방법은 the section called "키 정책 보기" 단원을 참조하십시오.

키 정책 문서를 검사하고 각 정책 문서의 `Principal` 요소에 지정된 모든 보안 주체를 기록해 둡니다. `Allow` 효과가 있는 정책 설명에서 IAM 사용자, IAM 역할 및 `Principal` 요소의는이 KMS 키 AWS 계정 에 액세스할 수 있습니다.

> **Note**
>
> 조건을 사용하여 키 정책을 제한하지 않는 한 권한을 허용하는 키 정책문에서 보안 주체를 별표(*)로 설정하지 마세요. 별표는 다른 정책 문이 명시적으로 거부하지 않는 한 모든 자격 증명에 KMS 키를 사용할 수 있는 AWS 계정 권한을 부여합니다. 다른의 사용자는 자신의 계정에 해당 권한이 있을 때마다 KMS 키를 사용할 AWS 계정 수 있습니다.

다음 예제는 기본 키 정책에 수록된 정책문을 사용하여 이를 수행하는 방법을 보여줍니다.

Example정책문 1

```
{
  "Sid": "Enable IAM User Permissions",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:root"},
  "Action": "kms:*",
  "Resource": "*"
}
```

정책 설명 1에서 `arn:aws:iam::111122223333:root`는 AWS 계정 111122223333을 참조하는 AWS 계정 보안 주체입니다. (계정 루트 사용자가 아닙니다.) 기본적으로를 사용하여 새 KMS 키를 생성 AWS Management Console하거나 프로그래밍 방식으로 새 KMS 키를 생성하지만 키 정책을 제공하지 않는 경우 이와 같은 정책 문이 키 정책 문서에 포함됩니다.

에 대한 액세스를 허용하는 문이 있는 키 정책 문서를 AWS 계정 사용하면 계정의 IAM 정책이 KMS 키에 대한 액세스를 허용할 수 있습니다. 즉 이 계정의 사용자와 역할은 키 정책 문서에 보안 주체로 명시

되지 않은 경우에도 KMS 키에 액세스할 수 있습니다. 보안 주체로 AWS 계정 나열된 모든의 모든 IAM 정책을 검사하여이 KMS 키에 대한 액세스를 허용하는지 확인합니다.

Example정책문 2

```
{
  "Sid": "Allow access for Key Administrators",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:role/KMSKeyAdmins"},
  "Action": [
    "kms:Describe*",
    "kms:Put*",
    "kms:Create*",
    "kms:Update*",
    "kms:Enable*",
    "kms:Revoke*",
    "kms:List*",
    "kms:Disable*",
    "kms:Get*",
    "kms:Delete*",
    "kms:ScheduleKeyDeletion",
    "kms:CancelKeyDeletion"
  ],
  "Resource": "*"
}
```

정책 설명 2에서는 AWS 계정 111122223333에서 KMSKeyAdmins라는 IAM 역할을 `arn:aws:iam::111122223333:role/KMSKeyAdmins` 참조합니다. 이 역할을 수임할 권한이 있는 사용자는 KMS 키 관리를 위한 관리 작업인 정책 문에 나열된 작업을 수행할 수 있습니다.

Example정책문 3

```
{
  "Sid": "Allow use of the key",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:role/EncryptionApp"},
  "Action": [
    "kms:DescribeKey",
    "kms:GenerateDataKey*",
    "kms:Encrypt",
    "kms:ReEncrypt*",
    "kms:Decrypt"
  ],
```

```
  "Resource": "*"
}
```

정책 설명 3에서는 EncryptionApp in AWS 계정 111122223333이라는 IAM 역할을 `arn:aws:iam::111122223333:role/EncryptionApp` 참조합니다. 이 역할을 맡을 권한이 있는 보안 주체는 대칭 암호화 KMS 키에 대한 암호화 작업을 포함하여 정책 문에 나열된 작업을 수행할 수 있습니다.

Example정책문 4

```
{
  "Sid": "Allow attachment of persistent resources",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:role/EncryptionApp"},
  "Action": [
    "kms:ListGrants",
    "kms:CreateGrant",
    "kms:RevokeGrant"
  ],
  "Resource": "*",
  "Condition": {"Bool": {"kms:GrantIsForAWSResource": true}}
}
```

정책 설명 4에서는 EncryptionApp in AWS 계정 111122223333이라는 IAM 역할을 `arn:aws:iam::111122223333:role/EncryptionApp` 참조합니다. 이 역할을 맡을 수 있는 권한을 지닌 보안 주체는 정책 문에 수록된 작업을 수행할 수 있습니다. 예제 정책문 3에서 허용된 작업과 결합된 이러한 작업은 AWS KMS와 통합되는 대부분의AWS 서비스, 특히 권한 부여를 사용하는 서비스에 KMS 키 사용을 위임하는 데 필요한 작업입니다. `Condition` 요소의 kms:GrantIsForAWSResource 값은 위임이 AWS KMS 와 통합되고 권한 부여를 사용하는 AWS 서비스인 경우에만 위임이 허용되도록 합니다.

키 정책 문서에서 보안 주체를 지정할 수 있는 모든 다양한 방법을 알아보려면 IAM 사용 설명서의 보안 주체 지정을 참조하십시오.

AWS KMS 키 정책에 대한 자세한 내용은 섹션을 참조하세요의 키 정책 AWS KMS.

IAM 정책 검사

키 정책 및 권한 부여 외에도 IAM 정책을 사용하여 KMS 키에 대한 액세스를 허용할 수도 있습니다. IAM 정책과 키 정책이 상호 작용하는 방식에 대한 자세한 내용은 AWS KMS 권한 문제 해결 단원을 참조하십시오.

현재 어떤 보안 주체가 IAM 정책을 통해 KMS 키에 액세스할 수 있는지 알아보기 위해, 브라우저 기반의 IAM 정책 시뮬레이터 도구를 사용하거나 IAM API에 요청을 제출할 수 있습니다.

IAM 정책을 검사하는 방법

- IAM 정책 시뮬레이터로 IAM 정책 검사
- IAM API로 IAM 정책 검사

IAM 정책 시뮬레이터로 IAM 정책 검사

IAM 정책 시뮬레이터에서 어떤 보안 주체가 IAM 정책을 통해 KMS 키에 액세스할 수 있는지 알아볼 수 있습니다.

IAM 정책 시뮬레이터를 사용해 KMS 키에 대한 액세스를 파악하려면

1. 에 로그인 AWS Management Console 한 다음에서 IAM 정책 시뮬레이터를 엽니다https://policysim.aws.amazon.com/.
2. 사용자, 그룹 및 역할(Users, Groups, and Roles) 창에서 정책을 시뮬레이션할 사용자, 그룹 또는 역할을 선택합니다.
3. (선택 사항) 시뮬레이션을 생략할 정책 옆 확인란의 선택을 취소합니다. 모든 정책을 시뮬레이션하려면 모든 정책을 선택해 둡니다.
4. [Policy Simulator] 창에서 다음과 같이 실행합니다.

    a. [Select service]에서 [Key Management Service]를 선택합니다.

    b. 특정 AWS KMS 작업을 시뮬레이션하려면 작업 선택에서 시뮬레이션할 작업을 선택합니다. 모든 AWS KMS 작업을 시뮬레이션하려면 모두 선택을 선택합니다.
5. (선택 사항) 정책 시뮬레이터는 기본적으로 모든 KMS 키에 대한 액세스를 시뮬레이션합니다. 특정 KMS 키에 대한 액세스를 시뮬레이션하려면 시뮬레이션 설정(Simulation Settings)을 선택한 후 시뮬레이션할 KMS 키의 Amazon 리소스 이름(ARN)을 입력합니다.
6. Run Simulation(시뮬레이션 실행)을 선택합니다.

[Results] 섹션에서 시뮬레이션 결과를 볼 수 있습니다. AWS 계정의 모든 사용자, 그룹, 역할에 대해 2~6단계를 반복합니다.

IAM API로 IAM 정책 검사

IAM API를 사용해 IAM 정책을 프로그래밍 방식으로 검사할 수 있습니다. 다음 절차는 이 방법의 개요를 제공합니다.

1. 키 정책에 보안 주체로 AWS 계정 나열된 각에 대해(즉, 형식으로 지정된 각 AWS 계정 보안 주체: "Principal": {"AWS": "arn:aws:iam::111122223333:root"}) IAM API의 ListUsers 및 ListRoles 작업을 사용하여 계정의 모든 사용자와 역할을 가져옵니다.
2. 목록에 있는 각 사용자 및 역할에 대해 IAM API에서 SimulatePrincipalPolicy 작업을 사용하여 다음 파라미터를 전달합니다.
   - PolicySourceArn에 대해, 목록에서 사용자나 역할의 Amazon 리소스 이름(ARN)을 지정합니다. SimulatePrincipalPolicy 요청별로 한 PolicySourceArn만 지정할 수 있으므로, 이 작업을 목록의 각 사용자 및 역할마다 한 번씩 여러 차례 호출해야 합니다.
   - ActionNames 목록에서 시뮬레이션할 모든 AWS KMS API 작업을 지정합니다. 모든 AWS KMS API 작업을 시뮬레이션하려면를 사용합니다kms:*. 개별 AWS KMS API 작업을 테스트하려면 각 API 작업 앞에 "kms:", 예: "kms:ListKeys"를 붙입니다. AWS KMS API 작업의 목록을 보려면 AWS Key Management Service API 참조의 작업을 참조하세요.
   - (선택 사항) 사용자나 역할이 특정 KMS 키에 액세스할 수 있는지 판단하려면 ResourceArns 파라미터를 이용해 KMS 키의 Amazon 리소스 이름(ARN) 목록을 지정합니다. 사용자나 역할이 모든 KMS 키에 액세스할 수 있는지 판단하기 위해 ResourceArns 파라미터를 생략하세요.

IAM은 각각의 SimulatePrincipalPolicy 요청에 대해 allowed, explicitDeny, 또는 implicitDeny의 평가 결정으로 응답합니다. 의 평가 결정이 포함된 각 응답allowed에 대해 응답에는 허용되는 특정 AWS KMS API 작업의 이름이 포함됩니다. 평가에서 사용된 KMS 키의 ARN도 포함되어 있습니다.

권한 부여 검사

권한 부여는 사용자 또는와 통합된 AWS 서비스가 KMS 키를 사용할 AWS KMS 수 있는 방법과 시기를 지정하는 데 사용할 수 있는 권한을 지정하는 고급 메커니즘입니다. 권한 부여는 KMS 키에 연결되며, 각 권한 부여에는 KMS 키를 사용할 권한을 받는 보안 주체, 허용되는 작업 목록이 포함됩니다. 권한 부여는 키 정책의 대안으로, 특정한 사용 사례에 유용합니다. 자세한 내용은 의 권한 부여 AWS KMS 단원을 참조하십시오.

KMS 키에 대한 권한 부여 목록을 가져오려면 AWS KMS ListGrants 작업을 사용합니다. KMS 키에 대한 권한 부여를 검사하여 현재 이러한 권한 부여를 통해 KMS 키를 사용할 권한이 있는 사람 또는 항목을 파악할 수 있습니다. 로 생성된 권한 부여의 경우 ListGrants 응답에서 "GranteePrincipal" 필드를 GranteePrincipal찾습니다. 로 생성된 권한 부여의 경우 ListGrants 응답에서 "GranteeServicePrincipal" 필드를 GranteeServicePrincipal찾습니다.

예를 들어, 다음은 AWS CLI에서 list-grants 명령으로부터 얻은 권한 부여의 JSON 표현입니다.

```
{"Grants": [{
  "Operations": ["Decrypt"],
  "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
  "Name": "0d8aa621-43ef-4657-b29c-3752c41dc132",
  "RetiringPrincipal": "arn:aws:iam::123456789012:root",
  "GranteePrincipal": "arn:aws:sts::111122223333:assumed-role/aws:ec2-infrastructure/
i-5d476fab",
  "GrantId": "dc716f53c93acacf291b1540de3e5a232b76256c83b2ecb22cdefa26576a2d3e",
  "IssuingAccount": "arn:aws:iam::111122223333:root",
  "CreationDate": 1.444151834E9,
  "Constraints": {"EncryptionContextSubset": {"aws:ebs:id": "vol-5cccfb4e"}}
}]}
```

KMS 키를 사용할 권한이 있는 사람이나 항목을 알아보려면 "GranteePrincipal" 요소를 확인하십시오. 앞의 예에서 피부여자 보안 주체는 EC2 인스턴스 i-5d476fab와 연결된 위임된 역할 사용자입니다. EC2 인프라는 이 역할을 사용해 암호화된 EBS 볼륨 vol-5cccfb4e를 인스턴스에 연결합니다. 이 경우, 사용자가 이전에 이 KMS 키로 보호 받는 암호화된 EBS 볼륨을 생성했기 때문에 EC2 인프라 역할은 KMS 키를 사용할 권한이 있습니다. 그리고 사용자는 EC2 인스턴스에 볼륨을 연결합니다.

다음은 AWS CLI에서 list-grants 명령으로부터 얻은 권한 부여의 또 다른 JSON 표현 예시입니다. 다음 예제에서 피부여자 보안 주체는 또 다른 보안 주체입니다 AWS 계정.

```
{"Grants": [{
  "Operations": ["Encrypt"],
  "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
  "Name": "",
  "GranteePrincipal": "arn:aws:iam::444455556666:root",
  "GrantId": "f271e8328717f8bde5d03f4981f06a6b3fc18bcae2da12ac38bd9186e7925d11",
  "IssuingAccount": "arn:aws:iam::111122223333:root",
  "CreationDate": 1.444151269E9
}]}
```

다음 예제에서는 로 생성된 권한 부여를 보여줍니다GranteeServicePrincipal. 응답에는 대신 GranteeServicePrincipal 필드가 포함GranteePrincipal되고 Constraints 필드에는 특정 AWS 리소스에 대한 권한 부여를 제한하는 SourceArn 값이 포함됩니다.

```
{"Grants": [{
  "Operations": ["Encrypt", "Decrypt", "GenerateDataKey", "DescribeKey"],
```

```
  "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
  "Name": "",
  "GranteeServicePrincipal": "service-name.amazonaws.com",
  "GrantId": "a1b2c3d4e5f6a7b8c9d0e1f2a3b4c5d6e7f8a9b0c1d2e3f4a5b6c7d8e9f0a1b2",
  "IssuingAccount": "arn:aws:iam::111122223333:root",
  "RetiringServicePrincipal": "service-name.amazonaws.com",
  "Constraints": {
    "SourceArn": "arn:aws:dynamodb:us-east-1:111122223333:table/ExampleTable"
  },
  "CreationDate": 1.718567315E9
}]}
```

암호화 컨텍스트

> **Note**
>
> 비대칭 KMS 키 또는 HMAC KMS 키를 사용하여 암호화 작업에서 암호화 컨텍스트를 지정할 수 없습니다. 비대칭 알고리즘 및 MAC 알고리즘은 암호화 컨텍스트를 지원하지 않습니다.

대칭 암호화 KMS 키를 사용하는 모든 AWS KMS 암호화 작업은 데이터에 대한 추가 컨텍스트 정보를 포함할 수 있는 비밀이 아닌 키-값 페어의 선택적 세트인 암호화 컨텍스트를 수락합니다. 의 Encrypt 작업에 암호화 컨텍스트를 삽입 AWS KMS 하여 AWS KMS API 복호화 호출의 권한 부여 및 감사 가능성을 향상시킬 수 있습니다.는 암호화 컨텍스트를 추가 인증 데이터(AAD)로 AWS KMS 사용하여 인증된 암호화를 지원합니다. 암호화 컨텍스트는 사이퍼텍스트에 암호적으로 바인딩되므로 데이터를 복호화하는 데 동일한 암호화 컨텍스트가 필요합니다.

암호화 컨텍스트는 암호가 아니며 암호화되지 않습니다. AWS CloudTrail 로그에 일반 텍스트로 표시되므로 암호화 작업을 식별하고 분류하는 데 사용할 수 있습니다. 암호화 컨텍스트에 민감한 정보가 포함되어서는 안 됩니다. 암호화 컨텍스트가 암호화 또는 해독되는 데이터를 설명하는 것이 좋습니다. 예를 들어 파일을 암호화하는 경우, 파일 경로의 일부를 암호화 컨텍스트로 사용할 수 있습니다.

```
"encryptionContext": {
    "department": "10103.0"
}
```

예를 들어 Amazon Elastic Block Store(Amazon EBS) CreateSnapshot 작업으로 생성된 볼륨 및 스냅샷을 암호화할 때 Amazon EBS는 볼륨 ID를 암호화 컨텍스트 값으로 사용합니다.

```
"encryptionContext": {
  "aws:ebs:id": "vol-abcde12345abc1234"
}
```

암호화 컨텍스트를 사용하여 계정 AWS KMS keys 의에 대한 액세스를 구체화하거나 제한할 수도 있습니다. 암호화 컨텍스트를 권한 부여의 제약 조건으로 그리고 정책문의 조건으로 사용할 수 있습니다. 암호화 컨텍스트 키와 그 값은 aws를 포함하는 임의의 문자열이 될 수 있습니다. 이 값들은 aws:cloudformation:stack-name 같은 AWS 생성 태그와 대조됩니다. 자세한 내용은 kms:EncryptionContext:context-key 섹션을 참조하세요.

암호화 컨텍스트를 사용하여 암호화된 데이터의 무결성을 보호하는 방법을 알아보려면 AWS 보안 블로그의 AWS Key Management Service 및 EncryptionContext를 사용하여 암호화된 데이터의 무결성을 보호하는 방법 게시물을 참조하세요.

암호화 컨텍스트 규칙

AWS KMS 는 암호화 컨텍스트 키 및 값에 대해 다음 규칙을 적용합니다.

- 암호화 컨텍스트 쌍의 키와 값은 단순 리터럴 문자열이어야 합니다. 정수 또는 부동 소수점과 같은 다른 형식을 사용할 경우 AWS KMS 는 이를 문자열로 해석합니다.
- 암호화 컨텍스트의 키와 값에는 유니코드 문자가 포함될 수 있습니다. 암호화 컨텍스트에 키 정책 또는 IAM 정책에서 허용되지 않는 문자가 포함된 경우 `kms:EncryptionContext:context-key` 및 `kms:EncryptionContextKeys`와 같은 정책 조건 키에 암호화 컨텍스트를 지정할 수 없습니다. 키 정책 문서 규칙에 대한 자세한 내용은 키 정책 형식 섹션을 참조하세요. IAM 정책 문서 규칙에 대한 자세한 내용은 IAM 사용 설명서의 IAM 이름 요구 사항 섹션을 참조하세요.

정책의 암호화 컨텍스트

암호화 컨텍스트는 주로 무결성 및 신뢰성을 확인하는 데 사용됩니다. 그러나 암호화 컨텍스트를 사용하여 키 정책 및 IAM 정책에서 대칭 암호화 AWS KMS keys 에 대한 액세스를 제어할 수도 있습니다.

kms:EncryptionContext: 및 kms:EncryptionContextKeys 조건 키는 요청에 특정 암호화 컨텍스트 키 또는 키-값 페어가 포함된 경우에만 권한을 허용(또는 거부)합니다.

예를 들어 다음 키 정책 문은 `RoleForExampleApp` 역할이 `Decrypt` 작업에 KMS 키를 사용하도록 허용합니다. 이 정책 문은 `kms:EncryptionContext:context-key` 조건 키를 사용하여 요청의 암호화 컨텍스트가 `AppName:ExampleApp` 암호화 컨텍스트 쌍을 포함하는 경우에만 이 권한을 허용합니다.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": "kms:Decrypt",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:EncryptionContext:AppName": "ExampleApp"
    }
  }
}
```

이러한 암호화 컨텍스트 키에 대한 자세한 내용은 에 대한 조건 키 AWS KMS 섹션을 참조하세요.

권한 부여의 암호화 컨텍스트

권한 부여를 생성할 때 권한 부여 권한에 대한 조건을 설정하는 권한 부여 제약 조건을 포함할 수 있습니다.는 두 암호화 컨텍스트 권한 부여 제약 조건인 EncryptionContextEquals 및를 AWS KMS 지원하며EncryptionContextSubset, 둘 다 암호화 작업에 대한 요청에 암호화 컨텍스트를 포함합니다. 이러한 권한 부여 제약 조건을 사용할 때 암호화 작업에 대한 요청의 암호화 컨텍스트가 권한 부여 제약 조건의 요구 사항을 충족하는 경우에만 권한이 유효합니다.

예를 들어, GenerateDataKey 작업을 허용하는 권한 부여에 EncryptionContextEquals 권한 부여 제약 조건을 추가할 수 있습니다. 이 제약 조건에서 권한 부여는 요청의 암호화 컨텍스트가 권한 부여 제약 조건의 암호화 컨텍스트와 대소문자 구분이 일치하는 경우에만 작업을 허용합니다.

```
$ aws kms create-grant \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --grantee-principal arn:aws:iam::111122223333:user/exampleUser \
    --retiring-principal arn:aws:iam::111122223333:role/adminRole \
    --operations GenerateDataKey \
    --constraints EncryptionContextEquals={Purpose=Test}
```

피부여자 보안 주체의 다음과 같은 요청은 EncryptionContextEquals 제약 조건을 충족합니다.

```
$ aws kms generate-data-key \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --key-spec AES_256 \
    --encryption-context Purpose=Test
```

권한 부여 제약 조건에 대한 자세한 내용은 권한 부여 제약 사용 섹션을 참조하세요. 권한 부여에 대한 자세한 내용은 the section called “권한 부여” 섹션을 참조하세요.

암호화 컨텍스트 로깅

AWS KMS 는 AWS CloudTrail 를 사용하여 암호화 컨텍스트를 로깅하므로 액세스한 KMS 키와 데이터를 확인할 수 있습니다. 이 로그 항목은 로그 항목에서 암호화 컨텍스트에서 참조한 특정 데이터를 암호화하거나 해독하기 위해 정확히 어떤 KMS 키가 사용되었는지 보여줍니다.

> **Important**
>
> 암호화 컨텍스트가 로깅되기 때문에 민감한 정보가 포함될 수 없습니다.

암호화 컨텍스트 저장

`Decrypt` 또는 `ReEncrypt` 작업 호출 시 암호화 컨텍스트 사용을 간소화하기 위해 암호화된 데이터와 함께 암호화 컨텍스트를 저장할 수 있습니다. 암호화 또는 해독을 위해 필요할 때 전체 암호화 컨텍스트를 만드는 데 도움이 되는 암호화 컨텍스트만 저장하는 것이 좋습니다.

예를 들어 암호화 컨텍스트가 파일의 정규화된 경로인 경우 해당 경로의 일부만 암호화된 파일 내용과 함께 저장합니다. 그런 다음 전체 암호화 컨텍스트가 필요할 때 저장된 조각으로부터 다시 구성합니다. 파일 이름을 변경하거나 다른 위치로 이동하는 등 파일을 변조하면 암호화 컨텍스트 값이 변경되고 해독 요청이 실패합니다.

권한 테스트

를 사용하려면 API 요청을 인증하는 데 사용할 AWS 수 있는 자격 증명이 AWS KMS있어야 합니다. 이러한 보안 인증 정보는 KMS 키 및 별칭에 액세스할 수 있는 권한을 포함해야 합니다. 권한은 키 정책, IAM 정책, 권한 부여, 크로스 계정 액세스 제어에 따라 결정됩니다. KMS 키에 대한 액세스를 제어하는 것 외에도 CloudHSM 및 사용자 지정 키 스토어에 대한 액세스를 제어할 수 있습니다.

`DryRun` API 파라미터를 지정하여 AWS KMS 키를 사용하는 데 필요한 권한이 있는지 확인할 수 있습니다. `DryRun`를 사용하여 AWS KMS API 호출의 요청 파라미터가 올바르게 지정되었는지 확인할 수도 있습니다.

주제

- DryRun 파라미터는 무엇인가요?
- API를 통해 DryRun 지정

DryRun 파라미터는 무엇인가요?

`DryRun`은 AWS KMS API 호출이 성공하는지 확인하기 위해 지정하는 선택적 API 파라미터입니다. 실제로 AWS KMS에 호출하기 전에 API 호출을 테스트하는 데 `DryRun`을 사용합니다. 다음을 확인할 수 있습니다.

- 사용자에게 AWS KMS 키를 사용하는 데 필요한 권한이 있는지 확인할 수 있습니다.
- 호출에서 파라미터를 올바르게 지정했는지 확인할 수 있습니다.

AWS KMS 는 특정 API 작업에서 `DryRun` 파라미터 사용을 지원합니다.

- CreateGrant
- Decrypt
- DeriveSharedSecret
- 암호화
- GenerateDataKey
- GenerateDataKeyPair
- GenerateDataKeyPairWithoutPlaintext
- GenerateDataKeyWithoutPlaintext
- GenerateMac
- ReEncrypt
- RetireGrant
- RevokeGrant
- Sign
- Verify
- VerifyMac

`DryRun` 파라미터를 사용하면 요금이 부과되며 표준 API 요청으로 요금이 청구됩니다. AWS KMS 요금에 대한 자세한 내용은 AWS Key Management Service 요금을 참조하세요.

`DryRun` 파라미터를 사용하는 모든 API 요청은 API의 요청 할당량에 적용되며, API 요청 할당량을 초과할 경우 스로틀링 예외가 발생할 수 있습니다. 예를 들어, `DryRun`이 있거나 `DryRun` 없이 Decrypt를 호출하면 동일한 암호화 작업 할당량으로 간주됩니다. 자세한 내용은 AWS KMS 요청 제한 섹션을 참조하세요.

AWS KMS API 작업에 대한 모든 호출은 이벤트로 캡처되고 AWS CloudTrail 로그에 기록됩니다. DryRun 파라미터를 지정하는 모든 작업의 출력은 CloudTrail 로그에 표시됩니다. 자세한 내용은 를 사용하여 AWS KMS API 호출 로깅 AWS CloudTrail 단원을 참조하십시오.

API를 통해 DryRun 지정

를 사용하려면 –dry-run 파라미터를 지원하는 명령 및 AWS KMS API 호출에서 AWS CLI 파라미터를 DryRun지정합니다. 이렇게 하면 AWS KMS 가 호출이 성공할지 확인합니다.를 사용하는 AWS KMS 호출DryRun은 항상 실패하고 호출이 실패한 이유에 대한 정보가 포함된 메시지를 반환합니다. 메시지에는 다음과 같은 예외 사항이 포함될 수 있습니다.

- DryRunOperationException - DryRun이 지정되지 않으면 요청이 성공합니다.
- ValidationException- 잘못된 API 파라미터를 지정하여 요청이 실패했습니다.
- AccessDeniedException- KMS 리소스에 대해 지정된 API 작업을 수행할 권한이 없습니다.

예를 들어 다음 명령은 CreateGrant 작업을 사용하고 keyUserRole 역할을 수임할 권한이 있는 사용자가 지정된 대칭 KMS 키에 대해 Decrypt 작업을 호출할 수 있도록 허용하는 권한 부여를 생성합니다. DryRun 파라미터가 지정되었습니다.

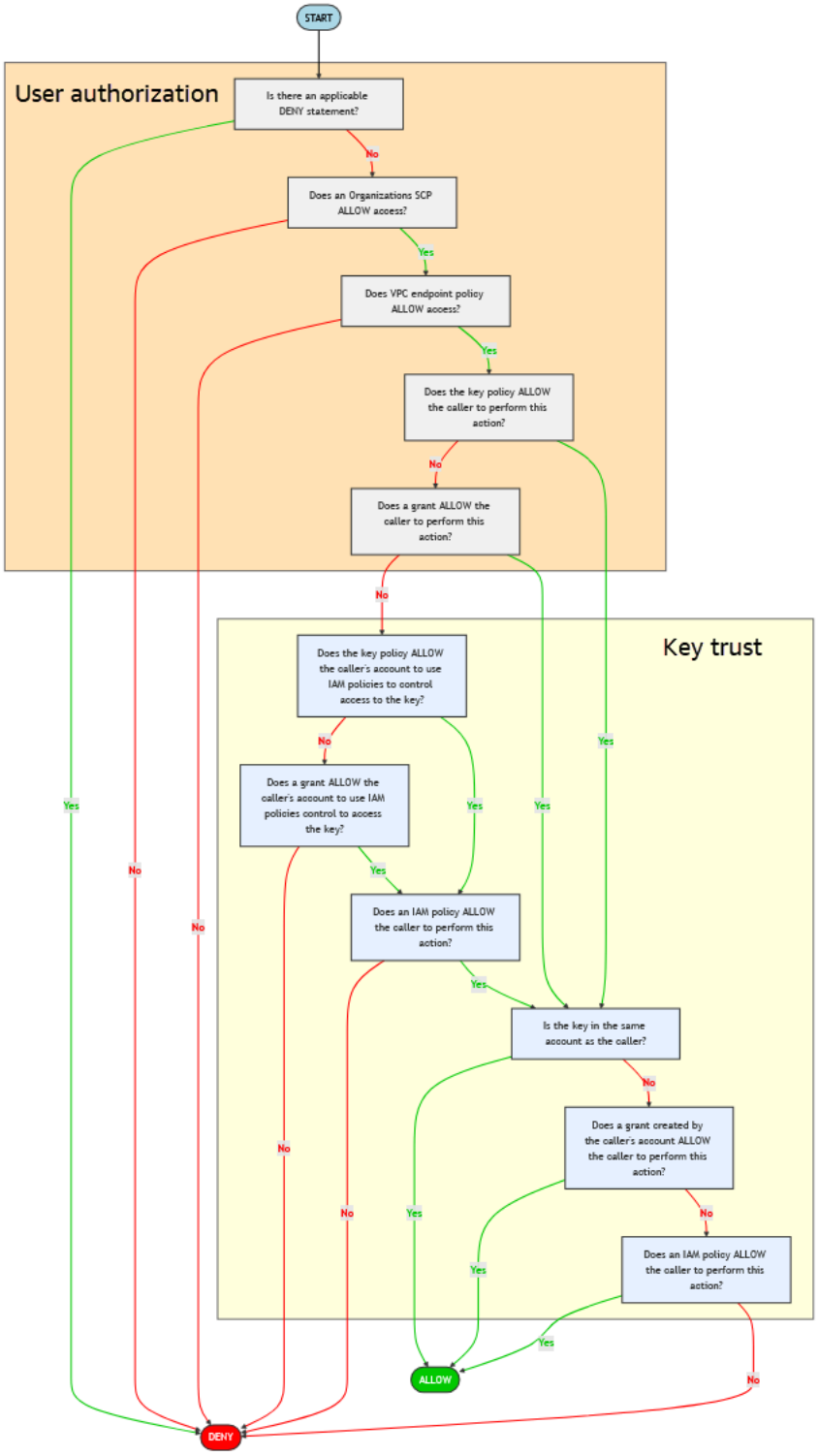
```
$  aws kms create-grant \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --grantee-principal arn:aws:iam::111122223333:role/keyUserRole \
    --operations Decrypt \
    --dry-run
```

AWS KMS 권한 문제 해결

KMS 키에 대한 액세스 권한을 부여할 때는 다음을 AWS KMS 평가합니다.

- KMS 키에 연결된 키 정책. 키 정책은 항상 KMS 키를 소유한 AWS 계정 및 리전에 정의됩니다.
- 요청을 수행하는 사용자 또는 역할에 연결된 모든 IAM 정책입니다. 보안 주체의 KMS 키 사용을 제어하는 IAM 정책은 항상 해당 보안 주체의 AWS 계정입니다.
- KMS 키에 적용되는 모든 권한 부여
- KMS 키를 사용하라는 요청에 적용될 수 있는 다른 유형의 정책(예:AWS Organizations 서비스 제어 정책 및 VPC 엔드포인트 정책). 이러한 정책은 선택 사항이며 기본적으로 모든 작업을 허용하지만 보안 주체에 부여된 권한을 제한하는 데 사용할 수 있습니다.

AWS KMS 는 이러한 정책 메커니즘을 함께 평가하여 KMS 키에 대한 액세스가 허용 또는 거부되는지 여부를 결정합니다. 이렇게 하려면 다음 순서도에 표시된 것과 유사한 프로세스를 AWS KMS 사용합니다. 다음 흐름 차트에는 정책 평가 프로세스가 시각적으로 표시됩니다.



이 흐름 차트는 두 부분으로 나뉩니다. 각 부분은 차례로 표시되지만 대개 동시에 평가됩니다.

- 권한 부여 사용에서는 해당 키 정책, IAM 정책, 권한 부여 및 기타 적용 가능한 정책을 기준으로 KMS 키 사용이 허용되는지 여부를 확인합니다.
- 키 신뢰에서는 사용이 허용된 KMS 키를 신뢰해야 하는지 여부를 결정합니다. 일반적으로의 리소스를 신뢰합니다 AWS 계정. 그러나 계정의 권한 부여 또는 IAM 정책이 KMS 키를 사용하도록 허용하는 AWS 계정 경우 다른에서 KMS 키를 사용하는 것에 대해 확신을 가질 수도 있습니다.

이 흐름 차트를 통해 호출자의 KMS 키 사용 권한이 부여되거나 거부된 이유를 확인할 수 있습니다. 사용자의 정책과 권한을 평가할 수도 있습니다. 예를 들어 이 흐름 차트에서는 키 정책, IAM 정책 또는 권한 부여에 명시적 DENY 문이 있거나 명시적 ALLOW 문이 없어서 호출자의 액세스가 거부될 수 있음을 보여줍니다.

이 흐름 차트를 사용하여 몇 가지 일반적인 권한 시나리오를 설명하겠습니다.

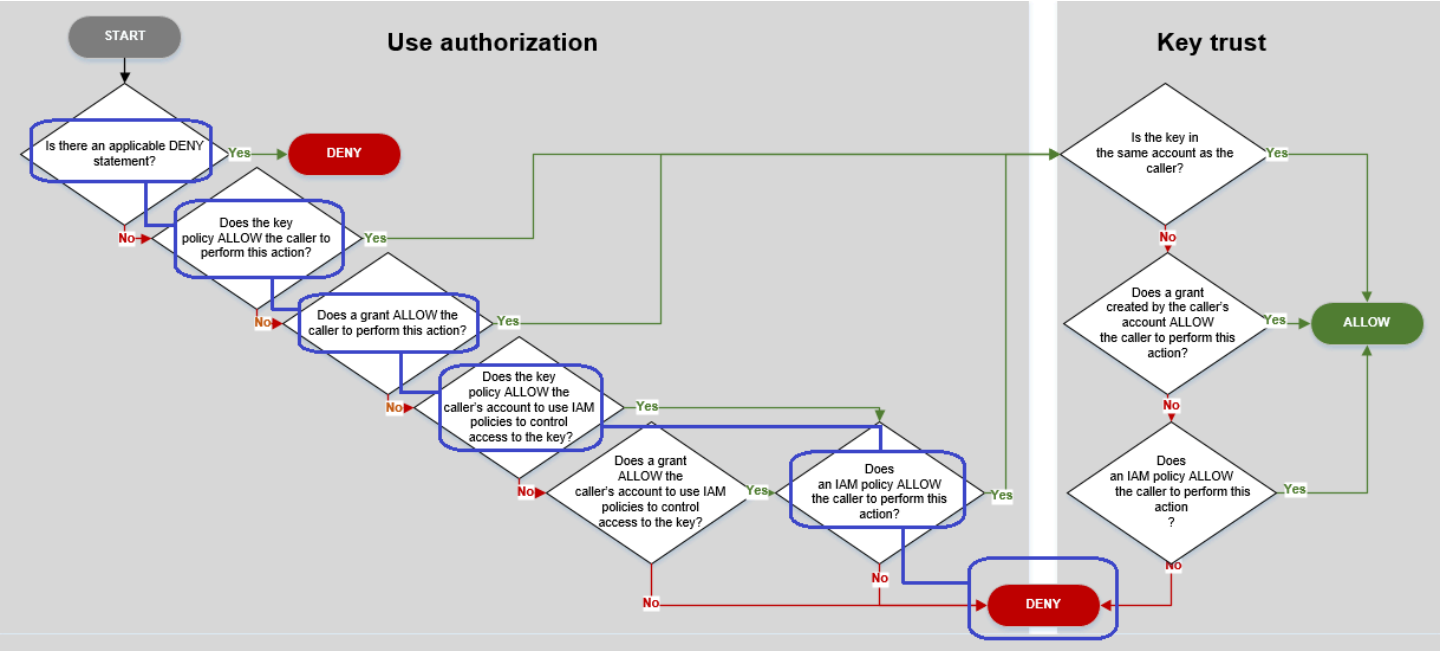
권한 예제

- 예제 1: 사용자의에서 KMS 키에 대한 액세스가 거부됨 AWS 계정
- 예제 2: 사용자가 다른에서 KMS 키를 사용할 권한이 있는 역할을 맡음 AWS 계정

예제 1: 사용자의에서 KMS 키에 대한 액세스가 거부됨 AWS 계정

Alice는 111122223333 AWS 계정의 IAM 사용자입니다. Alice는 동일한 AWS 계정의 KMS 키에 액세스할 수 없습니다. Alice가 KMS 키를 사용할 수 없는 이유는 무엇입니까?

이 경우 Alice는 본인에게 필요한 권한을 제공하는 권한 부여, IAM 정책 또는 키 정책이 없으므로 KMS 키에 대한 액세스가 거부됩니다. KMS 키의 키 정책은가 IAM 정책을 사용하여 KMS 키에 대한 액세스를 제어 AWS 계정 하도록 허용하지만 IAM 정책은 Alice에게 KMS 키를 사용할 수 있는 권한을 부여하지 않습니다.



이 예제와 관련된 정책을 살펴보겠습니다.

- Alice가 사용할 KMS 키에는 기본 키 정책이 있습니다. 이 정책은 KMS 키를 소유한 AWS 계정이 IAM 정책을 사용하여 KMS 키에 대한 액세스를 제어하도록 허용합니다. 이 키 정책은 순서도에 있는 키 정책으로 인해 호출자의 계정이 IAM 정책을 사용하여 키에 대한 액세스를 제어할 수 있습니까? 조건을 충족합니다.

  JSON

```
{
  "Version":"2012-10-17",
  "Id" : "key-test-1",
  "Statement" : [ {
    "Sid" : "DelegateToIAMPolicies",
    "Effect" : "Allow",
    "Principal" : {
      "AWS" : "arn:aws:iam::111122223333:root"
    },
    "Action" : "kms:*",
    "Resource" : "*"
  } ]
}
```

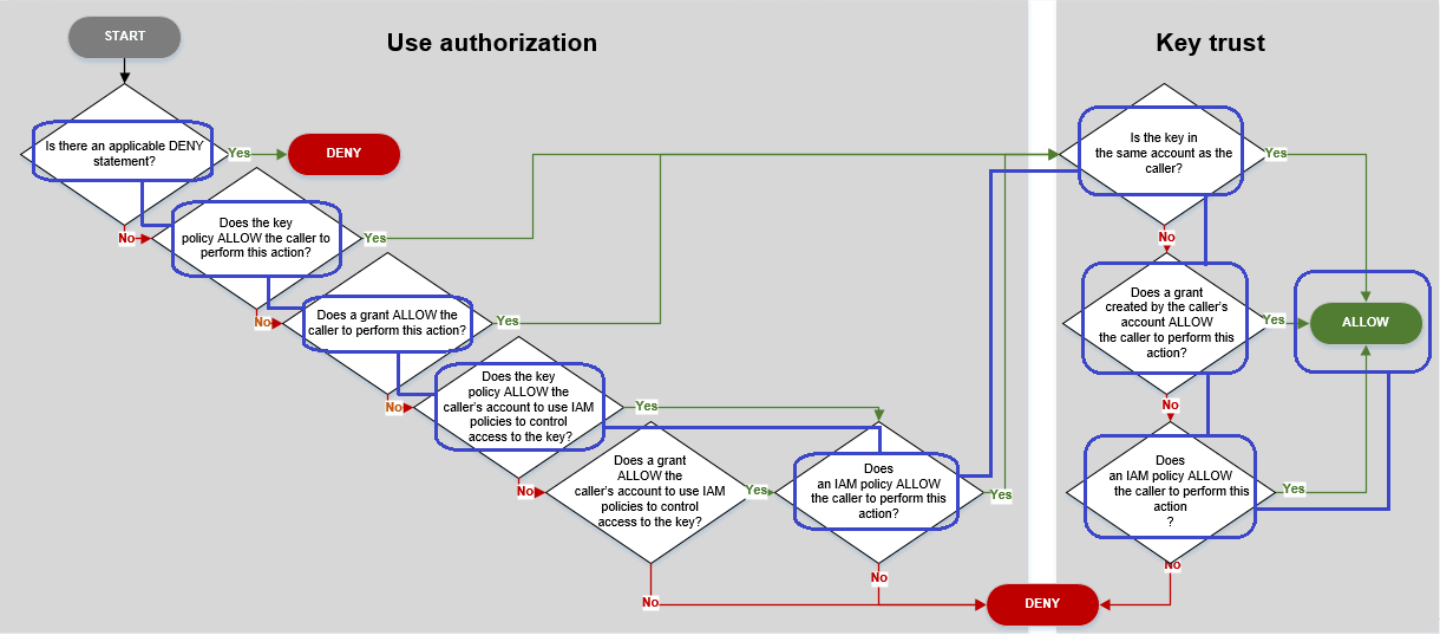
- 그러나 Alice에게는 키 정책, IAM 정책이 없고 KMS 키사용 권한도 부여되지 않았습니다. 따라서 Alice가 KMS 키를 사용할 수 있는 권한이 거부됩니다.

예제 2: 사용자가 다른에서 KMS 키를 사용할 권한이 있는 역할을 맡음 AWS 계정

Bob은 계정 1(111122223333)의 사용자입니다. Bob은 암호화 작업에서 계정 2(444455556666)의 KMS 키를 사용할 수 있습니다. 이렇게 할 수 있는 이유는 무엇입니까?

> **Tip**
>
> 교차 계정 권한을 평가할 때는 키 정책이 KMS 키의 계정에 지정되어 있다는 점을 명심하십시오. IAM 정책은 호출자가 다른 계정에 속하더라도 호출자의 계정에 지정됩니다. KMS 키에 대한 교차 계정 액세스를 제공하는 방법에 대한 자세한 내용은 다른 계정의 사용자가 KMS를 사용하도록 허용 단원을 참조하십시오.

- 계정 2의 KMS 키에 대한 키 정책에 따라 계정 2는 IAM 정책을 사용하여 KMS 키에 대한 액세스를 제어할 수 있습니다.
- 계정 2의 KMS 키에 대한 키 정책을 통해 계정 1이 암호화 작업에서 KMS 키를 사용할 수 있습니다. 그러나 계정 1은 IAM 정책을 사용하여 KMS 키에 액세스할 수 있는 권한을 해당 보안 주체에 제공해야 합니다.
- 계정 1의 IAM 정책은 `Engineering` 역할이 암호화 작업에 계정 2의 KMS 키를 사용하도록 허용합니다.
- 계정 1의 사용자인 Bob은 `Engineering` 역할을 맡을 수 있는 권한을 가집니다.
- Bob은 이 KMS 키를 신뢰할 수 있습니다. 이 키가 그의 계정에 없는 경우에도 그의 계정에 있는 IAM 정책이 이 KMS 키를 사용할 수 있는 명시적 권한을 부여하기 때문입니다.



계정 1의 사용자인 Bob이 계정 2로 KMS 키를 사용할 수 있도록 허용하는 정책을 살펴보겠습니다.

- KMS 키의 키 정책에 따라 계정 2(444455556666, KMS 키를 소유 계정)가 IAM 정책을 사용하여 KMS 키에 대한 액세스를 제어할 수 있습니다. 또한 이 키 정책을 통해 계정 1(111122223333)이 정책문의 `Action` 요소에 지정된 암호화 작업에 KMS 키를 사용할 수 있습니다. 그러나 계정 1이 KMS 키 액세스 권한을 보안 주체에 제공하는 IAM 정책을 정의할 때까지는 계정 1의 어떤 사용자도 계정 2로 KMS 키를 사용할 수 없습니다.

  이 흐름 차트에서 계정 2의 이 키 정책은 키 정책에서 호출자의 계정이 IAM 정책을 사용하여 키 액세스를 제어하도록 허용하는가? 조건을 충족합니다.

  JSON

```
{
    "Id": "key-policy-acct-2",
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "PermissionUseIAMpolicies",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::444455556666:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
```

```
        {
            "Sid": "AllowAccount1UseThisKMSKeys",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncryptFrom",
                "kms:ReEncryptTo",
                "kms:GenerateDataKey",
                "kms:GenerateDataKeyWithoutPlaintext",
                "kms:DescribeKey"
            ],
            "Resource": "*"
        }
    ]
}
```

- 호출자 AWS 계정 (계정 1, 111122223333)의 IAM 정책은 보안 주체에게 계정 2(444455556666)의 KMS 키를 사용하여 암호화 작업을 수행할 수 있는 권한을 부여합니다. `Action` 요소는 계정 2의 키 정책이 계정 1에 부여한 것과 동일한 권한을 보안 주체에게 위임합니다. 계정 1의 `Engineering` 역할에 이러한 권한을 부여하기 위해 [이 인라인 정책이 `Engineering` 역할에 포함](#)됩니다.

  이와 같은 계정 간 IAM 정책은 계정 2의 KMS 키에 대한 키 정책이 KMS 키를 사용할 수 있는 권한을 계정 1에 제공할 때에만 적용됩니다. 또한 계정 1은 키 정책이 계정에 제공한 작업을 수행할 수 있는 권한만 해당 보안 주체에 제공할 수 있습니다.

  이것은 흐름 차트에서 IAM 정책에서 호출자가 이 작업을 수행하도록 허용하는가? 조건을 충족합니다.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncryptFrom",
```

```
                "kms:ReEncryptTo",
                "kms:GenerateDataKey",
                "kms:GenerateDataKeyWithoutPlaintext",
                "kms:DescribeKey"
            ],
            "Resource": [
            "arn:aws:kms:us-
west-2:444455556666:key/1234abcd-12ab-34cd-56ef-1234567890ab"
            ]
        }
    ]
}
```

- 마지막 필수 요소는 계정 1의 Engineering 역할 정의입니다. 역할의 AssumeRolePolicyDocument에 따라 Bob은 Engineering 역할을 맡을 수 있습니다.

```
{
    "Role": {
        "Arn": "arn:aws:iam::111122223333:role/Engineering",
        "CreateDate": "2019-05-16T00:09:25Z",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": {
                "Principal": {
                    "AWS": "arn:aws:iam::111122223333:user/bob"
                },
                "Effect": "Allow",
                "Action": "sts:AssumeRole"
            }
        },
        "Path": "/",
        "RoleName": "Engineering",
        "RoleId": "AROA4KJY2TU23Y7NK62MV"
    }
}
```

AWS KMS 액세스 제어 용어집

다음 주제에서는 AWS KMS 액세스 제어의 중요한 용어와 개념을 설명합니다.

Authentication

인증은 ID를 확인하는 프로세스입니다. 요청을 보내려면 자격 증명을 AWS 사용하여 AWS 에 로그인 AWS KMS해야 합니다.

권한 부여

권한 부여는 AWS KMS 리소스를 생성, 관리 또는 사용하기 위한 요청을 보낼 수 있는 권한을 제공합니다. 예를 들어 암호화 작업에 KMS 키를 사용할 권한이 있어야 합니다.

AWS KMS 리소스에 대한 액세스를 제어하려면 키 정책, IAM 정책 및 권한 부여를 사용합니다. 모든 KMS 키에는 키 정책이 있어야 합니다. 키 정책에서 허용되는 경우 IAM 정책 및 권한 부여를 사용하여 보안 주체에게 KMS 키에 대한 액세스 권한을 부여할 수도 있습니다. 권한 부여를 구체화하기 위해서는 요청 또는 리소스가 지정한 조건을 충족하는 경우에만 액세스를 허용하거나 거부하는 조건 키를 사용할 수 있습니다. 다른 AWS 계정에서 신뢰하는 보안 주체에 대한 액세스를 허용할 수도 있습니다.

ID를 통한 인증

인증은 자격 증명 자격 증명을 AWS 사용하여에 로그인하는 방법입니다. AWS 계정 루트 사용자, IAM 사용자 또는 IAM 역할을 수임하여 인증해야 합니다.

AWS IAM Identity Center (IAM Identity Center), Single Sign-On 인증 또는 Google/Facebook 자격 증명과 같은 자격 증명 소스의 자격 증명을 사용하여 페더레이션 자격 증명으로 로그인할 수 있습니다. 로그인하는 방법에 대한 자세한 내용은 AWS 로그인 사용 설명서의 AWS 계정에 로그인하는 방법 섹션을 참조하세요.

프로그래밍 방식 액세스를 위해는 요청에 암호화 방식으로 서명할 수 있는 SDK 및 CLI를 AWS 제공합니다. 자세한 내용은 IAM 사용 설명서의 API 요청용AWS Signature Version 4 섹션을 참조하세요.

AWS 계정 루트 사용자

를 생성할 때 모든 AWS 서비스 및 리소스에 대한 완전한 액세스 권한이 있는 AWS 계정 theroot 사용자라는 하나의 로그인 자격 증명으로 AWS 계정시작합니다. 일상적인 태스크에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 자격 증명이 필요한 작업은 IAM 사용 설명서의 루트 사용자 자격 증명이 필요한 작업 섹션을 참조하세요.

페더레이션 ID

가장 좋은 방법은 인간 사용자에게 자격 증명 공급자와의 페더레이션을 사용하여 임시 자격 증명을 AWS 서비스 사용하여에 액세스하도록 요구하는 것입니다.

페더레이션 자격 증명은 엔터프라이즈 디렉터리, 웹 자격 증명 공급자 또는 자격 증명 소스의 자격 증명을 AWS 서비스 사용하여 Directory Service 에 액세스하는 사용자의 자격 증명입니다. 페더레이션 ID는 임시 자격 증명을 제공하는 역할을 수임합니다.

중앙 집중식 액세스 관리를 위해 AWS IAM Identity Center를 추천합니다. 자세한 정보는 AWS IAM Identity Center 사용 설명서의 What is IAM Identity Center?를 참조하세요.

IAM 사용자 및 그룹

IAM 사용자는 단일 개인 또는 애플리케이션에 대한 특정 권한을 가진 ID입니다. 장기 자격 증명이 있는 IAM 사용자 대신 임시 자격 증명을 사용하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 자격 증명 공급자와의 페더레이션을 사용하여 임시 자격 증명을 AWS 사용하여에 액세스하도록 인간 사용자에게 요구하기를 참조하세요.

IAM 그룹은 IAM 사용자 모음을 지정하고 대규모 사용자 집합에 대한 관리 권한을 더 쉽게 만듭니다. 자세한 내용은 IAM 사용 설명서의 IAM 사용자 사용 사례 섹션을 참조하세요.

IAM 역할

IAM 역할은 임시 자격 증명을 제공하는 특정 권한이 있는 자격 증명입니다. 사용자에서 IAM 역할(콘솔)로 전환하거나 또는 API 작업을 호출하여 역할을 수임할 수 있습니다. AWS CLI AWS 자세한 내용은 IAM 사용 설명서의 역할 수임 방법을 참조하세요.

IAM 역할은 페더레이션 사용자 액세스, 임시 IAM 사용자 권한, 교차 계정 액세스, 교차 서비스 액세스 및 Amazon EC2에서 실행되는 애플리케이션에 유용합니다. 자세한 내용은 IAM 사용 설명서의 교차 계정 리소스 액세스를 참조하세요.

정책을 사용하여 액세스 관리

정책을 AWS 생성하고 자격 증명 또는 리소스에 연결하여 AWS 에서 액세스를 제어합니다. 정책은 자격 증명 또는 리소스와 연결될 때 권한을 정의합니다.는 보안 주체가 요청할 때 이러한 정책을 AWS 평가합니다. 대부분의 정책은에 JSON 문서 AWS 로 저장됩니다. JSON 정책 문서에 대한 자세한 내용은 IAM 사용 설명서의 JSON 정책 개요 섹션을 참조하세요.

정책을 사용하여 관리자는 어떤 보안 주체가 어떤 리소스에 대해 어떤 조건에서 작업을 수행할 수 있는지 정의하여 누가 무엇을 액세스할 수 있는지 지정합니다.

기본적으로 사용자 및 역할에는 어떠한 권한도 없습니다. IAM 관리자는 IAM 정책을 생성하고 사용자가 수임할 수 있는 역할에 추가합니다. IAM 정책은 작업을 수행하기 위해 사용하는 방법과 관계없이 작업에 대한 권한을 정의합니다.

ID 기반 정책

ID 기반 정책은 ID(사용자, 사용자 그룹 또는 역할)에 연결하는 JSON 권한 정책 문서입니다. 이러한 정책은 자격 증명이 수행할 수 있는 작업, 대상 리소스 및 이에 관한 조건을 제어합니다. ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서에서 고객 관리형 정책으로 사용자 지정 IAM 권한 정의를 참조하세요.

ID 기반 정책은 인라인 정책(단일 ID에 직접 포함) 또는 관리형 정책(여러 ID에 연결된 독립 실행형 정책)일 수 있습니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 관리형 정책 및 인라인 정책 중에서 선택 섹션을 참조하세요.

리소스 기반 정책

AWS KMS Ankey 정책은 KMS 키에 대한 액세스를 제어하는 리소스 기반 정책입니다. 모든 KMS 키에는 키 정책이 있어야 합니다. 다른 인증 메커니즘을 사용하여 KMS 키에 대한 액세스를 허용할 수 있지만 이는 키 정책에서 허용되는 경우에만 가능합니다. (키 정책에서 명시적으로 허용되지 않는 경우에도 IAM 정책을 사용하여 KMS 키에 대한 액세스를 거부할 수 있습니다.)

리소스 기반 정책은 특정 리소스에 대한 액세스를 제어하기 위하여 리소스(예: KMS 키)에 연결하는 JSON 정책 문서입니다. 리소스 기반 정책은 지정된 보안 주체가 해당 리소스에 대해 어떤 조건에서 어떤 작업을 수행할 수 있는지 정의합니다. 리소스 기반 정책에서는 리소스를 지정하지 않지만 계정, 사용자, 역할, 페더레이션 사용자 또는 같은 보안 주체를 지정해야 합니다 AWS 서비스. 리소스 기반 정책은 리소스를 관리하는 해당 서비스에 있는 인라인 정책입니다. 리소스 기반 정책에서는 AWS 관리형 정책과 같은 IAM의 `AWSKeyManagementServicePowerUser` 관리형 정책을 사용할 수 없습니다.

기타 정책 유형

AWS 는 보다 일반적인 정책 유형에서 부여한 최대 권한을 설정할 수 있는 추가 정책 유형을 지원합니다.

- 권한 경계 - ID 기반 정책에서 IAM 엔터티에 부여할 수 있는 최대 권한을 설정합니다. 자세한 정보는 IAM 사용 설명서의 IAM 엔터티의 권한 범위를 참조하세요.
- 서비스 제어 정책(SCP) - AWS Organizations내 조직 또는 조직 단위에 대한 최대 권한을 지정합니다. 자세한 내용은AWS Organizations 사용 설명서의 서비스 제어 정책을 참조하세요.
- 리소스 제어 정책(RCP) – 계정의 리소스에 사용할 수 있는 최대 권한을 설정합니다. 자세한 내용은 AWS Organizations 사용 설명서의 리소스 제어 정책(RCP)을 참조하세요.
- 세션 정책 – 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 자세한 내용은 IAM 사용 설명서의 세션 정책을 참조하세요.

여러 정책 유형

여러 정책 유형이 요청에 적용되는 경우, 결과 권한은 이해하기가 더 복잡합니다. 에서 여러 정책 유형이 관련될 때 요청을 허용할지 여부를 AWS 결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하세요.

AWS KMS 리소스

에서 AWS KMS기본 리소스는 a입니다 AWS KMS key.는 KMS 키에 대한 표시 이름을 제공하는 독립 리소스인 별칭 AWS KMS 도 지원합니다. 일부 AWS KMS 작업을 사용하면 별칭을 사용하여 KMS 키를 식별할 수 있습니다.

KMS 키 또는 별칭의 각 인스턴스에는 표준 형식의 고유한 Amazon 리소스 이름(ARN)이 있습니다. AWS KMS 리소스에서 AWS 서비스 이름은 입니다kms.

- AWS KMS key

  ARN 형식:

  ```
  arn:AWS partition name:AWS service name:AWS ##:AWS ## ID:key/key ID
  ```

  ARN 예:

  ```
  arn:aws:kms:us-
  west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab
  ```

- 별칭

  ARN 형식:

  ```
  arn:AWS partition name:AWS service name:AWS ##:AWS ## ID:alias/alias name
  ```

  ARN 예:

  ```
  arn:aws:kms:us-west-2:111122223333:alias/example-alias
  ```

AWS KMS 는 AWS KMS 리소스와 함께 사용할 수 있는 API 작업 세트를 제공합니다. AWS Management Console 및 AWS KMS API 작업에서 KMS 키를 식별하는 방법에 대한 자세한 내용은 섹션을 참조하세요키 식별자(KeyId). AWS KMS 작업 목록은 AWS Key Management Service API 참조를 참조하세요.

KMS 키 생성

AWS KMS keys 에서 AWS Management Console또는 CreateKey 작업 또는 AWS::KMS::Key AWS CloudFormation 리소스를 사용하여를 생성할 수 있습니다. 이 과정에서 KMS 키에 대한 키 정책을 설정할 수 있으며, 이 정책은 언제든지 변경할 수 있습니다. 또한 생성하는 KMS 키 유형을 정의하는 다음 값도 선택합니다. KMS 키가 생성된 후에는 이러한 속성을 변경할 수 없습니다.

KMS 키 유형

키 유형은 생성되는 암호화 키 유형을 결정하는 속성입니다.는 데이터를 보호하기 위해 세 가지 키 유형을 AWS KMS 제공합니다.

- 고급 암호화 표준(AES) 대칭 키

  AES의 GCM(Galois Counter Mode) 모드에서 4KB 미만의 데이터에 대해 인증된 암호화/복호화를 제공하는 데 사용되는 256비트 키입니다. 이는 가장 일반적인 유형의 키이며 애플리케이션에서 사용되는 다른 데이터 암호화 키를 보호하고 사용자를 대신하여 데이터를 암호화하는 AWS 서비스 데 사용됩니다.

- RSA, 타원 곡선 또는 SM2(중국 리전만 해당) 비대칭 키

  이러한 키는 다양한 크기로 사용 가능하며 많은 알고리즘을 지원합니다. 이러한 키는 알고리즘 선택에 따라 암호화 및 복호화, 서명 및 확인 또는 공유 보안 비밀 작업에 사용할 수 있습니다.

- 해시 기반 메시지 인증 코드(HMAC) 작업을 수행하기 위한 대칭 키

  이러한 키는 서명 및 확인 작업에 사용되는 256비트 키입니다.

  KMS 키는 서비스에서 일반 텍스트로 내보낼 수 없습니다. KMS 키는 서비스에서 사용하는 하드웨어 보안 모듈(HSM)에 의해 생성되고 해당 보안 모듈 내에서만 사용될 수 있습니다. 이는 키가 손상되지 않도록 AWS KMS 하기 위한의 기본 보안 속성입니다.

키 사용

키 사용은 키가 지원하는 암호화 작업을 결정하는 속성입니다. KMS 키는 `ENCRYPT_DECRYPT`, `SIGN_VERIFY` `GENERATE_VERIFY_MAC` 또는 `KEY_AGREEMENT`의 키 사용을 가질 수 있습니다. 각 KMS 키에는 하나의 키 사용만 있을 수 있습니다. 이는 미국 국립표준기술원(NIST) 특별 간행물 800-57 키 관리 권고 사항의 섹션 5.2, 키 사용에 따른 키 사용 모범 사례를 따르는 것입니다. 두 가지 이상의 작업 유형에 KMS를 사용하면 두 작업의 결과물이 공격에 더 취약해집니다.

키 사양

키 사양은 키의 암호화 구성을 나타내는 속성입니다. 키 사양의 의미는 키 유형에 따라 다릅니다.

KMS 키의 경우 키 사양은 KMS 키가 대칭인지 또는 비대칭인지 여부를 결정합니다. 또한 키 구성 요소의 유형과 지원하는 알고리즘을 결정합니다.

기본 키 사양인 SYMMETRIC_DEFAULT는 256비트 대칭 암호화 키를 나타냅니다. 지원되는 모든 키 사양에 대한 자세한 설명은 키 사양 참조 단원을 참조하세요.

키 구성 요소 오리진

키 구성 요소 오리진은 KMS 키에서 키 구성 요소의 소스를 식별하는 KMS 키 속성입니다. 키 구성 요소 오리진은 KMS 키를 생성할 때 선택하며 이후에는 변경할 수 없습니다. 키 구성 요소의 소스는 KMS 키의 보안, 내구성, 가용성, 지연 시간 및 처리량 특성에 영향을 미칩니다.

각 KMS 키는 메타데이터에 해당 키 구성 요소에 대한 참조를 포함합니다. 대칭 암호화 KMS 키의 키 구성 요소 오리진은 다를 수 있습니다. AWS KMS 에서 생성하는 키 구성 요소, 사용자 지정 키 저장소에서 생성된 키 구성 요소를 사용하거나 고유한 키 구성 요소를 가져올 수 있습니다.

기본적으로 각 KMS 키에는 고유한 키 구성 요소가 있습니다. 그러나 동일한 키 구성 요소를 사용하여 다중 리전 키 집합을 생성할 수 있습니다

KMS 키는 AWS_KMS, EXTERNAL (가져온 키 구성 요소), (키 스토어의 KMS 키) 또는 (외부 키 스토어의 KMS 키) 중 하나의 키 구성 요소 오리진 값을 가질 수 있습니다. AWS_CLOUDHSM AWS CloudHSM EXTERNAL_KEY_STORE ???

주제

- KMS 키를 생성하기 위한 사용 권한
- 생성할 KMS 키 유형 선택
- 대칭 암호화 KMS 키 생성
- 비대칭 KMS 키 생성
- HMAC KMS 키 생성
- 다중 리전 프라이머리 키 생성
- 다중 리전 복제본 키 생성
- 가져온 키 구성 요소가 있는 KMS 키 생성
- 키 스토어에서 KMS AWS CloudHSM 키 생성
- 외부 키 저장소에서 KMS 키 생성

KMS 키를 생성하기 위한 사용 권한

콘솔에서 또는 API를 사용하여 KMS 키를 생성하려면 IAM 정책에서 다음 권한이 있어야 합니다. 가능하면 조건 키를 사용하여 사용 권한을 제한합니다. 예를 들어 IAM 정책에서 kms:KeySpec 조건 키를 사용하여 보안 주체가 대칭 암호화 키만을 생성하도록 할 수 있습니다.

키를 생성하는 보안 주체에 대한 IAM 정책의 예는 사용자가 KMS 키를 생성할 수 있도록 허용 섹션을 참조하세요.

> **Note**
>
> 보안 주체에게 태그 및 별칭을 관리할 수 있는 권한을 부여하는 데 주의해야 합니다. 태그 또는 별칭을 변경하면 고객 관리 키에 대한 권한을 허용하거나 거부할 수 있습니다. 자세한 내용은 에 대한 ABAC AWS KMS 섹션을 참조하세요.

- kms:CreateKey는 필수입니다.
- kms:CreateAlias는 콘솔에서 모든 새 KMS 키에 별칭이 필요한 KMS 키를 만드는 데 필요합니다.
- kms:TagResource는 KMS 키를 만드는 동안 태그를 추가하는 데 필요합니다.
- iam:CreateServiceLinkedRole는 다중 리전 기본 키를 만드는 데 필요합니다. 자세한 내용은 다중 리전 키에 대한 액세스 제어 섹션을 참조하세요.

kms:PutKeyPolicy 권한은 KMS 키를 만드는 데 필요하지 않습니다. `kms:CreateKey` 권한에는 초기 키 정책을 설정할 수 있는 권한이 포함됩니다. 그러나 KMS 키에 대한 액세스를 제어할 수 있도록 KMS 키를 만드는 동안 이 권한을 키 정책에 추가해야 합니다. 또 다른 방법은 BypassLockoutSafetyCheck 파라미터를 사용하는 것이지만, 이는 사용하지 않는 것이 좋습니다.

KMS 키는 키가 생성된 AWS 계정에 속합니다. KMS 키를 생성하는 IAM 사용자는 키 소유자로 간주되지 않으며, 사용자가 생성한 KMS 키를 사용하거나 관리할 수 있는 권한이 자동으로 부여되지 않습니다. 다른 보안 주체와 마찬가지로 키 생성자는 키 정책, IAM 정책 또는 권한 부여를 통해 허가를 받아야 합니다. 그러나, `kms:CreateKey` 권한이 있는 보안 주체는 초기 키 정책을 설정하고 키를 사용하거나 관리할 권한을 자신에게 부여할 수 있습니다.

생성할 KMS 키 유형 선택

생성할 KMS 키 유형은 주로 KMS 키 사용 계획, 보안 요구 사항 및 권한 부여 요구 사항에 따라 결정됩니다. KMS 키의 키 유형 및 키 사용에 따라 키가 수행할 수 있는 암호화 작업이 결정됩니다. 각 KMS 키

에는 하나의 키 사용만 있습니다. 두 가지 이상의 작업 유형에 KMS를 사용하면 모든 작업의 결과물이 공격에 더 취약해집니다.

보안 주체가 특정 키 사용에 대해서만 KMS 키를 생성하도록 하려면 kms:KeyUsage 조건 키를 사용합니다. kms:KeyUsage 조건 키를 사용하여 보안 주체가 해당 키 사용을 기반으로 KMS 키에 대한 API 작업을 호출하도록 허용할 수도 있습니다. 예를 들어 키 사용이 SIGN_VERIFY인 경우에만 KMS 키를 비활성화할 수 있는 권한을 허용할 수 있습니다.

사용 사례에 따라 필요한 KMS 키 유형을 결정하려면 다음 지침을 따르세요.

암호화 및 해독

데이터 암호화 및 해독이 필요한 대부분의 사용 사례에는 대칭 KMS 키를 사용합니다. AWS KMS에서 사용하는 대칭 암호화 알고리즘은 빠르고 효율적이며 데이터의 기밀성과 신뢰성을 보장합니다. 또한 암호화 컨텍스트로 정의된 추가 인증 데이터(AAD)를 사용하는 인증된 암호화를 지원합니다. 이 유형의 KMS 키를 사용하려면 암호화된 데이터의 발신자와 수신자가 모두 호출할 유효한 AWS 자격 증명을 가지고 있어야 합니다 AWS KMS.

사용 사례에서를 호출할 수 없는 사용자가 외부 AWS 에서 암호화해야 하는 경우 AWS KMS비대칭 KMS 키를 선택하는 것이 좋습니다. 이러한 사용자가 데이터를 암호화할 수 있도록 비대칭 KMS 키의 퍼블릭 키를 배포할 수 있습니다. 또한 해당 데이터의 암호를 해독해야 하는 애플리케이션은 AWS KMS내부에서 비대칭 KMS 키의 프라이빗 키를 사용할 수 있습니다.

메시지 서명 및 서명 확인

메시지에 서명하고 서명을 확인하려면 비대칭 KMS 키를 사용해야 합니다. RSA 키 페어, 타원 곡선(ECC) 키 페어, ML-DSA 키 페어 또는 SM2 키 페어(중국 리전만 해당)를 나타내는 키 사양을 가진 KMS 키를 사용할 수 있습니다. 선택하는 키 사양은 사용하려는 서명 알고리즘에 따라 결정됩니다. ECC 키 페어에서 지원되는 ECDSA 서명 알고리즘이 RSA 서명 알고리즘보다 권장됩니다. RSA 또는 ECC 키에서 양자 내성 키로 마이그레이션할 때는 ML-DSA 키 페어를 사용하세요. 그러나 AWS 외부에서 서명을 확인하는 사용자를 지원하기 위해서는 특정 키 사양 및 서명 알고리즘을 사용해야 할 수 있습니다.

비대칭 키 페어로 암호화

비대칭 키 페어가 있는 데이터를 암호화하려면 비대칭 KMS 키를 RSA키 사양 또는 SM2 키 사양(중국 리전만 해당)과 함께 사용해야 합니다. AWS KMS 에서 KMS 키 페어의 퍼블릭 키를 사용하여 데이터를 암호화하려면 암호화(Encrypt) 작업을 사용합니다. 또한 퍼블릭 키를 다운로드하여 외부에서 데이터를 암호화해야 하는 당사자와 공유할 수 있습니다 AWS KMS.

비대칭 KMS의 퍼블릭 키를 다운로드하면 AWS KMS외부에서 키를 사용할 수 있습니다. 그러나 더 이상 KMS 키를 보호하는 보안 제어가 적용되지 않습니다 AWS KMS. 예를 들어 AWS KMS 키 정책

또는 권한 부여를 사용하여 퍼블릭 키 사용을 제어할 수 없습니다. 또한가 AWS KMS 지원하는 암호화 알고리즘을 사용하여 키가 암호화 및 복호화에만 사용되는지 여부를 제어할 수 없습니다. 자세한 내용은 퍼블릭 키 다운로드 시 특별 고려 사항을 참조하세요.

외부의 퍼블릭 키로 암호화된 데이터를 복호화하려면 복호화 작업을 AWS KMS호출합니다. 데이터가 `SIGN_VERIFY`의 키 사용으로 KMS 키의 퍼블릭 키에서 암호화된 경우 `Decrypt` 작업이 실패합니다. 선택한 키 사양을 지원하지 AWS KMS 않는 알고리즘을 사용하여 암호화한 경우에도 실패합니다. 키 사양 및 지원되는 알고리즘에 대한 자세한 내용은 키 사양 참조 단원을 참조하세요.

이러한 오류를 방지하려면 외부에서 퍼블릭 키를 사용하는 모든 사용자가 키 구성을 저장 AWS KMS 해야 합니다. AWS KMS 콘솔과 GetPublicKey 응답은 퍼블릭 키를 공유할 때 포함해야 하는 정보를 제공합니다.

공유 보안 비밀 도출

공유 보안 암호를 도출하려면 NIST 표준 타원 곡선 또는 SM2(중국 리전만 해당) 키 구성 요소가 있는 KMS 키를 사용합니다.는 타원 곡선 퍼블릭-프라이빗 키 페어에서 공유 보안 암호를 도출하여 두 피어 간에 키 계약을 설정하기 위해 타원 곡선 암호화 Cofactor Diffie-Hellman Primitive(ECDH)를 AWS KMS 사용합니다. DeriveSharedSecret 작업이 반환하는 원시 공유 보안 암호를 사용하여 두 당사자 간에 전송되는 데이터를 암호화 및 해독할 수 있는 대칭 키를 추출하거나 원시 공유 보안 암호를 사용하여 대칭 키를 추출할 때 키 추출에 대한 NIST 권장 사항을 따르는 HMACs. AWS KMS recommds를 생성하고 확인할 수 있습니다.

HMAC 코드 생성 및 확인

해시 기반 메시지 인증 코드를 생성하고 확인하려면 HMAC 키를 사용합니다. 에서 HMAC 키를 생성할 때는 키 구성 요소를 AWS KMS AWS KMS 생성 및 보호하고 키에 올바른 MAC 알고리즘을 사용하도록 합니다. HMAC 코드는 의사 난수로 사용할 수도 있으며 대칭 서명 및 토큰화를 위한 특정 시나리오에서도 사용할 수 있습니다.

HMAC KMS 키는 대칭 키입니다. AWS KMS 콘솔에서 HMAC KMS 키를 생성할 때, `Symmetric` 키 유형을 선택합니다.

AWS 서비스와 함께 사용

AWS 와 통합된 서비스와 AWS KMS 함께 사용할 KMS 키를 생성하려면 데이터를 암호화하는 service. AWS services에 대한 설명서를 참조하세요. 대칭 암호화 KMS 키가 필요합니다.

이러한 고려 사항 외에도 키 사양이 다른 KMS 키에 대한 암호화 작업에는 가격과 요청 할당량이 다릅니다. AWS KMS 요금에 대한 자세한 내용은 AWS Key Management Service 요금을 참조하세요. 요청 할당량에 대한 자세한 내용은 요청 할당량 섹션을 참조하세요.

대칭 암호화 KMS 키 생성

이 주제에서는 키 구성 요소가 있는 단일 리전에 대한 대칭 암호화 KMS 키인 기본 KMS 키를 생성하는 방법을 설명합니다 AWS KMS. 이 KMS 키를 사용하여 AWS 서비스의 리소스를 보호할 수 있습니다.

AWS KMS 콘솔에서 CreateKey API를 사용하거나 AWS::KMS::Key CloudFormation 템플릿을 사용하여 대칭 암호화 KMS 키를 생성할 수 있습니다.

기본 키 사양인 SYMMETRIC_DEFAULT는 대칭 암호화 KMS 키의 키 사양입니다. AWS KMS 콘솔에서 대칭 키 유형과 암호화 및 복호화 키 사용을 선택하면 SYMMETRIC_DEFAULT 키 사양이 선택됩니다. CreateKey 작업에서 KeySpec 값을 지정하지 않으면 SYMMETRIC_DEFAULT가 선택됩니다. 다른 키 사양을 사용할 이유가 없다면 SYMMETRIC_DEFAULT가 좋은 선택입니다.

KMS 키에 적용되는 할당량에 대한 자세한 내용은 할당량 섹션을 참조하십시오.

AWS KMS 콘솔 사용

를 사용하여 AWS KMS keys (KMS 키) AWS Management Console 를 생성할 수 있습니다.

> **Important**
>
> 별칭, 설명 또는 태그에 기밀 또는 민감한 정보를 포함하지 마십시오. 이러한 필드는 CloudTrail 로그 및 기타 출력에 일반 텍스트로 표시될 수 있습니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 탐색 창에서 고객 관리형 키를 선택합니다.
4. 키 생성을 선택합니다.
5. 대칭 암호화 KMS 키를 생성하려면 키 유형(Key type)에 대칭(Symmetric)을 선택합니다.
6. 키 사용(Key usage)에서 암호화 및 해독(Encrypt and decrypt) 옵션이 선택됩니다.
7. 다음을 선택합니다.
8. KMS 키의 별칭을 입력합니다. 별칭은 **aws/**로 시작할 수 없습니다. **aws/** 접두사는 Amazon Web Services에서 예약하여 계정 AWS 관리형 키 에서 나타냅니다.

> **Note**
>
> 별칭을 추가, 삭제 또는 업데이트하면 KMS 키에 대한 권한을 허용하거나 거부할 수 있습니다. 자세한 내용은 에 대한 ABAC AWS KMS 및 별칭을 사용하여 KMS 키에 대한 액세스 제어 섹션을 참조하십시오.

별칭은 KMS 키를 식별하는 데 사용할 수 있는 표시 이름입니다. 보호하고자 하는 데이터의 유형 또는 KMS 키와 함께 사용할 애플리케이션을 나타내는 별칭을 선택하는 것이 좋습니다.

AWS Management Console에서 KMS 키를 생성할 때 별칭이 필요합니다. 이들은 CreateKey 작업을 사용할 때 선택 사항입니다.

9. (선택 사항) KMS 키에 대한 설명을 입력합니다.

   키 상태가 `Pending Deletion` 또는 `Pending Replica Deletion`이 아닌 한 지금 설명을 추가하거나 언제든지 설명을 업데이트할 수 있습니다. 기존 고객 관리형 키에 대한 설명을 추가, 변경 또는 삭제하려면의 KMS 키에 대한 세부 정보 페이지에서 설명을 편집 AWS Management Console 하거나 UpdateKeyDescription 작업을 사용합니다.

10. (선택 사항) 태그 키와 태그 값(선택)을 입력합니다. KMS 키에 두 개 이상의 태그를 추가하려면 태그 추가(Add tag)를 선택합니다.

> **Note**
>
> KMS 키에 태그를 지정하거나 해제하면 KMS 키에 대한 권한을 허용하거나 거부할 수 있습니다. 자세한 내용은 에 대한 ABAC AWS KMS 및 태그를 사용하여 KMS 키에 대한 액세스 제어 섹션을 참조하세요.

AWS 리소스에 태그를 추가하면가 태그별로 집계된 사용량 및 비용이 포함된 비용 할당 보고서를 AWS 생성합니다. KMS 키에 대한 액세스를 제어하는 데에도 태그를 사용할 수 있습니다. KMS 키 태그 지정에 대한 자세한 내용은 의 태그 AWS KMS 및 에 대한 ABAC AWS KMS 섹션을 참조하십시오.

11. 다음을 선택합니다.
12. KMS 키를 관리할 수 있는 IAM 사용자 및 역할을 선택합니다.

> **참고**
>
> 이 키 정책은이 KMS 키를 AWS 계정 완전히 제어합니다. 계정 관리자가 IAM 정책을 사용하여 KMS 키를 관리할 수 있는 권한을 다른 보안 주체에게 부여하도록 허용합니다. 자세한 내용은 the section called “기본 키 정책”을 참조하세요.
> IAM 모범 사례는 장기 보안 인증 정보가 있는 IAM 사용자의 사용을 장려하지 않습니다. 가능할 경우, 임시 보안 인증 정보를 제공하는 IAM 역할을 사용하세요. 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.
> AWS KMS 콘솔은 문 식별자 아래의 키 정책에 키 관리자를 추가합니다`"Allow access for Key Administrators"`. 이 문 식별자를 수정하면 문에 대해 콘솔이 업데이트 내용을 표시하는 방식에 영향을 미칠 수 있습니다.

13. (선택 사항) 선택한 IAM 사용자와 역할이 페이지 하단의 키 삭제 섹션에서 이 KMS 키를 삭제하지 못하도록 하려면 키 관리자가 이 키를 삭제하도록 허용(Allow key administrators to delete this key) 확인란의 선택을 취소합니다.
14. 다음을 선택합니다.
15. 암호화 작업에서 키를 사용할 수 있는 IAM 사용자 및 역할을 선택합니다.

> **참고**
>
> IAM 모범 사례는 장기 보안 인증 정보가 있는 IAM 사용자의 사용을 장려하지 않습니다. 가능할 경우, 임시 보안 인증 정보를 제공하는 IAM 역할을 사용하세요. 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.
> AWS KMS 콘솔은 명령문 식별자 `"Allow use of the key"` 및 아래의 키 정책에 키 사용자를 추가합니다`"Allow attachment of persistent resources"`. 이러한 문 식별자들을 수정하면 문에 대해 콘솔이 업데이트 내용을 표시하는 방식에 영향을 미칠 수 있습니다.

16. (선택 사항) 다른 사용자가 암호화 작업에이 KMS 키를 AWS 계정 사용하도록 허용할 수 있습니다. 이렇게 하려면 페이지 하단의 기타 AWS 계정 섹션에서 다른 추가 AWS 계정를 선택하고 외부 계정의 AWS 계정 식별 번호를 입력합니다. 외부 계정을 여러 개 추가하려면 이 단계를 반복합니다.

> **Note**
>
> 외부 계정의 보안 주체가 KMS 키를 사용하도록 허용하려면 외부 계정 관리자가 이러한 권한을 제공하는 IAM 정책을 생성해야 합니다. 자세한 정보는 다른 계정의 사용자가 KMS를 사용하도록 허용 섹션을 참조하세요.

17. 다음을 선택합니다.
18. 키에 대한 키 정책 문을 검토합니다. 키 정책을 변경하려면 편집을 선택하세요.
19. 다음을 선택합니다.
20. 선택한 키 설정을 검토합니다. 여전히 돌아가서 모든 설정을 변경할 수 있습니다.
21. 마침(Finish)을 선택하여 KMS 키를 생성합니다.

AWS KMS API 사용

CreateKey 작업을 사용하여 AWS KMS keys 모든 유형을 생성할 수 있습니다. 이 예제에서는 AWS Command Line Interface (AWS CLI)를 사용합니다. 다양한 프로그래밍 언어의 예는 AWS SDK 또는 CLI와 `CreateKey` 함께 사용 섹션을 참조하십시오.

> **Important**
>
> `Description` 또는 `Tags` 필드에 기밀 또는 민감한 정보를 포함하지 마십시오. 이러한 필드는 CloudTrail 로그 및 기타 출력에 일반 텍스트로 표시될 수 있습니다.

다음 작업은 AWS KMS에 의해 생성된 키 구성 요소를 기반으로 단일 리전에 대칭 암호화 키를 생성합니다. 이 작업에는 필수 파라미터가 없습니다. 그러나 `Policy` 파라미터를 사용하여 키 정책을 지정할 수도 있습니다. 언제든 키 정책(PutKeyPolicy)을 변경하고 설명 및 태그 등의 선택적 요소를 추가할 수 있습니다. 비대칭 키, 다중 리전 키, 가져온 키 구성 요소가 있는 키, 사용자 지정 키 스토어의 키를 생성할 수도 있습니다. 클라이언트 측 암호화를 위한 데이터 키를 생성하려면 GenerateDataKey 작업을 사용합니다.

`CreateKey` 작업을 사용하면 별칭을 지정할 수 없지만 CreateAlias 작업을 사용하여 새 KMS 키의 별칭을 만들 수 있습니다.

다음은 파라미터 없이 `CreateKey` 작업을 호출하는 예제입니다. 이 명령은 모든 기본값을 사용합니다. 그러면 AWS KMS가 생성한 키 구성 요소가 있는 대칭 암호화 KMS 키가 생성됩니다.

```
$ aws kms create-key
{
    "KeyMetadata": {
        "Origin": "AWS_KMS",
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "Description": "",
        "KeyManager": "CUSTOMER",
        "Enabled": true,
        "KeySpec": "SYMMETRIC_DEFAULT",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "KeyState": "Enabled",
        "CreationDate": 1502910355.475,
        "Arn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "AWSAccountId": "111122223333",
        "MultiRegion": false
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ],
    }
}
```

새 KMS 키에 대한 키 정책을 지정하지 않으면 `CreateKey`가 적용하는 기본 키 정책은 콘솔을 사용하여 새 KMS 키를 생성할 때 적용하는 기본 키 정책과 다릅니다.

예를 들어 이 GetKeyPolicy 작업 호출은 `CreateKey`가 적용하는 키 정책을 반환합니다. KMS 키에 대한 AWS 계정 액세스 권한을에 부여하고 KMS 키에 대한 AWS Identity and Access Management (IAM) 정책을 생성할 수 있습니다. IAM 정책 및 KMS 키의 키 정책에 대한 자세한 내용은 KMS 키 액세스 및 권한 섹션을 참조하십시오.

```
$ aws kms get-key-policy --key-id 1234abcd-12ab-34cd-56ef-1234567890ab --policy-name
 default --output text
```

JSON

```
{
  "Version":"2012-10-17",
  "Id" : "key-default-1",
  "Statement" : [ {
    "Sid" : "EnableIAMUserPermissions",
```

```
    "Effect" : "Allow",
    "Principal" : {
      "AWS" : "arn:aws:iam::111122223333:root"
    },
    "Action" : "kms:*",
    "Resource" : "*"
  } ]
}
```

비대칭 KMS 키 생성

AWS KMS 콘솔에서 CreateKey API를 사용하거나 AWS::KMS::Key CloudFormation 템플릿을 사용하여 비대칭 KMS 키를 생성할 수 있습니다. 비대칭 KMS 키는 암호화, 서명 또는 공유 비밀을 도출하는 데 사용할 수 있는 퍼블릭 및 프라이빗 키 페어를 나타냅니다. 프라이빗 키는 내에 유지됩니다 AWS KMS. 외부에서 사용할 퍼블릭 키를 다운로드하려면 섹션을 AWS KMS참조하세요퍼블릭 키 다운로드.

비대칭 KMS를 생성할 때 키 사양을 선택해야 합니다. 선택하는 키 사양은 규정, 보안 또는 비즈니스 요구 사항에 따라 결정됩니다. 또한 암호화하거나 서명해야 하는 메시지의 크기에 영향을 받을 수 있습니다. 일반적으로 암호화 키가 길수록 무차별 대입 공격에 더 강합니다. 지원되는 모든 키 사양에 대한 자세한 설명은 키 사양 참조 단원을 참조하세요.

AWS 와 통합되는 서비스는 비대칭 KMS 키를 지원하지 AWS KMS 않습니다. AWS 서비스에서 저장하거나 관리하는 데이터를 암호화하는 KMS 키를 생성하려면 대칭 암호화 KMS 키를 생성합니다.

KMS 키를 생성하는 데 필요한 권한에 대한 자세한 내용은 KMS 키를 생성하기 위한 사용 권한 섹션을 참조하세요.

AWS KMS 콘솔 사용

를 사용하여 비대칭 AWS KMS keys (KMS 키) AWS Management Console 을 생성할 수 있습니다. 각 비대칭 KMS 키는 퍼블릭 및 프라이빗 키 페어를 나타냅니다.

> **Important**
>
> 별칭, 설명 또는 태그에 기밀 또는 민감한 정보를 포함하지 마십시오. 이러한 필드는 CloudTrail 로그 및 기타 출력에 일반 텍스트로 표시될 수 있습니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.

2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 탐색 창에서 고객 관리형 키를 선택합니다.
4. 키 생성을 선택합니다.
5. 비대칭 KMS 키를 생성하려면 키 유형(Key type)에서 비대칭(Asymmetric)을 선택합니다.
6. 퍼블릭 키 암호화를 위한 비대칭 KMS 키를 생성하면 키 사용(Key usage)에서 암호화 및 해독(Encrypt and decrypt)를 선택합니다.

   메시지 서명 및 서명 확인을 위한 비대칭 KMS 키를 생성하려면 키 사용에서 서명 및 확인을 선택합니다.

   공유 보안 비밀을 도출하기 위한 비대칭 KMS 키를 생성하려면 키 사용에서 키 계약을 선택합니다.

   키 사용 값 선택에 대한 도움말은 생성할 KMS 키 유형 선택 섹션을 참조하세요.
7. 비대칭 KMS 키에 대한 사양(키 사양(Key spec))을 선택합니다.
8. 다음을 선택합니다.
9. KMS 키에 대한 별칭을 입력합니다. 별칭은 **aws/**로 시작할 수 없습니다. **aws/** 접두사는 Amazon Web Services에서 계정에서 AWS 관리형 키 를 나타내기 위해 예약한 것입니다.

   별칭은 콘솔 및 일부 AWS KMS APIs. 보호하고자 하는 데이터의 유형 또는 KMS 키와 함께 사용할 애플리케이션을 나타내는 별칭을 선택하는 것이 좋습니다.

   AWS Management Console에서 KMS 키를 생성할 때 별칭이 필요합니다. CreateKey 작업을 사용할 때는 별칭을 지정할 수 없지만 콘솔이나 CreateAlias 작업을 사용하여 기존 KMS 키의 별칭을 생성합니다. 자세한 내용은 의 별칭 AWS KMS 섹션을 참조하세요.
10. (선택 사항) KMS 키에 대한 설명을 입력합니다.

    보호하려는 데이터의 유형 또는 KMS 키와 함께 사용하려는 애플리케이션을 설명하는 내용을 입력합니다.

    키 상태가 `Pending Deletion` 또는 `Pending Replica Deletion`이 아닌 한 지금 설명을 추가하거나 언제든지 설명을 업데이트할 수 있습니다. 기존 고객 관리형 키에 대한 설명을 추가, 변경 또는 삭제하려면의 KMS 키에 대한 세부 정보 페이지에서 설명을 편집 AWS Management Console 하거나 UpdateKeyDescription 작업을 사용합니다.
11. (선택 사항) 태그 키와 태그 값(선택)을 입력합니다. KMS 키에 두 개 이상의 태그를 추가하려면 태그 추가(Add tag)를 선택합니다.

AWS 리소스에 태그를 추가하면가 태그별로 집계된 사용량 및 비용이 포함된 비용 할당 보고서를 AWS 생성합니다. KMS 키에 대한 액세스를 제어하는 데에도 태그를 사용할 수 있습니다. KMS 키 태그 지정에 대한 자세한 내용은 의 태그 AWS KMS 및 에 대한 ABAC AWS KMS 섹션을 참조하십시오.

12. 다음을 선택합니다.
13. KMS 키를 관리할 수 있는 IAM 사용자 및 역할을 선택합니다.

> **참고**
>
> 이 키 정책은이 KMS 키를 AWS 계정 완전히 제어합니다. 계정 관리자가 IAM 정책을 사용하여 KMS 키를 관리할 수 있는 권한을 다른 보안 주체에게 부여하도록 허용합니다. 자세한 내용은 the section called “기본 키 정책”을 참조하세요.
> IAM 모범 사례는 장기 보안 인증 정보가 있는 IAM 사용자의 사용을 장려하지 않습니다. 가능할 경우, 임시 보안 인증 정보를 제공하는 IAM 역할을 사용하세요. 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.
> AWS KMS 콘솔은 문 식별자 아래의 키 정책에 키 관리자를 추가합니다`"Allow access for Key Administrators"`. 이 문 식별자를 수정하면 문에 대해 콘솔이 업데이트 내용을 표시하는 방식에 영향을 미칠 수 있습니다.

14. (선택 사항) 선택한 IAM 사용자와 역할이 페이지 하단의 키 삭제 섹션에서 이 KMS 키를 삭제하지 못하도록 하려면 키 관리자가 이 키를 삭제하도록 허용(Allow key administrators to delete this key) 확인란의 선택을 취소합니다.
15. 다음을 선택합니다.
16. 암호화 작업에서 KMS 키를 사용할 수 있는 IAM 사용자 및 역할을 선택합니다.

> **참고**
>
> IAM 모범 사례는 장기 보안 인증 정보가 있는 IAM 사용자의 사용을 장려하지 않습니다. 가능할 경우, 임시 보안 인증 정보를 제공하는 IAM 역할을 사용하세요. 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.
> AWS KMS 콘솔은 명령문 식별자 `"Allow use of the key"` 및 아래의 키 정책에 키 사용자를 추가합니다`"Allow attachment of persistent resources"`. 이러한 문 식별자들을 수정하면 문에 대해 콘솔이 업데이트 내용을 표시하는 방식에 영향을 미칠 수 있습니다.

17. (선택 사항) 다른 사용자가 암호화 작업에이 KMS 키를 AWS 계정 사용하도록 허용할 수 있습니다. 이렇게 하려면 페이지 하단의 기타 AWS 계정 섹션에서 다른 AWS 계정추가를 선택하고 외부 계정의 AWS 계정 식별 번호를 입력합니다. 외부 계정을 여러 개 추가하려면 이 단계를 반복합니다.

> **Note**
>
> 외부 계정의 보안 주체가 KMS 키를 사용하도록 허용하려면 외부 계정 관리자가 이러한 권한을 제공하는 IAM 정책을 생성해야 합니다. 자세한 정보는 다른 계정의 사용자가 KMS를 사용하도록 허용 섹션을 참조하세요.

18. 다음을 선택합니다.
19. 키에 대한 키 정책 문을 검토합니다. 키 정책을 변경하려면 편집을 선택하세요.
20. 다음을 선택합니다.
21. 선택한 키 설정을 검토합니다. 여전히 돌아가서 모든 설정을 변경할 수 있습니다.
22. 마침(Finish)을 선택하여 KMS 키를 생성합니다.

AWS KMS API 사용

CreateKey 작업을 사용하여 비대칭을 생성할 수 있습니다 AWS KMS key. 이 예제들은 AWS Command Line Interface (AWS CLI)를 사용하지만, 사용자는 어떤 지원되는 프로그래밍 언어라도 사용할 수 있습니다.

비대칭 KMS 키를 생성할 때 `KeySpec` 파라미터를 지정해야 합니다. 이 파라미터는 생성할 키의 유형을 결정합니다. 또한 ENCRYPT_DECRYPT, SIGN_VERIFY 또는 KEY_AGREEMENT의 `KeyUsage` 값을 지정해야 합니다. KMS 키가 생성된 후에는 이러한 속성을 변경할 수 없습니다.

`CreateKey` 작업을 사용하면 별칭을 지정할 수 없지만 CreateAlias 작업을 사용하여 새 KMS 키의 별칭을 만들 수 있습니다.

> **Important**
>
> `Description` 또는 `Tags` 필드에 기밀 또는 민감한 정보를 포함하지 마십시오. 이러한 필드는 CloudTrail 로그 및 기타 출력에 일반 텍스트로 표시될 수 있습니다.

퍼블릭 암호화를 위한 비대칭 KMS 키 페어 생성

다음 예제에서는 CreateKey 작업을 사용하여 퍼블릭 키 암호화를 위해 설계된 4096비트 RSA 키의 비대칭 KMS 키를 생성합니다.

```
$ aws kms create-key --key-spec RSA_4096 --key-usage ENCRYPT_DECRYPT
{
    "KeyMetadata": {
        "KeyState": "Enabled",
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "KeyManager": "CUSTOMER",
        "Description": "",
        "Arn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "CreationDate": 1569973196.214,
        "MultiRegion": false,
        "KeySpec": "RSA_4096",
        "CustomerMasterKeySpec": "RSA_4096",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "EncryptionAlgorithms": [
            "RSAES_OAEP_SHA_1",
            "RSAES_OAEP_SHA_256"
        ],
        "AWSAccountId": "111122223333",
        "Origin": "AWS_KMS",
        "Enabled": true
    }
}
```

서명 및 확인을 위한 비대칭 KMS 키 페어 생성

다음 예제 명령은 서명 및 확인에 사용되는 ECC 키 페어를 나타내는 비대칭 KMS 키를 생성합니다. 암호화 및 암호 해독을 위한 타원 곡선 키 페어는 생성할 수 없습니다.

```
$ aws kms create-key --key-spec ECC_NIST_P521 --key-usage SIGN_VERIFY
{
    "KeyMetadata": {
        "KeyState": "Enabled",
        "KeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
        "CreationDate": 1570824817.837,
        "Origin": "AWS_KMS",
        "SigningAlgorithms": [
            "ECDSA_SHA_512"
        ],
```

```
        "Arn": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321",
        "AWSAccountId": "111122223333",
        "KeySpec": "ECC_NIST_P521",
        "CustomerMasterKeySpec": "ECC_NIST_P521",
        "KeyManager": "CUSTOMER",
        "Description": "",
        "Enabled": true,
        "MultiRegion": false,
        "KeyUsage": "SIGN_VERIFY"
    }
}
```

공유 보안 비밀을 도출하기 위한 비대칭 KMS 키 페어 생성

다음 예제 명령은 공유 비밀을 도출하는 데 사용되는 ECDSA 키 페어를 나타내는 비대칭 KMS 키를 생성합니다. 암호화 및 암호 해독을 위한 타원 곡선 키 페어는 생성할 수 없습니다.

```
$ aws kms create-key --key-spec ECC_NIST_P256 --key-usage KEY_AGREEMENT
{
    "KeyMetadata": {
        "AWSAccountId": "111122223333",
        "KeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
        "Arn": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321",
        "CreationDate": "2023-12-27T19:10:15.063000+00:00",
        "Enabled": true,
        "Description": "",
        "KeyUsage": "KEY_AGREEMENT",
        "KeyState": "Enabled",
        "Origin": "AWS_KMS",
        "KeyManager": "CUSTOMER",
        "CustomerMasterKeySpec": "ECC_NIST_P256",
        "KeySpec": "ECC_NIST_P256",
        "KeyAgreementAlgorithms": [
            "ECDH"
        ],
        "MultiRegion": false
    }
}
```

HMAC KMS 키 생성

AWS KMS 콘솔에서 API를 사용하거나 AWS::KMS::Key 템플릿을 CreateKey 사용하여 HMAC KMS 키를 생성할 수 있습니다. AWS::KMS::Key CloudFormation

HMAC KMS 키를 생성할 때 키 사양을 선택해야 합니다. HMAC KMS 키에 대한 여러 키 사양을 AWS KMS 지원합니다. 선택하는 키 사양은 규정, 보안 또는 비즈니스 요구 사항에 따라 결정될 수 있습니다. 일반적으로 키가 길수록 무차별 대입 공격에 더 강합니다.

KMS 키를 생성하는 데 필요한 권한에 대한 자세한 내용은 KMS 키를 생성하기 위한 사용 권한 섹션을 참조하세요.

AWS KMS 콘솔 사용

AWS Management Console 를 사용하여 HMAC KMS 키를 생성할 수 있습니다. HMAC KMS 키는 키 사용이 MAC 생성 및 확인(Generate and verify MAC)인 대칭 키입니다. 다중 리전 HMAC 키를 생성할 수도 있습니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 탐색 창에서 고객 관리형 키를 선택합니다.
4. 키 생성을 선택합니다.
5. 키 유형에 대칭을 선택합니다.

   HMAC KMS 키는 대칭입니다. 동일한 키를 사용하여 HMAC 태그를 생성하고 확인합니다.
6. 키 사용(Key usage)에서 MAC 생성 및 확인(Generate and verify MAC)을 선택합니다.

   MAC 생성 및 확인(Generate and verify MAC)은 HMAC KMS 키에 대한 유일한 유효한 키 사용입니다.

   > **Note**
   >
   > 선택한 리전에서 HMAC KMS 키가 지원되는 경우에만 대칭 키에 대한 키 사용(Key usage)이 표시됩니다.
7. HMAC KMS 키에 대한 사양(키 사양(Key spec))을 선택합니다.

선택하는 키 사양은 규정, 보안 또는 비즈니스 요구 사항에 따라 결정될 수 있습니다. 일반적으로 더 긴 키는 더 안전합니다.

8. 다중 리전 기본 HMAC 키를 생성하려면, 고급 옵션(Advanced options)에서 다중 리전 키(Multi-Region key)를 선택합니다. 키 유형, 키 사용 등과 같이 이 KMS 키에 대해 정의한 공유 속성은 해당 복제본 키와 공유됩니다.

   이 절차를 사용하여 복제본 키를 생성할 수 없습니다. 다중 리전 복제본 HMAC 키를 생성하려면 복제본 키 생성 지침을 따릅니다.

9. 다음을 선택합니다.

10. KMS 키에 대한 별칭을 입력합니다. 별칭은 **aws/**로 시작할 수 없습니다. **aws/** 접두사는 Amazon Web Services에서 계정에서 AWS 관리형 키 를 나타내기 위해 예약한 것입니다.

    `HMAC/test-key`와 같이 KMS 키를 HMAC 키로 식별하는 별칭을 사용하는 것이 좋습니다. 이렇게 하면 AWS KMS 콘솔에서 HMAC 키를 더 쉽게 식별할 수 있습니다. 여기서 키 사양이나 키 사용량이 아닌 태그 및 별칭별로 키를 정렬하고 필터링할 수 있습니다.

    AWS Management Console에서 KMS 키를 생성할 때 별칭이 필요합니다. CreateKey 작업을 사용할 때는 별칭을 지정할 수 없지만 콘솔이나 CreateAlias 작업을 사용하여 기존 KMS 키의 별칭을 생성합니다. 자세한 내용은 의 별칭 AWS KMS 섹션을 참조하세요.

11. (선택 사항) KMS 키에 대한 설명을 입력합니다.

    보호하려는 데이터의 유형 또는 KMS 키와 함께 사용하려는 애플리케이션을 설명하는 내용을 입력합니다.

    키 상태가 `Pending Deletion` 또는 `Pending Replica Deletion`이 아닌 한 지금 설명을 추가하거나 언제든지 설명을 업데이트할 수 있습니다. 기존 고객 관리형 키에 대한 설명을 추가, 변경 또는 삭제하려면의에서 KMS 키에 대한 세부 정보 페이지의 설명을 편집 AWS Management Console AWS Management Console 하거나 UpdateKeyDescription 작업을 사용합니다.

12. (선택 사항) 태그 키와 선택적 태그 값을 입력합니다. KMS 키에 두 개 이상의 태그를 추가하려면 태그 추가(Add tag)를 선택합니다.

    `Type=HMAC`와 같이 키를 HMAC 키로 식별하는 태그를 추가하는 것이 좋습니다. 이렇게 하면 AWS KMS 콘솔에서 HMAC 키를 더 쉽게 식별할 수 있습니다. 여기서 키 사양이나 키 사용량이 아닌 태그 및 별칭별로 키를 정렬하고 필터링할 수 있습니다.

    AWS 리소스에 태그를 추가하면는 태그별로 집계된 사용량 및 비용이 포함된 비용 할당 보고서를 AWS 생성합니다. KMS 키에 대한 액세스를 제어하는 데에도 태그를 사용할 수 있습니다. KMS 키

태그 지정에 대한 자세한 내용은 의 태그 AWS KMS 및 에 대한 ABAC AWS KMS 섹션을 참조하십시오.

13. 다음을 선택합니다.
14. KMS 키를 관리할 수 있는 IAM 사용자 및 역할을 선택합니다.

> **참고**
>
> 이 키 정책은이 KMS 키를 AWS 계정 완전히 제어합니다. 계정 관리자가 IAM 정책을 사용하여 KMS 키를 관리할 수 있는 권한을 다른 보안 주체에게 부여하도록 허용합니다. 자세한 내용은 the section called “기본 키 정책”을 참조하세요.
> IAM 모범 사례는 장기 보안 인증 정보가 있는 IAM 사용자의 사용을 장려하지 않습니다. 가능할 경우, 임시 보안 인증 정보를 제공하는 IAM 역할을 사용하세요. 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.
> AWS KMS 콘솔은 문 식별자 아래의 키 정책에 키 관리자를 추가합니다`"Allow access for Key Administrators"`. 이 문 식별자를 수정하면 문에 대해 콘솔이 업데이트 내용을 표시하는 방식에 영향을 미칠 수 있습니다.

15. (선택 사항) 선택한 IAM 사용자와 역할이 페이지 하단의 키 삭제 섹션에서 이 KMS 키를 삭제하지 못하도록 하려면 키 관리자가 이 키를 삭제하도록 허용(Allow key administrators to delete this key) 확인란의 선택을 취소합니다.
16. 다음을 선택합니다.
17. 암호화 작업에서 KMS 키를 사용할 수 있는 IAM 사용자 및 역할을 선택합니다.

> **참고**
>
> IAM 모범 사례는 장기 보안 인증 정보가 있는 IAM 사용자의 사용을 장려하지 않습니다. 가능할 경우, 임시 보안 인증 정보를 제공하는 IAM 역할을 사용하세요. 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.
> AWS KMS 콘솔은 문 식별자 `"Allow use of the key"` 및 아래의 키 정책에 키 사용자를 추가합니다`"Allow attachment of persistent resources"`. 이러한 문 식별자들을 수정하면 문에 대해 콘솔이 업데이트 내용을 표시하는 방식에 영향을 미칠 수 있습니다.

18. (선택 사항) 다른 사용자가 암호화 작업에이 KMS 키를 AWS 계정 사용하도록 허용할 수 있습니다. 이렇게 하려면 페이지 하단의 기타 AWS 계정 섹션에서 다른 AWS 계정추가를 선택하고 외부 계정의 AWS 계정 식별 번호를 입력합니다. 외부 계정을 여러 개 추가하려면 이 단계를 반복합니다.

> **Note**
>
> 외부 계정의 보안 주체가 KMS 키를 사용하도록 허용하려면 외부 계정 관리자가 이러한 권한을 제공하는 IAM 정책을 생성해야 합니다. 자세한 정보는 다른 계정의 사용자가 KMS를 사용하도록 허용 섹션을 참조하세요.

19. 다음을 선택합니다.
20. 키에 대한 키 정책 문을 검토합니다. 키 정책을 변경하려면 편집을 선택하세요.
21. 다음을 선택합니다.
22. 선택한 키 설정을 검토합니다. 여전히 돌아가서 모든 설정을 변경할 수 있습니다.
23. 마침(Finish)을 선택하여 HMAC KMS 키를 생성합니다.

AWS KMS API 사용

CreateKey 작업을 사용하여 HMAC KMS 키를 생성할 수 있습니다. 이 예제들은 AWS Command Line Interface (AWS CLI)를 사용하지만, 사용자는 어떤 지원되는 프로그래밍 언어라도 사용할 수 있습니다.

HMAC KMS 키를 생성할 때 `KeySpec` 파라미터를 지정해야 합니다. 이 파라미터는 KMS 키의 유형을 결정합니다. 또한 HMAC 키의 유일하게 유효한 키 사용 값인 경우에도 `KeyUsage` 값을 GENERATE_VERIFY_MAC로 지정해야 합니다. 다중 리전 HMAC KMS 키를 생성하려면, 값이 `true`인 `MultiRegion` 파라미터를 추가합니다. KMS 키가 생성된 후에는 이러한 속성을 변경할 수 없습니다.

`CreateKey` 작업을 사용하면 별칭을 지정할 수 없지만 CreateAlias 작업을 사용하여 새 KMS 키의 별칭을 만들 수 있습니다. `HMAC/test-key`와 같이 KMS 키를 HMAC 키로 식별하는 별칭을 사용하는 것이 좋습니다. 이렇게 하면 AWS KMS 콘솔에서 HMAC 키를 더 쉽게 식별할 수 있습니다. 여기서 키 사양이나 키 사용량이 아닌 별칭을 기준으로 키를 정렬하고 필터링할 수 있습니다.

HMAC 키가 지원되지 않는 AWS 리전 에서 HMAC KMS 키를 생성하려고 하면 `CreateKey` 작업이를 반환합니다. `UnsupportedOperationException`

다음 예제에서는 `CreateKey` 작업을 사용하여 512비트 HMAC KMS 키를 생성합니다.

```
$ aws kms create-key --key-spec HMAC_512 --key-usage GENERATE_VERIFY_MAC
{
    "KeyMetadata": {
```

```
        "KeyState": "Enabled",
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "KeyManager": "CUSTOMER",
        "Description": "",
        "Arn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "CreationDate": 1669973196.214,
        "MultiRegion": false,
        "KeySpec": "HMAC_512",
        "CustomerMasterKeySpec": "HMAC_512",
        "KeyUsage": "GENERATE_VERIFY_MAC",
        "MacAlgorithms": [
            "HMAC_SHA_512"
        ],
        "AWSAccountId": "111122223333",
        "Origin": "AWS_KMS",
        "Enabled": true
    }
}
```


다중 리전 프라이머리 키 생성

AWS KMS 콘솔에서 또는 AWS KMS API를 사용하여 다중 리전 기본 키를 생성할 수 있습니다. 가 다중 리전 키를 AWS KMS 지원하는 모든 AWS 리전 에서 기본 키를 생성할 수 있습니다.

다중 리전 기본 키를 생성하려면 보안 주체에게 IAM 정책의 kms:CreateKey 권한을 포함하여 KMS 키를 생성하는 데 필요한 것과 동일한 권한이 필요합니다. 보안 주체는 또한iam:CreateServiceLinkedRole 권한도 필요합니다. kms:MultiRegionKeyType 조건 키를 사용하여 다중 리전 기본 키 생성 권한을 허용하거나 거부할 수 있습니다.

> **Note**
>
> 다중 리전 프라이머리 키를 생성할 때, 키를 관리하고 사용하는 IAM 사용자와 역할을 신중하게 검토하여 선택해야 합니다. IAM 정책은 다른 IAM 사용자 및 역할에 KMS 키 관리 권한을 제공할 수 있습니다.
>
> IAM 모범 사례는 장기 보안 인증 정보가 있는 IAM 사용자의 사용을 장려하지 않습니다. 가능할 경우, 임시 보안 인증 정보를 제공하는 IAM 역할을 사용하세요. 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.

AWS KMS 콘솔 사용

AWS KMS 콘솔에서 다중 리전 기본 키를 생성하려면 KMS 키를 생성하는 데 사용하는 것과 동일한 프로세스를 사용합니다. 고급 옵션에서 다중 리전 키를 선택합니다. 전체 지침은 KMS 키 생성 섹션을 참조하십시오.

> **Important**
>
> 별칭, 설명 또는 태그에 기밀 또는 민감한 정보를 포함하지 마십시오. 이러한 필드는 CloudTrail 로그 및 기타 출력에 일반 텍스트로 표시될 수 있습니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 탐색 창에서 고객 관리형 키를 선택합니다.
4. 키 생성(Create key)을 선택합니다.
5. 대칭 또는 비대칭 키 유형을 선택합니다. 대칭 키가 기본값입니다.

   대칭인 다중 리전 HMAC KMS 키를 포함하여 다중 리전 대칭 및 비대칭 키를 생성할 수 있습니다.
6. 키 사용을 선택합니다. 암호화 및 해독(Encrypt and decrypt)이 기본값입니다.

   도움말은 KMS 키 생성, the section called “비대칭 KMS 키 생성” 또는 the section called “HMAC KMS 키 생성” 섹션을 참조하세요.
7. 고급 옵션을 확장합니다.
8. 키 구성 요소 오리진에서가 기본 키와 복제본 키가 공유할 키 구성 요소를 AWS KMS 생성하도록 하려면 KMS를 선택합니다. 기본 및 복제본 키로 키 구성 요소를 가져오는 경우 외부(키 구성 요소 가져오기)(External (Import key material))를 선택합니다.
9. 리전 특성에서 다중 리전 키를 선택합니다.

   KMS 키를 생성한 후에는 이 설정을 변경할 수 없습니다.
10. 기본 키의 별칭을 입력합니다.

    별칭은 다중 리전 키의 공유 속성이 아닙니다. 다중 리전 기본 키와 해당 복제본에 동일한 별칭 또는 다른 별칭을 부여할 수 있습니다.는 다중 리전 키의 별칭을 동기화하지 AWS KMS 않습니다.

> **Note**
>
> 별칭을 추가, 삭제 또는 업데이트하면 KMS 키에 대한 권한을 허용하거나 거부할 수 있습니다. 자세한 내용은 에 대한 ABAC AWS KMS 및 별칭을 사용하여 KMS 키에 대한 액세스 제어 섹션을 참조하십시오.

11. (선택 사항) 기본 키에 대한 설명을 입력합니다.

    설명은 다중 리전 키의 공유 속성이 아닙니다. 다중 리전 기본 키와 해당 복제본에 동일한 설명 또는 다른 설명을 제공할 수 있습니다.는 다중 리전 키의 키 설명을 동기화하지 AWS KMS 않습니다.

12. (선택 사항) 태그 키와 태그 값(선택)을 입력합니다. 기본 키에 두 개 이상의 태그를 할당하려면 태그 추가(Add tag)를 선택합니다.

    태그는 다중 리전 키의 공유 속성이 아닙니다. 다중 리전 기본 키와 해당 복제본에 동일한 태그 또는 다른 태그를 지정할 수 있습니다. AWS KMS 는 다중 리전 키의 태그를 동기화하지 않습니다. KMS 키의 태그는 언제든지 변경할 수 있습니다.

> **Note**
>
> KMS 키에 태그를 지정하거나 해제하면 KMS 키에 대한 권한을 허용하거나 거부할 수 있습니다. 자세한 내용은 에 대한 ABAC AWS KMS 및 태그를 사용하여 KMS 키에 대한 액세스 제어 섹션을 참조하세요.

13. 기본 키를 관리할 수 있는 IAM 사용자 및 역할을 선택합니다.

> **참고**
>
> - 이 단계는 기본 키에 대한 키 정책을 만드는 프로세스를 시작합니다. 키 정책은 다중 리전 키의 공유 속성이 아닙니다. 다중 리전 기본 키와 해당 복제본에 동일한 키 정책 또는 다른 키 정책을 부여할 수 있습니다.는 다중 리전 키의 키 정책을 동기화하지 AWS KMS 않습니다. KMS 키의 키 정책은 언제든지 변경할 수 있습니다.
> - 다중 리전 프라이머리 키를 생성할 때는 콘솔에서 생성된 기본 키 정책을 사용하는 것이 좋습니다. 이 정책을 수정하는 경우, 콘솔은 복제 키 생성 시 키 관리자 및 사용자를 선택하는 단계를 제공하지 않으며 해당 정책 문도 추가하지 않습니다. 따라서 이러한 항목은 수동으로 추가해야 합니다.

- AWS KMS 콘솔은 문 식별자 아래의 키 정책에 키 관리자를 추가합니다`"Allow access for Key Administrators"`. 이 문 식별자를 수정하면 문에 대해 콘솔이 업데이트 내용을 표시하는 방식에 영향을 미칠 수 있습니다.

14. (선택 사항) 선택한 IAM 사용자와 역할이 페이지 하단의 키 삭제 섹션에서 이 KMS 키를 삭제하지 못하도록 하려면 키 관리자가 이 키를 삭제하도록 허용(Allow key administrators to delete this key) 확인란의 선택을 취소합니다.
15. 다음을 선택합니다.
16. 암호화 작업에서 KMS 키를 사용할 수 있는 IAM 사용자 및 역할을 선택합니다.

> **참고**
>
> AWS KMS 콘솔은 문 식별자 `"Allow use of the key"` 및 아래의 키 정책에 키 사용자를 추가합니다`"Allow attachment of persistent resources"`. 이러한 문 식별자들을 수정하면 문에 대해 콘솔이 업데이트 내용을 표시하는 방식에 영향을 미칠 수 있습니다.

17. (선택 사항) 다른 사용자가 암호화 작업에이 KMS 키를 AWS 계정 사용하도록 허용할 수 있습니다. 이렇게 하려면 페이지 하단의 기타 AWS 계정 섹션에서 다른 AWS 계정추가를 선택하고 외부 계정의 AWS 계정 식별 번호를 입력합니다. 외부 계정을 여러 개 추가하려면 이 단계를 반복합니다.

> **Note**
>
> 외부 계정의 보안 주체가 KMS 키를 사용하도록 허용하려면 외부 계정 관리자가 이러한 권한을 제공하는 IAM 정책을 생성해야 합니다. 자세한 정보는 다른 계정의 사용자가 KMS를 사용하도록 허용 섹션을 참조하세요.

18. 다음을 선택합니다.
19. 키에 대한 키 정책 문을 검토합니다. 키 정책을 변경하려면 편집을 선택하세요.
20. 다음을 선택합니다.
21. 선택한 키 설정을 검토합니다. 여전히 돌아가서 모든 설정을 변경할 수 있습니다.
22. 마침을 선택하여 다중 리전 프라이머리 키를 생성하세요.

AWS KMS API 사용

다중 리전 기본 키를 만들려면 CreateKey 작업을 수행합니다. 값이 `True`인 `MultiRegion` 매개변수를 사용합니다.

예를 들어 다음 명령은 호출자의 AWS 리전 (us-east-1)에 다중 리전 기본 키를 생성합니다. 키 정책을 포함하여 다른 모든 속성을 위해 기본값을 허용합니다. 다중 리전 기본 키의 기본값은 기본 키 정책을 포함하여 다른 모든 KMS 키의 기본값과 동일합니다. 이 절차에서는 기본 KMS 키인 대칭 암호화 키를 생성합니다.

응답에는 복제본 키가 없는 다중 리전 기본 키에 대한 값과 일반적인 하위 요소가 있는 `MultiRegion` 요소 및 `MultiRegionConfiguration` 요소가 포함됩니다. 다중 리전 키의 키 ID는 항상 `mrk-`로 시작합니다.

> **Important**
>
> `Description` 또는 `Tags` 필드에 기밀 또는 민감한 정보를 포함하지 마십시오. 이러한 필드는 CloudTrail 로그 및 기타 출력에 일반 텍스트로 표시될 수 있습니다.

```
$ aws kms create-key --multi-region
{
    "KeyMetadata": {
        "Origin": "AWS_KMS",
        "KeyId": "mrk-1234abcd12ab34cd56ef1234567890ab",
        "Description": "",
        "KeyManager": "CUSTOMER",
        "Enabled": true,
        "KeySpec": "SYMMETRIC_DEFAULT",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "KeyState": "Enabled",
        "CreationDate": 1606329032.475,
        "Arn": "arn:aws:kms:us-east-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
        "AWSAccountId": "111122223333",
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ],
        "MultiRegion": true,
        "MultiRegionConfiguration": {
```

```
            "MultiRegionKeyType": "PRIMARY",
            "PrimaryKey": {
                "Arn": "arn:aws:kms:us-east-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
                "Region": "us-east-1"
            },
            "ReplicaKeys": [ ]
        }
    }
}
```

다중 리전 복제본 키 생성

AWS KMS 콘솔에서 ReplicateKey 작업을 사용하거나 AWS::KMS::ReplicaKey 템플릿을 사용하여 다중 리전 복제본 키를 생성할 수 있습니다. ReplicateKey AWS::KMS::ReplicaKey CloudFormation CreateKey 작업을 사용하여 복제본 키를 생성할 수 없습니다.

다음 절차를 사용하여 대칭 암호화 KMS 키, 비대칭 KMS 키 또는 HMAC KMS 키를 포함한 다중 리전 기본 키를 복제할 수 있습니다.

이 작업이 완료되면 새 복제본 키의 임시 키 상태가 `Creating`입니다. 이 키 상태는 새로운 복제본 키 생성 프로세스가 완료되면 몇 초 후에 `Enabled`로 변경됩니다(또는 가져온 키 구성 요소로 다중 리전 키를 생성하는 경우 `PendingImport`로 변경). 키 상태가 `Creating`인 동안에는 키를 관리할 수 있지만 아직 암호화 작업에 사용할 수는 없습니다. 복제 키를 프로그래밍 방식으로 만들고 사용하는 경우 `KMSInvalidStateException`에서 다시 시도하거나 DescribeKey를 호출하여 사용하기 전에 `KeyState` 값을 확인하십시오.

실수로 복제본 키를 삭제한 경우 이 절차를 사용하여 복제본을 다시 만들 수 있습니다. 동일한 프라이머리 키를 동일한 리전에 복제할 경우 생성하는 새 복제본 키가 원래 복제본 키와 동일한 공유 속성을 갖습니다.

> **Important**
>
> 별칭, 설명 또는 태그에 기밀 또는 민감한 정보를 포함하지 마십시오. 이러한 필드는 CloudTrail 로그 및 기타 출력에 일반 텍스트로 표시될 수 있습니다.

AWS CloudFormation 템플릿을 사용하여 복제본 키를 생성하려면 AWS CloudFormation 사용 설명서의 AWS::KMS::ReplicaKey를 참조하세요.

1단계: 복제본 리전 선택

일반적으로 비즈니스 모델 및 규제 요구 사항에 AWS 리전 따라 다중 리전 키를에 복제하도록 선택합니다. 예를 들어 리소스를 보관하는 리전에 키를 복제할 수 있습니다. 또는 재해 복구 요구 사항을 준수하기 위해 키를 지리적으로 먼 리전으로 복제할 수 있습니다.

다음은 복제본 리전에 대한 AWS KMS 요구 사항입니다. 선택한 리전이 이러한 요구 사항을 준수하지 않는 경우 키를 복제하려는 시도가 실패합니다.

- 리전당 관련 다중 리전 키 1개 — 기본 키와 동일한 리전에서 또는 기본 키의 다른 복제본과 동일한 리전에서 복제본 키를 만들 수 없습니다.

  해당 프라이머리 키의 복제본이 이미 있는 리전에서 프라이머리 키를 복제하려고 하면 시도가 실패합니다. 리전의 현재 복제본 키가 `PendingDeletion` 키 상태인 경우 복제본 키 삭제를 취소하거나 복제본 키가 삭제될 때까지 기다릴 수 있습니다.
- 동일한 리전에 있는 여러 개의 관련되지 않은 다중 리전 키 — 동일한 리전에 여러 개의 관련되지 않은 다중 리전 키를 가질 수 있습니다. 예를 들어 `us-east-1` 리전에 두 개의 다중 리전 기본 키가 있을 수 있습니다. 각 기본 키는 `us-west-2` 리전에 복제본 키를 가질 수 있습니다.
- 동일한 파티션의 리전 — 복제본 키 리전은 기본 키 리전과 동일한 AWS 파티션에 있어야 합니다.
- 리전을 활성화해야 함 — 지역이 기본적으로 비활성화되어 있는 경우 AWS 계정에 대해 활성화될 때까지 해당 리전에 리소스를 생성할 수 없습니다.

2단계: 복제본 키 생성

> **Note**
>
> 복제 키를 생성할 때도, 복제 키를 관리하고 사용하는 IAM 사용자와 역할 선택을 신중하게 검토해야 합니다. IAM 정책은 다른 IAM 사용자 및 역할에 KMS 키 관리 권한을 제공할 수 있습니다.
>
> IAM 모범 사례는 장기 보안 인증 정보가 있는 IAM 사용자의 사용을 장려하지 않습니다. 가능할 경우, 임시 보안 인증 정보를 제공하는 IAM 역할을 사용하세요. 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.

AWS KMS 콘솔 사용

AWS KMS 콘솔에서 동일한 작업으로 다중 리전 기본 키의 복제본을 하나 이상 생성할 수 있습니다.

이 절차는 콘솔에 표준 단일 리전 KMS 키를 만드는 것과 유사합니다. 그러나 복제 키는 기본 키를 기반으로 하기 때문에 키 사양(대칭 또는 비대칭), 키 사용 또는 키 오리진과 같은 공유 속성 값을 선택하지 않습니다.

별칭, 태그, 설명 및 키 정책을 포함하여 공유되지 않는 속성을 지정합니다. 편의상 콘솔에는 기본 키의 현재 속성 값이 표시되지만 변경할 수 있습니다. 기본 키 값을 유지하더라도는 이러한 값을 동기화된 상태로 유지하지 AWS KMS 않습니다.

> **Important**
>
> 별칭, 설명 또는 태그에 기밀 또는 민감한 정보를 포함하지 마십시오. 이러한 필드는 CloudTrail 로그 및 기타 출력에 일반 텍스트로 표시될 수 있습니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 탐색 창에서 고객 관리형 키를 선택합니다.
4. 다중 리전 기본 키의 키 ID 또는 별칭을 선택합니다. 그러면 KMS 키의 키 세부 정보 페이지가 열립니다.

   다중 리전 기본 키를 식별하려면 오른쪽 상단 모서리에 있는 도구 아이콘을 사용하여 테이블에 리전 구분(Regionality) 열을 추가합니다.
5. 리전 구분(Regionality) 탭을 선택합니다.
6. 관련 다중 리전 키 섹션에서 새 복제본 키 생성을 선택합니다.

   관련 다중 리전 키섹션에는 기본 키와 해당 복제 키의 리전이 표시됩니다. 이 표시를 사용하여 새 복제본 키의 리전을 선택할 수 있습니다.
7. 하나 이상의 AWS 리전을 선택합니다. 이 절차에서는 선택한 각 리전에 복제본 키를 만듭니다.

   메뉴에는 기본 키와 동일한 AWS 파티션의 리전만 포함됩니다. 이미 관련된 다중 리전 키가 있는 리전이 표시되지만 선택할 수는 없습니다. 메뉴의 모든 리전에 키를 복제할 권한이 없을 수 있습니다.

   리전 선택을 마치면 메뉴를 닫습니다. 선택한 리전이 표시됩니다. 리전으로의 복제를 취소하려면 리전 이름 옆에 있는 X를 선택합니다.
8. 복제본 키에 대한 별칭을 입력합니다.

콘솔에 기본 키의 현재 별칭 중 하나가 표시되지만 변경할 수 있습니다. 다중 리전 기본 키와 해당 복제본에 동일한 별칭 또는 다른 별칭을 지정할 수 있습니다. 별칭은 다중 리전 키의 공유 속성이 아닙니다.는 다중 리전 키의 별칭을 동기화하지 AWS KMS 않습니다.

별칭을 추가, 삭제 또는 업데이트하면 KMS 키에 대한 권한을 허용하거나 거부할 수 있습니다. 자세한 내용은 에 대한 ABAC AWS KMS 및 별칭을 사용하여 KMS 키에 대한 액세스 제어 섹션을 참조하십시오.

9. (선택 사항) 복제본 키에 대한 설명을 입력합니다.

   콘솔에 기본 키의 현재 설명이 표시되지만 변경할 수 있습니다. 설명은 다중 리전 키의 공유 속성이 아닙니다. 다중 리전 기본 키와 해당 복제본에 동일한 설명 또는 다른 설명을 제공할 수 있습니다.는 다중 리전 키의 키 설명을 동기화하지 AWS KMS 않습니다.

10. (선택 사항) 태그 키와 태그 값(선택)을 입력합니다. 복제본 키에 두 개 이상의 태그를 할당하려면 태그 추가를 선택합니다.

    콘솔에 기본 키에 현재 연결된 태그가 표시되지만 변경할 수 있습니다. 태그는 다중 리전 키의 공유 속성이 아닙니다. 다중 리전 기본 키와 해당 복제본에 동일한 태그 또는 다른 태그를 지정할 수 있습니다.는 다중 리전 키의 태그를 동기화하지 AWS KMS 않습니다.

    KMS 키에 태그를 지정하거나 해제하면 KMS 키에 대한 권한을 허용하거나 거부할 수 있습니다. 자세한 내용은 에 대한 ABAC AWS KMS 및 태그를 사용하여 KMS 키에 대한 액세스 제어 섹션을 참조하세요.

11. 복제본 키를 관리할 수 있는 IAM 사용자 및 역할을 선택합니다.

> **참고**
>
> - 다중 리전 프라이머리 키를 생성할 때 프라이머리 키 정책을 수정한 경우, 콘솔은 복제 키 생성 과정에서 키 관리자 또는 키 사용자(11~15단계)를 선택하라는 프롬프트를 표시하지 않습니다. 이 경우에는 키 정책 편집 단계(17단계)에서 편집을 선택해 키 관리자와 사용자에게 필요한 권한을 키 정책에 수동으로 추가해야 합니다.
> - 이 단계는 복제본 키에 대한 키 정책을 만드는 프로세스를 시작합니다. 콘솔에 기본 키의 현재 키 정책이 표시되지만 변경할 수 있습니다. 키 정책은 다중 리전 키의 공유 속성이 아닙니다. 다중 리전 기본 키와 해당 복제본에 동일한 키 정책 또는 다른 키 정책을 지정할 수 있습니다. AWS KMS 는 키 정책을 동기화하지 않습니다. 언제든지 KMS 키의 키 정책을 변경할 수 있습니다.

- AWS KMS 콘솔은 문 식별자 아래의 키 정책에 키 관리자를 추가합니다`"Allow access for Key Administrators"`. 이 문 식별자를 수정하면 문에 대해 콘솔이 업데이트 내용을 표시하는 방식에 영향을 미칠 수 있습니다.

12. (선택 사항) 선택한 IAM 사용자와 역할이 페이지 하단의 키 삭제 섹션에서 이 KMS 키를 삭제하지 못하도록 하려면 키 관리자가 이 키를 삭제하도록 허용(Allow key administrators to delete this key) 확인란의 선택을 취소합니다.
13. 다음을 선택합니다.
14. 암호화 작업에서 KMS 키를 사용할 수 있는 IAM 사용자 및 역할을 선택합니다.

> **Note**
>
> AWS KMS 콘솔은 명령문 식별자 `"Allow use of the key"` 및 아래의 키 정책에 키 사용자를 추가합니다`"Allow attachment of persistent resources"`. 이러한 문 식별자들을 수정하면 문에 대해 콘솔이 업데이트 내용을 표시하는 방식에 영향을 미칠 수 있습니다.

15. (선택 사항) 다른 사용자가 암호화 작업에이 KMS 키를 AWS 계정 사용하도록 허용할 수 있습니다. 이렇게 하려면 페이지 하단의 기타 AWS 계정 섹션에서 다른 AWS 계정추가를 선택하고 외부 계정의 AWS 계정 식별 번호를 입력합니다. 외부 계정을 여러 개 추가하려면 이 단계를 반복합니다.

> **Note**
>
> 외부 계정의 보안 주체가 KMS 키를 사용하도록 허용하려면 외부 계정 관리자가 이러한 권한을 제공하는 IAM 정책을 생성해야 합니다. 자세한 정보는 다른 계정의 사용자가 KMS를 사용하도록 허용 섹션을 참조하세요.

16. 다음을 선택합니다.
17. 키에 대한 키 정책 문을 검토합니다. 키 정책을 변경하려면 편집을 선택하세요.
18. 다음을 선택합니다.
19. 선택한 키 설정을 검토합니다. 여전히 돌아가서 모든 설정을 변경할 수 있습니다.
20. 마침을 선택하여 다중 리전 복제본 키를 생성하세요.

AWS KMS API 사용

다중 리전 복제본 키를 생성하려면 ReplicateKey 작업을 수행합니다. CreateKey 작업을 사용하여 복제본 키를 생성할 수 없습니다. 이 작업은 복제본 키를 한 번에 하나씩 생성합니다. 지정하는 리전은 복제본 키에 대한 리전 요구 사항을 준수해야 합니다.

ReplicateKey 작업을 사용할 때 다중 리전 키의 공유 속성에 대한 값을 지정하지 않습니다. 공유 속성 값은 기본 키에서 복사되고 동기화된 상태로 유지됩니다. 그러나 공유되지 않는 속성의 값을 지정할 수 있습니다. 그렇지 않으면는 기본 키의 값이 아닌 KMS 키의 표준 기본값을 AWS KMS 적용합니다.

> **Note**
>
> Description, KeyPolicy또는 Tags 파라미터에 값을 지정하지 않으면는 빈 문자열 설명, 기본 키 정책 및 태그가 없는 복제본 키를 AWS KMS 생성합니다.
> Description 또는 Tags 필드에 기밀 또는 민감한 정보를 포함하지 마십시오. 이러한 필드는 CloudTrail 로그 및 기타 출력에 일반 텍스트로 표시될 수 있습니다.

예를 들어 다음 명령은 아시아 태평양(시드니) 리전(ap-southeast-2)에 다중 리전 복제본 키를 만듭니다. 이 복제본 키는 KeyId 파라미터 값으로 식별되는 미국 동부(버지니아 북부) 리전(us-east-1)의 기본 키를 기반으로 모델링됩니다. 이 예에서는 키 정책을 포함하여 다른 모든 속성을 위해 기본값을 허용합니다.

응답은 새 복제 키를 설명합니다. 여기에는 공유 속성에 대한 필드(예: KeyId, KeySpec, KeyUsage 및 키 구성 요소 오리진(Origin))이 포함되어 있습니다. 여기에는 Description, 키 정책(ReplicaKeyPolicy) 및 태그(ReplicaTags)와 같이 기본 키와 무관한 속성도 포함됩니다.

응답에는 ap-southeast-2 리전에서 방금 생성된 키를 포함하여 기본 키의 키 ARN 및 리전 및 모든 복제본 키가 포함됩니다. 이 예에서 ReplicaKey 요소는 이 기본 키가 이미 유럽(아일랜드) 리전(eu-west-1)에 복제되었음을 보여줍니다.

```
$ aws kms replicate-key \
    --key-id arn:aws:kms:us-east-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab \
    --replica-region ap-southeast-2
{
    "ReplicaKeyMetadata": {
        "MultiRegion": true,
        "MultiRegionConfiguration": {
            "MultiRegionKeyType": "REPLICA",
            "PrimaryKey": {
```

```
                "Arn": "arn:aws:kms:us-east-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
                "Region": "us-east-1"
            },
            "ReplicaKeys": [
                {
                    "Arn": "arn:aws:kms:ap-southeast-2:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
                    "Region": "ap-southeast-2"
                },
                {
                    "Arn": "arn:aws:kms:eu-west-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
                    "Region": "eu-west-1"
                }
            ]
        },
        "AWSAccountId": "111122223333",
        "Arn": "arn:aws:kms:ap-southeast-2:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
        "CreationDate": 1607472987.918,
        "Description": "",
        "Enabled": true,
        "KeyId": "mrk-1234abcd12ab34cd56ef1234567890ab",
        "KeyManager": "CUSTOMER",
        "KeySpec": "SYMMETRIC_DEFAULT",
        "KeyState": "Enabled",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "Origin": "AWS_KMS",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ]
    },
    "ReplicaKeyPolicy": "{\n  \"Version\" : \"2012-10-17\",\n  \"Id\" : \"key-
default-1\",...,
    "ReplicaTags": []
}
```

가져온 키 구성 요소가 있는 KMS 키 생성

가져온 키 구성 요소를 사용하면 생성한 암호화 키로 AWS 리소스를 보호할 수 있습니다. 다음 개요에서는 AWS KMS로 키 구성 요소를 가져오는 방법을 설명합니다. 프로세스의 각 단계에 대한 자세한 정보는 해당 주제를 참조하세요.

1. 키 구성 요소 없이 KMS 키 생성 - 오리진은 EXTERNAL이어야 합니다. 의 키 오리진은 키가 가져온 키 구성 요소를 위해 설계되었음을 EXTERNAL 나타내며 AWS KMS 가 KMS 키에 대한 키 구성 요소를 생성하지 못하도록 합니다. 이후 단계에 이 KMS 키로 고유한 키 구성 요소를 가져오게 됩니다.

   가져오는 키 구성 요소는 연결된 키의 AWS KMS 키 사양과 호환되어야 합니다. 호환성에 대한 자세한 내용은 the section called “가져온 키 구성 요소에 대한 요구 사항” 페이지를 참조하세요.
2. 래핑 퍼블릭 키 및 가져오기 토큰 다운로드 – 1단계를 완료한 후 래핑 퍼블릭 키와 가져오기 토큰을 다운로드합니다. 이러한 항목은 키 구성 요소를 가져오는 동안 키 구성 요소를 보호합니다 AWS KMS.

   이 단계에서는 RSA 래핑 키의 유형('키 사양')과 AWS KMS로 전송되는 데이터를 암호화하는 데 사용할 래핑 알고리즘을 선택합니다. 동일한 키 구성 요소를 가져오거나 다시 가져올 때마다 다른 래핑 키 사양과 래핑 키 알고리즘을 선택할 수 있습니다.
3. 키 구성 요소 암호화 – 2단계에서 다운로드한 래핑 퍼블릭 키를 사용하여 고유한 시스템에 생성한 키 구성 요소를 암호화합니다.
4. 키 구성 요소 가져오기 – 3단계에 생성하여 암호화한 키 구성 요소와 2단계에 다운로드한 가져오기 토큰을 업로드합니다.

   이 단계에서 선택적 만료 시간을 설정할 수 있습니다. 가져온 키 구성 요소가 만료되면 AWS KMS 삭제되고 KMS 키를 사용할 수 없게 됩니다. KMS 키를 계속 사용하려면 동일한 키 구성 요소를 다시 가져와야 합니다.

   가져오기 작업이 성공적으로 완료되면 KMS 키의 키 상태가 `PendingImport`에서 `Enabled`로 변경됩니다. 이제 암호화 작업에 KMS 키를 사용할 수 있습니다.

AWS KMS 는 KMS 키를 생성하고, 래핑 퍼블릭 키 및 가져오기 토큰을 다운로드하고, 키 구성 요소를 가져올 때 AWS CloudTrail 로그에 항목을 기록합니다. AWS KMS 또한는 가져온 키 구성 요소를 삭제하거나가 AWS KMS 만료된 키 구성 요소를 삭제할 때 항목을 기록합니다.

키 구성 요소 가져오기 권한

가져온 키 구성 요소가 있는 KMS 키를 만들고 관리하려면 사용자에게 이 프로세스의 작업에 대한 권한이 필요합니다. KMS 키를 만들 때 키 정책에서 `kms:GetParametersForImport`, `kms:ImportKeyMaterial` 및 `kms:DeleteImportedKeyMaterial` 권한을 제공할 수 있습니다. AWS KMS 콘솔에서 이러한 권한은 외부 키 구성 요소 오리진으로 키를 생성할 때 키 관리자에 대해 자동으로 추가됩니다.

가져온 키 구성 요소가 있는 KMS 키를 만들려면 보안 주체에 다음 권한이 필요합니다.

- kms:CreateKey(IAM 정책)
  - 가져온 키 구성 요소가 있는 KMS 키로 이 권한을 제한하려면 값이 `EXTERNAL`인 kms:KeyOrigin 정책 조건을 사용하십시오.

```
{
  "Sid": "CreateKMSKeysWithoutKeyMaterial",
  "Effect": "Allow",
  "Resource": "*",
  "Action": "kms:CreateKey",
  "Condition": {
    "StringEquals": {
      "kms:KeyOrigin": "EXTERNAL"
    }
  }
}
```

- kms:GetParametersForImport (키 정책 또는 IAM 정책)
  - 이 권한을 특정 래핑 알고리즘 및 래핑 키 사양을 사용하는 요청으로 제한하려면 kms:WrappingAlgorithm 및 kms:WrappingKeySpec 정책 조건을 사용합니다.
- kms:ImportKeyMaterial(키 정책 또는 IAM 정책)
  - 만료되는 키 구성 요소를 허용하거나 금지하고 만료 날짜를 제어하려면 kms:ExpirationModel 및 kms:ValidTo 정책 조건을 사용합니다.

가져온 키 구성 요소를 다시 가져오려면 보안 주체가 kms:GetParametersForImport 및 kms:ImportKeyMaterial 권한을 필요합니다.

가져온 키 구성 요소를 삭제하려면 보안 주체에게 kms:DeleteImportedKeyMaterial 권한이 필요합니다.

예를 들어 가져온 키 구성 요소가 있는 KMS 키의 모든 측면을 관리할 수 있는 예제 KMSAdminRole 권한을 부여하려면 KMS 키의 키 정책에 다음과 같은 키 정책 설명을 포함합니다.

```
{
  "Sid": "Manage KMS keys with imported key material",
  "Effect": "Allow",
  "Resource": "*",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/KMSAdminRole"
  },
  "Action": [
    "kms:GetParametersForImport",
    "kms:ImportKeyMaterial",
    "kms:DeleteImportedKeyMaterial"
  ]
}
```

가져온 키 구성 요소에 대한 요구 사항

가져온 키 구성 요소는 관련 KMS 키의 키 사양과 호환되어야 합니다. 비대칭 키 페어의 경우 페어의 프라이빗 키만 가져옵니다.는 프라이빗 키에서 퍼블릭 키를 AWS KMS 파생합니다.

AWS KMS 는 가져온 키 구성 요소가 있는 KMS 키에 대해 다음 키 사양을 지원합니다.

- 대칭 암호화 키
  - 키 사양:
    - SYMMETRIC_DEFAULT.
  - 요구 사항:
    - 256비트(32바이트)의 이진 데이터.
    - 중국 리전의 경우 128비트(16바이트)의 이진 데이터여야 합니다.
- HMAC 키
  - 키 사양:
    - HMAC_224
    - HMAC_256
    - HMAC_384
    - HMAC_512
  - 요구 사항:

- HMAC 키 구성 요소는 RFC 2104를 준수해야 합니다.
- 키 길이는 키 사양에 지정된 길이 이상이어야 합니다. 최대 키 길이는 1024비트입니다.
- 키 구성 요소가 1024비트를 초과하는 경우, 키 구성 요소를 해시해 해시 출력값을 가져올 수 있습니다. 해싱 알고리즘은 생성하려는 HMAC KMS 키의 키 사양과 일치해야 합니다.

- 예:
  - 2048비트 키 구성 요소를 HMAC_256 키에 가져오려면 먼저 2048비트 키 구성 요소의 SHA-256 해시를 컴퓨팅한 뒤 해당 256비트 해시 출력값을 KMS 키로 가져오세요.
- 유효한 키 길이:
  - HMAC_224: 224~1024비트
  - HMAC_256: 256~1024비트
  - HMAC_384: 384~1024비트
  - HMAC_512: 512~1024비트

- RSA 비대칭 프라이빗 키
  - 키 사양:
    - RSA_2048
    - RSA_3072
    - RSA_4096
  - 요구 사항:
    - 가져오는 RSA 비대칭 프라이빗 키는 RFC 3447을 준수하는 키 쌍의 일부여야 합니다.
    - 모듈러스: 2048비트, 3072비트 또는 4096비트
    - 프라임 수: 2(다중 프라임 RSA 키는 지원되지 않음)
    - 비대칭 키 구성 요소는 RFC 5208을 준수하는 퍼블릭 키 암호화 표준(PKCS) #8 형식으로 BER 인코딩 또는 DER 인코딩이어야 합니다.
- 타원 곡선 비대칭 프라이빗 키
  - 키 사양:
    - ECC_NIST_P256(secp256r1)
    - ECC_NIST_P384(secp384r1)
    - ECC_NIST_P521(secp521r1)
    - ECC_SECG_P256K1(secp256k1)
    - ECC_NIST_EDWARDS25519(ed25519)
  - 요구 사항:

    - 가져오는 ECC 비대칭 프라이빗 키는 RFC 5915을 준수하는 키 쌍의 일부여야 합니다.
    - 곡선: NIST P-256, NIST P-384, NIST P-521, Secp256k1, NIST Ed25519.
    - 파라미터: 명명된 곡선만(명시적 파라미터가 있는 ECC 키는 거부됨).
    - 퍼블릭 포인트 좌표: 압축, 비압축 또는 투영형일 수 있음.
    - 비대칭 키 구성 요소는 RFC 5208을 준수하는 퍼블릭 키 암호화 표준(PKCS) #8 형식으로 BER 인코딩 또는 DER 인코딩이어야 합니다.
- SM2 비대칭 프라이빗 키(중국 리전만 해당)
  - 요구 사항:
    - 가져오는 SM2 비대칭 프라이빗 키는 GM/T 0003을 준수하는 키 페어의 일부여야 합니다.
    - 곡선: SM2.
    - 파라미터: 명명된 곡선만(명시적 파라미터가 있는 SM2 키는 거부됨).
    - 퍼블릭 포인트 좌표: 압축, 비압축 또는 투영형일 수 있음.
    - 비대칭 키 구성 요소는 RFC 5208을 준수하는 퍼블릭 키 암호화 표준(PKCS) #8 형식으로 BER 인코딩 또는 DER 인코딩이어야 합니다.

1단계: 키 구성 요소 AWS KMS key 없이 생성

기본적으로는 KMS 키를 AWS KMS 생성할 때 키 구성 요소를 생성합니다. 고유한 키 구성 요소를 대신 가져오려면, 키 구성 요소 없이 KMS 키를 만드는 것으로 시작합니다. 그런 다음 키 구성 요소를 가져옵니다. 키 구성 요소 없이 KMS 키를 생성하려면 AWS KMS 콘솔 또는 CreateKey 작업을 사용합니다.

키 구성 요소 없이 키를 만들려면 `EXTERNAL`의 오리진을 지정하세요. KMS 키의 오리진 속성은 변경할 수 없습니다. 키 구성 요소를 생성한 후에는 가져온 키 구성 요소용으로 설계된 KMS 키를 AWS KMS 또는 다른 소스의 키 구성 요소가 있는 KMS 키로 변환할 수 없습니다.

오리진이 `EXTERNAL`이고 키 구성 요소가 없는 KMS 키의 키 상태는 `PendingImport`입니다. KMS 키는 `PendingImport` 상태로 무기한 유지될 수 있습니다. 하지만 암호화 작업에서는 `PendingImport` 상태의 KMS 키를 사용할 수 없습니다. 키 구성 요소를 가져올 때 KMS 키의 키 상태는 `Enabled` 상태로 변경되고 암호화 작업에 KMS 키를 사용할 수 있습니다.

AWS KMS 는 KMS 키를 생성하고, 퍼블릭 키와 가져오기 토큰을 다운로드하고, 키 구성 요소를 가져올 때 AWS CloudTrail 로그에 이벤트를 기록합니다. AWS KMS 는 가져온 키 구성 요소를 삭제하거나 가 만료된 키 구성 요소를 AWS KMS 삭제할 때 CloudTrail 이벤트도 기록합니다.

주제

- 키 구성 요소 없이 KMS 키 생성(콘솔)
- 키 구성 요소 없이 KMS 키 생성(AWS KMS API)

키 구성 요소 없이 KMS 키 생성(콘솔)

가져온 키 구성 요소에 대한 KMS 키는 한 번만 생성하면 됩니다. 필요한 만큼 자주 동일한 키 구성 요소를 기존 KMS로 가져오고 다시 가져올 수 있지만, 다른 키 구성 요소를 KMS 키로 가져올 수는 없습니다. 자세한 내용은 2단계: 래핑 퍼블릭 키 및 가져오기 토큰 다운로드을 참조하세요.

고객 관리형 키(Customer managed keys) 테이블에서 가져온 키 구성 요소가 있는 기존 KMS 키를 찾기 위해서는 오른쪽 상단 모서리에 있는 톱니바퀴 아이콘을 사용하여 KMS 키 목록에 오리진(Origin) 열을 표시합니다. 가져온 키의 오리진 값은 외부(키 구성 요소 가져오기)입니다.

가져온 키 구성 요소가 있는 KMS 키를 생성하려면 먼저 원하는 키 유형의 KMS 키를 생성하는 지침을 따르세요. 단, 다음과 같은 경우는 예외입니다.

키 사용을 선택한 후 다음을 수행합니다.

1. 고급 옵션을 확장합니다.
2. 키 구성 요소 오리진(Key material origin)에서 키 구성 요소 가져오기(Import key material)를 선택합니다.
3. I understand the security and durability implications of using an imported key (가져온 키 사용의 보안 및 내구성 영향을 이해합니다) 옆 확인란을 선택하여, 가져온 키 구성 요소를 사용하는 것의 의미를 이해했음을 표시합니다. 이러한 의미에 대한 자세한 내용은 가져온 키 구성 요소 보호 섹션을 참조하세요.
4. 선택 사항: 가져온 키 구성 요소로 다중 리전 KMS 키를 생성하려면 리전 특성에서 다중 리전 키를 선택합니다.
5. 기본 지침으로 돌아갑니다. 기본 절차의 나머지 단계는 해당 유형의 모든 KMS 키에 대해 동일합니다.

마침을 선택하면 키 구성 요소가 없고 상태(키 상태)가 가져오기 보류 중인 KMS 키가 생성됩니다.

하지만 콘솔에는 고객 관리형 키 테이블로 돌아가는 대신 키 구성 요소를 가져오는 데 필요한 퍼블릭 키와 가져오기 토큰을 다운로드할 수 있는 페이지가 표시됩니다. 지금 다운로드 단계를 계속하거나 취소를 선택하여 이 시점에서 중단할 수 있습니다. 언제든지 이 다운로드 단계로 돌아갈 수 있습니다.

다음: 2단계: 래핑 퍼블릭 키 및 가져오기 토큰 다운로드.

키 구성 요소 없이 KMS 키 생성(AWS KMS API)

AWS KMS API를 사용하여 키 구성 요소가 없는 대칭 암호화 KMS 키를 생성하려면 `Origin` 파라미터가 `EXTERNAL`로 설정된 상태에서 CreateKey 요청을 보냅니다. 다음 예에서는 AWS Command Line Interface (AWS CLI)에서 이 작업을 수행하는 방법을 보여줍니다.

```
$ aws kms create-key --origin EXTERNAL
```

명령이 제대로 실행되면 다음과 비슷한 출력이 표시됩니다. AWS KMS 키는 `Origin` `EXTERNAL`이고 키는 `KeyState`입니다`PendingImport`.

> **Tip**
>
> 명령이 실패하면 `KMSInvalidStateException` 또는 `NotFoundException`이 표시될 수 있습니다. 요청을 재시도할 수 있습니다.

```
{
    "KeyMetadata": {
        "Origin": "EXTERNAL",
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "Description": "",
        "Enabled": false,
        "MultiRegion": false,
        "KeyUsage": "ENCRYPT_DECRYPT",
        "KeyState": "PendingImport",
        "CreationDate": 1568289600.0,
        "Arn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "AWSAccountId": "111122223333",
        "KeyManager": "CUSTOMER",
        "KeySpec": "SYMMETRIC_DEFAULT",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ]
    }
}
```

이후 단계에 사용할 명령 출력에서 `KeyId` 값을 복사한 후 2단계: 래핑 퍼블릭 키 및 가져오기 토큰 다운로드으로 넘어갑니다.

> **Note**
>
> 이 명령은 `SYMMETRIC_DEFAULT`의 `KeySpec`과 `ENCRYPT_DECRYPT`의 `KeyUsage`를 사용하여 대칭 암호화 KMS 키를 생성합니다. 선택적 파라미터 `--key-spec` 및 `--key-usage`를 사용하여 비대칭 또는 HMAC KMS 키를 생성할 수 있습니다. 자세한 내용은 CreateKey API 작업을 참조하세요.

2단계: 래핑 퍼블릭 키 및 가져오기 토큰 다운로드

키 구성 요소 없이 AWS KMS key 를 생성한 후 AWS KMS 콘솔 또는 GetParametersForImport API를 사용하여 해당 KMS 키에 대한 래핑 퍼블릭 키와 가져오기 토큰을 다운로드합니다. 래핑 퍼블릭 키와 가져오기 토큰은 분리할 수 없는 세트이므로 함께 사용해야 합니다.

래핑 퍼블릭 키를 사용하여 전송용 키 구성 요소를 암호화합니다. RSA 래핑 키 페어를 다운로드하기 전에 RSA 래핑 키 페어의 길이(키 사양)와 3단계에서 전송을 위해 가져온 키 구성 요소를 암호화하는 데 사용할 래핑 알고리즘을 선택합니다.는 SM2 래핑 키 사양(중국 리전만 해당) AWS KMS 도 지원합니다.

각 래핑 퍼블릭 키와 가져오기 토큰 세트는 24시간 동안 유효합니다. 다운로드한 후 24시간 내에 이를 이용해 키 구성 요소를 가져오지 않을 경우, 새로운 세트를 다운로드해야 합니다. 언제든지 새로운 래핑 퍼블릭 키를 다운로드하고 토큰 세트를 가져올 수 있습니다. 이를 통해 RSA 래핑 키 길이('키 사양')를 변경하거나 손실된 세트를 교체할 수 있습니다.

래핑 퍼블릭 키와 가져오기 토큰 세트를 다운로드하여 동일한 키 구성 요소를 KMS 키로 다시 가져올 수도 있습니다. 이 방법으로 키 구성 요소의 만료 시간을 설정 또는 변경하거나, 만료되었거나 삭제된 키 구성 요소를 복원할 수 있습니다. 키 구성 요소를 가져올 때마다 키 구성 요소를 다운로드하고 다시 암호화해야 합니다 AWS KMS.

래핑 퍼블릭 키 사용

다운로드에는 래핑 퍼블릭 키라고도 하는 AWS 계정에 고유한 퍼블릭 키가 포함됩니다.

키 구성 요소를 가져오기 전에 퍼블릭 래핑 키로 키 구성 요소를 암호화한 다음 암호화된 키 구성 요소를에 업로드합니다 AWS KMS. 가 암호화된 키 구성 요소를 AWS KMS 수신하면 해당 프라이빗 키로 키 구성 요소를 복호화한 다음 AWS KMS 하드웨어 보안 모듈(HSM) 내에서 AES 대칭 키로 키 구성 요소를 다시 암호화합니다.

가져오기 토큰 사용

다운로드에는 키 구성 요소를 올바르게 가져올 수 있도록 하는 메타데이터가 포함된 가져오기 토큰이 포함되어 있습니다. 암호화된 키 구성 요소를에 업로드 AWS KMS할 때는이 단계에서 다운로드한 것과 동일한 가져오기 토큰을 업로드해야 합니다.

래핑 퍼블릭 키 사양 선택

가져오기 중에 키 구성 요소를 보호하려면 다운로드한 래핑 퍼블릭 키 AWS KMS와 지원되는 래핑 알고리즘을 사용하여 키 구성 요소를 암호화합니다. 래핑 퍼블릭 키와 가져오기 토큰을 다운로드하기 전에 키 사양을 선택합니다. 모든 래핑 키 페어는 AWS KMS 하드웨어 보안 모듈(HSMs)에서 생성됩니다. 프라이빗 키는 절대 일반 텍스트로 HSM을 벗어나지 않습니다.

RSA 래핑 키 사양

래핑 퍼블릭 키의 키 사양에 따라 AWS KMS로 전송되는 동안 키 구성 요소를 보호하는 RSA 키 쌍의 키 길이가 결정됩니다. 일반적으로, 실용적이고 가장 긴 래핑 퍼블릭 키를 사용하는 것이 좋습니다. 다양한 HSM 및 키 관리자를 지원하기 위해 여러 가지 래핑 퍼블릭 키 사양을 제공하고 있습니다.

AWS KMS 는 명시된 경우를 제외하고 모든 유형의 키 구성 요소를 가져오는 데 사용되는 RSA 래핑 키에 대해 다음 키 사양을 지원합니다.

- RSA_4096(권장)
- RSA_3072
- RSA_2048

> **Note**
>
> ECC_NIST_P521 키 구성 요소, RSA_2048 퍼블릭 래핑 키 사양, RSAES_OAEP_SHA_* 래핑 알고리즘과 같은 조합은 지원되지 않습니다.
> ECC_NIST_P521 키 구성 요소를 RSA_2048 퍼블릭 래핑 키로 직접 래핑할 수는 없습니다. 더 큰 래핑 키나 RSA_AES_KEY_WRAP_SHA_* 래핑 알고리즘을 사용하세요.

SM2 래핑 키 사양(중국 리전만 해당)

AWS KMS 는 비대칭 키 구성 요소를 가져오는 데 사용되는 SM2 래핑 키에 대해 다음 키 사양을 지원합니다.

- SM2

래핑 알고리즘 선택

가져오는 동안 키 구성 요소를 보호하려면 다운로드한 래핑 퍼블릭 키와 지원되는 래핑 알고리즘을 사용하여 암호화합니다.

AWS KMS 는 여러 표준 RSA 래핑 알고리즘과 2단계 하이브리드 래핑 알고리즘을 지원합니다. 일반적으로, 가져온 키 구성 요소 및 래핑 키 사양과 호환되는 가장 안전한 래핑 알고리즘을 사용하는 것이 좋습니다. 일반적으로 키 구성 요소를 보호하는 하드웨어 보안 모듈(HSM) 또는 키 관리 시스템에서 지원되는 알고리즘을 선택합니다.

다음 테이블에는 각 유형의 키 구성 요소 및 KMS 키에 지원되는 래핑 알고리즘이 나와 있습니다. 알고리즘은 기본 설정 순서대로 나열됩니다.

키 구성 요소	지원되는 래핑 알고리즘 및 사양
대칭 암호화 키 256비트 AES 키 128비트 SM4 키(중국 리전 전용)	래핑 알고리즘: RSAES_OAEP_SHA_256 RSAES_OAEP_SHA_1 더 이상 사용되지 않는 래핑 알고리즘: RSAES_PKCS1_V1 **Note** 2023년 10월 10일부터 AWS KMS 는 RSAES_PKCS1_V1_5 래핑 알고리즘을 지원하지 않습니다. 래핑 키 사양: RSA_2048 RSA_3072 RSA_4096

키 구성 요소	지원되는 래핑 알고리즘 및 사양
비대칭 RSA 프라이빗 키	래핑 알고리즘: RSA_AES_KEY_WRAP_SHA_256 RSA_AES_KEY_WRAP_SHA_1 SM2PKE(중국 리전만 해당) 래핑 키 사양: RSA_2048 RSA_3072 RSA_4096 SM2(중국 리전 전용)
비대칭 타원 곡선(ECC) 프라이빗 키 RSA_2048 래핑 키 사양과 함께 RSAES_OAEP_SHA_* 래핑 알고리즘을 사용하여 ECC_NIST_P521 키 구성 요소를 래핑할 수 없습니다.	래핑 알고리즘: RSA_AES_KEY_WRAP_SHA_256 RSA_AES_KEY_WRAP_SHA_1 RSAES_OAEP_SHA_256 RSAES_OAEP_SHA_1 SM2PKE(중국 리전만 해당) 래핑 키 사양: RSA_2048 RSA_3072 RSA_4096 SM2(중국 리전 전용)

키 구성 요소	지원되는 래핑 알고리즘 및 사양
비대칭 SM2 프라이빗 키(중국 리전만 해당)	래핑 알고리즘: RSAES_OAEP_SHA_256 RSAES_OAEP_SHA_1 SM2PKE(중국 리전만 해당) 래핑 키 사양: RSA_2048 RSA_3072 RSA_4096 SM2(중국 리전 전용)
HMAC 키	래핑 알고리즘: RSAES_OAEP_SHA_256 RSAES_OAEP_SHA_1 래핑 키 사양: RSA_2048 RSA_3072 RSA_4096

> **Note**
>
> RSA_AES_KEY_WRAP_SHA_256 및 RSA_AES_KEY_WRAP_SHA_1 래핑 알고리즘은 중국 리전에서는 지원되지 않습니다.

- RSA_AES_KEY_WRAP_SHA_256 -생성한 AES 대칭 키로 키 구성 요소를 암호화한 다음 다운로드한 RSA 퍼블릭 래핑 키와 RSAES_OAEP_SHA_256 래핑 알고리즘을 사용하여 AES 대칭 키를 암호화하는 2단계 하이브리드 래핑 알고리즘입니다.

  SM2PKE 래핑 알고리즘을 사용해야 하는 중국 리전을 제외하고 RSA 프라이빗 키 구성 요소를 래핑하기 위해서는 RSA_AES_KEY_WRAP_SHA_* 래핑 알고리즘이 필요합니다.
- RSA_AES_KEY_WRAP_SHA_1 -생성한 AES 대칭 키로 키 구성 요소를 암호화한 다음 다운로드한 RSA 래핑 퍼블릭 키와 RSAES_OAEP_SHA_1 래핑 알고리즘을 사용하여 AES 대칭 키를 암호화하는 2단계 하이브리드 래핑 알고리즘입니다.

  SM2PKE 래핑 알고리즘을 사용해야 하는 중국 리전을 제외하고 RSA 프라이빗 키 구성 요소를 래핑하기 위해서는 RSA_AES_KEY_WRAP_SHA_* 래핑 알고리즘이 필요합니다.
- RSAES_OAEP_SHA_256 – SHA-256 해시 기능이 있는 OAEP(Optimal Asymmetric Encryption Padding)를 사용한 RSA 암호화 알고리즘입니다.
- RSAES_OAEP_SHA_1 – SHA-1 해시 기능이 있는 OAEP(Optimal Asymmetric Encryption Padding)를 사용한 RSA 암호화 알고리즘입니다.
- RSAES_PKCS1_V1_5 (사용되지 않음, 2023년 10월 10일부터 RSAES_PKCS1_V1_5 래핑 알고리즘을 지원하지 AWS KMS 않음) - PKCS #1 버전 1.5에 정의된 패딩 형식의 RSA 암호화 알고리즘입니다.
- SM2PKE(중국 리전만 해당) – OSCCA가 GM/T 0003.4-2012에서 정의한 타원형 곡선 기반 암호화 알고리즘입니다.

주제

- 래핑 퍼블릭 키 및 가져오기 토큰 다운로드(콘솔)
- 래핑 퍼블릭 키 및 가져오기 토큰 다운로드(AWS KMS API)

래핑 퍼블릭 키 및 가져오기 토큰 다운로드(콘솔)

AWS KMS 콘솔을 사용하여 래핑 퍼블릭 키와 가져오기 토큰을 다운로드할 수 있습니다.

1. 방금 키 구성 요소 없이 KMS 키를 생성하기 위한 절차를 마쳤고 현재 Download wrapping key and import token(래핑 키 및 가져오기 키 다운로드) 페이지에 있다면 Step 10으로 넘어갑니다.
2. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.
3. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.

4. 탐색 창에서 고객 관리형 키를 선택합니다.

> **Tip**
>
> 키 구성 요소는 오리진이 외부(키 구성 요소 가져오기)인 KMS 키로만 가져올 수 있습니다. 이는 키 구성 요소 없이 KMS 키가 생성되었음을 나타냅니다. 테이블에 Origin(오리진) 열을 추가하려면 페이지의 오른쪽 상단 모서리에서 설정 아이콘 ( ) 을 선택합니다. Origin(오리진)을 활성화한 다음, 확인을 선택합니다.

5. 가져오기가 보류 중인 KMS 키의 별칭 또는 키 ID를 선택합니다.
6. 암호화 구성(Cryptographic configuration) 탭을 선택하고 해당 값을 봅니다. 일반 구성 섹션 아래에 탭이 있습니다.

   키 구성 요소는 오리진이 외부(키 구성 요소 가져오기)인 KMS 키로만 가져올 수 있습니다. 가져온 키 구성 요소가 있는 KMS 키를 생성하는 방법은 키의 AWS KMS 키 구성 요소 가져오기 섹션을 참조하십시오.
7. 키 유형에 따라 적절한 탭을 선택합니다.

   - 비대칭 및 HMAC 키의 경우 키 구성 요소 탭을 선택합니다.
   - 대칭 암호화 키의 경우 키 구성 요소 및 교체 탭을 선택합니다.
8. 가져오기 작업을 선택합니다.

   - 비대칭 및 HMAC 키의 경우 키 구성 요소 가져오기를 선택합니다.
   - 대칭 암호화 키에서 다음 중 하나를 선택합니다.
     - 초기 키 구성 요소 가져오기(키 구성 요소를 아직 가져오지 않은 경우)
     - 새 키 구성 요소 가져오기(교체할 새 구성 요소 추가)
     - 키 구성 요소 다시 가져오기(키 구성 요소 테이블의 작업 메뉴에서 사용 가능)

> **Note**
>
> 다중 리전 키의 경우 먼저 새 키 구성 요소를 기본 리전 키로 가져와야 합니다. 그런 다음 동일한 키 구성 요소를 각 복제본 리전 키로 가져옵니다.
> 기본 다중 리전 키의 경우 키 자료 테이블에는 모든 복제본 리전의 가져오기 상태를 표시하는 복제본 가져오기 상태 열이 포함되어 있습니다(예: "가져온 3개 중 0개"). 복제본 가져

오기 상태 값을 선택하여 각 복제본 리전의 가져오기 상태를 보여주는 모달을 엽니다. 모달은 새 키 구성 요소를 가져오지 않은 복제본 리전에 대한 키 구성 요소 가져오기 링크를 제공합니다.

9. 래핑 키 사양 선택에서 KMS 키의 구성을 선택합니다. 이 키를 생성한 후에는 키 사양을 변경할 수 없습니다.
10. [Select wrapping algorithm]에서 키 구성 요소 암호화에 사용할 옵션을 선택합니다. 옵션에 대한 자세한 내용은 래핑 알고리즘 선택을 참조하십시오.
11. 래핑 퍼블릭 키 및 가져오기 토큰 다운로드를 선택한 후 파일을 저장합니다.

    다음 옵션이 있는 경우 지금 프로세스를 계속하려면 다음을 선택합니다. 나중에 계속하려면 취소를 선택합니다.
12. 앞 단계에 저장한 `.zip` 파일(`Import_Parameters_<key_id>_<timestamp>`)의 압축을 해제합니다.

    폴더에는 다음 파일이 포함되어 있습니다.

    - `WrappingPublicKey.bin`이라는 이름의 파일에 있는 래핑 퍼블릭 키
    - `ImportToken.bin`이라는 이름의 파일에 있는 가져오기 토큰입니다.
    - README.txt라는 이름의 텍스트 파일입니다. 이 파일에는 래핑 퍼블릭 키, 키 구성 요소를 암호화하는 데 사용할 래핑 알고리즘, 래핑 퍼블릭 키와 가져오기 토큰이 만료되는 날짜와 시간에 대한 정보가 포함되어 있습니다.
13. 프로세스를 계속하려면 키 구성 요소를 암호화를 참조하십시오.

래핑 퍼블릭 키 및 가져오기 토큰 다운로드(AWS KMS API)

GetParametersForImport API를 이용해 퍼블릭 키와 가져오기 토큰을 다운로드할 수 있습니다. 가져온 키 구성 요소와 연결될 KMS 키를 지정합니다. 이 KMS 키에는 `EXTERNAL`의 오리진 값이 있어야 합니다.

> **Note**
>
> ML-DSA KMS 키에는 키 구성 요소를 가져올 수 없습니다.

이 예시에서는 RSA_AES_KEY_WRAP_SHA_256 래핑 알고리즘, RSA_3072 래핑 퍼블릭 키 사양, 예시 키 ID를 지정합니다. 이러한 예시 값을 다운로드에 사용할 수 있는 유효한 값으로 바꾸세요. 키 ID의 경우, 이 작업에 키 ID나 키 ARN을 사용할 수 있지만 별칭 이름이나 별칭 ARN은 사용할 수 없습니다.

```
$ aws kms get-parameters-for-import \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --wrapping-algorithm RSA_AES_KEY_WRAP_SHA_256 \
    --wrapping-key-spec RSA_3072
```

명령이 제대로 실행되면 다음과 비슷한 출력이 표시됩니다.

```
{
    "ParametersValidTo": 1568290320.0,
    "PublicKey": "public key (base64 encoded)",
    "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "ImportToken": "import token (base64 encoded)"
}
```

다음 단계를 위한 데이터를 준비하려면 퍼블릭 키와 가져오기 토큰을 base64로 디코딩하고 디코딩된 값을 파일에 저장합니다.

퍼블릭 키 및 가져오기 토큰을 base64로 디코딩하려면

1. base64로 인코딩된 퍼블릭 키(예제 출력에 *### #(base64# ####)*로 표시됨)를 복사하여 새 파일에 붙여넣은 후 파일을 저장합니다. PublicKey.b64와 같은 파일을 설명하는 이름을 지정합니다.
2. OpenSSL을 이용해 파일의 콘텐츠를 base64로 디코딩하고 디코딩된 데이터를 새 파일에 저장합니다. 다음 예제는 앞의 단계에서 저장한 파일(PublicKey.b64)의 데이터를 디코딩하고 출력을 WrappingPublicKey.bin이라는 새 파일에 저장합니다.

   ```
   $ openssl enc -d -base64 -A -in PublicKey.b64 -out WrappingPublicKey.bin
   ```

3. base64로 인코딩된 가져오기 토큰(예제 출력에 *#### ##(base64# ####)*로 표시됨)을 복사하여 새 파일에 붙여넣은 후 파일을 저장합니다. importtoken.b64처럼 파일을 설명하는 이름을 지정합니다.
4. OpenSSL을 이용해 파일의 콘텐츠를 base64로 디코딩하고 디코딩된 데이터를 새 파일에 저장합니다. 다음 예제는 앞의 단계에서 저장한 파일(ImportToken.b64)의 데이터를 디코딩하고 출력을 ImportToken.bin이라는 새 파일에 저장합니다.

```
$ openssl enc -d -base64 -A -in importtoken.b64 -out ImportToken.bin
```

3단계: 키 구성 요소 암호화로 이동합니다.

3단계: 키 구성 요소 암호화

퍼블릭 키 및 가져오기 토큰을 다운로드한 후에는 다운로드한 퍼블릭 키와 지정한 래핑 알고리즘을 사용하여 키 구성 요소를 암호화합니다. 퍼블릭 키 또는 가져오기 토큰을 교체해야 하거나 래핑 알고리즘을 변경해야 하는 경우 새 퍼블릭 키와 가져오기 토큰을 다운로드해야 합니다. 에서 AWS KMS 지원하는 퍼블릭 키 및 래핑 알고리즘에 대한 자세한 내용은 래핑 퍼블릭 키 사양 선택 및 섹션을 참조하세요래핑 알고리즘 선택.

키 구성 요소는 이진 형식이어야 합니다. 자세한 내용은 가져온 키 구성 요소에 대한 요구 사항섹션을 참조하세요.

> **Note**
>
> 비대칭 키 페어의 경우 프라이빗 키만 암호화하고 가져옵니다.는 프라이빗 키에서 퍼블릭 키를 AWS KMS 파생합니다.
> ECC_NIST_P521 키 구성 요소, RSA_2048 퍼블릭 래핑 키 사양, RSAES_OAEP_SHA_* 래핑 알고리즘과 같은 조합은 지원되지 않습니다.
> ECC_NIST_P521 키 구성 요소를 RSA_2048 퍼블릭 래핑 키로 직접 래핑할 수는 없습니다. 더 큰 래핑 키나 RSA_AES_KEY_WRAP_SHA_* 래핑 알고리즘을 사용하세요.
> RSA_AES_KEY_WRAP_SHA_256 및 RSA_AES_KEY_WRAP_SHA_1 래핑 알고리즘은 중국 리전에서 지원되지 않습니다.

일반적으로 하드웨어 보안 모듈(HSM)이나 키 관리 시스템에서 내보낼 때 키 구성 요소를 암호화합니다. 이진 형식으로 키 구성 요소를 내보내는 방법은 HSM 또는 키 관리 시스템에 대한 문서를 참조하십시오. OpenSSL을 이용해 개념 증명 데모를 제공하는 다음 섹션을 참조할 수도 있습니다.

키 구성 요소를 암호화하는 경우 퍼블릭 키와 가져오기 토큰을 다운로드할 때 지정한 동일한 래핑 알고리즘을 사용합니다. 지정한 래핑 알고리즘을 찾으려면 관련 GetParametersForImport 요청에 대한 CloudTrail 로그 이벤트를 참조하세요.

테스트용 키 구성 요소 생성

다음 OpenSSL 명령은 테스트를 위해 지원되는 각 유형의 키 구성 요소를 생성합니다. 이러한 예시는 테스트 및 개념 증명 데모용으로만 제공됩니다. 프로덕션 시스템의 경우 보안이 보안 모듈이나 키 관리 시스템과 같은 보다 안전한 방법을 사용하여 키 구성 요소를 생성합니다.

비대칭 키 쌍의 프라이빗 키를 DER로 인코딩된 형식으로 변환하려면 키 구성 요소 생성 명령을 다음 `openssl pkcs8` 명령으로 파이프합니다. `topk8` 파라미터는 OpenSSL이 프라이빗 키를 입력으로 받아 PKCS#8 형식의 키를 반환하도록 지시합니다. (기본 동작은 반대입니다.)

```
openssl pkcs8 -topk8 -outform der -nocrypt
```

다음 명령은 지원되는 각 유형에 대한 테스트 키 구성 요소를 생성합니다.

- 대칭 암호화 키(32바이트)

  이 명령은 256비트 대칭 키(32바이트 임의 문자열)를 생성하여 `PlaintextKeyMaterial.bin` 파일에 저장합니다. 이 키 구성 요소를 인코딩할 필요는 없습니다.

  ```
  openssl rand -out PlaintextKeyMaterial.bin 32
  ```

  중국 리전에서만 128비트 대칭 키(16바이트 임의 문자열)를 생성해야 합니다.

  ```
  openssl rand -out PlaintextKeyMaterial.bin 16
  ```

- HMAC 키

  이 명령은 지정된 크기의 임의 바이트 문자열을 생성합니다. 이 키 구성 요소를 인코딩할 필요는 없습니다.

  HMAC 키의 길이는 KMS 키의 키 사양에 정의된 길이와 일치해야 합니다. 예를 들어 KMS 키가 HMAC_384인 경우 384비트(48바이트) 키를 가져와야 합니다.

  ```
  openssl rand -out HMAC_224_PlaintextKey.bin 28

  openssl rand -out HMAC_256_PlaintextKey.bin 32

  openssl rand -out HMAC_384_PlaintextKey.bin 48

  openssl rand -out HMAC_512_PlaintextKey.bin 64
  ```

- RSA 프라이빗 키

```
openssl genpkey -algorithm rsa -pkeyopt rsa_keygen_bits:2048 | openssl pkcs8 -topk8 -
outform der -nocrypt > RSA_2048_PrivateKey.der

openssl genpkey -algorithm rsa -pkeyopt rsa_keygen_bits:3072 | openssl pkcs8 -topk8 -
outform der -nocrypt > RSA_3072_PrivateKey.der

openssl genpkey -algorithm rsa -pkeyopt rsa_keygen_bits:4096 | openssl pkcs8 -topk8 -
outform der -nocrypt > RSA_4096_PrivateKey.der
```

- ECC 프라이빗 키

```
openssl genpkey -algorithm ec -pkeyopt ec_paramgen_curve:P-256 | openssl pkcs8 -topk8
 -outform der -nocrypt > ECC_NIST_P256_PrivateKey.der

openssl genpkey -algorithm ec -pkeyopt ec_paramgen_curve:P-384 | openssl pkcs8 -topk8
 -outform der -nocrypt > ECC_NIST_P384_PrivateKey.der

openssl genpkey -algorithm ec -pkeyopt ec_paramgen_curve:P-521 | openssl pkcs8 -topk8
 -outform der -nocrypt > ECC_NIST_P521_PrivateKey.der

openssl genpkey -algorithm ec -pkeyopt ec_paramgen_curve:secp256k1 | openssl pkcs8 -
topk8 -outform der -nocrypt > ECC_SECG_P256K1_PrivateKey.der
```

- SM2 프라이빗 키(중국 리전만 해당)

```
openssl genpkey -algorithm ec -pkeyopt ec_paramgen_curve:sm2 | openssl pkcs8 -topk8 -
outform der -nocrypt > SM2_PrivateKey.der
```

OpenSSL를 사용한 키 구성 요소 암호화의 예시

다음 예시는 OpenSSL을 사용하여 다운로드한 퍼블릭 키로 키 구성 요소를 암호화하는 방법을 보여줍니다. SM2 퍼블릭 키(중국 리전만 해당)를 사용하여 키 구성 요소를 암호화하려면 `SM2OfflineOperationHelper` 클래스를 사용합니다. 각 래핑 알고리즘에서 지원하는 주요 구성 요소 유형에 대한 자세한 내용은 the section called “래핑 알고리즘 선택” 단원을 참조하세요.

> **Important**
>
> 이 예시는 개념 증명 데모일 뿐입니다. 프로덕션 시스템에서 보안이 강화된 메서드(상용 HSM 또는 키 관리 시스템 등)를 사용해 키 구성 요소를 생성하고 저장합니다.

ECC_NIST_P521 키 구성 요소, RSA_2048 퍼블릭 래핑 키 사양, RSAES_OAEP_SHA_* 래핑 알고리즘과 같은 조합은 지원되지 않습니다.
ECC_NIST_P521 키 구성 요소를 RSA_2048 퍼블릭 래핑 키로 직접 래핑할 수는 없습니다. 더 큰 래핑 키나 RSA_AES_KEY_WRAP_SHA_* 래핑 알고리즘을 사용하세요.

RSAES_OAEP_SHA_1

AWS KMS 는 대칭 암호화 키(SYMMETRIC_DEFAULT), 타원 곡선(ECC) 프라이빗 키, SM2 프라이빗 키 및 HMAC 키에 대해 RSAES_OAEP_SHA_1을 지원합니다.

RSAES_OAEP_SHA_1는 RSA 프라이빗 키에는 지원되지 않습니다. 또한 RSAES_OAEP_SHA_* 래핑 알고리즘과 함께 RSA_2048 퍼블릭 래핑 키를 사용하여 ECC_NIST_P521(secp521r1) 프라이빗 키를 래핑할 수 없습니다. 더 큰 퍼블릭 래핑 키나 RSA_AES_KEY_WRAP 래핑 알고리즘을 사용해야 합니다.

다음 예시에서는 다운로드한 퍼블릭 키와 RSAES_OAEP_SHA_1 래핑 알고리즘을 사용하여 키 구성 요소를 암호화하고 이를 `EncryptedKeyMaterial.bin` 파일에 저장합니다.

이 예시에서는 다음과 같습니다.

- *`WrappingPublicKey.bin`*은 다운로드한 래핑 퍼블릭 키가 들어 있는 파일입니다.
- *`PlaintextKeyMaterial.bin`*은 암호화하려는 키 구성 요소(예: `PlaintextKeyMaterial.bin`, `HMAC_384_PlaintextKey.bin` 또는`ECC_NIST_P521_PrivateKey.der`)가 들어 있는 파일입니다.

```
$ openssl pkeyutl \
    -encrypt \
    -in PlaintextKeyMaterial.bin \
    -out EncryptedKeyMaterial.bin \
    -inkey WrappingPublicKey.bin \
    -keyform DER \
    -pubin \
    -pkeyopt rsa_padding_mode:oaep \
    -pkeyopt rsa_oaep_md:sha1
```

RSAES_OAEP_SHA_256

AWS KMS 는 대칭 암호화 키(SYMMETRIC_DEFAULT), 타원 곡선(ECC) 프라이빗 키, SM2 프라이빗 키 및 HMAC 키에 대해 RSAES_OAEP_SHA_256을 지원합니다.

RSAES_OAEP_SHA_256은 RSA 프라이빗 키에는 지원되지 않습니다. 또한 RSAES_OAEP_SHA_* 래핑 알고리즘과 함께 RSA_2048 퍼블릭 래핑 키를 사용하여 ECC_NIST_P521(secp521r1) 프라이빗 키를 래핑할 수 없습니다. 더 큰 퍼블릭 키나 RSA_AES_KEY_WRAP 래핑 알고리즘을 사용해야 합니다.

다음 예시에서는 다운로드한 퍼블릭 키와 RSAES_OAEP_SHA_256 래핑 알고리즘을 사용하여 키 구성 요소를 암호화하고 이를 `EncryptedKeyMaterial.bin` 파일에 저장합니다.

이 예시에서는 다음과 같습니다.

- *`WrappingPublicKey.bin`*은 다운로드한 래핑 퍼블릭 키가 들어 있는 파일입니다. 콘솔에서 퍼블릭 키를 다운로드한 경우, 이 파일 이름은 `wrappingKey_`*`KMS key_key_ID_timestamp`*(예: `wrappingKey_f44c4e20-f83c-48f4-adc6-a1ef38829760_0809092909`)입니다.
- *`PlaintextKeyMaterial.bin`*은 암호화하려는 키 구성 요소(예: `PlaintextKeyMaterial.bin`, `HMAC_384_PlaintextKey.bin` 또는`ECC_NIST_P521_PrivateKey.der`)가 들어 있는 파일입니다.

```
$ openssl pkeyutl \
    -encrypt \
    -in PlaintextKeyMaterial.bin \
    -out EncryptedKeyMaterial.bin \
    -inkey WrappingPublicKey.bin \
    -keyform DER \
    -pubin \
    -pkeyopt rsa_padding_mode:oaep \
    -pkeyopt rsa_oaep_md:sha256 \
    -pkeyopt rsa_mgf1_md:sha256
```

RSA_AES_KEY_WRAP_SHA_1

RSA_AES_KEY_WRAP_SHA_1 래핑 알고리즘에는 두 가지 암호화 작업이 포함됩니다.

1. 생성한 AES 대칭 키와 AES 대칭 암호화 알고리즘을 사용하여 키 구성 요소를 암호화합니다.
2. 다운로드한 퍼블릭 키와 RSAES_OAEP_SHA_1 래핑 알고리즘을 사용하여 사용한 AES 대칭 키를 암호화합니다.

RSA_AES_KEY_WRAP_SHA_1 래핑 알고리즘에는 OpenSSL 버전 3.x 이상이 필요합니다.

1. 256비트 AES 대칭 암호화 키 생성

   이 명령은 256개의 임의 비트로 구성된 AES 대칭 암호화 키를 생성하여 aes-key.bin 파일에 저장합니다.

   ```
   # Generate a 32-byte AES symmetric encryption key
   $ openssl rand -out aes-key.bin 32
   ```

2. AES 대칭 암호화 키를 사용하여 키 구성 요소 암호화

   이 명령은 AES 대칭 암호화 키를 사용하여 키 구성 요소를 암호화하고 암호화된 키 구성 요소를 key-material-wrapped.bin 파일에 저장합니다.

   이 예시 명령에서는 다음과 같습니다.

   - *PlaintextKeyMaterial.bin*은 가져오려는 키 구성 요소(예: PlaintextKeyMaterial.bin, HMAC_384_PlaintextKey.bin, RSA_3072_PrivateKey.der 또는 ECC_NIST_P521_PrivateKey.der)가 들어 있는 파일입니다.
   - *aes-key.bin*은 이전 명령에서 생성한 256비트 AES 대칭 암호화 키가 들어 있는 파일입니다.

   ```
   # Encrypt your key material with the AES symmetric encryption key
   $ openssl enc -id-aes256-wrap-pad \
           -K "$(xxd -p < aes-key.bin | tr -d '\n')" \
           -iv A65959A6 \
           -in PlaintextKeyMaterial.bin \
           -out key-material-wrapped.bin
   ```

3. 퍼블릭 키를 사용하여 AES 대칭 암호화 키 암호화

   이 명령은 다운로드한 퍼블릭 키와 RSAES_OAEP_SHA_1 래핑 알고리즘을 사용하여 AES 대칭 암호화 키를 암호화하고 DER 인코딩한 다음 aes-key-wrapped.bin 파일에 저장합니다.

   이 예시 명령에서는 다음과 같습니다.

   - *WrappingPublicKey.bin*은 다운로드한 래핑 퍼블릭 키가 들어 있는 파일입니다. 콘솔에서 퍼블릭 키를 다운로드한 경우, 이 파일 이름은 wrappingKey_*KMS key_key_ID_timestamp*(예: wrappingKey_f44c4e20-f83c-48f4-adc6-a1ef38829760_0809092909)입니다.

- *aes-key.bin*은 이 예시 시퀀스의 첫 번째 명령에서 생성한 256비트 AES 대칭 암호화 키가 들어 있는 파일입니다.

```
# Encrypt your AES symmetric encryption key with the downloaded public key
$ openssl pkeyutl \
    -encrypt \
    -in aes-key.bin \
    -out aes-key-wrapped.bin \
    -inkey WrappingPublicKey.bin \
    -keyform DER \
    -pubin \
    -pkeyopt rsa_padding_mode:oaep \
    -pkeyopt rsa_oaep_md:sha1 \
    -pkeyopt rsa_mgf1_md:sha1
```

4. 가져올 파일 생성

   암호화된 키 구성 요소가 있는 파일과 암호화된 AES 키가 있는 파일을 연결합니다. `EncryptedKeyMaterial.bin` 파일에 저장합니다. 이 파일은 4단계: 키 구성 요소 가져오기에서 가져올 파일입니다.

   이 예시 명령에서는 다음과 같습니다.

   - *key-material-wrapped.bin*은 암호화된 키 구성 요소가 들어 있는 파일입니다.
   - *aes-key-wrapped.bin*은 암호화된 AES 암호화 키가 들어 있는 파일입니다.

```
# Combine the encrypted AES key and encrypted key material in a file
$ cat aes-key-wrapped.bin key-material-wrapped.bin > EncryptedKeyMaterial.bin
```

RSA_AES_KEY_WRAP_SHA_256

RSA_AES_KEY_WRAP_SHA_256 래핑 알고리즘에는 두 가지 암호화 작업이 포함됩니다.

1. 생성한 AES 대칭 키와 AES 대칭 암호화 알고리즘을 사용하여 키 구성 요소를 암호화합니다.
2. 다운로드한 퍼블릭 키와 RSAES_OAEP_SHA_256 래핑 알고리즘을 사용하여 사용한 AES 대칭 키를 암호화합니다.

RSA_AES_KEY_WRAP_SHA_256 래핑 알고리즘에는 OpenSSL 버전 3.x 이상이 필요합니다.

1. 256비트 AES 대칭 암호화 키 생성

   이 명령은 256개의 임의 비트로 구성된 AES 대칭 암호화 키를 생성하여 `aes-key.bin` 파일에 저장합니다.

   ```
   # Generate a 32-byte AES symmetric encryption key
   $ openssl rand -out aes-key.bin 32
   ```

2. AES 대칭 암호화 키를 사용하여 키 구성 요소 암호화

   이 명령은 AES 대칭 암호화 키를 사용하여 키 구성 요소를 암호화하고 암호화된 키 구성 요소를 `key-material-wrapped.bin` 파일에 저장합니다.

   이 예시 명령에서는 다음과 같습니다.

   - *`PlaintextKeyMaterial.bin`*은 가져오려는 키 구성 요소(예: `PlaintextKeyMaterial.bin`, `HMAC_384_PlaintextKey.bin`, `RSA_3072_PrivateKey.der` 또는 `ECC_NIST_P521_PrivateKey.der`)가 들어 있는 파일입니다.
   - *`aes-key.bin`*은 이전 명령에서 생성한 256비트 AES 대칭 암호화 키가 들어 있는 파일입니다.

   ```
   # Encrypt your key material with the AES symmetric encryption key
   $ openssl enc -id-aes256-wrap-pad \
           -K "$(xxd -p < aes-key.bin | tr -d '\n')" \
           -iv A65959A6 \
           -in PlaintextKeyMaterial.bin \
           -out key-material-wrapped.bin
   ```

3. 퍼블릭 키를 사용하여 AES 대칭 암호화 키 암호화

   이 명령은 다운로드한 퍼블릭 키와 RSAES_OAEP_SHA_256 래핑 알고리즘을 사용하여 AES 대칭 암호화 키를 암호화하고 DER 인코딩한 다음 `aes-key-wrapped.bin` 파일에 저장합니다.

   이 예시 명령에서는 다음과 같습니다.

- *WrappingPublicKey.bin*은 다운로드한 래핑 퍼블릭 키가 들어 있는 파일입니다. 콘솔에서 퍼블릭 키를 다운로드한 경우, 이 파일 이름은 `wrappingKey_`*KMS key_key_ID_timestamp*(예: `wrappingKey_f44c4e20-f83c-48f4-adc6-a1ef38829760_0809092909`)입니다.
- *aes-key.bin*은 이 예시 시퀀스의 첫 번째 명령에서 생성한 256비트 AES 대칭 암호화 키가 들어 있는 파일입니다.

```
# Encrypt your AES symmetric encryption key with the downloaded public key
$ openssl pkeyutl \
    -encrypt \
    -in aes-key.bin \
    -out aes-key-wrapped.bin \
    -inkey WrappingPublicKey.bin \
    -keyform DER \
    -pubin \
    -pkeyopt rsa_padding_mode:oaep \
    -pkeyopt rsa_oaep_md:sha256 \
    -pkeyopt rsa_mgf1_md:sha256
```

4. 가져올 파일 생성

   암호화된 키 구성 요소가 있는 파일과 암호화된 AES 키가 있는 파일을 연결합니다. `EncryptedKeyMaterial.bin` 파일에 저장합니다. 이 파일은 4단계: 키 구성 요소 가져오기에서 가져올 파일입니다.

   이 예시 명령에서는 다음과 같습니다.

   - *key-material-wrapped.bin*은 암호화된 키 구성 요소가 들어 있는 파일입니다.
   - *aes-key-wrapped.bin*은 암호화된 AES 암호화 키가 들어 있는 파일입니다.

```
# Combine the encrypted AES key and encrypted key material in a file
$ cat aes-key-wrapped.bin key-material-wrapped.bin > EncryptedKeyMaterial.bin
```

4단계: 키 구성 요소 가져오기로 이동합니다.

4단계: 키 구성 요소 가져오기

키 구성 요소를 암호화한 후 AWS KMS key와 함께 사용할 키 구성 요소를 가져올 수 있습니다. 키 구성 요소를 가져오려면 3단계: 키 구성 요소 암호화에서 암호화한 키 구성 요소와 2단계: 래핑 퍼블릭 키 및 가져오기 토큰 다운로드에서 다운로드한 가져오기 토큰을 업로드합니다. 퍼블릭 키와 가져오기 토큰을 다운로드했을 때 지정한 KMS 키로 키 구성 요소를 가져와야 합니다. 키 구성 요소를 성공적으로 가져오면 KMS 키의 키 상태가 `Enabled` 상태로 변경되고 암호화 작업에 KMS 키를 사용할 수 있습니다.

키 구성 요소를 가져올 때 키 구성 요소에 대해 선택적으로 만료 시간을 설정할 수 있습니다. 키 구성 요소가 만료되면, AWS KMS 가 키 구성 요소를 삭제하고 KMS 키를 사용할 수 없게 됩니다. 키 구성 요소를 가져온 후에는 현재 가져오기에 대한 만료 날짜를 설정, 변경 또는 취소할 수 없습니다. 이러한 값을 변경하려면 동일한 키 구성 요소를 다시 가져와야 합니다.

오리진이 `EXTERNAL`인 모든 KMS 키의 경우, 처음 가져온 키 구성 요소가 현재 키 구성 요소가 되며 해당 키와 영구적으로 연결됩니다. `EXTERNAL` 오리진이 있는 대칭 암호화 키는 온디맨드 교체를 지원합니다. 온디맨드 교체를 지원하는 가져온 키에는 여러 키 구성 요소를 연결할 수 있습니다. 새 키 구성 요소를 가져오는 프로세스는 새 키 구성 요소 가져오기 섹션에 설명된 대로 단일 리전 및 다중 리전 키에 따라 다릅니다. ImportKeyMaterial 작업을 사용할 때 `importType` 파라미터를 `NEW_KEY_MATERIAL`로 설정해야 새 키 구성 요소를 KMS 키와 연결할 수 있습니다. 선택적 `ImportType` 파라미터의 기본값은 `EXISTING_KEY_MATERIAL`입니다. `ImportType` 파라미터를 생략하거나 `EXISTING_KEY_MATERIAL`로 지정하는 경우, KMS 키와 이전에 연결된 키 구성 요소를 가져와야 합니다.

비대칭 또는 `EXTERNAL` 오리진이 있는 HMAC KMS 키의 경우 키 구성 요소 하나만 키와 연결할 수 있습니다. AWS KMS 는 `ImportType` 파라미터와 함께 ImportKeyMaterial API 요청을 거부합니다.

KMS 키에 영구적으로 연결된 모든 키 구성 요소를 가져오면 KMS 키를 암호화 작업에 사용할 수 있습니다. 이 키 구성 요소 중 하나라도 삭제되거나 만료되도록 허용되면 KMS 키 상태는 `PendingImport`로 변경되며, 해당 키는 암호화 작업에 사용할 수 없습니다.

키 구성 요소를 가져오려면 AWS KMS 콘솔 또는 ImportKeyMaterial API를 사용할 수 있습니다. AWS SDK, AWS Command Line Interface 또는 AWS Tools for PowerShell을 사용하거나 직접HTTP 요청을 수행해 API를 사용할 수 있습니다.

키 구성 요소를 가져오면 ImportKeyMaterial 항목이 AWS CloudTrail 로그에 추가되어 `ImportKeyMaterial` 작업을 기록합니다. CloudTrail 항목은 AWS KMS 콘솔을 사용하든 AWS KMS API를 사용하든 동일합니다.

만료 시간 설정(선택 사항)

KMS 키에 대한 키 구성 요소를 가져올 때 가져오기 날짜로부터 최대 365일까지 키 구성 요소에 대한 선택적 만료 날짜 및 시간을 설정할 수 있습니다. 가져온 키 구성 요소가 만료되면 AWS KMS 삭제합니다. 이 작업은 KMS 키의 키 상태를 `PendingImport`로 변경하여 암호화 작업에 사용되지 않도록 합니다. KMS 키를 사용하려면 원본 키 구성 요소의 사본을 다시 가져와야 합니다.

가져온 키 구성 요소가 자주 만료되도록 하면 규정 요구 사항을 충족하는 데 도움이 될 수 있지만 KMS 키로 암호화된 데이터에 추가적인 위험이 발생합니다. 원래 키 구성 요소의 사본을 다시 가져올 때까지 키 구성 요소가 만료된 KMS 키는 사용할 수 없으며 KMS 키로 암호화된 모든 데이터에 액세스할 수 없습니다. 원래 키 구성 요소의 사본 손실을 포함하여 어떤 이유로든 키 구성 요소를 다시 가져오지 못하면 KMS 키를 영구적으로 사용할 수 없으며 KMS 키로 암호화된 데이터를 복구할 수 없습니다.

이러한 위험을 완화하려면 가져온 키 구성 요소의 사본에 액세스할 수 있는지 확인하고 키 구성 요소가 만료되고 AWS 워크로드가 중단되기 전에 키 구성 요소를 삭제하고 다시 가져오도록 시스템을 설계합니다. 만료되기 전에 키 구성 요소를 다시 가져올 수 있는 충분한 시간을 제공하는 가져온 키 구성 요소의 만료 경보를 설정하는 것이 좋습니다. 또한 CloudTrail 로그를 사용하여 키 재료를 가져오고 다시 가져오는 작업과 가져온 키 재료를 삭제하는 작업, 만료된 키 재료를 삭제하는 AWS KMS 작업을 감사할 수 있습니다.

AWS KMS 는 삭제된 키 구성 요소를 복원, 복구 또는 재현할 수 없습니다. 만료 시간을 설정하는 대신 주기적으로 가져온 키 구성 요소를 프로그래밍 방식으로 삭제하고 다시 가져올 수 있지만 원래 키 구성 요소의 사본을 유지하기 위한 요구 사항은 동일합니다.

키 구성 요소를 가져올 때 가져온 키 구성 요소의 만료 여부와 시기를 결정합니다. 그러나 키 구성 요소를 다시 가져오면 만료 기능을 켜거나 끌 수 있으며 새로운 만료 시간을 설정할 수도 있습니다. ImportKeyMaterial의 `ExpirationModel` 파라미터를 사용하여 만료를 켰다가(`KEY_MATERIAL_EXPIRES`) 끄고(`KEY_MATERIAL_DOES_NOT_EXPIRE`) `ValidTo` 파라미터를 사용하여 만료 시간을 설정합니다. 최대 시간은 가져오기 데이터로부터 365일입니다. 최소값은 없지만 시간은 미래여야 합니다.

키 구성 요소 설명 설정

`EXTERNAL` 오리진이 있는 대칭 암호화 키에는 여러 키 구성 요소가 연결될 수 있습니다. 이러한 키에 키 구성 요소를 가져올 때 선택적으로 키 구성 요소 설명을 지정할 수 있습니다. 이 설명은 해당 키 구성 요소가 AWS KMS외부에서 지속적으로 유지되는 위치를 추적하는 데 사용할 수 있습니다.

다중 리전 키의 경우 기본 리전 키에서만 키 구성 요소 설명을 설정하거나 변경할 수 있습니다.는 키 구성 요소 설명을 복제본 리전 키로 AWS KMS 자동 전파합니다.

새 키 구성 요소 가져오기

가져온 키 구성 요소를 사용하는 대칭 암호화 KMS 키에 대해 온디맨드 교체를 수행하려면 먼저 해당 KMS 키와 이전에 연결되지 않은 새 키 구성 요소를 가져와야 합니다.

- 단일 리전 키
  - 이 작업을 수행하려면 ImportKeyMaterial 작업에서 `ImportType` 파라미터를 `NEW_KEY_MATERIAL`로 설정해 사용하세요. 이 키 구성 요소는 RotateKeyOnDemand 작업을 수행하거나에서 키를 교체할 때까지 키와 영구적으로 연결되지 않습니다 AWS Management Console. 그때까지 이 키 구성 요소는 `PENDING_ROTATION` 상태로 유지됩니다. KMS 키는 언제나 `PENDING_ROTATION` 상태의 키 구성 요소를 최대 하나만 가질 수 있습니다. 암호화 작업에서 키의 사용성에 영향을 주지 않고 `PENDING_ROTATION` 상태의 키 구성 요소를 삭제할 수 있습니다.
- 다중 리전 키
  - 키 구성 요소를 다중 리전 키로 가져오려면 먼저 새 키 구성 요소를 기본 리전 키로 가져와야 합니다. 새 키 구성 요소를 복제본 리전 키로 직접 가져올 수 없습니다. 새 키 구성 요소를 기본 리전 키로 가져온 후 동일한 키 구성 요소를 복제본 리전 키로 가져올 수 있습니다.
  - 이 작업을 수행하려면 기본 리전 키에 **`NEW_KEY_MATERIAL`** 대해 `ImportType` 파라미터가 로 설정된 `ImportKeyMaterial` 작업을 사용합니다. 복제본 리전 키의 경우 `ImportKeyMaterial` 작업에 `ImportType`의 **`EXISTING_KEY_MATERIAL`** 파라미터를 사용합니다.
  - 대칭 암호화 다중 리전 키의 키 구성 요소는 키 구성 요소 상태가 `PENDING_ROTATION` 상태로 변경되기 전에 모든 복제본 리전 키와 기본 리전 키로 가져와야 합니다. 그때까지 새 키 구성 요소의 상태는 입니다`PENDING_MULTI_REGION_IMPORT_AND_ROTATION`. KMS 키는 언제든지 `PENDING_ROTATION` 또는 `PENDING_MULTI_REGION_IMPORT_AND_ROTATION`상태에 최대 하나의 키 구성 요소를 가질 수 있습니다(RotationsListEntry의 `KeyMaterialState` 설명 참조). `PENDING_MULTI_REGION_IMPORT_AND_ROTATION` 또는 `PENDING_ROTATION` 상태의 키 구성 요소는 키와 영구적으로 연결되지 않으며 암호화 작업에서 키의 사용성에 영향을 주지 않고 삭제할 수 있습니다.

키 구성 요소 다시 가져오기

가져온 키 구성 요소가 있는 KMS 키를 관리하는 경우, 키 구성 요소를 다시 가져와야 할 수도 있습니다. 만료되거나 삭제된 키 구성 요소를 바꾸거나 키 구성 요소의 만료 모델 또는 만료 날짜를 변경하기 위해 키 구성 요소를 다시 가져올 수 있습니다.

보안 요구 사항을 충족하는 일정에 따라 언제든지 키 구성 요소를 다시 가져올 수 있습니다. 키 구성 요소가 만료 시간에 도달하거나 근접할 때까지 기다릴 필요가 없습니다.

키 구성 요소를 다시 가져오는 절차는 처음 키 구성 요소를 가져오는 절차와 동일하지만, 다음과 같은 예외가 있습니다.

- 새 KMS 키를 만드는 대신 기존 KMS 키를 사용합니다. 가져오기 절차의 1단계를 건너뛸 수 있습니다.
- 키 구성 요소를 다시 가져올 때 만료 모델 및 만료 날짜를 변경할 수 있습니다. 대칭 암호화 키의 경우 키 구성 요소 설명을 변경할 수도 있습니다.

  다중 리전 키의 경우 기본 리전 키에서만 키 구성 요소 설명을 설정하거나 변경할 수 있습니다.는 키 구성 요소 설명을 복제본 리전 키로 AWS KMS 자동 전파합니다.

키 구성 요소를 KMS 키로 가져올 때마다 해당 KMS 키의 가져오기 토큰과 새 래핑 키를 다운로드하여 사용해야 합니다. 래핑 절차는 키 구성 요소의 내용에 영향을 주지 않으므로 다른 래핑 퍼블릭 키와 다른 래핑 알고리즘을 사용하여 동일한 키 구성 요소를 가져올 수 있습니다.

키 구성 요소 가져오기(콘솔)

AWS Management Console 를 사용하여 키 구성 요소를 가져올 수 있습니다.

1. 래핑된 키 구성 요소 업로드 페이지에 있는 경우 Step 10으로 건너뛰세요.
2. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.
3. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
4. 탐색 창에서 고객 관리형 키를 선택합니다.
5. 퍼블릭 키와 가져오기 토큰을 다운로드한 KMS 키의 키 ID 또는 별칭을 선택합니다.
6. 암호화 구성(Cryptographic configuration) 탭을 선택하고 해당 값을 봅니다. 탭은 일반 구성 섹션 아래의 KMS 키에 대한 세부 정보 페이지에 있습니다.

   키 구성 요소는 오리진이 외부(키 구성 요소 가져오기)인 KMS 키로만 가져올 수 있습니다. 가져온 키 구성 요소가 있는 KMS 키를 생성하는 방법은 키의 AWS KMS 키 구성 요소 가져오기 단원을 참조하십시오.
7. 키 유형에 따라 적절한 탭을 선택합니다.

   - 비대칭 키와 HMAC 키의 경우 키 구성 요소 탭을 선택합니다.

- 대칭 암호화 키의 경우 키 구성 요소 및 교체 탭을 선택합니다.

8. 가져오기 작업을 선택합니다.

   - 비대칭 및 HMAC 키의 경우 키 구성 요소 가져오기를 선택합니다.
   - 대칭 암호화 키에서 다음 중 하나를 선택합니다.
     - 초기 키 구성 요소 가져오기(키 구성 요소를 아직 가져오지 않은 경우)
     - 새 키 구성 요소 가져오기(교체할 새 구성 요소 추가)
     - 키 구성 요소 다시 가져오기(키 구성 요소 테이블의 작업 메뉴에서 사용 가능)

   > **Note**
   >
   > 다중 리전 키의 경우 먼저 새 키 구성 요소를 기본 리전 키로 가져와야 합니다. 그런 다음 동일한 키 구성 요소를 각 복제본 리전 키로 가져옵니다.
   > 기본 다중 리전 키의 경우 키 자료 테이블에는 모든 복제본 리전의 가져오기 상태를 표시하는 복제본 가져오기 상태 열이 포함되어 있습니다(예: "가져온 3개 중 0개"). 복제본 가져오기 상태 값을 선택하여 각 복제본 리전의 가져오기 상태를 보여주는 모달을 엽니다. 모달은 새 키 구성 요소를 가져오지 않은 복제본 리전에 대한 키 구성 요소 가져오기 링크를 제공합니다.

9. 키 구성 요소를 다운로드하고, 토큰을 가져오고, 키 구성 요소를 암호화한 경우 다음을 선택합니다.

   > **Note**
   >
   > 다중 리전 키의 경우 먼저 새 키 구성 요소를 기본 리전 키로 가져와야 합니다. 그런 다음 동일한 키 구성 요소를 복제본 리전 키로 가져올 수 있습니다.

10. 암호화된 키 구성 요소 및 가져오기 토큰 섹션에서 다음을 수행하세요.

    a. 래핑된 키 구성 요소에서 파일 선택을 선택합니다. 그런 다음, 래핑(암호화)된 키 구성 요소가 포함된 파일을 업로드합니다.

    b. 가져오기 토큰에서 파일 선택을 선택합니다. 다운로드한 가져오기 토큰이 포함된 파일을 업로드합니다.

11. Expiration option(만료 옵션) 섹션에서 키 구성 요소의 만료 여부를 결정합니다. 만료 날짜와 시간을 설정하려면 Key material expires(키 구성 요소 만료)를 선택하고 캘린더를 사용해 날짜와 시간을 선택합니다. 현재 날짜 및 시간으로부터 최대 365일까지 날짜를 지정할 수 있습니다.

12. 대칭 암호화 키의 경우, 가져오는 키 구성 요소에 대한 설명을 선택적으로 지정할 수 있습니다.
13. 키 구성 요소 가져오기를 선택합니다.

키 구성 요소 가져오기(AWS KMS API)

키 구성 요소를 가져오려면 ImportKeyMaterial작업을 사용합니다. 다음의 예제는 AWS CLI을(를) 사용하지만 지원되는 모든 프로그래밍 언어를 사용할 수 있습니다.

이 예제 사용

1. *1234abcd-12ab-34cd-56ef-1234567890ab*을(를) 퍼블릭 키와 가져오기 토큰을 다운로드할 때 지정한 KMS 키의 키 ID로 바꿉니다. KMS 키를 식별하려면, 이 KMS 키의 키 ID 또는 키 ARN을 사용합니다. 이 작업에는 별칭 이름이나 별칭 ARN을 사용할 수 없습니다.
2. *EncryptedKeyMaterial.bin*을 암호화된 키 구성 요소가 포함된 파일 이름으로 바꿉니다.
3. *ImportToken.bin*을 가져오기 토큰이 포함된 파일 이름으로 바꿉니다.
4. 가져온 키 구성 요소가 만료되도록 하려면 `expiration-model` 파라미터 값을 기본값, `KEY_MATERIAL_EXPIRES`로 설정하거나 `expiration-model` 파라미터를 생략합니다. 그런 다음 `valid-to` 파라미터 값을 만료할 날짜와 시간으로 바꿉니다. 날짜 및 시간은 요청 시점으로부터 최대 365일이 될 수 있습니다.

   ```
   $ aws kms import-key-material --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
       --encrypted-key-material fileb://EncryptedKeyMaterial.bin \
       --import-token fileb://ImportToken.bin \
       --expiration-model KEY_MATERIAL_EXPIRES \
       --valid-to 2023-06-17T12:00:00-08:00
   ```

   가져온 키 구성 요소가 만료되지 않도록 하려면 명령에서 `expiration-model` 파라미터의 값을 `KEY_MATERIAL_DOES_NOT_EXPIRE`로 설정하고 `valid-to` 파라미터를 생략합니다.

   ```
   $ aws kms import-key-material --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
       --encrypted-key-material fileb://EncryptedKeyMaterial.bin \
       --import-token fileb://ImportToken.bin \
       --expiration-model KEY_MATERIAL_DOES_NOT_EXPIRE
   ```

5. KMS 키와 이전에 연결되지 않은 새 키 구성 요소를 가져오려면 `ImportType` 파라미터를 `NEW_KEY_MATERIAL`로 설정하세요. 이 옵션은 대칭 암호화 키에만 사용할 수 있습니다. 이러한 키의 경우 선택적 `KeyMaterialDescription` 파라미터를 사용하여 다음 명령줄 예제에서 가져온 키 구성 요소에 대한 설명을 설정할 수도 있습니다.

```
$ aws kms import-key-material --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --encrypted-key-material fileb://EncryptedKeyMaterial.bin \
    --import-token fileb://ImportToken.bin \
    --expiration-model KEY_MATERIAL_EXPIRES \
    --valid-to 2023-06-17T12:00:00-08:00 \
    --import-type NEW_KEY_MATERIAL \
    --key-material-description "Q2 2025 Rotation"
```

6. 다중 리전 키의 경우 기본 리전 키에서만 키 구성 요소 설명을 설정하거나 변경할 수 있습니다.는 키 구성 요소 설명을 복제본 리전 키로 AWS KMS 자동 전파합니다.

> **Tip**
>
> 명령이 실패하면 `KMSInvalidStateException` 또는 `NotFoundException`이 표시될 수 있습니다. 요청을 재시도할 수 있습니다.

키 스토어에서 KMS AWS CloudHSM 키 생성

AWS CloudHSM 키 스토어를 생성한 후 키 스토어 AWS KMS keys 에서를 생성할 수 있습니다. 가 AWS KMS 생성하는 키 구성 요소가 있는 대칭 암호화 KMS 키여야 합니다. 사용자 지정 키 스토어에 비대칭 KMS 키, HMAC KMS 키 또는 가져온 키 구성 요소가 있는 KMS 키를 생성할 수 없습니다. 또한 사용자 지정 키 스토어에서 대칭 암호화 KMS 키를 사용하여 비대칭 데이터 키 페어를 생성할 수 없습니다. KMS는 IPv6를 통해 AWS CloudHSM 키 스토어와 통신할 수 없습니다.

AWS CloudHSM 키 스토어에서 KMS 키를 생성하려면 AWS CloudHSM 키 스토어를 연결된 AWS CloudHSM 클러스터에 연결하고 클러스터에 서로 다른 가용 영역에 활성 HSMs 두 개 이상 있어야 합니다. 연결 상태 및 HSM 수를 찾으려면 AWS Management Console에서 AWS CloudHSM 키 스토어 페이지를 봅니다. API 작업을 사용할 때는 DescribeCustomKeyStores 작업을 사용하여 AWS CloudHSM 키 스토어가 연결되어 있는지 확인합니다. 클러스터의 활성 HSMs 수와 가용 영역을 확인하려면 AWS CloudHSM DescribeClusters 작업을 사용합니다.

AWS CloudHSM 키 스토어에서 KMS 키를 생성하면가 KMS 키를 AWS KMS 생성합니다 AWS KMS. 그러나 연결된 AWS CloudHSM 클러스터에서 KMS 키에 대한 키 구성 요소를 생성합니다. 특히는 클러스터에 생성한 CU로 AWS KMS 로그인합니다. `kmsuser` 그런 다음, 클러스터에서 지속적이고 추출 불가능한 256비트 고급 암호화 표준(AES) 대칭 키를 생성합니다. AWS KMS 는 클러스터에서만 표시되는 키 라벨 속성의 값을 KMS 키의 Amazon 리소스 이름(ARN)으로 설정합니다.

명령이 성공하면 새 KMS 키의 키 상태는 Enabled가 되고 오리진은 AWS_CLOUDHSM이 됩니다. 생성 이후에 KMS 키의 오리진을 변경할 수 없습니다. AWS KMS 콘솔의 키 스토어에서 또는 DescribeKey 작업을 사용하여 KMS AWS CloudHSM 키를 볼 때 키 ID, 키 상태, 생성 날짜와 같은 일반적인 속성을 볼 수 있습니다. 하지만 사용자 지정 키 스토어 ID와 AWS CloudHSM 클러스터 ID(선택 사항)을 확인할 수도 있습니다.

AWS CloudHSM 키 스토어에서 KMS 키를 생성하려는 시도가 실패하면 오류 메시지를 사용하여 원인을 확인할 수 있습니다. AWS CloudHSM 키 스토어가 연결되지 않았거나(CustomKeyStoreInvalidStateException) 연결된 AWS CloudHSM 클러스터에이 작업()에 필요한 두 개의 활성 HSMs이 없음을 나타낼 수 있습니다CloudHsmClusterInvalidConfigurationException. 도움말은 사용자 지정 키 스토어 문제 해결 섹션을 참조하십시오.

AWS CloudHSM 키 스토어에서 KMS 키를 생성하는 작업 AWS CloudTrail 로그의 예는 섹션을 참조하세요CreateKey.

CloudHSM 키 저장소에서 새로운 KMS 키 생성

콘솔의 AWS KMS 키 스토어에서 또는 CreateKey 작업을 사용하여 대칭 암호화 KMS AWS CloudHSM 키를 생성할 수 있습니다.

AWS KMS 콘솔 사용

다음 절차에 따라 키 스토어에 대칭 암호화 KMS AWS CloudHSM 키를 생성합니다.

> **Note**
>
> 별칭, 설명 또는 태그에 기밀 또는 민감한 정보를 포함하지 마십시오. 이러한 필드는 CloudTrail 로그 및 기타 출력에 일반 텍스트로 표시될 수 있습니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 탐색 창에서 고객 관리형 키를 선택합니다.
4. 키 생성(Create key)을 선택합니다.
5. 대칭(Symmetric)을 선택합니다.
6. 키 사용(Key usage)에서 암호화 및 해독(Encrypt and decrypt) 옵션이 선택됩니다. 변경할 수 없습니다.

7. 고급 옵션을 선택합니다.
8. Key material origin(키 구성 요소 오리진)에서 AWS CloudHSM key store를 선택합니다.

   키 스토어에는 다중 리전 AWS CloudHSM 키를 생성할 수 없습니다.
9. 다음을 선택합니다.
10. 새 KMS AWS CloudHSM 키의 키 스토어를 선택합니다. 새 AWS CloudHSM 키 스토어를 생성하려면 사용자 지정 키 스토어 생성을 선택합니다.

    선택한 AWS CloudHSM 키 스토어의 상태는 연결됨이어야 합니다. 연결된 AWS CloudHSM 클러스터는 활성 상태여야 하며 서로 다른 가용 영역에 두 개 이상의 활성 HSMs을 포함해야 합니다.

    AWS CloudHSM 키 스토어 연결에 대한 도움말은 섹션을 참조하세요AWS CloudHSM 키 스토어 연결 해제. HSM 추가에 대한 도움말은 AWS CloudHSM 사용 설명서의 HSM 추가를 참조하십시오.
11. 다음을 선택합니다.
12. KMS 키의 별칭과 설명을 입력합니다.
13. (선택 사항). 태그 추가(Add Tags) 페이지에서 KMS 키를 식별 및 분류하는 태그를 추가합니다.

    AWS 리소스에 태그를 추가하면가 태그별로 집계된 사용량 및 비용이 포함된 비용 할당 보고서를 AWS 생성합니다. KMS 키에 대한 액세스를 제어하는 데에도 태그를 사용할 수 있습니다. KMS 키 태그 지정에 대한 자세한 내용은 의 태그 AWS KMS 및 에 대한 ABAC AWS KMS 섹션을 참조하십시오.
14. 다음을 선택합니다.
15. 키 관리자(Key Administrators) 섹션에서 KMS 키를 관리할 수 있는 IAM 사용자 및 역할을 선택합니다. 자세한 내용은 키 관리자가 KMS 키를 관리하도록 허용을 참조하세요.

> **참고**
>
> IAM 정책은 다른 IAM 사용자 및 역할에 KMS 키 사용 권한을 제공할 수 있습니다.
> IAM 모범 사례는 장기 보안 인증 정보가 있는 IAM 사용자의 사용을 장려하지 않습니다. 가능할 경우, 임시 보안 인증 정보를 제공하는 IAM 역할을 사용하세요. 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.
> AWS KMS 콘솔은 문 식별자 아래의 키 정책에 키 관리자를 추가합니다`"Allow access for Key Administrators"`. 이 문 식별자를 수정하면 문에 대해 콘솔이 업데이트 내용을 표시하는 방식에 영향을 미칠 수 있습니다.

16. (선택 사항) 이러한 키 관리자가 KMS 키를 삭제하지 못하도록 하려면 페이지 하단에 있는 키 관리자가 이 키를 삭제하도록 허용(Allow key administrators to delete this key) 확인란의 선택을 취소합니다.
17. 다음을 선택합니다.
18. 이 계정 섹션에서 암호화 작업에 KMS 키를 사용할 수 AWS 계정 있는 IAM 사용자 및 역할을 선택합니다. 자세한 내용은 키 사용자가 KMS 키를 사용하도록 허용을 참조하세요.

> **참고**
>
> IAM 모범 사례는 장기 보안 인증 정보가 있는 IAM 사용자의 사용을 장려하지 않습니다. 가능할 경우, 임시 보안 인증 정보를 제공하는 IAM 역할을 사용하세요. 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.
> AWS KMS 콘솔은 명령문 식별자 `"Allow use of the key"` 및 아래의 키 정책에 키 사용자를 추가합니다`"Allow attachment of persistent resources"`. 이러한 문 식별자들을 수정하면 문에 대해 콘솔이 업데이트 내용을 표시하는 방식에 영향을 미칠 수 있습니다.

19. (선택 사항) 다른 사용자가 암호화 작업에이 KMS 키를 AWS 계정 사용하도록 허용할 수 있습니다. 이렇게 하려면 페이지 하단의 기타 AWS 계정 섹션에서 다른 추가 AWS 계정를 선택하고 외부 계정의 AWS 계정 ID를 입력합니다. 외부 계정을 여러 개 추가하려면 이 단계를 반복합니다.

> **Note**
>
> 또한 다른의 관리자는 사용자에 대한 IAM 정책을 생성하여 KMS 키에 대한 액세스를 허용 AWS 계정 해야 합니다. 자세한 내용은 다른 계정의 사용자가 KMS를 사용하도록 허용 단원을 참조하십시오.

20. 다음을 선택합니다.
21. 키에 대한 키 정책 문을 검토합니다. 키 정책을 변경하려면 편집을 선택하세요.
22. 다음을 선택합니다.
23. 선택한 키 설정을 검토합니다. 여전히 돌아가서 모든 설정을 변경할 수 있습니다.
24. 완료했으면 마침(Finish)을 선택하여 키를 생성합니다.

프로시저가 성공하면 선택한 키 스토어에 새 KMS AWS CloudHSM 키가 표시됩니다. 새 KMS 키의 이름 또는 별칭을 선택하면 세부 정보 페이지의 암호화 구성 탭에 KMS 키의 오리진(AWS CloudHSM),

사용자 지정 키 스토어의 이름, ID 및 유형, AWS CloudHSM 클러스터의 ID가 표시됩니다. 이 절차가 실패하면 실패 원인을 설명하는 오류 메시지가 나타납니다.

> **Tip**
>
> 사용자 지정 키 스토어에서 KMS 키를 손쉽게 식별하려면 고객 관리형 키(Customer managed keys) 페이지에서 사용자 지정 키 스토어 ID(Custom key store ID) 열을 화면에 추가합니다. 오른쪽 상단의 기어 아이콘을 클릭하고 Custom key store ID(사용자 지정 키 스토어 ID)를 선택합니다. 자세한 내용은 콘솔 보기 사용자 지정을 참조하세요.

AWS KMS API 사용

키 스토어에서 새 AWS KMS key (KMS 키)를 AWS CloudHSM 생성하려면 CreateKey 작업을 사용합니다. `CustomKeyStoreId` 파라미터를 사용하여 사용자 지정 키 스토어를 식별하고 `AWS_CLOUDHSM`의 `Origin` 값을 지정합니다.

`Policy` 파라미터를 사용해 키 정책을 지정할 수도 있습니다. 언제든 키 정책(PutKeyPolicy)을 변경하고 설명 및 태그 등의 선택적 요소를 추가할 수 있습니다.

이 섹션의 예제는 AWS Command Line Interface (AWS CLI)를 사용하지만, 지원되는 모든 프로그래밍 언어를 사용할 수 있습니다.

다음 예제는 DescribeCustomKeyStores 작업을 호출하여 AWS CloudHSM 키 스토어가 연결된 AWS CloudHSM 클러스터에 연결되어 있는지 확인하는 것으로 시작합니다. 기본적으로 이 작업은 계정 및 리전에서 모든 사용자 지정 키 스토어를 반환합니다. 특정 AWS CloudHSM 키 스토어만 설명하려면 `CustomKeyStoreId` 또는 `CustomKeyStoreName` 파라미터(둘 다 아님)를 사용합니다.

이 명령을 실행하기 앞서 예제에 나온 사용자 지정 키 스토어 ID를 유효한 ID로 대체합니다.

> **Note**
>
> `Description` 또는 `Tags` 필드에 기밀 또는 민감한 정보를 포함하지 마십시오. 이러한 필드는 CloudTrail 로그 및 기타 출력에 일반 텍스트로 표시될 수 있습니다.

```
$ aws kms describe-custom-key-stores --custom-key-store-id cks-1234567890abcdef0
{
```

```
   "CustomKeyStores": [
      "CustomKeyStoreId": "cks-1234567890abcdef0",
      "CustomKeyStoreName": "ExampleKeyStore",
      "CustomKeyStoreType": "AWS CloudHSM key store",
      "CloudHsmClusterId": "cluster-1a23b4cdefg",
      "TrustAnchorCertificate": "<certificate string appears here>",
      "CreationDate": "1.499288695918E9",
      "ConnectionState": "CONNECTED"
   ],
}
```

다음 예제 명령은 DescribeClusters 작업을 사용하여 ExampleKeyStore (cluster-1a23b4cdefg)와 연결된 AWS CloudHSM 클러스터에 활성 HSMs이 두 개 이상 있는지 확인합니다. 클러스터의 HSM 수가 2개 미만이면 CreateKey 작업이 실패합니다.

```
$ aws cloudhsmv2 describe-clusters
{
    "Clusters": [
        {
            "SubnetMapping": {
               ...
            },
            "CreateTimestamp": 1507133412.351,
            "ClusterId": "cluster-1a23b4cdefg",
            "SecurityGroup": "sg-865af2fb",
            "HsmType": "hsm1.medium",
            "VpcId": "vpc-1a2b3c4d",
            "BackupPolicy": "DEFAULT",
            "Certificates": {
                "ClusterCertificate": "-----BEGIN CERTIFICATE-----\...\n-----END
 CERTIFICATE-----\n"
            },
            "Hsms": [
                {
                    "AvailabilityZone": "us-west-2a",
                    "EniIp": "10.0.1.11",
                    "ClusterId": "cluster-1a23b4cdefg",
                    "EniId": "eni-ea8647e1",
                    "StateMessage": "HSM created.",
                    "SubnetId": "subnet-a6b10bd1",
                    "HsmId": "hsm-abcdefghijk",
                    "State": "ACTIVE"
                },
```

```
                {
                    "AvailabilityZone": "us-west-2b",
                    "EniIp": "10.0.0.2",
                    "ClusterId": "cluster-1a23b4cdefg",
                    "EniId": "eni-ea8647e1",
                    "StateMessage": "HSM created.",
                    "SubnetId": "subnet-b6b10bd2",
                    "HsmId": "hsm-zyxwvutsrqp",
                    "State": "ACTIVE"
                },
            ],
            "State": "ACTIVE"
        }
    ]
}
```

이 예제 명령은 CreateKey 작업을 사용하여 키 스토어에 KMS AWS CloudHSM 키를 생성합니다. AWS CloudHSM 키 스토어에서 KMS 키를 생성하려면 키 스토어의 사용자 지정 AWS CloudHSM 키 스토어 ID를 제공하고 `Origin` 값을 지정해야 합니다`AWS_CLOUDHSM`.

응답에는 사용자 지정 키 스토어 및 AWS CloudHSM 클러스터의 IDs가 포함됩니다.

이 명령을 실행하기 앞서 예제에 나온 사용자 지정 키 스토어 ID를 유효한 ID로 대체합니다.

```
$ aws kms create-key --origin AWS_CLOUDHSM --custom-key-store-id cks-1234567890abcdef0
{
  "KeyMetadata": {
    "AWSAccountId": "111122223333",
    "Arn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "CreationDate": 1.499288695918E9,
    "Description": "Example key",
    "Enabled": true,
    "MultiRegion": false,
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyManager": "CUSTOMER",
    "KeyState": "Enabled",
    "KeyUsage": "ENCRYPT_DECRYPT",
    "Origin": "AWS_CLOUDHSM"
    "CloudHsmClusterId": "cluster-1a23b4cdefg",
    "CustomKeyStoreId": "cks-1234567890abcdef0"
    "KeySpec": "SYMMETRIC_DEFAULT",
    "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
```

```
    "EncryptionAlgorithms": [
        "SYMMETRIC_DEFAULT"
    ]
  }
}
```

외부 키 저장소에서 KMS 키 생성

외부 키 스토어를 생성하고 연결한 후 키 스토어 AWS KMS keys 에서를 생성할 수 있습니다. 오리진 값이 External key store(외부 키 스토어)(EXTERNAL_KEY_STORE)인 대칭 암호화 KMS 키여야 합니다. 사용자 지정 키 스토어에 비대칭 KMS 키, HMAC KMS 키 또는 가져온 키 구성 요소가 있는 KMS 키를 생성할 수 없습니다. 또한 사용자 지정 키 스토어에서 대칭 암호화 KMS 키를 사용하여 비대칭 데이터 키 페어를 생성할 수 없습니다.

외부 키 스토어의 KMS 키는 AWS외부에 있는 구성 요소에 의존하기 때문에 표준 KMS 키보다 지연 시간, 내구성 및 가용성이 낮을 수 있습니다. 외부 키 스토어에서 KMS 키를 생성하거나 사용하기 전에 외부 키 스토어 속성이 있는 키가 필요한지 확인합니다.

> **Note**
>
> 일부 외부 키 관리자는 외부 키 스토어에 KMS 키를 생성하는 더 간단한 방법을 제공합니다. 자세한 내용은 외부 키 관리자 설명서를 참조하세요.

외부 키 스토어에서 KMS 키를 생성하려면 다음을 지정하세요.

- 외부 키 스토어의 ID.
- 외부 키 스토어(EXTERNAL_KEY_STORE)의 키 구성 요소 오리진.
- 외부 키 스토어와 연결된 외부 키 관리자에 있는 기존 외부 키의 ID. 이 외부 키는 KMS 키의 키 구성 요소 역할을 합니다. KMS 키를 생성한 후에는 외부 키 ID를 변경할 수 없습니다.

  AWS KMS 는 암호화 및 복호화 작업에 대한 요청에서 외부 키 스토어 프록시에 외부 키 ID를 제공합니다.는 외부 키 관리자 또는 해당 암호화 키에 직접 액세스할 AWS KMS 수 없습니다.

외부 키 스토어의 KMS 키에는 외부 키 외에도 AWS KMS 키 구성 요소가 있습니다. KMS 키로 암호화된 모든 데이터는 먼저 키의 AWS KMS 키 구성 요소를 AWS KMS 사용하여에서 암호화된 다음 외부 키를 사용하여 외부 키 관리자가 암호화합니다. 이 이중 암호화 프로세스를 통해 외부 키 스토어에서

KMS 키로 보호되는 사이퍼텍스트가 항상 AWS KMS으로만 보호되는 사이퍼텍스트 이상으로 강력해집니다. 자세한 내용은 외부 키 스토어 작동 방식을 참조하세요.

CreateKey 작업이 성공하면 새 KMS 키의 키 상태는 Enabled입니다. 외부 키 저장소에서 KMS 키를 볼 때 키 ID, 키 사양, 키 사용, 키 상태 및 생성 날짜와 같은 일반적인 속성을 볼 수 있습니다. 그러나 외부 키 스토어의 ID 및 연결 상태와 외부 키의 ID도 볼 수 있습니다.

외부 키 스토어에서 KMS 키를 생성하려는 시도가 실패할 때 오류 메시지를 사용하면 원인을 식별하는 데 도움이 됩니다. 외부 키 스토어가 연결되어 있지 않거나 (CustomKeyStoreInvalidStateException), 외부 키 스토어 프록시가 지정된 외부 키 ID를 가진 외부 키를 찾을 수 없거나(XksKeyNotFoundException), 외부 키가 동일한 외부 키 스토어 XksKeyAlreadyInUseException의 KMS 키와 이미 연결되어 있음을 나타낼 수 있습니다.

외부 키 스토어에서 KMS 키를 생성하는 작업 AWS CloudTrail 로그의 예는 섹션을 참조하세요CreateKey.

주제

- 외부 키 스토어의 KMS 키 요구 사항
- 외부 키 저장소에서 새로운 KMS 키 생성

외부 키 스토어의 KMS 키 요구 사항

외부 키 스토어에서 KMS 키를 생성하려면 외부 키 스토어, KMS 키 및 KMS 키의 외부 암호화 키 구성 요소 역할을 하는 외부 키에 대해 다음 속성이 필요합니다.

외부 키 스토어 요구 사항

- 외부 키 스토어 프록시에 연결해야 합니다.

  외부 키 스토어의 연결 상태를 보려면 외부 키 저장소 보기 섹션을 참조하세요. 외부 키 스토어를 연결하려면 외부 키 저장소 연결 및 연결 해제 섹션을 참조하세요.

KMS 키 요구 사항

KMS 키를 생성한 후에는 이러한 속성을 변경할 수 없습니다.

- 키 사양: SYMMETRIC_DEFAULT
- 키 사용: ENCRYPT_DECRYPT

- 키 구성 요소 오리진: EXTERNAL_KEY_STORE
- 다중 리전: FALSE

외부 키 요구 사항

- 256비트 AES 암호화 키(256개의 임의 비트). 외부 키의 KeySpec이 AES_256이어야 합니다.
- 활성화되어 사용할 수 있습니다. 외부 키의 Status가 ENABLED여야 합니다.
- 암호화 및 암호 복호화용으로 구성되었습니다. 외부 키의 KeyUsage에 ENCRYPT와 DECRYPT가 포함되어야 합니다.
- 이 KMS 키에만 사용됩니다. 외부 키 스토어의 각 KMS key는 서로 다른 외부 키와 연결되어야 합니다.

  AWS KMS 또한 에서는 외부 키를 외부 키 스토어에만 사용할 것을 권장합니다. 이 제한으로 인해 키와 관련된 문제를 더 쉽게 식별하고 해결할 수 있습니다.
- 외부 키 스토어에 대한 외부 키 스토어 프록시에서 액세스할 수 있습니다.

  외부 키 스토어 프록시가 지정된 외부 키 ID를 사용하여 키를 찾을 수 없는 경우 CreateKey 작업이 실패합니다.
- 에서 AWS 서비스 생성하는 예상 트래픽을 처리할 수 있습니다.는 초당 최대 1,800개의 요청을 처리할 수 있도록 외부 키를 준비할 것을 AWS KMS 권장합니다.

외부 키 저장소에서 새로운 KMS 키 생성

AWS KMS 콘솔의 외부 키 스토어에서 또는 CreateKey 작업을 사용하여 새 KMS 키를 생성할 수 있습니다.

AWS KMS 콘솔 사용

외부 키 스토어에서 KMS 키를 생성하는 방법에는 두 가지가 있습니다.

- 방법 1(권장): 외부 키 스토어를 선택한 다음 해당 외부 키 스토어에서 KMS 키를 생성합니다.
- 방법 2: KMS 키를 생성한 다음 KMS 키가 외부 키 스토어에 있음을 표시합니다.

키를 생성하기 전에 외부 키 스토어를 선택하는 방법 1을 사용하는 경우는 필요한 모든 KMS 키 속성을 AWS KMS 선택하고 외부 키 스토어의 ID를 채웁니다. 이 방법을 사용하면 KMS 키를 생성할 때 발생할 수 있는 오류를 방지할 수 있습니다.

> **Note**
>
> 별칭, 설명 또는 태그에 기밀 또는 민감한 정보를 포함하지 마십시오. 이러한 필드는 CloudTrail 로그 및 기타 출력에 일반 텍스트로 표시될 수 있습니다.

방법 1(권장): 외부 키 스토어에서 시작

이 방법을 사용하려면 외부 키 스토어를 선택한 다음 KMS 키를 생성합니다. AWS KMS 콘솔은 필요한 모든 속성을 선택하고 외부 키 스토어의 ID를 채웁니다. 이 방법을 사용하면 KMS 키를 생성할 때 발생할 수 있는 많은 오류를 방지할 수 있습니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 탐색 창에서 Custom key stores(사용자 지정 키 스토어), External key stores(외부 키 스토어)를 선택합니다.
4. 외부 키 스토어의 이름을 선택합니다.
5. 오른쪽 상단에서 Create a KMS key in this key store(이 키 스토어에서 KMS 키 생성)을 선택합니다.

   외부 키 스토어가 연결되지 않은 경우 외부 키 스토어를 연결하라는 메시지가 나타납니다. 연결 시도가 실패할 경우 문제를 해결하고 외부 키 스토어를 연결해야 외부 키 스토어에서 새 KMS 키를 생성할 수 있습니다.

   외부 키 스토어가 연결된 경우 키 생성을 위한 Customer managed keys(고객 관리형 키) 페이지로 리디렉션됩니다. 필요한 Key configuration(키 구성) 값이 이미 선택되어 있습니다. 외부 키 스토어의 사용자 지정 키 스토어 ID도 입력되어 있지만 사용자가 이를 변경할 수 있습니다.
6. 외부 키 관리자에서 외부 키의 키 ID를 입력합니다. 이 외부 키는 KMS 키와 함께 사용하기 위한 요구 사항을 충족해야 합니다. 키가 생성된 후에는 이 값을 변경할 수 없습니다.

   외부 키에 ID가 여러 개 있는 경우 외부 키 스토어 프록시가 외부 키를 식별하는 데 사용하는 키 ID를 입력합니다.
7. 지정된 외부 키 스토어에서 KMS 키를 생성할지 확인합니다.
8. 다음을 선택합니다.

   이 절차의 나머지 부분은 표준 KMS 키를 생성하는 것과 동일합니다.

9. KMS 키의 별칭(필수)과 설명(선택 사항)을 입력합니다.
10. (선택 사항). 태그 추가(Add Tags) 페이지에서 KMS 키를 식별 및 분류하는 태그를 추가합니다.

    AWS 리소스에 태그를 추가하면가 태그별로 집계된 사용량 및 비용이 포함된 비용 할당 보고서를 AWS 생성합니다. KMS 키에 대한 액세스를 제어하는 데에도 태그를 사용할 수 있습니다. KMS 키 태그 지정에 대한 자세한 내용은 의 태그 AWS KMS 및 에 대한 ABAC AWS KMS 섹션을 참조하십시오.
11. 다음을 선택합니다.
12. 키 관리자(Key Administrators) 섹션에서 KMS 키를 관리할 수 있는 IAM 사용자 및 역할을 선택합니다. 자세한 내용은 키 관리자가 KMS 키를 관리하도록 허용을 참조하세요.

    > **Note**
    >
    > IAM 정책은 다른 IAM 사용자 및 역할에 KMS 키 사용 권한을 제공할 수 있습니다. IAM 모범 사례는 장기 보안 인증 정보가 있는 IAM 사용자의 사용을 장려하지 않습니다. 가능할 경우, 임시 보안 인증 정보를 제공하는 IAM 역할을 사용하세요. 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.
13. (선택 사항) 이러한 키 관리자가 KMS 키를 삭제하지 못하도록 하려면 Allow key administrators to delete this key(키 관리자가 이 키를 삭제하도록 허용) 확인란의 선택을 취소합니다.

    KMS 키 삭제는 사이퍼텍스트를 복구할 수 없게 만들 수 있는 파괴적이고 되돌릴 수 없는 작업입니다. 외부 키 구성 요소가 있더라도 외부 키 스토어에서 대칭 KMS 키를 다시 생성할 수 없습니다. 그러나 KMS 키를 삭제해도 연결된 외부 키에는 영향이 없습니다. 외부 키 저장소에서 KMS 키를 삭제하는 방법에 대한 자세한 내용은 키 삭제에 대한 특별 고려 사항 섹션을 참조하세요.
14. 다음을 선택합니다.
15. 이 계정 섹션에서 암호화 작업에 KMS 키를 사용할 수 AWS 계정 있는 IAM 사용자 및 역할을 선택합니다. 자세한 내용은 키 사용자가 KMS 키를 사용하도록 허용을 참조하세요.

    > **Note**
    >
    > IAM 정책은 다른 IAM 사용자 및 역할에 KMS 키 사용 권한을 제공할 수 있습니다. IAM 모범 사례는 장기 보안 인증 정보가 있는 IAM 사용자의 사용을 장려하지 않습니다. 가능할 경우, 임시 보안 인증 정보를 제공하는 IAM 역할을 사용하세요. 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.

16. (선택 사항) 다른 사용자가 암호화 작업에이 KMS 키를 AWS 계정 사용하도록 허용할 수 있습니다. 이렇게 하려면 페이지 하단의 기타 AWS 계정 섹션에서 다른 추가 AWS 계정를 선택하고 외부 계정의 AWS 계정 ID를 입력합니다. 외부 계정을 여러 개 추가하려면 이 단계를 반복합니다.

    > **Note**
    >
    > 또한 다른의 관리자는 사용자에 대한 IAM 정책을 생성하여 KMS 키에 대한 액세스를 허용 AWS 계정 해야 합니다. 자세한 내용은 다른 계정의 사용자가 KMS를 사용하도록 허용 단원을 참조하십시오.

17. 다음을 선택합니다.
18. 선택한 키 설정을 검토합니다. 여전히 돌아가서 모든 설정을 변경할 수 있습니다.
19. 완료했으면 마침(Finish)을 선택하여 키를 생성합니다.

방법 2: 고객 관리형 키에서 시작

이 절차는 AWS KMS 키 구성 요소가 있는 대칭 암호화 키를 생성하는 절차와 동일합니다. 그러나 이 절차에서는 외부 키 스토어의 사용자 지정 키 스토어 ID와 외부 키의 키 ID를 지정합니다. 또한 외부 키 스토어의 KMS 키에 필요한 속성 값(예: 키 사양 및 키 사용)을 지정해야 합니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 탐색 창에서 고객 관리형 키를 선택합니다.
4. 키 생성(Create key)을 선택합니다.
5. 대칭(Symmetric)을 선택합니다.
6. 키 사용(Key usage)에서 암호화 및 해독(Encrypt and decrypt) 옵션이 선택됩니다. 변경할 수 없습니다.
7. 고급 옵션을 선택합니다.
8. Key material origin(키 구성 요소 오리진)에서 External key store(외부 키 스토어)를 선택합니다.
9. 지정된 외부 키 스토어에서 KMS 키를 생성할지 확인합니다.
10. 다음을 선택합니다.
11. 새 KMS 키에 대한 외부 키 스토어를 나타내는 행을 선택합니다.

연결 해제된 외부 키 스토어는 선택할 수 없습니다. 연결 해제된 키 스토어를 연결하려면 키 스토어 이름을 선택한 다음 Key store actions(키 스토어 작업)에서 Connect(연결)를 선택합니다. 자세한 내용은 AWS KMS 콘솔 사용을 참조하세요.

12. 외부 키 관리자에서 외부 키의 키 ID를 입력합니다. 이 외부 키는 KMS 키와 함께 사용하기 위한 요구 사항을 충족해야 합니다. 키가 생성된 후에는 이 값을 변경할 수 없습니다.

    외부 키에 ID가 여러 개 있는 경우 외부 키 스토어 프록시가 외부 키를 식별하는 데 사용하는 키 ID를 입력합니다.

13. 다음을 선택합니다.

    이 절차의 나머지 부분은 표준 KMS 키를 생성하는 것과 동일합니다.

14. KMS 키의 별칭과 설명을 입력합니다.

15. (선택 사항). 태그 추가(Add Tags) 페이지에서 KMS 키를 식별 및 분류하는 태그를 추가합니다.

    AWS 리소스에 태그를 추가하면가 태그별로 집계된 사용량 및 비용이 포함된 비용 할당 보고서를 AWS 생성합니다. KMS 키에 대한 액세스를 제어하는 데에도 태그를 사용할 수 있습니다. KMS 키 태그 지정에 대한 자세한 내용은 의 태그 AWS KMS 및 에 대한 ABAC AWS KMS 섹션을 참조하십시오.

16. 다음을 선택합니다.

17. 키 관리자(Key Administrators) 섹션에서 KMS 키를 관리할 수 있는 IAM 사용자 및 역할을 선택합니다. 자세한 내용은 키 관리자가 KMS 키를 관리하도록 허용을 참조하세요.

    > **Note**
    >
    > IAM 정책은 다른 IAM 사용자 및 역할에 KMS 키 사용 권한을 제공할 수 있습니다.

18. (선택 사항) 이러한 키 관리자가 KMS 키를 삭제하지 못하도록 하려면 Allow key administrators to delete this key(키 관리자가 이 키를 삭제하도록 허용) 확인란의 선택을 취소합니다.

    KMS 키 삭제는 사이퍼텍스트를 복구할 수 없게 만들 수 있는 파괴적이고 되돌릴 수 없는 작업입니다. 외부 키 구성 요소가 있더라도 외부 키 스토어에서 대칭 KMS 키를 다시 생성할 수 없습니다. 그러나 KMS 키를 삭제해도 연결된 외부 키에는 영향이 없습니다. 외부 키 스토어에서 KMS 키를 삭제하는 방법에 대한 자세한 내용은 삭제 AWS KMS key 섹션을 참조하세요.

19. 다음을 선택합니다.

20. 이 계정 섹션에서 암호화 작업에 KMS 키를 사용할 수 AWS 계정 있는 IAM 사용자 및 역할을 선택합니다. 자세한 내용은 키 사용자가 KMS 키를 사용하도록 허용을 참조하세요.

> **Note**
>
> IAM 정책은 다른 IAM 사용자 및 역할에 KMS 키 사용 권한을 제공할 수 있습니다.

21. (선택 사항) 다른 사용자가 암호화 작업에이 KMS 키를 AWS 계정 사용하도록 허용할 수 있습니다. 이렇게 하려면 페이지 하단의 기타 AWS 계정 섹션에서 다른 추가 AWS 계정를 선택하고 외부 계정의 AWS 계정 ID를 입력합니다. 외부 계정을 여러 개 추가하려면 이 단계를 반복합니다.

> **Note**
>
> 또한 다른의 관리자는 사용자에 대한 IAM 정책을 생성하여 KMS 키에 대한 액세스를 허용 AWS 계정 해야 합니다. 자세한 내용은 다른 계정의 사용자가 KMS를 사용하도록 허용 단원을 참조하십시오.

22. 다음을 선택합니다.
23. 선택한 키 설정을 검토합니다. 여전히 돌아가서 모든 설정을 변경할 수 있습니다.
24. 완료했으면 마침(Finish)을 선택하여 키를 생성합니다.

절차가 성공하면 화면에서 선택한 외부 키 스토어에 새 KMS 키가 표시됩니다. 새 KMS 키의 이름 또는 별칭을 선택하면 세부 정보 페이지의 Cryptographic configuration(암호화 구성) 탭에 KMS 키(External key store(외부 키 스토어))의 오리진, 사용자 지정 키 스토어의 이름, ID 및 유형, 외부 키의 ID, 키 사용 및 상태가 표시됩니다. 이 절차가 실패하면 실패 원인을 설명하는 오류 메시지가 나타납니다. 외부 키 스토어 문제 해결 섹션을 참조하세요.

> **Tip**
>
> 사용자 지정 키 스토어에서 KMS 키를 손쉽게 식별하려면 Customer managed keys(고객 관리형 키) 페이지에서 Origin(오리진) 및 Custom key store ID(사용자 지정 키 스토어 ID) 열을 디스플레이에 추가합니다. 테이블 필드를 변경하려면 페이지의 오른쪽 상단에서 기어 아이콘을 선택합니다. 자세한 내용은 콘솔 보기 사용자 지정을 참조하세요.

AWS KMS API 사용

외부 키 스토어에서 새 KMS 키를 새로 생성하려면 CreateKey 작업을 사용합니다. 다음 파라미터는 필수 파라미터입니다.

- `Origin` 값은 `EXTERNAL_KEY_STORE`여야 합니다.
- `CustomKeyStoreId` 파라미터는 외부 키 스토어를 식별합니다. 지정된 외부 키 스토어의 `ConnectionState`는 `CONNECTED`여야 합니다. `CustomKeyStoreId`와 `ConnectionState`를 찾으려면 `DescribeCustomKeyStores` 작업을 사용합니다.
- `XksKeyId` 파라미터는 외부 키를 식별합니다. 이 외부 키는 KMS 키와의 연결 요구 사항을 충족해야 합니다.

`Policy` 또는 `Tags` 파라미터 사용과 같이 `CreateKey` 작업의 선택적 파라미터를 사용할 수도 있습니다.

> **Note**
>
> `Description` 또는 `Tags` 필드에 기밀 또는 민감한 정보를 포함하지 마십시오. 이러한 필드는 CloudTrail 로그 및 기타 출력에 일반 텍스트로 표시될 수 있습니다.

이 섹션의 예제는 AWS Command Line Interface (AWS CLI)를 사용하지만, 지원되는 모든 프로그래밍 언어를 사용할 수 있습니다.

이 예제 명령은 `CreateKey` 작업을 사용해 KMS 키의 외부 키 스토어를 생성합니다. 응답에는 KMS 키의 속성, 외부 키 스토어의 ID, 외부 키의 ID, 사용량 및 상태가 포함됩니다.

이 명령을 실행하기 앞서 예제에 나온 사용자 지정 키 스토어 ID를 유효한 ID로 대체합니다.

```
$ aws kms create-key --origin EXTERNAL_KEY_STORE --custom-key-store-
id cks-1234567890abcdef0 --xks-key-id bb8562717f809024
{
  "KeyMetadata": {
    "Arn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "AWSAccountId": "111122223333",
    "CreationDate": "2022-12-02T07:48:55-07:00",
    "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
    "CustomKeyStoreId": "cks-1234567890abcdef0",
    "Description": "",
    "Enabled": true,
    "EncryptionAlgorithms": [
      "SYMMETRIC_DEFAULT"
    ],
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
```

```
    "KeyManager": "CUSTOMER",
    "KeySpec": "SYMMETRIC_DEFAULT",
    "KeyState": "Enabled",
    "KeyUsage": "ENCRYPT_DECRYPT",
    "MultiRegion": false,
    "Origin": "EXTERNAL_KEY_STORE",
    "XksKeyConfiguration": {
      "Id": "bb8562717f809024"
    }
  }
}
```

키 식별 및 보기

AWS Management Console 또는 AWS Key Management Service (AWS KMS) API를 사용하여 관리하는 KMS 키와 관리되는 KMS 키를 포함하여 각 계정 및 리전 AWS KMS keys 에서를 볼 수 있습니다 AWS.

주제

- 키 ID 및 키 ARN 찾기
- KMS 키 세부 정보 액세스 및 나열
- 다양한 키 유형 식별
- 콘솔 보기 사용자 지정
- 키 스토어에서 KMS 키 및 AWS CloudHSM 키 구성 요소 찾기

키 ID 및 키 ARN 찾기

를 식별하려면 키 ID 또는 Amazon 리소스 이름(키 ARN)을 사용할 AWS KMS key수 있습니다. 암호화 작업에서 별칭 이름이나 별칭 ARN을 사용할 수도 있습니다.

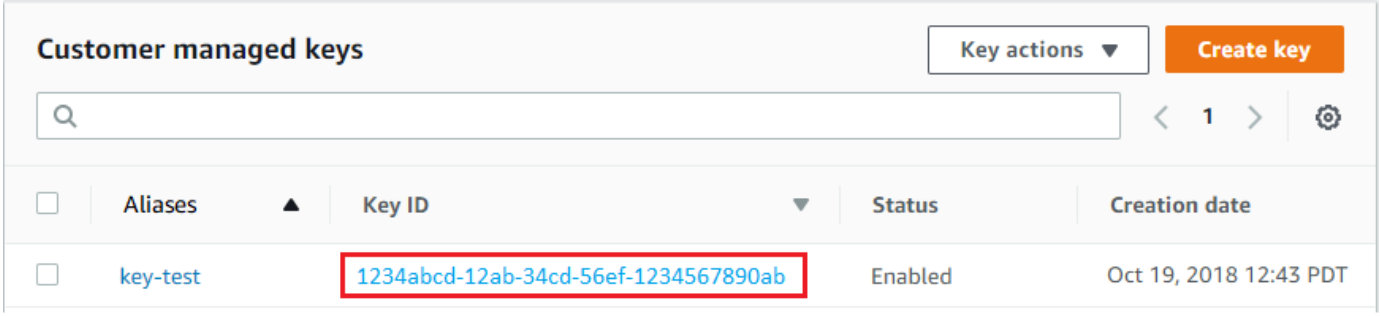
AWS KMS 콘솔에서 또는 ListKeys 작업을 사용하여 계정 및 리전에 있는 각 KMS 키의 키 ID와 키 ARN을 식별할 수 있습니다.

에서 지원하는 KMS 키 식별자에 대한 자세한 내용은 섹션을 AWS KMS참조하세요키 식별자(KeyId). 별칭 이름 및 별칭 ARN 찾는 방법에 대한 자세한 내용은 KMS 키의 별칭 이름 및 별칭 ARN 찾기 단원을 참조하십시오.

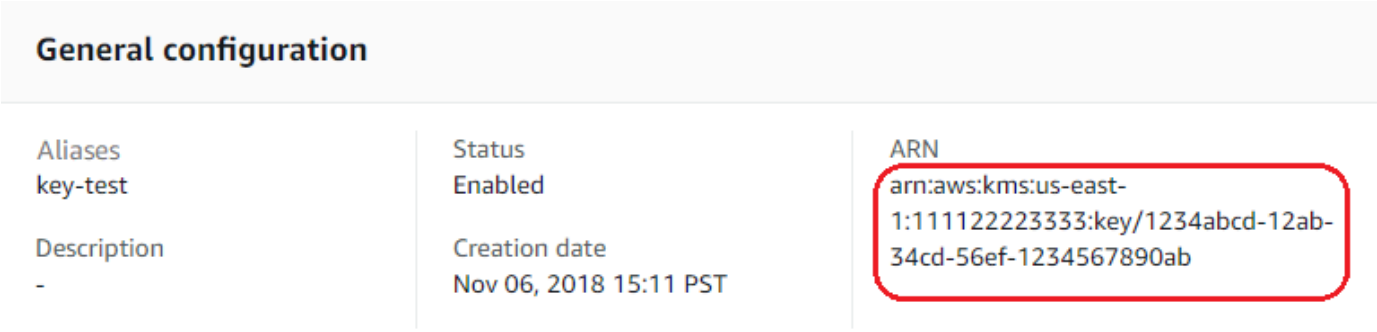
AWS KMS 콘솔 사용

1. https://console.aws.amazon.com/kms AWS KMS 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 해당 계정에서 직접 생성하고 관리하는 키를 보려면 탐색 창에서 고객 관리형 키를 선택합니다. 가 AWS 자동으로 생성하고 관리하는 계정의 키를 보려면 탐색 창에서 AWS 관리형 키를 선택합니다.
4. KMS 키의 키 ID를 찾으려면 KMS 키 별칭으로 시작하는 행을 확인합니다.

키 ID 열은 기본적으로 테이블에 표시됩니다. 키 ID 열이 테이블에 표시되지 않으면 the section called “콘솔 보기 사용자 지정” 단원에 설명된 절차에 따라 복원합니다. KMS 키의 키 ID는 세부 정보 페이지에서도 볼 수 있습니다.



5. KMS 키의 Amazon 리소스 이름(ARN)을 찾으려면 키 ID 또는 별칭을 선택합니다. 키 ARN은 일반 구성 섹션에 표시됩니다.



AWS KMS API 사용

의 키 ID와 키 ARN을 찾으려면 ListKeys 작업을 AWS KMS key사용합니다.

`ListKeys` 작업은 호출자의 계정 및 리전에 있는 모든 KMS 키의 ID와 Amazon 리소스 이름(ARN)을 반환합니다.

예를 들면, `ListKeys` 작업 호출은 이 가상 계정에 있는 각 KMS 키의 ID와 ARN을 반환합니다. 다양한 프로그래밍 언어의 예는 AWS SDK 또는 CLI와 `ListKeys` 함께 사용 섹션을 참조하십시오.

```
$ aws kms list-keys
{
    "Keys": [
        {
            "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
            "KeyArn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
```

```
        },
        {
            "KeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
            "KeyArn": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321"
        }
    ]
}
```

KMS 키 세부 정보 액세스 및 나열

AWS KMS 콘솔 또는 DescribeKey 작업을 사용하여 계정 및 리전의 KMS 키에 대한 상세한 정보에 액세스하고 나열할 수 있습니다.

다음 절차에서는 키 ID, 키 사양, 키 사용 등과 같은 KMS 키 세부 정보에 액세스하는 방법을 보여줍니다.

AWS KMS 콘솔 사용

각 KMS 키의 세부 정보 페이지에는 KMS 키의 속성이 표시됩니다. 세부 정보는 KMS 키의 종류에 따라 약간 다릅니다.

KMS 키에 대한 자세한 정보를 표시하려면 AWS 관리형 키 또는 고객 관리형 키 페이지에서 KMS 키의 별칭 또는 키 ID를 선택합니다.

KMS 키의 세부 정보 페이지에는 KMS 키의 기본 속성을 표시하는 일반 구성 섹션이 있습니다. 여기에는 키 정책, 암호화 구성, 태그, 키 구성 요소 및 교체(자동 또는 온디맨드 교체를 지원하는 KMS 키의 경우), 지역성(다중 리전 키의 경우), 퍼블릭 키(비대칭 KMS 키의 경우)처럼 KMS 키의 속성을 확인하고 수정할 수 있는 탭도 포함되어 있습니다.

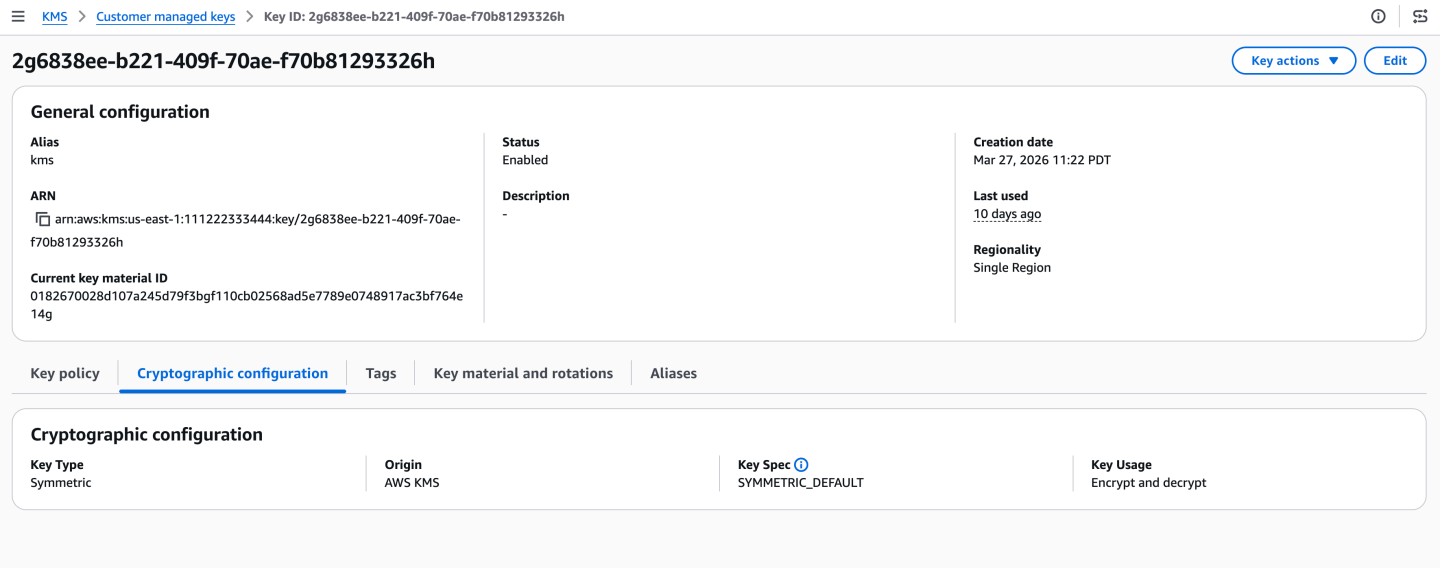
> **Note**
>
> AWS KMS 콘솔에는 계정 및 리전에서 볼 수 있는 권한이 있는 KMS 키가 표시됩니다. 다른의 KMS 키는 보기, 관리 및 사용할 권한이 있더라도 콘솔에 표시되지 AWS 계정 않습니다. 다른 계정의 KMS 키를 보려면 DescribeKey 작업을 사용합니다.

KMS 키의 키 세부 정보 페이지로 이동

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.

2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 해당 계정에서 직접 생성하고 관리하는 키를 보려면 탐색 창에서 고객 관리형 키를 선택합니다. 가 AWS 자동으로 생성하고 관리하는 계정의 키를 보려면 탐색 창에서 AWS 관리형 키를 선택합니다.
4. 키 세부 정보 페이지를 열려면 키 테이블에서 KMS 키의 키 ID 또는 별칭을 선택합니다.

   KMS 키에 여러 별칭이 있는 경우 별칭 요약(n개 이상)이 별칭 중 하나의 이름 옆에 나타납니다. 별칭 요약을 선택하면 키 세부 정보 페이지의 별칭 탭으로 바로 이동합니다.



다음 목록에서는 탭의 필드를 포함하여 세부 정보 표시의 필드에 대해 설명합니다. 이러한 필드 중 일부는 테이블 표시의 열로 사용할 수도 있습니다.

별칭

위치: 별칭 탭

KMS 키의 기억하기 쉬운 이름입니다. 별칭을 사용하여 콘솔 및 일부 AWS KMS APIs. 자세한 내용은 의 별칭 AWS KMS을 참조하세요.

별칭 탭에는 AWS 계정 및 리전의 KMS 키와 연결된 모든 별칭이 표시됩니다.

ARN

위치: 일반 구성 섹션

KMS 키의 Amazon 리소스 이름(ARN)입니다. 이 값은 KMS 키를 고유하게 식별합니다. AWS KMS API 작업에서 KMS 키를 식별하는 데 사용할 수 있습니다.

연결 상태

사용자 지정 키 스토어가 백업 키 스토어에 연결되었는지 여부를 나타냅니다. 이 필드는 KMS 키가 사용자 지정 키 스토어에서 생성된 경우에만 나타납니다.

이 필드의 값에 대한 자세한 내용은AWS KMS API 참조의 ConnectionState를 참조하세요.

Creation(생성) 날짜

위치: 일반 구성 섹션

KMS 키를 생성한 날짜와 시간입니다. 이 값은 디바이스의 로컬 시간으로 표시됩니다. 시간대는 리전에 의존하지 않습니다.

만료와 달리 생성은 키 구성 요소가 아니라 KMS 키만 참조합니다.

CloudHSM 클러스터 ID

위치: 암호화 구성 탭

KMS 키의 키 구성 요소가 포함된 클러스터의 AWS CloudHSM 클러스터 ID입니다. 이 필드는 KMS 키가 사용자 지정 키 스토어에서 생성된 경우에만 나타납니다.

CloudHSM 클러스터 ID를 선택하면 AWS CloudHSM 콘솔에서 클러스터 페이지가 열립니다.

현재 키 구성 요소

위치: 일반 구성 섹션

오리진이 AWS_KMS인 대칭 암호화 키는 자동 및 온디맨드 교체를 모두 지원합니다. EXTERNAL 오리진이 있는 대칭 암호화 키는 온디맨드 교체를 지원합니다. 이러한 키에는 다중 키 구성 요소가 있을 수 있습니다. 가장 최근에 교체된 키 구성 요소는 암호화와 복호화에 모두 사용할 수 있습니다. 이 키 구성 요소는 현재 키 구성 요소로 식별됩니다. 나머지 키 구성 요소는 복호화에만 사용할 수 있습니다. KMS 키의 자동 또는 온디맨드 교체는 현재 키 구성 요소를 변경합니다.

사용자 지정 키 스토어 ID

위치: 암호화 구성 탭

KMS 키가 포함된 사용자 지정 키 스토어의 ID입니다. 이 필드는 KMS 키가 사용자 지정 키 스토어에서 생성된 경우에만 나타납니다.

사용자 지정 키 스토어 ID를 선택하면 AWS KMS 콘솔에서 사용자 지정 키 스토어 페이지가 열립니다.

사용자 지정 키 스토어 이름

위치: 암호화 구성 탭

KMS 키가 포함된 사용자 지정 키 스토어의 이름입니다. 이 필드는 KMS 키가 사용자 지정 키 스토어에서 생성된 경우에만 나타납니다.

사용자 지정 키 스토어 유형

위치: 암호화 구성 탭

사용자 지정 키 스토어가 AWS CloudHSM 키 스토어인지 외부 키 스토어인지를 나타냅니다. 이 필드는 KMS 키가 사용자 지정 키 스토어에서 생성된 경우에만 나타납니다.

설명

위치: 일반 구성 섹션

작성 및 편집할 수 있는 KMS 키에 대한 간략한 설명(선택 사항)입니다. 고객 관리형 키에 대한 설명을 추가하거나 업데이트하려면 일반 구성(General Configuration) 위에서 편집(Edit)을 선택합니다.

암호화 알고리즘

위치: 암호화 구성 탭

AWS KMS의 KMS 키와 함께 사용할 수 있는 암호화 알고리즘을 나열합니다. 이 필드는 키 유형이 비대칭이고 키 사용이 암호화 및 암호 해독인 경우에만 표시됩니다. 에서 AWS KMS 지원하는 암호화 알고리즘에 대한 자세한 내용은 SYMMETRIC_DEFAULT 키 사양 및 섹션을 참조하세요암호화 및 해독을 위한 RSA 키 사양.

만료 날짜

위치: 키 구성 요소 탭

KMS 키의 키 구성 요소가 만료되는 날짜 및 시간입니다. 이 필드는 가져온 키 구성 요소가 있는 KMS 키, 즉 오리진(Origin)이 외부(External)이고 KMS 키에 만료되는 키 구성 요소가 있는 경우에만 표시됩니다. 대칭 암호화 키에는 여러 키 구성 요소가 연결될 수 있습니다. 이러한 키의 경우 이 필드는 연결된 키 구성 요소 중 하나가 만료되는 가장 이른 날짜와 시간을 나타냅니다.

외부 키 ID

위치: 암호화 구성 탭

외부 키 스토어의 KMS 키와 연결된 외부 키의 ID입니다. 이 필드는 외부 키 스토어의 KMS 키에만 표시됩니다.

외부 키 상태

위치: 암호화 구성 탭

외부 키 스토어 프록시가 KMS 키와 연결된 외부 키에 대해 보고한 가장 최근 상태입니다. 이 필드는 외부 키 스토어의 KMS 키에만 표시됩니다.

외부 키 사용

위치: 암호화 구성 탭

KMS 키와 연결된 외부 키에서 활성화된 암호화 작업입니다. 이 필드는 외부 키 스토어의 KMS 키에만 표시됩니다.

마지막으로 사용됨

위치: 일반 구성 섹션

KMS 키가 암호화 작업에 사용된 마지막 시간으로, 5일 전과 같은 상대 값으로 표시됩니다. 이 값은 이 키에 대한 사용량 추적을 AWS KMS 시작한 이후의 사용량만 포함합니다. 값 위로 마우스를 가져가면 정확한 타임스탬프, 사용량 추적을 AWS KMS 시작한 날짜, 수행된 AWS KMS 작업 및 관련 AWS CloudTrail 이벤트에 대한 링크를 볼 수 있습니다.

이 필드는 암호화 작업만 추적합니다. DescribeKey 또는 GetKeyPolicy와 같은 비암호화 작업은 추적되지 않습니다.

키 정책

위치: 키 정책 탭

IAM 정책 및 권한 부여와 함께 KMS 키에 대한 액세스를 제어합니다. 모든 KMS 키에는 하나의 키 정책이 있습니다. 이것이 유일한 필수 권한 부여 요소입니다. 고객 관리형 키의 키 정책을 변경하려면 키 정책(Key policy) 탭에서 편집(Edit)을 선택합니다. 자세한 내용은 the section called “키 정책”을 참조하세요.

키 구성 요소 및 교체

위치: 키 구성 요소 및 교체 탭

이 탭은 (자동 교체와 온디맨드 교체를 모두 지원하는) 대칭 암호화 키 중 오리진이 `AWS_KMS`인 경우와 (온디맨드 교체를 지원하는) 단일 리전 대칭 암호화 키 중 오리진이 `EXTERNAL`인 경우에만 표시됩니다.

탭에는 세 개의 패널이 있습니다.

자동 교체: 고객 관리형 KMS 키에서 키 구성 요소의 자동 교체를 활성화 및 비활성화합니다. 고객 관리형 키의 키 교체 상태를 변경하려면 확인란을 사용합니다. AWS 관리형 키에서 키 구성 요소의 교체를 활성화하거나 비활성화할 수 없습니다. AWS 관리형 키 는 매년 자동으로 교체됩니다.

온디맨드 교체: 고객 관리형 키의 키 구성 요소에 대한 온디맨드 교체를 시작할 수 있습니다. 가져온 키의 경우 지금 교체 옵션을 사용하려면 가져온 키 구성 요소가 이미 `PENDING_ROTATION` 상태여야 합니다.

키 구성 요소: KMS 키와 연관된 모든 키 구성 요소가 나열됩니다. 각 키 구성 요소에는 고유 식별자가 있으며, 해당 행에는 KMS에서 사용 가능해진 교체 날짜 등 키 구성 요소에 대한 추가 정보가 표시됩니다. 가져온 키의 경우 각 행에는 해당 키 구성 요소를 삭제하거나 KMS 키로 다시 가져오는 데 사용할 수 있는 작업 메뉴도 있습니다.

키 사양

위치: 암호화 구성 탭

KMS 키의 키 구성 요소 유형입니다.는 대칭 암호화 KMS 키(SYMMETRIC_DEFAULT), 길이가 다른 HMAC KMS 키, 길이가 다른 RSA 키에 대한 KMS 키, 곡선이 다른 타원 곡선 키를 AWS KMS 지원합니다. 자세한 내용은 Key spec을 참조하세요.

키 유형

위치: 암호화 구성 탭

KMS 키가 대칭 또는 비대칭인지 여부를 나타냅니다.

키 사용

위치: 암호화 구성 탭

KMS 키를 암호화 및 해독(Encrypt and decrypt), 서명 및 확인(Sign and verify) 또는 MAC 생성 및 확인(Generate and verify MAC)에 사용할 수 있는지 여부를 나타냅니다. 자세한 내용은 Key usage을 참조하세요.

오리진(Origin)

위치: 암호화 구성 탭

KMS 키에 대한 키 구성 요소의 출처입니다. 유효값은 다음과 같습니다.

- AWS KMS 가 생성하는 키 구성 요소의 경우 AWS KMS
- AWS CloudHSM 키 스토어의 KMS 키의 경우 AWS CloudHSM
- 가져온 키 구성 요소의 경우 External(외부)(BYOK)
- 외부 키 스토어의 KMS 키의 경우 External key store(외부 키 스토어)

MAC 알고리즘

위치: 암호화 구성 탭

AWS KMS의 HMAC KMS 키와 함께 사용할 수 있는 MAC 알고리즘을 나열합니다. 이 필드는 키 사양(Key spec)이 HMAC 키 사양(HMAC_*)인 경우에만 나타납니다. AWS KMS 가 지원하는 MAC 알고리즘에 대한 자세한 내용은 HMAC KMS 키의 키 사양 섹션을 참조하세요.

프라이머리 키

위치: 리전 구분 탭

이 KMS 키가 다중 리전 기본 키임을 나타냅니다. 권한이 있는 사용자는 이 섹션을 사용하여 다른 관련 다중 리전 키로 기본 키를 변경할 수 있습니다. 이 필드는 KMS 키가 다중 리전 기본 키인 경우에만 나타납니다.

퍼블릭 키

위치: 공개 키 탭

비대칭 KMS 키의 퍼블릭 키를 표시합니다. 권한이 부여된 사용자는 이 탭을 사용하여 퍼블릭 키를 복사하고 다운로드할 수 있습니다.

분류: 리전 구분

위치: 일반 구성 섹션 및 리전 구분(Regionality) 탭

KMS 키가 단일 리전 키, 다중 리전 기본 키 또는 다중 리전 복제본 키인지 여부를 나타냅니다. 이 필드는 KMS 키가 다중 리전 키인 경우에만 나타납니다.

관련 다중 리전 키

위치: 리전 구분 탭

현재 KMS 키를 제외한 모든 관련 다중 리전 기본 및 복제본 키를 표시합니다. 이 필드는 KMS 키가 다중 리전 키인 경우에만 나타납니다.

기본 키의 관련 다중 리전 키 섹션에서 권한 부여된 사용자는 새 복제본 키를 생성할 수 있습니다.

복제본 키

위치: 리전 구분 탭

이 KMS 키가 다중 리전 복제본 키임을 나타냅니다. 이 필드는 KMS 키가 다중 리전 복제본 키인 경우에만 나타납니다.

서명 알고리즘

위치: 암호화 구성 탭

AWS KMS의 KMS 키와 함께 사용할 수 있는 서명 알고리즘을 나열합니다. 이 필드는 키 유형이 비대칭이고 키 사용이 서명 및 확인인 경우에만 표시됩니다. 에서 AWS KMS 지원하는 서명 알고리즘에 대한 자세한 내용은 서명 및 확인을 위한 RSA 키 사양 및 단원을 참조하십시오타원 곡선 키 사양.

Status

위치: 일반 구성 섹션

KMS 키의 키 상태입니다. KMS 키는 상태가 활성화됨(Enabled)인 경우에만 암호화 작업(cryptographic operations)에 사용할 수 있습니다. 각 KMS 키 상태와 KMS 키에서 실행할 수 있는 작업에 미치는 영향에 대한 자세한 설명은 키의 AWS KMS 키 상태 섹션을 참조하십시오.

Tags

위치: 태그 탭

KMS 키를 설명하는 키-값 페어(선택 사항)입니다. KMS 키의 태그를 추가하거나 변경하려면 태그 탭에서 편집을 선택합니다.

AWS 리소스에 태그를 추가하면는 태그별로 집계된 사용량 및 비용이 포함된 비용 할당 보고서를 AWS 생성합니다. KMS 키에 대한 액세스를 제어하는 데에도 태그를 사용할 수 있습니다. KMS 키 태그 지정에 대한 자세한 내용은 의 태그 AWS KMS 및 에 대한 ABAC AWS KMS 섹션을 참조하십시오.

AWS KMS API 사용

DescribeKey 작업은 지정된 KMS 키에 관한 세부 정보를 반환합니다. KMS 키를 식별하려면 키 ID, 키 ARN, 별칭 이름 또는 별칭 ARN을 사용합니다.

호출자의 계정과 리전에 있는 KMS 키만 표시하는 ListKeys 작업과 달리, 권한이 있는 사용자는 `DescribeKey` 작업을 사용하여 다른 계정의 KMS 키에 대한 세부 정보를 얻을 수 있습니다.

> **Note**
>
> DescribeKey 응답에는 KeySpec 및 CustomerMasterKeySpec 멤버가 모두 포함됩니다. CustomerMasterKeySpec 멤버는 더 이상 사용되지 않습니다.

예를 들면, DescribeKey 호출은 대칭 암호화 KMS 키에 관한 정보를 반환합니다. 응답에 포함되는 필드는 AWS KMS key 사양, 키 상태 및 키 구성 요소 오리진(key material origin)에 따라 달라집니다. 다양한 프로그래밍 언어의 예는 AWS SDK 또는 CLI와 DescribeKey 함께 사용 섹션을 참조하십시오.

```
$ aws kms describe-key --key-id 1234abcd-12ab-34cd-56ef-1234567890ab

{
    "KeyMetadata": {
        "Origin": "AWS_KMS",
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "Description": "",
        "KeyManager": "CUSTOMER",
        "Enabled": true,
        "KeySpec": "SYMMETRIC_DEFAULT",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "KeyState": "Enabled",
        "CreationDate": 1499988169.234,
        "MultiRegion": false,
        "Arn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "AWSAccountId": "111122223333",
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ],
        "CurrentKeyMaterialId":
 "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0"
    }
}
```

이 예제에서는 서명 및 확인에 사용되는 비대칭 KMS 키에 대한 DescribeKey 작업을 호출합니다. 응답에는 AWS KMS 가 이 KMS 키에 대해 지원하는 서명 알고리즘이 포함됩니다.

```
$ aws kms describe-key --key-id 0987dcba-09fe-87dc-65ba-ab0987654321
```

```
{
    "KeyMetadata": {
        "KeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
        "Origin": "AWS_KMS",
        "Arn": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321",
        "KeyState": "Enabled",
        "KeyUsage": "SIGN_VERIFY",
        "CreationDate": 1569973196.214,
        "Description": "",
        "KeySpec": "ECC_NIST_P521",
        "CustomerMasterKeySpec": "ECC_NIST_P521",
        "AWSAccountId": "111122223333",
        "Enabled": true,
        "MultiRegion": false,
        "KeyManager": "CUSTOMER",
        "SigningAlgorithms": [
            "ECDSA_SHA_512"
        ]
    }
}
```

다양한 키 유형 식별

다음 주제에서는 AWS KMS 콘솔 및 DescribeKey 응답에서 다양한 키 유형을 식별하는 방법을 설명합니다.

KMS 키에 대한 세부 정보 페이지의 암호화 구성 탭으로 이동하는 데 도움이 필요하면 the section called “KMS 키 세부 정보 액세스 및 나열” 단원을 참조하세요.

주제

- 비대칭 KMS 키 식별
- HMAC KMS 키 식별
- 다중 리전 KMS 키 식별
- 가져온 키 구성 요소로 KMS 키 식별
- 키 스토어에서 KMS AWS CloudHSM 키 식별
- 외부 키 저장소에서 KMS 키 식별

비대칭 KMS 키 식별

AWS KMS 콘솔에서

고객 관리형 키 테이블의 키 유형 열에는 각 KMS 키가 대칭인지 또는 비대칭인지 여부가 표시됩니다. 키 유형 값을 기준으로 테이블을 필터링하여 비대칭 KMS 키만 표시할 수 있습니다. 자세한 내용은 the section called “KMS 키 정렬 및 필터링”을 참조하세요.

KMS 키의 세부 정보 페이지에 있는 암호화 구성 탭에는 키가 대칭인지 또는 비대칭인지 여부를 나타내는 키 유형이 표시됩니다. 또한 비대칭 KMS 키가 암호화 및 복호화, 서명 및 확인 또는 공유 비밀 도출에 사용되는지 여부를 나타내는 키 사용도 표시됩니다.

DescribeKey 응답에서

비대칭 KMS 키에서 `DescribeKey` 작업을 직접 호출하면 응답에 `KeySpec` 및 `KeyUsage` 값이 포함되며, 이러한 값은 KMS 키가 대칭인지 또는 비대칭인지 여부를 결정하는 데 사용할 수 있습니다.

`KeySpec` 값이 `SYMMETRIC_DEFAULT`인 경우 키는 대칭 암호화 KMS 키입니다. 비대칭 키 사양에 대한 자세한 내용은 키 사양 참조 단원을 참조하세요.

`KeyUsage` 값이 `SIGN_VERIFY` 또는 `KEY_AGREEMENT`인 경우 키는 비대칭 KMS 키입니다.

`DescribeKey` 작업은 비대칭 KMS 키에 대한 다음 세부 정보도 반환합니다.

- `KeyUsage` 값이 `ENCRYPT_DECRYPT`인 비대칭 KMS 키의 경우 이 작업은 키에 대한 유효한 암호화 알고리즘을 나열하는 `EncryptionAlgorithms`를 반환합니다.
- `KeyUsage` 값이 `SIGN_VERIFY`인 비대칭 KMS 키의 경우 작업은 키에 대한 유효한 서명 알고리즘을 나열하는 `SigningAlgorithms`를 반환합니다.
- `KeyUsage` 값이 `KEY_AGREEMENT`인 비대칭 KMS 키의 경우 작업은 키에 대한 유효한 키 계약 알고리즘을 나열하는 `KeyAgreementAlgorithms`를 반환합니다.

비대칭 KMS 키에 대한 자세한 내용은 the section called “비대칭 키” 단원을 참조하세요.

HMAC KMS 키 식별

AWS KMS 콘솔에서

HMAC KMS 키는 고객 관리형 키 테이블에 포함되지만 HMAC 키를 식별하는 키 사양 또는 키 사용 값을 기준으로 이 테이블을 정렬하거나 필터링할 수 없습니다. HMAC 키를 더 쉽게 찾으려면 HMAC 키에 고유 별칭이나 태그를 할당합니다. 그런 다음 별칭이나 태그를 기준으로 정렬하거나 필터링할 수 있습니다.

KMS 키의 세부 정보 페이지에 있는 암호화 구성 탭에는 키가 대칭인지 또는 비대칭인지 여부를 나타내는 키 유형이 표시됩니다. HMAC KMS 키는 대칭입니다. 암호화 구성 탭에는 키 사용도 표시됩니다. HMAC KMS 키의 경우 주요 사용 값은 항상 MAC 생성 및 확인입니다.

DescribeKey 응답에서

HMAC KMS 키에서 `DescribeKey` 작업을 직접 호출하면 응답에 `KeySpec` 및 `KeyUsage` 값이 포함됩니다. HMAC KMS 키의 경우 키 사용 값은 항상 `GENERATE_VERIFY_MAC`이고 키 사양 값은 항상 `HMAC_`로 시작합니다.

KMS 키에 대한 자세한 내용은 the section called “HMAC 키” 단원을 참조하세요.

다중 리전 KMS 키 식별

AWS KMS 콘솔에서

고객 관리형 키 테이블에는 선택한 리전의 KMS 키만 표시합니다. 선택한 리전에서 다중 리전 기본 키 및 복제본 키를 볼 수 있습니다. AWS 리전을 변경하려면 콘솔의 오른쪽 상단에 있는 리전 선택기를 사용합니다.

고객 관리형 키 테이블에서 다중 리전 키를 더 쉽게 식별할 수 있도록 테이블에 리전 특성 열을 추가합니다. 도움말은 the section called “KMS 키 테이블 사용자 지정”를 참조하십시오.

다중 리전 KMS 키의 세부 정보 페이지에는 리전 특성 탭이 포함되어 있습니다. 리전 구분(Regionality) 탭에는 기본 리전 변경 및 새 복제본 키 만들기 버튼이 포함됩니다. 복제본 키의 리전 구분(Regionality) 탭에는 버튼이 없습니다. 관련 다중 리전 키 섹션에는 현재 키와 관련된 모든 다중 리전 키가 나열됩니다. 현재 키가 복제본 키인 경우 이 목록에는 기본 키가 포함됩니다.

관련 다중 리전 키 테이블에서 관련 다중 리전 키를 선택하면 AWS KMS 콘솔이 선택한 키의 리전으로 변경되고 키의 세부 정보 페이지가 열립니다. 예를 들어 아래의 관련 다중 `sa-east-1` 리전 키 섹션에서 리전의 복제본 키를 선택하면 AWS KMS 콘솔이 `sa-east-1` 리전으로 변경되어 해당 복제본 키의 세부 정보 페이지가 표시됩니다. 이렇게 하면 복제본 키의 별칭 또는 키 정책을 볼 수 있습니다. 리전을 다시 변경하려면 페이지의 오른쪽 위 모서리에 있는 리전 선택기를 사용합니다.

DescribeKey 응답에서

기본적으로 AWS KMS API 작업은 리전 작업이며 현재 또는 지정된 리전의 리소스만 반환합니다. 그러나 다중 리전 KMS 키에서 `DescribeKey` 작업을 호출하면 응답에는 `MultiRegionConfiguration` 요소의 다른 AWS 리전에 있는 모든 관련 다중 리전 키가 포함됩니다.

다중 리전 KMS 키에 대한 자세한 내용은 the section called “다중 리전 키” 단원을 참조하세요.

가져온 키 구성 요소로 KMS 키 식별

AWS KMS 콘솔에서

고객 관리형 키 테이블에서 가져온 키 구성 요소를 사용하여 KMS 키를 더 쉽게 식별할 수 있도록 테이블에 오리진 열을 추가합니다. 오리진 열에서 외부(키 구성 요소 가져오기) 오리진 속성 값으로 KMS 키를 쉽게 식별할 수 있습니다. 도움말은 the section called “KMS 키 테이블 사용자 지정”를 참조하십시오.

KMS 키의 세부 정보 페이지에 있는 암호화 구성 탭에는 KMS 키에 대한 키 구성 요소의 소스를 식별하는 오리진이 표시됩니다. 가져온 키 구성 요소가 있는 KMS 키의 경우 오리진 값은 항상 외부(키 구성 요소 가져오기)입니다. 세부 정보 페이지에는 가져온 키 구성 요소에 대한 자세한 정보를 제공하는 키 구성 요소 탭도 포함되어 있습니다. EXTERNAL 오리진이 있는 대칭 암호화 키는 온디맨드 교체를 지원하며 여러 키 구성 요소를 연결할 수 있습니다. 이러한 키의 경우 탭에 키 구성 요소 및 교체 레이블이 지정됩니다.

DescribeKey 응답에서

가져온 키 구성 요소가 있는 KMS 키에서 `DescribeKey` 작업을 직접 호출하면 응답에 `Origin`, `ExpirationModel` 및 `ValidTo` 값이 포함됩니다. 가져온 키 구성 요소가 있는 KMS 키의 경우 오리진 값은 항상 `EXTERNAL`입니다. `ExpirationModel` 값은 키 구성 요소가 만료되도록 설정되었는지 여부를 나타내고 `ValidTo` 값은 키 구성 요소가 만료되는 시기를 나타냅니다. 다중 키 구성 요소가 키에 연결된 경우, `ValidTo` 값은 모든 키 구성 요소(보류 중인 교체 키 구성 요소를 제외) 중 가장 이른 만료 시간을 나타내며, 이러한 키 구성 요소 중 어느 것도 만료가 설정되지 않은 경우에만 `ExpirationModel` 값이 `DOES_NOT_EXPIRE`로 설정됩니다. 자세한 내용은 만료 시간 설정(선택 사항) 단원을 참조하십시오.

가져온 키 구성 요소가 있는 KMS 키에 대한 자세한 내용은 the section called “가져온 키 구성 요소” 단원을 참조하세요.

키 스토어에서 KMS AWS CloudHSM 키 식별

AWS KMS 콘솔에서

고객 관리형 키 테이블의 AWS CloudHSM 키 스토어에서 KMS 키를 더 쉽게 식별할 수 있도록 테이블에 오리진 열을 추가합니다. 오리진(Origin) 열에서 오리진 속성 값이 AWS CloudHSM인 KMS 키를 쉽게 식별할 수 있습니다. 도움말은 the section called “KMS 키 테이블 사용자 지정”를 참조하십시오.

KMS 키의 세부 정보 페이지에 있는 암호화 구성 탭에는 KMS 키에 대한 키 구성 요소의 소스를 식별하는 오리진이 표시됩니다. AWS CloudHSM 키 스토어의 KMS 키의 경우 오리진 값은 항상 입니다AWS CloudHSM.

AWS CloudHSM 키 스토어의 KMS 키의 경우 암호화 구성 탭에는 KMS 키와 연결된 키 스토어 및 클러스터에 대한 정보를 제공하는 추가 섹션인 사용자 지정 키 스토어가 포함되어 있습니다. AWS CloudHSM AWS CloudHSM

DescribeKey 응답에서

AWS CloudHSM 키 저장소의 KMS 키에서 `DescribeKey` 작업을 직접 호출하면 응답에 키 구성 요소의 소스를 식별하는 `Origin`이 포함됩니다. 키 스토어의 KMS AWS CloudHSM 키의 경우 오리진 값은 항상 입니다`AWS_CLOUDHSM`. 또한이 작업은 AWS CloudHSM 키 스토어의 KMS 키에 대해 다음과 같은 특수 필드를 반환합니다.

- `CloudHsmClusterId`
- `CustomKeyStoreId`

AWS CloudHSM 키 스토어에 대한 자세한 내용은 섹션을 참조하세요the section called “AWS CloudHSM 키 스토어”.

외부 키 저장소에서 KMS 키 식별

AWS KMS 콘솔에서

고객 관리형 키 테이블의 외부 키 저장소에서 KMS 키를 더 쉽게 식별할 수 있도록 테이블에 오리진 열을 추가합니다. 오리진 열에서 오리진 속성 값이 외부 키 저장소인 KMS 키를 쉽게 식별할 수 있습니다. 도움말은 the section called “KMS 키 테이블 사용자 지정”를 참조하십시오.

KMS 키의 세부 정보 페이지에 있는 암호화 구성 탭에는 KMS 키에 대한 키 구성 요소의 소스를 식별하는 오리진이 표시됩니다. 예를 들어 외부 키 저장소에 있는 KMS 키의 오리진 값은 외부 키 저장소입니다.

외부 키 저장소에 있는 KMS 키의 경우 암호화 구성 탭에는 사용자 지정 키 저장소 및 외부 키라는 두 개의 추가 섹션이 있습니다. 사용자 지정 키 저장소 테이블은 KMS 키와 연결된 외부 키 저장소에 대한 정보를 제공합니다. 외부 키 테이블은 외부 키 저장소의 KMS 키에 대해서만 AWS KMS 콘솔에 나타납니다. KMS 키와 연결된 외부 키에 대한 정보를 제공합니다. 외부 키는 외부 키 스토어의 KMS 키에 대한 키 구성 요소 역할을 AWS 하는 외부의 암호화 키입니다. KMS 키로 암호화하거나 복호화하면 외부 키 관리자가 지정된 외부 키를 사용하여 작업을 수행합니다.

다음 값이 External key(외부 키) 섹션에 나타납니다.

외부 키 ID

외부 키 관리자의 외부 키 식별자입니다. 외부 키 스토어 프록시가 외부 키를 식별하는 데 사용하는 값입니다. KMS 키를 생성할 때 외부 키의 ID를 지정하며 변경할 수 없습니다. KMS 키를 생성하는 데 사용한 외부 키 ID 값이 변경되거나 무효화되면 KMS 키 삭제를 예약하고 올바른 외부 키 ID 값으로 새 KMS 키를 생성해야 합니다.

DescribeKey 응답에서

외부 키 저장소의 KMS 키에서 `DescribeKey` 작업을 직접 호출하면 응답에 키 구성 요소의 소스를 식별하는 `Origin`이 포함됩니다. 키 스토어의 KMS AWS CloudHSM 키의 경우 오리진 값은 항상 입니다`EXTERNAL_KEY_STORE`. 이 작업은 KMS 키와 연결된 외부 키 저장소를 식별하는 `CustomKeyStoreId` 요소도 반환합니다.

외부 키 저장소에 대한 자세한 내용은 the section called “외부 키 스토어” 단원을 참조하세요.

콘솔 보기 사용자 지정

AWS KMS 콘솔 보기를 사용자 지정하여 KMS 키를 더 쉽게 찾을 수 있습니다. AWS 관리형 키 및 고객 관리형 키 페이지에 표시되는 테이블을 사용자 지정하여 가장 필요한 정보를 표시하거나 테이블에 반환된 KMS 키를 정렬하고 필터링할 수 있습니다.

주제

- KMS 키 정렬 및 필터링
- KMS 키 테이블 사용자 지정

KMS 키 정렬 및 필터링

콘솔에서 KMS 키를 더 쉽게 찾을 수 있도록 키 테이블을 정렬하고 필터링할 수 있습니다.

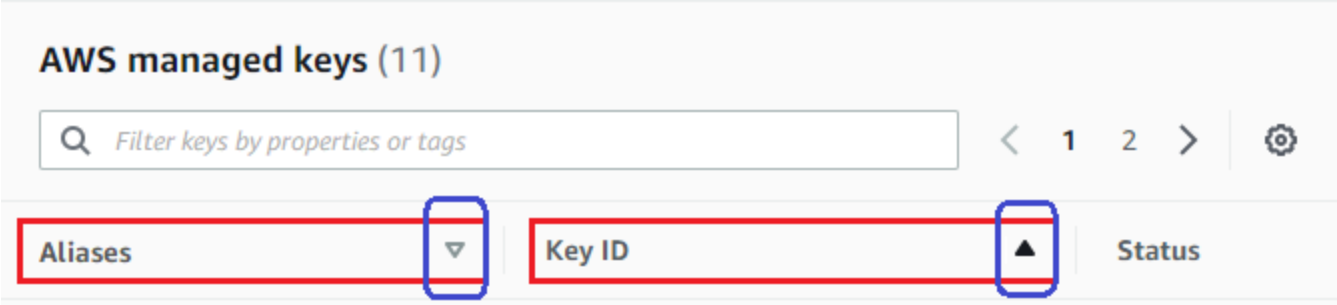
정렬

KMS 키를 열 값에 따라 오름차순 또는 내림차순으로 정렬할 수 있습니다. 이 기능은 현재 테이블 페이지에 표시되지 않더라도 테이블의 모든 KMS 키를 정렬합니다.

정렬 가능한 열은 열 이름 옆에 화살표가 표시됩니다. AWS 관리형 키 페이지에서는 별칭 또는 키 ID를 기준으로 정렬할 수 있습니다. 고객 관리형 키 페이지에서는 별칭, 키 ID 또는 키 유형을 기준으로 정렬할 수 있습니다.

오름차순으로 정렬하려면 화살표가 위쪽을 가리키도록 열 머리글을 선택합니다. 내림차순으로 정렬하려면 화살표가 아래쪽을 가리키도록 열 머리글을 선택합니다. 열은 한 번에 하나만 정렬할 수 있습니다.

예를 들어 기본값 별칭 대신 키 ID를 기준으로 KMS 키를 오름차순으로 정렬할 수 있습니다.



고객 관리형 키 페이지에서 KMS 키를 키 유형을 기준으로 오름차순으로 정렬하면 모든 비대칭 키가 모든 대칭 키 앞에 표시됩니다.

필터링

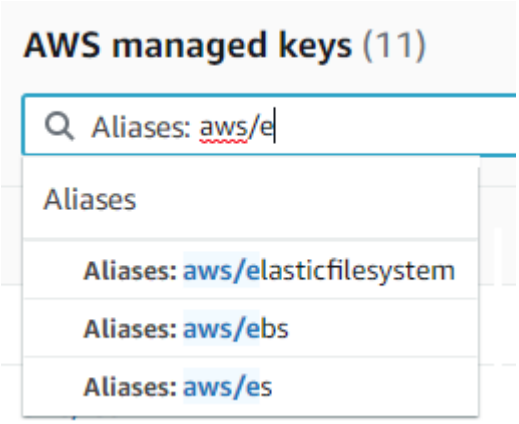
속성 값 또는 태그별로 KMS 키를 필터링할 수 있습니다. 필터는 현재 테이블 페이지에 표시되지 않더라도 테이블의 모든 KMS 키에 적용됩니다. 필터는 대소문자를 구분하지 않습니다.

필터링 가능한 속성이 필터 상자에 나열됩니다. AWS 관리형 키 페이지에서는 별칭 및 키 ID를 기준으로 필터링할 수 있습니다. 고객 관리형 키 페이지에서는 별칭, 키 ID 및 키 유형 속성을 기준으로, 그리고 태그를 기준으로 필터링할 수 있습니다.

- AWS 관리형 키 페이지에서는 별칭 및 키 ID를 기준으로 필터링할 수 있습니다.
- 고객 관리형 키 페이지에서는 태그를 기준으로 또는 별칭, 키 ID, 키 유형 또는 리전 구분 속성을 기준으로 필터링할 수 있습니다.

속성 값을 기준으로 필터링하려면 필터를 선택하고 속성 이름을 선택한 다음 실제 속성 값 목록에서 선택합니다. 태그별로 필터링하려면 태그 키를 선택한 다음 실제 태그 값 목록에서 선택합니다. 속성 또는 태그 키를 선택한 후 속성 값이나 태그 값의 전부 또는 일부를 입력할 수도 있습니다. 선택하기 전에 결과를 미리 볼 수 있습니다.

예를 들어 별칭 이름에 `aws/e`가 포함된 KMS 키를 표시하려면 필터 상자를 선택하고 별칭을 선택하고, `aws/e`를 입력한 다음 `Enter` 또는 `Return` 키를 눌러 필터를 추가합니다.



제안된 KMS 키 테이블 필터

비대칭 KMS 키에 대한 필터

고객 관리형 키 페이지에 비대칭 KMS 키만 표시하려면 필터 상자를 클릭하고, 키 유형(Key type)을 선택한 후, 키 유형: 비대칭(Key type: Asymmetric)을 선택합니다. 비대칭 옵션은 테이블에 비대칭 KMS 키가 있는 경우에만 나타납니다.

다중 리전 키에 대한 필터

다중 영역 키만 표시하려면 고객 관리형 키 페이지에서 필터 상자를 선택하고 리전 구분(Regionality)을 선택한 다음 리전 구분: 다중 리전(Regionality: Multi-Region)을 선택합니다. 다중 리전 옵션은 테이블에 다중 리전 키가 있는 경우에만 나타납니다.

태그에 대한 필터

특정 태그가 있는 KMS 키만 표시하려면 필터 상자를 선택하고 태그 키를 선택한 다음 실제 태그 값 중에서 선택합니다. 태그 값의 전부 또는 일부를 입력할 수도 있습니다.

결과 테이블에는 선택한 태그가 있는 모든 KMS 키가 표시됩니다. 그러나 태그를 표시하지는 않습니다. 태그를 보려면 KMS 키의 키 ID 또는 별칭을 선택하고 세부 정보 페이지에서 태그(Tags) 탭을 클릭합니다. 일반 구성 섹션 아래에 탭이 표시됩니다.

이 필터에는 태그 키와 태그 값이 모두 필요합니다. 태그 키만 입력하거나 값만 입력하여 KMS 키를 찾을 수 없습니다. 태그 키 또는 값의 전체 또는 일부를 기준으로 태그를 필터링하려면 ListResourceTags 작업을 사용하여 태그가 지정된 KMS 키를 가져온 다음 프로그래밍 언어의 필터링 기능을 사용합니다.

텍스트로 필터링

텍스트를 검색하려면 필터 상자에 별칭, 키 ID, 키 유형 또는 태그 키의 전부 또는 일부를 입력합니다. 태그 키를 선택한 후 태그 값을 검색할 수 있습니다. 선택하기 전에 결과를 미리 볼 수 있습니다.

예를 들어, 태그 키 또는 필터링 가능한 속성에 `test`가 있는 KMS 키를 표시하려면 필터 상자에 `test`를 입력합니다. 미리 보기에 필터가 선택할 KMS 키가 표시됩니다. 이 경우 `test`는 별칭 속성에만 표시됩니다.

KMS 키 테이블 사용자 지정

의 AWS 관리형 키 및 고객 관리형 키 페이지에 표시되는 테이블 AWS Management Console 을 필요에 맞게 사용자 지정할 수 있습니다. 테이블 열, AWS KMS keys 각 페이지의 수(페이지 크기), 텍스트 줄 바꿈을 선택할 수 있습니다. 선택한 구성은 확인하면 저장되고 페이지를 열 때마다 다시 적용됩니다.

KMS 키 테이블을 사용자 지정하려면

1. AWS 관리형 키 또는 고객 관리형 키 페이지에서 오른쪽 상단의 설정 아이콘 ( ) 을 선택합니다.
2. 기본 설정 페이지에서 선호하는 설정을 선택한 다음 확인을 선택합니다.

특히 스크롤하기 쉬운 디바이스를 주로 사용하는 경우 페이지 크기 설정을 사용하여 각 페이지에 표시되는 KMS 키 수를 늘리는 것이 좋습니다.

표시되는 데이터 열은 테이블, 사용자의 직무 역할 및 계정 및 리전의 KMS 키 유형에 따라 다를 수 있습니다. 다음 표에서는 몇 가지 권장 구성을 제공합니다. 열에 대한 설명은 AWS KMS 콘솔 사용 섹션을 참조하십시오.

제안된 KMS 키 테이블 구성

KMS 키 테이블에 표시되는 열을 사용자 지정하여 KMS 키에 대해 필요한 정보를 표시할 수 있습니다.

AWS 관리형 키

기본적으로 AWS 관리형 키 테이블에는 별칭, 키 ID 및 상태 열이 표시됩니다. 이러한 열은 대부분의 사용 사례에 이상적입니다.

대칭 암호화 KMS 키

AWS KMS가 생성하는 키 구성 요소가 있는 대칭 암호화 KMS 키만 사용하는 경우 별칭, 키 ID, 상태 및 생성 날짜 열이 가장 유용할 것입니다.

비대칭 KMS 키

비대칭 KMS 키를 사용하는 경우 별칭, 키 ID 및 상태 열 외에 키 유형, 키 사양 및 키 사용 열을 추가하는 것이 좋습니다. 이러한 열에는 KMS 키가 대칭 또는 비대칭인지 여부, 키 구성 요소의 유형, KMS 키를 암호화 또는 서명에 사용할 수 있는지 여부가 표시됩니다.

HMAC KMS 키

HMAC KMS 키를 사용하는 경우 별칭, 키 ID 및 상태 열 외에 키 사양 및 키 사용 열을 추가하는 것이 좋습니다. 이 열에는 KMS 키가 HMAC 키인지 여부가 표시됩니다. 키 사양 또는 키 사용량별로 KMS 키를 정렬할 수 없으므로 별칭 및 태그를 사용하여 HMAC 키를 식별한 다음 AWS KMS 콘솔의 필터 기능을 사용하여 별칭 또는 태그를 기준으로 필터링합니다.

가져온 키 구성 요소

가져온 키 구성 요소(imported key material)가 있는 KMS 키가 있는 경우 오리진(Origin) 및 만료 날짜(Expiration date) 열을 추가하는 것이 좋습니다. 이러한 열에는 KMS 키의 키 구성 요소를에서 가져오거나 생성할지 여부와 키 구성 요소가 만료 AWS KMS 되는 시기가 표시됩니다. 생성 날짜(Creation date) 필드에는 KMS 키가 생성된 날짜(키 구성 요소 없음)가 표시됩니다. 키 구성 요소의 특성은 전혀 반영하지 않습니다.

사용자 지정 키 스토어

사용자 지정 키 스토어에 KMS 키가 있는 경우 Origin(오리진) 열과 Custom key store ID(사용자 지정 키 스토어 ID) 열을 추가하는 것이 좋습니다. 이러한 열은 KMS 키가 사용자 지정 키 스토어에 있음을 보여주고, 사용자 지정 키 스토어 유형을 표시하고, 사용자 지정 키 스토어를 식별합니다.

다중 리전 키

다중 리전 키가 있는 경우 리전 분류(Regionality) 열을 추가하는 것이 좋습니다 KMS 키가 단일 리전 키, 다중 리전 기본 키 또는 다중 리전 복제본 키인지 여부를 나타냅니다.

키 스토어에서 KMS 키 및 AWS CloudHSM 키 구성 요소 찾기

AWS CloudHSM 키 스토어를 관리하는 경우 각 AWS CloudHSM 키 스토어에서 KMS 키를 식별해야 할 수 있습니다. 예를 들어, 다음 작업 중 일부를 수행해야 할 수 있습니다.

- AWS CloudTrail 로그의 키 스토어에서 KMS AWS CloudHSM 키를 추적합니다.

- 키 스토어 연결 해제의 KMS AWS CloudHSM 키에 미치는 영향을 예측합니다.
- 키 스토어를 삭제하기 전에 KMS AWS CloudHSM 키 삭제를 예약합니다.

또한 KMS 키의 키 구성 요소 역할을 하는 AWS CloudHSM 클러스터의 키를 식별할 수 있습니다. 는 KMS 키와 키 구성 요소를 AWS KMS 관리하지만 AWS CloudHSM 클러스터 관리와 HSMs 및 백업과 HSM의 키에 대한 제어 및 책임은 여전히 유지됩니다 HSMs. 키 구성 요소를 감사하거나, 실수로 삭제되지 않도록 보호하거나, KMS 키 삭제 이후 HSM 및 클러스터 백업에서 이를 삭제하기 위해 키 식별이 필요할 수 있습니다.

키 스토어의 KMS 키에 대한 모든 AWS CloudHSM 키 구성 요소는 `kmsuser` 암호화 사용자(CU)가 소유합니다.는 에서만 볼 수 있는 키 레이블 속성을 KMS 키의 Amazon 리소스 이름(ARN) AWS CloudHSM으로 AWS KMS 설정합니다.

KMS 키 및 키 구성 요소를 찾으려면 다음 기법 중 하나를 사용합니다.

- 키 스토어에서 KMS AWS CloudHSM 키 찾기 - 하나 또는 모든 키 스토어에서 KMS AWS CloudHSM 키를 식별하는 방법입니다.
- 키 스토어의 모든 AWS CloudHSM 키 찾기 - AWS CloudHSM 키 스토어의 KMS 키에서 키 구성 요소 역할을 하는 클러스터의 모든 키를 찾는 방법.
- KMS AWS CloudHSM 키의 키 찾기 - 클러스터에서 키 AWS CloudHSM 스토어의 특정 KMS 키에 대한 키 구성 요소 역할을 하는 키를 찾는 방법입니다.
- 키의 KMS AWS CloudHSM 키 찾기 — 클러스터에서 특정 키의 KMS 키를 찾는 방법.

키 스토어에서 KMS AWS CloudHSM 키 찾기

AWS CloudHSM 키 스토어를 관리하는 경우 각 AWS CloudHSM 키 스토어에서 KMS 키를 식별해야 할 수 있습니다. 이 정보를 사용하여 AWS CloudTrail 로그의 KMS 키 작업을 추적하거나, 사용자 지정 키 스토어가 KMS 키에 미치는 영향을 예측하거나, AWS CloudHSM 키 스토어를 삭제하기 전에 KMS 키 삭제를 예약할 수 있습니다.

키 스토어에서 KMS AWS CloudHSM 키를 찾으려면(콘솔)

특정 키 스토어에서 KMS AWS CloudHSM 키를 찾으려면 고객 관리형 키 페이지에서 사용자 지정 키 스토어 이름 또는 사용자 지정 키 스토어 ID 필드의 값을 확인합니다. AWS CloudHSM 키 스토어에서 KMS 키를 식별하려면 오리진 값이 인 KMS 키를 찾습니다AWS CloudHSM. 화면에 열 옵션을 추가하려면 페이지의 오른쪽 상단 모서리에서 기어 아이콘을 선택합니다.

키 스토어에서 KMS AWS CloudHSM 키를 찾으려면(API)

AWS CloudHSM 키 스토어에서 KMS 키를 찾으려면 ListKeys 및 DescribeKey 작업을 사용한 다음 `CustomKeyStoreId` 값을 기준으로 필터링합니다. 다음 예제를 실행하기 전에 가상의 사용자 지정 키 저장소 ID를 유효한 값으로 대체합니다.

Bash

특정 키 스토어에서 KMS AWS CloudHSM 키를 찾으려면 계정 및 리전의 모든 KMS 키를 가져옵니다. 그런 다음, 사용자 지정 키 스토어의 ID를 기준으로 필터링합니다.

```
for key in $(aws kms list-keys --query 'Keys[*].KeyId' --output text) ;
do aws kms describe-key --key-id $key |
grep '"CustomKeyStoreId": "cks-1234567890abcdef0"' --context 100; done
```

계정 및 리전의 모든 AWS CloudHSM 키 스토어에서 KMS 키를 가져오려면 값이 `CustomKeyStoreType`인를 검색합니다`AWS_CloudHSM`.

```
for key in $(aws kms list-keys --query 'Keys[*].KeyId' --output text) ;
do aws kms describe-key --key-id $key |
grep '"CustomKeyStoreType": "AWS_CloudHSM"' --context 100; done
```

PowerShell

특정 키 스토어에서 KMS AWS CloudHSM 키를 찾으려면 Get-KmsKeyList 및 Get-KmsKey cmdlet을 사용하여 계정 및 리전의 모든 KMS 키를 가져옵니다. 그런 다음, 사용자 지정 키 스토어의 ID를 기준으로 필터링합니다.

```
PS C:\> Get-KMSKeyList | Get-KMSKey | where CustomKeyStoreId -eq
 'cks-1234567890abcdef0'
```

계정 및 리전의 AWS CloudHSM 키 스토어에서 KMS 키를 가져오려면 CustomKeyStoreType 값을 필터링합니다`AWS_CLOUDHSM`.

```
PS C:\> Get-KMSKeyList | Get-KMSKey | where CustomKeyStoreType -eq 'AWS_CLOUDHSM'
```

키 스토어의 모든 AWS CloudHSM 키 찾기

키 스토어의 키 구성 요소 역할을 하는 AWS CloudHSM 클러스터의 키를 식별할 수 있습니다 AWS CloudHSM . 이렇게 하려면 CloudHSM CLI에서 key list 명령을 사용합니다.

키 목록 명령을 사용하여 키의 AWS KMS 를 AWS CloudHSM 찾을 수도 있습니다. 가 AWS CloudHSM 클러스터의 KMS 키에 대한 키 구성 요소를 AWS KMS 생성하면 키 레이블에 KMS 키의 Amazon 리소스 이름(ARN)을 기록합니다. key list 명령은 `key-reference` 및 `label`을 반환합니다.

> **참고**
>
> 다음 절차에서는 AWS CloudHSM 클라이언트 SDK 5 명령줄 도구인 CloudHSM CLI를 사용합니다. CloudHSM CLI에서 `key-handle`을 `key-reference`로 대체합니다.
> 2025년 1월 1일에 AWS CloudHSM 는 클라이언트 SDK 3 명령줄 도구, CloudHSM 관리 유틸리티(CMU) 및 키 관리 유틸리티(KMU)에 대한 지원을 종료합니다. Client SDK 3 명령줄 도구와 Client SDK 5 명령줄 도구의 차이점에 대한 자세한 내용은 AWS CloudHSM 사용 설명서의 클라이언트 SDK 3 CMU 및 KMU에서 클라이언트 SDK 5 CloudHSM CLI로 마이그레이션을 참조하세요.

이 절차를 실행하려면 `kmsuser` CU로 로그인할 수 있도록 AWS CloudHSM 키 스토어의 연결을 일시적으로 끊어야 합니다.

1. AWS CloudHSM 키 스토어가 아직 연결 해제되지 않은 경우에 설명된 `kmsuser`대로 로 로그인합니다연결 해제 및 로그인 방법.

   > **Note**
   >
   > 사용자 지정 키 스토어의 연결이 해제된 상태에서는 사용자 지정 키 스토어에서 KMS 키를 생성하거나, 암호화 작업을 위해 기존 KMS 키를 사용하려는 모든 시도가 실패합니다. 이 작업은 사용자가 기밀 데이터를 저장하거나 액세스하지 못하도록 차단합니다.

2. CloudHSM CLI의 키 목록 명령을 사용하여 AWS CloudHSM 클러스터에 있는 현재 사용자의 모든 키를 찾습니다.

   기본적으로 현재 로그인한 사용자의 키 10개만 표시되고 `key-reference` 및 `label`만 출력으로 표시됩니다. 자세한 옵션은 AWS CloudHSM 사용 설명서의 키 목록을 참조하세요.

   ```
   aws-cloudhsm > key list
   ```

```
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x0000000000000123",
        "attributes": {
          "label": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
      },
      {
        "key-reference": "0x0000000000000456",
        "attributes": {
          "label": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321"
        }
      },.
      ...8 keys later...
    ],
    "total_key_count": 56,
    "returned_key_count": 10,
    "next_token": "10"
  }
}
```


3. 에 설명된 대로 AWS CloudHSM 키 스토어를 로그아웃하고 다시 연결합니다로그아웃 및 재연결 방법.

키의 KMS AWS CloudHSM 키 찾기

가 클러스터에서 kmsuser 소유한 키의 키 참조 또는 ID를 알고 있는 경우 해당 값을 사용하여 AWS CloudHSM 키 스토어에서 연결된 KMS 키를 식별할 수 있습니다.

가 AWS CloudHSM 클러스터의 KMS 키에 대한 키 구성 요소를 AWS KMS 생성하면 키 레이블에 KMS 키의 Amazon 리소스 이름(ARN)을 기록합니다. 레이블 값을 변경하지 않은 한 CloudHSM CLI의 key list 명령을 사용하여 AWS CloudHSM 키와 연결된 KMS 키를 식별할 수 있습니다.

> **참고**
>
> 다음 절차에서는 AWS CloudHSM 클라이언트 SDK 5 명령줄 도구인 CloudHSM CLI를 사용합니다. CloudHSM CLI에서 `key-handle`을 `key-reference`로 대체합니다.

2025년 1월 1일에 AWS CloudHSM 는 클라이언트 SDK 3 명령줄 도구, CloudHSM 관리 유틸리티(CMU) 및 키 관리 유틸리티(KMU)에 대한 지원을 종료합니다. Client SDK 3 명령줄 도구와 Client SDK 5 명령줄 도구의 차이점에 대한 자세한 내용은 AWS CloudHSM 사용 설명서의 클라이언트 SDK 3 CMU 및 KMU에서 클라이언트 SDK 5 CloudHSM CLI로 마이그레이션을 참조하세요.

이러한 절차를 실행하려면 kmsuser CU로 로그인할 수 있도록 AWS CloudHSM 키 스토어의 연결을 일시적으로 끊어야 합니다.

> **Note**
>
> 사용자 지정 키 스토어의 연결이 해제된 상태에서는 사용자 지정 키 스토어에서 KMS 키를 생성하거나, 암호화 작업을 위해 기존 KMS 키를 사용하려는 모든 시도가 실패합니다. 이 작업은 사용자가 기밀 데이터를 저장하거나 액세스하지 못하도록 차단합니다.

주제

- 키 참조와 연결된 KMS 키 식별
- 백업 키 ID와 연결된 KMS 키 식별

키 참조와 연결된 KMS 키 식별

다음 절차에서는 key-reference 속성 필터와 함께 CloudHSM CLI에서 key list 명령을 사용하여 키 저장소의 특정 KMS 키의 키 구성 요소 역할을 하는 AWS CloudHSM 키를 클러스터에서 찾는 방법을 보여줍니다.

1. AWS CloudHSM 키 스토어가 아직 연결 해제되지 않은 경우에 설명된 kmsuser대로 로 로그인합니다연결 해제 및 로그인 방법.
2. CloudHSM CLI의 key list 명령을 사용하여 key-reference 속성을 기준으로 필터링합니다. 일치하는 키에 대한 모든 속성과 키 정보를 포함하도록 verbose 인수를 지정합니다. verbose 인수를 지정하지 않으면 key list 작업은 일치하는 키의 키 참조 및 레이블 속성만 반환합니다.

   이 명령을 실행하기 전에 예제 키 key-reference를 계정의 유효한 값으로 바꿉니다.

```
aws-cloudhsm > key list --filter attr.key-reference="0x0000000000120034" --verbose
{
  "error_code": 0,
```

```
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x0000000000120034",
        "key-info": {
          "key-owners": [
            {
              "username": "kmsuser",
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "aes",
          "label": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
          "id": "0xbacking-key-id",
          "check-value": "0x29bbd1",
          "class": "my_test_key",
          "encrypt": true,
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": false,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": false,
          "trusted": false,
          "unwrap": true,
          "verify": false,
          "wrap": true,
          "wrap-with-trusted": false,
          "key-length-bytes": 32
        }
      }
    ],
    "total_key_count": 1,
```

```
    "returned_key_count": 1
  }
}
```

3. 에 설명된 대로 AWS CloudHSM 키 스토어를 로그아웃하고 다시 연결합니다로그아웃 및 재연결 방법.

백업 키 ID와 연결된 KMS 키 식별

AWS CloudHSM 키 스토어에서 KMS 키를 사용하는 암호화 작업에 대한 모든 CloudTrail 로그 항목에는 customKeyStoreId 및가 있는 additionalEventData 필드가 포함됩니다backingKeyId. backingKeyId 필드에 반환된 값은 CloudHSM 키 id 속성과 관련이 있습니다. id 속성을 기준으로 key list 작업을 필터링하여 특정 backingKeyId와 연결된 KMS 키를 식별할 수 있습니다.

1. AWS CloudHSM 키 스토어가 아직 연결 해제되지 않은 경우에 설명된 kmsuser대로 로 로그인합니다연결 해제 및 로그인 방법.
2. 속성 필터와 함께 CloudHSM CLI의 key list 명령을 사용하여 AWS CloudHSM 키 저장소의 특정 KMS 키에 대한 키 구성 요소 역할을 하는 키를 클러스터에서 찾습니다.

   다음 예제에서는 id 속성을 기준으로 필터링하는 방법을 보여줍니다. AWS CloudHSM 은 id 값을 16진수 값으로 인식합니다. id 속성을 기준으로 키 목록 작업을 필터링하려면 먼저 CloudTrail 로그 항목에서 식별한 backingKeyId 값을가 AWS CloudHSM 인식하는 형식으로 변환해야 합니다.

   a. 다음 Linux 명령을 사용하여 backingKeyId를 16진수 표현으로 변환합니다.

   ```
   echo backingKeyId | tr -d '\n' |  xxd -p
   ```

   다음 예제에서는 backingKeyId 바이트 배열을 16진수 표현으로 변환하는 방법을 보여줍니다.

   ```
   echo 5890723622dc15f699aa9ab2387a9f744b2b884c18b2186ee8ada4f556a2eb9d | tr -d
    '\n' |  xxd -p
   3538393037323336323264633135663639396161396162323338376139663734346232623838346331386230
   ```

   b. 0x를 사용하여 backingKeyId의 16진수 표현을 준비합니다.

   ```
   0x3538393037323336323264633135663639396161396162323338376139663734346232623838346331386230
   ```

c. 변환된 backingKeyId 값을 사용하여 id 속성을 기준으로 필터링합니다. 일치하는 키에 대한 모든 속성과 키 정보를 포함하도록 verbose 인수를 지정합니다. verbose 인수를 지정하지 않으면 key list 작업은 일치하는 키의 키 참조 및 레이블 속성만 반환합니다.

```
aws-cloudhsm > key list --filter
 attr.id="0x353839303732333632326463313566363939616139616232333837613966373434623262383
 --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x0000000000120034",
        "key-info": {
          "key-owners": [
            {
              "username": "kmsuser",
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "aes",
          "label": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
          "id":
 "0x353839303732333632326463313566363939616139616232333837613966373434623262383834633133
          "check-value": "0x29bbd1",
          "class": "my_test_key",
          "encrypt": true,
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": false,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
```

```
          "sign": false,
          "trusted": false,
          "unwrap": true,
          "verify": false,
          "wrap": true,
          "wrap-with-trusted": false,
          "key-length-bytes": 32
        }
      }
    ],
    "total_key_count": 1,
    "returned_key_count": 1
  }
}
```

3. 에 설명된 대로 AWS CloudHSM 키 스토어를 로그아웃하고 다시 연결합니다로그아웃 및 재연결 방법.

KMS AWS CloudHSM 키의 키 찾기

키 스토어에서 KMS 키의 KMS AWS CloudHSM 키 ID를 사용하여 키 구성 요소 역할을 하는 AWS CloudHSM 클러스터의 키를 식별할 수 있습니다.

가 AWS CloudHSM 클러스터의 KMS 키에 대한 키 구성 요소를 AWS KMS 생성하면 키 레이블에 KMS 키의 Amazon 리소스 이름(ARN)을 기록합니다. 레이블 값을 변경하지 않은 경우 CloudHSM CLI에서 findKey 명령을 사용해 KMS 키대한 키 구성 요소의 key-resource와 ID를 찾을 수 있습니다.

AWS CloudHSM 키 스토어에서 KMS 키를 사용하는 암호화 작업에 대한 모든 CloudTrail 로그 항목에는 `customKeyStoreId` 및가 있는 `additionalEventData` 필드가 포함됩니다`backingKeyId`. `backingKeyId` 필드에 반환되는 값은 `id` AWS CloudHSM 키 속성입니다. KMS 키 ARN을 기준으로 키 목록 AWS CloudHSM CLI 작업을 필터링하여 특정 KMS 키와 연결된 CloudHSM 키 `id` 속성을 식별할 수 있습니다.

이 절차를 실행하려면 `kmsuser` CU로 로그인할 수 있도록 AWS CloudHSM 키 스토어의 연결을 일시적으로 끊어야 합니다.

> **참고**
>
> 다음 절차에서는 AWS CloudHSM 클라이언트 SDK 5 명령줄 도구인 CloudHSM CLI를 사용합니다. CloudHSM CLI에서 `key-handle`을 `key-reference`로 대체합니다.

2025년 1월 1일에 AWS CloudHSM 는 클라이언트 SDK 3 명령줄 도구, CloudHSM 관리 유틸리티(CMU) 및 키 관리 유틸리티(KMU)에 대한 지원을 종료합니다. Client SDK 3 명령줄 도구와 Client SDK 5 명령줄 도구의 차이점에 대한 자세한 내용은 AWS CloudHSM 사용 설명서의 클라이언트 SDK 3 CMU 및 KMU에서 클라이언트 SDK 5 CloudHSM CLI로 마이그레이션을 참조하세요.

1. AWS CloudHSM 키 스토어가 아직 연결 해제되지 않은 경우에 설명된 kmsuser대로 로 로그인합니다연결 해제 및 로그인 방법.

   > **Note**
   >
   > 사용자 지정 키 스토어의 연결이 해제된 상태에서는 사용자 지정 키 스토어에서 KMS 키를 생성하거나, 암호화 작업을 위해 기존 KMS 키를 사용하려는 모든 시도가 실패합니다. 이 작업은 사용자가 기밀 데이터를 저장하거나 액세스하지 못하도록 차단합니다.

2. CloudHSM CLI에서 키 목록 명령을 사용하고 로 필터링label하여 AWS CloudHSM 클러스터의 특정 키에 대한 KMS 키를 찾습니다. 일치하는 키에 대한 모든 속성과 키 정보를 포함하도록 verbose 인수를 지정합니다. verbose 인수를 지정하지 않으면 key list 작업은 일치하는 키의 key-reference 및 label 속성만 반환합니다.

   다음 예제에서는 KMS 키 ARN을 저장하는 label 속성을 기준으로 필터링하는 방법을 보여줍니다. 이 명령을 실행하기 전에 예제 KMS 키 ARN을 계정의 유효한 값으로 대체합니다.

```
aws-cloudhsm > key list --filter attr.label="arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab" --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x0000000000120034",
        "key-info": {
          "key-owners": [
            {
              "username": "kmsuser",
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
```

```
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "aes",
          "label": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
          "id": "0xbacking-key-id",
          "check-value": "0x29bbd1",
          "class": "my_test_key",
          "encrypt": true,
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": false,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": false,
          "trusted": false,
          "unwrap": true,
          "verify": false,
          "wrap": true,
          "wrap-with-trusted": false,
          "key-length-bytes": 32
        }
      }
    ],
    "total_key_count": 1,
    "returned_key_count": 1
  }
}
```

3. 에 설명된 대로 AWS CloudHSM 키 스토어를 로그아웃하고 다시 연결합니다로그아웃 및 재연결 방법.

KMS 키 활성화 및 비활성화

고객 관리형 키를 비활성화했다가 다시 활성화할 수 있습니다. KMS 키를 생성하면 필터가 기본적으로 활성화됩니다. KMS 키를 비활성화하면 다시 활성화할 때까지 어떠한 암호화 작업에서도 사용할 수 없습니다.

KMS 키를 비활성화하는 것은 일시적이고 쉽게 취소할 수 있으므로 파괴적이고 되돌릴 수 없는 작업인 KMS 키 삭제에 비해 안전한 대안입니다. KMS 키 삭제를 고려하는 경우 먼저 키를 비활성화하고, 암호화된 데이터를 해독하기 위해 키를 사용할 필요가 없도록 CloudWatch 경보 또는 유사한 메커니즘을 설정합니다.

KMS 키를 비활성화하면 즉시 사용할 수 없게 됩니다(최종 일관성에 따라 다름). 그러나 KMS 키로 보호되는 데이터 키로 암호화된 리소스는 데이터 키를 복호화하는 등 KMS 키를 다시 사용할 때까지 영향을 받지 않습니다. 이 문제는 영향을 미치며 AWS 서비스, 대부분 데이터 키를 사용하여 리소스를 보호합니다. 자세한 내용은 사용할 수 없는 KMS 키가 데이터 키에 미치는 영향을 참조하세요.

AWS 관리형 키 또는를 활성화 또는 비활성화할 수 없습니다AWS 소유 키. AWS 관리형 키 는를 사용하는 서비스 AWS KMS에서만 사용할 수 있도록 영구적으로 활성화됩니다. AWS 소유 키 는 해당 서비스를 소유한 서비스에서만 관리합니다.

> **Note**
>
> AWS KMS 는 고객 관리형 키가 비활성화된 상태에서 키 구성 요소를 교체하지 않습니다. 자세한 내용은 키 교체의 작동 방식 단원을 참조하십시오.

AWS KMS 콘솔 사용

AWS KMS 콘솔을 사용하여 고객 관리형 키를 활성화 및 비활성화할 수 있습니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 탐색 창에서 고객 관리형 키를 선택합니다.
4. 활성화하거나 비활성화할 KMS 키의 확인란을 선택합니다.
5. KMS 키를 활성화하려면 키 작업(Key actions), 활성화(Enable)를 선택합니다. KMS 키를 비활성화하려면 키 작업(Key actions), 비활성화(Disable)를 선택합니다.

AWS KMS API 사용

EnableKey 작업은 비활성화된를 활성화합니다 AWS KMS key. 이 예제들은 AWS Command Line Interface (AWS CLI)를 사용하지만, 사용자는 어떤 지원되는 프로그래밍 언어라도 사용할 수 있습니다. `key-id` 파라미터가 필요합니다.

이 작업은 출력을 반환하지 않습니다. 키 상태를 보려면 DescribeKey 작업을 사용합니다.

```
$ aws kms enable-key --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

DisableKey 작업은 활성화된 KMS 키를 비활성화합니다. `key-id` 파라미터가 필요합니다.

```
$ aws kms disable-key --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

이 작업은 출력을 반환하지 않습니다. 키 상태를 보려면 DescribeKey 작업을 사용하고 `Enabled` 필드를 확인합니다.

```
$ aws kms describe-key --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
{
    "KeyMetadata": {
        "Origin": "AWS_KMS",
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "Description": "",
        "KeyManager": "CUSTOMER",
        "MultiRegion": false,
        "Enabled": false,
        "KeyState": "Disabled",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "CreationDate": 1502910355.475,
        "Arn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "AWSAccountId": "111122223333"
        "KeySpec": "SYMMETRIC_DEFAULT",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ]
    }
}
```

회전 AWS KMS keys

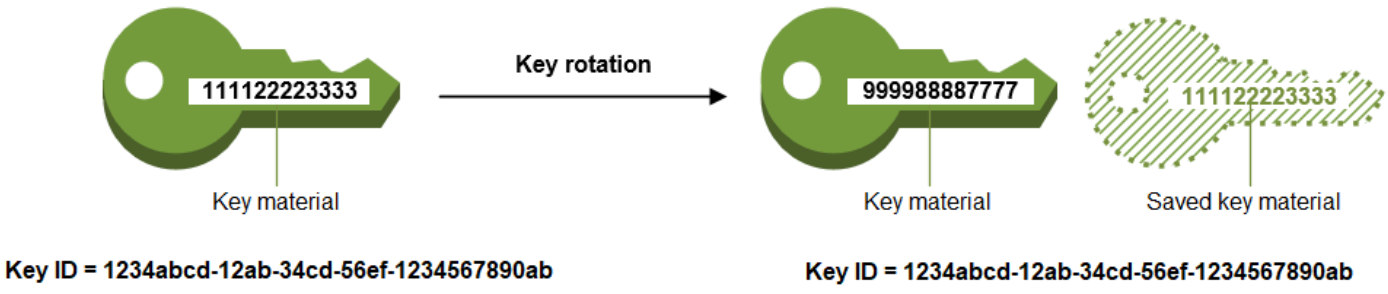
고객 관리형 키에 대한 새로운 암호화 자료를 만들려면 새 KMS 키를 만든 다음, 새 KMS 키를 사용하도록 애플리케이션 또는 별칭을 변경하면 됩니다. 또는 자동 키 교체를 활성화하거나 온디맨드 교체를 수행하여 기존 KMS 키와 연결된 키 구성 요소를 교체할 수 있습니다.

기본적으로 KMS 키에 대해 자동 키 교체를 활성화하면는 매년 KMS 키에 대한 새 암호화 구성 요소를 AWS KMS 생성합니다. 또한 사용자 지정을 지정rotation-period하여 키 구성 요소를 AWS KMS 교체할 자동 키 교체를 활성화한 후의 일수와 이후 각 자동 교체 사이의 일수를 정의할 수 있습니다. 키 구성 요소 교체를 즉시 시작해야 하는 경우 자동 키 교체 활성화 여부에 관계없이 온디맨드 교체를 수행할 수 있습니다. 온디맨드 교체는 기존 자동 교체 일정을 변경하지 않습니다.

Amazon CloudWatch AWS CloudTrail및 AWS Key Management Service 콘솔에서 KMS 키에 대한 키 구성 요소의 교체를 추적할 수 있습니다. GetKeyRotationStatus 작업을 사용하여 KMS 키에 대해 자동 교체가 활성화되어 있는지 확인하고 진행 중인 온디맨드 교체를 식별할 수도 있습니다. ListKeyRotations 작업을 사용하여 완료된 교체의 세부 정보를 볼 수 있습니다.

키를 교체하면 암호화 작업에 사용되는 암호화 비밀인 현재 키 구성 요소만 변경됩니다. 교체된 KMS 키를 사용하여 사이퍼텍스트를 복호화하면는 암호화에 사용된 키 구성 요소를 AWS KMS 사용합니다. 복호화 작업을 위해 키 구성 요소를 선택할 수 없으며 AWS KMS 에서 자동으로 올바른 키 구성 요소를 선택합니다. 는 적절한 키 구성 요소를 사용하여 AWS KMS 투명하게 복호화하므로 코드 변경 AWS 서비스 없이 애플리케이션에서 교체된 KMS 키를 안전하게 사용할 수 있습니다.

KMS 키는 키 구성 요소가 변경되는지 여부 또는 횟수에 관계없이 동일한 논리적 리소스입니다. 다음 이미지와 같이, KMS 키의 속성은 변경되지 않습니다.



새 KMS 키를 생성하여 원본 KMS 키 대신 사용하기로 결정할 수 있습니다. 이렇게 하면 기존 KMS 키에서 키 구성 요소를 교체하는 것과 동일한 효과가 있으므로, 이 방법은 일반적으로 키를 수동으로 교체하는 것으로 간주됩니다. 수동 교체는 비대칭 KMS 키, HMAC KMS 키 및 사용자 지정 키 스토어의 KMS 키를 포함하여 자동 또는 온디맨드 키 교체에 적합하지 않은 KMS 키를 교체하려는 경우에 적합합니다. ???

> **Note**
>
> 키 교체는 KMS 키가 보호하는 데이터에는 영향을 주지 않습니다. 이는 KMS 키가 생성한 데이터 키를 교체하지 않으며, KMS 키로 보호된 데이터를 다시 암호화하지 않습니다. 키 교체는 손상된 데이터 키의 영향을 완화할 수 없습니다.

키 교체 및 요금

AWS KMS 는 KMS 키에 대해 유지되는 키 구성 요소의 첫 번째 및 두 번째 교체에 대해 월별 요금을 부과합니다. 이 가격 인상은 두 번째 교체 시 제한되며 후속 교체에는 요금이 청구되지 않습니다. 자세한 내용은 AWS Key Management Service 요금을 참조하세요.

> **Note**
>
> AWS Cost Explorer Service을 사용하여 키 스토리지 요금 내역을 확인할 수 있습니다. 예를 들어 사용 유형에 대한 **$REGION-KMS-Keys**를 지정하고 API 작업별로 데이터를 그룹화하여 현재 KMS 키와 교체된 KMS 키로 청구되는 키의 총 요금을 확인할 수 있도록 보기를 필터링할 수 있습니다.
> 이전 날짜에 대한 레거시 Unknown API 작업 인스턴스가 계속 표시될 수 있습니다.

키 교체 및 할당량

각 KMS 키는 키 리소스 할당량을 계산할 때 교체된 키 구성 요소 버전의 수에 관계없이 하나의 키로 계산됩니다.

키 구성 요소 및 교체에 대한 자세한 내용은 AWS Key Management Service 암호화 세부 정보를 참조하세요.

주제

- KMS 키를 교체하는 이유는 무엇인가요?
- 키 교체의 작동 방식
- 자동 키 교체를 활성화하려면
- 자동 키 교체 비활성화
- 온디맨드 키 교체 수행
- 교체 및 키 구성 요소 나열
- 수동으로 키 교체

- 다중 리전 키 세트에서 프라이머리 키 변경

KMS 키를 교체하는 이유는 무엇인가요?

암호화 모범 사례에서는가 AWS KMS 생성하는 데이터 키와 같이 데이터를 직접 암호화하는 키를 광범위하게 재사용하지 않는 것이 좋습니다. 256비트 데이터 키가 수백만 개의 메시지를 암호화하면 데이터가 소진되어 미묘한 패턴을 가진 사이퍼텍스트를 생성하기 시작할 수 있습니다. 영리한 액터는 이를 악용하여 키의 비트를 찾아낼 수 있습니다. 키 소진 위험을 줄이기 위해 데이터 키는 한 번 또는 소수의 횟수만 사용하는 것이 가장 좋습니다.

하지만 KMS 키는 키 암호화 키라고도 알려진 래핑 키로 가장 자주 사용됩니다. 래핑 키는 데이터를 암호화하는 대신 데이터를 암호화하는 데이터 키를 암호화합니다. 따라서 데이터 키보다 사용 빈도가 훨씬 적고 키가 고갈될 위험이 있을 정도로 재사용되는 경우가 거의 없습니다.

이렇게 소진 위험은 매우 낮지만 비즈니스 또는 계약 규칙이나 정부 규정으로 인해 KMS 키를 교체해야 할 수도 있습니다. KMS 키를 교체해야 한다면 자동 교체가 지원되는 경우에는 자동 교체를 사용하고, 자동 교체가 지원되지 않는 경우에는 온디맨드 교체를 사용하며, 자동 교체와 온디맨드 교체가 모두 지원되지 않는 경우에는 수동 교체를 사용하는 것이 좋습니다.

주요 구성 요소 교체 기능을 시연하거나 자동화 스크립트를 검증하기 위해 온디맨드 교체를 수행하는 것이 좋습니다. 계획되지 않은 교체에는 온디맨드 교체를 사용하고 가능하면 사용자 지정 교체 기간과 함께 자동 키 교체를 사용하는 것이 좋습니다.

키 교체의 작동 방식

AWS KMS 키 교체는 투명하고 사용하기 쉽게 설계되었습니다.는 고객 관리형 키에 대해서만 선택적 자동 및 온디맨드 키 교체를 AWS KMS 지원합니다.

자동 키 교체

AWS KMS 는 교체 기간에 정의된 다음 교체 날짜에 KMS 키를 자동으로 교체합니다. 업데이트를 기억하거나 예약할 필요가 없습니다.

자동 키 교체는가 (AWS_KMS 오리진)을 AWS KMS 생성하는 키 구성 요소가 있는 대칭 암호화 KMS 키에서만 지원됩니다.

고객 관리형 KMS 키의 경우 자동 교체는 선택 사항입니다. AWS KMS 는 매년 AWS 관리형 KMS 키의 키 구성 요소를 항상 교체합니다. AWS 소유 KMS 키 교체는 키를 소유 AWS 서비스 한에서 관리합니다.

온디맨드 교체

자동 키 교체 활성화 여부에 관계없이 KMS 키와 연결된 키 구성 요소의 교체를 즉시 시작합니다.

온디맨드 키 교체는 (AWS_KMS 오리진)을 AWS KMS 생성하는 키 구성 요소가 있는 대칭 암호화 KMS 키와 가져온 키 구성 요소가 있는 대칭 암호화 KMS 키(EXTERNAL오리진)에서 지원됩니다.

수동 교체

다음 유형의 KMS 키는 자동 또는 온디맨드 교체를 지원하지 않지만 이러한 KMS 키는 수동으로 교체할 수 있습니다.

- 비대칭 KMS 키
- HMAC KMS 키
- 사용자 지정 키 스토어의 KMS 키

키 구성 요소 관리

AWS KMS 는 키 교체가 비활성화된 경우에도 AWS_KMS 오리진이 있는 KMS 키의 모든 키 구성 요소를 유지합니다.는 KMS 키를 AWS KMS 삭제할 때만 키 구성 요소를 삭제합니다.

오리진이 EXTERNAL인 대칭 암호화 키의 키 구성 요소는 사용자가 관리합니다. DeleteImportedKeyMaterial 작업을 사용해 키 구성 요소를 삭제하거나, 키 구성 요소를 가져올 때 만료 시간을 설정할 수 있습니다. KMS 키는 연결된 키 구성 요소 중 어느 하나라도 만료되거나 삭제되는 즉시 사용 불가능하게 됩니다.

키 구성 요소 사용

교체된 KMS 키를 사용하여 데이터를 암호화하는 경우는 현재 키 구성 요소를 AWS KMS 사용합니다. 교체된 KMS 키를 사용하여 암호화 텍스트를 해독하면 AWS KMS 는 해당 데이터를 암호화하는 데 사용한 키 구성 요소와 같은 버전을 사용합니다. 복호화 작업을 위해 특정 버전의 키 구성 요소를 선택할 수 없으며,는 AWS KMS 자동으로 올바른 버전을 선택합니다.

교체 기간

교체 기간은 키 구성 요소를 교체 AWS KMS 하는 자동 키 교체를 활성화한 후의 일수와 이후 각 자동 키 교체 사이의 일수를 정의합니다. 자동 키 교체를 활성화할 때 RotationPeriodInDays 값을 지정하지 않으면 기본값은 365일입니다.

kms:RotationPeriodInDays 조건 키를 사용하면 위탁자가 RotationPeriodInDays 파라미터에 지정할 수 있는 값을 더욱 제한할 수 있습니다.

교체 날짜

교체 날짜는 자동(예약) 교체 또는 온디맨드 키 교체로 인해 해당 KMS 키의 현재 키 구성 요소가 업데이트된 날짜를 나타냅니다.

교체 날짜

AWS KMS 는 교체 기간에 정의된 교체 날짜에 KMS 키를 자동으로 교체합니다. 기본 교체 기간은 365일입니다.

고객 관리형 키

고객 관리 키에서는 자동 키 교체가 선택 사항이며 언제든지 활성화 및 비활성화할 수 있으므로 교체 날짜는 가장 최근에 교체가 활성화된 날짜에 따라 달라집니다. 이전에 자동 키 교체를 활성화한 키의 교체 기간을 수정하면 날짜가 변경될 수 있습니다. 교체 날짜는 키의 수명 기간 동안 여러 번 변경할 수 있습니다.

예를 들어, 2022년 1월 1일에 고객 관리형 키를 생성하고 2022년 3월 15일에 기본 교체 기간을 365일로 자동 키 회전을 활성화하면 AWS KMS 는 2023년 3월 15일, 2024년 3월 15일, 그리고 그 이후로는 365일마다 키 구성 요소를 교체합니다.

다음 예제에서는 자동 키 교체가 기본 교체 기간인 365일로 활성화되었다고 가정합니다. 이러한 예제에서는 키의 교체 기간에 영향을 미칠 수 있는 특수 사례를 보여줍니다.

- 키 교체 비활성화 - 언제든지 자동 키 교체를 비활성화하면 KMS 키는 교체가 비활성화되었을 때 사용하던 키 구성 요소 버전을 계속 사용합니다. 자동 키 교체를 다시 활성화하면는 새 AWS KMS 교체 활성화 날짜를 기준으로 키 구성 요소를 교체합니다.
- 비활성화된 KMS 키 - KMS 키가 비활성화된 동안는 키를 교체 AWS KMS 하지 않습니다. 하지만 키 교체 상태는 변경되지 않으며 KMS 키가 비활성인 동안에는 해당 상태를 변경할 수 없습니다. KMS 키가 다시 활성화되면 키 구성 요소가 마지막으로 예약된 교체 날짜를 초과하면가 즉시 AWS KMS 키를 교체합니다. 키 구성 요소가 마지막으로 예약된 교체 날짜를 놓치지 않은 경우는 원래 키 교체 일정을 AWS KMS 재개합니다.
- 삭제 보류 중인 KMS 키 - KMS 키가 삭제 보류 중인 동안에는이 키를 교체 AWS KMS 하지 않습니다. 키 교체 상태는 `false`로 설정되며 삭제가 보류 중인 동안에는 해당 상태를 변경할 수 없습니다. 삭제가 취소되면 이전의 키 교체 상태가 복원됩니다. 키 구성 요소가 마지막으로 예약된 교체 날짜를 초과한 경우는 이를 즉시 AWS KMS 교체합니다. 키 구성 요소가 마지막으로 예약된 교체 날짜를 놓치지 않은 경우는 원래 키 교체 일정을 AWS KMS 재개합니다.

AWS 관리형 키

AWS KMS 는 AWS 관리형 키 매년(약 365일) 자동으로 교체됩니다. AWS 관리형 키에 대한 키의 교체를 사용 또는 사용 중지할 수 없습니다.

의 키 구성 요소는 생성 날짜로부터 1년 후에 처음 교체되고 그 이후에는 매년(마지막 교체로부터 약 365일) 교체 AWS 관리형 키 됩니다.

> **Note**
>
> 2022년 5월에의 교체 일정을 3년마다(약 1,095일) AWS 관리형 키 에서 1년마다(약 365일)로 AWS KMS 변경했습니다.

AWS 소유 키

AWS 소유 키에 대한 키의 교체를 사용 또는 사용 중지할 수 없습니다. 의 키 교체 전략은 키를 생성하고 관리하는 AWS 서비스에 의해 AWS 소유 키 결정됩니다. 자세한 내용은 서비스에 대한 사용 설명서 또는 개발자 안내서의 저장 시 암호화 주제를 참조하세요.

다중 리전 키 교체

교체 동작은 키 구성 요소가 AWS KMS (AWS_KMS 오리진)에 의해 생성되는지 아니면 (EXTERNAL오리진)에 의해 생성되는지에 따라 달라집니다.

AWS_KMS 오리진이 있는 다중 리전 키

오리진이 AWS_KMS인 대칭 암호화 다중 리전 키에서 키 구성 요소 자동 교체를 활성화 및 비활성화하고 온디맨드 교체를 수행할 수 있습니다. 키 교체는 다중 리전 키의 공유 속성입니다.

기본 키에서만 자동 키 교체를 활성화 및 비활성화합니다. 프라이머리 키에서만 온디맨드 교체를 시작합니다.

- 가 다중 리전 키를 AWS KMS 동기화하면 기본 키에서 모든 관련 복제본 키로 키 교체 속성 설정을 복사합니다.
- 가 키 구성 요소를 AWS KMS 교체하면 기본 키에 대한 새 키 구성 요소를 생성한 다음 리전 경계를 넘어 모든 관련 복제본 키에 새 키 구성 요소를 복사합니다. 키 구성 요소는 암호화 AWS KMS 되지 않은 상태로 두지 않습니다. 이 단계는 암호화 작업에 키가 사용되기 전에 키 구성 요소가 완전히 동기화되도록 신중하게 제어됩니다.
- AWS KMS 는 기본 키와 모든 복제본 키에서 해당 키 구성 요소를 사용할 수 있을 때까지 새 키 구성 요소로 데이터를 암호화하지 않습니다.
- 교체된 기본 키를 복제하면 새 복제 키에는 현재 키 구성 요소와 관련된 다중 리전 키에 대한 모든 이전 버전의 키 구성 요소가 포함됩니다.

이 패턴은 관련된 다중 리전 키가 완벽하게 상호 운용되도록 보장합니다. 다중 리전 키는 키가 생성되기 전에 암호문을 암호화한 경우에도 관련 다중 리전 키로 암호화된 모든 암호문을 해독할 수 있습니다.

EXTERNAL 오리진이 있는 다중 리전 키

EXTERNAL 오리진이 있는 대칭 암호화 다중 리전 키에서 키 구성 요소의 온디맨드 교체를 수행할 수 있습니다. 키 교체는 다중 리전 키의 공유 속성입니다.

새 키 구성 요소를 기본 키와 각 복제본 키로 가져온 후에만 기본 키에 대해 온디맨드 교체를 시작합니다.

- 새 키 구성 요소를 기본 키로 가져오면는 키 구성 요소 식별자와 키 구성 요소 설명을 기본 키에서 모든 관련 복제본 키로 AWS KMS 복사합니다. 키 구성 요소는 복사하지 않습니다.
- 동일한 키 구성 요소를 각 복제본 키로 개별적으로 가져와야 합니다. 키 구성 요소를 모든 관련 다중 리전 키로 가져온 후 기본 키에서 온디맨드 교체를 시작할 수 있습니다. 이렇게 하면 해당 키 구성 요소를 기본 키와 모든 복제본 키에서 사용할 수 있을 때까지가 새 키 구성 요소로 데이터를 암호화하지 AWS KMS 않습니다.
- 기본 키 또는 복제본 키의 각 키 구성 요소는 동일한 키 또는 기타 관련 다중 리전 키의 다른 키 구성 요소와 독립적으로 만료되거나 삭제될 수 있습니다.

AWS 서비스

AWS 서비스의 서버 측 암호화에 사용하는 고객 관리형 키에서 자동 키 교체를 활성화할 수 있습니다. 연간 교체는 투명하며 AWS 서비스와 호환됩니다.

키 교체 모니터링

가 AWS 관리형 키 또는 고객 관리형 키의 키 구성 요소를 AWS KMS 교체하면 Amazon EventBridge에 `KMS CMK Rotation` 이벤트를 쓰고 로그에 RotateKey 이벤트를 씁니다 AWS CloudTrail . 이러한 레코드를 사용하여 KMS 키가 교체되었는지 확인할 수 있습니다.

AWS Key Management Service 콘솔을 사용하여 남은 온디맨드 교체 횟수와 KMS 키에 대해 완료된 모든 키 구성 요소 교체 목록을 볼 수 있습니다.

ListKeyRotations 작업을 사용하여 완료된 교체의 세부 정보를 볼 수 있습니다.

최종 일관성

키 교체에는 다른 AWS KMS 관리 작업과 동일한 최종 일관성 효과가 적용됩니다. AWS KMS전체에서 새 키 자료를 사용할 수 있게 되기까지 약간의 지연이 있을 수 있습니다. 그러나 키 구성 요소를 교체해도 암호화 작업이 중단되거나 지연되지는 않습니다. 현재 키 구성 요소는 AWS KMS전체에서 새 키 구성 요소를 사용할 수 있을 때까지 암호화 작업에 사용됩니다. 다중 리전 키의 키 구성

요소가 자동으로 교체되면는 관련 다중 리전 키가 있는 모든 리전에서 새 키 구성 요소를 사용할 수 있을 때까지 현재 키 구성 요소를 AWS KMS 사용합니다.

자동 키 교체를 활성화하려면

기본적으로 KMS 키에 대해 자동 키 교체를 활성화하면는 매년 KMS 키에 대한 새 암호화 구성 요소를 AWS KMS 생성합니다. 또한 사용자 지정을 지정rotation-period하여 키 구성 요소를 AWS KMS 교체할 자동 키 교체를 활성화한 후의 일수와 이후 각 자동 교체 사이의 일수를 정의할 수 있습니다.

자동 키 교체에는 다음과 같은 이점이 있습니다.

- 키를 교체할 때 키 ID, 키 ARN, 리전, 정책 및 권한을 포함한 KMS 키의 속성은 변경되지 않습니다.
- KMS 키 ID 또는 키 ARN을 참조하는 애플리케이션이나 별칭을 변경할 필요가 없습니다.
- 키 구성 요소를 교체해도 AWS 서비스에서의 KMS 키 사용에 영향을 미치지 않습니다.
- 키 교체를 활성화하면는 AWS KMS 교체 기간에 정의된 다음 교체 날짜에 KMS 키를 자동으로 교체합니다. 업데이트를 기억하거나 예약할 필요가 없습니다.

AWS KMS 콘솔에서 또는 EnableKeyRotation 작업을 사용하여 자동 키 교체를 활성화할 수 있습니다. 자동 키 교체를 활성화하려면 `kms:EnableKeyRotation` 권한이 필요합니다. AWS KMS 권한에 대한 자세한 내용은 단원을 참조하십시오권한 참조.

AWS KMS 콘솔 사용

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 탐색 창에서 고객 관리형 키를 선택합니다. (AWS 관리형 키의 교체를 활성화하거나 비활성화할 수 없습니다. 매년 자동으로 교체됩니다.)
4. KMS 키의 별칭 또는 키 ID를 선택합니다.
5. 키 교체(Key rotation) 탭을 선택합니다.

   키 교체 탭은 다중 리전 대칭 암호화 KMS 키를 포함하여 AWS KMS 가 생성한 키 구성 요소가 있는 대칭 암호화 KMS 키의 세부 정보 페이지에만 나타납니다(오리진은 AWS_KMS).

   비대칭 KMS 키, HMAC KMS 키, 가져온 키 구성 요소가 있는 KMS 키 또는 사용자 지정 키 스토어의 KMS 키는 자동으로 교체할 수 없습니다. 그러나 이들을 수동으로 교체할 수 있습니다.

6. 자동 키 교체 섹션에서 편집을 선택합니다.
7. 키 교체에서 활성화를 선택합니다.

> **Note**
>
> KMS 키가 비활성화되거나 삭제 보류 중인 경우 AWS KMS 는 키 구성 요소를 교체하지 않으며 자동 키 교체 상태 또는 교체 기간을 업데이트할 수 없습니다. KMS 키를 활성화하거나 삭제를 취소하여 자동 키 교체 구성을 업데이트합니다. 자세한 내용은 키 교체의 작동 방식 및 키의 AWS KMS 키 상태 섹션을 참조하세요.

8. (선택 사항) 90~2,560일의 교체 기간을 입력합니다. 기본값은 365일입니다. 사용자 지정 교체 기간을 지정하지 않으면 AWS KMS 는 매년 키 구성 요소를 교체합니다.

   kms:RotationPeriodInDays 조건 키를 사용하여 위탁자가 교체 기간에 지정할 수 있는 값을 제한할 수 있습니다.
9. 저장을 선택합니다.

AWS KMS API 사용

AWS Key Management Service (AWS KMS) API를 사용하여 자동 키 교체를 활성화 및 비활성화하고 고객 관리형 키의 현재 교체 상태를 볼 수 있습니다. 이 예제들은 AWS Command Line Interface (AWS CLI)를 사용하지만, 사용자는 어떤 지원되는 프로그래밍 언어라도 사용할 수 있습니다.

EnableKeyRotation 작업은 지정된 KMS 키에 대한 자동 키 교체를 활성화합니다. 이 작업에서 KMS 키를 식별하려면 해당 키 ID 또는 키 ARN을 사용합니다. 기본적으로 고객 관리형 KMS 키에 대한 키 교체는 비활성화됩니다.

kms:RotationPeriodInDays 조건 키를 사용하여 위탁자가 `EnableKeyRotation` 요청의 `RotationPeriodInDays` 파라미터에 지정할 수 있는 값을 제한할 수 있습니다.

다음 예제에서는 지정된 대칭 암호화 KMS 키에 대해 교체 기간이 180일인 키 교체를 활성하고 GetKeyRotationStatus 작업을 사용하여 결과를 확인합니다.

```
$ aws kms enable-key-rotation \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --rotation-period-in-days 180

$ aws kms get-key-rotation-status --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
{
```

```
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyRotationEnabled": true,
    "RotationPeriodInDays": 180,
    "NextRotationDate": "2024-02-14T18:14:33.587000+00:00"
}
```

자동 키 교체 비활성화

고객 관리형 키에서 자동 키 교체를 활성화한 후 언제든지 비활성화하도록 선택할 수 있습니다.

자동 키 교체를 비활성화하면 KMS 키는 교체가 비활성화되었을 때 사용하던 키 구성 요소 버전을 계속 사용합니다. 자동 키 교체를 다시 활성화하면는 새 AWS KMS 교체 활성화 날짜를 기준으로 키 구성 요소를 교체합니다.

자동 교체를 비활성화해도 온디맨드 교체를 수행하는 기능에 영향을 주지 않으며 진행 중인 온디맨드 교체도 취소되지 않습니다.

AWS KMS 콘솔에서 또는 DisableKeyRotation 작업을 사용하여 자동 키 교체를 비활성화할 수 있습니다. 자동 키 교체를 비활성화하려면 `kms:DisableKeyRotation` 권한이 필요합니다. AWS KMS 권한에 대한 자세한 내용은 단원을 참조하십시오권한 참조.

AWS KMS 콘솔 사용

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 탐색 창에서 고객 관리형 키를 선택합니다. (AWS 관리형 키의 교체를 활성화하거나 비활성화할 수 없습니다. 매년 자동으로 교체됩니다.)
4. KMS 키의 별칭 또는 키 ID를 선택합니다.
5. 키 교체(Key rotation) 탭을 선택합니다.

   키 교체 탭은 다중 리전 대칭 암호화 KMS 키를 포함하여 AWS KMS 가 생성한 키 구성 요소가 있는 대칭 암호화 KMS 키의 세부 정보 페이지에만 나타납니다(오리진은 AWS_KMS).

   비대칭 KMS 키, HMAC KMS 키, 가져온 키 구성 요소가 있는 KMS 키 또는 사용자 지정 키 스토어의 KMS 키는 자동으로 교체할 수 없습니다. 그러나 이들을 수동으로 교체할 수 있습니다.
6. 자동 키 교체 섹션에서 편집을 선택합니다.

7. 키 교체에서 비활성화를 선택합니다.

> **Note**
>
> KMS 키가 비활성화되거나 삭제 보류 중인 경우 AWS KMS 는 키 구성 요소를 교체하지 않으며 자동 키 교체 상태 또는 교체 기간을 업데이트할 수 없습니다. KMS 키를 활성화하거나 삭제를 취소하여 자동 키 교체 구성을 업데이트합니다. 자세한 내용은 키 교체의 작동 방식 및 키의 AWS KMS 키 상태 섹션을 참조하세요.

8. 저장을 선택합니다.

AWS KMS API 사용

AWS Key Management Service (AWS KMS) API를 사용하여 자동 키 교체를 비활성화하고 고객 관리형 CMK의 현재 교체 상태를 볼 수 있습니다. 이 예제는 AWS Command Line Interface (AWS CLI)를 사용하지만, 사용자는 지원되는 어떤 프로그래밍 언어라도 사용할 수 있습니다.

DisableKeyRotation 작업은 자동 키 교체를 비활성화합니다. 이 작업에서 KMS 키를 식별하려면 해당 키 ID 또는 키 ARN을 사용합니다. 기본적으로 고객 관리형 KMS 키에 대한 키 교체는 비활성화됩니다.

다음 예제에서는 지정된 대칭 암호화 KMS 키에 대해 자동 키 교체를 비활성하고 GetKeyRotationStatus 작업을 사용하여 결과를 확인합니다.

```
$ aws kms disable-key-rotation --key-id 1234abcd-12ab-34cd-56ef-1234567890ab

$ aws kms get-key-rotation-status --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
{
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyRotationEnabled": false
}
```

온디맨드 키 교체 수행

자동 키 교체 활성화 여부에 관계없이 고객 관리형 KMS 키에서 키 구성 요소의 온디맨드 교체를 수행할 수 있습니다. 자동 교체를 비활성화(DisableKeyRotation)해도 온디맨드 교체를 수행하는 기능에 영향을 주지 않으며 진행 중인 온디맨드 교체도 취소되지 않습니다. 온디맨드 교체는 기존 자동 교체 일정을 변경하지 않습니다. 예를 들어 자동 키 교체가 활성화되고 교체 기간이 730일인 KMS 키를 가정해

보겠습니다. 키가 2024년 4월 14일에 자동으로 교체되도록 예정되어 있고 2024년 4월 10일에 온디맨드 교체를 수행하는 경우, 키는 2024년 4월 14일 및 그 이후 730일마다 예정대로 자동으로 교체됩니다.

온디맨드 키 교체는 KMS 키당 최대 25회 수행할 수 있습니다. AWS KMS 콘솔을 사용하여 KMS 키에 사용할 수 있는 나머지 온디맨드 교체 횟수를 볼 수 있습니다.

온디맨드 키 교체는 대칭 암호화 KMS 키에서만 지원됩니다. 사용자 지정 키 스토어에서 비대칭 KMS 키, HMAC KMS 키 또는 KMS 키의 온디맨드 교체를 수행할 수 없습니다. 관련 다중 리전 키 세트의 온디맨드 교체를 수행하려면 프라이머리 키에서 온디맨드 교체를 간접적으로 호출합니다.

`kms:RotateKeyOnDemand` 및 `kms:GetKeyRotationStatus` 권한이 있는 권한 있는 사용자는 AWS KMS 콘솔 및 AWS KMS API를 사용하여 온디맨드 키 교체를 시작하고 키 교체 상태를 볼 수 있습니다. ListKeyRotations를 사용해 KMS 키에 대해 완료된 교체 내역을 확인하세요.

주제

- 온디맨드 키 교체 시작(콘솔)
- 온디맨드 키 교체(AWS KMS API) 시작

온디맨드 키 교체 시작(콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 탐색 창에서 고객 관리형 키를 선택합니다. (AWS 관리형 키의 교체를 온디맨드로 수행할 수 없습니다. 키는 매년 자동으로 교체됩니다.)
4. KMS 키의 별칭 또는 키 ID를 선택합니다.
5. 키 구성 요소 및 교체 탭을 선택합니다.

   키 구성 요소 및 교체 탭은 자동 또는 온디맨드 교체를 지원하는 대칭 암호화 KMS 키의 상세 페이지에만 표시됩니다. 여기에는에서 AWS KMS 생성한 키 구성 요소가 있는 KMS 키(AWS_KMS 오리진)와 가져온 키 구성 요소가 있는 KMS 키(외부 오리진)가 포함됩니다.

   사용자 지정 키 스토어에서는 비대칭 KMS 키, HMAC KMS 키 또는 KMS 키의 온디맨드 교체를 수행할 수 없습니다. 그러나 이들을 수동으로 교체할 수 있습니다.
6. 지금 교체를 선택하세요. 가져온 키 구성 요소가 있는 대칭 암호화 키의 경우 지금 교체 옵션은 이전에 새 키 구성 요소를 가져왔고 교체 보류 중 상태인 경우에만 사용할 수 있습니다.

> **Note**
>
> 다중 리전 키의 경우 기본 리전 키만 교체할 수 있습니다.

7. 경고와 키의 나머지 온디맨드 교체 횟수에 대한 정보를 읽고 고려합니다. 교체 후 현재 상태가 될 키 구성 요소의 ID, 설명, 만료 시간 등의 정보도 확인할 수 있습니다. 온디맨드 교체를 진행하지 않기로 결정한 경우 취소를 선택합니다.
8. 키 교체를 선택하여 온디맨드 교체를 확인합니다.

> **Note**
>
> 온디맨드 교체에는 다른 AWS KMS 관리 작업과 동일한 최종 일관성 효과가 적용됩니다. AWS KMS전체에서 새 키 자료를 사용할 수 있게 되기까지 약간의 지연이 있을 수 있습니다. 콘솔 상단의 배너에서 온디맨드 교체가 완료되는 시기를 알려줍니다.

온디맨드 키 교체(AWS KMS API) 시작

AWS Key Management Service (AWS KMS) API를 사용하여 온디맨드 키 교체를 시작하고 고객 관리형 키의 현재 교체 상태를 볼 수 있습니다. 이 예제는 AWS Command Line Interface (AWS CLI)를 사용하지만, 사용자는 지원되는 어떤 프로그래밍 언어라도 사용할 수 있습니다.

RotateKeyOnDemand 작업은 지정된 KMS 키에 대한 온디맨드 키 교체를 즉시 시작합니다. 이러한 작업에서 KMS 키를 식별하려면 해당 키 ID 또는 키 ARN을 사용합니다.

다음 예제에서는 지정된 대칭 암호화 KMS 키에서 온디맨드 키 교체를 시작하고 GetKeyRotationStatus 작업을 사용하여 온디맨드 교체가 진행 중인지 확인합니다. `kms:GetKeyRotationStatus` 응답의 `OnDemandRotationStartDate`는 진행 중인 온디맨드 교체가 시작된 날짜와 시간을 식별합니다. 이 예시에서 KMS 키는 365일 주기의 자동 교체도 활성화되어 있습니다.

```
$ aws kms rotate-key-on-demand --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
{
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
}

$ aws kms get-key-rotation-status --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
{
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
```

```
    "KeyRotationEnabled": true,
    "NextRotationDate": "2024-03-14T18:14:33.587000+00:00",
    "OnDemandRotationStartDate": "2024-02-24T18:44:48.587000+00:00"
    "RotationPeriodInDays": 365
}
```

KMS 키가 자동 교체를 지원하지 않거나 자동 교체가 활성화되어 있지 않은 경우, `kms:GetKeyRotationStatus` 응답에는 다음 예시와 같이 표시되는 필드가 더 적습니다.

```
$ aws kms rotate-key-on-demand --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
{
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
}

$ aws kms get-key-rotation-status --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
{
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyRotationEnabled": false,
    "OnDemandRotationStartDate": "2024-02-24T18:44:48.587000+00:00"
}
```

교체 및 키 구성 요소 나열

자동 또는 온디맨드 교체를 지원하는 KMS 키에는 여러 개의 키 구성 요소가 연결될 수 있습니다. 이러한 키에는 자동 또는 온디맨드 교체마다 초기 키 구성 요소와 하나의 추가 키 구성 요소가 있습니다.

`kms:ListKeyRotations` 권한이 있는 승인된 사용자는 AWS KMS 콘솔과 ListKeyRotations API를 사용해 자동 및 온디맨드 교체가 완료된 키 구성 요소를 포함하여 KMS 키와 연결된 모든 키 구성 요소를 나열할 수 있습니다.

주제

- 교체 및 키 구성 요소 나열(콘솔)
- 교체 및 키 구성 요소 나열(AWS KMS API)

교체 및 키 구성 요소 나열(콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.

3. 탐색 창에서 고객 관리형 키를 선택합니다.
4. KMS 키의 별칭 또는 키 ID를 선택합니다.
5. 키 구성 요소 및 교체 탭을 선택합니다.

- 키 구성 요소 및 교체 탭은 자동 또는 온디맨드 교체를 지원하는 대칭 암호화 KMS 키의 상세 페이지에만 표시됩니다. 여기에는가 AWS KMS 생성한 키 구성 요소가 있는 KMS 키(AWS_KMS 오리진)와 가져온 키 구성 요소가 있는 KMS 키(EXTERNAL오리진)가 포함됩니다.
- 키 구성 요소 및 교체 탭의 키 구성 요소 테이블에는 KMS 키와 연결된 모든 키 구성 요소가 표시됩니다. 각 키 구성 요소의 경우, 해당 항목에는 AWS KMS가 할당한 고유 식별자, 교체 날짜, 키 구성 요소 상태가 표시됩니다. 교체 날짜는 자동 또는 온디맨드 키 교체 후 해당 키 구성 요소가 현재 키 구성 요소가 된 시점을 나타냅니다. 첫 번째 또는 Pending rotation 키 구성 요소에는 교체 날짜가 연결되지 않습니다. 키 구성 요소 상태는가 키 구성 요소를 AWS KMS 사용하는 방법을 결정합니다. 현재 키 구성 요소는 암호화와 복호화에 모두 사용됩니다. 현재가 아닌 키 구성 요소는 복호화에만 사용됩니다. 키 구성 요소 상태가 Pending rotation이면 키 구성 요소가 교체를 위해 준비된 상태임을 의미합니다. 이 키 구성 요소는 온디맨드 키 교체가 해당 키 구성 요소를 현재 키 구성 요소로 만들기 전까지 어떠한 암호화 작업에도 사용되지 않습니다. 키 구성 요소에 대해 표시되는 추가 정보는 KMS 키 유형에 따라 달라집니다.
- 오리진이 AWS_KMS인 대칭 암호화 KMS 키의 경우, 각 행에는 교체 유형(On-demand 또는 Automatic)도 표시됩니다.
- 가져온 키 구성 요소(EXTERNAL 오리진)가 있는 대칭 암호화 KMS 키는 On-demand 교체만 지원하므로 교체 유형 열이 없습니다. 대신 각 행에는 가져오기 상태, 사용자가 지정한 설명, 만료 정보, 그리고 작업 메뉴가 표시됩니다. 가져오기 상태는 내부에서 키 구성 요소를 사용할 수 있음을 나타내는 가져오기 AWS KMS 또는 내부에서 키 구성 요소를 사용할 수 없음을 나타내는 보류 중인 가져오기입니다 AWS KMS. 작업 메뉴를 사용해 가져온 키 구성 요소를 삭제하거나 다시 가져올 수 있습니다. 키 구성 요소의 가져오기 상태가 가져오기 보류 중인 경우 키 구성 요소 삭제 작업이 비활성화됩니다. 키 구성 요소 다시 가져오기 작업은 항상 사용할 수 있습니다. 키 구성 요소가 만료되거나 삭제될 때까지 기다릴 필요 없이 다시 가져올 수 있습니다.

교체 및 키 구성 요소 나열(AWS KMS API)

AWS Key Management Service (AWS KMS) API를 사용하여 온디맨드 키 교체를 시작하고 고객 관리형 키의 현재 교체 상태를 볼 수 있습니다. 이 예제는 AWS Command Line Interface (AWS CLI)를 사용하지만, 사용자는 지원되는 어떤 프로그래밍 언어라도 사용할 수 있습니다.

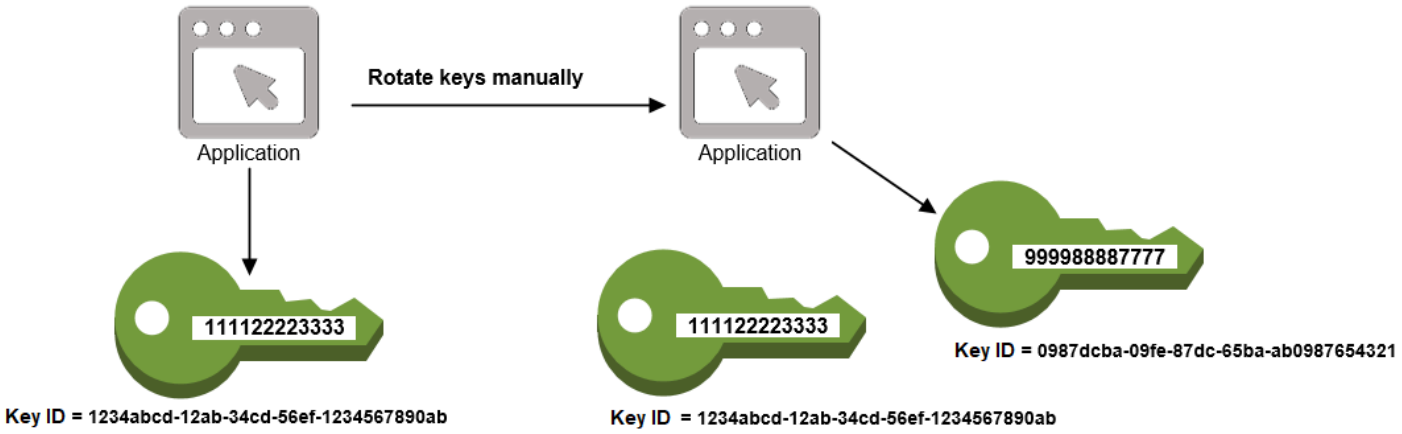
ListKeyRotations 작업은 지정된 KMS 키에 대한 모든 교체와 키 구성 요소를 나열합니다. 이러한 작업에서 KMS 키를 식별하려면 해당 키 ID 또는 키 ARN을 사용합니다.

이 작업은 선택적 IncludeKeyMaterial 파라미터를 지원합니다. 이 파라미터의 기본값은 ROTATIONS_ONLY입니다. 이 파라미터를 생략하면 자동 또는 온디맨드 키 교체로 생성된 키 구성 요소에 대한 정보가 AWS KMS 반환됩니다. 값을 ALL_KEY_MATERIAL로 지정하면 AWS KMS 는 응답에 첫 번째 키 구성 요소와 교체를 기다리는 가져온 키 구성 요소를 추가합니다. 이 파라미터는 자동 또는 온디맨드 키 교체를 지원하는 KMS 키에서만 사용할 수 있습니다.

```
$ aws kms list-key-rotations --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --inlcude-key-material ALL_KEY_MATERIAL
{
    "Rotations": [
        {
            "KeyId": 1234abcd-12ab-34cd-56ef-1234567890ab,
            "KeyMaterialId":
 123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0,
            "KeyMaterialDescription": "KeyMaterialA",
            "ImportState": "PENDING_IMPORT",
            "KeyMaterialState": "NON_CURRENT"
        },
        {
            "KeyId": 1234abcd-12ab-34cd-56ef-1234567890ab,
            "KeyMaterialId":
 96083e4fb6dbc41d77578a213a6b6669c044dd4c143e96755396d2bf11fd6068,
            "ImportState": "IMPORTED",
            "KeyMaterialState": "CURRENT",
            "ExpirationModel": "KEY_MATERIAL_DOES_NOT_EXPIRE",
            "RotationDate": "2025-05-01T15:50:51.045000-07:00",
            "RotationType": "ON_DEMAND"
        }
    ],
    "Truncated": false
}
```

수동으로 키 교체

자동 또는 온디맨드 교체 기능을 사용하는 대신 새 KMS 키를 생성해 현재 KMS 키를 대체하는 방법을 선택할 수도 있습니다. 새 KMS 키에 현재 KMS 키와 다른 암호화 구성 요소가 있는 경우 새 KMS 키를 사용하면 기존 KMS 키에서 키 구성 요소를 변경하는 것과 동일한 효과가 있습니다. 한 KMS 키를 다른 KMS 키로 교체하는 프로세스를 수동 키 교체라고 합니다.



수동 교체는 사용자 지정 키 스토어의 비대칭 KMS 키, HMAC KMS 키 및 KMS 키와 같이 자동 또는 온디맨드 키 교체에 적합하지 않은 KMS 키를 교체하려는 경우에 적합합니다.

> **Note**
>
> 새 KMS 키를 사용하기 시작할 때는가 원래 KMS 키가 암호화한 데이터를 해독할 AWS KMS 수 있도록 원래 KMS 키를 활성화된 상태로 유지해야 합니다.

KMS 키를 수동으로 교체하는 경우 애플리케이션에서 KMS 키 ID 또는 키 ARN에 대한 참조도 업데이트해야 합니다. 친숙한 이름을 KMS 키와 연결하는 별칭을 사용하면 이 프로세스를 더 쉽게 수행할 수 있습니다. 별칭을 사용하여 애플리케이션에서 KMS 키를 참조합니다. 그런 다음 애플리케이션에서 사용하는 KMS 키를 변경하려면 애플리케이션 코드를 편집하는 대신 별칭의 대상 KMS 키를 변경합니다. 자세한 내용은 애플리케이션에서 별칭을 사용하는 방법 알아보기을 참조하세요.

> **Note**
>
> 수동으로 교체된 KMS 키의 최신 버전을 가리키는 별칭은 DescribeKey, GetPublicKey 및 DeriveSharedSecret, Encrypt, GenerateDataKey, GenerateDataKeyPair, GenerateMac, VerifyMac, Sign, Verify와 같은 암호화 작업에 적합한 해결책입니다. DisableKey또는ScheduleKeyDeletion과 같이 KMS 키를 관리하는 작업에서는 별칭을 사용할 수 없습니다. 수동으로 교체된 대칭 암호화 KMS 키에서 Decrypt 작업을 호출할 때 명령에서 `KeyId` 파라미터를 생략합니다. AWS KMS 는 사이퍼텍스트를 암호화한 KMS 키를 자동으로 사용합니다. 비대칭 KMS 키로 `Decrypt` 또는 Verify를 호출하거나 HMAC KMS 키로 VerifyMac를 호출할 때 `KeyId` 파라미터가 필요합니다. 키를 수동으로 교체하는 경우와 같이 `KeyId` 파라미터 값이

암호화 작업을 수행한 KMS 키를 더 이상 가리키지 않는 별칭인 경우 이러한 요청이 실패합니다. 이 오류를 방지하려면 각 작업에 대해 올바른 KMS 키를 추적하고 지정해야 합니다.

별칭의 대상 KMS 키를 변경하려면 AWS KMS API에서 UpdateAlias 작업을 사용합니다. 예를 들어, 이 명령은 새 KMS 키를 가리키도록 `alias/TestKey` 별칭을 업데이트합니다. 작업이 출력을 반환하지 않기 때문에 예제에서는 ListAliases 작업을 사용하여 별칭이 이제 다른 KMS 키와 연결되고 `LastUpdatedDate` 필드가 업데이트되었음을 보여줍니다. ListAliases 명령은의 query 파라미터를 사용하여 AWS CLI `alias/TestKey`별칭만 가져옵니다.

```
$ aws kms list-aliases --query 'Aliases[?AliasName==`alias/TestKey`]'
{
    "Aliases": [
        {
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/TestKey",
            "AliasName": "alias/TestKey",
            "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
            "CreationDate": 1521097200.123,
            "LastUpdatedDate": 1521097200.123
        },
    ]
}


$ aws kms update-alias --alias-name alias/TestKey --target-key-id
 0987dcba-09fe-87dc-65ba-ab0987654321

$ aws kms list-aliases --query 'Aliases[?AliasName==`alias/TestKey`]'
{
    "Aliases": [
        {
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/TestKey",
            "AliasName": "alias/TestKey",
            "TargetKeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
            "CreationDate": 1521097200.123,
            "LastUpdatedDate": 1604958290.722
        },
    ]
}
```

다중 리전 키 세트에서 프라이머리 키 변경

관련된 다중 리전 키에는 기본 키가 있어야 합니다. 그러나 기본 키를 변경할 수 있습니다. 이 작업(기본 리전 업데이트)은 현재 기본 키를 복제본 키로 변환하고 관련 복제본 키 중 하나를 기본 키로 변환합니다. 복제본 키를 유지 관리하는 동안 현재 기본 키를 삭제하거나 키 관리자와 동일한 리전에서 기본 키를 찾아야 하는 경우 이 작업을 수행할 수 있습니다.

관련 복제 키를 새 기본 키로 선택할 수 있습니다. 작업이 시작될 때 기본 키와 복제본 키 모두 `Enabled` 키 상태에 있어야 합니다.

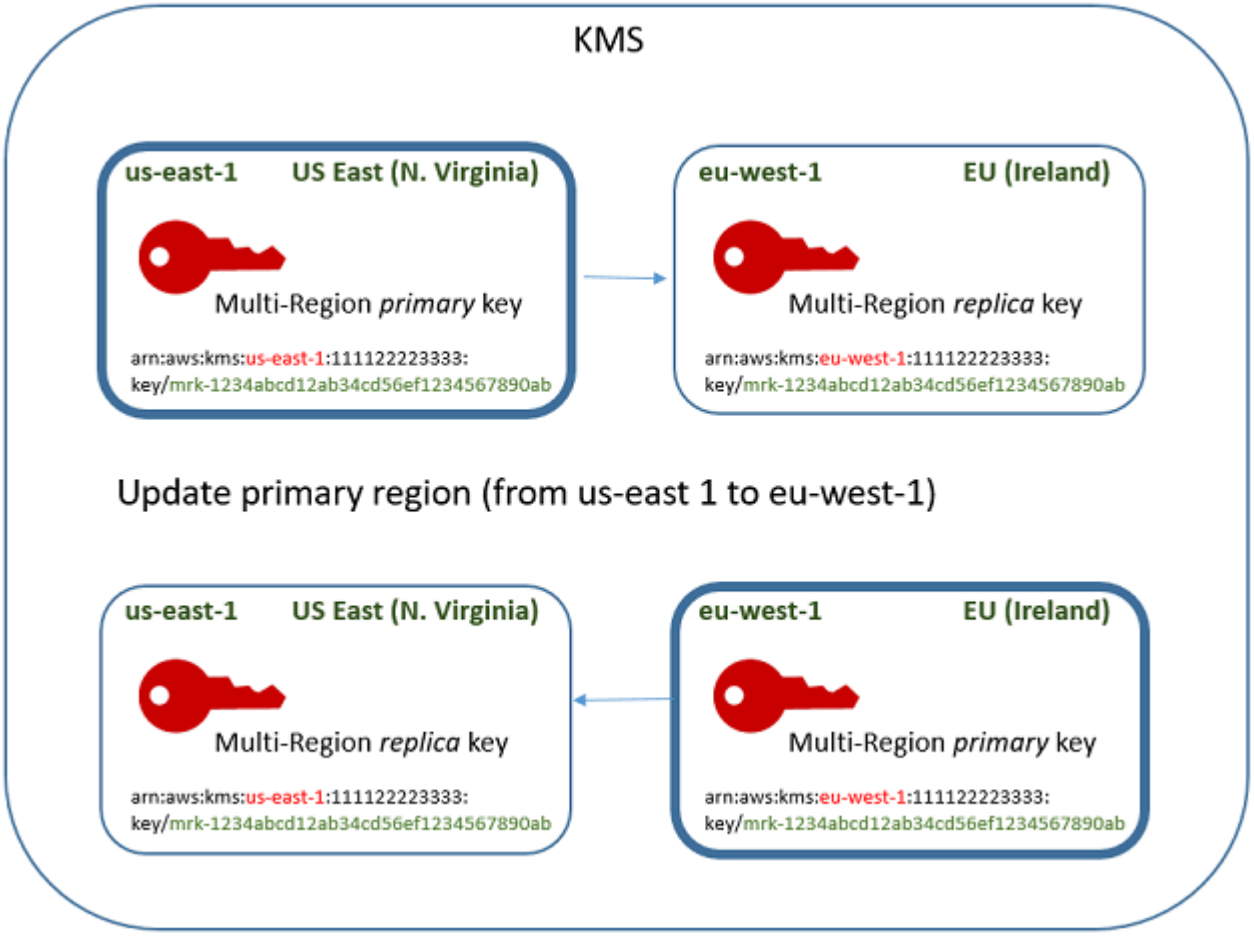
`Updating` 키 상태

`UpdatePrimaryRegion` 작업이 완료된 후에도 기본 리전을 업데이트하는 프로세스가 몇 초 더 진행 중일 수 있습니다. 이 시간 동안 이전 기본 키와 새 기본 키의 임시 키 상태는 업데이트 중입니다. 키 상태가 `Updating`인 동안에는 암호화 작업에서 키를 사용할 수 있지만 새 기본 키를 복제하거나 이러한 키 활성화 또는 비활성화와 같은 특정 관리 작업을 수행할 수 없습니다. DescribeKey와 같은 작업은 이전 기본 키와 새 기본 키를 모두 복제본으로 표시할 수 있습니다. `Enabled` 키 상태는 업데이트가 완료되면 복원됩니다.

`Updating` 키 상태의 효과에 대한 정보는 키의 AWS KMS 키 상태 단원을 참조하십시오.

작동 방식

기본 키가 미국 동부(버지니아 북부)(us-east-1) 있고 복제 키가 유럽(아일랜드)(eu-west-1)에 있다고 가정합니다. 업데이트 기능을 사용하여 미국 동부(버지니아 북부)(us-east-1)의 기본 키를 복제 키로 변경하고 유럽(아일랜드)(eu-west-1)의 복제 키를 기본 키로 변경할 수 있습니다.



업데이트 프로세스가 완료되면 유럽 (아일랜드)(eu-west-1) 리전의 다중 리전 키가 다중 리전 기본 키이고 미국 동부(버지니아 북부)(us-east-1) 리전의 키는 복제 키입니다. 다른 관련 복제 키가 있는 경우 새 기본 키의 복제본이 됩니다. 는 다음에 다중 리전 키의 공유 속성을 AWS KMS 동기화할 때 새 기본 키에서 공유 속성을 가져와서 이전 기본 키를 포함한 복제본 키에 복사합니다.

업데이트 작업은 다중 리전 키의 키 ARN에 영향을 주지 않습니다. 또한 키 구성 요소와 같은 공유 특성이나 키 정책과 같은 독립 특성에는 영향을 주지 않습니다. 그러나 새 기본 키의 키 정책을 업데이트하고자 할 수 있습니다. 예를 들어 신뢰할 수 있는 보안 주체에 대한 kms:ReplicateKey 새 기본 키에 추가하고 새 복제본 키에서 제거할 수 있습니다.

기본 리전 업데이트

복제본 키를 프라이머리 키로 변환하여 이전 프라이머리 키를 복제본으로 변경할 수 있습니다. 기본 리전을 업데이트하려면 두 리전 모두에서 kms:UpdatePrimaryRegion 권한이 필요합니다.

AWS KMS 콘솔에서 또는 UpdatePrimaryRegion 작업을 사용하여 기본 리전을 업데이트할 수 있습니다.

AWS KMS 콘솔 사용

AWS KMS 콘솔에서 기본 키를 업데이트할 수 있습니다. 현재 기본 키에 대한 키 세부 정보 페이지에서 시작합니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 탐색 창에서 고객 관리형 키를 선택합니다.
4. 다중 리전 기본 키의 키 ID 또는 별칭을 선택합니다. 그러면 기본 키에 대한 키 세부 정보 페이지가 열립니다.

   다중 리전 기본 키를 식별하려면 오른쪽 상단 모서리에 있는 도구 아이콘을 사용하여 테이블에 리전 구분(Regionality) 열을 추가합니다.
5. 리전 구분(Regionality) 탭을 선택합니다.
6. 기본 키 섹션에서 기본 리전 변경(Change primary Region)을 선택합니다.
7. 새 기본 키의 리전을 선택합니다. 메뉴에서 리전을 하나만 선택할 수 있습니다.

   기본 리전 변경 메뉴에는 관련 다중 리전 키가 있는 리전만 포함됩니다. 메뉴의 모든 리전에서기본 리전을 업데이트할 권한이 없을 수 있습니다.
8. 기본 리전 변경을 선택합니다.

AWS KMS API 사용

관련 다중 지역 키 집합에서 기본 키를 변경하려면 UpdatePrimaryRegion 작업을 수행합니다.

`KeyId` 파라미터를 사용하여 현재 기본 키를 식별합니다. `PrimaryRegion` 파라미터를 사용하여 새 기본 키 AWS 리전 의를 나타냅니다. 기본 키에 새 기본 리전에 복제본이 없는 경우 작업이 실패합니다.

다음 예에서는 기본 키를 `us-west-2` 리전의 다중 리전 키에서 `eu-west-1` 리전의 복제본으로 변경합니다. `KeyId` 파라미터는 `us-west-2` 리전의 현재 기본 키를 식별합니다. `PrimaryRegion` 파라미터는 새 기본 키 AWS 리전 의를 지정합니다`eu-west-1`.

```
$ aws kms update-primary-region \
```

```
      --key-id arn:aws:kms:us-west-2:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab \
      --primary-region eu-west-1
```

성공하면 이 작업은 출력을 반환하지 않고, HTTP 상태 코드만 반환합니다. 효과를 확인하려면 다중 리전 키 중 하나에서 DescribeKey 작업을 호출하십시오. 키 상태가 Enabled로 반환될 때까지 기다릴 수 있습니다. 키 상태가 업데이트 중인 경우 키 값은 여전히 유동적일 수 있습니다.

예를 들어 다음 DescribeKey 호출은 eu-west-1 리전의 다중 리전 키에 대한 세부 정보를 가져옵니다. 출력은 eu-west-1 리전의 다중 리전 키가 이제 기본 키임을 보여줍니다. us-west-2 리전의 관련 다중 리전 키(동일한 키 ID)는 이제 복제본 키입니다.

```
$ aws kms describe-key \
      --key-id arn:aws:kms:eu-west-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab \

{
    "KeyMetadata": {
        "AWSAccountId": "111122223333",
        "KeyId": "mrk-1234abcd12ab34cd56ef1234567890ab",
        "Arn": "arn:aws:kms:eu-west-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
        "CreationDate": 1609193147.831,
        "Enabled": true,
        "Description": "multi-region-key",
        "KeySpec": "SYMMETRIC_DEFAULT",
        "KeyState": "Enabled",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "Origin": "AWS_KMS",
        "KeyManager": "CUSTOMER",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ],
        "MultiRegion": true,
        "MultiRegionConfiguration": {
           "MultiRegionKeyType": "PRIMARY",
           "PrimaryKey": {
              "Arn": "arn:aws:kms:eu-west-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
              "Region": "eu-west-1"
           },
           "ReplicaKeys": [
```

```
            {
               "Arn": "arn:aws:kms:us-west-2:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
               "Region": "us-west-2"
            }
         ]
      }
   }
}
```

삭제 AWS KMS key

를 삭제하면 AWS KMS key 파괴적이고 위험할 수 있습니다. 이렇게 하면 KMS 키와 연결된 키 구성 요소와 모든 메타데이터가 삭제되고, 이 작업은 되돌릴 수 없습니다. KMS 키가 삭제된 후에는 해당 KMS 키로 암호화된 데이터를 더 이상 해독할 수 없습니다. 즉 데이터를 복구할 수 없게 됩니다. (유일한 예외는 다중 리전 복제 키와 가져온 키 구성 요소가 있는 비대칭 및 HMAC KMS 키입니다.) 이 위험은 경고나 오류 없이 사용자가 프라이빗 키가 삭제된 후 복호화할 수 없는 퍼블릭 키로 사이퍼텍스트를 계속 생성할 수 있는 암호화에 사용되는 비대칭 KMS 키에 중요합니다 AWS KMS.

더 이상 사용할 필요가 없다고 확신하는 경우에만 KMS 키를 삭제해야 합니다. 확실하지 않은 경우에는 삭제하는 대신 KMS 키를 비활성화하는 방법을 고려하십시오. 비활성화된 KMS 키를 다시 활성화하고 예약된 KMS 키 삭제를 취소할 수 있지만 삭제된 KMS 키는 복구할 수 없습니다.

고객 관리형 키 삭제만 예약할 수 있습니다. AWS 관리형 키 또는는 삭제할 수 없습니다 AWS 소유 키.

KMS 키를 삭제하기 전에 해당 KMS 키로 암호화된 사이퍼텍스트 수를 알고 싶을 수 있습니다. AWS KMS 는이 정보를 저장하지 않으며 사이퍼텍스트를 저장하지 않습니다. 이 정보를 얻으려면 과거 KMS 키 용도를 확인해야 합니다. 도움이 필요하면 KMS 키의 과거 사용 결정로 이동하세요.

AWS KMS 명시적으로 삭제를 예약하고 필수 대기 기간이 만료되지 않는 한는 KMS 키를 삭제하지 않습니다.

하지만 다음 중 하나 이상의 이유로 KMS 키를 삭제하기로 선택할 수 있습니다.

- 더 이상 필요하지 않은 KMS 키의 키 수명 주기를 완료하기 위해
- 사용하지 않는 KMS 키 유지로 인한 관리 오버헤드와 비용 발생을 방지하기 위해
- KMS 키 리소스 할당량에 반하는 KMS 키 수를 줄이기 위해

> **Note**
>
> 를 닫 AWS 계정으면 KMS 키에 액세스할 수 없게 되며 더 이상 요금이 청구되지 않습니다.

AWS KMS 는 KMS 키 삭제를 예약할 때와 KMS 키가 실제로 삭제될 때 AWS CloudTrail 로그에 항목을 기록합니다.

대기 기간에 대해

KMS 키를 삭제하는 것은 파괴적이고 잠재적으로 위험하기 때문에 대기 기간을 7~30일로 설정해야 AWS KMS 합니다. 기본 대기 기간은 30일입니다.

그러나 실제 대기 기간은 예약한 대기 기간보다 최대 24시간 더 길어질 수 있습니다. KMS 키가 삭제될 실제 날짜 및 시간을 가져오려면 DescribeKey 작업을 사용합니다. 또는 AWS KMS 콘솔의 KMS 키 세부 정보 페이지에 있는 일반 구성 섹션에서 예약된 삭제 날짜를 참조하세요. 시간대를 기록하세요.

이 대기 기간 동안 KMS 키 상태 및 키 상태는 삭제 대기 중(Pending deletion)입니다.

- 삭제 대기 중인 KMS 키는 어떠한 암호화 작업에도 사용할 수 없습니다.
- AWS KMS 는 삭제 보류 중인 KMS 키의 키 구성 요소를 교체하지 않습니다.

대기 기간이 끝나면는 KMS 키, 별칭 및 모든 관련 AWS KMS 메타데이터를 AWS KMS 삭제합니다.

KMS 키 삭제를 예약해도 KMS 키로 암호화된 데이터 키에는 즉각적인 영향이 없을 수 있습니다. 자세한 내용은 사용할 수 없는 KMS 키가 데이터 키에 미치는 영향을 참조하세요.

현재 또는 향후에 KMS 키가 필요하지 않도록 하기 위해 대기 기간을 사용합니다. 사람이나 애플리케이션이 대기 기간 중에 KMS 키를 사용하려고 할 때 경고하도록 Amazon CloudWatch 경보를 구성할 수 있습니다. KMS 키를 복구하기 위해 대기 기간이 종료되기 전에 키 삭제를 취소할 수 있습니다. 대기 기간이 끝나면 키 삭제를 취소할 수 없으며가 KMS 키를 AWS KMS 삭제합니다.

특별 고려 사항

삭제를 위해 키를 예약하기 전에 특수 목적의 KMS 키 삭제와 관련된 다음과 같은 특수한 고려 사항을 검토하세요.

비대칭 KMS 키 삭제

권한이 부여된 사용자는 대칭 또는 비대칭 KMS 키를 삭제할 수 있습니다. 이러한 KMS 키의 삭제를 예약하는 절차는 두 가지 유형의 키에 대해 동일합니다. 그러나 비대칭 KMS 키의 퍼블릭 키는 외부에서 다운로드하여 사용할 수 있으므로 AWS KMS작업은 특히 암호화에 사용되는 비대칭 KMS 키(키 사용:)의 경우 상당한 추가 위험을 초래합니다ENCRYPT_DECRYPT.

- KMS 키 삭제를 예약하면 해당 KMS 키는 키 상태가 삭제 보류 중(Pending deletion)으로 변경되며 암호화 작업에 사용할 수 없습니다. 그러나 삭제 예약은 외부의 퍼블릭 키에는 영향을 주지 않습니다 AWS KMS. 퍼블릭 키가 있는 사용자는 계속해서 해당 키를 사용하여 메시지를 암호화할

수 있습니다. 키 상태가 변경되었다는 알림은 받지 못합니다. 삭제가 취소되지 않으면 해당 퍼블릭 키로 생성된 암호화 텍스트는 해독할 수 없습니다.

- 삭제 보류 중인 KMS 키를 사용하려는 시도를 감지하는 경보, 로그 및 기타 전략은 AWS KMS외부에서의 퍼블릭 키 사용을 감지할 수 없습니다.
- KMS 키가 삭제되면 해당 KMS 키와 관련된 모든 AWS KMS 작업이 실패합니다. 그러나 퍼블릭 키가 있는 사용자는 계속해서 메시지를 암호화하는 데 사용할 수 있습니다. 이러한 암호화 텍스트는 해독할 수 없습니다.

키 사용이 ENCRYPT_DECRYPT인 비대칭 KMS 키를 삭제해야 하는 경우 CloudTrail Log 항목을 사용하여 퍼블릭 키가 다운로드 및 공유되었는지 확인합니다. 그러한 퍼블릭 키가 있을 경우 해당 키가 AWS KMS외부에서 사용되지 않는지 확인합니다. 그런 다음 KMS 키를 삭제하는 대신 비활성화하는 것이 좋습니다.

가져온 키 구성 요소가 있는 비대칭 KMS 키의 경우 비대칭 KMS 키를 삭제하여 발생하는 위험이 완화됩니다. 자세한 내용은 Deleting KMS keys with imported key material을 참조하세요.

다중 리전 키 삭제

기본 키를 삭제하려면 해당 복제 키를 모두 삭제하도록 예약한 다음 복제본 키가 삭제될 때까지 기다려야 합니다. 기본 키를 삭제하는 데 필요한 대기 기간은 마지막 복제 키가 삭제될 때 시작됩니다. 복제 키를 삭제하지 않고 특정 리전에서 기본 키를 삭제해야 하는 경우 기본 리전을 업데이트하여 기본 키를 복제본 키로 변경합니다.

언제든지 복제본 키를 삭제할 수 있습니다. 다른 KMS 키의 키 상태에 의존하지 않습니다. 실수로 복제본 키를 삭제한 경우 동일한 리전에 동일한 프라이머리 키를 복제하여 복제본을 다시 만들 수 있습니다. 새로 생성하는 복제본 키는 원래 복제본 키와 동일한 공유 속성을 갖습니다.

가져온 키 구성 요소가 있는 KMS 키 삭제

키 구성 요소가 있는 KMS 키의 키 구성 요소를 삭제하는 것은 일시적이며 되돌릴 수 있습니다. 키를 복원하려면 키 구성 요소를 다시 가져오세요.

반면 KMS 키 삭제 작업은 되돌릴 수 없습니다. 키 삭제를 예약하고 필요한 대기 기간이 만료되면는 KMS 키, 해당 키 구성 요소 및 KMS 키와 연결된 모든 메타데이터를 AWS KMS 영구적이고 되돌릴 수 없습니다.

하지만 가져온 키 구성 요소가 있는 KMS 키를 삭제할 경우의 위험과 결과는 KMS 키의 유형('키 사양')에 따라 달라집니다.

- 대칭 암호화 키 - 대칭 암호화 KMS 키를 삭제하면 해당 키로 암호화된 나머지 모든 사이퍼텍스트를 복구할 수 없습니다. 동일한 키 구성 요소가 있더라도 삭제된 대칭 암호화 KMS 키의 사이퍼

텍스트를 해독할 수 있는 새로운 대칭 암호화 KMS 키를 생성할 수 없습니다. 각 KMS 키에 고유한 메타데이터는 각 대칭 사이퍼텍스트에 암호화 방식으로 바인딩됩니다. 이 보안 기능은 대칭 사이퍼텍스트를 암호화한 KMS 키만 해당 사이퍼텍스트를 해독할 수 있도록 보장하지만, 동일한 KMS 키를 다시 생성할 수 없도록 합니다.

- 비대칭 및 HMAC 키 - 원래 키 구성 요소가 있는 경우 삭제된 비대칭 또는 HMAC KMS 키와 동일한 암호화 속성을 사용하여 새 KMS 키를 생성할 수 있습니다.는 고유한 보안 기능을 포함하지 않는 표준 RSA 사이퍼텍스트 및 서명, ECC 서명 및 HMAC 태그를 AWS KMS 생성합니다. AWS외부의 HMAC 키 또는 비대칭 키 쌍의 프라이빗 키를 사용할 수도 있습니다.

  동일한 비대칭 또는 HMAC 키 구성 요소를 사용하여 생성한 새 KMS 키는 다른 키 식별자를 갖게 됩니다. 새 키 정책을 생성하고, 별칭을 다시 만들고, 새 키를 참조하도록 기존 IAM 정책 및 권한 부여를 업데이트해야 합니다.

키 스토어에서 KMS AWS CloudHSM 키 삭제

AWS CloudHSM 키 스토어에서 KMS 키 삭제를 예약하면 키 상태가 삭제 보류 중으로 변경됩니다. KMS 키는 삭제 보류 중 상태로 인해 KMS 키를 사용할 수 없게 되는 경우에도 대기 기간 동안사용자 지정 키 스토어의 연결을 해제합니다.. 따라서 대기 시간 동안 언제라도 KMS 키의 삭제를 취소할 수 있습니다.

대기 기간이 만료되면가 KMS 키를 AWS KMS 삭제합니다 AWS KMS. 그런 다음 AWS KMS 는 연결된 AWS CloudHSM 클러스터에서 키 구성 요소를 삭제하기 위해 최선을 다합니다. AWS KMS에서 키 스토어의 연결이 해제된 경우처럼 AWS KMS 가 키 구성 요소를 삭제할 수 없는 경우에는 클러스터에서 수동으로 불필요한 키 구성 요소를 삭제해야 할 수 있습니다.

AWS KMS 는 클러스터 백업에서 키 구성 요소를 삭제하지 않습니다. 에서 KMS 키를 삭제 AWS KMS 하고 AWS CloudHSM 클러스터에서 키 구성 요소를 삭제하더라도 백업에서 생성된 클러스터에는 삭제된 키 구성 요소가 포함될 수 있습니다. 키 구성 요소를 영구적으로 삭제하려면 DescribeKey 작업을 사용하여 KMS 키의 생성 날짜를 식별합니다. 그런 다음, 키 구성 요소가 포함되어 있을 수 있는 모든 클러스터 백업을 삭제합니다.

AWS CloudHSM 키 스토어에서 KMS 키 삭제를 예약하면 KMS 키를 즉시 사용할 수 없게 됩니다 (최종 일관성에 따라 다름). 그러나 KMS 키로 보호되는 데이터 키로 암호화된 리소스는 데이터 키를 복호화하는 등 KMS 키를 다시 사용할 때까지 영향을 받지 않습니다. 이 문제는 영향을 미치며 AWS 서비스, 대부분은 데이터 키를 사용하여 리소스를 보호합니다. 자세한 내용은 사용할 수 없는 KMS 키가 데이터 키에 미치는 영향을 참조하세요.

외부 키 저장소에서 KMS 키 삭제

외부 키 스토어에서 KMS 키를 삭제해도 키 구성 요소로 사용된 외부 키에는 영향을 주지 않습니다.

외부 키 스토어에서 KMS 키 삭제를 예약하면 키 상태가 Pending deletion(삭제 보류 중)으로 변경됩니다. 외부 키 스토어를 연결 해제하여 KMS 키를 사용할 수 없게 되더라도 KMS 키는 대기 기간 내내 Pending deletion(삭제 보류 중) 상태로 유지됩니다. 따라서 대기 시간 동안 언제라도 KMS 키의 삭제를 취소할 수 있습니다. 대기 기간이 만료되면가 KMS 키를 AWS KMS 삭제합니다 AWS KMS.

외부 키 스토어에서 KMS 키 삭제를 예약하면 KMS 키는 즉시 사용할 수 없게 됩니다(최종 일관성에 따라 다름). 그러나 KMS 키로 보호되는 데이터 키로 암호화된 리소스는 데이터 키를 복호화하는 등 KMS 키를 다시 사용할 때까지 영향을 받지 않습니다. 이 문제는 리소스를 보호하기 위해 데이터 키를 사용하는 AWS 서비스에 영향을 미칩니다. 자세한 내용은 사용할 수 없는 KMS 키가 데이터 키에 미치는 영향을 참조하세요.

키 삭제에 대한 액세스 제어

IAM 정책을 사용하여 AWS KMS 권한을 허용하는 경우 관리자 액세스 권한("Action": "*") 또는 AWS KMS 전체 액세스 권한("Action": "kms:*")이 AWS 있는 IAM 자격 증명은 이미 KMS 키 삭제를 예약하고 키를 취소할 수 있습니다. 키 관리자가 키 정책에서 키 삭제를 예약하고 취소할 수 있도록 하려면 AWS KMS 콘솔 또는 AWS KMS API를 사용합니다.

일반적으로 키 관리자만 키 삭제를 예약하거나 취소할 수 있는 권한이 있습니다. 그러나 키 정책 또는 IAM 정책에 `kms:ScheduleKeyDeletion` 및 `kms:CancelKeyDeletion` 권한을 추가하여 다른 IAM ID에 이러한 권한을 부여할 수 있습니다. `kms:ScheduleKeyDeletionPendingWindowInDays` 조건 키를 사용하여 보안 주체가 ScheduleKeyDeletion 요청의 `PendingWindowInDays` 파라미터에 지정할 수 있는 값을 추가로 제한할 수도 있습니다.

키 관리자의 키 삭제 예약 및 취소 허용

AWS KMS 콘솔 사용

키 관리자에게 키 삭제를 예약하고 취소할 수 있는 권한을 부여하려면 다음을 수행하세요.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 탐색 창에서 고객 관리형 키를 선택합니다.
4. 권한을 변경하고 싶은 KMS 키의 별칭 또는 키 ID를 선택합니다.

5. key policy(키 정책) 탭을 선택합니다.
6. 다음 단계는 키 정책의 기본 보기와 정책 보기에 따라 다릅니다. 기본 콘솔 키 정책을 사용하는 경우에만 기본 보기를 사용할 수 있습니다. 이외의 경우에는 정책 보기만 사용할 수 있습니다.

   기본 보기를 사용할 수 있는 경우 Switch to policy view(정책 보기로 전환) 또는 Switch to default view(기본 보기로 전환) 버튼이 Key policy(키 정책) 탭에 나타납니다.

   - 기본 보기에서
     - Key deletion(키 삭제) 아래에서 Allow key administrators to delete this key(키 관리자가 이 키를 삭제하도록 허용합니다.)를 선택합니다.
   - 정책 보기에서
     a. 편집을 선택합니다.
     b. 키 관리자에 대한 정책 설명에서 `kms:ScheduleKeyDeletion` 및 `kms:CancelKeyDeletion` 권한을 `Action` 요소에 추가합니다.

```
{
  "Sid": "Allow access for Key Administrators",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:user/KMSKeyAdmin"},
  "Action": [
    "kms:Create*",
    "kms:Describe*",
    "kms:Enable*",
    "kms:List*",
    "kms:Put*",
    "kms:Update*",
    "kms:Revoke*",
    "kms:Disable*",
    "kms:Get*",
    "kms:Delete*",
    "kms:ScheduleKeyDeletion",
    "kms:CancelKeyDeletion"
  ],
  "Resource": "*"
}
```

     c. 변경 사항 저장을 선택합니다.

AWS KMS API 사용

를 사용하여 키 삭제 AWS Command Line Interface 를 예약하고 취소할 수 있는 권한을 추가할 수 있습니다.

키 삭제를 예약하고 취소할 권한을 추가하려면

1. aws kms get-key-policy 명령을 이용해 기존 키 정책을 검색한 후 정책 문서를 파일에 저장합니다.
2. 원하는 텍스트 편집기에서 정책 문서를 엽니다. 키 관리자에 대한 정책 설명에서 kms:ScheduleKeyDeletion 및 kms:CancelKeyDeletion 권한을 추가합니다. 다음 예는 이 두 권한이 포함된 정책 설명을 보여줍니다.

```
{
  "Sid": "Allow access for Key Administrators",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:user/KMSKeyAdmin"},
  "Action": [
    "kms:Create*",
    "kms:Describe*",
    "kms:Enable*",
    "kms:List*",
    "kms:Put*",
    "kms:Update*",
    "kms:Revoke*",
    "kms:Disable*",
    "kms:Get*",
    "kms:Delete*",
    "kms:ScheduleKeyDeletion",
    "kms:CancelKeyDeletion"
  ],
  "Resource": "*"
}
```

3. aws kms put-key-policy 명령을 사용하여 KMS 키에 키 정책을 적용합니다.

키 삭제 예약

다음 절차에서는 AWS Management Console 및 AWS KMS API를 AWS KMS 사용하여에서 AWS KMS keys (KMS 키)의 키 삭제를 예약하고 키 삭제를 취소하는 방법을 설명합니다.

> **Warning**
>
> KMS 키를 삭제하는 것은 파괴적이며 잠재적으로 위험합니다. 지금뿐 아니라 앞으로도 KMS 키를 더 이상 사용할 필요가 없다고 확신하는 경우에만 진행해야 합니다. 확실하지 않은 경우에는 삭제하는 대신 KMS 키를 비활성화해야 합니다.

KMS 키를 삭제하려면 삭제할 권한이 있어야 합니다. 주요 관리자에게 이러한 권한을 부여하는 방법에 대한 자세한 내용은 키 삭제에 대한 액세스 제어 섹션을 참조하세요. `kms:ScheduleKeyDeletionPendingWindowInDays` 조건 키를 사용하여 최소 대기 기간을 적용하는 등 대기 기간을 추가로 제한할 수도 있습니다.

AWS KMS 는 KMS 키 삭제를 예약할 때와 KMS 키가 실제로 삭제될 때 AWS CloudTrail 로그에 항목을 기록합니다.

AWS KMS 콘솔 사용

에서는 한 번에 여러 KMS 키의 삭제를 예약하고 취소할 AWS Management Console수 있습니다.

키 삭제를 예약하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 탐색 창에서 고객 관리형 키를 선택합니다.

   AWS 관리형 키 또는 AWS 소유 키의 삭제는 예약할 수 없습니다.
4. 삭제하려는 KMS 키 옆의 확인란을 선택합니다.
5. 키 작업(Key actions)과 키 삭제 예약(Schedule key deletion)을 차례로 선택합니다.
6. 경고와 대기 기간 동안 삭제 취소에 대한 정보를 읽고 고려하세요. 경고에는 최근에 암호화 작업에 사용된 선택한 키 수가 포함됩니다. 표에는 각 KMS 키가 마지막으로 사용된 시기를 보여주는 마지막으로 사용된 열이 표시됩니다. 삭제를 취소하려면 페이지 하단에서 취소를 선택합니다.
7. 대기 기간(일)(Waiting period(in days))에 7~30 범위의 일수를 입력합니다.
8. 삭제 중인 KMS 키를 검토합니다.
9. 이러한 키를 삭제하도록 예약하려면 확인 옆에 있는 확인란을 선택합니다. 키는 암호화 작업에 즉시 사용할 수 없게 됩니다. ***<# #>***일의 대기 기간이 지나면 키가 영구적으로 삭제됩니다.
10. 삭제 예약(Schedule deletion)을 선택합니다.

KMS 키 상태가 삭제 보류 중(Pending deletion)으로 변경됩니다.

AWS KMS API 사용

다음 예제와 같이 `aws kms schedule-key-deletion` 명령을 사용하여 고객 관리형 키의 키 삭제를 예약할 수 있습니다.

AWS 관리형 키 또는 삭제를 예약할 수 없습니다 AWS 소유 키.

```
$ aws kms schedule-key-deletion --key-id 1234abcd-12ab-34cd-56ef-1234567890ab --
pending-window-in-days 10
```

성공적으로 사용되면는 다음 예제에 표시된 출력과 같은 출력을 AWS CLI 반환합니다.

```
{
    "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "DeletionDate": 1598304792.0,
    "KeyState": "PendingDeletion",
    "PendingWindowInDays": 10
}
```

키 삭제 취소

삭제를 위해 KMS 키를 예약한 후 삭제 보류 중인 상태에 있는 동안 키 삭제를 취소할 수 있습니다. AWS KMS 콘솔에서 또는 CancelKeyDeletion 작업을 사용하여 키 삭제를 취소할 수 있습니다. KMS 키의 보류 중인 삭제를 취소하면 KMS 키의 키 상태는 `Disabled`입니다. KMS 키 활성화에 대한 자세한 내용은 KMS 키 활성화 및 비활성화 단원을 참조하세요.

AWS KMS 콘솔 사용

키 삭제를 취소하려면

1. https://console.aws.amazon.com/kms AWS KMS 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 탐색 창에서 고객 관리형 키를 선택합니다.
4. 복구할 KMS 키 옆의 확인란을 선택합니다.
5. 키 작업(Key actions)과 키 삭제 취소(Cancel key deletion)를 차례로 선택합니다.

KMS 키 상태가 삭제 보류 중(Pending deletion)에서 비활성(Disabled)으로 변경됩니다. KMS 키를 사용하려면 KMS 키를 활성화해야 합니다.

AWS KMS API 사용

다음 예제와 AWS CLI 같이 `aws kms cancel-key-deletion` 명령을 사용하여에서 키 삭제를 취소합니다.

```
$ aws kms cancel-key-deletion --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

성공적으로 사용되면는 다음 예제에 표시된 출력과 같은 출력을 AWS CLI 반환합니다.

```
{
    "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
}
```

KMS 키 상태가 삭제 보류 중(Pending deletion)에서 비활성(Disabled)으로 변경됩니다. KMS 키를 사용하려면 KMS 키를 활성화해야 합니다.

삭제 보류 중인 KMS 키 사용을 감지하는 경보 생성

AWS CloudTrail, Amazon CloudWatch Logs 및 Amazon Simple Notification Service(Amazon SNS)의 기능을 결합하여 계정의 누군가가 삭제 보류 중인 KMS 키를 사용하려고 할 때 알려주는 Amazon CloudWatch 경보를 생성할 수 있습니다. 이 알림을 받는 경우 KMS 키 삭제를 취소하고 삭제 결정을 재고해 볼 수 있습니다.

다음 절차에서는 "*Key ARN* `is pending deletion`" 오류 메시지가 CloudTrail 로그 파일에 기록될 때마다 알리는 경보를 생성합니다. 이 오류 메시지는 개인이나 애플리케이션이 암호화 작업 시 해당 KMS 키를 사용하려고 했음을 나타냅니다. 이 알림은 오류 메시지와 연결되어 있으므로 `ListKeys`, `CancelKeyDeletion` 및 `PutKeyPolicy` 같은 삭제 보류 중인 KMS 키에 대해 허용된 API 작업을 사용할 때는 트리거되지 않습니다. 이 오류 메시지를 반환하는 AWS KMS API 작업 목록을 보려면 섹션을 참조하세요키의 AWS KMS 키 상태.

수신하는 알림 이메일에 KMS 키 또는 암호화 작업이 나열되지 않습니다. CloudTrail 로그에서 해당 정보를 확인할 수 있습니다. 대신에, 경보 상태가 정상에서 경보로 변경되었다는 내용이 이메일로 보고됩니다. CloudWatch 경보 및 상태 변경에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서의 Amazon CloudWatch 경보 사용을 참조하십시오.

> **Warning**
>
> 이 Amazon CloudWatch 경보는 AWS KMS외부에서의 비대칭 KMS 키의 퍼블릭 키의 사용을 감지할 수 없습니다. 암호를 해독할 수 없는 암호화 텍스트를 만드는 등 퍼블릭 키 암호화에 사용되는 비대칭 KMS 키를 삭제할 때의 특별한 위험에 대한 자세한 내용은 Deleting asymmetric KMS keys 단원을 참조하십시오.

이 절차에서는 삭제 보류 예외의 인스턴스를 찾는 CloudWatch 로그 그룹 지표 필터를 생성합니다. 그런 다음 로그 그룹 지표에 기반하여 CloudWatch 경보를 생성합니다. 로그 그룹 지표 필터에 대한 자세한 내용을 알아보려면 Amazon CloudWatch Logs 사용 설명서의 필터를 사용하여 로그 이벤트에서 지표 생성을 참조하세요.

1. CloudTrail 로그를 구문 분석하는 CloudWatch 지표 필터를 생성합니다.

   다음 필수 값을 사용하여 로그 그룹에 대한 지표 필터 생성의 지침을 따르세요. 다른 필드의 경우 기본값을 그대로 사용하고 요청에 따라 이름을 제공합니다.

Field	값
필터 패턴	`{ $.eventSource = kms* && $.errorMessage = "* is pending deletion."}`
지표 값	1

2. 1단계에서 생성한 지표 필터를 기반으로 CloudWatch 경보를 생성합니다.

   다음 필수 값을 사용하여 로그 그룹 지표 필터를 기반으로 CloudWatch 경보 생성의 지침을 따르세요. 다른 필드의 경우 기본값을 그대로 사용하고 요청에 따라 이름을 제공합니다.

Field	값
지표 필터	1단계에서 생성한 지표 필터 이름입니다.
임계값 유형	정적
조건	*metric-name* 이 1 이상일 때마다

Field	값
경보를 보낼 데이터 포인트	1/1
누락 데이터 처리	누락 데이터 처리에서 양호(임계값을 위반하지 않음)

이 절차를 완료한 후 새 CloudWatch 경보가 ALARM 상태에 도달할 때마다 알림을 받게 됩니다. 이 경보에 대한 알림을 받으면 데이터를 암호화하거나 암호를 해독하기 위해서는 삭제가 예약된 KMS 키가 아직 필요하다는 뜻일 수 있습니다. 이 경우 KMS 키 삭제를 취소하고 삭제 결정을 다시 생각해 볼 수 있습니다.

가져온 키 구성 요소 삭제

언제든 KMS 키에서 가져온 키 구성 요소를 삭제할 수 있습니다. 또한 만료 날짜가 있는 가져온 키 구성 요소가 만료되면는 키 구성 요소를 AWS KMS 삭제합니다. 어느 경우든 키 구성 요소가 삭제되면 KMS 키의 키 상태는 가져오기 보류 중으로 변경되며, KMS 키는 어떠한 암호화 작업에도 사용할 수 없습니다.

대칭 암호화 키에는 여러 키 구성 요소가 연결될 수 있습니다. 이러한 키의 경우 KMS는 각 키 구성 요소에 고유 식별자를 할당합니다. ListKeyRotations API를 사용하여 이러한 키 구성 요소 식별자와 해당 키 구성 요소 상태를 볼 수 있습니다(RotationsListEntry 참조). PENDING_ROTATION 또는의 키 구성 요소 상태는 키 구성 요소가 KMS 키와 영구적으로 연결되지 않았음을 PENDING_MULTI_REGION_IMPORT_AND_ROTATION 나타냅니다. 영구적으로 연결된 키 구성 요소를 삭제하거나 만료하면 키 상태가 가져오기 보류 중으로 변경됩니다. DeleteImportedKeyMaterial API에서 key-material-id 파라미터로 키 구성 요소 식별자를 지정해 특정 키 구성 요소를 삭제할 수 있습니다.

다중 리전 키에 대한 고려 사항

- PENDING_ROTATION 또는 PENDING_MULTI_REGION_IMPORT_AND_ROTATION 상태의 기본 리전 키의 키 구성 요소를 삭제하면 복제본 리전 키의 키 구성 요소도 삭제됩니다.
- 기본 또는 복제본 리전 키에서 키 구성 요소를 삭제하면 해당 특정 키만 영향을 받으며 기타 관련 다중 리전 키는 변경되지 않습니다. 영구적으로 연결된 모든 키 구성 요소가 있는 기본 또는 복제본 리전 키는 암호화 작업에서 계속 사용할 수 있습니다.

> **Warning**
>
> `key-material-id` 파라미터는 선택 사항이며 이를 지정하지 않으면 AWS KMS 는 현재 키 구성 요소를 삭제합니다.

KMS 키를 비활성화하고 권한을 철회하는 것과 함께 키 구성 요소를 삭제하는 것은 KMS 키 사용을 신속하지만 일시적으로 중단하기 위한 전략으로 사용할 수 있습니다. 반대로, 가져온 키 구성 요소가 있는 KMS 키를 삭제하도록 예약하면 KMS 키 사용이 빠르게 중단되기도 합니다. 하지만 대기 기간 동안 삭제가 취소되지 않으면 KMS 키, 연결된 키 구성 요소 및 모든 키 메타데이터가 영구적으로 삭제됩니다. 자세한 내용은 Deleting KMS keys with imported key material을 참조하세요.

키 구성 요소를 삭제하려면 AWS KMS 콘솔 또는 DeleteImportedKeyMaterial API 작업을 사용할 수 있습니다.는 가져온 키 구성 요소를 삭제하고가 만료된 키 구성 요소를 삭제할 때 AWS CloudTrail 로그에 항목을 AWS KMS 기록합니다. AWS KMS

키 구성 요소를 삭제하면 AWS 서비스에 미치는 영향

키 구성 요소를 삭제하면 KMS 키는 즉시 사용할 수 없게 됩니다(최종 일관성에 따라 다름). 그러나 KMS 키로 보호되는 데이터 키로 암호화된 리소스는 데이터 키를 복호화하는 등 KMS 키를 다시 사용할 때까지 영향을 받지 않습니다. 이 문제는 영향을 미치며 AWS 서비스, 대부분 데이터 키를 사용하여 리소스를 보호합니다. 자세한 내용은 사용할 수 없는 KMS 키가 데이터 키에 미치는 영향을 참조하세요.

AWS KMS 콘솔 사용

AWS KMS 콘솔을 사용하여 키 구성 요소를 삭제할 수 있습니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 탐색 창에서 고객 관리형 키를 선택합니다.
4. 다음 중 하나를 수행하세요.

   - 가져온 키 구성 요소가 있는 KMS 키에 대한 확인란을 선택합니다. 키 작업(Key actions), 키 구성 요소 삭제(Delete key material)를 선택합니다. 다중 키 구성 요소가 연결된 대칭 암호화 키의 경우, 이 작업은 현재 키 구성 요소를 삭제합니다.

- 가져온 키 구성 요소가 있는 대칭 암호화 KMS 키의 경우 KMS 키의 별칭 또는 키 ID를 선택합니다. 키 구성 요소 및 교체 탭을 선택합니다. 키 구성 요소 테이블에는 키와 연결된 모든 키 구성 요소가 나열됩니다. 삭제하려는 키 구성 요소에 해당하는 행의 작업 메뉴에서 키 구성 요소 삭제를 선택합니다.

5. 키 구성 요소를 삭제하고자 함을 확인한 후 [Delete key material]을 선택합니다. 키 상태에 해당되는 KMS 키의 상태가 가져오기 보류 중(Pending import)으로 변경됩니다. 삭제된 키 구성 요소가 `PENDING_ROTATION` 상태였던 경우, KMS 키의 상태는 변경되지 않습니다.

AWS KMS API 사용

AWS KMS API를 이용해 키 구성 요소를 삭제하려면 DeleteImportedKeyMaterial 요청을 보냅니다. 다음 예에서는 AWS CLI에서 이 작업을 수행하는 방법을 보여줍니다.

키 구성 요소를 삭제할 KMS 키의 키 ID를 *1234abcd-12ab-34cd-56ef-1234567890ab*로 바꿉니다. KMS 키의 키 ID나 ARN을 사용할 수 있지만, 이 작업에 대한 별칭을 사용할 수 없습니다. 다음 명령은 현재 키 구성 요소를 삭제하며, 이는 해당 키에 연결된 유일한 키 구성 요소일 수도 있습니다.

```
$ aws kms delete-imported-key-material --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

특정 키 구성 요소를 삭제하려면 `key-material-id` 파라미터로 식별된 키 구성 요소를 지정하세요. *123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0*를 삭제하려는 키 구성 요소의 식별자로 교체하세요.

```
$ aws kms delete-imported-key-material --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --key-material-id 123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0
```

데이터 키 생성

데이터 키는 많은 양의 데이터 및 기타 데이터 암호화 키를 포함하여 데이터를 암호화하는 데 사용할 수 있는 대칭 키입니다. 다운로드할 수 없는 대칭 KMS 키와 달리 데이터 키는 AWS KMS외부 사용용으로 반환됩니다.

가 데이터 키를 AWS KMS 생성하면 즉시 사용할 수 있는 일반 텍스트 데이터 키(선택 사항)와 데이터와 함께 안전하게 저장할 수 있는 암호화된 데이터 키 사본을 반환합니다. 데이터를 해독할 준비가 되면 먼저 암호화된 데이터 키를 해독 AWS KMS 해 달라고 요청합니다.

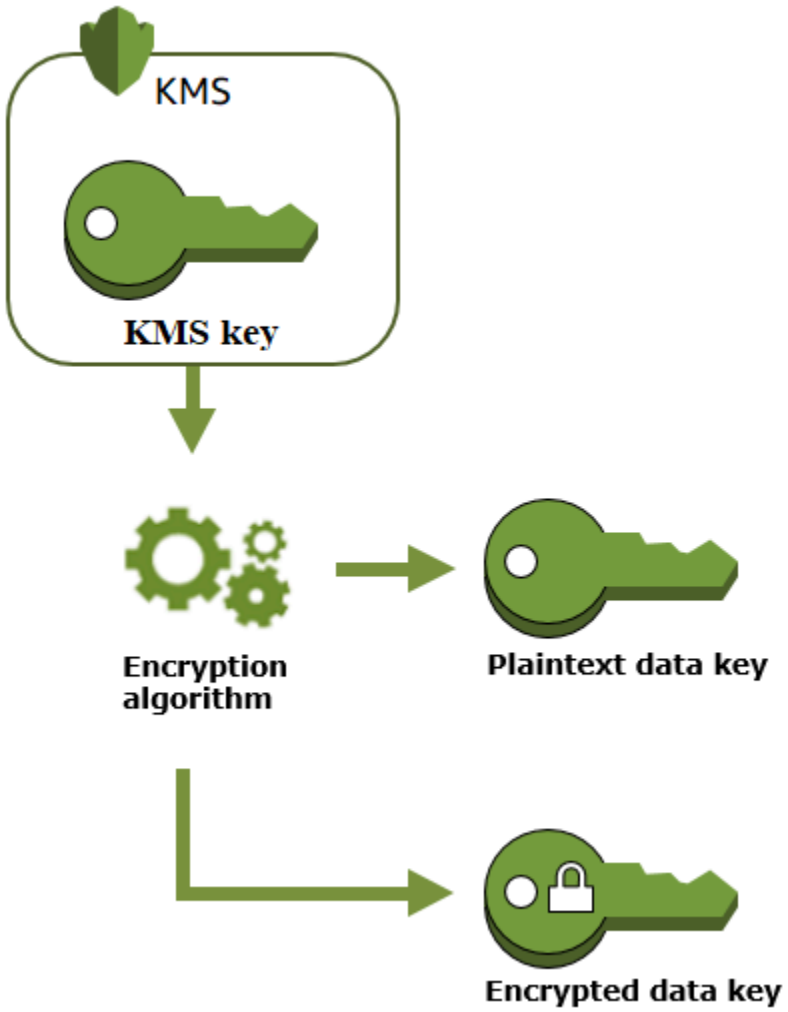
AWS KMS 는 데이터 키를 생성, 암호화 및 해독합니다. 그러나 AWS KMS 는 데이터 키를 저장, 관리 또는 추적하거나 데이터 키를 사용하여 암호화 작업을 수행하지 않습니다. 외부에서 데이터 키를 사용하고 관리해야 합니다 AWS KMS. 데이터 키를 안전하게 사용하는 데 도움이 필요하면 AWS Encryption SDK 섹션을 참조하세요.

주제

- 데이터 키 생성
- 데이터 키를 사용한 암호화 작업의 작동 방식
- 사용할 수 없는 KMS 키가 데이터 키에 미치는 영향

데이터 키 생성

데이터 키를 생성하려면 GenerateDataKey 작업을 호출합니다. 데이터 키를 AWS KMS 생성합니다. 그런 다음 KMS가 사용자에 의해 지정된 대칭 암호화 KMS 키로 데이터 키의 복사본을 암호화합니다. 이 작업은 데이터 키의 일반 텍스트 복사본과 KMS 키로 암호화된 데이터 키 복사본을 반환합니다. 다음 그림은 이 작업을 보여 줍니다.



AWS KMS 는 암호화된 데이터 키만 반환하는 GenerateDataKeyWithoutPlaintext 작업도 지원합니다. 데이터 키를 사용해야 하는 경우 AWS KMS 에 복호화를 요청합니다.

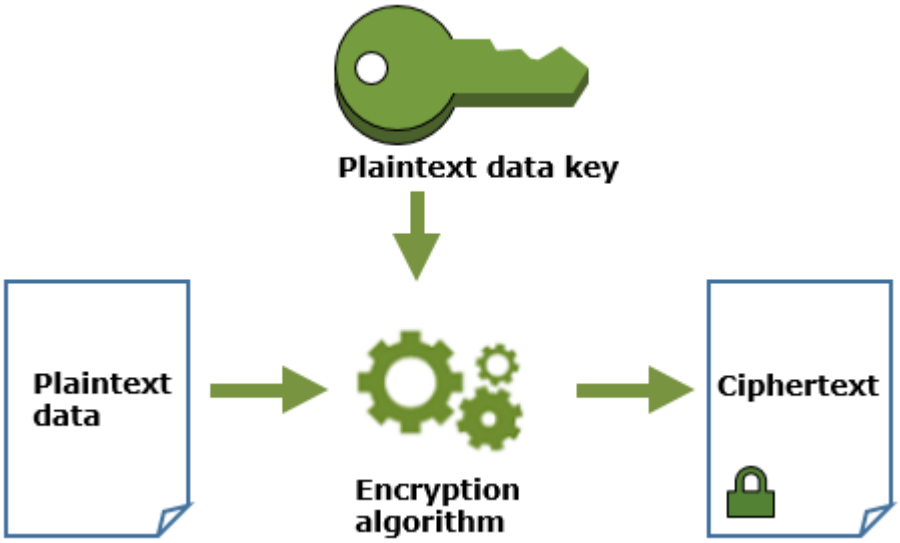
데이터 키를 사용한 암호화 작업의 작동 방식

다음 주제에서는 GenerateDataKey 또는 GenerateDataKeyWithoutPlaintext 작업에서 생성된 데이터 키의 작동 방식을 설명합니다.

데이터 키로 데이터 암호화

AWS KMS 는 데이터 키를 사용하여 데이터를 암호화할 수 없습니다. 그러나 OpenSSL 또는와 같은 암호화 라이브러리를 사용하는 AWS KMS등 외부에서 데이터 키를 사용할 수 있습니다AWS Encryption SDK.

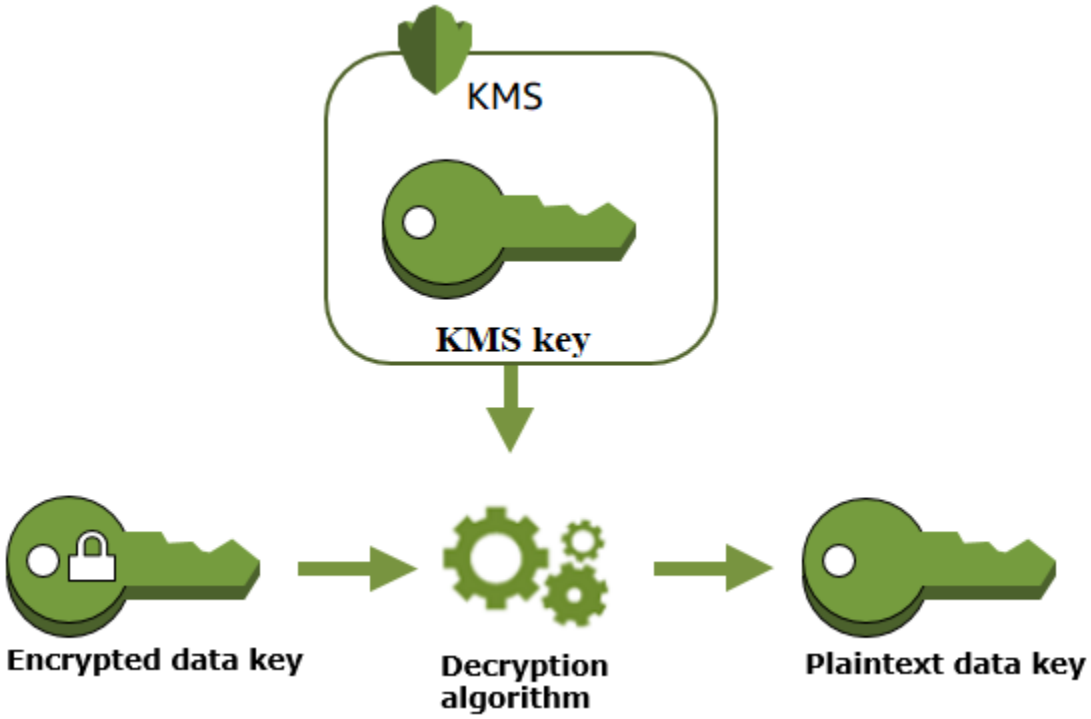
일반 텍스트 데이터 키를 사용하여 데이터를 암호화한 다음에는 가능한 빨리 메모리에서 제거하세요. 데이터 암호화를 해제하는 데 사용할 수 있도록 암호화된 데이터와 함께 암호화된 데이터 키를 안전하게 저장할 수 있습니다.



데이터 키로 데이터 해독

데이터를 복호화하려면 암호화된 데이터 키를 Decrypt 작업에 전달합니다. KMS 키를 AWS KMS 사용하여 데이터 키를 복호화한 다음 일반 텍스트 데이터 키를 반환합니다. 일반 텍스트 데이터 키를 사용하여 데이터를 해독한 다음, 가능한 빨리 메모리에서 일반 텍스트 데이터 키를 제거하세요.

다음 다이어그램은 Decrypt 작업을 사용하여 암호화된 데이터 키를 암호화 해제하는 방법을 보여줍니다.



사용할 수 없는 KMS 키가 데이터 키에 미치는 영향

KMS 키를 사용할 수 없게 되면 효과는 거의 즉각적으로 나타납니다(최종 일관성에 따라 다름). KMS 키의 키 상태는 새로운 조건을 반영하도록 변경되며 암호화 작업에서 KMS 키를 사용하려는 모든 요청은 실패합니다.

그러나 KMS 키로 암호화된 데이터 키와 데이터 키로 암호화된 데이터에 미치는 영향은 데이터 키 복호화 등을 위해 KMS 키를 다시 사용할 때까지 지연됩니다.

수행할 수 있는 다음 작업을 포함하여 다양한 이유로 KMS 키를 사용할 수 없게 될 수 있습니다.

- KMS 키 비활성화
- KMS 키 삭제 예약
- 가져온 키 구성 요소가 있는 KMS 키에서 키 구성 요소 삭제 또는 가져온 키 구성 요소가 만료되도록 허용 오리진이 EXTERNAL인 KMS 키에 다중 키 구성 요소가 연결된 경우, 어느 하나라도 삭제되거나 만료되면 키를 사용할 수 없게 됩니다.
- KMS AWS CloudHSM 키를 호스팅하는 키 스토어의 연결을 해제하거나 KMS 키의 키 구성 요소 역할을 하는 AWS CloudHSM 클러스터에서 키를 삭제합니다.
- KMS 키를 호스팅하는 외부 키 스토어 연결 해제 또는 외부 키 관리자에서 외부 키 삭제를 포함하여 외부 키 스토어 프록시에 대한 암호화 및 복호화 요청을 방해하는 기타 작업

이 효과는 데이터 키를 사용하여 서비스가 관리하는 리소스를 보호하는 많은 AWS 서비스 사용자에게 특히 중요합니다. 다음 예제에서는 Amazon Elastic Block Store(Amazon EBS)와 Amazon Elastic Compute Cloud(Amazon EC2)를 사용합니다. 서로 다른 방식으로 데이터 키를 AWS 서비스 사용합니다. 자세한 내용은 AWS 서비스에 대한 보안 장의 데이터 보호 섹션을 참조하세요.

예를 들어 다음 시나리오를 고려해 보십시오.

1. 암호화된 EBS 볼륨을 생성하고 KMS 키를 지정하여 보호합니다. Amazon EBS가 AWS KMS 에 KMS 키를 사용하여 볼륨에 대한 암호화된 데이터 키를 생성하도록 요청합니다. Amazon EBS는 볼륨의 메타데이터와 함께 암호화된 데이터 키를 저장합니다.
2. EC2 인스턴스에 EBS 볼륨을 연결하면 Amazon EC2는 KMS 키를 사용하여 EBS 볼륨의 암호화된 데이터 키를 복호화합니다. Amazon EC2는 Nitro 하드웨어의 데이터 키를 사용하며, 이 키는 EBS 볼륨에 모든 디스크 I/O를 암호화하는 작업을 담당합니다. EBS 볼륨이 EC2 인스턴스에 연결되어 있는 동안 데이터 키는 Nitro 하드웨어에 유지됩니다.
3. KMS 키를 사용할 수 없게 하는 작업을 수행합니다. 이로 인해 EC2 인스턴스나 EBS 볼륨에 즉시 영향이 미치지 않습니다. Amazon EC2는 KMS 키가 아닌 데이터 키를 사용하여 볼륨이 인스턴스에 연결되어 있는 동안 모든 디스크 I/O를 암호화합니다.
4. 단, 암호화된 EBS 볼륨이 EC2 인스턴스에서 삭제되면 Amazon EBS는 데이터 키를 Nitro 하드웨어에서 제거합니다. 다음에 암호화된 EBS 볼륨을 EC2 인스턴스에 연결할 때 Amazon EBS가 KMS 키를 사용하여 볼륨의 암호화된 데이터 키를 해독하지 못하므로 연결에 실패합니다. EBS 볼륨을 다시 사용하려면 KMS 키를 다시 사용할 수 있게 만들어야 합니다.

데이터 키 페어 생성

비대칭 KMS 키는 데이터 키 페어를 나타냅니다. 데이터 키 페어는 수학적으로 관련된 퍼블릭 키와 프라이빗 키로 구성된 비대칭 데이터 키입니다. 클라이언트 측 암호화 및 복호화, 외부에서 서명 및 확인 AWS KMS또는 두 피어 간에 공유 보안 암호를 설정하는 데 사용하도록 설계되었습니다.

OpenSSL과 같은 도구가 생성하는 데이터 키 페어와 달리는 AWS KMS 지정한의 대칭 암호화 KMS 키로 각 데이터 키 페어의 프라이빗 키를 AWS KMS 보호합니다. 그러나 AWS KMS 는 데이터 키 페어를 저장, 관리 또는 추적하거나 데이터 키 페어로 암호화 작업을 수행하지 않습니다. 따라서 AWS KMS외부에서 데이터 키를 사용하고 관리해야 합니다.

주제

- 데이터 키 페어 생성
- 데이터 키 페어를 사용하는 암호화 작업의 작동 방식

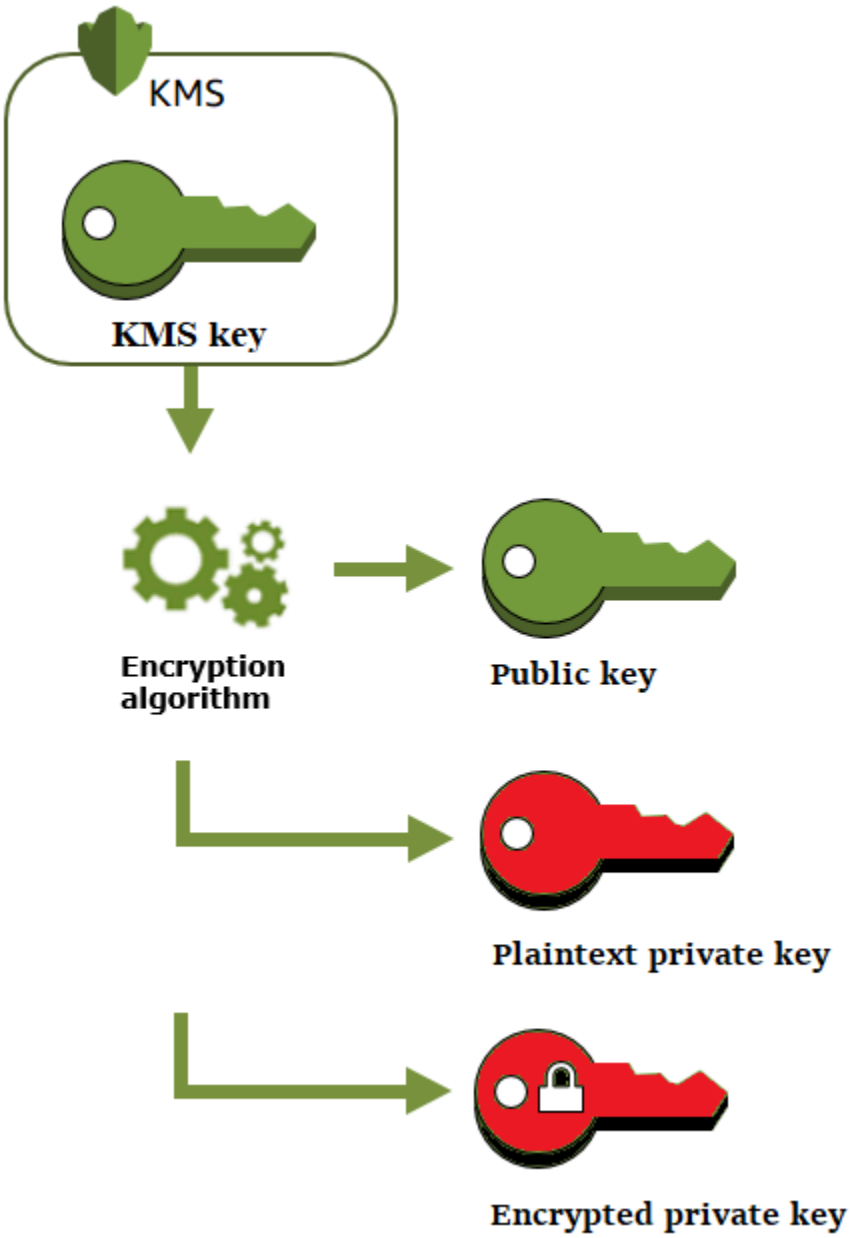
데이터 키 페어 생성

데이터 키 페어를 생성하려면 GenerateDataKeyPair 또는 GenerateDataKeyPairWithoutPlaintext 작업을 호출합니다. 프라이빗 키를 암호화하는 데 사용할 대칭 암호화 KMS 키를 지정합니다.

`GenerateDataKeyPair`는 일반 텍스트 퍼블릭 키, 일반 텍스트 프라이빗 키 및 암호화된 프라이빗 키를 반환합니다. 디지털 서명을 생성하는 경우와 같이 일반 텍스트 프라이빗 키가 즉시 필요할 때 이 작업을 사용합니다.

`GenerateDataKeyPairWithoutPlaintext`는 일반 텍스트 퍼블릭 키 및 암호화된 프라이빗 키를 반환하지만 일반 텍스트 프라이빗 키는 반환하지 않습니다. 퍼블릭 키로 암호화하는 경우와 같이 일반 텍스트 프라이빗 키가 즉시 필요하지 않은 경우 이 작업을 사용합니다. 나중에 데이터를 암호 해독하기 위해 일반 텍스트 프라이빗 키가 필요할 때 Decrypt 작업을 호출 할 수 있습니다.

다음 그림은 `GenerateDataKeyPair` 작업을 보여 줍니다.
`GenerateDataKeyPairWithoutPlaintext` 작업은 일반 텍스트 프라이빗 키를 생략합니다.



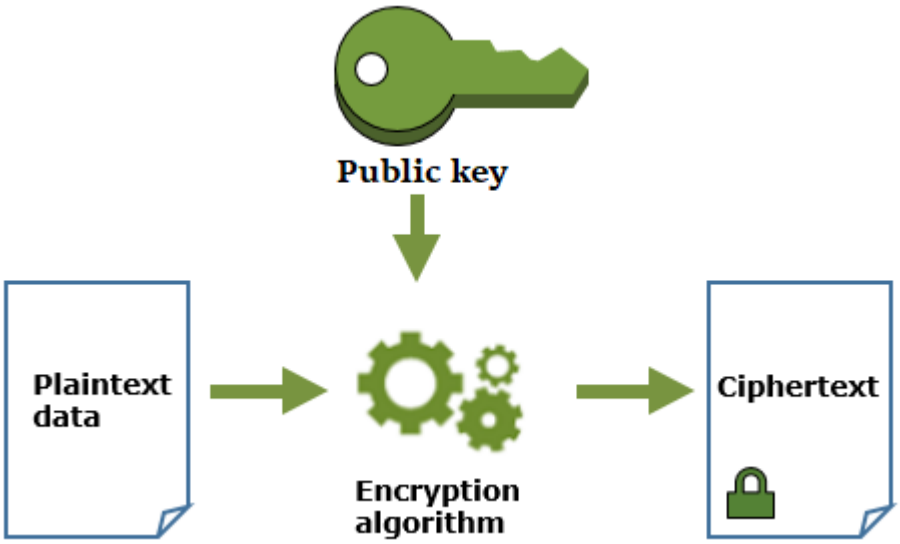
데이터 키 페어를 사용하는 암호화 작업의 작동 방식

다음 주제에서는 GenerateDataKeyPair 또는 GenerateDataKeyPairWithoutPlaintext 작업에서 생성된 데이터 키 페어로 수행할 수 있는 암호화 작업과 이 작업의 작동 방식을 설명합니다.

데이터 키 페어로 데이터 암호화

데이터 키 페어를 사용하여 암호화하는 경우 해당 페어의 퍼블릭 키를 사용하여 데이터를 암호화하고 동일한 페어의 프라이빗 키를 사용하여 데이터를 해독합니다. 일반적으로 데이터 키 페어는 많은 당사자가 프라이빗 키를 보유한 당사자만 해독할 수 있는 데이터를 암호화해야 할 때 사용됩니다.

퍼블릭 키가 있는 당사자는 다음 다이어그램과 같이 해당 키를 사용하여 데이터를 암호화합니다.

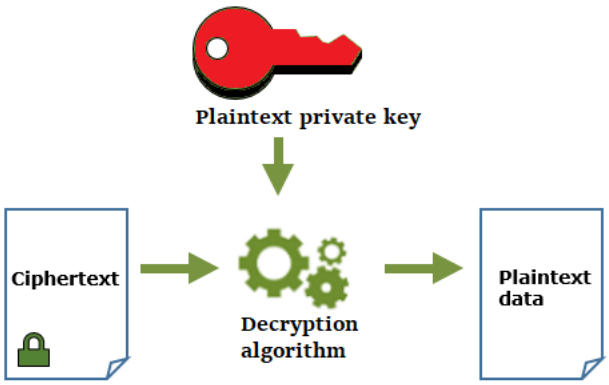


데이터 키 페어로 데이터 암호 해독

데이터를 암호 해독하려면 데이터 키 페어의 프라이빗 키를 사용합니다. 작업이 성공하려면 퍼블릭 키와 프라이빗 키가 동일한 데이터 키 페어에 속해야 하며 동일한 암호화 알고리즘을 사용해야 합니다.

암호화된 프라이빗 키를 해독하려면 Decrypt 작업에 키를 전달합니다. 일반 텍스트 프라이빗 키를 사용하여 데이터를 암호 해독합니다. 그런 다음 가능한 한 빨리 메모리에서 일반 텍스트 프라이빗 키를 제거하세요.

다음 다이어그램은 데이터 키 페어의 프라이빗 키를 사용하여 암호화 텍스트를 해독하는 방법을 보여 줍니다.



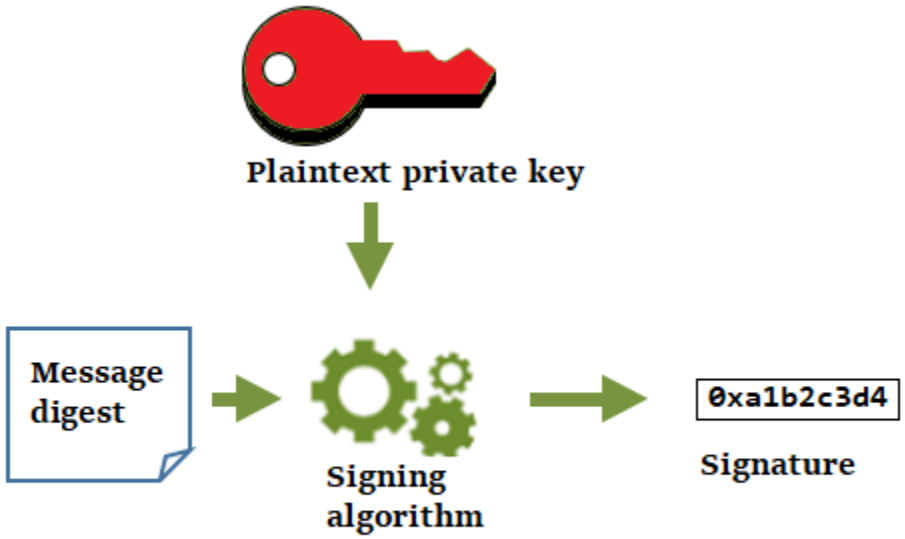
데이터 키 페어를 사용하여 메시지 서명

메시지에 대한 암호화 서명을 생성하려면 데이터 키 페어의 프라이빗 키를 사용합니다. 퍼블릭 키를 가진 사람은 누구나 이 키를 사용하여 메시지가 프라이빗 키로 서명되었는지, 서명 이후 변경되지 않았는지 확인할 수 있습니다.

프라이빗 키를 암호화하는 경우 암호화된 프라이빗 키를 Decrypt 작업에 전달합니다. KMS 키를 AWS KMS 사용하여 데이터 키를 복호화한 다음 일반 텍스트 프라이빗 키를 반환합니다. 일반 텍스트 프라이빗 키를 사용하여 서명을 생성합니다. 그런 다음 가능한 한 빨리 메모리에서 일반 텍스트 프라이빗 키를 제거하세요.

메시지에 서명하려면 OpenSSL의 `dgst` 명령과 같은 암호화 해시 함수를 사용하여 메시지 다이제스트를 생성합니다. 그런 다음 일반 텍스트 프라이빗 키를 서명 알고리즘에 전달합니다. 결과는 메시지의 내용을 나타내는 서명입니다. (먼저 다이제스트를 만들지 않고도 짧은 메시지에 서명할 수 있습니다. 최대 메시지 크기는 사용하는 서명 도구에 따라 다릅니다.)

다음 다이어그램은 데이터 키 페어의 프라이빗 키를 사용하여 메시지에 서명하는 방법을 보여줍니다.

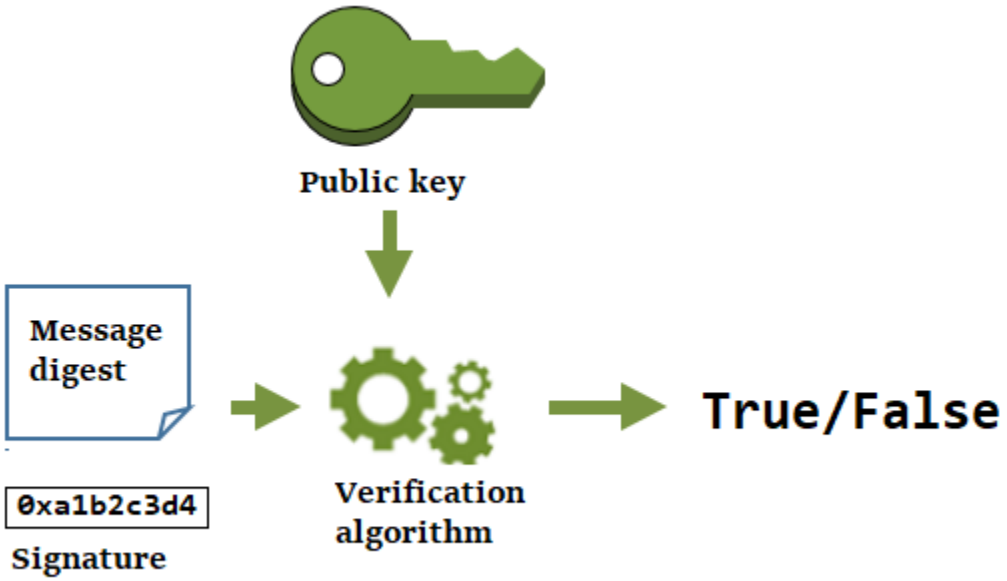


데이터 키 페어를 사용하여 서명 확인

데이터 키 페어의 퍼블릭 키가 있는 사람은 누구나 이 키를 사용하여 프라이빗 키로 생성한 서명을 확인할 수 있습니다. 확인은 권한이 부여된 사용자가 지정된 프라이빗 키 및 서명 알고리즘을 사용하여 메시지에 서명했으며 서명된 이후 메시지가 변경되지 않았음을 확인합니다.

확인이 성공하려면 서명을 확인하는 당사자가 동일한 유형의 다이제스트를 생성하고 동일한 알고리즘을 사용하며 메시지에 서명하는 데 사용되는 프라이빗 키에 해당하는 퍼블릭 키를 사용해야 합니다.

다음 다이어그램은 데이터 키 페어의 퍼블릭 키를 사용하여 메시지 서명을 확인하는 방법을 보여 줍니다.



데이터 키 페어로 공유 보안 비밀 도출

키 계약을 사용하면 각각 타원 곡선 퍼블릭 프라이빗 키 페어가 있는 두 피어가 안전하지 않은 채널을 통해 공유 비밀을 설정할 수 있습니다. 공유 보안 비밀을 도출하려면 두 피어가 안전하지 않은 통신 채널(예: 인터넷)을 통해 퍼블릭 키를 교환해야 합니다. 그런 다음, 각 당사자는 프라이빗 키와 피어의 퍼블릭 키를 사용하여 키 계약 알고리즘으로 동일한 공유 보안 암호를 계산합니다. 공유 비밀 값을 사용하여 두 피어 간에 전송되는 데이터를 암호화하고 복호화하거나 HMAC를 생성하고 확인할 수 있는 대칭 키를 도출할 수 있습니다.

> **Note**
>
> AWS KMS 에서는 공유 비밀을 도출하는 데 사용하기 전에 수신한 퍼블릭 키가 예상 당사자로부터 왔는지 확인할 것을 강력히 권장합니다.

퍼블릭 키를 사용하여 오프라인 작업 수행

비대칭 KMS 키에서 프라이빗 키는에서 생성 AWS KMS 되며 암호화되지 않은 상태로 두 AWS KMS 지 않습니다. 프라이빗 키를 사용하려면를 호출해야 합니다 AWS KMS. AWS KMS API 작업을 호출 AWS KMS 하여 내에서 퍼블릭 키를 사용할 수 있습니다. 또는 퍼블릭 키를 다운로드하여 외부에서 사용할 수 있도록 공유할 수 있습니다 AWS KMS.

퍼블릭 키를 공유하여 외부의 다른 사용자가 데이터를 암호화 AWS KMS 하도록 할 수 있습니다.이 경우 프라이빗 키로만 복호화할 수 있습니다. 또는 사용자가 프라이빗 키로 생성한 디지털 서명을 AWS KMS 에서 다른 사용자가 확인하도록 허용할 수 있습니다. 또는 퍼블릭 키를 피어와 공유하여 공유 비밀을 도출할 수 있습니다.

내에서 비대칭 KMS 키에 퍼블릭 키를 사용하면 모든 AWS KMS 작업의 일부인 인증, 권한 부여 및 로깅의 AWS KMS이점을 누릴 수 있습니다. 또한 해독할 수 없는 데이터 암호화의 위험을 줄일 수 있습니다. 이러한 기능은 외부에서는 효과적이지 않습니다 AWS KMS. 자세한 내용은 퍼블릭 키 다운로드 시 특별 고려 사항을 참조하세요.

> **Tip**
>
> 데이터 키 또는 SSH 키를 찾고 계십니까? 이 주제에서는 AWS Key Management Service 에서 비대칭 키(프라이빗 키를 내보낼 수 없음)를 관리하는 방법에 대해서 설명합니다. 내보낼 수 있는 데이터 키 페어(프라이빗 키가 대칭 암호화 KMS 키로 보호됨)에 대해서는 GenerateDataKeyPair 섹션을 참조하세요. Amazon EC2 인스턴스와 연결된 퍼블릭 키 다운로드에 대한 도움말은 Amazon EC2 Linux 인스턴스용 사용 설명서의 퍼블릭 키 검색과 Amazon EC2 Windows 인스턴스용 사용 설명서을 참조하세요.

주제

- 퍼블릭 키 다운로드 시 특별 고려 사항
- 퍼블릭 키 다운로드
- 예제 오프라인 작업

퍼블릭 키 다운로드 시 특별 고려 사항

KMS 키를 보호하기 위해는 모든 작업에 대한 액세스 제어, 인증된 암호화 및 세부 로그를 AWS KMS 제공합니다. AWS KMS 또한를 사용하면 KMS 키 사용을 일시적으로 또는 영구적으로 방지할 수 있습

니다. 마지막으로 AWS KMS 작업은 복호화할 수 없는 데이터 암호화 위험을 최소화하도록 설계되었습니다. 외부에서 다운로드한 퍼블릭 키를 사용하는 경우에는 이러한 기능을 사용할 수 없습니다 AWS KMS.

권한 부여

내에서 KMS 키에 대한 액세스를 제어하는 키 정책 및 IAM 정책은 외부에서 수행되는 작업에 영향을 미치지 AWS KMS 않습니다 AWS. 퍼블릭 키를 가져올 수 있는 사용자는 KMS 키를 사용하여 데이터를 암호화하거나 서명을 확인할 권한이 없 AWS KMS 더라도 외부에서 퍼블릭 키를 사용할 수 있습니다.

키 사용 제한 사항

키 사용 제한은 외부에서는 적용되지 않습니다 AWS KMS. RSA 키 페어는 키 사용량이 `ENCRYPT_DECRYPT` 또는 이든 구조적으로 동일합니다`SIGN_VERIFY`. 따라서 `SIGN_VERIFY` RSA KMS 키의 다운로드한 퍼블릭 키를 사용하여 외부의 데이터를 암호화할 수 있습니다 AWS KMS. 그러나 AWS KMS 는 해당 프라이빗 키를 사용하여 해당 데이터를 복호화할 수 없습니다.는 수행하는 작업에 키 사용을 AWS KMS 적용하므로 `SIGN_VERIFY` 키에 대한 복호화 요청을 거부합니다.

알고리즘 제한 사항

가 AWS KMS 지원하는 암호화 및 서명 알고리즘에 대한 제한은 외부에서는 유효하지 않습니다 AWS KMS. 외부의 KMS 키에서 퍼블릭 키로 데이터를 암호화 AWS KMS하고가 지원하지 AWS KMS 않는 암호화 알고리즘을 사용하는 경우 데이터를 해독할 수 없습니다.

KMS 키 비활성화 및 삭제

내 암호화 작업에서 KMS 키를 사용하지 않도록 하기 위해 취할 수 있는 조치는 외부에서 누구나 퍼블릭 키를 사용하지 못하도록 하지 AWS KMS 않습니다 AWS KMS. 예를 들어 KMS 키를 비활성화하거나, KMS 키 삭제를 예약하거나, KMS 키를 삭제하거나, KMS 키에서 키 구성 요소를 삭제해도 AWS KMS외부의 퍼블릭 키에는 영향을 주지 않습니다. 비대칭 KMS 키를 삭제하거나 키 구성 요소를 삭제하거나 분실하면 외부의 퍼블릭 키로 암호화하는 데이터를 복구할 수 AWS KMS 없습니다.

로깅

AWS CloudTrail 요청, 응답, 날짜, 시간 및 권한 있는 사용자를 포함하여 모든 AWS KMS 작업을 기록하는 로그는 외부의 퍼블릭 키 사용을 기록하지 않습니다 AWS KMS.

SM2 키 페어를 사용한 오프라인 인증(중국 리전 전용)

SM2 퍼블릭 키를 AWS KMS 사용하여 외부의 서명을 확인하려면 구분 ID를 지정해야 합니다. 기본적으로는를 구분 ID1234567812345678로 AWS KMS 사용합니다. 자세한 내용은 SM2 키 페어를 사용한 오프라인 인증(중국 리전 전용)을 참조하세요.

퍼블릭 키 다운로드

AWS KMS 콘솔의 비대칭 KMS 키 페어에서 또는 GetPublicKey 작업을 사용하여 퍼블릭 키를 다운로드할 수 있습니다. 퍼블릭 키를 다운로드하려면 비대칭 KMS 키에 대한 `kms:GetPublicKey` 권한이 있어야 합니다.

가 AWS KMS 반환하는 퍼블릭 키는 RFC 5280에 정의된 대로 `SubjectPublicKeyInfo` (SPKI)라고도 하는 DER 인코딩 X.509 퍼블릭 키입니다. https://datatracker.ietf.org/doc/html/rfc5280 HTTP API 또는를 사용하는 경우 AWS CLI값은 Base64-encoded됩니다. 그렇지 않은 경우에는 Base64 인코딩되지 않습니다.

비대칭 KMS 키 페어에서 퍼블릭 키를 다운로드하려면 `kms:GetPublicKey` 권한이 필요합니다. AWS KMS 권한에 대한 자세한 내용은 단원을 참조하십시오권한 참조.

AWS KMS 콘솔 사용

AWS Management Console 를 사용하여의 비대칭 KMS 키에서 퍼블릭 키를 보고, 복사하고, 다운로드할 수 있습니다 AWS 계정. 다른의 비대칭 KMS 키에서 퍼블릭 키를 다운로드하려면 AWS KMS API를 AWS 계정사용합니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 탐색 창에서 고객 관리형 키를 선택합니다.
4. 비대칭 KMS 키의 별칭 또는 키 ID를 선택합니다.
5. 암호화 구성 탭을 선택합니다. 키 사양, 키 사용 및 암호화 알고리즘 또는 서명 알고리즘 필드의 값을 기록합니다. 외부에서 퍼블릭 키를 사용하려면 이러한 값을 사용해야 합니다 AWS KMS. 퍼블릭 키를 공유할 때 이 정보를 공유해야 합니다.
6. 퍼블릭 키 탭을 선택합니다.
7. 퍼블릭 키를 클립보드에 복사하려면 복사를 선택합니다. 퍼블릭 키를 파일로 다운로드하려면 다운로드를 선택합니다.

AWS KMS API 사용

GetPublicKey 작업은 비대칭 KMS 키의 퍼블릭 키를 반환합니다. 또한 키 사용 및 암호화 알고리즘을 AWS KMS포함하여 외부에서 퍼블릭 키를 올바르게 사용하는 데 필요한 중요한 정보를 반환합니다. 퍼블릭 키를 공유 할 때마다 이러한 값을 저장하고 공유하세요.

이 섹션의 예제는 AWS Command Line Interface (AWS CLI)를 사용하지만, 지원되는 모든 프로그래밍 언어를 사용할 수 있습니다.

KMS 키를 식별하려면 키 ID, 키 ARN, 별칭 이름 또는 별칭 ARN을 사용합니다. 별칭 이름을 사용할 때 alias/를 접두사를 사용합니다. 다른에서 KMS 키를 지정하려면 해당 키 ARN 또는 별칭 ARN을 사용해야 AWS 계정합니다.

이 명령을 실행하기 전에 예제 별칭 이름을 KMS 키에 대한 유효한 식별자로 바꾸십시오. 이 명령을 실행하려면 KMS 키에 대한 `kms:GetPublicKey` 권한이 있어야 합니다.

```
$ aws kms get-public-key --key-id alias/example_RSA_3072

{
    "KeySpec": "RSA_3072",
    "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyUsage": "ENCRYPT_DECRYPT",
    "EncryptionAlgorithms": [
        "RSAES_OAEP_SHA_1",
        "RSAES_OAEP_SHA_256"
    ],
    "PublicKey": "MIIBojANBgkqhkiG..."
}
```

예제 오프라인 작업

비대칭 KMS 키 페어의 퍼블릭 키를 다운로드한 후 다른 사람과 공유하고 이를 사용하여 오프라인 작업을 수행할 수 있습니다.

AWS CloudTrail 요청, 응답, 날짜, 시간 및 승인된 사용자를 포함하여 모든 AWS KMS 작업을 기록하는 로그는 외부의 퍼블릭 키 사용을 기록하지 않습니다 AWS KMS.

이 주제에서는 오프라인 작업의 예를 제공하고 오프라인 작업을 더 쉽게 하기 위해 도구가 AWS KMS 제공하는 세부 정보를 제공합니다.

주제

- 공유 보안 비밀을 오프라인으로 도출
- ML-DSA 키 페어를 사용한 오프라인 검증
- SM2 키 페어를 사용한 오프라인 인증(중국 리전 전용)

공유 보안 비밀을 오프라인으로 도출

오프라인 작업, 즉 AWS KMS외부의 작업에 사용하는 ECC 키 페어의 퍼블릭 키를 다운로드할 수 있습니다.

다음 OpenSSL 연습에서는 ECC KMS 키 페어의 퍼블릭 키와 OpenSSL로 생성된 프라이빗 키를 AWS KMS 사용하여 외부에서 공유 보안 암호를 도출하는 한 가지 방법을 보여줍니다.

1. OpenSSL에서 ECC 키 페어를 생성하고 함께 사용할 준비를 합니다 AWS KMS.

```
// Create an ECC key pair in OpenSSL and save the private key in
 openssl_ecc_key_priv.pem
export OPENSSL_CURVE_NAME="P-256"
export KMS_CURVE_NAME="ECC_NIST_P256"

export OPENSSL_KEY1_PRIV_PEM="openssl_ecc_key1_priv.pem"
openssl ecparam -name ${OPENSSL_CURVE_NAME} -genkey -out ${OPENSSL_KEY1_PRIV_PEM}


// Derive the public key from the private key
export OPENSSL_KEY1_PUB_PEM="openssl_ecc_key1_pub.pem"
openssl ec -in ${OPENSSL_KEY1_PRIV_PEM} -pubout -outform pem \
    -out ${OPENSSL_KEY1_PUB_PEM}

// View the PEM file containing the public key and extract the public key as a
// Base64 encoded string into OPENSSL_KEY1_PUB_BASE64 for use with AWS KMS
export OPENSSL_KEY1_PUB_BASE64=`cat ${OPENSSL_KEY1_PUB_PEM} | \
    tee /dev/stderr | grep -v "PUBLIC KEY" | tr -d "\n"`
```

2. 에서 ECC 키 계약 키 페어를 생성하고 OpenSSL과 함께 사용할 AWS KMS 준비를 합니다.

```
// Create a KMS key on the same curve as the key pair from step 1
// with a key usage of KEY_AGREEMENT
// Save its ARN in KMS_KEY1_ARN.
export KMS_KEY1_ARN=`aws kms create-key --key-spec ${KMS_CURVE_NAME} \
    --key-usage KEY_AGREEMENT | tee /dev/stderr | jq -r .KeyMetadata.Arn`

// Download the public key and save the Base64-encoded version in KMS_KEY1_PUB_BASE64

export KMS_KEY1_PUB_BASE64=`aws kms get-public-key --key-id ${KMS_KEY1_ARN} | \
    tee /dev/stderr | jq -r .PublicKey`

// Create a PEM file for the public KMS key for use with OpenSSL
```

```
export KMS_KEY1_PUB_PEM="aws_kms_ecdh_key1_pub.pem"
echo "-----BEGIN PUBLIC KEY-----" > ${KMS_KEY1_PUB_PEM}
echo ${KMS_KEY1_PUB_BASE64} | fold -w 64 >> ${KMS_KEY1_PUB_PEM}
echo "-----END PUBLIC KEY-----" >> ${KMS_KEY1_PUB_PEM}
```

3. OpenSSL의 프라이빗 키와 퍼블릭 KMS 키를 사용하여 OpenSSL의 공유 보안 비밀을 도출합니다.

```
export OPENSSL_SHARED_SECRET1_BIN="openssl_shared_secret1.bin"
openssl pkeyutl -derive -inkey ${OPENSSL_KEY1_PRIV_PEM} \
    -peerkey ${KMS_KEY1_PUB_PEM} -out ${OPENSSL_SHARED_SECRET1_BIN}
```

ML-DSA 키 페어를 사용한 오프라인 검증

AWS KMS 는 최대 [4KB 바이트의 메시지에 대해 FIPS(연방 정보 처리 표준) 204 표준](#) 섹션 3.4에 설명된 대로 ML-DSA 서명의 헤지 변형을 지원합니다.

4KB보다 큰 메시지에 서명하려면 외부에서 메시지 사전 처리 단계를 수행합니다 AWS KMS. 이 해싱 단계는 NIST FIPS 204 섹션 6.2에 정의된 대로 64바이트 메시지 표현값 μ를 생성합니다.

AWS KMS 에는 4KB보다 큰 메시지에 `EXTERNAL_MU` 대해 라는 메시지 유형이 있습니다. `RAW` 메시지 유형 대신 이를 사용하는 경우 AWS KMS:

- 해싱 단계를 이미 수행했다고 가정함
- 내부 해싱 프로세스 건너뛰기
- 메시지 크기와 관계없이 사용 가능

메시지를 검증할 때 사용하는 방법은 외부 시스템 또는 라이브러리의 크기 제한 여부와 64바이트 메시지 대푯값 μ 지원 여부에 따라 달라집니다.

- 메시지가 제한 크기보다 작은 경우 `RAW` 메시지 유형을 사용하세요.
- 메시지가 제한 크기보다 큰 경우 외부 시스템에서 생성한 대푯값 μ를 사용하세요.

다음 섹션에서는를 사용하여 메시지에 서명하고 OpenSSL을 사용하여 메시지를 AWS KMS 확인하는 방법을 보여줍니다. 에서 부과한 4KB 메시지 크기 제한 미만 및 초과 메시지 모두에 대한 예를 제공합니다 AWS KMS. OpenSSL은 검증을 위한 메시지 크기 제한을 적용하지 않습니다.

두 예제 모두 먼저 퍼블릭 키를 가져옵니다 AWS KMS. 다음 AWS CLI 명령을 사용합니다.

```
aws kms get-public-key \
    --key-id _<1234abcd-12ab-34cd-56ef-1234567890ab>_ \
    --output text \
    --query PublicKey | base64 --decode > public_key.der
```

크기가 4KB 미만인 메시지

4KB 미만의 메시지의 경우 RAW 메시지 유형을와 함께 사용합니다 AWS KMS. EXTERNAL_MU를 사용할 수도 있지만 크기 제한 내의 메시지에는 필요하지 않습니다.

다음 AWS CLI 명령을 사용하여 메시지에 서명합니다.

```
aws kms sign \
    --key-id _<1234abcd-12ab-34cd-56ef-1234567890ab>_ \
    --message 'your message' \
    --message-type RAW \
    --signing-algorithm ML_DSA_SHAKE_256 \
    --output text \
    --query Signature | base64 --decode > ExampleSignature.bin
```

이 메시지를 OpenSSL로 검증하려면 다음 명령을 사용하세요.

```
echo -n 'your message' | ./openssl dgst -verify public_key.der -signature
 ExampleSignature.bin
```

크기가 4KB를 초과하는 메시지

4KB보다 큰 메시지에 서명하려면 EXTERNAL_MU 메시지 유형을 사용하세요. EXTERNAL_MU를 사용할 때는 NIST FIPS 204 6.2절에서 정의한 바와 같이 메시지를 외부에서 64바이트 대푯값 μ로 사전 해싱한 후 서명 또는 검증 작업에 전달합니다. 이는 NIST FIPS 204 섹션 5.4에 정의된 “Pre-hash MLDSA” 또는 HashML-DSA와는 다릅니다.

1. 먼저 메시지 접두사를 구성하세요. 접두사는 도메인 구분자, 컨텍스트의 길이 및 컨텍스트를 포함합니다. 도메인 구분자와 컨텍스트 길이의 기본값은 0입니다.
2. 메시지 앞에 메시지 접두사를 추가하세요.
3. SHAKE256을 사용하여 퍼블릭 키를 해시한 뒤, 이를 2단계 결과 앞에 추가하세요.
4. 마지막으로 3단계 결과를 해시해 64바이트 EXTERNAL_MU를 생성합니다.

다음 예제에서는 OpenSSL 3.5를 사용하여 EXTERNAL_MU를 구성합니다.

```
{
    openssl asn1parse -inform DER -in public_key.der -strparse 17 -noout -out - 2>/dev/
null |
    openssl dgst -provider default -shake256 -xoflen 64 -binary;
    printf '\x00\x00';
    echo -n "your message"
} | openssl dgst -provider default -shake256 -xoflen 64 -binary > mu.bin
```

mu.bin 파일을 생성한 후 다음 명령을 사용하여 AWS KMS API를 호출하여 메시지에 서명합니다.

```
aws kms sign \
    --key-id _<1234abcd-12ab-34cd-56ef-1234567890ab>_ \
    --message fileb://mu.bin \
    --message-type EXTERNAL_MU \
    --signing-algorithm ML_DSA_SHAKE_256 \
    --output text \
    --query Signature | base64 --decode > ExampleSignature.bin
```

원본 메시지에 대한 RAW 서명과 동일한 서명이 생성됩니다. 동일한 OpenSSL 3.5 명령을 사용해 메시지를 검증할 수 있습니다.

```
echo -n 'your message' | ./openssl dgst -verify public_key.der -signature
 ExampleSignature.bin
```

SM2 키 페어를 사용한 오프라인 인증(중국 리전 전용)

SM2 퍼블릭 키를 AWS KMS 사용하여 외부의 서명을 확인하려면 구분 ID를 지정해야 합니다. 원시 메시지인를 Sign APIMessageType:RAW에 전달하면 GM/T 0009-2012의 OSCCA에서 정의한 기본 구분 ID1234567812345678인를 AWS KMS 사용합니다. AWS KMS내에서 구분 ID를 지정할 수 없습니다.

그러나 외부에서 메시지 다이제스트를 생성하는 경우 고유한 구분 ID를 지정한 다음 메시지 다이제스트를에 전달MessageType:DIGEST하여 서명 AWS KMS 할 AWS수 있습니다. 이를 위해서는 SM2OfflineOperationHelper 클래스에서 DEFAULT_DISTINGUISHING_ID 값을 변경합니다. 구분 ID는 최대 8,192자 이내의 문자열로 지정할 수 있습니다. 가 메시지 다이제스트에 AWS KMS 서명한 후에는 메시지 다이제스트 또는 메시지와 다이제스트를 계산하여 오프라인으로 확인하는 데 사용되는 구분 ID가 필요합니다.

> **Important**
>
> 이 `SM2OfflineOperationHelper` 참조 코드는 Bouncy Castle 버전 1.68과 호환되도록 설계되었습니다. 다른 버전에 대한 도움말은 bouncycastle.org를 참조하세요.

`SM2OfflineOperationHelper` 클래스

SM2 키를 사용하는 오프라인 작업을 돕기 위해 Java용 `SM2OfflineOperationHelper` 클래스에는 이 태스크를 수행하는 메서드가 있습니다. 이 도우미 클래스를 다른 암호화 공급자의 모델로 사용할 수 있습니다.

내에서 AWS KMS원시 사이퍼텍스트 변환 및 SM2DSA 메시지 다이제스트 계산은 자동으로 수행됩니다. 모든 암호화 공급자가 동일한 방식으로 SM2를 구현하는 것은 아닙니다. OpenSSL 버전 1.1.1 이상과 같은 일부 라이브러리는 이러한 작업을 자동으로 수행합니다.는 OpenSSL 버전 3.0을 사용하여 테스트할 때이 동작을 AWS KMS 확인했습니다. Bouncy Castle과 같은 라이브러리와 함께 다음 `SM2OfflineOperationHelper` 클래스를 사용하여 이러한 변환과 계산을 수작업으로 수행해야 합니다.

`SM2OfflineOperationHelper` 클래스는 다음 오프라인 작업에 대한 메서드를 제공합니다.

- 메시지 다이제스트 계산

  오프라인 확인에 사용할 수 있거나에 전달하여 서명 AWS KMS 할 수 있는 메시지 다이제스트를 오프라인으로 생성하려면 `calculateSM2Digest` 메서드를 사용합니다. `calculateSM2Digest` 메서드는 SM3 해싱 알고리즘으로 메시지 다이제스트를 생성합니다. GetPublicKey API는 바이너리 형식으로 퍼블릭 키를 반환합니다. 바이너리 키는 Java PublicKey로 파싱해야 합니다. 파싱된 퍼블릭 키에 메시지를 제공합니다. 이 방법은 메시지와 기본 구분 ID, `1234567812345678`를 자동으로 결합하지만 `DEFAULT_DISTINGUISHING_ID` 값을 변경하면 자체적인 구분 ID를 설정할 수 있습니다.

- Verify

  오프라인에서 서명을 인증하려면 `offlineSM2DSAVerify` 메서드를 사용하세요. `offlineSM2DSAVerify` 메서드는 지정된 구분 ID에서 계산된 메시지 다이제스트와 디지털 서명 인증에 제공하는 원본 메시지를 사용합니다. GetPublicKey API는 바이너리 형식으로 퍼블릭 키를 반환합니다. 바이너리 키는 Java PublicKey로 파싱해야 합니다. 파싱된 공개 키에 원본 메시지와 인증하려는 서명을 제공합니다. 자세한 내용은 SM2 키 페어로 오프라인 인증을 참조하세요.

- 암호화

  일반 텍스트를 오프라인에서 암호화하려면 `offlineSM2PKEEncrypt` 메서드를 사용합니다. 이 방법은 사이퍼텍스트가 복호화할 AWS KMS 수 있는 형식인지 확인합니다. `offlineSM2PKEEncrypt` 메서드는 일반 텍스트를 암호화한 다음, SM2PKE에서 생성한 원본 사이퍼텍스트를 ASN.1 형식으로 변환합니다. GetPublicKey API는 바이너리 형식으로 퍼블릭 키를 반환합니다. 바이너리 키는 Java PublicKey로 파싱해야 합니다. 파싱된 공개 키에 암호화하려는 일반 텍스트를 제공합니다.

  변환을 수행해야 하는지 확신하기 어려운 경우, 다음 OpenSSL 작업을 사용하여 사이퍼텍스트의 형식을 테스트합니다. 작업이 실패할 경우, 사이퍼텍스트를 ASN.1 형식으로 변환합니다.

  ```
  openssl asn1parse -inform DER -in ciphertext.der
  ```

기본적으로 `SM2OfflineOperationHelper` 클래스는 SM2DSA 작업에 대한 메시지 다이제스트를 생성할 때 기본 구분 ID, `1234567812345678`를 사용합니다.

```
package com.amazon.kms.utils;

import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import java.io.IOException;
import java.math.BigInteger;
import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.security.InvalidKeyException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.security.NoSuchProviderException;
import java.security.PrivateKey;
import java.security.PublicKey;

import org.bouncycastle.crypto.CryptoException;
import org.bouncycastle.jce.interfaces.ECPublicKey;

import java.util.Arrays;

import org.bouncycastle.asn1.ASN1EncodableVector;
import org.bouncycastle.asn1.ASN1Integer;
```

```
import org.bouncycastle.asn1.DEROctetString;
import org.bouncycastle.asn1.DERSequence;
import org.bouncycastle.asn1.gm.GMNamedCurves;
import org.bouncycastle.asn1.x9.X9ECParameters;
import org.bouncycastle.crypto.CipherParameters;
import org.bouncycastle.crypto.params.ParametersWithID;
import org.bouncycastle.crypto.params.ParametersWithRandom;
import org.bouncycastle.crypto.signers.SM2Signer;
import org.bouncycastle.jcajce.provider.asymmetric.util.ECUtil;

public class SM2OfflineOperationHelper {
    // You can change the DEFAULT_DISTINGUISHING_ID value to set your own
 distinguishing ID,
    // the DEFAULT_DISTINGUISHING_ID can be any string up to 8,192 characters long.
    private static final byte[] DEFAULT_DISTINGUISHING_ID =
 "1234567812345678".getBytes(StandardCharsets.UTF_8);
    private static final X9ECParameters SM2_X9EC_PARAMETERS =
 GMNamedCurves.getByName("sm2p256v1");

    // ***calculateSM2Digest***
    // Calculate message digest
    public static byte[] calculateSM2Digest(final PublicKey publicKey, final byte[]
 message) throws
            NoSuchProviderException, NoSuchAlgorithmException {
        final ECPublicKey ecPublicKey = (ECPublicKey) publicKey;

        // Generate SM3 hash of default distinguishing ID, 1234567812345678
        final int entlenA = DEFAULT_DISTINGUISHING_ID.length * 8;
        final byte [] entla = new byte[] { (byte) (entlenA & 0xFF00), (byte) (entlenA &
 0x00FF) };
        final byte [] a = SM2_X9EC_PARAMETERS.getCurve().getA().getEncoded();
        final byte [] b = SM2_X9EC_PARAMETERS.getCurve().getB().getEncoded();
        final byte [] xg = SM2_X9EC_PARAMETERS.getG().getXCoord().getEncoded();
        final byte [] yg = SM2_X9EC_PARAMETERS.getG().getYCoord().getEncoded();
        final byte[] xa = ecPublicKey.getQ().getXCoord().getEncoded();
        final byte[] ya = ecPublicKey.getQ().getYCoord().getEncoded();
        final byte[] za = MessageDigest.getInstance("SM3", "BC")
                .digest(ByteBuffer.allocate(entla.length +
 DEFAULT_DISTINGUISHING_ID.length + a.length + b.length + xg.length + yg.length +
                        xa.length +
 ya.length).put(entla).put(DEFAULT_DISTINGUISHING_ID).put(a).put(b).put(xg).put(yg).put(xa).put
                        .array());
```

```
        // Combine hashed distinguishing ID with original message to generate final
digest
        return MessageDigest.getInstance("SM3", "BC")
                .digest(ByteBuffer.allocate(za.length +
message.length).put(za).put(message)
                        .array());
    }

    // ***offlineSM2DSAVerify***
    // Verify digital signature with SM2 public key
    public static boolean offlineSM2DSAVerify(final PublicKey publicKey, final byte []
message,
            final byte [] signature) throws InvalidKeyException {
        final SM2Signer signer = new SM2Signer();
        CipherParameters cipherParameters =
ECUtil.generatePublicKeyParameter(publicKey);
        cipherParameters = new ParametersWithID(cipherParameters,
DEFAULT_DISTINGUISHING_ID);
        signer.init(false, cipherParameters);
        signer.update(message, 0, message.length);
        return signer.verifySignature(signature);
    }

    // ***offlineSM2PKEEncrypt***
    // Encrypt data with SM2 public key
    public static byte[] offlineSM2PKEEncrypt(final PublicKey publicKey, final byte []
plaintext) throws
            NoSuchPaddingException, NoSuchAlgorithmException, NoSuchProviderException,
InvalidKeyException,
            BadPaddingException, IllegalBlockSizeException, IOException {
        final Cipher sm2Cipher = Cipher.getInstance("SM2", "BC");
        sm2Cipher.init(Cipher.ENCRYPT_MODE, publicKey);

        // By default, Bouncy Castle returns raw ciphertext in the c1c2c3 format
        final byte [] cipherText = sm2Cipher.doFinal(plaintext);

        // Convert the raw ciphertext to the ASN.1 format before passing it to AWS KMS
        final ASN1EncodableVector asn1EncodableVector = new ASN1EncodableVector();
        final int coordinateLength = (SM2_X9EC_PARAMETERS.getCurve().getFieldSize() +
7) / 8 * 2 + 1;
        final int sm3HashLength = 32;
        final int xCoordinateInCipherText = 33;
        final int yCoordinateInCipherText = 65;
        byte[] coords = new byte[coordinateLength];
```

```
        byte[] sm3Hash = new byte[sm3HashLength];
        byte[] remainingCipherText = new byte[cipherText.length - coordinateLength -
 sm3HashLength];

        // Split components out of the ciphertext
        System.arraycopy(cipherText, 0, coords, 0, coordinateLength);
        System.arraycopy(cipherText, cipherText.length - sm3HashLength, sm3Hash, 0,
 sm3HashLength);
        System.arraycopy(cipherText, coordinateLength, remainingCipherText,
 0,cipherText.length - coordinateLength - sm3HashLength);

        // Build standard SM2PKE ASN.1 ciphertext vector
        asn1EncodableVector.add(new ASN1Integer(new BigInteger(1,
 Arrays.copyOfRange(coords, 1, xCoordinateInCipherText))));
        asn1EncodableVector.add(new ASN1Integer(new BigInteger(1,
 Arrays.copyOfRange(coords, xCoordinateInCipherText, yCoordinateInCipherText))));
        asn1EncodableVector.add(new DEROctetString(sm3Hash));
        asn1EncodableVector.add(new DEROctetString(remainingCipherText));

        return new DERSequence(asn1EncodableVector).getEncoded("DER");
    }
}
```

모니터링 AWS KMS keys

모니터링은에서의 가용성, 상태 및 사용량 AWS KMS keys 을 이해하고 AWS 솔루션의 신뢰성, 가용성 및 성능을 AWS KMS 유지하는 데 중요한 부분입니다. AWS 솔루션의 모든 부분으로부터 모니터링 데이터를 수집하면 다중 지점 실패가 발생할 경우 디버깅하는 데 도움을 줍니다. 하지만 KMS 키 모니터링을 시작하기 전에 다음 질문에 대한 답변을 포함하는 모니터링 계획을 작성하십시오.

- 모니터링의 목표
- 모니터링할 리소스
- 이러한 리소스를 모니터링하는 빈도
- 사용할 모니터링 도구
- 모니터링 작업을 수행할 사람
- 문제 발생 시 알려야 할 대상

다음 단계는 시간 경과에 따라 KMS 키를 모니터링하여 현재 환경에서 통상적인 AWS KMS 사용량과 기대치의 기준선을 설정하는 것입니다. KMS 키를 모니터링하면서 모니터링 데이터 기록을 저장합니다. 그러면 현재 데이터를 이 과거 데이터와 비교하여 일반적인 패턴과 이상을 식별하고 이를 해결할 방법을 모색할 수 있습니다.

예를 들어 KMS 키에 영향을 미치는 AWS KMS API 활동 및 이벤트를 모니터링할 수 있습니다. 데이터가 정해진 기준을 초과하거나 미달하는 경우, 조사를 실시하거나 시정 조치를 취해야 할 수 있습니다.

정상 패턴의 기준선을 설정하려면 다음 항목을 모니터링합니다.

- AWS KMS 데이터 영역 작업을 위한 API 활동입니다. 이는 Decrypt, Encrypt, ReEncrypt, GenerateDataKey 등 KMS 키를 사용하는 암호화 작업입니다.
- AWS KMS 중요한 컨트롤 플레인 작업에 대한 API 활동입니다. 이러한 작업은 KMS 키를 관리합니다. 사용자는 KMS 키의 가용성을 변경하는 항목들(예: ScheduleKeyDeletion, CancelKeyDeletion, DisableKey, EnableKey, ImportKeyMaterial, DeleteImportedKeyMaterial)이나 KMS 키의 액세스 컨트롤을 변경하는 항목들(예: PutKeyPolicy 및 RevokeGrant)을 모니터링하고 싶을 수 있습니다.
- 기타 AWS KMS 지표(예: 가져온 키 구성 요소가 만료될 때까지 남은 시간) 및 이벤트(예: 가져온 키 구성 요소의 만료 또는 KMS 키의 삭제 또는 키 교체).

모니터링 도구

AWS 는 KMS 키를 모니터링하는 데 사용할 수 있는 다양한 도구를 제공합니다. 이들 도구 중에는 모니터링을 자동으로 수행하도록 구성할 수 있는 도구도 있지만, 수동 작업이 필요한 도구도 있습니다. 모니터링 작업은 최대한 자동화하는 것이 좋습니다.

자동 모니터링 도구

다음과 같은 자동 모니터링 도구를 사용하여 KMS 키를 관찰하고 변경 사항 발생 시 보고할 수 있습니다.

- AWS CloudTrail 로그 모니터링 - 계정 간에 로그 파일을 공유하고, CloudTrail CloudWatch 로그 파일을 실시간으로 모니터링하고, CloudTrail Processing Library를 사용하여 로그 처리 애플리케이션을 작성하고, CloudTrail에서 전송한 후 로그 파일이 변경되지 않았는지 확인합니다. 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail 로그 파일 작업을 참조하세요.
- Amazon CloudWatch 경보 – 지정한 기간 동안 단일 지표를 감시하고, 여러 기간에 대해 지정된 임곗값과 관련하여 지표 값을 기준으로 하나 이상의 작업을 수행합니다. 이 작업은 Amazon Simple Notification Service(Amazon SNS) 주제 또는 Amazon EC2 Auto Scaling 정책에 전송되는 알림입니다. CloudWatch 경보는 특정 상태에 있다는 이유만으로는 작업을 호출하지 않습니다. 상태가 변경되고 지정한 기간 동안 유지되어야 합니다. 자세한 내용은 Amazon CloudWatch를 사용하여 KMS 키 모니터링 단원을 참조하십시오.
- Amazon EventBridge – 이벤트를 일치시키고 하나 이상의 대상 함수 또는 스트림으로 라우팅하여 상태 정보를 캡처하거나 필요한 경우 변경 또는 수정 작업을 수행합니다. 자세한 내용은 Amazon EventBridge를 사용하여 KMS 키 모니터링 및 Amazon EventBridge 사용 설명서를 참조하세요.
- Amazon CloudWatch Logs - AWS CloudTrail 또는 기타 소스에서 로그 파일을 모니터링, 저장 및 액세스합니다. 자세한 내용은 Amazon CloudWatch Logs 사용자 안내서를 참조하세요.

수동 모니터링 도구

KMS 키 모니터링의 또 한 가지 중요한 부분은 CloudWatch 경보 및 이벤트에 포함되지 않는 항목을 수동으로 모니터링해야 한다는 점입니다. AWS Trusted Advisor, AWS KMS CloudWatch 및 기타 AWS 대시보드는 환경 상태를 at-a-glance 볼 수 있습니다 AWS .

AWS KMS 콘솔의 AWS 관리형 키 및 고객 관리형 키 페이지를 사용자 지정하여 각 KMS 키에 대한 다음 정보를 표시할 수 있습니다.

- 키 ID

- Status
- Creation(생성) 날짜
- 만료 날짜(가져온 키 구성 요소가 있는 KMS 키의 경우)
- 출처(Origin)
- 사용자 지정 키 스토어 ID(사용자 지정 키 스토어에 있는 KMS 키의 경우)

CloudWatch 콘솔 대시보드는 다음 정보를 보여줍니다.

- 현재 경보 및 상태
- 경보 및 리소스 그래프
- 서비스 상태

또한 CloudWatch를 사용하여 다음을 수행할 수 있습니다.

- 맞춤 대시보드를 생성하여 관심 있는 서비스 모니터링
- 지표 데이터를 그래프로 작성하여 문제를 해결하고 추세 파악
- 모든 AWS 리소스 지표 검색 및 찾아보기
- 문제에 대해 알려주는 경보 생성 및 편집

AWS Trusted Advisor 를 사용하면 AWS 리소스를 모니터링하여 성능, 신뢰성, 보안 및 비용 효율성을 개선할 수 있습니다. 모든 사용자는 네 가지 Trusted Advisor 검사를 사용할 수 있습니다. Business 또는 Enterprise 지원 플랜을 보유한 사용자는 50개 이상의 검사를 사용할 수 있습니다. 자세한 내용은 AWS Trusted Advisor 단원을 참조하십시오.

를 사용하여 AWS KMS API 호출 로깅 AWS CloudTrail

AWS KMS 는 사용자AWS CloudTrail, 역할 및 기타 서비스 AWS KMS 별로에 대한 모든 호출을 기록하는 AWS 서비스인와 통합됩니다. CloudTrail은 AWS KMS 콘솔, API, CloudFormation 템플릿, AWS Command Line Interface (AWS CLI) 및의 호출을 포함하여에 대한 모든 AWS KMS APIs 호출을 이벤트 AWS KMS 로 캡처합니다 AWS Tools for PowerShell.

CloudTrail은 ListAliases 및 GetKeyRotationStatus와 같은 읽기 전용 작업, CreateKey 및 PutKeyPolicy와 같은 KMS 키를 관리하는 작업, GenerateDataKey 및 Decrypt와 같은 암호화 작업을 AWS KMS 포함한 모든 작업을 로깅합니다. ??? 또한 DeleteExpiredKeyMaterial, DeleteKey,

SynchronizeMultiRegionKey DeleteKey 및 RotateKey와 같이가 자동으로 AWS KMS 호출하는 내부 작업을 로깅합니다.

CloudTrail은 모든 성공적인 작업을 로그로 기록하며, 호출자가 리소스 액세스가 거부되는 경우와 같이 실패한 직접 호출 시도도 일부 시나리오에서 기록합니다. KMS 키에 대한 크로스 계정 작업은 호출자 계정과 KMS 키 소유자 계정 모두에 기록됩니다. 그러나 액세스가 거부되어 거부된 교차 계정 AWS KMS 요청은 호출자의 계정에만 로깅됩니다.

보안상의 이유로 암호화 요청의 `Plaintext` 파라미터, GetKeyPolicy에 대한 응답 또는 암호화 작업과 같은 일부 필드는 AWS KMS 로그 항목에서 생략됩니다. 특정 KMS 키에 대한 CloudTrail 로그 항목을 더 쉽게 검색할 수 있도록는 API 작업이 키 ARN을 반환하지 않더라도 영향을 받는 KMS 키의 키 ARN을 일부 AWS KMS 키 관리 작업에 대한 로그 항목의 `responseElements` 필드에 AWS KMS 추가합니다.

기본적으로 모든 AWS KMS 작업은 CloudTrail 이벤트로 로깅되지만 CloudTrail 추적에서 작업을 제외 AWS KMS 할 수 있습니다. 자세한 내용은 추적에서 AWS KMS 이벤트 제외을 참조하세요.

자세히 알아보기:

- 증명된 플랫폼에 대한 AWS KMS 작업의 CloudTrail 로그 예제는 섹션을 참조하세요증명된 요청 모니터링.

주제

- CloudTrail에서 AWS KMS 로그 항목 찾기
- 추적에서 AWS KMS 이벤트 제외
- AWS KMS 로그 항목의 예

CloudTrail에서 AWS KMS 로그 항목 찾기

CloudTrail 로그 항목을 검색하려면 CloudTrail 콘솔 또는 CloudTrail LookupEvents 작업을 사용합니다. CloudTrail은 이벤트 이름, 사용자 이름, 이벤트 소스 등 검색 필터링을 위한 다양한 속성 값을 지원합니다.

CloudTrail에서 AWS KMS 로그 항목을 검색하는 데 도움이 되도록는 다음 CloudTrail 로그 항목 필드를 AWS KMS 채웁니다.

> **Note**
>
> 2022년 12월부터는 특정 KMS 키를 변경하는 모든 관리 작업에서 리소스 유형 및 리소스 이름 속성을 AWS KMS 채웁니다. 이러한 속성 값은 CreateAlias, CreateGrant, DeleteAlias, DeleteImportedKeyMaterial, ImportKeyMaterial, ReplicateKey, RetireGrant, RevokeGrant, UpdateAlias 및 UpdatePrimaryRegion 작업에 대한 이전 CloudTrail 항목에서 null일 수 있습니다.

속성	값	로그 항목
이벤트 소스(`EventSource`)	`kms.amazonaws.com`	모든 작업
리소스 유형(`ResourceType`)	`AWS::KMS::Key`	특정 KMS 키(예: `CreateKey` 및 `EnableKey`)는 변경하지만 `ListKeys`는 변경하지 않는 관리 작업
리소스 이름(`ResourceName`)	키 ARN(또는 키 ID 및 키 ARN)	특정 KMS 키(예: `CreateKey` 및 `EnableKey`)는 변경하지만 `ListKeys`는 변경하지 않는 관리 작업

특정 KMS 키에 대한 관리 작업의 로그 항목을 찾는 데 도움이 되도록는 AWS KMS AWS KMS API 작업이 키 ARN을 반환하지 않더라도 영향을 받는 KMS 키의 키 ARN을 로그 항목의 `responseElements.keyId` 요소에 기록합니다.

예를 들어 DisableKey 작업을 성공적으로 호출해도 응답에 값이 반환되지 않지만 DisableKey 로그 항목의 `responseElements.keyId` 값에는 null 값 대신 비활성화된 KMS 키의 키 ARN이 포함됩니다.

이 기능은 2022년 12월에 추가되었으며 CloudTrail 로그 항목 CreateAlias, CreateGrant, DeleteAlias, DeleteKey, DisableKey, EnableKey, EnableKeyRotation, ImportKeyMaterial, RotateKey, SynchronizeMultiRegionKey, TagResource, UntagResource, UpdateAlias 및 UpdatePrimaryRegion에 영향을 줍니다.

추적에서 AWS KMS 이벤트 제외

AWS KMS 리소스의 사용 및 관리에 대한 기록을 제공하기 위해 대부분의 AWS KMS 사용자는 CloudTrail 추적의 이벤트에 의존합니다. 추적은 생성, 비활성화 및 삭제, 키 정책 AWS KMS keys변경, AWS 서비스에서 사용자를 대신하여 KMS 키 사용 등 중요한 이벤트를 감사하는 데 유용한 데이터 소스가 될 수 있습니다. 경우에 따라 CloudTrail 로그 항목의 메타데이터(예: 암호화 작업의 암호화 컨텍스트)는 오류를 방지하거나 해결하는 데 도움이 될 수 있습니다.

그러나는 많은 수의 이벤트를 생성할 AWS KMS 수 있으므로 추적에서 AWS KMS 이벤트를 제외할 수 AWS CloudTrail 있습니다. 이 추적별 설정은 모든 AWS KMS 이벤트를 제외하며 특정 AWS KMS 이벤트를 제외할 수는 없습니다.

> **Warning**
>
> CloudTrail 로그에서 AWS KMS 이벤트를 제외하면 KMS 키를 사용하는 작업이 가려질 수 있습니다. 보안 주체에게 이 작업을 수행하는 데 필요한 `cloudtrail:PutEventSelectors` 권한을 부여할 때는 주의해야 합니다.

추적에서 AWS KMS 이벤트를 제외하려면:

- CloudTrail 콘솔에서 추적 생성 또는 추적 업데이트를 수행할 때 키 관리 서비스 이벤트 로그 설정을 사용합니다. 지침은 AWS CloudTrail 사용 설명서의를 사용하여 관리 이벤트 로깅 AWS Management Console을 참조하세요.
- CloudTrail API에서 PutEventSelectors 작업을 사용합니다. `ExcludeManagementEventSources`의 값을 사용하여 이벤트 선택기에 `kms.amazonaws.com` 속성을 추가합니다. 예제는 AWS CloudTrail 사용 설명서의 예제: AWS Key Management Service 이벤트를 로깅하지 않는 추적을 참조하세요.

콘솔 설정 또는 추적용 이벤트 선택기를 변경하여 언제든지 이 제외를 비활성화할 수 있습니다. 그러면 추적이 AWS KMS 이벤트 기록을 시작합니다. 그러나 제외가 유효한 동안 발생한 AWS KMS 이벤트는 복구할 수 없습니다.

콘솔 또는 API를 사용하여 AWS KMS 이벤트를 제외하면 결과 CloudTrail `PutEventSelectors` API 작업도 CloudTrail 로그에 로깅됩니다. AWS KMS 이벤트가 CloudTrail Logs에 나타나지 않으면 `ExcludeManagementEventSources`속성이 `kms.amazonaws.com`으로 설정된 `PutEventSelectors` 이벤트를 찾습니다.

AWS KMS 로그 항목의 예

AWS KMS 는 사용자가 AWS KMS 작업을 호출할 때와 서비스가 사용자를 대신하여 작업을 호출할 때 AWS CloudTrail 로그에 항목을 씁니다. AWS KMS 또한는 사용자를 대신하여 작업을 호출할 때 항목을 씁니다. 예를 들어, 삭제하도록 예약한 KMS 키를 삭제할 때 항목을 씁니다.

다음 주제에서는 AWS KMS 작업에 대한 CloudTrail 로그 항목의 예를 보여줍니다.

증명된 플랫폼에서 AWS KMS 에 대한 요청의 CloudTrail 로그 항목의 예는 섹션을 참조하세요증명된 요청 모니터링.

주제

- CancelKeyDeletion
- ConnectCustomKeyStore
- CreateAlias
- CreateCustomKeyStore
- CreateGrant
- CreateKey
- Decrypt
- DeleteAlias
- DeleteCustomKeyStore
- DeleteExpiredKeyMaterial
- DeleteImportedKeyMaterial
- DeleteKey
- DescribeCustomKeyStores
- DescribeKey
- DisableKey
- DisableKeyRotation
- DisconnectCustomKeyStore
- EnableKey
- EnableKeyRotation
- 암호화

- GenerateDataKey
- GenerateDataKeyPair
- GenerateDataKeyPairWithoutPlaintext
- GenerateDataKeyWithoutPlaintext
- GenerateMac
- GenerateRandom
- GetKeyLastUsage
- GetKeyPolicy
- GetKeyRotationStatus
- GetParametersForImport
- ImportKeyMaterial
- ListAliases
- ListGrants
- ListKeyRotations
- PutKeyPolicy
- ReEncrypt
- ReplicateKey
- RetireGrant
- RevokeGrant
- RotateKey
- RotateKeyOnDemand
- ScheduleKeyDeletion
- Sign
- SynchronizeMultiRegionKey
- TagResource
- UntagResource
- UpdateAlias
- UpdateCustomKeyStore
- UpdateKeyDescription

- UpdatePrimaryRegion
- VerifyMac
- 확인
- Amazon EC2 예제 1
- Amazon EC2 예제 2

CancelKeyDeletion

다음 예제는 CancelKeyDeletion 작업을 호출하여 생성된 AWS CloudTrail 로그 항목을 보여줍니다. 삭제에 대한 자세한 내용은 섹션을 AWS KMS keys참조하세요삭제 AWS KMS key.


```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2020-07-27T21:53:17Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CancelKeyDeletion",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": {
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "requestID": "e3452e68-d4b0-4ec7-a768-7ae96c23764f",
    "eventID": "d818bf03-6655-48e9-8b26-f279a07075fd",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
```

```
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

ConnectCustomKeyStore

다음 예제는 ConnectCustomKeyStore 작업을 호출하여 생성된 AWS CloudTrail 로그 항목을 보여 줍니다. 사용자 지정 키 스토어 연결에 대한 자세한 내용은 AWS CloudHSM 키 스토어 연결 해제 섹션을 참조하세요.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2021-10-21T20:17:32Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "ConnectCustomKeyStore",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "customKeyStoreId": "cks-1234567890abcdef0"
    },
    "responseElements": null,
    "additionalEventData": {
        "customKeyStoreName": "ExampleKeyStore",
        "clusterId": "cluster-1a23b4cdefg"
    },
    "requestID": "abcde9e1-f1a3-4460-a423-577fb6e695c9",
    "eventID": "114b61b9-0ea6-47f5-a9d2-4f2bdd0017d5",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
```

```
    "recipientAccountId": "111122223333"
}
```

CreateAlias

다음 예제에서는 CreateAlias 작업에 대한 AWS CloudTrail 로그 항목을 보여줍니다. `resources` 요소에는 별칭 및 KMS 키 리소스에 대한 필드가 포함됩니다. 에서 별칭을 생성하는 방법에 대한 자세한 내용은 섹션을 AWS KMS참조하세요별칭 생성.

2022년 12월 이후에 기록된 이 작업에 대한 CloudTrail 로그 항목에는 이 작업이 키 ARN을 반환하지 않더라도 영향을 받는 KMS 키의 키 ARN이 `responseElements.keyId` 값에 포함됩니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2022-08-14T23:08:31Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateAlias",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "aliasName": "alias/ExampleAlias",
        "targetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": {
        "keyId":"arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "requestID": "caec1e0c-ce03-419e-bdab-6ab1f7c57c01",
    "eventID": "2dd6e784-8286-46a6-befd-d64e5a02fb28",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
```

```
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        },
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:111122223333:alias/ExampleAlias"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

CreateCustomKeyStore

다음 예제는 AWS CloudHSM 키 스토어에서 `CreateCustomKeyStore` 작업을 호출하여 생성된 AWS CloudTrail 로그 항목을 보여줍니다. 사용자 지정 키 스토어 생성에 대한 자세한 내용은 AWS CloudHSM 키 스토어 생성 섹션을 참조하세요.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2021-10-21T20:17:32Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateCustomKeyStore",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "customKeyStoreName": "ExampleKeyStore",
        "clusterId": "cluster-1a23b4cdefg"
    },
    "responseElements": {
        "customKeyStoreId": "cks-1234567890abcdef0"
```

```
    },
    "requestID": "abcde9e1-f1a3-4460-a423-577fb6e695c9",
    "eventID": "114b61b9-0ea6-47f5-a9d2-4f2bdd0017d5",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333"
}
```

CreateGrant

다음 예제에서는 CreateGrant 작업에 대한 AWS CloudTrail 로그 항목을 보여줍니다. 에서 권한 부여를 생성하는 방법에 대한 자세한 내용은 섹션을 AWS KMS참조하세요의 권한 부여 AWS KMS.

2022년 12월 이후에 기록된 이 작업에 대한 CloudTrail 로그 항목에는 이 작업이 키 ARN을 반환하지 않더라도 영향을 받는 KMS 키의 키 ARN이 `responseElements.keyId` 값에 포함됩니다.

다음 예제에서는 암호화 컨텍스트 제약 조건이 있는 권한 부여에 대한 `CreateGrant` 로그 항목을 보여줍니다.

```
{
  "eventVersion": "1.02",
  "userIdentity": {
      "type": "IAMUser",
      "principalId": "EX_PRINCIPAL_ID",
      "arn": "arn:aws:iam::111122223333:user/Alice",
      "accountId": "111122223333",
      "accessKeyId": "EXAMPLE_KEY_ID",
      "userName": "Alice"
  },
  "eventTime": "2014-11-04T00:53:12Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "CreateGrant",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": {
      "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
      "constraints": {
          "encryptionContextSubset": {
              "ContextKey1": "Value1"
          }
```

```
      },
      "operations": [
        "Encrypt",
        "RetireGrant"
      ],
      "granteePrincipal": "service-name.amazonaws.com"
  },
  "responseElements": {
      "grantId": "f020fe75197b93991dc8491d6f19dd3cebb24ee62277a05914386724f3d48758",
      "keyId":"arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  },
  "requestID": "f3c08808-63bc-11e4-bc2b-4198b6150d5c",
  "eventID": "5d529779-2d27-42b5-92da-91aaea1fc4b5",
  "readOnly": false,
  "resources": [{
      "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
      "accountId": "111122223333"
  }],
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}
```

다음 예제에서는 서비스 보안 주체 권한 부여에 대한 CreateGrant 로그 항목을 보여줍니다. 이 권한 부여는 GranteeServicePrincipal 파라미터를 사용하여 AWS 서비스 보안 주체를 피부여자로 지정하고 SourceArn 권한 부여 제약 조건을 포함합니다.

```
{
  "eventVersion": "1.08",
  "userIdentity": {
      "type": "IAMUser",
      "principalId": "EX_PRINCIPAL_ID",
      "arn": "arn:aws:iam::111122223333:user/Alice",
      "accountId": "111122223333",
      "accessKeyId": "EXAMPLE_KEY_ID",
      "userName": "Alice"
  },
  "eventTime": "2026-03-04T18:22:45Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "CreateGrant",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
```

```
  "userAgent": "AWS Internal",
  "requestParameters": {
      "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
      "constraints": {
          "sourceArn": "arn:aws:dynamodb:us-east-1:111122223333:table/ExampleTable"
      },
      "operations": [
        "Encrypt",
        "Decrypt",
        "GenerateDataKey"
      ],
      "granteeServicePrincipal": "service-name.amazonaws.com",
      "retiringServicePrincipal": "service-name.amazonaws.com"
  },
  "responseElements": {
      "grantId": "a1b2c3d4e5f6a7b8c9d0e1f2a3b4c5d6e7f8a9b0c1d2e3f4a5b6c7d8e9f0a1b2",
      "keyId":"arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  },
  "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
  "readOnly": false,
  "resources": [{
      "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
      "accountId": "111122223333"
  }],
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}
```

Note

GranteeServicePrincipal 파라미터로 권한 부여가 생성되면 CreateGrant 작업에 대한 CloudTrail 로그 항목에는 대신 granteeServicePrincipal 필드가 포함됩니다granteePrincipal. 마찬가지로이 RetiringServicePrincipal 지정된 경우 로그 항목에는 대신 retiringServicePrincipal 필드가 포함됩니다retiringPrincipal. 이렇게 하면 AWS 서비스 보안 주체에 GranteeServicePrincipal 대해 로 명시적으로 생성된 권한 부여가 granteePrincipal 필드에 AWS 서비스가 표시되는 권한 부여와 구별됩니다.

CreateKey

이 예제에서는 CreateKey 작업에 대한 AWS CloudTrail 로그 항목을 보여줍니다.

CreateKey 로그 항목은 CreateKey 요청이나 ReplicateKey 요청에 대한 CreateKey 작업의 결과일 수 있습니다.

다음 예제는 대칭 암호화 KMS 키를 생성하는 CreateKey 작업에 대한 CloudTrail 로그 항목을 보여줍니다. KMS 키 생성에 대한 자세한 내용은 KMS 키 생성 섹션을 참조하세요.


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2022-08-10T22:38:27Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "description": "",
        "origin": "EXTERNAL",
        "bypassPolicyLockoutSafetyCheck": false,
        "customerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "keySpec": "SYMMETRIC_DEFAULT",
        "keyUsage": "ENCRYPT_DECRYPT"
    },
    "responseElements": {
        "keyMetadata": {
            "AWSAccountId": "111122223333",
            "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
            "arn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "creationDate": "Aug 10, 2022, 10:38:27 PM",
            "enabled": false,
            "description": "",
```

```
            "keyUsage": "ENCRYPT_DECRYPT",
            "keyState": "PendingImport",
            "origin": "EXTERNAL",
            "keyManager": "CUSTOMER",
            "customerMasterKeySpec": "SYMMETRIC_DEFAULT",
            "keySpec": "SYMMETRIC_DEFAULT",
            "encryptionAlgorithms": [
                "SYMMETRIC_DEFAULT"
            ],
            "multiRegion": false
        }
    },
    "requestID": "1aef6713-0223-4ff7-9a6d-781360521930",
    "eventID": "36327b37-f4f6-40a9-92ab-48064ec905a2",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

다음 예제는 AWS CloudHSM 키 스토어에서 대칭 암호화 KMS 키를 생성하는 `CreateKey` 작업의 CloudTrail 로그를 보여줍니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::111122223333:user/Alice",
            "accountId": "111122223333",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
    },
    "eventTime": "2021-10-14T17:39:50Z",
```

```
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyUsage": "ENCRYPT_DECRYPT",
        "bypassPolicyLockoutSafetyCheck": false,
        "origin": "AWS_CLOUDHSM",
        "keySpec": "SYMMETRIC_DEFAULT",
        "customerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "customKeyStoreId": "cks-1234567890abcdef0",
        "description": ""
    },
    "responseElements": {
        "keyMetadata": {
            "aWSAccountId": "111122223333",
            "keyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
            "arn": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321",
            "creationDate": "Oct 14, 2021, 5:39:50 PM",
            "enabled": true,
            "description": "",
            "keyUsage": "ENCRYPT_DECRYPT",
            "keyState": "Enabled",
            "origin": "AWS_CLOUDHSM",
            "customKeyStoreId": "cks-1234567890abcdef0",
            "cloudHsmClusterId": "cluster-1a23b4cdefg",
            "keyManager": "CUSTOMER",
            "customerMasterKeySpec": "SYMMETRIC_DEFAULT",
            "keySpec": "SYMMETRIC_DEFAULT",
            "encryptionAlgorithms": [
                "SYMMETRIC_DEFAULT"
            ],
            "multiRegion": false
        }
    },
    "additionalEventData": {
        "backingKey": "{\"backingKeyId\":\"backing-key-id\"}"
    },
    "requestID": "4f0b185c-588c-4767-9e90-c618f7e13cad",
    "eventID": "c73964b8-703d-49e4-bd9e-f773d0ee1e65",
    "readOnly": false,
    "resources": [
```

```
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

다음 예제는 외부 키 스토어에서 대칭 KMS 키를 생성하는 `CreateKey` 작업의 CloudTrail 로그를 보여 줍니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::111122223333:user/Alice",
            "accountId": "111122223333",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
    },
    "eventTime": "2022-09-07T22:37:45Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "tags": [],
        "keyUsage": "ENCRYPT_DECRYPT",
        "description": "",
        "origin": "EXTERNAL_KEY_STORE",
        "multiRegion": false,
        "keySpec": "SYMMETRIC_DEFAULT",
        "customerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "bypassPolicyLockoutSafetyCheck": false,
        "customKeyStoreId": "cks-1234567890abcdef0",
        "xksKeyId": "bb8562717f809024"
```

```
    },
    "responseElements": {
        "keyMetadata": {
            "aWSAccountId": "111122223333",
            "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
            "arn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "creationDate": "Dec 7, 2022, 10:37:45 PM",
            "enabled": true,
            "description": "",
            "keyUsage": "ENCRYPT_DECRYPT",
            "keyState": "Enabled",
            "origin": "EXTERNAL_KEY_STORE",
            "customKeyStoreId": "cks-1234567890abcdef0",
            "keyManager": "CUSTOMER",
            "customerMasterKeySpec": "SYMMETRIC_DEFAULT",
            "keySpec": "SYMMETRIC_DEFAULT",
            "encryptionAlgorithms": [
                "SYMMETRIC_DEFAULT"
            ],
            "multiRegion": false,
            "xksKeyConfiguration": {
                "id": "bb8562717f809024"
            }
        }
    },
    "requestID": "ba197c82-3ac7-487a-8ff4-7736bbeb1316",
    "eventID": "838ad5f4-5fdd-4044-afd7-4dbd88c6af56",
    "readOnly": false,
    "resources": [
        {
            "accountId": "227179770375",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-east-1:227179770375:key/39c5eb22-
f37c-4956-92ca-89e8f8b57ab2"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

Decrypt

이 예제에서는 Decrypt 작업에 대한 AWS CloudTrail 로그 항목을 보여줍니다.

`Decrypt` 작업에 대한 CloudTrail 로그 항목에는 요청에 암호화 알고리즘이 지정되지 않은 경우라도 항상 `requestParameters`에 `encryptionAlgorithm`이 포함되어 있습니다. 요청의 암호문과 응답의 일반 텍스트는 생략됩니다.

주제

- 표준 대칭 암호화 키로 복호화
- 표준 대칭 암호화 키로 복호화 실패
- 키 스토어의 KMS AWS CloudHSM 키로 복호화
- 외부 키 스토어에서 KMS 키로 복호화
- 외부 키 스토어에서 KMS 키로 복호화 실패
- 포스트 양자 TLS 연결을 통해 표준 대칭 암호화 키로 복호화

표준 대칭 암호화 키로 복호화

다음은 표준 대칭 암호화 키를 사용한 `Decrypt` 작업에 대한 예제 CloudTrail 로그 항목입니다.

```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2025-05-20T20:45:00Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
```

```
        "encryptionContext": {
            "Department": "Engineering",
            "Project": "Alpha"
        }
    },
    "responseElements": null,
    "additionalEventData": {
        "keyMaterialId":
 "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0"
    },
    "requestID": "12345126-30d5-4b28-98b9-9153da559963",
    "eventID": "abcde202-ba1a-467c-b4ba-f729d45ae521",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.2",
        "cipherSuite": "ECDHE-RSA-AES256-GCM-SHA384",
        "clientProvidedHostHeader": "kms.us-west-2.amazonaws.com"
    }
}
```

표준 대칭 암호화 키로 복호화 실패

다음 예제 CloudTrail 로그 항목은 표준 대칭 암호화 KMS 키를 사용하여 실패한 `Decrypt` 작업을 기록합니다. 예외(`errorCode`)와 오류 메시지(`errorMessage`)가 포함되어 있어 오류를 해결하는 데 도움이 됩니다.

이 경우 `Decrypt` 요청에 지정된 대칭 KMS 키는 데이터를 암호화하는 데 사용된 대칭 KMS 키가 아닙니다.

```
{
    "eventVersion": "1.08",
```

```
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2022-11-24T18:57:43Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "errorCode": "IncorrectKeyException"
    "errorMessage": "The key ID in the request does not identify a CMK that can perform
 this operation.",
    "requestParameters": {
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "encryptionContext": {
            "Department": "Engineering",
            "Project": "Alpha"
        }
    },
    "responseElements": null,
    "requestID": "22345126-30d5-4b28-98b9-9153da559963",
    "eventID": "abcde202-ba1a-467c-b4ba-f729d45ae521",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

키 스토어의 KMS AWS CloudHSM 키로 복호화

다음 예제 CloudTrail 로그 항목은 AWS CloudHSM 키 스토어에서 KMS 키를 사용한 Decrypt 작업을 기록합니다. 사용자 지정 키 저장소의 KMS 키를 사용하는 암호화 작업에 대한 모든 로그 항목에는 customKeyStoreId 및 backingKeyId가 있는 additionalEventData 필드가 포함됩니다. backingKeyId 필드에 반환되는 값은 CloudHSM 키 id 속성입니다. additionalEventData가 요청에 지정되지 않았습니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2021-10-26T23:41:27Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "requestParameters": {
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "encryptionContext": {
            "Department": "Development",
            "Purpose": "Test"
        }
    },
    "responseElements": null,
    "additionalEventData": {
        "customKeyStoreId": "cks-1234567890abcdef0"
    },
    "requestID": "e1b881f8-2048-41f8-b6cc-382b7857ec61",
    "eventID": "a79603d5-4cde-46fc-819c-a7cf547b9df4",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
```

```
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

외부 키 스토어에서 KMS 키로 복호화

다음 예제 CloudTrail 로그 항목은 외부 키 스토어에서 대칭 KMS 키를 사용한 Decrypt 작업을 기록합니다. additionalEventData 필드에는 customKeyStoreId 외에도 외부 키 ID(XksKeyId)가 포함됩니다. additionalEventData가 요청에 지정되지 않았습니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2022-11-24T00:26:58Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "AWS Internal",
    "requestParameters": {
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "keyId": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321",
        "encryptionContext": {
            "Department": "Engineering",
            "Purpose": "Test"
        }
    },
    "responseElements": null,
    "additionalEventData": {
        "customKeyStoreId": "cks-9876543210fedcba9",
```

```
        "xksKeyId": "abc01234567890fe"
    },
    "requestID": "f1b881f8-2048-41f8-b6cc-382b7857ec61",
    "eventID": "b79603d5-4cde-46fc-819c-a7cf547b9df4",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

외부 키 스토어에서 KMS 키로 복호화 실패

다음 예제 CloudTrail 로그 항목은 외부 키 스토어에서 KMS 키를 사용한 `Decrypt` 작업에 대한 실패한 요청을 기록합니다. CloudWatch는 성공한 요청 외에도 실패한 요청을 기록합니다. 실패를 기록할 때 CloudTrail 로그 항목에는 예외(errorCode)와 함께 제공되는 오류 메시지(errorMessage)가 포함됩니다.

이 예제와 같이 실패한 요청이 외부 키 스토어 프록시에 도달한 경우 `requestId` 값을 사용하여 실패한 요청을 외부 키 스토어 프록시가 기록하는 해당 요청과 연결할 수 있습니다(프록시가 제공하는 경우).

외부 키 스토어의 `Decrypt` 요청에 대한 도움말은 복호화 오류 섹션을 참조하세요.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2022-11-24T00:26:58Z",
```

```
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "errorCode": "KMSInvalidStateException",
    "errorMessage": "The external key store proxy rejected the request because the
 specified ciphertext or additional authenticated data is corrupted, missing, or
 otherwise invalid.",
    "requestParameters": {
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "keyId": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321",
        "encryptionContext": {
            "Department": "Engineering",
            "Purpose": "Test"
        }
    },
    "responseElements": null,
    "additionalEventData": {
        "customKeyStoreId": "cks-9876543210fedcba9",
        "xksKeyId": "abc01234567890fe"
    },
    "requestID": "f1b881f8-2048-41f8-b6cc-382b7857ec61",
    "eventID": "b79603d5-4cde-46fc-819c-a7cf547b9df4",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

포스트 양자 TLS 연결을 통해 표준 대칭 암호화 키로 복호화

다음은 포스트 양자 TLS 연결을 통해 표준 대칭 암호화 키를 사용하는 `Decrypt` 작업에 대한 CloudTrail 로그 항목의 예입니다. `tlsDetails` 섹션의 keyExchange 필드에는가 언급되어 있습니다X25519MLKEM768. 이 하이브리드 알고리즘은 오늘날 TLS에 사용되는 클래식 키 교환 알고리즘인 Curve 25519를 통한 타원 곡선 디피-헬만(ECDH)과 미국 국립 표준 기술 연구소(NIST)가 첫 번째 표준 포스트 양자 키-합의 알고리즘으로 지정한 ML-KEM퍼블릭 키 암호화 및 키 설정 알고리즘인 Module-Lattice-Based 키 캡슐화 메커니즘()을 결합하는 하이브리드 알고리즘입니다.


```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2025-11-12T15:16:26Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "aws-sdk-java/2.30.22 md/io#async md/http#AwsCommonRuntime ...",
    "requestParameters": {
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "encryptionContext": {
            "Department": "Engineering",
            "Project": "Alpha"
        }
    },
    "responseElements": null,
    "additionalEventData": {
        "keyMaterialId":
 "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0"
    },
    "requestID": "12345126-30d5-4b28-98b9-9153da559963",
    "eventID": "abcde202-ba1a-467c-b4ba-f729d45ae521",
    "readOnly": true,
    "resources": [
```

```
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_256_GCM_SHA384",
        "clientProvidedHostHeader": "kms.us-west-2.amazonaws.com",
        "keyExchange": "X25519MLKEM768"
    }
}
```

DeleteAlias

다음 예제에서는 DeleteAlias 작업에 대한 AWS CloudTrail 로그 항목을 보여줍니다. 별칭을 삭제하는 방법에 대한 자세한 내용은 별칭 삭제 단원을 참조하십시오.

2022년 12월 이후에 기록된 이 작업에 대한 CloudTrail 로그 항목에는 이 작업이 키 ARN을 반환하지 않더라도 영향을 받는 KMS 키의 키 ARN이 `responseElements.keyId` 값에 포함됩니다.

```
{
    "eventVersion": "1.02",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2014-11-04T00:52:27Z"
            }
        }
    },
```

```
    "eventTime": "2014-11-04T00:52:27Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DeleteAlias",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "aliasName": "alias/my_alias"
    },
    "responseElements": {
        "keyId":"arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "requestID": "d9542792-63bc-11e4-bc2b-4198b6150d5c",
    "eventID": "12f48554-bb04-4991-9cfc-e7e85f68eda0",
    "readOnly": false,
    "resources": [{
        "ARN": "arn:aws:kms:us-east-1:111122223333:alias/my_alias",
        "accountId": "111122223333"
    },
    {
        "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "accountId": "111122223333"
    }],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

DeleteCustomKeyStore

다음 예제는 `DeleteCustomKeyStore` 작업을 호출하여 생성된 AWS CloudTrail 로그 항목을 보여줍니다. 사용자 지정 키 스토어 생성에 대한 자세한 내용은 AWS CloudHSM 키 스토어 삭제 섹션을 참조하세요.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
```

```
        "userName": "Alice"
    },
    "eventTime": "2021-10-21T20:17:32Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DeleteCustomKeyStore",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "customKeyStoreId": "cks-1234567890abcdef0"
    },
    "responseElements": null,
    "additionalEventData": {
        "customKeyStoreName": "ExampleKeyStore",
        "clusterId": "cluster-1a23b4cdefg"
    },
    "requestID": "abcde9e1-f1a3-4460-a423-577fb6e695c9",
    "eventID": "114b61b9-0ea6-47f5-a9d2-4f2bdd0017d5",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333"
}
```


DeleteExpiredKeyMaterial

키 구성 요소를 AWS KMS key (KMS 키)로 가져올 때 해당 키 구성 요소의 만료 날짜 및 시간을 설정할 수 있습니다.는 키 구성 요소를 가져올 때(만료 설정 사용)와가 만료된 키 구성 요소를 AWS KMS 삭제할 때 CloudTrail 로그에 항목을 AWS KMS 기록합니다. 가져온 키 구성 요소가 있는 KMS 키를 생성하는 방법은 키의 AWS KMS 키 구성 요소 가져오기 단원을 참조하세요.

다음 예제는가 만료된 키 구성 요소를 AWS KMS 삭제할 때 생성된 AWS CloudTrail 로그 항목을 보여줍니다.


```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "accountId": "111122223333",
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2025-05-22T19:55:11Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DeleteExpiredKeyMaterial",
```

```
    "awsRegion": "us-east-1",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": null,
    "responseElements": null,
    "eventID": "cfa932fd-0d3a-4a76-a8b8-616863a2b547",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsServiceEvent",
    "recipientAccountId": "111122223333",
    "serviceEventDetails": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "keyMaterialId":
 "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0"
    },
    "eventCategory": "Management"
}
```

DeleteImportedKeyMaterial

키 구성 요소를 KMS 키로 가져오는 경우 DeleteImportedKeyMaterial 작업을 사용하여 언제든지 가져온 키 구성 요소를 삭제할 수 있습니다. KMS 키에서 가져온 키 구성 요소를 삭제하면 KMS 키의 키 상태가 PendingImport로 변경되고 KMS 키를 암호화 작업에 사용할 수 없습니다. 자세한 내용은 가져온 키 구성 요소 삭제을 참조하세요.

다음 예제에서는 DeleteImportedKeyMaterial 작업에 대해 생성된 AWS CloudTrail 로그 항목을 보여줍니다.

```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
```

```
        "userName": "Alice"
    },
    "eventTime": "2025-05-20T20:45:08Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DeleteImportedKeyMaterial",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "keyMaterialId":
 "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0"
    },
    "responseElements": {
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "keyMaterialId":
 "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0"
    },
    "requestID": "dcf0e82f-dad0-4622-a378-a5b964ad42c1",
    "eventID": "2afbb991-c668-4641-8a00-67d62e1fecbd",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.2",
        "cipherSuite": "ECDHE-RSA-AES256-GCM-SHA384",
        "clientProvidedHostHeader": "kms.us-west-2.amazonaws.com"
    }
}
```

DeleteKey

이 예제는 KMS 키가 삭제될 때 생성되는 AWS CloudTrail 로그 항목을 보여줍니다. KMS 키를 삭제하려면 ScheduleKeyDeletion 작업을 사용합니다. 지정된 대기 기간이 만료되면 AWS KMS 는 KMS 키를 삭제하고 CloudTrail 로그에 다음과 같은 항목을 기록하여 해당 이벤트를 기록합니다.

2022년 12월 이후에 기록된 이 작업에 대한 CloudTrail 로그 항목에는 이 작업이 키 ARN을 반환하지 않더라도 영향을 받는 KMS 키의 키 ARN이 `responseElements.keyId` 값에 포함됩니다.

`ScheduleKeyDeletion` 작업에 대한 CloudTrail 로그 항목의 예는 ScheduleKeyDeletion 단원을 참조하십시오. KMS 키 삭제에 대한 자세한 내용은 삭제 AWS KMS key 단원을 참조하십시오.

다음 예제 CloudTrail 로그 항목은 AWS KMS의 키 구성 요소가 있는 대칭 KMS 키의 `DeleteKey` 작업을 기록합니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "accountId": "111122223333",
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2020-07-31T00:07:00Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DeleteKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": null,
    "responseElements": null,
    "eventID": "b25f9cda-74e1-4458-847b-4972a0bf9668",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsServiceEvent",
    "recipientAccountId": "111122223333",
     "managementEvent": true,
    "eventCategory": "Management"
```

```
}
```

다음 CloudTrail 로그 항목은 AWS CloudHSM 사용자 지정 키 스토어에서 KMS 키의 `DeleteKey` 작업을 기록합니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "accountId": "111122223333",
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2021-10-26T23:41:27Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DeleteKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": null,
    "responseElements": {
        "keyId":"arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "additionalEventData": {
        "customKeyStoreId": "cks-1234567890abcdef0",
        "clusterId": "cluster-1a23b4cdefg",
        "backingKeys": "[{\"backingKeyId\":\"backing-key-id\"}]",
        "backingKeysDeletionStatus": "[{\"backingKeyId\":\"backing-key-id\",
\"deletionStatus\":\"SUCCESS\"}]"
    },
    "eventID": "1234585c-4b0c-4340-ab11-662414b79239",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsServiceEvent",
    "recipientAccountId": "111122223333",
    "managementEvent": true,
    "eventCategory": "Management"
```

```
}
```

DescribeCustomKeyStores

다음 예제는 DescribeCustomKeyStores 작업을 호출하여 생성된 AWS CloudTrail 로그 항목을 보여줍니다. 사용자 지정 키 스토어 보기에 대한 자세한 내용은 AWS CloudHSM 키 스토어 보기 섹션을 참조하세요.


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2021-10-21T20:17:32Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DescribeCustomKeyStores",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "customKeyStoreId": "cks-1234567890abcdef0"
    },
    "responseElements": null,
    "requestID": "abcde9e1-f1a3-4460-a423-577fb6e695c9",
    "eventID": "2ea1735f-628d-43e3-b2ee-486d02913a78",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333"
}
```


DescribeKey

다음 예제는 DescribeKey 작업에 대한 AWS CloudTrail 로그 항목을 보여줍니다.는 DescribeKey 작업을 호출하거나 AWS KMS 콘솔에서 KMS 키를 볼 때 다음과 같은 항목을 AWS KMS 기록합니다. 이 호출은 AWS KMS 관리 콘솔에서 키를 확인한 결과입니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2022-09-26T18:01:36Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DescribeKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": null,
    "requestID": "12345126-30d5-4b28-98b9-9153da559963",
    "eventID": "abcde202-ba1a-467c-b4ba-f729d45ae521",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```


DisableKey

다음 예제에서는 DisableKey 작업에 대한 AWS CloudTrail 로그 항목을 보여줍니다. 활성화 및 비활성화 AWS KMS keys 에 대한 자세한 내용은 섹션을 AWS KMS참조하세요KMS 키 활성화 및 비활성화.

2022년 12월 이후에 기록된 이 작업에 대한 CloudTrail 로그 항목에는 이 작업이 키 ARN을 반환하지 않더라도 영향을 받는 KMS 키의 키 ARN이 `responseElements.keyId` 값에 포함됩니다.

```
{
    "eventVersion": "1.02",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2014-11-04T00:52:43Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DisableKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": {
        "keyId":"arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "requestID": "12345126-30d5-4b28-98b9-9153da559963",
    "eventID": "abcde202-ba1a-467c-b4ba-f729d45ae521",
    "readOnly": false,
    "resources": [{
        "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "accountId": "111122223333"
    }],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

DisableKeyRotation

다음 예제에서는 DisableKeyRotation 작업을 호출하여 생성된 AWS CloudTrail 로그 항목을 보여줍니다. 자동 키 교체에 대한 자세한 내용은 회전 AWS KMS keys 단원을 참조하세요.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
```

```
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2022-09-01T19:31:39Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DisableKeyRotation",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": null,
    "requestID": "d6a9351a-ed6e-4581-88d1-2a9a8a538497",
    "eventID": "6313164c-83aa-4cc3-9e1a-b7c426f7a5b1",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

DisconnectCustomKeyStore

다음 예제는 `DisconnectCustomKeyStore` 작업을 호출하여 생성된 AWS CloudTrail 로그 항목을 보여줍니다. 사용자 지정 키 스토어 연결 해제에 대한 자세한 내용은 AWS CloudHSM 키 스토어 연결 해제 섹션을 참조하세요.

```
{
    "eventVersion": "1.08",
```

```
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2021-10-21T20:17:32Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DisconnectCustomKeyStore",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "customKeyStoreId": "cks-1234567890abcdef0"
    },
    "responseElements": null,
    "additionalEventData": {
        "customKeyStoreName": "ExampleKeyStore",
        "clusterId": "cluster-1a23b4cdefg"
    },
    "requestID": "abcde9e1-f1a3-4460-a423-577fb6e695c9",
    "eventID": "114b61b9-0ea6-47f5-a9d2-4f2bdd0017d5",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333"
}
```


EnableKey

다음 예제에서는 EnableKey 작업에 대한 AWS CloudTrail 로그 항목을 보여줍니다. 활성화 및 비활성화 AWS KMS keys 에 대한 자세한 내용은 단원을 AWS KMS참조하십시오KMS 키 활성화 및 비활성화.

2022년 12월 이후에 기록된 이 작업에 대한 CloudTrail 로그 항목에는 이 작업이 키 ARN을 반환하지 않더라도 영향을 받는 KMS 키의 키 ARN이 `responseElements.keyId` 값에 포함됩니다.


```
{
    "eventVersion": "1.02",
    "userIdentity": {
        "type": "IAMUser",
```

```
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2014-11-04T00:52:20Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "EnableKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": {
        "keyId":"arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "requestID": "d528a6fb-63bc-11e4-bc2b-4198b6150d5c",
    "eventID": "be393928-3629-4370-9634-567f9274d52e",
    "readOnly": false,
    "resources": [{
        "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "accountId": "111122223333"
    }],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```


EnableKeyRotation

다음 예제에서는 EnableKeyRotation 작업에 대한 호출의 AWS CloudTrail 로그 항목을 보여줍니다. 키가 교체될 때 기록되는 CloudTrail 로그 항목의 예는 RotateKey 단원을 참조하십시오. AWS KMS keys 교체에 대한 정보는 회전 AWS KMS keys 단원을 참조하십시오.

> **Note**
>
> rotation-period는 선택적 요청 파라미터입니다. 자동 키 교체를 활성화할 때 교체 기간을 지정하지 않으면 기본값은 365일입니다.

2022년 12월 이후에 기록된 이 작업에 대한 CloudTrail 로그 항목에는 이 작업이 키 ARN을 반환하지 않더라도 영향을 받는 KMS 키의 키 ARN이 `responseElements.keyId` 값에 포함됩니다.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2020-07-25T23:41:56Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "EnableKeyRotation",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "rotationPeriodInDays": 180
    },
    "responseElements": {
        "keyId":"arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "requestID": "81f5b794-452b-4d6a-932b-68c188165273",
    "eventID": "fefc43a7-8e06-419f-bcab-b3bf18d6a401",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

암호화

다음 예제에서는 암호화 작업에 대한 AWS CloudTrail 로그 항목을 보여줍니다.


```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2025-05-20T20:46:16Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Encrypt",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "encryptionContext": {
            "Department": "Engineering"
        },
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
    },
    "responseElements": null,
    "additionalEventData": {
        "keyMaterialId":
 "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0"
    },
    "requestID": "f3423043-63bc-11e4-bc2b-4198b6150d5c",
    "eventID": "91235988-eb87-476a-ac2c-0cdc244e6dca",
    "readOnly": true,
    "resources": [{
        "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "accountId": "111122223333"
    }],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333"
```

```
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_256_GCM_SHA384",
        "clientProvidedHostHeader": "kms.us-east-1.amazonaws.com"
    }
}
```

GenerateDataKey

다음 예제에서는 GenerateDataKey 작업에 대한 AWS CloudTrail 로그 항목을 보여줍니다.

```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2025-05-20T20:46:16Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "keySpec": "AES_256",
        "encryptionContext": {
            "Department": "Engineering",
            "Project": "Alpha"
        }
    },
    "responseElements": null,
    "additionalEventData": {
        "keyMaterialId":
 "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0"
    },
    "requestID": "e0eb83e3-63bc-11e4-bc2b-4198b6150d5c",
    "eventID": "a9dea4f9-8395-46c0-942c-f509c02c2b71",
```

```
    "readOnly": true,
    "resources": [{
        "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "accountId": "111122223333"
    }],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333"
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_256_GCM_SHA384",
        "clientProvidedHostHeader": "kms.us-east-1.amazonaws.com"
    }
}
```

GenerateDataKeyPair

다음 예제에서는 GenerateDataKeyPair 작업에 대한 AWS CloudTrail 로그 항목을 보여줍니다. 이 예에서는 대칭 암호화 AWS KMS key으로 암호화된 RSA 키 페어를 생성하는 작업을 기록합니다.

```
{
    "eventVersion": "1.11",
    "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::111122223333:user/Alice",
            "accountId": "111122223333",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
    },
    "eventTime": "2025-05-20T20:46:16Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKeyPair",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyPairSpec": "RSA_3072",
        "encryptionContext": {
            "Project": "Alpha"
        },
```

```
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": null,
    "additionalEventData": {
        "keyMaterialId":
 "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0"
    },
    "requestID": "52fb127b-0fe5-42bb-8e5e-f560febde6b0",
    "eventID": "9b6bd6d2-529d-4890-a949-593b13800ad7",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333"
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_256_GCM_SHA384",
        "clientProvidedHostHeader": "kms.us-west-2.amazonaws.com"
    }
}
```

GenerateDataKeyPairWithoutPlaintext

다음 예제에서는 GenerateDataKeyPairWithoutPlaintext 작업에 대한 AWS CloudTrail 로그 항목을 보여줍니다. 이 예에서는 대칭 암호화 AWS KMS key으로 암호화된 RSA 키 페어를 생성하는 작업을 기록합니다.

```
{
    "eventVersion": "1.11",
    "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::111122223333:user/Alice",
            "accountId": "111122223333",
            "accessKeyId": "EXAMPLE_KEY_ID",
```

```
            "userName": "Alice"
    },
    "eventTime": "2025-05-20T20:46:16Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKeyPairWithoutPlaintext",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyPairSpec": "RSA_4096",
        "encryptionContext": {
            "Index": "5"
        },
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": null,
    "additionalEventData": {
        "keyMaterialId":
 "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0"
    },
    "requestID": "52fb127b-0fe5-42bb-8e5e-f560febde6b0",
    "eventID": "9b6bd6d2-529d-4890-a949-593b13800ad7",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333"
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_256_GCM_SHA384",
        "clientProvidedHostHeader": "kms.us-west-2.amazonaws.com"
    }
}
```

GenerateDataKeyWithoutPlaintext

다음 예제에서는 GenerateDataKeyWithoutPlaintext 작업에 대한 AWS CloudTrail 로그 항목을 보여줍니다.


```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2025-05-20T20:46:16Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKeyWithoutPlaintext",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "keySpec": "AES_256",
        "encryptionContext": {
            "Project": "Alpha"
        }
    },
    "responseElements": null,
    "additionalEventData": {
        "keyMaterialId":
 "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0"
    },
    "requestID": "d6b8e411-63bc-11e4-bc2b-4198b6150d5c",
    "eventID": "f7734272-9ec5-4c80-9f36-528ebbe35e4a",
    "readOnly": true,
    "resources": [{
        "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "accountId": "111122223333"
    }],
    "eventType": "AwsApiCall",
    "managementEvent": true,
```

```
    "recipientAccountId": "111122223333"
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_256_GCM_SHA384",
        "clientProvidedHostHeader": "kms.us-east-1.amazonaws.com"
    }
}
```

GenerateMac

다음 예제에서는 GenerateMac 작업에 대한 AWS CloudTrail 로그 항목을 보여줍니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
     },
    "eventTime": "2022-12-23T19:26:54Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateMac",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "macAlgorithm": "HMAC_SHA_512",
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": null,
    "requestID": "e0eb83e3-63bc-11e4-bc2b-4198b6150d5c",
    "eventID": "a9dea4f9-8395-46c0-942c-f509c02c2b71",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
```

```
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```


GenerateRandom

다음 예제에서는 GenerateRandom 작업에 대한 AWS CloudTrail 로그 항목을 보여줍니다. 이 작업은 를 사용하지 않으므로 AWS KMS keyresources 필드가 비어 있습니다.


```
{
    "eventVersion": "1.02",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2014-11-04T00:52:37Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateRandom",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": null,
    "responseElements": null,
    "requestID": "df1e3de6-63bc-11e4-bc2b-4198b6150d5c",
    "eventID": "239cb9f7-ae05-4c94-9221-6ea30eef0442",
    "readOnly": true,
    "resources": [],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```


GetKeyLastUsage

다음 예제에서는 GetKeyLastUsage 작업에 대한 AWS CloudTrail 로그 항목을 보여줍니다. AWS KMS는 GetKeyLastUsage 작업을 호출하거나 콘솔에서 AWS KMS KMS 키 세부 정보를 볼 때 다음과 같은 항목을 기록합니다.

```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2026-04-27T17:11:06Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GetKeyLastUsage",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": null,
    "requestID": "93746dd6-63bc-11e4-bc2b-4198b6150d5c",
    "eventID": "4aa7e4d5-d047-452a-a5a6-2cce282a7e82",
    "readOnly": true,
    "resources": [{
        "accountId": "111122223333",
        "type": "AWS::KMS::Key",
        "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    }],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```


GetKeyPolicy

다음 예제에서는 GetKeyPolicy 작업에 대한 AWS CloudTrail 로그 항목을 보여줍니다. KMS 키의 키 정책을 보는 방법은 키 정책 보기 단원을 참조하십시오.


```
{
    "eventVersion": "1.02",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
```

```
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2014-11-04T00:50:30Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GetKeyPolicy",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "policyName": "default"
    },
    "responseElements": null,
    "requestID": "93746dd6-63bc-11e4-bc2b-4198b6150d5c",
    "eventID": "4aa7e4d5-d047-452a-a5a6-2cce282a7e82",
    "readOnly": true,
    "resources": [{
        "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "accountId": "111122223333"
    }],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

GetKeyRotationStatus

다음 예제에서는 GetKeyRotationStatus 작업에 대한 AWS CloudTrail 로그 항목을 보여줍니다. KMS 키의 키 구성 요소 자동 교체 및 온디맨드 교체에 대한 자세한 내용은 회전 AWS KMS keys 단원을 참조하세요.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
```

```
    },
    "eventTime": "2024-02-20T19:16:45Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GetKeyRotationStatus",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": null,
    "requestID": "12f9b7e8-49b9-4c1c-a7e3-34ac0cdf0467",
    "eventID": "3d082126-9e7d-4167-8372-a6cfcbed4be6",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
    "tlsDetails": {
        "tlsVersion": "TLSv1.2",
        "cipherSuite": "ECDHE-RSA-AES256-GCM-SHA384",
        "clientProvidedHostHeader": "kms.us-east-1.amazonaws.com"
    }
}
```


GetParametersForImport

다음 예제에서는 GetParametersForImport 작업을 사용할 때 생성된 AWS CloudTrail 로그 항목을 보여 줍니다. 이 작업은 KMS 키로 키 자료를 가져올 때 사용하는 퍼블릭 키와 가져오기 토큰을 반환합니다. `GetParametersForImport` 작업을 사용하거나 AWS KMS 콘솔을 사용하여 퍼블릭 키와 가져오기 토큰을 다운로드할 때 동일한 CloudTrail 항목이 기록됩니다.


```
{
    "eventVersion": "1.05",
    "userIdentity": {
```

```
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::111122223333:user/Alice",
            "accountId": "111122223333",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
    },
    "eventTime": "2020-07-25T23:58:23Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GetParametersForImport",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "wrappingAlgorithm": "RSAES_OAEP_SHA_256",
        "wrappingKeySpec": "RSA_2048"
    },
    "responseElements": null,
    "requestID": "b5786406-e3c7-43d6-8d3c-6d5ef96e2278",
    "eventID": "4023e622-0c3e-4324-bdef-7f58193bba87",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```


ImportKeyMaterial

다음 예제에서는 ImportKeyMaterial 작업을 사용할 때 생성된 AWS CloudTrail 로그 항목을 보여줍니다. `ImportKeyMaterial` 작업을 사용하거나 AWS KMS 콘솔을 사용하여 키 구성 요소를 로 가져올 때 동일한 CloudTrail 항목이 기록됩니다 AWS KMS key.

2022년 12월 이후에 기록된 이 작업에 대한 CloudTrail 로그 항목에는 이 작업이 키 ARN을 반환하지 않더라도 영향을 받는 KMS 키의 키 ARN이 `responseElements.keyId` 값에 포함됩니다.

```
{
    "eventVersion": "1.11",
    "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::111122223333:user/Alice",
            "accountId": "111122223333",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
    },
    "eventTime": "2025-05-21T05:42:31Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "ImportKeyMaterial",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "validTo": "May 21, 2025, 5:47:45 AM",
        "expirationModel": "KEY_MATERIAL_EXPIRES",
        "importType": "NEW_KEY_MATERIAL",
        "keyMaterialDescription": "ExampleKeyMaterialA"
    },
    "responseElements": {
        "keyId":"arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "keyMaterialId":
 "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0"
    },
    "requestID": "89e10ee7-a612-414d-95a2-a128346969fd",
    "eventID": "c7abd205-a5a2-4430-bbfa-fc10f3e2d79f",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333"
```

```
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_256_GCM_SHA384",
        "clientProvidedHostHeader": "kms.us-west-2.amazonaws.com"
    }
}
```

ListAliases

다음 예제에서는 ListAliases 작업에 대한 AWS CloudTrail 로그 항목을 보여줍니다. 이 작업은 특정 별칭 또는를 사용하지 않으므로 AWS KMS keyresources 필드가 비어 있습니다. 에서 별칭을 보는 방법에 대한 자세한 내용은 섹션을 AWS KMS참조하세요KMS 키의 별칭 이름 및 별칭 ARN 찾기.

```
{
    "eventVersion": "1.02",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2014-11-04T00:51:45Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "ListAliases",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "limit": 5,
        "marker":
 "eyJiIjoiYWxpYXMvZTU0Y2MxOTMtYTMwNC00YzEwLTliZWItYTJjZjA3NjA2OTJhIiwiYSI6ImFsaWFzL2U1NGNjMTkzL
    },
    "responseElements": null,
    "requestID": "bfe6c190-63bc-11e4-bc2b-4198b6150d5c",
    "eventID": "a27dda7b-76f1-4ac3-8b40-42dfba77bcd6",
    "readOnly": true,
    "resources": [],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
```

```
}
```

ListGrants

다음 예제에서는 ListGrant 작업에 대한 AWS CloudTrail 로그 항목을 보여줍니다. 의 권한 부여에 대한 자세한 내용은 섹션을 AWS KMS참조하세요의 권한 부여 AWS KMS.

```
{
    "eventVersion": "1.02",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2014-11-04T00:52:49Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "ListGrants",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "marker":
 "eyJncmFudElkIjoiMWY4M2U2ZmM0YTY2NDgxYjQ2Yzc4MTdhM2Y4YmQwMDFkZDNiYmQ1MGVlYTMyY2RmOWFiNWY1Nzc1N
\u003d\u003d",
        "limit": 10
    },
    "responseElements": null,
    "requestID": "e5c23960-63bc-11e4-bc2b-4198b6150d5c",
    "eventID": "d24380f5-1b20-4253-8e92-dd0492b3bd3d",
    "readOnly": true,
    "resources": [{
        "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "accountId": "111122223333"
    }],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

ListKeyRotations

다음 예제에서는 ListKeyRotations 작업에 대한 AWS CloudTrail 로그 항목을 보여줍니다. KMS 키의 키 구성 요소 자동 교체 및 온디맨드 교체에 대한 자세한 내용은 회전 AWS KMS keys 단원을 참조하세요.


```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2025-05-21T05:42:35Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "ListKeyRotations",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "includeKeyMaterial": "ALL_KEY_MATERIAL"
    },
    "responseElements": null,
    "requestID": "99c88d32-f2db-455e-8a9a-23855258a452",
    "eventID": "8ce0e74b-b9c7-45a2-96ef-83136d38068e",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management",
    "tlsDetails": {
```

```
        "tlsVersion": "TLSv1.2",
        "cipherSuite": "ECDHE-RSA-AES256-GCM-SHA384",
        "clientProvidedHostHeader": "kms.us-east-1.amazonaws.com"
    }
}
```

PutKeyPolicy

다음 예제는 PutKeyPolicy 작업을 호출하여 생성된 AWS CloudTrail 로그 항목을 보여줍니다. 키 정책 업데이트에 대한 자세한 내용은 키 정책 변경 단원을 참조하세요.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2022-09-01T20:06:16Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "PutKeyPolicy",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "policyName": "default",
        "policy": "{\n  \"Version\" : \"2012-10-17\",\n  \"Id\" : \"key-default-1\",\n
 \"Statement\" : [ {\n    \"Sid\" : \"Enable IAM User Permissions\",\n    \"Effect\" :
 \"Allow\",\n    \"Principal\" : {\n      \"AWS\" : \"arn:aws:iam::111122223333:root
\"\n    },\n    \"Action\" : \"kms:*\",\n    \"Resource\" : \"*\"\n  } ]\n}",
        "bypassPolicyLockoutSafetyCheck": false
    },
    "responseElements": null,
    "requestID": "7bb906fa-dc21-4350-b65c-808ff0f72f55",
    "eventID": "c217db1f-903f-4a2f-8f88-9580182d6313",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
```

```
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

ReEncrypt

다음 예제에서는 ReEncrypt 작업에 대한 AWS CloudTrail 로그 항목을 보여줍니다. 이 로그 항목의 `resources` 필드는 AWS KMS keys소스 KMS 키와 대상 KMS 키를 순서대로 지정합니다.

```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2025-05-22T19:34:55Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "ReEncrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "sourceEncryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "sourceEncryptionContext": {
            "Project": "Alpha",
            "Department": "Engineering"
        },
        "destinationKeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
        "destinationEncryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "destinationEncryptionContext": {
            "Level": "3A"
        }
```

```
    },
    "responseElements": null,
    "additionalEventData": {
        "destinationKeyMaterialId":
 "96083e4fb6dbc41d77578a213a6b6669c044dd4c143e96755396d2bf11fd6068",
        "sourceKeyMaterialId":
 "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0"
    },
    "requestID": "03769fd4-acf9-4b33-adf3-2ab8ca73aadf",
    "eventID": "542d9e04-0e8d-4e05-bf4b-4bdeb032e6ec",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        },
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333"
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_256_GCM_SHA384",
        "clientProvidedHostHeader": "kms.us-west-2.amazonaws.com"
    }
}
```


ReplicateKey

다음 예제는 ReplicateKey 작업을 호출하여 생성된 AWS CloudTrail 로그 항목을 보여줍니다. `ReplicateKey` 요청으로 인해 `ReplicateKey` 작업과 CreateKey 작업이 수행됩니다.

다중 리전 키 복제에 대한 자세한 내용은 다중 리전 복제본 키 생성 단원을 참조하십시오.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2020-11-18T01:29:18Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "ReplicateKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "replicaRegion": "us-west-2",
        "bypassPolicyLockoutSafetyCheck": false,
        "description": ""
    },
    "responseElements": {
        "replicaKeyMetadata": {
            "aWSAccountId": "111122223333",
            "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
            "arn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "creationDate": "Nov 18, 2020, 1:29:18 AM",
            "enabled": false,
            "description": "",
            "keyUsage": "ENCRYPT_DECRYPT",
            "keyState": "Creating",
            "origin": "AWS_KMS",
            "keyManager": "CUSTOMER",
            "keySpec": "SYMMETRIC_DEFAULT",
            "customerMasterKeySpec": "SYMMETRIC_DEFAULT",
            "encryptionAlgorithms": [
                "SYMMETRIC_DEFAULT"
            ],
            "multiRegion": true,
            "multiRegionConfiguration": {
                "multiRegionKeyType": "REPLICA",
```

```
                "primaryKey": {
                    "arn": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
                    "region": "us-east-1"
                },
                "replicaKeys": [
                    {
                        "arn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
                        "region": "us-west-2"
                    }
                ]
            }
        },
        "replicaPolicy": "{\n  \"Version\":\"2012-10-17\",\n  \"Statement\":[{\n
 \"Effect\":\"Allow\",\n    \"Principal\":{\"AWS\":\"arn:aws:iam::123456789012:user/
Alice\"},\n    \"Action\":\"kms:*\",\n    \"Resource\":\"*\"\n  }, {\n    \"Effect
\":\"Allow\",\n    \"Principal\":{\"AWS\":\"arn:aws:iam::012345678901:user/Bob\"},\n
    \"Action\":\"kms:CreateGrant\",\n    \"Resource\":\"*\"\n  }, {\n    \"Effect\":
\"Allow\",\n    \"Principal\":{\"AWS\":\"arn:aws:iam::012345678901:user/Charlie\"},\n
  \"Action\":\"kms:Encrypt\",\n    \"Resource\":\"*\"\n}]\n}",
    },
    "requestID": "abcdef68-63bc-11e4-bc2b-4198b6150d5c",
    "eventID": "fedcba44-6773-4f96-8763-1993aec9ae6a",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```


RetireGrant

다음 예제는 RetireGrant 작업을 호출하여 생성된 AWS CloudTrail 로그 항목을 보여줍니다. 권한 사용 중지에 대한 자세한 내용은 권한 부여 사용 중지 및 취소 단원을 참조하세요.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2022-09-01T19:39:33Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "RetireGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData": {
        "grantId": "abcde1237f76e4ba7987489ac329fbfba6ad343d6f7075dbd1ef191f0120514a"
    },
    "requestID": "1d274d57-5697-462c-a004-f25fcc29fa26",
    "eventID": "0771bcfb-3e24-4332-9ac8-e1c06563eecf",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```


RevokeGrant

다음 예제는 RevokeGrant 작업을 호출하여 생성된 AWS CloudTrail 로그 항목을 보여줍니다. 권한 부여 취소에 대한 자세한 내용은 권한 부여 사용 중지 및 취소 단원을 참조하세요.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2022-09-01T19:35:17Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "RevokeGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "grantId": "abcde1237f76e4ba7987489ac329fbfba6ad343d6f7075dbd1ef191f0120514a"
    },
    "responseElements": null,
    "requestID": "59d94c03-c5b7-428d-ae6e-f2c4b47d2917",
    "eventID": "07a23a39-6526-4ae2-b31e-d35fbe9e24ee",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```


RotateKey

이 예제에서는를 교체하는 작업에 대한 AWS CloudTrail 로그 항목을 보여줍니다 AWS KMS keys. KMS 키 교체에 대한 자세한 내용은 회전 AWS KMS keys 단원을 참조하십시오.

다음 예제에서는 자동 키 교체가 활성화된 대칭 암호화 KMS 키를 교체하는 작업에 대한 CloudTrail 로그 항목을 보여줍니다. 자동 키 교체 활성화에 대한 자세한 내용은 회전 AWS KMS keys 단원을 참조하세요.

`EnableKeyRotation` 작업을 기록하는 CloudTrail 로그 항목의 예제는 EnableKeyRotation 단원을 참조하십시오.


```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "accountId": "111122223333",
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2025-05-20T20:44:37Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "RotateKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": null,
    "responseElements": null,
    "eventID": "a24b3967-ddad-417f-9b22-2332b918db06",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "serviceEventDetails": {
        "rotationType": "AUTOMATIC",
        "keyOrigin": "AWS_KMS",
        "previousKeyMaterialId":
 "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0",
        "currentKeyMaterialId":
 "96083e4fb6dbc41d77578a213a6b6669c044dd4c143e96755396d2bf11fd6068",
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
```

```
    "eventCategory": "Management"
}
```

다음 예제는 RotateKeyOnDemand 작업으로 시작된 온디맨드 교체의 CloudTrail 로그 항목을 보여줍니다. 대칭 암호화 KMS 키의 온디맨드 교체에 대한 자세한 내용은 온디맨드 키 교체 수행 페이지를 참조하세요.

RotateKeyOnDemand 작업을 기록하는 CloudTrail 로그 항목의 예제는 RotateKeyOnDemand 단원을 참조하십시오.

```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "accountId": "111122223333",
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2025-05-20T20:44:37Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "RotateKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": null,
    "responseElements": null,
    "eventID": "a24b3967-ddad-417f-9b22-2332b918db06",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "serviceEventDetails": {
        "rotationType": "ON_DEMAND",
        "keyOrigin": "EXTERNAL",
        "previousKeyMaterialId":
 "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0",
```

```
        "currentKeyMaterialId":
 "96083e4fb6dbc41d77578a213a6b6669c044dd4c143e96755396d2bf11fd6068",
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "eventCategory": "Management"
}
```

RotateKeyOnDemand

다음 예제에서는 RotateKeyOnDemand 작업에 대한 AWS CloudTrail 로그 항목을 보여줍니다. 키가 교체될 때 기록되는 CloudTrail 로그 항목의 예는 RotateKey 단원을 참조하십시오. KMS 키의 키 구성 요소 온디맨드 교체에 대한 더 자세한 내용은 온디맨드 키 교체 수행 단원을 참조하세요.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2024-02-20T17:41:57Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "RotateKeyOnDemand",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "requestID": "9e1dee86-eb84-42fd-8f25-e3fc7dbb32c8",
    "eventID": "00a09fbc-20d6-4a58-9b92-7da85984ab77",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
```

```
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.2",
        "cipherSuite": "ECDHE-RSA-AES256-GCM-SHA384",
        "clientProvidedHostHeader": "kms.us-east-1.amazonaws.com"
    }
}
```

ScheduleKeyDeletion

이 예제에서는 ScheduleKeyDeletion 작업에 대한 AWS CloudTrail 로그 항목을 보여줍니다.

키 삭제 시 작성되는 CloudTrail 로그 항목의 예는 DeleteKey 단원을 참조하십시오. AWS KMS keys삭제에 대한 자세한 내용은 삭제 AWS KMS key 단원을 참조하십시오.

다음 예제에서는 단일 리전 KMS 키에 대한 `ScheduleKeyDeletion` 요청을 기록합니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::111122223333:user/Alice",
            "accountId": "111122223333",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
    },
    "eventTime": "2021-03-23T18:58:30Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "ScheduleKeyDeletion",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "pendingWindowInDays": 20,
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
```

```
    },
    "responseElements": {
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "keyState": "PendingDeletion",
        "deletionDate": "Apr 12, 2021 18:58:30 PM"
    },
    "requestID": "ee408f36-ea01-422b-ac14-b0f147c68334",
    "eventID": "3c4226b0-1e81-48a8-a333-7fa5f3cbd118",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

다음 예제에서는 다중 리전 KMS 키와 복제본 키에 대한 ScheduleKeyDeletion 요청을 기록합니다.

AWS KMS 는 모든 복제본 키가 삭제될 때까지 다중 리전 키를 삭제하지 않으므로 responseElements 필드에서 keyState는 PendingReplicaDeletion 이고 deletionDate 필드는 생략됩니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::111122223333:user/Alice",
            "accountId": "111122223333",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
    },
    "eventTime": "2021-10-28T17:59:05Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "ScheduleKeyDeletion",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
```

```
    "userAgent": "AWS Internal",
    "requestParameters": {
        "pendingWindowInDays": 30,
        "keyId": "mrk-1234abcd12ab34cd56ef1234567890ab"
    },
    "responseElements": {
        "keyId": "arn:aws:kms:us-west-2:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
        "keyState": "PendingReplicaDeletion",
        "pendingWindowInDays": 30
    },
    "requestID": "12341411-d846-42a6-a476-b1cbe3011f89",
    "eventID": "abcda5f-396d-494c-9380-0c47860df5f1",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

다음 예시에서는 AWS CloudHSM 사용자 지정 키 스토어에 KMS 키에 대한 `ScheduleKeyDeletion` 요청을 기록합니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::111122223333:user/Alice",
            "accountId": "111122223333",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
    },
    "eventTime": "2021-10-26T23:25:25Z",
    "eventSource": "kms.amazonaws.com",
```

```
    "eventName": "ScheduleKeyDeletion",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321",
        "pendingWindowInDays": 30
    },
    "responseElements": {
        "keyId": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321",
        "deletionDate": "Nov 2, 2021, 11:25:25 PM",
        "keyState": "PendingDeletion",
        "pendingWindowInDays": 30
    },
    "additionalEventData": {
        "customKeyStoreId": "cks-1234567890abcdef0",
        "clusterId": "cluster-1a23b4cdefg",
        "backingKeys": "[{\"backingKeyId\":\"backing-key-id\"}]"
    },
    "requestID": "abcd9f60-2c9c-4a0b-a456-d5d998f7f321",
    "eventID": "ca01996a-01b0-4edd-bbbb-25d7b6d1a6fa",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```


Sign

이 예제에서는 서명 작업에 대한 AWS CloudTrail 로그 항목을 보여줍니다.

다음은 파일의 디지털 서명을 생성하기 위해 비대칭 RSA KMS 키를 사용하는 Sign 작업에 대한 CloudTrail 로그 항목을 보여주는 예제입니다.


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2022-03-07T22:36:44Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Sign",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "messageType": "RAW",
        "keyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
        "signingAlgorithm": "RSASSA_PKCS1_V1_5_SHA_256"
    },
    "responseElements": null,
    "requestID": "8d0b35e0-46cf-48b9-be99-bf2ebc9ab9fb",
    "eventID": "107b3cac-b125-4556-9702-12a2b9afcff7",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

SynchronizeMultiRegionKey

다음 예제는가 다중 리전 키를 AWS KMS 동기화할 때 생성된 AWS CloudTrail 로그 항목을 보여줍니다. 동기화에는 다중 리전 기본 키의 공유 속성을 복제본 키에 복사하기 위한 교차 리전 호출이 포함됩니다.는 관련된 모든 다중 리전 키가 동일한 키 구성 요소를 갖도록 다중 리전 키를 주기적으로 AWS KMS 동기화합니다.

CloudTrail 로그 항목의 `resources` 요소에는 AWS 리전를 비롯한 다중 리전 기본 키의 키 ARN이 포함됩니다. 관련 다중 리전 복제 키와 해당 리전은 이 로그 항목에 나열되지 않습니다.

2022년 12월 이후에 기록된 이 작업에 대한 CloudTrail 로그 항목에는 이 작업이 키 ARN을 반환하지 않더라도 영향을 받는 KMS 키의 키 ARN이 `responseElements.keyId` 값에 포함됩니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "accountId": "111122223333",
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2020-11-18T02:04:37Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "SynchronizeMultiRegionKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": null,
    "responseElements": {
        "keyId":"arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "requestID": "12345681-de97-42e9-bed0-b02ae1abd8dc",
    "eventID": "abcdec99-2b5c-4670-9521-ddb8f031e146",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
```

```
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

TagResource

다음 예제는 태그 키가 Department이고 태그 값이 인 태그를 추가하기 위한 TagResource 작업에 대한 호출의 AWS CloudTrail 로그 항목을 보여줍니다IT.

키가 교체될 때 작성되는 UntagResource CloudTrail 로그 항목의 예는 UntagResource 단원을 참조하십시오. 태그 지정에 대한 자세한 내용은 섹션을 AWS KMS keys참조하세요의 태그 AWS KMS.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2020-07-01T21:19:25Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "TagResource",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "tags": [
            {
                "tagKey": "Department",
                "tagValue": "IT"
            }
        ]
    },
    "responseElements": null,
    "requestID": "b942584a-f77d-4787-9feb-b9c5be6e746d",
    "eventID": "0a091b9b-0df5-4cf9-b667-6f2879532b8f",
    "readOnly": false,
    "resources": [
```

```
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

UntagResource

다음 예제는 태그 키가 인 태그를 삭제하기 위한 UntagResource 작업에 대한 호출의 AWS CloudTrail 로그 항목을 보여줍니다`Dept`.

2022년 12월 이후에 기록된 이 작업에 대한 CloudTrail 로그 항목에는 이 작업이 키 ARN을 반환하지 않더라도 영향을 받는 KMS 키의 키 ARN이 `responseElements.keyId` 값에 포함됩니다.

`TagResource` CloudTrail 로그 항목에 예제는 TagResource 단원을 참조하십시오. AWS KMS keys태그 지정에 대한 자세한 내용은 의 태그 AWS KMS 단원을 참조하십시오.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2020-07-01T21:19:19Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "UntagResource",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "tagKeys": [
            "Dept"
```

```
        ]
    },
    "responseElements": {
        "keyId":"arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "requestID": "cb1d507b-6015-47f4-812b-179713af8068",
    "eventID": "0b00f4b0-036e-411d-aa75-87eb4a35a4b3",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

UpdateAlias

다음 예제에서는 UpdateAlias 작업에 대한 AWS CloudTrail 로그 항목을 보여줍니다. `resources` 요소에는 별칭 및 KMS 키 리소스에 대한 필드가 포함됩니다. 에서 별칭을 생성하는 방법에 대한 자세한 내용은 섹션을 AWS KMS참조하세요별칭 생성.

2022년 12월 이후에 기록된 이 작업에 대한 CloudTrail 로그 항목에는 이 작업이 키 ARN을 반환하지 않더라도 영향을 받는 KMS 키의 키 ARN이 `responseElements.keyId` 값에 포함됩니다.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
         "principalId": "EX_PRINCIPAL_ID",
         "arn": "arn:aws:iam::111122223333:user/Alice",
         "accountId": "111122223333",
         "accessKeyId": "EXAMPLE_KEY_ID",
         "userName": "Alice"
    },
    "eventTime": "2020-11-13T23:18:15Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "UpdateAlias",
```

```
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "aliasName": "alias/my_alias",
        "targetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": {
        "keyId":"arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "requestID": "d9472f40-63bc-11e4-bc2b-4198b6150d5c",
    "eventID": "f72d3993-864f-48d6-8f16-e26e1ae8dff0",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:111122223333:alias/my_alias"
        },
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

UpdateCustomKeyStore

다음 예제는 사용자 지정 키 스토어의 클러스터 ID를 업데이트하기 위해 UpdateCustomKeyStore 작업을 호출하여 생성된 AWS CloudTrail 로그 항목을 보여줍니다. 사용자 지정 키 스토어 편집에 대한 자세한 내용은 AWS CloudHSM 키 스토어 설정 편집 섹션을 참조하세요.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
```

```
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2021-10-21T20:17:32Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "UpdateCustomKeyStore",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "customKeyStoreId": "cks-1234567890abcdef0",
        "clusterId": "cluster-1a23b4cdefg"
    },
    "responseElements": null,
    "additionalEventData": {
        "customKeyStoreName": "ExampleKeyStore",
        "clusterId": "cluster-1a23b4cdefg"
    },
    "requestID": "abcde9e1-f1a3-4460-a423-577fb6e695c9",
    "eventID": "114b61b9-0ea6-47f5-a9d2-4f2bdd0017d5",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333"
}
```


UpdateKeyDescription

다음 예제는 UpdateKeyDescription 작업을 호출하여 생성된 AWS CloudTrail 로그 항목을 보여줍니다.


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2022-09-01T19:22:40Z",
    "eventSource": "kms.amazonaws.com",
```

```
    "eventName": "UpdateKeyDescription",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "description": "New key description"
    },
    "responseElements": null,
    "requestID": "8c3c1f8b-336d-4896-b034-4eb9916bc9b3",
    "eventID": "f5f3d548-2e9e-4658-8427-9dcb5b1ea791",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

UpdatePrimaryRegion

다음 예제는 다중 리전 키에서 UpdatePrimaryRegion 작업을 호출하여 생성되는 AWS CloudTrail 로그 항목을 보여줍니다.

`UpdatePrimaryRegion` 작업은 두 개의 CloudTrail 로그 항목을 작성합니다. 하나는 리전에 복제 키로 변환된 다중 리전 기본 키가 있고, 하나는 기본 키로 변환된 다중 리전 복제 키가 있는 리전에 있습니다.

2022년 12월 이후에 기록된 이 작업에 대한 CloudTrail 로그 항목에는 이 작업이 키 ARN을 반환하지 않더라도 영향을 받는 KMS 키의 키 ARN이 `responseElements.keyId` 값에 포함됩니다.

다음 예제는 다중 리전 키가 기본 키에서 복제 키로 변경된 리전(us-west-2)의 `UpdatePrimaryRegion`에 대한 CloudTrail 로그 항목을 보여줍니다. `primaryRegion` 필드는 이제 기본 키를 호스트하는 리전(ap-northeast-1)을 보여 줍니다.

```
{
```

```
    "eventVersion": "1.08",
    "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::111122223333:user/Alice",
            "accountId": "111122223333",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
    },
    "eventTime": "2021-03-10T20:23:37Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "UpdatePrimaryRegion",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "mrk-1234abcd12ab34cd56ef1234567890ab",
        "primaryRegion": "ap-northeast-1"
    },
    "responseElements": {
        "keyId":"arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "requestID": "ee408f36-ea01-422b-ac14-b0f147c68334",
    "eventID": "3c4226b0-1e81-48a8-a333-7fa5f3cbd118",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333"
}
```

다음 예제는 다중 리전 키가 복제 키에서 기본 키로 변경된 리전(ap-northeast-1)의 UpdatePrimaryRegion에 대한 CloudTrail 항목을 보여줍니다. 이 로그 항목은 이전 기본 리전을 식별하지 않습니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::111122223333:user/Alice",
            "accountId": "111122223333",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice",
            "invokedBy": "kms.amazonaws.com"
    },
    "eventTime": "2021-03-10T20:23:37Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "UpdatePrimaryRegion",
    "awsRegion": "ap-northeast-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "arn:aws:kms:ap-northeast-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
        "primaryRegion": "ap-northeast-1"
    },
    "responseElements": {
        "keyId":"arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "requestID": "ee408f36-ea01-422b-ac14-b0f147c68334",
    "eventID": "091e6be5-737f-43c6-8431-e3679d6d0619",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333"
}
```


VerifyMac

다음 예제에서는 VerifyMac 작업에 대한 AWS CloudTrail 로그 항목을 보여줍니다.


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
```

```
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
     },
    "eventTime": "2022-03-31T19:25:54Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "VerifyMac",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "macAlgorithm": "HMAC_SHA_384",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": null,
    "requestID": "f35da560-edff-4d6e-9b40-fb306fa9ef1e",
    "eventID": "6b464487-6dea-44cd-84ad-225d7450c975",
    "readOnly": true,
    "resources": [
        {
           "accountId": "111122223333",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

확인

이 예제에서는 확인 작업에 대한 AWS CloudTrail 로그 항목을 보여줍니다.

다음은 디지털 서명을 검증하기 위해 비대칭 RSA KMS 키를 사용하는 Verify 작업에 대한 CloudTrail 로그 항목을 보여주는 예제입니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
```

```
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2022-03-07T22:50:41Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Verify",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "signingAlgorithm": "RSASSA_PKCS1_V1_5_SHA_256",
        "keyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
        "messageType": "RAW"
    },
    "responseElements": null,
    "requestID": "c73ab82a-af82-4750-ae2c-b6bb790e9c28",
    "eventID": "3b4331cd-5b7b-4de5-bf5f-82ec22f0dac0",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```


Amazon EC2 예제 1

다음 예제는 Amazon EC2 관리 콘솔에서 기본 볼륨 키를 사용해 암호화된 볼륨을 생성하는 IAM 보안 주체를 기록합니다.

다음 예제는 사용자 Alice가 Amazon EC2 관리 콘솔에서 기본 볼륨 키로 암호화된 볼륨을 생성하는 CloudTrail 로그 항목을 보여줍니다. EC2 로그 파일 레코드에는 값이 "vol-13439757"인

volumeId 필드가 포함됩니다. AWS KMS 레코드에는 값이 인 encryptionContext 필드가 포함됩니다"aws:ebs:id": "vol-13439757". 마찬가지로, 두 레코드 사이의 principalId와 accountId가 일치합니다. 이 레코드는 암호화된 볼륨 생성 시 볼륨 콘텐츠를 암호화하는 데 사용되는 데이터 키가 생성된다는 사실을 보여줍니다.


```
{
  "Records": [
     {
      "eventVersion": "1.02",
      "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::111122223333:user/Alice",
            "accountId": "111122223333",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
        },
      "eventTime": "2014-11-05T20:50:18Z",
      "eventSource": "ec2.amazonaws.com",
      "eventName": "CreateVolume",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "192.0.2.0",
      "userAgent": "AWS Internal",
      "requestParameters": {
        "size": "10",
        "zone": "us-east-1a",
        "volumeType": "gp2",
        "encrypted": true
      },
      "responseElements": {
        "volumeId": "vol-13439757",
        "size": "10",
        "zone": "us-east-1a",
        "status": "creating",
        "createTime": 1415220618876,
        "volumeType": "gp2",
        "iops": 30,
        "encrypted": true
      },
      "requestID": "1565210e-73d0-4912-854c-b15ed349e526",
      "eventID": "a3447186-135f-4b00-8424-bc41f1a93b4f",
      "eventType": "AwsApiCall",
      "recipientAccountId": "123456789012"
```

```
    },
    {
      "eventVersion": "1.02",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
      },
      "eventTime": "2014-11-05T20:50:19Z",
      "eventSource": "kms.amazonaws.com",
      "eventName": "GenerateDataKeyWithoutPlaintext",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "192.0.2.0",
      "userAgent": "&AWS; Internal",
      "requestParameters": {
        "encryptionContext": {
          "aws:ebs:id": "vol-13439757"
        },
        "numberOfBytes": 64,
        "keyId": "alias/aws/ebs"
      },
      "responseElements": null,
      "requestID": "create-123456789012-758241111-1415220618",
      "eventID": "4bd2a696-d833-48cc-b72c-05e61b608399",
      "readOnly": true,
      "resources": [
        {
          "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
          "accountId": "111122223333"
        }
      ],
      "eventType": "AwsApiCall",
      "recipientAccountId": "111122223333"
    }
  ]
}
```

Amazon EC2 예제 2

다음 예제에서는 Amazon EC2 인스턴스를 실행하는 IAM 보안 주체가 KMS 키로 암호화된 데이터 볼륨을 생성하고 탑재합니다. 이 작업은 여러 CloudTrail 로그 레코드를 생성합니다. Amazon EBS 및 암호화에 대한 자세한 내용은 Amazon EBS 암호화 요구 사항을 참조하세요.

볼륨이 생성되면 고객을 대신하여 작업하는 Amazon EC2는 AWS KMS ()에서 암호화된 데이터 키를 가져옵니다`GenerateDataKeyWithoutPlaintext`. 그런 다음 데이터 키를 복호화할 수 있는 권한 부여(`CreateGrant`)를 생성합니다. 볼륨이 마운트되면 Amazon EC2는를 호출 AWS KMS 하여 데이터 키()를 해독합니다`Decrypt`.

Amazon EC2 인스턴스의 `instanceId("i-81e2f56c")`는 `RunInstances` 이벤트에 나타납니다. 동일한 인스턴스 ID는 생성된 권한 부여(`"111122223333:aws:ec2-infrastructure:i-81e2f56c"`)의 `granteePrincipal` 및 `Decrypt` 호출(`("arn:aws:sts::111122223333:assumed-role/aws:ec2-infrastructure/i-81e2f56c")`에서 보안 주체인 위임된 역할에 자격을 부여합니다.

데이터 볼륨을 보호하는 KMS 키의 키 ARN은 세 가지 AWS KMS 호출(`arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`,`CreateGrant` `GenerateDataKeyWithoutPlaintext`및 `Decrypt`) 모두에 나타납니다.

```
{
  "Records": [
    {
      "eventVersion": "1.02",
      "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::111122223333:user/Alice",
            "accountId": "111122223333",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
      },
      "eventTime": "2014-11-05T21:35:27Z",
      "eventSource": "ec2.amazonaws.com",
      "eventName": "RunInstances",
      "awsRegion": "us-west-2",
      "sourceIPAddress": "192.0.2.0",
      "userAgent": "AWS Internal",
      "requestParameters": {
        "instancesSet": {
          "items": [
```

```
            {
              "imageId": "ami-b66ed3de",
              "minCount": 1,
              "maxCount": 1
            }
          ]
        },
        "groupSet": {
          "items": [
            {
              "groupId": "sg-98b6e0f2"
            }
          ]
        },
        "instanceType": "m3.medium",
        "blockDeviceMapping": {
          "items": [
            {
              "deviceName": "/dev/xvda",
              "ebs": {
                "volumeSize": 8,
                "deleteOnTermination": true,
                "volumeType": "gp2"
              }
            },
            {
              "deviceName": "/dev/sdb",
              "ebs": {
                "volumeSize": 8,
                "deleteOnTermination": false,
                "volumeType": "gp2",
                "encrypted": true
              }
            }
          ]
        },
        "monitoring": {
          "enabled": false
        },
        "disableApiTermination": false,
        "instanceInitiatedShutdownBehavior": "stop",
        "clientToken": "XdKUT141516171819",
        "ebsOptimized": false
      },
```

```
      "responseElements": {
        "reservationId": "r-5ebc9f74",
        "ownerId": "111122223333",
        "groupSet": {
          "items": [
            {
              "groupId": "sg-98b6e0f2",
              "groupName": "launch-wizard-2"
            }
          ]
        },
        "instancesSet": {
          "items": [
            {
              "instanceId": "i-81e2f56c",
              "imageId": "ami-b66ed3de",
              "instanceState": {
                "code": 0,
                "name": "pending"
              },
              "amiLaunchIndex": 0,
              "productCodes": {

              },
              "instanceType": "m3.medium",
              "launchTime": 1415223328000,
              "placement": {
                "availabilityZone": "us-east-1a",
                "tenancy": "default"
              },
              "monitoring": {
                "state": "disabled"
              },
              "stateReason": {
                "code": "pending",
                "message": "pending"
              },
              "architecture": "x86_64",
              "rootDeviceType": "ebs",
              "rootDeviceName": "/dev/xvda",
              "blockDeviceMapping": {

              },
              "virtualizationType": "hvm",
```

```
            "hypervisor": "xen",
            "clientToken": "XdKUT1415223327917",
            "groupSet": {
              "items": [
                {
                  "groupId": "sg-98b6e0f2",
                  "groupName": "launch-wizard-2"
                }
              ]
            },
            "networkInterfaceSet": {

            },
            "ebsOptimized": false
          }
        ]
      }
    },
    "requestID": "41c4b4f7-8bce-4773-bf0e-5ae3bb5cbce2",
    "eventID": "cd75a605-2fee-4fda-b847-9c3d330ebaae",
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
  },
  {
    "eventVersion": "1.02",
    "userIdentity": {
          "type": "IAMUser",
          "principalId": "EX_PRINCIPAL_ID",
          "arn": "arn:aws:iam::111122223333:user/Alice",
          "accountId": "111122223333",
          "accessKeyId": "EXAMPLE_KEY_ID",
          "userName": "Alice"
    },
    "eventTime": "2014-11-05T21:35:35Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
      "constraints": {
        "encryptionContextSubset": {
          "aws:ebs:id": "vol-f67bafb2"
        }
```

```
        },
        "granteePrincipal": "111122223333:aws:ec2-infrastructure:i-81e2f56c",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
      },
      "responseElements": {
        "grantId": "abcde1237f76e4ba7987489ac329fbfba6ad343d6f7075dbd1ef191f0120514a"
      },
      "requestID": "41c4b4f7-8bce-4773-bf0e-5ae3bb5cbce2",
      "eventID": "c1ad79e3-0d3f-402a-b119-d5c31d7c6a6c",
      "readOnly": false,
      "resources": [
        {
          "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
          "accountId": "111122223333"
        }
      ],
      "eventType": "AwsApiCall",
      "recipientAccountId": "111122223333"
    },
    {
      "eventVersion": "1.02",
      "userIdentity": {
          "type": "IAMUser",
          "principalId": "EX_PRINCIPAL_ID",
          "arn": "arn:aws:iam::111122223333:user/Alice",
          "accountId": "111122223333",
          "accessKeyId": "EXAMPLE_KEY_ID",
          "userName": "Alice"
      },
      "eventTime": "2014-11-05T21:35:32Z",
      "eventSource": "kms.amazonaws.com",
      "eventName": "GenerateDataKeyWithoutPlaintext",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "192.0.2.0",
      "userAgent": "AWS Internal",
      "requestParameters": {
        "encryptionContext": {
          "aws:ebs:id": "vol-f67bafb2"
        },
        "numberOfBytes": 64,
        "keyId": "alias/aws/ebs"
      },
```

```
      "responseElements": null,
      "requestID": "create-111122223333-758247346-1415223332",
      "eventID": "ac3cab10-ce93-4953-9d62-0b6e5cba651d",
      "readOnly": true,
      "resources": [
        {
          "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
          "accountId": "111122223333"
        }
      ],
      "eventType": "AwsApiCall",
      "recipientAccountId": "111122223333"
    },
    {
      "eventVersion": "1.02",
      "userIdentity": {
        "type": "AssumedRole",
        "principalId": "111122223333:aws:ec2-infrastructure:i-81e2f56c",
        "arn": "arn:aws:sts::111122223333:assumed-role/aws:ec2-infrastructure/
i-81e2f56c",
        "accountId": "111122223333",
        "accessKeyId": "",
        "sessionContext": {
          "attributes": {
            "mfaAuthenticated": "false",
            "creationDate": "2014-11-05T21:35:38Z"
          },
          "sessionIssuer": {
            "type": "Role",
            "principalId": "111122223333:aws:ec2-infrastructure",
            "arn": "arn:aws:iam::111122223333:role/aws:ec2-infrastructure",
            "accountId": "111122223333",
            "userName": "aws:ec2-infrastructure"
          }
        }
      },
      "eventTime": "2014-11-05T21:35:47Z",
      "eventSource": "kms.amazonaws.com",
      "eventName": "Decrypt",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "192.0.2.0",
      "requestParameters": {
        "encryptionContext": {
```

```
          "aws:ebs:id": "vol-f67bafb2"
        }
      },
      "responseElements": null,
      "requestID": "b4b27883-6533-11e4-b4d9-751f1761e9e5",
      "eventID": "edb65380-0a3e-4123-bbc8-3d1b7cff49b0",
      "readOnly": true,
      "resources": [
        {
          "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
          "accountId": "111122223333"
        }
      ],
      "eventType": "AwsApiCall",
      "recipientAccountId": "111122223333"
    }
  ]
}
```


Amazon CloudWatch를 사용하여 KMS 키 모니터링

에서 원시 데이터를 수집하여 읽기 가능하며 실시간에 가까운 지표 AWS KMS 로 처리하는 AWS 서비스인 Amazon CloudWatch를 AWS KMS keys 사용하여를 모니터링할 수 있습니다. 이러한 데이터는 2주간 기록되므로 기록 정보를 보고 KMS 키의 사용 상황과 시간에 따른 변화를 더 잘 이해할 수 있습니다.

Amazon CloudWatch를 사용하여 다음과 같은 중요한 이벤트를 알릴 수 있습니다.

- KMS 키의 가져온 키 구성 요소의 만료 날짜가 가까워졌습니다.
- 삭제 보류 중인 KMS 키가 계속 사용되고 있습니다.
- KMS 키의 키 구성 요소가 자동으로 교체되었습니다.
- KMS 키가 삭제되었습니다.

요청 비율이 할당량 값의 일정 비율에 도달하면 경고하는 Amazon CloudWatch 경보를 생성할 수도 있습니다. 자세한 내용은 AWS 보안 블로그의 Service Quotas 및 Amazon CloudWatch를 사용하여 AWS KMS API 요청 속도 관리를 참조하세요.

AWS KMS 지표 및 차원

AWS KMS 는 Amazon CloudWatch 지표를 사전 정의하여 중요한 데이터를 더 쉽게 모니터링하고 경보를 생성할 수 있도록 합니다. AWS Management Console 및 Amazon CloudWatch API를 사용하여 AWS KMS 지표를 볼 수 있습니다.

이 섹션에서는 각 AWS KMS 지표와 각 지표의 차원을 나열하고 이러한 지표와 차원을 기반으로 CloudWatch 경보를 생성하기 위한 몇 가지 기본 지침을 제공합니다.

> **Note**
>
> 차원 그룹 이름:
> Amazon CloudWatch 콘솔에서 지표를 보려면 Metrics(지표) 섹션에서 차원 그룹 이름을 선택합니다. 그런 다음 Metric name(지표 이름)으로 필터링할 수 있습니다. 이 주제에는 각 AWS KMS 지표에 대한 지표 이름과 차원 그룹 이름이 포함됩니다.

AWS Management Console 및 Amazon CloudWatch API를 사용하여 AWS KMS 지표를 볼 수 있습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 사용 가능한 지표 보기를 참조하세요.

주제

- SuccessfulRequest
- SecondsUntilKeyMaterialExpiration
- CloudHSMKeyStoreThrottle
- ExternalKeyStoreThrottle
- XksProxyCertificateDaysToExpire
- XksProxyCredentialAge
- XksProxyErrors
- XksExternalKeyManagerStates
- XksProxyLatency

SuccessfulRequest

특정 KMS 키에 대한 암호화 작업 요청이 성공한 횟수입니다. `SuccessfulRequest` 지표를 사용하면 CloudWatch의 AWS KMS API 사용량에 키 수준 필터링을 적용할 수 있습니다. 이 지표의 `Sum` 통계는 해당 기간 동안의 성공적인 요청 수 전체를 나타냅니다.

이 지표를 사용해 요청 할당량 중 가장 큰 비중을 사용하거나 API 비용에 가장 많이 기여하는 KMS 키를 식별하세요. `SuccesfulRequest` 지표를 기반으로 CloudWatch 알람을 생성해 비정상적인 AWS KMS API 사용 패턴에 대한 알림을 받을 수 있습니다. 이러한 알림은 요청 할당량을 의도치 않게 초과하거나 예상치 못한 비용을 발생시킬 수 있는 비효율적인 워크플로를 식별하는 데 도움이 됩니다.

의 차원 **`SuccessfulRequest`**

차원	설명
KeyArn	각 KMS 키의 값입니다.
연산	각 AWS KMS API 작업의 값입니다. 이 지표는 암호화 작업에만 적용됩니다.

ReEncrypt 작업의 경우 `SuccessfulRequest` 지표에는 소스 KMS 키와 대상 KMS 키에 대한 차원이 모두 포함됩니다.

차원	설명
SourceKeyArn	사이퍼텍스트를 복호화한 KMS 키의 값입니다.
DestinationKeyArn	데이터를 다시 암호화한 KMS 키에 대한 값입니다.
연산	각 AWS KMS API 작업의 값입니다.이 경우 ReEncrypt입니다.

SecondsUntilKeyMaterialExpiration

KMS 키에서 가장 먼저 만료되는 가져온 키 구성 요소가 만료될 때까지 남은 시간(초)입니다. 이 지표는 가져온 키 구성 요소(`EXTERNAL`의 키 구성 요소 오리진)와 만료 날짜가 있는 KMS 키에만 유효합니다.

이 지표를 사용해 가장 먼저 만료되는 가져온 키 구성 요소가 만료되기까지 남은 시간을 추적하세요. 해당 시간이 사용자가 정의한 임곗값보다 작아지면 KMS 키를 계속 사용할 수 있도록 새 만료 날짜를 지정해 키 구성 요소를 다시 가져와야 합니다. `SecondsUntilKeyMaterialExpiration` 지표는 KMS 키에만 해당합니다. 이 지표를 사용하여 향후 생성할 수 있는 KMS 키를 하나 또는 그 이상 모니터링할 수는 없습니다. 이 지표를 모니터링하기 위해 CloudWatch 경보를 생성하는 방법에 대한 도움말은 가져온 키 구성 요소의 만료 여부에 대한 CloudWatch 경보 생성 섹션을 참조하세요.

이 지표에서 가장 유용한 통계는 `Minimum`으로서 지정한 통계 기간 중 모든 데이터 포인트에 남아있는 시간이 가장 적다는 것을 의미합니다. 이 지표에서 유일하게 사용되는 유효 단위는 `Seconds`입니다.

차원 그룹 이름: Per-Key Metrics(키별 지표)

`SecondsUntilKeyMaterialExpiration`의 차원

측정 기준	설명, 관련 AWS
KeyId	각 KMS 키의 값입니다.

KMS 키에 키 삭제를 예약하면 AWS KMS 가 KMS 키를 삭제하기 전에 대기 기간을 적용합니다. 현재 또는 향후에 KMS 키가 필요하지 않도록 하기 위해 대기 기간을 사용할 수 있습니다. 사람이나 애플리케이션이 대기 기간 중에 암호화 작업에서 KMS 키를 사용하려고 할 때 경고하도록 CloudWatch 경보를 구성할 수도 있습니다. 이러한 경보를 통해 알림을 받으면 KMS 키의 삭제를 취소하고 싶을 수 있습니다.

지침은 삭제 보류 중인 KMS 키 사용을 감지하는 경보 생성 섹션을 참조하세요.

CloudHSMKeyStoreThrottle

가 AWS KMS 제한하는 각 키 스토어의 KMS AWS CloudHSM 키에 대한 암호화 작업 요청 수입니다 (로 응답`ThrottlingException`). 이 지표는 AWS CloudHSM 키 스토어에만 적용됩니다.

지표는 키 스토어의 AWS CloudHSM KMS 키와 암호화 작업에 대한 요청에만 적용됩니다. `CloudHSMKeyStoreThrottle`는 요청 속도가 키 스토어에 대한 사용자 지정 키 스토어 요청 할당량을 초과할 때 AWS KMS 이러한 요청을 제한합니다 AWS CloudHSM . 이 지표에는 AWS CloudHSM 클러스터의 제한도 포함됩니다.

차원 그룹 이름: Keystore Throttle Metrics(키 스토어 제한 지표)

차원	설명
CustomKeyStoreId	각 AWS CloudHSM 키 스토어의 값입니다.
KmsOperation	각 AWS KMS API 작업의 값입니다. 이 지표는 AWS CloudHSM 키 저장소에 있는 KMS 키에 대한 암호화 작업에만 적용됩니다.

차원	설명
KeySpec	각 KMS 키 유형의 값입니다. 키 스토어에서 KMS 키에 대해 지원되는 유일한 키 사양은 SYMMETRIC_DEFAULT입니다. AWS CloudHSM

ExternalKeyStoreThrottle

가 AWS KMS 제한하는 각 외부 키 스토어의 KMS 키에 대한 암호화 작업 요청 수입니다(로 응답`ThrottlingException`). 이 지표는 외부 키 스토어에만 적용됩니다.

지표는 외부 키 스토어의 KMS 키와 암호화 작업에 대한 요청에만 적용됩니다. `ExternalKeyStoreThrottle`는 요청 속도가 외부 키 스토어에 대한 사용자 지정 키 스토어 요청 할당량을 초과할 때 AWS KMS 이러한 요청을 제한합니다. 이 지표에는 외부 키 스토어 프록시 또는 외부 키 관리자에 의한 제한이 포함되지 않습니다.

이 지표를 사용하여 사용자 지정 키 저장소 요청 할당량의 값을 검토하고 조정하세요. 이 지표에서 AWS KMS 가 이러한 KMS 키에 대한 요청을 자주 제한하고 있음을 나타내는 경우 사용자 지정 키 스토어 요청 할당량 값 증가를 요청하는 것이 좋습니다. 도움이 필요한 경우 Service Quotas 사용 설명서의 할당량 증가 요청을 참조하세요.

'매우 높은 요청 빈도로 인해' 또는 '외부 키 스토어 프록시가 제때 응답하지 않아' 요청이 거부되었음을 설명하는 메시지와 함께 `KMSInvalidStateException` 오류가 매우 자주 발생하는 경우 외부 키 관리자 또는 외부 키 스토어 프록시가 현재 요청 빈도를 따라갈 수 없는 것일 수 있습니다. 가능하면 요청 빈도를 낮춥니다. 사용자 지정 키 스토어 요청 할당량 값의 감소를 요청하는 것도 고려할 수 있습니다. 이 할당량 값을 줄이면 제한(및 `ExternalKeyStoreThrottle` 지표 값)이 증가할 수 있지만 외부 키 스토어 프록시 또는 외부 키 관리자로 전송되기 전에 초과 요청을 빠르게 AWS KMS 거부하고 있음을 나타냅니다. 할당량 감소를 요청하기 위해서는 AWS Support 센터를 방문하여 케이스를 생성하세요.

차원 그룹 이름: Keystore Throttle Metrics(키 스토어 제한 지표)

차원	설명
CustomKeyStoreId	각 외부 키 스토어의 값입니다.
KmsOperation	각 AWS KMS API 작업의 값입니다. 이 지표는 외부 키 저장소에 있는 KMS 키에 대한 암호화 작업에만 적용됩니다.

차원	설명
KeySpec	각 KMS 키 유형의 값입니다. 외부 키 스토어에서 KMS 키에 대해 지원되는 유일한 키 사양은 SYMMETRIC_DEFAULT입니다.

XksProxyCertificateDaysToExpire

외부 키 스토어 프록시 엔드포인트(`XksProxyUriEndpoint`)에 대한 TLS 인증서가 만료될 때까지의 일수입니다. 이 지표는 외부 키 스토어에만 적용됩니다.

TLS 인증서의 예정된 만료를 알리는 CloudWatch 경보를 생성하려면 이 지표를 사용합니다. 인증서가 만료되면는 외부 키 스토어 프록시와 통신할 AWS KMS 수 없습니다. 외부 키 스토어의 KMS 키로 보호되는 모든 데이터는 인증서를 갱신할 때까지 액세스할 수 없습니다.

인증서 경보는 암호화된 리소스에 액세스하지 못하게 할 수 있는 인증서 만료를 방지합니다. 인증서가 만료되기 전에 인증서를 갱신할 시간을 조직에 제공하도록 경보를 설정합니다.

차원 그룹 이름: XKS Proxy Certificate Metrics(XKS 프록시 인증서 지표)

차원	설명
CustomKeyStoreId	각 외부 키 스토어의 값입니다.
CertificateName	TLS 인증서의 주체 이름(CN)입니다.

외부 키 스토어와 외부 키 스토어의 KMS 키에 대한 지표를 기반으로 CloudWatch 경보를 생성할 수 있습니다. 지침은 외부 키 저장소 모니터링 섹션을 참조하세요.

XksProxyCredentialAge

현재 외부 키 스토어 프록시 인증 자격 증명(`XksProxyAuthenticationCredential`)이 외부 키 스토어와 연결된 이후의 일수입니다. 이 수는 외부 키 스토어를 만들거나 업데이트하는 과정에서 인증 자격 증명을 입력할 때 시작됩니다. 이 지표는 외부 키 스토어에만 적용됩니다.

이 값은 인증 자격 증명의 사용 기간을 알리도록 설계되었습니다. 그러나 외부 키 스토어 프록시에 인증 자격 증명을 생성할 때가 아니라 외부 키 스토어에 자격 증명을 연결할 때 계산을 시작하기 때문에 프록시의 자격 증명 사용 기간에 대한 정확한 지표가 아닐 수 있습니다.

이 지표를 사용하여 외부 키 스토어 프록시 인증 자격 증명을 교체하라고 알려주는 CloudWatch 경보를 생성합니다.

차원 그룹 이름: Per-Keystore Metrics(키 스토어별 지표)

차원	설명
CustomKeyStoreId	각 외부 키 스토어의 값입니다.

외부 키 스토어와 외부 키 스토어의 KMS 키에 대한 지표를 기반으로 CloudWatch 경보를 생성할 수 있습니다. 지침은 외부 키 저장소 모니터링 섹션을 참조하세요.

XksProxyErrors

외부 키 스토어 프록시에 대한 AWS KMS 요청과 관련된 예외 수입니다. 이 수에는 외부 키 스토어 프록시가 로 반환하는 예외 AWS KMS 와 외부 키 스토어 프록시가 250밀리초 제한 시간 간격 AWS KMS 내에 응답하지 않을 때 발생하는 제한 시간 오류가 포함됩니다. 이 지표는 외부 키 스토어에만 적용됩니다.

이 지표를 사용하여 외부 키 스토어에서 KMS 키의 오류율을 추적합니다. 이 지표는 가장 자주 발생하는 오류를 표시하므로 엔지니어링 작업의 우선 순위를 지정할 수 있습니다. KMS 키에서 생성하는 재시도할 수 없는 오류율이 높으면 외부 키 스토어의 구성에 문제가 있는 것일 수 있습니다. 외부 키 스토어 구성을 보려면 외부 키 저장소 보기 섹션을 참조하세요. 외부 키 스토어 설정을 편집하려면 외부 키 저장소 속성 편집 섹션을 참조하세요.

차원 그룹 이름: XKS Proxy Error Metrics(XKS 프록시 오류 지표)

차원	설명
CustomKeyStoreId	각 외부 키 스토어의 값입니다.
KmsOperation	XKS 프록시에 대한 요청을 생성한 각 AWS KMS API 작업의 값입니다.

차원	설명
XksOperation	각 외부 키 스토어 프록시 API 작업의 값입니다.
KeySpec	각 KMS 키 유형의 값입니다. 외부 키 스토어에서 KMS 키에 대해 지원되는 유일한 키 사양은 SYMMETRIC_DEFAULT입니다.
ErrorType	값: • 재시도할 수 있는 오류: 네트워킹 오류와 같이 일시적일 수 있습니다. • 재시도할 수 없는 오류: 사용자 지정 키 스토어 구성 또는 외부 구성 요소에 문제가 있는 것일 수 있습니다. • 해당 사항 없음: 성공한 요청, 오류 없음
Exception Name	값: • 예외 이름 • 없음: 성공한 요청, 오류 없음

외부 키 스토어와 외부 키 스토어의 KMS 키에 대한 지표를 기반으로 CloudWatch 경보를 생성할 수 있습니다. 지침은 외부 키 저장소 모니터링 섹션을 참조하세요.

XksExternalKeyManagerStates

각 상태(`Active`, `Degraded` 및 `Unavailable`)의 외부 키 관리자 인스턴스 수입니다. 이 지표에 대한 정보는 각 외부 키 스토어와 연결된 외부 키 스토어 프록시에서 가져옵니다. 이 지표는 외부 키 스토어에만 적용됩니다.

다음은 외부 키 스토어와 연결된 외부 키 관리자 인스턴스의 상태입니다. 각 외부 키 스토어 프록시는 서로 다른 표시기를 사용하여 외부 키 관리자의 상태를 측정할 수 있습니다. 자세한 내용은 외부 키 스토어 프록시의 설명서를 참조하세요.

- `Active`: 외부 키 관리자가 정상입니다.
- `Degraded`: 외부 키 관리자가 비정상이지만 여전히 트래픽을 처리할 수 있습니다.
- `Unavailable`: 외부 키 관리자가 트래픽을 처리할 수 없습니다.

이 지표를 사용하여 성능이 저하되고 사용할 수 없는 외부 키 관리자 인스턴스에 대해 경고하는 CloudWatch 경보를 생성합니다. 각 상태의 외부 키 관리자 인스턴스를 확인하려면 외부 키 스토어 프록시 로그를 참조하세요.

차원 그룹 이름: XKS External Key Manager Metrics(XKS 외부 키 관리자 지표)

차원	설명
CustomKeyStoreId	각 외부 키 스토어의 값입니다.
XksExternalKeyManagerState	각 상태에 대한 값입니다.

외부 키 스토어와 외부 키 스토어의 KMS 키에 대한 지표를 기반으로 CloudWatch 경보를 생성할 수 있습니다. 지침은 외부 키 저장소 모니터링 섹션을 참조하세요.

XksProxyLatency

외부 키 스토어 프록시가 AWS KMS 요청에 응답하는 데 걸리는 시간(밀리초)입니다. 요청 시간이 초과된 경우 기록된 값은 250밀리초 제한 시간입니다. 이 지표는 외부 키 스토어에만 적용됩니다.

이 지표를 사용하여 외부 키 스토어 프록시와 외부 키 관리자의 성능을 평가합니다. 예를 들어, 프록시의 암호화 및 복호화 작업 시간이 자주 초과되는 경우 외부 프록시 관리자에게 문의하세요.

느린 응답은 외부 키 관리자가 현재 요청 트래픽을 처리할 수 없음을 나타낼 수도 있습니다. AWS KMS는 외부 키 관리자가 초당 최대 1,800개의 암호화 작업 요청을 처리할 수 있도록 권장합니다. 외부 키 관리자가 초당 1,800개의 요청을 처리할 수 없는 경우 사용자 지정 키 스토어에서 KMS 키에 대한 요청 할당량 감소를 요청하는 것이 좋습니다. 외부 키 스토어에서 KMS 키를 사용하는 암호화 작업에 대한 요청은 외부 키 스토어 프록시 또는 외부 키 관리자에 의해 처리되고 나중에 거부되는 대신 제한 예외와 함께 빠르게 실패합니다.

차원 그룹 이름: XKS Proxy Latency Metrics(XKS 프록시 지연 시간 지표)

차원	설명
CustomKeyStoreId	각 외부 키 스토어의 값입니다.

차원	설명
KmsOperation	XKS 프록시에 대한 요청을 생성한 각 AWS KMS API 작업의 값입니다.
XksOperation	각 외부 키 스토어 프록시 API 작업의 값입니다.
KeySpec	각 KMS 키 유형의 값입니다. 외부 키 스토어에서 KMS 키에 대해 지원되는 유일한 키 사양은 SYMMETRIC_DEFAULT입니다.

외부 키 스토어와 외부 키 스토어의 KMS 키에 대한 지표를 기반으로 CloudWatch 경보를 생성할 수 있습니다. 지침은 외부 키 저장소 모니터링 단원을 참조하세요.

가져온 키 구성 요소의 만료 여부에 대한 CloudWatch 경보 생성

가져온 키 구성 요소의 만료일이 가까워지면 이를 알리는 CloudWatch 경보를 생성할 수 있습니다. 예를 들어 만료까지 남은 기간이 30일 미만인 경우 이 경보를 통해 알림을 받을 수 있습니다.

키 구성 요소를 KMS 키로 가져올 때 선택적으로 키 구성 요소의 만료 날짜 및 시간을 지정할 수 있습니다. 키 구성 요소가 만료되면는 키 구성 요소를 AWS KMS 삭제하고 KMS 키를 사용할 수 없게 됩니다. KMS 키를 다시 사용하려면 키 구성 요소를 다시 가져와야 합니다. 그러나 만료되기 전에 키 구성 요소를 다시 가져오면 해당 KMS 키를 사용하는 프로세스가 중단되는 상황을 방지할 수 있습니다.

이 경보는 만료되는 가져온 키 구성 요소가 있는 KMS 키에 대해 CloudWatch에 AWS KMS 게시하는 `SecondsUntilKeyMaterialExpires` 지표를 사용합니다. 각 경보는 이 지표를 사용하여 특정 KMS 키에 대해 가져온 키 구성 요소를 모니터링합니다. 만료되는 키 구성 요소가 있는 모든 KMS 키에 대해 단일 경보를 생성할 수 없으며 향후 생성할 수 있는 KMS 키에 대한 경보를 생성할 수는 없습니다.

요구 사항

가져온 키 구성 요소의 만료 여부를 모니터링하는 CloudWatch 경보에는 다음 리소스가 필요합니다.

- 만료되는 가져온 키 구성 요소가 있는 KMS 키.
- Amazon SNS 주제. 자세한 내용은 Amazon CloudWatch 사용 설명서의 Amazon SNS 주제 생성을 참조하세요.

경보 생성

다음 필수 값을 사용하여 정적 임계값을 기반으로 CloudWatch 경보 생성의 지침을 따르세요. 다른 필드의 경우 기본값을 그대로 사용하고 요청에 따라 이름을 제공합니다.

Field	값
지표 선택	KMS를 선택한 다음 키별 지표를 선택합니다. KMS 키가 있는 행과 `SecondsUntilKeyMaterialExpires` 지표를 선택합니다. 그런 다음 지표 선택을 선택합니다. 지표 목록에는 만료되는 가져온 키 구성 요소가 있는 KMS 키에 대한 `SecondsUntilKeyMaterialExpires` 지표만 표시됩니다. 계정 및 리전에 이러한 속성을 가진 KMS 키가 없는 경우 이 목록은 비어 있습니다.
통계	최소
Period	1분
임계값 유형	정적
다음과 같은 경우 항상 …	metric-name이 1 ## ##마다

외부 키 저장소에 대한 CloudWatch 경보 생성

외부 키 저장소 지표를 기반으로 Amazon CloudWatch 경보를 생성하여 지표 값이 지정한 임계값을 초과할 때 알려줄 수 있습니다. 이 경보는 메시지를 Amazon Simple Notification Service(Amazon SNS) 주제 또는 Amazon EC2 Auto Scaling 정책에 전송할 수 있습니다. CloudWatch 경보에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서의 Amazon CloudWatch 경보 사용을 참조하세요.

Amazon CloudWatch 경보를 생성하기 전에 Amazon SNS 주제가 필요합니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 Amazon SNS 주제 생성을 참조하세요.

주제

- 인증서 만료에 대한 경보 생성
- 응답 제한 시간에 대한 경보 생성
- 재시도 가능한 오류에 대한 경보 생성
- 재시도 불가능한 오류에 대한 경보 생성

인증서 만료에 대한 경보 생성

이 경보는 AWS KMS 가 CloudWatch에 게시하는 XksProxyCertificateDaysToExpire 지표를 사용하여 외부 키 스토어 프록시 엔드포인트와 연결된 TLS 인증서의 예상 만료를 기록합니다. 계정의 모든 외부 키 스토어에 대한 단일 경보나 향후 생성할 수 있는 외부 키 스토어에 대한 경보를 생성할 수 없습니다.

인증서가 만료되기 10일 전에 알리도록 경보를 설정하는 것이 좋지만 필요에 가장 맞는 임계값을 설정해야 합니다.

경보 생성

다음 필수 값을 사용하여 정적 임계값을 기반으로 CloudWatch 경보 생성의 지침을 따르세요. 다른 필드의 경우 기본값을 그대로 사용하고 요청에 따라 이름을 제공합니다.

Field	값
지표 선택	KMS를 선택한 다음 XKS Proxy Certificate Metrics(XKS 프록시 인증서 지표)를 선택합니다. 모니터링할 `XksProxyCertificateName` 옆의 확인란을 선택합니다. 그런 다음 지표 선택을 선택합니다.
통계	최소
Period	5분
임계값 유형	정적
다음과 같은 경우 항상 …	XksProxyCertificateDaysToExpire가 `10`보다 `Lower`일 때마다

응답 제한 시간에 대한 경보 생성

이 경보는 CloudWatch에 AWS KMS 게시하는 지표를 사용하여 외부 키 스토어 프록시가 AWS KMS 요청에 응답하는 데 걸리는 XksProxyLatency 밀리초 수를 기록합니다. 계정의 모든 외부 키 스토어에 대한 단일 경보나 향후 생성할 수 있는 외부 키 스토어에 대한 경보를 생성할 수 없습니다.

AWS KMS 는 외부 키 스토어 프록시가 250밀리초 이내에 각 요청에 응답할 것으로 예상합니다. 외부 키 스토어 프록시가 응답하는 데 200밀리초 이상 걸리는 경우 알리도록 경보를 설정하는 것이 좋지만 필요에 가장 맞는 임계값을 설정해야 합니다.

경보 생성

다음 필수 값을 사용하여 정적 임계값을 기반으로 CloudWatch 경보 생성의 지침을 따르세요. 다른 필드의 경우 기본값을 그대로 사용하고 요청에 따라 이름을 제공합니다.

Field	값
지표 선택	KMS를 선택한 다음 XKS Proxy Latency Metrics(XKS 프록시 지연 시간 지표)를 선택합니다. 모니터링할 `KmsOperation` 옆의 확인란을 선택합니다. 그런 다음 지표 선택을 선택합니다.
통계	평균
Period	5분
임계값 유형	정적
다음과 같은 경우 항상 …	XksProxyLatency가 `200`보다 `Greater`일 때마다

재시도 가능한 오류에 대한 경보 생성

이 경보는 CloudWatch에 AWS KMS 게시하는 XksProxyErrors 지표를 사용하여 외부 키 스토어 프록시에 대한 AWS KMS 요청과 관련된 예외 수를 기록합니다. 계정의 모든 외부 키 스토어에 대한 단일 경보나 향후 생성할 수 있는 외부 키 스토어에 대한 경보를 생성할 수 없습니다.

재시도할 수 있는 오류는 신뢰성(백분율)을 낮추고 네트워킹 오류를 나타낼 수 있습니다. 1분 동안 재시도할 수 있는 오류가 5개 이상 기록되면 알리도록 경보를 설정하는 것이 좋지만 필요에 가장 맞는 임계값을 설정해야 합니다.

다음 필수 값을 사용하여 정적 임계값을 기반으로 CloudWatch 경보 생성의 지침을 따르세요. 다른 필드의 경우 기본값을 그대로 사용하고 요청에 따라 이름을 제공합니다.

Field	값
지표 선택	쿼리(Query) 탭을 선택합니다. Namespace(네임스페이스)에서 AWS/KMS를 선택합니다. Metric name(지표 이름)에 SUM(XksProxyErrors) 을 입력합니다. Filter by(필터링 기준)에 ErrorType = Retryable 을 입력합니다. 실행을 선택합니다. 그런 다음 지표 선택을 선택합니다.
Label	#### # ## ##
Period	1분
임계값 유형	정적
다음과 같은 경우 항상 ...	q1이 5보다 Greater일 때마다

재시도 불가능한 오류에 대한 경보 생성

이 경보는 CloudWatch에 AWS KMS 게시하는 XksProxyErrors 지표를 사용하여 외부 키 스토어 프록시에 대한 AWS KMS 요청과 관련된 예외 수를 기록합니다. 계정의 모든 외부 키 스토어에 대한 단일 경보나 향후 생성할 수 있는 외부 키 스토어에 대한 경보를 생성할 수 없습니다.

재시도할 수 없는 오류는 외부 키 스토어 구성에 문제가 있음을 나타낼 수 있습니다. 1분 동안 재시도할 수 없는 오류가 5개 이상 기록되면 알리도록 경보를 설정하는 것이 좋지만 필요에 가장 맞는 임계값을 설정해야 합니다.

다음 필수 값을 사용하여 정적 임계값을 기반으로 CloudWatch 경보 생성의 지침을 따르세요. 다른 필드의 경우 기본값을 그대로 사용하고 요청에 따라 이름을 제공합니다.

Field	값
지표 선택	쿼리(Query) 탭을 선택합니다. Namespace(네임스페이스)에서 AWS/KMS를 선택합니다.

Field	값
	Metric name(지표 이름)에 `SUM(XksProxyErrors)` 을 입력합니다. Filter by(필터링 기준)에 `ErrorType = Non-retryable` 을 입력합니다. 실행을 선택합니다. 그런 다음 지표 선택을 선택합니다.
Label	`#### # ## ##`
Period	1분
임계값 유형	정적
다음과 같은 경우 항상 …	q1이 5보다 `Greater`일 때마다

KMS 키의 과거 사용 결정

모니터링 및 감사를 위해 KMS 키가 과거에 어떻게 사용되었는지 알고 싶을 수 있습니다. 예를 들어 삭제를 위해 KMS 키를 비활성화하거나 예약하기 전에 KMS 키가 여전히 활성 상태로 사용되는지 확인하거나 계정에서 미사용 키를 식별할 수 있습니다. 다음 전략은 KMS 키의 과거 사용량을 결정하는 데 도움이 될 수 있습니다.

> **Warning**
>
> 과거 사용량을 결정하기 위한 이러한 전략은 AWS 보안 주체 및 AWS KMS 운영에만 유효합니다. 에 대한 API 호출을 포함하지 않는 키의 사용을 감지할 수 없습니다 AWS KMS. 대칭 KMS 키에서 데이터 키가 생성된 후 외부에서 로컬 암호화 또는 복호화에 대한 후속 사용은 마지막 사용 정보에 반영 AWS KMS 되지 않습니다. 마찬가지로 이러한 전략은 외부에서 비대칭 KMS 키의 퍼블릭 키 사용을 감지할 수 없습니다 AWS KMS. 암호를 해독할 수 없는 암호화 텍스트를 만드는 등 퍼블릭 키 암호화에 사용되는 비대칭 KMS 키를 삭제할 때의 특별한 위험에 대한 자세한 내용은 Deleting asymmetric KMS keys 단원을 참조하십시오.

주제

- KMS 키 권한 확인을 통해 사용 가능 범위 파악
- KMS 키로 수행된 마지막 암호화 작업 검사

- AWS CloudTrail 로그를 검사하여 과거 사용량 확인

KMS 키 권한 확인을 통해 사용 가능 범위 파악

현재 KMS 키에 액세스할 수 있는 사용자 또는 대상을 결정하면 KMS 키가 얼마나 널리 사용되고 있는지, 여전히 필요한지 판단하는 데 도움이 될 수 있습니다. 현재 KMS 키에 액세스할 수 있는 사용자 또는 대상을 확인하려면 섹션을 참조하세요에 대한 액세스 결정 AWS KMS keys.

KMS 키로 수행된 마지막 암호화 작업 검사

AWS KMS 는 연결된 CloudTrail 이벤트 ID와 함께 각 KMS 키로 수행된 마지막으로 성공한 암호화 작업에 대한 사용 정보를 제공합니다. 이렇게 하면 사용되지 않는 KMS 키를 쉽게 식별할 수 있습니다. 또한 키 정책에서 kms:TrailingDaysWithoutKeyUsage 조건 키를 사용하여 최근에 사용한 키가 실수로 비활성화되거나 삭제 예약되지 않도록 할 수 있습니다.

AWS Management Console AWS CLI또는 AWS KMS API를 사용하여 KMS 키로 마지막으로 성공한 암호화 작업을 볼 수 있습니다.

> **Note**
>
> 특정 AWS 서비스는 데이터 보호를 위해 KMS 키에 의존하는 리소스를 생성하지만 해당 키에 대한 암호화 작업은 자주 호출하지 않습니다. 예를 들어 Amazon EC2 서비스는 볼륨이 인스턴스에 연결된 경우에만를 호출 AWS KMS 하여 암호화된 Amazon EBS 볼륨의 데이터 키를 해독합니다. 이러한 경우 KMS 키를 삭제할 수 있는지 확인하기 위해 마지막 사용 정보에만 의존해서는 안 됩니다. Amazon EBS 볼륨을 보호하는 KMS 키가 삭제되면 이미 연결된 Amazon EBS 볼륨이 중단되지 않지만 암호화된 Amazon EBS 볼륨을 다른 Amazon EC2 인스턴스에 연결하려는 후속 시도는 실패합니다.

사용량 추적 기간 이해

AWS KMS 는 각 KMS 키로 수행된 마지막으로 성공한 암호화 작업만 추적합니다. 암호화 작업이 발생하는 시간과 사용량이 기록되는 시간 사이에 최대 1시간의 지연이 있을 수 있습니다.

KMS 키의 마지막 사용 정보를 확인하면 응답에 추적 시작 날짜가 포함됩니다. 는가 해당 키에 대한 암호화 활동을 기록하기 AWS KMS 시작한 날짜`TrackingStartDate`입니다. 이 날짜를 키 생성 날짜와 함께 사용하여 키 생성 날짜와 추적 시작 날짜를 비교하여 사용 기록을 확인합니다.

- 마지막 사용 정보가 있는 경우 추적이 시작된 이후 암호화 작업에 키가 사용된 것입니다. 응답에는 작업 유형, 타임스탬프 및 관련 AWS CloudTrail 이벤트 ID가 포함됩니다.
- 마지막 사용 정보가 비어 있으면 추적이 시작된 이후 키에 기록된 암호화 작업이 없는 것입니다. 키의 생성 날짜를와 비교하여 이것이 무엇을 의미하는지 `TrackingStartDate` 확인합니다.
  - 에서 또는 이후에 키가 생성된 경우 `TrackingStartDate` 키는 생성된 이후 암호화 작업에 사용되지 않습니다.
  - 이전에 키가 생성된 경우 추적`TrackingStartDate`이 시작된 이후 사용 중인 키에 대한 레코드가 없습니다. 그러나 추적이 시작되기 전에 키가 사용되었을 수 있습니다. 키가 이전에 사용되었는지 확인하려면 과거 AWS CloudTrail 로그를 검사합니다.

> **Warning**
>
> 사용하지 않는 키를 삭제할 때 마지막 사용 정보에만 의존하지 마십시오. 대신 먼저 키를 비활성화하고 비활성화된 상태에서 키를 사용하려는 시도를 나타내는 `DisabledException` 항목이 있는지 모니터링합니다 AWS CloudTrail . 이렇게 하면 잠재적 종속성과 워크로드 장애를 식별하는 데 도움이 됩니다.는 키에 대한 모든 API 호출의 신뢰할 수 있는 소스로 AWS CloudTrail 남아 있습니다.

추적된 암호화 작업

마지막 사용 정보를 보고하기 위해 다음과 같은 성공적인 암호화 작업만 추적되고 기록됩니다. 비암호화 작업은 제외됩니다.

- `Decrypt`
- `DeriveSharedSecret`
- `Encrypt`
- `GenerateDataKey`
- `GenerateDataKeyPair`
- `GenerateDataKeyPairWithoutPlaintext`
- `GenerateDataKeyWithoutPlaintext`
- `GenerateMac`
- `ReEncrypt`
- `Sign`

- `Verify`
- `VerifyMac`

기타 고려 사항

사용 정보를 사용할 때는 다음 사항에 유의하세요.

- 다중 리전 KMS 키 - 다중 리전 KMS 키의 경우 기본 키와 복제본 키는 마지막 사용 정보를 독립적으로 추적합니다. 다중 리전 키 세트의 각 키는 자체 마지막 사용 정보를 유지합니다.
- ReEncrypt 작업 - `ReEncrypt` 작업은 복호화를 위한 소스 키와 암호화를 위한 대상 키라는 두 가지 키를 사용합니다. 마지막 사용 정보는 각 키 소유자 계정의 CloudTrail 이벤트 ID와 함께 두 키에 대해 독립적으로 기록됩니다.

다음 방법을 사용하여 마지막 사용 정보를 볼 수 있습니다.

AWS KMS 콘솔 사용

각 KMS 키의 세부 정보 페이지에서 KMS 키로 마지막으로 성공한 암호화 작업을 볼 수 있습니다. KMS 키의 세부 정보 페이지를 보는 방법에 대한 절차는 섹션을 참조하세요KMS 키 세부 정보 액세스 및 나열.

AWS KMS API 사용

GetKeyLastUsage 작업은 지정된 KMS 키로 수행된 마지막 암호화 작업에 대한 사용 정보를 반환합니다. KMS 키를 식별하려면 키 ID 또는 키 ARN을 사용합니다.

예를 들어에 대한 다음 호출은 키 ID가 *1234abcd-12ab-34cd-56ef-1234567890ab*인 KMS 키에 대한 사용 정보를 `GetKeyLastUsage` 검색합니다.

```
$ aws kms get-key-last-usage --key-id "1234abcd-12ab-34cd-56ef-1234567890ab"
{
    "KeyCreationDate": 1773253425.56,
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "TrackingStartDate": 1773253425.56,
    "KeyLastUsage": {
        "Timestamp": 1773253497.0,
        "Operation": "Encrypt",
        "KmsRequestId": "040cce3e-9ef3-4651-b8cf-e47c9bafdc9b",
        "CloudTrailEventId": "2cfd5892-ea8c-4342-ad49-4b9594b06a8b"
    }
```

```
}
```

반대로를 호출하면 키 ID가 *0987dcba-09fe-87dc-65ba-ab0987654321*인 KMS 키에 대한 사용 정보가 GetKeyLastUsage 표시되지 않습니다.

```
$ aws kms get-key-last-usage --key-id "0987dcba-09fe-87dc-65ba-ab0987654321"
{
    "KeyCreationDate": 1672531200.0,
    "KeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
    "TrackingStartDate": 1773253425.56,
    "KeyLastUsage": {}
}
```

AWS CloudTrail 로그를 검사하여 과거 사용량 확인

KMS 키의 사용 기록을 사용하여 특정 KMS 키로 암호화된 사이퍼텍스트가 있는지 확인할 수 있습니다.

모든 AWS KMS API 활동은 AWS CloudTrail 로그 파일에 기록됩니다. KMS 키가 위치한 리전에서 CloudTrail 추적을 생성한 경우 CloudTrail 로그 파일을 검사하여 특정 KMS 키에 대한 모든 AWS KMS API 활동의 기록을 볼 수 있습니다. 추적이 없는 경우에도 CloudTrail 이벤트 기록에서 최근 이벤트를 볼 수 있습니다. 에서 CloudTrail을 AWS KMS 사용하는 방법에 대한 자세한 내용은 섹션을 참조하세요를 사용하여 AWS KMS API 호출 로깅 AWS CloudTrail.

다음 예에서는 Amazon Simple Storage Service(Amazon S3)에 저장된 객체를 보호하는 용도로 KMS 키를 사용할 때 생성되는 CloudTrail 로그 항목을 보여 줍니다. 이 예에서 객체는 KMS 키를 사용한 서버 측 암호화(SSE-KMS) 데이터 보호 기능을 사용하여 Amazon S3에 업로드됩니다. SSE-KMS를 사용하여 Amazon S3에 객체를 업로드할 때 객체를 보호하는 데 사용할 KMS 키를 지정합니다. Amazon S3는 작업을 사용하여 객체에 AWS KMS GenerateDataKey 대한 고유한 데이터 키를 요청하며,이 요청 이벤트는 다음과 유사한 항목과 함께 CloudTrail에 로깅됩니다.

```
{
  "eventVersion": "1.02",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROACKCEVSQ6C2EXAMPLE:example-user",
    "arn": "arn:aws:sts::111122223333:assumed-role/Admins/example-user",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "attributes": {
```

```
        "mfaAuthenticated": "false",
        "creationDate": "2015-09-10T23:12:48Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::111122223333:role/Admins",
        "accountId": "111122223333",
        "userName": "Admins"
      }
    },
    "invokedBy": "internal.amazonaws.com"
  },
  "eventTime": "2015-09-10T23:58:18Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "GenerateDataKey",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "internal.amazonaws.com",
  "userAgent": "internal.amazonaws.com",
  "requestParameters": {
    "encryptionContext": {"aws:s3:arn": "arn:aws:s3:::example_bucket/example_object"},
    "keySpec": "AES_256",
    "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  },
  "responseElements": null,
  "requestID": "cea04450-5817-11e5-85aa-97ce46071236",
  "eventID": "80721262-21a5-49b9-8b63-28740e7ce9c9",
  "readOnly": true,
  "resources": [{
    "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "accountId": "111122223333"
  }],
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}
```

나중에 Amazon S3에서이 객체를 다운로드하면 Amazon S3는 지정된 KMS 키를 사용하여 객체의 데이터 키를 해독 AWS KMS 하라는 `Decrypt` 요청을에 보냅니다. 이렇게 하면 CloudTrail 로그 파일에 다음과 비슷한 항목이 포함됩니다.

```
{
```

```
  "eventVersion": "1.02",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROACKCEVSQ6C2EXAMPLE:example-user",
    "arn": "arn:aws:sts::111122223333:assumed-role/Admins/example-user",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2015-09-10T23:12:48Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::111122223333:role/Admins",
        "accountId": "111122223333",
        "userName": "Admins"
      }
    },
    "invokedBy": "internal.amazonaws.com"
  },
  "eventTime": "2015-09-10T23:58:39Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "Decrypt",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "internal.amazonaws.com",
  "userAgent": "internal.amazonaws.com",
  "requestParameters": {
    "encryptionContext": {"aws:s3:arn": "arn:aws:s3:::example_bucket/example_object"}},
  "responseElements": null,
  "requestID": "db750745-5817-11e5-93a6-5b87e27d91a0",
  "eventID": "ae551b19-8a09-4cfc-a249-205ddba330e3",
  "readOnly": true,
  "resources": [{
    "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "accountId": "111122223333"
  }],
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}
```

이러한 로그 항목을 평가하여 특정 KMS 키의 과거 사용량을 파악할 수 있고, 이렇게 하면 해당 항목을 삭제할지 말지 결정하는 데 도움이 될 수 있습니다.

CloudTrail 로그 파일에 AWS KMS API 활동이 표시되는 방법에 대한 추가 예제를 보려면 단원을 참조하십시오를 사용하여 AWS KMS API 호출 로깅 AWS CloudTrail. CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하십시오.

CloudTrail 쿼리에 대한 비용 고려 사항

Amazon CloudWatch Logs Insights를 사용하여 CloudTrail 로그를 쿼리하면 Amazon CloudWatch Logs Insights는 스캔한 데이터의 양을 기준으로 요금을 청구합니다. 제한 없는 대규모 쿼리는 예기치 않게 비용이 많이 들 수 있습니다. 비용을 관리하려면 모든 쿼리의 범위를 가장 좁은 관련 시간 범위로 지정합니다. 이벤트 이름, 리소스 ARN 또는 보안 주체 ID를 기준으로 필터링합니다. 쿼리를 시도하기 전에 먼저 KMS 키로 마지막으로 성공한 암호화 작업을 검사하여 불필요한 쿼리 비용이 발생하지 않도록 하는 것이 좋습니다. AWS 예산에서 예산 알림을 설정하여 예상치 못한 쿼리 볼륨을 포착할 수도 있습니다. 요금에 대한 자세한 내용은 Amazon Web Services 웹 사이트의 Amazon CloudWatch 요금을 참조하세요.

Amazon EventBridge를 사용하여 KMS 키 모니터링

Amazon EventBridge(이전의 Amazon CloudWatch Events)를 사용하여 KMS 키 수명 주기에서 다음과 같은 중요한 이벤트를 알릴 수 있습니다.

- KMS 키의 키 구성 요소는 자동 또는 온디맨드로 교체되었습니다.
- KMS 키에 가져온 키 구성 요소가 만료되었습니다.
- 삭제가 예약된 KMS 키가 삭제되었습니다.

AWS KMS 는 Amazon EventBridge와 통합되어 KMS 키에 영향을 미치는 중요한 이벤트를 알려줍니다. 각 이벤트는 JSON(JavaScript Object Notation)으로 표시되며, 이벤트 이름, 이벤트가 발생한 날짜와 시간, 영향받는 항목 등을 포함합니다. 이러한 이벤트를 수집하고 AWS Lambda 함수, Amazon SNS 주제, Amazon SQS 대기열, Amazon Kinesis Data Streams의 스트림 또는 기본 제공 대상과 같은 하나 이상의 대상으로 라우팅하는 규칙을 설정할 수 있습니다.

읽기/쓰기 API 요청을 기록할 AWS CloudTrail 때에서 내보내는 이벤트를 포함하여 다른 종류의 이벤트와 함께 EventBridge를 사용하는 방법에 대한 자세한 내용은 Amazon EventBridge 사용 설명서를 참조하세요.

다음 주제에서는가 AWS KMS 생성하는 EventBridge 이벤트에 대해 설명합니다.

KMS CMK 로테이션

AWS KMS 는 대칭 암호화 KMS 키에서 키 구성 요소의 자동 및 온디맨드 교체를 지원합니다.

가 키 구성 요소를 AWS KMS 교체할 때마다 EventBridge로 `KMS CMK Rotation` 이벤트를 전송합니다.는 최선을 다해이 이벤트를 AWS KMS 생성합니다.

다음은 이 이벤트의 예입니다.

```
{
  "version": "0",
  "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
  "detail-type": "KMS CMK Rotation",
  "source": "aws.kms",
  "account": "111122223333",
  "time": "2025-05-23T03:11:54Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  ],
  "detail": {
    "key-id": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "key-origin": "AWS_KMS",
    "rotation-type": "ON_DEMAND",
    "previous-key-material-id":
 "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0",
    "current-key-material-id":
 "96083e4fb6dbc41d77578a213a6b6669c044dd4c143e96755396d2bf11fd6068"
  }
}
```

KMS가 가져온 키 자료 만료

키 구성 요소를 KMS 키로 가져올 때 선택적으로 키 구성 요소의 만료 시간을 지정할 수 있습니다. 키 구성 요소가 만료되면는 키 구성 요소를 AWS KMS 삭제하고 해당 `KMS Imported Key Material Expiration` 이벤트를 EventBridge로 전송합니다.는 최선을 다해이 이벤트를 AWS KMS 생성합니다.

다음은 이 이벤트의 예입니다.

```
{
  "version": "0",
  "id": "9da9af57-9253-4406-87cb-7cc400e43465",
```

```
  "detail-type": "KMS Imported Key Material Expiration",
  "source": "aws.kms",
  "account": "111122223333",
  "time": "2025-05-23T03:11:54Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  ],
  "detail": {
    "key-id": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "key-material-id":
 "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0"
  }
}
```

KMS CMK 삭제

KMS 키에 삭제를 예약하면 AWS KMS 가 KMS 키를 삭제하기 전에 대기 기간을 적용합니다. 대기 기간이 끝나면 AWS KMS 는 KMS 키를 삭제하고 `KMS CMK Deletion` 이벤트를 EventBridge로 보냅니다. AWS KMS 가 이 EventBridge 이벤트를 보장합니다. 재시도로 인해 몇 초 내에 여러 개의 이벤트가 생성되어 동일한 KMS 키가 삭제될 수 있습니다.

다음은 이 이벤트의 예입니다.

```
{
  "version": "0",
  "id": "e9ce3425-7d22-412a-a699-e7a5fc3fbc9a",
  "detail-type": "KMS CMK Deletion",
  "source": "aws.kms",
  "account": "111122223333",
  "time": "2025-05-23T03:11:54Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  ],
  "detail": {
    "key-id": "1234abcd-12ab-34cd-56ef-1234567890ab"
  }
}
```

의 별칭 AWS KMS

별칭은 AWS KMS key의 알아보기 쉬운 다른 이름입니다. 예를 들어 별칭을 사용하면 KMS 키를 `1234abcd-12ab-34cd-56ef-1234567890ab` 대신 `test-key`로 지칭할 수 있습니다.

별칭을 사용하여 AWS KMS 콘솔, DescribeKey 작업 및 Encrypt 및 GenerateDataKey와 같은 암호화 작업에서 KMS 키를 식별할 수 있습니다. 별칭을 사용하면 쉽게 AWS 관리형 키를 인식할 수 있습니다. 이러한 KMS 키의 별칭은 항상 `aws/<service-name>` 형식으로 되어 있습니다. 예를 들어 Amazon DynamoDB AWS 관리형 키 용의 별칭은 입니다`aws/dynamodb`. 별칭을 프로젝트 또는 범주 이름으로 시작하는 것처럼 프로젝트에 비슷한 별칭 표준을 설정할 수 있습니다.

정책을 편집하거나 권한 부여를 관리하지 않고도 별칭을 기반으로 KMS 키에 대한 액세스를 허용하거나 거부할 수도 있습니다. 이 기능은 속성 기반 액세스 제어(ABAC) AWS KMS 지원의 일부입니다. 자세한 내용은 별칭을 사용하여 KMS 키에 대한 액세스 제어을 참조하세요.

별칭의 강력한 기능은 언제든지 별칭과 연결된 KMS 키를 변경할 수 있는 기능에서 비롯됩니다. 별칭을 사용하면 코드를 더 쉽게 작성하고 유지 관리할 수 있습니다. 예를 들어 별칭을 사용하여 특정 KMS 키를 참조하고 KMS 키를 변경한다고 가정합니다. 이 경우 별칭을 다른 KMS 키와 연결하기만 하면 됩니다. 코드를 변경할 필요가 없습니다.

또한 별칭을 사용하면 다른 AWS 리전에서 동일한 코드를 더 쉽게 재사용할 수 있습니다. 여러 리전에서 동일한 이름의 별칭을 만들고 각 별칭을 해당 리전의 KMS 키와 연결합니다. 각 리전에서 코드가 실행되는 경우 별칭은 해당 리전의 연결된 KMS 키를 참조합니다. 예제는 애플리케이션에서 별칭을 사용하는 방법 알아보기 섹션을 참조하세요.

AWS KMS 콘솔에서 CreateAlias API를 사용하거나 AWS::KMS::Alias AWS::KMS::Alias CloudFormation 칭을 생성할 수 있습니다.

AWS KMS API는 각 계정 및 리전의 별칭을 완벽하게 제어합니다. API에는 별칭 생성(CreateAlias), 별칭 이름 및 별칭 ARN 보기(ListAliases), 별칭과 연결된 KMS 키 변경(UpdateAlias), 별칭 삭제(DeleteAlias) 작업이 포함됩니다.

별칭의 작동 방식

AWS KMS에서 별칭의 작동 방식을 알아봅니다.

별칭은 독립적인 AWS 리소스입니다.

> 별칭은 KMS 키의 속성이 아닙니다. 별칭에 대해 수행하는 작업은 연결된 KMS 키에 영향을 주지 않습니다. KMS 키의 별칭을 만든 다음 다른 KMS 키와 연결되도록 별칭을 업데이트할 수 있습니다.

연결된 KMS 키에 영향을 주지 않고 별칭을 삭제할 수도 있습니다. 그러나 KMS 키를 삭제하면 해당 KMS 키와 연결된 모든 별칭이 삭제됩니다.

IAM 정책에서 별칭을 리소스로 지정하는 경우 정책은 연결된 KMS 키가 아니라 별칭을 참조합니다.

각 별칭에는 두 가지 형식이 있습니다.

별칭을 생성할 때 별칭 이름을 지정합니다.는 별칭 ARN을 AWS KMS 생성합니다.

- 별칭 ARN은 별칭을 고유하게 식별하는 Amazon 리소스 이름(ARN)입니다.

```
# Alias ARN
arn:aws:kms:us-west-2:111122223333:alias/<alias-name>
```

- 계정 및 리전에서 고유한 별칭 이름입니다. AWS KMS API에서 별칭 이름에는 항상 접두사가 붙습니다`alias/`. 이 접두사는 AWS KMS 콘솔에서 생략됩니다.

```
# Alias name
alias/<alias-name>
```

별칭은 비밀이 아닙니다.

별칭은 CloudTrail 로그 및 기타 출력에 일반 텍스트로 표시될 수 있습니다. 별칭 이름에 기밀 또는 민감한 정보를 포함시키지 마십시오.

각 별칭은 한 번에 하나의 KMS 키와 연결됩니다.

별칭과 별칭의 KMS 키는 동일한 계정 및 리전에 있어야 합니다.

별칭을 동일한 AWS 계정 및 리전의 모든 고객 관리형 키와 연결할 수 있습니다. 그러나 별칭을 AWS 관리형 키와 연결할 수 있는 권한은 없습니다.

예를 들어 이 ListAliases 출력은 `test-key` 별칭이 `TargetKeyId` 속성으로 표시되는 정확히 하나의 대상 KMS 키와 연결되어 있음을 보여줍니다.

```
{
     "AliasName": "alias/test-key",
     "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/test-key",
     "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
     "CreationDate": 1593622000.191,
     "LastUpdatedDate": 1593622000.191
}
```

동일한 KMS 키와 여러 별칭을 연결할 수 있습니다.

예를 들어, `test-key` 및 `project-key` 별칭을 동일한 KMS 키로 연결할 수 있습니다.

```
{
     "AliasName": "alias/test-key",
     "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/test-key",
     "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
     "CreationDate": 1593622000.191,
     "LastUpdatedDate": 1593622000.191
},
{
     "AliasName": "alias/project-key",
     "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/project-key",
     "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
     "CreationDate": 1516435200.399,
     "LastUpdatedDate": 1516435200.399
}
```

별칭은 계정 및 리전 내에서 고유해야 합니다.

예를 들어, 각 계정 및 리전에서 오직 한 개의 `test-key` 별칭만 가질 수 있습니다. 별칭은 대소문자를 구분하지만 대소문자만 다른 별칭은 오류가 발생하기 쉽습니다. 별칭 이름은 변경할 수 없습니다. 그러나 별칭을 삭제하고 원하는 이름으로 새 별칭을 생성할 수 있습니다.

다른 리전에서 같은 이름으로 별칭을 만들 수 있습니다.

예를 들어, 미국 동부(버지니아 북부)에 `finance-key` 별칭이 있고 유럽(프랑크푸르트)에 `finance-key` 별칭이 있을 수 있습니다. 각 별칭은 해당 리전의 KMS 키와 연결됩니다. 코드가 `alias/finance-key`와 같은 별칭 이름을 참조하는 경우 여러 리전에서 실행할 수 있습니다. 각 리전에서는 다른 KMS 키를 사용합니다. 자세한 내용은 애플리케이션에서 별칭을 사용하는 방법 알아보기 섹션을 참조하세요.

별칭과 연결된 KMS 키를 변경할 수 있습니다.

UpdateAlias 작업을 사용하여 별칭을 다른 KMS 키와 연결할 수 있습니다. 예를 들어, `finance-key` 별칭이 `1234abcd-12ab-34cd-56ef-1234567890ab` KMS 키와 연결된 경우 `0987dcba-09fe-87dc-65ba-ab0987654321` KMS 키와 연결되도록 업데이트할 수 있습니다.

그러나 현재 KMS 키와 새 KMS 키는 동일한 유형(모두 대칭, 모두 비대칭 또는 모두 HMAC)이어야 하며 키 사용(ENCRYPT_DECRYPT, SIGN_VERIFY 또는 GENERATE_VERIFY_MAC)이 동일해야 합니다. 이 제한은 별칭을 사용하는 코드의 오류를 방지합니다. 별칭을 다른 유형의 키와 연결해야 하고 위험을 완화한 경우 별칭을 삭제하고 다시 만들 수 있습니다.

일부 KMS 키에는 별칭이 없습니다.

AWS KMS 콘솔에서 KMS 키를 생성할 때 새 별칭을 지정해야 합니다. 그러나 CreateKey 작업을 사용하여 KMS 키를 만들 때는 별칭이 필요하지 않습니다. 또한 UpdateAlias 작업을 사용하여 별칭과 연결된 KMS 키를 변경하고 DeleteAlias 작업을 사용하여 별칭을 삭제할 수 있습니다. 따라서 일부 KMS 키에는 별칭이 여러 개 있을 수 있으며 일부 키에는 별칭이 없을 수도 있습니다.

AWS 는 계정에 별칭을 생성합니다.

AWS 는 계정에에 대한 별칭을 생성합니다AWS 관리형 키. 이러한 별칭에는 `alias/aws/<service-name>` 형식의 이름(예: `alias/aws/s3`)이 있습니다.

일부 AWS 별칭에는 KMS 키가 없습니다. 이러한 사전 정의된 별칭은 일반적으로 서비스 사용을 시작할 AWS 관리형 키 때와 연결됩니다.

별칭을 사용하여 KMS 키 식별

별칭 이름 또는 별칭 ARN을 사용하여 암호화 작업, DescribeKey 및 GetPublicKey에서 KMS 키를 식별할 수 있습니다. (KMS 키가 다른 AWS 계정에 있는 경우 해당 키 ARN 또는 별칭 ARN을 사용해야 합니다.) 별칭은 다른 AWS KMS 작업의 KMS 키에 대한 유효한 식별자가 아닙니다. 각 AWS KMS API 작업의 유효한 키 식별자에 대한 자세한 내용은 AWS Key Management Service API 참조의 `KeyId` 파라미터 설명을 참조하세요.

별칭 이름이나 별칭 ARN을 사용하여 IAM 정책에서 KMS 키를 식별할 수 없습니다. 별칭을 기반으로 KMS 키에 대한 액세스를 제어하려면 kms:RequestAlias 또는 kms:ResourceAliases 조건 키를 사용합니다. 자세한 내용은 에 대한 ABAC AWS KMS을 참조하세요.

별칭에 대한 액세스 제어

별칭을 만들거나 변경하면 별칭 및 연결된 KMS 키에 영향을 줍니다. 따라서 별칭을 관리하는 보안 주체는 별칭과 영향을 받는 모든 KMS 키에 대해 별칭 작업을 호출할 수 있는 권한이 있어야 합니다. 키 정책, IAM 정책 및 권한 부여를 사용해 이러한 권한을 제공할 수 있습니다.

> **Note**
>
> 보안 주체에 태그와 별칭을 관리할 수 있는 권한을 부여할 때는 주의해야 합니다. 태그나 별칭을 변경하면 고객 관리형 키의 권한을 허용하거나 거부할 수 있습니다. 자세한 내용은 에 대한 ABAC AWS KMS 및 별칭을 사용하여 KMS 키에 대한 액세스 제어 섹션을 참조하세요.

모든 AWS KMS 작업에 대한 액세스 제어에 대한 자세한 내용은 섹션을 참조하세요권한 참조.

별칭을 만들고 관리할 수 있는 권한은 다음과 같습니다.

kms:CreateAlias

별칭을 만들려면 보안 주체는 별칭과 연결된 KMS 키 모두에 대해 다음 권한이 필요합니다.

- 별칭에 대한 `kms:CreateAlias`. 별칭을 생성할 수 있는 보안 주체에 연결된 IAM 정책에서 이 권한을 제공합니다.

  다음 예제 정책 설명은 `Resource` 요소의 특정 별칭을 지정합니다. 그러나 여러 별칭 ARN을 나열하거나 'test*'와 같은 별칭 패턴을 지정할 수 있습니다. 보안 주체가 계정 및 리전에서 별칭을 생성할 수 있도록 `Resource`의 값을 `"*"`로 지정할 수도 있습니다. 별칭 생성 권한은 계정 및 리전의 모든 리소스에 대한 `kms:Create*` 권한에도 포함될 수 있습니다.

```
{
  "Sid": "IAMPolicyForAnAlias",
  "Effect": "Allow",
  "Action": [
    "kms:CreateAlias",
    "kms:UpdateAlias",
    "kms:DeleteAlias"
  ],
  "Resource": "arn:aws:kms:us-west-2:111122223333:alias/test-key"
}
```

- KMS 키에 대한 `kms:CreateAlias`. 이 권한은 키 정책 또는 키 정책에서 위임된 IAM 정책에서 제공되어야 합니다.

```
{
  "Sid": "Key policy for 1234abcd-12ab-34cd-56ef-1234567890ab",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:user/KMSAdminUser"},
  "Action": [
    "kms:CreateAlias",
    "kms:DescribeKey"
  ],
  "Resource": "*"
}
```

조건 키를 사용하여 별칭과 연결할 수 있는 KMS 키를 제한할 수 있습니다. 예를 들어 kms:KeySpec 조건 키를 사용하여 보안 주체가 비대칭 KMS 키에 대해서만 별칭을 생성하도록 할 수 있습니다. KMS 키

리소스에 대한 kms:CreateAlias 권한을 제한하는 데 사용할 수 있는 조건 키의 전체 목록은 AWS KMS 권한 섹션을 참조하세요.

kms:ListAliases

계정 및 리전의 별칭을 나열하려면 보안 주체에게 kms:ListAliases IAM 정책에서 권한이 있어야 합니다. 이 정책은 특정 KMS 키 또는 별칭 리소스와 관련이 없기 때문에 정책의 리소스 요소 값은 "*"여야 합니다.

예를 들어 다음 IAM 정책문은 보안 주체에게 계정 및 리전의 모든 KMS 키와 별칭을 나열할 수 있는 권한을 부여합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "kms:ListKeys",
      "kms:ListAliases"
    ],
    "Resource": "*"
  }
}
```

kms:UpdateAlias

별칭과 연결된 KMS 키를 변경하려면 보안 주체에 별칭, 현재 KMS 키에 대한 권한 요소 및 새 KMS 키에 대한 권한 요소가 필요합니다.

예를 들어 test-key 별칭을 키 ID가 1234abcd-12ab-34cd-56ef-1234567890ab인 KMS 키에서 키 ID가 0987dcba-09fe-87dc-65ba-ab0987654321인 KMS 키로 변경하려고 한다고 가정합니다. 이 경우 이 섹션의 예제와 비슷한 정책문을 포함합니다.

- 별칭에 대한 kms:UpdateAlias. 보안 주체에 연결된 IAM 정책에서 이 권한을 제공합니다. 다음 IAM 정책은 특정 별칭을 지정합니다. 그러나 여러 별칭 ARN을 나열하거나 별칭 패턴(예: "test*")을 지정할 수 있습니다. 보안 주체가 계정 및 리전에서 별칭을 업데이트할 수 있도록 Resource의 값을 "*"로 지정할 수도 있습니다.

```
{
  "Sid": "IAMPolicyForAnAlias",
  "Effect": "Allow",
  "Action": [
    "kms:UpdateAlias",
    "kms:ListAliases",
    "kms:ListKeys"
  ],
  "Resource": "arn:aws:kms:us-west-2:111122223333:alias/test-key"
}
```

- 현재 별칭과 연결된 KMS 키에 대한 `kms:UpdateAlias`. 이 권한은 키 정책 또는 키 정책에서 위임된 IAM 정책에서 제공되어야 합니다.

```
{
  "Sid": "Key policy for 1234abcd-12ab-34cd-56ef-1234567890ab",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:user/KMSAdminUser"},
  "Action": [
    "kms:UpdateAlias",
    "kms:DescribeKey"
  ],
  "Resource": "*"
}
```

- 작업으로 별칭에 연결 하는 KMS 키에 대한 `kms:UpdateAlias`. 이 권한은 키 정책 또는 키 정책에서 위임된 IAM 정책에서 제공되어야 합니다.

```
{
  "Sid": "Key policy for 0987dcba-09fe-87dc-65ba-ab0987654321",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:user/KMSAdminUser"},
  "Action": [
    "kms:UpdateAlias",
    "kms:DescribeKey"
  ],
  "Resource": "*"
}
```

조건 키를 사용하여 UpdateAlias 작업에서 KMS 키 중 하나 또는 둘 다를 제한할 수 있습니다. 예를 들어, kms:ResourceAliases 조건 키를 사용하여 대상 KMS 키에 이미 특정 별칭이 있는 경우에만 보안 주체가 별칭을 업데이트하도록 할 수 있습니다. KMS 키 리소스에 대한 kms:UpdateAlias 권한을 제한하는 데 사용할 수 있는 조건 키의 전체 목록은 AWS KMS 권한 섹션을 참조하세요.

kms:DeleteAlias

별칭을 삭제하려면 보안 주체는 별칭과 연결된 KMS 키에 대해 권한이 필요합니다.

보안 주체에 리소스를 삭제할 수 있는 권한을 부여할 때는 항상 주의해야 합니다. 그러나 별칭을 삭제해도 연결된 KMS 키에 영향을 미치지 않습니다. 별칭을 사용하는 애플리케이션에서 오류가 발생할 수 있지만 실수로 별칭을 삭제하면 다시 만들 수 있습니다.

- 별칭에 대한 kms:DeleteAlias. 별칭을 삭제할 수 있는 보안 주체에 연결된 IAM 정책에서 이 권한을 제공합니다.

  다음 예제 정책 설명은 Resource 요소의 별칭을 지정합니다. 그러나 여러 별칭 ARN을 나열하거나 별칭 패턴(예: "test*")을 지정할 수 있습니다. 또한 보안 주체가 계정 및 리전에서 별칭을 삭제할 수 있도록 Resource"*" 값을 지정할 수도 있습니다.

```
{
  "Sid": "IAMPolicyForAnAlias",
  "Effect": "Allow",
  "Action": [
    "kms:CreateAlias",
    "kms:UpdateAlias",
    "kms:DeleteAlias"
  ],
  "Resource": "arn:aws:kms:us-west-2:111122223333:alias/test-key"
}
```

- 연결된 KMS 키에 대한 kms:DeleteAlias. 이 권한은 키 정책 또는 키 정책에서 위임된 IAM 정책에서 제공되어야 합니다.

```
{
  "Sid": "Key policy for 1234abcd-12ab-34cd-56ef-1234567890ab",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:user/KMSAdminUser"
  },
  "Action": [
```

```
    "kms:CreateAlias",
    "kms:UpdateAlias",
    "kms:DeleteAlias",
    "kms:DescribeKey"
  ],
  "Resource": "*"
}
```

별칭 권한 제한

리소스가 KMS 키인 경우 조건 키를 사용하여 별칭 권한을 제한할 수 있습니다. 예를 들어, 다음 IAM 정책은 특정 계정 및 리전의 KMS 키에 대한 별칭 작업을 허용합니다. 그러나 kms:KeyOrigin 조건 키를 사용하여 키 구성 요소가 있는 KMS 키에 대한 권한을 추가로 제한합니다 AWS KMS.

KMS 키 리소스에 대한 별칭 권한을 제한하는 데 사용할 수 있는 조건 키의 전체 목록은 AWS KMS 권한 섹션을 참조하세요.

```
{
  "Sid": "IAMPolicyKeyPermissions",
  "Effect": "Allow",
  "Resource": "arn:aws:kms:us-west-2:111122223333:key/*",
  "Action": [
    "kms:CreateAlias",
    "kms:UpdateAlias",
    "kms:DeleteAlias"
  ],
  "Condition": {
    "StringEquals": {
      "kms:KeyOrigin": "AWS_KMS"
    }
  }
}
```

리소스가 별칭인 정책문에는 조건 키를 사용할 수 없습니다. 보안 주체가 관리할 수 있는 별칭을 제한하려면 별칭에 대한 액세스를 제어하는 IAM 정책문의 Resource 요소 값을 사용합니다. 예를 들어 다음 정책 설명을 통해 보안 주체는 별칭이 로 시작하지 않는 한 AWS 계정 및 리전에서 별칭을 생성, 업데이트 또는 삭제할 수 있습니다Restricted.

```
{
  "Sid": "IAMPolicyForAnAliasAllow",
```

```
  "Effect": "Allow",
  "Action": [
    "kms:CreateAlias",
    "kms:UpdateAlias",
    "kms:DeleteAlias"
  ],
  "Resource": "arn:aws:kms:us-west-2:111122223333:alias/*"
},
{
  "Sid": "IAMPolicyForAnAliasDeny",
  "Effect": "Deny",
  "Action": [
    "kms:CreateAlias",
    "kms:UpdateAlias",
    "kms:DeleteAlias"
  ],
  "Resource": "arn:aws:kms:us-west-2:111122223333:alias/Restricted*"
}
```

별칭 생성

AWS KMS 콘솔에서 또는 AWS KMS API 작업을 사용하여 별칭을 생성할 수 있습니다.

별칭은 1~256자의 문자열이어야 합니다. 여기에는 영숫자, 슬래시(/), 밑줄(_) 및 대시(-)만 포함할 수 있습니다. 고객 관리형 키의 별칭 이름은 `alias/aws/`로 시작할 수 없습니다. `alias/aws/` 접두사는 AWS 관리형 키용으로 예약되어 있습니다.

새 KMS 키 또는 기존 KMS 키에 대한 별칭을 만들 수 있습니다. 특정 KMS 키가 프로젝트 또는 애플리케이션에서 사용되도록 별칭을 추가할 수 있습니다.

AWS CloudFormation 템플릿을 사용하여 KMS 키의 별칭을 생성할 수도 있습니다. 자세한 내용은 AWS CloudFormation 사용 설명서의 AWS::KMS::Alias를 참조하세요.

AWS KMS 콘솔 사용

AWS KMS 콘솔에서 KMS 키를 생성할 때 새 KMS 키에 대한 별칭을 생성해야 합니다. 기존 KMS 키에 대한 별칭을 만들려면 KMS 키에 대한 세부 정보 페이지에서 별칭 탭을 사용합니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.

3. 탐색 창에서 고객 관리형 키를 선택합니다. AWS 관리형 키 또는에 대한 별칭은 관리할 수 없습니다 AWS 소유 키.
4. 테이블에서 KMS 키의 키 ID 또는 별칭을 선택합니다. 그런 다음 KMS 키 세부 정보 페이지에서별칭 탭을 선택합니다.

   KMS 키에 별칭이 여러 개 있는 경우 테이블의 별칭 열에는 하나의 별칭과 별칭 요약(예: (n개 이상))이 표시됩니다. 별칭 요약을 선택하면 KMS 키 세부 정보 페이지의 별칭 탭으로 바로 이동합니다.
5. 별칭 탭에서 별칭 생성을 선택합니다. 별칭 이름을 입력하고 별칭 생성을 선택합니다.

> **Important**
>
> 이 필드에 기밀 또는 민감한 정보를 포함하지 마십시오. 이 필드는 CloudTrail 로그 및 기타 출력에 일반 텍스트로 표시될 수 있습니다.

> **Note**
>
> `alias/` 접두사를 추가하지 마십시오. 콘솔이 자동으로 추가합니다. `alias/ExampleAlias`를 입력하면, 실제 별칭 이름은 `alias/alias/ExampleAlias`입니다.

AWS KMS API 사용

별칭을 생성하려면 CreateAlias 작업을 사용합니다. 콘솔에서 KMS 키를 만드는 프로세스와 달리 CreateKey 작업은 새 KMS 키에 대한 별칭을 만들지 않습니다.

> **Important**
>
> 이 필드에 기밀 또는 민감한 정보를 포함하지 마십시오. 이 필드는 CloudTrail 로그 및 기타 출력에 일반 텍스트로 표시될 수 있습니다.

`CreateAlias` 작업을 사용하여 별칭이 없는 새 KMS 키의 별칭을 만들 수 있습니다. `CreateAlias` 작업을 사용하여 기존 KMS 키에 별칭을 추가하거나 실수로 삭제된 별칭을 다시 만들 수도 있습니다.

AWS KMS API 작업에서 별칭 이름은 로 시작하고 그 `alias/` 뒤에 같은 이름이와 같아야 합니다.`alias/ExampleAlias`. 별칭은 계정 및 리전 내에서 고유해야 합니다. 이미 사용 중인 별칭 이름을 찾으려면 ListAliases 작업을 사용합니다. 별칭 이름은 대/소문자를 구분합니다.

`TargetKeyId`는 동일한 AWS 리전의 모든 고객 관리형 키일 수 있습니다. KMS 키를 식별하려면, 이 KMS 키의 키 ID 또는 키 ARN을 사용합니다. 다른 별칭을 사용할 수 없습니다.

다음 예제에서는 `example-key` 별칭을 만들고 지정된 KMS 키와 연결합니다. 이 예제에서는 AWS Command Line Interface ()를 사용합니다AWS CLI. 다양한 프로그래밍 언어의 예는 AWS SDK 또는 CLI와 `CreateAlias` 함께 사용 섹션을 참조하세요.

```
$ aws kms create-alias \
    --alias-name alias/example-key \
    --target-key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

`CreateAlias`는 출력을 반환하지 않습니다. 새 별칭을 보려면 `ListAliases` 작업을 사용합니다. 자세한 내용은 AWS KMS API 사용을 참조하세요.

KMS 키의 별칭 이름 및 별칭 ARN 찾기

별칭을 사용하면 AWS KMS 콘솔에서 KMS 키를 쉽게 인식할 수 있습니다. AWS KMS 콘솔에서 또는 ListAliases 작업을 사용하여 KMS 키의 별칭을 볼 수 있습니다. KMS 키의 속성을 반환하는 DescribeKey 작업에는 별칭이 포함되지 않습니다.

다음 절차에서는 AWS KMS 콘솔 및 AWS KMS API를 사용하여 KMS 키와 연결된 별칭을 보고 식별하는 방법을 보여줍니다. AWS KMS API 예제에서는 AWS Command Line Interface (AWS CLI)를 사용하지만 지원되는 모든 프로그래밍 언어를 사용할 수 있습니다.

AWS KMS 콘솔 사용

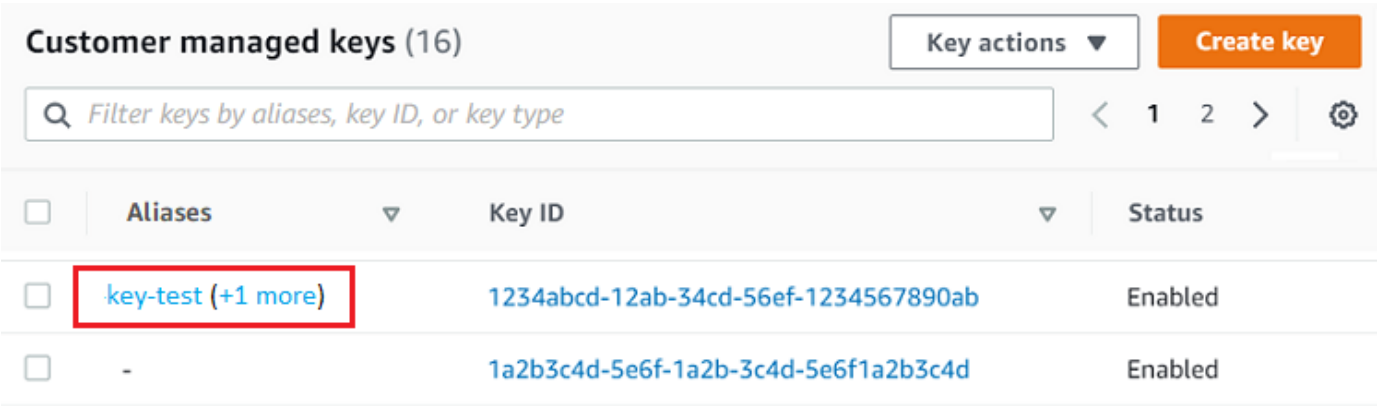
AWS KMS 콘솔에 KMS 키와 연결된 별칭이 표시됩니다.

1. https://console.aws.amazon.com/kms AWS KMS 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 해당 계정에서 직접 생성하고 관리하는 키를 보려면 탐색 창에서 고객 관리형 키를 선택합니다. 가 AWS 자동으로 생성하고 관리하는 계정의 키를 보려면 탐색 창에서 AWS 관리형 키를 선택합니다.
4. 별칭 열에는 각 KMS 키의 별칭이 표시됩니다. KMS 키에 별칭이 없는 경우 별칭 열에 대시(-) 가 나타납니다.

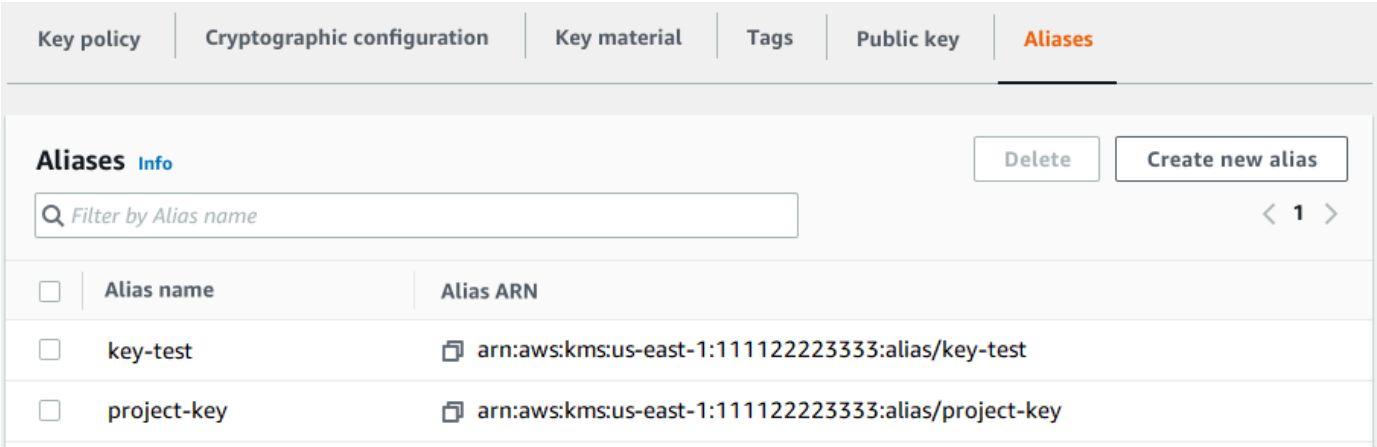
KMS 키에 별칭이 여러 개 있는 경우 별칭 열에는 별칭 요약(예: (n개 이상))도 있습니다. 예를 들어 다음 KMS 키에는 두 개의 별칭이 있으며 그 중 하나는 `key-test`입니다.

KMS 키에 대한 모든 별칭의 별칭 이름과 별칭 ARN을 찾으려면 별칭(Aliases) 탭을 사용합니다.

- 별칭(Aliases) 탭으로 직접 이동하려면 별칭(Aliases) 열에서 별칭 요약(n개 이상)을 선택합니다. 별칭 요약은 KMS 키에 둘 이상의 별칭이 있는 경우에만 나타납니다.
- 또는 KMS 키의 별칭 또는 키 ID를 선택(KMS 키의 세부 정보 페이지가 열림)한 다음별칭(Aliases) 탭을 선택합니다. 일반 구성 섹션 아래에 탭이 표시됩니다.

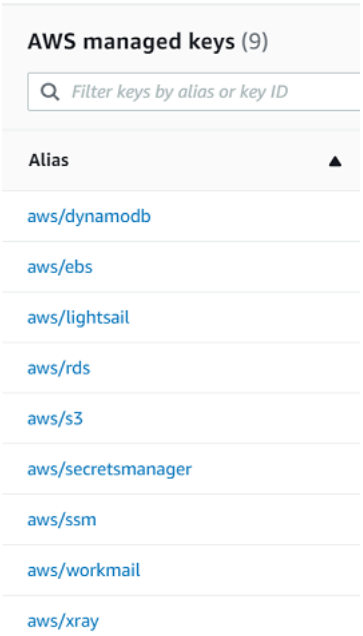


5. 별칭(Aliases) 탭에는 KMS 키에 대한 모든 별칭의 별칭 이름과 별칭 ARN이 표시됩니다. 이 탭에서 KMS 키의 별칭을 만들고 삭제할 수도 있습니다.



AWS 관리형 키

이 예제 AWS 관리형 키 페이지에 표시된 대로 AWS 관리형 키별칭을 사용하여를 인식할 수 있습니다. AWS 관리형 키 의 별칭 형식은 aws/*<service-name>*입니다. 예를 들어 Amazon DynamoDB AWS 관리형 키 용의 별칭은 입니다aws/dynamodb.



AWS KMS API 사용

ListAliases 작업은 계정 및 리전의 별칭 이름과 별칭 ARN을 반환합니다. 출력에는 고객 관리형 키에 대한 AWS 관리형 키 및 별칭이 포함됩니다. AWS 관리형 키 의 별칭 형식은 aws/*<service-name>*(예: aws/dynamodb)입니다.

응답에는 TargetKeyId 필드가 없는 별칭도 포함될 수 있습니다. 이는 AWS 가 생성했지만 아직 KMS 키와 연결되지 않은 사전 정의된 별칭입니다.

```
$ aws kms list-aliases
{
    "Aliases": [
        {
            "AliasName": "alias/access-key",
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/access-key",
            "TargetKeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
            "CreationDate": 1516435200.399,
            "LastUpdatedDate": 1516435200.399
        },
        {
            "AliasName": "alias/ECC-P521-Sign",
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/ECC-P521-Sign",
```

```
            "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
            "CreationDate": 1693622000.704,
            "LastUpdatedDate": 1693622000.704
        },
        {
            "AliasName": "alias/ImportedKey",
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/ImportedKey",
            "TargetKeyId": "1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d",
            "CreationDate": 1493622000.704,
            "LastUpdatedDate": 1521097200.235
        },
        {
            "AliasName": "alias/finance-project",
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/finance-project",
            "TargetKeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
            "CreationDate": 1604958290.014,
            "LastUpdatedDate": 1604958290.014
        },
        {
            "AliasName": "alias/aws/dynamodb",
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/aws/dynamodb",
            "TargetKeyId": "0987ab65-43cd-21ef-09ab-87654321cdef",
            "CreationDate": 1521097200.454,
            "LastUpdatedDate": 1521097200.454
        },
        {
            "AliasName": "alias/aws/ebs",
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/aws/ebs",
            "TargetKeyId": "abcd1234-09fe-ef90-09fe-ab0987654321",
            "CreationDate": 1466518990.200,
            "LastUpdatedDate": 1466518990.200
        }
    ]
}
```


특정 KMS 키와 연결된 별칭을 모두 가져오려면 `ListAliases` 작업의 `KeyId` 파라미터 옵션을 사용합니다. `KeyId` 파라미터는 KMS 키의 키 ID 또는 키 ARN을 사용합니다.

이 예에서는 `0987dcba-09fe-87dc-65ba-ab0987654321` KMS 키와 연결된 모든 별칭을 가져옵니다.

```
$ aws kms list-aliases --key-id 0987dcba-09fe-87dc-65ba-ab0987654321
{
```

```
    "Aliases": [
        {
            "AliasName": "alias/access-key",
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/access-key",
            "TargetKeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
            "CreationDate": "2018-01-20T15:23:10.194000-07:00",
            "LastUpdatedDate": "2018-01-20T15:23:10.194000-07:00"
        },
        {
            "AliasName": "alias/finance-project",
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/finance-project",
            "TargetKeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
            "CreationDate": 1604958290.014,
            "LastUpdatedDate": 1604958290.014
        }
    ]
}
```

KeyId 파라미터는 와일드카드 문자를 사용하지 않지만 프로그래밍 언어의 기능을 사용하여 응답을 필터링할 수 있습니다.

예를 들어 다음 AWS CLI 명령은의 별칭만 가져옵니다 AWS 관리형 키.

```
$ aws kms list-aliases --query 'Aliases[?starts_with(AliasName, `alias/aws/`)]'
```

다음 명령은 access-key 별칭만 가져옵니다. 별칭 이름은 대/소문자를 구분합니다.

```
$ aws kms list-aliases --query 'Aliases[?AliasName==`alias/access-key`]'
[
    {
        "AliasName": "alias/access-key",
        "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/access-key",
        "TargetKeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
        "CreationDate": "2018-01-20T15:23:10.194000-07:00",
        "LastUpdatedDate": "2018-01-20T15:23:10.194000-07:00"
    }
]
```

별칭 업데이트

별칭은 독립 리소스이므로 별칭과 연결된 KMS 키를 변경할 수 있습니다. 예를 들어, `test-key` 별칭이 하나의 KMS 키와 연결된 경우 UpdateAlias 작업을 사용하여 다른 KMS 키와 연결할 수 있습니다. 이는 키 구성 요소를 변경하지 않고 KMS 키를 수동으로 교체하는 여러 방법 중 하나입니다. 새 리소스에 대해 하나의 KMS 키를 사용하던 애플리케이션이 이제 다른 KMS 키를 사용하도록 KMS 키를 업데이트할 수도 있습니다.

AWS KMS 콘솔에서는 별칭을 업데이트할 수 없습니다. 또한 `UpdateAlias`(또는 다른 작업)를 사용하여 별칭 이름을 변경할 수 없습니다. 별칭 이름을 변경하려면 현재 별칭을 삭제한 후 KMS 키에 대해 새 별칭을 생성합니다.

별칭을 업데이트할 때 현재 KMS 키와 새 KMS 키의 유형이 동일해야 합니다(모두 대칭, 비대칭 또는 HMAC). 또한 키 사용도 동일해야 합니다(`ENCRYPT_DECRYPT`, `SIGN_VERIFY` 또는 `GENERATE_VERIFY_MAC`). 이 제한은 별칭을 사용하는 코드에서 암호화 오류를 방지합니다.

다음 예에서는 먼저 ListAliases 작업을 사용하여 `test-key` 별칭이 현재 KMS 키 `1234abcd-12ab-34cd-56ef-1234567890ab`와 연결되어 있음을 보여 줍니다.

```
$ aws kms list-aliases --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
{
    "Aliases": [
        {
            "AliasName": "alias/test-key",
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/test-key",
            "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
            "CreationDate": 1593622000.191,
            "LastUpdatedDate": 1593622000.191
        }
    ]
}
```

그런 다음 `UpdateAlias` 작업을 사용하여 `test-key` 별칭과 연결된 KMS 키를 KMS 키 `0987dcba-09fe-87dc-65ba-ab0987654321`로 변경합니다. 현재 연결된 KMS 키를 지정할 필요가 없으며 새 ("대상") KMS 키만 지정하면 됩니다. 별칭 이름은 대/소문자를 구분합니다.

```
$ aws kms update-alias --alias-name 'alias/test-key' --target-key-id
 0987dcba-09fe-87dc-65ba-ab0987654321
```

별칭이 이제 대상 KMS 키와 연결되어 있는지 확인하려면 `ListAliases` 작업을 다시 수행합니다. 이 AWS CLI 명령은 `--query` 파라미터를 사용하여 `test-key`별칭만 가져옵니다. `TargetKeyId` 및 `LastUpdatedDate` 필드가 업데이트됩니다.

```
$ aws kms list-aliases --query 'Aliases[?AliasName==`alias/test-key`]'
[
    {
        "AliasName": "alias/test-key",
        "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/test-key",
        "TargetKeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
        "CreationDate": 1593622000.191,
        "LastUpdatedDate": 1604958290.154
    }
]
```

별칭 삭제

AWS KMS 콘솔에서 또는 DeleteAlias 작업을 사용하여 별칭을 삭제할 수 있습니다. 별칭을 삭제하기 전에 별칭이 사용되지 않는지 확인합니다. 별칭을 삭제해도 연결된 KMS 키에는 영향을 주지 않지만 별칭을 사용하는 애플리케이션에 문제가 발생할 수 있습니다. 실수로 별칭을 삭제한 경우 이름이 같은 새 별칭을 만들어 동일한 또는 다른 KMS 키와 연결할 수 있습니다.

KMS 키를 삭제하면 해당 KMS 키와 연결된 모든 별칭이 삭제됩니다.

AWS KMS 콘솔 사용

AWS KMS 콘솔에서 별칭을 삭제하려면 KMS 키에 대한 세부 정보 페이지의 별칭 탭을 사용합니다. KMS 키의 별칭을 한 번에 여러 개 삭제할 수 있습니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 탐색 창에서 고객 관리형 키를 선택합니다. AWS 관리형 키 또는에 대한 별칭은 관리할 수 없습니다 AWS 소유 키.
4. 테이블에서 KMS 키의 키 ID 또는 별칭을 선택합니다. 그런 다음 KMS 키 세부 정보 페이지에서별칭 탭을 선택합니다.

KMS 키에 별칭이 여러 개 있는 경우 테이블의 별칭 열에는 하나의 별칭과 별칭 요약(예: (n개 이상))이 표시됩니다. 별칭 요약을 선택하면 KMS 키 세부 정보 페이지의 별칭 탭으로 바로 이동합니다.

5. 별칭 탭에서 삭제할 별칭 옆의 확인란을 선택합니다. 그런 다음 삭제(Delete)를 선택합니다.

AWS KMS API 사용

별칭을 삭제하려면 DeleteAlias 작업을 사용합니다. 이 작업은 한 번에 하나의 별칭을 삭제합니다. 별칭 이름은 대/소문자를 구분하며 앞에 `alias/`접두사가 앞에 와야 합니다.

예를 들어, 다음 명령은 `test-key` 별칭을 삭제합니다. 이 명령은 출력을 반환하지 않습니다.

```
$ aws kms delete-alias --alias-name alias/test-key
```

별칭이 삭제되었는지 확인하려면 ListAliases 작업을 사용합니다. 다음 명령은의 `--query` 파라미터를 사용하여 AWS CLI `test-key`별칭만 가져옵니다. 응답의 빈 대괄호는 `ListAliases` 응답에 `test-key` 별칭이 포함되지 않았음을 나타냅니다. 대괄호를 제거하려면 `--output text` 파라미터와 값을 사용합니다.

```
$ aws kms list-aliases --query 'Aliases[?AliasName==`alias/test-key`]'
[]
```

별칭을 사용하여 KMS 키에 대한 액세스 제어

KMS 키와 연결된 별칭을 기반으로 KMS 키에 대한 액세스를 제어할 수 있습니다. 이렇게 하려면 kms:RequestAlias 및 kms:ResourceAliases 조건 키를 사용합니다. 이 기능은 속성 기반 액세스 제어(ABAC) AWS KMS 지원의 일부입니다.

`kms:RequestAlias` 조건 키는 요청의 별칭을 기반으로 KMS 키에 대한 액세스를 허용하거나 거부합니다. `kms:ResourceAliases` 조건 키는 KMS 키와 연결된 별칭을 기반으로 KMS 키에 대한 액세스를 허용하거나 거부합니다.

이러한 기능을 사용하면 정책문의 `resource` 요소에 별칭을 사용하여 KMS 키를 식별할 수 없습니다. 별칭이 `resource` 요소의 값인 경우 정책은 별칭 리소스와 연결될 수 있는 KMS 키가 아니라 별칭 리소스에 적용됩니다.

> **Note**
>
> 태그 및 별칭 변경으로 KMS 키 인증에 영향을 미치는 데 최대 5분이 소요될 수 있습니다. 최근 변경 사항은 권한 부여에 영향을 미치기 전에 API 작업에서 볼 수 있습니다.

별칭을 사용하여 KMS 키에 대한 액세스를 제어할 때는 다음 사항을 고려해야 합니다.

- 최소 권한 액세스를 강화하는 최고의 방식을 별칭을 사용하는 것입니다. IAM 보안 주체에 사용하거나 관리해야 하는 KMS 키에만 필요한 권한만 부여합니다. 예를 들어 별칭을 사용하여 프로젝트에 사용되는 KMS 키를 식별합니다. 그런 다음 프로젝트 팀에 프로젝트 별칭과 함께 KMS 키만 사용할 수 있는 권한을 부여합니다.
- 보안 주체에게 별칭을 추가, 편집 및 삭제할 수 있는 `kms:CreateAlias`, `kms:UpdateAlias` 또는 `kms:DeleteAlias` 권한을 부여할 때는 주의해야 합니다. 별칭을 사용하여 KMS 키에 대한 액세스를 제어하는 경우 별칭을 변경하면 보안 주체에게 사용 권한이 없는 KMS 키를 사용할 수 있는 권한이 부여될 수도 있습니다. 또한 다른 보안 주체가 작업을 수행하는 데 필요한 KMS 키에 대한 액세스를 거부할 수도 있습니다.
- 현재 별칭을 관리할 권한이 AWS 계정 있는의 보안 주체를 검토하고 필요한 경우 권한을 조정합니다. 키 정책을 변경하거나 권한 부여를 생성할 권한이 없는 키 관리자는 별칭을 관리할 권한이 있는 경우 KMS 키에 대한 액세스를 제어할 수 있습니다.

  예를 들어, 키 관리자에 대한 콘솔 기본 키 정책에는 `kms:CreateAlias`, `kms:DeleteAlias` 및 `kms:UpdateAlias` 권한이 포함됩니다. IAM 정책은 AWS 계정의 모든 KMS 키에 대한 별칭 권한을 부여할 수 있습니다. 예를 들어, AWSKeyManagementServicePowerUser 관리형 정책은 보안 주체가 모든 KMS 키에 대한 별칭을 생성, 삭제 및 나열하도록 허용하지만 업데이트할 수는 없습니다.
- 별칭에 의존하는 정책을 설정하기 전에에서 KMS 키의 별칭을 검토합니다 AWS 계정. 포함하려는 별칭에만 정책을 적용해야 합니다. CloudTrail 로그 및 CloudWatch 경보를 사용하여 KMS 키에 대한 액세스에 영향을 줄 수 있는 별칭 변경 사항을 알립니다. 또한, ListAliases 응답에는 각 별칭에 대한 생성 날짜 및 최종 업데이트 날짜가 포함됩니다.
- 별칭 정책 조건은 패턴 일치를 사용하며 별칭의 특정 인스턴스에 연결되어 있지 않습니다. 별칭 기반 조건 키를 사용하는 정책은 패턴과 일치하는 모든 새 별칭과 기존 별칭에 영향을 줍니다. 정책 조건과 일치하는 별칭을 삭제했다가 다시 만들면 이전 별칭과 마찬가지로 새 별칭에도 조건이 적용됩니다.

kms:RequestAlias 조건 키는 작업 요청에 명시적으로 지정된 별칭에 따라 달라집니다. kms:ResourceAliases 조건 키는 요청에 나타나지 않더라도 KMS 키와 연결된 별칭에 따라 달라집니다.

kms:RequestAlias

요청에서 KMS 키를 식별하는 별칭을 기반으로 KMS 키에 대한 액세스를 허용하거나 거부합니다. 키 정책 또는 IAM 정책에서 kms:RequestAlias 조건 키를 사용할 수 있습니다. 별칭을 사용하여 요청에서 KMS 키를 식별하는 작업, 즉, 암호화 작업, DescribeKey 및 GetPublicKey에 적용됩니다. CreateAlias 또는 DeleteAlias와 같은 별칭 작업에는 유효하지 않습니다.

> **Important**
>
> 호출자는 키 ID, 키 ARN 또는 다른 별칭을 사용하여 요청에서 KMS 키를 식별하여 이를 우회할 수 있으므로 kms:RequestAlias 조건이 있는 Deny 정책 설명은 권장되지 않습니다. KMS 키와 연결된 별칭을 기반으로 액세스를 거부하려면 대신 kms:ResourceAliases 조건 키를 사용합니다.

조건 키에서 별칭 이름 또는 별칭 이름 패턴을 지정합니다. 별칭 ARN을 지정할 수 없습니다.

예를 들어 다음 키 정책문은 보안 주체가 KMS 키에 대해 지정된 작업을 사용할 수 있도록 허용합니다. 권한은 요청에서 alpha가 포함된 별칭을 사용하여 KMS 키를 식별하는 경우에만 유효합니다.

```
{
  "Sid": "Key policy using a request alias condition",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/alpha-developer"
  },
  "Action": [
    "kms:Decrypt",
    "kms:GenerateDataKey*",
    "kms:DescribeKey"
  ],
  "Resource": "*",
  "Condition": {
    "StringLike": {
      "kms:RequestAlias": "alias/*alpha*"
    }
  }
```

```
}
```

권한이 부여된 보안 주체의 다음 예제 요청은 조건을 충족합니다. 그러나, 키 ID, 키 ARN 또는 다른 별칭을 사용한 요청은 이러한 값이 동일한 KMS 키를 식별하더라도 조건을 충족하지 않습니다.

```
$ aws kms describe-key --key-id "arn:aws:kms:us-west-2:111122223333:alias/project-
alpha"
```

kms:ResourceAliases

별칭이 요청에 사용되지 않더라도 KMS 키와 연결된 별칭을 기반으로 KMS 키에 대한 액세스를 허용하거나 거부합니다. kms:ResourceAliases 조건 키를 사용하면 별칭 또는 별칭 패턴(예: `alias/test*`)을 지정할 수 있으므로 IAM 정책에서 이를 사용하여 동일한 리전의 여러 KMS 키에 대한 액세스를 제어할 수 있습니다. KMS 키를 사용하는 모든 AWS KMS 작업에 유효합니다.

예를 들어, 다음 IAM 정책은 보안 주체가 두 개의 KMS 키에 대해 지정된 작업을 호출하도록 허용합니다 AWS 계정. 그러나 권한은 `restricted`로 시작하는 별칭과 연결된 KMS 키에만 적용됩니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AliasBasedIAMPolicy",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": [
        "arn:aws:kms:*:111122223333:key/*",
        "arn:aws:kms:*:444455556666:key/*"
      ],
      "Condition": {
        "ForAnyValue:StringLike": {
          "kms:ResourceAliases": "alias/restricted*"
        }
      }
```

```
    }
  ]
}
```

kms:ResourceAliases 조건은 조건은 요청이 아니라 리소스의 조건입니다. 따라서 별칭을 지정하지 않은 요청은 여전히 조건을 충족할 수 있습니다.

일치하는 별칭을 지정하는 다음 예제 요청은 조건을 충족합니다.

```
$ aws kms enable-key-rotation --key-id "alias/restricted-project"
```

그러나 다음 예제 요청은 해당 별칭이 요청에 사용되지 않더라도 지정된 KMS 키에 restricted로 시작하는 별칭이 있는 경우 조건을 충족합니다.

```
$ aws kms enable-key-rotation --key-id "1234abcd-12ab-34cd-56ef-1234567890ab"
```

애플리케이션에서 별칭을 사용하는 방법 알아보기

별칭을 사용하여 애플리케이션 코드에서 KMS 키를 나타낼 수 있습니다. KeyId 파라미터 AWS KMS 암호화 작업인 DescribeKey 및 GetPublicKey는 별칭 이름 또는 별칭 ARN을 허용합니다.

예를 들어 다음 GenerateDataKey 명령은 별칭 이름(alias/finance)를 사용하여 KMS 키를 식별합니다. 별칭 이름은 KeyId 파라미터 값입니다.

```
$ aws kms generate-data-key --key-id alias/finance --key-spec AES_256
```

KMS 키가 다른에 AWS 계정있는 경우 이러한 작업에서 키 ARN 또는 별칭 ARN을 사용해야 합니다. 별칭 ARN 사용하는 경우 KMS 키의 별칭은 KMS 키를 소유하는 계정에 정의되어 있으며 리전마다 다를 수 있습니다. 별칭 ARN을 찾는 방법에 대한 도움말은 KMS 키의 별칭 이름 및 별칭 ARN 찾기 섹션을 참조하세요.

예를 들어 다음 GenerateDataKey 명령은 호출자의 계정에 없는 KMS 키를 사용합니다. ExampleAlias 별칭은 지정된 계정 및 리전의 KMS 키와 연결되어 있습니다.

```
$ aws kms generate-data-key --key-id arn:aws:kms:us-west-2:444455556666:alias/
ExampleAlias --key-spec AES_256
```

별칭의 가장 강력한 용도 중 하나는 여러 AWS 리전에서 실행되는 애플리케이션에서 사용하는 것입니다. 예를 들어 서명 및 확인을 위해 RSA 비대칭 KMS 키를 사용하는 글로벌 애플리케이션이 있을 수 있습니다.

- 미국 서부(오레곤)(us-west-2)에서는 `arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`를 사용하려고 합니다.
- 유럽(프랑크푸르트)(eu-central-1)에서는 `arn:aws:kms:eu-central-1:111122223333:key/0987dcba-09fe-87dc-65ba-ab0987654321`을 사용하려고 합니다.
- 아시아 태평양(싱가포르)(ap-southeast-1)에서는 `arn:aws:kms:ap-southeast-1:111122223333:key/1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d`를 사용하려고 합니다.

각 리전에서 다른 버전의 애플리케이션을 만들거나 사전 또는 switch 문을 사용하여 각 리전에 적합한 KMS 키를 선택할 수 있습니다. 그러나 각 리전에 동일한 별칭 이름으로 별칭을 만드는 것이 훨씬 쉽습니다. 별칭 이름은 대/소문자를 구분합니다.

```
aws --region us-west-2 kms create-alias \
    --alias-name alias/new-app \
    --key-id arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab

aws --region eu-central-1 kms create-alias \
    --alias-name alias/new-app \
    --key-id arn:aws:kms:eu-central-1:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321

aws --region ap-southeast-1 kms create-alias \
    --alias-name alias/new-app \
    --key-id arn:aws:kms:ap-
southeast-1:111122223333:key/1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d
```

그런 다음 코드에서 별칭을 사용합니다. 각 리전에서 코드가 실행되는 경우 별칭은 해당 리전의 연결된 KMS 키를 참조합니다. 예를 들어, 이 코드는 별칭 이름으로 Sign 작업을 호출합니다.

```
aws kms sign --key-id alias/new-app \
    --message $message \
    --message-type RAW \
```

```
    --signing-algorithm RSASSA_PSS_SHA_384
```

그러나 별칭이 삭제되거나 다른 KMS 키와 연결되도록 업데이트될 위험이 있습니다. 이 경우 애플리케이션에서 별칭 이름을 사용하여 서명을 확인하려는 시도가 실패하므로 별칭을 다시 만들거나 업데이트해야 할 수 있습니다.

이 위험을 완화하려면 보안 주체에 애플리케이션에서 사용하는 별칭을 관리할 수 있는 권한을 부여할 때 주의하세요. 자세한 내용은 별칭에 대한 액세스 제어 섹션을 참조하세요.

AWS Encryption SDK를 비롯해 애플리케이션에서 여러 AWS 리전의 데이터를 암호화하는 몇 가지 다른 솔루션이 있습니다.

AWS CloudTrail 로그에서 별칭 찾기

별칭을 사용하여 AWS KMS API 작업 AWS KMS key 에서를 나타낼 수 있습니다. 이렇게 하면 KMS 키의 별칭과 키 ARN이 이벤트의 AWS CloudTrail 로그 항목에 기록됩니다. 별칭이 `requestParameters` 필드에 나타납니다. 키 ARN이 `resources` 필드에 나타납니다. 이는 AWS 서비스가 계정 AWS 관리형 키 에서를 사용하는 경우에도 마찬가지입니다.

예를 들어 다음 GenerateDataKey 요청은 `project-key` 별칭을 사용하여 KMS 키를 나타냅니다.

```
$ aws kms generate-data-key --key-id alias/project-key --key-spec AES_256
```

이 요청이 CloudTrail 로그에 기록되면 로그 항목에는 사용된 실제 KMS 키의 별칭과 키 ARN이 모두 포함됩니다.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "ABCDE",
        "arn": "arn:aws:iam::111122223333:role/ProjectDev",
        "accountId": "111122223333",
        "accessKeyId": "FFHIJ",
        "userName": "example-dev"
    },
    "eventTime": "2020-06-29T23:36:41Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
    "awsRegion": "us-west-2",
```

```
    "sourceIPAddress": "205.205.123.000",
    "userAgent": "aws-cli/1.18.89 Python/3.6.10
 Linux/4.9.217-0.1.ac.205.84.332.metal1.x86_64 botocore/1.17.12",
    "requestParameters": {
        "keyId": "alias/project-key",
        "keySpec": "AES_256"
    },
    "responseElements": null,
    "requestID": "d93f57f5-d4c5-4bab-8139-5a1f7824a363",
    "eventID": "d63001e2-dbc6-4aae-90cb-e5370aca7125",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

CloudTrail 로그의 로깅 AWS KMS 작업에 대한 자세한 내용은 섹션을 참조하세요를 사용하여 AWS KMS API 호출 로깅 AWS CloudTrail.

의 태그 AWS KMS

태그는 AWS 리소스에 할당(또는 AWS 할당)할 수 있는 선택적 메타데이터 레이블입니다. 각 태그는태그 키 및 태그 값으로 구성되며, 둘 다 대소 문자를 구분하는 문자열입니다. 태그 값은 빈(null) 문자열일 수도 있습니다. 리소스의 각 태그에는 서로 다른 태그 키가 있어야 하지만 여러 AWS 리소스에 동일한 태그를 추가할 수 있습니다. 각 리소스에는 최대 50개의 사용자 생성 태그가 포함될 수 있습니다.

태그 키 또는 태그 값에 기밀 또는 민감한 정보를 포함하지 마세요. 태그는 결제를 AWS 서비스포함하여 많은 사용자가 액세스할 수 있습니다.

에서는 KMS 키를 생성할 때 고객 관리형 키에 태그를 추가하고 삭제 보류 중이 아닌 한 기존 KMS 키에 태그를 지정하거나 태그를 해제할 AWS KMS수 있습니다. 다른 에서는 별칭, 사용자 지정 키 스토어 AWS 관리형 키AWS 소유 키또는 KMS 키에 태그를 지정할 수 없습니다 AWS 계정. 태그는 선택 사항이지만 매우 유용할 수 있습니다.

예를 들어 Alpha 프로젝트에 사용하는 모든 KMS 키와 Amazon S3 버킷에 `"Project"="Alpha"` 태그를 추가할 수 있습니다.

```
TagKey   = "Project"
TagValue = "Alpha"
```

형식 및 구문을 포함한 태그에 대한 일반적인 내용은의 AWS 리소스 태그 지정을 참조하세요Amazon Web Services 일반 참조.

태그는 다음을 지원합니다.

- AWS 리소스를 식별하고 구성합니다. 많은 AWS 서비스가 태그 지정을 지원하므로 서로 다른 서비스의 리소스에 동일한 태그를 할당하여 리소스가 관련이 있음을 나타낼 수 있습니다. 예를 들어 KMS 키와 Amazon Elastic Block Store(Amazon EBS) 볼륨 또는 AWS Secrets Manager 보안 암호에 동일한 태그를 할당할 수 있습니다. 태그를 사용하여 자동화를 위해 KMS 키를 식별할 수도 있습니다.
- AWS 비용을 추적합니다. AWS 리소스에 태그를 추가하면는 태그별로 집계된 사용량 및 비용이 포함된 비용 할당 보고서를 AWS 생성합니다. 이 기능을 사용하여 프로젝트, 애플리케이션 또는 비용 센터의 AWS KMS 비용을 추적할 수 있습니다.

  비용 할당 태그 사용에 대한 자세한 내용은 AWS Billing 사용자 설명서의 비용 할당 태그 사용을 참조하세요. 태그 키 및 태그 값에 대한 규칙에 대한 자세한 내용은 AWS Billing 사용 설명서의 사용자 정의 태그 제한을 참조하세요.

- AWS 리소스에 대한 액세스를 제어합니다. 태그를 기반으로 KMS 키에 대한 액세스를 허용하고 거부하는 것은 속성 기반 액세스 제어(ABAC) AWS KMS 지원의 일부입니다. 태그를 AWS KMS keys 기반으로에 대한 액세스를 제어하는 방법에 대한 자세한 내용은 섹션을 참조하세요태그를 사용하여 KMS 키에 대한 액세스 제어. 태그를 사용하여 AWS 리소스에 대한 액세스를 제어하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 AWS 리소스 태그를 사용하여 리소스에 대한 액세스 제어를 참조하세요.

AWS KMS 는 TagResource, UntagResource 또는 ListResourceTags 작업을 사용할 때 AWS CloudTrail 로그에 항목을 기록합니다.

주제

- 태그에 대한 액세스 제어
- KMS 키에 태그 추가
- KMS 키와 연결된 태그 편집
- KMS 키와 연결된 태그 제거
- KMS 키와 연결된 태그 보기
- 태그를 사용하여 KMS 키에 대한 액세스 제어

태그에 대한 액세스 제어

AWS KMS 콘솔에서 또는 API를 사용하여 태그를 추가, 확인 및 삭제하려면 보안 주체에 태그 지정 권한이 필요합니다. 키 정책에서 이러한 권한을 제공할 수 있습니다. IAM 정책(VPC 엔드포인트 정책 포함)에서도 이를 제공할 수 있지만, 이는 키 정책에서 허용하는 경우에만 가능합니다. AWSKeyManagementServicePowerUser 관리형 정책은 보안 주체가 계정이 액세스할 수 있는 모든 KMS 키에 대해 태그를 지정하고 태그를 해제하고 나열할 수 있도록 허용합니다.

태그에 AWS 전역 조건 키를 사용하여 이러한 권한을 제한할 수도 있습니다. AWS KMS에서 이러한 조건은 태그 지정 작업(예: TagResource 및 UntagResource)에 대한 액세스를 제어할 수 있습니다.

> **Note**
>
> 보안 주체에게 태그 및 별칭을 관리할 수 있는 권한을 부여하는 데 주의해야 합니다. 태그나 별칭을 변경하면 고객 관리형 키의 권한을 허용하거나 거부할 수 있습니다. 자세한 내용은 에 대한 ABAC AWS KMS 및 태그를 사용하여 KMS 키에 대한 액세스 제어 섹션을 참조하세요.

예제 정책과 자세한 내용은 IAM 사용 설명서의 태그 키를 기반으로 액세스 제어 섹션을 참조하십시오.

태그를 만들고 관리할 수 있는 권한은 다음과 같습니다.

kms:TagResource

보안 주체가 태그를 추가하거나 편집할 수 있습니다. KMS 키를 생성하는 동안 태그를 추가하려면 보안 주체에 특정 KMS 키로 제한되지 않는 IAM 정책에 대한 권한이 있어야 합니다.

kms:ListResourceTags

보안 주체가 KMS 키의 태그를 볼 수 있도록 허용합니다.

kms:UntagResource

보안 주체가 KMS 키에서 태그를 삭제할 수 있도록 허용합니다.

정책에서 태그 지정 권한

키 정책 또는 IAM 정책에서 태그 지정 권한을 제공할 수 있습니다. 예를 들어 다음 예제 키 정책은 KMS 키에 대한 태그 지정 권한을 사용자에게 제공합니다. 예를 들어 관리자 또는 개발자 역할로 가정할 수 있는 모든 사용자에게 태그를 볼 수 있는 권한을 제공합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Id": "example-key-policy",
  "Statement": [
    {
      "Sid": "EnableIAMUserPermissions",
      "Effect": "Allow",
      "Principal": {"AWS": "arn:aws:iam::111122223333:root"},
      "Action": "kms:*",
      "Resource": "*"
    },
    {
      "Sid": "AllowAllTaggingPermissions",
      "Effect": "Allow",
      "Principal": {"AWS": [
        "arn:aws:iam::111122223333:user/LeadAdmin",
        "arn:aws:iam::111122223333:user/SupportLead"
      ]},
```

```
      "Action": [
          "kms:TagResource",
          "kms:ListResourceTags",
          "kms:UntagResource"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowRolesViewTags",
      "Effect": "Allow",
      "Principal": {"AWS": [
        "arn:aws:iam::111122223333:role/Administrator",
        "arn:aws:iam::111122223333:role/Developer"
      ]},
      "Action": "kms:ListResourceTags",
      "Resource": "*"
    }
  ]
}
```

보안 주체에게 여러 KMS 키에 대한 태그 지정 권한을 부여하려면 IAM 정책을 사용할 수 있습니다. 이 정책이 유효하려면 각 KMS 키의 키 정책으로 인해 계정이 IAM 정책을 사용하여 KMS 키에 대한 액세스를 제어할 수 있어야 합니다.

예를 들어, 다음 IAM 정책은 보안 주체가 KMS 키를 생성할 수 있도록 허용합니다. 또한 지정된 계정의 모든 KMS 키에 태그를 만들고 관리할 수 있습니다. 이 조합을 통해 보안 주체가 CreateKey 작업의 태그 파라미터를 사용하여 KMS 키를 만드는 동안 KMS 키에 태그를 추가할 수 있습니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "IAMPolicyCreateKeys",
      "Effect": "Allow",
      "Action": "kms:CreateKey",
      "Resource": "*"
    },
    {
      "Sid": "IAMPolicyTags",
```

```
      "Effect": "Allow",
      "Action": [
        "kms:TagResource",
        "kms:UntagResource",
        "kms:ListResourceTags"
      ],
      "Resource": "arn:aws:kms:*:111122223333:key/*"
    }
  ]
}
```

태그 지정 권한 제한

정책 조건을 사용하여 태그 지정 권한을 제한할 수 있습니다. 다음 정책 조건을 kms:TagResource 및 kms:UntagResource 권한에 적용할 수 있습니다. 예를 들어, aws:RequestTag/*tag-key* 조건을 사용하여 보안 주체가 특정 태그만 추가하거나 보안 주체가 특정 태그 키를 사용하여 태그를 추가하지 못하도록 할 수 있습니다. 또는 kms:KeyOrigin 조건을 사용하여 보안 주체가 가져온 키 구성 요소가 있는 KMS 키에 태그를 지정하거나 태그를 해제하지 못하도록 할 수 있습니다.

- aws:RequestTag
- aws:ResourceTag/tag-key(IAM 정책만 해당)
- aws:TagKeys
- kms:CallerAccount
- kms:KeySpec
- kms:KeyUsage
- kms:KeyOrigin
- kms:ViaService

태그를 사용하여 KMS 키에 대한 액세스를 제어할 때 가장 좋은 방법은 aws:RequestTag/*tag-key* 또 는aws:TagKeys 조건 키를 사용하여 허용되는 태그 (또는 태그 키)를 결정하는 것입니다.

예를 들어 다음 IAM 정책은 이전 것과 비슷합니다. 그러나 이 정책은 보안 주체가 Project 태그 키가 있는 태그에 대해서만 태그(TagResource)를 생성하고 태그 UntagResource를 삭제할 수 있도록 허용합니다.

왜냐하면 TagResource 및 UntagResource 요청에는 여러 태그가 포함될 수 있으므로 ForAllValues 또는 ForAnyValue 집합 연산자와 aws:TagKeys 조건을 지정해야 합니다.

ForAnyValue 연산자를 사용하려면 요청의 태그 키 중 적어도 하나가 정책의 태그 키 중 하나와 일치해야 합니다. ForAllValues 연산자를 사용하려면 요청의 모든 태그 키가 정책의 태그 키 중 하나와 일치해야 합니다. ForAllValues 연산자는 요청에 태그가 없는 경우에도 true를 반환하지만 태그가 지정되지 않으면 TagResource 및 UntagResource가 실패합니다. 집합 연산자에 대한 자세한 내용은 IAM 사용 설명서의 여러 키 및 값 사용을 참조하십시오.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "IAMPolicyCreateKey",
      "Effect": "Allow",
      "Action": "kms:CreateKey",
      "Resource": "*"
    },
    {
      "Sid": "IAMPolicyViewAllTags",
      "Effect": "Allow",
      "Action": "kms:ListResourceTags",
      "Resource": "arn:aws:kms:*:111122223333:key/*"
    },
    {
      "Sid": "IAMPolicyManageTags",
      "Effect": "Allow",
      "Action": [
        "kms:TagResource",
        "kms:UntagResource"
      ],
      "Resource": "arn:aws:kms:*:111122223333:key/*",
      "Condition": {
          "ForAllValues:StringEquals": {"aws:TagKeys": "Project"}
      }
    }
  ]
}
```

KMS 키에 태그 추가

태그는 AWS 리소스를 식별하고 구성하는 데 도움이 됩니다. KMS 키를 생성할 때 고객 관리형 키에 태그를 추가하거나 기존 KMS 키에 태그를 추가할 수 있습니다. 태그를 지정할 수 없습니다 AWS 관리형 키.

다음 절차에서는 콘솔 및 AWS KMS API를 사용하여 고객 관리형 키에 AWS KMS 태그를 추가하는 방법을 보여줍니다. AWS KMS API 예제에서는 AWS Command Line Interface (AWS CLI)를 사용하지만 지원되는 모든 프로그래밍 언어를 사용할 수 있습니다.

주제

- KMS 키 생성 중 태그 추가
- 기존 KMS 키에 태그 추가

KMS 키 생성 중 태그 추가

AWS KMS 콘솔 또는 CreateKey 작업을 사용하여 키를 생성할 때 KMS 키에 태그를 추가할 수 있습니다. KMS 키를 생성할 때 태그를 추가하려면 KMS 키를 생성하는 데 필요한 권한 외에 IAM 정책에 `kms:TagResource` 권한이 있어야 합니다. 최소한 권한에 계정 및 리전의 모든 KMS 키가 포함되어야 합니다. 자세한 내용은 태그에 대한 액세스 제어을 참조하세요.

AWS KMS 콘솔 사용

콘솔에서 KMS 키를 생성할 때 태그를 추가하려면 콘솔에서 KMS 키에 태그를 추가하고 KMS 키를 생성하는 데 필요한 권한 외에 KMS 키를 보는 데 필요한 권한이 있어야 합니다. 최소한 권한에 계정 및 리전의 모든 KMS 키가 포함되어야 합니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 탐색 창에서 고객 관리형 키를 선택합니다. (AWS 관리형 키의 태그를 관리할 수 없습니다.)
4. 키 유형을 선택한 다음 다음(Next)를 선택합니다.
5. 별칭과 설명(선택 사항)을 입력합니다.
6. 태그 키와 태그 값(선택 사항)을 입력합니다. 태그를 추가하려면 태그 추가를 선택합니다. 태그를 삭제하려면 제거(Remove)를 선택합니다. 새 KMS 키에 태그를 지정했으면 다음(Next)를 선택합니다.
7. KMS 키 만들기를 마칩니다.

AWS KMS API 사용

CreateKey 작업을 사용하여 키를 생성할 때 태그를 지정하려면 이 작업의 Tags 파라미터를 사용하세요.

CreateKey 의 값은Tags 파라미터 값은 대소문자를 구분하는 태그 키 및 태그 값 페어 모음입니다. KMS 키의 각 태그에는 다른 태그 이름이 있어야 합니다. 태그 값은 null이거나 빈 문자열일 수 있습니다.

예를 들어 다음 AWS CLI 명령은 Project:Alpha 태그가 있는 대칭 암호화 KMS 키를 생성합니다. 두 개 이상의 키-값 페어를 지정할 때는 공백을 사용하여 각 페어를 구분합니다.

```
$ aws kms create-key --tags TagKey=Project,TagValue=Alpha
```

이 명령이 성공하면 새 KMS 키에 대한 정보가 있는 KeyMetadata 객체를 반환합니다. 그러나 KeyMetadata에는 태그가 포함되지 않습니다. 태그를 가져오려면 ListResourceTags 작업을 사용합니다.

기존 KMS 키에 태그 추가

AWS KMS 콘솔에서 또는 TagResource 작업을 사용하여 기존 고객 관리형 KMS 키에 태그를 추가할 수 있습니다. 태그를 추가하려면 KMS 키에 대한 태그 지정 권한이 필요합니다. KMS 키의 키 정책이나, 키 정책에서 허용하는 경우 KMS 키가 포함된 IAM 정책에서 이 권한을 얻을 수 있습니다.

AWS KMS 콘솔 사용

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 탐색 창에서 고객 관리형 키를 선택합니다. (AWS 관리형 키의 태그를 관리할 수 없습니다.)
4. 테이블 필터를 사용하여 특정 태그가 있는 KMS 키만 표시할 수 있습니다. 자세한 내용은 AWS KMS 콘솔을 사용하여 태그 보기를 참조하세요.
5. KMS 키의 별칭 옆의 확인란을 선택합니다.
6. [Key actions], [Add or edit tags]를 선택합니다.
7. KMS 키의 세부 정보 페이지에서 태그 탭을 선택합니다.
   - 첫 번째 태그를 생성하려면 태그 생성을 선택하고 태그 키(필수)와 태그 값(선택 사항)을 입력한 다음, 저장을 선택합니다.

태그 값을 비워 두면 실제 태그 값은 null 또는 빈 문자열입니다.

- 태그를 추가하려면 편집을 선택하고 태그 추가를 선택하고 태그 키와 태그 값을 입력한 다음, 저장을 선택합니다.

8. 변경 사항을 저장하려면 변경 사항 저장을 선택합니다.

AWS KMS API 사용

TagResource 작업은 KMS 키에 하나 이상의 태그를 추가합니다. 이 작업을 사용하여 다른 AWS 계정의 태그를 추가할 수 없습니다. TagResource 작업을 사용하여 기존 태그를 편집할 수도 있습니다. 자세한 내용은 the section called “태그 편집” 단원을 참조하십시오.

태그를 추가하려면 새 태그 키와 태그 값을 지정합니다. KMS 키의 각 태그에는 다른 태그 키가 있어야 합니다. 태그 값은 null이거나 빈 문자열일 수 있습니다.

예를 들어 다음 명령은 예제 KMS 키에 **Purpose** 및 **Department** 태그를 추가합니다.

```
$ aws kms tag-resource \
        --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
        --tags TagKey=Purpose,TagValue=Pretest TagKey=Department,TagValue=Finance
```

이 명령이 제대로 실행되면 메타데이터를 반환하지 않습니다. KMS 키에 지정된 태그를 보려면 ListResourceTags 작업을 사용합니다.

KMS 키와 연결된 태그 편집

태그는 AWS 리소스를 식별하고 구성하는 데 도움이 됩니다. AWS KMS 콘솔에서 또는 TagResource 작업을 사용하여 고객 관리형 KMS 키와 연결된 태그를 편집할 수 있습니다. 의 태그는 편집할 수 없습니다 AWS 관리형 키.

다음 절차에서는 KMS 키와 연결된 태그를 편집하는 방법을 보여줍니다. AWS KMS API 예제들은 AWS Command Line Interface (AWS CLI)를 사용하지만, 사용자는 지원되는 프로그래밍 언어라면 무엇이든 사용할 수 있습니다.

AWS KMS 콘솔 사용

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.

2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 탐색 창에서 고객 관리형 키를 선택합니다. (의 태그는 편집할 수 없음 AWS 관리형 키)
4. 테이블 필터를 사용하여 특정 태그가 있는 KMS 키만 표시할 수 있습니다. 자세한 내용은 AWS KMS 콘솔을 사용하여 태그 보기를 참조하세요.
5. KMS 키의 별칭 옆의 확인란을 선택합니다.
6. [Key actions], [Add or edit tags]를 선택합니다.
7. KMS 키의 세부 정보 페이지에서 태그 탭을 선택합니다.
   - 태그의 이름이나 값을 변경하려면 편집을 선택하여 변경을 한 다음, 저장을 선택합니다.
8. 변경 사항을 저장하려면 변경 사항 저장을 선택합니다.

AWS KMS API 사용

TagResource 작업은 고객 관리형 키에 하나 이상의 태그를 추가합니다. 그러나 TagResource를 사용해 기존 태그의 태그 값을 변경할 수도 있습니다. 이 작업을 사용하여 다른 AWS 계정의 태그를 추가 또는 편집할 수 없습니다.

태그를 편집하려면 기존 태그 키와 새 태그 값을 지정합니다. KMS 키의 각 태그에는 다른 태그 키가 있어야 합니다. 태그 값은 null이거나 빈 문자열일 수 있습니다.

예를 들어 이 명령은 `Purpose` 태그 값을 `Pretest`에서 `Test`로 바꿉니다.

```
$ aws kms tag-resource \
        --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
        --tags TagKey=Purpose,TagValue=Test
```

KMS 키와 연결된 태그 제거

태그는 AWS 리소스를 식별하고 구성하는 데 도움이 됩니다. AWS KMS 콘솔에서 또는 UntagResource 작업을 사용하여 고객 관리형 KMS 키와 연결된 태그를 제거할 수 있습니다. 의 태그는 편집하거나 제거할 수 없습니다 AWS 관리형 키.

다음 절차에서는 KMS 키에서 태그를 제거하는 방법을 보여줍니다. AWS KMS API 예제들은 AWS Command Line Interface (AWS CLI)를 사용하지만, 사용자는 지원되는 프로그래밍 언어라면 무엇이든 사용할 수 있습니다.

AWS KMS 콘솔 사용

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 탐색 창에서 고객 관리형 키를 선택합니다. (AWS 관리형 키의 태그를 관리할 수 없습니다.)
4. 테이블 필터를 사용하여 특정 태그가 있는 KMS 키만 표시할 수 있습니다. 자세한 내용은 AWS KMS 콘솔을 사용하여 태그 보기를 참조하세요.
5. KMS 키의 별칭 옆의 확인란을 선택합니다.
6. [Key actions], [Add or edit tags]를 선택합니다.
7. KMS 키의 세부 정보 페이지에서 태그 탭을 선택합니다.
   - 태그를 삭제하려면 편집을 선택합니다. 태그 행에서 제거를 선택한 다음 저장을 선택합니다.
8. 변경 사항을 저장하려면 변경 사항 저장을 선택합니다.

AWS KMS API 사용

UntagResource 작업은 KMS 키에서 태그를 삭제합니다. 삭제할 태그를 식별하려면 태그 키를 지정합니다. 이 작업을 사용하여 다른 AWS 계정의 KMS 키에서 태그를 삭제할 수 없습니다.

성공하면 `UntagResource` 작업은 어떠한 출력도 반환하지 않습니다. 또한 지정된 태그 키가 KMS 키에서 발견되지 않으면 예외를 발생시키거나 응답을 반환하지 않습니다. 작업이 작동했는지 확인하려면 ListResourceTags 작업을 수행합니다.

예를 들어 이 명령은 KMS 키에서 **`Purpose`** 태그와 해당 값을 삭제합니다.

```
$ aws kms untag-resource --key-id 1234abcd-12ab-34cd-56ef-1234567890ab --tag-keys
 Purpose
```

KMS 키와 연결된 태그 보기

태그는 AWS 리소스를 식별하고 구성하는 데 도움이 됩니다. AWS KMS 콘솔에서 또는 ListResourceTags 작업을 사용하여 고객 관리형 KMS 키와 연결된 태그를 볼 수 있습니다.

다음 절차에서는 특정 KMS 키와 연결된 태그를 찾는 방법을 보여줍니다. AWS KMS API 예제에서는 AWS Command Line Interface (AWS CLI)를 사용하지만 지원되는 모든 프로그래밍 언어를 사용할 수 있습니다.

AWS KMS 콘솔 사용

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 탐색 창에서 고객 관리형 키를 선택합니다. (AWS 관리형 키의 태그를 관리할 수 없습니다.)
4. 테이블 필터를 사용하여 특정 태그가 있는 KMS 키만 표시할 수 있습니다.

   특정 태그가 있는 KMS 키만 표시하려면 필터 상자를 선택하고 태그 키를 선택한 다음 실제 태그 값 중에서 선택합니다. 태그 값의 전부 또는 일부를 입력할 수도 있습니다.

   결과 테이블에는 선택한 태그가 있는 모든 KMS 키가 표시됩니다. 그러나 태그를 표시하지는 않습니다. 태그를 보려면 KMS 키의 키 ID 또는 별칭을 선택하고 세부 정보 페이지에서 태그(Tags) 탭을 클릭합니다. 일반 구성 섹션 아래에 탭이 표시됩니다.

   이 필터에는 태그 키와 태그 값이 모두 필요합니다. 태그 키만 입력하거나 값만 입력하여 KMS 키를 찾을 수 없습니다. 태그 키 또는 값의 전체 또는 일부를 기준으로 태그를 필터링하려면 ListResourceTags 작업을 사용하여 태그가 지정된 KMS 키를 가져온 다음 프로그래밍 언어의 필터링 기능을 사용합니다.
5. KMS 키의 별칭 옆의 확인란을 선택합니다.
6. [Key actions], [Add or edit tags]를 선택합니다.
7. KMS 키의 세부 정보 페이지에서 태그 탭을 선택합니다.

AWS KMS API 사용

ListResourceTags 작업은 KMS 키에 지정된 태그를 가져옵니다. `KeyId` 파라미터가 필요합니다. 이 작업을 사용하여 다른 AWS 계정의 KMS 키에 있는 태그를 볼 수 없습니다.

예를 들어 다음 명령은 예제 KMS 키에 대한 태그를 가져옵니다.

```
$ aws kms list-resource-tags --key-id 1234abcd-12ab-34cd-56ef-1234567890ab

  "Truncated": false,
  "Tags": [
      {
        "TagKey": "Project",
        "TagValue": "Alpha"
     },
```

```
    {
      "TagKey": "Purpose",
      "TagValue": "Test"
    },
    {
      "TagKey": "Department",
      "TagValue": "Finance"
    }
  ]
}
```

태그를 사용하여 KMS 키에 대한 액세스 제어

KMS 키의 태그를 AWS KMS keys 기반으로에 대한 액세스를 제어할 수 있습니다. 예를 들어 보안 주체가 특정 태그가 있는 KMS 키만 활성화 및 비활성화할 수 있도록 허용하는 IAM 정책을 작성할 수 있습니다. 또는 IAM 정책을 사용하여 KMS 키에 특정 태그가 없으면 보안 주체가 암호화 작업에서 KMS 키를 사용하지 못하도록 할 수 있습니다.

이 기능은 속성 기반 액세스 제어(ABAC) AWS KMS 지원의 일부입니다. 태그를 사용하여 리소스에 AWS 대한 액세스를 제어하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 ABAC란 무엇입니까 AWS? 및 리소스 태그를 사용하여 리소스에 대한 액세스 제어를 참조하세요 AWS. ABAC와 관련된 액세스 문제를 해결하는 방법에 대한 자세한 내용은 에 대한 ABAC 문제 해결 AWS KMS 섹션을 참조하십시오.

> **Note**
>
> 태그 및 별칭 변경으로 KMS 키 인증에 영향을 미치는 데 최대 5분이 소요될 수 있습니다. 최근 변경 사항은 권한 부여에 영향을 미치기 전에 API 작업에서 볼 수 있습니다.

AWS KMS 는 KMS 키의 태그를 기반으로 KMS 키에 대한 액세스를 제어할 수 있는 aws:ResourceTag/tag-key 전역 조건 컨텍스트 키를 지원합니다. https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_condition-keys.html 여러 KMS 키가 동일한 태그를 가질 수 있으므로 이 기능을 사용하면 선택한 KMS 키 집합에 사용 권한을 적용할 수 있습니다. 태그를 변경하여 집합의 KMS 키를 쉽게 변경할 수도 있습니다.

에서 AWS KMS`aws:ResourceTag/`*`tag-key`* 조건 키는 IAM 정책에서만 지원됩니다. 하나의 KMS 키에만 적용되는 키 정책이나 ListKeys 또는 ListAliases 작업처럼 특정 KMS 키를 사용하지 않은 작업에서는 지원되지 않습니다.

태그를 사용하여 액세스를 제어하면 사용 권한을 단순하고 확장 가능하며 유연하게 관리할 수 있습니다. 그러나 제대로 설계되고 관리되지 않으면 실수로 KMS 키에 대한 액세스를 허용하거나 거부할 수 있습니다. 태그를 사용하여 액세스를 제어하는 경우 다음 방법을 고려하세요.

- 최소 권한 액세스를 강화하는 최고의 방식은 태그를 사용하는 것입니다. IAM 보안 주체에 사용하거나 관리해야 하는 KMS 키에만 필요한 권한만 부여합니다. 예를 들어 태그를 사용하여 프로젝트에 사용되는 KMS 키에 레이블을 지정합니다. 그런 다음 프로젝트 팀에 프로젝트 태그와 함께 KMS 키만 사용할 수 있는 권한을 부여합니다.
- 보안 주체에게 태그를 추가, 편집 및 삭제할 수 있는 `kms:TagResource` 및 `kms:UntagResource` 권한을 부여할 때는 주의해야 합니다. 태그를 사용하여 KMS 키에 대한 액세스를 제어하는 경우 태그를 변경하면 보안 주체에게 사용 권한이 없는 KMS 키를 사용할 수 있는 권한이 부여될 수도 있습니다. 또한 다른 보안 주체가 작업을 수행하는 데 필요한 KMS 키에 대한 액세스를 거부할 수도 있습니다. 키 정책을 변경하거나 권한 부여를 생성할 권한이 없는 키 관리자는 태그를 관리할 권한이 있는 경우 KMS 키에 대한 액세스를 제어할 수 있습니다.

  가능하면 정책 조건(예: `aws:RequestTag/`*`tag-key`* 또는 `aws:TagKeys`)을 사용하여 보안 주체의 태그 지정 권한을를 특정 KMS 키의 특정 태그 또는 태그 패턴으로 제한합니다.
- 현재 태그 지정 및 태그 해제 권한이 AWS 계정 있는의 보안 주체를 검토하고 필요한 경우 해당 보안 주체를 조정합니다. 예를 들어, 키 관리자에 대한 콘솔 기본 키 정책에는 해당 KMS 키에 대한 `kms:TagResource` 및 `kms:UntagResource` 권한이 포함됩니다. IAM 정책은 모든 KMS 키에 대한 태그 및 태그 해제 권한을 허용할 수 있습니다. 예를 들어 AWSKeyManagementServicePowerUser 관리형 정책은 보안 주체가 모든 KMS 키에 대해 태그를 지정하고 태그를 해제하고 나열할 수 있도록 허용합니다.
- 태그에 의존하는 정책을 설정하기 전에에서 KMS 키의 태그를 검토합니다 AWS 계정. 포함하려는 태그에만 정책을 적용해야 합니다. CloudTrail 로그 및 CloudWatch 경보를 사용하여 KMS 키에 대한 액세스에 영향을 줄 수 있는 태그 변경 사항을 알립니다.
- 태그 기반 정책 조건은 패턴 일치를 사용하며 태그의 특정 인스턴스에 연결되어 있지 않습니다. 태그 기반 조건 키를 사용하는 정책은 패턴과 일치하는 모든 새 태그와 기존 태그에 영향을 줍니다. 정책 조건과 일치하는 태그를 삭제했다가 다시 만들면 이전 태그와 마찬가지로 조건이 새 태그에 적용됩니다.

예를 들어 다음과 같은 IAM 정책을 살펴보십시오. 이를 통해 보안 주체는 아시아 태평양(싱가포르) 지역이고 `"Project"="Alpha"` 태그가 있는 계정의 KMS 키에서만 GenerateDataKeyWithoutPlaintext 및 Decrypt 작업을 호출할 수 있습니다. 이 정책은 예제 Alpha 프로젝트의 역할에 연결할 수 있습니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "IAMPolicyWithResourceTag",
      "Effect": "Allow",
      "Action": [
        "kms:GenerateDataKeyWithoutPlaintext",
        "kms:Decrypt"
      ],
      "Resource": "arn:aws:kms:ap-southeast-1:111122223333:key/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Project": "Alpha"
        }
      }
    }
  ]
}
```

다음 예제 IAM 정책은 보안 주체가 특정 암호화 작업에 대해 계정의 모든 KMS 키를 사용하도록 허용합니다. 그러나 보안 주체가 "Type"="Reserved" 태그가 있거나 "Type" 태그가 없는 KMS 키에서 이러한 암호화 작업을 사용하는 것은 허용되지 않습니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "IAMAllowCryptographicOperations",
      "Effect": "Allow",
      "Action": [
        "kms:Encrypt",
        "kms:GenerateDataKey*",
        "kms:Decrypt",
        "kms:ReEncrypt*"
      ],
```

```
      "Resource": "arn:aws:kms:*:111122223333:key/*"
    },
    {
      "Sid": "IAMDenyOnTag",
      "Effect": "Deny",
      "Action": [
        "kms:Encrypt",
        "kms:GenerateDataKey*",
        "kms:Decrypt",
        "kms:ReEncrypt*"
      ],
      "Resource": "arn:aws:kms:*:111122223333:key/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Type": "Reserved"
        }
      }
    },
    {
      "Sid": "IAMDenyNoTag",
      "Effect": "Deny",
      "Action": [
        "kms:Encrypt",
        "kms:GenerateDataKey*",
        "kms:Decrypt",
        "kms:ReEncrypt*"
      ],
      "Resource": "arn:aws:kms:*:111122223333:key/*",
      "Condition": {
        "Null": {
          "aws:ResourceTag/Type": "true"
        }
      }
    }
  ]
}
```

키 저장소

키 저장소는 암호화 키를 저장하고 사용하기에 안전한 장소입니다. 의 표준 키 스토어는 저장하는 키를 생성하고 관리하는 방법 AWS KMS 도 지원합니다. 생성하는 KMS 키의 암호화 키 구성 요소는 FIPS 140-3 레벨 3이 검증된 하드웨어 보안 모듈(HSMs)에서 생성되고 보호됩니다. KMS 키의 키 구성 요소는 HSMs을 암호화되지 않은 상태로 두지 않으며 키에 대한 모든 암호화 작업은 이러한 FIPS 140-3 HSMs 내에서 수행됩니다.

특수 요구 사항을 충족하기 위해는 사용자 지정 키 스토어 AWS KMS 도 지원합니다. 고객 소유의 전용 AWS CloudHSM 클러스터에서 지원하는 AWS CloudHSM 키 스토어와 외부의 HSM 또는 키 관리 시스템에서 지원하는 외부 키 스토어의 두 가지 유형이 있습니다 AWS 클라우드. 다음 섹션에서는 특정 요구 사항에 적합한 키 스토어를 선택하는 데 도움이 됩니다.

올바른 키 스토어 선택

대부분의 고객에게 AWS KMS 표준 키 스토어는 보안, 성능, 비용 및 운영 단순성의 최적의 균형을 제공합니다. AWS KMS HSMs은 글로벌, 리전, 정부 및 산업별 프로그램에 걸쳐 전 세계의 광범위한 보안 표준 및 규정 준수 인증에 대해 지속적으로 감사 및 검토됩니다. 자세한 내용은 AWS KMS 규정 준수를 참조하세요.

많은 규정 준수 요구 사항의 경우 KMS 키 정책을 통한 키 액세스에 대한 제어면 충분합니다. 키 구성 요소 자체를 제어해야 하는 경우 AWS KMS 가져온 키를 AWS KMS 표준 키 스토어와 함께 사용하는 것이 좋습니다. 이 접근 방식은 고성능 및 가용성을 포함하여 AWS KMS 인프라의 운영 이점을 유지하면서 고객 제어 키 구성 요소의 규정 준수 이점을 제공합니다.

다음 표에서는 AWS KMS 키 스토어 옵션의 상대적 운영 복잡성을 비교합니다. 모든 옵션은 똑같이 안전합니다. 차이점은 운영상 장단점입니다.

AWS KMS Key Store Options — Comparison

All options are equally secure and the difference is operational trade-offs

	AWS KMS Standard Key Store	AWS KMS Standard Key Store with Imported Key Material	AWS CloudHSM Key Store	External Key Store
Effort to manage	Low	Medium	Medium	High
Availability Risk	Low (99.999% SLA)	Low (99.999% SLA)	Medium	High
TPS Constraints	Low	Low	High	High
Latency	Low	Low	Medium	High

조직에 단일 테넌트 HSM 또는 외부에서 제어하는 HSM에서 키 구성 요소 보호를 명시적으로 요구하는 규제 요구 사항이 있는 경우에만 사용자 지정 키 스토어를 생성하는 것이 좋습니다 AWS. 하지만 이러한 요구 사항을 충족하지 않을 위험이 사용자 지정 키 스토어의 추가 비용, 운영 복잡성 및 성능 저하를 능가하는 경우에만 하나를 생성합니다.

AWS KMS 표준 키 스토어

모든 KMS 키는 표준 AWS KMS HSMs. 이 HSM 유형은 가장 확장 가능하고 가장 저렴하며 관리하기 쉬운 키 스토어를 지원하는 테넌트에 구애받지 않는 HSMs 플릿으로 생각할 수 있습니다. 서비스가 사용자를 대신하여 데이터를 암호화할 수 AWS 서비스 있도록 하나 이상의 내에서 사용할 KMS 키를 생성하는 경우 대칭 키를 생성합니다. 자체 애플리케이션 설계에 KMS 키를 사용하는 경우 대칭 암호화 키, 비대칭 키 또는 HMAC 키를 생성하도록 선택할 수 있습니다.

표준 키 스토어 옵션에서는 키를 AWS KMS 생성한 다음 서비스가 내부적으로 관리하는 키로 암호화합니다. AWS KMS 그런 다음는 내구성을 위해 설계된 시스템에 키의 암호화된 사본을 여러 개 저장합니다. 표준 키 스토어 유형에서 키 구성 요소를 생성하고 보호하면 AWS 키 스토어의 운영 부담과 비용을 최소화 AWS KMS 하면서의 확장성, 가용성 및 내구성을 최대한 활용할 수 있습니다.

특성

표준 키 스토어는 모든 AWS KMS 기능을 지원합니다.

운영

표준 키 스토어는 지연 시간이 가장 짧고 초당 트랜잭션(TPS) 속도가 가장 높으므로 처리량이 많은 애플리케이션 및 AWS 서비스 통합에 적합합니다.는 표준 키 스토어의 키 가용성에 대해 99.999%의 퍼블릭 SLA를 AWS KMS 제공하고 워크로드 수요에 맞게 자동 성능 조정을 제공합니다.

책임 모델

AWS 는 모든 인프라, 규모 조정 및 유지 관리를 관리합니다. 따라서 완전한 확장성, 가용성 및 내구성과 함께 운영 부담이 없습니다. 각 키를 언제, 어디서, 누가, 어떻게 사용할 수 있는지를 제어하는 IAM 및 KMS 키 정책에 대한 책임은 사용자에게 있습니다. 또한 키 태그, 별칭 및 교체, 키 활성화 또는 비활성화, 삭제 예약과 같은 수명 주기 이벤트에 대한 책임이 있습니다.

AWS KMS 가져온 키 구성 요소가 있는 표준 키 스토어

표준 키 스토어에서 KMS 키는 FIPS 검증 AWS KMS HSMs을 사용하여 키 구성 요소를 생성합니다. 그러나 키 구성 요소 없이 KMS 키를 생성하고 자체 키를 가져올 수 있습니다. 때로는 자체 키 가져오기(BYOK)라고도 합니다. 이를 통해 키 구성 요소가 생성되는 방식, 사용 가능한 기간 AWS KMS및 삭제 시점을 제어할 수 있습니다. 가져오기 또는 호출 시 만료 시간을 설정`DeleteImportedKeyMaterial`하여 액세스를 즉시 취소할 수 있습니다.

키 구성 요소 가져오기는 특정 시스템 또는 엔트로피 소스를 사용하여 키를 생성하거나, 백업 복사본을 외부에 보관하거나 AWS, 정의된 일정에 따라 키 구성 요소를 제거해야 하는 경우에 유용합니다.

특성

가져온 키 구성 요소는 자동 키 교체 및 포스트 양자(ML-DSA) 키를 제외한 모든 AWS KMS 기능을 지원합니다.

운영

가져온 키 구성 요소가 있는 키는 성능 측면에서 표준 키 스토어의 키와 구별할 수 없으므로 처리량이 많은 애플리케이션 및 AWS 서비스 통합에 적합합니다.는 워크로드 수요에 맞게 가져온 키 구성 요소가 있는 키의 가용성과 자동 성능 조정에 대해 99.999%의 퍼블릭 SLA를 AWS KMS 제공합니다.

책임 모델

앞서 설명한 표준 키 스토어 책임 외에도 가져온 키 구성 요소의 내구성에 대한 책임은 사용자에게 있습니다.는 가용성을 위해 인프라 및 규모 조정을 AWS 관리합니다. `DeleteImportedKeyMaterial`

API를 사용하여 키 구성 요소를 즉시 삭제할 수 있습니다. KMS 키와 연결된 키 구성 요소를 삭제하면 해당 키와 연결된 키 구성 요소를 다시 가져올 때까지 암호화 작업에 사용할 수 없게 됩니다.

AWS KMS 사용자 지정 키 스토어

키 구성 요소를 저장하고 보호하는 HSMs을 직접 소유하고 관리해야 하는 경우 사용자 지정 키 스토어를 생성할 수 있습니다. 사용자 지정 키 스토어는 AWS KMS사용자가 소유하고 관리하는 외부의 키 관리자가 지원하는 내의 키 스토어 AWS KMS 입니다. 사용자 지정 키 스토어는의 익숙 AWS KMS 한 키 관리 인터페이스를 자체 HSMs. 사용자 지정 키 스토어에서 KMS 키를 사용하는 경우 암호화 작업은 HSM 또는 키 관리자 내에 유지되는 키 구성 요소를 사용하여 수행됩니다.

사용자 지정 키 스토어는 표준 키 스토어보다 안전하지 않지만 관리 및 비용에 미치는 영향은 다릅니다 (이상). 표준 키 스토어를 사용하든 사용자 지정 키 스토어를 사용하든 관계없이이 서비스는 AWS 직원을 포함한 누구도 권한 없이 일반 텍스트 키를 검색하거나 사용할 수 없도록 설계되었습니다.

AWS KMS 는 AWS CloudHSM 키 스토어와 외부 키 스토어라는 두 가지 유형의 사용자 지정 키 스토어를 지원합니다.

지원되는 기능

사용자 지정 키 스토어는 대칭 암호화 KMS 키만 지원합니다. 비대칭 키, HMAC 키, 자동 키 교체, 다중 리전 키, 가져온 키 구성 요소와 같은 다른 AWS KMS 기능은 사용자 지정 키 스토어에서 사용할 수 없습니다.

AWS CloudHSM 키 스토어

AWS 는 원래 고객이 온프레미스 HSMs에서 클라우드의 단일 테넌트 HSMs 되도록 AWS CloudHSM 제품을 출시했습니다. 확장을 통해 AWS KMS 나중에 동일한 고객이 자체 애플리케이션에서를 계속 사용하고를 통해 AWS 서비스 통합할 수 있도록 사용자 지정 키 스토어를 도입 AWS CloudHSM 했습니다 AWS KMS.

소유 및 관리하는 AWS CloudHSM 클러스터에서 AWS CloudHSM 키가 생성, 저장 및 사용되는 키 스토어를 AWS KMS 사용하도록를 구성할 수 있습니다. 에 대한 요청은 AWS CloudHSM 클러스터로 AWS KMS 전달됩니다. AWS CloudHSM 클러스터는 내부에 호스팅되지만 사용자가 소유하고 관리하는 단일 테넌트 HSM 인스턴스를 AWS제공합니다. AWS CloudHSM 클러스터는 외부에서 관리되므로 AWS KMS AWS CloudHSM 키 스토어는 AWS CloudHSM 클러스터에 따라 가용성, 내구성 및 성능 일관성과 함께 낮은 TPS를 제공합니다.

사용자 지정 키 스토어를 사용해야 하는 경우 AWS CloudHSM 키 스토어가 외부 키 스토어보다 선호됩니다. AWS KMS 와 HSMs 간의 연결은 AWS 네트워크 내에 유지되므로 AWS 는 XKS 프록시를 AWS

통해 외부에서 실행되는 경로보다 가용성과 일관성을 소유합니다. 어느 쪽이든 키 구성 요소를 격리하고 키에 대한 액세스를 잠글 수 있는 함수를 계속 제어할 수 있습니다.

운영

AWS CloudHSM 키 스토어는 표준 키보다 낮은 TPS와 높은 지연 시간을 제공하며 가용성은 클러스터 구성에 따라 달라지고 SLA 적용 범위는 없습니다. Amazon Elastic Block Store 볼륨 암호화 또는 Amazon RDS 데이터베이스 암호화와 같은 초당 낮은 트랜잭션(TPS) 워크로드에만 적합하며 데이터 분석 서비스, 대용량 Amazon Simple Storage Service 작업 또는 DynamoDB 워크로드를 포함한 높은 TPS 시나리오에 사용해서는 안 됩니다.

책임 모델

앞서 설명한 표준 키 스토어 책임 외에도 암호화 키의 가용성 및 내구성과 클러스터 내 작업의 확장성에 대한 책임은 사용자에게 있습니다 AWS CloudHSM . 설계상는 고객 소유 클러스터의 구성 및 로그에 대한 가시성 AWS CloudHSM 이 제한적입니다. 이렇게 하면가 사용자를 대신하여 주요 액세스 문제를 AWS 해결하는 기능이 제한됩니다.

외부 키 스토어

외부 키 관리 시스템에서 키 구성 요소가 생성, 저장 및 사용되는 외부 키 스토어(XKS)를 AWS KMS 사용하도록를 구성할 수 있습니다 AWS. 에 대한 요청은 네트워크의 XKS 프록시를 통해 외부 호스팅 시스템으로 AWS KMS 전달됩니다. XKS 프록시 API 사양은 공개되어 있으며 많은 상용 키 관리 공급업체가 이를 지원합니다. XKS 프록시는 사용자 또는 외부의 타사에서 호스팅하므로 AWS외부 키 스토어는 모든 키 스토어 옵션 중 가장 낮은 TPS와 가장 높은 지연 시간을 제공하며 가용성, 내구성 및 성능 일관성은 외부 인프라에 따라 달라집니다.

외부 키 스토어는 권장되지 않습니다. 키 구성 요소를 외부에 유지해야 하는 명시적이고 변경할 수 없는 요구 사항이 있는 경우에만 외부 키 스토어를 고려하세요 AWS.

운영

외부 키 스토어는 모든 키 스토어 옵션 중 가장 낮은 TPS와 가장 높은 지연 시간을 제공하며 가용성, 내구성 및 성능 일관성은 외부 인프라에 따라 달라지고 SLA 적용 범위는 없습니다. Amazon Elastic Block Store 볼륨 암호화 또는 Amazon RDS 데이터베이스 암호화와 같은 초당 낮은 트랜잭션(TPS) 워크로드에만 적합하며 데이터 분석 서비스, 대용량 Amazon Simple Storage Service 작업 또는 DynamoDB 워크로드를 포함한 높은 TPS 시나리오에 사용해서는 안 됩니다.

책임 모델

앞서 설명한 표준 키 스토어 책임 외에도 암호화 키의 가용성 및 내구성과 XKS 프록시 및 HSMs. AWS 와 XKS 프록시 간의 통신 경로는 AWS 네트워크 외부에서 실행되므로 AWS 는 해당 경로의 네트워킹 문제에 대한 가시성이 제한되어 있으며 사용자를 대신하여 문제를 해결할 수 없습니다.

AWS CloudHSM 키 스토어

AWS CloudHSM 키 스토어는 AWS CloudHSM 클러스터가 지원하는 사용자 지정 키 스토어입니다. 사용자 지정 키 스토어 AWS KMS key 에서를 생성하면는 사용자가 소유하고 관리하는 AWS CloudHSM 클러스터에서 KMS 키에 대한 추출할 수 없는 키 구성 요소를 AWS KMS 생성하고 저장합니다. 사용자 지정 키 스토어에서 KMS 키를 사용할 때 클러스터의 HSM에서 암호화 작업이 수행됩니다. 이 기능은의 편의성과 광범위한 통합을의 AWS CloudHSM 클러스터에 대한 추가 제어 AWS KMS 와 결합합니다 AWS 계정.

AWS KMS 는 사용자 지정 키 스토어를 생성, 사용 및 관리하기 위한 전체 콘솔 및 API 지원을 제공합니다. KMS 키를 사용하는 것과 동일한 방식으로 사용자 지정 키 스토어에서 KMS 키를 사용할 수 있습니다. 예를 들어 KMS 키를 사용해 데이터 키를 생성하고 데이터를 암호화할 수 있습니다. 고객 관리형 키를 지원하는 AWS 서비스와 함께 사용자 지정 키 스토어의 KMS 키를 사용할 수도 있습니다.

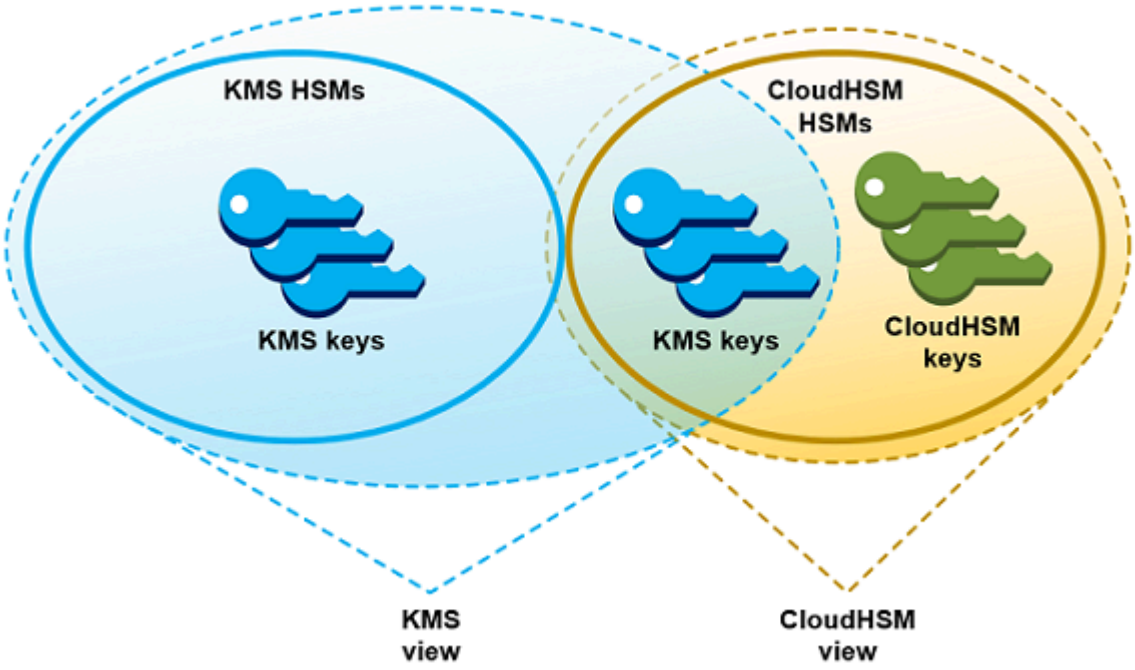
AWS CloudHSM 키 스토어가 필요합니까?

AWS CloudHSM 키 스토어가 요구 사항에 적합한지 확인하려면 섹션을 참조하세요올바른 키 스토어 선택.

AWS CloudHSM 키 스토어는 어떻게 작동하나요?

각 AWS CloudHSM 키 스토어는의 AWS CloudHSM 클러스터와 연결됩니다 AWS 계정. AWS CloudHSM 키 스토어를 클러스터에 연결하면가 연결을 지원하는 네트워크 인프라를 AWS KMS 생성합니다. 그런 다음 클러스터에 있는 전용 암호화 사용자의 자격 증명을 사용하여 클러스터의 키 AWS CloudHSM 클라이언트에 로그인합니다.

에서 AWS CloudHSM 키 스토어를 생성 및 관리하고에서 HSM 클러스터를 AWS KMS 생성 및 관리합니다 AWS CloudHSM. AWS CloudHSM 키 스토어 AWS KMS keys 에서를 생성할 때에서 KMS 키를 보고 관리합니다 AWS KMS. 그러나 클러스터의 다른 키에서와 마찬가지로 AWS CloudHSM에서 키 구성 요소를 확인 및 관리할 수도 있습니다.



키 스토어 AWS KMS 에서 가 생성한 키 구성 요소를 사용하여 대칭 암호화 KMS AWS CloudHSM 키를 생성할 수 있습니다. 그런 다음 동일한 기법을 사용하여 AWS CloudHSM 키 스토어의 KMS 키에 사용하는 AWS KMS 키 스토어의 KMS 키를 보고 관리합니다. IAM 및 키 정책을 통해 액세스를 제어하고 태그 및 별칭을 생성하며 KMS 키를 활성화 및 비활성화하고 키 삭제를 예약할 수 있습니다. KMS 키를 암호화 작업에 사용하고와 통합되는 AWS 서비스와 함께 사용할 수 있습니다 AWS KMS.

또한 HSMs 생성 및 삭제, 백업 관리를 포함하여 AWS CloudHSM 클러스터를 완벽하게 제어할 수 있습니다. AWS CloudHSM 클라이언트 및 지원되는 소프트웨어 라이브러리를 사용하여 KMS 키의 키 구성 요소를 보고, 감사하고, 관리할 수 있습니다. AWS CloudHSM 키 스토어의 연결이 해제된 동안에는가 키 스토어에 액세스할 수 AWS KMS 없으며 사용자는 AWS CloudHSM 키 스토어의 KMS 키를 암호화 작업에 사용할 수 없습니다. 이렇게 추가된 제어 계층은 AWS CloudHSM 키 스토어가 필요한 조직에 강력한 솔루션을 제공합니다.

어디에서 시작합니까?

AWS CloudHSM 키 스토어를 생성하고 관리하려면 AWS KMS 및의 기능을 사용합니다 AWS CloudHSM.

1. 에서 시작합니다 AWS CloudHSM. 활성 AWS CloudHSM 클러스터를 생성하거나 기존 클러스터를 선택합니다. 클러스터는 서로 다른 가용 영역에 최소 2개의 활성 HSM을 가지고 있어야 합니다. 그런 다음 해당 클러스터에 AWS KMS에 대한 전용 암호화 사용자(CU) 계정을 만듭니다.

2. 에서 선택한 AWS CloudHSM 클러스터와 연결된 사용자 지정 키 스토어를 AWS KMS생성합니다. 는 사용자 지정 키 스토어를 생성, 보기, 편집 및 삭제할 수 있는 완전한 관리 인터페이스를 AWS KMS 제공합니다. ???
3. 사용자 지정 키 스토어를 사용할 준비가 되면 연결된 AWS CloudHSM 클러스터에 연결합니다.는 연결을 지원하는 데 필요한 네트워크 인프라를 AWS KMS 생성합니다. 그런 다음, 클러스터에서 키 구성 요소를 생성 및 관리할 수 있도록 전용 CU(Crypto User) 계정 자격 증명을 사용해 클러스터에 로그인합니다.
4. 이제 사용자 지정 키 스토어에서 대칭 암호화 KMS 키를 생성할 수 있습니다. KMS 키를 생성할 때 사용자 지정 키 스토어를 지정만 하면 됩니다.

어떤 지점에서 막히면 사용자 지정 키 스토어 문제 해결 주제에서 도움말을 참조할 수 있습니다. 문제가 해결되지 않은 경우 이 가이드의 각 페이지 하단에 있는 피드백 링크를 사용하거나 AWS Key Management Service 토론 포럼(Discussion Forum)에 질문을 게시합니다.

할당량

AWS KMS 는 연결 상태에 관계없이 키 스토어와 외부 키 스토어를 모두 포함하여 각 및 리전에서 최대 10개의 사용자 지정 키 스토어를 허용합니다. AWS 계정 AWS CloudHSM 또한 AWS KMS AWS CloudHSM 키 스토어의 KMS 키 사용에 대한 요청 할당량이 있습니다.

요금

AWS KMS 사용자 지정 키 스토어 및 사용자 지정 키 스토어의 고객 관리형 키 비용에 대한 자세한 내용은 AWS Key Management Service 요금을 참조하세요. AWS CloudHSM 클러스터 및 HSMs 비용에 대한 자세한 내용은 AWS CloudHSM 요금을 참조하세요.

리전

AWS KMS 는 아시아 태평양(멜버른), 중국(베이징), 중국(닝샤) 및 유럽(스페인)을 제외하고 AWS 리전 AWS KMS 가 지원되는 모든에서 AWS CloudHSM 키 스토어를 지원합니다.

지원되지 않는 기능

AWS KMS 는 사용자 지정 키 스토어에서 다음 기능을 지원하지 않습니다.

- 비대칭 KMS 키
- HMAC KMS 키
- 가져온 키 구성 요소가 있는 KMS 키
- 자동 키 교체

- 다중 리전 키

AWS CloudHSM 키 스토어 개념

이 주제에서는 AWS CloudHSM 키 스토어에 사용되는 몇 가지 용어와 개념을 설명합니다.

AWS CloudHSM 키 스토어

AWS CloudHSM 키 스토어는 사용자가 소유하고 관리하는 클러스터와 AWS CloudHSM 연결된 사용자 지정 키 스토어입니다. AWS CloudHSM 클러스터는 FIPS 140-2 또는 FIPS 140-3Level 3에서 인증된 하드웨어 보안 모듈(HSMs)의 지원을 받습니다.

AWS CloudHSM 키 스토어에서 KMS 키를 생성하면는 연결된 AWS CloudHSM 클러스터에서 내보내기 불가능한 256비트 영구 고급 암호화 표준(AES) 대칭 키를 AWS KMS 생성합니다. 이 키 구성 요소는 HSM을 암호화되지 않은 상태로 두지 않습니다. AWS CloudHSM 키 스토어에서 KMS 키를 사용하는 경우 암호화 작업은 클러스터의 HSMs에서 수행됩니다.

AWS CloudHSM 키 스토어는의 편리하고 포괄적인 키 관리 인터페이스를의 AWS CloudHSM 클러스터에서 제공하는 추가 제어 AWS KMS 와 결합합니다 AWS 계정. 이 통합 기능을 사용하면 클러스터, HSMs 저장하는 HSMs을 완벽하게 제어 AWS KMS 하면서에서 KMS 키를 생성, 관리 및 사용할 수 있습니다. AWS KMS 콘솔과 APIs를 사용하여 키 스토어와 해당 KMS 키를 관리할 AWS CloudHSM 수 있습니다. AWS CloudHSM 콘솔, APIs, 클라이언트 소프트웨어 및 관련 소프트웨어 라이브러리를 사용하여 연결된 클러스터를 관리할 수도 있습니다.

AWS CloudHSM 키 스토어를 보고 관리하고, 속성을 편집하고, 연결된 AWS CloudHSM 클러스터에 연결하고 연결을 해제할 수 있습니다. 키 스토어를 AWS CloudHSM 삭제해야 하는 경우 먼저 삭제를 예약하고 유예 기간이 만료될 때까지 기다려 AWS CloudHSM 키 스토어에서 KMS 키를 삭제해야 합니다. 키 스토어를 AWS CloudHSM 삭제하면에서 리소스가 제거 AWS KMS되지만 클러스터에는 AWS CloudHSM 영향을 주지 않습니다.

AWS CloudHSM 클러스터

모든 AWS CloudHSM 키 스토어는 하나의 AWS CloudHSM 클러스터와 연결됩니다. AWS CloudHSM 키 스토어 AWS KMS key 에서를 생성하면 AWS KMS 는 연결된 클러스터에 키 구성 요소를 생성합니다. AWS CloudHSM 키 스토어에서 KMS 키를 사용할 때 연결 클러스터에서 암호화 작업이 수행됩니다.

각 AWS CloudHSM 클러스터는 하나의 AWS CloudHSM 키 스토어에만 연결할 수 있습니다. 선택한 클러스터는 다른 AWS CloudHSM 키 스토어와 연결하거나 다른 AWS CloudHSM 키 스토어와 연결

된 클러스터와 백업 기록을 공유할 수 없습니다. 클러스터는 초기화되고 활성 상태여야 하며 키 스토어와 동일한 AWS 계정 및 리전 AWS CloudHSM 에 있어야 합니다. 새 클러스터를 생성하거나 기존 클러스터를 사용할 수 AWS KMS 있습니다. 클러스터를 독점적으로 사용할 필요가 없습니다. AWS CloudHSM 키 스토어에서 KMS 키를 생성하려면 연결된 클러스터에 활성 HSMs이 두 개 이상 포함되어야 합니다. 다른 모든 작업에서는 오직 하나의 HSM만 필요합니다.

AWS CloudHSM 키 스토어를 생성할 때 AWS CloudHSM 클러스터를 지정하면 클러스터를 변경할 수 없습니다. 한편 다른 클러스터와 백업 기록을 공유하는 모든 클러스터를 대체할 수 있습니다. 이렇게 하면 필요 시 클러스터를 삭제하고 이를 백업 중 하나에서 생성된 클러스터로 바꿀 수 있습니다. 사용자는 사용자 및 키를 관리하고 HSM을 생성 및 삭제하며 백업을 관리할 수 있도록 연결 AWS CloudHSM 클러스터에 대한 완벽한 제어권을 보유합니다.

AWS CloudHSM 키 스토어를 사용할 준비가 되면 연결된 AWS CloudHSM 클러스터에 연결합니다. 언제든지 사용자 지정 키 스토어를 연결 및 연결 해제할 수 있습니다. 사용자 지정 키 스토어가 연결되면 KMS 키를 생성해 사용할 수 있습니다. 연결이 해제되면 AWS CloudHSM 키 스토어와 해당 KMS 키를 보고 관리할 수 있습니다. 그러나 암호화 작업을 위해 새 KMS 키를 생성하거나 AWS CloudHSM 키 스토어의 KMS 키를 사용할 수는 없습니다.

`kmsuser` 암호화 사용자(CU)

사용자를 대신하여 연결된 AWS CloudHSM 클러스터에서 키 구성 요소를 생성하고 관리하기 위해는 라는 클러스터에서 전용 AWS CloudHSM CU(Crypto User)를 AWS KMS 사용합니다`kmsuser`. `kmsuser` CU는 클러스터의 모든 HSM에 자동으로 동기화되고 클러스터 백업에 저장되는 표준 CU 계정입니다.

AWS CloudHSM 키 스토어를 생성하기 전에 CloudHSM CLI에서 사용자 생성 명령을 사용하여 AWS CloudHSM 클러스터에 `kmsuser` CU 계정을 생성합니다. 그런 다음 AWS CloudHSM 키 스토어를 생성할 때 `kmsuser` 계정 암호를 제공합니다 AWS KMS. 사용자 지정 키 스토어를 연결하면 `kmsuser`는 CU로 클러스터에 AWS KMS 로그인하고 암호를 교체합니다.는 `kmsuser` 암호를 안전하게 저장하기 전에 AWS KMS 암호화합니다. 암호가 교체되면 새 암호가 암호화되어 동일한 방식으로 저장됩니다.

AWS KMS 는 AWS CloudHSM 키 스토어가 연결되어 있는 `kmsuser` 한 로그인 상태로 유지됩니다. 이러한 CU 계정을 다른 목적으로 사용해서는 안 됩니다. 그러나 `kmsuser` CU 계정에 대한 궁극적인 제어권은 보유할 수 있습니다. 언제든지 `kmsuser`가 소유하는 키의 키를 찾을 수 있습니다. 필요한 경우 사용자 지정 키 저장소의 연결을 해제하고, `kmsuser` 암호를 변경하고, `kmsuser`로 클러스터에 로그인하고, `kmsuser`가 소유하는 키를 조회하고 관리할 수 있습니다.

`kmsuser` CU 계정 생성에 대한 지침은 `kmsuser` CU(Crypto User) 생성을 참조하세요.

키 스토어의 KMS AWS CloudHSM 키

AWS KMS 또는 AWS KMS API를 사용하여 AWS CloudHSM 키 스토어 AWS KMS keys 에서를 생성할 수 있습니다. KMS 키에서 사용하는 것과 동일한 기법을 사용합니다. 유일한 차이점은 AWS CloudHSM 키 스토어를 식별하고 키 구성 요소의 오리진이 AWS CloudHSM 클러스터임을 지정해야 한다는 것입니다.

AWS CloudHSM 키 스토어에서 KMS 키를 생성하면는에서 KMS 키를 AWS KMS 생성하고 연결된 클러스터에서 내보내기 불가능한 256비트의 영구 고급 암호화 표준(AES) 대칭 키 구성 요소를 AWS KMS 생성합니다. 암호화 작업에서 AWS KMS 키를 사용하면 AWS CloudHSM 클러스터 기반 AES 키를 사용하여 클러스터에서 작업이 수행됩니다. AWS CloudHSM 는 다양한 유형의 대칭 및 비대칭 키를 지원하지만 AWS CloudHSM 키 스토어는 AES 대칭 암호화 키만 지원합니다.

AWS KMS 콘솔의 키 스토어에서 KMS AWS CloudHSM 키를 보고 콘솔 옵션을 사용하여 사용자 지정 키 스토어 ID를 표시할 수 있습니다. DescribeKey 작업을 사용하여 AWS CloudHSM 키 스토어 ID와 AWS CloudHSM 클러스터 ID를 찾을 수도 있습니다.

AWS CloudHSM 키 스토어의 KMS 키는의 KMS 키와 동일하게 작동합니다 AWS KMS. 권한 있는 사용자는 KMS 키를 사용 및 관리하기 위해 동일한 권한이 필요합니다. 동일한 콘솔 절차와 API 작업을 사용하여 AWS CloudHSM 키 스토어에서 KMS 키를 보고 관리합니다. KMS 키 활성화 및 비활성화, 태그 및 별칭 생성 및 사용, IAM 및 키 정책 변경 등이 여기에 포함됩니다. 암호화 작업을 위해 AWS CloudHSM 키 스토어의 KMS 키를 사용하고 고객 관리형 키 사용을 지원하는 통합 AWS 서비스와 함께 사용할 수 있습니다. 그러나 자동 키 교체를 활성화하거나 키 스토어의 KMS 키로 키 구성 요소를 가져올 수는 없습니다. AWS CloudHSM

또한 동일한 프로세스를 사용하여 키 스토어에서 KMS AWS CloudHSM 키 삭제를 예약합니다. 대기 기간이 만료되면는 KMS에서 KMS 키를 AWS KMS 삭제합니다. 그런 다음 연결된 AWS CloudHSM 클러스터에서 KMS 키의 키 구성 요소를 삭제하기 위해 최선을 다합니다. 하지만 클러스터 및 백업에서 수동으로 불필요한 키 구성 요소를 삭제하고 싶어할 수 있습니다.

AWS CloudHSM 키 스토어에 대한 액세스 제어

IAM 정책을 사용하여 AWS CloudHSM 키 스토어 및 AWS CloudHSM 클러스터에 대한 액세스를 제어합니다. 키 정책, IAM 정책 및 권한 부여를 사용하여 AWS CloudHSM 키 스토어 AWS KMS keys 의에 대한 액세스를 제어할 수 있습니다. 사용자, 그룹 및 역할에 수행할 가능성이 있는 작업에 필요한 권한만 제공하는 것이 좋습니다.

AWS CloudHSM 키 스토어를 지원하려면 클러스터에 대한 정보를 가져올 수 있는 AWS CloudHSM 권한이 AWS KMS 필요합니다. 또한 키 스토어를 AWS CloudHSM 클러스터에 연결하는 AWS

CloudHSM 네트워크 인프라를 생성할 수 있는 권한이 필요합니다. 이러한 권한을 얻기 위해는에서 AWSServiceRoleForKeyManagementServiceCustomKeyStores 서비스 연결 역할을 AWS KMS 생성합니다 AWS 계정. 자세한 내용은 AWS CloudHSM 및 Amazon EC2 리소스를 관리할 수 AWS KMS 있는 권한 부여 단원을 참조하십시오.

AWS CloudHSM 키 스토어를 설계할 때는 키 스토어를 사용하고 관리하는 보안 주체에게 필요한 권한만 있어야 합니다. 다음 목록은 AWS CloudHSM 키 스토어 관리자 및 사용자에게 필요한 최소 권한을 설명합니다.

- AWS CloudHSM 키 스토어를 생성하고 관리하는 보안 주체는 AWS CloudHSM 키 스토어 API 작업을 사용하려면 다음 권한이 필요합니다.
  - `cloudhsm:DescribeClusters`
  - `kms:CreateCustomKeyStore`
  - `kms:ConnectCustomKeyStore`
  - `kms:DeleteCustomKeyStore`
  - `kms:DescribeCustomKeyStores`
  - `kms:DisconnectCustomKeyStore`
  - `kms:UpdateCustomKeyStore`
  - `iam:CreateServiceLinkedRole`
- 키 스토어와 연결된 AWS CloudHSM AWS CloudHSM 클러스터를 생성하고 관리하는 보안 주체는 AWS CloudHSM 클러스터를 생성하고 초기화할 수 있는 권한이 필요합니다. 여기에는 Virtual Private Cloud(VPC)를 생성 또는 사용하고 서브넷을 생성하며 Amazon EC2 인스턴스를 생성하는 권한이 포함되어 있습니다. 보안 주체가 HSM을 생성 및 삭제하고 백업을 관리해야 하는 경우도 있습니다. 필요한 권한 목록은 AWS CloudHSM 사용 설명서의 AWS CloudHSM용 ID 및 액세스 관리를 참조하세요.
- AWS CloudHSM 키 스토어 AWS KMS keys 에서를 생성하고 관리하는 보안 주체는에서 KMS 키를 생성하고 관리하는 보안 주체와 동일한 권한이 필요합니다 AWS KMS. 키 스토어의 KMS 키에 대한 기본 키 정책은의 KMS 키에 대한 기본 키 정책과 동일합니다 AWS KMS. AWS CloudHSM 태그와 별칭을 사용하여 KMS 키에 대한 액세스를 제어하는 속성 기반 액세스 제어(ABAC)도 AWS CloudHSM 키 스토어의 KMS 키에 효과적입니다.
- 암호화 작업에 AWS CloudHSM 키 스토어의 KMS 키를 사용하는 보안 주체는 kms:Decrypt와 같은 KMS 키로 암호화 작업을 수행할 수 있는 권한이 필요합니다. 키 정책, IAM 정책에서 이러한 권한을 제공할 수 있습니다. 그러나 AWS CloudHSM 키 스토어에서 KMS 키를 사용하기 위한 추가 권한은 필요하지 않습니다.

AWS CloudHSM 키 스토어 생성

계정에 하나 이상의 AWS CloudHSM 키 스토어를 생성할 수 있습니다. 각 AWS CloudHSM 키 스토어는 동일한 AWS 계정 및 리전에 있는 하나의 AWS CloudHSM 클러스터와 연결됩니다. AWS CloudHSM 키 스토어를 생성하기 전에 사전 조건을 수집해야 합니다. 그런 다음 AWS CloudHSM 키 스토어를 사용하려면 먼저 키 스토어를 AWS CloudHSM 클러스터에 연결해야 합니다.

> **참고**
>
> KMS는 IPv6를 통해 AWS CloudHSM 키 스토어와 통신할 수 없습니다.
> 연결이 해제 AWS CloudHSM 된 기존 AWS CloudHSM 키 스토어와 동일한 속성 값을 모두 사용하여 키 스토어를 생성하려고 하면 AWS KMS 가 새 AWS CloudHSM 키 스토어를 생성하지 않으며 예외가 발생하거나 오류가 표시되지 않습니다. 대신는 재시도의 결과로 중복을 AWS KMS 인식하고 기존 AWS CloudHSM 키 스토어의 ID를 반환합니다.
> AWS CloudHSM 키 스토어를 즉시 연결할 필요는 없습니다. 사용 준비가 될 때까지 연결 해제 상태로 남겨둘 수 있습니다. 하지만 적절하게 구성이 되었는지 확인하기 위해 이를 연결하고, 연결 상태를 확인한 다음, 연결 해제하고 싶을 수 있습니다.

주제

- 사전 조건 수집
- 새 AWS CloudHSM 키 스토어 생성

사전 조건 수집

각 AWS CloudHSM 키 스토어는 AWS CloudHSM 클러스터에서 지원됩니다. AWS CloudHSM 키 스토어를 생성하려면 다른 키 스토어와 아직 연결되지 않은 활성 AWS CloudHSM 클러스터를 지정해야 합니다. 또한를 사용하여 사용자를 대신하여 키를 생성하고 관리할 AWS KMS 수 있는 클러스터의 HSMs에 전용 CU(Crypto User)를 생성해야 합니다.

AWS CloudHSM 키 스토어를 생성하기 전에 다음을 수행합니다.

AWS CloudHSM 클러스터 선택

모든 AWS CloudHSM 키 스토어는 정확히 하나의 AWS CloudHSM 클러스터와 연결됩니다. AWS CloudHSM 키 스토어 AWS KMS keys 에서를 생성하면가 ID 및 Amazon 리소스 이름(ARN)과 같은 KMS 키 메타데이터를 AWS KMS 생성합니다 AWS KMS. 그런 다음, 연결 클러스터의 HSM에서 키

구성 요소를 생성합니다. 새 클러스터를 생성 AWS CloudHSM하거나 기존 클러스터를 사용할 수 있습니다. 클러스터에 대한 독점적 액세스 AWS KMS 가 필요하지 않습니다.

선택한 AWS CloudHSM 클러스터는 AWS CloudHSM 키 스토어와 영구적으로 연결됩니다. AWS CloudHSM 키 스토어를 생성한 후 연결된 클러스터의 클러스터 ID를 변경할 수 있지만 지정한 클러스터는 백업 기록을 원래 클러스터와 공유해야 합니다. 관련 없는 클러스터를 사용하려면 새 AWS CloudHSM 키 스토어를 생성해야 합니다.

선택한 AWS CloudHSM 클러스터에는 다음과 같은 특성이 있어야 합니다.

- 클러스터는 활성 상태여야 합니다.

  클러스터를 생성하고, 초기화하고, 플랫폼용 AWS CloudHSM 클라이언트 소프트웨어를 설치한 다음 클러스터를 활성화해야 합니다. 자세한 지침을 보려면 AWS CloudHSM 사용 설명서의 AWS CloudHSM시작을 참조하세요.

- 상호 TLS(mTLS)는 활성화되어 있지 않습니다.

  KMS는 클러스터에 대해 mTLS를 지원하지 않습니다. 이 설정은 활성화해서는 안 됩니다.

- 클러스터는 키 스토어와 동일한 계정 및 리전에 있어야 합니다. AWS CloudHSM 한 리전의 AWS CloudHSM 키 스토어를 다른 리전의 클러스터와 연결할 수 없습니다. 여러 리전에서 키 인프라를 생성하려면 각 리전에서 AWS CloudHSM 키 스토어와 클러스터를 생성해야 합니다.

- 이 클러스터는 동일한 계정 및 리전의 다른 사용자 지정 키 스토어에 연결할 수 없습니다. 계정 및 리전의 각 AWS CloudHSM 키 스토어는 다른 AWS CloudHSM 클러스터와 연결되어야 합니다. 사용자 지정 키 스토어에 이미 연결되어 있는 클러스터나 연결 클러스터와 백업 기록을 공유하는 클러스터를 지정할 수 없습니다. 백업 기록을 공유하는 클러스터들은 동일한 클러스터 인증서를 가지고 있습니다. 클러스터의 클러스터 인증서를 보려면 AWS CloudHSM 콘솔 또는 DescribeClusters 작업을 사용합니다.

  AWS CloudHSM 클러스터를 다른 리전으로 백업하는 경우, 이는 다른 클러스터로 간주되며 백업을 해당 리전의 사용자 지정 키 스토어와 연결할 수 있습니다. 그러나 두 사용자 지정 키 스토어의 KMS 키는 동일한 백업 키가 있더라도 상호 운용할 수 없습니다.는 메타데이터를 사이퍼텍스트에 AWS KMS 바인딩하므로 암호화한 KMS 키로만 복호화할 수 있습니다.

- 리전에서 가용 영역이 두 개 이상인 프라이빗 서브넷으로 클러스터를 구성해야 합니다. AWS CloudHSM 는 모든 가용 영역에서 지원되지 않으므로 리전의 모든 가용 영역에서 프라이빗 서브넷을 생성하는 것이 좋습니다. 기존 클러스터에 대한 서브넷을 재구성할 수는 없지만, 클러스터 구성에서 서브넷을 서로 달리하여 백업으로부터 클러스터를 생성할 수 있습니다.

> **Important**
>
> AWS CloudHSM 키 스토어를 생성한 후에는 AWS CloudHSM 클러스터에 대해 구성된 프라이빗 서브넷을 삭제하지 마십시오. 가 클러스터 구성에서 모든 서브넷을 찾을 AWS KMS 수 없는 경우 사용자 지정 키 스토어에 대한 연결 시도가 실패하고 SUBNET_NOT_FOUND 연결 오류 상태가 됩니다. 자세한 내용은 연결 오류를 수정하는 방법을 참조하세요.

- 클러스터용 보안 그룹(cloudhsm-cluster-*<cluster-id>*-sg)은 포트 2223~2225에서 IPv4를 통한 TCP 트래픽을 허용하는 인바운드 규칙과 아웃바운드 규칙을 포함해야 합니다. 인바운드 규칙의 소스와 아웃바운드 규칙의 대상이 보안 그룹 ID와 일치해야 합니다. 이들 규칙은 클러스터를 생성할 때 기본적으로 설정됩니다. 삭제하거나 변경하지 마세요.
- 클러스터는 서로 다른 가용 영역에 활성 HSM을 최소 2개 포함하고 있어야 합니다. HSMs 수를 확인하려면 AWS CloudHSM 콘솔 또는 DescribeClusters 작업을 사용합니다. 필요한 경우 HSM을 추가할 수 있습니다.

트러스트 앵커 인증서 찾기

사용자 지정 키 스토어를 생성할 때 AWS CloudHSM 클러스터의 트러스트 앵커 인증서를에 업로드해야 합니다 AWS KMS. AWS CloudHSM 키 스토어를 연결된 AWS CloudHSM 클러스터에 연결하려면 트러스트 앵커 인증서가 AWS KMS 필요합니다.

모든 활성 AWS CloudHSM 클러스터에는 트러스트 앵커 인증서가 있습니다. 클러스터를 초기화할 때 이러한 인증서를 생성해서 customerCA.crt 파일에 저장하고 클러스터를 연결하는 호스트에 이를 복사합니다.

에 대한 **kmsuser** 암호화 사용자 생성 AWS KMS

AWS CloudHSM 키 스토어를 관리하려면가 선택한 클러스터의 kmsuser CU(Crypto User) 계정에 AWS KMS 로그인합니다. AWS CloudHSM 키 스토어를 생성하기 전에 kmsuser CU를 생성해야 합니다. 그런 다음 AWS CloudHSM 키 스토어를 생성할 때에 대한 암호를 제공합니다kmsuser AWS KMS. AWS CloudHSM 키 스토어를 연결된 AWS CloudHSM 클러스터에 연결할 때마다는 로 AWS KMS 로그인kmsuser하고 kmsuser 암호를 교체합니다.

> **Important**
>
> kmsuser CU를 생성할 때 2FA 옵션을 지정해서는 안 됩니다. 이렇게 하면 AWS KMS 가 로그인할 수 없으며 AWS CloudHSM 키 스토어를이 AWS CloudHSM 클러스터에 연결할

수 없습니다. 일단 2FA를 지정하면 실행 취소를 할 수 없습니다. 대신에 CU를 삭제한 다음 다시 생성해야 합니다.

> **참고**
>
> 다음 절차에서는 AWS CloudHSM 클라이언트 SDK 5 명령줄 도구인 CloudHSM CLI를 사용합니다. CloudHSM CLI에서 `key-handle`을 `key-reference`로 대체합니다. 2025년 1월 1일에 AWS CloudHSM 는 클라이언트 SDK 3 명령줄 도구, CloudHSM 관리 유틸리티(CMU) 및 키 관리 유틸리티(KMU)에 대한 지원을 종료합니다. Client SDK 3 명령줄 도구와 Client SDK 5 명령줄 도구의 차이점에 대한 자세한 내용은 AWS CloudHSM 사용 설명서의 클라이언트 SDK 3 CMU 및 KMU에서 클라이언트 SDK 5 CloudHSM CLI로 마이그레이션을 참조하세요.

1. AWS CloudHSM 사용 설명서의 CloudHSM 명령줄 인터페이스(CLI) 시작하기 항목에 설명된 시작 절차를 따릅니다.
2. user create 명령을 사용하여 `kmsuser`라는 CU를 생성합니다.

   암호는 7~32자의 영숫자로만 구성되어야 합니다. 대소문자가 구분되며 어떤 특수 문자도 포함해서는 안 됩니다.

   다음 예제 명령은 `kmsuser` CU를 생성합니다.

```
aws-cloudhsm > user create --username kmsuser --role crypto-user
Enter password:
Confirm password:
{
 "error_code": 0,
 "data": {
   "username": "kmsuser",
   "role": "crypto-user"
 }
}
```

새 AWS CloudHSM 키 스토어 생성

사전 조건을 수집한 후 AWS KMS 콘솔에서 또는 CreateCustomKeyStore 작업을 사용하여 새 AWS CloudHSM 키 스토어를 생성할 수 있습니다.

AWS KMS 콘솔 사용

에서 AWS CloudHSM 키 스토어를 생성할 때 워크플로의 일부로 사전 조건을 추가하고 생성할 AWS Management Console수 있습니다. 그러나 이들을 미리 수집하면 프로세스가 더 빨라집니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 탐색 창에서 Custom key stores(사용자 지정 키 스토어), AWS CloudHSM key stores를 선택합니다.
4. 키 스토어 생성을 선택합니다.
5. 기억하기 쉬운 사용자 지정 키 스토어 이름을 입력합니다. 이름은 계정의 모든 사용자 지정 키 스토어에서 고유해야 합니다.

   > **Important**
   >
   > 이 필드에 기밀 또는 민감한 정보를 포함하지 마십시오. 이 필드는 CloudTrail 로그 및 기타 출력에 일반 텍스트로 표시될 수 있습니다.

6. AWS CloudHSM 키 스토어의 AWS CloudHSM 클러스터를 선택합니다. 또는 새 AWS CloudHSM 클러스터를 생성하려면 클러스터 생성 AWS CloudHSM 링크를 선택합니다.

   메뉴에는 아직 AWS CloudHSM 키 스토어와 연결되지 않은 계정 및 리전의 AWS CloudHSM 클러스터가 표시됩니다. 사용자 지정 키 스토어를 연결하려면 클러스터가 요구 사항을 충족해야 합니다.
7. 파일 선택을 선택한 다음 선택한 AWS CloudHSM 클러스터에 대한 트러스트 앵커 인증서를 업로드합니다. 이것은 클러스터를 초기화할 때 생성한 `customerCA.crt` 파일입니다.
8. 선택한 클러스터에서 생성한 `kmsuser` CU(Crypto User)의 암호를 입력합니다.
9. 생성(Create)을 선택합니다.

프로시저가 성공하면 계정 및 리전의 AWS CloudHSM AWS CloudHSM 키 스토어 목록에 새 키 스토어가 나타납니다. 절차가 실패하면 오류 메시지가 나타나서 문제를 설명하고 이를 수정할 수 있는 방법

에 대한 도움말을 제공합니다. 도움이 더 필요한 경우 사용자 지정 키 스토어 문제 해결 섹션을 참조하십시오.

연결이 해제 AWS CloudHSM 된 기존 AWS CloudHSM 키 스토어와 동일한 속성 값을 모두 사용하여 키 스토어를 생성하려고 하면 AWS KMS 가 새 AWS CloudHSM 키 스토어를 생성하지 않으며 예외가 발생하거나 오류가 표시되지 않습니다. 대신는 재시도의 결과로 중복을 AWS KMS 인식하고 기존 AWS CloudHSM 키 스토어의 ID를 반환합니다.

다음: 새 AWS CloudHSM 키 스토어는 자동으로 연결되지 않습니다. AWS CloudHSM 키 스토어 AWS KMS keys 에서를 생성하려면 먼저 사용자 지정 키 스토어를 연결된 클러스터에 연결해야 합니다. AWS CloudHSM

AWS KMS API 사용

CreateCustomKeyStore 작업을 사용하여 계정 및 리전의 AWS CloudHSM 클러스터와 연결된 새 AWS CloudHSM 키 스토어를 생성할 수 있습니다. 이 예제들은 AWS Command Line Interface (AWS CLI)를 사용하지만, 사용자는 어떤 지원되는 프로그래밍 언어라도 사용할 수 있습니다.

`CreateCustomKeyStore` 작업에서는 다음과 같은 파라미터 값이 필요합니다.

- CustomKeyStoreName – 계정에서 고유한 기억하기 쉬운 사용자 지정 키 스토어의 이름입니다.

> **Important**
>
> 이 필드에 기밀 또는 민감한 정보를 포함하지 마십시오. 이 필드는 CloudTrail 로그 및 기타 출력에 일반 텍스트로 표시될 수 있습니다.

- CloudHsmClusterId - AWS CloudHSM 키 스토어에 대한 요구 사항을 충족하는 클러스터의 AWS CloudHSM 클러스터 ID입니다.
- KeyStorePassword – 지정된 클러스터의 `kmsuser` CU 계정 암호입니다.
- TrustAnchorCertificate – 클러스터를 초기화할 때 생성한 `customerCA.crt` 파일의 내용입니다.

다음 예제는 가상의 클러스터 ID를 사용합니다. 명령을 실행하기 전에 유효한 클러스터 ID로 이를 바꿉니다.

```
$ aws kms create-custom-key-store
        --custom-key-store-name ExampleCloudHSMKeyStore \
        --cloud-hsm-cluster-id cluster-1a23b4cdefg \
```

```
        --key-store-password kmsPswd \
        --trust-anchor-certificate <certificate-goes-here>
```

를 사용하는 경우 콘텐츠 대신 트러스트 앵커 인증서 파일을 지정할 AWS CLI수 있습니다. 다음 예제에서 `customerCA.crt` 파일은 루트 디렉터리에 있습니다.

```
$ aws kms create-custom-key-store
        --custom-key-store-name ExampleCloudHSMKeyStore \
        --cloud-hsm-cluster-id cluster-1a23b4cdefg \
        --key-store-password kmsPswd \
        --trust-anchor-certificate file://customerCA.crt
```

작업이 성공하면 `CreateCustomKeyStore`가 사용자 지정 키 스토어 ID를 반환합니다(다음 예제 응답 참조).

```
{
    "CustomKeyStoreId": cks-1234567890abcdef0
}
```

작업이 실패할 경우 예외로 표시된 오류를 정정하고 다시 시도해보세요. 추가적인 도움말은 사용자 지정 키 스토어 문제 해결 섹션을 참조하십시오.

연결이 해제 AWS CloudHSM 된 기존 AWS CloudHSM 키 스토어와 동일한 속성 값을 모두 사용하여 키 스토어를 생성하려고 하면 AWS KMS 가 새 AWS CloudHSM 키 스토어를 생성하지 않으며 예외가 발생하거나 오류가 표시되지 않습니다. 대신는 재시도의 결과로 중복을 AWS KMS 인식하고 기존 AWS CloudHSM 키 스토어의 ID를 반환합니다.

다음: AWS CloudHSM 키 스토어를 사용하려면 AWS CloudHSM 클러스터에 연결합니다.

AWS CloudHSM 키 스토어 보기

AWS KMS 콘솔 또는 DescribeCustomKeyStores 작업을 사용하여 각 계정 및 리전의 AWS CloudHSM 키 스토어를 볼 수 있습니다.

AWS KMS 콘솔 사용

에서 AWS CloudHSM 키 스토어를 보면 다음을 볼 AWS Management Console수 있습니다.

- 사용자 지정 키 스토어 이름 및 ID

- 연결된 AWS CloudHSM 클러스터의 ID
- 클러스터의 HSM 개수
- 현재 연결 상태

연결 상태(상태) 값이 연결 해제됨이면 사용자 지정 키 스토어가 새 키 스토어이고 연결되지 않았거나 의도적으로 AWS CloudHSM 클러스터에서 연결 해제되었음을 나타냅니다. 그러나 연결된 사용자 지정 키 스토어에서 KMS 키를 사용하려는 시도가 실패하면 사용자 지정 키 스토어 또는 해당 AWS CloudHSM 클러스터에 문제가 있음을 나타낼 수 있습니다. 도움말은 오류가 발생하는 KMS 키를 수정하는 방법를 참조하십시오.

지정된 계정 및 리전의 AWS CloudHSM 키 스토어를 보려면 다음 절차를 사용합니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 탐색 창에서 Custom key stores(사용자 지정 키 스토어), AWS CloudHSM key stores를 선택합니다.

화면을 사용자 지정하려면 Create key store(키 스토어 생성) 버튼 아래에 나타나는 기어 아이콘을 클릭합니다.

AWS KMS API 사용

AWS CloudHSM 키 스토어를 보려면 DescribeCustomKeyStores 작업을 사용합니다. 기본적으로 이 작업은 계정 및 리전에서 모든 사용자 지정 키 스토어를 반환합니다. 그러나 `CustomKeyStoreId` 또는 `CustomKeyStoreName` 파라미터(둘 중 하나만)를 사용해서 특정한 사용자 지정 키 스토어로 출력을 제한할 수 있습니다. AWS CloudHSM 키 스토어의 경우 출력은 사용자 지정 키 스토어 ID 및 이름, 사용자 지정 키 스토어 유형, 연결된 AWS CloudHSM 클러스터의 ID 및 연결 상태로 구성됩니다. 연결 상태가 오류를 나타내는 경우, 출력에는 오류 원인을 설명하는 오류 코드도 포함됩니다.

이 섹션의 예제는 AWS Command Line Interface (AWS CLI)를 사용하지만, 지원되는 모든 프로그래밍 언어를 사용할 수 있습니다.

예를 들어 다음 명령은 계정 및 리전에 모든 사용자 지정 키 스토어를 반환합니다. `Limit` 및 `Marker` 파라미터를 사용해 출력에서 사용자 지정 키 스토어를 탐색할 수 있습니다.

```
$ aws kms describe-custom-key-stores
```

다음 예제 명령은 CustomKeyStoreName 파라미터를 사용해 기억하기 쉬운 이름이 ExampleCloudHSMKeyStore인 사용자 지정 키 스토어만 검색합니다. 각 명령에서 CustomKeyStoreName 또는 CustomKeyStoreId 파라미터(둘 중 하나만)를 사용할 수 있습니다.

다음 예제 출력은 클러스터에 연결된 AWS CloudHSM 키 스토어를 AWS CloudHSM 나타냅니다.

> **Note**
>
> 키 스토어와 외부 AWS CloudHSM 키 스토어를 구분하기 위해 DescribeCustomKeyStores 응답에 CustomKeyStoreType 필드가 추가되었습니다.

```
$ aws kms describe-custom-key-stores --custom-key-store-name ExampleCloudHSMKeyStore
{
   "CustomKeyStores": [
      {
         "CloudHsmClusterId": "cluster-1a23b4cdefg",
         "ConnectionState": "CONNECTED",
         "CreationDate": "1.499288695918E9",
         "CustomKeyStoreId": "cks-1234567890abcdef0",
         "CustomKeyStoreName": "ExampleCloudHSMKeyStore",
         "CustomKeyStoreType": "AWS_CLOUDHSM",
         "TrustAnchorCertificate": "<certificate appears here>"
      }
   ]
}
```

ConnectionState의는 사용자 지정 키 스토어가 연결되지 않았거나 AWS CloudHSM 클러스터에서 의도적으로 연결 해제되었음을 Disconnected 나타냅니다. 그러나 연결된 키 스토어에서 KMS AWS CloudHSM 키를 사용하려는 시도가 실패하면 AWS CloudHSM 키 스토어 또는 AWS CloudHSM 클러스터에 문제가 있음을 나타낼 수 있습니다. 도움말은 오류가 발생하는 KMS 키를 수정하는 방법를 참조하십시오.

사용자 지정 키 스토어의 ConnectionState가 FAILED이면 DescribeCustomKeyStores 응답에 오류 원인을 설명하는 ConnectionErrorCode 요소가 포함됩니다.

예를 들어 다음 출력에서 INVALID_CREDENTIALS 값은 kmsuser 암호가 잘못되었기 때문에 사용자 지정 키 스토어 연결이 실패했음을 나타냅니다. 이를 비롯해 기타 연결 오류 문제에 대한 도움말은 사용자 지정 키 스토어 문제 해결 단원을 참조하십시오.

```
$ aws kms describe-custom-key-stores --custom-key-store-id cks-1234567890abcdef0
```

```
{
   "CustomKeyStores": [
      {
         "CloudHsmClusterId": "cluster-1a23b4cdefg",
         "ConnectionErrorCode": "INVALID_CREDENTIALS",
         "ConnectionState": "FAILED",
         "CustomKeyStoreId": "cks-1234567890abcdef0",
         "CustomKeyStoreName": "ExampleCloudHSMKeyStore",
         "CustomKeyStoreType": "AWS_CLOUDHSM",
         "CreationDate": "1.499288695918E9",
         "TrustAnchorCertificate": "<certificate appears here>"
      }
   ]
}
```

자세히 알아보기:

- 외부 키 저장소 보기
- 키 스토어에서 KMS AWS CloudHSM 키 식별
- 를 사용하여 AWS KMS API 호출 로깅 AWS CloudTrail

AWS CloudHSM 키 스토어 설정 편집

기존 AWS CloudHSM 키 스토어의 설정을 변경할 수 있습니다. 사용자 지정 키 스토어는 AWS CloudHSM 클러스터의 연결을 해제해야 합니다.

AWS CloudHSM 키 스토어 설정을 편집하려면:

1. AWS CloudHSM 클러스터에서 사용자 지정 키 스토어의 연결이 해제되어 있어야 합니다.

   사용자 지정 키 스토어의 연결이 해제된 동안에는 사용자 지정 키 스토어에서 AWS KMS keys (KMS 키)를 생성할 수 없으며 암호화 작업에 포함된 KMS 키를 사용할 수 없습니다.
2. 하나 이상의 AWS CloudHSM 키 스토어 설정을 편집합니다.

   사용자 지정 키 스토어에서 다음 설정을 편집할 수 있습니다.

   기억하기 쉬운 사용자 지정 키 스토어 이름입니다.

   기억하기 쉬운 이름을 새로 입력합니다. 새 이름은의 모든 사용자 지정 키 스토어에서 고유해야 합니다 AWS 계정.

> **Important**
>
> 이 필드에 기밀 또는 민감한 정보를 포함하지 마십시오. 이 필드는 CloudTrail 로그 및 기타 출력에 일반 텍스트로 표시될 수 있습니다.

연결된 클러스터의 AWS CloudHSM 클러스터 ID입니다.

관련 AWS CloudHSM 클러스터를 원래 클러스터로 대체하려면이 값을 편집합니다. AWS CloudHSM 클러스터가 손상되거나 삭제된 경우이 기능을 사용하여 사용자 지정 키 스토어를 복구할 수 있습니다.

원래 AWS CloudHSM 클러스터와 백업 기록을 공유하고 서로 다른 가용 영역에 있는 두 개의 활성 HSMs을 포함하여 사용자 지정 키 스토어와의 연결 요구 사항을 충족하는 클러스터를 지정합니다. 백업 기록을 공유하는 클러스터들은 동일한 클러스터 인증서를 가지고 있습니다. 클러스터의 클러스터 인증서를 보려면 DescribeClusters 작업을 사용합니다. 관련 없는 AWS CloudHSM 클러스터에 사용자 지정 키 스토어를 연결하기 위해 편집 기능을 사용할 수는 없습니다.

`kmsuser`CU(Crypto User)의 현재 암호입니다.

클러스터 AWS KMS 에 있는 `kmsuser` CU AWS CloudHSM 의 현재 암호를 알려줍니다. 이 작업은 AWS CloudHSM 클러스터에서 `kmsuser` CU의 암호를 변경하지 않습니다.

AWS CloudHSM 클러스터에서 `kmsuser` CU의 암호를 변경하는 경우이 기능을 사용하여 AWS KMS 새 `kmsuser` 암호를 알려줍니다. 그렇지 않으면 AWS KMS 가 클러스터에 로그인을 할 수 없고, 사용자 지정 키 스토어를 클러스터에 연결하려는 모든 시도가 실패로 돌아갑니다.

3. AWS CloudHSM 클러스터에 사용자 지정 키 스토어를 다시 연결합니다.

키 저장소 설정 편집

AWS KMS 콘솔에서 또는 UpdateCustomKeyStore 작업을 사용하여 AWS CloudHSM 키 스토어 설정을 편집할 수 있습니다.

AWS KMS 콘솔 사용

AWS CloudHSM 키 스토어를 편집할 때 구성 가능한 값 중 일부 또는 일부를 변경할 수 있습니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 탐색 창에서 Custom key stores(사용자 지정 키 스토어), AWS CloudHSM key stores를 선택합니다.
4. 편집할 AWS CloudHSM 키 스토어의 행을 선택합니다.

   연결 상태 열의 값이 연결 해제됨이 아니면 먼저 사용자 지정 키 스토어의 연결을 해제해야 편집을 할 수 있습니다. Key store actions(키 스토어 작업) 메뉴에서 Disconnect(연결 해제)를 선택합니다.

   AWS CloudHSM 키 스토어의 연결이 끊긴 동안에는 AWS CloudHSM 키 스토어와 해당 KMS 키를 관리할 수 있지만 AWS CloudHSM 키 스토어에서 KMS 키를 생성하거나 사용할 수는 없습니다.
5. Key store actions(키 스토어 작업) 메뉴에서 Edit(편집)를 선택합니다.
6. 다음 작업 중 한 개 이상을 수행합니다.

   - 기억하기 쉬운 사용자 지정 키 스토어 이름을 새로 입력합니다.
   - 관련 클러스터의 AWS CloudHSM 클러스터 ID를 입력합니다.
   - 연결된 AWS CloudHSM 클러스터에 있는 `kmsuser` 암호화 사용자의 현재 암호를 입력합니다.
7. 저장을 선택합니다.

   절차가 성공하면 편집한 설정을 설명하는 메시지가 표시됩니다. 절차가 실패하면 오류 메시지가 나타나서 문제를 설명하고 이를 수정할 수 있는 방법에 대한 도움말을 제공합니다. 도움이 더 필요한 경우 사용자 지정 키 스토어 문제 해결 섹션을 참조하십시오.
8. 사용자 지정 키 스토어를 다시 연결합니다.

   AWS CloudHSM 키 스토어를 사용하려면 편집 후 다시 연결해야 합니다. AWS CloudHSM 키 스토어를 연결 해제된 상태로 둘 수 있습니다. 그러나 연결이 해제된 동안에는 AWS CloudHSM 키 스토어에서 KMS 키를 생성하거나 암호화 작업에서 AWS CloudHSM 키 스토어의 KMS 키를 사용할 수 없습니다.

AWS KMS API 사용

AWS CloudHSM 키 스토어의 속성을 변경하려면 UpdateCustomKeyStore 작업을 사용합니다. 동일한 명령으로 사용자 지정 키 스토어에서 여러 개의 속성을 변경할 수 있습니다. 작업이 성공하면는 HTTP 200 응답과 속성이 없는 JSON 객체를 AWS KMS 반환합니다. 변경 사항이 유효한지 확인하려면 DescribeCustomKeyStores 작업을 사용합니다.

이 섹션의 예제는 AWS Command Line Interface (AWS CLI)를 사용하지만, 지원되는 모든 프로그래밍 언어를 사용할 수 있습니다.

먼저 DisconnectCustomKeyStore를 사용하여 AWS CloudHSM 클러스터에서 사용자 지정 키 스토어의 연결을 해제합니다. 예제에 나온 사용자 지정 키 스토어 ID인 cks-1234567890abcdef0을 실제 ID로 대체합니다.

```
$ aws kms disconnect-custom-key-store --custom-key-store-id cks-1234567890abcdef0
```

첫 번째 예제에서는 UpdateCustomKeyStore를 사용하여 AWS CloudHSM 키 스토어의 표시 이름을 로 변경합니다DevelopmentKeys. 명령은 CustomKeyStoreId 파라미터를 사용하여 AWS CloudHSM 키 스토어를 식별하고를 CustomKeyStoreName 사용하여 사용자 지정 키 스토어의 새 이름을 지정합니다.

```
$ aws kms update-custom-key-store --custom-key-store-id cks-1234567890abcdef0 --new-custom-key-store-name DevelopmentKeys
```

다음 예시에서는 AWS CloudHSM 키 스토어와 연결된 클러스터를 동일한 클러스터의 다른 백업으로 변경합니다. 명령은 CustomKeyStoreId 파라미터를 사용하여 AWS CloudHSM 키 스토어를 식별하고 CloudHsmClusterId 파라미터를 사용하여 새 클러스터 ID를 지정합니다.

```
$ aws kms update-custom-key-store --custom-key-store-id cks-1234567890abcdef0 --cloud-hsm-cluster-id cluster-1a23b4cdefg
```

다음 예제에서는 현재 kmsuser 암호가 AWS KMS 임을 알려줍니다ExamplePassword.
명령은 CustomKeyStoreId 파라미터를 사용하여 AWS CloudHSM 키 스토어를 식별하고 KeyStorePassword 파라미터를 사용하여 현재 암호를 지정합니다.

```
$ aws kms update-custom-key-store --custom-key-store-id cks-1234567890abcdef0 --key-store-password ExamplePassword
```

마지막 명령은 AWS CloudHSM 키 스토어를 AWS CloudHSM 클러스터에 다시 연결합니다. 사용자 지정 키 스토어를 연결 해제 상태로 남겨둘 수 있지만, 새 KMS 키를 생성하거나 기존 KMS 키를 암호화 작업에 사용할 수 있으려면 먼저 이를 연결해야 합니다. 예제에 나온 사용자 지정 키 스토어 ID를 실제 ID로 대체합니다.

```
$ aws kms connect-custom-key-store --custom-key-store-id cks-1234567890abcdef0
```

AWS CloudHSM 키 스토어 연결

새 AWS CloudHSM 키 스토어가 연결되지 않았습니다. AWS CloudHSM 키 스토어 AWS KMS keys 에서를 생성하고 사용하려면 먼저 연결된 AWS CloudHSM 클러스터에 연결해야 합니다. 언제든지 AWS CloudHSM 키 스토어를 연결 및 연결 해제하고 연결 상태를 볼 수 있습니다.

AWS CloudHSM 키 스토어를 연결할 필요는 없습니다. AWS CloudHSM 키 스토어를 무기한 연결 해제된 상태로 두고 사용해야 하는 경우에만 연결할 수 있습니다. 하지만 설정이 올바른지, 연결이 가능한지 확인하기 위해 정기적으로 연결을 테스트하고 싶을 수 있습니다.

> **Note**
>
> AWS CloudHSM 키 스토어는 키 스토어가 `DISCONNECTED` 연결되지 않았거나 사용자가 명시적으로 연결을 해제한 경우에만 연결 상태가 됩니다. AWS CloudHSM 키 스토어 연결 상태가 `CONNECTED`이지만 사용하는 데 문제가 있는 경우 연결된 AWS CloudHSM 클러스터가 활성 상태이고 하나 이상의 활성 HSMs 포함되어 있는지 확인합니다. 연결 실패에 대한 도움말은 the section called “사용자 지정 키 스토어 문제 해결” 단원을 참조하십시오.

AWS CloudHSM 키 스토어를 연결하면 연결된 AWS CloudHSM 클러스터를 AWS KMS 찾고 연결한 다음 CU(`kmsuser`Crypto User)로 AWS CloudHSM 클라이언트에 로그인한 다음 AWS CloudHSM 키 스토어가 연결되어 있는 한 AWS CloudHSM 클라이언트에 AWS KMS 로그인한 `kmsuser` 암호를 교체합니다.

연결을 설정하기 위해는 클러스터의 Virtual Private Cloud(VPC)`kms-<custom key store ID>`에라는 보안 그룹을 AWS KMS 생성합니다. 보안 그룹에는 클러스터 보안 그룹의 인바운드 트래픽을 허용하는 단일 규칙이 있습니다. AWS KMS 또한는 클러스터에 대한 프라이빗 서브넷의 각 가용 영역에 탄력적 네트워크 인터페이스(ENI)를 생성합니다.는 클러스터의 `kms-<cluster ID>` 보안 그룹 및 보안 그룹에 ENIs를 AWS KMS 추가합니다. 각 ENI에 대한 설명은 `KMS managed ENI for cluster <cluster-ID>`입니다.

연결 프로세스를 완료하는 데 최대 20분이 걸릴 수 있습니다.

AWS CloudHSM 키 스토어를 연결하기 전에 키 스토어가 요구 사항을 충족하는지 확인합니다.

- 연결된 AWS CloudHSM 클러스터에는 하나 이상의 활성 HSM이 포함되어야 합니다. 클러스터에서 HSMs 수를 찾으려면 AWS CloudHSM 콘솔에서 클러스터를 보거나 DescribeClusters 작업을 사용합니다. 필요한 경우 HSM을 추가할 수 있습니다.

- 클러스터에는 CU(kmsuserCrypto User) 계정이 있어야 하지만 AWS CloudHSM 키 스토어를 연결할 때 해당 CU를 클러스터에 로그인할 수 없습니다. 로그아웃에 대한 도움말은 로그아웃 및 재연결 방법 단원을 참조하십시오.
- AWS CloudHSM 키 스토어의 연결 상태는 DISCONNECTING 또는 일 수 없습니다FAILED. 연결 상태를 보려면 AWS KMS 콘솔 또는 DescribeCustomKeyStores 응답을 사용합니다. 연결 상태가 FAILED면 사용자 지정 키 스토어를 연결 해제하고 문제를 수정한 다음 다시 연결합니다.

연결 실패에 대한 도움말은 연결 오류를 수정하는 방법 단원을 참조하십시오.

AWS CloudHSM 키 스토어가 연결되면 키 스토어에서 KMS 키를 생성하고 암호화 작업에서 기존 KMS 키를 사용할 수 있습니다.

AWS CloudHSM 키 스토어에 연결 및 다시 연결

AWS KMS 콘솔에서 또는 ConnectCustomKeyStore 작업을 사용하여 AWS CloudHSM 키 스토어를 연결하거나 다시 연결할 수 있습니다.

AWS KMS 콘솔 사용

에서 AWS CloudHSM 키 스토어를 연결하려면 AWS Management Console먼저 사용자 지정 AWS CloudHSM 키 스토어 페이지에서 키 스토어를 선택합니다. 연결 프로세스가 완료되는 데 최대 20분이 걸릴 수 있습니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 탐색 창에서 Custom key stores(사용자 지정 키 스토어), AWS CloudHSM key stores를 선택합니다.
4. 연결하려는 AWS CloudHSM 키 스토어의 행을 선택합니다.

   AWS CloudHSM 키 스토어의 연결 상태가 실패인 경우 연결하기 전에 사용자 지정 키 스토어의 연결을 해제해야 합니다.
5. Key store actions(키 스토어 작업) 메뉴에서 Connect(연결)를 선택합니다.

AWS KMS 는 사용자 지정 키 스토어를 연결하는 프로세스를 시작합니다. 또한 연결 AWS CloudHSM 클러스터를 찾고, 필요한 네트워크 인프라를 구축하여 연결하고, kmsuser CU로 AWS CloudHSM 클러스터에 로그인하고, kmsuser 암호를 교체합니다. 이 작업이 완료되면 연결 상태가 연결됨으로 변경됩니다.

작업이 실패하면 실패 원인을 설명하는 오류 메시지가 나타납니다. 다시 연결하기 전에 AWS CloudHSM 키 스토어의 연결 상태를 확인합니다. 연결 상태가 실패이면 다시 연결을 하기 앞서 사용자 지정 키 스토어의 연결을 해제해야 합니다. 도움이 필요한 경우 사용자 지정 키 스토어 문제 해결 단원을 참조하십시오.

다음: the section called “키 스토어에서 KMS AWS CloudHSM 키 생성”.

AWS KMS API 사용

연결 해제된 AWS CloudHSM 키 스토어를 연결하려면 ConnectCustomKeyStore 작업을 사용합니다. 연결된 AWS CloudHSM 클러스터에는 하나 이상의 활성 HSM이 포함되어야 하며 연결 상태는 일 수 없습니다`FAILED`.

연결 프로세스를 완료하는 데 최대 20분이 걸릴 수 있습니다. 빨리 실패하지 않는 한, 작업은 속성 없이 HTTP 200 응답 및 JSON 객체를 반환합니다. 하지만 이러한 초기 응답은 연결이 성공했음을 의미하지는 않습니다. 사용자 지정 키 스토어의 연결 상태를 확인하려면 DescribeCustomKeyStores 응답을 봅니다.

이 섹션의 예제는 AWS Command Line Interface (AWS CLI)를 사용하지만, 지원되는 모든 프로그래밍 언어를 사용할 수 있습니다.

AWS CloudHSM 키 스토어를 식별하려면 사용자 지정 키 스토어 ID를 사용합니다. 콘솔의 Custom key stores(사용자 지정 키 스토어) 페이지에서, 또는 파라미터 없이 DescribeCustomKeyStores 작업을 사용하여 ID를 찾을 수 있습니다. 이 예제를 실행하기 앞서 예제 ID를 유효한 ID로 바꿉니다.

```
$ aws kms connect-custom-key-store --custom-key-store-id cks-1234567890abcdef0
```

AWS CloudHSM 키 스토어가 연결되어 있는지 확인하려면 DescribeCustomKeyStores 작업을 사용합니다. 기본적으로 이 작업은 계정 및 리전에서 모든 사용자 지정 키 스토어를 반환합니다. 그러나 `CustomKeyStoreId` 또는 `CustomKeyStoreName` 파라미터(둘 중 하나만)를 사용해서 특정한 사용자 지정 키 스토어로 응답을 제한할 수 있습니다. `ConnectionState` 값이 `CONNECTED`라는 것은 AWS CloudHSM 클러스터에 사용자 지정 키 스토어가 연결되어 있다는 것을 의미합니다.

> **Note**
>
> 키 스토어와 외부 AWS CloudHSM 키 스토어를 구분하기 위해 `DescribeCustomKeyStores` 응답에 `CustomKeyStoreType` 필드가 추가되었습니다.

```
$ aws kms describe-custom-key-stores --custom-key-store-id cks-1234567890abcdef0
```

```
{
   "CustomKeyStores": [
      "CustomKeyStoreId": "cks-1234567890abcdef0",
      "CustomKeyStoreName": "ExampleCloudHSMKeyStore",
      "CloudHsmClusterId": "cluster-1a23b4cdefg",
      "CustomKeyStoreType": "AWS_CLOUDHSM",
      "TrustAnchorCertificate": "<certificate string appears here>",
      "CreationDate": "1.499288695918E9",
      "ConnectionState": "CONNECTED"
   ],
}
```

ConnectionState 값이 실패인 경우, ConnectionErrorCode 요소는 실패 원인을 나타냅니다. 이 경우 계정에서 AWS CloudHSM 클러스터 ID가 인 클러스터를 찾을 수 AWS KMS 없습니다cluster-1a23b4cdefg. 클러스터를 삭제한 경우에는 원래 클러스터의 백업에서 이를 복원한 다음, 사용자 지정 키 스토어의 클러스터 ID를 편집할 수 있습니다. 연결 오류 코드에 대응하는 방법에 대한 도움말은 연결 오류를 수정하는 방법 섹션을 참조하세요.

```
$ aws kms describe-custom-key-stores --custom-key-store-id cks-1234567890abcdef0
{
   "CustomKeyStores": [
      "CustomKeyStoreId": "cks-1234567890abcdef0",
      "CustomKeyStoreName": "ExampleKeyStore",
      "CloudHsmClusterId": "cluster-1a23b4cdefg",
      "CustomKeyStoreType": "AWS_CLOUDHSM",
      "TrustAnchorCertificate": "<certificate string appears here>",
      "CreationDate": "1.499288695918E9",
      "ConnectionState": "FAILED"
      "ConnectionErrorCode": "CLUSTER_NOT_FOUND"
   ],
}
```

AWS CloudHSM 키 스토어 연결 해제

AWS CloudHSM 키 스토어의 연결을 해제하면는 AWS CloudHSM 클라이언트에서 AWS KMS 로그아웃하고, 연결된 AWS CloudHSM 클러스터에서 연결을 해제하고, 연결을 지원하기 위해 생성한 네트워크 인프라를 제거합니다.

AWS CloudHSM 키 스토어의 연결이 해제된 동안에는 AWS CloudHSM 키 스토어와 해당 KMS 키를 관리할 수 있지만 AWS CloudHSM 키 스토어에서 KMS 키를 생성하거나 사용할 수는 없습니다. 키 스토어의 연결 상태는 DISCONNECTED이며 사용자 지정 키 스토어의 KMS 키 상태는 Unavailable입

니다(PendingDeletion가 아닌 경우). 언제든지 AWS CloudHSM 키 스토어를 다시 연결할 수 있습니다.

> **Note**
>
> AWS CloudHSM 키 스토어는 키 스토어가 DISCONNECTED 연결되지 않았거나 사용자가 명시적으로 연결을 해제한 경우에만 연결 상태가 됩니다. AWS CloudHSM 키 스토어 연결 상태가 CONNECTED이지만 사용하는 데 문제가 있는 경우 연결된 AWS CloudHSM 클러스터가 활성 상태이고 하나 이상의 활성 HSMs 포함되어 있는지 확인합니다. 연결 실패에 대한 도움말은 the section called “사용자 지정 키 스토어 문제 해결” 단원을 참조하십시오.

사용자 지정 키 스토어를 연결 해제하면 키 스토어의 KMS 키를 즉시 사용할 수 없게 됩니다(최종 일관성에 따라 다름). 그러나 KMS 키로 보호되는 데이터 키로 암호화된 리소스는 데이터 키를 복호화하는 등 KMS 키를 다시 사용할 때까지 영향을 받지 않습니다. 이 문제는 리소스를 보호하기 위해 데이터 키를 사용하는 AWS 서비스에 영향을 미칩니다. 자세한 내용은 사용할 수 없는 KMS 키가 데이터 키에 미치는 영향을 참조하세요.

> **Note**
>
> 사용자 지정 키 스토어의 연결이 해제된 상태에서는 사용자 지정 키 스토어에서 KMS 키를 생성하거나, 암호화 작업을 위해 기존 KMS 키를 사용하려는 모든 시도가 실패합니다. 이 작업은 사용자가 기밀 데이터를 저장하거나 액세스하지 못하도록 차단합니다.

사용자 지정 키 스토어의 연결 해제가 미치는 영향을 정확하게 예측하려면 사용자 지정 키 스토어에서 KMS 키를 식별하고 과거 사용량을 판단합니다.

다음과 같은 이유로 AWS CloudHSM 키 스토어의 연결을 끊을 수 있습니다.

- **kmsuser** 암호를 교체하려면 AWS KMS 는 AWS CloudHSM 클러스터에 연결될 때마다 kmsuser 암호를 변경합니다. 강제로 암호 교체를 수행하려면 연결을 해제하고 다시 연결하기만 하면 됩니다.
- AWS CloudHSM 클러스터의 KMS 키에 대한 키 구성 요소를 감사합니다. 사용자 지정 키 스토어의 연결을 해제하면가 AWS CloudHSM 클라이언트의 kmsuser 암호화 사용자 계정에서 AWS KMS 로그아웃합니다. 따라서 사용자는 kmsuser CU로 클러스터에 로그인하여 KMS 키의 키 구성 요소를 감사 및 관리할 수 있습니다.
- AWS CloudHSM 키 스토어에서 모든 KMS 키를 즉시 비활성화하려면 다음을 수행하세요. AWS Management Console 또는 DisableKey 작업을 사용하여 키 스토어에서 KMS 키를 비활성화하고

다시 활성화할 수 있습니다. AWS CloudHSM 이러한 작업들은 신속하게 완료되지만, 한 번에 하나의 KMS 키에 대해서만 수행됩니다. AWS CloudHSM 키 스토어를 연결 해제하면 키 스토어에 있는 AWS CloudHSM 모든 KMS 키의 키 상태가 로 즉시 변경`Unavailable`되어 암호화 작업에 사용되지 않습니다.

- 실패한 연결 시도를 복구하려면. AWS CloudHSM 키 스토어 연결 시도가 실패하면(사용자 지정 키 스토어의 연결 상태가 임`FAILED`) 다시 연결을 시도하기 전에 AWS CloudHSM 키 스토어의 연결을 끊어야 합니다.

AWS CloudHSM 키 스토어 연결 해제

AWS KMS 콘솔에서 또는 DisconnectCustomKeyStore 작업을 사용하여 AWS CloudHSM 키 스토어의 연결을 해제할 수 있습니다.

AWS KMS 콘솔을 사용하여 연결 해제

AWS KMS 콘솔에서 연결된 AWS CloudHSM 키 스토어의 연결을 해제하려면 먼저 사용자 지정 AWS CloudHSM 키 스토어 페이지에서 키 스토어를 선택합니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 탐색 창에서 Custom key stores(사용자 지정 키 스토어), AWS CloudHSM key stores를 선택합니다.
4. 연결 해제하려는 외부 키 스토어의 행을 선택합니다.
5. Key store actions(키 스토어 작업) 메뉴에서 Disconnect(연결 해제)를 선택합니다.

작업이 완료되면 연결 상태가 연결 해제 중에서 연결 해제됨으로 변경됩니다. 작업이 실패하면 오류 메시지가 나타나서 문제를 설명하고 이를 수정할 수 있는 방법에 대한 도움말을 제공합니다. 도움이 더 필요한 경우 사용자 지정 키 스토어 문제 해결 섹션을 참조하십시오.

AWS KMS API를 사용하여 연결 해제

연결된 AWS CloudHSM 키 스토어의 연결을 해제하려면 DisconnectCustomKeyStore 작업을 사용합니다. 작업이 성공하면 HTTP 200 응답과 속성이 없는 JSON 객체를 AWS KMS 반환합니다.

이 섹션의 예제는 AWS Command Line Interface (AWS CLI)를 사용하지만, 지원되는 모든 프로그래밍 언어를 사용할 수 있습니다.

이 예시에서는 AWS CloudHSM 키 스토어의 연결을 해제합니다. 이 예제를 실행하기 앞서 예제 ID를 유효한 ID로 바꿉니다.

```
$ aws kms disconnect-custom-key-store --custom-key-store-id cks-1234567890abcdef0
```

AWS CloudHSM 키 스토어의 연결이 해제되었는지 확인하려면 DescribeCustomKeyStores 작업을 사용합니다. 기본적으로 이 작업은 계정 및 리전에서 모든 사용자 지정 키 스토어를 반환합니다. 그러나 CustomKeyStoreId 또는 CustomKeyStoreName 파라미터(둘 중 하나만)를 사용해서 특정한 사용자 지정 키 스토어로 응답을 제한할 수 있습니다. 의 ConnectionState 값은이 예제 AWS CloudHSM 키 스토어가 AWS CloudHSM 클러스터에 연결되지 않았음을 DISCONNECTED 나타냅니다.

```
$ aws kms describe-custom-key-stores --custom-key-store-id cks-1234567890abcdef0
{
   "CustomKeyStores": [
      "CloudHsmClusterId": "cluster-1a23b4cdefg",
      "ConnectionState": "DISCONNECTED",
      "CreationDate": "1.499288695918E9",
      "CustomKeyStoreId": "cks-1234567890abcdef0",
      "CustomKeyStoreName": "ExampleKeyStore",
      "CustomKeyStoreType": "AWS_CLOUDHSM",
      "TrustAnchorCertificate": "<certificate string appears here>"
   ],
}
```

AWS CloudHSM 키 스토어 삭제

AWS CloudHSM 키 스토어를 삭제하면는 AWS CloudHSM 클러스터와의 연결에 대한 정보를 포함하여 KMS에서 AWS CloudHSM 키 스토어에 대한 모든 메타데이터를 AWS KMS 삭제합니다. 이 작업은 AWS CloudHSM 클러스터, HSMs 또는 사용자에게 영향을 주지 않습니다. 동일한 AWS CloudHSM 클러스터와 연결된 새 AWS CloudHSM 키 스토어를 생성할 수 있지만 삭제 작업을 실행 취소할 수는 없습니다.

AWS CloudHSM 클러스터에서 연결이 끊어지고 키 스토어가 포함되지 않은 AWS CloudHSM 키 스토어만 삭제할 수 있습니다 AWS KMS keys. 사용자 지정 키 스토어를 삭제하기 전에 다음을 수행합니다.

- 모든 암호화 작업에서 키 스토어의 KMS 키를 전혀 사용할 필요가 없는지 확인합니다. 그런 다음, 키 스토어에서 모든 KMS 키의 삭제를 예약합니다. 키 스토어에서 KMS AWS CloudHSM 키를 찾는 방법에 대한 도움말은 섹션을 참조하세요키 스토어에서 KMS AWS CloudHSM 키 찾기.
- 모든 KMS 키가 삭제되었는지 확인합니다. 키 스토어에서 KMS AWS CloudHSM 키를 보려면 섹션을 참조하세요the section called “키 스토어에서 KMS AWS CloudHSM 키 식별”.

- 클러스터에서 AWS CloudHSM 키 스토어의 연결을 해제합니다 AWS CloudHSM .

AWS CloudHSM 키 스토어를 삭제하는 대신 연결된 AWS CloudHSM 클러스터에서 연결을 해제하는 것이 좋습니다. AWS CloudHSM 키 스토어의 연결이 해제된 상태에서 AWS CloudHSM 키 스토어와 해당를 관리할 수 있습니다 AWS KMS keys. 하지만 키 스토어에서는 KMS AWS CloudHSM 키를 생성하거나 사용할 수 없습니다. 언제든지 AWS CloudHSM 키 스토어를 다시 연결할 수 있습니다.

AWS CloudHSM 키 스토어 삭제

AWS KMS 콘솔에서 또는 DeleteCustomKeyStore 작업을 사용하여 AWS CloudHSM 키 스토어를 삭제할 수 있습니다.

AWS KMS 콘솔 사용

에서 AWS CloudHSM 키 스토어를 삭제하려면 AWS Management Console먼저 사용자 지정 AWS CloudHSM 키 스토어 페이지에서 키 스토어를 선택합니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 탐색 창에서 Custom key stores(사용자 지정 키 스토어), AWS CloudHSM key stores를 선택합니다.
4. 삭제할 AWS CloudHSM 키 스토어를 나타내는 행을 찾습니다. AWS CloudHSM 키 스토어의 연결 상태가 연결 해제되지 않은 경우 삭제하기 전에 AWS CloudHSM 키 스토어의 연결을 해제해야 합니다.
5. Key store actions(키 스토어 작업) 메뉴에서 Delete(삭제)를 선택합니다.

작업이 완료되면 성공 메시지가 나타나고 AWS CloudHSM 키 스토어 목록에 키 스토어가 더 이상 나타나지 않습니다. 작업이 실패하면 오류 메시지가 나타나서 문제를 설명하고 이를 수정할 수 있는 방법에 대한 도움말을 제공합니다. 도움이 더 필요한 경우 사용자 지정 키 스토어 문제 해결 섹션을 참조하십시오.

AWS KMS API 사용

AWS CloudHSM 키 스토어를 삭제하려면 DeleteCustomKeyStore 작업을 사용합니다. 작업이 성공하면 속성이 없는 JSON 객체와 HTTP 200 응답을 AWS KMS 반환합니다.

시작하려면 AWS CloudHSM 키 스토어에이 포함되어 있지 않은지 확인합니다 AWS KMS keys. KMS 키를 포함하는 사용자 지정 키 스토어를 삭제할 수 없습니다. 첫 번째 예제 명령은 ListKeys 및 DescribeKey를 사용하여 예제 *cks-1234567890abcdef0* 사용자 지정 AWS CloudHSM 키 스토어 ID로 키 스토어 AWS KMS keys 에서를 검색합니다. 이 경우, 명령은 어떤 KMS 키도 반환하지 않습니다. 그리고 ScheduleKeyDeletion 작업을 사용하여 각 KMS 키에 대한 삭제를 예약합니다.

Bash

```
for key in $(aws kms list-keys --query 'Keys[*].KeyId' --output text) ;
do aws kms describe-key --key-id $key |
grep '"CustomKeyStoreId": "cks-1234567890abcdef0"' --context 100; done
```

PowerShell

```
PS C:\> Get-KMSKeyList | Get-KMSKey | where CustomKeyStoreId -eq
 'cks-1234567890abcdef0'
```

그런 다음 AWS CloudHSM 키 스토어의 연결을 해제합니다. 이 예제 명령은 DisconnectCustomKeyStore 작업을 사용하여 AWS CloudHSM 클러스터에서 AWS CloudHSM 키 스토어의 연결을 해제합니다. 이 명령을 실행하기 앞서 예시에 나온 사용자 지정 키 저장소 ID를 유효한 ID로 대체합니다.

Bash

```
$ aws kms disconnect-custom-key-store --custom-key-store-id cks-1234567890abcdef0
```

PowerShell

```
PS C:\> Disconnect-KMSCustomKeyStore -CustomKeyStoreId cks-1234567890abcdef0
```

사용자 지정 키 스토어가 연결 해제된 후에는 DeleteCustomKeyStore 작업을 사용하여 이를 삭제할 수 있습니다.

Bash

```
$ aws kms delete-custom-key-store --custom-key-store-id cks-1234567890abcdef0
```

PowerShell

```
PS C:\> Remove-KMSCustomKeyStore -CustomKeyStoreId cks-1234567890abcdef0
```

사용자 지정 키 스토어 문제 해결

AWS CloudHSM 키 스토어는 가용성과 복원력을 제공하도록 설계되었습니다. 그러나 AWS CloudHSM 키 스토어를 계속 작동하려면 몇 가지 오류 조건을 복구해야 할 수 있습니다.

주제

- 사용 불가 KMS 키를 수정하는 방법
- 오류가 발생하는 KMS 키를 수정하는 방법
- 연결 오류를 수정하는 방법
- 암호화 작업 실패에 응답하는 방법
- 유효하지 않은 kmsuser 자격 증명을 수정하는 방법
- 불필요한 키 구성 요소를 삭제하는 방법
- KMS 키에서 삭제된 키 구성 요소를 복구하는 방법
- kmsuser로 로그인하는 방법

사용 불가 KMS 키를 수정하는 방법

키 스토어에서 의 AWS CloudHSM 키 상태는 일반적으로 입니다`Enabled`. AWS KMS keys 모든 KMS 키와 마찬가지로 키 스토어에서 KMS 키를 비활성화하거나 삭제를 예약하면 AWS CloudHSM 키 상태가 변경됩니다. 하지만 다른 KMS 키와 달리 사용자 지정 키 스토어의 KMS 키는 키 상태가 `Unavailable`일 수도 있습니다.

키 상태가 `Unavailable`이면 KMS 키가 의도적으로 연결 해제된 사용자 지정 키 스토어에 있고, 연결 실패 시 재연결을 시도한다는 것을 나타냅니다. KMS 키가 사용 불가 상태인 경우에는 KMS 키를 확인 및 관리할 수 있지만, 이를 암호화 작업에 사용할 수 없습니다.

KMS 키의 키 상태를 확인하려면 (고객 관리형 키 페이지에서 KMS 키의 상태 필드를 확인합니다. 또는 DescribeKey 작업을 사용해 응답의 `KeyState` 요소를 확인합니다. 자세한 내용은 키 식별 및 보기을 참조하세요.

연결이 해제된 사용자 지정 키 스토어에서 KMS 키의 키 상태는 `Unavailable` 또는 `PendingDeletion`이 됩니다. 사용자 지정 키 스토어가 연결 해제되었더라도 사용자 지정 키 스토어

에서 삭제가 예약된 KMS 키의 키 상태는 `Pending Deletion`입니다. 따라서 사용자 지정 키 스토어를 재연결하지 않고도 예약된 키 삭제를 취소할 수 있습니다.

사용 불가 상태인 KMS 키를 수정하려면 사용자 지정 키 스토어를 다시 연결합니다. 사용자 지정 키 스토어가 다시 연결되고 나면 사용자 지정 키 스토어의 KMS 키 상태가 이전 상태(`Enabled` 또는 `Disabled`)로 자동 복구됩니다. 삭제 보류 중인 KMS 키는 `PendingDeletion` 상태로 유지됩니다. 하지만 문제가 지속되는 동안 사용 불가 상태인 KMS 키를 활성화 및 비활성화해도 키 상태는 변경되지 않습니다. 이러한 활성화 또는 비활성화 작업은 키가 사용 가능한 상태가 될 때만 효과가 있습니다.

연결 실패에 대한 도움말은 연결 오류를 수정하는 방법 단원을 참조하십시오.

오류가 발생하는 KMS 키를 수정하는 방법

AWS CloudHSM 키 스토어에서 KMS 키를 생성하고 사용하는 데 발생하는 문제는 AWS CloudHSM 키 스토어, 연결된 AWS CloudHSM 클러스터, KMS 키 또는 키 구성 요소의 문제로 인해 발생할 수 있습니다.

AWS CloudHSM 키 스토어가 AWS CloudHSM 클러스터에서 연결 해제되면 사용자 지정 키 스토어에 있는 KMS 키의 키 상태는 입니다`Unavailable`. 연결 해제된 키 스토어에서 KMS AWS CloudHSM 키를 생성하라는 모든 요청은 `CustomKeyStoreInvalidStateException` 예외를 반환합니다. 데이터 키를 암호화, 해독 또는 생성하라는 모든 요청은 `KMSInvalidStateException` 예외 메시지를 반환합니다. 문제를 해결하려면 AWS CloudHSM 키 스토어를 다시 연결합니다.

그러나 AWS CloudHSM 키 스토어에서 암호화 작업에 KMS 키를 사용하려는 시도는 키 상태가 `Enabled`이고 키 AWS CloudHSM 스토어의 연결 상태가 인 경우에도 실패할 수 있습니다`Connected`. 이 오류는 다음 문제 중 하나로 인해 발생할 수 있습니다.

- KMS 키에 대한 키 구성 요소가 연결 AWS CloudHSM 클러스터에서 삭제되었을 수 있습니다. 조사하려면 KMS 키의 키 구성 요소의 키 ID를 확인하고 필요할 경우 키 구성 요소 복구를 시도합니다.
- AWS CloudHSM 키 스토어와 연결된 AWS CloudHSM 클러스터에서 모든 HSMs이 삭제되었습니다. 암호화 작업의 키 스토어에서 KMS AWS CloudHSM 키를 사용하려면 클러스터에 하나 이상의 활성 HSM이 AWS CloudHSM 포함되어야 합니다. AWS CloudHSM 클러스터에서 HSMs의 수와 상태를 확인하려면 AWS CloudHSM 콘솔 또는 DescribeClusters 작업을 사용합니다. DescribeClusters 클러스터에 HSM을 추가하려면 AWS CloudHSM 콘솔 또는 CreateHsm 작업을 사용합니다.
- AWS CloudHSM 키 스토어와 연결된 AWS CloudHSM 클러스터가 삭제되었습니다. 이 문제를 수정하려면 원래 클러스터와 관련된 백업에서 클러스터를 생성합니다(예: 원래 클러스터의 백업이나 원래 클러스터를 생성하는 데 사용된 백업). 그런 다음, 사용자 지정 키 스토어 설정에서 클러스터 ID를 편집합니다. 지침은 KMS 키에서 삭제된 키 구성 요소를 복구하는 방법 섹션을 참조하세요.

- 사용자 지정 키 스토어와 연결된 AWS CloudHSM 클러스터에 사용 가능한 PKCS #11 세션이 없습니다. 이는 일반적으로 트래픽을 처리하기 위해 추가 세션이 필요한 버스트 트래픽이 높은 기간 동안 발생합니다. PKCS #11 세션에 대한 오류 메시지와 함께 `KMSInternalException`에 응답하려면 요청을 백오프하고 다시 시도합니다.

연결 오류를 수정하는 방법

AWS CloudHSM 키 스토어를 클러스터에 연결하려고 하는데 작업이 실패하면 AWS CloudHSM 키 스토어의 연결 상태가 로 변경됩니다`FAILED`. AWS CloudHSM AWS CloudHSM 키 스토어의 연결 상태를 찾으려면 콘솔 또는 DescribeCustomKeyStores 작업을 사용합니다 AWS KMS .

또는, 일부 연결 시도가 쉽게 감지되는 클러스터 구성 오류로 인해 빠르게 실패합니다. 이 경우 연결 상태는 여전히 `DISCONNECTED`입니다. 이러한 실패는 시도가 실패한 이유를 설명하는 오류 메시지 또는 예외를 반환합니다. 예외 설명 및 클러스터 요구 사항을 검토하고, 문제를 해결하고, 필요한 경우 AWS CloudHSM 키 스토어를 업데이트하고, 다시 연결을 시도합니다.

연결 상태가 `FAILED`인 경우 DescribeCustomKeyStores 작업을 실행하고 응답에서 `ConnectionErrorCode` 요소를 확인합니다.

> **Note**
>
> AWS CloudHSM 키 스토어의 연결 상태가 인 경우 다시 연결하기 전에 AWS CloudHSM 키 스토어의 연결을 해제해야 `FAILED`합니다. AWS CloudHSM 키 스토어를 연결 `FAILED` 상태로 연결할 수 없습니다.

- `CLUSTER_NOT_FOUND`는가 지정된 AWS CloudHSM 클러스터 ID로 클러스터를 찾을 AWS KMS 수 없음을 나타냅니다. 잘못된 클러스터 ID가 API 작업에 제공되었거나 클러스터가 삭제되었지만 대체되지 않았기 때문에 이런 문제가 발생할 수 있습니다. 이 오류를 해결하려면 AWS CloudHSM 콘솔 또는 DescribeClusters 작업을 사용하여 클러스터 ID를 확인합니다. 클러스터가 삭제된 경우에는 원래 클러스터의 최신 백업에서 클러스터를 생성합니다. 그런 다음 AWS CloudHSM 키 스토어의 연결을 해제하고 AWS CloudHSM 키 스토어 클러스터 ID 설정을 편집한 다음 AWS CloudHSM 키 스토어를 클러스터에 다시 연결합니다.
- `INSUFFICIENT_CLOUDHSM_HSMS`는 연결된 AWS CloudHSM 클러스터에 HSMs이 포함되어 있지 않음을 나타냅니다. 연결을 하려면 클러스터에 최소 하나의 HSM이 포함되어 있어야 합니다. 클러스터에서 HSM의 수를 확인하려면 DescribeClusters 작업을 사용합니다. 이 오류를 해결하려면 클러스터에 최소 하나의 HSM을 추가합니다. 여러 개의 HSM을 추가하는 경우에는 서로 다른 가용 영역에서 이들을 생성하는 것이 가장 좋습니다.

- `INSUFFICIENT_FREE_ADDRESSES_IN_SUBNET`는 AWS CloudHSM 클러스터와 연결된 하나 이상의 프라이빗 서브넷에 사용 가능한 IP 주소가 없기 때문에가 AWS CloudHSM 키 스토어를 클러스터에 AWS KMS 연결할 수 없음을 나타냅니다. https://docs.aws.amazon.com/cloudhsm/latest/userguide/create-subnets.html AWS CloudHSM 키 스토어 연결에는 연결된 각 프라이빗 서브넷에 1개의 무료 IP 주소가 필요하지만 2개가 선호됩니다.

  기존 서브넷에 IP 주소(CIDR 블록)를 추가할 수 없습니다. 가능한 경우, 서브넷에서 사용 중인 다른 리소스(예: 사용하지 않은 EC2 인스턴스, 탄력적 네트워크 ㅇ니터페이스)를 이동하거나 삭제합니다. 그렇지 않으면 사용 가능한 주소 공간이 더 많은 신규 또는 기존 프라이빗 서브넷이 있는 클러스터의 최근 백업에서 AWS CloudHSM 클러스터를 생성할 수 있습니다. 그런 다음 새 클러스터를 AWS CloudHSM 키 스토어와 연결하려면 사용자 지정 키 스토어의 연결을 해제하고 AWS CloudHSM 키 스토어의 클러스터 ID를 새 클러스터의 ID로 변경한 다음 다시 연결을 시도합니다.

  > **Tip**
  >
  > `kmsuser` 암호 재설정을 방지하려면 AWS CloudHSM 클러스터의 최신 백업을 사용합니다.

- `INTERNAL_ERROR`는가 내부 오류로 인해 요청을 완료할 AWS KMS 수 없음을 나타냅니다. 요청을 다시 시도하세요. `ConnectCustomKeyStore` 요청의 경우 연결을 다시 시도하기 전에 AWS CloudHSM 키 스토어의 연결을 해제합니다.
- `INVALID_CREDENTIALS`는가 올바른 `kmsuser` 계정 암호가 없기 때문에 연결된 AWS CloudHSM 클러스터에 AWS KMS 로그인할 수 없음을 나타냅니다. 이러한 오류에 대한 도움말은 유효하지 않은 `kmsuser` 자격 증명을 수정하는 방법 섹션을 참조하세요.
- `NETWORK_ERRORS`는 일반적으로 일시적인 네트워크 문제를 나타냅니다. AWS CloudHSM 키 스토어의 연결을 해제하고 몇 분 정도 기다린 다음 다시 연결을 시도합니다.
- `SUBNET_NOT_FOUND`는 AWS CloudHSM 클러스터 구성에서 하나 이상의 서브넷이 삭제되었음을 나타냅니다. 가 클러스터 구성에서 모든 서브넷을 찾을 AWS KMS 수 없는 경우 AWS CloudHSM 키 스토어를 AWS CloudHSM 클러스터에 연결하려는 시도가 실패합니다.

  이 오류를 해결하려면 동일한 클러스터의 최근 백업에서 클러스터를 생성합니다. AWS CloudHSM (이 프로세스는 VPC 및 프라이빗 서브넷을 사용하여 새 클러스터 구성을 만듭니다.) 새 클러스터가 사용자 지정 키 스토어에 대한 요구 사항을 충족하는지 확인하고, 새 클러스터 ID를 적어 둡니다. 그런 다음 새 클러스터를 AWS CloudHSM 키 스토어와 연결하려면 사용자 지정 키 스토어의 연결을 해제하고 AWS CloudHSM 키 스토어의 클러스터 ID를 새 클러스터의 ID로 변경한 다음 다시 연결을 시도합니다.

> **Tip**
>
> kmsuser 암호 재설정을 방지하려면 AWS CloudHSM 클러스터의 최신 백업을 사용합니다.

- USER_LOCKED_OUT은 너무 많은 암호 시도의 실패로 인해 kmsuser CU(Crypto User) 계정이 연결된 AWS CloudHSM 클러스터에 액세스할 수 없도록 잠겨 있음을 나타냅니다. 이러한 오류에 대한 도움말은 유효하지 않은 kmsuser 자격 증명을 수정하는 방법 섹션을 참조하세요.

  이 오류를 해결하려면 AWS CloudHSM 키 스토어의 연결을 해제하고 CloudHSM CLI의 사용자 change-password 명령을 사용하여 kmsuser 계정 암호를 변경합니다. 그런 다음, 사용자 지정 키 스토어의 kmsuser 암호 설정을 편집하고 다시 연결을 시도합니다. 도움말은 유효하지 않은 kmsuser 자격 증명을 수정하는 방법 주제에 설명되어 있는 절차를 참조하세요.
- USER_LOGGED_IN는 kmsuser CU 계정이 연결된 AWS CloudHSM 클러스터에 로그인되었음을 나타냅니다. 이렇게 하면가 kmsuser 계정 암호를 교체하고 클러스터에 로그인할 수 AWS KMS 없습니다. 이 오류를 해결하려면 클러스터에서 kmsuser CU를 로그아웃합니다. 클러스터에 로그인하도록 kmsuser 암호를 변경한 경우 키 스토어의 AWS CloudHSM 키 스토어 암호 값도 업데이트해야 합니다. 도움말은 로그아웃 및 재연결 방법를 참조하십시오.
- USER_NOT_FOUND는가 연결된 AWS CloudHSM 클러스터에서 kmsuser CU 계정을 찾을 수 AWS KMS 없음을 나타냅니다. 이 오류를 해결하려면 클러스터에서 kmsuser CU 계정을 생성한 다음 키 스토어의 키 스토어 암호 값을 업데이트합니다. AWS CloudHSM 도움말은 유효하지 않은 kmsuser 자격 증명을 수정하는 방법를 참조하십시오.

암호화 작업 실패에 응답하는 방법

사용자 지정 키 스토어에서 KMS 키를 사용하는 암호화 작업은 KMSInvalidStateException 오류와 함께 실패할 수 있습니다. 다음과 같은 오류 메시지가 KMSInvalidStateException과 함께 나타날 수 있습니다.

KMS는 CloudHSM 클러스터와 통신할 수 없습니다. 일시적인 네트워크 문제일 수 있습니다. 이 오류가 반복적으로 표시되면 AWS CloudHSM 클러스터의 VPC에 대한 네트워크 ACLs 및 보안 그룹 규칙이 올바른지 확인합니다.

- HTTPS 400 오류이지만 일시적인 네트워크 문제로 인해 발생할 수 있습니다. 응답하려면 먼저 요청을 다시 시도합니다. 그러나 계속 실패하면 네트워킹 구성 요소의 구성을 검사하십시오. 이 오류는 나가는 트래픽을 차단하는 방화벽 규칙 또는 VPC 보안 그룹 규칙과 같은 네트워킹 구성 요소의 잘못

된 구성으로 인해 발생할 수 있습니다. 예를 들어 KMS는 IPv6를 통해 AWS CloudHSM 클러스터와 통신할 수 없습니다. 사전 조건에 대한 자세한 내용은 AWS CloudHSM 키 스토어 생성 페이지를 참조하세요.

kmsuser가 잠겨 있기 때문에 KMS가 AWS CloudHSM 클러스터와 통신할 수 없습니다. 이 오류가 반복적으로 표시되면 AWS CloudHSM 키 스토어의 연결을 해제하고 kmsuser 계정 암호를 재설정합니다. 사용자 지정 키 스토어의 kmsuser 암호를 업데이트하고 요청을 다시 시도하세요.

- 이 오류 메시지는 너무 많은 암호 시도의 실패로 인해 kmsuser CU(Crypto User) 계정이 연결된 AWS CloudHSM 클러스터에 액세스할 수 없도록 잠겨 있음을 나타냅니다. 이러한 오류에 대한 도움말은 연결 해제 및 로그인 방법 섹션을 참조하세요.

유효하지 않은 **kmsuser** 자격 증명을 수정하는 방법

AWS CloudHSM 키 스토어를 연결하면가 연결된 AWS CloudHSM 클러스터에 CU(kmsuserCrypto User)로 AWS KMS 로그인합니다. AWS CloudHSM 키 스토어의 연결이 끊어질 때까지 로그인된 상태로 유지됩니다. DescribeCustomKeyStores 응답에서는 `ConnectionState`의 값이 `FAILED`, `ConnectionErrorCode` 값이 `INVALID_CREDENTIALS`로 표시됩니다(아래 예제 참조).

AWS CloudHSM 키 스토어의 연결을 해제하고 `kmsuser` 암호를 AWS KMS 변경하면는 `kmsuser` CU 계정의 자격 증명으로 AWS CloudHSM 클러스터에 로그인할 수 없습니다. 따라서 AWS CloudHSM 키 스토어를 연결하려는 모든 시도가 실패합니다. 다음 예와 같이 `DescribeCustomKeyStores` 응답은 `ConnectionState` 값을 `FAILED`로 표시하고 `ConnectionErrorCode` 값을 `INVALID_CREDENTIALS`로 표시합니다.

```
$ aws kms describe-custom-key-stores --custom-key-store-name ExampleKeyStore
{
   "CustomKeyStores": [
      "CloudHsmClusterId": "cluster-1a23b4cdefg",
      "ConnectionErrorCode": "INVALID_CREDENTIALS"
      "CustomKeyStoreId": "cks-1234567890abcdef0",
      "CustomKeyStoreName": "ExampleKeyStore",
      "TrustAnchorCertificate": "<certificate string appears here>",
      "CreationDate": "1.499288695918E9",
      "ConnectionState": "FAILED"
   ],
}
```

또한 올바르지 않은 암호로 클러스터에 로그인하려는 시도가 5번 실패하면 AWS CloudHSM 이 사용자 계정을 잠급니다. 클러스터에 로그인하려면 계정 암호를 변경해야 합니다.

가 클러스터에 kmsuser CU로 로그인하려고 할 때 잠금 응답을 AWS KMS 받으면 AWS CloudHSM 키 스토어 연결 요청이 실패합니다. DescribeCustomKeyStores 응답에서 ConnectionState의 값은 FAILED, ConnectionErrorCode의 값은 USER_LOCKED_OUT입니다(아래 예제 참조).

```
$ aws kms describe-custom-key-stores --custom-key-store-name ExampleKeyStore
{
   "CustomKeyStores": [
      "CloudHsmClusterId": "cluster-1a23b4cdefg",
      "ConnectionErrorCode": "USER_LOCKED_OUT"
      "CustomKeyStoreId": "cks-1234567890abcdef0",
      "CustomKeyStoreName": "ExampleKeyStore",
      "TrustAnchorCertificate": "<certificate string appears here>",
      "CreationDate": "1.499288695918E9",
      "ConnectionState": "FAILED"
   ],
}
```

이러한 조건 중 하나를 복구하려면 다음 절차를 따르세요.

1. AWS CloudHSM 키 스토어의 연결을 해제합니다.
2. DescribeCustomKeyStores 작업을 실행하고 응답에서 ConnectionErrorCode 요소의 값을 확인합니다.

   - ConnectionErrorCode 값이 INVALID_CREDENTIALS인 경우에는 kmsuser 계정의 현재 암호를 확인합니다. 필요한 경우, CloudHSM CLI에서 user change-password 명령을 사용하여 암호를 알려진 값으로 설정합니다.
   - ConnectionErrorCode 값이 USER_LOCKED_OUT인 경우, CloudHSM CLI에서 user change-password 명령을 사용하여 kmsuser 암호를 변경해야 합니다.
3. 클러스터의 현재 kmsuser 암호와 일치하도록 kmsuser 암호 설정을 편집합니다. 이 작업은 AWS KMS 에 클러스터에 로그인하기 위해 사용하는 암호를 알려줍니다. 클러스터의 kmsuser 암호는 변경되지 않습니다.
4. 사용자 지정 키 스토어를 연결합니다.

불필요한 키 구성 요소를 삭제하는 방법

AWS CloudHSM 키 스토어에서 KMS 키 삭제를 예약한 후 연결된 AWS CloudHSM 클러스터에서 해당 키 구성 요소를 수동으로 삭제해야 할 수 있습니다.

AWS CloudHSM 키 스토어에서 KMS 키를 생성하면는에서 KMS 키 메타데이터를 AWS KMS 생성하고 연결된 AWS CloudHSM 클러스터에서 키 구성 요소를 AWS KMS 생성합니다. AWS CloudHSM 키 스토어에서 KMS 키 삭제를 예약할 때 대기 기간이 지나면가 KMS 키 메타데이터를 AWS KMS 삭제합니다. 그런 다음 AWS CloudHSM 클러스터에서 해당 키 구성 요소를 삭제하기 위해 AWS KMS 최선을 다합니다. AWS CloudHSM 키 스토어와의 연결이 끊어지거나 `kmsuser` 암호가 변경되는 경우와 같이가 클러스터에 액세스할 수 AWS KMS 없는 경우 시도가 실패할 수 있습니다. AWS KMS 는 클러스터 백업에서 키 구성 요소를 삭제하려고 시도하지 않습니다.

AWS KMS 는 AWS CloudTrail 로그의 `DeleteKey` 이벤트 항목에 있는 클러스터에서 키 구성 요소를 삭제하려는 시도의 결과를 보고합니다. 이는 다음 예제 항목에서 보듯이 `additionalEventData` 요소의 `backingKeysDeletionStatus` 요소에 표시됩니다. 항목에는 KMS 키 ARN, AWS CloudHSM 클러스터 ID 및 키 구성 요소의 ID(`backing-key-id`)도 포함됩니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "accountId": "111122223333",
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2021-12-10T14:23:51Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DeleteKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": null,
    "responseElements":  {
        "keyId":"arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "additionalEventData": {
        "customKeyStoreId": "cks-1234567890abcdef0",
        "clusterId": "cluster-1a23b4cdefg",
        "backingKeys": "[{\"backingKeyId\":\"backing-key-id\"}]",
        "backingKeysDeletionStatus": "[{\"backingKeyId\":\"backing-key-id\",
\"deletionStatus\":\"FAILURE\"}]"
    },
```

```
    "eventID": "c21f1f47-f52b-4ffe-bff0-6d994403cf40",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:eu-
west-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsServiceEvent",
    "recipientAccountId": "111122223333",
    "managementEvent": true,
    "eventCategory": "Management"
}
```


> **참고**
>
> 다음 절차에서는 AWS CloudHSM 클라이언트 SDK 5 명령줄 도구인 CloudHSM CLI를 사용합니다. CloudHSM CLI에서 `key-handle`을 `key-reference`로 대체합니다.
> 2025년 1월 1일에 AWS CloudHSM 는 클라이언트 SDK 3 명령줄 도구, CloudHSM 관리 유틸리티(CMU) 및 키 관리 유틸리티(KMU)에 대한 지원을 종료합니다. Client SDK 3 명령줄 도구와 Client SDK 5 명령줄 도구의 차이점에 대한 자세한 내용은 AWS CloudHSM 사용 설명서의 클라이언트 SDK 3 CMU 및 KMU에서 클라이언트 SDK 5 CloudHSM CLI로 마이그레이션을 참조하세요.

다음 절차에서는 연결된 AWS CloudHSM 클러스터에서 분리된 키 구성 요소를 삭제하는 방법을 보여줍니다.

1. AWS CloudHSM 키 스토어가 아직 연결 해제되지 않은 경우에 설명된 대로 로그인합니다연결 해제 및 로그인 방법.

   > **Note**
   >
   > 사용자 지정 키 스토어의 연결이 해제된 상태에서는 사용자 지정 키 스토어에서 KMS 키를 생성하거나, 암호화 작업을 위해 기존 KMS 키를 사용하려는 모든 시도가 실패합니다. 이 작업은 사용자가 기밀 데이터를 저장하거나 액세스하지 못하도록 차단합니다.

2. CloudHSM CLI의 key delete 명령을 사용하여 클러스터의 HSM에서 키를 삭제합니다.

AWS CloudHSM 키 스토어에서 KMS 키를 사용하는 암호화 작업에 대한 모든 CloudTrail 로그 항목에는 `customKeyStoreId` 및가 있는 `additionalEventData` 필드가 포함됩니다.backingKey. `backingKeyId` 필드에 반환되는 값은 CloudHSM 키 `id` 속성입니다. `id`를 기준으로 key delete 작업을 필터링하여 CloudTrail 로그에서 식별한 분리된 키 구성 요소를 삭제하는 것이 좋습니다.

AWS CloudHSM 는 `backingKeyId` 값을 16진수 값으로 인식합니다. `id`를 기준으로 필터링하려면 `0x` 앞에 `backingKeyId`를 추가해야 합니다. 예를 들어 CloudTrail 로그의 `backingKeyId`가 `1a2b3c45678abcdef`인 경우 **`0x`**`1a2b3c45678abcdef`를 기준으로 필터링합니다.

다음 예제에서는 클러스터의 HSMs에서 키를 삭제합니다. `backing-key-id`는 CloudTrail 로그 항목에 나열됩니다. 이 명령을 실행하기 전에 예제 키 `backing-key-id`를 계정의 유효한 값으로 바꿉니다.

```
aws-cloudhsm key delete --filter attr.id="0x<backing-key-id>"
{
  "error_code": 0,
  "data": {
    "message": "Key deleted successfully"
  }
}
```

3. 에 설명된 대로 AWS CloudHSM 키 스토어를 로그아웃하고 다시 연결합니다로그아웃 및 재연결 방법.

KMS 키에서 삭제된 키 구성 요소를 복구하는 방법

의 키 구성 요소가 AWS KMS key 삭제되면 KMS 키를 사용할 수 없으며 KMS 키로 암호화된 모든 사이퍼텍스트를 복호화할 수 없습니다. 이는 키 스토어의 KMS 키에 대한 AWS CloudHSM 키 구성 요소가 연결된 AWS CloudHSM 클러스터에서 삭제될 때 발생할 수 있습니다. 하지만 키 구성 요소를 복구하는 것이 가능할 수 있습니다.

AWS CloudHSM 키 스토어에서 AWS KMS key (KMS 키)를 생성하면는 연결된 AWS CloudHSM 클러스터에 AWS KMS 로그인하고 KMS 키에 대한 키 구성 요소를 생성합니다. 또한 AWS CloudHSM 키 스토어가 연결되어 있는 동안 자신만 알고 로그인한 상태로 유지되는 값으로 암호를 변경합니다. 키 소유자이자 키를 생성한 CU만 키를 삭제할 수 있기 때문에 HSM에서 실수로 키가 삭제될 일이 없습니다.

하지만 KMS 키의 키 구성 요소가 클러스터의 HSM에서 삭제된 경우 KMS 키 상태가 궁극적으로 `UNAVAILABLE`로 변경됩니다. 암호화 작업에서 KMS 키를 사용하려고 시도하면

KMSInvalidStateException 예외 메시지와 함께 작업이 실패합니다. 가장 중요한 것은 KMS 키 하에서 암호화된 데이터는 해독이 불가능합니다.

특정 상황에서 키 구성 요소가 포함된 백업에서 클러스터를 생성하여 삭제된 키 구성 요소를 복구할 수 있습니다. 이 전략은 삭제 전 키가 존재한 동안 최소 하나의 백업이 생성되었을 때만 효과가 있습니다.

다음 프로세스를 이용해 키 구성 요소를 복구합니다.

1. 키 구성 요소가 포함된 클러스터 백업을 찾습니다. 백업에는 클러스터와 암호화된 데이터를 지원하는 데 필요한 모든 사용자와 키도 포함되어 있어야 합니다.

   DescribeBackups 작업을 사용해 클러스터의 백업을 나열합니다. 그런 다음, 백업을 선택하는 데 도움이 되는 백업 타임스탬프를 사용합니다. 출력을 AWS CloudHSM 키 스토어와 연결된 클러스터로 제한하려면 다음 예제와 같이 Filters 파라미터를 사용합니다.

   ```
   $ aws cloudhsmv2 describe-backups --filters clusterIds=<cluster ID>
   {
       "Backups": [
           {
               "ClusterId": "cluster-1a23b4cdefg",
               "BackupId": "backup-9g87f6edcba",
               "CreateTimestamp": 1536667238.328,
               "BackupState": "READY"
           },
                ...
       ]
   }
   ```

2. 선택한 백업에서 클러스터를 생성합니다. 백업에 삭제된 키와 기타 사용자 및 클러스터에서 필요한 키가 포함되어 있는지 확인합니다.
3. 속성을 편집할 수 있도록 AWS CloudHSM 키 스토어의 연결을 해제합니다.
4. AWS CloudHSM 키 스토어의 클러스터 ID를 편집합니다. 백업에서 생성한 클러스터의 클러스터 ID를 입력합니다. 클러스터가 원래 클러스터와 백업 기록을 공유하기 때문에 새 클러스터 ID는 유효해야 합니다.
5. AWS CloudHSM 키 스토어를 다시 연결합니다.

`kmsuser`로 로그인하는 방법

키 스토어 AWS KMS 의 AWS CloudHSM 클러스터에서 AWS CloudHSM 키 구성 요소를 생성하고 관리하기 위해 kmsuser는 CU(Crypto User) 계정을 사용합니다. 클러스터에서 kmsuser CU 계정을 생성하고 AWS CloudHSM 키 스토어를 생성할 AWS KMS 때 해당 암호를에 제공합니다.

일반적으로는 kmsuser 계정을 AWS KMS 관리합니다. 그러나 일부 작업의 경우 AWS CloudHSM 키 스토어의 연결을 해제하고, 클러스터에 kmsuser CU로 로그인하고, CloudHSM 명령줄 인터페이스(CLI)를 사용해야 합니다.

> **Note**
>
> 사용자 지정 키 스토어의 연결이 해제된 상태에서는 사용자 지정 키 스토어에서 KMS 키를 생성하거나, 암호화 작업을 위해 기존 KMS 키를 사용하려는 모든 시도가 실패합니다. 이 작업은 사용자가 기밀 데이터를 저장하거나 액세스하지 못하도록 차단합니다.

이 주제에서는 AWS CloudHSM 키 스토어의 연결을 해제하고 로 로그인하고kmsuser, AWS CloudHSM 명령줄 도구를 실행하고, AWS CloudHSM 키 스토어를 로그아웃했다가 다시 연결하는 방법을 설명합니다.

주제

- 연결 해제 및 로그인 방법
- 로그아웃 및 재연결 방법

연결 해제 및 로그인 방법

연결 클러스터에 kmsuser Crypto User(CU)로 로그인해야 할 때마다 다음 절차를 따릅니다.

> **참고**
>
> 다음 절차에서는 AWS CloudHSM 클라이언트 SDK 5 명령줄 도구인 CloudHSM CLI를 사용합니다. CloudHSM CLI에서 `key-handle`을 `key-reference`로 대체합니다.
> 2025년 1월 1일에 AWS CloudHSM 는 클라이언트 SDK 3 명령줄 도구, CloudHSM 관리 유틸리티(CMU) 및 키 관리 유틸리티(KMU)에 대한 지원을 종료합니다. Client SDK 3 명령줄 도구와 Client SDK 5 명령줄 도구의 차이점에 대한 자세한 내용은 AWS CloudHSM 사용 설명서의 클라이언트 SDK 3 CMU 및 KMU에서 클라이언트 SDK 5 CloudHSM CLI로 마이그레이션을 참조하세요.

1. 아직 연결 해제되지 않은 경우 AWS CloudHSM 키 스토어의 연결을 해제합니다. AWS KMS 콘솔 또는 AWS KMS API를 사용할 수 있습니다.

   AWS CloudHSM 키가 연결되어 있는 동안 AWS KMS 는 로 로그인됩니다`kmsuser`. 따라서 `kmsuser`로 로그인하거나 `kmsuser` 암호를 변경할 수 없습니다.

   예를 들어 이 명령은 DisconnectCustomKeyStore를 사용해 예제 키 스토어의 연결을 해제합니다. 예제 AWS CloudHSM 키 스토어 ID를 유효한 키 스토어 ID로 바꿉니다.

   ```
   $ aws kms disconnect-custom-key-store --custom-key-store-id cks-1234567890abcdef0
   ```

2. login 명령을 사용하고 관리자로 로그인합니다. AWS CloudHSM 사용 설명서의 CloudHSM CLI 사용 섹션에 설명된 절차를 사용합니다.

   ```
   aws-cloudhsm > login --username admin --role admin
             Enter password:
   {
     "error_code": 0,
     "data": {
       "username": "admin",
       "role": "admin"
     }
   }
   ```

3. CloudHSM CLI의 사용자 change-password 명령을 사용하여 `kmsuser` 계정의 암호를 알고 있는 암호로 변경합니다(AWS CloudHSM 키 스토어를 연결할 때에서 암호AWS KMS 교체). 암호는 7~32자의 영숫자로만 구성되어야 합니다. 대소문자가 구분되며 어떤 특수 문자도 포함해서는 안 됩니다.

4. 설정한 암호를 사용하여 `kmsuser`로 로그인합니다. 자세한 지침은 AWS CloudHSM 사용 설명서의 시작하기 섹션을 참조하세요.

   ```
   aws-cloudhsm > login --username kmsuser --role crypto-user
             Enter password:
   {
     "error_code": 0,
     "data": {
       "username": "kmsuser",
       "role": "crypto-user"
     }
   }
   ```

로그아웃 및 재연결 방법

kmsuser Crypto User(CU)로 로그아웃하고 키 저장소를 다시 연결해야 할 때마다 다음 절차를 따르세요.

> **참고**
>
> 다음 절차에서는 AWS CloudHSM 클라이언트 SDK 5 명령줄 도구인 CloudHSM CLI를 사용합니다. CloudHSM CLI에서 `key-handle`을 `key-reference`로 대체합니다.
> 2025년 1월 1일에 AWS CloudHSM 는 클라이언트 SDK 3 명령줄 도구, CloudHSM 관리 유틸리티(CMU) 및 키 관리 유틸리티(KMU)에 대한 지원을 종료합니다. Client SDK 3 명령줄 도구와 Client SDK 5 명령줄 도구의 차이점에 대한 자세한 내용은 AWS CloudHSM 사용 설명서의 클라이언트 SDK 3 CMU 및 KMU에서 클라이언트 SDK 5 CloudHSM CLI로 마이그레이션을 참조하세요.

1. 작업을 수행한 다음, CloudHSM CLI의 logout 명령을 사용하여 로그아웃합니다. 로그아웃하지 않으면 AWS CloudHSM 키 스토어 재연결 시도가 실패합니다.

```
aws-cloudhsm  logout
{
  "error_code": 0,
  "data": "Logout successful"
}
```

2. 사용자 지정 키 스토어에 대한 kmsuser 암호 설정을 편집합니다.

   클러스터 AWS KMS 의 kmsuser에 대한 현재 암호를 알려줍니다. 이 단계를 생략하면 AWS KMS 가 클러스터에 로 로그인할 수 없으며 kmsuser사용자 지정 키 스토어를 다시 연결하려는 모든 시도가 실패합니다. AWS KMS 콘솔 또는 UpdateCustomKeyStore 작업의 `KeyStorePassword` 파라미터를 사용할 수 있습니다.

   예를 들어이 명령은 현재 암호가 AWS KMS 임을 알려줍니다`tempPassword`. 예제 암호를 실제 암호로 바꿉니다.

```
$ aws kms update-custom-key-store --custom-key-store-id cks-1234567890abcdef0 --
key-store-password tempPassword
```

3. AWS KMS 키 스토어를 AWS CloudHSM 클러스터에 다시 연결합니다. 예제 AWS CloudHSM 키 스토어 ID를 유효한 키 스토어 ID로 바꿉니다. 연결 프로세스 중에는 암호만 알고 있는 값으로 `kmsuser` 암호를 AWS KMS 변경합니다.

   ConnectCustomKeyStore 작업이 신속하게 값을 반환하지만, 연결 프로세스의 소요 시간이 연장될 수 있습니다. 하지만 이러한 초기 응답은 연결 프로세스가 성공했음을 의미하지는 않습니다.

   ```
   $ aws kms connect-custom-key-store --custom-key-store-id cks-1234567890abcdef0
   ```

4. DescribeCustomKeyStores 작업을 사용하여 AWS CloudHSM 키 스토어가 연결되어 있는지 확인합니다. 예제 AWS CloudHSM 키 스토어 ID를 유효한 키 스토어 ID로 바꿉니다.

   이 예제에서 연결 상태 필드는 AWS CloudHSM 키 스토어가 이제 연결되었음을 보여줍니다.

   ```
   $ aws kms describe-custom-key-stores --custom-key-store-
   id cks-1234567890abcdef0
   {
      "CustomKeyStores": [
         "CustomKeyStoreId": "cks-1234567890abcdef0",
         "CustomKeyStoreName": "ExampleKeyStore",
         "CloudHsmClusterId": "cluster-1a23b4cdefg",
         "TrustAnchorCertificate": "<certificate string appears here>",
         "CreationDate": "1.499288695918E9",
         "ConnectionState": "CONNECTED"
      ],
   }
   ```

외부 키 스토어

외부 키 스토어를 사용하면 외부의 암호화 키를 사용하여 AWS 리소스를 보호할 수 있습니다 AWS. 이 고급 기능은 사용자가 제어하는 외부 키 관리 시스템에 저장된 암호화 키로 보호해야 하는 규제 대상 워크로드용으로 설계되었습니다. 외부 키 스토어는 외부에서 소유하고 제어하는 키 구성 요소로 암호화할 수 있는 기능을 AWS포함하여의 데이터에 대한 주권을 부여하는 AWS 디지털 주권 서약을 지원합니다 AWS.

외부 키 스토어는 외부에서 소유하고 관리하는 외부 키 관리자가 지원하는 사용자 지정 키 스토어입니다 AWS. 외부 키 관리자는 물리적 또는 가상 하드웨어 보안 모듈(HSM)이거나 암호화 키를 생성하고 사용할 수 있는 하드웨어 기반 또는 소프트웨어 기반 시스템일 수 있습니다. 외부 키 스토어에서 KMS

키를 사용하는 암호화 및 복호화 작업은 HYOK(Hold Your Own Key)라는 기능인 암호화 키 구성 요소를 사용하여 외부 키 관리자에 의해 수행됩니다.

AWS KMS 는 외부 키 관리자와 직접 상호 작용하지 않으며 키를 생성, 보기, 관리 또는 삭제할 수 없습니다. 대신는 사용자가 제공하는 외부 키 스토어 프록시(XKS 프록시) 소프트웨어와만 AWS KMS 상호 작용합니다. 외부 키 스토어 프록시는 AWS KMS 와 외부 키 관리자 간의 모든 통신을 중재합니다. 의 모든 요청을 외부 키 관리자 AWS KMS 로 전송하고 외부 키 관리자의 응답을 로 다시 전송합니다 AWS KMS. 또한 외부 키 스토어 프록시는의 일반 요청을 외부 키 관리자가 이해할 수 있는 AWS KMS 공급업체별 형식으로 변환하므로 다양한 공급업체의 키 관리자와 함께 외부 키 스토어를 사용할 수 있습니다.

AWS Encryption SDK를 포함하여 클라이언트측 암호화를 위해 외부 키 스토어에서 KMS 키를 사용할 수 있습니다. 그러나 외부 키 스토어는 서버 측 암호화에 중요한 리소스이므로 외부의 암호화 키를 사용하여 여러에서 AWS 리소스를 보호할 AWS 서비스 수 있습니다 AWS. AWS 서비스 대칭 암호화를 위한 고객 관리형 키를 지원하는는 외부 키 스토어의 KMS 키도 지원합니다. 서비스 지원에 관한 자세한 내용을 확인하려면 AWS 서비스 통합을 참조하세요.

외부 키 스토어를 사용하면 외부에서 암호화 키를 저장하고 사용해야 하는 규제된 워크로드 AWS KMS 에를 사용할 수 있습니다 AWS. 그러나 이는 표준 공유 책임 모델에서 크게 벗어나며 추가 운영 부담이 필요합니다. 대부분의 고객에게 가용성 및 지연 시간에 대한 더 큰 위험은 외부 키 스토어의 인식된 보안 이점을 초과합니다.

외부 키 스토어를 사용하면 신뢰할 수 있는 루트를 제어할 수 있습니다. 외부 키 스토어의 KMS 키로 암호화된 데이터는 사용자가 제어하는 외부 키 관리자를 통해서만 복호화할 수 있습니다. 외부 키 스토어 연결을 해제하거나 외부 키 스토어 프록시에서 외부 키 관리자 연결을 해제하는 등 외부 키 관리자에 대한 액세스를 일시적으로 취소하면 복원할 때까지 암호화 키에 대한 모든 액세스 권한이 AWS 상실됩니다. 이 기간 동안에는 KMS 키로 암호화된 사이퍼텍스트를 복호화할 수 없습니다. 외부 키 관리자에 대한 액세스를 영구적으로 취소하면 외부 키 스토어의 KMS 키로 암호화된 모든 사이퍼텍스트를 복구할 수 없게 됩니다. 유일한 예외는 KMS 키로 보호되는 데이터 키를 잠시 캐싱하는 AWS 서비스입니다. 이러한 데이터 키는 리소스를 비활성화하거나 캐시가 만료될 때까지 계속 작동합니다. 자세한 내용은 사용할 수 없는 KMS 키가 데이터 키에 미치는 영향을 참조하세요.

외부 키 스토어는 암호화 키가 전적으로 사용자의 제어 하에 있고 액세스할 수 없는 규제된 워크로드의 몇 가지 사용 사례를 차단 해제합니다 AWS. 그러나 이는 클라우드 기반 인프라 운영 방식의 큰 변화이자 공동 책임 모델의 중요한 변화입니다. 대부분의 워크로드에서 가용성과 성능에 대한 추가 운영 부담과 더 큰 위험은 외부 키 스토어의 인식된 보안 이점을 초과합니다.

외부 키 스토어가 필요한가요?

외부 키 스토어가 요구 사항에 적합한지 확인하려면 섹션을 참조하세요올바른 키 스토어 선택.

공동 책임 모델

표준 KMS 키는가 AWS KMS 소유하고 관리하는 HSMs에서 생성되고 사용되는 키 구성 요소를 사용합니다. KMS 키에 대한 액세스 제어 정책을 설정하고 KMS 키를 사용하여 리소스를 보호 AWS 서비스 하도록를 구성합니다.는 KMS 키에 있는 키 구성 요소의 보안, 가용성, 지연 시간 및 내구성에 대한 책임을 AWS KMS 집니다.

외부 키 스토어의 KMS 키는 외부 키 관리자의 키 구성 요소 및 작업에 의존합니다. 따라서 책임의 균형이 사용자의 방향으로 바뀝니다. 외부 키 관리자에서 암호화 키의 보안, 신뢰성, 내구성 및 성능에 대한 책임은 사용자에게 있습니다. AWS KMS 는 요청에 즉시 응답하고 외부 키 스토어 프록시와 통신하고 보안 표준을 유지할 책임이 있습니다. 모든 외부 키가 최소한 표준 사이퍼텍스트보다 강력한 AWS KMS 사이퍼텍스트를 저장하도록 하기 위해는 AWS KMS 먼저 KMS 키와 관련된 AWS KMS 키 구성 요소로 모든 일반 텍스트를 암호화한 다음 이중 암호화라고 하는 외부 키로 암호화하기 위해 외부 키 관리자에게 전송합니다. 결과적으로 AWS KMS 나 외부 키 구성 요소의 소유자는 이중 암호화된 사이퍼텍스트를 단독으로 복호화할 수 없습니다.

규정 및 성능 표준을 충족하는 외부 키 관리자를 유지 관리하고, AWS KMS 외부 키 스토어 프록시 API 사양을 준수하는 외부 키 스토어 프록시를 제공 및 유지 관리하고, 키 구성 요소의 가용성 및 내구성을 보장할 책임은 사용자에게 있습니다. 또한 외부 키 스토어를 생성, 구성 및 유지 관리해야 합니다. 유지 관리하는 구성 요소로 인해 오류가 발생하는 경우 서비스가 과도한 중단 없이 리소스에 액세스할 AWS 수 있도록 오류를 식별하고 해결할 준비가 되어 있어야 합니다.는 문제의 원인과 가장 가능성이 높은 해결 방법을 결정하는 데 도움이 되는 문제 해결 지침을 AWS KMS 제공합니다.

외부 키 스토어에 대해가 기록하는 AWS KMS Amazon CloudWatch 지표 및 차원을 검토합니다. AWS KMS 에서는 성능 및 운영 문제의 초기 징후를 발생하기 전에 감지할 수 있도록 외부 키 스토어를 모니터링하는 CloudWatch 경보를 생성할 것을 강력히 권장합니다.

무엇이 바뀌고 있나요?

외부 키 스토어는 대칭 암호화 KMS 키만 지원합니다. 내에서 AWS KMS는 액세스 제어 정책 설정 및 키 사용 모니터링을 포함하여 다른 고객 관리형 키를 관리하는 것과 거의 동일한 방식으로 외부 키 스토어에서 KMS 키를 사용하고 관리합니다. 동일한 파라미터와 동일한 API를 사용하여 모든 KMS 키에 사용하는 외부 키 스토어에서 KMS 키로 암호화 작업을 요청합니다. 가격도 표준 KMS 키와 동일합니다. 자세한 내용은 외부 키 저장소의 KMS 키 및 AWS Key Management Service 요금을 참조하세요.

그러나 외부 키 스토어를 사용하면 다음 원칙이 변경됩니다.

- 키 작업의 가용성, 내구성 및 지연 시간에 대한 책임은 사용자에게 있습니다.
- 외부 키 관리자 시스템의 개발, 구매, 운영 및 라이선스에 대한 모든 비용은 사용자의 책임입니다.
- 에서 외부 키 스토어 프록시로의 모든 요청에 대해 독립적인 권한 부여 AWS KMS 를 구현할 수 있습니다.
- 외부 키 스토어 프록시의 모든 작업과 AWS KMS 요청과 관련된 외부 키 관리자의 모든 작업을 모니터링, 감사 및 로깅할 수 있습니다.

어디에서 시작합니까?

외부 키 스토어를 생성하고 관리하려면 외부 키 스토어 프록시 연결 옵션을 선택하고, 사전 조건을 수집하고, 외부 키 스토어를 생성하고 구성해야 합니다.

할당량

AWS KMS 는 연결 상태에 관계없이 키 스토어와 외부 키 스토어를 모두 포함하여 각 및 리전에서 최대 10개의 사용자 지정 키 스토어를 허용합니다. AWS 계정 AWS CloudHSM 또한 외부 키 스토어에 KMS 키 사용에 대한 AWS KMS 요청 할당량이 있습니다.

외부 키 스토어 프록시에 대해 VPC 프록시 연결을 선택하는 경우 VPC, 서브넷 및 Network Load Balancer와 같은 필수 구성 요소에 할당량이 있을 수도 있습니다. 이러한 할당량에 대한 자세한 내용을 보려면 Service Quotas 콘솔을 사용합니다.

리전

네트워크 지연 시간을 최소화하려면 외부 키 관리자와 가장 가까운 AWS 리전 에 외부 키 스토어 구성 요소를 생성합니다. 가능하면 네트워크 왕복 시간(RTT)이 35밀리초 이하인 리전을 선택합니다.

외부 키 스토어는 중국(베이징) 및 중국(닝샤)을 제외하고 AWS KMS 가 지원되는 모든 AWS 리전 에서 지원됩니다.

지원되지 않는 기능

AWS KMS 는 사용자 지정 키 스토어에서 다음 기능을 지원하지 않습니다.

- 비대칭 KMS 키
- HMAC KMS 키

- 가져온 키 구성 요소가 있는 KMS 키
- 자동 키 교체
- 다중 리전 키

자세히 알아보기:

- AWS News Blog의 AWS KMS KMS 외부 키 스토어 발표

외부 키 스토어 개념

외부 키 저장소에서 사용하는 기본 용어와 개념에 대해 알아봅니다.

외부 키 스토어

외부 키 스토어는 사용자가 소유하고 관리하는 외부 키 관리자가 지원하는 AWS KMS 사용자 지정 키 스토어 AWS 입니다. 외부 키 스토어의 각 KMS 키는 외부 키 관리자의 외부 키와 연결됩니다. 암호화 또는 복호화를 위해 외부 키 스토어에서 KMS 키를 사용하는 경우 HYOK(Hold your Own Keys)로 알려진 배열인 외부 키를 사용하여 외부 키 관리자에서 작업이 수행됩니다. 이 기능은 자체 외부 키 관리자에서 암호화 키를 유지해야 하는 조직을 위해 설계되었습니다.

외부 키 스토어는 AWS 리소스를 보호하는 암호화 키와 작업이 사용자가 제어하는 외부 키 관리자에 남아 있도록 합니다.는 데이터를 암호화 및 해독하기 위해 외부 키 관리자에게 요청을 AWS KMS 보내지만 외부 키를 생성, 삭제 또는 관리할 AWS KMS 수는 없습니다. 에서 외부 키 관리자 AWS KMS 로의 모든 요청은 사용자가 제공, 소유 및 관리하는 외부 키 스토어 프록시 소프트웨어 구성 요소에 의해 조정됩니다.

AWS AWS KMS 고객 관리형 키를 지원하는 서비스는 외부 키 스토어의 KMS 키를 사용하여 데이터를 보호할 수 있습니다. 결과적으로 데이터는 궁극적으로 외부 키 관리자의 암호화 작업을 사용하여 키로 보호됩니다.

외부 키 스토어의 KMS 키는 표준 KMS 키와 신뢰 모델, 공동 규정 배열 및 성능 기대치가 근본적으로 다릅니다. 외부 키 스토어 사용 시 키 구성 요소 및 암호화 작업의 보안 및 무결성에 대한 책임은 사용자에게 있습니다. 외부 키 스토어의 KMS 키 가용성 및 지연 시간은 하드웨어, 소프트웨어, 네트워킹 구성 요소 및 AWS KMS 와 외부 키 관리자 간 거리의 영향을 받습니다. 또한 외부 키 관리자와 통신하는 데 필요한 네트워킹 및 로드 밸런싱 인프라에 대해 추가 비용이 발생할 수 있습니다. AWS KMS

보다 광범위한 데이터 보호 전략의 일부로 외부 키 스토어를 사용할 수 있습니다. 보호하는 각 AWS 리소스에 대해 외부 키 스토어에 KMS 키가 필요한 리소스와 표준 KMS 키로 보호할 수 있는 리소스를 결

정할 수 있습니다. 이를 통해 특정 데이터 분류, 애플리케이션 또는 프로젝트에 대한 KMS 키를 유연하게 선택할 수 있습니다.

외부 키 관리자

외부 키 관리자는 256비트 AES 대칭 키를 생성하고 대칭 암호화 및 복호화를 수행할 수 있는 AWS 외부의 구성 요소입니다. 외부 키 스토어의 외부 키 관리자는 물리적 하드웨어 보안 모듈(HSM), 가상 HSM 또는 HSM 구성 요소가 있거나 없는 소프트웨어 키 관리자일 수 있습니다. 온프레미스 AWS, 로컬 또는 원격 데이터 센터, 클라우드 등 외부 어디에나 위치할 수 있습니다. 단일 외부 키 관리자 또는 암호화 키를 공유하는 여러 관련 키 관리자 인스턴스(예: HSM 클러스터)에서 외부 키 스토어를 지원할 수 있습니다. 외부 키 스토어는 여러 공급업체의 다양한 외부 관리자를 지원하도록 설계되었습니다. 외부 키 관리자에 연결하는 방법에 대한 자세한 내용은 외부 키 저장소 프록시 연결 옵션 선택 단원을 참조하세요.

외부 키

외부 키 스토어의 각 KMS 키는 외부 키라고 하는 외부 키 관리자의 암호화 키와 연결됩니다. 외부 키 스토어에서 KMS 키로 암호화하거나 복호화하면 외부 키를 사용하여 외부 키 관리자에서 암호화 작업이 수행됩니다.

> **Warning**
>
> 외부 키는 KMS 키 작동에 필수적입니다. 외부 키가 분실되거나 삭제되면 연결된 KMS 키로 암호화된 사이퍼텍스트를 복구할 수 없습니다.

외부 키 스토어의 경우 외부 키는 활성화되고 암호화와 복호화를 수행할 수 있는 256비트 AES 키여야 합니다. 자세한 외부 키 요구 사항은 외부 키 스토어의 KMS 키 요구 사항 섹션을 참조하세요.

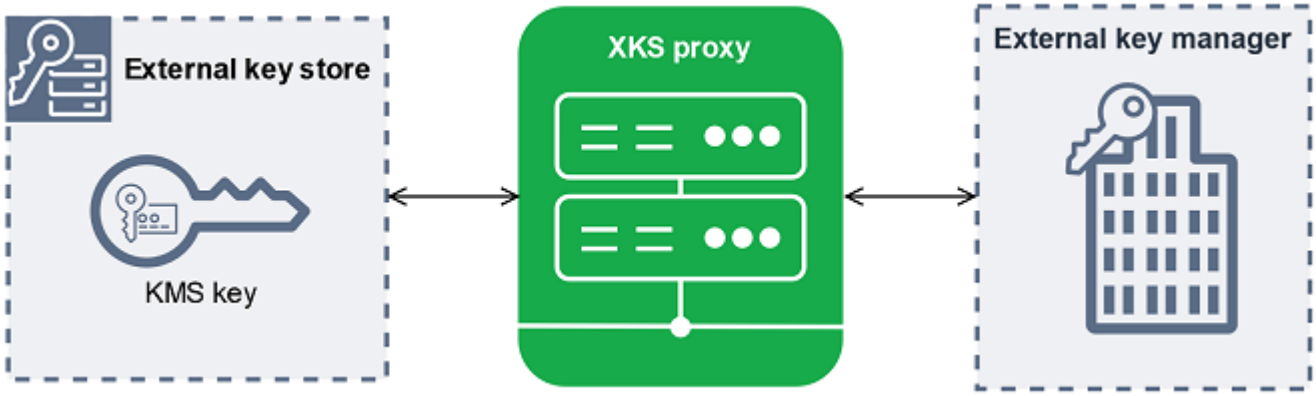
AWS KMS 는 외부 키를 생성, 삭제 또는 관리할 수 없습니다. 암호화 키 구성 요소는 외부 키 관리자를 떠나지 않습니다. 외부 키 스토어에서 KMS 키를 생성할 때는 외부 키의 ID(`XksKeyId`)를 제공합니다. 외부 키 관리자가 외부 키 ID와 연결된 키 구성 요소를 교체할 수 있지만 KMS 키와 연결된 외부 키 ID는 변경할 수 없습니다.

외부 키 스토어의 KMS 키에는 외부 키 외에도 AWS KMS 키 구성 요소가 있습니다. KMS 키로 보호되는 데이터는 먼저 AWS KMS 키 구성 요소를 AWS KMS 사용하여 암호화된 다음 외부 키를 사용하여 외부 키 관리자가 암호화합니다. 이 이중 암호화 프로세스를 통해 KMS 키로 보호되는 사이퍼텍스트가 항상 AWS KMS으로만 보호되는 사이퍼텍스트 이상으로 강력해집니다.

많은 암호화 키에는 서로 다른 유형의 식별자가 있습니다. 외부 키 스토어에 KMS 키를 생성할 때 외부 키 스토어 프록시가 외부 키를 참조하는 데 사용하는 외부 키의 ID를 제공하세요. 잘못된 식별자를 사용하면 외부 키 스토어에서 KMS 키를 생성하려는 시도가 실패합니다.

외부 키 스토어 프록시

외부 키 스토어 프록시(“XKS 프록시”)는와 외부 키 관리자 간의 모든 통신을 매개하는 고객 소유 AWS KMS 및 고객 관리형 소프트웨어 애플리케이션입니다. 또한 일반 AWS KMS 요청을 공급업체별 외부 키 관리자가 이해하는 형식으로 변환합니다. 외부 키 스토어 프록시는 외부 키 스토어에 필요합니다. 각 외부 키 스토어는 하나의 외부 키 스토어 프록시와 연결됩니다.



AWS KMS 는 외부 키를 생성, 삭제 또는 관리할 수 없습니다. 암호화 키 구성 요소는 절대 외부 키 관리자를 떠나지 않습니다. AWS KMS 와 외부 키 관리자 간의 모든 통신은 외부 키 스토어 프록시에 의해 조정됩니다.는 외부 키 스토어 프록시에 요청을 AWS KMS 보내고 외부 키 스토어 프록시로부터 응답을 받습니다. 외부 키 스토어 프록시는에서 외부 키 관리자 AWS KMS 로 요청을 전송하고 외부 키 관리자의 응답을 로 다시 전송하는 역할을 합니다. AWS KMS

외부 키 스토어의 외부 키 스토어 프록시는 사용자가 소유하고 관리하며 유지 관리 및 운영에 대한 책임은 사용자에게 있습니다. 공급업체로부터 프록시 애플리케이션을 AWS KMS 게시하거나 구매하는 오픈 소스 외부 키 스토어 프록시 API 사양을 기반으로 외부 키 스토어 프록시를 개발할 수 있습니다. 외부 키 스토어 프록시는 외부 키 관리자에 포함될 수 있습니다. 프록시 개발을 지원하기 위해는 샘플 외부 키 스토어 프록시(aws-kms-xks-proxy)와 외부 키 스토어 프록시가 사양을 준수하는지 확인하는 테스트 클라이언트(xks-kms-xksproxy-test-client) AWS KMS 도 제공합니다.

프록시 AWS KMS는에 인증하기 위해 서버 측 TLS 인증서를 사용합니다. 프록시에 인증하기 위해는 SigV4 프록시 인증 자격 증명을 사용하여 외부 키 스토어 프록시에 대한 모든 요청에 AWS KMS 서명합니다.

외부 키 스토어 프록시는 다음 암호화 제품군 중 하나 이상을 포함하는 HTTP/1.1 이상 및 TLS 1.2 이상을 지원해야 합니다.

- TLS_AES_256_GCM_SHA384(TLS 1.3)
- TLS_CHACHA20_POLY1305_SHA256(TLS 1.3)

> **Note**
>
> AWS GovCloud (US) Region 는 TLS_CHACHA20_POLY1305_SHA256을 지원하지 않습니다.

- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384(TLS 1.2)
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384(TLS 1.2)

외부 키 스토어에서 KMS 키를 생성하고 사용하려면 먼저 외부 키 스토어 프록시에 외부 키 스토어를 연결해야 합니다. 요청 시 프록시에서 외부 키 스토어를 연결 해제할 수도 있습니다. 이렇게 하면 외부 키 스토어의 모든 KMS 키를 사용할 수 없게 되어 암호화 작업에 사용할 수 없습니다.

외부 키 스토어 프록시 연결

외부 키 스토어 프록시 연결("XKS 프록시 연결")은가 외부 키 스토어 프록시와 통신하는 데 AWS KMS 사용하는 방법을 설명합니다.

외부 키 스토어를 생성할 때 프록시 연결 옵션을 지정하면 해당 옵션이 외부 키 스토어의 속성이 됩니다. 사용자 지정 키 스토어 속성을 업데이트하여 프록시 연결 옵션을 변경할 수 있지만 외부 키 스토어 프록시가 여전히 동일한 외부 키에 액세스할 수 있는지 확인해야 합니다.

AWS KMS 는 다음과 같은 연결 옵션을 지원합니다.

- 퍼블릭 엔드포인트 연결 - 인터넷을 통해 외부 키 스토어 프록시에 대한 요청을 제어하는 퍼블릭 엔드포인트로 AWS KMS 보냅니다. 이 옵션은 생성 및 유지 관리가 간단하지만 모든 설치에 대한 보안 요구 사항을 충족하지 못할 수 있습니다.
- VPC 엔드포인트 서비스 연결 - 사용자가 생성하고 유지 관리하는 Amazon Virtual Private Cloud(Amazon VPC) 엔드포인트 서비스에 요청을 AWS KMS 보냅니다. Amazon VPC 내에서 외부 키 스토어 프록시를 호스팅하거나 외부에서 외부 키 스토어 프록시를 호스팅 AWS 하고 통신에만 Amazon VPC를 사용할 수 있습니다. 또한 외부 키 저장소를 다른 AWS 계정이 소유한 Amazon VPC 엔드포인트 서비스에 연결할 수도 있습니다.

외부 키 스토어 프록시 연결 옵션에 대한 자세한 내용은 외부 키 저장소 프록시 연결 옵션 선택 섹션을 참조하세요.

외부 키 스토어 프록시 인증 자격 증명

외부 키 스토어 프록시에 인증하기 위해는 서명 V4(SigV4) 인증 자격 증명을 사용하여 외부 키 스토어 프록시에 대한 모든 요청에 AWS KMS 서명합니다. 프록시에서 인증 자격 증명을 설정하고 유지 관리한 다음 외부 저장소를 생성할 AWS KMS 때이 자격 증명을에 제공합니다.

> **Note**
>
> 가 XKS 프록시에 대한 요청에 서명하는 데 AWS KMS 사용하는 SigV4 자격 증명은의 AWS Identity and Access Management 보안 주체와 연결된 SigV4 자격 증명과 관련이 없습니다 AWS 계정. 외부 키 스토어 프록시에 IAM SigV4 자격 증명을 재사용하지 마세요.

각 프록시 인증 자격 증명은 두 부분으로 구성됩니다. 외부 키 스토어를 생성하거나 외부 키 스토어의 인증 자격 증명을 업데이트할 때 두 부분을 모두 제공해야 합니다.

- 액세스 키 ID: 비밀 액세스 키를 식별합니다. 이 ID를 일반 텍스트로 제공할 수 있습니다.
- 보안 액세스 키: 보안 인증의 보안 암호 부분입니다. 보안 인증 정보를 저장하기 전에 보안 암호 액세스 키를 AWS KMS 암호화합니다.

잘못된 값을 입력하거나 프록시에서 자격 증명을 변경하거나 프록시가 자격 증명을 교체하는 경우와 같이 언제든지 자격 증명 설정을 편집할 수 있습니다. 외부 키 스토어 프록시에 대한 인증에 대한 AWS KMS 자세한 내용은 AWS KMS 외부 키 스토어 프록시 API 사양의 인증을 참조하세요.

외부 키 스토어에서 KMS 키를 AWS 서비스 사용하는를 중단하지 않고 자격 증명을 교체할 수 있도록 하려면 외부 키 스토어 프록시가 2개 이상의 유효한 인증 자격 증명을 지원하는 것이 좋습니다 AWS KMS. 이렇게 하면 AWS KMS에 새 자격 증명을 제공하는 동안 이전 자격 증명이 계속 작동합니다.

프록시 인증 자격 증명의 수명을 추적하는 데 도움이 되도록는 Amazon CloudWatch 지표 XksProxyCredentialAge를 AWS KMS 정의합니다. 이 지표를 사용하여 자격 증명의 사용 기간이 설정한 임계값에 도달하면 알려주는 CloudWatch 경보를 생성할 수 있습니다.

프록시 API

AWS KMS 외부 키 스토어를 지원하려면 외부 키 스토어 프록시가 외부 키 스토어 프록시 APIs 사양에 설명된 대로 필요한 프록시 API를 구현해야 합니다. AWS KMS 이러한 프록시 API 요청은가 프록시로 AWS KMS 보내는 유일한 요청입니다. 이러한 요청을 직접 전송하지는 않지만 이에 대해 알면 외부 키 스토어 또는 해당 프록시에서 발생할 수 있는 문제를 해결하는 데 도움이 될 수 있습니다. 예를 들어는

외부 키 스토어에 대한 Amazon CloudWatch 지표에 이러한 API 호출의 지연 시간 및 성공률에 대한 정보를 AWS KMS 포함합니다. 자세한 내용은 외부 키 저장소 모니터링을 참조하세요.

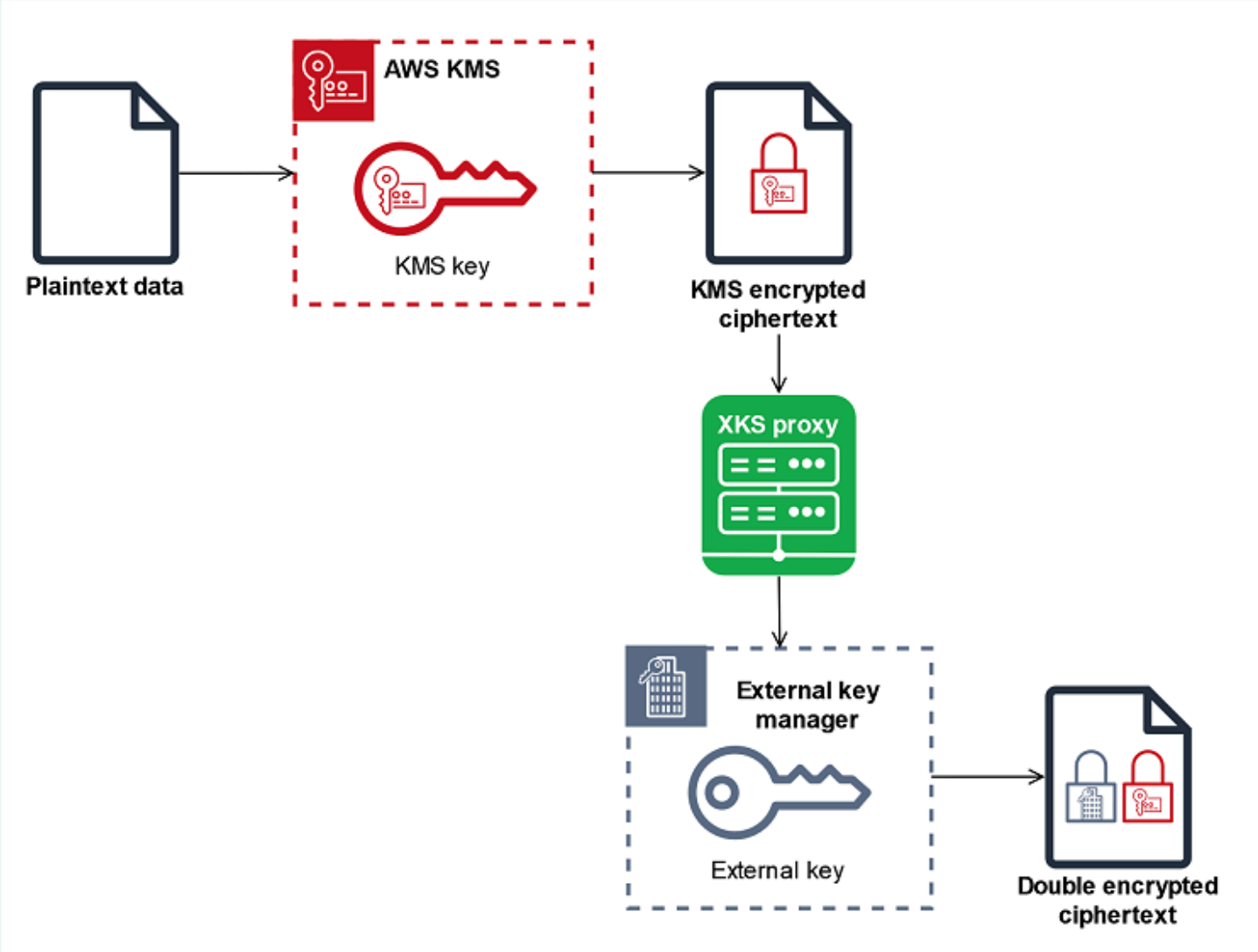
다음 표에는 각 프록시 API와 그에 대한 설명이 나와 있습니다. 또한 프록시 API에 대한 호출을 트리거하는 AWS KMS 작업과 프록시 API와 관련된 모든 AWS KMS 작업 예외도 포함됩니다.

프록시 API	설명	관련 AWS KMS 작업
Decrypt	AWS KMS 는 암호 해독할 사이퍼 텍스트와 사용할 외부 키의 ID를 전송합니다. 필수 암호화 알고리즘은 AES_GCM입니다.	Decrypt, ReEncrypt
암호화	AWS KMS 는 암호화할 데이터와 사용할 외부 키의 ID를 전송합니다. 필수 암호화 알고리즘은 AES_GCM입니다.	Encrypt, GenerateDataKey, GenerateDataKeyWithoutPlaintext, ReEncrypt
GetHealthStatus	AWS KMS 는 프록시 및 외부 키 관리자의 상태에 대한 정보를 요청합니다. 각 외부 키 관리자의 상태는 다음 중 하나일 수 있습니다. • `Active`: 정상, 트래픽 처리 가능 • `Degraded`: 비정상이지만 트래픽 처리 가능 • `Unavailable` : 비정상이고 트래픽 처리 불가능	CreateCustomKeyStore(퍼블릭 엔드포인트 연결용), ConnectCustomKeyStore(VPC 엔드포인트 서비스 연결용) 모든 외부 키 관리자 인스턴스가 `Unavailable` 인 경우 키 스토어를 생성하거나 연결하려는 시도는 `XksProxyUriUnreachableException` 과 함께 실패합니다.
GetKeyMetadata	AWS KMS 는 외부 키 스토어의 KMS 키와 연결된 외부 키에 대한 정보를 요청합니다. 응답에는 키 사양(`AES_256`), 키 사용(`[ENCRYPT, DECRYPT]`) 및 외부 키가 `ENABLED`인지 아니면 `DISABLED`인지가 포함됩니다.	CreateKey 키 사양이 `AES_256`이 아니거나, 키 사용이 `[ENCRYPT, DECRYPT]`가 아니거나, 상태가 `DISABLED`인 경우 `XksKeyInvalidConfigurationException` 과 함께 `CreateKey` 작업이 실패합니다.

이중 암호화

외부 키 스토어의 KMS 키로 암호화된 데이터는 두 번 암호화됩니다. 먼저는 KMS AWS KMS 키와 관련된 키 구성 요소로 데이터를 AWS KMS 암호화합니다. 그런 다음 AWS KMS로 암호화된 사이퍼텍스트가 외부 키를 사용하여 외부 키 관리자에 의해 암호화됩니다. 이 프로세스를 이중 암호화라고 합니다.

이중 암호화를 사용하면 외부 키 스토어의 KMS 키로 암호화된 데이터가 표준 KMS 키로 암호화된 사이퍼텍스트 이상으로 강력해집니다. 또한에서 외부 키 스토어 프록시 AWS KMS 로 전송되는 일반 텍스트를 보호합니다. 이중 암호화를 사용하면 사이퍼텍스트를 완벽하게 제어할 수 있습니다. 외부 프록시를 통해 외부 키에 대한 AWS 액세스를 영구적으로 취소하면 AWS 에 남아 있는 사이퍼텍스트가 효과적으로 암호 파쇄됩니다.



이중 암호화를 활성화하기 위해 외부 키 스토어의 각 KMS 키에는 두 개의 암호화 백업 키가 있습니다.

- KMS AWS KMS 키에 고유한 키 구성 요소입니다. 이 키 구성 요소는 AWS KMS FIPS 140-3 보안 수준 3 인증 하드웨어 보안 모듈(HSMs)에서만 생성되고 사용됩니다.
- 외부 키 관리자의 외부 키.

이중 암호화는 다음과 같은 효과가 있습니다.

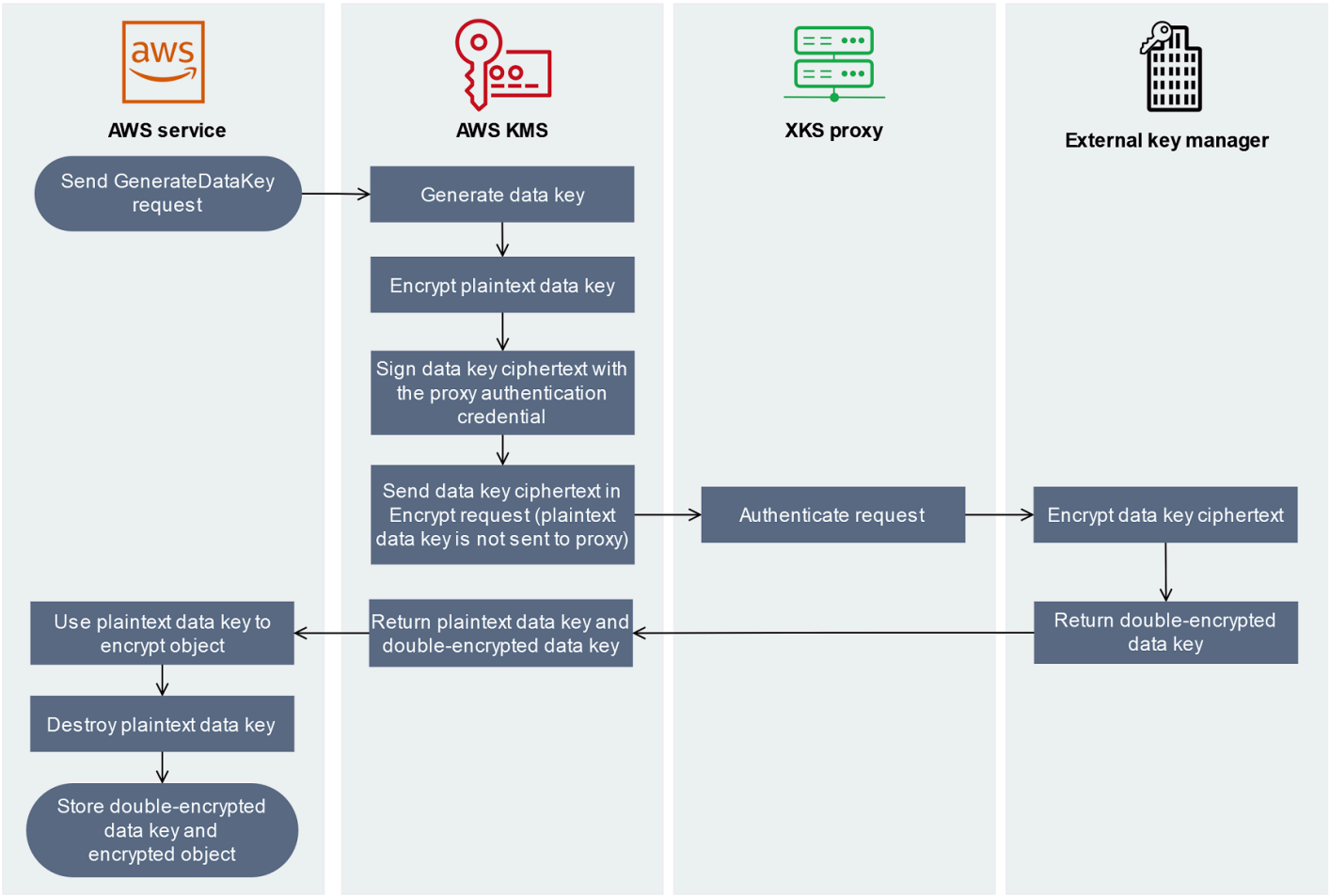
- AWS KMS 는 외부 키 스토어 프록시를 통해 외부 키에 액세스하지 않고 외부 키 스토어의 KMS 키로 암호화된 사이퍼텍스트를 해독할 수 없습니다.
- 외부 키 구성 요소가 AWS있더라도 외부 키 스토어의 KMS 키로 암호화된 사이퍼텍스트는 복호화할 수 없습니다.
- 외부 키 구성 요소가 있더라도 외부 키 스토어에서 삭제된 KMS 키를 다시 생성할 수 없습니다. 각 KMS 키에는 대칭 사이퍼텍스트에 포함된 고유한 메타데이터가 있습니다. 새 KMS 키는 동일한 외부 키 구성 요소를 사용하더라도 원래 키로 암호화된 사이퍼텍스트를 복호화할 수 없습니다.

실제 이중 암호화의 예는 외부 키 스토어 작동 방식 섹션을 참조하세요.

외부 키 스토어 작동 방식

외부 키 저장소, 외부 키 저장소 프록시 및 외부 키 관리자가 함께 작동하여 AWS 리소스를 보호합니다. 다음 절차는 KMS 키로 보호되는 고유한 데이터 키로 각 객체를 암호화하는 일반적인 AWS 서비스 의 암호화 워크플로를 보여줍니다. 이 경우 객체를 보호하기 위해 외부 키 스토어에서 KMS 키를 선택했습니다. 이 예제는가 이중 암호화를 AWS KMS 사용하여 전송 중인 데이터 키를 보호하고 외부 키 스토어의 KMS 키로 생성된 사이퍼텍스트가 항상 최소한 키 구성 요소가 있는 표준 대칭 KMS 키로 암호화된 사이퍼텍스트만큼 강력한지 확인하는 방법을 보여줍니다 AWS KMS.

와 통합 AWS 서비스 되는 각 실제에서 사용하는 암호화 방법은 AWS KMS 다양합니다. 자세한 내용은 AWS 서비스 설명서의 보안 장에 있는 '데이터 보호' 주제를 참조하세요.



1. AWS 서비스 리소스에 새 객체를 추가합니다. 객체를 암호화하기 위해는 외부 키 스토어에서 KMS 키를 AWS KMS 사용하여에 GenerateDataKey 요청을 AWS 서비스 보냅니다.
2. AWS KMS 는 256비트 대칭 데이터 키를 생성하고 외부 키 스토어 프록시를 통해 외부 키 관리자에게 일반 텍스트 데이터 키의 사본을 보낼 준비를 합니다.는 외부 키 스토어의 KMS 키와 연결된 AWS KMS 키 구성 요소로 일반 텍스트 데이터 키를 암호화하여 이중 암호화 프로세스를 AWS KMS 시작합니다.
3. AWS KMS 는 외부 키 스토어와 연결된 외부 키 스토어 프록시에 암호화 요청을 보냅니다. 요청에는 암호화할 데이터 키 사이퍼텍스트와 KMS 키와 연결된 외부 키의 ID가 포함됩니다.는 외부 키 스토어 프록시에 대한 프록시 인증 자격 증명을 사용하여 요청에 AWS KMS 서명합니다.

   데이터 키의 일반 텍스트 사본은 외부 키 스토어 프록시로 전송되지 않습니다.
4. 외부 키 스토어 프록시는 요청을 인증한 다음 암호화 요청을 외부 키 관리자에게 전달합니다.

   일부 외부 키 스토어 프록시는 선택한 보안 주체만 특정 조건에서 작업을 수행할 수 있도록 허용하는 선택적 권한 부여 정책도 구현합니다.

5. 외부 키 관리자는 지정된 외부 키를 사용하여 데이터 키 사이퍼텍스트를 암호화합니다. 외부 키 관리자는 이중 암호화된 데이터 키를 외부 키 스토어 프록시로 반환하고, 외부 키 스토어 프록시는 이를 AWS KMS로 반환합니다.
6. AWS KMS 는 일반 텍스트 데이터 키와 해당 데이터 키의 이중 암호화 사본을에 반환합니다 AWS 서비스.
7. 는 일반 텍스트 데이터 키를 AWS 서비스 사용하여 리소스 객체를 암호화하고, 일반 텍스트 데이터 키를 삭제하고, 암호화된 데이터 키를 암호화된 객체와 함께 저장합니다.

   일부 AWS 서비스 는 여러 객체에 사용하거나 리소스가 사용 중인 동안 재사용하기 위해 일반 텍스트 데이터 키를 캐시할 수 있습니다. 자세한 내용은 사용할 수 없는 KMS 키가 데이터 키에 미치는 영향을 참조하세요.

암호화된 객체를 복호화하려면가 복호화 요청 AWS KMS 에서 암호화된 데이터 키를 로 다시 전송 AWS 서비스 해야 합니다. 암호화된 데이터 키를 해독하려면 암호화된 데이터 키를 외부 키의 ID와 함께 외부 키 스토어 프록시로 다시 전송해야 AWS KMS 합니다. 어떤 이유로든 외부 키 스토어 프록시에 대한 복호화 요청이 실패하면 AWS KMS 는 암호화된 데이터 키를 복호화할 수 없으며는 암호화된 객체를 복호화할 AWS 서비스 수 없습니다.

외부 키 저장소에 대한 액세스 제어

표준 KMS 키와 함께 사용하는 키 정책, IAM 정책 및 권한 부여와 같은 모든 AWS KMS 액세스 제어 기능은 외부 키 스토어의 KMS 키에 대해 동일한 방식으로 작동합니다. IAM 정책을 사용하여 외부 키 스토어를 생성하고 관리하는 API 작업에 대한 액세스를 제어할 수 있습니다. IAM 정책 및 키 정책을 사용하여 외부 키 스토어 AWS KMS keys 의에 대한 액세스를 제어합니다. AWS 조직의 서비스 제어 정책과 VPC 엔드포인트 정책을 사용하여 외부 키 스토어의 KMS 키에 대한 액세스를 제어할 수도 있습니다.

사용자와 역할에 수행할 가능성이 있는 작업에 필요한 권한만 제공하는 것이 좋습니다.

주제

- 외부 키 스토어 관리자 권한 부여
- 외부 키 스토어의 KMS 키 사용자에게 권한 부여
- 외부 키 스토어 프록시와 통신 AWS KMS 할 수 있는 권한 부여
- 외부 키 스토어 프록시 권한 부여(선택 사항)
- mTLS 인증(사용되지 않음)

외부 키 스토어 관리자 권한 부여

외부 키 스토어를 생성하고 관리하는 보안 주체는 사용자 지정 키 스토어 작업에 대한 권한을 필요로 합니다. 다음 목록에는 외부 키 스토어 관리자에게 필요한 최소 권한이 설명되어 있습니다. 사용자 지정 키 스토어는 AWS 리소스가 아니므로 다른의 보안 주체에 대해 외부 키 스토어에 권한을 제공할 수 없습니다 AWS 계정.

- `kms:CreateCustomKeyStore`
- `kms:DescribeCustomKeyStores`
- `kms:ConnectCustomKeyStore`
- `kms:DisconnectCustomKeyStore`
- `kms:UpdateCustomKeyStore`
- `kms:DeleteCustomKeyStore`

Amazon VPC 엔드포인트 서비스 연결로 외부 키 스토어를 생성하고 VPC 엔드포인트 서비스를 다른이 소유하려면 다음 권한도 필요합니다 AWS 계정.

- `ec2:DescribeVPCEndpointServices`

외부 키 스토어를 생성하는 보안 주체는 외부 키 스토어 구성 요소를 생성하고 구성할 수 있는 권한을 필요로 합니다. 보안 주체는 자신의 계정에서만 외부 키 스토어를 생성할 수 있습니다. VPC 엔드포인트 서비스 연결이 있는 외부 키 스토어를 생성하려면 보안 주체에 다음 구성 요소를 생성할 수 있는 권한이 있어야 합니다.

- Amazon VPC
- 퍼블릭 및 프라이빗 서브넷
- Network Load Balancer 및 대상 그룹
- Amazon VPC 엔드포인트 서비스

자세한 내용은 Amazon VPC용 Identity and Access Management, VPC 엔드포인트 및 VPC 엔드포인트 서비스에 대한 ID 및 액세스 관리 및 Elastic Load Balancing API 권한을 참조하세요.

외부 키 스토어의 KMS 키 사용자에게 권한 부여

외부 키 스토어 AWS KMS keys 에서를 생성하고 관리하는 보안 주체는에서 KMS 키를 생성하고 관리하는 보안 주체와 동일한 권한이 필요합니다 AWS KMS. 외부 키 스토어의 KMS 키에 대한 기본 키 정

책은 AWS KMS의 KMS 키에 대한 기본 키 정책과 동일합니다. 태그와 별칭을 사용하여 KMS 키에 대한 액세스를 제어하는 속성 기반 액세스 제어(ABAC)는 외부 키 스토어의 KMS 키에도 유효합니다.

암호화 작업을 위해 사용자 지정 키 스토어에서 KMS 키를 사용하는 보안 주체에게는 kms:Decrypt 같은 암호화 작업을 KMS 키에서 수행할 수 있는 권한이 필요합니다. IAM 또는 키 정책에서 이러한 권한을 제공할 수 있습니다. 하지만 사용자 지정 키 스토어에서 KMS 키를 사용하기 위해 어떠한 추가 권한도 필요하지 않습니다.

외부 키 스토어의 KMS 키에만 적용되는 권한을 설정하려면 값이 EXTERNAL_KEY_STORE인 kms:KeyOrigin 정책 조건을 사용합니다. 이 조건을 사용하여 kms:CreateKey 권한 또는 KMS 키 리소스와 관련된 모든 권한을 제한할 수 있습니다. 예를 들어 다음 IAM 정책은 KMS 키가 외부 키 스토어에 있는 경우 연결된 ID가 계정의 모든 KMS 키에 대해 지정된 작업을 호출하도록 허용합니다. 권한을 외부 키 스토어의 KMS 키와의 KMS 키로 제한할 수 AWS 계정있지만 계정의 특정 외부 키 스토어로 제한할 수는 없습니다.

```
{
  "Sid": "AllowKeysInExternalKeyStores",
  "Effect": "Allow",
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey"
  ],
  "Resource": "arn:aws:kms:us-west-2:111122223333:key/*",
  "Condition": {
    "StringEquals": {
      "kms:KeyOrigin": "EXTERNAL_KEY_STORE"
    }
  }
}
```

외부 키 스토어 프록시와 통신 AWS KMS 할 수 있는 권한 부여

AWS KMS 는 사용자가 제공한 외부 키 스토어 프록시를 통해서만 외부 키 관리자와 통신합니다. AWS KMS 는 사용자가 지정한 외부 키 스토어 프록시 인증 자격 증명으로 서명 버전 4(SigV4) 프로세스를 사용하여 요청에 서명하여 프록시에 인증합니다. ??? 외부 키 스토어 프록시에 퍼블릭 엔드포인트 연결을 사용하는 경우 AWS KMS 에 추가 권한이 필요하지 않습니다.

그러나 VPC 엔드포인트 서비스 연결을 사용하는 경우 Amazon VPC 엔드포인트 서비스에 대한 인터페이스 엔드포인트를 생성할 수 있는 AWS KMS 권한을 부여해야 합니다. 이 권한은 외부 키 스토어 프록시가 VPC에 있는지 아니면 외부 키 스토어 프록시가 다른 위치에 있는지에 관계없이 필요하지만는 VPC 엔드포인트 서비스를 사용하여 통신합니다 AWS KMS.

가 인터페이스 엔드포인트를 생성 AWS KMS 하도록 허용하려면 Amazon VPC 콘솔 또는 ModifyVpcEndpointServicePermissions 작업을 사용합니다. 보안 주체 `cks.kms.<region>.amazonaws.com`에 대한 권한을 허용합니다.

Amazon VPC 엔드포인트 서비스를 외부 키 스토어(XKS) 소유 이외의 AWS 계정 다른에서 AWS 계정 소유한 경우 VPC 엔드포인트 서비스에 대한 XKS 액세스도 허용해야 합니다. 이렇게 하려면 XKS AWS 계정 ID를 Amazon VPC 엔드포인트 서비스의 보안 주체로 허용 목록에 추가하세요.

Same AWS 계정

VPC 엔드포인트 서비스를 외부 키 스토어 AWS 계정 와 동일한에서 소유한 경우 VPC 엔드포인트 서비스에 대한 보안 주체 허용 목록에 AWS KMS 를 추가해야 합니다.

다음 예제에서는 `modify-vpc-endpoint-service-permissions` AWS CLI 명령을 사용하여 AWS KMS 가 미국 서부(오레곤)(us-west-2) 리전에서 지정된 VPC 엔드포인트 서비스에 연결하도록 허용합니다. 이 명령을 사용하기 전에 Amazon VPC 서비스 ID 및를 구성에 유효한 AWS 리전 값으로 바꿉니다.

```
modify-vpc-endpoint-service-permissions
--service-id vpce-svc-12abc34567def0987
--add-allowed-principals '["cks.kms.us-west-2.amazonaws.com"]'
```

이 권한을 제거하려면 Amazon VPC 콘솔 또는 `RemoveAllowedPrincipals` 파라미터와 함께 ModifyVpcEndpointServicePermissions를 사용합니다.

Cross AWS 계정

VPC 엔드포인트 서비스를 다른이 소유한 경우 VPC 엔드포인트 서비스의 보안 주체 허용 목록에 AWS KMS 및 외부 키 스토어를 모두 추가 AWS 계정해야 합니다.

다음 예제에서는 `modify-vpc-endpoint-service-permissions` AWS CLI 명령을 사용하여 AWS KMS 및 외부 키 스토어(XKS)가 모두 미국 서부(오레곤)(us-west-2) 리전에서 지정된 VPC 엔드포인트 서비스에 연결할 수 있도록 허용합니다. 이 명령을 사용하기 전에 Amazon VPC 서비스 ID, AWS 리전및 IAM 위탁자 ARN을 구성에 유효한 값으로 바꾸세요. IAM 보안 주체는 XKS 소유자의 보안 주체로 대체되어야 합니다 AWS 계정.

이 예제에서 arn:aws:iam::*123456789012*:role/*cks_role*는 XKS 소유자 계정의 IAM 위탁자이며, XKS를 생성, 업데이트 또는 VPC 엔드포인트 서비스에 연결하는 데 사용됩니다. XKS 소유자 계정의 모든 위탁자가 VPC 엔드포인트 서비스에 액세스하도록 허용하려면 arn:aws:iam::*123456789012*:root을 지정할 수 있습니다.

```
modify-vpc-endpoint-service-permissions
--service-id vpce-svc-12abc34567def0987
--add-allowed-principals '["cks.kms.us-west-2.amazonaws.com",
 "arn:aws:iam::123456789012:role/cks_role"]'
```

이 권한을 제거하려면 Amazon VPC 콘솔 또는 RemoveAllowedPrincipals 파라미터와 함께 ModifyVpcEndpointServicePermissions를 사용합니다.

외부 키 스토어 프록시 권한 부여(선택 사항)

일부 외부 키 스토어 프록시는 외부 키 사용에 대한 권한 부여 요구 사항을 구현합니다. 외부 키 스토어 프록시는 특정 사용자가 특정 조건에서만 특정 작업을 요청할 수 있도록 하는 권한 부여 체계를 설계하고 구현하는 데 허용되지만 필수는 아닙니다. 예를 들어 사용자 A가 특정 외부 키로 암호화할 수는 있지만 복호화할 수는 없도록 프록시를 구성할 수 있습니다.

프록시 권한 부여는 모든 외부 키 스토어 프록시에가 AWS KMS 필요한 SigV4-based 프록시 인증과 독립적입니다. 또한 외부 키 스토어 또는 해당 KMS 키에 영향을 미치는 작업에 대한 액세스 권한을 부여하는 키 정책, IAM 정책 및 권한 부여와도 독립적입니다.

외부 키 스토어 프록시에 의한 권한 부여를 활성화하려면 호출자, KMS 키, AWS KMS 작업, AWS 서비스 (있는 경우)를 포함하여 각 프록시 API 요청에 메타데이터를 AWS KMS 포함합니다. 외부 키 프록시 API의 버전 1(v1)에 대한 요청 메타데이터는 다음과 같습니다.

```
"requestMetadata": {
    "awsPrincipalArn": string,
    "awsSourceVpc": string, // optional
    "awsSourceVpce": string, // optional
    "kmsKeyArn": string,
    "kmsOperation": string,
    "kmsRequestId": string,
    "kmsViaService": string // optional
}
```

예를 들어 특정 보안 주체(awsPrincipalArn)의 요청을 허용하도록 프록시를 구성할 수 있습니다. 단, 특정 보안 주체 AWS 서비스 ()가 보안 주체를 대신하여 요청을 하는 경우에만 가능합니다kmsViaService.

프록시 권한 부여에 실패하면 오류를 설명하는 메시지와 함께 관련 AWS KMS 작업이 실패합니다. 자세한 내용은 프록시 권한 부여 문제 섹션을 참조하세요.

mTLS 인증(사용되지 않음)

이 가이드의 이전 버전에서는 상호 전송 계층 보안(mTLS)을 요청을 인증하기 위한 선택적 보조 인증 메커니즘으로 언급했습니다 AWS KMS. mTLS를 사용하면 TLS 채널을 통해 통신하는 두 당사자(AWS KMS 클라이언트 및 서버로서의 XKS 프록시)가 인증서를 사용하여 서로 인증합니다.

그러나 Chrome 루트 프로그램 정책(섹션 4.2.2)을 변경하면 Chrome 루트 스토어에 포함된 공개적으로 신뢰할 수 있는 루트 CAs가 2026년 6월 15일 이후 clientAuth 확장 키 사용(EKU) 확장으로 인증서를 발급할 수 없습니다. 따라서는 더 이상 Amazon Trust Services에서 mTLS에 적합한 클라이언트 인증서를 얻을 AWS KMS 수 없습니다. 2026년 3월 16일 AWS KMS 이후에서 새 외부 키 스토어를 생성하는 데 사용되는 XKS 프록시에는 mTLS가 필요하지 않습니다. 2026년 6월 15일 이후에는 mTLS를 요구하도록 구성된 XKS 프록시가와 통신할 수 없습니다 AWS KMS. 고객은 SigV4 인증을 사용하여 요청이 시작되었는지 확인해야 합니다 AWS KMS. 자세한 내용은 외부 키 스토어 프록시 인증 자격 증명을 참조하세요.

외부 키 저장소 프록시 연결 옵션 선택

외부 키 스토어를 생성하기 전에가 외부 키 스토어 구성 요소와 AWS KMS 통신하는 방식을 결정하는 연결 옵션을 선택합니다. 선택한 연결 옵션에 따라 나머지 계획 프로세스가 결정됩니다.

외부 키 스토어를 생성하는 경우가 외부 키 스토어 프록시와 AWS KMS 통신하는 방식을 결정해야 합니다. 이 선택에 따라 필요한 구성 요소와 구성 방법이 결정됩니다.는 다음 연결 옵션을 AWS KMS 지원합니다.

- 퍼블릭 엔드포인트 연결
- VPC 엔드포인트 서비스 연결

성능과 보안 목표에 맞는 옵션을 선택합니다.

시작하기 전에 외부 키 스토어가 필요한지 확인합니다. 대부분의 고객은 키 구성 요소가 지원하는 KMS AWS KMS 키를 사용할 수 있습니다.

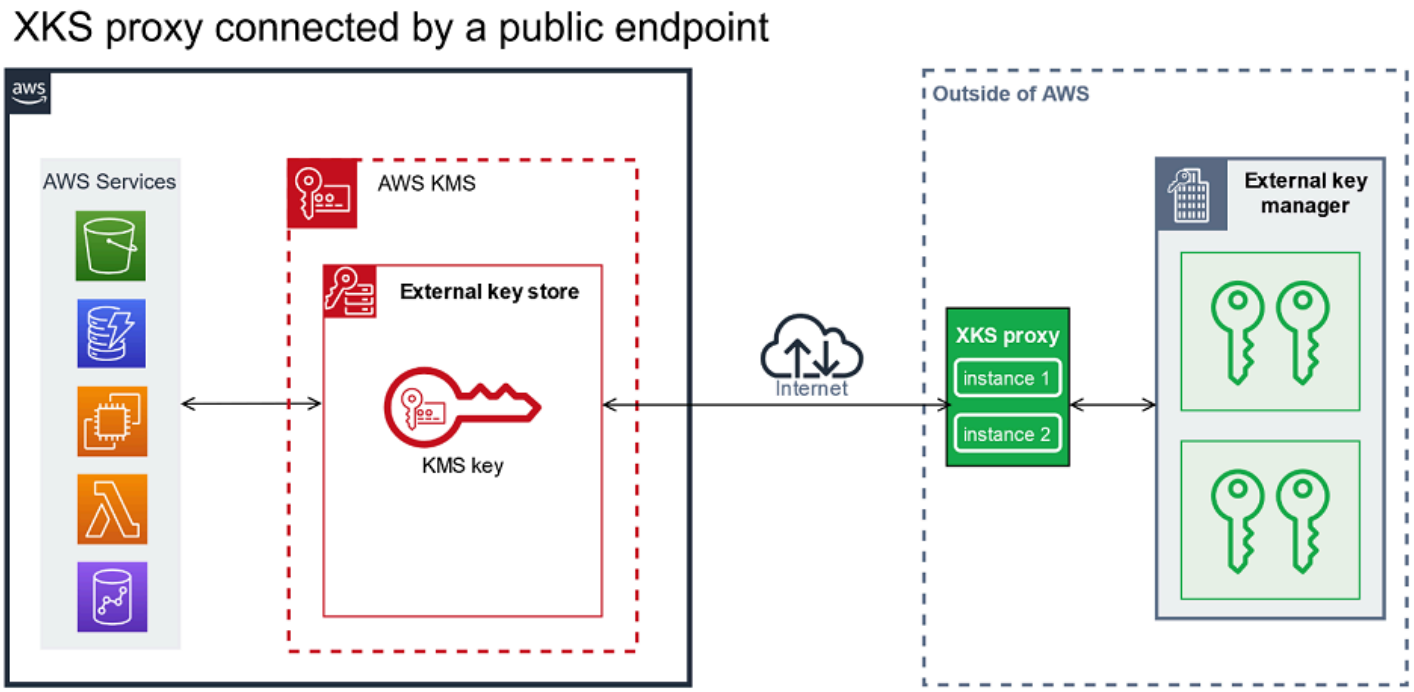
고려 사항

- 외부 키 스토어 프록시가 외부 키 관리자에 내장된 경우 연결이 미리 결정될 수 있습니다. 지침은 외부 키 관리자 또는 외부 키 스토어 프록시의 설명서를 참조하세요.
- 작동 중인 외부 키 스토어에서도 외부 키 스토어 프록시 연결 옵션을 변경할 수 있습니다. 그러나 중단을 최소화하고 오류를 방지하고 데이터를 암호화하는 암호화 키에 지속적으로 액세스할 수 있도록 프로세스를 신중하게 계획하고 실행해야 합니다.

퍼블릭 엔드포인트 연결

AWS KMS 는 퍼블릭 엔드포인트를 사용하여 인터넷을 통해 외부 키 스토어 프록시(XKS 프록시)에 연결합니다.

이 연결 옵션은 설정 및 유지 관리가 더 쉽고 일부 키 관리 모델과 잘 맞습니다. 그러나 일부 조직의 보안 요구 사항을 충족하지 못할 수 있습니다.



요구 사항

퍼블릭 엔드포인트 연결을 선택하는 경우 다음이 필요합니다.

- 외부 키 스토어 프록시는 공개적으로 라우팅 가능한 엔드포인트에서 연결할 수 있어야 합니다.
- 서로 다른 프록시 URI 경로 값을 사용하는 경우 여러 외부 키 스토어에 동일한 퍼블릭 엔드포인트를 사용할 수 있습니다.

- 키 스토어가 다른에 AWS 리전있더라도 퍼블릭 엔드포인트 연결이 있는 외부 키 스토어와 동일한 에 VPC 엔드포인트 서비스 연결이 있는 외부 키 스토어에 동일한 엔드포인트를 사용할 수 없습니다 AWS 계정.
- 외부 키 스토어에 대해 지원되는 공인 인증 기관에서 발급한 TLS 인증서를 받아야 합니다. 목록은 신뢰할 수 있는 인증 기관을 참조하세요.

  TLS 인증서의 주체 일반 이름(CN)은 외부 키 스토어 프록시에 대한 프록시 URI 엔드 포인트의 도메인 이름과 일치해야 합니다. 예를 들어 퍼블릭 엔드포인트가 `https://myproxy.xks.example.com`인 경우 TLS, TLS 인증서의 CN은 `myproxy.xks.example.com` 또는 `*.xks.example.com`이어야 합니다.
- AWS KMS 와 외부 키 스토어 프록시 사이의 방화벽이 IPv4를 통해 포트 443에서 프록시 AWS KMS 통신의 포트 443과 주고받는 트래픽을 허용하는지 확인합니다. 이 값은 구성할 수 없습니다.

외부 키 스토어의 모든 요구 사항은 사전 조건 수집을 참조하세요.

VPC 엔드포인트 서비스 연결

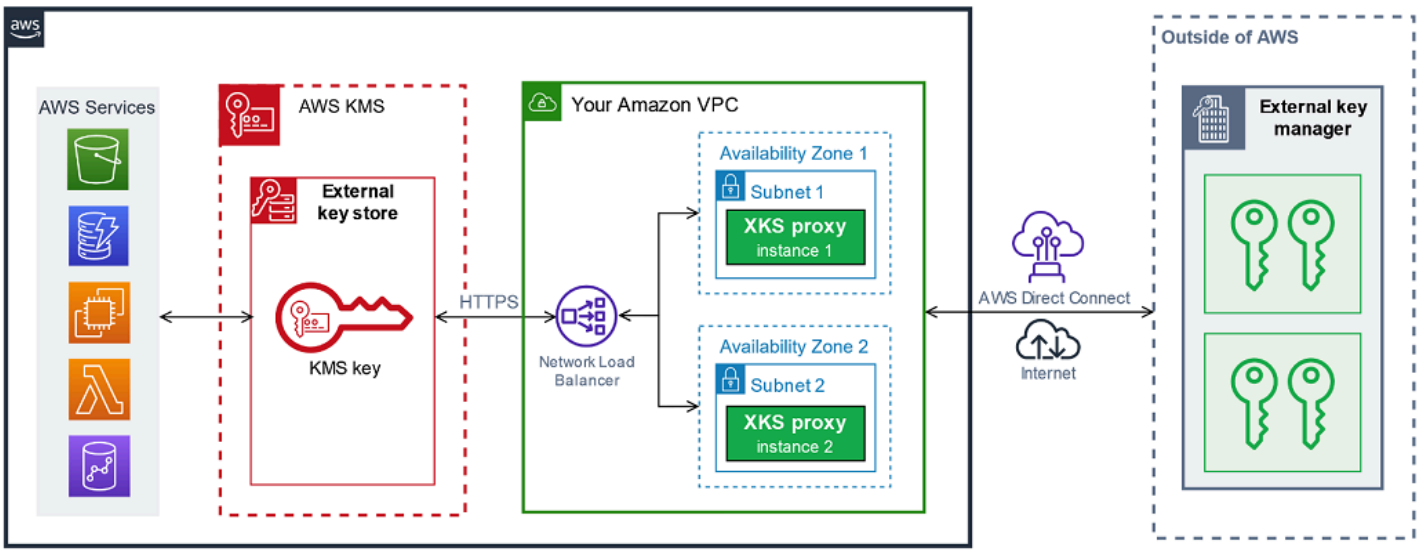
AWS KMS 는 사용자가 생성하고 구성하는 Amazon VPC 엔드포인트 서비스에 대한 인터페이스 엔드포인트를 생성하여 외부 키 스토어 프록시(XKS 프록시)에 연결합니다. VPC 엔드포인트 서비스를 생성하고 VPC를 외부 키 관리자에 연결하는 것은 사용자의 책임입니다.

엔드포인트 서비스는 AWS Direct Connect를 포함하여 지원되는 네트워크와 Amazon VPC 간 옵션을 사용할 수 있습니다.

이 연결 옵션은 설정 및 유지 관리가 더 복잡합니다. 그러나가 퍼블릭 인터넷 AWS KMS 을 사용하지 않고 Amazon VPC 및 외부 키 스토어 프록시에 비공개로 연결할 수 AWS PrivateLink있는를 사용합니다.

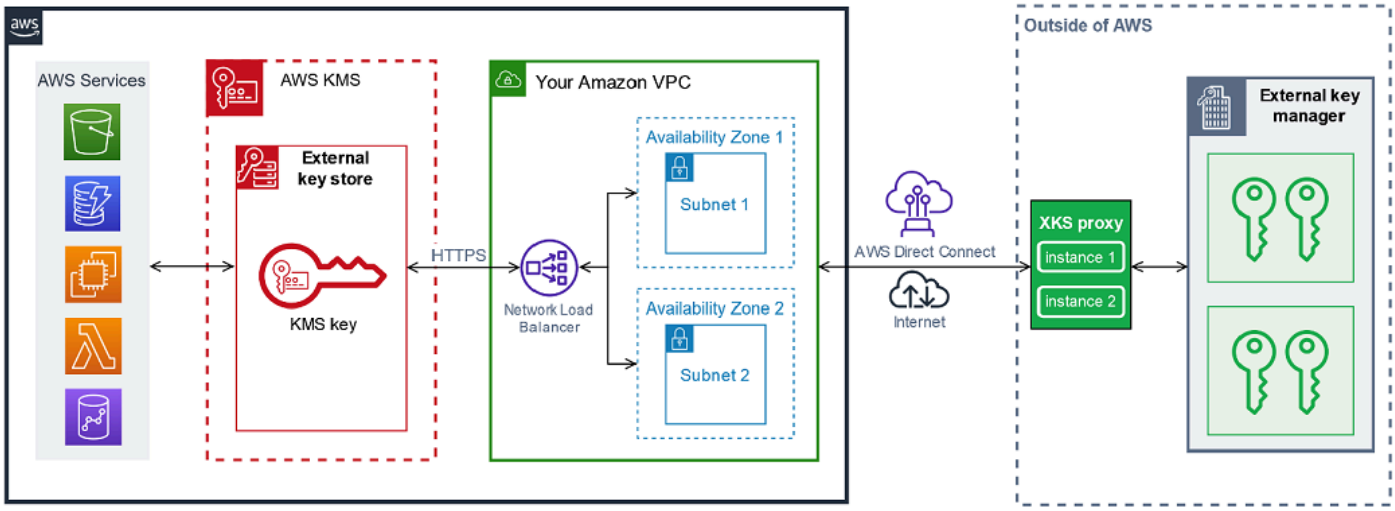
Amazon VPC에서 외부 키 스토어 프록시를 찾을 수 있습니다.

XKS proxy hosted in Amazon VPC



또는 외부에서 외부 키 스토어 프록시를 찾고 보안 통신을 위해서만 Amazon VPC 엔드포인트 서비스를 AWS 클라우드 사용할 수 있습니다 AWS KMS.

XKS proxy connected via Amazon VPC endpoint service



외부 키 스토어를 다른이 소유한 Amazon VPC 엔드포인트 서비스에 연결할 수도 있습니다 AWS 계정. 둘 다 AWS KMS 와 VPC 엔드포인트 서비스 간의 통신을 허용하는 데 필요한 권한이 AWS 계정 필요합니다.

자세히 알아보기:

- 사전 요구 사항 취합을 포함하여 외부 키 스토어 생성 프로세스를 검토합니다. 이를 통해 외부 키 스토어를 생성할 때 필요한 모든 구성 요소가 있는지 확인할 수 있습니다.

- 외부 키 스토어 관리자 및 사용자에게 필요한 권한을 포함하여 외부 키 스토어에 대한 액세스를 제어하는 방법을 알아봅니다.
- 가 외부 키 스토어에 대해 AWS KMS 기록하는 Amazon CloudWatch 지표 및 차원에 대해 알아봅니다. 성능 및 운영 문제의 초기 징후를 감지할 수 있도록 외부 키 스토어를 모니터링하는 경보를 생성하는 것이 좋습니다.

VPC 엔드포인트 서비스 연결 구성

이 섹션의 지침에 따라 VPC 엔드포인트 서비스 연결을 사용하는 외부 키 스토어에 필요한 AWS 리소스 및 관련 구성 요소를 생성하고 구성합니다. 이 연결 옵션에 대해 나열된 리소스는 모든 외부 키 스토어에 필요한 리소스를 보충하는 것입니다. 필요한 리소스를 생성하고 구성한 후 외부 키 스토어를 생성할 수 있습니다.

Amazon VPC에서 외부 키 스토어 프록시를 찾거나 외부에서 프록시를 찾 AWS 은 다음 VPC 엔드포인트 서비스를 사용하여 통신할 수 있습니다.

시작하기 전에 외부 키 스토어가 필요한지 확인합니다. 대부분의 고객은 키 구성 요소가 지원하는 KMS AWS KMS 키를 사용할 수 있습니다.

> **Note**
>
> VPC 엔드포인트 서비스 연결에 필요한 일부 요소가 외부 키 관리자에 포함될 수 있습니다. 또한 소프트웨어에 추가 구성 요구 사항이 있을 수 있습니다. 이 단원에서 AWS 리소스를 생성하고 구성하기 전에 프록시 및 키 관리자 설명서를 참조하세요.

주제

- VPC 엔드포인트 서비스 연결 요구 사항
- 1단계: Amazon VPC와 서브넷 생성
- 2단계: 대상 그룹 생성
- 3단계: 새 Network Load Balancer 생성
- 4단계: VPC 엔드포인트 서비스 생성
- 5단계: 프라이빗 DNS 이름 도메인 확인
- 6단계: VPC 엔드포인트 서비스에 AWS KMS 연결하도록 권한 부여

VPC 엔드포인트 서비스 연결 요구 사항

외부 키 스토어에 대해 VPC 엔드포인트 서비스 연결을 선택하는 경우 다음 리소스가 필요합니다.

- 외부 키 관리자에 연결된 Amazon VPC. 두 개의 서로 다른 가용성 영역에 두 개 이상의 프라이빗 서브넷이 있어야 합니다.

  외부 키 스토어와 함께 사용하기 위한 요구 사항을 충족하는 경우 외부 키 스토어에 기존 Amazon VPC를 사용할 수 있습니다. 여러 외부 키 스토어가 Amazon VPC를 공유할 수 있지만 각 외부 키 스토어에는 자체 VPC 엔드포인트 서비스와 프라이빗 DNS 이름이 있어야 합니다.
- Network Load Balancer 및 대상 그룹과 함께 AWS PrivateLink로 구동되는 Amazon VPC 엔드포인트 서비스.

  엔드포인트 서비스에서 수락을 요구할 수 없습니다. 또한 AWS KMS 를 허용된 보안 주체로 추가해야 합니다. 이렇게 하면 AWS KMS 가 인터페이스 엔드포인트를 생성하여 외부 키 스토어 프록시와 통신할 수 있습니다.
- AWS 리전에서 고유한 VPC 엔드포인트 서비스의 프라이빗 DNS 이름.

  프라이빗 DNS 이름은 상위 수준 퍼블릭 도메인의 하위 도메인이어야 합니다. 예를 들어 프라이빗 DNS 이름이 `myproxy-private.xks.example.com`인 경우 `xks.example.com` 또는 `example.com`과 같은 퍼블릭 도메인의 하위 도메인이어야 합니다.

  프라이빗 DNS 이름에 대한 DNS 도메인의 소유권을 확인해야 합니다.
- 외부 키 스토어 프록시에 대해 지원되는 공인 인증 기관에서 발급한 TLS 인증서.

  TLS 인증서의 주체 일반 이름(CN)은 프라이빗 DNS 이름과 일치해야 합니다. 예를 들어 프라이빗 DNS 이름이 `myproxy-private.xks.example.com`인 경우 TLS 인증서의 CN은 `myproxy-private.xks.example.com` 또는 `*.xks.example.com`이어야 합니다.
- 네트워크 지연 시간을 최소화하려면 외부 키 관리자와 가장 가까운 지원되는 AWS 리전에서 AWS 구성 요소를 생성합니다. 가능하면 네트워크 왕복 시간(RTT)이 35밀리초 이하인 리전을 선택합니다.

외부 키 스토어의 모든 요구 사항은 사전 조건 수집을 참조하세요.

1단계: Amazon VPC와 서브넷 생성

VPC 엔드포인트 서비스 연결에는 프라이빗 서브넷이 2개 이상 있는 외부 키 관리자에 연결된 Amazon VPC가 필요합니다. Amazon VPC를 생성하거나 외부 키 스토어의 요구 사항을 충족하는 기존

Amazon VPC를 사용할 수 있습니다. 새 Amazon VPC를 생성하는 방법에 대한 도움말은 Amazon Virtual Private Cloud 사용 설명서의 VPC 생성을 참조하세요.

Amazon VPC의 요구 사항

외부 키 저장소와 함께 사용하려면 Amazon VPC 엔드포인트 서비스는 다음과 같은 속성을 갖추어야 합니다.

- 외부 키 저장소와 동일한 지원 리전에 있어야 합니다.
- 각각 다른 가용 영역에 있는 두 개 이상의 프라이빗 서브넷이 필요합니다.
- Amazon VPC의 프라이빗 IP 주소 범위는 외부 키 관리자를 호스팅하는 데이터 센터의 프라이빗 IP 주소 범위와 겹치지 않아야 합니다.
- 모든 구성 요소는 IPv4를 사용해야 합니다.

Amazon VPC를 외부 키 스토어 프록시에 연결하기 위한 많은 옵션이 있습니다. 성능과 보안 요구 사항에 맞는 옵션을 선택합니다. 목록을 보려면 다른 네트워크에 VPC 연결 및 네트워크와 Amazon VPC 간 연결 옵션을 참조하세요. 자세한 내용은 Direct Connect와 AWS Site-to-Site VPN 사용 설명서를 참조하세요.

외부 키 스토어용 Amazon VPC 생성

다음 지침에 따라 외부 키 스토어용 Amazon VPC를 생성하세요. Amazon VPC는 VPC 엔드포인트 서비스 연결 옵션을 선택한 경우에만 필요합니다. 외부 키 스토어의 요구 사항을 충족하는 기존 Amazon VPC를 사용할 수 있습니다.

다음 필수 값을 사용하여 VPC, 서브넷 및 기타 VPC 리소스 생성 주제의 지침을 따르세요. 다른 필드의 경우 기본값을 그대로 사용하고 요청에 따라 이름을 제공합니다.

Field	값
IPv4 CIDR block	VPC의 IP 주소를 입력합니다. Amazon VPC의 프라이빗 IP 주소 범위는 외부 키 관리자를 호스팅하는 데이터 센터의 프라이빗 IP 주소 범위와 겹치지 않아야 합니다.
가용 영역 수	2 이상
퍼블릭 서브넷 수	필요 없음(0)

Field	값
프라이빗 서브넷 수	각 AZ에 대해 하나씩
NAT 게이트웨이	필요 없음
VPC 엔드포인트	필요 없음
Enable DNS hostnames	예
DNS 확인 활성화	예

VPC 통신을 테스트해야 합니다. 예를 들어 외부 키 스토어 프록시가 Amazon VPC에 없는 경우 Amazon VPC에서 Amazon EC2 인스턴스를 생성하고 Amazon VPC가 외부 키 스토어 프록시와 통신할 수 있는지 확인합니다.

외부 키 관리자에 VPC 연결

Amazon VPC가 지원하는 네트워크 연결 옵션을 사용하여 외부 키 관리자를 호스팅하는 데이터 센터에 VPC를 연결합니다. VPC의 Amazon EC2 인스턴스 또는 외부 키 스토어 프록시(VPC에 있는 경우)가 데이터 센터 및 외부 키 관리자와 통신할 수 있는지 확인합니다.

2단계: 대상 그룹 생성

필수 VPC 엔드포인트 서비스를 생성하기 전에 필수 구성 요소, Network Load Balancer(NLB) 및 대상 그룹을 생성합니다. Network Load Balancer(NLB)는 요청을 서비스할 수 있는 여러 정상 대상에 요청을 분산합니다. 이 단계에서는 외부 키 스토어 프록시에 사용할 호스트가 두 개 이상 있는 대상 그룹을 만들고 대상 그룹에 IP 주소를 등록합니다.

다음 필수 값을 사용하여 대상 그룹 구성 주제의 지침을 따르세요. 다른 필드의 경우 기본값을 그대로 사용하고 요청에 따라 이름을 제공합니다.

Field	값
대상 유형	IP 주소
프로토콜	TCP
포트	443
IP 주소 유형	IPv4
VPC	외부 키 스토어에 대한 VPC 엔드포인트 서비스를 생성할 VPC를 선택합니다.
상태 확인 프로토콜 및 경로	상태 확인 프로토콜 및 경로는 외부 키 스토어 프록시 구성에 따라 달라집니다. 외부 키 관리자 또는 외부 키 스토어 프록시의 설명서를 참조하세요. 대상 그룹의 상태 확인 구성에 대한 일반 정보는 Elastic Load Balancing - Network Load Balancer 사용 설명서의 대상 그룹에 대한 상태 확인을 참조하세요.
Network	다른 프라이빗 IP 주소
IPv4 주소	외부 키 스토어 프록시의 프라이빗 주소
포트	443

3단계: 새 Network Load Balancer 생성

Network Load Balancer는 AWS KMS 에서 외부 키 스토어 프록시로의 요청을 포함하여 네트워크 트래픽을 구성된 대상으로 분산합니다.

로드 밸런서 및 리스너 구성 주제의 지침에 따라 리스너를 구성 및 추가하고 다음 필수 값을 사용하여 로드 밸런서를 생성합니다. 다른 필드의 경우 기본값을 그대로 사용하고 요청에 따라 이름을 제공합니다.

Field	값
스킴	Internal(내부)
IP 주소 유형	IPv4

Field	값
네트워크 매핑	외부 키 스토어에 대한 VPC 엔드포인트 서비스를 생성할 VPC를 선택합니다.
매핑	VPC 서브넷용으로 구성한 가용 영역(최소 2개)을 모두 선택합니다. 서브넷 이름과 프라이빗 IP 주소를 확인합니다.
프로토콜	TCP
포트	443
Default action: Forward to(기본 작업: 다음으로 전달)	Network Load Balancer의 대상 그룹을 선택합니다.

4단계: VPC 엔드포인트 서비스 생성

일반적으로 서비스에 대한 엔드포인트를 생성합니다. 그러나 VPC 엔드포인트 서비스를 생성하면 공급자가 되고 서비스에 대한 엔드포인트가 AWS KMS 생성됩니다. 외부 키 스토어의 경우 이전 단계에서 생성한 Network Load Balancer를 사용하여 VPC 엔드포인트 서비스를 생성합니다. VPC 엔드포인트 서비스는 외부 키 스토어 AWS 계정 와 동일한 또는 다른에 있을 수 있습니다 AWS 계정.

여러 외부 키 스토어가 Amazon VPC를 공유할 수 있지만 각 외부 키 스토어에는 자체 VPC 엔드포인트 서비스와 프라이빗 DNS 이름이 있어야 합니다.

엔드포인트 서비스 생성 주제의 지침에 따라 다음과 같은 필수 값으로 VPC 엔드포인트 서비스를 생성합니다. 다른 필드의 경우 기본값을 그대로 사용하고 요청에 따라 이름을 제공합니다.

Field	값
로드 밸런서 유형	Network
사용 가능한 로드 밸런서	이전 단계에서 생성한 Network Load Balancer를 선택합니다. 새 로드 밸런서가 목록에 나타나지 않으면 해당 상태가 활성인지 확인합니다. 로드 밸런서 상태가 프로비저닝에서 활성으로 변경되는 데 몇 분 정도 걸릴 수 있습니다.

Field	값
수락 필요	False. 확인란 선택을 취소합니다. acceptance.cannot connect to the VPC endpoint service without 수동 수락이 필요하지 않습니다. AWS KMS 수락이 필요한 경우 외부 키 스토어를 생성하려는 시도는 `XksProxyInvalidConfigurationException` 예외와 함께 실패합니다.
프라이빗 DNS 이름 활성화	프라이빗 DNS 이름을 서비스와 연결
프라이빗 DNS 이름	AWS 리전에서 고유한 프라이빗 DNS 이름을 입력합니다. 프라이빗 DNS 이름은 상위 수준 퍼블릭 도메인의 하위 도메인이어야 합니다. 예를 들어 프라이빗 DNS 이름이 `myproxy-private.xks.example.com` 인 경우 `xks.example.com` 또는 `example.com` 과 같은 퍼블릭 도메인의 하위 도메인이어야 합니다. 이 프라이빗 DNS 이름은 외부 키 스토어 프록시에 구성된 TLS 인증서의 주체 일반 이름(CN)과 일치해야 합니다. 예를 들어 프라이빗 DNS 이름이 `myproxy-private.xks.example.com` 인 경우 TLS 인증서의 CN은 `myproxy-private.xks.example.com` 또는 `*.xks.example.com` 이어야 합니다. 인증서와 프라이빗 DNS 이름이 일치하지 않으면 외부 키 스토어 프록시에 외부 키 스토어를 연결하려는 시도가 연결 오류 코드(`XKS_PROXY_INVALID_TLS_CONFIGURATION`)와 함께 실패합니다. 자세한 내용은 일반 구성 오류을 참조하세요.
지원되는 IP 주소 유형	IPv4

5단계: 프라이빗 DNS 이름 도메인 확인

VPC 엔드포인트 서비스를 생성할 때 해당 도메인 확인 상태는 `pendingVerification`입니다. VPC 엔드포인트 서비스를 사용하여 외부 키 스토어를 생성하기 전에 이 상태는 `verified`여야 합니다. 프라이빗 DNS 이름과 연결된 도메인을 소유하고 있는지 확인하려면 퍼블릭 DNS 서버에 TXT 레코드를 생성해야 합니다.

예를 들어 VPC 엔드포인트 서비스의 프라이빗 DNS 이름이 인 경우 `myproxy-private.xks.example.com`먼저에서 TXT 레코드에 대해 `xks.example.com` 또는 `example.com`와 같은 퍼블릭 도메인에 TXT 레코드를 생성한 `xks.example.com` 다음에서 public. AWS PrivateLink looks를 생성해야 합니다`example.com`.

> **Tip**
>
> TXT 레코드를 추가한 후 Domain verification status(도메인 확인 상태) 값이 `pendingVerification`에서 `verify`로 변경되는 데 몇 분 정도 걸릴 수 있습니다.

시작하려면 다음 방법 중 하나를 사용하여 도메인의 확인 상태를 찾습니다. 유효한 값은 `verified`, `pendingVerification`, `failed`입니다.

- Amazon VPC 콘솔에서 Endpoint services(엔드포인트 서비스)를 선택하고 엔드포인트 서비스를 선택합니다. 세부 정보 창에서 Domain verification status(도메인 확인 상태)를 봅니다.
- DescribeVpcEndpointServiceConfigurations 작업을 사용합니다. `State` 값은 `ServiceConfigurations.PrivateDnsNameConfiguration.State` 필드에 있습니다.

확인 상태가 `verified`가 아니면 도메인 소유권 확인 주제의 지침에 따라 도메인의 DNS 서버에 TXT 레코드를 추가하고 TXT 레코드가 게시되었는지 확인합니다. 그런 다음 인증 상태를 다시 확인합니다.

프라이빗 DNS 도메인 이름에 대한 A 레코드를 생성할 필요는 없습니다. 가 VPC 엔드포인트 서비스에 대한 인터페이스 엔드포인트를 AWS KMS 생성하면 AWS PrivateLink 는 AWS KMS VPC의 프라이빗 도메인 이름에 필요한 A 레코드가 있는 호스팅 영역을 자동으로 생성합니다. VPC 엔드포인트 서비스 연결이 있는 외부 키 스토어의 경우 외부 키 스토어 프록시에 외부 키 스토어를 연결할 때 이러한 작업이 수행됩니다.

6단계: VPC 엔드포인트 서비스에 AWS KMS 연결하도록 권한 부여

Amazon VPC 엔드포인트 서비스 권한을 관리하기 위한 절차는 다음을 참조하세요. 각 단계는 외부 키 저장소, VPC 엔드포인트 서비스 및 AWS 계정사이의 연결 상태 및 구성에 따라 달라집니다.

Same AWS 계정

VPC 엔드포인트 서비스를 외부 키 스토어 AWS 계정 와 동일한에서 소유한 경우 VPC 엔드포인트 서비스에 대한 보안 주체 허용 목록에 AWS KMS 를 추가해야 합니다. 이렇게 하면 AWS KMS 가 VPC 엔드포인트 서비스에 대한 인터페이스 엔드포인트를 생성할 수 있

습니다. AWS KMS 가 허용된 보안 주체가 아닌 경우 외부 키 스토어를 생성하려는 시도는 `XksProxyVpcEndpointServiceNotFoundException` 예외와 함께 실패합니다.

AWS PrivateLink 가이드의 Manage permissions(권한 관리) 주제에 나와 있는 지침을 따르세요. 다음 필수 값을 사용합니다.

Field	값
AWS KMS ARN	`cks.kms.<region>.amazonaws.com` 예: `cks.kms.us-east-1.amazonaws.com`

Cross AWS 계정

VPC 엔드포인트 서비스를 다른이 소유한 경우 보안 주체 허용 목록에 AWS KMS 및 계정을 모두 추가 AWS 계정 해야 합니다. 이렇게 하면 AWS KMS 및 외부 키 스토어가 VPC 엔드포인트 서비스에 대한 인터페이스 엔드포인트를 생성할 수 있습니다. AWS KMS 가 허용된 보안 주체가 아닌 경우 외부 키 스토어를 생성하려는 시도는 `XksProxyVpcEndpointServiceNotFoundException` 예외와 함께 실패합니다. 외부 키 스토어가 있는 AWS 계정 ARN을 제공해야 합니다.

AWS PrivateLink 가이드의 Manage permissions(권한 관리) 주제에 나와 있는 지침을 따르세요. 다음 필수 값을 사용합니다.

Field	값
AWS KMS ARN	`cks.kms.<region>.amazonaws.com` 예: `cks.kms.us-east-1.amazonaws.com`
AWS 계정 ARN	`arn:aws:iam:: 111122223333 :role/role_name` 예: `arn:aws:iam:: 123456789012 :role/cks_role`

다음: 외부 키 저장소 생성

외부 키 저장소 생성

각 AWS 계정 및 리전에서 하나 이상의 외부 키 스토어를 생성할 수 있습니다. 각 외부 키 스토어는 외부의 외부 키 관리자 AWS및와 외부 키 관리자 간의 통신을 매개하는 외부 키 스토어 프록시(XKS 프록

시) AWS KMS 와 연결되어야 합니다. 자세한 내용은 외부 키 저장소 프록시 연결 옵션 선택을 참조하세요. 시작하기 전에 외부 키 스토어가 필요한지 확인합니다. 대부분의 고객은 키 구성 요소가 지원하는 KMS AWS KMS 키를 사용할 수 있습니다.

> **Tip**
>
> 일부 외부 키 관리자는 외부 키 스토어를 생성하는 더 간단한 방법을 제공합니다. 자세한 내용은 외부 키 관리자 설명서를 참조하세요.

외부 키 스토어를 생성하기 전에 사전 조건을 수집해야 합니다. 생성 프로세스 중 외부 키 스토어의 속성을 지정합니다. 가장 중요한 것은의 외부 키 스토어가 퍼블릭 엔드포인트를 AWS KMS 사용하는지 아니면 VPC 엔드포인트 서비스를 사용하여 외부 키 스토어 프록시에 연결하는지를 나타냅니다. 또한 프록시의 URI 엔드포인트와가 API 요청을 프록시로 AWS KMS 보내는 프록시 엔드포인트 내의 경로를 포함하여 연결 세부 정보를 지정합니다.

고려 사항

- KMS는 IPv6를 통해 외부 키 저장소와 통신할 수 없습니다.
- 퍼블릭 엔드포인트 연결을 사용하는 경우가 HTTPS 연결을 사용하여 인터넷을 통해 프록시와 통신 AWS KMS 할 수 있는지 확인합니다. 여기에는 외부 키 스토어 프록시에서 TLS를 구성하고 AWS KMS 와 프록시 사이의 방화벽이 프록시의 포트 443에서 송수신되는 IPv4 트래픽을 허용하는지 확인하는 작업이 포함됩니다. 퍼블릭 엔드포인트 연결이 있는 외부 키 스토어를 생성하는 동안는 외부 키 스토어 프록시에 상태 요청을 전송하여 연결을 AWS KMS 테스트합니다. 이 테스트는 엔드포인트에 연결할 수 있고 외부 키 스토어 프록시가 외부 키 스토어 프록시 인증 자격 증명으로 서명된 요청을 수락하는지 확인합니다. 이 테스트 요청이 실패하면 외부 키 스토어를 생성하는 작업이 실패합니다.
- VPC 엔드포인트 서비스 연결을 사용하는 경우 Network Load Balancer, 프라이빗 DNS 이름 및 VPC 엔드포인트 서비스가 올바르게 구성되고 작동하는지 확인합니다. 외부 키 스토어 프록시가 VPC에 없는 경우 VPC 엔드포인트 서비스가 외부 키 스토어 프록시와 통신할 수 있는지 확인해야 합니다. (외부 키 스토어를 외부 키 스토어 프록시에 연결할 때 VPC 엔드포인트 서비스 연결을AWS KMS 테스트합니다.)
- AWS KMS 는 특히 외부 키 스토어에 대한 Amazon CloudWatch 지표 및 차원을 기록합니다. 이러한 지표 중 일부를 기반으로 한 모니터링 그래프는 각 외부 키 스토어의 AWS KMS 콘솔에 표시됩니다. 이러한 지표를 사용하여 외부 키 저장소를 모니터링하는 경보를 생성하는 것을 적극 권장합니다. 이러한 경보는 성능 및 작동 문제가 발생하기 전에 조기 징후를 알립니다. 지침은 외부 키 저장소 모니터링 섹션을 참조하세요.

- 외부 키 스토어에는 리소스 할당량이 적용됩니다. 외부 키 스토어에서 KMS 키를 사용할 경우 요청 할당량이 적용됩니다. 외부 키 스토어 구현을 설계하기 전에 이러한 할당량을 검토합니다.

> **Note**
>
> 작동을 방해할 수 있는 순환 종속성이 있는지 구성을 검토하세요.
> 예를 들어 AWS 리소스를 사용하여 외부 키 스토어 프록시를 생성하는 경우 프록시를 운영할 때 해당 프록시를 통해 액세스하는 외부 키 스토어에서 KMS 키를 사용할 필요가 없는지 확인합니다.

모든 새 외부 키 스토어는 연결 해제된 상태로 생성됩니다. 외부 키 스토어에서 KMS 키를 생성하려면 먼저 외부 키 스토어 프록시에 연결해야 합니다. 외부 키 스토어의 속성을 변경하려면 외부 키 스토어 설정을 편집합니다.

주제

- 사전 조건 수집
- 새로운 외부 키 저장소 생성

사전 조건 수집

외부 키 스토어를 생성하기 전에 외부 키 스토어를 지원하는 데 사용할 외부 키 관리자와 요청을 외부 키 관리자가 이해할 수 있는 형식으로 변환하는 외부 키 스토어 프록시 AWS KMS 를 포함하여 필요한 구성 요소를 조합해야 합니다.

모든 외부 키 스토어에는 다음 구성 요소가 필요합니다. 이러한 구성 요소 외에도 선택한 외부 키 스토어 프록시 연결 옵션을 지원하는 구성 요소를 제공해야 합니다.

> **Tip**
>
> 외부 키 관리자에 이러한 구성 요소 중 일부가 포함되어 있거나 사용자에게 맞게 구성될 수 있습니다. 자세한 내용은 외부 키 관리자 설명서를 참조하세요.
> AWS KMS 콘솔에서 외부 키 스토어를 생성하는 경우 프록시 URI 경로와 프록시 인증 자격 증명을 지정하는 JSON 기반 프록시 구성 파일을 업로드할 수 있습니다. 일부 외부 키 스토어 프록시는 이 파일을 자동으로 생성합니다. 자세한 내용은 외부 키 스토어 프록시 또는 외부 키 관리자의 설명서를 참조하세요.

외부 키 관리자

외부 키 스토어마다 하나 이상의 외부 키 관리자 인스턴스가 필요합니다. 이는 물리적 또는 가상 하드웨어 보안 모듈(HSM)이나 키 관리 소프트웨어일 수 있습니다.

단일 키 관리자를 사용할 수 있지만 중복성을 위해 암호화 키를 공유하는 관련 키 관리자 인스턴스를 두 개 이상 사용하는 것이 좋습니다. 외부 키 스토어에는 외부 키 관리자의 독점 사용이 필요 없습니다. 그러나 외부 키 관리자는 리소스를 보호하기 위해 외부 키 스토어에서 KMS 키를 사용하는 AWS 서비스의 예상 암호화 및 복호화 요청 빈도를 처리할 수 있는 용량이 있어야 합니다. 외부 키 관리자는 초당 최대 1,800개의 요청을 처리하고 각 요청에 대해 250밀리초 제한 시간 내에 응답하도록 구성되어야 합니다. 네트워크 왕복 시간(RTT)이 35밀리초 이하가 AWS 리전 되도록 외부 키 관리자를에 가깝게 배치하는 것이 좋습니다.

외부 키 스토어 프록시에서 허용하는 경우 외부 키 스토어 프록시와 연결하는 외부 키 관리자를 변경할 수 있지만 새 외부 키 관리자는 동일한 키 구성 요소를 사용하는 백업 또는 스냅샷이어야 합니다. KMS 키와 연결하는 외부 키를 외부 키 스토어 프록시에서 더 이상 사용할 수 없는 경우는 KMS 키로 암호화된 사이퍼텍스트를 해독 AWS KMS 할 수 없습니다.

외부 키 관리자가 외부 키 스토어 프록시에 액세스할 수 있어야 합니다. 프록시의 GetHealthStatus 응답이 모든 외부 키 관리자 인스턴스가 `Unavailable` 상태라고 보고하는 경우 외부 키 스토어를 생성하려고 할 때마다 `XksProxyUriUnreachableException`이 발생하여 실패합니다.

외부 키 스토어 프록시

AWS KMS 외부 키 스토어 프록시 API 사양의 설계 요구 사항을 준수하는 외부 키 스토어 프록시(XKS 프록시)를 지정해야 합니다. 외부 키 스토어 프록시를 개발 또는 구매하거나 외부 키 관리자가 제공하거나 외부 키 관리자에 내장된 외부 키 스토어 프록시를 사용할 수 있습니다.는 외부 키 스토어 프록시를 초당 최대 1,800개의 요청을 처리하고 각 요청에 대해 250밀리초 제한 시간 내에 응답하도록 구성할 AWS KMS 것을 권장합니다. 네트워크 왕복 시간(RTT)이 35밀리초 이하가 AWS 리전 되도록 외부 키 관리자를에 가깝게 배치하는 것이 좋습니다.

둘 이상의 외부 키 스토어에 대해 외부 키 스토어 프록시를 사용할 수 있지만, 각 외부 키 스토어에는 요청에 대한 외부 키 스토어 프록시 내에 고유한 URI 엔드포인트 및 경로가 있어야 합니다.

VPC 엔드포인트 서비스 연결을 사용하는 경우 Amazon VPC에서 외부 키 스토어 프록시를 찾을 수 있지만 이것이 필수는 아닙니다. 프라이빗 데이터 센터 AWS와 같이 외부에서 프록시를 찾고 VPC 엔드포인트 서비스를 사용하여 프록시와 통신할 수 있습니다.

프록시 인증 자격 증명

외부 키 스토어를 생성하려면 외부 키 스토어 프록시 인증 자격 증명 (`XksProxyAuthenticationCredential`)을 지정해야 합니다.

외부 키 스토어 프록시 인증 자격 증명과 함께 서명 버전 4(SigV4`XksProxyAuthenticationCredential`) 프로세스를 사용하여 요청에 서명 AWS KMS 하여 외부 키 스토어 프록시에 AWS KMS 대한 인증 자격 증명()을 설정해야 합니다. SigV4 외부 키 스토어를 생성할 때 인증 자격 증명을 지정하며 언제든지 이를 변경할 수 있습니다. 프록시가 자격 증명을 교체하는 경우 외부 키 스토어의 자격 증명 값을 업데이트해야 합니다.

프록시 인증 자격 증명은 두 부분으로 구성됩니다. 외부 키 스토어에 대해 두 부분을 모두 제공해야 합니다.

- 액세스 키 ID: 비밀 액세스 키를 식별합니다. 이 ID를 일반 텍스트로 제공할 수 있습니다.
- 보안 액세스 키: 보안 인증의 보안 암호 부분입니다. 보안 인증 정보를 저장하기 전에 보안 암호 액세스 키를 AWS KMS 암호화합니다.

가 외부 키 스토어 프록시에 대한 요청에 서명하는 데 AWS KMS 사용하는 SigV4 자격 증명은 AWS 계정의 보안 주체와 연결된 SigV4 자격 증명과 관련이 없습니다 AWS Identity and Access Management . 외부 키 스토어 프록시에 IAM SigV4 자격 증명을 재사용하지 마세요.

프록시 연결

외부 키 스토어를 생성하려면 외부 키 스토어 프록시 연결 옵션(`XksProxyConnectivity`)을 지정해야 합니다.

AWS KMS 는 퍼블릭 엔드포인트 또는 Amazon Virtual Private Cloud(VPC) 엔드포인트 서비스를 사용하여 외부 키 스토어 프록시와 통신할 수 있습니다. 퍼블릭 엔드포인트는 구성 및 유지 관리가 더 간단하지만 모든 설치에 대한 보안 요구 사항을 충족하지 못할 수 있습니다. Amazon VPC 엔드포인트 서비스 연결 옵션을 선택하는 경우 서로 다른 두 가용 영역에 두 개 이상의 서브넷이 있는 Amazon VPC, Network Load Balancer와 대상 그룹이 있는 VPC 엔드포인트 서비스, VPC 엔드포인트 서비스의 프라이빗 DNS 이름을 비롯하여 필요한 구성 요소를 생성하고 유지 관리해야 합니다.

외부 키 스토어에 대한 프록시 연결 옵션을 변경할 수 있습니다. 그러나 외부 키 스토어에서 KMS 키와 연결된 키 구성 요소를 계속 사용할 수 있는지 확인해야 합니다. 그렇지 않으면 해당 KMS 키로 암호화된 사이퍼텍스트를 해독할 AWS KMS 수 없습니다.

외부 키 스토어에 가장 적합한 프록시 연결 옵션을 결정하는 방법에 대한 도움말은 외부 키 저장소 프록시 연결 옵션 선택 섹션을 참조하세요. VPC 엔드포인트 서비스 연결을 생성하고 구성하는 방법에 대한 도움말은 VPC 엔드포인트 서비스 연결 구성 섹션을 참조하세요.

프록시 URI 엔드포인트

외부 키 스토어를 생성하려면가 외부 키 스토어 프록시로 요청을 보내는 데 AWS KMS 사용하는 엔드포인트(`XksProxyUriEndpoint`)를 지정해야 합니다.

프로토콜은 포트 443의 IPv4를 통해 HTTPS. AWS KMS communicates여야 합니다. 프록시 URI 엔드포인트 값에 포트를 지정하지 마세요.

- 퍼블릭 엔드포인트 연결 - 외부 키 스토어 프록시에 대해 공개적으로 사용 가능한 엔드포인트를 지정합니다. 외부 키 스토어를 생성하기 전에 이 엔드포인트에 연결할 수 있어야 합니다.
- VPC 엔드포인트 서비스 연결 - `https://` 뒤에 VPC 엔드포인트 서비스의 프라이빗 DNS 이름을 지정합니다.

외부 키 스토어 프록시에 구성된 TLS 서버 인증서는 외부 키 스토어 프록시 URI 엔드포인트의 도메인 이름과 일치해야 하며 외부 키 스토어에 대해 지원되는 인증 기관에서 발급해야 합니다. 목록은 신뢰할 수 있는 인증 기관을 참조하세요. 인증 기관은 TLS 인증서를 발급하기 전에 도메인 소유권 증명을 요구합니다.

TLS 인증서의 주체 일반 이름(CN)은 프라이빗 DNS 이름과 일치해야 합니다. 예를 들어 프라이빗 DNS 이름이 `myproxy-private.xks.example.com`인 경우 TLS 인증서의 CN은 `myproxy-private.xks.example.com` 또는 `*.xks.example.com`이어야 합니다.

프록시 URI 엔드포인트를 변경할 수 있지만, 외부 키 스토어 프록시가 외부 키 스토어의 KMS 키와 연결된 키 구성 요소에 액세스할 수 있는지 확인합니다. 그렇지 않으면 해당 KMS 키로 암호화된 사이퍼텍스트를 해독할 AWS KMS 수 없습니다.

고유성 요구 사항

- 결합된 프록시 URI 엔드포인트(`XksProxyUriEndpoint`) 및 프록시 URI 경로(`XksProxyUriPath`) 값은 AWS 계정 과 리전에서 고유해야 합니다.
- 퍼블릭 엔드포인트 연결이 있는 외부 키 스토어는 프록시 URI 경로 값이 다른 경우 동일한 프록시 URI 엔드포인트를 공유할 수 있습니다.
- 퍼블릭 엔드포인트 연결이 있는 외부 키 스토어는 키 스토어가 다르 AWS 리전더라도 VPC 엔드포인트 서비스 연결이 동일한 외부 키 스토어와 동일한 프록시 URI 엔드포인트 값을 사용할 수 없습니다 AWS 계정.

- VPC 엔드포인트 연결이 있는 각 외부 키 스토어에는 자체 프라이빗 DNS 이름이 있어야 합니다. 프록시 URI 엔드포인트(프라이빗 DNS 이름)는 AWS 계정 및 리전에서 고유해야 합니다.

프록시 URI 경로

외부 키 스토어를 생성하려면 외부 키 스토어 프록시에 필수 프록시 API에 대한 기본 경로를 지정해야 합니다. 값은 /로 시작하고 /kms/xks/v1로 끝나야 하며, 여기서 v1은 외부 키 스토어 프록시에 대한 AWS KMS API의 버전을 나타냅니다. 이 경로에는 /example-prefix/kms/xks/v1과 같은 필수 요소 사이에 선택적 접두사가 포함될 수 있습니다. 이 값을 찾으려면 외부 키 스토어 프록시에 대한 설명서를 참조하세요.

AWS KMS 는 프록시 URI 엔드포인트와 프록시 URI 경로의 연결로 지정된 주소로 프록시 요청을 보냅니다. 예를 들어 프록시 URI 엔드포인트가 https://myproxy.xks.example.com이고 프록시 URI 경로가 인 경우 /kms/xks/v1는 프록시 API 요청을에 AWS KMS 보냅니다https://myproxy.xks.example.com/kms/xks/v1.

프록시 URI 경로를 변경할 수 있지만, 외부 키 스토어 프록시가 외부 키 스토어의 KMS 키와 연결된 키 구성 요소에 액세스할 수 있는지 확인합니다. 그렇지 않으면 해당 KMS 키로 암호화된 사이퍼텍스트를 해독할 AWS KMS 수 없습니다.

고유성 요구 사항

- 결합된 프록시 URI 엔드포인트(XksProxyUriEndpoint) 및 프록시 URI 경로(XksProxyUriPath) 값은 AWS 계정 과 리전에서 고유해야 합니다.

VPC 엔드포인트 서비스

외부 키 스토어 프록시와 통신하는 데 사용되는 Amazon VPC 엔드포인트 서비스의 이름을 지정합니다. 이 구성 요소는 VPC 엔드포인트 서비스 연결을 사용하는 외부 키 스토어에만 필요합니다. 외부 키 스토어의 VPC 엔드포인트 서비스를 설정하고 구성하는 방법에 대한 도움말은 VPC 엔드포인트 서비스 연결 구성 섹션을 참조하세요.

VPC 엔드포인트 서비스에 다음과 같은 속성이 있어야 합니다.

- VPC 엔드포인트 서비스는 외부 키 저장소와 동일한 또는 서로 다른 AWS 계정 에 위치할 수 있습니다.
  - VPC 엔드포인트 서비스는 외부 키 스토어 AWS 리전 와 동일한에 있어야 합니다.
  - VPC 엔드포인트 서비스가 다른에 있는 경우 VPC 엔드포인트 서비스의 AWS 계정 ID를 제공해야 합니다 AWS 계정.

- 각각 다른 가용 영역에 있는 두 개 이상의 서브넷에 연결된 NLB(Network Load Balancer)가 있어야 합니다.
- VPC 엔드포인트 서비스의 허용 보안 주체 목록에는 `cks.kms.<region>.amazonaws.com`와 같은 리전의 AWS KMS 서비스 보안 주체가 포함되어야 합니다`cks.kms.us-east-1.amazonaws.com`.
  - Amazon VPC 엔드포인트 서비스를 외부 키 스토어(XKS) 소유가 아닌 AWS 계정 다른에서 AWS 계정 소유한 경우 VPC 엔드포인트 서비스에 대한 XKS 액세스도 허용해야 합니다. 이렇게 하려면 XKS AWS 계정 ID를 Amazon VPC 엔드포인트 서비스의 보안 주체로 허용 목록에 추가하세요.
- 연결 요청 수락이 필요하지 않아야 합니다.
- 상위 수준의 퍼블릭 도메인 내에 프라이빗 DNS 이름이 있어야 합니다. 예를 들어 퍼블릭 `xks.example.com` 도메인의 프라이빗 DNS 이름이 myproxy-private.xks.example.com일 수 있습니다.

  VPC 엔드포인트 서비스 연결이 있는 외부 키 스토어의 프라이빗 DNS 이름은 AWS 리전에서 고유해야 합니다.
- 프라이빗 DNS 이름 도메인의 도메인 확인 상태는 `verified`여야 합니다.
- 외부 키 스토어 프록시에 구성된 TLS 서버 인증서는 엔드포인트에 연결할 수 있는 프라이빗 DNS 호스트 이름을 지정해야 합니다.

고유성 요구 사항

- VPC 엔드포인트 연결이 있는 외부 키 스토어가 Amazon VPC를 공유할 수 있지만 각 외부 키 스토어에는 자체 VPC 엔드포인트 서비스와 프라이빗 DNS 이름이 있어야 합니다.

프록시 구성 파일

프록시 구성 파일은 외부 키 스토어의 프록시 URI 경로와 프록시 인증 자격 증명 속성에 대한 값을 포함하는 선택적 JSON 기반 파일입니다. AWS KMS 콘솔에서 외부 키 스토어를 생성하거나 편집할 때 프록시 구성 파일을 업로드하여 외부 키 스토어에 대한 구성 값을 제공할 수 있습니다. 이 파일을 사용하면 입력 및 붙여넣기 오류를 방지하고 외부 키 스토어의 값이 외부 키 스토어 프록시의 값과 일치하는지 확인할 수 있습니다.

프록시 구성 파일은 외부 키 스토어 프록시에 의해 생성됩니다. 외부 키 스토어 프록시가 프록시 구성 파일을 제공하는지 알아보려면 외부 키 스토어 프록시 설명서를 참조하세요.

다음은 가상 값이 포함된 올바른 형식의 프록시 구성 파일의 예입니다.

```
{
  "XksProxyUriPath": "/example-prefix/kms/xks/v1",
  "XksProxyAuthenticationCredential": {
    "AccessKeyId": "ABCDE12345670EXAMPLE",
    "RawSecretAccessKey": "0000EXAMPLEFA5FT0mCc3DrGUe2sti527BitkQ0Zr9MO9+vE="
  }
}
```

AWS KMS 콘솔에서 외부 키 스토어를 생성하거나 편집할 때만 프록시 구성 파일을 업로드할 수 있습니다. CreateCustomKeyStore 또는 UpdateCustomKeyStore 작업과 함께 사용할 수는 없지만 프록시 구성 파일의 값을 사용하여 파라미터 값이 올바른지 확인할 수 있습니다.

새로운 외부 키 저장소 생성

필요한 사전 조건을 조합한 후에는 AWS KMS 콘솔에서 또는 CreateCustomKeyStore 작업을 사용하여 새 외부 키 스토어를 생성할 수 있습니다.

AWS KMS 콘솔 사용

외부 키 저장소를 생성하기 전에 프록시 연결 유형을 선택한 다음, 필요한 구성 요소를 모두 생성하고 구성했는지 확인합니다. 필요한 값을 찾는 데 도움이 필요한 경우 외부 키 스토어 프록시 또는 키 관리 소프트웨어에 대한 설명서를 참조하세요.

> **Note**
>
> 에서 외부 키 스토어를 생성할 때 프록시 URI 경로 및 프록시 인증 자격 증명 값이 포함된 JSON 기반 프록시 구성 파일을 업로드 AWS Management Console할 수 있습니다. ??? ??? 일부 프록시는 이 파일을 자동으로 생성합니다. 이 값은 필수가 아닙니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 탐색 창에서 Custom key stores(사용자 지정 키 스토어), External key stores(외부 키 스토어)를 선택합니다.
4. Create external key store(외부 키 스토어 생성)를 선택합니다.
5. 기억하기 쉬운 외부 키 스토어 이름을 입력합니다. 이름은 계정의 모든 외부 키 스토어에서 고유해야 합니다.

> **Important**
>
> 이 필드에 기밀 또는 민감한 정보를 포함하지 마십시오. 이 필드는 CloudTrail 로그 및 기타 출력에 일반 텍스트로 표시될 수 있습니다.

6. 프록시 연결 유형을 선택합니다.

   프록시 연결 선택에 따라 외부 키 스토어 프록시에 필요한 구성 요소가 결정됩니다. 이 선택에 대한 도움말은 외부 키 저장소 프록시 연결 옵션 선택 섹션을 참조하세요.

   a. VPC 엔드포인트 서비스가 다른 AWS 계정에 있는 경우 교차 계정 VPC 엔드포인트 서비스를 선택합니다. 그런 다음 VPC 엔드포인트 서비스 소유자 계정 AWS 계정 ID 필드에 VPC 엔드포인트 소유자의 ID를 입력합니다.

   b. 이 외부 키 스토어에 대한 VPC 엔드포인트 서비스의 이름을 선택하거나 입력합니다. 이 단계는 외부 키 저장소 프록시 연결 유형이 VPC 엔드포인트 서비스인 경우에만 나타납니다.

      VPC 엔드포인트 서비스와 해당 VPC가 외부 키 스토어의 요구 사항을 충족해야 합니다. 자세한 내용은 the section called “사전 조건 수집”을 참조하세요.

7. 이 외부 키 스토어에 대한 VPC 엔드포인트 서비스의 이름을 선택하거나 입력합니다. 이 단계는 외부 키 스토어 프록시 연결 유형이 VPC endpoint service(VPC 엔드포인트 서비스)인 경우에만 나타납니다.

   VPC 엔드포인트 서비스와 해당 VPC가 외부 키 스토어의 요구 사항을 충족해야 합니다. 자세한 내용은 the section called “사전 조건 수집”을 참조하세요.

8. 프록시 URI 엔드포인트를 입력합니다. 프로토콜은 포트 443의 IPv4를 통해 HTTPS. AWS KMS communicates여야 합니다. 프록시 URI 엔드포인트 값에 포트를 지정하지 마세요.

   가 이전 단계에서 지정한 VPC 엔드포인트 서비스를 AWS KMS 인식하면이 필드가 자동으로 작성됩니다.

   퍼블릭 엔드포인트 연결로 공개적으로 사용 가능한 엔드포인트 URI를 입력합니다. VPC 엔드포인트 연결로 `https://` 뒤에 VPC 엔드포인트 서비스의 프라이빗 DNS 이름을 지정합니다.

9. 프록시 URI 경로 접두사 및 프록시 인증 자격 증명의 값을 입력하려면 프록시 구성 파일을 업로드하거나 값을 수동으로 입력합니다.

   - 프록시 URI 경로 및 프록시 인증 자격 증명에 대한 값이 포함된 선택적 프록시 구성 파일이 있는 경우 Upload configuration file(구성 파일 업로드)을 선택합니다. 단계에 따라 파일을 업로드합니다.

파일이 업로드되면 콘솔은 편집 가능한 필드에 파일의 값을 표시합니다. 지금 값을 변경하거나 외부 키 스토어를 생성한 후 이 값을 편집할 수 있습니다.

비밀 액세스 키의 값을 표시하려면 Show secret access key(비밀 액세스 키 표시)를 선택합니다.

- 프록시 구성 파일이 없는 경우 프록시 URI 경로 및 프록시 인증 자격 증명 값을 수동으로 입력할 수 있습니다.

  a. 프록시 구성 파일이 없는 경우에는 프록시 URI를 수동으로 입력할 수 있습니다. 콘솔에서 필요한 /kms/xks/v1 값을 제공합니다.

  프록시 URI 경로에 선택적 접두사(예: `/example-prefix/kms/xks/v1`의 `example-prefix`)가 포함된 경우 Proxy URI path prefix(프록시 URI 경로 접두사) 필드에 접두사를 입력합니다. 그렇지 않은 경우 필드를 비워둡니다.

  b. 프록시 구성 파일이 없는 경우 프록시 인증 자격 증명을 수동으로 입력할 수 있습니다. 액세스 키 ID와 비밀 액세스 키가 모두 필요합니다.

  - Proxy credential: Access key ID(프록시 자격 증명: 액세스 키 ID)에 프록시 인증 자격 증명의 액세스 키 ID를 입력합니다. 액세스 키 ID가 비밀 액세스 키를 식별합니다.
  - Proxy credential: Secret access key(프록시 자격 증명: 비밀 액세스 키)에 프록시 인증 자격 증명의 비밀 액세스 키를 입력합니다.

  비밀 액세스 키의 값을 표시하려면 Show secret access key(비밀 액세스 키 표시)를 선택합니다.

  이 절차는 외부 키 스토어 프록시에 설정한 인증 자격 증명을 설정하거나 변경하지 않습니다. 단지 이러한 값을 외부 키 스토어와 연결합니다. 프록시 인증 자격 증명 설정, 변경 및 교체에 대한 자세한 내용은 외부 키 스토어 프록시 또는 키 관리 소프트웨어 설명서를 참조하세요.

  프록시 인증 자격 증명이 변경되면 외부 키 스토어의 자격 증명 설정을 편집합니다.

10. Create external key store(외부 키 스토어 생성)를 선택합니다.

절차가 성공하면 계정 및 리전의 외부 키 스토어 목록에 새로운 외부 키 스토어가 나타납니다. 절차가 실패하면 오류 메시지가 나타나서 문제를 설명하고 이를 수정할 수 있는 방법에 대한 도움말을 제공합니다. 도움이 더 필요한 경우 외부 키에 대한 CreateKey 오류 섹션을 참조하십시오.

다음: 새로운 외부 키 스토어는 자동으로 연결되지 않습니다. 외부 키 스토어 AWS KMS keys 에서를 생성하려면 먼저 외부 키 스토어 프록시에 외부 키 스토어를 연결해야 합니다.

AWS KMS API 사용

CreateCustomKeyStore 작업을 사용하여 외부 키 스토어를 새로 생성할 수 있습니다. 필수 파라미터의 값을 찾는 방법에 대한 도움말은 외부 키 스토어 프록시 또는 키 관리 소프트웨어 설명서를 참조하세요.

> **Tip**
>
> `CreateCustomKeyStore` 작업을 사용할 때는 프록시 구성 파일을 업로드할 수 없습니다. 그러나 프록시 구성 파일의 값을 사용하여 파라미터 값이 올바른지 확인할 수 있습니다.

외부 키 스토어를 생성하려면 `CreateCustomKeyStore` 작업에 다음 파라미터 값이 필요합니다.

- `CustomKeyStoreName` – 계정에서 고유한 기억하기 쉬운 외부 키 스토어의 이름입니다.

  > **Important**
  >
  > 이 필드에 기밀 또는 민감한 정보를 포함하지 마십시오. 이 필드는 CloudTrail 로그 및 기타 출력에 일반 텍스트로 표시될 수 있습니다.

- `CustomKeyStoreType` - `EXTERNAL_KEY_STORE`를 지정합니다.
- `XksProxyConnectivity` – `PUBLIC_ENDPOINT` 또는 `VPC_ENDPOINT_SERVICE`를 지정합니다.
- `XksProxyAuthenticationCredential` - 액세스 키 ID와 비밀 액세스 키를 모두 지정합니다.
- `XksProxyUriEndpoint` - AWS KMS 가 외부 키 스토어 프록시와 통신하는 데 사용하는 엔드포인트입니다.
- `XksProxyUriPath` - 프록시 API에 대한 프록시 내의 경로입니다.
- `XksProxyVpcEndpointServiceName` - `XksProxyConnectivity` 값이 `VPC_ENDPOINT_SERVICE`인 경우에만 필요합니다.

> **Note**
>
> AWS CLI 버전 1.0을 사용하는 경우 파라미터와 같은 HTTP 또는 HTTPS 값으로 `XksProxyUriEndpoint` 파라미터를 지정하기 전에 다음 명령을 실행합니다.

```
aws configure set cli_follow_urlparam false
```

그렇지 않으면 AWS CLI 버전 1.0이 파라미터 값을 해당 URI 주소에 있는 콘텐츠로 대체하여 다음 오류가 발생합니다.

```
Error parsing parameter '--xks-proxy-uri-endpoint': Unable to retrieve
https:// : received non 200 status code of 404
```

다음 예제에서는 가상의 값을 사용합니다. 명령을 실행하기 전에 해당 값을 외부 키 스토어의 유효한 값으로 바꾸세요.

퍼블릭 엔드포인트 연결이 있는 외부 키 스토어를 생성합니다.

```
$ aws kms create-custom-key-store
        --custom-key-store-name ExampleExternalKeyStorePublic \
        --custom-key-store-type EXTERNAL_KEY_STORE \
        --xks-proxy-connectivity PUBLIC_ENDPOINT \
        --xks-proxy-uri-endpoint https://myproxy.xks.example.com \
        --xks-proxy-uri-path /kms/xks/v1 \
        --xks-proxy-authentication-credential
 AccessKeyId=<value>,RawSecretAccessKey=<value>
```

VPC 엔드포인트 서비스 연결이 있는 외부 키 스토어를 생성합니다.

```
$ aws kms create-custom-key-store
        --custom-key-store-name ExampleExternalKeyStoreVPC \
        --custom-key-store-type EXTERNAL_KEY_STORE \
        --xks-proxy-connectivity VPC_ENDPOINT_SERVICE \
        --xks-proxy-vpc-endpoint-service-name com.amazonaws.vpce.us-east-1.vpce-svc-
example \
        --xks-proxy-uri-endpoint https://myproxy-private.xks.example.com \
        --xks-proxy-uri-path /kms/xks/v1 \
        --xks-proxy-authentication-credential
 AccessKeyId=<value>,RawSecretAccessKey=<value>
```

작업이 성공하면 CreateCustomKeyStore가 사용자 지정 키 스토어 ID를 반환합니다(다음 예제 응답 참조).

```
{
    "CustomKeyStoreId": cks-1234567890abcdef0
}
```

작업이 실패할 경우 예외로 표시된 오류를 정정하고 다시 시도해보세요. 추가적인 도움말은 외부 키 스토어 문제 해결 섹션을 참조하십시오.

다음: 외부 키 스토어를 사용하려면 외부 키 스토어 프록시에 연결합니다.

외부 키 저장소 속성 편집

기존 외부 키 스토어의 선택된 속성을 편집할 수 있습니다.

외부 키 스토어가 연결되거나 연결 해제된 상태에서 일부 속성을 편집할 수 있습니다. 다른 속성의 경우 먼저 외부 키 스토어 프록시에서 외부 키 스토어를 연결 해제해야 합니다. 외부 키 스토어의 연결 상태는 `DISCONNECTED`여야 합니다. 외부 키 스토어가 연결 해제된 동안 사용자가 키 스토어와 해당 KMS 키를 관리할 수 있지만, 외부 키 스토어에서 KMS 키를 생성 또는 사용할 수 없습니다. 외부 키 스토어의 연결 상태를 찾으려면 DescribeCustomKeystore 작업을 사용하거나 외부 키 스토어의 세부 정보 페이지에서 General configuration(일반 구성) 섹션을 봅니다.

외부 키 스토어의 속성을 업데이트하기 전에는 새 값을 사용하여 외부 키 스토어 프록시에 GetHealthStatus 요청을 AWS KMS 보냅니다. 요청이 성공하면 업데이트된 속성 값을 사용하여 외부 키 스토어 프록시에 연결하고 인증할 수 있는 것입니다. 요청이 실패하면 오류를 식별하는 예외와 함께 편집 작업이 실패합니다.

편집 작업이 완료되면 외부 키 스토어의 업데이트된 속성 값이 AWS KMS 콘솔과 `DescribeCustomKeyStores` 응답에 나타납니다. 그러나 변경 내용이 완전히 적용되는 데 최대 5분이 걸릴 수 있습니다.

AWS KMS 콘솔에서 외부 키 스토어를 편집하는 경우 프록시 URI 경로와 프록시 인증 자격 증명을 지정하는 JSON 기반 프록시 구성 파일을 업로드할 수 있습니다. 일부 외부 키 스토어 프록시는 이 파일을 자동으로 생성합니다. 자세한 내용은 외부 키 스토어 프록시 또는 외부 키 관리자의 설명서를 참조하세요.

> **Warning**
>
> 업데이트된 속성 값은 이전 값과 동일한 외부 키 관리자 또는 동일한 암호화 키를 사용하는 외부 키 관리자의 백업 또는 스냅샷에 대한 프록시에 외부 키 스토어를 연결해야 합니다. 외부 키 스토어가 KMS 키와 연결된 외부 키에 대한 액세스 권한을 영구적으로 상실하는 경우 해당 외

부 키로 암호화된 사이퍼텍스트를 복구할 수 없습니다. 특히 외부 키 스토어의 프록시 연결을 변경하면가 외부 키 AWS KMS 에 액세스하지 못할 수 있습니다.

> **Tip**
>
> 일부 외부 키 관리자는 외부 키 스토어 속성을 편집하는 더 간단한 방법을 제공합니다. 자세한 내용은 외부 키 관리자 설명서를 참조하세요.

외부 키 스토리의 다음 속성을 변경할 수 있습니다.

편집 가능한 외부 키 스토어 속성	연결 상태	연결 해제됨 상태 필요
사용자 지정 키 스토어 이름 사용자 지정 키 스토어에 필요한 이름입니다. **Important** 이 필드에 기밀 또는 민감한 정보를 포함하지 마십시오. 이 필드는 CloudTrail 로그 및 기타 출력에 일반 텍스트로 표시될 수 있습니다.	✓	
프록시 인증 자격 증명(XksProxyAuthenticationCredential) (액세스 키 ID와 비밀 액세스 키 모두 지정해야 합니다. 하나의 요소만 변경하는 경우에도 마찬가지입니다.)	✓	
프록시 URI 경로(XksProxyUriPath)	✓	
프록시 연결(XksProxyConnectivity)		✓

편집 가능한 외부 키 스토어 속성	연결 상태	연결 해제됨 상태 필요
(프록시 URI 엔드포인트도 업데이트해야 합니다. VPC 엔드포인트 서비스 연결로 변경하는 경우 프록시 VPC 엔드포인트 서비스 이름을 지정해야 합니다.)		
프록시 URI 엔드포인트(XksProxyUriEndpoint) 프록시 엔드포인트 URI를 변경하는 경우 연결된 TLS 인증서도 변경해야 할 수 있습니다.		✓
프록시 VPC 엔드포인트 서비스 이름(XksProxyVpcEndpointServiceName) (이 필드는 VPC 엔드포인트 서비스 연결에 필요합니다.)		✓
VPC 엔드포인트 서비스 소유자(XksProxyVpcEndpointServiceOwner) (이 필드는 VPC 엔드포인트 서비스 연결에 필요합니다.)		✓

외부 키 저장소의 속성 편집

AWS KMS 콘솔에서 또는 UpdateCustomKeyStore 작업을 사용하여 외부 키 스토어의 속성을 편집할 수 있습니다.

AWS KMS 콘솔 사용

키 스토어를 편집할 때 편집 가능한 값을 변경할 수 있습니다. 일부 변경을 위해서는 외부 키 스토어와 외부 키 스토어 프록시를 연결 해제해야 합니다.

프록시 URI 경로 또는 프록시 인증 자격 증명을 편집하는 경우 새 값을 입력하거나 새 값이 포함된 외부 키 스토어 프록시 구성 파일을 업로드할 수 있습니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.

2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 탐색 창에서 Custom key stores(사용자 지정 키 스토어), External key stores(외부 키 스토어)를 선택합니다.
4. 편집하고 싶은 키 저장소를 선택합니다.

   - 필요한 경우 외부 키 스토어 프록시에서 외부 키 스토어를 연결 해제합니다. Key store actions(키 스토어 작업) 메뉴에서 Disconnect(연결 해제)를 선택합니다.
5. Key store actions(키 스토어 작업) 메뉴에서 Edit(편집)를 선택합니다.
6. 편집 가능한 외부 키 스토어 속성을 하나 이상 변경합니다. 프록시 URI 경로 및 프록시 인증 자격 증명에 대한 값이 있는 외부 키 스토어 프록시 구성 파일을 업로드할 수도 있습니다. 파일에 지정된 일부 값이 변경되지 않은 경우에도 프록시 구성 파일을 사용할 수 있습니다.
7. Update external key store(외부 키 스토어 업데이트)를 선택합니다.
8. 경고를 검토하고 계속하기로 결정한 경우 경고를 확인한 다음 Update external key store(외부 키 스토어 업데이트)를 선택합니다.

   절차가 성공하면 편집한 속성을 설명하는 메시지가 표시됩니다. 절차가 실패하면 오류 메시지가 나타나서 문제를 설명하고 이를 수정할 수 있는 방법에 대한 도움말을 제공합니다.
9. 필요한 경우 외부 키 스토어를 다시 연결합니다. Key store actions(키 스토어 작업) 메뉴에서 Connect(연결)를 선택합니다.

   외부 키 스토어를 연결 해제된 상태로 남겨둘 수 있습니다. 하지만 연결 해제되어 있는 동안에는 외부 키 스토어에서 KMS 키를 생성하거나, 암호화 작업에서 외부 키 스토어의 KMS 키를 사용할 수 없습니다.

AWS KMS API 사용

외부 키 스토어의 속성을 변경하려면 UpdateCustomKeyStore 작업을 사용합니다. 동일한 작업으로 외부 키 스토어에서 여러 개의 속성을 변경할 수 있습니다. 작업이 성공하면 HTTP 200 응답과 속성이 없는 JSON 객체를 AWS KMS 반환합니다.

`CustomKeyStoreId` 파라미터를 사용하여 외부 키 스토어를 식별합니다. 다른 파라미터를 사용하여 속성을 변경합니다. `UpdateCustomKeyStore` 작업과 함께 프록시 구성 파일을 사용할 수 없습니다. 프록시 구성 파일은 AWS KMS 콘솔에서만 지원됩니다. 그러나 프록시 구성 파일을 사용하여 외부 키 스토어 프록시에 대한 올바른 파라미터 값을 결정할 수 있습니다.

이 섹션의 예제는 AWS Command Line Interface (AWS CLI)를 사용하지만, 지원되는 모든 프로그래밍 언어를 사용할 수 있습니다.

시작하기 전에 필요한 경우 외부 키 스토어 프록시에서 외부 키 스토어를 연결 해제합니다. 업데이트 후 필요한 경우 외부 키 스토어 프록시에 외부 키 스토어를 다시 연결할 수 있습니다. 외부 키 스토어를 연결 해제 상태로 남겨둘 수 있지만, 키 스토어에서 새 KMS 키를 생성하거나 키 스토어에서 기존 KMS 키를 암호화 작업에 사용할 수 있으려면 먼저 이를 다시 연결해야 합니다.

> **Note**
>
> AWS CLI 버전 1.0을 사용하는 경우 파라미터와 같은 HTTP 또는 HTTPS 값으로 `XksProxyUriEndpoint` 파라미터를 지정하기 전에 다음 명령을 실행합니다.
>
> ```
> aws configure set cli_follow_urlparam false
> ```
>
> 그렇지 않으면 AWS CLI 버전 1.0이 파라미터 값을 해당 URI 주소에 있는 콘텐츠로 대체하여 다음 오류가 발생합니다.
>
> ```
> Error parsing parameter '--xks-proxy-uri-endpoint': Unable to retrieve
> https:// : received non 200 status code of 404
> ```

외부 키 스토어의 이름 변경

첫 번째 예제에서 UpdateCustomKeyStore 작업을 사용하여 기억하기 쉬운 외부 키 스토어의 이름을 `XksKeyStore`로 변경합니다. 이 명령은 `CustomKeyStoreId` 파라미터를 사용해 사용자 지정 키 스토어를 식별하고, `CustomKeyStoreName` 파라미터를 사용해 사용자 지정 키 스토어에서 새 이름을 지정합니다. 모든 예제 값을 외부 키 스토어의 실제 값으로 바꿉니다.

```
$ aws kms update-custom-key-store --custom-key-store-id cks-1234567890abcdef0 --new-
custom-key-store-name XksKeyStore
```

프록시 인증 자격 증명 변경

다음 예제에서는 AWS KMS 가 외부 키 스토어 프록시에 인증하는 데 사용하는 프록시 인증 자격 증명을 업데이트합니다. 이와 같은 명령을 사용하여 프록시에서 교체되는 경우 자격 증명을 업데이트할 수 있습니다.

먼저 외부 키 스토어 프록시에서 자격 증명을 업데이트합니다. 그런 다음 이 기능을 사용하여 AWS KMS에 변경 사항을 보고합니다. (프록시가 이전 자격 증명과 새 자격 증명을 모두 지원하므로 자격 증명을 업데이트할 시간이 있습니다 AWS KMS.)

자격 증명에 액세스 키 ID와 보안 액세스 키를 모두 지정해야 합니다. 하나의 값만 변경된 경우에도 마찬가지입니다.

처음 두 명령은 자격 증명 값을 보유하도록 변수를 설정합니다. UpdateCustomKeyStore 작업은 CustomKeyStoreId 파라미터를 사용하여 외부 키 스토어를 식별합니다. XksProxyAuthenticationCredential 파라미터를 AccessKeyId 및 RawSecretAccessKey 필드와 함께 사용하여 새 자격 증명을 지정합니다. 모든 예제 값을 외부 키 스토어의 실제 값으로 바꿉니다.

```
$ accessKeyID=access key id
$ secretAccessKey=secret access key

$ aws kms update-custom-key-store --custom-key-store-id cks-1234567890abcdef0 \
       --xks-proxy-authentication-credential \
           AccessKeyId=$accessKeyId,RawSecretAccessKey=$secretAccessKey
```

프록시 URI 경로 변경

다음 예제에서는 프록시 URI 경로(XksProxyUriPath)를 업데이트합니다. 프록시 URI 엔드포인트와 프록시 URI 경로의 조합은 AWS 계정 및 리전에서 고유해야 합니다. 모든 예제 값을 외부 키 스토어의 실제 값으로 바꿉니다.

```
$ aws kms update-custom-key-store --custom-key-store-id cks-1234567890abcdef0 \
           --xks-proxy-uri-path /kms/xks/v1
```

VPC 엔드포인트 서비스 연결로 변경

다음 예제에서는 UpdateCustomKeyStore 작업을 사용하여 외부 키 스토어 프록시 연결 유형을 VPC_ENDPOINT_SERVICE로 변경합니다. 이렇게 변경하려면 VPC 엔드포인트 서비스 이름(XksProxyVpcEndpointServiceName)과 VPC 엔드포인트 서비스의 프라이빗 DNS 이름을 포함하는 프록시 URI 엔드포인트(XksProxyUriEndpoint) 값을 포함하여 VPC 엔드포인트 서비스 연결에 필요한 값을 지정해야 합니다. 모든 예제 값을 외부 키 스토어의 실제 값으로 바꿉니다.

```
$ aws kms update-custom-key-store --custom-key-store-id cks-1234567890abcdef0 \
           --xks-proxy-connectivity "VPC_ENDPOINT_SERVICE" \
           --xks-proxy-uri-endpoint https://myproxy-private.xks.example.com \
           --xks-proxy-vpc-endpoint-service-name com.amazonaws.vpce.us-east-1.vpce-
svc-example
```

퍼블릭 엔드포인트 연결로 변경

다음 예제에서는 외부 키 스토어 프록시 연결 유형을 PUBLIC_ENDPOINT로 변경합니다. 이렇게 변경할 때는 프록시 URI 엔드포인트(XksProxyUriEndpoint) 값을 업데이트해야 합니다. 모든 예제 값을 외부 키 스토어의 실제 값으로 바꿉니다.

> **Note**
>
> VPC 엔드포인트 연결은 퍼블릭 엔드포인트 연결보다 더 강력한 보안을 제공합니다. 퍼블릭 엔드포인트 연결로 변경하기 전에 온프레미스에서 외부 키 스토어 프록시를 찾고 통신에만 VPC를 사용하는 등 다른 옵션을 고려해 보세요.

```
$ aws kms update-custom-key-store --custom-key-store-id cks-1234567890abcdef0 \
            --xks-proxy-connectivity "PUBLIC_ENDPOINT" \
            --xks-proxy-uri-endpoint https://myproxy.xks.example.com
```

외부 키 저장소 보기

AWS KMS 콘솔을 사용하거나 DescribeCustomKeyStores 작업을 사용하여 각 계정 및 리전의 외부 키 스토어를 볼 수 있습니다.

외부 키 스토어를 볼 때 다음이 표시됩니다.

- 키 스토어의 이름, ID, 키 스토어 유형 및 생성 날짜를 포함한 키 스토어에 대한 기본 정보.
- 연결 유형, 프록시 URI 엔드포인트 및 경로, 현재 프록시 인증 자격 증명의 액세스 키 ID를 비롯한 외부 키 스토어 프록시에 대한 구성 정보.
- 외부 키 스토어 프록시가 VPC 엔드포인트 서비스 연결을 사용하는 경우 콘솔에 VPC 엔드포인트 서비스의 이름이 표시됩니다.
- 현재 연결 상태.

> **Note**
>
> 연결 상태 값이 Disconnected(연결 해제됨)면 외부 키 스토어가 연결된 적이 없거나 의도적으로 해당 외부 키 스토어 프록시에서 연결 해제된 것입니다. 하지만 연결된 외부 키 스토어에서 KMS 키를 사용하려는 시도가 실패하는 것은 외부 키 스토어 또는 프록시에 문제가 있음을 의미할 수 있습니다. 도움말은 외부 키 스토어 연결 오류를 참조하십시오.

- 외부 키 스토어의 문제를 감지하고 해결하는 데 도움이 되도록 설계된 Amazon CloudWatch 지표 그래프가 포함된 Monitoring(모니터링) 섹션. 그래프를 해석하고, 계획 및 문제 해결에 그래프를 사용하고, 그래프의 지표를 기반으로 CloudWatch 경보를 생성하는 방법에 대한 도움말은 외부 키 저장소 모니터링 섹션을 참조하세요.

외부 키 스토어 속성

외부 키 스토어의 다음 속성은 AWS KMS 콘솔과 DescribeCustomKeyStores 응답에 표시됩니다.

사용자 지정 키 스토어 속성

다음 값은 각 사용자 지정 키 스토어에 대한 세부 정보 페이지의 일반 구성 섹션에 표시됩니다. 이러한 속성은 AWS CloudHSM 키 스토어 및 외부 키 스토어를 포함한 모든 사용자 지정 키 스토어에 적용됩니다.

사용자 지정 키 스토어 ID

가 사용자 지정 키 스토어에 AWS KMS 할당하는 고유 ID입니다.

사용자 지정 키 스토어 이름

사용자 지정 키 스토어를 생성할 때 할당하는 친숙한 이름입니다. 이 값은 언제든지 변경할 수 있습니다.

사용자 지정 키 스토어 유형

사용자 지정 키 스토어의 유형입니다. 유효한 값은 AWS CloudHSM (`AWS_CLOUDHSM`) 또는 외부 키 스토어(`EXTERNAL_KEY_STORE`)입니다. 사용자 지정 키 스토어를 생성한 후에는 유형을 변경할 수 없습니다.

Creation(생성) 날짜

사용자 지정 키 스토어가 생성된 날짜입니다. 이 날짜는 AWS 리전의 로컬 시간으로 표시됩니다.

연결 상태

사용자 지정 키 스토어가 백업 키 스토어에 연결되어 있는지 여부를 나타냅니다. 연결 상태는 사용자 지정 키 스토어가 백업 키 스토어에 연결된 적이 없거나 의도적으로 연결 해제된 경우에만 `DISCONNECTED`입니다. 자세한 내용은 the section called “연결 상태”을 참조하세요.

외부 키 스토어 구성 속성

각 외부 키 스토어에 대한 세부 정보 페이지의 External key store proxy configuration(외부 키 스토어 프록시 구성) 섹션과 DescribeCustomKeyStores 응답의 `XksProxyConfiguration` 요소에 다음 값이 나타납니다. 고유성 요구 사항을 포함한 각 필드에 대한 자세한 설명과 각 필드의 올바른 값 결정에 대한 도움말은 외부 키 스토어 생성 주제의 the section called “사전 조건 수집” 섹션을 참조하세요.

프록시 연결

외부 키 스토어가 퍼블릭 엔드포인트 연결을 사용하는지 아니면 VPC 엔드포인트 서비스 연결을 사용하는지를 나타냅니다.

프록시 URI 엔드포인트

가 외부 키 스토어 프록시에 연결하는 데 AWS KMS 사용하는 엔드포인트입니다.

프록시 URI 경로

가 프록시 API 요청을 AWS KMS 보내는 프록시 URI 엔드포인트의 경로입니다. ???

프록시 자격 증명: 액세스 키 ID

외부 키 스토어 프록시에서 설정하는 프록시 인증 자격 증명의 일부입니다. 액세스 키 ID는 자격 증명에서 비밀 액세스 키를 식별합니다.

AWS KMS 는 SigV4 서명 프로세스와 프록시 인증 자격 증명을 사용하여 외부 키 스토어 프록시에 대한 요청에 서명합니다. 서명의 자격 증명을 사용하면 외부 키 스토어 프록시가 사용자를 대신하여 요청을 인증할 수 있습니다 AWS KMS.

VPC 엔드포인트 서비스 이름

외부 키 스토어를 지원하는 Amazon VPC 엔드포인트 서비스의 이름입니다. 이 값은 외부 키 스토어가 PC 엔드포인트 서비스 연결을 사용하는 경우에만 나타납니다. VPC에서 외부 키 스토어 프록시를 찾거나 VPC 엔드포인트 서비스를 사용하여 외부 키 스토어 프록시와 안전하게 통신할 수 있습니다.

VPC 엔드포인트 서비스 소유자 ID

외부 키 저장소를 지원하는 Amazon VPC 엔드포인트 서비스의 ID입니다. 이 값은 외부 키 스토어가 PC 엔드포인트 서비스 연결을 사용하는 경우에만 나타납니다. VPC에서 외부 키 스토어 프록시를 찾거나 VPC 엔드포인트 서비스를 사용하여 외부 키 스토어 프록시와 안전하게 통신할 수 있습니다.

외부 키 저장소 속성 보기

AWS KMS 콘솔에서 또는 DescribeCustomKeyStores 작업을 사용하여 외부 키 스토어 및 관련 속성을 볼 수 있습니다.

AWS KMS 콘솔 사용

해당 계정 및 리전에서 외부 키 스토어를 보려면 다음 절차를 사용하세요.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단의 리전 선택기를 AWS 리전사용합니다.
3. 탐색 창에서 Custom key stores(사용자 지정 키 스토어), External key stores(외부 키 스토어)를 선택합니다.
4. 외부 키 스토어에 대한 자세한 정보를 보려면 키 스토어 이름을 선택합니다.

AWS KMS API 사용

외부 키 스토어를 보려면 DescribeCustomKeyStores 작업을 사용합니다. 기본적으로 이 작업은 계정 및 리전에서 모든 사용자 지정 키 스토어를 반환합니다. 그러나 `CustomKeyStoreId` 또는 `CustomKeyStoreName` 파라미터(둘 중 하나만)를 사용해서 특정한 사용자 지정 키 스토어로 출력을 제한할 수 있습니다.

사용자 지정 키 스토어의 경우 출력은 사용자 지정 키 스토어 ID, 이름 및 유형과 키 스토어의 연결 상태로 이루어집니다. 연결 상태가 `FAILED`인 경우 출력에는 오류 원인을 설명하는 `ConnectionErrorCode`도 포함됩니다. 외부 키 스토어에 대한 `ConnectionErrorCode`를 해석하는 방법에 대한 도움말은 외부 키 스토어의 연결 오류 코드 섹션을 참조하세요.

외부 키 스토어의 경우 출력에 `XksProxyConfiguration` 요소도 포함됩니다. 이 요소에는 연결 유형, 프록시 URI 엔드포인트, 프록시 URI 경로 및 프록시 인증 자격 증명의 액세스 키 ID가 포함됩니다.

이 섹션의 예제는 AWS Command Line Interface (AWS CLI)를 사용하지만, 지원되는 모든 프로그래밍 언어를 사용할 수 있습니다.

예를 들어 다음 명령은 계정 및 리전에 모든 사용자 지정 키 스토어를 반환합니다. `Limit` 및 `Marker` 파라미터를 사용해 출력에서 사용자 지정 키 스토어를 탐색할 수 있습니다.

```
$ aws kms describe-custom-key-stores
```

다음 명령은 CustomKeyStoreName 파라미터를 사용해 기억하기 쉬운 이름이 ExampleXksPublic인 예제 외부 키 스토어만 검색합니다. 이 예제 키 스토어는 퍼블릭 엔드포인트 연결을 사용하며 외부 키 스토어 프록시에 연결됩니다.

```
$ aws kms describe-custom-key-stores --custom-key-store-name ExampleXksPublic
{
    "CustomKeyStores": [
    {
      "CustomKeyStoreId": "cks-1234567890abcdef0",
      "CustomKeyStoreName": "ExampleXksPublic",
      "ConnectionState": "CONNECTED",
      "CreationDate": "2022-12-14T20:17:36.419000+00:00",
      "CustomKeyStoreType": "EXTERNAL_KEY_STORE",
      "XksProxyConfiguration": {
        "AccessKeyId": "ABCDE12345670EXAMPLE",
        "Connectivity": "PUBLIC_ENDPOINT",
        "UriEndpoint": "https://xks.example.com:6443",
        "UriPath": "/example/prefix/kms/xks/v1"
      }
    }
  ]
}
```

이 명령은 VPC 엔드포인트 서비스 연결이 있는 예제 외부 키 스토어를 가져옵니다. 이 예제에서는 외부 키 스토어가 해당 외부 키 스토어 프록시에 연결됩니다.

```
$ aws kms describe-custom-key-stores --custom-key-store-name ExampleXksVpc
{
    "CustomKeyStores": [
    {
      "CustomKeyStoreId": "cks-9876543210fedcba9",
      "CustomKeyStoreName": "ExampleXksVpc",
      "ConnectionState": "CONNECTED",
      "CreationDate": "2022-12-13T18:34:10.675000+00:00",
      "CustomKeyStoreType": "EXTERNAL_KEY_STORE",
      "XksProxyConfiguration": {
        "AccessKeyId": "ABCDE98765432EXAMPLE",
        "Connectivity": "VPC_ENDPOINT_SERVICE",
        "UriEndpoint": "https://example-proxy-uri-endpoint-vpc",
        "UriPath": "/example/prefix/kms/xks/v1",
        "VpcEndpointServiceName": "com.amazonaws.vpce.us-east-1.vpce-svc-example"
      }
```

```
    }
  ]
}
```

ConnectionState가 Disconnected면 외부 키 스토어가 연결된 적이 없거나 의도적으로 해당 외부 키 스토어 프록시에서 연결 해제된 것입니다. 하지만 연결된 외부 키 스토어에서 KMS 키를 사용하려는 시도가 실패하는 것은 외부 키 스토어 프록시 또는 다른 외부 구성 요소에 문제가 있음을 의미할 수 있습니다.

외부 키 스토어의 ConnectionState가 FAILED면 DescribeCustomKeyStores 응답에 오류 원인을 설명하는 ConnectionErrorCode 요소가 포함됩니다.

예를 들어 다음 출력에서 XKS_PROXY_TIMED_OUT 값은가 외부 키 스토어 프록시에 연결할 AWS KMS 수 있음을 나타내지만 외부 키 스토어 프록시가 할당된 시간 AWS KMS 내에에 응답하지 않아 연결이 실패했습니다. 이 연결 오류 코드가 반복적으로 표시되면 외부 키 스토어 프록시 공급업체에 알립니다. 이를 비롯해 기타 연결 오류 문제에 대한 도움말은 외부 키 스토어 문제 해결 단원을 참조하십시오.

```
$ aws kms describe-custom-key-stores --custom-key-store-name ExampleXksVpc
{
    "CustomKeyStores": [
    {
      "CustomKeyStoreId": "cks-9876543210fedcba9",
      "CustomKeyStoreName": "ExampleXksVpc",
      "ConnectionState": "FAILED",
      "ConnectionErrorCode": "XKS_PROXY_TIMED_OUT",
      "CreationDate": "2022-12-13T18:34:10.675000+00:00",
      "CustomKeyStoreType": "EXTERNAL_KEY_STORE",
      "XksProxyConfiguration": {
        "AccessKeyId": "ABCDE98765432EXAMPLE",
        "Connectivity": "VPC_ENDPOINT_SERVICE",
        "UriEndpoint": "https://example-proxy-uri-endpoint-vpc",
        "UriPath": "/example/prefix/kms/xks/v1",
        "VpcEndpointServiceName": "com.amazonaws.vpce.us-east-1.vpce-svc-example"
      }
    }
  ]
}
```

외부 키 저장소 모니터링

AWS KMS 는 외부 키 스토어와의 각 상호 작용에 대한 지표를 수집하여 CloudWatch 계정에 게시합니다. 이러한 지표는 각 외부 키 스토어에 대한 세부 정보 페이지의 모니터링 섹션에서 그래프를 생성하는 데 사용됩니다. 다음 주제에서는 그래프를 사용하여 외부 키 스토어에 영향을 미치는 운영 및 구성 문제를 식별하고 해결하는 방법을 자세히 설명합니다. CloudWatch 지표를 사용하여 외부 키 스토어가 예상대로 작동하지 않을 때 알려주는 경보를 설정하는 것이 좋습니다. 자세한 내용은 Amazon CloudWatch를 사용한 모니터링을 참조하세요.

주제

- 그래프 보기
- 그래프 해석

그래프 보기

다양한 수준의 세부 정보에서 그래프를 볼 수 있습니다. 기본적으로 각 그래프는 3시간의 시간 범위와 5분의 집계 기간을 사용합니다. 콘솔 내에서 그래프 보기를 조정할 수 있지만 외부 키 스토어 세부 정보 페이지를 닫거나 브라우저를 새로 고치면 변경 사항이 기본 설정으로 되돌아갑니다. Amazon CloudWatch 용어에 대한 도움말은 Amazon CloudWatch 개념을 참조하세요.

데이터 포인트 세부 정보 보기

각 그래프의 데이터는 AWS KMS 지표에 의해 수집됩니다. 특정 데이터 포인트에 대한 자세한 정보를 보려면 선 그래프의 데이터 포인트 위에 마우스를 놓습니다. 그러면 그래프가 파생된 지표에 대한 자세한 정보가 포함된 팝업이 표시됩니다. 각 목록 항목은 해당 데이터 포인트에 기록된 차원 값을 표시합니다. 해당 데이터 포인트의 차원 값에 사용할 수 있는 지표 데이터가 없는 경우 팝업에 null 값(–)이 표시됩니다. 일부 그래프는 단일 데이터 포인트에 대해 여러 차원과 값을 기록합니다. 신뢰성 그래프 등의 다른 그래프는 지표에서 수집한 데이터를 사용하여 고유한 값을 계산합니다. 각 목록 항목은 서로 다른 선 그래프 색과 연결됩니다.

시간 범위 수정

시간 범위를 수정하려면 모니터링 섹션의 오른쪽 상단에서 미리 정의된 시간 범위 중 하나를 선택합니다. 미리 정의된 시간 범위는 1시간~1주(1h(1시간), 3h(3시간), 12h(12시간), 1d(1일), 3d(3일) 또는 1w(1주))입니다. 이렇게 하면 모든 그래프의 시간 범위가 조정됩니다. 하나의 특정 그래프를 다른 시간 범위에서 보거나 사용자 지정 시간 범위를 설정하려면 그래프를 확대하거나 Amazon CloudWatch 콘솔에서 봅니다.

그래프 확대

미니 맵 확대/축소 기능을 사용하여 확대/축소 보기를 변경하지 않고도 선 그래프와 그래프의 누적된 부분의 섹션에 초점을 맞출 수 있습니다. 예를 들어, 미니 맵 확대/축소 기능을 사용하여 선 그래프의 최고점에 초점을 맞출 수 있으므로 동일한 타임라인에서 모니터링 섹션의 다른 그래프와 스파이크를 비교할 수 있습니다.

1. 초점을 맞추려는 그래프 영역을 선택하여 드래그한 다음 마우스 버튼을 놓습니다.
2. 확대/축소를 재설정하려면 안에 빼기(-) 기호가 있는 돋보기처럼 보이는 확대/축소 재설정 아이콘을 선택합니다.

그래프 확대

그래프를 확대하려면 개별 그래프의 오른쪽 상단에서 메뉴 아이콘을 선택하고 Enlarge(확대)를 선택합니다. 그래프 위로 마우스를 가져갈 때 메뉴 아이콘 옆에 나타나는 확대 아이콘을 선택할 수도 있습니다.

그래프를 확대하면 다른 기간, 사용자 지정 시간 범위 또는 새로 고침 간격을 지정하여 그래프 보기를 추가로 수정할 수 있습니다. 이러한 변경 사항은 확대 보기를 닫을 때 기본 설정으로 되돌아갑니다.

기간 수정

1. Period options(기간 옵션) 메뉴를 선택합니다. 기본적으로 이 메뉴에는 값이 5 minutes(5분)로 표시됩니다.
2. 기간을 선택합니다. 미리 정의된 기간은 1초~30일입니다.

   예를 들어, 문제 해결 시 유용할 수 있는 1분 보기를 선택할 수 있습니다. 또는 세부적이지 않은 1시간 보기를 선택할 수 있습니다. 이 보기는 시간 경과에 따른 추세를 볼 수 있도록 넓은 시간 범위(예: 3일)를 볼 때 유용할 수 있습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 기간을 참조하세요.

시간 범위 또는 시간대 수정

1. 1시간~1주(1h(1시간), 3h(3시간), 12h(12시간), 1d(1일), 3d(3일) 또는 1w(1주))의 미리 정의된 시간 범위 중 하나를 선택합니다. 또는 사용자 지정(Custom)을 선택하여 나만의 시간 범위를 설정할 수 있습니다.
2. Custom(사용자 지정)을 선택합니다.

   a. 시간 범위: 상자의 왼쪽 상단에서 Absolute(절대) 탭을 선택합니다. 캘린더 선택기나 텍스트 필드 상자를 사용하여 시간 범위를 지정합니다.

b. 시간대: 상자의 오른쪽 상단에서 드롭다운을 선택합니다. 시간대를 UTC 또는 현지 시간대(Local time zone)로 변경할 수 있습니다.

3. 시간 범위를 지정한 후 적용(Apply)을 선택합니다.

그래프의 데이터 새로 고침 빈도 수정

1. 오른쪽 상단에서 Refresh options(새로 고침 옵션) 메뉴를 선택합니다.
2. 새로 고침 간격(Off(끄기), 10 Seconds(10초), 1 Minute(1분), 2 Minutes(2분), 5 Minutes(5분) 또는 15 Minutes(15분))을 선택합니다.

Amazon CloudWatch 콘솔에서 그래프 보기

모니터링 섹션의 그래프는 AWS KMS 가 Amazon CloudWatch에 게시하는 사전 정의된 지표에서 파생됩니다. CloudWatch 콘솔에서 그래프를 열고 CloudWatch 대시보드에 저장할 수 있습니다. 외부 키 스토어가 여러 개 있는 경우 CloudWatch에서 해당 그래프를 열고 단일 대시보드에 저장하여 상태와 사용량을 비교할 수 있습니다.

CloudWatch 대시보드에 추가

오른쪽 상단에서 Add to dashboard(대시보드에 추가)를 선택하여 모든 그래프를 Amazon CloudWatch 대시보드에 추가합니다. 기존 대시보드를 선택하거나 새로 생성할 수 있습니다. 이 대시보드를 사용하여 그래프와 경보의 사용자 지정 보기를 생성하는 방법에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서의 Amazon CloudWatch 대시보드 사용을 참조하세요.

CloudWatch에서 지표 보기

개별 그래프의 오른쪽 상단에 있는 메뉴 아이콘을 선택하고 View in metrics(지표에서 보기)를 선택하여 Amazon CloudWatch 콘솔에서 이 그래프를 봅니다. CloudWatch 콘솔에서 이 단일 그래프를 대시보드에 추가하고 시간 범위, 기간 및 새로 고침 간격을 수정할 수 있습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 지표 그래프 작성을 참조하세요.

그래프 해석

AWS KMS 는 AWS KMS 콘솔 내 외부 키 스토어의 상태를 모니터링하기 위한 여러 그래프를 제공합니다. 이러한 그래프는 자동으로 구성되고 AWS KMS 지표에서 파생됩니다.

그래프 데이터는 외부 키 스토어와 외부 키에 대한 호출의 일부로 수집됩니다. 호출하지 않은 시간 범위 동안 데이터 채우기 그래프가 표시될 수 있습니다.이 데이터는가 외부 키 스토어 프록시 및 외부 키 관리자의 상태를 확인하기 위해 사용자를 대신하여 AWS KMS 수행하는 정기 `GetHealthStatus` 호

출에서 가져옵니다. 그래프에 No data available(사용 가능한 데이터 없음) 메시지가 표시되면 해당 시간 범위 동안 호출이 기록되지 않았거나 외부 키 스토어가 `DISCONNECTED` 상태입니다. 더 넓은 시간 범위로 보기를 조정하여 외부 키 스토어가 연결 해제된 시간을 식별할 수 있습니다.

주제

- 전체 요청
- 신뢰성
- Latency
- 상위 5개 예외
- 인증서 만료 날짜

전체 요청

지정된 시간 범위 동안 특정 외부 키 스토어에 대해 수신되는 총 AWS KMS 요청 수입니다. 이 그래프를 사용하여 제한이 발생할 위험이 있는지 확인합니다.

AWS KMS 에서는 외부 키 관리자가 초당 최대 1,800개의 암호화 작업 요청을 처리할 수 있도록 권장합니다. 5분 동안 호출이 540,000건에 가까워지면 제한이 발생할 위험이 있습니다.

ExternalKeyStoreThrottle 지표로 AWS KMS 제한되는 외부 키 스토어의 KMS 키에 대한 암호화 작업 요청 수를 모니터링할 수 있습니다.

'매우 높은 요청 빈도로 인해' 요청이 거부되었음을 설명하는 메시지와 함께 `KMSInvalidStateException` 오류가 매우 자주 발생하는 경우 외부 키 관리자 또는 외부 키 스토어 프록시가 현재 요청 빈도를 따라갈 수 없는 것일 수 있습니다. 가능하면 요청 빈도를 낮춥니다. 사용자 지정 키 스토어 요청 할당량 값의 감소를 요청하는 것도 고려할 수 있습니다. 이 할당량 값을 줄이면 제한이 증가할 수 있지만 AWS KMS 가 초과 요청이 외부 키 스토어 프록시 또는 외부 키 관리자로 전송되기 전에 신속하게 거부함을 나타냅니다. 할당량 감소를 요청하기 위해서는 AWS Support 센터를 방문하여 케이스를 생성하세요.

총 요청 그래프는 AWS KMS 가 외부 키 스토어 프록시에서 수신하는 성공 및 실패 응답 모두에 대한 데이터를 수집하는 XksProxyErrors 지표에서 파생됩니다. 특정 데이터 포인트를 보면 팝업에 해당 데이터 포인트에 기록된 총 AWS KMS 요청 수와 함께 `CustomKeyStoreId`차원 값이 표시됩니다. `CustomKeyStoreId`는 항상 같습니다.

신뢰성

외부 키 스토어 프록시가 성공한 응답 또는 재시도할 수 없는 오류를 반환한 AWS KMS 요청의 백분율입니다. 이 그래프를 사용하여 외부 키 스토어 프록시의 운영 상태를 평가합니다.

그래프에 100% 미만의 값이 표시되면 프록시가 응답하지 않았거나 재시도할 수 있는 오류로 응답한 것입니다. 이는 네트워크 문제, 외부 키 스토어 프록시 또는 외부 키 관리자의 속도 저하 또는 구현 버그를 나타낼 수 있습니다.

요청에 잘못된 자격 증명이 포함되어 있고 프록시가 `AuthenticationFailedException`으로 응답하는 경우 프록시가 외부 키 스토어 프록시 API 요청에서 잘못된 값을 식별하여 실패가 예상되기 때문에 그래프는 여전히 100% 신뢰성을 나타냅니다. 신뢰성 그래프의 백분율이 100%이면 외부 키 스토어 프록시가 예상대로 응답하고 있는 것입니다. 그래프에 100% 미만의 값이 표시되면 프록시가 재시도할 수 있는 오류로 응답했거나 시간이 초과된 것입니다. 예를 들어 프록시가 매우 높은 요청 빈도로 인해 `ThrottlingException`으로 응답하는 경우 프록시가 실패를 유발한 요청의 특정 문제를 식별할 수 없기 때문에 그래프에 낮은 신뢰성(백분율)이 표시됩니다. 이는 재시도할 수 있는 오류는 요청을 재시도하여 해결할 수 있는 일시적인 문제일 가능성이 높기 때문입니다.

다음 오류 응답은 신뢰성(백분율)을 낮춥니다. 상위 5개 예외 그래프와 XksProxyErrors 지표를 사용하여 프록시가 각 재시도할 수 있는 오류를 반환하는 빈도를 추가로 모니터링할 수 있습니다.

- `InternalException`
- `DependencyTimeoutException`
- `ThrottlingException`
- `XksProxyUnreachableException`

신뢰성 그래프는 외부 키 스토어 프록시에서 XksProxyErrors가 AWS KMS 수신하는 성공 및 실패 응답 모두에 대한 데이터를 수집하는 지표에서 파생됩니다. 응답의 `ErrorType` 값이 `Retryable`인 경우에만 신뢰성(백분율)이 낮아집니다. 특정 데이터 포인트를 보면 팝업에 해당 데이터 포인트에 기록된 AWS KMS 요청에 대한 신뢰성 백분율과 함께 `CustomKeyStoreId`차원 값이 표시됩니다. `CustomKeyStoreId`는 항상 같습니다.

XksProxyErrors 지표를 사용하여 1분 동안 재시도할 수 있는 오류가 5개 이상 기록되면 알림을 통해 잠재적인 네트워킹 문제를 통지하는 CloudWatch 경보를 생성하는 것이 좋습니다. 자세한 내용은 재시도 가능한 오류에 대한 경보 생성 단원을 참조하십시오.

Latency

외부 키 스토어 프록시가 AWS KMS 요청에 응답하는 데 걸리는 밀리초 수입니다. 이 그래프를 사용하여 외부 키 스토어 프록시와 외부 키 관리자의 성능을 평가합니다.

AWS KMS 는 외부 키 스토어 프록시가 250밀리초 이내에 각 요청에 응답할 것으로 예상합니다. 네트워크 시간 초과의 경우 AWS KMS 는 요청을 한 번 재시도합니다. 프록시가 두 번째로 실패하는 경우

기록된 지연 시간은 두 요청 시도에 대한 제한 시간을 합산한 값이며 그래프에는 약 500밀리초가 표시됩니다. 프록시가 250밀리초 제한 시간 내에 응답하지 않는 다른 모든 경우에 기록된 지연 시간은 250밀리초입니다. 프록시의 암호화 및 복호화 작업 시간이 자주 초과되는 경우 외부 프록시 관리자에게 문의하세요. 지연 시간 문제를 해결하는 방법에 대한 도움말은 지연 시간 및 제한 시간 오류 섹션을 참조하세요.

느린 응답은 외부 키 관리자가 현재 요청 트래픽을 처리할 수 없음을 나타낼 수도 있습니다. AWS KMS는 외부 키 관리자가 초당 최대 1,800개의 암호화 작업 요청을 처리할 수 있도록 권장합니다. 외부 키 관리자가 초당 1,800개의 요청을 처리할 수 없는 경우 사용자 지정 키 스토어에서 KMS 키에 대한 요청 할당량 감소를 요청하는 것이 좋습니다. 외부 키 스토어에서 KMS 키를 사용하는 암호화 작업에 대한 요청은 외부 키 스토어 프록시 또는 외부 키 관리자에 의해 처리되고 나중에 거부되는 대신 제한 예외와 함께 빠르게 실패합니다.

지연 시간 그래프는 XksProxyLatency 지표에서 파생됩니다. 특정 데이터 포인트를 볼 때 팝업에는 해당 데이터 포인트의 작업에 대해 기록된 평균 지연 시간과 함께 해당 `KmsOperation` 및 `XksOperation` 차원 값이 표시됩니다. 목록 항목은 지연 시간이 가장 긴 것부터 가장 짧은 것 순으로 정렬됩니다.

XksProxyLatency 지표를 사용하여 지연 시간이 제한 시간에 도달하면 알려주는 CloudWatch 경보를 생성하는 것이 좋습니다. 자세한 내용은 응답 제한 시간에 대한 경보 생성 단원을 참조하십시오.

상위 5개 예외

지정된 시간 범위 동안 실패한 암호화 및 관리 작업에 대한 상위 5개 예외입니다. 이 그래프를 사용하여 가장 자주 발생하는 오류를 추적하면 엔지니어링 작업의 우선순위를 정할 수 있습니다.

이 수에는 외부 키 스토어 프록시에서 AWS KMS 수신한 예외와 외부 키 스토어 프록시와 통신을 설정할 수 없을 때 내부적으로 AWS KMS 반환`XksProxyUnreachableException`하는가 포함됩니다.

재시도할 수 있는 오류의 비율이 높으면 네트워킹 오류가 발생한 것일 수 있고, 재시도할 수 없는 오류의 비율이 높으면 외부 키 스토어 구성에 문제가 있는 것일 수 있습니다. 예를 들어 의 스파이크는에 구성된 인증 자격 증명 AWS KMS 과 외부 키 스토어 프록시 간의 불일치를 `AuthenticationFailedExceptions` 나타냅니다. 외부 키 스토어 구성을 보려면 외부 키 저장소 보기 섹션을 참조하세요. 외부 키 스토어 설정을 편집하려면 외부 키 저장소 속성 편집 섹션을 참조하세요.

가 외부 키 스토어 프록시에서 AWS KMS 수신하는 예외는 작업이 실패할 때가 AWS KMS 반환하는 예외와 다릅니다. AWS KMS 암호화 작업은 외부 키 스토어의 외부 구성 또는 연결 상태와 관련된 모든 실패에 `KMSInvalidStateException` 대해를 반환합니다. 문제를 식별하려면 함께 제공되는 오류 메시지 텍스트를 사용합니다.

다음 표에는 상위 5개 예외 그래프에 나타날 수 있는 예외와가 AWS KMS 반환하는 해당 예외가 나와 있습니다.

오류 유형	그래프에 표시된 예외	에서 AWS KMS 반환한 예외
재시도 불가능	**AccessDeniedException** 문제 해결에 대한 도움말은 프록시 권한 부여 문제 섹션을 참조하세요.	`CreateKey` 작업에 대한 응답으로 **CustomKeyStoreInvalidStateException** 암호화 작업에 대한 응답으로 **KMSInvalidStateException**
재시도 불가능	**AuthenticationFailedException** 문제 해결에 대한 도움말은 인증 자격 증명 오류 섹션을 참조하세요.	`CreateCustomKeyStore` 및 `UpdateCustomKeyStore` 작업에 대한 응답으로 **XksProxyIncorrectAuthenticationCredentialException** `CreateKey` 작업에 대한 응답으로 **CustomKeyStoreInvalidStateException** 암호화 작업에 대한 응답으로 **KMSInvalidStateException**
재시도 가능	**DependencyTimeoutException** 문제 해결에 대한 도움말은 지연 시간 및 제한 시간 오류 섹션을 참조하세요.	`CreateCustomKeyStore` 및 `UpdateCustomKeyStore` 작업에 대한 응답으로 **XksProxyUriUnreachableException** `CreateKey` 작업에 대한 응답으로 **CustomKey**

오류 유형	그래프에 표시된 예외	에서 AWS KMS 반환한 예외
		StoreInvalidStateException 암호화 작업에 대한 응답으로 **KMSInvalidStateException**
재시도 가능	**InternalException** 외부 키 스토어 프록시가 외부 키 관리자와 통신할 수 없기 때문에 요청을 거부했습니다. 외부 키 스토어 프록시 구성이 올바른지, 외부 키 관리자를 사용할 수 있는지 확인합니다.	CreateCustomKeyStore 및 UpdateCustomKeyStore 작업에 대한 응답으로 **XksProxyInvalidResponseException** CreateKey 작업에 대한 응답으로 **CustomKeyStoreInvalidStateException** 암호화 작업에 대한 응답으로 **KMSInvalidStateException**
재시도 불가능	**InvalidCiphertextException** 문제 해결에 대한 도움말은 복호화 오류 섹션을 참조하세요.	암호화 작업에 대한 응답으로 **KMSInvalidStateException**
재시도 불가능	**InvalidKeyUsageException** 문제 해결에 대한 도움말은 외부 키에 대한 암호화 작업 오류 섹션을 참조하세요.	CreateKey 작업에 대한 응답으로 **XksKeyInvalidConfigurationException** 암호화 작업에 대한 응답으로 **KMSInvalidStateException**

오류 유형	그래프에 표시된 예외	에서 AWS KMS 반환한 예외
재시도 불가능	**InvalidStateException** 문제 해결에 대한 도움말은 외부 키에 대한 암호화 작업 오류 섹션을 참조하세요.	CreateKey 작업에 대한 응답으로 **XksKeyInvalidConfigurationException** 암호화 작업에 대한 응답으로 **KMSInvalidStateException**
재시도 불가능	**InvalidUriPathException** 문제 해결에 대한 도움말은 일반 구성 오류 섹션을 참조하세요.	CreateCustomKeyStore 및 UpdateCustomKeyStore 작업에 대한 응답으로 **XksProxyInvalidConfigurationException** CreateKey 작업에 대한 응답으로 **CustomKeyStoreInvalidStateException** 암호화 작업에 대한 응답으로 **KMSInvalidStateException**
재시도 불가능	**KeyNotFoundException** 문제 해결에 대한 도움말은 외부 키 오류 섹션을 참조하세요.	CreateKey 작업에 대한 응답으로 **XksKeyNotFoundException** 암호화 작업에 대한 응답으로 **KMSInvalidStateException**

오류 유형	그래프에 표시된 예외	에서 AWS KMS 반환한 예외
재시도 가능	**ThrottlingException** 매우 높은 요청 빈도로 인해 외부 키 스토어 프록시가 요청을 거부했습니다. 이 외부 키 스토어에서 KMS 키를 사용하여 호출 빈도를 줄입니다.	CreateCustomKeyStore 및 UpdateCustomKeyStore 작업에 대한 응답으로 **XksProxyUriUnreachableException** CreateKey 작업에 대한 응답으로 **CustomKeyStoreInvalidStateException** 암호화 작업에 대한 응답으로 **KMSInvalidStateException**
재시도 불가능	**UnsupportedOperationException** 문제 해결에 대한 도움말은 외부 키에 대한 암호화 작업 오류 섹션을 참조하세요.	CreateKey 작업에 대한 응답으로 **XksKeyInvalidResponseException** 암호화 작업에 대한 응답으로 **KMSInvalidStateException**

오류 유형	그래프에 표시된 예외	에서 AWS KMS 반환한 예외
재시도 불가능	**ValidationException** 문제 해결에 대한 도움말은 프록시 문제 섹션을 참조하세요.	CreateCustomKeyStore 및 UpdateCustomKeyStore 작업에 대한 응답으로 **XksProxyInvalidResponseException** CreateKey 작업에 대한 응답으로 **CustomKeyStoreInvalidStateException** 암호화 작업에 대한 응답으로 **KMSInvalidStateException**
재시도 가능	**XksProxyUnreachableException** 이 오류가 반복적으로 표시되는 경우 외부 키 스토어 프록시가 활성 상태이고 네트워크에 연결되어 있는지, 해당 URI 경로와 엔드포인트 URI 또는 VPC 서비스 이름이 외부 키 스토어에서 올바른지 확인합니다.	CreateCustomKeyStore 및 UpdateCustomKeyStore 작업에 대한 응답으로 **XksProxyUriUnreachableException** CreateKey 작업에 대한 응답으로 **CustomKeyStoreInvalidStateException** 암호화 작업에 대한 응답으로 **KMSInvalidStateException**

상위 5개 예외 그래프는 XksProxyErrors 지표에서 파생됩니다. 특정 데이터 포인트를 볼 때 팝업에는 해당 데이터 포인트에서 예외가 기록된 횟수와 함께 ExceptionName 차원의 값이 표시됩니다. 5개의 목록 항목은 가장 빈번한 예외에서 가장 덜 빈번한 예외 순으로 정렬됩니다.

XksProxyErrors 지표를 사용하여 1분 동안 재시도할 수 없는 오류가 5개 이상 기록되면 알림을 통해 잠재적인 구성 문제를 통지하는 CloudWatch 경보를 생성하는 것이 좋습니다. 자세한 내용은 재시도 불가능한 오류에 대한 경보 생성 단원을 참조하십시오.

인증서 만료 날짜

외부 키 스토어 프록시 엔드포인트(`XksProxyUriEndpoint`)에 대한 TLS 인증서가 만료될 때까지의 일수입니다. 이 그래프를 사용하여 TLS 인증서의 예정된 만료를 모니터링합니다.

인증서가 만료되면는 외부 키 스토어 프록시와 통신할 AWS KMS 수 없습니다. 외부 키 스토어의 KMS 키로 보호되는 모든 데이터는 인증서를 갱신할 때까지 액세스할 수 없습니다.

인증서 만료 날짜 그래프는 XksProxyCertificateDaysToExpire 지표에서 파생됩니다. 예정된 만료를 알리는 CloudWatch 경보를 생성하려면 이 지표를 사용하는 것이 좋습니다. 인증서 만료로 인해 암호화된 리소스에 액세스하지 못할 수 있습니다. 인증서가 만료되기 전에 인증서를 갱신할 시간을 조직에 제공하도록 경보를 설정합니다. 자세한 내용은 인증서 만료에 대한 경보 생성 단원을 참조하십시오.

외부 키 저장소 연결 및 연결 해제

새로운 외부 키 스토어는 연결되지 않습니다. 외부 키 스토어 AWS KMS keys 에서를 생성하고 사용하려면 외부 키 스토어 프록시에 외부 키 스토어를 연결해야 합니다. 외부 키 스토어를 언제든 연결 및 연결 해제하고 연결 상태를 확인할 수 있습니다.

외부 키 스토어의 연결이 끊긴 동안에는가 외부 키 스토어 프록시와 통신할 수 AWS KMS 없습니다. 따라서 외부 키 스토어와 기존 KMS 키를 보고 관리할 수 있습니다. 그러나 외부 키 스토어에서 KMS 키를 생성하거나 암호화 작업에 KMS 키를 사용할 수는 없습니다. 속성을 편집할 때와 같이 특정 시점에 외부 키 스토어를 연결 해제해야 할 수 있지만 그에 따라 계획해야 합니다. 키 스토어를 연결 해제하면 KMS 키를 사용하는 AWS 서비스의 작동이 중단될 수 있습니다.

외부 키 스토어를 연결할 필요는 없습니다. 외부 키 스토어를 연결 해제 상태로 무기한 남겨두고 사용 시에만 이를 연결할 수 있습니다. 하지만 설정이 올바른지, 연결이 가능한지 확인하기 위해 정기적으로 연결을 테스트하고 싶을 수 있습니다.

사용자 지정 키 스토어를 연결 해제하면 키 스토어의 KMS 키를 즉시 사용할 수 없게 됩니다(최종 일관성에 따라 다름). 그러나 KMS 키로 보호되는 데이터 키로 암호화된 리소스는 데이터 키를 복호화하는 등 KMS 키를 다시 사용할 때까지 영향을 받지 않습니다. 이 문제는 리소스를 보호하기 위해 데이터 키를 사용하는 AWS 서비스에 영향을 미칩니다. 자세한 내용은 사용할 수 없는 KMS 키가 데이터 키에 미치는 영향을 참조하세요.

> **Note**
>
> 외부 키 스토어는 키 스토어가 연결되지 않았거나 연결을 직접 해제한 경우에만 `DISCONNECTED` 상태가 됩니다. `CONNECTED` 상태는 외부 키 스토어 또는 해당 지원 구성 요소가 효율적으로 작동하고 있음을 나타내지 않습니다. 외부 키 스토어 구성 요소의 성능에 대한 자세한 내용은 각 외부 키 스토어에 대한 세부 정보 페이지의 Monitoring(모니터링) 섹션에 있는 그래프를 참조하세요. 자세한 내용은 외부 키 저장소 모니터링을 참조하세요.
> 외부 키 관리자는 외부 키 스토어와 AWS KMS 외부 키 스토어 프록시 간 또는 외부 키 스토어 프록시와 외부 키 관리자 간의 통신을 중지하고 다시 시작하는 추가 방법을 제공할 수 있습니다. 자세한 내용은 외부 키 관리자 설명서를 참조하세요.

주제

- 연결 상태
- 외부 키 저장소 연결
- 외부 키 저장소 연결 해제

연결 상태

연결 및 연결 해제는 사용자 지정 키 스토어의 연결 상태를 변경합니다. 연결 상태 값은 AWS CloudHSM 키 스토어와 외부 키 스토어에서 동일합니다.

사용자 지정 키 스토어의 연결 상태를 보려면 DescribeCustomKeyStores 작업 또는 AWS KMS 콘솔을 사용합니다. Connection state(연결 상태)는 각 사용자 지정 키 스토어 테이블, 각 사용자 지정 키 스토어에 대한 세부 정보 페이지의 General configuration(일반 구성) 섹션 및 사용자 지정 키 스토어에 있는 KMS 키의 Cryptographic configuration(암호화 구성) 탭에 표시됩니다. 자세한 내용은 AWS CloudHSM 키 스토어 보기 및 외부 키 저장소 보기 섹션을 참조하세요.

사용자 지정 키 스토어는 다음 연결 상태 중 하나를 가질 수 있습니다.

- `CONNECTED`: 사용자 지정 키 스토어가 백업 키 스토어에 연결되어 있습니다. 사용자 지정 키 스토어에서 KMS 키를 생성하거나 사용할 수 있습니다.

  키 스토어의 백업 AWS CloudHSM 키 스토어는 연결된 AWS CloudHSM 클러스터입니다. 외부 키 스토어의 백업 키 스토어는 외부 키 스토어 프록시와 이 프록시가 지원하는 외부 키 관리자입니다.

  CONNECTED 상태는 연결에 성공했으며 사용자 지정 키 스토어의 연결이 의도적으로 해제되지 않았음을 의미합니다. 그러나 연결이 제대로 작동하고 있음을 나타내지는 않습니다. AWS CloudHSM

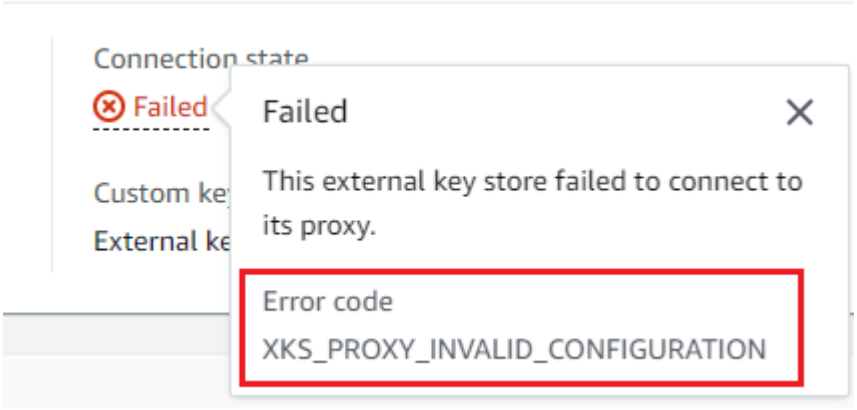
키 스토어와 연결된 AWS CloudHSM 클러스터의 상태에 대한 자세한 내용은 AWS CloudHSM 사용 설명서의에 대한 CloudWatch 지표 가져오기 AWS CloudHSM를 참조하세요. 외부 키 스토어의 상태 및 운영에 대한 자세한 내용은 각 외부 키 스토어에 대한 세부 정보 페이지의 Monitoring(모니터링) 섹션에 있는 그래프를 참조하세요. 자세한 내용은 외부 키 저장소 모니터링을 참조하세요.

- `CONNECTING`: 사용자 지정 키 스토어 연결 작업이 진행 중입니다. 이것은 일시적인 상태입니다.
- `DISCONNECTED`: 사용자 지정 키 스토어가 백업에 연결되지 않았거나 AWS KMS 콘솔 또는 DisconnectCustomKeyStore 작업을 사용하여 의도적으로 연결이 해제되었습니다.
- `DISCONNECTING`: 사용자 지정 키 스토어 연결 해제 작업이 진행 중입니다. 이것은 일시적인 상태입니다.
- `FAILED`: 사용자 지정 키 스토어를 연결하려는 시도가 실패했습니다. DescribeCustomKeyStores 응답의 `ConnectionErrorCode`는 문제를 나타냅니다.

사용자 지정 키 스토어를 연결하려면 연결 상태가 `DISCONNECTED`여야 합니다. 연결 상태가 `FAILED`인 경우 `ConnectionErrorCode`를 사용하여 문제를 식별하고 해결합니다. 사용자 지정 키 스토어를 연결 해제한 후 다시 연결을 시도하세요. 연결 실패에 대한 도움말은 외부 키 스토어 연결 오류 단원을 참조하십시오. 연결 오류 코드에 대응하는 방법에 대한 도움말은 외부 키 스토어의 연결 오류 코드 섹션을 참조하세요.

연결 오류 코드를 보려면 다음을 수행하세요.

- DescribeCustomKeyStores 응답에서 `ConnectionErrorCode` 요소의 값을 확인합니다. 이 요소는 `ConnectionState`가 `FAILED`인 경우에만 `DescribeCustomKeyStores` 응답에 나타납니다.
- AWS KMS 콘솔에서 연결 오류 코드를 보려면 외부 키 스토어의 세부 정보 페이지에서 실패 값 위에 마우스를 올려 놓습니다.



외부 키 저장소 연결

외부 키 스토어가 외부 키 스토어 프록시에 연결되면 외부 키 스토어에 KMS 키를 생성하고 암호화 작업에서 기존 KMS 키를 사용할 수 있습니다.

외부 키 스토어를 외부 키 스토어 프록시에 연결하는 프로세스는 외부 키 스토어의 연결에 따라 다릅니다.

- 퍼블릭 엔드포인트 연결이 있는 외부 키 스토어를 연결하면 AWS KMS 가 GetHealthStatus 요청을 외부 키 스토어 프록시로 전송하여 프록시 URI 엔드포인트, 프록시 URI 경로 및 프록시 인증 자격 증명을 검증합니다. 프록시의 성공적인 응답은 프록시 URI 엔드포인트와 프록시 URI 경로가 정확하고 액세스 가능하며 프록시가 외부 키 스토어에 대한 프록시 인증 자격 증명으로 서명된 요청을 인증했음을 확인합니다.
- VPC 엔드포인트 서비스 연결이 있는 외부 키 스토어를 외부 키 스토어 프록시에 연결하면가 다음을 AWS KMS 수행합니다.
  - 프록시 URI 엔드포인트에 지정된 프라이빗 DNS 이름의 도메인이 검증되었는지 확인합니다.
  - AWS KMS VPC에서 VPC 엔드포인트 서비스로 인터페이스 엔드포인트를 생성합니다.
  - 프록시 URI 엔드포인트에 지정된 프라이빗 DNS 이름에 대한 프라이빗 호스팅 영역을 생성합니다.
  - 외부 키 스토어 프록시로 GetHealthStatus 요청을 전송합니다. 프록시의 성공적인 응답은 프록시 URI 엔드포인트와 프록시 URI 경로가 정확하고 액세스 가능하며 프록시가 외부 키 스토어에 대한 프록시 인증 자격 증명으로 서명된 요청을 인증했음을 확인합니다.

연결 작업은 사용자 지정 키 스토어를 연결하는 프로세스를 시작하지만 외부 키 스토어를 외부 프록시에 연결하는 데 5분가량 걸립니다. 연결 작업의 성공 응답은 외부 키 스토어가 연결되었음을 나타내지 않습니다. 연결이 성공했는지 확인하려면 AWS KMS 콘솔 또는 DescribeCustomKeyStores 작업을 사용하여 키 스토어 외부의 연결 상태를 확인합니다.

연결 상태가 이면 `FAILED`연결 오류 코드가 AWS KMS 콘솔에 표시되고 `DescribeCustomKeyStore` 응답에 추가됩니다. 연결 오류 코드를 해석하는 방법에 대한 도움말은 외부 키 스토어의 연결 오류 코드 섹션을 참조하세요.

외부 키 저장소에 연결 및 다시 연결

AWS KMS 콘솔에서 또는 ConnectCustomKeyStore 작업을 사용하여 외부 키 스토어를 연결하거나 다시 연결할 수 있습니다.

AWS KMS 콘솔 사용

AWS KMS 콘솔을 사용하여 외부 키 스토어 프록시에 외부 키 스토어를 연결할 수 있습니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 탐색 창에서 Custom key stores(사용자 지정 키 스토어), External key stores(외부 키 스토어)를 선택합니다.
4. 연결하려는 외부 키 스토어의 행을 선택합니다.

   외부 키 스토어의 연결 상태가 FAILED(실패)면 연결에 앞서 외부 키 스토어를 연결 해제해야 합니다.
5. Key store actions(키 스토어 작업) 메뉴에서 Connect(연결)를 선택합니다.

연결 프로세스를 완료하는 데 일반적으로 5분가량 걸립니다. 작업이 완료되면 연결 상태가 CONNECTED(연결됨)로 변경됩니다.

연결 상태가 Failed(실패)인 경우 연결 상태 위로 마우스를 가져가면 오류의 원인을 설명하는 연결 오류 코드가 표시됩니다. 연결 오류 코드에 대응하는 방법에 대한 도움말은 외부 키 스토어의 연결 오류 코드 섹션을 참조하세요. 연결 상태가 Failed(실패)인 외부 키 스토어를 연결하려면 먼저 사용자 지정 키 스토어를 연결 해제해야 합니다.

AWS KMS API 사용

연결 해제된 외부 키 스토어를 연결하려면 ConnectCustomKeyStore 작업을 사용합니다.

연결하기 전에 외부 키 스토어의 ???연결 상태가 `DISCONNECTED`여야 합니다. 현재 연결 상태가 `FAILED`면 외부 키 스토어를 연결 해제한 다음 다시 연결합니다.

이 연결 프로세스를 완료하는 데 5분가량 소요됩니다. 빨리 실패하지 않는 한, `ConnectCustomKeyStore`는 속성 없이 HTTP 200 응답과 JSON 객체를 반환합니다. 하지만 이러한 초기 응답은 연결이 성공했음을 의미하지는 않습니다. 외부 키 스토어가 연결되어 있는지 확인하려면 DescribeCustomKeystore 응답에서 연결 상태를 확인합니다.

이 섹션의 예제는 AWS Command Line Interface (AWS CLI)를 사용하지만, 지원되는 모든 프로그래밍 언어를 사용할 수 있습니다.

외부 키 스토어를 식별하려면 사용자 지정 키 스토어 ID를 사용합니다. 콘솔의 Custom key stores(사용자 지정 키 스토어) 페이지에서, 또는 DescribeCustomKeyStores 작업을 사용하여 ID를 찾을 수 있습니다. 이 예제를 실행하기 앞서 예제 ID를 유효한 ID로 바꿉니다.

```
$ aws kms connect-custom-key-store --custom-key-store-id cks-1234567890abcdef0
```

ConnectCustomKeyStore 작업은 응답에 ConnectionState를 반환하지 않습니다. 외부 키 스토어가 연결되어 있는지 확인하려면 DescribeCustomKeyStores 작업을 사용합니다. 기본적으로 이 작업은 계정 및 리전에서 모든 사용자 지정 키 스토어를 반환합니다. 그러나 CustomKeyStoreId 또는 CustomKeyStoreName 파라미터(둘 중 하나만)를 사용해서 특정한 사용자 지정 키 스토어로 응답을 제한할 수 있습니다. ConnectionState 값이 CONNECTED면 외부 키 스토어가 외부 키 스토어 프록시에 연결되어 있는 것입니다.

```
$ aws kms describe-custom-key-stores --custom-key-store-name ExampleXksVpc
{
    "CustomKeyStores": [
    {
      "CustomKeyStoreId": "cks-9876543210fedcba9",
      "CustomKeyStoreName": "ExampleXksVpc",
      "ConnectionState": "CONNECTED",
      "CreationDate": "2022-12-13T18:34:10.675000+00:00",
      "CustomKeyStoreType": "EXTERNAL_KEY_STORE",
      "XksProxyConfiguration": {
        "AccessKeyId": "ABCDE98765432EXAMPLE",
        "Connectivity": "VPC_ENDPOINT_SERVICE",
        "UriEndpoint": "https://example-proxy-uri-endpoint-vpc",
        "UriPath": "/example/prefix/kms/xks/v1",
        "VpcEndpointServiceName": "com.amazonaws.vpce.us-east-1.vpce-svc-example"
      }
    }
  ]
}
```

DescribeCustomKeyStores 응답의 ConnectionState 값이 FAILED면 ConnectionErrorCode 요소는 실패의 원인을 나타냅니다.

다음 예제에서의 XKS_VPC_ENDPOINT_SERVICE_NOT_FOUND 값은가 외부 키 스토어 프록시와 통신하는 데 사용하는 VPC 엔드포인트 서비스를 찾을 AWS KMS 수 없음을 ConnectionErrorCode 나타냅니다. XksProxyVpcEndpointServiceName가 올바른지, AWS KMS 서비스 보안 주체가 Amazon VPC 엔드포인트 서비스에서 허용되는 보안 주체인지, VPC 엔드포인트 서비스가 연결 요청

을 수락할 필요가 없는지 확인합니다. 연결 오류 코드에 대응하는 방법에 대한 도움말은 외부 키 스토어의 연결 오류 코드 섹션을 참조하세요.

```
$ aws kms describe-custom-key-stores --custom-key-store-name ExampleXksVpc
{
    "CustomKeyStores": [
    {
      "CustomKeyStoreId": "cks-9876543210fedcba9",
      "CustomKeyStoreName": "ExampleXksVpc",
      "ConnectionState": "FAILED",
      "ConnectionErrorCode": "XKS_VPC_ENDPOINT_SERVICE_NOT_FOUND",
      "CreationDate": "2022-12-13T18:34:10.675000+00:00",
      "CustomKeyStoreType": "EXTERNAL_KEY_STORE",
      "XksProxyConfiguration": {
        "AccessKeyId": "ABCDE98765432EXAMPLE",
        "Connectivity": "VPC_ENDPOINT_SERVICE",
        "UriEndpoint": "https://example-proxy-uri-endpoint-vpc",
        "UriPath": "/example/prefix/kms/xks/v1",
        "VpcEndpointServiceName": "com.amazonaws.vpce.us-east-1.vpce-svc-example"
      }
    }
  ]
}
```

외부 키 저장소 연결 해제

VPC 엔드포인트 서비스 연결이 있는 외부 키 스토어를 외부 키 스토어 프록시에서 연결 해제하면 AWS KMS 는 VPC 엔드포인트 서비스에 대한 인터페이스 엔드포인트를 삭제하고 연결을 지원하기 위해 생성한 네트워크 인프라를 제거합니다. 퍼블릭 엔드포인트 연결이 있는 외부 키 스토어에는 동등한 프로세스가 필요하지 않습니다. 이 작업은 VPC 엔드포인트 서비스 또는 지원 구성 요소에 영향을 주지 않으며 외부 키 스토어 프록시 또는 외부 구성 요소에도 영향을 주지 않습니다.

외부 키 스토어의 연결이 해제된 동안 AWS KMS 는 외부 키 스토어 프록시에 요청을 보내지 않습니다. 외부 키 스토어의 연결 상태는 DISCONNECTED입니다. 연결 해제된 외부 키 스토어의 KMS 키는 UNAVAILABLE 키 상태(삭제 보류 중이 아닌 경우)이므로 암호화 작업에 사용할 수 없습니다. 그러나 외부 키 스토어와 기존 KMS 키를 계속 보고 관리할 수 있습니다.

연결 해제된 상태는 일시적이고 되돌릴 수 있도록 설계되었습니다. 언제든지 외부 키 스토어를 다시 연결할 수 있습니다. 일반적으로 재구성이 필요하지 않습니다. 그러나 연결 해제된 동안 연결된 외부 키 스토어 프록시의 속성이 변경된 경우(예: 프록시 인증 자격 증명의 교체) 다시 연결하기 전에 외부 키 스토어 설정을 편집해야 합니다.

> **Note**
>
> 사용자 지정 키 스토어의 연결이 해제된 상태에서는 사용자 지정 키 스토어에서 KMS 키를 생성하거나, 암호화 작업을 위해 기존 KMS 키를 사용하려는 모든 시도가 실패합니다. 이 작업은 사용자가 기밀 데이터를 저장하거나 액세스하지 못하도록 차단합니다.

외부 키 스토어의 연결 해제가 미치는 영향을 정확하게 예측하려면 외부 키 스토어에서 KMS 키를 식별하고 과거 사용량을 판단합니다.

다음과 같은 이유로 외부 키 스토어를 연결 해제할 수 있습니다.

- 속성을 편집하기 위해. 외부 키 스토어가 연결되어 있는 동안 사용자 지정 키 스토어 이름, 프록시 URI 경로 및 프록시 인증 자격 증명을 편집할 수 있습니다. 그러나 프록시 연결 유형, 프록시 URI 엔드포인트 또는 VPC 엔드포인트 서비스 이름을 편집하려면 먼저 외부 키 스토어를 연결 해제해야 합니다. 자세한 내용은 외부 키 저장소 속성 편집을 참조하세요.
- AWS KMS 와 외부 키 스토어 프록시 간의 모든 통신을 중지합니다. 엔드포인트 또는 VPC 엔드포인트 서비스를 비활성화하여 AWS KMS 와 프록시 간의 통신을 중지할 수도 있습니다. 또한 외부 키 스토어 프록시 또는 키 관리 소프트웨어는가 프록시와 AWS KMS 통신하지 못하도록 하거나 프록시가 외부 키 관리자에 액세스하지 못하도록 하는 추가 메커니즘을 제공할 수 있습니다.
- 외부 키 스토어에서 모든 KMS 키를 비활성화하기 위해. AWS KMS 콘솔 또는 DisableKey 작업을 사용하여 외부 키 스토어에서 KMS 키를 비활성화하고 다시 활성화할 수 있습니다. 이러한 작업은 빠르게 완료되지만(최종 일관성에 따라 다름) 한 번에 하나의 KMS 키에 대해 작동합니다. 외부 키 스토어를 연결 해제하면 외부 키 스토어에서 모든 KMS 키의 키 상태가 `Unavailable`로 변경되면서 암호화 작업에서 사용이 불가능한 상태가 됩니다.
- 실패한 연결 시도를 복구하려면. 외부 키 스토어를 연결하려는 시도가 실패하면(사용자 지정 키 스토어의 연결 상태가 `FAILED`) 다시 연결을 시도하기에 앞서 외부 키 스토어를 연결 해제해야 합니다.

외부 키 저장소 연결 해제

AWS KMS 콘솔에서 또는 DisconnectCustomKeyStore 작업을 사용하여 외부 키 스토어의 연결을 해제할 수 있습니다.

AWS KMS 콘솔 사용

AWS KMS 콘솔을 사용하여 외부 키 스토어 프록시에 외부 키 스토어를 연결할 수 있습니다. 이 프로세스를 완료하는 데 5분가량 소요됩니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 탐색 창에서 Custom key stores(사용자 지정 키 스토어), External key stores(외부 키 스토어)를 선택합니다.
4. 연결 해제하려는 외부 키 스토어의 행을 선택합니다.
5. Key store actions(키 스토어 작업) 메뉴에서 Disconnect(연결 해제)를 선택합니다.

작업이 완료되면 연결 상태가 DISCONNECTING(연결 해제 중)에서 DISCONNECTED(연결 해제됨)로 변경됩니다. 작업이 실패하면 오류 메시지가 나타나서 문제를 설명하고 이를 수정할 수 있는 방법에 대한 도움말을 제공합니다. 도움이 더 필요한 경우 외부 키 스토어 연결 오류 섹션을 참조하십시오.

AWS KMS API 사용

연결된 외부 키 스토어를 연결 해제하려면 DisconnectCustomKeyStore 작업을 사용합니다. 작업이 성공하면 HTTP 200 응답과 속성이 없는 JSON 객체를 AWS KMS 반환합니다. 이 프로세스를 완료하는 데 5분가량 소요됩니다. 외부 키 스토어의 연결 상태를 확인하려면 DescribeCustomKeyStores 작업을 사용합니다.

이 섹션의 예제는 AWS Command Line Interface (AWS CLI)를 사용하지만, 지원되는 모든 프로그래밍 언어를 사용할 수 있습니다.

이 예제에서는 VPC 엔드포인트 서비스 연결이 있는 외부 키 스토어를 연결 해제합니다. 이 예제를 실행하기 앞서 예제에 나온 사용자 지정 키 스토어 ID를 유효한 ID로 대체합니다.

```
$ aws kms disconnect-custom-key-store --custom-key-store-id cks-1234567890abcdef0
```

외부 키 스토어가 연결 해제되어 있는지 확인하려면 DescribeCustomKeyStores 작업을 사용합니다. 기본적으로 이 작업은 계정 및 리전에서 모든 사용자 지정 키 스토어를 반환합니다. 그러나 `CustomKeyStoreId` 또는 `CustomKeyStoreName` 파라미터(둘 중 하나만)를 사용해서 특정한 사용자 지정 키 스토어로 응답을 제한할 수 있습니다. `ConnectionState` 값이 `DISCONNECTED`면 외부 키 스토어가 이 예제 외부 키 스토어 프록시에 더 이상 연결되어 있지 않은 것입니다.

```
$ aws kms describe-custom-key-stores --custom-key-store-name ExampleXksVpc
{
    "CustomKeyStores": [
    {
```

```
      "CustomKeyStoreId": "cks-9876543210fedcba9",
      "CustomKeyStoreName": "ExampleXksVpc",
      "ConnectionState": "DISCONNECTED",
      "CreationDate": "2022-12-13T18:34:10.675000+00:00",
      "CustomKeyStoreType": "EXTERNAL_KEY_STORE",
      "XksProxyConfiguration": {
        "AccessKeyId": "ABCDE98765432EXAMPLE",
        "Connectivity": "VPC_ENDPOINT_SERVICE",
        "UriEndpoint": "https://example-proxy-uri-endpoint-vpc",
        "UriPath": "/example/prefix/kms/xks/v1",
        "VpcEndpointServiceName": "com.amazonaws.vpce.us-east-1.vpce-svc-example"
      }
    }
  ]
}
```

외부 키 저장소 삭제

외부 키 스토어를 삭제하면는 외부 키 스토어 프록시에 대한 정보를 AWS KMS포함하여 외부 키 스토어에 대한 모든 메타데이터를 AWS KMS 삭제합니다. 이 작업은 외부 키 스토어 프록시, 외부 키 관리자, 외부 키 또는 Amazon VPC 또는 VPC 엔드포인트 서비스와 같이 외부 키 스토어를 지원하기 위해 생성한 AWS 리소스에는 영향을 주지 않습니다.

외부 키 스토어를 삭제하기 전에 키 스토어에서 모든 KMS 키를 삭제하고 외부 키 스토어 프록시에서 키 스토어를 연결 해제해야 합니다. 그렇지 않으면 키 스토어를 삭제하려는 시도가 실패합니다.

외부 키 스토어 삭제는 되돌릴 수 없지만 새 외부 키 스토어를 생성하여 동일한 외부 키 스토어 프록시 및 외부 키 관리자와 연결할 수 있습니다. 그러나 동일한 외부 키 구성 요소에 액세스할 수 있더라도 외부 키 스토어에서 대칭 암호화 KMS 키를 다시 생성할 수 없습니다.는 각 KMS 키에 고유한 대칭 사이퍼텍스트에 메타데이터를 AWS KMS 포함합니다. 이 보안 기능은 데이터를 암호화한 KMS 키만 해당 데이터를 복호화할 수 있도록 합니다.

외부 키 스토어를 삭제하는 대신 연결 해제하는 것을 고려해 보세요. 외부 키 스토어의 연결이 끊긴 동안 외부 키 스토어와 해당를 관리할 수 AWS KMS keys 있지만 외부 키 스토어에서 KMS 키를 생성하거나 사용할 수는 없습니다. 언제든지 외부 키 스토어를 다시 연결하고 KMS 키를 사용하여 데이터를 암호화 및 복호화를 재개할 수 있습니다. 연결 해제된 외부 키 스토어 프록시 또는 사용할 수 없는 KMS 키에 대한 비용은 없습니다.

AWS KMS 콘솔에서 또는 DeleteCustomKeyStore 작업을 사용하여 외부 키 스토어를 삭제할 수 있습니다.

AWS KMS 콘솔 사용

AWS KMS 콘솔을 사용하여 외부 키 스토어를 삭제할 수 있습니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kms AWS Key Management Service (AWS KMS) 콘솔을 엽니다.
2. 를 변경하려면 페이지 오른쪽 상단에 있는 리전 선택기를 AWS 리전사용합니다.
3. 탐색 창에서 Custom key stores(사용자 지정 키 스토어), External key stores(외부 키 스토어)를 선택합니다.
4. 삭제하려는 외부 키 스토어를 나타내는 열을 찾습니다. 외부 키 스토어의 Connection state(연결 상태)가 DISCONNECTED(연결 해제됨)가 아닌 경우 외부 키 스토어를 연결 해제한 다음 삭제해야 합니다.
5. Key store actions(키 스토어 작업) 메뉴에서 Delete(삭제)를 선택합니다.

작업이 완료되면 성공 메시지가 나타나고, 외부 키 스토어는 더 이상 외부 키 스토어 목록에 나타나지 않습니다. 작업이 실패하면 오류 메시지가 나타나서 문제를 설명하고 이를 수정할 수 있는 방법에 대한 도움말을 제공합니다. 도움이 더 필요한 경우 외부 키 스토어 문제 해결 섹션을 참조하십시오.

AWS KMS API 사용

외부 키 스토어를 삭제하려면 DeleteCustomKeyStore 작업을 사용합니다. 작업이 성공하면 HTTP 200 응답과 속성이 없는 JSON 객체를 AWS KMS 반환합니다.

시작하려면 외부 키 스토어를 연결 해제합니다. 이 명령을 실행하기 앞서 예시에 나온 사용자 지정 키 저장소 ID를 유효한 ID로 대체합니다.

```
$ aws kms disconnect-custom-key-store --custom-key-store-id cks-1234567890abcdef0
```

외부 키 스토어가 연결 해제된 후에는 DeleteCustomKeyStore 작업을 사용하여 이를 삭제할 수 있습니다.

```
$ aws kms delete-custom-key-store --custom-key-store-id cks-1234567890abcdef0
```

외부 키 스토어가 삭제되었는지 확인하려면 DescribeCustomKeyStores 작업을 사용합니다.

```
$ aws kms describe-custom-key-stores

{
```

```
    "CustomKeyStores": []
}
```

더 이상 존재하지 않는 사용자 지정 키 스토어 이름 또는 ID를 지정하면가 `CustomKeyStoreNotFoundException` 예외를 AWS KMS 반환합니다.

```
$ aws kms describe-custom-key-stores --custom-key-store-id cks-1234567890abcdef0

An error occurred (CustomKeyStoreNotFoundException) when calling the
 DescribeCustomKeyStore operation:
```

외부 키 스토어 문제 해결

외부 키 스토어와 관련된 대부분의 문제에 대한 해결 방법은 각 예외와 함께 AWS KMS 표시되는 오류 메시지 또는 외부 키 스토어를 외부 키 스토어 프록시에 연결하려는 시도가 실패할 때 AWS KMS 가 반환하는 연결 오류 코드로 표시됩니다. 그러나 몇몇 문제는 조금 더 복잡합니다.

외부 키 스토어 문제를 진단할 때는 먼저 원인을 찾습니다. 이렇게 하면 해결 방법의 범위가 좁아지고 문제 해결이 보다 효율적으로 이루어집니다.

- AWS KMS - 외부 키 스토어 구성의 잘못된 값과 AWS KMS같은 문제가 내에 있을 수 있습니다.
- 외부 - 외부 키 스토어 프록시 AWS KMS, 외부 키 관리자, 외부 키 또는 VPC 엔드포인트 서비스의 구성 또는 작업 문제를 포함하여 외부에서 문제가 발생할 수 있습니다.
- 네트워킹 - 연결 또는 네트워킹 문제일 수 있습니다(예: 프록시 엔드포인트, 포트, IP 스택 또는 프라이빗 DNS 이름 또는 도메인 문제).

> **Note**
>
> 외부 키 스토어의 관리 작업이 실패하면 여러 가지 예외가 생성됩니다. 그러나 암호화 작업은 외부 키 스토어의 외부 구성 또는 연결 상태와 관련된 모든 실패에 `KMSInvalidStateException` 대해 AWS KMS 를 반환합니다. 문제를 식별하려면 함께 제공되는 오류 메시지 텍스트를 사용합니다.
>
> ConnectCustomKeyStore 작업은 연결 프로세스가 완료되기 전에 빠르게 성공합니다. 연결 프로세스의 성공 여부를 확인하려면 외부 키 스토어의 연결 상태를 확인합니다. 연결 프로세스가 실패하면 AWS KMS 는 원인을 설명하고 해결 방법을 제안하는 연결 오류 코드를 반환합니다.

주제

- 외부 키 스토어에 대한 문제 해결 도구
- 구성 오류
- 외부 키 스토어 연결 오류
- 지연 시간 및 제한 시간 오류
- 인증 자격 증명 오류
- 키 상태 오류
- 복호화 오류
- 외부 키 오류
- 프록시 문제
- 프록시 권한 부여 문제

외부 키 스토어에 대한 문제 해결 도구

AWS KMS 는 외부 키 스토어 및 해당 키의 문제를 식별하고 해결하는 데 도움이 되는 여러 도구를 제공합니다. 이러한 도구를 외부 키 스토어 프록시 및 외부 키 관리자와 함께 제공되는 도구와 결합하여 사용하세요.

> **Note**
>
> 외부 키 스토어 프록시와 외부 키 관리자는 외부 키 스토어와 해당 KMS 키를 생성하고 유지 관리하는 더 쉬운 방법을 제공할 수 있습니다. 자세한 내용은 외부 도구 설명서를 참조하세요.

AWS KMS 예외 및 오류 메시지

AWS KMS 는 발생하는 모든 문제에 대한 자세한 오류 메시지를 제공합니다. AWS KMS 예외에 대한 추가 정보는 AWS Key Management Service API 참조 및 AWS SDKs. AWS KMS 콘솔을 사용하더라도 이러한 참조가 유용할 수 있습니다. 예를 들어, `CreateCustomKeyStores` 작업에 대한 Errors(오류) 목록을 참조하세요.

외부 키 저장소 프록시의 성능을 최적화하기 위해 AWS KMS 는 지정된 집계 기간인 5분 내에 프록시의 신뢰성을 기반으로 예외를 반환합니다. 500 내부 서버 오류, 503 서비스 사용 불가 또는 연결 제한 시간이 발생하는 경우 신뢰성이 높은 프록시에서 `KMSInternalException`을 반환하고 자동 재시도를 트리거하여 요청이 최종적으로 성공할 수 있도록 합니다. 하지만 신뢰성이 낮은 프록시는 `KMSInvalidStateException`을 반환합니다. 자세한 내용은 외부 키 저장소 모니터링을 참조하세요.

외부 키 스토어의 KMS 키를 사용하여 다른 AWS 서비스의 리소스를 보호하는 경우와 같이 다른 AWS 서비스에 문제가 표시되는 경우, AWS 이 서비스는 문제를 식별하는 데 도움이 되는 추가 정보를 제공할 수 있습니다. AWS 서비스가 메시지를 제공하지 않는 경우 KMS 키 사용을 기록하는 CloudTrail 로그에서 오류 메시지를 볼 수 있습니다.

CloudTrail 로그

AWS KMS 콘솔의 작업을 포함한 모든 AWS KMS API 작업은 AWS CloudTrail 로그에 기록됩니다. 는 성공 및 실패한 작업에 대한 로그 항목을 AWS KMS 기록합니다. 실패한 작업의 경우 로그 항목에 AWS KMS 예외 이름(`errorCode`)과 오류 메시지(`errorMessage`)가 포함됩니다. 이 정보를 사용하여 오류를 식별하고 해결할 수 있습니다. 예제는 외부 키 스토어에서 KMS 키로 복호화 실패 섹션을 참조하세요.

로그 항목에는 요청 ID도 포함됩니다. 요청이 외부 키 스토어 프록시에 도달한 경우 로그 항목의 요청 ID를 사용하여 프록시 로그에서 해당 요청을 찾을 수 있습니다(프록시에서 제공하는 경우).

CloudWatch 지표

AWS KMS 는 지연 시간, 제한, 프록시 오류, 외부 키 관리자 상태, TLS 인증서가 만료될 때까지의 일수, 프록시 인증 자격 증명의 보고된 수명을 포함하여 외부 키 스토어의 운영 및 성능에 대한 자세한 Amazon CloudWatch 지표를 기록합니다. 이러한 지표를 사용하여 임박한 문제가 발생하기 전에 알리는 외부 키 스토어 및 CloudWatch 경보 작동을 위한 데이터 모델을 개발할 수 있습니다.

> **Important**
>
> AWS KMS 에서는 외부 키 스토어 지표를 모니터링하기 위해 CloudWatch 경보를 생성할 것을 권장합니다. 이러한 경보는 문제가 발생하기 전에 문제의 조기 징후를 알려줍니다.

모니터링 그래프

AWS KMS 는 AWS KMS 콘솔의 각 외부 키 스토어에 대한 세부 정보 페이지에 외부 키 스토어 CloudWatch 지표의 그래프를 표시합니다. 그래프의 데이터를 사용하여 오류의 원인을 찾고, 임박한 문제를 감지하고, 기준선을 설정하고, CloudWatch 경보 임계값을 세부 조정할 수 있습니다. 모니터링 그래프 해석 및 해당 데이터 사용에 대한 자세한 내용은 외부 키 저장소 모니터링 섹션을 참조하세요.

외부 키 스토어 및 KMS 키 표시

AWS KMS 는 외부 키 스토어 및 KMS 키에 대한 자세한 정보를 AWS KMS 콘솔의 외부 키 스토어와 DescribeCustomKeyStores 및 DescribeKey 작업에 대한 응답으로 표시합니다. 외부 키 스토어

의 연결 상태 및 KMS 키와 연결된 외부 키의 ID를 비롯하여 문제 해결에 사용할 수 있는 정보와 함께 외부 키 스토어 및 KMS 키에 대한 특수 필드 등이 표시됩니다. 자세한 내용은 외부 키 저장소 보기을 참조하세요.

XKS 프록시 테스트 클라이언트

AWS KMS 는 외부 키 스토어 프록시가 AWS KMS 외부 키 스토어 프록시 API 사양을 준수하는지 확인하는 오픈 소스 테스트 클라이언트를 제공합니다. 이 테스트 클라이언트를 사용하여 외부 키 스토어 프록시의 문제를 식별하고 해결할 수 있습니다.

구성 오류

외부 키 스토어를 생성할 때 프록시 인증 자격 증명, 프록시 URI 엔드포인트, 프록시 URI 경로 및 VPC 엔드포인트 서비스 이름과 같은 외부 키 스토어의 구성을 포함하는 속성 값을 지정합니다. 가 속성 값의 오류를 AWS KMS 감지하면 작업이 실패하고 결함이 있는 값을 나타내는 오류를 반환합니다.

잘못된 값을 수정하여 많은 구성 문제를 해결할 수 있습니다. 외부 키 스토어를 연결 해제하지 않고도 잘못된 프록시 URI 경로 또는 프록시 인증 자격 증명을 수정할 수 있습니다. 고유성 요구 사항을 포함하여 이러한 값의 정의는 사전 조건 수집 섹션을 참조하세요. 이러한 값 업데이트에 대한 지침은 외부 키 저장소 속성 편집 섹션을 참조하세요.

프록시 URI 경로 및 프록시 인증 자격 증명 값의 오류를 방지하려면 외부 키 스토어를 생성하거나 업데이트할 때 프록시 구성 파일을 AWS KMS 콘솔에 업로드합니다. 외부 키 스토어 프록시 또는 외부 키 관리자가 제공하는 프록시 URI 경로 및 프록시 인증 자격 증명 값이 있는 JSON 기반 파일입니다. AWS KMS API 작업에는 프록시 구성 파일을 사용할 수 없지만 파일의 값을 사용하여 프록시의 값과 일치하는 API 요청에 대한 파라미터 값을 제공할 수 있습니다.

일반 구성 오류

예외: `CustomKeyStoreInvalidStateException`(`CreateKey`), `KMSInvalidStateException`(암호화 작업), `XksProxyInvalidConfigurationException`(관리 작업, `CreateKey` 제외)

연결 오류 코드: `XKS_PROXY_INVALID_CONFIGURATION`, `XKS_PROXY_INVALID_TLS_CONFIGURATION`

퍼블릭 엔드포인트 연결이 있는 외부 키 스토어의 경우는 외부 키 스토어를 생성하고 업데이트할 때 속성 값을 AWS KMS 테스트합니다. VPC 엔드포인트 서비스 연결이 있는 외부 키 스토어의 경우 AWS KMS 는 외부 키 스토어를 연결하고 업데이트할 때 속성 값을 테스트합니다.

> **Note**
>
> 비동기식인 `ConnectCustomKeyStore` 작업은 외부 키 스토어를 외부 키 스토어 프록시에 연결하려는 시도가 실패하더라도 성공할 수 있습니다. 이 경우 예외는 없지만 외부 키 스토어의 연결 상태는 Failed(실패)이며 연결 오류 코드는 오류 메시지를 설명합니다. 자세한 내용은 외부 키 스토어 연결 오류 단원을 참조하십시오.

가 속성 값의 오류를 AWS KMS 감지하면 작업이 실패하고 다음 오류 메시지 중 하나와 `XksProxyInvalidConfigurationException` 함께가 반환됩니다.

외부 키 스토어 프록시가 잘못된 URI 경로 때문에 요청을 거부했습니다. 외부 키 스토어의 URI 경로를 확인하고 필요한 경우 업데이트합니다.

- 프록시 URI 경로는 프록시 API에 대한 AWS KMS 요청의 기본 경로입니다. APIs 이 경로가 올바르지 않으면 프록시에 대한 모든 요청이 실패합니다. 외부 키 스토어의 현재 프록시 URI 경로를 보려면 AWS KMS 콘솔 또는 `DescribeCustomKeyStores` 작업을 사용합니다. 올바른 프록시 URI 경로를 찾으려면 외부 키 스토어 프록시 설명서를 참조하세요. 프록시 URI 경로 값을 수정하는 방법에 대한 도움말은 외부 키 저장소 속성 편집 섹션을 참조하세요.
- 외부 키 스토어 프록시의 프록시 URI 경로는 외부 키 스토어 프록시 또는 외부 키 관리자 업데이트에 따라 변경될 수 있습니다. 이러한 변경 사항에 대한 자세한 내용은 외부 키 스토어 프록시 또는 외부 키 관리자의 설명서를 참조하세요.

`XKS_PROXY_INVALID_TLS_CONFIGURATION`
AWS KMS 가 외부 키 스토어 프록시에 대한 TLS 연결을 설정할 수 없습니다. 인증서를 포함하여 TLS 구성을 확인합니다.

- 모든 외부 키 스토어 프록시에는 TLS 인증서가 필요합니다. 외부 키 스토어에 대해 지원되는 공인 인증 기관(CA)에서 TLS 인증서를 발급받아야 합니다. 지원되는 CA 목록은 AWS KMS 외부 키 스토어 프록시 API 사양의 Trusted Certificate Authorities(신뢰할 수 있는 인증 기관)를 참조하세요.
- 퍼블릭 엔드포인트 연결의 경우 TLS 인증서의 주체 일반 이름(CN)은 외부 키 스토어 프록시에 대한 프록시 URI 엔드포인트의 도메인 이름과 일치해야 합니다. 예를 들어 퍼블릭 엔드포인트가 https://myproxy.xks.example.com인 경우 TLS, TLS 인증서의 CN은 `myproxy.xks.example.com` 또는 `*.xks.example.com`이어야 합니다.

- VPC 엔드포인트 서비스 연결의 경우 TLS 인증서의 주체 일반 이름(CN)이 VPC 엔드포인트 서비스의 프라이빗 DNS 이름과 일치해야 합니다. 예를 들어 프라이빗 DNS 이름이 myproxy-private.xks.example.com인 경우 TLS 인증서의 CN은 `myproxy-private.xks.example.com` 또는 `*.xks.example.com`이어야 합니다.
- TLS 인증서는 만료될 수 없습니다. TLS 인증서의 만료 날짜를 확인하려면 OpenSSL과 같은 SSL 도구를 사용합니다. 외부 키 스토어와 연결된 TLS 인증서의 만료 날짜를 모니터링하려면 XksProxyCertificateDaysToExpire CloudWatch 지표를 사용합니다. TLS 인증 만료 날짜까지의 일수는 AWS KMS 콘솔의 모니터링 섹션에도 표시됩니다.
- 퍼블릭 엔드포인트 연결을 사용하는 경우 SSL 테스트 도구를 사용하여 SSL 구성을 테스트합니다. 잘못된 인증서 연결로 인해 TLS 연결 오류가 발생할 수 있습니다.

VPC 엔드포인트 서비스 연결 구성 오류

예외: `XksProxyVpcEndpointServiceNotFoundException`, `XksProxyVpcEndpointServiceInvalidConfigurationException`

일반적인 연결 문제 외에도 VPC 엔드포인트 서비스 연결을 사용하여 외부 키 스토어를 생성, 연결 또는 업데이트하는 동안 다음과 같은 문제가 발생할 수 있습니다.는 외부 키 스토어를 생성, 연결 및 업데이트하는 동안 VPC 엔드포인트 서비스 연결을 사용하여 외부 키 스토어의 속성 값을 AWS KMS 테스트합니다. 구성 오류로 인해 관리 작업이 실패하면 다음 예외가 생성됩니다.

```
XksProxyVpcEndpointServiceNotFoundException
```

원인은 다음 중 하나일 수 있습니다.

- VPC 엔드포인트 서비스 이름이 잘못되었습니다. 외부 키 스토어의 VPC 엔드포인트 서비스 이름이 올바르고 외부 키 스토어의 프록시 URI 엔드포인트 값과 일치하는지 확인합니다. VPC 엔드포인트 서비스 이름을 찾으려면 Amazon VPC 콘솔 또는 DescribeVpcEndpointServices 작업을 사용합니다. 기존 외부 키 스토어의 VPC 엔드포인트 서비스 이름과 프록시 URI 엔드포인트를 찾으려면 AWS KMS 콘솔 또는 DescribeCustomKeyStores 작업을 사용합니다. 자세한 내용은 외부 키 저장소 보기을 참조하세요.
- VPC 엔드포인트 서비스는 외부 키 스토어 AWS 리전 와 다른에 있을 수 있습니다. VPC 엔드포인트 서비스와 외부 키 스토어가 동일한 리전에 있는지 확인합니다. (`us-east-1`과 같은 리전 이름의 외부 이름은 com.amazonaws.vpce.useast-1.vpce-svc-example과 같은 VPC 엔드포인트 서비스 이름의 일부입니다.) 외부 키 스토어의 VPC 엔드포인트 서비스에 대한 요구 사항 목록은 VPC 엔드포인트 서비스 섹션을 참조하세요. VPC 엔드포인트 서비스 또는 외부 키 스토어를 다른 리전으로 이동할

수 없습니다. 그러나 VPC 엔드포인트 서비스와 동일한 리전에서 새 외부 키 스토어를 생성할 수 있습니다. 자세한 내용은 VPC 엔드포인트 서비스 연결 구성 및 외부 키 저장소 생성 섹션을 참조하세요.

- AWS KMS 는 VPC 엔드포인트 서비스에 허용되는 보안 주체가 아닙니다. VPC 엔드포인트 서비스에 대한 Allow principals(보안 주체 허용) 목록에는 `cks.kms.<region>.amazonaws.com` 값(예: `cks.kms.eu-west-3.amazonaws.com`)이 포함되어야 합니다. 이 값을 추가하는 방법에 대한 지침은AWS PrivateLink 가이드의 Manage permissions(권한 관리)를 참조하세요.

XksProxyVpcEndpointServiceInvalidConfigurationException

이 오류는 VPC 엔드포인트 서비스가 다음 요구 사항 중 하나를 충족하지 못할 때 발생합니다.

- VPC에는 각각 다른 가용 영역에 있는 두 개 이상의 프라이빗 서브넷이 필요합니다. VPC에 서브넷을 추가하는 방법에 대한 도움말은 Amazon VPC 사용 설명서의 VPC에 서브넷 생성을 참조하세요.
- VPC 엔드포인트 서비스 유형은 Gateway Load Balancer가 아닌 Network Load Balancer를 사용해야 합니다.
- VPC 엔드포인트 서비스에 대해 수락이 필요하지 않아야 합니다(Acceptance required(수락 필요)는 false여야 함). 각 연결 요청을 수동으로 수락해야 하는 경우 VPC 엔드포인트 서비스를 사용하여 외부 키 스토어 프록시에 연결할 수 AWS KMS 없습니다. 자세한 내용은AWS PrivateLink 가이드의 Accept or reject connection requests(연결 요청 수락 또는 거부)를 참조하세요.
- VPC 엔드포인트 서비스에는 퍼블릭 도메인의 하위 도메인인 프라이빗 DNS 이름이 있어야 합니다. 예를 들어 프라이빗 DNS 이름이 `https://myproxy-private.xks.example.com`인 경우 `xks.example.com` 또는 `example.com` 도메인에 퍼블릭 DNS 서버가 있어야 합니다. VPC 엔드포인트 서비스의 프라이빗 DNS 이름을 보거나 변경하려면AWS PrivateLink 가이드의 Manage DNS names for VPC endpoint services(VPC 엔드포인트 서비스의 DNS 이름 관리)를 참조하세요.
- 프라이빗 DNS 이름 도메인의 Domain verification status(도메인 확인 상태)는 `verified`여야 합니다. 프라이빗 DNS 이름 도메인의 확인 상태를 보고 업데이트하려면 5단계: 프라이빗 DNS 이름 도메인 확인 섹션을 참조하세요. 필수 텍스트 레코드를 추가한 후 업데이트된 확인 상태가 표시되는 데 몇 분 정도 걸릴 수 있습니다.

> **Note**
>
> 프라이빗 DNS 도메인은 퍼블릭 도메인의 하위 도메인인 경우에만 확인할 수 있습니다. 그렇지 않으면 필요한 TXT 레코드를 추가한 후에도 프라이빗 DNS 도메인의 확인 상태가 변경되지 않습니다.

- AWS KMS 와 외부 키 스토어 프록시 사이의 방화벽이 IPv4를 통해 포트 443에서 proxy. AWS KMS communicates의 포트 443과 주고받는 트래픽을 허용하는지 확인합니다. 이 값은 구성할 수 없습니다.
- VPC 엔드포인트 서비스의 프라이빗 DNS 이름은 외부 키 스토어의 프록시 URI 엔드포인트 값과 일치해야 합니다. VPC 엔드포인트 서비스 연결이 있는 외부 키 스토어의 경우 프록시 URI 엔드포인트는 `https://` 뒤에 VPC 엔드포인트 서비스의 프라이빗 DNS 이름이 와야 합니다. 프록시 URI 엔드포인트 값을 보려면 외부 키 저장소 보기 섹션을 참조하세요. 프록시 URI 엔드포인트 값을 변경하려면 외부 키 저장소 속성 편집 섹션을 참조하세요.

외부 키 스토어 연결 오류

외부 키 스토어 프록시에 외부 키 스토어를 연결하는 프로세스는 완료하는 데 5분가량 걸립니다. 빨리 실패하지 않는 한, `ConnectCustomKeyStore` 작업은 속성 없이 HTTP 200 응답 및 JSON 객체를 반환합니다. 하지만 이러한 초기 응답은 연결이 성공했음을 의미하지는 않습니다. 외부 키 스토어가 연결되어 있는지 파악하려면 해당 연결 상태를 확인합니다. 연결이 실패하면 외부 키 스토어의 연결 상태가 `FAILED`로 변경되고 실패 원인을 설명하는 연결 오류 코드가 AWS KMS 반환됩니다.

> **Note**
>
> 사용자 지정 키 스토어의 연결 상태가 `FAILED`면 재연결을 시도하기에 앞서 사용자 지정 키 스토어를 연결 해제해야 합니다. 연결 상태가 `FAILED`이면 사용자 지정 키 스토어를 연결할 수 없습니다.

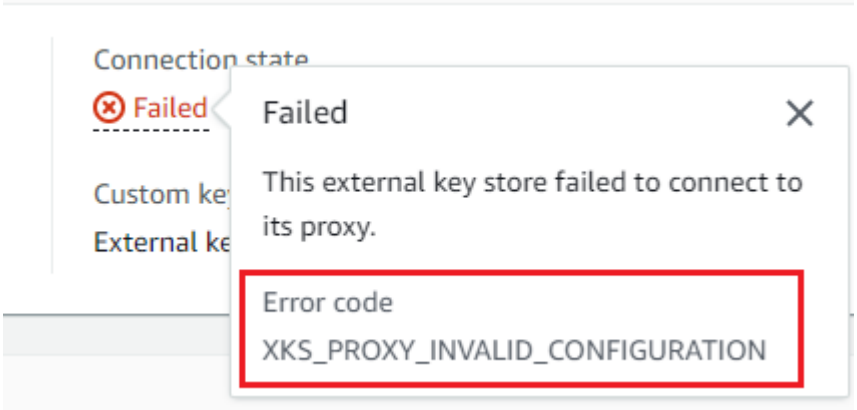
외부 키 스토어의 연결 상태를 보려면 다음을 수행하세요.

- DescribeCustomKeyStores 응답에서 `ConnectionState` 요소의 값을 확인합니다.
- AWS KMS 콘솔에서 연결 상태가 외부 키 스토어 테이블에 나타납니다. 또한 각 외부 키 스토어의 세부 정보 페이지에서 General configuration(일반 구성) 섹션에 Connection state(연결 상태)가 표시됩니다.

연결 상태가 FAILED인 경우 연결 오류 코드는 오류를 설명하는 데 도움이 됩니다.

연결 오류 코드를 보려면 다음을 수행하세요.

- DescribeCustomKeyStores 응답에서 ConnectionErrorCode 요소의 값을 확인합니다. 이 요소는 ConnectionState가 FAILED인 경우에만 DescribeCustomKeyStores 응답에 나타납니다.
- AWS KMS 콘솔에서 연결 오류 코드를 보려면 외부 키 스토어의 세부 정보 페이지에서 실패 값 위에 마우스를 올려 놓습니다.



외부 키 스토어의 연결 오류 코드

다음 연결 오류 코드는 외부 키 스토어에 적용됩니다.

INTERNAL_ERROR

AWS KMS 내부 오류로 인해에서 요청을 완료하지 못했습니다. 요청을 다시 시도하세요. ConnectCustomKeyStore 요청의 경우, 사용자 지정 키 스토어를 연결 해제한 후 다시 연결을 시도하세요.

INVALID_CREDENTIALS

XksProxyAuthenticationCredential 값 중 하나 또는 둘 다 지정된 외부 키 스토어 프록시에서 유효하지 않습니다.

NETWORK_ERRORS

네트워크 오류로 인해가 사용자 지정 키 스토어를 백업 키 스토어에 연결할 수 AWS KMS 없습니다.

`XKS_PROXY_ACCESS_DENIED`

AWS KMS 요청은 외부 키 스토어 프록시에 대한 액세스가 거부됩니다. 외부 키 스토어 프록시에 권한 부여 규칙이 있는 경우 AWS KMS 가 사용자를 대신하여 프록시와 통신하도록 허용하는지 확인합니다.

`XKS_PROXY_INVALID_CONFIGURATION`

구성 오류로 인해 외부 키 스토어가 해당 프록시에 연결할 수 없습니다. `XksProxyUriPath`의 값을 확인합니다.

`XKS_PROXY_INVALID_RESPONSE`

AWS KMS 는 외부 키 스토어 프록시의 응답을 해석할 수 없습니다. 이 연결 오류 코드가 반복적으로 표시되면 외부 키 스토어 프록시 공급업체에 알립니다.

`XKS_PROXY_INVALID_TLS_CONFIGURATION`

AWS KMS TLS 구성이 유효하지 않아가 외부 키 스토어 프록시에 연결할 수 없습니다. 외부 키 스토어 프록시가 TLS 1.2 또는 1.3을 지원하는지 확인합니다. 또한 TLS 인증서가 만료되지 않았는지, `XksProxyUriEndpoint` 값의 호스트 이름과 일치하는지, Trusted Certificate Authorities(신뢰할 수 있는 인증 기관) 목록에 포함된 신뢰할 수 있는 인증 기관에서 서명했는지 확인합니다.

`XKS_PROXY_NOT_REACHABLE`

AWS KMS 가 외부 키 스토어 프록시와 통신할 수 없습니다. `XksProxyUriEndpoint`와 `XksProxyUriPath`가 올바른지 확인합니다. 외부 키 스토어 프록시용 도구를 사용하여 프록시가 활성 상태이고 해당 네트워크에서 사용 가능한지 확인합니다. 또한 외부 키 관리자 인스턴스가 제대로 작동하고 있는지 확인합니다. 프록시가 모든 외부 키 관리자 인스턴스를 사용할 수 없다고 보고하는 경우 이 연결 오류 코드와 함께 연결 시도가 실패합니다.

`XKS_PROXY_TIMED_OUT`

AWS KMS 는 외부 키 스토어 프록시에 연결할 수 있지만 프록시는 할당된 시간 AWS KMS 내에에 응답하지 않습니다. 이 연결 오류 코드가 반복적으로 표시되면 외부 키 스토어 프록시 공급업체에 알립니다.

`XKS_VPC_ENDPOINT_SERVICE_INVALID_CONFIGURATION`

Amazon VPC 엔드포인트 서비스 구성은 AWS KMS 외부 키 스토어의 요구 사항을 준수하지 않습니다.

- VPC 엔드포인트 서비스는 호출자의 AWS 계정에 있는 인터페이스 엔드포인트에 대한 엔드포인트 서비스여야 합니다.

- 각각 다른 가용 영역에 있는 두 개 이상의 서브넷에 연결된 NLB(Network Load Balancer)가 있어야 합니다.
- `Allow principals` 목록에는와 `cks.kms.<region>.amazonaws.com`같은 리전의 AWS KMS 서비스 보안 주체가 포함되어야 합니다`cks.kms.us-east-1.amazonaws.com`.
- 연결 요청 수락을 요구하지 않아야 합니다.
- 프라이빗 DNS 이름이 있어야 합니다. `VPC_ENDPOINT_SERVICE` 연결이 있는 외부 키 스토어의 프라이빗 DNS 이름은 해당 AWS 리전에서 고유해야 합니다.
- 프라이빗 DNS 이름 도메인의 확인 상태는 `verified`여야 합니다.
- TLS 인증서는 엔드포인트에 도달할 수 있는 프라이빗 DNS 호스트 이름을 지정합니다.

`XKS_VPC_ENDPOINT_SERVICE_NOT_FOUND`

AWS KMS 가 외부 키 스토어 프록시와 통신하는 데 사용하는 VPC 엔드포인트 서비스를 찾을 수 없습니다. `XksProxyVpcEndpointServiceName`이 올바르고 AWS KMS 서비스 보안 주체에 Amazon VPC 엔드포인트 서비스에 대한 서비스 소비자 권한이 있는지 확인합니다.

지연 시간 및 제한 시간 오류

예외: `CustomKeyStoreInvalidStateException`(`CreateKey`), `KMSInvalidStateException`(암호화 작업), `XksProxyUriUnreachableException`(관리 작업)

연결 오류 코드: `XKS_PROXY_NOT_REACHABLE`, `XKS_PROXY_TIMED_OUT`

가 250밀리초 제한 시간 간격 내에 프록시에 연결할 AWS KMS 수 없는 경우 예외를 반환하고 `CreateCustomKeyStore`를 `UpdateCustomKeyStore` 반환합니다`XksProxyUriUnreachableException`. 암호화 작업은 문제를 설명하는 오류 메시지와 함께 표준 `KMSInvalidStateException`을 반환합니다. `ConnectCustomKeyStore` 실패하면 문제를 설명하는 연결 오류 코드를 AWS KMS 반환합니다.

제한 시간 오류는 요청을 다시 시도하여 해결할 수 있는 일시적인 문제일 수 있습니다. 문제가 지속되면 외부 키 스토어 프록시가 활성 상태이고 네트워크에 연결되어 있으며 해당 프록시 URI 엔드포인트, 프록시 URI 경로 및 VPC 엔드포인트 서비스 이름(있는 경우)이 외부 키 스토어에서 올바른지 확인합니다. 또한 외부 키 관리자가 외부 키 스토어의 AWS 리전 에 가까운지 확인합니다. 이러한 값을 업데이트해야 하는 경우 외부 키 저장소 속성 편집 섹션을 참조하세요.

지연 시간 패턴을 추적하려면 AWS KMS 콘솔의 Monitoring(모니터링) 섹션에서 `XksProxyLatency` CloudWatch 지표와 Average latency(평균 지연 시간) 그래프(해당 지표 기반)를 사용합니다. 외부 키 스토어 프록시는 지연 시간과 제한 시간을 추적하는 로그와 지표를 생성할 수도 있습니다.

`XksProxyUriUnreachableException`
AWS KMS 는 외부 키 스토어 프록시와 통신할 수 없습니다. 일시적인 네트워크 문제일 수 있습니다. 이 오류가 반복적으로 표시되는 경우 외부 키 스토어 프록시가 활성 상태이고 네트워크에 연결되어 있는지, 해당 엔드포인트 URI가 외부 키 스토어에서 올바른지 확인합니다.

- 외부 키 스토어 프록시가 250밀리초 제한 시간 간격 내에 AWS KMS 프록시 API 요청에 응답하지 않았습니다. 이는 일시적인 네트워크 문제나 프록시의 작동 또는 성능 문제를 나타낼 수 있습니다. 다시 시도해도 문제가 해결되지 않으면 외부 키 스토어 프록시 관리자에게 알리세요.

지연 시간 및 제한 시간 오류는 종종 연결 실패로 나타납니다. ConnectCustomKeyStore 작업이 실패하면 외부 키 스토어의 연결 상태가 로 변경`FAILED`되고 오류를 설명하는 연결 오류 코드가 AWS KMS 반환됩니다. 연결 오류 코드 목록과 오류 해결을 위한 제안 사항은 외부 키 스토어의 연결 오류 코드 섹션을 참조하세요. 모든 사용자 지정 키 스토어와 외부 키 스토어의 연결 코드 목록은 외부 키 스토어에 적용됩니다. 다음 연결 오류는 지연 시간 및 제한 시간과 관련이 있습니다.

`XKS_PROXY_NOT_REACHABLE`

-또는-

`CustomKeyStoreInvalidStateException` , `KMSInvalidStateException` , `XksProxyUriUnreachableException`
AWS KMS 는 외부 키 스토어 프록시와 통신할 수 없습니다. 외부 키 스토어 프록시가 활성 상태이고 네트워크에 연결되어 있는지, 해당 URI 경로와 엔드포인트 URI 또는 VPC 서비스 이름이 외부 키 스토어에서 올바른지 확인합니다.

이 오류는 다음과 같은 이유로 발생할 수 있습니다.

- 외부 키 스토어 프록시가 활성 상태가 아니거나 네트워크에 연결되어 있지 않습니다.
- 외부 키 스토어 구성의 프록시 URI 엔드포인트, 프록시 URI 경로 또는 VPC 엔드포인트 서비스 이름(해당하는 경우) 값에 오류가 있습니다. 외부 키 스토어 구성을 보려면 DescribeCustomKeyStores 작업을 사용하거나 AWS KMS 콘솔에서 외부 키 스토어의 세부 정보 페이지를 확인합니다.
- AWS KMS 와 외부 키 스토어 프록시 간의 네트워크 경로에 포트 오류와 같은 네트워크 구성 오류가 있을 수 있습니다. IPv4를 통해 포트 443의 외부 키 스토어 프록시와 AWS KMS 통신합니다. 이 값은 구성할 수 없습니다.

- 외부 키 스토어 프록시가 GetHealthStatus 응답에서 모든 외부 키 관리자 인스턴스가 UNAVAILABLE이라고 보고하면 XKS_PROXY_NOT_REACHABLE의 ConnectionErrorCode와 함께 ConnectCustomKeyStore 작업이 실패합니다. 도움말은 외부 키 관리자 설명서를 참조하세요.
- 이 오류는 외부 키 관리자와 외부 키 스토어가 AWS 리전 있는 간의 물리적 거리가 길기 때문에 발생할 수 있습니다. AWS 리전 와 외부 키 관리자 간의 ping 지연 시간(네트워크 왕복 시간(RTT))은 35밀리초 이하여야 합니다. 외부 키 관리자에 AWS 리전 더 가까운에서 외부 키 스토어를 생성하거나 외부 키 관리자를에 더 가까운 데이터 센터로 이동해야 할 수 있습니다 AWS 리전.

`XKS_PROXY_TIMED_OUT`

-또는-

`CustomKeyStoreInvalidStateException` , `KMSInvalidStateException` , `XksProxyUriUnreachableException`
외부 키 스토어 프록시가 제시간에 응답하지 않았기 때문에AWS KMS 가 요청을 거부했습니다. 요청을 다시 시도하세요. 이 오류가 반복적으로 표시되면 외부 키 스토어 프록시 관리자에게 보고하세요.

이 오류는 다음과 같은 이유로 발생할 수 있습니다.

- 이 오류는 외부 키 관리자와 외부 키 스토어 프록시 간의 물리적 거리가 멀기 때문에 발생할 수 있습니다. 가능한 경우 외부 키 스토어 프록시를 외부 키 관리자에 더 가깝게 이동합니다.
- 프록시가 요청의 볼륨과 빈도를 처리하도록 설계되지 않은 경우 제한 시간 오류가 발생할 수 있습니다 AWS KMS. CloudWatch 지표가 지속적인 문제를 나타내는 경우 외부 키 스토어 프록시 관리자에게 알리세요.
- 외부 키 관리자와 외부 키 스토어에 대한 Amazon VPC 간의 연결이 제대로 작동하지 않는 경우 제한 시간 오류가 발생할 수 있습니다. 를 사용하는 경우 VPC와 외부 키 관리자가 효과적으로 통신할 수 있는지 AWS Direct Connect확인합니다. 문제를 해결하는 데 도움이 필요하면 Direct Connect 사용 설명서의 문제 해결을 참조하세요 AWS Direct Connect.

`XKS_PROXY_TIMED_OUT`

-또는-

`CustomKeyStoreInvalidStateException` , `KMSInvalidStateException` , `XksProxyUriUnreachableException`
외부 키 스토어 프록시가 할당된 시간 내에 요청에 응답하지 않았습니다. 요청을 다시 시도하세요. 이 오류가 반복적으로 표시되면 외부 키 스토어 프록시 관리자에게 보고하세요.

- 이 오류는 외부 키 관리자와 외부 키 스토어 프록시 간의 물리적 거리가 멀기 때문에 발생할 수 있습니다. 가능한 경우 외부 키 스토어 프록시를 외부 키 관리자에 더 가깝게 이동합니다.

인증 자격 증명 오류

예외: `CustomKeyStoreInvalidStateException`(`CreateKey`), `KMSInvalidStateException`(암호화 작업), `XksProxyIncorrectAuthenticationCredentialException`(`CreateKey` 이외의 관리 작업)

외부 키 스토어 프록시 AWS KMS 에서에 대한 인증 자격 증명을 설정하고 유지 관리합니다. 그런 다음 외부 키 스토어 AWS KMS 를 생성할 때 자격 증명 값을 알려줍니다. 인증 자격 증명을 변경하려면 외부 키 스토어 프록시에서 변경합니다. 그런 다음 외부 키 스토어에 대한 자격 증명을 업데이트합니다. 프록시가 자격 증명을 교체하는 경우 외부 키 스토어에 대한 자격 증명을 업데이트해야 합니다.

외부 키 스토어 프록시가 외부 키 스토어에 대한 프록시 인증 자격 증명으로 서명된 요청을 인증하지 않는 경우 효과는 요청에 따라 다릅니다.

- `CreateCustomKeyStore`와 `UpdateCustomKeyStore`는 `XksProxyIncorrectAuthenticationCredentialException`과 함께 실패합니다.
- `ConnectCustomKeyStore`는 성공하지만 연결은 실패합니다. 연결 상태는 `FAILED`이고 연결 오류 코드는 `INVALID_CREDENTIALS`입니다. 자세한 내용은 외부 키 스토어 연결 오류을 참조하세요.
- 암호화 작업은 외부 키 저장소의 모든 외부 구성 오류 및 연결 상태 오류에 대하여 `KMSInvalidStateException`를 반환합니다. 함께 제공되는 오류 메시지에 문제가 설명되어 있습니다.

외부 키 스토어 프록시가 AWS KMS를 인증할 수 없기 때문에 요청을 거부했습니다. 외부 키 스토어의 자격 증명을 확인하고 필요한 경우 업데이트합니다.

이 오류는 다음과 같은 이유로 발생할 수 있습니다.

- 외부 키 스토어의 액세스 키 ID 또는 보안 액세스 키가 외부 키 스토어 프록시에 설정된 값과 일치하지 않습니다.

  이 오류를 해결하려면 외부 키 스토어의 프록시 인증 자격 증명을 업데이트합니다. 외부 키 스토어를 연결 해제하지 않고 이 변경을 수행할 수 있습니다.
- AWS KMS 와 외부 키 스토어 프록시 간의 역방향 프록시는 SigV4 서명을 무효화하는 방식으로 HTTP 헤더를 조작할 수 있습니다. 이 오류를 수정하려면 프록시 관리자에게 알리세요.

키 상태 오류

예외: `KMSInvalidStateException`

`KMSInvalidStateException`은 사용자 지정 키 스토어의 KMS 키에 대해 두 가지 다른 용도로 사용됩니다.

- `CancelKeyDeletion`과 같은 관리 작업이 실패하고 이 예외를 반환하면 KMS 키의 키 상태가 작업과 호환되지 않는 것입니다.
- 사용자 지정 키 스토어의 KMS 키에 대한 암호화 작업이 `KMSInvalidStateException`과 함께 실패하면 KMS 키의 키 상태에 문제가 있는 것일 수 있습니다. 그러나 암호화 작업은 외부 키 스토어의 모든 외부 구성 오류 및 연결 상태 오류에 `KMSInvalidStateException` 대해 AWS KMS 를 반환합니다. 문제를 식별하려면 예외와 함께 제공되는 오류 메시지를 사용합니다.

AWS KMS API 작업에 필요한 키 상태를 찾으려면 섹션을 참조하세요키의 AWS KMS 키 상태. KMS 키의 키 상태를 확인하려면 (고객 관리형 키 페이지에서 KMS 키의 상태 필드를 확인합니다. 또는 DescribeKey 작업을 사용해 응답의 `KeyState` 요소를 확인합니다. 자세한 내용은 키 식별 및 보기을 참조하세요.

> **Note**
>
> 외부 키 스토어에 있는 KMS 키의 키 상태는 연결된 외부 키의 상태에 대해 아무 것도 나타내지 않습니다. 외부 키 상태에 대한 자세한 내용을 보려면 외부 키 관리자와 외부 키 스토어 프록시 도구를 사용하세요.
>
> `CustomKeyStoreInvalidStateException`은 KMS 키의 키 상태가 아니라 외부 키 스토어의 연결 상태를 나타냅니다.

KMS 키의 키 상태가 Unavailable 또는 PendingDeletion이므로 사용자 지정 스토어의 KMS 키에 대한 암호화 작업이 실패할 수 있습니다. (비활성화된 키에서는 DisabledException이 반환됩니다.)

- KMS 키는 AWS KMS 콘솔에서 KMS Disabled 키를 의도적으로 비활성화하거나 DisableKey 작업을 사용하는 경우에만 키 상태가 됩니다. KMS 키가 비활성화된 동안에는 키를 보고 관리할 수 있지만 암호화 작업에 사용할 수는 없습니다. 이 문제를 해결하려면 키를 활성화합니다. 자세한 내용은 KMS 키 활성화 및 비활성화을 참조하세요.
- KMS 키는 외부 키 스토어가 외부 키 스토어 프록시에서 연결 해제되는 경우 Unavailable 키 상태가 됩니다. 사용 불가 상태인 KMS 키를 수정하려면 외부 키 스토어를 다시 연결합니다. 외부 키 스토어가 다시 연결되고 나면 외부 키 스토어의 KMS 키 상태가 이전 상태(Enabled 또는 Disabled)로 자동 복구됩니다.

  KMS 키는 삭제 예약이 되어 있고 대기 기간에 있는 경우 PendingDeletion 키 상태가 됩니다. 삭제 보류 중인 KMS 키의 키 상태 오류는 키가 암호화에 사용되고 있거나 복호화에 필요하기 때문에 삭제해서는 안 됨을 나타냅니다. KMS 키를 다시 활성화하려면 예약된 삭제를 취소한 다음 키를 활성화합니다. 자세한 내용은 키 삭제 예약을 참조하세요.

복호화 오류

예외: KMSInvalidStateException

외부 키 스토어의 KMS 키를 사용한 복호화 작업이 실패하면 KMSInvalidStateException는 외부 키 스토어의 모든 외부 구성 오류 및 연결 상태 오류에 암호화 작업이 사용하는 표준을 AWS KMS 반환합니다. 오류 메시지가 문제를 나타냅니다.

이중 암호화를 사용하여 암호화된 사이퍼텍스트를 복호화하기 위해 외부 키 관리자는 먼저 외부 키를 사용하여 사이퍼텍스트의 외부 계층을 복호화합니다. 그런 다음 AWS KMS KMS AWS KMS 키의 키 구성 요소를 사용하여 사이퍼텍스트의 내부 계층을 복호화합니다. 유효하지 않거나 손상된 사이퍼텍스트는 외부 키 관리자 또는 AWS KMS에 의해 거부될 수 있습니다.

복호화가 실패하면 KMSInvalidStateException과 함께 다음 오류 메시지가 나타납니다. 요청의 사이퍼텍스트 또는 선택적 암호화 컨텍스트에 문제가 있는 것입니다.

> 외부 키 스토어 프록시가 지정된 사이퍼텍스트나 추가 인증 데이터가 손상 또는 누락되었거나 유효하지 않기 때문에 요청을 거부했습니다.

- 외부 키 스토어 프록시 또는 외부 키 관리자가 사이퍼텍스트 또는 암호화 컨텍스트가 유효하지 않다고 보고하면 일반적으로 전송된 `Decrypt` 요청의 사이퍼텍스트 또는 암호화 컨텍스트에 문제가 있음을 나타냅니다 AWS KMS. `Decrypt` 작업의 경우는 `Decrypt` 요청에서 수신하는 것과 동일한 사이퍼텍스트 및 암호화 컨텍스트를 프록시에 AWS KMS 전송합니다.

  이 오류는 플립드 비트와 같은 전송 중 네트워킹 문제로 인해 발생할 수 있습니다. `Decrypt` 요청을 다시 시도하세요. 문제가 지속되면 사이퍼텍스트가 변경되거나 손상되지 않았는지 확인합니다. 또한에 대한 `Decrypt` 요청의 암호화 컨텍스트가 데이터를 암호화한 요청의 암호화 컨텍스트와 AWS KMS 일치하는지 확인합니다.

암호 복호화를 위해 외부 키 스토어 프록시가 제출한 사이퍼텍스트 또는 암호화 컨텍스트가 손상되었거나 누락되었거나 유효하지 않습니다.

- 가 프록시에서 수신한 사이퍼텍스트를 AWS KMS 거부하면 외부 키 관리자 또는 프록시가 유효하지 않거나 손상된 사이퍼텍스트를 반환했음을 나타냅니다 AWS KMS.

  이 오류는 플립드 비트와 같은 전송 중 네트워킹 문제로 인해 발생할 수 있습니다. `Decrypt` 요청을 다시 시도하세요. 문제가 지속되면 외부 키 관리자가 제대로 작동하는지, 외부 키 스토어 프록시가 외부 키 관리자로부터 받은 사이퍼텍스트를 변경하지 않았는지 확인한 후 반환합니다 AWS KMS.

외부 키 오류

외부 키는 KMS 키의 외부 키 구성 요소 역할을 하는 외부 키 관리자의 암호화 키입니다. AWS KMS 는 외부 키에 직접 액세스할 수 없습니다. 외부 키 스토어 프록시를 통해 외부 키 관리자에게 외부 키를 사용하여 데이터를 암호화하거나 사이퍼텍스트를 복호화하도록 요청해야 합니다.

외부 키 스토어에서 KMS 키를 생성할 때 외부 키 관리자에서 외부 키의 ID를 지정합니다. KMS 키를 생성한 후에는 외부 키 ID를 변경할 수 없습니다. KMS 키 관련 문제를 방지하기 위해 `CreateKey` 작업은 외부 키의 ID와 구성을 확인하도록 외부 키 스토어 프록시에 요청합니다. 외부 키가 KMS 키와 함께 사용하기 위한 요구 사항을 충족하지 않는 경우 문제를 식별하는 예외 및 오류 메시지와 함께 `CreateKey` 작업이 실패합니다.

그러나 KMS 키가 생성된 후 문제가 발생할 수 있습니다. 외부 키 문제로 인해 암호화 작업이 실패하면 작업이 실패하고 문제를 나타내는 오류 메시지와 함께 `KMSInvalidStateException`이 반환됩니다.

외부 키에 대한 CreateKey 오류

예외: `XksKeyAlreadyInUseException`, `XksKeyNotFoundException`, `XksKeyInvalidConfigurationException`

CreateKey 작업은 External key ID(외부 키 ID)(콘솔) 또는 `XksKeyId`(API) 파라미터에 제공한 외부 키의 ID와 속성을 확인하려고 시도합니다. 이 방법은 KMS 키와 함께 외부 키를 사용하려고 시도하기 전에 조기에 오류를 감지하도록 설계되었습니다.

외부 키 사용 중

외부 키 스토어의 각 KMS 키는 다른 외부 키를 사용해야 합니다. `CreateKey`는 KMS 키의 외부 키 ID(XksKeyId)가 외부 키 스토어에서 고유하지 않음을 인식하면 `XksKeyAlreadyInUseException`과 함께 실패합니다.

동일한 외부 키에 대해 여러 ID를 사용하는 경우 `CreateKey`는 중복을 인식하지 않습니다. 그러나 외부 키가 동일한 KMS 키는 AWS KMS 키 구성 요소와 메타데이터가 다르기 때문에 상호 운용할 수 없습니다.

외부 키를 찾을 수 없음

외부 키 스토어 프록시가 KMS 키에 대한 외부 키 ID(XksKeyId)를 사용하여 외부 키를 찾을 수 없다고 보고하면 `CreateKey` 작업이 실패하고 다음 오류 메시지와 함께 `XksKeyNotFoundException`을 반환합니다.

```
외부 키 스토어 프록시가 외부 키를 찾을 수 없기 때문에 요청을 거부했습니다.
```

이 오류는 다음과 같은 이유로 발생할 수 있습니다.

- KMS 키의 외부 키(`XksKeyId`) ID(XksKeyId)가 유효하지 않을 수 있습니다. 외부 키 프록시가 외부 키를 식별하는 데 사용하는 ID를 찾으려면 외부 키 스토어 프록시 또는 외부 키 관리자 설명서를 참조하세요.
- 외부 키가 외부 키 관리자에서 삭제되었을 수 있습니다. 조사하려면 외부 키 관리자 도구를 사용합니다. 외부 키가 영구적으로 삭제된 경우 KMS 키와 함께 다른 외부 키를 사용합니다. 외부 키의 목록 또는 요구 사항은 외부 키 스토어의 KMS 키 요구 사항 섹션을 참조하세요.

외부 키 요구 사항이 충족되지 않음

외부 키 스토어 프록시에서 외부 키가 KMS 키와 함께 사용하기 위한 요구 사항을 충족하지 않는다고 보고하면 `CreateKey` 작업이 실패하고 다음 오류 메시지 중 하나와 함께 `XksKeyInvalidConfigurationException`을 반환합니다.

외부 키의 키 사양은 AES_256이어야 합니다. 지정된 외부 키의 키 사양은 *<key-spec>* .입니다.

- 외부 키는 키 사양이 AES_256인 256비트 대칭 암호화 키여야 합니다. 지정된 외부 키가 다른 유형인 경우 이 요구 사항을 충족하는 외부 키의 ID를 지정합니다.

외부 키의 상태가 ENABLED여야 합니다. 지정된 외부 키의 상태는 *<status>*여야 합니다.

- 외부 키 관리자에서 외부 키를 활성화해야 합니다. 지정된 외부 키가 활성화되지 않은 경우 외부 키 관리자 도구를 사용하여 활성화하거나 활성화된 외부 키를 지정합니다.

외부 키의 키 사용에는 ENCRYPT와 DECRYPT가 포함되어야 합니다. 지정된 외부 키의 키 사용은 <*key-usage* >입니다.

- 외부 키 관리자에서 암호화와 복호화를 위해 외부 키를 구성해야 합니다. 지정된 외부 키에 이러한 작업이 포함되지 않은 경우 외부 키 관리자 도구를 사용하여 작업을 변경하거나 다른 외부 키를 지정합니다.

외부 키에 대한 암호화 작업 오류

예외: `KMSInvalidStateException`

외부 키 스토어 프록시가 KMS 키와 연결된 외부 키를 찾을 수 없거나 외부 키가 KMS 키와 함께 사용하기 위한 요구 사항을 충족하지 못하면 암호화 작업이 실패합니다.

암호화 작업 중 감지되는 외부 키 문제는 KMS 키를 생성하기 전에 감지된 외부 키 문제보다 해결하기 더 어렵습니다. KMS 키를 생성한 후에는 외부 키 ID를 변경할 수 없습니다. KMS 키가 아직 데이터를 암호화하지 않은 경우 KMS 키를 삭제하고 다른 외부 키 ID로 새 키를 생성할 수 있습니다. 그러나 KMS 키로 생성된 사이퍼텍스트는 키의 키 메타데이터와 키 구성 요소가 다르기 때문에 외부 키가 동

일한 키라도 다른 KMS AWS KMS 키로 복호화할 수 없습니다. 대신 가능한 한 외부 키 관리자 도구를 사용하여 외부 키 관련 문제를 해결하세요.

외부 키 스토어 프록시가 외부 키에 문제가 있다고 보고하면 암호화 작업은 문제를 식별하는 오류 메시지와 함께 `KMSInvalidStateException`을 반환합니다.

외부 키를 찾을 수 없음

외부 키 스토어 프록시가 KMS 키에 대한 외부 키 ID(XksKeyId)를 사용하여 외부 키를 찾을 수 없다고 보고하면 암호화 작업이 다음 오류 메시지와 함께 `KMSInvalidStateException`을 반환합니다.

```
외부 키 스토어 프록시가 외부 키를 찾을 수 없기 때문에 요청을 거부했습니다.
```

이 오류는 다음과 같은 이유로 발생할 수 있습니다.

- KMS 키의 외부 키 ID(`XksKeyId`)가 더 이상 유효하지 않습니다.

  KMS 키와 연결된 외부 키 ID를 찾으려면 KMS 키의 세부 정보를 봅니다. 외부 키 프록시가 외부 키를 식별하는 데 사용하는 ID를 찾으려면 외부 키 스토어 프록시 또는 외부 키 관리자 설명서를 참조하세요.

  AWS KMS 는 외부 키 스토어에서 KMS 키를 생성할 때 외부 키 ID를 확인합니다. 그러나 특히 외부 키 ID 값이 별칭이거나 변경 가능한 이름인 경우 ID가 무효화될 수 있습니다. 기존 KMS 키와 연결된 외부 키 ID는 변경할 수 없습니다. KMS 키로 암호화된 사이퍼텍스트를 복호화하려면 외부 키를 기존 외부 키 ID와 다시 연결해야 합니다.

  아직 KMS 키를 사용하여 데이터를 암호화하지 않은 경우 유효한 외부 키 ID로 새 KMS 키를 생성할 수 있습니다. 그러나 KMS 키로 사이퍼텍스트를 생성한 경우 동일한 외부 키를 사용하더라도 다른 KMS 키를 사용하여 사이퍼텍스트를 복호화할 수 없습니다.
- 외부 키가 외부 키 관리자에서 삭제되었을 수 있습니다. 조사하려면 외부 키 관리자 도구를 사용합니다. 가능하면 외부 키 관리자의 사본 또는 백업에서 키 구성 요소를 복구해 봅니다. 외부 키가 영구적으로 삭제되면 연결된 KMS 키로 암호화된 사이퍼텍스트를 복구할 수 없습니다.

외부 키 구성 오류

외부 키 스토어 프록시에서 외부 키가 KMS 키와 함께 사용하기 위한 요구 사항을 충족하지 않는다고 보고하면 암호화 작업이 다음 오류 메시지 중 하나와 함께 `KMSInvalidStateException`을 반환합니다.

외부 키가 요청된 작업을 지원하지 않기 때문에 외부 키 스토어 프록시가 요청을 거부했습니다.

- 외부 키가 암호화와 복호화를 모두 지원해야 합니다. 키 사용에 암호화와 복호화가 포함되지 않은 경우 외부 키 관리자 도구를 사용하여 키 사용을 변경합니다.

외부 키 관리자에서 외부 키가 활성화되지 않았기 때문에 외부 키 스토어 프록시가 요청을 거부했습니다.

- 외부 키를 활성화하고 외부 키 관리자에서 사용할 수 있어야 합니다. 외부 키의 상태가 `Enabled`가 아닌 경우 외부 키 관리자 도구를 사용하여 외부 키를 활성화합니다.

프록시 문제

예외:

`CustomKeyStoreInvalidStateException` (`CreateKey`), `KMSInvalidStateException`(암호화 작업), `UnsupportedOperationException`, `XksProxyUriUnreachableException`, `XksProxyInvalidResponseException`(`CreateKey` 이외의 관리 작업)

외부 키 스토어 프록시는 AWS KMS 와 외부 키 관리자 간의 모든 통신을 중재합니다. 일반 AWS KMS 요청을 외부 키 관리자가 이해할 수 있는 형식으로 변환합니다. 외부 키 스토어 프록시가 AWS KMS 외부 키 스토어 프록시 API 사양을 준수하지 않거나가 제대로 작동하지 않거나와 통신할 수 없는 경우 외부 키 스토어에서 KMS 키를 생성하거나 사용할 수 AWS KMS없습니다.

많은 오류가 외부 키 스토어 아키텍처에서의 중요한 역할 때문에 외부 키 스토어 프록시를 언급하지만 이러한 문제는 외부 키 관리자 또는 외부 키에서 발생할 수 있습니다.

이 섹션의 문제는 외부 키 스토어 프록시의 설계 또는 작업 문제와 관련이 있습니다. 이러한 문제를 해결하려면 프록시 소프트웨어를 변경해야 할 수 있습니다. 프록시 관리자에게 문의하세요. 프록시 문제 진단에 도움이 되도록 AWS KMS 는 외부 키 스토어 프록시가 AWS KMS 외부 키 스토어 프록시 API 사양을 준수하는지 확인하는 오픈 소스 테스트 클라이언트인 XKS 프록시 테스트 클라이언트를 제공합니다.

`CustomKeyStoreInvalidStateException` , `KMSInvalidStateException` 또는 `XksProxyUriUnreachableException`

외부 키 스토어 프록시가 비정상 상태입니다. 이 메시지가 반복적으로 표시되면 외부 키 스토어 프록시 관리자에게 알리세요.

- 이 오류는 외부 키 스토어 프록시의 운영 문제 또는 소프트웨어 오류를 나타낼 수 있습니다. 각 오류를 생성한 AWS KMS API 작업에 대한 CloudTrail 로그 항목을 찾을 수 있습니다. 이 오류는 작업을 다시 시도하여 해결할 수 있습니다. 그러나 이 오류가 지속되면 외부 키 스토어 프록시 관리자에게 알리세요.
- 외부 키 스토어 프록시가 GetHealthStatus 응답에서 모든 외부 키 관리자 인스턴스가 `UNAVAILABLE`이라고 보고하면 이 예외와 함께 외부 키 스토어를 생성하거나 업데이트하려는 시도가 실패합니다. 이 오류가 지속되면 외부 키 관리자 설명서를 참조하세요.

`CustomKeyStoreInvalidStateException`, `KMSInvalidStateException` 또는 `XksProxyInvalidResponseException`
AWS KMS 는 외부 키 스토어 프록시의 응답을 해석할 수 없습니다. 이 오류가 반복적으로 표시되면 외부 키 스토어 프록시 관리자에게 문의하세요.

- AWS KMS 프록시가 구문 분석 또는 해석할 AWS KMS 수 없는 정의되지 않은 응답을 반환하면 작업이이 예외를 생성합니다. 이 오류는 일시적인 외부 문제나 산발적인 네트워크 오류로 인해 가끔 발생할 수 있습니다. 그러나 지속되면 외부 키 스토어 프록시가 AWS KMS 외부 키 스토어 프록시 API 사양을 준수하지 않는 것일 수 있습니다. 외부 키 스토어 관리자 또는 공급업체에 알리세요.

`CustomKeyStoreInvalidStateException`, `KMSInvalidStateException` 또는 `UnsupportedOperationException`

외부 키 스토어 프록시가 요청된 작업을 암호화 작업을 지원하지 않기 때문에 요청을 거부했습니다.

- 외부 키 스토어 프록시는 AWS KMS 외부 키 스토어 프록시 API 사양에 정의된 모든 프록시 API를 지원해야 합니다. 이 오류는 프록시가 요청과 관련된 작업을 지원하지 않음을 나타냅니다. 외부 키 스토어 관리자 또는 공급업체에 알리세요.

프록시 권한 부여 문제

예외: `CustomKeyStoreInvalidStateException`, `KMSInvalidStateException`

일부 외부 키 스토어 프록시는 외부 키 사용에 대한 권한 부여 요구 사항을 구현합니다. 외부 키 스토어 프록시는 특정 사용자가 특정 조건에서 특정 작업을 요청할 수 있도록 하는 권한 부여 체계를 설계하고 구현하는 데 허용되지만 필수는 아닙니다. 예를 들어 프록시를 사용하면 사용자가 특정 외부 키로 암호화할 수 있지만 복호화할 수는 없습니다. 자세한 내용은 외부 키 스토어 프록시 권한 부여(선택 사항) 단원을 참조하십시오.

프록시 권한 부여는가 프록시에 대한 요청에 AWS KMS 포함하는 메타데이터를 기반으로 합니다. `awsSourceVpc` 및 `awsSourceVpce` 필드는 요청이 VPC 엔드포인트에서 발생하고 호출자가 KMS 키와 동일한 계정에 있는 경우에만 메타데이터에 포함됩니다.

```
"requestMetadata": {
    "awsPrincipalArn": string,
    "awsSourceVpc": string, // optional
    "awsSourceVpce": string, // optional
    "kmsKeyArn": string,
    "kmsOperation": string,
    "kmsRequestId": string,
    "kmsViaService": string // optional
}
```

프록시가 권한 부여 실패로 인해 요청을 거부하면 관련 AWS KMS 작업이 실패합니다.는를 `CreateKey` 반환합니다`CustomKeyStoreInvalidStateException`. AWS KMS 암호화 작업은를 반환합니다`KMSInvalidStateException`. 둘 다 다음 오류 메시지를 사용합니다.

> 외부 키 스토어 프록시가 작업에 대한 액세스를 거부했습니다. 사용자와 외부 키 모두에 이 작업에 대한 권한이 있는지 확인하고 요청을 다시 시도하세요.

- 오류를 해결하려면 외부 키 관리자 또는 외부 키 스토어 프록시 도구를 사용하여 인증에 실패한 이유를 확인합니다. 그런 다음 무단 요청을 유발한 절차를 업데이트하거나 외부 키 스토어 프록시 도구를 사용하여 권한 부여 정책을 업데이트합니다. AWS KMS에서는 이 오류를 해결할 수 없습니다.

의 보안 AWS Key Management Service

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. 공동 책임 모델은 이 사항을 클라우드의 보안 및 클라우드 내 보안으로 설명합니다.

- 클라우드 보안 - AWS 는 AWS 클라우드에서 AWS 서비스를 실행하는 인프라를 보호할 책임이 있습니다. AWS 또한는 안전하게 사용할 수 있는 서비스를 제공합니다. 타사 감사자는 규정 AWS 준수 프로그램 일환으로 보안의 효과를 정기적으로 테스트하고 확인합니다. AWS Key Management Service (AWS KMS)에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 규정 준수 프로그램 AWS 제공 범위 내 서비스규정 준수 프로그램 .
- 클라우드의 보안 - 사용자의 책임은 사용하는 AWS 서비스에 따라 결정됩니다. 또한 AWS KMS의 구성 및 사용 외에도 데이터의 민감도 AWS KMS keys, 회사의 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대한 책임은 사용자에게 있습니다.

이 설명서는를 사용할 때 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다 AWS Key Management Service. 보안 및 규정 준수 목표를 충족하도록 AWS KMS 를 구성하는 방법을 보여줍니다.

주제

- 의 데이터 보호 AWS Key Management Service
- 에 대한 자격 증명 및 액세스 관리 AWS Key Management Service
- 에서 로깅 및 모니터링 AWS Key Management Service
- 에 대한 규정 준수 검증 AWS Key Management Service
- 의 복원력 AWS Key Management Service
- 의 인프라 보안 AWS Key Management Service

의 데이터 보호 AWS Key Management Service

AWS Key Management Service 는 암호화 키를 저장하고 보호하여 가용성을 높이는 동시에 강력하고 유연한 액세스 제어를 제공합니다.

주제

- 키 구성 요소 보호
- 데이터 암호화
- 인터네트워크 트래픽 개인 정보 보호

키 구성 요소 보호

기본적으로는 KMS 키의 암호화 키 구성 요소를 AWS KMS 생성하고 보호합니다. 또한는 외부에서 생성 및 보호되는 키 구성 요소에 대한 옵션을 AWS KMS 제공합니다 AWS KMS.

에서 생성된 키 구성 요소 보호 AWS KMS

KMS 키를 생성할 때 기본적으로는 KMS 키의 암호화 구성 요소를 AWS KMS 생성하고 보호합니다.

KMS 키의 키 구성 요소를 보호하기 위해 AWS KMS 는 FIPS 140-3 보안 수준 3 검증 하드웨어 보안 모듈(HSMs)의 분산 플릿을 사용합니다. 각 AWS KMS HSM은의 보안 및 확장성 요구 사항을 충족하기 위해 전용 암호화 기능을 제공하도록 설계된 전용 독립 실행형 하드웨어 어플라이언스입니다 AWS KMS. (중국 리전에서를 AWS KMS 사용하는 HSMs은 OSCCA의 인증을 받았으며 모든 관련 중국 규정을 준수하지만 FIPS 140-3 암호화 모듈 검증 프로그램에서는 검증되지 않습니다.)

KMS 키의 키 구성 요소는 HSM에서 생성될 때 기본적으로 암호화됩니다. 키 구성 요소는 HSM 휘발성 메모리 내에서만 암호화 작업에 사용하는 데 걸리는 몇 밀리초 동안만 암호 해독됩니다. 키 구성 요소가 활성 상태로 사용되지 않을 때마다 HSM 내에서 암호화되어 내구성이 뛰어나고(99.999999999%), 지연 시간이 짧은 영구 스토리지로 전송되어 HSM과 분리되어 격리된 상태로 유지됩니다. 일반 텍스트 키 구성 요소는 HSM 보안 경계를 절대 벗어나지 않으며, 결코 디스크에 기록되거나 저장 매체에 지속되지 않습니다. (유일한 예외는 비밀이 아닌 비대칭 키 쌍의 퍼블릭 키입니다.)

AWS 는 모든 유형의 일반 텍스트 암호화 키 구성 요소와 사람이 상호 작용하지 않는다는 기본 보안 원칙으로를 어설션합니다 AWS 서비스. AWS 서비스 연산자를 포함하여 일반 텍스트 키 구성 요소를 보거나 액세스하거나 내보낼 수 있는 메커니즘은 없습니다. 이 원칙은 심각한 장애 및 재해 복구 이벤트 중에도 적용됩니다. 의 일반 텍스트 고객 키 구성 AWS KMS 요소는 고객 또는 대리인의 서비스에 대한 승인된 요청에 대한 응답으로만 AWS KMS FIPS 140-3 검증 HSMs 내의 암호화 작업에 사용됩니다.

고객 관리형 키의 경우 키를 AWS 계정 생성하는는 키의 유일하고 이전할 수 없는 소유자입니다. 소유 계정은 키에 대한 액세스를 제어하는 권한 부여 정책에 대한 완전하고 독점적인 제어를 갖습니다. 의 경우 AWS 관리형 키 AWS 계정 는에 대한 요청을 승인하는 IAM 정책을 완벽하게 제어할 수 있습니다 AWS 서비스.

외부에서 생성된 키 구성 요소 보호 AWS KMS

AWS KMS 는에서 생성된 키 구성 요소에 대한 대안을 제공합니다 AWS KMS.

선택적 AWS KMS 기능인 사용자 지정 키 스토어를 사용하면 외부에서 생성 및 사용되는 키 구성 요소가 지원하는 KMS 키를 생성할 수 있습니다 AWS KMS. AWS CloudHSM 키 스토어의 KMS 키는 사용자가 제어하는 AWS CloudHSM 하드웨어 보안 모듈의 키로 지원됩니다. 이러한 HSM은 FIPS 140-2 보안 수준 3 또는 FIPS 140-3 보안 수준 3 인증을 받았습니다. 외부 키 스토어의 KMS 키는 프라이빗 데이터 센터의 물리적 HSM AWS과 같이 외부에서 제어하고 관리하는 외부 키 관리자의 키로 지원됩니다.

또 다른 옵션 기능을 사용하면 KMS 키에 대한 키 구성 요소를 가져올 수 있습니다. 가져온 키 구성 요소가 전송되는 동안 보호하려면 AWS KMS HSM에서 생성된 RSA 키 페어의 퍼블릭 키를 사용하여 키 구성 요소를 암호화 AWS KMS합니다. 가져온 키 구성 요소는 AWS KMS HSM에서 복호화되고 HSM의 대칭 키로 다시 암호화됩니다. 모든 AWS KMS 키 구성 요소와 마찬가지로 일반 텍스트로 가져온 키 구성 요소는 HSMs되지 않은 상태로 두지 않습니다. 그러나 키 구성 요소를 제공한 고객은 AWS KMS 외부 키 구성 요소를 안전하게 사용하고, 지속적으로 관리 및 유지 보수할 할 책임이 있습니다.

데이터 암호화

의 데이터는 AWS KMS keys 및 해당 데이터가 나타내는 암호화 키 구성 요소로 AWS KMS 구성됩니다. 이 키 구성 요소는 AWS KMS 하드웨어 보안 모듈(HSM) 내에서만 일반 텍스트로 존재하며 사용 중일 때만 존재합니다. 그렇지 않으면 키 구성 요소가 암호화되어 내구성이 뛰어난 영구 스토리지에 저장됩니다.

가 KMS 키에 대해 AWS KMS 생성하는 키 구성 요소는 HSM의 AWS KMS 경계를 암호화되지 않은 상태로 두지 않습니다. HSMs AWS KMS API 작업에서는 내보내거나 전송하지 않습니다. 다중 리전 키의 경우 예외입니다. 여기서는 교차 리전 복제 메커니즘을 AWS KMS 사용하여 한의 HSM에서 다른의 AWS 리전 HSM으로 다중 리전 키의 키 구성 요소를 복사합니다 AWS 리전. 자세한 내용은 AWS Key Management Service 암호화 세부 정보의 다중 리전 키에 대한 복제 프로세스를 참조하세요.

주제

- 저장 시 암호화
- 전송 중 암호화

저장 시 암호화

AWS KMS 는 FIPS 140-3 보안 수준 3 호환 하드웨어 보안 모듈(HSMs) AWS KMS keys 에서에 대한 키 구성 요소를 생성합니다. 유일한 예외는가 KMS 키를 생성하는 데 AWS KMS 사용하는 HSMs이 모

든 관련 중국 규정을 준수하지만 FIPS 140-3 암호화 모듈 검증 프로그램에 따라 검증되지 않는 중국 리전입니다. 사용하지 않을 경우 키 구성 요소는 HSM 키에 의해 암호화되어 내구성이 뛰어난 영구 스토리지에 기록됩니다. KMS 키의 키 구성 요소와 키 구성 요소를 보호하는 암호화 키는 HSM을 일반 텍스트 형식으로 남기지 않습니다.

KMS 키에 대한 키 구성 요소의 암호화 및 관리는 전적으로 AWS KMS에 의해 처리됩니다.

자세한 내용은 AWS Key Management Service 암호화 세부 정보에서 작업을 AWS KMS keys 참조하세요.

전송 중 암호화

가 KMS 키에 대해 AWS KMS 생성하는 키 구성 요소는 AWS KMS API 작업에서 내보내거나 전송하지 않습니다.는 키 식별자를 AWS KMS 사용하여 API 작업에서 KMS 키를 나타냅니다. 마찬가지로 사용자 AWS KMS 지정 키 스토어의 KMS 키에 대한 키 구성 요소는 내보낼 수 없으며 AWS KMS 또는 AWS CloudHSM API 작업으로 전송되지 않습니다.

그러나 일부 AWS KMS API 작업은 데이터 키를 반환합니다. 또한 고객은 API 작업을 사용하여 선택한 KMS 키에 대한 키 구성 요소를 가져옵니다.

모든 AWS KMS API 호출은 TLS(전송 계층 보안)를 사용하여 서명하고 전송해야 합니다.는 TLS 1.2 AWS KMS 를 요구하고 모든 리전에서 TLS 1.3을 권장합니다. AWS KMS 또한는 중국 리전을 제외한 모든 리전의 AWS KMS 서비스 엔드포인트에 대해 하이브리드 포스트 퀀텀 TLS를 지원합니다. AWS KMS 는의 FIPS 엔드포인트에 대해 하이브리드 포스트 퀀텀 TLS를 지원하지 않습니다 AWS GovCloud (US). AWS KMS 에 대한 호출은 또한 완전 순방향 암호화를 지원하는 최신 암호 제품군을 필요로 합니다. 즉, 프라이빗 키와 같은 암호가 손상되어도 세션 키가 손상되지 않습니다.

명령줄 인터페이스 또는 API를 AWS 통해에 액세스할 때 FIPS 140-3 검증 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용합니다. 표준 AWS KMS 엔드포인트 또는 AWS KMS FIPS 엔드포인트를 사용하려면 클라이언트가 TLS 1.2 이상을 지원해야 합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 연방 정보 처리 표준(FIPS) 140-3을 참조하세요. AWS KMS FIPS 엔드포인트 목록은 AWS Key Management Service 의 엔드포인트 및 할당량을 참조하세요 AWS 일반 참조.

AWS KMS 서비스 호스트와 HSMs 간의 통신은 인증된 암호화 체계에서 타원 곡선 암호화(ECC) 및 고급 암호화 표준(AES)을 사용하여 보호됩니다. 자세한 내용은 AWS Key Management Service 암호화 세부 정보의 내부 통신 보안을 참조하세요.

인터네트워크 트래픽 개인 정보 보호

AWS KMS 는 AWS Management Console 및 암호화 작업에서 생성 및 관리하고 AWS KMS keys 사용할 수 있는 API 작업 세트를 지원합니다.

AWS KMS 는 프라이빗 네트워크에서 로의 두 가지 네트워크 연결 옵션을 지원합니다 AWS.

- 인터넷을 통한 IPSec VPN 연결
- AWS Direct Connect- 표준 이더넷 광섬유 케이블을 통해 내부 네트워크를 Direct Connect 위치에 연결합니다.

모든 AWS KMS API 호출은 서명되어야 하며 전송 계층 보안(TLS)을 사용하여 전송되어야 합니다. 호출에는 또한 완벽한 순방향 보안을 지원하는 최신 암호 제품군도 필요합니다. KMS 키의 키 구성 요소를 저장하는 하드웨어 보안 모듈(HSMs)에 대한 트래픽은 AWS 내부 네트워크를 통해 알려진 AWS KMS API 호스트에서만 허용됩니다.

퍼블릭 인터넷을 통해 트래픽을 전송하지 않고 Virtual Private Cloud(VPC) AWS KMS 에서에 직접 연결하려면 로 구동되는 VPC 엔드포인트를 사용합니다AWS PrivateLink. 자세한 내용은 VPC 엔드포인트를 AWS KMS 통해에 연결 단원을 참조하십시오.

AWS KMS 는 전송 계층 보안(TLS) 네트워크 암호화 프로토콜에 대한 하이브리드 포스트 양자 키 교환 옵션도 지원합니다. AWS KMS API 엔드포인트에 연결할 때 TLS에서이 옵션을 사용할 수 있습니다.

에 대한 자격 증명 및 액세스 관리 AWS Key Management Service

AWS Identity and Access Management (IAM)를 사용하면 AWS 리소스에 대한 액세스를 안전하게 제어할 수 있습니다. 관리자는 누가 AWS KMS 리소스를 사용할 수 있는 인증(로그인) 및 권한(권한 있음)을 받을 수 있는지 제어합니다. 자세한 내용은 에서 IAM 정책 사용 AWS KMS 단원을 참조하십시오.

키 정책은의 KMS 키에 대한 액세스를 제어하는 기본 메커니즘입니다 AWS KMS. 모든 KMS 키에는 키 정책이 있어야 합니다. 또한 키 정책과 함께 IAM 정책 및 권한 부여를 사용하여 KMS 키에 대한 액세스를 제어할 수 있습니다. 자세한 내용은 KMS 키 액세스 및 권한 단원을 참조하십시오.

Amazon Virtual Private Cloud(VPC)를 사용하는 경우 로 AWS KMS 구동되는 인터페이스 VPC 엔드포인트를 생성할 수 있습니다AWS PrivateLink. VPC 엔드포인트 정책을 사용하여 AWS KMS 엔드포인트에 액세스할 수 있는 보안 주체, 수행할 수 있는 API 호출, 액세스할 수 있는 KMS 키를 확인할 수도 있습니다.

주제

- AWS 에 대한 관리형 정책 AWS Key Management Service
- 에 대한 서비스 연결 역할 사용 AWS KMS

AWS 에 대한 관리형 정책 AWS Key Management Service

AWS 관리형 정책은에서 생성하고 관리하는 독립 실행형 정책입니다 AWS. AWS 관리형 정책은 사용자, 그룹 및 역할에 권한 할당을 시작할 수 있도록 많은 일반적인 사용 사례에 대한 권한을 제공하도록 설계되었습니다.

AWS 관리형 정책은 모든 AWS 고객이 사용할 수 있으므로 특정 사용 사례에 대해 최소 권한을 부여하지 않을 수 있습니다. 사용 사례에 고유한 고객 관리형 정책을 정의하여 권한을 줄이는 것이 좋습니다.

AWS 관리형 정책에 정의된 권한은 변경할 수 없습니다. 가 관리형 정책에 정의된 권한을 AWS 업데이트하는 AWS 경우 업데이트는 정책이 연결된 모든 보안 주체 자격 증명(사용자, 그룹 및 역할)에 영향을 줍니다. AWS AWS 서비스 는 새가 시작되거나 기존 서비스에 새 API 작업을 사용할 수 있게 될 때 AWS 관리형 정책을 업데이트할 가능성이 높습니다.

자세한 내용은 IAM 사용자 가이드의 AWS 관리형 정책을 참조하세요.

AWS 관리형 정책: AWSKeyManagementServicePowerUser

`AWSKeyManagementServicePowerUser` 정책을 IAM ID에 연결할 수 있습니다.

`AWSKeyManagementServicePowerUser` 관리형 정책을 사용하여 계정의 IAM 보안 주체에게 고급 사용자 권한을 부여할 수 있습니다. 고급 사용자는 KMS 키를 생성하고, 자신이 생성한 KMS 키를 사용 및 관리하고, 모든 KMS 키와 IAM 자격 증명을 볼 수 있습니다. `AWSKeyManagementServicePowerUser` 관리형 정책이 있는 보안 주체는 키 정책, 기타 IAM 정책 및 권한 부여를 비롯한 다른 소스에서 권한을 얻을 수도 있습니다.

`AWSKeyManagementServicePowerUser`는 AWS 관리형 IAM 정책입니다. AWS 관리형 정책에 대한 자세한 내용은 IAM 사용 설명서의 AWS 관리형 정책을 참조하세요.

> **Note**
>
> 이 정책의 KMS 키와 관련된 권한(예: `kms:TagResource` 및 `kms:GetKeyRotationStatus`)은 해당 KMS 키에 대한 키 정책에서, AWS 계정 이 IAM 정책을 사용하여 키에 대한 액세스를 제어하도록 명시적으로 허용하는 경우에만 유효합니다. 권한이 KMS 키와 관련된 것인지 확인하려면 AWS KMS 권한 섹션의 리소스(Resources) 열에 KMS 키(KMS key)라는 값이 있는지 확인합니다.
> 이 정책은 고급 사용자에게 작업을 허용하는 키 정책이 있는 모든 KMS 키에 대한 권한을 부여합니다. 계정 간 권한(예: `kms:DescribeKey` 및 `kms:ListGrants`)의 경우 여기에는 신뢰할 수 없는 AWS 계정의 KMS 키가 포함될 수 있습니다. 자세한 내용은 IAM 정책 모범 사례 및 다른 계정의 사용자가 KMS를 사용하도록 허용 섹션을 참조하세요. 다른 계정의 KMS 키에 대한

권한이 유효한지 확인하려면 AWS KMS 권한 섹션에서 계정 간 사용(Cross-account use) 열에 예(Yes)라는 값이 있는지 확인합니다.
보안 주체가 오류 없이 AWS KMS 콘솔을 볼 수 있도록 하려면 보안 주체에 `AWSKeyManagementServicePowerUser` 정책에 포함되지 않은 tag:GetResources 권한이 필요합니다. 별도의 IAM 정책에서 이 권한을 허용할 수 있습니다.

AWSKeyManagementServicePowerUser 관리형 IAM 정책에는 다음 권한이 포함됩니다.

- 보안 주체가 KMS 키를 생성하도록 허용합니다. 이 프로세스에는 키 정책 설정이 포함되므로 고급 사용자는 자신이 생성한 KMS 키를 사용하고 관리할 수 있는 권한을 자신과 다른 사용자에게 부여할 수 있습니다.
- 보안 주체가 모든 KMS 키에서 별칭과 태그를 만들고 삭제할 수 있습니다. 태그나 별칭을 변경하면 KMS 키를 사용하고 관리할 수 있는 권한을 허용하거나 거부할 수 있습니다. 자세한 내용은 에 대한 ABAC AWS KMS 섹션을 참조하세요.
- 보안 주체가 키 ARN, 암호화 구성, 키 정책, 별칭, 태그 및 교체 상태를 포함하여 모든 KMS 키에 대한 자세한 정보를 얻을 수 있습니다.
- 보안 주체가 IAM 사용자, 그룹 및 역할을 나열하도록 허용합니다.
- 이 정책은 보안 주체가 자신이 생성하지 않은 KMS 키를 사용하거나 관리하도록 허용하지 않습니다. 하지만 모든 KMS 키의 별칭 및 태그를 변경할 수 있으므로, KMS 키를 사용하거나 관리할 권한이 허용 또는 거부될 수 있습니다.

이 정책의 권한을 보려면 AWS 관리형 정책 참조의 AWSKeyManagementServicePowerUser를 참조하세요.

AWS 관리형 정책: AWSServiceRoleForKeyManagementServiceCustomKeyStores

`AWSServiceRoleForKeyManagementServiceCustomKeyStores`를 IAM 엔티티에 연결할 수 없습니다. 이 정책은 키 스토어와 연결된 AWS CloudHSM 클러스터를 보고 사용자 지정 AWS CloudHSM 키 스토어와 AWS CloudHSM 클러스터 간의 연결을 지원하는 네트워크를 생성할 수 있는 권한을 부여하는 AWS KMS 서비스 연결 역할에 연결됩니다. 자세한 내용은 AWS CloudHSM 및 Amazon EC2 리소스를 관리할 수 AWS KMS 있는 권한 부여 단원을 참조하십시오.

AWS 관리형 정책: AWSServiceRoleForKeyManagementServiceMultiRegionKeys

`AWSServiceRoleForKeyManagementServiceMultiRegionKeys`를 IAM 엔티티에 연결할 수 없습니다. 이 정책은 서비스 연결 역할에 연결되며, 다중 리전 프라이머리 키의 키 구성 요소 변경 사항을

복제 키에 동기화할 수 있는 AWS KMS 권한을 부여합니다. 자세한 내용은 다중 리전 키 AWS KMS 동기화 권한 부여 단원을 참조하십시오.

AWS KMS AWS 관리형 정책에 대한 업데이트

이 서비스가 이러한 변경 사항을 추적하기 시작한 AWS KMS 이후부터의 AWS 관리형 정책 업데이트에 대한 세부 정보를 봅니다. 이 페이지의 변경 사항에 대한 자동 알림을 받아보려면 AWS KMS 문서 기록 페이지에서 RSS 피드를 구독하세요.

변경	설명	Date
AWSKeyManagementServiceMultiRegionKeysServiceRolePolicy - 기존 정책 업데이트	AWS KMS 는 정책 버전 v2의 관리형 정책에 문 ID(`Sid`) 필드를 추가했습니다.	2024년 11월 21일
AWSKeyManagementServiceCustomKeyStoresServiceRolePolicy - 기존 정책 업데이트	AWS KMS 는 실패 시가 명확한 오류 메시지를 제공할 AWS KMS 수 있도록 AWS CloudHSM 클러스터가 포함된 VPC의 변경 사항을 모니터링할 수 있는 `ec2:DescribeVpcs` `ec2:DescribeNetworkAcls` , 및 `ec2:DescribeNetworkInterfaces` 권한을 추가했습니다.	2023년 11월 10일
AWS KMS 에서 변경 내용 추적 시작	AWS KMS 가 AWS 관리형 정책에 대한 변경 내용 추적을 시작했습니다.	2023년 11월 10일

에 대한 서비스 연결 역할 사용 AWS KMS

AWS Key Management Service 는 AWS Identity and Access Management (IAM) 서비스 연결 역할을 사용합니다. 서비스 연결 역할은 직접 연결된 고유한 유형의 IAM 역할입니다 AWS KMS. 서비스 연결

역할은에서 정의 AWS KMS 하며 서비스에서 사용자를 대신하여 다른 AWS 서비스를 호출하는 데 필요한 모든 권한을 포함합니다.

필요한 권한을 수동으로 추가할 필요가 없으므로 서비스 연결 역할을 더 AWS KMS 쉽게 설정할 수 있습니다.는 서비스 연결 역할의 권한을 AWS KMS 정의하며, 달리 정의되지 않은 한 만 해당 역할을 수임 AWS KMS 할 수 있습니다. 정의된 권한에는 신뢰 정책과 권한 정책이 포함되며, 이 권한 정책은 다른 IAM 엔터티에 연결할 수 없습니다.

먼저 관련 리소스를 삭제한 후에만 서비스 링크 역할을 삭제할 수 있습니다. 이렇게 하면 AWS KMS 리소스에 대한 액세스 권한을 실수로 제거할 수 없기 때문에 리소스가 보호됩니다.

서비스 연결 역할을 지원하는 기타 서비스에 대한 자세한 내용은 IAM으로 작업하는AWS 서비스를 참조해 서비스 연결 역할 열이 예(Yes)인 서비스를 찾으세요. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예(Yes) 링크를 선택합니다.

이 주제에서 다루는 서비스 연결 역할의 업데이트에 대한 세부 정보를 확인하려면 AWS KMS AWS 관리형 정책에 대한 업데이트 페이지를 참조하세요.

주제

- AWS CloudHSM 및 Amazon EC2 리소스를 관리할 수 AWS KMS 있는 권한 부여
- 다중 리전 키 AWS KMS 동기화 권한 부여

AWS CloudHSM 및 Amazon EC2 리소스를 관리할 수 AWS KMS 있는 권한 부여

AWS CloudHSM 키 스토어를 지원하려면 AWS CloudHSM 클러스터에 대한 정보를 가져올 수 있는 권한이 AWS KMS 필요합니다. 또한 AWS CloudHSM 키 스토어를 AWS CloudHSM 클러스터에 연결하는 네트워크 인프라를 생성할 수 있는 권한이 필요합니다. 이러한 권한을 얻기 위해는에서 AWSServiceRoleForKeyManagementServiceCustomKeyStores 서비스 연결 역할을 AWS KMS 생성합니다 AWS 계정. AWS CloudHSM 키 스토어를 생성하는 사용자는 서비스 연결 역할을 생성할 수 있는 `iam:CreateServiceLinkedRole` 권한이 있어야 합니다.

AWSKeyManagementServiceCustomKeyStoresServiceRolePolicy 관리형 정책의 업데이트에 대한 세부 정보를 보려면 AWS KMS AWS 관리형 정책에 대한 업데이트 페이지를 참조하세요.

주제

- AWS KMS 서비스 연결 역할 정보
- 서비스 연결 역할 생성
- 서비스 연결 역할 설명 편집

- 서비스 연결 역할 삭제

AWS KMS 서비스 연결 역할 정보

서비스 연결 역할은 사용자를 대신하여 다른 AWS 서비스를 호출할 수 있는 권한을 한 AWS 서비스에 부여하는 IAM 역할입니다. 복잡한 IAM 정책을 생성하고 유지 관리할 필요 없이 여러 통합 AWS 서비스의 기능을 더 쉽게 사용할 수 있도록 설계되었습니다. 자세한 내용은 에 대한 서비스 연결 역할 사용 AWS KMS 단원을 참조하십시오.

AWS CloudHSM 키 스토어의 경우 AWSServiceRoleForKeyManagementServiceCustomKeyStores AWSKeyManagementServiceCustomKeyStoresServiceRolePolicy 서비스 연결 역할을 AWS KMS 생성합니다. 이 정책은 이 역할에 다음 권한을 부여합니다.

- cloudhsm:Describe* - 사용자 지정 키 스토어에 연결된 AWS CloudHSM 클러스터의 변경 사항을 감지합니다.
- ec2:CreateSecurityGroup - AWS CloudHSM 키 스토어를 연결하여 AWS KMS 와 AWS CloudHSM 클러스터 간의 네트워크 트래픽 흐름을 지원하는 보안 그룹을 생성할 때 사용됩니다.
- ec2:AuthorizeSecurityGroupIngress – AWS CloudHSM 키 스토어를 연결하여에서 AWS CloudHSM 클러스터가 포함된 VPC AWS KMS 로 네트워크 액세스를 허용할 때 사용됩니다.
- ec2:CreateNetworkInterface – AWS CloudHSM 키 스토어를 연결하여 AWS KMS 와 AWS CloudHSM 클러스터 간의 통신에 사용되는 네트워크 인터페이스를 생성할 때 사용됩니다.
- ec2:RevokeSecurityGroupEgress – AWS CloudHSM 키 스토어를 연결하여가 AWS KMS 생성한 보안 그룹에서 모든 아웃바운드 규칙을 제거할 때 사용됩니다.
- ec2:DeleteSecurityGroup – AWS CloudHSM 키 스토어를 연결 해제하여 AWS CloudHSM 키 스토어를 연결할 때 생성된 보안 그룹을 삭제할 때 사용됩니다.
- ec2:DescribeSecurityGroups -가 실패 시 명확한 오류 메시지를 제공할 AWS KMS 수 있도록 AWS CloudHSM 클러스터가 포함된 VPC에서가 AWS KMS 생성한 보안 그룹의 변경 사항을 모니터링하는 데 사용됩니다.
- ec2:DescribeVpcs - 실패 시가 명확한 오류 메시지를 제공할 AWS KMS 수 있도록 AWS CloudHSM 클러스터가 포함된 VPC의 변경 사항을 모니터링하는 데 사용됩니다.
- ec2:DescribeNetworkAcls –가 장애 발생 시 명확한 오류 메시지를 제공할 AWS KMS 수 있도록 AWS CloudHSM 클러스터가 포함된 VPC의 네트워크 ACLs 변경 사항을 모니터링하는 데 사용됩니다.
- ec2:DescribeNetworkInterfaces –가 장애 발생 시 명확한 오류 메시지를 제공할 수 있도록 AWS CloudHSM AWS KMS 클러스터가 포함된 VPC에서 AWS KMS 생성된 네트워크 인터페이스의 변경 사항을 모니터링하는 데 사용됩니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudhsm:Describe*",
        "ec2:CreateNetworkInterface",
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:CreateSecurityGroup",
        "ec2:DescribeSecurityGroups",
        "ec2:RevokeSecurityGroupEgress",
        "ec2:DeleteSecurityGroup",
        "ec2:DescribeVpcs",
        "ec2:DescribeNetworkAcls",
        "ec2:DescribeNetworkInterfaces"
      ],
      "Resource": "*"
    }
  ]
}
```

AWSServiceRoleForKeyManagementServiceCustomKeyStores 서비스 연결 역할은 만 신뢰하므로 `cks.kms.amazonaws.com`만이 서비스 연결 역할을 수임 AWS KMS 할 수 있습니다. 이 역할은가 AWS CloudHSM 클러스터를 보고 AWS CloudHSM 키 스토어를 연결된 AWS CloudHSM 클러스터에 연결 AWS KMS 해야 하는 작업으로 제한됩니다. 추가 AWS KMS 권한을 부여하지 않습니다. 예를 들어 에는 AWS CloudHSM 클러스터, HSMs 또는 백업을 생성, 관리 또는 삭제할 수 있는 권한이 AWS KMS 없습니다.

리전

AWS CloudHSM 키 스토어 기능과 마찬가지로 AWSServiceRoleForKeyManagementServiceCustomKeyStores 역할은 AWS KMS 및를 사용할 수 AWS 리전 있는 모든에서 지원 AWS CloudHSM 됩니다. AWS 리전 각 서비스가 지원하는 목록은의 AWS Key Management Service 엔드포인트 및 할당량과 AWS CloudHSM 엔드포인트 및 할당량을 참조하세요Amazon Web Services 일반 참조.

AWS 서비스에서 서비스 연결 역할을 사용하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 사용을 참조하세요.

서비스 연결 역할 생성

AWS KMS 역할이 아직 없는 경우 AWS CloudHSM 키 스토어를 생성할 때에서 AWS 계정 AWSServiceRoleForKeyManagementServiceCustomKeyStores 서비스 연결 역할을 자동으로 생성합니다. 사용자는 이러한 서비스 연결 역할을 직접 생성하거나 다시 생성할 수 없습니다.

서비스 연결 역할 설명 편집

AWSServiceRoleForKeyManagementServiceCustomKeyStores 서비스 연결 역할에서 역할 이름 또는 정책 설명을 편집할 수 없지만, 역할 설명은 편집이 가능합니다. 지침은 IAM 사용 설명서의 서비스 연결 역할 편집을 참조하세요.

서비스 연결 역할 삭제

AWS KMS 키 스토어를 모두 삭제했더라도는에서 AWS 계정 AWSServiceRoleForKeyManagementServiceCustomKeyStores 서비스 연결 역할을 삭제하지 않습니다. AWS CloudHSM 현재 AWSServiceRoleForKeyManagementServiceCustomKeyStores 서비스 연결 역할을 삭제하는 절차는 없지만 활성 AWS CloudHSM 키 스토어가 없는 한는이 역할을 수임하거나 권한을 사용하지 AWS KMS 않습니다.

다중 리전 키 AWS KMS 동기화 권한 부여

다중 리전 키를 지원하려면 다중 리전 기본 키의 공유 속성을 복제본 키와 동기화할 수 있는 권한이 AWS KMS 필요합니다. 이러한 권한을 얻기 위해는에서 AWSServiceRoleForKeyManagementServiceMultiRegionKeys 서비스 연결 역할을 AWS KMS 생성합니다 AWS 계정. 다중 리전 키를 생성하는 사용자는 서비스 연결 역할을 생성할 수 있는 `iam:CreateServiceLinkedRole` 권한을 보유해야 합니다.

AWS CloudTrail 로그에 공유 속성 AWS KMS 동기화를 기록하는 SynchronizeMultiRegionKey CloudTrail 이벤트를 볼 수 있습니다.

AWSKeyManagementServiceMultiRegionKeysServiceRolePolicy 관리형 정책의 업데이트에 대한 세부 정보를 보려면 AWS KMS AWS 관리형 정책에 대한 업데이트 페이지를 참조하세요.

주제

- 다중 리전 키에 대한 서비스 연결 역할

- 서비스 연결 역할 생성
- 서비스 연결 역할 설명 편집
- 서비스 연결 역할 삭제

다중 리전 키에 대한 서비스 연결 역할

서비스 연결 역할은 사용자를 대신하여 다른 AWS 서비스를 호출할 수 있는 권한을 AWS 한 서비스에 부여하는 IAM 역할입니다. 복잡한 IAM 정책을 생성하고 유지 관리할 필요 없이 여러 통합 AWS 서비스의 기능을 더 쉽게 사용할 수 있도록 설계되었습니다.

다중 리전 키의 경우는 AWSServiceRoleForKeyManagementServiceMultiRegionKeys AWSKeyManagementServiceMultiRegionKeysServiceRolePolicy 서비스 연결 역할을 AWS KMS 생성합니다. 이 정책은 역할에 `kms:SynchronizeMultiRegionKey` 권한을 부여하여 다중 리전 키의 공유 속성을 동기화할 수 있도록 합니다.

AWSServiceRoleForKeyManagementServiceMultiRegionKeys 서비스 연결 역할은 만 신뢰하므로 `mrk.kms.amazonaws.com`만이 서비스 연결 역할을 수임 AWS KMS 할 수 있습니다. 이 역할은 다중 리전 공유 속성을 동기화 AWS KMS 해야 하는 작업으로 제한됩니다. 추가 AWS KMS 권한을 부여하지 않습니다. 예를 들어 AWS KMS 에는 KMS 키를 생성, 복제 또는 삭제할 수 있는 권한이 없습니다.

AWS 서비스에서 서비스 연결 역할을 사용하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 사용을 참조하세요.

JSON

```
{
    "Version":"2012-10-17",
    "Statement" : [
        {
            "Sid" : "KMSSynchronizeMultiRegionKey",
            "Effect" : "Allow",
            "Action" : [
                "kms:SynchronizeMultiRegionKey"
            ],
            "Resource" : "*"
        }
    ]
}
```

서비스 연결 역할 생성

AWS KMS 역할이 아직 없는 경우 다중 리전 키를 생성할 AWS 계정 때에서 AWSServiceRoleForKeyManagementServiceMultiRegionKeys 서비스 연결 역할을 자동으로 생성합니다. 사용자는 이러한 서비스 연결 역할을 직접 생성하거나 다시 생성할 수 없습니다.

서비스 연결 역할 설명 편집

AWSServiceRoleForKeyManagementServiceMultiRegionKeys 서비스 연결 역할에서 역할 이름 또는 정책문을 편집할 수 없지만, 역할 설명은 편집이 가능합니다. 지침은 IAM 사용 설명서의 서비스 연결 역할 편집 단원을 참조하십시오.

서비스 연결 역할 삭제

AWS KMS 는에서 AWSServiceRoleForKeyManagementServiceMultiRegionKeys 서비스 연결 역할을 삭제하지 AWS 계정 않으며 삭제할 수 없습니다. 그러나 AWS 계정 및 리전에 다중 리전 키가 없는 한 AWS KMS 는 AWSServiceRoleForKeyManagementServiceMultiRegionKeys 역할을 수임하거나 해당 권한을 사용하지 않습니다.

에서 로깅 및 모니터링 AWS Key Management Service

모니터링은 AWS KMS에서 AWS KMS keys 의 가용성, 상태 및 사용량을 이해하는 데 중요한 부분입니다. 모니터링은 AWS 솔루션의 보안, 신뢰성, 가용성 및 성능을 유지하는 데 도움이 됩니다.는 KMS 키를 모니터링하기 위한 여러 도구를 AWS 제공합니다.

AWS CloudTrail 로그

AWS KMS API 작업에 대한 모든 호출은 AWS CloudTrail 로그에 이벤트로 캡처됩니다. 이러한 로그는 AWS KMS 콘솔의 모든 API 호출과 및 기타 AWS 서비스의 호출 AWS KMS 을 기록합니다. 다른 AWS 계정에서 KMS 키를 사용하기 위한 호출과 같은 교차 계정 API 호출은 두 계정의 CloudTrail 로그에 기록됩니다.

문제 해결 또는 감사 시 로그를 사용하여 KMS 키의 수명 주기를 재구성할 수 있습니다. 암호화 작업에서 KMS 키의 관리 및 사용을 볼 수도 있습니다. 자세한 내용은 the section called “를 사용하여 로깅 AWS CloudTrail” 단원을 참조하십시오.

Amazon CloudWatch Logs

AWS CloudTrail 및 기타 소스에서 로그 파일을 모니터링, 저장 및 액세스합니다. 자세한 설명은 Amazon CloudWatch 사용 설명서를 참조하세요.

AWS KMS의 경우 CloudWatch는 KMS 키 및 키가 보호하는 리소스의 문제를 방지하는 데 도움이 되는 유용한 정보를 저장합니다. 자세한 내용은 the section called “CloudWatch를 사용하여 키 모니터링” 단원을 참조하십시오.

Amazon EventBridge

AWS KMS 는 KMS 키가 교체 또는 삭제되거나 KMS 키에서 가져온 키 구성 요소가 만료될 때 EventBridge 이벤트를 생성합니다. AWS KMS 이벤트(API 작업)를 검색하고 하나 이상의 대상 함수 또는 스트림으로 라우팅하여 상태 정보를 캡처합니다. 자세한 내용은 the section called “Amazon EventBridge를 사용하여 키 모니터링” 및 Amazon EventBridge 사용 설명서를 참조하세요.

Amazon CloudWatch 지표

에서 원시 데이터를 수집하여 성능 지표 AWS KMS 로 처리하는 CloudWatch 지표를 사용하여 KMS 키를 모니터링할 수 있습니다. 데이터는 2주 간격으로 기록되므로 현재 및 과거 정보의 추세를 볼 수 있습니다. 이렇게 하면 KMS 키가 사용되는 방식과 시간이 지남에 따라 KMS 키의 사용이 어떻게 변경되는지 이해할 수 있습니다. CloudWatch 지표를 사용하여 KMS 키를 모니터링하는 방법에 대한 자세한 내용은 AWS KMS 지표 및 차원 단원을 참조하세요.

Amazon CloudWatch 경보

지정한 기간 동안 단일 지표가 변경되는 것을 관찰합니다. 그런 다음 여러 기간 동안 임계값과 관련된 메트릭 값을 기반으로 작업을 수행합니다. 예를 들어 누군가가 암호화 작업에서 삭제되도록 예약된 KMS 키를 사용하려고 할 때 트리거되는 CloudWatch 경보를 생성할 수 있습니다. 이것은 KMS 키가 여전히 사용 중이며 삭제해서는 안 됨을 나타냅니다. 자세한 내용은 the section called “경보 만들기” 단원을 참조하십시오.

AWS Security Hub CSPM

를 사용하여 보안 업계 표준 및 모범 사례 규정 준수에 대한 AWS KMS 사용량을 모니터링할 수 있습니다 AWS Security Hub CSPM. Security Hub CSPM은 보안 제어를 사용해 리소스 구성 및 보안 표준을 평가하여 다양한 규정 준수 프레임워크를 준수할 수 있도록 지원합니다. 자세한 내용은 AWS Security Hub 사용 설명서의 AWS Key Management Service 제어를 참조하세요.

에 대한 규정 준수 검증 AWS Key Management Service

타사 감사자는 여러 규정 준수 프로그램의 AWS Key Management Service 일환으로의 보안 및 AWS 규정 준수를 평가합니다. 여기에는 SOC, PCI, FedRAMP, HIPAA 등이 포함됩니다.

주제

- 규정 준수 및 보안 문서

- 자세히 알아보기

규정 준수 및 보안 문서

다음 규정 준수 및 보안 문서에서 다룹니다 AWS KMS. AWS Artifact를 사용하여 살펴보십시오.

- C5(Cloud Computing Compliance Controls Catalogue)
- ISO 27001:2013 적용성 보고서(SoA)
- ISO 27001:2013 인증서
- ISO 27017:2015 적용성 보고서(SoA)
- ISO 27017:2015 인증서
- ISO 27018:2015 적용성 보고서(SoA)
- ISO 27018:2014 인증서
- ISO 9001:2015 인증서
- PCI DSS 규정 준수 증명(AOC) 및 책임 요약
- SOC(Service Organization Controls) 1 보고서
- SOC(Service Organization Controls) 2 보고서
- 기밀성에 대한 SOC(Service Organization Controls) 2 보고서
- FedRAMP-High

사용에 대한 도움말은 아티팩트에서 보고서 다운로드를 AWS Artifact참조하세요. AWS

자세히 알아보기

사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표, 관련 법률 및 규정에 따라 AWS KMS 결정됩니다. 를 사용할 때 게시된 표준을 준수해야 AWS KMS 하는 경우는 도움이 되는 리소스를 AWS 제공합니다.

- AWS 규정 준수 프로그램 제공 범위 내 서비스 -이 페이지에는 특정 규정 준수 프로그램의 범위에 속하는 AWS 서비스가 나열됩니다. 일반 정보는 AWS 규정 준수 프로그램을 참조하세요.
- 보안 및 규정 준수 빠른 시작 가이드 -이 배포 가이드에서는 아키텍처 고려 사항에 대해 설명하고 보안 및 규정 준수 중심 기준 환경을 배포하기 위한 단계를 제공합니다 AWS.
- AWS 규정 준수 리소스 -이 워크북 및 가이드 모음은 산업 및 위치에 적용될 수 있습니다.

- AWS Config -이 AWS 서비스는 리소스 구성이 내부 관행, 업계 지침 및 규정을 얼마나 잘 준수하는지 평가합니다.
- AWS Security Hub CSPM -이 AWS 서비스는 내 보안 상태에 대한 포괄적인 보기를 제공합니다 AWS. Security Hub CSPM은 보안 제어를 사용하여 AWS 리소스를 평가하고 보안 업계 표준 및 모범 사례에 대한 규정 준수를 확인합니다. 지원되는 서비스 및 제어 목록은 Security Hub CSPM 제어 참조를 참조하세요.

의 복원력 AWS Key Management Service

AWS 글로벌 인프라는 AWS 리전 및 가용 영역을 중심으로 구축됩니다.는 지연 시간이 짧고 처리량이 많으며 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.

AWS 글로벌 인프라 외에도는 데이터 복원성과 백업 요구 사항을 지원하는 몇 가지 기능을 AWS KMS 제공합니다. AWS 리전 및 가용 영역에 대한 자세한 내용은 AWS 글로벌 인프라를 참조하세요.

리전별 격리

AWS Key Management Service (AWS KMS)는 모든에서 사용할 수 있는 자급자족 리전 서비스입니다 AWS 리전. 의 리전별로 격리된 설계는 한의 가용성 문제가 다른 리전의 AWS KMS 작업에 영향을 AWS 리전 미치지 않도록 AWS KMS 합니다. AWS KMS 는 모든 소프트웨어 업데이트 및 조정 작업이 원활하고 눈에 띄지 않게 수행되므로 계획된 가동 중지 시간이 없도록 설계되었습니다.

AWS KMS 서비스 수준 계약(SLA)에는 모든 KMS APIs. 이 보장을 이행하기 위해 AWS KMS 는 API 요청을 실행하는 데 필요한 모든 데이터 및 권한 부여 정보가 요청을 수신하는 모든 리전 호스트에서 사용할 수 있도록 합니다.

AWS KMS 인프라는 각 리전에서 최소 3개의 가용 영역(AZs)에 복제됩니다. 여러 호스트 장애가 AWS KMS 성능 AWS KMS 에 영향을 미치지 않도록는 리전의 모든 AZs에서 고객 트래픽을 처리하도록 설계되었습니다.

KMS 키의 속성 또는 권한에 대한 변경 사항은 리전의 모든 호스트에서 후속 요청을 올바르게 처리할 수 있도록 리전의 모든 호스트에 복제됩니다. KMS 키를 사용한 암호화 작업 요청은 AWS KMS 하드웨어 보안 모듈(HSMs 플릿으로 전달되며,이 플릿은 KMS 키로 작업을 수행할 수 있습니다.

멀티 테넌트 디자인

의 멀티 테넌트 설계를 AWS KMS 통해 99.999% 가용성 SLA를 충족하고 높은 요청 속도를 유지하면서 키와 데이터의 기밀성을 보호할 수 있습니다.

암호화 작업에 지정한 KMS 키가 항상 사용되는 키인지 확인하기 위해 여러 무결성 보장 메커니즘이 배포됩니다.

KMS 키의 일반 텍스트 키 구성 요소는 광범위하게 보호됩니다. 키 구성 요소는 생성되는 즉시 HSM에서 암호화되며 암호화된 키 구성 요소는 즉시 안전하고 지연 시간이 짧은 스토리지로 옮겨집니다. 암호화된 키는 사용 시 HSM 내에서 검색 및 해독됩니다. 일반 텍스트 키는 암호화 작업을 완료하는 데 필요한 시간 동안만 HSM 메모리에 남아 있습니다. 그런 다음 HSM에서 다시 암호화되고 암호화된 키가 스토리지로 반환됩니다. 일반 텍스트 키 구성 요소는 HSM을 절대 떠나지 않으며 영구 스토리지에 절대 기록되지 않습니다.

의 복원력 모범 사례 AWS KMS

AWS KMS 리소스의 복원력을 최적화하려면 다음 전략을 고려하세요.

- 백업 및 재해 복구 전략을 지원하려면 단일 AWS 리전 에서 생성되며 지정한 리전에만 복제되는 KMS 키인 다중 리전 키를 고려하세요. 다중 리전 키를 사용하면 일반 텍스트를 노출하지 않고 암호화된 리소스를 사이 AWS 리전 (동일한 파티션 내)로 이동하고 필요한 경우 대상 리전에서 리소스를 해독할 수 있습니다. 관련 다중 리전 키는 동일한 키 구성 요소와 키 ID를 공유하므로 상호 운용이 가능하지만 고해상도 액세스 제어를 위한 독립적인 키 정책을 가집니다. 자세한 내용은 AWS KMS의 다중 리전 키를 참조하세요.
- 와 같은 다중 테넌트 서비스에서 키를 보호하려면 키 정책 및 IAM 정책을 포함한 액세스 제어를 사용해야 AWS KMS합니다. 또한 기반 VPC 인터페이스 엔드포인트를 AWS KMS 사용하여에 요청을 보낼 수 있습니다 AWS PrivateLink. 이렇게 하면 Amazon VPC와 간의 모든 통신 AWS KMS 이 VPC로 제한된 전용 AWS KMS 엔드포인트를 사용하여 AWS 네트워크 내에서 완전히 수행됩니다. VPC 엔드포인트 정책을 사용하여 추가 권한 부여 계층을 생성하여 이러한 요청의 보안을 더욱 강화할 수 있습니다. 자세한 내용은 VPC 엔드포인트를 통해 AWS KMS 로 연결을 참조하세요.

의 인프라 보안 AWS Key Management Service

관리형 서비스인 AWS Key Management Service (AWS KMS)는 Amazon Web Services: 보안 프로세스 개요에 설명된 AWS 글로벌 네트워크 보안 절차로 보호됩니다. https://d0.awsstatic.com/whitepapers/Security/AWS_Security_Whitepaper.pdf

네트워크를 AWS KMS 통해에 액세스하려면 AWS KMS API AWS Key Management Service 참조에 설명된 API 작업을 호출할 수 있습니다.는 TLS 1.2를 AWS KMS 요구하고 모든 리전에서 TLS 1.3을 권장합니다. AWS KMS 또한는 중국 리전을 제외한 모든 리전의 AWS KMS 서비스 엔드포인트에 대해 하이브리드 포스트 양자 TLS를 지원합니다. AWS KMS 는의 FIPS 엔드포인트에 대해 하이브리드 포스트 양자 TLS를 지원하지 않습니다 AWS GovCloud (US). 표준 AWS KMS 엔드포인트 또는 AWS KMS FIPS 엔트포인트를 사용하려면 클라이언트가 TLS 1.2 이상을 지원해야 합니다. 클라이언트는 Ephemeral Diffie-Hellman(DHE) 또는 Elliptic Curve Ephemeral Diffie-Hellman(ECDHE)과 같은 완전 전송 보안(PFS)이 포함된 암호 제품군도 지원해야 합니다. Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

또한 요청은 액세스 키 ID 및 IAM 위탁자와 관련된 시크릿 액세스 키를 사용하여 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)를 사용하여 임시 보안 자격 증명을 생성하여 요청에 서명할 수 있습니다.

모든 네트워크 위치에서 이러한 API 작업을 호출할 수 있지만는 소스 IP 주소, VPC 및 VPC 엔드포인트를 기반으로 KMS 키에 대한 액세스를 제어할 수 있는 글로벌 정책 조건을 AWS KMS 지원합니다. 키 정책 및 IAM 정책에서 이러한 조건 키를 사용할 수 있습니다. 그러나 이러한 조건은가 사용자를 대신하여 KMS 키를 AWS 사용하지 못하게 할 수 있습니다. 자세한 내용은 AWS 전역 조건 키을 참조하세요.

예를 들어 다음 키 정책 문을 사용하면 소스 IP 주소가 정책에 지정된 IP 주소 중 하나가 아닌 한 `KMSTestRole` 역할을 수임할 수 있는 사용자가 지정된 암호화 작업에 AWS KMS key 이를 사용할 수 있습니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {"AWS":
    "arn:aws:iam::111122223333:role/KMSTestRole"},
    "Action": [
      "kms:Encrypt",
      "kms:Decrypt",
      "kms:ReEncrypt*",
      "kms:GenerateDataKey*",
      "kms:DescribeKey"
    ],
    "Resource": "*",
```

```
    "Condition": {
      "NotIpAddress": {
        "aws:SourceIp": [
          "192.0.2.0/24",
          "203.0.113.0/24"
        ]
      }
    }
  }
}
```

물리적 호스트의 격리

가 AWS KMS 사용하는 물리적 인프라의 보안에는 Amazon Web Services: 보안 프로세스 개요의 물리적 및 환경적 보안 섹션에 설명된 제어가 적용됩니다. 이전 섹션에 나열된 규정 준수 보고서와 타사 감사 결과에 대한 자세한 내용을 확인할 수 있습니다.

AWS KMS 는 물리적 공격에 저항하기 위한 특정 제어로 설계된 전용 강화 하드웨어 보안 모듈(HSMs)에서 지원됩니다. HSM은 여러 논리적 테넌트 간에 물리적 장치를 공유하는 하이퍼바이저와 같은 가상화 계층이 없는 물리적 장치입니다. AWS KMS keys 에 대한 키 구성 요소는 HSM의 휘발성 메모리에만 저장되며 KMS 키가 사용되는 동안에만 저장됩니다. 이 메모리는 의도한 종료 및 재설정을 포함하여 HSM이 작동 상태를 벗어나면 지워집니다. AWS KMS HSMs. AWS Key Management Service

할당량

모든 사용자에게 AWS KMS 응답성과 성능을 제공하기 위해는 리소스 할당량과 요청 할당량이라는 두 가지 유형의 할당량을 AWS KMS 적용합니다. 각 할당량은 각 AWS 계정의 각 리전마다 독립적으로 계산됩니다.

온디맨드 교체 리소스 AWS KMS 할당량과 AWS CloudHSM 키 스토어 요청 할당량을 제외한 모든 할당량은 조정할 수 있습니다. ??? 할당량 증가를 요청하려면 Service Quotas 사용 설명서의 할당량 증가 요청을 참조하세요. 할당량 감소를 요청하거나, Service Quotas에 나열되지 않은 할당량을 변경하거나, AWS 리전 Service Quotas for를 사용할 수 AWS KMS 없는에서 할당량을 변경하려면 AWS Support Center를 방문하여 사례를 생성하세요.

주제

- 리소스 할당량
- 요청 할당량
- AWS KMS 요청 제한

리소스 할당량

AWS KMS 는 모든 고객에게 빠르고 탄력적인 서비스를 제공할 수 있도록 리소스 할당량을 설정합니다. 일부 리소스 할당량은 생성하는 리소스에만 적용되며 AWS 서비스가 생성하는 리소스에는 적용되지 않습니다. 사용하지만 AWS 계정에는 없는 리소스(예: AWS 소유 키)에는 이러한 할당량이 적용되지 않습니다.

리소스 제한을 초과하면 해당 유형의 추가 리소스를 생성하라는 요청에 `LimitExceededException` 오류 메시지가 생성됩니다.

온디맨드 교체 AWS KMS 리소스 할당량을 제외하고 모든 리소스 할당량은 조정할 수 있습니다. ??? 할당량 증가를 요청하려면 Service Quotas 사용 설명서의 할당량 증가 요청을 참조하세요. 할당량 감소를 요청하거나, Service Quotas에 나열되지 않은 할당량을 변경하거나, AWS 리전 Service Quotas for를 사용할 수 AWS KMS 없는에서 할당량을 변경하려면 AWS Support Center를 방문하여 사례를 생성하세요.

다음 표에서는 각 및 리전의 AWS KMS 리소스 할당량을 나열 AWS 계정 하고 설명합니다.

할당량 이름	기본값	적용 대상	조정 가능
AWS KMS keys	100,000건	고객 관리형 키	예
KMS 키당 별칭	50	고객이 생성한 별칭	예
KMS 키당 권한 부여	50,000	고객 관리형 키	예
사용자 지정 키 스토어 리소스 할당량	10	AWS 계정 및 리전	예
온디맨드 교체	25	고객 관리형 키	아니요

리소스 할당량 외에도 요청 할당량을 AWS KMS 사용하여 서비스의 응답성을 보장합니다. 자세한 내용은 the section called “요청 할당량”을 참조하세요.

AWS KMS keys: 100,000건

AWS 계정의 각 리전에서 최대 100,000개의 고객 관리형 키를 보유할 수 있습니다. 이 할당량은 키 사양 또는 키 상태와 관계없이 모든 AWS 리전 의 모든 고객 관리형 키에 적용됩니다. 각 KMS 키는 하나의 리소스로 간주됩니다. AWS 관리형 키 및 AWS 소유 키이 이 할당량 계산에 포함되지 않습니다.

KMS 키당 별칭: 50

각 고객 관리형 키에 최대 50개의 별칭을 연결할 수 있습니다. 와 AWS 연결된 별칭은이 할당량에 포함되지 AWS 관리형 키 않습니다. 별칭을 만들거나 업데이트할 때 이 할당량이 발생할 수 있습니다.

> **Note**
>
> kms:ResourceAliases 조건은 KMS 키가 이 할당량을 준수하는 경우에만 유효합니다. KMS 키가 이 할당량을 초과하면 `kms:ResourceAliases` 조건에 따라 KMS 키를 사용할 권한이 있는 보안 주체는 KMS 키에 대한 액세스가 거부됩니다. 자세한 내용은 별칭 할당량으로 인한 액세스 거부을 참조하세요.

KMS 키당 별칭 할당량은 a AWS 계정. AWS KMS has에서 리전당 별칭 할당량을 제거한 각 리전의 총 별칭 수를 제한하는 리전당 별칭 할당량을 대체합니다.

KMS 키당 권한 부여: 50,000

각 고객 관리형 키는 AWS KMS와 통합된AWS 서비스에서 생성한 권한 부여를 포함해 최대 50,000개의 권한을 부여할 수 있습니다. 이 할당량은 AWS 관리형 키 또는 AWS 소유 키에 적용되지 않습니다.

이 할당량의 한 가지 효과는 동일한 KMS 키를 동시에 사용하는 50,000개 이상의 권한 부여된 작업을 수행할 수 없다는 것입니다. 할당량에 도달한 후에는 활성 권한 부여가 사용 중지되거나 취소된 경우에만 KMS 키에 대한 새 권한 부여를 생성할 수 있습니다.

예를 들어 Amazon Elastic Block Store(Amazon EBS) 볼륨을 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스에 연결하면 볼륨이 복호화되어 읽을 수 있습니다. 데이터를 복호화할 수 있는 권한을 얻기 위해 Amazon EBS가 각 볼륨에 대한 권한 부여를 생성합니다. 따라서 모든 Amazon EBS 볼륨이 동일한 KMS 키를 사용하는 경우 한 번에 50,000개 이상의 볼륨을 연결할 수 없습니다.

사용자 지정 키 스토어 리소스 할당량: 10

각 AWS 계정 및 리전에 최대 10개의 사용자 지정 키 스토어를 생성할 수 있습니다. 더 생성하려고 하면 CreateCustomKeyStore 작업이 실패합니다.

이 할당량은 연결 상태에 관계없이 모든 AWS CloudHSM 키 스토어 및 외부 키 스토어를 포함하여 각 계정 및 리전의 총 사용자 지정 키 스토어 수에 적용됩니다.

온디맨드 교체: 25

온디맨드 키 교체는 KMS 키당 최대 25회 수행할 수 있습니다. 온디맨드 교체를 더 많이 수행하려고 하면 RotateKeyOnDemand 작업이 실패합니다.

이 할당량은 조정할 수 없습니다. Service Quotas를 사용하거나 사례를 생성하여 늘릴 수 없습니다 AWS Support. 온디맨드 교체 할당량에 도달하지 않으려는 경우 가능하면 자동 키 교체를 사용하는 것이 좋습니다.

요청 할당량

AWS KMS 는 초당 요청된 API 작업 수에 대한 할당량을 설정합니다. 요청 할당량은 API 작업, AWS 리전 및 KMS 키 유형과 같은 기타 요인에 따라 다릅니다. API 요청 할당량을 초과하면가 AWS KMS 요청을 제한합니다.

키 스토어 AWS KMS 요청 할당량을 제외한 모든 요청 할당량은 조정할 수 있습니다. AWS CloudHSM 할당량 증가를 요청하려면 Service Quotas 사용 설명서의 할당량 증가 요청을 참조하세요. 할당량 감

소를 요청하거나, Service Quotas에 나열되지 않은 할당량을 변경하거나, AWS 리전 Service Quotas for를 사용할 수 AWS KMS 없는에서 할당량을 변경하려면 AWS Support Center를 방문하여 사례를 생성하세요.

GenerateDataKey 작업에 대한 요청 할당량을 초과하는 경우 AWS Encryption SDK의 데이터 키 캐싱 기능을 사용하는 것이 좋습니다. 데이터 키를 재사용하면 요청 빈도가 줄어들 수 있습니다 AWS KMS.

요청 할당량 외에도는 리소스 할당량을 AWS KMS 사용하여 모든 사용자의 용량을 보장합니다. 자세한 내용은 리소스 할당량을 참조하세요.

요청 비율의 추세를 보려면 Service Quotas 콘솔을 사용하세요. 요청 비율이 할당량 값의 일정 비율에 도달하면 경고하는 Amazon CloudWatch 경보를 생성할 수도 있습니다. 자세한 내용은 AWS 보안 블로그의 Service Quotas 및 Amazon CloudWatch를 사용하여 AWS KMS API 요청 속도 관리를 참조하세요.

주제

- 각 AWS KMS API 작업에 대한 요청 할당량
- 요청 할당량 적용
- 암호화 작업에 대한 공유 할당량
- 자동으로 이루어지는 API 요청
- 교차 계정 요청
- 사용자 지정 키 스토어 요청 할당량

각 AWS KMS API 작업에 대한 요청 할당량

이 표에는 Service Quotas 할당량 코드와 각 AWS KMS 요청 할당량의 기본값이 나열되어 있습니다. 키 스토어 AWS KMS 요청 할당량을 제외한 모든 요청 할당량은 조정할 수 있습니다. AWS CloudHSM

> **Note**
>
> 이 테이블의 모든 데이터를 보려면 가로 또는 세로로 스크롤해야 할 수도 있습니다.

할당량 이름	기본값(초당 요청 수)

`Cryptographic operations (symmetric) request rate`
 	
적용 대상:
 	
• `Decrypt`
• `Encrypt`
• `GenerateDataKey`
• `GenerateDataKeyWithoutPlaintext`
• `GenerateMac`
• `GenerateRandom`
• `ReEncrypt`
• `VerifyMac`
	이러한 공유 할당량은 AWS 리전 및 요청에 사용되는 KMS 키 유형에 따라 달라집니다. 각 할당량은 별도로 계산됩니다.
	
	• 10,000(공유)
	• 다음 리전에서는 20,000(공유):
	  • 미국 동부(오하이오) us-east-2
	  • 아시아 태평양(싱가포르) ap-southeast-1
	  • 아시아 태평양(시드니) ap-southeast-2
	  • 아시아 태평양(도쿄) ap-northeast-1
	  • EU(프랑크푸르트) eu-central-1
	  • EU(런던) eu-west-2
	  • 유럽(스페인), eu-south-2
	• 다음 리전에서는 100,000(공유):
	  • 미국 동부(버지니아 북부), us-east-1
	  • 미국 서부(오리건), us-west-2
	  • 유럽(아일랜드) eu-west-1

`Cryptographic operations (RSA) request rate` 적용 대상: • `Decrypt` • `Encrypt` • `ReEncrypt` • `Sign` • `Verify`	RSA KMS 키의 경우 1,000(공유)
`Cryptographic operations (ML-DSA) request rate` 적용 대상:	ML-DSA KMS 키의 경우 1,000(공유)

할당량 이름	기본값(초당 요청 수)
• Sign • Verify	
Cryptographic operations (ECC and SM2) request rate 적용 대상: • Decrypt- SM2(중국 리전만 해당) KMS 키에만 지원됨 • DeriveSharedSecret • Encrypt- SM2(중국 리전만 해당) KMS 키에만 지원됨 • ReEncrypt - SM2(중국 리전만 해당) KMS 키에만 지원됨 • Sign • Verify	타원 곡선(ECC) 및 SM2(중국 리전만 해당) KMS 키의 경우 1,000(공유)
Custom key store request quotas 적용 대상: • Decrypt • DeriveSharedSecret • Encrypt • GenerateDataKey • GenerateDataKeyWithoutPlaintext • GenerateRandom • ReEncrypt	사용자 지정 키 스토어 요청 할당량은 각 사용자 지정 키 스토어에 별도로 계산됩니다. • 각 AWS CloudHSM 키 스토어에 대해 1,800(공유) • 각 외부 키 스토어에 대해 1,800회(공유)
CancelKeyDeletion request rate	5

할당량 이름	기본값(초당 요청 수)
ConnectCustomKeyStore request rate	5
CreateAlias request rate	5
CreateCustomKeyStore request rate	5
CreateGrant request rate	50
CreateKey request rate	5
DeleteAlias request rate	15
DeleteCustomKeyStore request rate	5
DeleteImportedKeyMaterial request rate	15
DescribeCustomKeyStores request rate	5
DescribeKey request rate	2000
DisableKey request rate	5
DisableKeyRotation request rate	5
DisconnectCustomKeyStore request rate	5
EnableKey request rate	5
EnableKeyRotation request rate	15

할당량 이름	기본값(초당 요청 수)
GenerateDataKeyPair (ECC_NIST_P256) request rate 적용 대상: • GenerateDataKeyPair • GenerateDataKeyPairWithoutPlaintext	100
GenerateDataKeyPair (ECC_NIST_P384) request rate 적용 대상: • GenerateDataKeyPair • GenerateDataKeyPairWithoutPlaintext	100
GenerateDataKeyPair (ECC_NIST_P521) request rate 적용 대상: • GenerateDataKeyPair • GenerateDataKeyPairWithoutPlaintext	100
GenerateDataKeyPair (ECC_SECG_P256K1) request rate 적용 대상: • GenerateDataKeyPair • GenerateDataKeyPairWithoutPlaintext	100

할당량 이름	기본값(초당 요청 수)
GenerateDataKeyPair (ECC_NIST_EDWARDS25519) request rate 적용 대상: • GenerateDataKeyPair • GenerateDataKeyPairWithoutPlaintext	100

GenerateDataKeyPair (RSA_2048) request rate
 	
적용 대상:
 	
• GenerateDataKeyPair
• GenerateDataKeyPairWithoutPlaintext
	이 할당량은에 따라 다릅니다 AWS 리전.
	
	• 20(다른 모든 리전)
	• 제한은 다음 리전에서 1입니다.
	  • 중국(베이징) cn-north-1
	  • 중국(닝샤) cn-northwest-1

GenerateDataKeyPair (RSA_3072) request rate
 	
적용 대상:
 	
• GenerateDataKeyPair
• GenerateDataKeyPairWithoutPlaintext
	이 할당량은에 따라 다릅니다 AWS 리전.
	
	• 4(다른 모든 리전)
	• 다음 리전에서 제한은 0.5입니다.
	  • 중국(베이징) cn-north-1
	  • 중국(닝샤) cn-northwest-1

GenerateDataKeyPair (RSA_4096) request rate
 	
적용 대상:
 	
• GenerateDataKeyPair
• GenerateDataKeyPairWithoutPlaintext
	이 할당량은에 따라 다릅니다 AWS 리전.
	
	• 1(다른 모든 리전)
	• 다음 리전에서 제한은 0.1입니다.
	  • 중국(베이징) cn-north-1
	  • 중국(닝샤) cn-northwest-1

할당량 이름	기본값(초당 요청 수)
GenerateDataKeyPair (SM2 – China Regions only) request rate 적용 대상: • GenerateDataKeyPair • GenerateDataKeyPairWithoutPlaintext	25
GetKeyLastUsage request rate	5
GetKeyPolicy request rate	1000
GetKeyRotationStatus request rate	1000
GetParametersForImport request rate	1
GetPublicKey request rate	2000
ImportKeyMaterial request rate	15
ListAliases request rate	500
ListGrants request rate	100
ListKeyPolicies request rate	100
ListKeys request rate	500
ListKeyRotations request rate	100
ListResourceTags request rate	2000
ListRetirableGrants request rate	100
PutKeyPolicy request rate	15

할당량 이름	기본값(초당 요청 수)
ReplicateKey request rate ReplicateKey 작업은 기본 키의 지역에서 하나의 ReplicateKey 요청으로 계산되고 복제본의 지역에서 두 개의 CreateKey 요청으로 계산됩니다. CreateKey 요청 중 하나는 키를 생성하기 전에 잠재적인 문제를 감지하기 위한 테스트 실행입니다.	5
RetireGrant request rate	50
RevokeGrant request rate	50
RotateKeyOnDemand request rate	5
ScheduleKeyDeletion request rate	15
TagResource request rate	10
UntagResource request rate	5
UpdateAlias request rate	5
UpdateCustomKeyStore request rate	5
UpdateKeyDescription request rate	5
UpdatePrimaryRegion request rate UpdatePrimaryRegion 작업은 두 개의 UpdatePrimaryRegion 요청, 즉 영향을 받는 두 리전 각각에서 하나의 요청으로 계산됩니다.	5

요청 할당량 적용

요청 할당량 검토 시 다음 정보에 유의하십시오.

- 요청 할당량은 고객 관리형 키 및 AWS 관리형 키 둘 다에 적용됩니다. 의 사용은 계정의 리소스를 보호하는 데 사용 AWS 계정되더라도에 대한 요청 할당량에 포함되지 AWS 소유 키 않습니다.
- 요청 할당량은 FIPS 엔드포인트 및 비 FIPS 엔드포인트로 전송된 요청에 적용됩니다. AWS KMS 서비스 엔드포인트 목록은 AWS Key Management Service 의 엔드포인트 및 할당량을 참조하세요 AWS 일반 참조.
- 제한은 리전에 있는 모든 유형의 KMS 키에 대한 모든 요청을 기준으로 합니다. 이 합계에는 사용자를 대신하여 서비스의 요청을 포함하여 AWS 계정에 있는 모든 보안 주체의 AWS 요청이 포함됩니다.
- 각 요처 할당량은 독립적으로 계산됩니다. 예를 들어 CreateKey 작업에 대한 요청은 CreateAlias 작업에 대한 요청 할당량에 영향을 주지 않습니다. `CreateAlias` 요청이 제한되는 경우에도 `CreateKey` 요청이 성공적으로 완료될 수 있습니다.
- 암호화 작업이 할당량을 공유하지만 공유 할당량은 다른 작업에 대한 할당량과 독립적으로 계산됩니다. 예를 들어 요청 할당량은 Encrypt 및 Decrypt 작업에 대한 호출에 공유되지만 이 할당량은 EnableKey와 같은 관리 작업의 할당량과는 독립적입니다. 예를 들어 유럽(런던) 리전에서는 제한 없이 대칭 KMS 키에 대해 10,000개의 암호화 작업과 이에 덧붙여 초당 5개의 `EnableKey` 작업을 수행할 수 있습니다.

암호화 작업에 대한 공유 할당량

AWS KMS 암호화 작업은 요청 할당량을 공유합니다. KMS 키에서 지원하는 암호화 작업의 조합을 요청할 수 있습니다. 이렇게 하면 총 암호화 작업 수가 KMS 키 유형에 대한 요청 할당량을 초과하지 않습니다. 할당량을 공유하는 GenerateDataKeyPair 및 GenerateDataKeyPairWithoutPlaintext는 예외입니다.

다른 KMS 키 유형에 대한 할당량도 독립적으로 계산됩니다. 각 할당량은 1초 간격으로 지정된 키 유형을 사용하여 AWS 계정 및 리전에서 이러한 작업에 대한 모든 요청에 적용됩니다.

- 암호화 작업(대칭) 요청 비율은 계정 및 리전에서 대칭 KMS 키를 사용하는 암호화 작업에 대한 공유 요청 할당량입니다. 이 할당량은 대칭 암호화 키와 역시 대칭인 HMAC 키를 사용하는 암호화 작업에 적용됩니다.

  예를 들어, 공유 할당량이 초당 요청 10,000 AWS 리전 개인에서 대칭 KMS 키를 사용할 수 있습니다. 초당 7,000개의 GenerateDataKey 요청과 초당 2,000개의 복호화 요청을 수행하는 경우 AWS KMS 는 요청을 제한하지 않습니다. 그러나 매초 9,500건의 `GenerateDataKey` 요청과 1,000건의 Encrypt 요청을 하는 경우 공유 제한을 초과하므로 AWS KMS 는 요청을 제한합니다.

사용자 지정 키 스토어의 대칭 암호화 KMS 키에 대한 암호화 작업은 계정에 대한 암호화 작업(대칭) 요청 빈도와 사용자 지정 키 스토어에 대한 사용자 지정 키 스토어 요청 할당량 모두에 포함됩니다.

- RSA(암호화 작업) 요청 비율은 RSA 비대칭 KMS 키를 사용하는 암호화 작업에 대한 공유 요청 할당량입니다.

  예를 들어 요청 할당량이 초당 1,000건인 경우, 암호화 및 복호화가 가능한 RSA KMS 키로 400건의 암호화 요청 및 200건의 복호화 요청과 서명 및 확인이 가능한 RSA KMS 키로 250건의 서명 요청 및 150건의 확인 요청을 할 수 있습니다.

- 암호화 작업(ECC) 요청 비율은 타원 곡선(ECC) 비대칭 KMS 키와 SM 비대칭 KMS 키를 사용하는 암호화 작업에 대한 공유 요청 할당량입니다.

  예를 들어 요청 할당량이 초당 1,000건인 경우, 서명 및 확인이 가능한 ECC KMS 키로 400건의 서명 요청 및 200건의 확인 요청과 서명 및 확인이 가능한 SM2 KMS 키로 250건의 서명 요청 및 150건의 확인 요청을 할 수 있습니다.

- 사용자 지정 키 스토어 요청 할당량은 사용자 지정 키 스토어의 KMS 키의 암호화 작업에 대한 공유 요청 할당량입니다. 이 할당량은 각 사용자 지정 키 스토어에 별도로 계산됩니다.

  사용자 지정 키 저장소의 대칭 암호화 KMS 키에 대한 암호화 작업은 계정에 대한 암호화 작업(대칭) 요청 빈도와 사용자 지정 키 저장소에 대한 사용자 지정 키 저장소 요청 할당량 모두에 포함됩니다.

다른 키 유형에 대한 할당량도 독립적으로 계산됩니다. 예를 들어 아시아 태평양(싱가포르) 리전에서 대칭 KMS 키와 비대칭 KMS 키를 모두 사용하는 경우, 대칭 KMS 키(HMAC 키 포함)로 초당 최대 10,000건의 호출을 하고, 더해서 RSA 비대칭 KMS 키로 초당 최대 500건의 추가 호출을 하고, 여기에 더해서 ECC 기반 KMS 키로 초당 최대 300건의 추가 요청을 할 수 있습니다.

자동으로 이루어지는 API 요청

직접 API 요청을 하거나 사용자를 대신하여에 API 요청을 하는 통합 AWS 서비스를 사용하여 API 요청을 할 수 AWS KMS 있습니다. 이 할당량은 두 유형의 요청에 모두 적용됩니다.

예를 들어 KMS 키(SSE-KMS)를 사용한 서버 측 암호화를 통해 Amazon S3에 데이터를 저장할 수 있습니다. SSE-KMS로 암호화된 S3 객체를 업로드하거나 다운로드할 때마다 Amazon S3는 사용자를 대신하여 AWS KMS 에 `GenerateDataKey` (업로드용) 또는 `Decrypt` (다운로드용) 요청을 수행합니다. 이러한 요청은 할당량에 포함되므로 SSE-KMS로 암호화된 S3 객체의 초당 총 5,500개(또는에 따라 10,000개 또는 50,000개 AWS 리전)의 업로드 또는 다운로드를 초과하는 경우가 요청을 AWS KMS 제한합니다.

교차 계정 요청

한의 애플리케이션이 다른 계정이 소유한 KMS 키를 AWS 계정 사용하는 경우 이를 교차 계정 요청이라고 합니다. 교차 계정 요청의 경우, AWS KMS 는 KMS 키를 소유한 계정이 아니라 요청을 하는 계정을 제한합니다. 예를 들어 계정 A의 애플리케이션이 계정 B의 KMS 키를 사용하는 경우 KMS 키 사용은 계정 A의 할당량에만 적용됩니다.

사용자 지정 키 스토어 요청 할당량

AWS KMS 는 사용자 지정 키 스토어의 KMS 키에 대한 암호화 작업에 대한 요청 할당량을 유지합니다. 이러한 요청 할당량은 각 사용자 지정 키 스토어에 별도로 계산됩니다.

사용자 지정 키 스토어 요청 할당량	각 사용자 지정 키 스토어의 기본값(초당 요청 수)	조정 가능
AWS CloudHSM 키 스토어 요청 할당량	1800	아니요
외부 키 스토어 요청 할당량	1800	아니요

> **Note**
>
> AWS KMS 사용자 지정 키 스토어 요청 할당량은 Service Quotas 콘솔에 표시되지 않습니다. Service Quotas API 작업을 사용하여 이러한 할당량을 보거나 관리할 수 없습니다.
> AWS CloudHSM 키 스토어와 연결된 AWS CloudHSM 클러스터가 사용자 지정 키 스토어와 관련이 없는 명령을 포함하여 많은 명령을 처리하는 경우 lower-than-expected 속도로 `ThrottlingException` AWS KMS 를 가져올 수 있습니다. 이 경우 요청 속도를 로 낮추거나 AWS KMS, 관련 없는 로드를 줄이거나, AWS CloudHSM 키 스토어 전용 AWS CloudHSM 클러스터를 사용합니다.
> AWS KMS 는 `ExternalKeyStoreThrottle` CloudWatch 지표에서 외부 키 스토어 요청의 제한을 보고합니다. 이 지표를 사용하여 제한 패턴을 보고, 경보를 생성하고, 외부 키 스토어 요청 할당량을 조정할 수 있습니다.

사용자 지정 키 스토어에서 KMS 키의 암호화 작업에 대한 요청은 2개의 할당량에 포함됩니다.

- 암호화 작업(대칭) 요청 빈도 할당량(계정당)

사용자 지정 키 스토어의 KMS 키에 대한 암호화 작업 요청은 각 AWS 계정 및 리전의 `Cryptographic operations (symmetric) request rate` 할당량에 포함됩니다. 예를 들어 미국 동부(버지니아 북부)(us-east-1)에서는 각각 사용자 지정 키 스토어에서 KMS 키를 사용하는 요청을 포함하여 대칭 암호화 KMS 키에 대해 초당 최대 100,000개의 요청을 가질 AWS 계정 수 있습니다.

- 사용자 지정 키 스토어 요청 할당량(사용자 지정 키 스토어당)

  사용자 지정 키 스토어의 KMS 키의 암호화 작업에 대한 요청 역시 초당 작업 1,800회의 `Custom key store request quota`에 포함됩니다. 이러한 할당량은 각 사용자 지정 키 스토어에 별도로 계산됩니다. 여기에는 사용자 지정 키 스토어에서 KMS 키를 AWS 계정 사용하는 여러의 요청이 포함될 수 있습니다.

예를 들어 미국 동부(버지니아 북부)(us-east-1) 리전의 사용자 지정 키 스토어(어느 유형이든)에 있는 KMS 키에 대한 암호화 작업은 `Cryptographic operations (symmetric) request rate` 해당 계정 및 리전의 계정 수준 할당량(초당 요청 100,000개)과 사용자 지정 키 스토어의 `Custom key store request quota` (초당 요청 1,800개)에 포함됩니다. 하지만 사용자 지정 키 스토어의 KMS 키에 대한 관리 작업에 대한 요청(예: PutKeyPolicy)은 해당 계정 수준 할당량(초당 15개 요청)에만 적용됩니다.

AWS KMS 요청 제한

가 모든 고객의 API 요청에 빠르고 안정적인 응답을 제공할 AWS KMS 수 있도록 특정 경계를 초과하는 API 요청을 제한합니다.

조절은가 유효할 수 있는 요청을 AWS KMS 거부하고 다음과 같은 `ThrottlingException` 오류를 반환할 때 발생합니다.

```
You have exceeded the rate at which you may call KMS. Reduce the frequency of your
 calls.
(Service: AWSKMS; Status Code: 400; Error Code: ThrottlingException; Request ID: <ID>
```

AWS KMS 는 다음 조건에 대한 요청을 제한합니다.

- 초당 요청 속도가 계정 및 리전에 대한 AWS KMS 요청 할당량을 초과합니다.

  예를 들어 계정의 사용자가 초당 1,000개의 `DescribeKey` 요청을 제출하는 경우는 해당 초당 모든 후속 `DescribeKey` 요청을 AWS KMS 제한합니다.

제한에 대응하기 위해 백오프 및 재시도 전략을 사용합니다. 이 전략은 일부 AWS SDKs.

- 동일한 KMS 키의 상태를 변경하기 위한 요청 수가 버스트되거나 지속되는 빠른 속도입니다. 이 조건은 종종 “핫 키(hot key)”라고 합니다.

  예를 들어 계정의 애플리케이션이 동일한 KMS 키에 대한 `EnableKey` 및 `DisableKey` 요청의 지속적인 배를 보내는 경우는 요청을 AWS KMS 제한합니다. 이 제한은 요청이 `EnableKey` 및 `DisableKey` 작업에 대한 초당 요청의 요청 제한을 초과하지 않는 경우에도 발생합니다.

  제한에 대처하려면 필요한 요청만 하거나 여러 기능의 요청을 통합하도록 애플리케이션 로직을 조정하세요.

- AWS CloudHSM 키 스토어와 연결된 클러스터가 키 스토어와 관련이 없는 명령을 포함하여 많은 명령을 처리할 때 AWS CloudHSM AWS CloudHSM 키 스토어의 KMS AWS CloudHSM 키에 대한 작업 요청이 lower-than-expected 속도로 제한될 수 있습니다.

  (AWS CloudHSM 클러스터에 사용할 수 있는 PKCS #11 세션이 없는 경우는 더AWS KMS 이상 AWS CloudHSM 키 스토어의 KMS 키에 대한 작업 요청을 제한하지 않습니다. 대신가 발생`KMSInternalException`하고 요청을 다시 시도할 것을 권장합니다.)

요청 비율의 추세를 보려면 Service Quotas 콘솔을 사용하세요. 요청 비율이 할당량 값의 일정 비율에 도달하면 경고하는 Amazon CloudWatch 경보를 생성할 수도 있습니다. 자세한 내용은 AWS 보안 블로그의 Service Quotas 및 Amazon CloudWatch를 사용하여 AWS KMS API 요청 속도 관리를 참조하세요.

온디맨드 교체 리소스 AWS KMS 할당량과 AWS CloudHSM 키 스토어 요청 할당량을 제외한 모든 할당량은 조정할 수 있습니다. ??? 할당량 증가를 요청하려면 Service Quotas 사용 설명서의 할당량 증가 요청을 참조하세요. 할당량 감소를 요청하거나, Service Quotas에 나열되지 않은 할당량을 변경하거나, AWS 리전 Service Quotas for를 사용할 수 AWS KMS 없는에서 할당량을 변경하려면 AWS Support Center를 방문하여 사례를 생성하세요.

> **Note**
>
> AWS KMS 사용자 지정 키 스토어 요청 할당량은 Service Quotas 콘솔에 표시되지 않습니다. Service Quotas API 작업을 사용하여 이러한 할당량을 보거나 관리할 수 없습니다.

AWS SDKs AWS KMS 사용에 대한 코드 예제

다음 코드 예제에서는 AWS 소프트웨어 개발 키트(SDK)와 AWS KMS 함께를 사용하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

코드 예시

- AWS SDKs AWS KMS 사용에 대한 기본 예제
  - 안녕하세요 AWS Key Management Service
  - AWS SDK를 AWS KMS 사용하여의 기본 사항 알아보기
  - AWS SDKs AWS KMS 를 사용하기 위한 작업
    - AWS SDK 또는 CLI와 CreateAlias 함께 사용
    - AWS SDK 또는 CLI와 CreateGrant 함께 사용
    - AWS SDK 또는 CLI와 CreateKey 함께 사용
    - AWS SDK 또는 CLI와 Decrypt 함께 사용
    - AWS SDK 또는 CLI와 DeleteAlias 함께 사용
    - AWS SDK 또는 CLI와 DescribeKey 함께 사용
    - AWS SDK 또는 CLI와 DisableKey 함께 사용
    - AWS SDK 또는 CLI와 EnableKey 함께 사용
    - AWS SDK 또는 CLI와 EnableKeyRotation 함께 사용
    - AWS SDK 또는 CLI와 Encrypt 함께 사용
    - AWS SDK 또는 CLI와 GenerateDataKey 함께 사용
    - AWS SDK 또는 CLI와 GenerateDataKeyWithoutPlaintext 함께 사용
    - AWS SDK 또는 CLI와 GenerateRandom 함께 사용

    - AWS SDK 또는 CLI와 GetKeyPolicy 함께 사용
    - AWS SDK 또는 CLI와 ListAliases 함께 사용
    - AWS SDK 또는 CLI와 ListGrants 함께 사용
    - AWS SDK 또는 CLI와 ListKeyPolicies 함께 사용
    - AWS SDK 또는 CLI와 ListKeys 함께 사용
    - AWS SDK 또는 CLI와 PutKeyPolicy 함께 사용
    - AWS SDK 또는 CLI와 ReEncrypt 함께 사용
    - AWS SDK 또는 CLI와 RetireGrant 함께 사용
    - AWS SDK 또는 CLI와 RevokeGrant 함께 사용
    - AWS SDK 또는 CLI와 ScheduleKeyDeletion 함께 사용
    - AWS SDK 또는 CLI와 Sign 함께 사용
    - AWS SDK 또는 CLI와 TagResource 함께 사용
    - AWS SDK 또는 CLI와 UpdateAlias 함께 사용
    - AWS SDK 또는 CLI와 Verify 함께 사용
- AWS SDKs AWS KMS 사용 시나리오
    - AWS Command Line Interface v2를 사용한 DynamoDB 테이블 암호화 작업

AWS SDKs AWS KMS 사용에 대한 기본 예제

다음 코드 예제에서는 AWS Key Management Service SDKs에서의 기본 사항을 AWS 사용하는 방법을 보여줍니다.

예제

- 안녕하세요 AWS Key Management Service
- AWS SDK를 AWS KMS 사용하여의 기본 사항 알아보기
- AWS SDKs AWS KMS 를 사용하기 위한 작업
    - AWS SDK 또는 CLI와 CreateAlias 함께 사용
    - AWS SDK 또는 CLI와 CreateGrant 함께 사용
    - AWS SDK 또는 CLI와 CreateKey 함께 사용
    - AWS SDK 또는 CLI와 Decrypt 함께 사용
    - AWS SDK 또는 CLI와 DeleteAlias 함께 사용

- AWS SDK 또는 CLI와 DescribeKey 함께 사용
- AWS SDK 또는 CLI와 DisableKey 함께 사용
- AWS SDK 또는 CLI와 EnableKey 함께 사용
- AWS SDK 또는 CLI와 EnableKeyRotation 함께 사용
- AWS SDK 또는 CLI와 Encrypt 함께 사용
- AWS SDK 또는 CLI와 GenerateDataKey 함께 사용
- AWS SDK 또는 CLI와 GenerateDataKeyWithoutPlaintext 함께 사용
- AWS SDK 또는 CLI와 GenerateRandom 함께 사용
- AWS SDK 또는 CLI와 GetKeyPolicy 함께 사용
- AWS SDK 또는 CLI와 ListAliases 함께 사용
- AWS SDK 또는 CLI와 ListGrants 함께 사용
- AWS SDK 또는 CLI와 ListKeyPolicies 함께 사용
- AWS SDK 또는 CLI와 ListKeys 함께 사용
- AWS SDK 또는 CLI와 PutKeyPolicy 함께 사용
- AWS SDK 또는 CLI와 ReEncrypt 함께 사용
- AWS SDK 또는 CLI와 RetireGrant 함께 사용
- AWS SDK 또는 CLI와 RevokeGrant 함께 사용
- AWS SDK 또는 CLI와 ScheduleKeyDeletion 함께 사용
- AWS SDK 또는 CLI와 Sign 함께 사용
- AWS SDK 또는 CLI와 TagResource 함께 사용
- AWS SDK 또는 CLI와 UpdateAlias 함께 사용
- AWS SDK 또는 CLI와 Verify 함께 사용

안녕하세요 AWS Key Management Service

다음 코드 예제에서는 AWS Key Management Service의 사용을 시작하는 방법을 보여 줍니다.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.services.kms.KmsAsyncClient;
import software.amazon.awssdk.services.kms.model.ListKeysRequest;
import software.amazon.awssdk.services.kms.paginators.ListKeysPublisher;
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloKMS {
    public static void main(String[] args) {
        listAllKeys();
    }

    public static void listAllKeys() {
        KmsAsyncClient kmsAsyncClient = KmsAsyncClient.builder()
            .build();
        ListKeysRequest listKeysRequest = ListKeysRequest.builder()
            .limit(15)
            .build();

        /*
         * The `subscribe` method is required when using paginator methods in the
 AWS SDK
         * because paginator methods return an instance of a `ListKeysPublisher`,
 which is
```

```
         * based on a reactive stream. This allows asynchronous retrieval of
 paginated
         * results as they become available. By subscribing to the stream, we can
 process
         * each page of results as they are emitted.
         */
        ListKeysPublisher keysPublisher =
 kmsAsyncClient.listKeysPaginator(listKeysRequest);
        CompletableFuture<Void> future = keysPublisher
            .subscribe(r -> r.keys().forEach(key ->
                System.out.println("The key ARN is: " + key.keyArn() + ". The key
 Id is: " + key.keyId())))
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    System.err.println("Error occurred: " +
 exception.getMessage());
                } else {
                    System.out.println("Successfully listed all keys.");
                }
            });

        try {
            future.join();
        } catch (Exception e) {
            System.err.println("Failed to list keys: " + e.getMessage());
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 ListKeys를 참조하세요.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
include "vendor/autoload.php";

use Aws\Kms\KmsClient;

echo "This file shows how to connect to the KmsClient, uses a paginator to get
 the keys for the account, and lists the KeyIds for up to 10 keys.\n";

$client = new KmsClient([]);

$pageLength = 10; // Change this value to change the number of records shown, or
 to break up the result into pages.

$keys = [];
$keysPaginator = $client->getPaginator("ListKeys", ['Limit' => $pageLength]);
foreach($keysPaginator as $page){
    foreach($page['Keys'] as $index => $key){
        echo "The $index index Key's ID is: {$key['KeyId']}\n";
    }
    echo "End of page one of results. Alter the \$pageLength variable to see more
 results.\n";
    break;
}
```

- API에 대한 세부 정보는 AWS SDK for PHP API 참조의 ListKeys를 참조하세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        oo_result = lo_kms->listkeys( ).
        MESSAGE 'Retrieved KMS keys list.' TYPE 'I'.
      CATCH /aws1/cx_kmskmsinternalex.
```

```
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 ListKeys를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 AWS KMS 사용하여의 기본 사항 알아보기

다음 코드 예제는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- KMS 키를 생성합니다.
- 계정의 KMS 키를 나열하고 해당 키에 대한 세부 정보를 확인하세요.
- KMS 키를 활성화 및 비활성화합니다.
- 클라이언트 측 암호화에 사용할 수 있는 대칭 데이터 키를 생성하세요.
- 데이터에 디지털 방식으로 서명하는 데 사용되는 비대칭 키를 생성합니다.
- 키에 태그를 지정합니다.
- KMS 키를 삭제합니다.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

명령 프롬프트에서 시나리오를 실행합니다.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.kms.model.AlreadyExistsException;
```

```
import software.amazon.awssdk.services.kms.model.DisabledException;
import software.amazon.awssdk.services.kms.model.EnableKeyRotationResponse;
import software.amazon.awssdk.services.kms.model.KmsException;
import software.amazon.awssdk.services.kms.model.NotFoundException;
import software.amazon.awssdk.services.kms.model.RevokeGrantResponse;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class KMSScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    private static String accountId = "";

    private static final Logger logger =
 LoggerFactory.getLogger(KMSScenario.class);

    static KMSActions kmsActions = new KMSActions();

    static Scanner scanner = new Scanner(System.in);

    static String aliasName = "alias/dev-encryption-key";

    public static void main(String[] args) {
        final String usage = """
            Usage: <granteePrincipal>

            Where:
               granteePrincipal - The principal (user, service account, or group)
 to whom the grant or permission is being given.
            """;

        if (args.length != 1) {
```

```
            logger.info(usage);
            return;
        }
        String granteePrincipal = args[0];
        String policyName = "default";

        accountId = kmsActions.getAccountId();
        String keyDesc = "Created by the AWS KMS API";

        logger.info(DASHES);
        logger.info("""
            Welcome to the AWS Key Management SDK Basics scenario.

            This program demonstrates how to interact with AWS Key Management
using the AWS SDK for Java (v2).
            The AWS Key Management Service (KMS) is a secure and highly available
service that allows you to create
            and manage AWS KMS keys and control their use across a wide range of
AWS services and applications.
            KMS provides a centralized and unified approach to managing
encryption keys, making it easier to meet your
            data protection and regulatory compliance requirements.

            This Basics scenario creates two key types:

            - A symmetric encryption key is used to encrypt and decrypt data.
            - An asymmetric key used to digitally sign data.

            Let's get started...
            """);
        waitForInputToContinue(scanner);

        try {
        // Run the methods that belong to this scenario.
        String targetKeyId = runScenario(granteePrincipal, keyDesc, policyName);
        requestDeleteResources(aliasName, targetKeyId);

        } catch (Throwable rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
```

```
            }
        }
    }

    private static String runScenario(String granteePrincipal, String keyDesc,
String policyName) throws Throwable {
        logger.info(DASHES);
        logger.info("1. Create a symmetric KMS key\n");
        logger.info("First, the program will creates a symmetric KMS key that you
can used to encrypt and decrypt data.");
        waitForInputToContinue(scanner);
        String targetKeyId;
        try {
            CompletableFuture<String> futureKeyId =
kmsActions.createKeyAsync(keyDesc);
            targetKeyId = futureKeyId.join();
            logger.info("A symmetric key was successfully created " +
targetKeyId);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("""
            2. Enable a KMS key

            By default, when the SDK creates an AWS key, it is enabled. The next
bit of code checks to
            determine if the key is enabled.
             """);
        waitForInputToContinue(scanner);
        boolean isEnabled;
        try {
            CompletableFuture<Boolean> futureIsKeyEnabled =
kmsActions.isKeyEnabledAsync(targetKeyId);
```

```
            isEnabled = futureIsKeyEnabled.join();
            logger.info("Is the key enabled? {}", isEnabled);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }

        if (!isEnabled)
            try {
                CompletableFuture<Void> future =
kmsActions.enableKeyAsync(targetKeyId);
                future.join();

            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof KmsException kmsEx) {
                    logger.info("KMS error occurred: Error message: {}, Error
code {}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                } else {
                    logger.info("An unexpected error occurred: " +
rt.getMessage());
                }
                throw cause;
            }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("3. Encrypt data using the symmetric KMS key");
        String plaintext = "Hello, AWS KMS!";
        logger.info("""
            One of the main uses of symmetric keys is to encrypt and decrypt
data.
            Next, the code encrypts the string {} with the SYMMETRIC_DEFAULT
encryption algorithm.
            """, plaintext);
        waitForInputToContinue(scanner);
        SdkBytes encryptedData;
```

```
        try {
            CompletableFuture<SdkBytes> future =
kmsActions.encryptDataAsync(targetKeyId, plaintext);
            encryptedData = future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof DisabledException kmsDisabledEx) {
                logger.info("KMS error occurred due to a disabled
key: Error message: {}, Error code {}", kmsDisabledEx.getMessage(),
kmsDisabledEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("4. Create an alias");
        logger.info("""

            The alias name should be prefixed with 'alias/'.
            The default, 'alias/dev-encryption-key'.
             """);
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<Void> future =
kmsActions.createCustomAliasAsync(targetKeyId, aliasName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof AlreadyExistsException kmsExistsEx) {
                if (kmsExistsEx.getMessage().contains("already exists")) {
                    logger.info("The alias '" + aliasName + "' already exists.
Moving on...");
                }
            } else {
                logger.error("An unexpected error occurred: " + rt.getMessage(),
rt);
                deleteKey(targetKeyId);
```

```
                throw cause;
            }
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("5. List all of your aliases");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Object> future = kmsActions.listAllAliasesAsync();
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("6. Enable automatic rotation of the KMS key");
        logger.info("""

            By default, when the SDK enables automatic rotation of a KMS key,
            KMS rotates the key material of the KMS key one year (approximately
365 days) from the enable date and every year
            thereafter.
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<EnableKeyRotationResponse> future =
kmsActions.enableKeyRotationAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
```

```
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("""
            7. Create a grant

            A grant is a policy instrument that allows Amazon Web Services
principals to use KMS keys.
            It also can allow them to view a KMS key (DescribeKey) and create and
manage grants.
            When authorizing access to a KMS key, grants are considered along
with key policies and IAM policies.
            """);

        waitForInputToContinue(scanner);
        String grantId = null;
        try {
            CompletableFuture<String> futureGrantId =
kmsActions.grantKeyAsync(targetKeyId, granteePrincipal);
            grantId = futureGrantId.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);
```

```
        logger.info(DASHES);
        logger.info("8. List grants for the KMS key");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Object> future =
kmsActions.displayGrantIdsAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("9. Revoke the grant");
        logger.info("""
            The revocation of a grant immediately removes the permissions and
access that the grant had provided.
            This means that any principal (user, role, or service) that was
granted access to perform specific
            KMS operations on a KMS key will no longer be able to perform those
operations.
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<RevokeGrantResponse> future =
kmsActions.revokeKeyGrantAsync(targetKeyId, grantId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                if (kmsEx.getMessage().contains("Grant does not exist")) {
                    logger.info("The grant ID '" + grantId + "' does not exist.
Moving on...");
```

```
                } else {
                    logger.info("KMS error occurred: Error message: {}, Error
code {}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                    throw cause;
                }
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
                deleteAliasName(aliasName);
                deleteKey(targetKeyId);
                throw cause;
            }
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("10. Decrypt the data\n");
        logger.info("""
            Lets decrypt the data that was encrypted in an early step.
            The code uses the same key to decrypt the string that we encrypted
earlier in the program.
            """);
        waitForInputToContinue(scanner);
        String decryptedData = "";
        try {
            CompletableFuture<String> future =
kmsActions.decryptDataAsync(encryptedData, targetKeyId);
            decryptedData = future.join();
            logger.info("Decrypted data: " + decryptedData);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        logger.info("Decrypted text is: " + decryptedData);
        waitForInputToContinue(scanner);
```

```
        logger.info(DASHES);
        logger.info("11. Replace a key policy\n");
        logger.info("""
            A key policy is a resource policy for a KMS key. Key policies are the
primary way to control
            access to KMS keys. Every KMS key must have exactly one key policy.
The statements in the key policy
            determine who has permission to use the KMS key and how they can use
it.
            You can also use IAM policies and grants to control access to the KMS
key, but every KMS key
            must have a key policy.

            By default, when you create a key by using the SDK, a policy is
created that
            gives the AWS account that owns the KMS key full access to the KMS
key.

            Let's try to replace the automatically created policy with the
following policy.

                "Version":"2012-10-17",
                "Statement": [{
                "Effect": "Allow",
                "Principal": {"AWS": "arn:aws:iam::0000000000:root"},
                "Action": "kms:*",
                "Resource": "*"
                }]
            """);

        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Boolean> future =
kmsActions.replacePolicyAsync(targetKeyId, policyName, accountId);
            boolean success = future.join();
            if (success) {
                logger.info("Key policy replacement succeeded.");
            } else {
                logger.error("Key policy replacement failed.");
            }

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
```

```
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("12. Get the key policy\n");
        logger.info("The next bit of code that runs gets the key policy to make
sure it exists.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future =
kmsActions.getKeyPolicyAsync(targetKeyId, policyName);
            String policy = future.join();
            if (!policy.isEmpty()) {
                logger.info("Retrieved policy: " + policy);
            }

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("13. Create an asymmetric KMS key and sign your data\n");
        logger.info("""
             Signing your data with an AWS key can provide several benefits that
make it an attractive option
```

```
            for your data signing needs. By using an AWS KMS key, you can
leverage the
            security controls and compliance features provided by AWS,
            which can help you meet various regulatory requirements and enhance
the overall security posture
            of your organization.
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Boolean> future = kmsActions.signVerifyDataAsync();
            boolean success = future.join();
            if (success) {
                logger.info("Sign and verify data operation succeeded.");
            } else {
                logger.error("Sign and verify data operation failed.");
            }

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("14. Tag your symmetric KMS Key\n");
        logger.info("""
            By using tags, you can improve the overall management, security, and
governance of your
            KMS keys, making it easier to organize, track, and control access to
your encrypted data within
            your AWS environment
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
kmsActions.tagKMSKeyAsync(targetKeyId);
```

```
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
 {}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);
        return targetKeyId;
    }

    // Deletes KMS resources with user input.
    private static void requestDeleteResources(String aliasName, String
 targetKeyId) {
        logger.info(DASHES);
        logger.info("15. Schedule the deletion of the KMS key\n");
        logger.info("""
            By default, KMS applies a waiting period of 30 days,
            but you can specify a waiting period of 7-30 days. When this
 operation is successful,
            the key state of the KMS key changes to PendingDeletion and the key
 can't be used in any
            cryptographic operations. It remains in this state for the duration
 of the waiting period.

            Deleting a KMS key is a destructive and potentially dangerous
 operation. When a KMS key is deleted,
            all data that was encrypted under the KMS key is unrecoverable.
            """);
        logger.info("Would you like to delete the Key Management resources? (y/
n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            logger.info("You selected to delete the AWS KMS resources.");
            waitForInputToContinue(scanner);
            try {
```

```
                CompletableFuture<Void> future =
kmsActions.deleteSpecificAliasAsync(aliasName);
                future.join();

            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof KmsException kmsEx) {
                    logger.info("KMS error occurred: Error message: {}, Error
code {}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                } else {
                    logger.info("An unexpected error occurred: " +
rt.getMessage());
                }
            }
            waitForInputToContinue(scanner);
            try {
                CompletableFuture<Void> future =
kmsActions.disableKeyAsync(targetKeyId);
                future.join();

            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof KmsException kmsEx) {
                    logger.info("KMS error occurred: Error message: {}, Error
code {}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                } else {
                    logger.info("An unexpected error occurred: " +
rt.getMessage());
                }
            }

            try {
                CompletableFuture<Void> future =
kmsActions.deleteKeyAsync(targetKeyId);
                future.join();

            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof KmsException kmsEx) {
                    logger.info("KMS error occurred: Error message: {}, Error
code {}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                } else {
                    logger.info("An unexpected error occurred: " +
rt.getMessage());
```

```
                }
            }

        } else {
            logger.info("The Key Management resources will not be deleted");
        }

        logger.info(DASHES);
        logger.info("This concludes the AWS Key Management SDK scenario");
        logger.info(DASHES);
    }

    // This method is invoked from Exceptions to clean up the resources.
    private static void deleteKey(String targetKeyId) {
        try {
            CompletableFuture<Void> future =
kmsActions.disableKeyAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }

        try {
            CompletableFuture<Void> future =
kmsActions.deleteKeyAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
    }
```

```
    // This method is invoked from Exceptions to clean up the resources.
    private static void deleteAliasName(String aliasName) {
        try {
            CompletableFuture<Void> future =
 kmsActions.deleteSpecificAliasAsync(aliasName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
 {}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

KMS 작업을 래핑하는 클래스를 정의합니다.

```
public class KMSActions {
    private static final Logger logger =
 LoggerFactory.getLogger(KMSActions.class);
```

```
    private static KmsAsyncClient kmsAsyncClient;

    /**
     * Retrieves an asynchronous AWS Key Management Service (KMS) client.
     * <p>
     * This method creates and returns a singleton instance of the KMS async
client, with the following configurations:
     * <ul>
     *   <li>Max concurrency: 100</li>
     *   <li>Connection timeout: 60 seconds</li>
     *   <li>Read timeout: 60 seconds</li>
     *   <li>Write timeout: 60 seconds</li>
     *   <li>API call timeout: 2 minutes</li>
     *   <li>API call attempt timeout: 90 seconds</li>
     *   <li>Retry policy: up to 3 retries</li>
     *   <li>Credentials provider: environment variable credentials provider</li>
     * </ul>
     * <p>
     * If the client instance has already been created, it is returned instead of
creating a new one.
     *
     * @return the KMS async client instance
     */
    private static KmsAsyncClient getAsyncClient() {
        if (kmsAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryPolicy(RetryPolicy.builder()
                    .numRetries(3)
                    .build())
                .build();

            kmsAsyncClient = KmsAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
```

```
                .build();
        }
        return kmsAsyncClient;
    }

    /**
     * Creates a new symmetric encryption key asynchronously.
     *
     * @param keyDesc the description of the key to be created
     * @return a {@link CompletableFuture} that completes with the ID of the
newly created key
     * @throws RuntimeException if an error occurs while creating the key
     */
    public CompletableFuture<String> createKeyAsync(String keyDesc) {
        CreateKeyRequest keyRequest = CreateKeyRequest.builder()
            .description(keyDesc)
            .keySpec(KeySpec.SYMMETRIC_DEFAULT)
            .keyUsage(KeyUsageType.ENCRYPT_DECRYPT)
            .build();

        return getAsyncClient().createKey(keyRequest)
            .thenApply(resp -> resp.keyMetadata().keyId())
            .exceptionally(ex -> {
                throw new RuntimeException("An error occurred while creating the
key: " + ex.getMessage(), ex);
            });
    }

    /**
     * Asynchronously checks if a specified key is enabled.
     *
     * @param keyId the ID of the key to check
     * @return a {@link CompletableFuture} that, when completed, indicates
whether the key is enabled or not
     *
     * @throws RuntimeException if an exception occurs while checking the key
state
     */
    public CompletableFuture<Boolean> isKeyEnabledAsync(String keyId) {
        DescribeKeyRequest keyRequest = DescribeKeyRequest.builder()
            .keyId(keyId)
            .build();
```

```
        CompletableFuture<DescribeKeyResponse> responseFuture =
getAsyncClient().describeKey(keyRequest);
        return responseFuture.whenComplete((resp, ex) -> {
            if (resp != null) {
                KeyState keyState = resp.keyMetadata().keyState();
                if (keyState == KeyState.ENABLED) {
                    logger.info("The key is enabled.");
                } else {
                    logger.info("The key is not enabled. Key state: {}",
keyState);
                }
            } else {
                throw new RuntimeException(ex);
            }
        }).thenApply(resp -> resp.keyMetadata().keyState() == KeyState.ENABLED);
    }

    /**
     * Asynchronously enables the specified key.
     *
     * @param keyId the ID of the key to enable
     * @return a {@link CompletableFuture} that completes when the key has been
enabled
     */
    public CompletableFuture<Void> enableKeyAsync(String keyId) {
        EnableKeyRequest enableKeyRequest = EnableKeyRequest.builder()
            .keyId(keyId)
            .build();

        CompletableFuture<EnableKeyResponse> responseFuture =
getAsyncClient().enableKey(enableKeyRequest);
        responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {
                logger.info("Key with ID [{}] has been enabled.", keyId);
            } else {
                if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("KMS error occurred while enabling
key: " + kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred
while enabling key: " + exception.getMessage(), exception);
                }
            }
        });
```

```
        return responseFuture.thenApply(response -> null);
    }

    /**
     * Encrypts the given text asynchronously using the specified KMS client and
key ID.
     *
     * @param keyId the ID of the KMS key to use for encryption
     * @param text the text to encrypt
     * @return a CompletableFuture that completes with the encrypted data as an
SdkBytes object
     */
    public CompletableFuture<SdkBytes> encryptDataAsync(String keyId, String
text) {
        SdkBytes myBytes = SdkBytes.fromUtf8String(text);
        EncryptRequest encryptRequest = EncryptRequest.builder()
            .keyId(keyId)
            .plaintext(myBytes)
            .build();

        CompletableFuture<EncryptResponse> responseFuture =
getAsyncClient().encrypt(encryptRequest).toCompletableFuture();
        return responseFuture.whenComplete((response, ex) -> {
            if (response != null) {
                String algorithm = response.encryptionAlgorithm().toString();
                logger.info("The string was encrypted with algorithm {}.",
algorithm);
            } else {
                throw new RuntimeException(ex);
            }
        }).thenApply(EncryptResponse::ciphertextBlob);
    }

    /**
     * Creates a custom alias for the specified target key asynchronously.
     *
     * @param targetKeyId the ID of the target key for the alias
     * @param aliasName   the name of the alias to create
     * @return a {@link CompletableFuture} that completes when the alias creation
operation is finished
     */
    public CompletableFuture<Void> createCustomAliasAsync(String targetKeyId,
String aliasName) {
```

```
        CreateAliasRequest aliasRequest = CreateAliasRequest.builder()
            .aliasName(aliasName)
            .targetKeyId(targetKeyId)
            .build();

        CompletableFuture<CreateAliasResponse> responseFuture =
getAsyncClient().createAlias(aliasRequest);
        responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {
                logger.info("{} was successfully created.", aliasName);
            } else {
                if (exception instanceof ResourceExistsException) {
                    logger.info("Alias [{}] already exists. Moving on...",
aliasName);
                } else if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("KMS error occurred while creating
alias: " + kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred
while creating alias: " + exception.getMessage(), exception);
                }
            }
        });

        return responseFuture.thenApply(response -> null);
    }

    /**
     * Asynchronously lists all the aliases in the current AWS account.
     *
     * @return a {@link CompletableFuture} that completes when the list of
aliases has been processed
     */
    public CompletableFuture<Object> listAllAliasesAsync() {
        ListAliasesRequest aliasesRequest = ListAliasesRequest.builder()
            .limit(15)
            .build();

        ListAliasesPublisher paginator =
getAsyncClient().listAliasesPaginator(aliasesRequest);
        return paginator.subscribe(response -> {
                response.aliases().forEach(alias ->
                    logger.info("The alias name is: " + alias.aliasName())
                );
```

```
            })
            .thenApply(v -> null)
            .exceptionally(ex -> {
                if (ex.getCause() instanceof KmsException) {
                    KmsException e = (KmsException) ex.getCause();
                    throw new RuntimeException("A KMS exception occurred: " +
e.getMessage());
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
ex.getMessage());
                }
            });
   }

   /**
    * Enables key rotation asynchronously for the specified key ID.
    *
    * @param keyId the ID of the key for which to enable key rotation
    * @return a CompletableFuture that represents the asynchronous operation of
enabling key rotation
    * @throws RuntimeException if there was an error enabling key rotation,
either due to a KMS exception or an unexpected error
    */
   public CompletableFuture<EnableKeyRotationResponse>
enableKeyRotationAsync(String keyId) {
       EnableKeyRotationRequest enableKeyRotationRequest =
EnableKeyRotationRequest.builder()
            .keyId(keyId)
            .build();

       CompletableFuture<EnableKeyRotationResponse> responseFuture =
getAsyncClient().enableKeyRotation(enableKeyRotationRequest);
       responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {
                logger.info("Key rotation has been enabled for key with id [{}]",
keyId);
            } else {
                if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("Failed to enable key rotation: "
+ kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
exception.getMessage(), exception);
                }
```

```
            }
        });

        return responseFuture;
    }

    /**
     * Grants permissions to a specified principal on a customer master key (CMK)
asynchronously.
     *
     * @param keyId             The unique identifier for the customer master key
(CMK) that the grant applies to.
     * @param granteePrincipal  The principal that is given permission to perform
the operations that the grant permits on the CMK.
     * @return A {@link CompletableFuture} that, when completed, contains the ID
of the created grant.
     * @throws RuntimeException If an error occurs during the grant creation
process.
     */
    public CompletableFuture<String> grantKeyAsync(String keyId, String
granteePrincipal) {
        List<GrantOperation> grantPermissions = List.of(
            GrantOperation.ENCRYPT,
            GrantOperation.DECRYPT,
            GrantOperation.DESCRIBE_KEY
        );

        CreateGrantRequest grantRequest = CreateGrantRequest.builder()
            .keyId(keyId)
            .name("grant1")
            .granteePrincipal(granteePrincipal)
            .operations(grantPermissions)
            .build();

        CompletableFuture<CreateGrantResponse> responseFuture =
getAsyncClient().createGrant(grantRequest);
        responseFuture.whenComplete((response, ex) -> {
            if (ex == null) {
                logger.info("Grant created successfully with ID: " +
response.grantId());
            } else {
                if (ex instanceof KmsException kmsEx) {
                    throw new RuntimeException("Failed to create grant: " +
kmsEx.getMessage(), kmsEx);
```

```
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
ex.getMessage(), ex);
                }
            }
        });

        return responseFuture.thenApply(CreateGrantResponse::grantId);
    }

    /**
     * Asynchronously displays the grant IDs for the specified key ID.
     *
     * @param keyId the ID of the AWS KMS key for which to list the grants
     * @return a {@link CompletableFuture} that, when completed, will be null
if the operation succeeded, or will throw a {@link RuntimeException} if the
operation failed
     * @throws RuntimeException if there was an error listing the grants, either
due to an {@link KmsException} or an unexpected error
     */
    public CompletableFuture<Object> displayGrantIdsAsync(String keyId) {
        ListGrantsRequest grantsRequest = ListGrantsRequest.builder()
            .keyId(keyId)
            .limit(15)
            .build();

        ListGrantsPublisher paginator =
getAsyncClient().listGrantsPaginator(grantsRequest);
        return paginator.subscribe(response -> {
                response.grants().forEach(grant -> {
                    logger.info("The grant Id is: " + grant.grantId());
                });
            })
            .thenApply(v -> null)
            .exceptionally(ex -> {
                Throwable cause = ex.getCause();
                if (cause instanceof KmsException) {
                    throw new RuntimeException("Failed to list grants: " +
cause.getMessage(), cause);
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
cause.getMessage(), cause);
                }
            });
```

```
    }

    /**
     * Revokes a grant for the specified AWS KMS key asynchronously.
     *
     * @param keyId   The ID or key ARN of the AWS KMS key.
     * @param grantId The identifier of the grant to be revoked.
     * @return A {@link CompletableFuture} representing the asynchronous
operation of revoking the grant.
     *         The {@link CompletableFuture} will complete with a {@link
RevokeGrantResponse} object
     *         if the operation is successful, or with a {@code null} value if an
error occurs.
     */
    public CompletableFuture<RevokeGrantResponse> revokeKeyGrantAsync(String
keyId, String grantId) {
        RevokeGrantRequest grantRequest = RevokeGrantRequest.builder()
            .keyId(keyId)
            .grantId(grantId)
            .build();

        CompletableFuture<RevokeGrantResponse> responseFuture =
getAsyncClient().revokeGrant(grantRequest);
        responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {
                logger.info("Grant ID: [" + grantId + "] was successfully
revoked!");
            } else {
                if (exception instanceof KmsException kmsEx) {
                    if (kmsEx.getMessage().contains("Grant does not exist")) {
                        logger.info("The grant ID '" + grantId + "' does not
exist. Moving on...");
                    } else {
                        throw new RuntimeException("KMS error occurred: " +
kmsEx.getMessage(), kmsEx);
                    }
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
exception.getMessage(), exception);
                }
            }
        });

        return responseFuture;
```

```
    }


    /**
     * Asynchronously decrypts the given encrypted data using the specified key
ID.
     *
     * @param encryptedData The encrypted data to be decrypted.
     * @param keyId The ID of the key to be used for decryption.
     * @return A CompletableFuture that, when completed, will contain the
decrypted data as a String.
     *         If an error occurs during the decryption process, the
CompletableFuture will complete
     *         exceptionally with the error, and the method will return an empty
String.
     */
    public CompletableFuture<String> decryptDataAsync(SdkBytes encryptedData,
String keyId) {
        DecryptRequest decryptRequest = DecryptRequest.builder()
            .ciphertextBlob(encryptedData)
            .keyId(keyId)
            .build();

        CompletableFuture<DecryptResponse> responseFuture =
getAsyncClient().decrypt(decryptRequest);
        responseFuture.whenComplete((decryptResponse, exception) -> {
            if (exception == null) {
                logger.info("Data decrypted successfully for key ID: " + keyId);
            } else {
                if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("KMS error occurred while
decrypting data: " + kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred
while decrypting data: " + exception.getMessage(), exception);
                }
            }
        });

        return responseFuture.thenApply(decryptResponse ->
decryptResponse.plaintext().asString(StandardCharsets.UTF_8));
    }

    /**
```

```
     * Asynchronously replaces the policy for the specified KMS key.
     *
     * @param keyId       the ID of the KMS key to replace the policy for
     * @param policyName  the name of the policy to be replaced
     * @param accountId   the AWS account ID to be used in the policy
     * @return a {@link CompletableFuture} that completes with a boolean
indicating
     *         whether the policy replacement was successful or not
     */
    public CompletableFuture<Boolean> replacePolicyAsync(String keyId, String
policyName, String accountId) {
        String policy = """
    {
      "Version":"2012-10-17",
      "Statement": [{
        "Effect": "Allow",
        "Principal": {"AWS": "arn:aws:iam::%s:root"},
        "Action": "kms:*",
        "Resource": "*"
      }]
    }
    """.formatted(accountId);

        PutKeyPolicyRequest keyPolicyRequest = PutKeyPolicyRequest.builder()
            .keyId(keyId)
            .policyName(policyName)
            .policy(policy)
            .build();

        // First, get the current policy to check if it exists
        return getAsyncClient().getKeyPolicy(r ->
r.keyId(keyId).policyName(policyName))
            .thenCompose(response -> {
                logger.info("Current policy exists. Replacing it...");
                return getAsyncClient().putKeyPolicy(keyPolicyRequest);
            })
            .thenApply(putPolicyResponse -> {
                logger.info("The key policy has been replaced.");
                return true;
            })
            .exceptionally(throwable -> {
                if (throwable.getCause() instanceof LimitExceededException) {
                    logger.error("Cannot replace policy, as only one policy is
allowed per key.");
```

```
                    return false;
                }
                throw new RuntimeException("Error replacing policy", throwable);
            });
    }


    /**
     * Asynchronously retrieves the key policy for the specified key ID and
policy name.
     *
     * @param keyId       the ID of the AWS KMS key for which to retrieve the
policy
     * @param policyName the name of the key policy to retrieve
     * @return a {@link CompletableFuture} that, when completed, contains the key
policy as a {@link String}
     */
    public CompletableFuture<String> getKeyPolicyAsync(String keyId, String
policyName) {
        GetKeyPolicyRequest policyRequest = GetKeyPolicyRequest.builder()
            .keyId(keyId)
            .policyName(policyName)
            .build();

        return getAsyncClient().getKeyPolicy(policyRequest)
            .thenApply(response -> {
                String policy = response.policy();
                logger.info("The response is: " + policy);
                return policy;
            })
            .exceptionally(ex -> {
                throw new RuntimeException("Failed to get key policy", ex);
            });
    }

    /**
     * Asynchronously signs and verifies data using AWS KMS.
     *
     * <p>The method performs the following steps:
     * <ol>
     *     <li>Creates an AWS KMS key with the specified key spec, key usage, and
origin.</li>
     *     <li>Signs the provided message using the created KMS key and the
RSASSA-PSS-SHA-256 algorithm.</li>
```

```
     *     <li>Verifies the signature of the message using the created KMS key
and the RSASSA-PSS-SHA-256 algorithm.</li>
     * </ol>
     *
     * @return a {@link CompletableFuture} that completes with the result of the
signature verification,
     *         {@code true} if the signature is valid, {@code false} otherwise.
     * @throws KmsException if any error occurs during the KMS operations.
     * @throws RuntimeException if an unexpected error occurs.
     */
    public CompletableFuture<Boolean> signVerifyDataAsync() {
        String signMessage = "Here is the message that will be digitally signed";

        // Create an AWS KMS key used to digitally sign data.
        CreateKeyRequest createKeyRequest = CreateKeyRequest.builder()
            .keySpec(KeySpec.RSA_2048)
            .keyUsage(KeyUsageType.SIGN_VERIFY)
            .origin(OriginType.AWS_KMS)
            .build();

        return getAsyncClient().createKey(createKeyRequest)
            .thenCompose(createKeyResponse -> {
                String keyId = createKeyResponse.keyMetadata().keyId();

                SdkBytes messageBytes = SdkBytes.fromString(signMessage,
Charset.defaultCharset());
                SignRequest signRequest = SignRequest.builder()
                    .keyId(keyId)
                    .message(messageBytes)
                    .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
                    .build();

                return getAsyncClient().sign(signRequest)
                    .thenCompose(signResponse -> {
                        byte[] signedBytes =
signResponse.signature().asByteArray();

                        VerifyRequest verifyRequest = VerifyRequest.builder()
                            .keyId(keyId)

.message(SdkBytes.fromByteArray(signMessage.getBytes(Charset.defaultCharset())))

.signature(SdkBytes.fromByteBuffer(ByteBuffer.wrap(signedBytes)))
```

```
.signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
                            .build();

                        return getAsyncClient().verify(verifyRequest)
                            .thenApply(verifyResponse -> {
                                return (boolean) verifyResponse.signatureValid();
                            });
                    });
            })
            .exceptionally(throwable -> {
               throw new RuntimeException("Failed to sign or verify data",
throwable);
            });
    }

    /**
     * Asynchronously tags a KMS key with a specific tag.
     *
     * @param keyId the ID of the KMS key to be tagged
     * @return a {@link CompletableFuture} that completes when the tagging
operation is finished
     */
    public CompletableFuture<Void> tagKMSKeyAsync(String keyId) {
        Tag tag = Tag.builder()
            .tagKey("Environment")
            .tagValue("Production")
            .build();

        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .keyId(keyId)
            .tags(tag)
            .build();

        return getAsyncClient().tagResource(tagResourceRequest)
            .thenRun(() -> {
                logger.info("{} key was tagged", keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to tag the KMS key",
throwable);
            });
    }
```

```
    /**
     * Deletes a specific KMS alias asynchronously.
     *
     * @param aliasName the name of the alias to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous
operation of deleting the specified alias
     */
    public CompletableFuture<Void> deleteSpecificAliasAsync(String aliasName) {
        DeleteAliasRequest deleteAliasRequest = DeleteAliasRequest.builder()
            .aliasName(aliasName)
            .build();

        return getAsyncClient().deleteAlias(deleteAliasRequest)
            .thenRun(() -> {
                logger.info("Alias {} has been deleted successfully", aliasName);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to delete alias: " +
aliasName, throwable);
            });
    }

    /**
     * Asynchronously disables the specified AWS Key Management Service (KMS)
key.
     *
     * @param keyId the ID or Amazon Resource Name (ARN) of the KMS key to be
disabled
     * @return a CompletableFuture that, when completed, indicates that the key
has been disabled successfully
     */
    public CompletableFuture<Void> disableKeyAsync(String keyId) {
        DisableKeyRequest keyRequest = DisableKeyRequest.builder()
            .keyId(keyId)
            .build();

        return getAsyncClient().disableKey(keyRequest)
            .thenRun(() -> {
                logger.info("Key {} has been disabled successfully",keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to disable key: " + keyId,
throwable);
            });
```

```
    }

    /**
     * Deletes a KMS key asynchronously.
     *
     * <p><strong>Warning:</strong> Deleting a KMS key is a destructive and
 potentially dangerous operation.
     * When a KMS key is deleted, all data that was encrypted under the KMS key
 becomes unrecoverable.
     * This means that any files, databases, or other data that were encrypted
 using the deleted KMS key
     * will become permanently inaccessible. Exercise extreme caution when
 deleting KMS keys.</p>
     *
     * @param keyId the ID of the KMS key to delete
     * @return a {@link CompletableFuture} that completes when the key deletion
 is scheduled
     */
    public CompletableFuture<Void> deleteKeyAsync(String keyId) {
        ScheduleKeyDeletionRequest deletionRequest =
 ScheduleKeyDeletionRequest.builder()
            .keyId(keyId)
            .pendingWindowInDays(7)
            .build();

        return getAsyncClient().scheduleKeyDeletion(deletionRequest)
            .thenRun(() -> {
                logger.info("Key {} will be deleted in 7 days", keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to schedule key deletion for
 key ID: " + keyId, throwable);
            });
    }


    public String getAccountId(){
        try (StsClient stsClient = StsClient.create()){
            GetCallerIdentityResponse callerIdentity =
 stsClient.getCallerIdentity();
            return callerIdentity.account();
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - CreateAlias
  - CreateGrant
  - CreateKey
  - Decrypt
  - DescribeKey
  - DisableKey
  - EnableKey
  - 암호화
  - GetKeyPolicy
  - ListAliases
  - ListGrants
  - ListKeys
  - RevokeGrant
  - ScheduleKeyDeletion
  - Sign
  - TagResource

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        echo "\n";
        echo "-------------------------------------\n";
        echo <<<WELCOME
Welcome to the AWS Key Management Service SDK Basics scenario.
```

```
This program demonstrates how to interact with AWS Key Management Service using
 the AWS SDK for PHP (v3).
The AWS Key Management Service (KMS) is a secure and highly available service
 that allows you to create
and manage AWS KMS keys and control their use across a wide range of AWS services
 and applications.
KMS provides a centralized and unified approach to managing encryption keys,
 making it easier to meet your
data protection and regulatory compliance requirements.

This KMS Basics scenario creates two key types:
- A symmetric encryption key is used to encrypt and decrypt data.
- An asymmetric key used to digitally sign data.

Let's get started...\n
WELCOME;
        echo "-------------------------------------\n";
        $this->pressEnter();

        $this->kmsClient = new KmsClient([]);
        // Initialize the KmsService class with the client. This allows you to
 override any defaults in the client before giving it to the service class.
        $this->kmsService = new KmsService($this->kmsClient);

        // 1. Create a symmetric KMS key.
        echo "\n";
        echo "1. Create a symmetric KMS key.\n";
        echo "First, we will create a symmetric KMS key that is used to encrypt
 and decrypt data by invoking createKey().\n";
        $this->pressEnter();

        $key = $this->kmsService->createKey();
        $this->resources['symmetricKey'] = $key['KeyId'];
        echo "Created a customer key with ARN {$key['Arn']}.\n";
        $this->pressEnter();

        // 2. Enable a KMS key.
        echo "\n";
        echo "2. Enable a KMS key.\n";
        echo "By default when you create an AWS key, it is enabled. The code
 checks to
determine if the key is enabled. If it is not enabled, the code enables it.\n";
        $this->pressEnter();
```

```
        $keyInfo = $this->kmsService->describeKey($key['KeyId']);
        if(!$keyInfo['Enabled']){
            echo "The key was not enabled, so we will enable it.\n";
            $this->pressEnter();
            $this->kmsService->enableKey($key['KeyId']);
            echo "The key was successfully enabled.\n";
        }else{
            echo "The key was already enabled, so there was no need to enable it.
\n";
        }
        $this->pressEnter();

        // 3. Encrypt data using the symmetric KMS key.
        echo "\n";
        echo "3. Encrypt data using the symmetric KMS key.\n";
        echo "One of the main uses of symmetric keys is to encrypt and decrypt
 data.\n";
        echo "Next, we'll encrypt the string 'Hello, AWS KMS!' with the
 SYMMETRIC_DEFAULT encryption algorithm.\n";
        $this->pressEnter();
        $text = "Hello, AWS KMS!";
        $encryption = $this->kmsService->encrypt($key['KeyId'], $text);
        echo "The plaintext data was successfully encrypted with the algorithm:
 {$encryption['EncryptionAlgorithm']}.\n";
        $this->pressEnter();

        // 4. Create an alias.
        echo "\n";
        echo "4. Create an alias.\n";
        $aliasInput = testable_readline("Please enter an alias prefixed with
 \"alias/\" or press enter to use a default value: ");
        if($aliasInput == ""){
            $aliasInput = "alias/dev-encryption-key";
        }
        $this->kmsService->createAlias($key['KeyId'], $aliasInput);
        $this->resources['alias'] = $aliasInput;
        echo "The alias \"$aliasInput\" was successfully created.\n";
        $this->pressEnter();

        // 5. List all of your aliases.
        $aliasPageSize = 10;
        echo "\n";
        echo "5. List all of your aliases, up to $aliasPageSize.\n";
```

```
        $this->pressEnter();
        $aliasPaginator = $this->kmsService->listAliases();
        foreach($aliasPaginator as $pages){
            foreach($pages['Aliases'] as $alias){
                echo $alias['AliasName'] . "\n";
            }
            break;
        }
        $this->pressEnter();

        // 6. Enable automatic rotation of the KMS key.
        echo "\n";
        echo "6. Enable automatic rotation of the KMS key.\n";
        echo "By default, when the SDK enables automatic rotation of a KMS key,
KMS rotates the key material of the KMS key one year (approximately 365 days)
 from the enable date and every year
thereafter.";
        $this->pressEnter();
        $this->kmsService->enableKeyRotation($key['KeyId']);
        echo "The key's rotation was successfully set for key:
 {$key['KeyId']}\n";
        $this->pressEnter();

        // 7. Create a grant.
        echo "7. Create a grant.\n";
        echo "\n";
        echo "A grant is a policy instrument that allows Amazon Web Services
 principals to use KMS keys.
It also can allow them to view a KMS key (DescribeKey) and create and manage
 grants.
When authorizing access to a KMS key, grants are considered along with key
 policies and IAM policies.\n";
        $granteeARN = testable_readline("Please enter the Amazon Resource Name
 (ARN) of an Amazon Web Services principal. Valid principals include Amazon
 Web Services accounts, IAM users, IAM roles, federated users, and assumed
 role users. For help with the ARN syntax for a principal, see IAM ARNs in the
 Identity and Access Management User Guide. \nTo skip this step, press enter
 without any other values: ");
        if($granteeARN){
            $operations = [
                "ENCRYPT",
                "DECRYPT",
                "DESCRIBE_KEY",
            ];
```

```
            $grant = $this->kmsService->createGrant($key['KeyId'], $granteeARN,
$operations);
            echo "The grant Id is: {$grant['GrantId']}\n";
        }else{
            echo "Steps 7, 8, and 9 will be skipped.\n";
        }
        $this->pressEnter();

        // 8. List grants for the KMS key.
        if($granteeARN){
            echo "8. List grants for the KMS key.\n\n";
            $grantsPaginator = $this->kmsService->listGrants($key['KeyId']);
            foreach($grantsPaginator as $page){
                foreach($page['Grants'] as $grant){
                    echo $grant['GrantId'] . "\n";
                }
            }
        }else{
            echo "Skipping step 8...\n";
        }
        $this->pressEnter();

        // 9. Revoke the grant.
        if($granteeARN) {
            echo "\n";
            echo "9. Revoke the grant.\n";
            $this->pressEnter();
            $this->kmsService->revokeGrant($grant['GrantId'], $keyInfo['KeyId']);
            echo "{$grant['GrantId']} was successfully revoked!\n";
        }else{
            echo "Skipping step 9...\n";
        }
        $this->pressEnter();

        // 10. Decrypt the data.
        echo "\n";
        echo "10. Decrypt the data.\n";
        echo "Let's decrypt the data that was encrypted before.\n";
        echo "We'll use the same key to decrypt the string that we encrypted
earlier in the program.\n";
        $this->pressEnter();
        $decryption = $this->kmsService->decrypt($keyInfo['KeyId'],
$encryption['CiphertextBlob'], $encryption['EncryptionAlgorithm']);
        echo "The decrypted text is: {$decryption['Plaintext']}\n";
```

```
        $this->pressEnter();

        // 11. Replace a Key Policy.
        echo "\n";
        echo "11. Replace a Key Policy.\n";
        echo "A key policy is a resource policy for a KMS key. Key policies are
 the primary way to control access to KMS keys.\n";
        echo "Every KMS key must have exactly one key policy. The statements in
 the key policy determine who has permission to use the KMS key and how they can
 use it.\n";
        echo " You can also use IAM policies and grants to control access to the
 KMS key, but every KMS key must have a key policy.\n";
        echo "We will replace the key's policy with a new one:\n";
        $stsClient = new StsClient([]);
        $result = $stsClient->getCallerIdentity();
        $accountId = $result['Account'];
        $keyPolicy = <<< KEYPOLICY
{
    "Version":"2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {"AWS": "arn:aws:iam::$accountId:root"},
        "Action": "kms:*",
        "Resource": "*"
    }]
}
KEYPOLICY;
        echo $keyPolicy;
        $this->pressEnter();
        $this->kmsService->putKeyPolicy($keyInfo['KeyId'], $keyPolicy);
        echo "The Key Policy was successfully replaced!\n";
        $this->pressEnter();

        // 12. Retrieve the key policy.
        echo "\n";
        echo "12. Retrieve the key policy.\n";
        echo "Let's get some information about the new policy and print it to the
 screen.\n";
        $this->pressEnter();
        $policyInfo = $this->kmsService->getKeyPolicy($keyInfo['KeyId']);
        echo "We got the info! Here is the policy: \n";
        echo $policyInfo['Policy'] . "\n";
        $this->pressEnter();
```

```
        // 13. Create an asymmetric KMS key and sign data.
        echo "\n";
        echo "13. Create an asymmetric KMS key and sign data.\n";
        echo "Signing your data with an AWS key can provide several benefits that
make it an attractive option for your data signing needs.\n";
        echo "By using an AWS KMS key, you can leverage the security controls and
compliance features provided by AWS, which can help you meet various regulatory
requirements and enhance the overall security posture of your organization.\n";
        echo "First we'll create the asymmetric key.\n";
        $this->pressEnter();
        $keySpec = "RSA_2048";
        $keyUsage = "SIGN_VERIFY";
        $asymmetricKey = $this->kmsService->createKey($keySpec, $keyUsage);
        $this->resources['asymmetricKey'] = $asymmetricKey['KeyId'];
        echo "Created the key with ID: {$asymmetricKey['KeyId']}\n";
        echo "Next, we'll sign the data.\n";
        $this->pressEnter();
        $algorithm = "RSASSA_PSS_SHA_256";
        $sign = $this->kmsService->sign($asymmetricKey['KeyId'], $text,
$algorithm);
        $verify = $this->kmsService->verify($asymmetricKey['KeyId'], $text,
$sign['Signature'], $algorithm);
        echo "Signature verification result: {$sign['signature']}\n";
        $this->pressEnter();

        // 14. Tag the symmetric KMS key.
        echo "\n";
        echo "14. Tag the symmetric KMS key.\n";
        echo "By using tags, you can improve the overall management, security,
and governance of your KMS keys, making it easier to organize, track, and
control access to your encrypted data within your AWS environment.\n";
        echo "Let's tag our symmetric key as Environment->Production\n";
        $this->pressEnter();
        $this->kmsService->tagResource($key['KeyId'], [
            [
                'TagKey' => "Environment",
                'TagValue' => "Production",
            ],
        ]);
        echo "The key was successfully tagged!\n";
        $this->pressEnter();

        // 15. Schedule the deletion of the KMS key
        echo "\n";
```

```
        echo "15. Schedule the deletion of the KMS key.\n";
        echo "By default, KMS applies a waiting period of 30 days, but you can
 specify a waiting period of 7-30 days.\n";
        echo "When this operation is successful, the key state of the KMS key
 changes to PendingDeletion and the key can't be used in any cryptographic
 operations.\n";
        echo "It remains in this state for the duration of the waiting period.\n
\n";

        echo "Deleting a KMS key is a destructive and potentially dangerous
 operation. When a KMS key is deleted, all data that was encrypted under the KMS
 key is unrecoverable.\n\n";

        $cleanUp = testable_readline("Would you like to delete the resources
 created during this scenario, including the keys? (y/n): ");
        if($cleanUp == "Y" || $cleanUp == "y"){
            $this->cleanUp();
        }

        echo
 "--------------------------------------------------------------------------------
\n";
        echo "This concludes the AWS Key Management SDK Basics scenario\n";
        echo
 "--------------------------------------------------------------------------------
\n";



namespace Kms;

use Aws\Kms\Exception\KmsException;
use Aws\Kms\KmsClient;
use Aws\Result;
use Aws\ResultPaginator;
use AwsUtilities\AWSServiceClass;

class KmsService extends AWSServiceClass
{

    protected KmsClient $client;
    protected bool $verbose;

    /***
```

```
     * @param KmsClient|null $client
     * @param bool $verbose
     */
    public function __construct(KmsClient $client = null, bool $verbose = false)
    {
        $this->verbose = $verbose;
        if($client){
            $this->client = $client;
            return;
        }
        $this->client = new KmsClient([]);
    }


    /***
     * @param string $keySpec
     * @param string $keyUsage
     * @param string $description
     * @return array
     */
    public function createKey(string $keySpec = "", string $keyUsage = "", string
$description = "Created by the SDK for PHP")
    {
        $parameters = ['Description' => $description];
        if($keySpec && $keyUsage){
            $parameters['KeySpec'] = $keySpec;
            $parameters['KeyUsage'] = $keyUsage;
        }
        try {
            $result = $this->client->createKey($parameters);
            return $result['KeyMetadata'];
        }catch(KmsException $caught){
            // Check for error specific to createKey operations
            if ($caught->getAwsErrorMessage() == "LimitExceededException"){
                echo "The request was rejected because a quota was exceeded. For
more information, see Quotas in the Key Management Service Developer Guide.";
            }
            throw $caught;
        }
    }



    /***
```

```
     * @param string $keyId
     * @param string $ciphertext
     * @param string $algorithm
     * @return Result
     */
    public function decrypt(string $keyId, string $ciphertext, string $algorithm
 = "SYMMETRIC_DEFAULT")
    {
        try{
            return $this->client->decrypt([
                'CiphertextBlob' => $ciphertext,
                'EncryptionAlgorithm' => $algorithm,
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem decrypting the data: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param string $text
     * @return Result
     */
    public function encrypt(string $keyId, string $text)
    {
        try {
            return $this->client->encrypt([
                'KeyId' => $keyId,
                'Plaintext' => $text,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "DisabledException"){
                echo "The request was rejected because the specified KMS key is
 not enabled.\n";
            }
            throw $caught;
        }
    }
```

```
    /***
     * @param string $keyId
     * @param int $limit
     * @return ResultPaginator
     */
    public function listAliases(string $keyId = "", int $limit = 0)
    {
        $args = [];
        if($keyId){
            $args['KeyId'] = $keyId;
        }
        if($limit){
            $args['Limit'] = $limit;
        }
        try{
            return $this->client->getPaginator("ListAliases", $args);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidMarkerException"){
                echo "The request was rejected because the marker that specifies
where pagination should next begin is not valid.\n";
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param string $alias
     * @return void
     */
    public function createAlias(string $keyId, string $alias)
    {
        try{
            $this->client->createAlias([
                'TargetKeyId' => $keyId,
                'AliasName' => $alias,
            ]);
        }catch (KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidAliasNameException"){
```

```
                echo "The request was rejected because the specified alias name
is not valid.";
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param string $granteePrincipal
     * @param array $operations
     * @param array $grantTokens
     * @return Result
     */
    public function createGrant(string $keyId, string $granteePrincipal, array
$operations, array $grantTokens = [])
    {
        $args = [
            'KeyId' => $keyId,
            'GranteePrincipal' => $granteePrincipal,
            'Operations' => $operations,
        ];
        if($grantTokens){
            $args['GrantTokens'] = $grantTokens;
        }
        try{
            return $this->client->createGrant($args);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidGrantTokenException"){
                echo "The request was rejected because the specified grant token
is not valid.\n";
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @return array
     */
```

```
    public function describeKey(string $keyId)
    {
        try {
            $result = $this->client->describeKey([
                "KeyId" => $keyId,
            ]);
            return $result['KeyMetadata'];
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
 resource could not be found.\n";
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @return void
     */
    public function disableKey(string $keyId)
    {
        try {
            $this->client->disableKey([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem disabling the key: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @return void
     */
    public function enableKey(string $keyId)
    {
        try {
```

```
            $this->client->enableKey([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
 resource could not be found.\n";
            }
            throw $caught;
        }
    }



    /***
     * @return array
     */
    public function listKeys()
    {
        try {
            $contents = [];
            $paginator = $this->client->getPaginator("ListKeys");
            foreach($paginator as $result){
                foreach ($result['Content'] as $object) {
                    $contents[] = $object;
                }
            }
            return $contents;
        }catch(KmsException $caught){
            echo "There was a problem listing the keys: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @return Result
     */
    public function listGrants(string $keyId)
    {
        try{
```

```
            return $this->client->listGrants([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "    The request was rejected because the specified entity
 or resource could not be found.\n";
            }
            throw $caught;
        }
    }


    /***
     * @param string $keyId
     * @return Result
     */
    public function getKeyPolicy(string $keyId)
    {
        try {
            return $this->client->getKeyPolicy([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem getting the key policy: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }


    /***
     * @param string $grantId
     * @param string $keyId
     * @return void
     */
    public function revokeGrant(string $grantId, string $keyId)
    {
        try{
            $this->client->revokeGrant([
                'GrantId' => $grantId,
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
```

```
            echo "There was a problem with revoking the grant: {$caught-
>getAwsErrorMessage()}.\n";
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param int $pendingWindowInDays
     * @return void
     */
    public function scheduleKeyDeletion(string $keyId, int $pendingWindowInDays =
 7)
    {
        try {
            $this->client->scheduleKeyDeletion([
                'KeyId' => $keyId,
                'PendingWindowInDays' => $pendingWindowInDays,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem scheduling the key deletion: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param array $tags
     * @return void
     */
    public function tagResource(string $keyId, array $tags)
    {
        try {
            $this->client->tagResource([
                'KeyId' => $keyId,
                'Tags' => $tags,
            ]);
        }catch(KmsException $caught){
```

```
            echo "There was a problem applying the tag(s): {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param string $message
     * @param string $algorithm
     * @return Result
     */
    public function sign(string $keyId, string $message, string $algorithm)
    {
        try {
            return $this->client->sign([
                'KeyId' => $keyId,
                'Message' => $message,
                'SigningAlgorithm' => $algorithm,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem signing the data: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param int $rotationPeriodInDays
     * @return void
     */
    public function enableKeyRotation(string $keyId, int $rotationPeriodInDays =
 365)
    {
        try{
            $this->client->enableKeyRotation([
                'KeyId' => $keyId,
                'RotationPeriodInDays' => $rotationPeriodInDays,
            ]);
```

```
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
 resource could not be found.\n";
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param string $policy
     * @return void
     */
    public function putKeyPolicy(string $keyId, string $policy)
    {
        try {
            $this->client->putKeyPolicy([
                'KeyId' => $keyId,
                'Policy' => $policy,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem replacing the key policy: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



    /***
     * @param string $aliasName
     * @return void
     */
    public function deleteAlias(string $aliasName)
    {
        try {
            $this->client->deleteAlias([
                'AliasName' => $aliasName,
            ]);
        }catch(KmsException $caught){
```

```
            echo "There was a problem deleting the alias: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param string $message
     * @param string $signature
     * @param string $signingAlgorithm
     * @return bool
     */
    public function verify(string $keyId, string $message, string $signature,
 string $signingAlgorithm)
    {
        try {
            $result = $this->client->verify([
                'KeyId' => $keyId,
                'Message' => $message,
                'Signature' => $signature,
                'SigningAlgorithm' => $signingAlgorithm,
            ]);
            return $result['SignatureValid'];
        }catch(KmsException $caught){
            echo "There was a problem verifying the signature: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }


}
```

- API 세부 정보는 AWS SDK for PHP API 참조의 다음 주제를 참조하세요.
  - CreateAlias
  - CreateGrant
  - CreateKey

- Decrypt
- DescribeKey
- DisableKey
- EnableKey
- 암호화
- GetKeyPolicy
- ListAliases
- ListGrants
- ListKeys
- RevokeGrant
- ScheduleKeyDeletion
- Sign
- TagResource

Python

SDK for Python(Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class KMSScenario:
    """Runs an interactive scenario that shows how to get started with KMS."""

    def __init__(
        self,
        key_manager: KeyManager,
        key_encryption: KeyEncrypt,
        alias_manager: AliasManager,
        grant_manager: GrantManager,
        key_policy: KeyPolicy,
    ):
        self.key_manager = key_manager
```

```
        self.key_encryption = key_encryption
        self.alias_manager = alias_manager
        self.grant_manager = grant_manager
        self.key_policy = key_policy
        self.key_id = ""
        self.alias_name = ""
        self.asymmetric_key_id = ""

    def kms_scenario(self):
        key_description = "Created by the AWS KMS API"

        print(DASHES)
        print(
            """
Welcome to the AWS Key Management SDK Basics scenario.

This program demonstrates how to interact with AWS Key Management using the AWS
 SDK for Python (Boto3).
The AWS Key Management Service (KMS) is a secure and highly available service
 that allows you to create
and manage AWS KMS keys and control their use across a wide range of AWS services
 and applications.
KMS provides a centralized and unified approach to managing encryption keys,
 making it easier to meet your
data protection and regulatory compliance requirements.

This Basics scenario creates two key types:

- A symmetric encryption key is used to encrypt and decrypt data.
- An asymmetric key used to digitally sign data.

Let's get started...
        """
        )
        q.ask("Press Enter to continue...")

        print(DASHES)
        print(f"1. Create a symmetric KMS key\n")
        print(
            f"First, the program will creates a symmetric KMS key that you can
 used to encrypt and decrypt data."
        )
        q.ask("Press Enter to continue...")
        self.key_id = self.key_manager.create_key(key_description)["KeyId"]
```

```
        print(f"A symmetric key was successfully created {self.key_id}.")
        q.ask("Press Enter to continue...")
        print(DASHES)
        print(
            """
2. Enable a KMS key

By default, when the SDK creates an AWS key, it is enabled. The next bit of code
 checks to
determine if the key is enabled.
        """
        )
        q.ask("Press Enter to continue...")
        is_enabled = self.is_key_enabled(self.key_id)
        print(f"Is the key enabled? {is_enabled}")
        if not is_enabled:
            self.key_manager.enable_key(self.key_id)
        q.ask("Press Enter to continue...")
        print(DASHES)
        print(f"3. Encrypt data using the symmetric KMS key")
        plain_text = "Hello, AWS KMS!"
        print(
            f"""
One of the main uses of symmetric keys is to encrypt and decrypt data.
Next, the code encrypts the string "{plain_text}" with the SYMMETRIC_DEFAULT
 encryption algorithm.
        """
        )
        q.ask("Press Enter to continue...")
        encrypted_text = self.key_encryption.encrypt(self.key_id, plain_text)
        print(DASHES)
        print(f"4. Create an alias")
        print(
            """
Now, the program will create an alias for the KMS key. An alias is a friendly
 name that you
can associate with a KMS key. The alias name should be prefixed with 'alias/'.
        """
        )
        alias_name = q.ask("Enter an alias name: ", q.non_empty)
        self.alias_manager.create_alias(self.key_id, alias_name)
        print(f"{alias_name} was successfully created.")
        self.alias_name = alias_name
        print(DASHES)
```

```
        print(f"5. List all of your aliases")
        q.ask("Press Enter to continue...")
        self.alias_manager.list_aliases(10)
        q.ask("Press Enter to continue...")
        print(DASHES)
        print(f"6. Enable automatic rotation of the KMS key")
        print(
            """

By default, when the SDK enables automatic rotation of a KMS key,
KMS rotates the key material of the KMS key one year (approximately 365 days)
 from the enable date and every year
thereafter.
        """
        )
        q.ask("Press Enter to continue...")
        self.key_manager.enable_key_rotation(self.key_id)
        print(DASHES)
        print(f"Key rotation has been enabled for key with id {self.key_id}")
        print(
            """
7. Create a grant

A grant is a policy instrument that allows Amazon Web Services principals to use
 KMS keys.
It also can allow them to view a KMS key (DescribeKey) and create and manage
 grants.
When authorizing access to a KMS key, grants are considered along with key
 policies and IAM policies.
        """
        )
        print(
            """
To create a grant you must specify a account_id. To specify the grantee
 account_id, use the Amazon Resource Name (ARN)
of an AWS account_id. Valid principals include AWS accounts, IAM users, IAM
 roles, federated users,
and assumed role users.
        """
        )
        account_id = q.ask(
            "Enter an account_id, or press enter to skip creating a grant... "
        )
        grant = None
```

```
        if account_id != "":
            grant = self.grant_manager.create_grant(
                self.key_id,
                account_id,
                [
                    "Encrypt",
                    "Decrypt",
                    "DescribeKey",
                ],
            )
            print(f"Grant created successfully with ID: {grant['GrantId']}")

        q.ask("Press Enter to continue...")
        print(DASHES)
        print(DASHES)
        print(f"8. List grants for the KMS key")
        q.ask("Press Enter to continue...")
        self.grant_manager.list_grants(self.key_id)
        q.ask("Press Enter to continue...")
        print(DASHES)
        print(f"9. Revoke the grant")
        print(
            """
The revocation of a grant immediately removes the permissions and access that the
 grant had provided.
This means that any account_id (user, role, or service) that was granted access
 to perform specific
KMS operations on a KMS key will no longer be able to perform those operations.
        """
        )
        q.ask("Press Enter to continue...")

        if grant is not None:
            self.grant_manager.revoke_grant(self.key_id, grant["GrantId"])
            print(f"Grant ID: {grant['GrantId']} was successfully revoked!")

        q.ask("Press Enter to continue...")
        print(DASHES)
        print(f"10. Decrypt the data\n")
        print(
            """
Lets decrypt the data that was encrypted in an early step.
The code uses the same key to decrypt the string that we encrypted earlier in the
 program.
```

```
        """
        )
        q.ask("Press Enter to continue...")
        decrypted_data = self.key_encryption.decrypt(self.key_id, encrypted_text)
        print(f"Data decrypted successfully for key ID: {self.key_id}")
        print(f"Decrypted data: {decrypted_data}")

        q.ask("Press Enter to continue...")
        print(DASHES)
        print(f"11. Replace a key policy\n")
        print(
            """
A key policy is a resource policy for a KMS key. Key policies are the primary way
 to control
access to KMS keys. Every KMS key must have exactly one key policy. The
 statements in the key policy
determine who has permission to use the KMS key and how they can use it.
You can also use IAM policies and grants to control access to the KMS key, but
 every KMS key
must have a key policy.

By default, when you create a key by using the SDK, a policy is created that
gives the AWS account that owns the KMS key full access to the KMS key.

Let's try to replace the automatically created policy with the following policy.
{
"Version":"2012-10-17",
"Statement": [{
"Effect": "Allow",
"Principal": {"AWS": "arn:aws:iam::0000000000:root"},
"Action": "kms:*",
"Resource": "*"
}]
}
        """
        )
        account_id = q.ask("Enter your account ID or press enter to skip: ")
        if account_id != "":
            policy = {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {"AWS": f"arn:aws:iam::{account_id}:root"},
```

```
                    "Action": "kms:*",
                    "Resource": "*",
                }
            ],
        }

        self.key_policy.set_new_policy(self.key_id, policy)
        print("Key policy replacement succeeded.")
        q.ask("Press Enter to continue...")
    else:
        print("Skipping replacing the key policy.")

    print(DASHES)
    print(f"12. Get the key policy\n")
    print(
        f"The next bit of code that runs gets the key policy to make sure it
exists."
    )
    q.ask("Press Enter to continue...")
    policy = self.key_policy.get_policy(self.key_id)
    print(f"The key policy is: {policy}")

    q.ask("Press Enter to continue...")
    print(DASHES)
    print(f"13. Create an asymmetric KMS key and sign your data\n")
    print(
        """
    Signing your data with an AWS key can provide several benefits that make
it an attractive option
    for your data signing needs. By using an AWS KMS key, you can leverage
the
    security controls and compliance features provided by AWS,
    which can help you meet various regulatory requirements and enhance the
overall security posture
    of your organization.
    """
    )
    q.ask("Press Enter to continue...")
    print(f"Sign and verify data operation succeeded.")
    self.asymmetric_key_id = self.key_manager.create_asymmetric_key()
    message = "Here is the message that will be digitally signed"
    signature = self.key_encryption.sign(self.asymmetric_key_id, message)
    if self.key_encryption.verify(self.asymmetric_key_id, message,
signature):
```

```
            print("Signature verification succeeded.")
        else:
            print("Signature verification failed.")

        q.ask("Press Enter to continue...")
        print(DASHES)
        print(f"14. Tag your symmetric KMS Key\n")
        print(
            """
        By using tags, you can improve the overall management, security, and
 governance of your
        KMS keys, making it easier to organize, track, and control access to your
 encrypted data within
        your AWS environment
        """
        )
        q.ask("Press Enter to continue...")
        self.key_manager.tag_resource(self.key_id, "Environment", "Production")
        self.clean_up()

    def is_key_enabled(self, key_id: str) -> bool:
        """
        Check if the key is enabled or not.

        :param key_id: The key to check.
        :return: True if the key is enabled, otherwise False.
        """
        response = self.key_manager.describe_key(key_id)
        return response["Enabled"] is True

    def clean_up(self):
        """
        Delete resources created by this scenario.
        """
        if self.alias_name != "":
            print(f"Deleting the alias {self.alias_name}.")
            self.alias_manager.delete_alias(self.alias_name)
        window = 7  # The window in days for a scheduled deletion.
        if self.key_id != "":
            print(
                """
Warning:
Deleting a KMS key is a destructive and potentially dangerous operation. When a
 KMS key is deleted,
```

```
all data that was encrypted under the KMS key is unrecoverable.
                """
            )
            if q.ask(
                f"Do you want to delete the key with ID {self.key_id} (y/n)?",
                q.is_yesno,
            ):
                print(
                    f"The key {self.key_id} will be deleted with a window of
 {window} days. You can cancel the deletion before"
                )
                print("the window expires.")
                self.key_manager.delete_key(self.key_id, window)
                self.key_id = ""

        if self.asymmetric_key_id != "":
            if q.ask(
                f"Do you want to delete the asymmetric key with ID
 {self.asymmetric_key_id} (y/n)?",
                q.is_yesno,
            ):
                print(
                    f"The key {self.asymmetric_key_id} will be deleted with a
 window of {window} days. You can cancel the deletion before"
                )
                print("the window expires.")
                self.key_manager.delete_key(self.asymmetric_key_id, window)
                self.asymmetric_key_id = ""


if __name__ == "__main__":
    kms_scenario = None
    try:
        kms_client = boto3.client("kms")
        a_key_manager = KeyManager(kms_client)
        a_key_encrypt = KeyEncrypt(kms_client)
        an_alias_manager = AliasManager(kms_client)
        a_grant_manager = GrantManager(kms_client)
        a_key_policy = KeyPolicy(kms_client)
        kms_scenario = KMSScenario(
            key_manager=a_key_manager,
            key_encryption=a_key_encrypt,
            alias_manager=an_alias_manager,
            grant_manager=a_grant_manager,
```

```
            key_policy=a_key_policy,
        )
        kms_scenario.kms_scenario()
    except Exception:
        logging.exception("Something went wrong with the demo!")
        if kms_scenario is not None:
            kms_scenario.clean_up()
```

KMS 키 관리를 위한 래퍼 클래스 및 메서드입니다.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def create_key(self, key_description: str) -> dict[str, any]:
        """
        Creates a key with a user-provided description.

        :param key_description: A description for the key.
        :return: The key ID.
        """
        try:
            key = self.kms_client.create_key(Description=key_description)
["KeyMetadata"]
            self.created_keys.append(key)
            return key
        except ClientError as err:
            logging.error(
```

```
                "Couldn't create your key. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise


    def describe_key(self, key_id: str) -> dict[str, any]:
        """
        Describes a key.

        :param key_id: The ARN or ID of the key to describe.
        :return: Information about the key.
        """

        try:
            key = self.kms_client.describe_key(KeyId=key_id)["KeyMetadata"]
            return key
        except ClientError as err:
            logging.error(
                "Couldn't get key '%s'. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise


    def enable_key_rotation(self, key_id: str) -> None:
        """
        Enables rotation for a key.

        :param key_id: The ARN or ID of the key to enable rotation for.
        """
        try:
            self.kms_client.enable_key_rotation(KeyId=key_id)
        except ClientError as err:
            logging.error(
                "Couldn't enable rotation for key '%s'. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise


    def create_asymmetric_key(self) -> str:
```

```
        """
        Creates an asymmetric key in AWS KMS for signing messages.

        :return: The ID of the created key.
        """
        try:
            key = self.kms_client.create_key(
                KeySpec="RSA_2048", KeyUsage="SIGN_VERIFY", Origin="AWS_KMS"
            )["KeyMetadata"]
            self.created_keys.append(key)
            return key["KeyId"]
        except ClientError as err:
            logger.error(
                "Couldn't create your key. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise


    def tag_resource(self, key_id: str, tag_key: str, tag_value: str) -> None:
        """
        Add or edit tags on a customer managed key.

        :param key_id: The ARN or ID of the key to enable rotation for.
        :param tag_key: Key for the tag.
        :param tag_value: Value for the tag.
        """
        try:
            self.kms_client.tag_resource(
                KeyId=key_id, Tags=[{"TagKey": tag_key, "TagValue": tag_value}]
            )
        except ClientError as err:
            logging.error(
                "Couldn't add a tag for the key '%s'. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise


    def delete_key(self, key_id: str, window: int) -> None:
        """
        Deletes a list of keys.
```

```
        Warning:
        Deleting a KMS key is a destructive and potentially dangerous operation.
 When a KMS key is deleted,
        all data that was encrypted under the KMS key is unrecoverable.

        :param key_id: The ARN or ID of the key to delete.
        :param window: The waiting period, in days, before the KMS key is
 deleted.
        """

        try:
            self.kms_client.schedule_key_deletion(
                KeyId=key_id, PendingWindowInDays=window
            )
        except ClientError as err:
            logging.error(
                "Couldn't delete key %s. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

KMS 키 별칭을 위한 래퍼 클래스 및 메서드입니다.

```
class AliasManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_key = None

    @classmethod
    def from_client(cls) -> "AliasManager":
        """
        Creates an AliasManager instance with a default KMS client.

        :return: An instance of AliasManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)
```

```
    def create_alias(self, key_id: str, alias: str) -> None:
        """
        Creates an alias for the specified key.

        :param key_id: The ARN or ID of a key to give an alias.
        :param alias: The alias to assign to the key.
        """
        try:
            self.kms_client.create_alias(AliasName=alias, TargetKeyId=key_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "AlreadyExistsException":
                logger.error(
                    "Could not create the alias %s because it already exists.",
key_id
                )
            else:
                logger.error(
                    "Couldn't encrypt text. Here's why: %s",
                    err.response["Error"]["Message"],
                )
                raise


    def list_aliases(self, page_size: int) -> None:
        """
        Lists aliases for the current account.
        :param page_size: The number of aliases to list per page.
        """
        try:
            alias_paginator = self.kms_client.get_paginator("list_aliases")
            for alias_page in alias_paginator.paginate(
                PaginationConfig={"PageSize": page_size}
            ):
                print(f"Here are {page_size} aliases:")
                pprint(alias_page["Aliases"])
                if alias_page["Truncated"]:
                    answer = input(
                        f"Do you want to see the next {page_size} aliases (y/n)?
"
                    )
                    if answer.lower() != "y":
                        break
                else:
                    print("That's all your aliases!")
```

```
        except ClientError as err:
            logging.error(
                "Couldn't list your aliases. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise


    def delete_alias(self, alias: str) -> None:
        """
        Deletes an alias.

        :param alias: The alias to delete.
        """
        try:
            self.kms_client.delete_alias(AliasName=alias)
        except ClientError as err:
            logger.error(
                "Couldn't delete alias %s. Here's why: %s",
                alias,
                err.response["Error"]["Message"],
            )
            raise
```

KMS 키 암호화를 위한 래퍼 클래스 및 메서드입니다.

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.

        :return: An instance of KeyEncrypt initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)
```

```
   def encrypt(self, key_id: str, text: str) -> bytes:
       """
       Encrypts text by using the specified key.

       :param key_id: The ARN or ID of the key to use for encryption.
       :param text: The text to encrypt.
       :return: The encrypted version of the text.
       """
       try:
           response = self.kms_client.encrypt(KeyId=key_id,
Plaintext=text.encode())
           print(
               f"The string was encrypted with algorithm
{response['EncryptionAlgorithm']}"
           )
           return response["CiphertextBlob"]
       except ClientError as err:
           if err.response["Error"]["Code"] == "DisabledException":
               logger.error(
                   "Could not encrypt because the key %s is disabled.", key_id
               )
           else:
               logger.error(
                   "Couldn't encrypt text. Here's why: %s",
                   err.response["Error"]["Message"],
               )
           raise


   def decrypt(self, key_id: str, cipher_text: bytes) -> str:
       """
       Decrypts text previously encrypted with a key.

       :param key_id: The ARN or ID of the key used to decrypt the data.
       :param cipher_text: The encrypted text to decrypt.
       :return: The decrypted text.
       """
       try:
           return self.kms_client.decrypt(KeyId=key_id,
CiphertextBlob=cipher_text)[
               "Plaintext"
           ].decode()
```

```
        except ClientError as err:
            logger.error(
                "Couldn't decrypt your ciphertext. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise


    def sign(self, key_id: str, message: str) -> str:
        """
        Signs a message with a key.

        :param key_id: The ARN or ID of the key to use for signing.
        :param message: The message to sign.
        :return: The signature of the message.
        """
        try:
            return self.kms_client.sign(
                KeyId=key_id,
                Message=message.encode(),
                SigningAlgorithm="RSASSA_PSS_SHA_256",
            )["Signature"]
        except ClientError as err:
            logger.error(
                "Couldn't sign your message. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise


    def verify(self, key_id: str, message: str, signature: str) -> bool:
        """
        Verifies a signature against a message.

        :param key_id: The ARN or ID of the key used to sign the message.
        :param message: The message to verify.
        :param signature: The signature to verify.
        :return: True when the signature matches the message, otherwise False.
        """
        try:
            response = self.kms_client.verify(
                KeyId=key_id,
                Message=message.encode(),
                Signature=signature,
```

```
                SigningAlgorithm="RSASSA_PSS_SHA_256",
            )
            valid = response["SignatureValid"]
            print(f"The signature is {'valid' if valid else 'invalid'}.")
            return valid
        except ClientError as err:
            if err.response["Error"]["Code"] == "SignatureDoesNotMatchException":
                print("The signature is not valid.")
            else:
                logger.error(
                    "Couldn't verify your signature. Here's why: %s",
                    err.response["Error"]["Message"],
                )
            raise
```

KMS 키 권한 부여를 위한 래퍼 클래스 및 메서드입니다.

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "GrantManager":
        """
        Creates a GrantManager instance with a default KMS client.

        :return: An instance of GrantManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def create_grant(
        self, key_id: str, principal: str, operations: [str]
    ) -> dict[str, str]:
        """
        Creates a grant for a key that lets a principal generate a symmetric data
        encryption key.

        :param key_id: The ARN or ID of the key.
```

```
        :param principal: The principal to grant permission to.
        :param operations: The operations to grant permission for.
        :return: The grant that is created.
        """
        try:
            return self.kms_client.create_grant(
                KeyId=key_id,
                GranteePrincipal=principal,
                Operations=operations,
            )
        except ClientError as err:
            logger.error(
                "Couldn't create a grant on key %s. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise


    def list_grants(self, key_id):
        """
        Lists grants for a key.

        :param key_id: The ARN or ID of the key to query.
        :return: The grants for the key.
        """
        try:
            paginator = self.kms_client.get_paginator("list_grants")
            grants = []
            page_iterator = paginator.paginate(KeyId=key_id)
            for page in page_iterator:
                grants.extend(page["Grants"])

            print(f"Grants for key {key_id}:")
            pprint(grants)
            return grants
        except ClientError as err:
            logger.error(
                "Couldn't list grants for key %s. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

```
    def revoke_grant(self, key_id: str, grant_id: str) -> None:
        """
        Revokes a grant so that it can no longer be used.

        :param key_id: The ARN or ID of the key associated with the grant.
        :param grant_id: The ID of the grant to revoke.
        """
        try:
            self.kms_client.revoke_grant(KeyId=key_id, GrantId=grant_id)
        except ClientError as err:
            logger.error(
                "Couldn't revoke grant %s. Here's why: %s",
                grant_id,
                err.response["Error"]["Message"],
            )
            raise
```

KMS 키 정책을 위한 래퍼 클래스 및 메서드입니다.

```
class KeyPolicy:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyPolicy":
        """
        Creates a KeyPolicy instance with a default KMS client.

        :return: An instance of KeyPolicy initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def set_new_policy(self, key_id: str, policy: dict[str, any]) -> None:
        """
        Sets the policy of a key. Setting a policy entirely overwrites the
 existing
```

```
        policy, so care is taken to add a statement to the existing list of
statements
        rather than simply writing a new policy.

        :param key_id: The ARN or ID of the key to set the policy to.
        :param policy: A new key policy. The key policy must allow the calling
principal to make a subsequent
                       PutKeyPolicy request on the KMS key. This reduces the risk
that the KMS key becomes unmanageable
        """

        try:
            self.kms_client.put_key_policy(KeyId=key_id,
Policy=json.dumps(policy))
        except ClientError as err:
            logger.error(
                "Couldn't set policy for key %s. Here's why %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise



    def get_policy(self, key_id: str) -> dict[str, str]:
        """
        Gets the policy of a key.

        :param key_id: The ARN or ID of the key to query.
        :return: The key policy as a dict.
        """
        if key_id != "":
            try:
                response = self.kms_client.get_key_policy(
                    KeyId=key_id,
                )
                policy = json.loads(response["Policy"])
            except ClientError as err:
                logger.error(
                    "Couldn't get policy for key %s. Here's why: %s",
                    key_id,
                    err.response["Error"]["Message"],
                )
                raise
```

```
            else:
                pprint(policy)
                return policy
        else:
            print("Skipping get policy demo.")
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 다음 주제를 참조하세요.
  - CreateAlias
  - CreateGrant
  - CreateKey
  - Decrypt
  - DescribeKey
  - DisableKey
  - EnableKey
  - 암호화
  - GetKeyPolicy
  - ListAliases
  - ListGrants
  - ListKeys
  - RevokeGrant
  - ScheduleKeyDeletion
  - Sign
  - TagResource

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDKs AWS KMS 를 사용하기 위한 작업

다음 코드 예제에서는 AWS SDKs를 사용하여 개별 AWS KMS 작업을 수행하는 방법을 보여줍니다. 각 예제에는 GitHub에 대한 링크가 포함되어 있습니다. 여기에서 코드 설정 및 실행에 대한 지침을 찾을 수 있습니다.

이러한 발췌문은 AWS KMS API를 호출하며 컨텍스트에서 실행해야 하는 더 큰 프로그램에서 발췌한 코드입니다. AWS SDKs AWS KMS 사용 시나리오에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

다음 예제에는 가장 일반적으로 사용되는 작업만 포함되어 있습니다. 전체 목록은 AWS Key Management Service API 참조를 참조하세요.

예제

- AWS SDK 또는 CLI와 CreateAlias 함께 사용
- AWS SDK 또는 CLI와 CreateGrant 함께 사용
- AWS SDK 또는 CLI와 CreateKey 함께 사용
- AWS SDK 또는 CLI와 Decrypt 함께 사용
- AWS SDK 또는 CLI와 DeleteAlias 함께 사용
- AWS SDK 또는 CLI와 DescribeKey 함께 사용
- AWS SDK 또는 CLI와 DisableKey 함께 사용
- AWS SDK 또는 CLI와 EnableKey 함께 사용
- AWS SDK 또는 CLI와 EnableKeyRotation 함께 사용
- AWS SDK 또는 CLI와 Encrypt 함께 사용
- AWS SDK 또는 CLI와 GenerateDataKey 함께 사용
- AWS SDK 또는 CLI와 GenerateDataKeyWithoutPlaintext 함께 사용
- AWS SDK 또는 CLI와 GenerateRandom 함께 사용
- AWS SDK 또는 CLI와 GetKeyPolicy 함께 사용
- AWS SDK 또는 CLI와 ListAliases 함께 사용
- AWS SDK 또는 CLI와 ListGrants 함께 사용
- AWS SDK 또는 CLI와 ListKeyPolicies 함께 사용
- AWS SDK 또는 CLI와 ListKeys 함께 사용
- AWS SDK 또는 CLI와 PutKeyPolicy 함께 사용
- AWS SDK 또는 CLI와 ReEncrypt 함께 사용
- AWS SDK 또는 CLI와 RetireGrant 함께 사용
- AWS SDK 또는 CLI와 RevokeGrant 함께 사용
- AWS SDK 또는 CLI와 ScheduleKeyDeletion 함께 사용
- AWS SDK 또는 CLI와 Sign 함께 사용
- AWS SDK 또는 CLI와 TagResource 함께 사용

- AWS SDK 또는 CLI와 UpdateAlias 함께 사용
- AWS SDK 또는 CLI와 Verify 함께 사용

AWS SDK 또는 CLI와 **CreateAlias** 함께 사용

다음 코드 예시는 CreateAlias의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

/// <summary>
/// Creates an alias for an AWS Key Management Service (AWS KMS) key.
/// </summary>
public class CreateAlias
{
    public static async Task Main()
    {
        var client = new AmazonKeyManagementServiceClient();

        // The alias name must start with alias/ and can be
        // up to 256 alphanumeric characters long.
        var aliasName = "alias/ExampleAlias";
```

```
            // The value supplied as the TargetKeyId can be either
            // the key ID or key Amazon Resource Name (ARN) of the
            // AWS KMS key.
            var keyId = "1234abcd-12ab-34cd-56ef-1234567890ab";

            var request = new CreateAliasRequest
            {
                AliasName = aliasName,
                TargetKeyId = keyId,
            };

            var response = await client.CreateAliasAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Alias, {aliasName}, successfully created.");
            }
            else
            {
                Console.WriteLine($"Could not create alias.");
            }
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for .NET API 참조의 CreateAlias를 참조하세요.

CLI

AWS CLI

KMS 키의 별칭을 생성하는 방법

다음 `create-alias` 명령은 키 ID `1234abcd-12ab-34cd-56ef-1234567890ab`로 식별되는 KMS 키에 대해 `example-alias`라는 별칭을 만듭니다.

별칭 이름은 `alias/`로 시작해야 합니다. 로 시작하는 별칭 이름은에서 사용하도록 예약되어 `alias/aws`있으므로 사용하지 마십시오 AWS.

```
aws kms create-alias \
    --alias-name alias/example-alias \
```

```
    --target-key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

이 명령은 출력을 반환하지 않습니다. 새 별칭을 보려면 `list-aliases` 명령을 사용합니다.

자세한 내용은 AWS Key Management Service 개발자 안내서의 별칭 사용을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 CreateAlias를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Creates a custom alias for the specified target key asynchronously.
    *
    * @param targetKeyId the ID of the target key for the alias
    * @param aliasName   the name of the alias to create
    * @return a {@link CompletableFuture} that completes when the alias creation
operation is finished
    */
   public CompletableFuture<Void> createCustomAliasAsync(String targetKeyId,
String aliasName) {
       CreateAliasRequest aliasRequest = CreateAliasRequest.builder()
           .aliasName(aliasName)
           .targetKeyId(targetKeyId)
           .build();

       CompletableFuture<CreateAliasResponse> responseFuture =
getAsyncClient().createAlias(aliasRequest);
       responseFuture.whenComplete((response, exception) -> {
           if (exception == null) {
               logger.info("{} was successfully created.", aliasName);
           } else {
               if (exception instanceof ResourceExistsException) {
                   logger.info("Alias [{}] already exists. Moving on...",
aliasName);
```

```
                } else if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("KMS error occurred while creating
 alias: " + kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred
 while creating alias: " + exception.getMessage(), exception);
                }
            }
        });

        return responseFuture.thenApply(response -> null);
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateAlias를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createCustomAlias(
    targetKeyIdVal: String?,
    aliasNameVal: String?,
) {
    val request =
        CreateAliasRequest {
            aliasName = aliasNameVal
            targetKeyId = targetKeyIdVal
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        kmsClient.createAlias(request)
        println("$aliasNameVal was successfully created")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateAlias를 참조하세요.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /***
     * @param string $keyId
     * @param string $alias
     * @return void
     */
    public function createAlias(string $keyId, string $alias)
    {
        try{
            $this->client->createAlias([
                'TargetKeyId' => $keyId,
                'AliasName' => $alias,
            ]);
        }catch (KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidAliasNameException"){
                echo "The request was rejected because the specified alias name
is not valid.";
            }
            throw $caught;
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for PHP API 참조의 CreateAlias를 참조하세요.

Python

SDK for Python(Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class AliasManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_key = None

    @classmethod
    def from_client(cls) -> "AliasManager":
        """
        Creates an AliasManager instance with a default KMS client.

        :return: An instance of AliasManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def create_alias(self, key_id: str, alias: str) -> None:
        """
        Creates an alias for the specified key.

        :param key_id: The ARN or ID of a key to give an alias.
        :param alias: The alias to assign to the key.
        """
        try:
            self.kms_client.create_alias(AliasName=alias, TargetKeyId=key_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "AlreadyExistsException":
                logger.error(
                    "Could not create the alias %s because it already exists.",
 key_id
                )
```

```
            else:
                logger.error(
                    "Couldn't encrypt text. Here's why: %s",
                    err.response["Error"]["Message"],
                )
                raise
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 CreateAlias를 참조하세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        " iv_alias_name = 'alias/my-key-alias'
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        lo_kms->createalias(
          iv_aliasname = iv_alias_name
          iv_targetkeyid = iv_key_id
        ).
        MESSAGE 'Alias created successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsalreadyexistsex.
        MESSAGE 'Alias already exists.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmsinvalidaliasnameex.
        MESSAGE 'Invalid alias name.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 CreateAlias를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **CreateGrant** 함께 사용

다음 코드 예시는 CreateGrant의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        public static async Task Main()
        {
            var client = new AmazonKeyManagementServiceClient();

            // The identity that is given permission to perform the operations
            // specified in the grant.
            var grantee = "arn:aws:iam::111122223333:role/ExampleRole";

            // The identifier of the AWS KMS key to which the grant applies. You
            // can use the key ID or the Amazon Resource Name (ARN) of the KMS
key.
            var keyId = "7c9eccc2-38cb-4c4f-9db3-766ee8dd3ad4";

            var request = new CreateGrantRequest
            {
                GranteePrincipal = grantee,
```

```
                KeyId = keyId,

                // A list of operations that the grant allows.
                Operations = new List<string>
                {
                    "Encrypt",
                    "Decrypt",
                },
            };

            var response = await client.CreateGrantAsync(request);

            string grantId = response.GrantId; // The unique identifier of the
 grant.
            string grantToken = response.GrantToken; // The grant token.

            Console.WriteLine($"Id: {grantId}, Token: {grantToken}");
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for .NET API 참조의 CreateGrant를 참조하세요.

CLI

AWS CLI

권한 부여를 생성하는 방법

다음 `create-grant` 예시에서는 `exampleUser` 사용자가 `1234abcd-12ab-34cd-56ef-1234567890ab` 예시 KMS 키에서 `decrypt` 명령을 사용할 수 있는 권한을 생성합니다. 사용 중지하는 위탁자는 `adminRole` 역할입니다. 이 권한 부여는 `decrypt` 요청의 암호화 컨텍스트에 `"Department": "IT"` 키-값 페어가 포함된 경우에만 이 권한을 허용하도록 `EncryptionContextSubset` 권한 부여 제약 조건을 사용합니다.

```
aws kms create-grant \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --grantee-principal arn:aws:iam::123456789012:user/exampleUser \
    --operations Decrypt \
    --constraints EncryptionContextSubset={Department=IT} \
    --retiring-principal arn:aws:iam::123456789012:role/adminRole
```

출력:

```
{
    "GrantId":
 "1a2b3c4d2f5e69f440bae30eaec9570bb1fb7358824f9ddfa1aa5a0dab1a59b2",
    "GrantToken": "<grant token here>"
}
```

권한 부여에 대한 자세한 정보를 보려면 `list-grants` 명령을 사용합니다.

자세한 내용은 AWS Key Management Service 개발자 안내서의 AWS KMS의 권한 부여를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 CreateGrant를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Grants permissions to a specified principal on a customer master key (CMK)
asynchronously.
    *
    * @param keyId             The unique identifier for the customer master key
(CMK) that the grant applies to.
    * @param granteePrincipal  The principal that is given permission to perform
the operations that the grant permits on the CMK.
    * @return A {@link CompletableFuture} that, when completed, contains the ID
of the created grant.
    * @throws RuntimeException If an error occurs during the grant creation
process.
    */
   public CompletableFuture<String> grantKeyAsync(String keyId, String
granteePrincipal) {
       List<GrantOperation> grantPermissions = List.of(
           GrantOperation.ENCRYPT,
```

```
            GrantOperation.DECRYPT,
            GrantOperation.DESCRIBE_KEY
        );

        CreateGrantRequest grantRequest = CreateGrantRequest.builder()
            .keyId(keyId)
            .name("grant1")
            .granteePrincipal(granteePrincipal)
            .operations(grantPermissions)
            .build();

        CompletableFuture<CreateGrantResponse> responseFuture =
getAsyncClient().createGrant(grantRequest);
        responseFuture.whenComplete((response, ex) -> {
            if (ex == null) {
                logger.info("Grant created successfully with ID: " +
response.grantId());
            } else {
                if (ex instanceof KmsException kmsEx) {
                    throw new RuntimeException("Failed to create grant: " +
kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
ex.getMessage(), ex);
                }
            }
        });

        return responseFuture.thenApply(CreateGrantResponse::grantId);
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateGrant를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createNewGrant(
    keyIdVal: String?,
    granteePrincipalVal: String?,
    operation: String,
): String? {
    val operationOb = GrantOperation.fromValue(operation)
    val grantOperationList = ArrayList<GrantOperation>()
    grantOperationList.add(operationOb)

    val request =
        CreateGrantRequest {
            keyId = keyIdVal
            granteePrincipal = granteePrincipalVal
            operations = grantOperationList
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.createGrant(request)
        return response.grantId
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateGrant를 참조하세요.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /***
     * @param string $keyId
     * @param string $granteePrincipal
     * @param array $operations
     * @param array $grantTokens
```

```
     * @return Result
     */
    public function createGrant(string $keyId, string $granteePrincipal, array
 $operations, array $grantTokens = [])
    {
        $args = [
            'KeyId' => $keyId,
            'GranteePrincipal' => $granteePrincipal,
            'Operations' => $operations,
        ];
        if($grantTokens){
            $args['GrantTokens'] = $grantTokens;
        }
        try{
            return $this->client->createGrant($args);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidGrantTokenException"){
                echo "The request was rejected because the specified grant token
 is not valid.\n";
            }
            throw $caught;
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for PHP API 참조의 CreateGrant를 참조하세요.

Python

SDK for Python(Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
```

```
    @classmethod
    def from_client(cls) -> "GrantManager":
        """
        Creates a GrantManager instance with a default KMS client.

        :return: An instance of GrantManager initialized with the default KMS
client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def create_grant(
        self, key_id: str, principal: str, operations: [str]
    ) -> dict[str, str]:
        """
        Creates a grant for a key that lets a principal generate a symmetric data
        encryption key.

        :param key_id: The ARN or ID of the key.
        :param principal: The principal to grant permission to.
        :param operations: The operations to grant permission for.
        :return: The grant that is created.
        """
        try:
            return self.kms_client.create_grant(
                KeyId=key_id,
                GranteePrincipal=principal,
                Operations=operations,
            )
        except ClientError as err:
            logger.error(
                "Couldn't create a grant on key %s. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 CreateGrant를 참조하세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_grantee_principal = 'arn:aws:iam::123456789012:role/my-role'
        " it_operations contains 'Encrypt', 'Decrypt', 'GenerateDataKey'
        oo_result = lo_kms->creategrant(
          iv_keyid = iv_key_id
          iv_granteeprincipal = iv_grantee_principal
          it_operations = it_operations
        ).
        MESSAGE 'Grant created successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsdisabledexception.
        MESSAGE 'The key is disabled.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 CreateGrant를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 `CreateKey` 함께 사용

다음 코드 예시는 CreateKey의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기
- 테이블 암호화 작업

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.KeyManagementService;
    using Amazon.KeyManagementService.Model;

    /// <summary>
    /// Shows how to create a new AWS Key Management Service (AWS KMS)
    /// key.
    /// </summary>
    public class CreateKey
    {
        public static async Task Main()
        {
            // Note that if you need to create a Key in an AWS Region
            // other than the Region defined for the default user, you need to
            // pass the Region to the client constructor.
            var client = new AmazonKeyManagementServiceClient();

            // The call to CreateKeyAsync will create a symmetrical AWS KMS
            // key. For more information about symmetrical and asymmetrical
            // keys, see:
            //
            // https://docs.aws.amazon.com/kms/latest/developerguide/symm-asymm-
choose.html
            var response = await client.CreateKeyAsync(new CreateKeyRequest());

            // The KeyMetadata object contains information about the new AWS KMS
 key.
```

```
            KeyMetadata keyMetadata = response.KeyMetadata;

            if (keyMetadata is not null)
            {
                Console.WriteLine($"KMS Key: {keyMetadata.KeyId} was successfully
created.");
            }
            else
            {
                Console.WriteLine("Could not create KMS Key.");
            }
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for .NET API 참조의 CreateKey를 참조하세요.

CLI

AWS CLI

예제 1: KMS에서 고객 관리형 AWS KMS 키 생성

다음 `create-key` 예시에서는 대칭 암호화 KMS 키를 생성합니다.

대칭 암호화 키인 기본 KMS 키를 생성하기 위해 파라미터를 지정할 필요가 없습니다. 이러한 파라미터의 기본값은 대칭 암호화 키를 생성합니다.

이 명령은 키 정책을 지정하지 않으므로 KMS 키는 프로그래밍 방식으로 만든 KMS 키에 대한 기본 키 정책을 가져옵니다. 키 정책을 보려면 `get-key-policy` 명령을 사용합니다. 키 정책을 변경하려면 `put-key-policy` 명령을 사용합니다.

```
aws kms create-key
```

`create-key` 명령은 새 KMS 키의 키 ID 및 ARN을 포함한 키 메타데이터를 반환합니다. 이러한 값을 사용하여 다른 KMS 작업에서 AWS KMS 키를 식별할 수 있습니다. 출력에 태그가 포함되지 않습니다. KMS 키에 지정된 태그를 보려면 `list-resource-tags command`를 사용합니다.

출력:

```
{
    "KeyMetadata": {
        "AWSAccountId": "111122223333",
        "Arn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "CreationDate": "2017-07-05T14:04:55-07:00",
        "CurrentKeyMaterialId":
 "0b7fd7ddbac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "Description": "",
        "Enabled": true,
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "KeyManager": "CUSTOMER",
        "KeySpec": "SYMMETRIC_DEFAULT",
        "KeyState": "Enabled",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "MultiRegion": false,
        "Origin": "AWS_KMS"
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ]
    }
}
```

참고: `create-key` 명령으로는 별칭을 지정할 수 없습니다. 새 KMS 키의 별칭을 만들려면 `create-alias` 명령을 사용합니다.

자세한 내용은 AWS Key Management Service 개발자 안내서의 키 생성을 참조하세요.

예시 2: 암호화 및 복호화를 위한 비대칭 RSA KMS 키 생성

다음 `create-key` 예시에서는 암호화 및 복호화를 위한 비대칭 RSA 키 페어가 포함된 KMS 키를 생성합니다. 키가 생성된 후에는 키 사양 및 키 사용량을 변경할 수 없습니다.

```
aws kms create-key \
   --key-spec RSA_4096 \
   --key-usage ENCRYPT_DECRYPT
```

출력:

```
{
    "KeyMetadata": {
```

```
        "Arn": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "AWSAccountId": "111122223333",
        "CreationDate": "2021-04-05T14:04:55-07:00",
        "CustomerMasterKeySpec": "RSA_4096",
        "Description": "",
        "Enabled": true,
        "EncryptionAlgorithms": [
            "RSAES_OAEP_SHA_1",
            "RSAES_OAEP_SHA_256"
        ],
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "KeyManager": "CUSTOMER",
        "KeySpec": "RSA_4096",
        "KeyState": "Enabled",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "MultiRegion": false,
        "Origin": "AWS_KMS"
    }
}
```

자세한 내용은 Key Management Service 개발자 안내서의 AWS KMS의 비대칭 키를 참조하세요. AWS

예시 3: 서명 및 확인을 위한 비대칭 타원 곡선 KMS 키 생성

서명 및 확인을 위해 비대칭 타원 곡선(ECC) 키 페어가 포함된 비대칭 KMS 키를 생성합니다. ECC KMS 키에 사용할 수 있는 유일한 값이 SIGN_VERIFY이더라도 --key-usage 파라미터는 필수입니다. 키가 생성된 후에는 키 사양 및 키 사용량을 변경할 수 없습니다.

```
aws kms create-key \
    --key-spec ECC_NIST_P521 \
    --key-usage SIGN_VERIFY
```

출력:

```
{
    "KeyMetadata": {
        "Arn": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "AWSAccountId": "111122223333",
        "CreationDate": "2019-12-02T07:48:55-07:00",
```

```
        "CustomerMasterKeySpec": "ECC_NIST_P521",
        "Description": "",
        "Enabled": true,
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "KeyManager": "CUSTOMER",
        "KeySpec": "ECC_NIST_P521",
        "KeyState": "Enabled",
        "KeyUsage": "SIGN_VERIFY",
        "MultiRegion": false,
        "Origin": "AWS_KMS",
        "SigningAlgorithms": [
            "ECDSA_SHA_512"
        ]
    }
}
```

자세한 내용은 Key Management Service 개발자 안내서의 AWS KMS의 비대칭 키를 참조하세요. AWS

예제 4: 서명 및 확인을 위한 비대칭 ML-DSA KMS 키 만들기

이 예제에서는 서명 및 확인을 위한 모듈 격자 디지털 서명 알고리즘(ML-DSA) 키를 만듭니다. ML-DSA 키에 사용할 수 있는 유일한 값이 SIGN_VERIFY라도 key-usage 파라미터는 필수입니다.

```
aws kms create-key \
    --key-spec ML_DSA_65 \
    --key-usage SIGN_VERIFY
```

출력:

```
{
    "KeyMetadata": {
        "Arn": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "AWSAccountId": "111122223333",
        "CreationDate": "2019-12-02T07:48:55-07:00",
        "Description": "",
        "Enabled": true,
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "KeyManager": "CUSTOMER",
        "KeySpec": "ML_DSA_65",
        "KeyState": "Enabled",
```

```
        "KeyUsage": "SIGN_VERIFY",
        "MultiRegion": false,
        "Origin": "AWS_KMS",
        "SigningAlgorithms": [
            "ML_DSA_SHAKE_256"
        ]
    }
}
```

자세한 내용은 Key Management Service 개발자 안내서의 AWS KMS의 비대칭 키를 참조하세요. AWS

예제 5: HMAC KMS 키 만들기

다음 `create-key` 예시에서는 384비트 HMAC KMS 키를 생성합니다. HMAC KMS 키에 사용할 수 있는 유일한 값이더라도 `--key-usage` 파라미터의 `GENERATE_VERIFY_MAC` 값은 필수입니다.

```
aws kms create-key \
    --key-spec HMAC_384 \
    --key-usage GENERATE_VERIFY_MAC
```

출력:

```
{
    "KeyMetadata": {
        "Arn": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "AWSAccountId": "111122223333",
        "CreationDate": "2022-04-05T14:04:55-07:00",
        "CustomerMasterKeySpec": "HMAC_384",
        "Description": "",
        "Enabled": true,
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "KeyManager": "CUSTOMER",
        "KeySpec": "HMAC_384",
        "KeyState": "Enabled",
        "KeyUsage": "GENERATE_VERIFY_MAC",
        "MacAlgorithms": [
            "HMAC_SHA_384"
        ],
        "MultiRegion": false,
```

```
        "Origin": "AWS_KMS"
    }
}
```

자세한 내용은 Key Management Service 개발자 안내서의 AWS KMS의 HMAC 키를 참조하세요. AWS

예제 6: 다중 리전 프라이머리 KMS 키 만들기

다음 `create-key` 예시에서는 다중 리전 프라이머리 대칭 암호화 키를 생성합니다. 모든 파라미터의 기본값이 대칭 암호화 키를 생성하므로 이 KMS 키에는 `--multi-region` 파라미터만 필요합니다. AWS CLI에서 부울 파라미터가 true임을 나타내려면 파라미터 이름만 지정하면 됩니다.

```
aws kms create-key \
    --multi-region
```

출력:

```
{
    "KeyMetadata": {
        "Arn": "arn:aws:kms:us-west-2:111122223333:key/
mrk-1234abcd12ab34cd56ef12345678990ab",
        "AWSAccountId": "111122223333",
        "CreationDate": "2021-09-02T016:15:21-09:00",
        "CurrentKeyMaterialId":
 "0b7fd7ddbac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "Description": "",
        "Enabled": true,
        "EncryptionAlgorithms": [
          "SYMMETRIC_DEFAULT"
        ],
        "KeyId": "mrk-1234abcd12ab34cd56ef12345678990ab",
        "KeyManager": "CUSTOMER",
        "KeySpec": "SYMMETRIC_DEFAULT",
        "KeyState": "Enabled",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "MultiRegion": true,
        "MultiRegionConfiguration": {
            "MultiRegionKeyType": "PRIMARY",
            "PrimaryKey": {
```

```
                "Arn": "arn:aws:kms:us-west-2:111122223333:key/
mrk-1234abcd12ab34cd56ef12345678990ab",
                "Region": "us-west-2"
            },
            "ReplicaKeys": []
        },
        "Origin": "AWS_KMS"
    }
}
```

자세한 내용은 Key Management Service 개발자 안내서의 AWS KMS의 비대칭 키를 참조하세요. AWS

예제 7: 가져온 키 자료에 대한 KMS 키 만들기

다음 create-key 예시에서는 키 구성 요소 없이 KMS 키를 생성합니다. 작업이 완료되면 자체 키 구성 요소를 KMS 키로 가져올 수 있습니다. 이 KMS 키를 생성하려면 --origin 파라미터를 EXTERNAL로 설정하세요.

```
aws kms create-key \
    --origin EXTERNAL
```

출력:

```
{
     "KeyMetadata": {
         "Arn": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
         "AWSAccountId": "111122223333",
         "CreationDate": "2019-12-02T07:48:55-07:00",
         "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
         "Description": "",
         "Enabled": false,
         "EncryptionAlgorithms": [
             "SYMMETRIC_DEFAULT"
         ],
         "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
         "KeyManager": "CUSTOMER",
         "KeySpec": "SYMMETRIC_DEFAULT",
         "KeyState": "PendingImport",
         "KeyUsage": "ENCRYPT_DECRYPT",
         "MultiRegion": false,
         "Origin": "EXTERNAL"
```

```
    }
}
```

자세한 내용은 Key Management Service 개발자 안내서의 AWS KMS 키에서 키 구성 요소 가져오기를 참조하세요. AWS

예제 6: AWS CloudHSM 키 스토어에서 KMS 키를 생성하는 방법

다음 `create-key` 예제에서는 지정된 AWS CloudHSM 키 스토어에 KMS 키를 생성합니다. 작업은 KMS 키와 해당 메타데이터를 AWS KMS에 생성하고 사용자 지정 키 스토어와 연결된 AWS CloudHSM 클러스터에 키 구성 요소를 생성합니다. `--custom-key-store-id` 및 `--origin` 파라미터가 필요합니다.

```
aws kms create-key \
    --origin AWS_CLOUDHSM \
    --custom-key-store-id cks-1234567890abcdef0
```

출력:

```
{
    "KeyMetadata": {
        "Arn": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "AWSAccountId": "111122223333",
        "CloudHsmClusterId": "cluster-1a23b4cdefg",
        "CreationDate": "2019-12-02T07:48:55-07:00",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "CustomKeyStoreId": "cks-1234567890abcdef0",
        "Description": "",
        "Enabled": true,
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ],
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "KeyManager": "CUSTOMER",
        "KeySpec": "SYMMETRIC_DEFAULT",
        "KeyState": "Enabled",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "MultiRegion": false,
        "Origin": "AWS_CLOUDHSM"
    }
}
```

자세한 내용은 AWS Key Management Service 개발자 안내서의 AWS CloudHSM 키 저장소를 참조하세요.

예제 8: 외부 키 저장소에 KMS 키 만들기

다음 `create-key` 예시에서는 지정된 외부 키 저장소에 KMS 키를 생성합니다. 이 명령에는 `--custom-key-store-id`, `--origin`, `--xks-key-id` 파라미터가 필요합니다.

`--xks-key-id` 파라미터는 외부 키 관리자에 있는 기존 대칭 암호화 키의 ID를 지정합니다. 이 키는 KMS 키의 외부 키 구성 요소 역할을 합니다. `--origin` 파라미터의 값은 `EXTERNAL_KEY_STORE`여야 합니다. `custom-key-store-id` 파라미터는 외부 키 저장소 프록시에 연결된 외부 키 저장소를 식별해야 합니다.

```
aws kms create-key \
    --origin EXTERNAL_KEY_STORE \
    --custom-key-store-id cks-9876543210fedcba9 \
    --xks-key-id bb8562717f809024
```

출력:

```
{
    "KeyMetadata": {
        "Arn": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "AWSAccountId": "111122223333",
        "CreationDate": "2022-12-02T07:48:55-07:00",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "CustomKeyStoreId": "cks-9876543210fedcba9",
        "Description": "",
        "Enabled": true,
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ],
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "KeyManager": "CUSTOMER",
        "KeySpec": "SYMMETRIC_DEFAULT",
        "KeyState": "Enabled",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "MultiRegion": false,
        "Origin": "EXTERNAL_KEY_STORE",
        "XksKeyConfiguration": {
            "Id": "bb8562717f809024"
        }
```

```
    }
}
```

자세한 내용은 AWS Key Management Service 개발자 안내서의 외부 키 저장소를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 CreateKey를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Creates a new symmetric encryption key asynchronously.
    *
    * @param keyDesc the description of the key to be created
    * @return a {@link CompletableFuture} that completes with the ID of the
newly created key
    * @throws RuntimeException if an error occurs while creating the key
    */
   public CompletableFuture<String> createKeyAsync(String keyDesc) {
       CreateKeyRequest keyRequest = CreateKeyRequest.builder()
           .description(keyDesc)
           .keySpec(KeySpec.SYMMETRIC_DEFAULT)
           .keyUsage(KeyUsageType.ENCRYPT_DECRYPT)
           .build();

       return getAsyncClient().createKey(keyRequest)
           .thenApply(resp -> resp.keyMetadata().keyId())
           .exceptionally(ex -> {
               throw new RuntimeException("An error occurred while creating the
key: " + ex.getMessage(), ex);
           });
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateKey를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createKey(keyDesc: String?): String? {
    val request =
        CreateKeyRequest {
            description = keyDesc
            customerMasterKeySpec = CustomerMasterKeySpec.SymmetricDefault
            keyUsage = KeyUsageType.fromValue("ENCRYPT_DECRYPT")
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val result = kmsClient.createKey(request)
        println("Created a customer key with id " + result.keyMetadata?.arn)
        return result.keyMetadata?.keyId
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateKey를 참조하세요.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /***
```

```
     * @param string $keySpec
     * @param string $keyUsage
     * @param string $description
     * @return array
     */
    public function createKey(string $keySpec = "", string $keyUsage = "", string
$description = "Created by the SDK for PHP")
    {
        $parameters = ['Description' => $description];
        if($keySpec && $keyUsage){
            $parameters['KeySpec'] = $keySpec;
            $parameters['KeyUsage'] = $keyUsage;
        }
        try {
            $result = $this->client->createKey($parameters);
            return $result['KeyMetadata'];
        }catch(KmsException $caught){
            // Check for error specific to createKey operations
            if ($caught->getAwsErrorMessage() == "LimitExceededException"){
                echo "The request was rejected because a quota was exceeded. For
more information, see Quotas in the Key Management Service Developer Guide.";
            }
            throw $caught;
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for PHP API 참조의 CreateKey를 참조하세요.

Python

SDK for Python(Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class KeyManager:
    def __init__(self, kms_client):
```

```
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def create_key(self, key_description: str) -> dict[str, any]:
        """
        Creates a key with a user-provided description.

        :param key_description: A description for the key.
        :return: The key ID.
        """
        try:
            key = self.kms_client.create_key(Description=key_description)
["KeyMetadata"]
            self.created_keys.append(key)
            return key
        except ClientError as err:
            logging.error(
                "Couldn't create your key. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 CreateKey를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-kms' # v2: require 'aws-sdk'

# Create a AWS KMS key.
# As long we are only encrypting small amounts of data (4 KiB or less) directly,
# a KMS key is fine for our purposes.
# For larger amounts of data,
# use the KMS key to encrypt a data encryption key (DEK).

client = Aws::KMS::Client.new

resp = client.create_key({
                           tags: [
                             {
                               tag_key: 'CreatedBy',
                               tag_value: 'ExampleUser'
                             }
                           ]
                         })

puts resp.key_metadata.key_id
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 CreateKey를 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
async fn make_key(client: &Client) -> Result<(), Error> {
    let resp = client.create_key().send().await?;

    let id = resp.key_metadata.as_ref().unwrap().key_id();

    println!("Key: {}", id);

    Ok(())
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 CreateKey를 참조하세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        " iv_description = 'Created by the AWS SDK for SAP ABAP'
        oo_result = lo_kms->createkey( iv_description = iv_description ).
        MESSAGE 'KMS key created successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
      CATCH /aws1/cx_kmslimitexceededex.
```

```
        MESSAGE 'Limit exceeded for KMS resources.' TYPE 'E'.
    ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 CreateKey를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **Decrypt** 함께 사용

다음 코드 예시는 Decrypt의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기

CLI

AWS CLI

예 1: 대칭 KMS 키를 사용하여 암호화된 메시지를 복호화하는 방법(Linux 및 macOS)

다음 decrypt 명령 예제에서는 AWS CLI를 사용하여 데이터를 해독하는 권장 방법을 보여줍니다. 이 버전은 대칭 KMS 키로 데이터를 복호화하는 방법을 보여줍니다.

파일에 사이퍼텍스트를 입력합니다. --ciphertext-blob 파라미터 값에는 바이너리 파일에서 데이터를 읽도록 CLI에 지시하는 fileb:// 접두사를 사용합니다. 파일이 현재 디렉터리에 없는 경우 파일의 전체 경로를 입력합니다. 파일에서 AWS CLI 파라미터 값을 읽는 방법에 대한 자세한 내용은 명령줄 인터페이스 사용 설명서의 <https://docs.aws.amazon.com/cli/latest/userguide/cli-usage-parameters-file.html> 파일에서 AWS CLI 파라미터 로드 및 명령AWS 줄 도구 블로그의 로컬 파일 파라미터 모범 사례<https://aws.amazon.com/blogs/developer/best-practices-for-local-file-parameters/>를 참조하세요.암호 텍스트를 해독하려면 KMS 키를 지정하세요. 대칭 KMS 키로 복호화할 때는 --key-id 파라미터가 필요하지 않습니다. AWS KMS는 사이퍼텍스트의 메타데이터에서 데이터를 암호화하는 데 사용된 KMS 키의 키 ID를 가져올 수 있습니다. AWS 그러나 사용 중인 KMS 키를 지정하는 것이 항상 좋습니다. 이렇게 하면 의도한 KMS 키를 사용할 수 있으며 신뢰하지 않는 KMS 키를 사용하여 사이퍼텍스트를 실수로 복호화하는 것을 방지할 수 있습니다. 일반 텍스트 출력을 텍스트 값으로 요청하세요. --query 파라

미터는 CLI에 출력에서 Plaintext 필드 값만 가져오도록 지시합니다. --output 파라미터는 출력을 텍스트로 반환합니다. 일반 텍스트를 Base64로 디코딩하여 파일에 저장합니다. 다음 예시에서는 Plaintext 파라미터 값을 Base64 유틸리티에 파이프(|)로 구분하며 유틸리티가 이를 디코딩합니다. 그런 다음 디코딩된 출력을 ExamplePlaintext 파일로 리디렉션(>)합니다.

이 명령을 실행하기 전에 예제 키 ID를 AWS 계정의 유효한 키 ID로 바꿉니다.

```
aws kms decrypt \
    --ciphertext-blob fileb://ExampleEncryptedFile \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --output text \
    --query Plaintext | base64 \
    --decode > ExamplePlaintextFile
```

이 명령은 출력을 생성하지 않습니다. decrypt 명령의 출력은 base64로 디코딩되어 파일에 저장됩니다.

자세한 내용은 AWS Key Management Service API 참조의 암호화 해제를 참조하세요.

예시 2: 대칭 KMS 키를 사용하여 암호화된 메시지 복호화(Windows 명령 프롬프트)

다음 예시는 certutil 유틸리티를 사용하여 일반 텍스트 데이터를 base64로 디코딩한다는 점을 제외하면 이전 예시와 동일합니다. 이 프로시저에는 다음 예시와 같이 두 개의 명령이 필요합니다.

이 명령을 실행하기 전에 예제 키 ID를 AWS 계정의 유효한 키 ID로 바꿉니다.

```
aws kms decrypt ^
    --ciphertext-blob fileb://ExampleEncryptedFile ^
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab ^
    --output text ^
    --query Plaintext > ExamplePlaintextFile.base64
```

certutil 명령을 실행합니다.

```
certutil -decode ExamplePlaintextFile.base64 ExamplePlaintextFile
```

출력:

```
Input Length = 18
Output Length = 12
```

```
CertUtil: -decode command completed successfully.
```

자세한 내용은 AWS Key Management Service API 참조의 암호화 해제를 참조하세요.

예시 3: 비대칭 KMS 키를 사용하여 암호화된 메시지 복호화(Linux 및 macOS)

다음 `decrypt` 명령 예시에서는 RSA 비대칭 KMS 키로 암호화된 데이터를 복호화하는 방법을 보여줍니다.

비대칭 KMS 키를 사용하는 경우 일반 텍스트를 암호화하는 데 사용되는 알고리즘을 지정하는 `encryption-algorithm` 파라미터가 필요합니다.

이 명령을 실행하기 전에 예제 키 ID를 AWS 계정의 유효한 키 ID로 바꿉니다.

```
aws kms decrypt \
    --ciphertext-blob fileb://ExampleEncryptedFile \
    --key-id 0987dcba-09fe-87dc-65ba-ab0987654321 \
    --encryption-algorithm RSAES_OAEP_SHA_256 \
    --output text \
    --query Plaintext | base64 \
    --decode > ExamplePlaintextFile
```

이 명령은 출력을 생성하지 않습니다. `decrypt` 명령의 출력은 base64로 디코딩되어 파일에 저장됩니다.

자세한 내용은 Key Management Service 개발자 안내서의 AWS KMS의 비대칭 키를 참조하세요. AWS

- API 세부 정보는 AWS CLI 명령 참조의 Decrypt를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
```

```
    * Asynchronously decrypts the given encrypted data using the specified key
ID.
    *
    * @param encryptedData The encrypted data to be decrypted.
    * @param keyId The ID of the key to be used for decryption.
    * @return A CompletableFuture that, when completed, will contain the
decrypted data as a String.
    *         If an error occurs during the decryption process, the
CompletableFuture will complete
    *         exceptionally with the error, and the method will return an empty
String.
    */
   public CompletableFuture<String> decryptDataAsync(SdkBytes encryptedData,
String keyId) {
       DecryptRequest decryptRequest = DecryptRequest.builder()
           .ciphertextBlob(encryptedData)
           .keyId(keyId)
           .build();

       CompletableFuture<DecryptResponse> responseFuture =
getAsyncClient().decrypt(decryptRequest);
       responseFuture.whenComplete((decryptResponse, exception) -> {
           if (exception == null) {
               logger.info("Data decrypted successfully for key ID: " + keyId);
           } else {
               if (exception instanceof KmsException kmsEx) {
                   throw new RuntimeException("KMS error occurred while
decrypting data: " + kmsEx.getMessage(), kmsEx);
               } else {
                   throw new RuntimeException("An unexpected error occurred
while decrypting data: " + exception.getMessage(), exception);
               }
           }
       });

       return responseFuture.thenApply(decryptResponse ->
decryptResponse.plaintext().asString(StandardCharsets.UTF_8));
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 Decrypt를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun encryptData(keyIdValue: String): ByteArray? {
    val text = "This is the text to encrypt by using the AWS KMS Service"
    val myBytes: ByteArray = text.toByteArray()

    val encryptRequest =
        EncryptRequest {
            keyId = keyIdValue
            plaintext = myBytes
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.encrypt(encryptRequest)
        val algorithm: String = response.encryptionAlgorithm.toString()
        println("The encryption algorithm is $algorithm")

        // Return the encrypted data.
        return response.ciphertextBlob
    }
}

suspend fun decryptData(
    encryptedDataVal: ByteArray?,
    keyIdVal: String?,
) {
    val decryptRequest =
        DecryptRequest {
            ciphertextBlob = encryptedDataVal
            keyId = keyIdVal
        }
    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val decryptResponse = kmsClient.decrypt(decryptRequest)
        val myVal = decryptResponse.plaintext
```

```
        // Print the decrypted data.
        print(myVal)
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 Decrypt를 참조하세요.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /***
     * @param string $keyId
     * @param string $ciphertext
     * @param string $algorithm
     * @return Result
     */
    public function decrypt(string $keyId, string $ciphertext, string $algorithm
 = "SYMMETRIC_DEFAULT")
    {
        try{
            return $this->client->decrypt([
                'CiphertextBlob' => $ciphertext,
                'EncryptionAlgorithm' => $algorithm,
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem decrypting the data: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for PHP API 참조의 Decrypt를 참조하세요.

Python

SDK for Python(Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.

        :return: An instance of KeyEncrypt initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def decrypt(self, key_id: str, cipher_text: bytes) -> str:
        """
        Decrypts text previously encrypted with a key.

        :param key_id: The ARN or ID of the key used to decrypt the data.
        :param cipher_text: The encrypted text to decrypt.
        :return: The decrypted text.
        """
        try:
            return self.kms_client.decrypt(KeyId=key_id,
 CiphertextBlob=cipher_text)[
                "Plaintext"
```

```
            ].decode()
        except ClientError as err:
            logger.error(
                "Couldn't decrypt your ciphertext. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 Decrypt를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-kms' # v2: require 'aws-sdk'

# Decrypted blob

blob =
 '01020200785d68faeec386af1057904926253051eb2919d3c16078badf65b808b26dd057c101747cadf3593
blob_packed = [blob].pack('H*')

client = Aws::KMS::Client.new(region: 'us-west-2')

resp = client.decrypt({
                        ciphertext_blob: blob_packed
                      })

puts 'Raw text: '
puts resp.plaintext
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 Decrypt를 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
async fn decrypt_key(client: &Client, key: &str, filename: &str) -> Result<(),
 Error> {
    // Open input text file and get contents as a string
    // input is a base-64 encoded string, so decode it:
    let data = fs::read_to_string(filename)
        .map(|input| {
            base64::decode(input).expect("Input file does not contain valid base
 64 characters.")
        })
        .map(Blob::new);

    let resp = client
        .decrypt()
        .key_id(key)
        .ciphertext_blob(data.unwrap())
        .send()
        .await?;

    let inner = resp.plaintext.unwrap();
    let bytes = inner.as_ref();

    let s = String::from_utf8(bytes.to_vec()).expect("Could not convert to
 UTF-8");

    println!();
    println!("Decoded string:");
    println!("{}", s);

    Ok(())
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 Decrypt를 참조하세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_ciphertext_blob contains the encrypted data
        oo_result = lo_kms->decrypt(
          iv_keyid = iv_key_id
          iv_ciphertextblob = iv_ciphertext_blob
        ).
        MESSAGE 'Text decrypted successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsdisabledexception.
        MESSAGE 'The key is disabled.' TYPE 'E'.
      CATCH /aws1/cx_kmsincorrectkeyex.
        MESSAGE 'Incorrect key for decryption.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 Decrypt를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **DeleteAlias** 함께 사용

다음 코드 예시는 DeleteAlias의 사용 방법을 보여 줍니다.

CLI

AWS CLI

AWS KMS 별칭을 삭제하려면

다음 delete-alias 예시에서는 alias/example-alias 별칭을 삭제합니다. 별칭 이름은 alias/로 시작해야 합니다.

```
aws kms delete-alias \
    --alias-name alias/example-alias
```

이 명령은 출력을 생성하지 않습니다. 별칭을 찾으려면 list-aliases 명령을 사용합니다.

자세한 내용은 AWS Key Management Service 개발자 안내서의 별칭 삭제를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DeleteAlias를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Deletes a specific KMS alias asynchronously.
     *
     * @param aliasName the name of the alias to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous
operation of deleting the specified alias
     */
    public CompletableFuture<Void> deleteSpecificAliasAsync(String aliasName) {
        DeleteAliasRequest deleteAliasRequest = DeleteAliasRequest.builder()
            .aliasName(aliasName)
```

```
            .build();

        return getAsyncClient().deleteAlias(deleteAliasRequest)
            .thenRun(() -> {
                logger.info("Alias {} has been deleted successfully", aliasName);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to delete alias: " +
 aliasName, throwable);
            });
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteAlias를 참조하세요.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /***
     * @param string $aliasName
     * @return void
     */
    public function deleteAlias(string $aliasName)
    {
        try {
            $this->client->deleteAlias([
                'AliasName' => $aliasName,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem deleting the alias: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for PHP API 참조의 DeleteAlias를 참조하세요.

Python

SDK for Python(Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class AliasManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_key = None

    @classmethod
    def from_client(cls) -> "AliasManager":
        """
        Creates an AliasManager instance with a default KMS client.

        :return: An instance of AliasManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def delete_alias(self, alias: str) -> None:
        """
        Deletes an alias.

        :param alias: The alias to delete.
        """
        try:
            self.kms_client.delete_alias(AliasName=alias)
        except ClientError as err:
            logger.error(
```

```
                "Couldn't delete alias %s. Here's why: %s",
                alias,
                err.response["Error"]["Message"],
            )
            raise
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 DeleteAlias를 참조하세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
TRY.
    " iv_alias_name = 'alias/my-key-alias'
    lo_kms->deletealias( iv_aliasname = iv_alias_name ).
    MESSAGE 'Alias deleted successfully.' TYPE 'I'.
  CATCH /aws1/cx_kmsnotfoundexception.
    MESSAGE 'Alias not found.' TYPE 'E'.
  CATCH /aws1/cx_kmskmsinternalex.
    MESSAGE 'An internal error occurred.' TYPE 'E'.
ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 DeleteAlias를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **`DescribeKey`** 함께 사용

다음 코드 예시는 `DescribeKey`의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

/// <summary>
/// Retrieve information about an AWS Key Management Service (AWS KMS) key.
/// You can supply either the key Id or the key Amazon Resource Name (ARN)
/// to the DescribeKeyRequest KeyId property.
/// </summary>
public class DescribeKey
{
    public static async Task Main()
    {
        var keyId = "7c9eccc2-38cb-4c4f-9db3-766ee8dd3ad4";
        var request = new DescribeKeyRequest
        {
            KeyId = keyId,
        };

        var client = new AmazonKeyManagementServiceClient();

        var response = await client.DescribeKeyAsync(request);
        var metadata = response.KeyMetadata;
```

```
            Console.WriteLine($"{metadata.KeyId} created on:
 {metadata.CreationDate}");
            Console.WriteLine($"State: {metadata.KeyState}");
            Console.WriteLine($"{metadata.Description}");
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for .NET API 참조의 DescribeKey를 참조하세요.

CLI

AWS CLI

예시 1: KMS 키에 대한 세부 정보를 찾는 방법

다음 `describe-key` 예제에서는 예제 계정 및 리전에서 Amazon S3의 AWS 관리형 키에 대한 자세한 정보를 가져옵니다. 이 명령을 사용하여 AWS 관리형 키 및 고객 관리형 키에 대한 세부 정보를 찾을 수 있습니다.

KMS 키를 지정하려면 `key-id` 파라미터를 사용합니다. 이 예시에서는 별칭 이름 값을 사용하지만 이 명령에는 키 ID, 키 ARN, 별칭 이름 또는 별칭 ARN을 사용할 수 있습니다.

```
aws kms describe-key \
    --key-id alias/aws/s3
```

출력:

```
{
    "KeyMetadata": {
        "AWSAccountId": "846764612917",
        "KeyId": "b8a9477d-836c-491f-857e-07937918959b",
        "Arn": "arn:aws:kms:us-west-2:846764612917:key/
b8a9477d-836c-491f-857e-07937918959b",
        "CurrentKeyMaterialId":
 "0b7fd7ddbac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6",
        "CreationDate": 2017-06-30T21:44:32.140000+00:00,
        "Enabled": true,
        "Description": "Default KMS key that protects my S3 objects when no other
 key is defined",
```

```
        "KeyUsage": "ENCRYPT_DECRYPT",
        "KeyState": "Enabled",
        "Origin": "AWS_KMS",
        "KeyManager": "AWS",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ]
    }
}
```

자세한 내용은 AWS Key Management Service 개발자 안내서의 키 보기를 참조하세요.

예시 2: RSA 비대칭 KMS 키의 세부 정보 가져오기

다음 `describe-key` 예시에서는 서명 및 확인에 사용되는 비대칭 RSA KMS 키의 세부 정보를 가져옵니다.

```
aws kms describe-key \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

출력:

```
{
    "KeyMetadata": {
        "AWSAccountId": "111122223333",
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "Arn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "CreationDate": "2019-12-02T19:47:14.861000+00:00",
        "CustomerMasterKeySpec": "RSA_2048",
        "Enabled": false,
        "Description": "",
        "KeyState": "Disabled",
        "Origin": "AWS_KMS",
        "MultiRegion": false,
        "KeyManager": "CUSTOMER",
        "KeySpec": "RSA_2048",
        "KeyUsage": "SIGN_VERIFY",
        "SigningAlgorithms": [
            "RSASSA_PKCS1_V1_5_SHA_256",
            "RSASSA_PKCS1_V1_5_SHA_384",
```

```
            "RSASSA_PKCS1_V1_5_SHA_512",
            "RSASSA_PSS_SHA_256",
            "RSASSA_PSS_SHA_384",
            "RSASSA_PSS_SHA_512"
        ]
    }
}
```

예시 3: 다중 리전 복제본 키의 세부 정보 가져오기

다음 describe-key 예시에서는 다중 리전 복제본 키의 메타데이터를 가져옵니다. 이 다중 리전 키는 대칭 암호화 키입니다. 모든 다중 리전 키에 대한 describe-key 명령 출력은 프라이머리 키와 모든 해당 복제본의 정보를 반환합니다.

```
aws kms describe-key \
    --key-id arn:aws:kms:ap-northeast-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab
```

출력:

```
{
    "KeyMetadata": {
        "MultiRegion": true,
        "AWSAccountId": "111122223333",
        "Arn": "arn:aws:kms:ap-northeast-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
        "CreationDate": "2021-06-28T21:09:16.114000+00:00",
        "CurrentKeyMaterialId":
 "0b7fd7ddbac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6",
        "Description": "",
        "Enabled": true,
        "KeyId": "mrk-1234abcd12ab34cd56ef1234567890ab",
        "KeyManager": "CUSTOMER",
        "KeyState": "Enabled",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "Origin": "AWS_KMS",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ],
        "MultiRegionConfiguration": {
            "MultiRegionKeyType": "PRIMARY",
```

```
            "PrimaryKey": {
                "Arn": "arn:aws:kms:us-west-2:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
                "Region": "us-west-2"
            },
            "ReplicaKeys": [
                {
                    "Arn": "arn:aws:kms:eu-west-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
                    "Region": "eu-west-1"
                },
                {
                    "Arn": "arn:aws:kms:ap-northeast-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
                    "Region": "ap-northeast-1"
                },
                {
                    "Arn": "arn:aws:kms:sa-east-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
                    "Region": "sa-east-1"
                }
            ]
        }
    }
}
```

예시 4: HMAC KMS 키의 세부 정보 가져오기

다음 `describe-key` 예시에서는 HMAC KMS 키의 세부 정보를 가져옵니다.

```
aws kms describe-key \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

출력:

```
{
    "KeyMetadata": {
        "AWSAccountId": "123456789012",
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "Arn": "arn:aws:kms:us-
west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "CreationDate": "2022-04-03T22:23:10.194000+00:00",
        "Enabled": true,
```

```
        "Description": "Test key",
        "KeyUsage": "GENERATE_VERIFY_MAC",
        "KeyState": "Enabled",
        "Origin": "AWS_KMS",
        "KeyManager": "CUSTOMER",
        "CustomerMasterKeySpec": "HMAC_256",
        "MacAlgorithms": [
            "HMAC_SHA_256"
        ],
        "MultiRegion": false
    }
}
```

- API 세부 정보는 AWS CLI 명령 참조의 DescribeKey를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Asynchronously checks if a specified key is enabled.
     *
     * @param keyId the ID of the key to check
     * @return a {@link CompletableFuture} that, when completed, indicates
whether the key is enabled or not
     *
     * @throws RuntimeException if an exception occurs while checking the key
state
     */
    public CompletableFuture<Boolean> isKeyEnabledAsync(String keyId) {
        DescribeKeyRequest keyRequest = DescribeKeyRequest.builder()
            .keyId(keyId)
            .build();

        CompletableFuture<DescribeKeyResponse> responseFuture =
getAsyncClient().describeKey(keyRequest);
```

```
        return responseFuture.whenComplete((resp, ex) -> {
            if (resp != null) {
                KeyState keyState = resp.keyMetadata().keyState();
                if (keyState == KeyState.ENABLED) {
                    logger.info("The key is enabled.");
                } else {
                    logger.info("The key is not enabled. Key state: {}",
 keyState);
                }
            } else {
                throw new RuntimeException(ex);
            }
        }).thenApply(resp -> resp.keyMetadata().keyState() == KeyState.ENABLED);
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeKey를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun describeSpecifcKey(keyIdVal: String?) {
    val request =
        DescribeKeyRequest {
            keyId = keyIdVal
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.describeKey(request)
        println("The key description is ${response.keyMetadata?.description}")
        println("The key ARN is ${response.keyMetadata?.arn}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeKey를 참조하세요.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /***
     * @param string $keyId
     * @return array
     */
    public function describeKey(string $keyId)
    {
        try {
            $result = $this->client->describeKey([
                "KeyId" => $keyId,
            ]);
            return $result['KeyMetadata'];
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
resource could not be found.\n";
            }
            throw $caught;
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for PHP API 참조의 DescribeKey를 참조하세요.

Python

SDK for Python(Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def describe_key(self, key_id: str) -> dict[str, any]:
        """
        Describes a key.

        :param key_id: The ARN or ID of the key to describe.
        :return: Information about the key.
        """

        try:
            key = self.kms_client.describe_key(KeyId=key_id)["KeyMetadata"]
            return key
        except ClientError as err:
            logging.error(
                "Couldn't get key '%s'. Here's why: %s",
                key_id,
```

```
                err.response["Error"]["Message"],
            )
            raise
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 DescribeKey를 참조하세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        oo_result = lo_kms->describekey( iv_keyid = iv_key_id ).
        DATA(lo_key) = oo_result->get_keymetadata( ).
        MESSAGE 'Retrieved key information successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 DescribeKey를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 `DisableKey` 함께 사용

다음 코드 예시는 `DisableKey`의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

/// <summary>
/// Disable an AWS Key Management Service (AWS KMS) key and then retrieve
/// the key's status to show that it has been disabled.
/// </summary>
public class DisableKey
{
    public static async Task Main()
    {
        var client = new AmazonKeyManagementServiceClient();

        // The identifier of the AWS KMS key to disable. You can use the
        // key Id or the Amazon Resource Name (ARN) of the AWS KMS key.
        var keyId = "1234abcd-12ab-34cd-56ef-1234567890ab";

        var request = new DisableKeyRequest
        {
            KeyId = keyId,
        };

        var response = await client.DisableKeyAsync(request);
```

```
            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                // Retrieve information about the key to show that it has now
                // been disabled.
                var describeResponse = await client.DescribeKeyAsync(new
DescribeKeyRequest
                {
                    KeyId = keyId,
                });
                Console.WriteLine($"{describeResponse.KeyMetadata.KeyId} - state:
{describeResponse.KeyMetadata.KeyState}");
            }
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for .NET API 참조의 DisableKey를 참조하세요.

CLI

AWS CLI

KMS 키를 일시적으로 비활성화하는 방법

다음 `disable-key` 명령은 고객 관리형 KMS 키를 비활성화합니다. KMS 키를 다시 활성화하려면 `enable-key` 명령을 사용합니다.

```
aws kms disable-key \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS Key Management Service 개발자 안내서의 키 활성화 및 비활성화를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DisableKey를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Asynchronously disables the specified AWS Key Management Service (KMS)
key.
    *
    * @param keyId the ID or Amazon Resource Name (ARN) of the KMS key to be
disabled
    * @return a CompletableFuture that, when completed, indicates that the key
has been disabled successfully
    */
   public CompletableFuture<Void> disableKeyAsync(String keyId) {
       DisableKeyRequest keyRequest = DisableKeyRequest.builder()
           .keyId(keyId)
           .build();

       return getAsyncClient().disableKey(keyRequest)
           .thenRun(() -> {
               logger.info("Key {} has been disabled successfully",keyId);
           })
           .exceptionally(throwable -> {
               throw new RuntimeException("Failed to disable key: " + keyId,
throwable);
           });
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DisableKey를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun disableKey(keyIdVal: String?) {
    val request =
        DisableKeyRequest {
            keyId = keyIdVal
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        kmsClient.disableKey(request)
        println("$keyIdVal was successfully disabled")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DisableKey을 참조하세요.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /***
     * @param string $keyId
     * @return void
     */
```

```
    public function disableKey(string $keyId)
    {
        try {
            $this->client->disableKey([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem disabling the key: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for PHP API 참조의 DisableKey를 참조하세요.

Python

SDK for Python(Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)
```

```
    def disable_key(self, key_id: str) -> None:
        try:
            self.kms_client.disable_key(KeyId=key_id)
        except ClientError as err:
            logging.error(
                "Couldn't disable key '%s'. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 DisableKey를 참조하세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        lo_kms->disablekey( iv_keyid = iv_key_id ).
        MESSAGE 'KMS key disabled successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 DisableKey를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **EnableKey** 함께 사용

다음 코드 예시는 EnableKey의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

/// <summary>
/// Enable an AWS Key Management Service (AWS KMS) key.
/// </summary>
public class EnableKey
{
    public static async Task Main()
    {
        var client = new AmazonKeyManagementServiceClient();

        // The identifier of the AWS KMS key to enable. You can use the
        // key Id or the Amazon Resource Name (ARN) of the AWS KMS key.
        var keyId = "1234abcd-12ab-34cd-56ef-1234567890ab";

        var request = new EnableKeyRequest
```

```
            {
                KeyId = keyId,
            };

            var response = await client.EnableKeyAsync(request);
            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                // Retrieve information about the key to show that it has now
                // been enabled.
                var describeResponse = await client.DescribeKeyAsync(new
DescribeKeyRequest
                {
                    KeyId = keyId,
                });
                Console.WriteLine($"{describeResponse.KeyMetadata.KeyId} - state:
{describeResponse.KeyMetadata.KeyState}");
            }
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for .NET API 참조의 EnableKey를 참조하세요.

CLI

AWS CLI

KMS 키를 활성화하는 방법

다음 enable-key 예시에서는 고객 관리형 키를 활성화합니다. disable-key 명령을 사용하여 일시적으로 비활성화한 KMS 키를 활성화하려면 이와 같은 명령을 사용할 수 있습니다. 또한 삭제 일정이 잡혀 있다가 삭제가 취소되어 사용 중지된 KMS 키를 활성화하는 데도 사용할 수 있습니다.

KMS 키를 지정하려면 key-id 파라미터를 사용합니다. 이 예시에서는 키 ID 값을 사용하지만 이 명령에는 키 ID 또는 키 ARN 값을 사용할 수 있습니다.

이 명령을 실행하기 전에 예시에 나온 키 ID를 유효한 키 핸들로 바꾸세요.

```
aws kms enable-key \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

이 명령은 출력을 생성하지 않습니다. KMS 키가 활성화되었는지 확인하려면 `describe-key` 명령을 사용합니다. `describe-key` 출력의 `KeyState` 및 `Enabled` 필드 값을 확인하세요.

자세한 내용은 AWS Key Management Service 개발자 안내서의 키 활성화 및 비활성화를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 EnableKey를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Asynchronously enables the specified key.
     *
     * @param keyId the ID of the key to enable
     * @return a {@link CompletableFuture} that completes when the key has been
enabled
     */
    public CompletableFuture<Void> enableKeyAsync(String keyId) {
        EnableKeyRequest enableKeyRequest = EnableKeyRequest.builder()
            .keyId(keyId)
            .build();

        CompletableFuture<EnableKeyResponse> responseFuture =
getAsyncClient().enableKey(enableKeyRequest);
        responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {
                logger.info("Key with ID [{}] has been enabled.", keyId);
            } else {
                if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("KMS error occurred while enabling
key: " + kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred
while enabling key: " + exception.getMessage(), exception);
```

```
                }
            }
        });

        return responseFuture.thenApply(response -> null);
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 EnableKey를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun enableKey(keyIdVal: String?) {
    val request =
        EnableKeyRequest {
            keyId = keyIdVal
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        kmsClient.enableKey(request)
        println("$keyIdVal was successfully enabled.")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 EnableKey를 참조하세요.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /***
     * @param string $keyId
     * @return void
     */
    public function enableKey(string $keyId)
    {
        try {
            $this->client->enableKey([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
resource could not be found.\n";
            }
            throw $caught;
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for PHP API 참조의 EnableKey를 참조하세요.

Python

SDK for Python(Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def enable_key(self, key_id: str) -> None:
        """
        Enables a key. Gets the key state after each state change.

        :param key_id: The ARN or ID of the key to enable.
        """
        try:
            self.kms_client.enable_key(KeyId=key_id)
        except ClientError as err:
            logging.error(
                "Couldn't enable key '%s'. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 EnableKey를 참조하세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        lo_kms->enablekey( iv_keyid = iv_key_id ).
        MESSAGE 'KMS key enabled successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 EnableKey를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **EnableKeyRotation** 함께 사용

다음 코드 예시는 EnableKeyRotation의 사용 방법을 보여 줍니다.

CLI

AWS CLI

KMS 키의 자동 교체를 활성화하려면

다음 enable-key-rotation 예시에서는 고객 관리형 KMS 키의 자동 교체를 180일의 교체 기간으로 활성화합니다. KMS 키는 이 명령이 완료된 날짜로부터 1년(약 365일) 후, 그 이후에는 매년 교체됩니다.

--key-id 파라미터는 KMS 키를 식별합니다. 이 예시에서는 키 ARN 값을 사용하지만 KMS 키의 키 ID 또는 ARN을 사용할 수 있습니다. --rotation-period-in-days 파라미터는 각 교체 날짜 사이의 일 수를 지정합니다. 90~2560의 값을 지정하세요. 값을 지정하지 않을 경우 기본값은 365일입니다.

```
aws kms enable-key-rotation \
    --key-id arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab \
    --rotation-period-in-days 180
```

이 명령은 출력을 생성하지 않습니다. KMS 키가 활성화되었는지 확인하려면 get-key-rotation-status 명령을 사용합니다.

자세한 내용은 AWS Key Management Service 개발자 안내서의 Rotating keys 섹션을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 EnableKeyRotation을 참조하세요.

Python

SDK for Python(Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
```

```
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def enable_key_rotation(self, key_id: str) -> None:
        """
        Enables rotation for a key.

        :param key_id: The ARN or ID of the key to enable rotation for.
        """
        try:
            self.kms_client.enable_key_rotation(KeyId=key_id)
        except ClientError as err:
            logging.error(
                "Couldn't enable rotation for key '%s'. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- API 세부 정보는 Python용AWS SDK(Boto3) 참조의 EnableKeyRotation을 참조하세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
```

```
        lo_kms->enablekeyrotation( iv_keyid = iv_key_id ).
        MESSAGE 'Key rotation enabled successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsdisabledexception.
        MESSAGE 'The key is disabled.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmsunsupportedopex.
        MESSAGE 'Operation not supported for this key.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 EnableKeyRotation을 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **Encrypt** 함께 사용

다음 코드 예시는 Encrypt의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기

CLI

AWS CLI

예 1: Linux 또는 macOS에서 파일 콘텐츠를 암호화하는 방법

다음 encrypt 명령은 AWS CLI를 사용하여 데이터를 암호화하는 권장 방법을 보여줍니다.

```
aws kms encrypt \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --plaintext fileb://ExamplePlaintextFile \
    --output text \
    --query CiphertextBlob | base64 \
    --decode > ExampleEncryptedFile
```

이 명령은 여러 가지 작업을 수행합니다.

`--plaintext` 파라미터를 사용하여 암호화할 데이터를 표시합니다. 이 파라미터 값은 base64로 인코딩되어야 합니다. `plaintext` 파라미터 값은 base64로 인코딩되거나 AWS CLI에 파일에서 이진 데이터를 읽도록 지시하는 `fileb://` 접두사를 사용해야 합니다. 파일이 현재 디렉터리에 없는 경우 파일의 전체 경로를 입력합니다. 예: `fileb:///var/tmp/ExamplePlaintextFile` 또는 `fileb://C:\Temp\ExamplePlaintextFile`. 파일에서 AWS CLI 파라미터 값을 읽는 방법에 대한 자세한 내용은 명령AWS 줄 인터페이스 사용 설명서의 파일에서 파라미터 로드 및 명령 AWS 줄 도구 블로그의 로컬 파일 파라미터 모범 사례를 참조하세요. `--output` 및 `--query` 파라미터를 사용하여 명령의 출력을 제어합니다. 이러한 파라미터는 암호화된 데이터를 추출합니다. 는 사이퍼텍스트를 호출했습니다. 명령의 출력에서. 출력 제어에 대한 자세한 내용은 명령AWS 줄 인터페이스 사용 설명서의 명령 출력 제어를 참조하세요. `base64` 유틸리티를 사용하여 추출된 출력을 이진 데이터로 디코딩합니다. 성공적인 `encrypt` 명령으로 반환되는 사이퍼텍스트는 base64로 인코딩된 텍스트입니다. AWS CLI를 사용하여 복호화하려면 먼저이 텍스트를 복호화해야 합니다. 이진 사이퍼텍스트를 파일에 저장합니다. 명령의 마지막 부분(`> ExampleEncryptedFile`)은 이진 사이퍼텍스트를 파일에 저장하여 복호화를 더 쉽게 만듭니다. AWS CLI를 사용하여 데이터를 복호화하는 예제 명령은 복호화 예제를 참조하세요.

예제 2: AWS CLI를 사용하여 Windows에서 데이터 암호화

이 예시는 `base64` 대신 `certutil` 도구를 사용한다는 점을 제외하면 이전 예와 동일합니다. 이 절차에는 다음 예시와 같이 두 개의 명령이 필요합니다.

```
aws kms encrypt \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --plaintext fileb://ExamplePlaintextFile \
    --output text \
    --query CiphertextBlob > C:\Temp\ExampleEncryptedFile.base64

certutil -decode C:\Temp\ExampleEncryptedFile.base64 C:\Temp\ExampleEncryptedFile
```

예 3: 비대칭 KMS 키를 사용한 암호화

다음 `encrypt` 명령은 비대칭 KMS 키를 사용하여 일반 텍스트를 암호화하는 방법을 보여줍니다. `--encryption-algorithm` 파라미터가 필요합니다. 모든 `encrypt` CLI 명령과 마찬가지로 `plaintext` 파라미터는 base64로 인코딩되거나 AWS CLI에 파일에서 이진 데이터를 읽도록 지시하는 `fileb://` 접두사를 사용해야 합니다.

```
aws kms encrypt \
```

```
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --encryption-algorithm RSAES_OAEP_SHA_256 \
    --plaintext fileb://ExamplePlaintextFile \
    --output text \
    --query CiphertextBlob | base64 \
    --decode > ExampleEncryptedFile
```

이 명령은 출력을 생성하지 않습니다.

- API 세부 정보는 AWS CLI 명령 참조의 Encrypt를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Encrypts the given text asynchronously using the specified KMS client and
key ID.
    *
    * @param keyId the ID of the KMS key to use for encryption
    * @param text the text to encrypt
    * @return a CompletableFuture that completes with the encrypted data as an
SdkBytes object
    */
   public CompletableFuture<SdkBytes> encryptDataAsync(String keyId, String
text) {
       SdkBytes myBytes = SdkBytes.fromUtf8String(text);
       EncryptRequest encryptRequest = EncryptRequest.builder()
           .keyId(keyId)
           .plaintext(myBytes)
           .build();

       CompletableFuture<EncryptResponse> responseFuture =
getAsyncClient().encrypt(encryptRequest).toCompletableFuture();
       return responseFuture.whenComplete((response, ex) -> {
           if (response != null) {
```

```
                String algorithm = response.encryptionAlgorithm().toString();
                logger.info("The string was encrypted with algorithm {}.",
 algorithm);
            } else {
                throw new RuntimeException(ex);
            }
        }).thenApply(EncryptResponse::ciphertextBlob);
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 Encrypt를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun encryptData(keyIdValue: String): ByteArray? {
    val text = "This is the text to encrypt by using the AWS KMS Service"
    val myBytes: ByteArray = text.toByteArray()

    val encryptRequest =
        EncryptRequest {
            keyId = keyIdValue
            plaintext = myBytes
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.encrypt(encryptRequest)
        val algorithm: String = response.encryptionAlgorithm.toString()
        println("The encryption algorithm is $algorithm")

        // Return the encrypted data.
        return response.ciphertextBlob
    }
}
```

```
suspend fun decryptData(
    encryptedDataVal: ByteArray?,
    keyIdVal: String?,
) {
    val decryptRequest =
        DecryptRequest {
            ciphertextBlob = encryptedDataVal
            keyId = keyIdVal
        }
    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val decryptResponse = kmsClient.decrypt(decryptRequest)
        val myVal = decryptResponse.plaintext

        // Print the decrypted data.
        print(myVal)
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 Encrypt를 참조하세요.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /***
     * @param string $keyId
     * @param string $text
     * @return Result
     */
    public function encrypt(string $keyId, string $text)
    {
        try {
            return $this->client->encrypt([
                'KeyId' => $keyId,
```

```
                'Plaintext' => $text,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "DisabledException"){
                echo "The request was rejected because the specified KMS key is
 not enabled.\n";
            }
            throw $caught;
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for PHP API 참조의 Encrypt를 참조하세요.

Python

SDK for Python(Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.

        :return: An instance of KeyEncrypt initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def encrypt(self, key_id: str, text: str) -> bytes:
```

```
        """
        Encrypts text by using the specified key.

        :param key_id: The ARN or ID of the key to use for encryption.
        :param text: The text to encrypt.
        :return: The encrypted version of the text.
        """
        try:
            response = self.kms_client.encrypt(KeyId=key_id,
Plaintext=text.encode())
            print(
                f"The string was encrypted with algorithm
{response['EncryptionAlgorithm']}"
            )
            return response["CiphertextBlob"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DisabledException":
                logger.error(
                    "Could not encrypt because the key %s is disabled.", key_id
                )
            else:
                logger.error(
                    "Couldn't encrypt text. Here's why: %s",
                    err.response["Error"]["Message"],
                )
            raise
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 Encrypt를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-kms' # v2: require 'aws-sdk'
```

```
# ARN of the AWS KMS key.
#
# Replace the fictitious key ARN with a valid key ID

keyId = 'arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab'

text = '1234567890'

client = Aws::KMS::Client.new(region: 'us-west-2')

resp = client.encrypt({
                        key_id: keyId,
                        plaintext: text
                      })

# Display a readable version of the resulting encrypted blob.
puts 'Blob:'
puts resp.ciphertext_blob.unpack('H*')
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 Encrypt를 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
async fn encrypt_string(
    verbose: bool,
    client: &Client,
    text: &str,
    key: &str,
    out_file: &str,
) -> Result<(), Error> {
    let blob = Blob::new(text.as_bytes());
```

```
    let resp = client.encrypt().key_id(key).plaintext(blob).send().await?;

    // Did we get an encrypted blob?
    let blob = resp.ciphertext_blob.expect("Could not get encrypted text");
    let bytes = blob.as_ref();

    let s = base64::encode(bytes);

    let mut ofile = File::create(out_file).expect("unable to create file");
    ofile.write_all(s.as_bytes()).expect("unable to write");

    if verbose {
        println!("Wrote the following to {:?}", out_file);
        println!("{}", s);
    }

    Ok(())
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 Encrypt를 참조하세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_plaintext contains the data to encrypt
        oo_result = lo_kms->encrypt(
          iv_keyid = iv_key_id
          iv_plaintext = iv_plaintext
        ).
        MESSAGE 'Text encrypted successfully.' TYPE 'I'.
```

```
      CATCH /aws1/cx_kmsdisabledexception.
        MESSAGE 'The key is disabled.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 암호화를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **GenerateDataKey** 함께 사용

다음 코드 예시는 GenerateDataKey의 사용 방법을 보여 줍니다.

CLI

AWS CLI

예시 1: 256비트 대칭 데이터 키 생성

다음 generate-data-key 예제에서는 외부에서 사용할 256비트 대칭 데이터 키를 요청합니다 AWS. 명령은 즉시 사용 및 삭제할 수 있는 일반 텍스트 데이터 키와 지정된 KMS 키로 암호화된 해당 데이터 키의 사본을 반환합니다. 암호화된 데이터 키를 암호화된 데이터와 함께 안전하게 저장할 수 있습니다.

256비트 데이터 키를 요청하려면 값이 AES_256인 key-spec 파라미터를 사용합니다. 128비트 데이터 키를 요청하려면 값이 AES_128인 key-spec 파라미터를 사용합니다. 다른 모든 데이터 키 길이에는 number-of-bytes 파라미터를 사용합니다.

지정하는 KMS 키는 대칭 암호화 KMS 키, 즉 키 사양 값이 SYMMETRIC_DEFAULT인 KMS 키여야 합니다.

```
aws kms generate-data-key \
    --key-id alias/ExampleAlias \
    --key-spec AES_256
```

출력:

```
{
    "Plaintext": "VdzKNHGzUAzJeRBVY+uUmofUGGiDzyB3+i9fVkh3piw=",
    "KeyId": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyMaterialId":
 "0b7fd7ddbac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6",
    "CiphertextBlob":
 "AQEDAHjRYf5WytIc0C857tFSnBaPn2F8DgfmThbJlGfR8P3WlwAAAH4wfAYJKoZIhvcNAQcGoG8wbQIBADBoBgk
+YdhV8MrkBQPeac0ReRVNDt9qleAt+SHgIRF8P0H+7U="
}
```

Plaintext(일반 텍스트 데이터 키) 및 CiphertextBlob(암호화된 데이터 키)은 base64로 인코딩된 형식으로 반환됩니다.

자세한 내용은 AWS Key Management Service 개발자 안내서의 데이터 키를 참조하세요. 예시 2: 512비트 대칭 데이터 키 생성

다음 generate-data-key 예시에서는 암호화 및 복호화를 위한 512비트 대칭 데이터 키를 요청합니다. 명령은 즉시 사용 및 삭제할 수 있는 일반 텍스트 데이터 키와 지정된 KMS 키로 암호화된 해당 데이터 키의 사본을 반환합니다. 암호화된 데이터 키를 암호화된 데이터와 함께 안전하게 저장할 수 있습니다.

128비트 또는 256비트가 아닌 키 길이를 요청하려면 number-of-bytes 파라미터를 사용합니다. 512비트 데이터 키를 요청하기 위해 다음 예시에서는 값이 64(바이트)인 number-of-bytes 파라미터를 사용합니다.

지정하는 KMS 키는 대칭 암호화 KMS 키, 즉 키 사양 값이 SYMMETRIC_DEFAULT인 KMS 키여야 합니다.

참고: 이 예시의 출력에 있는 값은 잘려서 표시됩니다.

```
aws kms generate-data-key \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --number-of-bytes 64
```

출력:

```
{
    "CiphertextBlob": "AQIBAHi6LtupRpdKl2aJTzkK6FbhOtQkMlQJJH3PdtHvS/y+hAEnX/
QQNmMwDfg2korNMEc8AAACaDCCAmQGCSqGSIb3DQEHBqCCAlUwggJRAgEAMIICSgYJKoZ...",
```

```
    "Plaintext": "ty8Lr0Bk6OF07M2BWt6qbFdNB
+G00ZLtf5MSEb4al3R2UKWGOp06njAwy2n72VRm2m7z/
Pm9Wpbvttz6a4lSo9hgPvKhZ5y6RTm4OovEXiVfBveyX3DQxDzRSwbKDPk/...",
    "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyMaterialId":
 "0b7fd7ddbac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6"
}
```

`Plaintext`(일반 텍스트 데이터 키) 및 `CiphertextBlob`(암호화된 데이터 키)은 base64로 인코딩된 형식으로 반환됩니다.

자세한 내용은 AWS Key Management Service 개발자 안내서의 데이터 키를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 GenerateDataKey를 참조하세요.

Python

SDK for Python(Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)
```

```
    def generate_data_key(self, key_id):
        """
        Generates a symmetric data key that can be used for client-side
 encryption.
        """
        answer = input(
            f"Do you want to generate a symmetric data key from key {key_id} (y/
n)? "
        )
        if answer.lower() == "y":
            try:
                data_key = self.kms_client.generate_data_key(
                    KeyId=key_id, KeySpec="AES_256"
                )
            except ClientError as err:
                logger.error(
                    "Couldn't generate a data key for key %s. Here's why: %s",
                    key_id,
                    err.response["Error"]["Message"],
                )
            else:
                pprint(data_key)
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조용 GenerateDataKey를 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
async fn make_key(client: &Client, key: &str) -> Result<(), Error> {
    let resp = client
```

```
        .generate_data_key()
        .key_id(key)
        .key_spec(DataKeySpec::Aes256)
        .send()
        .await?;

    // Did we get an encrypted blob?
    let blob = resp.ciphertext_blob.expect("Could not get encrypted text");
    let bytes = blob.as_ref();

    let s = base64::encode(bytes);

    println!();
    println!("Data key:");
    println!("{}", s);

    Ok(())
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 GenerateDataKey를 참조하세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_keyspec = 'AES_256'
        oo_result = lo_kms->generatedatakey(
          iv_keyid = iv_key_id
          iv_keyspec = 'AES_256'
        ).
        MESSAGE 'Data key generated successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsdisabledexception.
```

```
        MESSAGE 'The key is disabled.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 GenerateDataKey를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **GenerateDataKeyWithoutPlaintext** 함께 사용

다음 코드 예시는 GenerateDataKeyWithoutPlaintext의 사용 방법을 보여 줍니다.

CLI

AWS CLI

일반 텍스트 키 없이 256비트 대칭 데이터 키 생성

다음 generate-data-key-without-plaintext 예시에서는 AWS외부에서 사용할 256비트 대칭 데이터 키의 암호화된 사본을 요청합니다. 사용할 준비가 되면 AWS KMS를 호출하여 데이터 키를 해독할 수 있습니다.

256비트 데이터 키를 요청하려면 값이 AES_256인 key-spec 파라미터를 사용합니다. 128비트 데이터 키를 요청하려면 값이 AES_128인 key-spec 파라미터를 사용합니다. 다른 모든 데이터 키 길이에는 number-of-bytes 파라미터를 사용합니다.

지정하는 KMS 키는 대칭 암호화 KMS 키, 즉 키 사양 값이 SYMMETRIC_DEFAULT인 KMS 키여야 합니다.

```
aws kms generate-data-key-without-plaintext \
    --key-id "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab" \
    --key-spec AES_256
```

출력:

```
{
```

```
    "CiphertextBlob":
 "AQEDAHjRYf5WytIc0C857tFSnBaPn2F8DgfmThbJlGfR8P3WlwAAAH4wfAYJKoZIhvcNAQcGoG8wbQIBADBoBgk
    "KeyId": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyMaterialId":
 "0b7fd7ddbac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6"
}
```

`CiphertextBlob`(암호화된 데이터 키)은 base64로 인코딩된 형식으로 반환됩니다.

자세한 내용은 AWS Key Management Service 개발자 안내서의 데이터 키를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 GenerateDataKeyWithoutPlaintext를 참조하세요.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
async fn make_key(client: &Client, key: &str) -> Result<(), Error> {
    let resp = client
        .generate_data_key_without_plaintext()
        .key_id(key)
        .key_spec(DataKeySpec::Aes256)
        .send()
        .await?;

    // Did we get an encrypted blob?
    let blob = resp.ciphertext_blob.expect("Could not get encrypted text");
    let bytes = blob.as_ref();

    let s = base64::encode(bytes);

    println!();
    println!("Data key:");
    println!("{}", s);
```

```
    Ok(())
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 GenerateDataKeyWithoutPlaintext를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **GenerateRandom** 함께 사용

다음 코드 예시는 GenerateRandom의 사용 방법을 보여 줍니다.

CLI

AWS CLI

예시 1: 256비트 무작위 바이트 문자열 생성(Linux 또는 macOS)

다음 generate-random 예시에서는 256비트(32바이트), base64로 인코딩된 무작위 바이트 문자열을 생성합니다. 이 예시에서는 바이트 문자열을 디코딩하여 무작위 파일에 저장합니다.

이 명령을 실행할 때는 number-of-bytes 파라미터를 사용하여 무작위 값의 길이를 바이트 단위로 지정해야 합니다.

이 명령을 실행할 때는 KMS 키를 지정하지 않습니다. 무작위 바이트 문자열은 어떤 KMS 키와도 관련이 없습니다.

기본적으로 AWS KMS는 난수를 생성합니다. 그러나 사용자 지정 키 스토어를 지정하면 사용자 지정 키 스토어와 연결된 AWS CloudHSM 클러스터에 임의 바이트 문자열이 생성됩니다.

이 예시에서는 다음 파라미터와 값을 사용합니다.

값이 인 필수 --number-of-bytes 파라미터를 사용하여 32바이트(256비트)를 32 요청합니다. string.It 값이 인 --output 파라미터를 사용하여 AWS CLI에 출력을 텍스트로 반환하도록 text 지시합니다. JSON.It 대신 --query parameter를 사용하여 base64 유틸리티에 대한 명령의 출력을 response.It 파이프(|)에서 Plaintext 속성 값을 추출합니다. 는 추출된 output.It 리디렉션 연산자(>)를 사용하여 디코딩된 바이트 문자열을 ExampleRandomfile.It 리디렉션 연산자(>)를 사용하여 바이너리 사이퍼텍스트를 파일에 저장합니다.

```
aws kms generate-random \
```

```
    --number-of-bytes 32 \
    --output text \
    --query Plaintext | base64 --decode > ExampleRandom
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS Key Management Service API 참조의 GenerateRandom을 참조하세요.

예시 2: 256비트 무작위 수 생성(Windows 명령 프롬프트)

다음 예시에서는 `generate-random` 명령을 사용하여 256비트(32바이트), base64로 인코딩된 무작위 바이트 문자열을 생성합니다. 이 예시에서는 바이트 문자열을 디코딩하여 무작위 파일에 저장합니다. 이 예시는 Windows의 `certutil` 유틸리티를 사용하여 무작위 바이트 문자열을 base64로 디코딩한 다음 파일에 저장한다는 점을 제외하면 이전 예와 동일합니다.

먼저 base64로 인코딩된 무작위 바이트 문자열을 생성하여 임시 파일 `ExampleRandom.base64`에 저장합니다.

```
aws kms generate-random \
    --number-of-bytes 32 \
    --output text \
    --query Plaintext > ExampleRandom.base64
```

`generate-random` 명령의 출력이 파일에 저장되기 때문에 이 예시에서는 출력이 생성되지 않습니다.

이제 `certutil -decode` 명령을 사용하여 `ExampleRandom.base64` 파일에서 base64로 인코딩된 바이트 문자열을 디코딩합니다. 그런 다음 디코딩된 바이트 문자열을 `ExampleRandom` 파일에 저장합니다.

```
certutil -decode ExampleRandom.base64 ExampleRandom
```

출력:

```
Input Length = 18
Output Length = 12
CertUtil: -decode command completed successfully.
```

자세한 내용은 AWS Key Management Service API 참조의 GenerateRandom을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 GenerateRandom을 참조하세요.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
async fn make_string(client: &Client, length: i32) -> Result<(), Error> {
    let resp = client
        .generate_random()
        .number_of_bytes(length)
        .send()
        .await?;

    // Did we get an encrypted blob?
    let blob = resp.plaintext.expect("Could not get encrypted text");
    let bytes = blob.as_ref();

    let s = base64::encode(bytes);

    println!();
    println!("Data key:");
    println!("{}", s);

    Ok(())
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 GenerateRandom을 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **GetKeyPolicy** 함께 사용

다음 코드 예시는 GetKeyPolicy의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기

CLI

AWS CLI

한 KMS 키에서 다른 KMS 키로 키 정책 복사

다음 `get-key-policy` 예시에서는 한 KMS 키에서 키 정책을 가져와 텍스트 파일에 저장합니다. 그런 다음 텍스트 파일을 정책 입력으로 사용하여 다른 KMS 키의 정책을 대체합니다.

`put-key-policy`의 `--policy` 파라미터에는 문자열이 필요하므로 출력을 JSON 대신 텍스트 문자열로 반환하려면 `--output text` 옵션을 사용해야 합니다.

```
aws kms get-key-policy \
    --policy-name default \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --query Policy \
    --output text > policy.txt

aws kms put-key-policy \
    --policy-name default \
    --key-id 0987dcba-09fe-87dc-65ba-ab0987654321 \
    --policy file://policy.txt
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS KMS API 참조의 PutKeyPolicy를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 GetKeyPolicy를 참조하세요.

Python

SDK for Python(Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class KeyPolicy:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyPolicy":
        """
        Creates a KeyPolicy instance with a default KMS client.

        :return: An instance of KeyPolicy initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def get_policy(self, key_id: str) -> dict[str, str]:
        """
        Gets the policy of a key.

        :param key_id: The ARN or ID of the key to query.
        :return: The key policy as a dict.
        """
        if key_id != "":
            try:
                response = self.kms_client.get_key_policy(
                    KeyId=key_id,
                )
                policy = json.loads(response["Policy"])
            except ClientError as err:
                logger.error(
                    "Couldn't get policy for key %s. Here's why: %s",
```

```
                    key_id,
                    err.response["Error"]["Message"],
                )
                raise
            else:
                pprint(policy)
                return policy
        else:
            print("Skipping get policy demo.")
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 GetKeyPolicy를 참조하세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        oo_result = lo_kms->getkeypolicy(
          iv_keyid = iv_key_id
          iv_policyname = 'default'
        ).
        MESSAGE 'Retrieved key policy successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 GetKeyPolicy를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 `ListAliases` 함께 사용

다음 코드 예시는 `ListAliases`의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.KeyManagementService;
    using Amazon.KeyManagementService.Model;

    /// <summary>
    /// List the AWS Key Management Service (AWS KMS) aliases that have been
defined for
    /// the keys in the same AWS Region as the default user. If you want to list
    /// the aliases in a different Region, pass the Region to the client
    /// constructor.
    /// </summary>
    public class ListAliases
    {
        public static async Task Main()
        {
            var client = new AmazonKeyManagementServiceClient();
            var request = new ListAliasesRequest();
            var response = new ListAliasesResponse();
```

```
            do
            {
                response = await client.ListAliasesAsync(request);

                response.Aliases.ForEach(alias =>
                {
                    Console.WriteLine($"Created: {alias.CreationDate} Last
 Update: {alias.LastUpdatedDate} Name: {alias.AliasName}");
                });

                request.Marker = response.NextMarker;
            }
            while (response.Truncated);
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for .NET API 참조의 ListAliases를 참조하세요.

CLI

AWS CLI

예제 1: AWS 계정 및 리전의 모든 별칭 나열

다음 예제에서는 `list-aliases` 명령을 사용하여 AWS 계정의 기본 리전에 있는 모든 별칭을 나열합니다. 출력에는 AWS 관리형 KMS 키 및 고객 관리형 KMS 키와 연결된 별칭이 포함됩니다.

```
aws kms list-aliases
```

출력:

```
{
    "Aliases": [
        {
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/testKey",
            "AliasName": "alias/testKey",
            "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
```

```
        },
        {
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/FinanceDept",
            "AliasName": "alias/FinanceDept",
            "TargetKeyId": "0987dcba-09fe-87dc-65ba-ab0987654321"
        },
        {
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/aws/dynamodb",
            "AliasName": "alias/aws/dynamodb",
            "TargetKeyId": "1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d"
        },
        {
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/aws/ebs",
            "AliasName": "alias/aws/ebs",
            "TargetKeyId": "0987ab65-43cd-21ef-09ab-87654321cdef"
        },
        ...
    ]
}
```

예시 2: 특정 KMS 키의 모든 별칭 나열

다음 예시에서는 `list-aliases` 명령과 해당 `key-id` 파라미터를 사용하여 특정 KMS 키와 연결된 별칭을 모두 나열합니다.

각 별칭은 단 하나의 KMS 키와 연결되지만 KMS 키는 여러 개의 별칭을 가질 수 있습니다. KMS 콘솔에는 각 AWS KMS 키에 대해 별칭이 하나만 나열되므로이 명령은 매우 유용합니다. KMS 키의 모든 별칭을 찾으려면 `list-aliases` 명령을 사용해야 합니다.

이 예시에서는 `--key-id` 파라미터에 KMS 키의 키 ID를 사용하지만 이 명령에는 키 ID, 키 ARN, 별칭 이름 또는 별칭 ARN을 사용할 수 있습니다.

```
aws kms list-aliases --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

출력:

```
{
    "Aliases": [
        {
            "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/oregon-test-
key",
```

```
            "AliasName": "alias/oregon-test-key"
        },
        {
            "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/project121-
test",
            "AliasName": "alias/project121-test"
        }
    ]
}
```

자세한 내용은 AWS Key Management Service 개발자 안내서의 별칭으로 작업을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListAliases를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Asynchronously lists all the aliases in the current AWS account.
     *
     * @return a {@link CompletableFuture} that completes when the list of
aliases has been processed
     */
    public CompletableFuture<Object> listAllAliasesAsync() {
        ListAliasesRequest aliasesRequest = ListAliasesRequest.builder()
            .limit(15)
            .build();

        ListAliasesPublisher paginator =
getAsyncClient().listAliasesPaginator(aliasesRequest);
        return paginator.subscribe(response -> {
                response.aliases().forEach(alias ->
                    logger.info("The alias name is: " + alias.aliasName())
                );
```

```
            })
            .thenApply(v -> null)
            .exceptionally(ex -> {
                if (ex.getCause() instanceof KmsException) {
                    KmsException e = (KmsException) ex.getCause();
                    throw new RuntimeException("A KMS exception occurred: " +
 e.getMessage());
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
 ex.getMessage());
                }
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListAliases를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listAllAliases() {
    val request =
        ListAliasesRequest {
            limit = 15
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listAliases(request)
        response.aliases?.forEach { alias ->
            println("The alias name is ${alias.aliasName}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListAliases를 참조하세요.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /***
     * @param string $keyId
     * @param int $limit
     * @return ResultPaginator
     */
    public function listAliases(string $keyId = "", int $limit = 0)
    {
        $args = [];
        if($keyId){
            $args['KeyId'] = $keyId;
        }
        if($limit){
            $args['Limit'] = $limit;
        }
        try{
            return $this->client->getPaginator("ListAliases", $args);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidMarkerException"){
                echo "The request was rejected because the marker that specifies
where pagination should next begin is not valid.\n";
            }
            throw $caught;
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for PHP API 참조의 ListAliases를 참조하세요.

Python

SDK for Python(Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class AliasManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_key = None

    @classmethod
    def from_client(cls) -> "AliasManager":
        """
        Creates an AliasManager instance with a default KMS client.

        :return: An instance of AliasManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def list_aliases(self, page_size: int) -> None:
        """
        Lists aliases for the current account.
        :param page_size: The number of aliases to list per page.
        """
        try:
            alias_paginator = self.kms_client.get_paginator("list_aliases")
            for alias_page in alias_paginator.paginate(
                PaginationConfig={"PageSize": page_size}
            ):
                print(f"Here are {page_size} aliases:")
                pprint(alias_page["Aliases"])
                if alias_page["Truncated"]:
                    answer = input(
```

```
                    f"Do you want to see the next {page_size} aliases (y/n)?
"
                )
                if answer.lower() != "y":
                    break
            else:
                print("That's all your aliases!")
        except ClientError as err:
            logging.error(
                "Couldn't list your aliases. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 ListAliases를 참조하세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
TRY.
    oo_result = lo_kms->listaliases( ).
    MESSAGE 'Retrieved KMS aliases list.' TYPE 'I'.
  CATCH /aws1/cx_kmskmsinternalex.
    MESSAGE 'An internal error occurred.' TYPE 'E'.
ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 ListAliases를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 `ListGrants` 함께 사용

다음 코드 예시는 `ListGrants`의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.KeyManagementService;
    using Amazon.KeyManagementService.Model;

    /// <summary>
    /// List the AWS Key Management Service (AWS KMS) grants that are associated
with
    /// a specific key.
    /// </summary>
    public class ListGrants
    {
        public static async Task Main()
        {
            // The identifier of the AWS KMS key to disable. You can use the
            // key Id or the Amazon Resource Name (ARN) of the AWS KMS key.
            var keyId = "1234abcd-12ab-34cd-56ef-1234567890ab";
            var client = new AmazonKeyManagementServiceClient();
            var request = new ListGrantsRequest
            {
                KeyId = keyId,
            };
```

```
            var response = new ListGrantsResponse();

            do
            {
                response = await client.ListGrantsAsync(request);

                response.Grants.ForEach(grant =>
                {
                    Console.WriteLine($"{grant.GrantId}");
                });

                request.Marker = response.NextMarker;
            }
            while (response.Truncated);
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for .NET API 참조의 ListGrants를 참조하세요.

CLI

AWS CLI

AWS KMS 키에 대한 권한 부여를 보려면

다음 `list-grants` 예시에서는 계정의 Amazon DynamoDB에 대해 지정된 AWS 관리형 KMS 키에 대한 모든 권한 부여를 표시합니다. 이 권한 부여를 사용하면 DynamoDB가 사용자 대신 KMS 키를 사용하여 DynamoDB 테이블을 디스크에 쓰기 전에 암호화할 수 있습니다. 이와 같은 명령을 사용하여 AWS 계정 및 리전의 AWS 관리형 KMS 키 및 고객 관리형 KMS 키에 대한 권한 부여를 볼 수 있습니다.

이 명령은 키 ID가 있는 `key-id` 파라미터를 사용하여 KMS 키를 식별합니다. 키 ID 또는 키 ARN을 사용하여 KMS 키를 식별할 수 있습니다. AWS 관리형 KMS 키의 키 ID 또는 키 ARN을 가져오려면 `list-keys` 또는 `list-aliases` 명령을 사용합니다.

```
aws kms list-grants \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

출력은 권한 부여가 Amazon DynamoDB에 KMS 키를 암호화 작업에 사용할 권한을 부여하고 KMS 키(DescribeKey)에 대한 세부 정보를 보고 권한 부여를 사용 중지(RetireGrant)할 수 있는 권한을 부여함을 보여줍니다. EncryptionContextSubset 제약 조건은 이러한 권한을 지정된 암호화 컨텍스트 페어를 포함하는 요청으로 제한합니다. 따라서 권한 부여의 권한은 지정된 계정 및 DynamoDB 테이블에만 유효합니다.

```
{
    "Grants": [
        {
            "Constraints": {
                "EncryptionContextSubset": {
                    "aws:dynamodb:subscriberId": "123456789012",
                    "aws:dynamodb:tableName": "Services"
                }
            },
            "IssuingAccount": "arn:aws:iam::123456789012:root",
            "Name": "8276b9a6-6cf0-46f1-b2f0-7993a7f8c89a",
            "Operations": [
                "Decrypt",
                "Encrypt",
                "GenerateDataKey",
                "ReEncryptFrom",
                "ReEncryptTo",
                "RetireGrant",
                "DescribeKey"
            ],
            "GrantId":
 "1667b97d27cf748cf05b487217dd4179526c949d14fb3903858e25193253fe59",
            "KeyId": "arn:aws:kms:us-
west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "RetiringPrincipal": "dynamodb.us-west-2.amazonaws.com",
            "GranteePrincipal": "dynamodb.us-west-2.amazonaws.com",
            "CreationDate": "2021-05-13T18:32:45.144000+00:00"
        }
    ]
}
```

자세한 내용은 AWS Key Management Service 개발자 안내서의 AWS KMS의 권한 부여를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListGrants를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Asynchronously displays the grant IDs for the specified key ID.
     *
     * @param keyId the ID of the AWS KMS key for which to list the grants
     * @return a {@link CompletableFuture} that, when completed, will be null
if the operation succeeded, or will throw a {@link RuntimeException} if the
operation failed
     * @throws RuntimeException if there was an error listing the grants, either
due to an {@link KmsException} or an unexpected error
     */
    public CompletableFuture<Object> displayGrantIdsAsync(String keyId) {
        ListGrantsRequest grantsRequest = ListGrantsRequest.builder()
            .keyId(keyId)
            .limit(15)
            .build();

        ListGrantsPublisher paginator =
getAsyncClient().listGrantsPaginator(grantsRequest);
        return paginator.subscribe(response -> {
                response.grants().forEach(grant -> {
                    logger.info("The grant Id is: " + grant.grantId());
                });
            })
            .thenApply(v -> null)
            .exceptionally(ex -> {
                Throwable cause = ex.getCause();
                if (cause instanceof KmsException) {
                    throw new RuntimeException("Failed to list grants: " +
cause.getMessage(), cause);
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
cause.getMessage(), cause);
```

```
                }
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListGrants를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun displayGrantIds(keyIdVal: String?) {
    val request =
        ListGrantsRequest {
            keyId = keyIdVal
            limit = 15
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listGrants(request)
        response.grants?.forEach { grant ->
            println("The grant Id is ${grant.grantId}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListGrants를 참조하세요.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /***
     * @param string $keyId
     * @return Result
     */
    public function listGrants(string $keyId)
    {
        try{
            return $this->client->listGrants([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "    The request was rejected because the specified entity
or resource could not be found.\n";
            }
            throw $caught;
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for PHP API 참조의 ListGrants를 참조하세요.

Python

SDK for Python(Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "GrantManager":
        """
        Creates a GrantManager instance with a default KMS client.

        :return: An instance of GrantManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def list_grants(self, key_id):
        """
        Lists grants for a key.

        :param key_id: The ARN or ID of the key to query.
        :return: The grants for the key.
        """
        try:
            paginator = self.kms_client.get_paginator("list_grants")
            grants = []
            page_iterator = paginator.paginate(KeyId=key_id)
            for page in page_iterator:
                grants.extend(page["Grants"])

            print(f"Grants for key {key_id}:")
            pprint(grants)
```

```
            return grants
        except ClientError as err:
            logger.error(
                "Couldn't list grants for key %s. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 ListGrants를 참조하세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        oo_result = lo_kms->listgrants( iv_keyid = iv_key_id ).
        MESSAGE 'Retrieved grants list.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 ListGrants를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **ListKeyPolicies** 함께 사용

다음 코드 예시는 ListKeyPolicies의 사용 방법을 보여 줍니다.

CLI

AWS CLI

KMS 키의 키 정책 이름 가져오기

다음 list-key-policies 예시에서는 예시 계정 및 리전의 고객 관리형 키에 대한 키 정책 이름을 가져옵니다. 이 명령을 사용하여 AWS 관리형 키 및 고객 관리형 키에 대한 키 정책의 이름을 찾을 수 있습니다.

유효한 키 정책 이름은 default뿐이므로 이 명령은 유용하지 않습니다.

KMS 키를 지정하려면 key-id 파라미터를 사용합니다. 이 예시에서는 키 ID 값을 사용하지만 이 명령에는 키 ID 또는 키 ARN을 사용할 수 있습니다.

```
aws kms list-key-policies \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

출력:

```
{
    "PolicyNames": [
    "default"
    ]
}
```

AWS KMS 키 정책에 대한 자세한 내용은 Key AWS Management Service 개발자 안내서의 AWS KMS에서 키 정책 사용을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListKeyPolicies를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Asynchronously retrieves the key policy for the specified key ID and
policy name.
    *
    * @param keyId       the ID of the AWS KMS key for which to retrieve the
policy
    * @param policyName the name of the key policy to retrieve
    * @return a {@link CompletableFuture} that, when completed, contains the key
policy as a {@link String}
    */
   public CompletableFuture<String> getKeyPolicyAsync(String keyId, String
policyName) {
       GetKeyPolicyRequest policyRequest = GetKeyPolicyRequest.builder()
           .keyId(keyId)
           .policyName(policyName)
           .build();

       return getAsyncClient().getKeyPolicy(policyRequest)
           .thenApply(response -> {
               String policy = response.policy();
               logger.info("The response is: " + policy);
               return policy;
           })
           .exceptionally(ex -> {
               throw new RuntimeException("Failed to get key policy", ex);
           });
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListKeyPolicies를 참조하세요.

Python

SDK for Python(Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class KeyPolicy:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyPolicy":
        """
        Creates a KeyPolicy instance with a default KMS client.

        :return: An instance of KeyPolicy initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def list_policies(self, key_id):
        """
        Lists the names of the policies for a key.

        :param key_id: The ARN or ID of the key to query.
        """
        try:
            policy_names = self.kms_client.list_key_policies(KeyId=key_id)[
                "PolicyNames"
            ]
        except ClientError as err:
            logging.error(
                "Couldn't list your policies. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise
```

```
        else:
            print(f"The policies for key {key_id} are:")
            pprint(policy_names)
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 ListKeyPolicies를 참조하세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        oo_result = lo_kms->listkeypolicies( iv_keyid = iv_key_id ).
        MESSAGE 'Retrieved key policies list.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 ListKeyPolicies를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **ListKeys** 함께 사용

다음 코드 예시는 ListKeys의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.KeyManagementService;
    using Amazon.KeyManagementService.Model;

    /// <summary>
    /// List the AWS Key Managements Service (AWS KMS) keys for the AWS Region
    /// of the default user. To list keys in another AWS Region, supply the
Region
    /// as a parameter to the client constructor.
    /// </summary>
    public class ListKeys
    {
        public static async Task Main()
        {
            var client = new AmazonKeyManagementServiceClient();
            var request = new ListKeysRequest();
            var response = new ListKeysResponse();

            do
            {
                response = await client.ListKeysAsync(request);

                response.Keys.ForEach(key =>
                {
                    Console.WriteLine($"ID: {key.KeyId}, {key.KeyArn}");
```

```
                });

                // Set the Marker property when response.Truncated is true
                // in order to get the next keys.
                request.Marker = response.NextMarker;
            }
            while (response.Truncated);
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for .NET API 참조의 ListKeys를 참조하세요.

CLI

AWS CLI

계정 및 리전에서 KMS 키를 가져오는 방법

다음 `list-keys` 예시에서는 계정과 리전의 KMS 키를 가져옵니다. 이 명령은 AWS 관리형 키와 고객 관리형 키를 모두 반환합니다.

```
aws kms list-keys
```

출력:

```
{
    "Keys": [
        {
            "KeyArn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
        },
        {
            "KeyArn": "arn:aws:kms:us-
west-2:111122223333:key/0987dcba-09fe-87dc-65ba-ab0987654321",
            "KeyId": "0987dcba-09fe-87dc-65ba-ab0987654321"
        },
        {
            "KeyArn": "arn:aws:kms:us-
east-2:111122223333:key/1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d",
```

```
            "KeyId": "1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d"
        }
    ]
}
```

자세한 내용은 AWS Key Management Service 개발자 안내서의 키 보기를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListKeys를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.services.kms.KmsAsyncClient;
import software.amazon.awssdk.services.kms.model.ListKeysRequest;
import software.amazon.awssdk.services.kms.paginators.ListKeysPublisher;
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloKMS {
    public static void main(String[] args) {
        listAllKeys();
    }

    public static void listAllKeys() {
        KmsAsyncClient kmsAsyncClient = KmsAsyncClient.builder()
            .build();
        ListKeysRequest listKeysRequest = ListKeysRequest.builder()
```

```
            .limit(15)
            .build();

        /*
         * The `subscribe` method is required when using paginator methods in the
AWS SDK
         * because paginator methods return an instance of a `ListKeysPublisher`,
which is
         * based on a reactive stream. This allows asynchronous retrieval of
paginated
         * results as they become available. By subscribing to the stream, we can
process
         * each page of results as they are emitted.
         */
        ListKeysPublisher keysPublisher =
kmsAsyncClient.listKeysPaginator(listKeysRequest);
        CompletableFuture<Void> future = keysPublisher
            .subscribe(r -> r.keys().forEach(key ->
                System.out.println("The key ARN is: " + key.keyArn() + ". The key
Id is: " + key.keyId())))
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    System.err.println("Error occurred: " +
exception.getMessage());
                } else {
                    System.out.println("Successfully listed all keys.");
                }
            });

        try {
            future.join();
        } catch (Exception e) {
            System.err.println("Failed to list keys: " + e.getMessage());
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListKeys를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listAllKeys() {
    val request =
        ListKeysRequest {
            limit = 15
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listKeys(request)
        response.keys?.forEach { key ->
            println("The key ARN is ${key.keyArn}")
            println("The key Id is ${key.keyId}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListKeys를 참조하세요.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /***
```

```
     * @return array
     */
    public function listKeys()
    {
        try {
            $contents = [];
            $paginator = $this->client->getPaginator("ListKeys");
            foreach($paginator as $result){
                foreach ($result['Content'] as $object) {
                    $contents[] = $object;
                }
            }
            return $contents;
        }catch(KmsException $caught){
            echo "There was a problem listing the keys: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for PHP API 참조의 ListKeys를 참조하세요.

Python

SDK for Python(Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
```

```
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def list_keys(self):
        """
        Lists the keys for the current account by using a paginator.
        """
        try:
            page_size = 10
            print("\nLet's list your keys.")
            key_paginator = self.kms_client.get_paginator("list_keys")
            for key_page in key_paginator.paginate(PaginationConfig={"PageSize":
10}):
                print(f"Here are {len(key_page['Keys'])} keys:")
                pprint(key_page["Keys"])
                if key_page["Truncated"]:
                    answer = input(
                        f"Do you want to see the next {page_size} keys (y/n)? "
                    )
                    if answer.lower() != "y":
                        break
                else:
                    print("That's all your keys!")
        except ClientError as err:
            logging.error(
                "Couldn't list your keys. Here's why: %s",
                err.response["Error"]["Message"],
            )
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 ListKeys를 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
async fn show_keys(client: &Client) -> Result<(), Error> {
    let resp = client.list_keys().send().await?;

    let keys = resp.keys.unwrap_or_default();

    let len = keys.len();

    for key in keys {
        println!("Key ARN: {}", key.key_arn.as_deref().unwrap_or_default());
    }

    println!();
    println!("Found {} keys", len);

    Ok(())
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 ListKeys를 참조하세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        oo_result = lo_kms->listkeys( ).
        MESSAGE 'Retrieved KMS keys list.' TYPE 'I'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 ListKeys를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **PutKeyPolicy** 함께 사용

다음 코드 예시는 PutKeyPolicy의 사용 방법을 보여 줍니다.

CLI

AWS CLI

KMS 키의 키 정책 변경

다음 put-key-policy 예시에서는 고객 관리형 키의 키 정책을 변경합니다.

시작하려면 키 정책을 생성하고 로컬 JSON 파일로 저장합니다. 이 예시에서 파일은 key_policy.json입니다. 키 정책을 policy 파라미터의 문자열 값으로 지정할 수도 있습니다.

이 키 정책의 첫 번째 문은 AWS 계정에 IAM 정책을 사용하여 KMS 키에 대한 액세스를 제어할 수 있는 권한을 부여합니다. 두 번째 문은 test-user 사용자에게 KMS 키에서 describe-key 및 list-keys 명령을 실행할 권한을 부여합니다.

key_policy.json의 콘텐츠:

```
{
    "Version":"2012-10-17",
    "Id" : "key-default-1",
    "Statement" : [
        {
            "Sid" : "Enable IAM User Permissions",
            "Effect" : "Allow",
```

```
            "Principal" : {
                "AWS" : "arn:aws:iam::111122223333:root"
            },
            "Action" : "kms:*",
            "Resource" : "*"
        },
        {
            "Sid" : "Allow Use of Key",
            "Effect" : "Allow",
            "Principal" : {
                "AWS" : "arn:aws:iam::111122223333:user/test-user"
            },
            "Action" : [
                "kms:DescribeKey",
                "kms:ListKeys"
            ],
            "Resource" : "*"
        }
    ]
}
```

KMS 키를 식별하기 위해 이 예시에서는 키 ID를 사용하지만 키 ARN을 사용할 수도 있습니다. 키 정책을 지정하기 위해 이 명령은 `policy` 파라미터를 사용합니다. 정책이 파일에 있음을 나타내기 위해 필수 `file://` 접두사를 사용합니다. 이 접두사는 지원되는 모든 운영 체제에서 파일을 식별하는 데 필요합니다. 마지막으로, 이 명령은 값이 `default`인 `policy-name` 파라미터를 사용합니다. 정책 이름을 지정하지 않을 경우 기본값은 `default`입니다. 유일한 유효 값은 `default`입니다.

```
aws kms put-key-policy \
    --policy-name default \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --policy file://key_policy.json
```

이 명령은 출력을 생성하지 않습니다. 명령이 적용되었는지 확인하려면 `get-key-policy` 명령을 사용합니다. 다음 예시 명령은 동일한 KMS 키에 대한 키 정책을 가져옵니다. 값이 `text`인 `output` 파라미터는 읽기 쉬운 텍스트 형식을 반환합니다.

```
aws kms get-key-policy \
    --policy-name default \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --output text
```

출력:

```
{
    "Version":"2012-10-17",
    "Id" : "key-default-1",
    "Statement" : [
        {
            "Sid" : "Enable IAM User Permissions",
            "Effect" : "Allow",
            "Principal" : {
                "AWS" : "arn:aws:iam::111122223333:root"
            },
            "Action" : "kms:*",
            "Resource" : "*"
            },
            {
            "Sid" : "Allow Use of Key",
            "Effect" : "Allow",
            "Principal" : {
                "AWS" : "arn:aws:iam::111122223333:user/test-user"
            },
            "Action" : [ "kms:Describe", "kms:List" ],
            "Resource" : "*"
        }
    ]
}
```

자세한 내용은 AWS Key Management Service 개발자 안내서의 키 정책 변경을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 PutKeyPolicy를 참조하세요.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /***
     * @param string $keyId
     * @param string $policy
     * @return void
     */
    public function putKeyPolicy(string $keyId, string $policy)
    {
        try {
            $this->client->putKeyPolicy([
                'KeyId' => $keyId,
                'Policy' => $policy,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem replacing the key policy: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 PutKeyPolicy를 참조하세요.

Python

SDK for Python(Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class KeyPolicy:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyPolicy":
        """
        Creates a KeyPolicy instance with a default KMS client.
```

```
        :return: An instance of KeyPolicy initialized with the default KMS
client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def set_policy(self, key_id: str, policy: dict[str, any]) -> None:
        """
        Sets the policy of a key. Setting a policy entirely overwrites the
existing
        policy, so care is taken to add a statement to the existing list of
statements
        rather than simply writing a new policy.

        :param key_id: The ARN or ID of the key to set the policy to.
        :param policy: The existing policy of the key.
        :return: None
        """
        principal = input(
            "Enter the ARN of an IAM role to set as the principal on the policy:
"
        )
        if key_id != "" and principal != "":
            # The updated policy replaces the existing policy. Add a new
statement to
            # the list along with the original policy statements.
            policy["Statement"].append(
                {
                    "Sid": "Allow access for ExampleRole",
                    "Effect": "Allow",
                    "Principal": {"AWS": principal},
                    "Action": [
                        "kms:Encrypt",
                        "kms:GenerateDataKey*",
                        "kms:Decrypt",
                        "kms:DescribeKey",
                        "kms:ReEncrypt*",
                    ],
                    "Resource": "*",
                }
            )
            try:
```

```
                self.kms_client.put_key_policy(KeyId=key_id,
 Policy=json.dumps(policy))
            except ClientError as err:
                logger.error(
                    "Couldn't set policy for key %s. Here's why %s",
                    key_id,
                    err.response["Error"]["Message"],
                )
                raise
            else:
                print(f"Set policy for key {key_id}.")
        else:
            print("Skipping set policy demo.")
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 PutKeyPolicy를 참조하세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_policy = '{"Version":"2012-10-17",        "Statement": [...]}'
        lo_kms->putkeypolicy(
          iv_keyid = iv_key_id
          iv_policyname = 'default'
          iv_policy = iv_policy
        ).
        MESSAGE 'Key policy updated successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmsmalformedplydocex.
        MESSAGE 'Malformed policy document.' TYPE 'E'.
```

```
    CATCH /aws1/cx_kmskmsinternalex.
      MESSAGE 'An internal error occurred.' TYPE 'E'.
  ENDTRY.
```

- API 세부 정보는 SDK for SAP ABAP API 참조의 PutKeyPolicy를 참조하세요. AWS

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **ReEncrypt** 함께 사용

다음 코드 예시는 ReEncrypt의 사용 방법을 보여 줍니다.

CLI

AWS CLI

예시 1: 다른 대칭 KMS 키를 사용하여 암호화된 메시지 다시 암호화(Linux 및 macOS)

다음 re-encrypt 명령 예제에서는 AWS CLI를 사용하여 데이터를 다시 암호화하는 권장 방법을 보여줍니다.

파일에 사이퍼텍스트를 입력합니다. --ciphertext-blob 파라미터 값에는 바이너리 파일에서 데이터를 읽도록 CLI에 지시하는 fileb:// 접두사를 사용합니다. 파일이 현재 디렉터리에 없는 경우 파일의 전체 경로를 입력합니다. 파일에서 AWS CLI 파라미터 값을 읽는 방법에 대한 자세한 내용은 AWS 명령줄 인터페이스 사용 설명서의 파일에서 AWS CLI 파라미터 로드 및 AWS 명령줄 도구 블로그의 로컬 파일 파라미터 모범 사례를 참조하세요. 사이퍼텍스트를 복호화하는 소스 KMS 키를 지정합니다. 대칭 암호화 KMS 키로 복호화할 때는 --source-key-id 파라미터가 필요하지 않습니다. AWS KMS는 사이퍼텍스트 블롭의 메타데이터에서 데이터를 암호화하는 데 사용된 KMS 키를 가져올 수 있습니다. 그러나 사용 중인 KMS 키를 지정하는 것이 항상 좋습니다. 이렇게 하면 의도한 KMS 키를 사용할 수 있으며 신뢰하지 않는 KMS 키를 사용하여 사이퍼텍스트를 실수로 복호화하는 것을 방지할 수 있습니다. 데이터를 다시 암호화하는 대상 KMS 키를 지정하세요. --destination-key-id 파라미터는 항상 필요합니다. 이 예시에서는 키 ARN을 사용하지만 모든 유효한 키 식별자를 사용할 수 있습니다. 일반 텍스트 출력을 텍스트 값으로 요청하세요. --query 파라미터는 출력에서 Plaintext 필드 값만 가져오도록 CLI에 지시합니다. --output 파라미터는 출력을 텍스트로 반환합니다. 일반 텍스트를 Base64로 디코딩하여 파일에 저장합니다. 다음 예시에서는 Plaintext 파라미터 값을 Base64 유틸리티에 파이프(|)로 구분하며 유틸리티가 이를 디코딩합니다. 그런 다음 디코딩된 출력을 ExamplePlaintext 파일로 리디렉션(>)합니다.

이 명령을 실행하기 전에 예제 키 IDs AWS 계정의 유효한 키 식별자로 바꿉니다.

```
aws kms re-encrypt \
    --ciphertext-blob fileb://ExampleEncryptedFile \
    --source-key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --destination-key-id 0987dcba-09fe-87dc-65ba-ab0987654321 \
    --query CiphertextBlob \
    --output text | base64 --decode > ExampleReEncryptedFile
```

이 명령은 출력을 생성하지 않습니다. `re-encrypt` 명령의 출력은 base64로 디코딩되어 파일에 저장됩니다.

자세한 내용은 AWS Key Management Service API 참조의 ReEncrypt를 참조하세요.

예시 2: 다른 대칭 KMS 키를 사용하여 암호화된 메시지 다시 암호화(Windows 명령 프롬프트)

다음 `re-encrypt` 명령 예시는 `certutil` 유틸리티를 사용하여 일반 텍스트 데이터를 base64로 디코딩한다는 점을 제외하면 이전 예시와 동일합니다. 이 프로시저에는 다음 예시와 같이 두 개의 명령이 필요합니다.

이 명령을 실행하기 전에 예제 키 ID를 AWS 계정의 유효한 키 ID로 바꿉니다.

```
aws kms re-encrypt ^
    --ciphertext-blob fileb://ExampleEncryptedFile ^
    --source-key-id 1234abcd-12ab-34cd-56ef-1234567890ab ^
    --destination-key-id 0987dcba-09fe-87dc-65ba-ab0987654321 ^
    --query CiphertextBlob ^
    --output text > ExampleReEncryptedFile.base64
```

그런 다음 `certutil` 유틸리티를 사용합니다.

```
certutil -decode ExamplePlaintextFile.base64 ExamplePlaintextFile
```

출력:

```
Input Length = 18
Output Length = 12
CertUtil: -decode command completed successfully.
```

자세한 내용은 AWS Key Management Service API 참조의 ReEncrypt를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ReEncrypt를 참조하세요.

Python

SDK for Python(Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.

        :return: An instance of KeyEncrypt initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def re_encrypt(self, source_key_id, cipher_text):
        """
        Takes ciphertext previously encrypted with one key and reencrypt it by
 using
        another key.

        :param source_key_id: The ARN or ID of the original key used to encrypt
 the
                              ciphertext.
        :param cipher_text: The encrypted ciphertext.
        :return: The ciphertext encrypted by the second key.
        """
        destination_key_id = input(
            f"Your ciphertext is currently encrypted with key {source_key_id}. "
```

```
            f"Enter another key ID or ARN to reencrypt it: "
        )
        if destination_key_id != "":
            try:
                cipher_text = self.kms_client.re_encrypt(
                    SourceKeyId=source_key_id,
                    DestinationKeyId=destination_key_id,
                    CiphertextBlob=cipher_text,
                )["CiphertextBlob"]
            except ClientError as err:
                logger.error(
                    "Couldn't reencrypt your ciphertext. Here's why: %s",
                    err.response["Error"]["Message"],
                )
            else:
                print(f"Reencrypted your ciphertext as: {cipher_text}")
                return cipher_text
        else:
            print("Skipping reencryption demo.")
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 ReEncrypt를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-kms' # v2: require 'aws-sdk'

# Human-readable version of the ciphertext of the data to reencrypt.

blob =
 '01020200785d68faeec386af1057904926253051eb2919d3c16078badf65b808b26dd057c101747cadf3593
sourceCiphertextBlob = [blob].pack('H*')
```

```
# Replace the fictitious key ARN with a valid key ID

destinationKeyId = 'arn:aws:kms:us-
west-2:111122223333:key/0987dcba-09fe-87dc-65ba-ab0987654321'

client = Aws::KMS::Client.new(region: 'us-west-2')

resp = client.re_encrypt({
                           ciphertext_blob: sourceCiphertextBlob,
                           destination_key_id: destinationKeyId
                         })

# Display a readable version of the resulting re-encrypted blob.
puts 'Blob:'
puts resp.ciphertext_blob.unpack('H*')
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 ReEncrypt를 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
async fn reencrypt_string(
    verbose: bool,
    client: &Client,
    input_file: &str,
    output_file: &str,
    first_key: &str,
    new_key: &str,
) -> Result<(), Error> {
    // Get blob from input file
    // Open input text file and get contents as a string
    // input is a base-64 encoded string, so decode it:
    let data = fs::read_to_string(input_file)
```

```
        .map(|input_file| base64::decode(input_file).expect("invalid base 64"))
        .map(Blob::new);

    let resp = client
        .re_encrypt()
        .ciphertext_blob(data.unwrap())
        .source_key_id(first_key)
        .destination_key_id(new_key)
        .send()
        .await?;

    // Did we get an encrypted blob?
    let blob = resp.ciphertext_blob.expect("Could not get encrypted text");
    let bytes = blob.as_ref();

    let s = base64::encode(bytes);
    let o = &output_file;

    let mut ofile = File::create(o).expect("unable to create file");
    ofile.write_all(s.as_bytes()).expect("unable to write");

    if verbose {
        println!("Wrote the following to {}:", output_file);
        println!("{}", s);
    } else {
        println!("Wrote base64-encoded output to {}", output_file);
    }

    Ok(())
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 ReEncrypt를 참조하세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        " iv_source_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_destination_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/5678dcba-56cd-78ef-90ab-5678901234cd'
        " iv_ciphertext_blob contains the encrypted data
        oo_result = lo_kms->reencrypt(
          iv_sourcekeyid = iv_source_key_id
          iv_destinationkeyid = iv_destination_key_id
          iv_ciphertextblob = iv_ciphertext_blob
        ).
        MESSAGE 'Ciphertext reencrypted successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsdisabledexception.
        MESSAGE 'The key is disabled.' TYPE 'E'.
      CATCH /aws1/cx_kmsincorrectkeyex.
        MESSAGE 'Incorrect source key for decryption.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 ReEncrypt를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **RetireGrant** 함께 사용

다음 코드 예시는 RetireGrant의 사용 방법을 보여 줍니다.

CLI

AWS CLI

고객 마스터 키에 대한 권한 부여 사용 중지

다음 retire-grant 예시에서는 KMS 키에서 권한 부여를 삭제합니다.

다음 예시 명령은 grant-id 및 key-id 파라미터를 지정합니다. key-id 파라미터 값은 KMS 키의 키 ARN이어야 합니다.

```
aws kms retire-grant \
    --grant-id 1234a2345b8a4e350500d432bccf8ecd6506710e1391880c4f7f7140160c9af3 \
    --key-id arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab
```

이 명령은 출력을 생성하지 않습니다. 권한 부여가 사용 중지되었는지 확인하려면 `list-grants` 명령을 사용합니다.

자세한 내용은 AWS Key Management Service 개발자 안내서의 권한 부여 사용 중지 및 취소를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 RetireGrant를 참조하세요.

Python

SDK for Python(Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "GrantManager":
        """
        Creates a GrantManager instance with a default KMS client.

        :return: An instance of GrantManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def retire_grant(self, grant):
        """
```

```
        Retires a grant so that it can no longer be used.

        :param grant: The grant to retire.
        """
        try:
            self.kms_client.retire_grant(GrantToken=grant["GrantToken"])
        except ClientError as err:
            logger.error(
                "Couldn't retire grant %s. Here's why: %s",
                grant["GrantId"],
                err.response["Error"]["Message"],
            )
        else:
            print(f"Grant {grant['GrantId']} retired.")
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 RetireGrant를 참조하세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
TRY.
    " iv_grant_token = 'AQpAM2RhZ...'
    lo_kms->retiregrant( iv_granttoken = iv_grant_token ).
    MESSAGE 'Grant retired successfully.' TYPE 'I'.
  CATCH /aws1/cx_kmsnotfoundexception.
    MESSAGE 'Grant not found.' TYPE 'E'.
  CATCH /aws1/cx_kmsinvgranttokenex.
    MESSAGE 'Invalid grant token.' TYPE 'E'.
  CATCH /aws1/cx_kmskmsinternalex.
    MESSAGE 'An internal error occurred.' TYPE 'E'.
ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 RetireGrant를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **RevokeGrant** 함께 사용

다음 코드 예시는 RevokeGrant의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기

CLI

AWS CLI

고객 마스터 키에 대한 권한 부여 사용 중지

다음 revoke-grant 예시에서는 KMS 키에서 권한 부여를 삭제합니다. 다음 예시 명령은 grant-id 및 key-id 파라미터를 지정합니다. key-id 파라미터 값은 KMS 키의 키 ID 또는 키 ARN일 수 있습니다.

```
aws kms revoke-grant \
    --grant-id 1234a2345b8a4e350500d432bccf8ecd6506710e1391880c4f7f7140160c9af3 \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

이 명령은 출력을 생성하지 않습니다. 권한 부여가 취소되었는지 확인하려면 list-grants 명령을 사용합니다.

자세한 내용은 AWS Key Management Service 개발자 안내서의 권한 부여 사용 중지 및 취소를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 RevokeGrant를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Revokes a grant for the specified AWS KMS key asynchronously.
    *
    * @param keyId   The ID or key ARN of the AWS KMS key.
    * @param grantId The identifier of the grant to be revoked.
    * @return A {@link CompletableFuture} representing the asynchronous
operation of revoking the grant.
    *         The {@link CompletableFuture} will complete with a {@link
RevokeGrantResponse} object
    *         if the operation is successful, or with a {@code null} value if an
error occurs.
    */
   public CompletableFuture<RevokeGrantResponse> revokeKeyGrantAsync(String
keyId, String grantId) {
       RevokeGrantRequest grantRequest = RevokeGrantRequest.builder()
           .keyId(keyId)
           .grantId(grantId)
           .build();

       CompletableFuture<RevokeGrantResponse> responseFuture =
getAsyncClient().revokeGrant(grantRequest);
       responseFuture.whenComplete((response, exception) -> {
           if (exception == null) {
               logger.info("Grant ID: [" + grantId + "] was successfully
revoked!");
           } else {
               if (exception instanceof KmsException kmsEx) {
                   if (kmsEx.getMessage().contains("Grant does not exist")) {
                       logger.info("The grant ID '" + grantId + "' does not
exist. Moving on...");
                   } else {
```

```
                        throw new RuntimeException("KMS error occurred: " +
 kmsEx.getMessage(), kmsEx);
                    }
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
 exception.getMessage(), exception);
                }
            }
        });

        return responseFuture;
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 RevokeGrant를 참조하세요.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /***
     * @param string $grantId
     * @param string $keyId
     * @return void
     */
    public function revokeGrant(string $grantId, string $keyId)
    {
        try{
            $this->client->revokeGrant([
                'GrantId' => $grantId,
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem with revoking the grant: {$caught-
>getAwsErrorMessage()}.\n";
```

```
            throw $caught;
        }
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 RevokeGrant를 참조하세요.

Python

SDK for Python(Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "GrantManager":
        """
        Creates a GrantManager instance with a default KMS client.

        :return: An instance of GrantManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def revoke_grant(self, key_id: str, grant_id: str) -> None:
        """
        Revokes a grant so that it can no longer be used.

        :param key_id: The ARN or ID of the key associated with the grant.
        :param grant_id: The ID of the grant to revoke.
        """
        try:
```

```
            self.kms_client.revoke_grant(KeyId=key_id, GrantId=grant_id)
        except ClientError as err:
            logger.error(
                "Couldn't revoke grant %s. Here's why: %s",
                grant_id,
                err.response["Error"]["Message"],
            )
            raise
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 RevokeGrant를 참조하세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_grant_id = '1a2b3c4d5e6f7g8h9i0j1k2l3m4n5o6p'
        lo_kms->revokegrant(
          iv_keyid = iv_key_id
          iv_grantid = iv_grant_id
        ).
        MESSAGE 'Grant revoked successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Grant or key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmsinvalidgrantidex.
        MESSAGE 'Invalid grant ID.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 RevokeGrant를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **ScheduleKeyDeletion** 함께 사용

다음 코드 예시는 ScheduleKeyDeletion의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기

CLI

AWS CLI

고객 관리형 키 삭제를 예약하는 방법

다음 schedule-key-deletion 예시에서는 지정된 고객 관리형 KMS 키가 15일 후에 삭제되도록 예약합니다.

--key-id 파라미터는 KMS 키를 식별합니다. 이 예시에서는 키 ARN 값을 사용하지만 KMS 키의 키 ID 또는 ARN을 사용할 수 있습니다. --pending-window-in-days 파라미터는 7~30일의 대기 기간을 지정합니다. 기본 대기 기간은 30일입니다. 이 예제에서는 값을 15로 지정합니다.이 값은 명령이 완료된 후 15일이 지나면에 KMS 키를 영구적으로 삭제 AWS 하도록 지시합니다.

```
aws kms schedule-key-deletion \
    --key-id arn:aws:kms:us-
west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab \
    --pending-window-in-days 15
```

응답에는 키 ARN, 키 상태, 대기 기간(PendingWindowInDays), 삭제 날짜(Unix 시간)이 포함됩니다. 삭제 날짜를 현지 시간으로 보려면 AWS KMS 콘솔을 사용합니다. PendingDeletion 키 상태의 KMS 키는 암호화 작업에 사용될 수 없습니다.

```
{
```

```
    "KeyId": "arn:aws:kms:us-
west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "DeletionDate": "2022-06-18T23:43:51.272000+00:00",
    "KeyState": "PendingDeletion",
    "PendingWindowInDays": 15
}
```

자세한 내용은 AWS Key Management Service 개발자 안내서의 키 삭제를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ScheduleKeyDeletion을 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Deletes a KMS key asynchronously.
    *
    * <p><strong>Warning:</strong> Deleting a KMS key is a destructive and
potentially dangerous operation.
    * When a KMS key is deleted, all data that was encrypted under the KMS key
becomes unrecoverable.
    * This means that any files, databases, or other data that were encrypted
using the deleted KMS key
    * will become permanently inaccessible. Exercise extreme caution when
deleting KMS keys.</p>
    *
    * @param keyId the ID of the KMS key to delete
    * @return a {@link CompletableFuture} that completes when the key deletion
is scheduled
    */
   public CompletableFuture<Void> deleteKeyAsync(String keyId) {
       ScheduleKeyDeletionRequest deletionRequest =
ScheduleKeyDeletionRequest.builder()
           .keyId(keyId)
           .pendingWindowInDays(7)
```

```
            .build();

        return getAsyncClient().scheduleKeyDeletion(deletionRequest)
            .thenRun(() -> {
                logger.info("Key {} will be deleted in 7 days", keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to schedule key deletion for
 key ID: " + keyId, throwable);
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ScheduleKeyDeletion을 참조하세요.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /***
     * @param string $keyId
     * @param int $pendingWindowInDays
     * @return void
     */
    public function scheduleKeyDeletion(string $keyId, int $pendingWindowInDays =
 7)
    {
        try {
            $this->client->scheduleKeyDeletion([
                'KeyId' => $keyId,
                'PendingWindowInDays' => $pendingWindowInDays,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem scheduling the key deletion: {$caught-
>getAwsErrorMessage()}\n";
```

```
            throw $caught;
        }
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 ScheduleKeyDeletion을 참조하세요.

Python

SDK for Python(Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def delete_key(self, key_id: str, window: int) -> None:
        """
        Deletes a list of keys.

        Warning:
        Deleting a KMS key is a destructive and potentially dangerous operation.
 When a KMS key is deleted,
```

```
        all data that was encrypted under the KMS key is unrecoverable.

        :param key_id: The ARN or ID of the key to delete.
        :param window: The waiting period, in days, before the KMS key is
 deleted.
        """

        try:
            self.kms_client.schedule_key_deletion(
                KeyId=key_id, PendingWindowInDays=window
            )
        except ClientError as err:
            logging.error(
                "Couldn't delete key %s. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 ScheduleKeyDeletion를 참조하세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_pending_window_days = 7
        oo_result = lo_kms->schedulekeydeletion(
          iv_keyid = iv_key_id
          iv_pendingwindowindays = iv_pending_window_days
```

```
        ).
        MESSAGE 'Key scheduled for deletion.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API 세부 정보는 SDK for SAP ABAP API 참조의 ScheduleKeyDeletion을 참조하세요. AWS

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **`Sign`** 함께 사용

다음 코드 예시는 `Sign`의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기

CLI

AWS CLI

예제 1: 메시지에 대한 디지털 서명을 생성하려면

다음 `sign` 예시에서는 짧은 메시지에 대한 암호화 서명을 생성합니다. 명령 출력에는 `verify` 명령을 사용하여 확인할 수 있는 base-64로 인코딩된 `Signature` 필드가 포함됩니다.

서명할 메시지와 비대칭 KMS 키가 지원하는 서명 알고리즘을 지정해야 합니다. KMS 키의 서명 알고리즘을 가져오려면 `describe-key` 명령을 사용합니다.

AWS CLI v2에서 `message` 파라미터 값은 Base64-encoded되어야 합니다. 또는 메시지를 파일에 저장하고 AWS CLI에 파일에서 이진 데이터를 읽도록 지시하는 `fileb://` 접두사를 사용할 수 있습니다.

이 명령을 실행하기 전에 예제 키 ID를 AWS 계정의 유효한 키 ID로 바꿉니다. 키 ID는 SIGN_VERIFY라는 키가 사용된 비대칭 KMS 키를 나타내야 합니다.

```
msg=(echo 'Hello World' | base64)

aws kms sign \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --message fileb://UnsignedMessage \
    --message-type RAW \
    --signing-algorithm RSASSA_PKCS1_V1_5_SHA_256
```

출력:

```
{
    "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "Signature": "ABCDEFhpyVYyTxbafE74ccSvEJLJr3zuoV1Hfymz4qv+/
fxmxNLA7SE1SiF8lHw80fKZZ3bJ...",
    "SigningAlgorithm": "RSASSA_PKCS1_V1_5_SHA_256"
}
```

AWS KMS에서 비대칭 KMS 키를 사용하는 방법에 대한 자세한 내용은 Key AWS Management Service 개발자 안내서의 AWS KMS의 비대칭 키를 참조하세요.

예시 2: 디지털 서명을 파일에 저장(Linux 및 macOS)

다음 `sign` 예시에서는 로컬 파일에 저장된 짧은 메시지에 대한 암호화 서명을 생성합니다. 또한 명령은 응답에서 `Signature` 속성을 가져오고 Base64로 디코딩하여 ExampleSignature 파일에 저장합니다. 서명을 확인하는 `verify` 명령에서 서명 파일을 사용할 수 있습니다.

`sign` 명령에는 Base64로 인코딩된 메시지와 비대칭 KMS 키가 지원하는 서명 알고리즘이 필요합니다. KMS 키가 지원하는 서명 알고리즘을 가져오려면 `describe-key` 명령을 사용합니다.

이 명령을 실행하기 전에 예제 키 ID를 AWS 계정의 유효한 키 ID로 바꿉니다. 키 ID는 SIGN_VERIFY라는 키가 사용된 비대칭 KMS 키를 나타내야 합니다.

```
echo 'hello world' | base64 > EncodedMessage

aws kms sign \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --message fileb://EncodedMessage \
    --message-type RAW \
    --signing-algorithm RSASSA_PKCS1_V1_5_SHA_256 \
    --output text \
```

```
    --query Signature | base64 --decode > ExampleSignature
```

이 명령은 출력을 생성하지 않습니다. 이 예시에서는 출력의 `Signature` 속성을 추출하여 파일에 저장합니다.

AWS KMS에서 비대칭 KMS 키를 사용하는 방법에 대한 자세한 내용은 Key AWS Management Service 개발자 안내서의 AWS KMS의 비대칭 키를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 Sign을 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Asynchronously signs and verifies data using AWS KMS.
    *
    * <p>The method performs the following steps:
    * <ol>
    *     <li>Creates an AWS KMS key with the specified key spec, key usage, and
origin.</li>
    *     <li>Signs the provided message using the created KMS key and the
RSASSA-PSS-SHA-256 algorithm.</li>
    *     <li>Verifies the signature of the message using the created KMS key
and the RSASSA-PSS-SHA-256 algorithm.</li>
    * </ol>
    *
    * @return a {@link CompletableFuture} that completes with the result of the
signature verification,
    *         {@code true} if the signature is valid, {@code false} otherwise.
    * @throws KmsException if any error occurs during the KMS operations.
    * @throws RuntimeException if an unexpected error occurs.
    */
   public CompletableFuture<Boolean> signVerifyDataAsync() {
       String signMessage = "Here is the message that will be digitally signed";
```

```
        // Create an AWS KMS key used to digitally sign data.
        CreateKeyRequest createKeyRequest = CreateKeyRequest.builder()
            .keySpec(KeySpec.RSA_2048)
            .keyUsage(KeyUsageType.SIGN_VERIFY)
            .origin(OriginType.AWS_KMS)
            .build();

        return getAsyncClient().createKey(createKeyRequest)
            .thenCompose(createKeyResponse -> {
                String keyId = createKeyResponse.keyMetadata().keyId();

                SdkBytes messageBytes = SdkBytes.fromString(signMessage,
Charset.defaultCharset());
                SignRequest signRequest = SignRequest.builder()
                    .keyId(keyId)
                    .message(messageBytes)
                    .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
                    .build();

                return getAsyncClient().sign(signRequest)
                    .thenCompose(signResponse -> {
                        byte[] signedBytes =
signResponse.signature().asByteArray();

                        VerifyRequest verifyRequest = VerifyRequest.builder()
                            .keyId(keyId)

.message(SdkBytes.fromByteArray(signMessage.getBytes(Charset.defaultCharset())))

.signature(SdkBytes.fromByteBuffer(ByteBuffer.wrap(signedBytes)))

.signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
                            .build();

                        return getAsyncClient().verify(verifyRequest)
                            .thenApply(verifyResponse -> {
                                return (boolean) verifyResponse.signatureValid();
                            });
                    });
            })
            .exceptionally(throwable -> {
               throw new RuntimeException("Failed to sign or verify data",
throwable);
            });
```

```
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 Sign을 참조하세요.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /***
     * @param string $keyId
     * @param string $message
     * @param string $algorithm
     * @return Result
     */
    public function sign(string $keyId, string $message, string $algorithm)
    {
        try {
            return $this->client->sign([
                'KeyId' => $keyId,
                'Message' => $message,
                'SigningAlgorithm' => $algorithm,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem signing the data: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 Sign을 참조하세요.

Python

SDK for Python(Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.

        :return: An instance of KeyEncrypt initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def sign(self, key_id: str, message: str) -> str:
        """
        Signs a message with a key.

        :param key_id: The ARN or ID of the key to use for signing.
        :param message: The message to sign.
        :return: The signature of the message.
        """
        try:
            return self.kms_client.sign(
                KeyId=key_id,
                Message=message.encode(),
                SigningAlgorithm="RSASSA_PSS_SHA_256",
            )["Signature"]
        except ClientError as err:
            logger.error(
```

```
                "Couldn't sign your message. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise
```

- API 세부 정보는 Python용AWS SDK(Boto3) 참조의 Sign을 참조하세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab' (asymmetric key)
        " iv_message contains the message to sign
        " iv_signing_algorithm = 'RSASSA_PSS_SHA_256'
        oo_result = lo_kms->sign(
          iv_keyid = iv_key_id
          iv_message = iv_message
          iv_signingalgorithm = iv_signing_algorithm
        ).
        MESSAGE 'Message signed successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsdisabledexception.
        MESSAGE 'The key is disabled.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmsinvalidkeyusageex.
        MESSAGE 'Key cannot be used for signing.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API 세부 정보는 Sign in AWS SDK for SAP ABAP API 참조를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **TagResource** 함께 사용

다음 코드 예시는 TagResource의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기

CLI

AWS CLI

KMS 키에 태그를 추가하려면

다음 tag-resource 예시에서는 고객 관리형 KMS 키에 "Purpose":"Test" 및 "Dept":"IT" 태그를 추가합니다. 이와 같은 태그를 사용하여 KMS 키에 레이블을 지정하고 권한 및 감사를 위한 KMS 키 범주를 생성할 수 있습니다.

KMS 키를 지정하려면 key-id 파라미터를 사용합니다. 이 예시에서는 키 ID 값을 사용하지만 이 명령에는 키 ID 또는 키 ARN을 사용할 수 있습니다.

```
aws kms tag-resource \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --tags TagKey='Purpose',TagValue='Test' TagKey='Dept',TagValue='IT'
```

이 명령은 출력을 생성하지 않습니다. AWS KMS KMS 키의 태그를 보려면 list-resource-tags 명령을 사용합니다.

AWS KMS에서 태그를 사용하는 방법에 대한 자세한 내용은 Key AWS Management Service 개발자 안내서의 키 태그 지정을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 TagResource를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Asynchronously tags a KMS key with a specific tag.
     *
     * @param keyId the ID of the KMS key to be tagged
     * @return a {@link CompletableFuture} that completes when the tagging
operation is finished
     */
    public CompletableFuture<Void> tagKMSKeyAsync(String keyId) {
        Tag tag = Tag.builder()
            .tagKey("Environment")
            .tagValue("Production")
            .build();

        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .keyId(keyId)
            .tags(tag)
            .build();

        return getAsyncClient().tagResource(tagResourceRequest)
            .thenRun(() -> {
                logger.info("{} key was tagged", keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to tag the KMS key",
throwable);
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 TagResource를 참조하세요.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * @param string $keyId
     * @param array $tags
     * @return void
     */
    public function tagResource(string $keyId, array $tags)
    {
        try {
            $this->client->tagResource([
                'KeyId' => $keyId,
                'Tags' => $tags,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem applying the tag(s): {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 TagResource를 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def tag_resource(self, key_id: str, tag_key: str, tag_value: str) -> None:
        """
        Add or edit tags on a customer managed key.

        :param key_id: The ARN or ID of the key to enable rotation for.
        :param tag_key: Key for the tag.
        :param tag_value: Value for the tag.
        """
        try:
            self.kms_client.tag_resource(
                KeyId=key_id, Tags=[{"TagKey": tag_key, "TagValue": tag_value}]
            )
        except ClientError as err:
            logging.error(
                "Couldn't add a tag for the key '%s'. Here's why: %s",
```

```
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 TagResource를 참조하세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    DATA lt_tags TYPE /aws1/cl_kmstag=>tt_taglist.

    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_tag_key = 'Environment'
        " iv_tag_value = 'Production'
        APPEND NEW /aws1/cl_kmstag(
          iv_tagkey = iv_tag_key
          iv_tagvalue = iv_tag_value
        ) TO lt_tags.

        lo_kms->tagresource(
          iv_keyid = iv_key_id
          it_tags = lt_tags
        ).
        MESSAGE 'Tag added to KMS key successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmstagexception.
        MESSAGE 'Invalid tag format.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
```

```
    ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 TagResource를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **UpdateAlias** 함께 사용

다음 코드 예시는 UpdateAlias의 사용 방법을 보여 줍니다.

CLI

AWS CLI

별칭을 다른 KMS 키에 연결

다음 update-alias 예시에서는 별칭 alias/test-key를 다른 KMS 키와 연결합니다.

--alias-name 파라미터는 별칭을 지정합니다. 별칭 이름 값은 alias/로 시작해야 합니다. --target-key-id 파라미터는 별칭과 연결할 KMS 키를 지정합니다. 별칭에 대한 현재 KMS 키를 지정할 필요는 없습니다.

```
aws kms update-alias \
    --alias-name alias/test-key \
    --target-key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

이 명령은 출력을 생성하지 않습니다. 별칭을 찾으려면 list-aliases 명령을 사용합니다.

자세한 내용은 AWS Key Management Service 개발자 안내서의 별칭 업데이트를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 UpdateAlias를 참조하세요.

Python

SDK for Python(Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class AliasManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_key = None

    @classmethod
    def from_client(cls) -> "AliasManager":
        """
        Creates an AliasManager instance with a default KMS client.

        :return: An instance of AliasManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def update_alias(self, alias, current_key_id):
        """
        Updates an alias by assigning it to another key.

        :param alias: The alias to reassign.
        :param current_key_id: The ARN or ID of the key currently associated with
 the alias.
        """
        new_key_id = input(
            f"Alias {alias} is currently associated with {current_key_id}. "
            f"Enter another key ID or ARN that you want to associate with
 {alias}: "
        )
        if new_key_id != "":
            try:
```

```
                self.kms_client.update_alias(AliasName=alias,
 TargetKeyId=new_key_id)
            except ClientError as err:
                logger.error(
                    "Couldn't associate alias %s with key %s. Here's why: %s",
                    alias,
                    new_key_id,
                    err.response["Error"]["Message"],
                )
            else:
                print(f"Alias {alias} is now associated with key {new_key_id}.")
        else:
            print("Skipping alias update.")
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 UpdateAlias를 참조하세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        " iv_alias_name = 'alias/my-key-alias'
        " iv_target_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/5678dcba-56cd-78ef-90ab-5678901234cd'
        lo_kms->updatealias(
          iv_aliasname = iv_alias_name
          iv_targetkeyid = iv_target_key_id
        ).
        MESSAGE 'Alias updated successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Alias or key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 UpdateAlias를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 `Verify` 함께 사용

다음 코드 예시는 `Verify`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

디지털 서명을 확인하려면

다음 `verify` 명령은 Base64로 인코딩된 짧은 메시지에 대한 암호화 서명을 확인합니다. 키 ID, 메시지, 메시지 유형 및 서명 알고리즘은 메시지 서명에 사용된 것과 동일해야 합니다.

AWS CLI v2에서 `message` 파라미터 값은 Base64-encoded되어야 합니다. 또는 메시지를 파일에 저장하고 AWS CLI에 파일에서 이진 데이터를 읽도록 지시하는 `fileb://` 접두사를 사용할 수 있습니다.

지정하는 서명은 base64로 인코딩할 수 없습니다. `sign` 명령이 반환하는 서명을 디코딩하는 데 도움이 필요하면 `sign` 명령 예시를 참조하세요.

명령의 출력에는 서명이 확인되었음을 나타내는 부울 `SignatureValid` 필드가 포함됩니다. 서명 검증에 실패하면 `verify` 명령도 실패합니다.

이 명령을 실행하기 전에 예제 키 ID를 AWS 계정의 유효한 키 ID로 바꿉니다.

```
aws kms verify \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --message fileb://EncodedMessage \
    --message-type RAW \
    --signing-algorithm RSASSA_PKCS1_V1_5_SHA_256 \
    --signature fileb://ExampleSignature
```

출력:

```
{
```

```
    "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "SignatureValid": true,
    "SigningAlgorithm": "RSASSA_PKCS1_V1_5_SHA_256"
}
```

AWS KMS에서 비대칭 KMS 키를 사용하는 방법에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 비대칭 키 사용을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 Verify를 참조하세요.

Python

SDK for Python(Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.

        :return: An instance of KeyEncrypt initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def verify(self, key_id: str, message: str, signature: str) -> bool:
        """
        Verifies a signature against a message.

        :param key_id: The ARN or ID of the key used to sign the message.
```

```
        :param message: The message to verify.
        :param signature: The signature to verify.
        :return: True when the signature matches the message, otherwise False.
        """
        try:
            response = self.kms_client.verify(
                KeyId=key_id,
                Message=message.encode(),
                Signature=signature,
                SigningAlgorithm="RSASSA_PSS_SHA_256",
            )
            valid = response["SignatureValid"]
            print(f"The signature is {'valid' if valid else 'invalid'}.")
            return valid
        except ClientError as err:
            if err.response["Error"]["Code"] == "SignatureDoesNotMatchException":
                print("The signature is not valid.")
            else:
                logger.error(
                    "Couldn't verify your signature. Here's why: %s",
                    err.response["Error"]["Message"],
                )
                raise
```

- API 세부 정보는 Python용AWS SDK(Boto3) 참조의 Verify를 참조하세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab' (asymmetric key)
        " iv_message contains the original message
```

```
        " iv_signature contains the signature to verify
        " iv_signing_algorithm = 'RSASSA_PSS_SHA_256'
        oo_result = lo_kms->verify(
          iv_keyid = iv_key_id
          iv_message = iv_message
          iv_signature = iv_signature
          iv_signingalgorithm = iv_signing_algorithm
        ).
        DATA(lv_valid) = oo_result->get_signaturevalid( ).
        IF lv_valid = abap_true.
          MESSAGE 'Signature is valid.' TYPE 'I'.
        ELSE.
          MESSAGE 'Signature is invalid.' TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_kmsdisabledexception.
        MESSAGE 'The key is disabled.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinvalidsigex.
        MESSAGE 'Invalid signature.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 확인을 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDKs AWS KMS 사용 시나리오

다음 코드 예제에서는 AWS SDKs를 사용하여에서 AWS KMS 일반적인 시나리오를 구현하는 방법을 보여줍니다. 이러한 시나리오에서는 내에서 AWS KMS 또는 다른 함수와 결합된 여러 함수를 호출하여 특정 작업을 수행하는 방법을 보여줍니다 AWS 서비스. 각 시나리오에는 전체 소스 코드에 대한 링크가 포함되어 있습니다. 여기에서 코드를 설정 및 실행하는 방법에 대한 지침을 찾을 수 있습니다.

시나리오는 컨텍스트에 맞는 서비스 작업을 이해하는 데 도움이 되도록 중급 수준의 경험을 대상으로 합니다.

예시

- AWS Command Line Interface v2를 사용한 DynamoDB 테이블 암호화 작업

AWS Command Line Interface v2를 사용한 DynamoDB 테이블 암호화 작업

다음 코드 예제에서는 DynamoDB 테이블의 암호화 옵션을 관리하는 방법을 보여줍니다.

- 기본 암호화를 사용하여 테이블을 생성합니다.
- 고객 관리형 CMK를 사용하여 테이블을 생성합니다.
- 테이블 암호화 설정을 업데이트합니다.
- 테이블 암호화를 설명합니다.

Bash

AWS CLI Bash 스크립트 사용

기본 암호화를 사용하여 테이블을 생성합니다.

```
# Create a table with default encryption (AWS owned key)
aws dynamodb create-table \
    --table-name CustomerData \
    --attribute-definitions \
        AttributeName=CustomerID,AttributeType=S \
    --key-schema \
        AttributeName=CustomerID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --sse-specification Enabled=true,SSEType=KMS
```

고객 관리형 CMK를 사용하여 테이블을 생성합니다.

```
# Step 1: Create a customer managed key in KMS
aws kms create-key \
    --description "Key for DynamoDB table encryption" \
    --key-usage ENCRYPT_DECRYPT \
    --customer-master-key-spec SYMMETRIC_DEFAULT

# Store the key ID for later use
KEY_ID=$(aws kms list-keys --query "Keys[?contains(KeyArn, 'Key for
 DynamoDB')].KeyId" --output text)
```

```
# Step 2: Create a table with the customer managed key
aws dynamodb create-table \
    --table-name SensitiveData \
    --attribute-definitions \
        AttributeName=RecordID,AttributeType=S \
    --key-schema \
        AttributeName=RecordID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --sse-specification Enabled=true,SSEType=KMS,KMSMasterKeyId=$KEY_ID
```

테이블 암호화를 업데이트합니다.

```
# Update a table to use a different KMS key
aws dynamodb update-table \
    --table-name CustomerData \
    --sse-specification Enabled=true,SSEType=KMS,KMSMasterKeyId=$KEY_ID
```

테이블 암호화를 설명합니다.

```
# Describe the table to see encryption settings
aws dynamodb describe-table \
    --table-name CustomerData \
    --query "Table.SSEDescription"
```

- API 세부 정보는 AWS CLI 명령 참조의 다음 토픽을 참조하세요.
  - CreateKey
  - CreateTable
  - DescribeTable
  - UpdateTable

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

에서 암호화 증명 지원 AWS KMS

AWS KMS 는 AWS Nitro Enclaves 및 AWS NitroTPM에 대한 암호화 증명을 지원합니다. 이러한 증명 방법을 지원하는 애플리케이션은 서명된 증명 문서를 사용하여 다음과 같은 AWS KMS 암호화 작업을 호출합니다.는 증명 문서가 유효한 소스(Nitro 엔클레이브 또는 NitroTPM)에서 왔는지 AWS KMS 확인합니다. 그런 다음 이러한 API는 응답에서 평문 데이터를 반환하는 대신 증명 문서의 퍼블릭 키로 평문을 암호화하고 엔클레이브 또는 EC2 인스턴스의 해당 프라이빗 키로만 복호화할 수 있는 사이퍼텍스트를 반환합니다.

- Decrypt
- DeriveSharedSecret
- GenerateDataKey
- GenerateDataKeyPair
- GenerateRandom

다음 테이블에서는 증명된 요청에 대한 응답이 각 API 작업에 대한 표준 응답과 어떻게 다른지 보여줍니다.

AWS KMS 작업	표준 응답	증명된 요청에 대한 응답
`Decrypt`	일반 텍스트 데이터를 반환합니다.	증명 문서에서 퍼블릭 키로 암호화된 일반 텍스트 데이터를 반환합니다.
`DeriveSharedSecret`	원시 공유 비밀 반환	증명 문서에서 퍼블릭 키로 암호화된 원시 공유을 반환합니다.
`GenerateDataKey`	데이터 키의 일반 텍스트 복사본을 반환합니다. (KMS 키로 암호화된 데이터 키의 복사본도 반환합니다.)	증명 문서에서 퍼블릭 키로 암호화된 데이터 키의 복사본을 반환합니다. (KMS 키로 암호화된 데이터 키의 복사본도 반환합니다.)

AWS KMS 작업	표준 응답	증명된 요청에 대한 응답
`GenerateDataKeyPair`	프라이빗 키의 일반 텍스트 복사본을 반환합니다. (퍼블릭 키와 KMS 키로 암호화된 프라이빗 키의 복사본도 반환합니다.)	증명 문서에서 퍼블릭 키로 암호화된 프라이빗 키의 복사본을 반환합니다. (퍼블릭 키와 KMS 키로 암호화된 프라이빗 키의 복사본도 반환합니다.)
`GenerateRandom`	무작위 바이트 문자열을 반환합니다.	증명 문서에서 퍼블릭 키로 암호화된 무작위 바이트 문자열을 반환합니다.

AWS KMS 는 증명 문서의 내용을 기반으로 키로 증명된 작업을 허용하거나 거부하는 데 사용할 수 있는 정책 조건 AWS KMS 키를 지원합니다. AWS CloudTrail 로그에서 에 대해 증명된 요청을 모니터링할 AWS KMS 수도 있습니다.

자세히 알아보기

- 암호화 증명
- AWS KMS 증명된 플랫폼의 조건 키
- 에 대해 증명된 호출을 수행하는 방법 AWS KMS
- 증명된 요청 모니터링

에 대해 증명된 호출을 수행하는 방법 AWS KMS

에 대해 증명된 호출을 수행하려면 요청의 `Recipient` 파라미터를 AWS KMS사용하여 서명된 증명 문서와 증명 문서의 퍼블릭 키와 함께 사용할 암호화 알고리즘을 제공합니다. 요청에 서명된 증명 문서가 있는 `Recipient` 파라미터가 포함된 경우, 응답에는 퍼블릭 키로 암호화된 사이퍼텍스트가 포함된 `CiphertextForRecipient` 필드가 포함됩니다. 일반 텍스트 필드가 null이거나 비어있습니다.

`Recipient` 파라미터는 AWS Nitro Enclaves 또는 AWS NitroTPM.의 서명된 증명 문서를 지정해야 합니다. AWS KMS 는 증명 문서의 디지털 서명을 사용하여 요청의 퍼블릭 키가 유효한 소스에서 왔음을 증명합니다. 증명 문서에 디지털 서명하기 위한 자체 인증서를 제공할 수 없습니다.

AWS Nitro 엔클레이브 내에서만 지원되는 Nitro Enclaves SDK는 모든 AWS KMS 요청에 `Recipient` 파라미터와 해당 값을 자동으로 추가합니다.

AWS SDKs에서 증명된 요청을 수행하려면 `Recipient` 파라미터와 해당 값을 지정해야 합니다. 증명 문서는 nitro-tpm-attest 유틸리티를 사용하여 NitroTPM에서 검색하거나 NSM API를 사용하여 Nitro 보안 모듈(NSM)에서 검색할 수 있습니다.

AWS KMS 는 증명 문서의 내용을 기반으로 키로 증명된 작업을 허용하거나 거부하는 데 사용할 수 있는 정책 조건 AWS KMS 키를 지원합니다. AWS CloudTrail 로그에서 에 대해 증명된 요청을 모니터링할 AWS KMS 수도 있습니다.

`Recipient` 파라미터 및 AWS `CiphertextForRecipient` 응답 필드에 대한 자세한 내용은 AWS Key Management Service API 참조의 Decrypt, DeriveSharedSecret, GenerateDataKey, GenerateDataKeyPair 및 GenerateRandom 주제, AWS Nitro Enclaves SDK 또는 모든 AWS SDK를 참조하세요. 암호화를 위한 데이터 및 데이터 키 설정에 대한 자세한 내용은 에서 암호화 증명 사용을 AWS KMS 참조하세요.

증명된 요청 모니터링

AWS CloudTrail 로그를 사용하여 증명을 사용하는 Decrypt, DeriveSharedSecret, GenerateDataKey, GenerateDataKeyPair 및 GenerateRandom 작업을 모니터링할 수 있습니다. 이러한 로그 항목에서 `additionalEventData` 필드에는 요청의 증명 문서의 정보가 포함된 `recipient` 필드가 있습니다. 이러한 필드는 요청의 `Recipient` 파라미터가 서명된 증명 문서를 지정하는 경우에만 포함됩니다.

CloudTrail 로그에 포함되는 구체적인 정보는 사용된 증명 방식에 따라 달라집니다.

Nitro 엔클레이브에 대한 요청 모니터링

Nitro 엔클레이브 증명의 경우 CloudTrail 로그에는 증명 문서의 모듈 ID(`attestationDocumentModuleId`), 이미지 다이제스트 (`attestationDocumentEnclaveImageDigest`), 플랫폼 구성 레지스터(PCR)가 포함됩니다.

모듈 ID는 Nitro 엔클레이브의 엔클레이브 ID입니다. 이미지 다이제스트는 엔클레이브 이미지의 SHA384 해시입니다. 키 정책 및 IAM 정책의 조건에서 이미지 다이제스트 및 PCR 값을 사용할 수 있습니다. PCR에 대한 자세한 내용은 AWS Nitro Enclaves 사용 설명서의 엔클레이브 측정값을 얻을 수 있는 위치를 참조하세요.

이 섹션에서는 AWS KMS에 대한 지원되는 각 Nitro 엔클레이브 요청에 대한 CloudTrail 로그 항목의 예시를 보여줍니다.

Decrypt(엔클레이브용)

다음 예제는 AWS Nitro 엔클레이브에 대한 Decrypt 작업의 AWS CloudTrail 로그 항목을 보여줍니다.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2020-07-27T22:58:24Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": null,
    "additionalEventData": {
        "recipient": {
            "attestationDocumentModuleId": "i-123456789abcde123-enc123456789abcde12",
            "attestationDocumentEnclaveImageDigest": "<AttestationDocument.PCR0>",
            "attestationDocumentEnclavePCR1": "<AttestationDocument.PCR1>",
            "attestationDocumentEnclavePCR2": "<AttestationDocument.PCR2>",
            "attestationDocumentEnclavePCR3": "<AttestationDocument.PCR3>",
            "attestationDocumentEnclavePCR4": "<AttestationDocument.PCR4>",
            "attestationDocumentEnclavePCR8": "<AttestationDocument.PCR8>"
        }
    },
    "requestID": "b4a65126-30d5-4b28-98b9-9153da559963",
    "eventID": "e5a2f202-ba1a-467c-b4ba-f729d45ae521",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
```

```
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

GenerateDataKey(엔클레이브용)

다음 예제는 AWS Nitro 엔클레이브에 대한 GenerateDataKey 작업의 AWS CloudTrail 로그 항목을 보여줍니다.

```
{
    "eventVersion": "1.02",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2014-11-04T00:52:40Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "numberOfBytes": 32
    },
    "responseElements": null,
    "additionalEventData": {
        "recipient": {
            "attestationDocumentModuleId": "i-123456789abcde123-enc123456789abcde12",
            "attestationDocumentEnclaveImageDigest": "<AttestationDocument.PCR0>",
            "attestationDocumentEnclavePCR1": "<AttestationDocument.PCR1>",
            "attestationDocumentEnclavePCR2": "<AttestationDocument.PCR2>",
            "attestationDocumentEnclavePCR3": "<AttestationDocument.PCR3>",
            "attestationDocumentEnclavePCR4": "<AttestationDocument.PCR4>",
            "attestationDocumentEnclavePCR8": "<AttestationDocument.PCR8>"
```

```
        }
    },
    "requestID": "e0eb83e3-63bc-11e4-bc2b-4198b6150d5c",
    "eventID": "a9dea4f9-8395-46c0-942c-f509c02c2b71",
    "readOnly": true,
    "resources": [{
        "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "accountId": "111122223333"
    }],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

GenerateDataKeyPair(엔클레이브용)

다음 예제는 AWS Nitro 엔클레이브에 대한 GenerateDataKeyPair 작업의 AWS CloudTrail 로그 항목을 보여줍니다.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::111122223333:user/Alice",
            "accountId": "111122223333",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
    },
    "eventTime": "2020-07-27T18:57:57Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKeyPair",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyPairSpec": "RSA_3072",
        "encryptionContext": {
            "Project": "Alpha"
        },
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": null,
```

```
    "additionalEventData": {
        "recipient": {
            "attestationDocumentModuleId": "i-123456789abcde123-enc123456789abcde12",
            "attestationDocumentEnclaveImageDigest": "<AttestationDocument.PCR0>",
            "attestationDocumentEnclavePCR1": "<AttestationDocument.PCR1>",
            "attestationDocumentEnclavePCR2": "<AttestationDocument.PCR2>",
            "attestationDocumentEnclavePCR3": "<AttestationDocument.PCR3>",
            "attestationDocumentEnclavePCR4": "<AttestationDocument.PCR4>",
            "attestationDocumentEnclavePCR8": "<AttestationDocument.PCR8>"
        }
    },
    "requestID": "52fb127b-0fe5-42bb-8e5e-f560febde6b0",
    "eventID": "9b6bd6d2-529d-4890-a949-593b13800ad7",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

GenerateRandom(엔클레이브용)

다음 예제는 AWS Nitro 엔클레이브에 대한 GenerateRandom 작업의 AWS CloudTrail 로그 항목을 보여줍니다.

```
{
    "eventVersion": "1.02",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2014-11-04T00:52:37Z",
    "eventSource": "kms.amazonaws.com",
```

```
    "eventName": "GenerateRandom",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData": {
        "recipient": {
            "attestationDocumentModuleId": "i-123456789abcde123-enc123456789abcde12",
            "attestationDocumentEnclaveImageDigest": "<AttestationDocument.PCR0>",
            "attestationDocumentEnclavePCR1": "<AttestationDocument.PCR1>",
            "attestationDocumentEnclavePCR2": "<AttestationDocument.PCR2>",
            "attestationDocumentEnclavePCR3": "<AttestationDocument.PCR3>",
            "attestationDocumentEnclavePCR4": "<AttestationDocument.PCR4>",
            "attestationDocumentEnclavePCR8": "<AttestationDocument.PCR8>"
        }
    },
    "requestID": "df1e3de6-63bc-11e4-bc2b-4198b6150d5c",
    "eventID": "239cb9f7-ae05-4c94-9221-6ea30eef0442",
    "readOnly": true,
    "resources": [],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

NitroTPM에 대한 요청 모니터링

NitroTPM 증명의 경우 CloudTrail 로그에는 증명 문서의 모듈 ID(`attestationDocumentModuleId`)와 플랫폼 구성 레지스터(PCR)가 포함됩니다.

모듈 ID는 TPM 식별자가 있는 NitroTPM EC2 인스턴스의 ID입니다. 키 정책 및 IAM 정책의 조건에서 PCR 값을 사용할 수 있습니다.

이 섹션에서는 지원되는 각 NitroTPM 요청에 대한 CloudTrail 로그 항목의 예를 보여줍니다 AWS KMS.

Decrypt(NitroTPM용)

다음 예제는 NitroTPM에 대한 Decrypt 작업의 AWS CloudTrail 로그 항목을 보여줍니다.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
```

```
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2020-07-27T22:58:24Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": null,
    "additionalEventData": {
        "recipient": {
           "attestationDocumentModuleId": "i-123456789abcde123-tpm0000000000000000",
           "attestationDocumentNitroTPMPCR4": "<AttestationDocument.PCR4>",
           "attestationDocumentNitroTPMPCR7": "<AttestationDocument.PCR7>",
           "attestationDocumentNitroTPMPCR8": "<AttestationDocument.PCR8>",
           "attestationDocumentNitroTPMPCR9": "<AttestationDocument.PCR9>",
           "attestationDocumentNitroTPMPCR16": "<AttestationDocument.PCR16>",
           "attestationDocumentNitroTPMPCR23": "<AttestationDocument.PCR23>"
        }
    },
    "requestID": "b4a65126-30d5-4b28-98b9-9153da559963",
    "eventID": "e5a2f202-ba1a-467c-b4ba-f729d45ae521",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
```

```
}
```

GenerateDataKey(NitroTPM용)

다음 예제는 NitroTPM에 대한 GenerateDataKey 작업의 AWS CloudTrail 로그 항목을 보여줍니다.

```
{
    "eventVersion": "1.02",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2014-11-04T00:52:40Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "numberOfBytes": 32
    },
    "responseElements": null,
    "additionalEventData": {
        "recipient": {
           "attestationDocumentModuleId": "i-123456789abcde123-tpm0000000000000000",
           "attestationDocumentNitroTPMPCR4": "<AttestationDocument.PCR4>",
           "attestationDocumentNitroTPMPCR7": "<AttestationDocument.PCR7>",
           "attestationDocumentNitroTPMPCR8": "<AttestationDocument.PCR8>",
           "attestationDocumentNitroTPMPCR9": "<AttestationDocument.PCR9>",
           "attestationDocumentNitroTPMPCR16": "<AttestationDocument.PCR16>",
           "attestationDocumentNitroTPMPCR23": "<AttestationDocument.PCR23>"
        }
    },
    "requestID": "e0eb83e3-63bc-11e4-bc2b-4198b6150d5c",
    "eventID": "a9dea4f9-8395-46c0-942c-f509c02c2b71",
    "readOnly": true,
    "resources": [{
```

```
        "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "accountId": "111122223333"
    }],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

GenerateDataKeyPair(NitroTPM용)

다음 예제는 NitroTPM에 대한 GenerateDataKeyPair 작업의 AWS CloudTrail 로그 항목을 보여줍니다.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::111122223333:user/Alice",
            "accountId": "111122223333",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
    },
    "eventTime": "2020-07-27T18:57:57Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKeyPair",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyPairSpec": "RSA_3072",
        "encryptionContext": {
            "Project": "Alpha"
        },
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": null,
    "additionalEventData": {
        "recipient": {
           "attestationDocumentModuleId": "i-123456789abcde123-tpm0000000000000000",
           "attestationDocumentNitroTPMPCR4": "<AttestationDocument.PCR4>",
           "attestationDocumentNitroTPMPCR7": "<AttestationDocument.PCR7>",
           "attestationDocumentNitroTPMPCR8": "<AttestationDocument.PCR8>",
           "attestationDocumentNitroTPMPCR9": "<AttestationDocument.PCR9>",
```

```
            "attestationDocumentNitroTPMPCR16": "<AttestationDocument.PCR16>",
            "attestationDocumentNitroTPMPCR23": "<AttestationDocument.PCR23>"
        }
    },
    "requestID": "52fb127b-0fe5-42bb-8e5e-f560febde6b0",
    "eventID": "9b6bd6d2-529d-4890-a949-593b13800ad7",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

GenerateRandom(NitroTPM용)

다음 예제는 NitroTPM에 대한 GenerateRandom 작업의 AWS CloudTrail 로그 항목을 보여줍니다.

```
{
    "eventVersion": "1.02",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2014-11-04T00:52:37Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateRandom",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData": {
        "recipient": {
```

```
            "attestationDocumentModuleId": "i-123456789abcde123-tpm0000000000000000",
            "attestationDocumentNitroTPMPCR4": "<AttestationDocument.PCR4>",
            "attestationDocumentNitroTPMPCR7": "<AttestationDocument.PCR7>",
            "attestationDocumentNitroTPMPCR8": "<AttestationDocument.PCR8>",
            "attestationDocumentNitroTPMPCR9": "<AttestationDocument.PCR9>",
            "attestationDocumentNitroTPMPCR16": "<AttestationDocument.PCR16>",
            "attestationDocumentNitroTPMPCR23": "<AttestationDocument.PCR23>"
        }
    },
    "requestID": "df1e3de6-63bc-11e4-bc2b-4198b6150d5c",
    "eventID": "239cb9f7-ae05-4c94-9221-6ea30eef0442",
    "readOnly": true,
    "resources": [],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

AWS 서비스와 함께 AWS KMS 암호화 사용

AWS Key Management Service를 사용하면 다른 AWS 서비스의 데이터를 보호하기 위한 암호화 키를 제공할 수 있습니다. AWS 서비스와 함께 AWS KMS 암호화를 사용하는 것은 AWS KMS 다른 AWS 서비스와 통합하여 저장 또는 전송 중인 데이터를 암호화하고 복호화하는 프로세스를 말합니다. 개발자, 시스템 관리자 및 보안 전문가는 AWS 서비스를 통해 저장되거나 전송되는 민감한 데이터를 보호하거나, 규정 준수 요구 사항을 충족하거나, 암호화 모범 사례를 구현하기 위해이 주제에 관심이 있을 수 있습니다. 일반적인 사용 사례에는 Amazon EBS 볼륨, Amazon S3 버킷 및 Amazon RDS 데이터베이스 암호화가 포함됩니다. 다음 섹션에서는 특정 AWS 서비스에 대한 AWS KMS 암호화 키를 구성하고 관리하여 AWS 환경 전반의 데이터 기밀성과 무결성을 보장하는 단계를 다룹니다.와 통합된 AWS 서비스의 전체 목록은 AWS 서비스 통합을 AWS KMS참조하세요.

다음 주제에서는 지원하는 KMS 키 AWS KMS, 데이터 키 관리 방법, 필요한 권한, 계정에서 각 서비스의 KMS 키 사용을 추적하는 방법 등 특정 서비스가를 사용하는 방법을 자세히 설명합니다.

> **Important**
>
> AWS 와 통합된 서비스는 AWS KMS 대칭 암호화 KMS 키만 사용하여 데이터를 암호화합니다. 이러한 서비스는 비대칭 KMS 키를 사용한 암호화를 지원하지 않습니다. KMS 키가 대칭 또는 비대칭인지 여부를 확인하는 방법은 다양한 키 유형 식별 섹션을 참조하세요.

주제

- Amazon Elastic Block Store(Amazon EBS)의 사용 방식 AWS KMS
- Amazon EMR의 사용 방식 AWS KMS
- Amazon Redshift의 사용 방식 AWS KMS

Amazon Elastic Block Store(Amazon EBS)의 사용 방식 AWS KMS

이 주제에서는 Amazon Elastic Block Store(Amazon EBS)가 AWS KMS 를 사용하여 볼륨 및 스냅샷을 암호화하는 방법을 자세히 설명합니다. Amazon EBS 볼륨을 암호화하는 방법에 대한 기본 지침은 Amazon EBS 암호화를 참조하세요.

주제

- Amazon EBS 암호화

- KMS 키 및 데이터 키 사용
- Amazon EBS 암호화 컨텍스트
- Amazon EBS 오류 감지
- AWS CloudFormation 를 사용하여 암호화된 Amazon EBS 볼륨 생성

Amazon EBS 암호화

암호화된 Amazon EBS 볼륨을 지원되는 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스 유형에 연결하면 볼륨, I/O, 그리고 볼륨에서 생성된 스냅샷에 저장된 데이터가 모두 암호화됩니다. 암호화는 Amazon EC2 인스턴스를 호스팅하는 서버에서 이루어집니다.

이 기능은 모든 Amazon EBS 볼륨 유형에 지원됩니다. 암호화된 볼륨에 액세스하는 방법은 다른 볼륨의 경우와 동일합니다. 암호화 및 암호 해독은 중단 없이 처리되므로 사용자, EC2 인스턴스 또는 애플리케이션에서 별도로 조치할 부분은 없습니다. 암호화된 볼륨의 스냅샷은 자동으로 암호화되며, 암호화된 스냅샷으로 생성한 볼륨도 자동으로 암호화됩니다.

EBS 볼륨의 암호화 상태는 볼륨을 생성할 때 결정됩니다. 기존 볼륨의 암호화 상태는 변경할 수 없습니다. 그러나 암호화된 볼륨과 암호화되지 않은 볼륨 사이에서 데이터를 마이그레이션하고 스냅샷을 복사하는 동안 새 암호화 상태를 적용할 수 있습니다.

Amazon EBS는 기본적으로 선택적 암호화를 지원합니다. AWS 계정 및 리전의 모든 새 EBS 볼륨 및 스냅샷 복사본에서 암호화를 자동으로 활성화할 수 있습니다. 이 구성 설정은 기존 볼륨이나 스냅샷에는 영향을 주지 않습니다. 자세한 내용은 Amazon EBS 사용 설명서의 Amazon EBS encryption을 참조하세요.

KMS 키 및 데이터 키 사용

암호화된 Amazon EBS 볼륨을 생성할 때 AWS KMS key를 지정합니다. 기본적으로 Amazon EBS는 계정(aws/ebs)의 Amazon EBS에 AWS 관리형 키를 사용합니다. 그러나 사용자가 자신이 생성 및 관리하는 고객 관리형 키를 지정할 수 있습니다.

고객 관리형 KMS 키를 사용하려면 사용자를 대신하여 KMS 키를 사용할 수 있는 Amazon EBS 권한을 부여해야 합니다. 자세한 내용은 Amazon EBS 사용 설명서의 Amazon EBS 암호화 작동 방식을 참조하세요.

> **Important**
>
> Amazon EBS는 대칭 KMS 키만 지원합니다. 비대칭 KMS 키를 사용하여 Amazon EBS 볼륨을 암호화할 수 없습니다. KMS 키가 대칭 또는 비대칭인지 여부를 확인하는 방법은 다양한 키 유형 식별 단원을 참조하세요.

각 볼륨에 대해 Amazon EBS는 사용자가 지정한 KMS 키로 암호화된 고유한 데이터 키를 생성 AWS KMS 하도록 요청합니다. Amazon EBS는 볼륨과 함께 암호화된 데이터 키를 저장합니다. 그런 다음 볼륨을 Amazon EC2 인스턴스에 연결하면 Amazon EBS가를 호출 AWS KMS 하여 데이터 키를 해독합니다. Amazon EBS는 하이퍼바이저 메모리에서 일반 텍스트 데이터 키를 사용하여 모든 디스크 I/O를 볼륨으로 암호화합니다. 자세한 내용은 Amazon EC2 사용 설명서 또는 Amazon EC2 사용 설명서의 EBS의 암호화 방식을 참조하세요.

Amazon EBS 암호화 컨텍스트

GenerateDataKeyWithoutPlaintext 및 Decrypt 요청에서 AWS KMS Amazon EBS는 요청의 볼륨 또는 스냅샷을 식별하는 이름-값 페어와 함께 암호화 컨텍스트를 사용합니다. 암호화 컨텍스트의 이름은 달라지지 않습니다.

암호화 컨텍스트는 보안되지 않은 임의의 데이터를 포함하는 키-값 페어 세트입니다. 데이터 암호화 요청에 암호화 컨텍스트를 포함하면 암호화 컨텍스트가 암호화된 데이터에 AWS KMS 암호화 방식으로 바인딩됩니다. 따라서 동일한 암호화 컨텍스트로 전달해야 이 데이터를 해독할 수 있습니다.

모든 볼륨 및 Amazon EBS CreateSnapshot 작업으로 생성된 암호화된 스냅샷에 대해 Amazon EBS는 볼륨 ID를 암호화 컨텍스트 값으로 사용합니다. CloudTrail 로그 항목의 `requestParameters` 필드에서 암호화 컨텍스트는 다음과 비슷하게 표시됩니다.

```
"encryptionContext": {
  "aws:ebs:id": "vol-0cfb133e847d28be9"
}
```

Amazon EC2 CopySnapshot 작업을 통해 생성된 암호화된 스냅샷에서 Amazon EBS는 스냅샷 ID를 암호화 컨텍스트 값으로 사용합니다. CloudTrail 로그 항목의 `requestParameters` 필드에서 암호화 컨텍스트는 다음과 비슷하게 표시됩니다.

```
"encryptionContext": {
  "aws:ebs:id": "snap-069a655b568de654f"
```

```
}
```

Amazon EBS 오류 감지

암호화된 EBS 볼륨을 생성하거나 볼륨을 EC2 인스턴스에 연결하려면, Amazon EBS와 Amazon EC2 인프라는 사용자가 EBS 볼륨 암호화를 위해 지정한 KMS 키를 사용할 수 있어야 합니다. KMS 키를 사용할 수 없는 경우(예를 들어 키 상태가 `Enabled`가 아닌 경우) 볼륨 생성 또는 볼륨 연결에 실패합니다.

이 경우 Amazon EBS는 이벤트를 Amazon EventBridge(이전의 CloudWatch Events)로 보내 실패를 알립니다. EventBridge에서는 이러한 이벤트에 대한 응답으로 자동 작업을 트리거하는 규칙을 수립할 수 있습니다. 자세한 내용은 Amazon EBS 사용 설명서의 Amazon EBS용 Amazon CloudWatch Events를 참조하세요. 특히 다음 섹션을 확인하세요.

- 볼륨 연결 또는 다시 연결에 유효하지 않은 암호화 키
- 볼륨 생성에 유효하지 않은 암호화 키

이러한 장애를 해결하려면 EBS 볼륨 암호화를 위해 지정한 KMS 키가 활성화되어 있는지 확인합니다. 이렇게 하려면 먼저 KMS 키를 보고 현재 키 상태(의 상태 열)를 확인합니다 AWS Management Console. 그리고 다음 링크 중 하나에서 정보를 참조합니다.

- KMS 키의 키 상태가 비활성화되어 있으면 활성화합니다.
- KMS 키의 키 상태가 가져오기 보류 중인 경우, 키 구성 요소를 가져옵니다.
- KMS 키의 키 상태가 삭제 보류 중인 경우, 키 구성 요소를 삭제합니다.

AWS CloudFormation 를 사용하여 암호화된 Amazon EBS 볼륨 생성

AWS CloudFormation을 이용해 암호화된 Amazon EBS 볼륨을 생성할 수 있습니다. 자세한 내용은 AWS CloudFormation 사용 설명서의 AWS::EC2::Volume을 참조하세요.

Amazon EMR의 사용 방식 AWS KMS

Amazon EMR 클러스터를 사용하는 경우에는 영구 스토리지 위치에 저장하기 전에 저장된 데이터를 암호화하도록 클러스터를 구성할 수 있습니다. EMR 파일 시스템(EMRFS), 클러스터 노드의 스토리지 볼륨 또는 두 곳에 저장된 데이터를 모두 암호화할 수 있습니다. 저장된 데이터를 암호화하기 위해 AWS KMS key를 사용할 수 있습니다. 다음 주제에서는 Amazon EMR 클러스터가 어떻게 KMS 키를 이용해 저장된 데이터를 암호화하는지 설명합니다.

> **Important**
>
> Amazon EMR은 대칭 KMS 키만 지원합니다. 비대칭 KMS 키를 사용하여 Amazon EMR 클러스터의 저장된 데이터를 암호화할 수 없습니다. KMS 키가 대칭 또는 비대칭인지 여부를 확인하는 방법에 대한 도움말은 다양한 키 유형 식별 단원을 참조하십시오.

Amazon EMR 클러스터는 전송 중인 데이터도 암호화할 수 있습니다. 그러면 클러스터가 네트워크를 통해 데이터를 보내기 전에 암호화합니다. KMS 키를 이용해 전송 중인 데이터를 암호화할 수 없습니다. 자세한 내용은 Amazon EMR 관리 안내서의 전송 중 데이터 암호화를 참조하세요.

Amazon EMR에서 사용할 수 있는 모든 암호화 옵션에 대한 자세한 내용은 Amazon EMR 관리 가이드의 암호화 옵션을 참조하십시오.

주제

- EMR 파일 시스템(EMRFS)에서 데이터 암호화
- 클러스터 노드의 스토리지 볼륨에서 데이터 암호화
- 암호화 컨텍스트

EMR 파일 시스템(EMRFS)에서 데이터 암호화

Amazon EMR 클러스터는 배포된 두 파일 시스템을 사용합니다.

- Hadoop 분산 파일 시스템(HDFS) HDFS 암호화는 AWS KMS에서 KMS 키를 사용하지 않습니다.
- EMR 파일 시스템(EMRFS). EMRFS는 Amazon EMR 클러스터가 Amazon Simple Storage Service(Amazon S3)에 데이터를 저장할 수 있도록 하는 HDFS 구현입니다. EMRFS는 네 가지 암호화 옵션을 지원하는데, 그 중 두 개가 AWS KMS의 KMS 키를 사용합니다. EMRFS에서 사용할 수 있는 4가지 암호화 옵션에 대한 자세한 내용은 Amazon EMR 관리 가이드의 암호화 옵션을 참조하십시오.

KMS 키를 사용하는 두 가지 EMRFS 암호화 옵션은 Amazon S3이 제공하는 다음 암호화 기능을 사용합니다.

- AWS Key Management Service (SSE-KMS)를 사용한 서버 측 암호화를 사용하여 데이터를 보호합니다. Amazon EMR 클러스터가 Amazon S3로 데이터를 보냅니다. Amazon S3가 S3 버킷에 데이터를 저장하기 전에 KMS 키를 사용해 암호화합니다. 이 작업이 이루어지는 과정에 대한 자세한 내용은 SSE-KMS를 통해 EMRFS에 데이터를 암호화하는 프로세스 단원을 참조하십시오.

- 클라이언트 측 암호화(CSE-KMS)를 사용하여 데이터 보호. Amazon EMR의 데이터는 저장을 위해 Amazon S3로 전송되기 전에 AWS KMS key 에서 암호화됩니다. 이 작업이 이루어지는 과정에 대한 자세한 내용은 CSE-KMS를 통해 EMRFS에 데이터를 암호화하는 프로세스 단원을 참조하십시오.

KMS 키를 사용하여 EMRFS에서 데이터를 암호화하도록 Amazon EMR 클러스터를 구성할 때 Amazon S3 또는 Amazon EMR 클러스터가 사용할 KMS 키를 선택합니다. SSE-KMS를 사용하면 별칭이 aws/s3인 Amazon S3에 대해 AWS 관리형 키 를 선택하거나 직접 생성한 대칭 고객 관리형 키를 선택할 수 있습니다. 클라이언트 측 암호화를 사용하는 경우 사용자가 생성한 대칭 고객 관리형 키를 선택해야 합니다. 고객 관리형 키를 선택하는 경우 Amazon EMR 클러스터에 KMS 키를 사용할 권한이 있는지 확인해야 합니다. 자세한 내용은 Amazon EMR 관리 안내서의 암호화 AWS KMS keys 에 사용을 참조하세요.

서버 측 암호화와 클라이언트 측 암호화 모두 공통적으로, 선택하는 KMS 키는 envelope 암호화 워크플로의 루트 키입니다. 데이터는의 KMS 키로 암호화된 고유한 데이터 키로 암호화됩니다 AWS KMS. 암호화된 데이터와 그 데이터 키의 암호화된 사본이 S3 버킷에서 암호화된 단일 객체로 저장됩니다. 이 과정에 대한 자세한 내용은 다음 주제를 참조하십시오.

주제

- SSE-KMS를 통해 EMRFS에 데이터를 암호화하는 프로세스
- CSE-KMS를 통해 EMRFS에 데이터를 암호화하는 프로세스

SSE-KMS를 통해 EMRFS에 데이터를 암호화하는 프로세스

Amazon EMR 클러스터가 SSE-KMS를 사용하도록 구성하면 암호화 프로세스가 다음과 같이 작동합니다.

1. 클러스터는 S3 버킷에 저장하기 위해 Amazon S3로 데이터를 보냅니다.
2. Amazon S3는 SSE-KMS를 사용하도록 클러스터를 구성할 때 선택한 KMS 키의 키 ID를 AWS KMS 지정하여 GenerateDataKey 요청을에 보냅니다. 요청에는 암호화 컨텍스트가 포함됩니다. 자세한 내용은 암호화 컨텍스트 단원을 참조하십시오.
3. AWS KMS 는 고유한 데이터 암호화 키(데이터 키)를 생성한 다음이 데이터 키의 복사본 2개를 Amazon S3로 보냅니다. 사본 하나는 암호화되지 않고(일반 텍스트), 다른 하나는 KMS 키 하에서 암호화됩니다.
4. Amazon S3는 일반 텍스트 데이터 키를 사용해 1단계에 받은 데이터를 암호화하고, 사용 후 가급적 빨리 메모리에서 일반 텍스트 데이터 키를 제거합니다.

5. Amazon S3는 암호화된 데이터와 그 데이터 키의 암호화된 사본을 S3 버킷에서 암호화된 단일 객체로 저장합니다.

복호화 프로세스는 다음과 같습니다.

1. 클러스터가 S3 버킷에서 암호화된 데이터 객체를 요청합니다.
2. Amazon S3는 S3 객체에서 암호화된 데이터 키를 추출한 다음 Decrypt 요청과 AWS KMS 함께 암호화된 데이터 키를에 전송합니다. 이 요청은 암호화 컨텍스트를 포함합니다.
3. AWS KMS 는 암호화에 사용된 것과 동일한 KMS 키를 사용하여 암호화된 데이터 키를 복호화한 다음 복호화된(일반 텍스트) 데이터 키를 Amazon S3로 전송합니다.
4. Amazon S3는 일반 텍스트 데이터 키를 사용해 암호화된 데이터를 복호화하고, 사용 후 가급적 빨리 메모리에서 일반 텍스트 데이터 키를 제거합니다.
5. Amazon S3는 복호화한 데이터를 클러스터로 보냅니다.

CSE-KMS를 통해 EMRFS에 데이터를 암호화하는 프로세스

Amazon EMR 클러스터가 CSE-KMS를 사용하도록 구성하면 암호화 프로세스가 다음과 같이 작동합니다.

1. Amazon S3에 데이터를 저장할 준비가 되면 클러스터는 CSE-KMS를 사용하도록 클러스터를 구성할 때 선택한 KMS 키의 키 ID를 AWS KMS지정하여 GenerateDataKey 요청을에 보냅니다. 요청에는 암호화 컨텍스트가 포함됩니다. 자세한 내용은 암호화 컨텍스트 단원을 참조하십시오.
2. AWS KMS 는 고유한 데이터 암호화 키(데이터 키)를 생성한 다음이 데이터 키의 복사본 2개를 클러스터로 보냅니다. 사본 하나는 암호화되지 않고(일반 텍스트), 다른 하나는 KMS 키 하에서 암호화됩니다.
3. 클러스터는 일반 텍스트 데이터 키를 사용해 데이터를 암호화하고, 사용 후 가급적 빨리 메모리에서 일반 텍스트 데이터 키를 제거합니다.
4. 클러스터는 암호화된 데이터와 그 데이터 키의 암호화된 사본을 암호화된 단일 객체로 결합합니다.
5. 클러스터는 저장을 위해 Amazon S3로 암호화된 객체를 보냅니다.

복호화 프로세스는 다음과 같습니다.

1. 클러스터가 S3 버킷에서 암호화된 데이터 객체를 요청합니다.
2. Amazon S3는 클러스터로 암호화된 객체를 보냅니다.

3. 클러스터는 암호화된 객체에서 암호화된 데이터 키를 추출한 다음 Decrypt 요청과 AWS KMS 함께 암호화된 데이터 키를에 전송합니다. 이 요청은 암호화 컨텍스트를 포함합니다.
4. AWS KMS 는 암호화에 사용된 것과 동일한 KMS 키를 사용하여 암호화된 데이터 키를 복호화한 다음 복호화된(일반 텍스트) 데이터 키를 클러스터로 전송합니다.
5. 클러스터는 일반 텍스트 데이터 키를 사용해 암호화된 데이터를 해독하고, 사용 후 가급적 빨리 메모리에서 일반 텍스트 데이터 키를 제거합니다.

클러스터 노드의 스토리지 볼륨에서 데이터 암호화

Amazon EMR 클러스터는 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스의 모음입니다. 클러스터에 있는 각 인스턴스를 클러스터 노드 또는 노드라고 합니다. 각 노드에는 인스턴스 스토어 볼륨과 Amazon Elastic Block Store(Amazon EBS) 볼륨 등 두 가지 유형의 스토리지 볼륨이 있을 수 있습니다. LUKS(Linux Unified Key Setup)를 사용해 노드에서 두 가지 유형의 스토리지 볼륨을 모두 암호화하도록(각 노드의 부팅 볼륨은 제외) 클러스터를 구성할 수 있습니다. 이를 로컬 디스크 암호화라고 합니다.

클러스터에 로컬 디스크 암호화를 활성화하면 AWS KMS에서 KMS 키로 LUKS 키를 암호화할 수 있습니다. 직접 생성한 고객 관리형 키를 선택해야 합니다. AWS 관리형 키는 사용할 수 없습니다. 고객 관리형 키를 선택하는 경우 Amazon EMR 클러스터에 KMS 키를 사용할 권한이 있는지 확인해야 합니다. 자세한 내용은 Amazon EMR 관리 안내서의 암호화 AWS KMS keys 에 사용을 참조하세요.

KMS 키를 사용해 로컬 디스크 암호화를 활성화하면 암호화 프로세스가 다음과 같이 작동합니다.

1. 각 클러스터 노드가 시작되면 클러스터에 대해 로컬 디스크 암호화를 활성화할 때 선택한 KMS 키의 키 ID를 AWS KMS지정하여 GenerateDataKey 요청을에 보냅니다.
2. AWS KMS 는 고유한 데이터 암호화 키(데이터 키)를 생성한 다음이 데이터 키의 복사본 2개를 노드로 보냅니다. 사본 하나는 암호화되지 않고(일반 텍스트), 다른 하나는 KMS 키 하에서 암호화됩니다.
3. 노드는 LUKS 키를 보호하는 암호로 일반 텍스트 데이터 키의 base64 인코딩된 버전을 사용합니다. 노드는 부팅 볼륨에 데이터 키의 암호화된 사본을 저장합니다.
4. 노드가 재부팅되면 재부팅된 노드는 Decrypt 요청과 AWS KMS 함께 암호화된 데이터 키를에 전송합니다.
5. AWS KMS 는 암호화에 사용된 것과 동일한 KMS 키를 사용하여 암호화된 데이터 키를 복호화한 다음 복호화된(일반 텍스트) 데이터 키를 노드로 전송합니다.
6. 노드는 LUKS 키의 잠금을 해제하는 암호로 일반 텍스트 데이터 키의 base64 인코딩된 버전을 사용합니다.

암호화 컨텍스트

와 통합된 각 AWS 서비스는 서비스가를 사용하여 데이터 키를 생성하거나 데이터를 암호화 또는 복호화 AWS KMS 할 때 암호화 컨텍스트를 지정할 AWS KMS 수 있습니다. 암호화 컨텍스트는가 데이터 무결성을 확인하는 데 AWS KMS 사용하는 추가 인증 정보입니다. 서비스가 암호화 작업에 대해 암호화 컨텍스트를 지정할 때 해당 해독 작업에 대해서 동일한 암호화 컨텍스트를 지정해야 합니다. 그렇지 않으면 해독이 실패합니다. 암호화 컨텍스트는 AWS CloudTrail 로그 파일에도 작성되므로 특정 KMS 키가 사용된 이유를 이해하는 데 도움이 될 수 있습니다.

다음 섹션에서는 KMS 키를 사용하는 각 Amazon EMR 암호화 시나리오에 사용되는 암호화 컨텍스트를 설명합니다.

SSE-KMS를 이용한 EMRFS 암호화를 위한 암호화 컨텍스트

SSE-KMS를 통해 Amazon EMR 클러스터가 Amazon S3로 데이터를 보내면 Amazon S3가 S3 버킷에 데이터를 저장하기 전에 KMS 키를 이용해 암호화합니다. 이 경우 Amazon S3는 S3 객체의 Amazon 리소스 이름(ARN)을 각 GenerateDataKey 및 Decrypt 요청과 함께 암호화 컨텍스트로 사용합니다 AWS KMS. 다음 예는 Amazon S3가 사용하는 암호화 컨텍스트의 JSON 표시를 보여줍니다.

```
{ "aws:s3:arn" : "arn:aws:s3:::S3_bucket_name/S3_object_key" }
```

CSE-KMS를 이용한 EMRFS 암호화를 위한 암호화 컨텍스트

CSE-KMS에서는 Amazon EMR 클러스터가 데이터를 Amazon S3로 보내 저장하기 전에 KMS 키를 이용해 암호화합니다. 이 경우 클러스터는 보내는 각 GenerateDataKey 및 Decrypt 요청과 함께 KMS 키의 Amazon 리소스 이름(ARN)을 암호화 컨텍스트로 사용합니다 AWS KMS. 다음 예는 클러스터가 사용하는 암호화 컨텍스트의 JSON 표시를 보여줍니다.

```
{ "kms_cmk_id" : "arn:aws:kms:us-
east-2:111122223333:key/0987ab65-43cd-21ef-09ab-87654321cdef" }
```

LUKS를 이용한 로컬 디스크 암호화를 위한 암호화 컨텍스트

Amazon EMR 클러스터가 LUKS로 로컬 디스크 암호화를 사용하는 경우 클러스터 노드는 AWS KMS로 보내는 GenerateDataKey 및 Decrypt 요청으로 암호화 컨텍스트를 지정하지 않습니다.

Amazon Redshift의 사용 방식 AWS KMS

이 주제에서는 Amazon Redshift가 AWS KMS 를 사용하여 데이터를 암호화하는 방법을 설명합니다.

주제

- Amazon Redshift 암호화
- 암호화 컨텍스트

Amazon Redshift 암호화

Amazon Redshift 데이터 웨어하우스는 노드라는 컴퓨팅 리소스의 모음으로, 노드는 클러스터라는 그룹을 구성합니다. 각 클러스터는 Amazon Redshift 엔진을 실행하며, 하나 이상의 데이터베이스를 포함합니다.

Amazon Redshift는 암호화를 위해 4개 티어, 키 기반 계층 구조를 사용합니다. 이 계층 구조는 데이터 암호화 키, 데이터베이스 키, 클러스터 키, 루트 키로 구성됩니다. 를 루트 키 AWS KMS key 로 사용할 수 있습니다.

데이터 암호화 키는 클러스터의 데이터 블록을 암호화합니다. 각 데이터 블록은 임의로 생성된 AES-256에 할당됩니다. 이러한 키는 클러스터에 대한 데이터베이스 키를 사용해 암호화됩니다.

데이터베이스 키는 클러스터의 데이터 암호화 키를 암호화합니다. 데이터베이스 키는 임의로 생성된 AES-256 키입니다. 이 키는 Amazon Redshift 클러스터와 분리된 네트워크의 디스크에 저장되며 보안 채널을 통해 클러스터로 전달됩니다.

클러스터 키는 Amazon Redshift 클러스터에 대한 데이터베이스 키를 암호화합니다. AWS KMS AWS CloudHSM, 또는 외부 하드웨어 보안 모듈(HSM)을 사용하여 클러스터 키를 관리할 수 있습니다. 자세한 내용은 Amazon Redshift 데이터베이스 암호화 문서를 참조하십시오.

Amazon Redshift 콘솔에서 해당 확인란을 선택하여 암호화를 요청할 수 있습니다. 암호화 상자 아래에 표시되는 목록에서 하나를 선택해 사용할 고객 관리형 키를 지정할 수 있습니다. 고객 관리형 키를 지정하지 않으면 Amazon Redshift는 계정에서 Amazon Redshift용 AWS 관리형 키를 사용합니다.

> **Important**
>
> Amazon Redshift는 대칭 암호화 KMS 키만 지원합니다. Amazon Redshift 암호화 워크플로에서 비대칭 KMS 키를 사용할 수 없습니다. KMS 키가 대칭 또는 비대칭인지 여부를 확인하는 방법은 다양한 키 유형 식별 섹션을 참조하세요.

암호화 컨텍스트

와 통합된 각 서비스는 데이터 키 요청, 암호화 및 복호화 시 암호화 컨텍스트를 AWS KMS 지정합니다. 암호화 컨텍스트는가 데이터 무결성을 확인하는 데 AWS KMS 사용하는 추가 인증 데이터(AAD)입니다. 즉 암호화 작업에 대해 암호화 컨텍스트가 지정되면 서비스가 해독 작업에 대해서도 이를 지정합니다. 그렇지 않으면 해독이 실패하게 됩니다. Amazon Redshift는 암호화 컨텍스트에 대해 클러스터 ID와 생성 시간을 사용합니다. CloudTrail 로그 파일의 `requestParameters` 필드에 암호화 컨텍스트가 이와 비슷하게 표시됩니다.

```
"encryptionContext": {
    "aws:redshift:arn": "arn:aws:redshift:region:account_ID:cluster:cluster_name",
    "aws:redshift:createtime": "20150206T1832Z"
},
```

CloudTrail 로그에서 클러스터 이름을 검색하여 AWS KMS key (KMS 키)를 사용하여 수행된 작업을 파악할 수 있습니다. 이러한 작업으로는 클러스터 암호화, 클러스터 해독, 데이터 키 생성 등이 있습니다.

AWS KMS 참조

다음 참조 자료에서는 KMS 키 사용 및 관리에 대한 유용한 정보를 제공합니다.

- 키 유형 참조 각 AWS KMS API 작업을 지원하는 KMS 키의 유형을 나열합니다.

  찾는 방법: RSA 서명 KMS 키 활성화 및 비활성화를 설정할 수 있나요?
- 키 상태 테이블 KMS 키의 키 상태가 AWS KMS API 작업에서의 사용에 미치는 영향을 보여줍니다.

  찾는 방법: 삭제 보류 중인 KMS 키의 별칭을 변경할 수 있나요?
- AWS KMS API 권한 참조. 각 AWS KMS API 작업에 필요한 권한에 대한 정보를 제공합니다.

  찾으려면: 다른 AWS 계정의 키에서 GetKeyPolicy를 실행할 수 있나요? IAM 정책에서 `kms:Decrypt` 권한을 허용할 수 있나요?
  - ViaService 참조 `kms:ViaService` 조건 키를 지원하는 AWS 서비스를 나열합니다.

    찾는 방법: Amazon ElastiCache에서 오는 경우에만 권한을 허용하기 위해 `kms:ViaService` 조건 키를 사용할 수 있나요? Amazon Neptune은 어떤가요?
- AWS KMS 요금. KMS 키의 요금을 나열하고 설명합니다.

  찾는 방법: 비대칭 키를 사용하는 데 비용이 얼마나 드나요?
- AWS KMS 요청 할당량입니다. 각 계정 및 리전의 AWS KMS API 요청에 대한 초당 할당량을 나열합니다.

  찾는 방법: 매초마다 몇 개의 복호화 요청을 실행할 수 있나요? 사용자 지정 키 스토어의 KMS 키에서 몇 개의 복호화 요청을 실행할 수 있나요?
- AWS KMS 리소스 할당량입니다. AWS KMS 리소스의 할당량을 나열합니다.

  찾는 방법: 계정의 각 리전에 몇 개의 KMS 키를 추가할 수 있나요? 각 KMS 키에 몇 개의 별칭을 추가할 수 있나요?
- AWS 와 통합된 서비스 AWS KMS. KMS 키를 사용하여 생성, 저장 및 관리하는 리소스를 보호하는 AWS 서비스를 나열합니다.

  찾으려면: Connect 고객이 KMS 키를 사용하여 Connect 리소스를 보호하나요?

키의 AWS KMS 키 상태

에는 AWS KMS key 항상 키 상태가 있습니다. KMS 키 및 해당 환경에서 수행되는 작업은 키 상태를 변경할 수 있습니다. 키 상태는 일시적으로 변경될 수 있으며, 다른 작업이 키 상태를 다시 변경할 때까지 그 상태가 유지될 수 있습니다. 이러한 작업은 비동기적으로 또는 API 직접 호출을 통해 수행됩니다.

이 섹션의 표는 키 상태가 AWS KMS API 작업 호출에 미치는 영향을 보여줍니다. 키 상태의 결과로 KMS 키에 대한 작업은 성공(#), 실패(X) 또는 특정 조건에서만 성공(?)할 것으로 예상됩니다. 결과는 종종 가져온 키 구성 요소가 있는 KMS 키에 따라 다릅니다.

이 테이블에는 기존 KMS 키를 사용하는 API 작업만 포함됩니다. 다른 작업(예: CreateKey 및 ListKeys)은 생략됩니다.

주제

- 키 상태 및 KMS 키 유형
- 키 상태 테이블

키 상태 및 KMS 키 유형

KMS 키의 유형에 따라 키가 가질 수 있는 상태가 결정됩니다.

- 모든 KMS 키는 `Enabled`, `Disabled`, 및 `PendingDeletion` 상태일 수 있습니다.
- 대부분의 KMS 키는 `Enabled` 상태에서 생성됩니다. 가져온 키 자료가 있는 키는 `PendingImport` 상태에서 생성됩니다.
- 이 `PendingImport` 상태는 가져온 키 구성 요소가 있는 KMS 키에만 적용됩니다. 가져온 키의 키 구성 요소가 삭제되거나 만료되면 키 상태는 `Enabled`에서 `PendingImport`로 변경됩니다.
- `Unavailable` 상태는 사용자 지정 키 스토어의 KMS 키에만 적용됩니다. AWS CloudHSM 키 스토어의 KMS 키는 사용자 지정 키 스토어가 의도적으로 AWS CloudHSM 클러스터에서 연결 해제된 `Unavailable` 경우입니다. 외부 키 스토어가 외부 키 스토어 프록시에서 의도적으로 연결 해제된 경우 외부 키 스토어의 KMS 키는 `Unavailable`입니다. 사용할 수 없는 KMS 키를 확인 및 관리할 수 있지만, 암호화 작업에서 이들을 사용할 수 없습니다.

  사용자 지정 키 스토어에 있는 KMS 키의 키 상태는 백업 키가 변경되어도 영향을 받지 않습니다. AWS CloudHSM 키 스토어의 KMS 키는 AWS CloudHSM 클러스터의 연결된 키 구성 요소에 대한 변경의 영향을 받지 않습니다. 외부 키 스토어의 KMS 키는 외부 키 관리자의 외부 키가 변경되어

도 영향을 받지 않습니다. 백업 키가 비활성화되거나 삭제된 경우 KMS 키 상태는 변경되지 않지만 KMS 키를 사용한 암호화 작업은 실패합니다.

- `Creating`, `Updating` 및`PendingReplicaDeletion` 키 상태는 다중 리전 키에만 적용됩니다.
  - 다중 리전 복제 키는 생성되는 동안 일시적인 `Creating` 키 상태에 있습니다. 이 프로세스는 ReplicateKey 작업이 완료될 때 계속 진행 중일 수 있습니다. 복제 프로세스가 완료되면 복제 키는`Enabled` 또는 `PendingImport` 상태입니다.
  - 다중 리전 키는 일시적인 `Updating` 키 상태를 유지하며 기본 리전을 업데이트할 수 있습니다. 이 프로세스는 UpdatePrimaryRegion 작업이 완료될 때 계속 진행 중일 수 있습니다. 업데이트 프로세스가 완료되면 기본 키와 복제본 키가 `Enabled` 키 상태를 재개합니다.
  - 복제 키가 있는 다중 리전 기본 키의 삭제를 예약하면 모든 복제 키가 삭제될 때까지 기본 키는 `PendingReplicaDeletion` 상태입니다. 그런 다음 키 상태가 `PendingDeletion`로 변경됩니다. 자세한 내용은 Deleting multi-Region keys을 참조하세요.

키 상태 테이블

다음 표에서는 KMS 키의 키 상태가 AWS KMS 작업에 미치는 영향을 보여줍니다.

번호가 매겨진 각주에 대한 설명([n])은 이 주제의 끝 부분에 있습니다.

> **Note**
>
> 이 테이블의 모든 데이터를 보려면 가로 또는 세로로 스크롤해야 할 수도 있습니다.

API	활성화됨	비활성화됨	삭제 보류 중 복제본 삭제 보류 중	가져오기 보류 중	Unavailable	생성 중	업데이트 중
CancelKeyDeletion	✗ [4]	✗ [4]	✓	✗ [4]	✗ [4], [13]	✗ [4]	✗ [4]

API	활성화됨	비활성화됨	삭제 보류 중 복제본 삭제 보류 중	가져오기 보류 중	Unavailable	생성 중	업데이트 중
CreateAlias	✓	✓	⊗ [3]	✓	✓	✓	✓
CreateGrant	✓	⊗ [1]	⊗ [2] 또는 [3]	⊗ [5]	✓	⊗ [14]	✓
Decrypt	✓	⊗ [1]	⊗ [2] 또는 [3]	⊗ [5]	⊗ [11]	⊗ [14]	✓
DeleteAlias	✓	✓	✓	✓	✓	✓	✓
DeleteImportedKeyMaterial	✓ [9]	✓ [9]	✓ [9]	✓	해당 사항 없음	⊗ [14]	⊗ [15]
DeriveSharedSecret	✓	⊗ [1]	⊗ [2] 또는 [3]	⊗ [5]	해당 사항 없음	⊗ [14]	✓
DescribeKey	✓	✓	✓	✓	✓	✓	✓

API	활성화됨	비활성화됨	삭제 보류 중 복제본 삭제 보류 중	가져오기 보류 중	Unavailable	생성 중	업데이트 중
DisableKey	✓	✓	⊗ [3]	⊗ [5]	✓ [12]	⊗ [14]	⊗ [15]
DisableKeyRotation	? [7]	⊗ [1] 또는 [7]	⊗ [3] 또는 [7]	⊗ [6]	⊗ [7]	⊗ [14]	? [7]
EnableKey	✓	✓	⊗ [3]	⊗ [5]	✓ [12]	⊗ [14]	⊗ [15]
EnableKeyRotation	? [7]	⊗ [1] 또는 [7]	⊗ [3] 또는 [7]	⊗ [6]	⊗ [7]	⊗ [14]	? [7]
암호화	✓	⊗ [1]	⊗ [2] 또는 [3]	⊗ [5]	⊗ [11]	⊗ [14]	✓
GenerateDataKey	✓	⊗ [1]	⊗ [2] 또는 [3]	⊗ [5]	⊗ [11]	⊗ [14]	✓

API	활성화됨	비활성화됨	삭제 보류 중 복제본 삭제 보류 중	가져오기 보류 중	Unavailable	생성 중	업데이트 중
GenerateDataKeyPair	✓	⊗ [1]	⊗ [2] 또는 [3]	⊗ [5]	⊗ [7]	⊗ [14]	✓
GenerateDataKeyPairWithoutPlaintext	✓	⊗ [1]	⊗ [2] 또는 [3]	⊗ [5]	⊗ [7]	⊗ [14]	✓
GenerateDataKeyWithoutPlaintext	✓	⊗ [1]	⊗ [2] 또는 [3]	⊗ [5]	⊗ [11]	⊗ [14]	✓
GenerateMac	✓	⊗ [1]	⊗ [2] 또는 [3]	⊗ [5]	해당 사항 없음	⊗ [14]	✓
GetKeyPolicy	✓	✓	✓	✓	✓	✓	✓
GetKeyRotationStatus	? [7]	? [7]	? [7]	⊗ [6]	⊗ [7]	? [7]	? [7]

API	활성화됨	비활성화됨	삭제 보류 중 복제본 삭제 보류 중	가져오기 보류 중	Unavailable	생성 중	업데이트 중
GetParametersForImport	 [9]	 [9]	 [8] 또는 [9]		 [9]	 [14]	 [15]
GetPublicKey			 [2] 또는 [3]		해당 사항 없음	 [14]	
ImportKeyMaterial	 [9]	 [9]	 [9]		 [9]	 [14]	
ListAliases							
ListGrants							
ListKeyPolicies							
ListKeyRotations	[7]	[7]	[7]		[7]	[7]	[7]
ListResourceTags							

API	활성화됨	비활성화됨	삭제 보류 중 복제본 삭제 보류 중	가져오기 보류 중	Unavailable	생성 중	업데이트 중
PutKeyPolicy	✓	✓	✓	✓	✓	✓	✓
ReEncrypt	✓	⊗ [1]	⊗ [2] 또는 [3]	⊗ [5]	⊗ [11]	⊗ [14]	✓
ReplicateKey	✓	⊗ [1]	⊗ [2] 또는 [3]	⊗ [5]	해당 사항 없음	⊗ [14]	⊗ [15]
RetireGrant	✓	✓	✓	✓	✓	✓	✓
RevokeGrant	✓	✓	✓	✓	✓	✓	✓
RotateKeyOnDemand	? [7]	⊗ [1] 또는 [7]	⊗ [3] 또는 [7]	⊗ [5]	⊗ [7]	⊗ [14]	? [7]
ScheduleKeyDeletion	✓	✓	⊗ [3]	✓	✓	✓	⊗ [15]

API	활성화됨	비활성화됨	삭제 보류 중 복제본 삭제 보류 중	가져오기 보류 중	Unavailable	생성 중	업데이트 중
Sign	✓	⊗ [1]	⊗ [2] 또는 [3]	⊗ [5]	해당 사항 없음	⊗ [14]	✓
TagResource	✓	✓	⊗ [3]	✓	✓	✓	✓
UntagResource	✓	✓	⊗ [3]	✓	✓	✓	✓
UpdateAlias	✓	✓	? [10]	✓	✓	✓	✓
UpdateKeyDescription	✓	✓	⊗ [3]	✓	✓	✓	✓
UpdatePrimaryRegion	✓	⊗ [1]	⊗ [2] 또는 [3]	⊗ [5]	해당 사항 없음	⊗ [14]	✓

API	활성화됨	비활성화됨	삭제 보류 중 복제본 삭제 보류 중	가져오기 보류 중	Unavailable	생성 중	업데이트 중
확인		 [1]	 [2] 또는 [3]	 [5]	해당 사항 없음	 [14]	
VerifyMac		 [1]	 [2] 또는 [3]	 [5]	해당 사항 없음	 [14]	

테이블 세부 정보

- [1] DisabledException: *<key ARN>* is disabled.
- [2] DisabledException: *<key ARN>* is pending deletion (or pending replica deletion).
- [3] KMSInvalidStateException: *<key ARN>* is pending deletion (or pending replica deletion).
- [4] KMSInvalidStateException: *<key ARN>* is not pending deletion (or pending replica deletion).
- [5] KMSInvalidStateException: *<key ARN>* is pending import because no key material has ever been imported or one of the imported key materials is deleted or expired.
- [6] UnsupportedOperationException: *<key ARN>* origin is EXTERNAL which is not valid for this operation.
- [7] KMS 키가 사용자 지정 키 저장소에 있는 경우: UnsupportedOperationException.
- [8] KMS 키에 가져온 키 구성 요소가 있는 경우: KMSInvalidStateException.
- [9] KMS 키에 키 구성 요소를 가져올 수 없는 경우: UnsupportedOperationException.

- [10] 소스 KMS 키가 삭제 보류 중인 경우 명령이 성공합니다. 대상 KMS 키가 삭제 보류 중인 경우 명령은 오류와 함께 실패합니다. `KMSInvalidStateException : <key ARN> is pending deletion.`
- [11] `KMSInvalidStateException: <key ARN> is unavailable.` 사용할 수 없는 KMS 키에서 이 작업을 수행할 수 없습니다.
- [12] 작업이 성공하지만, KMS 키의 키 상태는 사용 가능한 상태가 될 때까지 변경되지 않습니다.
- [13] 사용자 지정 키 스토어의 KMS 키에서 삭제가 보류 중인 동안 KMS 키가 사용 불가능한 상태더라도 키 상태는 `PendingDeletion`으로 유지됩니다. 따라서 대기 시간 동안 언제라도 KMS 키의 삭제를 취소할 수 있습니다.
- [14]는 다중 리전 키()를 복제하는 동안이 예외를 `KMSInvalidStateException: <key ARN> is creating.` AWS KMS 발생시킵니다`ReplicateKey`.
- [15] 다중 리전 키()의 기본 리전을 업데이트하는 동안이 예외가 `KMSInvalidStateException: <key ARN> is updating.` AWS KMS 발생합니다`UpdatePrimaryRegion`.

키 유형 참조

AWS KMS 는 다양한 유형의 KMS 키에 대해 다양한 기능을 지원합니다. 예를 들어 대칭 암호화 KMS 키만 사용하여 대칭 데이터 키와 비대칭 데이터 키 페어를 생성할 수 있습니다. 또한 키 구성 요소 가져오기 및 자동 키 교체는 대칭 암호화 KMS 키에 대해서만 지원되며 사용자 지정 키 저장소에서는 대칭 암호화 KMS 키만 생성할 수 있습니다.

이 참조에는 두 개의 표가 포함되어 있습니다.

- 키 유형 테이블에는 대칭 암호화 KMS 키, 비대칭 KMS 키 및 HMAC KMS 키에 유효한 AWS KMS 작업이 나열되어 있습니다.
- 특수 기능 표에는 다중 리전 KMS 키, 가져온 키 구성 요소가 있는 KMS 키 및 사용자 지정 키 스토어의 KMS 키에 유효한 AWS KMS 작업이 나열되어 있습니다.

키 유형 표

이 테이블의 모든 데이터를 보려면 가로 또는 세로로 스크롤해야 할 수도 있습니다.

AWS KMS API 작업	대칭 암호화 KMS 키	HMAC KMS 키	비대칭 KMS 키 (ENCRYP _DECRYF	비대칭 KMS 키 (SIGN_VERIFY)	비대칭 KMS 키(KEY_AGR EEMENT)
CancelKeyDeletion	예	예	예	예	예
CreateAlias	예	예	예	예	예
CreateGrant	예	예	예	예	예
CreateKey	예	예	예	예	예
Decrypt	예	아니요	예	아니요	아니요
DeleteAlias	예	예	예	예	예
DeleteImp ortedKeyMaterial 가져온 키 구성 요소가 있는 KMS 키에서만 유효(`Origin`이 `EXTERNAL`임)	예	예	예	예	예
DeriveSha redSecret	아니요	아니요	아니요	아니요	예
DescribeKey	예	예	예	예	예
DisableKey	예	예	예	예	예
DisableKe yRotation	예 키 구성 요소가 있는 KMS AWS	아니요	아니요	아니요	아니요

AWS KMS API 작업	대칭 암호화 KMS 키	HMAC KMS 키	비대칭 KMS 키 (ENCRYP _DECRYF	비대칭 KMS 키 (SIGN_VERIFY)	비대칭 KMS 키(KEY_AGR EEMENT)
	KMS 키에 만 유효 합니다 (Origin는 AWS_KMS)				
EnableKey	예	예	예	예	예
EnableKeyRotation	예 키 구성 요소가 있는 KMS AWS KMS 키에 만 유효 합니다 (Origin는 AWS_KMS)	아니요	아니요	아니요	아니요
암호화	예	아니요	예	아니요	아니요
GenerateDataKey	예	아니요	아니요	아니요	아니요

AWS KMS API 작업	대칭 암호화 KMS 키	HMAC KMS 키	비대칭 KMS 키 (ENCRYP _DECRYF	비대칭 KMS 키 (SIGN_VERIFY)	비대칭 KMS 키(KEY_AGR EEMENT)
GenerateD ataKeyPair 대칭 암호화 KMS 키로 보호되는 비대칭 데이터 키 페어를 생성합니다.	예 사용자 지정 키 스토어의 KMS 키에는 유효하지 않습니다.	아니요	아니요	아니요	아니요
GenerateD ataKeyPai rWithoutPlaintext 대칭 암호화 KMS 키로 보호되는 비대칭 데이터 키 페어를 생성합니다.	예 사용자 지정 키 스토어의 KMS 키에는 유효하지 않습니다.	아니요	아니요	아니요	아니요
GenerateD ataKeyWit houtPlaintext	예	아니요	아니요	아니요	아니요
GenerateMac	아니요	예	아니요	아니요	아니요
GetKeyPolicy	예	예	예	예	예

AWS KMS API 작업	대칭 암호화 KMS 키	HMAC KMS 키	비대칭 KMS 키 (ENCRYP _DECRYF	비대칭 KMS 키 (SIGN_VERIFY)	비대칭 KMS 키(KEY_AGR EEMENT)
GetKeyRotationStatus	예	예 (KeyRota onEnabl 는 항상 false.)	예 (KeyRota onEnabl 는 항상 false.)	예 (KeyRotati onEnabled 는 항상 false.)	예 (KeyRotati onEnabled 는 항상 false.)
GetParametersForImport 가져온 키 구성 요소가 있는 KMS 키에서만 유효(Origin이 EXTERNAL임)	예	예	예	예	예
GetPublicKey	아니요	아니요	예	예	예
ImportKeyMaterial 가져온 키 구성 요소가 있는 KMS 키에서만 유효(Origin이 EXTERNAL임)	예	예	예	예	예
ListAliases	예	예	예	예	예
ListGrants	예	예	예	예	예
ListKeyPolicies	예	예	예	예	예

AWS KMS API 작업	대칭 암호화 KMS 키	HMAC KMS 키	비대칭 KMS 키 (ENCRYP _DECRYF	비대칭 KMS 키 (SIGN_VERIFY)	비대칭 KMS 키(KEY_AGR EEMENT)
ListKeyRotations	예	예 (Rotatic 필드는 항상 null이거나 비어 있습니다.)	예 (Rotatic 필드는 항상 null이거나 비어 있습니다.)	예 (Rotations 필드는 항상 null이거나 비어 있습니다.)	예 (Rotations 필드는 항상 null이거나 비어 있습니다.)
ListResourceTags	예	예	예	예	예
ListRetirableGrants	예	예	예	예	예
PutKeyPolicy	예	예	예	예	예
ReEncrypt	예	아니요	예	아니요	아니요
ReplicateKey - 다중 리전 키에만 유효	예	예	예	예	예
RetireGrant	예	예	예	예	예
RevokeGrant	예	예	예	예	예

AWS KMS API 작업	대칭 암호화 KMS 키	HMAC KMS 키	비대칭 KMS 키 (ENCRYP _DECRYF	비대칭 KMS 키 (SIGN_VERIFY)	비대칭 KMS 키(KEY_AGR EEMENT)
RotateKey OnDemand	예 오리진이 AWS_KMS 또는 EXTERNAI 고객 관리형 대칭 암호화 KMS 키에만 유효합니다.	아니요	아니요	아니요	아니요
ScheduleK eyDeletion	예	예	예	예	예
Sign	아니요	아니요	아니요	예	아니요
TagResource	예	예	예	예	예
UntagResource	예	예	예	예	예
UpdateAlias 현재 KMS 키와 새 KMS 키는 동일한 유형(모두 대칭, 모두 비대칭 또는 모두 HMAC)이어야 하며 키 사용이 동일해야 합니다.	예	예	예	예	예

AWS KMS API 작업	대칭 암호화 KMS 키	HMAC KMS 키	비대칭 KMS 키 (ENCRYP _DECRYF	비대칭 KMS 키 (SIGN_VERIFY)	비대칭 KMS 키(KEY_AGR EEMENT)
UpdateKey Description	예	예	예	예	예
UpdateRep licaRegion - 다중 리전 키에만 유효	예	예	예	예	예
Verify	아니요	아니요	아니요	예	아니요
VerifyMac	아니요	예	아니요	아니요	아니요

특수 기능 테이블

이 표에는 각 유형의 특수 용도 키에서 지원되는 AWS KMS API 작업이 나와 있습니다.

이 표를 읽는 동안 다음과 같은 상호 작용에 유의하세요.

- 다중 리전 키:
  - 다중 리전 키는 대칭 KMS 키, 비대칭 KMS 키, HMAC KMS 키 및 가져온 키 구성 요소가 있는 KMS 키일 수 있습니다.
  - 사용자 지정 키 스토어에서는 다중 리전 키를 만들 수 없습니다.
- 가져온 키 구성 요소
  - 대칭 암호화 KMS 키, 비대칭 KMS 키 및 HMAC KMS 키에 대한 키 구성 요소를 가져올 수 있습니다.
  - 가져온 키 구성 요소가 있는 다중 리전 키를 생성할 수 있습니다.
  - 사용자 지정 키 스토어에 가져온 키 구성 요소가 있는 키를 생성할 수 없습니다.
  - 가져온 키 구성 요소가 있는 KMS 키에는 자동 키 교체(`EnableKeyRotation`, `DisableKeyRotation`)가 지원되지 않습니다.

  - 온디맨드 키 교체(RotateKeyOnDemand)는 가져온 키 구성 요소가 있는 대칭 암호화 KMS 키에 대해 지원됩니다.
- 사용자 지정 키 스토어
  - 사용자 지정 키 스토어는 대칭 암호화 KMS 키만 지원합니다.
  - 비대칭 키 페어(GenerateDataKeyPair, GenerateDataKeyPairWithoutPlaintext)에 대한 대칭 작업은 사용자 지정 키 스토어의 KMS 키에서 지원되지 않습니다.
  - 사용자 지정 키 스토어의 KMS 키에는 자동 키 교체(EnableKeyRotation, DisableKeyRotation)가 지원되지 않습니다.
  - 사용자 지정 키 스토어에서는 다중 리전 키를 생성할 수 없습니다.

이 테이블의 모든 데이터를 보려면 가로 또는 세로로 스크롤해야 할 수도 있습니다.

AWS KMS API 작업	다중 리전 키	가져온 키 구성 요소	사용자 지정 키 스토어의 KMS 키
CancelKeyDeletion	✓	✓	✓
CreateAlias	✓	✓	✓
CreateGrant	✓	✓	✓
CreateKey CreateKey 를 사용하여 다중 리전 프라이머리 키, 가져온 키 구성 요소가 있는 KMS 키 또는 사용자 지정 키 스토어의 KMS 키를 생성할 수 있습니다. 다중 리전 복제본 키를 생성하려면 ReplicateKey 를 사용합니다.	✓	✓	✓
Decrypt	✓ KeyUsage이 ENCRYPT_D	✓	✓

AWS KMS API 작업	다중 리전 키	가져온 키 구성 요소	사용자 지정 키 스토어의 KMS 키
	ECRYPT 인 경우에만 유효		
DeleteAlias	✓	✓	✓
DeleteImportedKeyMaterial	✓ 가져온 키 구성 요소가 있는 키에만 유효(Origin이 EXTERNAL임)	✓	⊗
DeriveSharedSecret	✓ KeyUsage가 KEY_AGREEMENT 인 경우에만 유효)	✓ KeyUsage가 KEY_AGREEMENT 인 경우에만 유효)	⊗
DescribeKey	✓	✓	✓
DisableKey	✓	✓	✓
DisableKeyRotation	✓ 키 구성 요소가 있는 대칭 암호화 AWS KMS 키에만 유효합니다(Origin는 AWS_KMS).	⊗	⊗

AWS KMS API 작업	다중 리전 키	가져온 키 구성 요소	사용자 지정 키 스토어의 KMS 키
EnableKey	✓ 대칭 암호화 KMS 키에서만 유효	✓	✓
EnableKeyRotation	✓ 키 구성 요소가 있는 대칭 암호화 AWS KMS 키에만 유효합니다(`Origin`는 `AWS_KMS`).	⊗	⊗
암호화	✓ `KeyUsage`이 `ENCRYPT_DECRYPT` 인 경우에만 유효	✓	✓
GenerateDataKey	✓ 대칭 암호화 KMS 키에서만 유효	✓	✓

AWS KMS API 작업	다중 리전 키	가져온 키 구성 요소	사용자 지정 키 스토어의 KMS 키
GenerateDataKeyPair	✓ 대칭 암호화 KMS 키에서만 유효	✓	⊗
GenerateDataKeyPairWithoutPlaintext	✓ 대칭 암호화 KMS 키에서만 유효	✓	⊗
GenerateDataKeyWithoutPlaintext	✓ 대칭 암호화 KMS 키에서만 유효	✓	✓
GenerateMac HMAC KMS 키에서만 유효	✓	✓	⊗
GetKeyPolicy	✓	✓	✓
GetKeyRotationStatus	✓	✓ (KeyRotationEnabled 는 항상 false.)	⊗

AWS KMS API 작업	다중 리전 키	가져온 키 구성 요소	사용자 지정 키 스토어의 KMS 키
GetParametersForImport	✓ 가져온 키 구성 요소가 있는 키에만 유효(Origin이 EXTERNAL임)	✓	⊗
GetPublicKey 비대칭 KMS 키에만 유효	✓	✓	⊗
ImportKeyMaterial	✓ 가져온 키 구성 요소가 있는 키에만 유효(Origin이 EXTERNAL임)	✓	⊗
ListAliases	✓	✓	✓
ListGrants	✓	✓	✓
ListKeyPolicies	✓	✓	✓

AWS KMS API 작업	다중 리전 키	가져온 키 구성 요소	사용자 지정 키 스토어의 KMS 키
ListKeyRotations	✓ AWS_KMS 또는 EXTERNAL 오리진이 있는 대칭 암호화 키에만 유효합니다.	✓ 대칭 암호화 키에만 유효합니다.	✓
ListResourceTags	✓	✓	✓
ListRetirableGrants	✓	✓	✓
PutKeyPolicy	✓	✓	✓
ReEncrypt	✓ KeyUsage이 ENCRYPT_DECRYPT 인 경우에만 유효	✓	✓
ReplicateKey	✓ 다중 리전 프라이머리 키에만 유효	✓ 다중 리전 프라이머리 키에만 유효	⊗
RetireGrant	✓	✓	✓
RevokeGrant	✓	✓	✓

AWS KMS API 작업	다중 리전 키	가져온 키 구성 요소	사용자 지정 키 스토어의 KMS 키
RotateKeyOnDemand	✓ AWS_KMS 또는 EXTERNAL 오리진이 있는 대칭 암호화 키에만 유효합니다.	✓ 대칭 암호화 키에만 유효합니다.	⊗
ScheduleKeyDeletion	✓	✓	✓
Sign KeyUsage가 SIGN_VERIFY 인 경우에만 유효	✓	✓	⊗
TagResource	✓	✓	✓
UntagResource	✓	✓	✓
UpdateAlias - 현재 KMS 키와 새 KMS 키는 동일한 유형(모두 대칭, 모두 비대칭 또는 모두 HMAC)이어야 하며 키 사용이 동일해야 합니다.	✓	✓	✓
UpdateKeyDescription	✓	✓	✓
UpdateReplicaRegion	✓	✓ 다중 리전 키에만 유효	⊗

AWS KMS API 작업	다중 리전 키	가져온 키 구성 요소	사용자 지정 키 스토어의 KMS 키
Verify KeyUsage이 SIGN_VERIFY 인 경우에만 유효합니다.	✓	✓	⊗
VerifyMac HMAC KMS 키에서만 유효	✓	✓	⊗

키 사양 참조

비대칭 KMS 또는 HMAC KMS 키를 생성할 때 키 사양(key spec)을 선택합니다. 모든의 속성인 키 사양은 KMS 키의 암호화 구성을 AWS KMS key나타냅니다. 키 사양은 KMS 키를 생성할 때 선택하며 이후에는 변경할 수 없습니다. 잘못된 키 사양을 선택한 경우 KMS 키를 삭제하고 새 KMS를 생성합니다.

> **Note**
>
> KMS 키의 키 사양은 “고객 마스터 키 사양”으로 알려졌습니다. CreateKey 작업의 CustomerMasterKeySpec 파라미터는 더 이상 사용되지 않습니다. 대신 KeySpec 파라미터를 사용합니다. CreateKey 및 DescribeKey 작업의 응답에는 값이 동일한 KeySpec 및 CustomerMasterKeySpec 멤버를 포함합니다.

키 사양은 KMS 키가 대칭인지 비대칭인지, KMS 키의 키 구성 요소 유형,가 KMS 키에 대해 AWS KMS 지원하는 암호화 알고리즘, 서명 알고리즘 또는 메시지 인증 코드(MAC) 알고리즘을 결정합니다. 선택하는 키 사양은 일반적으로 사용 사례 및 규정 요구 사항에 따라 결정됩니다. 그러나 키 사양이 다른 KMS 키에 대한 암호화 작업은 가격이 다르게 책정되며 할당량도 다릅니다. 요금에 대한 자세한 내용은 AWS Key Management Service Pricing을 참조하세요. 요청 할당량에 대한 자세한 내용은 요청 할당량 섹션을 참조하세요.

KMS 키를 생성할 때 보안 주체가 사용할 수 있는 키 사양을 제한하려면 kms:KeySpec 조건 키를 사용합니다. kms:KeySpec 조건 키를 사용하여 보안 주체가 특정 키 사양이 있는 KMS 키에서만 AWS KMS 작업을 호출하도록 허용할 수도 있습니다. 예를 들어 RSA_4096 키 사양을 사용하여 KMS 키 삭제를 예약할 수 있는 권한을 거부할 수 있습니다.

AWS KMS 는 KMS 키에 대해 다음과 같은 키 사양을 지원합니다.

대칭 암호화 키 사양(기본값)

- SYMMETRIC_DEFAULT

RSA 키 사양(암호화 및 복호화 또는 서명 및 확인)

- RSA_2048
- RSA_3072
- RSA_4096

타원 곡선 키 사양

- 비대칭 NIST 표준 타원 곡선 키 페어(서명 및 확인 또는 공유 보안 암호 도출)
  - ECC_NIST_P256(secp256r1)
  - ECC_NIST_P384(secp384r1)
  - ECC_NIST_P521(secp521r1)
  - ECC_NIST_EDWARDS25519(ed25519) - 서명 및 확인 전용
    - 참고: ECC_NIST_EDWARDS25519 KMS 키의 경우 ED25519_SHA_512 서명 알고리즘에는가 필요한 반면`MessageType:RAW`, ED25519_PH_SHA_512에는가 필요합니다`MessageType:DIGEST`. 이러한 메시지 유형은 서로 바꿔서 사용할 수 없습니다.
- 기타 비대칭 타원 곡선 키 페어(서명 및 확인)
  - ECC_SECG_P256K1(secp256k1), 일반적으로 암호 화폐에 사용됨.

SM2 키 사양(암호화 및 복호화 또는 서명 및 확인 또는 공유 비밀 도출)

- SM2(중국 리전 전용)

HMAC 키 사양

- HMAC_224
- HMAC_256
- HMAC_384
- HMAC_512

ML-DSA 키 사양

- ML_DSA_44
- ML_DSA_65
- ML_DSA_87

SYMMETRIC_DEFAULT 키 사양

기본 키 사양인 SYMMETRIC_DEFAULT는 대칭 암호화 KMS 키의 키 사양입니다. AWS KMS 콘솔에서 대칭 키 유형과 암호화 및 복호화 키 사용을 선택하면 SYMMETRIC_DEFAULT 키 사양이 선택됩니다. CreateKey 작업에서 KeySpec 값을 지정하지 않으면 SYMMETRIC_DEFAULT가 선택됩니다. 다른 키 사양을 사용할 이유가 없다면 SYMMETRIC_DEFAULT가 좋은 선택입니다.

SYMMETRIC_DEFAULT는 보안 암호화를 위한 업계 표준인 256비트 키가 있는 Galois Counter Mode(GCM)의 고급 암호화 표준(AES)을 기반으로 하는 대칭 알고리즘인 AES-256-GCM을 나타냅니다. 이 알고리즘이 생성하는 암호화 텍스트는 암호화 컨텍스트와 같은 추가 인증 데이터(AAD)를 지원하며 GCM은 암호화 텍스트에 대한 추가 무결성 검사를 제공합니다.

AES-256-GCM으로 암호화된 데이터는 미래에도 보호됩니다. 암호 전문가는 이 알고리즘을 양자 저항으로 간주합니다. 256비트 AES-GCM 키로 생성된 암호화 텍스트에 대한 이론적인 미래의 대규모 양자 컴퓨팅 공격은 키의 효과적 보안을 128비트로 감소시킵니다. 그러나이 보안 수준은 AWS KMS 사이퍼텍스트에 대한 무차별 대입 공격을 실행 불가능하게 만들기에 충분합니다.

단, 중국 리전은 SYMMETRIC_DEFAULT가 SM4 암호화를 사용하는 128비트 대칭적 키를 나타냅니다. 중국 리전 내에서는 128비트 SM4 키만 생성할 수 있습니다. 중국 리전에서는 256비트 AES-GCM KMS 키를 생성할 수 없습니다.

에서 대칭 암호화 KMS 키를 사용하여 데이터를 암호화, AWS KMS 복호화 및 재암호화하고 생성된 데이터 키 및 데이터 키 페어를 보호할 수 있습니다.와 통합된 AWS 서비스는 대칭 암호화 KMS 키를 AWS KMS 사용하여 저장된 데이터를 암호화합니다. 대칭 암호화 KMS 키로 자체 키 구성 요소를 가져오고 사용자 지정 키 스토어에서 대칭 암호화 KMS 키를 생성할 수 있습니다. 대칭 및 비대칭 KMS 키에서 수행할 수 있는 작업을 비교하는 표는 대칭 및 비대칭 KMS 키 비교 섹션을 참조하세요.

에서 대칭 암호화 KMS 키를 사용하여 데이터를 암호화, 복호화 및 재암호화하고 데이터 키 및 데이터 키 페어 AWS KMS 를 생성할 수 있습니다. 다중 리전 대칭 암호화 KMS 키를 생성하고, 대칭 암호화 KMS 키로 자체 키 구성 요소를 가져오고, 사용자 지정 키 스토어에서 대칭 암호화 KMS 키를 생성할 수 있습니다. 유형별 KMS 키에 대해 수행할 수 있는 작업을 비교하는 표는 키 유형 참조 섹션을 참조하세요.

RSA 키 사양

RSA 키 사양을 사용하는 경우는 RSA 키 페어가 있는 비대칭 KMS 키를 AWS KMS 생성합니다. 프라이빗 키는 암호화 AWS KMS 되지 않은 상태로 두지 않습니다. 내에서 퍼블릭 키를 사용하거나 외부에서 사용할 퍼블릭 키를 AWS KMS다운로드할 수 있습니다 AWS KMS.

> **Warning**
>
> 외부에서 데이터를 암호화할 때는 사이퍼텍스트를 해독할 수 AWS KMS있어야 합니다. AWS KMS에서 삭제된 KMS 키의 퍼블릭 키, 서명 및 확인을 위해 구성된 KMS 키의 퍼블릭 키 또는 KMS 키에서 지원하지 않는 암호화 알고리즘을 사용하는 경우 데이터를 복구할 수 없습니다.

에서는 암호화 및 복호화 AWS KMS또는 서명 및 확인에 RSA 키 페어와 함께 비대칭 KMS 키를 사용할 수 있지만 둘 다 사용할 수는 없습니다. Key usage라고 하는 이 속성은 키 사양과는 별도로 결정되지만 키 사양을 선택하기 전에 먼저 결정해야 합니다.

AWS KMS 는 암호화 및 복호화 또는 서명 및 확인을 위해 다음과 같은 RSA 키 사양을 지원합니다.

- RSA_2048
- RSA_3072
- RSA_4096

RSA 키 사양은 RSA 키 길이(비트)에 따라 다릅니다. 선택하는 RSA 키 사양은 보안 표준 또는 작업 요구 사항에 따라 결정될 수 있습니다. 일반적으로 작업에 실용적이고 저렴한 키 중에서 가장 큰 키를 사용하세요. RSA 키 사양이 다른 KMS 키에 대한 암호화 작업의 경우에는 가격이 다르게 책정됩니다. AWS KMS 요금에 대한 자세한 내용은 AWS Key Management Service 요금을 참조하세요. 요청 할당량에 대한 자세한 내용은 요청 할당량 섹션을 참조하세요.

암호화 및 해독을 위한 RSA 키 사양

RSA 비대칭 KMS 키가 암호화 및 해독에 사용되는 경우 퍼블릭 키로 암호화하고 프라이빗 키로 해독합니다. RSA KMS 키에 AWS KMS 대해에서 `Encrypt` 작업을 호출하면는 RSA 키 페어의 퍼블릭 키와 사용자가 지정한 암호화 알고리즘을 AWS KMS 사용하여 데이터를 암호화합니다. 사이퍼텍스트를 복호화하려면 `Decrypt` 작업을 호출하고 동일한 KMS 키 및 암호화 알고리즘을 지정합니다. AWS KMS 그런 다음는 RSA 키 페어의 프라이빗 키를 사용하여 데이터를 복호화합니다.

퍼블릭 키를 다운로드하여 외부에서 데이터를 암호화하는 데 사용할 수도 있습니다 AWS KMS. RSA KMS 키에 대해가 AWS KMS 지원하는 암호화 알고리즘을 사용해야 합니다. 암호화 텍스트를 해독하려면 동일한 KMS 키 및 암호화 알고리즘을 사용하여 `Decrypt` 함수를 호출합니다.

AWS KMS 는 RSA 키 사양이 있는 KMS 키에 대해 두 가지 암호화 알고리즘을 지원합니다. PKCS #1 v2.2에 정의된 이러한 알고리즘은 내부적으로 사용하는 해시 함수가 다릅니다. AWS KMS에서 RSAES_OAEP 알고리즘은 해싱 용도와 마스크 생성 함수(MGF1)에 항상 동일한 해시 함수를 사용합

니다. 암호화(Encrypt) 및 해독(Decrypt) 작업을 호출할 때 암호화 알고리즘을 지정해야 합니다. 각 요청마다 다른 알고리즘을 선택할 수 있습니다.

RSA 키 사양에 지원되는 암호화 알고리즘

암호화 알고리즘	알고리즘 설명
RSAES_OAEP_SHA_1	PKCS #1 v2.2, Section 7.1. 빈 레이블과 함께 해시 및 MGF1 마스크 생성 기능 모두에 SHA-1을 사용하는 OAEP 패딩 포함 RSA 암호화입니다.
RSAES_OAEP_SHA_256	PKCS #1, Section 7.1. 빈 레이블과 함께 해시 및 MGF1 마스크 생성 기능 모두에 SHA-256을 사용하는 OAEP 패딩 포함 RSA 암호화입니다.

특정 암호화 알고리즘을 사용하도록 KMS 키를 구성할 수는 없습니다. 그러나 kms:EncryptionAlgorithm 정책 조건을 사용하여 보안 주체가 KMS 키와 함께 사용할 수 있는 암호화 알고리즘을 지정할 수 있습니다.

KMS 키의 암호화 알고리즘을 가져오려면 AWS KMS 콘솔에서 KMS 키의 암호화 구성을 보거나 DescribeKey 작업을 사용합니다. AWS KMS 는 AWS KMS 콘솔에서 또는 GetPublicKey 작업을 사용하여 퍼블릭 키를 다운로드할 때 키 사양 및 암호화 알고리즘도 제공합니다.

각 요청에서 암호화할 수 있는 일반 텍스트 데이터의 길이를 기준으로 RSA 키 사양을 선택할 수 있습니다. 다음 표에서는 Encrypt 작업을 한 번 호출하여 암호화할 수 있는 일반 텍스트의 최대 크기(바이트)를 보여 줍니다. 값은 키 사양 및 암호화 알고리즘에 따라 다릅니다. 예를 들어, 대칭 암호화 KMS 키를 사용하여 한 번에 최대 4,096바이트까지 암호화할 수 있습니다.

이러한 알고리즘에 대한 최대 일반 텍스트 길이(바이트)를 계산하려면 다음 공식을 사용하세요. (*# ## (##)* / 8) - (2 * *## ##(##)*/8) - 2. 예를 들어 SHA-256을 사용하는 RSA_2048의 경우 바이트 단위의 최대 일반 텍스트 크기는 (2048/8) - (2 * 256/8) - 2 = 190입니다.

암호화 작업의 최대 일반 텍스트 크기(바이트)

	암호화 알고리즘	
키 사양	RSAES_OAEP_SHA_1	RSAES_OAEP_SHA_256
RSA_2048	214	190

	암호화 알고리즘	
키 사양	RSAES_OAEP_SHA_1	RSAES_OAEP_SHA_256
RSA_3072	342	318
RSA_4096	470	446

서명 및 확인을 위한 RSA 키 사양

서명 및 확인에 RSA 비대칭 KMS 키를 사용하는 경우 프라이빗 키를 사용하여 메시지에 대한 서명을 생성하고 퍼블릭 키를 사용하여 서명을 확인합니다.

비대칭 KMS 키에 AWS KMS 대해에서 `Sign` 작업을 호출하면는 RSA 키 페어의 프라이빗 키, 메시지 및 지정한 서명 알고리즘을 AWS KMS 사용하여 서명을 생성합니다. 서명을 확인하려면 확인 작업을 호출합니다. 서명을 지정하고 동일한 KMS 키, 메시지 및 서명 알고리즘을 지정합니다. AWS KMS 그런 다음 RSA 키 페어의 퍼블릭 키를 사용하여 서명을 확인합니다. 퍼블릭 키를 다운로드하여 외부에서 서명을 확인하는 데 사용할 수도 있습니다 AWS KMS.

AWS KMS 는 RSA 키 사양이 있는 모든 KMS 키에 대해 다음과 같은 서명 알고리즘을 지원합니다. 서명 및 확인 작업을 호출할 때 서명 알고리즘을 지정해야 합니다. 각 요청마다 다른 알고리즘을 선택할 수 있습니다. RSA 키 페어로 서명하는 경우에는 RSASSA-PSS 알고리즘을 사용하는 것이 선호됩니다. 기존 애플리케이션과의 호환성을 위해 RSASSA-PKCS1-V1_5 알고리즘이 포함되었습니다.

RSA 키 사양에 지원되는 서명 알고리즘

서명 알고리즘	알고리즘 설명
RSASSA_PSS_SHA_256	PKCS #1 v2.2, Section 8.1(256비트 솔트와 함께 메시지 다이제스트 및 MGF1 마스크 생성 기능 모두에 SHA-256을 사용하는 PSS 패딩 포함 RSA 서명)
RSASSA_PSS_SHA_384	PKCS #1 v2.2, Section 8.1(384비트 솔트와 함께 메시지 다이제스트 및 MGF1 마스크 생성 기능 모두에 SHA-384를 사용하는 PSS 패딩 포함 RSA 서명)

서명 알고리즘	알고리즘 설명
RSASSA_PSS_SHA_512	PKCS #1 v2.2, Section 8.1(512비트 솔트와 함께 메시지 다이제스트 및 MGF1 마스크 생성 기능 모두에 SHA-512를 사용하는 PSS 패딩 포함 RSA 서명)
RSASSA_PKCS1_V1_5_SHA_256	PKCS #1 v2.2, Section 8.2(PKCS #1v1.5 패딩 및 SHA-256을 사용하는 RSA 서명)
RSASSA_PKCS1_V1_5_SHA_384	PKCS #1 v2.2, Section 8.2(PKCS #1v1.5 패딩 및 SHA-384를 사용하는 RSA 서명)
RSASSA_PKCS1_V1_5_SHA_512	PKCS #1 v2.2, Section 8.2(PKCS #1v1.5 패딩 및 SHA-512를 사용하는 RSA 서명)

특정 서명 알고리즘을 사용하도록 KMS 키를 구성할 수는 없습니다. 그러나 kms:SigningAlgorithm 정책 조건을 사용하여 보안 주체가 KMS 키와 함께 사용할 수 있는 서명 알고리즘을 지정할 수 있습니다.

KMS 키의 서명 알고리즘을 가져오려면 AWS KMS 콘솔에서 또는 DescribeKey 작업을 사용하여 KMS 키의 암호화 구성을 확인합니다. AWS KMS 는 AWS KMS 콘솔에서 또는 GetPublicKey 작업을 사용하여 퍼블릭 키를 다운로드할 때 키 사양 및 서명 알고리즘도 제공합니다.

타원 곡선 키 사양

타원 곡선(ECC) 키 사양을 사용하는 경우는 서명 및 확인 또는 공유 보안 암호 도출(둘 다 아님)을 위해 ECC 키 페어가 있는 비대칭 KMS 키를 AWS KMS 생성합니다. 서명을 생성하거나 공유 비밀을 도출하는 프라이빗 키는 절대로 암호화되지 않은 상태로 AWS KMS 를 떠나지 않습니다. 퍼블릭 키를 사용하여 내부 서명을 확인하거나 외부에서 사용할 퍼블릭 키를 AWS KMS다운로드할 수 있습니다 AWS KMS. ???

Edwards Curve 키 사양을 사용하는 경우는 서명 및 확인을 위해 Ed25519 키 페어가 있는 비대칭 KMS 키를 AWS KMS 생성합니다. 서명을 생성하는 프라이빗 키는 암호화 AWS KMS 되지 않은 상태로 두지 않습니다. 퍼블릭 키를 사용하여 내에서 서명을 확인하거나 외부에서 사용할 퍼블릭 키를 AWS KMS다운로드할 수 있습니다 AWS KMS. ???

AWS KMS 는 비대칭 KMS 키에 대해 다음과 같은 ECC 키 사양을 지원합니다.

- 비대칭 NIST 표준 타원 곡선 키 페어(서명 및 확인 또는 공유 보안 암호 도출)
  - ECC_NIST_P256(secp256r1)
  - ECC_NIST_P384(secp384r1)
  - ECC_NIST_P521(secp521r1)
  - ECC_NIST_EDWARDS25519(ed25519) - 서명 및 확인 전용
    - 참고: ECC_NIST_EDWARDS25519 KMS 키의 경우 ED25519_SHA_512 서명 알고리즘에는가 필요한 반면`MessageType:RAW`, ED25519_PH_SHA_512에는가 필요합니다`MessageType:DIGEST`. 이러한 메시지 유형은 서로 바꿔서 사용할 수 없습니다.
- 기타 비대칭 타원 곡선 키 페어(서명 및 확인)
  - ECC_SECG_P256K1(secp256k1), 일반적으로 암호 화폐에 사용됨.

선택하는 ECC 키 사양은 보안 표준 또는 작업 요구 사항에 따라 결정될 수 있습니다. 일반적으로 작업에 실용적이고 저렴한 곡선 중에서 가장 포인트가 많은 곡선을 사용하세요.

공유 보안 암호를 추출하기 위해 비대칭 KMS 키를 생성하는 경우 NIST 표준 타원 곡선 키 사양 중 하나를 사용합니다(ECC_SECG_P256K1 및 ECC_NIST_EDWARDS25519 제외). 공유 보안 비밀을 도출하는 데 지원되는 유일한 키 계약 알고리즘은 ECDH(Elliptic Curve Cryptography Cofactor Diffie-Hellman Primitive)입니다. 공유 보안 비밀을 오프라인으로 도출하는 방법의 예는 the section called “공유 보안 비밀을 오프라인으로 도출” 단원을 참조하세요.

암호 화폐에 사용할 비대칭 KMS 키를 생성하는 경우 ECC_SECG_P256K1 키 사양을 사용하세요. 이 키 사양은 다른 용도로도 사용할 수 있지만 비트코인 및 기타 암호 화폐에는 필수입니다.

다음 표에는 각 ECC 키 사양에 대해가 AWS KMS 지원하는 서명 알고리즘이 나와 있습니다. 특정 서명 알고리즘을 사용하도록 KMS 키를 구성할 수는 없습니다. 그러나 kms:SigningAlgorithm 정책 조건을 사용하여 보안 주체가 KMS 키와 함께 사용할 수 있는 서명 알고리즘을 지정할 수 있습니다.

ECC 키 사양에 지원되는 서명 알고리즘

키 사양	서명 알고리즘	알고리즘 설명
ECC_NIST_P256	ECDSA_SHA_256	NIST FIPS 186-4, Section 6.4(메시지 다이제스트를 위해 키 및 SHA-256에 의해 지정된 곡선을 사용하는 ECDSA 서명.)

키 사양	서명 알고리즘	알고리즘 설명
ECC_NIST_P384	ECDSA_SHA_384	NIST FIPS 186-4, Section 6.4(메시지 다이제스트를 위해 키 및 SHA-384에 의해 지정된 곡선을 사용하는 ECDSA 서명.)
ECC_NIST_P521	ECDSA_SHA_512	NIST FIPS 186-4, Section 6.4(메시지 다이제스트를 위해 키 및 SHA-512에 의해 지정된 곡선을 사용하는 ECDSA 서명.)
ECC_SECG_P256K1	ECDSA_SHA_256	NIST FIPS 186-4, Section 6.4(메시지 다이제스트를 위해 키 및 SHA-256에 의해 지정된 곡선을 사용하는 ECDSA 서명.)
ECC_NIST_EDWARDS25519	ED25519_SHA_512	NIST FIPS 186-5, 섹션 7.6, 메시지 다이제스트에 대해 키 및 SHA-512로 지정된 곡선을 사용하는 EdDSA 서명. 이 알고리즘을 사용하려면 KMS`MessageType:RAW`에가 필요합니다.

키 사양	서명 알고리즘	알고리즘 설명
ECC_NIST_EDWARDS25519	ED25519_PH_SHA_512	메시지 다이제스트에 대해 키 및 SHA-512로 지정된 곡선을 사용하는 NIST FIPS 186-5, 섹션 7.8, HashEdDSA 서명. 이 알고리즘을 사용하려면 KMS`MessageType:DIGEST`에가 필요합니다. 를 사용하여 ED25519_PH_SHA_512 서명 알고리즘을 지정하면는 섹션 7.8.1의 1단계에 설명된 SHA-512 접두사를 `MessageType:DIGEST` AWS KMS 수행합니다. 즉, 입력은 사용자가 한 번, KMS가 한 번 두 번 해시됩니다.

HMAC KMS 키의 키 사양

AWS KMS 는 다양한 길이의 대칭 HMAC 키를 지원합니다. 선택하는 키 사양은 보안, 규정 또는 비즈니스 요구 사항에 따라 달라질 수 있습니다. 키의 길이에 따라 GenerateMac 및 VerifyMac 작업에 사용되는 MAC 알고리즘이 결정됩니다. 일반적으로 더 긴 키는 더 안전합니다. 사용 사례에 맞는 가장 긴 키를 사용합니다.

HMAC 키 사양	MAC 알고리즘
HMAC_224	HMAC_SHA_224
HMAC_256	HMAC_SHA_256
HMAC_384	HMAC_SHA_384
HMAC_512	HMAC_SHA_512

ML-DSA 키 사양

ML-DSA 키는 양자 내성 암호를 위해 설계된 ML-DSA(모듈-격자 기반 디지털 서명 알고리즘)에서 사용되는 암호화 키입니다. 이 알고리즘은 미국 국립표준기술원(NIST)의 표준화 작업의 일부이며, 연방정보처리표준(FIPS) 204에 구체적으로 명시되어 있습니다.

ML-DSA 키는 퍼블릭-프라이빗 키 페어 시스템에서 사용됩니다. 프라이빗 키는 데이터 서명에 사용되고, 퍼블릭 키는 서명 검증에 사용됩니다. 이 시스템은 잠재적인 양자 컴퓨터 위협에도 불구하고 디지털 메시지 또는 문서의 진정성, 무결성, 부인 방지를 보장합니다.

ML-DSA 키 사양으로 키를 생성하면는 서명 및 확인을 위해 ML-DSA 키 페어가 있는 비대칭 KMS 키를 AWS KMS 생성합니다. 서명을 생성하는 프라이빗 키는 암호화 AWS KMS 되지 않은 상태로 두지 않습니다. 퍼블릭 키를 사용하여 내부 서명을 확인하거나 외부에서 사용할 퍼블릭 키를 AWS KMS다운로드할 수 있습니다 AWS KMS. https://docs.aws.amazon.com/kms/latest/developerguide/importing-keys-get-public-key-and-token.html

AWS KMS 는 비대칭 KMS 키에 대해 다음과 같은 ML-DSA 키 사양을 지원합니다.

- ML_DSA_44
- ML_DSA_65
- ML_DSA_87

AWS KMS 는 모든 ML-DSA 키 사양에 대해 ML_DSA_SHAKE_256 서명 알고리즘을 지원합니다.

SM2 키 사양(중국 리전 전용)

SM2 키 사양은 China's Office of State Commercial Cryptography Administration(OSCCA)에서 공개한 GM/T 사양 시리즈 내에 정의된 타원형 커브 키 사양입니다. SM2 키 사양은 중국 리전에서만 사용할 수 있습니다. SM2 키 사양을 사용하는 경우는 SM2 키 페어가 있는 비대칭 KMS 키를 AWS KMS 생성합니다. 내에서 SM2 키 페어를 사용하거나 외부에서 사용할 퍼블릭 키를 AWS KMS다운로드할 수 있습니다 AWS KMS. 자세한 내용은 the section called “SM2 키 페어를 사용한 오프라인 인증(중국 리전 전용)” 단원을 참조하십시오.

각 KMS 키에는 하나의 Key usage만 있을 수 있습니다. SM2 KMS 키는 서명 및 확인, 암호화 및 복호화 또는 공유 비밀 도출에 사용할 수 있습니다. KMS 키를 생성할 때 키 사용을 지정해야 하고, 키를 만든 후에는 변경할 수 없습니다.

공유 보안 비밀을 도출하기 위해 비대칭 KMS 키를 생성하는 경우 SM2 키 사양을 사용합니다. 공유 보안 비밀을 도출하는 데 지원되는 유일한 키 계약 알고리즘은 ECDH(Elliptic Curve Cryptography Cofactor Diffie-Hellman Primitive)입니다.

AWS KMS 는 다음과 같은 SM2 암호화 및 서명 알고리즘을 지원합니다.

- SM2PKE 암호화 알고리즘

  SM2PKE는 OSCCA가 GM/T 0003.4-2012에서 정의한 타원형 커브 기반 암호화 알고리즘입니다.
- SM2DSA 서명 알고리즘

  SM2DSA는 OSCCA가 GM/T 0003.2-2012에서 정의한 타원형 커브 기반 서명 알고리즘입니다. SM2DSA에는 SM3 해싱 알고리즘으로 해시된 다음 전달한 메시지 또는 메시지 다이제스트와 결합된 구분 ID가 필요합니다 AWS KMS. 그런 다음이 연결 값은 해시되고 서명됩니다 AWS KMS.

AWS KMS 권한

이 표는 AWS KMS 리소스에 대한 액세스를 제어할 수 있도록 AWS KMS 권한을 이해하는 데 도움이 되도록 설계되었습니다. 열 머리글의 정의가 테이블 아래에 나와 있습니다.

서비스 승인 참조의 주제에 대한 작업, 리소스 및 조건 키에서 AWS KMS 권한에 대해 알아볼 수도 있습니다. AWS Key Management Service 그러나 각 권한을 세부적으로 설정할 때 사용할 수 있는 모든 조건 키가 해당 주제에 나열되어 있지는 않습니다.

대칭 암호화 KMS 키, 비대칭 KMS 키 및 HMAC KMS 키에 유효한 AWS KMS 작업에 대한 자세한 내용은 섹션을 참조하세요키 유형 참조.

> **Note**
>
> 테이블의 모든 데이터를 보려면 가로 또는 세로로 스크롤해야 할 수 있습니다.

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
CancelKeyDeletion	키 정책	아니요	KMS 키	KMS 키 작업에 대한 조건: kms:CallerAccount

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
kms:CancelKeyDeletion				kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService
ConnectCustomKeyStore kms:ConnectCustomKeyStore	IAM 정책	아니요	*	kms:CallerAccount

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
CreateAlias kms:CreateAlias	IAM 정책 (별칭의 경우)	아니요	Alias	없음(별칭에 대한 액세스 제어 시)
이 연산을 사용하려면 호출자가 두 리소스에 대한 kms:CreateAlias 권한을 가지고 있어야 합니다. • 별칭(IAM 정책에서) • KMS 키 (키 정책에서) 자세한 내용은 별칭에 대한 액세스 제어 섹션을 참조하십시오.	키 정책 (KMS 키의 경우)	아니요	KMS 키	KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService
CreateCustomKeyStore kms:CreateCustomKeyStore	IAM 정책	아니요	*	kms:CallerAccount

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
CreateGrant kms:CreateGrant	키 정책	예	KMS 키	암호화 컨텍스트 조건: kms:EncryptionContext:context-key kms:EncryptionContextKeys 권한 부여 조건: kms:GrantConstraintSourceArn kms:GrantConstraintType kms:GranteePrincipal kms:GranteeServicePrincipal kms:GrantIsForAWSResource kms:GrantOperations kms:RetiringPrincipal kms:RetiringServicePrincipal KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
				kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
CreateKey kms:CreateKey	IAM 정책	아니요	*	kms:BypassPolicyLockoutSafetyCheck kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ViaService aws:RequestTag/tag-key(AWS 글로벌 조건 키) aws:ResourceTag/tag-key(AWS 글로벌 조건 키) aws:TagKeys(AWS 글로벌 조건 키)

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
Decrypt kms:Decrypt	키 정책	예	KMS 키	암호화 작업에 대한 조건 kms:EncryptionAlgorithm kms:RequestAlias 암호화 컨텍스트 조건: kms:EncryptionContext:context-key kms:EncryptionContextKeys KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
DeleteAlias kms:DeleteAlias	IAM 정책 (별칭의 경우)	아니요	Alias	없음(별칭에 대한 액세스 제어 시)
이 연산을 사용하려면 호출자가 두 리소스에 대한 kms:DeleteAlias 권한을 가지고 있어야 합니다. • 별칭(IAM 정책에서) • KMS 키 (키 정책에서) 자세한 내용은 별칭에 대한 액세스 제어 섹션을 참조하십시오.	키 정책 (KMS 키의 경우)	아니요	KMS 키	KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService
DeleteCustomKeyStore kms:DeleteCustomKeyStore	IAM 정책	아니요	*	kms:CallerAccount

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
DeleteImportedKeyMaterial `kms:DeleteImportedKeyMaterial`	키 정책	아니요	KMS 키	KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
DeriveSharedSecret kms:DeriveSharedSecret	키 정책	예	KMS 키	KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService 암호화 작업에 대한 조건: kms:KeyAgreementAlgorithm
DescribeCustomKeyStores kms:DescribeCustomKeyStores	IAM 정책	아니요	*	kms:CallerAccount

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
DescribeKey kms:DescribeKey	키 정책	예	KMS 키	KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService 기타 조건: kms:RequestAlias

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
DisableKey kms:DisableKey	키 정책	아니요	KMS 키	KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService 키 마지막 사용 조건: kms:TrailingDaysWithoutKeyUsage

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
DisableKeyRotation `kms:DisableKeyRotation`	키 정책	아니요	KMS 키	KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService
DisconnectCustomKeyStore `kms:DisconnectCustomKeyStore`	IAM 정책	아니요	*	kms:CallerAccount

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
EnableKey kms:EnableKey	키 정책	아니요	KMS 키	KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
EnableKeyRotation `kms:EnableKeyRotation`	키 정책	아니요	KMS 키	KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService 자동 키 교체 조건: kms:RotationPeriodInDays

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
암호화 `kms:Encrypt`	키 정책	예	KMS 키	암호화 작업에 대한 조건 kms:EncryptionAlgorithm kms:RequestAlias 암호화 컨텍스트 조건: kms:EncryptionContext:context-key kms:EncryptionContextKeys KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
GenerateDataKey kms:GenerateDataKey	키 정책	예	KMS 키	암호화 작업에 대한 조건 kms:EncryptionAlgorithm kms:RequestAlias 암호화 컨텍스트 조건: kms:EncryptionContext:context-key kms:EncryptionContextKeys KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
GenerateDataKeyPair `kms:GenerateDataKeyPair`	키 정책	예	KMS 키 대칭 암호화 KMS 키로 보호되는 비대칭 데이터 키 페어를 생성합니다.	데이터 키 페어의 조건: kms:DataKeyPairSpec 암호화 작업에 대한 조건 kms:EncryptionAlgorithm kms:RequestAlias 암호화 컨텍스트 조건: kms:EncryptionContext:context-key kms:EncryptionContextKeys KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
GenerateDataKeyPairWithoutPlaintext kms:GenerateDataKeyPairWithoutPlaintext	키 정책	예	KMS 키 대칭 암호화 KMS 키로 보호되는 비대칭 데이터 키 페어를 생성합니다.	데이터 키 페어의 조건: kms:DataKeyPairSpec 암호화 작업에 대한 조건 kms:EncryptionAlgorithm kms:RequestAlias 암호화 컨텍스트 조건: kms:EncryptionContext:context-key kms:EncryptionContextKeys KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
GenerateDataKeyWithoutPlaintext `kms:GenerateDataKeyWithoutPlaintext`	키 정책	예	KMS 키	암호화 작업에 대한 조건 kms:EncryptionAlgorithm kms:RequestAlias 암호화 컨텍스트 조건: kms:EncryptionContext:context-key kms:EncryptionContextKeys KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
GenerateMac kms:GenerateMac	키 정책	예	KMS 키	KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService 암호화 작업에 대한 조건: kms:MacAlgorithm kms:RequestAlias
GenerateRandom kms:GenerateRandom	IAM 정책	해당 사항 없음	*	없음

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
GetKeyLastUsage kms:GetKeyLastUsage	키 정책	아니요	KMS 키	KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
GetKeyPolicy kms:GetKeyPolicy	키 정책	아니요	KMS 키	KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
GetKeyRotationStatus `kms:GetKeyRotationStatus`	키 정책	예	KMS 키	KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
GetParametersForImport kms:GetParametersForImport	키 정책	아니요	KMS 키	kms:WrappingAlgorithm kms:WrappingKeySpec KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
GetPublicKey kms:GetPublicKey	키 정책	예	KMS 키	KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService 기타 조건: kms:RequestAlias

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
ImportKeyMaterial kms:ImportKeyMaterial	키 정책	아니요	KMS 키	KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService 기타 조건: kms:ExpirationModel kms:ValidTo
ListAliases kms:ListAliases	IAM 정책	아니요	*	없음

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
ListGrants kms:ListGrants	키 정책	예	KMS 키	KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService 기타 조건: kms:GrantIsForAWSResource

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
ListKeyPolicies kms:ListKeyPolicies	키 정책	아니요	KMS 키	KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
ListKeyRotations kms:ListKeyRotations	키 정책	아니요	KMS 키	KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService
ListKeys kms:ListKeys	IAM 정책	아니요	*	없음

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
ListResourceTags kms:ListResourceTags	키 정책	아니요	KMS 키	KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService
ListRetirableGrants kms:ListRetirableGrants	IAM 정책	지정된 보안 주체는 로컬 계정에 있어야 하지만 작업은 모든 계정에서 권한을 반환합니다.	*	없음

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
PutKeyPolicy kms:PutKeyPolicy	키 정책	아니요	KMS 키	KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService 기타 조건: kms:BypassPolicyLockoutSafetyCheck

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
ReEncrypt kms:ReEncryptFrom kms:ReEncryptTo 이 연산을 사용하려면 호출자가 두 KMS 키에 대한 권한을 가지고 있어야 합니다. • 해독에 사용되는 KMS 키의 kms:ReEncryptFrom • 암호화에 사용되는 KMS 키의 kms:ReEncryptTo	키 정책	예	KMS 키	암호화 작업에 대한 조건 kms:EncryptionAlgorithm kms:RequestAlias 암호화 컨텍스트 조건: kms:EncryptionContext:context-key kms:EncryptionContextKeys KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService 기타 조건:

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
				kms:ReEncryptOnSameKey
ReplicateKey kms:ReplicateKey 이 작업을 사용하려면 호출자에게 다음 권한이 필요합니다. • 다중 리전 기본 키의 kms:ReplicateKey • 복제본 리전의 IAM 정책에서 kms:CreateKey	키 정책	아니요	KMS 키	KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService 기타 조건: kms:ReplicaRegion

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
RetireGrant `kms:RetireGrant` 권한 부여를 사용 중지할 권한은 주로 권한 부여에 의해 결정됩니다. 정책만으로는 이 작업에 대한 액세스를 허용할 수 없습니다. 자세한 내용은 권한 부여 사용 중지 및 취소 단원을 참조하십시오.	IAM 정책 (이 권한은 키 정책에서 유효하지 않습니다.)	예	KMS 키	암호화 컨텍스트 조건: kms:EncryptionContext:context-key kms:EncryptionContextKeys 권한 부여 조건: kms:GrantConstraintType KMS 키 작업에 대한 조건: kms:CallerAccount kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
RevokeGrant `kms:RevokeGrant`	키 정책	예	KMS 키	KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService 기타 조건: kms:GrantIsForAWSResource

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
RotateKeyOnDemand kms:RotateKeyOnDemand	키 정책	아니요	KMS 키	KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
ScheduleKeyDeletion kms:ScheduleKeyDeletion	키 정책	아니요	KMS 키	KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService 키 마지막 사용 조건: kms:TrailingDaysWithoutKeyUsage

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
Sign kms:Sign	키 정책	예	KMS 키	서명 및 확인을 위한 조건: kms:MessageType kms:RequestAlias kms:SigningAlgorithm KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
TagResource kms:TagResource	키 정책	아니요	KMS 키	KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService 태그 지정 조건: aws:RequestTag/tag-key(AWS 글로벌 조건 키) aws:TagKeys(AWS 글로벌 조건 키)

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
UntagResource kms:UntagResource	키 정책	아니요	KMS 키	KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService 태그 지정 조건: aws:RequestTag/tag-key(AWS 글로벌 조건 키) aws:TagKeys(AWS 글로벌 조건 키)

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
UpdateAlias kms:UpdateAlias	IAM 정책 (별칭의 경우)	아니요	Alias	없음(별칭에 대한 액세스 제어 시)
이 연산을 사용하려면 호출자가 세 가지 리소스에 대한 kms:UpdateAlias 권한을 가지고 있어야 합니다. • 별칭 • 현재 연결된 KMS 키 • 새로 연결된 KMS 키 자세한 내용은 별칭에 대한 액세스 제어 섹션을 참조하세요.	키 정책 (KMS 키의 경우)	아니요	KMS 키	KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService
UpdateCustomKeyStore kms:UpdateCustomKeyStore	IAM 정책	아니요	*	kms:CallerAccount

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
UpdateKeyDescription `kms:UpdateKeyDescription`	키 정책	아니요	KMS 키	KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
UpdatePrimaryRegion kms:UpdatePrimaryRegion 이 작업을 사용하려면 호출자가 다중 리전 기본 키(복제본 키가 됨)와 다중 리전 복제본 키(기본 키가 됨) 모두에서 kms:UpdatePrimaryRegion 권한이 필요합니다.	키 정책	아니요	KMS 키	KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService 기타 조건 kms:PrimaryRegion

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
Verify kms:Verify	키 정책	예	KMS 키	서명 및 확인을 위한 조건: kms:MessageType kms:RequestAlias kms:SigningAlgorithm KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService

작업 및 권한	정책 유형	교차 계정 사용	리소스(IAM 정책의 경우)	AWS KMS 조건 키
VerifyMac `kms:VerifyMac`	키 정책	예	KMS 키	KMS 키 작업에 대한 조건: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key(AWS 글로벌 조건 키) kms:ViaService 암호화 작업에 대한 조건: kms:MacAlgorithm kms:RequestAlias

열 설명

이 테이블의 열에는 다음 정보가 표시됩니다.

- 작업 및 권한에는 각 AWS KMS API 작업과 작업을 허용하는 권한이 나열됩니다. 정책 문의 `Action` 요소에서 작업을 지정합니다.
- 정책 유형은 키 정책 또는 IAM 정책에서 권한을 사용할 수 있는지 여부를 나타냅니다.

  키 정책은 키 정책에서 권한을 설정할 수 있음을 의미합니다. 키 정책에 정책을 활성화하는 정책문이 포함되어 있는 경우, IAM 정책에서 권한을 설정할 수 있습니다.

IAM 정책은 IAM 정책에서만 권한을 지정할 수 있음을 의미합니다.

- 교차 계정 사용은 권한 있는 사용자가 다른 AWS 계정의 리소스에 대해 수행할 수 있는 작업을 보여줍니다.

  예 값은 보안 주체가 다른 AWS 계정의 리소스에 대해 작업을 수행할 수 있음을 의미합니다.

  아니요 값은 보안 주체가 자체 AWS 계정의 리소스에 대해서만 작업을 수행할 수 있음을 의미합니다.

  다른 계정의 보안 주체에 교차 계정 리소스에서 사용할 수 없는 권한을 부여하면 사용 권한이 적용되지 않습니다. 예를 들어 다른 계정의 보안 주체가 계정의 KMS 키에 대해 kms:TagResource 권한을 부여하면 계정의 KMS 키에 태그를 지정하려는 시도가 실패합니다.

- 리소스에는 권한이 적용되는 AWS KMS 리소스가 나열됩니다.는 KMS 키와 별칭이라는 두 가지 리소스 유형을 AWS KMS 지원합니다. 키 정책에서 `Resource` 요소의 값은 항상 `*`이며, 이는 키 정책에 연결된 KMS 키를 나타냅니다.

  다음 값을 사용하여 IAM 정책의 AWS KMS 리소스를 나타냅니다.

  KMS 키

  리소스가 KMS 키일 때는 키 ARN을 사용합니다. 도움말은 the section called “키 ID 및 키 ARN 찾기”를 참조하십시오.

  `arn:`*`AWS_partition_name`*`:kms:`*`AWS_Region`*`:`*`AWS_account_ID`*`:key/`*`key_ID`*

  예:

  arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab

  Alias

  리소스가 별칭인 경우 별칭 ARN을 사용합니다. 도움말은 the section called “별칭 이름 및 별칭 ARN 찾기”를 참조하십시오.

  `arn:`*`AWS_partition_name`*`:kms:`*`AWS_region`*`:`*`AWS_account_ID`*`:alias/`*`alias_name`*

  예:

  arn:aws:kms:us-west-2:111122223333:alias/ExampleAlias

***(별표)**

권한이 특정 리소스(KMS 키 또는 별칭)에 적용되지 않는 경우 별표(*)를 사용합니다.

AWS KMS 권한에 대한 IAM 정책에서 `Resource` 요소의 별표는 모든 AWS KMS 리소스(KMS 키 및 별칭)를 나타냅니다. AWS KMS 권한이 특정 KMS 키 또는 별칭에 적용되지 않는 경우 `Resource` 요소에 별표를 사용할 수도 있습니다. 예를 들어, `kms:CreateKey` 또는 `kms:ListKeys` 권한을 허용하거나 거부할 때는 `Resource` 요소를 `*`로 설정해야 합니다.

- AWS KMS 조건 키에는 작업에 대한 액세스를 제어하는 데 사용할 수 있는 AWS KMS 조건 키가 나열됩니다. 정책의 `Condition` 요소에 조건을 지정합니다. 자세한 내용은 AWS KMS 조건 키 단원을 참조하십시오. 이 열에는에서 지원되지 AWS KMS만 모든 AWS 서비스에서 지원되지 않는 AWS 전역 조건 키도 포함되어 있습니다.

AWS KMS 내부 작업

AWS Key Management Service (AWS KMS)는 클라우드에 맞게 조정된 FIPS 140-3 보안 수준 3 검증 하드웨어 보안 모듈(HSM)로 보호되는 암호화 키 및 작업을 제공합니다. AWS KMS 키 및 기능은 여러 AWS 클라우드 서비스에서 사용되며 애플리케이션의 데이터를 보호하는 데 사용할 수 있습니다. 이 기술 가이드에서는 사용 AWS 시 내에서 실행되는 암호화 작업에 대한 세부 정보를 제공합니다 AWS KMS.

AWS KMS 내부는 전 세계에 분산된 키 관리 서비스의 HSMs을 확장하고 보호하는 데 필요합니다.

주제

- 도메인 및 도메인 상태
- 내부 통신 보안
- 다중 리전 키의 복제 프로세스
- 내구성 보호

도메인 및 도메인 상태

내에서 신뢰할 수 있는 내부 AWS KMS 엔터티의 공동 컬렉션을 도메인이라고 AWS 리전 합니다. 도메인에는 신뢰할 수 있는 엔터티 세트, 규칙 세트 및 도메인 키라는 보안 키 세트가 포함됩니다. 도메인 키는 도메인의 멤버인 HSM 간에 공유됩니다. 도메인 상태는 다음 필드로 구성됩니다.

이름(Name)

이 도메인을 식별하는 도메인 이름입니다.

멤버(Members)

퍼블릭 서명 키 및 퍼블릭 계약 키를 포함하여 도메인의 멤버인 HSM의 목록입니다.

운영자(Operators)

이 서비스의 연산자를 나타내는 엔터티, 퍼블릭 서명 키 및 역할(AWS KMS 작업자 또는 서비스 호스트)의 목록입니다.

규칙(Rules)

HSM에서 명령을 실행하기 위해 충족해야 하는 각 명령의 쿼럼 규칙 목록입니다.

도메인 키

도메인 내에서 현재 사용 중인 도메인 키(대칭 키)의 목록입니다.

전체 도메인 상태는 HSM에서만 사용할 수 있습니다. 도메인 상태는 HSM 도메인 멤버 간에 내보낸 도메인 토큰으로서 동기화됩니다.

도메인 키

도메인의 모든 HSM은 $\{DK_r\}$라는 일련의 도메인 키를 공유합니다. 이러한 키는 도메인 상태 내보내기 루틴을 통해 공유됩니다. 내보낸 도메인 상태는 도메인의 구성원인 HSM으로 가져올 수 있습니다.

도메인 키 세트 $\{DK_r\}$에는 항상 활성 도메인 키 하나와 비활성화된 여러 도메인 키가 포함됩니다. 도메인 키는 키 관리 권장 사항 - 1부를 AWS 준수하도록 매일 교체됩니다. 도메인 키 교체 중에 나가는 도메인 키로 암호화된 모든 기존 KMS 키는 새 활성 도메인 키로 다시 암호화됩니다. 활성 도메인 키는 새 EKT를 암호화하는 데 사용됩니다. 만료된 도메인 키는 최근에 교체된 도메인 키 수와 동일한 일수 동안 이전에 암호화된 EKT를 복호화하는 데에만 사용할 수 있습니다.

내보낸 도메인 토큰

도메인 참가자 간에 상태를 정기적으로 동기화해야 합니다. 이 작업은 도메인이 변경될 때마다 도메인 상태를 내보냄으로써 수행됩니다. 도메인 상태는 내보낸 도메인 토큰으로서 내보내집니다.

이름(Name)

이 도메인을 식별하는 도메인 이름입니다.

멤버(Members)

서명 및 계약 퍼블릭 키를 포함하여 도메인의 멤버인 HSM의 목록입니다.

연산자

이 서비스의 운영자를 나타내는 엔터티, 퍼블릭 서명 키 및 역할의 목록입니다.

규칙(Rules)

HSM 도메인 멤버에서 명령을 실행하기 위해 충족해야 하는 각 명령의 쿼럼 규칙 목록입니다.

암호화된 도메인 키(Encrypted domain keys)

봉투 암호화된 도메인 키입니다. 도메인 키는 위에 나열된 각 멤버의 서명 멤버에 의해 암호화되고 퍼블릭 계약 키로 봉투 처리됩니다.

서명(Signature)

HSM에서 생성한 도메인 상태에 대한 서명으로, 도메인 상태를 내보낸 도메인의 멤버여야 합니다.

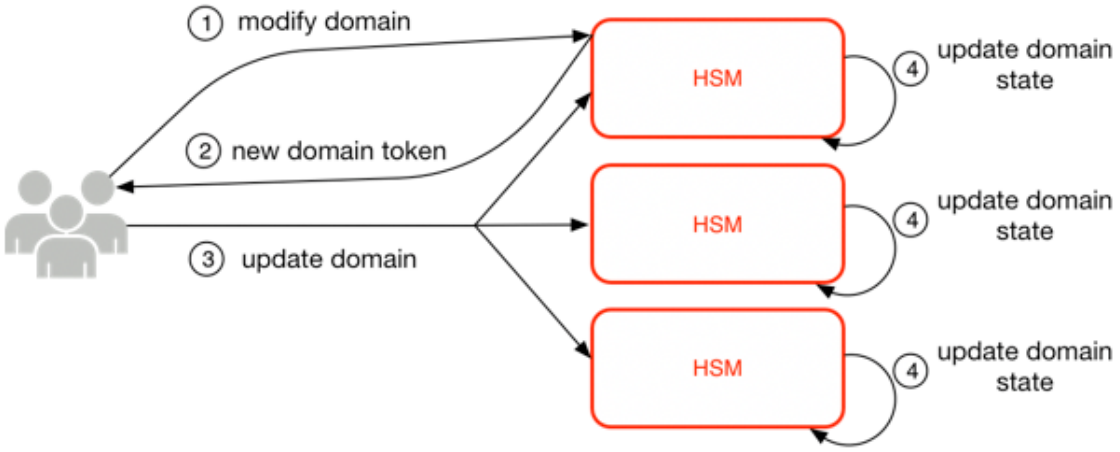
내보낸 도메인 토큰은 도메인 내에서 작동하는 엔터티에 대한 기본 트러스트 소스를 형성합니다.

도메인 상태 관리

도메인 상태는 쿼럼 인증 명령을 통해 관리됩니다. 이러한 변경 사항에는 도메인의 신뢰할 수 있는 참가자 목록 수정, HSM 명령 실행에 대한 쿼럼 규칙 수정, 도메인 키 교체 등이 포함됩니다. 다음 이미지에 표시된 것과 같이 이러한 명령은 인증된 세션 작업과 달리 명령별로 인증됩니다.

초기화되고 정상 작동 상태인 HSM에는 자체 생성된 비대칭 자격 증명 키 세트, 서명 키 페어 및 키 설정 키 페어가 포함됩니다. 수동 프로세스를 통해 AWS KMS 운영자는 리전의 첫 번째 HSM에서 생성할 초기 도메인을 설정할 수 있습니다. 이 초기 도메인은 이 주제에서 이전에 정의한 전체 도메인 상태로 구성됩니다. 조인 명령을 통해 도메인의 정의된 각 HSM 멤버에 설치됩니다.

HSM이 초기 도메인에 조인한 후에는 해당 도메인에 정의된 규칙에 바인딩됩니다. 이러한 규칙은 고객 암호화 키를 사용하는 명령이나 호스트 또는 도메인 상태를 변경하는 명령을 제어합니다. 암호화 키를 사용하는 인증된 세션 API 작업이 이전에 정의되었습니다.



위의 이미지는 도메인 상태가 수정되는 방법을 보여줍니다. 프로세스는 다음 네 단계로 구성됩니다.

1. 도메인을 수정하는 쿼럼 기반 명령이 HSM으로 전송됩니다.
2. 새 도메인 상태가 생성되고 새로 내보낸 도메인 토큰으로 내보내집니다. HSM의 상태는 수정되지 않습니다. 즉, 변경 사항이 HSM에 적용되지 않습니다.
3. 두 번째 명령은 새로 내보낸 도메인 토큰의 각 HSM으로 전송되어 도메인 상태를 새 도메인 토큰으로 업데이트합니다.
4. 내보낸 새 도메인 토큰에 나열된 HSM은 명령과 도메인 토큰을 인증할 수 있습니다. 또한 도메인 키의 압축을 풀고 도메인의 모든 HSM에서 도메인 상태를 업데이트할 수도 있습니다.

HSM은 서로 직접 통신하지 않습니다. 대신 운영자 쿼럼이 도메인 상태에 대한 변경을 요청하여 새로 내보낸 도메인 토큰이 생성됩니다. 도메인의 서비스 호스트 멤버는 도메인의 모든 HSM에 새 도메인 상태를 배포하는 데 사용됩니다.

도메인의 탈퇴 및 조인은 HSM 관리 함수를 통해 수행됩니다. 도메인 상태의 수정은 도메인 관리 함수를 통해 수행됩니다.

도메인 탈퇴

HSM이 도메인에서 탈퇴하게 되어 해당 도메인의 나머지 모든 부분과 키가 메모리에서 삭제됩니다.

도메인 조인

HSM이 새 도메인에 조인되거나 현재 도메인 상태가 새 도메인 상태로 업데이트됩니다. 이 메시지를 인증하기 위한 초기 규칙 세트의 소스로 기존 도메인이 사용됩니다.

도메인 생성

HSM에 새 도메인을 만듭니다. 도메인의 멤버 HSM에 배포할 수 있는 첫 번째 도메인 토큰을 반환합니다.

운영자 수정

도메인의 인증된 운영자 및 해당 역할의 목록에서 운영자를 추가하거나 제거합니다.

멤버 수정

도메인의 인증된 HSM 목록에서 HSM을 추가하거나 제거합니다.

규칙 수정

HSM에서 명령을 실행하는 데 필요한 쿼럼 규칙 세트를 수정합니다.

도메인 키 교체

새 도메인 키를 만들어 활성 도메인 키로 표시합니다. 이렇게 하면 기존 활성 키가 비활성화된 키로 이동하고 도메인 상태에서 가장 오래된 비활성화된 키가 제거됩니다.

내부 통신 보안

서비스 호스트 또는 AWS KMS 연산자와 HSMs 간의 명령은에 설명된 두 가지 메커니즘, 인증된 세션즉 쿼럼 서명 요청 방법과 HSM 서비스 호스트 프로토콜을 사용하는 인증된 세션을 통해 보호됩니다.

쿼럼 서명 명령은 단일 운영자가 HSM이 제공하는 중요한 보안 보호 기능을 수정할 수 없도록 설계되었습니다. 인증된 세션을 통해 실행되는 명령을 사용하면 인증된 서비스 운영자만 KMS 키와 관련된 작업을 수행하도록 할 수 있습니다. 모든 고객 기반 보안 암호 정보는 AWS 인프라 전체에서 보호됩니다.

키 설정

내부 통신을 보호하기 위해는 서로 다른 두 가지 키 설정 방법을 AWS KMS 사용합니다. 첫 번째는 이산 로그 암호화를 사용한 페어 방식 키 설정 체계에 대한 권장 사항에 C(1, 2, ECC DH)로 정의되어 있습니다. 이 체계에는 정적 서명 키가 있는 초기자가 있습니다. 이 초기자는 정적 ECDH 계약 키가 있는 수신자를 위한 임시 타원 곡선 디Diffie-Hellman(ECDH) 키를 생성하고 서명합니다. 이 방법은 ECDH를 사용하는 하나의 임시 키와 2개의 정적 키를 사용합니다. 이는 레이블 C(1, 2, ECC DH)의 파생입니다. 이 방법을 원 패스 ECDH라고도 합니다.

두 번째 키 설정 방법은 C(2, 2, ECC, DH)입니다. 이 방식에서 양 당사자는 정적 서명 키를 가지며 임시 ECDH 키를 생성, 서명 및 교환합니다. 이 방법은 각각 ECDH를 사용하는 2개의 정적 키와 2개의 임시

키를 사용합니다. 이는 레이블 C(2, 2, ECC, DH)의 파생입니다. 이 방법은 ECDH 임시 또는 ECDHE라고도 합니다. 모든 ECDH 키는 secp384r1(NIST-P384) 곡선에 생성됩니다.

HSM 보안 경계

의 내부 보안 경계는 HSM AWS KMS 입니다. HSM은 독점적 인터페이스를 제공하며 작동 상태일 때 다른 활성 물리적 인터페이스는 없습니다. 작동 HSM은 초기화 중에 도메인에서 역할을 설정하는 데 필요한 암호화 키를 사용하여 프로비저닝됩니다. HSM의 민감한 암호화 구성 요소는 휘발성 메모리에만 저장되며, HSM이 의도하거나 의도하지 않은 종료 또는 재설정을 비롯한 작동 상태를 벗어날 때 지워집니다.

HSM API 작업은 개별 명령에 의해 또는 서비스 호스트가 설정한 상호 인증된 기밀 세션을 통해 인증됩니다.

쿼럼 서명 명령

Quorum-signed 명령은 운영자가 HSM에 대해 실행합니다. 이 섹션에서는 쿼럼 기반 명령을 만들고 서명하고 인증하는 방법에 대해 설명합니다. 이 규칙은 매우 간단합니다. 예를 들어 Foo 명령의 경우 Bar 역할의 두 멤버가 인증되어야 합니다. 쿼럼 기반 명령의 생성 및 검증은 세 단계로 이루어집니다. 첫 번째 단계는 초기 명령을 생성하는 것이고, 두 번째 단계는 서명할 추가 운영자에게 제출하는 것이며, 세 번째 단계는 검증하고 실행하는 것입니다.

이 개념을 도입하기 위해 운영자의 퍼블릭 키 및 역할 세트가 $\{QOS_s\}$이고, 쿼럼 규칙 세트가 QR = $\{Command_i, Rule_{\{i,\ t\}}\}$(여기서 각 Rule은 역할의 세트)이며, 최소 개수가 N $\{Role_t, N_t\}$이라고 가정합니다. 명령이 이 쿼럼 규칙을 충족하려면 해당 명령에 대해 나열된 규칙 중 하나를 충족하도록 $\{QOS_s\}$가에 나열된 작업 세트로 명령 데이터 집합에 서명해야 합니다. 앞서 언급했듯이 쿼럼 규칙 및 운영자 세트는 도메인 상태 및 내보낸 도메인 토큰에 저장됩니다.

실제로 초기 Signer는 $Sig_1 = Sign(dO_{p1}, Command)$ 명령에 서명합니다. 두 번째 운영자는 $Sig_2 = Sign(dO_{p2}, Command)$ 명령에 서명합니다. 이중으로 서명된 메시지는 실행을 위해 HSM으로 전송됩니다. HSM은 다음을 수행합니다.

1. 각 서명에 대해 도메인 상태에서 서명자의 퍼블릭 키를 추출하고 명령의 서명을 검증합니다.
2. Signer 세트가 명령에 대한 규칙을 충족하는지 확인합니다.

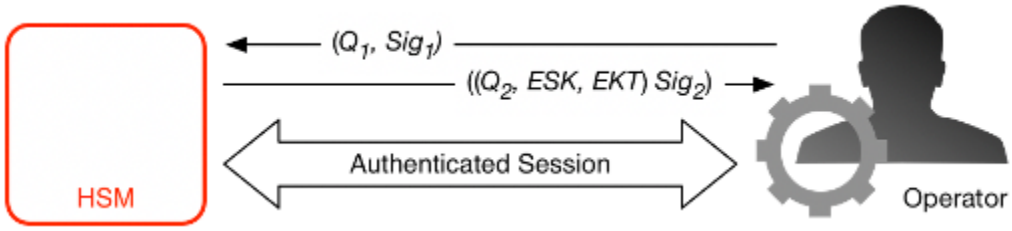
인증된 세션

키 작업은 외부 대상 AWS KMS 호스트와 HSMs. 이러한 명령은 암호화 키의 생성 및 사용, 그리고 보안 난수 생성과 관련이 있습니다. 이들 명령은 서비스 호스트와 HSM 간의 세션 인증 채널을 통해 실행됩니다. 이러한 세션에는 신뢰성 외에 기밀성도 요구됩니다. 이러한 세션을 통해 실행되는 명령에서는

일반 텍스트 데이터 키 및 복호화된 메시지가 반환됩니다. 중간자 공격을 통해 이러한 세션을 악용할 수 없도록 하기 위해 세션이 인증됩니다.

이 프로토콜은 HSM과 서비스 호스트 간에 상호 인증된 ECDHE 키 계약을 수행합니다. 교환은 서비스 호스트에 의해 시작되고 HSM에 의해 완료됩니다. 또한 HSM은 협상된 키로 암호화된 세션 키(SK)와 이 세션 키가 포함된 내보낸 키 토큰을 반환합니다. 내보낸 키 토큰에는 유효 기간이 포함되며, 이 기간이 지나면 서비스 호스트가 세션 키를 다시 협상해야 합니다.

서비스 호스트는 도메인의 멤버이며 자격 증명 서명 키 페어($dHOS_i$, $QHOS_i$)와 HSM의 자격 증명 퍼퍼블릭 키의 올바른 사본을 가지고 있습니다. 자격 증명 서명 키 세트를 사용하여 도메인의 서비스 호스트와 HSM 간에 사용할 수 있는 세션 키를 안전하게 협상합니다. 내보낸 키 토큰에는 유효 기간이 있으며 그 기간이 지나면 새 키를 협상해야 합니다.



이 프로세스는 그 자체와 도메인의 HSM 멤버 간의 중요한 통신 흐름을 보내고 받기 위해 세션 키가 필요하다는 것을 서비스 호스트가 인식하면서 시작됩니다.

1. 서비스 호스트가 ECDH 임시 키 페어 (d_1, Q_1)를 생성하고 자격 증명 키 Sig1 = Sign(dOS,Q_1)를 사용하여 서명합니다.
2. HSM이 현재 도메인 토큰을 사용하여 수신된 퍼블릭 키의 서명을 확인하고 ECDH 임시 키 페어 (d_2, Q_2)를 생성합니다. 그런 다음 이산 로그 암호화를 사용한 페어 방식 키 설정 체계에 대한 권장 사항에 따라 ECDH 키 교환을 완료하여 협상된 256비트 AES-GCM 키를 구성합니다. HSM이 새로운 256비트 AES-GCM 세션 키를 생성합니다. 협상된 키로 세션 키를 암호화하여 암호화된 세션 키(ESK)를 만듭니다. 또한 내보낸 키 토큰 EKT으로서 도메인 키를 사용하여 세션 키를 암호화합니다. 마지막으로 자격 증명 키 페어 Sig_2 = Sign(dHSK, (Q_2, ESK, EKT))를 사용하여 반환 값에 서명합니다.
3. 서비스 호스트가 현재 도메인 토큰을 사용하여 수신된 키의 서명을 확인합니다. 그런 다음 서비스 호스트는 이산 로그 암호화를 사용한 페어 방식 키 설정 체계에 대한 권장 사항에 따라 ECDH 키 교환을 완료합니다. 다음으로, ESK를 복호화하여 세션 키 SK를 가져옵니다.

EKT의 유효 기간 동안, 서비스 호스트는 협상된 세션 키 SK를 사용하여 봉투 암호화 명령을 HSM으로 보낼 수 있습니다. 이 인증된 세션을 통해 서비스 호스트가 시작한 모든 명령에는 EKT가 포함됩니다. HSM은 동일한 협상된 세션 키 SK를 사용하여 응답합니다.

다중 리전 키의 복제 프로세스

AWS KMS 는 교차 리전 복제 메커니즘을 사용하여 KMS 키의 키 구성 요소를 하나의 HSM에서 다른의 HSM AWS 리전 으로 복사합니다 AWS 리전. 이 메커니즘이 작동하려면 복제되는 KMS 키가 AWS_KMS 오리진이 있는 다중 리전 키여야 합니다. 가져온 키 구성 요소(EXTERNAL 오리진)가 있는 다중 리전 키의 경우 동일한 키 구성 요소를 각 복제본으로 개별적으로 가져와 키 구성 요소를 복제하는 것은 사용자의 책임입니다.

한 리전에서 다른 리전으로 KMS 키를 복제할 때 리전의 HSM은 격리된 네트워크에 있으므로 직접 통신할 수 없습니다. 대신 리전 간 복제 중에 교환된 메시지가 프록시 서비스를 통해 전달됩니다. 리전 간 복제 중에 AWS KMS HSM에서 생성된 모든 메시지는 복제 서명 키를 사용하여 암호화 방식으로 서명됩니다. RSK(복제 서명 키)는 NIST P-384 곡선의 ECDSA 키입니다. 모든 리전은 하나 이상의 RSK를 소유하며, 각 RSK의 퍼블릭 구성 요소는 동일한 AWS 파티션에 있는 다른 모든 리전과 공유됩니다.

리전 A에서 리전 B로 주요 구성 요소를 복사하는 리전 간 복제 프로세스는 다음과 같이 작동합니다.

1. 리전 B의 HSM은 NIST P-384 곡선의 임시 ECDH 키인 RAKB(Replication Agreement Key B)를 생성합니다. RAKB의 퍼블릭 구성 요소는 프록시 서비스를 통해 리전 A의 HSM으로 전송됩니다.
2. 리전 A의 HSM은 RAKB의 퍼블릭 구성 요소를 수신한 다음 NIST P-384 곡선의 다른 임시 ECDH 키인 RAKA(Replication Agreement Key A)를 생성합니다. HSM은 RAKA 및 RAKB의 퍼블릭 구성 요소에 ECDH 키 설정 체계를 실행하고 출력에서 대칭 키 RWK(Replication Wrapping Key)를 파생합니다. RWK는 복제되는 다중 리전 KMS 키의 키 구성 요소를 암호화하는 데 사용됩니다.
3. RAKA의 퍼블릭 구성 요소와 RWK로 암호화된 키 구성 요소는 프록시 서비스를 통해 리전 B의 HSM으로 전송됩니다.
4. 리전 B의 HSM은 RAKA의 퍼블릭 구성 요소와 RWK를 사용하여 암호화된 키 구성 요소를 수신합니다. HSM은 RAKB와 RAKA의 퍼블릭 구성 요소에 대해 ECDH 키 설정 체계를 실행하여 RWK를 파생합니다.
5. 리전 B의 HSM은 RWK를 사용하여 리전 A의 키 구성 요소를 복호화합니다.

내구성 보호

이 서비스에 의해 생성된 키의 추가 서비스 내구성은 오프라인 HSM, 내보낸 도메인 토큰의 다중 비휘발성 스토리지, 암호화된 KMS 키의 이중화 스토리지를 사용하여 제공됩니다. 오프라인 HSM은 기존 도메인의 멤버입니다. 온라인 상태가 아닌 채로 일반 도메인 작업에 참여하는 것을 제외하고, 오프라인 HSM은 도메인 상태에 기존 HSM 멤버와 동일하게 표시됩니다.

내구성 설계는 리전의 모든 KMS 키가 기본 스토리지 시스템에 저장된 온라인 HSMs 또는 KMS 키 세트가 광범위하게 손실될 AWS 수 있도록 보호하기 위한 것입니다. 가져온 키 구성 요소가 AWS KMS keys 있는는 다른 KMS 키가 제공하는 내구성 보호에 포함되지 않습니다. 리전 전체에 장애가 발생하는 경우 AWS KMS가져온 키 구성 요소를 KMS 키로 다시 가져와야 할 수 있습니다.

오프라인 HSM과 해당 HSM에 액세스하기 위한 자격 증명은 독립적인 여러 지리적 위치에 있는 모니터링되는 안전실 내의 금고에 저장됩니다. 각 금고에는 이러한 자료를 얻 AWS기 위해 두 개의 독립 팀에서 한 명 이상의 AWS 보안 책임자와 한 명의 AWS KMS 운영자가 필요합니다. 이러한 재료의 사용은 AWS KMS 연산자 쿼럼이 있어야 하는 내부 정책에 의해 관리됩니다.

문서 기록

이 주제에서는 AWS Key Management Service 개발자 가이드에 대한 중요한 업데이트 사항에 대해 설명합니다.

주제

- 최신 업데이트
- 이전 업데이트

최신 업데이트

다음 표에서는 2018년 1월 이후 이 설명서의 중요한 변경 사항을 설명합니다. Amazon은 여기 나와 있는 주요 변경 사항 외에도 설명과 예제를 업데이트하고 고객이 제공한 피드백을 반영하도록 설명서를 자주 업데이트하고 있습니다. 중요한 변경 사항에 대해 알림을 받으려면 RSS 피드를 구독합니다.

이 테이블의 모든 데이터를 보려면 가로 또는 세로로 스크롤해야 할 수도 있습니다.

변경 사항	설명	날짜
문서 업데이트	가 지원하는 암호화 알고리즘을 정확하게 나열하도록 지원되는 암호화 알고리즘 페이지를 업데이트했습니다 AWS KMS .	2026년 9월 29일
서비스 보안 주체 권한 부여	AWS KMS 는 이제 AWS 서비스 보안 주체에 대한 권한 부여 생성을 지원합니다. `CreateGrant` 작업에서 새 `GranteeServicePrincipal` 및 `RetiringServicePrincipal` 파라미터를 사용하여 서비스 보안 주체를 권한 부여의 피부여자 또는 사용 중지 보안 주체로 지정할 수 있습니다. 또한이 릴	2026년 3월 6일

	리스에는 특정 AWS 리소스를 대신하여 이루어진 요청에 대한 권한 부여 권한을 제한하는 권한 `SourceArn` 부여 제약 조건이 도입되었습니다. `CreateGrant` 요청에 사용할 수 있는 세 가지 새로운 조건 키는 `kms:GranteeServicePrincipal` , `kms:RetiringServicePrincipal` 및 입니다`kms:GrantConstraintSourceArn` . 자세한 내용은 에서 권한 부여를 참조하세요 AWS KMS.	
기능 업데이트	AWS KMS 는 가져온 키 구성 요소가 있는 대칭 암호화 다중 리전 키의 온디맨드 교체를 지원합니다.	2025년 11월 26일
기능 업데이트	AWS KMS 는 CloudTrail 이벤트의 `tlsDetails` 섹션에서 새 `keyExchange` 필드를 지원합니다.	2025년 11월 24일
기능 업데이트	AWS KMS 는 새로운 `KeySpec` ECC_NIST_EDWARDS25519 를 지원합니다.	2025년 11월 7일
듀얼 스택 지원	AWS KMS 는 듀얼 스택을 지원합니다.	2025년 6월 18일
기능 업데이트	모듈-격자 디지털 서명 알고리즘(ML-DSA) 양자 내성 암호화 서명에 대한 지원을 추가합니다.	2025년 6월 13일

가져온 키 교체	가져온 키 구성 요소(EXTERNAL 오리진)를 사용하는 대칭 암호화 KMS 키의 온디맨드 교체를 수행할 수 있습니다.	2025년 6월 5일
기능 업데이트	중국 리전에서 다중 리전 KMS 키에 대한 지원이 추가되었습니다.	2024년 11월 21일
AWS 관리형 정책 업데이트	관리형 정책 버전 v2에 문 ID(Sid)를 추가해 AWSKeyManagementServiceMultiRegionKeysServiceRolePolicy 서비스 연결 역할을 업데이트했습니다.	2024년 11월 21일
할당량 변경	ImportKeyMaterial 및 DeleteImportedKeyMaterial 요청의 기본 요청 속도를 높였습니다.	2024년 7월 23일
할당량 변경	대칭 암호화 KMS 키, RSA KMS 키, ECC 및 SM2 KMS 키에 대한 기본 암호화 작업 요청 속도를 높였습니다.	2024년 7월 8일
새로운 기능	NIST 권장 타원 곡선(ECC) 및 SM2(중국 리전만 해당) KMS 키에 대한 새로운 KeyUsage 유형 KEY_AGREEMENT 를 추가하고 공유 보안 비밀을 도출하기 위한 지원을 추가했습니다.	2024년 6월 13일

키 교체에 대한 업데이트	자동 키 교체, 온디맨드 키 교체 및 키 구성 요소 교체의 가시성을 위한 사용자 지정 교체 기간에 대한 지원을 추가했습니다.	2024년 4월 12일
관리형 정책으로 업데이트	`AWSKeyManagementServiceCustomKeyStoresServiceRolePolicy` 가 실패 시 명확한 오류 메시지를 제공할 수 있도록가 AWS CloudHSM 클러스터가 포함된 VPC의 변경 사항을 AWS KMS 모니터링할 AWS KMS 수 있는 새로운 권한을에 추가했습니다.	2023년 11월 10일
기능 업데이트	`DryRun` API 파라미터에 대한 지원이 추가되었습니다.	2023년 7월 5일
기능 업데이트	사용자 지정 키 스토어를 제외한 모든 유형의 AWS KMS 키에 대한 키 구성 요소 가져오기에 대한 지원이 추가되었습니다.	2023년 6월 5일
기능 업데이트	Nitro AWS KMS APIs 업데이트	2023년 3월 10일

기능 업데이트	RSAES_PKCS1_V1_5 래핑 알고리즘은 더 이상 사용되지 않습니다.는 미국 국립 표준 기술 연구소(NIST)의 암호화 키 관리 지침에 따라 2023년 10월 1RSAES_PKCS1_V1_5 일까지에 대한 모든 지원을 AWS KMS 종료합니다. 즉시 다른 래핑 알고리즘을 사용하는 것을 권장합니다.	2023년 2월 28일
기능 업데이트	외부의 암호화 키를 사용하여 리소스를 보호할 수 있는 기능인 외부 키 스토어에 AWS 대한 지원이 추가되었습니다 AWS.	2022년 11월 29일
할당량 변경	각 계정 및 리전에서 AWS KMS keys 리소스 할당량을 100,000개의 KMS 키로 늘렸습니다.	2022년 7월 8일
기능 업데이트	에 HMAC KMS 키에 대한 지원 추가 AWS 리전	2022년 7월 8일
새 주제	AWS KMS 개발자 안내서의 보안 장에 복원력 AWS Key Management Service 주제를 추가했습니다.	2022년 6월 14일
새로운 기능	HMAC 코드를 생성하고 확인하는 AWS KMS 키 및 API 작업에 대한 지원이 추가되었습니다.	2022년 4월 19일
문서 변경	고객 마스터 키(CMK) 라는 용어가 AWS KMS key과 KMS 키로 바뀌었습니다.	2021년 8월 30일

새로운 기능	동일한 키 ID와 키 구성 요소를 가진 서로 다른 리전의 상호 운용 가능한 KMS 키 집합인 다중 리전 키에 대한 지원이 추가되었습니다. 다중 지역 키를 사용하여 한 지역의 데이터를 암호화하고 다른 지역의 데이터를 해독할 수 있습니다.	2021년 6월 8일
새로운 기능	속성 기반 액세스 제어(ABAC)에 대한 지원을 추가했습니다. 태그와 별칭을 사용하여에 대한 액세스를 제어할 수 있습니다 AWS KMS keys.	2020년 12월 17일
새로운 기능	VPC 엔드포인트 정책에 대한 지원이 추가되었습니다.	2020년 7월 9일
새 콘텐츠	의 보안 속성을 설명합니다 AWS KMS.	2020년 6월 18일
새로운 기능	비대칭 AWS KMS keys 및 비대칭 데이터 키에 대한 지원이 추가되었습니다.	2019년 11월 25일
업데이트된 기능	AWS KMS 콘솔 AWS 관리형 키 에서의 키 정책을 볼 수 있습니다. 이전에는 이 기능을 고객 관리형 키에만 사용할 수 있었습니다.	2019년 11월 15일
새로운 기능	AWS KMS에 대한 호출을 위해 TLS에서 하이브리드포스트 퀀텀 키 교환 알고리즘을 사용하는 방법을 설명합니다.	2019년 11월 4일

할당량 변경	KMS 키를 관리하는 일부 API의 리소스 할당량을 늘렸습니다.	2019년 9월 18일
할당량 변경	KMS 키, 별칭 및 KMS 키당 권한 부여에 대한 리소스 할당량을 변경했습니다.	2019년 3월 27일
할당량 변경	사용자 지정 키 스토어에서 AWS KMS keys 를 사용하는 암호화 작업에 대한 초당 공유 요청 할당량을 변경했습니다.	2019년 3월 7일
새로운 기능	AWS KMS 사용자 지정 키 스토어를 생성하고 관리하는 방법을 설명합니다. 각 키 스토어는 사용자가 소유하고 제어하는 AWS CloudHSM 클러스터에 의해 지원됩니다.	2018년 11월 26일
새로운 콘솔	IAM AWS KMS 콘솔과 독립적인 새 콘솔을 사용하는 방법을 설명합니다. 사용자에게 새 콘솔을 익힐 시간을 주기 위해 원래 콘솔과 이를 사용하기 위한 지침이 잠시 동안 사용 가능한 상태로 유지됩니다.	2018년 11월 7일
할당량 변경	사용에 대한 공유 요청 할당량을 변경했습니다 AWS KMS keys.	2018년 8월 21일
새 콘텐츠	가 키를 AWS Secrets Manager 사용하여 보안 암호의 보안 암호 값을 암호화하는 방법을 AWS KMS 설명합니다.	2018년 7월 13일

새 콘텐츠	DynamoDB가 AWS KMS AWS KMS keys 를 사용하여 서버 측 암호화 옵션을 지원하는 방법을 설명합니다.	2018년 5월 23일
새로운 기능	인터넷을 통해 연결하는 AWS KMS대신 VPC의 프라이빗 엔드포인트를 사용하여에 직접 연결하는 방법을 설명합니다.	2018년 1월 22일

이전 업데이트

다음 표에서는 2018년 이전 AWS Key Management Service 개발자 안내서의 주요 변경 사항을 설명합니다.

이 테이블의 모든 데이터를 보려면 가로 또는 세로로 스크롤해야 할 수도 있습니다.

변경	설명	Date
새로운 내용	의 태그 AWS KMS에 대한 문서를 추가했습니다.	2017년 2월 15일
새로운 내용	모니터링 AWS KMS keys 및 Amazon CloudWatch를 사용하여 KMS 키 모니터링에 대한 문서를 추가했습니다.	2016년 8월 31일
새로운 내용	가져온 키 구성 요소에 대한 문서를 추가했습니다.	2016년 8월 11일
새로운 내용	IAM 정책, 권한 참조 및 조건 키 문서가 추가되었습니다.	2016년 7월 5일
업데이트	설명서의 KMS 키 액세스 및 권한장 일부를 업데이트되었습니다.	2016년 7월 5일

변경	설명	Date
업데이트	새로운 기본 할당량을 반영하여 할당량 페이지를 업데이트했습니다.	2016년 5월 31일
업데이트	새로운 기본 할당량을 반영하여 할당량 페이지를 업데이트하고 권한 부여 토큰 설명서를 더욱 간결하고 정확하게 업데이트했습니다.	2016년 4월 11일
새로운 내용	여러 IAM 보안 주체가 KMS 키에 액세스하도록 허용 및 IP 주소 조건 사용에 대한 문서를 추가했습니다.	2016년 2월 17일
업데이트	의 키 정책 AWS KMS 및 키 정책 변경 페이지를 더욱 간결하고 정확하게 업데이트했습니다.	2016년 2월 17일
업데이트	KMS 키 관리 주제 페이지를 더욱 간결하게 업데이트했습니다.	2016년 1월 5일
새로운 내용	CloudTrail에 대한 문서를 추가했습니다.	2015년 11월 18일
새로운 내용	키 정책 변경에 대한 지침을 추가했습니다.	2015년 11월 18일
업데이트	Amazon Relational Database Service의 사용 방식에 대한 설명서를 업데이트했습니다 AWS KMS.	2015년 11월 18일
새로운 내용	Amazon WorkSpaces에 대한 설명서를 추가했습니다.	2015년 11월 6일

변경	설명	Date
업데이트	의 키 정책 AWS KMS 페이지를 더욱 간결하게 업데이트했습니다.	2015년 10월 22일
새로운 내용	경보 만들기 및 KMS 키의 과거 사용 결정에 대한 지원 문서를 포함하여 삭제 AWS KMS key에 대한 문서를 추가했습니다.	2015년 10월 15일
새로운 내용	에 대한 액세스 결정 AWS KMS keys에 대한 문서를 추가했습니다.	2015년 10월 15일
새로운 내용	키의 AWS KMS 키 상태에 대한 문서를 추가했습니다.	2015년 10월 15일
새로운 내용	Amazon Simple Email Service에 대한 설명서를 추가했습니다.	2015년 10월 1일
업데이트	새 요청 할당량을 설명하기 위해 할당량 페이지를 업데이트했습니다.	2015년 8월 31일
새로운 내용	사용 요금에 대한 정보가 추가되었습니다 AWS KMS. AWS KMS 요금을 참조하세요.	2015년 8월 14일
새로운 내용	에 요청 할당량을 추가했습니다 AWS KMS 할당량.	2015년 6월 11일
새로운 내용	`UpdateAlias` 작업 사용을 보여주는 새로운 Java 코드 샘플을 추가했습니다.	2015년 6월 1일

변경	설명	Date
업데이트	AWS Key Management Service 리전 표를 AWS 일반 참조로 옮겼습니다.	2015년 5월 29일
새로운 내용	Amazon EMR의 사용 방식 AWS KMS에 대한 문서를 추가했습니다.	2015년 1월 28일
새로운 내용	Amazon WorkMail에 대한 설명서를 추가했습니다.	2015년 1월 28일
새로운 내용	Amazon Relational Database Service의 사용 방식에 대한 설명서가 추가되었습니다 AWS KMS.	2015년 1월 6일
새로운 내용	Amazon Elastic Transcoder에 대한 설명서를 추가했습니다.	2014년 11월 24일
새 안내서	AWS Key Management Service 개발자 안내서를 도입했습니다.	2014년 11월 12일

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.