Add a permission의

AWS Secrets Manager

AWS Secrets Manager: Add a permission의

Table of Contents

Secrets Manager란 무엇인가요? 1
Secrets Manager 시작하기 1
표준 준수 2
가격 책정 2
Secrets Manager 액세스 4
Secrets Manager 콘솔 4
명령행 도구 4
AWS SDKs 5
HTTPS 쿼리 API 5
Secrets Manager 엔드포인트 6
모범 사례 11
에 자격 증명 및 기타 민감한 정보 저장 AWS Secrets Manager 11
코드에서 보호되지 않은 보안 암호 찾기 11
보안 암호에 대한 암호화 키 선택 12
캐싱을 사용하여 보안 암호 검색 12
보안 암호 교체 13
CLI 사용의 위험 완화 13
보안 암호에 대한 액세스 제한 13
BlockPublicPolicy 조건 13
정책의 IP 주소 조건에 주의하세요. 14
VPC 엔드포인트 조건이 있는 요청 제한 15
보안 암호 복제 15
보안 암호 모니터링 15
프라이빗 네트워크에서 인프라 실행 16
자습서 17
Amazon CodeGuru Reviewer 17
하드 코딩된 보안 암호 대체 17
1단계: 보안 암호 생성 18
2단계: 코드 업데이트 20
3단계: 보안 암호 업데이트 21
다음 단계 21
하드 코딩된 DB 보안 인증 정보 교체 22
1단계: 보안 암호 생성 22
2단계: 코드 업데이트 24

3단계: 보안 암호 교체 24
다음 단계 26
대체 사용자 교체 26
권한 27
사전 조건 27
1단계: Amazon RDS 데이터베이스 사용자 생성 30
2단계: 사용자 자격 증명에 대한 보안 암호 생성 33
3단계: 교체된 보안 암호 테스트 34
4단계: 리소스 정리 35
다음 단계 35
단일 사용자 교체 36
권한 36
사전 조건 36
1단계: Amazon RDS 데이터베이스 사용자 생성 37
2단계: 데이터베이스 사용자 자격 증명에 대한 보안 암호 생성 38
3단계: 교체된 암호 테스트 39
4단계: 리소스 정리 39
다음 단계 40
보안 암호 생성 41
AWS CLI 43
AWS SDK 45
보안 암호의 요소 45
Metadata 45
보안 암호 버전 46
보안 암호의 JSON 구조 47
Amazon RDS 및 Aurora 자격 증명 48
Amazon Redshift 자격 증명 51
Amazon Redshift Serverless 자격 증명 51
Amazon DocumentDB 자격 증명 52
Amazon Timestream for InfluxDB 보안 암호 구조 52
Amazon ElastiCache 자격 증명 52
Active Directory 자격 증명 53
보안 암호 관리 55
보안 암호 값 업데이트 55
AWS CLI 56
AWS SDK 56

Secrets Manager를 사용하여 암호 생성 57
보안 암호를 이전 버전으로 롤백 57
보안 암호에 대한 암호화 키 변경 57
AWS CLI 59
보안 암호 수정 60
AWS CLI 61
AWS SDK 61
보안 암호 찾기 61
검색 필터 62
AWS CLI 63
AWS SDK 64
보안 암호 삭제 64
AWS CLI 65
AWS SDK 66
보안 암호 복원 66
AWS CLI 67
AWS SDK 67
보안 암호 태그 지정 68
태그 기본 사항 검토 68
태그 지정을 사용하여 비용 추적 69
태그 제한 이해 69
콘솔에서 보안 암호의 태그 지정 70
AWS CLI 71
API 72
SDK 72
다중 리전 복제 73
AWS CLI 74
AWS SDK 75
복제본 보안 암호를 독립 실행형 보안 암호로 승격 75
AWS CLI 76
AWS SDK 76
복제 방지 76
복제 문제 해결 78
선택한 리전에 동일한 이름의 보안 암호가 있습니다 78
복제를 완료하기 위해 KMS 키에 사용할 수 있는 권한이 없습니다 78
KMS 키가 비활성화되었거나 찾을 수 없음 79

복제가 발생하는 리전을 활성화하지 않았습니다 79
보안 암호 가져오기 80
Java 81
클라이언트 측 캐싱을 사용한 Java 81
보안 암호의 자격 증명과 JDBC 연결 87
Java AWS SDK 97
Python 99
클라이언트 측 캐싱을 사용한 Python 99
Python AWS SDK 105
보안 암호 값 배치 가져오기 107
.NET 108
클라이언트 측 캐싱을 사용한 .NET 109
SDK for .NET 115
Go 118
클라이언트 측 캐싱을 사용한 Go 118
Go AWS SDK 122
Rust 124
클라이언트 측 캐싱을 사용한 Rust 124
Rust 126
Amazon EKS 127
IRSA(서비스 계정에 대한 IAM 역할)를 사용하는 ASCP 127
Pod Identity를 사용하는 ASCP 128
올바른 접근 방식 선택 128
ASCP for Amazon EKS 설치 128
Pod Identity for Amazon EKS와 ASCP 통합 132
IRSA for Amazon EKS와 ASCP 통합 136
ASCP 예제 139
AWS Lambda 147
Lambda에서 Secrets Manager 보안 암호 가져오기 147
Parameter Store 통합 148
Secrets Manager Agent 148
Secrets Manager Agent 작동 방식 148
Secrets Manager Agent 캐싱 이해 149
Secrets Manager Agent 빌드 150
Secrets Manager Agent 설치 154
Secrets Manager Agent를 사용하여 보안 암호 검색 158

refreshNow 파라미터 이해 160
구성 옵션 162
선택적 기능 164
로깅 164
보안 고려 사항 165
C++ 165
JavaScript 166
Kotlin 167
PHP 168
Ruby 169
AWS CLI 170
를 사용하여 일괄적으로 보안 암호 그룹 가져오기 AWS CLI 170
AWS 콘솔 171
AWS Batch 172
CloudFormation 172
GitHub 작업 173
사전 조건 173
사용법 174
환경 변수 이름 지정 175
예제 176
GitLab 178
고려 사항 179
사전 조건 179
GitLab AWS Secrets Manager 과 통합 181
문제 해결 182
AWS IoT Greengrass 182
파라미터 스토어 183
보안 암호 교체 184
관리형 교체 184
관리형 외부 보안 암호 교체 186
콘솔에서 교체 설정 186
CLI를 사용하여 교체 설정 187
Lambda 함수로 교체 187
데이터베이스 보안 암호 자동 교체(콘솔) 189
비데이터베이스 보안 암호 자동 교체(콘솔) 192
자동 교체(AWS CLI) 197

Lambda 함수 교체 전략 200
Lambda 교체 함수 202
교체 함수 템플릿 205
교체 권한 213
AWS Lambda 교체 함수에 대한 네트워크 액세스 217
교체 문제 해결 218
교체 일정 235
교체 기간 236
rate 표현식 236
cron 표현식 237
보안 암호 즉시 교체 242
AWS CLI 242
교체되지 않은 보안 암호 찾기 242
자동 교체 취소 243
다른 서비스에서 관리하는 보안 암호 244
보안 암호를 사용하는 서비스 245
App Runner 247
AWS App2Container 247
AWS AppConfig 247
Amazon AppFlow 248
AWS AppSync 248
Amazon Athena 248
Amazon Aurora 248
AWS CodeBuild 249
Amazon Data Firehose 249
AWS DataSync 249
Amazon DataZone 249
Direct Connect 250
AWS Directory Service 250
Amazon DocumentDB 250
AWS Elastic Beanstalk 251
Amazon Elastic Container Registry 251
Amazon Elastic Container Service 251
Amazon ElastiCache 252
AWS Elemental Live 252
AWS Elemental MediaConnect 252

AWS Elemental MediaConvert 253
AWS Elemental MediaLive 253
AWS Elemental MediaPackage 253
AWS Elemental MediaTailor 253
Amazon EMR 253
Amazon EventBridge 254
Amazon FSx 254
AWS Glue DataBrew 255
AWS Glue Studio 255
AWS IoT SiteWise 255
Amazon Kendra 255
Amazon Kinesis Video Streams 256
AWS Launch Wizard 256
Amazon Lookout for Metrics 256
Amazon Managed Grafana 256
AWS Managed Services 257
Amazon Managed Streaming for Apache Kafka 257
Amazon Managed Workflows for Apache Airflow 257
AWS Marketplace 257
AWS Migration Hub 258
AWS Panorama 258
AWS 병렬 컴퓨팅 서비스 258
AWS ParallelCluster 259
Amazon Q 259
Amazon OpenSearch Ingestion 259
AWS OpsWorks for Chef Automate 260
Amazon Quick 260
Amazon RDS 260
Amazon Redshift 260
Amazon Redshift Query Editor V2 261
Amazon SageMaker AI 261
AWS SCT 262
Amazon Timestream for InfluxDB 262
AWS Toolkit for JetBrains 262
AWS Transfer Family 263
AWS Wickr 263

타사 애플리케이션에서 관리하는 보안 암호 264
주요 기능 264
통합 파트너 265
Salesforce 클라이언트 보안 암호 265
빅 ID 새로 고침 토큰 267
Snowflake 키 페어 268
보안 및 권한 270
모니터링 및 문제 해결 272
기존 보안 암호 마이그레이션 272
제한 사항 및 고려 사항 272
CloudFormation 273
보안 암호 생성 273
JSON 274
YAML 274
자동 교체되는 Amazon RDS 자격 증명을 사용한 보안 암호 생성 275
Amazon Redshift 자격 증명을 사용하여 보안 암호 생성 275
Amazon DocumentDB 자격 증명을 사용하여 보안 암호 생성 275
JSON 275
YAML 280
Secrets Manager의 사용 방식 CloudFormation 282
AWS CDK 283
보안 암호 모니터링 284
를 사용하여 로그 AWS CloudTrail 284
AWS CLI 285
CloudTrail 항목 285
CloudWatch를 사용하여 모니터링 290
CloudWatch 경보 291
EventBridge를 사용하여 Secrets Manager 이벤트 매칭 292
모든 변경 사항을 지정된 암호와 매칭 292
암호 값이 교체될 때의 이벤트 매칭 292
삭제하도록 예약된 보안 암호 모니터링 293
1단계: CloudTrail 로그 파일이 CloudWatch Logs에 전송되도록 구성 293
2단계: CloudWatch 경보 생성 294
3단계: CloudWatch 경보 테스트 295
보안 암호의 규정 준수 모니터링 295
Secrets Manager 비용 모니터링 296

GuardDuty로 위협 탐지 ... 296
규정 준수 확인 ... 298
규정 준수 표준 ... 298
보안 ... 300
를 사용하여 AWS Secrets Manager 보안 암호를 AWS CLI 저장할 때의 위험 완화 ... 300
인증 및 액세스 제어 ... 303
권한 참조 ... 303
Secrets Manager 관리자 권한 ... 303
보안 암호에 액세스할 수 있는 권한 ... 304
Lambda 교체 함수에 대한 권한 ... 304
암호화 키 권한 ... 304
복제 권한 ... 304
ID 기반 정책 ... 304
리소스 기반 정책 ... 312
태그를 사용하여 보안 암호에 대한 액세스 제어 ... 318
AWS 관리형 정책 ... 320
보안 암호에 대한 권한이 있는 사용자 확인 ... 325
크로스 계정 액세스 ... 326
온프레미스 액세스 ... 329
Secrets Manager의 데이터 보호 ... 329
저장 시 암호화 ... 330
전송 중 데이터 암호화 ... 330
인터네트워크 트래픽 개인 정보 보호 ... 330
암호화 키 관리 ... 331
보안 암호 암호화 및 복호화 ... 331
AWS KMS 키 선택 ... 332
무엇을 암호화하나요? ... 333
암호화 및 복호화 프로세스 ... 333
KMS 키에 대한 권한 ... 334
Secrets Manager에서 KMS 키를 사용하는 방법 ... 334
AWS 관리형 키 (aws/secretsmanager)의 키 정책 ... 336
Secrets Manager 보안 암호 컨텍스트 ... 338
와의 Secrets Manager 상호 작용 모니터링 AWS KMS ... 340
인프라 보안 ... 344
VPC 엔드포인트(AWS PrivateLink) ... 344
엔드포인트 정책을 생성 ... 345

공유 서브넷 346
IPv4 및 IPv6 액세스 347
IPv6란? 347
듀얼 스택 정책 사용 347
정책에 IPv6 추가 348
클라이언트가 IPv6를 지원하는지 확인하기 350
복원력 351
포스트 양자 TLS 351
문제 해결 353
‘액세스가 거부됨’ 메시지 353
임시 보안 자격 증명에 대한 “액세스 거부됨”이라는 메시지 발생 353
변경 사항이 경우에 따라 즉시 표시되지 않습니다. 354
보안 암호를 생성할 때 “비대칭 KMS 키로 데이터 키를 생성할 수 없습니다.”라는 메시지 발생 .. 354
AWS CLI 또는 AWS SDK 작업이 부분 ARN에서 내 보안 암호를 찾을 수 없음 355
이 보안 암호는 AWS 서비스에서 관리하며 해당 서비스를 사용하여 업데이트해야 합니다. 355
`Transform: AWS::SecretsManager-2024-09-16` 사용 시 Python 모듈 가져오기 실패 356
할당량 357
Secrets Manager 할당량 357
애플리케이션에 재시도 추가 360
문서 기록 362
이전 업데이트 363
..... ccclxiv

AWS Secrets Manager란 무엇인가요?

AWS Secrets Manager 를 사용하면 수명 주기 동안 데이터베이스 자격 증명, 애플리케이션 자격 증명, OAuth 토큰, API 키 및 기타 보안 암호를 관리, 검색 및 교체할 수 있습니다. 많은 AWS 서비스가 Secrets Manager에 암호를 저장하고 사용합니다.

Secrets Manager를 사용하면 더 이상 애플리케이션 소스 코드에 하드 코딩된 보안 인증 정보가 필요하지 않으므로 보안 태세를 개선할 수 있습니다. Secrets Manager에 보안 인증 정보를 저장하면 애플리케이션 또는 구성 요소를 조사할 수 있는 누군가로 인해 손상될 가능성을 방지할 수 있습니다. 하드 코딩된 보안 인증 정보를 Secrets Manager 서비스에 대한 런타임 호출로 대체하여 필요할 때 동적으로 보안 인증 정보를 검색합니다.

Secrets Manager를 사용하면 암호에 대한 자동 교체 일정을 구성할 수 있습니다. 따라서 단기 보안 암호로 장기 보안 암호를 교체할 수 있어 손상 위험이 크게 줄어듭니다. 보안 인증 정보가 더 이상 애플리케이션에 저장되지 않으므로 보안 인증 정보를 교체할 때 더 이상 애플리케이션을 업데이트하거나 애플리케이션 클라이언트에 변경 사항을 배포하지 않아도 됩니다.

조직에 있을 수 있는 다른 유형의 암호는 다음과 같습니다.

- AWS 자격 증명 -를 사용하는 것이 좋습니다AWS Identity and Access Management.
- 암호화 키 — AWS Key Management Service 권장.
- SSH 키 – Amazon EC2 Instance Connect 권장.
- 프라이빗 키 및 인증서 – AWS Certificate Manager 권장.

Secrets Manager 시작하기

Secrets Manager를 처음 사용하는 경우 다음 자습서 중 하나로 시작하세요.

- the section called “하드 코딩된 보안 암호 대체 ”
- the section called “하드 코딩된 DB 보안 인증 정보 교체 ”
- the section called “대체 사용자 교체”
- the section called “단일 사용자 교체”

암호를 사용하여 수행할 수 있는 기타 작업은 다음과 같습니다.

- 보안 암호 관리

- 암호에 대한 액세스 제어
- 보안 암호 가져오기
- 보안 암호 교체
- 보안 암호 모니터링
- 보안 암호의 규정 준수 모니터링
- 에서 보안 암호 생성 AWS CloudFormation

표준 준수

AWS Secrets Manager 는 여러 표준에 대한 감사를 거쳤으며 규정 준수 인증을 받아야 할 때 솔루션의 일부가 될 수 있습니다. 자세한 내용은 규정 준수 확인 단원을 참조하십시오.

가격 책정

Secrets Manager를 사용할 경우 사용하는 내역에 대해서만 지불하며 최소 또는 설정 요금이 없습니다. 삭제하도록 표시한 보안 암호에 대해서는 요금이 부과되지 않습니다. 현재 기준의 전체적인 요금 목록은 AWS Secrets Manager 요금을 참조하세요. 비용을 모니터링하려면 the section called “Secrets Manager 비용 모니터링” 섹션을 참조하세요.

Secrets Manager가 AWS 관리형 키 `aws/secretsmanager` 생성하는를 사용하여 보안 암호를 무료로 암호화할 수 있습니다. 자체 KMS 키를 생성하여 보안 암호를 암호화하는 경우는 현재 AWS KMS 요금으로 AWS 요금을 청구합니다. 자세한 내용은 AWS Key Management Service 요금을 참조하세요.

자동 교체(관리형 교체 제외)를 켜면 Secrets Manager는 AWS Lambda 함수를 사용하여 보안 암호를 교체하며 현재 Lambda 속도로 교체 함수에 대한 요금이 부과됩니다. 자세한 내용은 AWS Lambda 요금을 참조하세요.

계정 AWS CloudTrail 에서를 활성화하면 Secrets Manager가 보내는 API 호출의 로그를 얻을 수 있습니다. Secrets Manager는 모든 이벤트를 관리 이벤트로 기록합니다.는 모든 관리 이벤트의 첫 번째 사본을 무료로 AWS CloudTrail 저장합니다. 하지만 알림을 활성화한 경우 로그 스토리지용 Amazon S3 및 Amazon SNS에 대한 비용이 발생할 수 있습니다. 또한 추적을 추가로 설정한 경우 관리 이벤트의 추가 사본으로 인해 비용이 발생할 수 있습니다. 자세한 내용은 AWS CloudTrail 요금 섹션을 참조하세요.

Secrets Manager에서 비용 할당 태그를 사용하면 특정 보안 암호나 프로젝트와 관련된 비용을 추적하고 분류할 수 있습니다. 자세한 내용은이 가이드the section called “ 보안 암호 태그 지정”의 및 AWS Billing 사용 설명서의 AWS 비용 할당 태그 사용을 참조하세요.

액세스 AWS Secrets Manager

다음 방법 중 하나를 사용하여 Secrets Manager로 작업할 수 있습니다.

- Secrets Manager 콘솔
- 명령행 도구
- AWS SDKs
- HTTPS 쿼리 API
- AWS Secrets Manager 엔드포인트

Secrets Manager 콘솔

브라우저 기반 Secrets Manager 콘솔을 사용하여 보안 암호를 관리하고 보안 암호와 관련된 거의 모든 작업을 수행할 수 있습니다.

명령행 도구

AWS 명령줄 도구를 사용하면 시스템 명령줄에서 명령을 실행하여 Secrets Manager 및 기타 AWS 작업을 수행할 수 있습니다. 명령줄을 사용하는 것이 콘솔을 사용하는 것보다 더 빠르고 편리할 수 있습니다. 명령줄 도구는 AWS 작업을 수행하기 위해 스크립트를 빌드하려는 경우에 유용할 수 있습니다.

명령 셸에 명령을 입력하면 명령 기록이 액세스되거나 유틸리티가 명령 파라미터에 액세스할 위험이 있습니다. the section called “를 사용하여 AWS Secrets Manager 보안 암호를 AWS CLI 저장할 때의 위험 완화”을(를) 참조하세요.

명령줄 도구는 AWS 리전의 서비스에 대한 기본 엔드포인트를 자동으로 사용합니다. API 요청에 다른 엔드포인트를 지정할 수 있습니다. the section called “Secrets Manager 엔드포인트”을(를) 참조하세요.

AWS 는 두 가지 명령줄 도구 세트를 제공합니다.

- AWS Command Line Interface (AWS CLI)
- AWS Tools for Windows PowerShell

AWS SDKs

AWS SDKs는 다양한 프로그래밍 언어 및 플랫폼을 위한 라이브러리와 샘플 코드로 구성됩니다. SDK는 요청에 암호화 방식으로 서명, 오류 관리 및 자동으로 요청 재시도와 같은 작업을 포함합니다. SDK를 다운로드하고 설치하려면 Amazon Web Services용 도구를 참조하세요.

AWS SDKs는 AWS 리전의 서비스에 대한 기본 엔드포인트를 자동으로 사용합니다. API 요청에 다른 엔드포인트를 지정할 수 있습니다. the section called “Secrets Manager 엔드포인트”을(를) 참조하세요.

SDK 설명서는 다음 섹션을 참조하세요.

- C++
- Go
- Java
- JavaScript
- Kotlin
- .NET
- PHP
- Python(Boto3)
- Ruby
- Rust
- SAP ABAP
- Swift

HTTPS 쿼리 API

HTTPS 쿼리 API를 사용하면 Secrets Manager 및에 프로그래밍 방식으로 액세스할 수 있습니다 AWS. HTTPS 쿼리 API를 이용하면 HTTPS 요청을 서비스에 직접 보낼 수 있습니다.

Secrets Manager HTTPS 쿼리 API를 직접 호출할 수도 있지만 SDK 중에서 한 가지를 대신 사용하는 것이 좋습니다. 직접 수행해야 하는 많은 유용한 작업을 SDK를 사용하여 수행할 수 있습니다. 예를 들어, SDK는 자동으로 요청에 서명하고, 응답을 해당 언어에 구문상 적절한 구조로 변환합니다.

Secrets Manager에 HTTPS 호출을 하려면 ???에 연결해야 합니다.

AWS Secrets Manager 엔드포인트

Secrets Manager에 프로그래밍 방식으로 연결하려면 해당 서비스에 대한 진입점의 URL인 엔드포인트를 사용합니다. Secrets Manager 엔드포인트는 듀얼 스택 엔드포인트이므로 IPv4와 IPv6를 모두 지원합니다.

Secrets Manager는 일부 리전에서 FIPS(Federal Information Processing Standard) 140-2를 지원하는 엔드포인트를 제공합니다.

Secrets Manager는 TLS 1.2 및 1.3을 지원합니다. Secrets Manager는 중국 리전을 제외한 모든 리전에서 PQTLS를 지원합니다.

> **Note**
>
> Python AWS SDK와 IPv6 및 IPv4를 순차적으로 호출하려는 AWS CLI 시도로 인해 IPv6가 활성화되지 않은 경우 호출 시간이 초과되고 IPv4로 재시도하는 데 시간이 걸릴 수 있습니다. 이 문제를 해결하려면 IPv6를 완전히 비활성화하거나 IPv6로 마이그레이션합니다.

Secrets Manager에 대한 서비스 엔드포인트는 다음과 같습니다. 이름 지정은 일반적인 듀얼 스택 이름 지정 규칙과 다르다는 점에 유의하세요. Secrets Manager에서 듀얼 스택 주소 지정을 사용하는 방법에 대한 자세한 내용은 IPv4 및 IPv6 액세스 섹션을 참조하세요.

리전 이름	리전	엔드포인트	프로토콜
미국 동부(오하이오)	us-east-2	secretsmanager.us-east-2.amazonaws.com	HTTPS
		secretsmanager-fips.us-east-2.amazonaws.com	HTTPS
미국 동부(버지니아 북부)	us-east-1	secretsmanager.us-east-1.amazonaws.com	HTTPS
		secretsmanager-fips.us-east-1.amazonaws.com	HTTPS
미국 서부(캘리포니아 북부)	us-west-1	secretsmanager.us-west-1.amazonaws.com	HTTPS
		secretsmanager-fips.us-west-1.amazonaws.com	HTTPS

리전 이름	리전	엔드포인트	프로토콜
미국 서부(오리곤)	us-west-2	secretsmanager.us-west-2.amazonaws.com secretsmanager-fips.us-west-2.amazonaws.com	HTTPS HTTPS
아프리카(케이프타운)	af-south-1	secretsmanager.af-south-1.amazonaws.com	HTTPS
아시아 태평양(홍콩)	ap-east-1	secretsmanager.ap-east-1.amazonaws.com	HTTPS
아시아 태평양(하이데라바드)	ap-south-2	secretsmanager.ap-south-2.amazonaws.com	HTTPS
아시아 태평양(자카르타)	ap-southeast-3	secretsmanager.ap-southeast-3.amazonaws.com	HTTPS
아시아 태평양(말레이시아)	ap-southeast-5	secretsmanager.ap-southeast-5.amazonaws.com	HTTPS
아시아 태평양(멜버른)	ap-southeast-4	secretsmanager.ap-southeast-4.amazonaws.com	HTTPS
아시아 태평양(뭄바이)	ap-south-1	secretsmanager.ap-south-1.amazonaws.com	HTTPS
아시아 태평양(뉴질랜드)	ap-southeast-6	secretsmanager.ap-southeast-6.amazonaws.com	HTTPS

리전 이름	리전	엔드포인트	프로토콜
아시아 태평양(오사카)	ap-northeast-3	secretsmanager.ap-northeast-3.amazonaws.com	HTTPS
아시아 태평양(서울)	ap-northeast-2	secretsmanager.ap-northeast-2.amazonaws.com	HTTPS
아시아 태평양(싱가포르)	ap-southeast-1	secretsmanager.ap-southeast-1.amazonaws.com	HTTPS
아시아 태평양(시드니)	ap-southeast-2	secretsmanager.ap-southeast-2.amazonaws.com	HTTPS
아시아 태평양(타이베이)	ap-east-2	secretsmanager.ap-east-2.amazonaws.com	HTTPS
아시아 태평양(태국)	ap-southeast-7	secretsmanager.ap-southeast-7.amazonaws.com	HTTPS
아시아 태평양(도쿄)	ap-northeast-1	secretsmanager.ap-northeast-1.amazonaws.com	HTTPS
캐나다(중부)	ca-central-1	secretsmanager.ca-central-1.amazonaws.com secretsmanager-fips.ca-central-1.amazonaws.com	HTTPS HTTPS
캐나다 서부(캘거리)	ca-west-1	secretsmanager.ca-west-1.amazonaws.com secretsmanager-fips.ca-west-1.amazonaws.com	HTTPS HTTPS

리전 이름	리전	엔드포인트	프로토콜
유럽(프랑크푸르트)	eu-central-1	secretsmanager.eu-central-1.amazonaws.com	HTTPS
유럽(아일랜드)	eu-west-1	secretsmanager.eu-west-1.amazonaws.com	HTTPS
유럽(런던)	eu-west-2	secretsmanager.eu-west-2.amazonaws.com	HTTPS
유럽(밀라노)	eu-south-1	secretsmanager.eu-south-1.amazonaws.com	HTTPS
유럽(파리)	eu-west-3	secretsmanager.eu-west-3.amazonaws.com	HTTPS
유럽(스페인)	eu-south-2	secretsmanager.eu-south-2.amazonaws.com	HTTPS
유럽(스톡홀름)	eu-north-1	secretsmanager.eu-north-1.amazonaws.com	HTTPS
유럽(취리히)	eu-central-2	secretsmanager.eu-central-2.amazonaws.com	HTTPS
이스라엘(텔아비브)	il-central-1	secretsmanager.il-central-1.amazonaws.com	HTTPS
멕시코(중부)	mx-central-1	secretsmanager.mx-central-1.amazonaws.com	HTTPS
중동(바레인)	me-south-1	secretsmanager.me-south-1.amazonaws.com	HTTPS
중동(UAE)	me-central-1	secretsmanager.me-central-1.amazonaws.com	HTTPS

리전 이름	리전	엔드포인트	프로토콜	
남아메리카(상파울루)	sa-east-1	secretsmanager.sa-east-1.amazonaws.com	HTTPS	
AWS GovCloud(미국 동부)	us-gov-east-1	secretsmanager.us-gov-east-1.amazonaws.com secretsmanager-fips.us-gov-east-1.amazonaws.com	HTTPS HTTPS	
AWS GovCloud(미국 서부)	us-gov-west-1	secretsmanager.us-gov-west-1.amazonaws.com secretsmanager-fips.us-gov-west-1.amazonaws.com	HTTPS HTTPS	

AWS Secrets Manager 모범 사례

Secrets Manager는 자체 보안 정책을 개발하고 구현할 때 고려해야 할 여러 보안 기능을 제공합니다. 다음 모범 사례는 일반적인 지침이며 완벽한 보안 솔루션을 나타내지는 않습니다. 이러한 모범 사례는 환경에 적절하지 않거나 충분하지 않을 수 있으므로 참고용으로만 사용하세요.

보안 암호를 저장하고 관리하기 위한 다음과 같은 모범 사례를 고려하세요.

- 에 자격 증명 및 기타 민감한 정보 저장 AWS Secrets Manager
- 코드에서 보호되지 않은 보안 암호 찾기
- 보안 암호에 대한 암호화 키 선택
- 캐싱을 사용하여 보안 암호 검색
- 보안 암호 교체
- CLI 사용의 위험 완화
- 보안 암호에 대한 액세스 제한
- 보안 암호 복제
- 보안 암호 모니터링
- 프라이빗 네트워크에서 인프라 실행

에 자격 증명 및 기타 민감한 정보 저장 AWS Secrets Manager

Secrets Manager는 보안 태세와 규정 준수를 개선하고, 민감한 정보에 대한 무단 액세스 위험을 줄이는 데 도움이 될 수 있습니다. Secrets Manager는 사용자가 소유하고 저장하는 암호화 키 AWS Key Management Service ()를 사용하여 저장 시 암호를 암호화합니다AWS KMS. 보안 암호를 검색하면 Secrets Manager는 보안 암호를 복호화한 후 TLS를 통해 이를 로컬 환경으로 안전하게 전송합니다. 자세한 내용은 보안 암호 생성 단원을 참조하십시오.

코드에서 보호되지 않은 보안 암호 찾기

CodeGuru Reviewer는 Secrets Manager와 통합되어 코드에서 보호되지 않는 보안 암호를 찾는 보안 암호 탐지기를 사용합니다. 보안 암호 탐지기는 하드코딩된 암호, 데이터베이스 연결 문자열, 사용자 이름 등을 검색합니다. 자세한 내용은 the section called “ Amazon CodeGuru Reviewer” 단원을 참조하십시오.

Amazon Q는 코드베이스에서 보안 취약성 및 코드 품질 문제를 스캔하여 개발 주기 전체에서 애플리케이션의 태세를 개선할 수 있습니다. 자세한 내용은 Amazon Q Developer 사용 설명서의 Scanning your code with Amazon Q 섹션을 참조하세요.

보안 암호에 대한 암호화 키 선택

대부분의 경우 `aws/secretsmanager` AWS 관리형 키를 사용하여 보안 암호를 암호화하는 것이 좋습니다. 이를 사용하는 데 드는 비용은 없습니다.

다른 계정의 보안 암호에 액세스하거나 키 정책을 암호화 키에 적용하려면 고객 관리형 키를 사용하여 보안 암호를 암호화합니다.

- 키 정책에서 `secretsmanager.<region>.amazonaws.com` 값을 `kms:ViaService` 조건 키에 할당합니다. 이렇게 하면 Secrets Manager의 요청에만 키를 사용하도록 제한됩니다.
- 올바른 컨텍스트를 가진 Secrets Manager의 요청에만 키를 사용하도록 제한하려면, 다음 항목을 생성하여 KMS 키를 사용하는 조건으로 Secrets Manager 암호화 컨텍스트의 키 또는 값을 사용합니다.
  - IAM 정책 또는 키 정책의 문자열 조건 연산자
  - 권한 부여의 권한 부여 제약 조건

자세한 내용은 the section called “보안 암호 암호화 및 복호화” 단원을 참조하십시오.

캐싱을 사용하여 보안 암호 검색

보안 암호를 가장 효율적으로 사용하려면 다음과 같은 지원되는 Secrets Manager 캐싱 구성 요소 중 하나를 사용하여 보안 암호를 캐싱하고, 필요한 경우에만 업데이트하는 것이 좋습니다.

- 클라이언트 측 캐싱을 사용한 Java
- 클라이언트 측 캐싱을 사용한 Python
- 클라이언트 측 캐싱을 사용한 .NET
- 클라이언트 측 캐싱을 사용한 Go
- 클라이언트 측 캐싱을 사용한 Rust
- AWS 파라미터 및 보안 암호 Lambda 확장
- the section called “Amazon EKS”

- Amazon Elastic Container Service AWS Lambda, Amazon Elastic Kubernetes Service, Amazon Elastic Compute Cloud와 같은 환경에서 Secrets Manager의 보안 암호 사용을 표준화하는 the section called “Secrets Manager Agent” 데 사용합니다.

보안 암호 교체

오랜 기간 동안 보안 암호를 바꾸지 않으면 보안 암호가 손상될 가능성이 높아집니다. Secrets Manager를 사용하면 최대 4시간 간격으로 자동 교체를 설정할 수 있습니다. Secrets Manager에서는 단일 사용자 및 대체 사용자라는 두 가지 교체 전략이 제공됩니다. 자세한 내용은 보안 암호 교체 단원을 참조하십시오.

CLI 사용의 위험 완화

AWS CLI 를 사용하여 AWS 작업을 호출하는 경우 명령 셸에 해당 명령을 입력합니다. 대부분의 명령 쉘은 로깅 및 마지막으로 입력한 명령을 볼 수 있는 기능처럼 보안 암호를 손상시킬 수 있는 기능을 제공합니다. AWS CLI 를 사용하여 민감한 정보를 입력하기 전에 먼저 the section called “를 사용하여 AWS Secrets Manager 보안 암호를 AWS CLI 저장할 때의 위험 완화” 섹션을 확인하세요.

보안 암호에 대한 액세스 제한

보안 암호에 대한 액세스를 제어하는 IAM 정책 설명에서 최소 권한 액세스 원칙을 사용합니다. IAM 역할 및 정책, 리소스 정책, 속성 기반 액세스 제어(ABAC)를 사용할 수 있습니다. 자세한 내용은 the section called “인증 및 액세스 제어” 단원을 참조하십시오.

주제

- 보안 암호에 대한 광범위한 액세스 차단
- 정책의 IP 주소 조건에 주의하세요.
- VPC 엔드포인트 조건이 있는 요청 제한

보안 암호에 대한 광범위한 액세스 차단

`PutResourcePolicy` 작업을 허용하는 자격 증명 정책에서 `BlockPublicPolicy: true`를 사용하는 것이 좋습니다. 이 조건은 정책에서 광범위한 액세스를 허용하지 않는 경우에만 사용자가 리소스 정책을 보안 암호에 연결할 수 있음을 의미합니다.

Secrets Manager는 Zelkova 자동화된 추론을 사용하여 광범위한 액세스에 대한 리소스 정책을 분석합니다. Zelkova에 대한 자세한 내용은 AWS 보안 블로그의 자동 추론을 AWS 사용하여 대규모 보안을 달성하는 방법을 참조하세요.

다음 예에서는 `BlockPublicPolicy`를 사용하는 방법을 보여줍니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "secretsmanager:PutResourcePolicy",
        "Resource": "arn:aws:secretsmanager:us-
east-1:123456789012:secret:secretName-AbCdEf",
    "Condition": {
            "Bool": {
                "secretsmanager:BlockPublicPolicy": "true"
            }
        }
    }
}
```

정책의 IP 주소 조건에 주의하세요.

그러나 Secrets Manager에 대한 액세스를 허용하거나 거부하는 정책문에서 IP 주소 조건 연산자 또는 `aws:SourceIp` 조건 키를 지정할 때는 주의해야 합니다 예를 들어 회사 네트워크 IP 주소 범위의 요청으로 AWS 작업을 제한하는 정책을 보안 암호로 연결하면 회사 네트워크에서 요청을 호출하는 IAM 사용자 요청은 예상대로 작동합니다. 그러나 Lambda 함수로 교체를 활성화하는 경우와 같이 다른 서비스가 사용자를 대신하여 보안 암호에 액세스하도록 활성화하는 경우 해당 함수는 AWS내부 주소 공간에서 Secrets Manager 작업을 호출합니다. IP 주소 필터가 있는 정책의 영향을 받는 요청은 실패합니다.

또한 요청이 Amazon VPC 엔드포인트에서 이루어지는 경우 `aws:sourceIP` 조건 키는 유효하지 않습니다. 요청을 특정 VPC 엔드포인트로 제한하려면 the section called “VPC 엔드포인트 조건이 있는 요청 제한”를 사용합니다.

VPC 엔드포인트 조건이 있는 요청 제한

특정 VPC 또는 VPC 엔드포인트의 요청으로 액세스를 허용하거나 거부하려면 `aws:SourceVpc`를 사용하여 지정된 VPC 요청으로 액세스를 제한하거나 `aws:SourceVpce`를 사용하여 지정된 VPC 엔드포인트의 요청으로 액세스를 제한합니다. the section called “예: 권한 및 VPC”을(를) 참조하세요.

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

Secrets Manager 보안 암호에 대한 액세스를 허용하거나 거부하는 보안 암호 정책문에 이러한 조건 키를 사용하면 사용자를 대신해 Secrets Manager를 사용하여 보안 암호에 액세스하는 서비스에 대한 액세스를 실수로 거부하게 될 수 있습니다. 일부 AWS 서비스만 VPC 내의 엔드포인트로 실행할 수 있습니다. 보안 암호에 대한 요청을 VPC 또는 VPC 엔드포인트로 제한하면 구성되지 않은 서비스로부터의 Secrets Manager 호출에 실패할 수 있습니다.

the section called “VPC 엔드포인트(AWS PrivateLink)”을(를) 참조하세요.

보안 암호 복제

Secrets Manager는 복원력 또는 재해 복구 요구 사항을 충족하기 위해 여러 AWS 리전에 보안 암호를 자동으로 복제할 수 있습니다. 자세한 내용은 다중 리전 복제 단원을 참조하십시오.

보안 암호 모니터링

Secrets Manager를 사용하면 AWS 로깅, 모니터링 및 알림 서비스와의 통합을 통해 보안 암호를 감사하고 모니터링할 수 있습니다. 자세한 내용은 다음을 참조하세요.

- the section called “를 사용하여 로그 AWS CloudTrail ”
- the section called “CloudWatch를 사용하여 모니터링”
- the section called “보안 암호의 규정 준수 모니터링”
- the section called “Secrets Manager 비용 모니터링”
- the section called “GuardDuty로 위협 탐지”

프라이빗 네트워크에서 인프라 실행

가능한 경우 대부분의 인프라를 퍼블릭 인터넷에서 액세스할 수 없는 프라이빗 네트워크에서 실행하는 것이 좋습니다. 인터페이스 VPC 엔드포인트를 생성하여 VPC와 Secrets Manager 간에 프라이빗 연결을 설정할 수 있습니다. 자세한 내용은 the section called “VPC 엔드포인트(AWS PrivateLink)” 단원을 참조하십시오.

AWS Secrets Manager 자습서

주제

- Amazon CodeGuru Reviewer를 사용하여 코드에서 보호되지 않는 보안 암호 찾기
- 하드코딩된 보안 암호를 로 이동 AWS Secrets Manager
- 하드코딩된 데이터베이스 자격 증명을 로 이동 AWS Secrets Manager
- 에 대한 대체 사용자 교체 설정 AWS Secrets Manager
- 에 대한 단일 사용자 교체 설정 AWS Secrets Manager

Amazon CodeGuru Reviewer를 사용하여 코드에서 보호되지 않는 보안 암호 찾기

Amazon CodeGuru Reviewer는 프로그램 분석 및 기계 학습을 사용하여 개발자가 찾기 어려운 잠재적 결함을 감지하고 Java 및 Python 코드를 개선하기 위한 제안을 제공하는 서비스입니다. CodeGuru Reviewer는 Secrets Manager와 통합되어 코드에서 보호되지 않는 보안 암호를 찾습니다. 찾을 수 있는 보안 암호 유형은 Amazon CodeGuru Reviewer 사용 설명서의 CodeGuru Reviewer가 감지하는 보안 암호 유형을 참조하세요.

하드 코딩된 보안 암호를 발견하면 이를 대체하기 위한 조치를 취하세요.

- the section called “하드 코딩된 DB 보안 인증 정보 교체 ”
- the section called “하드 코딩된 보안 암호 대체 ”

하드코딩된 보안 암호를 로 이동 AWS Secrets Manager

코드에 일반 텍스트 보안 암호가 있는 경우 보안 암호를 교체하고 Secrets Manager에 저장하는 것이 좋습니다. 보안 암호를 Secrets Manager로 이동하면 코드를 보는 모든 사용자가 보안 암호를 볼 수 있는 문제가 해결됩니다. 앞으로는 코드가 Secrets Manager에서 직접 보안 암호를 검색하기 때문입니다. 보안 암호를 교체하면 현재의 하드 코딩된 보안 암호가 취소되어 더 이상 유효하지 않게 됩니다.

데이터베이스 보안 인증 정보 보안 암호에 대해서는 하드코딩된 데이터베이스 자격 증명을 로 이동 AWS Secrets Manager 섹션을 참조하세요.

시작하기 전에 보안 암호에 액세스해야 하는 사용자를 결정해야 합니다. 다음과 같이 두 개의 IAM 역할을 사용하여 보안 암호에 대한 권한을 관리하는 것이 좋습니다.

- 조직의 보안 암호를 관리하는 역할. 자세한 정보는 the section called “Secrets Manager 관리자 권한” 섹션을 참조하세요. 이 역할을 사용하여 보안 암호를 생성하고 교체합니다.
- 런타임 시 보안 암호를 사용할 수 있는 역할. 예를 들어 이 자습서에서는 *RoleToRetrieveSecretAtRuntime*을 사용합니다. 코드가 이 역할을 수임하여 보안 암호를 검색합니다. 이 자습서에서는 역할에 하나의 보안 암호 값을 검색할 수 있는 권한만 부여하고, 보안 암호의 리소스 정책을 사용하여 권한을 부여합니다. 다른 방법을 보려면 the section called “다음 단계” 섹션을 참조하세요.

단계:

- 1단계: 보안 암호 생성
- 2단계: 코드 업데이트
- 3단계: 보안 암호 업데이트
- 다음 단계

1단계: 보안 암호 생성

첫 번째 단계는 기존의 하드 코딩된 보안 암호를 Secrets Manager로 복사하는 것입니다. 보안 암호가 AWS 리소스와 관련된 경우 리소스와 동일한 리전에 저장합니다. 그렇지 않으면 사용 사례에 대해 지연 시간이 가장 짧은 리전에 저장합니다.

보안 암호(콘솔) 생성

1. https://console.aws.amazon.com/secretsmanager/에서 Secrets Manager 콘솔을 엽니다.
2. Store a new secret(새 보안 암호 저장)을 선택합니다.
3. 보안 암호 유형 선택(Choose secret type) 페이지에서 다음을 수행합니다.
   a. 보안 암호 유형에서 다른 유형의 보안 암호를 선택합니다.
   b. 보안 암호를 키/값 쌍(Key/value pairs) 또는 일반 텍스트(Plaintext)로 입력합니다. 다음은 몇 가지 예제입니다.

      API key

      키/값 페어로 입력:

      ClientID: *my_client_id*

      ClientSecret : *wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY*

OAuth token

일반 텍스트로 입력:

AKIAI44QH8DHBEXAMPLE

Digital certificate

일반 텍스트로 입력:

```
-----BEGIN CERTIFICATE-----
EXAMPLE
-----END CERTIFICATE-----
```

Private key

일반 텍스트로 입력:

```
---- BEGIN PRIVATE KEY ----
EXAMPLE
---- END PRIVATE KEY ----
```

c. 암호화 키(Encryption key)에서 aws/secretsmanager를 선택하여 Secrets Manager에 대해 AWS 관리형 키 를 사용합니다. 이 키를 사용하는 데 드는 비용은 없습니다. 예를 들어 다른 AWS 계정에서 보안 암호에 액세스하기 위해 자체 고객 관리형 키를 사용할 수도 있습니다. 고객 관리형 키 사용 비용에 대한 자세한 내용은 가격 책정을 참조하세요.

d. 다음을 선택합니다.

4. 보안 암호 유형 선택(Choose secret type) 페이지에서 다음을 수행합니다.

a. 설명이 포함된 Secret name(보안 암호 이름)과 Description(설명)을 입력합니다.

b. 리소스 권한(Resource permissions)에서 권한 편집(Edit permissions)을 선택합니다. *RoleToRetrieveSecretAtRuntime*이 보안 암호를 검색할 수 있도록 하는 다음 정책을 붙여넣은 후 저장(Save)을 선택합니다.

JSON

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

```
            "Effect": "Allow",
            "Principal": {
                "AWS":
 "arn:aws:iam::111122223333:role/RoleToRetrieveSecretAtRuntime"
            },
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "*"
        }
    ]
}
```

c. 페이지 하단에서 다음(Next)을 선택합니다.

5. 교체 구성(Configure rotation) 페이지에서 교체를 꺼 둡니다. 다음을 선택합니다.
6. 검토(Review) 페이지에서 보안 암호 세부 정보를 검토한 후 저장(Store)을 선택합니다.

2단계: 코드 업데이트

코드가 반드시 IAM 역할 *RoleToRetrieveSecretAtRuntime*을 수임하여 보안 암호를 검색할 수 있도록 합니다. 자세한 내용은 IAM 역할로 전환(AWS API)을 참조하세요.

그런 다음 Secrets Manager에서 제공하는 샘플 코드를 사용하여 Secrets Manager에서 보안 암호를 검색하도록 코드를 업데이트합니다.

샘플 코드 찾기

1. https://console.aws.amazon.com/secretsmanager/에서 Secrets Manager 콘솔을 엽니다.
2. 보안 암호(Secrets) 페이지에서 보안 암호를 선택합니다.
3. 아래로 스크롤하여 샘플 코드(Sample code)로 이동합니다. 프로그래밍 언어를 선택한 다음 코드 조각을 복사합니다.

애플리케이션에서 하드 코딩된 보안 암호를 제거하고 코드 조각을 붙여넣습니다. 코드 언어에 따라 코드 조각의 함수 또는 메서드에 호출을 추가해야 할 수 있습니다.

하드 코딩된 보안 암호 대신 보안 암호를 사용하여 애플리케이션이 예상대로 작동하는지 테스트합니다.

3단계: 보안 암호 업데이트

마지막 단계는 하드 코딩된 보안 암호를 취소하고 업데이트하는 것입니다. 보안 암호의 소스를 참조하여 보안 암호를 취소하고 업데이트하기 위한 지침을 찾습니다. 예를 들어 현재 보안 암호를 비활성화하고 새 보안 암호를 생성해야 할 수 있습니다.

보안 암호를 새 값으로 업데이트

1. https://console.aws.amazon.com/secretsmanager/에서 Secrets Manager 콘솔을 엽니다.
2. 보안 암호(Secrets)를 선택한 다음 보안 암호를 선택합니다.
3. 보안 암호 세부 정보(Secret details) 페이지에서 아래로 스크롤하여 보안 암호 값 검색(Retrieve secret value)을 선택한 다음 편집(Edit)을 선택합니다.
4. 보안 암호를 업데이트 한 다음 저장(Save)을 선택합니다.

그 다음, 새 보안 암호로 애플리케이션이 예상대로 작동하는지 테스트합니다.

다음 단계

코드에서 하드 코딩된 보안 암호를 제거한 후에는 다음과 같은 몇 가지 아이디어를 고려해 볼 수 있습니다.

- Java 및 Python 애플리케이션에서 하드 코딩된 보안 암호를 찾으려면 Amazon CodeGuru Reviewer를 사용하는 것이 좋습니다.
- 보안 암호를 캐싱하여 성능을 개선하고 비용을 절감할 수 있습니다. 자세한 정보는 보안 암호 가져오기 섹션을 참조하세요.
- 여러 리전에서 액세스하는 보안 암호의 경우 지연 시간을 개선하기 위해 보안 암호를 복제하는 것을 고려해 보세요. 자세한 내용은 다중 리전 복제 단원을 참조하십시오.
- 이 자습서에서는 *RoleToRetrieveSecretAtRuntime*에 보안 암호 값을 검색할 권한만 부여했습니다. 역할에 추가 권한(예: 보안 암호에 대한 메타데이터를 가져오거나 보안 암호 목록 보기)을 부여하려면 the section called “리소스 기반 정책” 섹션을 참조하세요.
- 이 자습서에서는 보안 암호의 리소스 정책을 사용하여 *RoleToRetrieveSecretAtRuntime*에 권한을 부여했습니다. 권한을 부여하는 다른 방법은 the section called “ID 기반 정책” 섹션을 참조하세요.

하드코딩된 데이터베이스 자격 증명을 로 이동 AWS Secrets Manager

코드에 일반 텍스트 데이터베이스 보안 인증 정보가 있는 경우 보안 인증 정보를 Secrets Manager로 이동한 다음 즉시 교체하는 것이 좋습니다. 보안 인증 정보를 Secrets Manager로 이동하면 코드를 보는 모든 사용자가 보안 인증 정보를 볼 수 있는 문제가 해결됩니다. 앞으로는 코드가 Secrets Manager에서 직접 보안 인증 정보를 검색하기 때문입니다. 보안 암호를 교체하면 암호가 업데이트된 다음 현재의 하드 코딩된 암호가 취소되어 더 이상 유효하지 않게 됩니다.

Amazon RDS, Amazon Redshift, Amazon DocumentDB 데이터베이스의 경우, 이 페이지에 제시된 단계를 사용하여 하드 코딩된 보안 인증 정보를 Secrets Manager로 이동합니다. 다른 유형의 보안 인증 정보 및 다른 보안 암호에 대해서는 the section called “하드 코딩된 보안 암호 대체 ” 섹션을 참조하세요.

시작하기 전에 보안 암호에 액세스해야 하는 사용자를 결정해야 합니다. 다음과 같이 두 개의 IAM 역할을 사용하여 보안 암호에 대한 권한을 관리하는 것이 좋습니다.

- 조직의 보안 암호를 관리하는 역할. 자세한 정보는 the section called “Secrets Manager 관리자 권한” 섹션을 참조하세요. 이 역할을 사용하여 보안 암호를 생성하고 교체합니다.
- 런타임 시 보안 인증 정보를 사용할 수 있는 역할. 이 자습서에서는 *RoleToRetrieveSecretAtRuntime*을 사용합니다. 코드가 이 역할을 수임하여 보안 암호를 검색합니다.

단계:

- 1단계: 보안 암호 생성
- 2단계: 코드 업데이트
- 3단계: 보안 암호 교체
- 다음 단계

1단계: 보안 암호 생성

첫 번째 단계는 기존의 하드 코딩된 보안 인증 정보를 Secrets Manager의 보안 암호로 복사하는 것입니다. 지연 시간을 최소화하려면 보안 암호를 데이터베이스와 동일한 리전에 저장합니다.

보안 암호 생성

1. https://console.aws.amazon.com/secretsmanager/에서 Secrets Manager 콘솔을 엽니다.
2. Store a new secret(새 보안 암호 저장)을 선택합니다.
3. 보안 암호 유형 선택(Choose secret type) 페이지에서 다음을 수행합니다.
   a. 보안 암호 유형(Secret type)에서 저장할 데이터베이스 보안 인증 정보 유형을 선택합니다.
      - Amazon RDS 데이터베이스
      - Amazon DocumentDB 데이터베이스
      - Amazon Redshift 데이터 웨어하우스.
      - 다른 유형의 보안 암호에 대해서는 하드 코딩된 보안 암호 대체를 참조하세요.
   b. 보안 인증 정보에서 데이터베이스에 대한 기존의 하드 코딩된 보안 인증 정보를 입력합니다.
   c. 암호화 키(Encryption key)에서 aws/secretsmanager를 선택하여 Secrets Manager에 대해 AWS 관리형 키 를 사용합니다. 이 키를 사용하는 데 드는 비용은 없습니다. 예를 들어 다른 AWS 계정에서 보안 암호에 액세스하기 위해 자체 고객 관리형 키를 사용할 수도 있습니다. 고객 관리형 키 사용 비용에 대한 자세한 내용은 가격 책정을 참조하세요.
   d. 데이터베이스에서 데이터베이스(Database)를 선택합니다.
   e. 다음을 선택합니다.
4. 보안 구성(Configure secret) 페이지에서 다음을 수행합니다.
   a. 설명이 포함된 Secret name(보안 암호 이름)과 Description(설명)을 입력합니다.
   b. 리소스 권한(Resource permissions)에서 권한 편집(Edit permissions)을 선택합니다. *RoleToRetrieveSecretAtRuntime*이 보안 암호를 검색할 수 있도록 하는 다음 정책을 붙여넣은 후 저장(Save)을 선택합니다.

      JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS":
 "arn:aws:iam::111122223333:role/RoleToRetrieveSecretAtRuntime"
            },
```

```
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "*"
        }
    ]
}
```

c. 페이지 하단에서 다음(Next)을 선택합니다.

5. 교체 구성(Configure rotation) 페이지에서 지금은 교체를 꺼 둡니다. 나중에 켜게 됩니다. 다음을 선택합니다.
6. 검토(Review) 페이지에서 보안 암호 세부 정보를 검토한 후 저장(Store)을 선택합니다.

2단계: 코드 업데이트

코드가 반드시 IAM 역할 *RoleToRetrieveSecretAtRuntime*을 수임하여 보안 암호를 검색할 수 있도록 합니다. 자세한 내용은 IAM 역할로 전환(AWS API)을 참조하세요.

그런 다음 Secrets Manager에서 제공하는 샘플 코드를 사용하여 Secrets Manager에서 보안 암호를 검색하도록 코드를 업데이트합니다.

샘플 코드 찾기

1. https://console.aws.amazon.com/secretsmanager/에서 Secrets Manager 콘솔을 엽니다.
2. 보안 암호(Secrets) 페이지에서 보안 암호를 선택합니다.
3. 아래로 스크롤하여 샘플 코드(Sample code)로 이동합니다. 언어를 선택한 다음 코드 조각을 복사합니다.

애플리케이션에서 하드 코딩된 보안 인증 정보를 제거하고 코드 조각을 붙여넣습니다. 코드 언어에 따라 코드 조각의 함수 또는 메서드에 호출을 추가해야 할 수 있습니다.

하드 코딩된 보안 인증 정보 대신 보안 암호를 사용하여 애플리케이션이 예상대로 작동하는지 테스트합니다.

3단계: 보안 암호 교체

마지막 단계는 보안 암호를 교체하여 하드 코딩된 보안 인증 정보를 취소하는 것입니다. 교체는 주기적으로 보안 암호를 업데이트하는 프로세스입니다. 보안 암호를 교체하면 보안 암호와 데이터베이스 모두에서 보안 인증 정보가 업데이트됩니다. Secrets Manager는 사용자가 설정한 일정에 따라 보안 암호를 자동으로 교체할 수 있습니다.

교체 설정의 일부는 Lambda 교체 함수가 Secrets Manager와 데이터베이스에 모두 액세스할 수 있도록 하는 것입니다. 자동 교체를 설정하면 Secrets Manager가 데이터베이스에 대한 네트워크 액세스 권한을 갖도록 데이터베이스와 동일한 VPC에 Lambda 교체 함수를 생성합니다. 또한 Lambda 교체 함수는 Secrets Manager를 호출하여 보안 암호를 업데이트할 수 있어야 합니다. Lambda에서 Secrets Manager로의 호출이 AWS 인프라를 벗어나지 않도록 VPC에서 Secrets Manager 엔드포인트를 생성하는 것이 좋습니다. 지침은 the section called “VPC 엔드포인트(AWS PrivateLink)” 섹션을 참조하세요.

교체 켜기

1. https://console.aws.amazon.com/secretsmanager/에서 Secrets Manager 콘솔을 엽니다.
2. 보안 암호(Secrets) 페이지에서 보안 암호를 선택합니다.
3. 보안 암호 세부 정보(Secret details) 페이지의 교체 구성(Rotation configuration) 섹션에서 교체 편집(Edit rotation)을 선택합니다.
4. Edit rotation configuration(교체 구성 편집) 대화 상자에서 다음을 수행합니다.

    a. Automatic rotation(자동 교체)을 켭니다.

    b. 교체 일정(Rotation schedule)에서 UTC 표준 시간대로 일정을 입력합니다.

    c. 변경 사항을 저장할 때 보안 암호가 교체되게 하려면 보안 암호를 저장할 때 즉시 교체(Rotate immediately when the secret is stored)를 선택합니다.

    d. 교체 함수(Rotation function)에서 새 Lambda 함수 생성(Create a new Lambda function)을 선택하고 새 함수의 이름을 입력합니다. Secrets Manager에서 함수 이름의 시작 부분에 "SecretsManager"를 추가합니다.

    e. 교체 전략에서 단일 사용자를 선택합니다.

    f. 저장(Save)을 선택합니다.

보안 암호가 교체되었는지 확인

1. https://console.aws.amazon.com/secretsmanager/에서 Secrets Manager 콘솔을 엽니다.
2. 보안 암호(Secrets)를 선택한 다음 보안 암호를 선택합니다.
3. 보안 암호 세부 정보(Secret details) 페이지를 아래로 스크롤하고 보안 암호 값 검색(Retrieve secret value)을 선택합니다.

    보안 암호 값이 변경되었으면 교체가 성공한 것입니다. 보안 암호 값이 변경되지 않은 경우 CloudWatch Logs에서 교체 함수를 확인하여 교체 문제 해결을 수행해야 합니다.

교체된 보안 암호로 애플리케이션이 예상대로 작동하는지 테스트합니다.

다음 단계

코드에서 하드 코딩된 보안 암호를 제거한 후에는 다음과 같은 몇 가지 아이디어를 고려해 볼 수 있습니다.

- 보안 암호를 캐싱하여 성능을 개선하고 비용을 절감할 수 있습니다. 자세한 정보는 보안 암호 가져오기 섹션을 참조하세요.
- 다른 교체 일정을 선택할 수 있습니다. 자세한 내용은 the section called “교체 일정” 단원을 참조하십시오.
- Java 및 Python 애플리케이션에서 하드 코딩된 보안 암호를 찾으려면 Amazon CodeGuru Reviewer를 사용하는 것이 좋습니다.

에 대한 대체 사용자 교체 설정 AWS Secrets Manager

이 자습서에서는 데이터베이스 자격 증명이 포함된 보안 암호에 대해 대체 사용자 교체를 설정하는 방법을 알아봅니다. 대체 사용자 교체는 Secrets Manager가 사용자를 복제한 다음 업데이트되는 사용자의 자격 증명을 대체하는 교체 전략입니다. 이 전략은 보안 암호의 고가용성이 필요한 경우 선택하는 것이 좋습니다. 대체 사용자 중 한 명이 업데이트되는 동안 다른 사용자에게 데이터베이스에 대한 현재 자격 증명이 있기 때문입니다. 자세한 내용은 the section called “대체 사용자” 단원을 참조하십시오.

대체 사용자 교체를 설정하려면 다음 두 개의 보안 암호가 필요합니다.

- 교체하려는 자격 증명이 포함된 보안 암호 1개와
- 관리자 자격 증명이 있는 두 번째 암호입니다.

  이 사용자는 첫 번째 사용자를 복제하고 첫 번째 사용자의 암호를 변경할 권한이 있습니다. 이 자습서에서는 Amazon RDS에서 관리자 사용자를 위해 이 암호를 생성하도록 합니다. Amazon RDS는 관리자 암호 교체도 관리합니다. 자세한 내용은 the section called “관리형 교체” 단원을 참조하십시오.

이 자습서의 첫 부분은 사실적인 환경을 설정하는 것입니다. 이 자습서에서는 교체 방식을 보여주기 위해 Amazon RDS MySQL 데이터베이스 예제를 사용합니다. 보안을 위해 데이터베이스는 인바운드 인터넷 액세스를 제한하는 VPC에 있습니다. 인터넷을 통해 로컬 컴퓨터에서 데이터베이스에 연결하려면 Bastion Host를 사용합니다. 이것은 데이터베이스에 연결할 수 있지만 인터넷으로부터의 SSH 연결

도 허용하는 VPC 서버입니다. 이 자습서의 Bastion Host는 Amazon EC2 인스턴스이며, 인스턴스의 보안 그룹은 다른 유형의 연결을 차단합니다.

자습서를 마친 후에는 자습서에서 리소스를 정리하는 것이 좋습니다. 프로덕션 환경에서 리소스를 사용하지 마세요.

Secrets Manager 교체는 AWS Lambda 함수를 사용하여 보안 암호와 데이터베이스를 업데이트합니다. Lambda 함수 사용 비용에 대한 자세한 내용은 가격 책정 섹션을 참조하세요.

자습서

- 권한
- 사전 조건
- 1단계: Amazon RDS 데이터베이스 사용자 생성
- 2단계: 사용자 자격 증명에 대한 보안 암호 생성
- 3단계: 교체된 보안 암호 테스트
- 4단계: 리소스 정리
- 다음 단계

권한

자습서 사전 조건으로 AWS 계정에 대한 관리 권한이 필요합니다. 프로덕션 설정에서는 각 단계에 대해 서로 다른 역할을 사용하는 것이 가장 좋습니다. 예를 들어 데이터베이스 관리자 권한이 있는 역할은 Amazon RDS 데이터베이스를 생성하고 네트워크 관리자 권한이 있는 역할은 VPC 및 보안 그룹을 설정합니다. 자습서 단계에서는 동일한 자격 증명을 계속 사용하는 것이 좋습니다.

프로덕션 환경에서 권한을 설정하는 방법에 대한 자세한 내용은 the section called “인증 및 액세스 제어” 섹션을 참조하세요.

사전 조건

이 자습서를 이해하려면 다음이 필요합니다.

- 사전 조건 A: Amazon VPC
- 사전 조건 B: Amazon EC2 인스턴스
- 사전 조건 C: Amazon RDS 데이터베이스 및 관리자 자격 증명을 위한 Secrets Manager 암호
- 사전 조건 D: 로컬 컴퓨터가 EC2 인스턴스에 연결하도록 허용

사전 조건 A: Amazon VPC

이 단계에서는 Amazon RDS 데이터베이스와 Amazon EC2 인스턴스를 시작할 수 있는 VPC를 생성합니다. 나중 단계에서는 컴퓨터를 사용하여 인터넷을 통해 Bastion에 연결한 다음 데이터베이스에 연결하게 되므로 VPC에서 나가는 트래픽을 허용해야 합니다. 이를 위해 Amazon VPC는 VPC에 인터넷 게이트웨이를 연결하고 VPC 외부로 향하는 트래픽을 인터넷 게이트웨이로 보내도록 라우팅 테이블에 경로를 추가합니다.

VPC 내에서 Secrets Manager 엔드포인트와 Amazon RDS 엔드포인트를 생성합니다. 나중 단계에서 자동 교체를 설정하면 Secrets Manager가 데이터베이스에 액세스할 수 있도록 VPC 내에 Lambda 교체 함수를 생성합니다. 또한 Lambda 교체 함수는 Secrets Manager 를 호출하여 암호를 업데이트하고 Amazon RDS를 호출하여 데이터베이스 연결 정보를 가져옵니다. VPC 내에 엔드포인트를 생성하면 Lambda 함수에서 Secrets Manager 및 Amazon RDS로의 호출이 AWS 인프라를 벗어나지 않도록 할 수 있습니다. 대신 VPC 내의 엔드포인트로 라우팅됩니다.

VPC를 생성하려면

1. https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.
2. VPC 생성을 선택합니다.
3. Create VPC(VPC 생성) 페이지에서 VPC 등을 선택합니다.
4. Name tag auto-generation(이름 태그 자동 생성)의 Auto-generate(자동 생성)에서 **`SecretsManagerTutorial`**을 입력합니다.
5. DNS options(DNS 옵션)에서 **`Enable DNS hostnames`** 및 **`Enable DNS resolution`**을 모두 선택합니다.
6. VPC 생성을 선택합니다.

VPC 내에 Secrets Manager 엔드포인트를 생성하려면

1. Amazon VPC 콘솔의 Endpoints(엔드포인트)에서 Create Endpoint(엔드포인트 생성)를 선택합니다.
2. Endpoint settings(엔드포인트 설정)에서 Name(이름)에 **`SecretsManagerTutorialEndpoint`**를 입력합니다.
3. Services(서비스)에서 **`secretsmanager`**를 입력하여 목록을 필터링한 다음 AWS 리전에서 Secrets Manager 엔드포인트를 선택합니다. 예를 들어 미국 동부(버지니아 북부)에서 `com.amazonaws.us-east-1.secretsmanager`를 선택합니다.
4. VPC에 **`vpc**** (SecretsManagerTutorial)`**를 선택합니다.

5. 서브넷(Subnets)에 가용 영역(Availability Zones)을 모두 선택한 다음 각각에 대해 포함할 서브넷 ID(Subnet ID)를 선택합니다.
6. IP address type(IP 주소 유형)에서 **IPv4**를 선택합니다.
7. 보안 그룹(Security groups)에서 기본 보안 그룹을 선택합니다.
8. 정책(Policy)에서 **Full access**를 선택합니다.
9. 엔드포인트 생성(Create endpoint)을 선택합니다.

VPC 내에 Amazon RDS 엔드포인트를 생성하려면

1. Amazon VPC 콘솔의 Endpoints(엔드포인트)에서 Create Endpoint(엔드포인트 생성)를 선택합니다.
2. Endpoint settings(엔드포인트 설정)에서 Name(이름)에 **RDSTutorialEndpoint**를 입력합니다.
3. Services(서비스)에서 **rds**를 입력하여 목록을 필터링한 다음 AWS 리전에서 Amazon RDS 엔드포인트를 선택합니다. 예를 들어 미국 동부(버지니아 북부)에서 `com.amazonaws.us-east-1.rds`를 선택합니다.
4. VPC에 **vpc**** (SecretsManagerTutorial)**를 선택합니다.
5. 서브넷(Subnets)에 가용 영역(Availability Zones)을 모두 선택한 다음 각각에 대해 포함할 서브넷 ID(Subnet ID)를 선택합니다.
6. IP address type(IP 주소 유형)에서 **IPv4**를 선택합니다.
7. 보안 그룹(Security groups)에서 기본 보안 그룹을 선택합니다.
8. 정책(Policy)에서 **Full access**를 선택합니다.
9. 엔드포인트 생성(Create endpoint)을 선택합니다.

사전 조건 B: Amazon EC2 인스턴스

나중 단계에서 생성하는 Amazon RDS 데이터베이스는 VPC에 있게 되므로 여기에 액세스하려면 Bastion Host가 필요합니다. Bastion Host는 VPC에도 있지만 나중 단계에서 SSH를 사용하여 로컬 컴퓨터가 Bastion Host에 연결할 수 있도록 보안 그룹을 구성합니다.

Bastion Host에 대한 EC2 인스턴스를 생성하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 인스턴스(Instances)를 선택한 다음 인스턴스 시작(Launch Instances)을 선택합니다.

3. Name and tags(이름 및 태그) 아래의 Name(이름)에 **SecretsManagerTutorialInstance**을 입력하세요.
4. Application and OS Images(애플리케이션 및 OS 이미지)에서 기본값 **Amazon Linux 2 AMI (HMV) Kernel 5.10**을 유지합니다.
5. Instance type(인스턴스 유형)에서 기본값 **t2.micro**를 유지합니다.
6. Key pair(키 페어)에서 Create key pair(키 페어 생성)를 선택합니다.

   Create key pair(키 페어 생성) 대화 상자에서 Key pair name(키 페어 이름)에 **SecretsManagerTutorialKeyPair**를 입력한 다음 Create key pair(키 페어 생성)를 선택합니다.

   키 페어가 자동으로 다운로드됩니다.
7. Network settings(네트워크 설정)에서 Edit(편집)를 선택하고 다음을 수행합니다.
   a. VPC에 **vpc-**** SecretsManagerTutorial**를 선택합니다.
   b. 퍼블릭 IP 자동 할당(Auto-assign Public IP)에서 **Enable**을 선택합니다.
   c. Firewall(방화벽)에서 Select existing security group(기존 보안 그룹 선택)을 선택합니다.
   d. Common security groups(일반 보안 그룹)에서 **default**를 선택합니다.
8. 인스턴스 시작을 선택합니다.

사전 조건 C: Amazon RDS 데이터베이스 및 관리자 자격 증명을 위한 Secrets Manager 암호

이 단계에서는 Amazon RDS MySQL 데이터베이스를 생성하고 Amazon RDS가 관리자 자격 증명을 포함할 암호를 생성하도록 구성합니다. 그러면 Amazon RDS에서 관리자 암호의 교체를 자동으로 관리합니다. 자세한 내용은 관리형 교체 단원을 참조하십시오.

데이터베이스 생성 과정의 일환으로 이전 단계에서 생성한 Bastion Host를 지정합니다. 그러면 Amazon RDS에서 데이터베이스와 인스턴스가 서로 액세스할 수 있도록 보안 그룹을 설정합니다. 로컬 컴퓨터도 연결할 수 있도록 인스턴스에 연결된 보안 그룹에 규칙을 추가합니다.

관리자 자격 증명이 포함된 Secrets Manager 암호를 사용하여 Amazon RDS 데이터베이스를 생성하려면

1. Amazon RDS 콘솔에서 Create database(데이터베이스 생성)를 선택합니다.
2. Engine options(엔진 옵션) 섹션의 Engine type(엔진 유형)에서 **MySQL**을 선택합니다.

3. Templates(템플릿) 섹션에서 **Free tier**를 선택합니다.
4. Settings(설정) 섹션에서 다음을 수행합니다.
   a. DB instance identifier(DB 인스턴스 식별자)에 **SecretsManagerTutorial**을 입력합니다.
   b. 자격 증명 설정에서 마스터 자격 증명 관리를 AWS Secrets Manager 선택합니다.
5. Connectivity(연결성)의 Computer resource(컴퓨터 리소스)에서 Connect to an EC2 computer resource(EC2 컴퓨터 리소스에 연결)를 선택한 다음 EC2 Instance(EC2 인스턴스)에서 **SecretsManagerTutorialInstance**를 선택합니다.
6. 데이터베이스 생성을 선택합니다.

사전 조건 D: 로컬 컴퓨터가 EC2 인스턴스에 연결하도록 허용

이 단계에서는 사전 조건 B에서 생성한 EC2 인스턴스를 구성하여 로컬 컴퓨터가 해당 인스턴스에 연결할 수 있도록 합니다. 이렇게 하려면 컴퓨터의 IP 주소가 SSH와 연결할 수 있도록 하는 규칙을 포함하도록 Amazon RDS가 사전 조건 C에 추가한 보안 그룹을 편집해야 합니다. 이 규칙을 사용하면 로컬 컴퓨터(현재 IP 주소로 식별됨)가 인터넷을 통해 SSH를 사용하여 Bastion Host에 연결할 수 있습니다.

로컬 컴퓨터가 EC2 인스턴스에 연결하도록 허용하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. EC2 인스턴스 SecretsManagerTutorialInstance의 Security(보안) 탭에 있는 Security groups(보안 그룹)에서 **sg-*** (ec2-rds-X)**를 선택합니다.
3. Input rules(입력 규칙)에서 Edit inbound rules(인바운드 규칙 편집)를 선택합니다.
4. Add rule(규칙 추가)을 선택하고 규칙에 대해 다음을 수행합니다.
   a. Type(유형)에서 **SSH**을 선택합니다.
   b. Source type(소스 유형)에 **My IP**를 선택합니다.

1단계: Amazon RDS 데이터베이스 사용자 생성

먼저 보안 암호에 자격 증명을 저장할 사용자가 필요합니다. 사용자를 생성하려면 관리자 자격 증명으로 Amazon RDS 데이터베이스에 로그인합니다. 간소화를 위해 자습서에서는 데이터베이스에 대한 전체 권한을 가진 사용자를 생성합니다. 프로덕션 환경에서 이는 일반적이지 않으므로 최소 권한 원칙을 따르는 것이 좋습니다.

데이터베이스에 연결하려면 MySQL 클라이언트 도구를 사용합니다. 이 자습서에서는 GUI 기반 애플리케이션인 MySQL Workbench를 사용합니다. MySQL Workbench를 설치하려면 MySQL Workbench 다운로드를 참조하세요.

데이터베이스에 연결하려면 MySQL Workbench에서 연결 구성을 생성합니다. 구성을 위해 Amazon EC2 및 Amazon RDS의 일부 정보가 필요합니다.

MySQL Workbench에서 데이터베이스 연결을 생성하려면

1. MySQL Workbench에서 MySQL 연결(MySQL Connections) 옆에 있는 (+) 버튼을 선택합니다.
2. 새 연결 설정(Setup New Connection) 대화 상자에서 다음을 수행합니다.
   a. 연결 이름(Connection Name)에 **SecretsManagerTutorial**을 입력합니다.
   b. 연결 방법(Connection Method)에서 **Standard TCP/IP over SSH**를 선택합니다.
   c. 파라미터(Parameters) 탭에서 다음을 수행합니다.
      i. SSH 호스트 이름(SSH Hostname)에 Amazon EC2 인스턴스의 퍼블릭 IP 주소를 입력합니다.

         인스턴스 SecretsManagerTutorialInstance를 선택하여 Amazon EC2 콘솔에서 IP 주소를 찾을 수 있습니다. Public IPv4 DNS 아래에 IP 주소를 복사합니다.
      ii. SSH 사용자 이름(SSH Username)에 **ec2-user**를 입력합니다.
      iii. SSH 키파일(SSH Keyfile)에서 이전 사전 조건에서 다운로드한 키 페어 파일 SecretsManagerTutorialKeyPair.pem을 선택합니다.
      iv. MySQL 호스트 이름(MySQL Hostname)에 Amazon RDS 엔드포인트 주소를 입력합니다.

         데이터베이스 인스턴스 secretsmanagertutorialdb를 선택하여 Amazon RDS 콘솔에서 엔드포인트 주소를 찾을 수 있습니다. 엔드포인트(Endpoint) 아래에 주소를 복사합니다.
      v. 사용자 이름(Username)에 **admin**을 입력합니다.
   d. 확인(OK)을 선택합니다.

관리자 암호를 검색하려면

1. Amazon RDS 콘솔에서 데이터베이스로 이동합니다.
2. Configuration(구성) 탭의 Master Credentials ARN(마스터 자격 증명 ARN)에서 Manage in Secrets Manager(Secrets Manager에서 관리)를 선택합니다.

Secrets Manager 콘솔이 열립니다.

3. 보안 암호 세부 정보 페이지에서 Retrieve secret value(보안 암호 값 검색)를 선택합니다.
4. 암호가 Secret value(암호 값) 섹션에 표시됩니다.

데이터베이스 사용자를 생성하려면

1. MySQL Workbench에서 SecretsManagerTutorial 연결을 선택합니다.
2. 암호에서 검색한 관리자 암호를 입력합니다.
3. MySQL Workbench의 Query(쿼리) 창에서 다음 명령(강력한 암호 포함)을 입력한 다음 Execute(실행)를 선택합니다. 교체 함수는 SELECT를 사용하여 업데이트된 보안 암호를 테스트하므로, **appuser**에는 최소한 해당 권한이 있어야 합니다.

```
CREATE DATABASE myDB;
CREATE USER 'appuser'@'%' IDENTIFIED BY 'EXAMPLE-PASSWORD';
GRANT SELECT ON myDB . * TO 'appuser'@'%';
```

출력(Output) 창에서 명령이 성공한 것을 볼 수 있습니다.

2단계: 사용자 자격 증명에 대한 보안 암호 생성

다음으로, 방금 생성한 사용자의 자격 증명을 저장하는 보안 암호를 생성합니다. 이것이 교체하게 될 보안 암호입니다. 자동 교체를 설정하고 대체 사용자 전략을 나타내려면 첫 번째 사용자의 암호를 변경할 권한이 있는 별도의 슈퍼 사용자 보안 암호를 선택합니다.

1. https://console.aws.amazon.com/secretsmanager/에서 Secrets Manager 콘솔을 엽니다.
2. Store a new secret(새 보안 암호 저장)을 선택합니다.
3. 보안 암호 유형 선택(Choose secret type) 페이지에서 다음을 수행합니다.
   a. 보안 암호 유형(Secret type)에서 Amazon RDS 데이터베이스에 대한 자격 증명(Credentials for Amazon RDS database)을 선택합니다.
   b. 자격 증명(Credentials)에서 사용자 이름 **appuser**와 MySQL Workbench를 사용하여 생성한 데이터베이스 사용자에 대해 입력한 암호를 입력합니다.
   c. 데이터베이스(Database)에서 secretsmanagertutorialdb를 선택합니다.
   d. 다음을 선택합니다.

4. 보안 암호 구성(Configure secret) 페이지에서 보안 암호 이름(Secret name)에 **SecretsManagerTutorialAppuser**를 입력한 후 다음(Next)을 선택합니다.
5. 교체 구성(Configure rotation) 페이지에서 다음을 수행합니다.

   a. Automatic rotation(자동 교체)을 켭니다.

   b. 교체 일정(Rotation schedule)에서 일(Days): **2**일, 기간(Duration): **2h**의 일정을 설정합니다. 즉시 교체(Rotate immediately)를 선택합니다.

   c. 교체 함수(Rotation function)에서 교체 함수 생성(Create a rotation function)을 선택한 다음 함수 이름에 **tutorial-alternating-users-rotation**을 입력합니다.

   d. 교체 전략에서 대체 사용자를 선택한 다음 관리자 보안 인증 정보 암호에서 이 자습서에서 생성한 데이터베이스의 이름 **secretsmanagertutorial**을 포함하는 설명이 있는(예: `Secret associated with primary RDS DB instance: arn:aws:rds:`*`Region`*`:`*`AccountId`*`:db:secretsmanagertutorial`) rds!cluster...라는 이름의 암호를 선택합니다.

   e. 다음을 선택합니다.
6. 검토(Review) 페이지에서 시작(Store)을 선택합니다.

   Secrets Manager 보안 암호 세부 정보 페이지로 돌아갑니다. 페이지 상단에서 교체 구성 상태를 확인할 수 있습니다. Secrets Manager는 CloudFormation을 사용하여 Lambda 교체 함수 및 Lambda 함수를 실행하는 실행 역할과 같은 리소스를 생성합니다. CloudFormation이 완료되면 배너가 보안 암호 교체 예약됨(Secret scheduled for rotation)으로 변경됩니다. 첫 번째 교체가 완료되었습니다.

3단계: 교체된 보안 암호 테스트

이제 보안 암호가 교체되었으므로 보안 암호에 유효한 새 자격 증명이 포함되어 있는지 확인할 수 있습니다. 보안 암호의 암호가 원래 자격 증명에서 변경되었습니다.

보안 암호에서 새 암호를 검색하려면

1. https://console.aws.amazon.com/secretsmanager/에서 Secrets Manager 콘솔을 엽니다.
2. 보안 암호(Secrets)를 선택한 다음 보안 암호 **SecretsManagerTutorialAppuser**를 선택합니다.
3. 보안 암호 세부 정보(Secret details) 페이지를 아래로 스크롤하고 보안 암호 값 검색(Retrieve secret value)을 선택합니다.
4. 키/값(Key/value) 테이블에서 **password**에 대한 보안 암호 값(Secret value)을 복사합니다.

자격 증명을 테스트하려면

1. MySQL Workbench에서 SecretsManagerTutorial 연결을 마우스 오른쪽 버튼으로 클릭한 다음 연결 편집(Edit Connection)을 선택합니다.
2. 서버 연결 관리(Manage Server Connections) 대화 상자에서 사용자 이름(Username)에 **appuser**를 입력한 다음 닫기(Close)를 선택합니다.
3. MySQL Workbench로 돌아가서 SecretsManagerTutorial 연결을 선택합니다.
4. SSH 연결 열기(Open SSH Connection) 대화 상자에서 암호(Password)에 보안 암호에서 검색한 암호를 붙여넣은 다음 확인(OK)을 선택합니다.

   자격 증명이 유효한 경우 MySQL Workbench가 데이터베이스의 디자인 페이지에 열립니다.

이는 보안 암호 교체가 성공했음을 나타냅니다. 보안 암호의 자격 증명이 업데이트되었으며 데이터베이스에 연결할 수 있는 유효한 암호입니다.

4단계: 리소스 정리

다른 교체 전략인 single user rotation(단일 사용자 교체)을 시도하려는 경우 리소스 정리를 건너뛰고 the section called “단일 사용자 교체”로 이동합니다.

그렇지 않으면 잠재적 요금이 부과되지 않도록 하고 인터넷에 액세스할 수 있는 EC2 인스턴스를 제거하기 위해 이 자습서에서 생성한 다음 리소스와 그 사전 조건을 삭제합니다.

- Amazon RDS 데이터베이스 인스턴스. 자세한 내용은 Amazon RDS 사용 설명서의 DB 인스턴스 삭제를 참조하세요.
- Amazon EC2 인스턴스. 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스 종료를 참조하세요.
- Secrets Manager 보안 암호 `SecretsManagerTutorialAppuser` 지침은 the section called “보안 암호 삭제” 섹션을 참조하세요.
- Secrets Manager 엔드포인트. 자세한 내용은AWS PrivateLink 설명서의 VPC 엔드포인트 삭제를 참조하세요.
- VPC 엔드포인트. 자세한 내용은AWS PrivateLink 설명서의 VPC 삭제를 참조하세요.

다음 단계

- 애플리케이션에서 보안 암호를 검색하는 방법을 알아봅니다.
- 다른 교체 일정에 대해 알아봅니다.

에 대한 단일 사용자 교체 설정 AWS Secrets Manager

이 자습서에서는 데이터베이스 자격 증명이 포함된 보안 암호에 대해 단일 사용자 교체를 설정하는 방법을 알아봅니다. 단일 사용자 교체는 Secrets Manager가 보안 암호 및 데이터베이스 모두에서 사용자의 자격 증명을 업데이트하는 교체 전략입니다. 자세한 내용은 the section called “단일 사용자” 단원을 참조하십시오.

자습서를 마친 후에는 자습서에서 리소스를 정리하는 것이 좋습니다. 프로덕션 환경에서 리소스를 사용하지 마세요.

Secrets Manager 교체는 AWS Lambda 함수를 사용하여 보안 암호와 데이터베이스를 업데이트합니다. Lambda 함수 사용 비용에 대한 자세한 내용은 가격 책정 섹션을 참조하세요.

목차

- 권한
- 사전 조건
- 1단계: Amazon RDS 데이터베이스 사용자 생성
- 2단계: 데이터베이스 사용자 자격 증명에 대한 보안 암호 생성
- 3단계: 교체된 암호 테스트
- 4단계: 리소스 정리
- 다음 단계

권한

자습서 사전 조건으로 AWS 계정에 대한 관리 권한이 필요합니다. 프로덕션 설정에서는 각 단계에 대해 서로 다른 역할을 사용하는 것이 가장 좋습니다. 예를 들어 데이터베이스 관리자 권한이 있는 역할은 Amazon RDS 데이터베이스를 생성하고 네트워크 관리자 권한이 있는 역할은 VPC 및 보안 그룹을 설정합니다. 자습서 단계에서는 동일한 자격 증명을 계속 사용하는 것이 좋습니다.

프로덕션 환경에서 권한을 설정하는 방법에 대한 자세한 내용은 the section called “인증 및 액세스 제어” 섹션을 참조하세요.

사전 조건

이 자습서의 사전 조건은 the section called “대체 사용자 교체”입니다. 첫 번째 자습서가 끝나면 리소스를 정리하지 마세요. 이 자습서를 마친 후에는 Amazon RDS 데이터베이스와 Secrets Manager 보안

암호가 있는 실제 환경을 갖춥니다. 데이터베이스 사용자의 자격 증명이 포함된 두 번째 암호도 있지만 이 자습서에서는 해당 암호를 사용하지 않습니다.

또한 관리자 자격 증명을 사용하여 데이터베이스에 연결하도록 MySQL Workbench에 구성되어 있습니다.

1단계: Amazon RDS 데이터베이스 사용자 생성

먼저 보안 암호에 자격 증명을 저장할 사용자가 필요합니다. 사용자를 생성하려면 암호에 저장된 관리자 자격 증명으로 Amazon RDS 데이터베이스에 로그인합니다. 간소화를 위해 자습서에서는 데이터베이스에 대한 전체 권한을 가진 사용자를 생성합니다. 프로덕션 환경에서 이는 일반적이지 않으므로 최소 권한 원칙을 따르는 것이 좋습니다.

관리자 암호를 검색하려면

1. Amazon RDS 콘솔에서 데이터베이스로 이동합니다.
2. Configuration(구성) 탭의 Master Credentials ARN(마스터 자격 증명 ARN)에서 Manage in Secrets Manager(Secrets Manager에서 관리)를 선택합니다.

   Secrets Manager 콘솔이 열립니다.
3. 보안 암호 세부 정보 페이지에서 Retrieve secret value(보안 암호 값 검색)를 선택합니다.
4. 암호가 Secret value(암호 값) 섹션에 표시됩니다.

데이터베이스 사용자를 생성하려면

1. MySQL Workbench에서 SecretsManagerTutorial 연결을 마우스 오른쪽 버튼으로 클릭한 다음 연결 편집(Edit Connection)을 선택합니다.
2. 서버 연결 관리(Manage Server Connections) 대화 상자에서 사용자 이름(Username)에 **admin**를 입력한 다음 닫기(Close)를 선택합니다.
3. MySQL Workbench로 돌아가서 SecretsManagerTutorial 연결을 선택합니다.
4. 암호에서 검색한 관리자 암호를 입력합니다.
5. MySQL Workbench의 Query(쿼리) 창에서 다음 명령(강력한 암호 포함)을 입력한 다음 Execute(실행)를 선택합니다. 교체 함수는 SELECT를 사용하여 업데이트된 보안 암호를 테스트하므로, **dbuser**에는 최소한 해당 권한이 있어야 합니다.

```
CREATE USER 'dbuser'@'%' IDENTIFIED BY 'EXAMPLE-PASSWORD';
GRANT SELECT ON myDB . * TO 'dbuser'@'%';
```

출력(Output) 창에서 명령이 성공한 것을 볼 수 있습니다.

2단계: 데이터베이스 사용자 자격 증명에 대한 보안 암호 생성

다음으로, 방금 생성한 사용자의 자격 증명을 저장하는 보안 암호를 생성하고, 즉시 교체를 포함한 자동 교체를 켭니다. Secrets Manager는 암호를 교체하므로 암호는 프로그래밍 방식으로 생성됩니다. 이 새 암호를 본 사람은 아무도 없습니다. 교체가 즉시 시작되도록 하면 교체가 올바르게 설정되었는지 확인할 수 있습니다.

1. https://console.aws.amazon.com/secretsmanager/에서 Secrets Manager 콘솔을 엽니다.
2. Store a new secret(새 보안 암호 저장)을 선택합니다.
3. 보안 암호 유형 선택(Choose secret type) 페이지에서 다음을 수행합니다.
   a. 보안 암호 유형(Secret type)에서 Amazon RDS 데이터베이스에 대한 자격 증명(Credentials for Amazon RDS database)을 선택합니다.
   b. 자격 증명(Credentials)에서 사용자 이름 **dbuser**와 MySQL Workbench를 사용하여 생성한 데이터베이스 사용자에 대해 입력한 암호를 입력합니다.
   c. 데이터베이스(Database)에서 secretsmanagertutorialdb를 선택합니다.
   d. 다음을 선택합니다.
4. 보안 암호 구성(Configure secret) 페이지에서 보안 암호 이름(Secret name)에 **SecretsManagerTutorialDbuser**를 입력한 후 다음(Next)을 선택합니다.
5. 교체 구성(Configure rotation) 페이지에서 다음을 수행합니다.
   a. Automatic rotation(자동 교체)을 켭니다.
   b. 교체 일정(Rotation schedule)에서 일(Days): **2**일, 기간(Duration): **2h**의 일정을 설정합니다. 즉시 교체(Rotate immediately)를 선택합니다.
   c. 교체 함수(Rotation function)에서 교체 함수 생성(Create a rotation function)을 선택한 다음 함수 이름에 **tutorial-single-user-rotation**을 입력합니다.
   d. 교체 전략에서 단일 사용자를 선택합니다.
   e. 다음을 선택합니다.
6. 검토(Review) 페이지에서 시작(Store)을 선택합니다.

   Secrets Manager 보안 암호 세부 정보 페이지로 돌아갑니다. 페이지 상단에서 교체 구성 상태를 확인할 수 있습니다. Secrets Manager는 CloudFormation을 사용하여 Lambda 교체 함수 및 Lambda 함수를 실행하는 실행 역할과 같은 리소스를 생성합니다. CloudFormation이 완료되면 배

너가 보안 암호 교체 예약됨(Secret scheduled for rotation)으로 변경됩니다. 첫 번째 교체가 완료되었습니다.

3단계: 교체된 암호 테스트

몇 초가 걸릴 수 있는 첫 번째 보안 암호 교체 이후 보안 암호에 유효한 자격 증명이 여전히 포함되어 있는지 확인할 수 있습니다. 보안 암호의 암호가 원래 자격 증명에서 변경되었습니다.

보안 암호에서 새 암호를 검색하려면

1. https://console.aws.amazon.com/secretsmanager/에서 Secrets Manager 콘솔을 엽니다.
2. 보안 암호(Secrets)를 선택한 다음 보안 암호 **`SecretsManagerTutorialDbuser`**를 선택합니다.
3. 보안 암호 세부 정보(Secret details) 페이지를 아래로 스크롤하고 보안 암호 값 검색(Retrieve secret value)을 선택합니다.
4. 키/값(Key/value) 테이블에서 **`password`**에 대한 보안 암호 값(Secret value)을 복사합니다.

자격 증명을 테스트하려면

1. MySQL Workbench에서 SecretsManagerTutorial 연결을 마우스 오른쪽 버튼으로 클릭한 다음 연결 편집(Edit Connection)을 선택합니다.
2. 서버 연결 관리(Manage Server Connections) 대화 상자에서 사용자 이름(Username)에 **`dbuser`**를 입력한 다음 닫기(Close)를 선택합니다.
3. MySQL Workbench로 돌아가서 SecretsManagerTutorial 연결을 선택합니다.
4. SSH 연결 열기(Open SSH Connection) 대화 상자에서 암호(Password)에 보안 암호에서 검색한 암호를 붙여넣은 다음 확인(OK)을 선택합니다.

   자격 증명이 유효한 경우 MySQL Workbench가 데이터베이스의 디자인 페이지에 열립니다.

4단계: 리소스 정리

잠재적 요금을 방지하려면 이 자습서에서 생성한 암호를 삭제합니다. 지침은 the section called “보안 암호 삭제” 섹션을 참조하세요.

이전 자습서에서 만든 리소스를 정리하려면 the section called “4단계: 리소스 정리”를 참조하세요.

다음 단계

- 애플리케이션에서 보안 암호를 검색하는 방법을 알아봅니다. 보안 암호 가져오기을(를) 참조하세요.
- 다른 교체 일정에 대해 알아봅니다. the section called “교체 일정”을(를) 참조하세요.

AWS Secrets Manager 보안 암호 생성

보안 암호는 암호, 사용자 이름 및 암호와 같은 자격 증명 집합, OAuth 토큰 또는 Secrets Manager에 암호화된 형식으로 저장하는 기타 비밀 정보일 수 있습니다.

> **Tip**
>
> Amazon RDS 및 Amazon Redshift의 마스터 사용자 보안 인증 정보를 관리하려면 관리형 보안 암호를 사용하는 것이 좋습니다. 관리 서비스를 통해 관리형 보안 암호를 생성한 다음, 관리형 교체를 사용할 수 있습니다.

콘솔을 사용하여 다른 리전으로 복제된 소스 데이터베이스에 대한 데이터베이스 보안 인증 정보를 저장하면 보안 암호에 해당 소스 데이터베이스에 대한 연결 정보가 포함됩니다. 그 후에 보안 암호를 복제하면 복제본은 소스 보안 암호의 복사본이 되며 동일한 연결 정보를 포함합니다. 리전 연결 정보를 위해 보안 암호에 추가로 키/값 쌍을 추가할 수 있습니다.

보안 암호를 만들려면 SecretsManagerReadWrite managed policy 관리형 정책에서 부여한 권한이 필요합니다.

Secrets Manager는 암호를 생성할 때 CloudTrail 로그 항목을 생성합니다. 자세한 내용은 the section called “를 사용하여 로그 AWS CloudTrail ” 단원을 참조하십시오.

보안 암호(콘솔) 생성

1. https://console.aws.amazon.com/secretsmanager/에서 Secrets Manager 콘솔을 엽니다.
2. Store a new secret(새 보안 암호 저장)을 선택합니다.
3. 보안 암호 유형 선택(Choose secret type) 페이지에서 다음을 수행합니다.

   a. 보안 암호 유형(Secret type)에서 다음 중 하나를 수행하세요.
      - 데이터베이스 자격 증명을 저장하려면 저장할 데이터베이스 자격 증명의 유형을 선택합니다. 그런 다음, 데이터베이스를 선택한 후 자격 증명을 입력합니다.
      - 데이터베이스용이 아닌 API 키, 액세스 토큰, 자격 증명을 저장하려면 다른 유형의 보안 암호를 선택합니다.

        키/값 페어에서, JSON 키/값 페어에 보안 암호를 입력하거나 일반 텍스트 탭을 클릭하고 원하는 형식으로 보안 암호를 입력합니다. 보안 암호에는 최대 65,536바이트까지 저장할 수 있습니다. 다음은 몇 가지 예제입니다.

API key

키/값 페어로 입력:

ClientID: *my_client_id*

ClientSecret : *wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY*

OAuth token

일반 텍스트로 입력:

AKIAI44QH8DHBEXAMPLE

Digital certificate

일반 텍스트로 입력:

```
-----BEGIN CERTIFICATE-----
EXAMPLE
-----END CERTIFICATE-----
```

Private key

일반 텍스트로 입력:

```
---- BEGIN PRIVATE KEY ----
EXAMPLE
---- END PRIVATE KEY ----
```

- Secrets Manager 파트너의 관리형 외부 보안 암호를 저장하려면 파트너 보안 암호를 선택합니다. 그런 다음 파트너를 선택하고 파트너의 보안 암호를 식별하는 세부 정보를 제공합니다. 자세한 내용은 AWS Secrets Manager 관리형 외부 보안 암호를 사용하여 타사 보안 암호 관리을 참조하세요.

b. 암호화 키에서 Secrets Manager AWS KMS key 가 보안 암호 값을 암호화하는 데 사용하는를 선택합니다. 자세한 내용은 보안 암호 암호화 및 복호화 단원을 참조하십시오.

- 대부분의 경우 Secrets Manager AWS 관리형 키 용 를 사용하려면 aws/secretsmanager를 선택합니다. 이 키를 사용하는 데 드는 비용은 없습니다.
- 다른에서 보안 암호에 액세스해야 AWS 계정하거나 자체 KMS 키를 사용하여 교체하거나 키 정책을 적용하려는 경우 목록에서 고객 관리형 키를 선택하고 고객 관리형 키의 키

ARN 또는 별칭 ARN을 입력하거나 새 키 추가를 선택하여 보안 암호를 생성합니다. 고객 관리형 키 사용 비용에 대한 자세한 내용은 가격 책정을 참조하세요.

the section called “KMS 키에 대한 권한”이(가) 있어야 합니다. 크로스 계정 액세스에 대한 자세한 내용은 the section called “크로스 계정 액세스” 섹션을 참조하세요.

c. 다음을 선택합니다.

4. 보안 구성(Configure secret) 페이지에서 다음을 수행합니다.

   a. 설명이 포함된 Secret name(보안 암호 이름)과 Description(설명)을 입력합니다. 보안 암호 이름에는 1~512자의 영숫자와 /_+=.@- 문자를 포함할 수 있습니다.

   b. (선택 사항) 외부 보안 암호를 생성한 경우 보안 암호를 보유한 Secrets Manager 파트너가 요구하는 메타데이터를 입력합니다.

   c. (선택 사항) Tags(태그) 섹션에서 보안 암호에 태그를 추가합니다. 태깅 전략에 대한 자세한 내용은 the section called “ 보안 암호 태그 지정” 섹션을 참조하세요. 민감한 정보는 암호화되지 않으므로 태그에 저장하지 마세요.

   d. (선택 사항) 리소스 권한(Resource permissions)에서 리소스 정책을 보안 암호에 추가하려면 권한 편집(Edit permissions)을 선택합니다. 자세한 내용은 the section called “리소스 기반 정책” 단원을 참조하십시오.

   e. (선택 사항) 보안 암호 복제에서 보안 암호를 다른에 복제하려면 보안 암호 복제를 AWS 리전 선택합니다. 보안 암호를 지금 복제하거나 페이지로 다시 돌아와서 나중에 복제할 수 있습니다. 자세한 내용은 다중 리전 복제 단원을 참조하십시오.

   f. 다음(Next)을 선택합니다.

5. (선택 사항) 교체 구성(Configure rotation) 페이지에서 자동 교체를 켤 수 있습니다. 현재 교체를 끈 다음 나중에 켤 수도 있습니다. 자세한 정보는 보안 암호 교체을(를) 참조하세요. 다음을 선택합니다.

6. 검토(Review) 페이지에서 보안 암호 세부 정보를 검토한 후 저장(Store)을 선택합니다.

   Secrets Manager는 보안 암호 목록으로 돌아갑니다. 암호가 표시되지 않으면 Refresh(새로 고침)를 선택합니다.

AWS CLI

명령 셸에 명령을 입력하면 명령 기록이 액세스되거나 유틸리티가 명령 파라미터에 액세스할 위험이 있습니다. the section called “를 사용하여 AWS Secrets Manager 보안 암호를 AWS CLI 저장할 때의 위험 완화”을(를) 참조하세요.

Example JSON 파일의 데이터베이스 자격 증명으로 보안 암호 만들기

다음 create-secret 예시에서는 파일의 자격 증명을 사용하여 시크릿을 생성합니다. 자세한 내용은 AWS CLI 사용 설명서의 파일에서 파라미터 로드를 AWS CLI 참조하세요.

Secrets Manager에서 보안 암호를 교체할 수 있게 하려면 JSON이 보안 암호의 JSON 구조에 일치해야 합니다.

```
aws secretsmanager create-secret \
    --name MyTestSecret \
    --secret-string file://mycreds.json
```

mycreds.json의 콘텐츠:

```
{
    "engine": "mysql",
    "username": "saanvis",
    "password": "EXAMPLE-PASSWORD",
    "host": "my-database-endpoint.us-west-2.rds.amazonaws.com",
    "dbname": "myDatabase",
    "port": "3306"
}
```

Example보안 암호 생성

다음 create-secret 예시에서는 두 개의 키-값 쌍으로 보안 암호를 만듭니다.

```
aws secretsmanager create-secret \
    --name MyTestSecret \
    --description "My test secret created with the CLI." \
    --secret-string '{"user":"diegor","password":"EXAMPLE-PASSWORD"}'
```

Example보안 암호 생성

다음 create-secret 예시에서는 2개의 태그로 보안 암호를 생성합니다.

```
aws secretsmanager create-secret \
    --name MyTestSecret \
    --description "My test secret created with the CLI." \
    --secret-string '{"user":"diegor","password":"EXAMPLE-PASSWORD"}'  \
    --tags '[{"Key": "FirstTag", "Value": "FirstValue"}, {"Key": "SecondTag", "Value":
 "SecondValue"}]'
```

AWS SDK

AWS SDKs 중 하나를 사용하여 보안 암호를 생성하려면 `CreateSecret` 작업을 사용합니다. 자세한 내용은 the section called “AWS SDKs” 단원을 참조하십시오.

Secrets Manager 보안 암호에는 어떤 요소가 있나요?

Secrets Manager에서 보안 암호는 보안 암호 정보, 보안 암호 값 및 보안 암호에 대한 메타데이터로 구성됩니다. 보안 암호 값은 문자열 또는 이진수일 수 있습니다.

여러 문자열 값을 한 보안 암호에 저장하려면 JSON 텍스트 문자열을 키-값 쌍으로 사용하는 것이 좋습니다. 예를 들면 다음과 같습니다.

```
{
  "host"       : "ProdServer-01.databases.example.com",
  "port"       : "8888",
  "username"   : "administrator",
  "password"   : "EXAMPLE-PASSWORD",
  "dbname"     : "MyDatabase",
  "engine"     : "mysql"
}
```

데이터베이스 보안 암호의 경우, 자동 교체를 켜려면 보안 암호에 데이터베이스에 대한 연결 정보가 올바른 JSON 구조로 포함되어야 합니다. 자세한 내용은 the section called “보안 암호의 JSON 구조” 단원을 참조하십시오.

Metadata

보안 암호의 메타데이터에는 다음이 포함됩니다.

- 다음 형식의 Amazon 리소스 이름(ARN):

  ```
  arn:aws:secretsmanager:<Region>:<AccountId>:secret:SecretName-6RandomCharacters
  ```

  Secrets Manager는 보안 암호 ARN이 고유한지 확인하기 위해 보안 암호 이름의 끝에 임의의 문자 6개를 포함합니다. 원래 보안 암호를 삭제한 후 동일한 이름으로 새 보안 암호를 생성하면 해당 문자로 인해 두 보안 암호의 ARN이 달라집니다. ARN이 다르기 때문에 이전 보안 암호에 액세스할 수 있는 사용자는 새 보안 암호에 자동으로 액세스할 수 없습니다.

- 보안 암호의 이름, 설명, 리소스 정책 및 태그입니다.
- Secrets Manager가 보안 암호 값을 암호화하고 해독 AWS KMS key 하는 데 사용하는 암호화 키의 ARN입니다. Secrets Manager는 보안 암호 텍스트를 암호화된 형식으로 저장하고, 전송 중인 보안 암호를 암호화합니다. the section called “보안 암호 암호화 및 복호화” 섹션을 참조하세요.
- 교체를 설정한 경우의 보안 암호 교체 방법에 대한 정보입니다. 보안 암호 교체을(를) 참조하세요.

Secrets Manager는 IAM 권한 정책을 사용하여 권한 있는 사용자만 보안 암호에 액세스하거나 보안 암호를 수정할 수 있도록 합니다. 에 대한 인증 및 액세스 제어 AWS Secrets Manager을(를) 참조하세요.

보안 암호에는 암호화된 보안 암호 값의 사본을 보유하는 버전이 있습니다. 보안 암호 값을 변경하거나 보안 암호를 교체할 경우 Secrets Manager는 새 버전을 만듭니다. the section called “보안 암호 버전”을(를) 참조하세요.

보안 암호를 복제 AWS 리전 하여 여러에서 사용할 수 있습니다. 보안 암호를 복제할 때 원본 또는 기본 보안 암호의 사본(복제 보안 암호)을 생성합니다. 복제 보안 암호는 기본 보안 암호에 연결된 상태로 유지됩니다. 다중 리전 복제을(를) 참조하세요.

보안 암호 관리을(를) 참조하세요.

보안 암호 버전

보안 암호에는 암호화된 보안 암호 값의 사본을 보유하는 버전이 있습니다. 보안 암호 값을 변경하거나 보안 암호를 교체할 경우 Secrets Manager는 새 버전을 만듭니다.

Secrets Manager는 보안 암호를 버전과 함께 저장하지 않습니다. 대신 다음과 같은 레이블을 지정하여 세 가지 특정 버전을 추적합니다.

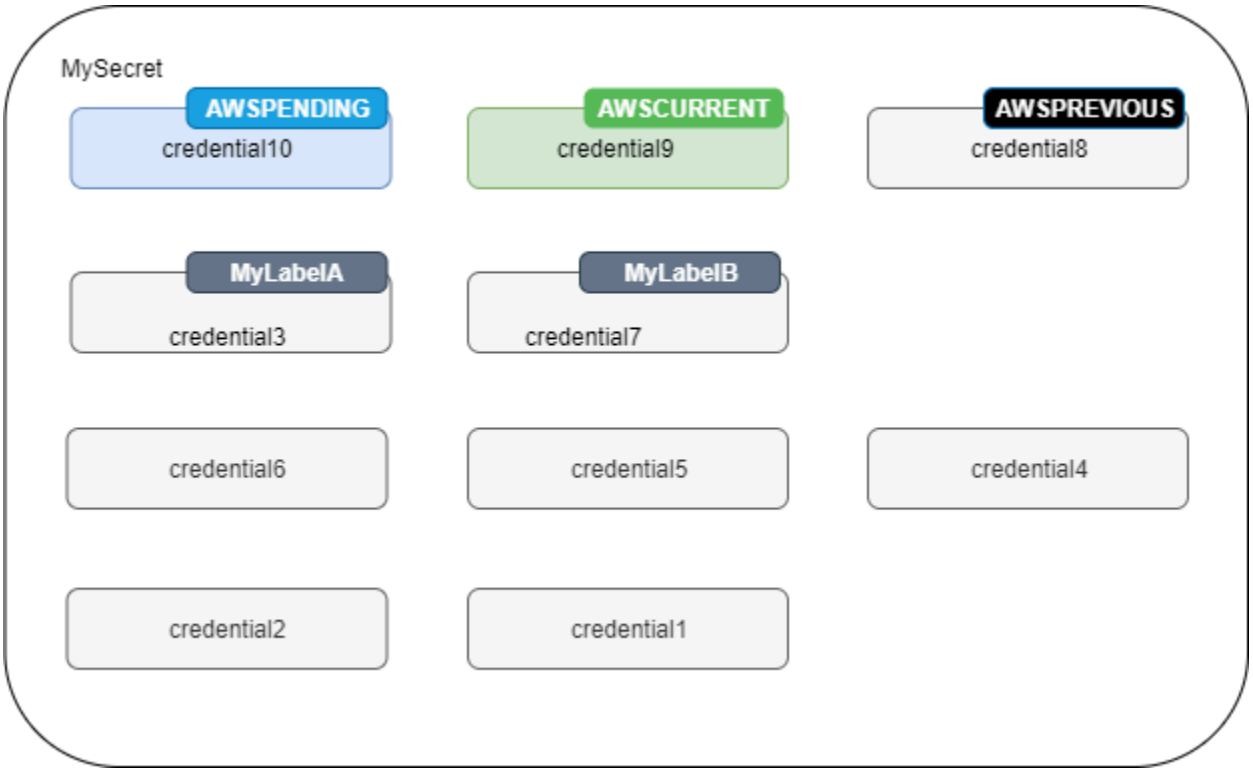
- 현재 버전 – `AWSCURRENT`
- 이전 버전 – `AWSPREVIOUS`
- 보류 버전(교체 중) – `AWSPENDING`

보안 암호에는 항상 라벨 `AWSCURRENT`이 붙은 버전이 있으며, Secrets Manager 사용자가 보안 암호 값을 검색할 때 기본적으로 해당 버전을 반환합니다.

`update-secret-version-stage`에서를 호출하여 버전에 자체 레이블을 지정할 수도 있습니다 AWS CLI. 보안 암호에는 20개까지 레이블을 지정할 수 있습니다. 보안 암호의 두 버전에는 동일한 스테이징 레이블을 지정할 수 없습니다. 버전은 라벨이 복수일 수도 있습니다.

Secrets Manager는 레이블이 지정된 버전을 제거하지 않지만 레이블이 지정되지 않은 버전은 더 이상 사용되지 않는 것으로 간주됩니다. Secrets Manager는 더 이상 사용되지 않는 보안 암호 버전이 100개를 초과하면 제거합니다. Secrets Manager는 24 시간 이내에 만든 버전을 제거하지 않습니다.

다음 그림은 AWS 레이블이 지정된 버전과 고객이 레이블이 지정된 버전이 있는 보안 암호를 보여줍니다. 레이블이 없는 버전은 더 이상 사용되지 않는 것으로 간주되며 미래에 Secrets Manager에서 제거될 예정입니다.



AWS Secrets Manager 보안 암호의 JSON 구조

Secrets Manager 보안 암호에 텍스트 또는 바이너리를 최대 크기인 65,536바이트까지 저장할 수 있습니다.

the section called “Lambda 함수로 교체”를 사용할 경우 보안 암호에는 교체 함수가 예상하는 특정 JSON 필드가 포함되어야 합니다. 예를 들어 데이터베이스 자격 증명이 포함된 보안 암호의 경우, 교체 함수가 데이터베이스에 연결하여 자격 증명을 업데이트하므로 보안 암호에 데이터베이스 연결 정보가 포함되어야 합니다.

콘솔을 사용하여 데이터베이스 보안 암호의 교체를 편집할 경우, 보안 암호에는 데이터베이스를 식별하는 특정 JSON 키-값 페어가 포함되어야 합니다. Secrets Manager는 이러한 필드를 사용해 데이터베이스를 쿼리하여 교체 함수를 저장할 올바른 VPC를 찾습니다.

JSON 키 이름은 대/소문자를 구분합니다.

주제

- Amazon RDS 및 Aurora 자격 증명
- Amazon Redshift 자격 증명
- Amazon Redshift Serverless 자격 증명
- Amazon DocumentDB 자격 증명
- Amazon Timestream for InfluxDB 보안 암호 구조
- Amazon ElastiCache 자격 증명
- Active Directory 자격 증명

Amazon RDS 및 Aurora 자격 증명

Secrets Manager에서 제공하는 교체 함수 템플릿을 사용하려면 다음과 같은 JSON 구조를 사용합니다. 더 많은 키/값 쌍을 추가할 수 있습니다. 예를 들어 다른 리전의 복제본 데이터베이스에 대한 연결 정보를 포함할 수 있습니다.

DB2

Amazon RDS Db2 인스턴스의 경우 사용자가 자신의 암호를 변경할 수 없으므로 별도의 보안 암호로 관리자 자격 증명을 제공해야 합니다.

```
{
  "engine": "db2",
  "host": "<instance host name/resolvable DNS name>",
  "username": "<username>",
  "password": "<password>",
  "dbname": "<database name. If not specified, defaults to None>",
  "port": <TCP port number. If not specified, defaults to 3306>,
  "masterarn": "<ARN of the elevated secret>",
  "dbInstanceIdentifier": <optional: ID of the instance. Alternately, use
 dbClusterIdentifier. Required for configuring rotation in the console.>",
  "dbClusterIdentifier": <optional: ID of the cluster. Alternately, use
 dbInstanceIdentifier. Required for configuring rotation in the console.>"
}
```

MariaDB

```
{
```

```
  "engine": "mariadb",
  "host": "<instance host name/resolvable DNS name>",
  "username": "<username>",
  "password": "<password>",
  "dbname": "<database name. If not specified, defaults to None>",
  "port": <TCP port number. If not specified, defaults to 3306>,
  "masterarn": "<optional: ARN of the elevated secret. Required for the the section called “## ###”.>",
  "dbInstanceIdentifier": <optional: ID of the instance. Alternately, use dbClusterIdentifier. Required for configuring rotation in the console.>",
  "dbClusterIdentifier": <optional: ID of the cluster. Alternately, use dbInstanceIdentifier. Required for configuring rotation in the console.>"
}
```

MySQL

```
{
  "engine": "mysql",
  "host": "<instance host name/resolvable DNS name>",
  "username": "<username>",
  "password": "<password>",
  "dbname": "<database name. If not specified, defaults to None>",
  "port": <TCP port number. If not specified, defaults to 3306>,
  "masterarn": "<optional: ARN of the elevated secret. Required for the the section called “## ###”.>",
  "dbInstanceIdentifier": <optional: ID of the instance. Alternately, use dbClusterIdentifier. Required for configuring rotation in the console.>",
  "dbClusterIdentifier": <optional: ID of the cluster. Alternately, use dbInstanceIdentifier. Required for configuring rotation in the console.>"
}
```

Oracle

```
{
  "engine": "oracle",
  "host": "<instance host name/resolvable DNS name>",
  "username": "<username>",
  "password": "<password>",
  "dbname": "<database name>",
  "port": <TCP port number. If not specified, defaults to 1521>,
  "masterarn": "<optional: ARN of the elevated secret. Required for the the section called “## ###”.>",
```

```
  "dbInstanceIdentifier": <optional: ID of the instance. Alternately, use
 dbClusterIdentifier. Required for configuring rotation in the console.>",
  "dbClusterIdentifier": <optional: ID of the cluster. Alternately, use
 dbInstanceIdentifier. Required for configuring rotation in the console.>"
}
```

Postgres

```
{
  "engine": "postgres",
  "host": "<instance host name/resolvable DNS name>",
  "username": "<username>",
  "password": "<password>",
  "dbname": "<database name. If not specified, defaults to 'postgres'>",
  "port": <TCP port number. If not specified, defaults to 5432>,
  "masterarn": "<optional: ARN of the elevated secret. Required for the the section
 called “## ###”.>",
  "dbInstanceIdentifier": <optional: ID of the instance. Alternately, use
 dbClusterIdentifier. Required for configuring rotation in the console.>",
  "dbClusterIdentifier": <optional: ID of the cluster. Alternately, use
 dbInstanceIdentifier. Required for configuring rotation in the console.>"
}
```

SQLServer

```
{
  "engine": "sqlserver",
  "host": "<instance host name/resolvable DNS name>",
  "username": "<username>",
  "password": "<password>",
  "dbname": "<database name. If not specified, defaults to 'master'>",
  "port": <TCP port number. If not specified, defaults to 1433>,
  "masterarn": "<optional: ARN of the elevated secret. Required for the the section
 called “## ###”.>",
  "dbInstanceIdentifier": <optional: ID of the instance. Alternately, use
 dbClusterIdentifier. Required for configuring rotation in the console.>",
  "dbClusterIdentifier": <optional: ID of the cluster.Alternately, use
 dbInstanceIdentifier.  Required for configuring rotation in the console.>"
}
```

Amazon Redshift 자격 증명

Secrets Manager에서 제공하는 교체 함수 템플릿을 사용하려면 다음과 같은 JSON 구조를 사용합니다. 더 많은 키/값 쌍을 추가할 수 있습니다. 예를 들어 다른 리전의 복제본 데이터베이스에 대한 연결 정보를 포함할 수 있습니다.

```
{
  "engine": "redshift",
  "host": "<instance host name/resolvable DNS name>",
  "username": "<username>",
  "password": "<password>",
  "dbname": "<database name. If not specified, defaults to None>",
  "dbClusterIdentifier": "<optional: database ID. Required for configuring rotation in
 the console.>"
  "port": <optional: TCP port number. If not specified, defaults to 5439>
  "masterarn": "<optional: ARN of the elevated secret. Required for the the section
 called “## ###”.>"
}
```

Amazon Redshift Serverless 자격 증명

Secrets Manager에서 제공하는 교체 함수 템플릿을 사용하려면 다음과 같은 JSON 구조를 사용합니다. 더 많은 키/값 쌍을 추가할 수 있습니다. 예를 들어 다른 리전의 복제본 데이터베이스에 대한 연결 정보를 포함할 수 있습니다.

```
{
  "engine": "redshift",
  "host": "<instance host name/resolvable DNS name>",
  "username": "<username>",
  "password": "<password>",
  "dbname": "<database name. If not specified, defaults to None>",
  "namespaceName": "<optional: namespace name, Required for configuring rotation in the
 console.> "
  "port": <optional: TCP port number. If not specified, defaults to 5439>
  "masterarn": "<optional: ARN of the elevated secret. Required for the the section
 called “## ###”.>"
}
```

Amazon DocumentDB 자격 증명

Secrets Manager에서 제공하는 교체 함수 템플릿을 사용하려면 다음과 같은 JSON 구조를 사용합니다. 더 많은 키/값 쌍을 추가할 수 있습니다. 예를 들어 다른 리전의 복제본 데이터베이스에 대한 연결 정보를 포함할 수 있습니다.

```
{
  "engine": "mongo",
  "host": "<instance host name/resolvable DNS name>",
  "username": "<username>",
  "password": "<password>",
  "dbname": "<database name. If not specified, defaults to None>",
  "port": <TCP port number. If not specified, defaults to 27017>,
  "ssl": <true|false. If not specified, defaults to false>,
  "masterarn": "<optional: ARN of the elevated secret. Required for the the section
 called “## ###”.>",
  "dbClusterIdentifier": "<optional: database cluster ID. Alternately, use
 dbInstanceIdentifier. Required for configuring rotation in the console.>"
  "dbInstanceIdentifier": "<optional: database instance ID. Alternately, use
 dbClusterIdentifier. Required for configuring rotation in the console.>"
}
```

Amazon Timestream for InfluxDB 보안 암호 구조

Timestream 보안 암호를 교체하려는 경우 the section called “Amazon Timestream for InfluxDB” 교체 템플릿을 사용할 수 있습니다.

자세한 내용은 Amazon Timestream 개발자 안내서의 How Amazon Timestream for InfluxDB uses secrets 섹션을 참조하세요.

교체 템플릿을 사용하려면 Timestream 보안 암호가 올바른 JSON 구조로 되어 있어야 합니다. 자세한 내용은 Amazon Timestream 개발자 안내서의 What's in the secret 섹션을 참조하세요.

Amazon ElastiCache 자격 증명

다음 예제에서는 ElastiCache 자격 증명을 저장하는 보안 암호의 JSON 구조를 보여줍니다.

```
{
  "password": "<password>",
  "username": "<username>"
  "user_arn": "ARN of the Amazon EC2 user"
```

```
}
```

자세한 내용은 Amazon ElastiCache 사용 설명서의 사용자의 암호 자동 교체를 참조하세요.

Active Directory 자격 증명

AWS Directory Service 는 보안 암호를 사용하여 Active Directory 자격 증명을 저장합니다. 자세한 내용은 AWS Directory Service 관리 가이드의 Seamlessly join an Amazon EC2 Linux instance to your Managed AD Active Directory 섹션을 참조하세요. 원활한 도메인 조인을 사용하려면 다음 예제의 키 이름이 필요합니다. 원활한 도메인 조인을 사용하지 않을 경우, 교체 함수 템플릿 코드에 설명된 대로 환경 변수를 사용하여 보안 암호의 키 이름을 변경할 수 있습니다.

Active Directory 보안 암호를 교체하려면 Active Directory 교체 템플릿 을 사용하면 됩니다.

Active Directory credential

```
{
  "awsSeamlessDomainUsername": "<username>",
  "awsSeamlessDomainPassword": "<password>"
}
```

보안 암호를 교체하려면 도메인 디렉터리 ID를 포함합니다.

```
{
  "awsSeamlessDomainDirectoryId": "d-12345abc6e",
  "awsSeamlessDomainUsername": "<username>",
  "awsSeamlessDomainPassword": "<password>"
}
```

보안 암호를 keytab이 포함된 보안 암호와 함께 사용하는 경우에는 keytab 보안 암호 ARN을 포함합니다.

```
{
  "awsSeamlessDomainDirectoryId": "d-12345abc6e",
  "awsSeamlessDomainUsername": "<username>",
  "awsSeamlessDomainPassword": "<password>",
  "directoryServiceSecretVersion": 1,
  "schemaVersion": "1.0",
  "keytabArns": [
    "<ARN of child keytab secret 1>,
    "<ARN of child keytab secret 2>,
```

```
    "<ARN of child keytab secret 3>",
  ],
  "lastModifiedDateTime": "2021-07-19 17:06:58"
}
```

Active Directory keytab

keytab 파일을 사용하여 Amazon EC2의 Active Directory 계정에 인증하는 방법에 대한 내용은 Deploying and configuring Active Directory authentication with SQL Server 2017 on Amazon Linux 2 문서를 참조하세요.

```
{
  "awsSeamlessDomainDirectoryId": "d-12345abc6e",
  "schemaVersion": "1.0",
  "name": "< name>",
  "principals": [
    "aduser@MY.EXAMPLE.COM",
    "MSSQLSvc/test:1433@MY.EXAMPLE.COM"
  ],
  "keytabContents": "<keytab>",
  "parentSecretArn": "<ARN of parent secret>",
  "lastModifiedDateTime": "2021-07-19 17:06:58"
  "version": 1
}
```

를 사용하여 보안 암호 관리 AWS Secrets Manager

주제

- AWS Secrets Manager 보안 암호 값 업데이트
- Secrets Manager를 사용하여 암호 생성
- 보안 암호를 이전 버전으로 롤백
- AWS Secrets Manager 보안 암호의 암호화 키 변경
- AWS Secrets Manager 보안 암호 수정
- 에서 보안 암호 찾기 AWS Secrets Manager
- AWS Secrets Manager 보안 암호 삭제
- AWS Secrets Manager 보안 암호 복원
- 에서 보안 암호 태그 지정 AWS Secrets Manager

AWS Secrets Manager 보안 암호 값 업데이트

콘솔, CLI 또는 SDK를 사용하여 보안 암호 값을 업데이트할 수 있습니다. 보안 암호 값을 업데이트하면 Secrets Manager는 스테이징 레이블 `AWSCURRENT`을(를) 사용하여 보안 암호의 새 버전을 생성합니다. 레이블 `AWSPREVIOUS`이(가) 있는 이전 버전에는 계속 액세스할 수 있습니다. 자체 레이블을 추가할 수도 있습니다. 자세한 내용은 Secrets Manager 버저닝을 참조하세요.

보안 암호 값 업데이트(콘솔)

1. https://console.aws.amazon.com/secretsmanager/에서 Secrets Manager 콘솔을 엽니다.
2. 보안 암호 목록에서 보안 암호를 선택합니다.
3. 보안 암호 세부 정보 페이지의 개요 탭에 있는 보안 암호 값 섹션에서 보안 암호 값 검색을 선택한 다음 편집을 선택합니다.

AWS CLI

보안 암호 값 업데이트(AWS CLI)

- 명령 셸에 명령을 입력하면 명령 기록이 액세스되거나 유틸리티가 명령 파라미터에 액세스할 위험이 있습니다. the section called “를 사용하여 AWS Secrets Manager 보안 암호를 AWS CLI 저장할 때의 위험 완화”을(를) 참조하세요.

  다음 put-secret-value에서는 두 개의 키-값 쌍으로 새 버전의 보안 암호를 만듭니다.

  ```
  aws secretsmanager put-secret-value \
        --secret-id MyTestSecret \
        --secret-string "{\"user\":\"diegor\",\"password\":\"EXAMPLE-PASSWORD\"}"
  ```

  다음 put-secret-value은(는) 사용자 지정 스테이징 레이블이 있는 새 버전을 생성합니다. 새 버전에는 MyLabel 및 AWSCURRENT 레이블이 있습니다.

  ```
  aws secretsmanager put-secret-value \
        --secret-id MyTestSecret \
        --secret-string "{\"user\":\"diegor\",\"password\":\"EXAMPLE-PASSWORD\"}"
        --version-stages "MyLabel"
  ```

AWS SDK

10분마다 두 번 이상 지속적으로 PutSecretValue 또는 UpdateSecret을 호출하지 않는 것이 좋습니다. PutSecretValue 또는 UpdateSecret을 호출하여 보안 암호 값을 업데이트하면 Secrets Manager는 새 버전의 보안 암호를 생성합니다. Secrets Manager는 100개를 넘는 버전이 있을 때 레이블이 지정되지 않은 버전을 제거하지만 24시간 이내에 생성된 버전은 제거하지 않습니다. 10분마다 두 번 이상 보안 암호 값을 업데이트하면 Secrets Manager에서 제거하는 버전보다 더 많은 버전이 생성되고 보안 암호 버전 할당량에 도달하게 됩니다.

보안 암호 값을 업데이트하려면 UpdateSecret 또는 PutSecretValue을(를) 사용하세요. 자세한 내용은 the section called “AWS SDKs” 단원을 참조하십시오.

Secrets Manager를 사용하여 암호 생성

Secrets Manager를 사용하는 일반적인 패턴은 Secrets Manager에서 암호를 생성한 다음, 데이터베이스 또는 서비스에서 해당 암호를 사용하는 것입니다. 다음과 같은 메서드를 사용하여 이를 수행할 수 있습니다.

- CloudFormation - 단원을 참조하십시오CloudFormation.
- AWS CLI - 단원을 참조하십시오`get-random-password`.
- AWS SDKs- 단원을 참조하십시오`GetRandomPassword`.

보안 암호를 이전 버전으로 롤백

AWS CLI를 사용해 보안 암호 버전에 연결된 레이블을 이동하여 보안 암호를 이전 버전으로 되돌릴 수 있습니다. Secrets Manager가 보안 암호의 버전을 저장하는 방법에 대한 내용은 the section called “보안 암호 버전” 섹션을 참조하세요.

다음 `update-secret-version-stage` 예제에서는 AWSCURRENT 스테이징 레이블을 이전 버전의 보안 암호로 이동합니다. 이를 통해 보안 암호를 이전 버전으로 되돌립니다. 이전 버전의 ID를 찾으려면 Secrets Manager 콘솔에서 `list-secret-version-ids`를 사용하거나 버전을 확인합니다.

이 예제에서 AWSCURRENT 레이블이 있는 버전은 a1b2c3d4-5678-90ab-cdef-EXAMPLE11111이고, AWSPREVIOUS 레이블이 있는 버전은 a1b2c3d4-5678-90ab-cdef-EXAMPLE22222입니다. 이 예제에서는 AWSCURRENT 레이블을 버전 11111에서 22222로 이동합니다. AWSCURRENT 레이블이 버전에서 제거되므로, `update-secret-version-stage`는 AWSPREVIOUS 레이블을 해당 버전(11111)으로 자동으로 이동합니다. 그 결과, AWSCURRENT 및 AWSPREVIOUS 버전이 서로 교체됩니다.

```
aws secretsmanager update-secret-version-stage \
  --secret-id MyTestSecret \
  --version-stage AWSCURRENT \
  --move-to-version-id a1b2c3d4-5678-90ab-cdef-EXAMPLE22222 \
  --remove-from-version-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

AWS Secrets Manager 보안 암호의 암호화 키 변경

Secrets Manager는 AWS KMS 키와 데이터 키를 사용한 봉투 암호화를 사용하여 각 보안 암호 값을 보호합니다. 각 보안 암호에 사용할 KMS 키를 선택할 수 있습니다. 를 사용하거나 고객 관리형 키를 사

용할 AWS 관리형 키 aws/secretsmanager수 있습니다. 대부분의 경우 aws/secretsmanager 사용을 권장하며, 이를 사용하는 데 드는 비용은 없습니다. 다른에서 보안 암호에 액세스해야 AWS 계정하거나 교체하거나 키 정책을 적용할 수 있도록 자체 KMS 키를 사용하려면를 사용합니다 고객 관리형 키. the section called “KMS 키에 대한 권한”이(가) 있어야 합니다. 고객 관리형 키 사용 비용에 대한 자세한 내용은 가격 책정을 참조하세요.

보안 암호에 대한 암호화 키를 변경할 수 있습니다. 예를 들어 다른 계정에서 보안 암호에 액세스하려는 경우 현재 AWS 관리형 키를 사용하여 보안 암호가 암호화된 `aws/secretsmanager`경우 로 전환할 수 있습니다 고객 관리형 키.

> **Tip**
>
> 를 교체하려면 AWS KMS 자동 키 교체를 사용하는 고객 관리형 키것이 좋습니다. 자세한 내용은 AWS KMS 키 교체를 참조하세요.

암호화 키를 변경하면 Secrets Manager가 새 키로 `AWSCURRENT`, `AWSPENDING`, `AWSPREVIOUS` 버전을 다시 암호화합니다. 보안 암호가 잠기는 것을 방지하기 위해 Secrets Manager는 모든 기존 버전을 이전 키로 암호화합니다. 즉, 이전 키 또는 새 키를 사용하여 `AWSCURRENT`, `AWSPENDING`, `AWSPREVIOUS` 버전을 복호화할 수 있습니다. 이전 키에 대한 `kms:Decrypt` 권한이 없는 경우, 암호화 키를 변경하면 Secrets Manager는 보안 암호 버전을 복호화하여 이를 다시 암호화할 수 없습니다. 이 경우 기존 버전은 다시 암호화되지 않습니다.

`AWSCURRENT`가 새 암호화 키로만 복호화할 수 있도록 하려면 새 키로 새 버전의 보안 암호를 생성합니다. 그런 다음, `AWSCURRENT` 보안 암호 버전을 복호화하려면 새 키에 대한 권한이 있어야 합니다.

이전 암호화 키를 비활성화하면 `AWSCURRENT`, `AWSPENDING` 및 `AWSPREVIOUS`을(를) 제외한 암호 버전을 해독할 수 없습니다. 레이블이 지정된 다른 보안 암호 버전에 계속 액세스하려는 경우 the section called “AWS CLI”을(를) 사용하여 새 암호화 키로 해당 버전을 다시 생성해야 합니다.

보안 암호에 대한 암호화 키 변경(콘솔)

1. https://console.aws.amazon.com/secretsmanager/에서 Secrets Manager 콘솔을 엽니다.
2. 보안 암호 목록에서 보안 암호를 선택합니다.
3. 보안 암호 세부 정보 페이지의 보안 암호 세부 정보 섹션에서 조치를 선택한 후 암호화 키 편집을 선택합니다.

AWS CLI

보안 암호의 암호화 키를 변경한 다음 이전 암호화 키를 비활성화하면 AWSCURRENT, AWSPENDING 및 AWSPREVIOUS을(를) 제외한 보안 암호 버전을 해독할 수 없습니다. 레이블이 지정된 다른 보안 암호 버전에 계속 액세스하려는 경우 the section called “AWS CLI”을(를) 사용하여 새 암호화 키로 해당 버전을 다시 생성해야 합니다.

보안 암호에 대한 암호화 키 변경(AWS CLI)

1. 다음 update-secret 예시에서는 시크릿 값을 암호화하는 데 사용되는 KMS 키를 업데이트합니다. KMS 키는 보안 암호와 동일한 리전에 있어야 합니다.

```
aws secretsmanager update-secret \
      --secret-id MyTestSecret \
      --kms-key-id arn:aws:kms:us-west-2:123456789012:key/EXAMPLE1-90ab-cdef-fedc-
ba987EXAMPLE
```

2. (선택 사항) 사용자 지정 레이블이 있는 보안 암호 버전이 있는 경우 새 키를 사용하여 다시 암호화하려면 해당 버전을 다시 만들어야 합니다.

   명령 셸에 명령을 입력하면 명령 기록이 액세스되거나 유틸리티가 명령 파라미터에 액세스할 위험이 있습니다. the section called “를 사용하여 AWS Secrets Manager 보안 암호를 AWS CLI 저장할 때의 위험 완화”을(를) 참조하세요.

   a. 보안 암호 버전의 값을 가져옵니다.

```
aws secretsmanager get-secret-value \
      --secret-id MyTestSecret \
      --version-stage MyCustomLabel
```

      보안 암호 값을 기록해 둡니다.

   b. 해당 값으로 새 버전을 생성합니다.

```
aws secretsmanager put-secret-value \
    --secret-id testDescriptionUpdate \
    --secret-string "SecretValue" \
    --version-stages "MyCustomLabel"
```

AWS Secrets Manager 보안 암호 수정

보안 암호를 만든 사람에 따라 보안 암호가 생성된 후에 보안 암호의 메타데이터를 수정할 수 있습니다. 다른 서비스에서 생성한 보안 암호의 경우 다른 서비스를 사용하여 암호를 업데이트하거나 교체해야 할 수 있습니다.

보안 암호 관리자를 결정하기 위해 보안 암호 이름을 검토할 수 있습니다. 다른 서비스에서 관리하는 보안 암호는 해당 서비스의 ID가 접두사로 붙습니다. 또는에서 describe-secret을 AWS CLI호출한 다음 필드를 검토합니다`OwningService`. 자세한 내용은 다른 서비스에서 관리하는 보안 암호 단원을 참조하십시오.

관리하는 보안 암호에 대해 설명, 리소스 기반 정책, 암호화 키, 태그를 수정할 수 있습니다. 암호화된 보안 암호 값을 변경할 수도 있지만 교체를 사용하여 자격 증명을 포함하는 보안 암호 값을 업데이트하는 것이 좋습니다. 교체를 하면 Secrets Manager의 보안 암호와 데이터베이스 또는 서비스의 자격 증명이 모두 업데이트됩니다. 이렇게 하면 클라이언트가 보안 암호 값을 요청할 때 항상 작업 중인 자격 증명 세트를 가져오도록 보안 암호가 자동으로 동기화됩니다. 자세한 내용은 보안 암호 교체 단원을 참조하십시오.

Secrets Manager는 암호를 수정할 때 CloudTrail 로그 항목을 생성합니다. 자세한 내용은 the section called “를 사용하여 로그 AWS CloudTrail ” 단원을 참조하십시오.

보안 암호(콘솔)를 업데이트하려면

1. https://console.aws.amazon.com/secretsmanager/에서 Secrets Manager 콘솔을 엽니다.
2. 보안 암호 목록에서 보안 암호를 선택합니다.
3. 보안 암호 세부 정보 페이지에서 다음 작업 중 하나를 수행합니다.

   보안 암호의 이름이나 ARN은 변경할 수 없다는 것을 유념하세요.

   - 설명을 업데이트하려면 보안 암호 세부 정보(Secrets details) 섹션에서 작업(Actions)을 선택한 후 설명 편집(Edit description)을 선택합니다.
   - 암호화 키를 업데이트하려면 the section called “보안 암호에 대한 암호화 키 변경”을(를) 참조하세요.
   - 태그를 업데이트하려면 태그 탭에서 태그 편집을 선택합니다. the section called “ 보안 암호 태그 지정”을(를) 참조하세요.
   - 보안 암호 값을 업데이트하려면 the section called “보안 암호 값 업데이트”을(를) 참조하세요.
   - 보안 암호에 대한 권한을 업데이트하려면 개요 탭에서 권한 편집을 선택합니다. the section called “리소스 기반 정책”을(를) 참조하세요.

- 보안 암호의 교체를 업데이트하려면 교체 탭에서 교체 편집을 선택합니다. 보안 암호 교체을(를) 참조하세요.
- 보안 암호를 다른 리전으로 복제하려면 다중 리전 복제 섹션을 참조하세요.
- 보안 암호에 복제본이 있는 경우 복제본에 대한 암호화 키를 변경할 수 있습니다. 복제 탭에서 복제본에 대한 라디오 버튼을 선택한 다음 작업 메뉴에서 암호화 키 편집을 선택합니다. the section called “보안 암호 암호화 및 복호화”을(를) 참조하세요.
- 보안 암호를 다른 서비스에서 관리하도록 변경하려면 해당 서비스에서 보안 암호를 다시 만들어야 합니다. 다른 서비스에서 관리하는 보안 암호을(를) 참조하세요.

AWS CLI

Example보안 암호 설명 업데이트

다음 `update-secret` 예에서는 보안 암호에 대한 설명을 업데이트합니다.

```
aws secretsmanager update-secret \
    --secret-id MyTestSecret \
    --description "This is a new description for the secret."
```

AWS SDK

10분마다 두 번 이상 지속적으로 `PutSecretValue` 또는 `UpdateSecret`을 호출하지 않는 것이 좋습니다. `PutSecretValue` 또는 `UpdateSecret`을 호출하여 보안 암호 값을 업데이트하면 Secrets Manager는 새 버전의 보안 암호를 생성합니다. Secrets Manager는 100개를 넘는 버전이 있을 때 레이블이 지정되지 않은 버전을 제거하지만 24시간 이내에 생성된 버전은 제거하지 않습니다. 10분마다 두 번 이상 보안 암호 값을 업데이트하면 Secrets Manager에서 제거하는 버전보다 더 많은 버전이 생성되고 보안 암호 버전 할당량에 도달하게 됩니다.

보안 암호를 업데이트하려면 `UpdateSecret` 또는 `ReplicateSecretToRegions`을(를) 사용하세요. 자세한 내용은 the section called “AWS SDKs” 단원을 참조하십시오.

에서 보안 암호 찾기 AWS Secrets Manager

필터 없이 보안 암호를 검색할 경우, Secrets Manager는 보안 암호 이름, 설명, 태그 키 및 태그 값의 키워드를 일치시킵니다. 필터 없는 검색은 대/소문자를 구분하지 않으며 공백, /, _, =, #과 같은 특수 문자를 무시하고 숫자와 문자만 사용합니다. 필터 없이 검색할 때 Secrets Manager는 검색 문자열을 분석

하여 별개의 단어로 변환합니다. 대문자에서 소문자로, 문자에서 숫자로, 또는 숫자/문자에서 구두점으로 단어를 바꾸어 구분합니다. 예를 들어 `credsDatabase#892`를 검색 단어로 입력하면 이름, 설명, 태그 키 및 값으로 `creds`, `Database`, `892`를 검색합니다.

Secrets Manager는 암호를 나열할 때 CloudTrail 로그 항목을 생성합니다. 자세한 내용은 the section called “를 사용하여 로그 AWS CloudTrail ” 단원을 참조하십시오.

Secrets Manager는 리전 서비스이며 선택한 리전 내부의 보안 암호만 반환됩니다.

검색 필터

필터를 사용하지 않을 경우 Secrets Manager는 검색 문자열을 단어로 나누고, 모든 속성에서 일치하는 항목을 검색합니다. 이 검색은 대/소문자를 구분하지 않습니다. 예를 들어 **`My_Secret`**를 검색하면 이름, 설명 또는 태그에 my 또는 secret이라는 단어가 포함된 보안 암호와 일치하게 됩니다.

다음 필터를 검색에 적용할 수 있습니다.

이름

보안 암호 이름의 시작 부분과 일치하고 대소문자를 구분합니다. 예를 들어 이름(Name): **`Data`**는 `DatabaseSecret`이라는 보안 암호를 반환하지만 `databaseSecret` 또는 `MyData`를 반환하지는 않습니다.

설명

보안 암호 설명의 단어와 일치하고 대소문자를 구분하지 않습니다. 예를 들어 설명(Description): **`My Description`**은 보안 암호를 다음 설명과 일치시킵니다.

- `My Description`
- `my description`
- `My basic description`
- `Description of my secret`

관리 주체

외부 서비스에서 관리하는 보안 암호를 찾습니다 AWS. 예를 들면 다음과 같습니다.

- 1Password
- Akeyless
- CyberArk
- HashiCorp

소유 서비스

대소문자를 구분 없이 관리 서비스 ID 접두사의 시작 부분과 일치합니다. 예를 들어, **my-ser**은(는) 접두사가 `my-serv` 및 `my-service`인 서비스에서 관리하는 보안 암호와 일치합니다. 자세한 내용은 다른 서비스에서 관리하는 보안 암호 단원을 참조하십시오.

복제된 보안 암호

기본 보안 암호, 복제본 보안 암호 또는 복제되지 않은 보안 암호를 필터링할 수 있습니다.

태그 키

태그 키의 시작 부분과 일치하고 대소문자를 구분합니다. 예를 들어 태그 키(Tag key): **Prod**는 태그 `Production` 및 `Prod1`이 포함된 보안 암호를 반환하지만 태그 `prod` 또는 `1 Prod`가 포함된 보안 암호는 반환하지 않습니다.

태그 값

태그 값의 시작 부분과 일치하고 대소문자를 구분합니다. 예를 들어 태그 값(Tag value): **Prod**는 태그 `Production` 및 `Prod1`이 포함된 보안 암호를 반환하지만 태그 값 `prod` 또는 `1 Prod`가 포함된 보안 암호는 반환하지 않습니다.

AWS CLI

Example계정의 보안 암호 목록

다음 `list-secrets` 예시에서는 계정에 있는 보안 암호 목록을 가져옵니다.

```
aws secretsmanager list-secrets
```

Example계정의 보안 암호 목록 필터링

다음 `list-secrets` 예시에서는 계정에서 이름에 Test가 있는 보안 암호 목록을 가져옵니다. 이름의 필터링은 대소문자를 구분합니다.

```
aws secretsmanager list-secrets \
    --filters Key="name",Values="Test"
```

Example다른 AWS 서비스에서 관리하는 보안 암호 찾기

다음 `list-secrets` 예시는 서비스에서 관리하는 보안 암호 목록을 가져옵니다. ID로 서비스를 지정합니다. 자세한 내용은 다른 서비스에서 관리하는 보안 암호 단원을 참조하십시오.

```
aws secretsmanager list-secrets \
    --filters Key="owning-service",Values="<service ID prefix>"
```

AWS SDK

AWS SDKs 중 하나를 사용하여 보안 암호를 찾으려면를 사용합니다`ListSecrets`. 자세한 내용은 the section called “AWS SDKs” 단원을 참조하십시오.

AWS Secrets Manager 보안 암호 삭제

보안 암호의 중요한 특성 때문에는 AWS Secrets Manager 의도적으로 보안 암호를 삭제하기 어렵게 만듭니다. Secrets Manager는 보안 암호를 바로 삭제하지 않습니다. 대신 Secrets Manager는 즉시 이 보안 암호에 액세스할 수 없도록 하고 최소 7일의 복구 기간 후에 삭제되도록 예약합니다. 복구 기간이 끝날 때까지 이전에 삭제한 보안 암호를 복구할 수 있습니다. 삭제하도록 표시한 보안 암호에 대해서는 요금이 부과되지 않습니다.

다른 리전에 복제된 기본 보안 암호는 삭제할 수 없습니다. 우선 복제본을 삭제한 다음 기본 보안 암호를 삭제합니다. 복제본을 삭제할 경우, 즉시 삭제됩니다.

또한 보안 암호 버전을 직접 삭제할 수 없습니다. 대신 AWS CLI 또는 AWS SDK를 사용하여 버전에서 모든 스테이징 레이블을 제거합니다. 이는 버전을 사용 중지 상태로 표시하고 Secrets Manager가 배경에서 해당 버전을 자동으로 삭제할 수 있게 합니다.

애플리케이션에서 보안 암호가 여전히 사용 중인지 모르는 경우 복구 기간 중에 보안 암호에 액세스하려고 하면 알려주는 Amazon CloudWatch 경보를 생성할 수 있습니다. 자세한 정보는 삭제 예약된 AWS Secrets Manager 보안 암호에 액세스하는 시기 모니터링을(를) 참조하세요.

보안 암호를 삭제하려면 `secretsmanager:ListSecrets` 및 `secretsmanager:DeleteSecret` 권한이 있어야 합니다.

Secrets Manager는 암호를 삭제할 때 CloudTrail 로그 항목을 생성합니다. 자세한 내용은 the section called “를 사용하여 로그 AWS CloudTrail ” 단원을 참조하십시오.

보안 암호(콘솔)를 삭제하려면

1. https://console.aws.amazon.com/secretsmanager/에서 Secrets Manager 콘솔을 엽니다.
2. 보안 암호 목록에서 삭제할 보안 암호를 선택합니다.

3. 보안 암호 세부 정보(Secret details) 섹션에서 작업(Actions)을 선택한 후 보안 암호 삭제(Delete secret)를 선택합니다.
4. 보안 암호 사용 중지 및 삭제 예약(Disable secret and schedule deletion) 대화 상자에서 대기 기간(Waiting period)에 영구적으로 삭제되기 전까지 대기할 일수를 입력합니다. Secrets Manager는 `DeletionDate`라는 필드를 연결하고 해당 필드를 현재 날짜 및 시간에, 복구 기간에 지정된 일수를 더한 값으로 설정합니다.
5. 삭제 예약(Schedule deletion)을 선택합니다.

삭제된 보안 암호를 보려면

1. https://console.aws.amazon.com/secretsmanager/에서 Secrets Manager 콘솔을 엽니다.
2. 보안 암호(Secrets) 페이지에서 기본 설정 (Preferences)()을 선택합니다.
3. 기본 설정 대화 상자에서 삭제 예정 암호 표시를 선택한 후 저장을 선택합니다.

복제본 보안 암호를 삭제하려면

1. https://console.aws.amazon.com/secretsmanager/에서 Secrets Manager 콘솔을 엽니다.
2. 기본 보안 암호를 선택합니다.
3. 보안 암호 복제(Replicate Secret) 섹션에서 복제본 보안 암호를 선택합니다.
4. 작업(Actions) 메뉴에서 복제본 삭제(Delete Replica)를 선택합니다.

AWS CLI

Example보안 암호 삭제

다음 `delete-secret` 예시에서는 보안 암호를 삭제합니다. DeletionDate 응답 필드의 날짜 및 시간까지 `restore-secret`로 보안 암호를 복구할 수 있습니다. 다른 리전에 복제된 보안 암호를 삭제하려면 먼저 `remove-regions-from-replication`(으)로 해당 복제본을 삭제한 다음 `delete-secret`을(를) 호출합니다.

```
aws secretsmanager delete-secret \
    --secret-id MyTestSecret \
    --recovery-window-in-days 7
```

Example보안 암호 즉시 삭제

다음 delete-secret 예시는 복구 기간 없이 즉시 보안 암호를 삭제합니다. 이러한 보안 암호는 복구할 수 없습니다.

```
aws secretsmanager delete-secret \
    --secret-id MyTestSecret \
    --force-delete-without-recovery
```

Example복제본 보안 암호 삭제

다음 remove-regions-from-replication 예시에서는 eu-west-3의 복제 보안 암호를 삭제합니다. 다른 리전에 복제된 기본 시크릿을 삭제하려면 먼저 복제본을 삭제한 다음 delete-secret을 직접적으로 호출합니다.

```
aws secretsmanager remove-regions-from-replication \
    --secret-id MyTestSecret \
    --remove-replica-regions eu-west-3
```

AWS SDK

보안 암호를 삭제하려면 DeleteSecret 명령을 사용합니다. 보안 암호의 버전을 삭제하려면 UpdateSecretVersionStage 명령을 사용합니다. 복제본을 삭제하려면 StopReplicationToReplica 명령을 사용합니다. 자세한 내용은 the section called “AWS SDKs” 단원을 참조하십시오.

AWS Secrets Manager 보안 암호 복원

Secrets Manager에서는 삭제 예약된 보안 암호를 더 이상 사용되지 않는 것으로 간주하며 이제 직접 액세스할 수 없습니다. 복구 기간이 지난 후 Secrets Manager는 보안 암호를 영구적으로 삭제합니다. Secrets Manager에서 보안 암호를 삭제한 후에는 복구할 수 없습니다. 복구 기간이 끝나기 전에 보안 암호를 복구하고 다시 액세스할 수 있습니다. 이렇게 하면 영구 삭제 예약을 취소하는 DeletionDate 필드가 제거됩니다.

콘솔에서 보안 암호와 메타데이터를 복원하려면 secretsmanager:ListSecrets 및 secretsmanager:RestoreSecret 권한이 있어야 합니다.

Secrets Manager는 암호를 복원할 때 CloudTrail 로그 항목을 생성합니다. 자세한 내용은 the section called “를 사용하여 로그 AWS CloudTrail ” 단원을 참조하십시오.

보안 암호(콘솔) 복원

1. https://console.aws.amazon.com/secretsmanager/에서 Secrets Manager 콘솔을 엽니다.
2. 보안 암호 목록에서 복원할 보안 암호를 선택합니다.

   삭제된 보안 암호가 보안 암호 목록에 나타나지 않는 경우 기본 설정 (Preferences)()을 선택합니다. 기본 설정 대화 상자에서 삭제 예정 암호 표시를 선택한 후 저장을 선택합니다.
3. 보안 암호 세부 정보(Secret details) 페이지에서 삭제 취소(Cancel deletion)를 선택합니다.
4. 보안 암호 삭제 취소(Cancel secret deletion) 대화 상자에서 삭제 취소(Cancel deletion)를 선택합니다.

AWS CLI

Example이전에 삭제한 보안 암호 복원하기

다음 `restore-secret` 예시에서는 이전에 삭제가 예정된 보안 암호를 복원합니다.

```
aws secretsmanager restore-secret \
    --secret-id MyTestSecret
```

AWS SDK

삭제하도록 표시된 보안 암호를 복원하려면 `RestoreSecret` 명령을 사용합니다. 자세한 내용은 the section called “AWS SDKs” 단원을 참조하십시오.

에서 보안 암호 태그 지정 AWS Secrets Manager

에서는 태그를 사용하여 보안 암호에 메타데이터를 할당 AWS Secrets Manager할 수 있습니다. 태그는 보안 암호에 대해 정의하는 키-값 쌍입니다. 태그는 AWS 리소스를 관리하고 결제 정보를 포함한 데이터를 구성하는 데 도움이 됩니다.

태그를 사용하면 다음을 수행할 수 있습니다.

- AWS 계정 내 보안 암호 및 기타 리소스를 관리, 검색, 필터링
- 연결된 태그를 기반으로 보안 암호에 대한 액세스 제어
- 특정 보안 암호나 프로젝트와 관련된 비용을 추적 및 분류

태그를 사용한 액세스 제어에 대한 자세한 내용은 the section called “태그를 사용하여 보안 암호에 대한 액세스 제어” 섹션을 참조하세요.

비용 할당 태그에 대한 자세한 내용은 AWS Billing 사용 설명서의 AWS 비용 할당 태그 사용을 참조하세요.

태그 할당량 및 이름 지정 제한은 AWS 일반 참조 가이드의 태그 지정을 위한 Service Quotas을 참조하세요. 태그는 대소문자를 구분합니다.

Secrets Manager는 암호에 태그를 지정하거나 태그를 해제할 때 CloudTrail 로그 항목을 생성합니다. 자세한 내용은 the section called “를 사용하여 로그 AWS CloudTrail ” 단원을 참조하십시오.

> **Tip**
>
> 모든 AWS 리소스에서 일관된 태그 지정 체계를 사용합니다. 모범 사례는 태그 지정 모범 사례 백서를 참조하세요.

태그 기본 사항 검토

콘솔에서 태그별로 보안 암호를 찾을 수 있으며 AWS CLI, SDKs Resource Groups 도구를 AWS 제공하여 태그를 기반으로 리소스를 통합하고 구성하는 사용자 지정 콘솔을 생성합니다. 특정 태그가 있는 보안 암호를 찾으려면 the section called “보안 암호 찾기”를 참조하세요.

Secrets Manager 콘솔 AWS CLI또는 Secrets Manager API를 사용하여 다음을 수행할 수 있습니다.

- 태그를 포함한 보안 암호 생성

- 보안 암호에 태그 추가
- 보안 암호의 태그 나열
- 보안 암호에서 태그 제거

태그를 사용하여 보안 암호를 *분류할 수 있습니다. 예를 들어 보안 암호를 용도, 소유자 또는 환경별로 분류할 수 있습니다. 각 태그에 대해 키와 값이 정의되기 때문에 특정 요구를 충족하는 사용자 지정 범주 세트를 생성할 수 있습니다. 다음은 태그의 몇 가지 예제입니다.

- `Project: Project name`
- `Owner: Name`
- `Purpose: Load testing`
- `Application: Application name`
- `Environment: Production`

태그 지정을 사용하여 비용 추적

태그를 사용하여 AWS 비용을 분류하고 추적할 수 있습니다. 보안 암호를 포함하여 AWS 리소스에 태그를 적용하면 AWS 비용 할당 보고서에는 태그별로 집계된 사용량 및 비용이 포함됩니다. 비즈니스 카테고리를 나타내는 태그(예: 비용 센터, 애플리케이션 이름 또는 소유자)를 적용하여 여러 서비스에 대한 비용을 정리할 수 있습니다. 자세한 내용은 AWS Billing 사용 설명서의 사용자 지정 결제 보고서에 비용 할당 태그 사용을 참조하세요.

태그 제한 이해

태그에 적용되는 제한은 다음과 같습니다.

기본 제한 사항

- 리소스(보안 암호)당 최대 태그 수는 50개입니다.
- 태그 키와 값은 대/소문자를 구분합니다.
- 삭제된 보안 암호에 대한 태그는 변경하거나 편집할 수 없습니다.

태그 키 제한 사항

- 각 태그 키는 고유해야 합니다. 이미 사용 중인 키를 가진 태그를 추가하면 기존 키-값 쌍에 새 태그가 덮어쓰기 됩니다.

- 이 접두사는 사용자 대신이 접두사로 시작하는 태그를 AWS AWS 생성하지만 편집하거나 삭제할 수는 aws: 없으므로 로 태그 키를 시작할 수 없습니다.
- 태그 키의 길이는 유니코드 1~128자여야 합니다.
- 태그 키의 문자로는 유니코드 문자, 숫자, 공백 그리고 _ . / = + - @ 같은 특수 문자가 허용됩니다.

태그 값 제한 사항

- 태그 값의 길이는 유니코드 0~255자여야 합니다.
- 태그 값은 공백 상태로 둘 수 있습니다. 아니면 유니코드 문자, 숫자, 공백 그리고 _ . / = + - @ 같은 특수 문자를 사용할 수 있습니다.

Secrets Manger 콘솔을 사용하여 보안 암호에 태그 지정

Secrets Manager 콘솔을 사용하여 보안 암호의 태그를 관리할 수 있습니다.

태그 지정 기능에 액세스하려면 다음을 수행합니다.

1. Secrets Manager 콘솔을 엽니다.
2. 탐색 표시줄에서 원하는 리전을 선택합니다.
3. 보안 암호 페이지에서 보안 암호를 선택합니다.

보안 암호에 대한 태그를 확인하려면

- 보안 암호 세부 정보 페이지에서 태그 탭을 선택합니다.

태그가 포함된 보안 암호를 생성하려면

- 보안 암호 생성 섹션의 단계를 따르세요.

보안 암호에 대한 태그를 추가하거나 편집하려면

1. 보안 암호 세부 정보 페이지에서 태그 탭을 선택한 다음 태그 편집을 선택합니다.
2. 키 필드에 태그 키를 입력합니다. 원할 경우 값 필드에 태그 값을 입력합니다.
3. 저장을 선택합니다. 새 태그 또는 업데이트된 태그가 태그 목록에 나타납니다.

> **Note**
>
> 저장 버튼이 활성화되지 않는 경우, 태그 키나 값이 태그 제한 조건을 충족하지 못했을 수 있습니다. 자세한 내용은 태그 제한 이해 단원을 참조하십시오.

보안 암호에서 태그를 제거하려면

1. 보안 암호 세부 정보 페이지에서 태그 탭을 선택한 다음, 제거하려는 태그 옆의 제거 아이콘을 선택합니다.
2. 저장을 선택하여 제거를 확인하거나 실행 취소를 선택하여 취소합니다.

를 사용하여 보안 암호에 태그 지정 AWS CLI

AWS CLI 예제

Example보안 암호에 태그 추가

다음 `tag-resource` 예시에서는 간편 구문으로 태그를 연결하는 방법을 보여줍니다.

```
aws secretsmanager tag-resource \
            --secret-id MyTestSecret \
            --tags Key=FirstTag,Value=FirstValue
```

Example보안 암호에 여러 태그 추가

다음 `tag-resource` 예시에서는 두 개의 키-값 태그를 보안 암호에 연결합니다.

```
aws secretsmanager tag-resource \
            --secret-id MyTestSecret \
            --tags '[{"Key": "FirstTag", "Value": "FirstValue"}, {"Key": "SecondTag",
 "Value": "SecondValue"}]'
```

Example보안 암호에서 태그 제거

다음 `untag-resource` 예시에서는 보안 암호에서 두 개의 태그를 제거합니다. 각 태그의 키와 값이 모두 제거됩니다.

```
aws secretsmanager untag-resource \
```

```
            --secret-id MyTestSecret \
            --tag-keys '[ "FirstTag", "SecondTag"]'
```

Secrets Manger API를 사용하여 보안 암호에 태그 지정

Secrets Manager API를 사용하면 태그를 추가, 조회 및 제거할 수 있습니다. 예제는 다음 설명서를 참조하세요.

- ListSecrets: 보안 암호에 적용된 태그를 조회하는 데 `ListSecrets` 사용
- TagResource: 보안 암호에 태그 추가
- Untag: 보안 암호에서 태그 제거

Secrets Manager AWS SDK를 사용하여 보안 암호에 태그 지정

보안 암호의 태그를 변경하려면 다음 API 작업을 사용합니다.

- ListSecrets: 보안 암호에 적용된 태그를 조회하는 데 `ListSecrets` 사용
- `TagResource`: 보안 암호에 태그 추가
- `UntagResource`: 보안 암호에서 태그 제거

SDK 사용에 대한 자세한 내용은 the section called “AWS SDKs” 섹션을 참조하세요.

리전 간 AWS Secrets Manager 보안 암호 복제

여러에 보안 암호를 복제 AWS 리전 하여 해당 리전에 분산된 애플리케이션을 지원하여 리전별 액세스 및 짧은 지연 시간 요구 사항을 충족할 수 있습니다. 나중에 필요한 경우 복제본 보안 암호를 독립 실행형으로 승격한 다음 독립적으로 복제하도록 설정할 수 있습니다. Secrets Manager는 지정된 리전에 걸쳐 태그 및 리소스 정책과 같은 암호화된 보안 암호 데이터 및 메타데이터를 복제합니다.

복제된 암호의 ARN 리전을 제외하고 기본 암호와 동일합니다. 예를 들면 다음과 같습니다.

- 기본 보안 암호: `arn:aws:secretsmanager:Region1:123456789012:secret:MySecret-a1b2c3`
- 복제본 보안 암호: `arn:aws:secretsmanager:Region2:123456789012:secret:MySecret-a1b2c3`

복제본 보안 암호에 대한 요금 정보는 AWS Secrets Manager 요금을 참조하세요.

다른 리전으로 복제된 소스 데이터베이스에 대한 데이터베이스 보안 인증 정보를 저장하면 보안 암호에 해당 소스 데이터베이스에 대한 연결 정보가 포함됩니다. 그 후에 보안 암호를 복제하면 복제본은 소스 보안 암호의 복사본이 되며 동일한 연결 정보를 포함합니다. 리전 연결 정보를 위해 보안 암호에 추가로 키/값 쌍을 추가할 수 있습니다.

기본 보안 암호에 대한 교체를 켜면 Secrets Manager가 기본 리전의 보안 암호를 교체하고 새 보안 암호 값이 연결된 모든 복제본 보안 암호에 전파됩니다. 모든 복제본 보안 암호에 대해 개별적인 교체를 관리할 필요가 없습니다.

활성화된 모든 AWS 리전에서 보안 암호를 복제할 수 있습니다. 그러나 AWS GovCloud (US) 또는 중국 AWS 리전과 같은 특수 리전에서 Secrets Manager를 사용하는 경우 이러한 특수 AWS 리전 내에서만 보안 암호와 복제본을 구성할 수 있습니다. 활성화된 AWS 리전의 보안 암호를 특수 리전에 복제하거나 특수 리전의 보안 암호를 상용 리전에 복제할 수 없습니다.

보안 암호를 다른 리전으로 복제하려면 먼저 해당 리전을 사용해야 합니다. 자세한 내용은 AWS 리전 관리를 참조하세요.

보안 암호가 저장된 리전의 Secrets Manager 엔드포인트를 호출하여 복제하지 않고 여러 리전에서 보안 암호를 사용할 수 있습니다. 엔드포인트 목록은 the section called “Secrets Manager 엔드포인트” 섹션을 참조하세요. 복제를 사용하여 워크로드의 복원력을 개선하려면 의 재해 복구(DR) 아키텍처 AWS, I부: 클라우드의 복구 전략을 참조하세요.

Secrets Manager는 암호를 복제할 때 CloudTrail 로그 항목을 생성합니다. 자세한 내용은 the section called “를 사용하여 로그 AWS CloudTrail ” 단원을 참조하십시오.

보안 암호를 다른 리전으로 복제하려면(콘솔)

1. https://console.aws.amazon.com/secretsmanager/에서 Secrets Manager 콘솔을 엽니다.
2. 보안 암호 목록에서 보안 암호를 선택합니다.
3. 보안 암호 세부 정보 페이지의 복제 탭에서 다음 중 하나를 수행합니다.
   - 보안 암호가 복제되지 않은 경우에는 보안 암호 복제(Replicate secret)를 선택합니다.
   - 보안 암호가 복제된 경우에는 보안 암호 복제(Replicate secret) 섹션에서 리전 추가(Add Region)를 선택합니다.
4. 복제본 리전 추가(Add replica regions) 대화 상자에서 다음을 수행합니다.

   a. AWS 리전(Region)에서 보안 암호를 복제하고자 하는 리전을 선택합니다.
   b. (선택 사항) 암호화 키(Encryption key)에서 보안 암호를 암호화할 KMS 키를 선택합니다. 키가 복제본 리전에 있어야 합니다.
   c. (선택 사항) 다른 리전을 추가하려면 더 많은 리전 추가(Add more regions)를 선택합니다.
   d. 복제(Replicate)를 선택합니다.

   보안 암호 세부 정보 페이지로 돌아갑니다. 보안 암호 복제(Replicate Secret) 섹션에서 각 리전의 복제 상태(Replication Status)가 표시됩니다.

AWS CLI

Example다른 리전으로 보안 암호 복제

다음 `replicate-secret-to-regions` 예시에서는 eu-west-3으로 보안 암호를 복제합니다. 복제본은 AWS 관리형 키 로 암호화됩니다aws/secretsmanager.

```
aws secretsmanager replicate-secret-to-regions \
        --secret-id MyTestSecret \
        --add-replica-regions Region=eu-west-3
```

Example보안 암호 생성 후 복제

다음 예시에서는 eu-west-3으로 보안 암호를 생성한 후 복제합니다. 복제본은 로 암호화됩니다 AWS 관리형 키 aws/secretsmanager.

```
aws secretsmanager create-secret \
    --name MyTestSecret \
    --description "My test secret created with the CLI." \
    --secret-string "{\"user\":\"diegor\",\"password\":\"EXAMPLE-PASSWORD\"}"
    --add-replica-regions Region=eu-west-3
```

AWS SDK

보안 암호를 복제하려면 `ReplicateSecretToRegions` 명령을 사용합니다. 자세한 내용은 the section called “AWS SDKs” 단원을 참조하십시오.

에서 복제본 보안 암호를 독립 실행형 보안 암호로 승격 AWS Secrets Manager

복제본 보안 암호는 다른의 기본에서 복제되는 보안 암호입니다 AWS 리전. 이 보안 암호는 기본 암호 값과 메타데이터가 동일하지만 다른 KMS 키를 사용하여 암호화할 수 있습니다. 복제본 보안 암호는 암호화 키를 제외하고 기본 보안 암호와 독립적으로 업데이트할 수 없습니다. 복제본 보안 암호를 승격할 경우 복제본 보안 암호가 기본 보안 암호에서 분리되고 복제본 보안 암호가 독립 실행형 보안 암호로 설정됩니다. 기본 보안 암호에 대한 변경 사항은 독립 실행형 보안 암호에 복제되지 않습니다.

기본 암호를 사용할 수 없게 되면 재해 복구 솔루션으로 복제본 암호를 독립 실행형 암호로 승격할 수 있습니다. 또는 복제본에 대해 교체를 켜려는 경우 복제본을 독립 실행형 보안 암호로 승격해야 합니다.

복제본을 승격한 경우, 독립 실행형 보안 암호를 사용하도록 해당 애플리케이션을 업데이트해야 합니다.

Secrets Manager는 암호의 수준을 올릴 때 CloudTrail 로그 항목을 생성합니다. 자세한 내용은 the section called “를 사용하여 로그 AWS CloudTrail ” 단원을 참조하십시오.

복제본 보안 암호를 승격하려면(콘솔)

1. https://console.aws.amazon.com/secretsmanager/에서 Secrets Manager 콘솔에 로그인합니다.
2. 복제본 리전으로 이동합니다.
3. 보안 암호(Secrets) 페이지에서 복제 보안 암호를 선택합니다.
4. 복제 보안 암호 세부 정보 페이지에서 독립 실행형 보안 암호로 승격(Promote to standalone secret)을 선택합니다.

5. 복제본을 독립형 보안 암호로 승격(Promote replica to standalone secret) 대화 상자에서 리전을 입력한 다음 복제본 승격(Promote replica)을 선택합니다.

AWS CLI

Example복제 보안 암호를 기본으로 승격

다음 `stop-replication-to-replica` 예시에서는 복제 암호와 기본 암호 간의 링크를 제거합니다. 복제 보안 암호는 복제본 리전의 기본 보안 암호로 승격됩니다. 복제 리전 내에서 `stop-replication-to-replica`를 직접적으로 호출해야 합니다.

```
aws secretsmanager stop-replication-to-replica \
    --secret-id MyTestSecret
```

AWS SDK

복제본을 독립 실행형 보안 암호로 승격하려면 `StopReplicationToReplica` 명령을 사용합니다. 이 명령은 반드시 복제본 보안 암호 리전에서 호출해야 합니다. 자세한 내용은 the section called “AWS SDKs” 단원을 참조하십시오.

AWS Secrets Manager 복제 방지

보안 암호는 `ReplicateSecretToRegions`를 사용하여 복제할 수 있고, `CreateSecret`를 사용하여 생성한 경우에도 복제할 수 있으므로 사용자가 보안 암호를 복제하는 걸 방지하려면 `AddReplicaRegions` 파라미터가 포함된 작업을 차단하는 것이 좋습니다. 권한 정책의 `Condition` 문을 사용하면 복제본 리전을 추가하지 않는 작업만 허용할 수 있습니다. 사용할 수 있는 조건문은 아래의 정책 예제를 참조하세요.

Example복제 권한 방지

다음 정책 예제에서는 복제본 리전을 추가하지 않는 모든 작업을 허용하는 방법을 보여줍니다. 이렇게 하면 사용자가 `ReplicateSecretToRegions` 및 `CreateSecret`를 통해 보안 암호를 복제할 수 없습니다.

JSON

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

```
    {
      "Effect": "Allow",
      "Action": "secretsmanager:*",
      "Resource": "*",
      "Condition": {
        "Null": {
          "secretsmanager:AddReplicaRegions": "true"
        }
      }
    }
  ]
}
```

Example특정 리전에만 복제 권한 허용

아래의 정책은 다음 작업을 모두 허용하는 방법을 보여줍니다.

- 복제 없이 보안 암호 생성
- 미국 및 캐나다의 리전에만 복제를 포함한 보안 암호 생성
- 미국 및 캐나다의 리전에만 보안 암호 복제

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:CreateSecret",
        "secretsmanager:ReplicateSecretToRegions"
      ],
      "Resource": "*",
      "Condition": {
        "ForAllValues:StringLike": {
          "secretsmanager:AddReplicaRegions": [
            "us-*",
            "ca-*"
          ]
        }
```

```
      }
    }
  ]
}
```

AWS Secrets Manager 복제 문제 해결

AWS Secrets Manager 복제는 다양한 이유로 실패할 수 있습니다. 보안 암호가 복제되지 않은 원인을 확인하려면 다음 중 하나를 수행할 수 있습니다.

- `DescribeSecret` API 작업 호출
- AWS CloudTrail 이벤트 검토

복제가 실패한 경우 다음과 같은 동작이 발생합니다.

- 사용할 수 있는 보안 암호 버전이 없는 경우, Secrets Manager는 복제된 리전에서 해당 보안 암호를 제거합니다.
- 성공적으로 복제된 보안 암호 버전이 있는 경우, 사용자가 `RemoveRegionsFromReplication` API 작업을 사용해 명시적으로 제거할 때까지 복제된 리전에 남아 있습니다.

다음 섹션에서는 복제가 실패하는 몇 가지 일반적인 원인을 설명합니다.

선택한 리전에 동일한 이름의 보안 암호가 있습니다

이 문제를 해결하려면 복제본 리전에서 중복된 이름 보안 암호를 덮어쓸 수 있습니다. 복제를 재시도한 다음 복제 재시도 대화 상자에서 덮어쓰기를 선택합니다.

복제를 완료하기 위해 KMS 키에 사용할 수 있는 권한이 없습니다

Secrets Manager는 복제본 리전의 새 KMS 키를 사용하여 다시 암호화하기 전에 먼저 암호를 해독합니다. 기본 리전의 암호화 키에 대한 `kms:Decrypt` 권한이 없는 경우 이 오류가 발생합니다. `aws/secretsmanager` 이외의 KMS 키를 사용하여 복제된 보안 암호를 암호화하려면 키에 `kms:GenerateDataKey` 및 `kms:Encrypt`가 필요합니다. the section called “KMS 키에 대한 권한”을(를) 참조하세요.

KMS 키가 비활성화되었거나 찾을 수 없음

기본 리전의 암호화 키가 비활성화되거나 삭제된 경우 Secrets Manager는 보안 암호를 복제할 수 없습니다. 보안 암호에 비활성화되거나 삭제된 암호화 키를 사용하여 암호화된 사용자 지정 레이블 버전이 있는 경우, 암호화 키를 변경한 경우에도 이 오류가 발생할 수 있습니다. Secrets Manager가 암호화를 수행하는 방법에 대한 자세한 내용은 the section called “보안 암호 암호화 및 복호화”을(를) 참조하세요. 이 문제를 해결하려면 Secrets Manager가 현재 암호화 키를 사용하여 암호화하도록 보안 암호 버전을 다시 생성하면 됩니다. 자세한 내용은 보안 암호에 대한 암호화 키 변경을 참조하세요. 그런 다음 복제를 다시 시도합니다.

```
aws secretsmanager put-secret-value \
  --secret-id testDescriptionUpdate \
  --secret-string "SecretValue" \
  --version-stages "MyCustomLabel"
```

복제가 발생하는 리전을 활성화하지 않았습니다

리전을 활성화하는 방법에 대한 자세한 내용은 AWS 계정 관리 참조 가이드의 AWS 리전 관리를 참조하세요.

에서 보안 암호 가져오기 AWS Secrets Manager

Secrets Manager는 암호를 검색할 때 CloudTrail 로그 항목을 생성합니다. 자세한 내용은 the section called “를 사용하여 로그 AWS CloudTrail ” 단원을 참조하십시오.

다음을 사용하여 보안 암호 값을 검색할 수 있습니다.

- Java를 사용하여 Secrets Manager 보안 암호 값 가져오기
- Python을 사용하여 Secrets Manager 보안 암호 값 가져오기
- .NET을 사용하여 Secrets Manager 보안 암호 값 가져오기
- Go를 사용하여 Secrets Manager 보안 암호 값 가져오기
- Rust를 사용하여 Secrets Manager 보안 암호 값 가져오기
- Amazon Elastic Kubernetes Service에서 AWS Secrets Manager 보안 암호 사용
- AWS Lambda 함수에서 AWS Secrets Manager 보안 암호 사용
- AWS Secrets Manager 에이전트 사용
- C++ AWS SDK를 사용하여 Secrets Manager 보안 암호 값 가져오기
- JavaScript AWS SDK를 사용하여 Secrets Manager 보안 암호 값 가져오기
- Kotlin AWS SDK를 사용하여 Secrets Manager 보안 암호 값 가져오기
- PHP AWS SDK를 사용하여 Secrets Manager 보안 암호 값 가져오기
- Ruby AWS SDK를 사용하여 Secrets Manager 보안 암호 값 가져오기
- 를 사용하여 보안 암호 값 가져오기 AWS CLI
- AWS 콘솔을 사용하여 보안 암호 값 가져오기
- 에서 AWS Secrets Manager 보안 암호 사용 AWS Batch
- CloudFormation 리소스에서 AWS Secrets Manager 보안 암호 가져오기
- GitHub 작업에서 AWS Secrets Manager 보안 암호 사용
- GitLab AWS Secrets Manager 에서 사용
- 에서 AWS Secrets Manager 보안 암호 사용 AWS IoT Greengrass
- Parameter Store에서 AWS Secrets Manager 보안 암호 사용

Java를 사용하여 Secrets Manager 보안 암호 값 가져오기

애플리케이션에서 `GetSecretValue` 또는 SDK`BatchGetSecretValue`를 호출하여 보안 암호를 검색할 수 있습니다. AWS SDKs 그러나 클라이언트 측 캐싱을 사용하여 보안 암호 값을 캐싱하는 것이 좋습니다. 보안 암호 캐싱은 속도를 향상시키고 비용을 절감합니다.

보안 암호의 자격 증명을 사용하여 데이터베이스에 연결하려는 경우, 기본 JDBC 드라이버를 래핑하는 Secrets Manager SQL Connection 드라이버를 사용할 수 있습니다. 또한 이 방법은 클라이언트 측 캐싱을 사용하므로, Secrets Manager API 직접 호출 비용을 줄일 수 있습니다.

주제

- 클라이언트 측 캐싱과 함께 Java를 사용하여 Secrets Manager 보안 암호 값 가져오기
- AWS Secrets Manager 보안 암호의 자격 증명과 함께 JDBC를 사용하여 SQL 데이터베이스에 연결
- Java AWS SDK를 사용하여 Secrets Manager 보안 암호 값 가져오기

클라이언트 측 캐싱과 함께 Java를 사용하여 Secrets Manager 보안 암호 값 가져오기

보안 암호를 검색할 때 Secret Manager Java 기반 캐싱 구성 요소를 사용하여 나중에 사용할 수 있도록 캐싱할 수 있습니다. 캐싱된 보안 암호를 검색하는 것이 Secret Manager에서 검색하는 것보다 빠릅니다. Secrets Manager API를 호출하는 데는 비용이 발생하므로 캐시를 사용하면 비용을 줄일 수 있습니다. 암호를 검색할 수 있는 모든 방법은 보안 암호 가져오기을(를) 참조하세요.

캐시 정책은 LRU(가장 오랫동안 사용되지 않음)이므로, 캐시에서 보안 암호를 폐기해야 하는 경우 가장 오랫동안 사용되지 않은 보안 암호가 삭제됩니다. 기본적으로 캐시는 보안 암호를 매시간 새로 고칩니다. 캐시에서 보안 암호를 새로 고치는 주기를 구성하고 보안 암호 검색에 연결하여 더 많은 기능을 추가할 수 있습니다.

캐시 참조가 해제되면 캐시는 가비지 수집을 강제로 적용하지 않습니다. 캐시 구현에는 캐시 무효화가 포함되지 않습니다. 캐시 구현은 캐시 자체에 중점을 두며, 보안을 강화하거나 보안에 초점을 맞추지 않습니다. 캐시에서 항목 암호화와 같은 추가 보안이 필요한 경우 제공된 인터페이스 및 추상 메서드를 사용하세요.

이 구성 요소를 사용하려면 다음이 필요합니다.

- Java 8 이상 개발 환경입니다. Oracle 웹 사이트의 Java SE 다운로드를 참조하세요.

소스 코드를 다운로드하려면 GitHub의 Secrets Manager Java 기반 캐싱 클라이언트 구성 요소를 참조하세요.

프로젝트에 구성 요소를 추가하려면 Maven pom.xml 파일에 다음 dependency를 포함합니다. Maven에 대한 자세한 내용은 Apache Maven Project 웹 사이트의 시작 안내서를 참조하세요.

```
<dependency>
  <groupId>com.amazonaws.secretsmanager</groupId>
  <artifactId>aws-secretsmanager-caching-java</artifactId>
  <version>1.0.2</version>
</dependency>
```

필요한 권한:

- `secretsmanager:DescribeSecret`
- `secretsmanager:GetSecretValue`

자세한 내용은 권한 참조 단원을 참조하십시오.

레퍼런스

- SecretCache
- SecretCacheConfiguration
- SecretCacheHook

Example보안 암호 검색

다음 코드 예제에서는 보안 암호 문자열을 검색하는 Lambda 함수를 보여줍니다. 이 예제는 함수 핸들러 외부의 캐시를 인스턴스화하는 모범 사례를 따르므로 Lambda 함수를 다시 호출하는 경우 API를 계속 호출하지 않습니다.

```
package com.amazonaws.secretsmanager.caching.examples;

    import com.amazonaws.services.lambda.runtime.Context;
    import com.amazonaws.services.lambda.runtime.RequestHandler;
    import com.amazonaws.services.lambda.runtime.LambdaLogger;

    import com.amazonaws.secretsmanager.caching.SecretCache;
```

```
    public class SampleClass implements RequestHandler<String, String> {

         private final SecretCache cache  = new SecretCache();

         @Override public String handleRequest(String secretId,  Context context) {
             final String secret  = cache.getSecretString(secretId);

            // Use the secret, return success;

        }
    }
```

SecretCache

Secret Manager에서 요청한 보안 암호에 대한 인 메모리 캐시. the section called “getSecretString” 또는 the section called “getSecretBinary”를 사용하여 캐시에서 보안 암호를 검색합니다. 생성자의 the section called “SecretCacheConfiguration” 객체에 전달하여 캐시 설정을 구성할 수 있습니다.

예제를 포함한 자세한 내용은 the section called “클라이언트 측 캐싱을 사용한 Java”을 참조하세요.

Constructors

`public SecretCache()`

SecretCache 객체에 대한 기본 생성자.

`public SecretCache(AWSSecretsManagerClientBuilder builder)`

제공된 `AWSSecretsManagerClientBuilder`로 생성한 Secrets Manager 클라이언트를 사용하여 새 캐시를 생성합니다. 이 생성자를 사용하여 Secrets Manager 클라이언트를 사용자 지정합니다(예: 특정 리전 또는 엔드포인트를 사용하도록 사용자 지정).

`public SecretCache(AWSSecretsManager client)`

제공된 `AWSSecretsManagerClient`를 사용하여 새 보안 암호 캐시를 생성합니다. 이 생성자를 사용하여 Secrets Manager 클라이언트를 사용자 지정합니다(예: 특정 리전 또는 엔드포인트를 사용하도록 사용자 지정).

`public SecretCache(SecretCacheConfiguration config)`

제공된 `the section called “SecretCacheConfiguration”`을 사용하여 새 보안 암호 캐시를 생성합니다.

메서드

getSecretString

```
public String getSecretString(final String secretId)
```

Secrets Manager에서 문자열 보안 암호를 검색합니다. `String`을 반환합니다.

getSecretBinary

```
public ByteBuffer getSecretBinary(final String secretId)
```

Secrets Manager에서 이진 보안 암호를 검색합니다. `ByteBuffer`을 반환합니다.

refreshNow

```
public boolean refreshNow(final String secretId) throws
InterruptedException
```

캐시를 강제로 새로 고칩니다. 오류 없이 새로 고침이 완료되는 경우 `true`를 반환하고 그렇지 않으면 `false`를 반환합니다.

close

```
public void close()
```

캐시를 닫습니다.

SecretCacheConfiguration

캐싱된 보안 암호에 대한 최대 캐시 크기 및 유지 시간(TTL)과 같은 the section called “SecretCache”에 대한 캐시 구성 옵션.

생성자

```
public SecretCacheConfiguration
```

`SecretCacheConfiguration` 객체에 대한 기본 생성자.

메서드

getClient

```
public AWSSecretsManager getClient()
```

캐시가 보안 암호를 검색할 AWSSecretsManagerClient를 반환합니다.

setClient

```
public void setClient(AWSSecretsManager client)
```

캐시가 보안 암호를 검색할 AWSSecretsManagerClient를 설정합니다.

getCacheHook

```
public SecretCacheHook getCacheHook()
```

캐시 업데이트를 연결하는 데 사용되는 the section called “SecretCacheHook” 인터페이스를 반환합니다.

setCacheHook

```
public void setCacheHook(SecretCacheHook cacheHook)
```

캐시 업데이트를 연결하는 데 사용되는 the section called “SecretCacheHook” 인터페이스를 설정합니다.

getMaxCacheSize

```
public int getMaxCacheSize()
```

최대 캐시 크기를 반환합니다. 기본값은 보안 암호 1,024개입니다.

setMaxCacheSize

```
public void setMaxCacheSize(int maxCacheSize)
```

최대 캐시 크기를 설정합니다. 기본값은 보안 암호 1,024개입니다.

getCacheItemTTL

```
public long getCacheItemTTL()
```

캐싱된 항목에 대한 TTL을 밀리초 단위로 반환합니다. 캐싱된 보안 암호가 이 TTL을 초과하면 캐시는 AWSSecretsManagerClient에서 보안 암호의 새 사본을 검색합니다. 기본값은 밀리초 단위로 1시간입니다.

TTL 이후에 보안 암호가 요청되면 캐시가 보안 암호를 동기식으로 새로 고칩니다. 동기식 새로 고침이 실패하면 캐시는 오래된 보안 암호를 반환합니다.

setCacheItemTTL

```
public void setCacheItemTTL(long cacheItemTTL)
```

캐싱된 항목에 대한 TTL을 밀리초 단위로 설정합니다. 캐싱된 보안 암호가 이 TTL을 초과하면 캐시는 AWSSecretsManagerClient에서 보안 암호의 새 사본을 검색합니다. 기본값은 밀리초 단위로 1시간입니다.

getVersionStage

```
public String getVersionStage()
```

캐싱할 보안 암호의 버전을 반환합니다. 자세한 내용은 보안 암호 버전을 참조하세요. 기본값은 "AWSCURRENT"입니다.

setVersionStage

```
public void setVersionStage(String versionStage)
```

캐싱할 보안 암호의 버전을 설정합니다. 자세한 내용은 보안 암호 버전을 참조하세요. 기본값은 "AWSCURRENT"입니다.

SecretCacheConfiguration withClient

```
public SecretCacheConfiguration withClient(AWSSecretsManager client)
```

보안 암호를 검색할 AWSSecretsManagerClient를 설정합니다. 새 설정으로 업데이트된 SecretCacheConfiguration 객체를 반환합니다.

SecretCacheConfiguration withCacheHook

```
public SecretCacheConfiguration withCacheHook(SecretCacheHook cacheHook)
```

인 메모리 캐시를 연결하는 데 사용되는 인터페이스를 설정합니다. 새 설정으로 업데이트된 SecretCacheConfiguration 객체를 반환합니다.

SecretCacheConfiguration withMaxCacheSize

```
public SecretCacheConfiguration withMaxCacheSize(int maxCacheSize)
```

최대 캐시 크기를 설정합니다. 새 설정으로 업데이트된 SecretCacheConfiguration 객체를 반환합니다.

SecretCacheConfiguration withCacheItemTTL

```
public SecretCacheConfiguration withCacheItemTTL(long cacheItemTTL)
```

캐싱된 항목에 대한 TTL을 밀리초 단위로 설정합니다. 캐싱된 보안 암호가 이 TTL을 초과하면 캐시는 AWSSecretsManagerClient에서 보안 암호의 새 사본을 검색합니다. 기본값은 밀리초 단위로 1시간입니다. 새 설정으로 업데이트된 SecretCacheConfiguration 객체를 반환합니다.

SecretCacheConfiguration withVersionStage

```
public SecretCacheConfiguration withVersionStage(String versionStage)
```

캐싱할 보안 암호의 버전을 설정합니다. 자세한 내용은 보안 암호 버전을 참조하세요. 새 설정으로 업데이트된 SecretCacheConfiguration 객체를 반환합니다.

SecretCacheHook

캐시에 저장 중인 보안 암호에 대한 작업을 수행하기 위해 the section called “SecretCache”에 연결되는 인터페이스.

put

```
Object put(final Object o)
```

캐시에 저장할 객체를 준비합니다.

캐시에 저장할 객체를 반환합니다.

시작

```
Object get(final Object cachedObject)
```

캐싱된 객체에서 객체를 추출합니다.

캐시에서 객체를 반환합니다.

AWS Secrets Manager 보안 암호의 자격 증명과 함께 JDBC를 사용하여 SQL 데이터베이스에 연결

Java 애플리케이션에서는 Secrets Manager SQL Connection 드라이버를 사용하여 Secrets Manager에 저장된 자격 증명으로 MySQL, PostgreSQL, Oracle, MSSQLServer, Db2, Redshift 데이터베이스에 연결할 수 있습니다. 각 드라이버는 기본 JDBC 드라이버를 래핑하므로 JDBC 호출을 사용하여 데이터베이스에 액세스할 수 있습니다. 그러나 연결을 위한 사용자 이름과 암호를 전달하는 대신 보안 암호의

ID를 제공합니다. 드라이버는 Secrets Manager를 호출하여 보안 암호 값을 검색한 다음 보안 암호의 보안 인증 정보를 사용하여 데이터베이스에 연결합니다. 또한 드라이버는 Java 클라이언트 측 캐싱 라이브러리를 사용하여 자격 증명을 캐싱하므로 향후 연결에는 Secret Manager에 대한 호출이 필요하지 않습니다. 기본적으로 캐시는 매시간, 그리고 보안 암호가 교체될 때마다 새로 고침됩니다. 캐시를 구성하려면 the section called “SecretCacheConfiguration” 섹션을 참조하세요.

GitHub에서 소스 코드를 다운로드할 수 있습니다.

Secrets Manager SQL Connection 드라이버를 사용하려면:

- 애플리케이션이 Java 8 이상이어야 합니다.
- 보안 암호는 다음 형식 중 하나이어야 합니다:
  - 예상 JSON 구조의 데이터베이스 보안 암호. 형식을 확인하려면 Secrets Manager 콘솔에서 보안 암호를 확인하고 보안 암호 값 검색(Retrieve secret value)을 선택합니다. 또는에서 get-secret-value를 AWS CLI호출합니다.
  - Amazon RDS 관리형 보안 암호. 이 유형의 암호에 대해서는 연결을 설정할 때 엔드포인트 및 포트를 지정해야 합니다.
  - Amazon Redshift 관리형 보안 암호입니다. 이 유형의 암호에 대해서는 연결을 설정할 때 엔드포인트 및 포트를 지정해야 합니다.

데이터베이스가 다른 리전으로 복제되는 경우 다른 리전의 복제본 데이터베이스에 연결하려면 연결을 생성할 때 리전 엔드포인트 및 포트를 지정합니다. 리전 연결 정보를 보안 암호에 추가 키/값 쌍으로 저장하거나, SSM Parameter Store 파라미터에 저장하거나, 코드 구성에 저장할 수 있습니다.

프로젝트에 드라이버를 추가하려면 Maven 빌드 파일 `pom.xml`에 드라이버에 대한 다음 종속성을 추가합니다. 자세한 내용은 Maven Central Repository 웹 사이트의 Secrets Manager SQL Connection Library를 참조하세요.

```
<dependency>
    <groupId>com.amazonaws.secretsmanager</groupId>
    <artifactId>aws-secretsmanager-jdbc</artifactId>
    <version>1.0.12</version>
</dependency>
```

드라이버는 기본 자격 증명 공급자 체인을 사용합니다. Amazon EKS에서 드라이버를 실행하는 경우 서비스 계정 역할 대신 실행 중인 노드의 보안 인증 정보를 가져올 수 있습니다. 이 사항을 처리하려면 Gradle 또는 Maven 프로젝트 파일에 `com.amazonaws:aws-java-sdk-sts` 버전 1을 종속 항목으로 추가하세요.

`secretsmanager.properties` 파일에서 AWS PrivateLink DNS 엔드포인트 URL 및 리전을 설정하려면:

```
drivers.vpcEndpointUrl = endpoint URL
drivers.vpcEndpointRegion = endpoint region
```

기본 리전을 재정의하려면 `AWS_SECRET_JDBC_REGION` 환경 변수를 설정하거나 `secretsmanager.properties` 파일을 다음과 같이 변경하세요.

```
drivers.region = region
```

필요한 권한:

- `secretsmanager:DescribeSecret`
- `secretsmanager:GetSecretValue`

자세한 내용은 권한 참조 단원을 참조하십시오.

예시:

- 데이터베이스에 대한 연결 설정
- 엔드포인트 및 포트를 지정하여 연결 설정
- c3p0 연결 풀링을 사용하여 연결 설정
- c3p0 연결 풀링을 사용하여 엔드포인트 및 포트 지정을 통한 연결 설정

데이터베이스에 대한 연결 설정

다음 예시에서는 보안 암호의 자격 증명 및 연결 정보를 사용하여 데이터베이스에 대한 연결을 설정하는 방법을 보여줍니다. 연결이 설정되면 JDBC 호출을 사용하여 데이터베이스에 액세스할 수 있습니다. 자세한 내용은 Java 설명서 웹 사이트의 JDBC Basics를 참조하세요.

MySQL

```
// Load the JDBC driver
Class.forName( "com.amazonaws.secretsmanager.sql.AWSSecretsManagerMySQLDriver" ).newInstance

// Retrieve the connection info from the secret using the secret ARN
String URL = "secretId";
```

```
// Populate the user property with the secret ARN to retrieve user and password from
 the secret
Properties info = new Properties( );
info.put( "user", "secretId" );

// Establish the connection
conn = DriverManager.getConnection(URL, info);
```

PostgreSQL

```
// Load the JDBC driver
Class.forName( "com.amazonaws.secretsmanager.sql.AWSSecretsManagerPostgreSQLDriver" ).newIns

// Retrieve the connection info from the secret using the secret ARN
String URL = "secretId";

// Populate the user property with the secret ARN to retrieve user and password from
 the secret
Properties info = new Properties( );
info.put( "user", "secretId" );

// Establish the connection
conn = DriverManager.getConnection(URL, info);
```

Oracle

```
// Load the JDBC driver
Class.forName( "com.amazonaws.secretsmanager.sql.AWSSecretsManagerOracleDriver" ).newInstanc

// Retrieve the connection info from the secret using the secret ARN
String URL = "secretId";

// Populate the user property with the secret ARN to retrieve user and password from
 the secret
Properties info = new Properties( );
info.put( "user", "secretId" );

// Establish the connection
conn = DriverManager.getConnection(URL, info);
```

MSSQLServer

```
// Load the JDBC driver
```

```
Class.forName( "com.amazonaws.secretsmanager.sql.AWSSecretsManagerMSSQLServerDriver" ).newIn

// Retrieve the connection info from the secret using the secret ARN
String URL = "secretId";

// Populate the user property with the secret ARN to retrieve user and password from
 the secret
Properties info = new Properties( );
info.put( "user", "secretId" );

// Establish the connection
conn = DriverManager.getConnection(URL, info);
```

Db2

```
// Load the JDBC driver
Class.forName( "com.amazonaws.secretsmanager.sql.AWSSecretsManagerDb2Driver" ).newInstance()

// Retrieve the connection info from the secret using the secret ARN
String URL = "secretId";

// Populate the user property with the secret ARN to retrieve user and password from
 the secret
Properties info = new Properties( );
info.put( "user", "secretId" );

// Establish the connection
conn = DriverManager.getConnection(URL, info);
```

Redshift

```
// Load the JDBC driver
Class.forName( "com.amazonaws.secretsmanager.sql.AWSSecretsManagerRedshiftDriver" ).newInsta

// Retrieve the connection info from the secret using the secret ARN
String URL = "secretId";

// Populate the user property with the secret ARN to retrieve user and password from
 the secret
Properties info = new Properties( );
info.put( "user", "secretId" );

// Establish the connection
```

```
conn = DriverManager.getConnection(URL, info);
```

엔드포인트 및 포트를 지정하여 연결 설정

다음 예에서는 지정된 엔드포인트 및 포트와 함께 보안 암호의 보안 인증 정보를 사용하여 데이터베이스에 대한 연결을 설정하는 방법을 보여줍니다.

Amazon RDS 관리형 보안 암호에는 데이터베이스의 엔드포인트 및 포트가 포함되지 않습니다. Amazon RDS에서 관리하는 보안 암호의 마스터 보안 인증을 사용하여 데이터베이스에 연결하려면 이를 해당 코드로 지정해야 합니다.

다른 리전으로 복제된 보안 암호는 리전 데이터베이스에 대한 연결 지연 시간을 줄여 줄 수 있지만, 소스 보안 암호와 다른 연결 정보는 포함하지 않습니다. 각 복제본은 소스 보안 암호의 복사본입니다. 리전 연결 정보를 보안 암호에 저장하려면 엔드포인트에 대한 키/값 쌍 및 리전에 대한 포트 정보를 추가합니다.

연결이 설정되면 JDBC 호출을 사용하여 데이터베이스에 액세스할 수 있습니다. 자세한 내용은 Java 설명서 웹 사이트의 JDBC Basics를 참조하세요.

MySQL

```
// Load the JDBC driver
Class.forName( "com.amazonaws.secretsmanager.sql.AWSSecretsManagerMySQLDriver" ).newInstance

// Set the endpoint and port. You can also retrieve it from a key/value pair in the
 secret.
String URL = "jdbc-secretsmanager:mysql://example.com:3306";

// Populate the user property with the secret ARN to retrieve user and password from
 the secret
Properties info = new Properties( );
info.put( "user", "secretId" );

// Establish the connection
conn = DriverManager.getConnection(URL, info);
```

PostgreSQL

```
// Load the JDBC driver
Class.forName( "com.amazonaws.secretsmanager.sql.AWSSecretsManagerPostgreSQLDriver" ).newIns
```

```
// Set the endpoint and port. You can also retrieve it from a key/value pair in the
 secret.
String URL = "jdbc-secretsmanager:postgresql://example.com:5432/database";

// Populate the user property with the secret ARN to retrieve user and password from
 the secret
Properties info = new Properties( );
info.put( "user", "secretId" );

// Establish the connection
conn = DriverManager.getConnection(URL, info);
```

Oracle

```
// Load the JDBC driver
Class.forName( "com.amazonaws.secretsmanager.sql.AWSSecretsManagerOracleDriver" ).newInstanc

// Set the endpoint and port. You can also retrieve it from a key/value pair in the
 secret.
String URL = "jdbc-secretsmanager:oracle:thin:@example.com:1521/ORCL";

// Populate the user property with the secret ARN to retrieve user and password from
 the secret
Properties info = new Properties( );
info.put( "user", "secretId" );

// Establish the connection
conn = DriverManager.getConnection(URL, info);
```

MSSQLServer

```
// Load the JDBC driver
Class.forName( "com.amazonaws.secretsmanager.sql.AWSSecretsManagerMSSQLServerDriver" ).newIn

// Set the endpoint and port. You can also retrieve it from a key/value pair in the
 secret.
String URL = "jdbc-secretsmanager:sqlserver://example.com:1433";

// Populate the user property with the secret ARN to retrieve user and password from
 the secret
Properties info = new Properties( );
info.put( "user", "secretId" );
```

```
// Establish the connection
conn = DriverManager.getConnection(URL, info);
```

Db2

```
// Load the JDBC driver
Class.forName( "com.amazonaws.com.amazonaws.secretsmanager.sql.AWSSecretsManagerDb2Driver" )

// Set the endpoint and port. You can also retrieve it from a key/value pair in the
 secret.
String URL = "jdbc-secretsmanager:db2://example.com:50000";

// Populate the user property with the secret ARN to retrieve user and password from
 the secret
Properties info = new Properties( );
info.put( "user", "secretId" );

// Establish the connection
conn = DriverManager.getConnection(URL, info);
```

Redshift

```
// Load the JDBC driver
Class.forName( "com.amazonaws.com.amazonaws.secretsmanager.sql.AWSSecretsManagerRedshiftDriv

// Set the endpoint and port. You can also retrieve it from a key/value pair in the
 secret.
String URL = "jdbc-secretsmanager:redshift://example.com:5439";

// Populate the user property with the secret ARN to retrieve user and password from
 the secret
Properties info = new Properties( );
info.put( "user", "secretId" );

// Establish the connection
conn = DriverManager.getConnection(URL, info);
```

c3p0 연결 풀링을 사용하여 연결 설정

다음 예에서는 드라이버를 사용하여 보안 암호에서 보안 인증 정보 및 연결 정보를 검색하는 `c3p0.properties` 파일로 연결 풀을 설정하는 방법을 보여줍니다. `user`와 `jdbcUrl`에 대해 보안 암

호 ID를 입력하여 연결 풀을 구성합니다. 그런 다음 풀에서 연결을 검색하여 다른 데이터베이스 연결과 동일하게 사용할 수 있습니다. 자세한 내용은 Java 설명서 웹 사이트의 JDBC Basics를 참조하세요.

c3p0에 대한 자세한 내용은 Machinery For Change 웹 사이트의 c3p0을 참조하세요.

MySQL

```
c3p0.user=secretId
c3p0.driverClass=com.amazonaws.secretsmanager.sql.AWSSecretsManagerMySQLDriver
c3p0.jdbcUrl=secretId
```

PostgreSQL

```
c3p0.user=secretId
c3p0.driverClass=com.amazonaws.secretsmanager.sql.AWSSecretsManagerPostgreSQLDriver
c3p0.jdbcUrl=secretId
```

Oracle

```
c3p0.user=secretId
c3p0.driverClass=com.amazonaws.secretsmanager.sql.AWSSecretsManagerOracleDriver
c3p0.jdbcUrl=secretId
```

MSSQLServer

```
c3p0.user=secretId
c3p0.driverClass=com.amazonaws.secretsmanager.sql.AWSSecretsManagerMSSQLServerDriver
c3p0.jdbcUrl=secretId
```

Db2

```
c3p0.user=secretId
c3p0.driverClass=com.amazonaws.secretsmanager.sql.AWSSecretsManagerDb2Driver
c3p0.jdbcUrl=secretId
```

Redshift

```
c3p0.user=secretId
c3p0.driverClass=com.amazonaws.secretsmanager.sql.AWSSecretsManagerRedshiftDriver
c3p0.jdbcUrl=secretId
```

c3p0 연결 풀링을 사용하여 엔드포인트 및 포트 지정을 통한 연결 설정

다음 예에서는 지정된 엔드포인트 및 포트와 함께 드라이버를 사용하여 보안 암호의 보안 인증 정보를 검색하는 `c3p0.properties` 파일로 연결 풀을 설정하는 방법을 보여줍니다. 그런 다음 풀에서 연결을 검색하여 다른 데이터베이스 연결과 동일하게 사용할 수 있습니다. 자세한 내용은 Java 설명서 웹 사이트의 JDBC Basics를 참조하세요.

Amazon RDS 관리형 보안 암호에는 데이터베이스의 엔드포인트 및 포트가 포함되지 않습니다. Amazon RDS에서 관리하는 보안 암호의 마스터 보안 인증을 사용하여 데이터베이스에 연결하려면 이를 해당 코드로 지정해야 합니다.

다른 리전으로 복제된 보안 암호는 리전 데이터베이스에 대한 연결 지연 시간을 줄여 줄 수 있지만, 소스 보안 암호와 다른 연결 정보는 포함하지 않습니다. 각 복제본은 소스 보안 암호의 복사본입니다. 리전 연결 정보를 보안 암호에 저장하려면 엔드포인트에 대한 키/값 쌍 및 리전에 대한 포트 정보를 추가합니다.

MySQL

```
c3p0.user=secretId
c3p0.driverClass=com.amazonaws.secretsmanager.sql.AWSSecretsManagerMySQLDriver
c3p0.jdbcUrl=jdbc-secretsmanager:mysql://example.com:3306
```

PostgreSQL

```
c3p0.user=secretId
c3p0.driverClass=com.amazonaws.secretsmanager.sql.AWSSecretsManagerPostgreSQLDriver
c3p0.jdbcUrl=jdbc-secretsmanager:postgresql://example.com:5432/database
```

Oracle

```
c3p0.user=secretId
c3p0.driverClass=com.amazonaws.secretsmanager.sql.AWSSecretsManagerOracleDriver
c3p0.jdbcUrl=jdbc-secretsmanager:oracle:thin:@example.com:1521/ORCL
```

MSSQLServer

```
c3p0.user=secretId
c3p0.driverClass=com.amazonaws.secretsmanager.sql.AWSSecretsManagerMSSQLServerDriver
c3p0.jdbcUrl=jdbc-secretsmanager:sqlserver://example.com:1433
```

Db2

```
c3p0.user=secretId
c3p0.driverClass=com.amazonaws.secretsmanager.sql.AWSSecretsManagerDb2Driver
c3p0.jdbcUrl=jdbc-secretsmanager:db2://example.com:50000
```

Redshift

```
c3p0.user=secretId
c3p0.driverClass=com.amazonaws.secretsmanager.sql.AWSSecretsManagerRedshiftDriver
c3p0.jdbcUrl=jdbc-secretsmanager:redshift://example.com:5439
```

Java AWS SDK를 사용하여 Secrets Manager 보안 암호 값 가져오기

애플리케이션에서 GetSecretValue 또는 SDKBatchGetSecretValue를 호출하여 보안 암호를 검색할 수 있습니다. AWS SDKs 그러나 클라이언트 측 캐싱을 사용하여 보안 암호 값을 캐싱하는 것이 좋습니다. 보안 암호 캐싱은 속도를 향상시키고 비용을 절감합니다.

- 보안 암호에 데이터베이스 자격 증명을 저장하는 경우, Secrets Manager SQL 연결 드라이버를 사용하여 보안 암호의 자격 증명으로 데이터베이스에 연결합니다.
- 다른 보안 암호 유형의 경우 Secrets Manager Java 기반 캐싱 구성 요소를 사용하거나 GetSecretValue 또는 BatchGetSecretValue를 사용하여 SDK를 직접 호출합니다.

다음 코드 예시는 GetSecretValue의 사용 방법을 보여 줍니다.

필요한 권한:secretsmanager:GetSecretValue

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;
import software.amazon.awssdk.services.secretsmanager.model.SecretsManagerException;

/**
 * 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
```

```
 *
 * We recommend that you cache your secret values by using client-side caching.
 *
 * Caching secrets improves speed and reduces your costs. For more information,
 * see the following documentation topic:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/retrieving-secrets.html
 */
public class GetSecretValue {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <secretName>\s

                Where:
                    secretName - The name of the secret (for example, tutorials/
MyFirstSecret).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String secretName = args[0];
        Region region = Region.US_EAST_1;
        SecretsManagerClient secretsClient = SecretsManagerClient.builder()
                .region(region)
                .build();

        getValue(secretsClient, secretName);
        secretsClient.close();
    }

    public static void getValue(SecretsManagerClient secretsClient, String secretName)
 {
        try {
            GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
                    .secretId(secretName)
                    .build();

            GetSecretValueResponse valueResponse =
 secretsClient.getSecretValue(valueRequest);
```

```
            String secret = valueResponse.secretString();
            System.out.println(secret);

        } catch (SecretsManagerException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Python을 사용하여 Secrets Manager 보안 암호 값 가져오기

애플리케이션에서 GetSecretValue 또는 SDKBatchGetSecretValue를 호출하여 보안 암호를 검색할 수 있습니다. AWS SDKs 그러나 클라이언트 측 캐싱을 사용하여 보안 암호 값을 캐싱하는 것이 좋습니다. 보안 암호 캐싱은 속도를 향상시키고 비용을 절감합니다.

주제

- 클라이언트 측 캐싱과 함께 Python을 사용하여 Secrets Manager 보안 암호 값 가져오기
- Python AWS SDK를 사용하여 Secrets Manager 보안 암호 값 가져오기
- Python AWS SDK를 사용하여 Secrets Manager 보안 암호 값 배치 가져오기

클라이언트 측 캐싱과 함께 Python을 사용하여 Secrets Manager 보안 암호 값 가져오기

보안 암호를 검색할 때 Secret Manager Python 기반 캐싱 구성 요소를 사용하여 나중에 사용할 수 있도록 캐싱할 수 있습니다. 캐싱된 보안 암호를 검색하는 것이 Secret Manager에서 검색하는 것보다 빠릅니다. Secrets Manager API를 호출하는 데는 비용이 발생하므로 캐시를 사용하면 비용을 줄일 수 있습니다. 암호를 검색할 수 있는 모든 방법은 보안 암호 가져오기을(를) 참조하세요.

캐시 정책은 LRU(가장 오랫동안 사용되지 않음)이므로, 캐시에서 보안 암호를 폐기해야 하는 경우 가장 오랫동안 사용되지 않은 보안 암호가 삭제됩니다. 기본적으로 캐시는 보안 암호를 매시간 새로 고칩니다. 캐시에서 보안 암호를 새로 고치는 주기를 구성하고 보안 암호 검색에 연결하여 더 많은 기능을 추가할 수 있습니다.

캐시 참조가 해제되면 캐시는 가비지 수집을 강제로 적용하지 않습니다. 캐시 구현에는 캐시 무효화가 포함되지 않습니다. 캐시 구현은 캐시 자체에 중점을 두며, 보안을 강화하거나 보안에 초점을 맞추지 않습니다. 캐시에서 항목 암호화와 같은 추가 보안이 필요한 경우 제공된 인터페이스 및 추상 메서드를 사용하세요.

이 구성 요소를 사용하려면 다음이 필요합니다.

- Python 3.6 이상
- botocore 1.12 이상. AWS SDK for Python 및 Botocore를 참조하세요.
- setuptools_scm 3.2 이상. https://pypi.org/project/setuptools-scm/을 참조하세요.

소스 코드를 다운로드하려면 GitHub의 Secrets Manager Python 기반 캐싱 클라이언트 구성 요소를 참조하세요.

구성 요소를 설치하려면 다음 명령을 사용합니다.

```
$ pip install aws-secretsmanager-caching
```

필요한 권한:

- `secretsmanager:DescribeSecret`
- `secretsmanager:GetSecretValue`

자세한 내용은 권한 참조 단원을 참조하십시오.

레퍼런스

- SecretCache
- SecretCacheConfig
- SecretCacheHook
- @InjectSecretString
- @InjectKeywordedSecretString

Example보안 암호 검색

다음 예에서는 *mysecret*이라는 보안 암호에 대한 보안 암호 값을 가져오는 방법을 보여 줍니다.

```
import botocore
import botocore.session
from aws_secretsmanager_caching import SecretCache, SecretCacheConfig

client = botocore.session.get_session().create_client('secretsmanager')
cache_config = SecretCacheConfig()
```

```
cache = SecretCache( config = cache_config, client = client)

secret = cache.get_secret_string('mysecret')
```

SecretCache

Secret Manager에서 검색된 암호에 대한 인 메모리 캐시. the section called “get_secret_string” 또는 the section called “get_secret_binary”를 사용하여 캐시에서 보안 암호를 검색합니다. 생성자의 the section called “SecretCacheConfig” 객체에 전달하여 캐시 설정을 구성할 수 있습니다.

예제를 포함한 자세한 내용은 the section called “클라이언트 측 캐싱을 사용한 Python”을 참조하세요.

```
cache = SecretCache(
    config = the section called “SecretCacheConfig”,
    client = client
)
```

사용 가능한 메서드는 다음과 같습니다.

- get_secret_string
- get_secret_binary

get_secret_string

보안 암호 문자열 값을 검색합니다.

요청 구문

```
response = cache.get_secret_string(
    secret_id='string',
    version_stage='string' )
```

Parameters

- `secret_id`(문자열) [필수] 보안 암호의 ARN 이름.
- `version_stage`(문자열) 검색하려는 보안 암호의 버전. 자세한 내용은 보안 암호 버전을 참조하세요. 기본값은 'AWSCURRENT'입니다.

반환 타입

문자열

get_secret_binary

보안 암호 이진 값을 검색합니다.

요청 구문

```
response = cache.get_secret_binary(
    secret_id='string',
    version_stage='string'
)
```

Parameters

- `secret_id`(문자열) [필수] 보안 암호의 ARN 이름.
- `version_stage`(문자열) 검색하려는 보안 암호의 버전. 자세한 내용은 보안 암호 버전을 참조하세요. 기본값은 'AWSCURRENT'입니다.

반환 타입

Base64로 인코딩된 문자열

SecretCacheConfig

캐싱된 보안 암호에 대한 최대 캐시 크기 및 유지 시간(TTL)과 같은 the section called “SecretCache”에 대한 캐시 구성 옵션.

Parameters

`max_cache_size` (int)

최대 캐시 크기. 기본값은 보안 암호 1024개입니다.

`exception_retry_delay_base` (int)

예외가 발생한 후 요청을 재시도하기 전에 대기할 시간(초). 기본값은 1입니다.

`exception_retry_growth_factor` (int)pur

실패한 요청 재시도 간의 대기 시간을 계산하는 데 사용할 성장 계수. 기본값은 2입니다.

`exception_retry_delay_max` (int)

실패한 요청 사이에 대기할 최대 시간(초). 기본값은 3600입니다.

default_version_stage (str)

캐싱할 보안 암호의 버전. 자세한 내용은 보안 암호 버전을 참조하세요. 기본값은 'AWSCURRENT'입니다.

secret_refresh_interval (int)

캐싱된 보안 암호 정보를 새로 고치는 동안 대기할 시간(초). 기본값은 3600입니다.

secret_cache_hook (SecretCacheHook)

SecretCacheHook 추상 클래스의 구현. 기본값은 None입니다.

SecretCacheHook

캐시에 저장 중인 보안 암호에 대한 작업을 수행하기 위해 the section called “SecretCache”에 연결되는 인터페이스.

사용 가능한 메서드는 다음과 같습니다.

- put
- 시작

put

캐시에 저장할 객체를 준비합니다.

요청 구문

```
response = hook.put(
    obj='secret_object'
)
```

Parameters

- obj (객체) -- [필수] 보안 암호 또는 보안 암호를 포함하는 객체.

반환 타입

객체

시작

캐싱된 객체에서 객체를 추출합니다.

요청 구문

```
response = hook.get(
    obj='secret_object'
)
```

Parameters

- obj(객체) [필수] 보안 암호 또는 보안 암호를 포함하는 객체.

반환 타입

객체

@InjectSecretString

이 데코레이터는 첫 번째와 두 번째 인수로 보안 암호 ID 문자열과 the section called “SecretCache”을(를) 기대합니다. 데코레이터는 보안 암호 문자열 값을 반환합니다. 보안 암호는 문자열을 포함해야 합니다.

```
from aws_secretsmanager_caching import SecretCache
from aws_secretsmanager_caching import InjectKeywordedSecretString,
 InjectSecretString

cache = SecretCache()

@InjectSecretString ( 'mysecret' ,  cache )
def function_to_be_decorated( arg1,  arg2,  arg3):
```

@InjectKeywordedSecretString

이 데코레이터는 첫 번째와 두 번째 인수로 보안 암호 ID 문자열과 the section called “SecretCache”을(를) 기대합니다. 나머지 인수는 래핑된 함수의 파라미터를 보안 암호의 JSON 키로 매핑합니다. 보안 암호에는 JSON 구조의 문자열이 포함되어야 합니다.

이 JSON이 포함된 보안 암호의 경우:

```
{
  "username": "saanvi",
  "password": "EXAMPLE-PASSWORD"
}
```

다음 예에서는 보안 암호에서 username과 password에 대한 JSON 값을 추출하는 방법을 보여줍니다.

```
from aws_secretsmanager_caching import SecretCache
  from aws_secretsmanager_caching import InjectKeywordedSecretString,
 InjectSecretString

  cache = SecretCache()

  @InjectKeywordedSecretString ( secret_id = 'mysecret' ,  cache = cache ,
 func_username = 'username' ,  func_password = 'password' )
  def function_to_be_decorated( func_username,  func_password):
       print( 'Do something with the func_username and func_password parameters')
```

Python AWS SDK를 사용하여 Secrets Manager 보안 암호 값 가져오기

애플리케이션에서 GetSecretValue 또는 SDKBatchGetSecretValue를 호출하여 보안 암호를 검색할 수 있습니다. AWS SDKs 그러나 클라이언트 측 캐싱을 사용하여 보안 암호 값을 캐싱하는 것이 좋습니다. 보안 암호 캐싱은 속도를 향상시키고 비용을 절감합니다.

Python 애플리케이션의 경우 Secrets Manager Python 기반 캐싱 구성 요소를 사용하거나 get_secret_value 또는 batch_get_secret_value를 사용하여 SDK를 직접 호출합니다.

다음 코드 예시는 GetSecretValue의 사용 방법을 보여 줍니다.

필요한 권한:secretsmanager:GetSecretValue

```
"""
Purpose

Shows how to use the AWS SDK for Python (Boto3) with AWS
Secrets Manager to get a specific of secrets that match a
specified name
"""
import boto3
import logging

from get_secret_value import GetSecretWrapper

# Configure logging
logging.basicConfig(level=logging.INFO)
```

```
def run_scenario(secret_name):
    """
    Retrieve a secret from AWS Secrets Manager.

    :param secret_name: Name of the secret to retrieve.
    :type secret_name: str
    """
    try:
        # Validate secret_name
        if not secret_name:
            raise ValueError("Secret name must be provided.")
        # Retrieve the secret by name
        client = boto3.client("secretsmanager")
        wrapper = GetSecretWrapper(client)
        secret = wrapper.get_secret(secret_name)
        # Note: Secrets should not be logged.
        return secret
    except Exception as e:
        logging.error(f"Error retrieving secret: {e}")
        raise

class GetSecretWrapper:
    def __init__(self, secretsmanager_client):
        self.client = secretsmanager_client


    def get_secret(self, secret_name):
        """
        Retrieve individual secrets from AWS Secrets Manager using the get_secret_value
 API.
        This function assumes the stack mentioned in the source code README has been
 successfully deployed.
        This stack includes 7 secrets, all of which have names beginning with
 "mySecret".

        :param secret_name: The name of the secret fetched.
        :type secret_name: str
        """
        try:
            get_secret_value_response = self.client.get_secret_value(
                SecretId=secret_name
            )
            logging.info("Secret retrieved successfully.")
```

```
            return get_secret_value_response["SecretString"]
        except self.client.exceptions.ResourceNotFoundException:
            msg = f"The requested secret {secret_name} was not found."
            logger.info(msg)
            return msg
        except Exception as e:
            logger.error(f"An unknown error occurred: {str(e)}.")
            raise
```

Python AWS SDK를 사용하여 Secrets Manager 보안 암호 값 배치 가져오기

다음 코드 예시에서는 Secrets Manager 보안 암호 값의 배치를 가져오는 방법을 보여줍니다.

필요한 권한:

- `secretsmanager:BatchGetSecretValue`
- 검색할 각 보안 암호에 대한 `secretsmanager:GetSecretValue` 권한이 있어야 합니다.
- 필터를 사용하는 경우 `secretsmanager:ListSecrets`도 있어야 합니다.

권한 정책 예시는 the section called “예: 보안 암호 값 그룹을 일괄적으로 검색할 수 있는 권한” 섹션을 참조하세요.

> **Important**
>
> 검색 중인 그룹에서 개별 보안 암호를 검색할 수 있는 권한을 거부하는 VPCE 정책이 있는 경우 `BatchGetSecretValue`는 보안 암호 값을 반환하지 않고 오류를 반환합니다.

```
class BatchGetSecretsWrapper:
    def __init__(self, secretsmanager_client):
        self.client = secretsmanager_client


    def batch_get_secrets(self, filter_name):
        """
        Retrieve multiple secrets from AWS Secrets Manager using the
 batch_get_secret_value API.
```

```
        This function assumes the stack mentioned in the source code README has been
successfully deployed.
        This stack includes 7 secrets, all of which have names beginning with
"mySecret".

        :param filter_name: The full or partial name of secrets to be fetched.
        :type filter_name: str
        """
        try:
            secrets = []
            response = self.client.batch_get_secret_value(
                Filters=[{"Key": "name", "Values": [f"{filter_name}"]}]
            )
            for secret in response["SecretValues"]:
                secrets.append(json.loads(secret["SecretString"]))
            if secrets:
                logger.info("Secrets retrieved successfully.")
            else:
                logger.info("Zero secrets returned without error.")
            return secrets
        except self.client.exceptions.ResourceNotFoundException:
            msg = f"One or more requested secrets were not found with filter:
{filter_name}"
            logger.info(msg)
            return msg
        except Exception as e:
            logger.error(f"An unknown error occurred:\n{str(e)}.")
            raise
```

.NET을 사용하여 Secrets Manager 보안 암호 값 가져오기

애플리케이션에서 GetSecretValue 또는 SDKBatchGetSecretValue를 호출하여 보안 암호를 검색할 수 있습니다. AWS SDKs 그러나 클라이언트 측 캐싱을 사용하여 보안 암호 값을 캐싱하는 것이 좋습니다. 보안 암호 캐싱은 속도를 향상시키고 비용을 절감합니다.

주제

- 클라이언트 측 캐싱과 함께 .NET을 사용하여 Secrets Manager 보안 암호 값 가져오기
- 를 사용하여 Secrets Manager 보안 암호 값 가져오기 SDK for .NET

클라이언트 측 캐싱과 함께 .NET을 사용하여 Secrets Manager 보안 암호 값 가져오기

보안 암호를 검색할 때 Secret Manager .NET 기반 캐싱 구성 요소를 사용하여 나중에 사용할 수 있도록 캐싱할 수 있습니다. 캐싱된 보안 암호를 검색하는 것이 Secret Manager에서 검색하는 것보다 빠릅니다. Secrets Manager API를 호출하는 데는 비용이 발생하므로 캐시를 사용하면 비용을 줄일 수 있습니다. 암호를 검색할 수 있는 모든 방법은 보안 암호 가져오기을(를) 참조하세요.

캐시 정책은 LRU(가장 오랫동안 사용되지 않음)이므로, 캐시에서 보안 암호를 폐기해야 하는 경우 가장 오랫동안 사용되지 않은 보안 암호가 삭제됩니다. 기본적으로 캐시는 보안 암호를 매시간 새로 고칩니다. 캐시에서 보안 암호를 새로 고치는 주기를 구성하고 보안 암호 검색에 연결하여 더 많은 기능을 추가할 수 있습니다.

캐시 참조가 해제되면 캐시는 가비지 수집을 강제로 적용하지 않습니다. 캐시 구현에는 캐시 무효화가 포함되지 않습니다. 캐시 구현은 캐시 자체에 중점을 두며, 보안을 강화하거나 보안에 초점을 맞추지 않습니다. 캐시에서 항목 암호화와 같은 추가 보안이 필요한 경우 제공된 인터페이스 및 추상 메서드를 사용하세요.

이 구성 요소를 사용하려면 다음이 필요합니다.

- .NET Framework 4.6.2 이상 또는 .NET Standard 2.0 이상. Microsoft .NET 웹 사이트의 .NET 다운로드를 참조하세요.
- .NET용 AWS SDK. the section called “AWS SDKs”을(를) 참조하세요.

소스 코드를 다운로드하려면 GitHub의 .NET용 캐싱 클라이언트를 참조하세요.

캐시를 사용하려면 먼저 캐시를 인스턴스화한 다음 `GetSecretString` 또는 `GetSecretBinary`를 사용하여 보안 암호를 검색합니다. 연속적인 검색에서 캐시는 보안 암호의 캐싱된 사본을 반환합니다.

캐싱 패키지를 가져오려면

- 다음 중 하나를 수행하세요.
  - 프로젝트 디렉터리에서 다음 .NET CLI 명령을 실행합니다.

    ```
    dotnet add package AWSSDK.SecretsManager.Caching --version 1.0.6
    ```

  - `.csproj` 파일에 다음 패키지 참조를 추가합니다.

    ```
    <ItemGroup>
    ```

```
    <PackageReference Include="AWSSDK.SecretsManager.Caching" Version="1.0.6" /
>
</ItemGroup>
```

필요한 권한:

- `secretsmanager:DescribeSecret`
- `secretsmanager:GetSecretValue`

자세한 내용은 권한 참조 단원을 참조하십시오.

레퍼런스

- SecretsManagerCache
- SecretCacheConfiguration
- ISecretCacheHook

Example보안 암호 검색

다음 코드 예제에서는 *MySecret*이라는 이름의 보안 암호를 검색하는 메서드를 보여 줍니다.

```
using Amazon.SecretsManager.Extensions.Caching;

namespace LambdaExample
{
    public class CachingExample
    {
        private const string MySecretName ="MySecret";

        private SecretsManagerCache cache = new SecretsManagerCache();

        public async Task<Response>  FunctionHandlerAsync(string input, ILambdaContext
 context)
        {
            string MySecret = await cache.GetSecretString(MySecretName);

            // Use the secret, return success

        }
    }
```

```
}
```

Example TTL(Time To Live) 캐시 새로 고침 기간 구성

다음 코드 예제에서는 *MySecret*이라는 보안 암호를 검색하고 TTL 캐시 새로 고침 기간을 24시간으로 설정하는 메서드를 보여줍니다.

```
using Amazon.SecretsManager.Extensions.Caching;

namespace LambdaExample
{
    public class CachingExample
    {
        private const string MySecretName = "MySecret";

        private static SecretCacheConfiguration cacheConfiguration = new
 SecretCacheConfiguration
        {
            CacheItemTTL = 86400000
        };
        private SecretsManagerCache cache = new
 SecretsManagerCache(cacheConfiguration);
        public async Task<Response> FunctionHandlerAsync(string input, ILambdaContext
 context)
        {
            string mySecret = await cache.GetSecretString(MySecretName);

            // Use the secret, return success
        }
    }
}
```

SecretsManagerCache

Secret Manager에서 요청한 보안 암호에 대한 인 메모리 캐시. the section called “GetSecretString” 또는 the section called “GetSecretBinary”를 사용하여 캐시에서 보안 암호를 검색합니다. 생성자의 the section called “SecretCacheConfiguration” 객체에 전달하여 캐시 설정을 구성할 수 있습니다.

예제를 포함한 자세한 내용은 the section called “클라이언트 측 캐싱을 사용한 .NET”을 참조하세요.

Constructors

`public SecretsManagerCache()`

SecretsManagerCache 객체에 대한 기본 생성자.

`public SecretsManagerCache(IAmazonSecretsManager secretsManager)`

제공된 AmazonSecretsManagerClient로 생성한 Secrets Manager 클라이언트를 사용하여 새 캐시를 생성합니다. 이 생성자를 사용하여 Secrets Manager 클라이언트를 사용자 지정합니다(예: 특정 리전 또는 엔드포인트를 사용하도록 사용자 지정).

파라미터

secretsManager

보안 암호를 검색할 AmazonSecretsManagerClient.

`public SecretsManagerCache(SecretCacheConfiguration config)`

제공된 the section called “SecretCacheConfiguration”을 사용하여 새 보안 암호 캐시를 생성합니다. 이 생성자를 사용하여 캐시를 구성합니다(예: 캐싱할 보안 암호 수 및 새로 고침 빈도).

파라미터

config

캐시에 대한 구성 정보를 포함한 the section called “SecretCacheConfiguration”.

```
public SecretsManagerCache(IAmazonSecretsManager secretsManager,
SecretCacheConfiguration config)
```

제공된 AmazonSecretsManagerClient 및 the section called “SecretCacheConfiguration”으로 생성한 Secrets Manager 클라이언트를 사용하여 새 캐시를 생성합니다. 이 생성자를 사용하여 Secrets Manager 클라이언트를 사용자 지정(예: 특정 리전 또는 엔드포인트를 사용하도록 사용자 지정)하거나 캐시를 구성합니다(예: 캐싱할 보안 암호 수 및 새로 고침 빈도).

파라미터

secretsManager

보안 암호를 검색할 AmazonSecretsManagerClient.

config

캐시에 대한 구성 정보를 포함한 the section called “SecretCacheConfiguration”.

메서드

GetSecretString

```
public async Task<String> GetSecretString(String secretId)
```

Secrets Manager에서 문자열 보안 암호를 검색합니다.

파라미터

secretId

검색할 보안 암호의 ARN 또는 이름입니다.

GetSecretBinary

```
public async Task<byte[]> GetSecretBinary(String secretId)
```

Secrets Manager에서 이진 보안 암호를 검색합니다.

파라미터

secretId

검색할 보안 암호의 ARN 또는 이름입니다.

RefreshNowAsync

```
public async Task<bool> RefreshNowAsync(String secretId)
```

Secrets Manager에서 보안 암호 값을 요청하고 변경 사항으로 캐시를 업데이트합니다. 기존 캐시 항목이 없는 경우 새 캐시 항목을 생성합니다. 새로 고침이 성공한 경우 `true`를 반환합니다.

파라미터

secretId

검색할 보안 암호의 ARN 또는 이름입니다.

GetCachedSecret

```
public SecretCacheItem GetCachedSecret(string secretId)
```

지정한 보안 암호가 캐시에 존재하는 경우 해당 암호에 대한 캐시 항목을 반환합니다. 그렇지 않으면 Secrets Manager에서 보안 암호를 검색하고 새 캐시 항목을 생성합니다.

파라미터

secretId

검색할 보안 암호의 ARN 또는 이름입니다.

SecretCacheConfiguration

캐싱된 보안 암호에 대한 최대 캐시 크기 및 유지 시간(TTL)과 같은 the section called "SecretsManagerCache"에 대한 캐시 구성 옵션.

속성

CacheItemTTL

```
public uint CacheItemTTL { get; set; }
```

밀리초 단위로 된 캐시 항목에 대한 TTL. 기본값은 3600000ms 또는 1시간입니다. 최댓값은 4294967295ms이며 약 49.7일입니다.

MaxCacheSize

```
public ushort MaxCacheSize { get; set; }
```

최대 캐시 크기. 기본값은 보안 암호 1,024개입니다. 최댓값은 65,535입니다.

VersionStage

```
public string VersionStage { get; set; }
```

캐싱할 보안 암호의 버전. 자세한 내용은 보안 암호 버전을 참조하세요. 기본값은 "AWSCURRENT"입니다.

클라이언트

```
public IAmazonSecretsManager Client { get; set; }
```

보안 암호를 검색할 AmazonSecretsManagerClient. null인 경우 캐시가 새 클라이언트를 인스턴스화합니다. 기본값은 null입니다.

CacheHook

```
public ISecretCacheHook CacheHook { get; set; }
```

the section called “ISecretCacheHook”.

ISecretCacheHook

캐시에 저장 중인 보안 암호에 대한 작업을 수행하기 위해 the section called “SecretsManagerCache”에 연결되는 인터페이스.

메서드

Put

```
object Put(object o);
```

캐시에 저장할 객체를 준비합니다.

캐시에 저장할 객체를 반환합니다.

Get

```
object Get(object cachedObject);
```

캐싱된 객체에서 객체를 추출합니다.

캐시에서 객체를 반환합니다.

를 사용하여 Secrets Manager 보안 암호 값 가져오기 SDK for .NET

애플리케이션에서 `GetSecretValue` 또는 SDK`BatchGetSecretValue`를 호출하여 보안 암호를 검색할 수 있습니다. AWS SDKs 그러나 클라이언트 측 캐싱을 사용하여 보안 암호 값을 캐싱하는 것이 좋습니다. 보안 암호 캐싱은 속도를 향상시키고 비용을 절감합니다.

.NET 애플리케이션의 경우 Secrets Manager .NET 기반 캐싱 구성 요소를 사용하거나 `GetSecretValue` 또는 `BatchGetSecretValue`를 사용하여 SDK를 직접 호출합니다.

다음 코드 예시는 `GetSecretValue`의 사용 방법을 보여 줍니다.

필요한 권한:`secretsmanager:GetSecretValue`

```
    using System;
    using System.IO;
    using System.Threading.Tasks;
```

```
using Amazon.SecretsManager;
using Amazon.SecretsManager.Model;

/// <summary>
/// This example uses the Amazon Web Service Secrets Manager to retrieve
/// the secret value for the provided secret name.
/// </summary>
public class GetSecretValue
{
    /// <summary>
    /// The main method initializes the necessary values and then calls
    /// the GetSecretAsync and DecodeString methods to get the decoded
    /// secret value for the secret named in secretName.
    /// </summary>
    public static async Task Main()
    {
        string secretName = "<<{{MySecretName}}>>";
        string secret;

        IAmazonSecretsManager client = new AmazonSecretsManagerClient();

        var response = await GetSecretAsync(client, secretName);

        if (response is not null)
        {
            secret = DecodeString(response);

            if (!string.IsNullOrEmpty(secret))
            {
                Console.WriteLine($"The decoded secret value is: {secret}.");
            }
            else
            {
                Console.WriteLine("No secret value was returned.");
            }
        }
    }

    /// <summary>
    /// Retrieves the secret value given the name of the secret to
    /// retrieve.
    /// </summary>
    /// <param name="client">The client object used to retrieve the secret
    /// value for the given secret name.</param>
```

```
        /// <param name="secretName">The name of the secret value to retrieve.</param>
        /// <returns>The GetSecretValueReponse object returned by
        /// GetSecretValueAsync.</returns>
        public static async Task<GetSecretValueResponse> GetSecretAsync(
            IAmazonSecretsManager client,
            string secretName)
        {
            GetSecretValueRequest request = new GetSecretValueRequest()
            {
                SecretId = secretName,
                VersionStage = "AWSCURRENT", // VersionStage defaults to AWSCURRENT if
unspecified.
            };

            GetSecretValueResponse response = null;

            // For the sake of simplicity, this example handles only the most
            // general SecretsManager exception.
            try
            {
                response = await client.GetSecretValueAsync(request);
            }
            catch (AmazonSecretsManagerException e)
            {
                Console.WriteLine($"Error: {e.Message}");
            }

            return response;
        }

        /// <summary>
        /// Decodes the secret returned by the call to GetSecretValueAsync and
        /// returns it to the calling program.
        /// </summary>
        /// <param name="response">A GetSecretValueResponse object containing
        /// the requested secret value returned by GetSecretValueAsync.</param>
        /// <returns>A string representing the decoded secret value.</returns>
        public static string DecodeString(GetSecretValueResponse response)
        {
            // Decrypts secret using the associated AWS Key Management Service
            // Customer Master Key (CMK.) Depending on whether the secret is a
            // string or binary value, one of these fields will be populated.
            if (response.SecretString is not null)
            {
```

```
                var secret = response.SecretString;
                return secret;
            }
            else if (response.SecretBinary is not null)
            {
                var memoryStream = response.SecretBinary;
                StreamReader reader = new StreamReader(memoryStream);
                string decodedBinarySecret =
System.Text.Encoding.UTF8.GetString(Convert.FromBase64String(reader.ReadToEnd()));
                return decodedBinarySecret;
            }
            else
            {
                return string.Empty;
            }
        }
    }
```

Go를 사용하여 Secrets Manager 보안 암호 값 가져오기

애플리케이션에서 `GetSecretValue` 또는 SDK`BatchGetSecretValue`를 호출하여 보안 암호를 검색할 수 있습니다. AWS SDKs 그러나 클라이언트 측 캐싱을 사용하여 보안 암호 값을 캐싱하는 것이 좋습니다. 보안 암호 캐싱은 속도를 향상시키고 비용을 절감합니다.

주제

- 클라이언트 측 캐싱과 함께 Go를 사용하여 Secrets Manager 보안 암호 값 가져오기
- Go AWS SDK를 사용하여 Secrets Manager 보안 암호 값 가져오기

클라이언트 측 캐싱과 함께 Go를 사용하여 Secrets Manager 보안 암호 값 가져오기

보안 암호를 검색할 때 Secret Manager Go 기반 캐싱 구성 요소를 사용하여 나중에 사용할 수 있도록 캐싱할 수 있습니다. 캐싱된 보안 암호를 검색하는 것이 Secret Manager에서 검색하는 것보다 빠릅니다. Secrets Manager API를 호출하는 데는 비용이 발생하므로 캐시를 사용하면 비용을 줄일 수 있습니다. 암호를 검색할 수 있는 모든 방법은 보안 암호 가져오기을(를) 참조하세요.

캐시 정책은 LRU(가장 오랫동안 사용되지 않음)이므로, 캐시에서 보안 암호를 폐기해야 하는 경우 가장 오랫동안 사용되지 않은 보안 암호가 삭제됩니다. 기본적으로 캐시는 보안 암호를 매시간 새로 고칩

니다. 캐시에서 보안 암호를 새로 고치는 주기를 구성하고 보안 암호 검색에 연결하여 더 많은 기능을 추가할 수 있습니다.

캐시 참조가 해제되면 캐시는 가비지 수집을 강제로 적용하지 않습니다. 캐시 구현에는 캐시 무효화가 포함되지 않습니다. 캐시 구현은 캐시 자체에 중점을 두며, 보안을 강화하거나 보안에 초점을 맞추지 않습니다. 캐시에서 항목 암호화와 같은 추가 보안이 필요한 경우 제공된 인터페이스 및 추상 메서드를 사용하세요.

이 구성 요소를 사용하려면 다음이 필요합니다.

- AWS Go용 SDK. the section called “AWS SDKs”을(를) 참조하세요.

소스 코드를 다운로드하려면 GitHub의 Secrets Manager Go 캐싱 클라이언트를 참조하세요.

Go 개발 환경을 설정하려면 Go Programming Language 웹 사이트의 Golang 시작하기를 참조하세요.

필요한 권한:

- `secretsmanager:DescribeSecret`
- `secretsmanager:GetSecretValue`

자세한 내용은 권한 참조 단원을 참조하십시오.

레퍼런스

- type Cache
- type CacheConfig
- type CacheHook

Example보안 암호 검색

다음 코드 예제에서는 보안 암호를 검색하는 Lambda 함수를 보여줍니다.

```
package main

import (
  "github.com/aws/aws-lambda-go/lambda"
  "github.com/aws/aws-secretsmanager-caching-go/secretcache"
)
```

```
var (
  secretCache, _ = secretcache.New()
)

func HandleRequest(secretId string) string {
  result, _ := secretCache.GetSecretString(secretId)

  // Use the secret, return success
}

 func main() {
  lambda. Start( HandleRequest)
}
```

type Cache

Secret Manager에서 요청한 보안 암호에 대한 인 메모리 캐시. the section called “GetSecretString” 또는 the section called “GetSecretBinary”를 사용하여 캐시에서 보안 암호를 검색합니다.

다음 예제에서는 캐시 설정을 구성하는 방법을 보여줍니다.

```
// Create a custom secretsmanager client
client := getCustomClient()

// Create a custom CacheConfig struct
config := secretcache. CacheConfig{
    MaxCacheSize:  secretcache.DefaultMaxCacheSize + 10,
    VersionStage:  secretcache.DefaultVersionStage,
    CacheItemTTL:  secretcache.DefaultCacheItemTTL,
}

// Instantiate the cache
cache, _ := secretcache.New(
    func( c *secretcache.Cache) {  c. CacheConfig = config },
    func( c *secretcache.Cache) {  c. Client = client },
)
```

예제를 포함한 자세한 내용은 the section called “클라이언트 측 캐싱을 사용한 Go”을 참조하세요.

메서드

New

```
func New(optFns ...func(*Cache)) (*Cache, error)
```

New는 함수 옵션을 사용하여 보안 암호 캐시를 구성하고, 그렇지 않으면 기본값을 사용합니다. 새 세션에서 SecretsManager 클라이언트를 초기화합니다. CacheConfig를 기본값으로 초기화합니다. LRU 캐시를 기본 최대 크기로 초기화합니다.

GetSecretString

```
func (c *Cache) GetSecretString(secretId string) (string, error)
```

GetSecretString은 주어진 보안 암호 ID에 대한 캐시에서 보안 암호 문자열 값을 가져옵니다. 작업이 실패하면 보안 암호 문자열과 오류를 반환합니다.

GetSecretStringWithStage

```
func (c *Cache) GetSecretStringWithStage(secretId string, versionStage
string) (string, error)
```

GetSecretStringWithStage는 주어진 보안 암호 ID와 버전 단계에 대한 캐시에서 보안 암호 문자열 값을 가져옵니다. 작업이 실패하면 보안 암호 문자열과 오류를 반환합니다.

GetSecretBinary

```
func (c *Cache) GetSecretBinary(secretId string) ([]byte, error) {
```

GetSecretBinary는 주어진 보안 암호 ID에 대한 캐시에서 보안 암호 이진 값을 가져옵니다. 작업이 실패하면 보안 암호 이진 값과 오류를 반환합니다.

GetSecretBinaryWithStage

```
func (c *Cache) GetSecretBinaryWithStage(secretId string, versionStage
string) ([]byte, error)
```

GetSecretBinaryWithStage는 주어진 보안 암호 ID와 버전 단계에 대한 캐시에서 보안 암호 이진 값을 가져옵니다. 작업이 실패하면 보안 암호 이진 값과 오류를 반환합니다.

type CacheConfig

캐싱된 보안 암호에 대한 최대 캐시 크기, 기본 버전 단계 및 유지 시간(TTL)과 같은 캐시에 대한 캐시 구성 옵션.

```
type CacheConfig struct {

    // The maximum cache size. The default is 1024 secrets.
    MaxCacheSize int

    // The TTL of a cache item in nanoseconds. The default is
    // 3.6e10^12 ns or 1 hour.
    CacheItemTTL int64

    // The version of secrets that you want to cache. The default
    // is "AWSCURRENT".
    VersionStage string

    // Used to hook in-memory cache updates.
    Hook CacheHook
}
```

type CacheHook

캐시에 저장 중인 보안 암호에 대한 작업을 수행하기 위해 캐시에 연결되는 인터페이스.

메서드

Put

`Put(data interface{}) interface{}`

캐시에 저장할 객체를 준비합니다.

Get

`Get(data interface{}) interface{}`

캐싱된 객체에서 객체를 추출합니다.

Go AWS SDK를 사용하여 Secrets Manager 보안 암호 값 가져오기

애플리케이션에서 `GetSecretValue` 또는 SDK`BatchGetSecretValue`를 호출하여 보안 암호를 검색할 수 있습니다. AWS SDKs 그러나 클라이언트 측 캐싱을 사용하여 보안 암호 값을 캐싱하는 것이 좋습니다. 보안 암호 캐싱은 속도를 향상시키고 비용을 절감합니다.

Go 애플리케이션의 경우 Secrets Manager Go 기반 캐싱 구성 요소를 사용하거나 `GetSecretValue` 또는 `BatchGetSecretValue`를 사용하여 SDK를 직접 호출합니다.

다음 코드 예제에서는 Secrets Manager 보안 암호 값을 가져오는 방법을 보여줍니다.

필요한 권한:secretsmanager:GetSecretValue

```
 // Use this code snippet in your app.
 // If you need more information about configurations or implementing the sample code,
visit the AWS docs:
 // https://aws.github.io/aws-sdk-go-v2/docs/getting-started/

 import (
   "context"
   "log"

   "github.com/aws/aws-sdk-go-v2/aws"
   "github.com/aws/aws-sdk-go-v2/config"
   "github.com/aws/aws-sdk-go-v2/service/secretsmanager"
 )

 func main() {
   secretName := "<<{{MySecretName}}>>"
   region := "<<{{MyRegionName}}>>"

   config, err := config.LoadDefaultConfig(context.TODO(), config.WithRegion(region))
   if err != nil {
     log.Fatal(err)
   }

   // Create Secrets Manager client
   svc := secretsmanager.NewFromConfig(config)

   input := &secretsmanager.GetSecretValueInput{
     SecretId:     aws.String(secretName),
     VersionStage: aws.String("AWSCURRENT"), // VersionStage defaults to AWSCURRENT if
unspecified
   }

   result, err := svc.GetSecretValue(context.TODO(), input)
   if err != nil {
     // For a list of exceptions thrown, see
     // https://<<{{DocsDomain}}>>/secretsmanager/latest/apireference/
API_GetSecretValue.html
     log.Fatal(err.Error())
   }
```

```
    // Decrypts secret using the associated KMS key.
    var secretString string = *result.SecretString

    // Your code goes here.
}
```

Rust를 사용하여 Secrets Manager 보안 암호 값 가져오기

애플리케이션에서 `GetSecretValue` 또는 SDK`BatchGetSecretValue`를 호출하여 보안 암호를 검색할 수 있습니다. AWS SDKs 그러나 클라이언트 측 캐싱을 사용하여 보안 암호 값을 캐싱하는 것이 좋습니다. 보안 암호 캐싱은 속도를 향상시키고 비용을 절감합니다.

주제

- 클라이언트 측 캐싱과 함께 Rust를 사용하여 Secrets Manager 보안 암호 값 가져오기
- Rust AWS SDK를 사용하여 Secrets Manager 보안 암호 값 가져오기

클라이언트 측 캐싱과 함께 Rust를 사용하여 Secrets Manager 보안 암호 값 가져오기

보안 암호를 검색할 때 Secret Manager Rust 기반 캐싱 구성 요소를 사용하여 나중에 사용할 수 있도록 캐싱할 수 있습니다. 캐싱된 보안 암호를 검색하는 것이 Secret Manager에서 검색하는 것보다 빠릅니다. Secrets Manager API를 호출하는 데는 비용이 발생하므로 캐시를 사용하면 비용을 줄일 수 있습니다. 암호를 검색할 수 있는 모든 방법은 보안 암호 가져오기을(를) 참조하세요.

캐시 정책은 FIFO(선입선출)이므로 캐시가 보안 암호를 폐기해야 할 경우 가장 오래된 보안 암호가 폐기됩니다. 기본적으로 캐시는 보안 암호를 매시간 새로 고칩니다. 다음을 구성할 수 있습니다.

- `max_size` - 최근에 액세스하지 않은 보안 암호를 제거하기 전까지 유지할 수 있는 캐시된 보안 암호의 최대 수입니다.
- `ttl` - 보안 암호 상태를 새로 고침해야 하기 전까지 캐시된 항목이 유효하다고 간주되는 기간입니다.

캐시 구현에는 캐시 무효화가 포함되지 않습니다. 캐시 구현은 캐시 자체에 중점을 두며, 보안을 강화하거나 보안에 초점을 맞추지 않습니다. 캐시에서 항목 암호화와 같은 추가 보안이 필요한 경우 제공된 특성을 사용하여 캐시를 수정하세요.

구성 요소를 사용하려면 tokio가 포함된 Rust 2021 개발 환경이 있어야 합니다. 자세한 내용은 Rust 프로그래밍 언어 웹 사이트에서 시작하기를 참조하세요.

소스 코드를 다운로드하려면 GitHub의 Secrets Manager Rust 기반 캐싱 클라이언트 구성 요소를 참조하세요.

캐싱 구성 요소를 설치하려면 다음 명령을 사용합니다.

```
cargo add aws_secretsmanager_caching
```

필요한 권한:

- secretsmanager:DescribeSecret
- secretsmanager:GetSecretValue

자세한 내용은 권한 참조 단원을 참조하십시오.

Example보안 암호 검색

다음 예에서는 *MyTest*라는 보안 암호에 대한 보안 암호 값을 가져오는 방법을 보여 줍니다.

```
use aws_secretsmanager_caching::SecretsManagerCachingClient;
use std::num::NonZeroUsize;
use std::time::Duration;

let client = match SecretsManagerCachingClient::default(
    NonZeroUsize::new(10).unwrap(),
    Duration::from_secs(60),
)
.await
{
    Ok(c) => c,
    Err(_) => panic!("Handle this error"),
};

let secret_string = match client.get_secret_value("MyTest", None, None).await {
    Ok(s) => s.secret_string.unwrap(),
    Err(_) => panic!("Handle this error"),
};

// Your code here
```

Example사용자 지정 구성 및 사용자 지정 클라이언트를 사용하여 캐시 인스턴스화

다음 예에서는 캐시를 구성한 다음, *MyTest*라는 보안 암호에 대한 보안 암호 값을 가져오는 방법을 보여 줍니다.

```
let config = aws_config::load_defaults(BehaviorVersion::latest())
    .await
    .into_builder()
    .region(Region::from_static("us-west-2"))
    .build();

let asm_builder = aws_sdk_secretsmanager::config::Builder::from(&config);

let client = match SecretsManagerCachingClient::from_builder(
    asm_builder,
    NonZeroUsize::new(10).unwrap(),
    Duration::from_secs(60),
)
.await
{
    Ok(c) => c,
    Err(_) => panic!("Handle this error"),
};

let secret_string = client
    .get_secret_value("MyTest", None, None)
    .await
    {
        Ok(c) => c.secret_string.unwrap(),
        Err(_) => panic!("Handle this error"),
    };

// Your code here
```

Rust AWS SDK를 사용하여 Secrets Manager 보안 암호 값 가져오기

애플리케이션에서 `GetSecretValue` 또는 SDK`BatchGetSecretValue`를 호출하여 보안 암호를 검색할 수 있습니다. AWS SDKs 그러나 클라이언트 측 캐싱을 사용하여 보안 암호 값을 캐싱하는 것이 좋습니다. 보안 암호 캐싱은 속도를 향상시키고 비용을 절감합니다.

Rust 애플리케이션의 경우 Secrets Manager Go 기반 캐싱 구성 요소를 사용하거나 GetSecretValue 또는 BatchGetSecretValue를 사용하여 SDK를 직접 호출합니다.

다음 코드 예제에서는 Secrets Manager 보안 암호 값을 가져오는 방법을 보여줍니다.

필요한 권한:`secretsmanager:GetSecretValue`

```
async fn show_secret(client: &Client, name: &str) -> Result<(), Error> {
    let resp = client.get_secret_value().secret_id(name).send().await?;

    println!("Value: {}", resp.secret_string().unwrap_or("No value!"));

    Ok(())
}
```

Amazon Elastic Kubernetes Service에서 AWS Secrets Manager 보안 암호 사용

AWS Secrets Manager (ASCP)의 보안 암호를 Amazon EKS 포드에 탑재된 파일로 표시하려면 Kubernetes AWS 보안 암호 스토어 CSI 드라이버의 보안 암호 및 구성 공급자를 사용할 수 있습니다. ASCP는 Amazon EC2 노드 그룹을 실행하는 Amazon Elastic Kubernetes Service 1.17 이상에서 작동합니다. AWS Fargate 노드 그룹은 지원되지 않습니다. ASCP를 사용하면 Secrets Manager 보안 암호를 저장 및 관리한 후 Amazon EKS에서 실행되는 워크로드를 통해 해당 검색할 수 있습니다. 보안 암호에 JSON 형식의 여러 키/값 쌍이 포함되어 있는 경우 Amazon EKS에서 탑재할 키/값 쌍을 선택할 수 있습니다. ASCP는 JMESPath 구문을 사용하여 보안 암호의 키-값 쌍을 쿼리할 수 있습니다. ASCP는 파라미터 스토어 파라미터로도 작동합니다. ASCP는 Amazon EKS를 사용한 두 가지 인증 방법을 제공합니다. 첫 번째 접근 방식은 IRSA(서비스 계정에 대한 IAM 역할)를 사용합니다. 두 번째 접근 방식은 Pod Identity를 사용합니다. 각 접근 방식에는 이점과 사용 사례가 있습니다.

IRSA(서비스 계정에 대한 IAM 역할)를 사용하는 ASCP

서비스 계정에 대한 IAM 역할(IRSA)이 있는 ASCP를 사용하면의 보안 암호를 Amazon EKS 포드의 AWS Secrets Manager 파일로 탑재할 수 있습니다. 이 접근 방식이 적합한 경우:

- 보안 암호를 포드에 파일로 탑재해야 하는 경우
- Amazon EC2 노드 그룹에서 Amazon EKS 버전 1.17 이상을 사용하고 있는 경우
- JSON 형식 보안 암호에서 특정 키-값 쌍을 검색하려는 경우

자세한 내용은 the section called “IRSA for Amazon EKS와 ASCP 통합” 단원을 참조하십시오.

Pod Identity를 사용하는 ASCP

EKS Pod Identity를 사용하는 ASCP

Pod Identity를 사용하는 ASCP 방법은 보안을 강화하고, Amazon EKS에서 파라미터 액세스 구성을 간소화합니다. 이 접근 방식이 유용한 경우:

- 포드 수준에서 더 세분화된 권한 관리가 필요합니다.
- Amazon EKS 버전 1.24 이상을 사용하고 있습니다.
- 성능 및 확장성 개선을 원합니다.

자세한 내용은 the section called “Pod Identity for Amazon EKS와 ASCP 통합” 단원을 참조하십시오.

올바른 접근 방식 선택

IRSA를 사용하는 ASCP와 Pod Identity를 사용하는 ASCP 중에서 결정할 때 고려할 요소:

- Amazon EKS 버전: Pod Identity의 경우 Amazon EKS 1.24 이상이 필요한 반면 CSI 드라이버는 Amazon EKS 1.17 이상을 사용합니다.
- 보안 요구 사항: Pod Identity는 포드 수준에서 더 세분화된 제어가 가능합니다.
- 성능: Pod Identity는 일반적으로 대규모 환경에서 더 나은 성능을 발휘합니다.
- 복잡성: Pod Identity는 별도의 서비스 계정이 필요하지 않아 설정이 간소화됩니다.

특정 요구 사항 및 Amazon EKS 환경에 가장 적합한 방법을 선택합니다.

ASCP for Amazon EKS 설치

이 섹션에서는 Amazon EKS용 AWS 보안 암호 및 구성 공급자를 설치하는 방법을 설명합니다. ASCP를 사용하면 Secrets Manager의 보안 암호와의 파라미터를 Amazon EKS 포드의 AWS Systems Manager 파일로 탑재할 수 있습니다.

사전 조건

- Amazon EKS 클러스터
  - Pod Identity 버전 1.24 이상

  - IRSA 버전 1.17 이상
- AWS CLI 설치 및 구성된
- Amazon EKS 클러스터에 kubectl 설치 및 구성
- Helm(버전 3.0 이상)

ASCP 설치 및 구성

ASCP는 secrets-store-csi-provider-aws 리포지토리의 GitHub에서 제공됩니다. 리포지토리에는 보안 암호를 생성하고 탑재하기 위한 예제 YAML 파일도 포함되어 있습니다.

설치 과정에서, FIPS 엔드포인트를 사용하도록 ASCP를 구성할 수 있습니다. 엔드포인트 목록은 the section called “Secrets Manager 엔드포인트” 섹션을 참조하세요.

ASCP를 EKS 추가 기능으로 설치하려면

1. 설치`eksctl`(설치 지침)
2. 다음 명령을 실행하여 기본 구성으로 추가 기능을 설치합니다.

   ```
   eksctl create addon --cluster <your_cluster> --name aws-secrets-store-csi-driver-provider
   ```

   추가 기능을 구성하려면 대신 다음 설치 명령을 실행합니다.

   ```
   aws eks create-addon --cluster-name <your_cluster> --addon-name aws-secrets-store-csi-driver-provider --configuration-values 'file://path/to/config.yaml'
   ```

   구성 파일은 YAML 또는 JSON 파일일 수 있습니다. 추가 기능의 구성 스키마를 보려면:

   a. 다음 명령을 실행하고 추가 기능의 최신 버전을 기록해 둡니다.

      ```
      aws eks describe-addon-versions --addon-name aws-secrets-store-csi-driver-provider
      ```

   b. 다음 명령을 실행하여 추가 기능의 구성 스키마를 확인하고를 이전 단계의 버전으로 `<version>` 바꿉니다.

      ```
      aws eks describe-addon-configuration --addon-name aws-secrets-store-csi-driver-provider --addon-version <version>
      ```

Helm을 사용하여 ASCP를 설치하는 방법

1. 리포지토리가 최신 차트를 가리키고 있는지 확인하려면 `helm repo update.`를 사용합니다.
2. 차트를 설치합니다. 다음은 `helm install` 명령의 예입니다.

```
helm install -n kube-system secrets-provider-aws aws-secrets-manager/secrets-store-csi-driver-provider-aws
```

   a. FIPS 엔드포인트를 사용하려면 다음 플래그를 추가합니다. `--set useFipsEndpoint=true`
   b. 스로틀링을 구성하려면 다음 플래그를 추가합니다. `--set-json 'k8sThrottlingParams={"qps": "number of queries per second", "burst": "number of queries per second"}'`
   c. Secrets Store CSI 드라이버가 클러스터에 이미 설치되어 있는 경우 플래그(`--set secrets-store-csi-driver.install=false`)를 추가합니다. 이 플래그를 사용하면 Secrets Store CSI Driver를 종속 항목으로 설치하지 않습니다.

리포지토리에서 YAML을 사용하여 설치하는 방법

- 다음 명령을 사용합니다.

```
helm repo add secrets-store-csi-driver https://kubernetes-sigs.github.io/secrets-store-csi-driver/charts
helm install -n kube-system csi-secrets-store secrets-store-csi-driver/secrets-store-csi-driver
kubectl apply -f https://raw.githubusercontent.com/aws/secrets-store-csi-driver-provider-aws/main/deployment/aws-provider-installer.yaml
```

설치 확인

EKS 클러스터, Secrets Store CSI 드라이버 및 ASCP 플러그인의 설치를 확인하려면 다음 단계를 따릅니다.

1. EKS 클러스터 확인:

```
eksctl get cluster --name clusterName
```

이 명령을 사용하면 클러스터에 대한 정보가 반환되어야 합니다.

2. Secrets Store CSI 드라이버 설치를 확인합니다.

```
kubectl get pods -n kube-system -l app=secrets-store-csi-driver
```

`csi-secrets-store-secrets-store-csi-driver-xxx`와 같이 이름과 함께 실행 중인 포드가 표시될 것입니다.

3. ASCP 플러그인 설치 확인:

YAML installation

```
$ kubectl get pods -n kube-system -l app=csi-secrets-store-provider-aws
```

출력 예시:

```
NAME                                     READY   STATUS    RESTARTS   AGE
csi-secrets-store-provider-aws-12345      1/1     Running   0          2m
```

Helm installation

```
$  kubectl get pods -n kube-system -l app=secrets-store-csi-driver-provider-aws
```

출력 예시:

```
NAME                                                READY   STATUS    RESTARTS
 AGE
secrets-provider-aws-secrets-store-csi-driver-provider-67890       1/1
 Running   0          2m
```

포드가 `Running` 상태로 표시될 것입니다.

명령을 실행한 후 모든 것이 올바르게 설정되면 오류 없이 실행 중인 모든 구성 요소가 표시됩니다. 문제가 발생하면 문제가 있는 특정 포드의 로그를 확인하여 문제를 해결해야 할 수 있습니다.

문제 해결

1. 다음을 실행하여 ASCP 공급자의 로그 확인:

```
kubectl logs -n kube-system -l app=csi-secrets-store-provider-aws
```

2. `kube-system` 네임스페이스에서 모든 포드의 상태를 확인합니다.

```
kubectl -n kube-system get pods
```

```
kubectl -n kube-system logs pod/PODID
```

CSI 드라이버 및 ASCP와 관련된 모든 포드는 'Running' 상태여야 합니다.

3. CSI 드라이버 버전 확인:

```
kubectl get csidriver secrets-store.csi.k8s.io -o yaml
```

이 명령을 사용하면 설치된 CSI 드라이버에 대한 정보가 반환되어야 합니다.

추가 리소스

Amazon EKS에서 ASCP 사용에 대한 자세한 내용은 다음 리소스를 참조하세요.

- Amazon EKS에서 Pod Identity 사용
- GitHub의AWS Secrets Store CSI 드라이버

Amazon EKS용 포드 자격 증명과 함께 AWS 보안 암호 및 구성 공급자 CSI 사용

Amazon Elastic Kubernetes Service용 Pod Identity Agent와 AWS Secrets and Configuration Provider의 통합은 Amazon EKS에서 실행되는 애플리케이션에 대한 향상된 보안, 간소화된 구성 및 향상된 성능을 제공합니다. Pod Identity는 Secrets Manager에서 보안 암호를 검색하거나 AWS Systems Manager 파라미터 스토어에서 파라미터를 검색할 때 Amazon EKS에 대한 IAM 인증을 간소화합니다.

Amazon EKS Pod Identity는 Amazon EKS 인터페이스를 통해 직접 권한이 설정되도록 허용하여 Kubernetes 애플리케이션의 IAM 권한을 구성하는 프로세스를 간소화하며, 이를 통해 설정 단계 수가 감소하고 Amazon EKS와 IAM 서비스 간에 전환할 필요가 없습니다. Pod Identity를 사용하면 신뢰 정

책을 업데이트하지 않고도 여러 클러스터에서 단일 IAM 역할을 사용할 수 있고, 역할 세션 태그가 지원되어 더 세분화된 액세스 제어가 가능합니다. 이 접근 방식은 역할 간에 권한 정책을 재사용할 수 있도록 하여 정책 관리를 간소화할 뿐만 아니라 일치하는 태그를 기반으로 AWS 리소스에 대한 액세스를 활성화하여 보안을 강화합니다.

작동 방식

1. Pod Identity는 포드에 IAM 역할을 할당합니다.
2. ASCP는이 역할을 사용하여를 인증합니다 AWS 서비스.
3. 권한이 부여된 경우 ASCP는 요청된 보안 암호를 검색하여 포드에서 사용할 수 있도록 합니다.

자세한 내용은 Amazon EKS 사용 설명서의 Amazon EKS Pod Identity 작동 방식 이해를 참조하세요.

사전 조건

> **Important**
>
> Pod Identity는 클라우드의 Amazon EKS에서만 지원됩니다. Amazon EKS Anywhere, Red Hat OpenShift Service on AWS, Amazon EC2 인스턴스의 자체 관리형 Kubernetes 클러스터에서는 지원되지 않습니다.

- Amazon EKS 클러스터(버전 1.24 이상)
- 를 통해 AWS CLI 및 Amazon EKS 클러스터에 액세스 `kubectl`
- 2개에 대한 액세스 AWS 계정 (교차 계정 액세스용)

Amazon EKS Pod Identity Agent 설치

클러스터에서 Pod Identity를 사용하려면 Amazon EKS Pod Identity Agent 추가 기능을 설치해야 합니다.

Pod Identity Agent 설치

- 클러스터에 Pod Identity Agent 추가 기능을 설치합니다.

```
eksctl create addon \
  --name eks-pod-identity-agent \
```

```
  --cluster clusterName \
  --region region
```

Pod Identity를 사용하는 ASCP 설정

1. 포드가 액세스해야 하는 보안 암호에 `secretsmanager:GetSecretValue` 및 `secretsmanager:DescribeSecret` 권한을 부여하는 권한 정책을 생성합니다. 정책 예제는 the section called “예: 개별 보안 암호를 읽고 설명할 수 있는 권한”을 참조하세요.
2. Pod Identity에 대한 Amazon EKS 서비스 보안 주체가 맡을 수 있는 IAM 역할 생성:

   JSON

   ```
   {
       "Version":"2012-10-17",
       "Statement": [
         {
           "Effect": "Allow",
           "Principal": {
             "Service": "pods.eks.amazonaws.com"
           },
           "Action": [
             "sts:AssumeRole",
             "sts:TagSession"
           ]
         }
       ]
     }
   ```

   IAM 정책을 역할에 연결합니다.

   ```
   aws iam attach-role-policy \
     --role-name MY_ROLE \
     --policy-arn POLICY_ARN
   ```

3. Pod Identity 연결을 생성합니다. 예시는 Amazon EKS 사용 설명서의 Pod Identity 연결 생성을 참조하세요.
4. 포드에 탑재할 보안 암호를 지정하는 `SecretProviderClass`를 생성합니다.

```
kubectl apply -f https://raw.githubusercontent.com/aws/secrets-store-csi-driver-
provider-aws/main/examples/ExampleSecretProviderClass-PodIdentity.yaml
```

IRSA와 Pod Identity 사이에서 SecretProviderClass의 주요 차이점은 선택적 파라미터 usePodIdentity입니다. 인증 접근 방식을 결정하는 선택적 필드입니다. 지정하지 않으면 기본적으로 IRSA(서비스 계정에 대한 IAM 역할)를 사용합니다.

- EKS Pod Identity를 사용하려면 다음 값 중 하나를 사용합니다. "true", "True", "TRUE", "t", "T".
- IRSA를 명시적으로 사용하려면 값을 "false", "False", "FALSE", "f", or "F"로 설정합니다.

5. 보안 암호를 /mnt/secrets-store에 탑재하는 포드를 배포합니다.

```
kubectl apply -f https://raw.githubusercontent.com/aws/secrets-store-csi-driver-
provider-aws/main/examples/ExampleDeployment-PodIdentity.yaml
```

6. 프라이빗 Amazon EKS 클러스터를 사용하는 경우 클러스터가 있는 VPC에 AWS STS 엔드포인트가 있는지 확인합니다. 엔드포인트 생성에 대한 자세한 내용은 AWS Identity and Access Management 사용 설명서의 인터페이스 VPC 엔드포인트 섹션을 참조하세요.

보안 암호 탑재 확인

보안 암호가 제대로 탑재되었는지 확인하려면 다음 명령을 실행합니다.

```
kubectl exec -it $(kubectl get pods | awk '/pod-identity-deployment/{print $1}' | head
 -1) -- cat /mnt/secrets-store/MySecret
```

Amazon EKS 포드에 Secrets Manager 보안 암호에 대한 액세스 권한을 설정하는 방법

1. 포드가 액세스해야 하는 보안 암호에 secretsmanager:GetSecretValue 및 secretsmanager:DescribeSecret 권한을 부여하는 권한 정책을 생성합니다. 정책 예제는 the section called “예: 개별 보안 암호를 읽고 설명할 수 있는 권한”을 참조하세요.
2. 보안 암호가 아직 없는 경우 Secrets Manager에서 보안 암호를 생성합니다.

문제 해결

포드 배포를 설명하여 대부분의 오류를 볼 수 있습니다.

컨테이너에 대한 오류 메시지 확인

1. 다음 명령을 사용하여 포드 이름 목록을 가져옵니다. 기본 네임스페이스를 사용하지 않는 경우에는 `-n NAMESPACE`를 사용합니다.

```
kubectl get pods
```

2. 포드를 설명하기 위해 다음 명령에서 *PODID*에 이전 단계에서 찾은 포드의 포드 ID를 사용합니다. 기본 네임스페이스를 사용하지 않는 경우에는 `-n NAMESPACE`를 사용합니다.

```
kubectl describe pod/PODID
```

ASCP에 대한 오류를 확인하려면

- 공급자 로그에서 자세한 정보를 찾으려면 다음 명령에서 *PODID*에 대해 csi-secrets-store-provider-aws 포드의 ID를 사용합니다.

```
kubectl -n kube-system get pods
kubectl -n kube-system logs pod/PODID
```

서비스 계정에 대한 IAM 역할(IRSA)과 함께 AWS 보안 암호 및 구성 공급자 CSI 사용

주제

- 사전 조건
- 액세스 제어 설정
- 탑재할 보안 암호 파악
- 문제 해결

사전 조건

- Amazon EKS 클러스터(버전 1.17 이상)
- 를 통해 AWS CLI 및 Amazon EKS 클러스터에 액세스 `kubectl`

액세스 제어 설정

ASCP는 Amazon EKS Pod Identity를 검색하고 IAM 역할에 대한 자격 증명을 교환합니다. IAM 정책에서 해당 IAM 역할에 대한 권한을 설정합니다. ASCP가 IAM 역할을 가정할 경우, ASCP는 사용자가 권한을 부여한 보안 암호에 대한 액세스 권한을 가져옵니다. 다른 컨테이너는 IAM 역할과 연결하지 않는 한 보안 암호에 액세스할 수 없습니다.

Amazon EKS 포드에 Secrets Manager 보안 암호에 대한 액세스 권한을 부여하는 방법

1. 포드가 액세스해야 하는 보안 암호에 `secretsmanager:GetSecretValue` 및 `secretsmanager:DescribeSecret` 권한을 부여하는 권한 정책을 생성합니다. 정책 예제는 the section called “예: 개별 보안 암호를 읽고 설명할 수 있는 권한”을 참조하세요.
2. 아직 없는 경우 클러스터에 대한 IAM OpenID Connect(OIDC) 공급자를 생성합니다. 자세한 내용은 Amazon EKS 사용 설명서의 Create an IAM OIDC provider for your cluster 섹션을 참조하세요.
3. 서비스 계정용 IAM 역할을 생성하고 정책을 연결합니다. 자세한 내용은 Amazon EKS 사용 설명서의 Create an IAM role for a service account 섹션을 참조하세요.
4. 프라이빗 Amazon EKS 클러스터를 사용하는 경우 클러스터가 있는 VPC에 AWS STS 엔드포인트가 있는지 확인합니다. 엔드포인트 생성에 대한 자세한 내용은 AWS Identity and Access Management 사용 설명서의 인터페이스 VPC 엔드포인트 섹션을 참조하세요.

탑재할 보안 암호 파악

ASCP가 Amazon EKS에 파일 시스템의 파일로 탑재하는 보안 암호를 확인하려면 the section called “SecretProviderClass” YAML 파일을 생성합니다. `SecretProviderClass`는 탑재할 보안 암호와 이를 탑재할 파일 이름을 나열합니다. 이 `SecretProviderClass`는 참조하는 Amazon EKS 포드와 동일한 네임스페이스에 있어야 합니다.

보안 암호를 파일로 탑재

다음 지침에서는 예제 YAML 파일 ExampleSecretProviderClass.yaml 및 ExampleDeployment.yaml을 사용하여 보안 암호를 파일로 탑재하는 방법을 보여줍니다.

Amazon EKS에 보안 암호를 탑재하는 방법

1. 포드에 `SecretProviderClass` 적용:

```
kubectl apply -f ExampleSecretProviderClass.yaml
```

2. 포드 배포:

```
kubectl apply -f ExampleDeployment.yaml
```

3. ASCP가 파일을 탑재합니다.

문제 해결

포드 배포를 설명하여 대부분의 오류를 볼 수 있습니다.

컨테이너에 대한 오류 메시지 확인

1. 다음 명령을 사용하여 포드 이름 목록을 가져옵니다. 기본 네임스페이스를 사용하지 않는 경우에는 `-n` *nameSpace*를 사용합니다.

   ```
   kubectl get pods
   ```

2. 포드를 설명하기 위해 다음 명령에서 *podId*에 이전 단계에서 찾은 포드의 포드 ID를 사용합니다. 기본 네임스페이스를 사용하지 않는 경우에는 `-n` *nameSpace*를 사용합니다.

   ```
   kubectl describe pod/podId
   ```

ASCP에 대한 오류를 확인하려면

- 공급자 로그에서 자세한 정보를 확인하려면, 다음 명령에서 *podId*에 csi-secrets-store-provider-aws 포드의 ID를 사용합니다.

  ```
  kubectl -n kube-system get pods
  kubectl -n kube-system logs Pod/podId
  ```

- **SecretProviderClass** CRD 설치 여부 확인:

  ```
  kubectl get crd secretproviderclasses.secrets-store.csi.x-k8s.io
  ```

  이 명령에서 `SecretProviderClass` 사용자 지정 리소스 정의에 관한 정보가 반환되어야 합니다.

- SecretProviderClass 객체가 생성되었는지 확인합니다.

```
kubectl get secretproviderclass SecretProviderClassName -o yaml
```

AWS 보안 암호 및 구성 공급자 코드 예제

ASCP 인증 및 액세스 제어 예제

예: Amazon EKS Pod Identity 서비스(pods.eks.amazonaws.com)가 역할을 수임하고 세션에 태그를 지정하도록 허용하는 IAM 정책:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "pods.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

SecretProviderClass

YAML을 사용하여 ASCP를 통해 Amazon EKS에 탑재하는 보안 암호를 설명합니다. 예시는 the section called “SecretProviderClass 사용” 섹션을 참조하세요.

SecretProviderClass YAML 구조

```
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
```

```
metadata:
   name: name
spec:
  provider: aws
  parameters:
    region:
    failoverRegion:
    pathTranslation:
    usePodIdentity:
    preferredAddressType:
    objects:
```

파라미터 필드에는 탑재 요청의 세부 정보 포함:

리전

(선택 사항) 보안 암호 AWS 리전 의 입니다. 이 필드를 사용하지 않는 경우 ASCP는 노드의 주석에서 리전을 조회합니다. 이 조회는 탑재 요청에 오버헤드를 추가하므로 많은 수의 포드를 사용하는 클러스터에 리전을 제공하는 것이 좋습니다.

또한 `failoverRegion`을(를) 지정하는 경우 ASCP는 두 리전 모두에서 보안 암호 검색을 시도합니다. 두 리전 중 하나가 예를 들어 인증 문제에 대해 4xx 오류를 반환하는 경우 ASCP는 어느 암호도 탑재하지 않습니다. `region`에서 보안 암호가 성공적으로 검색되면 ASCP는 해당 암호 값을 탑재합니다. `region`에서는 보안 암호가 성공적으로 검색되지 않고 `failoverRegion`에서 성공적으로 검색되는 경우 ASCP는 해당 암호 값을 탑재합니다.

failoverRegion

(선택 사항) 이 필드를 포함하는 경우 ASCP는 `region` 및 이 필드에 정의된 리전에서 보안 암호를 검색하려고 시도합니다. 두 리전 중 하나가 예를 들어 인증 문제에 대해 4xx 오류를 반환하는 경우 ASCP는 어느 암호도 탑재하지 않습니다. `region`에서 보안 암호가 성공적으로 검색되면 ASCP는 해당 암호 값을 탑재합니다. `region`에서는 보안 암호가 성공적으로 검색되지 않고 `failoverRegion`에서 성공적으로 검색되는 경우 ASCP는 해당 암호 값을 탑재합니다. 이 필드 사용 방법의 예는 다중 리전 보안 암호 장애 조치 섹션을 참조하세요.

pathTranslation

(선택 사항) Amazon EKS의 파일 이름에 Linux의 슬래시(/)와 같은 경로 구분 문자가 포함된 경우 사용할 단일 대체 문자입니다. ASCP는 경로 구분 문자가 포함된 탑재 파일을 생성할 수 없습니다. 그 대신 ASCP는 경로 구분 문자를 다른 문자로 대체합니다. 이 필드를 사용하지 않는 경우 대체 문자는 밑줄(_)이므로 예를 들어 `My/Path/Secret`은 `My_Path_Secret`으로 탑재됩니다.

문자 대체를 방지하려면 `False` 문자열을 입력합니다.

usePodIdentity

(선택 사항) 인증 접근 방식을 결정합니다. 지정하지 않으면 기본적으로 IRSA(서비스 계정에 대한 IAM 역할)를 사용합니다.

- EKS Pod Identity를 사용하려면 다음 값 중 하나를 사용합니다. `"true"`", `"True"`, `"TRUE"`, `"t"` 또는 `"T"`.
- IRSA를 명시적으로 사용하려면 다음 값 중 하나를 사용합니다. `"false"`, `"False"`, `"FALSE"`, `"f"` 또는 `"F""=.

preferredAddressType

(선택 사항) Pod Identity Agent 엔드포인트 통신에 사용할 기본 설정 IP 주소 유형을 지정합니다. 필드는 EKS Pod Identity 기능을 사용할 때만 적용되고 서비스 계정에 IAM 역할을 사용할 때는 무시됩니다. 값은 대/소문자를 구분하지 않습니다. 유효값은 다음과 같습니다.

- `"ipv4"`, `"IPv4""` 또는 `"IPV4"` - Pod Identity Agent IPv4 엔드포인트 강제 사용
- `"ipv6"`, `"IPv6"` 또는 `"IPV6"` - Pod Identity Agent IPv6 엔드포인트 강제 사용
- 지정되지 않음 - 자동 엔드포인트 선택 사용, IPv4 엔드포인트를 먼저 시도하고 IPv4 실패 시 IPv6 엔드포인트로 폴백

객체

탑재할 보안 암호의 YAML 선언을 포함하는 문자열입니다. YAML 다중 행 문자열 또는 파이프(|) 문자를 사용하는 것이 좋습니다.

objectName

필수 사항입니다. 가져올 보안 암호의 이름을 지정합니다. Secrets Manager의 경우 파라미터의 `SecretId`로 보안 암호의 친숙한 이름 또는 전체 ARN일 수 있습니다. SSM Parameter Store의 경우, 이는 파라미터의 `Name`이며 파라미터의 이름 또는 전체 ARN일 수 있습니다.

objectType

`objectName`으로 Secrets Manager ARN을 사용하지 않는 경우에 필요합니다. `secretsmanager` 또는 `ssmparameter`입니다.

objectAlias

(선택 사항) Amazon EKS 포드에 있는 보안 암호의 파일 이름입니다. 이 필드를 지정하지 않은 경우 `objectName`이 파일 이름으로 나타납니다.

filePermission

(선택 사항) 보안 암호를 탑재할 파일 권한을 지정하는 4자리 8진수 문자열입니다. 이 필드를 지정하지 않으면 "0644"로 기본 설정됩니다.

objectVersion

(선택 사항) 보안 암호의 버전 ID입니다. 보안 암호를 업데이트할 때마다 버전 ID를 업데이트해야 하므로 권장하지 않습니다. 기본적으로 가장 최신 버전이 사용됩니다. failoverRegion을(를) 포함하는 경우 이 필드는 기본 objectVersion을(를) 나타냅니다.

objectVersionLabel

(선택 사항) 버전의 별칭입니다. 기본 버전은 가장 최근 버전 AWSCURRENT입니다. 자세한 내용은 the section called “보안 암호 버전” 단원을 참조하십시오. failoverRegion을(를) 포함하는 경우 이 필드는 기본 objectVersionLabel을(를) 나타냅니다.

jmesPath

(선택 사항) Amazon EKS에 탑재할 파일에 대한 보안 암호의 키 맵입니다. 이 필드를 사용하려면 보안 암호 값이 JSON 형식이어야 합니다. 이 필드를 사용하는 경우 path 및 objectAlias 하위 필드를 포함해야 합니다.

경로

보안 암호 값의 JSON에 있는 키/값 쌍의 키입니다. 필드에 하이픈이 포함된 경우 작은 따옴표를 사용하여 하이픈을 이스케이프 처리합니다. 예: path: '"hyphenated-path"'

objectAlias

Amazon EKS 포드에 탑재할 파일 이름입니다. 필드에 하이픈이 포함된 경우 작은 따옴표를 사용하여 하이픈을 이스케이프 처리합니다. 예: objectAlias: '"hyphenated-alias"'

filePermission

(선택 사항) 보안 암호를 탑재할 파일 권한을 지정하는 4자리 8진수 문자열입니다. 이 필드를 지정하지 않으면 상위 객체의 파일 권한으로 기본 설정됩니다.

failoverObject

(선택 사항) 이 필드를 지정하는 경우 ASCP는 기본 objectName에 지정된 보안 암호와 failoverObject objectName 하위 필드에 지정된 보안 암호를 모두 검색하려고 시도합니다. 둘 중 하나가 예를 들어 인증 문제에 대해 4xx 오류를 반환하는 경우 ASCP는 어느 암호도 탑

재하지 않습니다. 기본 objectName에서 보안 암호가 성공적으로 검색되면 ASCP는 해당 암호 값을 탑재합니다. 기본 objectName에서는 보안 암호가 성공적으로 검색되지 않고 장애 조치 objectName에서 성공적으로 검색되는 경우 ASCP는 해당 암호 값을 탑재합니다. 이 필드를 포함하는 경우 필드 objectAlias도 포함해야 합니다. 이 필드 사용 방법의 예는 다른 보안 암호로 장애 조치 섹션을 참조하세요.

일반적으로 장애 조치 암호가 복제본이 아닌 경우 이 필드를 사용합니다. 복제본을 지정하는 방법에 대한 예는 다중 리전 보안 암호 장애 조치 섹션을 참조하세요.

objectName

장애 조치 암호의 이름 또는 전체 ARN입니다. ARN을 사용하는 경우 ARN의 리전이 필드 failoverRegion와(과) 일치해야 합니다.

objectVersion

(선택 사항) 보안 암호의 버전 ID입니다. 기본 objectVersion와(과) 일치해야 합니다. 보안 암호를 업데이트할 때마다 버전 ID를 업데이트해야 하므로 권장하지 않습니다. 기본적으로 가장 최신 버전이 사용됩니다.

objectVersionLabel

(선택 사항) 버전의 별칭입니다. 기본 버전은 가장 최근 버전 AWSCURRENT입니다. 자세한 내용은 the section called “보안 암호 버전” 단원을 참조하십시오.

Amazon EKS 포드에 보안 암호를 탑재하기 위한 기본 SecretProviderClass 구성을 생성합니다.

Pod Identity

동일한 Amazon EKS 클러스터에서 보안 암호를 사용하는 SecretProviderClass:

```
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: aws-secrets-manager
spec:
  provider: aws
  parameters:
    objects: |
      - objectName: "mySecret"
        objectType: "secretsmanager"
    usePodIdentity: "true"
```

IRSA

```
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: deployment-aws-secrets
spec:
  provider: aws
  parameters:
    objects: |
        - objectName: "MySecret"
          objectType: "secretsmanager"
```

SecretProviderClass 사용

다음 예제를 사용하여 다양한 시나리오에 대한 SecretProviderClass 구성을 생성합니다.

예: 이름 또는 ARN으로 보안 암호 탑재

이 예제에서는 세 가지 유형의 보안 암호를 탑재하는 방법을 보여줍니다.

- 전체 ARN에 의해 지정된 보안 암호
- 이름으로 지정된 보안 암호
- 보안 암호의 특정 버전

```
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: aws-secrets
spec:
  provider: aws
  parameters:
    objects: |
      - objectName: "arn:aws:secretsmanager:us-east-2:777788889999:secret:MySecret2-
d4e5f6"
      - objectName: "MySecret3"
        objectType: "secretsmanager"
      - objectName: "MySecret4"
        objectType: "secretsmanager"
        objectVersionLabel: "AWSCURRENT"
```

예: 보안 암호에서 키/값 쌍 탑재

이 예제는 JSON 형식 보안 암호에서 특정 키-값 쌍을 탑재하는 방법을 보여줍니다.

```
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: aws-secrets
spec:
  provider: aws
  parameters:
    objects: |
      - objectName: "arn:aws:secretsmanager:us-east-2:777788889999:secret:MySecret-
a1b2c3"
        jmesPath:
            - path: username
              objectAlias: dbusername
            - path: password
              objectAlias: dbpassword
```

예: 파일 권한으로 보안 암호 탑재

이 예제에서는 특정 파일 권한으로 보안 암호를 탑재하는 방법을 보여줍니다.

```
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: aws-secrets
spec:
  provider: aws
  parameters:
    objects: |
      - objectName: "mySecret"
        objectType: "secretsmanager"
        filePermission: "0600"
        jmesPath:
            - path: username
              objectAlias: dbusername
              filePermission: "0400"
```

예: 장애 조치 구성 예제

이 예제에서는 보안 암호에 대한 장애 조치 구성 방법을 보여줍니다.

다중 리전 보안 암호 장애 조치

이 예제에서는 여러 리전에 복제된 보안 암호에 대한 자동 장애 조치 구성 방법을 보여줍니다.

```
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: aws-secrets
spec:
  provider: aws
  parameters:
    region: us-east-1
    failoverRegion: us-east-2
    objects: |
      - objectName: "MySecret"
```

다른 보안 암호로 장애 조치

이 예제에서는 (복제본이 아닌) 다른 보안 암호로의 장애 조치 구성 방법을 보여줍니다.

```
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: aws-secrets
spec:
  provider: aws
  parameters:
    region: us-east-1
    failoverRegion: us-east-2
    objects: |
      - objectName: "arn:aws:secretsmanager:us-east-1:777788889999:secret:MySecret-
a1b2c3"
        objectAlias: "MyMountedSecret"
        failoverObject:
          - objectName: "arn:aws:secretsmanager:us-
east-2:777788889999:secret:MyFailoverSecret-d4e5f6"
```

추가 리소스

Amazon EKS에서 ASCP 사용에 대한 자세한 내용은 다음 리소스를 참조하세요.

- Amazon EKS에서 Pod Identity 사용

- AWS 보안 암호 및 구성 공급자 사용
- GitHub의AWS Secrets Store CSI 드라이버

AWS Lambda 함수에서 AWS Secrets Manager 보안 암호 사용

AWS Lambda 는 서버를 프로비저닝하거나 관리하지 않고도 코드를 실행할 수 있는 서버리스 컴퓨팅 서비스입니다. 의 기능인 Parameter Store는 구성 데이터 관리 및 보안 암호 관리를 위한 안전한 계층적 스토리지를 AWS Systems Manager제공합니다. AWS 파라미터 및 보안 암호 Lambda 확장을 사용하여 SDK를 사용하지 않고도 Lambda 함수에서 AWS Secrets Manager 보안 암호 및 파라미터 스토어 파라미터를 검색하고 캐시할 수 있습니다. 이 확장 기능 사용에 대한 자세한 내용은 Lambda 개발자 가이드의 Lambda 함수에서 Secrets Manager 보안 암호 사용을 참조하세요.

Lambda에서 Secrets Manager 보안 암호 사용

Lambda 개발자 가이드는 Lambda 함수에서 Secrets Manager 보안 암호를 사용하는 방법에 대한 종합적인 지침을 제공합니다. 시작하려면 다음을 수행하세요.

1. Lambda 함수에서 Secrets Manager 보안 암호 사용의 단계별 자습서를 따릅니다. 여기에는 다음 내용이 포함되어 있습니다.
   - 선호하는 런타임(Python, Node.js, Java)으로 Lambda 함수 생성
   - AWS 파라미터 및 보안 Lambda 확장을 계층으로 추가
   - 필요한 권한 구성
   - 확장 기능에서 보안 암호를 가져오는 코드 작성
   - 함수 테스트
2. 확장 기능 동작을 구성하는 환경 변수, 캐시 설정 및 타임아웃 설정에 대해 알아봅니다.
3. 보안 암호 교체 작업 시 모범 사례를 이해합니다.

VPC에서 Secrets Manager 및 Lambda 사용

Lambda 함수가 VPC에서 실행될 경우, 확장이 Secrets Manager를 직접적으로 호출할 수 있도록 VPC 엔드포인트를 생성해야 합니다. 자세한 내용은 the section called “VPC 엔드포인트(AWS PrivateLink)” 단원을 참조하십시오.

AWS 파라미터 및 보안 암호 Lambda 확장 사용

이 확장 기능은 Secrets Manager 보안 암호와 Parameter Store 파라미터를 모두 검색할 수 있습니다. 이 확장 기능에서 Parameter Store 파라미터를 사용하는 방법에 대한 자세한 내용은 AWS Systems Manager 사용 설명서의 Lambda 함수에서 Parameter Store 파라미터 사용을 참조하세요.

Systems Manager 설명서에는 다음이 포함됩니다.

- 이 확장 기능이 Parameter Store와 작동하는 방식에 대한 자세한 설명
- Lambda 함수에 확장 기능 추가 관련 지침
- 확장 기능 구성을 위한 환경 변수
- 파라미터 검색을 위한 샘플 명령
- 지원되는 모든 아키텍처 및 리전에 대한 확장 ARN의 전체 목록

AWS Secrets Manager 에이전트 사용

Secrets Manager Agent 작동 방식

AWS Secrets Manager 에이전트는 컴퓨팅 환경에서 Secrets Manager의 보안 암호를 사용하는 방법을 표준화하는 데 도움이 되는 클라이언트 측 HTTP 서비스입니다. 다음 서비스와 함께 사용할 수 있습니다.

- AWS Lambda
- Amazon Elastic Container Service
- Amazon Elastic Kubernetes Service:
- - Amazon Elastic Compute Cloud

Secrets Manager Agent는 보안 암호를 메모리에 가져와 캐싱하므로, 애플리케이션은 Secrets Manager에 직접 호출하지 않고도 localhost에서 보안 암호를 가져올 수 있습니다. Secrets Manager Agent는 보안 암호를 읽기만 할 수 있으며, 수정은 불가능합니다.

> **Important**
>
> Secrets Manager 에이전트는 환경의 AWS 자격 증명을 사용하여 Secrets Manager를 호출합니다. 또한 서버 측 요청 위조(SSRF)로부터 보호 기능을 제공하여 보안 암호의 안전성을 높입

니다. Secrets Manager Agent는 기본적으로 포스트 양자 ML-KEM 키 교환을 최우선 키 교환 방식으로 사용합니다.

Secrets Manager Agent 캐싱 이해

Secrets Manager Agent는 인 메모리 캐시를 사용하며, Secrets Manager Agent가 재시작되면 캐시가 초기화됩니다. 캐시된 보안 암호 값은 다음 기준으로 주기적으로 갱신됩니다.

- 기본 갱신 주기(TTL)는 300초
- 구성 파일을 통해 TTL을 수정할 수 있음
- TTL이 만료된 후 보안 암호를 요청하면 갱신이 발생

> **Note**
>
> Secrets Manager Agent에는 캐시 무효화 기능이 없습니다. 만약 캐시 항목이 만료되기 전에 보안 암호가 교체되면, Secrets Manager Agent는 오래된 보안 암호 값을 반환할 수도 있습니다.

Secrets Manager Agent는 `GetSecretValue`의 응답과 동일한 형식으로 보안 암호 값을 반환합니다. 보안 암호 값은 캐시에서 암호화되지 않습니다.

주제

- Secrets Manager Agent 빌드
- Secrets Manager Agent 설치
- Secrets Manager Agent를 사용하여 보안 암호 검색
- refreshNow 파라미터 이해
- Secrets Manager Agent 구성
- 선택적 기능
- 로깅
- 보안 고려 사항

Secrets Manager Agent 빌드

시작하기 전에 플랫폼에 맞는 표준 개발 도구 및 Rust 도구가 설치되어 있는지 확인합니다.

> **Note**
>
> macOS에서 `fips` 기능을 활성화하여 에이전트를 빌드하려면 현재 다음 해결 방법이 필요합니다.
>
> - `xcrun --show-sdk-path` 실행의 결과로 설정된 `SDKROOT`라는 환경 변수를 생성합니다.

RPM-based systems

RPM 기반 시스템에서 빌드하는 방법

1. 리포지토리에 제공된 `install` 스크립트를 실행합니다.

   스크립트는 시작 시 무작위 SSRF 토큰을 생성한 후 이를 `/var/run/awssmatoken` 파일에 저장합니다. 설치 스크립트가 생성하는 `awssmatokenreader` 그룹에서 토큰을 읽을 수 있습니다.
2. 애플리케이션이 토큰 파일을 읽을 수 있도록 하려면 애플리케이션을 실행하는 사용자 계정을 `awssmatokenreader` 그룹에 추가해야 합니다. 예를 들어 다음 usermod 명령을 사용하여 토큰 파일을 읽을 수 있는 권한을 애플리케이션에 부여할 수 있습니다. 여기서 *<APP_USER>*는 애플리케이션을 실행하는 사용자 ID입니다.

   ```
   sudo usermod -aG awssmatokenreader <APP_USER>
   ```

   개발 도구 설치

   AL2023과 같은 RPM 기반 시스템에서는 개발 도구 그룹을 설치합니다.

   ```
   sudo yum -y groupinstall "Development Tools"
   ```
3. Rust 설치

   Rust 설명서의 Rust 설치 지침에 따라 설치합니다.

   ```
   curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh # Follow the on-
   screen instructions
   ```

```
. "$HOME/.cargo/env"
```

4. 에이전트 빌드

   cargo build 명령을 사용해 Secrets Manager Agent를 빌드합니다.

   ```
   cargo build --release
   ```

   `target/release/aws_secretsmanager_agent`에서 실행 파일을 찾을 수 있습니다.

Debian-based systems

Debian 기반 시스템에서 빌드하는 방법

1. 개발 도구 설치

   Ubuntu와 같은 Debian 기반 시스템에서는 build-essential 패키지를 설치합니다.

   ```
   sudo apt install build-essential
   ```

2. Rust 설치

   Rust 설명서의 Rust 설치 지침에 따라 설치합니다.

   ```
   curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh # Follow the on-
   screen instructions
   . "$HOME/.cargo/env"
   ```

3. 에이전트 빌드

   cargo build 명령을 사용해 Secrets Manager Agent를 빌드합니다.

   ```
   cargo build --release
   ```

   `target/release/aws_secretsmanager_agent`에서 실행 파일을 찾을 수 있습니다.

Windows

Windows에서 빌드하는 방법

1. 개발 환경 설정

   Microsoft Windows 설명서의 Windows용 Rust 개발 환경 설정 지침에 따라 개발 환경을 설정합니다.

2. 에이전트 빌드

   cargo build 명령을 사용해 Secrets Manager Agent를 빌드합니다.

   ```
   cargo build --release
   ```

   `target/release/aws_secretsmanager_agent.exe`에서 실행 파일을 찾을 수 있습니다.

Cross-compile natively

기본적으로 교차 컴파일하는 방법

1. 교차 컴파일 도구 설치

   Ubuntu와 같이 mingw-w64 패키지를 사용할 수 있는 배포판에서는 다음과 같이 교차 컴파일 도구 체인을 설치합니다.

   ```
   # Install the cross compile tool chain
   sudo add-apt-repository universe
   sudo apt install -y mingw-w64
   ```

2. Rust 빌드 대상 추가

   Windows GNU 빌드 대상을 설치합니다.

   ```
   rustup target add x86_64-pc-windows-gnu
   ```

3. Windows용으로 빌드

   Windows용으로 에이전트를 교차 컴파일합니다.

   ```
   cargo build --release --target x86_64-pc-windows-gnu
   ```

`target/x86_64-pc-windows-gnu/release/aws_secretsmanager_agent.exe`에서 실행 파일을 찾을 수 있습니다.

Cross compile with Rust cross

Rust cross를 사용하여 교차 컴파일하는 방법

시스템에서 교차 컴파일 도구가 기본적으로 제공되지 않는 경우 Rust 크로스 프로젝트를 사용할 수 있습니다. 자세한 내용은 https://github.com/cross-rs/cross를 참조하세요.

> **Important**
>
> 빌드 환경에는 32GB 디스크 공간을 권장합니다.

1. Docker 설정

   Docker를 설치 및 구성합니다.

   ```
   # Install and start docker
   sudo yum -y install docker
   sudo systemctl start docker
   sudo systemctl enable docker # Make docker start after reboot
   ```

2. Docker 권한 구성

   Docker 그룹에 사용자를 추가합니다.

   ```
   # Give ourselves permission to run the docker images without sudo
   sudo usermod -aG docker $USER
   newgrp docker
   ```

3. Windows용으로 빌드

   cross를 설치하고 실행 파일을 빌드합니다.

   ```
   # Install cross and cross compile the executable
   cargo install cross
   cross build --release --target x86_64-pc-windows-gnu
   ```

Secrets Manager Agent 설치

다음 설치 옵션 중에서 사용할 컴퓨팅 환경을 선택합니다.

Amazon EC2

Amazon EC2에 Secrets Manager Agent를 설치하는 방법

1. 구성 디렉터리로 이동

   구성 디렉터리로 변경합니다.

   ```
   cd aws_secretsmanager_agent/configuration
   ```

2. 설치 스크립트 실행

   리포지토리에 제공된 `install` 스크립트를 실행합니다.

   스크립트는 시작 시 무작위 SSRF 토큰을 생성한 후 이를 `/var/run/awssmatoken` 파일에 저장합니다. 설치 스크립트가 생성하는 `awssmatokenreader` 그룹에서 토큰을 읽을 수 있습니다.

3. 애플리케이션 권한 구성

   애플리케이션이 실행되는 사용자 계정을 `awssmatokenreader` 그룹에 추가합니다.

   ```
   sudo usermod -aG awssmatokenreader APP_USER
   ```

   *APP_USER*를 애플리케이션이 실행되는 사용자 ID로 바꿔 입력합니다.

Container Sidecar

Docker를 사용하여 Secrets Manager Agent를 애플리케이션과 함께 사이드카 컨테이너로 실행할 수 있습니다. 이렇게 하면 애플리케이션은 Secrets Manager Agent가 제공하는 로컬 HTTP 서버에서 보안 암호를 검색할 수 있습니다. Docker에 대한 자세한 내용은 Docker 설명서를 참조하세요.

Secrets Manager Agent의 사이드카 컨테이너를 생성하는 방법

1. 에이전트 Dockerfile 생성

   Secrets Manager Agent 사이드카 컨테이너용 Dockerfile을 생성합니다.

```
# Use the latest Debian image as the base
FROM debian:latest

# Set the working directory inside the container
WORKDIR /app

# Copy the Secrets Manager Agent binary to the container
COPY secrets-manager-agent .

# Install any necessary dependencies
RUN apt-get update && apt-get install -y ca-certificates

# Set the entry point to run the Secrets Manager Agent binary
ENTRYPOINT ["./secrets-manager-agent"]
```

2. 애플리케이션 Dockerfile 생성

   클라이언트 애플리케이션용 Dockerfile을 생성합니다.

3. Docker Compose 파일 생성

   두 컨테이너를 공유 네트워크 인터페이스로 실행하기 위한 Docker Compose 파일을 생성합니다.

> **Important**
>
> 애플리케이션이 Secrets Manager 에이전트를 사용할 수 있으려면 AWS 자격 증명과 SSRF 토큰을 로드해야 합니다. Amazon EKS 및 Amazon ECS의 경우 다음 문서를 참조하세요.
>
> - Amazon EKS 사용 설명서의 액세스 관리
> - Amazon ECS 개발자 가이드의 Amazon ECS 작업 IAM 역할

```
version: '3'
services:
    client-application:
    container_name: client-application
    build:
        context: .
        dockerfile: Dockerfile.client
```

```
    command: tail -f /dev/null  # Keep the container running


    secrets-manager-agent:
    container_name: secrets-manager-agent
    build:
        context: .
        dockerfile: Dockerfile.agent
    network_mode: "container:client-application"  # Attach to the client-
application container's network
    depends_on:
        - client-application
```

4. 에이전트 바이너리 복사

   Dockerfiles 및 Docker Compose 파일이 포함된 동일한 디렉터리에 `secrets-manager-agent` 바이너리를 복사합니다.

5. 컨테이너 빌드 및 실행

   Docker Compose를 사용하여 컨테이너를 빌드하고 실행합니다.

   ```
   docker-compose up --build
   ```

6. 다음 단계

   이제 Secrets Manager Agent를 사용하여 클라이언트 컨테이너에서 보안 암호를 가져올 수 있습니다. 자세한 내용은 the section called “Secrets Manager Agent를 사용하여 보안 암호 검색” 단원을 참조하십시오.

Lambda

Secrets Manager Agent를 Lambda 확장으로 패키징할 수 있습니다. 그런 다음, Lambda 함수에 계층으로 추가한 후 Lambda 함수에서 Secrets Manager Agent를 직접적으로 호출하여 보안 암호를 가져올 수 있습니다.

다음 지침에서는 https://github.com/aws/aws-secretsmanager-agent의 예제 스크립트 `secrets-manager-agent-extension.sh`를 사용하여 Secrets Manager Agent를 Lambda 확장으로 설치함으로써 MyTest라는 보안 암호를 가져오는 방법을 보여줍니다.

Secrets Manager Agent용 Lambda 확장을 생성하는 방법

1. 에이전트 계층 패키징

   Secrets Manager Agent 코드 패키지의 루트에서 다음 명령을 실행합니다.

```
AWS_ACCOUNT_ID=AWS_ACCOUNT_ID
LAMBDA_ARN=LAMBDA_ARN

# Build the release binary
cargo build --release --target=x86_64-unknown-linux-gnu

# Copy the release binary into the `bin` folder
mkdir -p ./bin
cp ./target/x86_64-unknown-linux-gnu/release/aws_secretsmanager_agent ./bin/
secrets-manager-agent

# Copy the `secrets-manager-agent-extension.sh` example script into the
 `extensions` folder.
mkdir -p ./extensions
cp aws_secretsmanager_agent/examples/example-lambda-extension/secrets-manager-
agent-extension.sh ./extensions

# Zip the extension shell script and the binary
zip secrets-manager-agent-extension.zip bin/* extensions/*

# Publish the layer version
LAYER_VERSION_ARN=$(aws lambda publish-layer-version \
    --layer-name secrets-manager-agent-extension \
    --zip-file "fileb://secrets-manager-agent-extension.zip" | jq -r
 '.LayerVersionArn')
```

2. SSRF 토큰 구성

   에이전트의 기본 구성에서는 SSRF 토큰을 미리 설정된 AWS_SESSION_TOKEN 또는 AWS_CONTAINER_AUTHORIZATION_TOKEN 환경 변수에 설정된 값(후자는 SnapStart가 활성화된 Lambda 함수에 사용됨)으로 자동 설정합니다. 또는 Lambda 함수에서 임의 값으로 AWS_TOKEN 환경 변수를 정의할 수 있으며, 이 경우 위 두 변수보다 우선 적용됩니다. AWS_TOKEN 환경 변수를 사용하기로 선택하면, 반드시 lambda:UpdateFunctionConfiguration 호출을 통해 해당 환경 변수를 설정해야 합니다.

3. 계층을 함수에 연결

계층을 Lambda 함수에 연결합니다.

```
# Attach the layer version to the Lambda function
aws lambda update-function-configuration \
    --function-name $LAMBDA_ARN \
    --layers "$LAYER_VERSION_ARN"
```

4. 함수 코드 업데이트

   Lambda 함수가 `http://localhost:2773/secretsmanager/get?secretId=MyTest`를 호출하도록 코드를 업데이트하고, 요청 헤더 `X-Aws-codes-Secrets-Token`에는 위에서 설정한 SSRF 토큰 값을 넣어 보안 암호를 가져옵니다. 애플리케이션 코드에 재시도 로직을 구현하여 Lambda 확장의 초기화 및 등록 지연을 수용할 수 있도록 해야 합니다.

5. 함수 테스트

   Lambda 함수를 간접적으로 호출하여 보안 암호를 올바르게 가져오는지 확인합니다.

Secrets Manager Agent를 사용하여 보안 암호 검색

보안 암호를 가져오려면, 보안 암호 이름 또는 ARN을 쿼리 파라미터로 사용하여 로컬 Secrets Manager Agnet 엔드포인트를 호출합니다. 기본적으로 Secrets Manager Agent는 보안 암호의 `AWSCURRENT` 버전을 가져옵니다. 다른 버전을 가져오려면 versionStage 또는 versionId 파라미터를 사용합니다.

> **Important**
>
> Secrets Manager Agent를 보호하려면 각 요청의 일부로 SSRF 토큰 헤더 `X-Aws-Parameters-Secrets-Token`을 포함해야 합니다. Secrets Manager Agent는 이 헤더가 없거나, 잘못된 SSRF 토큰이 있는 요청을 거부합니다. the section called “구성 옵션”에 있는 SSRF 헤더 이름을 사용자 지정할 수 있습니다.

필수 권한

Secrets Manager 에이전트는 자격 AWS 증명 공급자 체인을 사용하는 AWS SDK for Rust를 사용합니다. 이러한 IAM 자격 증명의 ID는 Secrets Manager Agent가 보안 암호를 검색하는 데 필요한 권한을 결정합니다.

- `secretsmanager:DescribeSecret`
- `secretsmanager:GetSecretValue`

권한에 대한 자세한 내용은 the section called “권한 참조” 섹션을 참조하세요.

> **Important**
>
> 보안 암호 값을 Secrets Manager Agent로 가져오면, 컴퓨팅 환경 및 SSRF 토큰에 액세스할 수 있는 모든 사용자가 Secrets Manager Agent 캐시에서 보안 암호에 액세스할 수 있습니다. 자세한 내용은 the section called “보안 고려 사항” 단원을 참조하십시오.

요청 예시

curl

Example예 - curl을 사용하여 보안 암호 가져오기

다음 curl 예제에서는 Secrets Manager Agent에서 보안 암호 값을 가져오는 방법을 보여줍니다. 이 예제는 파일에 있는 SSRF를 기반으로 하며, SSRF는 설치 스크립트에 저장됩니다.

```
curl -v -H \\
    "X-Aws-Parameters-Secrets-Token: $(/var/run/awssmatoken)" \\
    'http://localhost:2773/secretsmanager/get?secretId=YOUR_SECRET_ID' \\
    echo
```

Python

Example예 - Python을 사용하여 보안 암호 가져오기

다음 Python 예제에서는 Secrets Manager Agent에서 보안 암호 값을 가져오는 방법을 보여줍니다. 이 예제는 파일에 있는 SSRF를 기반으로 하며, SSRF는 설치 스크립트에 저장됩니다.

```
import requests
import json

# Function that fetches the secret from Secrets Manager Agent for the provided
 secret id.
def get_secret():
    # Construct the URL for the GET request
    url = f"http://localhost:2773/secretsmanager/get?secretId=YOUR_SECRET_ID"
```

```
# Get the SSRF token from the token file
with open('/var/run/awssmatoken') as fp:
    token = fp.read()

headers = {
    "X-Aws-Parameters-Secrets-Token": token.strip()
}

try:
    # Send the GET request with headers
    response = requests.get(url, headers=headers)

    # Check if the request was successful
    if response.status_code == 200:
        # Return the secret value
        return response.text
    else:
        # Handle error cases
        raise Exception(f"Status code {response.status_code} - {response.text}")

except Exception as e:
    # Handle network errors
    raise Exception(f"Error: {e}")
```

refreshNow 파라미터 이해

Secrets Manager Agent는 보안 암호 값을 인 메모리 캐시에 저장하고 주기적으로 갱신합니다. 기본적으로 이 갱신은 TTL(Time to Live)이 만료된 후 보안 암호를 요청할 때 발생하며, TTL은 보통 300초입니다. 하지만 이 방식은 보안 암호가 캐시 만료 전에 교체되면 오래된 보안 암호 값을 반환할 수 있다는 한계가 있습니다.

이 한계를 해결하기 위해 Secrets Manager Agent는 URL에 `refreshNow`라는 파라미터를 지원합니다. 이 파라미터를 사용하면 캐시를 무시하고 즉시 보안 암호 값을 갱신하여 항상 최신 정보를 가져올 수 있습니다.

기본 동작(**refreshNow** 미사용)

- TTL이 만료될 때까지 캐시된 값을 사용
- TTL(기본 300초) 이후에만 보안 암호 갱신
- TTL 만료 전에 보안 암호가 교체되면 오래된 값을 반환할 수 있음

`refreshNow=true` 사용 시 동작

- 캐시를 완전히 우회
- Secrets Manager에서 최신 보안 암호 값을 직접 가져옴
- 캐시를 최신 값으로 업데이트하고 TTL 재설정
- 항상 가장 최신 보안 암호 값을 확보

보안 암호 값 강제 갱신

> **Important**
>
> `refreshNow`의 기본값은 `false`입니다. `true`로 설정하면 Secrets Manager Agent 구성 파일에 지정된 TTL을 무시하고 Secrets Manager에 API 호출을 수행합니다.

curl

Example예 - curl을 사용하여 보안 암호 강제 갱신

다음 curl 예제에서는 Secrets Manager Agent에서 보안 암호 값을 강제로 갱신하는 방법을 보여줍니다. 이 예제는 파일에 있는 SSRF를 기반으로 하며, SSRF는 설치 스크립트에 저장됩니다.

```
curl -v -H \\
"X-Aws-Parameters-Secrets-Token: $(/var/run/awssmatoken)" \\
'http://localhost:2773/secretsmanager/get?secretId=YOUR_SECRET_ID&refreshNow=true' \
\
echo
```

Python

Example예 - Python을 사용하여 보안 암호 강제 갱신

다음 Python 예제에서는 Secrets Manager Agent에서 보안 암호 값을 가져오는 방법을 보여줍니다. 이 예제는 파일에 있는 SSRF를 기반으로 하며, SSRF는 설치 스크립트에 저장됩니다.

```
import requests
import json
```

```
# Function that fetches the secret from Secrets Manager Agent for the provided
 secret id.
def get_secret():
    # Construct the URL for the GET request
    url = f"http://localhost:2773/secretsmanager/get?
secretId=YOUR_SECRET_ID&refreshNow=true"

    # Get the SSRF token from the token file
    with open('/var/run/awssmatoken') as fp:
        token = fp.read()

    headers = {
        "X-Aws-Parameters-Secrets-Token": token.strip()
    }

    try:
        # Send the GET request with headers
        response = requests.get(url, headers=headers)

        # Check if the request was successful
        if response.status_code == 200:
            # Return the secret value
            return response.text
        else:
            # Handle error cases
            raise Exception(f"Status code {response.status_code} - {response.text}")

    except Exception as e:
        # Handle network errors
        raise Exception(f"Error: {e}")
```

Secrets Manager Agent 구성

Secrets Manager Agent의 구성을 변경하려면 TOML 구성 파일을 생성한 다음 `./aws_secretsmanager_agent --config config.toml`을 직접적으로 호출합니다.

구성 옵션

`log_level`

Secrets Manager Agent: DEBUG, INFO, WARN, ERROR 또는 NONE에 대해 로그에 보고되는 세부 정보의 수준입니다. 기본값은 INFO입니다.

`log_to_file`

로그를 파일 또는 stdout/stderr로 출력할지 여부(`true` 또는 `false`)를 설정합니다. 기본값은 `true`입니다.

`http_port`

로컬 HTTP 서버의 포트로, 포트 범위는 1024~65535입니다. 기본값은 2773입니다.

`region`

요청에 사용할 AWS 리전입니다. 리전이 지정되지 않은 경우 Secrets Manager Agent가 SDK에서 리전을 결정합니다. 자세한 내용은 AWS SDK for Rust 개발자 가이드의 Specify your credentials and default Region 섹션을 참조하세요.

`ttl_seconds`

캐시된 항목의 TTL(초 단위)로, 범위는 0~3600입니다. 기본값은 300입니다. 0은 캐싱이 없음을 나타냅니다.

`cache_size`

캐시에 저장할 수 있는 보안 암호의 최대 개수로, 범위는 1~1000입니다. 기본값은 1000입니다.

`ssrf_headers`

Secrets Manager Agent가 SSRF 토큰이 있는지 확인하는 헤더 이름 목록입니다. 기본값은 'X-Aws-Parameters-Secrets-Token, X-Vault-Token'입니다.

`ssrf_env_variables`

Secrets Manager Agent가 순차적으로 확인할 SSRF 토큰 환경 변수 이름 목록입니다. 환경 변수에는 토큰 또는 `AWS_TOKEN=file:///var/run/awssmatoken`에서와 같이 토큰 파일에 대한 참조가 포함될 수 있습니다. 기본값은 "AWS_TOKEN, AWS_SESSION_TOKEN, AWS_CONTAINER_AUTHORIZATION_TOKEN"입니다.

`path_prefix`

요청이 경로 기반 요청인지 여부를 결정하는 데 사용되는 URI 접두사입니다. 기본값은 '/v1/'입니다.

`max_conn`

Secrets Manager Agent가 허용하는 HTTP 클라이언트의 최대 연결 수로, 범위는 1~1000입니다. 기본값은 800입니다.

선택적 기능

Secrets Manager Agent는 `cargo build`에 `--features` 플래그를 전달하여 선택적 기능을 포함해 빌드할 수 있습니다. 사용 가능한 기능은 다음과 같습니다.

빌드 기능

`prefer-post-quantum`

`X25519MLKEM768` 알고리즘을 가장 우선 순위의 키 교환 알고리즘으로 설정합니다. 기본적으로는 사용 가능하지만 최우선으로 설정되지는 않습니다. `X25519MLKEM768`은 하이브리드, 포스트 양자 보안 키 교환 알고리즘입니다.

`fips`

에이전트에서 사용하는 암호 제품군을 FIPS 승인 암호화만 사용하도록 제한합니다.

로깅

로컬 로깅

Secrets Manager Agent는 오류를 로컬 파일(`logs/secrets_manager_agent.log`)이나 `log_to_file` 구성 변수에 따라 stdout/stderr로 기록합니다. 애플리케이션이 Secrets Manager Agent를 호출하여 보안 암호를 가져오면 해당 호출이 로컬 로그에 표시됩니다. CloudTrail 로그에는 표시되지 않습니다.

로그 교체

Secrets Manager Agent는 파일이 10MB에 도달하면 새 로그 파일을 생성하고 총 5개의 로그 파일을 저장합니다.

AWS 서비스 로깅

로그는 Secrets Manager, CloudTrail 또는 CloudWatch로 이동하지 않습니다. Secrets Manager Agent에서 보안 암호를 가져오라는 요청은 이러한 로그에 표시되지 않습니다. Secrets Manager Agent가 암호를 가져오기 위해 Secrets Manager를 직접적으로 호출하면 해당 호출은 `aws-secrets-manager-agent`가 포함된 사용자 에이전트 문자열과 함께 CloudTrail에 기록됩니다.

the section called “구성 옵션”에서 로깅 옵션을 구성할 수 있습니다.

보안 고려 사항

신뢰 도메인

에이전트 아키텍처의 경우, 신뢰 도메인은 에이전트 엔드포인트 및 SSRF 토큰이 액세스할 수 있는 곳으로, 대개 전체 호스트입니다. 동일한 보안 태세를 유지하려면 Secrets Manager Agent의 신뢰 도메인은 Secrets Manager 자격 증명이 제공되는 도메인과 일치해야 합니다. 예를 들어 Amazon EC2에서 Secrets Manager Agent의 신뢰 도메인은 Amazon EC2에 대한 역할을 사용할 경우 자격 증명의 도메인과 동일합니다.

Important

Secrets Manager 자격 증명이 애플리케이션에 잠겨 있는 에이전트 솔루션을 아직 사용하지 않는 보안 인식 애플리케이션은 언어별 AWS SDKs 또는 캐싱 솔루션 사용을 고려해야 합니다. 자세한 내용은 보안 암호 가져오기를 참조하세요.

C++ AWS SDK를 사용하여 Secrets Manager 보안 암호 값 가져오기

C++ 애플리케이션의 경우 GetSecretValue 또는 BatchGetSecretValue를 사용하여 SDK를 직접 호출합니다.

다음 코드 예제에서는 Secrets Manager 보안 암호 값을 가져오는 방법을 보여줍니다.

필요한 권한:`secretsmanager:GetSecretValue`

```
//! Retrieve an AWS Secrets Manager encrypted secret.
/*!
  \param secretID: The ID for the secret.
  \return bool: Function succeeded.
 */
bool AwsDoc::SecretsManager::getSecretValue(const Aws::String &secretID,
                                            const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SecretsManager::SecretsManagerClient
 secretsManagerClient(clientConfiguration);

    Aws::SecretsManager::Model::GetSecretValueRequest request;
```

```
    request.SetSecretId(secretID);

    Aws::SecretsManager::Model::GetSecretValueOutcome getSecretValueOutcome =
 secretsManagerClient.GetSecretValue(
            request);
    if (getSecretValueOutcome.IsSuccess()) {
        std::cout << "Secret is: "
                  << getSecretValueOutcome.GetResult().GetSecretString() << std::endl;
    }
    else {
        std::cerr << "Failed with Error: " << getSecretValueOutcome.GetError()
                  << std::endl;
    }

    return getSecretValueOutcome.IsSuccess();
}
```

JavaScript AWS SDK를 사용하여 Secrets Manager 보안 암호 값 가져오기

JavaScript 애플리케이션의 경우 `getSecretValue` 또는 `batchGetSecretValue`를 사용하여 SDK를 직접 호출합니다.

다음 코드 예제에서는 Secrets Manager 보안 암호 값을 가져오는 방법을 보여줍니다.

필요한 권한:`secretsmanager:GetSecretValue`

```
import {
  GetSecretValueCommand,
  SecretsManagerClient,
} from "@aws-sdk/client-secrets-manager";

export const getSecretValue = async (secretName = "SECRET_NAME") => {
  const client = new SecretsManagerClient();
  const response = await client.send(
    new GetSecretValueCommand({
      SecretId: secretName,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
```

```
  //     httpStatusCode: 200,
  //     requestId: '584eb612-f8b0-48c9-855e-6d246461b604',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   ARN: 'arn:aws:secretsmanager:us-east-1:xxxxxxxxxxxx:secret:binary-
secret-3873048-xxxxxx',
  //   CreatedDate: 2023-08-08T19:29:51.294Z,
  //   Name: 'binary-secret-3873048',
  //   SecretBinary: Uint8Array(11) [
  //      98, 105, 110, 97, 114,
  //     121,  32, 100, 97, 116,
  //      97
  //   ],
  //   VersionId: '712083f4-0d26-415e-8044-16735142cd6a',
  //   VersionStages: [ 'AWSCURRENT' ]
  // }

  if (response.SecretString) {
    return response.SecretString;
  }

  if (response.SecretBinary) {
    return response.SecretBinary;
  }
};
```

Kotlin AWS SDK를 사용하여 Secrets Manager 보안 암호 값 가져오기

Kotlin 애플리케이션의 경우 GetSecretValue 또는 BatchGetSecretValue를 사용하여 SDK를 직접 호출합니다.

다음 코드 예제에서는 Secrets Manager 보안 암호 값을 가져오는 방법을 보여줍니다.

필요한 권한:`secretsmanager:GetSecretValue`

```
suspend fun getValue(secretName: String?) {
    val valueRequest =
        GetSecretValueRequest {
```

```
            secretId = secretName
        }

    SecretsManagerClient.fromEnvironment { region = "us-east-1" }.use { secretsClient -
>
        val response = secretsClient.getSecretValue(valueRequest)
        val secret = response.secretString
        println("The secret value is $secret")
    }
}
```

PHP AWS SDK를 사용하여 Secrets Manager 보안 암호 값 가져오기

PHP 애플리케이션의 경우 GetSecretValue 또는 BatchGetSecretValue를 사용하여 SDK를 직접 호출합니다.

다음 코드 예제에서는 Secrets Manager 보안 암호 값을 가져오는 방법을 보여줍니다.

필요한 권한:secretsmanager:GetSecretValue

```
<?php

  /**
    * Use this code snippet in your app.
    *
    * If you need more information about configurations or implementing the sample
 code, visit the AWS docs:
    * https://aws.amazon.com/developer/language/php/
    */

  require 'vendor/autoload.php';

  use Aws\SecretsManager\SecretsManagerClient;
  use Aws\Exception\AwsException;

  /**
    * This code expects that you have AWS credentials set up per:
    * https://<<{{DocsDomain}}>>/sdk-for-php/v3/developer-guide/guide_credentials.html
    */

  // Create a Secrets Manager Client
```

```
  $client = new SecretsManagerClient([
      'profile' => 'default',
      'version' => '2017-10-17',
      'region' => '<<{{MyRegionName}}>>',
  ]);

  $secret_name = '<<{{MySecretName}}>>';

  try {
      $result = $client->getSecretValue([
          'SecretId' => $secret_name,
      ]);
  } catch (AwsException $e) {
      // For a list of exceptions thrown, see
      // https://<<{{DocsDomain}}>>/secretsmanager/latest/apireference/
API_GetSecretValue.html
      throw $e;
  }

  // Decrypts secret using the associated KMS key.
  $secret = $result['SecretString'];

  // Your code goes here
```

Ruby AWS SDK를 사용하여 Secrets Manager 보안 암호 값 가져오기

Ruby 애플리케이션의 경우get_secret_value 또는 batch_get_secret_value를 사용하여 SDK를 직접 호출합니다.

다음 코드 예제에서는 Secrets Manager 보안 암호 값을 가져오는 방법을 보여줍니다.

필요한 권한:secretsmanager:GetSecretValue

```
  # Use this code snippet in your app.
  # If you need more information about configurations or implementing the sample code,
 visit the AWS docs:
  # https://aws.amazon.com/developer/language/ruby/

  require 'aws-sdk-secretsmanager'

  def get_secret
```

```
    client = Aws::SecretsManager::Client.new(region: '<<{{MyRegionName}}>>')

    begin
      get_secret_value_response = client.get_secret_value(secret_id:
 '<<{{MySecretName}}>>')
    rescue StandardError => e
      # For a list of exceptions thrown, see
      # https://<<{{DocsDomain}}>>/secretsmanager/latest/apireference/
API_GetSecretValue.html
      raise e
    end

    secret = get_secret_value_response.secret_string
    # Your code goes here.
  end
```

를 사용하여 보안 암호 값 가져오기 AWS CLI

필요한 권한:`secretsmanager:GetSecretValue`

Example보안 암호의 암호화된 암호 값 검색

다음 `get-secret-value` 예에서는 현재 보안 암호 값을 가져옵니다.

```
aws secretsmanager get-secret-value \
    --secret-id MyTestSecret
```

Example이전 보안 암호 값 검색

다음 `get-secret-value` 예에서는 이전 보안 암호 값을 가져옵니다.

```
aws secretsmanager get-secret-value \
        --secret-id MyTestSecret
        --version-stage AWSPREVIOUS
```

를 사용하여 일괄적으로 보안 암호 그룹 가져오기 AWS CLI

필요한 권한:

- `secretsmanager:BatchGetSecretValue`

- 검색할 각 보안 암호에 대한 `secretsmanager:GetSecretValue` 권한이 있어야 합니다.
- 필터를 사용하는 경우 `secretsmanager:ListSecrets`도 있어야 합니다.

권한 정책 예시는 the section called “예: 보안 암호 값 그룹을 일괄적으로 검색할 수 있는 권한” 섹션을 참조하세요.

> **Important**
>
> 검색 중인 그룹에서 개별 보안 암호를 검색할 수 있는 권한을 거부하는 VPCE 정책이 있는 경우 `BatchGetSecretValue`는 보안 암호 값을 반환하지 않고 오류를 반환합니다.

Example이름별로 나열된 보안 암호 그룹의 보안 암호 값을 검색합니다.

다음 `batch-get-secret-value` 예시에서는 현재 보안 암호 값을 가져옵니다.

```
aws secretsmanager batch-get-secret-value \
          --secret-id-list MySecret1 MySecret2 MySecret3
```

Example필터로 선택한 보안 암호 그룹의 보안 암호 값을 검색합니다.

다음 `batch-get-secret-value` 예시에서는 태그가 "Test"인 보안 암호의 보안 암호 값을 가져옵니다.

```
aws secretsmanager batch-get-secret-value \
          --filters Key="tag-key",Values="Test"
```

AWS 콘솔을 사용하여 보안 암호 값 가져오기

보안 암호를 검색하려면(콘솔)

1. https://console.aws.amazon.com/secretsmanager/에서 Secrets Manager 콘솔을 엽니다.
2. 보안 암호 목록에서 검색할 보안 암호를 선택합니다.
3. 보안 암호 값(Secret value) 섹션에서 보안 암호 값 검색(Retrieve secret value)을 선택합니다.

   Secrets Manager는 보안 암호의 현재 버전(`AWSCURRENT`) 을 표시합니다. 보안 암호의 다른 버전(예`AWSPREVIOUS`: 사용자 지정 레이블이 지정된 버전)을 보려면 the section called “AWS CLI”을(를) 사용하세요.

에서 AWS Secrets Manager 보안 암호 사용 AWS Batch

AWS Batch 를 사용하면에서 배치 컴퓨팅 워크로드를 실행할 수 있습니다 AWS 클라우드. 를 사용하면 민감한 데이터를 AWS Secrets Manager 보안 암호에 저장한 다음 작업 정의에서 참조하여 작업에 민감한 데이터를 주입할 AWS Batch수 있습니다. 자세한 내용은 Secrets Manager를 사용해 민감한 데이터 지정을 참조하세요.

CloudFormation 리소스에서 AWS Secrets Manager 보안 암호 가져오기

를 사용하면 다른 CloudFormation 리소스에서 사용할 보안 암호를 검색할 CloudFormation수 있습니다. 일반적인 시나리오는 먼저 Secrets Manager에서 생성한 암호로 보안 암호를 생성한 새 데이터베이스의 자격 증명으로 사용할 보안 암호에서 사용자 이름과 암호를 검색합니다. 를 사용하여 보안 암호를 생성하는 방법에 대한 자세한 내용은 섹션을 CloudFormation참조하세요CloudFormation.

CloudFormation 템플릿에서 보안 암호를 검색하려면 동적 참조를 사용합니다. 스택을 생성할 때 동적 참조는 보안 암호 값을 CloudFormation 리소스로 가져오므로 보안 암호 정보를 하드코딩할 필요가 없습니다. 대신, 이름이나 ARN을 사용하여 보안 암호를 참조합니다. 모든 리소스 속성의 보안 암호에 동적 참조를 사용할 수 있습니다. `AWS::CloudFormation::Init`와(과) 같은 리소스 메타데이터의 보안 암호에 동적 참조를 사용할 수 없습니다. 이렇게 하면 콘솔에 보안 암호 값이 표시되기 때문입니다.

보안 암호에 대한 동적 참조 패턴은 다음과 같습니다.

```
{{resolve:secretsmanager:secret-id:SecretString:json-key:version-stage:version-id}}
```

secret-id

보안 암호의 이름 또는 ARN입니다. AWS 계정의 보안 암호에 액세스하려면 보안 암호 이름을 사용할 수 있습니다. 다른 AWS 계정의 보안 암호에 액세스하려면 보안 암호의 ARN을 사용합니다.

json-key(선택)

검색하고자 하는 값을 보유한 키-값 페어의 키 이름입니다. 를 지정하지 않으면가 전체 보안 암호 텍스트를 `json-key` CloudFormation 검색합니다. 이 세그먼트에는 콜론 문자(`:`)가 포함되지 않을 수 있습니다.

version-stage(선택)

사용하려는 보안 암호의 버전입니다. Secrets Manager는 교체 프로세스 도중 다른 버전을 추적하는 데 스테이징 레이블을 사용합니다. `version-stage`을 사용하는 경우 `version-id`를 지정하

지 마세요. version-stage 또는 version-id를 지정하지 않은 경우 기본값은 AWSCURRENT 버전입니다. 이 세그먼트에는 콜론 문자(:)가 포함되지 않을 수 있습니다.

version-id(선택)

사용하고자 하는 보안 암호의 버전에 대한 고유 식별자입니다. version-id을 지정할 경우 version-stage을 지정하지 마세요. version-stage 또는 version-id를 지정하지 않은 경우 기본값은 AWSCURRENT 버전입니다. 이 세그먼트에는 콜론 문자(:)가 포함되지 않을 수 있습니다.

자세한 내용은 동적 참조를 사용하여 Secrets Manager 보안 암호 지정 섹션을 참조하세요.

> **Note**
>
> 백슬래시를 최종 값으로 사용하여 동적 참조(\)를 생성하지 마십시오.는 이러한 참조를 해결할 CloudFormation 수 없으므로 리소스 오류가 발생합니다.

GitHub 작업에서 AWS Secrets Manager 보안 암호 사용

GitHub 작업에서 보안 암호를 사용하려면 GitHub 작업을 사용하여에서 보안 암호를 검색 AWS Secrets Manager 하고 GitHub 워크플로에서 마스킹된 환경 변수로 추가할 수 있습니다. GitHub 작업에 대한 자세한 내용은 GitHub 문서의 GitHub 작업 이해를 참조하세요.

GitHub 환경에 보안 암호를 추가하면 GitHub 작업의 다른 모든 단계에서 보안 암호를 사용할 수 있습니다. GitHub 작업에 대한 보안 강화의 지침에 따라 환경 내에서 보안 암호가 잘못 사용되는 것을 방지하세요.

보안 암호 값의 전체 문자열을 환경 변수 값으로 설정하거나 문자열이 JSON인 경우 JSON을 구문 분석하여 각 JSON 키-값 쌍에 대한 개별 환경 변수를 설정할 수 있습니다. 보안 암호 값이 이진수인 경우 이 작업을 수행하면 보안 암호 값이 문자열로 변환됩니다.

보안 암호에서 생성된 환경 변수를 보려면 디버그 로깅을 켜세요. 자세한 내용은 GitHub 문서의 디버그 로깅 활성화를 참조하세요.

보안 암호에서 생성된 환경 변수를 사용하려면 GitHub 문서에서 환경 변수를 참조하세요.

사전 조건

이 작업을 사용하려면 먼저 configure-aws-credentials 단계를 사용하여 자격 AWS 증명을 구성하고 AWS 리전 GitHub 환경에서를 설정해야 합니다. GitHub OIDC 공급자를 사용하여 역할을 직접

수임하려면 GitHub 작업을 위한 AWS 보안 인증 작업 구성의 지침을 따르세요. 그러면 수명이 짧은 보안 인증을 사용할 수 있으므로 Secrets Manager 외부에 추가 액세스 키를 저장하지 않아도 됩니다.

작업에서 맡는 IAM 역할은 다음과 같은 권한이 있어야 합니다.

- 검색하려는 보안 암호에 대한 `GetSecretValue`
- 모든 보안 암호에 대한 `ListSecrets`
- (선택 사항) 보안 암호가 로 암호화된 KMS key 경우 `Decrypt`의 고객 관리형 키.

자세한 내용은 the section called “인증 및 액세스 제어” 단원을 참조하십시오.

사용법

작업을 사용하려면 워크플로에 다음 구문을 사용하는 단계를 추가하세요.

```
- name: Step name
  uses: aws-actions/aws-secretsmanager-get-secrets@v2
  with:
    secret-ids: |
      secretId1
      ENV_VAR_NAME, secretId2
    name-transformation: (Optional) uppercase|lowercase|none
    parse-json-secrets: (Optional) true|false
```

Parameters

`secret-ids`

보안 암호 ARNS, 이름 및 이름 접두사.

환경 변수 이름을 설정하려면 보안 암호 ID 앞에 환경 변수 이름을 입력한 다음 쉼표를 입력합니다. 예를 들어 `ENV_VAR_1, secretId`는 보안 암호 `secretId`에서 ENV_VAR_1이라는 환경 변수를 생성합니다. 환경 변수 이름은 대문자, 숫자, 밑줄로 구성될 수 있습니다.

접두사를 사용하려면 3자 이상을 입력하고 그 뒤에 별표를 붙입니다. 예를 들어 `dev*`는 모든 보안 암호를 dev로 시작하는 이름과 일치시킵니다. 검색할 수 있는 일치하는 보안 암호의 최대 개수는 100개입니다. 변수 이름을 설정하고 접두사가 여러 보안 암호와 일치하는 경우 작업이 실패합니다.

`name-transformation`

기본적으로 이 단계에서는 보안 암호 이름에서 각 환경 변수 이름을 생성하며, 이름은 대문자, 숫자 및 밑줄만 포함하고 숫자로 시작되지 않도록 변환됩니다. 이름에 있는 문자의 경우, `lowercase`를

활용하여 소문자를 사용하도록 단계를 구성하거나, none을 활용하여 문자의 대소문자를 변경하지 않도록 할 수 있습니다. 기본값은 uppercase입니다.

`parse-json-secrets`

(선택 사항) 기본적으로 이 작업에서는 환경 변수 값을 보안 암호 값의 전체 JSON 문자열로 설정합니다. JSON에 있는 각 키-값 쌍의 환경 변수를 생성하려면 `parse-json-secrets`를 `true`로 설정합니다.

JSON에서 “name” 및 “Name”과 같이 대소문자를 구분하는 키를 사용하는 경우 이 작업에서는 중복 이름 충돌이 발생합니다. 이 경우 `parse-json-secrets`를 `false`로 설정하고 JSON 보안 암호 값을 별도로 구문 분석합니다.

환경 변수 이름 지정

작업에 의해 생성된 환경 변수의 이름은 해당 변수의 보안 암호와 동일하게 지정됩니다. 환경 변수는 보안 암호보다 이름 지정 요구 사항이 더 엄격하므로, 작업을 수행하면 보안 암호 이름이 해당 요구 사항에 맞게 변환됩니다. 예를 들어 이 작업을 수행하면 소문자가 대문자로 변환됩니다. 보안 암호의 JSON을 구문 분석할 경우, 환경 변수 이름에는 보안 암호 이름과 JSON 키 이름(예: `MYSECRET_KEYNAME`)이 모두 포함됩니다. 소문자를 변환하지 않도록 작업을 구성할 수 있습니다.

두 환경 변수의 이름이 같으면 작업이 실패하게 됩니다. 이러한 경우, 환경 변수에 사용할 이름을 별칭으로 지정해야 합니다.

이름이 충돌할 수 있는 경우의 예:

- 이름이 ‘MySecret’인 보안 암호와 ‘mysecret’인 보안 암호는 둘 다 ‘MYSECRET’이라는 환경 변수가 됩니다.
- 이름이 ‘Secret_keyname’인 보안 암호와 ‘keyname’이라는 키가 포함된 ‘Secret’이라는 JSON 구문 분석 암호는 둘 다 ‘SECRET_KEYNAME’이라는 환경 변수가 됩니다.

다음 예제와 같이 별칭을 지정하여 환경 변수 이름을 설정할 수 있습니다. 그러면 `ENV_VAR_NAME`이라는 변수가 생성됩니다.

```
secret-ids: |
  ENV_VAR_NAME, secretId2
```

빈 별칭

- `parse-json-secrets: true`를 설정하고 빈 별칭을 입력한 다음 쉼표와 보안 암호 ID를 입력할 경우, 이 작업으로 인해 환경 변수의 이름은 구문 분석된 JSON 키와 동일하게 지정됩니다. 변수 이름에는 보안 암호 이름이 포함되지 않습니다.

  보안 암호에 유효한 JSON이 포함되어 있지 않은 경우 작업은 하나의 환경 변수를 생성하고, 환경 변수의 이름을 보안 암호 이름과 동일하게 지정합니다.
- `parse-json-secrets: false`를 설정하고 빈 별칭을 입력한 다음 쉼표와 보안 암호 ID를 입력할 경우, 이 작업은 환경 변수의 이름을 별칭을 지정하지 않은 것처럼 지정합니다.

다음 예제는 빈 별칭을 보여줍니다.

```
,secret2
```

예제

Example 1 이름 및 ARN으로 보안 암호 가져오기

다음 예제에서는 이름 및 ARN으로 식별된 보안 암호에 대한 환경 변수를 생성합니다.

```
- name: Get secrets by name and by ARN
  uses: aws-actions/aws-secretsmanager-get-secrets@v2
  with:
    secret-ids: |
      exampleSecretName
      arn:aws:secretsmanager:us-east-2:123456789012:secret:test1-a1b2c3
      0/test/secret
      /prod/example/secret
      SECRET_ALIAS_1,test/secret
      SECRET_ALIAS_2,arn:aws:secretsmanager:us-east-2:123456789012:secret:test2-a1b2c3
      ,secret2
```

생성된 환경 변수:

```
EXAMPLESECRETNAME: secretValue1
TEST1: secretValue2
_0_TEST_SECRET: secretValue3
_PROD_EXAMPLE_SECRET: secretValue4
SECRET_ALIAS_1: secretValue5
```

```
SECRET_ALIAS_2: secretValue6
SECRET2: secretValue7
```

Example 2 접두사로 시작하는 모든 보안 암호 가져오기

다음 예제에서는 *beta*로 시작하는 이름을 가진 모든 보안 암호에 대한 환경 변수를 생성합니다.

```
- name: Get Secret Names by Prefix
  uses: 2
  with:
    secret-ids: |
      beta*    # Retrieves all secrets that start with 'beta'
```

생성된 환경 변수:

```
BETASECRETNAME: secretValue1
BETATEST: secretValue2
BETA_NEWSECRET: secretValue3
```

Example 3 보안 암호에서 JSON 구문 분석

다음 예제에서는 보안 암호에서 JSON을 구문 분석하여 환경 변수를 생성합니다.

```
- name: Get Secrets by Name and by ARN
  uses: aws-actions/aws-secretsmanager-get-secrets@v2
  with:
    secret-ids: |
      test/secret
      ,secret2
    parse-json-secrets: true
```

`test/secret` 보안 암호의 보안 암호 값은 다음과 같습니다.

```
{
  "api_user": "user",
  "api_key": "key",
  "config": {
    "active": "true"
  }
}
```

secret2 보안 암호의 보안 암호 값은 다음과 같습니다.

```
{
  "myusername": "alejandro_rosalez",
  "mypassword": "EXAMPLE_PASSWORD"
}
```

생성된 환경 변수:

```
TEST_SECRET_API_USER: "user"
TEST_SECRET_API_KEY: "key"
TEST_SECRET_CONFIG_ACTIVE: "true"
MYUSERNAME: "alejandro_rosalez"
MYPASSWORD: "EXAMPLE_PASSWORD"
```

Example환경 변수 이름에 4가지 소문자 사용

다음 예제에서는 소문자 이름을 사용하여 환경 변수를 생성합니다.

```
- name: Get secrets
  uses: aws-actions/aws-secretsmanager-get-secrets@v2
  with:
    secret-ids: exampleSecretName
    name-transformation: lowercase
```

생성된 환경 변수:

```
examplesecretname: secretValue
```

GitLab AWS Secrets Manager 에서 사용

AWS Secrets Manager 는 GitLab과 통합됩니다. Secrets Manager 보안 암호를 활용하여 GitLab 자격 증명을 보호할 수 있으며, 더 이상 GitLab에 하드코딩할 필요가 없습니다. 대신, GitLab Runner는 GitLab CI/CD 파이프라인에서 애플리케이션이 작업을 실행할 때 Secrets Manager에서 이러한 보안 암호를 가져옵니다.

이 통합을 사용하려면 IAM 및 IAM 역할에 OpenID Connect(OIDC) 자격 증명 공급자 AWS Identity and Access Management 를 생성합니다. 이를 통해 GitLab Runner가 Secrets Manager 보안 암호에 액세스할 수 있습니다. GitLab CI/CD 및 OIDC에 대한 자세한 내용은 GitLab 설명서를 참조하세요.

고려 사항

비공개 GitLab 인스턴스를 사용하는 경우, 이 Secrets Manager 통합 기능을 사용할 수 없습니다. 이 경우 비공개 인스턴스에 대한 GitLab 문서를 참조하세요.

사전 조건

Secrets Manager를 GitLab과 통합하려면 다음 사전 준비 단계를 완료해야 합니다.

1. AWS Secrets Manager 보안 암호 생성

   GitLab 작업에서 가져올 Secrets Manager 보안 암호를 생성해야 합니다. 이렇게 하면 자격 증명을 GitLab 설정 파일에 하드코딩할 필요가 없습니다. GitLab 파이프라인을 구성할 때 Secrets Manager 보안 암호 ID가 필요합니다. 자세한 내용은 AWS Secrets Manager 보안 암호 생성 섹션을 참조하세요.
2. IAM 콘솔에서 GitLab을 OIDC 공급자로 설정

   이 단계에서는 IAM 콘솔에서 GitLab을 OIDC 공급자로 등록합니다. 자세한 내용은 OpenID Connect(OIDC) 자격 증명 공급자 생성 및 GitLab 설명서를 참조하세요.

   OIDC 공급자를 생성할 때는 다음 구성을 사용합니다.

   a. `provider URL`을 GitLab 인스턴스로 설정합니다. 예를 들어 **`gitlab.example.com`**입니다.

   b. `audience` 또는 `aud`를 **`sts.amazonaws.com`**으로 설정합니다.
3. IAM 역할 및 정책 생성

   IAM 정책 및 역할을 생성해야 합니다. 이 역할은 GitLab이 AWS Security Token Service (STS)를 통해 담당합니다. 자세한 내용은 사용자 지정 신뢰 정책을 사용하여 역할 생성을 참조하세요.

   a. IAM 콘솔에서 IAM 역할을 생성할 때 다음 설정을 사용합니다.

   - `Trusted entity type`을 **`Web identity`**으로 설정합니다.
   - `Group`를 **`your GitLab group`**로 설정합니다.
   - `Identity provider`를 2단계에서 사용한 것과 동일한 공급자 URL(GitLab 인스턴스)로 설정합니다.
   - `Audience`를 2단계에서 사용한 것과 동일한 대상으로 설정합니다.

b. 다음은 GitLab가 역할을 담당하도록 허용하는 신뢰 정책의 예입니다. 신뢰 정책에는 AWS 계정, GitLab URL 및 프로젝트 경로가 나열되어야 합니다.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:oidc-
provider/gitlab.example.com"
      },
      "Condition": {
        "StringEquals": {
          "gitlab.example.com:aud": [
            "sts.amazon.com"
          ]
        },
        "StringLike": {
          "gitlab.example.com:sub": [
            "project_path:mygroup/project-*:ref_type:branch-*:ref:main*"
          ]
        }
      }
    }
  ]
}
```

c. 또한 GitLab이 AWS Secrets Manager액세스할 수 있도록 허용하는 IAM 정책도 생성해야 합니다. 이 정책을 신뢰 정책에 추가할 수 있습니다. 자세한 내용은 IAM 정책 생성을 참조하세요.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "secretsmanager:GetSecretValue",
      "Resource": "arn:aws:secretsmanager:us-
east-1:111122223333:secret:your-secret"
```

```
    }
  ]
}
```

GitLab AWS Secrets Manager 과 통합

위의 사전 단계를 완료한 후, GitLab을 구성하여 Secrets Manager를 통해 자격 증명을 보호할 수 있습니다.

Secrets Manager를 사용하기 위해 GitLab 파이프라인 구성

GitLab CI/CD 구성 파일을 다음 정보로 업데이트해야 합니다.

- STS로 설정된 토큰의 대상.
- Secrets Manager 보안 암호.
- GitLab 파이프라인에서 작업을 실행할 때 GitLab Runner가 담당할 IAM 역할.
- 보안 암호가 저장 AWS 리전 되는 입니다.

GitLab은 Secrets Manager에서 보안 암호를 가져와 임시 파일에 이 값을 저장합니다. 이 파일의 경로는 파일 형식 CI/CD 변수와 유사한 CI/CD 변수에 저장됩니다.

다음은 GitLab CI/CD 구성 파일의 YAML 파일 예시입니다.

```
variables:
  AWS_REGION: us-east-1
  AWS_ROLE_ARN: 'arn:aws:iam::111122223333:role/gitlab-role'
job:
  id_tokens:
    AWS_ID_TOKEN:
      aud: 'sts.amazonaws.com'
  secrets:
    DATABASE_PASSWORD:
      aws_secrets_manager:
        secret_id: "arn:aws:secretsmanager:us-east-1:111122223333:secret:secret-name"
```

자세한 내용은 Secrets Manager 통합 설명서를 참조하세요.

원할 경우 GitLab에서 OIDC 구성을 테스트할 수도 있습니다. 자세한 내용은 OIDC 구성 테스트에 대한 GitLab 설명서를 참조하세요.

문제 해결

다음은 Secrets Manager를 GitLab과 통합할 때 발생할 수 있는 일반적인 문제를 해결하는 데 도움이 될 수 있습니다.

GitLab 파이프라인 관련 문제

GitLab 파이프라인에서 문제가 발생한 경우, 다음 사항을 확인하세요.

- YAML 파일 형식이 올바른지 확인합니다. 자세한 내용은 GitLab 설명서를 참조하세요.
- GitLab 파이프라인이 올바른 역할을 수임하고, 적절한 권한을 가지며, 올바른 AWS Secrets Manager 보안 암호에 액세스합니다.

추가 리소스

다음 리소스는 GitLab 및 AWS Secrets Manager관련 문제를 해결하는 데 도움이 될 수 있습니다.

- GitLab OIDC 문제 해결
- GitLab CI/CD 파이프라인 디버깅
- 문제 해결

에서 AWS Secrets Manager 보안 암호 사용 AWS IoT Greengrass

AWS IoT Greengrass 는 클라우드 기능을 로컬 디바이스로 확장하는 소프트웨어입니다. 덕분에 장치는 정보 소스와 보다 밀접한 데이터를 수집 및 분석하고, 로컬 이벤트에 자율적으로 응답하며, 로컬 네트워크에서 서로 안전하게 통신할 수 있습니다.

AWS IoT Greengrass 를 사용하면 하드 코딩 암호, 토큰 또는 기타 보안 암호 없이 AWS IoT Greengrass 디바이스의 서비스 및 애플리케이션으로 인증할 수 있습니다. AWS Secrets Manager 를 사용하여 클라우드에서 보안 암호를 안전하게 저장하고 관리할 수 있습니다. AWS IoT Greengrass 는 Secrets Manager를 AWS IoT Greengrass 코어 디바이스에 확장하므로 커넥터와 Lambda 함수가 로컬 보안 암호를 사용하여 서비스 및 애플리케이션과 상호 작용할 수 있습니다.

보안 암호를 AWS IoT Greengrass 그룹에 통합하려면 Secrets Manager 보안 암호를 참조하는 그룹 리소스를 생성합니다. 이 보안 암호 리소스는 연결된 ARN을 사용하여 클라우드 보안 암호를 참조합니다. 보안 암호 리소스를 생성, 관리 및 사용하는 방법을 알아보려면 AWS IoT 개발자 안내서의 보안 암호 리소스 작업을 참조하세요.

AWS IoT Greengrass 코어에 보안 암호를 배포하려면 코어에 보안 암호 배포를 AWS IoT Greengrass 참조하세요.

Parameter Store에서 AWS Secrets Manager 보안 암호 사용

AWS Systems Manager Parameter Store는 구성 데이터 관리 및 보안 암호 관리를 위한 안전한 계층적 스토리지를 제공합니다. 암호, 데이터베이스 문자열, 라이선스 코드와 같은 데이터를 파라미터 값으로 저장할 수 있습니다. 하지만 파라미터 스토어에서는 저장된 보안 암호에 대한 자동 교체 서비스를 제공하지 않습니다. 대신, 파라미터 스토어를 사용하여 Secrets Manage에 보안 암호를 저장한 다음 해당 보안 암호를 파라미터 스토어 파라미터로 참조할 수 있습니다.

Secrets Manage를 사용하여 파라미터 스토어를 구성하는 경우 `secret-id` 파라미터 스토어의 이름 문자열 앞에 슬래시(/)가 필요합니다.

자세한 내용은 AWS Systems Manager 사용 설명서의 파라미터 스토어 파라미터에서 AWS Secrets Manager 보안 암호 참조를 참조하세요.

AWS Secrets Manager 보안 암호 교체

교체는 주기적으로 보안 암호를 업데이트하는 프로세스입니다. 보안 암호를 교체하면 보안 암호 및 데이터베이스 또는 서비스 모두에서 자격 증명이 업데이트됩니다. Secrets Manager에서 보안 암호에 대한 자동 교체를 설정할 수 있습니다. 두 가지 형태의 교체가 있습니다.

- 관리형 교체 - 대부분의 관리형 보안 암호에는 관리형 교체를 사용합니다. 여기서 서비스는 교체를 구성하고 관리합니다. 관리형 교체는 Lambda 함수를 사용하지 않습니다.
- Secrets Manager 관리형 외부 보안 암호 교체 - Secrets Manager 파트너가 보유한 보안 암호의 경우 관리형 외부 보안 암호 교체를 사용하여 파트너의 시스템에서 보안 암호를 업데이트합니다. 여기에는 Lambda 함수가 필요하지 않습니다.
- the section called “Lambda 함수로 교체” - 다른 유형의 보안 암호의 경우, Secrets Manager 교체는 Lambda 함수를 사용하여 보안 암호와 데이터베이스 또는 서비스를 업데이트합니다.

AWS Secrets Manager 보안 암호에 대한 관리형 교체

일부 서비스는 서비스에서 자동으로 교체를 구성하고 관리하는 관리형 교체를 제공합니다. 관리형 교체에서는 AWS Lambda 함수를 사용하여 데이터베이스의 보안 암호와 자격 증명을 업데이트하지 않습니다.

다음 서비스는 관리형 교체를 제공합니다:

- Amazon Aurora는 마스터 사용자 자격 증명에 관리형 교체를 제공합니다. 자세한 내용은 Amazon Aurora 사용 설명서의 Amazon Aurora 및 AWS Secrets Manager을 사용한 암호 관리를 참조하세요.
- Amazon ECS Service Connect는 AWS Private Certificate Authority TLS 인증서에 대한 관리형 교체를 제공합니다. 자세한 내용은 Amazon Elastic Container Service 개발자 가이드의 Service Connect를 활용한 TLS를 참조하세요.
- Amazon RDS는 마스터 사용자 자격 증명에 관리형 교체를 제공합니다. 자세한 내용은 Amazon RDS 사용 설명서의 Amazon RDS 및 AWS Secrets Manager을 사용한 암호 관리를 참조하세요.
- Amazon DocumentDB는 마스터 사용자 자격 증명에 대한 관리형 교체를 제공합니다. 자세한 내용은 Amazon DocumentDB 사용 설명서의 Amazon DocumentDB를 사용한 암호 관리 및 AWS Secrets Manager 섹션을 참조하세요. Amazon DocumentDB
- Amazon Redshift는 관리자 암호에 대한 관리형 교체 기능을 제공합니다. 자세한 내용은 Amazon Redshift 관리 가이드의 AWS Secrets Manager를 사용한 Amazon Redshift 관리자 암호 관리를 참조하세요.

- 관리형 외부 보안 암호는 Secrets Manager 파트너가 보유한 보안 암호에 대한 관리형 교체를 제공합니다. 자세한 내용은 AWS Secrets Manager 관리형 외부 보안 암호를 사용하여 타사 보안 암호 관리 단원을 참조하십시오.

> **Tip**
>
> 다른 유형의 보안 암호에 대해서는 the section called “Lambda 함수로 교체” 섹션을 참조하세요.

관리형 보안 암호의 교체는 일반적으로 1분 이내에 완료됩니다. 교체 중에 보안 암호를 검색하는 새 연결은 이전 버전의 보안 인증 정보를 가져올 수 있습니다. 애플리케이션에서는 마스터 사용자를 사용하는 대신에 애플리케이션에 필요한 최소한의 권한으로 생성한 데이터베이스 사용자를 사용하는 모범 사례를 따르는 것이 좋습니다. 애플리케이션 사용자의 경우 가용성을 극대화하기 위해 대체 사용자 교체 전략을 사용할 수 있습니다.

Secrets Manager 파트너가 보유한 보안 암호의 경우

관리형 교체 일정을 변경하는 방법

1. Secrets Manager 콘솔에서 관리형 보안 암호를 엽니다. 관리 서비스에서 링크를 따라가거나 Secrets Manager 콘솔에서 보안 암호를 검색할 수 있습니다.
2. Rotation schedule(교체 일정)에서 Schedule expression builder(예약 표현식 빌더) 또는 Schedule expression(예약 표현식)으로 일정(UTC 표준 시간대)을 입력합니다. Secrets Manager는 일정을 `rate()` 또는 `cron()` 표현식으로 저장합니다. 교체 기간은 Start time(시작 시간)을 지정하지 않는 한 자정에 자동으로 시작됩니다. 보안 암호를 4시간마다 교체할 수 있습니다. 자세한 내용은 교체 일정 단원을 참조하십시오.
3. (선택 사항) Window duration(지속 시간)에서 Secrets Manager가 보안 암호를 교체할 기간의 길이를 입력합니다. 예를 들어 **3h**는 3시간입니다. 기간은 그 다음 교체 기간까지 연장되지 않아야 합니다. 시간 단위의 교체 일정에서 지속 시간을 지정하지 않으면 한 시간 후에 자동으로 종료됩니다. 일 단위의 교체 일정인 경우 해당 날짜의 하루가 끝나면 자동으로 종료됩니다.
4. 저장을 선택합니다.

관리형 교체 일정 변경하기 (AWS CLI)

- `rotate-secret`를 호출합니다. 다음 예에서는 매월 1일과 15일 16:00 ~ 18:00 (UTC) 사이에 암호가 교체됩니다. 자세한 내용은 교체 일정 단원을 참조하십시오.

```
aws secretsmanager rotate-secret \
    --secret-id MySecret \
    --rotation-rules \
        "{\"ScheduleExpression\": \"cron(0 16 1,15 * ? *)\", \"Duration\": \"2h\"}"
```

Secrets Manager 관리형 외부 보안 암호 교체

Secrets Manager는 일부 소프트웨어 공급업체와 협력하여 관리형 외부 보안 암호를 제공합니다. 이 기능은 교체를 자동으로 처리하여 고객이 보안 암호 수명 주기를 관리하는 데 도움이 됩니다. 관리형 외부 보안 암호를 사용하면 고객은 더 이상 서로 다른 파트너에게 저장된 각 보안 암호에 대해 특정 교체 로직을 유지할 필요가 없습니다. 이는 Secrets Manager에서 처리합니다.

Secrets Manager에 온보딩된 파트너 목록을 보려면 관리형 외부 보안 암호 파트너를 참조하세요.

콘솔에서 교체 설정

각 통합 파트너가 지정한 보안 암호 유형과 값을 지정하여 생성한 기존 관리형 외부 보안 암호에 대한 교체를 구성하려면 다음 단계를 사용합니다.

1. Secrets Manager 콘솔을 엽니다.
2. 목록에서 관리형 외부 보안 암호를 선택합니다.
3. 구성 탭을 선택합니다.
4. 교체 구성 섹션에서 교체 편집을 선택합니다.
5. Automatic rotation(자동 교체)을 켭니다.
6. 교체 메타데이터에서 교체에 필요한 파트너별 메타데이터를 추가합니다.

   기타 필수 메타데이터에 대해 통합 파트너가 제공하는 지침을 따릅니다.
7. 보안 암호 교체를 위한 서비스 권한에서 교체를 위한 IAM 역할을 선택하거나 생성합니다.
   - 새 역할 생성을 선택하여 필요한 권한이 있는 역할을 자동으로 생성합니다.
   - 또는 파트너에게 적절한 권한이 있는 기존 역할을 선택합니다.

   기본적으로 권한은 보안 암호가 생성된 리전의 개별 파트너로 범위가 지정됩니다.
8. 교체 일정을 설정합니다(예: 30일마다 자동으로 교체).
9. 저장을 선택하여 교체 구성을 적용합니다.

이 프로세스 중에 구성된 두 가지 주요 메타데이터 필드는 다음과 같습니다.

필드	설명
ExternalSecretRotationMetadata	Salesforce용 API 버전과 같은 교체에 필요한 파트너별 메타데이터
ExternalSecretRotationRoleArn	교체에 사용되는 IAM 역할의 ARN으로, 통합 파트너로 범위가 지정된 권한 포함

이러한 필드에 대한 자세한 내용은 Secrets Manager 관리형 외부 보안 암호를 사용하여 타사 보안 암호 관리를 참조하세요.

CLI를 사용하여 교체 설정

다음 명령을 실행하여 Salesforce 보안 암호에 대한 교체를 설정합니다. 이 명령은 보안 암호 ID, 교체를 위한 IAM 역할 ARN, 교체 일정 및 교체 프로세스에 필요한 파트너별 메타데이터를 지정합니다.

```
aws secretsmanager rotate-secret \
            --secret-id SampleSecret \
            --external-secret-rotation-role-arn arn:aws:iam::123412341234:role/xyz \
            --rotation-rules AutomaticallyAfterDays=1 \
            --external-secret-rotation-metadata
 '[{"Key":"apiVersion","Value":"v65.0"}]'
```

Lambda 함수로 교체

여러 유형의 보안 암호의 경우 Secrets Manager는 AWS Lambda 함수를 사용하여 보안 암호와 데이터베이스 또는 서비스를 업데이트합니다. Lambda 함수 사용 비용에 대한 자세한 내용은 가격 책정 섹션을 참조하세요.

일부 다른 서비스에서 관리하는 보안 암호의 경우 관리형 교체를 사용합니다. 관리형 교체을(를) 사용하려면 먼저 관리 서비스를 통해 보안 암호를 생성해야 합니다.

교체하는 동안 Secrets Manager는 교체 상태를 나타내는 이벤트를 로그합니다. 자세한 내용은 the section called “를 사용하여 로그 AWS CloudTrail ” 단원을 참조하십시오.

보안 암호를 교체하기 위해 Secrets Manager는 설정한 교체 일정에 따라 Lambda 함수를 호출합니다. 자동 교체가 설정된 상태에서 암호 값을 수동으로도 업데이트하는 경우, Secrets Manager는 다음 교체 날짜를 계산할 때 이를 유효한 교체로 간주합니다.

교체를 수행하는 동안 Secrets Manager에서는 파라미터가 다를 때마다 동일한 함수를 여러 번 호출합니다. Secrets Manager에서는 파라미터의 다음 JSON 요청 구조를 사용하여 함수를 호출합니다.

```
{
    "Step" : "request.type",
    "SecretId" : "string",
    "ClientRequestToken" : "string",
    "RotationToken" : "string"
}
```

파라미터:

- 단계 - 교체 단계: `create_secret`, `set_secret`, `test_secret`, 또는 `finish_secret`. 자세한 내용은 the section called “교체 함수의 4단계” 단원을 참조하십시오.
- SecretId - 교체할 보안 암호의 ARN입니다.
- ClientRequestToken - 새 버전의 보안 암호에 대한 고유 식별자입니다. 이 값은 멱등성을 보장하는 데 도움이 됩니다. 자세한 내용은 AWS Secrets Manager API 참조의 PutSecretValue: ClientRequestToken을 참조하세요.
- RotationToken - 요청의 소스를 나타내는 고유 식별자입니다. 다른 계정에서 Lambda 교체 함수를 사용하여 한 계정의 보안 암호를 교체하는 수임된 역할 또는 교차 계정 회전에 의한 보안 암호 교체 시 필요합니다. 두 경우 모두, 교체 함수는 Secrets Manager를 호출하기 위해 IAM 역할을 담당하며, 이후 Secrets Manager는 교체 토큰을 사용하여 IAM 역할의 신원을 검증합니다.

교체 단계가 실패하면 Secrets Manager는 전체 교체 프로세스를 여러 번 다시 시도합니다.

주제

- Amazon Aurora, Amazon Redshift 또는 Amazon DocumentDB 보안 암호 자동 교체 설정
- 데이터베이스가 아닌 AWS Secrets Manager 보안 암호에 대한 자동 교체 설정
- 를 사용하여 자동 교체 설정 AWS CLI
- Lambda 함수 교체 전략
- Lambda 교체 함수
- AWS Secrets Manager 교체 함수 템플릿

- 에 대한 Lambda 교체 함수 실행 역할 권한 AWS Secrets Manager
- AWS Lambda 교체 함수에 대한 네트워크 액세스
- AWS Secrets Manager 교체 문제 해결

Amazon Aurora, Amazon Redshift 또는 Amazon DocumentDB 보안 암호 자동 교체 설정

이 자습서에서는 데이터베이스 보안 암호에 대해 the section called “Lambda 함수로 교체”를 설정하는 방법을 설명합니다. 교체는 주기적으로 보안 암호를 업데이트하는 프로세스입니다. 보안 암호를 교체하면 보안 암호와 데이터베이스 모두에서 보안 인증 정보가 업데이트됩니다. Secrets Manager에서 데이터베이스 보안 암호에 대한 자동 교체를 설정할 수 있습니다.

콘솔을 사용하여 교체를 설정하려면 먼저 교체 전략을 선택해야 합니다. 그런 다음 교체에 대한 보안 암호를 구성하여 Lambda 교체 함수(아직 없는 경우)를 생성합니다. 콘솔은 Lambda 함수 실행 역할에 대한 권한도 설정합니다. 마지막 단계에서는 Lambda 교체 함수가 네트워크를 통해 Secrets Manager와 데이터베이스 모두에 액세스할 수 있는지 확인합니다.

> **Warning**
>
> 자동 교체를 켜려면 Lambda 교체 함수에 대한 IAM 실행 역할을 생성하고 이 역할에 권한 정책을 연결할 수 있는 권한이 있어야 합니다. `iam:CreateRole` 및 `iam:AttachRolePolicy` 권한이 모두 필요합니다. 자격 증명에 이러한 권한을 부여하면 해당 자격 증명은 자신에게 모든 권한을 부여할 수 있습니다.

단계:

- 1단계: 교체 전략 선택 및 (선택 사항) 슈퍼유저 보안 암호 생성
- 2단계: 교체 구성 및 교체 함수 생성
- 3단계: (선택 사항) 교체 함수에 대한 추가 권한 조건 설정
- 4단계: 교체 함수에 대한 네트워크 액세스 설정
- 다음 단계

1단계: 교체 전략 선택 및 (선택 사항) 슈퍼유저 보안 암호 생성

Secrets Manager에서 제공하는 전략에 대한 내용은 the section called “Lambda 함수 교체 전략” 섹션을 참조하세요.

대체 사용자 전략을 선택하는 경우 보안 암호 생성하고 데이터베이스 슈퍼유저 보안 인증을 저장해야 합니다. 교체에서는 첫 번째 사용자를 복제하므로 슈퍼유저 보안 인증을 가진 보안 암호가 필요하며, 대부분의 사용자는 해당 권한이 없습니다. Amazon RDS Proxy는 대체 사용자 전략을 지원하지 않습니다.

2단계: 교체 구성 및 교체 함수 생성

Amazon RDS, Amazon DocumentDB, Amazon Redshift 보안 암호 교체를 켜려면

1. https://console.aws.amazon.com/secretsmanager/에서 Secrets Manager 콘솔을 엽니다.
2. 보안 암호(Secrets) 페이지에서 보안 암호를 선택합니다.
3. 보안 암호 세부 정보(Secret details) 페이지의 교체 구성(Rotation configuration) 섹션에서 교체 편집(Edit rotation)을 선택합니다.
4. Edit rotation configuration(교체 구성 편집) 대화 상자에서 다음을 수행합니다.

   a. Automatic rotation(자동 교체)을 켭니다.

   b. Rotation schedule(교체 일정)에서 Schedule expression builder(예약 표현식 빌더) 또는 Schedule expression(예약 표현식)으로 일정(UTC 표준 시간대)을 입력합니다. Secrets Manager는 일정을 `rate()` 또는 `cron()` 표현식으로 저장합니다. 교체 기간은 Start time(시작 시간)을 지정하지 않는 한 자정에 자동으로 시작됩니다. 보안 암호를 4시간마다 교체할 수 있습니다. 자세한 내용은 교체 일정 단원을 참조하십시오.

   c. (선택 사항) Window duration(지속 시간)에서 Secrets Manager가 보안 암호를 교체할 기간의 길이를 입력합니다. 예를 들어 **3h**는 3시간입니다. 기간은 그 다음 교체 기간까지 연장되지 않아야 합니다. 시간 단위의 교체 일정에서 지속 시간을 지정하지 않으면 한 시간 후에 자동으로 종료됩니다. 일 단위의 교체 일정인 경우 해당 날짜의 하루가 끝나면 자동으로 종료됩니다.

   d. (선택 사항) 변경 사항을 저장할 때 보안 암호가 교체되게 하려면 보안 암호를 저장할 때 즉시 교체(Rotate immediately when the secret is stored)를 선택합니다. 확인란의 선택을 취소하는 경우에는 설정한 예약에 따라 첫 번째 교체가 시작됩니다.

      예를 들어 3단계와 4단계를 아직 완료하지 않아서 교체에 실패하는 경우 Secrets Manager는 교체 프로세스를 여러 번 재시도합니다.

   e. 교체 함수(Rotation function)에서 다음 중 하나를 수행합니다.

      - Create a new Lambda function(새 Lambda 함수 생성)을 선택하고 새 함수의 이름을 입력합니다. Secrets Manager에서 함수 이름의 시작 부분에 `SecretsManager`를 추가합니다. Secrets Manager는 적절한 템플릿을 기반으로 함수를 생성하고 Lambda 실행 역할에 필요한 권한을 설정합니다.

- 다른 보안 암호에 사용한 교체 함수를 재사용하려면 Use an existing Lambda function(기존 Lambda 함수 사용)을 선택합니다. Recommended VPC configurations(권장 VPC 구성)에 나열된 교체 함수에는 데이터베이스와 동일한 VPC 및 보안 그룹이 있으므로, 함수가 데이터베이스에 액세스하는 데 도움이 됩니다.

f. 교체 전략에서 단일 사용자 또는 대체 사용자 전략을 선택합니다. 자세한 내용은 the section called “1단계: 교체 전략 선택 및 (선택 사항) 슈퍼유저 보안 암호 생성” 단원을 참조하십시오.

5. 저장을 선택합니다.

3단계: (선택 사항) 교체 함수에 대한 추가 권한 조건 설정

교체 함수에 대한 리소스 정책에서는 컨텍스트 키 `aws:SourceAccount`를 포함하여 Lambda가 혼동된 대리자로 사용되지 않도록 하는 것이 좋습니다. 일부 AWS 서비스의 경우 혼동된 대리자 시나리오를 방지하려면 `aws:SourceArn` 및 `aws:SourceAccount` 전역 조건 키를 모두 사용하는 것이 AWS 좋습니다. 그러나 교체 함수 정책에 `aws:SourceArn` 조건을 포함하는 경우 교체 함수는 해당 ARN에서 지정한 보안 암호를 교체하는 데만 사용할 수 있습니다. 여러 보안 암호에 대해 교체 기능을 사용할 수 있도록 컨텍스트 키 `aws:SourceAccount`만 포함하는 것이 좋습니다.

교체 함수 리소스 정책을 업데이트하려면

1. Secrets Manager 콘솔에서 보안 암호를 선택한 다음 세부 정보 페이지의 Rotation configuration(교체 구성)에서 Lambda 교체 함수를 선택합니다. Lambda 콘솔이 열립니다.
2. Lambda에 리소스 기반 정책 사용의 지침에 따라 `aws:sourceAccount` 조건을 추가합니다.

```
"Condition": {
    "StringEquals": {
        "AWS:SourceAccount": "123456789012"
    }
},
```

AWS 관리형 키 `aws/secretsmanager` 이외의 KMS 키를 사용하여 보안 암호를 암호화할 경우 Secrets Manager가 Lambda 실행 역할에 키 사용 권한을 부여합니다. SecretARN 암호화 컨텍스트를 사용하여 암호 해독 기능의 사용을 제한할 수 있으므로 교체 함수 역할은 교체를 담당하는 암호의 암호 해독에만 액세스할 수 있습니다.

교체 함수의 실행 역할 업데이트

1. Lambda 교체 함수에서 구성을 선택한 다음 실행 역할에서 역할 이름을 선택합니다.

2. kms:EncryptionContext:SecretARN 조건을 추가하려면 역할 권한 정책 수정의 지침을 따르세요.

```
"Condition": {
    "StringEquals": {
        "kms:EncryptionContext:SecretARN": "SecretARN"
    }
},
```

4단계: 교체 함수에 대한 네트워크 액세스 설정

자세한 내용은 the section called “AWS Lambda 교체 함수에 대한 네트워크 액세스” 단원을 참조하십시오.

다음 단계

the section called “교체 문제 해결”을(를) 참조하세요.

데이터베이스가 아닌 AWS Secrets Manager 보안 암호에 대한 자동 교체 설정

이 자습서에서는 비데이터베이스 보안 암호에 대해 the section called “Lambda 함수로 교체”를 설정하는 방법을 설명합니다. 교체는 주기적으로 보안 암호를 업데이트하는 프로세스입니다. 보안 암호를 교체하면 보안 암호와 보안 암호가 사용되는 데이터베이스 또는 서비스 모두에서 보안 인증이 업데이트됩니다.

데이터베이스 보안 암호는 데이터베이스 보안 암호 자동 교체(콘솔) 섹션을 참조하세요.

> **Warning**
>
> 자동 교체를 켜려면 Lambda 교체 함수에 대한 IAM 실행 역할을 생성하고 이 역할에 권한 정책을 연결할 수 있는 권한이 있어야 합니다. iam:CreateRole 및 iam:AttachRolePolicy 권한이 모두 필요합니다. 자격 증명에 이러한 권한을 부여하면 해당 자격 증명은 자신에게 모든 권한을 부여할 수 있습니다.

단계:

- 1단계: 일반 교체 함수 생성

- 2단계: 교체 함수 코드 작성
- 3단계: 보안 암호에 대한 교체 구성
- 4단계: 교체 함수가 Secrets Manager와 데이터베이스 또는 서비스에 액세스하도록 허용
- 5단계: Secrets Manager가 교체 함수를 간접적으로 호출하도록 허용
- 6단계: 교체 함수에 대한 네트워크 액세스 설정
- 다음 단계

1단계: 일반 교체 함수 생성

시작하려면 Lambda 교체 함수를 생성합니다. 보안 암호를 교체하기 위한 코드는 포함되지 않으므로 이후 단계에서 작성하겠습니다. 교체 함수의 작동 방식에 대한 내용은 the section called “Lambda 교체 함수” 섹션을 참조하세요.

지원되는 리전에서는 AWS Serverless Application Repository 를 사용하여 템플릿에서 함수를 생성할 수 있습니다. 지원되는 리전 목록은 AWS Serverless Application Repository FAQ 섹션을 참조하세요. 다른 리전의 경우, 함수를 처음부터 새로 생성하고 템플릿 코드를 함수에 복사합니다.

일반 교체 함수를 생성하는 방법

1. AWS Serverless Application Repository 가 해당 리전에서 지원되는지 확인하려면 AWS 일반 참조의 AWS Serverless Application Repository 엔드포인트 및 할당량을 참조하세요.
2. 다음 중 하나를 수행하세요.
   - 해당 리전에서 AWS Serverless Application Repository 가 지원되는 경우:
     a. 탐색 창에서 애플리케이션을 선택한 다음 애플리케이션 생성을 선택합니다.
     b. 애플리케이션 생성 페이지에서 서버리스 애플리케이션 탭을 선택합니다.
     c. 퍼블릭 애플리케이션 아래의 검색 상자에 **`SecretsManagerRotationTemplate`**을 입력합니다.
     d. Show apps that create custom IAM roles or resource policies를 선택합니다.
     e. SecretsManagerRotationTemplate 타일을 선택합니다.
     f. 검토, 구성 및 배포 페이지의 애플리케이션 설정 타일에서 필수 필드를 채웁니다.
        - 엔드포인트의 경우 **`https://`**를 포함하여, 리전의 엔드포인트를 입력합니다. 엔드포인트 목록은 the section called “Secrets Manager 엔드포인트” 섹션을 참조하세요.
        - Lambda 함수를 VPC에 입력하려면 vpcSecurityGroupIds 및 vpcSubnetIds를 포함합니다.

g. 배포(Deploy)를 선택합니다.

- AWS Serverless Application Repository 가 해당 리전에서 지원되지 않는 경우:

a. Lambda 콘솔 페이지의 함수를 선택하고 함수 생성을 선택합니다.

b. 함수 생성 페이지에서 다음을 수행합니다.

i. 새로 작성을 선택합니다.

ii. Function name(함수 이름)에 교체 함수의 이름을 입력합니다.

iii. 런타임에서 Python 3.10을 선택합니다.

iv. 함수 생성을 선택합니다.

2단계: 교체 함수 코드 작성

이 단계에서는 보안 암호를 업데이트하고, 보안 암호가 사용되는 서비스 또는 데이터베이스를 업데이트하는 코드를 작성합니다. 교체 함수를 직접 작성할 경우에 대한 팁을 포함하여, 교체 함수의 기능에 대한 자세한 내용은 the section called “Lambda 교체 함수” 섹션을 참조하세요. 교체 함수 템플릿도 참조로 사용할 수 있습니다.

3단계: 보안 암호에 대한 교체 구성

이 단계에서는 보안 암호에 대한 교체 일정을 설정하고, 교체 함수를 보안 암호에 연결합니다.

교체를 구성하고 교체 함수를 생성하려면

1. https://console.aws.amazon.com/secretsmanager/에서 Secrets Manager 콘솔을 엽니다.
2. 보안 암호(Secrets) 페이지에서 보안 암호를 선택합니다.
3. 보안 암호 세부 정보(Secret details) 페이지의 교체 구성(Rotation configuration) 섹션에서 교체 편집(Edit rotation)을 선택합니다. Edit rotation configuration(교체 구성 편집) 대화 상자에서 다음을 수행합니다.

a. Automatic rotation(자동 교체)을 켭니다.

b. Rotation schedule(교체 일정)에서 Schedule expression builder(예약 표현식 빌더) 또는 Schedule expression(예약 표현식)으로 일정(UTC 표준 시간대)을 입력합니다. Secrets Manager는 일정을 `rate()` 또는 `cron()` 표현식으로 저장합니다. 교체 기간은 Start time(시작 시간)을 지정하지 않는 한 자정에 자동으로 시작됩니다. 보안 암호를 4시간마다 교체할 수 있습니다. 자세한 내용은 교체 일정 단원을 참조하십시오.

c. (선택 사항) Window duration(지속 시간)에서 Secrets Manager가 보안 암호를 교체할 기간의 길이를 입력합니다. 예를 들어 **3h**는 3시간입니다. 기간은 그 다음 교체 기간까지 연장되지 않아야 합니다. 시간 단위의 교체 일정에서 지속 시간을 지정하지 않으면 한 시간 후에 자동으로 종료됩니다. 일 단위의 교체 일정인 경우 해당 날짜의 하루가 끝나면 자동으로 종료됩니다.

d. (선택 사항) 변경 사항을 저장할 때 보안 암호가 교체되게 하려면 보안 암호를 저장할 때 즉시 교체(Rotate immediately when the secret is stored)를 선택합니다. 확인란의 선택을 취소하는 경우에는 설정한 예약에 따라 첫 번째 교체가 시작됩니다.

e. 교체 함수에서 1단계에서 생성한 Lambda 함수를 선택합니다.

f. 저장을 선택합니다.

4단계: 교체 함수가 Secrets Manager와 데이터베이스 또는 서비스에 액세스하도록 허용

Lambda 교체 함수에는 Secrets Manager 보안 암호에 액세스할 수 있는 권한과, 데이터베이스 또는 서비스에 액세스할 수 있는 권한이 필요합니다. 이 단계에서는 Lambda 실행 역할에 이러한 권한을 부여합니다. AWS 관리형 키 `aws/secretsmanager` 이외의 KMS 키를 사용하여 보안 암호를 암호화할 경우 Lambda 실행 역할에 키 사용 권한을 부여해야 합니다. SecretARN 암호화 컨텍스트를 사용하여 암호 해독 기능의 사용을 제한할 수 있으므로 교체 함수 역할은 교체를 담당하는 암호의 암호 해독에만 액세스할 수 있습니다. 정책 예시는 교체 권한 섹션을 참조하세요.

지침은 AWS Lambda 개발자 가이드의 Lambda 실행 역할을 참조하세요.

5단계: Secrets Manager가 교체 함수를 간접적으로 호출하도록 허용

Secrets Manager가 설정한 교체 일정에 따라 교체 함수를 간접적으로 호출하도록 허용하려면, Lambda 함수의 리소스 정책에서 Secrets Manager 서비스 보안 주체에게 `lambda:InvokeFunction` 권한을 부여해야 합니다.

교체 함수에 대한 리소스 정책에서는 컨텍스트 키 `aws:SourceAccount`를 포함하여 Lambda가 혼동된 대리자로 사용되지 않도록 하는 것이 좋습니다. 일부 AWS 서비스의 경우 혼동된 대리자 시나리오를 방지하려면 `aws:SourceArn` 및 `aws:SourceAccount` 전역 조건 키를 모두 사용하는 것이 AWS 좋습니다. 그러나 교체 함수 정책에 `aws:SourceArn` 조건을 포함하는 경우 교체 함수는 해당 ARN에서 지정한 보안 암호를 교체하는 데만 사용할 수 있습니다. 여러 보안 암호에 대해 교체 기능을 사용할 수 있도록 컨텍스트 키 `aws:SourceAccount`만 포함하는 것이 좋습니다.

리소스 정책을 Lambda 함수에 연결하려면 Lambda에 대한 리소스 기반 정책 사용을 참조하세요.

다음 정책에서는 Secrets Manager가 Lambda 함수를 간접적으로 호출하도록 허용하는 방법을 보여줍니다.

JSON

```
{
    "Version":"2012-10-17",
    "Id": "default",
    "Statement": [
    {
        "Effect": "Allow",
        "Principal": {
            "Service": "secretsmanager.amazonaws.com"
            },
        "Action": "lambda:InvokeFunction",
        "Condition": {
            "StringEquals": {
                "AWS:SourceAccount": "123456789012"
            }
        },
        "Resource": "arn:aws:lambda:us-east-1:123456789012:function:function-
name"
    }
    ]
}
```

6단계: 교체 함수에 대한 네트워크 액세스 설정

이 단계에서는 교체 함수가 Secrets Manager에 연결하고, 보안 암호가 사용되는 서비스 또는 데이터베이스에도 연결할 수 있도록 허용합니다. 보안 암호를 교체하려면 교체 함수가 두 가지 모두에 액세스할 수 있어야 합니다. the section called “AWS Lambda 교체 함수에 대한 네트워크 액세스”을(를) 참조하세요.

다음 단계

3단계에서 교체를 구성한 경우, 보안 암호를 교체하기 위한 일정을 설정합니다. 예약된 시간에 교체가 실패하면 Secrets Manager는 교체를 여러 번 시도합니다. 보안 암호 즉시 교체의 지침에 따라 즉시 교체를 시작할 수도 있습니다.

교체에 실패한 경우 교체 문제 해결 단원을 참조하세요.

를 사용하여 자동 교체 설정 AWS CLI

이 자습서에서는를 사용하여 the section called “Lambda 함수로 교체”를 설정하는 방법을 설명합니다 AWS CLI. 보안 암호를 교체하면 보안 암호와 보안 암호가 사용되는 데이터베이스 또는 서비스 모두에서 보안 인증이 업데이트됩니다.

콘솔을 사용하여 교체를 설정할 수도 있습니다. 데이터베이스 보안 암호는 데이터베이스 보안 암호 자동 교체(콘솔) 섹션을 참조하세요. 다른 유형의 보안 암호에 대해서는 비데이터베이스 보안 암호 자동 교체(콘솔) 섹션을 참조하세요.

를 사용하여 교체를 설정하려면 데이터베이스 보안 암호를 교체하는 AWS CLI경우 먼저 교체 전략을 선택해야 합니다. 대체 사용자 전략을 선택하는 경우 별도의 보안 암호를 데이터베이스 슈퍼유저의 보안 인증과 함께 저장해야 합니다. 그런 다음 교체 함수 코드를 작성합니다. Secrets Manager는 함수의 기반이 될 수 있는 템플릿을 제공합니다. 그러면 코드를 사용하여 Lambda 함수를 생성하고 Lambda 함수와 Lambda 실행 역할 모두에 대한 권한을 설정합니다. 다음 단계에서는 Lambda 함수가 네트워크를 통해 Secrets Manager와 데이터베이스 또는 서비스 모두에 액세스할 수 있는지 확인합니다. 마지막으로 교체에 대한 보안 암호를 구성합니다.

단계:

- 데이터베이스 보안 암호의 사전 조건: 교체 전략 선택
- 1단계: 교체 함수 코드 작성
- 2단계: Lambda 함수 만들기
- 3단계: 네트워크 액세스 설정
- 4단계: 보안 암호에 대한 교체 구성
- 다음 단계

데이터베이스 보안 암호의 사전 조건: 교체 전략 선택

Secrets Manager에서 제공하는 전략에 대한 내용은 the section called “Lambda 함수 교체 전략” 섹션을 참조하세요.

옵션 1: 단일 사용자 전략

단일 사용자 전략을 선택할 경우 1단계를 계속 진행하면 됩니다.

옵션 2: 대체 사용자 전략

대체 사용자 전략을 선택할 경우 다음을 수행해야 합니다.

- 보안 암호를 생성하고 데이터베이스 슈퍼 사용자 자격 증명을 저장합니다. 대체 사용자 교체에서는 첫 번째 사용자를 복제하므로 슈퍼 사용자 보안 인증을 가진 보안 암호가 필요하며, 대부분의 사용자는 해당 권한이 없습니다.
- 수퍼 사용자 보안 암호의 ARN을 원본 보안 암호에 추가합니다. 자세한 내용은 the section called “보안 암호의 JSON 구조” 단원을 참조하십시오.

Amazon RDS Proxy는 대체 사용자 전략을 지원하지 않습니다.

1단계: 교체 함수 코드 작성

보안 암호를 교체하려면 교체 함수가 필요합니다. 교체 함수는 Secrets Manager에서 보안 암호를 교체하기 위해 호출하는 Lambda 함수입니다. 자세한 내용은 the section called “Lambda 함수로 교체” 단원을 참조하십시오. 이 단계에서는 보안 암호를 업데이트하고, 보안 암호가 사용되는 서비스 또는 데이터베이스를 업데이트하는 코드를 작성합니다.

Secrets Manager는 교체 함수 템플릿에서 Amazon RDS, Amazon Aurora, Amazon Redshift, Amazon DocumentDB 보안 암호에 대한 템플릿을 제공합니다.

교체 함수 코드를 작성하는 방법

1. 다음 중 하나를 수행하세요.
   - 교체 함수 템플릿 목록을 확인합니다. 서비스 및 교체 전략과 일치하는 항목이 있을 경우 해당 코드를 복사합니다.
   - 다른 유형의 보안 암호의 경우 교체 함수를 직접 작성할 수 있습니다. 지침은 the section called “Lambda 교체 함수” 섹션을 참조하세요.
2. 파일을 필요한 종속성과 함께 *my-function.zip* ZIP 파일에 저장합니다.

2단계: Lambda 함수 만들기

이 단계에서는 1단계에서 생성한 ZIP 파일을 사용하여 Lambda 함수를 생성합니다. Lambda 실행 역할은 함수가 호출될 때 Lambda가 맡는 역할입니다.

Lambda 교체 함수 및 실행 역할을 생성하려면

1. Lambda 실행 역할에 대한 신뢰 정책을 생성한 후 JSON 파일로 저장합니다. 예제와 추가 정보는 the section called “교체 권한” 섹션을 참조하세요. 정책은 다음을 충족해야 합니다.
   - 역할이 보안 암호에 대해 Secrets Manager 작업을 호출할 수 있도록 허용합니다.

- 역할이 보안 암호가 사용되는 서비스를 직접적으로 호출하여 새 암호를 생성하도록 허용합니다.

2. iam create-role을 호출하여 Lambda 실행 역할을 생성하고 이전 단계에서 생성한 신뢰 정책을 적용합니다.

```
aws iam create-role \
    --role-name rotation-lambda-role \
    --assume-role-policy-document file://trust-policy.json
```

3. lambda create-function을 호출하여 ZIP 파일에서 Lambda 함수를 생성합니다.

```
aws lambda create-function \
  --function-name my-rotation-function \
  --runtime python3.7 \
  --zip-file fileb://my-function.zip \
  --handler .handler \
  --role arn:aws:iam::123456789012:role/service-role/rotation-lambda-role
```

4. Secrets Manager가 lambda add-permission을 호출하여 호출할 수 있도록 Lambda 함수에 대한 리소스 정책을 설정합니다.

```
aws lambda add-permission \
  --function-name my-rotation-function \
  --action lambda:InvokeFunction \
  --statement-id SecretsManager \
  --principal secretsmanager.amazonaws.com \
  --source-account 123456789012
```

3단계: 네트워크 액세스 설정

자세한 내용은 the section called “AWS Lambda 교체 함수에 대한 네트워크 액세스” 단원을 참조하십시오.

4단계: 보안 암호에 대한 교체 구성

보안 암호에 대한 자동 교체를 켜려면 rotate-secret을 호출합니다. cron() 또는 rate() 일정 표현식을 사용하여 교체 일정을 설정하고 교체 기간을 설정할 수 있습니다. 자세한 내용은 the section called “교체 일정” 단원을 참조하십시오.

```
aws secretsmanager rotate-secret \
```

```
    --secret-id MySecret \
    --rotation-lambda-arn arn:aws:lambda:Region:123456789012:function:my-rotation-
function \
    --rotation-rules "{\"ScheduleExpression\": \"cron(0 16 1,15 * ? *)\", \"Duration\":
 \"2h\"}"
```

다음 단계

the section called “교체 문제 해결”을(를) 참조하세요.

Lambda 함수 교체 전략

the section called “Lambda 함수로 교체”의 경우, 데이터베이스 보안 암호에 대해 Secrets Manager는 두 가지 교체 전략을 제공합니다.

교체 전략: 단일 사용자

이 전략은 한 보안 암호로 한 사용자에 대한 보안 인증을 업데이트합니다. Amazon RDS Db2 인스턴스의 경우 사용자가 자신의 암호를 변경할 수 없으므로 별도의 보안 암호로 관리자 자격 증명을 제공해야 합니다. 이 전략은 가장 간단한 교체 전략이며 대부분의 사용 사례에 적합합니다. 특히 일회성 (임시) 또는 대화형 사용자의 보안 인증에 이 전략을 사용하는 것이 좋습니다.

보안 암호가 교체될 때 열린 데이터베이스 연결은 삭제되지 않습니다. 교체가 진행되는 동안 데이터베이스의 암호가 변경되는 시점부터 보안 암호가 업데이트되는 시점까지 짧은 기간이 소요됩니다. 이 시간동안 데이터베이스가 교체된 보안 인증을 사용하는 호출을 거부할 위험은 적습니다. 적절한 재시도 전략을 사용하여 이 위험을 완화할 수 있습니다. 교체 후에는 새 연결에서 새 보안 인증이 사용됩니다.

교체 전략: 대체 사용자

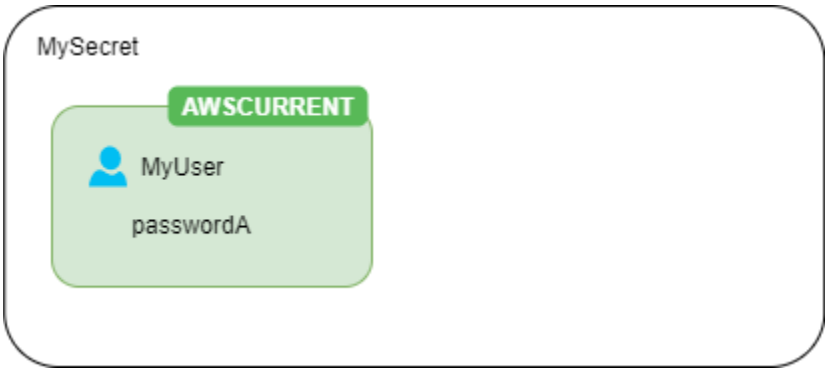
이 전략은 한 보안 암호로 두 사용자에 대한 보안 인증을 업데이트합니다. 첫 번째 사용자를 생성하고 첫 번째 교체 중에 교체 함수가 해당 사용자를 복제하여 두 번째 사용자를 생성합니다. 보안 암호가 교체될 때마다 교체 함수는 업데이트하는 사용자의 암호를 교체합니다. 대부분의 사용자는 본인을 복제할 수 있는 권한이 없으므로 다른 보안 암호에서 `superuser`에 대한 보안 인증을 제공해야 합니다. 데이터베이스의 복제된 사용자가 원래 사용자와 동일한 권한을 가지고 있지 않은 경우(일회성(임시) 또는 대화형 사용자라고도 함), 단일 사용자 교체 전략을 사용하는 것이 좋습니다.

이 전략은 첫 번째 역할이 데이터베이스 테이블을 소유하고 두 번째 역할이 데이터베이스 테이블에 액세스할 수 있는 권한이 있는 권한 모델을 포함하는 데이터베이스에 적합합니다. 이 전략은 고가용성이 필요한 애플리케이션에도 적합합니다. 애플리케이션은 교체 중에 보안 암호가 검색되더라도 유효한

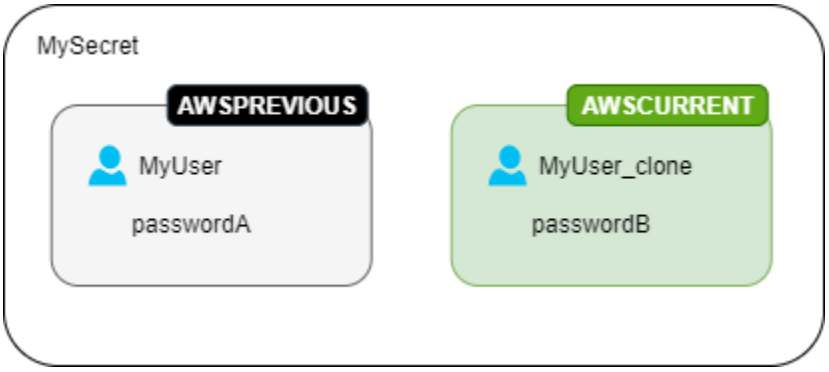
보안 인증 세트를 가져옵니다. 교체 후에는 `user` 및`user_clone` 보안 인증이 모두 유효합니다. 이러한 유형의 교체 중에 애플리케이션이 거부될 가능성은 단일 사용자 교체보다 훨씬 적습니다. 암호 변경 사항이 모든 서버로 전파되는 데 시간이 걸리는 서버 팜에서 데이터베이스가 호스트되는 경우 데이터베이스가 새 보안 인증을 사용하는 호출을 거부할 위험이 있습니다. 적절한 재시도 전략을 사용하여 이 위험을 완화할 수 있습니다.

Secrets Manager는 원래 사용자와 동일한 권한이 있는 복제된 사용자를 생성합니다. 클론이 생성된 후 원래 사용자의 권한을 변경하는 경우 복제된 사용자의 권한도 변경해야 합니다.

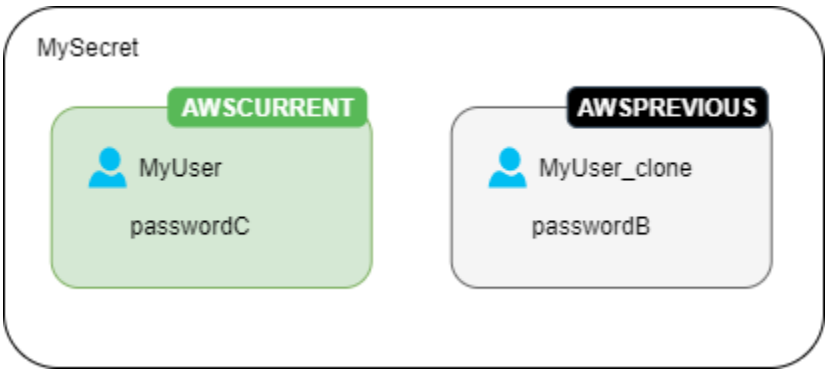
예를 들어 데이터베이스 사용자의 보안 인증으로 암호를 만드는 경우 암호에는 해당 보안 인증이 있는 버전 하나가 포함됩니다.



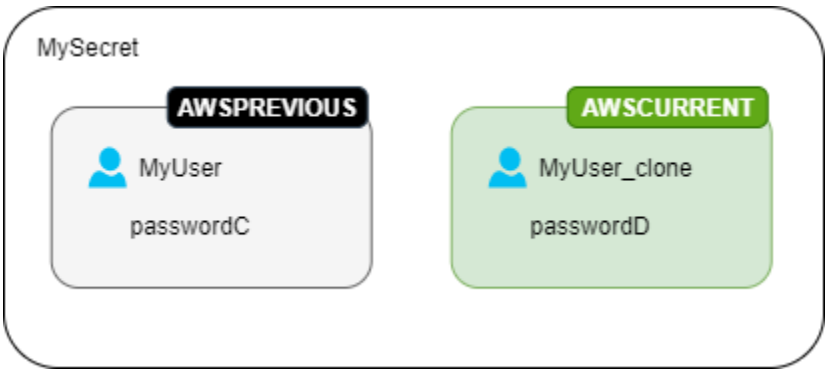
첫 번째 교체 – 교체 기능은 생성된 암호로 사용자의 클론을 생성하고 해당 보안 인증은 현재 암호 버전이 됩니다.



두 번째 교체 – 교체 기능은 원래 사용자의 암호를 업데이트합니다.



세 번째 교체 – 교체 기능은 복제된 사용자의 암호를 업데이트합니다.



Lambda 교체 함수

에서 AWS Lambda 함수the section called “Lambda 함수로 교체”는 보안 암호를 교체합니다. AWS Secrets Manager 는 스테이징 레이블을 사용하여 교체 중에 보안 암호 버전을 식별합니다.

AWS Secrets Manager 가 보안 암호 유형에 대한 교체 함수 템플릿을 제공하지 않는 경우 사용자 지정 교체 함수를 생성할 수 있습니다. 교체 함수를 작성할 때 다음 지침을 따르세요.

사용자 지정 교체 함수 모범 사례

- 일반 교체 템플릿을 시작점으로 사용하세요.
- 디버깅 또는 로깅 문에 주의하세요. 이 문이 Amazon CloudWatch Logs에 기록될 수 있으므로, 로그에 민감한 정보가 포함되지 않도록 하세요.

  로그 문 예시는 the section called “교체 함수 템플릿” 소스 코드를 참조하세요.
- 보안을 위해 Lambda 교체 함수만 보안 암호를 직접 교체하도록 AWS Secrets Manager 허용합니다. 교체 함수가 다른 Lambda 함수를 호출하여 보안 암호를 교체할 수 없습니다.
- 디버깅 안내를 원한다면 서버리스 애플리케이션 테스트 및 디버깅을 참조하세요.

- 외부 바이너리나 라이브러리를 사용하는 경우(예: 리소스 연결용) 패치 및 업데이트는 사용자가 책임져야 합니다.
- 교체 함수와 모든 종속 항목을 ZIP 파일(예: *my-function.zip*)로 패키징하세요.

> **Warning**
>
> 프로비저닝된 동시성 파라미터를 10보다 낮게 설정하면 Lambda 함수 실행 스레드가 부족하여 스로틀링이 발생할 수 있습니다. 자세한 내용은 AWS Lambda AWS Lambda 개발자 가이드의 예약된 동시성 및 프로비저닝된 동시성 이해를 참조하세요.

교체 함수의 4단계

주제

- createSecret: 새로운 버전의 보안 암호 생성
- setSecret: 데이터베이스 또는 서비스의 보안 인증 변경
- testSecret: 새 보안 암호 버전 테스트
- finishSecret: 교체 완료

createSecret: 새로운 버전의 보안 암호 생성

createSecret 메서드에서는 먼저 전달된 ClientRequestToken으로 get_secret_value를 호출하여 보안 암호가 존재하는지 확인합니다. 보안 암호가 없는 경우 create_secret 및 토큰을 VersionId로 사용하여 새 보안 암호를 생성합니다. 그러면 get_random_password를 사용하여 새 보안 암호 값이 생성됩니다. 그다음, put_secret_value를 호출하여 스테이징 레이블 AWSPENDING와 함께 저장합니다. 새 보안 암호 값을 AWSPENDING에 저장하면 멱등성을 보장하는 데 도움이 됩니다. 어떤 이유로든 교체에 실패할 경우 후속 호출에서 해당 보안 암호 값을 참조할 수 있습니다. 멱등성 Lambda 함수를 만들려면 어떻게 해야 합니까? 섹션을 참조하세요.

교체 함수를 직접 작성하기 위한 팁

- 새 보안 암호 값에 데이터베이스 또는 서비스에 유효한 문자만 포함되어 있는지 확인해야 합니다. ExcludeCharacters 파라미터를 사용하여 문자를 제외합니다.
- 함수를 테스트할 때 AWS CLI 를 사용하여 버전 단계를 확인합니다.를 호출describe-secret하고를 확인합니다VersionIdsToStages.

- Amazon RDS MySQL의 경우 대체 사용자 교체 시 Secrets Manager는 이름이 16자를 넘지 않는 클론 사용자를 생성합니다. 더 긴 사용자 이름을 허용하도록 교체 함수를 수정할 수 있습니다. MySQL 버전 5.7 이상에서는 최대 32자의 사용자 이름을 지원하지만 Secrets Manager는 사용자 이름 끝에 "_clone"(6자)을 추가하므로 사용자 이름을 최대 26자로 유지해야 합니다.

setSecret: 데이터베이스 또는 서비스의 보안 인증 변경

`setSecret` 메서드는 데이터베이스 또는 서비스의 자격 증명을 `AWSPENDING` 버전 보안 암호의 새 보안 암호 값과 일치하도록 변경합니다.

교체 함수를 직접 작성하기 위한 팁

- 데이터베이스처럼 문을 해석하는 서비스에 문을 전달할 경우 쿼리 파라미터화를 사용합니다. 자세한 내용은 OWASP 웹 사이트의 Query Parameterization Cheat Sheet를 참조하세요.
- 교체 함수는 Secrets Manager 보안 암호와 대상 리소스 모두에서 고객 보안 인증을 액세스하고 수정할 수 있는 권한을 가진 대리자입니다. 잠재적으로 혼동된 대리자 공격을 방지하려면 공격자가 함수를 사용하여 다른 리소스에 액세스할 수 없도록 해야 합니다. 보안 인증을 업데이트하기 전에:
  - `AWSCURRENT` 버전 보안 암호의 보안 인증이 유효한지 확인합니다. `AWSCURRENT` 보안 인증이 유효하지 않은 경우 교체 시도를 중단합니다.
  - `AWSCURRENT` 및 `AWSPENDING` 보안 암호 값이 동일한 리소스에 대한 값인지 확인합니다. 사용자 이름과 암호의 경우 `AWSCURRENT` 및 `AWSPENDING` 사용자 이름이 동일한지 확인합니다.
  - 대상 서비스 리소스가 동일한지 확인합니다. 데이터베이스의 경우 `AWSCURRENT` 및 `AWSPENDING` 호스트 이름이 동일한지 확인합니다.
- 드문 경우지만 데이터베이스에 대한 기존 교체 함수를 사용자 지정할 수 있습니다. 예를 들어 대체 사용자 교체가 있는 경우 Secrets Manager는 첫 번째 사용자의 런타임 구성 매개변수를 복사하여 복제된 사용자를 생성합니다. 더 많은 속성을 포함하거나 복제된 사용자에게 부여되는 속성을 변경하려면 `set_secret` 함수의 코드를 업데이트해야 합니다.

testSecret: 새 보안 암호 버전 테스트

다음으로 Lambda 교체 함수는 데이터베이스나 서비스에 액세스하는 데 보안 암호의 `AWSPENDING` 버전을 사용하여 해당 버전을 테스트합니다. 교체 함수 템플릿 기반 교체 함수는 읽기 액세스를 사용하여 새 보안 암호를 테스트합니다.

finishSecret: 교체 완료

마지막으로 Lambda 교체 함수는 AWSCURRENT 레이블을 이전 보안 암호 버전에서 이 버전으로 이동하며 동일한 API 직접 호출에서 AWSPENDING 레이블도 제거합니다. Secrets Manager에서 AWSPREVIOUS 스테이징 레이블을 이전 버전으로 추가하여 보안 암호의 마지막으로 확인된 정상 버전을 유지할 수 있습니다.

finish_secret 메서드는 update_secret_version_stage를 사용하여 AWSCURRENT 스테이징 레이블을 이전 보안 암호 버전에서 새 보안 암호 버전으로 이동합니다. Secrets Manager에서 AWSPREVIOUS 스테이징 레이블을 이전 버전에 자동으로 추가하여 보안 암호의 마지막으로 확인된 정상 버전을 유지할 수 있습니다.

교체 함수를 직접 작성하기 위한 팁

- 이 시점 이전에는 AWSPENDING 을 제거해서는 안 되며, 별도의 API 직접 호출을 사용하여 제거하면 교체가 성공적으로 완료되지 않았음을 Secrets Manager에 알릴 수 있으므로 별도의 API 직접 호출을 사용하여 제거해서도 안 됩니다. Secrets Manager에서 AWSPREVIOUS 스테이징 레이블을 이전 버전으로 추가하여 보안 암호의 마지막으로 확인된 정상 버전을 유지할 수 있습니다.

교체가 성공하면 AWSPENDING 스테이징 레이블은 AWSCURRENT 버전과 동일한 버전에 첨부되거나 어떤 버전에도 연결되지 않을 수 있습니다. AWSPENDING 스테이징 레이블이 있지만 AWSCURRENT와 동일한 버전에 연결되지 않은 경우 이후의 교체 호출은 이전 교체 요청이 아직 진행 중인 것으로 간주하여 오류를 반환합니다. 교체에 실패하면 AWSPENDING 스테이징 레이블은 빈 암호 버전에 연결될 수 있습니다. 자세한 내용은 교체 문제 해결 단원을 참조하십시오.

AWS Secrets Manager 교체 함수 템플릿

AWS Secrets Manager 는 다양한 데이터베이스 시스템 및 서비스에 대한 자격 증명의 보안 관리를 자동화하는 데 도움이 되는 교체 함수 템플릿 세트를 제공합니다. 이 템플릿들은 바로 사용 가능한 Lambda 함수로, 자격 증명 교체에 대한 모범 사례를 구현하여 수동 개입 없이 보안 상태를 유지할 수 있도록 도와줍니다.

이 템플릿들은 두 가지 주요 교체 전략을 지원합니다.

- 단일 사용자 교체: 단일 사용자의 자격 증명을 업데이트합니다.
- 교대 사용자 교체: 두 개의 별도 사용자를 유지하여 자격 증명 변경 시 가동 중지 시간을 최소화합니다.

또한 Secrets Manager는 모든 유형의 보안 암호를 위한 출발점으로 사용할 수 있는 일반 템플릿도 제공합니다.

템플릿을 사용하려면 다음 섹션을 참조하세요.

- 데이터베이스 보안 암호 자동 교체(콘솔)
- 비데이터베이스 보안 암호 자동 교체(콘솔)

자체 교체 함수를 작성하려면 교체 함수 쓰기를 참조하세요.

템플릿

- Amazon RDS 및 Amazon Aurora
  - Amazon RDS Db2 단일 사용자
  - Amazon RDS Db2 대체 사용자
  - Amazon RDS MariaDB 단일 사용자
  - Amazon RDS MariaDB 대체 사용자
  - Amazon RDS 및 Amazon Aurora MySQL 단일 사용자
  - Amazon RDS 및 Amazon Aurora MySQL 대체 사용자
  - Amazon RDS Oracle 단일 사용자
  - Amazon RDS Oracle 대체 사용자
  - Amazon RDS 및 Amazon Aurora PostgreSQL 단일 사용자
  - Amazon RDS 및 Amazon Aurora PostgreSQL 대체 사용자
  - Amazon RDS Microsoft SQLServer 단일 사용자
  - Amazon RDS Microsoft SQLServer 대체 사용자
- Amazon DocumentDB(MongoDB 호환)
  - Amazon DocumentDB 단일 사용자
  - Amazon DocumentDB 대체 사용자
- Amazon Redshift
  - Amazon Redshift 단일 사용자
  - Amazon Redshift 대체 사용자
- Amazon Timestream for InfluxDB
  - Amazon Timestream for InfluxDB 단일 사용자
  - Amazon Timestream for InfluxDB 대체 사용자

- Amazon ElastiCache
- Active Directory
  - Active Directory 자격 증명
  - Active Directory keytab
- 다른 유형의 보안 암호

Amazon RDS 및 Amazon Aurora

Amazon RDS Db2 단일 사용자

- 템플릿 이름: SecretsManagerRDSDb2RotationSingleUser
- 교체 전략: 교체 전략: 단일 사용자.
- **`SecretString`** 구조: the section called “Amazon RDS 및 Aurora 자격 증명”.
- 소스 코드: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSDb2RotationSingleUser/lambda_function.py
- 종속성: python-ibmdb

Amazon RDS Db2 대체 사용자

- 템플릿 이름: SecretsManagerRDSDb2RotationMultiUser
- 교체 전략: the section called “대체 사용자”.
- **`SecretString`** 구조: the section called “Amazon RDS 및 Aurora 자격 증명”.
- 소스 코드: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSDb2RotationMultiUser/lambda_function.py
- 종속성: python-ibmdb

Amazon RDS MariaDB 단일 사용자

- 템플릿 이름: SecretsManagerRDSMariaDBRotationSingleUser
- 교체 전략: 교체 전략: 단일 사용자.
- **`SecretString`** 구조: the section called “Amazon RDS 및 Aurora 자격 증명”.
- 소스 코드: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSMariaDBRotationSingleUser/lambda_function.py

- 종속성: PyMySQL 1.0.2. 인증에 sha256 암호를 사용할 경우에는 PyMySQL[rsa]입니다. Lambda 런타임에서 컴파일된 코드가 있는 패키지를 사용하는 방법에 대한 자세한 내용은 AWS 지식 센터의 컴파일된 바이너리가 포함된 Python 패키지를 배포 패키지에 추가하고 해당 패키지가 Lambda와 호환되도록 하려면 어떻게 해야 하나요?를 참조하세요.

Amazon RDS MariaDB 대체 사용자

- 템플릿 이름: SecretsManagerRDSMariaDBRotationMultiUser
- 교체 전략: the section called “대체 사용자”.
- **`SecretString`** 구조: the section called “Amazon RDS 및 Aurora 자격 증명”.
- 소스 코드: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSMariaDBRotationMultiUser/lambda_function.py
- 종속성: PyMySQL 1.0.2. 인증에 sha256 암호를 사용할 경우에는 PyMySQL[rsa]입니다. Lambda 런타임에서 컴파일된 코드가 있는 패키지를 사용하는 방법에 대한 자세한 내용은 AWS 지식 센터의 컴파일된 바이너리가 포함된 Python 패키지를 배포 패키지에 추가하고 해당 패키지가 Lambda와 호환되도록 하려면 어떻게 해야 하나요?를 참조하세요.

Amazon RDS 및 Amazon Aurora MySQL 단일 사용자

- 템플릿 이름: SecretsManagerRDSMySQLRotationSingleUser
- 교체 전략: the section called “단일 사용자”.
- 예상 **`SecretString`** 구조: the section called “Amazon RDS 및 Aurora 자격 증명”.
- 소스 코드: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSMySQLRotationSingleUser/lambda_function.py
- 종속성: PyMySQL 1.0.2. 인증에 sha256 암호를 사용할 경우에는 PyMySQL[rsa]입니다. Lambda 런타임에서 컴파일된 코드가 있는 패키지를 사용하는 방법에 대한 자세한 내용은 AWS 지식 센터의 컴파일된 바이너리가 포함된 Python 패키지를 배포 패키지에 추가하고 해당 패키지가 Lambda와 호환되도록 하려면 어떻게 해야 하나요?를 참조하세요.

Amazon RDS 및 Amazon Aurora MySQL 대체 사용자

- 템플릿 이름: SecretsManagerRDSMySQLRotationMultiUser
- 교체 전략: the section called “대체 사용자”.
- 예상 **`SecretString`** 구조: the section called “Amazon RDS 및 Aurora 자격 증명”.

- 소스 코드: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSMySQLRotationMultiUser/lambda_function.py
- 종속성: PyMySQL 1.0.2. 인증에 sha256 암호를 사용할 경우에는 PyMySQL[rsa]입니다. Lambda 런타임에서 컴파일된 코드가 있는 패키지를 사용하는 방법에 대한 자세한 내용은 AWS 지식 센터의 컴파일된 바이너리가 포함된 Python 패키지를 배포 패키지에 추가하고 해당 패키지가 Lambda와 호환되도록 하려면 어떻게 해야 하나요?를 참조하세요.

Amazon RDS Oracle 단일 사용자

- 템플릿 이름: SecretsManagerRDSOracleRotationSingleUser
- 교체 전략: the section called “단일 사용자”.
- 예상 **`SecretString`** 구조: the section called “Amazon RDS 및 Aurora 자격 증명”.
- 소스 코드: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSOracleRotationSingleUser/lambda_function.py
- 종속성: python-oracledb 2.4.1

Amazon RDS Oracle 대체 사용자

- 템플릿 이름: SecretsManagerRDSOracleRotationMultiUser
- 교체 전략: the section called “대체 사용자”.
- 예상 **`SecretString`** 구조: the section called “Amazon RDS 및 Aurora 자격 증명”.
- 소스 코드: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSOracleRotationMultiUser/lambda_function.py
- 종속성: python-oracledb 2.4.1

Amazon RDS 및 Amazon Aurora PostgreSQL 단일 사용자

- 템플릿 이름: SecretsManagerRDSPostgreSQLRotationSingleUser
- 교체 전략: 교체 전략: 단일 사용자.
- 예상 **`SecretString`** 구조: the section called “Amazon RDS 및 Aurora 자격 증명”.
- 소스 코드: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSPostgreSQLRotationSingleUser/lambda_function.py
- 종속성: PyGreSQL 5.2.5

Amazon RDS 및 Amazon Aurora PostgreSQL 대체 사용자

- 템플릿 이름: SecretsManagerRDSPostgreSQLRotationMultiUser
- 교체 전략: the section called “대체 사용자”.
- 예상 **`SecretString`** 구조: the section called “Amazon RDS 및 Aurora 자격 증명”.
- 소스 코드: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSPostgreSQLRotationMultiUser/lambda_function.py
- 종속성: PyGreSQL 5.2.5

Amazon RDS Microsoft SQLServer 단일 사용자

- 템플릿 이름: SecretsManagerRDSSQLServerRotationSingleUser
- 교체 전략: the section called “단일 사용자”.
- 예상 **`SecretString`** 구조: the section called “Amazon RDS 및 Aurora 자격 증명”.
- 소스 코드: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSSQLServerRotationSingleUser/lambda_function.py
- 종속성: Pymssql 2.2.2

Amazon RDS Microsoft SQLServer 대체 사용자

- 템플릿 이름: SecretsManagerRDSSQLServerRotationMultiUser
- 교체 전략: the section called “대체 사용자”.
- 예상 **`SecretString`** 구조: the section called “Amazon RDS 및 Aurora 자격 증명”.
- 소스 코드: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSSQLServerRotationMultiUser/lambda_function.py
- 종속성: Pymssql 2.2.2

Amazon DocumentDB(MongoDB 호환)

Amazon DocumentDB 단일 사용자

- 템플릿 이름: SecretsManagerMongoDBRotationSingleUser
- 교체 전략: the section called “단일 사용자”.

- 예상 **`SecretString`** 구조: the section called “Amazon DocumentDB 자격 증명”.
- 소스 코드: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerMongoDBRotationSingleUser/lambda_function.py
- 종속성: PyMongo 4.2.0

Amazon DocumentDB 대체 사용자

- 템플릿 이름: SecretsManagerMongoDBRotationMultiUser
- 교체 전략: the section called “대체 사용자”.
- 예상 **`SecretString`** 구조: the section called “Amazon DocumentDB 자격 증명”.
- 소스 코드: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerMongoDBRotationMultiUser/lambda_function.py
- 종속성: PyMongo 4.2.0

Amazon Redshift

Amazon Redshift 단일 사용자

- 템플릿 이름: SecretsManagerRedshiftRotationSingleUser
- 교체 전략: the section called “단일 사용자”.
- 예상 **`SecretString`** 구조: the section called “Amazon Redshift 자격 증명”.
- 소스 코드: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRedshiftRotationSingleUser/lambda_function.py
- 종속성: PyGreSQL 5.2.5

Amazon Redshift 대체 사용자

- 템플릿 이름: SecretsManagerRedshiftRotationMultiUser
- 교체 전략: the section called “대체 사용자”.
- 예상 **`SecretString`** 구조: the section called “Amazon Redshift 자격 증명”.
- 소스 코드: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRedshiftRotationMultiUser/lambda_function.py
- 종속성: PyGreSQL 5.2.5

Amazon Timestream for InfluxDB

이러한 템플릿을 사용하려면 Amazon Timestream 개발자 가이드의 How Amazon Timestream for InfluxDB uses secrets를 참조하세요.

Amazon Timestream for InfluxDB 단일 사용자

- 템플릿 이름: SecretsManagerInfluxDBRotationSingleUser
- 예상 **`SecretString`** 구조: the section called “Amazon Timestream for InfluxDB 보안 암호 구조”.
- 소스 코드: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerInfluxDBRotationSingleUser/lambda_function.py
- 종속성: InfluxDB 2.0 python 클라이언트

Amazon Timestream for InfluxDB 대체 사용자

- 템플릿 이름: SecretsManagerInfluxDBRotationMultiUser
- 예상 **`SecretString`** 구조: the section called “Amazon Timestream for InfluxDB 보안 암호 구조”.
- 소스 코드: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerInfluxDBRotationMultiUser/lambda_function.py
- 종속성: InfluxDB 2.0 python 클라이언트

Amazon ElastiCache

이 템플릿을 사용하려면 Amazon ElastiCache 사용 설명서의 사용자를 위한 자동 암호 교체를 참조하세요.

- 템플릿 이름: SecretsManagerElasticacheUserRotation
- 예상 **`SecretString`** 구조: the section called “Amazon ElastiCache 자격 증명”.
- 소스 코드: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerElasticacheUserRotation/lambda_function.py

Active Directory

Active Directory 자격 증명

- 템플릿 이름: SecretsManagerActiveDirectoryRotationSingleUser

- 예상 **`SecretString`** 구조: the section called “Active Directory 자격 증명”.
- 소스 코드: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerActiveDirectoryRotationSingleUser/lambda_function.py

Active Directory keytab

- 템플릿 이름: SecretsManagerActiveDirectoryAndKeytabRotationSingleUser
- 예상 **`SecretString`** 구조: the section called “Active Directory 자격 증명”.
- 소스 코드: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerActiveDirectoryAndKeytabRotationSingleUser/lambda_function.py
- 종속성: msktutil

다른 유형의 보안 암호

Secrets Manager은 이 템플릿을 모든 유형의 보안 암호에 대한 교체 함수를 생성할 수 있는 시작점으로 제공합니다.

- 템플릿 이름: SecretsManagerRotationTemplate
- 소스 코드: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRotationTemplate/lambda_function.py

에 대한 Lambda 교체 함수 실행 역할 권한 AWS Secrets Manager

the section called “Lambda 함수로 교체”의 경우, Secrets Manager가 Lambda 함수를 사용하여 보안 암호를 교체하면 Lambda는 IAM 실행 역할을 담당하고 해당 자격 증명을 Lambda 함수 코드에 제공합니다. 자동 교체를 설정하는 방법에 대한 지침은 다음 섹션을 참조하세요.

- 데이터베이스 보안 암호 자동 교체(콘솔)
- 비데이터베이스 보안 암호 자동 교체(콘솔)
- 자동 교체(AWS CLI)

다음 예제에서는 Lambda 교체 함수 실행 역할에 대한 인라인 정책을 보여 줍니다. 실행 역할을 생성하고 권한 정책을 연결하려면 AWS Lambda 실행 역할을 참조하세요.

예시:

- Lambda 교체 함수 실행 역할에 대한 정책
- 고객 관리형 키에 대한 정책 설명
- 대체 사용자 전략에 대한 정책 설명

Lambda 교체 함수 실행 역할에 대한 정책

다음 예제 정책에서는 교체 함수가 다음 작업을 수행할 수 있도록 허용합니다.

- *SecretARN*에 대한 Secrets Manager 작업을 실행합니다.
- 새 암호를 생성합니다.
- 데이터베이스 또는 서비스가 VPC에서 실행되는 경우 필수 구성을 설정하세요. VPC에서 리소스에 액세스하도록 Lambda 함수 구성을 참조하세요.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "secretsmanager:UpdateSecretVersionStage"
            ],
            "Resource": "arn:aws:secretsmanager:us-
east-1:123456789012:secret:secretName-AbCdEf"
    },
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetRandomPassword"
            ],
            "Resource": "*"
        },
        {
            "Action": [
                "ec2:CreateNetworkInterface",
```

```
                "ec2:DeleteNetworkInterface",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DetachNetworkInterface"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

고객 관리형 키에 대한 정책 설명

AWS 관리형 키 aws/secretsmanager 이외의 KMS 키를 사용하여 보안 암호를 암호화할 경우 Lambda 실행 역할에 키 사용 권한을 부여해야 합니다. SecretARN 암호화 컨텍스트를 사용하여 암호 해독 기능의 사용을 제한할 수 있으므로 교체 함수 역할은 교체를 담당하는 암호의 암호 해독에만 액세스할 수 있습니다. 다음 예시에서는 함수가 KMS 키를 이용하여 비밀을 해독할 수 있도록 실행 역할 정책에 추가할 선언문을 보여 줍니다.

```
    {
        "Effect": "Allow",
        "Action": [
            "kms:Decrypt",
            "kms:DescribeKey",
            "kms:GenerateDataKey"
        ],
        "Resource": "KMSKeyARN",
        "Condition": {
            "StringEquals": {
                "kms:EncryptionContext:SecretARN": "SecretARN"
            }
        }
    }
```

고객 관리형 키로 암호화된 여러 암호에 교체 기능을 사용하려면 다음 예와 같은 명령문을 추가하여 실행 역할이 암호를 해독하도록 허용하세요.

```
    {
        "Effect": "Allow",
        "Action": [
            "kms:Decrypt",
            "kms:DescribeKey",
```

```
            "kms:GenerateDataKey"
        ],
        "Resource": "KMSKeyARN",
        "Condition": {
            "StringEquals": {
                "kms:EncryptionContext:SecretARN": [
                    "arn1",
                    "arn2"
                ]
            }
        }
    }
```

대체 사용자 전략에 대한 정책 설명

대체 사용자 교체 전략에 대한 자세한 내용은 the section called “Lambda 함수 교체 전략” 섹션을 참조하세요.

Amazon RDS 보안 인증 정보가 포함된 암호의 경우, 대체 사용자 전략을 사용하고 Amazon RDS에서 슈퍼 사용자 암호를 관리한다면 교체 함수가 Amazon RDS에서 읽기 전용 API를 호출하도록 허용해야 데이터베이스의 연결 정보를 가져올 수도 있습니다. AWS 관리형 정책 AmazonRDSReadOnlyAccess를 연결하는 것이 좋습니다.

다음 예제 정책에서는 함수가 다음 작업을 수행할 수 있도록 허용합니다.

- *SecretARN*에 대한 Secrets Manager 작업을 실행합니다.
- 슈퍼유저 보안 암호에서 보안 인증을 검색합니다. 슈퍼유저 보안 암호의 보안 인증은 Secrets Manager가 교체된 보안 암호의 보안 인증을 업데이트하는 데 사용됩니다.
- 새 암호를 생성합니다.
- 데이터베이스 또는 서비스가 VPC에서 실행되는 경우 필수 구성을 설정하세요. 자세한 내용은 VPC의 리소스에 액세스하도록 Lambda 함수 구성(Configuring a Lambda function to access resources in a VPC)을 참조하세요.

JSON

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

```
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "secretsmanager:UpdateSecretVersionStage"
            ],
            "Resource": "arn:aws:secretsmanager:us-
east-1:123456789012:secret:secretName-AbCdEf"
    },
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource": "arn:aws:secretsmanager:us-
east-1:123456789012:secret:secretName-AbCdEf"
    },
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetRandomPassword"
            ],
            "Resource": "*"
        },
        {
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterface",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DetachNetworkInterface"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

AWS Lambda 교체 함수에 대한 네트워크 액세스

the section called “Lambda 함수로 교체”의 경우, Secrets Manager가 Lambda 함수를 사용하여 보안 암호를 교체할 때 Lambda 교체 함수가 보안 암호에 액세스할 수 있어야 합니다. 보안 암호에 보안 인

증이 포함된 경우 Lambda 함수가 해당 보안 인증의 소스(예: 데이터베이스 또는 서비스)에도 액세스할 수 있어야 합니다.

보안 암호에 액세스하려면

Lambda 교체 함수는 Secrets Manager 엔드포인트에 액세스할 수 있어야 합니다. Lambda 함수가 인터넷에 액세스할 수 있는 경우 퍼블릭 엔드포인트를 사용할 수 있습니다. 엔드포인트를 찾으려면 the section called “Secrets Manager 엔드포인트” 섹션을 참조하세요.

Lambda 함수가 인터넷에 액세스할 수 없는 VPC에서 실행되는 경우 VPC 내에서 Secrets Manager 서비스 프라이빗 엔드포인트를 구성하는 것이 좋습니다. 그러면 VPC가 퍼블릭 리전 엔드포인트로 전달된 요청을 가로채서 프라이빗 엔드포인트로 리디렉션할 수 있습니다. 자세한 내용은 VPC 엔드포인트(AWS PrivateLink) 단원을 참조하십시오.

또는 VPC에 NAT 게이트웨이 또는 인터넷 게이트웨이를 추가하여 Lambda 함수를 통해 퍼블릭 Secrets Manager 엔드포인트에 액세스할 수 있도록 합니다. 그러면 VPC에서 트래픽이 퍼블릭 엔드포인트에 도달할 수 있습니다. 이렇게 하면 게이트웨이에 대한 IP 주소가 퍼블릭 인터넷에서 공격을 받을 수 있으므로 VPC가 더 많은 위험에 노출됩니다.

(선택 사항) 데이터베이스 또는 서비스에 액세스하려면

보안 암호(예: API 키)의 경우 보안 암호와 함께 업데이트해야 하는 소스 데이터베이스 또는 서비스가 없습니다.

데이터베이스 또는 인스턴스가 VPC의 Amazon EC2 인스턴스에서 실행 중인 경우, 동일한 VPC에서 Lambda 함수가 실행되도록 구성하는 것이 좋습니다. 그러면 교체 함수가 서비스와 직접 통신할 수 있습니다. 자세한 내용은 VPC 액세스 구성을 참조하세요.

Lambda 함수가 데이터베이스 또는 서비스에 액세스하도록 허용하려면 Lambda 교체 함수에 연결된 보안 그룹이 데이터베이스 또는 서비스에 아웃바운드 연결을 허용하는지 확인해야 합니다. 데이터베이스 또는 서비스에 연결된 보안 그룹이 Lambda 교체 함수에 인바운드 연결을 허용하는지도 확인해야 합니다.

AWS Secrets Manager 교체 문제 해결

많은 서비스에서 Secrets Manager는 Lambda 함수를 사용하여 보안 암호를 교체합니다. 자세한 내용은 the section called “Lambda 함수로 교체” 단원을 참조하십시오. Lambda 교체 함수는 Secrets Manager뿐만 아니라 보안 암호가 사용되는 데이터베이스 또는 서비스와도 상호 작용합니다. 교체가 예상대로 작동하지 않을 경우 먼저 CloudWatch 로그를 확인해야 합니다.

> **Note**
>
> 자동 교체 관리를 비롯한 일부 서비스는 보안 암호를 대신 관리할 수 있습니다. 자세한 내용은 the section called “관리형 교체” 단원을 참조하십시오.

주제

- AWS Lambda 함수의 보안 암호 교체 실패 문제를 해결하는 방법
- “환경 변수에서 보안 인증을 찾았습니다.” 이후 활동 없음
- “createSecret” 이후 활동 없음
- 오류: “KMS에 대한 액세스가 허용되지 않습니다.”
- 오류: “보안 암호 JSON에 키가 없습니다.”
- 오류: “setSecret: 데이터베이스에 로그인할 수 없음”
- 오류: “모듈 'lambda_function'을 가져올 수 없음”
- 기존 교체 함수를 Python 3.7에서 3.9로 업그레이드
- 기존 교체 함수를 Python 3.9에서 3.10으로 업그레이드
- AWS Lambda 를 사용한 보안 암호 교체 PutSecretValue 실패
- 오류: "<a rotation> 단계에서 Lambda <arn>을 실행할 때 오류"

AWS Lambda 함수의 보안 암호 교체 실패 문제를 해결하는 방법

Lambda 함수에서 보안 암호 교체 실패가 발생하면 다음 단계에 따라 문제를 해결할 수 있습니다.

가능한 원인

- Lambda 함수의 동시 실행 수 부족
- 교체 중 여러 API 호출로 인한 경쟁 조건
- Lambda 함수의 잘못된 로직
- Lambda 함수와 데이터베이스 간의 네트워킹 문제

일반적인 문제 해결 단계

1. CloudWatch 로그 분석:
   - Lambda 함수 로그에서 구체적인 오류 메시지나 예상치 못한 동작을 확인합니다.

- 모든 교체 단계(CreateSecret, SetSecret, TestSecret, FinishSecret)가 시도되고 있는지 확인합니다.

2. 교체 중 API 호출 검토:
   - Lambda 교체 중 보안 암호에 대한 변경 API 호출을 피합니다.
   - RotateSecret와 PutSecretValue 호출 간 경쟁 조건이 없는지 확인합니다.
3. Lambda 함수 로직 확인:
   - 보안 암호 교체를 위해 최신 AWS 샘플 코드를 사용하고 있는지 확인합니다.
   - 사용자 지정 코드를 사용하는 경우 모든 교체 단계가 올바르게 처리되는지 검토합니다.
4. 네트워크 구성 확인:
   - Lambda 함수가 데이터베이스에 액세스할 수 있도록 보안 그룹 규칙을 확인합니다.
   - Secrets Manager에 대한 VPC 엔드포인트 또는 퍼블릭 엔드포인트 액세스가 올바른지 확인합니다.
5. 보안 암호 버전 테스트:
   - 보안 암호의 AWSCURRENT 버전이 데이터베이스 액세스를 허용하는지 확인합니다.
   - AWSPREVIOUS 또는 AWSPENDING 버전이 유효한지 확인합니다.
6. 보류 중인 교체 지우기:
   - 교체가 계속 실패하면 AWSPENDING 스테이징 레이블을 제거하고 교체를 재시도합니다.
7. Lambda 동시성 설정 확인:
   - 워크로드에 맞는 동시성 설정인지 확인합니다.
   - 동시성 관련 문제가 의심되면 "동시성 관련 교체 실패 문제 해결" 섹션을 참조하세요.

“환경 변수에서 보안 인증을 찾았습니다.” 이후 활동 없음

“환경 변수에서 보안 인증을 찾았습니다.” 이후에 활동이 없고 작업이 오래 실행되는 경우(예: 기본 Lambda 제한 시간 30000ms), Secrets Manager 엔드포인트에 연결하는 동안 Lambda 함수가 시간 초과될 수 있습니다.

Lambda 교체 함수는 Secrets Manager 엔드포인트에 액세스할 수 있어야 합니다. Lambda 함수가 인터넷에 액세스할 수 있는 경우 퍼블릭 엔드포인트를 사용할 수 있습니다. 엔드포인트를 찾으려면 the section called “Secrets Manager 엔드포인트” 섹션을 참조하세요.

Lambda 함수가 인터넷에 액세스할 수 없는 VPC에서 실행되는 경우 VPC 내에서 Secrets Manager 서비스 프라이빗 엔드포인트를 구성하는 것이 좋습니다. 그러면 VPC가 퍼블릭 리전 엔드포인트로 전달된 요청을 가로채서 프라이빗 엔드포인트로 리디렉션할 수 있습니다. 자세한 내용은 VPC 엔드포인트(AWS PrivateLink) 단원을 참조하십시오.

또는 VPC에 NAT 게이트웨이 또는 인터넷 게이트웨이를 추가하여 Lambda 함수를 통해 퍼블릭 Secrets Manager 엔드포인트에 액세스할 수 있도록 합니다. 그러면 VPC에서 트래픽이 퍼블릭 엔드포인트에 도달할 수 있습니다. 이렇게 하면 게이트웨이에 대한 IP 주소가 퍼블릭 인터넷에서 공격을 받을 수 있으므로 VPC가 더 많은 위험에 노출됩니다.

“createSecret” 이후 활동 없음

다음은 createSecret 이후에 교체가 중지될 수 있는 문제입니다.

VPC 네트워크 ACL은 HTTPS 트래픽 송수신을 허용하지 않습니다.

> 자세한 내용은 Amazon VPC 사용 설명서의 네트워크 ACL을 사용하여 서브넷에 대한 트래픽 제어를 참조하세요.

Lambda 함수 제한 시간 구성이 너무 짧아서 작업을 수행할 수 없습니다.

> 자세한 내용은 AWS Lambda 개발자 가이드의 Lambda 함수 옵션 구성을 참조하세요.

Secrets Manager VPC 엔드포인트는 할당된 보안 그룹에서 VPC CIDR 수신을 허용하지 않습니다.

> 자세한 내용은 Amazon VPC 사용 설명서의 보안 그룹을 사용하여 리소스에 대한 트래픽 제어를 참조하세요.

Secrets Manager VPC 엔드포인트 정책에서는 Lambda에서 VPC 엔드포인트를 사용하는 것을 허용하지 않습니다.

> 자세한 내용은 the section called “VPC 엔드포인트(AWS PrivateLink)” 단원을 참조하십시오.

암호는 교대로 사용자를 교체하여 사용하고, 슈퍼 사용자 암호는 Amazon RDS에서 관리하며, Lambda 함수는 RDS API에 액세스할 수 없습니다.

> 슈퍼 사용자 보안 암호를 다른 AWS 서비스에서 관리하는 대체 사용자 교체의 경우, Lambda 교체 함수가 서비스 엔드포인트를 호출하여 데이터베이스 연결 정보를 가져올 수 있어야 합니다. 데이터베이스 서비스의 VPC 엔드포인트를 구성하는 것이 좋습니다. 자세한 내용은 다음 섹션을 참조하세요.
>
> - Amazon RDS 사용 설명서의 Amazon RDS API 및 인터페이스 VPC 엔드포인트.
> - Amazon Redshift 관리 가이드의 VPC 엔드포인트 작업.

오류: “KMS에 대한 액세스가 허용되지 않습니다.”

`ClientError: An error occurred (AccessDeniedException) when calling the GetSecretValue operation: Access to KMS is not allowed`에서 보듯 교체 함수에는 암호를 암호화하는 데 사용된 KMS 키를 사용하여 암호를 해독할 권한이 없습니다. 권한 정책에 암호화 컨텍스트를 특정 보안 암호로 제한하는 조건이 있을 수도 있습니다. 필요한 권한에 대한 자세한 내용은 the section called “고객 관리형 키에 대한 정책 설명” 섹션을 참조하세요.

오류: “보안 암호 JSON에 키가 없습니다.”

Lambda 교체 함수를 사용하려면 보안 암호 값이 특정 JSON 구조에 있어야 합니다. 이 오류가 표시되면 JSON에 교체 함수에서 액세스하려는 키가 없을 수 있습니다. 각 보안 암호 유형별 JSON 구조에 대한 자세한 내용은 the section called “보안 암호의 JSON 구조” 섹션을 참조하세요.

오류: “setSecret: 데이터베이스에 로그인할 수 없음”

이 오류를 유발할 수 있는 문제는 다음과 같습니다.

교체 함수가 데이터베이스에 액세스할 수 없습니다.

- 작업 기간이 긴 경우(예: 5000ms 이상) Lambda 교체 함수가 네트워크를 통해 데이터베이스에 액세스하지 못할 수 있습니다.
- 데이터베이스 또는 인스턴스가 VPC의 Amazon EC2 인스턴스에서 실행 중인 경우, 동일한 VPC에서 Lambda 함수가 실행되도록 구성하는 것이 좋습니다. 그러면 교체 함수가 서비스와 직접 통신할 수 있습니다. 자세한 내용은 VPC 액세스 구성을 참조하세요.
- Lambda 함수가 데이터베이스 또는 서비스에 액세스하도록 허용하려면 Lambda 교체 함수에 연결된 보안 그룹이 데이터베이스 또는 서비스에 아웃바운드 연결을 허용하는지 확인해야 합니다. 데이터베이스 또는 서비스에 연결된 보안 그룹이 Lambda 교체 함수에 인바운드 연결을 허용하는지도 확인해야 합니다.

보안 암호의 보안 인증이 올바르지 않습니다.

- 작업 기간이 짧은 경우 Lambda 교체 함수가 보안 암호의 보안 인증으로 인증하지 못할 수 있습니다. AWS CLI 명령을 사용하여 보안 암호의 `AWSCURRENT` 및 `AWSPREVIOUS` 버전에 있는 정보로 수동으로 로그인하여 자격 증명을 확인합니다`get-secret-value`.

데이터베이스는 **`scram-sha-256`**을(를) 사용하여 암호를 암호화합니다.

- 데이터베이스가 Aurora PostgreSQL 버전 13 이상이고 `scram-sha-256`을(를) 사용하여 암호를 암호화하지만, 교체 함수가 `scram-sha-256`을(를) 지원하지 않는 `libpq` 버전 9 이하를 사용하는 경우, 교체 함수가 데이터베이스에 연결할 수 없습니다.

scram-sha-256 암호화를 사용하는 데이터베이스 사용자 확인 방법

- PostgreSQL 13을 위한 RDS의 SCRAM 인증 블로그에서 비-SCRAM 암호를 사용하는 사용자의 확인을 참조하세요.

해당 교체 함수가 사용하는 **libpq**의 버전 확인 방법

1. Linux 기반 컴퓨터의 Lambda 콘솔에서 교체 함수로 이동하여 배포 번들을 다운로드합니다. 작업 디렉터리에 zip 파일을 압축 해제합니다.
2. 명령줄의 작업 디렉터리에서 다음을 실행합니다:

   ```
   readelf -a libpq.so.5 | grep RUNPATH
   ```

3. 문자열 *PostgreSQL-9.4.x*이 표시되거나 메이저 버전이 10 미만인 경우 교체 함수가 scram-sha-256을(를) 지원하지 않는 것입니다.

   - scram-sha-256을(를) 지원하지 않는 교체 함수의 출력:

     ```
     0x000000000000001d (RUNPATH) Library runpath: [/
     local/p4clients/pkgbuild-a1b2c/workspace/build/
     PostgreSQL/PostgreSQL-9.4.x_client_only.123456.0/AL2_x86_64/
     DEV.STD.PTHREAD/build/private/tmp/brazil-path/build.libfarm/lib:/
     local/p4clients/pkgbuild-a1b2c/workspace/src/PostgreSQL/build/
     private/install/lib]
     ```

   - scram-sha-256을(를) 지원하는 교체 함수의 출력:

     ```
     0x000000000000001d (RUNPATH) Library runpath: [/
     local/p4clients/pkgbuild-a1b2c/workspace/build/
     PostgreSQL/PostgreSQL-10.x_client_only.123456.0/AL2_x86_64/
     DEV.STD.PTHREAD/build/private/tmp/brazil-path/build.libfarm/lib:/
     local/p4clients/pkgbuild-a1b2c/workspace/src/PostgreSQL/build/
     private/install/lib]
     ```

   - scram-sha-256을(를) 지원하는 교체 함수의 출력:

     ```
     0x000000000000001d (RUNPATH) Library runpath: [/local/
     p4clients/pkgbuild- a1b2c /workspace/build/PostgreSQL/
     PostgreSQL-14.x_client_only. 123456 .0/AL2_x86_64/
     DEV.STD.PTHREAD/build/private/tmp/brazil-path/build.libfarm/lib:/
     ```

```
local/p4clients/pkgbuild- a1b2c /workspace/src/PostgreSQL/build/
private/install/lib]
```

- scram-sha-256을(를) 지원하는 교체 함수의 출력:

```
0x000000000000001d (RUNPATH) Library runpath: [/local/p4clients/
pkgbuild- a1b2c/workspace/build/PostgreSQL/PostgreSQL-
14.x_client_only.123456.0/AL2_x86_64/DEV.STD.PTHREAD/build/
private/tmp/brazil- path/build.libfarm/lib:/local/p4clients/
pkgbuild- a1b2c/workspace/src/PostgreSQL/build/private/install/
lib]
```

> **Note**
>
> 2021년 12월 30일 이전에 자동 보안 암호 교체를 설정한 경우, 교체 함수에는 scram-sha-256을 지원하지 않는 이전 버전의 libpq가 포함되어 있습니다. scram-sha-256을 지원하려면 교체 함수를 다시 생성해야 합니다.

데이터베이스에 SSL/TLS 액세스가 필요합니다.

데이터베이스에 SSL/TLS 연결이 필요하지만 교체 함수가 암호화되지 않은 연결을 사용하는 경우, 교체 함수가 데이터베이스에 연결할 수 없습니다. Amazon RDS(Oracle 및 Db2 제외) 및 Amazon DocumentDB에 대한 교체 함수는 사용 가능한 보안 소켓 계층(SSL) 또는 전송 계층 보안(TLS)을 사용하여 데이터베이스에 연결합니다. 그렇지 않으면 암호화되지 않은 연결을 사용합니다.

> **Note**
>
> 2021년 12월 20일 전에 자동 보안 암호 교체을 설정한 경우, 교체 함수는 SSL/TLS를 지원하지 않는 이전 템플릿을 기반으로 할 수 있습니다. SSL/TLS를 사용하는 연결을 지원하려면 교체 함수를 재생성해야 합니다.

교체 함수가 언제 생성되었는지 확인하려면

1. Secrets Manager 콘솔(https://console.aws.amazon.com/secretsmanager/)에서 보안 암호를 엽니다. 교체 구성(Rotation configuration) 섹션의 Lambda 교체 함수(Lambda rotation function)에서 arn:aws:lambda:*aws-region*:*123456789012*:function:*SecretsManagerMyRotationFunction* 과 같은

Lambda 함수 ARN(Lambda function ARN)을 볼 수 있습니다. ARN 끝에서 함수 이름을 복사합니다(이 예에서는 *SecretsManagerMyRotationFunction* 입니다).

2. AWS Lambda 콘솔 https://console.aws.amazon.com/lambda/ 함수에서 검색 상자에 Lambda 함수 이름을 붙여 넣고 Enter를 선택한 다음 Lambda 함수를 선택합니다.
3. 함수 세부 정보 페이지의 구성(Configuration) 탭에서, 태그(Tags) 아래의 aws:cloudformation:stack-name 키 옆에 있는 값을 복사합니다.
4. AWS CloudFormation 콘솔 https://console.aws.amazon.com/cloudformation 스택에서 검색 상자에 키 값을 붙여넣은 다음 Enter를 선택합니다.
5. 스택 목록은 Lambda 교체 함수를 생성한 스택만 표시되도록 필터링합니다. 생성된 날짜(Created date) 열에서 스택이 생성된 날짜를 확인합니다. 이것은 Lambda 교체 함수가 생성된 날짜입니다.

오류: “모듈 'lambda_function'을 가져올 수 없음”

Python 3.7에서 최신 버전의 Python으로 자동 업그레이드된 이전 Lambda 함수를 실행하는 경우 이 오류가 발생할 수 있습니다. 오류를 해결하려면 Lambda 함수 버전을 다시 Python 3.7로 변경한 다음 the section called “기존 교체 함수를 Python 3.7에서 3.9로 업그레이드” 합니다. 자세한 내용은 AWS re:Post에서 Secrets Manager Lambda 함수 교체가 “pg 모듈을 찾을 수 없음” 오류로 인해 실패한 이유는 무엇인가요?를 참조하세요.

기존 교체 함수를 Python 3.7에서 3.9로 업그레이드

2022년 11월 이전에 생성된 일부 교체 함수는 Python 3.7을 사용했습니다. Python용 AWS SDK는 2023년 12월에 Python 3.7 지원을 중단했습니다. 자세한 내용은 AWS SDKs. Python 3.9를 사용하는 새 교체 함수로 전환하려면 기존 교체 함수에 런타임 속성을 추가하거나 교체 함수를 다시 생성할 수 있습니다.

Python 3.7을 사용하는 Lambda 교체 함수를 찾으려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/lambda/ AWS Lambda 콘솔을 엽니다.
2. 함수 목록에서 **`SecretsManager`** 를 필터링합니다.
3. 필터링된 함수 목록의 런타임에서 Python 3.7을 찾습니다.

Python 3.9로 업그레이드하려면

- 옵션 1:를 사용하여 교체 함수 다시 생성 CloudFormation

- 옵션 2:를 사용하여 기존 교체 함수의 런타임 업데이트 CloudFormation
- 옵션 3: AWS CDK 사용자의 경우 CDK 라이브러리 업그레이드

옵션 1:를 사용하여 교체 함수 다시 생성 CloudFormation

Secrets Manager 콘솔을 사용하여 교체를 켜면 Secrets Manager는 CloudFormation 를 사용하여 Lambda 교체 함수를 포함하여 필요한 리소스를 생성합니다. 콘솔을 사용하여 교체를 켜거나 CloudFormation 스택을 사용하여 교체 함수를 생성한 경우 동일한 CloudFormation 스택을 사용하여 새 이름으로 교체 함수를 다시 생성할 수 있습니다. 새 함수는 최신 버전의 Python을 사용합니다.

교체 함수를 생성한 CloudFormation 스택을 찾으려면

- Lambda 함수 세부 정보 페이지의 구성 탭에서 태그를 선택합니다. aws:cloudformation:stack-id 옆의 ARN을 확인합니다.

  스택 이름은 다음 예와 같이 ARN에 포함되어 있습니다.

  - ARN: `arn:aws:cloudformation:us-west-2:408736277230:stack/`**`SecretsManagerRDSMySQLRotationSingleUser5c2-SecretRotationScheduleHostedRotationLambda`**`-3CUDHZMDMBO8/79fc9050-2eef-11ed-`
  - 스택 이름: **`SecretsManagerRDSMySQLRotationSingleUser5c2-SecretRotationScheduleHostedRotationLambda`**

교체 함수를 다시 생성하려면(CloudFormation)

1. 에서 이름으로 스택을 CloudFormation검색한 다음 업데이트를 선택합니다.

   루트 스택 업데이트를 권장하는 대화 상자가 나타나면 루트 스택으로 이동을 선택한 다음 업데이트를 선택합니다.
2. 스택 업데이트 페이지의 템플릿 준비 에서 Application Composer에서 편집을 선택한 다음, Application Composer에서 템플릿 편집에서 Application Composer에서 편집 버튼을 선택합니다.
3. Application Composer에서 다음을 수행합니다.

   a. 템플릿 코드에 있는 `SecretRotationScheduleHostedRotationLambda`의 `"functionName": "SecretsManagerTestRotationRDS"` 값을 새 함수 이름(예: JSON의 경우 **`"functionName": "SecretsManagerTestRotationRDSupdated"`**)으로 바꿉니다.

   b. 템플릿 업데이트를 선택합니다.

c. CloudFormation계속하기 대화 상자에서 확인 후 CloudFormation계속 진행을 선택하여 계속합니다.

4. CloudFormation 스택 워크플로를 계속 진행한 다음 제출을 선택합니다.

옵션 2:를 사용하여 기존 교체 함수의 런타임 업데이트 CloudFormation

Secrets Manager 콘솔을 사용하여 교체를 켜면 Secrets Manager는 CloudFormation 를 사용하여 Lambda 교체 함수를 포함하여 필요한 리소스를 생성합니다. 콘솔을 사용하여 교체를 켜거나 CloudFormation 스택을 사용하여 교체 함수를 생성한 경우 동일한 CloudFormation 스택을 사용하여 교체 함수의 런타임을 업데이트할 수 있습니다.

교체 함수를 생성한 CloudFormation 스택을 찾으려면

- Lambda 함수 세부 정보 페이지의 구성 탭에서 태그를 선택합니다. aws:cloudformation:stack-id 옆의 ARN을 확인합니다.

  스택 이름은 다음 예와 같이 ARN에 포함되어 있습니다.

  - ARN: `arn:aws:cloudformation:us-west-2:408736277230:stack/`**`SecretsManagerRDSMySQLRotationSingleUser5c2-SecretRotationScheduleHostedRotationLambda`**`-3CUDHZMDMBO8/79fc9050-2eef-11ed-`
  - 스택 이름: **`SecretsManagerRDSMySQLRotationSingleUser5c2-SecretRotationScheduleHostedRotationLambda`**

교체 함수의 런타임을 업데이트하려면(CloudFormation)

1. 에서 이름으로 스택을 CloudFormation검색한 다음 업데이트를 선택합니다.

   루트 스택 업데이트를 권장하는 대화 상자가 나타나면 루트 스택으로 이동을 선택한 다음 업데이트를 선택합니다.

2. 스택 업데이트 페이지의 템플릿 준비 에서 Application Composer에서 편집을 선택한 다음, Application Composer에서 템플릿 편집에서 Application Composer에서 편집 버튼을 선택합니다.

3. Application Composer에서 다음을 수행합니다.

   a. 템플릿 JSON에서 `SecretRotationScheduleHostedRotationLambda`의 경우 `Properties` 아래의 `Parameters`에서 **`"runtime": "python3.9"`**을 추가합니다.

   b. 템플릿 업데이트를 선택합니다.

c. CloudFormation계속하기 대화 상자에서 확인 후 CloudFormation계속 진행을 선택하여 계속합니다.

4. CloudFormation 스택 워크플로를 계속 진행한 다음 제출을 선택합니다.

옵션 3: AWS CDK 사용자의 경우 CDK 라이브러리 업그레이드

버전 v2.94.0 AWS CDK 이전를 사용하여 보안 암호에 대한 교체를 설정한 경우 v2.94.0 이상으로 업그레이드하여 Lambda 함수를 업데이트할 수 있습니다. 자세한 내용은 AWS Cloud Development Kit (AWS CDK) v2 개발자 가이드를 참조하세요.

기존 교체 함수를 Python 3.9에서 3.10으로 업그레이드

Secrets Manager는 Lambda 교체 함수에서 Python 3.9에서 3.10으로 전환하고 있습니다. Python 3.10을 사용하는 새 교체 함수로 전환하려면, 배포 방법에 따라 업그레이드 경로를 따라야 합니다. 아래 절차를 통해 Python 버전과 기본 종속성을 모두 업그레이드할 수 있습니다.

Python 3.9를 사용하는 Lambda 교체 함수를 찾으려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/lambda/ AWS Lambda 콘솔을 엽니다.
2. 함수 목록에서 **`SecretsManager`** 를 필터링합니다.
3. 필터링된 함수 목록에서 런타임이 **`Python 3.9`**로 설정된 함수를 확인합니다.

배포 방법별 경로 업데이트

이 목록에 식별된 Lambda 교체 함수는 Secrets Manager 콘솔, AWS Serverless Application Repository 앱 또는 CloudFormation 변환을 통해 배포할 수 있습니다. 각 배포 방식에 따라 업그레이드 경로가 다릅니다.

함수가 배포된 방식을 기준으로, 다음 절차 중 하나를 사용하여 Lambda 교체 함수를 업데이트합니다.

AWS Secrets Manager console-deployed functions

기존 Lambda 함수에 대한 종속성을 수동으로 업데이트할 수 없으므로 AWS Secrets Manager 콘솔을 통해 새 Lambda 함수를 배포해야 합니다.

AWS Secrets Manager 콘솔에서 배포한 함수를 업그레이드하려면 다음 절차를 따르십시오.

1. https://console.aws.amazon.com/secretsmanager/에서 Secrets Manager 콘솔을 엽니다.

2. AWS Secrets Manager에서 보안 암호를 선택합니다. 업데이트하려는 Lambda 함수를 사용하는 보안 암호를 선택합니다.
3. 교체 탭으로 이동하여 교체 구성 업데이트 옵션을 선택합니다.
4. 교체 함수에서 새 함수 생성을 선택하고 새 Lambda 교체 함수 이름을 입력합니다.

   a. (선택 사항) 업데이트가 완료되면, 업데이트된 Lambda 함수가 정상 작동하는지 테스트할 수 있습니다. 교체 탭에서 즉시 보안 암호 교체를 선택하여 즉시 교체를 실행합니다.

   b. (선택 사항) Amazon CloudWatch에서 함수 로그와 런타임에서 사용된 Python 버전을 확인할 수 있습니다. 자세한 내용은 AWS Lambda 개발자 가이드의 Lambada 함수에 대한 CloudWatch Logs 보기를 참조하세요.
5. 새 교체 함수 설정이 완료되면 이전 교체 함수를 삭제할 수 있습니다.

AWS Serverless Application Repository deployments

다음 절차에서는 AWS Serverless Application Repository 배포를 업그레이드하는 방법을 보여줍니다. AWS Serverless Application Repository 를 통해 배포된 Lambda 함수에는 함수가 속한 Lambda 애플리케이션에 대한 링크가 `This function belongs to an application. Click here to manage it.` 포함된 배너가 있습니다.

> **Important**
>
> AWS Serverless Application Repository 가용성은 AWS 리전 종속적입니다.

AWS Serverless Application Repository 배포된 함수를 업데이트하려면 다음 절차를 따르십시오.

1. https://console.aws.amazon.com/lambda/ AWS Lambda 콘솔을 엽니다.
2. 업데이트가 필요한 Lambda 함수의 구성 탭으로 이동합니다.

   - 배포된 AWS Serverless Application Repository 애플리케이션을 업데이트할 때 함수에 대한 다음 정보가 필요합니다. 이 정보는 Lambda 콘솔에서 확인할 수 있습니다.

     - Lambda 애플리케이션 이름
       - Lambda 애플리케이션 이름은 배너에 있는 링크를 통해 확인할 수 있습니다. 예를 들어, 배너에 `serverlessrepo-`*`SecretsManagerRedshiftRotationSingleUser`*라고 표시

된 경우, 애플리케이션 이름은 `SecretsManagerRedshiftRotationSingleUser`입니다.

- Lambda 교체 함수 이름
- Secrets Manager 엔드포인트
  - 이 엔드포인트는 구성 탭과 환경 변수 탭에서 SECRETS_MANAGER_ENDPOINT 변수에 할당된 값을 통해 확인할 수 있습니다.

3. Python 버전을 업그레이드하려면, 서버리스 애플리케이션의 시맨틱 버전을 업데이트해야 합니다. 자세한 내용은 AWS Serverless Application Repository 개발자 가이드의 애플리케이션 업데이트를 참조하세요.

Custom Lambda rotation functions

사용자 지정 Lambda 교체 함수를 생성한 경우, 각 함수의 패키지 종속성과 런타임을 업그레이드해야 합니다. 자세한 내용은 Lambda 함수 런타임을 최신 버전으로 업그레이드를 참조하세요.

AWS::SecretsManager-2024-09-16 transform macro

Lambda 함수가 이 변환을 통해 배포된 경우, 기존 템플릿을 사용하여 스택을 업데이트하면 업데이트된 Lambda 런타임을 사용할 수 있습니다.

기존 템플릿을 사용하여 CloudFormation 스택을 업데이트하려면 다음 절차를 따르십시오.

1. https://console.aws.amazon.com/cloudformation CloudFormation 콘솔을 엽니다.
2. 스택 페이지에서 업데이트하려는 스택을 선택합니다.
3. 스택 세부 정보 창에서 업데이트를 선택합니다.
4. 템플릿 업데이트 방법 선택 화면에서 직접 업데이트를 선택합니다.
5. 템플릿 지정 페이지에서 기존 템플릿 사용을 선택합니다.
6. 나머지 옵션은 기본값으로 유지한 후 스택 업데이트를 선택합니다.

스택 업데이트 중 문제가 발생하면 CloudFormation 사용 설명서의 스택 실패 원인 파악을 참조하세요.

AWS::SecretsManager-2020-07-23 transform macro

`AWS::SecretsManager-2020-07-23`를 사용 중이라면 최신 변환 버전으로 마이그레이션하는 것을 권장합니다. 자세한 내용은 AWS 보안 블로그의 AWS Secrets Manager 변환의 향상된 버전 소개: AWS::SecretsManager-2024-09-16를 참조하세요. 만약 `AWS::SecretsManager-2020-07-23`를 계속 사용하면 런타임 버전과 Lambda 함수 코드

아티팩트 간 불일치 오류가 발생할 수 있습니다. 자세한 내용은 CloudFormation 템플릿 참조의 AWS::SecretsManager::RotationSchedule HostedRotationLambda 섹션을 확인하세요.

스택 업데이트 중 문제가 발생하면 CloudFormation 사용 설명서의 스택 실패 원인 파악을 참조하세요.

Python 업그레이드 확인

Python 업그레이드를 확인하려면 Lambda 콘솔(https://console.aws.amazon.com/lambda/)을 열고 함수 페이지에 액세스합니다. 업데이트한 함수를 선택합니다. 코드 소스 섹션에서 디렉터리에 포함된 파일을 검토하고 Python .so 파일이 버전 `3.10`인지 확인합니다.

AWS Lambda 를 사용한 보안 암호 교체 **PutSecretValue** 실패

Secrets Manager에서 수임된 역할 또는 교차 계정 교체를 사용하는 경우, CloudTrail에서 다음과 같은 메시지와 함께 RotationFailed 이벤트가 나타날 수 있습니다. Pending secret version *VERSION_ID* for Secret *SECRET_ARN* was not created by *Lambda LAMBDA_ARN.* 이 경우 `AWSPENDING` 스테이징 레이블을 제거하고 교체를 다시 시작한 다음, Lambda 함수를 업데이트하여 `RotationToken` 파라미터를 사용정해야 합니다.

`RotationToken`을 포함하도록 Lambda 교체 함수 업데이트

1. Lambda 함수 코드 다운로드

   - Lambda 콘솔을 엽니다.
   - 탐색 창에서 함수를 선택합니다.
   - 함수 이름에서 Lambda 보안 암호 교체 함수를 선택합니다.
   - 다운로드에서 함수 코드 .zip, AWS SAM 파일, 둘 다 중 하나를 선택합니다.
   - 확인을 선택하여 로컬 시스템에 함수를 저장합니다.

2. `Lambda_handler` 편집

   교차 계정 교체를 위해 create_secret 단계에 rotation_token 파라미터를 포함시킵니다.

```
def lambda_handler(event, context):
    """Secrets Manager Rotation Template

    This is a template for creating an AWS Secrets Manager rotation lambda
```

```
   Args:
       event (dict): Lambda dictionary of event parameters. These keys must
include the following:
           - SecretId: The secret ARN or identifier
           - ClientRequestToken: The ClientRequestToken of the secret version
           - Step: The rotation step (one of createSecret, setSecret, testSecret,
or finishSecret)
           - RotationToken: the rotation token to put as parameter for
PutSecretValue call

       context (LambdaContext): The Lambda runtime information

   Raises:
       ResourceNotFoundException: If the secret with the specified arn and stage
does not exist

       ValueError: If the secret is not properly configured for rotation

       KeyError: If the event parameters do not contain the expected keys

   """
   arn = event['SecretId']
   token = event['ClientRequestToken']
   step = event['Step']
   # Add the rotation token
   rotation_token = event['RotationToken']

   # Setup the client
   service_client = boto3.client('secretsmanager',
endpoint_url=os.environ['SECRETS_MANAGER_ENDPOINT'])

   # Make sure the version is staged correctly
   metadata = service_client.describe_secret(SecretId=arn)
   if not metadata['RotationEnabled']:
       logger.error("Secret %s is not enabled for rotation" % arn)
       raise ValueError("Secret %s is not enabled for rotation" % arn)
   versions = metadata['VersionIdsToStages']
   if token not in versions:
       logger.error("Secret version %s has no stage for rotation of secret %s." %
(token, arn))
       raise ValueError("Secret version %s has no stage for rotation of secret
%s." % (token, arn))
   if "AWSCURRENT" in versions[token]:
```

```
        logger.info("Secret version %s already set as AWSCURRENT for secret %s." %
 (token, arn))
        return
    elif "AWSPENDING" not in versions[token]:
        logger.error("Secret version %s not set as AWSPENDING for rotation of
 secret %s." % (token, arn))
        raise ValueError("Secret version %s not set as AWSPENDING for rotation of
 secret %s." % (token, arn))
    # Use rotation_token
    if step == "createSecret":
        create_secret(service_client, arn, token, rotation_token)

    elif step == "setSecret":
        set_secret(service_client, arn, token)

    elif step == "testSecret":
        test_secret(service_client, arn, token)

    elif step == "finishSecret":
        finish_secret(service_client, arn, token)

    else:
        raise ValueError("Invalid step parameter")
```

3. `create_secret` 코드 편집

   `rotation_token` 파라미터를 수락하고 사용하도록 `create_secret` 함수를 수정합니다.

```
# Add rotation_token to the function
def create_secret(service_client, arn, token, rotation_token):
"""Create the secret

This method first checks for the existence of a secret for the passed in token. If
 one does not exist, it will generate a
new secret and put it with the passed in token.

Args:
service_client (client): The secrets manager service client

arn (string): The secret ARN or other identifier

token (string): The ClientRequestToken associated with the secret version
```

```
rotation_token (string): the rotation token to put as parameter for PutSecretValue
 call

Raises:
ResourceNotFoundException: If the secret with the specified arn and stage does not
 exist

"""
# Make sure the current secret exists
service_client.get_secret_value(SecretId=arn, VersionStage="AWSCURRENT")

# Now try to get the secret version, if that fails, put a new secret
try:
service_client.get_secret_value(SecretId=arn, VersionId=token,
 VersionStage="AWSPENDING")
logger.info("createSecret: Successfully retrieved secret for %s." % arn)
except service_client.exceptions.ResourceNotFoundException:
# Get exclude characters from environment variable
exclude_characters = os.environ['EXCLUDE_CHARACTERS'] if 'EXCLUDE_CHARACTERS' in
 os.environ else '/@"\'\\'
# Generate a random password
passwd = service_client.get_random_password(ExcludeCharacters=exclude_characters)

# Put the secret, using rotation_token
service_client.put_secret_value(SecretId=arn, ClientRequestToken=token,
 SecretString=passwd['RandomPassword'], VersionStages=['AWSPENDING'],
 RotationToken=rotation_token)
logger.info("createSecret: Successfully put secret for ARN %s and version %s." %
 (arn, token))
```

4. 업데이트된 Lambda 함수 코드 업로드

   Lambda 함수 코드를 업데이트한 후 업로드하여 보안 암호를 교체합니다.

오류: "*<a rotation>* 단계에서 Lambda *<arn>*을 실행할 때 오류"

만약 Lambda 함수가 CreateSecret과 SetSecret 단계 사이에서 반복적으로 멈추는 등 간헐적인 보안 암호 교체 실패가 발생한다면, 문제는 동시성 설정과 관련이 있을 수 있습니다.

동시성 문제 해결 단계

> **Warning**
>
> 프로비저닝된 동시성 파라미터를 10보다 낮게 설정하면 Lambda 함수 실행 스레드가 부족하여 스로틀링이 발생할 수 있습니다. 자세한 내용은 AWS Lambda AWS Lambda 개발자 가이드의 예약된 동시성 및 프로비저닝된 동시성 이해를 참조하세요.

1. Lambda 동시성 설정 확인 및 조정:
   - `reserved_concurrent_executions` 값이 너무 낮게 설정되어 있지 않은지 확인합니다(예: 1).
   - 예약된 동시성을 사용하는 경우 10 이상으로 설정합니다.
   - 더 유연하게 사용하려면 예약되지 않은 동시성을 고려합니다.
2. 프로비저닝된 동시성 사용 시:
   - 프로비저닝 동시성 파라미터를 명시적으로 설정하지 마세요(예: Terraform에서).
   - 반드시 설정해야 하는 경우 10 이상의 값을 사용합니다.
   - 선택한 값이 실제 사용 사례에 적합한지 충분히 테스트합니다.
3. 동시성 모니터링 및 조정:
   - 다음 공식을 사용하여 동시성을 계산합니다. 동시성 수 = (초당 평균 요청 수) * (요청 평균 처리 시간(초)) 자세한 내용은 예약된 동시성 예측을 참조하세요.
   - 교체 중에 값을 관찰하고 기록하여 적절한 동시성 설정을 결정합니다.
   - 낮은 동시성 값을 설정할 경우 주의합니다. 실행 스레드가 충분하지 않으면 스로틀링이 발생할 수 있습니다.

Lambda 동시성 구성에 대한 자세한 내용은 AWS Lambda 개발자 안내서의 예약된 동시성 구성 및 프로비저닝된 동시성 구성을 참조하세요.

교체 일정

Secrets Manager는 설정된 교체 기간 중 일정에 따라 보안 암호를 교체합니다. 일정 및 기간을 설정하려면 cron() 또는 rate() 표현식을 기간과 함께 사용합니다. Secrets Manager는 교체 기간 중 임의의 시

점에 보안 암호를 교체합니다. 4시간마다 암호를 교체할 수 있으며, 교체 기간은 최소 1시간 이내입니다.

교체를 켜려면 다음을 참조하세요.

- the section called “관리형 교체”
- the section called “데이터베이스 보안 암호 자동 교체(콘솔)”
- the section called “비데이터베이스 보안 암호 자동 교체(콘솔)”

Secrets Manager 교체 일정은 UTC 표준 시간대를 사용합니다.

교체 기간

Secrets Manager 교체 기간은 유지 관리 기간과 비슷합니다. 보안 암호를 교체하려는 경우 교체 기간을 설정합니다. 이렇게 하면 Secrets Manager가 교체 기간 중 임의의 시기에 보안 암호를 교체합니다.

Secrets Manager의 교체 기간은 항상 정각에 시작됩니다. 일 단위로 `rate()` 표현식을 사용하는 교체 일정의 경우, 교체 기간은 자정에 시작됩니다. `cron()` 표현식을 사용하여 교체 기간의 시작 시간을 설정할 수 있습니다. 예시는 the section called “cron 표현식” 섹션을 참조하세요.

기본적으로 교체 기간은 시간 단위 교체 일정의 경우 1시간 후에 끝나고, 일 단위 교체 일정의 경우 일과 종료 시 끝납니다.

지속 시간을 설정하여 교체 기간의 길이를 변경할 수 있습니다. 교체 기간을 최소 1시간으로 설정할 수 있습니다. 교체 기간은 그 다음 교체 기간까지 연장되지 않아야 합니다. 즉, 시간 단위 교체 일정의 경우 교체 기간이 교체 간 시간 간격 이하인지 확인합니다. 일 단위 교체 일정의 경우 시작 시간 + 지속 시간이 24시간 이하인지 확인합니다.

rate 표현식

Secrets Manager rate 표현식의 형식은 다음과 같습니다. 여기서 *`Value`*는 양의 정수이고 *`Unit`*은 `hour`, `hours`, `day` 또는 `days`이(가) 될 수 있습니다:

```
rate(Value Unit)
```

보안 암호를 4시간마다 교체할 수 있습니다. 최대 교체 기간은 999일입니다. 예시:

- `rate(4 hours)`은(는) 보안 암호가 4시간마다 교체된다는 뜻입니다.

- `rate(1 day)`은(는) 보안 암호가 매일 교체된다는 뜻입니다.
- `rate(10 days)`은(는) 보안 암호가 10일마다 교체된다는 뜻입니다.

cron 표현식

Secrets Manager cron 표현식의 형식은 다음과 같습니다.

```
cron(Minutes Hours Day-of-month Month Day-of-week Year)
```

시간 증분을 포함하는 cron 표현식은 매일 재설정됩니다. 예를 들어, `cron(0 4/12 * * ? *)`은(는) 오전 4시, 오후 4시, 다음 날 오전 4시, 오후 4시를 의미합니다. Secrets Manager 교체 일정은 UTC 표준 시간대를 사용합니다.

일정 예약 예	표현식
자정부터 8시간마다.	`cron(0 /8 * * ? *)`
오전 8시부터 8시간마다.	`cron(0 8/8 * * ? *)`
오전 2시부터 10시간마다. 교체 기간은 2:00시, 12:00, 22:00, 다음 날 2:00시, 12:00, 22:00에 시작됩니다.	`cron(0 2/10 * * ? *)`
매일 오전 10시	`cron(0 10 * * ? *)`
매주 토요일 오후 6시.	`cron(0 18 ? * SAT *)`
매월 1일 오전 8시.	`cron(0 8 1 * ? *)`
매 3개월마다 첫 번째 일요일 오전 1시.	`cron(0 1 ? 1/3 SUN#1 *)`
매월 마지막날 오후 5시.	`cron(0 17 L * ? *)`
월요일부터 금요일까지 오전 8시.	`cron(0 8 ? * MON-FRI *)`
매월 1일과 15일 오후 4시.	`cron(0 16 1,15 * ? *)`
매월 첫 번째 일요일 자정.	`cron(0 0 ? * SUN#1 *)`

일정 예약 예	표현식
1월을 시작으로, 11개월마다 첫 번째 월요일 자정.	`cron(0 0 ? 1/11 2#1 *)`

Secrets Manager의 Cron 표현식 요구 사항

Secrets Manager에는 cron 표현식에 사용할 수 있는 항목에 대한 제한 사항이 몇 가지 있습니다. Secrets Manager 교체 기간은 정시에 시작되므로 Secrets Manager 대한 cron 표현식에서 minutes(분) 필드의 값은 0이 되어야 합니다. Secrets Manager는 1년 이상 간격의 교체 일정을 지원하지 않으므로 Year 필드의 값은 *여야 합니다. 다음 표에 사용할 수 있는 옵션이 나와 있습니다.

필드	값	와일드카드
Minutes	0이어야 함	없음
Hours	0~23	/(슬래시)를 사용하여 증분을 지정합니다. 예를 들어 `2/10`은(는) 오전 2시부터 10시간마다를 의미합니다. 보안 암호를 4시간마다 교체할 수 있습니다.
Day-of-month	1~31	,(쉼표)를 사용하여 추가 값을 포함합니다. 예를 들어 `1,15`은(는) 해당 월의 1일과 15일을 의미합니다. -(대시)를 사용하여 범위를 지정합니다. 예를 들어 `1-15`은(는) 해당 월의 1일~15일을 의미합니다. *(별표)를 사용하여 필드에 모든 값을 포함합니다. 예를 들어 *은(는) 해당 월의 모든 날짜를 의미합니다.

필드	값	와일드카드
		?(물음표) 와일드카드는 어떤 한 가지나 다른 것을 지정합니다. 동일한 cron 표현식에 `Day-of-month` 와 `Day-of-week` 필드를 지정할 수 없습니다. 이 필드 중 하나에 값을 지정하는 경우에는 다른 필드에서 반드시 ?(물음표)를 사용해야 합니다. /(슬래시)를 사용하여 증분을 지정합니다. 예를 들어, 1/2은(는) 1일째부터 이틀마다, 즉 1일째, 3일째, 5일째 등을 의미합니다. L을 사용하여 해당 월의 마지막 날을 지정합니다. ***DAY***L을 사용하여 해당 월의 마지막 명명일을 지정합니다. 예를 들어 SUNL은(는) 해당 월의 마지막 일요일을 의미합니다.

필드	값	와일드카드
Month	1~12 또는 JAN~DEC	,(쉼표)를 사용하여 추가 값을 포함합니다. 예를 들어, `JAN,APR,JUL,OCT` 은(는) 1월, 4월, 7월, 10월을 의미합니다. -(대시)를 사용하여 범위를 지정합니다. 예를 들어 `1-3`은(는) 해당 연도의 1월~3월을 의미합니다. *(별표)를 사용하여 필드에 모든 값을 포함합니다. 예를 들어 *은(는) 모든 월을 의미합니다. /(슬래시)를 사용하여 증분을 지정합니다. 예를 들어, `1/3`은(는) 첫번째 월에서 시작하여 매 3개월마다, 즉 첫째 달, 넷째 달, 일곱째 달, 열번째 달을 의미합니다.

필드	값	와일드카드
Day-of-week	1~7 또는 SUN~SAT	#을 사용하여 한 달 내의 요일을 지정합니다. 예를 들어 TUE#3은 그 달의 세 번째 화요일을 의미합니다. ,(쉼표)를 사용하여 추가 값을 포함합니다. 예를 들어 1,4은(는) 해당 주의 첫째 요일과 넷째 요일을 의미합니다. -(대시)를 사용하여 범위를 지정합니다. 예를 들어 1-4은(는) 해당 주의 첫째 요일~넷째 요일을 의미합니다. *(별표)를 사용하여 필드에 모든 값을 포함합니다. 예를 들어 *은(는) 해당 주의 모든 요일을 의미합니다. ?(물음표) 와일드카드는 어떤 한 가지나 다른 것을 지정합니다. 동일한 cron 표현식에 Day-of-month 와 Day-of-week 필드를 지정할 수 없습니다. 이 필드 중 하나에 값을 지정하는 경우에는 다른 필드에서 반드시 ?(물음표)를 사용해야 합니다. /(슬래시)를 사용하여 증분을 지정합니다. 예를 들어, 1/2은(는) 해당 주의 첫 번째 요일부터 시작하여 격일로, 즉 첫째 요일, 셋째 요일, 다섯째 요일, 일곱째 요일을 의미합니다.

필드	값	와일드카드
		L을 사용하여 해당 주의 마지막 요일을 지정합니다.
Year	*여야 합니다.	없음

AWS Secrets Manager 보안 암호를 즉시 교체

교체가 구성된 보안 암호만 교체할 수 있습니다. 교체에 대한 보안 암호가 구성되어 있는지 확인하려면 콘솔에서 보안 암호를 확인하고 Rotation configuration(교체 구성) 섹션으로 스크롤합니다. Rotation status(교체 상태)가 Enabled(활성)이면 교체에 대한 보안 암호가 구성되어 있습니다. 그렇지 않은 경우 보안 암호 교체를 참조하세요.

보안 암호를 즉시 교체하려면(콘솔)

1. https://console.aws.amazon.com/secretsmanager/에서 Secrets Manager 콘솔을 엽니다.
2. 보안 암호를 선택합니다.
3. 보안 암호 세부 정보 페이지의 교체 구성(Rotation configuration)에서 보안 암호 즉시 교체(Rotate secret immediately)를 선택합니다.
4. 보안 암호 교체(Rotate secret) 대화 상자에서 교체(Rotate)를 선택합니다.

AWS CLI

Example보안 암호 즉시 교체

다음 rotate-secret 예에서는 즉시 교체를 시작합니다. 보안 암호에 교체가 미리 구성되어 있어야 합니다.

```
$ aws secretsmanager rotate-secret \
    --secret-id MyTestSecret
```

교체되지 않은 보안 암호 찾기

AWS Config 를 사용하여 보안 암호를 평가하여 보안 암호가 표준에 따라 교체되고 있는지 확인할 수 있습니다. AWS Config 규칙을 사용하여 보안 암호에 대한 내부 보안 및 규정 준수 요구 사항을 정의합

니다. 그러면 AWS Config 가 규칙을 준수하지 않는 보안 암호를 식별할 수 있습니다. 또한 보안 암호 메타데이터, 교체 구성, 보안 암호 암호화에 사용되는 KMS 키, Lambda 교체 함수 및 보안 암호와 연결된 태그에 대한 변경 사항을 추적할 수 있습니다.

여러 AWS 계정 및 조직에 보안 암호가 있는 경우 해당 구성 및 규정 준수 데이터를 집계할 수 AWS 리전 있습니다. 자세한 내용은 다중 계정 다중 리전 데이터 집계를 참조하세요.

보안 암호가 교체되고 있는지 평가하는 방법

1. AWS Config 규칙을 사용하여 리소스 평가에 대한 지침을 따르고 다음 규칙 중 하나를 선택합니다.

   - `secretsmanager-rotation-enabled-check` - Secrets Manager에 저장된 보안 암호에 대해 교체가 구성되었는지 확인합니다.
   - `secretsmanager-scheduled-rotation-success-check`— 마지막으로 성공한 교체가 설정된 교체 주기 내에 있는지 확인합니다. 최소 확인 주기는 매일입니다.
   - `secretsmanager-secret-periodic-rotation` - 보안 암호를 지정된 일수 내에 교체했는지 확인합니다.
2. 선택적으로 보안 암호가 규정을 준수하지 않을 때 알리 AWS Config 도록를 구성합니다. 자세한 내용은 가 Amazon SNS 주제로 AWS Config 보내는 알림을 참조하세요.

Secrets Manager에서 자동 교체 취소

보안 암호에 대한 자동 교체를 구성한 후 자동 교체를 중지하려는 경우, 교체를 취소할 수 있습니다.

자동 교체를 취소하는 방법

1. https://console.aws.amazon.com/secretsmanager/에서 Secrets Manager 콘솔을 엽니다.
2. 보안 암호를 선택합니다.
3. 보안 암호 세부 정보 페이지의 교체 구성 섹션에서 교체 편집을 선택합니다.
4. 교체 구성 편집 대화 상자에서 자동 교체를 끈 다음, 저장을 선택합니다.

   교체를 다시 켜기로 결정한 경우, Secrets Manager는 교체 구성 정보를 유지하여 나중에 사용할 수 있도록 합니다.

AWS Secrets Manager 다른 AWS 서비스에서 관리하는 보안 암호

많은 AWS 서비스가에 보안 암호를 저장하고 사용합니다 AWS Secrets Manager. 이러한 보안 암호는 관리형 보안 암호인 경우도 있습니다. 즉, 이러한 보안 암호를 생성한 서비스가 이를 관리하는 데 도움이 됩니다. 예를 들어 일부 관리형 보안 암호에는 관리형 교체가 포함되므로 직접 교체를 구성할 필요가 없습니다. 관리 서비스는 사용자가 보안 암호를 업데이트하거나 복구 기간 없이 삭제하지 못하도록 제한할 수도 있습니다. 이렇게 하면 보안 암호에 의존하여 관리 서비스가 수행되므로 중단을 방지하는 데 도움이 됩니다.

> **Note**
>
> 관리형 보안 암호는 이를 관리하는 AWS 서비스에서만 생성할 수 있습니다.

관리형 보안 암호는 식별하기 쉽도록 관리 서비스 ID가 포함된 명명 규칙을 사용합니다.

```
Secret name: ServiceID!MySecret
Secret ARN : arn:aws:us-east-1:ServiceID!MySecret-a1b2c3
```

보안 암호를 관리하는 서비스의 ID

- `appflow` – the section called “Amazon AppFlow”
- `databrew` – the section called “AWS Glue DataBrew”
- `datasync` – the section called “AWS DataSync”
- `directconnect` – the section called “Direct Connect”
- `ecs-sc` – the section called “Amazon Elastic Container Service”
- `events` – the section called “Amazon EventBridge”
- `marketplace-deployment` – the section called “AWS Marketplace”
- `opsworks-cm` – the section called “AWS OpsWorks for Chef Automate”
- `pcs` – the section called “AWS 병렬 컴퓨팅 서비스”
- `rds` – the section called “Amazon RDS”
- `redshift` – the section called “Amazon Redshift”

- `sqlworkbench` – the section called “Amazon Redshift Query Editor V2”

다른 AWS 서비스에서 관리하는 보안 암호를 찾으려면 관리형 보안 암호 찾기를 참조하세요.

AWS 서비스 AWS Secrets Manager 보안 암호를 사용하는

다음과 같은 각 AWS 서비스 가 Secrets Manager와 어떻게 통합되는지에 대한 정보를 얻습니다.

- 에서를 AWS App Runner 사용하는 방법 AWS Secrets Manager
- AWS App2Container가를 사용하는 방법 AWS Secrets Manager
- 에서를 AWS AppConfig 사용하는 방법 AWS Secrets Manager
- Amazon AppFlow의 사용 방식 AWS Secrets Manager
- 에서를 AWS AppSync 사용하는 방법 AWS Secrets Manager
- Amazon Athena의 사용 방식 AWS Secrets Manager
- Amazon Aurora의 사용 방식 AWS Secrets Manager
- 에서를 AWS CodeBuild 사용하는 방법 AWS Secrets Manager
- Amazon Data Firehose의 사용 방식 AWS Secrets Manager
- 에서를 AWS DataSync 사용하는 방법 AWS Secrets Manager
- Amazon DataZone의 사용 방식 AWS Secrets Manager
- 에서를 AWS Direct Connect 사용하는 방법 AWS Secrets Manager
- 에서를 AWS Directory Service 사용하는 방법 AWS Secrets Manager
- Amazon DocumentDB(MongoDB 호환)의 사용 방식 AWS Secrets Manager
- 에서를 AWS Elastic Beanstalk 사용하는 방법 AWS Secrets Manager
- Amazon Elastic Container Registry의 사용 방식 AWS Secrets Manager
- Amazon Elastic Container Service
- Amazon ElastiCache의 사용 방식 AWS Secrets Manager
- 에서를 AWS Elemental Live 사용하는 방법 AWS Secrets Manager
- 에서를 AWS Elemental MediaConnect 사용하는 방법 AWS Secrets Manager
- 에서를 AWS Elemental MediaConvert 사용하는 방법 AWS Secrets Manager
- 에서를 AWS Elemental MediaLive 사용하는 방법 AWS Secrets Manager

- 에서를 AWS Elemental MediaPackage 사용하는 방법 AWS Secrets Manager
- 에서를 AWS Elemental MediaTailor 사용하는 방법 AWS Secrets Manager
- Amazon EMR에서 Secrets Manager를 사용하는 방법
- Amazon EventBridge의 사용 방식 AWS Secrets Manager
- Amazon FSx에서 AWS Secrets Manager 보안 암호를 사용하는 방법
- 에서를 AWS Glue DataBrew 사용하는 방법 AWS Secrets Manager
- AWS Glue Studio에서를 사용하는 방법 AWS Secrets Manager
- 에서를 AWS IoT SiteWise 사용하는 방법 AWS Secrets Manager
- Amazon Kendra의 사용 방식 AWS Secrets Manager
- Amazon Kinesis Video Streams의 사용 방식 AWS Secrets Manager
- 에서를 AWS Launch Wizard 사용하는 방법 AWS Secrets Manager
- Amazon Lookout for Metrics의 사용 방식 AWS Secrets Manager
- Amazon Managed Grafana의 사용 방식 AWS Secrets Manager
- 에서를 AWS Managed Services 사용하는 방법 AWS Secrets Manager
- Amazon Managed Streaming for Apache Kafka의 사용 방식 AWS Secrets Manager
- Amazon Managed Workflows for Apache Airflow의 사용 방식 AWS Secrets Manager
- AWS Marketplace
- 에서를 AWS Migration Hub 사용하는 방법 AWS Secrets Manager
- 에서 Secrets Manager를 AWS Panorama 사용하는 방법
- AWS 병렬 컴퓨팅 서비스의 사용 방식 AWS Secrets Manager
- 에서를 AWS ParallelCluster 사용하는 방법 AWS Secrets Manager
- Amazon Q의 Secrets Manager 활용 방식
- Amazon OpenSearch Ingestion이 Secrets Manager를 사용하는 방법
- 에서를 AWS OpsWorks for Chef Automate 사용하는 방법 AWS Secrets Manager
- Amazon Quick에서를 사용하는 방법 AWS Secrets Manager
- Amazon RDS의 사용 방식 AWS Secrets Manager
- Amazon Redshift의 사용 방식 AWS Secrets Manager
- Amazon Redshift Query Editor V2
- Amazon SageMaker AI의 사용 방식 AWS Secrets Manager

- 에서를 AWS Schema Conversion Tool 사용하는 방법 AWS Secrets Manager
- Amazon Timestream for InfluxDB의 사용 방식 AWS Secrets Manager
- 에서를 AWS Toolkit for JetBrains 사용하는 방법 AWS Secrets Manager
- 에서 AWS Secrets Manager 보안 암호를 AWS Transfer Family 사용하는 방법
- 에서 AWS Secrets Manager 보안 암호를 AWS Wickr사용하는 방법

에서를 AWS App Runner 사용하는 방법 AWS Secrets Manager

AWS App Runner 는 소스 코드 또는 컨테이너 이미지에서 AWS 클라우드의 확장 가능하고 안전한 웹 애플리케이션으로 직접 배포할 수 있는 빠르고 간단하며 비용 효율적인 방법을 제공하는 AWS 서비스입니다. 새로운 기술을 배우거나, 사용할 컴퓨팅 서비스를 결정하거나, AWS 리소스를 프로비저닝하고 구성하는 방법을 알 필요가 없습니다.

App Runner를 사용하면 서비스를 만들거나 서비스 구성을 업데이트할 때 보안 암호와 구성을 서비스의 환경 변수로 참조할 수 있습니다. 자세한 내용은 AWS App Runner 개발자 가이드의 환경 변수 참조 및 환경 변수 관리를 참조하세요.

AWS App2Container가를 사용하는 방법 AWS Secrets Manager

AWS App2Container 는 온프레미스 데이터 센터 또는 가상 머신에서 실행되는 애플리케이션을 리프트 앤 시프트하여 Amazon ECS, Amazon EKS 또는에서 관리하는 컨테이너에서 실행되는 명령줄 도구입니다 AWS App Runner.

App2Container는 Secrets Manager를 사용하여 원격 명령을 실행하기 위해 작업자 머신을 애플리케이션 서버에 연결하는 데 필요한 자격 증명을 관리합니다. 자세한 내용은 AWS App2Container 사용 설명서의 App2Container의 보안 암호 관리를 참조하세요. AWS App2Container

에서를 AWS AppConfig 사용하는 방법 AWS Secrets Manager

AWS AppConfig 는 애플리케이션 구성을 생성, 관리 및 빠르게 배포하는 데 사용할 수 AWS Systems Manager 있는의 기능입니다. 구성에는 Secrets Manager에 저장된 자격 증명 데이터 또는 기타 중요한 정보가 포함될 수 있습니다. 자유 형식 구성 프로필을 만들 때 Secrets Manager를 구성 데이터의 소스로 선택할 수 있습니다. 자세한 내용은AWS AppConfig 사용 설명서의 Creating a freeform configuration profile(자유 형식 구성 프로필 생성)을 참조하세요. 가 자동 교체가 활성화된 보안 암호를 AWS AppConfig 처리하는 방법에 대한 자세한 내용은 AWS AppConfig 사용 설명서의 Secrets Manager 키 교체를 참조하세요.

Amazon AppFlow의 사용 방식 AWS Secrets Manager

Amazon AppFlow는 Salesforce와 같은 애플리케이션, Amazon Simple Storage Service(Amazon S3) 및 Amazon Redshift 등의 AWS 서비스같은 서비스형 소프트웨어(SaaS) 애플리케이션 간에 데이터를 안전하게 교환할 수 있는 완전관리형 통합 서비스입니다.

Amazon AppFlow에서 SaaS 애플리케이션을 소스 또는 대상으로 구성하여 연결을 생성합니다. 인증 토큰, 사용자 이름, 암호 등 SaaS 애플리케이션에 연결하는 데 필요한 정보가 여기에 포함됩니다. Amazon AppFlow는 연결 데이터를 접두사가 `appflow`인 Secrets Manager 관리형 보안 암호에 저장합니다. 보안 암호 저장 비용은 Amazon AppFlow 요금에 포함됩니다. 자세한 내용은 Amazon AppFlow 사용 설명서의 Amazon AppFlow의 데이터 보호를 참조하세요.

에서를 AWS AppSync 사용하는 방법 AWS Secrets Manager

AWS AppSync 는 애플리케이션 개발자가 Amazon DynamoDB 및 AWS Lambda HTTP API를 비롯한 여러 소스의 데이터를 결합할 수 있는 강력하고 확장 가능한 GraphQL 인터페이스를 제공합니다. APIs

AWS AppSync 는 Secrets Manager 보안 암호의 자격 증명을 사용하여 Amazon RDS 및 Aurora에 연결합니다. 자세한 내용은 AWS AppSync 개발자 가이드의 자습서: Aurora Serverless를 참조하세요.

Amazon Athena의 사용 방식 AWS Secrets Manager

Amazon Athena는표준 SQL을 사용하여 Amazon Simple Storage Service(S3)에 있는 데이터를 직접 간편하게 분석할 수 있는 대화형 쿼리 서비스입니다.

Amazon Athena 데이터 원본 커넥터는 Secrets Manager 보안 암호로 Athena 연합 쿼리 기능을 사용해 데이터를 쿼리할 수 있습니다. 자세한 내용은 Amazon Athena 사용 설명서의 Amazon Athena 페더레이션 쿼리 사용을 참조하세요.

Amazon Aurora의 사용 방식 AWS Secrets Manager

Amazon Aurora는 MySQL 및 PostgreSQL과 호환되는 완전 관리형 관계형 데이터베이스 엔진입니다.

Aurora의 마스터 사용자 자격 증명을 관리하기 위해 Aurora는 관리형 보안 암호를 생성할 수 있습니다. 보안 암호에 대한 요금이 청구됩니다. 또한 Aurora는 이러한 보안 인증 정보의 교체를 관리합니다. 자세한 내용은 Amazon Aurora 사용 설명서의 Amazon Aurora 및 AWS Secrets Manager을 사용한 암호 관리를 참조하세요.

기타 Aurora 자격 증명에 대한 내용은 보안 암호 생성 섹션을 참조하세요.

Amazon RDS 데이터 API를 호출할 때 Secrets Manager 보안 암호를 사용하여 데이터베이스에 대한 자격 증명을 전달할 수 있습니다. 자세한 내용을 알아보려면 Amazon Aurora 사용 설명서의 Aurora Serverless에 데이터 API 사용을 참조하세요.

Amazon RDS 쿼리 편집기를 사용하여 데이터베이스에 연결하면 데이터 베이스에 대한 보안 인증 정보를 Secrets Manager에 저장할 수 있습니다. 자세한 내용은 Amazon RDS 사용 설명서의 쿼리 편집기 사용을 참조하세요.

에서를 AWS CodeBuild 사용하는 방법 AWS Secrets Manager

AWS CodeBuild 는 클라우드의 완전 관리형 빌드 서비스입니다. CodeBuild는 소스 코드를 컴파일하고 단위 테스트를 실행하며 배포 준비가 완료된 아티팩트를 생성합니다.

Secrets Manager를 사용하여 프라이빗 레지스트리 자격 증명을 저장할 수 있습니다. 자세한 내용은 AWS CodeBuild 사용 설명서의 CodeBuild용 AWS Secrets Manager 샘플을 사용한 프라이빗 레지스트리를 참조하세요.

Amazon Data Firehose의 사용 방식 AWS Secrets Manager

Amazon Data Firehose를 사용하여 실시간 스트리밍 데이터를 다양한 스트리밍 대상으로 전송할 수 있습니다. 대상에 자격 증명 또는 키가 필요한 경우, Firehose는 런타임 시 Secrets Manager에서 보안 암호를 검색하여 대상에 연결합니다. 자세한 내용은 Amazon Data Firehose 개발자 안내서 AWS Secrets Manager 의 Amazon Data Firehose에서 로 인증을 참조하세요.

에서를 AWS DataSync 사용하는 방법 AWS Secrets Manager

AWS DataSync 는 스토리지 시스템과 서비스 간의 데이터 이동을 간소화, 자동화 및 가속화하는 온라인 데이터 전송 서비스입니다.

DataSync에서 지원하는 일부 스토리지 시스템은 데이터 읽기 및 쓰기를 위해 자격 증명이 필요합니다. DataSync는 Secrets Manager를 사용하여 스토리지 자격 증명을 저장하거나 액세스합니다. DataSync가 사용자를 대신해 보안 암호를 생성하도록 구성하거나, 사용자 지정 보안 암호를 제공할 수 있습니다. 서비스 관리형 보안 암호는 `aws-datasync` 접두사로 시작합니다. DataSync 외부에서 사용자가 생성한 보안 암호에 대해서만 요금이 부과됩니다. 자세한 내용은 AWS DataSync 사용 설명서의 스토리지 위치에 대한 자격 증명 제공을 참조하세요.

Amazon DataZone의 사용 방식 AWS Secrets Manager

Amazon DataZone은 데이터를 카탈로깅, 검색, 관리, 공유 및 분석할 수 있는 데이터 관리 서비스입니다. AWS Glue 크롤러 작업을 사용하여 크롤링되는 Amazon Redshift 클러스터의 테이블 및 뷰에

서 데이터 자산을 사용할 수 있습니다. Amazon Redshift에 연결하려면 Secrets Manager 보안 암호에서 Amazon DataZone 보안 인증 정보를 제공합니다. 자세한 내용은 Amazon DataZone 사용 설명서의 새 AWS Glue 연결을 사용하여 Amazon Redshift 데이터베이스의 데이터 소스 생성을 참조하세요. DataZone

에서를 AWS Direct Connect 사용하는 방법 AWS Secrets Manager

Direct Connect 는 표준 이더넷 광섬유 케이블을 통해 내부 네트워크를 Direct Connect 위치에 연결합니다. 이 연결을 사용하면 퍼블릭에 직접 가상 인터페이스를 생성할 수 있습니다 AWS 서비스.

Direct Connect 는 접두사가 인 관리형 보안 암호에 연결 연결 키 이름 및 연결 연결 키 페어(CKN/CAK 페어)를 저장합니다`directconnect`. 보안 암호 비용은 요금에 포함됩니다 Direct Connect. 보안 암호를 업데이트하려면 Secrets Manager Direct Connect 대신를 사용해야 합니다. 자세한 내용은 Direct Connect 사용 설명서 Associate a MACsec CKN/CAK with a LAG(MACsec CKN/CAK와 LAG 연결)을 참조하세요.

에서를 AWS Directory Service 사용하는 방법 AWS Secrets Manager

Directory Service 는 Microsoft Active Directory(AD)를 다른 AWS 서비스와 함께 사용하는 여러 가지 방법을 제공합니다. 보안 인증에 보안 암호를 사용하여 Amazon EC2 인스턴스를 디렉터리에 조인할 수 있습니다. 자세한 정보는 Direct Connect 사용 설명서의 다음 섹션을 참조하세요:

- Linux EC2 인스턴스를 AWS 관리형 Microsoft AD 디렉터리에 원활하게 조인
- AD Connector 디렉터리에 Linux EC2 인스턴스 원활하게 조인
- Simple AD 디렉터리에 Linux EC2 인스턴스 원활하게 조인

Amazon DocumentDB(MongoDB 호환)의 사용 방식 AWS Secrets Manager

Amazon DocumentDB(MongoDB 호환)는 MongoDB 워크로드를 지원하는 완전 관리형 문서형 데이터베이스 서비스입니다. Amazon DocumentDB는 Secrets Manager와 통합되어 클러스터의 기본 사용자 암호를 관리하며, 이를 통해 보안을 강화하고 자격 증명 관리를 간소화합니다.

Amazon DocumentDB는 암호를 생성하여 Secrets Manager에 저장하고, 보안 암호 설정을 관리합니다. 기본적으로 Amazon DocumentDB는 보안 암호를 7일마다 자동으로 교체하지만, 필요에 따라 교체 일정을 수정할 수 있습니다. Amazon DocumentDB 클러스터를 생성하거나 수정할 때, 기본 사용자 암호를 Secrets Manager에서 관리하도록 지정할 수 있습니다. 자세한 내용은 Amazon DocumentDB 개발자 가이드의 Amazon DocumentDB와 Secrets Manager를 통한 암호 관리를 참조하세요.

에서를 AWS Elastic Beanstalk 사용하는 방법 AWS Secrets Manager

를 사용하면 애플리케이션을 실행하는 인프라에 대해 알 필요 없이 AWS 클라우드에서 애플리케이션을 빠르게 배포하고 관리할 AWS Elastic Beanstalk수 있습니다. Elastic Beanstalk는 Dockerfile에 명시된 이미지를 빌드하거나 원격 Docker 이미지를 가져와 Docker 환경을 시작할 수 있습니다. Elastic Beanstalk는 개인 리포지토리를 호스팅하는 온라인 레지스트리를 사용하여 인증하는 데 Secrets Manager 보안 암호를 사용합니다. 자세한 내용은 AWS Elastic Beanstalk 개발자 가이드에서 Docker 구성을 참조하세요.

Amazon Elastic Container Registry의 사용 방식 AWS Secrets Manager

Amazon Elastic Container Registry(Amazon ECR)는 안전하고 확장 가능하며 안정적인 AWS 관리형 컨테이너 이미지 레지스트리 서비스입니다. Docker CLI를 사용하거나 선호하는 클라이언트를 사용하여 이미지를 리포지토리로 푸시 및 풀링할 수 있습니다. Amazon ECR 프라이빗 레지스트리에서 캐시하려는 이미지가 포함된 각 업스트림 레지스트리에 대해 풀스루 캐시 규칙을 생성해야 합니다. 인증이 필요한 업스트림 레지스트리의 경우 보안 인증 정보를 Secrets Manager 보안 암호로 저장해야 합니다. Amazon ECR 또는 Secrets Manager 콘솔에서 Secrets Manager 보안 암호를 생성할 수 있습니다. 자세한 내용은 Amazon ECR 사용 설명서에서 풀스루 캐시 규칙 생성을 참조하세요.

Amazon Elastic Container Service

Amazon Elastic Container Service(Amazon ECS)는 컨테이너 애플리케이션을 쉽게 배포, 관리 및 확대할 수 있도록 도와주는 완전 관리형 컨테이너 오케스트레이션 서비스입니다. Secrets Manager 보안 암호를 참조하여 민감한 데이터를 컨테이너에 주입할 수 있습니다. 자세한 내용은 Amazon Elastic Container Service 개발자 가이드의 다음 페이지를 참조하세요.

- 자습서: Secrets Manager 보안 암호를 사용해 민감한 데이터 지정
- 애플리케이션을 통해 프로그래밍 방식으로 보안 암호 검색
- 환경 변수를 통해 보안 암호 검색
- 로깅 구성을 위한 보안 암호 검색

Amazon ECS는 컨테이너용 Windows File Server 볼륨용 FSx를 지원합니다. Amazon ECS는 Secrets Manager 보안 암호에 저장된 보안 인증 정보를 사용하여 Active Directory에 도메인 가입을 하고 FSx for Windows File Server 파일 시스템을 연결합니다. 자세한 내용은 Amazon Elastic Container Service 개발자 가이드의 자습서: Amazon ECS에서 FSx for Windows File Server 파일 시스템 사용 및 FSx for Windows File Server 볼륨을 참조하세요.

Secrets Manager 보안 암호를 레지스트리 자격 증명과 함께 사용하여 인증 AWS 이 필요한 외부 프라이빗 레지스트리의 컨테이너 이미지를 참조할 수 있습니다. 자세한 내용을 알아보려면 Amazon Elastic Container Service 개발자 가이드의 태스크에 대한 프라이빗 레지스트리 인증을 참조하세요.

Amazon ECS Service Connect를 사용하는 경우 Amazon ECS는 Secrets Manager 관리형 보안 암호를 사용하여 AWS Private Certificate Authority TLS 인증서를 저장합니다. 보안 암호를 저장하는 비용은 Amazon ECS 요금에 포함됩니다. 보안 암호를 업데이트하려면 Secrets Manager 대신 Amazon ECS를 사용해야 합니다. 자세한 내용은 Amazon Elastic Container Service 개발자 가이드의 Service Connect를 활용한 TLS를 참조하세요.

Amazon ElastiCache의 사용 방식 AWS Secrets Manager

ElastiCache에서는 역할 기반 액세스 제어(RBAC)라는 기능을 사용하여 클러스터를 보호할 수 있습니다. 이러한 보안 인증 정보는 Secrets Manager에 저장할 수 있습니다. Secrets Manager는 이런 유형의 보안 암호에 대한 교체 템플릿을 제공합니다. 자세한 내용은 Amazon ElastiCache 사용 설명서의 사용자의 암호 자동 교체를 참조하세요.

에서를 AWS Elemental Live 사용하는 방법 AWS Secrets Manager

AWS Elemental Live 는 브로드캐스트 및 스트리밍 전송을 위한 라이브 출력을 생성할 수 있는 실시간 비디오 서비스입니다.

AWS Elemental Live 는 보안 암호 ARN을 사용하여 Secrets Manager에서 암호화 키가 포함된 보안 암호를 가져옵니다. Elemental 라이브는 암호화 키를 사용하여 비디오를 암호화/해독합니다. 자세한 내용은 Elemental Live 사용 설명서의 런타임에에서 MediaConnect AWS Elemental Live 로 전송하는 방법을 참조하세요.

에서를 AWS Elemental MediaConnect 사용하는 방법 AWS Secrets Manager

AWS Elemental MediaConnect 는 방송사 및 기타 프리미엄 비디오 공급자가에 라이브 비디오를 안정적으로 수집하여 내부 또는 외부의 여러 대상에 AWS 클라우드 배포할 수 있도록 해주는 서비스입니다 AWS 클라우드.

정적 키 암호화를 사용하여 소스, 출력, 권한 부여를 보호할 수 있으며 암호화 키를 AWS Secrets Manager에 저장할 수 있습니다. 자세한 내용은 AWS Elemental MediaConnect 사용 설명서의 Static key encryption in AWS Elemental MediaConnect 섹션을 참조하세요.

에서를 AWS Elemental MediaConvert 사용하는 방법 AWS Secrets Manager

AWS Elemental MediaConvert 는 모든 크기의 미디어 라이브러리가 있는 콘텐츠 소유자 및 배포자에게 확장 가능한 비디오 처리를 제공하는 파일 기반 비디오 처리 서비스입니다. MediaConvert로 Kantar 워터마크를 인코딩하려면 Secrets Manager를 사용하여 Kantar 자격 증명을 저장합니다. 자세한 내용은 AWS Elemental MediaConvert 사용 설명서의 AWS Elemental MediaConvert 출력에서 오디오 워터마킹에 Kantar 사용을 참조하세요.

에서를 AWS Elemental MediaLive 사용하는 방법 AWS Secrets Manager

AWS Elemental MediaLive 는 브로드캐스트 및 스트리밍 전송을 위한 라이브 출력을 생성할 수 있는 실시간 비디오 서비스입니다. 조직에서 AWS Elemental MediaLive 또는와 함께 AWS Elemental Link 디바이스를 사용하는 AWS Elemental MediaConnect경우 디바이스를 배포하고 디바이스를 구성해야 합니다. MediaLive 사용 설명서의 Setting up MediaLive as a trusted entity 섹션을 참조하세요.

에서를 AWS Elemental MediaPackage 사용하는 방법 AWS Secrets Manager

AWS Elemental MediaPackage 는에서 실행되는 just-in-time 비디오 패키징 및 발신 서비스입니다 AWS 클라우드. MediaPackage를 사용하면 매우 안전하고 확장 가능하며 안정적인 비디오 스트림을 다양한 재생 디바이스 및 콘텐츠 전송 네트워크(CDN)에 전달할 수 있습니다. 자세한 내용은 AWS Elemental MediaPackage 사용 설명서의 Secrets Manager access for CDN authorization을 참조하세요.

에서를 AWS Elemental MediaTailor 사용하는 방법 AWS Secrets Manager

AWS Elemental MediaTailor 는에서 실행되는 확장 가능한 광고 삽입 및 채널 어셈블리 서비스입니다 AWS 클라우드.

MediaTailor는 소스 위치에 Secrets Manager 액세스 토큰 인증을 지원합니다. Secrets Manager 액세스 토큰 인증으로 MediaTailor는 Secrets Manager 보안 암호를 사용하여 오리진에 대한 요청을 인증합니다. 자세한 내용은 AWS Elemental MediaTailor 사용 설명서의 AWS Secrets Manager 액세스 토큰 인증 구성을 참조하세요.

Amazon EMR에서 Secrets Manager를 사용하는 방법

Amazon EMR은 Apache Hadoop 및 Apache Spark와 같은 빅 데이터 프레임워크 실행을 간소화 AWS 하여 방대한 양의 데이터를 처리하고 분석하는 플랫폼입니다. 이러한 프레임워크와 함께 Apache Hive

및 Apache Pig와 같은 관련 오픈 소스 프로젝트를 사용하는 경우, 분석용 데이터와 비즈니스 인텔리전스 워크로드를 처리할 수 있습니다. Amazon EMR을 사용하여 Amazon S3 및 Amazon DynamoDB와 같은 다른 데이터 스토어 및 데이터베이스 안팎으로 대량의 AWS 데이터를 변환하고 이동할 수도 있습니다.

Amazon EC2에서 실행되는 Amazon EMR에서 Secrets Manager를 사용하는 방법

Amazon EMR에 클러스터를 만들 때 Secrets Manager 보안 암호를 사용하여 클러스터에 응용 프로그램 구성 데이터를 제공할 수 있습니다. 자세한 내용은 Amazon EMR 관리 가이드의 Secrets Manager에 중요 데이터 보관을 참조하세요.

또한 Secrets Manager를 사용하여 EMR Notebook을 생성하고 프라이빗 Git 기반 레지스트리 보안 인증 정보를 저장할 수 있습니다. 자세한 내용은 Amazon EMR 관리 가이드의 Amazon EMR에 Git 기반 리포지토리 추가를 참조하세요.

EMR Serverless에서 Secrets Manager를 사용하는 방법

EMR Serverless는 분석 애플리케이션 운영을 간소화하는 서버리스 런타임 환경을 제공하므로 클러스터를 구성하거나 최적화, 보호, 운영할 필요가 없습니다.

에 데이터를 저장 AWS Secrets Manager 한 다음 EMR Serverless 구성에서 보안 암호 ID를 사용할 수 있습니다. 이렇게 하면 민감한 구성 데이터를 일반 텍스트로 전달하여 외부 API에 노출하지 않아도 됩니다.

자세한 내용은 Amazon EMR Serverless 사용 설명서에서 EMR Serverless를 이용한 데이터 보호를 위한 Secrets Manager를 참조하세요.

Amazon EventBridge의 사용 방식 AWS Secrets Manager

Amazon EventBridge는 애플리케이션을 다양한 소스의 데이터와 연결하는 데 사용할 수 있는 서버리스 이벤트 버스 서비스입니다.

Amazon EventBridge API 데스티네이션을 만들 때는 EventBridge가 이에 대한 연결을 `events` 접두사가 붙은 Secrets Manager 관리형 보안 암호에 저장합니다. 보안 암호 저장 비용은 API 대상 사용 요금에 포함됩니다. 보안 암호를 업데이트하려면 Secrets Manager 대신 EventBridge를 사용해야 합니다. 자세한 내용은 Amazon EventBridge 사용 설명서의 API 대상을 참조하세요.

Amazon FSx에서 AWS Secrets Manager 보안 암호를 사용하는 방법

Amazon FSx for Windows File Server는 완전한 네이티브 Windows 파일 시스템이 지원하는 완전관리형 Microsoft Windows 파일 서버를 제공합니다. 파일 공유를 생성하거나 관리할 때 AWS Secrets

Manager 보안 암호에서 자격 증명을 전달할 수 있습니다. 자세한 내용은 Amazon FSx for Windows File Server 사용 설명서의 파일 공유 및 Amazon FSx로 파일 공유 구성 마이그레이션을 참조하세요.

에서를 AWS Glue DataBrew 사용하는 방법 AWS Secrets Manager

AWS Glue DataBrew 는 코드를 작성하지 않고도 데이터를 정리하고 정규화하는 데 사용할 수 있는 시각적 데이터 준비 도구입니다. DataBrew에서는 일련의 데이터 변환 단계를 레시피라고 합니다.는 Secrets Manager 보안 암호에 저장된 암호화 키를 사용하는 데이터 세트의 개인 식별 정보(PII)에 대한 변환을 수행하기 위한 , `DETERMINISTIC_DECRYPT` `DETERMINISTIC_ENCRYPT`및 `CRYPTOGRAPHIC_HASH` 레시피 단계를 AWS Glue DataBrew 제공합니다. 암호화 키를 저장하기 위해 DataBrew 기본 보안 암호를 사용하는 경우 DataBrew는 `databrew` 접두사가 붙은 관리형 보안 암호를 생성합니다. 보안 암호 저장 비용은 DataBrew 사용 요금에 포함됩니다. 암호화 키를 저장하기 위해 새 보안 기본 암호를 생성하는 경우 DataBrew는 `AwsGlueDataBrew` 접두사가 붙은 보안 암호를 생성합니다. 보안 암호에 대한 요금이 청구됩니다.

AWS Glue Studio에서를 사용하는 방법 AWS Secrets Manager

AWS Glue Studio 는 ETL(추출, 변환 및 로드) 작업을 쉽게 생성, 실행 및 모니터링할 수 있는 그래픽 인터페이스입니다 AWS Glue. AWS Glue Studio에서 Elasticsearch Spark Connector를 구성하여 Amazon OpenSearch Service를 추출, 전환, 적재(ETL) 작업의 데이터 스토어로 사용할 수 있습니다. OpenSearch 클러스터에 연결하기 위해 Secrets Manager에서 보안 암호를 사용할 수 있습니다. 자세한 내용은 AWS Glue 개발자 가이드의 자습서: Elasticsearch용 AWS Glue 커넥터 사용을 참조하세요.

에서를 AWS IoT SiteWise 사용하는 방법 AWS Secrets Manager

AWS IoT SiteWise 는 대규모 산업 장비에서 데이터를 수집, 모델링, 분석 및 시각화할 수 있는 관리형 서비스입니다. AWS IoT SiteWise 콘솔을 사용하여 게이트웨이를 생성할 수 있습니다. 그런 다음 게이트웨이에 연결된 데이터 원본, 로컬 서버 또는 산업 장비를 추가합니다. 소스에 인증이 필요할 경우 보안 암호를 사용하여 인증합니다. 자세한 내용은 AWS IoT SiteWise 사용 설명서의 데이터 소스 인증 구성을 참조하세요.

Amazon Kendra의 사용 방식 AWS Secrets Manager

Amazon Kendra는 사용자가 자연어 처리 및 고급 검색 알고리즘을 사용하여 구조화되지 않은 데이터와 구조화된 데이터를 검색할 수 있도록 해주는 매우 정확하고 지능적인 검색 서비스입니다.

해당 데이터베이스의 자격 증명을 포함한 보안 암호를 지정하여 데이터베이스에 저장된 문서를 인덱싱할 수 있습니다. 자세한 내용은 Amazon Kendra 사용 설명서의 데이터베이스 데이터 소스 사용을 참조하세요.

Amazon Kinesis Video Streams의 사용 방식 AWS Secrets Manager

Amazon Kinesis Video Streams를 사용하여 고객의 온프레미스 IP 카메라에 연결하고, 카메라의 비디오를 로컬로 녹화 및 저장하며, 장기간 저장, 재생 및 분석 처리를 위해 비디오를 클라우드로 스트리밍할 수 있습니다. IP 카메라에서 미디어를 녹화하고 업로드하려면 Kinesis Video Streams Edge Agent를 AWS IoT Greengrass에 배포합니다. 카메라로 스트리밍되는 미디어 파일에 액세스하는 데 필요한 보안 인증 정보를 Secrets Manager 보안 암호에 저장합니다. 자세한 내용은 Amazon Kinesis Video Streams 개발자 가이드에서 AWS IoT Greengrass에 Amazon Kinesis Video Streams Edge Agent 배포를 참조하세요.

에서를 AWS Launch Wizard 사용하는 방법 AWS Secrets Manager

AWS Launch Wizard Active Directory용는 AWS 클라우드 애플리케이션 모범 사례를 적용하여 또는 AWS 클라우드 온프레미스에서 새 Active Directory 인프라를 설정하거나 기존 인프라에 도메인 컨트롤러를 추가하는 방법을 안내하는 서비스입니다.

AWS Launch Wizard 에서는 도메인 컨트롤러를 Active Directory에 조인하기 위해 Secrets Manager에 도메인 관리자 자격 증명을 추가해야 합니다. 자세한 내용은 AWS Launch Wizard 사용 설명서의 Active Directory를 위한 AWS Launch Wizard 설정을 참조하세요.

Amazon Lookout for Metrics의 사용 방식 AWS Secrets Manager

Amazon Lookout for Metrics는 데이터에서 이상을 찾고 근본 원인을 파악하여 신속하게 조치를 취하는 서비스입니다. Amazon Redshift 또는 Amazon RDS를 Lookout for Metrics 감지기의 데이터 원본으로 사용할 수 있습니다. 데이터 원본을 구성하는 데 데이터베이스 암호가 포함된 보안 암호를 사용할 수 있습니다. 자세한 내용은 Amazon Lookout for Metrics 개발자 가이드의 Lookout for Metrics와 함께 Amazon RDS 사용 및 Lookout for Metrics와 함께 Amazon Redshift 사용을 참조하세요.

Amazon Managed Grafana의 사용 방식 AWS Secrets Manager

Amazon Managed Grafana는 여러 소스의 운영 지표, 로그 및 추적을 즉시 쿼리, 상관관계 파악 및 시각화하는 데 사용할 수 있는 완전 관리형의 보안 데이터 시각화 서비스입니다. Amazon Redshift를 데이터 소스로 사용하는 경우 보안 AWS Secrets Manager 암호를 사용하여 Amazon Redshift 자격 증명을 제공할 수 있습니다. 자세한 내용은 Amazon Managed Grafana 사용 설명서의 Amazon Redshift 구성 섹션을 참조하세요.

에서를 AWS Managed Services 사용하는 방법 AWS Secrets Manager

AWS Managed Services 는 인프라를 지속적으로 관리하는 엔터프라이즈 서비스입니다 AWS . AMS 셀프 서비스 프로비저닝(SSP) 모드는 AMS 관리형 계정의 기본 AWS 서비스 및 API 기능에 대한 전체 액세스를 제공합니다. AMS에서 Secrets Manager에 대한 액세스를 요청하는 방법에 대한 자세한 내용은 AMS Advanced User Guide에서 AWS Secrets Manager (AMS Self-Service Provisioning)을 참조하세요.

Amazon Managed Streaming for Apache Kafka의 사용 방식 AWS Secrets Manager

Amazon Managed Streaming for Apache Kafka(Amazon MSK)는 Apache Kafka를 사용하여 스트리밍 데이터를 처리하는 애플리케이션의 구축 및 실행을 위해 사용할 수 있는 완전관리형 서비스입니다. AWS Secrets Manager을 사용하여 저장되고 보호되는 사용자 이름 및 암호를 사용하여 Amazon MSK 클러스터에 대한 액세스를 제어할 수 있습니다. 자세한 내용은 Amazon Managed Streaming for Apache Kafka 개발자 가이드의 AWS Secrets Manager를 사용한 사용자 이름 및 암호 인증을 참조하세요.

Amazon Managed Workflows for Apache Airflow의 사용 방식 AWS Secrets Manager

Amazon Managed Workflows for Apache Airflow는 Apache Airflow에 대한 관리형 오케스트레이션 서비스로 클라우드에서 종단 간 데이터 파이프라인을 대규모로 쉽게 설정하고 운영할 수 있습니다.

Secrets Manager 보안 암호를 사용하여 Apache Airflow 연결을 구성할 수 있습니다. 자세한 내용은 Amazon Managed Workflows for Apache Airflow 사용 설명서의 Secrets Manager 보안 암호를 사용하여 Apache Airflow 연결 구성 및 AWS Secrets Manager Apache Airflow 변수에 대한의 보안 키 사용을 참조하세요.

AWS Marketplace

AWS Marketplace Quick Launch를 사용하면가 소프트웨어를 라이선스 키와 함께 AWS Marketplace 배포합니다.는 계정에 라이선스 키를 Secrets Manager 관리형 보안 암호로 AWS Marketplace 저장합니다. 보안 암호 저장 비용은 요금에 포함됩니다 AWS Marketplace. 보안 암호를 업데이트하려면 Secrets Manager AWS Marketplace 대신를 사용해야 합니다. 자세한 내용은 AWS Marketplace 판매자 설명서의 빠른 실행 구성을 참조하세요.

에서를 AWS Migration Hub 사용하는 방법 AWS Secrets Manager

AWS Migration Hub 는 여러 AWS 도구 및 파트너 솔루션에서 마이그레이션 작업을 추적할 수 있는 단일 위치를 제공합니다.

AWS Migration Hub 오케스트레이터는 서버 및 엔터프라이즈 애플리케이션의 마이그레이션을 간소화하고 자동화합니다 AWS. Migration Hub Orchestrator 는 소스 서버에 대한 연결 정보에 보안 암호를 사용합니다. 자세한 내용은 AWS Migration Hub 오케스트레이터 사용 설명서에서 사용 방법을 참조하세요:

- SAP NetWeaver 애플리케이션을 로 마이그레이션 AWS
- Amazon EC2에서 애플리케이션 리호스팅

Migration Hub 전략 권장 사항은 애플리케이션 변환을 실행할 수 있는 경로를 제공할 마이그레이션 및 현대화 전략 권장 사항을 제공합니다. 전략 권장 사항은 연결 정보에 대한 보안 암호를 사용하여 SQL Server 데이터베이스를 분석할 수 있습니다. 자세한 내용은 전략 권장 사항 데이터베이스 분석을 참조하세요.

에서 Secrets Manager를 AWS Panorama 사용하는 방법

AWS Panorama 는 온프레미스 카메라 네트워크에 컴퓨터 비전을 제공하는 서비스입니다. AWS Panorama 를 사용하여 어플라이언스를 등록하고, 소프트웨어를 업데이트하고, 어플라이언스에 애플리케이션을 배포합니다. 비디오 스트림을 애플리케이션의 데이터 소스로 등록할 때 스트림이 암호로 보호되어 있으면 AWS Panorama 에서 해당 보안 인증 정보를 Secrets Manager 보안 암호에 저장합니다. 자세한 내용은 AWS Panorama 개발자 가이드의 AWS Panorama에서 카메라 스트림 관리를 참조하세요.

AWS 병렬 컴퓨팅 서비스의 사용 방식 AWS Secrets Manager

AWS 병렬 컴퓨팅 서비스(AWS PCS)는 HPC(고성능 컴퓨팅) 및 분산 기계 학습 워크로드를 더 쉽게 실행하고 확장할 수 있는 관리형 서비스입니다 AWS.

클러스터 작업 스케줄러에 연결하기 위해 AWS PCS는 스케줄러 키를 `pcs` 저장할 접두사가 있는 관리형 보안 암호를 생성합니다. 보안 암호 저장 비용은 AWS PCS에 대한 요금과 함께 포함됩니다. AWS PCS는 AWS PCS 클러스터를 삭제할 때 보안 암호를 자동으로 삭제합니다. 자세한 내용은 AWS PCS 사용 설명서의 PCS에서 클러스터 보안 암호 작업을 참조하세요. AWS

> **Important**
>
> AWS PCS 클러스터 보안 암호를 수정하거나 삭제하지 마십시오.

에서를 AWS ParallelCluster 사용하는 방법 AWS Secrets Manager

AWS ParallelCluster 는에서 고성능 컴퓨팅(HPC) 클러스터를 배포하고 관리하는 데 사용할 수 있는 오픈 소스 클러스터 관리 도구입니다 AWS 클라우드. AWS 관리형 Microsoft AD(Active Directory)와 통합된 AWS ParallelCluster 가 포함된 여러 사용자 환경을 생성할 수 있습니다. 는 Secrets Manager 보안 암호를 AWS ParallelCluster 사용하여 Active Directory에 대한 로그인을 검증합니다. 자세한 내용은 AWS ParallelCluster 사용 설명서에서 Active Directory 통합을 참조하세요.

Amazon Q의 Secrets Manager 활용 방식

데이터 소스에 액세스하기 위해 Amazon Q를 인증하려면 Secrets Manager 보안 암호를 사용하여 Amazon Q에 데이터 소스 액세스 자격 증명을 제공합니다. 콘솔을 사용하는 경우 새 보안 암호를 생성하거나, 기존 보안 암호를 사용하는 옵션 중에서 선택할 수 있습니다. 자세한 내용은 Amazon Q Developer 가이드서의 개념 - 인증 섹션을 참조하세요.

Amazon OpenSearch Ingestion이 Secrets Manager를 사용하는 방법

Amazon OpenSearch Ingestion은 완전 관리형 서버리스 데이터 수집기로, 실시간 로그, 메트릭, 추적 데이터를 Amazon OpenSearch Service 도메인과 Serverless 컬렉션으로 스트리밍합니다. Secrets Manager와 함께 OpenSearch Ingestion 파이프라인을 사용하여 자격 증명을 안전하게 관리할 수 있습니다. 자세한 내용은 다음을 참조하세요.

- Atlassian Services와 함께 Ingestion 파이프라인 사용
- Amazon DocumentDB와 함께 Ingestion 파이프라인 사용
- Confluent Cloud Kafka와 함께 Ingestion 파이프라인 사용
- Kafka와 함께 Ingestion 파이프라인 사용
- Amazon OpenSearch Ingestion을 사용하여 자체 관리 OpenSearch 클러스터에서 데이터 마이그레이션

에서를 AWS OpsWorks for Chef Automate 사용하는 방법 AWS Secrets Manager

OpsWorks 는 OpsWorks for Puppet Enterprise 또는를 사용하여 클라우드 엔터프라이즈에서 애플리케이션을 구성하고 운영하는 데 도움이 되는 구성 관리 서비스입니다 AWS OpsWorks for Chef Automate.

에서 새 서버를 생성하면 AWS OpsWorks CM OpsWorks CM은 서버 정보를 접두사가 인 Secrets Manager 관리형 보안 암호에 저장합니다`opsworks-cm`. 보안 암호 비용은 OpsWorks요금에 포함됩니다. 자세한 내용은 OpsWorks 사용 설명서의 AWS Secrets Manager과 통합을 참조하세요.

Amazon Quick에서를 사용하는 방법 AWS Secrets Manager

Amazon Quick은 분석, 데이터 시각화 및 보고에 사용할 수 있는 클라우드 규모의 비즈니스 인텔리전스(BI) 서비스입니다. Quick에서 다양한 데이터 소스를 사용할 수 있습니다. Secrets Manager 보안 암호에 데이터베이스 자격 증명을 저장하는 경우 Quick은 이러한 보안 암호를 사용하여 데이터베이스에 연결할 수 있습니다. 자세한 내용은 Amazon Quick 사용 설명서의 Amazon Quick에서 데이터베이스 자격 증명 대신 보안 암호 사용을 AWS Secrets Manager 참조하세요.

Amazon RDS의 사용 방식 AWS Secrets Manager

Amazon Relational Database Service(Amazon RDS)는 AWS 클라우드에서 관계형 데이터베이스를 더 쉽게 설치, 운영 및 확장할 수 있는 웹 서비스입니다.

Aurora를 포함한 Amazon Relational Database Service(Amazon RDS)의 마스터 사용자 자격 증명을 관리하기 위해 Amazon RDS는 관리형 보안 암호를 생성할 수 있습니다. 보안 암호에 대한 요금이 청구됩니다. 또한 Amazon RDS는 이러한 보안 인증 정보의 교체를 관리합니다. 자세한 내용은 Amazon RDS 사용 설명서의 Amazon RDS 및 AWS Secrets Manager을 사용한 암호 관리를 참조하세요.

다른 Amazon RDS 보안 인증에 대한 자세한 내용은 보안 암호 생성을 참조하세요.

Amazon RDS 쿼리 편집기를 사용하여 데이터베이스에 연결하면 데이터 베이스에 대한 보안 인증 정보를 Secrets Manager에 저장할 수 있습니다. 자세한 내용은 Amazon RDS 사용 설명서의 쿼리 편집기 사용을 참조하세요.

Amazon Redshift의 사용 방식 AWS Secrets Manager

Amazon Redshift는 클라우드에서 완전히 관리되는 페타바이트급 데이터 웨어하우스 서비스입니다.

Amazon Redshift의 관리자 보안 인증 정보를 관리하기 위해 Amazon Redshift는 사용자를 대신하여 관리형 보안 암호를 생성할 수 있습니다. 보안 암호에 대한 요금이 청구됩니다. 또한, Amazon Redshift는 이러한 자격 증명의 교체를 관리합니다. 자세한 내용은 Amazon Redshift 관리 가이드의 AWS Secrets Manager를 사용한 Amazon Redshift 관리자 암호 관리를 참조하세요.

다른 Amazon Redshift 보안 인증의 경우, 보안 암호 생성을(를) 참조하세요.

Amazon Redshift 데이터 API를 호출할 때 Secrets Manager 보안 암호를 사용하여 클러스터에 대한 자격 증명을 전달할 수 있습니다. 자세한 내용은 Amazon Redshift 데이터 API 사용을 참조하세요.

Amazon RDS 쿼리 편집기를 사용하여 데이터베이스에 연결하면 Amazon Redshift는 사용자 보안 인증 정보를 `redshiftqueryeditor` 접두사가 붙은 Secrets Manager 보안 암호에 저장할 수 있습니다. 보안 암호에 대한 요금이 청구됩니다. 자세한 정보는 Amazon Redshift 관리 가이드의 쿼리 편집기를 사용하여 데이터베이스 쿼리를 참조하세요.

쿼리 편집기 v2에 대해서는 the section called “Amazon Redshift Query Editor V2” 섹션을 참조하세요.

Amazon Redshift Query Editor V2

Amazon Redshift 쿼리 편집기 v2는 Amazon Redshift 데이터 웨어하우스에서 쿼리를 작성하고 실행하는 데 사용하는 웹 기반 SQL 클라이언트 애플리케이션입니다. Amazon Redshift Query Editor V2를 사용하여 데이터베이스에 연결하면 Amazon Redshift는 사용자 자격 증명을 `sqlworkbench` 접두사가 붙은 Secrets Manager 관리형 보안 암호에 저장할 수 있습니다. 보안 암호 저장 비용은 Amazon Redshift 사용 요금에 포함됩니다. 보안 암호를 업데이트하려면 Secrets Manager 대신 Amazon Redshift를 사용해야 합니다. 자세한 정보는 Amazon Redshift 관리 가이드의 쿼리 편집기 v2 작업을 참조하세요.

이전 쿼리 편집기에 관해서는 the section called “Amazon Redshift” 섹션을 참조하세요.

Amazon SageMaker AI의 사용 방식 AWS Secrets Manager

SageMaker AI는 종합 관리형 기계 학습 서비스입니다. SageMaker AI를 통해 데이터 과학자와 개발자들은 기계 학습 모델을 빠르고 쉽게 구축하고 훈련시킬 수 있으며, 그런 다음 이들 모델을 프로덕션 지원 호스팅 환경에 직접 배포할 수 있습니다. 탐색 및 분석에 필요한 데이터 원본에 대한 쉬운 액세스를 위해 내장형 Jupyter 작성 노트북 인스턴스를 제공하기 때문에 서버를 관리할 필요가 없습니다.

Git 리포지토리를 Jupyter Notebook 인스턴스와 연결하여 노트북 인스턴스를 중지 또는 삭제하더라도 유지되는 소스 제어 환경에 노트북을 저장할 수 있습니다. Secrets Manager를 사용하여 프라이빗 리포지토리 자격 증명을 관리할 수 있습니다. 자세한 내용은 Amazon SageMaker AI 개발자 가이드의 Amazon SageMaker 노트북 인스턴스와 Git 리포지토리 연결을 참조하세요.

Databricks에서 데이터를 가져오기 위해 Data Wrangler는 Secrets Manager에 JDBC URL을 저장합니다. 자세한 내용은 Databricks 에서 데이터 가져오기(JDBC)를 참조하세요.

Snowflake에서 데이터를 가져오기 위해 Data Wrangler는 Secrets Manager 보안 암호에 자격 증명을 저장합니다. 자세한 내용은 Snowflake에서 데이터 가져오기를 참조하세요.

에서를 AWS Schema Conversion Tool 사용하는 방법 AWS Secrets Manager

AWS Schema Conversion Tool (AWS SCT)를 사용하여 기존 데이터베이스 스키마를 한 데이터베이스 엔진에서 다른 데이터베이스 엔진으로 변환할 수 있습니다. 관계형 OLTP 스키마 또는 데이터 웨어하우스 스키마를 변환할 수 있습니다. 변환된 스키마는 Amazon Relational Database Service(Amazon RDS) MySQL, MariaDB, Oracle, SQL Server, PostgreSQL DB, Amazon Aurora DB 클러스터 또는 Amazon Redshift 클러스터에 적합합니다. 변환된 스키마는 Amazon Elastic Compute Cloud 인스턴스에서 데이터베이스와 함께 사용하거나 S3 버킷에 데이터로 저장할 수 있습니다.

데이터베이스 스키마를 변환할 때는 저장한 데이터베이스 자격 증명을 사용할 AWS SCT 수 있습니다 AWS Secrets Manager. 자세한 내용은 사용 AWS Schema Conversion Tool 설명서AWS Secrets Manager 의 AWS SCT 사용자 인터페이스에서 사용을 참조하세요.

Amazon Timestream for InfluxDB의 사용 방식 AWS Secrets Manager

Timestream for InfluxDB는 관리형 시계열 데이터베이스 엔진으로, 오픈 소스 API를 사용하여 실시간 시계열 애플리케이션에 AWS 대해에서 InfluxDB 데이터베이스를 쉽게 실행할 수 있습니다. APIs Timestream for InfluxDB를 사용하면 한 자릿수 밀리초 쿼리 응답 시간으로 쿼리에 응답할 수 있는 시계열 워크로드를 설정, 작동하고 규모를 조정할 수 있습니다.

Timestream for InfluxDB 데이터베이스를 생성하면 Timestream은 관리자 보안 자격 증명을 저장하는 보안 암호를 자동으로 생성합니다. 자세한 내용은 Timestream 개발자 가이드의 Timestream for InfluxDB에서 보안 암호를 사용하는 방법을 참조하세요.

에서를 AWS Toolkit for JetBrains 사용하는 방법 AWS Secrets Manager

AWS Toolkit for JetBrains 는 JetBrains의 통합 개발 환경(IDEs)을 위한 오픈 소스 플러그인입니다. JetBrains 개발자가 도구 키트를 사용하여 AWS를 사용하는 서버리스 애플리케이션을 쉽게 개발, 디버깅, 배포할 수 있습니다. 도구 키트를 사용하여 Amazon Redshift 클러스터에 연결할 때 Secrets Manager 보안 암호를 사용하여 인증할 수 있습니다. 자세한 내용은 AWS Toolkit for JetBrains 사용 설명서의 Amazon Redshift 클러스터에 대한 액세스를 참조하세요.

에서 AWS Secrets Manager 보안 암호를 AWS Transfer Family 사용하는 방법

AWS Transfer Family 는 AWS 스토리지 서비스 내부 및 외부로 파일을 전송할 수 있는 보안 전송 서비스입니다.

Transfer Family는 이제 Applicability Statement 2(AS2) 프로토콜을 사용하는 서버에 대해 기본 인증 사용을 지원합니다. 새 Secrets Manager 보안 암호를 생성하거나 보안 인증 정보에 대한 기존 보안 암호를 선택할 수 있습니다. 자세한 내용은 AWS Transfer Family 사용 설명서의 AS2 커넥터에 대한 기본 인증을 참조하세요.

Transfer Family 사용자를 인증하기 위해를 자격 증명 공급자 AWS Secrets Manager 로 사용할 수 있습니다. 자세한 내용은 AWS Transfer Family 사용 설명서의 사용자 지정 자격 증명 공급자 작업 및 AWS Transfer Family 사용을 위한 암호 인증 활성화 AWS Secrets Manager 블로그 문서를 참조하세요.

Transfer Family가 워크플로를 통해 처리하는 파일에 프리티 굿 프라이버시(PGP) 암호 해독을 사용할 수 있습니다. 워크플로 단계에서 암호 해독을 사용하려면 Secrets Manager에서 관리하는 PGP 키를 제공해야 합니다. 자세한 내용은AWS Transfer Family 사용 설명서의 Generate and manage PGP keys(PGP 키 생성 및 관리)를 참조하세요.

에서 AWS Secrets Manager 보안 암호를 AWS Wickr사용하는 방법

AWS Wickr 는 조직과 정부 기관이 one-to-one 및 그룹 메시징, 음성 및 영상 통화, 파일 공유, 화면 공유 등을 통해 안전하게 통신할 수 있도록 지원하는 end-to-end 간 암호화 서비스입니다. Wickr 데이터 보존 봇을 사용하여 워크플로를 자동화할 수 있습니다. 봇이 액세스할 수 있는 경우 Secrets Manager 보안 암호를 생성하여 봇 자격 증명을 저장 AWS 서비스해야 합니다. 자세한 내용은 AWS Wickr 관리 가이드에서 데이터 보존 봇 시작을 참조하세요.

AWS Secrets Manager 관리형 외부 보안 암호를 사용하여 타사 보안 암호 관리

관리형 외부 보안 암호는 통합 파트너의 보안 인증을 저장하고 자동으로 교체할 수 AWS Secrets Manager 있는의 새로운 보안 암호 유형입니다. 이 기능을 사용하면 통합 파트너 보안 암호를 교체하기 위한 사용자 지정 AWS Lambda 함수를 생성하고 유지할 필요가 없습니다. 온보딩된 모든 파트너의 전체 목록은 통합 파트너를 참조하세요.

애플리케이션을 빌드할 때 워크로드는 API 키 AWS, OAuth 토큰 또는 자격 증명 페어와 같은 보안 자격 증명을 통해 타사 애플리케이션과 상호 작용해야 하는 경우가 많습니다. 이전에는 각 애플리케이션에 고유한 복잡한 교체 Lambda 함수를 구축하고 지속적인 유지 관리를 요구하는 등 이러한 자격 증명을 보호하고 관리하기 위한 사용자 지정 접근 방식을 개발해야 했습니다.

관리형 외부 보안 암호는 각 파트너가 지정한 사전 정의된 형식으로 타사 자격 증명을 저장하기 위한 표준화된 접근 방식을 제공합니다. 이 기능에는 보안 암호 생성 중에 활성화된 자동 교체(기본적으로 콘솔에서), 보안 암호 관리 워크플로에 대한 완전한 투명성 및 사용자 제어, 세분화된 권한 관리, 관찰성, 거버넌스, 규정 준수, 재해 복구, 모니터링 제어를 포함하여 Secrets Manager에서 제공하는 전체 기능 세트가 포함됩니다.

주요 기능

관리형 외부 보안 암호는 타사 자격 증명 관리를 간소화하는 몇 가지 주요 기능을 제공합니다.

- Lambda가 없는 관리형 교체는 사용자 지정 교체 함수를 생성하고 관리하는 데 드는 오버헤드를 제거합니다. 외부를 생성하면 계정에 배포된 Lambda 함수 없이 교체가 자동으로 활성화됩니다.
- 미리 정의된 보안 암호 형식을 사용하면 보안 암호가 통합 파트너와 올바르게 연결되고 교체에 필요한 메타데이터를 포함할 수 있습니다. 각 파트너는 필요한 형식을 정의합니다.
- 통합 파트너 에코시스템은 표준화된 온보딩 프로세스를 통해 여러 파트너를 지원합니다. 파트너는 Secrets Manager와 직접 통합되어 보안 암호 생성 및 관리형 교체 기능에 대한 프로그래밍 지침을 제공합니다.
- 완전한 감사 가능성은 모든 교체 활동, 보안 암호 값 업데이트 및 관리 작업에 대한 AWS CloudTrail 로깅을 통해 완전한 투명성을 유지합니다.

관리형 외부 보안 암호 파트너

Secrets Manager는 기본적으로 타사 애플리케이션과 통합되어 파트너가 보유한 보안 암호를 교체합니다. 각 파트너는 보안 암호를 교체하는 데 필요한 메타데이터 및 보안 암호 값 필드를 정의합니다.

보안 암호 값에는 타사 클라이언트와 연결하는 데 필요하고 CreateSecret 호출 중에 저장되는 필드가 포함됩니다. 교체 메타데이터에는 교체 중에 보안 암호를 업데이트하는 데 사용되는 필드와 RotateSecret 호출에 사용되는 필드가 들어 있습니다. 이러한 필드는 관리형 교체 흐름을 허용하도록 통합 파트너가 정의합니다.

교체가 제대로 작동하려면 Secrets Manager에 보안 암호 수명 주기를 관리할 수 있는 특정 권한을 제공해야 합니다. 자세한 내용은 보안 및 권한을 참조하세요.

다음 주제에는 보안 암호를 교체하는 데 필요한 각 메타데이터 필드에 대한 설명과 교체하는 데 필요한 Secrets Manager 보안 암호에 필요한 각 필드에 대한 설명이 포함되어 있습니다.

주제

통합 파트너	보안 암호 유형
Salesforce	SalesforceClientSecret
BigID	BigIDClientSecret
Snowflake	SnowflakeKeyPairAuthentication

Salesforce 클라이언트 보안 암호

보안 암호 값 필드

다음은 Secrets Manager 보안 암호에 포함되어야 하는 필드입니다.

```
{
  "consumerKey": "client ID",
  "consumerSecret": "client secret",
  "baseUri": "https://domain.my.salesforce.com",
  "appId": "app ID",
  "consumerId": "consumer ID"
}
```

consumerKey

클라이언트 ID라고도 하는 소비자 키는 OAuth 2.0 자격 증명의 자격 증명 식별자입니다. Salesforce 외부 클라이언트 앱 관리자 OAuth 설정에서 직접 소비자 키를 검색할 수 있습니다.

consumerSecret

클라이언트 보안 암호라고도 하는 소비자 보안 암호는 OAuth 2.0 클라이언트 보안 인증 흐름을 사용하여 인증하기 위해 소비자 키와 함께 사용되는 프라이빗 암호입니다. Salesforce 외부 클라이언트 앱 관리자 OAuth 설정에서 직접 소비자 암호를 검색할 수 있습니다.

baseUri

기본 URI는 Salesforce APIs. 이는 예의 형태를 취합니다`https://domainName.my.salesforce.com`.

appId

앱 ID는 Salesforce 외부 클라이언트 애플리케이션(ECA)의 식별자입니다. Salesforce OAuth 사용량 엔드포인트를 호출하여 이를 검색할 수 있습니다. 로 시작하고 영숫자 문자만 `0x` 포함해야 합니다. 이 필드는 Salesforce 교체 안내서의 external_client_app_identifier를 참조합니다.

consumerId

소비자 ID는 Salesforce 외부 클라이언트 애플리케이션(ECA) 소비자의 식별자입니다. 앱 ID 엔드포인트별 Salesforce OAuth 자격 증명을 호출하여 이를 검색할 수 있습니다. 이 필드는 Salesforce 교체 가이드의 consumer_id를 참조합니다.

보안 암호 메타데이터 필드

다음은 Salesforce가 보유한 보안 암호를 교체하는 데 필요한 메타데이터 필드입니다.

```
{
  "apiVersion": "v65.0",
  "adminSecretArn": "arn:aws:secretsmanager:us-
east-1:111122223333:secret:SalesforceClientSecret"
}
```

apiVersion

Salesforce API 버전은 Salesforce 조직의 API 버전입니다. 버전은 v65.0 이상이어야 합니다. 가 숫자 문자`vXX.XX`인 형식이어야 합니다.

adminSecretArn

(선택 사항) 관리자 보안 암호 ARN은이 Salesforce 클라이언트 보안 암호를 교체하는 데 사용되는 관리 OAuth 보안 인증 정보가 포함된 보안 암호의 Amazon 리소스 이름(ARN)입니다. 최소한 관리자 보안 암호에는 보안 암호 구조 내에 consumerKey 및 consumerSecret 값이 포함되어야 합니다. 이 필드는 선택적 필드이며 생략하면 교체 중에 Secrets Manager는이 보안 암호 내의 OAuth 자격 증명을 사용하여 Salesforce로 인증합니다.

사용 흐름

에 Salesforce 보안 암호를 저장하는 고객은 동일한 보안 암호에 저장된 자격 증명으로 보안 암호를 교체하거나 교체 AWS Secrets Manager 를 위해 관리자 보안 암호의 자격 증명을 사용할 수 있습니다. 위에서 언급한 필드와 보안 암호 유형을 SalesforceClientSecret으로 포함하는 보안 암호 값을 사용하여 CreateSecret 호출을 사용하여 보안 암호를 생성할 수 있습니다. SalesforceClientSecret 교체 구성은 RotateSecret 호출을 사용하여 설정할 수 있습니다. 이 호출에는 위의 예제와 같이 메타데이터 필드의 사양이 필요합니다. 동일한 보안 암호의 자격 증명을 사용하여 교체를 선택하는 경우 adminSecretArn 필드를 건너뛸 수 있습니다. 또한 고객은 RotateSecret 호출에서 보안 암호를 교체하는 데 필요한 권한을 서비스에 부여하는 역할 ARN을 제공해야 합니다. 권한 정책의 예는 보안 및 권한을 참조하세요.

별도의 보안 인증 정보 세트(관리자 보안 암호에 저장됨)를 사용하여 보안 암호를 교체하려는 고객의 경우 소비자 보안 암호와 정확히 동일한 단계에 AWS Secrets Manager 따라에서 관리자 보안 암호를 생성해야 합니다. 소비자 보안 암호에 대한 RotateSecret 호출의 교체 메타데이터에이 관리자 보안 암호의 ARN을 제공해야 합니다.

교체 로직은 Salesforce에서 제공하는 지침을 따릅니다.

빅 ID 새로 고침 토큰

보안 암호 값 필드

다음은 Secrets Manager 보안 암호에 포함되어야 하는 필드입니다.

```
{
  "hostname": "Host Name",
  "refreshToken": "Refresh Token"
}
```

hostname

BigID 인스턴스가 호스팅되는 호스트 이름입니다. 인스턴스의 정규화된 도메인 이름을 입력해야 합니다.

refreshToken

관리 → 액세스 관리 → 사용자 선택 → 토큰 생성 → 저장을 통해 BigID 콘솔에서 생성된 JWT 사용자 새로 고침 토큰

사용 흐름

위에서 언급한 필드와 보안 암호 유형을 BigIDClientSecret으로 포함하는 보안 암호 값을 사용하여 CreateSecret 호출을 사용하여 보안 암호를 생성할 수 있습니다. BigIDClientSecret 교체 구성은 RotateSecret 호출을 사용하여 설정할 수 있습니다. 또한 RotateSecret 호출에 보안 암호를 교체하는 데 필요한 권한을 서비스에 부여하는 역할 ARN을 제공해야 합니다. 권한 정책의 예는 보안 및 권한을 참조하세요. 이 파트너의 경우 교체 메타데이터 필드를 비워 둘 수 있습니다.

Snowflake 키 페어

보안 암호 값 필드

다음은 Secrets Manager 보안 암호에 포함되어야 하는 필드입니다.

```
{
  "account": "Your Account Identifier",
  "user": "Your user name",
  "privateKey": "Your private Key",
  "publicKey": "Your public Key",
  "passphrase": "Your Passphrase"
}
```

user

이 키 페어 인증과 연결된 Snowflake 사용자 이름입니다. 이 사용자는 키 페어 인증을 수락하도록 Snowflake에서 구성되어야 하며, 퍼블릭 키는이 사용자의 프로필에 할당되어야 합니다.

계정

연결을 설정하는 데 사용되는 Snowflake 계정 식별자입니다. 이는 Snowflake URL(.snowflakecomputing.com 이전 부분)에서 추출할 수 있습니다.

privateKey

인증에 사용되는 PEM 형식의 RSA 프라이빗 키입니다. BEGIN/END 마커는 선택 사항입니다.

publicKey

프라이빗 키에 해당하는 PEM 형식의 퍼블릭 키입니다. BEGIN/END 마커는 선택 사항입니다.

암호

(선택 사항)이 필드는 암호화된 프라이빗 키를 해독하는 데 사용되는 암호를 나타냅니다.

보안 암호 메타데이터 필드

다음은 Snowflake의 메타데이터 필드입니다.

```
{
  "cryptographicAlgorithm": "Your Cryptographic algorithm",
  "encryptPrivateKey": "True/False"
}
```

cryptographicAlgorithm

(선택 사항) 키 생성에 사용되는 알고리즘을 나타냅니다. 의 세 가지 알고리즘을 선택할 수 있습니다RS256|RS384|RS512. 이 필드는 선택 사항이며 선택한 기본 알고리즘은 RS256입니다.

encryptPrivateKey

(선택 사항)이 필드를 사용하여 프라이빗 키를 암호화할지 여부를 선택할 수 있습니다. 이 파라미터의 기본값은 false입니다. 암호화를 위한 암호는 무작위로 생성됩니다.

사용 흐름

위에서 언급한 필드와 보안 암호 유형을 SnowflakeKeyPairAuthentication으로 포함하는 보안 암호 값을 사용하여 CreateSecret 호출을 사용하여 보안 암호를 생성할 수 있습니다. 교체 구성은 RotateSecret 호출을 사용하여 설정할 수 있습니다. 필요에 따라 보안 암호 메타데이터 필드(들)를 제공할 수 있습니다. 또한 RotateSecret 호출에서 보안 암호를 교체하는 데 필요한 권한을 서비스에 부여하는 역할 ARN을 제공해야 합니다. 권한 정책의 예는 보안 및 권한을 참조하세요. 이 파트너의 경우 교체 메타데이터 필드를 비워 둘 수 있습니다.

보안 및 권한

관리형 외부 보안 암호는 타사 애플리케이션 계정의 관리자 수준 권한을 공유할 필요가 없습니다 AWS. 대신 교체 프로세스는 사용자가 제공하는 자격 증명과 메타데이터를 사용하여 자격 증명 업데이트 및 검증을 위해 타사 애플리케이션에 승인된 API를 호출합니다.

관리형 외부 보안 암호는 다른 Secrets Manager 보안 암호 유형과 동일한 보안 표준을 유지합니다. 보안 암호 값은 KMS 키를 사용하여 저장 시 암호화되고 TLS를 사용하여 전송 중에 암호화됩니다. 보안 암호에 대한 액세스는 IAM 정책 및 리소스 기반 정책을 통해 제어됩니다. 고객 관리형 키를 사용하여 보안 암호를 암호화하는 경우 교체 역할의 IAM 정책과 CMK 신뢰 정책을 업데이트하여 성공적인 교체를 보장하는 데 필요한 권한을 제공해야 합니다.

교체가 제대로 작동하려면 Secrets Manager에 보안 암호 수명 주기를 관리할 수 있는 특정 권한을 제공해야 합니다. 이러한 권한은 개별 보안 암호로 범위가 지정될 수 있으며 최소 권한 원칙을 따릅니다. 제공하는 교체 역할은 설정 중에 검증되며 교체 작업에만 사용됩니다.

보안 암호가 있는 리전에서 EC2의 IP AWS 범위만 허용하여 외부 리소스로 IP 수신을 제한할 수 있습니다. 이 IP 범위 목록은 변경될 수 있으므로 수신 규칙을 주기적으로 새로 고쳐야 합니다.

AWS Secrets Manager 또한는 Secrets Manager 콘솔을 통해 보안 암호를 생성할 때 보안 암호를 관리하는 데 필요한 권한으로 IAM 정책을 생성하는 원터치 솔루션을 제공합니다. 이 역할에 대한 권한은 각 리전의 각 통합 파트너에 대해 범위가 축소됩니다.

권한 정책 예제:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowRotationAccess",
      "Action": [
        "secretsmanager:DescribeSecret",
        "secretsmanager:GetSecretValue",
        "secretsmanager:PutSecretValue",
        "secretsmanager:UpdateSecretVersionStage"
      ],
      "Resource": "*",
      "Effect": "Allow",
      "Condition": {
        "StringEquals": {
          "secretsmanager:resource/Type": "SalesforceClientSecret"
```

```
        }
      }
    },
    {
      "Sid": "AllowPasswordGenerationAccess",
      "Action": [
        "secretsmanager:GetRandomPassword"
      ],
      "Resource": "*",
      "Effect": "Allow"
    }
  ]
}
```

참고: secretsmanager:resource/Type에 사용할 수 있는 보안 암호 유형 목록은 통합 파트너에서 확인할 수 있습니다.

신뢰 정책 예:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "SecretsManagerPrincipalAccess",
      "Effect": "Allow",
      "Principal": {
        "Service": "secretsmanager.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "111122223333"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:secretsmanager:us-east-1:111122223333:secret:*"
        }
      }
    }
  ]
}
```

관리형 외부 보안 암호 모니터링 및 문제 해결

관리형 외부 보안 암호는 AWS CloudTrail 로그 및 Amazon CloudWatch 지표를 통해 포괄적인 모니터링 기능을 제공합니다. 모든 교체 활동은 성공, 실패 및 프로세스 중에 발생한 오류에 대한 자세한 정보와 함께 기록됩니다.

교체 워크플로의 일반적인 문제에는 역할 권한 또는 보안 암호 값의 잘못된 구성이 포함됩니다. 이러한 필드를 설정하지 않으면 서비스가 보안 암호에 액세스하거나 통합 파트너 클라이언트와 연결하여 보안 암호를 업데이트할 수 없으므로 통합 파트너가 지정한 형식이 교체 실패를 일으킬 수 있습니다. 다른 문제는 네트워크 연결 문제, 자격 증명 만료 또는 파트너 서비스 가용성일 수 있습니다. 관리형 교체 서비스에는 신뢰성을 극대화하기 위한 재시도 로직 및 오류 처리가 포함됩니다.

Amazon CloudWatch를 통해 교체 일정, 성공률 및 성능 지표를 모니터링할 수 있습니다. 이벤트 브리지를 통해 사용자 지정 경보를 구성하여 교체 실패 또는 주의가 필요한 기타 문제를 알릴 수 있습니다.

기존 보안 암호 마이그레이션

기존 파트너 보안 암호를 관리형 외부 보안 암호로 마이그레이션할 수 있습니다. 이는 UpdateSecret 호출을 통해 수행할 수 있습니다. 가이드에 언급된 대로 보안 암호 값과 메타데이터를 업데이트해야 합니다. 이러한 보안 암호에 대해 사용자 지정 교체 로직이 이미 설정되어 있는 경우 먼저 CancelRotateSecret 호출을 사용하여 교체를 취소해야 합니다.

제한 사항 및 고려 사항

관리형 외부 보안 암호는 수명이 4시간 미만인 임시 보안 암호를 지원하지 않습니다. 퍼블릭 키 인프라 인증서와 연결된 보안 암호도 지원되지 않습니다.

관리형 외부 보안 암호는에 온보딩한 파트너에 대해서만 지원됩니다 AWS Secrets Manager. 전체 목록은 통합 파트너를 참조하세요. 목록에 파트너가 보이지 않나요? 에 온보딩하도록 지시 AWS Secrets Manager

Secrets Manager 교체 엔진 외부의 파트너 클라이언트 서비스에서 직접 보안 암호 값을 업데이트하거나 교체하면 시스템 간 동기화가 중단될 수 있습니다. Secrets Manager는 수동 보안 암호 값 업데이트에 대한 콘솔 경고 및 프로그래밍 방식 방지를 제공하지만 타사 애플리케이션에서 직접 값을 수정할 수 있습니다. out-of-band 업데이트 후 동기화를 다시 설정하려면 올바른 보안 암호를 반영하도록 보안 암호 값을 업데이트한 다음 RotateSecret API를 호출하여 계속 성공적인 교체를 보장해야 합니다.

에서 AWS Secrets Manager 보안 암호 생성 AWS CloudFormation

보안 암호 생성에 나와 있는 것처럼 CloudFormation 템플릿의 `AWS::SecretsManager::Secret` 리소스를 사용하여 CloudFormation 스택에 보안 암호를 생성할 수 있습니다.

Amazon RDS 또는 Aurora에 대한 관리자 암호를 생성하려면 `AWS::RDS::DBCluster`에서 `ManageMasterUserPassword`를 사용하는 것이 좋습니다 . 그러면 Amazon RDS가 암호를 생성하고 자동으로 교체를 관리합니다. 자세한 내용은 관리형 교체 단원을 참조하십시오.

Amazon Redshift 및 Amazon DocumentDB 자격 증명에 대해서는 먼저 Secrets Manager에서 생성한 암호로 보안 암호를 생성한 다음 동적 참조를 사용하여 새 데이터베이스의 자격 증명으로 사용할 보안 암호에서 사용자 이름과 암호를 검색합니다. 다음으로 `AWS::SecretsManager::SecretTargetAttachment` 리소스를 사용하여 Secrets Manager가 보안 암호를 교체하는 데 필요한 보안 암호에 데이터베이스에 관한 세부 정보를 추가합니다. 마지막으로 자동 교체를 켜려면 `AWS::SecretsManager::RotationSchedule` 리소스를 사용하고 교체 함수와 일정을 제공합니다. 다음 예를 참조하세요.

- Amazon Redshift 자격 증명을 사용하여 보안 암호 생성
- Amazon DocumentDB 자격 증명을 사용하여 보안 암호 생성

보안 암호에 리소스 정책을 연결하려면 `AWS::SecretsManager::ResourcePolicy` 리소스를 사용합니다.

를 사용하여 리소스를 생성하는 방법에 대한 자세한 내용은 CloudFormation 사용 설명서의 템플릿 기본 사항 알아보기를 CloudFormation참조하세요. https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/gettingstarted.templatebasics.html AWS Cloud Development Kit (AWS CDK)도 사용할 수 있습니다. 자세한 내용은 AWS Secrets Manager 라이브러리 구성을 참조하세요.

를 사용하여 AWS Secrets Manager 보안 암호 생성 CloudFormation

이 예제에서는 **`CloudFormationCreatedSecret-`*`a1b2c3d4e5f6`***이라는 이름의 보안 암호를 생성합니다. 보안 암호 값은 다음 JSON으로, 보안 암호를 생성할 때 생성된 32자 암호를 포함합니다.

```
{
    "password": "EXAMPLE-PASSWORD",
```

```
    "username": "saanvi"
}
```

이 예제에서는 다음 CloudFormation 리소스를 사용합니다.

- AWS::SecretsManager::Secret

를 사용하여 리소스를 생성하는 방법에 대한 자세한 내용은 CloudFormation 사용 설명서의 템플릿 기본 사항 알아보기를 CloudFormation참조하세요. https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/gettingstarted.templatebasics.html

JSON

```
{
    "Resources": {
        "CloudFormationCreatedSecret": {
            "Type": "AWS::SecretsManager::Secret",
            "Properties": {
                "Description": "Simple secret created by CloudFormation.",
                "GenerateSecretString": {
                    "SecretStringTemplate": "{\"username\": \"saanvi\"}",
                    "GenerateStringKey": "password",
                    "PasswordLength": 32
                }
            }
        }
    }
}
```

YAML

```
Resources:
  CloudFormationCreatedSecret:
    Type: 'AWS::SecretsManager::Secret'
    Properties:
      Description: Simple secret created by CloudFormation.
      GenerateSecretString:
        SecretStringTemplate: '{"username": "saanvi"}'
        GenerateStringKey: password
        PasswordLength: 32
```

자동 교체를 사용하여 AWS Secrets Manager 보안 암호 생성 및를 사용하여 Amazon RDS MySQL DB 인스턴스 생성 CloudFormation

Amazon RDS 또는 Aurora에 대한 관리자 암호를 만들려면 `AWS::RDS::DBCluster`의 마스터 암호에 대한 Secrets Manager 암호 생성 예에 나와 있는 것처럼 `ManageMasterUserPassword`를 사용하는 것이 좋습니다. 그러면 Amazon RDS가 암호를 생성하고 자동으로 교체를 관리합니다. 자세한 내용은 관리형 교체 단원을 참조하십시오.

를 사용하여 AWS Secrets Manager 보안 암호 및 Amazon Redshift 클러스터 생성 CloudFormation

Amazon Redshift의 관리자 보안 암호를 생성하려면 `AWS::Redshift::Cluster` 및 `AWS::RedshiftServerless::Namespace`의 예제를 사용하는 것이 좋습니다.

를 사용하여 AWS Secrets Manager 보안 암호 및 Amazon DocumentDB 인스턴스 생성 CloudFormation

이 예제에서는 보안 암호를 생성하고, 보안 암호의 자격 증명을 사용자와 암호로 사용하는 Amazon DocumentDB 인스턴스를 생성합니다. 보안 암호에는 보안 암호에 액세스할 수 있는 사용자를 정의하는 리소스 기반 정책이 연결되어 있습니다. 또한 템플릿은 교체 함수 템플릿에서 Lambda 교체 함수를 생성하고 매월 첫째 날 오전 8시에서 오전 10시 사이에 보안 암호를 자동 교체하도록 구성합니다. 보안 모범 사례로서 인스턴스는 Amazon VPC에 있습니다.

이 예제에서는 Secrets Manager에 대해 다음 CloudFormation 리소스를 사용합니다.

- `AWS::SecretsManager::Secret`
- `AWS::SecretsManager::SecretTargetAttachment`
- `AWS::SecretsManager::RotationSchedule`

를 사용하여 리소스를 생성하는 방법에 대한 자세한 내용은 CloudFormation 사용 설명서의 템플릿 기본 사항 알아보기를 CloudFormation참조하세요. https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/gettingstarted.templatebasics.html

JSON

```
{
```

```
"AWSTemplateFormatVersion":"2010-09-09",
"Transform":"AWS::SecretsManager-2020-07-23",
"Resources":{
   "TestVPC":{
      "Type":"AWS::EC2::VPC",
      "Properties":{
         "CidrBlock":"10.0.0.0/16",
         "EnableDnsHostnames":true,
         "EnableDnsSupport":true
      }
   },
   "TestSubnet01":{
      "Type":"AWS::EC2::Subnet",
      "Properties":{
         "CidrBlock":"10.0.96.0/19",
         "AvailabilityZone":{
            "Fn::Select":[
               "0",
               {
                  "Fn::GetAZs":{
                     "Ref":"AWS::Region"
                  }
               }
            ]
         },
         "VpcId":{
            "Ref":"TestVPC"
         }
      }
   },
   "TestSubnet02":{
      "Type":"AWS::EC2::Subnet",
      "Properties":{
         "CidrBlock":"10.0.128.0/19",
         "AvailabilityZone":{
            "Fn::Select":[
               "1",
               {
                  "Fn::GetAZs":{
                     "Ref":"AWS::Region"
                  }
               }
            ]
         },
```

```
            "VpcId":{
               "Ref":"TestVPC"
            }
         }
      },
      "SecretsManagerVPCEndpoint":{
         "Type":"AWS::EC2::VPCEndpoint",
         "Properties":{
            "SubnetIds":[
               {
                  "Ref":"TestSubnet01"
               },
               {
                  "Ref":"TestSubnet02"
               }
            ],
            "SecurityGroupIds":[
               {
                  "Fn::GetAtt":[
                     "TestVPC",
                     "DefaultSecurityGroup"
                  ]
               }
            ],
            "VpcEndpointType":"Interface",
            "ServiceName":{
               "Fn::Sub":"com.amazonaws.${AWS::Region}.secretsmanager"
            },
            "PrivateDnsEnabled":true,
            "VpcId":{
               "Ref":"TestVPC"
            }
         }
      },
      "MyDocDBClusterRotationSecret":{
         "Type":"AWS::SecretsManager::Secret",
         "Properties":{
            "GenerateSecretString":{
               "SecretStringTemplate":"{\"username\": \"someadmin\",\"ssl\": true}",
               "GenerateStringKey":"password",
               "PasswordLength":16,
               "ExcludeCharacters":"\"@/\\"
            },
            "Tags":[
```

```
                {
                   "Key":"AppName",
                   "Value":"MyApp"
                }
             ]
          }
      },
      "MyDocDBCluster":{
         "Type":"AWS::DocDB::DBCluster",
         "Properties":{
            "DBSubnetGroupName":{
               "Ref":"MyDBSubnetGroup"
            },
            "MasterUsername":{
               "Fn::Sub":"{{resolve:secretsmanager:
${MyDocDBClusterRotationSecret}::username}}"
            },
            "MasterUserPassword":{
               "Fn::Sub":"{{resolve:secretsmanager:
${MyDocDBClusterRotationSecret}::password}}"
            },
            "VpcSecurityGroupIds":[
               {
                  "Fn::GetAtt":[
                     "TestVPC",
                     "DefaultSecurityGroup"
                  ]
               }
            ]
         }
      },
      "DocDBInstance":{
         "Type":"AWS::DocDB::DBInstance",
         "Properties":{
            "DBClusterIdentifier":{
               "Ref":"MyDocDBCluster"
            },
            "DBInstanceClass":"db.r5.large"
         }
      },
      "MyDBSubnetGroup":{
         "Type":"AWS::DocDB::DBSubnetGroup",
         "Properties":{
            "DBSubnetGroupDescription":"",
```

```
            "SubnetIds":[
               {
                  "Ref":"TestSubnet01"
               },
               {
                  "Ref":"TestSubnet02"
               }
            ]
         }
      },
      "SecretDocDBClusterAttachment":{
         "Type":"AWS::SecretsManager::SecretTargetAttachment",
         "Properties":{
            "SecretId":{
               "Ref":"MyDocDBClusterRotationSecret"
            },
            "TargetId":{
               "Ref":"MyDocDBCluster"
            },
            "TargetType":"AWS::DocDB::DBCluster"
         }
      },
      "MySecretRotationSchedule":{
         "Type":"AWS::SecretsManager::RotationSchedule",
         "DependsOn":"SecretDocDBClusterAttachment",
         "Properties":{
            "SecretId":{
               "Ref":"MyDocDBClusterRotationSecret"
            },
            "HostedRotationLambda":{
               "RotationType":"MongoDBSingleUser",
               "RotationLambdaName":"MongoDBSingleUser",
               "VpcSecurityGroupIds":{
                  "Fn::GetAtt":[
                     "TestVPC",
                     "DefaultSecurityGroup"
                  ]
               },
               "VpcSubnetIds":{
                  "Fn::Join":[
                     ",",
                     [
                        {
                           "Ref":"TestSubnet01"
```

```
                        },
                        {
                           "Ref":"TestSubnet02"
                        }
                     ]
                  ]
               }
            },
            "RotationRules":{
              "Duration": "2h",
              "ScheduleExpression": "cron(0 8 1 * ? *)"
            }
         }
      }
   }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::SecretsManager-2020-07-23
Resources:
  TestVPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsHostnames: true
      EnableDnsSupport: true
  TestSubnet01:
    Type: AWS::EC2::Subnet
    Properties:
      CidrBlock: 10.0.96.0/19
      AvailabilityZone: !Select
        - '0'
        - !GetAZs
          Ref: AWS::Region
      VpcId: !Ref TestVPC
  TestSubnet02:
    Type: AWS::EC2::Subnet
    Properties:
      CidrBlock: 10.0.128.0/19
      AvailabilityZone: !Select
        - '1'
```

```
        - !GetAZs
          Ref: AWS::Region
      VpcId: !Ref TestVPC
  SecretsManagerVPCEndpoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      SubnetIds:
        - !Ref TestSubnet01
        - !Ref TestSubnet02
      SecurityGroupIds:
        - !GetAtt TestVPC.DefaultSecurityGroup
      VpcEndpointType: Interface
      ServiceName: !Sub com.amazonaws.${AWS::Region}.secretsmanager
      PrivateDnsEnabled: true
      VpcId: !Ref TestVPC
  MyDocDBClusterRotationSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      GenerateSecretString:
        SecretStringTemplate: '{"username": "someadmin","ssl": true}'
        GenerateStringKey: password
        PasswordLength: 16
        ExcludeCharacters: '"@/\'
      Tags:
        - Key: AppName
          Value: MyApp
  MyDocDBCluster:
    Type: AWS::DocDB::DBCluster
    Properties:
      DBSubnetGroupName: !Ref MyDBSubnetGroup
      MasterUsername: !Sub '{{resolve:secretsmanager:
${MyDocDBClusterRotationSecret}::username}}'
      MasterUserPassword: !Sub '{{resolve:secretsmanager:
${MyDocDBClusterRotationSecret}::password}}'
      VpcSecurityGroupIds:
        - !GetAtt TestVPC.DefaultSecurityGroup
  DocDBInstance:
    Type: AWS::DocDB::DBInstance
    Properties:
      DBClusterIdentifier: !Ref MyDocDBCluster
      DBInstanceClass: db.r5.large
  MyDBSubnetGroup:
    Type: AWS::DocDB::DBSubnetGroup
    Properties:
```

```
      DBSubnetGroupDescription: ''
      SubnetIds:
        - !Ref TestSubnet01
        - !Ref TestSubnet02
  SecretDocDBClusterAttachment:
    Type: AWS::SecretsManager::SecretTargetAttachment
    Properties:
      SecretId: !Ref MyDocDBClusterRotationSecret
      TargetId: !Ref MyDocDBCluster
      TargetType: AWS::DocDB::DBCluster
  MySecretRotationSchedule:
    Type: AWS::SecretsManager::RotationSchedule
    DependsOn: SecretDocDBClusterAttachment
    Properties:
      SecretId: !Ref MyDocDBClusterRotationSecret
      HostedRotationLambda:
        RotationType: MongoDBSingleUser
        RotationLambdaName: MongoDBSingleUser
        VpcSecurityGroupIds: !GetAtt TestVPC.DefaultSecurityGroup
        VpcSubnetIds: !Join
          - ','
          - - !Ref TestSubnet01
            - !Ref TestSubnet02
      RotationRules:
        Duration: 2h
        ScheduleExpression: cron(0 8 1 * ? *)
```

Secrets Manager의 사용 방식 AWS CloudFormation

콘솔을 사용하여 교체를 켜면 Secrets Manager는 AWS CloudFormation 를 사용하여 교체를 위한 리소스를 생성합니다. 해당 프로세스 중에 새 교체 함수를 생성하는 경우는 적절한를 `AWS::Serverless::Function` 기반으로를 CloudFormation 생성합니다교체 함수 템플릿. 그런 다음는 보안 CloudFormation 암호에 대한 교체 함수 및 교체 규칙을 `RotationSchedule`설정하는를 설정합니다. 자동 교체를 켠 후 배너에서 CloudFormation 스택 보기를 선택하여 스택을 볼 수 있습니다.

자동 교체 활성화에 대한 정보는 보안 암호 교체를 참조하세요.

에서 AWS Secrets Manager 보안 암호 생성 AWS Cloud Development Kit (AWS CDK)

CDK 앱에서 비밀을 생성, 관리 및 검색하려면 `ResourcePolicy`, `RotationSchedule`, `Secret`, `SecretRotation`, `SecretTargetAttachment` 구성이 들어 있는 AWS Secrets Manager 구성 라이브러리를 이용하면 됩니다.

CDK 애플리케이션에서 보안 암호를 사용하는 모범 사례는 우선 콘솔 또는 CLI를 사용하여 보안 암호를 생성한 다음, 보안 암호를 CDK 애플리케이션으로 가져오는 것입니다.

예를 들어 참조할 섹션:

- 보안 암호 생성
- 보안 암호 가져오기
- 보안 암호 검색
- 암호 사용 권한 부여
- 보안 암호 교체
- 데이터베이스 보안 암호 교체
- 다른 리전으로 보안 암호 복제

CDK 에 대한 자세한 내용은 AWS Cloud Development Kit (AWS CDK) v2 개발자 안내서를 참조하세요.

AWS Secrets Manager 보안 암호 모니터링

AWS 는 Secrets Manager 보안 암호를 모니터링하고, 이상이 있을 때 보고하고, 필요한 경우 자동 조치를 취할 수 있는 모니터링 도구를 제공합니다. 예기치 않은 사용 또는 변경을 조사해야 하는 경우 로그를 사용하여 원치 않는 변경 사항을 취소할 수 있습니다. 또한 보안 암호의 부적절한 사용 및 보안 암호 삭제 시도에 대한 자동 검사를 설정할 수 있습니다.

주제

- 를 사용하여 AWS Secrets Manager 이벤트 로깅 AWS CloudTrail
- Amazon CloudWatch AWS Secrets Manager 로 모니터링
- Amazon EventBridge와 AWS Secrets Manager 이벤트 일치
- 삭제 예약된 AWS Secrets Manager 보안 암호에 액세스하는 시기 모니터링
- 를 사용하여 AWS Secrets Manager 보안 암호의 규정 준수 모니터링 AWS Config
- Secrets Manager 비용 모니터링
- Amazon GuardDuty로 위협 탐지

를 사용하여 AWS Secrets Manager 이벤트 로깅 AWS CloudTrail

AWS CloudTrail 는 Secrets Manager 콘솔의 호출을 포함하여 Secrets Manager에 대한 모든 API 호출과 교체 및 보안 암호 버전 삭제를 위한 기타 여러 이벤트를 이벤트로 기록합니다. 로그 항목 Secrets Manager 기록의 목록은 CloudTrail 항목을 참조하세요.

CloudTrail 콘솔을 사용하여 지난 90일 동안 기록된 이벤트를 볼 수 있습니다. Secrets Manager에 대한 이벤트를 포함하여 AWS 계정에 이벤트를 지속적으로 기록하려면 CloudTrail이 Amazon S3 버킷에 로그 파일을 전송하도록 추적을 생성합니다. AWS 계정에 대한 추적 생성을 참조하세요. 또한 여러 AWS 계정 및 AWS 리전에서 CloudTrail 로그 파일을 수신하도록 CloudTrail을 구성할 수도 있습니다.

CloudTrail 로그에서 수집된 데이터를 추가로 분석하고 조치를 취하도록 다른 AWS 서비스를 구성할 수 있습니다. CloudTrail 로그와의AWS 서비스 통합을 참조하세요. CloudTrail이 Amazon S3 버킷에 새 로그 파일을 게시할 때도 이에 대한 알림을 받을 수 있습니다. CloudTrail에 대한 Amazon SNS 알림 구성 섹션을 참조하세요.

CloudTrail 로그에서 Secrets Manager 이벤트를 검색하려면(콘솔)

1. https://console.aws.amazon.com/cloudtrail/에서 CloudTrail 콘솔을 엽니다.

2. 콘솔에서 이벤트가 발생한 리전을 가리키고 있는지 확인합니다. 콘솔에는 선택한 리전에서 발생한 이벤트만 표시됩니다. 오른쪽 상단 모서리에 있는 드롭다운 목록에서 리전을 선택합니다.
3. 왼쪽 탐색 창에서 Event history(이벤트 기록)을 선택합니다.
4. 필터 기준 및/또는 시간 범위를 선택하면 원하는 이벤트를 찾는 데 도움이 됩니다. 예제:

   a. 예를 들어, 모든 Secrets Manager 이벤트를 보려면 속성 조회에서 이벤트 소스를 선택합니다. 그런 다음 이벤트 소스 입력에서 **secretsmanager.amazonaws.com**을 선택합니다.

   b. 보안 암호의 모든 이벤트를 보려면 조회 속성에서 리소스 이름을 선택합니다. 그런 다음, 리소스 이름 입력에 보안 암호의 이름을 입력합니다.
5. 자세한 정보를 보려면 이벤트 옆에 있는 확장 화살표를 선택합니다. 사용 가능한 모든 정보를 보려면 이벤트 보기를 선택합니다.

AWS CLI

Example CloudTrail 로그에서 Secrets Manager 이벤트 검색

다음 `lookup-events` 예에서는 Secrets Manager 이벤트를 검색합니다.

```
aws cloudtrail lookup-events \
    --region us-east-1 \
    --lookup-attributes
 AttributeKey=EventSource,AttributeValue=secretsmanager.amazonaws.com
```

AWS CloudTrail Secrets Manager에 대한 항목

AWS Secrets Manager 는 모든 Secrets Manager 작업 및 교체 및 삭제와 관련된 기타 이벤트에 대한 항목을 AWS CloudTrail 로그에 기록합니다. 이러한 이벤트를 설정하는 방법에 대한 자세한 내용은 EventBridge를 사용하여 Secrets Manager 이벤트 매칭 섹션을 참조하세요.

로그 항목 유형

- Secrets Manager 작업에 대한 로그 항목
- 삭제할 로그 항목
- 복제를 위한 로그 항목
- 교체에 대한 로그 항목

Secrets Manager 작업에 대한 로그 항목

Secrets Manager 작업을 호출하여 생성되는 이벤트에는 `"detail-type": ["AWS API Call via CloudTrail"]`이(가) 있습니다.

> **Note**
>
> 2024년 2월 이전에 일부 Secrets Manager 작업에서 보안 암호 ARN에 대하여 'arn' 대신 'aRN'이 포함된 이벤트가 보고되었습니다. 자세한 내용은 AWS re:Post를 참조하세요.

사용자 또는 서비스가 API나 SDK, CLI를 통해 Secrets Manager 작업을 호출할 때 생성되는 CloudTrail 항목은 다음과 같습니다.

BatchGetSecretValue

BatchGetSecretValue 작업에 의해 생성됩니다. 보안 암호 검색에 대한 정보는 보안 암호 가져오기를 참조하세요.

CancelRotateSecret

CancelRotateSecret 작업에 의해 생성됩니다. 교체에 대한 정보는 보안 암호 교체를 참조하세요.

CreateSecret

CreateSecret 작업에 의해 생성됩니다. 보안 암호 생성에 대한 정보는 보안 암호 관리를 참조하세요.

DeleteResourcePolicy

DeleteResourcePolicy 작업에 의해 생성됩니다. 권한에 대한 정보는 the section called "인증 및 액세스 제어"를 참조하세요.

DeleteSecret

DeleteSecret 작업에 의해 생성됩니다. 보안 암호 삭제에 대한 정보는 the section called "보안 암호 삭제"를 참조하세요.

DescribeSecret

DescribeSecret 작업에 의해 생성됩니다.

GetRandomPassword

GetRandomPassword 작업에 의해 생성됩니다.

GetResourcePolicy

GetResourcePolicy 작업에 의해 생성됩니다. 권한에 대한 정보는 the section called “인증 및 액세스 제어”를 참조하세요.

GetSecretValue

GetSecretValue 및 BatchGetSecretValue 작업에 의해 생성됩니다. 보안 암호 검색에 대한 정보는 보안 암호 가져오기를 참조하세요.

ListSecrets

ListSecrets 작업에 의해 생성됩니다. 보안 암호 목록에 대한 정보는 the section called “보안 암호 찾기”를 참조하세요.

ListSecretVersionIds

ListSecretVersionIds 작업에 의해 생성됩니다.

PutResourcePolicy

PutResourcePolicy 작업에 의해 생성됩니다. 권한에 대한 정보는 the section called “인증 및 액세스 제어”를 참조하세요.

PutSecretValue

PutSecretValue 작업에 의해 생성됩니다. 보안 암호 업데이트에 대한 정보는 the section called “보안 암호 수정”을 참조하세요.

RemoveRegionsFromReplication

RemoveRegionsFromReplication 작업에 의해 생성됩니다. 보안 암호 복제에 대한 정보는 다중 리전 복제를 참조하세요.

ReplicateSecretToRegions

ReplicateSecretToRegions 작업에 의해 생성됩니다. 보안 암호 복제에 대한 정보는 다중 리전 복제를 참조하세요.

RestoreSecret

RestoreSecret 작업에 의해 생성됩니다. 삭제된 보안 암호 복원에 대한 정보는 the section called “보안 암호 복원”을 참조하세요.

RotateSecret

RotateSecret 작업에 의해 생성됩니다. 교체에 대한 정보는 보안 암호 교체를 참조하세요.

StopReplicationToReplica

StopReplicationToReplica 작업에 의해 생성됩니다. 보안 암호 복제에 대한 정보는 다중 리전 복제를 참조하세요.

TagResource

TagResource 작업에 의해 생성됩니다. 보안 암호 태그 지정에 대한 정보는 the section called “ 보안 암호 태그 지정”을 참조하세요.

UntagResource

UntagResource 작업에 의해 생성됩니다. 보안 암호 태그 해제에 대한 정보는 the section called “ 보안 암호 태그 지정”을 참조하세요.

UpdateSecret

UpdateSecret 작업에 의해 생성됩니다. 보안 암호 업데이트에 대한 정보는 the section called “보안 암호 수정”을 참조하세요.

UpdateSecretVersionStage

UpdateSecretVersionStage 작업에 의해 생성됩니다. 버전 단계에 대한 정보는 the section called “보안 암호 버전”을 참조하세요.

ValidateResourcePolicy

ValidateResourcePolicy 작업에 의해 생성됩니다. 권한에 대한 정보는 the section called “인증 및 액세스 제어”를 참조하세요.

삭제할 로그 항목

Secrets Manager는 Secrets Manager 작업을 위한 이벤트 외에도 삭제와 관련된 다음과 같은 이벤트를 생성합니다. 이러한 이벤트에는 `"detail-type": ["AWS Service Event via CloudTrail"]`이(가) 포함되어 있습니다.

CancelSecretVersionDelete

Secrets Manager 서비스에서 생성됩니다. 버전이 있는 암호에 대해 `DeleteSecret`을 호출한 다음 나중에 `RestoreSecret`을 호출하면 Secrets Manager는 복원된 각 보안 암호 버전에 대해 이 이벤트를 로그합니다. 삭제된 보안 암호 복원에 대한 정보는 the section called “보안 암호 복원”을 참조하세요.

EndSecretVersionDelete

암호 버전이 삭제되면 Secrets Manager 서비스에서 생성됩니다. 자세한 내용은 the section called "보안 암호 삭제" 단원을 참조하십시오.

StartSecretVersionDelete

Secrets Manager가 암호 버전 삭제를 시작하면 Secrets Manager 서비스에서 생성됩니다. 보안 암호 삭제에 대한 정보는 the section called "보안 암호 삭제"를 참조하세요.

보안 암호 버전 삭제

Secrets Manager가 더 이상 사용되지 않는 보안 암호 버전 삭제를 시작하면 Secrets Manager 서비스에서 생성됩니다. 자세한 내용은 보안 암호 버전을 참조하세요.

복제를 위한 로그 항목

Secrets Manager는 Secrets Manager 작업을 위한 이벤트 외에도 복제와 관련된 다음과 같은 이벤트를 생성합니다. 이러한 이벤트에는 `"detail-type": ["AWS Service Event via CloudTrail"]`이(가) 포함되어 있습니다.

ReplicationFailed

복제가 실패하면 Secrets Manager 서비스에서 생성됩니다. 보안 암호 복제에 대한 정보는 다중 리전 복제를 참조하세요.

ReplicationStarted

Secrets Manager가 보안 암호 복제를 시작하면 Secrets Manager 서비스에서 생성됩니다. 보안 암호 복제에 대한 정보는 다중 리전 복제를 참조하세요.

ReplicationSucceeded

보안 암호가 성공적으로 복제되면 Secrets Manager 서비스에서 생성됩니다. 보안 암호 복제에 대한 정보는 다중 리전 복제를 참조하세요.

교체에 대한 로그 항목

Secrets Manager는 Secrets Manager 작업을 위한 이벤트 외에도 교체와 관련된 다음과 같은 이벤트를 생성합니다. 이러한 이벤트에는 `"detail-type": ["AWS Service Event via CloudTrail"]`이(가) 포함되어 있습니다.

RotationStarted

Secrets Manager가 암호 교체를 시작하면 Secrets Manager 서비스에서 생성됩니다. 교체에 대한 정보는 보안 암호 교체를 참조하세요.

RotationAbandoned

Secrets Manager가 교체 시도를 중단하고 기존 보안 암호 버전에서 `AWSPENDING` 라벨을 제거하면 Secrets Manager 서비스에서 생성됩니다. Secrets Manager는 교체 중에 보안 암호의 새 버전을 생성하면 교체를 중단합니다. 교체에 대한 정보는 보안 암호 교체를 참조하세요.

RotationFailed

교체가 실패하면 Secrets Manager 서비스에서 생성됩니다. 교체에 대한 정보는 the section called "교체 문제 해결"를 참조하세요.

RotationSucceeded

암호가 성공적으로 교체되면 Secrets Manager 서비스에서 생성됩니다. 교체에 대한 정보는 보안 암호 교체를 참조하세요.

TestRotationStarted

Secrets Manager에서 즉시 교체가 예약되지 않은 보안 암호에 대한 교체 테스트를 시작하면 Secrets Manager 서비스에서 생성됩니다. 교체에 대한 정보는 보안 암호 교체를 참조하세요.

TestRotationSucceeded

Secrets Manager에서 즉시 교체가 예약되지 않은 보안 암호에 대한 교체 테스트를 성공적으로 마치면 Secrets Manager 서비스에서 생성됩니다. 교체에 대한 정보는 보안 암호 교체를 참조하세요.

TestRotationFailed

Secrets Manager에서 즉시 교체가 예약되지 않은 보안 암호에 대한 교체 테스트한 후 교체에 실패하면 Secrets Manager 서비스에서 생성됩니다. 교체에 대한 정보는 the section called "교체 문제 해결"를 참조하세요.

Amazon CloudWatch AWS Secrets Manager 로 모니터링

Amazon CloudWatch를 사용하면 AWS 서비스를 모니터링하고 경보를 생성하여 지표가 변경될 때 이를 알릴 수 있습니다. CloudWatch는 이러한 통계를 15개월간 보관하므로 기록 정보에 액세스하고 웹 애플리케이션 또는 서비스가 어떻게 실행되고 있는지 전체적으로 더 잘 파악할 수 있습니다. 의 AWS Secrets Manager경우 삭제로 표시된 보안 암호를 포함하여 계정의 보안 암호 수와 콘솔을 통한 호출을

포함하여 Secrets Manager에 대한 API 호출을 모니터링할 수 있습니다. 지표 모니터링에 대한 자세한 내용은 CloudWatch 사용 설명서의 CloudWatch 지표 사용을 참조하세요.

Secrets Manager 지표를 찾는 방법

1. CloudWatch 콘솔의 지표 에서 모든 지표를 선택합니다.
2. 지표 검색 상자에 `secret`을 입력합니다.
3. 해결 방법:
   - 계정의 보안 암호 수를 모니터링하려면 AWS/SecretsManager를 선택한 다음, SecretCount를 선택합니다. 이 지표는 매시간 게시됩니다.
   - 콘솔을 통한 호출을 포함하여 Secrets Manager에 대한 API 호출을 모니터링하려면 사용량 > AWS 리소스별을 선택한 다음 모니터링할 API 호출을 선택합니다. Secrets Manager API 목록은 Secrets Manager 작업 섹션을 참조하세요.
4. 해결 방법:
   - 지표의 그래프를 생성하려면 Amazon CloudWatch 사용 설명서의 지표 그래프 작성을 참조하세요.
   - 이상 징후를 탐지하려면 Amazon CloudWatch 사용 설명서의 CloudWatch 이상 탐지 사용 섹션을 참조하세요.
   - 지표에 대한 통계를 얻는 방법은 Amazon CloudWatch 사용 설명서의 지표에 대한 통계 얻기를 참조하세요.

CloudWatch 경보

지표 값이 변화하면 Amazon SNS 메시지를 보내고 경보가 상태를 변경하도록 하는 CloudWatch 경보를 만들 수 있습니다. 계정의 보안 암호 수를 나타내는 Secrets Manager 지표 `ResourceCount`에서 경보를 설정할 수 있습니다. 경보를 설정할 수도 있습니다. 경보는 지정한 기간에 지표를 감시하고 여러 기간에 지정된 임계값에 대한 지표 값을 기준으로 작업을 수행합니다. 경보는 지속적인 상태 변경에 대해서만 작업을 호출합니다. CloudWatch 경보는 특정 상태에 있다는 이유만으로는 작업을 호출하지 않습니다. 상태가 변경되고 지정한 기간 동안 유지되어야 합니다.

자세한 내용은 CloudWatch 사용 설명서의 Amazon CloudWatch 경보 사용 및 이상 탐지를 기반으로 CloudWatch 경보 생성 섹션을 참조하세요.

특정 임계값을 주시하다가 해당 임계값이 충족될 때 알림을 전송하거나 조치를 취하도록 경보를 설정할 수도 있습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서를 참조하세요.

Amazon EventBridge와 AWS Secrets Manager 이벤트 일치

Amazon EventBridge에서 CloudTrail 로그 항목의 Secrets Manager 이벤트를 매칭할 수 있습니다. 이러한 이벤트를 찾은 다음 조치를 취할 대상으로 새로 생성된 이벤트를 전송하는 EventBridge 규칙을 구성할 수 있습니다. Secrets Manager가 로그하는 CloudTrail 항목 목록은 CloudTrail 항목을(를) 참조하세요. EventBridge 설정 지침은 EventBridge 사용 설명서의 EventBridge 시작하기를 참조하세요.

모든 변경 사항을 지정된 암호와 매칭

> **Note**
>
> 일부 Secrets Manager 이벤트는 대소문자가 다른 암호의 ARN을 반환하므로 둘 이상의 작업과 매칭되는 이벤트 패턴에서 ARN으로 암호를 지정하려면 `arn` 와(과) `aRN` 키를 모두 포함해야 할 수 있습니다. 자세한 내용은 AWS re:Post를 참조하세요.

다음 예제는 암호 변경에 대한 로그 항목과 매칭되는 EventBridge 이벤트 패턴을 보여줍니다.

```
{
    "source": ["aws.secretsmanager"],
    "detail-type": ["AWS API Call via CloudTrail"],
    "detail": {
        "eventSource": ["secretsmanager.amazonaws.com"],
        "eventName": ["DeleteResourcePolicy", "PutResourcePolicy", "RotateSecret",
 "TagResource", "UntagResource", "UpdateSecret"],
        "responseElements": {
            "arn": ["arn:aws:secretsmanager:us-west-2:012345678901:secret:mySecret-
a1b2c3"]
        }
    }
}
```

암호 값이 교체될 때의 이벤트 매칭

다음 예제는 수동 업데이트 또는 자동 교체로 인해 발생하는 암호 값 변경에 대한 CloudTrail 로그 항목과 매칭되는 EventBridge 이벤트 패턴을 보여줍니다. 이러한 이벤트 중 일부는 Secrets Manager 작업에서 생성되고 일부는 Secrets Manager 서비스에서 생성되므로 두 이벤트 모두에 대해 `detail-type`을(를) 포함해야 합니다.

```
{
```

```
    "source": ["aws.secretsmanager"],
    "detail-type": [
        "AWS API Call via CloudTrail",
        "AWS Service Event via CloudTrail"
    ],
    "detail": {
        "eventSource": ["secretsmanager.amazonaws.com"],
        "eventName": ["PutSecretValue", "UpdateSecret", "RotationSucceeded"]
    }
}
```

삭제 예약된 AWS Secrets Manager 보안 암호에 액세스하는 시기 모니터링

AWS CloudTrail Amazon CloudWatch Logs 및 Amazon Simple Notification Service(Amazon SNS)의 조합을 사용하여 삭제 보류 중인 보안 암호에 액세스하려는 시도를 알리는 경보를 생성할 수 있습니다. 이러한 경보로부터 알림을 받으면 보안 암호 삭제를 취소하여, 정말로 삭제할지 판단할 시간을 확보하고 싶을 수 있습니다. 조사를 통해 필요하다고 판단되면 보안 암호를 복원할 수 있습니다. 또는 사용자가 사용할 새 암호 정보로 사용자를 업데이트해야 할 수 있습니다.

다음 절차에서는 특정 오류 메시지를 생성하는 `GetSecretValue` 작업에 대한 요청이 CloudTrail 로그 파일에 기록될 때 알림을 받는 방법에 대해 설명합니다. 경보를 트리거하지 않고 보안 암호에 대해 다른 API 작업을 수행할 수 있습니다. 이 CloudWatch 경보는 오래된 자격 증명을 사용하는 사용자나 애플리케이션을 나타낼 수 있는 사용을 감지합니다.

이러한 절차를 시작하기 전에 AWS Secrets Manager API 요청을 모니터링하려는 AWS 리전 및 계정에서 CloudTrail을 켜야 합니다. 지침은 AWS CloudTrail 사용 설명서의 첫 번째 추적 생성을 참조하세요.

1단계: CloudTrail 로그 파일이 CloudWatch Logs에 전송되도록 구성

CloudTrail 로그 파일을 CloudWatch Logs에 전송되도록 구성해야 합니다. 이렇게 하면 CloudWatch Logs에서 삭제 보류 중인 보안 암호를 검색하려는 Secrets Manager API 요청을 모니터링할 수 있습니다.

CloudTrail 로그 파일이 CloudWatch Logs에 전송되도록 구성

1. https://console.aws.amazon.com/cloudtrail/에서 CloudTrail 콘솔을 엽니다.
2. 상단 탐색 모음에서 보안 암호를 모니터링할 AWS 리전 를 선택합니다.

3. 왼쪽 탐색 창에서 추적(Trails)을 선택한 다음 CloudWatch에 대해 구성할 추적 이름을 선택합니다.
4. 추적 구성(Trails Configuration) 페이지에서 CloudWatch Logs 섹션까지 아래로 스크롤한 다음 편집 아이콘
   (✏)
   을 선택합니다.
5. New or existing log group(새 또는 기존 로그 그룹)에 로그 그룹 이름을 입력합니다(예: **`CloudTrail/MyCloudWatchLogGroup`**).
6. IAM 역할(IAM role)에 CloudTrail_CloudWatchLogs_Role이라는 기본 역할을 사용할 수 있습니다. 이 역할에는 CloudTrail 이벤트를 로그 그룹으로 전달하는 필수 권한이 있는 기본 역할 정책이 있습니다.
7. 계속을 선택하여 구성을 저장합니다.
8. AWS CloudTrail 은 이 계정의 API 활동과 관련된 CloudTrail 이벤트를 CloudWatch Logs 로그 그룹에 전달합니다 페이지에서 허용을 선택합니다.

2단계: CloudWatch 경보 생성

Secrets Manager `GetSecretValue` API 작업이 삭제 보류 중인 보안 암호에 대한 액세스를 요청할 때 알림을 받으려면 CloudWatch 경보를 생성하고 알림을 구성해야 합니다.

CloudWatch 경보를 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔에 로그인합니다.
2. 상단 탐색 모음에서 보안 암호를 모니터링할 AWS 리전을 선택합니다.
3. 왼쪽 탐색 창에서 로그를 선택합니다.
4. 로그 그룹 목록에서 CloudTrail/MyCloudWatchLogGroup처럼 이전 절차에서 생성한 로그 그룹 옆의 확인란을 선택합니다. 그런 다음 [Create Metric Filter]를 선택합니다.
5. [Filter Pattern]에 다음을 입력하거나 붙여 넣습니다.

   ```
   { $.eventName = "GetSecretValue" && $.errorMessage = "*secret because it was marked
    for deletion*" }
   ```

   [Assign Metric]을 선택합니다.
6. [Create Metric Filter and Assign a Metric] 페이지에서 다음을 수행합니다.

   a. 지표 네임스페이스에 **`CloudTrailLogMetrics`**를 입력합니다.

b. 지표 이름에 **AttemptsToAccessDeletedSecrets**를 입력합니다.

c. 고급 지표 설정 표시를 선택한 후 필요에 따라 지표 값에 **1**을 입력합니다.

d. 필터 생성을 선택합니다.

7. 필터 상자에서 [Create Alarm]을 선택합니다.

8. [Create Alarm] 창에서 다음과 같이 실행합니다.

a. Name에 **AttemptsToAccessDeletedSecretsAlarm**를 입력합니다.

b. 다음 경우 항상:의 결과 값:에서 **>=**을 선택하고 **1**을 입력합니다.

c. [Send notification to:] 옆에서 다음 중 하나를 실시합니다.

- 새로운 Amazon SNS 주제를 생성해 사용하려면 새 목록(New list)을 선택한 후 새 주제 이름을 입력합니다. 메일 주소 목록(Email list):에 이메일 주소를 하나 이상 입력합니다. 쉼표로 구분하여 두 개 이상의 이메일 주소를 입력할 수 있습니다.
- 기존 Amazon SNS 주제를 사용하려면 사용할 주제의 이름을 선택합니다. 목록이 없으면 목록 선택을 선택합니다.

d. 경보 생성을 선택합니다.

3단계: CloudWatch 경보 테스트

경보를 테스트하려면 보안 암호를 생성한 다음 삭제를 예약합니다. 그런 다음 보안 암호 값을 검색해 봅니다. 잠시 후 경보에 구성한 주소로 이메일이 수신됩니다. 삭제하도록 예정된 보안 암호 사용에 대해 알려줍니다.

를 사용하여 AWS Secrets Manager 보안 암호의 규정 준수 모니터링 AWS Config

AWS Config 를 사용하여 보안 암호를 평가하여 보안 암호가 표준을 준수하는지 확인할 수 있습니다. AWS Config 규칙을 사용하여 보안 암호에 대한 내부 보안 및 규정 준수 요구 사항을 정의합니다. 그런 다음 AWS Config 는 규칙을 준수하지 않는 보안 암호를 식별할 수 있습니다. 또한 보안 암호 메타데이터, 교체 구성, 보안 암호 암호화에 사용되는 KMS 키, Lambda 교체 함수 및 보안 암호와 연결된 태그에 대한 변경 사항을 추적할 수 있습니다.

변경 사항을 알리 AWS Config 도록를 구성할 수 있습니다. 자세한 내용은 가 Amazon SNS 주제로 AWS Config 보내는 알림을 참조하세요.

여러 AWS 계정 및 조직에 보안 암호가 있는 경우 해당 구성 및 규정 준수 데이터를 집계할 수 AWS 리전 있습니다. 자세한 내용은 다중 계정 다중 리전 데이터 집계를 참조하세요.

보안 암호가 규정을 준수하는지 평가하는 방법

- AWS Config 규칙을 사용하여 리소스 평가에 대한 지침을 따르고 다음 규칙 중 하나를 선택합니다.
  - `secretsmanager-secret-unused` - 보안 암호를 지정된 일수 내에 액세스했는지 확인합니다.
  - `secretsmanager-using-cmk` - 보안 암호가 또는 생성한 고객 관리형 키를 사용하여 AWS 관리형 키 `aws/secretsmanager` 암호화되는지 확인합니다 AWS KMS.
  - `secretsmanager-rotation-enabled-check` - Secrets Manager에 저장된 보안 암호에 대해 교체가 구성되었는지 확인합니다.
  - `secretsmanager-scheduled-rotation-success-check`— 마지막으로 성공한 교체가 설정된 교체 주기 내에 있는지 확인합니다. 최소 확인 주기는 매일입니다.
  - `secretsmanager-secret-periodic-rotation` - 보안 암호를 지정된 일수 내에 교체했는지 확인합니다.

Secrets Manager 비용 모니터링

Amazon CloudWatch를 사용하여 예상 AWS Secrets Manager 요금을 모니터링할 수 있습니다. 자세한 내용은 CloudWatch 사용 설명서의 예상 AWS 요금을 모니터링하기 위한 결제 경보 생성을 참조하세요.

비용 모니터링을 위한 또 다른 옵션은 AWS 비용 이상 탐지입니다. 자세한 내용은 AWS 비용 관리 사용 설명서의 Detecting unusual spend with AWS Cost Anomaly Detection 섹션을 참조하세요.

Secrets Manager 사용량 모니터링에 대한 자세한 내용은 the section called “CloudWatch를 사용하여 모니터링” 및 the section called “를 사용하여 로그 AWS CloudTrail ” 섹션을 참조하세요.

AWS Secrets Manager 요금에 대한 자세한 내용은 단원을 참조하십시오the section called “가격 책정”.

Amazon GuardDuty로 위협 탐지

Amazon GuardDuty는 AWS 환경으로 계정, 컨테이너, 워크로드 및 데이터를 보호하는 데 도움이 되는 위협 탐지 서비스입니다. GuardDuty는 기계 학습(ML) 모델, 이상 및 위협 탐지 기능을 통해 다양한 로

그 소스를 지속적으로 모니터링하여 사용자 환경의 잠재적 보안 위험과 악의적 활동을 식별하고 우선 순위를 지정합니다. 예를 들어, GuardDuty는 잠재적인 위협을 탐지합니다. 이러한 위협에는 보안 암호에 대한 비정상적이거나 의심스러운 액세스, 그리고 자격 증명 유출 등이 포함됩니다. 자격 증명 유출을 탐지했다는 건 인스턴스 시작 역할을 통해 Amazon EC2 인스턴스 전용으로 자격 증명이 생성되었으나, AWS내의 다른 계정에서 이 자격 증명을 사용 중이라는 사실이 탐지되었다는 의미입니다. 자세한 내용은 Amazon GuardDuty 사용 설명서를 참조하세요.

탐지에 대한 또 다른 사용 사례는 이상 동작입니다. 예를 들어가 AWS Secrets Manager 일반적으로 Java SDK를 사용하여 엔터티에서 `create-secret`, `get-secret-value`describe-secret, 및 `list-secrets` 호출을 가져온 다음 다른 엔터티가 VPN 외부 AWS CLI 에서를 호출batch-get-secret-value하고 `get-secret-value` 사용하기 시작하는 경우 GuardDuty는 두 번째 엔터티가 API를 비정상적으로 호출하고 APIs. 자세한 내용은 GuardDuty IAM finding type CredentialAccess:IAMUser/AnomalousBehavior 섹션을 참조하세요.

에 대한 규정 준수 검증 AWS Secrets Manager

Secrets Manager 사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표 및 관련 법률과 규정에 따라 결정됩니다.는 규정 준수를 지원하기 위해 다음 리소스를 AWS 제공합니다.

- 보안 및 규정 준수 빠른 시작 안내서: 이 배포 안내서에서는 아키텍처 고려 사항에 관해 설명하고 AWS에서 보안 및 규정 준수에 중점을 둔 기본 환경을 배포하기 위한 단계를 제공합니다.
- HIPAA 보안 및 규정 준수를 위한 설계 백서 -이 백서에서는 기업이 AWS 를 사용하여 HIPAA 준수 애플리케이션을 생성하는 방법을 설명합니다.
- AWS 규정 준수 리소스 -이 워크북 및 가이드 모음은 산업 및 위치에 적용될 수 있습니다.
- AWS Config으로 리소스 구성이 내부 관행, 업계 지침 및 규정을 준수하는 정도를 평가할 수 있습니다. 자세한 내용은 the section called “보안 암호의 규정 준수 모니터링” 단원을 참조하십시오.
- AWS Security Hub CSPM는 보안 업계 표준 및 모범 사례 준수 여부를 확인하는 데 도움이 AWS 되는 내 보안 상태에 대한 포괄적인 보기를 제공합니다. Security Hub CSPM을 사용하여 Secrets Manager 리소스를 평가하는 방법에 대한 자세한 내용은 AWS Security Hub CSPM 사용 설명서의 AWS Secrets Manager 제어를 참조하세요.
- IAM Access Analyzer는 외부 엔터티가 보안 암호에 액세스할 수 있도록 허용하는 정책의 조건 문을 비롯해 정책을 분석합니다. 자세한 내용은 Access Analyzer를 사용하여 액세스 미리 보기를 참조하세요.
- AWS Systems Manager은 Secrets Manager에 대해 미리 정의된 실행서를 제공합니다. 자세한 내용은 Secrets Manager에 대한 Systems Manager Automation 실행서 참조를 참조하세요.
- 를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다 AWS Artifact. 자세한 내용은 에서 보고서 다운로드 AWS Artifact에서 .

규정 준수 표준

AWS Secrets Manager 는 다음 표준에 대한 감사를 거쳤으며 규정 준수 인증을 받아야 할 때 솔루션의 일부가 될 수 있습니다.

- HIPAA - AWS 를 HIPAA 적격 서비스로 포함하도록 AWS Secrets Manager HIPAA(Health Insurance Portability and Accountability Act) 규정 준수 프로그램을 확장했습니다. 와 BAA(Business Associate Agreement)를 체결한 경우 Secrets Manager를 사용하여 HIPAA 준수 애플리케이션을 빌드할 AWS수 있습니다.는 건강 정보의 처리 및 저장 AWS 에를 활용하는 방법에 대해 자세히 알고 싶은 고객에게 HIPAA 중심 백서를 AWS 제공합니다. 자세한 내용은 HIPAA 규정 준수를 참조하세요.

- PCI 참여 조직 - 서비스 공급자 수준 1에 PCI(지불 카드 산업 규정 준수 증명) DSS(데이터 보안 표준) 버전 3.2 AWS Secrets Manager 가 있습니다. AWS 제품 및 서비스를 사용하여 카드 소지자 데이터를 저장, 처리 또는 전송하는 고객은 자체 PCI DSS 규정 준수 인증을 관리할 AWS Secrets Manager 때를 사용할 수 있습니다. PCI 규정 준수 패키지의 사본을 요청하는 방법을 포함하여 AWS PCI DSS에 대한 자세한 내용은 PCI DSS 레벨 1을 참조하세요.
- ISO - AWS Secrets Manager 는 ISO/IEC 27001, ISO/IEC 27017, ISO/IEC 27018 및 ISO 9001에 대한 규정 준수 인증을 성공적으로 완료했습니다. 자세한 정보는 ISO 27001, ISO 27017, ISO 27018, ISO 9001을 참조하세요.
- AICPA SOC – 시스템 및 조직 제어(SOC) 보고서는 Secrets Manager가 주요 규정 준수 제어 기능을 어떻게 확보하고 목표를 어떻게 달성하고 있는지 입증하는 독립적 서드 파티 검사 기관의 심사 보고서입니다. 이러한 보고서의 목적은 사용자와 감사자가 운영 및 규정 준수를 지원하기 위해 설정된 AWS 제어를 이해하는 데 도움이 되는 것입니다. 자세한 내용은 SOC 규정 준수를 참조하세요.
- FedRAMP – FedRAMP(연방정부 위험 및 권한 부여 관리 프로그램)는 클라우드 제품 및 서비스의 보안 평가, 권한 부여 및 지속적인 모니터링에 대한 표준화된 접근 방식을 제공하는 정부 차원의 프로그램입니다. 또한 FedRAMP 프로그램은 동부/서부 및 GovCloud에서 정부 또는 규제 데이터를 사용할 수 있도록 서비스 및 리전에 대한 잠정 권한을 제공합니다. 자세한 내용은 FedRAMP 규정 준수를 참조하세요.
- 국방부 – 국방부(DoD) 클라우드 컴퓨팅 보안 요구 사항 안내서(SRG)에서는 DoD 고객에게 서비스를 제공하기 위해 클라우드 서비스 제공업체(CSP)가 DoD 임시 인증을 획득할 수 있도록 표준화된 평가 및 인증 프로세스를 제공합니다. 자세한 내용은 DoD SRG 리소스를 참조하세요.
- IRAP – IRAP(Information Security Registered Assessors Program)를 통해 호주 정부 고객은 적절한 제어가 되는지 검증하고 호주 사이버 보안 센터(ACSC)에서 제작한 호주 정부 ISM(Information Security Manual)의 요구 사항을 해결하기 위한 적절한 책임 모델을 결정할 수 있습니다. 자세한 내용은 IRAP 리소스를 참조하세요.
- OSPAR - Amazon Web Services(AWS)는 아웃소싱 서비스 공급자의 감사 보고서(OSPAR) 인증. AWS Association of Banks in Singapore(ABS) Guidelines on Control Objectives and Procedures for Outsourced Service Providers(ABS Guidelines)에 따라 싱가포르 금융 서비스 산업에서 설정한 클라우드 서비스 공급자에 대한 높은 기대치를 충족하려는 고객의 AWS 노력을 입증했습니다. 자세한 내용은 OSPAR 리소스를 참조하세요.

의 보안 AWS Secrets Manager

의 보안 AWS 이 최우선 순위입니다. AWS 고객은 보안에 가장 민감한 조직의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 이점을 누릴 수 있습니다.

사용자와 AWS 는 보안에 대한 책임을 공유합니다. 공동 책임 모델은 이 사항을 클라우드 내 보안 및 클라우드의 보안으로 설명합니다.

- 클라우드 보안 - AWS 는 AWS 클라우드에서 AWS 서비스를 실행하는 인프라를 보호할 책임이 있습니다. AWS 또한는 안전하게 사용할 수 있는 서비스를 제공합니다. 서드 파티 감사자는 정기적으로 AWS 규정 준수 프로그램의 일환으로 보안 효과를 테스트하고 검증합니다. 에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 AWS 규정 준수 프로그램 제공 범위 내 서비스를 AWS Secrets Manager참조하세요.
- 클라우드의 보안 - AWS 서비스가 사용자의 책임을 결정합니다. 또한 귀하는 귀사의 데이터 민감도, 귀사의 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

더 많은 리소스를 보려면 보안 원칙 – AWS Well-Architected 프레임워크를 참조하세요.

주제

- 를 사용하여 AWS Secrets Manager 보안 암호를 AWS CLI 저장할 때의 위험 완화
- 에 대한 인증 및 액세스 제어 AWS Secrets Manager
- 의 데이터 보호 AWS Secrets Manager
- 의 보안 암호 암호화 및 복호화 AWS Secrets Manager
- 의 인프라 보안 AWS Secrets Manager
- AWS Secrets Manager VPC 엔드포인트 사용
- IAM 정책을 사용하여 API 액세스 제어
- 의 복원력 AWS Secrets Manager
- 포스트 양자 TLS

를 사용하여 AWS Secrets Manager 보안 암호를 AWS CLI 저장할 때의 위험 완화

AWS Command Line Interface (AWS CLI)를 사용하여 AWS 작업을 호출하는 경우 명령 셸에 해당 명령을 입력합니다. 예를 들어 Windows 명령 프롬프트나 Windows PowerShell 또는 Bash나 Z shell 등을

사용할 수 있습니다. 이러한 명령 셸 대부분에는 생산성을 높이기 위해 설계된 기능이 포함되어 있습니다. 하지만, 이 기능은 보안 암호의 보안을 약화시키는 데 사용될 수 있습니다. 예를 들어 대부분의 셸에서 위쪽 화살표 키를 사용하면 마지막으로 입력한 명령을 볼 수 있습니다. 명령 기록 기능은 보호되지 않는 세션에 액세스한 누군가가 악용할 수 있습니다. 또한 백그라운드에서 작동하는 다른 기능이 명령 파라미터에 액세스할 수 있습니다. 이러한 기능은 모두 작업을 보다 효율적으로 수행하도록 지원하기 위한 것입니다. 이러한 위험을 줄이기 위해서는 다음 단계를 수행해야 합니다.

- 콘솔에서 자리를 비우는 경우에는 항상 컴퓨터를 잠급니다.
- 필요 없거나 더 이상 사용하지 않는 콘솔 유틸리티는 제거하거나 비활성화합니다.
- 셸 또는 원격 액세스 프로그램(사용하는 경우)에서 입력한 명령을 로깅하지 않도록 합니다.
- 셸 명령 기록에서 캡처하지 않는 파라미터를 전달하는 기술을 사용합니다. 다음 예제에서는 보안 암호 텍스트를 텍스트 파일에 입력한 다음 파일을 AWS Secrets Manager 명령에 전달하고 파일을 즉시 삭제하는 방법을 보여줍니다. 이는 일반적인 셸 기록에 보안 암호 텍스트가 캡처되지 않는다는 것을 의미합니다.

  다음 예에는 일반적인 Linux 명령이 나와 있습니다. 사용 중인 셸에 약간 다른 명령이 필요할 수 있습니다.

```
$ touch secret.txt
        # Creates an empty text file
$ chmod go-rx secret.txt
        # Restricts access to the file to only the user
$ cat > secret.txt
        # Redirects standard input (STDIN) to the text file
ThisIsMyTopSecretPassword^D
        # Everything the user types from this point up to the CTRL-D (^D) is saved in
 the file
$ aws secretsmanager create-secret --name TestSecret --secret-string file://
secret.txt       # The Secrets Manager command takes the --secret-string parameter
 from the contents of the file
$ shred -u secret.txt
        # The file is destroyed so it can no longer be accessed.
```

이러한 명령을 실행한 후에는 위쪽 및 아래쪽 화살표를 사용해 명령 기록을 스크롤하고 보안 암호 텍스트가 임의의 행에 표시되지 않았는지 확인할 수 있어야 합니다.

> **Important**
>
> 기본적으로 명령 기록 버퍼 크기를 먼저 1로 줄이지 않으면 Windows에서는 동일한 기능을 수행할 수 없습니다.

Windows 명령 프롬프트를 1개의 명령 기록 버퍼만 갖도록 구성하려면

1. 관리자 명령 프롬프트(관리자로 실행)를 엽니다.
2. 왼쪽 위에 있는 아이콘을 선택한 다음 속성을 선택합니다.
3. 옵션 탭에서 버퍼 크기 및 버퍼 수를 둘 다 **1**로 설정하고 확인을 선택합니다.
4. 기록에 표시하지 않으려는 명령을 입력해야 할 때마다 다음과 같은 다른 명령 하나를 즉시 입력합니다.

   ```
   echo.
   ```

   그러면 민감한 명령이 플러시됩니다.

Windows 명령 프롬프트 셸에서 SysInternals SDelete 도구를 다운로드하고 다음과 유사한 명령을 사용할 수 있습니다.

```
C:\> echo. 2> secret.txt
        # Creates an empty file
C:\> icacls secret.txt /remove "BUILTIN\Administrators" "NT AUTHORITY/SYSTEM" /
inheritance:r   # Restricts access to the file to only the owner
C:\> copy con secret.txt /y
        # Redirects the keyboard to text file, suppressing prompt to overwrite
THIS IS MY TOP SECRET PASSWORD^Z
      # Everything the user types from this point up to the CTRL-Z (^Z) is saved in the
 file
C:\> aws secretsmanager create-secret --name TestSecret --secret-string file://
secret.txt      # The Secrets Manager command takes the --secret-string parameter from
 the contents of the file
C:\> sdelete secret.txt
        # The file is destroyed so it can no longer be accessed.
```

에 대한 인증 및 액세스 제어 AWS Secrets Manager

Secrets Manager는 AWS Identity and Access Management (IAM)를 사용하여 보안 암호에 대한 액세스를 제어합니다. IAM은 인증 및 액세스 제어를 제공합니다. 인증은 개인 요청의 자격 증명을 확인합니다. Secrets Manager는 암호, 액세스 키 및 멀티 팩터 인증(MFA) 토큰을 통해 로그인 프로세스를 사용하여 사용자의 자격 증명을 확인합니다. 로그인을 참조하세요 AWS. 액세스 제어는 승인된 개인만 AWS 리소스(예: 보안 암호)에 대한 작업을 수행할 수 있도록 합니다. Secrets Manager는 정책을 사용하여 리소스에 액세스할 수 있는 사용자 및 해당 리소스에 대해 자격 증명이 수행할 수 있는 작업을 정의합니다. IAM의 정책 및 권한을 참조하세요.

주제

- 에 대한 권한 참조 AWS Secrets Manager
- Secrets Manager 관리자 권한
- 보안 암호에 액세스할 수 있는 권한
- Lambda 교체 함수에 대한 권한
- 암호화 키 권한
- 복제 권한
- ID 기반 정책
- 리소스 기반 정책
- 속성 기반 액세스 제어(ABAC)를 사용하여 보안 암호에 대한 액세스 제어
- AWS 에 대한 관리형 정책 AWS Secrets Manager
- AWS Secrets Manager 보안 암호에 대한 권한이 있는 사용자 확인
- 다른 계정에서 AWS Secrets Manager 보안 암호 액세스
- 온프레미스 환경에서 보안 암호에 액세스

에 대한 권한 참조 AWS Secrets Manager

Secrets Manager의 권한 참조를 보려면 서비스 승인 참조의 Actions, resources, and condition keys for AWS Secrets Manager 섹션을 참조하세요.

Secrets Manager 관리자 권한

Secrets Manager 관리자 권한을 부여하려면 IAM 자격 증명 권한 추가 및 제거의 지침을 따르고 다음 정책을 연결합니다.

- SecretsManagerReadWrite
- IAMFullAccess

최종 사용자에게는 관리자 권한을 부여하지 않는 것이 좋습니다. 이를 통해 사용자는 자신의 보안 암호를 생성하고 관리할 수 있지만 교체를 활성화하는 데 필요한 권한(IAMFullAccess)은 최종 사용자에게 적합하지 않은 중요한 권한을 부여합니다.

보안 암호에 액세스할 수 있는 권한

IAM 권한 정책을 사용하여 보안 암호에 액세스하는 사용자 또는 서비스를 제어할 수 있습니다. 권한 정책은 누가 어떤 리소스에 대해 어떤 작업을 수행할 수 있는지 설명합니다. 다음을 수행할 수 있습니다.

- the section called “ID 기반 정책”
- the section called “리소스 기반 정책”

Lambda 교체 함수에 대한 권한

Secrets Manager는 AWS Lambda 함수를 사용하여 보안 암호를 교체합니다. Lambda 함수는 보안 암호에 자격 증명이 포함된 데이터베이스 또는 서비스뿐 아니라 보안 암호에 액세스할 수 있어야 합니다. 교체 권한을(를) 참조하세요.

암호화 키 권한

Secrets Manager는 AWS Key Management Service (AWS KMS) 키를 사용하여 보안 암호를 암호화합니다. 에는 AWS 관리형 키 `aws/secretsmanager` 자동으로 올바른 권한이 있습니다. 다른 KMS 키를 사용하는 경우 Secrets Manager는 해당 키에 대한 권한이 필요합니다. the section called “KMS 키에 대한 권한”을(를) 참조하세요.

복제 권한

IAM 권한 정책을 사용하여 보안 암호를 다른 리전에 복제할 수 있는 사용자 또는 서비스를 제어할 수 있습니다. the section called “복제 방지”을(를) 참조하세요.

ID 기반 정책

권한 정책을 IAM 자격 증명: 사용자, 사용자, 그룹 및 역할에 연결합니다. 자격 증명 기반 정책에서 자격 증명에 액세스할 수 있는 보안 암호와, 자격 증명이 보안 암호에 대해 수행할 수 있는 작업을 지정합니다. 자세한 내용은 IAM 자격 증명 권한 추가 및 제거를 참조하세요.

다른 서비스의 애플리케이션이나 사용자를 나타내는 역할에 권한을 부여할 수 있습니다. 예를 들어 Amazon EC2 인스턴스에서 실행되는 애플리케이션은 데이터베이스에 액세스해야 할 수 있습니다. EC2 인스턴스 프로파일에 연결된 IAM 역할을 생성한 후 권한 정책을 사용하여 데이터베이스의 보안 인증 정보를 포함한 보안 암호에 대한 액세스 권한을 역할에 부여할 수 있습니다. 자세한 내용은 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여를 참조하세요. 역할을 연결할 수 있는 기타 서비스에는 Amazon Redshift, AWS Lambda, Amazon ECS가 있습니다.

IAM 이외의 자격 증명 시스템으로 인증된 사용자에게 권한을 부여할 수도 있습니다. 예를 들어 IAM 역할을 Amazon Cognito로 로그인하는 모바일 앱 사용자와 연결할 수 있습니다. 이 역할은 앱에 역할 권한 정책에 있는 권한과 함께 임시 자격 증명을 부여합니다. 그런 다음 권한 정책을 사용하여 해당 역할에 보안 암호에 대한 액세스 권한을 부여할 수 있습니다. 자세한 내용은 Identity providers and federation(자격 증명 공급자 및 페더레이션)을 참조하세요.

자격 증명 기반 정책을 사용하여 다음을 수행할 수 있습니다.

- 여러 보안 암호에 대한 자격 증명 액세스 권한을 부여합니다.
- 새 보안 암호를 만들 수 있는 사용자와, 아직 생성되지 않은 보안 암호에 액세스할 수 있는 사용자를 제어합니다.
- IAM 그룹에 보안 암호에 대한 액세스 권한을 부여합니다.

예시:

- 예: 개별 보안 암호 값을 검색할 수 있는 권한
- 예: 개별 보안 암호를 읽고 설명할 수 있는 권한
- 예: 보안 암호 값 그룹을 일괄적으로 검색할 수 있는 권한
- 예: 와일드카드
- 예: 보안 암호 생성 권한
- 예: 보안 암호를 암호화하기 위한 특정 AWS KMS 키 거부

예: 개별 보안 암호 값을 검색할 수 있는 권한

보안 암호 값을 검색할 수 있는 권한을 부여하기 위해 정책을 암호 또는 자격 증명에 연결할 수 있습니다. 사용할 정책 유형을 결정하는 방법에 대한 도움말은 자격 증명 기반 정책 및 리소스 기반 정책을 참조하세요. 정책 연결 방법에 대한 자세한 내용은 the section called “리소스 기반 정책” 및 the section called “ID 기반 정책” 섹션을 참조하세요.

이 예시는 IAM 그룹에 대한 액세스 권한을 부여하려는 경우에 유용합니다. 배치 API 직접 호출에서 보안 암호 그룹을 검색할 권한을 부여하려면 the section called “예: 보안 암호 값 그룹을 일괄적으로 검색할 수 있는 권한” 섹션을 참조하세요.

Example고객 관리형 키를 사용하여 암호화된 보안 암호 읽기

고객 관리형 키를 사용하여 보안 암호를 암호화하는 경우 다음 정책을 자격 증명에 연결하여 보안 암호를 읽을 액세스 권한을 부여할 수 있습니다.

JSON

```
{
"Version":"2012-10-17",
"Statement": [
{
  "Effect": "Allow",
  "Action": "secretsmanager:GetSecretValue",
  "Resource": "arn:aws:secretsmanager:us-east-1:123456789012:secret:secretName-AbCdEf"
},
{
  "Effect": "Allow",
  "Action": "kms:Decrypt",
  "Resource": "arn:aws:kms:us-east-1:123456789012:key/key-id"
}
]
}
```

예: 개별 보안 암호를 읽고 설명할 수 있는 권한

Example하나의 보안 암호를 읽고 설명

자격 증명에 다음 정책을 연결하여 보안 암호에 대한 액세스 권한을 부여할 수 있습니다.

JSON

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

```
    "Effect": "Allow",
    "Action": [
      "secretsmanager:GetSecretValue",
      "secretsmanager:DescribeSecret"
      ],
      "Resource": "arn:aws:secretsmanager:us-
east-1:123456789012:secret:secretName-AbCdEf"
    }
]
}
```

예: 보안 암호 값 그룹을 일괄적으로 검색할 수 있는 권한

Example일괄적으로 보안 암호 그룹 읽기

자격 증명에 다음 정책을 연결하여 배치 API 직접 호출에서 보안 암호 그룹을 검색할 수 있는 액세스 권한을 부여할 수 있습니다. 이 정책은 배치 호출에 다른 보안 암호가 포함되어 있더라도 *SecretARN1*, *SecretARN2* 및 *SecretARN3*으로 지정한 보안 암호만 검색할 수 있도록 호출자를 제한합니다. 호출자가 배치 API 직접 호출에서 다른 보안 암호도 요청하는 경우 Secrets Manager는 이를 반환하지 않습니다. 자세한 내용은 BatchGetSecretValue 섹션을 참조하세요.

JSON

```
{
"Version":"2012-10-17",
"Statement": [
{
  "Effect": "Allow",
  "Action": [
    "secretsmanager:BatchGetSecretValue",
    "secretsmanager:ListSecrets"
  ],
  "Resource": "*"
},
{
  "Effect": "Allow",
  "Action": [
    "secretsmanager:GetSecretValue"
  ],
  "Resource": [
    "arn:aws:secretsmanager:us-east-1:123456789012:secret:secretName1-AbCdEf",
```

```
    "arn:aws:secretsmanager:us-east-1:123456789012:secret:secretName2-AbCdEf",
    "arn:aws:secretsmanager:us-east-1:123456789012:secret:secretName3-AbCdEf"
    ]
}
]
}
```

예: 와일드카드

와일드카드를 사용하여 정책 요소에 값 집합을 포함할 수 있습니다.

Example경로의 모든 보안 암호에 액세스

다음 정책은 "*TestEnv/*"로 시작하는 이름의 모든 보안 암호를 검색할 수 있는 액세스 권한을 부여합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "secretsmanager:GetSecretValue",
        "Resource": "arn:aws:secretsmanager:us-
east-1:123456789012:secret:TestEnv/*"
    }
}
```

Example모든 보안 암호의 메타데이터에 액세스

다음 정책은 `DescribeSecret` 및 `List: ListSecrets, ListSecretVersionIds`로 시작하는 권한을 부여합니다.

JSON

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

```
"Action": [
  "secretsmanager:DescribeSecret",
  "secretsmanager:List*"
],
"Resource": "*"
}
}
```

Example보안 암호 이름 일치

다음 정책은 보안 암호에 대한 모든 Secrets Manager 권한을 해당 이름으로 부여합니다. 이 정책을 사용하려면 the section called “ID 기반 정책” 섹션을 참조하세요.

보안 암호 이름을 일치시키려면 리전, 계정 ID, 보안 암호 이름 및 와일드카드(?)를 조합하여 임의의 개별 문자와 일치시켜 보안 암호에 대한 ARN을 생성합니다. Secrets Manager는 보안 암호 이름에 6개의 임의 문자를 ARN의 일부로 추가하므로 이 와일드카드를 사용하여 해당 문자와 일치시킬 수 있습니다. `"another_secret_name-*"` 구문을 사용하는 경우 Secrets Manager는 6개의 임의 문자와 해당 보안 암호를 일치시킬 뿐 아니라 `"another_secret_name-<anything-here>a1b2c3"`과도 일치시킵니다.

6개의 임의 문자를 제외한 보안 암호 ARN의 모든 부분을 예측할 수 있기 때문에 와일드카드 문자 `'??????'` 구문을 사용하면 아직 존재하지 않는 보안 암호에 안전하게 권한을 부여할 수 있습니다. 하지만 보안 암호를 삭제한 후 이름이 동일한 보안 암호를 다시 생성할 경우, 임의의 문자 6자가 변경되더라도 새 보안 암호에 대한 권한이 사용자에게 자동으로 부여됩니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "secretsmanager:*",
            "Resource": [
                "arn:aws:secretsmanager:us-
east-1:123456789012:secret:a_specific_secret_name-a1b2c3",
                "arn:aws:secretsmanager:us-
east-1:123456789012:secret:another_secret_name-??????"
            ]
        }
```

```
    ]
}
```

예: 보안 암호 생성 권한

사용자에게 보안 암호 생성 권한을 부여하려면 사용자가 속한 IAM 그룹에 권한 정책을 연결하는 것이 좋습니다. IAM 사용자 그룹을 참조하세요.

Example보안 암호 생성

다음 정책은 보안 암호를 생성하고 보안 암호 목록을 볼 수 있는 권한을 부여합니다. 이 정책을 사용하려면 the section called “ID 기반 정책” 섹션을 참조하세요.

JSON

```
{
"Version":"2012-10-17",
"Statement": [
{
  "Effect": "Allow",
  "Action": [
    "secretsmanager:CreateSecret",
    "secretsmanager:ListSecrets"
  ],
  "Resource": "*"
}
]
}
```

예: 보안 암호를 암호화하기 위한 특정 AWS KMS 키 거부

> **Important**
>
> 고객 관리형 키를 거부하려면 키 정책 또는 키 부여를 사용하여 액세스를 제한하는 것이 좋습니다. 자세한 내용은 AWS Key Management Service 개발자 가이드의 Authentication and access control for AWS KMS를 참조하세요.

Example AWS 관리형 키 거부 **aws/secretsmanager**

다음 정책은 보안 암호 생성 또는 업데이트를 위한 사용을 AWS 관리형 키 aws/secretsmanager 거부합니다. 이 정책은 보안 암호가 반드시 고객 관리형 키로 암호화되도록 요구합니다. 이 정책에는 두 가지 문이 포함됩니다.

1. 첫 번째 문인는 Sid: "RequireCustomerManagedKeysOnSecrets"를 사용하여 보안 암호를 생성하거나 업데이트하기 위한 요청을 거부합니다 AWS 관리형 키 aws/secretsmanager.
2. 두 번째 문인는 Secrets Manager가 기본적으로를 사용하기 때문에 KMS 키가 포함되지 않은 보안 암호 생성 요청을 Sid: "RequireKmsKeyIdParameterOnCreate"거부합니다 AWS 관리형 키 aws/secretsmanager.

JSON

```
{
"Version":"2012-10-17",
"Statement": [
  {
      "Sid": "RequireCustomerManagedKeysOnSecrets",
      "Effect": "Deny",
      "Action": [
          "secretsmanager:CreateSecret",
          "secretsmanager:UpdateSecret"
      ],
      "Resource": "*",
      "Condition": {
          "StringLikeIfExists": {
              "secretsmanager:KmsKeyArn": "<key_ARN_of_the_AWS_managed_key>"
          }
      }
  },
  {
      "Sid": "RequireKmsKeyIdParameterOnCreate",
      "Effect": "Deny",
      "Action": "secretsmanager:CreateSecret",
      "Resource": "*",
      "Condition": {
          "Null": {
              "secretsmanager:KmsKeyArn": "true"
          }
      }
```

```
  }
]
}
```

리소스 기반 정책

리소스 기반 정책에서 보안 암호에 액세스할 수 있는 사용자와, 보안 암호에 대해 수행할 수 있는 작업을 지정합니다. 리소스 기반 정책을 사용하여 다음을 수행할 수 있습니다.

- 여러 사용자 및 역할에 단일 보안 암호에 대한 액세스 권한을 부여합니다.
- 다른 AWS 계정의 사용자 또는 역할에 액세스 권한을 부여합니다.

리소스 기반 정책을 콘솔의 보안 암호에 연결하면 Secret Secrets Manager는 자동화된 추론 엔진 Zelkova 및 API `ValidateResourcePolicy`을(를) 사용하여 다양한 IAM 보안 주체에 보안 암호에 대한 액세스 권한을 부여하지 못하도록 방지할 수 있습니다. 또는 CLI나 SDK에서 `BlockPublicPolicy` 파라미터가 있는 `PutResourcePolicy` API 를 호출할 수 있습니다.

> **Important**
>
> 리소스 정책 검증 및 `BlockPublicPolicy` 파라미터는 보안 암호에 직접 연결된 리소스 정책을 통해 퍼블릭 액세스 권한이 부여되지 않도록 하여 리소스를 보호하도록 지원합니다. 이러한 기능을 사용하는 것 외에도 다음 정책을 주의 깊게 검토하여 퍼블릭 액세스를 허용하지 않는지 확인하세요.
>
> - 연결된 AWS 보안 주체에 연결된 자격 증명 기반 정책(예: IAM 역할)
> - 연결된 리소스에 연결된 AWS 리소스 기반 정책(예: AWS Key Management Service (AWS KMS) 키)
>
> 보안 암호에 대한 권한을 검토하려면 보안 암호에 대한 권한이 있는 사용자 확인 섹션을 참조하세요.

보안 암호(콘솔)에 대한 리소스 정책을 보거나 변경하거나 삭제하려면

1. https://console.aws.amazon.com/secretsmanager/에서 Secrets Manager 콘솔을 엽니다.
2. 보안 암호 목록에서 보안 암호를 선택합니다.

3. 보안 암호 세부 정보 페이지의 개요 탭에 있는 리소스 권한 섹션에서 권한 편집을 선택합니다.
4. 코드 필드에서 다음 중 하나를 수행한 후 저장(Save)을 선택합니다.
   - 리소스 정책을 연결하거나 수정하려면 정책을 입력합니다.
   - 정책을 삭제하려면 코드 필드를 지웁니다.

AWS CLI

Example리소스 정책의 검색

다음 get-resource-policy 예시에서는 보안 암호에 연결된 리소스 기반 정책을 검색합니다.

```
aws secretsmanager get-resource-policy \
    --secret-id MyTestSecret
```

Example리소스 정책 삭제

다음 delete-resource-policy 예시에서는 보안 암호에 연결된 리소스 기반 정책을 삭제합니다.

```
aws secretsmanager delete-resource-policy \
    --secret-id MyTestSecret
```

Example리소스 정책 추가

다음 put-resource-policy 예시에서는 보안 암호에 사용 권한 정책을 추가하여 해당 정책이 암호에 대한 광범위한 액세스를 제공하지 않는지 먼저 확인합니다. 파일에서 해당 정책을 읽습니다. 자세한 내용은 AWS CLI 사용 설명서의 파일에서 AWS CLI 파라미터 로드를 참조하세요.

```
aws secretsmanager put-resource-policy \
    --secret-id MyTestSecret \
    --resource-policy file://mypolicy.json \
    --block-public-policy
```

mypolicy.json의 콘텐츠:

JSON

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

```
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:role/MyRole"
            },
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "*"
        }
    ]
}
```

AWS SDK

보안 암호에 연결된 정책을 검색하려면 `GetResourcePolicy`를 사용합니다.

보안 암호에 연결된 정책을 삭제하려면 `DeleteResourcePolicy`를 사용합니다.

보안 암호에 정책을 연결하려면 `PutResourcePolicy`를 사용합니다. 이미 정책이 연결되어 있는 경우 이 명령은 해당 정책을 새 정책으로 대체합니다. 정책 문서는 JSON 구조의 텍스트 형식이어야 합니다. JSON 정책 문서 구조를 참조하세요.

자세한 내용은 the section called “AWS SDKs” 단원을 참조하십시오.

예제

예시:

- 예: 개별 보안 암호 값을 검색할 수 있는 권한
- 예: 권한 및 VPC
- 예: 서비스 보안 주체

예: 개별 보안 암호 값을 검색할 수 있는 권한

보안 암호 값을 검색할 수 있는 권한을 부여하기 위해 정책을 암호 또는 자격 증명에 연결할 수 있습니다. 사용할 정책 유형을 결정하는 방법에 대한 도움말은 자격 증명 기반 정책 및 리소스 기반 정책을 참조하세요. 정책 연결 방법에 대한 자세한 내용은 the section called “리소스 기반 정책” 및 the section called “ID 기반 정책” 섹션을 참조하세요.

이 예시는 여러 사용자 또는 역할에 단일 보안 암호에 대한 액세스 권한을 부여하려는 경우에 유용합니다. 배치 API 직접 호출에서 보안 암호 그룹을 검색할 권한을 부여하려면 the section called “예: 보안 암호 값 그룹을 일괄적으로 검색할 수 있는 권한” 섹션을 참조하세요.

Example보안 암호 1개 읽기

다음 정책을 보안 암호에 연결하여 보안 암호에 대한 액세스 권한을 부여할 수 있습니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/EC2RoleToAccessSecrets"
            },
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "*"
        }
    ]
}
```

예: 권한 및 VPC

VPC 내에서 Secrets Manager에 액세스해야 하는 경우 권한 정책에 조건을 포함시켜 Secrets Manager에 대한 요청이 VPC에서 전송되도록 할 수 있습니다. 자세한 내용은 VPC 엔드포인트 조건이 있는 요청 제한 및 the section called “VPC 엔드포인트(AWS PrivateLink)” 섹션을 참조하세요.

다른 AWS 서비스의 보안 암호에 대한 액세스 요청도 VPC에서 가져와야 합니다. 그렇지 않으면이 정책이 액세스를 거부합니다.

Example VPC 엔드포인트를 통해 요청되도록 요구

다음 정책은 요청이 VPC 엔드포인트 *vpce-1234a5678b9012c*를 통해 이루어질 때만 사용자가 Secrets Manager 작업을 수행할 수 있도록 허용합니다.

JSON

```
{
"Id": "example-policy-1",
"Version":"2012-10-17",
"Statement": [
```

```
{
  "Sid": "RestrictGetSecretValueoperation",
  "Effect": "Deny",
  "Principal": "*",
  "Action": "secretsmanager:GetSecretValue",
  "Resource": "*",
  "Condition": {
    "StringNotEquals": {
      "aws:sourceVpce": "vpce-12345678"
    }
  }
}
]
}
```

Example VPC를 통해 요청되도록 요구

다음 정책은 명령이 *vpc-12345678*에서 이루어진 경우에만 보안 암호를 생성 및 관리하는 명령을 허용합니다. 또한 정책에서는 요청이 vpc-2b2b2b2b에서 이루어진 경우에만 보안 암호와 암호화된 값에 액세스하는 작업을 허용합니다. 애플리케이션이 하나의 VPC에서 실행 중이지만 관리 용도로 두 번째 격리된 VPC를 사용하는 경우, 이와 같은 정책을 사용할 수 있습니다.

JSON

```
{
"Id": "example-policy-2",
"Version":"2012-10-17",
"Statement": [
{
  "Sid": "AllowAdministrativeActionsfromONLYvpc-12345678",
  "Effect": "Deny",
  "Principal": "*",
  "Action": [
    "secretsmanager:Create*",
    "secretsmanager:Put*",
    "secretsmanager:Update*",
    "secretsmanager:Delete*",
    "secretsmanager:Restore*",
    "secretsmanager:RotateSecret",
    "secretsmanager:CancelRotate*",
    "secretsmanager:TagResource",
```

```
      "secretsmanager:UntagResource"
    ],
    "Resource": "*",
    "Condition": {
      "StringNotEquals": {
        "aws:sourceVpc": "vpc-12345678"
      }
    }
  },
  {
    "Sid": "AllowSecretValueAccessfromONLYvpc-2b2b2b2b",
    "Effect": "Deny",
    "Principal": "*",
    "Action": [
      "secretsmanager:GetSecretValue"
    ],
    "Resource": "*",
    "Condition": {
      "StringNotEquals": {
        "aws:sourceVpc": "vpc-2b2b2b2b"
      }
    }
  }
  ]
  }
```

예: 서비스 보안 주체

보안 암호에 연결된 리소스 정책에 AWS 서비스 보안 주체가 포함된 경우 aws:SourceArn 및 aws:SourceAccount 전역 조건 키를 사용하는 것이 좋습니다. ARN 및 계정 값은 요청이 다른 AWS 서비스에서 Secrets Manager로 오는 경우에만 권한 부여 컨텍스트에 포함됩니다. 이러한 조건 조합은 잠재적 혼동된 대리자 시나리오를 방지합니다.

리소스 ARN에 리소스 정책에서 허용되지 않는 문자가 포함된 경우, 해당 리소스 ARN을 `aws:SourceArn` 조건 키의 값에 사용할 수 없습니다. 대신 `aws:SourceAccount` 조건 키를 사용합니다. 자세한 내용은 IAM 요구 사항을 참조하세요.

서비스 보안 주체는 일반적으로 보안 암호에 연결된 정책에서 보안 주체로 사용되지 않지만 일부 AWS 서비스에서는 보안 주체가 필요합니다. 서비스에서 보안 암호에 연결하도록 요구하는 리소스 정책에 대한 자세한 내용은 서비스 설명서를 참조하세요.

Example서비스가 서비스 보안 주체를 사용하여 보안 암호에 액세스하도록 허용

JSON

```
{
"Version":"2012-10-17",
"Statement": [
{
  "Effect": "Allow",
  "Principal": {
    "Service": [
      "s3.amazonaws.com"
    ]
  },
  "Action": "secretsmanager:GetSecretValue",
  "Resource": "*",
  "Condition": {
    "ArnLike": {
      "aws:sourceArn": "arn:aws:s3::123456789012:*"
    },
    "StringEquals": {
      "aws:sourceAccount": "123456789012"
    }
  }

}
]
}
```

속성 기반 액세스 제어(ABAC)를 사용하여 보안 암호에 대한 액세스 제어

속성 기반 액세스 제어(ABAC)는 사용자, 데이터 또는 환경의 속성 또는 특성(예: 부서, 사업부 또는 권한 부여 결과에 영향을 미칠 수 있는 기타 요인)을 기준으로 권한을 정의하는 권한 부여 전략입니다. 에서는 AWS이러한 속성을 태그라고 합니다.

태그를 사용하여 권한을 제어하면 빠르게 성장하는 환경에서 유용하며 정책 관리가 번거로운 상황에 도움이 됩니다. ABAC 규칙은 런타임 시 동적으로 평가되므로, 사용자의 애플리케이션 및 데이터에 대한 액세스 권한과 허용되는 작업 유형은 정책의 컨텍스트 요인에 따라 자동으로 변경됩니다. 예를 들어 사용자가 부서를 변경할 경우, 권한을 업데이트하거나 새 역할을 요청하지 않아도 액세스 권한이 자동으로 조정됩니다. 자세한 내용은 ABAC란 무엇입니까 AWS?, 태그를 기반으로 보안 암호에 액세스할

수 있는 권한 정의, IAM Identity Center와 함께 ABAC를 사용하여 Secrets Manager에 대한 권한 부여 요구 사항 조정을 참조하세요.

예: 특정 태그가 있는 보안 암호에 대한 ID 액세스 허용

다음 정책은 키가 *ServerName*이고 값이 *ServerABC*인 태그가 포함된 보안 암호에 대해 DescribeSecret 액세스를 허용합니다. 이 정책을 ID에 연결하면 해당 ID는 계정 내에서 해당 태그가 있는 모든 보안 암호에 대한 권한을 갖습니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
  "Effect": "Allow",
  "Action": "secretsmanager:DescribeSecret",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "secretsmanager:ResourceTag/ServerName": "ServerABC"
      }
    }
  }
}
```

예: 보안 암호의 태그와 일치하는 태그를 사용하여 ID에 대해서만 액세스 허용

다음 정책은 ID의 *AccessProject* 태그 값이 보안 암호의 *AccessProject* 태그 값과 동일한 경우, 계정의 모든 ID가 계정의 모든 보안 암호에 대한 GetSecretValue 액세스 권한을 갖도록 허용합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
  "Effect": "Allow",
  "Principal": {
    "AWS": "123456789012"
  },
  "Condition": {
```

```
    "StringEquals": {
      "aws:ResourceTag/AccessProject": "${ aws:PrincipalTag/AccessProject }"
    }
  },
  "Action": "secretsmanager:GetSecretValue",
  "Resource": "*"
  }
}
```

AWS 에 대한 관리형 정책 AWS Secrets Manager

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 미칩니다. 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: SecretsManagerReadWrite

이 정책은 Amazon RDS AWS Secrets Manager, Amazon Redshift 및 Amazon DocumentDB 리소스를 설명할 수 있는 권한과를 사용하여 보안 암호를 암호화 및 해독 AWS KMS 할 수 있는 권한을 포함하여 에 대한 읽기/쓰기 액세스를 제공합니다. 또한이 정책은 AWS CloudFormation 변경 세트를 생성하고, 에서 관리하는 Amazon S3 버킷에서 교체 템플릿을 가져오고 AWS, AWS Lambda 함수를 나열하고, Amazon EC2 VPCs 설명할 수 있는 권한을 제공합니다. 콘솔에서 기존 교체 함수를 사용하여 교체를 설정하려면 이러한 권한이 필요합니다.

새 교체 함수를 생성하려면 AWS CloudFormation 스택 및 AWS Lambda 실행 역할을 생성할 수 있는 권한도 있어야 합니다. IAMFullAccess 관리형 정책을 할당할 수 있습니다. 교체 권한을(를) 참조하세요.

권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

- `secretsmanager` – 보안 주체가 모든 Secrets Manager 작업을 수행할 수 있도록 허용합니다.
- `cloudformation` - 보안 주체가 CloudFormation 스택을 생성할 수 있도록 허용합니다. 이는 콘솔을 사용하여 교체를 켜는 보안 주체가 CloudFormation 스택을 통해 Lambda 교체 함수를 생성할 수 있도록 하기 위해 필요합니다. 자세한 내용은 the section called “Secrets Manager의 사용 방식 CloudFormation” 단원을 참조하십시오.
- `ec2` – 보안 주체가 Amazon EC2 VPC에 대해 설명할 수 있도록 허용합니다. 이는 콘솔을 사용하는 보안 주체가 보안 암호에 저장하는 보안 인증 정보의 데이터베이스와 동일한 VPC에서 교체 함수를 생성할 수 있도록 하기 위해 필요합니다.
- `kms` - 보안 주체가 암호화 작업에 AWS KMS 키를 사용할 수 있도록 허용합니다. 이는 Secrets Manager가 보안 암호를 암호화하고 해독할 수 있도록 하기 위해 필요합니다. 자세한 내용은 the section called “보안 암호 암호화 및 복호화” 단원을 참조하십시오.
- `lambda` – 보안 주체가 Lambda 교체 함수를 나열할 수 있도록 허용합니다. 이는 콘솔을 사용하는 보안 주체가 기존 교체 함수를 선택할 수 있도록 하기 위해 필요합니다.
- `rds` – 보안 주체가 Amazon RDS의 클러스터 및 인스턴스를 설명할 수 있도록 허용합니다. 이는 콘솔을 사용하는 보안 주체가 Amazon RDS 클러스터 또는 인스턴스를 선택할 수 있도록 하기 위해 필요합니다.
- `redshift` – 보안 주체가 Amazon Redshift의 클러스터를 설명할 수 있도록 허용합니다. 이는 콘솔을 사용하는 보안 주체가 Amazon Redshift 클러스터를 선택할 수 있도록 하기 위해 필요합니다.
- `redshift-serverless` - 보안 주체가 Amazon Redshift Serverless의 네임스페이스를 설명할 수 있도록 허용합니다. 이는 콘솔을 사용하는 보안 주체가 Amazon Redshift Serverless 네임스페이스를 선택할 수 있도록 하기 위해 필요합니다.
- `docdb-elastic` – 보안 주체가 Amazon DocumentDB의 탄력적 클러스터를 설명할 수 있도록 허용합니다. 이는 콘솔을 사용하는 보안 주체가 Amazon DocumentDB 탄력적 클러스터를 선택할 수 있도록 하기 위해 필요합니다.
- `tag` – 보안 주체가 계정 내에서 태그가 지정된 모든 리소스를 가져올 수 있도록 허용합니다.
- `serverlessrepo` - 보안 주체가 CloudFormation 변경 세트를 생성할 수 있도록 허용합니다. 이는 콘솔을 사용하는 보안 주체가 Lambda 교체 함수를 생성할 수 있도록 하기 위해 필요합니다. 자세한 내용은 the section called “Secrets Manager의 사용 방식 CloudFormation” 단원을 참조하십시오.
- `s3` - 보안 주체가에서 관리하는 Amazon S3 버킷에서 객체를 가져오도록 허용합니다 AWS. 이 버킷에는 Lambda 교체 함수 템플릿이 포함되어 있습니다. 이 권한은 콘솔을 사용하는 보안 주체가 버킷의 템플릿을 기반으로 Lambda 교체 함수를 생성할 수 있도록 하기 위해 필요합니다. 자세한 내용은 the section called “Secrets Manager의 사용 방식 CloudFormation” 단원을 참조하십시오.

정책을 보려면 SecretsManagerReadWrite JSON 정책 문서를 참조하세요.

AWS 관리형 정책: AWSSecretsManagerClientReadOnlyAccess

이 정책은 클라이언트 애플리케이션의 AWS Secrets Manager 보안 암호에 대한 읽기 전용 액세스를 제공합니다. 이를 통해 보안 주체는 보안 암호 값을 검색하고 보안 암호 메타데이터를 설명하고 고객 관리형 키로 암호화된 보안 암호를 해독하는 데 필요한 AWS KMS 권한을 부여할 수 있습니다.

권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

- `secretsmanager` - 보안 주체가 보안 암호 값을 검색하고 보안 암호 메타데이터를 설명할 수 있도록 허용합니다.
- `kms` - 보안 주체가 AWS KMS 키를 사용하여 보안 암호를 복호화할 수 있도록 허용합니다. 이 권한은 서비스별 조건을 통해 Secrets Manager에서 사용하는 키로 범위가 지정됩니다.

최신 버전의 JSON 정책 문서를 포함하여 정책에 대한 자세한 내용은 AWS 관리형 정책 참조 안내서의 AWSSecretsManagerClientReadOnlyAccess를 참조하세요.

AWS 관리형 정책에 대한 Secrets Manager 업데이트

Secrets Manager의 AWS 관리형 정책 업데이트에 대한 세부 정보를 봅니다.

변경	설명	Date	버전
AWSSecretsManagerClientReadOnlyAccess – 새로운 관리형 정책	Secrets Manager는 클라이언트 애플리케이션의 보안 암호에 대한 읽기 전용 액세스를 제공하는 새로운 관리형 정책을 생성했습니다. 이 정책은 보안 암호 값을 검색하고 보안 암호 메타데이터를 설명하는 데 필요한 AWS KMS 권한을 허용합니다.	2025년 11월 5일	v1

변경	설명	Date	버전
SecretsManagerReadWrite – 기존 정책 업데이트	이 정책은 콘솔 사용자가 Amazon Redshift 보안 암호를 생성할 때 Amazon Redshift Serverless를 선택할 수 있도록 Amazon Redshift Serverless에 대한 설명 액세스를 허용하도록 업데이트되었습니다.	2024년 3월 12일	v5
SecretsManagerReadWrite – 기존 정책 업데이트	이 정책은 콘솔 사용자가 Amazon DocumentDB 보안 암호를 생성할 때 탄력적 클러스터를 선택할 수 있도록 Amazon DocumentDB 탄력적 클러스터에 대한 설명 액세스를 허용하도록 업데이트되었습니다.	2023년 9월 12일	v4

변경	설명	Date	버전
SecretsManagerReadWrite – 기존 정책 업데이트	이 정책은 콘솔 사용자가 Amazon Redshift 보안 암호를 생성할 때 Amazon Redshift 클러스터를 선택할 수 있도록 Amazon Redshift에 대한 설명 액세스를 허용하도록 업데이트되었습니다. 또한 이 업데이트에는 Lambda 교체 함수 템플릿을 AWS 저장하는에서 관리하는 Amazon S3 버킷에 대한 읽기 액세스를 허용하는 새 권한이 추가되었습니다.	2020년 6월 24일	v3
SecretsManagerReadWrite – 기존 정책 업데이트	이 정책은 콘솔 사용자가 Amazon RDS 암호를 생성할 때 클러스터를 선택할 수 있도록 Amazon RDS 클러스터에 대한 설명 액세스를 허용하도록 업데이트되었습니다.	2018년 5월 3일	v2
SecretsManagerReadWrite – 새 정책	Secrets Manager는 Secrets Manager에 대한 모든 읽기/쓰기 액세스 권한과 함께 콘솔을 사용하는 데 필요한 권한을 부여하는 정책을 만들었습니다.	2018년 4월 4일	v1

AWS Secrets Manager 보안 암호에 대한 권한이 있는 사용자 확인

기본적으로 IAM 자격 증명에는 보안 암호에 대한 액세스 권한이 없습니다. 보안 암호에 대한 액세스를 승인할 때 Secrets Manager는 보안 암호에 연결된 리소스 기반 정책과, 요청을 제출하는 IAM 사용자나 역할에 연결된 모든 자격 증명 기반 정책을 평가합니다. 이를 수행하기 위해 Secrets Manager는 IAM 사용 설명서의 요청 허용 또는 거부 여부 결정에 설명된 프로세스와 유사한 프로세스를 사용합니다.

요청에 적용되는 정책이 여러 개인 경우 Secrets Manager는 계층 구조를 사용하여 권한을 제어합니다.

1. 명시적인 deny가 있는 정책의 설명이 요청 작업 및 리소스와 일치하는 경우:

   명시적인 deny는 다른 모든 것을 무시하고 작업을 차단합니다.
2. 명시적인 deny가 없지만 명시적인 allow가 있는 설명이 요청 작업 및 리소스와 일치하는 경우:

   명시적인 allow는 설명의 리소스에 대한 액세스 권한을 요청의 작업에 부여합니다.

   자격 증명과 보안 암호가 서로 다른 두 계정에 있는 경우 보안 암호에 대한 리소스 정책과 자격 증명allow에 연결된 정책 모두에가 있어야 합니다. 그렇지 않으면 AWS 요청을 거부합니다. 자세한 내용은 크로스 계정 액세스 단원을 참조하십시오.
3. 요청 작업 및 리소스와 일치하는 명시적인 allow가 있는 설명이 없는 경우:

   AWS 는 기본적으로 요청을 거부하며, 이를 암시적 거부라고 합니다.

보안 암호에 대한 리소스 기반 정책을 보려면

- 다음 중 하나를 수행하세요.
  - https://console.aws.amazon.com/secretsmanager/에서 Secrets Manager 콘솔을 엽니다. 보안 암호에 대한 보안 암호 세부 정보 페이지의 리소스 권한(Resource permissions) 섹션에서 권한 편집(Edit permissions)을 선택합니다.
  - 를 사용하여 AWS CLI 를 호출get-resource-policy하거나 AWS SDK를 사용하여를 호출합니다GetResourcePolicy.

자격 증명 기반 정책을 통해 액세스할 수 있는 사용자를 확인하려면 다음을 수행합니다.

- IAM 정책 시뮬레이터를 사용합니다. IAM 정책 시뮬레이터로 IAM 정책 테스트를 참조하세요.

다른 계정에서 AWS Secrets Manager 보안 암호 액세스

하나의 계정에 속한 사용자가 다른 계정의 보안 암호에 액세스하도록 허용하려면(교차 계정 액세스) 리소스 정책 및 ID 정책 모두에서 액세스를 허용해야 합니다. 이는 보안 암호와 동일한 계정의 자격 증명에 액세스를 부여하는 것과 다릅니다.

교차 계정 권한은 다음 작업에만 적용됩니다.

- CancelRotateSecret
- DeleteResourcePolicy
- DeleteSecret
- DescribeSecret
- GetRandomPassword
- GetResourcePolicy
- GetSecretValue
- ListSecretVersionIds
- PutResourcePolicy
- PutSecretValue
- RemoveRegionsFromReplication
- ReplicateSecretToRegions
- RestoreSecret
- RotateSecret
- StopReplicationToReplica
- TagResource
- UntagResource
- UpdateSecret
- UpdateSecretVersionStage
- ValidateResourcePolicy

PutResourcePolicy 작업에서 `BlockPublicPolicy` 파라미터를 사용하면, 보안 암호에 직접 연결된 리소스 정책을 통해 퍼블릭 액세스 권한이 부여되는 것을 방지하여 리소스를 보호할 수 있습니다. IAM Access Analyzer를 사용하여 교차 계정 액세스를 확인할 수도 있습니다.

또한 보안 암호를 암호화하는 KMS 키를 자격 증명에서 사용하도록 허용해야 합니다. 이는 교차 계정 액세스에 AWS 관리형 키 (`aws/secretsmanager`)를 사용할 수 없기 때문입니다. 대신 생성한 KMS 키로 보안 암호를 암호화한 후 키 정책을 연결해야 합니다. KMS 키를 생성하는 데에는 요금이 부과됩니다. 보안 암호에 대한 암호화 키를 변경하려면 the section called “보안 암호 수정” 섹션을 참조하세요.

> **Important**
>
> 보안 암호에 대한 `secretsmanager:PutResourcePolicy` 권한을 부여하는 리소스 기반 정책은 다른 계정에 있는 주체에게도 리소스 기반 정책을 수정할 수 있는 권한을 제공합니다. 이 권한을 가진 주체는 기존 권한을 확장하여 보안 암호에 대한 전체 관리 액세스 권한을 얻을 수도 있습니다. 따라서 정책에는 최소 권한 액세스 원칙을 적용하는 것이 권장됩니다. 자세한 내용은 리소스 기반 정책 단원을 참조하십시오.

다음 예제 정책에서는 계정1에 보안 암호 및 암호화 키가 있고, 보안 암호 값에 액세스할 수 있도록 허용할 자격 증명이 계정2에 있다고 가정합니다.

1단계: Account1의 보안 암호에 리소스 정책 연결

- 다음 정책은 *##2*의 *ApplicationRole*이 *##1*의 보안 암호에 액세스하도록 허용합니다. 이 정책을 사용하려면 the section called “리소스 기반 정책” 섹션을 참조하세요.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/ApplicationRole"
            },
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "*"
        }
    ]
}
```

2단계: Account1의 KMS 키에 대한 키 정책에 명령문 추가

- 다음 키 정책 문은 *##2*의 *ApplicationRole*이 *##1*의 KMS 키를 사용하여 *##1*의 보안 암호를 복호화하도록 허용합니다. 이 명령문을 사용하려면 KMS 키의 키 정책에 추가합니다. 자세한 내용은 키 정책 변경을 참조하세요.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::Account2:role/ApplicationRole"
  },
  "Action": [
    "kms:Decrypt",
    "kms:DescribeKey"
  ],
  "Resource": "*"
}
```

3단계: Account2의 자격 증명에 자격 증명 정책 연결

- 다음 정책은 *##2*의 *ApplicationRole*이 *##1*의 보안 암호에 액세스하고 *##1*의 암호화 키도 사용하여 보안 암호 값을 복호화하도록 허용합니다. 이 정책을 사용하려면 the section called “ID 기반 정책” 섹션을 참조하세요. Secrets Manager 콘솔의 보안 암호 세부 정보 페이지에 있는 보안 암호 ARN 아래에서 보안 암호에 대한 ARN을 찾을 수 있습니다. 또는 `describe-secret`을 호출할 수 있습니다.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "arn:aws:secretsmanager:us-
east-1:123456789012:secret:secretName-AbCdEf"
        },
        {
            "Effect": "Allow",
            "Action": "kms:Decrypt",
```

```
            "Resource": "arn:aws:kms:us-
east-1:123456789012:key/EncryptionKey"
        }
    ]
}
```

온프레미스 환경에서 보안 암호에 액세스

AWS Identity and Access Management Roles Anywhere를 사용하여 외부에서 실행되는 서버, 컨테이너 및 애플리케이션과 같은 워크로드에 대해 IAM에서 임시 보안 자격 증명을 얻을 수 있습니다 AWS. 워크로드는 AWS 애플리케이션에 사용하는 것과 동일한 IAM 정책과 IAM 역할을 사용하여 AWS 리소스에 액세스할 수 있습니다. IAM Roles Anywhere를 사용하면 Secrets Manager를 사용하여 애플리케이션 서버와 같은 온프레미스 장치뿐만 아니라 AWS 의 리소스에서 액세스할 수 있는 보안 인증을 저장하고 관리할 수 있습니다. 자세한 내용은 IAM Roles Anywhere 사용 설명서를 참조하세요.

의 데이터 보호 AWS Secrets Manager

AWS 공동 책임 모델의 데이터 보호에 적용됩니다 AWS Secrets Manager. 이 모델에 설명된 대로 AWS 는 모든를 실행하는 글로벌 인프라를 보호할 책임이 있습니다 AWS 클라우드. 사용자는 인프라에서 호스팅되는 콘텐츠를 관리해야 합니다. 이 콘텐츠에는 사용하는 AWS 서비스 서비스의 보안 구성과 관리 작업이 포함되어 있습니다. 데이터 프라이버시에 대한 자세한 내용은 데이터 프라이버시 FAQ를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 AWS 공동 책임 모델 및 GDPR 블로그 게시물을 참조하세요.

데이터 보호를 위해 자격 AWS 계정 증명을 보호하고 AWS Identity and Access Management (IAM)을 사용하여 개별 사용자 계정을 설정하는 것이 좋습니다. 이러한 방식에서는 각 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 다중 인증(MFA)을 사용합니다.
- SSL/TLS를 사용하여 AWS 리소스와 통신합니다. Secrets Manager는 모든 리전에서 TLS 1.2 및 1.3을 지원합니다. Secrets Manager는 TLS (PQTLS) 네트워크 암호화 프로토콜에 대한 하이브리드 포스트 양자 키 교환 옵션도 지원합니다.
- IAM 보안 주체와 연결되어 있는 액세스 키 ID 및 시크릿 액세스 키를 사용하여 Secrets Manager에 대한 프로그래밍 방식 요청에 서명합니다. 또는 AWS Security Token Service(AWS STS)를 사용하여 임시 보안 자격 증명을 생성하여 요청에 서명할 수 있습니다.

- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다 AWS CloudTrail. the section called “를 사용하여 로그 AWS CloudTrail ”을(를) 참조하세요.
- 명령줄 인터페이스 또는 API를 AWS 통해에 액세스할 때 FIPS 140-2 검증 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용합니다. the section called “Secrets Manager 엔드포인트”을(를) 참조하세요.
- 를 사용하여 Secrets Manager AWS CLI 에 액세스하는 경우 . the section called “를 사용하여 AWS Secrets Manager 보안 암호를 AWS CLI 저장할 때의 위험 완화”

저장 시 암호화

Secrets Manager는 AWS Key Management Service (AWS KMS)를 통한 암호화를 사용하여 저장 데이터의 기밀성을 보호합니다.는 많은 서비스에서 사용하는 키 스토리지 및 암호화 AWS 서비스를 AWS KMS 제공합니다. Secrets Manager의 모든 보안 암호는 고유한 데이터 키로 암호화됩니다. 각 데이터 키는 KMS 키에 의해 보호됩니다. 계정에 대한 Secrets Manager AWS 관리형 키 로 기본 암호화를 사용하거나, AWS KMS에서 고유한 고객 관리형 키를 생성할 수도 있습니다. 고객 관리형 키를 사용하면 KMS 키 활동에 대한 권한 부여를 보다 세부적으로 제어할 수 있습니다. 자세한 내용은 the section called “보안 암호 암호화 및 복호화” 단원을 참조하십시오.

전송 중 데이터 암호화

Secrets Manager는 전송 중인 데이터를 암호화하기 위해 안전한 프라이빗 엔드포인트를 제공합니다. 보안 및 프라이빗 엔드포인트를 사용하면 AWS 가 Secrets Manager에 대한 API 요청의 무결성을 보호할 수 있습니다. AWS 에서는 호출자가 X.509 인증서 및/또는 Secrets Manager 보안 액세스 키를 사용하여 API 호출에 서명해야 합니다. 이 요구 사항은 Signature 버전 4 서명 프로세스(Sigv4)에 명시되어 있습니다.

AWS Command Line Interface (AWS CLI) 또는 AWS SDKs를 사용하여를 호출하는 경우 사용할 액세스 키를 AWS구성합니다. 그런 다음 이러한 도구는 자동으로 액세스 키를 사용하여 요청에 서명합니다. the section called “를 사용하여 AWS Secrets Manager 보안 암호를 AWS CLI 저장할 때의 위험 완화”을(를) 참조하세요.

인터네트워크 트래픽 개인 정보 보호

AWS 는 알려진 및 프라이빗 네트워크 경로를 통해 트래픽을 라우팅할 때 개인 정보 보호를 유지하기 위한 옵션을 제공합니다.

서비스와 온프레미스 클라이언트 및 애플리케이션 간의 트래픽

프라이빗 네트워크와 간에 AWS Secrets Manager는 두 가지 연결 옵션이 있습니다.

- An AWS Site-to-Site VPN 연결. 자세한 내용은 What is AWS Site-to-Site VPN?을 참조하세요.
- AWS Direct Connect 연결. 자세한 내용은 AWS Direct Connect란 무엇입니까?를 참조하세요.

동일한 리전의 AWS 리소스 간 트래픽

에서 Secrets Manager와 API 클라이언트 간의 트래픽을 보호하려면 Secrets Manager API 엔드포인트에 비공개로 액세스하도록 AWS PrivateLink를 AWS설정합니다.

암호화 키 관리

Secrets Manager가 보호된 보안 암호 데이터의 새 버전을 암호화해야 하는 경우 Secrets Manager는 KMS 키에서 새 데이터 키를 생성 AWS KMS 하도록에 요청을 보냅니다. Secrets Manager는 봉투 암호화(envelope encryption)에 이 데이터 키를 사용합니다. Secrets Manager는 암호화된 데이터 키를 암호화된 보안 암호와 함께 저장합니다. 보안 암호를 해독해야 하는 경우 Secrets Manager는 데이터 키를 해독 AWS KMS 하도록 요청합니다. 그런 다음 Secrets Manager는 복호화된 데이터 키를 사용하여 암호화된 보안 암호를 복호화합니다. Secrets Manager는 데이터 키를 암호화되지 않은 형식으로 저장하지 않으며 가능한 한 빨리 메모리에서 키를 제거합니다. 자세한 내용은 the section called “보안 암호 암호화 및 복호화” 단원을 참조하십시오.

의 보안 암호 암호화 및 복호화 AWS Secrets Manager

Secrets Manager는 AWS KMS 키 및 데이터 키로 봉투 암호화를 사용하여 각 보안 암호 값을 보호합니다. 보안 암호의 보안 암호 값이 변경될 때마다 Secrets Manager는 AWS KMS 에서 새 데이터 키를 요청하여 보안 암호 값을 보호합니다. KMS 키 아래에 암호화된 데이터 키는 보안 암호의 메타데이터에 저장합니다. 보안 암호를 해독하기 위해 Secrets Manager는 먼저 KMS 키를 사용하여 암호화된 데이터 키를 해독합니다 AWS KMS.

Secrets Manager에서는 KMS를 사용하여 보안 암호 값을 직접 암호화하지 않습니다. 대신 KMS 키를 사용하여 256비트 고급 암호화 표준(AES) 대칭 데이터 키를 생성하고 암호화하며 데이터 키를 사용하여 보안 암호 값을 암호화합니다. Secrets Manager는 일반 텍스트 데이터 키를 사용하여 외부의 보안 암호 값을 암호화 AWS KMS한 다음 메모리에서 제거합니다. 데이터 키의 암호화된 사본을 암호의 메타데이터에 저장합니다.

주제

- AWS KMS 키 선택
- 무엇을 암호화하나요?
- 암호화 및 복호화 프로세스
- KMS 키에 대한 권한
- Secrets Manager에서 KMS 키를 사용하는 방법
- AWS 관리형 키 (aws/secretsmanager)의 키 정책
- Secrets Manager 보안 암호 컨텍스트
- 와의 Secrets Manager 상호 작용 모니터링 AWS KMS

AWS KMS 키 선택

보안 암호를 생성할 때 AWS 계정 및 리전에서 대칭 암호화 고객 관리형 키를 선택하거나 Secrets Manager() AWS 관리형 키 용를 사용할 수 있습니다`aws/secretsmanager`. 를 AWS 관리형 키 선택하고 아직 존재하지 `aws/secretsmanager` 않는 경우 Secrets Manager는 이를 생성하고 보안 암호와 연결합니다. 계정의 각 보안 암호에 대해 동일한 KMS 또는 다른 KMS 키를 사용할 수 있습니다. 암호 그룹의 키에 대한 사용자 지정 권한을 설정하거나 해당 키에 대한 특정 작업을 감사하려는 경우 다른 KMS 키를 사용할 수 있습니다. Secrets Manager는 대칭 암호화 KMS 키만 지원합니다. 외부 키 스토어에서 KMS 키를 사용하는 경우 요청이 AWS외부로 이동해야 하므로 KMS 키에 대한 암호화 작업이 더 오래 걸리고 신뢰성과 내구성이 떨어질 수 있습니다.

보안 암호의 암호화 키를 변경하는 방법에 대한 자세한 내용은 the section called “보안 암호에 대한 암호화 키 변경”을(를) 참조하세요.

암호화 키를 변경하면 Secrets Manager가 새 키로 `AWSCURRENT`, `AWSPENDING`, `AWSPREVIOUS` 버전을 다시 암호화합니다. 보안 암호가 잠기는 것을 방지하기 위해 Secrets Manager는 모든 기존 버전을 이전 키로 암호화합니다. 즉, 이전 키 또는 새 키를 사용하여 `AWSCURRENT`, `AWSPENDING`, `AWSPREVIOUS` 버전을 복호화할 수 있습니다. 이전 키에 대한 `kms:Decrypt` 권한이 없는 경우, 암호화 키를 변경하면 Secrets Manager는 보안 암호 버전을 복호화하여 이를 다시 암호화할 수 없습니다. 이 경우 기존 버전은 다시 암호화되지 않습니다.

`AWSCURRENT`가 새 암호화 키로만 복호화할 수 있도록 하려면 새 키로 새 버전의 보안 암호를 생성합니다. 그런 다음, `AWSCURRENT` 보안 암호 버전을 복호화하려면 새 키에 대한 권한이 있어야 합니다.

에 AWS 관리형 키 `aws/secretsmanager` 대한 권한을 거부하고 고객 관리형 키로 보안 암호를 암호화하도록 요구할 수 있습니다. 자세한 내용은 the section called “예: 보안 암호를 암호화하기 위한 특정 AWS KMS 키 거부” 단원을 참조하십시오.

보안 암호와 연결된 KMS 키를 찾으려면 콘솔에서 보안 암호를 확인하거나 ListSecrets 또는 DescribeSecret에서 호출합니다. 보안 암호가 AWS 관리형 키 Secrets Manager(`aws/secretsmanager`)용와 연결된 경우 이러한 작업은 KMS 키 식별자를 반환하지 않습니다.

무엇을 암호화하나요?

Secrets Manager는 암호 값을 암호화하지만 다음 항목은 암호화하지 않습니다.

- 보안 암호 이름 및 설명
- 교체 설정
- 보안 암호와 연결된 KMS 키의 ARN
- 연결된 모든 AWS 태그

암호화 및 복호화 프로세스

Secrets Manager는 다음 프로세스에 따라 보안 암호의 보안 암호 값을 암호화합니다.

1. Secrets Manager는 보안 암호에 대한 KMS 키의 ID와 256비트 AES 대칭 키에 대한 요청을 사용하여 AWS KMS GenerateDataKey 작업을 호출합니다.는 KMS 키로 암호화된 일반 텍스트 데이터 키와 해당 데이터 키의 사본을 AWS KMS 반환합니다.
2. Secrets Manager는 일반 텍스트 데이터 키와 고급 암호화 표준(AES) 알고리즘을 사용하여 외부에서 보안 암호 값을 암호화합니다 AWS KMS. 사용한 후에는 가능한 빨리 메모리에서 일반 텍스트 키를 제거합니다.
3. Secrets Manager는 보안 암호 값을 복호화할 수 있도록 보안 암호의 메타데이터에 암호화된 데이터 키를 저장합니다. 하지만 Secrets Manager API 중에서 어떤 것도 암호화된 보안 암호나 암호화된 데이터 키를 반환하지 않습니다.

암호화된 암호 값을 해독하려면:

1. Secrets Manager는 AWS KMS Decrypt 작업을 호출하고 암호화된 데이터 키를 전달합니다.
2. AWS KMS 는 보안 암호에 KMS 키를 사용하여 데이터 키를 복호화합니다. 그런 다음 일반 텍스트 데이터 키를 반환합니다.
3. Secrets Manager는 일반 텍스트 데이터 키를 사용하여 보안 암호 값을 복호화합니다. 그런 다음 가능한 빨리 메모리에서 데이터 키를 제거합니다.

KMS 키에 대한 권한

Secrets Manager가 암호화 작업에서 KMS 키를 사용하는 경우 보안 암호 값에 액세스하거나 업데이트하는 사용자를 대신하여 KMS 키가 작동합니다. IAM 정책 또는 키 정책에서 권한을 부여할 수 있습니다. 다음 Secrets Manager 작업에는 AWS KMS 권한이 필요합니다.

- CreateSecret
- GetSecretValue
- PutSecretValue
- UpdateSecret
- ReplicateSecretToRegions

Secrets Manager의 권한 정책에서 생성된 요청에 대해서만 KMS 키를 사용할 수 있도록 하려면 kms:ViaService 조건 키를 `secretsmanager.<Region>.amazonaws.com` 값과 함께 사용합니다.

암호화 작업에 대한 KMS 키 사용 조건으로서 암호화 컨텍스트에서 키나 값을 사용할 수도 있습니다. 예를 들면 IAM 또는 키 정책 문서에서 문자열 조건 연산자를 사용하거나 권한 부여에서 권한 부여 제약을 사용할 수 있습니다. KMS 키 부여 전파에는 최대 5분이 걸릴 수 있습니다. 자세한 내용은 CreateGrant를 참조하세요.

Secrets Manager에서 KMS 키를 사용하는 방법

Secrets Manager는 KMS 키를 사용하여 다음 AWS KMS 작업을 호출합니다.

GenerateDataKey

Secrets Manager는 다음 Secrets Manager 작업에 대한 응답으로 AWS KMS GenerateDataKey 작업을 호출합니다.

- CreateSecret - 새 보안 암호가 보안 암호 값을 포함하는 경우 Secrets Manager는 보안 암호 값을 암호화하기 위해 새 데이터 키를 요청합니다.
- PutSecretValue – Secrets Manager는 지정된 보안 암호 값을 암호화하기 위해 새 데이터 키를 요청합니다.
- ReplicateSecretToRegions - Secrets Manager는 복제된 보안 암호를 암호화하기 위해 복제본 리전의 KMS 키에 대한 데이터 키를 요청합니다.
- UpdateSecret - 보안 암호 값 또는 KMS 키가 변경되는 경우 Secrets Manager는 새 보안 암호 값을 암호화하기 위해 새 데이터 키를 요청합니다.

RotateSecret 작업은 암호 값을 변경하지 않으므로 GenerateDataKey를 호출하지 않습니다. 하지만 RotateSecret에서 호출하는 Lambda 함수가 보안 암호 값을 변경하는 경우 PutSecretValue 작업에 대한 호출로 GenerateDataKey 요청이 트리거됩니다.

Decrypt

Secrets Manager는 다음 Secrets Manager 작업에 대한 응답으로 Decrypt 작업을 호출합니다.

- GetSecretValue 및 BatchGetSecretValue – Secrets Manager는 보안 암호 값을 발신자에게 반환하기 전에 보안 암호 값을 복호화합니다. 암호화된 보안 암호 값을 해독하기 위해 Secrets Manager는 AWS KMS Decrypt 작업을 호출하여 보안 암호의 암호화된 데이터 키를 해독합니다. 그런 다음, 일반 텍스트 데이터 키를 사용하여 암호화된 암호 값을 해독합니다. 배치 명령의 경우 Secrets Manager는 해독된 키를 재사용할 수 있으므로 일부 호출에서만 Decrypt 요청이 발생합니다.
- PutSecretValue 및 UpdateSecret - 대부분의 PutSecretValue 및 UpdateSecret 요청은 Decrypt 작업을 트리거하지 않습니다. 하지만 PutSecretValue 또는 UpdateSecret 요청이 보안 암호의 기존 버전에서 보안 암호 값을 변경하려 할 경우 Secrets Manager는 기존 보안 암호 값을 복호화하고 요청의 보안 암호 값과 비교하여 동일한지 확인합니다. 이 작업을 통해 Secrets Manager 작업의 idempotent가 유지됩니다. 암호화된 보안 암호 값을 해독하기 위해 Secrets Manager는 AWS KMS 암호 해독 작업을 호출하여 보안 암호의 암호화된 데이터 키를 해독합니다. 그런 다음, 일반 텍스트 데이터 키를 사용하여 암호화된 암호 값을 해독합니다.
- ReplicateSecretToRegions – Secrets Manager는 복제본 리전의 KMS 키를 사용하여 다시 암호화하기 전에 먼저 보안 암호 값을 해독합니다.

암호화

Secrets Manager는 다음 Secrets Manager 작업에 대한 응답으로 Encrypt 작업을 호출합니다.

- UpdateSecret - KMS 키를 변경하면 Secrets Manager는 새 키로 AWSCURRENT, AWSPREVIOUS 및 AWSPENDING 보안 암호 버전을 보호하는 데이터 키를 다시 암호화합니다.

DescribeKey

Secrets Manager는 Secrets Manager 콘솔에서 암호를 생성하거나 편집할 때 DescribeKey 작업을 호출하여 KMS 키를 나열할지 여부를 결정합니다.

KMS 키에 대한 액세스 검증

보안 암호와 연결된 KMS 키를 설정하거나 변경할 때 Secrets Manager는 지정된 KMS 키를 사용하여 GenerateDataKey 및 Decrypt 작업을 호출합니다. 이러한 호출은 발신자가 해당 작업에 대해 KMS 키를 사용할 수 있는 권한이 있는지 확인합니다. Secrets Manager는 이러한 작업의 결과를 무시하며, 암호화된 작업에서 사용하지 않습니다.

이러한 요청에서는 SecretVersionId 키 암호화 컨텍스트 값이 RequestToValidateKeyAccess이므로 이와 같은 검증 호출을 식별할 수 있습니다.

> **Note**
>
> 예전에는 Secrets Manager 검증 호출에 암호화 컨텍스트가 포함되지 않았습니다. 이전 AWS CloudTrail 로그에서 암호화 컨텍스트가 없는 호출을 찾을 수 있습니다.

AWS 관리형 키 (aws/secretsmanager)의 키 정책

Secrets Manager(aws/secretsmanager) AWS 관리형 키 용에 대한 키 정책은 Secrets Manager가 사용자를 대신하여 요청할 때만 지정된 작업에 KMS 키를 사용할 수 있는 권한을 사용자에게 부여합니다. 키 정책에서는 사용자가 KMS 키를 직접 사용하도록 허용하지 않습니다.

이 키 정책은 모든 AWS 관리형 키의 정책처럼 서비스에 의해 설정됩니다. 키 정책은 변경할 수 없지만 언제든지 볼 수 있습니다. 자세한 내용은 키 정책 보기를 참조하세요.

키 정책의 정책 설명문은 다음 효과를 갖습니다.

- 계정 내 사용자가 Secrets Manager를 통해 대신 요청할 때만 암호화 작업에 대해 KMS 키를 사용할 수 있도록 합니다. kms:ViaService 조건 키는 이 제한을 강제 적용합니다.
- AWS 계정이 KMS 키 속성을 보고 권한 부여를 취소할 수 있도록 허용하는 IAM 정책을 생성하도록 허용합니다.
- Secrets Manager는 권한 부여를 사용하여 KMS 키에 대한 액세스 권한을 얻지는 않지만 정책은 Secrets Manager로 하여금 사용자를 대신하여 KMS 키에 대한 권한 부여를 생성하도록 하고 계정으로 하여금 Secrets Manager가 KMS 키를 사용하도록 허용하는 모든 권한 부여를 취소하도록 합니다. 다음은에 대한 정책 문서의 표준 요소입니다 AWS 관리형 키.

다음은 AWS 관리형 키 Secrets Manager 예제의 키 정책입니다.

JSON

```
{
  "Id": "auto-secretsmanager-2",
  "Version":"2012-10-17",
  "Statement": [
    {
```

```
    "Sid": "Allow access through AWS Secrets Manager for all principals in the
account that are authorized to use AWS Secrets Manager",
    "Effect": "Allow",
    "Principal": {
      "AWS": [
        "*"
      ]
    },
    "Action": [
      "kms:Encrypt",
      "kms:Decrypt",
      "kms:ReEncrypt*",
      "kms:CreateGrant",
      "kms:DescribeKey"
    ],
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "kms:CallerAccount": "111122223333",
        "kms:ViaService": "secretsmanager.us-west-2.amazonaws.com"
      }
    }
  },
  {
    "Sid": "Allow access through AWS Secrets Manager for all principals in the
account that are authorized to use AWS Secrets Manager",
    "Effect": "Allow",
    "Principal": {
      "AWS": [
        "*"
      ]
    },
    "Action": "kms:GenerateDataKey*",
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "kms:CallerAccount": "111122223333"
      },
      "StringLike": {
        "kms:ViaService": "secretsmanager.us-west-2.amazonaws.com"
      }
    }
  },
  {
```

```
      "Sid": "Allow direct access to key metadata to the account",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::111122223333:root"
        ]
      },
      "Action": [
        "kms:Describe*",
        "kms:Get*",
        "kms:List*",
        "kms:RevokeGrant"
      ],
      "Resource": "*"
    }
  ]
}
```

Secrets Manager 보안 암호 컨텍스트

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.

에 대한 GenerateDataKey 및 Decrypt 요청에서 AWS KMS Secrets Manager는 다음 예제와 같이 보안 암호와 해당 버전을 식별하는 두 개의 이름-값 페어가 있는 암호화 컨텍스트를 사용합니다. 이름은 달라지지 않지만, 결합된 암호화 컨텍스트 값은 각 암호 값마다 다릅니다.

```
"encryptionContext": {
    "SecretARN": "arn:aws:secretsmanager:us-east-2:111122223333:secret:test-secret-
a1b2c3",
    "SecretVersionId": "EXAMPLE1-90ab-cdef-fedc-ba987SECRET1"
}
```

암호화 컨텍스트를 사용하여 AWS CloudTrail 및 Amazon CloudWatch Logs 같은 감사 레코드 및 로그에서 정책 및 권한 부여의 승인 조건으로서 이러한 암호화 작업을 식별할 수 있습니다.

Secrets Manager 암호화 컨텍스트는 이름-값 페어 두 개로 구성됩니다.

- SecretARN - 첫 번째 이름-값 페어는 보안 암호를 식별합니다. 키는 SecretARN입니다. 이 값은 암호의 Amazon 리소스 이름(ARN)입니다.

```
"SecretARN": "ARN of an Secrets Manager secret"
```

예를 들어, 보안 암호 ARN이 `arn:aws:secretsmanager:us-east-2:111122223333:secret:test-secret-a1b2c3`이면 암호화 컨텍스트는 다음 페어를 포함합니다.

```
"SecretARN": "arn:aws:secretsmanager:us-east-2:111122223333:secret:test-secret-a1b2c3"
```

- SecretVersionId - 두 번째 이름-값 페어는 보안 암호의 버전을 식별합니다. 키는 `SecretVersionId`입니다. 이 값은 버전 ID입니다.

```
"SecretVersionId": "<version-id>"
```

예를 들어, 보안 암호의 버전 ID가 `EXAMPLE1-90ab-cdef-fedc-ba987SECRET1`이면 암호화 컨텍스트에는 다음 페어가 포함됩니다.

```
"SecretVersionId": "EXAMPLE1-90ab-cdef-fedc-ba987SECRET1"
```

보안 암호에 대한 KMS 키를 설정하거나 변경하면 Secrets Manager는 호출자가 이러한 작업에 KMS 키를 사용할 권한이 있는지 확인하기 AWS KMS 위해 GenerateDataKey 및 Decrypt 요청을에 보냅니다. 응답을 무시하므로 암호 값에 응답을 사용하지 않습니다.

이러한 검증 요청에서 `SecretARN`의 값은 암호의 실제 ARN이지만, `SecretVersionId` 값은 다음 예시 암호화 컨텍스트에서 나와 있는 것처럼 `RequestToValidateKeyAccess`입니다. 이 특수 값은 로그 및 감사 내역에서 검증 요청을 식별하는 데 도움이 됩니다.

```
"encryptionContext": {
    "SecretARN": "arn:aws:secretsmanager:us-east-2:111122223333:secret:test-secret-a1b2c3",
    "SecretVersionId": "RequestToValidateKeyAccess"
}
```

> **Note**
>
> 예전에는 Secrets Manager 검증 호출에 암호화 컨텍스트가 포함되지 않았습니다. 이전 AWS CloudTrail 로그에서 암호화 컨텍스트가 없는 호출을 찾을 수 있습니다.

와의 Secrets Manager 상호 작용 모니터링 AWS KMS

AWS CloudTrail 및 Amazon CloudWatch Logs를 사용하여 Secrets Manager가 사용자를 대신하여에 보내는 요청을 추적할 수 AWS KMS 있습니다. 보안 암호 사용 모니터링에 대한 자세한 내용은 보안 암호 모니터링 섹션을 참조하세요.

GenerateDataKey

보안 암호에서 보안 암호 값을 생성하거나 변경하면 Secrets Manager는 보안 암호에 대한 KMS 키를 AWS KMS 지정하는 GenerateDataKey 요청을에 보냅니다.

`GenerateDataKey` 작업을 기록하는 이벤트는 다음 예시 이벤트와 유사합니다. 이 요청은 `secretsmanager.amazonaws.com`에 의해 호출됩니다. 파라미터로는 보안 암호용 KMS 키의 Amazon 리소스 이름(ARN), 256비트 키를 요구하는 키 지정자, 그리고 보안 암호와 버전을 식별하는 암호화 컨텍스트가 있습니다.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AROAIGDTESTANDEXAMPLE:user01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/user01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2018-05-31T23:23:41Z"
            }
        },
        "invokedBy": "secretsmanager.amazonaws.com"
    },
    "eventTime": "2018-05-31T23:23:41Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
```

```
    "awsRegion": "us-east-2",
    "sourceIPAddress": "secretsmanager.amazonaws.com",
    "userAgent": "secretsmanager.amazonaws.com",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "keySpec": "AES_256",
        "encryptionContext": {
            "SecretARN": "arn:aws:secretsmanager:us-east-2:111122223333:secret:test-
secret-a1b2c3",
            "SecretVersionId": "EXAMPLE1-90ab-cdef-fedc-ba987SECRET1"
        }
    },
    "responseElements": null,
    "requestID": "a7d4dd6f-6529-11e8-9881-67744a270888",
    "eventID": "af7476b6-62d7-42c2-bc02-5ce86c21ed36",
    "readOnly": true,
    "resources": [
        {
            "ARN": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "accountId": "111122223333",
            "type": "AWS::KMS::Key"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

Decrypt

보안 암호의 보안 암호 값을 가져오거나 변경하면 Secrets Manager는 AWS KMS 에 복호화 요청을 보내 암호화된 데이터 키를 복호화합니다. 배치 명령의 경우 Secrets Manager는 해독된 키를 재사용할 수 있으므로 일부 호출에서만 `Decrypt` 요청이 발생합니다.

`Decrypt` 작업을 기록하는 이벤트는 다음 예시 이벤트와 유사합니다. 사용자는 테이블에 액세스하는 AWS 계정의 보안 주체입니다. 파라미터에는 암호화된 테이블 키(암호 텍스트 BLOB)와 테이블과 AWS 계정을 식별하는 암호화 컨텍스트가 포함됩니다.는 사이퍼텍스트에서 KMS 키의 ID를 AWS KMS 도출합니다.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
```

```
        "type": "IAMUser",
        "principalId": "AROAIGDTESTANDEXAMPLE:user01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/user01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2018-05-31T23:36:09Z"
            }
        },
        "invokedBy": "secretsmanager.amazonaws.com"
    },
    "eventTime": "2018-05-31T23:36:09Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "secretsmanager.amazonaws.com",
    "userAgent": "secretsmanager.amazonaws.com",
    "requestParameters": {
        "encryptionContext": {
            "SecretARN": "arn:aws:secretsmanager:us-east-2:111122223333:secret:test-
secret-a1b2c3",
            "SecretVersionId": "EXAMPLE1-90ab-cdef-fedc-ba987SECRET1"
        }
    },
    "responseElements": null,
    "requestID": "658c6a08-652b-11e8-a6d4-ffee2046048a",
    "eventID": "f333ec5c-7fc1-46b1-b985-cbda13719611",
    "readOnly": true,
    "resources": [
        {
            "ARN": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "accountId": "111122223333",
            "type": "AWS::KMS::Key"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

암호화

보안 암호와 연결된 KMS 키를 변경하면 Secrets Manager는에 암호화 요청을 보내 새 키로 AWSCURRENT, AWSPREVIOUS및 AWSPENDING 보안 암호 버전을 AWS KMS 다시 암호화합니다. 보안 암호를 다른 리전에 복제하는 경우에도 Secrets Manager가 AWS KMS로 암호화 요청을 보냅니다.

Encrypt 작업을 기록하는 이벤트는 다음 예시 이벤트와 유사합니다. 사용자는 테이블에 액세스하는 AWS 계정의 보안 주체입니다.


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AROAIGDTESTANDEXAMPLE:user01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/user01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "attributes": {
                "creationDate": "2023-06-09T18:11:34Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "secretsmanager.amazonaws.com"
    },
    "eventTime": "2023-06-09T18:11:34Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Encrypt",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "secretsmanager.amazonaws.com",
    "userAgent": "secretsmanager.amazonaws.com",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-east-2:111122223333:key/EXAMPLE1-f1c8-4dce-8777-
aa071ddefdcc",
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "encryptionContext": {
            "SecretARN": "arn:aws:secretsmanager:us-
east-2:111122223333:secret:ChangeKeyTest-5yKnKS",
            "SecretVersionId": "EXAMPLE1-5c55-4d7c-9277-1b79a5e8bc50"
        }
    },
    "responseElements": null,
```

```
    "requestID": "129bd54c-1975-4c00-9b03-f79f90e61d60",
    "eventID": "f7d9ff39-15ab-47d8-b94c-56586de4ab68",
    "readOnly": true,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:111122223333:key/EXAMPLE1-f1c8-4dce-8777-
aa071ddefdcc"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```


의 인프라 보안 AWS Secrets Manager

관리형 서비스인 AWS Secrets Manager 는 AWS 글로벌 네트워크 보안으로 보호됩니다. AWS 보안 서비스 및가 인프라를 AWS 보호하는 방법에 대한 자세한 내용은 AWS 클라우드 보안을 참조하세요. 인프라 보안 모범 사례를 사용하여 AWS 환경을 설계하려면 보안 원칙 AWS Well-Architected Framework의 인프라 보호를 참조하세요.

AWS TLS를 사용하여 게시된 API를 통한 네트워크를 이용하여 Secrets Manager에 액세스합니다. 네트워크 위치에서 Secrets Manage API를 호출할 수 있습니다. 하지만 Secrets Manager는 원본 IP 주소 기반의 제한을 포함할 수 있는 리소스 기반 액세스 정책을 지원합니다. Secrets Manager 리소스 정책을 사용하여 대한 액세스를 제어할 수도 있습니다.특정 Virtual Private Cloud(VPC) 엔드포인트 또는 특정 VPC의 암호에 대한 액세스를 제어할 수도 있습니다. 이렇게 하면 네트워크 내의 특정 VPC에서만 지정된 보안 암호에 대한 AWS 네트워크 액세스가 효과적으로 격리됩니다. 자세한 내용은 the section called “VPC 엔드포인트(AWS PrivateLink)” 단원을 참조하십시오.

AWS Secrets Manager VPC 엔드포인트 사용

가능한 경우 대부분의 인프라를 퍼블릭 인터넷에서 액세스할 수 없는 프라이빗 네트워크에서 실행하는 것이 좋습니다. 인터페이스 VPC 엔드포인트를 생성하여 VPC와 Secrets Manager 간에 프라이빗 연결을 설정할 수 있습니다. 인터페이스 엔드포인트는 인터넷 게이트웨이AWS PrivateLink, NAT 디바이스, VPN 연결 또는 Direct Connect 연결 없이 비공개로 Secrets Manager APIs에 액세스할 수 있는 기술로 구동됩니다. VPC의 인스턴스는 Secrets Manager API와 통신하는 데 퍼블릭 IP 주소를 필요로

하지 않습니다. VPC와 Secrets Manager 간의 트래픽은 AWS 네트워크를 벗어나지 않습니다. 자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 VPC 엔드포인트(AWS PrivateLink)를 참조하세요.

Secret Manager가 Lambda 교체 함수를 사용하여 암호를 교체할 때(예: 데이터베이스 자격 증명이 포함된 암호) Lambda 함수는 데이터베이스와 Secret Manager 모두에게 요청을 합니다. 콘솔을 사용하여 자동 교체를 설정하면 Secrets Manager가 데이터베이스와 동일한 VPC에 Lambda 함수를 생성합니다. Lambda 교체 함수에서 Secrets Manager로의 요청이 Amazon 네트워크를 벗어나지 않도록 동일한 VPC의 Secrets Manager 엔드포인트를 생성하는 것이 좋습니다.

엔드포인트에 프라이빗 DNS를 사용하도록 설정하는 경우, 리전에 대한 기본 DNS 이름(예: `secretsmanager.us-east-1.amazonaws.com`)을 사용하여 Secrets Manager에 API 요청을 할 수 있습니다. 자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 엔드포인트를 통해 서비스 액세스를 참조하세요.

권한 정책에 조건을 포함시켜 Secrets Manager에 대한 요청이 VPC 액세스에서 나오도록 할 수 있습니다. 자세한 내용은 the section called “예: 권한 및 VPC” 단원을 참조하십시오.

AWS CloudTrail 로그를 사용하여 VPC 엔드포인트를 통해 보안 암호 사용을 감사할 수 있습니다.

Secrets Manager에 대한 VPC 엔드포인트를 생성하려면

1. 자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 엔드포인트 생성을 참조하세요. 다음 서비스 이름 중 하나를 사용합니다.
   - `com.amazonaws.region.secretsmanager`
   - `com.amazonaws.region.secretsmanager-fips`
2. 엔드포인트에 대한 액세스를 제어하려면 Control access to VPC endpoints using endpoint policies 섹션을 참조하세요.
3. IPv6 및 듀얼 스택 주소 지정을 사용하려면 IPv4 및 IPv6 액세스 섹션을 참조하세요.

엔드포인트의 엔드포인트 정책 생성

엔드포인트 정책은 인터페이스 엔드포인트에 연결할 수 있는 IAM 리소스입니다. 기본 엔드포인트 정책은 인터페이스 엔드포인트를 통해 Secrets Manager에 대한 전체 액세스 권한을 허용합니다. VPC에서 Secrets Manager로 허용되는 액세스를 제어하려면 인터페이스 엔드포인트에 사용자 지정 엔드포인트 정책을 연결합니다.

엔드포인트 정책은 다음 정보를 지정합니다.

- 작업을 수행할 수 있는 위탁자(AWS 계정, IAM 사용자, IAM 역할)
- 수행할 수 있는 작업
- 작업을 수행할 수 있는 리소스.

자세한 정보는 AWS PrivateLink 가이드의 엔드포인트 정책을 사용하여 서비스에 대한 액세스 제어를 참조하세요.

예제: Secrets Manager 작업에 대한 VPC 엔드포인트 정책

다음은 사용자 지정 엔드포인트 정책의 예입니다. 이 정책을 인터페이스 엔드포인트에 연결하면, 지정된 보안 암호에 대해 나열된 Secrets Manager 작업에 대한 액세스 권한이 부여됩니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "Allow all users to use GetSecretValue and DescribeSecret on the
 specified secret.",
      "Effect": "Allow",
      "Principal": "*",
      "Action": [
        "secretsmanager:GetSecretValue",
        "secretsmanager:DescribeSecret"
      ],
      "Resource": "arn:aws:secretsmanager:us-
east-1:111122223333:secret:secretName-AbCdEf"
    }
  ]
}
```

공유 서브넷

공유하는 서브넷의 VPC 엔드포인트는 생성, 설명, 수정 또는 삭제할 수 없습니다. 그러나 공유하는 서브넷의 VPC 엔드포인트를 사용할 수는 있습니다. VPC 공유에 대한 자세한 내용은 Amazon Virtual Private Cloud 사용 설명서에서 다른 계정과 VPC 공유를 참조하세요.

IAM 정책을 사용하여 API 액세스 제어

IAM 정책을 사용하여 IP 주소를 AWS 서비스 기반으로에 대한 액세스를 제어하는 경우 IPv6 주소 범위를 포함하도록 정책을 업데이트해야 할 수 있습니다. 이 가이드는 IPv4와 IPv6의 차이점을 설명하고, 두 프로토콜을 모두 지원하도록 IAM 정책을 업데이트하는 방법을 안내합니다. 이러한 변경을 구현하면 IPv6를 지원하면서도 AWS 리소스에 대한 보안 액세스를 유지할 수 있습니다.

IPv6란?

IPv6는 결국에는 IPv4를 대체하기 위해 개발된 차세대 IP 표준입니다. 이전 버전인 IPv4는 32비트 주소 지정 체계를 사용하여 43억 개의 디바이스를 지원합니다. IPv6는 128비트 주소 지정 체계를 사용하여 약 340조(또는 2의 128승) 개의 디바이스를 지원합니다.

자세한 내용은 VPC IPv6 웹 페이지를 참조하세요.

다음은 IPv6 주소의 예시입니다.

```
2001:cdba:0000:0000:0000:0000:3257:9652 # This is a full, unabbreviated IPv6 address.
2001:cdba:0:0:0:3257:9652               # The same address with leading zeros in each
 group omitted
2001:cdba::3257:965                     # A compressed version of the same address.
```

IAM 듀얼 스택(IPv4 및 IPv6) 정책

IAM 정책을 사용하여 Secrets Manager API에 대한 액세스를 제어하고, 구성된 범위를 벗어난 IP 주소가 Secrets Manager API에 액세스하지 못하도록 할 수 있습니다.

Secrets Manager API의 secretsmanager.{region}.amazonaws.com 듀얼 스택 엔드포인트는 IPv6와 IPv4 모두를 지원합니다.

IPv4와 IPv6를 모두 지원해야 하는 경우, IPv6 주소를 처리하도록 IP 주소 필터링 정책을 업데이트해야 합니다. 그러지 않으면 IPv6를 통해 Secrets Manager에 연결할 수 없습니다.

누가 이 변경을 해야 하나요?

`aws:sourceIp`를 포함한 정책에서 이중 주소 지정을 사용하는 경우, 이 변경이 필요합니다. 이중 주소 지정이란 네트워크에서 IPv4와 IPv6를 모두 지원한다는 의미입니다.

이중 주소 지정을 사용하는 경우, 현재 IPv4 형식 주소만 사용하는 IAM 정책을 IPv6 형식 주소도 포함하도록 업데이트해야 합니다.

누가 이 변경을 하지 않아도 되나요?

IPv4 네트워크만 사용하는 경우, 이 변경은 필요하지 않습니다.

IAM 정책에 IPv6 추가

IAM 정책은 `aws:SourceIp` 조건 키를 사용하여 특정 IP 주소에서의 액세스를 제어합니다. 네트워크가 이중 주소 지정(IPv4 및 IPv6)을 사용하는 경우, IAM 정책을 IPv6 주소 범위도 포함하도록 업데이트해야 합니다.

정책의 `Condition` 요소에서는 IP 주소 조건에 `IpAddress` 및 `NotIpAddress` 연산자를 사용합니다. 문자열 연산자는 다양한 유효 IPv6 주소 형식을 처리할 수 없으므로 사용하지 마세요.

이 예제에서는 `aws:SourceIp`를 사용했습니다. VPC의 경우 `aws:VpcSourceIp`를 대신 사용하세요.

다음은 IAM 사용 설명서의 소스 IP 참조 정책에 AWS 따라에 대한 액세스 거부입니다. `Condition` 요소의 `NotIpAddress`에 두 개의 IPv4 주소 범위(`192.0.2.0/24` 및 `203.0.113.0/24`)가 나열되어 있으며, 이 범위의 주소는 API에 대한 액세스가 거부됩니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Deny",
        "Action": "*",
        "Resource": "*",
        "Condition": {
            "NotIpAddress": {
                "aws:SourceIp": [
                    "192.0.2.0/24",
                    "203.0.113.0/24"
                ]
            },
            "Bool": {
                "aws:ViaAWSService": "false"
            }
        }
    }
}
```

이 정책을 업데이트하려면 IPv6 주소 범위 2001:DB8:1234:5678::/64 및 2001:cdba:3257:8593::/64를 포함하도록 Condition 요소를 변경합니다.

> **Note**
>
> 기존 IPv4 주소는 삭제하지 마세요. 하위 호환성을 위해 필요합니다.

```
"Condition": {
                "NotIpAddress": {
                    "aws:SourceIp": [
                        "192.0.2.0/24", <<DO NOT REMOVE existing IPv4 address>>
                        "203.0.113.0/24", <<DO NOT REMOVE existing IPv4 address>>
                        "2001:DB8:1234:5678::/64", <<New IPv6 IP address>>
                        "2001:cdba:3257:8593::/64" <<New IPv6 IP address>>
                    ]
                },
                "Bool": {
                    "aws:ViaAWSService": "false"
                }
            }
```

이 정책을 VPC에 맞게 업데이트하려면 aws:SourceIp 대신 aws:VpcSourceIp를 사용하세요.

```
"Condition": {
                "NotIpAddress": {
                    "aws:VpcSourceIp": [
                        "10.0.2.0/24", <<DO NOT REMOVE existing IPv4 address>>
                        "10.0.113.0/24", <<DO NOT REMOVE existing IPv4 address>>
                        "fc00:DB8:1234:5678::/64", <<New IPv6 IP address>>
                        "fc00:cdba:3257:8593::/64" <<New IPv6 IP address>>
                    ]
                },
                "Bool": {
                    "aws:ViaAWSService": "false"
                }
            }
```

클라이언트가 IPv6를 지원하는지 확인하기

secretsmanager.{region}.amazonaws.com 엔드포인트를 사용하는 경우, 연결이 가능한지 확인해야 합니다. 다음 단계에서는 확인을 수행하는 방법을 설명합니다.

이 예제에서는 Linux와 curl 8.6.0 버전을 사용하며, IPv6가 활성화된 AWS Secrets Manager 서비스 엔드포인트(amazonaws.com)를 대상으로 합니다.

> **Note**
>
> secretsmanager.{region}.amazonaws.com은 일반적인 듀얼 스택 명명 규칙과 다릅니다. Secrets Manager 엔드포인트의 전체 목록은 AWS Secrets Manager 엔드포인트 섹션을 참조하세요.
> 를 서비스가 위치한 리전과 동일한 리전 AWS 리전 으로 변경합니다. 이 예제에서는 미국 동부(버지니아 북부) – `us-east-1` 엔드포인트를 사용합니다.

1. 다음 `dig` 명령을 사용하여 엔드포인트가 IPv6 주소로 환원되는지 확인합니다.

```
$ dig +short AAAA secretsmanager.us-east-1.amazonaws.com

> 2600:1f18:e2f:4e05:1a8a:948e:7c08:c1c3
```

2. 이제 다음 `curl` 명령을 사용하여 클라이언트 네트워크에서 IPv6 연결을 만들 수 있는지 확인합니다. 404 응답 코드는 연결 성공, 0 응답 코드는 연결 실패를 의미합니다.

```
$ curl --ipv6 -o /dev/null --silent -w "\nremote ip: %{remote_ip}\nresponse code:
 %{response_code}\n" https://secretsmanager.us-east-1.amazonaws.com

> remote ip: 2600:1f18:e2f:4e05:1a8a:948e:7c08:c1c3
> response code: 404
```

원격 IP 주소가 식별되었으며 동시에 응답 코드가 `0`이 아닌 경우 IPv6를 사용하여 엔드포인트에 네트워크가 성공적으로 연결되었습니다. 운영 체제가 클라이언트에 유효한 프로토콜을 선택해야 하므로 원격 IP가 IPv6 주소여야 합니다.

원격 IP가 비어 있거나 응답 코드가 `0`인 경우 클라이언트 네트워크 또는 엔드포인트에 대한 네트워크 경로는 IPv4 전용입니다. 다음 `curl` 명령으로 이 구성을 확인할 수 있습니다.

```
$ curl -o /dev/null --silent -w "\nremote ip: %{remote_ip}\nresponse code:
 %{response_code}\n" https://secretsmanager.us-east-1.amazonaws.com

> remote ip: 3.123.154.250
> response code: 404
```

의 복원력 AWS Secrets Manager

AWS 는 AWS 리전 및 가용 영역을 중심으로 글로벌 인프라를 구축합니다.는 물리적으로 분리되고 격리된 여러 가용 영역을 AWS 리전 제공합니다.이 가용 영역은 지연 시간이 짧고 처리량이 높으며 중복성이 높은 네트워킹과 연결됩니다. 가용 영역을 사용하면 중단 없이 영역 간에 자동으로 장애 극복 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

복원력 및 재해 복구에 대한 자세한 내용은 신뢰성 원칙 - AWS Well-Architected Framework를 참조하세요.

AWS 리전 및 가용 영역에 대한 자세한 내용은 AWS 글로벌 인프라를 참조하세요.

포스트 양자 TLS

Secrets Manager는 전송 계층 보안(TLS) 네트워크 암호화 프로토콜에 대한 하이브리드 포스트 퀀텀 키 교환 옵션을 지원합니다. Secrets Manager API 엔드포인트에 연결할 때 이 TLS 옵션을 사용할 수 있습니다. 포스트 양자 알고리즘이 표준화되기 전에 이 기능을 제공하므로 이러한 키 교환 프로토콜이 Secrets Manager 호출에 미치는 영향을 테스트할 수 있습니다. 선택 사항인 이 하이브리드 포스트 양자 키 교환 기능은 적어도 우리가 현재 사용하는 TLS 암호화만큼 안전하며 보안 이점을 추가로 제공할 수도 있습니다. 그러나 현재 사용 중인 클래식 키 교환 프로토콜과 달리 지연 시간 및 처리량에 영향을 미칩니다. Secrets Manager Agent는 기본적으로 포스트 양자 ML-KEM 키 교환을 최우선 키 교환 방식으로 사용합니다.

현재 암호화된 데이터를 잠재적 향후 공격으로부터 보호하기 위해 AWS 는 양자 내성 또는 양자 사후 알고리즘 개발에 암호화 커뮤니티에 참여하고 있습니다. Secrets Manager 엔드포인트에서 하이브리드 포스트 양자 키 교환 암호 제품군을 구현했습니다. 클래식 및 포스트 양자 요소를 결합한 이 하이브리드 암호 제품군을 통해 TLS 연결은 적어도 클래식 암호 제품군과 동등한 수준으로 강력해집니다. 그러나 하이브리드 암호 제품군의 성능 특성 및 대역폭 요구 사항은 클래식 키 교환 메커니즘의 해당 요구 사항과 다르기 때문에 API 직접 호출에 대해 이 제품군을 테스트하는 것이 좋습니다.

Secrets Manager는 중국 리전을 제외한 모든 리전에서 PQTLS를 지원합니다.

하이브리드 포스트 양자 TLS 구성

1. Maven 종속성에 AWS 공통 런타임 클라이언트를 추가합니다. 사용 가능한 최신 버전을 사용하는 것이 좋습니다. 예를 들어 이 문은 2.20.0 버전을 추가합니다.

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>aws-crt-client</artifactId>
  <version>2.20.0</version>
</dependency>
```

2. 프로젝트에 Java 2.x용 AWS SDK를 추가하고 초기화합니다. HTTP 클라이언트에서 하이브리드 포스트 양자 암호 제품군을 활성화합니다.

```
SdkAsyncHttpClient awsCrtHttpClient = AwsCrtAsyncHttpClient.builder()
            .postQuantumTlsEnabled(true)
            .build();
```

3. Secrets Manager 비동기 클라이언트를 생성합니다.

```
SecretsManagerAsyncClient SecretsManagerAsync = SecretsManagerAsyncClient.builder()
            .httpClient(awsCrtHttpClient)
            .build();
```

   이제 Secrets Manager API 작업을 호출하면 호출은 하이브리드 포스트 양자 TLS를 사용하여 Secrets Manager 엔드포인트로 전송됩니다.

하이브리드 포스트 양자 TLS 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS SDK for Java 2.x 개발자 안내서 및 AWS SDK for Java 2.x 릴리스된 블로그 게시물.
- 새 오픈 소스 TLS 구현, 즉 s2n-tls 소개 및 s2n-tls 사용.
- 미국 국립 표준 기술 연구소 (NIST)에서의 포스트 양자 암호화.
- TLS(전송 계층 보안) 1.2에 대한 PQ KEM(하이브리드 포스트 양자 키 캡슐화 방법).

Secrets Manager용 포스트 양자 TLS는 중국을 AWS 리전 제외한 모든에서 사용할 수 있습니다.

문제 해결 AWS Secrets Manager

여기 정보를 사용하여 Secrets Manager 작업 시 발생할 수 있는 문제를 진단하고 수정하세요.

교체와 관련된 문제는 the section called “교체 문제 해결” 섹션을 참조하세요.

주제

- ‘액세스가 거부됨’ 메시지
- 임시 보안 자격 증명에 대한 “액세스 거부됨”이라는 메시지 발생
- 변경 사항이 경우에 따라 즉시 표시되지 않습니다.
- 보안 암호를 생성할 때 “비대칭 KMS 키로 데이터 키를 생성할 수 없습니다.”라는 메시지 발생
- AWS CLI 또는 AWS SDK 작업이 부분 ARN에서 내 보안 암호를 찾을 수 없음
- 이 보안 암호는 AWS 서비스에서 관리하며 해당 서비스를 사용하여 업데이트해야 합니다.
- Transform: AWS::SecretsManager-2024-09-16 사용 시 Python 모듈 가져오기 실패

‘액세스가 거부됨’ 메시지

GetSecretValue 또는 CreateSecret to Secrets Manager 같은 API를 직접적으로 호출할 경우, 이러한 호출을 할 수 있는 IAM 권한이 있어야 합니다. 콘솔을 사용할 때 콘솔은 사용자를 대신하여 동일한 API 직접 호출을 수행하므로 IAM 권한도 있어야 합니다. IAM 정책을 여러분의 IAM 사용자나 여러분이 멤버인 그룹에 연결하면 관리자가 권한을 부여할 수 있습니다. 이러한 권한을 부여하는 정책 명령문에 시간이나 IP 주소 제한 정책이 포함된다면, 요청을 전송할 때 이러한 요구사항을 충족해야 합니다. IAM 사용자, 그룹 또는 역할에 대한 정책을 확인 또는 수정하는 방법은 IAM 사용 설명서에 있는 정책 작업 섹션을 참조하세요. Secrets Manager에 필요한 권한에 대한 자세한 정보는 the section called “인증 및 액세스 제어” 섹션을 참조하세요.

AWS SDK를 사용하지 않고 API 요청에 수동으로 서명할 경우 올바르게 요청에 서명했는지 확인합니다.

임시 보안 자격 증명에 대한 “액세스 거부됨”이라는 메시지 발생

요청을 위해 사용 중인 IAM 사용자나 역할에 올바른 권한이 있는지 확인합니다. 임시 보안 자격 증명에 대한 권한은 IAM 사용자 또는 역할에서 파생됩니다. 이는 권한이 IAM 사용자 또는 역할에 부여된 권한으로 제한됨을 의미합니다. 임시 보안 자격 증명의 권한이 결정되는 방법에 대한 자세한 내용은 IAM 사용 설명서의 임시 보안 자격 증명을 위한 권한 제어 섹션을 참조하세요.

요청에 올바르게 서명했고 요청이 잘 구성되었는지 확인합니다. 자세한 내용은 IAM 사용 설명서의 선택한 SDK에 대한 도구 키트 설명서 또는 임시 보안 자격 증명을 사용하여 AWS 리소스에 대한 액세스 요청을 참조하세요.

임시 보안 자격 증명이 만료되지 않았는지 확인합니다. 자세한 내용은 IAM 사용 설명서의 임시 보안 자격 증명 요청 섹션을 참조하세요.

Secrets Manager에 필요한 권한에 대한 자세한 정보는 the section called “인증 및 액세스 제어” 섹션을 참조하세요.

변경 사항이 경우에 따라 즉시 표시되지 않습니다.

Secrets Manager는 최종 일관성이라는 분산 컴퓨팅 모델을 사용합니다. Secrets Manager(또는 기타 AWS 서비스)에서 변경한 내용은 가능한 모든 엔드포인트에서 표시되는 데 시간이 걸립니다. 일부 지연은 서버에서 서버로, 복제 영역에서 복제 영역으로, 전세계의 리전에서 리전으로 데이터를 보내는 데 걸리는 시간으로 인해 발생합니다. 또한 Secrets Manager는 성능 향상을 위해 캐싱을 사용하지만, 어떤 경우에는 이로 인해 시간이 더 걸릴 수 있습니다. 이러한 변화는 이전에 캐싱된 데이터가 끝날 때까지 가시화되지 않을 수 있습니다.

이러한 잠재적 지연을 고려하도록 전역 애플리케이션을 설계합니다. 또한 한 위치에서 변경한 내용이 다른 위치에서 즉시 보이지 않을 때조차도 예상대로 작동하는지 확인합니다.

일부 다른 AWS 서비스가 최종 일관성의 영향을 받는 방식에 대한 자세한 내용은 다음을 참조하세요.

- Amazon Redshift 데이터베이스 개발자 가이드의 데이터 일관성 관리
- Amazon Simple Storage Service 사용 설명서의 Amazon S3 데이터 일관성 모델
- AWS 빅 데이터 블로그에서 ETL 워크플로에 대해 Amazon S3 및 Amazon EMR 사용 시 일관성 유지
- Amazon EC2 API 참조의 Amazon EC2 최종 일관성

보안 암호를 생성할 때 “비대칭 KMS 키로 데이터 키를 생성할 수 없습니다.”라는 메시지 발생

Secrets Manager는 보안 암호와 연결된 대칭 암호화 KMS 키를 사용하여 각 보안 암호 값에 대한 데이터 키를 생성합니다. 비대칭 KMS 키를 사용할 수 없습니다. 비대칭 KMS 키 대신 대칭 암호화 KMS 키를 사용하고 있는지 확인하세요. 지침은 비대칭 KMS 키 식별 섹션을 참조하세요.

AWS CLI 또는 AWS SDK 작업이 부분 ARN에서 내 보안 암호를 찾을 수 없음

많은 경우 Secrets Manager는 전체 ARN이 아닌 ARN의 일부에서 보안 암호를 찾을 수 있습니다. 그러나, 보안 암호 이름이 하이픈 다음에 6자로 끝나는 경우 Secrets Manager는 ARN의 일부에서만 보안 암호를 찾지 못할 수 있습니다. 그 대신 전체 ARN 또는 암호 이름을 사용할 것을 권장합니다.

추가 세부 정보

Secrets Manager는 보안 암호 ARN이 고유한지 확인하기 위해 보안 암호 이름의 끝에 임의의 문자 6개를 포함합니다. 원래 보안 암호를 삭제한 후 동일한 이름으로 새 보안 암호를 생성하면 해당 문자로 인해 두 보안 암호의 ARN이 달라집니다. ARN이 다르기 때문에 이전 보안 암호에 액세스할 수 있는 사용자는 새 보안 암호에 자동으로 액세스할 수 없습니다.

Secrets Manager는 다음과 같이 리전, 계정, 보안 암호 이름, 하이픈 및 추가 6자를 사용하여 보안 암호에 대한 ARN을 구성합니다.

```
arn:aws:secretsmanager:us-east-2:111122223333:secret:SecretName-abcdef
```

보안 암호 이름이 하이픈과 6자로 끝나는 경우 ARN의 일부만 사용하면 Secrets Manager에 전체 ARN을 지정하는 것처럼 보일 수 있습니다. 예를 들어 ARN과 함께 이름이 `MySecret-abcdef`인 보안 암호가 있을 수 있습니다.

```
arn:aws:secretsmanager:us-east-2:111122223333:secret:MySecret-abcdef-nutBrk
```

보안 암호 ARN의 일부만 사용하는 다음 작업을 호출할 경우, Secret Manager가 해당 보안 암호를 찾지 못할 수 있습니다.

```
$ aws secretsmanager describe-secret --secret-id arn:aws:secretsmanager:us-
east-2:111122223333:secret:MySecret-abcdef
```

이 보안 암호는 AWS 서비스에서 관리하며 해당 서비스를 사용하여 업데이트해야 합니다.

보안 암호를 수정하려고 할 때 이 메시지가 나타나면 메시지에 나열된 관리 서비스를 사용해야만 보안 암호를 업데이트할 수 있습니다. 자세한 내용은 다른 서비스에서 관리하는 보안 암호 단원을 참조하십시오.

보안 암호 관리 담당자를 결정하기 위해 보안 암호 이름을 검토할 수 있습니다. 다른 서비스에서 관리하는 보안 암호는 해당 서비스의 ID가 접두사로 붙습니다. 또는에서 describe-secret을 AWS CLI호출한 다음 필드를 검토합니다`OwningService`.

`Transform: AWS::SecretsManager-2024-09-16` 사용 시 Python 모듈 가져오기 실패

만약 변환: `AWS::SecretsManager-2024-09-16`를 사용하면서 교체 Lambda 함수 실행 시 Python 모듈 가져오기 오류가 발생하면, 이는 `Runtime` 값이 호환되지 않아서 생긴 문제일 가능성이 높습니다. 이 변환 버전에서는 AWS CloudFormation 이 런타임 버전, 코드, 공유 객체 파일을 관리합니다. 따라서 사용자가 별도로 관리할 필요가 없습니다.

AWS Secrets Manager 할당량

Secrets Manager 읽기 API는 TPS 할당량이 높고, 호출 빈도가 낮은 컨트롤 플레인 API는 TPS 할당량이 낮습니다. 10분마다 두 번 이상 지속적으로 `PutSecretValue` 또는 `UpdateSecret`을 호출하지 않는 것이 좋습니다. `PutSecretValue` 또는 `UpdateSecret`을 호출하여 보안 암호 값을 업데이트하면 Secrets Manager는 새 버전의 보안 암호를 생성합니다. Secrets Manager는 100개를 넘는 버전이 있을 때 레이블이 지정되지 않은 버전을 제거하지만 24시간 이내에 생성된 버전은 제거하지 않습니다. 10분마다 두 번 이상 보안 암호 값을 업데이트하면 Secrets Manager에서 제거하는 버전보다 더 많은 버전이 생성되고 보안 암호 버전 할당량에 도달하게 됩니다.

계정에서 여러 리전을 운영할 수 있으며 각 할당량은 각 리전에 따라 다릅니다.

한의 애플리케이션이 다른 계정이 소유한 보안 암호를 AWS 계정 사용하는 경우 이를 교차 계정 요청이라고 합니다. 교차 계정 요청의 경우, Secrets Manager는 보안 암호를 소유한 계정이 아니라 요청하는 자격 증명의 계정을 제한합니다. 예를 들어 계정 A의 자격 증명이 계정 B의 보안 암호를 사용하는 경우 보안 암호 사용은 계정 A의 할당량에만 적용됩니다.

Secrets Manager 할당량

이름	기본값	조정 가능	설명
DeleteResourcePolicy, GetResourcePolicy, PutResourcePolicy, ValidateResourcePolicy API 요청의 합산 비율	각각 지원되는 리전: 초당 50개	아니요	DeleteResourcePolicy, GetResourcePolicy, PutResourcePolicy, ValidateResourcePolicy API 요청의 합산에 대한 초당 최대 트랜잭션 수입니다.
PutSecretValue, RemoveRegionsFromReplication, ReplicateSecretToRegion, StopReplicationToReplica, UpdateSecret, UpdateSecretVersionStage API 요청의 합산 비율	각각 지원되는 리전: 초당 50개	아니요	PutSecretValue, RemoveRegionsFromReplication, ReplicateSecretToRegion, StopReplicationToR

이름	기본값	조정 가능	설명
			eplica, UpdateSecret, UpdateSecretVersionStage API 요청의 합산에 대한 초당 최대 트랜잭션 수입니다.
RestoreSecret API 요청의 합산 비율	각각 지원되는 리전: 초당 50개	아니요	RestoreSecret API 요청에 대한 초당 최대 트랜잭션 수입니다.
RotateSecret 및 CancelRotateSecret API 요청의 합산 비율	각각 지원되는 리전: 초당 50개	아니요	RotateSecret 및 CancelRotateSecret API 요청의 합산에 대한 초당 최대 트랜잭션 수입니다.
TagResource 및 UntagResource API 요청의 합산 비율	각각 지원되는 리전: 초당 50개	아니요	TagResource 및 UntagResource API 요청의 합산에 대한 초당 최대 트랜잭션 수입니다.
BatchGetSecretValue API 요청 비율	지원되는 리전별: 초당 100개	아니요	BatchGetSecretValue API 요청에 대한 초당 최대 트랜잭션 수입니다.
CreateSecret API 요청 비율	각각 지원되는 리전: 초당 50개	아니요	CreateSecret API 요청에 대한 초당 최대 트랜잭션 수입니다.
DeleteSecret API 요청 비율	각각 지원되는 리전: 초당 50개	아니요	DeleteSecret API 요청에 대한 초당 최대 트랜잭션 수입니다.

이름	기본값	조정 가능	설명
DescribeSecret API 요청 속도	지원되는 각 리전: 초당 40,000개	아니요	DescribeSecret API 요청에 대한 초당 최대 트랜잭션 수입니다.
GetRandomPassword API 요청 비율	각각 지원되는 리전: 초당 50개	아니요	GetRandomPassword API 요청에 대한 초당 최대 트랜잭션 수입니다.
GetSecretValue API 요청 속도	지원되는 각 리전: 초당 10,000개	아니요	GetSecretValue API 요청에 대한 초당 최대 트랜잭션 수입니다.
ListSecretVersionIds API 요청 비율	각각 지원되는 리전: 초당 50개	아니요	ListSecretVersionIds API 요청에 대한 초당 최대 트랜잭션 수입니다.
ListSecrets API 요청 비율	지원되는 리전별: 초당 100개	아니요	ListSecrets API 요청에 대한 초당 최대 트랜잭션 수입니다.
리소스 기반 정책 길이	각각 지원되는 리전: 20,480	아니요	보안 암호에 연결된 리소스 기반 권한 정책의 최대 문자 수입니다.
보안 암호 값 크기	각각 지원되는 리전: 65,536바이트	아니요	암호화된 보안 암호 값의 최대 크기입니다. 암호 값이 문자열인 경우 이는 암호 값에 허용되는 문자 수입니다.
보안 암호	각각 지원되는 리전: 500,000	아니요	이 AWS 계정의 각 AWS 리전에 있는 최대 보안 암호 수입니다.

이름	기본값	조정 가능	설명
보안 암호의 모든 버전에 연결된 스테이징 레이블	지원되는 각 리전: 20	아니요	보안 암호의 모든 버전에 연결된 스테이징 레이블의 최대 수입니다.
보안 암호별 버전	지원되는 각 리전: 100	아니요	보안 암호의 최대 버전 수입니다.

애플리케이션에 재시도 추가

AWS 클라이언트 측에서 예기치 않은 문제로 인해 Secrets Manager에 대한 호출이 실패할 수 있습니다. 또는 Secrets Manager의 속도 제한으로 인해 호출이 실패할 수 있습니다. API 요청 할당량을 초과하면 Secrets Manager에서 요청을 제한합니다. Secrets Manager는 다른 경우라면 유효한 요청을 거부하고 throttling 오류를 반환합니다. 두 유형의 실패 모두 짧은 대기 기간 후에 다시 호출하는 것이 좋습니다. 이것을 백오프 및 재시도 전략이라고 부릅니다.

다음과 같은 오류가 발생할 경우 애플리케이션 코드에 재시도를 추가할 수 있습니다.

일시적 오류 및 예외

- `RequestTimeout`
- `RequestTimeoutException`
- `PriorRequestNotComplete`
- `ConnectionError`
- `HTTPClientError`

서비스 측 조절 및 제한 오류와 예외

- `Throttling`
- `ThrottlingException`
- `ThrottledException`

- `RequestThrottledException`
- `TooManyRequestsException`
- `ProvisionedThroughputExceededException`
- `TransactionInProgressException`
- `RequestLimitExceeded`
- `BandwidthLimitExceeded`
- `LimitExceededException`
- `RequestThrottled`
- `SlowDown`

재시도, 지수 백오프 및 지터에 대한 자세한 내용 및 예시 코드는 다음 리소스를 참조하세요.

- 지수 백오프 및 지터
- 시간 초과, 지터를 사용한 재시도 및 백오프
- 오류 재시도 및 지수 백오프. AWS

문서 기록

다음 표에서는 마지막 릴리스 이후 설명서에 대한 중요한 변경 사항을 설명합니다 AWS Secrets Manager. 이 설명서에 대한 업데이트 알림을 받으려면 RSS 피드를 구독하면 됩니다.

변경 사항	설명	날짜
새로운 AWS 관리형 정책	Secrets Manager는 클라이언트 애플리케이션의 보안 암호에 대한 읽기 전용 액세스를 `AWSSecretsManagerClientReadOnlyAccess` 제공하는 새로운 관리형 정책을 릴리스했습니다. 자세한 내용은 AWS 관리형 정책에 대한 Secrets Manager 업데이트를 참조하세요.	2025년 11월 5일
비용 할당 태그 지원 추가	Secrets Manager가 비용 할당 태그를 지원함에 따라 이제 부서, 팀, 애플리케이션별로 비용을 분류하고 추적할 수 있습니다. 자세한 내용은에서 비용 할당 태그 사용을 참조하세요 AWS Secrets Manager.	2025년 5월 27일
IPv6 및 듀얼 스택 지원 추가	Secrets Manager가 이제 듀얼 스택 엔드포인트를 지원합니다. 자세한 내용은 IPv4 및 IPv6 액세스를 참조하세요.	2024년 12월 20일
Secrets Manager를 AWS 관리형 정책으로 변경	`SecretsManagerReadWrite` 관리형 정책에 이제 `redshift-serverless` 권한이 포함됩니다. 자세한 내용은에 대한 AWS 관리형 정책	2024년 3월 12일

을 참조하세요 AWS Secrets Manager.

이전 업데이트

다음 표에서는 2024년 2월 이전 AWS Secrets Manager 사용 설명서의 각 릴리스에서 변경된 중요 사항에 대해 설명합니다.

변경	설명	Date
정식 출시	Secrets Manager의 최초 공개 릴리스입니다.	2018년 4월 4일