



ユーザーガイド

AWS Secrets Manager



AWS Secrets Manager: ユーザーガイド

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

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

Table of Contents

Secrets Manager とは	1
Secrets Manager の使用を開始する	1
標準への準拠	2
料金	2
Secrets Manager にアクセスする	4
Secrets Manager コンソール	4
コマンドラインツール	4
AWS SDKs	5
HTTPS クエリ API	5
Secrets Manager エンドポイント	6
シークレットの内容	11
メタデータ	11
シークレットバージョン	12
チュートリアル	14
Amazon CodeGuru Reviewer	14
ハードコードされたシークレットの置き換え	14
ステップ 1: シークレットを作成する	15
ステップ 2: コードを更新する	17
ステップ 3: シークレットを更新する	18
次のステップ	18
ハードコードされたデータベース認証情報の置き換え	19
ステップ 1: シークレットを作成する	20
ステップ 2: コードを更新する	21
ステップ 3: シークレットをローテーションする	22
次のステップ	23
交代ユーザーローテーション	24
許可	25
前提条件	25
ステップ 1: Amazon RDS データベースユーザーを作成する	28
ステップ 2: ユーザーの認証情報用のシークレットを作成する	31
ステップ 3: ローテーションされたシークレットをテストする	32
ステップ 4: リソースをクリーンアップする	33
次のステップ	34
シングルユーザーローテーション	34

許可	35
前提条件	35
ステップ 1: Amazon RDS データベースユーザーを作成する	35
ステップ 2: データベースユーザー認証情報のシークレットを作成する	36
ステップ 3: ローテーションされたパスワードをテストする	37
ステップ 4: リソースをクリーンアップする	38
次のステップ	38
認証とアクセスコントロール	39
Secrets Manager 管理者のアクセス許可	39
シークレットへのアクセス許可	39
Lambda ローテーション関数のアクセス許可	40
暗号化キーのアクセス許可	40
レプリケーションのアクセス許可	40
アクセス許可ポリシーをアイデンティティにアタッチする	40
アクセス許可ポリシーをシークレットにアタッチする	41
AWS CLI	42
AWS SDK	43
AWS 管理ポリシー	44
SecretsManagerReadWrite	44
ポリシーの更新	46
シークレットへのアクセス許可を持つユーザーを特定する	48
クロスアカウントアクセス	49
オンプレミスアクセス	51
アクセス許可ポリシーの例	51
例: 個別のシークレット値を取得するためのアクセス許可	53
例: 個々のシークレットを読み書きするアクセス許可	54
例: バッチ内のシークレット値のグループを取得するアクセス許可	55
例: ワイルドカード	56
例: シークレットを作成するアクセス許可	57
例: シークレットを暗号化するための特定の AWS KMS キーを拒否する	58
例: アクセス許可と VPC	59
例: タグを使用してシークレットへのアクセスを制御する	61
例: シークレットのタグに一致しているタグを持つアイデンティティへのアクセスを制限する	62
例: サービスプリンシパル	63
アクセス許可に関するリファレンス	64

Secrets Manager のアクション	65
Secrets Manager リソース	85
条件キー	86
BlockPublicPolicy 条件	89
IP アドレス条件	89
VPC エンドポイントの条件	90
シークレットの作成と管理	91
データベースシークレットを作成する	91
AWS CLI	94
AWS SDK	94
シークレットの JSON 構造	94
Amazon RDS Db2 シークレット構造	95
Amazon RDS MariaDB シークレット構造	96
Amazon RDS と Amazon Aurora MySQL のシークレット構造	96
Amazon RDS Oracle シークレット構造	97
Amazon RDS と Amazon Aurora PostgreSQL のシークレット構造	97
Amazon RDS Microsoft SQLServer シークレット構造	98
Amazon DocumentDB シークレット構造	98
Amazon Redshift のシークレット構造	99
Amazon Redshift Serverless シークレット構造	100
Amazon ElastiCache シークレット構造	100
Active Directory シークレット構造	101
シークレットを作成する	102
AWS CLI	105
AWS SDK	106
シークレット値の更新	106
AWS CLI	106
AWS SDK	107
Secrets Manager でパスワードを生成する	107
シークレットの暗号化キーを変更する	107
AWS CLI	109
シークレットの変更	110
AWS CLI	111
AWS SDK	111
シークレットを検索する	112
AWS CLI	113

AWS SDK	114
シークレットの削除	114
AWS CLI	116
AWS SDK	116
シークレットを復元する	117
AWS CLI	117
AWS SDK	118
シークレットにタグ付けする	118
AWS CLI	119
AWS SDK	119
リージョン間でシークレットをレプリケートする	120
AWS CLI	122
AWS SDK	122
レプリカシークレットをスタンドアロンシークレットに昇格させる	122
AWS CLI	123
AWS SDK	123
レプリケーションの防止	124
レプリケーションをトラブルシューティングする	125
選択したリージョンに同じ名前のシークレットがある	125
KMS キーにレプリケーションを完成させるためのアクセス許可がない	125
KMS キーが無効になっているか、見つかりません	126
レプリケーションを行うリージョンが有効化されていない	126
シークレットを取得する	127
Java	127
Java とクライアント側のキャッシュ	128
シークレット内の認証情報を使用した JDBC 接続	135
Java AWS SDK	145
Python	147
Python とクライアント側のキャッシュ	147
Python AWS SDK	153
シークレット値をバッチ取得する	154
.NET	156
.NET とクライアント側のキャッシュ	156
.NET AWS SDK	163
Go	166
クライアント側のキャッシュを使用する	166

Go AWS SDK	171
C++	172
JavaScript	173
Kotlin	174
PHP	175
Ruby	176
Rust	177
AWS CLI	177
を使用してバッチでシークレットのグループを取得する AWS CLI	178
AWS コンソール	179
AWS Batch	179
AWS CloudFormation	179
Amazon EKS	181
ステップ 1: アクセス制御を設定する	182
ステップ 2: ASACP をインストールして設定する	182
ステップ 3: マウントするシークレットを特定する	184
ステップ 4: Amazon EKS ポッドにシークレットをファイルとしてマウントする	187
トラブルシューティング	187
SecretProviderClass	188
GitHub ジョブ	191
前提条件	192
使用方法	192
環境変数の命名	193
例	194
AWS IoT Greengrass	197
AWS Lambda	197
環境変数	200
パラメータストア	202
シークレットのローテーション	203
マネージドローテーション	203
Lambda 関数によるローテーション	205
データベースシークレットの自動ローテーション	206
データベース以外のシークレットの自動ローテーション	210
Lambda 関数のローテーション戦略	215
Lambda ローテーション関数	217
ローテーション関数のテンプレート	221

ローテーションへのアクセス許可	228
Lambda ローテーション関数のネットワークアクセス	232
におけるローテーションのトラブルシューティング	233
すぐにシークレットをローテーションする	242
AWS CLI	242
ローテーションスケジュール	243
Rate 式	243
Cron 式	244
ローテーションされていないシークレットを検索する	249
自動ローテーションをキャンセルする	250
マネージドシークレット	251
シークレットを使用するサービス	253
App Runner	254
AWS App2Container	255
AWS AppConfig	255
Amazon AppFlow	255
AWS AppSync	256
Amazon Athena	256
Amazon Aurora	256
AWS CodeBuild	257
AWS DataSync	257
Amazon DataZone	257
AWS Direct Connect	258
AWS Directory Service	258
Amazon DocumentDB	258
AWS Elastic Beanstalk	259
Amazon Elastic Container Registry	259
Amazon Elastic Container Service	259
Amazon ElastiCache	260
AWS Elemental Live	260
AWS Elemental MediaConnect	261
AWS Elemental MediaConvert	261
AWS Elemental MediaLive	261
AWS Elemental MediaPackage	261
AWS Elemental MediaTailor	262
Amazon EMR	262

EMR on EC2	262
EMR Serverless	262
Amazon EventBridge	263
Amazon FSx	263
AWS Glue DataBrew	263
AWS Glue Studio	264
AWS IoT SiteWise	264
Amazon Kendra	264
Amazon Kinesis Video Streams	264
AWS Launch Wizard	265
Amazon Lookout for Metrics	265
Amazon Managed Grafana	265
AWS Managed Services	266
Amazon Managed Streaming for Apache Kafka	266
Amazon Managed Workflows for Apache Airflow	266
AWS Marketplace	266
AWS Migration Hub	267
AWS Panorama	267
AWS ParallelCluster	267
Amazon Q	268
AWS OpsWorks for Chef Automate	268
Amazon QuickSight	268
Amazon RDS	268
Amazon Redshift	269
Amazon Redshift クエリエディタ v2	269
Amazon SageMaker	270
AWS SCT	270
AWS Toolkit for JetBrains	271
AWS Transfer Family	271
AWS ウィッカー	272
VPC エンドポイント	273
共有サブネット	274
AWS CloudFormation	275
シークレットを作成する	276
JSON	276
YAML	277

自動ローテーション付きの Amazon RDS 認証情報を使ってシークレットを作成する	277
Amazon Redshift 認証情報を使用してシークレットを作成する	277
Amazon DocumentDB 認証情報を使用してシークレットを作成する	277
JSON	278
YAML	282
Secrets Manager が AWS CloudFormation を使用する方法	285
AWS CDK	286
シークレットをモニタリングする	287
でログ記録する AWS CloudTrail	287
AWS CLI	288
CloudTrail エントリ	288
によるモニタリング CloudWatch	294
CloudWatch アラーム	295
Secrets Manager イベントを と一致させる EventBridge	295
指定したシークレットに対するすべての変更 matches	296
シークレット値がローテーションされたらイベントを matches	296
削除予定のシークレットのモニタリング	297
ステップ 1: CloudTrail ログへのログファイル配信を設定する CloudWatch	297
ステップ 2: アラームを作成する CloudWatch	298
ステップ 3: アラームを CloudWatch テストする	299
シークレットのコンプライアンスをモニタリングする	300
Secrets Manager のコストをモニタリングする	301
コンプライアンス検証	302
コンプライアンス標準	302
Secrets Manager のセキュリティ	305
AWS CLI を使用して AWS Secrets Manager のシークレットを保存する際のリスクを軽減する	305
Secrets Manager でのデータ保護	308
保管中の暗号化	308
転送中の暗号化	309
ネットワーク間トラフィックのプライバシー	309
暗号化キーの管理	310
シークレット暗号化と復号	310
AWS KMS キーの選択	311
暗号化されるもの	311
プロセスの暗号化と復号	312

KMS キーのアクセス許可	312
Secrets Manager による KMS キーの使用方法	313
キーポリシー AWS マネージドキー (aws/secretsmanager)	315
Secrets Manager の暗号化コンテキスト	317
Secrets Manager と のインタラク션을 모니터링 하는 AWS KMS	319
인프라스트럭처 보안	323
耐障害성	324
포스트 양자 TLS	324
트러블슈팅	327
「アクセス拒否」 메시지	327
一時的なセキュリティ 인증 정보에 「액세스가 거부되었습니다」로 표시되는	328
변경이 즉시 표시되지 않음.	328
시크릿을 생성할 때 「비대칭 KMS 키를 사용하여 데이터 키를 생성할 수 없습니다」라는 메시지가 표시되는	329
AWS CLI 또는 AWS SDK 操作이 부분적인 ARN 从 시크릿을 찾을 수	329
이 시크릿은 AWS 서비스 由 관리되는 爲, 서비스 使用하여 업데이트	330
필요가 있습니다.	330
쿼터	331
Secrets Manager 的 쿼터	331
애플리케이션 向의 재시행을 추가하는	335
ドキュメント履歷	337
이전의 업데이트	337
.....	cccxxxviii

とは AWS Secrets Manager

AWS Secrets Manager は、データベース認証情報、アプリケーション認証情報、OAuth トークン、API キー、およびその他のシークレットをライフサイクルを通じて管理、取得、ローテーションするのに役立ちます。多くの AWS サービスは、Secrets Manager でシークレットを保存して使用します。

Secrets Manager は、アプリケーションのソースコードにハードコーディングされた認証情報が不要になるため、セキュリティ態勢を改善するのに役立ちます。認証情報を Secrets Manager に保存することで、アプリケーションまたはそのコンポーネントを調べることができるすべてのユーザーによる侵害の可能性を回避できます。ハードコーディングされた認証情報を Secrets Manager サービスへのランタイム呼び出しに置き換えることができるため、必要に応じて認証情報を動的に取得できます。

Secrets Manager で、シークレットの自動ローテーションスケジュールを設定することができます。これにより、長期のシークレットを短期のシークレットに置き換えることが可能となり、侵害されるリスクを大幅に減少させるのに役立ちます。認証情報はアプリケーションに保存されなくなったため、認証情報を変更しても、アプリケーションを更新したり、アプリケーションクライアントに変更を反映させたりする必要はなくなりました。

組織内に存在する可能性のある他のタイプのシークレット:

- AWS 認証情報 – をお勧めします [AWS Identity and Access Management](#)。
- 暗号化キー — [AWS Key Management Service](#) を推奨
- SSH キー – [Amazon EC2 Instance Connect](#) を推奨
- プライベートキーと証明書 – [AWS Certificate Manager](#) を推奨

Secrets Manager の使用を開始する

Secrets Manager を初めて使用する場合は、次のいずれかのチュートリアルから始めます。

- [the section called “ハードコードされたシークレットの置き換え”](#)
- [the section called “ハードコードされたデータベース認証情報の置き換え”](#)
- [the section called “交代ユーザーローテーション”](#)
- [the section called “シングルユーザーローテーション”](#)

シークレットを使って実行できるその他のタスク:

- [シークレットの作成と管理](#)
- [シークレットへのアクセスを制御する](#)
- [シークレットを取得する](#)
- [シークレットのローテーション](#)
- [シークレットをモニタリングする](#)
- [シークレットのコンプライアンスをモニタリングする](#)
- [でシークレットを作成する AWS CloudFormation](#)

標準への準拠

AWS Secrets Manager は、複数の規格の監査を受けており、コンプライアンス認定を取得する必要がある場合にソリューションの一部になる可能性があります。詳細については、「[コンプライアンス検証](#)」を参照してください。

料金

Secrets Manager では、使用した分のみ料金が発生し、最低料金や設定料金はありません。削除対象としてマークされたシークレットに対しては料金は発生しません。現在の価格の詳細なリストについては、「[AWS Secrets Manager 料金表](#)」を参照してください。コストをモニタリングするには、「[」を参照してくださいthe section called “Secrets Manager のコストをモニタリングする”。](#)

Secrets Manager が AWS マネージドキー `aws/secretsmanager`作成する を使用して、シークレットを無料で暗号化できます。独自の KMS キーを作成してシークレットを暗号化すると、は現在の AWS KMS レートで AWS 課金します。詳細については、「[AWS Key Management Service の料金](#)」を参照してください。

自動ローテーション ([マネージドローテーション を除く](#)) を有効にすると、Secrets Manager は AWS Lambda 関数を使用してシークレットをローテーションし、ローテーション関数に対して現在の Lambda レートで課金されます。詳細については、「[AWS Lambda の料金](#)」を参照してください。

アカウント AWS CloudTrail で を有効にすると、Secrets Manager が送信する API コールのログを取得できます。Secrets Manager は、すべてのイベントを管理イベントとしてログに記録します。は、すべての管理イベントの最初のコピーを無料で AWS CloudTrail 保存します。ただし、通知を有効にすると、Amazon S3 のログストレージと Amazon SNS の料金が発生する場合があります。ま

た、追加の証跡を設定している場合、管理イベントの追加コピーについては、料金が発生する可能性があります。詳細については、「[AWS CloudTrail 料金](#)」を参照してください。

アクセス AWS Secrets Manager

Secrets Manager は、次の方法で利用できます。

- [Secrets Manager コンソール](#)
- [コマンドラインツール](#)
- [AWS SDKs](#)
- [HTTPS クエリ API](#)
- [AWS Secrets Manager エンドポイント](#)

Secrets Manager コンソール

ブラウザベースの [Secrets Manager コンソール](#) を使用してシークレットを管理し、コンソールではシークレットに関連するほぼすべてのタスクを実行できます。

コマンドラインツール

AWS コマンドラインツールを使用すると、システムコマンドラインでコマンドを発行して、Secrets Manager やその他の AWS タスクを実行できます。これは、コンソールを使用するよりも高速でより便利になります。コマンドラインツールは、AWS タスクを実行するスクリプトを構築する場合に便利です。

コマンドシェルにコマンドを入力すると、コマンド履歴がアクセスされたり、ユーティリティからコマンドパラメータにアクセスされたりするリスクがあります。[the section called “AWS CLI を使用して AWS Secrets Manager のシークレットを保存する際のリスクを軽減する”](#) を参照してください。

コマンドラインツールは、AWS リージョン内のサービスのデフォルトエンドポイントを自動的に使用します。API リクエストに別のエンドポイントを指定することができます。[the section called “Secrets Manager エンドポイント”](#) を参照してください。

AWS には、次の 2 セットのコマンドラインツールが用意されています。

- [AWS Command Line Interface \(AWS CLI\)](#)
- [AWS Tools for Windows PowerShell](#)

AWS SDKs

AWS SDKs は、さまざまなプログラミング言語とプラットフォーム用のライブラリとサンプルコードで構成されています。SDK には、暗号署名によるリクエスト、エラーの管理、リクエストの自動再試行などのタスクが含まれます。いずれかの SDK をダウンロードしてインストールするには、「[アマゾン ウェブ サービスのツール](#)」を参照してください。

AWS SDKs、AWS リージョン内のサービスのデフォルトエンドポイントを自動的に使用します。API リクエストに別のエンドポイントを指定することができます。[the section called “Secrets Manager エンドポイント”](#) を参照してください。

SDK ドキュメントについては、以下を参照してください。

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

HTTPS クエリ API

HTTPS クエリ API を使用すると、Secrets Manager とに [プログラムでアクセスできます](#) AWS。HTTPS クエリ API を使用すると、サービスに HTTPS リクエストを直接発行できます。

Secrets Manager HTTPS クエリ API を直接呼び出すこともできますが、いずれかの SDK を使用することをお勧めします。SDK は、本来は手動で行う必要がある多くの便利なタスクを実行します。例えば、SDK は自動的にリクエストに署名し、該当言語で構文的に適切な構造になるようレスポンスを変換します。

Secrets Manager に対し HTTPS 呼び出しを行うには、[???](#) に接続します。

AWS Secrets Manager エンドポイント

プログラムにより Secrets Manager に接続するには、エンドポイント (サービスのエン트리ポイントの URL) を使用します。Secrets Manager エンドポイントは、IPv4 と IPv6 の両方をサポートするデュアルスタックのエンドポイントです。

Secrets Manager では、[連邦情報処理規格 \(FIPS、Federal Information Processing Standard\) 140-2](#) をサポートするエンドポイントが、一部のリージョンで利用可能です。

Secrets Manager は TLS 1.2 および 1.3 をサポートしています。Secrets Manager は、中国リージョンを除くすべてのリージョンで [PQTLS](#) をサポートしています。

Note

Python AWS SDK とは IPv6 と IPv4 を順番に呼び AWS CLI 出そうとするため、IPv6 が有効になっていない場合、呼び出しがタイムアウトして IPv4 で再試行されるまでに時間がかかることがあります。この問題に対処するには、IPv6 を完全に無効にするか、[IPv6 に移行](#)します。

Secrets Manager のサービスエンドポイントは次のとおりです。この命名は、[一般的なデュアルスタックの命名規則](#)とは異なることに注意してください。

リージョン名	リージョン	エンドポイント	プロトコル
米国東部 (オハイオ)	us-east-2	secretsmanager.us-east-2.amazonaws.com	HTTPS
		secretsmanager-fips.us-east-2.amazonaws.com	HTTPS
米国東部 (バージニア北部)	us-east-1	secretsmanager.us-east-1.amazonaws.com	HTTPS
		secretsmanager-fips.us-east-1.amazonaws.com	HTTPS
米国西部 (北カリ)	us-west-1	secretsmanager.us-west-1.amazonaws.com	HTTPS

リージョン名	リージョン	エンドポイント	プロトコル
フォールニア)		secretsmanager-fips.us-west-1.amazonaws.com	HTTPS
米国西部 (オレゴン)	us-west-2	secretsmanager.us-west-2.amazonaws.com	HTTPS
		secretsmanager-fips.us-west-2.amazonaws.com	HTTPS
アフリカ (ケープタウン)	af-south-1	secretsmanager.af-south-1.amazonaws.com	HTTPS
アジアパシフィック (香港)	ap-east-1	secretsmanager.ap-east-1.amazonaws.com	HTTPS
アジアパシフィック (ハイデラバード)	ap-south-2	secretsmanager.ap-south-2.amazonaws.com	HTTPS
アジアパシフィック (ジャカルタ)	ap-southeast-3	secretsmanager.ap-southeast-3.amazonaws.com	HTTPS
アジアパシフィック (メルボルン)	ap-southeast-4	secretsmanager.ap-southeast-4.amazonaws.com	HTTPS
アジアパシフィック (ムンバイ)	ap-south-1	secretsmanager.ap-south-1.amazonaws.com	HTTPS

リージョン名	リージョン	エンドポイント	プロトコル
アジアパシフィック (大阪)	ap-northeast-3	secretsmanager.ap-northeast-3.amazonaws.com	HTTPS
アジアパシフィック (ソウル)	ap-northeast-2	secretsmanager.ap-northeast-2.amazonaws.com	HTTPS
アジアパシフィック (シンガポール)	ap-southeast-1	secretsmanager.ap-southeast-1.amazonaws.com	HTTPS
アジアパシフィック (シドニー)	ap-southeast-2	secretsmanager.ap-southeast-2.amazonaws.com	HTTPS
アジアパシフィック (東京)	ap-northeast-1	secretsmanager.ap-northeast-1.amazonaws.com	HTTPS
カナダ (中部)	ca-central-1	secretsmanager.ca-central-1.amazonaws.com	HTTPS
		secretsmanager-fips.ca-central-1.amazonaws.com	HTTPS
カナダ西部 (カルガリー)	ca-west-1	secretsmanager.ca-west-1.amazonaws.com	HTTPS
		secretsmanager-fips.ca-west-1.amazonaws.com	HTTPS
欧州 (フランクフルト)	eu-central-1	secretsmanager.eu-central-1.amazonaws.com	HTTPS

リージョン名	リージョン	エンドポイント	プロトコル
欧州 (アイルランド)	eu-west-1	secretsmanager.eu-west-1.amazonaws.com	HTTPS
欧州 (ロンドン)	eu-west-2	secretsmanager.eu-west-2.amazonaws.com	HTTPS
ヨーロッパ (ミラノ)	eu-south-1	secretsmanager.eu-south-1.amazonaws.com	HTTPS
欧州 (パリ)	eu-west-3	secretsmanager.eu-west-3.amazonaws.com	HTTPS
欧州 (スペイン)	eu-south-2	secretsmanager.eu-south-2.amazonaws.com	HTTPS
欧州 (ストックホルム)	eu-north-1	secretsmanager.eu-north-1.amazonaws.com	HTTPS
欧州 (チューリッヒ)	eu-central-2	secretsmanager.eu-central-2.amazonaws.com	HTTPS
イスラエル (テルアビブ)	il-central-1	secretsmanager.il-central-1.amazonaws.com	HTTPS
中東 (バーレーン)	me-south-1	secretsmanager.me-south-1.amazonaws.com	HTTPS
中東 (アラブ首長国連邦)	me-central-1	secretsmanager.me-central-1.amazonaws.com	HTTPS

リージョン名	リージョン	エンドポイント	プロトコル
南米 (サンパウロ)	sa-east-1	secretsmanager.sa-east-1.amazonaws.com	HTTPS
AWS GovCloud (米国東部)	us-gov-east-1	secretsmanager.us-gov-east-1.amazonaws.com	HTTPS
		secretsmanager-fips.us-gov-east-1.amazonaws.com	HTTPS
AWS GovCloud (米国西部)	us-gov-west-1	secretsmanager.us-gov-west-1.amazonaws.com	HTTPS
		secretsmanager-fips.us-gov-west-1.amazonaws.com	HTTPS

Secrets Manager シークレットの内容

Secrets Manager では、シークレットは、シークレット情報、シークレット値、およびシークレットに関するメタデータで構成されます。シークレット値には、文字列またはバイナリを使用できます。

複数の文字列値をシークレットに保存するには、キーと値のペアを使用した JSON テキスト文字列を使用することをお勧めします。次に例を示します。

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

データベースシークレットの場合、自動ローテーションを有効にするには、シークレットにデータベースの接続情報を正しい JSON 構造に含める必要があります。詳細については、「[the section called “シークレットの JSON 構造”](#)」を参照してください。

メタデータ

シークレットのメタデータには以下が含まれます。

- 次の形式の Amazon リソースネーム (ARN)

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

Secrets Manager は、シークレット ARN が確実に一意であるようにするのに役立つよう、シークレット名の末尾に 6 つのランダムな文字を含めます。元のシークレットが削除され、同じ名前で作成された場合、これらの文字により 2 つのシークレット ARN は異なったものとなります。ARN が異なるため、古いシークレットにアクセスできるユーザーであっても、新しいシークレットへのアクセスを自動的に取得するわけではありません。

- シークレットの名前、説明、リソースポリシー、タグ
- 暗号化キーの ARN。Secrets Manager AWS KMS key がシークレット値を暗号化および復号するために使用する です。Secrets Manager はシークレットテキストを常に暗号化された形式で保存

し、転送中のシークレットを暗号化します。[the section called “シークレット暗号化と復号”](#) を参照してください。

- シークレットをローテーションする方法に関する情報 (ローテーションを設定した場合) [シークレットのローテーション](#) を参照してください。

Secrets Manager は、IAM アクセス許可ポリシーを使用して、認証されたユーザーのみがシークレットにアクセスまたは変更できるようにします。[の認証とアクセスコントロール AWS Secrets Manager](#) を参照してください。

シークレットには、暗号化されたシークレット値のコピーを保持している複数のバージョンがあります。シークレットの値を変更するか、シークレットをローテーションすると、Secrets Manager は新しいバージョンを作成します。[the section called “シークレットバージョン”](#) を参照してください。

シークレットをレプリケート AWS リージョン することで、複数の でシークレットを使用できます。シークレットをレプリケートする場合、元のシークレットをコピーするか、レプリカシークレットと呼ばれるプライマリシークレットを作成します。レプリカシークレットは、プライマリシークレットにリンクされたままになっています。「[リージョン間でシークレットをレプリケートする](#)」を参照してください

「[シークレットの作成と管理](#)」を参照してください

シークレットバージョン

シークレットには、暗号化されたシークレット値のコピーを保持している複数のバージョンがあります。シークレットの値を変更するか、シークレットをローテーションすると、Secrets Manager は新しいバージョンを作成します。

Secrets Manager は、シークレットの履歴をバージョン順には保存しません。代わりに、以下の 3 つの特定のバージョンにラベルを付け、それらを追跡します。

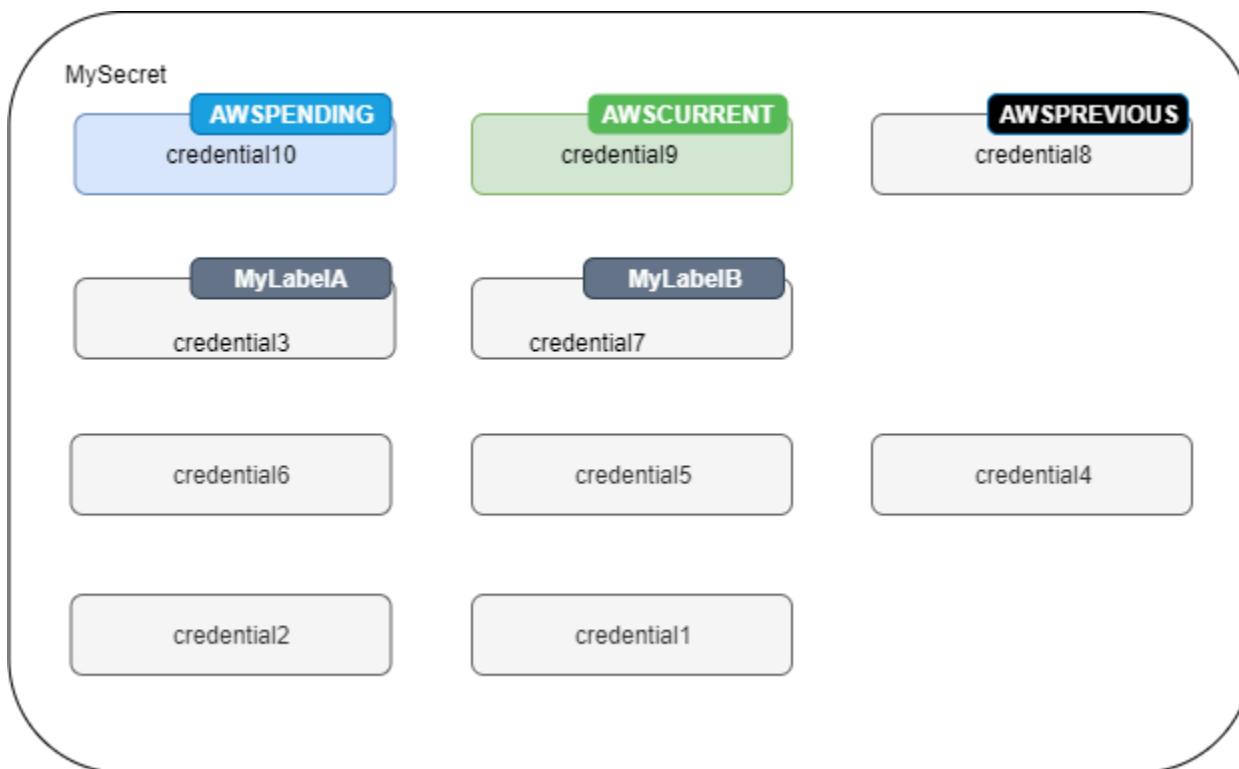
- 現在のバージョン – AWSCURRENT
- 以前のバージョン – AWSPREVIOUS
- 保留中のバージョン (ローテーション中) – AWSPENDING

シークレットには常に、AWSCURRENT というラベルが付いたバージョンがあり、シークレット値を取得する際は、そのバージョンがデフォルトで Secrets Manager から返されます。

[update-secret-version-stage](#) を呼び出すことで、独自のラベルでバージョンにラベルを付けることもできます AWS CLI。シークレットには、最大 20 個のラベルをアタッチできます。シークレットの 2 つのバージョンで同じステージングラベルを持つことはできません。各バージョンには、複数のラベル付けることが可能です。

ラベル付きのバージョンであれば、Secrets Manager により削除されることはありませんが、ラベルのないバージョンは非推奨と見なされます。非推奨のバージョンの数が 100 を超えた場合、Secrets Manager はそれらを削除します。ただし、Secrets Manager は 24 時間以内に作成されたバージョンは削除しません。

次の図は、AWS ラベル付きバージョンとカスタマーラベル付きバージョンのシークレットを示しています。ラベルのないバージョンは非推奨と見なされ、将来のある時点で Secrets Manager によって削除されます。



AWS Secrets Manager のチュートリアル

トピック

- [Amazon CodeGuru Reviewer を使って、コードの中の保護されていないシークレットを見つける](#)
- [ハードコードされたシークレットを に移動する AWS Secrets Manager](#)
- [ハードコードされたデータベース認証情報を に移動 AWS Secrets Manager](#)
- [AWS Secrets Manager に交代ユーザーローテーションを設定する](#)
- [AWS Secrets Manager のシングルユーザーローテーションを設定する](#)

Amazon CodeGuru Reviewer を使って、コードの中の保護されていないシークレットを見つける

Amazon CodeGuru Reviewer は、プログラム解析と機械学習により、開発者が見つけにくい潜在的な不具合を検出し、Java や Python のコードの改善案を提示するサービスです。CodeGuru Reviewer は Secrets Manager と統合し、コードにある保護されていないシークレットを見つけます。検出できるシークレットの種類については、「Amazon CodeGuru Reviewer User Guide」(Amazon CodeGuru Reviewer ユーザーガイド) の「[Types of secrets detected by CodeGuru Reviewer](#)」(CodeGuru Reviewer で検出されるシークレットの種類) を参照してください。

ハードコードされたシークレットを見つけたら、それを置き換えるためにアクションを起こします。

- [the section called “ハードコードされたデータベース認証情報の置き換え ”](#)
- [the section called “ハードコードされたシークレットの置き換え ”](#)

ハードコードされたシークレットを に移動する AWS Secrets Manager

コード内に平文のシークレットがある場合は、ローテーションしてシークレットマネージャーに保存することをお勧めします。Secrets Managerへの移行により、コードがSecrets Managerから直接シークレットを取得するため、コードを見た人が誰でもシークレットを見ることができるといった問題は解決されます。シークレットをローテーションすると、現在ハードコードされているシークレットが無効になります。

データベース認証のシークレットについては、[ハードコードされたデータベース認証情報をに移動 AWS Secrets Manager](#) を参照してください。

開始する前に、シークレットへのアクセスが必要なユーザーを決める必要があります。シークレットへのアクセス権限を管理するために、2つの IAM ロールを使用することをお勧めします。

- 組織のシークレットを管理するロール。詳細については、「[the section called “Secrets Manager 管理者のアクセス許可”](#)」を参照してください。このロールを使用してシークレットを作成し、ローテーションします。
- 実行時にシークレットを使用できるロール。例えば、このチュートリアルでは `RoleToRetrieveSecretAtRuntime` を使用します。コードは、このロールを担ってシークレットを取得します。このチュートリアルでは、ロールに1つのシークレットの値を取得する権限のみを与え、シークレットの値のリソース・ポリシーを使用して権限を付与しています。代替手段については、[the section called “次のステップ”](#) を参照してください。

ステップ:

- [ステップ 1: シークレットを作成する](#)
- [ステップ 2: コードを更新する](#)
- [ステップ 3: シークレットを更新する](#)
- [次のステップ](#)

ステップ 1: シークレットを作成する

最初のステップは、既存のハードコードされたシークレットを、Secrets Manager にコピーすることです。シークレットが AWS リソースに関連付けられている場合は、リソースと同じリージョンに保存します。それ以外の場合は、ユースケースに応じて最もレイテンシーの低いリージョンに保存します。

シークレットを作成するには (コンソール)

1. Secrets Manager のコンソール (<https://console.aws.amazon.com/secretsmanager/>) を開きます。
2. [Store a new secret] (新しいシークレットを保存する) を選択します。
3. [Choose secret type] (シークレットタイプを選択する) ページで、次の操作を行います。

- a. [Secret type] (シークレットタイプ) で、[Other type of secret] (他の種類のシークレット) を選択します。
- b. シークレットのキーとバリューのペア、または平文で入力してください。例:

API キーのキーと値のペア:

ClientID: *client_id*

ClientSecret : *wJalrXUtnFEMI /K7MDENG/bPxrFiCYEXAMPLEKEY*

認証情報キーと値のペア:

Username: *Saanvis*

Password: *# - #####*

OAuth トークンの平文:

AKIAI44QH8DHB#

デジタル証明書平文:

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

平文のプライベートキー

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

- c. [Secrets Manager] (暗号化キー) で `aws/secretsmanager` を選択すると、Secrets Manager に AWS マネージドキー マネージドキーを使用します。このキーを使用してもコストは発生しません。独自のカスタマーマネージドキーを使用することもできます (例えば、[別の AWS アカウントアカウントからシークレットにアクセスする場合など](#))。カスタマーマネージドキーの使用料金の詳細については、「[料金](#)」を参照してください。
- d. [次へ] をクリックします。

4. [Choose secret type] (シークレットタイプを選択する) ページで、次の操作を行います。
 - a. わかりやすいシークレット名と説明を入力します。
 - b. [Resource permissions] (リソースのアクセス許可) で、[Edit permissions] (アクセス許可の編集) を選択します。シークレットの取得を許可する次のポリシー `RoleToRetrieveSecretAtRuntime` を貼り付け、保存 を選択します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::AccountId:role/RoleToRetrieveSecretAtRuntime"
      },
      "Action": "secretsmanager:GetSecretValue",
      "Resource": "*"
    }
  ]
}
```

- c. ページの最下部にある [Next] (次へ) を選択します。
5. [Configure rotation] (ローテーションを設定する) ページで、回転をオフにしておきます。[Next] を選択します。
6. [Review] (レビュー) ページで、シークレットの詳細を確認し、[Store] (保存) を選択します。

ステップ 2: コードを更新する

シークレットを取得 `RoleToRetrieveSecretAtRuntime` できるようにするには、コードが IAM ロールを引き受ける必要があります。詳細については、[「IAM ロールへの切り替え \(AWS API\)」](#) を参照してください。

次に、Secrets Manager が提供するサンプルコードを使用して、Secrets Manager からシークレットを取得するようにコードを更新します。

サンプルコードを見るには

1. Secrets Manager コンソール (<https://console.aws.amazon.com/secretsmanager/>) を開きます。
2. [Secrets] (シークレット) ページで、自分のシークレットを選択します。

3. [サンプルコード] を下にスクロールします。プログラミング言語を選択し、コードスニペットをコピーします。

アプリケーションで、ハードコードされたシークレットを削除し、コードスニペットを貼り付けます。コード言語によっては、スニペットの中に関数やメソッドの呼び出しを追加する必要があります。

ハードコードされたシークレットを使用して、アプリケーションが期待通りに動作することをテストしてください。

ステップ 3: シークレットを更新する

最後のステップは、ハードコードされたシークレットを失効させ、更新することです。シークレットの発行元を参照し、シークレットの取り消しや更新の手順を確認します。例えば、現在のシークレットを無効にして、新しいシークレットを生成する必要がある場合があります。

シークレットを新しい値で更新するには

1. Secrets Manager のコンソール (<https://console.aws.amazon.com/secretsmanager/>) を開きます。
2. [Secrets] (シークレット) で、シークレットを選択します。
3. [Secret details] (シークレットの詳細) ページでスクロールダウンし、[Rotation configuration] (ローテーション設定) セクションの [Edit rotation] (ローテーションの編集) を選択します。
4. シークレットを更新し、[Save] (保存) を選択します。

次に、新しいシークレットを使用して、アプリケーションが期待どおりに動作するかどうかテストします。

次のステップ

コードからハードコードされたシークレットを削除した後、次に検討すべきいくつかの項目を挙げます。

- Java および Python アプリケーションでハードコードされたシークレットを見つけるには、[Amazon CodeGuru Reviewer](#) をお勧めします。
- シークレットをキャッシュすることで、パフォーマンスを向上させ、コストを削減することができます。詳細については、「[シークレットを取得する](#)」を参照してください。

- 複数のリージョンからアクセスするシークレットについては、レイテンシーを改善するためにシークレットをレプリケーションすることを検討してください。詳細については、「[リージョン間でシークレットをレプリケートする](#)」を参照してください。
- このチュートリアルでは、シークレット値を取得するアクセス許可 `RoleToRetrieveSecretAtRuntime` のみを付与しました。シークレットに関するメタデータの取得やシークレットの一覧の表示など、より多くの権限をロールに付与するには、[the section called “アクセス許可ポリシーの例”](#) を参照してください。
- このチュートリアルでは、シークレットのリソースポリシー `RoleToRetrieveSecretAtRuntime` を使用して にアクセス許可を付与しました。権限を付与するその他の方法については、「[the section called “アクセス許可ポリシーをアイデンティティにアタッチする”](#)」を参照してください。

ハードコードされたデータベース認証情報をに移動 AWS Secrets Manager

コード内に平文のデータベース認証情報がある場合は、認証情報を Secrets Manager に移動して、すぐにローテートすることをお勧めします。認証情報を Secrets Manager に移動することで、コードを見た人に認証情報が見えるという問題が解決されます。なぜなら、今後、あなたのコードは Secrets Manager から直接認証情報を取得するからです。シークレットを回転させると、パスワードが更新され、現在のハードコードされたパスワードが無効となるように取り消されます。

Amazon RDS、Amazon Redshift、および Amazon DocumentDB データベースの場合、このページの手順を使用してハードコードされた認証情報を Secrets Manager に移動させることができます。他のタイプの認証情報や他のシークレットについては、「[the section called “ハードコードされたシークレットの置き換え”](#)」を参照してください。

開始する前に、シークレットへのアクセスが必要なユーザーを決める必要があります。シークレットへのアクセス権限を管理するために、2つの IAM ロールを使用することをお勧めします。

- 組織のシークレットを管理するロール。詳細については、「[the section called “Secrets Manager 管理者のアクセス許可”](#)」を参照してください。このロールを使用してシークレットを作成し、ローテーションします。
- `RoleToRetrieveSecretAtRuntime` このチュートリアルでは、実行時に認証情報を使用できるロールです。コードは、このロールを担ってシークレットを取得します。

ステップ:

- [ステップ 1: シークレットを作成する](#)
- [ステップ 2: コードを更新する](#)
- [ステップ 3: シークレットをローテーションする](#)
- [次のステップ](#)

ステップ 1: シークレットを作成する

最初のステップは、既存のハードコードされた認証情報を、Secrets Manager でシークレットにコピーすることです。レイテンシーを最低限に抑えるために、シークレットをデータベースと同じリージョンに保存します。

シークレットを作成する

1. Secrets Manager のコンソール (<https://console.aws.amazon.com/secretsmanager/>) を開きます。
2. [Store a new secret] (新しいシークレットを保存する) を選択します。
3. [Choose secret type] (シークレットタイプを選択する) ページで、次の操作を行います。
 - a. [Secret type] (シークレットタイプ) で、保存するデータベース認証情報のタイプを選択します。
 - Amazon RDS データベース
 - Amazon DocumentDB データベース
 - Amazon Redshift データウェアハウス。
 - その他のタイプのシークレットについては、「[ハードコードされたシークレットを置き換える](#)」を参照してください。
 - b. 認証情報を使用する場合、データベースの既存のハードコードされた認証情報を入力します。
 - c. [Secrets Manager] (暗号化キー) で aws/secretsmanager を選択すると、Secrets Manager に AWS マネージドキー マネージドキーを使用します。このキーを使用してもコストは発生しません。独自のカスターマネージドキーを使用することもできます (例えば、[別の AWS アカウントアカウントからシークレットにアクセスする場合など](#))。カスターマネージドキーの使用料金の詳細については、「[料金](#)」を参照してください。
 - d. [Database] (データベース) で、データベースを選択します。
 - e. [次へ] を選択します。

4. [Configure secret] (シークレットを設定する) ページで、次の操作を行います。
 - a. わかりやすいシークレット名と説明を入力します。
 - b. [Resource permissions] (リソースのアクセス許可) で、[Edit permissions] (アクセス許可の編集) を選択します。 *RoleToRetrieveSecretAtRuntime* シークレットの取得を許可する次のポリシーを貼り付けて、[Save] を選択します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::AccountId:role/RoleToRetrieveSecretAtRuntime"
      },
      "Action": "secretsmanager:GetSecretValue",
      "Resource": "*"
    }
  ]
}
```

- c. ページの最下部にある [Next] (次へ) を選択します。
5. リポジトリの [Configure rotation] (ローテーションを設定する) ページでは、今のところローテーションはオフにしておきます。後でオンにします。[Next] (次へ) を選択します。
6. [Review] (レビュー) ページで、シークレットの詳細を確認し、[Store] (保存) を選択します。

ステップ 2: コードを更新する

シークレットを取得するには、コードが IAM *RoleToRetrieveSecretAtRuntime* ロールを引き継ぐ必要があります。詳細については、「[IAM ロール \(AWS API\) への切り替え](#)」を参照してください。

次に、Secrets Manager が提供するサンプルコードを使用して、Secrets Manager からシークレットを取得するようにコードを更新します。

サンプルコードを見るには

1. Secrets Manager コンソール (<https://console.aws.amazon.com/secretsmanager/>) を開きます。
2. [Secrets] (シークレット) ページで、自分のシークレットを選択します。

3. [サンプルコード] を下にスクロールします。言語を選択し、コードスニペットをコピーします。

アプリケーションで、ハードコードされた認証情報を削除し、コードスニペットを貼り付けます。コード言語によっては、スニペットの中に関数やメソッドの呼び出しを追加する必要があります。

ハードコードされた認証情報の代わりにシークレットを使用して、アプリケーションが期待通りに動作することをテストしてください。

ステップ 3: シークレットをローテーションする

最後のステップは、シークレットのローテーションによってハードコードされた認証情報を失効させることです。ローテーションとは、シークレットを定期的に更新するためのプロセスのことです。シークレットのローテーションを行うと、シークレットとデータベースの両方で認証情報が更新されます。Secrets Manager は、設定したスケジュールで自動的にシークレットをローテーションすることができます。

ローテーションの設定の一部として、Lambda ローテーション関数が Secrets Manager とデータベースの両方にアクセスできることを確認する必要があります。自動ローテーションをオンにすると、Secrets Manager は Lambda ローテーション機能をデータベースと同じ VPC に作成し、データベースへのネットワークアクセスができるようにします。Lambda ローテーション関数は、Secrets Manager を呼び出してシークレットを更新することも可能でなければなりません。Lambda から Secrets Manager への呼び出しがインフラストラクチャから離れないように、VPC に Secrets Manager エンドポイントを作成することをお勧めします。AWS 手順については、「[VPC エンドポイント](#)」を参照してください。

ローテーションを有効にするには

1. Secrets Manager コンソール (<https://console.aws.amazon.com/secretsmanager/>) を開きます。
2. [Secrets] (シークレット) ページで、自分のシークレットを選択します。
3. シークレットの詳細ページで、[Rotation configuration] (ローテーション設定) セクションの [Edit rotation] (ローテーションの編集) を選択します。
4. [Edit rotation configuration] (ローテーション設定の編集) ダイアログボックスで、次の操作を行います。
 - a. [Automatic rotation] (自動ローテーション) を有効化します。
 - b. [ローテーションのスケジュール] で、UTC タイムゾーンでスケジュールを入力します。

- c. シークレットを保存したときにシークレットをローテーションするには、[Rotate immediately when the secret is stored] (シークレットが保存されたときにすぐにローテーションする) を選択します。
- d. ローテーション関数で、[Create a new Lambda function] (新しい Lambda 関数の作成) を選択し、新しい関数の名前を入力します。Secrets Manager は、関数名の先頭に「SecretsManager」を追加します。
- e. [ローテーション戦略] で、[単一ユーザー] を選択します。
- f. [保存] を選択します。

シークレットがローテーションしたことを確認するには

1. Secrets Manager のコンソール (<https://console.aws.amazon.com/secretsmanager/>) を開きます。
2. [Secrets] (シークレット) で、シークレットを選択します。
3. [Secret details] (シークレットの詳細) ページで、下へスクロールし、[Retrieve secret value] (シークレットの値を取得する) を選択します。

シークレットの値が変更された場合、ローテーションは成功したことになります。シークレットの値が変わらない場合は、[におけるローテーションのトラブルシューティング](#) CloudWatch ローテーション関数のログを確認する必要があります。

ローテーションされたシークレットを使用して、アプリケーションが期待どおりに動作するかどうかテストします。

次のステップ

コードからハードコードされたシークレットを削除した後、次に検討すべきいくつかの項目を挙げます。

- シークレットをキャッシュすることで、パフォーマンスを向上させ、コストを削減することができます。詳細については、「[シークレットを取得する](#)」を参照してください。
- 別のローテーションスケジュールを選択できます。詳細については、「[the section called “ローテーションスケジュール”](#)」を参照してください。
- Java および Python アプリケーション内のハードコードされたシークレットを見つけるには、[Amazon CodeGuru](#) Reviewer をお勧めします。

AWS Secrets Manager に交代ユーザーローテーションを設定する

このチュートリアルでは、データベース認証情報を含むシークレットに対して、交代ユーザーローテーションを設定する方法について学びます。交代ユーザーローテーションとは、Secrets Manager がユーザーのクローンを作成し、ユーザーの認証情報を交代で更新するローテーション戦略です。この戦略は、シークレットの高可用性が必要な場合に適しています。これは、代替ユーザーの 1 人がデータベースへの現在の認証情報を保持し、もう 1 人が更新されているためです。詳細については、「[the section called “交代ユーザー”](#)」を参照してください。

交代ユーザーローテーションを設定するには、次の 2 つのシークレットが必要です。

- ローテーションさせたい認証情報を持つ 1 つのシークレット。
- 管理者認証情報を持つ 2 番目のシークレット。

このユーザーには、最初のユーザーのクローンを作成し、最初のユーザーのパスワードを変更する権限があります。このチュートリアルでは、Amazon RDS で管理者ユーザー用にこのシークレットを作成します。Amazon RDS は、管理者パスワードのローテーションも管理します。詳細については、「[the section called “マネージドローテーション”](#)」を参照してください。

このチュートリアルの最初の部分では、現実的な環境をセットアップします。ローテーションの仕組みをご覧ください。このチュートリアルでは Amazon RDS MySQL データベースの例を使用します。セキュリティのため、データベースはインバウンドのインターネットアクセスを制限する VPC 内にあります。お使いのローカルコンピュータからインターネット経由でこのデータベースに接続するときは、このデータベースに接続できるが、インターネット経由での SSH 接続も可能な VPC 内のサーバー、踏み台ホストを使用します。このチュートリアルで使用する踏み台ホストは Amazon EC2 インスタンスであり、このインスタンスのセキュリティグループでは、他のタイプの接続は禁止されています。

チュートリアルを終了したら、チュートリアルからリソースをクリーンアップすることをお勧めします。本番環境では使用しないでください。

Secrets Manager のローテーションでは、シークレットとデータベースの更新に AWS Lambda 関数を使用されます。Lambda 関数を使用する場合のコストについては、「[料金](#)」を参照してください。

チュートリアル:

- [許可](#)
- [前提条件](#)
- [ステップ 1: Amazon RDS データベースユーザーを作成する](#)

- [ステップ 2: ユーザーの認証情報用のシークレットを作成する](#)
- [ステップ 3: ローテーションされたシークレットをテストする](#)
- [ステップ 4: リソースをクリーンアップする](#)
- [次のステップ](#)

許可

このチュートリアルの前提条件として、AWS アカウント の管理者権限が必要となります。本番稼働環境では、各ステップで異なるロールを使用するのがベストプラクティスとなります。例えば、Amazon RDS データベースの作成にはデータベース管理者権限を持つロールを使用し、VPC とセキュリティグループの設定にはネットワーク管理者権限を持つロールを使用します。チュートリアルの各ステップでは同じアイデンティティを使用することが推奨されます。

本番稼働環境で許可を設定する方法については、「[認証とアクセスコントロール](#)」を参照してください。

前提条件

このチュートリアルでは、以下が必要になります。

- [前提条件 A: Amazon VPC](#)
- [前提条件 B: Amazon EC2 インスタンス](#)
- [前提条件 C: Amazon RDS データベースと、管理者認証情報の Secrets Manager シークレット](#)
- [前提条件 D: ローカル コンピューターが EC2 インスタンスに接続できるようにする](#)

前提条件 A: Amazon VPC

このステップでは、Amazon RDS データベースと Amazon EC2 インスタンスを起動できる VPC を作成します。後のステップで、コンピューターを使用してインターネット経由で踏み台に接続し、データベースに接続するため、VPC からのトラフィックを許可する必要があります。これを行うために、Amazon VPC はインターネットゲートウェイを VPC にアタッチし、ルートテーブルにルートを追加して、VPC の外部に向かうトラフィックがインターネットゲートウェイに送信されるようにします。

VPC 内で、Secrets Manager エンドポイントと Amazon RDS エンドポイントを作成します。後のステップで自動ローテーションを設定すると、Secrets Manager は VPC 内に Lambda ローテーション関数を作成して、データベースにアクセスできるようにします。また、Lambda ローテーション関

数は Secrets Manager を呼び出してシークレットを更新し、Amazon RDS を呼び出してデータベース接続情報を取得します。VPC 内にエンドポイントを作成することで、Lambda 関数から Secrets Manager および Amazon RDS への呼び出しが AWS インフラストラクチャを離れないようにします。代わりに、VPC 内のエンドポイントにルーティングされます。

VPC を作成するには

1. Amazon VPC コンソール (<https://console.aws.amazon.com/vpc/>) を開きます。
2. [VPC の作成] を選択します。
3. [VPC の作成] ページで、[VPC など] を選択します。
4. [名前タグの自動生成] で、[自動生成] に **SecretsManagerTutorial** と入力します。
5. DNS オプションでは、**Enable DNS hostnames** と **Enable DNS resolution** の両方を選択します。
6. [VPC の作成] を選択します。

VPC 内に Secrets Manager エンドポイントを作成するには

1. Amazon VPC コンソールの [エンドポイント] 下で、[エンドポイントの作成] を選択します。
2. [Endpoint Settings] (エンドポイントの設定) の [Name] (名前) に **SecretsManagerTutorialEndpoint** と入力します。
3. [Services] (サービス) で **secretsmanager** と入力して、リストをフィルタリングし、AWS リージョンで Secrets Manager エンドポイントを選択します。米国東部 (バージニア北部) であれば、**com.amazonaws.us-east-1.secretsmanager** を選択します。
4. [VPC] で、**vpc**** (SecretsManagerTutorial)** を選択します。
5. [Subnets] (サブネット) で、すべての [Availability Zones] (アベイラビリティゾーン) を選択し、各アベイラビリティゾーンに [Subnet ID] (サブネット ID) を挿入します。
6. [IP address type] (IP アドレスタイプ) で、**IPv4** を選択します。
7. [Security groups] (セキュリティグループ) で、デフォルトのセキュリティグループを選択します。
8. [Policy type] (ポリシーの種類) で、**Full access** を選択します。
9. [Create endpoint] (エンドポイントの作成) を選択します。

VPC 内に Amazon RDS エンドポイントを作成するには

1. Amazon VPC コンソールの [エンドポイント] 下で、[エンドポイントの作成] を選択します。
2. [Endpoint Settings] (エンドポイントの設定) の [Name] (名前) に **RDS Tutorial Endpoint** と入力します。
3. [Services] (サービス) で、**rds** と入力してリストをフィルタリングし、AWS リージョンで Amazon RDS エンドポイントを選択します。米国東部 (バージニア北部) であれば、**com.amazonaws.us-east-1.rds** を選択します。
4. [VPC] で、**vpc**** (SecretsManagerTutorial)** を選択します。
5. [Subnets] (サブネット) で、すべての [Availability Zones] (アベイラビリティゾーン) を選択し、各アベイラビリティゾーンに [Subnet ID] (サブネット ID) を挿入します。
6. [IP address type] (IP アドレスタイプ) で、**IPv4** を選択します。
7. [Security groups] (セキュリティグループ) で、デフォルトのセキュリティグループを選択します。
8. [Policy type] (ポリシーの種類) で、**Full access** を選択します。
9. [Create endpoint] (エンドポイントの作成) を選択します。

前提条件 B: Amazon EC2 インスタンス

後のステップで作成する Amazon RDS データベースは VPC にあるため、それにアクセスするには踏み台ホストが必要です。踏み台ホストも VPC 内にありますが、後のステップで、セキュリティグループを構成して、ローカルコンピューターが SSH で踏み台ホストに接続できるようにします。

踏み台ホストの EC2 インスタンスを作成するには

1. Amazon EC2 コンソール (<https://console.aws.amazon.com/ec2/>) を開きます。
2. [Instances] (インスタンス) を選択し、[Launch instance] (インスタンスの作成) を選択します。
3. [Name and tags] (名前とタグ) の、[Name] (名前) で、**SecretsManagerTutorialInstance** と入力します。
4. [アプリケーションと OS イメージ] で、デフォルトの **Amazon Linux 2 AMI (HVM) Kernel 5.10** をそのまま使用します。
5. [インスタンス タイプ] で、デフォルトの **t2.micro** をそのまま使用します。
6. [Key pair] (キーペア) で [Create key pair] (キーペアの作成) を選択します。

[キーペア作成] ダイアログボックスの [キーペア名] に **SecretsManagerTutorialKeyPair** を入力し、[作成] を選択します。

キーペアは自動的にダウンロードされます。

7. [Network settings] (ネットワーク設定) で、[Edit] (編集) を選択し、次の操作を実行します。
 - a. [VPC] で、**vpc-**** SecretsManagerTutorial** を選択します。
 - b. [Auto-assign Public IP] (自動割り当てパブリック IP) で、**Enable** を選択します。
 - c. [Firewall] (ファイアウォール) の既存のセキュリティグループを選択します。
 - d. [Common security groups] (共通のセキュリティグループ) で、**default** を選択します。
8. Launch instance (インスタンスの起動) を選択します。

前提条件 C: Amazon RDS データベースと、管理者認証情報の Secrets Manager シークレット

このステップでは、Amazon RDS MySQL データベースを作成し、Amazon RDS が管理者認証情報を含むシークレットを作成するように設定します。次に、Amazon RDS が管理者シークレットのローテーションを自動的に管理します。詳細については、「[マネージドローテーション](#)」を参照してください。

データベース作成の一環として、前のステップで作成した踏み台ホストを指定します。次に、Amazon RDS は、データベースとインスタンスが相互にアクセスできるようにセキュリティグループを設定します。インスタンスにアタッチされたセキュリティグループにルールを追加して、ローカルコンピューターもインスタンスに接続できるようにします。

管理者認証情報を含む Secrets Manager シークレットを使用して Amazon RDS データベースを作成するには

1. Amazon RDS コンソールで、[Databases] (データベース) を選択します。
2. [Engine options] (エンジンオプション) セクションで、エンジンタイプとして **MySQL** を選択します。
3. [Templates] (テンプレート) セクションで、**Free tier** を選択します。
4. 設定 セクションで、以下の手順を実行します。
 - a. [DB instance identifier] (DB インスタンス識別子) に **SecretsManagerTutorial** と入力します。

- b. [認証情報の設定] で、[AWS Secrets Manager でマスター資格情報を管理する] を選択します。
5. [Connect] (接続) セクションの [Computer resource] (コンピューターリソース) で、[EC2 computer resource] を選択し、[EC2 インスタンス] (EC2 コンピュータリソースに接続) で **SecretsManagerTutorialInstance** を選択します。
6. [データベースの作成] を選択します。

前提条件 D: ローカル コンピューターが EC2 インスタンスに接続できるようにする

このステップでは、前提条件 B で作成した EC2 インスタンスを構成して、ローカルコンピューターが EC2 インスタンスに接続できるようにします。これを行うには、Amazon RDS が Prereq C に追加したセキュリティグループを編集して、コンピューターの IP アドレスが SSH に接続できるようにするルールを含めます。このルールにより、ローカルコンピューター (現在の IP アドレスで識別される) は、インターネット経由で SSH を使用して踏み台ホストに接続できます。

ローカルコンピューターが EC2 インスタンスに接続できるようにするには

1. Amazon EC2 コンソール (<https://console.aws.amazon.com/ec2/>) を開きます。
2. EC2 インスタンスの [SecretsManager TutorialInstance] (セキュリティマネージャー チュートリアルインスタンス) の [Security] (セキュリティ) タブの [Security groups] (セキュリティグループ) で、**sg-*** (ec2-rds-X)** を選択します。
3. [Input rules] (インプットルール) で、[Edit inbound rules] (インバウンドルールを編集) を選択します。
4. [Add rule] (ルールを追加) を選択し、次の操作を行います。
 - a. [Type] (タイプ) で、**SSH** を選択します。
 - b. [Source type] (ソースタイプ) で **My IP** を選択します。

ステップ 1: Amazon RDS データベースユーザーを作成する

まず、シークレットにその認証情報を保存するユーザーが必要になります。ユーザーを作成するには、管理者の認証情報を使用して Amazon RDS データベースにログインします。簡単にするために、このチュートリアルでは、データベースへの完全なアクセス許可を持つユーザーを作成します。運用環境では、これは一般的ではないため、最小権限の原則に従うことをお勧めします。

データベースに接続するには、MySQL クライアントツールを使用します。このチュートリアルでは、GUI ベースのアプリケーション、MySQL Workbench を使用します。MySQL Workbench のインストール方法については、[「Download MySQL Workbench」](#) を参照してください。

データベースに接続するには、MySQL Workbench で接続構成を作成します。設定には、Amazon EC2 と Amazon RDS の両方からの情報が必要です。

MySQL Workbench でデータベースの接続を設定するには

1. MySQL Workbench で、[MySQL Connections] (MySQL 接続) の横にある (+) ボタンをクリックします。
2. [Setup New Connection] (新しい接続の設定) ダイアログボックスで、次を実行します。
 - a. [Connection Name] (接続名) に、**SecretsManagerTutorial** と入力します。
 - b. [Connection method] (接続方法) で、**Standard TCP/IP over SSH** を選択します。
 - c. [Parameters] (パラメータ) タブで、次を実行します。

- i. [SSH Hostname] (SSH ホスト名) に、Amazon EC2 インスタンスのパブリック IP アドレスを入力します。

この IP アドレスは、インスタンス [SecretsManagerTutorialInstance] を選択すると、Amazon EC2 コンソールに表示されます。[Public IPv4 DNS] の下に表示された IP アドレスをコピーします。

- ii. [SSH Username] (SSH ユーザー名) に、**ec2-user** と入力します。
- iii. [SSH Keyfile] (SSH キーファイル) で、前回の前提条件でダウンロードしたキーペアファイル [SecretsManagerTutorialKeyPair.pem] を選択します。
- iv. [MySQL Hostname] (MySQL ホスト名) に、Amazon RDS エンドポイントアドレスを入力します。

このエンドポイントアドレスは、データベースインスタンス [secretsmanagertutorialdb] を選択すると、Amazon RDS コンソールに表示されます。[Endpoint] (エンドポイント) の下に表示されたアドレスをコピーします。

- v. [Username] (ユーザー名) に、**admin** と入力します。
- d. [OK] をクリックします。

管理者パスワードを取得するには

1. Amazon RDS コンソールで、データベースに移動します。
2. [Configuration] (設定) タブの [Master Credentials ARN] (マスター認証情報 ARN) で、[Manage in Secrets Manager] (Secrets Managerの管理) を選択します。

Secrets Manager コンソールが開きます。

3. シークレットの詳細ページで、[Retrieve secret value] (シークレット値の取得) を選択します。
4. パスワードは [Secret value] (シークレット値) セクションに表示されます。

データベースユーザーを作成するには

1. MySQL Workbench で、接続 [SecretsManagerTutorial] を選択します。
2. シークレットから取得した管理者パスワードを入力します。
3. MySQL Workbenchの[Query] (クエリ) ウィンドウで、次のコマンド (強力なパスワードを含む) を入力し、[Execute] (実行) を選択します。

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

[Output] (出力) ウィンドウに、コマンドが実行されたことが表示されます。

ステップ 2: ユーザーの認証情報用のシークレットを作成する

次に、いま作成したユーザーの認証情報を保存するための、シークレットを作成します。これが、後ほどローテーションするシークレットです。自動ローテーションをオンにし、交代ユーザー戦略を指定するために、1人目のユーザーのパスワードを変更する権限を持った、別のスーパーユーザーのシークレットを選択します。

1. Secrets Manager のコンソール (<https://console.aws.amazon.com/secretsmanager/>) を開きます。
2. [Store a new secret] (新しいシークレットを保存する) を選択します。
3. [Choose secret type] (シークレットタイプを選択する) ページで、次の操作を行います。
 - a. [Secret type] (シークレットタイプ) で、[Credentials for Amazon RDS database] (Amazon RDS データベースの認証情報) を選択します。

- b. [Credentials] (認証情報) で、ユーザーネーム **appuser** と、MySQL Workbench を使って作成したデータベースユーザー向けに入力したパスワードを入力します。
 - c. [Database] (データベース) で、**secretsmanagertutorialdb** を選択します。
 - d. [Next] (次へ) をクリックします。
4. [Configure secret] (シークレットを設定する) ページで、[Secret name] (シークレット名) に **SecretsManagerTutorialAppuser** と入力し、[Next] (次) を選択します。
 5. [Configure Routing] (ローテーションを設定する) ページで、以下を実行します。
 - a. [Automatic rotation] (自動ローテーション) を有効化します。
 - b. [Rotation schedule] (ローテーションスケジュール) で、[Days] (日数) のスケジュールを設定します: [Duration] (期間) の 2 日間: **2h**。[Rotate immediately] (すぐにローテーションする) は選択したままにしておきます。
 - c. [Rotation function] (ローテーション関数) で [Create a rotation function] (ローテーション関数を作成) を選択し、[Function Name] (関数名) に **tutorial-alternating-users-rotation** と入力します。
 - d. [ローテーション戦略] で [代替ユーザー] を選択し、[管理者認証情報シークレット] で **rds!cluster...** という名前のシークレットを選択します。これには、このチュートリアル **secretsmanagertutorial** で作成したデータベースの名前を含む [説明] が含まれています (例: Secret associated with primary RDS DB instance: `arn:aws:rds:Region:AccountId:db:secretsmanagertutorial`)。
 - e. [Next] (次へ) をクリックします。
 6. [Review] (確認) ページで [Store] (保存) を選択します。

Secrets Manager はシークレットの詳細ページに戻ります。ローテーション設定のステータスは、ページの上で確認できます。Secrets Manager では、Lambda ローテーション関数や Lambda 関数を実行する実行ロールなどのリソースを作成するときは CloudFormation を使用します。CloudFormation が終了すると、バナーがローテーションが予定されているシークレットに切り替わります。これで、最初のローテーションは完了です。

ステップ 3: ローテーションされたシークレットをテストする

シークレットがローテーションされたので、シークレットに有効な新しい認証情報が含まれていることを確認できます。シークレット内のパスワードが、元の認証情報から変更されました。

シークレットから新しいパスワードを取得するには

1. Secrets Manager のコンソール (<https://console.aws.amazon.com/secretsmanager/>) を開きます。
2. [Secrets] (シークレット) を選択し、シークレット **SecretsManagerTutorialAppuser** を選択します。
3. [Secret details] (シークレットの詳細) ページで、下へスクロールし、[Retrieve secret value] (シークレットの値を取得する) を選択します。
4. [Key/value] (キー/バリュー) テーブルで、**password** のシークレットの値をコピーします。

認証情報をテストするには

1. MySQL ワークベンチで、接続 [SecretsManagerTutorial] を右クリックし、[Edit Connection] (接続を編集) を選択します。
2. [Manage Server Connections] (サーバー接続を管理) ダイアログボックスで、[Username] (ユーザーネーム) に **appuser** と入力し、[Close] (閉じる) を選択します。
3. MySQL Workbench に戻り、接続 [SecretsManagerTutorial] を選択します。
4. [Open SSH Connection] (SSH 接続を開く) ダイアログボックスで、シークレットから取得したパスワードを [Password] (パスワード) に貼り付けて、[OK] をクリックします。

認証情報が有効であれば、MySQL Workbench でデータベースの設計ページが開きます。

こちらは、シークレットのローテーションが完了したことを示しています。シークレット内の認証情報が更新されます。これが、データベースに接続するための有効なパスワードとなります。

ステップ 4: リソースをクリーンアップする

別のローテーション戦略であるシングルユーザーローテーションを試したい場合は、リソースのクリーンアップをスキップして [the section called “シングルユーザーローテーション”](#) に進みます。

それ以外の場合は、潜在的な課金を回避し、インターネットにアクセスできる EC2 インスタンスを削除するために、このチュートリアルで作成した次のリソースとその前提条件を削除します。

- Amazon RDS データベースインスタンス。手順については、「Amazon RDS ユーザーガイド」の「[DB インスタンスの削除](#)」を参照してください。
- Amazon EC2 インスタンス。手順については、Linux インスタンス用 Amazon EC2 ユーザーガイドの「[インスタンスの終了](#)」を参照してください。

- Secrets Manager シークレット SecretsManagerTutorialAppuser。手順については、「[the section called “シークレットの削除”](#)」を参照してください。
- Secrets Manager エンドポイント。手順については、AWS PrivateLink ガイドの「[VPC エンドポイントを削除する](#)」を参照してください。
- VPC エンドポイント 手順については、AWS PrivateLink ガイドの「[VPC の削除](#)」を参照してください。

次のステップ

- [アプリケーションでシークレットを取得する方法](#)について説明します。
- [その他のローテーションスケジュール](#)について説明します。

AWS Secrets Manager のシングルユーザーローテーションを設定する

このチュートリアルでは、データベース認証情報を含むシークレットの単一ユーザーローテーションを設定する方法を学習します。シングルユーザーローテーションは、Secrets Manager がシークレットとデータベースの両方でユーザーの認証情報を更新するローテーション戦略です。詳細については、「[the section called “シングルユーザー”](#)」を参照してください。

チュートリアルを終了したら、チュートリアルからリソースをクリーンアップすることをお勧めします。本番環境では使用しないでください。

Secrets Manager のローテーションでは、シークレットとデータベースの更新に AWS Lambda 関数を使用されます。Lambda 関数を使用する場合のコストについては、「[料金](#)」を参照してください。

目次

- [許可](#)
- [前提条件](#)
- [ステップ 1: Amazon RDS データベースユーザーを作成する](#)
- [ステップ 2: データベースユーザー認証情報のシークレットを作成する](#)
- [ステップ 3: ローテーションされたパスワードをテストする](#)
- [ステップ 4: リソースをクリーンアップする](#)
- [次のステップ](#)

許可

このチュートリアル の前提条件として、AWS アカウント の管理者権限が必要となります。本番稼働環境では、各ステップで異なるロールを使用するのがベストプラクティスとなります。例えば、Amazon RDS データベースの作成にはデータベース管理者権限を持つロールを使用し、VPC とセキュリティグループの設定にはネットワーク管理者権限を持つロールを使用します。チュートリアル の各ステップでは同じアイデンティティを使用することが推奨されます。

本番稼働環境で許可を設定する方法については、「[認証とアクセスコントロール](#)」を参照してください。

前提条件

このチュートリアル の前提条件は [the section called “交代ユーザーローテーション”](#) です。最初のチュートリアルが完了した時点で、リソースのクリーンアップを実行しないでください。このチュートリアル の後、Amazon RDS データベースと、データベースの管理者認証情報を含む Secrets Manager シークレットを備えた現実的な環境ができました。データベースユーザーの認証情報を含む 2 番目のシークレットもありますが、このチュートリアルではそのシークレットを使用しません。

また、管理者認証情報を使用してデータベースに接続するための接続が、MySQL Workbench に設定されます。

ステップ 1: Amazon RDS データベースユーザーを作成する

まず、シークレットにその認証情報を保存するユーザーが必要になります。ユーザーを作成するには、シークレットに保存されている管理者認証情報を使用して Amazon RDS データベースにログインします。簡単にするために、このチュートリアルでは、データベースへの完全なアクセス許可を持つユーザーを作成します。運用環境では、これは一般的ではないため、最小権限の原則に従うことをお勧めします。

管理者パスワードを取得するには

1. Amazon RDS コンソールで、データベースに移動します。
2. [Configuration] (設定) タブの [Master Credentials ARN] (マスター認証情報 ARN)で、[Manage in Secrets Manager] (Secrets Managerの管理) を選択します。

Secrets Manager コンソールが開きます。

3. シークレットの詳細ページで、[Retrieve secret value] (シークレット値の取得) を選択します。

4. パスワードは [Secret value] (シークレット値) セクションに表示されます。

データベースユーザーを作成するには

1. MySQL ワークベンチで、接続 [SecretsManagerTutorial] を右クリックし、[Edit Connection] (接続を編集) を選択します。
2. [Manage Server Connections] (サーバー接続を管理) ダイアログボックスで、[Username] (ユーザーネーム) に **admin** と入力し、[Close] (閉じる) を選択します。
3. MySQL Workbench に戻り、接続 [SecretsManagerTutorial] を選択します。
4. シークレットから取得した管理者パスワードを入力します。
5. MySQL Workbenchの[Query] (クエリ) ウィンドウで、次のコマンド (強力なパスワードを含む) を入力し、[Execute] (実行) を選択します。

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

[Output] (出力) ウィンドウに、コマンドが実行されたことが表示されます。

ステップ 2: データベースユーザー認証情報のシークレットを作成する

次に、作成したばかりのユーザーの認証情報を格納するためのシークレットを作成し、即時ローテーションを含む自動ローテーションをオンにします。Secrets Manager はシークレットをローテーションします。これは、パスワードがプログラムによって生成されることを意味します - 誰もこの新しいパスワードを見たことはありません。ローテーションをすぐに開始すると、ローテーションが正しく設定されているかどうかを判断するのも役立ちます。

1. Secrets Manager のコンソール (<https://console.aws.amazon.com/secretsmanager/>) を開きます。
2. [Store a new secret] (新しいシークレットを保存する) を選択します。
3. [Choose secret type] (シークレットタイプを選択する) ページで、次の操作を行います。
 - a. [Secret type] (シークレットタイプ) で、[Credentials for Amazon RDS database] (Amazon RDS データベースの認証情報) を選択します。
 - b. [Credentials] (認証情報) で、ユーザーネーム **dbuser** と、MySQL Workbench を使って作成したデータベースユーザー向けに入力したパスワードを入力します。
 - c. [Database] (データベース) で、**secretsmanagertutorialdb** を選択します。

- d. [Next] (次へ) をクリックします。
4. [Configure secret] (シークレットを設定する) ページで、[Secret name] (シークレット名) に **SecretsManagerTutorialDbuser** と入力し、[Next] (次) を選択します。
5. [Configure Routing] (ローテーションを設定する) ページで、以下を実行します。
 - a. [Automatic rotation] (自動ローテーション) を有効化します。
 - b. [Rotation schedule] (ローテーションスケジュール) で、[Days] (日数) のスケジュールを設定します: [Duration] (期間) の 2 日間: **2h**。[Rotate immediately] (すぐにローテーションする) は選択したままにしておきます。
 - c. [Rotation function] (ローテーション関数) で [Create a rotation function] (ローテーション関数を作成) を選択し、[Function Name] (関数名) に **tutorial-single-user-rotation** と入力します。
 - d. [ローテーション戦略] で、[単一ユーザー] を選択します。
 - e. [Next] (次へ) をクリックします。
6. [Review] (確認) ページで [Store] (保存) を選択します。

Secrets Manager はシークレットの詳細ページに戻ります。ローテーション設定のステータスは、ページの上で確認できます。Secrets Manager では、Lambda ローテーション関数や Lambda 関数を実行する実行ロールなどのリソースを作成するときは CloudFormation を使用します。CloudFormation が終了すると、バナーがローテーションが予定されているシークレットに切り替わります。これで、最初のローテーションは完了です。

ステップ 3: ローテーションされたパスワードをテストする

最初のシークレットローテーションが完了すると (所要時間は数秒)、シークレットに有効な認証情報が残っているかどうかを確認できます。シークレット内のパスワードが、元の認証情報から変更されました。

シークレットから新しいパスワードを取得するには

1. Secrets Manager のコンソール (<https://console.aws.amazon.com/secretsmanager/>) を開きます。
2. [Secrets] (シークレット) を選択し、シークレット **SecretsManagerTutorialDbuser** を選択します。
3. [Secret details] (シークレットの詳細) ページで、下へスクロールし、[Retrieve secret value] (シークレットの値を取得する) を選択します。

4. [Key/value] (キー/バリュー) テーブルで、**password** のシークレットの値をコピーします。

認証情報をテストするには

1. MySQL ワークベンチで、接続 [SecretsManagerTutorial] を右クリックし、[Edit Connection] (接続を編集) を選択します。
2. [Manage Server Connections] (サーバー接続を管理) ダイアログボックスで、[Username] (ユーザーネーム) に **dbuser** と入力し、[Close] (閉じる) を選択します。
3. MySQL Workbench に戻り、接続 [SecretsManagerTutorial] を選択します。
4. [Open SSH Connection] (SSH 接続を開く) ダイアログボックスで、シークレットから取得したパスワードを [Password] (パスワード) に貼り付けて、[OK] をクリックします。

認証情報が有効であれば、MySQL Workbench でデータベースの設計ページが開きます。

ステップ 4: リソースをクリーンアップする

料金を発生させないために、このチュートリアルで作成したシークレットは削除します。手順については、「[the section called “シークレットの削除”](#)」を参照してください。

前のチュートリアルで作成したリソースをクリーンアップするには、[the section called “ステップ 4: リソースをクリーンアップする”](#) を参照してください。

次のステップ

- アプリケーションでシークレットを取得する方法について説明します。「[シークレットを取得する](#)」を参照してください。
- その他のローテーションスケジュールについて説明します。「[the section called “ローテーションスケジュール”](#)」を参照してください。

の認証とアクセスコントロール AWS Secrets Manager

Secrets Manager は [AWS Identity and Access Management \(IAM\)](#) を使用して、シークレットへのアクセスを保護します。IAM は認証とアクセスコントロールを提供します。認証は、リクエストを実行するアイデンティティを検証するものです。Secrets Manager は、パスワード、アクセスキー、および多要素認証 (MFA) トークンでのサインインプロセスを使用して、ユーザーのアイデンティティを検証します。「[へのサインイン AWS](#)」を参照してください。アクセスコントロールは、シークレットなどの AWS リソースで、承認された個人のみがオペレーションを実行できるようにします。Secrets Manager は、ポリシーを使用して、リソースにアクセスできるユーザー、およびそのアイデンティティがそれらのリソースに対して実行できるアクションを定義します。[IAM でのポリシーとアクセス許可](#)を参照してください。

Secrets Manager 管理者のアクセス許可

Secrets Manager 管理者のアクセス許可を付与するには、[\[Adding and removing IAM identity permissions\]](#) (IAM アイデンティティのアクセス許可の追加および削除) をクリックし、次のポリシーをアタッチします。

- [SecretsManagerReadWrite](#)
- [IAMFullAccess](#)

エンドユーザーには管理者のアクセス許可を付与しないことをお勧めします。これを付与すると、ユーザーはシークレットを作成および管理できますが、ローテーションを有効にするために必要なアクセス許可 (IAMFullAccess) により、エンドユーザーには適切ではない重要なアクセス許可が付与されます。

シークレットへのアクセス許可

IAM アクセス許可ポリシーを使用すると、シークレットにアクセスできるユーザーまたはサービスを制御できます。アクセス許可のポリシーは、誰がどのアクションをどのリソースで実行できるかを示します。次のようにできます。

- [the section called “アクセス許可ポリシーをアイデンティティにアタッチする”](#)
- [the section called “アクセス許可ポリシーをシークレットにアタッチする”](#)

Lambda ローターション関数のアクセス許可

Secrets Manager は、AWS Lambda 関数を使用してシークレットをローテーションします。Lambda 関数には、シークレットと、シークレットにその認証情報が含まれているデータベースまたはサービスへのアクセス許可が必要です。[ローテーションへのアクセス許可](#) を参照してください。

暗号化キーのアクセス許可

Secrets Manager は AWS Key Management Service (AWS KMS) キーを使用してシークレットを暗号化します。には、正しいアクセス許可 AWS マネージドキー `aws/secretsmanager` が自動的に付与されます。別の KMS キーを使用する場合、Secrets Manager にはそのキーに対するアクセス許可が必要です。「[the section called “KMS キーのアクセス許可”](#)」を参照してください。

レプリケーションのアクセス許可

IAM アクセス許可ポリシーを使用すると、シークレットを他のリージョンにレプリケートできるユーザーまたはサービスを制御します。[the section called “レプリケーションの防止”](#) を参照してください。

アクセス許可ポリシーをアイデンティティにアタッチする

アクセス許可ポリシーは [IAM アイデンティティ: ユーザー、ユーザーグループ、およびロール](#) にアタッチすることができます。アイデンティティベースのポリシーで、アイデンティティがアクセスできるシークレットと、アイデンティティがそのシークレットで実行できるアクションを指定します。詳細については、「[Adding and removing IAM identity permissions](#)」(IAM アクセス許可の追加と削除) を参照してください。

別のサービスのアプリケーションまたはユーザーを表すロールに権限を付与できます。例えば、Amazon EC2 インスタンスで実行されているアプリケーションは、データベースにアクセスする必要がある場合があります。EC2 インスタンスのプロファイルに IAM ロールを作成し、権限ポリシーを使用して、データベースの資格情報を含むシークレットへのアクセスをロールに付与することができます。詳細については、「[Using an IAM role to grant permissions to applications running on Amazon EC2 instances](#)」(IAM ロールを使用して Amazon EC2 インスタンス上で動作するアプリケーションに権限を付与する) を参照してください。その他、[Amazon Redshift](#)、[AWS Lambda](#)、[Amazon ECS](#) などのサービスにもロールをアタッチすることができます。

また、IAM 以外のアイデンティティシステムによって認証されたユーザーにアクセス許可を付与することもできます。例えば、Amazon Cognito 使用してサインインするモバイルアプリケーションユーザーに IAM ロールを関連付けることができます。ロールは、ロールのアクセス許可ポリシーにあるアクセス許可を持つ、一時的な認証情報をアプリに付与します。次に、アクセス許可ポリシーを使用して、シークレットへのアクセス許可をロールに付与できます。詳細については、「[Identity providers and federation](#)」(ID プロバイダとフェデレーション)を参照してください。

アイデンティティベースのポリシーを使用して以下を行うことができます。

- 複数のシークレットへのアクセスをアイデンティティに許可します。
- 新しいシークレットを作成できるユーザーと、まだ作成されていないシークレットにアクセスできるユーザーを制御します。
- IAM グループにシークレットへのアクセス許可を付与します。

詳細については、「[the section called “アクセス許可ポリシーの例”](#)」を参照してください。

アクセス許可ポリシーを AWS Secrets Manager シークレットにアタッチする

リソースベースのポリシーで、シークレットにアクセスできるユーザーと、ユーザーがそのシークレットで実行できるアクションを指定します。リソースベースのポリシーを使用して、以下を行うことができます。

- 単一のシークレットへのアクセスを複数のユーザーまたはロールに許可します。
- AWS 他のアカウントのユーザーまたはロールにアクセス権を付与します。

[the section called “アクセス許可ポリシーの例”](#) を参照してください。

コンソールでリソースベースのポリシーをシークレットにアタッチすると、Secrets Manager は自動化された推論エンジン [Zelkova](#) と API `ValidateResourcePolicy` を使用して、シークレットへのアクセス許可を幅広い範囲の IAM プリンシパルに付与しないようにします。または、CLI または SDK からの `BlockPublicPolicy` パラメータを含む `PutResourcePolicy` API を呼び出します。

⚠ Important

BlockPublicPolicyリソースポリシーの検証とパラメータは、シークレットに直接関連付けられているリソースポリシーによってパブリックアクセスが許可されるのを防ぐことで、リソースを保護するのに役立ちます。これらの機能を使用することに加えて、以下のポリシーを注意深く調べて、パブリックアクセスを許可していないことを確認してください。

- AWS 関連するプリンシパル (IAM ロールなど) にアタッチされている ID ベースのポリシー
- AWS 関連リソースにアタッチされたリソースベースのポリシー (キーなど) AWS Key Management Service AWS KMS

シークレットへの権限を確認するには、[を参照してください。](#) [シークレットへのアクセス許可を持つユーザーを特定する](#)

シークレットのリソースポリシーを表示、変更、または削除するには (コンソール)

1. Secrets Manager のコンソール (<https://console.aws.amazon.com/secretsmanager/>) を開きます。
2. シークレットのリストから、自分のシークレットを選択します。
3. シークレットの詳細ページの [概要] タブの [リソース許可] セクションで、[許可を編集] を選択します。
4. [Code] (コード) フィールドで、次のいずれかの操作を行い、[Save] (保存する) をクリックします。
 - リソースポリシーをアタッチまたは変更するには、ポリシーを入力します。
 - ポリシーを削除するには、[Code] (コード) フィールドをクリアします。

AWS CLI

Example リソースポリシーを取得する

次に、シークレットにアタッチされたリソースベースのポリシーを取得する、[get-resource-policy](#) の例を示します。

```
aws secretsmanager get-resource-policy \  
  --secret-id MyTestSecret
```

Example リソースポリシーを削除する

次に、シークレットにアタッチされているリソースベースのアポリシーを削除する、[delete-resource-policy](#) の例を示します。

```
aws secretsmanager delete-resource-policy \  
  --secret-id MyTestSecret
```

Example リソースポリシーを追加する

次の [put-resource-policy](#) の例では、ポリシーが広範なアクセスをシークレットに提供していないことを最初に確認しながら、シークレットに許可ポリシーを追加しています。このポリシーは、ファイルから読み込まれます。詳細については、『AWS CLI ユーザーガイド』の「[AWS CLI ファイルからのパラメータのロード](#)」を参照してください。

```
aws secretsmanager put-resource-policy \  
  --secret-id MyTestSecret \  
  --resource-policy file://mypolicy.json \  
  --block-public-policy
```

mypolicy.json の内容 :

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Principal": {  
        "AWS": "arn:aws:iam::123456789012:role/MyRole"  
      },  
      "Action": "secretsmanager:GetSecretValue",  
      "Resource": "*"   
    }   
  ]  
}
```

AWS SDK

シークレットにアタッチされているポリシーを取得するには、[GetResourcePolicy](#) を使用します。

シークレットにアタッチされているポリシーを削除するには、[DeleteResourcePolicy](#) を使用します。

ポリシーをシークレットにアタッチするには、[PutResourcePolicy](#) を使用します。ポリシーがすでにアタッチされている場合、コマンドはそのポリシーを新しいポリシーに置き換えます。ポリシーは、JSON 構造化テキストとしてフォーマットする必要があります。[JSON ポリシードキュメント構造](#)を参照してください。[the section called “アクセス許可ポリシーの例”](#) を使用して、ポリシーの記述を開始します。

詳細については、「[the section called “AWS SDKs”](#)」を参照してください。

AWS の管理ポリシー AWS Secrets Manager

AWS 管理ポリシーは、によって作成および管理されるスタンドアロンのポリシーです。AWS AWS 管理ポリシーは、ユーザー、グループ、ロールにアクセス権限を割り当てることができるように、多くの一般的な使用事例にアクセス許可を与えるように設計されています。

AWS 管理ポリシーでは、AWS すべての顧客が使用できるようになっているため、特定のユースケースでは最小権限のアクセス権限が付与されない場合があることに注意してください。ユースケースに固有の [カスターマネージドポリシー](#) を定義して、許可をさらに減らすことをお勧めします。

AWS 管理ポリシーで定義されている権限は変更できません。AWS 管理ポリシーで定義されている権限を更新すると AWS、その更新はポリシーがアタッチされているすべてのプリンシパル ID (ユーザー、グループ、ロール) に影響します。AWS AWS 管理ポリシーが更新される可能性が最も高いのは、新しい API 操作が既存のサービスで開始されたときや、新しい API AWS のサービス操作が使用可能になったときです。

詳細については、「IAM ユーザーガイド」の「[AWS マネージドポリシー](#)」を参照してください。

AWS 管理ポリシー: SecretsManagerReadWrite

このポリシーは、Amazon RDS、Amazon Redshift AWS Secrets Manager、Amazon DocumentDB リソースへの読み取り/書き込みアクセスを提供します。これには、Amazon RDS、Amazon Redshift、Amazon DocumentDB リソースを記述する権限、AWS KMS およびシークレットの暗号化と復号化に使用するアクセス権限が含まれます。このポリシーでは、AWS CloudFormation 変更セットを作成したり、が管理する Amazon S3 バケットからローテーションテンプレートを取得したり、AWS Lambda 関数を一覧表示したり AWS、Amazon EC2 VPC を記述したりする権限も付与されます。これらのアクセス許可は、コンソールが既存のローテーション機能を使用してローテーションを設定するために必要です。

新しいローテーション関数を作成するには、AWS CloudFormation AWS Lambda スタックと実行ロールを作成する権限も必要です。[IAM FullAccess](#) 管理ポリシーを割り当てることができます。[ローテーションへのアクセス許可](#) を参照してください。

アクセス許可の詳細

このポリシーには、以下のアクセス許可が含まれています。

- `secretsmanager` – プリンシパルに Secrets Manager の全アクションの実行を許可します。
- `cloudformation` – AWS CloudFormation プリンシパルがスタックを作成できるようにします。これは、コンソールを使用してローテーションをオンにするプリンシパルがスタックを通じて Lambda ローテーション関数を作成できるようにするために必要です。AWS CloudFormation 詳細については、「[the section called “Secrets Manager が AWS CloudFormation を使用する方法”](#)」を参照してください。
- `ec2` – プリンシパルに Amazon EC2 VPC の記述を許可します。これは、コンソールを使用するプリンシパルが、シークレットに保存している認証情報のデータベースと同じ VPC 内に、ローテーション関数を作成できるようにするために必要です。
- `kms` – AWS KMS プリンシパルが暗号操作にキーを使用できるようにします。これは、Secrets Manager がシークレットを暗号化および復号できるようにするために必要です。詳細については、「[the section called “シークレット暗号化と復号”](#)」を参照してください。
- `lambda` – プリンシパルに Lambda ローテーション関数の一覧表示を許可します。これは、コンソールを使用するプリンシパルが、既存のローテーション関数を選択できるようにするために必要です。
- `rds` – プリンシパルに Amazon RDS DB のクラスターとインスタンスの記述を許可します。これは、コンソールを使用するプリンシパルが Amazon RDS クラスターまたはインスタンスを選択できるようにするために必要です。
- `redshift` – プリンシパルに Amazon Redshift でのクラスターの記述を許可します。これは、コンソールを使用するプリンシパルが Amazon Redshift クラスターを選択できるようにするために必要です。
- `redshift-serverless` – プリンシパルが Amazon Redshift サーバーレスで名前空間を記述できるようにします。これは、コンソールを使用するプリンシパルが Amazon Redshift サーバーレス名前空間を選択できるようにするために必要です。
- `docdb-elastic` – プリンシパルに Amazon DocumentDB での Elastic クラスターの記述を許可します。これは、コンソールを使用するプリンシパルが、Amazon DocumentDB の Elastic クラスターを選択できるようにするために必要です。

- tag – プリンシパルに、アカウント内のタグ付けされた全リソースの取得を許可します。
- serverlessrepo – プリンシパルが変更セットを作成できるようにします。AWS CloudFormation これは、コンソールを使用するプリンシパルが Lambda ローターション関数を作成できるようにするために必要です。詳細については、「[the section called “Secrets Manager が AWS CloudFormation を使用する方法”](#)」を参照してください。
- s3 – プリンシパルが管理する Amazon S3 バケットからオブジェクトを取得できるようにします。AWS このバケットには Lambda [ローテーション関数のテンプレート](#) が含まれます。このアクセス許可は、コンソールを使用するプリンシパルがバケット内のテンプレートに基づいて、Lambda ローターション関数を作成できるようにするために必要です。詳細については、「[the section called “Secrets Manager が AWS CloudFormation を使用する方法”](#)」を参照してください。

ポリシーを表示するには、[SecretsManagerReadWrite JSON ポリシードキュメントを参照してください](#)。

Secrets Manager AWS による管理ポリシーの更新

Secrets Manager AWS の管理ポリシーの更新に関する詳細を表示します。

変更	説明	日付
SecretsManagerReadWrite - 既存ポリシーへの更新	このポリシーは、Amazon Redshift サーバーレスへの記述アクセスを許可するように更新されました。これにより、コンソールユーザーは Amazon Redshift シークレットを作成するときに Amazon Redshift サーバーレス名前空間を選択できます。	2024 年 3 月 12 日
SecretsManagerReadWrite - 既存ポリシーへの更新	このポリシーは、Amazon DocumentDB の Elastic クラスターへの記述アクセスを許可するように更新されました。これにより、コンソールユーザーは Amazon	2023 年 9 月 12 日

変更	説明	日付
	DocumentDB シークレットを作成するときに Elastic クラスターを選択できます。	
SecretsManagerReadWrite - 既存ポリシーへの更新	<p>このポリシーは、Amazon Redshift への記述アクセスを許可するように更新されました。これにより、コンソールユーザーは Amazon Redshift シークレットを作成するときに Amazon Redshift クラスターを選択できます。この更新では、Lambda ローターション関数テンプレートを格納する Amazon S3 AWS バケットへの読み取りアクセスを許可する新しい権限も追加されました。</p>	2020 年 6 月 24 日
SecretsManagerReadWrite - 既存ポリシーへの更新	<p>このポリシーは、Amazon RDS クラスターへの記述アクセスを許可するように更新されました。これにより、コンソールユーザーは Amazon RDS シークレットを作成するときにクラスターを選択できます。</p>	2018 年 5 月 3 日
SecretsManagerReadWrite - 新しいポリシー	<p>Secrets Manager は、Secrets Manager へのすべての読み取り/書き込みアクセスで、コンソールを使用するために必要なアクセス許可を付与するポリシーを作成しました。</p>	2018 年 04 月 4 日

変更	説明	日付
Secrets Manager が変更の追跡を開始	Secrets Manager は、AWS 管理ポリシーの変更の追跡を開始しました。	2018 年 04 月 4 日

AWS Secrets Manager シークレットへのアクセス許可を持つユーザーを特定する

デフォルトでは、IAM アイデンティティにはシークレットへのアクセス許可がありません。シークレットへのアクセスを承認する時、Secrets Manager はシークレットにアタッチされたリソースベースのポリシー、およびリクエストを送信している IAM ユーザーまたは IAM ロールにアタッチされたすべてのアイデンティティベースのポリシーを評価します。これを行うために、Secrets Manager は、IAM ユーザーガイドの [\[Determining whether a request is allowed or denied\]](#) (リクエストの許可または拒否を決定する) に記載されているものと類似したプロセスを使用します。

複数のポリシーが 1 つのリクエストに適用される場合、Secrets Manager は階層を使用してアクセス許可を制御します。

1. ポリシー内のステートメントに明示的な deny が含まれている場合、リクエストアクションとリソースに一致します。

この明示的な deny によって、他のすべての内容が上書きされ、アクションがブロックされません。

2. 明示的な deny はないが、ステートメントに明示的な allow が含まれている場合、リクエストアクションとリソースに一致します。

この明示的な allow によって、リクエスト内のアクションにステートメント内のリソースへのアクセスが付与されます。

アイデンティティとシークレットが 2 つの異なるアカウントにある場合は、シークレットのリソースポリシーとアイデンティティにアタッチされたポリシーの両方で allow が含まれている必要があります。含まれていない場合、AWS はリクエストを拒否します。詳細については、「[クロスアカウントアクセス](#)」を参照してください。

3. リクエストアクションとリソースに一致する明示的な allow が含まれているステートメントがない場合

AWS はデフォルトでリクエストを拒否します。これは暗黙的拒否と呼ばれます。

シークレットのリソースベースのポリシーを表示するには

- 以下のいずれかを実行します。
 - Secrets Manager のコンソール (<https://console.aws.amazon.com/secretsmanager/>) を開きます。シークレットの詳細ページの [Resource permissions] (リソースに対するアクセス許可) セクションで、[Edit permissions] (アクセス許可の編集) をクリックします。
 - [get-resource-policy](#) を呼び出す場合は AWS CLI を、[GetResourcePolicy](#) を呼び出す場合は AWS SDK を使用します。

アイデンティティベースのポリシーを使用してアクセスできるユーザーを特定するには

- IAM ポリシーシミュレーターを使用します。[IAM Policy Simulator を使用した IAM ポリシーのテスト](#) を参照してください。

別のアカウントから AWS Secrets Manager シークレットにアクセスする

1つのアカウントのユーザーに別のアカウントのシークレットへのアクセス (クロスアカウントアクセス) を許可するには、リソースポリシーとアイデンティティポリシーの両方でアクセスを許可する必要があります。これは、シークレットと同じアカウントのアイデンティティにアクセスを許可することとは異なります。

また、シークレットの暗号化に使用している KMS キーをアイデンティティで使用できるようにする必要があります。これは、クロスアカウントアクセスに AWS マネージドキー (aws/secretsmanager) を使用できないためです。代わりに、作成した KMS キーを使用してシークレットを暗号化し、キーポリシーをそれにアタッチする必要があります。KMS キーの作成には料金が発生します。シークレットの暗号化キーを変更するには、[the section called “シークレットの変更”](#) を参照してください。

以下のポリシーの例では、Account1 にシークレットと暗号化キーがあり、Account2 にシークレット値へのアクセスを許可するアイデンティティがあると仮定します。

ステップ 1: リソースポリシーを Account1 のシークレットにアタッチする

- 次のポリシーでは、*Account2 ApplicationRole* のが Account*Account1* のシークレットにアクセスすることを許可します。このポリシーを使用するには、「[the section called “アクセス許可ポリシーをシークレットにアタッチする”](#)」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::Account2:role/ApplicationRole"
      },
      "Action": "secretsmanager:GetSecretValue",
      "Resource": "*"
    }
  ]
}
```

ステップ 2: Account1 の KMS キーのキーポリシーにステートメントを追加する

- 次のキーポリシーステートメントでは、*Account2 ApplicationRole* のが Account*Account1* の KMS キーを使用して Account*Account1* のシークレットを復号することを許可します。このステートメントを使用するには、それを KMS キーのキーポリシーに追加します。詳細については、[キーポリシーの変更](#)を参照してください。

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::Account2:role/ApplicationRole"
  },
  "Action": [
    "kms:Decrypt",
    "kms:DescribeKey"
  ],
  "Resource": "*"
}
```

ステップ 3: Account2 のアイデンティティにアイデンティティポリシーをアタッチする

- 次のポリシーでは、*Account2 ApplicationRole* のが *Account1* のシークレットにアクセスし、*Account1* にも存在する暗号化キーを使用してシークレット値を復号することを許可します。このポリシーを使用するには、[the section called “アクセス許可ポリシーをアイデンティティにアタッチする”](#) を参照してください。シークレットの ARN は、Secrets Manager コンソールのシークレット詳細ページの [Secret ARN] (シークレット ARN) で確認できます。または、[describe-secret](#) を呼び出すこともできます。

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    {
      "Effect": "Allow",
      "Action": "secretsmanager:GetSecretValue",
      "Resource": "SecretARN"
    },
    {
      "Effect": "Allow",
      "Action": "kms:Decrypt",
      "Resource": "arn:aws:kms:Region:Account1:key/EncryptionKey"
    }
  ]
}
```

オンプレミス環境からシークレットにアクセスする

AWS Identity and Access Management Roles Anywhere を使用して、の外部で実行されるサーバー、コンテナ、アプリケーションなどのワークロードの一時的なセキュリティ認証情報を IAM で取得できます AWS。ワークロードは、AWS アプリケーションで使用するのと同じ IAM ポリシーと IAM ロールを使用して AWS リソースにアクセスできます。IAM Roles Anywhere を使用すると、Secrets Manager を使用して、のリソースやアプリケーションサーバーなどの AWS オンプレミスデバイスからアクセスできる認証情報を保存および管理できます。詳細については、「[IAM Roles Anywhere ユーザーガイド](#)」を参照してください。

のアクセス許可ポリシーの例 AWS Secrets Manager

アクセス許可ポリシーは JSON 構造化テキストです。「[JSON policy document structure](#)」(JSON ポリシードキュメント構造) を参照してください。

リソースやアイデンティティにアタッチするアクセス許可ポリシーは、互いに非常によく似ています。シークレットにアクセスするためにポリシーに含める要素には、次のようなものがあります。

- **Principal:** アクセス許可を付与するユーザー。「IAM ユーザーガイド」の「[Specifying a principal](#)」(プリンシパルの指定)を参照してください。ポリシーをアイデンティティにアタッチする場合、ポリシーに Principal 要素は含めません。
- **Action:** ロールが実行できるアクション。「[the section called “Secrets Manager のアクション”](#)」を参照してください。
- **Resource:** ロールがアクセスできるシークレット。[the section called “Secrets Manager リソース”](#)を参照してください。

ワイルドカード文字 (*) の意味は、ポリシーをアタッチする内容によって異なります。

- シークレットにアタッチされたポリシーでは、* は、ポリシーがこのシークレットに適用されることを意味します。
- アイデンティティにアタッチされたポリシーでは、* は、ポリシーがアカウント内のすべてのリソース (シークレットを含む) に適用されることを意味します。

ポリシーをシークレットにアタッチするには、[the section called “アクセス許可ポリシーをシークレットにアタッチする”](#)を参照してください。

ポリシーをアイデンティティにアタッチするには、[the section called “アクセス許可ポリシーをアイデンティティにアタッチする”](#)を参照してください。

トピック

- [例: 個別のシークレット値を取得するためのアクセス許可](#)
- [例: 個々のシークレットを読み書きするアクセス許可](#)
- [例: バッチ内のシークレット値のグループを取得するアクセス許可](#)
- [例: ワイルドカード](#)
- [例: シークレットを作成するアクセス許可](#)
- [例: シークレットを暗号化するための特定の AWS KMS キーを拒否する](#)
- [例: アクセス許可と VPC](#)
- [例: タグを使用してシークレットへのアクセスを制御する](#)
- [例: シークレットのタグに一致しているタグを持つアイデンティティへのアクセスを制限する](#)
- [例: サービスプリンシパル](#)

例: 個別のシークレット値を取得するためのアクセス許可

シークレット値を取得するアクセス許可を付与するには、ポリシーをシークレットまたはアイデンティティにアタッチできます。使用するポリシーの種類を決定する方法については、「[アイデンティティベースのポリシーおよびリソースベースのポリシー](#)」を参照してください。ポリシーをアタッチする方法については、[the section called “アクセス許可ポリシーをシークレットにアタッチする”](#) および [the section called “アクセス許可ポリシーをアイデンティティにアタッチする”](#) を参照してください。

次の例は、シークレットへのアクセスを許可する 2 つの異なる方法を示しています。最初の例は、シークレットにアタッチできるリソースベースのポリシーです。この例は、単一のシークレットへのアクセスを複数のユーザーまたはロールに付与する場合に役立てることができます。2 番目の例は、IAM のユーザーまたはロールにアタッチできるアイデンティティベースのポリシーです。この例は、IAM グループにアクセス許可を付与する場合に役立てることができます。バッチ API コールでシークレットのグループを取得するためのアクセス許可を付与するには、「[the section called “例: バッチ内のシークレット値のグループを取得するアクセス許可”](#)」を参照してください。

Example 1 つのシークレットを読み取る (シークレットにアタッチする)

シークレットに次のポリシーをアタッチすると、シークレットへのアクセスを許可することができます。このポリシーを使用するには、「[the section called “アクセス許可ポリシーをシークレットにアタッチする”](#)」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::AccountId:role/EC2RoleToAccessSecrets"
      },
      "Action": "secretsmanager:GetSecretValue",
      "Resource": "*"
    }
  ]
}
```


Example カスタマーマネージドキー (ID にアタッチ) を使用して暗号化されたシークレットを読み込む

シークレットがカスタマーマネージドキーを使用して暗号化されている場合、ID に次のポリシーをアタッチすることで、シークレットの読み取りを許可できます。このポリシーを使用するには、「[the section called “アクセス許可ポリシーをアイデンティティにアタッチする”](#)」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "secretsmanager:GetSecretValue",
      "Resource": "SecretARN"
    },
    {
      "Effect": "Allow",
      "Action": "kms:Decrypt",
      "Resource": "KMSKeyARN"
    }
  ]
}
```

例: 個々のシークレットを読み書きするアクセス許可

Example 1 つのシークレットを読み取って記述する (アイデンティティにアタッチする)

次のポリシーをアイデンティティにアタッチすると、シークレットへのアクセスを許可することができます。このポリシーを使用するには、「[the section called “アクセス許可ポリシーをアイデンティティにアタッチする”](#)」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue",
        "secretsmanager:DescribeSecret"
      ],

```

```
    "Resource": "SecretARN"
  }
]
}
```

例: バッチ内のシークレット値のグループを取得するアクセス許可

Example バッチ内のシークレットのグループを読み取る (アイデンティティにアタッチする)

次のポリシーをアイデンティティにアタッチすると、バッチ API コールのシークレットのグループを取得するためのアクセス許可を付与できます。このポリシーは、バッチコールに他のシークレットが含まれていても、*SecretARN1*、*SecretARN2*、*SecretARN3* で指定されたシークレットのみを取得できるように呼び出し元を制限します。呼び出し元がバッチ API コールで他のシークレットもリクエストしたとしても、Secrets Manager はそれらを返しません。詳細については、「」を参照してください[BatchGetSecretValue](#)。このポリシーを使用するには、「[the section called “アクセス許可ポリシーをアイデンティティにアタッチする”](#)」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:BatchGetSecretValue",
        "secretsmanager:ListSecrets"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Resource": [
        "SecretARN1",
        "SecretARN2",
        "SecretARN3"
      ]
    }
  ]
}
```

例: ワイルドカード

ワイルドカードを使用して、ポリシーの要素に値のセットを含めることができます。

Example パス内のすべてのシークレットにアクセスする (アイデンティティにアタッチする)

次のポリシーは、`TestEnv#/*` で始まる名前のすべてのシークレットを取得するアクセスを許可します。このポリシーを使用するには、「[the section called “アクセス許可ポリシーをアイデンティティにアタッチする”](#)」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "secretsmanager:GetSecretValue",
    "Resource": "arn:aws:secretsmanager:Region:AccountId:secret:TestEnv/*"
  }
}
```

Example すべてのシークレットのメタデータへのアクセス (アイデンティティにアタッチする)

以下のポリシーは、DescribeSecret および List で始まるアクセス許可 (ListSecrets および ListSecretVersionIds) を付与します。このポリシーを使用するには、「[the section called “アクセス許可ポリシーをアイデンティティにアタッチする”](#)」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "secretsmanager:DescribeSecret",
      "secretsmanager:List*"
    ],
    "Resource": "*"
  }
}
```

Example シークレット名が一致する (アイデンティティにアタッチする)

次のポリシーは、Secrets Manager のシークレット名へのすべてのアクセス許可を付与します。このポリシーを使用するには、「[the section called “アクセス許可ポリシーをアイデンティティにアタッチする”](#)」を参照してください。

シークレット名を照合するには、リージョン、アカウント ID、シークレット名、ワイルドカード (?) を組み合わせてシークレットの ARN を作成し、個々のランダムな文字に一致させます。Secrets Manager は、ARN の一部としてシークレット名に 6 つのランダムな文字を追加するため、このワイルドカードを使用してこれらの文字に一致させることができます。"another_secret_name-*" 構文を使用した場合、Secrets Manager は、意図した 6 つのランダムな文字があるシークレットだけでなく、"another_secret_name-<anything-here>a1b2c3" にも一致します。

シークレットの ARN の 6 つのランダムな文字以外のすべての部分を予測できるため、ワイルドカード文字の '??????' 構文を使用することで、まだ存在していないシークレットに安全にアクセス許可を付与することができます。ただし、シークレットを削除して同じ名前で作成すると、6 つの文字が変更されたにもかかわらず、ユーザーが自動的に新しいシークレットへのアクセス許可を受け取ることに注意してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "secretsmanager:*",
      "Resource": [
        "arn:aws:secretsmanager:Region:AccountId:secret:a_specific_secret_name-a1b2c3",
        "arn:aws:secretsmanager:Region:AccountId:secret:another_secret_name-??????"
      ]
    }
  ]
}
```

例: シークレットを作成するアクセス許可

シークレットを作成するアクセス許可を付与するには、ユーザーが属する IAM グループにアクセス許可ポリシーをアタッチすることをお勧めします。「[IAM ユーザーグループ](#)」を参照してください。

Example シークレットを作成する (アイデンティティにアタッチする)

次のポリシーは、シークレットを作成してシークレットのリストを表示するアクセス許可を付与します。このポリシーを使用するには、「[the section called “アクセス許可ポリシーをアイデンティティにアタッチする”](#)」を参照してください。

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "secretsmanager:CreateSecret",
      "secretsmanager:ListSecrets"
    ],
    "Resource": "*"
  }
]
```

例: シークレットを暗号化するための特定の AWS KMS キーを拒否する

Important

カスタマーマネージドキーを拒否するには、キーポリシーまたはキー許可を使用してアクセスを制限することをお勧めします。詳細については、「[AWS Key Management Service デベロッパーガイド](#)」の「[の認証とアクセスコントロール AWS KMS](#)」を参照してください。

Example AWS マネージドキーを拒否する `aws/secretsmanager` (ID にアタッチする)

次のポリシーは、シークレットを作成または更新`aws/secretsmanager`するための AWS マネージドキーの使用を拒否する方法を示しています。つまり、シークレットはカスタマーマネージドキーを使用して暗号化する必要があります。`aws/secretsmanager` キーが存在する場合は、そのキー ID も含める必要があります。Secrets Manager はそれを AWS マネージドキーとして解釈するため、空の文字列も含めます`aws/secretsmanager`。2 番目のステートメントは、KMS キーを含まないシークレットを作成するリクエストを拒否します。これは、Secrets Manager がそれを AWS マネージドキーとして解釈するためです`aws/secretsmanager`。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "RequireCustomerManagedKeysOnSecrets",
      "Effect": "Deny",
      "Action": [
        "secretsmanager:CreateSecret",
        "secretsmanager:UpdateSecret"
      ]
    }
  ]
}
```

```
    ],
    "Resource": "*",
    "Condition": {
      "ForAnyValue:StringLikeIfExists": {
        "secretsmanager:KmsKeyId": [
          "*alias/aws/secretsmanager",
          "*<key_ID_of_the_AWS_managed_key>",
          ""
        ]
      }
    }
  },
  {
    "Sid": "RequireKmsKeyIdParameterOnCreate",
    "Effect": "Deny",
    "Action": "secretsmanager:CreateSecret",
    "Resource": "*",
    "Condition": {
      "Null": {
        "secretsmanager:KmsKeyId": "true"
      }
    }
  }
]
```

例: アクセス許可と VPC

VPC 内から Secrets Manager にアクセスする必要がある場合は、アクセス許可ポリシーに条件を含めることで、Secret Manager へのリクエストが VPC から確実に送信されるようにすることができます。詳細については、「[VPC エンドポイントの条件](#)」および「[VPC エンドポイント](#)」を参照してください。

他の AWS サービスからのシークレットへのアクセスリクエストも VPC から送信されていることを確認してください。そうしないと、このポリシーによってアクセスが拒否されます。

Example VPC エンドポイントを通過するようリクエストを要求する (シークレットにアタッチする)

例えば、次のポリシーでは、リクエストが VPC エンドポイント `vpce-1234a5678b9012c` を通過して送信されている場合にのみ、ユーザーが Secrets Manager オペレーションを実行できます。このポリシーを使用するには、「[the section called “アクセス許可ポリシーをシークレットにアタッチする”](#)」を参照してください。

```
{
  "Id": "example-policy-1",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "RestrictGetSecretValueoperation",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "secretsmanager:GetSecretValue",
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "aws:sourceVpce": "vpce-1234a5678b9012c"
        }
      }
    }
  ]
}
```

Example VPC から送信するようリクエストを要求する (シークレットにアタッチする)

次のポリシーでは、シークレットを作成および管理するコマンドが許可されるのは、そのコマンドが `vpce-12345678` から送信されている場合のみです。また、このポリシーでは、リクエストが `vpc-2b2b2b2b` から届いた場合にのみ、シークレットの暗号化された値へのアクセスを使用するオペレーションを許可します。1つのVPCでアプリケーションを実行する場合は、このようなポリシーを使用できますが、管理機能には2番目の切り離されたVPCを使用します。このポリシーを使用するには、「[the section called “アクセス許可ポリシーをシークレットにアタッチする”](#)」を参照してください。

```
{
  "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"
    }
  }
}
]
```

例: タグを使用してシークレットへのアクセスを制御する

タグを使用してシークレットへのアクセスを制御することができます。タグを使用したアクセス許可の制御は、急速に成長している環境や、ポリシー管理が煩雑になっている状況で役に立ちます。シークレットに特定のタグがある場合、1つの方法として、シークレットにタグをアタッチしてから、アイデンティティにアクセス許可を付与することができます。

Example 特定のタグを持つシークレットへのアクセスを許可する (アイデンティティにアタッチする)

次のポリシーでは、キー `ServerName` 「」と値 `ServerABC` のタグを持つシークレット `DescribeSecret` に対して を許可します。このポリシーを使用するには、「[the section called “アクセス許可ポリシーをアイデンティティにアタッチする”](#)」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "secretsmanager:DescribeSecret",
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "secretsmanager:ResourceTag/ServerName": "ServerABC"
      }
    }
  }
}
```

例: シークレットのタグに一致しているタグを持つアイデンティティへのアクセスを制限する

1つの方法として、シークレットと IAM アイデンティティの両方にタグをアタッチすることができます。次に、アイデンティティのタグがシークレットのタグと一致している場合にオペレーションを許可するアクセス許可ポリシーを作成します。完全なチュートリアルについては、「[タグに基づいてシークレットにアクセスするためのアクセス許可を定義する](#)」を参照してください。

タグを使用したアクセス許可の制御は、急速に成長している環境や、ポリシー管理が煩雑になっている状況で役に立ちます。詳細については、「[AWSの ABAC とは](#)」を参照してください。

Example シークレットと同じタグを持つロールへのアクセスを許可する (シークレットにアタッチする)

以下のポリシーは、タグ `AccessProject` がシークレットとロールに対して同じ値を持つ場合にのみ、アカウント `123456789012` に `GetSecretValue` を付与します。このポリシーを使用するには、「[the section called “アクセス許可ポリシーをシークレットにアタッチする”](#)」を参照してください。

```
{
```

```
"Version": "2012-10-17",
"Statement": {
  "Effect": "Allow",
  "Principal": {
    "AWS": "123456789012"
  },
  "Condition": {
    "StringEquals": {
      "aws:ResourceTag/AccessProject": "${ aws:PrincipalTag/AccessProject }"
    }
  },
  "Action": "secretsmanager:GetSecretValue",
  "Resource": "*"
}
}
```

例: サービスプリンシパル

シークレットにアタッチされたリソースポリシーに [AWS サービスプリンシパル](#) が含まれている場合は、[aws:SourceArn](#) および [aws:SourceAccount](#) グローバル条件キーを使用することをお勧めします。ARN とアカウントの値は、別の AWS サービスから Secrets Manager にリクエストが来たときにのみ、承認コンテキストに含まれます。この条件の組み合わせにより、混同される可能性のある [副次的なシナリオ](#) を回避することができます。

リソース ARN にリソースポリシーで許可されていない文字が含まれている場合、そのリソース ARN を [aws:SourceArn](#) 条件キーの値として使用することはできません。その代わりに [aws:SourceAccount](#) 条件キーを使用してください。その他の要件の詳細については、「[IAM requirements](#)」(IAMの要件) を参照してください。

サービスプリンシパルは通常、シークレットにアタッチされたポリシーではプリンシパルとして使用されませんが、一部の AWS サービスではそれが必要です。サービスがシークレットにアタッチする必要があるリソースポリシーの詳細については、サービスのドキュメントを参照してください。

Example サービスがサービスプリンシパルを使用してシークレットにアクセスできるようにする (シークレットにアタッチ)

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

```
    "Effect": "Allow",
    "Principal": {
      "Service": [
        "service-name.amazonaws.com"
      ]
    },
    "Action": "secretsmanager:GetSecretValue",
    "Resource": "*",
    "Condition": {
      "ArnLike": {
        "aws:sourceArn": "arn:aws:service-name::123456789012:*"
      },
      "StringEquals": {
        "aws:sourceAccount": "123456789012"
      }
    }
  }
}
```

のアクセス許可リファレンス AWS Secrets Manager

アクセス許可のポリシーを構成している要素については、「[JSON policy document structure](#)」と「[IAM JSON policy elements reference](#)」を参照してください。

独自のアクセス許可ポリシーの作成を開始するには、「[the section called “アクセス許可ポリシーの例”](#)」を参照してください。

[アクション] テーブルの [リソースタイプ] 列は、各アクションがリソースレベルの許可をサポートしているかどうかを示します。この列に値がない場合は、ポリシーステートメントの Resource 要素で、ポリシーが適用されるすべてのリソース（「*」）を指定する必要があります。列にリソースタイプが含まれる場合、そのアクションを含むステートメントでそのタイプの ARN を指定できます。アクションで 1 つ以上のリソースが必須となっている場合、呼び出し元には、それらのリソースを伴うアクションを使用するための許可が付与されている必要があります。必須リソースは、アスタリスク (*) でテーブルに示されています。IAM ポリシーの Resource 要素でリソースアクセスを制限する場合は、必要なリソースタイプごとに ARN またはパターンを含める必要があります。一部のアクションでは、複数のリソースタイプがサポートされています。リソースタイプがオプション（必須として示されていない）の場合、オプションのリソースタイプのいずれかを使用することを選択できます。

[アクション] テーブルの [条件キー] 列には、ポリシーステートメントの Condition 要素で指定できるキーが含まれます。サービスのリソースに関連付けられている条件キーの詳細については、[リソースタイプ] テーブルの [条件キー] 列を参照してください。

Secrets Manager のアクション

アクション	説明	アクセスレベル	リソースタイプ (* 必須)	条件キー	依存アクション
BatchGetSecretValue	シークレットのリストを取得および復号化するためのアクセス許可を付与	リスト			
CancelRotateSecret	進行中のシークレットのローテーションをキャンセルするアクセス許可を付与します。	書き込み	Secret*	secretsmanager:SecretId secretsmanager:resource/AllowRotationLambdaAction secretsmanager:ResourceTag/tag-key aws:ResourceTag/\${TagKey} secretsmanager:Sec	

アクション	説明	アクセスレベル	リソースタイプ (* 必須)	条件キー	依存アクション
				retPrimaryRegion	
CreateSecret	クエリおよびローテーションが可能な暗号化されたデータを保存するシークレットを作成する許可を付与します。	書き込み	Secret*		

アクション	説明	アクセスレベル	リソースタイプ (* 必須)	条件キー	依存アクション
				secretsmanager:Name secretsmanager:Description secretsmanager:KmsKeyId aws:RequestTag/\${TagKey} aws:ResourceTag/\${TagKey} aws:TagKeys secretsmanager:ResourceTag/tag-key secretsmanager:AddReplicaRegions secretsmanager:For	

アクション	説明	アクセスレベル	リソースタイプ (* 必須)	条件キー	依存アクション
				ceOverwriteReplicaSecret	
DeleteResourcePolicy	シークレットに取り付けられたリソースポリシーを削除する許可を付与します。	経営へのアクセス権	Secret*	secretsmanager:SecretId secretsmanager:resource/AllowRotationLambdaAction secretsmanager:ResourceTag/tag-key aws:ResourceTag/\${TagKey} secretsmanager:SecretPrimaryRegion	
DeleteSecret	シークレットを削除するアクセス権限を付与します。	書き込み	Secret*		

アクション	説明	アクセスレベル	リソースタイプ (* 必須)	条件キー	依存アクション
				secretsmanager:SecretId secretsmanager:resource/AllowRotationLambdaArn secretsmanager:RecoveryWindowInDays secretsmanager:ForceDeleteWithoutRecovery secretsmanager:ResourceTag/tag-key aws:ResourceTag/\${TagKey} secretsmanager:Sec	

アクション	説明	アクセスレベル	リソースタイプ (* 必須)	条件キー	依存アクション
				retPrimaryRegion	
DescribeSecret	シークレットに関するメタデータを取得するアクセス許可を付与します。暗号化されたデータは取得できません	読み込み	Secret*	secretsmanager:SecretId secretsmanager:resource/AllowRotationLambdaArn secretsmanager:ResourceTag/tag-key aws:ResourceTag/\${TagKey} secretsmanager:SecretPrimaryRegion	
GetRandomPassword	パスワードの作成に使用するランダムな文字列を生成するアクセス許可を付与します。	読み込み			

アクション	説明	アクセスレベル	リソースタイプ (* 必須)	条件キー	依存アクション
GetResourcePolicy	シークレットに取り付けられているリソースポリシーを取得するアクセス許可を付与します。	読み込み	Secret*	secretsmanager:SecretId secretsmanager:resource/AllowRotationLambdaArn secretsmanager:ResourceTag/tag-key aws:ResourceTag/\${TagKey} secretsmanager:SecretPrimaryRegion	
GetSecretValue	暗号化されたデータを取得および復号化するアクセス許可を付与します。	読み込み	Secret*		

アクション	説明	アクセス レベル	リソース タイプ (* 必須)	条件キー	依存アク ション
				secretsmanager:SecretId secretsmanager:VersionId secretsmanager:VersionStage secretsmanager:resource/AllowRotationLambdaArn secretsmanager:ResourceTag/tag-key aws:ResourceTag/\${TagKey} secretsmanager:SecretPrimaryRegion	
ListSecretVersionIds		読み込み	Secret*		

アクション	説明	アクセスレベル	リソースタイプ (* 必須)	条件キー	依存アクション
	利用可能なシークレットのバージョンを一覧表示する許可を付与します。			secretsmanager:SecretId secretsmanager:resource/AllowRotationLambdaAction secretsmanager:ResourceTag/tag-key aws:ResourceTag/\${TagKey} secretsmanager:SecretPrimaryRegion	
ListSecrets	利用可能なシークレットを一覧表示するためのアクセス許可を付与します。	リスト			
PutResourcePolicy	リソースポリシーをシークレットに取り付ける許可を付与します。	経営へのアクセス権	Secret*		

アクション	説明	アクセスレベル	リソースタイプ (* 必須)	条件キー	依存アクション
				secretsmanager:SecretId secretsmanager:resource/AllowRotationLambdaArn secretsmanager:ResourceTag/tag-key aws:ResourceTag/\${TagKey} secretsmanager:BlockPublicPolicy secretsmanager:SecretPrimaryRegion	
PutSecretValue	<p>新しい暗号化されたデータで新しいバージョンのシークレットを作成するアクセス許可を付与します。</p>	書き込み	Secret*		

アクション	説明	アクセスレベル	リソースタイプ (* 必須)	条件キー	依存アクション
				secretsmanager:SecretId secretsmanager:resource/AllowRotationLambdaArn secretsmanager:ResourceTag/tag-key aws:ResourceTag/\${TagKey} secretsmanager:SecretPrimaryRegion	
RemoveReplicationsFromReplication	レプリケーションからリージョンを削除するアクセス許可を付与します。	書き込み	Secret*		

アクション	説明	アクセス レベル	リソース タイプ (* 必須)	条件キー	依存アク ション
				secretsmanager:SecretId secretsmanager:resource/AllowRotationLambdaArn secretsmanager:ResourceTag/tag-key aws:ResourceTag/\${TagKey} secretsmanager:SecretPrimaryRegion	

アクション	説明	アクセスレベル	リソースタイプ (* 必須)	条件キー	依存アクション
Replicate SecretToRegions	<p>既存のシークレットをマルチリージョンシークレットに変換し、新しいリージョンのリストへのシークレットのレプリケートを開始する許可を付与します。</p>	書き込み	Secret*	secretsmanager:SecretId secretsmanager:resource/AllowRotationLambdaAction secretsmanager:ResourceTag/tag-key aws:ResourceTag/\${TagKey} secretsmanager:SecretPrimaryRegion secretsmanager:AddReplicaRegions secretsmanager:ForceOverwrite	

アクション	説明	アクセスレベル	リソースタイプ (* 必須)	条件キー	依存アクション
				secretsmanager:Secret	
RestoreSecret	シークレットの削除をキャンセルするアクセス許可を付与します。	書き込み	Secret*	secretsmanager:SecretId secretsmanager:resource/AllowRotationLambdaAction secretsmanager:ResourceTag/tag-key aws:ResourceTag/\${TagKey} secretsmanager:SecretPrimaryRegion	
RotateSecret	シークレットのローテーションを開始するアクセス許可を付与します。	書き込み	Secret*		

アクション	説明	アクセス レベル	リソース タイプ (* 必須)	条件キー	依存アク ション
				secretsmanager:SecretId secretsmanager:RotationLambdaARN secretsmanager:resource/AllowRotationLambdaArn secretsmanager:ResourceTag/tag-key aws:ResourceTag/\${TagKey} secretsmanager:SecretPrimaryRegion secretsmanager:ModifyRotationRules	

アクション	説明	アクセスレベル	リソースタイプ (* 必須)	条件キー	依存アクション
				secretsmanager:RotateImmediately	
StopReplicationToReplica	レプリケーションからシークレットを削除し、レプリカリージョンのリージョンシークレットにシークレットを昇格する許可を付与します。	書き込み	Secret*	secretsmanager:SecretId secretsmanager:resource/AllowRotationLambdaAction secretsmanager:ResourceTag/tag-key aws:ResourceTag/\${TagKey} secretsmanager:SecretPrimaryRegion	
TagResource	シークレットにタグを追加する許可を付与します。	タグ付け	Secret*		

アクション	説明	アクセス レベル	リソース タイプ (* 必須)	条件キー	依存アク ション
				secretsma nager:Sec retId aws:Reque stTag/\${T agKey} aws:TagKe ys secretsma nager:res ource/All owRotatio nLambdaA n secretsma nager:Res ourceTag/ tag-key aws:Resou rceTag/\${ TagKey} secretsma nager:Sec retPrimar yRegion	
UntagReso urce	シークレットからタグを削除する許可を付与します。	タグ付け	Secret*		

アクション	説明	アクセスレベル	リソースタイプ (* 必須)	条件キー	依存アクション
				secretsmanager:SecretId aws:TagKeys secretsmanager:resource/AllowRotationLambdaArn secretsmanager:ResourceTag/tag-key aws:ResourceTag/\${TagKey} secretsmanager:SecretPrimaryRegion	
UpdateSecret	新しいメタデータもしくは新しいバージョンの暗号化データを使用してシークレットを更新する許可を付与します。	書き込み	Secret*		

アクション	説明	アクセス レベル	リソース タイプ (* 必須)	条件キー	依存アク ション
				secretsmanager:SecretId secretsmanager:Description secretsmanager:KmsKeyId secretsmanager:resource/AllowRotationLambdaArn secretsmanager:ResourceTag/tag-key aws:ResourceTag/\${TagKey} secretsmanager:SecretPrimaryRegion	

アクション	説明	アクセスレベル	リソースタイプ (* 必須)	条件キー	依存アクション
UpdateSecretVersionStage	あるシークレットから別のシークレットにステージを移動するアクセス許可を付与します。	書き込み	Secret*	secretsmanager:SecretId secretsmanager:VersionStage secretsmanager:resource/AllowRotationLambdaAction secretsmanager:ResourceTag/tag-key aws:ResourceTag/\${TagKey} secretsmanager:SecretPrimaryRegion	
ValidateResourcePolicy	ポリシーをアタッチする前にリソースポリシーを検証するアクセス許可を付与します。	経営へのアクセス権	Secret*		

アクション	説明	アクセスレベル	リソースタイプ (* 必須)	条件キー	依存アクション
				secretsmanager:SecretId secretsmanager:resource/AllowRotationLambdaArn secretsmanager:ResourceTag/tag-key aws:ResourceTag/\${TagKey} secretsmanager:SecretPrimaryRegion	

Secrets Manager リソース

リソースタイプ	ARN	条件キー
Secret	arn:\${Partition}:secretsmanager:\${Region}:\${Account}:secret:\${SecretId}	aws:RequestTag/\${TagKey}

リソースタイプ	ARN	条件キー
		aws:ResourceTag/\${TagKey} aws:TagKeys secretsmanager:ResourceTag/tag-key secretsmanager:resource/AllowRotationLambdaArn

Secrets Manager は、シークレットの名前の最後にダッシュと 6 つのランダムな英数字を加えることで、シークレット ARN の最後の部分を構成します。シークレットを削除した後に同じ名前の別のシークレットを作成すると、Secrets Manager が 6 つのランダムな文字を生成するため、この形式により、元のシークレットへのアクセス許可を持つユーザーが新しいシークレットに自動でアクセスできないようにすることができます。

シークレットの ARN は、シークレットの詳細ページにある Secrets Manager コンソールで、または [DescribeSecret](#) を呼び出すことで見つけることができます。

条件キー

以下の表の文字列条件をアクセス許可ポリシーに含める場合、Secrets Manager の呼び出し元は、一致するパラメータを渡す必要があります。渡さない場合、アクセスが拒否されます。欠落しているパラメータの呼び出し元を拒否しないようにするには、条件演算子名の末尾に `IfExists` を追加します (例: `StringLikeIfExists`)。詳細については、「[IAM JSON ポリシー要素: 条件演算子](#)」を参照してください。

条件キー	説明	[Type] (タイプ)
aws:ResourceTag/\${TagKey}	ユーザーが Secrets Manager サービスに対して行うリクエストに含まれるキーによってアクセスをフィルタリングします。	文字列

条件キー	説明	[Type] (タイプ)
aws:ResourceTag/\${TagKey}	リソースに関連付けられたタグでアクセスをフィルタリングします	文字列
aws:TagKeys	ユーザーが Secrets Manager サービスに行うリクエストに存在するすべてのタグキー名のリストに基づいて、アクセスをフィルタリングします。	ArrayOfString
secretsmanager:AddReplicaRegions	シークレットをレプリケートするリージョンのリストでアクセスをフィルタリングします。	ArrayOfString
secretsmanager:BlockPublicPolicy	リソースポリシーが広範なアクセスをブロックするかどうかで AWS アカウント アクセスをフィルタリングします	Bool
secretsmanager:Description	リクエスト内の記述テキストによるアクセスをフィルタリングします。	文字列
secretsmanager:ForceDeleteWithoutRecovery	シークレットがリカバリウィンドウなしですぐに削除されるかどうかによりアクセスをフィルタリングします。	Bool
secretsmanager:ForceOverwriteReplicaSecret	宛先リージョンで同じ名前のシークレットを上書きするかどうかでアクセスをフィルタリングします。	Bool
secretsmanager:KmsKeyId	リクエスト内の KMS キーの ARN によるアクセスをフィルタリングします。	文字列
secretsmanager:ModifyRotationRules	シークレットのローテーションルールを変更するかどうかに基づいて、アクセスをフィルタリングします。	Bool

条件キー	説明	[Type] (タイプ)
secretsmanager:Name	リクエスト内のシークレットのフレンドリー名によるアクセスをフィルタリングします。	文字列
secretsmanager:RecoveryWindowInDays	Secrets Manager がシークレットを削除するまで待機する日数でアクセスをフィルタリングします。	数値
secretsmanager:ResourceTag/tag-key	タグキーおよび値のペアでアクセスをフィルタリングします。	文字列
secretsmanager:RotateImmediately	シークレットを直ちにローテーションするかどうかに基づいて、アクセスをフィルタリングします。	Bool
secretsmanager:RotationLambdaARN	リクエスト内のローテーション Lambda 関数の ARN によるアクセスをフィルタリングします。	ARN
secretsmanager:SecretId	リクエスト内の SecretID 値によるアクセスをフィルタリングします。	ARN
secretsmanager:SecretPrimaryRegion	シークレットが作成されるプライマリリージョンでアクセスをフィルタリングします。	文字列
secretsmanager:VersionId	リクエスト内のシークレットのバージョンの唯一の識別子でアクセスをフィルタリングします。	文字列
secretsmanager:VersionStage	リクエスト内のバージョンステージのリストでアクセスをフィルタリングします。	文字列

条件キー	説明	[Type] (タイプ)
secretsmanager:resource/AllowRotationLambdaArn	シークレットに関連するローテーション Lambda 関数の ARN によるアクセスをフィルタリングします。	ARN

BlockPublicPolicy 条件を利用したシークレットへの広範なアクセスのブロック

アクション `PutResourcePolicy` を許可するアイデンティティポリシーでは、`BlockPublicPolicy: true` を使用することをお勧めします。この条件は、ポリシーが広範なアクセスを許可していない場合、ユーザーがリソースポリシーのみをシークレットにアタッチできることを意味します。

Secrets Manager は、Zelkova の自動推論を使用して、広範なアクセスのためのリソースポリシーを分析します。Zelkova の詳細については、AWS セキュリティブログの [「が自動推論 AWS を使用して大規模なセキュリティを実現する方法」](#) を参照してください。

次の例は、BlockPublicPolicy の使用方法を示しています。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "secretsmanager:PutResourcePolicy",
    "Resource": "SecretId",
    "Condition": {
      "Bool": {
        "secretsmanager:BlockPublicPolicy": "true"
      }
    }
  }
}
```

IP アドレス条件

Secrets Manager へのアクセスを許可または拒否するポリシーステートメントで、[IP アドレス条件の演算子](#)または `aws:SourceIp` 条件キーを指定するときは、注意が必要です。例えば、企業ネット

ワークの IP アドレス範囲からのリクエストに AWS アクションを制限するポリシーをシークレットにアタッチすると、企業ネットワークからリクエストを呼び出す IAM ユーザーとしてのリクエストは期待どおりに機能します。ただし、Lambda 関数でローテーションを有効にする場合など、他のサービスがユーザーに代わってシークレットにアクセスできるようにすると、その関数は AWS-内部アドレス空間から Secrets Manager オペレーションを呼び出します。IP アドレスのフィルターがあるポリシーによって影響されたリクエストは失敗します。

また、リクエストが Amazon VPC エンドポイントから送信されている場合、`aws:sourceIP` 条件キーは効果が低下します。特定の VPC エンドポイントに対するリクエストを制限するには、[the section called “VPC エンドポイントの条件”](#) を使用します。

VPC エンドポイントの条件

特定の VPC または VPC エンドポイントからのリクエストに対するアクセスを許可または拒否するには、`aws:SourceVpc` を使用して、指定された VPC からのリクエストに対するアクセスを制限するか、`aws:SourceVpcce` を使用して、指定された VPC エンドポイントからのリクエストに対するアクセスを制限します。「[the section called “例: アクセス許可と VPC”](#)」を参照してください。

- `aws:SourceVpc` は、指定した VPC からのリクエストにアクセスを制限します。
- `aws:SourceVpcce` は、指定した VPC エンドポイントからのリクエストにアクセスを制限します。

これらの条件キーをシークレットポリシーステートメントで使用し、Secrets Manager シークレットへのアクセスを許可または拒否すると、ユーザーに代わってシークレットへのアクセスに Secrets Manager を使用しているサービスへのアクセスを、意図せずに拒否してしまうことがあります。VPC 内のエンドポイントで実行できる AWS のサービスは、一部のみです。シークレットのリクエストを VPC または VPC エンドポイントに制限すると、サービスに設定されていないサービスからの Secrets Manager への呼び出しは失敗します。

「[VPC エンドポイント](#)」を参照してください。

でシークレットを作成および管理 AWS Secrets Manager

シークレットは、パスワード、ユーザーネームやパスワードなどの一連の認証情報、OAuth トークン、または、暗号化された形式で Secrets Manager に保存されるその他のシークレット情報にすることができます。

トピック

- [AWS Secrets Manager データベースシークレットを作成する](#)
- [AWS Secrets Manager シークレットの JSON 構造](#)
- [AWS Secrets Manager シークレットを作成する](#)
- [AWS Secrets Manager シークレットの値を更新する](#)
- [Secrets Manager でパスワードを生成する](#)
- [AWS Secrets Manager シークレットの暗号化キーを変更](#)
- [AWS Secrets Manager シークレットの変更](#)
- [でシークレットを検索する AWS Secrets Manager](#)
- [AWS Secrets Manager シークレットを削除する](#)
- [AWS Secrets Manager シークレットを復元する](#)
- [AWS Secrets Manager シークレットにタグ付けする](#)

AWS Secrets Manager データベースシークレットを作成する

Amazon RDS、Amazon Aurora、Amazon Redshift、または Amazon DocumentDB でユーザーを作成したら、次の手順に従って、それらの認証情報を Secrets Manager に保存することができます。AWS CLI またはいずれかの SDKs を使用してシークレットを保存する場合は、[正しい JSON 構造](#) でシークレットを指定する必要があります。コンソールを使用してデータベースシークレットを保存すると、Secrets Manager は正しい JSON 構造で自動的に作成します。

Tip

Amazon RDS および Amazon Redshift 管理者ユーザーの認証情報には、[マネージドシークレット](#) を使用することをお勧めします。マネージドシークレットは管理サービスを通じて作成し、[マネージドローテーション](#) を使用できます。

他のリージョンにレプリケートされているソースデータベースのデータベース認証情報を保存すると、シークレットにはソースデータベースの接続情報が含まれます。その後、シークレットをレプリケートすると、レプリカはソースシークレットのコピーとなり、同じ接続情報が含まれます。リージョン接続情報のシークレットにキー/値ペアを追加できます。

シークレットを作成するには、SecretsManagerReadWrite によって付与されたアクセス許可が必要です [AWS 管理ポリシー](#)。

Secrets Manager は、シークレットの作成時に CloudTrail ログエントリを生成します。詳細については、「[the section called “でログ記録する AWS CloudTrail”](#)」を参照してください。

シークレットを作成するには (コンソール)

1. Secrets Manager のコンソール (<https://console.aws.amazon.com/secretsmanager/>) を開きます。
2. [Store a new secret] (新しいシークレットを保存する) を選択します。
3. [Choose secret type] (シークレットタイプを選択する) ページで、次の操作を行います。
 - a. [シークレットタイプ] で、保存するデータベース認証情報のタイプを選択します。
 - Amazon RDS データベース (Aurora を含む)
 - Amazon DocumentDB データベース
 - Amazon Redshift データウェアハウス
 - b. 認証情報を使用する場合、データベースの認証情報を入力します。
 - c. 暗号化キー で、Secrets Manager AWS KMS key がシークレット値を暗号化するために使用する を選択します。詳細については、「[シークレット暗号化と復号](#)」を参照してください。
 - 多くの場合、Secrets Manager に AWS マネージドキー を使用するときは、aws/secretsmanager を選択します。このキーを使用してもコストは発生しません。
 - 別の からシークレットにアクセスする必要がある場合 AWS アカウント、または独自の KMS キーを使用してローテーションしたりキーポリシーを適用したりする場合は、リストからカスターマネージドキーを選択するか、新しいキーを追加を選択して作成します。カスターマネージドキーの使用料金の詳細については、「[料金](#)」を参照してください。

必要なもの: [the section called “KMS キーのアクセス許可”](#) クロスアカウントアクセスの詳細については、「[the section called “クロスアカウントアクセス”](#)」を参照してください。

- d. [Database] (データベース) で、データベースを選択します。
 - e. [次へ] をクリックします。
4. [Configure secret] (シークレットを設定する) ページで、次の操作を行います。
- a. わかりやすいシークレット名と説明を入力します。シークレット名は 1~512 文字の Unicode 文字で構成されます。
 - b. (オプション) [Tags] (タグ) セクションで、タグをシークレットに追加します。タグ付け戦略については、「[the section called “シークレットにタグ付けする”](#)」を参照してください。機密情報は暗号化されていないため、タグに保存しないでください。
 - c. (オプション) [Resource permissions] (リソースに対するアクセス許可) でリソースポリシーをシークレットに追加するには、[Edit permissions] (アクセス許可の編集) をクリックします。詳細については、「[the section called “アクセス許可ポリシーをシークレットにアタッチする”](#)」を参照してください。
 - d. (オプション) シークレットのレプリケートで、シークレットを別の にレプリケートするには AWS リージョン、シークレットのレプリケートを選択します。シークレットのレプリケーションは、この段階で実行することも、後に戻ってきて実行することもできます。詳細については、「[リージョン間でシークレットをレプリケートする](#)」を参照してください。
 - e. [Next] (次へ) を選択します。
5. (オプション) [Configure rotation] (ローテーションを設定する) ページで、シークレットの自動ローテーションを有効にできます。ローテーションをオフにしておいて、後でオンにすることもできます。詳しくは、「[シークレットのローテーション](#)」を参照してください。[Next] (次へ) を選択します。
6. [Review] (レビュー) ページで、シークレットの詳細を確認し、[Store] (保存) を選択します。

Secrets Manager はシークレットのリストに戻ります。新しいシークレットが表示されない場合は、更新ボタンを選択します。

AWS CLI

コマンドシェルにコマンドを入力すると、コマンド履歴がアクセスされたり、ユーティリティからコマンドパラメータにアクセスされたりするリスクがあります。[the section called “AWS CLI を使用して AWS Secrets Manager のシークレットを保存する際のリスクを軽減する”](#) を参照してください。

Example JSON ファイルの認証情報からシークレットを作成する

次の [create-secret](#) の例では、ファイル内の認証情報からシークレットを作成しています。詳細については、「ユーザーガイド」の「[ファイルからの AWS CLI パラメータのロード AWS CLI](#)」を参照してください。

Secrets Manager がシークレットをローテーションできるようにするには、必ず JSON が [シークレットの JSON 構造](#) にマッチしている必要があります。

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

mycreds.json の内容:

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

AWS SDK

AWS SDKs [CreateSecret](#) アクションを使用します。詳細については、「[the section called “AWS SDKs”](#)」を参照してください。

AWS Secrets Manager シークレットの JSON 構造

Secrets Manager シークレットには、任意のテキストまたはバイナリを保存できます。Secrets Manager のシークレットで自動ローテーションをオンにする場合、そのシークレットは正しい JSON 構造である必要があります。ローテーション中、Secrets Manager はシークレット内の情報を

使用して認証情報ソースに接続し、そこにある認証情報を更新します。JSON キー名では、大文字と小文字が区別されます。

コンソールを使用してデータベースシークレットを保存すると、Secrets Manager が自動的にシークレットを正しい JSON 構造で作成します。

データベースシークレットなどのシークレットにキーと値のペアを追加して、他のリージョンのレプリカデータベースの接続情報を含めることができます。

トピック

- [Amazon RDS Db2 シークレット構造](#)
- [Amazon RDS MariaDB シークレット構造](#)
- [Amazon RDS と Amazon Aurora MySQL のシークレット構造](#)
- [Amazon RDS Oracle シークレット構造](#)
- [Amazon RDS と Amazon Aurora PostgreSQL のシークレット構造](#)
- [Amazon RDS Microsoft SQLServer シークレット構造](#)
- [Amazon DocumentDB シークレット構造](#)
- [Amazon Redshift のシークレット構造](#)
- [Amazon Redshift Serverless シークレット構造](#)
- [Amazon ElastiCache シークレット構造](#)
- [Active Directory シークレット構造](#)

Amazon RDS Db2 シークレット構造

Amazon RDS Db2 インスタンスの場合、ユーザーは自分のパスワードを変更できないため、管理者の認証情報を別のシークレットで提供する必要があります。

```
{
  "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": "<the ARN of the elevated secret>"
}
```

Amazon RDS MariaDB シークレット構造

```
{
  "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>
}
```

を使用するには[the section called “交代ユーザー”](#)、管理者またはスーパーユーザーの認証情報を含むシークレットmasterarnに を含めます。

```
{
  "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": "<the ARN of the elevated secret>"
}
```

Amazon RDS と Amazon Aurora MySQL のシークレット構造

```
{
  "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>
}
```

を使用するには[the section called “交代ユーザー”](#)、管理者またはスーパーユーザーの認証情報を含むシークレットmasterarnに を含めます。

```
{
  "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": "<the ARN of the elevated secret>"
}
```

Amazon RDS Oracle シークレット構造

```
{
  "engine": "oracle",
  "host": "<required: instance host name/resolvable DNS name>",
  "username": "<required: username>",
  "password": "<required: password>",
  "dbname": "<required: database name>",
  "port": <optional: TCP port number. If not specified, defaults to 1521>
}
```

を使用するには[the section called “交代ユーザー”](#)、管理者またはスーパーユーザーの認証情報を含むシークレットmasterarnに を含めます。

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

Amazon RDS と Amazon Aurora PostgreSQL のシークレット構造

```
{
  "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>
}
```

を使用するには[the section called “交代ユーザー”](#)、管理者またはスーパーユーザーの認証情報を含むシークレットmasterarnに を含めます。

```
{
  "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": "<the ARN of the elevated secret>"
}
```

Amazon RDS Microsoft SQLServer シークレット構造

```
{
  "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>
}
```

を使用するには[the section called “交代ユーザー”](#)、管理者またはスーパーユーザーの認証情報を含むシークレットmasterarnに を含めます。

```
{
  "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": "<the ARN of the elevated secret>"
}
```

Amazon DocumentDB シークレット構造

```
{
  "engine": "mongo",
```

```
"host": "<i><instance host name/resolvable DNS name></i>",
"username": "<i><username></i>",
"password": "<i><password></i>",
"dbname": "<i><database name. If not specified, defaults to None></i>",
"port": "<i><TCP port number. If not specified, defaults to 27017></i>",
"ssl": "<i><true/false. If not specified, defaults to false></i>"
}
```

を使用するには[the section called “交代ユーザー”](#)、管理者またはスーパーユーザーの認証情報を含むシークレットmasterarnに を含めます。

```
{
  "engine": "mongo",
  "host": "<i><instance host name/resolvable DNS name></i>",
  "username": "<i><username></i>",
  "password": "<i><password></i>",
  "dbname": "<i><database name. If not specified, defaults to None></i>",
  "port": "<i><TCP port number. If not specified, defaults to 27017></i>",
  "masterarn": "<i><the ARN of the elevated secret></i>",
  "ssl": "<i><true/false. If not specified, defaults to false></i>"
}
```

Amazon Redshift のシークレット構造

```
{
  "engine": "redshift",
  "host": "<i><instance host name/resolvable DNS name></i>",
  "username": "<i><username></i>",
  "password": "<i><password></i>",
  "dbname": "<i><database name. If not specified, defaults to None></i>",
  "port": "<i><TCP port number. If not specified, defaults to 5439></i>"
}
```

を使用するには[the section called “交代ユーザー”](#)、管理者またはスーパーユーザーの認証情報を含むシークレットmasterarnに を含めます。

```
{
  "engine": "redshift",
  "host": "<i><instance host name/resolvable DNS name></i>",
  "username": "<i><username></i>",
  "password": "<i><password></i>",
```

```
"dbname": "<database name. If not specified, defaults to None>",
"port": <TCP port number. If not specified, defaults to 5439>,
"masterarn": "<the ARN of the elevated secret>"
}
```

Amazon Redshift Serverless シークレット構造

```
{
  "engine": "redshift",
  "host": "<instance host name/resolvable DNS name>",
  "username": "<username>",
  "password": "<password>",
  "dbname": "<database name. If not specified, defaults to None>",
  "namespaceName": <namespace name>,
  "port": <TCP port number. If not specified, defaults to 5439>
}
```

を使用するには[the section called “交代ユーザー”](#)、管理者またはスーパーユーザーの認証情報を含むシークレットmasterarnに を含めます。

```
{
  "engine": "redshift",
  "host": "<instance host name/resolvable DNS name>",
  "username": "<username>",
  "password": "<password>",
  "dbname": "<database name. If not specified, defaults to None>",
  "namespaceName": <namespace name>,
  "port": <TCP port number. If not specified, defaults to 5439>,
  "masterarn": "<the ARN of the elevated secret>"
}
```

Amazon ElastiCache シークレット構造

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

詳細については、「Amazon ユーザーガイド」の「[ユーザーのパスワードの自動ローテーション](#)」を参照してください。 ElastiCache

Active Directory シークレット構造

AWS Directory Service はシークレットを使用して Active Directory 認証情報を保存します。詳細については、AWS Directory Service 管理ガイドの[Amazon EC2 Linux インスタンスを Managed AD Active Directory にシームレスに結合する](#)」を参照してください。シームレスなドメイン結合には、次の例のキー名が必要です。シームレスなドメイン結合を使用しない場合は、ローテーション関数のテンプレートコードで説明されているように、環境変数を使用してシークレット内のキーの名前を変更できます。

Active Directory シークレットをローテーションするには、[Active Directory ローテーションテンプレート](#)を使用できます。

Active Directory 認証情報シークレット構造

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

シークレットをローテーションする場合は、ドメインディレクトリ ID を含めます。

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

シークレットをキータブを含むシークレットと組み合わせて使用する場合は、キータブシークレット ARNs。

```
{
  "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 のキータブシークレット構造

キータブファイルを使用して Amazon EC2 の Active Directory アカウントを認証する方法については、[「Amazon Linux 2 での SQL Server 2017 を使用した Active Directory 認証のデプロイと設定」](#)を参照してください。

```
{  
  "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  
}
```

AWS Secrets Manager シークレットを作成する

API キー、アクセストークン、データベース用ではない認証情報、およびその他のシークレットを Secrets Manager に保存するには、次の手順を実行します。Amazon ElastiCache シークレットでローテーションを有効にする場合は、シークレットを[期待される JSON 構造](#)に保存する必要があります。

シークレットを作成するには、SecretsManagerReadWrite によって付与されたアクセス許可が必要です[AWS 管理ポリシー](#)。

Secrets Manager は、シークレットの作成時に CloudTrail ログエントリを生成します。詳細については、「[the section called “でログ記録する AWS CloudTrail ”](#)」を参照してください。

シークレットを作成するには (コンソール)

1. Secrets Manager のコンソール (<https://console.aws.amazon.com/secretsmanager/>) を開きます。

2. [Store a new secret] (新しいシークレットを保存する) を選択します。
3. [Choose secret type] (シークレットタイプを選択する) ページで、次の操作を行います。
 - a. [Secret type] (シークレットタイプ) で、[Other type of secret] (他の種類のシークレット) を選択します。
 - b. [キーと値のペア] に、JSON の キーと値 ペアでシークレットを入力するか、[プレーンテキスト] タブを開き、任意の形式でシークレットを入力します。シークレットには最大 65536 バイトまで保存できます。例:

API キーのキーと値のペア:

ClientID: *client_id*

ClientSecret : *wJalrXUtnFEMI /K7MDENG/bPxRfiCYEXAMPLEKEY*

認証情報キーと値のペア:

Username: *Saanvis*

Password: *# - #####*

OAuth トークンの平文:

AKIAI44QH8DHB#

デジタル証明書平文:

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

平文のプライベートキー

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

- c. 暗号化キーで、Secrets Manager AWS KMS key がシークレット値を暗号化するために使用する を選択します。詳細については、「[シークレット暗号化と復号](#)」を参照してください。
 - 多くの場合、Secrets Manager に AWS マネージドキー を使用するときは、aws/secretsmanager を選択します。このキーを使用してもコストは発生しません。
 - 別の からシークレットにアクセスする必要がある場合 AWS アカウント、または独自の KMS キーを使用してローテーションしたりキーポリシーを適用したりする場合は、リストからカスタマーマネージドキーを選択するか、新しいキーを追加を選択して作成します。カスタマーマネージドキーの使用料金の詳細については、「[料金](#)」を参照してください。

必要なもの: [the section called “KMS キーのアクセス許可”](#) クロスアカウントアクセスの詳細については、「[the section called “クロスアカウントアクセス”](#)」を参照してください。
 - d. [次へ] をクリックします。
4. [Configure secret] (シークレットを設定する) ページで、次の操作を行います。
 - a. わかりやすいシークレット名と説明を入力します。シークレット名は 1~512 文字の Unicode 文字で構成されます。
 - b. (オプション) [Tags] (タグ) セクションで、タグをシークレットに追加します。タグ付け戦略については、「[the section called “シークレットにタグ付けする”](#)」を参照してください。機密情報は暗号化されていないため、タグに保存しないでください。
 - c. (オプション) [Resource permissions] (リソースに対するアクセス許可) でリソースポリシーをシークレットに追加するには、[Edit permissions] (アクセス許可の編集) をクリックします。詳細については、「[the section called “アクセス許可ポリシーをシークレットにアタッチする”](#)」を参照してください。
 - d. (オプション) シークレットのレプリケートで、シークレットを別の にレプリケートするには AWS リージョン、シークレットのレプリケート を選択します。シークレットのレプリケーションは、この段階で実行することも、後に戻ってきて実行することもできます。詳細については、「[リージョン間でシークレットをレプリケートする](#)」を参照してください。
 - e. [Next] (次へ) を選択します。
 5. (オプション) [Configure rotation] (ローテーションを設定する) ページで、シークレットの自動ローテーションを有効にできます。ローテーションをオフにしておいて、後でオンにすることもできます。詳しくは、「[シークレットのローテーション](#)」を参照してください。[Next] (次へ) を選択します。

6. [Review] (レビュー) ページで、シークレットの詳細を確認し、[Store] (保存) を選択します。

Secrets Manager はシークレットのリストに戻ります。新しいシークレットが表示されない場合は、更新ボタンを選択します。

AWS CLI

コマンドシェルにコマンドを入力すると、コマンド履歴がアクセスされたり、ユーティリティからコマンドパラメータにアクセスされたりするリスクがあります。[the section called “AWS CLI を使用して AWS Secrets Manager のシークレットを保存する際のリスクを軽減する”](#) を参照してください。

Example シークレットを作成する

次に、2 つのキーと値のペアを持つシークレットを作成する、[create-secret](#) の例をします。

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

Example JSON ファイルの認証情報からシークレットを作成する

次の [create-secret](#) の例では、ファイル内の認証情報からシークレットを作成しています。詳細については、「[ユーザーガイド](#)」の「[ファイルからの AWS CLI パラメータのロード](#) AWS CLI」を参照してください。

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

mycreds.json の内容:

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

AWS SDK

AWS SDKs [CreateSecret](#) アクションを使用します。詳細については、「[the section called “AWS SDKs”](#)」を参照してください。

AWS Secrets Manager シークレットの値を更新する

シークレットの値を更新するには、コンソール、CLI、または SDK を使用できます。シークレット値を更新すると、Secrets Manager は、ステージングラベル AWSCURRENT 付きのシークレットの新しいバージョンを作成します。ラベル AWSPREVIOUS の付いた古いバージョンには引き続きアクセスできます。独自のラベルを追加することもできます。詳細については、「[Secrets Manager のバージョンング](#)」を参照してください。

シークレット値を更新するには (コンソール)

1. Secrets Manager のコンソール (<https://console.aws.amazon.com/secretsmanager/>) を開きます。
2. シークレットのリストから、自分のシークレットを選択します。
3. シークレットの詳細ページの [概要] タブの [シークレットの値] セクションで、[シークレットの値を取得する] を選択し、[編集] を選択します。

AWS CLI

シークレット値を更新するには (AWS CLI)

- コマンドシェルにコマンドを入力すると、コマンド履歴がアクセスされたり、ユーティリティからコマンドパラメータにアクセスされたりするリスクがあります。「[the section called “AWS CLI を使用して AWS Secrets Manager のシークレットを保存する際のリスクを軽減する”](#)」を参照してください。

次の [put-secret-value](#) は、キーと値のペア 2 つを含むシークレットの新しいバージョンを作成します。

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

次の [put-secret-value](#) は、カスタムステージラベル付きの新しいバージョンを作成します。新しいバージョンには、MyLabel と AWSCURRENT のラベルが付けられます。

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

AWS SDK

10 分に 1 回以上の持続頻度で PutSecretValue または UpdateSecret を呼び出すことは避けることが推奨されます。PutSecretValue または UpdateSecret を呼び出してシークレット値を更新すると、Secrets Manager はシークレットの新しいバージョンを作成します。Secrets Manager は、ラベルのないバージョンが 100 を超えると削除しますが、24 時間以内に作成されたバージョンは削除しません。10 分に 1 回以上の頻度でシークレット値を更新すると、Secrets Manager が削除した数よりも多くバージョンが作成され、シークレットバージョンのクォータに達します。

シークレット値を更新するには、[UpdateSecret](#) アクションまたは [PutSecretValue](#) アクションを使用します。詳細については、「[the section called “AWS SDKs”](#)」を参照してください。

Secrets Manager でパスワードを生成する

Secrets Manager を使用する一般的なパターンは、Secrets Manager でパスワードを生成し、そのパスワードをデータベースまたはサービスで使用するということです。これを行うには、次の方法を使用します。

- AWS CloudFormation – 「」を参照してください [AWS CloudFormation](#)。
- AWS CLI – 「」を参照してください [get-random-password](#)。
- AWS SDKs 「」を参照してください [GetRandomPassword](#)。

AWS Secrets Manager シークレットの暗号化キーを変更

Secrets Manager は、[AWS KMS キーとデータキーによるエンベロープ暗号化](#)を使用して各シークレット値を保護します。シークレットごとに使用する KMS キーを選択できます。AWS マネージドキー aws/secretsmanager を使用することも、顧客が管理するキーを使用することもできます。

ほとんどのケースでは、aws/secretsmanager の使用をお勧めします。利用料金は発生しません。別のシークレットからシークレットにアクセスする必要がある場合や AWS アカウント、独自の KMS キーを使用してそのキーをローテーションしたり、キーポリシーを適用したりしたい場合は、を使用してください。カスタマー管理キー必要なもの: [the section called “KMS キーのアクセス許可”](#) カスタマーマネージドキーの使用料金の詳細については、「[料金](#)」を参照してください。

シークレットの暗号化キーを変更できます。たとえば、[別のアカウントからシークレットにアクセスしたいときに](#)、AWS そのシークレットが現在マネージドキーを使用して暗号化されている場合はaws/secretsmanager、に切り替えることができます。カスタマー管理キー

i Tip

をローテーションしたい場合は カスタマー管理キー、AWS KMS 自動キーローテーションを使用することをおすすめします。詳細については、「[AWS KMS キーのローテーション](#)」を参照してください。

暗号化キーを変更すると、Secrets Manager はAWSCURRENT、AWSPENDING、AWSPREVIOUSを新しいキーで再暗号化します。シークレットからロックアウトされないように、SecSecrets Manager は既存のすべてのバージョンを以前のキーで暗号化したままにします。つまり、以前のキーまたは新しいキーを使用してAWSCURRENTAWSPENDING、、、AWSPREVIOUSのバージョンを復号化できます。

AWSCURRENT新しい暗号化キーでのみ復号化できるようにするには、新しいキーを使用して新しいバージョンのシークレットを作成します。その後、AWSCURRENTシークレットバージョンを復号化できるようにするには、新しいキーに対する権限が必要です。

以前の暗号化キーを非アクティブ化すると、AWSCURRENT、AWSPENDING、AWSPREVIOUS 以外のシークレットバージョンを復号できなくなります。アクセスを保持する必要がある他のラベル付きシークレットバージョンがある場合、[the section called “AWS CLI”](#) を使用して新しい暗号化キーでそのバージョンを再作成する必要があります。

シークレットの暗号化キーを変更するには (コンソール)

1. Secrets Manager のコンソール (<https://console.aws.amazon.com/secretsmanager/>) を開きます。
2. シークレットのリストから、自分のシークレットを選択します。
3. シークレットの詳細ページの [シークレットの詳細] セクションで、[アクション] を選択して [暗号化キーの編集] を選択します。

AWS CLI

シークレットの暗号化キーを変更した後で以前の暗号化キーを非アクティブ化すると、AWSCURRENT、AWSPENDING、AWSPREVIOUS 以外のシークレットバージョンを復号できなくなります。アクセスを保持する必要がある他のラベル付きシークレットバージョンがある場合、[the section called “AWS CLI”](#) を使用して新しい暗号化キーでそのバージョンを再作成する必要があります。

シークレットの暗号化キーを変更するには (AWS CLI)

1. 次の [update-secret](#) の例は、シークレット値の暗号化に使用される KMS キーを更新します。KMS キーは、シークレットと同じリージョンに存在する必要があります。

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

2. (オプション) カスタムラベルの付いたシークレットバージョンがある場合、新しいキーを使用してそれらを再暗号化するには、そのバージョンを再作成する必要があります。

コマンドシェルにコマンドを入力すると、コマンド履歴がアクセスされたり、ユーティリティからコマンドパラメータにアクセスされたりするリスクがあります。[the section called “AWS CLI を使用して AWS Secrets Manager のシークレットを保存する際のリスクを軽減する”](#) を参照してください。

- a. シークレットバージョンの値を取得します。

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

シークレット値を書き留めておきます。

- b. その値で新しいバージョンを作成します。

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


AWS Secrets Manager シークレットの変更

シークレットの作成者に応じて、そのシークレットの作成後にそのメタデータを変更できます。他のサービスによって作成されたシークレットについては、他のサービスを使用して更新またはローテーションする必要がある場合があります。

シークレットを管理しているユーザーを特定するには、シークレット名を確認します。他のサービスによって管理されるシークレットには、そのサービスの ID がプレフィックスとして付けられます。または、で AWS CLI `describe-secret` を呼び出し、フィールドを確認します `OwningService`。詳細については、「[マネージドシークレット](#)」を参照してください。

管理するシークレットについては、説明、リソースベースのポリシー、暗号化キー、およびタグを変更できます。また、暗号化されたシークレット値を変更することもできますが、ローテーションを使用して認証情報を含むシークレット値を更新することが推奨されます。ローテーションによって、Secrets Manager のシークレットと、データベースまたはサービスの認証情報の両方が更新されます。これによりシークレットが自動的に同期されるため、クライアントがシークレット値をリクエストしたとき、常に有効な認証情報を取得できます。詳細については、「[シークレットのローテーション](#)」を参照してください。

Secrets Manager は、シークレットを変更すると CloudTrail ログエントリを生成します。詳細については、「[the section called “でログ記録する AWS CloudTrail”](#)」を参照してください。

管理するシークレットを更新するには (コンソール)

1. Secrets Manager のコンソール (<https://console.aws.amazon.com/secretsmanager/>) を開きます。
2. シークレットのリストから、自分のシークレットを選択します。
3. シークレットの詳細ページで、次のいずれかの操作を実行します。

シークレットの名前や ARN は変更できないことに注意してください。

- 説明を更新するには、[Secrets details] (シークレットの詳細) セクションで [Actions] (アクション) を選択し、[Edit description] (説明の編集) を選択します。
- 暗号化キーを更新するには、「[the section called “シークレットの暗号化キーを変更する”](#)」を参照してください。
- タグを更新するには、[タグ] タブで [タグを編集] を選択します。[the section called “シークレットにタグ付けする”](#) を参照してください。
- シークレット値を更新するには、「[the section called “シークレット値の更新”](#)」を参照してください。

- シークレットの許可を更新するには、[概要] タブで、[許可を編集] を選択します。 [the section called “アクセス許可ポリシーをシークレットにアタッチする”](#) を参照してください。
- シークレットのローテーションを更新するには、[ローテーション] タブで [ローテーションを編集] を選択します。 [シークレットのローテーション](#) を参照してください。
- シークレットを他のリージョンにレプリケーションするには、「[リージョン間でシークレットをレプリケートする](#)」を参照してください。
- シークレットにレプリカがある場合は、レプリカの暗号化キーを変更できます。[レプリケーション] タブで、レプリカのラジオボタンを選択し、[アクション] メニューで、[暗号化キーを編集] を選択します。 [the section called “シークレット暗号化と復号”](#) を参照してください。
- シークレットを別のサービスによって管理されるように変更するには、そのサービスでシークレットを再作成する必要があります。 [マネージドシークレット](#) を参照してください。

AWS CLI

Example シークレットの説明を更新する

次の [update-secret](#) の例では、シークレットの説明が更新されます。

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

AWS SDK

10 分に 1 回以上の持続頻度で PutSecretValue または UpdateSecret を呼び出すことは避けることが推奨されます。PutSecretValue または UpdateSecret を呼び出してシークレット値を更新すると、Secrets Manager はシークレットの新しいバージョンを作成します。Secrets Manager は、ラベルのないバージョンが 100 を超えると削除しますが、24 時間以内に作成されたバージョンは削除しません。10 分に 1 回以上の頻度でシークレット値を更新すると、Secrets Manager が削除した数よりも多くバージョンが作成され、シークレットバージョンのクォータに達します。

シークレットを更新するには、[UpdateSecret](#) アクションまたは [ReplicateSecretToRegions](#) アクションを使用します。詳細については、「[the section called “AWS SDKs”](#)」を参照してください。

でシークレットを検索する AWS Secrets Manager

フィルタなしでシークレットを検索すると、Secrets Manager はシークレット名、説明、タグキー、およびタグ値のキーワードを照合します。フィルタなしで検索すると、大文字と小文字は区別されず、スペース、/、_、=、# などの特殊文字は無視され、数字と文字のみが使用されます。フィルタなしで検索すると、Secrets Manager は検索文字列を分析して個別の単語に変換します。単語は、大文字から小文字、文字から数字、または数字/文字から句読点への変更によって区切られます。たとえば、名前、説明、およびタグのキーと値で、creds、Database、および 892 を検索する検索語 credsDatabase#892 を入力するとします。

Secrets Manager は、シークレットを一覧表示するときに CloudTrail ログエントリを生成します。詳細については、「[the section called “でログ記録する AWS CloudTrail”](#)」を参照してください。

検索に次のフィルタを適用できます。

名前

シークレット名の先頭に一致します。大文字と小文字は区別されます。たとえば、名前: **Data** は、databaseSecret でも MyData でもない DatabaseSecret という名前のシークレットを返します。

説明

シークレットの説明内の単語に一致します。大文字と小文字は区別されません。たとえば、説明: **My Description** は、シークレットを次の説明で照合します。

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

によって管理されます

や など AWS、 の外部でサービスによって管理されるシークレット CyberArk を検索します HashiCorp。

所有しているサービス

管理サービスの ID プレフィックスの先頭に一致します。大文字と小文字は区別されません。例えば、**my-ser** は、プレフィックス my-serv および my-service を持つサービスが管理するシークレットと一致します。詳細については、「[マネージドシークレット](#)」を参照してください。

レプリケート:

プライマリシークレット、レプリカシークレット、またはレプリケートされないシークレットをフィルタリングできます。

タグキー

タグキーの先頭に一致します。大文字と小文字は区別されます。たとえば、タグキー: **Prod** は、タグ Production そして Prod1 付きのシークレットを返しますが、タグ prod または 1 Prod 付きのシークレットは返しません。

タグ値

タグ値の先頭に一致します。大文字と小文字は区別されます。たとえば、タグ値: **Prod** はタグ Production そして Prod1 付きのシークレットを返しますが、タグ値 prod または 1 Prod を持つシークレットは返しません。

Secrets Manager はリージョンのサービスであり、検索は、選択されたリージョンのシークレットのみを返します。

AWS CLI

Example アカウントにあるシークレットを一覧表示する

以下の [list-secrets](#) 例は、アカウント内にあるシークレットの一覧を取得します。

```
aws secretsmanager list-secrets
```

Example アカウントにあるシークレットの一覧をフィルタリングする

次の [list-secrets](#) の例は、アカウント内にあり、名前に Test が含まれているシークレットの一覧を取得します。名前によるフィルタリングでは、大文字と小文字が区別されます。

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

Example 他の AWS のサービスによって管理されているシークレットを検索する

次の [list-secrets](#) の例では、サービスによって管理されているシークレットの一覧を取得します。サービスは ID で指定します。詳細については、「[マネージドシークレット](#)」を参照してください。

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

AWS SDK

AWS SDKs のいずれかを使用してシークレットを検索するには、[ListSecrets](#) を使用します。詳細については、「[the section called “AWS SDKs”](#)」を参照してください。

AWS Secrets Manager シークレットを削除する

シークレットには重要な性質があるため、AWS Secrets Manager は意図的にシークレットの削除を困難にします。Secrets Manager は、シークレットをすぐには削除しません。Secrets Manager は、シークレットをすぐにアクセス不能にし、最短で 7 日間の復旧期間が経過した後に削除されるようスケジュールを設定します。ウィンドウの復旧期間が終了するまで、以前に削除したシークレットを復旧することができます。削除対象としてマークしたシークレットに対しては料金は発生しません。

プライマリシークレットが他のリージョンにレプリケートされている場合、プライマリシークレットを削除することはできません。最初にレプリカを削除してから、プライマリシークレットを削除します。レプリカを削除すると、すぐに削除されます。

シークレットのバージョンを直接削除することはできません。代わりに、AWS CLI または AWS SDK を使用して、バージョンからすべてのステージングラベルを削除します。これにより、そのバージョンは非推奨とマークされ、Secrets Manager はバックグラウンドでバージョンを自動的に削除できるようになります。

アプリケーションがまだシークレットを使用しているかどうかわからない場合は、復旧ウィンドウ中にシークレットにアクセスしようとしたときに警告する Amazon CloudWatch アラームを作成できます。詳細については、「[削除が予定されている AWS Secrets Manager シークレットへのアクセスをモニタリングする](#)」を参照してください。


シークレットを削除するには、`secretsmanager:ListSecrets` と `secretsmanager:DeleteSecret` のアクセス許可が必要です。

Secrets Manager は、シークレットを削除すると CloudTrail ログエントリを生成します。詳細については、「[the section called “でログ記録する AWS CloudTrail”](#)」を参照してください。

シークレットを削除するには (コンソール)

1. Secrets Manager のコンソール (<https://console.aws.amazon.com/secretsmanager/>) を開きます。
2. シークレットのリストで、削除するシークレットを選択します。
3. [Secrets details] (シークレットの詳細) セクションで、[Actions] (アクション) を選択し、[Delete secret] (シークレットの削除) を選択します。
4. [Disable secret and schedule deletion] (シークレットの無効化と削除のスケジュール) ダイアログボックスの、[Waiting period] (待機期間) に、永続的に削除するまでの待機日数を入力します。Secrets Manager は DeletionDate というフィールドをアタッチし、現在の日付と時刻に、復旧期間として指定した日数を加えたものを設定します。
5. [Schedule deletion] (削除をスケジュールする) を選択します。

削除済みのシークレットを表示するには

1. Secrets Manager のコンソール (<https://console.aws.amazon.com/secretsmanager/>) を開きます。
2. [Secrets] (シークレット) ページで、[Preferences] (設定) ) を選択します。
3. [設定] ダイアログボックスで、[削除予定のシークレットを表示] を選択し、[保存] を選択します。

レプリカシークレットを削除する

1. Secrets Manager コンソール (<https://console.aws.amazon.com/secretsmanager/>) を開く。
2. プライマリシークレットを選択します。
3. [Replicate Secret] (シークレットのレプリケーション) セクションで、レプリカのシークレットを選択します。
4. [Actions] (アクション) メニューから [Delete Replica] (レプリカの削除) を選択します。

AWS CLI

Example シークレットの削除

次の [delete-secret](#) の例では、シークレットの削除を行います。DeletionDate レスポンスフィールドの日付と時刻 [restore-secret](#) まで、を使用してシークレットを復元できます。他のリージョンにレプリカが作成されているシークレットを削除する場合は、まずそのレプリカを [remove-regions-from-replication](#) で削除してから、[delete-secret](#) を呼び出します。

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

Example シークレットを直ちに削除する

次の [delete-secret](#) の例では、復旧期間なしでシークレットを直ちに削除します。この場合のシークレットは復元できません。

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

Example レプリカシークレットを削除する

次の [remove-regions-from-replication](#) の例では、eu-west-3 にあるレプリカシークレットを削除しています。他のリージョンにレプリカが作成されているプライマリシークレットを削除するには、まずそのレプリカを削除してから [delete-secret](#) を呼び出します。

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

AWS SDK

シークレットを削除するには、[DeleteSecret](#) コマンドを使用します。シークレットのバージョンを削除するには、[UpdateSecretVersionStage](#) コマンドを使用します。レプリカを削除するには、[StopReplicationToReplica](#) コマンドを使用します。詳しくは、「[the section called “AWS SDKs”](#)」を参照してください。

AWS Secrets Manager シークレットを復元する

Secrets Manager は、削除が予定されているシークレットを非推奨とみなし、ユーザーはこれに直接アクセスすることはできなくなります。復旧期間が過ぎると、Secrets Manager はシークレットを完全に削除します。Secrets Manager がシークレットを削除すると、復元することはできません。復旧期間が終了する前に、シークレットを復元して再度アクセス可能にすることができます。これにより、スケジュールされた完全削除をキャンセルする DeletionDate フィールドが削除されます。

コンソールでシークレットとそのメタデータを復元するには、`secretsmanager:ListSecrets` と `secretsmanager:RestoreSecret` のアクセス許可が必要です。

Secrets Manager は、シークレットを復元するときに CloudTrail ログエントリを生成します。詳細については、「[the section called “でログ記録する AWS CloudTrail”](#)」を参照してください。

シークレットを復元するには (コンソール)

1. Secrets Manager のコンソール (<https://console.aws.amazon.com/secretsmanager/>) を開きます。
2. シークレットのリストで、復元するシークレットを選択します。

削除したシークレットがシークレットのリストに表示されないときは、[Preferences] (設定)



を選択します。[設定] ダイアログボックスで、[削除予定のシークレットを表示] を選択し、[保存] を選択します。

3. [Secret details] (シークレットの詳細) ページで、[Cancel deletion] (削除のキャンセル) を選択します。
4. [Cancel secret deletion] (シークレットの削除のキャンセル) ダイアログボックスで、[Cancel deletion] (削除のキャンセル) を選択します。

AWS CLI

Example 以前に削除したシークレットを復元する

次の [restore-secret](#) の例では、スケジュールにより以前に削除されたシークレットを復元します。

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


AWS SDK

削除対象としてマークされたシークレットを復元するには、[RestoreSecret](#) コマンドを実行します。詳細については、「[the section called “AWS SDKs”](#)」を参照してください。

AWS Secrets Manager シークレットにタグ付けする

Secrets Manager は、タグを定義されるキーとオプションの値から構成されるラベルとして定義します。タグを使用すると、お使いの AWS アカウントでシークレットとその他のリソースを簡単に管理、検索、フィルタリングできます。シークレットにタグ付けする時、すべてのリソースに標準化された命名規則を使用すること。詳細については、「[タグ付けのベストプラクティス](#)」ホワイトペーパーを参照してください。

シークレットにアタッチされているタグをチェックすることにより、シークレットへのアクセスを許可または拒否できます。詳細については、「[the section called “例: タグを使用してシークレットへのアクセスを制御する”](#)」を参照してください。

コンソール、AWS CLI、SDK ではタグでシークレットを見つけることができます。また、AWS の [Resource Groups](#) ツールを使用すると、タグに基づいてリソースを統合整理するカスタムコンソールを作成できます。特定のタグを持つシークレットを検索するには、「[the section called “シークレットを検索する”](#)」を参照してください。Secrets Manager は、タグベースのコスト配分をサポートしていません。

決して、シークレットの機密情報をタグに保存しないでください。

タグクォータと名前付けの制限については、AWS の全般的なリファレンスガイドの「[タグ付けのサービスクォータ](#)」を参照してください。タグでは、大文字と小文字が区別されます。

Secrets Manager では、シークレットのタグ付けやタグ解除を行うと、CloudTrail のログエントリが生成されます。詳細については、「[the section called “でログ記録する AWS CloudTrail”](#)」を参照してください。

シークレットのタグを変更するには (コンソール)

1. Secrets Manager のコンソール (<https://console.aws.amazon.com/secretsmanager/>) を開きます。
2. シークレットのリストから、自分のシークレットを選択します。
3. シークレットの詳細ページの [タグ] タブで、[タグを編集] を選択します。タグキーの名前と値は、大文字と小文字は区別されます。また、タグキーは一つだけである必要があります。

AWS CLI

Example シークレットにタグを追加する

次の [tag-resource](#) の例は、短縮構文を使用してタグをアタッチする方法を示しています。

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

Example シークレットに複数のタグを追加する

次の [tag-resource](#) の例では、キーと値のタグ 2 個がシークレットにアタッチされます。

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

Example シークレットからタグを削除する

次の [untag-resource](#) の例では、シークレットから 2 個のタグが削除されます。タグごとに、キーと値の両方が削除されます。

```
aws secretsmanager untag-resource \  
    --secret-id MyTestSecret \  
    --tag-keys ['FirstTag', 'SecondTag']'
```

AWS SDK

シークレットのタグを変更するには、[TagResource](#) または [UntagResource](#) を使用します。詳しくは、「[the section called “AWS SDKs”](#)」を参照してください。

リージョン間でシー AWS Secrets Manager クレートをレプリケートする

シークレットを複数のリージョンにレプリケートして、それらのリージョンにまたがるアプリケーションをサポートし、リージョンアクセスと低レイテンシーの要件を満たすことができます。後で必要な場合は、[レプリカシークレットをスタンダードに昇格](#)させ、レプリケーション用に個別に設定できます。Secrets Manager は、指定したリージョン全体で、タグ、リソースポリシー、シークレットの更新など、暗号化されたシークレットデータおよびメタデータをレプリケートします。

複製されたシークレットの ARN は、リージョンを除いてプライマリシークレットと同じです。以下にその例を示します。

- プライマリシークレット:
`arn:aws:secretsmanager:Region1:123456789012:secret:MySecret-a1b2c3`
- レプリカシークレット:
`arn:aws:secretsmanager:Region2:123456789012:secret:MySecret-a1b2c3`

レプリカシークレットの料金情報については、[AWS Secrets Manager の料金](#)を参照してください。

他のリージョンにレプリケートされているソースデータベースのデータベース認証情報を保存すると、シークレットにはソースデータベースの接続情報が含まれます。その後、シークレットをレプリケートすると、レプリカはソースシークレットのコピーとなり、同じ接続情報が含まれます。リージョン接続情報のシークレットにキー/値ペアを追加できます。

プライマリシークレットのローテーションを設定すると、Secrets Manager はプライマリリージョンでシークレットのローテーションを実行し、新しいシークレット値が関連するすべてのレプリカシークレットに反映されます。すべてのレプリカシークレットのローテーションを個別に管理する必要はありません。

シークレットは、有効なすべての AWS リージョンでレプリケートできます。ただし、AWS GovCloud (US) や中国 AWS リージョンなどの特別なリージョンで Secrets Manager を使用する場合、シークレットとレプリカはこれらの特別な AWS リージョン内でのみ設定できます。有効なリージョンのシークレットを特殊な AWS リージョンにレプリケートしたり、特殊なリージョンから商用リージョンにシークレットをレプリケートしたりすることはできません。

シークレットを別のリージョンにレプリケートするには、そのリージョンを有効にする必要があります。詳細については、「[AWS リージョンの管理](#)」を参照してください。

シークレットが保存されているリージョンで Secrets Manager エンドポイントを呼び出すことで、レプリケートせずに複数のリージョンでシークレットを使用できます。; エンドポイントのリストについては、「[the section called “Secrets Manager エンドポイント”](#)」を参照してください。レプリケーションを使用してワークロードの耐障害性を向上させるには、「[でのディザスタリカバリ \(DR\) アーキテクチャ AWS](#)」、「[パート I: クラウドでのリカバリの戦略](#)」を参照してください。

Secrets Manager は、シークレットをレプリケートするときに CloudTrail ログエントリを生成します。詳細については、「[the section called “でログ記録する AWS CloudTrail”](#)」を参照してください。

シークレットを他のリージョンにレプリケートするには (コンソール)

1. Secrets Manager コンソール (<https://console.aws.amazon.com/secretsmanager/>) を開きます。
2. シークレットのリストから、自分のシークレットを選択します。
3. シークレットの詳細ページの [レプリケーション] タブで、次のいずれかを実行します。
 - シークレットがレプリケートされない場合は、[Replicate secret] (シークレットをレプリケート) をクリックします。
 - シークレットがレプリケートされている場合は、[Replicate secret] (シークレットをレプリケート) セクションで、[Add Regions] (リージョンを追加) をクリックします。
4. [Add replica regions] (レプリカリージョンの追加) ダイアログボックスで、次の操作を行います。
 - a. [AWS Region] (AWS リージョン) で、シークレットのレプリケート先となるリージョンを選択します。
 - b. (オプション) [Encryption key] (暗号化キー) で、シークレットの暗号化に使用する KMS キーを選択します。このキーは、レプリカのリージョンに存在する必要があります。
 - c. (オプション) 別のリージョンを追加するには、[Add more regions] (リージョンを追加) をクリックします。
 - d. [Replicate] (レプリケート) を選択します。

シークレットの詳細ページに戻ります。[Replicate Secret] (シークレットをレプリケート) セクションで、[Replication Status] (レプリケーションステータス) がそれぞれのリージョンを表示します。

AWS CLI

Example シークレットを異なるリージョンにレプリケートする

次に、シークレットをeu-west-3にレプリケートする、[replicate-secret-to-regions](#)の例を示します。レプリカは、AWS マネージドキー aws/secretsmanager で暗号化されます。

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

Example シークレットを作成してレプリケートする

次の例では、シークレットを作成し、eu-west-3にレプリケートします。レプリカは、AWS マネージドキー aws/secretsmanager で暗号化されます。

```
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

シークレットをレプリケートするには、[ReplicateSecretToRegions](#) コマンドを使用してください。詳細については、「[the section called “AWS SDKs”](#)」を参照してください。

AWS Secrets Manager でレプリカシークレットをスタンドアロンシークレットに昇格させる

レプリカシークレットは、別のプライマリからレプリケートされるシークレットです。AWS リージョン。プライマリと同じシークレット値とメタデータを持ちますが、別の KMS キーで暗号化できません。レプリカシークレットは、暗号化キーを除き、プライマリシークレットとは別に更新することはできません。レプリカシークレットを昇格すると、プライマリシークレットからレプリカシークレットが切断され、レプリカシークレットがスタンドアロンのシークレットになります。プライマリシークレットに加えた変更は、スタンドアロンのシークレットにレプリケーションされません。

プライマリシークレットが使用できなくなった場合、災害対策ソリューションとして、レプリカシークレットをスタンドアロンのインスタンスに昇格させることができます。または、レプリカのロー

テーションを有効にする場合は、レプリカをスタンドアロンのシークレットに昇格させることができます。

レプリカを昇格する場合は、スタンドアロンシークレットを使用するために、必ず対応するアプリケーションを更新します。

Secrets Manager では、シークレットをプロモートすると CloudTrail ログエントリを生成します。詳細については、「[the section called “でログ記録する AWS CloudTrail”](#)」を参照してください。

レプリカシークレットを昇格するには (コンソール)

1. 次の場所から Secrets Manager にログインします (<https://console.aws.amazon.com/secretsmanager/>)。
2. レプリカのリージョンに移動します。
3. [Secrets] (シークレット) ページで、レプリカシークレットを選択します。
4. レプリカシークレットの詳細ページで、[Promote to standalone secret] (スタンドアロンシークレットに昇格させる) を選択します。
5. [Promote replica to standalone secret] (レプリカのスタンドアロンシークレットへの昇格) ダイアログボックスで、リージョンを入力してから、[Promote replica] (レプリカを昇格させる) を選択します。

AWS CLI

Example レプリカシークレットをプライマリに昇格させる

次の [stop-replication-to-replica](#) の例は、レプリカシークレットからプライマリへのリンクを削除します。このレプリカシークレットは、レプリカのリージョンでプライマリシークレットに昇格されます。[stop-replication-to-replica](#) は、レプリカリージョン内から呼び出す必要があります。

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

AWS SDK

レプリカをスタンドアロンシークレットに昇格させるには、[StopReplicationToReplica](#) コマンドを使用します。このコマンドはレプリカシークレットリージョンから呼び出す必要があります。詳細については、「[the section called “AWS SDKs”](#)」を参照してください。

AWS Secrets Manager レプリケーションの防止

シークレットは [ReplicateSecretToRegions](#) を使用してレプリケートすることも [CreateSecret](#)、 を使用して作成されたときにレプリケートすることもできます。ユーザーがシークレットをレプリケートできないようにする場合は、 `AddReplicaRegions` パラメータを含むアクションを禁止することをお勧めします。アクセス許可ポリシーで `Condition` ステートメントを使用して、レプリカリージョンを追加しないアクションのみを許可できます。使用できる条件ステートメントについては、次のポリシー例を参照してください。

Example レプリケーション許可の禁止

次のポリシー例は、レプリカリージョンを追加しないすべてのアクションを許可する方法を示しています。これにより、ユーザーは `ReplicateSecretToRegions` と `CreateSecret` の両方を通じてシークレットをレプリケートできなくなります。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "secretsmanager:*",
      "Resource": "*",
      "Condition": {
        "Null": {
          "secretsmanager:AddReplicaRegions": "true"
        }
      }
    }
  ]
}
```

Example 特定のリージョンにのみレプリケーション許可を付与する

次のポリシーは、次のすべてを許可する方法を示しています。

- レプリケーションなしでシークレットを作成する
- 米国およびカナダのリージョンのみにレプリケーションするシークレットを作成する
- シークレットを米国およびカナダのリージョンにのみレプリケートする

```
{
```

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

AWS Secrets Manager レプリケーションのトラブルシューティング

レプリケーションが失敗する原因として、次のようなものがあります。

選択したリージョンに同じ名前のシークレットがある

この問題を解決するには、レプリカリージョンにある重複した名前のシークレットを上書きします。レプリケーションを再試行し、[レプリケーションを再試行] ダイアログボックスで [上書き] をクリックします。

KMS キーにレプリケーションを完成させるためのアクセス許可がない

Secrets Manager は、レプリカリージョンにある新しい KMS キーを使用して再暗号化する前に、まずシークレットを復号します。プライマリリージョンの暗号化キーに対する `kms:Decrypt` アクセス許可がない場合、このエラーが発生します。aws/secretsmanager 以外の KMS キーでレプリケートされたシークレットを暗号化するには、キーに `kms:GenerateDataKey` と `kms:Encrypt` が必要です。 [the section called “KMS キーのアクセス許可”](#) を参照してください。

KMS キーが無効になっているか、見つかりません

プライマリリージョンの暗号化キーが無効化または削除されている場合、Secrets Manager はシークレットをレプリケートできません。このエラーは、暗号化キーを変更した場合でも、無効化または削除された暗号化キーで暗号化された[カスタムラベルが付いたバージョン](#)がシークレットにある場合に発生することがあります。Secrets Manager が暗号化を行う方法については、「[the section called “シークレット暗号化と復号”](#)」を参照してください。この問題を回避するには、Secrets Manager が現在の暗号化キーで暗号化するようにシークレットバージョンを再作成できます。詳細については、「[シークレットの暗号化キーを変更する](#)」を参照してください。その後でレプリケーションを再試行します。

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

レプリケーションを行うリージョンが有効化されていない

リージョンを有効にする方法については、「AWS アカウント管理リファレンスガイド」の「[AWS リージョンの管理](#)」を参照してください。

からシークレットを取得する AWS Secrets Manager

Secrets Manager は、シークレットを取得するときに CloudTrail ログエントリを生成します。詳細については、「[the section called “でログ記録する AWS CloudTrail”](#)」を参照してください。

シークレット値は、以下を使用して取得できます。

- [Java を使用して Secrets Manager のシークレット値を取得する](#)
- [Python を使用して Secrets Manager のシークレット値を取得する](#)
- [.NET を使用して Secrets Manager のシークレット値を取得する](#)
- [Go を使用して Secrets Manager のシークレット値を取得する](#)
- [C++ AWS SDK を使用して Secrets Manager のシークレット値を取得する](#)
- [JavaScript AWS SDK を使用して Secrets Manager のシークレット値を取得する](#)
- [Kotlin AWS SDK を使用して Secrets Manager のシークレット値を取得する](#)
- [PHP AWS SDK を使用して Secrets Manager のシークレット値を取得する](#)
- [Ruby AWS SDK を使用して Secrets Manager のシークレット値を取得する](#)
- [Rust AWS SDK を使用して Secrets Manager のシークレット値を取得する](#)
- [を使用してシークレット値を取得する AWS CLI](#)
- [AWS コンソールを使用してシークレット値を取得する](#)
- [AWS Secrets Manager で AWS Batch シークレットを使用する](#)
- [リソースで AWS Secrets Manager AWS CloudFormation シークレットを取得する](#)
- [Amazon Elastic Kubernetes Service で AWS Secrets Manager シークレットを使用する](#)
- [GitHub ジョブで AWS Secrets Manager シークレットを使用する](#)
- [AWS Secrets Manager で AWS IoT Greengrass シークレットを使用する](#)
- [AWS Lambda 関数で AWS Secrets Manager シークレットを使用する](#)
- [パラメータストア内で AWS Secrets Manager シークレットを使用する](#)

Java を使用して Secrets Manager のシークレット値を取得する

アプリケーションでは、任意の SDK `BatchGetSecretValue` で `GetSecretValue` または `getSecretValue` を呼び出してシークレットを取得できます。AWS SDKs ただし、シークレット値はクライアント側のキャッシュを使用してキャッシュすることをお勧めします。シークレットをキャッシュすることで、速度が向上し、コストを削減できます。

シークレットの認証情報を使用してデータベースに接続するには、Secrets Manager の SQL Connection ドライバーを使用します。これにより、基本 JDBC ドライバーがラップされます。また、クライアント側のキャッシュを使用するため、Secrets Manager APIs を呼び出すコストを削減できます。

トピック

- [クライアント側のキャッシュで Java を使用して Secrets Manager のシークレット値を取得する](#)
- [AWS Secrets Manager シークレットの認証情報を使用して JDBC を使用して SQL データベースに接続する](#)
- [Java AWS SDK を使用して Secrets Manager のシークレット値を取得する](#)

クライアント側のキャッシュで Java を使用して Secrets Manager のシークレット値を取得する

シークレットを取得するときに、Secrets Manager の Java ベースのキャッシュコンポーネントを使用して、将来使用するためにキャッシュすることができます。キャッシュされたシークレットの取得は、Secrets Manager からの取得よりも高速です。Secrets Manager API を呼び出すにはコストがかかるため、キャッシュを使用するとコストを削減できます。シークレットを取得するすべての方法については、「[シークレットを取得する](#)」を参照してください。

キャッシュポリシーは LRU (最近最も使われていない) であるため、キャッシュでシークレットを破棄する必要が生じた場合は、最も最近使われていないシークレットが破棄されます。デフォルトでは、1 時間ごとにキャッシュでシークレットが更新されます。キャッシュで[シークレットが更新される頻度](#)を設定できるだけでなく、[シークレットの取得にフック](#)させて機能を追加することもできます。

キャッシュ参照が解放されると、キャッシュはガベージコレクションを強制しません。キャッシュの実装には、キャッシュの無効化は含まれていません。キャッシュを実装するのはキャッシュを使用するためであり、セキュリティを強化するためでもセキュリティに焦点を当てるためでもありません。キャッシュ内のアイテムを暗号化するなど、セキュリティを強化する必要がある場合は、所定のインターフェイスと抽象メソッドを使用してください。

このコンポーネントを使用するには、以下が必要です。

- Java 8 以上の開発環境。Oracle のウェブサイトの「[Java SE Downloads](#)」(Java SE のダウンロード) を参照してください。

- AWS SDK 1.x for Java。プロジェクトでは、AWS SDK for Java の両方のバージョンを使用できます。詳細については、「[SDK for Java 1.x および 2.x side-by-sideの使用](#)」を参照してください。

ソースコードをダウンロードするには、「」の「[Secrets Manager Java ベースのキャッシュクライアントコンポーネント](#)」を参照してください GitHub。

コンポーネントをプロジェクトに追加するには、Maven pom.xml ファイルに、次の依存関係を含めます。Maven の詳細については、Apache Maven プロジェクトのウェブサイトの「[Getting Started Guide](#)」(入門ガイド)を参照してください。

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

必要な許可:

- secretsmanager:DescribeSecret
- secretsmanager:GetSecretValue

詳細については、「[アクセス許可に関するリファレンス](#)」を参照してください。

リファレンス

- [SecretCache](#)
- [SecretCacheConfiguration](#)
- [SecretCacheHook](#)

Example シークレットを取得する

次のコード例は、シークレット文字列を取得する Lambda 関数を示しています。これは関数ハンドラーの外部でのキャッシュのインスタンス化の[ベストプラクティス](#)に従うため、Lambda 関数を再度呼び出しても、API は継続して呼び出されません。

```
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

Secrets Manager からリクエストされたシークレットのインメモリキャッシュ。 [the section called “getSecretString”](#) または [the section called “getSecretBinary”](#) を使用して、キャッシュからシークレットを取得します。キャッシュの設定は、コンストラクタで [the section called “SecretCacheConfiguration”](#) オブジェクトを渡すことで設定できます。

詳細と例については、「[the section called “Java とクライアント側のキャッシュ”](#)」を参照してください。

コンストラクタ

```
public SecretCache()
```

SecretCache オブジェクトのデフォルトコンストラクタ。

```
public SecretCache(AWSSecretsManagerClientBuilder builder)
```

提供された [AWSSecretsManagerClientBuilder](#) を用いて作成された Secrets Manager クライアントを使用して、新しいキャッシュを構築します。このコンストラクタを使用して、Secrets Manager クライアントをカスタマイズします (特定のリージョンまたはエンドポイントを使用するなど)。

```
public SecretCache(AWSSecretsManager client)
```

提供された [AWSSecretsManagerClient](#) を使用して、新しいシークレットキャッシュを構築します。このコンストラクタを使用して、Secrets Manager クライアントをカスタマイズします (特定のリージョンまたはエンドポイントを使用するなど)。

```
public SecretCache(SecretCacheConfiguration config)
```

提供された [the section called “SecretCacheConfiguration”](#) を使用して、新しいシークレットキャッシュを構築します。

方法

```
getSecretString
```

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

Secrets Manager から文字列シークレットを取得します。戻り値は [String](#)。

```
getSecretBinary
```

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

Secrets Manager からバイナリシークレットを取得します。戻り値は [ByteBuffer](#)。

```
refreshNow
```

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

キャッシュを強制的に更新します。エラーが発生せずに更新が完了した場合は true を返し、そうでない場合は false を返します。

```
close
```

```
public void close()
```

キャッシュを終了します。

SecretCacheConfiguration

キャッシュされるシークレットの最大キャッシュサイズや有効期限 (TTL) などの、[the section called “SecretCache”](#) のキャッシュ設定オプション。

コンストラクタ

```
public SecretCacheConfiguration
```

SecretCacheConfiguration オブジェクトのデフォルトコンストラクタ。

方法

getClient

```
public AWSecretsManager getClient()
```

キャッシュがシークレットを取得する [AWSecretsManagerClient](#) を返します。

setClient

```
public void setClient(AWSecretsManager client)
```

キャッシュがシークレットを取得する [AWSecretsManagerClient](#) クライアントを設定します。

getCacheHook

```
public SecretCacheHook getCacheHook()
```

キャッシュ更新に接続するために使用される [the section called "SecretCacheHook"](#) インターフェイスを返します。

setCacheHook

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

キャッシュ更新に接続するために使用される [the section called "SecretCacheHook"](#) インターフェイスを設定します。

getMaxCacheサイズ

```
public int getMaxCacheSize()
```

最大キャッシュサイズを返します。デフォルトは 1,024 個のシークレットです。

setMaxCacheサイズ

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

最大キャッシュサイズを設定します。デフォルトは 1,024 個のシークレットです。

getCacheItemTTL

```
public long getCacheItemTTL()
```

キャッシュされた項目の TTL をミリ秒単位で返します。キャッシュされたシークレットがこの TTL を超えると、キャッシュは [AWSecretsManagerClient](#) から新しいシークレットのコピーを取得します。デフォルトは 1 時間 (ミリ秒単位) です。

TTL の後にシークレットがリクエストされると、キャッシュはシークレットを同期的に更新します。同期更新が失敗した場合、キャッシュは古いシークレットを返します。

setCacheItemTTL

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

キャッシュされた項目の TTL をミリ秒単位で設定します。キャッシュされたシークレットがこの TTL を超えると、キャッシュは [AWSecretsManagerClient](#) から新しいシークレットのコピーを取得します。デフォルトは 1 時間 (ミリ秒単位) です。

getVersionStage

```
public String getVersionStage()
```

キャッシュするシークレットのバージョンを返します。詳細については、「[Secret versions](#)」(シークレットバージョン) を参照してください。デフォルトは "AWSCURRENT" です。

setVersionStage

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

キャッシュするシークレットのバージョンを設定します。詳細については、「[Secret versions](#)」(シークレットバージョン) を参照してください。デフォルトは "AWSCURRENT" です。

SecretCacheConfiguration withClient

```
public SecretCacheConfiguration withClient(AWSecretsManager client)
```

シークレットを取得する [AWSecretsManagerClient](#) を設定します。新しい設定を持つ更新された SecretCacheConfiguration オブジェクトを返します。

SecretCacheConfiguration withCacheHook

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

インメモリキャッシュに接続するために使用されるインターフェイスを設定します。新しい設定を持つ更新された SecretCacheConfiguration オブジェクトを返します。

SecretCacheConfiguration withMaxCacheサイズ

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

最大キャッシュサイズを設定します。新しい設定を持つ更新された SecretCacheConfiguration オブジェクトを返します。

SecretCacheConfiguration withCacheItemTTL

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

キャッシュされた項目の TTL をミリ秒単位で設定します。キャッシュされたシークレットがこの TTL を超えると、キャッシュは [AWSSecretsManagerClient](#) から新しいシークレットのコピーを取得します。デフォルトは 1 時間 (ミリ秒単位) です。新しい設定を持つ更新された SecretCacheConfiguration オブジェクトを返します。

SecretCacheConfiguration withVersionStage

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

キャッシュするシークレットのバージョンを設定します。詳細については、「[Secret versions](#)」(シークレットバージョン) を参照してください。新しい設定を持つ更新された SecretCacheConfiguration オブジェクトを返します。

SecretCacheHook

[the section called “SecretCache”](#) に接続して、キャッシュに保存されているシークレットに対してアクションを実行するインターフェイス。

```
put
```

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

キャッシュに保存するオブジェクトを準備します。

キャッシュに保存するオブジェクトを返します。

```
get
```

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

キャッシュされたオブジェクトからオブジェクトを派生させます。

キャッシュから返すオブジェクトを返します

AWS Secrets Manager シークレットの認証情報を使用して JDBC を使用して SQL データベースに接続する

Java アプリケーションでは、Secrets Manager SQL Connection ドライバーを使用して、Secrets Manager に保存されている認証情報を使用して MySQL、PostgreSQL、Oracle、MSSQLServer、Db2、および Redshift データベースに接続できます。各ドライバーはベース JDBC ドライバーをラップしているため、JDBC 呼び出しを使用してデータベースにアクセスすることができます。ただし、接続用のユーザーネームとパスワードを渡す代わりに、シークレットの ID を指定します。ドライバーは、Secrets Manager を呼び出してシークレット値を取得してから、シークレット内の認証情報と接続情報を使用してデータベースに接続します。また、ドライバーは [Java のクライアント側キャッシュライブラリ](#) を使用して認証情報をキャッシュするため、その後の接続では Secrets Manager を呼び出す必要はありません。デフォルトでは、1 時間ごと、およびシークレットがローテーションされたときに、キャッシュが更新されます。キャッシュを設定するには、[the section called "SecretCacheConfiguration"](#) を参照してください。

ソースコードは からダウンロードできます [GitHub](#)。

Secrets Manager SQL 接続ドライバーを使用するには、以下が必要です。

- アプリケーションが Java 8 以降である必要があります。
- シークレットが次のいずれかである必要があります。
 - [期待される JSON 構造のデータベースシークレット](#)。シークレットの形式を確認するには、Secrets Manager コンソールで、シークレットを表示して [Retrieve secret value] を選択します。または、`aws secretsmanager get-secret-value` を AWS CLI 呼び出します [get-secret-value](#)。
 - Amazon RDS [マネージドシークレット](#)。このタイプのシークレットでは、接続を確立するときにエンドポイントとポートを指定する必要があります。
 - Amazon Redshift [マネージドシークレット](#)。このタイプのシークレットでは、接続を確立するときにエンドポイントとポートを指定する必要があります。

データベースが他のリージョンにレプリケートされている場合、別のリージョンのレプリカデータベースに接続するには、接続の作成時にリージョンのエンドポイントとポートを指定します。リージョン接続情報は、追加のキー/値のペア、SSM パラメータストアパラメータ、またはコード構成でシークレットに格納できます。

ドライバーをプロジェクトに追加するには、Maven ビルドファイル pom.xml で、次のドライバーの依存関係を追加します。詳細については、Maven Central Repository の web サイトの「[Secrets Manager SQL Connection Library](#)」(Secrets Manager SQL 接続ライブラリ) を参照してください。

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

このドライバーでは[デフォルトの認証情報プロバイダーチェーン](#)を使用します。Amazon EKS でドライバーを実行すると、サービスアカウントロールの代わりに、実行中のノードの認証情報が取得される可能性があります。これに対処するには、com.amazonaws:aws-java-sdk-sts のバージョン 1 を Gradle または Maven プロジェクトファイルに依存関係として追加します。

secretsmanager.properties ファイルに AWS PrivateLink DNS エンドポイント URL とリージョンを設定するには：

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

プライマリリージョンをオーバーライドするには、AWS_SECRET_JDBC_REGION 環境変数を設定するか、secretsmanager.properties ファイルに次の変更を加えます。

```
drivers.region = region
```

必要な許可:

- secretsmanager:DescribeSecret
- secretsmanager:GetSecretValue

詳細については、「[アクセス許可に関するリファレンス](#)」を参照してください。

例:

- [データベースへの接続を確立する](#)
- [エンドポイントとポートを指定して接続を確立する](#)
- [c3p0 接続プールを使用して接続を確立する](#)
- [c3p0 接続プールを使用して、エンドポイントとポートを指定して接続を確立する](#)

データベースへの接続を確立する

次の例では、シークレット内の認証情報と接続情報を使用してデータベースへの接続を確立する方法を示しています。接続が確立すると、JDBC 呼び出しを使用してデータベースにアクセスすることができます。詳細については、Java ドキュメントのウェブサイトの「[JDBC Basics](#)」(JDBC の基本)を参照してください。

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" ).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);
```

Oracle

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

MSSQLServer

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

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" ).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);
```

エンドポイントとポートを指定して接続を確立する

次の例は、シークレット内の認証情報を使用して、指定したエンドポイントとポートでデータベースへの接続を確立する方法を示しています。

[Amazon RDS マネージドシークレット](#)には、データベースのエンドポイントおよびポートは含まれていません。Amazon RDS が管理するシークレットのマスター認証情報を使用してデータベースに接続するには、コードでマスター認証情報を指定します。

[他のリージョンにレプリケートされるシークレット](#)は、リージョンデータベースへの接続のレイテンシーを改善できますが、ソースシークレット以外の接続情報を保持しません。各レプリカは、ソースシークレットのコピーです。リージョン接続情報をシークレットに保存するには、リージョンのエンドポイントとポート情報のキー/値のペアを追加します。

接続が確立すると、JDBC 呼び出しを使用してデータベースにアクセスすることができます。詳細については、Java ドキュメントのウェブサイトの「[JDBC Basics](#)」(JDBC の基本) を参照してください。

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" ).newInstance();

// 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" ).newInstance();
```

```
// 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" ).newInstance();

// 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.AWSSecretsManagerRedshiftDriver");

// 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 接続プールを使用して接続を確立する

次の例は、ドライバーを使用してシークレットから認証情報および接続情報を取得する c3p0.properties ファイルで接続プールを確立する方法を示しています。user と jdbcUrl には、シークレット ID を入力して接続プールを設定します。その後、プールから接続を取得し、他のデータベース接続として使用することができます。詳細については、Java ドキュメントのウェブサイトの「[JDBC Basics](#)」(JDBC の基本)を参照してください。

c3p0 の詳細については、Machinery For Change のウェブサイトの「[c3p0](#)」を参照してください。

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 接続プールを使用して、エンドポイントとポートを指定して接続を確立する

次の例は、ドライバーを使用して指定されたエンドポイントとポートでシークレットの認証情報を取得する `c3p0.properties` ファイルで接続プールを確立する方法を示しています。その後、プールから接続を取得し、他のデータベース接続として使用することができます。詳細については、Java ドキュメントのウェブサイトの「[JDBC Basics](#)」(JDBC の基本)を参照してください。

[Amazon RDS マネージドシークレット](#)には、データベースのエンドポイントおよびポートは含まれていません。Amazon RDS が管理するシークレットのマスター認証情報を使用してデータベースに接続するには、コードでマスター認証情報を指定します。

[他のリージョンにレプリケートされるシークレット](#)は、リージョンデータベースへの接続のレイテンシーを改善できますが、ソースシークレット以外の接続情報を保持しません。各レプリカは、ソース

シークレットのコピーです。リージョン接続情報をシークレットに保存するには、リージョンのエンドポイントとポート情報のキー/値のペアを追加します。

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 SDK を使用して Secrets Manager のシークレット値を取得する

アプリケーションでは、任意の SDK `BatchGetSecretValue` で `GetSecretValue` または `BatchGetSecretValue` を呼び出してシークレットを取得できます。AWS SDKs ただし、シークレット値はクライアント側のキャッシュを使用してキャッシュすることをお勧めします。シークレットをキャッシュすることで、速度が向上し、コストを削減できます。

- データベースの認証情報をシークレットに保存する場合は、[Secrets Manager SQL 接続ドライバー](#)を使用し、シークレット内の認証情報を用いてデータベースに接続します。
- 他のタイプのシークレットについては、[Secrets Manager の Java ベースのキャッシュコンポーネントを使用するか、または](#) `BatchGetSecretValue` を使用して SDK を直接呼び出します [BatchGetSecretValue](#)。
[GetSecretValue](#)

以下のコード例は、`GetSecretValue` の使用方法を示しています。

必要な許可: `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);
    }
}
}
```

Python を使用して Secrets Manager のシークレット値を取得する

アプリケーションでは、任意の SDK BatchGetSecretValue で GetSecretValue または を呼び出してシークレットを取得できます。AWS SDKs に対し、シークレット値はクライアント側のキャッシュを使用してキャッシュすることをお勧めします。シークレットをキャッシュすることで、速度が向上し、コストを削減できます。

トピック

- [クライアント側のキャッシュで Python を使用して Secrets Manager のシークレット値を取得する](#)
- [Python AWS SDK を使用して Secrets Manager のシークレット値を取得する](#)
- [Python AWS SDK を使用して Secrets Manager のシークレット値のバッチを取得する](#)

クライアント側のキャッシュで Python を使用して Secrets Manager のシークレット値を取得する

シークレットを取得するときに、Secrets Manager の Python ベースのキャッシュコンポーネントを使用して、将来使用するためにキャッシュすることができます。キャッシュされたシークレットの取得は、Secrets Manager からの取得よりも高速です。Secrets Manager API を呼び出すにはコストがかかるため、キャッシュを使用するとコストを削減できます。シークレットを取得するすべての方法については、「[シークレットを取得する](#)」を参照してください。

キャッシュポリシーは LRU (最近最も使われていない) であるため、キャッシュでシークレットを破棄する必要がある場合は、最も最近使われていないシークレットが破棄されます。デフォルトでは、1 時間ごとにキャッシュでシークレットが更新されます。キャッシュで[シークレットが更新される頻度](#)を設定できるだけでなく、[シークレットの取得にフック](#)させて機能を追加することもできます。

キャッシュ参照が解放されると、キャッシュはガベージコレクションを強制しません。キャッシュの実装には、キャッシュの無効化は含まれていません。キャッシュを実装するのはキャッシュを使用するためであり、セキュリティを強化するためでもセキュリティに焦点を当てるためでもありません。キャッシュ内のアイテムを暗号化するなど、セキュリティを強化する必要がある場合は、所定のインターフェイスと抽象メソッドを使用してください。

このコンポーネントを使用するには、以下が必要です。

- Python 3.6 以降。
- botocore 1.12 以降。「[AWS SDK for Python](#)」および「[Botocore](#)」を参照してください。

- `setuptools_scm` 3.2 以降。 <https://pypi.org/project/setuptools-scm/> を参照してください。

ソースコードをダウンロードするには、「」の「[Secrets Manager Python ベースのキャッシュクライアントコンポーネント](#)」を参照してください GitHub。

コンポーネントをインストールするには、次のコマンドを使用します。

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

必要な許可:

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

詳細については、「[アクセス許可に関するリファレンス](#)」を参照してください。

リファレンス

- [SecretCache](#)
- [SecretCacheConfig](#)
- [SecretCacheHook](#)
- [@InjectSecretString](#)
- [@InjectKeywordedSecretString](#)

Example シークレットを取得する

次の例は、*mysecret* という名前のシークレットのシークレット値を取得する方法を示しています。

```
import boto3
import boto3.session
from aws_secretsmanager_caching import SecretCache, SecretCacheConfig

client = boto3.session.get_session().create_client('secretsmanager')
cache_config = SecretCacheConfig()
cache = SecretCache( config = cache_config, client = client)

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

SecretCache

Secrets Manager から取得されたシークレットのインメモリキャッシュ。 [the section called “get_secret_string”](#) または [the section called “get_secret_binary”](#) を使用して、キャッシュからシークレットを取得します。キャッシュの設定は、コンストラクタで [the section called “SecretCacheConfig”](#) オブジェクトを渡すことで設定できます。

詳細と例については、「[the section called “Python とクライアント側のキャッシュ”](#)」を参照してください。

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

使用できるメソッドは次のとおりです。

- [get_secret_string](#)
- [get_secret_binary](#)

get_secret_string

シークレット文字列値を取得します。

リクエストの構文

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

パラメータ

- `secret_id` (string) -- [必須] シークレットの名前または ARN。
- `version_stage` (string) -- 取得するシークレットのバージョン。詳細については、「[シークレットバージョン](#)」を参照してください。デフォルトはAWSCURRENT「」です。

戻り型

string

get_secret_binary

シークレットバイナリ値を取得します。

リクエストの構文

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

パラメータ

- `secret_id` (string) -- [必須] シークレットの名前または ARN。
- `version_stage` (string) -- 取得するシークレットのバージョン。詳細については、「[シークレットバージョン](#)」を参照してください。デフォルトは `AWSCURRENT` 「」です。

戻り型

[base64 でエンコードされた文字列](#)

SecretCacheConfig

キャッシュされるシークレットの最大キャッシュサイズや有効期限 (TTL) などの、[the section called “SecretCache”](#) のキャッシュ設定オプション。

パラメータ

`max_cache_size` (int)

最大キャッシュサイズ。デフォルトは 1024 個のシークレットです。

`exception_retry_delay_base` (int)

例外が発生してから、リクエストを再試行するまで待機する秒数。デフォルトは 1 です。

`exception_retry_growth_factor` (int)

失敗したリクエストの再試行間の待機時間を計算するために使用する増加係数。デフォルトは 2 です。

`exception_retry_delay_max` (int)

失敗したリクエスト間の最大待機時間 (秒)。デフォルトは 3600 です。

default_version_stage (str)

キャッシュするシークレットのバージョン。詳細については、「[Secret versions](#)」(シークレットバージョン)を参照してください。デフォルトは 'AWSCURRENT' です。

secret_refresh_interval (int)

キャッシュされたシークレット情報の更新間の待機秒数。デフォルトは 3600 です。

secret_cache_hook (SecretCacheHook)

SecretCacheHook 抽象クラスの実装。デフォルト値は、Noneです。

SecretCacheHook

[the section called “SecretCache”](#) に接続して、キャッシュに保存されているシークレットに対してアクションを実行するインターフェイス。

使用できるメソッドは次のとおりです。

- [put](#)
- [get](#)

put

キャッシュに保存するオブジェクトを準備します。

リクエストの構文

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

パラメータ

- obj (object) -- [必須] シークレットまたはシークレットを含むオブジェクト。

戻り型

オブジェクト

get

キャッシュされたオブジェクトからオブジェクトを派生させます。

リクエストの構文

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

パラメータ

- obj (object) -- [必須] シークレットまたはシークレットを含むオブジェクト。

戻り型

オブジェクト

@InjectSecretString

このデコレータは、1 番目と 2 番目の引数として、シークレット ID 文字列と [the section called “SecretCache”](#) を必要とします。このデコレータはシークレット文字列値を返します。シークレットに文字列が含まれている必要があります。

```
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

このデコレータは、1 番目と 2 番目の引数として、シークレット ID 文字列と [the section called “SecretCache”](#) を必要とします。残りの引数は、ラップされた関数のパラメータをシークレット内の JSON キーにマッピングします。シークレットに JSON 構造の文字列が含まれている必要があります。

この JSON を含むシークレットの場合は、次のようになります。

```
{  
    "username": "saanvi",
```

```
"password": "EXAMPLE-PASSWORD"  
}
```

次の例では、シークレットから username および password の JSON 値を抽出する方法を示しています。

```
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 SDK を使用して Secrets Manager のシークレット値を取得する

アプリケーションでは、任意の SDK BatchGetSecretValue で GetSecretValue または を呼び出してシークレットを取得できます。AWS SDKs ただし、シークレット値はクライアント側のキャッシュを使用してキャッシュすることをお勧めします。シークレットをキャッシュすることで、速度が向上し、コストを削減できます。

Python アプリケーションの場合は、[Secrets Manager の Python ベースのキャッシュコンポーネント](#)を使用するか、[get_secret_value](#) または [batch_get_secret_value](#) を使用して SDK を直接呼び出します。

以下のコード例は、GetSecretValue の使用方法を示しています。

必要な許可:secretsmanager:GetSecretValue

```
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 SDK を使用して Secrets Manager のシークレット値のバッチを取得する

次のコード例は、Secrets Manager のシークレットの値をバッチ取得する方法を示しています。

必要な許可:

- `secretsmanager:BatchGetSecretValue`
- `secretsmanager:GetSecretValue` 取得する各シークレットの アクセス許可。
- フィルターを使用する場合は、`secretsmanager:ListSecrets` も必要です。

アクセス許可ポリシーの例については、「[the section called “例: バッチ内のシークレット値のグループを取得するアクセス許可”](#)」を参照してください。

⚠ Important

取得しようとしているグループ内の個別シークレットを取得するためのアクセス許可を拒否する VPCE ポリシーを設定している場合、BatchGetSecretValue はシークレット値を返さず、エラーが返されます。

```
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
```

.NET を使用して Secrets Manager のシークレット値を取得する

アプリケーションでは、任意の SDK BatchGetSecretValue で GetSecretValue または を呼び出してシークレットを取得できます。AWS SDKs ただし、シークレット値はクライアント側のキャッシュを使用してキャッシュすることをお勧めします。シークレットをキャッシュすることで、速度が向上し、コストを削減できます。

トピック

- [クライアント側のキャッシュで .NET を使用して Secrets Manager のシークレット値を取得する](#)
- [.NET AWS SDK を使用して Secrets Manager のシークレット値を取得する](#)

クライアント側のキャッシュで .NET を使用して Secrets Manager のシークレット値を取得する

シークレットを取得するときに、Secrets Manager の .NET ベースのキャッシュコンポーネントを使用して、将来使用するためにキャッシュすることができます。キャッシュされたシークレットの取得は、Secrets Manager からの取得よりも高速です。Secrets Manager API を呼び出すにはコストがかかるため、キャッシュを使用するとコストを削減できます。シークレットを取得するすべての方法については、「[シークレットを取得する](#)」を参照してください。

キャッシュポリシーは LRU (最近最も使われていない) であるため、キャッシュでシークレットを破棄する必要が生じた場合は、最も最近使われていないシークレットが破棄されます。デフォルトでは、1 時間ごとにキャッシュでシークレットが更新されます。キャッシュで[シークレットが更新される頻度](#)を設定できるだけでなく、[シークレットの取得にフック](#)させて機能を追加することもできます。

キャッシュ参照が解放されると、キャッシュはガベージコレクションを強制しません。キャッシュの実装には、キャッシュの無効化は含まれていません。キャッシュを実装するのはキャッシュを使用するためであり、セキュリティを強化するためでもセキュリティに焦点を当てるためでもありません。キャッシュ内のアイテムを暗号化するなど、セキュリティを強化する必要がある場合は、所定のインターフェイスと抽象メソッドを使用してください。

このコンポーネントを使用するには、以下が必要です。

- .NET Framework 4.6.2 以上、または .NET Standard 2.0 以上。Microsoft .NET のウェブサイトの「[.NET のダウンロード](#)」を参照してください。
- AWS SDK for .NET。 [the section called “AWS SDKs”](#) を参照してください。

ソースコードをダウンロードするには、「[の .NET 用キャッシュクライアント](#)」を参照してください
い GitHub。

キャッシュを使用するには、まずキャッシュをインスタンス化してから、GetSecretString または GetSecretBinary を使用して自分のシークレットを取得する必要があります。連続して取得すると、キャッシュはシークレットのキャッシュされたコピーを返します。

キャッシュパッケージを入手するには

- 次のいずれかを行います。
 - プロジェクトディレクトリで次の .NET CLI コマンドを実行します。

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

- .csproj ファイルに次のパッケージリファレンスを追加します。

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

必要な許可:

- secretsmanager:DescribeSecret
- secretsmanager:GetSecretValue

詳細については、「[アクセス許可に関するリファレンス](#)」を参照してください。

リファレンス

- [SecretsManagerCache](#)
- [SecretCacheConfiguration](#)

- [ISecretCacheHook](#)

Example シークレットを取得する

次のコード例は、`MySecret` という名前のシークレットを取得するメソッドを示しています。

```
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 有効期限 (TTL) キャッシュ更新期間を設定する

次のコード例は、`MySecret` という名前のシークレットを取得し、TTL キャッシュの更新期間を 24 時間に設定する方法を示しています。

```
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

Secrets Manager からリクエストされたシークレットのインメモリキャッシュ。 [the section called “GetSecretString”](#) または [the section called “GetSecretBinary”](#) を使用して、キャッシュからシークレットを取得します。キャッシュの設定は、コンストラクタで [the section called “SecretCacheConfiguration”](#) オブジェクトを渡すことで設定できます。

詳細と例については、「[the section called “.NET とクライアント側のキャッシュ”](#)」を参照してください。

コンストラクタ

```
public SecretsManagerCache()
```

SecretsManagerCache オブジェクトのデフォルトコンストラクタ。

```
public SecretsManagerCache(IAmazonSecretsManager secretsManager)
```

提供された `secretsManager` を使用して作成された Secrets Manager クライアントを使用して新しいキャッシュを構築します [AmazonSecretsManagerClient](#)。このコンストラクタを使用して、Secrets Manager クライアントをカスタマイズします (特定のリージョンまたはエンドポイントを使用するなど)。

パラメータ

`secretsManager`

シークレット [AmazonSecretsManagerClient](#) を取得する。

```
public SecretsManagerCache(SecretCacheConfiguration config)
```

提供された [the section called “SecretCacheConfiguration”](#) を使用して、新しいシークレット キャッシュを構築します。このコンストラクタを使用してキャッシュを設定します (キャッシュするシークレットの数や更新頻度など)。

パラメータ

config

キャッシュの設定情報が含まれている [the section called “SecretCacheConfiguration”](#)。

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

提供された [AmazonSecretsManagerClient](#) とを使用して作成された Secrets Manager クライアントを使用して新しいキャッシュを構築します [the section called “SecretCacheConfiguration”](#)。このコンストラクタを使用して Secrets Manager クライアントをカスタマイズし (特定のリージョンまたはエンドポイントを使用するなど)、キャッシュを構成します (キャッシュするシークレットの数や更新頻度など)。

パラメータ

secretsManager

シークレット [AmazonSecretsManagerClient](#) を取得する。

config

キャッシュの設定情報が含まれている [the section called “SecretCacheConfiguration”](#)。

方法

GetSecretString

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

Secrets Manager から文字列シークレットを取得します。

パラメータ

secretId

取得するシークレットの ARN または名前。

GetSecretBinary

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

Secrets Manager からバイナリシークレットを取得します。

パラメータ

secretId

取得するシークレットの ARN または名前。

RefreshNowAsync

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

Secrets Manager からのシークレット値をリクエストし、変更があればキャッシュを更新します。既存のキャッシュエントリがない場合は、新しいキャッシュエントリを作成します。更新に成功した場合は、true を返します。

パラメータ

secretId

取得するシークレットの ARN または名前。

GetCachedSecret

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

指定されたシークレットのキャッシュエントリがキャッシュに存在する場合、そのキャッシュエントリを返します。それ以外の場合は、Secrets Manager からシークレットを取得し、新しいキャッシュエントリを作成します。

パラメータ

secretId

取得するシークレットの ARN または名前。

SecretCacheConfiguration

キャッシュされるシークレットの最大キャッシュサイズや有効期限 (TTL) などの、[the section called “SecretsManagerCache”](#) のキャッシュ設定オプション。

プロパティ

CacheItemTTL

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

キャッシュ項目の TTL (ミリ秒単位)。デフォルトは 3600000 ミリ秒 (1 時間) です。最大値は 4294967295 ms で、約 49.7 日です。

MaxCacheSize

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

最大キャッシュサイズ。デフォルトは 1,024 個のシークレットです。最大値は 65,535 です。

VersionStage

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

キャッシュするシークレットのバージョン。詳細については、「[Secret versions](#)」(シークレットバージョン) を参照してください。デフォルトは "AWSCURRENT" です。

クライアント

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

シークレット [AmazonSecretsManagerClient](#) を取得する。null の場合、キャッシュは新しいクライアントをインスタンス化します。デフォルトは null です。

CacheHook

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

[the section called “ISecretCacheHook”](#)。

ISecretCacheHook

[the section called “SecretsManagerCache”](#) に接続して、キャッシュに保存されているシークレットに対してアクションを実行するインターフェイス。

方法

プット

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

キャッシュに保存するオブジェクトを準備します。

キャッシュに保存するオブジェクトを返します。

Get

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

キャッシュされたオブジェクトからオブジェクトを派生させます。

キャッシュから返すオブジェクトを返します

.NET AWS SDK を使用して Secrets Manager のシークレット値を取得する

アプリケーションでは、任意の SDK `BatchGetSecretValue` で `GetSecretValue` または `BatchGetSecretValue` を呼び出してシークレットを取得できます。AWS SDKs ただし、シークレット値はクライアント側のキャッシュを使用してキャッシュすることをお勧めします。シークレットをキャッシュすることで、速度が向上し、コストを削減できます。

.NET アプリケーションの場合は、[Secrets Manager の .NET ベースのキャッシュコンポーネント](#) を使用するか、[GetSecretValue](#) または [BatchGetSecretValue](#) を使用して SDK を直接呼び出します。

以下のコード例は、`GetSecretValue` の使用方法を示しています。

必要な許可: `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 GetSecretValueResponse 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;
    }
}
}
```

Go を使用して Secrets Manager のシークレット値を取得する

アプリケーションでは、任意の SDK BatchGetSecretValue で GetSecretValue または を呼び出してシークレットを取得できます。AWS SDKs ただし、シークレット値はクライアント側のキャッシュを使用してキャッシュすることをお勧めします。シークレットをキャッシュすることで、速度が向上し、コストを削減できます。

トピック

- [Go とクライアント側のキャッシュを使用して Secrets Manager のシークレット値を取得する](#)
- [Go AWS SDK を使用して Secrets Manager のシークレット値を取得する](#)

Go とクライアント側のキャッシュを使用して Secrets Manager のシークレット値を取得する

シークレットを取得するときに、Secrets Manager の GO ベースのキャッシュコンポーネントを使用して、将来の使用のためにキャッシュすることができます。キャッシュされたシークレットの取得は、Secrets Manager からの取得よりも高速です。Secrets Manager API を呼び出すにはコストがかかるため、キャッシュを使用するとコストを削減できます。シークレットを取得するすべての方法については、「[シークレットを取得する](#)」を参照してください。

キャッシュポリシーは LRU (最近最も使われていない) であるため、キャッシュでシークレットを破棄する必要が生じた場合は、最も最近使われていないシークレットが破棄されます。デフォルトでは、1 時間ごとにキャッシュでシークレットが更新されます。キャッシュで[シークレットが更新される頻度](#)を設定できるだけでなく、[シークレットの取得にフック](#)させて機能を追加することもできます。

キャッシュ参照が解放されると、キャッシュはガベージコレクションを強制しません。キャッシュの実装には、キャッシュの無効化は含まれていません。キャッシュを実装するのはキャッシュを使用するためであり、セキュリティを強化するためでもセキュリティに焦点を当てるためでもありません。キャッシュ内のアイテムを暗号化するなど、セキュリティを強化する必要がある場合は、所定のインターフェイスと抽象メソッドを使用してください。

このコンポーネントを使用するには、以下が必要です。

- AWS SDK for Go。 [the section called “AWS SDKs”](#) を参照してください。

ソースコードをダウンロードするには、「」の [「Secrets Manager Go キャッシュクライアント」](#) を参照してください GitHub。

Go の開発環境を設定するには、Go プログラミング言語のウェブサイトの [「Golang Getting Started」](#) (Go 言語の使用開始) を参照してください。

必要な許可:

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

詳細については、 [「アクセス許可に関するリファレンス」](#) を参照してください。

リファレンス

- [type Cache](#)
- [タイプ CacheConfig](#)
- [タイプ CacheHook](#)

Example シークレットを取得する

次のコード例は、シークレットを取得する Lambda 関数を示しています。

```
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

Secrets Manager からリクエストされたシークレットのインメモリキャッシュ。 [the section called “GetSecretString”](#) または [the section called “GetSecretBinary”](#) を使用して、キャッシュからシークレットを取得します。

次に、キャッシュの設定方法の例を示します。

```
// 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 },
)
```

詳細と例については、「[the section called “クライアント側のキャッシュを使用する”](#)」を参照してください。

方法

New

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

New は機能オプションを使用してシークレットキャッシュを構築します。それ以外の場合はデフォルトが使用されます。新しいセッションから SecretsManager クライアントを初期化します。をデフォルト値 CacheConfig に初期化します。LRU キャッシュをデフォルトの最大サイズで初期化します。

GetSecretString

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

GetSecretString は、指定されたシークレット ID のシークレット文字列値をキャッシュから取得します。シークレット文字列を返し、オペレーションが失敗した場合はエラーを返します。

GetSecretStringWithStage

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

GetSecretStringWithStage は、指定されたシークレット ID と [バージョンステージ](#) のシークレット文字列値をキャッシュから取得します。シークレット文字列を返し、オペレーションが失敗した場合はエラーを返します。

GetSecretBinary

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

GetSecretBinary は、指定されたシークレット ID のシークレットバイナリ値をキャッシュから取得します。シークレットバイナリを返し、オペレーションが失敗した場合はエラーを返します。

GetSecretBinaryWithStage

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

GetSecretBinaryWithStage は、指定されたシークレット ID と [バージョンステージ](#) のキャッシュからシークレットバイナリ値を取得します。シークレットバイナリを返し、オペレーションが失敗した場合はエラーを返します。

タイプ CacheConfig

キャッシュされるシークレットの最大キャッシュサイズ、デフォルト[バージョンステージ](#)、および有効期限 (TTL) などの、[Cache](#) のキャッシュ設定オプション。

```
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  
}
```

タイプ CacheHook

[Cache](#) に接続して、キャッシュに保存されているシークレットに対してアクションを実行するインターフェイス。

方法

プット

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

キャッシュに保存するオブジェクトを準備します。

Get

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

キャッシュされたオブジェクトからオブジェクトを派生させます。

Go AWS SDK を使用して Secrets Manager のシークレット値を取得する

アプリケーションでは、任意の SDK `BatchGetSecretValue` で `GetSecretValue` または `BatchGetSecretValue` を呼び出してシークレットを取得できます。AWS SDKs ただし、シークレット値はクライアント側のキャッシュを使用してキャッシュすることをお勧めします。シークレットをキャッシュすることで、速度が向上し、コストを削減できます。

Go アプリケーションの場合は、[Secrets Manager の Go ベースのキャッシュコンポーネント](#) を使用するか、[GetSecretValue](#) または [BatchGetSecretValue](#) を使用して SDK を直接呼び出します。

次の例で、Secrets Manager シークレットの値を取得する方法を示します。

必要な許可: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.
}

```

C++ AWS SDK を使用して Secrets Manager のシークレット値を取得する

C++ アプリケーションの場合は、[GetSecretValue](#) または [BatchGetSecretValue](#) を使用して SDK を直接呼び出します。

次の例で、Secrets Manager シークレットの値を取得する方法を示します。

必要な許可: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 SDK を使用して Secrets Manager のシークレット値を取得する

JavaScript アプリケーションの場合は、[getSecretValue](#) または [batchGetSecretValue](#) を使用して SDK を直接呼び出します。

次の例で、Secrets Manager シークレットの値を取得する方法を示します。

必要な許可: `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 SDK を使用して Secrets Manager のシークレット値を取得する

Kotlin アプリケーションの場合は、[GetSecretValue](#) または [BatchGetSecretValue](#) を使用して SDK を直接呼び出します。

次の例で、Secrets Manager シークレットの値を取得する方法を示します。

必要な許可: `secretsmanager:GetSecretValue`

```
suspend fun getValue(secretName: String?) {
```

```
val valueRequest = GetSecretValueRequest {
    secretId = secretName
}

SecretsManagerClient { region = "us-east-1" }.use { secretsClient ->
    val response = secretsClient.getSecretValue(valueRequest)
    val secret = response.secretString
    println("The secret value is $secret")
}
}
```

PHP AWS SDK を使用して Secrets Manager のシークレット値を取得する

PHP アプリケーションの場合は、[GetSecretValue](#) または [BatchGetSecretValue](#) を使用して SDK を直接呼び出します。

次の例で、Secrets Manager シークレットの値を取得する方法を示します。

必要な許可: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 SDK を使用して Secrets Manager のシークレット値を取得する

Ruby アプリケーションの場合は、[get_secret_value](#) または [batch_get_secret_value](#) を使用して SDK を直接呼び出します。

次の例で、Secrets Manager シークレットの値を取得する方法を示します。

必要な許可: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
end
```

Rust AWS SDK を使用して Secrets Manager のシークレット値を取得する

Rust アプリケーションの場合は、[GetSecretValue](#) または [BatchGetSecretValue](#) を使用して SDK を直接呼び出します。

次の例で、Secrets Manager シークレットの値を取得する方法を示します。

必要な許可: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(())
}
```

を使用してシークレット値を取得する AWS CLI

必要な許可:secretsmanager:GetSecretValue

Example シークレットの暗号化されたシークレット値を取得する

次の [get-secret-value](#) の例では、現在のシークレット値が取得されます。

```
aws secretsmanager get-secret-value \  
  --secret-id MyTestSecret
```

Example 前のシークレット値を取得する

次の [get-secret-value](#) の例では、前のシークレット値が取得されます。

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

を使用してバッチでシークレットのグループを取得する AWS CLI

必要な許可:

- `secretsmanager:BatchGetSecretValue`
- `secretsmanager:GetSecretValue` 取得する各シークレットの アクセス許可。
- フィルターを使用する場合は、`secretsmanager:ListSecrets` も必要です。

アクセス許可ポリシーの例については、「[the section called “例: バッチ内のシークレット値のグループを取得するアクセス許可”](#)」を参照してください。

Important

取得しようとしているグループ内の個別シークレットを取得するためのアクセス許可を拒否する VPCE ポリシーを設定している場合、`BatchGetSecretValue` はシークレット値を返さず、エラーが返されます。

Example 名前順に表示されたシークレットグループのシークレット値を取得する

次の [batch-get-secret-value](#) の例では、3 つのシークレットのシークレット値を取得します。

```
aws secretsmanager batch-get-secret-value \  
  --secret-id MyTestSecret
```

```
--secret-id-list MySecret1 MySecret2 MySecret3
```

Example フィルターで選択したシークレットグループのシークレット値を取得する

次の [batch-get-secret-value](#) の例では、「Test」という名前のタグが付いたシークレットのシークレット値を取得します。

```
aws secretsmanager batch-get-secret-value \  
  --filters Key="tag-key",Values="Test"
```

AWS コンソールを使用してシークレット値を取得する

シークレットを取得するには (コンソール)

1. Secrets Manager のコンソール (<https://console.aws.amazon.com/secretsmanager/>) を開きます。
2. シークレットのリストで、取得するシークレットを選択します。
3. [シークレット値] セクションで、[シークレット値の取得] を選択します。

Secrets Manager にシークレットの現在のバージョン (AWSCURRENT) が表示されます。シークレットの [他のバージョン](#) (AWSPREVIOUS やカスタムラベル付きバージョンなど) を表示するには、[the section called "AWS CLI"](#) を使用します。

AWS Secrets Manager で AWS Batch シークレットを使用する

AWS Batch を使用すると、AWS クラウド でバッチコンピューティングワークロードを実行できます。AWS Batch を使用すると、AWS Secrets Manager シークレットに機密データを保存してジョブ定義でそれを参照することによって、ジョブの機密のデータを挿入できます。詳細については、[「Secrets Manager を使用した機密データの指定」](#) を参照してください。

リソースで AWS Secrets Manager AWS CloudFormation シークレットを取得する

では AWS CloudFormation、別の AWS CloudFormation リソースで使用するシークレットを取得できます。一般的なシナリオでは、まずシークレットマネージャーが生成したパスワードを使ってシークレットを作成し、次に、シークレットから、新しいデータベースの認証情報として使用する

るユーザーネームとパスワードを取得します。でシークレットを作成する方法については、AWS CloudFormation「」を参照してください[AWS CloudFormation](#)。

AWS CloudFormation テンプレート内のシークレットを取得するには、動的リファレンスを使用します。スタックを作成すると、動的参照によってシークレット値が AWS CloudFormation リソースにプルされるため、シークレット情報をハードコーディングする必要はありません。代わりに、シークレットを名前または ARN で参照します。ダイナミックリファレンスは、任意のリソースプロパティ内のシークレットに対し使用できます。リソースメタデータ ([AWS::CloudFormation::Init](#) など) 内にあるシークレットに対しては、ダイナミックリファレンスを使用することはできません。これにより、シークレット値がコンソールに表示されてしまうためです。

シークレットに対するダイナミックリファレンスのパターンを、次に示します。

```
{{resolve:secretsmanager:secret-id:SecretString:json-key:version-stage:version-id}}
```

シークレットID

シークレット名またはシークレット ARN。AWS アカウントのシークレットにアクセスするには、シークレット名を使用できます。別の AWS アカウントのシークレットにアクセスするには、シークレットの ARN を使用します。

json-key (オプション)

値を取得するペアのキー名を指定します。を指定しない場合 json-key、 はシークレットテキスト全体 AWS CloudFormation を取得します。このセグメントにはコロン文字 (:) を含めることはできません。

version-stage (オプション)

使用するクエリの[バージョン](#)シークレットマネージャーは、ステージングラベルがローテーション処理中にさまざまなバージョンを追跡するために使用されます。version-stage を使用する場合は、version-id を指定することはできません。version-stage または version-id、を指定しない場合、デフォルトでは AWSCURRENT というラベルの付いたバージョンが取得されます。このセグメントにはコロン文字 (:) を含めることはできません。

version-id (オプション)

使用したいシークレットのバージョンの固有識別子を指定します。version-id を指定した場合は、version-stage を指定しないでください。version-stage または version-id を指定しない場合、次にデフォルトでは AWSCURRENT というバージョンが取得されます。このセグメントにはコロン文字 (:) を含めることはできません。

詳細については、「[シークレットマネージャーのシークレットを指定するための、ダイナミックレファレンスを使用する](#)」を参照してください。

Note

最終値(\)としてバックスラッシュを使用して動的参照を作成しないでください。はこれらの参照を解決 AWS CloudFormation できないため、リソースに障害が発生します。

Amazon Elastic Kubernetes Service で AWS Secrets Manager シークレットを使用する

Secrets Manager からのシークレットを [Amazon EKS](#) ポッドにマウントされたファイルとして表示するには、Kubernetes AWS Secrets [Store CSI ドライバーのシークレットと設定プロバイダー \(ASCP\)](#) を使用できます。ASCP は、Amazon EC2 ノードグループを実行する Amazon Elastic Kubernetes Service (Amazon EKS) 1.17 以降で動作します。AWS Fargate ノードグループはサポートされていません。ASCP を使用すると、Secrets Manager でシークレットを保存および管理し、Amazon EKS で実行されているワークロードからシークレットを取得できます。シークレットに JSON 形式の複数のキー/値ペアが含まれている場合は、Amazon EKS にマウントするキー/値ペアを選択できます。ASCP は [JMESPath 構文](#) を使用して、シークレット内のキー/値のペアをクエリします。ASCP は [パラメータストアパラメータ](#) を使用しても動作します。

プライベート Amazon EKS クラスターを使用する場合は、クラスターがある VPC に、Secrets Manager エンドポイントが存在している必要があります。Secrets Store CSI ドライバーは、このエンドポイントを使用して Secrets Manager を呼び出します。VPC エンドポイントの作成については、「[VPC エンドポイント](#)」を参照してください。

シークレットに Secrets Manager の自動ローテーションを使用する場合は、シークレットストア CSI ドライバーのローテーション調整機能を使用して、Secrets Manager から最新のシークレットを取得することもできます。詳細については、「[マウントされたコンテンツと同期された Kubernetes シークレットの自動ローテーション](#)」を参照してください。

トピック

- [ステップ 1: アクセス制御を設定する](#)
- [ステップ 2: ASCP をインストールして設定する](#)
- [ステップ 3: マウントするシークレットを特定する](#)

- [ステップ 4: Amazon EKS ポッドにシークレットをファイルとしてマウントする](#)
- [トラブルシューティング](#)
- [SecretProviderClass](#)

ステップ 1: アクセス制御を設定する

ASCP は Amazon EKS ポッド ID を取得し、IAM ロールと交換します。その IAM ロールの IAM ポリシーでアクセス許可を設定します。ASCP が IAM ロールを引き受けると、承認したシークレットにアクセスできます。他のコンテナは、IAM ロールに関連付けられていない限り、シークレットにアクセスできません。

ポッドに関連付けられたリージョンと IAM ロールを検索するための ASCP からの呼び出しが Kubernetes によってスロットリングされている場合、ステップ 2 に示すように `helm install`、を使用してスロットリングクォータを変更できます。

Amazon EKS ポッドに Secrets Manager のシークレットへのアクセスを許可するには

1. ポッドがアクセスする必要があるシークレットに `secretsmanager:GetSecretValue` とアクセス許可を付与するアクセス `secretsmanager:DescribeSecret` 許可ポリシーを作成します。ポリシーの例については、「[the section called “例: 個々のシークレットを読み書きするアクセス許可”](#)」を参照してください。
2. クラスター用に IAM OpenID Connect (OIDC) プロバイダーがない場合は作成します。詳細については、「Amazon EKS [ユーザーガイド](#)」の「[クラスターの IAM OIDC プロバイダーを作成する](#)」を参照してください。
3. [サービスアカウントの IAM ロール](#) を作成し、ポリシーをアタッチします。詳細については、「Amazon EKS [ユーザーガイド](#)」の「[サービスアカウントの IAM ロールを作成する](#)」を参照してください。
4. プライベート Amazon EKS クラスターを使用する場合は、クラスターがある VPC に AWS STS エンドポイントがあることを確認します。エンドポイントの作成の詳細については、「[ユーザーガイド](#)」の「[インターフェイス VPC エンドポイント](#) AWS Identity and Access Management」を参照してください。

ステップ 2: ASCP をインストールして設定する

ASCP は、[secrets-store-csi-provider-aws](#) リポジトリの GitHub で使用できます。リポジトリには、シークレットを作成してマウントするための YAML ファイルの例も含まれています。

インストール中に、FIPS エンドポイントを使用するように ASCP を設定できます。; エンドポイントのリストについては、「[the section called “Secrets Manager エンドポイント”](#)」を参照してください。

Helm を使用して ASCP をインストールするには

1. リポジトリが最新のチャートを指していることを確認するには、を使用します。helm repo update.
2. Secrets Store CSI ドライバーチャートを追加します。

```
helm repo add secrets-store-csi-driver https://kubernetes-sigs.github.io/secrets-store-csi-driver/charts
```

3. グラフをインストールします。スロットリングを設定するには、次のフラグを追加します。--set-json 'k8sThrottlingParams={"qps": "<number of queries per second>", "burst": "<number of queries per second>"}'

```
helm install -n kube-system csi-secrets-store secrets-store-csi-driver/secrets-store-csi-driver
```

4. ASCP チャートを追加します。

```
helm repo add aws-secrets-manager https://aws.github.io/secrets-store-csi-driver-provider-aws
```

5. グラフをインストールします。FIPS エンドポイントを使用するには、次のフラグを追加します。--set useFipsEndpoint=true

```
helm install -n kube-system secrets-provider-aws aws-secrets-manager/secrets-store-csi-driver-provider-aws
```

リポジトリで YAML を使用して をインストールするには

- 次のコマンドを使用します。

```
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
```

ステップ 3: マウントするシークレットを特定する

ASCP が (ファイルシステム上のファイルとして) Amazon EKS 内にマウントするシークレットを判別するには、[the section called “SecretProviderClass”](#) YAML ファイルを作成します。には、マウントするシークレットと、マウントするファイル名が SecretProviderClass 一覧表示されます。SecretProviderClass は、参照する Amazon EKS ポッドと同じ名前空間にある必要があります。

以下の例は、SecretProviderClass を使用して、マウントするシークレットと、Amazon EKS ポッドにマウントされたファイルの名前を記述する方法を示しています。

例:

- [例: 名前または ARN でシークレットをマウントする](#)
- [例: シークレットからキーと値のペアをマウントする](#)
- [例: マルチリージョンシークレットのフェイルオーバーリージョンを定義する](#)
- [例: マウントするフェイルオーバーシークレットを選択する](#)

例: 名前または ARN でシークレットをマウントする

以下は、Amazon EKS で 3 ファイルをマウントする SecretProviderClass の例です。

1. 完全な ARN によって指定されたシークレット
2. 名前で指定されたシークレット
3. シークレットの特定のバージョン

```
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-2:111122223333:secret:MySecret2-d4e5f6"
- objectName: "MySecret3"
  objectType: "secretsmanager"
- objectName: "MySecret4"
  objectType: "secretsmanager"
  objectVersionLabel: "AWSCURRENT"
```

例: シークレットからキーと値のペアをマウントする

以下は、Amazon EKS で 3 ファイルをマウントする SecretProviderClass の例です。

1. 完全な ARN によって指定されたシークレット
2. 同じシークレットから取得した username キー/値のペア
3. 同じシークレットから取得した password キー/値のペア

```
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-2:111122223333:secret:MySecret-a1b2c3"
        jmesPath:
          - path: username
            objectAlias: dbusername
          - path: password
            objectAlias: dbpassword
```

例: マルチリージョンシークレットのフェイルオーバーリージョンを定義する

接続停止時あるいはディザスタリカバリ設定において可用性を確保するために、ASCP では自動フェイルオーバー機能により、セカンダリリージョンからシークレットを取得することができます。

次に、複数のリージョンにレプリケートされたシークレットを SecretProviderClass により取得する例を示します。この例では、ASCP が us-east-1 と us-east-2 の両方からシークレットの取得を試行します。どちらかのリージョンが認証の問題などで 4xx エラーを返した場合、ASCP

はいずれのシークレットもマウントしません。シークレットが us-east-1 から正常に取得されると、ASCP はそのシークレット値をマウントします。us-east-1 からはシークレットが正常に取得されないものの、us-east-2 からは正常に取得された場合、ASCP は取得された方のシークレット値をマウントします。

```
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"
```

例: マウントするフェイルオーバーシークレットを選択する

次に、フェイルオーバー時にマウントするシークレットを SecretProviderClass により指定する例を示します。フェイルオーバーシークレットとレプリカは異なります。この例では、ASCP が objectName で指定された 2 つのシークレットの取得を試行します。どちらかが認証の問題などで 4xx エラーを返した場合、ASCP はいずれのシークレットもマウントしません。シークレットが us-east-1 から正常に取得されると、ASCP はそのシークレット値をマウントします。us-east-1 からはシークレットが正常に取得されないものの、us-east-2 からは正常に取得された場合、ASCP は取得された方のシークレット値をマウントします。Amazon EKS にマウントされたファイルには、MyMountedSecret という名前が付けられています。

```
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-1:111122223333:secret:MySecret-a1b2c3"
        objectAlias: "MyMountedSecret"
```

```
failoverObject:
  - objectName: "arn:aws:secretsmanager:us-
east-2:111122223333:secret:MyFailoverSecret-d4e5f6"
```

ステップ 4: Amazon EKS ポッドにシークレットをファイルとしてマウントする

Amazon EKS にシークレットをマウントするには

1. コマンド `SecretProviderClass` を使用して をポッドに適用します `kubectl apply -f ExampleSecretProviderClass.yaml`。
2. コマンド を使用してポッドをデプロイします `kubectl apply -f ExampleDeployment.yaml`。
3. ASCP はファイルをマウントします。

トラブルシューティング

ポッドデプロイを記述すると、ほとんどのエラーを表示できます。

コンテナのエラーメッセージを表示するには

1. 次のコマンドで、ポッド名のリストを取得します。デフォルトの名前空間を使用していない場合は、`-n <NAMESPACE>` を使用してください。

```
kubectl get pods
```

2. ポッドを記述するには、次のコマンドで、前のステップで見つけたポッドのポッド ID を使用します (`<PODID>` の場合)。デフォルトの名前空間を使用していない場合は、`-n <NAMESPACE>` を使用してください。

```
kubectl describe pod/<PODID>
```

ASCP のエラーを表示するには

- プロバイダーログで詳細情報を確認するには、次のコマンドで、`<PODID>` に `csi-secrets-store-provider-aws` ポッドの ID を使用します。

```
kubectl -n kube-system get pods
kubectl -n kube-system logs pod/<PODID>
```

SecretProviderClass

YAML を使用して、[ASCP を使用して Amazon EKS にマウントするシークレットを記述します](#)。例については、「[the section called “名前または ARN によるシークレットをマウントする”](#)」を参照してください。

```
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: <NAME>
spec:
  provider: aws
  parameters:
    region:
    failoverRegion:
    pathTranslation:
    objects:
```

parameters には、以下のマウントリクエストの詳細が含まれます。

region

(オプション) シークレット AWS リージョンの。このフィールドを使用しない場合、ASCP はノード上のアノテーションからリージョンを検索します。この検索では、マウントリクエストにオーバーヘッドが追加されるため、大量のポッドを使用するクラスターでリージョンを指定することをお勧めします。

同時に failoverRegion も指定すると、ASCP は両方のリージョンからシークレットを試行します。どちらかのリージョンが認証の問題などで 4xx エラーを返した場合、ASCP はいずれのシークレットもマウントしません。シークレットが region から正常に取得されると、ASCP はそのシークレット値をマウントします。region からはシークレットが正常に取得されないものの、failoverRegion からは正常に取得された場合、ASCP は取得された方のシークレット値をマウントします。

failoverRegion

(オプション) このフィールドを含めると、ASCP は region で定義されているリージョンと、このフィールドで定義されているリージョンからのシークレット取得を試行します。どちらかのリージョンが認証の問題などで 4xx エラーを返した場合、ASCP はいずれのシークレットもマウントしません。シークレットが region から正常に取得されると、ASCP はそのシークレット値をマウントします。region からはシークレットが正常に取得されないものの、failoverRegion からは正常に取得された場合、ASCP は取得された方のシークレット値をマウントします。このフィールドの使用例については、「[マルチリージョンシークレットのフェイルオーバーリージョンを定義する](#)」を参照してください。

pathTranslation

(オプション) Amazon EKS のファイル名に、Linux のスラッシュ (/) のようなパス区切り文字が含まれている場合に使用する単一の置換文字。ASCP では、パス区切り文字が含まれているファイルを作成し、マウントすることはできません。代わりに、ASCP はパス区切り文字を別の文字に置き換えます。このフィールドを使用しない場合、置換文字にはアンダースコア (_) が使用されます。例えば、My/Path/Secret は My_Path_Secret としてマウントされます。

文字の置換を防ぐには、文字列 False を入力してください。

objects

マウントするシークレットの YAML 宣言を含む文字列 YAML 複数行の文字列またはパイプ (|) 文字を使用することをお勧めします。

objectName

シークレットの名前または完全な ARN ARN を使用する場合、objectType は省略できます。objectAlias を指定しない限り、このフィールドが Amazon EKS ポッドのシークレットのファイル名として使用されます。ARN を使用する場合、ARN 内のリージョンはフィールド region と一致する必要があります。failoverRegion を含めると、このフィールドはプライマリの objectName になります。

objectType

objectName で Secrets Manager ARN を使用しない場合は必須。secretsmanager または ssmparameter のいずれかになります。

objectAlias

(オプション) Amazon EKS ポッド内のシークレットのファイル名 このフィールドを指定しなかった場合、objectName がファイル名として表示されます。

objectVersion

(オプション) シークレットのバージョン ID。バージョン ID は、シークレットを変更するたびに更新の必要が生じるため、これは推奨されません。デフォルトでは、最新バージョンが使用されます。failoverRegion を含めると、このフィールドはプライマリの objectVersion になります。

objectVersionLabel

(オプション) バージョンのエイリアス。デフォルトは最新バージョンです AWSCURRENT。詳細については、「[the section called “シークレットバージョン”](#)」を参照してください。failoverRegion を含めると、このフィールドはプライマリの objectVersionLabel になります。

jmesPath

(オプション) シークレット内のキーから Amazon EKS にマウントされるファイルへのマッピング。このフィールドを使用するには、シークレット値を JSON 形式にする必要があります。このフィールドを使用する場合は、サブフィールド path および objectAlias を含める必要があります。

パス

シークレット値の JSON 内のキー/値のペアからのキー。フィールドにハイフンが含まれている場合は、一重引用符でエスケープします。例: path: '"hyphenated-path"'

objectAlias

Amazon EKS ポッドにマウントされるファイル名。フィールドにハイフンが含まれている場合は、一重引用符でエスケープします。例: objectAlias: '"hyphenated-alias"'

failoverObject

(オプション) このフィールドを指定した場合、ASCP はプライマリ objectName で指定されたシークレットと、failoverObject、objectName サブフィールドで指定されたシークレットの両方の取得を試行します。どちらかが認証の問題などで 4xx エラーを返した場合、ASCP はいずれのシークレットもマウントしません。シークレットがプライマリ objectName から正常に取得されると、ASCP はそのシークレット値のマウントを行います。シークレットが、プライマリ objectName からは正常に取得されなかったものの、フェイルオーバー objectName からは正常に取得された場合、ASCP はその (取得された) シークレット値をマウントします。このフィールドを含める場合は、フィールド objectAlias も含める必要があります。このフィールドの使用例については、「[マウントするフェイルオーバーシークレットを選択する](#)」を参照してください。

通常、このフィールドはフェイルオーバーシークレットがレプリカではない場合に使用します。レプリカを指定する際の例については、「[マルチリージョンシークレットのフェイルオーバーリージョンを定義する](#)」を参照してください。

objectName

フェイルオーバーシークレットの名前または完全な ARN を使用する場合、ARN 内のリージョンはフィールド `failoverRegion` と一致する必要があります。

objectVersion

(オプション) シークレットのバージョン ID プライマリ `objectVersion` と一致している必要があります。バージョン ID は、シークレットを変更するたびに更新の必要が生じるため、これは推奨されません。デフォルトでは、最新バージョンが使用されます。

objectVersionLabel

(オプション) バージョンのエイリアス デフォルトは最新バージョン です `AWSCURRENT`。詳細については、「[the section called “シークレットバージョン”](#)」を参照してください。

GitHub ジョブで AWS Secrets Manager シークレットを使用する

GitHub ジョブでシークレットを使用するには、GitHub アクションを使用してからシークレットを取得し AWS Secrets Manager、GitHub ワークフローでマスクされた[環境変数](#)として追加できます。GitHub アクションの詳細については、「[ドキュメント](#)」の [GitHub](#) 「アクションについて」を参照してください。GitHub

GitHub 環境にシークレットを追加すると、GitHub ジョブ内の他のすべてのステップでシークレットを使用できます。[GitHub](#) 「[アクションのセキュリティ強化](#)」のガイダンスに従って、環境内のシークレットが悪用されるのを防ぎます。

シークレット値の文字列全体を環境変数値として設定できます。また、文字列が JSON の場合は、JSON を解析して JSON キーと値のペアごとに環境変数を設定することもできます。シークレット値がバイナリの場合は文字列に変換されます。

シークレットから作成された環境変数を表示するには、デバッグログを有効にします。詳細については、「[Docs](#) [でのデバッグログの有効化](#)」を参照してください。GitHub

シークレットから作成された環境変数を使用するには、Docs の [「環境変数」](#) を参照してください。GitHub

前提条件

このアクションを使用するには、まず AWS 認証情報を設定し、`configure-aws-credentials` ステップを使用して AWS リージョン GitHub 環境で を設定する必要があります。「[Configure AWS Credentials Action For GitHub Actions to Assume role directly using GitHub OIDC provider](#)」の手順に従ってください。これにより、有効期間の短い認証情報を使用でき、追加のアクセスキーを Secrets Manager の外部に保存する必要がなくなります。

アクションが引き受ける IAM ロールには、以下のアクセス許可が必要です。

- 取得するシークレットに対する `GetSecretValue`
- すべてのシークレットに対する `ListSecrets`
- (オプション) シークレットが で暗号化 KMS key されている場合は、`Decrypt` のカスタマー管理キー。

詳細については、「[認証とアクセスコントロール](#)」を参照してください。

使用方法

アクションを使用するには、次の構文を使用するステップをワークフローに追加します。

```
- 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
```

パラメータ

secret-ids

シークレット ARN、名前、名前プレフィックス。

環境変数名を設定するには、まず環境変数名を入力し、その後にシークレット ID とカンマを順に入力します。例えば、`ENV_VAR_1`, `secretId` はシークレット `secretId` から `ENV_VAR_1` という名前の環境変数を作成します。環境変数名には、大文字の英字、数字、およびアンダースコアを使用できます。

プレフィックスを使用するには、3文字以上を入力し、その後にアスタリスクを付けます。例えば、`dev*` は名前が `dev` で始まるすべてのシークレットに一致します。取得できる一致シークレットの最大数は 100 です。変数名を設定し、プレフィックスが複数のシークレットと一致する場合、アクションは失敗します。

`name-transformation`

デフォルトでは、このステップはシークレット名から各環境変数名を作成し、大文字の英字、数字、アンダースコアのみが含まれるように変換し、数字で始まらないようにします。名前の文字には、`lowercase` で小文字を使用するか、`uppercase` で文字の大文字と小文字を変更 `lowercase` しないようにステップを設定できます `none`。デフォルト値は、`uppercase` です。

`parse-json-secrets`

(オプション) デフォルトでは、環境変数値がシークレット値内の JSON 文字列全体に設定されません。`parse-json-secrets` を `true` に設定すると、JSON 内のキー/値のペアごとに環境変数が作成されます。

「`name`」や「`Name`」など JSON で使用されるキーが大文字と小文字を区別する場合、アクション名が重複して競合することに注意してください。この場合は、`parse-json-secrets` を `false` に設定し、JSON シークレット値を別途解析します。

環境変数の命名

アクションによって作成された環境変数の名前は、シークレットの名前と同じです。環境変数の命名要件はシークレットよりも厳しいため、アクションはシークレット名を変換してこれらの要件を満たすようにします。例えば、小文字は大文字に変換されます。シークレットの JSON を解析する場合、環境変数名にはシークレット名と JSON キー名の両方が含まれます `MYSECRET_KEYNAME`。例えば、`secret-keyname`。小文字を変換しないようにアクションを設定できます。

2つの環境変数が同じ名前以終わる場合、アクションは失敗します。この場合、環境変数に使用する名前をエイリアスとして指定する必要があります。

名前が競合する可能性がある場合の例：

- 「`secret`」という名前のシークレット `MySecret` と「`secret`」という名前のシークレットはどちらも「`MYSECRET`」という名前の環境変数になります。
- 「`Secret_keyname`」という名前のシークレットと「`keyname`」という名前のキーを持つ「`Secret`」という名前の JSON 解析されたシークレットは、両方とも「`SECRET_KEYNAME`」という名前の環境変数になります。

環境変数名は、という名前の変数を作成する次の例に示すように、エイリアスを指定することで設定できますENV_VAR_NAME。

```
secret-ids: |
  ENV_VAR_NAME, secretId2
```

空白のエイリアス

- 空白のエイリアスを設定parse-json-secrets: trueして入力し、その後にカンマとシークレット ID を入力すると、アクションは解析された JSON キーと同じ名前を環境変数に付けます。変数名にはシークレット名は含まれません。

シークレットに有効な JSON が含まれていない場合、アクションは 1 つの環境変数を作成し、シークレット名と同じ名前を付けます。

- 空白のエイリアスを設定parse-json-secrets: falseして入力し、その後にカンマとシークレット ID を入力すると、アクションは環境変数にエイリアスを指定しなかったかのように名前を付けます。

次の例は、空白のエイリアスを示しています。

```
,secret2
```

例

Example 1 名前と ARN でシークレットを取得する

次の例では、名前と ARN で識別されるシークレットの環境変数を作成しています。

```
- 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
```

作成された環境変数

```
EXAMPLESECRETNAME: secretValue1
TEST1: secretValue2
_0_TEST_SECRET: secretValue3
_PROD_EXAMPLE_SECRET: secretValue4
SECRET_ALIAS_1: secretValue5
SECRET_ALIAS_2: secretValue6
SECRET2: secretValue7
```

Example 2 プレフィックスで始まるシークレットをすべて取得する

次の例では、名前が *beta* で始まるすべてのシークレットの環境変数を作成しています。

```
- name: Get Secret Names by Prefix
  uses: 2
  with:
    secret-ids: |
      beta*    # Retrieves all secrets that start with 'beta'
```

作成された環境変数

```
BETASECRETNAME: secretValue1
BETATEST: secretValue2
BETA_NEWSECRET: secretValue3
```

Example 3 シークレット内の JSON を解析する

次の例では、シークレット内の JSON を解析して環境変数を作成しています。

```
- 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
```

シークレット test/secret には、次のシークレット値があります。

```
{
```

```
"api_user": "user",
"api_key": "key",
"config": {
  "active": "true"
}
}
```

シークレット secret2 には、次のシークレット値があります。

```
{
  "myusername": "alejandro_rosalez",
  "mypassword": "EXAMPLE_PASSWORD"
}
```

作成された環境変数

```
TEST_SECRET_API_USER: "user"
TEST_SECRET_API_KEY: "key"
TEST_SECRET_CONFIG_ACTIVE: "true"
MYUSERNAME: "alejandro_rosalez"
MYPASSWORD: "EXAMPLE_PASSWORD"
```

Example 4 環境変数名には小文字を使用する

次の例では、小文字の名前で環境変数を作成します。

```
- name: Get secrets
  uses: aws-actions/aws-secretsmanager-get-secrets@v2
  with:
    secret-ids: exampleSecretName
    name-transformation: lowercase
```

作成された環境変数 :

```
examplesecretname: secretValue
```

AWS Secrets Manager で AWS IoT Greengrass シークレットを使用する

AWS IoT Greengrass は、クラウドの機能をローカルデバイスに拡張するソフトウェアです。これにより、デバイスは情報源に近いデータを収集および分析して、ローカルイベントに自動的に反応し、ローカルネットワークで互いに安全に通信することができます。

AWS IoT Greengrass では、パスワードやトークンなどのシークレットをハードコーディングすることなく、Greengrass デバイスからサービスやアプリケーションで認証できます。AWS Secrets Manager を使用して、シークレットをクラウド内に安全に保存して管理することができます。AWS IoT Greengrass は、Secrets Manager を Greengrass コアデバイスに拡張するため、コネクタおよび Lambda 関数ではローカルシークレットを使用してサービスやアプリケーションとやり取りすることができます。

シークレットを Greengrass グループ内に統合するには、Secrets Manager シークレットを参照するグループリソースを作成します。このシークレットリソースは、関連付けられた ARN を使用してクラウドシークレットを参照します。シークレットリソースを作成、管理、および使用方法については、「AWS IoT デベロッパーガイド」の「[シークレットリソースを使用する](#)」を参照してください。

シークレットを AWS IoT Greengrass コアにデプロイするには、「[AWS IoT Greengrass にシークレットをデプロイする](#)」を参照してください。

AWS Lambda 関数で AWS Secrets Manager シークレットを使用する

AWS Parameters and Secrets Lambda Extension を使用すると、SDK を使用せずに Lambda 関数で AWS Secrets Manager シークレットを取得してキャッシュできます。キャッシュされたシークレットの取得は、Secrets Manager からの取得よりも高速です。Secrets Manager API を呼び出すにはコストがかかるため、キャッシュを使用するとコストを削減できます。この拡張機能では、Secrets Manager のシークレットと Parameter Store のパラメータの両方を取得できます。Parameter Store については、「AWS Systems Manager ユーザーガイド」の「[Parameter Store integration with Lambda extensions](#)」(Parameter Store と Lambda 拡張機能の統合) を参照してください。

Lambda 拡張機能は、Lambda 関数の機能に追加するコンパニオンプロセスです。詳細については、「Lambda 開発者ガイド」の「[Lambda 拡張機能](#)」を参照してください。コンテナイメージでの拡張

機能の使用については、「[コンテナイメージ内で Lambda レイヤーと拡張機能を動作させる](#)」を参照してください。Lambda は、Amazon CloudWatch Logs を使用して、拡張機能の実行情報を関数とともにログに記録します。デフォルトでは、拡張機能は最小限の情報をに記録します CloudWatch。詳細をログに記録するには、[環境変数](#) `PARAMETERS_SECRETS_EXTENSION_LOG_LEVEL` を `debug` に設定します。

パラメータとシークレットのインメモリキャッシュを提供するために、拡張機能は、ローカル HTTP エンドポイント (localhost ポート 2773) を Lambda 環境に公開します。このポートを設定するには、[環境変数](#) `PARAMETERS_SECRETS_EXTENSION_HTTP_PORT` を設定します。

Lambda は、関数が必要とする同時実行レベルに対応する個別のインスタンスをインスタンス化します。各インスタンスは分離され、設定データの独自のローカルキャッシュが保持されます。Lambda インスタンスと同時実行の詳細については、「Lambda 開発者ガイド」の「[Lambda 関数の同時実行数の管理](#)」を参照してください。

ARM の拡張機能を追加するには、Lambda 関数の `arm64` アーキテクチャを使用する必要があります。詳細については、「Lambda 開発者ガイド」の「[Lambda 命令セットアーキテクチャ](#)」を参照してください。この拡張機能は、次のリージョンで ARM をサポートしています: アジアパシフィック (ムンバイ)、米国東部 (オハイオ)、欧州 (アイルランド)、欧州 (フランクフルト)、欧州 (チューリッヒ)、米国東部 (バージニア北部)、欧州 (ロンドン)、欧州 (スペイン)、アジアパシフィック (東京)、米国西部 (オレゴン)、アジアパシフィック (シンガポール)、アジアパシフィック (ハイデラバード)、およびアジアパシフィック (シドニー)。

拡張機能は AWS クライアントを使用します。AWS クライアントの設定の詳細については、SDK およびツール[リファレンスガイド](#)の「[設定リファレンス](#)」を参照してください。AWS Lambda 関数が VPC で実行されている場合、拡張機能が Secrets Manager を呼び出すことができるように VPC エンドポイントを作成する必要があります。詳細については、「[VPC エンドポイント](#)」を参照してください。

必要な許可:

- Lambda [実行ロール](#)には、シークレットに対する `secretsmanager:GetSecretValue` アクセス許可が必要です。
- シークレットがではなくカスタマーマネージドキーで暗号化されている場合 AWS マネージドキー `aws/secretsmanager`、実行ロールには KMS キーの `kms:Decrypt` アクセス許可も必要です。

Parameters AWS and Secrets Lambda Extension を使用するには

1. Parameters AWS and Secrets Lambda Extension という名前のAWS レイヤーを関数に追加します。手順については、「Lambda [デベロッパーガイド](#)」の「[関数へのレイヤーの追加](#)」を参照してください。
2. シークレットにアクセスできるアクセス許可を Lambda [実行ロール](#) に付与します。
 - シークレットの `secretsmanager:GetSecretValue` アクセス許可。 [the section called “例: 個別のシークレット値を取得するためのアクセス許可”](#) を参照してください。
 - (オプション) シークレットがではなくカスタマーマネージドキーで暗号化されている場合 AWS マネージドキー `aws/secretsmanager`、実行ロールには KMS キーの `kms:Decrypt` アクセス許可も必要です。
 - Lambda ロールで属性ベースのアクセス制御 (ABAC) を使用すると、アカウントのシークレットへのアクセスをきめ細かく制御できます。詳細については、「[the section called “例: タグを使用してシークレットへのアクセスを制御する”](#)」および「[the section called “例: シークレットのタグに一致しているタグを持つアイデンティティへのアクセスを制限する”](#)」を参照してください。
3. Lambda [環境変数](#) を使用してキャッシュを設定します。
4. 拡張キャッシュからシークレットを取得するには、まずリクエストヘッダーに `X-AWS-Parameters-Secrets-Token` を追加する必要があります。トークンを `AWS_SESSION_TOKEN` に設定します。これは実行中のすべての関数に対して Lambda によって提供されます。このヘッダーを使用すると、呼び出し元が Lambda 環境内にあることがわかります。

次の Python の例では、ヘッダーを追加する方法を示しています。

```
import os
headers = {"X-Aws-Parameters-Secrets-Token": os.environ.get('AWS_SESSION_TOKEN')}
```

5. Lambda 関数内のシークレットを取得するには、次の HTTP GET リクエストのいずれかを使用します。
 - `secretId` の場合、シークレットを取得するには、シークレット ARN または名前を使用します。

```
GET: /secretsmanager/get?secretId=secretId
```

- ステージングラベルで以前のシークレット値または特定のバージョンを取得するには、secretId の場合、シークレットの ARN または名前を使用し、versionStage の場合、ステージングラベルを使用します。

```
GET: /secretsmanager/get?secretId=secretId&versionStage=AWSPREVIOUS
```

- ID で特定のシークレットバージョンを取得するには、secretId の場合、シークレットの ARN または名前を使用し、versionId の場合、バージョン ID を使用します。

```
GET: /secretsmanager/get?secretId=secretId&versionId=versionId
```

Example シークレットを取得する (Python)

次の Python の例では、[json.loads](#) を使用してシークレットを取得し、その結果を解析する方法を示しています。

```
secrets_extension_endpoint = "http://localhost:" + \  
    secrets_extension_http_port + \  
    "/secretsmanager/get?secretId=" + \  
    <secret_name>  
  
r = requests.get(secrets_extension_endpoint, headers=headers)  
  
secret = json.loads(r.text)["SecretString"] # load the Secrets Manager response  
into a Python dictionary, access the secret
```

AWS Parameters and Secrets Lambda Extension 環境変数

次の環境変数で拡張機能を設定できます。

環境変数を使用する方法については、「Lambda 開発者ガイド」の「[Using Lambda environment variables](#)」(Lambda 環境変数の使用) を参照してください。

PARAMETERS_SECRETS_EXTENSION_CACHE_ENABLED

パラメータとシークレットをキャッシュするには true に設定します。キャッシュしない場合は false に設定します。デフォルトは true です。

PARAMETERS_SECRETS_EXTENSION_CACHE_SIZE

キャッシュするシークレットとパラメータの最大数。0 ~ 1000 の値にする必要があります。値を 0 にすると、キャッシングは行われません。SSM_PARAMETER_STORE_TTL と SECRETS_MANAGER_TTL の両方が 0 の場合、この変数は無視されます。デフォルトは 1000 です。

PARAMETERS_SECRETS_EXTENSION_HTTP_PORT

ローカル HTTP サーバーのポート。デフォルトは、2773 です。

PARAMETERS_SECRETS_EXTENSION_LOG_LEVEL

拡張機能が提供するログ記録のレベル: debug、info、warn、error、または none。debug に設定すると、キャッシュ設定が表示されます。デフォルトは info です。

PARAMETERS_SECRETS_EXTENSION_MAX_CONNECTIONS

拡張機能が Parameter Store または Secrets Manager へのリクエストに使用する HTTP クライアントの最大接続数。これはクライアントごとの設定です。デフォルトは 3 です。

SECRETS_MANAGER_TIMEOUT_MILLIS

Secrets Manager へのリクエストのタイムアウト (ミリ秒単位)。値を 0 にすると、タイムアウトは発生しません。デフォルトは 0 です。

SECRETS_MANAGER_TTL

キャッシュ内のシークレットの TTL (秒単位)。値を 0 にすると、キャッシングは行われません。最大値は 300 秒です。PARAMETERS_SECRETS_CACHE_SIZE が 0 の場合、この変数は無視されます。デフォルトは 300 秒です。

SSM_PARAMETER_STORE_TIMEOUT_MILLIS

Parameter Store へのリクエストのタイムアウト (ミリ秒単位)。値を 0 にすると、タイムアウトは発生しません。デフォルトは 0 です。

SSM_PARAMETER_STORE_TTL

キャッシュ内のパラメータの TTL (秒単位)。値を 0 にすると、キャッシングは行われません。最大値は 300 秒です。PARAMETERS_SECRETS_CACHE_SIZE が 0 の場合、この変数は無視されます。デフォルトは 300 秒です。

パラメータストア内で AWS Secrets Manager シークレットを使用する

AWS Systems Manager パラメータストアは、設定データ管理とシークレット管理のための安全な階層型ストレージを提供します。パスワード、データベース文字列、およびライセンスコードなどのデータをパラメータ値として保存することができます。ただし、パラメータストアは保存されるシークレットの自動ローテーションサービスを提供していません。代わりに、パラメータストアを使用して、Secrets Manager にシークレットを保存することができ、パラメータストアのパラメータとしてそのシークレットを参照できます。

Secrets Manager を使用してパラメータストアを設定する場合、secret-id パラメータストアでは、名前文字列の前にスラッシュ (/) が必要です。

詳細については、「AWS Systems Manager ユーザーガイド」の、「[パラメータストアのパラメータから AWS Secrets Manager シークレットを参照する](#)」を参照してください。

AWS Secrets Manager シークレットのローテーション

ローテーションとは、シークレットを定期的に更新するためのプロセスのことです。シークレットのローテーションを行うと、シークレット、ならびに、データベースまたはサービスの認証情報が更新されます。Secrets Manager では、シークレットの自動ローテーションを設定できます。ローテーションには 2 つの形式があります。

- [マネージドローテーション](#) – ほとんどの[マネージドシークレット](#)では、マネージドローテーションを使用します。このローテーションでは、サービスがローテーションを設定および管理します。マネージドローテーションは Lambda 関数を使用しません。
- [the section called “Lambda 関数によるローテーション”](#) – 他のタイプのシークレットの場合、Secrets Manager のローテーションは Lambda 関数を使用してシークレットとデータベースまたはサービスを更新します。

AWS Secrets Manager シークレットのマネージドローテーション

一部のサービスは、ユーザーに代わってローテーションの設定と管理を行うマネージドローテーションを提供しています。マネージドローテーションでは、AWS Lambda 関数を使用してデータベース内のシークレットと認証情報を更新しません。

次のサービスは、マネージドローテーションを提供しています。

- Amazon Aurora は、マスターユーザー認証情報のマネージドローテーションを提供します。詳細については、「Amazon Aurora ユーザーガイド」の「[Amazon Aurora および AWS Secrets Manager でのパスワード管理](#)」を参照してください。
- Amazon ECS Service Connect は、AWS Private Certificate Authority TLS 証明書のマネージドローテーションを提供します。詳細については、「Amazon Elastic Container Service デベロッパーガイド」の「[TLS with Service Connect](#)」を参照してください。
- Amazon RDS は、マスターユーザー認証情報のマネージドローテーションを提供します。詳細については、「Amazon RDS ユーザーガイド」の「[Amazon RDS および AWS Secrets Manager でのパスワード管理](#)」を参照してください。
- Amazon Redshift は、管理者パスワードのマネージドローテーションを提供します。詳細については、「Amazon Redshift 管理ガイド」の「[AWS Secrets Managerを使用して Amazon Redshift 管理者パスワードを管理する](#)」を参照してください。

i Tip

他のすべてのタイプのシークレットについては、「[the section called “Lambda 関数によるローテーション”](#)」を参照してください。

マネージドシークレットのローテーションは、通常 1 分以内に完了します。ローテーション中、シークレットを取得する新しい接続では、以前のバージョンの認証情報が取得される場合があります。アプリケーションでは、マスターユーザーを使用する代わりに、アプリケーションに必要な最小の特権で作成されたデータベースユーザーを使用するというベストプラクティスに従うことを強くお勧めします。アプリケーションユーザーの場合、可用性を最大限に高めるには、[交代ユーザーローテーション戦略](#)を使用できます。

マネージドローテーションのスケジュールを変更するには

1. Secrets Manager コンソールでマネージドシークレットを開きます。管理サービスからのリンクをたどるか、Secrets Manager コンソールで[シークレットを検索](#)できます。
2. [Rotation schedule] (ローテーションスケジュール) において、UTC タイムゾーンで [Schedule expression builder] (スケジュール式ビルダー) にスケジュールを入力するか、[Schedule expression] (スケジュール式) としてスケジュールを入力します。Secrets Manager は、スケジュールを `rate()` 式または `cron()` 式として保存します。[Start time] (開始時刻) を指定しない限り、ローテーションウィンドウは午前 0 時に自動的に開始されます。シークレットが 4 時間ごとにローテーションされるように設定できます。詳細については、「[ローテーションスケジュール](#)」を参照してください。
3. (オプション) [Window duration] (ウィンドウ期間) では、Secrets Manager がシークレットをローテーションするウィンドウの長さを選択します (3 時間のウィンドウの場合は **3h** など)。ウィンドウが次のローテーションウィンドウに重ならないようにしてください。時間単位のローテーションスケジュールでは、ウィンドウ期間を指定しない場合、ウィンドウは 1 時間後に自動的に終了します。日数単位のローテーションスケジュールの場合、ウィンドウは 1 日の終わりに自動的に終了します。
4. [保存] を選択します。

マネージドローテーションのスケジュールを変更するには (AWS CLI)

- [rotate-secret](#) を呼び出します。次の例では、月の 1 日と 15 日の 16 時から 18 時 (UTC) の間にシークレットがローテーションされます。詳細については、「[ローテーションスケジュール](#)」を参照してください。

```
aws secretsmanager rotate-secret \  
  --secret-id MySecret \  
  --rotation-rules "{\"ScheduleExpression\": \"cron(0 16 1,15 * ? *)\" \  
  \",  
  \"Duration\": \"2h\"}"
```

Lambda 関数によるローテーション

多くのタイプのシークレットについて、Secrets Manager は AWS Lambda 関数を使用してシークレットとデータベースまたはサービスを更新します。Lambda 関数を使用する場合のコストについては、「[料金](#)」を参照してください。

[マネージドシークレット](#) の場合、マネージドローテーションを使用します。[マネージドローテーション](#) を使用するには、最初に管理サービスを通じてシークレットを作成します。

ローテーション中、Secrets Manager はローテーションの状態を示すイベントをログに記録します。詳細については、「[the section called “でログ記録する AWS CloudTrail”](#)」を参照してください。

シークレットをローテーションするために、Secrets Manager は設定したローテーションスケジュールに従って [Lambda 関数](#) を呼び出します。自動ローテーションの設定中にシークレット値も手動で更新した場合、Secrets Manager は次のローテーション日を計算するときにそれを有効なローテーションと見なします。

ローテーション中、Secrets Manager は毎回異なるパラメータを使用して、同じ関数を複数回呼び出します。Secrets Manager は、次の JSON リクエスト構造のパラメータを使用して 関数を呼び出します。

```
{  
  "Step" : "request.type",  
  "SecretId" : "string",  
  "ClientRequestToken" : "string"  
}
```

ローテーションステップが失敗すると、Secrets Manager はローテーションプロセス全体を複数回再試行します。

トピック

- [Amazon RDS、Amazon Aurora、Amazon Redshift、または Amazon DocumentDB シークレットの自動ローテーションを設定する](#)

- [データベース以外の AWS Secrets Manager シークレットの自動ローテーションを設定する](#)
- [Lambda 関数のローテーション戦略](#)
- [Lambda ローテーション関数](#)
- [AWS Secrets Manager ローテーション関数テンプレート](#)
- [の Lambda ローテーション関数の実行ロールのアクセス許可 AWS Secrets Manager](#)
- [Lambda ローテーション関数のネットワークアクセス](#)
- [AWS Secrets Manager ローテーションのトラブルシューティング](#)

Amazon RDS、Amazon Aurora、Amazon Redshift、または Amazon DocumentDB シークレットの自動ローテーションを設定する

このチュートリアルでは、データベースシークレット [the section called “Lambda 関数によるローテーション”](#) のセットアップ方法について説明します。ローテーションとは、シークレットを定期的に更新するためのプロセスのことです。シークレットのローテーションを行うと、シークレットとデータベースの両方で認証情報が更新されます。Secrets Manager では、データベースシークレットの自動ローテーションを設定できます。

コンソールを使用してローテーションを設定するには、まずローテーション戦略を選択する必要があります。次に、ローテーションのシークレットを設定します。これにより、もしまだ未作成の場合は Lambda ローテーション関数が作成されます。コンソールは Lambda 関数実行ロールのアクセス権限も設定します。最後のステップとして、Lambda ローテーション関数が Secrets Manager とデータベースの両方にネットワーク経由でアクセスできることを確認する必要があります。

Warning

自動ローテーションを有効にするには、Lambda ローテーション関数の IAM 実行ロールを作成し、アクセス許可ポリシーをアタッチするアクセス許可が必要です。iam:CreateRole 許可と iam:AttachRolePolicy 許可の両方が必要です。これらのアクセス許可を付与すると、アイデンティティは自分自身に任意のアクセス許可を付与できます。

ステップ:

- [ステップ 1: ローテーション戦略を選択し、オプションでスーパーユーザーシークレットを作成する](#)
- [ステップ 2: ローテーションの設定とローテーション関数の作成](#)

- [ステップ 3: \(オプション\) ローテーション関数に追加のアクセス許可条件を設定する](#)
- [ステップ 4: ローテーション関数用のネットワークアクセスを設定する](#)
- [次のステップ](#)

ステップ 1: ローテーション戦略を選択し、オプションでスーパーユーザーシークレットを作成する

Secrets Manager が提供する戦略については、「」を参照してください[the section called “Lambda 関数のローテーション戦略”](#)。

交代ユーザー戦略を選択する場合は、[データベースシークレットを作成する](#) して、データベースのスーパーユーザーの認証情報をそこに保存する必要があります。ローテーションでは最初のユーザーのクローンが作成されますが、ほとんどのユーザーにはその権限がないため、スーパーユーザーの認証情報を含むシークレットが必要です。Amazon RDS Proxy は交代ユーザー戦略をサポートしていないことに注意してください。

ステップ 2: ローテーションの設定とローテーション関数の作成

Amazon RDS、Amazon DocumentDB、または Amazon Redshift のシークレットでローテーションを有効にするには

1. Secrets Manager コンソール (<https://console.aws.amazon.com/secretsmanager/>) を開きます。
2. [Secrets] (シークレット) ページで、自分のシークレットを選択します。
3. シークレットの詳細ページで、[Rotation configuration] (ローテーション設定) セクションの [Edit rotation] (ローテーションの編集) を選択します。
4. [Edit rotation configuration] (ローテーション設定の編集) ダイアログボックスで、次の操作を行います。
 - a. [Automatic rotation] (自動ローテーション) を有効化します。
 - b. [Rotation schedule] (ローテーションスケジュール) において、UTC タイムゾーンで [Schedule expression builder] (スケジュール式ビルダー) にスケジュールを入力するか、[Schedule expression] (スケジュール式) としてスケジュールを入力します。Secrets Manager は、スケジュールを `rate()` 式または `cron()` 式として保存します。[Start time] (開始時刻) を指定しない限り、ローテーションウィンドウは午前 0 時に自動的に開始されます。シークレットが 4 時間ごとにローテーションされるように設定できます。詳細については、「[ローテーションスケジュール](#)」を参照してください。

- c. (オプション) [Window duration] (ウィンドウ期間) では、Secrets Manager がシークレットをローテーションするウィンドウの長さを選択します (3 時間のウィンドウの場合は **3h** など)。ウィンドウが次のローテーションウィンドウに重ならないようにしてください。時間単位のローテーションスケジュールでは、ウィンドウ期間を指定しない場合、ウィンドウは 1 時間後に自動的に終了します。日数単位のローテーションスケジュールの場合、ウィンドウは 1 日の終わりに自動的に終了します。
- d. (オプション) 変更を保存したときにシークレットをローテーションするには、[Rotate immediately when the secret is stored] (シークレットが保存されたときにすぐにローテーションする) を選択します。チェックボックスをオフにすると、最初のローテーションは設定したスケジュールから開始されます。

手順 3 と 4 がまだ完了していないなどの理由でローテーションが失敗した場合、Secrets Manager はローテーションプロセスを複数回再試行します。

- e. [Rotation function] (ローテーション関数) で、次のいずれかを行います。
 - [Create a new Lambda function] (新しい Lambda 関数の作成) を選択し、新しい関数の名前を 1 つ入力します。Secrets Manager は、関数名の先頭に「SecretsManager」を追加します。Secrets Manager は適切な[テンプレート](#)に基づいて関数を作成し、Lambda 実行ロールに必要な[アクセス権限](#)を設定します。
 - [Use an existing Lambda function] (既存の Lambda 関数を使用する) を選択して、別のシークレットに使用したローテーション関数を再利用します。推奨 VPC 設定に記載されているローテーション関数は、データベースと同じ VPC とセキュリティグループを持つため、関数がデータベースにアクセスしやすくなっています。
- f. [ローテーション戦略] では、[単一ユーザー] または [代替ユーザー] 戦略を選択します。詳細については、「[the section called “ステップ 1: ローテーション戦略を選択し、オプションでスーパーユーザーシークレットを作成する”](#)」を参照してください

5. 保存を選択します。

ステップ 3: (オプション) ローテーション関数に追加のアクセス許可条件を設定する

ローテーション関数のリソースポリシーには、Lambda が[混乱した代理プログラム](#)として使用されることを防ぐために、コンテキストキー「[aws:SourceAccount](#)」を含めることを推奨します。一部のサービスでは AWS、混乱した代理シナリオを避けるために、AWS は [aws:SourceArn](#) と [aws:SourceAccount](#) グローバル条件キーの両方を使用することをお勧めします。ただし、ローテーション関数のポリシーに [aws:SourceArn](#) の条件を含めると、その ARN で指定されたシークレットだけをローテーションさせるためにローテーション関数を使用することができます。コンテキ

ストキーのみを含めることをお勧めしますaws:SourceAccount複数のシークレットに対して回転関数を使用できるようにする。

ローテーション関数のリソースポリシーを更新するには

1. Secrets Manager のコンソールでシークレットを選択し、詳細ページの [Rotation configuration] (ローテーション設定) で、Lambda ローテーション関数を選択します。Lambda コンソールが開きます。
2. 「[Lambda のリソースベースのポリシーを使用する](#)」の指示に従って aws:sourceAccount の条件を追加します。

```
"Condition": {
  "StringEquals": {
    "AWS:SourceAccount": "123456789012"
  }
},
```

AWS マネージドキー aws/secretsmanager 以外の KMS キーを使用しシークレットを暗号化する場合、Secrets Manager は Lambda の実行ロールに対し、そのキーの使用に関するアクセス許可を付与します。[SecretArn 暗号化コンテキスト](#)を使用して復号化関数の使用を制限できます。この場合、ローテーション関数ロールには、ローテーションに使用するシークレットを復号化するアクセスのみが許可されます。

ローテーション関数の実行ロールを更新するには

1. Lambda ローテーション関数で [設定] を選択し、次に [実行ロール] で [ロール名] を選択します。
2. 「[ロールのアクセス許可ポリシーの変更](#)」の指示に従って、kms:EncryptionContext:SecretARN 条件を追加します。

```
"Condition": {
  "StringEquals": {
    "kms:EncryptionContext:SecretARN": "SecretARN"
  }
},
```

ステップ 4: ローターション関数用のネットワークアクセスを設定する

詳細については、「[the section called “Lambda ローターション関数のネットワークアクセス”](#)」を参照してください。

次のステップ

「[the section called “におけるローテーションのトラブルシューティング”](#)」を参照してください。

データベース以外の AWS Secrets Manager シークレットの自動ローテーションを設定する

このチュートリアルでは、データベース以外のシークレット [the section called “Lambda 関数によるローテーション”](#) をセットアップする方法について説明します。ローテーションとは、シークレットを定期的に更新するためのプロセスのことです。シークレットのローテーションを行うと、シークレット、ならびに、そのシークレットのデータベースまたはサービスの認証情報が更新されます。

データベースシークレットについては、「[データベースシークレットの自動ローテーション](#)」を参照してください。

Warning

自動ローテーションを有効にするには、Lambda ローターション関数の IAM 実行ロールを作成し、アクセス許可ポリシーをアタッチするアクセス許可が必要です。iam:CreateRole 許可と iam:AttachRolePolicy 許可の両方が必要です。これらのアクセス許可を付与すると、アイデンティティは自分自身に任意のアクセス許可を付与できます。

ステップ:

- [ステップ 1: 汎用ローテーション関数を作成する](#)
- [ステップ 2: ローターション関数コードを記述する](#)
- [ステップ 3: ローターション用のシークレットを設定する](#)
- [ステップ 4: ローターション関数が Secrets Manager とデータベースまたはサービスにアクセスすることを許可する](#)
- [ステップ 5: Secrets Manager にローテーション関数の呼び出しを許可する](#)
- [ステップ 6: ローターション関数のネットワークアクセスを設定する](#)
- [次のステップ](#)

ステップ 1: 汎用ローテーション関数を作成する

まず、Lambda ローテーション関数を作成します。シークレットをローテーションするためのコードは含まれないため、後のステップで記述します。ローテーション関数の仕組みについては、「」を参照してください[the section called “Lambda ローテーション関数”](#)。

サポートされているリージョンでは、AWS Serverless Application Repository を使用してテンプレートから関数を作成できます。サポートされているリージョンのリストについては、[AWS Serverless Application Repository FAQs](#)」を参照してください。他のリージョンでは、関数を最初から作成し、テンプレートコードを関数にコピーします。

汎用ローテーション関数を作成するには

1. お客様のリージョンで AWS Serverless Application Repository サポートされているかどうかを確認するには、AWS 全般のリファレンスの[AWS Serverless Application Repository 「エンドポイントとクォータ」](#)を参照してください。
2. 次のいずれかを行います。
 - AWS Serverless Application Repository リージョンで がサポートされている場合：
 - a. Lambda コンソールで、アプリケーション を選択し、アプリケーションの作成 を選択します。
 - b. 「アプリケーションの作成」ページで、「サーバーレスアプリケーション」タブを選択します。
 - c. 「パブリックアプリケーション」の検索ボックスに「」と入力します **SecretsManagerRotationTemplate**。
 - d. カスタム IAM ロールまたはリソースポリシーを作成するアプリを表示する を選択します。
 - e. SecretsManagerRotationTemplate タイルを選択します。
 - f. 「確認、設定、デプロイ」ページの「アプリケーション設定」タイルで、必須フィールドに入力します。
 - エンドポイント には、 を含むリージョンのエンドポイントを入力します **https://**。 ; エンドポイントのリストについては、「[the section called “Secrets Manager エンドポイント”](#)」を参照してください。
 - Lambda 関数を VPC に配置するには、vpcSecurityGroupId と を含め **vpcSubnetIds**。
 - g. [デプロイ] を選択します。

- リージョンで がサポートされ AWS Serverless Application Repository ていない場合：
 - a. Lambda コンソールで、関数 を選択し、関数の作成 を選択します。
 - b. [関数の作成] ページで、次の操作を実行します。
 - i. Author from scratch (製作者を最初から) を選択します。
 - ii. [Function name] (関数名) には、関数の名前を入力します。
 - iii. [Runtime] (ランタイム) では、[Python 3.9] を選択します。
 - iv. [関数を作成] を選択します。

ステップ 2: ローテーション関数コードを記述する

このステップでは、シークレットを更新するコードと、シークレットの対象となるサービスまたはデータベースを記述します。独自のローテーション関数の記述に関するヒントなど、ローテーション関数の動作については、「」を参照してください [the section called “Lambda ローテーション関数”](#)。をリファレンス [ローテーション関数のテンプレート](#) として使用することもできます。

ステップ 3: ローテーション用のシークレットを設定する

このステップでは、シークレットのローテーションスケジュールを設定し、ローテーション関数をシークレットに接続します。

ローテーションを設定して空のローテーション関数を作成するには

1. Secrets Manager コンソール (<https://console.aws.amazon.com/secretsmanager/>) を開きます。
2. [Secrets] (シークレット) ページで、自分のシークレットを選択します。
3. シークレットの詳細ページで、[Rotation configuration] (ローテーション設定) セクションの [Edit rotation] (ローテーションの編集) を選択します。[Edit rotation configuration] (ローテーション設定の編集) ダイアログボックスで、次の操作を行います。
 - a. [Automatic rotation] (自動ローテーション) を有効化します。
 - b. [Rotation schedule] (ローテーションスケジュール) において、UTC タイムゾーンで [Schedule expression builder] (スケジュール式ビルダー) にスケジュールを入力するか、[Schedule expression] (スケジュール式) としてスケジュールを入力します。Secrets Manager は、スケジュールを `rate()` 式または `cron()` 式として保存します。[Start time] (開始時刻) を指定しない限り、ローテーションウィンドウは午前 0 時に自動的に開始され

ます。シークレットが 4 時間ごとにローテーションされるように設定できます。詳細については、「[ローテーションスケジュール](#)」を参照してください。

- c. (オプション) [Window duration] (ウィンドウ期間) では、Secrets Manager がシークレットをローテーションするウィンドウの長さを選択します (3 時間のウィンドウの場合は **3h** など)。ウィンドウが次のローテーションウィンドウに重ならないようにしてください。時間単位のローテーションスケジュールでは、ウィンドウ期間を指定しない場合、ウィンドウは 1 時間後に自動的に終了します。日数単位のローテーションスケジュールの場合、ウィンドウは 1 日の終わりに自動的に終了します。
- d. (オプション) 変更を保存したときにシークレットをローテーションするには、[Rotate immediately when the secret is stored] (シークレットが保存されたときにすぐにローテーションする) を選択します。チェックボックスをオフにすると、最初のローテーションは設定したスケジュールから開始されます。
- e. ローテーション関数で、ステップ 1 で作成した Lambda 関数を選択します。
- f. [保存] を選択します。

ステップ 4: ローテーション関数が Secrets Manager とデータベースまたはサービスにアクセスすることを許可する

Lambda ローテーション関数には、Secrets Manager のシークレットにアクセスする権限と、データベースまたはサービスにアクセスする権限が必要です。このステップでは、これらのアクセス権限を Lambda 実行ロールに付与します。AWS マネージドキー `aws/secretsmanager` 以外の KMS キーを使用しシークレットを暗号化する場合は、そのキーの使用に関するアクセス許可を、Lambda の実行ロールに付与する必要があります。[SecretArn 暗号化コンテキスト](#)を使用して復号化関数の使用を制限できます。この場合、ローテーション関数ロールには、ローテーションに使用するシークレットを復号化するアクセスのみが許可されます。IAM ポリシーの例については、「[ローテーションへのアクセス許可](#)」を参照してください。

手順については、AWS Lambda 開発者ガイドの「[Lambda 実行ロール](#)」を参照してください。

ステップ 5: Secrets Manager にローテーション関数の呼び出しを許可する

設定したローテーションスケジュールで Secrets Manager がローテーション関数を呼び出せるようにするには、Lambda 関数のリソースポリシーで Secrets Manager サービスプリンシパルに `aws:InvokeFunction` 許可を付与する必要があります。

ローテーション関数のリソースポリシーには、Lambda が [混乱した代理プログラム](#)として使用されることを防ぐために、コンテキストキー「[aws:SourceAccount](#)」を含めることを推奨します。

一部のサービスでは AWS、混乱した代理シナリオを避けるために、AWS は [aws:SourceArn](#) と [aws:SourceAccount](#) グローバル条件キーの両方を使用することをお勧めします。ただし、ローテーション関数のポリシーに `aws:SourceArn` の条件を含めると、その ARN で指定されたシークレットだけをローテーションさせるためにローテーション関数を使用することができます。コンテキストキーのみを含めることをお勧めします `aws:SourceAccount` 複数のシークレットに対して回転関数を使用できるようにする。

Lambda 関数にリソースポリシーをアタッチするには、「[リソースベースのポリシーを使用する](#)」を参照してください。

次のポリシーでは、Secrets Manager が Lambda 関数を呼び出すことを許可します。

```
{
  "Version": "2012-10-17",
  "Id": "default",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "secretsmanager.amazonaws.com"
      },
      "Action": "lambda:InvokeFunction",
      "Condition": {
        "StringEquals": {
          "AWS:SourceAccount": "123456789012"
        }
      },
      "Resource": "LambdaRotationFunctionARN"
    }
  ]
}
```

ステップ 6: ローテーション関数のネットワークアクセスを設定する

このステップでは、ローテーション関数が Secrets Manager と、シークレットの対象となるサービスまたはデータベースの両方に c 接続できるようにします。シークレットをローテーションできるようにするには、ローテーション関数が両方にアクセスする必要があります。[the section called “Lambda ローテーション関数のネットワークアクセス”](#) を参照してください。

次のステップ

ステップ 3 でローテーションを設定したら、シークレットをローテーションするスケジュールを設定します。スケジュール時にローテーションが失敗した場合、Secrets Manager はローテーションを複数回試行します。「」の手順に従って、ローテーションをすぐに開始することもできます[すぐにシークレットをローテーションする](#)。

ローテーションが失敗した場合は、「」を参照してください [におけるローテーションのトラブルシューティング](#)。

Lambda 関数のローテーション戦略

の場合 [the section called “Lambda 関数によるローテーション”](#)、データベースシークレットの場合、Secrets Manager には 2 つのローテーション戦略があります。

ローテーション戦略: シングルユーザー

この戦略は、1 つのシークレット内で 1 人のユーザーの認証情報を更新します。Amazon RDS Db2 インスタンスの場合、ユーザーは自分のパスワードを変更できないため、管理者の認証情報を別のシークレットで提供する必要があります。これは最も簡単なローテーション戦略であり、ほとんどのユースケースに使用することができます。特に、1 回限りの (アドホック) ユーザーまたはインタラクティブユーザーの認証情報には、この方法を使用することを推奨します。

シークレットがローテーションしても、開いているデータベース接続は切断されません。ローテーションの実行中、データベース内のパスワードが変更されてからシークレットが更新されるまでには少し時間がかかります。その間に、ローテーションされた認証情報を使用する呼び出しがデータベースによって拒否されるリスクは低く抑えられています。このリスクは、[適切な再試行戦略](#)を適用することで低減することが可能です。ローテーション後、新しい接続は新しい認証情報を使用します。

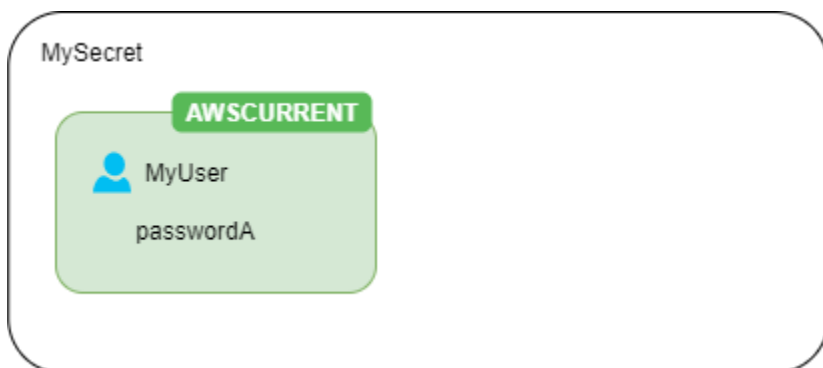
ローテーション戦略: 交代ユーザー

この戦略は、1 つのシークレット内で 2 人のユーザーの認証情報を更新します。最初のユーザーを作成し、その最初のローテーション中に、ローテーション関数がユーザーのクローンを作成して 2 人目のユーザーを作成します。シークレットがローテーションされるたびに、ローテーション関数はパスワードを更新するユーザーを切り替えます。ほとんどのユーザーにはクローン作成権限がないため、superuser の認証情報を別のシークレット内で用意する必要があります。データベース内のクローンユーザーがオリジナルユーザーと同じ権限を持っていない場合や、1 回限り (アドホック) またはインタラクティブなユーザーの認証情報には、シングルユーザーローテーション方法を使用することを推奨します。

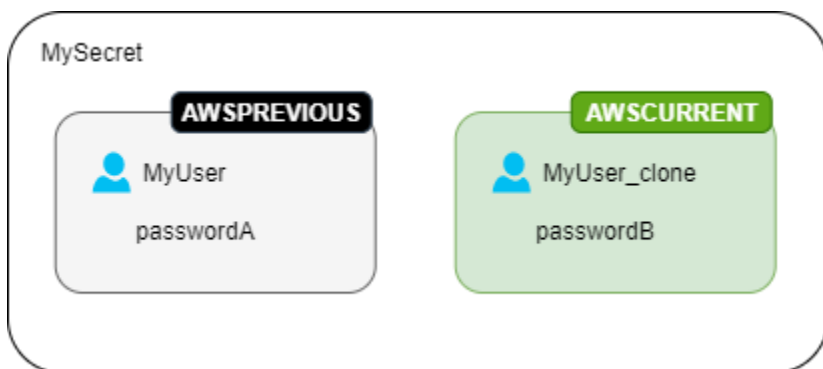
この戦略は、1つ目のロールがデータベーステーブルを所有し、2つ目のロールにそのデータベーステーブルへのアクセス許可を付与するといった権限モデルのデータベースに適しています。また、高可用性を必要とするアプリケーションにも適しています。アプリケーションは、ローテーション中にシークレットを取得しても、引き続き有効な認証情報セットを取得します。ローテーション後、`user` と `user_clone` の両方の認証情報が有効になります。このタイプのローテーション中にアプリケーションが拒否される可能性は、シングルユーザーのローテーションを比較すると一段と低くなります。データベースをホストしているサーバーファームでパスワードの変更がサーバー全体に伝播するまでに時間がかかる場合には、新しい認証情報を使用する呼び出しがデータベースによって拒否されるおそれがあります。このリスクは、[適切な再試行戦略](#)を適用することで低減することが可能です。

Secrets Manager は、元のユーザーと同じアクセス許可を持つクローンユーザーを作成します。クローンユーザーが作成された後に元のユーザーの権限を変更する場合は、クローンユーザー側の権限も変更する必要があります。

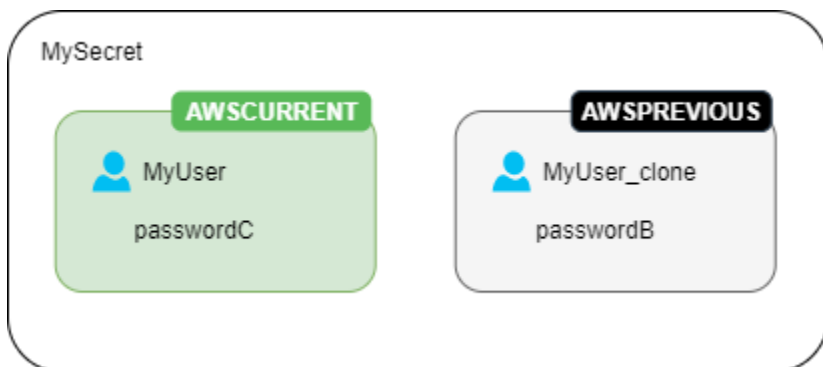
例えば、データベースユーザーの認証情報を使用してシークレットを作成した場合、そのシークレットには、使用した認証情報によるバージョンが1つ含まれます。



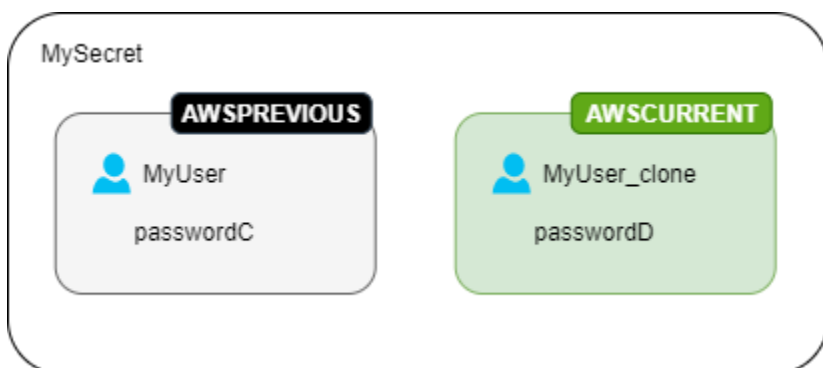
1回目のローテーション – ローテーション関数がパスワードを生成し、それを使用してユーザーのクローンを作成します。それらの認証情報は、現在のシークレットバージョンになります。



2 回目のローテーション – ローテーション関数は、元のユーザーのパスワードを更新します。



3 回目のローテーション – ローテーション関数は、クローンされたユーザーのパスワードを更新します。



Lambda ローテーション関数

では[the section called “Lambda 関数によるローテーション”](#)、Lambda 関数がシークレットをローテーションする作業を行います。ローテーション中、Secrets Manager は[ステージングラベル](#)を使用して、シークレットのバージョンにラベル付けを行います。

Secrets Manager がシークレットのタイプに[ローテーション関数テンプレート](#)を提供していない場合は、ローテーション関数を作成できます。ローテーション関数を作成するときは、各ステップのガイドに従ってください。

独自のローテーション関数を記述するためのヒント

- [汎用ローテーションテンプレート](#)を開始点として使用して、独自のローテーション関数を記述します。
- 関数を記述する場合、デバッグステートメントまたはロギングステートメントの記述に細心の注意を払ってください。これらのステートメントにより、関数内の情報が Amazon に書き込まれる可

能性があるため CloudWatch、開発中に収集された機密情報がログに含まれていないことを確認する必要があります。

ログステートメントの例については、[the section called “ローテーション関数のテンプレート”](#) のソースコードを参照してください。

- セキュリティ上の理由から、Secrets Manager では、Lambda ローテーション関数が直接シークレットをローテーションすることのみを許可しています。ローテーション関数では、2 つ目の Lambda 関数を呼び出してシークレットをローテーションすることはできません。
- デバッグの提案については、「[サーバーレスアプリケーションのテストとデバッグ](#)」を参照してください。
- 例えば、外部バイナリとライブラリを使用してリソースに接続する場合は、パッチ適用を管理し、を維持する必要があります up-to-date。
- ローテーション関数は、必要な依存関係と共に ZIP ファイル [*my-function.zip*] に保存します。

ローテーション関数の 4 つのステップ

トピック

- [create_secret: シークレットの新しいバージョンを作成する](#)
- [set_secret: データベースまたはサービスの認証情報を変更する](#)
- [test_secret: 新しいシークレットバージョンをテストする](#)
- [finish_secret: ローテーションを完了する](#)

create_secret: シークレットの新しいバージョンを作成する

メソッドは create_secret まず、渡された [get_secret_value](#) で を呼び出してシークレットが存在するかどうかを確認します ClientRequestToken。シークレットがない場合は、[create_secret](#) とトークン をとって新しいシークレットを作成します VersionId。次に、 を使用して新しいシークレット値を生成します [get_random_password](#)。次に、 [put_secret_value](#) を呼び出してステージングラベル とともに保存します AWSPENDING。新しいシークレット値を AWSPENDING に格納することで、冪等性を確保することができます。何らかの理由でローテーションが失敗した場合は、その後の呼び出しでそのシークレット値を参照できます。詳細については、「[Lambda 関数を冪等にするにはどうすればよいですか?](#)」を参照してください。

独自のローテーション関数を記述するためのヒント

- 新しいシークレット値に、データベースまたはサービスに有効な文字のみが含まれていることを確認します。ExcludeCharacters のパラメータを使用して文字を除外します。
- 関数をテストするときは、 を使用してバージョンステージを確認します。AWS CLI を呼び出し [describe-secret](#) を確認しますVersionIdsToStages。
- Amazon RDS MySQL の場合、交代ユーザーのローテーションでは、Secrets Manager は 16 文字以下の名前で作成されたユーザーを作成します。ローテーション関数を変更して、長いユーザー名を許可することができます。MySQL バージョン 5.7 以降では最大 32 文字のユーザー名がサポートされていますが、Secrets Manager ではユーザー名の末尾に「_clone」(6 文字) が追加されるため、ユーザー名は最大 26 文字にする必要があります。

set_secret: データベースまたはサービスの認証情報を変更する

メソッドは、シークレットAWSPENDINGのバージョンの新しいシークレット値と一致するように、データベースまたはサービスの認証情報set_secretを変更します。

独自のローテーション関数を記述するためのヒント

- データベースなどのステートメントを解釈するサービスにステートメントを渡す場合は、クエリパラメータ化を使用します。詳細については、OWASP ウェブサイト [の「クエリパラメータ化シート」](#)を参照してください。
- ローテーション機能は、Secrets Manager のシークレットとターゲットリソースの両方にある顧客認証情報にアクセスして変更する権限を持つ特権的な代理プログラムです。[混乱した代理攻撃](#)を防ぐには、攻撃者がこの関数を使用して他のリソースにアクセスできないようにする必要があります。認証情報を更新する前に:
 - シークレットの AWSCURRENT バージョンの認証情報が有効であることを確認してください。AWSCURRENT の認証情報が有効でない場合は、ローテーションの試行を中止してください。
 - AWSCURRENT と AWSPENDING のシークレット値が同じリソース用であることを確認してください。ユーザー名とパスワードについては、AWSCURRENT と AWSPENDING のユーザー名が同じであることを確認してください。
 - 送信先のサービスリソースが同じであることを確認してください。データベースの場合、AWSCURRENT と AWSPENDING のホスト名が同じであることを確認してください。
- まれに、データベースの既存のローテーション関数をカスタマイズすることもできます。例えば、交代ユーザーローテーションの場合、Secrets Manager は最初のユーザーの[ランタイム設定パラ](#)

[メータ](#)をコピーしてクローンユーザーを作成します。さらに属性を追加したり、クローンユーザーに付与する属性を変更したりする場合は、`set_secret` 関数のコードを更新する必要があります。

test_secret: 新しいシークレットバージョンをテストする

次に、Lambda ローテーション関数は、データベースまたはサービスにアクセスすることで、シークレットの `AWSPENDING` バージョンをテストします。[ローテーション関数のテンプレート](#) に基づくローテーション関数では、読み取りアクセスを使用して、新しいシークレットをテストします。

finish_secret: ローテーションを完了する

最後に、Lambda ローテーション関数はラベル `AWSCURRENT` を以前のシークレットバージョンからこのバージョンに移動します。これにより、同じ API コール内の `AWSPENDING` ラベルも削除されます。Secrets Manager は、以前のバージョンに対しステージングラベル `AWSPREVIOUS` を付加します。これにより、シークレットの最後の有効なバージョンが保持されます。

メソッドは `finish_secret` を使用して [update_secret_version_stage](#)、ステージングラベルを以前のシークレットバージョン `AWSCURRENT` から新しいシークレットバージョンに移動します。Secrets Manager は、以前のバージョンに対しステージングラベル `AWSPREVIOUS` を自動的に付加します。これにより、シークレットの最後の有効なバージョンが保持されます。

独自のローテーション関数を記述するためのヒント

- この時点より `AWSPENDING` 前に を削除しないでください。また、別の API コールを使用して削除しないでください。これは、ローテーションが正常に完了しなかったことを Secrets Manager に示す可能性があるためです。Secrets Manager は、以前のバージョンに対しステージングラベル `AWSPREVIOUS` を付加します。これにより、シークレットの最後の有効なバージョンが保持されます。

ローテーションが成功すると、`AWSPENDING` ステージングラベルは `AWSCURRENT` バージョンと同じバージョンにアタッチされるか、どのバージョンにもアタッチされない可能性があります。`AWSPENDING` ステージングラベルは存在するが、`AWSCURRENT` と同じバージョンにアタッチされていない場合、それ以降に呼び出されたローテーションでは、以前のローテーションリクエストがまだ進行中であるとみなされ、エラーが返されます。ローテーションに失敗すると、`AWSPENDING` ステージングラベルはバージョンが空のシークレットにアタッチされる可能性があります。詳細については、「[におけるローテーションのトラブルシューティング](#)」を参照してください。

AWS Secrets Manager ローテーション関数テンプレート

の場合 [the section called “Lambda 関数によるローテーション”](#)、Secrets Manager は多数のローテーション関数テンプレートを提供します。テンプレートの使用方法については、以下を参照してください。

- [データベースシークレットの自動ローテーション](#)
- [データベース以外のシークレットの自動ローテーション](#)

テンプレートは Python 3.9 をサポートしています。

独自のローテーション関数を記述するには、「[ローテーション関数の記述](#)」を参照してください。

テンプレート

- [Amazon RDS と Amazon Aurora](#)
 - [Amazon RDS Db2 シングルユーザー](#)
 - [Amazon RDS Db2 交代ユーザー](#)
 - [Amazon RDS MariaDB シングルユーザー](#)
 - [Amazon RDS MariaDB 交代ユーザー](#)
 - [Amazon RDS および Amazon Aurora MySQL シングルユーザー](#)
 - [Amazon RDS および Amazon Aurora MySQL 交代ユーザー](#)
 - [Amazon RDS Oracle シングルユーザー](#)
 - [Amazon RDS Oracle 交代ユーザー](#)
 - [Amazon RDS および Amazon Aurora PostgreSQL シングルユーザー](#)
 - [Amazon RDS と Amazon Aurora PostgreSQL 交代ユーザー](#)
 - [Amazon RDS Microsoft SQLServer シングルユーザー](#)
 - [Amazon RDS Microsoft SQLServer 交代ユーザー](#)
- [Amazon DocumentDB \(MongoDB 互換性\)](#)
 - [Amazon DocumentDB シングルユーザー](#)
 - [Amazon DocumentDB 交代ユーザー](#)
- [Amazon Redshift](#)
 - [Amazon Redshift シングルユーザー](#)
 - [Amazon Redshift 交代ユーザー](#)

- [Amazon ElastiCache](#)
- [アクティブディレクトリ](#)
 - [Active Directory 認証情報](#)
 - [Active Directory キータブ](#)
- [その他のタイプのシークレット](#)

Amazon RDS と Amazon Aurora

Amazon RDS Db2 シングルユーザー

- テンプレート名 : SecretsManagerRDSDB2RotationSingleUser
- ローテーション戦略: [ローテーション戦略: シングルユーザー](#)
- **SecretString** 構造: [the section called “Amazon RDS Db2 シークレット構造”](#)
- ソースコード: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSDB2RotationSingleUser/lambda_function.py
- 依存関係: [python-ibmdb](#)

Amazon RDS Db2 交代ユーザー

- テンプレート名 : SecretsManagerRDSDB2RotationMultiUser
- ローテーション戦略: [the section called “交代ユーザー”](#)
- **SecretString** 構造: [the section called “Amazon RDS Db2 シークレット構造”](#)
- ソースコード : https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSDB2RotationMultiUser/lambda_function.py
- 依存関係: [python-ibmdb](#)

Amazon RDS MariaDB シングルユーザー

- テンプレート名 : SecretsManagerRDSMariaDBRotationSingleUser
- ローテーション戦略: [ローテーション戦略: シングルユーザー](#)
- **SecretString** 構造: [the section called “Amazon RDS MariaDB シークレット構造”](#)
- ソースコード: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSMariaDBRotationSingleUser/lambda_function.py

- 依存関係: PyMySQL 1.0.2。認証に sha256 パスワードを使用している場合は、PyMySQL[rsa]。Lambda ランタイムでコンパイルされたコードでパッケージを使用する方法については、「[AWS ナレッジセンター](#)」の「[コンパイルされたバイナリを含む Python パッケージをデプロイパッケージに追加し、パッケージを Lambda と互換性を持たせる方法](#)」を参照してください。

Amazon RDS MariaDB 交代ユーザー

- テンプレート名 : SecretsManagerRDSMariaDBRotationMultiUser
- ローテーション戦略: [the section called “交代ユーザー”](#)
- **SecretString** 構造: [the section called “Amazon RDS MariaDB シークレット構造”](#)
- ソースコード : https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSMariaDBRotationMultiUser/lambda_function.py
- 依存関係: PyMySQL 1.0.2。認証に sha256 パスワードを使用している場合は、PyMySQL[rsa]。Lambda ランタイムでコンパイルされたコードでパッケージを使用する方法については、「[AWS ナレッジセンター](#)」の「[コンパイルされたバイナリを含む Python パッケージをデプロイパッケージに追加し、パッケージを Lambda と互換性を持たせる方法](#)」を参照してください。

Amazon RDS および Amazon Aurora MySQL シングルユーザー

- テンプレート名 : SecretsManagerRDSMySQLRotationSingleUser
- ローテーション戦略: [the section called “シングルユーザー”](#)
- 期待される **SecretString** 構造体: [the section called “Amazon RDS と Amazon Aurora MySQL のシークレット構造”](#)
- ソースコード : https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSMySQLRotationSingleUser/lambda_function.py
- 依存関係: PyMySQL 1.0.2。認証に sha256 パスワードを使用している場合は、PyMySQL[rsa]。Lambda ランタイムでコンパイルされたコードでパッケージを使用する方法については、「[AWS ナレッジセンター](#)」の「[コンパイルされたバイナリを含む Python パッケージをデプロイパッケージに追加し、パッケージを Lambda と互換性を持たせる方法](#)」を参照してください。

Amazon RDS および Amazon Aurora MySQL 交代ユーザー

- テンプレート名 : SecretsManagerRDSMySQLRotationMultiUser
- ローテーション戦略: [the section called “交代ユーザー”](#)
- 期待される **SecretString**構造体: [the section called “Amazon RDS と Amazon Aurora MySQL のシークレット構造”](#)
- ソースコード: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSMySQLRotationMultiUser/lambda_function.py
- 依存関係: PyMySQL 1.0.2。認証に sha256 パスワードを使用している場合は、PyMySQL[rsa]。Lambda ランタイムでコンパイルされたコードでパッケージを使用する方法については、「[AWS ナレッジセンター](#)」の「[コンパイルされたバイナリを含む Python パッケージをデプロイパッケージに追加し、パッケージを Lambda と互換性を持たせる方法](#)」を参照してください。

Amazon RDS Oracle シングルユーザー

- テンプレート名 : SecretsManagerRDSOracleRotationSingleUser
- ローテーション戦略: [the section called “シングルユーザー”](#)
- 期待される **SecretString**構造体: [the section called “Amazon RDS Oracle シークレット構造”](#)
- ソースコード: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDS OracleRotationSingleUser/lambda_function.py
- 依存関係: [Python-oracledb 2.0.1](#)

Amazon RDS Oracle 交代ユーザー

- テンプレート名 : SecretsManagerRDSOracleRotationMultiUser
- ローテーション戦略: [the section called “交代ユーザー”](#)
- 期待される **SecretString**構造体: [the section called “Amazon RDS Oracle シークレット構造”](#)
- ソースコード: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDS OracleRotationMultiUser/lambda_function.py
- 依存関係: [Python-oracledb 2.0.1](#)

Amazon RDS および Amazon Aurora PostgreSQL シングルユーザー

- テンプレート名 : SecretsManagerRDSPostgreSQLRotationSingleUser
- ローターション戦略: [ローテーション戦略: シングルユーザー](#)
- 期待される **SecretString** 構造体: [the section called “Amazon RDS と Amazon Aurora PostgreSQL のシークレット構造”](#)
- ソースコード: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSPostgreSQLRotationSingleUser/lambda_function.py
- 依存関係: PyGreSQL 5.0.7

Amazon RDS と Amazon Aurora PostgreSQL 交代ユーザー

- テンプレート名 : SecretsManagerRDSPostgreSQLRotationMultiUser
- ローターション戦略: [the section called “交代ユーザー”](#)
- 期待される **SecretString** 構造体: [the section called “Amazon RDS と Amazon Aurora PostgreSQL のシークレット構造”](#)
- ソースコード: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSPostgreSQLRotationMultiUser/lambda_function.py
- 依存関係: PyGreSQL 5.0.7

Amazon RDS Microsoft SQLServer シングルユーザー

- テンプレート名 : SecretsManagerRDSSQLServerRotationSingleUser
- ローターション戦略: [the section called “シングルユーザー”](#)
- 期待される **SecretString** 構造体: [the section called “Amazon RDS Microsoft SQLServer シークレット構造”](#)
- ソースコード: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSSQLServerRotationSingleUser/lambda_function.py
- 依存関係: Pymssql 2.2.2

Amazon RDS Microsoft SQLServer 交代ユーザー

- テンプレート名 : SecretsManagerRDSSQLServerRotationMultiUser
- ローターション戦略: [the section called “交代ユーザー”](#)

- 期待される **SecretString** 構造体: [the section called “Amazon RDS Microsoft SQLServer シークレット構造”](#)
- ソースコード: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSPostgreSQLRotationMultiUser/lambda_function.py
- 依存関係: Pymssql 2.2.2

Amazon DocumentDB (MongoDB 互換性)

Amazon DocumentDB シングルユーザー

- テンプレート名: SecretsManagerMongoDBRotationSingleUser
- ローターション戦略: [the section called “シングルユーザー”](#)
- 期待される **SecretString** 構造体: [the section called “Amazon DocumentDB シークレット構造”](#)
- ソースコード: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerMongoDBRotationSingleUser/lambda_function.py
- 依存関係: Pymongo 3.2

Amazon DocumentDB 交代ユーザー

- テンプレート名: SecretsManagerMongoDBRotationMultiUser
- ローターション戦略: [the section called “交代ユーザー”](#)
- 期待される **SecretString** 構造体: [the section called “Amazon DocumentDB シークレット構造”](#)
- ソースコード: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerMongoDBRotationMultiUser/lambda_function.py
- 依存関係: Pymongo 3.2

Amazon Redshift

Amazon Redshift シングルユーザー

- テンプレート名: SecretsManagerRedshiftRotationSingleUser
- ローターション戦略: [the section called “シングルユーザー”](#)
- 予想される **SecretString** 構造: [the section called “Amazon Redshift のシークレット構造”](#) または [the section called “Amazon Redshift Serverless シークレット構造”](#)。

- ソースコード: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRedshiftRotationSingleUser/lambda_function.py
- 依存関係: PyGreSQL 5.0.7

Amazon Redshift 交代ユーザー

- テンプレート名: SecretsManagerRedshiftRotationMultiUser
- ローターション戦略: [the section called “交代ユーザー”](#)
- 予想される **SecretString** 構造: [the section called “Amazon Redshift のシークレット構造”](#) または [the section called “Amazon Redshift Serverless シークレット構造”](#)。
- ソースコード: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRedshiftRotationMultiUser/lambda_function.py
- 依存関係: PyGreSQL 5.0.7

Amazon ElastiCache

このテンプレートを使用するには、「Amazon ユーザーガイド」の「[ユーザーのパスワードの自動ローテーション](#)」を参照してください。 ElastiCache

- テンプレート名: SecretsManagerElasticacheUserRotation
- 期待される **SecretString** 構造体: [the section called “Amazon ElastiCache シークレット構造”](#)
- ソースコード: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerElasticacheUserRotation/lambda_function.py

アクティブディレクトリ

Active Directory 認証情報

- テンプレート名: SecretsManagerActiveDirectoryRotationSingleUser
- 期待される **SecretString** 構造体: [the section called “Active Directory 認証情報シークレット構造”](#)
- ソースコード: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerActiveDirectoryRotationSingleUser/lambda_function.py

Active Directory キータブ

- テンプレート名 : SecretsManagerActiveDirectoryAndKeytabRotationSingleUser
- 期待される **SecretString** 構造体: [the section called “Active Directory シークレット構造”](#)
- ソースコード: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerActiveDirectoryAndKeytabRotationSingleUser/lambda_function.py
- 依存関係 : msutil

その他のタイプのシークレット

Secrets Manager は、任意のタイプのシークレットのローテーション関数を作成するための開始点として、このテンプレートを提供します。

- テンプレート名 : SecretsManagerRotationTemplate
- ソースコード: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRotationTemplate/lambda_function.py

の Lambda ローテーション関数の実行ロールのアクセス許可 AWS Secrets Manager

の場合 [the section called “Lambda 関数によるローテーション”](#)、Secrets Manager が Lambda 関数を使用してシークレットをローテーションすると、Lambda は [IAM 実行ロール](#) を引き受け、それらの認証情報を Lambda 関数コードに提供します。自動ローテーションを設定する方法については、以下を参照してください。

- [データベースシークレットの自動ローテーション](#)
- [データベース以外のシークレットの自動ローテーション](#)

次の例は、Lambda ローテーション関数の実行ロールのインラインポリシーを示しています。実行ロールを作成し、アクセス権限ポリシーをアタッチするには、[を参照してください。AWS Lambda 実行ロール。](#)

例:

- [Lambda ローテーション関数の実行ロールのポリシー](#)
- [カスタマーマネージドキーのポリシーステートメント](#)

• [交代ユーザー戦略のポリシーステートメント](#)

Lambda ローテーション関数の実行ロールのポリシー

次のポリシーの例では、ローテーション関数が次の操作を許可します。

- *SecretARN* の Secrets Manager 操作を実行します。
- 新しいパスワードを作成します。
- データベースまたはサービスが VPC で実行されている場合、必要な設定のセットアップを行います。[VPC 内のリソースにアクセスするように Lambda 関数を設定する](#)、を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:DescribeSecret",
        "secretsmanager:GetSecretValue",
        "secretsmanager:PutSecretValue",
        "secretsmanager:UpdateSecretVersionStage"
      ],
      "Resource": "SecretARN"
    },
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetRandomPassword"
      ],
      "Resource": "*"
    },
    {
      "Action": [
        "ec2:CreateNetworkInterface",
        "ec2:DeleteNetworkInterface",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DetachNetworkInterface"
      ],
      "Resource": "*",
      "Effect": "Allow"
    }
  ]
}
```



```
]
}
```

カスタマーマネージドキーのポリシーステートメント

AWS マネージドキー `aws/secretsmanager` 以外の KMS キーを使用しシークレットを暗号化する場合は、そのキーの使用に関するアクセス許可を、Lambda の実行ロールに付与する必要があります。[SecretArn 暗号化コンテキスト](#)を使用して復号化関数の使用を制限できます。この場合、ローテーション関数ロールには、ローテーションに使用するシークレットを復号化するアクセスのみが許可されます。次に、実行ロールポリシーに追加してKMS キーを使用してシークレットを復号化するステートメントの例を示します。

```
{
  "Effect": "Allow",
  "Action": [
    "kms:Decrypt",
    "kms:DescribeKey",
    "kms:GenerateDataKey"
  ],
  "Resource": "KMSKeyARN"
  "Condition": {
    "StringEquals": {
      "kms:EncryptionContext:SecretARN": "SecretARN"
    }
  }
}
```

カスタマーマネージドキーで暗号化された複数のシークレットに対してローテーション機能を使用するには、以下の例のようなステートメントを追加して、実行ロールがシークレットを復号化できるようにします。

```
{
  "Effect": "Allow",
  "Action": [
    "kms:Decrypt",
    "kms:DescribeKey",
    "kms:GenerateDataKey"
  ],
  "Resource": "KMSKeyARN"
  "Condition": {
    "StringEquals": {
```

```
        "kms:EncryptionContext:SecretARN": [
            "arn1",
            "arn2"
        ]
    }
}
```

交代ユーザー戦略のポリシーステートメント

交代ユーザーローテーション戦略については、「[the section called “Lambda 関数のローテーション戦略”](#)」を参照してください。

Amazon RDS 認証情報を含むシークレットで、代替ユーザー戦略を使用しており、スーパーユーザーシークレットが [Amazon RDS によって管理](#)されている場合、ローテーション関数が Amazon RDS の読み取り専用 API を呼び出して、データベースの接続情報を取得できるようにする必要があります。AWS 管理ポリシー [AmazonRDSReadOnlyAccess](#) をアタッチすることをお勧めします。

次のポリシーの例では、関数が次の操作を許可します。

- **SecretARN** の Secrets Manager 操作を実行します。
- スーパーユーザーシークレットで認証情報を取得します。Secrets Manager は、スーパーユーザーシークレットの認証情報を使用し、ローテーションされたシークレットの認証情報を更新します。
- 新しいパスワードを作成します。
- データベースまたはサービスが VPC で実行される場合、必要な設定のセットアップを行います。詳細については、「[VPC 内のリソースにアクセスするように Lambda 関数を設定する](#)」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:DescribeSecret",
        "secretsmanager:GetSecretValue",
        "secretsmanager:PutSecretValue",
        "secretsmanager:UpdateSecretVersionStage"
      ]
    }
  ],
}
```

```
    "Resource": "SecretARN",
  },
  {
    "Effect": "Allow",
    "Action": [
      "secretsmanager:GetSecretValue"
    ],
    "Resource": "SuperuserSecretARN"
  },
  {
    "Effect": "Allow",
    "Action": [
      "secretsmanager:GetRandomPassword"
    ],
    "Resource": "*"
  },
  {
    "Action": [
      "ec2:CreateNetworkInterface",
      "ec2>DeleteNetworkInterface",
      "ec2:DescribeNetworkInterfaces",
      "ec2:DetachNetworkInterface"
    ],
    "Resource": "*",
    "Effect": "Allow"
  }
]
```

Lambda ローテーション関数のネットワークアクセス

の場合 [the section called “Lambda 関数によるローテーション”](#)、Secrets Manager が Lambda 関数を使用してシークレットをローテーションするとき、Lambda ローテーション関数はシークレットにアクセスできる必要があります。シークレットに認証情報が含まれている場合、Lambda 関数はそれらの認証情報のソース (データベースやサービスなど) にもアクセスできる必要があります。

シークレットにアクセスするには

ローテーション用の Lambda 関数は、Secrets Manager のエンドポイントにアクセスする必要があります。Lambda 関数がインターネットにアクセスできる場合は、パブリックなエンドポイントを使用できます。エンドポイントを見つけるには、「[the section called “Secrets Manager エンドポイント”](#)」を参照してください。

インターネットにアクセスできない VPC で Lambda 関数を実行する場合は、Secrets Manager サービスのプライベートなエンドポイントを、VPC 内に設定することをお勧めします。VPC は、リージョンのパブリックなエンドポイントに向けられたリクエストを傍受し、それらをプライベートエンドポイントにリダイレクトします。(詳しくは、「[VPC エンドポイント](#)」を参照してください。)

別の方法としては、[NAT ゲートウェイ](#)または[インターネットゲートウェイ](#)を VPC に追加して(これで VPC のトラフィックはパブリックエンドポイントに到達できます)、Lambda 関数から Secrets Manager のパブリックなエンドポイントへのアクセスを許可することも考えられます。この方法では、VPC がある程度のリスクにさらされることになります。ゲートウェイ向けの IP アドレスには、パブリックなインターネットから攻撃が可能なためです。

(オプション) データベースまたはサービスにアクセスするには

API キーなどのシークレットについては、シークレットと一緒に更新する必要があるソースデータベースやサービスはありません。

データベースまたはサービスを VPC の Amazon EC2 インスタンスで実行している場合は、同じ VPC で Lambda 関数を設定することをお勧めします。こうすることで、ローテーション関数はサービスと直接通信できるようになります。詳細については、[Configuring VPC access](#) を参照してください。

Lambda 関数からデータベースまたはサービスへのアクセスを可能にするには、ローテーション用の Lambda 関数にアタッチされたセキュリティグループによって、そのデータベースまたはサービスに対するアウトバウンド接続が許可されている必要があります。同時に、データベースまたはサービスにアタッチされているセキュリティグループでは、ローテーション用 Lambda 関数からのインバウンド接続を許可する必要もあります。

AWS Secrets Manager ローテーションのトラブルシューティング

多くのサービスでは、Secrets Manager は、Lambda 関数を使用してシークレットをローテーションします。詳細については、「[the section called “Lambda 関数によるローテーション”](#)」を参照してください。Lambda ローテーション関数は、シークレットの対象となるデータベースまたはサービス、および Secrets Manager とやり取りします。ローテーションが期待どおりに機能しない場合は、まず CloudWatch ログを確認する必要があります。

Note

一部のサービスは、ユーザーのためにシークレットを管理できます (自動ローテーションの管理など)。詳細については、「[the section called “マネージドローテーション”](#)」を参照してください。

Lambda 関数の CloudWatch ログを表示するには

1. Secrets Manager のコンソール (<https://console.aws.amazon.com/secretsmanager/>) を開きます。
2. シークレットを選択し、詳細ページの [Rotation configuration] (ローテーション設定) で、Lambda ローテーション関数を選択します。Lambda コンソールが開きます。
3. Monitor タブで、Logs を選択し、次に でログを表示する CloudWatchを選択します。CloudWatch コンソールが開き、関数のログが表示されます。

ログを解釈するには

- [「環境変数に認証情報が見つかりました」の後にアクティビティがない](#)
- [「CreateSecret」の後にアクティビティがない](#)
- [エラー: 「KMS へのアクセスは許可されていません」](#)
- [エラー: 「シークレット JSON にキーがありません」](#)
- [エラー: 「setSecret: データベースにログインできません」](#)
- [エラー: 「モジュール 'lambda_function' をインポートできません」](#)
- [既存のローテーション関数を Python 3.7 から 3.9 にアップグレードする](#)

「環境変数に認証情報が見つかりました」の後にアクティビティがない

「環境変数に認証情報が見つかりました」の後にアクティビティがなく、タスクの所要時間が長い (例: デフォルトの Lambda タイムアウトは 30000 ms) 場合は、Secrets Manager エンドポイントへのアクセス時に Lambda 関数がタイムアウトしている可能性があります。

ローテーション用の Lambda 関数は、Secrets Manager のエンドポイントにアクセスする必要があります。Lambda 関数がインターネットにアクセスできる場合は、パブリックなエンドポイントを使用できます。エンドポイントを見つけるには、「[the section called “Secrets Manager エンドポイント”](#)」を参照してください。

インターネットにアクセスできない VPC で Lambda 関数を実行する場合は、Secrets Manager サービスのプライベートなエンドポイントを、VPC 内に設定することをお勧めします。VPC は、リージョンのパブリックなエンドポイントに向けられたリクエストを傍受し、それらをプライベートエンドポイントにリダイレクトします。(詳しくは、「[VPC エンドポイント](#)」を参照してください。)

別の方法としては、[NAT ゲートウェイ](#)または[インターネットゲートウェイ](#)を VPC に追加して(これで VPC のトラフィックはパブリックエンドポイントに到達できます)、Lambda 関数から Secrets Manager のパブリックなエンドポイントへのアクセスを許可することも考えられます。この方法では、VPC がある程度リスクにさらされることとなります。ゲートウェイ向けの IP アドレスには、パブリックなインターネットから攻撃が可能なためです。

「CreateSecret」の後にアクティビティがない

CreateSecret の実行後にローテーションが停止する原因となる問題は次のとおりです。

VPC ネットワーク ACL では、HTTPS トラフィックの送受信が許可されません。

詳細については、「Amazon VPC ユーザーガイド」の「[ネットワーク ACL を使用してサブネットへのトラフィックを制御する](#)」を参照してください。

Lambda 関数のタイムアウト設定が短すぎてタスクを実行できません。

詳細については、「AWS Lambda デベロッパーガイド」の「[Lambda 関数オプションの設定](#)」を参照してください。

Secrets Manager VPC エンドポイントは、割り当てられたセキュリティグループへの進入時に VPC CIDR を許可しません。

詳細については、「Amazon Virtual Private Cloud ユーザーガイド」の「[Control traffic to resources using security groups](#)」(セキュリティグループを使用してリソースへのトラフィックを制御する)を参照してください。

Secrets Manager VPC エンドポイントポリシーでは、Lambda が VPC エンドポイントを使用することを許可していません。

詳細については、「[VPC エンドポイント](#)」を参照してください。

シークレットは交代ユーザーローテーションを使用し、スーパーユーザーシークレットは Amazon RDS によって管理され、Lambda 関数は RDS API にアクセスできません。

スーパー[ユーザーシークレット](#)が別の AWS サービスによって管理される交代ユーザーローテーションの場合、Lambda ローテーション関数はサービスエンドポイントを呼び出してデータベース接続情報を取得する必要があります。データベースサービスに VPC エンドポイントを設定することを推奨します。詳細については、以下を参照してください。

- 「Amazon RDS ユーザーガイド」の「[Amazon RDS API およびインターフェース VPC エンドポイント](#)」
- 「Amazon Redshift 管理ガイド」の「[VPC エンドポイントの操作](#)」

エラー: 「KMS へのアクセスは許可されていません」

ClientError: An error occurred (AccessDeniedException) when calling the GetSecretValue operation: Access to KMS is not allowed が表示されている場合、シークレットの暗号化に使用された KMS キーを使用してシークレットを復号化するアクセス許可が、ローテーション関数に付与されていません。暗号化コンテキストを特定のシークレットに制限する条件が、アクセス許可ポリシーに含まれている可能性があります。必要なアクセス許可の詳細については、「[the section called “カスタマーマネージドキーのポリシーステートメント”](#)」を参照してください。

エラー: 「シークレット JSON にキーがありません」

Lambda ローテーション関数では、シークレット値が特定の JSON 構造になっている必要があります。このエラーが表示される場合は、ローテーション関数がアクセスしようとしたキーが JSON にはない可能性があります。各タイプのシークレットの JSON 構造については、「[the section called “シークレットの JSON 構造”](#)」を参照してください。

エラー: 「setSecret: データベースにログインできません」

このエラーを引き起こす可能性のある問題は次のとおりです。

ローテーション関数はデータベースにアクセスできません。

タスクの所要時間が長い (例: 5000 ミリ秒以上) 場合、Lambda ローテーション関数はネットワーク経由でデータベースにアクセスできない可能性があります。

データベースまたはサービスを VPC の Amazon EC2 インスタンスで実行している場合は、同じ VPC で Lambda 関数を設定することをお勧めします。こうすることで、ローテーション関数はサービスと直接通信できるようになります。詳細については、[Configuring VPC access](#) を参照してください。

Lambda 関数からデータベースまたはサービスへのアクセスを可能にするには、ローテーション用の Lambda 関数にアタッチされたセキュリティグループによって、そのデータベースまたはサービスに対するアウトバウンド接続が許可されている必要があります。同時に、データベース

またはサービスにアタッチされているセキュリティグループでは、ローテーション用 Lambda 関数からのインバウンド接続を許可する必要があります。

シークレットの認証情報が正しくありません。

タスクの所要時間が短い場合、Lambda ローテーション関数がシークレット内の認証情報を使用しても認証できない可能性があります。AWS CLI コマンド を使用して、シークレットの `AWSCURRENT` および `AWSPREVIOUS` バージョンの情報を使用して手動でログインし、認証情報を確認します [get-secret-value](#)。

データベースは `scram-sha-256` を使用してパスワードを暗号化します。

Aurora PostgreSQL バージョン 13 以降のデータベースで、パスワードの暗号化に `scram-sha-256` を使用しているが、ローテーション関数が `scram-sha-256` をサポートしていない `libpq` バージョン 9 以前を使用している場合、ローテーション関数はデータベースに接続できません。

scram-sha-256 暗号化を使用するデータベースユーザーを判別するには

- ブログ「[SCRAM Authentication in RDS for PostgreSQL 13](#)」(RDS for PostgreSQL 13 での SCRAM 認証) の「Checking for users with non-SCRAM passwords」(SCRAM 以外のパスワードを持つユーザーの確認) を参照してください。

ローテーション関数が使用する `libpq` のバージョンを判別するには

1. Linux ベースのコンピュータの Lambda コンソールで、ローテーション関数に移動し、デプロイバンドルをダウンロードします。zip ファイルを作業ディレクトリに解凍します。
2. コマンドラインの作業ディレクトリで、以下を実行します。

```
readelf -a libpq.so.5 | grep RUNPATH
```

3. 文字列 `PostgreSQL-9.4.x`、または 10 未満のメジャーバージョンが表示されている場合、ローテーション関数は `scram-sha-256` をサポートしていません。
 - `scram-sha-256` をサポートしていないローテーション関数の出力を次に示します。

```
0x0000000000000001d (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-256` をサポートしているローテーション関数の出力を次に示します。

```
0x0000000000000001d (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]
```

Note

2021 年 12 月 30 日より前に自動シークレットローテーションを設定した場合、ローテーション関数には `scram-sha-256` をサポートしていない古いバージョンの `libpq` がバンドルされています。`scram-sha-256` をサポートするには、[ローテーション関数を再作成](#)する必要があります。

データベースには SSL/TLS アクセスが必要です。

SSL/TLS 接続が必要なデータベースを使用しているが、ローテーション関数が暗号化されていない接続を使用する場合、ローテーション関数はデータベースに接続できません。Amazon RDS (Oracle と Db2 を除く) および Amazon DocumentDB のローテーション関数では、使用可能な場合、データベースへの接続に Secure Sockets Layer (SSL) または Transport Layer Security (TLS) が使用されます。使用できない場合は、暗号化されていない接続が使用されます。

Note

2021 年 12 月 20 日より前に自動シークレットローテーションを設定した場合は、ローテーション関数が SSL/TLS をサポートしていない古いテンプレートに基づいている可能性があります。SSL/TLS を使用する接続をサポートするには、[ローテーション関数を再作成](#)する必要があります。

ローテーション関数がいつ作成されたかを特定するには

1. Secrets Manager コンソール (<https://console.aws.amazon.com/secretsmanager/>) で、シークレットを開きます。[Rotation configuration] (ローテーション構成) セクションの [Lambda rotation function] (Lambda ローテーション関数) の下に、[Lambda

function ARN] (Lambda 関数 ARN) が表示されます (`arn:aws:lambda:aws-region:123456789012:function:SecretsManagerMyRotationFunction` など)。ARN の末尾から関数名をコピーします (この例では `SecretsManagerMyRotationFunction`)。

2. AWS Lambda コンソール <https://console.aws.amazon.com/lambda/> の関数 で、検索ボックスに Lambda 関数名を貼り付け、Enter を選択し、Lambda 関数を選択します。
3. 関数の詳細ページで、[Configuration] (設定) タブの [Tags] (タグ) で、`aws:cloudformation:stack-name` キーの横にある値をコピーします。
4. AWS CloudFormation コンソール <https://console.aws.amazon.com/cloudformation> のスタックで、検索ボックスにキー値を貼り付け、Enter を選択します。
5. スタックのリストがフィルタリングされ、Lambda ローテーション関数を作成したスタックだけが表示されます。[Created date] (作成日) 列に、スタックが作成された日付が表示されます。これが、Lambda ローテーション関数が作成された日付です。

エラー: 「モジュール 'lambda_function' をインポートできません」

古い (Python 3.7 から新しいバージョンの Python に自動的にアップグレードされた) Lambda 関数を実行している場合に、このエラーが表示されることがあります。このエラーを解決するには、Lambda 関数のバージョンを Python 3.7 に戻してから、[the section called “既存のローテーション関数を Python 3.7 から 3.9 にアップグレードする”](#) を実行します。詳細については、「AWS re:Post」の「[Secrets Manager Lambda 関数のローテーションが「pg モジュールが見つかりません」というエラーで失敗したのはなぜですか?](#)」を参照してください。

既存のローテーション関数を Python 3.7 から 3.9 にアップグレードする

2022 年 11 月よりも前に作成された一部のローテーション関数では、Python 3.7 が使用されていました。AWS SDK for Python は 2023 年 12 月に Python 3.7 のサポートを停止しました。詳細については、[AWS SDKs](#) を参照してください。Python 3.9 を使用する新しいローテーション関数に切り替えるには、既存のローテーション関数にランタイムプロパティを追加するか、またはローテーション関数を再作成します。

Python 3.7 を使用する Lambda ローテーション関数を見つけるには

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/lambda/> で AWS Lambda コンソールを開きます。
2. [関数] のリストで、**SecretsManager** をフィルタリングします。
3. フィルタリングされた関数のリストの [ランタイム] で、Python 3.7 を見つけます。

Python 3.9 にアップグレードするには:

- [オプション 1: を使用してローテーション関数を再作成する AWS CloudFormation](#)
- [オプション 2: を使用して既存のローテーション関数のランタイムを更新する AWS CloudFormation](#)
- [オプション 3: AWS CDK ユーザーの場合は CDK ライブラリをアップグレードする](#)

オプション 1: を使用してローテーション関数を再作成する AWS CloudFormation

Secrets Manager コンソールを使用してローテーションを有効にすると、Secrets Manager は AWS CloudFormation を使用して Lambda ローテーション関数を含む必要なリソースを作成します。コンソールを使用してローテーションを有効にした場合、または AWS CloudFormation スタックを使用してローテーション関数を作成した場合は、同じ AWS CloudFormation スタックを使用してローテーション関数を新しい名前で作成できます。新しい関数は、より新しいバージョンの Python を使用します。

ローテーション関数を作成した AWS CloudFormation スタックを検索するには

- Lambda 関数の詳細ページの [設定] タブで、[タグ] を選択します。aws:cloudformation:stack-id の横にある ARN を表示します。

次の例に示すように、スタック名は ARN に埋め込まれます。

- ARN: `arn:aws:cloudformation:us-west-2:408736277230:stack/SecretsManagerRDSMySQLRotationSingleUser5c2-SecretRotationScheduleHostedRotationLambda-3CUDHZMDMB08/79fc9050-2eef-11ed-`
- スタック名: `SecretsManagerRDSMySQLRotationSingleUser5c2-SecretRotationScheduleHostedRotationLambda`

ローテーション関数を再作成するには (AWS CloudFormation)

1. で AWS CloudFormation、スタックを名前で検索し、 の更新を選択します。

ルートスタックの更新を推奨するダイアログボックスが表示された場合は、[ルートスタックに移動] を選択し、[更新] を選択します。

2. [スタックを更新] ページで、[デザイナーでテンプレートを編集] を選択し、[デザイナーで表示] を選択します。

3. デザイナーのテンプレートコードの `SecretRotationScheduleHostedRotationLambda` で、`"functionName": "SecretsManagerTestRotationRDS"` の値を (JSON など) 新しい関数名 `"functionName": "SecretsManagerTestRotationRDSUpdated"` に置き換えます。
4. AWS CloudFormation スタックワークフローを続行し、送信 を選択します。

オプション 2: を使用して既存のローテーション関数のランタイムを更新する AWS CloudFormation

Secrets Manager コンソールを使用してローテーションを有効にすると、Secrets Manager は AWS CloudFormation を使用して Lambda ローテーション関数を含む必要なリソースを作成します。コンソールを使用してローテーションを有効にした場合、または AWS CloudFormation スタックを使用してローテーション関数を作成した場合は、同じ AWS CloudFormation スタックを使用してローテーション関数のランタイムを更新できます。

ローテーション関数を作成した AWS CloudFormation スタックを検索するには

- Lambda 関数の詳細ページの [設定] タブで、[タグ] を選択します。aws:cloudformation:stack-id の横にある ARN を表示します。

次の例に示すように、スタック名は ARN に埋め込まれます。

- ARN: `arn:aws:cloudformation:us-west-2:408736277230:stack/SecretsManagerRDSMySQLRotationSingleUser5c2-SecretRotationScheduleHostedRotationLambda-3CUDHZMDMB08/79fc9050-2eef-11ed-`
- スタック名: `SecretsManagerRDSMySQLRotationSingleUser5c2-SecretRotationScheduleHostedRotationLambda`

ローテーション関数のランタイムを更新するには (AWS CloudFormation)

1. で AWS CloudFormation、スタックを名前を検索し、 の更新を選択します。

ルートスタックの更新を推奨するダイアログボックスが表示された場合は、[ルートスタックに移動] を選択し、[更新] を選択します。

2. [スタックを更新] ページで、[デザイナーでテンプレートを編集] を選択し、[デザイナーで表示] を選択します。

3. デザイナーのテンプレート JSON で、 の `SecretRotationScheduleHostedRotationLambda`、 の `Properties`、 の `Parameters` で、 を追加します。 `"runtime": "python3.9"`
4. AWS CloudFormation スタックワークフローを続行し、送信 を選択します。

オプション 3: AWS CDK ユーザーの場合は CDK ライブラリをアップグレードする

バージョン v2.94.0 より AWS CDK 前の を使用してシークレットのローテーションを設定した場合は、v2.94.0 以降にアップグレードすることで Lambda 関数を更新できます。詳細については、「[AWS Cloud Development Kit \(AWS CDK\) v2 デベロッパーガイド](#)」を参照してください。

シー AWS Secrets Manager クレットをすぐにローテーションする

ローテーションできるのは、ローテーションが設定されているシークレットのみです。シークレットにローテーションが設定されているかどうかを確認するには、コンソールでシークレットを表示し、[Rotation configuration] (ローテーション設定) セクションまでスクロールします。[Rotation status] (ローテーションステータス) が [Enabled] (有効) の場合、シークレットにローテーションが設定されます。そうでない場合は、「[シークレットのローテーション](#)」を参照してください。

すぐにシークレットをローテーションするには (コンソール)

1. Secrets Manager のコンソール (<https://console.aws.amazon.com/secretsmanager/>) を開きます。
2. シークレットを選択します。
3. [Secret Details] (シークレットの詳細) ページの、[Rotation configuration] (ローテーション設定) で、[Rotate secret immediately] (すぐにシークレットをローテーションさせる) をクリックします。
4. [Rotate secret] (シークレットのローテーション) ダイアログボックスで、[Rotate] (ローテーション) をクリックします。

AWS CLI

Example `rotate-secret` を使ってすぐにシークレットをローテーションする

次の `rotate-secret` の例では、すぐにローテーションが開始されます。シークレットのローテーションは、すでに設定されている必要があります。

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

ローテーションスケジュール

自動ローテーションをオンにすると、Cron() または Rate() 式を使用して、シークレットをローテーションするスケジュールを設定できます。Rate 式を使用すると、一定の時間間隔や日数間隔で繰り返されるローテーションスケジュールを作成できます。Cron 式を使用すると、ローテーション間隔よりも詳細なローテーションスケジュールを作成できます。Secrets Manager のローテーションスケジュールでは、UTC タイムゾーンが使用されます。シークレットが 4 時間ごとにローテーションされるように設定できます。Secrets Manager は、ローテーションウィンドウ中の任意の時刻にシークレットをローテーションします。

ローテーションの有効化については、以下を参照してください。

- [the section called “マネージドローテーション”](#)
- [the section called “データベースシークレットの自動ローテーション”](#)
- [the section called “データベース以外のシークレットの自動ローテーション”](#)

Rate 式

Secrets Manager の Rate 式は、次のような形式です。*Value* は正の整数で、*Unit* は hour、hours、day、または days にできます。

```
rate(Value Unit)
```

シークレットが 4 時間ごとにローテーションされるように設定できます。例:

- rate(4 hours) は、シークレットが 4 時間ごとにローテーションされることを意味します。
- rate(1 day) は、シークレットが毎日ローテーションされることを意味します。
- rate(10 days) は、シークレットが 10 日ごとにローテーションされることを意味します。

Rate 式の hours の場合、デフォルトのローテーションウィンドウは午前 0 時に開始され、1 時間後に終了します。ウィンドウ期間を設定して、ローテーションウィンドウを変更できます。ローテーションウィンドウが次のローテーションウィンドウに重ならないようにしてください。これをチェッ

クする方法の 1 つは、ローテーションウィンドウの時間数がローテーション間の時間数以下であることを確認することです。

Rate 式の days の場合、デフォルトのローテーションウィンドウは午前 0 時に開始され、1 日の終わりに終了します。ウィンドウ期間を設定して、ローテーションウィンドウを変更できます。ウィンドウが次の UTC 日に入ってはなりません。これをチェックする方法の 1 つは、開始時間にウィンドウ期間を加えた時間が 24 時間以下であることを確認することです。

Cron 式

Cron 式の形式は次のようになります。

```
cron(Minutes Hours Day-of-month Month Day-of-week Year)
```

時間の増分を含む cron 式は、毎日リセットされます。例えば、cron(0 4/12 * * ? *) は午前 4 時、および午後 4 時、次いで翌日の午前 4 時、および午後 4 時を意味します。Secrets Manager のローテーションスケジュールでは、UTC タイムゾーンが使用されます。

時間単位のスケジュールの場合、デフォルトのローテーションウィンドウは 1 時間後に終了します。ウィンドウ期間を設定して、ローテーションウィンドウを変更できます。ローテーションウィンドウが次のローテーションウィンドウに重ならないようにしてください。シークレットが 4 時間ごとにローテーションされるように設定できます。

スケジュールの例	式
8 時間ごと、午前 0 時から開始。	cron(0 /8 * * ? *)
8 時間ごと、午前 8 時から開始。	cron(0 8/8 * * ? *)
10 時間ごと、午前 2 時から開始。	cron(0 2/10 * * ? *)
ローテーションウィンドウは 2:00、12:00、および 22:00、次いで翌日の 2:00、12:00、および 22:00 に開始されます。	
毎日午前 10:00。	cron(0 10 * * ? *)
毎週土曜日の午後 6:00。	cron(0 18 ? * SAT *)
毎月 1 日の午前 8:00。	cron(0 8 1 * ? *)

スケジュールの例	式
3 か月ごとに第 1 日曜日の午前 1:00。	<code>cron(0 1 ? 1/3 SUN#1 *)</code>
毎月最終日の午後 5:00。	<code>cron(0 17 L * ? *)</code>
月曜日から金曜日までの午前 8:00。	<code>cron(0 8 ? * MON-FRI *)</code>
毎月 1 日と 15 日の午後 4:00。	<code>cron(0 16 1,15 * ? *)</code>
毎月第 1 日曜日の午前 0:00。	<code>cron(0 0 ? * SUN#1 *)</code>

Secrets Manager の cron 式要件

Secrets Manager では、cron 式に使用できる設定値にはいくつかの制限があります。Secrets Manager の cron 式では、分フィールドに 0 が必要です。これは、Secrets Manager のローテーションウィンドウが正時に開始されるためです。年フィールドには、* が必要です。これは、Secrets Manager では 1 年以上離れているローテーションスケジュールがサポートされていないためです。次の表に、使用できるオプションを示します。

フィールド	値	ワイルドカード
分	0 を指定	なし
時間	0 ~ 23	/ (フォワードスラッシュ) を使用して増分を指定します。例えば、2/10 は午前 2 時から 10 時間ごとを意味します。シークレットが 4 時間ごとにローテーションされるように設定できます。
Day-of-month	1 ~ 31	, (カンマ) を使用して追加の値を含めます。例えば、1,15 は月の 1 日と 15 日を意味します。

フィールド	値	ワイルドカード
		<p>– (ダッシュ) を使用して範囲を指定します。例えば、1-15 は月の 1 日から 15 日までの日を意味します。</p> <p>* (アスタリスク) を使用してフィールド内のすべての値を含めます。例えば、* は月のすべての日を意味します。</p> <p>[?] (疑問符) のワイルドカードは、任意を意味します。Cron 式の Day-of-month フィールドと Day-of-week フィールドを同時に指定することはできません。一方のフィールドに値を指定する場合、もう一方のフィールドで [?] (疑問符) を使用する必要があります。</p> <p>/ (フォワードスラッシュ) を使用して増分を指定します。例えば、1/2 は 1 日目から 2 日おき、つまり 1 日目、3 日目、5 日目などを意味します。</p> <p>L を使用して月の最終日を指定します。</p> <p>DAYL を使用して、月の最終曜日を指定します。例えば、SUNL は月の最終日曜日を意味します。</p>

フィールド	値	ワイルドカード
月	1~12 または JAN~DEC	<p>, (カンマ) を使用して追加の値を含めます。例えば、JAN, APR, JUL, OCT は 1 月、4 月、7 月、および 10 月を意味します。</p> <p>- (ダッシュ) を使用して範囲を指定します。例えば、1-3 とは 1 年の 1 か月目から 3 か月目までを意味します。</p> <p>* (アスタリスク) を使用してフィールド内のすべての値を含めます。例えば、* はすべての月を意味します。</p> <p>/ (フォワードスラッシュ) を使用して増分を指定します。例えば、1/3 は、1 か月目から 3 か月おき、つまり 1 か月目、4 か月目、7 か月目、10 か月目などを意味します。</p>

フィールド	値	ワイルドカード
Day-of-week	1~7 または SUN~SAT	<p># を使用して月内の曜日を指定します。例えば、TUE#3 は月の第 3 火曜日を意味します。</p> <p>,(カンマ) を使用して追加の値を含めます。例えば、1,4 は 1 日目の曜日と 4 日目の曜日を意味します。</p> <p>-(ダッシュ) を使用して範囲を指定します。例えば、1-4 は 1 日目から 4 日目までの曜日を意味します。</p> <p>*(アスタリスク) を使用してフィールド内のすべての値を含めます。例えば、* はすべての曜日を意味します。</p> <p>[?](疑問符) のワイルドカードは、任意を意味します。 。Cron 式の Day-of-month フィールドと Day-of-week フィールドを同時に指定することはできません。一方のフィールドに値を指定する場合、もう一方のフィールドで [?](疑問符) を使用する必要があります。</p> <p>/(フォワードスラッシュ) を使用して増分を指定します。例えば、1/2 は 1 日目の曜日から 2 日おき、つまり 1 日</p>

フィールド	値	ワイルドカード
		目、3日目、5日目、7日目などの曜日を意味します。 Lを使用して最終曜日を指定します。
年	* を指定してください	なし

ローテーションされていないシークレットを検索する

AWS Config を使用してシークレットを評価し、標準に従ってローテーションしているかどうかを確認できます。AWS Config ルールを使用して、シークレットの内部セキュリティおよびコンプライアンス要件を定義します。その後 AWS Config、はルールに準拠していないシークレットを識別できます。また、シークレットメタデータ、ローテーション設定、シークレット暗号化に使用される KMS キー、Lambda ローテーション関数、およびシークレットに関連付けられたタグの変更を追跡することもできます。

複数の AWS アカウント と組織 AWS リージョン 内にシークレットがある場合は、その設定とコンプライアンスデータを集約できます。詳細については、[「マルチアカウントマルチリージョンデータ集約」](#)を参照してください。

シークレットがローテーションしているかどうかを評価するには

1. [AWS Config ルール を使用してリソースを評価する](#) 手順に従い、次のルールから選択します。
 - [secretsmanager-rotation-enabled-check](#) — Secrets Manager に保存されているシークレットに対してローテーションが設定されているかどうかを確認します。
 - [secretsmanager-scheduled-rotation-success-check](#) – 最後に成功したローテーションが、設定されたローテーション頻度の範囲内であるかどうかをチェックします。チェックの最小頻度は日次です。
 - [secretsmanager-secret-periodic-rotation](#) — 指定した日数内にシークレットがローテーションされたかどうかを確認します。
2. オプションで、シークレットが準拠していない場合に通知する AWS Config ように を設定します。詳細については、[「 が Amazon SNS トピック AWS Config に送信する通知」](#)を参照してください。

Secrets Manager で自動ローテーションをキャンセルする

シークレットの[自動ローテーション](#)を設定し、ローテーションを停止する場合は、ローテーションをキャンセルできます。

自動ローテーションをキャンセルするには

1. Secrets Manager のコンソール (<https://console.aws.amazon.com/secretsmanager/>) を開きます。
2. シークレットを選択します。
3. シークレットの詳細ページのローテーション設定 で、ローテーションの編集 を選択します。
4. 「ローテーション設定の編集」ダイアログボックスで、「自動ローテーション」をオフにし、「保存」を選択します。

Secrets Manager はローテーション設定情報を保持するため、ローテーションを再度有効にする場合は、後で使用できます。

AWS Secrets Manager 他の AWS のサービスによって管理されるシークレット

多くの AWS サービスは、にシークレットを保存して使用します AWS Secrets Manager。これらのシークレットは、マネージドシークレットである場合もあります。これは、シークレットを作成したサービスが、シークレットの管理をサポートしていることを意味します。例えば、いくつかのマネージドシークレットには [マネージドローテーション](#)が含まれているため、ローテーションを自分で設定する必要はありません。また、管理サービスでは、復旧期間なしでシークレットを更新または削除することが制限されています。管理サービスはシークレットに依存するため、この制限は機能停止を防ぐのに役立ちます。

マネージドシークレットは、識別しやすいように管理サービス ID を含む命名規則を使用しています。

```
Secret name: ServiceID!MySecret
Secret ARN : arn:aws:us-east-1:ServiceID!MySecret-a1b2c3
```

シークレットを管理するサービスの ID

- appflow – [the section called “Amazon AppFlow”](#)
- databrew – [the section called “AWS Glue DataBrew”](#)
- datasync – [the section called “AWS DataSync”](#)
- directconnect – [the section called “AWS 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”](#)
- rds – [the section called “Amazon RDS”](#)
- redshift – [the section called “Amazon Redshift”](#)
- sqlworkbench – [the section called “Amazon Redshift クエリエディタ v2”](#)

他の AWS のサービスによって管理されているシークレットを検索するには、[「マネージドシークレットの検索」](#)を参照してください。

シークレットを使用するサービスの完全なリストについては、「」を参照してください[シークレットを使用するサービス](#)。

AWSAWS Secrets Manager シークレットを使用する サービス

AWS Secrets Manager と統合する サービス :

- [AWS App Runner の使用方法 AWS Secrets Manager](#)
- [AWS App2Container が を使用する方法 AWS Secrets Manager](#)
- [AWS AppConfig の使用方法 AWS Secrets Manager](#)
- [Amazon が AppFlow を使用する方法 AWS Secrets Manager](#)
- [AWS AppSync の使用方法 AWS Secrets Manager](#)
- [Amazon Athena の AWS Secrets Manager使用方法](#)
- [Amazon Aurora での の使用方法 AWS Secrets Manager](#)
- [AWS CodeBuild の使用方法 AWS Secrets Manager](#)
- [AWS DataSync の使用方法 AWS Secrets Manager](#)
- [Amazon が DataZone を使用する方法 AWS Secrets Manager](#)
- [AWS Direct Connect の使用方法 AWS Secrets Manager](#)
- [AWS Directory Service の使用方法 AWS Secrets Manager](#)
- [Amazon DocumentDB \(MongoDB 互換性\)の AWS Secrets Manager使用方法](#)
- [AWS Elastic Beanstalk の使用方法 AWS Secrets Manager](#)
- [Amazon Elastic Container Registry が を使用する方法 AWS Secrets Manager](#)
- [Amazon Elastic Container Service](#)
- [Amazon が ElastiCache を使用する方法 AWS Secrets Manager](#)
- [AWS Elemental Live の使用方法 AWS Secrets Manager](#)
- [AWS Elemental MediaConnect の使用方法 AWS Secrets Manager](#)
- [AWS Elemental MediaConvert の使用方法 AWS Secrets Manager](#)
- [AWS Elemental MediaLive の使用方法 AWS Secrets Manager](#)
- [AWS Elemental MediaPackage の使用方法 AWS Secrets Manager](#)
- [AWS Elemental MediaTailor の使用方法 AWS Secrets Manager](#)
- [Amazon EMR が Secrets Manager を使用する方法](#)
- [Amazon が EventBridge を使用する方法 AWS Secrets Manager](#)
- [Amazon FSx が AWS Secrets Manager シークレットを使用する方法](#)

- [AWS Glue DataBrew の使用方法 AWS Secrets Manager](#)
- [AWS Glue Studio が を使用する方法 AWS Secrets Manager](#)
- [AWS IoT SiteWise の使用方法 AWS Secrets Manager](#)
- [Amazon Kendra が を使用する方法 AWS Secrets Manager](#)
- [Amazon Kinesis Video Streams が を使用する方法 AWS Secrets Manager](#)
- [AWS Launch Wizard の使用方法 AWS Secrets Manager](#)
- [Amazon Lookout for Metrics の使用方法](#)
- [Amazon Managed Grafana が を使用する方法 AWS Secrets Manager](#)
- [AWS Managed Services の使用方法 AWS Secrets Manager](#)
- [Amazon Managed Streaming for Apache Kafkaの AWS Secrets Manager使用方法](#)
- [Amazon Managed Workflows for Apache Airflow が を使用する方法 AWS Secrets Manager](#)
- [AWS Marketplace](#)
- [AWS Migration Hub の使用方法 AWS Secrets Manager](#)
- [が Secrets Manager AWS Panorama を使用する方法](#)
- [AWS ParallelCluster の使用方法 AWS Secrets Manager](#)
- [Amazon Q が Secrets Manager を使用する方法](#)
- [AWS OpsWorks for Chef Automate の使用方法 AWS Secrets Manager](#)
- [Amazon が QuickSight を使用する方法 AWS Secrets Manager](#)
- [Amazon RDS](#)
- [Amazon Redshift が を使用する方法 AWS Secrets Manager](#)
- [Amazon Redshift クエリエディタ v2](#)
- [Amazon が SageMaker を使用する方法 AWS Secrets Manager](#)
- [AWS Schema Conversion Tool の使用方法 AWS Secrets Manager](#)
- [AWS Toolkit for JetBrains の使用方法 AWS Secrets Manager](#)
- [が AWS Secrets Manager シークレット AWS Transfer Family を使用する方法](#)
- [AWS Wickr がシーク AWS Secrets Manager レットを使用する方法](#)

AWS App Runner の使用方法 AWS Secrets Manager

AWS App Runner は、ソースコードまたはコンテナイメージから AWS クラウド内のスケラブルで安全なウェブアプリケーションに直接デプロイするための、高速でシンプルで費用対効果の高い方

法を提供する AWS サービスです。新しいテクノロジーを学習したり、使用するコンピューティングサービスを決定したり、AWS リソースをプロビジョニングして設定する方法を知っている必要はありません。

App Runner を使用すると、サービスを作成、もしくはその設定を更新した際に、シークレットや設定内容をサービスの環境変数として参照できるようになります。詳細については、「AWS App Runner 開発者ガイド」の「[Referencing environment variables](#)」(環境変数の参照)と「[Managing environment variables](#)」(環境変数の管理)を参照してください。

AWS App2Container が 使用方法 AWS Secrets Manager

AWS App2Container (A2C) は、オンプレミスのデータセンターまたは仮想マシンで実行されるアプリケーションをリフトアンドシフトして、Amazon ECS、Amazon EKS、またはによって管理されるコンテナで実行できるようにするコマンドラインツールです AWS App Runner。

App2Container は Secrets Manager を使用して、リモートコマンドを実行するためにワーカーマシンをアプリケーションサーバーに接続するための資格を管理します。詳細については、[AWS App2Container](#)の「[App2 コンテナのシークレットを管理する](#) AWS App2Container」を参照してください。

AWS AppConfig の使用方法 AWS Secrets Manager

AWS AppConfig は、アプリケーション設定の作成、管理、および迅速なデプロイ AWS Systems Manager に使用できる の機能です。設定には、Secrets Manager に保存されている認証情報データまたはその他の機密情報を含めることができます。自由形式の設定プロファイルを作成する場合、Secrets Manager を設定データのソースとして選択できます。詳細については、「AWS AppConfig ユーザーガイド」の「[自由形式の設定プロファイルの作成](#)」を参照してください。自動ローテーションが有効になっているシークレットを が AWS AppConfig 処理する方法の詳細については、AWS AppConfig 「ユーザーガイド」の「[Secrets Manager のキーローテーション](#)」を参照してください。

Amazon が AppFlow を使用方法 AWS Secrets Manager

Amazon AppFlow は、Salesforce などの Software as a Service (SaaS) アプリケーションと AWS 、 Amazon Simple Storage Service (Amazon S3) や Amazon Redshift などのサービス間でデータを安全に交換できるフルマネージド統合サービスです。

Amazon では AppFlow、SaaS アプリケーションをソースまたは送信先として設定するときに、接続を作成します。これには、認証トークン、ユーザー名、およびパスワードなど、SaaS アプリケーションへの接続に必要な情報が含まれます。Amazon は、プレフィックスが付いた Secrets Manager [マネージドシークレット](#) に接続データ AppFlow を保存します appflow。シークレットの保存コストは、Amazon の料金に含まれています AppFlow。詳細については、「[Amazon ユーザーガイド](#)」の「[Amazon でのデータ保護 AppFlow](#) AppFlow」を参照してください。

AWS AppSync の使用方法 AWS Secrets Manager

AWS AppSync は、アプリケーションデベロッパーが Amazon DynamoDB、HTTP APIs などの複数のソースからのデータを組み合わせるための AWS Lambda、堅牢でスケーラブルな GraphQL インターフェイスを提供します。

AWS AppSync は CLI コマンド [rds execute-statement](#) を使用して、シークレット内の認証情報を使用して Amazon RDS に接続します。詳細については、「AWS AppSync デベロッパーガイド」の「[Tutorial: Aurora Serverless](#)」(チュートリアル: Aurora Serverless) を参照してください。

Amazon Athena の AWS Secrets Manager 使用方法

Amazon Athena は、標準的な SQL を使用して Amazon Simple Storage Service (Amazon S3) 内のデータを直接分析することを容易にする、インタラクティブなクエリサービスです。

Amazon Athena データソースコネクタは、Secrets Manager シークレットと Athena フェデレーションクエリ機能を使用して、データをクエリできます。詳細については、「Amazon Athena ユーザーガイド」の「[Amazon Athena 横串検索の使用](#)」を参照してください。

Amazon Aurora での の使用方法 AWS Secrets Manager

Amazon Aurora はフルマネージド型のリレーショナルデータベースエンジンで、MySQL および PostgreSQL と互換性があります。

Aurora のマスターユーザー認証情報を管理するために、Aurora は [マネージドシークレット](#) を作成できます。そのシークレットの料金が請求されます。Aurora は、[これらの認証情報のローテーションも管理](#) します。詳細については、「Amazon Aurora ユーザーガイド」の「[Amazon Aurora および AWS Secrets Manager でのパスワード管理](#)」を参照してください。

その他の Aurora 認証情報については、「」を参照してください [the section called “データベースシークレットを作成する”](#)。

Amazon RDS の Data API を呼び出すと、Secrets Manager を使用してデータベースの認証情報を渡すことができます。詳細については、「Amazon Aurora ユーザーガイド」の「[Aurora Serverless のデータ API の使用](#)」を参照してください。

Amazon RDS クエリエディタを使用してデータベースに接続する場合、データベースの認証情報を Secrets Manager に保存できます。詳細については、「Amazon RDS ユーザーガイド」の「[クエリエディタの使用](#)」を参照してください。

AWS CodeBuild の使用方法 AWS Secrets Manager

AWS CodeBuild は、クラウドのフルマネージドビルドサービスです。はソースコードを CodeBuild コンパイルし、ユニットテストを実行し、すぐにデプロイできるアーティファクトを生成します。

Secrets Manager を使用して、プライベートレジストリの認証情報を保存できます。詳細については、「ユーザーガイド」の「[AWS Secrets Manager のサンプルを含むプライベートレジストリ CodeBuild](#)」を参照してください。AWS CodeBuild

AWS DataSync の使用方法 AWS Secrets Manager

AWS DataSync は、ストレージシステムとサービス間のデータの移動を簡素化、自動化、高速化するオンラインデータ転送サービスです。DataSync Discovery は、への移行を加速するのに役立ちます AWS。

オンプレミスストレージシステムに関する情報を収集するために、DataSync Discovery はストレージシステムの管理インターフェイスの認証情報を使用します。は、プレフィックスが付いた Secrets Manager [マネージドシークレット](#) にそれらの認証情報 DataSync を保存します datasync。そのシークレットの料金が請求されます。詳細については、「ユーザーガイド」の [DataSync 「Discovery へのオンプレミスストレージシステムの追加AWS DataSync](#)」を参照してください。

Amazon が DataZone を使用する方法 AWS Secrets Manager

Amazon DataZone は、データのカタログ化、検出、管理、共有、分析を可能にするデータ管理サービスです。AWS Glue クローラー ジョブを使用してクロールされた Amazon Redshift クラスターのテーブルとビューのデータアセットを使用できます。Amazon Redshift に接続するには、Secrets Manager シークレットで Amazon DataZone 認証情報を指定します。詳細については、「[Amazon ユーザーガイド](#)」の「[新しい AWS Glue 接続を使用して Amazon Redshift データベースのデータソースを作成する](#)」を参照してください。DataZone

AWS Direct Connect の使用方法 AWS Secrets Manager

AWS Direct Connect は、標準のイーサネット光ファイバケーブルを介して内部ネットワークを AWS Direct Connect 口ケーションにリンクします。この接続により、パブリックの AWS のサービスへの仮想インターフェイスを直接作成できます。

AWS Direct Connect は、接続関連付けキー名と接続関連付けキーペア (CKN/CAK ペア) を、プレフィックスが付いた [マネージドシークレット](#) に保存します `directconnect`。シークレットのコストは、の料金に含まれています AWS Direct Connect。シークレットを更新するには、Secrets Manager AWS Direct Connect ではなく `directconnect` を使用する必要があります。詳細については、「AWS Direct Connect ユーザーガイド」の「[Associate a MACsec CKN/CAK with a LAG](#)」(MACsec CKN/CAK を LAG に関連付ける) を参照してください。

AWS Directory Service の使用方法 AWS Secrets Manager

AWS Directory Service には、他の AWS のサービスで Microsoft Active Directory (AD) を使用する複数の方法が用意されています。認証情報にシークレットを使用すると、Amazon EC2 インスタンスをディレクトリに参加させることができます。詳細については、「AWS Direct Connect ユーザーガイド」で以下を参照してください。

- [Linux EC2 インスタンスを AWS Managed Microsoft AD ディレクトリにシームレスに結合する](#)
- [Linux EC2 インスタンスを AD Connector ディレクトリにシームレスに結合する](#)
- [Linux EC2 インスタンスを Simple AD ディレクトリにシームレスに結合する](#)

Amazon DocumentDB (MongoDB 互換性) の AWS Secrets Manager 使用方法

Amazon DocumentDB の場合、ユーザーはクラスターへの認証にパスワードを接続します。AWS Secrets Manager を使用すると、コード内のハードコードされた認証情報 (パスワードを含む) を Secrets Manager への API コールで置き換えて、プログラムでシークレットを取得することができます。詳細については、「Amazon DocumentDB デベロッパーガイド」で、[the section called “データベースシークレットを作成する”](#) および「[Managing Amazon DocumentDB Users](#)」(Amazon DocumentDB ユーザーの管理) を参照してください。

AWS Elastic Beanstalk の使用方法 AWS Secrets Manager

を使用すると AWS Elastic Beanstalk、アプリケーションを実行するインフラストラクチャについて知ることなく、AWS クラウドでアプリケーションをすばやくデプロイおよび管理できます。Elastic Beanstalk は、Dockerfile に記述されたイメージを構築することによって、またはリモートの Docker イメージを取り込むことによって、Docker 環境を起動できます。プライベートリポジトリをホストするオンラインレジストリで認証するために、Elastic Beanstalk は Secrets Manager シークレットを使用します。詳細については、「AWS Elastic Beanstalk デベロッパーガイド」の「[Docker の設定](#)」を参照してください。

Amazon Elastic Container Registry が を使用する方法 AWS Secrets Manager

Amazon Elastic Container Registry (Amazon ECR) は、安全でスケーラブル、信頼性の高い AWS マネージドコンテナイメージレジストリサービスです。Docker CLI またはその他のクライアントを使用して、リポジトリからイメージをプッシュおよびプルできます。Amazon ECR プライベートレジストリにキャッシュするイメージを含むアップストリームレジストリごとに、プルスルーキャッシュルールを作成する必要があります。認証が必要なアップストリームレジストリでは、認証情報を Secrets Manager シークレットに保存する必要があります。Secrets Manager シークレットは、Amazon ECR コンソールまたは Secrets Manager コンソールのいずれかで作成できます。詳細については、「Amazon ECR [ユーザーガイド](#)」の「[プルスルーキャッシュルールの作成](#)」を参照してください。

Amazon Elastic Container Service

Amazon Elastic Container Service (Amazon ECS) は、コンテナ化されたアプリケーションを簡単にデプロイ、管理、スケーリングできる、完全マネージド型のコンテナオーケストレーションサービスです。Secrets Manager のシークレットを参照することによって、機密データをコンテナに挿入できます。詳細については、「Amazon Elastic Container Service デベロッパーガイド」の次のページを参照してください:

- [チュートリアル: Secrets Manager のシークレットを使用して機密データを指定する](#)
- [アプリケーションを介してプログラムによりシークレットを取得する](#)
- [環境変数経由でシークレットを取得する](#)
- [ログ記録設定のシークレットを取得する](#)

Amazon ECS は、コンテナ用に FSx for Windows File Server ボリュームをサポートします。Amazon ECS は、Secrets Manager のシークレットに保存されている認証情報を使用して、Active Directory にドメイン参加したり、FSx for Windows File Server ファイルシステムをアタッチしたりします。詳細については、「Amazon Elastic Container Service デベロッパーガイド」の「[チュートリアル: Amazon ECS で FSx for Windows File Server ファイルシステムを使用](#)」と「[FSx for Windows File Server ボリューム](#)」を参照してください。

レジストリ認証情報で Secrets Manager シークレットを使用することで、認証 AWS が必要な以外のプライベートレジストリでコンテナイメージを参照できます。詳細については、「Amazon Elastic Container Service デベロッパーガイド」の「[タスクのプライベートレジストリの認証](#)」を参照してください。

Amazon ECS Service Connect を使用する場合、Amazon ECS は Secrets Manager [マネージドシークレット](#)を使用して AWS Private Certificate Authority TLS 証明書を保存します。シークレットの保存コストは、Amazon ECS の料金に含まれています。シークレットを更新するには、Secrets Manager ではなく Amazon ECS を使用する必要があります。詳細については、「Amazon Elastic Container Service デベロッパーガイド」の「[TLS with Service Connect](#)」を参照してください。

Amazon が ElastiCache を使用する方法 AWS Secrets Manager

では、ロールベースのアクセスコントロール (RBAC) と呼ばれる機能を使用してクラスターを保護 ElastiCache できます。これらの認証情報は Secrets Manager に保存します。Secrets Manager は、このタイプのシークレットのために、[ローテーションプレート](#)を提供します。詳細については、「Amazon ユーザーガイド」の「[ユーザーのパスワードの自動ローテーション](#)」を参照してください。ElastiCache

AWS Elemental Live の使用方法 AWS Secrets Manager

AWS Elemental Live は、ブロードキャストおよびストリーミング配信用のライブ出力を作成できるリアルタイムビデオサービスです。

AWS Elemental Live はシークレット ARN を使用して、Secrets Manager から暗号化キーを含むシークレットを取得します。Elemental Live は、暗号化キーを使用してビデオを暗号化/復号化します。詳細については、「Elemental Live [ユーザーガイド](#)」の「[から AWS Elemental Live への配信がランタイムに MediaConnect 機能する方法](#)」を参照してください。

AWS Elemental MediaConnect の使用方法 AWS Secrets Manager

AWS Elemental MediaConnect は、ブロードキャスターやその他のプレミアムビデオプロバイダーがライブビデオを確実に取り込み AWS クラウド、内外の複数の宛先に配信できるようにするサービスです AWS クラウド。

静的キー暗号化を使用してソース、出力、およびエンタイトルメントを保護し、暗号化キーを AWS Secrets Manager に蓄積します。詳細については、「[ユーザーガイド](#)」の「[での静的キー AWS Elemental MediaConnect](#)暗号化AWS Elemental MediaConnect」を参照してください。

AWS Elemental MediaConvert の使用方法 AWS Secrets Manager

AWS Elemental MediaConvert はファイルベースの動画処理サービスで、コンテンツ所有者やディストリビューターに、あらゆる規模のメディアライブラリでスケーラブルな動画処理を提供します。MediaConvert を使用して Kantar ウォーターマークをエンコードするには、Secrets Manager を使用して Kantar 認証情報を保存します。詳細については、「[AWS Elemental MediaConvert ユーザーガイド](#)」の[AWS Elemental MediaConvert 「出力でのオーディオウォーターマークに Kantar を使用する」](#)を参照してください。

AWS Elemental MediaLive の使用方法 AWS Secrets Manager

AWS Elemental MediaLive は、ブロードキャストおよびストリーミング配信用のライブ出力を作成できるリアルタイムビデオサービスです。組織が AWS Elemental MediaLive または AWS Elemental で AWS Elemental Link デバイスを使用している場合は MediaConnect、デバイスをデプロイしてデバイスを設定する必要があります。詳細については、「[ユーザーガイド](#)」の「[信頼されたエンティティ MediaLive としてを設定するMediaLive](#)」を参照してください。

AWS Elemental MediaPackage の使用方法 AWS Secrets Manager

AWS Elemental MediaPackage は、で実行される just-in-time ビデオパッケージ化および発信サービスです AWS クラウド。を使用すると MediaPackage、さまざまな再生デバイスやコンテンツ配信ネットワーク (CDNs) に、安全性、スケーラビリティ、信頼性に優れたビデオストリームを提供できます。詳細については、「[AWS Elemental MediaPackage ユーザーガイド](#)」の「[CDN 認証のための Secrets Manager アクセス](#)」を参照してください。

AWS Elemental MediaTailor の使用方法 AWS Secrets Manager

AWS Elemental MediaTailor は、 で実行されるスケーラブルな広告挿入およびチャネルアセンブリサービスです AWS クラウド。

MediaTailor は、ソースロケーションへの Secrets Manager アクセストークン認証をサポートします。Secrets Manager アクセストークン認証 MediaTailor では、 は Secrets Manager シークレットを使用してオリジンへのリクエストを認証します。詳細については、「ユーザーガイド」の [AWS Secrets Manager 「アクセストークン認証の設定AWS Elemental MediaTailor」](#) を参照してください。

Amazon EMR が Secrets Manager を使用する方法

Amazon EMR は、Apache Hadoop や Apache Spark などのビッグデータフレームワークを で簡単に実行 AWS して、膨大な量のデータを処理および分析するプラットフォームです。これらのフレームワークおよび関連するオープンソースプロジェクト (Apache Hive や Apache Pig など)、分析用のデータおよびビジネスインテリジェンスワークロードを処理できます。Amazon EMR を使用して、Amazon S3 や Amazon DynamoDB などの他の AWS データストアやデータベースとの間で大量のデータを変換および移動することもできます。

Amazon EC2 上で実行する Amazon EMR が Secrets Manager を使用する方法

Amazon EMR でクラスターを作成する場合、Secrets Manager のシークレットでアプリケーション設定データをクラスターに提供できます。詳細については、「Amazon EMR 管理ガイド」の「[機密性の高い設定データを Secrets Manager に保存する](#)」を参照してください。

さらに、EMR ノートブックを作成するときに、Secrets Manager を使用してプライベート Git ベースのレジストリ認証情報を保存できます。詳細については、「Amazon EMR 管理ガイド」の「[Add a Git-based Repository to Amazon EMR](#)」(Git ベースのリポジトリを Amazon EMR に追加する) を参照してください。

EMR Serverless が Secrets Manager を使用する方法

EMR Serverless は、分析アプリケーションの運用を簡素化するサーバーレスのランタイム環境を提供するため、クラスターを設定、最適化、保護、および運用する必要がありません。

データを に保存し AWS Secrets Manager、EMR Serverless 設定でシークレット ID を使用できます。この方法を使用すると、機密性の高い設定データをプレーンテキストで渡して外部 API に公開する必要がなくなります。

詳細については、「Amazon EMR Serverless ユーザーガイド」の「[EMR Serverless でのデータ保護に Secrets Manager を使用する](#)」を参照してください。

Amazon が EventBridge を使用する方法 AWS Secrets Manager

Amazon EventBridge は、アプリケーションをさまざまなソースのデータに接続するために使用できるサーバーレスイベントバスサービスです。

Amazon EventBridge API 送信先を作成すると、はプレフィックスが付いた Secrets Manager [マネージドシークレット](#)にその接続 EventBridge を保存します events。シークレットを保存する費用は、API 送信先を使用する料金に含まれています。シークレットを更新するには、Secrets Manager ではなくを使用する必要があります EventBridge。詳細については、「Amazon ユーザーガイド」の「[API 送信先](#)」を参照してください。 EventBridge

Amazon FSx が AWS Secrets Manager シークレットを使用する方法

Amazon FSx for Windows File Server は、フルマネージドの Microsoft Windows ファイルサーバーで、完全にネイティブの Windows ファイルシステムでバックアップされています。ファイル共有を作成または管理するときは、AWS Secrets Manager シークレットから認証情報を渡すことができます。詳細については、「Amazon FSx for Windows File Server ユーザーガイド」の「[ファイル共有](#)」および「[ファイル共有設定を Amazon FSx に移行する](#)」を参照してください。

AWS Glue DataBrew の使用方法 AWS Secrets Manager

AWS Glue DataBrew は、コードを記述せずにデータをクリーニングおよび正規化するために使用できるビジュアルデータ準備ツールです。では DataBrew、一連のデータ変換ステップを recipe と呼びます。AWS Glue DataBrew は [DETERMINISTIC_DECRYPT](#)、Secrets Manager シークレットに保存されている暗号化キーを使用するデータセット内の個人を特定できる情報 (PII) に対して変換を実行するための、[DETERMINISTIC_ENCRYPT](#) および [CRYPTOGRAPHIC_HASH](#) recipe ステップを提供します。DataBrew デフォルトのシークレットを使用して暗号化キーを保存する場合、DataBrew はプレフィックスを持つ [マネージドシークレット](#) を作成します databrew。シークレットの保存コスト

トは、の使用料金に含まれています DataBrew。暗号化キーを保存する新しいシークレットを作成すると、はプレフィックスを持つシークレット DataBrew を作成します AwsGlueDataBrew。そのシークレットの料金が請求されます。

AWS Glue Studio が を使用する方法 AWS Secrets Manager

AWS Glue Studio は、で抽出、変換、ロード (ETL) ジョブの作成、実行、モニタリングを容易にするグラフィカルインターフェイスです AWS Glue。AWS Glue Studio で Elasticsearch Spark Connector を設定することで、を抽出、変換、ロード (ETL) ジョブのデータストア OpenSearch として使用できます。OpenSearch クラスターに接続するには、Secrets Manager でシークレットを使用できます。詳細については、「[AWS Glue デベロッパーガイド](#)」の「[チュートリアル: Elasticsearch 用の AWS Glue コネクタの使用](#)」を参照してください。

AWS IoT SiteWise の使用方法 AWS Secrets Manager

AWS IoT SiteWise は、産業機器からデータを大規模に収集、モデル化、分析、視覚化できるマネージドサービスです。AWS IoT SiteWise コンソールを使用してゲートウェイを作成できます。そしてデータソースは、ゲートウェイに接続されたローカルサーバーまたは産業機器です。ソースで認証が必要な場合は、シークレットを使用して認証します。詳細については、「[AWS IoT SiteWise ユーザーガイド](#)」の「[Configuring data source authentication](#)」(データソース認証を設定する)を参照してください。

Amazon Kendra が を使用する方法 AWS Secrets Manager

Amazon Kendra は、ユーザーが自然言語を使用して高度な検索アルゴリズムデータを検索できるようにする、高精度のインテリジェント検索サービスです。

データベースの認証情報を含むシークレットを指定して、データベースに保存されているドキュメントにインデックスを作成することができます。詳細については、「[Amazon Kendra ユーザーガイド](#)」の「[Using a database data source](#)」(データベースデータソースの使用)を参照してください。

Amazon Kinesis Video Streams が を使用する方法 AWS Secrets Manager

Amazon Kinesis Video Streams を使用して、顧客構内の IP カメラに接続し、カメラからの動画をローカルで記録および保存し、動画をクラウドにストリーミングして長期保存、再生、分析処理

を行うことができます。IP カメラからメディアを録画してアップロードするには、Kinesis Video Streams Edge Agent を AWS IoT Greengrass にデプロイします。カメラにストリーミングされるメディアファイルにアクセスするために必要な認証情報は、Secrets Manager シークレットに保存します。詳細については、「Amazon Kinesis Video Streams デベロッパーガイド」の「[Amazon Kinesis Video Streams Edge Agent を AWS IoT Greengrass にデプロイする](#)」を参照してください。

AWS Launch Wizard の使用方法 AWS Secrets Manager

AWS Launch Wizard for Active Directory は、AWS クラウドアプリケーションのベストプラクティスを適用して、新しい Active Directory インフラストラクチャをセットアップしたり、ドメインコントローラーを AWS クラウドまたはオンプレミスの既存のインフラストラクチャに追加したりする手順を説明するサービスです。

AWS Launch Wizard では、ドメインコントローラーを Active Directory に参加させるために、ドメイン管理者の認証情報を Secrets Manager に追加する必要があります。詳細については、「[ユーザーガイド](#)」の「[の Active Directory のセットアップ AWS Launch Wizard](#)」を参照してください。

Amazon Lookout for Metrics の使用方法

Amazon Lookout for Metrics は、データ内の異常を検出し、その根本原因を特定し、迅速な対応を可能にします。Amazon Redshift または Amazon RDS を Lookout for Metrics データソースとして使用できます。データソースを設定するには、データベースパスワードを含むシークレットを使用します。詳細については、「Amazon Lookout for Metrics Developer Guide」(Amazon Lookout for Metrics デベロッパーガイド)の「[Using Amazon RDS with Lookout for Metrics](#)」(Amazon RDS での Lookout for Metrics の使用)、および「[Using Amazon Redshift with Lookout for Metrics](#)」(Amazon Redshift での Lookout for Metrics の使用)を参照してください。

Amazon Managed Grafana が を使用する方法 AWS Secrets Manager

Amazon Managed Grafana は、フルマネージド型で安全なデータ可視化サービスであり、複数のソースからの運用メトリクス、ログ、トレースを即座に照会、関連付け、視覚化できます。Amazon Redshift をデータソースとして使用する場合は、AWS Secrets Manager シークレットを使用して Amazon Redshift 認証情報を指定できます。詳細については、「Amazon Managed Grafana ガイド」の「[Amazon Redshift の設定](#)」を参照してください。

AWS Managed Services の使用方法 AWS Secrets Manager

AWS Managed Services は、AWS インフラストラクチャを継続的に管理するエンタープライズサービスです。AMS セルフサービスプロビジョニング (SSP) モードは、AMS マネージドアカウントのネイティブ AWS サービスおよび API 機能へのフルアクセスを提供します。AMS で Secrets Manager に対するアクセスをリクエストする方法については、「AMS アドバンスドユーザーガイド」の「[AWS Secrets Manager \(AMS セルフサービスプロビジョニング\)](#)」を参照してください。

Amazon Managed Streaming for Apache Kafkaの AWS Secrets Manager使用方法

Amazon Managed Streaming for Apache Kafka (Amazon MSK) は、Apache Kafka を使ってストリーミングデータを処理するアプリケーションの構築および実行を可能にする、フルマネージドサービスです。AWS Secrets Managerを使用して保存および保護されているユーザーネームとパスワードを使用して、Amazon MSK クラスターへのアクセスを制御できます。詳細については、Amazon Managed Streaming for Apache Kafka デベロッパーガイドの [Username and password authentication with AWS Secrets Manager](#) を参照してください。

Amazon Managed Workflows for Apache Airflow が 使用方法 AWS Secrets Manager

Amazon Managed Workflows for Apache Airflow (MWAA) は、[Apache Airflow](#) 用のマネージドオーケストレーションサービスであり、クラウドでの大規模な end-to-end データパイプラインのセットアップと運用を容易にします。

Apache Airflow 接続は、Secrets Manager シークレットを使用して設定できます。詳細については、「Amazon Managed Workflows for [Apache Airflow ユーザーガイド](#)」の「[Secrets Manager シークレットを使用した Apache Airflow 接続の設定](#)」および「[Apache Airflow 変数 AWS Secrets Manager のでのシークレットキーの使用](#)」を参照してください。

AWS Marketplace

AWS Marketplace クイック起動を使用すると、はライセンスキーとともにソフトウェアを AWS Marketplace 配布します。はライセンスキーを Secrets Manager [マネージドシークレット](#) としてアカウントに AWS Marketplace 保存します。シークレットの保存コストは、の料金に含まれていません AWS Marketplace。シークレットを更新するには、Secrets Manager AWS Marketplace ではなく

を使用する必要があります。詳細については、「AWS Marketplace 販売者ガイド」の「[Configure Quick Launch](#)」を参照してください。

AWS Migration Hub の使用方法 AWS Secrets Manager

AWS Migration Hub は、複数の AWS ツールやパートナーソリューション間で移行タスクを追跡するための単一の場所を提供します。

AWS Migration Hub Orchestrator は、サーバーとエンタープライズアプリケーションへの移行を簡素化および自動化します。AWS Migration Hub Orchestrator は、ソースサーバーへの接続情報にシークレットを使用します。詳細については、「AWS Migration Hub Orchestrator User Guide」(Orchestrator ユーザーガイド)で、以下を参照してください。

- [SAP NetWeaver アプリケーションをに移行する AWS](#)
- [Amazon EC2 上のアプリケーションのリホスト](#)

Migration Hub 戦略の推奨事項は、アプリケーションの実行可能なトランスフォーメーションパスに関する移行およびモダナイゼーション戦略の推奨事項を提供します。戦略の推奨事項で、接続情報のシークレットを使用して SQL Server データベースを分析できます。詳細については、「[戦略推奨データベース分析](#)」を参照してください。

が Secrets Manager AWS Panorama を使用する方法

AWS Panorama は、コンピュータビジョンをオンプレミスのカメラネットワークにもたらすサービスです。を使用してアプライアンスを登録 AWS Panorama し、そのソフトウェアを更新し、そのアプライアンスにアプリケーションをデプロイします。ビデオストリームをアプリケーションのデータソースとして登録するときに、ストリームがパスワードで保護されている場合、はビデオストリームの認証情報を Secrets Manager シークレットに AWS Panorama 保存します。詳細については、「AWS Panorama デベロッパーガイド」の「[AWS Panorama でのカメラストリームの管理](#)」を参照してください。

AWS ParallelCluster の使用方法 AWS Secrets Manager

AWS ParallelCluster は、AWS クラウドでハイパフォーマンスコンピューティング (HPC) クラスターをデプロイおよび管理するために使用できるオープンソースのクラスター管理ツールです。AWS Managed Microsoft AD (Active Directory) と統合された AWS ParallelCluster を含む複数のユーザー環境を作成できます。AWS ParallelCluster は、Secrets Manager シークレットを使用して

Active Directory へのログインを検証します。詳細については、「AWS ParallelCluster ユーザーガイド」の「[Active Directory の統合](#)」を参照してください。

Amazon Q が Secrets Manager を使用する方法

Amazon Q を認証してデータソースにアクセスするには、Secrets Manager シークレットを使用してデータソースのアクセス認証情報を Amazon Q に提供します。コンソールを使用する場合は、新しいシークレットを作成するか、既存のシークレットを使用するかを選択できます。詳細については、「Amazon Q デベロッパーガイド」の「[概念 - 認証](#)」を参照してください。

AWS OpsWorks for Chef Automate の使用方法 AWS Secrets Manager

AWS OpsWorks は、Puppet または Chef を使用してクラウドエンタープライズでアプリケーションを設定および運用するのに役立つ設定管理サービスです。

で新しいサーバーを作成すると AWS OpsWorks CM、OpsWorks CM はプレフィックスが付いた Secrets Manager [マネージドシークレット](#) にサーバーの情報を保存します opsworks-cm。シークレットのコストは、の料金に含まれています AWS OpsWorks。詳細については、「AWS OpsWorks ユーザーガイド」の「[Integration with AWS Secrets Manager](#)」(との統合)を参照してください。

Amazon が QuickSight を使用する方法 AWS Secrets Manager

Amazon QuickSight は、分析、データの可視化、レポート作成に使用できるクラウド規模のビジネスインテリジェンス (BI) サービスです。Amazon では、さまざまなデータソースを使用できます QuickSight。データベース認証情報を Secrets Manager シークレットに保存する場合、Amazon QuickSight はそれらのシークレットを使用してデータベースに接続できます。詳細については、「[Amazon ユーザーガイド](#)」の「[Amazon でのデータベース認証情報の代わりに AWS Secrets Manager シークレット QuickSight を使用する](#)」を参照してください。 QuickSight

Amazon RDS

Amazon Relational Database Service (Amazon RDS) は、AWS クラウドでのリレーショナルデータベースのセットアップ、運用、スケーリングを容易にするウェブサービスです。

Aurora を含む Amazon Relational Database Service (Amazon RDS) のマスターユーザー認証情報を管理するために、Amazon RDS は [マネージドシークレット](#) を作成できます。そのシークレットの料

金が請求されます。Amazon RDS は、これらの認証情報の[ローテーションも管理](#)します。詳細については、「Amazon RDS ユーザーガイド」の「[Amazon RDS および AWS Secrets Manager でのパスワード管理](#)」を参照してください。

Amazon RDS の他の認証情報については「[the section called “データベースシークレットを作成する”](#)」を参照してください。

Amazon RDS クエリエディタを使用してデータベースに接続する場合、データベースの認証情報を Secrets Manager に保存できます。詳細については、「Amazon RDS ユーザーガイド」の「[クエリエディタの使用](#)」を参照してください。

Amazon Redshift が を使用する方法 AWS Secrets Manager

Amazon Redshift は、クラウド内でのフルマネージド型、ペタバイト規模のデータウェアハウスサービスです。

Amazon Redshift の管理者認証情報を管理するために、Amazon Redshift はマネージドシークレットを作成できます。そのシークレットの料金が請求されます。Amazon Redshift は、[これらの認証情報のローテーションも管理](#)します。詳細については、「Amazon Redshift 管理ガイド」の「[AWS Secrets Manager を使用して Amazon Redshift 管理者パスワードを管理する](#)」を参照してください。

その他の Amazon Redshift 認証情報については、「[the section called “データベースシークレットを作成する”](#)」を参照してください。

Amazon Redshift Data API を呼び出すと、Secrets Manager のシークレットを使用してクラスターの認証情報を渡すことができます。詳細については、「[Using the Amazon Redshift Data API](#)」(Amazon Redshift Data API の使用) を参照してください。

Amazon Redshift クエリエディタを使用してデータベースに接続すると、Amazon Redshift は、プレフィックス `redshiftqueryeditor` の Secrets Manager シークレットに認証情報を保存できます。そのシークレットの料金が請求されます。詳細については、「Amazon Redshift 管理ガイド」の「[クエリエディタを使用してデータベースのクエリを実行する](#)」を参照してください。

クエリエディタ v2 については、「[the section called “Amazon Redshift クエリエディタ v2”](#)」を参照してください。

Amazon Redshift クエリエディタ v2

Amazon Redshift クエリエディタ v2 は、Amazon Redshift データウェアハウスでクエリを作成および実行するために使用できるウェブベースの SQL クライアントアプリケーションで

す。Amazon Redshift クエリエディタ v2 を使用してデータベースに接続すると、Amazon Redshift はプレフィックスが付いた Secrets Manager [マネージドシークレット](#) に認証情報を保存できます。sqlworkbench。シークレットを保存する費用は、Amazon Redshift の使用料金に含まれていません。シークレットを更新するには、Secrets Manager ではなく Amazon Redshift を使用する必要があります。詳細については、「Amazon Redshift 管理ガイド」の「[クエリエディタ v2 の使用](#)」を参照してください。

以前のクエリエディタについては、「[the section called “Amazon Redshift”](#)」を参照してください。

Amazon が SageMaker を使用する方法 AWS Secrets Manager

SageMaker はフルマネージド型の機械学習サービスです。を使用すると SageMaker、データサイエンティストとデベロッパーは、機械学習モデルを迅速かつ簡単に構築してトレーニングし、本番環境に対応したホスト環境に直接デプロイできます。統合された Jupyter オーサリングノートブックインスタンスから、調査および分析用のデータソースに簡単にアクセスできるため、サーバーを管理する必要がありません。

Jupyter Notebook インスタンスを Git リポジトリに関連付けると、ノートブックインスタンスを停止または削除しても持続するソース管理環境にノートブックを保存できます。Secrets Manager を使用して、プライベートリポジトリの認証情報を管理できます。詳細については、「[Amazon デベロッパーガイド](#)」の「[Git リポジトリを Amazon SageMaker ノートブックインスタンスに関連付ける](#)」を参照してください。 SageMaker

Databricks からデータをインポートするために、Data Wrangler は JDBC URL を Secrets Manager に保存します。詳細については、「[Databricks \(JDBC\) からデータをインポートする](#)」を参照してください。

Snowflake からデータをインポートするために、Data Wrangler は Secrets Manager のシークレットに資格情報を格納します。詳細については、「[Snowflake からデータをインポートする](#)」を参照してください。

AWS Schema Conversion Tool の使用方法 AWS Secrets Manager

AWS Schema Conversion Tool (AWS SCT) を使用して、既存のデータベーススキーマをあるデータベースエンジンから別のデータベースエンジンに変換できます。リレーショナル OLTP スキーマやデータウェアハウススキーマを変換できます。変換されたスキーマは、Amazon Relational Database Service (Amazon RDS) MySQL、MariaDB、Oracle、SQL Server、PostgreSQL DB、Amazon Aurora DB クラスター、または Amazon Redshift クラスターに適しています。変換

されたスキーマは、Amazon Elastic Compute Cloud インスタンスでデータベースと共に使用する
か、Amazon Simple Storage Service バケットにデータとして保存することもできます。

データベーススキーマを変換するときは、に保存したデータベース認証情報 AWS SCT を使用でき
ます AWS Secrets Manager。詳細については、「ユーザーガイド [AWS Secrets Manager](#)」の [AWS
SCT「ユーザーインターフェイスでの の使用AWS Schema Conversion Tool](#)」を参照してくださ
い。

AWS Toolkit for JetBrains の使用方法 AWS Secrets Manager

AWS Toolkit for JetBrains は、の統合開発環境 (IDEs用のオープンソースプラグインです
JetBrains。このツールキットにより、開発者は AWSを使用するサーバーレスアプリケーションの
開発、デバッグ、およびデプロイが簡単になります。ツールキットを使用して Amazon Redshift
クラスターに接続する場合、Secrets Manager シークレット を使用して認証できます。詳細につ
いては、「AWS Toolkit for JetBrains ユーザーガイド」の「[Accessing Amazon Redshift clusters](#)」
(Amazon Redshift へのアクセス) を参照してください。

が AWS Secrets Manager シークレット AWS Transfer Family を使 用する方法

AWS Transfer Family は、AWS ストレージサービスとの間でファイルを転送できる安全な転送サー
ビスです。

Transfer Family は、Applicability Statement 2 (AS2) プロトコルを使用するサーバー向けに基本認証
の使用をサポートするようになりました。認証情報のために Secrets Manager の新しいシークレ
ットを作成することも、既存のシークレットを選択することもできます。詳細については、「AWS
Transfer Family ユーザーガイド」の「[AS2 コネクタの基本認証](#)」を参照してください。

Transfer Family ユーザーを認証するには、を ID プロバイダー AWS Secrets Manager として使用
できます。詳細については、「AWS Transfer Family ユーザーガイド」の「[カスタム ID プロバイ
ダーの使用](#)」およびブログ記事「[AWS Transfer Family を使用したパスワード認証の有効化 AWS
Secrets Manager](#)」を参照してください。

Transfer Family がワークフローで処理するファイルで、Pretty Good Privacy (PGP) 復号化を使用
できます。ワークフローステップで復号化を使用するには、Secrets Manager で管理する PGP キーを
提供します。詳細については、「AWS Transfer Family ユーザーガイド」の「[PGP キーの生成と管
理](#)」を参照してください。

AWS Wickr がシーク AWS Secrets Manager レットを使用する方法

AWS Wickr は、組織や政府機関がメッセージング、音声 one-to-one およびビデオ通話、ファイル共有、画面共有などを通じて安全に通信できるようにする end-to-end 暗号化されたサービスです。Wickr データ保持ボットを使用すると、ワークフローを自動化できます。ボットが AWS サービスにアクセスできる場合は、Secrets Manager シークレットを作成してボットの認証情報を保存する必要があります。詳細については、「AWS Wickr 管理ガイド」の「[データ保持ボットの開始](#)」を参照してください。

AWS Secrets Manager VPC エンドポイントの使用

パブリックインターネットからアクセスできないプライベートネットワーク上で、できるだけ多くのインフラストラクチャを実行することをお勧めします。VPC と Secrets Manager とのプライベート接続は、インターフェイス VPC エンドポイントを作成すると、確立できます。インターフェイスエンドポイントは、インターネットゲートウェイ、NAT デバイス、VPN 接続、AWS Direct Connect 接続のいずれも必要とせずに Secrets Manager にプライベートにアクセスできるテクノロジーである、[AWS PrivateLink](#) を利用します。VPC にあるインスタンスは、Secrets Manager API と通信するときに、パブリック IP アドレスを必要としません。VPC と Secrets Manager 間のトラフィックは、AWS ネットワークから離れません。詳細については、「Amazon VPC ユーザーガイド」の「[インターフェイス VPC エンドポイント \(AWS PrivateLink\)](#)」を参照してください。

シークレットマネージャーが [Lambda ローターション関数を使用して、例えばデータベース認証情報を含むシークレットをローテーションすると](#)、Lambda 関数は、データベースと Secrets Manager の両方にリクエストを送信します。[コンソールを使用して自動ローテーションをオンにすると](#)、Secrets Manager はデータベースと同じ VPC に Lambda 関数を作成します。Lambda ローターション関数から Secrets Manager へのリクエストが Amazon ネットワークから出ないように、同じ VPC に Secrets Manager エンドポイントを作成することをお勧めします。

エンドポイントのプライベート DNS を有効にすると、リージョンのデフォルト DNS 名 (secretsmanager.us-east-1.amazonaws.com など) を使って Secrets Manager への API リクエストを実行できます。詳細については、「Amazon VPC ユーザーガイド」の「[インターフェイスエンドポイントを介したサービスへのアクセス](#)」を参照してください。

アクセス許可ポリシーに条件を含めることで、Secrets Manager へのリクエストが VPC アクセスから来るようにすることができます。詳細については、「[the section called “例: アクセス許可と VPC”](#)」を参照してください。

VPC エンドポイントを介したシークレットの使用を監査するときは、AWS CloudTrail ログを使用できます。

Secrets Manager の VPC エンドポイントを作成するには

1. 「Amazon VPC [ユーザーガイド](#)」の「[インターフェイスエンドポイントの作成](#)」を参照してください。サービス名を使用します: `com.amazonaws.region.secretsmanager`
2. エンドポイントへのアクセスを制御するには、「[エンドポイントポリシーを使用して VPC エンドポイントへのアクセスを制御する](#)」を参照してください。

共有サブネット

自分と共有されているサブネットで VPC エンドポイントを作成、説明、変更、または削除することはできません。ただし、VPC エンドポイントを使用することはできます。VPC 共有の詳細については、「Amazon Virtual Private Cloud ユーザーガイド」の「[VPC を他のアカウントと共有する](#)」を参照してください。

AWS CloudFormation で AWS Secrets Manager シークレットを作成する

[シークレットを作成する](#) に示すように、CloudFormation テンプレートで [AWS::SecretsManager::Secret](#) リソースを使用して、CloudFormation スタックでシークレットを作成できます。

Amazon RDS または Aurora の管理者シークレットを作成するには、[AWS::RDS::DBCluster](#) で `ManageMasterUserPassword` を使用することをお勧めします。次に、Amazon RDS がシークレットを作成し、ローテーションを管理します。詳細については、「[マネージドローテーション](#)」を参照してください。

Amazon Redshift および Amazon DocumentDB の認証情報については、最初に Secrets Manager によって生成されたパスワードでシークレットを作成し、[動的参照](#)を使用してシークレットからユーザー名とパスワードを取得し、新しいデータベースの認証情報として使用します。次に、[AWS::SecretsManager::SecretTargetAttachment](#) リソースを使用して、Secrets Manager がシークレットをローテーションするために必要なシークレットに、データベースに関する詳細を追加します。最後に、自動ローテーションを有効にするには、[AWS::SecretsManager::RotationSchedule](#) リソースを使用して、[ローテーション関数](#) と [スケジュール](#) を提供します。以下の例を参照してください。

- [Amazon Redshift 認証情報を使用してシークレットを作成する](#)
- [Amazon DocumentDB 認証情報を使用してシークレットを作成する](#)

シークレットにリソースポリシーをアタッチするには、[AWS::SecretsManager::ResourcePolicy](#) リソースを使用します。

AWS CloudFormation を使用したリソースの作成の詳細については、「AWS CloudFormation ユーザーガイド」の「[テンプレートの基礎についての学習](#)」を参照してください。また、AWS Cloud Development Kit (AWS CDK) を使用することもできます。詳細については、[AWS Secrets Manager Construct ライブラリ](#)を参照してください。

AWS CloudFormation で AWS Secrets Manager シークレットを作成する

この例では、**CloudFormationCreatedSecret-*a1b2c3d4e5f6*** という名前のシークレットが作成されます。シークレット値は次の JSON で、シークレットの作成時に生成された 32 文字のパスワードが使用されます。

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

この例では、次の CloudFormation リソースが使用されています。

- [AWS::SecretsManager::Secret](#)

AWS CloudFormation を使用したリソースの作成の詳細については、「AWS CloudFormation ユーザーガイド」の「[テンプレートの基礎についての学習](#)」を参照してください。

JSON

```
{
  "Resources": {
    "CloudFormationCreatedSecret": {
      "Type": "AWS::SecretsManager::Secret",
      "Properties": {
        "Description": "Simple secret created by AWS CloudFormation.",
        "GenerateSecretString": {
          "SecretStringTemplate": "{\"username\": \"saanvi\"}",
          "GenerateStringKey": "password",
          "PasswordLength": 32
        }
      }
    }
  }
}
```

YAML

```
Resources:
  CloudFormationCreatedSecret:
    Type: 'AWS::SecretsManager::Secret'
    Properties:
      Description: Simple secret created by AWS CloudFormation.
      GenerateSecretString:
        SecretStringTemplate: '{"username": "saanvi"}'
        GenerateStringKey: password
        PasswordLength: 32
```

自動ローテーション付きの AWS Secrets Manager シークレットと、AWS CloudFormation 付きの Amazon RDS MySQL DB インスタンスを作成する

Amazon RDS または Aurora の管理者シークレットを作成するには、[AWS::RDS::DBCluster](#) でマスター パスワードの Secrets Manager シークレットを作成する例に示すよう

に、`ManageMasterUserPassword` を使用することをお勧めします。次に、Amazon RDS がシークレットを作成し、ローテーションを管理します。詳細については、[マネージドローテーション](#) を参照してください。

AWS Secrets Manager を使用してシークレットと Amazon Redshift クラスターを作成します AWS CloudFormation

Amazon Redshift の管理者シークレットを作成するに

は、[AWS::Redshift::Cluster](#)[AWS::RedshiftServerless::Namespace](#) およびの例を使用することをお勧めします。

で AWS Secrets Manager シークレットと Amazon DocumentDB インスタンスを作成する AWS CloudFormation

次の例では、シークレットと、そのシークレット内の認証情報をユーザーおよびパスワードとして使用する Amazon DocumentDB インスタンスを作成します。シークレットには、シークレットにアクセスできるユーザーを定義するリソースベースのポリシーがアタッチされています。また、このテンプレートでは、[ローテーション関数のテンプレート](#) から Lambda ローテーション関数を作成し、こ

のシークレットが毎月の最初の日の午前 8 時から午前 10 時 (UTC) に自動的にローテーションされるように設定します。セキュリティのベストプラクティスとして、インスタンスは Amazon VPC に置かれています。

この例では、Secrets Manager に次の CloudFormation リソースを使用します。

- [AWS::SecretsManager::Secret](#)
- [AWS::SecretsManager::SecretTargetAttachment](#)
- [AWS::SecretsManager::RotationSchedule](#)

でリソースを作成する方法については AWS CloudFormation、「AWS CloudFormation ユーザーガイド」の「[テンプレートの基本を学ぶ](#)」を参照してください。

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 Manager が AWS CloudFormation を使用する方法

コンソールを使用してローテーションをオンにすると、Secrets Manager は AWS CloudFormation を使用して、ローテーション用のリソースを作成します。そのプロセスで新しいローテーション関数を作成すると、AWS CloudFormation は適切な [ローテーション関数のテンプレート](#) に基づいて [AWS::Serverless::Function](#) を作成します。次に AWS CloudFormation は [RotationSchedule](#) を設定し、シークレットのローテーション関数とローテーションルールを設定します。自動ローテーションをオンにした後、バナーで [View stack] (スタックの表示) を選択すると、AWS CloudFormation スタックを表示できます。

自動ローテーションを有効にする方法については、[シークレットのローテーション](#) を参照してください。

で AWS Secrets Manager シークレットを作成する AWS Cloud Development Kit (AWS CDK)

CDK アプリでシークレットを作成、管理、および取得するには、[AWS Secrets Manager Construct Library](#) を使用します。これに

は、[ResourcePolicy](#)、[RotationSchedule](#)、[Secret](#)、[SecretRotation](#)、および [SecretTargetAttachment](#) のコンストラクトが含まれています。

CDK アプリケーションでシークレットを使用するためのベストプラクティスは、まず [コンソールまたは CLI を使用してシークレットを作成し](#)、次にシークレットを CDK アプリケーションにインポートすることです。

例については以下を参照してください。

- [シークレットを作成する](#)
- [シークレットをインポートする](#)
- [シークレットを取得する](#)
- [シークレットの使用許可を付与する](#)
- [シークレットをローテーションする](#)
- [データベースシークレットを作成する](#)
- [シークレットを他のリージョンにレプリケートする](#)

CDK の詳細については、「[AWS Cloud Development Kit \(AWS CDK\) v2 デベロッパーガイド](#)」を参照してください。

AWS Secrets Manager シークレットのモニタリング

AWS には、Secrets Manager のシークレットを監視し、問題が発生したときに報告し、必要に応じて自動アクションを実行するためのモニタリングツールが用意されています。ログは予期しない使用や変更を調査する場合に使用することができ、不要な変更をロールバックできます。シークレットの不適切な使用や、シークレットを削除しようとする試みに対する自動チェックを設定できます。

トピック

- [で AWS Secrets Manager イベントをログに記録する AWS CloudTrail](#)
- [Amazon AWS Secrets Manager によるモニタリング CloudWatch](#)
- [Amazon で AWS Secrets Manager イベントを照合する EventBridge](#)
- [削除が予定されている AWS Secrets Manager シークレットへのアクセスをモニタリングする](#)
- [を使用してシー AWS Secrets Manager クレットのコンプライアンスをモニタリングする AWS Config](#)
- [Secrets Manager のコストをモニタリングする](#)

で AWS Secrets Manager イベントをログに記録する AWS CloudTrail

AWS CloudTrail は、Secrets Manager コンソールからの呼び出しや、ローテーションおよびシークレットバージョン削除のための他のいくつかのイベントなど、Secrets Manager のすべての API コールをイベントとして記録します。Secrets Manager レコードのログエントリのリストについては、「[CloudTrail エントリ](#)」を参照してください。

CloudTrail コンソールを使用して、過去 90 日間の記録されたイベントを表示できます。Secrets Manager のイベントなど、AWS アカウント内のイベントの継続的な記録については、[ログファイルを Amazon S3 バケットに CloudTrail 配信するように証跡を作成します](#)。AWS [「アカウントの証跡の作成」](#)を参照してください。[複数の AWS アカウント](#) および [から CloudTrail ログファイルを受信する CloudTrail](#) ように [を設定することもできます](#) [AWS リージョン](#)。

CloudTrail ログで収集されたデータをより詳細に分析し、それに基づいて処理するように、他の AWS サービスを設定できます。[AWS 「サービスと CloudTrail ログの統合」](#)を参照してください。[が新しいログファイルを Amazon S3 バケットに CloudTrail 発行するときに通知を受け取ることもできます](#)。「[の Amazon SNS 通知の設定 CloudTrail](#)」を参照してください。

CloudTrail ログから Secrets Manager イベントを取得するには (コンソール)

1. <https://console.aws.amazon.com/cloudtrail/> で CloudTrail コンソールを開きます。
2. コンソールが、イベントの発生したリージョンを示していることを確認します。コンソールには、選択したリージョンで発生したイベントのみが表示されます。コンソールの右上のドロップダウンリストからリージョンを選択してください。
3. 左のナビゲーションペインで [Event history] (イベント履歴) を選択します。
4. [Filter] (フィルター) 条件、および [Time range] (時間範囲) (またはその両方) を選択すると探しているイベントを見つけるのに役立ちます。例:
 - a. すべての Secrets Manager イベントを表示するには、ルックアップ属性 で、イベントソース を選択します。次に、[Enter event source] (イベントソースの入力) で、**secretsmanager.amazonaws.com** を選択します。
 - b. シークレットのすべてのイベントを表示するには、ルックアップ属性 でリソース名 を選択します。次に、リソース名 を入力する に、シークレットの名前を入力します。
5. 追加の詳細を表示するには、イベントの横の拡張矢印を選択します。利用可能なすべての情報を表示するには、[View event] (イベントの表示) を選択します。

AWS CLI

Example CloudTrail ログから Secrets Manager イベントを取得する

次の [lookup-events](#) の例では、Secrets Manager のイベントが検索されます。

```
aws cloudtrail lookup-events \  
  --region us-east-1 \  
  --lookup-attributes  
  AttributeKey=EventSource,AttributeValue=secretsmanager.amazonaws.com
```

AWS CloudTrail Secrets Manager の エントリ

AWS Secrets Manager は、すべての Secrets Manager オペレーション、およびローテーションと削除に関連するその他のイベントについて AWS CloudTrail、ログにエントリを書き込みます。これらのイベントに対するアクションの詳細については、「[Secrets Manager イベントを と一致させる EventBridge](#)」を参照してください。

ログエントリの種類

- [Secrets Manager の操作ログエントリ](#)
- [削除用のログエントリ](#)
- [レプリケーションのログエントリ](#)
- [ローテーションのログエントリ](#)

Secrets Manager の操作ログエントリ

Secrets Manager の操作の呼び出しによって発生するイベントには "detail-type": ["AWS API Call via CloudTrail"] があります。

Note

2024 年 2 月以前は、一部の Secrets Manager オペレーションで、シークレット ARN の「arn」ではなく「aRN」を含むイベントが報告されていました。詳細については、「[AWS re:Post](#)」を参照してください。

以下は、ユーザーまたはサービスが API、SDK、または CLI を介して Secrets Manager オペレーションを呼び出すときに生成される CloudTrail エントリです。

BatchGetSecretValue

[BatchGetSecretValue](#) オペレーションによって生成されます。シークレットを取得する方法の詳細については、「[シークレットを取得する](#)」を参照してください。

CancelRotateSecret

[CancelRotateSecret](#) オペレーションによって生成されます。ローテーションの詳細は、「[シークレットのローテーション](#)」を参照してください。

CreateSecret

[CreateSecret](#) オペレーションによって生成されます。シークレットを作成する方法の詳細については、「[シークレットの作成と管理](#)」を参照してください。

DeleteResourcePolicy

[DeleteResourcePolicy](#) オペレーションによって生成されます。許可の詳細については、「[認証とアクセスコントロール](#)」を参照してください。

DeleteSecret

[DeleteSecret](#) オペレーションによって生成されます。シークレットの削除については、「[the section called “シークレットの削除”](#)」を参照してください。

DescribeSecret

[DescribeSecret](#) オペレーションによって生成されます。

GetRandomPassword

[GetRandomPassword](#) オペレーションによって生成されます。

GetResourcePolicy

[GetResourcePolicy](#) オペレーションによって生成されます。許可の詳細については、「[認証とアクセスコントロール](#)」を参照してください。

GetSecretValue

[GetSecretValue](#) および [BatchGetSecretValue](#) オペレーションによって生成されます。シークレットを取得する方法の詳細については、「[シークレットを取得する](#)」を参照してください。

ListSecrets

[ListSecrets](#) オペレーションによって生成されます。シークレットを一覧する方法の詳細については、「[the section called “シークレットを検索する”](#)」を参照してください。

ListSecretVersionIds

[ListSecretVersionIds](#) オペレーションによって生成されます。

PutResourcePolicy

[PutResourcePolicy](#) オペレーションによって生成されます。許可の詳細については、「[認証とアクセスコントロール](#)」を参照してください。

PutSecretValue

[PutSecretValue](#) オペレーションによって生成されます。シークレットを更新する方法の詳細については、「[the section called “シークレットの変更”](#)」を参照してください。

RemoveRegionsFromReplication

[RemoveRegionsFromReplication](#) オペレーションによって生成されます。シークレットのレプリケーションの詳細については、「[リージョン間でシークレットをレプリケートする](#)」を参照してください。

ReplicateSecretToRegions

[ReplicateSecretToRegions](#) オペレーションによって生成されます。シークレットのレプリケーションの詳細については、「[リージョン間でシークレットをレプリケートする](#)」を参照してください。

RestoreSecret

[RestoreSecret](#) オペレーションによって生成されます。削除されたシークレットを復元する方法については、「[the section called “シークレットを復元する”](#)」を参照してください。

RotateSecret

[RotateSecret](#) オペレーションによって生成されます。ローテーションの詳細は、「[シークレットのローテーション](#)」を参照してください。

StopReplicationToReplica

[StopReplicationToReplica](#) オペレーションによって生成されます。シークレットのレプリケーションの詳細については、「[リージョン間でシークレットをレプリケートする](#)」を参照してください。

TagResource

[TagResource](#) オペレーションによって生成されます。シークレットのタグ付けの詳細については、「[the section called “シークレットにタグ付けする”](#)」を参照してください。

UntagResource

[UntagResource](#) オペレーションによって生成されます。シークレットのタグ解除の詳細については、「[the section called “シークレットにタグ付けする”](#)」を参照してください。

UpdateSecret

[UpdateSecret](#) オペレーションによって生成されます。シークレットを更新する方法の詳細については、「[the section called “シークレットの変更”](#)」を参照してください。

UpdateSecretVersionStage

[UpdateSecretVersionStage](#) オペレーションによって生成されます。バージョンステージの詳細については、「[the section called “シークレットバージョン”](#)」を参照してください。

ValidateResourcePolicy

[ValidateResourcePolicy](#) オペレーションによって生成されます。許可の詳細については、「[認証とアクセスコントロール](#)」を参照してください。

削除用のログエントリ

Secrets Manager 操作のイベントに加えて、Secrets Manager は削除に関連する次のイベントを生成します。これらのイベントには "detail-type": ["AWS Service Event via CloudTrail"] があります。

CancelSecretVersionDelete

Secrets Manager サービスによって生成されるもの。バージョンを持つシークレットで DeleteSecret を呼び出し、後で RestoreSecret を呼び出すと、Secrets Manager は、復元されたシークレットバージョンごとにこのイベントをログに記録します。削除されたシークレットを復元する方法については、「[the section called “シークレットを復元する”](#)」を参照してください。

EndSecretVersionDelete

シークレットバージョンが削除されたときに、Secrets Manager サービスによって生成されるもの。詳細については、「[the section called “シークレットの削除”](#)」を参照してください。

StartSecretVersionDelete

Secrets Manager がシークレットバージョンの削除を開始する際に、Secrets Manager サービスによって生成されるもの。シークレットの削除については、「[the section called “シークレットの削除”](#)」を参照してください。

SecretVersionDeletion

Secrets Manager が廃止されたシークレットバージョンを削除ときに Secrets Manager サービスによって生成。詳細については、「[Secret versions](#)」(シークレットバージョン)を参照してください。

レプリケーションのログエントリ

Secrets Manager 操作のイベントに加えて、Secrets Manager はレプリケーションに関連する次のイベントを生成します。これらのイベントには "detail-type": ["AWS Service Event via CloudTrail"] があります。

ReplicationFailed

レプリケーションが失敗したときに Secrets Manager サービスによって生成されます。シークレットのレプリケーションの詳細については、「[リージョン間でシークレットをレプリケートする](#)」を参照してください。

ReplicationStarted

Secrets Manager がシークレットのレプリケーションを開始する際に、Secrets Manager サービスによって生成されます。シークレットのレプリケーションの詳細については、「[リージョン間でシークレットをレプリケートする](#)」を参照してください。

ReplicationSucceeded

Secrets Manager サービスが正常にレプリケートされたときに Secrets Manager サービスによって生成されます。シークレットのレプリケーションの詳細については、「[リージョン間でシークレットをレプリケートする](#)」を参照してください。

ローテーションのログエントリ

Secrets Manager 操作のイベントに加えて、Secrets Manager はローテーションに関連する次のイベントを生成します。これらのイベントには "detail-type": ["AWS Service Event via CloudTrail"] があります。

RotationStarted

Secrets Manager がシークレットのローテーションを開始する際に、Secrets Manager サービスによって生成されるもの。ローテーションの詳細は、「[シークレットのローテーション](#)」を参照してください。

RotationAbandoned

Secrets Manager がローテーションの試みを放棄し、シークレットの既存のバージョンから AWSPENDING ラベルを削除するときに、Secrets Manager サービスによって生成されるもの。Secrets Manager は、ローテーション中にシークレットの新しいバージョンを作成すると、ローテーションを中止します。ローテーションの詳細は、「[シークレットのローテーション](#)」を参照してください。

RotationFailed

ローテーションに失敗したときに、Secrets Manager サービスによって生成されるもの。ローテーションの詳細は、「[the section called “におけるローテーションのトラブルシューティング”](#)」を参照してください。

RotationSucceeded

Secrets Manager サービスが正常にローテーションされたときに Secrets Manager サービスによって生成されるもの。ローテーションの詳細は、「[シークレットのローテーション](#)」を参照してください。

TestRotationStarted

Secrets Manager が、即時ローテーションが予定されていないシークレットのローテーションテストを開始したときに Secrets Manager サービスによって生成されるもの。ローテーションの詳細は、「[シークレットのローテーション](#)」を参照してください。

TestRotationSucceeded

Secrets Manager が、即時ローテーションが予定されていないシークレットのローテーションテストに成功したときに生成されるもの。ローテーションの詳細は、「[シークレットのローテーション](#)」を参照してください。

TestRotationFailed

Secrets Manager が、即時ローテーションが予定されていないシークレットのローテーションをテストし、ローテーションが失敗したときに、Secrets Manager サービスによって生成されるもの。ローテーションの詳細は、「[the section called “におけるローテーションのトラブルシューティング”](#)」を参照してください。

Amazon AWS Secrets Manager によるモニタリング CloudWatch

Amazon を使用すると CloudWatch、AWS サービスをモニタリングし、アラームを作成して、メトリクスがいつ変更されたかを知らせることができます。はこれらの統計を 15 か月間 CloudWatch 保持するため、履歴情報にアクセスしてウェブアプリケーションまたはサービスの動作をよりの確に把握できます。では AWS Secrets Manager、削除対象としてマークされたシークレットを含むアカウント内のシークレットの数、およびコンソールを介した呼び出しを含む Secrets Manager への API コールをモニタリングできます。メトリクスをモニタリングする方法については、「[ユーザーガイド](#)」の [CloudWatch](#) 「メトリクスの使用CloudWatch」を参照してください。

Secrets Manager メトリクスを検索するには

1. CloudWatch コンソールのメトリクスで、すべてのメトリクスを選択します。
2. メトリクス検索のボックスに、と入力しますsecret。
3. 以下の操作を実行します。
 - アカウント内のシークレットの数をモニタリングするには、AWS/SecretsManager を選択し、を選択しますSecretCount。このメトリクスは 1 時間ごとに発行されます。
 - コンソールを介した呼び出しを含む Secrets Manager への API 呼び出しをモニタリングするには、使用状況 > AWS リソースで、モニタリングする API 呼び出しを選択します。Secrets Manager APIs [「Secrets Manager オペレーション」](#)を参照してください。

4. 以下の操作を実行します。

- メトリクスのグラフを作成するには、「Amazon CloudWatch ユーザーガイド」の「[メトリクスのグラフ化](#)」を参照してください。
- 異常を検出するには、「Amazon CloudWatch [ユーザーガイド](#)」の CloudWatch 「[異常検出の使用](#)」を参照してください。
- メトリクスの統計を取得するには、「Amazon [ユーザーガイド](#)」の「[メトリクスの統計を取得する](#)」を参照してください。 CloudWatch

CloudWatch アラーム

メトリクスの値が変更され、CloudWatch アラームの状態が変化したときに Amazon SNS メッセージを送信するアラームを作成できます。アカウントのシークレット数ResourceCountである Secrets Manager メトリクスにアラームを設定できます。アラームは、指定した期間にわたってメトリクスを監視し、複数の期間にわたって特定のしきい値に対するメトリクスの値に基づいてアクションを実行することもできます。アラームは、持続している状態変化に対してのみアクションを呼び出します。CloudWatch アラームは、特定の状態にあるという理由だけではアクションを呼び出しません。状態が変更され、指定された期間にわたって維持されている必要があります。

詳細については、「CloudWatch ユーザーガイド」の「[Amazon CloudWatch アラームの使用](#)」および「[異常検出に基づいて CloudWatch アラームを作成する](#)」を参照してください。

また、特定のしきい値をモニタリングするアラームを設定し、しきい値に達したときに通知を送信したりアクションを実行したりできます。詳細については、「[Amazon ユーザーガイド CloudWatch](#)」を参照してください。

Amazon で AWS Secrets Manager イベントを照合する EventBridge

Amazon では EventBridge、CloudTrail ログエントリから Secrets Manager イベントを照合できます。これらのイベントを検索し、新しく生成されたイベントをターゲットに送信してアクションを実行する EventBridge ルールを設定できます。Secrets Manager がログに記録する CloudTrail エントリのリストについては、「」を参照してください[CloudTrail エントリ](#)。をセットアップする手順については EventBridge、「ユーザーガイド」の「[の開始 EventBridge](#)方法EventBridge」を参照してください。

指定したシークレットに対するすべての変更にもマッチさせる

Note

[Secrets Manager イベント](#)の中には、シークレットの ARN を異なる大文字で返すものもあるため、複数のアクションにマッチするイベントパターンでは、ARN でシークレットを指定するために、arn キーと aRN の両方を含める必要があります。これは既知の問題であり、すべてのイベントが arn をレポートするよう、レポートを修正する修正プログラムを 2024 年 1 月に開発中です。詳細については、「[AWS re:Post](#)」を参照してください。

次の例は、シークレットへの変更のログエントリに一致する EventBridge イベントパターンを示しています。

```
{
  "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"]
    }
  }
}
```

シークレット値がローテーションされたらイベントをマッチさせる

次の例は、手動更新または自動ローテーションによって発生するシークレット値の変更の CloudTrail ログエントリに一致する EventBridge イベントパターンを示しています。これらのイベントには、Secrets Manager の操作によるものもあれば、Secrets Manager サービスによって生成されるものもあるため、detail-type を両方に含める必要があります。

```
{
  "source": ["aws.secretsmanager"],
  "$or": [
    { "detail-type": ["AWS API Call via CloudTrail"] },

```

```
{ "detail-type": ["AWS Service Event via CloudTrail"] },
],
"detail": {
  "eventSource": ["secretsmanager.amazonaws.com"],
  "eventName": ["PutSecretValue", "UpdateSecret", "RotationSucceeded"]
}
}
```

削除が予定されている AWS Secrets Manager シークレットへのアクセスをモニタリングする

AWS CloudTrail、Amazon CloudWatch Logs、および Amazon Simple Notification Service (Amazon SNS) の組み合わせを使用して、削除保留中のシークレットへのアクセス試行を通知するアラームを作成できます。アラームから通知を受け取ると、削除が本当に必要かどうかを時間をかけて判断するために、シークレットの削除をキャンセルしなければならない場合があります。調査の結果、シークレットが実際にまだ必要であるため、シークレットが保存されたままになる可能性があります。または、使用する新しいシークレットの詳細を使用して、ユーザーを更新する必要がある場合があります。

次の手順では、GetSecretValueオペレーションのリクエストによって特定のエラーメッセージが CloudTrail ログファイルに書き込まれたときに通知を受け取る方法について説明します。他の API オペレーションは、アラームをトリガーせずにシークレットで実行できます。この CloudWatch アラームは、古い認証情報を使用しているユーザーまたはアプリケーションを示す可能性のある使用状況を検出します。

これらの手順を開始する前に、AWS Secrets Manager API リクエストをモニタリングする AWS リージョン アカウントと アカウント CloudTrail で を有効にする必要があります。手順については、「AWS CloudTrail ユーザーガイド」の「[Creating a trail for the first time](#)」を参照してください。

ステップ 1: CloudTrail ログへのログファイル配信を設定する CloudWatch

CloudTrail ログファイルの CloudWatch Logs への配信を設定する必要があります。これにより、CloudWatch Logs は Secrets Manager API リクエストを監視して、削除保留中のシークレットを取得できます。

CloudWatch Logs への CloudTrail ログファイル配信を設定するには

1. <https://console.aws.amazon.com/cloudtrail/> で CloudTrail コンソールを開きます。

2. 上部のナビゲーションバーで、シークレットをモニタリングする AWS リージョンを選択します。
3. 左側のナビゲーションペインで、証跡 を選択し、 に設定する証跡の名前を選択します
CloudWatch。
4. 証跡設定ページで、CloudWatch ログセクションまでスクロールし、編集アイコン () を選択します 。
5. [New or existing log group] (新規または既存のロググループ)にロググループの名前 (**CloudTrail/MyCloudWatchLogGroup**) を入力します。
6. IAM ロール では、CloudTrail_CloudWatchLogs_Role という名前のデフォルトロールを使用できます。このロールには、ロググループに CloudTrail イベントを配信するために必要なアクセス許可を持つデフォルトのロールポリシーがあります。
7. [Continue] (続行) を選択して設定を保存します。
8. AWS CloudTrail では、アカウントの API アクティビティに関連する CloudTrail イベントを CloudWatch ログロググループページに配信し、 を許可する を選択します。

ステップ 2: アラームを作成する CloudWatch

Secrets Manager GetSecretValue API オペレーションが削除保留中のシークレットへのアクセスをリクエストしたときに通知を受け取るには、CloudWatch アラームを作成して通知を設定する必要があります。

CloudWatch アラームを作成するには

1. <https://console.aws.amazon.com/cloudwatch/> で CloudWatch コンソールにサインインします。
2. 上部のナビゲーションバーで、シークレットをモニタリングする AWS リージョンを選択します。
3. 左のナビゲーションペインで [Logs] (ログ) を選択します。
4. ロググループ のリストで、前の手順で作成したロググループの横にある CloudTrail/MyCloudWatchLogGroup などのチェックボックスをオンにします。その後、[Create Metric Filter] (メトリクスフィルタの作成) を選択します。
5. [Filter Pattern] (フィルタパターン) に、以下を入力するか貼り付けます。

```
{ $.eventName = "GetSecretValue" && $.errorMessage = "*secret because it was marked for deletion*" }
```

- [Assign Metric] (メトリクスの割り当て) を選択します。
6. [Create Metric Filter and Assign a Metric] (メトリクスフィルタの作成とメトリクスの割り当て) ページで、次の操作を実行します。
 - a. [Metric Namespace] (メトリクス名前空間) に「**CloudTrailLogMetrics**」と入力します。
 - b. [Metric Name] (メトリクス名) に「**AttemptsToAccessDeletedSecrets**」と入力します。
 - c. [Show advanced metric settings] (メトリクスの詳細設定の表示) を選択した後、必要に応じて [Metric Value] (メトリクス値) に「**1**」と入力します。
 - d. [Create Filter] (フィルタの作成) を選択します。
 7. フィルタボックスで、[Create Alarm] (アラームの作成) を選択します。
 8. [Create Alarm] (アラームの作成) ウィンドウで、以下の操作を行います。
 - a. [Name] (名前) に、「**AttemptsToAccessDeletedSecretsAlarm**」と入力します。
 - b. [Whenever:] (次の時:) の [is:] (が:) で [>=] を選択し、「**1**」と入力します。
 - c. [Send notification to:] (通知の送信:) の横で、次のいずれかを実行します。
 - 新しい Amazon SNS トピックを作成し使用するには、[New list] (新しいリスト) を選択し、新しいトピック名を入力します。[Email list:] (E メールリスト:) に、E メールアドレスを少なくとも 1 つ入力します。カンマで区切って、複数の E メールアドレスを入力できます。
 - 既存の Amazon SNS トピックを使用するには、使用するトピックの名前を選択します。リストが存在しない場合は、[Select list] (リストの選択) を選択します。
 - d. [Create Alarm] (アラームの作成) を選択します。

ステップ 3: アラームを CloudWatch テストする

アラームをテストするには、シークレットを作成し、それを削除用にスケジュールします。次に、シークレット値の取得を試みます。アラームで設定したアドレスに間もなく E メールが届きます。これは、削除予定のシークレットの使用について警告します。

を使用してシー AWS Secrets Manager クレットのコンプライアンスをモニタリングする AWS Config

AWS Config を使用してシークレットを評価し、標準に準拠しているかどうかを確認できます。AWS Config ルールを使用して、シークレットの内部セキュリティおよびコンプライアンス要件を定義します。その後 AWS Config、はルールに準拠していないシークレットを識別できます。シークレットメタデータ、[ローテーション設定](#)、シークレット暗号化に使用される KMS キー、Lambda ローテーション関数、およびシークレットに関連付けられたタグの変更を追跡することもできます。

変更を通知する AWS Config ように を設定できます。詳細については、[「が Amazon SNS トピック AWS Config に送信する通知」](#)を参照してください。

複数の AWS アカウント と組織 AWS リージョン 内にシークレットがある場合は、その設定とコンプライアンスデータを集約できます。詳細については、[「マルチアカウントマルチリージョンデータ集約」](#)を参照してください。

シークレットがコンプライアンスに準拠しているかどうかを評価するには

- [AWS Config ルール を使用してリソースを評価する](#) 手順に従い、次のいずれかのルールを選択します。
 - [secretsmanager-secret-unused](#) — 指定した日数内にシークレットがアクセスされたかどうかを確認します。
 - [secretsmanager-using-cmk](#) — シークレットが または AWS マネージドキー aws/secretsmanager で作成したカスタマー マネージドキーを使用して暗号化されているかどうかを確認します AWS KMS。
 - [secretsmanager-rotation-enabled-check](#) — Secrets Manager に保存されているシークレットに対してローテーションが設定されているかどうかを確認します。
 - [secretsmanager-scheduled-rotation-success-check](#) — 最後に成功したローテーションが、設定されたローテーション頻度の範囲内であるかどうかをチェックします。チェックの最小頻度は日次です。
 - [secretsmanager-secret-periodic-rotation](#) — 指定した日数内にシークレットがローテーションされたかどうかを確認します。

Secrets Manager のコストをモニタリングする

Amazon を使用して CloudWatch、推定 AWS Secrets Manager 請求額をモニタリングできます。詳細については、「ユーザーガイド」の [「推定 AWS 請求額をモニタリングするための請求アラームの作成CloudWatch」](#) を参照してください。

コストをモニタリングするもう 1 つのオプションは、AWS コスト異常検出です。詳細については、「[AWS コスト管理ユーザーガイド](#)」の「[コスト異常検出による異常な支出の検出](#)」を参照してください。AWS

Secrets Manager の使用状況をモニタリングする方法については、[the section called “によるモニタリング CloudWatch”](#)「」および「」を参照してください[the section called “でログ記録する AWS CloudTrail”](#)。

AWS Secrets Manager 料金の詳細については、「」を参照してください[the section called “料金”](#)。

のコンプライアンス検証 AWS Secrets Manager

Secrets Manager を使用する際のお客様のコンプライアンス責任は、お客様のデータの機密性、貴社のコンプライアンス目的、適用される法律および規制によって決まります。では、コンプライアンスに役立つ以下のリソース AWS を提供しています。

- [「セキュリティ & コンプライアンス クイックリファレンスガイド」](#) – これらのデプロイガイドには、アーキテクチャ上の考慮事項の説明と、AWS でセキュリティとコンプライアンスに重点を置いたベースライン環境をデプロイするための手順が記載されています。
- [HIPAA セキュリティとコンプライアンスのアーキテクチャに関するホワイトペーパー](#) – このホワイトペーパーでは、企業が AWS を使用して HIPAA 準拠のアプリケーションを作成する方法について説明します。
- [AWS コンプライアンスリソース](#) – このワークブックとガイドのコレクションは、お客様の業界や地域に適用される場合があります。
- AWS Config では、自社プラクティス、業界ガイドライン、および規制に対するリソースの設定の準拠状態が評価されます。詳細については、「[the section called “シークレットのコンプライアンスをモニタリングする”](#)」を参照してください。
- [AWS Security Hub](#) は、内のセキュリティ状態を包括的に把握し、セキュリティ業界標準とベストプラクティスへの準拠を確認するのに役立ちます。Security Hub を使用して Secrets Manager リソースを評価する方法の詳細については、「AWS Security Hub ユーザーガイド」の「[AWS Secrets Manager コントロール](#)」を参照してください。
- IAM アクセスアナライザーは、外部エンティティにシークレットへのアクセスを許可するポリシー (ポリシー内の条件ステートメントを含む) を分析します。詳細については、「[アクセスアナライザーを使用したアクセスのプレビュー](#)」を参照してください。
- AWS Systems Manager には、Secrets Manager の定義済みのランブックが用意されています。詳細については、「[シークレットマネージャーの、Systems Manager Automation ランブックリファレンス](#)」を参照してください。
- を使用して、サードパーティーの監査レポートをダウンロードできます AWS Artifact。詳細については、「[でのレポートのダウンロード AWS Artifact](#)」の」を参照してください。

コンプライアンス標準

AWS Secrets Manager は、以下の規格の監査を受けており、コンプライアンス認定を取得する必要がある場合にソリューションの一部になる可能性があります。

- HIPAA – は、医療保険の相互運用性と説明責任に関する法律 (HIPAA) コンプライアンスプログラムを拡張し、HIPAA の対象となるサービス AWS Secrets Manager としてを含め AWS ました。<https://aws.amazon.com/compliance/hipaa-eligible-services-reference/>と事業提携契約 (BAA) を締結している場合は AWS、Secrets Manager を使用して HIPAA 準拠のアプリケーションを構築できます。は、医療情報の処理と保存 AWS に を活用する方法について詳しく知りたいお客様向けに、[HIPAA に焦点を当てたホワイトペーパー](#) AWS を提供しています。詳細については、「[HIPAA コンプライアンス](#)」を参照してください。
- PIC 参加組織 – サービスプロバイダーレベル 1 で、Payment Card Industry (PCI) データセキュリティスタンダード (DSS) バージョン 3.2 のコンプライアンスの認証 AWS Secrets Manager を受けています。AWS 製品やサービスを使用してカード所有者データを保存、処理、または送信するお客様は、独自の PCI DSS コンプライアンス証明書を管理する AWS Secrets Manager 際に を使用できます。PCI コンプライアンスパッケージのコピーをリクエストする方法など、AWS PCI DSS の詳細については、「[PCI DSS レベル 1](#)」を参照してください。
- ISO – ISO/IEC 27001、ISO/IEC 27017、ISO/IEC 27018、および ISO 9001 のコンプライアンス認定を正常に完了 AWS Secrets Manager しました。詳細については、「[ISO 27001](#)」、「[ISO 27017](#)」、「[ISO 27018](#)」、および「[ISO 9001](#)」を参照してください。
- AICPA SOC – System and Organization Control (SOC) レポートは、Secrets Manager が主要なコンプライアンスコントロールと目的をどのように達成したかを示す、独立したサードパーティーによる審査レポートです。これらのレポートの目的は、ユーザーと監査者が運用とコンプライアンスをサポートするために確立された AWS コントロールを理解できるようにすることです。詳細については、「[SOC Compliance](#)」(SOC コンプライアンス) を参照してください。
- FedRAMP – Federal Risk and Authorization Management Program (FedRAMP) は、クラウド製品やサービスのセキュリティ評価、認可、継続的モニタリングに対する標準化されたアプローチを提供する政府全体のプログラムです。FedRAMP プログラムは、東部/西部のサービスおよびリージョン、および政府または規制対象のデータを消費 GovCloud するための暫定認可も提供します。詳細については、「[FedRAMP への準拠](#)」を参照してください。
- 国防総省 - 国防総省 (DoD) クラウドコンピューティングセキュリティ要件ガイド (SRG) は、クラウドサービスプロバイダー (CSPs) が DoD の暫定認可を取得するための標準化された評価および認可プロセスを提供し、DoD のお客様にサービスを提供できるようにします。詳細については、「[DoD SRG リソース](#)」を参照してください。
- IRAP – Information Security Registered Assessors Program (IRAP) を使用すると、オーストラリア政府のお客様は、適切なコントロールが実施されていることを検証し、オーストラリアサイバーセキュリティセンター (ACSC) によって作成されたオーストラリア政府情報セキュリティマニュアル (ISM) の要件に対処するための適切な責任モデルを決定できます。詳細については、「[IRAP リソース](#)」を参照してください。

- OSPAR — Amazon Web Services (AWS) は、シンガポール銀行協会 (ABS) の「アウトソースサービスプロバイダーの統制目標と手順に関するガイドライン」(ABS ガイドライン) AWS に従って、シンガポールの金融サービス業界が設定したクラウドサービスプロバイダーに対する高い期待を満たすことを顧客に AWS 約束しました。リソースの詳細については、[OSPAR リソース](#)を参照してください。

AWS Secrets Manager でのセキュリティ

AWS では、セキュリティが最優先事項です。AWS のお客様は、セキュリティを最も重視する組織の要件を満たすよう構築されたデータセンターとネットワークアーキテクチャを利用できます。

お客様と AWS では、セキュリティの責任を共有します。[責任共有モデル](#)では、これをクラウドのセキュリティおよびクラウド内のセキュリティとして説明しています。

- クラウドのセキュリティ - AWS は、AWS クラウド内で AWS サービスを実行するインフラストラクチャを保護する責任を負います。また、AWS は安全に使用できるサービスを提供します。サードパーティーの監査人は、[AWS コンプライアンスプログラム](#)の一環として、セキュリティの有効性を定期的にテストおよび検証します。AWS Secrets Manager に適用されるコンプライアンスプログラムの詳細については、「[コンプライアンスプログラムによる対象範囲内の AWS のサービス](#)」を参照してください。
- クラウド内のセキュリティ - お客様の責任は、お客様の AWS のサービスによって決まります。また、ユーザーは、データの機密性、企業要件、および適用法令と規制などのその他要因に対する責任も担います。

その他のリソースについては、[セキュリティの柱 - AWS Well-Architected フレームワーク](#)を参照してください。

トピック

- [AWS CLI を使用して AWS Secrets Manager のシークレットを保存する際のリスクを軽減する](#)
- [AWS Secrets Manager でのデータ保護](#)
- [でのシークレットの暗号化と復号 AWS Secrets Manager](#)
- [AWS Secrets Manager でのインフラストラクチャセキュリティ](#)
- [AWS Secrets Manager での回復性](#)
- [ポスト量子 TLS](#)

AWS CLI を使用して AWS Secrets Manager のシークレットを保存する際のリスクを軽減する

AWS Command Line Interface (AWS CLI) を使用して AWS オペレーション呼び出す場合は、それらのコマンドをコマンドシェルに入力します。たとえば、Windows コマンドプロンプトや Windows

PowerShell、または Bash や Z シェルなどを使用できます。これらのコマンドシェルの多くには、生産性を向上させるために設計された機能が含まれています。ただし、この機能はシークレットを侵害するために使用される可能性があります。たとえば、ほとんどのシェルでは、上矢印キーを使用して、最後に入力されたコマンドを表示できます。コマンド履歴機能は、セキュリティ保護されていないセッションにアクセスされた場合に、悪用される可能性があります。また、バックグラウンドで動作する他のユーティリティは、タスクをより効率的に実行できるようにする目的でコマンドパラメータにアクセスできます。このようなリスクを軽減するには、次の手順を実行します。

- コンソールから離れるときは、常にコンピュータをロックします。
- 要らないまたは使用しなくなったコンソールユーティリティをアンインストールするか無効にします。
- シェルまたはリモートアクセスプログラムを使用している場合は、入力したコマンドのログを記録しないようにします。
- シェルのコマンド履歴によってキャプチャされないようにパラメータを渡す方法を使用します。次の例は、シークレットテキストをテキストファイルに入力し、AWS Secrets Manager コマンドに渡してすぐにそのファイルを破棄する方法を示しています。これは、典型的なシェル履歴がシークレットテキストをキャプチャしないことを意味します。

次の例は、典型的な Linux コマンドを示しています (シェルには多少異なるコマンドが必要な場合があります)。

```
$ 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.
```

これらのコマンドを実行した後、上下の矢印を使用してコマンド履歴をスクロールし、シークレットテキストがどの行にも表示されないようにする必要があります。

⚠ Important

デフォルトでは、コマンド履歴バッファのサイズを最初に [1] に減らさないかぎり、Windows で同様のテクニックを実行することはできません。

Windows コマンドプロンプトで 1 コマンドのコマンド履歴バッファのみを使用するように設定するには

1. 管理者コマンドプロンプトを開きます ([Run as administrator (管理者として実行)])。
2. 左上にあるアイコンを選択し、[プロパティ] を選択します。
3. [オプション] タブで、[バッファサイズ] と [Number of Buffers (バッファ数)] の両方を **1** に設定し、[OK] を選択します。
4. 履歴に入れたくないコマンドを入力する必要がある場合は、直後に次のようなコマンドを 1 つ続けます。

```
echo.
```

これにより、機密性の高いコマンドがフラッシュされます。

Windows Command Prompt シェルでは、[SysInternals SDelete](#) ツールをダウンロードして、次のようなコマンドを使用することができます。

```
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.
```

AWS Secrets Manager でのデータ保護

AWS [責任共有モデル](#) は、AWS Secrets Manager でのデータ保護に適用されます。このモデルで説明されているように、AWS は、AWS クラウド のすべてを実行するグローバルインフラストラクチャを保護する責任を負います。顧客は、このインフラストラクチャでホストされているコンテンツに対する管理を維持する責任があります。このコンテンツには、使用される AWS のサービスのセキュリティ構成と管理タスクが含まれます。データプライバシーの詳細については、「[データプライバシーのよくある質問](#)」を参照してください。欧州でのデータ保護の詳細については、[AWS セキュリティブログ](#) に投稿された AWS 責任共有モデルおよび GDPR ブログを参照してください。

データを保護するため、AWS アカウント の認証情報を保護し、AWS Identity and Access Management (IAM) を使用して個々のユーザーアカウントをセットアップすることをお勧めします。この方法により、それぞれのジョブを遂行するために必要な許可のみを各ユーザーに付与できます。また、次の方法でデータを保護することをお勧めします。

- 各アカウントで [多要素認証 \(MFA\)](#) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。Secrets Manager は、すべてのリージョンで TLS 1.2 および 1.3 をサポートしています。また、Secrets Manager は [TLS \(PQTL\) ネットワーク暗号化プロトコル用のハイブリッドポスト量子キー交換オプション](#) もサポートしています。
- IAM プリンシパルに関連付けられているアクセスキー ID とシークレットアクセスキーを使用して、Secrets Manager へのプログラムリクエストに署名します。または、[AWS Security Token Service](#) (AWS STS) を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。
- AWS CloudTrail で API とユーザーアクティビティログをセットアップします。「[the section called “でログ記録する AWS CloudTrail”](#)」を参照してください。
- コマンドラインインターフェイスまたは API を使用して AWS にアクセスするときに FIPS 140-2 検証済みの暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。「[the section called “Secrets Manager エンドポイント”](#)」を参照してください。
- AWS CLI を使用して Secrets Manager にアクセスする場合、「[the section called “AWS CLI を使用して AWS Secrets Manager のシークレットを保存する際のリスクを軽減する”](#)」の手順に従います。

保管中の暗号化

Secrets Manager は AWS Key Management Service (AWS KMS) を介して暗号化を使用し、保管中のデータの機密性を保護します。AWS KMS には、多くの AWS のサービスが使用するキーストレ

ジおよび暗号化サービスが用意されています。Secrets Manager のシークレットはすべて、一意のデータキーで暗号化されます。各データキーは、KMS キーで保護されます。Secrets Manager AWS マネージドキーでアカウントにデフォルトの暗号化を使用することも、AWS KMS で独自のカスタマー管理キーを作成することもできます。カスタマー管理キーを使用すると、KMS キーアクティビティの認証をきめ細かく制御できます。詳細については、「[the section called “シークレット暗号化と復号”](#)」を参照してください。

転送中の暗号化

Secrets Manager は、転送中のデータを暗号化するための安全なプライベートエンドポイントを提供します。安全なプライベートエンドポイントにより、AWS では、Secrets Manager への API リクエストの整合性を保護できます。AWS では、X.509 証明書や Secrets Manager シークレットアクセスキーを使用して、発信者が API コールに署名する必要があります。この要件は、[署名バージョン 4 署名プロセス \(Sigv4\)](#) に記載されています。

AWS Command Line Interface (AWS CLI)、またはいずれかの AWS SDK を使用して AWS を呼び出す場合は、アクセスキーを設定します。その後、これらのツールは自動的にアクセスキーを使用してリクエストに署名します。「[the section called “AWS CLI を使用して AWS Secrets Manager のシークレットを保存する際のリスクを軽減する”](#)」を参照してください。

ネットワーク間トラフィックのプライバシー

AWS には、既知のネットワークルートとプライベートネットワークルートを経由してトラフィックをルーティングする際にプライバシーを維持するためのオプションが用意されています。

サービスとオンプレミスのクライアントおよびアプリケーションとの間のトラフィック

プライベートネットワークと AWS Secrets Manager との間には 2 つの接続オプションがあります

- AWS Site-to-Site VPN 接続。詳細については、「[AWS Site-to-Site VPN とは](#)」を参照してください。
- AWS Direct Connect 接続。詳細については、「[AWS Direct Connect とは](#)」を参照してください。

同じリージョン内の AWS リソース間のトラフィック

AWS で Secrets Manager と API クライアント間のトラフィックを保護する場合、[AWS PrivateLink](#) を設定して、Secrets Manager API エンドポイントにプライベートにアクセスするようにします。

暗号化キーの管理

Secrets Manager が保護されたシークレットデータの新しいバージョンを暗号化する必要がある場合、Secrets Manager は AWS KMS にリクエストを送信し、KMS キーから新しいデータキーを生成します。Secrets Manager は、このデータキーを [エンベロープ暗号化](#) に使用します。Secrets Manager は、暗号化されたシークレットを使用して、暗号化されたデータキーを保存します。シークレットを復号する必要がある場合、Secrets Manager は AWS KMS にデータキーを復号するよう求めます。Secrets Manager は、復号されたデータキーを使用して、暗号化されたシークレットを復号します。Secrets Manager では、データキーは暗号化されていない形式で保存されることはなく、キーはメモリから速やかに削除されます。詳細については、「[the section called “シークレット暗号化と復号”](#)」を参照してください。

でのシークレットの暗号化と復号 AWS Secrets Manager

Secrets Manager は AWS KMS [、キーとデータキーによるエンベロープ暗号化](#) を使用して、各シークレット値を保護します。シークレットのシークレット値が変更されるたびに、Secrets Manager は新しいデータキーをリクエスト AWS KMS して保護します。データキーは、KMS キーの下で暗号化され、シークレットのメタデータに保存されます。シークレットを復号するために、Secrets Manager はまず [の KMS キーを使用して暗号化されたデータキーを復号します AWS KMS](#)。

Secrets Manager は、シークレット値を直接暗号化するとき、KMS キーを使用しません。代わりに、KMS キーを使用して 256 ビット Advanced Encryption Standard (AES) 対称型 [データキー](#) を生成し、このデータキーを使用してシークレット値を暗号化します。Secrets Manager は、プレーンテキストのデータキーを使用して [の外部でシークレット値を暗号化し AWS KMS](#)、メモリから削除します。また、データキーの暗号化されたコピーを、シークレットのメタデータに保存します。

トピック

- [AWS KMS キーの選択](#)
- [暗号化されるもの](#)
- [プロセスの暗号化と復号](#)
- [KMS キーのアクセス許可](#)
- [Secrets Manager による KMS キーの使用方法](#)
- [キーポリシー AWS マネージドキー \(aws/secretsmanager\)](#)
- [Secrets Manager の暗号化コンテキスト](#)
- [Secrets Manager と のインタラクションをモニタリングする AWS KMS](#)

AWS KMS キーの選択

シークレットを作成するときは、AWS アカウント およびリージョンで任意の対称暗号化カスタマーマネージドキーを選択するか、Secrets Manager () AWS マネージドキー のを使用できません `aws/secretsmanager`。を選択し AWS マネージドキー `aws/secretsmanager`、まだ存在しない場合は、Secrets Manager によって作成され、シークレットに関連付けられます。アカウントの各シークレットに、同じ KMS キーまたは異なる KMS キーを使用できます。異なる KMS キーを使用して、シークレットのグループのキーにカスタムアクセス許可を設定したり、それらのキーの特定の操作を監査したりする場合があります。Secrets Manager は、[対称型 暗号化 KMS キー](#)のみをサポートします。[外部キーストア](#)で KMS キーを使用する場合、要求が AWSの外に移動する必要があるため、KMS キーの暗号化操作に時間がかかり、信頼性と耐久性が低下する可能性があります。

シークレットの暗号化キーの変更の詳細については、「[the section called “シークレットの暗号化キーを変更する”](#)」を参照してください。

暗号化キーを変更すると、Secrets Manager は `AWSCURRENT`、`AWSPENDING`、および `AWSPREVIOUS`のバージョンを新しいキーで再暗号化します。シークレットからロックアウトされないように、Secrets Manager は既存のすべてのバージョンを以前のキーで暗号化します。つまり `AWSCURRENT`、前のキーまたは新しいキーを使用して、`AWSPENDING`、および `AWSPREVIOUS`のバージョンを復号できます。

新しい暗号化キー `AWSCURRENT`でのみ復号できるようにするには、新しいキーを使用してシークレットの新しいバージョンを作成します。次に、`AWSCURRENT`シークレットバージョンを復号するには、新しいキーに対するアクセス許可が必要です。

へのアクセス許可を拒否し、AWS マネージドキー `aws/secretsmanager`シークレットがカスタマーマネージドキーで暗号化されるように要求できます。詳細については、「[the section called “例: シークレットを暗号化するための特定の AWS KMS キーを拒否する”](#)」を参照してください。

シークレットに関連付けられている KMS キーを検索するには、コンソールでシークレットを表示するか、[ListSecrets](#) または を呼び出します [DescribeSecret](#)。シークレットが Secrets Manager (`aws/secretsmanager`) AWS マネージドキー のに関連付けられている場合、これらのオペレーションは KMS キー識別子を返しません。

暗号化されるもの

Secrets Manager ではシークレット値を暗号化しますが、次の値は暗号化しません。

- シークレットの名前と説明
- ローテーション設定

- シークレットに関連付けられた KMS キーの ARN
- アタッチされた AWS タグ

プロセスの暗号化と復号

シークレットのシークレット値を暗号化するには、Secrets Manager は次のプロセスを使用します。

1. Secrets Manager は、AWS KMS [GenerateDataKey](#) シークレットの KMS キーの ID と 256 ビット AES 対称キーのリクエストを使用して オペレーションを呼び出します。AWS KMS は、プレーンテキストのデータキーと、KMS キーで暗号化されたそのデータキーのコピーを返します。
2. Secrets Manager は、プレーンテキストのデータキーと Advanced Encryption Standard (AES) アルゴリズムを使用して、の外部でシークレット値を暗号化します AWS KMS。次に、使用後可能な限り早く、メモリからプレーンテキストキーが削除されます。
3. Secrets Manager は、暗号化されたデータキーをシークレットのメタデータに保存するので、シークレット値を復号化できません。ただし、Secrets Manager API のいずれも、暗号化されたシークレットまたは暗号化されたデータキーを返しません。

暗号化されたシークレット値を復号するには:

1. Secrets Manager は AWS KMS [Decrypt](#) オペレーションを呼び出し、暗号化されたデータキーを渡します。
2. AWS KMS はシークレットの KMS キーを使用してデータキーを復号します。プレーンテキストのデータキーを返します。
3. Secrets Manager は、プレーンテキストのデータキーを使用してシークレット値を復号化します。次に、可能な限り早く、メモリからデータキーが削除されます。

KMS キーのアクセス許可

Secrets Manager が暗号化オペレーションで KMS キーを使用する場合、シークレット値をアクセスまたは更新しているユーザーの代わりに動作します。IAM ポリシーまたはキーポリシーでアクセス許可を付与できます。次の Secrets Manager オペレーションには AWS KMS アクセス許可が必要です。

- [CreateSecret](#)
- [GetSecretValue](#)

- [PutSecretValue](#)
- [UpdateSecret](#)
- [ReplicateSecretToRegions](#)

Secrets Manager から送信されるリクエストにのみ KMS キーを使用できるようにするには、アクセス許可ポリシーで、`secretsmanager.<Region>.amazonaws.com`値で [kms:ViaService condition キー](#)を使用できます。

また、暗号化オペレーションに KMS キーを使用する条件として、[暗号化コンテキスト](#)でキーまたは値を使用することもできます。例えば、IAM またはキーポリシードキュメントで[文字列条件演算子](#)を使用したり、[制約許可](#)を与られます。KMS キー付与の伝播には、最大 5 分かかります。詳細については、「」を参照してください[CreateGrant](#)。

Secrets Manager による KMS キーの使用方法

Secrets Manager は、KMS キーを使用して次の AWS KMS オペレーションを呼び出します。

GenerateDataKey

Secrets Manager は、AWS KMS [GenerateDataKey](#) 次の Secrets Manager オペレーションに 응답して オペレーションを呼び出します。

- [CreateSecret](#) – 新しいシークレットにシークレット値が含まれている場合、Secrets Manager は新しいデータキーをリクエストして暗号化します。
- [PutSecretValue](#) – Secrets Manager は、指定されたシークレット値を暗号化するための新しいデータキーをリクエストします。
- [ReplicateSecretToRegions](#) – レプリケートされたシークレットを暗号化するために、Secrets Manager はレプリカリージョンの KMS キーのデータキーをリクエストします。
- [UpdateSecret](#) – シークレット値または KMS キーを変更すると、Secrets Manager は新しいデータキーをリクエストして新しいシークレット値を暗号化します。

シークレット値は変更されないため [GenerateDataKey](#)、[RotateSecret](#) オペレーションは を呼び出しません。ただし、[RotateSecret](#) がシークレット値を変更する Lambda 関数を呼び出す場合、[PutSecretValue](#) オペレーションの呼び出しにより、[GenerateDataKey](#) リクエストがトリガーされます。

Decrypt

Secrets Manager は、次の Secrets Manager オペレーションに 응답して [Decrypt](#) オペレーションを呼び出します。

- [GetSecretValue](#) および [BatchGetSecretValue](#) - Secrets Manager は、シークレット値を復号してから呼び出し元に返します。暗号化されたシークレット値を復号するために、Secrets Manager AWS KMS は [Decrypt](#) オペレーションを呼び出して、シークレット内の暗号化されたデータキーを復号します。次に、プレーンテキストのデータキーを使って、暗号化されたシークレット値を復号します。バッチコマンドの場合、Secrets Manager は復号化されたキーを再利用できるため、すべての呼び出しが Decrypt リクエストにつながるわけではありません。
- [PutSecretValue](#) および [UpdateSecret](#) - ほとんどの PutSecretValue および UpdateSecret リクエストは Decrypt オペレーションをトリガーしません。ただし、PutSecretValue または UpdateSecret 要求がシークレットの既存のバージョンでシークレット値を変更しようとする、Secrets Manager は既存のシークレット値を復号化し、リクエスト内のシークレット値と比較して、それらが同じであることを確認します。このアクションは、Secrets Manager の操作が冪等であることを保証します。暗号化されたシークレット値を復号するために、Secrets Manager AWS KMS は [Decrypt](#) オペレーションを呼び出して、シークレット内の暗号化されたデータキーを復号します。次に、プレーンテキストのデータキーを使って、暗号化されたシークレット値を復号します。
- [ReplicateSecretToRegions](#) - Secrets Manager は、レプリカリージョンの KMS キーを使用してシークレット値を再暗号化する前に、最初にプライマリリージョンのシークレット値を復号します。

暗号化

Secrets Manager は、次の Secrets Manager オペレーションに応答して [Encrypt](#) オペレーションを呼び出します。

- [UpdateSecret](#) - KMS キーを変更すると、Secrets Manager は、AWSCURRENT、AWSPREVIOUS および AWSPENDING シークレットバージョンを保護するデータキーを新しいキーで再暗号化します。
- [ReplicateSecretToRegions](#) - Secrets Manager は、レプリカリージョンの KMS キーを使用して、レプリケーション中にデータキーを再暗号化します。

DescribeKey

Secrets Manager は [DescribeKey](#) オペレーションを呼び出して、Secrets Manager コンソールでシークレットを作成または編集するときに KMS キーを一覧表示するかどうかを決定します。

KMS キーへのアクセスの検証

シークレットに関連付けられている KMS キーを確立または変更すると、Secrets Manager は、指定された KMS キーを使用して GenerateDataKey および Decrypt オペレーションを呼び出します。これらの呼び出しでは、呼び出し元に、これらのオペレーションで KMS キーを使用す

るアクセス許可があることが確認されます。Secrets Manager は、これらの操作の結果を破棄します。暗号化操作ではそれらを使用しません。

これらのリクエストの SecretVersionId キーの[暗号化コンテキスト](#)の値は RequestToValidateKeyAccess であるため、この検証呼び出しを識別できます。

Note

以前は、Secrets Manager の検証呼び出しに暗号化コンテキストが含まれていませんでした。古い AWS CloudTrail ログには、暗号化コンテキストのない呼び出しが見つかる場合があります。

キーポリシー AWS マネージドキー (aws/secretsmanager)

Secrets Manager AWS マネージドキー ののキーポリシー (aws/secretsmanager) は、Secrets Manager がユーザーに代わってリクエストを行った場合にのみ、指定されたオペレーションに KMS キーを使用するアクセス許可をユーザーに付与します。このキーポリシーでは、ユーザーが KMS キーを直接使用することは許可されません。

このキーポリシーは、すべての [AWS マネージドキー](#) のポリシーと同様に、サービスによって確立されます。キーポリシーは変更できませんが、いつでも表示できます。詳細については、「[Viewing a key policy](#)」を参照してください。

このキーポリシーのポリシーステートメントには次の効果があります

- アカウントのユーザーが暗号化オペレーションに KMS キーを使用できるようにするのは、リクエストが Secrets Manager から送信される場合のみです。kms:ViaService 条件キーで、この制限を適用します。
- AWS アカウントが KMS キープロパティの表示と許可の取り消しをユーザーに許可する IAM ポリシーを作成できるようにします。
- Secrets Manager は、KMS キーへのアクセスを取得するときは許可を使用しませんが、このポリシーは、ユーザーに代わって Secrets Manager が KMS キーの[許可を作成](#)すること、および、アカウントが KMS キーの使用を Secrets Manager に許可するための[許可を取り消す](#)ことを、許可します。これらは、AWS マネージドキーのポリシードキュメントの標準要素です。

AWS マネージドキー Secrets Manager の例のキーポリシーを次に示します。

```
{
  "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 の暗号化コンテキスト

[暗号化コンテキスト](#) は、一連のキー値のペアおよび任意非シークレットデータを含みます。データを暗号化するリクエストに暗号化コンテキストを含めると、は暗号化コンテキストを暗号化されたデータに AWS KMS 暗号的にバインドします。データを復号するには、同じ暗号化コンテキストに渡す必要があります。

への [GenerateDataKey](#) および [Decrypt](#) リクエストでは AWS KMS、Secrets Manager は、次の例に示すように、シークレットとそのバージョンを識別する 2 つの名前と値のペアを持つ暗号化コンテキストを使用します。名前は変わりませんが、組み合わせられた暗号化コンテキストの値は、シークレット値ごとに異なります。

```
"encryptionContext": {
  "SecretARN": "arn:aws:secretsmanager:us-east-2:111122223333:secret:test-secret-a1b2c3",
  "SecretVersionId": "EXAMPLE1-90ab-cdef-fedc-ba987SECRET1"
}
```


暗号化コンテキストを使用して、[AWS CloudTrail](#)や Amazon CloudWatch Logs などの監査レコードやログでこれらの暗号化オペレーションを識別し、ポリシーや許可での承認の条件として使用できます。

Secrets Manager の暗号化コンテキストは、2 つの名前と値のペアで構成されます。

- SecretArn – 最初の名前と値のペアがシークレットを識別します。キーは、SecretARN です。値はシークレットの Amazon リソースネーム (ARN) です。

```
"SecretARN": "ARN of an Secrets Manager secret"
```

例えば、シークレットの ARN が `arn:aws:secretsmanager:us-east-2:111122223333:secret:test-secret-a1b2c3` である場合、暗号化コンテキストには次のペアが含まれます。

```
"SecretARN": "arn:aws:secretsmanager:us-east-2:111122223333:secret:test-secret-a1b2c3"
```

- SecretVersionId – 2 番目の名前と値のペアは、シークレットのバージョンを識別します。キーは、SecretVersionId です。値は、バージョン ID です。

```
"SecretVersionId": "<version-id>"
```

例えば、シークレットのバージョン ID が `EXAMPLE1-90ab-cdef-fedc-ba987SECRET1` である場合、暗号化コンテキストには次のペアが含まれます。

```
"SecretVersionId": "EXAMPLE1-90ab-cdef-fedc-ba987SECRET1"
```

シークレットの KMS キーを確立または変更すると、Secrets Manager は [GenerateDataKey](#) および [Decrypt](#) リクエストを AWS KMS に送信して、呼び出し元がこれらのオペレーションに KMS キーを使用するアクセス許可を持っていることを確認します。レスポンスは廃棄され、シークレット値では使用されません。

これらの検証リクエストでは、SecretARN の値がシークレットの実際の ARN となりますが、SecretVersionId 値は、次の暗号化コンテキストの例に示すように RequestToValidateKeyAccess になります。この特殊な値は、ログと監査証跡で検証リクエストを識別するうえで役立ちます。

```
"encryptionContext": {
  "SecretARN": "arn:aws:secretsmanager:us-east-2:111122223333:secret:test-secret-
a1b2c3",
  "SecretVersionId": "RequestToValidateKeyAccess"
}
```

Note

以前は、Secrets Manager の検証要求に暗号化コンテキストが含まれていませんでした。古い AWS CloudTrail ログには、暗号化コンテキストのない呼び出しが見つかる場合があります。

Secrets Manager と のインタラクションをモニタリングする AWS KMS

AWS CloudTrail と Amazon CloudWatch Logs を使用して、Secrets Manager が AWS KMS ユーザーに代わって に送信するリクエストを追跡できます。シークレットの使用のモニタリングについては、「[シークレットをモニタリングする](#)」を参照してください。

GenerateDataKey

シークレットでシークレット値を作成または変更すると、Secrets Manager はシークレットの KMS キー AWS KMS を指定する [GenerateDataKey](#) リクエストを に送信します。

GenerateDataKey 演算を記録するイベントは、次のようなサンプルイベントになります。リクエストは `secretsmanager.amazonaws.com` によって起動されます。このパラメータには、シークレットの KMS キーの Amazon リソースネーム (ARN)、256 ビットキーを要求するキー識別子、およびシークレットとバージョンを識別する [暗号化コンテキスト](#) が含まれます。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AROAIQDTESTANDEXAMPLE: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

シークレットのシークレット値を取得または変更すると、Secrets Manager は [Decrypt](#) リクエストを送信 AWS KMS して、暗号化されたデータキーを復号します。バッチコマンドの場合、Secrets Manager は復号化されたキーを再利用できるため、すべての呼び出しが Decrypt リクエストにつながるわけではありません。

Decrypt 演算を記録するイベントは、次のようなサンプルイベントになります。ユーザーは、テーブルにアクセスする AWS アカウントのプリンシパルです。パラメータには、暗号化されたテーブルキー (暗号文 blob として) と、テーブルと AWS account. AWS KMS を識別する [暗号化コンテキスト](#)が含まれます。は、暗号文から KMS キーの ID を取得します。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AROAIQDTESTANDEXAMPLE: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"
}

```

暗号化

シークレットに関連付けられた KMS キーを変更すると、Secrets Manager は [Encrypt](#) リクエストを送信 AWS KMS して AWSCURRENT、AWSPREVIOUS、およびシークレットバージョンを新しいキーで再暗号化します。シークレットを別のリージョンにレプリケートすると、Secrets Manager は [Encrypt](#) リクエストも AWS KMS に送信します。

Encrypt 演算を記録するイベントは、次のようなサンプルイベントになります。ユーザーは、テーブルにアクセスする AWS アカウントのプリンシパルです。

```

{
  "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"
}
```

AWS Secrets Manager でのインフラストラクチャセキュリティ

マネージドサービスである AWS Secrets Manager は AWS グローバルネットワークセキュリティで保護されています。AWSセキュリティサービスと AWS がインフラストラクチャを保護する方法については、「[AWS クラウドセキュリティ](#)」を参照してください。インフラストラクチャセキュリティのベストプラクティスを使用して AWS 環境を設計するには、「セキュリティの柱 - AWS Well-Architected Framework」の「[インフラストラクチャ保護](#)」を参照してください。

ネットワークを介した Secrets Manager へのアクセスには、[TSL を使用した API が発行する AWS](#) が使用されます。これらの Secrets Manager API はネットワークの任意の場所から呼び出すことができます。ただし、Secrets Manager は [リソースベースのアクセスポリシー](#) をサポートしており、それらのポリシーには、ソース IP アドレスに基づく制限を含めることができます。また、Secrets Manager ソースのポリシーを使用して、[特定の仮想プライベートクラウド \(VPC\) エンドポイント](#)、または特定の VPC からのシークレットへのアクセスを制御することもできます。これにより効果的に、AWS ネットワーク内の特定の VPC のみから、特定のシークレットへのネットワークアクセスが分離されることとなります。詳細については、「[VPC エンドポイント](#)」を参照してください。

AWS Secrets Manager での回復性

AWS では、グローバルインフラストラクチャは AWS のリージョンとアベイラビリティゾーンを中心に構築されています。AWSリージョンには、低レイテンシー、高スループット、および冗長性の高いネットワークで接続されている物理的に独立していて、かつ、分離されている複数のアベイラビリティゾーンがあります。アベイラビリティゾーンでは、ゾーン間で中断することなく自動的にフェイルオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも優れた可用性、耐障害性、および拡張性をもたらします。

回復性と災害対策の詳細については、[信頼性の柱 - AWS Well-Architected フレームワーク](#)を参照してください。

AWS リージョンとアベイラビリティゾーンの詳細については、[AWS グローバルインフラストラクチャ](#)を参照してください。

ポスト量子 TLS

Secrets Manager は、Transport Layer Security (TLS) ネットワーク暗号化プロトコル用のハイブリッドポスト量子キー交換オプションをサポートします。この TLS オプションは、Secrets Manager API エンドポイントに接続するときに使用できます。この機能はポスト量子アルゴリズムが標準化される前に提供されているため、これらのキー交換プロトコルの Secrets Manager コールへの影響のテストを開始できます。これらのオプションのハイブリッドポスト量子キー交換機能は、現在使用している TLS 暗号化と同等以上に安全であり、セキュリティ上のさらなる利点をもたらす可能性があります。ただし、現在使用されている従来のキー交換プロトコルと比較して、レイテンシーとスループットに影響します。

潜在的な将来の攻撃から現在暗号化されたデータを保護するために、AWS は量子耐性またはポスト量子アルゴリズムを開発する暗号化コミュニティに参加しています。Secrets Manager エンドポイントにハイブリッドポスト量子キー交換暗号スイートを実装しました。これらのハイブリッド暗号スイートは、従来の要素とポスト量子要素を組み合わせたもので、これにより TLS 接続が少なくとも従来の暗号スイートと同じくらい強力になります。ただし、ハイブリッド暗号スイートのパフォーマンス特性と帯域幅要件は従来のキー交換メカニズムのものとは異なるため、API コールでテストすることをお勧めします。

Secrets Manager は、中国リージョンを除くすべてのリージョンで PQTLS をサポートしています。

ハイブリッドポスト量子 TLS の設定する

1. Maven 依存関係に AWS 共通ランタイムクライアントを追加します。利用可能な最新バージョンを使用することをお勧めします。たとえば、以下のステートメントはバージョン 2.20.0 を追加します。

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>aws-crt-client</artifactId>
  <version>2.20.0</version>
</dependency>
```

2. AWS SDK for Java 2.x をプロジェクトに追加して初期化します。HTTP クライアントでハイブリッドポスト量子暗号スイートを有効にします。

```
SdkAsyncHttpClient awsCrtHttpClient = AwsCrtAsyncHttpClient.builder()
    .postQuantumTlsEnabled(true)
    .build();
```

3. [Secrets Manager 非同期クライアント](#)を作成します。

```
SecretsManagerAsyncClient secretsManagerAsync = SecretsManagerAsyncClient.builder()
    .httpClient(awsCrtHttpClient)
    .build();
```

これで Secrets Manager API オペレーションを呼び出すと、コールはハイブリッドポスト量子 TLS を使用して Secrets Manager エンドポイントに送信されます。

ハイブリッドポスト量子 TLS の使用の詳細については、次を参照してください。

- 「[AWS SDK for Java 2.x 開発者ガイド](#)」と「[AWS SDK for Java 2.x released](#)」(ガリリリースされました)のブログ記事。
- 「[Introducing s2n-tls, a New Open Source TLS Implementation](#)」(新しいオープンソース TLS 実装の のご紹介) および「[Using s2n-tls](#)」(の使用)。
- 米国国立標準技術研究所 (NIST) の [ポスト量子暗号](#)。
- [Hybrid Post-Quantum Key Encapsulation Methods \(PQ KEM\) for Transport Layer Security 1.2 \(TLS\)](#) (Transport Layer Security 1.2 (TLS) 用のハイブリッド Post-Quantum Key Encapsulation Methods (PQ KEM))。

Secrets Manager のポスト量子 TLS は、中国を除くすべての AWS リージョン で利用できます。

トラブルシューティング AWS Secrets Manager

こちらの情報は、Secrets Manager を操作するときに発生する可能性がある問題の、診断や修復に役立ちます。

ローテーションに関連する問題については、[「the section called “におけるローテーションのトラブルシューティング”」](#)を参照してください。

トピック

- [「アクセス拒否」メッセージ](#)
- [一時的なセキュリティ認証情報に「アクセスが拒否されました」と表示される](#)
- [変更がすぐに表示されない。](#)
- [シークレットを作成するときに「非対称 KMS キーを使ってデータキーを生成することはできません」というメッセージが表示される](#)
- [AWS CLI または AWS SDK オペレーションが部分的な ARN からシークレットを見つけられない](#)
- [このシークレットは AWS サービスによって管理されるため、そのサービスを使用して更新する必要があります。](#)

「アクセス拒否」メッセージ

Secrets Manager CreateSecret に対して GetSecretValue や などの API コールを行うときは、そのコールを行うための IAM アクセス許可が必要です。コンソールを使用する場合、コンソールはユーザーに代わって同じ API コールを行うため、IAM アクセス許可も必要です。管理者は、IAM ポリシーを IAM ユーザー、またはユーザーが属するグループにアタッチすることで、アクセス許可を付与できます。これらのアクセス許可を付与するポリシーステートメントに、time-of-day や IP アドレスの制限などの条件が含まれている場合は、リクエストを送信するときにこれらの要件を満たす必要があります。IAM ユーザー、グループ、またはロールのポリシーの表示や修正の詳細については、「IAM ユーザーガイド」の「[Working with Policies](#)」を参照してください。Secrets Manager に必要な許可の詳細については、「[認証とアクセスコントロール](#)」を参照してください。

手動で API リクエストに署名する ([AWS SDK](#) を使用しない) 場合は、正しく [リクエストに署名](#) していることを確認します。

一時的なセキュリティ認証情報に「アクセスが拒否されました」と表示される

リクエストの作成に使用している IAM ユーザーまたはロールに適切なアクセス許可があることを確認します。一時的なセキュリティ認証情報のアクセス許可は IAM ユーザーまたはロールから取得されます。つまり、アクセス許可は IAM ユーザーまたはロールに付与されたものに限定されます。一時的なセキュリティ認証情報のアクセス権限がどのように決定されるかについては、「IAM ユーザーガイド」の「[Controlling Permissions for Temporary Security Credentials](#)」を参照してください。

リクエストが正しく署名されており、そのリクエストの形式が整っていることを確認します。詳細については、選択した SDK の[ツールキットドキュメント](#)、または IAM [ユーザーガイドの「一時的なセキュリティ認証情報を使用して AWS リソースへのアクセスをリクエストする」](#)を参照してください。

一時的な認証情報が失効していないことを確認します。詳細については、「IAM ユーザーガイド」の「[Requesting Temporary Security Credentials](#)」を参照してください。

Secrets Manager に必要な許可の詳細については、「[認証とアクセスコントロール](#)」を参照してください。

変更がすぐに表示されない。

Secrets Manager では、[結果整合性](#)と呼ばれる分散コンピューティングモデルが使用されています。Secrets Manager (または他の AWS サービス) で行った変更は、可能なすべてのエンドポイントから表示されるまでに時間がかかります。この遅延は、サーバー間、レプリケーションゾーン間、世界中のリージョン間でのデータ送信にかかる時間から発生している場合もあります。Secrets Manager ではパフォーマンス向上のためにキャッシュも使用しているため、これが原因で遅延が発生することがあります。変更は、以前にキャッシュされたデータがタイムアウトになるまで反映されない場合があります。

発生する可能性のあるこれらの遅延を考慮して、グローバルなアプリケーションを設計します。また、ある場所で行われた変更が他の場所で直ちに表示されない場合でも、期待どおりに機能するように確かめて下さい。

他の AWS のサービスが結果整合性の影響を受ける方法の詳細については、以下を参照してください。

- 「Amazon Redshift Database デベロッパーガイド」の「[Managing data consistency](#)」

- 「Amazon Simple Storage Service ユーザーガイド」の「[Amazon S3 Data Consistency Model](#)」
- 「[Ensuring Consistency When Using Amazon S3 and Amazon EMR for ETL Workflows](#)」 (AWS ビッグデータブログ)
- 「Amazon EC2 API リファレンス」の「[Amazon EC2 Eventual Consistency](#)」

シークレットを作成するときに「非対称 KMS キーを使ってデータキーを生成することはできません」というメッセージが表示される

Secrets Manager は、シークレットに関連付けられている[対称 KMS キー](#)を使用して、各シークレット値のデータキーを生成します。非対称 KMS キーを使用することはできません。非対称 KMS キーではなく対称暗号化キーが使用されていることを確認します。手順については、「[非対称 KMS キーの識別](#)」を参照してください。

AWS CLI または AWS SDK オペレーションが部分的な ARN からシークレットを見つけられない

多くの場合、Secrets Manager は完全な ARN ではなく ARN の一部からシークレットを見つけることができます。しかし、シークレットの名前がハイフンで終わり、その後 6 つの文字が続く場合、Secrets Manager は ARN の一部からシークレットを見つけられないことがあります。代わりに、完全な ARN またはシークレットの名前を使用することを推奨します。

詳細

Secrets Manager は、シークレット ARN が確実に一意であるようにするのに役立つよう、シークレット名の末尾に 6 つのランダムな文字を含めます。元のシークレットが削除され、同じ名前での新しいシークレットが作成された場合、これらの文字により 2 つのシークレット ARN は異なるものとなります。ARN が異なるため、古いシークレットにアクセスできるユーザーであっても、新しいシークレットへのアクセスを自動的に取得するわけではありません。

Secrets Manager は、リージョン、アカウント、シークレット名、ハイフン、さらに 6 文字を含むシークレットの ARN を次のように構築します。

```
arn:aws:secretsmanager:us-east-2:111122223333:secret:SecretName-abcdef
```

シークレット名がハイフンと 6 文字で終わる場合、ARN の一部のみを使用すると、完全な ARN を指定しているかのように Secrets Manager に表示されます。たとえば、ARN で MySecret-abcdef という名前のシークレットがあるとします。

```
arn:aws:secretsmanager:us-east-2:111122223333:secret:MySecret-abcdef-nutBrk
```

シークレット ARN の一部のみを使用する次のオペレーションを呼び出すと、Secrets Manager がシークレットを見つけられないことがあります。

```
$ aws secretsmanager describe-secret --secret-id arn:aws:secretsmanager:us-east-2:111122223333:secret:MySecret-abcdef
```

このシークレットは AWS サービスによって管理されるため、そのサービスを使用して更新する必要があります。

シークレットを変更しようとしているときにこのメッセージが表示された場合、シークレットはメッセージに記載されている管理サービスを使用してのみ更新できます。詳細については、「[マネージドシークレット](#)」を参照してください。

シークレットを管理しているユーザーを特定するには、シークレット名を確認します。他のサービスによって管理されるシークレットには、そのサービスの ID がプレフィックスとして付けられます。または、で AWS CLI `describe-secret` を呼び出し、フィールド `OwningService` を確認します。

AWS Secrets Manager のクォータ

Secrets Manager の読み取り API は TPS クォータが高く、あまり呼び出されないコントロールプレーン API は TPS クォータが低くなります。10 分に 1 回以上の持続頻度で PutSecretValue または UpdateSecret を呼び出すことは避けることが推奨されます。PutSecretValue または UpdateSecret を呼び出してシークレット値を更新すると、Secrets Manager はシークレットの新しいバージョンを作成します。Secrets Manager は、ラベルのないバージョンが 100 を超えると削除しますが、24 時間以内に作成されたバージョンは削除しません。10 分に 1 回以上の頻度でシークレット値を更新すると、Secrets Manager が削除した数よりも多くバージョンが作成され、シークレットバージョンのクォータに達します。

お使いのアカウントで複数のリージョンを運用できます。各クォータは各リージョンに固有です。

1 つの AWS アカウント のアプリケーションが、異なるアカウントの所有するシークレットを使用することを、クロスアカウントリクエストと呼びます。クロスアカウントリクエストでは、Secrets Manager は、シークレットを所有するアカウントではなく、リクエストを行うアイデンティティのアカウントをスロットリングします。例えば、アカウント A のアイデンティティがアカウント B のシークレットを使用する場合、このシークレットの使用は、アカウント A のクォータにのみ適用されます。

Secrets Manager のクォータ

名前	デフォルト	引き上げ可能	説明
DeleteResourcePolicy、GetResourcePolicy、PutResourcePolicy および ValidateResourcePolicy API リクエストの合計レート	サポートされている各リージョン: 50/秒	はい	DeleteResourcePolicy、GetResourcePolicy、PutResourcePolicy および ValidateResourcePolicy API リクエストの合計の 1 秒あたり

名前	デフォルト	引き上げ可能	説明
			の最大トランザクション数。
DescribeSecret および GetSecretValue API リクエストの合計レート	サポートされている各リージョン: 10,000/秒	はい	DescribeSecret および GetSecretValue API リクエストの合計の 1 秒あたりの最大トランザクション数。
PutSecretValue、RemoveRegionsFromReplication、ReplicateSecretToRegion、StopReplicationToReplica、UpdateSecret および UpdateSecretVersionStage API リクエストの合計レート	サポートされている各リージョン: 50/秒	はい	PutSecretValue、RemoveRegionsFromReplication、ReplicateSecretToRegion、StopReplicationToReplica、UpdateSecret および UpdateSecretVersionStage API リクエストの合計の 1 秒あたりの最大トランザクション数。
RestoreSecret API リクエストの合計レート	サポートされている各リージョン: 50/秒	はい	RestoreSecret API リクエストの 1 秒あたりの最大トランザクション数。
RotateSecret および CancelRotateSecret API リクエストの合計レート	サポートされている各リージョン: 50/秒	はい	RotateSecret および CancelRotateSecret API リクエストの合計の 1 秒あたりの最大トランザクション数。

名前	デフォルト	引き上げ可能	説明
TagResource および UntagResource API リクエストの合計レート	サポートされている各リージョン: 50/秒	いいえ	TagResource および UntagResource API リクエストの合計の 1 秒あたりの最大トランザクション数。
BatchGetSecretValue API リクエストのレート	サポートされている各リージョン: 100/秒	いいえ	BatchGetSecretValue API リクエストの 1 秒あたりの最大トランザクション数。
CreateSecret API リクエストのレート	サポートされている各リージョン: 50/秒	いいえ	CreateSecret API リクエストの 1 秒あたりの最大トランザクション数。
DeleteSecret API リクエストのレート	サポートされている各リージョン: 50/秒	いいえ	DeleteSecret API リクエストの 1 秒あたりの最大トランザクション数。
GetRandomPassword API リクエストのレート	サポートされている各リージョン: 50/秒	いいえ	GetRandomPassword API リクエストの 1 秒あたりの最大トランザクション数。
ListSecretVersionIds API リクエストのリスト	サポートされている各リージョン: 50/秒	いいえ	ListSecretVersionIds API リクエストの 1 秒あたりの最大トランザクション数

名前	デフォルト	引き上げ可能	説明
ListSecrets API リクエストのレート	サポートされている各リージョン: 100/秒	はい	ListSecrets API リクエストの 1 秒あたりの最大トランザクション数
リソースベースのポリシーの長さ	サポートされている各リージョン: 20,480	はい	シークレットにアタッチされているリソースベースのアクセス権限ポリシーの最大文字数。
シークレット値のサイズ	サポートされている各リージョン: 65,536 バイト	はい	暗号化されたシークレット値の最大サイズ。シークレット値が文字列の場合、これはシークレット値で許可される文字数です。
シークレット	サポートされている各リージョン: 500,000	はい	この AWS アカウントの各 AWS リージョンでのシークレットの最大数。
シークレットのすべてのバージョンにアタッチされたステージングラベル	サポートされている各リージョン: 20	はい	シークレットのすべてのバージョンにアタッチされたステージングラベルの最大数。
シークレットあたりのバージョン	サポートされている各リージョン: 100	はい	1 つのシークレットのバージョンの最大数。

アプリケーションへの再試行を追加する

AWS クライアントは、クライアント側で予期していなかった問題が発生したときに、Secrets Manager への呼び出しが失敗したと判断することがあります。あるいは、Secrets Manager によるレート制限が原因で、呼び出しが失敗する場合があります。API リクエストクォータを超えると、Secrets Manager はリクエストをスロットルします。それ以外の場合は有効なリクエストを拒否し、throttling というエラーを出力します。どちらの種類の実行失敗に関しても、短い待機時間後に、呼び出しを再試行することをお勧めします。これは、[バックオフと再試行の戦略](#)と呼ばれます。

以下のようなエラーが発生した場合は、アプリケーションコードに再試行の処理を追加します。

一時的なエラーおよび例外

- RequestTimeout
- RequestTimeoutException
- PriorRequestNotComplete
- ConnectionError
- HTTPClientError

サービス側のスロットリングと制限のエラーおよび例外

- Throttling
- ThrottlingException
- ThrottledException
- RequestThrottledException
- TooManyRequestsException
- ProvisionedThroughputExceededException
- TransactionInProgressException
- RequestLimitExceeded
- BandwidthLimitExceeded
- LimitExceededException
- RequestThrottled
- SlowDown

再試行、エクスポネンシャルバックオフ、ジッターに関する詳細およびコード例については、次のリソースを参照してください。

- [エクスポネンシャルバックオフとジッター](#)
- [ジッターを伴うタイムアウト、再試行、およびバックオフ](#)
- [AWS でのエラー再試行とエクスポネンシャルバックオフ](#)

ドキュメント履歴

次の表は、の前のリリース以降のドキュメントの重要な変更点を示しています AWS Secrets Manager。このドキュメントの更新に関する通知を受け取るには、RSS フィードにサブスクライブできます。

変更	説明	日付
Secrets Manager の AWS マネージドポリシーへの変更	SecretsManagerRead Write 管理ポリシーに アクセスredshift-serverless 許可が含まれるようになりました。詳細については、「の AWS 管理ポリシー AWS Secrets Manager 」を参照してください。	2024 年 3 月 12 日

以前の更新

次の表は、2024 年 2 月以前の AWS Secrets Manager ユーザーガイドの各リリースにおける重要な変更点を示しています。

変更	説明	日付
一般提供	これは Secrets Manager の最初のパブリックリリースです。	2018 年 4 月 4 日

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。