



User Guide

AWS Secrets Manager



AWS Secrets Manager: User Guide

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

Die Handelsmarken und Handelsaufmachung von Amazon dürfen nicht in einer Weise in Verbindung mit nicht von Amazon stammenden Produkten oder Services verwendet werden, durch die Kunden irregeführt werden könnten oder Amazon in schlechtem Licht dargestellt oder diskreditiert werden könnte. Alle anderen Handelsmarken, die nicht Eigentum von Amazon sind, gehören den jeweiligen Besitzern, die möglicherweise zu Amazon gehören oder nicht, mit Amazon verbunden sind oder von Amazon gesponsert werden.

Table of Contents

Was ist Secrets Manager?	1
Erste Schritte mit Secrets Manager	1
Einhaltung von Standards	2
Preisgestaltung	2
Zugriff auf Secrets Manager	4
Secrets-Manager-Konsole	4
Befehlszeilen-Tools	4
AWS SDKs	5
HTTPS-Query-API	5
Secrets-Manager-Endpunkte	6
Was ist in einem Geheimnis	11
Metadaten	11
Geheime Versionen	12
Tutorials	14
Amazon CodeGuru Reviewer	14
Fest codierte Secrets ersetzen	14
Schritt 1: Das Secret erstellen	15
Schritt 2: Ihren Code aktualisieren	17
Schritt 3: Das Secret aktualisieren	18
Nächste Schritte	18
Ersetzen von fest codierten DB-Anmeldeinformationen	19
Schritt 1: Das Secret erstellen	20
Schritt 2: Ihren Code aktualisieren	21
Schritt 3: Drehen des Geheimnisses	22
Nächste Schritte	23
Drehung für wechselnde Benutzer	24
Berechtigungen	25
Voraussetzungen	25
Schritt 1: Erstellen eines Amazon-RDS-Datenbankbenutzers	29
Schritt 2: Erstellen Sie ein Geheimnis für die Benutzer-Anmeldeinformationen	32
Schritt 3: Testen des gedrehten Geheimnisses	33
Schritt 4: Bereinigen von Ressourcen	34
Nächste Schritte	35
Einzelbenutzer-Drehung	35

Berechtigungen	36
Voraussetzungen	36
Schritt 1: Erstellen eines Amazon-RDS-Datenbankbenutzers	36
Schritt 2: Erstellen eines Secrets für die Benutzer-Anmeldeinformationen	37
Schritt 3: Testen des rotierten Passworts	38
Schritt 4: Bereinigen von Ressourcen	39
Nächste Schritte	39
Authentifizierung und Zugriffskontrolle	40
Administratorberechtigungen für den Secrets Manager	40
Berechtigungen für den Zugriff auf Secrets	40
Berechtigungen für Lambda Rotationsfunktionen	41
Berechtigungen für die Verschlüsselung	41
Berechtigungen für die Replikation	41
Anhängen einer Berechtigungsrichtlinie an eine Identität	41
Zuordnen einer Berechtigungsrichtlinie zu einem Secret	42
AWS CLI	43
AWS SDK	44
AWS verwaltete Richtlinien	45
SecretsManagerReadWrite	45
Richtlinienaktualisierungen	47
Bestimmen, wer Berechtigungen für Ihre -Secrets hat	49
Kontoubergreifender Zugriff	50
Lokaler Zugriff	53
Beispiele für Richtlinien für Berechtigungen	53
Beispiel: Berechtigung zum Abrufen von einzelnen Secret-Werten	54
Beispiel: Erlaubnis, einzelne Geheimnisse zu lesen und zu beschreiben	56
Beispiel: Berechtigung zum Abrufen einer Gruppe geheimer Werte in einem Batch	56
Beispiel: Platzhalter	57
Beispiel: Berechtigung zum Erstellen von Secrets	59
Beispiel: Verweigern Sie einen bestimmten AWS KMS Schlüssel zur Verschlüsselung von Geheimnissen	60
Beispiel: Berechtigungen und VPCs	61
Beispiel: Steuern des Zugriffs auf Secrets mit Tags	63
Beispiel: Beschränken Sie den Zugriff auf Identitäten mit Tags, die mit den Tags der Secrets übereinstimmen	64
Beispiel: Service-Prinzipal	65

Berechtigungsreferenz	66
Secrets-Manager-Aktionen	67
Secrets-Manager-Ressourcen	88
Bedingungsschlüssel	88
BlockPublicPolicy-Bedingung	91
Bedingungen für die IP-Adresse	92
VPC-Endpunktbedingungen	92
Erstellen und Verwalten von Secrets	94
Erstellen eines Datenbank-Secrets	94
AWS CLI	97
AWS SDK	97
JSON-Struktur eines Secrets	98
Secret-Struktur von Amazon RDS Db2	98
Amazon RDS MariaDB – Secret-Struktur	99
Secret-Struktur von Amazon RDS und Amazon Aurora MySQL	99
Amazon RDS Oracle – Secret-Struktur	100
Secret-Struktur von Amazon RDS und Amazon Aurora PostgreSQL	100
Amazon RDS Microsoft SQLServer – Secret-Struktur	101
Amazon DocumentDB – Secret-Struktur	102
Amazon Redshift – Secret-Struktur	102
Geheime Struktur von Amazon Redshift Serverless	103
ElastiCache Geheime Struktur von Amazon	103
Geheime Active Directory-Strukturen	104
Ein Secret erstellen	105
AWS CLI	108
AWS SDK	109
Aktualisieren eines Geheimniswertes	109
AWS CLI	110
AWS-SDK	110
Generieren Sie ein Passwort mit Secrets Manager	111
Ein Geheimnis auf eine frühere Version zurücksetzen	111
Ändern des Verschlüsselungsschlüssels für ein Secret	112
AWS CLI	113
Ändern eines Secrets	114
AWS CLI	116
AWS SDK	116

Finden von Geheimnissen	116
AWS CLI	118
AWS SDK	118
Löschen eines Secrets	119
AWS CLI	120
AWS SDK	121
Wiederherstellen eines Secrets	121
AWS CLI	122
AWS SDK	122
-Secrets markieren	123
AWS CLI	124
AWS-SDK	124
Replizieren Sie Geheimnisse regionsübergreifend	125
AWS CLI	127
AWS SDK	127
Hochstufen eines Secret-Replikats zu einem eigenständigen Secret	127
AWS CLI	128
AWS-SDK	129
Replizierung verhindern	129
Fehlerbehebung bei der Replikation	130
Ein Secret mit demselben Namen ist in der ausgewählten Region bereits vorhanden.	130
Keine Berechtigungen für den KMS-Schlüssel verfügbar, um die Replikation abzuschließen	131
Der KMS-Schlüssel wurde deaktiviert oder wurde nicht gefunden	131
Sie haben die Region, in der die Replikation stattfindet, nicht aktiviert.	131
Holen Sie sich Geheimnisse	132
Java	132
Java mit clientseitigem Caching	133
JDBC-Verbindung mit geheimen Anmeldeinformationen	140
AWS Java-SDK	150
Python	152
Python mit clientseitigem Caching	152
AWS Python-SDK	158
Holen Sie sich einen Stapel geheimer Werte	159
.NET	161
.NET mit clientseitigem Caching	161

.NET AWS SDK	168
Go	171
Entscheiden Sie sich für clientseitiges Caching	172
Go AWS SDK	176
C++	177
JavaScript	178
Kotlin	180
PHP	180
Ruby	181
Rust	182
AWS CLI	183
Holen Sie sich eine Gruppe von Geheimnissen in einem Batch mit dem AWS CLI	183
AWS Konsole	184
AWS Batch	185
AWS CloudFormation	185
Amazon EKS	186
Schritt 1: Einrichten der Zugriffssteuerung	187
Schritt 2: Installieren und konfigurieren Sie den ASCP	188
Schritt 3: Identifizieren Sie, welche Secrets bereitgestellt werden sollen	189
Schritt 4: Mounten Sie die Geheimnisse als Dateien im Amazon EKS-Pod	192
Fehlerbehebung	193
SecretProviderClass	193
GitHub Jobs	197
Voraussetzungen	197
Verwendung	198
Benennung von Umgebungsvariablen	199
Beispiele	200
AWS IoT Greengrass	203
AWS Lambda	203
Umgebungsvariablen	206
Parameter Store	208
Rotieren von -Geheimnissen	209
Verwaltete Rotation	209
Rotation durch Lambda-Funktion	211
Automatische Rotierung für Datenbank-Secrets (Konsole)	212
Automatische Rotation für Nicht-Datenbankgeheimnisse (Konsole)	216

Automatische Drehung (AWS CLI)	221
Strategien zur Rotation von Lambda-Funktionen	225
Lambda-Rotationsfunktionen	227
Rotationsfunktionsvorlagen	231
Berechtigungen für Rotation	238
Netzwerkzugriff für die Lambda-Rotationsfunktion	242
Fehlerbehebung bei der -Rotation	244
Secret sofort drehen	253
AWS CLI	253
Rotationspläne	253
Rate-Ausdrücke	254
Cron-Ausdrücke	254
Finde Geheimnisse, die nicht rotiert werden	260
Automatische Rotation abbrechen	261
Verwaltete Geheimnisse	262
Dienste, die Geheimnisse verwenden	264
App Runner	266
AWS -App2Container	266
AWS AppConfig	266
Amazon AppFlow	267
AWS AppSync	267
Amazon Athena	267
Amazon Aurora	267
AWS CodeBuild	268
Amazon Data Firehose	268
AWS DataSync	269
Amazon DataZone	269
AWS Direct Connect	269
AWS Directory Service	270
Amazon DocumentDB	270
AWS Elastic Beanstalk	270
Amazon Elastic Container Registry	271
Amazon Elastic Container Service	271
Amazon ElastiCache	272
AWS Elemental Live	272
AWS Elemental MediaConnect	272

AWS Elemental MediaConvert	273
AWS Elemental MediaLive	273
AWS Elemental MediaPackage	273
AWS Elemental MediaTailor	273
Amazon EMR	274
EMR in EC2	274
EMR Serverless	274
Amazon EventBridge	275
Amazon FSx	275
AWS Glue DataBrew	275
AWS Glue Studio	276
AWS IoT SiteWise	276
Amazon Kendra	276
Amazon Kinesis Video Streams	276
AWS Launch Wizard	277
Amazon Lookout für Metrics	277
Amazon Managed Grafana	277
AWS Managed Services	278
Amazon Managed Streaming für Apache Kafka	278
Amazon Managed Workflows für Apache Airflow	278
AWS Marketplace	278
AWS Migration Hub	279
AWS Panorama	279
AWS ParallelCluster	279
Amazon Q	280
AWS OpsWorks for Chef Automate	280
Amazon QuickSight	280
Amazon RDS	281
Amazon-Redshift	281
Amazon-Redshift-Abfrage-Editor v2	282
Amazon SageMaker	282
AWS SCT	283
AWS Toolkit for JetBrains	283
AWS Transfer Family	283
AWS Wickr	284
VPC-Endpoint	285

Gemeinsam genutzte Subnetze	286
AWS CloudFormation	287
Ein Secret erstellen	288
JSON	288
YAML	289
Erstellen Sie ein Geheimnis mit Amazon-RDS-Anmeldeinformationen mit automatischer Drehung	289
Erstellen eines Secrets mit Amazon-Redshift-Anmeldeinformationen	289
Erstellen eines Secrets mit Amazon-DocumentDB-Anmeldeinformationen	289
JSON	290
YAML	294
So verwendet Secrets Manager AWS CloudFormation	297
AWS CDK	298
Überwachung von Geheimnissen	299
Loggen Sie sich mit AWS CloudTrail	299
AWS CLI	300
CloudTrail Einträge	301
Überwachen Sie mit CloudWatch	306
CloudWatch Alarme	307
Kombiniere Secrets Manager Manager-Ereignisse mit EventBridge	308
Alle Änderungen mit einem bestimmten Secret abgleichen	308
Ereignisse abgleichen, wenn ein Secret-Wert rotiert	309
Überwachung von Geheimnissen, die zum Löschen eingeplant sind	309
Schritt 1: Konfigurieren Sie die Übertragung der CloudTrail Protokolldatei in CloudWatch Logs	310
Schritt 2: Erstellen Sie den CloudWatch Alarm	311
Schritt 3: Testen Sie den CloudWatch Alarm	312
Überwachen Sie geheime Daten auf Einhaltung der Vorschriften	312
Secrets Manager Manager-Kosten überwachen	313
Compliance-Validierung	314
Konformitätsstandards	315
Sicherheit in Secrets Manager	317
Reduzieren von Risiken durch die Verwendung der AWS CLI zur Speicherung Ihrer AWS Secrets Manager-Secrets	318
Datenschutz in Secrets Manager	320
Verschlüsselung im Ruhezustand	321

Verschlüsselung während der Übertragung	321
Datenschutz für den Datenverkehr zwischen Netzwerken	322
Verwaltung von Verschlüsselungsschlüsseln	322
Ver- und Entschlüsselung von Secrets	322
Auswahl eines Schlüssels AWS KMS	323
Was ist verschlüsselt?	324
Ver- und Entschlüsselungsprozesse	325
Berechtigungen für den KMS-Schlüssel	325
Wie Secrets Manager den KMS-Schlüssel verwendet	326
Schlüsselrichtlinie des Von AWS verwalteter Schlüssel (aws/secretsmanager)	328
Verschlüsselungskontext für Secrets Manager	330
Überwachen Sie die Secrets Manager Manager-Interaktion mit AWS KMS	332
Sicherheit der Infrastruktur	337
Ausfallsicherheit	337
Post-Quantum-TLS	337
Fehlerbehebung	340
Meldungen mit dem Hinweis „Zugriff verweigert“	340
„Access denied“ (Zugriff verweigert) für temporäre Sicherheitsanmeldeinformationen	341
Änderungen, die ich vornehme, sind nicht immer direkt sichtbar.	341
„Cannot generate a data key with an asymmetric KMS key“ (Kann keinen Datenschlüssel mit einem asymmetrischen KMS-Schlüssel generieren) beim Erstellen eines Geheimnisses	342
Eine AWS CLI oder AWS SDK-Operation kann mein Geheimnis nicht aus einem partiellen ARN finden	342
Dieses Geheimnis wird von einem AWS Dienst verwaltet, und Sie müssen diesen Dienst verwenden, um es zu aktualisieren.	343
Kontingente	344
Secrets-Manager-Kontingente	344
Hinzufügen von Wiederholungen zu Ihrer Anwendung	348
Dokumentverlauf	350
Frühere Aktualisierungen	350
.....	cccli

Was ist AWS Secrets Manager?

AWS Secrets Manager hilft Ihnen dabei, Datenbankanmeldedaten, Anwendungsanmeldedaten, OAuth-Token, API-Schlüssel und andere Geheimnisse während ihres gesamten Lebenszyklus zu verwalten, abzurufen und zu rotieren. Viele AWS Dienste speichern und verwenden Geheimnisse in Secrets Manager.

Secrets Manager hilft Ihnen, Ihre Sicherheitslage zu verbessern, da hartcodierte Anmeldeinformationen im Quellcode der Anwendung nicht mehr innerhalb oder mit der Anwendung gespeichert werden. Das Speichern der Anmeldeinformationen im Secrets Manager trägt dazu bei, eine mögliche Kompromittierung durch jemanden zu verhindern, der Ihre Anwendung oder die Komponenten inspizieren kann. Mit Secrets Manager können Sie hartkodierte Anmeldeinformationen durch einen Laufzeitaufruf des Secrets-Manager-Webservice ersetzen und somit die Anmeldeinformationen bei Bedarf dynamisch abrufen.

Mit Secrets Manager können Sie einen automatischen Rotationsplan für Ihre Secrets konfigurieren. Dies ermöglicht es Ihnen, langfristige Secrets durch kurzfristige zu ersetzen, was das Risiko einer Kompromittierung erheblich verringert. Da die Anmeldeinformationen nicht mehr in der Anwendung gespeichert werden, müssen für rotierende Anmeldeinformationen Ihre Anwendungen nicht mehr aktualisiert und Änderungen an den Anwendungsclients bereitgestellt werden.

Für andere Arten von Secrets, die Sie in Ihrer Organisation haben könnten:

- AWS Anmeldeinformationen — Wir empfehlen [AWS Identity and Access Management](#).
- Verschlüsselungsschlüssel – Wir empfehlen [AWS Key Management Service](#).
- SSH-Schlüssel – [Amazon EC2 Instance Connect](#).
- Private Schlüssel und Zertifikate – [AWS Certificate Manager](#).

Erste Schritte mit Secrets Manager

Wenn Sie Secrets Manager noch nicht kennen, beginnen Sie mit einem der folgenden Tutorials:

- [the section called “Fest codierte Secrets ersetzen ”](#)
- [the section called “Ersetzen von fest codierten DB-Anmeldeinformationen ”](#)
- [the section called “Drehung für wechselnde Benutzer”](#)
- [the section called “Einzelbenutzer-Drehung”](#)

Andere Aufgaben, die Sie mit Secrets erledigen können:

- [Erstellen und Verwalten von Secrets](#)
- [Zugriff auf Ihre Secrets steuern](#)
- [Holen Sie sich Geheimnisse](#)
- [Rotieren von -Geheimnissen](#)
- [Überwachung von Geheimnissen](#)
- [Überwachen Sie geheime Daten auf Einhaltung der Vorschriften](#)
- [Erstellen Sie Geheimnisse in AWS CloudFormation](#)

Einhaltung von Standards

AWS Secrets Manager wurde nach den verschiedenen Standards geprüft und kann Teil Ihrer Lösung sein, wenn Sie eine Konformitätszertifizierung benötigen. Weitere Informationen finden Sie unter [Compliance-Validierung](#).

Preisgestaltung

Bei der Nutzung von Secrets Manager zahlen Sie nur für das, was Sie nutzen, ohne Mindest- oder Einrichtungsgebühren. Es fallen keine Gebühren für Secrets an, die zum Löschen markiert wurden. Die aktuelle vollständige Preisliste finden Sie unter [AWS Secrets Manager – Preise](#). Informationen zur Überwachung Ihrer Kosten finden Sie unter [the section called “Secrets Manager Manager-Kosten überwachen”](#).

Sie können den Von AWS verwalteter Schlüssel `aws/secretsmanager`, den Secrets Manager erstellt, verwenden, um Ihre Geheimnisse kostenlos zu verschlüsseln. Wenn Sie Ihre eigenen KMS-Schlüssel zur Verschlüsselung Ihrer Geheimnisse erstellen, wird Ihnen der aktuelle AWS KMS Tarif AWS berechnet. Weitere Informationen finden Sie unter [AWS Key Management Service - Preisgestaltung](#).

Wenn Sie die automatische Rotation (außer der [verwalteten Rotation](#)) aktivieren, verwendet Secrets Manager eine AWS Lambda Funktion, um das Geheimnis zu rotieren, und Ihnen wird die Rotationsfunktion mit der aktuellen Lambda-Rate in Rechnung gestellt. Weitere Informationen finden Sie unter [AWS Lambda -Preisgestaltung](#).

Wenn Sie die Option in AWS CloudTrail Ihrem Konto aktivieren, können Sie Protokolle der API-Aufrufe abrufen, die Secrets Manager sendet. Secrets Manager protokolliert alle Ereignisse als

Verwaltungsereignisse. AWS CloudTrail speichert die erste Kopie aller Verwaltungsereignisse kostenlos. Es können jedoch Gebühren für Amazon S3 für die Speicherung der Protokolle und für Amazon SNS anfallen, wenn Sie die Benachrichtigung aktivieren. Wenn Sie zusätzliche Trails einrichten, können zudem für die zusätzlichen Kopien von Verwaltungsereignissen Kosten anfallen. Weitere Informationen finden Sie unter [AWS CloudTrail Preise](#).

Zugriff AWS Secrets Manager

Sie können folgendermaßen mit Secrets Manager arbeiten:

- [Secrets-Manager-Konsole](#)
- [Befehlszeilen-Tools](#)
- [AWS SDKs](#)
- [HTTPS-Query-API](#)
- [AWS Secrets Manager Endpunkte](#)

Secrets-Manager-Konsole

Sie können Ihre Secrets über die browserbasierte [Secrets-Manager-Konsole](#) verwalten und fast jede Aufgabe im Zusammenhang mit Ihren Secrets ausführen, indem Sie die Konsole verwenden.

Befehlszeilen-Tools

Mit den AWS Befehlszeilentools können Sie Befehle an der Befehlszeile Ihres Systems ausgeben, um Secrets Manager und andere AWS Aufgaben auszuführen. Dies kann schneller und bequemer sein als die Verwendung der Konsole. Die Befehlszeilentools können nützlich sein, wenn Sie Skripts zur Ausführung von AWS Aufgaben erstellen möchten.

Wenn Sie Befehle in eine Befehls-Shell eingeben, besteht die Gefahr, dass auf den Befehlsverlauf zugegriffen wird oder Serviceprogramme Zugriff auf Ihre Befehlsparameter haben. Siehe [the section called “Reduzieren von Risiken durch die Verwendung der AWS CLI zur Speicherung Ihrer AWS Secrets Manager-Secrets”](#).

Die Befehlszeilentools verwenden automatisch den Standardendpunkt für den Dienst in einer AWS Region. Sie können jedoch einen alternativen Endpunkt für Ihre API-Anforderungen festlegen. Siehe [the section called “Secrets-Manager-Endpunkte”](#).

AWS stellt zwei Gruppen von Befehlszeilentools bereit:

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

AWS SDKs

Die AWS SDKs bestehen aus Bibliotheken und Beispielcode für verschiedene Programmiersprachen und Plattformen. Mit den SDKs werden auch Aufgaben wie das kryptografische Signieren, die Behandlung von Fehlern und die automatische Wiederholung von Anforderungen abgedeckt. Informationen zum Herunterladen und Installieren eines der SDKs finden Sie unter [Tools für Amazon Web Services](#).

Die AWS SDKs verwenden automatisch den Standardendpunkt für den Dienst in einer AWS Region. Sie können jedoch einen alternativen Endpunkt für Ihre API-Anforderungen festlegen. Siehe [the section called “Secrets-Manager-Endpunkte”](#).

Eine SDK-Dokumentation finden Sie hier:

- [C++](#)
- [Go](#)
- [Java](#)
- [JavaScript](#)
- [Kotlin](#)
- [.NET](#)
- [PHP](#)
- [Python \(Teil 3\)](#)
- [Ruby](#)
- [Rust](#)
- [SAP ABAP](#)
- [Swift](#)

HTTPS-Query-API

Die HTTPS-Abfrage-API bietet Ihnen [programmatischen Zugriff auf](#) Secrets Manager und AWS. Mit der HTTPS-Abfrage-API können Sie HTTPS-Anforderungen direkt an den Service richten.

Obwohl Sie die Secrets-Manager-HTTPS-Query-API direkt aufrufen können, empfehlen wir Ihnen, stattdessen eines der SDKs zu verwenden. Das SDK führt viele nützliche Aufgaben durch, die

Sie ansonsten manuell erledigen müssen. Die SDKs signieren beispielsweise Ihre Anforderungen automatisch und konvertieren die Antworten in eine der Syntax Ihrer Sprache entsprechende Struktur.

Um HTTPS-Aufrufe an Secrets Manager zu tätigen, stellen Sie eine Verbindung zu [???](#) her.

AWS Secrets Manager Endpunkte

Um eine programmatische Verbindung mit Secrets Manager herzustellen, verwenden Sie einen Endpunkt, die URL des Einstiegspunkts für den Service. Secrets-Manager-Endpunkte sind Dual-Stack-Endpunkte, was bedeutet, dass sie sowohl IPv4 als auch IPv6 unterstützen.

Secrets Manager bietet Endpunkte, die [Federal Information Processing Standard \(FIPS\) 140-2](#) in einigen Regionen unterstützen.

Secrets Manager unterstützt TLS 1.2 und 1.3. Secrets Manager unterstützt [PQTLs](#) in allen Regionen bis auf Regionen in China.

Note

Das AWS Python-SDK und der AWS CLI Versuch, IPv6 und dann IPv4 nacheinander aufzurufen. Wenn Sie also IPv6 nicht aktiviert haben, kann es einige Zeit dauern, bis der Anruf das Timeout überschreitet und es erneut mit IPv4 versucht. Um dieses Problem zu umgehen, können Sie IPv6 vollständig deaktivieren oder [zu IPv6 migrieren](#).

Im Folgenden finden Sie die Service-Endpunkte für Secrets Manager. Beachten Sie, dass sich die Benennung von der [typischen Dual-Stack-Namenskonvention](#) unterscheidet.

Name der Region	Region	Endpunkt	Protocol (Protokoll)
USA Ost (Ohio)	us-east-2	secretsmanager.us-east-2.amazonaws.com	HTTPS
		secretsmanager-fips.us-east-2.amazonaws.com	HTTPS

Name der Region	Region	Endpoint	Protocol (Protokoll)
USA Ost (Nord-Virginia)	us-east-1	secretsmanager.us-east-1.amazonaws.com	HTTPS
		secretsmanager-fips.us-east-1.amazonaws.com	HTTPS
USA West (Nordkalifornien)	us-west-1	secretsmanager.us-west-1.amazonaws.com	HTTPS
		secretsmanager-fips.us-west-1.amazonaws.com	HTTPS
USA West (Oregon)	us-west-2	secretsmanager.us-west-2.amazonaws.com	HTTPS
		secretsmanager-fips.us-west-2.amazonaws.com	HTTPS
Afrika (Kapstadt)	af-south-1	secretsmanager.af-south-1.amazonaws.com	HTTPS
Asien-Pazifik (Hongkong)	ap-east-1	secretsmanager.ap-east-1.amazonaws.com	HTTPS
Asien-Pazifik (Hyderabad)	ap-south-2	secretsmanager.ap-south-2.amazonaws.com	HTTPS
Asien-Pazifik (Jakarta)	ap-southeast-3	secretsmanager.ap-southeast-3.amazonaws.com	HTTPS

Name der Region	Region	Endpoint	Protocol (Protokoll)
Asien-Pazifik (Melbourne)	ap-southeast-4	secretsmanager.ap-southeast-4.amazonaws.com	HTTPS
Asien-Pazifik (Mumbai)	ap-south-1	secretsmanager.ap-south-1.amazonaws.com	HTTPS
Asien-Pazifik (Osaka)	ap-northeast-3	secretsmanager.ap-northeast-3.amazonaws.com	HTTPS
Asien-Pazifik (Seoul)	ap-northeast-2	secretsmanager.ap-northeast-2.amazonaws.com	HTTPS
Asien-Pazifik (Singapur)	ap-southeast-1	secretsmanager.ap-southeast-1.amazonaws.com	HTTPS
Asien-Pazifik (Sydney)	ap-southeast-2	secretsmanager.ap-southeast-2.amazonaws.com	HTTPS
Asien-Pazifik (Tokio)	ap-northeast-1	secretsmanager.ap-northeast-1.amazonaws.com	HTTPS
Kanada (Zentral)	ca-central-1	secretsmanager.ca-central-1.amazonaws.com	HTTPS
		secretsmanager-fips.ca-central-1.amazonaws.com	HTTPS

Name der Region	Region	Endpoint	Protocol (Protokoll)
Kanada West (Calgary)	ca-west-1	secretsmanager.ca-west-1.amazonaws.com	HTTPS
		secretsmanager-fips.ca-west-1.amazonaws.com	HTTPS
Europa (Frankfurt)	eu-central-1	secretsmanager.eu-central-1.amazonaws.com	HTTPS
Europa (Irland)	eu-west-1	secretsmanager.eu-west-1.amazonaws.com	HTTPS
Europa (London)	eu-west-2	secretsmanager.eu-west-2.amazonaws.com	HTTPS
Europa (Mailand)	eu-south-1	secretsmanager.eu-south-1.amazonaws.com	HTTPS
Europa (Paris)	eu-west-3	secretsmanager.eu-west-3.amazonaws.com	HTTPS
Europa (Spanien)	eu-south-2	secretsmanager.eu-south-2.amazonaws.com	HTTPS
Europa (Stockholm)	eu-north-1	secretsmanager.eu-north-1.amazonaws.com	HTTPS
Europa (Zürich)	eu-central-2	secretsmanager.eu-central-2.amazonaws.com	HTTPS
Israel (Tel Aviv)	il-central-1	secretsmanager.il-central-1.amazonaws.com	HTTPS

Name der Region	Region	Endpunkt	Protocol (Protokoll)
Naher Osten (Bahrain)	me-south-1	secretsmanager.me-south-1.amazonaws.com	HTTPS
Naher Osten (VAE)	me-central-1	secretsmanager.me-central-1.amazonaws.com	HTTPS
Südamerika (São Paulo)	sa-east-1	secretsmanager.sa-east-1.amazonaws.com	HTTPS
AWS GovCloud (US-Ost)	us-gov-east-1	secretsmanager.us-gov-east-1.amazonaws.com	HTTPS
		secretsmanager-fips.us-gov-east-1.amazonaws.com	HTTPS
AWS GovCloud (US-West)	us-gov-west-1	secretsmanager.us-gov-west-1.amazonaws.com	HTTPS
		secretsmanager-fips.us-gov-west-1.amazonaws.com	HTTPS

Was ist in einem Secrets Manager Manager-Geheimnis enthalten?

Im Secrets Manager besteht ein Secret aus Secret-Informationen, dem Secret-Wert sowie Metadaten über das Secret. Ein Secret-Wert kann eine Zeichenfolge oder ein Binärwert sein.

Um mehrere Zeichenkettenwerte in einem Geheimnis zu speichern, empfehlen wir, eine JSON-Textzeichenfolge mit Schlüssel-Wert-Paaren zu verwenden, zum Beispiel:

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

Wenn Sie bei Datenbankgeheimnissen die automatische Rotation aktivieren möchten, muss das Geheimnis Verbindungsinformationen für die Datenbank in der richtigen JSON-Struktur enthalten. Weitere Informationen finden Sie unter [the section called "JSON-Struktur eines Secrets"](#).

Metadaten

Die Metadaten eines Secrets enthalten:

- Ein Amazon-Ressourcenname (ARN) mit dem folgenden Format:

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

Secrets Manager enthält sechs zufällige Zeichen am Ende des Secret-Namens, um sicherzustellen, dass der Secret-ARN einzigartig ist. Wenn das ursprüngliche Secret gelöscht und anschließend ein neues Secret mit demselben Namen erstellt wird, haben die beiden Secrets aufgrund dieser Zeichen unterschiedliche ARNs. Benutzer mit Zugriff auf das alte Secret erhalten nicht automatisch Zugriff auf das neue Secret, da die ARNs unterschiedlich sind.

- Der Name des Secrets, eine Beschreibung, eine Ressourcenrichtlinie und Tags.

- Der ARN für einen Verschlüsselungsschlüssel, den Secrets Manager zum Verschlüsseln und Entschlüsseln des geheimen Werts verwendet. AWS KMS key Secrets Manager speichert den Secret-Text in verschlüsselter Form und verschlüsselt das Secret während der Übertragung. Siehe [the section called “Ver- und Entschlüsselung von Secrets”](#).
- Informationen zum Rotieren des Secrets, wenn Sie die Rotation einrichten. Siehe [Rotieren von - Geheimnissen](#).

Secrets Manager verwendet IAM-Berechtigungsrichtlinien, um sicherzustellen, dass nur autorisierte Benutzer auf ein Geheimnis zugreifen oder es ändern können. Siehe [Authentifizierung und Zugriffskontrolle für AWS Secrets Manager](#).

Ein Geheimnis hat Versionen, die Kopien des verschlüsselten Geheimwerts enthalten. Wenn Sie den Secret-Wert ändern oder das Secret rotiert wird, erstellt Secrets Manager eine neue Version. Siehe [the section called “Geheime Versionen”](#).

Sie können ein Geheimnis für mehrere verwenden, AWS-Regionen indem Sie es replizieren. Wenn Sie ein Secret replizieren, erstellen Sie eine Kopie des Originals oder des primären Secrets. Diese Kopie wird als Secret-Replikat bezeichnet. Das Secret-Replikat bleibt mit dem primären Secret verknüpft. Siehe [Replizieren Sie Geheimnisse regionsübergreifend](#).

Siehe [Erstellen und Verwalten von Secrets](#).

Geheime Versionen

Ein Geheimnis hat Versionen, die Kopien des verschlüsselten geheimen Werts enthalten. Wenn Sie den Secret-Wert ändern oder das Secret rotiert wird, erstellt Secrets Manager eine neue Version.

Secrets Manager speichert keinen linearen Verlauf von Secrets mit Versionen. Stattdessen verfolgt es drei spezifische Versionen, indem es sie kennzeichnet:

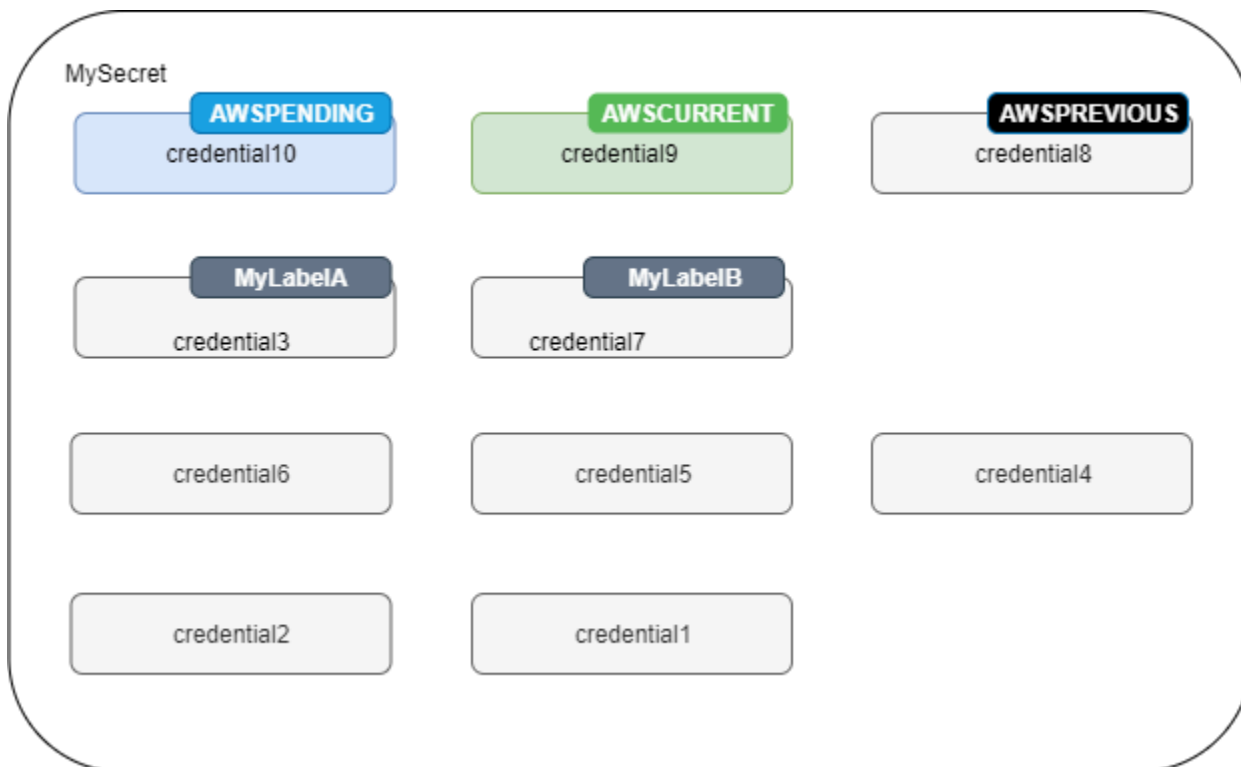
- Die aktuelle Version – AWSCURRENT
- Die vorherige Version – AWSPREVIOUS
- Die ausstehende Version (während der Rotation) – AWSPENDING

Ein Secrets verfügt immer über eine Version mit der Bezeichnung AWSCURRENT, und Secrets Manager gibt diese Version standardmäßig zurück, wenn Sie den Secrets-Wert abrufen.

Sie können Versionen auch mit Ihren eigenen Labels kennzeichnen, indem Sie die aufrufen [update-secret-version-stage](#) AWS CLI. Sie können einem Secret bis zu 20 Bezeichnungen zu Versionen anfügen. Zwei Versionen eines Secrets können nicht dieselbe Staging-Bezeichnung besitzen. Versionen können mehrere Bezeichnungen haben.

Secrets Manager entfernt niemals gekennzeichnete Versionen, Versionen ohne Bezeichnung gelten jedoch als veraltet. Secrets Manager entfernt veraltete Versionen, wenn mehr als 100 davon vorhanden sind. Secrets Manager entfernt keine Versionen, die vor weniger als 24 Stunden erstellt wurden.

Die folgende Abbildung zeigt ein Geheimnis AWS mit beschrifteten Versionen und vom Kunden beschrifteten Versionen. Die Versionen ohne Kennzeichnung gelten als veraltet und werden von Secrets Manager irgendwann in der Zukunft entfernt.



AWS Secrets Manager-Tutorials

Themen

- [Mit Amazon CodeGuru Reviewer ungeschützte Secrets in Ihrem Code finden](#)
- [Verschieben Sie hartcodierte Geheimnisse nach AWS Secrets Manager](#)
- [Verschieben Sie hartcodierte Datenbankanmeldedaten nach AWS Secrets Manager](#)
- [Richten Sie eine wechselnde Benutzerrotation ein für AWS Secrets Manager](#)
- [Einrichten der Einzelbenutzer-Drehung für AWS Secrets Manager](#)

Mit Amazon CodeGuru Reviewer ungeschützte Secrets in Ihrem Code finden

Amazon CodeGuru Reviewer ist ein Service, der Programmanalyse und Machine Learning verwendet, um potenzielle Defekte zu erkennen, die für Entwickler schwer zu finden sind, und Vorschläge zur Verbesserung Ihres Java- und Python-Codes bietet. CodeGuru Reviewer lässt sich in Secrets Manager integrieren, um ungeschützte Secrets in Ihrem Code zu finden. Unter [Secret-Typen, die von CodeGuru Reviewer erkannt wurden](#) im Benutzerhandbuch zu Amazon CodeGuru Reviewer erfahren Sie, welche Secret-Typen das Tool finden kann.

Sobald Sie fest codierte Secrets gefunden haben, ersetzen Sie sie:

- [the section called “Ersetzen von fest codierten DB-Anmeldeinformationen ”](#)
- [the section called “Fest codierte Secrets ersetzen ”](#)

Verschieben Sie hartcodierte Geheimnisse nach AWS Secrets Manager

Wenn Sie Klartext-Secrets in Ihrem Code haben, empfehlen wir Ihnen, sie zu drehen und in Secrets Manager zu speichern. Das Verschieben des Secrets in Secrets Manager löst das Problem, dass das Secret für jeden sichtbar ist, der den Code sieht, da Ihr Code in Zukunft das Secret direkt aus Secrets Manager abrufen wird. Durch Drehen des Secret wird das aktuelle fest codierte Secret widerrufen, sodass es nicht mehr gültig ist.

Informationen zu Secrets für Datenbankanmeldeinformationen finden Sie unter [Verschieben Sie hartcodierte Datenbankanmeldedaten nach AWS Secrets Manager](#).

Bevor Sie beginnen, müssen Sie ermitteln, wer Zugriff auf das Secret benötigt. Wir empfehlen, die Berechtigung für Ihr Secret von zwei IAM-Rollen verwalten zu lassen:

- Eine Rolle, die die Secrets in der Organisation verwaltet. Weitere Informationen finden Sie unter [the section called “Administratorberechtigungen für den Secrets Manager”](#). Mit dieser Rolle erstellen und drehen Sie das Secret.
- Eine Rolle, die das Geheimnis zur Laufzeit verwenden kann, zum Beispiel in diesem Tutorial, das Sie verwenden *RoleToRetrieveSecretAtRuntime*. Ihr Code übernimmt diese Rolle, um das Secret abzurufen. In diesem Tutorial erteilen Sie der Rolle nur die Berechtigung, einen Secret-Wert abzurufen, und Sie erteilen die Berechtigung mithilfe der Ressourcenrichtlinie des Secret. Weitere Alternativen finden Sie unter [the section called “Nächste Schritte”](#).

Schritte:

- [Schritt 1: Das Secret erstellen](#)
- [Schritt 2: Ihren Code aktualisieren](#)
- [Schritt 3: Das Secret aktualisieren](#)
- [Nächste Schritte](#)

Schritt 1: Das Secret erstellen

Der erste Schritt besteht darin, das vorhandene fest codierte Secret in Secrets Manager zu kopieren. Wenn sich das Geheimnis auf eine AWS Ressource bezieht, speichern Sie es in derselben Region wie die Ressource. Andernfalls speichern Sie es in der Region, die für Ihren Anwendungsfall die niedrigste Latenz aufweist.

Ein Secret erstellen (Konsole)

1. Öffnen Sie die Secrets-Manager-Konsole unter <https://console.aws.amazon.com/secretsmanager/>.
2. Wählen Sie Store a new secret (Ein neues Secret speichern).
3. Führen Sie auf der Seite Choose secret type (Secret-Typ auswählen) die folgenden Schritte aus:
 - a. Als Secret-Typ wählen Sie Anderer Secret-Typ aus.

- b. Geben Sie Ihr Secret als Schlüssel/Wert-Paare oder in Klartext an. Hier einige Beispiele:

Schlüssel-Wert-Paare des API-Schlüssels:

ClientID : *my_client_id*

ClientSecret : *bpXrfiWJALRXUTNFEMI/K7MDENG/CY* BEISPIELS
CHLÜSSEL

Schlüssel-Wert-Paare von Anmeldeinformationen:

Username: *Saanvis*

Password : *EXAMPLE-PASSWORD*

Klartext für OAuth-Token:

AKIAI44QH8DHBEXAMPLE

Klartext für digitales Zertifikat:

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

Klartext für privaten Schlüssel:

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

- c. Wählen Sie für Encryption key (Verschlüsselungsschlüssel) `aws/secretsmanager` aus, um den Von AWS verwalteter Schlüssel für Secrets Manager zu benutzen. Für die Verwendung dieses Schlüssels fallen keine Kosten an. Sie können auch Ihren eigenen vom Kunden verwalteten Schlüssel verwenden, z. B. um [von einem anderen AWS-Konto aus auf das Secret zuzugreifen](#). Informationen zu den Kosten der Verwendung eines vom Kunden verwalteten Schlüssels finden Sie unter [Preisgestaltung](#).
- d. Wählen Sie Weiter aus.
4. Führen Sie auf der Seite Choose secret type (Secret-Typ auswählen) die folgenden Schritte aus:

- a. Geben Sie einen beschreibenden Secret-Namen und eine Beschreibung ein.
- b. Unter Resource permissions (Ressourcenberechtigungen) wählen Sie Edit permissions (Berechtigungen bearbeiten) aus. Fügen Sie die folgende Richtlinie ein, die das Abrufen des Geheimnisses ermöglicht, und wählen Sie dann Speichern.

RoleToRetrieveSecretAtRuntime

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

- c. Wählen Sie unten auf der Seite Next (Weiter) aus.
5. Lassen Sie auf der Seite Configure rotation (Drehung konfigurieren) die Drehung deaktiviert. Wählen Sie Weiter aus.
 6. Prüfen Sie auf der Seite Review (Prüfen) die Secret-Details und wählen Sie Store (Speichern).

Schritt 2: Ihren Code aktualisieren

Ihr Code muss die IAM-Rolle übernehmen *RoleToRetrieveSecretAtRuntime*, um das Geheimnis abrufen zu können. Weitere Informationen finden Sie unter [Zu einer IAM-Rolle \(AWS API\) wechseln](#).

Als Nächstes aktualisieren Sie Ihren Code, um das Secret aus Secrets Manager mit dem von Secrets Manager bereitgestellten Beispielcode abzurufen.

So suchen Sie den Beispielcode

1. Öffnen Sie die Secrets-Manager-Konsole unter <https://console.aws.amazon.com/secretsmanager/>.

2. Wählen Sie auf der Seite Secrets Ihr Secret aus.
3. Scrollen Sie nach unten zu Sample code (Beispielcode). Wählen Sie Ihre Programmiersprache aus und kopieren Sie dann das Code-Snippet.

Entfernen Sie in Ihrer Anwendung das fest codierte Secret und fügen Sie das Code-Snippet ein. Abhängig von Ihrer Codesprache müssen Sie im Snippet möglicherweise einen Aufruf der Funktion oder Methode hinzufügen.

Testen Sie, ob Ihre Anwendung mit dem Secret anstelle des fest codierten Secret wie erwartet funktioniert.

Schritt 3: Das Secret aktualisieren

Der letzte Schritt besteht darin, das fest codierte Secret zu widerrufen und zu aktualisieren. Beziehen Sie sich auf die Quelle des Secret, um Anweisungen zum Widerrufen und Aktualisieren des Secret zu finden. Sie müssen beispielsweise möglicherweise das aktuelle Secret deaktivieren und ein neues Secret generieren.

So aktualisieren Sie das Secret mit dem neuen Wert

1. Öffnen Sie die Secrets-Manager-Konsole unter <https://console.aws.amazon.com/secretsmanager/>.
2. Wählen Sie Secrets und dann das gewünschte Secret aus.
3. Scrollen Sie auf der Seite Secret details (Secret-Details) nach unten und wählen Sie Retrieve secret value (Secret-Wert abrufen) und dann Edit (Bearbeiten) aus.
4. Aktualisieren Sie das Secret und wählen Sie dann Save (Speichern) aus.

Testen Sie anschließend, ob Ihre Anwendung mit dem neuen Secret wie erwartet funktioniert.

Nächste Schritte

Nachdem Sie ein fest codiertes Secret aus Ihrem Code entfernt haben, sind hier einige Vorschläge für die nächsten Schritte:

- Um hartcodierte Geheimnisse in Ihren Java- und Python-Anwendungen zu finden, empfehlen wir [Amazon CodeGuru Reviewer](#).

- Sie können die Leistung verbessern und Kosten senken, indem Sie Secrets zwischenspeichern. Weitere Informationen finden Sie unter [Holen Sie sich Geheimnisse](#).
- Für Secrets, auf die Sie aus mehreren Regionen zugreifen, sollten Sie erwägen, Ihr Secret zu replizieren, um die Latenz zu verbessern. Weitere Informationen finden Sie unter [Replizieren Sie Geheimnisse regionsübergreifend](#).
- In diesem Tutorial haben Sie `RoleToRetrieveSecretAtRuntime` nur die Erlaubnis erteilt, den geheimen Wert abzurufen. Unter [the section called “Beispiele für Richtlinien für Berechtigungen”](#) erfahren Sie, wie Sie der Rolle mehr Berechtigungen erteilen, z. B. zum Abrufen von Metadaten über das Secret oder Anzeigen einer Liste von Secrets.
- In diesem Tutorial haben Sie `RoleToRetrieveSecretAtRuntime` mithilfe der Ressourcenrichtlinie des Geheimnisses die Erlaubnis erteilt. Weitere Möglichkeiten zum Erteilen von Berechtigungen finden Sie unter [the section called “Anhängen einer Berechtigungsrichtlinie an eine Identität”](#).

Verschieben Sie hartcodierte Datenbankanmeldedaten nach AWS Secrets Manager

Wenn Sie Anmeldeinformationen für Klartext-Datenbanken in Ihrem Code haben, empfehlen wir Ihnen, die Anmeldeinformationen in Secrets Manager zu verschieben und sie dann sofort zu drehen. Das Verschieben der Anmeldeinformationen in Secrets Manager löst das Problem, dass die Anmeldeinformationen für jeden sichtbar sind, der den Code sieht, da Ihr Code in Zukunft die Anmeldeinformationen direkt aus Secrets Manager abrufen wird. Durch Drehen des Secret wird das Passwort aktualisiert und daraufhin das aktuelle fest codierte Passwort widerrufen, sodass es nicht mehr gültig ist.

Befolgen Sie für Amazon-RDS-, Amazon-Redshift- und Amazon-DocumentDB-Datenbanken die Schritte auf dieser Seite, um fest codierte Anmeldeinformationen in Secrets Manager zu verschieben. Informationen zu anderen Anmeldeinformationstypen und anderen Secrets finden Sie unter [the section called “Fest codierte Secrets ersetzen”](#).

Bevor Sie beginnen, müssen Sie ermitteln, wer Zugriff auf das Secret benötigt. Wir empfehlen, die Berechtigung für Ihr Secret von zwei IAM-Rollen verwalten zu lassen:

- Eine Rolle, die die Secrets in der Organisation verwaltet. Weitere Informationen finden Sie unter [the section called “Administratorberechtigungen für den Secrets Manager”](#). Mit dieser Rolle erstellen und drehen Sie das Secret.

- Eine Rolle, die die Anmeldeinformationen zur Laufzeit verwenden kann, finden Sie *RoleToRetrieveSecretAtRuntime* in diesem Tutorial. Ihr Code übernimmt diese Rolle, um das Secret abzurufen.

Schritte:

- [Schritt 1: Das Secret erstellen](#)
- [Schritt 2: Ihren Code aktualisieren](#)
- [Schritt 3: Drehen des Geheimnisses](#)
- [Nächste Schritte](#)

Schritt 1: Das Secret erstellen

Der erste Schritt besteht darin, die vorhandenen fest codierten Anmeldeinformationen in ein Secret in Secrets Manager zu kopieren. Speichern Sie das Secret in derselben Region wie die Datenbank, um die niedrigste Latenz zu erreichen.

So erstellen Sie ein Secret

1. Öffnen Sie die Secrets-Manager-Konsole unter <https://console.aws.amazon.com/secretsmanager/>.
2. Wählen Sie Store a new secret (Ein neues Secret speichern).
3. Führen Sie auf der Seite Choose secret type (Secret-Typ auswählen) die folgenden Schritte aus:
 - a. Wählen Sie als Secret-Typ den Typ der zu speichernden Datenbank-Anmeldeinformation aus:
 - Amazon-RDS-Datenbank
 - Amazon-DocumentDB-Datenbank
 - Amazon Redshift Redshift-Datawarehouse.
 - Weitere Secret-Typen finden Sie unter [Ersetzen von fest codierten Secrets](#) .
 - b. Geben Sie als Anmeldeinformationen die vorhandenen fest codierten Anmeldeinformationen für die Datenbank ein.
 - c. Wählen Sie für Encryption key (Verschlüsselungsschlüssel) aws/secretsmanager aus, um den Von AWS verwalteter Schlüssel für Secrets Manager zu benutzen. Für die Verwendung dieses Schlüssels fallen keine Kosten an. Sie können auch Ihren eigenen vom Kunden

verwalteten Schlüssel verwenden, z. B. um [von einem anderen AWS-Konto aus auf das Secret zuzugreifen](#). Informationen zu den Kosten der Verwendung eines vom Kunden verwalteten Schlüssels finden Sie unter [Preisgestaltung](#).

- d. Als Datenbank wählen Sie Ihre Datenbank aus.
 - e. Wählen Sie Weiter aus.
4. Führen Sie auf der Seite Configure secret (Secret konfigurieren) die folgenden Schritte aus:
- a. Geben Sie einen beschreibenden Secret-Namen und eine Beschreibung ein.
 - b. Unter Resource permissions (Ressourcenberechtigungen) wählen Sie Edit permissions (Berechtigungen bearbeiten) aus. Fügen Sie die folgende Richtlinie ein, die das Abrufen des Geheimnisses ermöglicht *RoleToRetrieveSecretAtRuntime*, und wählen Sie dann Speichern.

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

- c. Wählen Sie unten auf der Seite Next (Weiter) aus.
5. Lassen Sie auf der Seite Configure rotation (Drehung konfigurieren) die Drehung vorerst deaktiviert. Sie werden sie später aktivieren. Wählen Sie Weiter aus.
6. Prüfen Sie auf der Seite Review (Prüfen) die Secret-Details und wählen Sie Store (Speichern).

Schritt 2: Ihren Code aktualisieren

Ihr Code muss die IAM-Rolle übernehmen *RoleToRetrieveSecretAtRuntime*, um das Geheimnis abrufen zu können. Weitere Informationen finden Sie unter [Zu einer IAM-Rolle \(AWS API\) wechseln](#).

Als Nächstes aktualisieren Sie Ihren Code, um das Secret aus Secrets Manager mit dem von Secrets Manager bereitgestellten Beispielcode abzurufen.

So suchen Sie den Beispielcode

1. Öffnen Sie die Secrets-Manager-Konsole unter <https://console.aws.amazon.com/secretsmanager/>.
2. Wählen Sie auf der Seite Secrets Ihr Secret aus.
3. Scrollen Sie nach unten zu Sample code (Beispielcode). Wählen Sie Ihre Sprache aus und kopieren Sie dann das Code-Snippet.

Entfernen Sie in Ihrer Anwendung die fest codierten Anmeldeinformationen und fügen Sie das Code-Snippet ein. Abhängig von Ihrer Codesprache müssen Sie im Snippet möglicherweise einen Aufruf der Funktion oder Methode hinzufügen.

Testen Sie, ob Ihre Anwendung mit dem Secret anstelle der fest codierten Anmeldeinformationen wie erwartet funktioniert.

Schritt 3: Drehen des Geheimnisses

Der letzte Schritt besteht darin, die fest codierten Anmeldeinformationen durch Drehen des Secret zu widerrufen. Drehung ist der Prozess der periodischen Aktualisierung eines Secrets. Wenn Sie ein Secret drehen, werden die Anmeldeinformationen sowohl im Secret als auch in der Datenbank aktualisiert. Sie können einen Zeitplan einrichten, zu dem Secrets Manager Ihr Secret automatisch dreht.

Ein Teil der Einrichtung der Drehung besteht darin, sicherzustellen, dass die Lambda-Drehungsfunktion sowohl auf Secrets Manager als auch auf Ihre Datenbank zugreifen kann. Wenn Sie die automatische Drehung aktivieren, erstellt Secrets Manager die Lambda-Drehungsfunktion in derselben VPC wie Ihre Datenbank, damit sie Netzwerkzugriff auf die Datenbank erhält. Die Lambda-Drehungsfunktion muss auch in der Lage sein, Aufrufe an Secrets Manager zu tätigen, um das Secret zu aktualisieren. Wir empfehlen, dass Sie einen Secrets Manager-Endpunkt in der VPC erstellen, damit Aufrufe von Lambda an Secrets Manager die Infrastruktur nicht verlassen AWS. Anweisungen finden Sie unter [VPC-Endpunkt](#).

So aktivieren Sie die Drehung

1. Öffnen Sie die Secrets-Manager-Konsole unter <https://console.aws.amazon.com/secretsmanager/>.

2. Wählen Sie auf der Seite Secrets Ihr Secret aus.
3. Klicken Sie auf der Seite mit den Secret-Details im Abschnitt Rotation configuration (Rotationskonfiguration) auf Edit rotation (Rotation bearbeiten).
4. Führen Sie im Dialogfeld Edit rotation configuration (Rotationskonfiguration bearbeiten) die folgenden Schritte aus:
 - a. Schalten Sie die automatische Rotation ein.
 - b. Geben Sie unter Rotation schedule (Rotationszeitplan) den Zeitplan in der UTC-Zeitzone ein.
 - c. Wählen Sie Rotate immediately when the secret is stored (Sofort rotieren, wenn das Secret gespeichert ist) aus, um Ihr Secret zu drehen, wenn Sie Ihre Änderungen speichern.
 - d. Wählen Sie unter Rotation function (Rotationsfunktion) Create a new Lambda function (Eine neue Lambda-Funktion erstellen) aus und geben Sie einen Namen für die neue Funktion ein. Secrets Manager fügt „SecretsManager“ am Anfang Ihres Funktionsnamens hinzu.
 - e. Wählen Sie für die Rotationsstrategie Einzelbenutzer aus.
 - f. Wählen Sie Speichern.

So überprüfen Sie, ob das Secret gedreht wurde

1. Öffnen Sie die Secrets-Manager-Konsole unter <https://console.aws.amazon.com/secretsmanager/>.
2. Wählen Sie Secrets und dann das gewünschte Secret aus.
3. Scrollen Sie auf der Seite Secret details (Geheimnis-Details) nach unten und wählen Sie Retrieve secret value (Geheimnis-Wert abrufen) aus.

Wenn sich der Secret-Wert geändert hat, war die Drehung erfolgreich. Wenn sich der geheime Wert nicht geändert hat, müssen Sie sich [Fehlerbehebung bei der -Rotation](#) die CloudWatch Logs für die Rotationsfunktion ansehen.

Testen Sie, ob Ihre Anwendung mit dem gedrehten Secret wie erwartet funktioniert.

Nächste Schritte

Nachdem Sie ein fest codiertes Secret aus Ihrem Code entfernt haben, sind hier einige Vorschläge für die nächsten Schritte:

- Sie können die Leistung verbessern und Kosten senken, indem Sie Secrets zwischenspeichern. Weitere Informationen finden Sie unter [Holen Sie sich Geheimnisse](#).
- Sie können einen anderen Drehungszeitplan auswählen. Weitere Informationen finden Sie unter [the section called “Rotationspläne”](#).
- Um hartcodierte Geheimnisse in Ihren Java- und Python-Anwendungen zu finden, empfehlen wir [Amazon CodeGuru Reviewer](#).

Richten Sie eine wechselnde Benutzerrotation ein für AWS Secrets Manager

In diesem Lernprogramm erfahren Sie, wie Sie die Drehung von wechselnden Benutzern für ein Geheimnis einrichten, das Datenbank-Anmeldeinformationen enthält. Drehung wechselnder Benutzer ist eine Drehungsstrategie, bei der Secrets Manager den Benutzer kloniert und dann wechselt, welche Anmeldeinformationen des Benutzers aktualisiert werden. Diese Strategie ist eine gute Wahl, wenn Sie eine hohe Verfügbarkeit für Ihr Secret benötigen, da einer der alternierenden Benutzer über aktuelle Anmeldeinformationen für die Datenbank verfügt, während der andere aktualisiert wird. Weitere Informationen finden Sie unter [the section called “Wechselnde Benutzer”](#).

Um die Drehung alternativer Benutzer einzurichten, benötigen Sie zwei Geheimnisse:

- Ein Geheimnis mit den Anmeldeinformationen, die Sie drehen möchten.
- Ein zweites Secret mit Administratoranmeldeinformationen.

Dieser Benutzer ist berechtigt, den ersten Benutzer zu klonieren und das Passwort des ersten Benutzers zu ändern. In diesem Tutorial lassen Sie Amazon RDS dieses Secret für einen Admin-Benutzer erstellen. Amazon RDS verwaltet auch die Admin-Passwortrotation. Weitere Informationen finden Sie unter [the section called “Verwaltete Rotation”](#).

Im ersten Teil dieses Tutorials geht es um das Einrichten einer realistischen Umgebung. Um Ihnen die Funktionsweise der Drehung zu zeigen, verwendet dieses Tutorial ein beispielhafte MySQL-Datenbank von Amazon RDS. Aus Sicherheitsgründen befindet sich die Datenbank in einer VPC, die eingehenden Internetzugriff beschränkt. Um von Ihrem lokalen Computer über das Internet eine Verbindung zur Datenbank herzustellen, verwenden Sie einen Bastion-Host, einen Server in der VPC, der eine Verbindung zur Datenbank herstellen kann, aber auch SSH-Verbindungen aus dem Internet erlaubt. Der Bastion-Host in diesem Tutorial ist eine Amazon-EC2-Instance und die Sicherheitsgruppen für die Instance verhindern andere Arten von Verbindungen.

Nachdem Sie das Tutorial abgeschlossen haben, empfehlen wir, dass Sie die Ressourcen aus dem Tutorial bereinigen. Verwenden Sie sie nicht in einer Produktionsumgebung.

Die Secrets Manager Manager-Rotation verwendet eine AWS Lambda Funktion, um das Geheimnis und die Datenbank zu aktualisieren. Hinweise zu den Kosten der Verwendung einer Lambda-Funktion finden Sie unter [Preisgestaltung](#).

Tutorial:

- [Berechtigungen](#)
- [Voraussetzungen](#)
- [Schritt 1: Erstellen eines Amazon-RDS-Datenbankbenutzers](#)
- [Schritt 2: Erstellen Sie ein Geheimnis für die Benutzer-Anmeldeinformationen](#)
- [Schritt 3: Testen des gedrehten Geheimnisses](#)
- [Schritt 4: Bereinigen von Ressourcen](#)
- [Nächste Schritte](#)

Berechtigungen

Als Teil der Voraussetzungen für das Tutorial benötigen Sie Administratorberechtigungen für Ihr AWS-Konto. In einer Produktionsumgebung ist es eine bewährte Methode, für jeden der Schritte verschiedene Rollen zu verwenden. Beispielsweise würde eine Rolle mit Datenbank-Administratorberechtigungen die Amazon-RDS-Datenbank erstellen, und eine Rolle mit Netzwerk-Administratorberechtigungen würde die VPC und Sicherheitsgruppen einrichten. Für die Tutorial-Schritte empfehlen wir Ihnen, weiterhin dieselbe Identität zu verwenden.

Informationen zum Einrichten von Berechtigungen in einer Produktionsumgebung finden Sie unter [Authentifizierung und Zugriffskontrolle](#).

Voraussetzungen

Für dieses Tutorial benötigen Sie Folgendes:

- [Voraussetzung A: Amazon VPC](#)
- [Voraussetzung B: Amazon-EC2-Instance](#)
- [Voraussetzung C: Amazon-RDS-Datenbank und ein Secrets-Manager-Secret für die Administrator-Anmeldeinformationen](#)

- [Voraussetzung D: Ihrem lokalen Computer gestatten, eine Verbindung mit der EC2-Instance herzustellen](#)

Voraussetzung A: Amazon VPC

In diesem Schritt erstellen Sie eine VPC, in die Sie eine Amazon-RDS-Datenbank und eine Amazon-EC2-Instance starten können. In einem späteren Schritt verwenden Sie Ihren Computer, um über das Internet eine Verbindung zur Bastion und dann zur Datenbank herzustellen. Sie müssen also Datenverkehr aus der VPC zulassen. Dazu fügt Amazon VPC ein Internet-Gateway an die VPC an und fügt eine Route in der Routing-Tabelle hinzu, sodass Datenverkehr, der für außerhalb der VPC bestimmt ist, an das Internet-Gateway gesendet wird.

Innerhalb der VPC erstellen Sie einen Secrets-Manager-Endpunkt und einen Amazon-RDS-Endpunkt. Wenn Sie die automatische Rotation einrichten, erstellt Secrets Manager die Lambda-Rotationsfunktion innerhalb der VPC, sodass sie Zugriff auf die Datenbank hat. Die Lambda-Rotationsfunktion ruft auch Secrets Manager auf, um das Secret zu aktualisieren, und sie ruft Amazon RDS auf, um die Datenbankverbindungsinformationen abzurufen. Durch die Erstellung von Endpunkten innerhalb der VPC stellen Sie sicher, dass Aufrufe der Lambda-Funktion an Secrets Manager und Amazon RDS die Infrastruktur nicht verlassen. AWS Stattdessen werden sie an die Endpunkte innerhalb der VPC weitergeleitet.

So erstellen Sie eine VPC

1. Öffnen Sie die Amazon VPC-Konsole unter <https://console.aws.amazon.com/vpc/>.
2. Wählen Sie VPC erstellen aus.
3. Wählen Sie auf der Seite Create VPC (VPC erstellen) die Option VPC and more (VPC und mehr) aus.
4. Geben Sie unter Name tag auto-generation (Automatische Generierung des Nametags) unter Auto-generate (Automatisch generieren) **SecretsManagerTutorial** ein.
5. Wählen Sie für DNS options (DNS-Optionen) sowohl **Enable DNS hostnames** als auch **Enable DNS resolution** aus.
6. Wählen Sie VPC erstellen aus.

So erstellen Sie einen Secrets-Manager-Endpunkt innerhalb der VPC

1. Wählen Sie in der Amazon-VPC-Konsole unter Endpoints (Endpunkte) die Option Create Endpoint (Endpunkt erstellen) aus.

2. Geben Sie unter Endpoint settings (Endpunkt-Einstellungen) als Name (Name) **SecretsManagerTutorialEndpoint** ein.
3. Geben Sie unter Services **secretsmanager** ein, um die Liste zu filtern, und wählen Sie dann den Secrets-Manager-Endpunkt in Ihrer AWS-Region aus. Wählen Sie zum Beispiel in USA-Ost (Nord-Virginia) `com.amazonaws.us-east-1.secretsmanager` aus.
4. Wählen Sie für VPC **vpc**** (SecretsManagerTutorial)** aus.
5. Wählen Sie für Subnets (Subnetze) alle Availability Zones aus und wählen Sie dann für jede einzelne eine einzuschließende Subnetz-ID aus.
6. Wählen Sie für IP address type (Typ der IP-Adresse) die Option **IPv4** aus.
7. Wählen Sie unter Security groups (Sicherheitsgruppen) die Standard-Sicherheitsgruppe aus.
8. Wählen Sie für Policy (Richtlinie) **Full access** aus.
9. Wählen Sie Endpunkt erstellen aus.

So erstellen Sie einen Amazon-RDS-Endpunkt innerhalb der VPC

1. Wählen Sie in der Amazon-VPC-Konsole unter Endpoints (Endpunkte) die Option Create Endpoint (Endpunkt erstellen) aus.
2. Geben Sie unter Endpoint settings (Endpunkt-Einstellungen) als Name (Name) **RDS TutorialEndpoint** ein.
3. Geben Sie unter Services **rds** ein, um die Liste zu filtern, und wählen Sie dann den Amazon-RDS-Endpunkt in Ihrer AWS-Region aus. Wählen Sie zum Beispiel in USA-Ost (Nord-Virginia) `com.amazonaws.us-east-1.rds` aus.
4. Wählen Sie für VPC **vpc**** (SecretsManagerTutorial)** aus.
5. Wählen Sie für Subnets (Subnetze) alle Availability Zones aus und wählen Sie dann für jede einzelne eine einzuschließende Subnetz-ID aus.
6. Wählen Sie für IP address type (Typ der IP-Adresse) die Option **IPv4** aus.
7. Wählen Sie unter Security groups (Sicherheitsgruppen) die Standard-Sicherheitsgruppe aus.
8. Wählen Sie für Policy (Richtlinie) **Full access** aus.
9. Wählen Sie Endpunkt erstellen aus.

Voraussetzung B: Amazon-EC2-Instance

Die Amazon RDS-Datenbank, die Sie in einem späteren Schritt erstellen, befindet sich in der VPC. Um darauf zuzugreifen, benötigen Sie also einen Bastion-Host. Der Bastion-Host befindet sich ebenfalls in der VPC, aber in einem späteren Schritt konfigurieren Sie eine Sicherheitsgruppe, sodass Ihr lokaler Computer eine Verbindung mit SSH mit dem Bastion-Host herstellen kann.

So erstellen Sie eine EC2-Instance für einen Bastion-Host

1. Öffnen Sie die Amazon EC2-Konsole unter <https://console.aws.amazon.com/ec2/>.
2. Wählen Sie Instances und dann Launch instances (Instances launchen) aus.
3. Geben Sie unter Name and tags (Name und Tags) als Name den Namen **SecretsManagerTutorialInstance** ein.
4. Behalten Sie unter Application and OS Images (Anwendungs- und Betriebssystemabbilder) die Standardeinstellung **Amazon Linux 2 AMI (HVM) Kernel 5.10** bei.
5. Behalten Sie unter Instance type (Typ der Instance) die Standardeinstellung **t2.micro** bei.
6. Wählen Sie unter Key pair (Schlüsselpaar) die Option Create key pair (Schlüsselpaar erstellen) aus.

Geben Sie im Dialogfeld Create Key Pair (Schlüsselpaar erstellen) im Feld Key pair name (Schlüsselpaarname) als Namen **SecretsManagerTutorialKeyPair** ein und klicken Sie auf Create key pair (Schlüsselpaar erstellen).

Das Schlüsselpaar wird automatisch heruntergeladen.

7. Wählen Sie unter Network settings (Netzwerkeinstellungen) die Option Edit (Bearbeiten) und gehen Sie wie folgt vor:
 - a. Wählen Sie für VPC **vpc-**** SecretsManagerTutorial** aus.
 - b. Wählen Sie für Auto-assign public IP (Öffentliche IP automatisch zuweisen) **Enable** aus.
 - c. Wählen Sie bei Firewall (Firewall) Select existing security group (Vorhandene Sicherheitsgruppe auswählen) aus.
 - d. Wählen Sie bei Common security groups (Allgemeine Sicherheitsgruppen) **default** aus.
8. Wählen Sie Launch Instance (Instance starten) aus.

Voraussetzung C: Amazon-RDS-Datenbank und ein Secrets-Manager-Secret für die Administrator-Anmeldeinformationen

In diesem Schritt erstellen Sie eine Amazon-RDS-MySQL-Datenbank und konfigurieren sie so, dass Amazon RDS ein Secret erstellt, das die Administrator-Anmeldeinformationen enthält. Dann verwaltet Amazon RDS automatisch die Rotation des Admin-Secrets für Sie. Weitere Informationen finden Sie unter [Verwaltete Rotation](#).

Im Rahmen der Erstellung Ihrer Datenbank geben Sie den Bastion-Host an, den Sie im vorherigen Schritt erstellt haben. Dann richtet Amazon RDS Sicherheitsgruppen ein, sodass die Datenbank und die Instance aufeinander zugreifen können. Sie fügen der Sicherheitsgruppe, die an die Instance angehängt ist, eine Regel hinzu, damit auch Ihr lokaler Computer eine Verbindung zu ihr herstellen kann.

So erstellen Sie eine Amazon-RDS-Datenbank mit einem Secrets-Manager-Secret, das die Administratoranmeldeinformationen enthält

1. Wählen Sie in der Amazon-RDS-Konsole Databases (Datenbanken) aus.
2. Wählen Sie im Abschnitt Engine options (Engine-Optionen) bei Engine type (Engine-Typ) die Option **MySQL** aus.
3. Wählen Sie im Abschnitt Templates (Vorlagen) die Option **Free tier** aus.
4. Gehen Sie im Abschnitt Settings (Einstellungen) wie folgt vor:
 - a. Geben Sie als DB instance identifier (DB-Instance-ID) **SecretsManagerTutorial** ein.
 - b. Wählen Sie unter Einstellungen für Anmeldeinformationen die Option Masteranmeldedaten verwalten in aus. AWS Secrets Manager
5. Wählen Sie im Abschnitt Connectivity (Konnektivität) für Computer resource (Computerressource) die Option Connect to an EC2 computer resource (Mit einer EC2-Computerressource verbinden) aus, und wählen Sie dann bei EC2 Instance (EC2-Instance) die Option **SecretsManagerTutorialInstance**.
6. Wählen Sie Datenbank erstellen aus.

Voraussetzung D: Ihrem lokalen Computer gestatten, eine Verbindung mit der EC2-Instance herzustellen

In diesem Schritt konfigurieren Sie die EC2-Instance, die Sie in Voraussetzung B erstellt haben, so, dass Ihr lokaler Computer eine Verbindung zu ihr herstellen kann. Dazu bearbeiten Sie die

Sicherheitsgruppe, die Amazon RDS in Voraussetzung C hinzugefügt hat, sodass sie eine Regel enthält, die es der IP-Adresse Ihres Computers ermöglicht, eine Verbindung mit SSH herzustellen. Die Regel ermöglicht es Ihrem lokalen Computer (identifiziert anhand Ihrer aktuellen IP-Adresse), mithilfe von SSH über das Internet eine Verbindung mit dem Bastion-Host herzustellen.

So gestatten Sie Ihrem lokalen Computer, eine Verbindung mit der EC2-Instance herzustellen

1. Öffnen Sie die Amazon EC2-Konsole unter <https://console.aws.amazon.com/ec2/>.
2. Wählen Sie auf der EC2-Instance SecretsManagerTutorialInstance auf der Registerkarte Sicherheit unter Sicherheitsgruppen die Option. **sg-*** (ec2-rds-X)**
3. Wählen Sie auf der Registerkarte Inbound rules (Regeln für eingehenden Datenverkehr) die Option Edit inbound rules (Regeln für eingehenden Datenverkehr bearbeiten) aus.
4. Wählen Sie Add rule (Regel hinzufügen) und gehen Sie wie folgt vor:
 - a. Wählen Sie für Type (Typ) die Option **SSH** aus.
 - b. Wählen Sie im Feld Source type (Quellentyp) die Option **My IP** aus.

Schritt 1: Erstellen eines Amazon-RDS-Datenbankbenutzers

Zuerst benötigen Sie einen Benutzer, dessen Anmeldeinformationen im Geheimnis gespeichert werden. Um den Benutzer zu erstellen, melden Sie sich mit Administratoranmeldeinformationen bei der Amazon-RDS-Datenbank an. Der Einfachheit halber erstellen Sie im Tutorial einen Benutzer mit voller Berechtigung für eine Datenbank. In einer Produktionsumgebung ist dies nicht typisch, und wir empfehlen, dem Prinzip der geringsten Berechtigung zu folgen.

Um eine Verbindung mit der Datenbank herzustellen, verwenden Sie ein MySQL-Client-Tool. In diesem Tutorial verwenden Sie MySQL Workbench, eine GUI-basierte Anwendung. Informationen zum Installieren von MySQL Workbench finden Sie unter [MySQL Workbench herunterladen](#).

Um eine Verbindung mit der Datenbank herzustellen, erstellen Sie eine Verbindungskonfiguration in MySQL Workbench. Für die Konfiguration benötigen Sie einige Informationen sowohl von Amazon EC2 als auch von Amazon RDS.

So erstellen Sie eine Datenbankverbindung in MySQL Workbench

1. Wählen Sie in MySQL Workbench neben MySQL Connections (MySQL-Verbindungen) die Schaltfläche (+) aus.
2. Gehen Sie im Dialogfeld Setup New Connection (Neue Verbindung einrichten) wie folgt vor:

- a. Geben Sie für Connection name (Verbindungsname) **SecretsManagerTutorial** ein.
- b. Wählen Sie für Connection method (Verbindungsmethode) **Standard TCP/IP over SSH** aus.
- c. Gehen Sie auf der Registerkarte Parameters (Parameter) folgendermaßen vor:
 - i. Geben Sie für SSH-Hostname die öffentliche IP-Adresse der Amazon-EC2-Instance ein.

Sie finden die IP-Adresse auf der Amazon EC2 EC2-Konsole, indem Sie die Instance **SecretsManagerTutorialInstance** auswählen. Kopieren Sie die IP-Adresse unter Public IPv4 DNS.
 - ii. Geben Sie für SSH user name (SSH-Benutzername) **ec2-user** ein.
 - iii. Wählen Sie für SSH Keyfile die Schlüsselpaardatei **SecretsManagerTutorialKeyPair.pem** aus, die Sie in der vorherigen Voraussetzung heruntergeladen haben.
 - iv. Geben Sie für MySQL Hostname (MySQL-Hostname) die Amazon-RDS-Endpunktadresse ein.

Sie finden die Endpunktadresse in der Amazon-RDS-Konsole, indem Sie die Datenbank-Instance **secretsmanagertutorialdb** auswählen. Kopieren Sie die Adresse unter Endpoint (Endpunkt).
 - v. Geben Sie für Username (Benutzername) **admin** ein.
- d. Wählen Sie OK aus.

So rufen Sie das Admin-Passwort ab

1. Gehen Sie in der Amazon-RDS-Konsole zu Ihrer Datenbank.
2. Wählen Sie auf der Registerkarte Configuration (Konfiguration) unter Master Credentials ARN (Master-Anmeldeinformationen-ARN) die Option Manage in Secrets Manager (In Secrets Manager verwalten) aus.

Die Secrets-Manager-Konsole wird geöffnet.

3. Wählen Sie auf der Seite „Secret details“ (Secret-Details) die Option Retrieve secret value (Secret-Wert abrufen) aus.
4. Das Passwort wird im Abschnitt Secret value (Secret-Wert) angezeigt.

So erstellen Sie einen Datenbankbenutzer

1. Wählen Sie in MySQL Workbench die Verbindung `SecretsManagerTutorialaus`.
2. Geben Sie das Admin-Passwort ein, das Sie aus dem Secret abgerufen haben.
3. Geben Sie in MySQL Workbench im Fenster Query (Abfragen) die folgenden Befehle (einschließlich eines sicheren Passworts) ein und wählen Sie dann Execute (Ausführen) aus.

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

Im Fenster Output (Ausgabe) sehen Sie, dass die Befehle erfolgreich sind.

Schritt 2: Erstellen Sie ein Geheimnis für die Benutzer-Anmeldeinformationen

Als Nächstes erstellen Sie ein Geheimnis zum Speichern der Anmeldeinformationen des gerade erstellten Benutzers. Das ist das Geheimnis, das Sie drehen werden. Sie aktivieren die automatische Drehung und, um die Strategie für alternative Benutzer anzugeben, wählen Sie ein separates Superuser-Geheimnis aus, das berechtigt ist, das Passwort des ersten Benutzers zu ändern.

1. Öffnen Sie die Secrets-Manager-Konsole unter <https://console.aws.amazon.com/secretsmanager/>.
2. Wählen Sie Store a new secret (Ein neues Secret speichern).
3. Führen Sie auf der Seite Choose secret type (Secret-Typ auswählen) die folgenden Schritte aus:
 - a. Wählen Sie für Secret type (Geheimnistyp) Credentials for Amazon RDS database (Anmeldedaten für die Amazon-RDS-Datenbank) aus.
 - b. Geben Sie für Credentials (Anmeldeinformationen) den Benutzernamen **appuser** und das Passwort ein, das Sie für den Datenbankbenutzer eingegeben haben, den Sie mit MySQL Workbench erstellt haben.
 - c. Wählen Sie für Database (Datenbank) `secretsmanagertutorialdb` aus.
 - d. Wählen Sie Weiter aus.
4. Geben Sie auf der Seite Configure secret (Secret konfigurieren) für Secret name (Secret-Name) **SecretsManagerTutorialAppuser** ein und wählen Sie dann Next (Weiter) aus.

5. Führen Sie auf der Seite `Configure rotation` (Drehung konfigurieren) die folgenden Schritte aus:
 - a. Schalten Sie die automatische Rotation ein.
 - b. Legen Sie für `Rotation schedule` (Drehungsplan) einen Zeitplan von `Days` (Tagen) fest: **2** Tage mit `Duration` (Dauer): **2h**. Behalten Sie die Auswahl `Rotate immediately` (Sofort drehen) bei.
 - c. Wählen Sie für `Rotation function` (Drehungsfunktion) `Create a rotation function` (Drehungsfunktion erstellen) und geben Sie dann als Funktionsnamen **tutorial-alternating-users-rotation** ein.
 - d. Wählen Sie für Rotationsstrategie die Option `Alternierende Benutzer` aus und wählen Sie dann unter `Administrator-Anmeldeinformationen-Secret` das Secret mit dem Namen `rds!cluster...` aus, dessen Beschreibung den Namen der Datenbank enthält, die Sie in diesem Tutorial **secretsmanagertutorial** erstellt haben, z. B. `Secret associated with primary RDS DB instance: arn:aws:rds:Region:AccountId:db:secretsmanagertutorial`.
 - e. Wählen Sie `Weiter` aus.
6. Wählen Sie auf der Seite `Review` (Überprüfung) `Store` (Speichern) aus.

Secrets Manager kehrt zur Seite mit den geheimen Details zurück. Oben auf der Seite sehen Sie den Status der Drehungskonfiguration. Secrets Manager erstellt CloudFormation damit Ressourcen wie die Lambda-Rotationsfunktion und eine Ausführungsrolle, die die Lambda-Funktion ausführt. Wenn der Vorgang CloudFormation abgeschlossen ist, ändert sich das Banner in `Secret`, das zur Rotation geplant ist. Die erste Drehung ist abgeschlossen.

Schritt 3: Testen des gedrehten Geheimnisses

Nachdem das Secret rotiert wurde, können Sie überprüfen, ob es gültige Anmeldeinformationen enthält. Das Passwort im Geheimnis hat sich gegenüber den ursprünglichen Anmeldeinformationen geändert.

So rufen Sie das neue Passwort aus dem Geheimnis ab

1. Öffnen Sie die Secrets-Manager-Konsole unter <https://console.aws.amazon.com/secretsmanager/>.
2. Wählen Sie `Secrets` (Geheimnisse) und dann das Geheimnis **SecretsManagerTutorialAppuser** aus.

3. Scrollen Sie auf der Seite Secret details (Geheimnis-Details) nach unten und wählen Sie Retrieve secret value (Geheimnis-Wert abrufen) aus.
4. Kopieren Sie in der Tabelle Key/value (Schlüssel/Wert) den Secret value (Geheimnis-Wert) für **password**.

So testen Sie die Anmeldeinformationen

1. Klicken Sie in MySQL Workbench mit der rechten Maustaste auf die Verbindung SecretsManagerTutorial und wählen Sie dann Verbindung bearbeiten.
2. Geben Sie im Dialogfeld Manage Server Connections (Serververbindungen verwalten) für Username (Benutzername) **appuser** ein und wählen Sie dann Close (Schließen) aus.
3. Zurück in MySQL Workbench wählen Sie die Verbindung SecretsManagerTutorial aus.
4. Fügen Sie im Dialogfeld Open SSH Connection (SSH-Verbindung öffnen) für Password (Passwort) das Passwort ein, das Sie aus dem Geheimnis abgerufen haben, und wählen Sie dann OK aus.

Wenn die Anmeldeinformationen gültig sind, wird MySQL Workbench zur Entwurfsseite für die Datenbank geöffnet.

Dies zeigt, dass die Geheimnis-Drehung erfolgreich ist. Die Anmeldeinformationen im Geheimnis wurden aktualisiert und es ist ein gültiges Passwort für die Verbindung mit der Datenbank vorhanden.

Schritt 4: Bereinigen von Ressourcen

Wenn Sie eine andere Rotationsstrategie ausprobieren möchten, single user rotation (Rotation eines einzelnen Benutzers), überspringen Sie die Bereinigung von Ressourcen und gehen Sie zu [the section called "Einzelbenutzer-Drehung"](#).

Andernfalls, um mögliche Kosten zu vermeiden und die EC2-Instance, die Zugriff auf das Internet hat, zu entfernen, löschen Sie die folgenden Ressourcen, die Sie in diesem Tutorial und seinen Voraussetzungen erstellt haben:

- Amazon-RDS-Datenbank-Instance. Eine Anleitung finden Sie unter [Löschen einer DB-Instance](#) im Amazon-RDS-Benutzerhandbuch.
- Amazon-EC2-Instance. Anweisungen finden Sie unter [Eine Instance beenden](#) im Amazon EC2 Benutzerhandbuch.

- Secrets-Manager-Geheimnis `SecretsManagerTutorialAppuser`. Anweisungen finden Sie unter [the section called “Löschen eines Secrets”](#).
- Secrets-Manager-Endpunkt. Weitere Informationen finden Sie unter [Löschen eines VPC-Endpunkts](#) im AWS PrivateLink -Handbuch.
- VPC-Endpunkt. Weitere Informationen finden Sie unter [Löschen Ihrer VPC](#) im AWS PrivateLink -Handbuch.

Nächste Schritte

- Erfahren Sie, wie Sie [Secrets in Ihren Anwendungen abrufen](#).
- Erfahren Sie mehr über [andere Rotationspläne](#).

Einrichten der Einzelbenutzer-Drehung für AWS Secrets Manager

In diesem Tutorial erfahren Sie, wie Sie eine Einzelbenutzerrotation für ein Secret einrichten, das Datenbankanmeldeinformationen enthält. Einzelbenutzerrotation ist eine Rotationsstrategie, bei der Secrets Manager die Anmeldeinformationen eines einzelnen Benutzers sowohl im Secret als auch in der Datenbank aktualisiert. Weitere Informationen finden Sie unter [the section called “Einzelbenutzer”](#).

Nachdem Sie das Tutorial abgeschlossen haben, empfehlen wir, dass Sie die Ressourcen aus dem Tutorial bereinigen. Verwenden Sie sie nicht in einer Produktionsumgebung.

Die Secrets Manager Manager-Rotation verwendet eine AWS Lambda Funktion, um das Geheimnis und die Datenbank zu aktualisieren. Hinweise zu den Kosten der Verwendung einer Lambda-Funktion finden Sie unter [Preisgestaltung](#).

Inhalt

- [Berechtigungen](#)
- [Voraussetzungen](#)
- [Schritt 1: Erstellen eines Amazon-RDS-Datenbankbenutzers](#)
- [Schritt 2: Erstellen eines Secrets für die Benutzer-Anmeldeinformationen](#)
- [Schritt 3: Testen des rotierten Passworts](#)
- [Schritt 4: Bereinigen von Ressourcen](#)
- [Nächste Schritte](#)

Berechtigungen

Als Teil der Voraussetzungen für das Tutorial benötigen Sie Administratorberechtigungen für Ihr AWS-Konto. In einer Produktionsumgebung ist es eine bewährte Methode, für jeden der Schritte verschiedene Rollen zu verwenden. Beispielsweise würde eine Rolle mit Datenbank-Administratorberechtigungen die Amazon-RDS-Datenbank erstellen, und eine Rolle mit Netzwerk-Administratorberechtigungen würde die VPC und Sicherheitsgruppen einrichten. Für die Tutorial-Schritte empfehlen wir Ihnen, weiterhin dieselbe Identität zu verwenden.

Informationen zum Einrichten von Berechtigungen in einer Produktionsumgebung finden Sie unter [Authentifizierung und Zugriffskontrolle](#).

Voraussetzungen

Voraussetzung für dieses Tutorial ist [the section called “Drehung für wechselnde Benutzer”](#).

Bereinigen Sie die Ressourcen am Ende des ersten Tutorials nicht. Nach diesem Tutorial haben Sie eine realistische Umgebung mit einer Amazon-RDS-Datenbank und einem Secrets-Manager-Secret, das Admin-Anmeldeinformationen für die Datenbank enthält. Sie haben auch ein zweites Secret, das Anmeldeinformationen für einen Datenbankbenutzer enthält, aber Sie verwenden dieses Secret in diesem Tutorial nicht.

Sie haben auch eine Verbindung in MySQL Workbench konfiguriert, um sich mit den Administrator-Anmeldeinformationen mit der Datenbank zu verbinden.

Schritt 1: Erstellen eines Amazon-RDS-Datenbankbenutzers

Zuerst benötigen Sie einen Benutzer, dessen Anmeldeinformationen im Geheimnis gespeichert werden. Um den Benutzer zu erstellen, melden Sie sich bei der Amazon-RDS-Datenbank mit Admin-Anmeldeinformationen an, die in einem Secret gespeichert sind. Der Einfachheit halber erstellen Sie im Tutorial einen Benutzer mit voller Berechtigung für eine Datenbank. In einer Produktionsumgebung ist dies nicht typisch, und wir empfehlen, dem Prinzip der geringsten Berechtigung zu folgen.

So rufen Sie das Admin-Passwort ab

1. Gehen Sie in der Amazon-RDS-Konsole zu Ihrer Datenbank.
2. Wählen Sie auf der Registerkarte Configuration (Konfiguration) unter Master Credentials ARN (Master-Anmeldeinformationen-ARN) die Option Manage in Secrets Manager (In Secrets Manager verwalten) aus.

Die Secrets-Manager-Konsole wird geöffnet.

3. Wählen Sie auf der Seite „Secret details“ (Secret-Details) die Option Retrieve secret value (Secret-Wert abrufen) aus.
4. Das Passwort wird im Abschnitt Secret value (Secret-Wert) angezeigt.

So erstellen Sie einen Datenbankbenutzer

1. Klicken Sie in MySQL Workbench mit der rechten Maustaste auf die Verbindung SecretsManagerTutorial und wählen Sie dann Verbindung bearbeiten.
2. Geben Sie im Dialogfeld Manage Server Connections (Serververbindungen verwalten) für Username (Benutzername) **admin** ein und wählen Sie dann Close (Schließen) aus.
3. Zurück in MySQL Workbench wählen Sie die Verbindung SecretsManagerTutorial aus.
4. Geben Sie das Admin-Passwort ein, das Sie aus dem Secret abgerufen haben.
5. Geben Sie in MySQL Workbench im Fenster Query (Abfragen) die folgenden Befehle (einschließlich eines sicheren Passworts) ein und wählen Sie dann Execute (Ausführen) aus.

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

Im Fenster Output (Ausgabe) sehen Sie, dass die Befehle erfolgreich sind.

Schritt 2: Erstellen eines Secrets für die Benutzer-Anmeldeinformationen

Als Nächstes erstellen Sie ein Secret zum Speichern der Anmeldeinformationen des gerade erstellten Benutzers. Secrets Manager rotiert das Secret, was bedeutet, dass das Passwort programmgesteuert generiert wird – kein Mensch hat dieses neue Passwort gesehen. Da die Rotation sofort beginnt, können Sie auch feststellen, ob die Rotation richtig eingerichtet ist.

1. Öffnen Sie die Secrets-Manager-Konsole unter <https://console.aws.amazon.com/secretsmanager/>.
2. Wählen Sie Store a new secret (Ein neues Secret speichern).
3. Führen Sie auf der Seite Choose secret type (Secret-Typ auswählen) die folgenden Schritte aus:
 - a. Wählen Sie für Secret type (Geheimnistyp) Credentials for Amazon RDS database (Anmeldedaten für die Amazon-RDS-Datenbank) aus.

- b. Geben Sie für Credentials (Anmeldeinformationen) den Benutzernamen **dbuser** und das Passwort ein, das Sie für den Datenbankbenutzer eingegeben haben, den Sie mit MySQL Workbench erstellt haben.
 - c. Wählen Sie für Database (Datenbank) **secretsmanagertutorialdb** aus.
 - d. Wählen Sie Weiter aus.
4. Geben Sie auf der Seite Configure secret (Secret konfigurieren) für Secret name (Secret-Name) **SecretsManagerTutorialDbuser** ein und wählen Sie dann Next (Weiter) aus.
5. Führen Sie auf der Seite Configure rotation (Drehung konfigurieren) die folgenden Schritte aus:
 - a. Schalten Sie die automatische Rotation ein.
 - b. Legen Sie für Rotation schedule (Drehungsplan) einen Zeitplan von Days (Tagen) fest: **2** Tage mit Duration (Dauer): **2h**. Behalten Sie die Auswahl Rotate immediately (Sofort drehen) bei.
 - c. Wählen Sie für Rotation function (Drehungsfunktion) Create a rotation function (Drehungsfunktion erstellen) und geben Sie dann als Funktionsnamen **tutorial-single-user-rotation** ein.
 - d. Wählen Sie für die Rotationsstrategie Einzelbenutzer aus.
 - e. Wählen Sie Weiter aus.
6. Wählen Sie auf der Seite Review (Überprüfung) Store (Speichern) aus.

Secrets Manager kehrt zur Seite mit den geheimen Details zurück. Oben auf der Seite sehen Sie den Status der Drehungskonfiguration. Secrets Manager erstellt CloudFormation damit Ressourcen wie die Lambda-Rotationsfunktion und eine Ausführungsrolle, die die Lambda-Funktion ausführt. Wenn der Vorgang CloudFormation abgeschlossen ist, ändert sich das Banner in Secret, das zur Rotation geplant ist. Die erste Drehung ist abgeschlossen.

Schritt 3: Testen des rotierten Passworts

Nach der ersten Geheimnis-Drehung, die einige Sekunden dauern kann, können Sie überprüfen, ob das Geheimnis immer noch gültige Anmeldeinformationen enthält. Das Passwort im Geheimnis hat sich gegenüber den ursprünglichen Anmeldeinformationen geändert.

So rufen Sie das neue Passwort aus dem Geheimnis ab

1. Öffnen Sie die Secrets-Manager-Konsole unter <https://console.aws.amazon.com/secretsmanager/>.

2. Wählen Sie Secrets (Geheimnisse) und dann das Geheimnis **SecretsManagerTutorialDbuser** aus.
3. Scrollen Sie auf der Seite Secret details (Geheimnis-Details) nach unten und wählen Sie Retrieve secret value (Geheimnis-Wert abrufen) aus.
4. Kopieren Sie in der Tabelle Key/value (Schlüssel/Wert) den Secret value (Geheimnis-Wert) für **password**.

So testen Sie die Anmeldeinformationen

1. Klicken Sie in MySQL Workbench mit der rechten Maustaste auf die Verbindung SecretsManagerTutorial und wählen Sie dann Verbindung bearbeiten.
2. Geben Sie im Dialogfeld Manage Server Connections (Serververbindungen verwalten) für Username (Benutzername) **dbuser** ein und wählen Sie dann Close (Schließen) aus.
3. Zurück in MySQL Workbench wählen Sie die Verbindung SecretsManagerTutorial aus.
4. Fügen Sie im Dialogfeld Open SSH Connection (SSH-Verbindung öffnen) für Password (Passwort) das Passwort ein, das Sie aus dem Geheimnis abgerufen haben, und wählen Sie dann OK aus.

Wenn die Anmeldeinformationen gültig sind, wird MySQL Workbench zur Entwurfsseite für die Datenbank geöffnet.

Schritt 4: Bereinigen von Ressourcen

Um mögliche Gebühren zu vermeiden, löschen Sie das Geheimnis, das Sie in diesem Tutorial erstellt haben. Anweisungen finden Sie unter [the section called “Löschen eines Secrets”](#).

Informationen zum Bereinigen von Ressourcen, die im vorherigen Tutorial erstellt wurden, finden Sie unter [the section called “Schritt 4: Bereinigen von Ressourcen”](#).

Nächste Schritte

- Erfahren Sie, wie Sie Geheimnisse in Ihren Anwendungen abrufen. Siehe [Holen Sie sich Geheimnisse](#).
- Erfahren Sie mehr über andere Drehungspläne. Siehe [the section called “Rotationspläne”](#).

Authentifizierung und Zugriffskontrolle für AWS Secrets Manager

Secrets Manager verwendet [AWS Identity and Access Management \(IAM\)](#), um den Zugriff auf Secrets zu sichern. IAM bietet Authentifizierung und Zugriffskontrolle. Die Authentifizierung verifiziert die Identität der Personen, die Anforderungen senden. Secrets Manager verwendet einen Anmeldeprozess mit Passwörtern, Zugriffsschlüsseln und Multi-Faktor-Authentifizierung (MFA)-Tokens, um die Identität der Benutzer zu überprüfen. Siehe [Anmelden bei AWS](#). Die Zugriffskontrolle stellt sicher, dass nur zugelassene Personen Operationen an AWS -Ressourcen wie z. B. Secrets durchführen können. Der Secrets Manager verwendet Richtlinien, um zu definieren, wer Zugriff auf welche Ressourcen hat und welche Aktionen die Identität für diese Ressourcen ausführen kann. Siehe [Policies and permissions in IAM \(Berechtigungen und Richtlinien in IAM\)](#).

Administratorberechtigungen für den Secrets Manager

Folgen Sie den Anweisungen unter [Adding and removing IAM identity permissions \(Hinzufügen und Entfernen von IAM-Identitätsberechtigungen\)](#), um Secrets Manager Administratorberechtigungen zu gewähren und die folgenden Richtlinien hinzuzufügen:

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

Es wird empfohlen, Endbenutzern keine Administratorberechtigungen zu erteilen. Während dies Ihren Benutzern die Möglichkeit gibt, ihre Secrets zu erstellen und zu verwalten, gewährt die zum Aktivieren der Rotation erforderliche Berechtigung (IAMFullAccess) erhebliche Berechtigungen, die für Endbenutzer nicht geeignet sind.

Berechtigungen für den Zugriff auf Secrets

Durch die Verwendung der IAM-Berechtigungsrichtlinien können Sie steuern, welche Benutzer oder Services Zugriff auf Ihre Secrets haben. Eine Berechtigungsrichtlinie beschreibt, wer welche Aktionen für welche Ressourcen ausführen darf. Sie haben folgende Möglichkeiten:

- [the section called “Anhängen einer Berechtigungsrichtlinie an eine Identität”](#)
- [the section called “Zuordnen einer Berechtigungsrichtlinie zu einem Secret”](#)

Berechtigungen für Lambda Rotationsfunktionen

Secrets Manager verwendet AWS Lambda Funktionen, um [Geheimnisse zu rotieren](#). Die Lambda-Funktion muss Zugriff auf das Secret sowie auf die Datenbank oder den Dienst haben, für den das Secret Anmeldeinformationen enthält. Siehe [Berechtigungen für Rotation](#).

Berechtigungen für die Verschlüsselung

Secrets Manager verwendet AWS Key Management Service (AWS KMS) -Schlüssel, um [Geheimnisse zu verschlüsseln](#). Der hat Von AWS verwalteter Schlüssel `aws/secretsmanager` automatisch die richtigen Berechtigungen. Wenn Sie einen anderen KMS-Schlüssel verwenden, benötigt der Secrets Manager Berechtigungen für diesen Schlüssel. Siehe [the section called "Berechtigungen für den KMS-Schlüssel"](#).

Berechtigungen für die Replikation

Mithilfe von IAM-Berechtigungsrichtlinien steuern Sie, welche Benutzer oder Dienste Ihre geheimen Daten in andere Regionen replizieren können. Siehe [the section called "Replizierung verhindern"](#).

Anhängen einer Berechtigungsrichtlinie an eine Identität

Fügen Sie Berechtigungsrichtlinien an [IAM-Identitäten, Benutzer, Benutzergruppen und Rollen](#) an. Bei einer identitätsbasierten Richtlinie geben Sie an, auf welche Secrets die Identität zugreifen darf und welche Aktionen die Identität für die Secrets ausführen darf. Weitere Informationen finden Sie unter [Hinzufügen und Entfernen von IAM-Identitätsberechtigungen](#).

Sie können Berechtigungen für eine Rolle erteilen, die eine Anwendung oder einen Benutzer in einem anderen Service darstellt. Beispielsweise benötigt eine Anwendung auf einer Amazon-EC2-Instance Zugriff auf eine Datenbank. Sie können eine IAM-Rolle erstellen, die dem EC2-Instance-Profil zugeordnet ist, und dann eine Berechtigungsrichtlinie verwenden, um der Rolle Zugriff auf das Secret zu gewähren, das die Anmeldeinformationen für die Datenbank enthält. Weitere Informationen finden Sie unter [Verwenden einer IAM-Rolle zum Erteilen von Berechtigungen für Anwendungen, die auf Amazon-EC2-Instances ausgeführt werden](#). Andere Services, denen Sie Rollen anhängen können, sind unter anderem [Amazon Redshift](#), [AWS Lambda](#) und [Amazon ECS](#).

Sie können Benutzern, die von einem anderen Identitätssystem als IAM authentifiziert wurden, Berechtigungen erteilen. Sie können z. B. IAM-Rollen Benutzern mobiler Apps zuordnen, die sich

mit Amazon Cognito anmelden. Die Rolle gewährt der App temporäre Anmeldeinformationen mit den Berechtigungen in der Berechtigungsrichtlinie der Rolle. Anschließend können Sie eine Berechtigungsrichtlinie verwenden, um der Rolle Zugriff auf das Secret zu gewähren. Weitere Informationen finden Sie unter [Identitätsanbieter und Verbund](#).

Sie können identitätsbasierte Richtlinien für Folgendes verwenden:

- Gewähren Sie einer Identität Zugriff auf mehrere Secrets.
- Steuern Sie, wer neue Secrets erstellen und auf Secrets zugreifen kann, die noch nicht erstellt wurden.
- Gewähren Sie einer IAM-Gruppe Zugriff auf Secrets.

Weitere Informationen finden Sie unter [the section called “Beispiele für Richtlinien für Berechtigungen”](#).

Zuordnen einer Berechtigungsrichtlinie zu einem AWS Secrets Manager -Secret

Bei einer ressourcenbasierten Richtlinie geben Sie an, wer auf Secrets zugreifen darf und welche Aktionen die Identität für die Secrets ausführen darf. Sie können ressourcenbasierte Richtlinien für Folgendes verwenden:

- Gewähren Sie Zugriff auf ein einzelnes Secret für mehrere Benutzer und Rollen.
- Gewähren Sie Benutzern oder Rollen in anderen AWS Konten Zugriff.

Siehe [the section called “Beispiele für Richtlinien für Berechtigungen”](#).

Wenn Sie eine ressourcenbasierte Richtlinie zu einem Secret in der Konsole zuordnen, verwendet Secrets Manager die Automated-Reasoning-Engine [Zelkova](#) und die API `ValidateResourcePolicy`, um zu verhindern, dass Sie einer breiten Palette von IAM-Prinzipalen Zugriff auf Ihre Secrets gewähren. Alternativ können Sie auch die `PutResourcePolicy`-API mit dem `BlockPublicPolicy`-Parameter von der CLI oder dem SDK aus aufrufen.

Important

Die Überprüfung der Ressourcenrichtlinien und der `BlockPublicPolicy` Parameter tragen zum Schutz Ihrer Ressourcen bei, indem verhindert wird, dass der öffentliche Zugriff über

die Ressourcenrichtlinien gewährt wird, die direkt mit Ihren Geheimnissen verknüpft sind. Zusätzlich zur Nutzung dieser Funktionen sollten Sie die folgenden Richtlinien sorgfältig prüfen, um sicherzustellen, dass sie keinen öffentlichen Zugriff gewähren:

- Identitätsbasierte Richtlinien, die mit zugehörigen AWS Principals verknüpft sind (z. B. IAM-Rollen)
- Ressourcenbasierte Richtlinien, die mit zugehörigen AWS Ressourcen verknüpft sind (z. B. () -Schlüssel) AWS Key Management Service AWS KMS

Informationen zu den Zugriffsberechtigungen für Ihre geheimen Daten finden Sie unter [Bestimmen, wer Berechtigungen für Ihre -Secrets hat](#)

Die Ressourcenrichtlinie für ein Secret anzeigen, ändern oder löschen (Konsole)

1. Öffnen Sie die Secrets-Manager-Konsole unter <https://console.aws.amazon.com/secretsmanager/>.
2. Wählen Sie aus der Liste der Secrets Ihr Secret aus.
3. Wählen Sie auf der Seite zu den Secret-Details auf der Registerkarte Übersicht im Abschnitt Ressourcenberechtigungen die Option Berechtigungen bearbeiten aus.
4. Führen Sie einen der folgenden Schritte im Codefeld aus und wählen Sie dann die Option Save (Speichern):
 - Um eine Ressourcenrichtlinie anzuhängen oder zu ändern, geben Sie die Richtlinie ein.
 - Um die Richtlinie zu löschen, löschen Sie den Inhalt des Codefelds.

AWS CLI

Example Eine Ressourcenrichtlinie abrufen

Im folgenden [get-resource-policy](#)-Beispiel wird die an ein Secret angefügte ressourcenbasierte Richtlinie abgerufen.

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

Example Eine Ressourcenrichtlinie löschen

Im folgenden [delete-resource-policy](#)-Beispiel wird die an ein Secret angefügte ressourcenbasierte Richtlinie gelöscht.

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

Example Eine Ressourcenrichtlinie hinzufügen

Im folgenden [put-resource-policy](#)-Beispiel wird einem Secret eine Berechtigungsrichtlinie hinzugefügt, wobei zunächst geprüft wird, ob die Richtlinie keinen umfassenden Zugriff auf das Secret gewährt. Die Richtlinie wird aus einer Datei gelesen. Weitere Informationen finden Sie im AWS CLI Benutzerhandbuch unter [Laden von AWS CLI Parametern aus einer Datei](#).

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

Inhalt von `mypolicy.json`:

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

AWS SDK

Um die Richtlinie abzurufen, die dem Secret zugeordnet ist, verwenden Sie [GetResourcePolicy](#).

Um die Richtlinie zu löschen, die dem Secret zugeordnet ist, verwenden Sie [DeleteResourcePolicy](#).

Um eine Richtlinie zu einem Secret hinzuzufügen, verwenden Sie [PutResourcePolicy](#). Wenn bereits eine Richtlinie angefügt ist, ersetzt der Befehl sie durch die neue Richtlinie. Die Richtlinie muss als JSON-strukturierter Text formatiert sein. Weitere Informationen finden Sie unter [JSON policy document structure \(JSON-Richtliniendokumentstruktur\)](#). Verwenden Sie [the section called "Beispiele für Richtlinien für Berechtigungen"](#), um mit dem Schreiben Ihrer Richtlinie zu beginnen.

Weitere Informationen finden Sie unter [the section called "AWS SDKs"](#).

AWS verwaltete Richtlinie für AWS Secrets Manager

Eine AWS verwaltete Richtlinie ist eine eigenständige Richtlinie, die von erstellt und verwaltet wird AWS. AWS Verwaltete Richtlinien sind so konzipiert, dass sie Berechtigungen für viele gängige Anwendungsfälle bereitstellen, sodass Sie damit beginnen können, Benutzern, Gruppen und Rollen Berechtigungen zuzuweisen.

Beachten Sie, dass AWS verwaltete Richtlinien für Ihre speziellen Anwendungsfälle möglicherweise keine Berechtigungen mit den geringsten Rechten gewähren, da sie allen AWS Kunden zur Verfügung stehen. Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie [kundenverwaltete Richtlinien](#) definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind.

Sie können die in AWS verwalteten Richtlinien definierten Berechtigungen nicht ändern. Wenn die in einer AWS verwalteten Richtlinie definierten Berechtigungen AWS aktualisiert werden, wirkt sich das Update auf alle Prinzidentitäten (Benutzer, Gruppen und Rollen) aus, denen die Richtlinie zugeordnet ist. AWS aktualisiert eine AWS verwaltete Richtlinie höchstwahrscheinlich, wenn eine neue Richtlinie eingeführt AWS-Service wird oder neue API-Operationen für bestehende Dienste verfügbar werden.

Weitere Informationen finden Sie unter [Von AWS verwaltete Richtlinien](#) im IAM-Benutzerhandbuch.

AWS verwaltete Richtlinie: SecretsManagerReadWrite

Diese Richtlinie gewährt Lese-/Schreibzugriff auf Amazon RDS- AWS Secrets Manager, Amazon Redshift- und Amazon DocumentDB DocumentDB-Ressourcen, einschließlich der Genehmigung zur Beschreibung, sowie die Genehmigung zur Verwendung AWS KMS zum Verschlüsseln und Entschlüsseln von Geheimnissen. Diese Richtlinie gewährt auch die Erlaubnis, AWS CloudFormation Änderungssätze zu erstellen, Rotationsvorlagen aus einem Amazon S3 S3-Bucket abzurufen, der von verwaltet wird AWS, AWS Lambda Funktionen aufzulisten und Amazon EC2 EC2-VPcs

zu beschreiben. Diese Berechtigungen werden in der Konsole benötigt, um die Rotation mit vorhandenen Rotationsfunktionen einzurichten.

Um neue Rotationsfunktionen zu erstellen, benötigen Sie auch die Erlaubnis, AWS CloudFormation Stacks und AWS Lambda Ausführungsrollen zu erstellen. Sie können die von [IAM FullAccess verwaltete Richtlinie](#) zuweisen. Siehe [Berechtigungen für Rotation](#).

Details zu Berechtigungen

Diese Richtlinie umfasst die folgenden Berechtigungen.

- `secretsmanager` – Hiermit können Prinzipale alle Secrets-Manager-Aktionen ausführen.
- `cloudformation`— Ermöglicht Prinzipalen das Erstellen AWS CloudFormation von Stacks. Dies ist erforderlich, damit Principals, die die Konsole zum Aktivieren der Rotation verwenden, Lambda-Rotationsfunktionen über AWS CloudFormation Stacks erstellen können. Weitere Informationen finden Sie unter [the section called “So verwendet Secrets Manager AWS CloudFormation”](#).
- `ec2` – Hiermit können Prinzipale Amazon-EC2-VPCs beschreiben. Dies ist erforderlich, damit Prinzipale, die die Konsole nutzen, Rotationsfunktionen in derselben VPC erstellen können wie die Datenbank der Anmeldeinformationen, die sie in einem Secret speichern.
- `kms`— Ermöglicht Prinzipalen die Verwendung von AWS KMS Schlüsseln für kryptografische Operationen. Dies ist erforderlich, damit Secrets Manager Secrets ver- und entschlüsseln kann. Weitere Informationen finden Sie unter [the section called “Ver- und Entschlüsselung von Secrets”](#).
- `lambda` – Hiermit können Prinzipale Lambda-Rotationsfunktionen auflisten. Dies ist erforderlich, damit Prinzipale, die die Konsole nutzen, vorhandene Rotationsfunktionen auswählen können.
- `rds` – Hiermit können Prinzipale Cluster und Instances in Amazon RDS beschreiben. Dies ist erforderlich, damit Prinzipale, die die Konsole nutzen, Amazon-RDS-Cluster oder -Instances auswählen können.
- `redshift` – Hiermit können Prinzipale Cluster in Amazon Redshift beschreiben. Dies ist erforderlich, damit Prinzipale, die die Konsole nutzen, Amazon-Redshift-Cluster auswählen können.
- `redshift-serverless`— Ermöglicht Prinzipalen die Beschreibung von Namespaces in Amazon Redshift Serverless. Dies ist erforderlich, damit Prinzipale, die die Konsole verwenden, Amazon Redshift Serverless-Namespaces auswählen können.
- `docdb-elastic` – Hiermit können Prinzipale elastische Cluster in Amazon DocumentDB beschreiben. Dies ist erforderlich, damit Prinzipale, die die Konsole nutzen, elastische Amazon-DocumentDB-Cluster auswählen können.

- `tag` – Hiermit können Prinzipale alle getaggten Ressourcen im Konto abrufen.
- `serverlessrepo`— Ermöglicht Prinzipalen das Erstellen von Änderungssätzen. AWS CloudFormation Dies ist erforderlich, damit Prinzipale, die die Konsole nutzen, Lambda-Rotationsfunktionen erstellen können. Weitere Informationen finden Sie unter [the section called “So verwendet Secrets Manager AWS CloudFormation”](#).
- `s3`— Ermöglicht Prinzipalen das Abrufen von Objekten aus einem Amazon S3 S3-Bucket, der von AWS verwaltet wird. Dieser Bucket enthält Lambda [Rotationsfunktionsvorlagen](#). Diese Berechtigung ist erforderlich, damit Prinzipale, die die Konsole nutzen, Lambda-Rotationsfunktionen auf Grundlage der Vorlagen im Bucket erstellen können. Weitere Informationen finden Sie unter [the section called “So verwendet Secrets Manager AWS CloudFormation”](#).

Die Richtlinie finden Sie im [SecretsManagerReadWrite JSON-Richtliniendokument](#).

Secrets Manager Manager-Updates für AWS verwaltete Richtlinien

Sehen Sie sich Details zu Aktualisierungen der AWS verwalteten Richtlinien für Secrets Manager an.

Änderung	Beschreibung	Datum
SecretsManagerReadWrite – Aktualisierung auf eine bestehende Richtlinie	Diese Richtlinie wurde aktualisiert, um den beschreibenden Zugriff auf Amazon Redshift Serverless zu ermöglichen, sodass Konsolenbenutzer einen Amazon Redshift Serverless-Namespace wählen können, wenn sie einen Amazon Redshift Secret erstellen.	12. März 2024
SecretsManagerReadWrite – Aktualisierung auf eine bestehende Richtlinie	Diese Richtlinie wurde aktualisiert, um den Beschreibungszugriff auf elastische Amazon-DocumentDB-Cluster zu ermöglichen, sodass Konsolenbenutzer	12. September 2023

Änderung	Beschreibung	Datum
	beim Erstellen eines Amazon-DocumentDB-Secrets einen elastischen Cluster auswählen können.	
SecretsManagerReadWrite – Aktualisierung auf eine bestehende Richtlinie	Diese Richtlinie wurde aktualisiert, um den Beschreibungszugriff auf Amazon Redshift zu ermöglichen, sodass Konsolenbenutzer beim Erstellen eines Amazon-Redshift-Secrets einen Amazon-Redshift-Cluster auswählen können. Das Update fügte auch neue Berechtigungen hinzu, um den Lesezugriff auf einen Amazon S3 S3-Bucket zu ermöglichen, der verwaltet wird AWS, in dem die Lambda-Rotationsfunktionsvorlagen gespeichert sind.	24. Juni 2020
SecretsManagerReadWrite – Aktualisierung auf eine bestehende Richtlinie	Diese Richtlinie wurde aktualisiert, um den Beschreibungszugriff auf Amazon-RDS-Cluster zu ermöglichen, sodass Konsolenbenutzer beim Erstellen eines Amazon-RDS-Secrets einen Cluster auswählen können.	3. Mai 2018

Änderung	Beschreibung	Datum
SecretsManagerReadWrite – Neue Richtlinie.	Für Secrets Manager wurde eine Richtlinie erstellt, um Berechtigungen zu gewähren, die für die Nutzung der Konsole mit vollständigem Lese-/Schreibzugriff auf Secrets Manager erforderlich sind.	04. April 2018
Beginn der Verfolgung von Änderungen in Secrets Manager	Secrets Manager begann, Änderungen an seinen AWS verwalteten Richtlinien zu verfolgen.	04. April 2018

Bestimmen, wer Berechtigungen für Ihre AWS Secrets Manager-Secrets hat

Standardmäßig haben IAM-Identitäten keine Berechtigung für den Zugriff auf Secrets. Bei der Autorisierung des Zugriffs auf ein Secret bewertet der Secrets Manager die dem Secret angefügte Secret-Richtlinie und alle identitätsbasierten Richtlinien, die dem IAM-Benutzer oder der Rolle angefügt sind, der/die die Anforderung sendet. Zu diesem Zweck verwendet der Secrets Manager einen Prozess ähnlich dem in [Determining whether a request is allowed or denied \(Ermitteln, ob eine Anforderung erlaubt oder verweigert wird\)](#) im IAM-Benutzerhandbuch beschriebenen.

Wenn mehrere Richtlinien auf eine Anforderung angewendet werden, verwendet Secrets Manager eine Hierarchie zum Steuern von Berechtigungen:

1. Wenn eine Anweisung in einer Richtlinie mit einem expliziten deny der Anforderungsaktion und Ressource entspricht:

Explizite deny-Codes überschreiben alles andere und blockieren die Aktion.
2. Wenn es keinen expliziten deny-Code gibt aber eine Anweisung mit einem expliziten allow-Code der Anforderungsaktion und Ressource entspricht:

Der explizite `allow`-Code gewährt der Aktion in der Anforderung Zugriff auf die Ressourcen in der Anweisung.

Wenn sich die Identität und das Secret in zwei verschiedenen Konten befinden, muss ein `allow`-Code sowohl in der Ressourcenrichtlinie für das Secret als auch in der Richtlinie, die mit der Identität verknüpft ist, vorhanden sein, da AWS ansonsten die Anforderung ablehnt. Weitere Informationen finden Sie unter [Kontoübergreifender Zugriff](#).

3. Wenn es keine Anweisung mit einem expliziten `allow`-Code gibt, die der Anforderungsaktion und Ressource entspricht:

AWS verweigert die Anforderung standardmäßig. Dies wird auch als implizite Zugriffsverweigerung bezeichnet.

Die ressourcenbasierte Richtlinie für ein Secret anzeigen

- Führen Sie eine der folgenden Aktionen aus:
 - Öffnen Sie die Secrets-Manager-Konsole unter <https://console.aws.amazon.com/secretsmanager/>. Klicken Sie auf der Detail-Seite für Ihr Secret im Abschnitt Resource permissions (Ressourcenberechtigungen) auf Edit permissions (Berechtigungen bearbeiten).
 - Verwenden der AWS CLI zum Aufruf von [get-resource-policy](#) oder des AWS SDK zum Aufruf von [GetResourcePolicy](#).

Bestimmen, wer über identitätsbasierte Richtlinien Zugriff hat

- Verwenden Sie den IAM-Richtliniensimulator. Weitere Informationen finden Sie unter [Testen IAM policies with the IAM policy simulator \(Testen von IAM-Richtlinien mit dem IAM-Richtliniensimulator\)](#).

Greifen Sie von einem AWS Secrets Manager anderen Konto aus auf Geheimnisse zu

So können Sie Benutzern in einem Konto den Zugriff auf Secrets in einem anderen Konto gewähren (Kontoübergreifender Zugriff). Sie müssen den Zugriff sowohl in einer Ressourcenrichtlinie als auch

in einer Identitätsrichtlinie zulassen. Dies unterscheidet sich von dem Gewähren des Zugriffs auf Identitäten in demselben Konto wie das Secret.

Sie müssen auch zulassen, dass die Identität den KMS-Schlüssel verwendet, mit dem das Secret verschlüsselt ist. Das liegt daran, dass Sie das Von AWS verwaltete Schlüssel (aws/secretsmanager) nicht für den kontoübergreifenden Zugriff verwenden können. Stattdessen müssen Sie Ihr Secret mit einem von Ihnen erstellten KMS-Schlüssel verschlüsseln und ihm dann eine Schlüsselrichtlinie anhängen. Für die Erstellung von KMS-Schlüsseln fällt eine Gebühr an. Informationen zum Ändern des Verschlüsselungsschlüssels für ein Secret finden Sie unter [the section called "Ändern eines Secrets"](#).

In den folgenden Beispielrichtlinien wird davon ausgegangen, dass Sie ein Secret und einen Verschlüsselungsschlüssel in Konto1 und eine Identität in Konto2 besitzen, womit Sie auf den Secret-Wert zugreifen möchten.

Schritt 1: Fügen Sie dem Secret in Account1 eine Ressourcen-Richtlinie hinzu

- *Die folgende Richtlinie ermöglicht ApplicationRole in Account2 den Zugriff auf das Geheimnis in Account1.* Informationen zur Verwendung dieser Richtlinie finden Sie unter [the section called "Zuordnen einer Berechtigungsrichtlinie zu einem Secret"](#).

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

Schritt 2: Fügen Sie der Schlüsselrichtlinie für den KMS-Schlüssel in Account1 eine Anweisung hinzu

- *Die folgende wichtige Richtlinienanweisung ermöglicht es ApplicationRole in Account2, den KMS-Schlüssel in Account1 zu verwenden,*

um das Geheimnis in Account1 zu entschlüsseln. Um diese Anweisung zu verwenden, fügen Sie sie der Schlüsselrichtlinie für Ihren KMS-Schlüssel hinzu. Weitere Informationen finden Sie unter [Changing a key policy \(Ändern einer Schlüsselrichtlinie\)](#).

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

Schritt 3: Hängen Sie eine Identitätsrichtlinie an die Identität in Account2 an

- *Die folgende Richtlinie ermöglicht es Account2, auf das Geheimnis ApplicationRole in Account1 zuzugreifen und den geheimen Wert mithilfe des Verschlüsselungsschlüssels zu entschlüsseln, der sich ebenfalls in Account1 befindet.* Informationen zur Verwendung dieser Richtlinie finden Sie unter [the section called "Anhängen einer Berechtigungsrichtlinie an eine Identität"](#). Sie finden den ARN für Ihr Secret in der Secrets-Manager-Konsole auf der Seite mit den Secret-Details unter Secret ARN. Alternativ können Sie [describe-secret](#) aufrufen.

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

```
}
```

Greifen Sie von einer lokalen Umgebung aus auf geheime Daten zu

Sie können AWS Identity and Access Management Roles Anywhere verwenden, um temporäre Sicherheitsanmeldeinformationen in IAM für Workloads wie Server, Container und Anwendungen abzurufen, die außerhalb von ausgeführt werden. AWS Ihre Workloads können dieselben IAM-Richtlinien und IAM-Rollen verwenden, die Sie mit AWS Anwendungen für den Zugriff auf Ressourcen verwenden. AWS Mit IAM Roles Anywhere können Sie Secrets Manager verwenden, um Anmeldeinformationen zu speichern und zu verwalten, auf die Ressourcen sowohl auf AWS als auch auf lokalen Geräten wie Anwendungsservern zugreifen können. Weitere Informationen finden Sie im [Benutzerhandbuch zu IAM Roles Anywhere](#).

Beispiele für Berechtigungsrichtlinien für AWS Secrets Manager

Eine Berechtigungsrichtlinie ist JSON-strukturierter Text. Weitere Informationen finden Sie unter [JSON policy document structure \(JSON-Richtliniendokumentstruktur\)](#).

Berechtigungsrichtlinien, die Sie Ressourcen und Identitäten anhängen, sind sehr ähnlich. Einige Elemente, die Sie in eine Richtlinie für den Zugriff auf Secrets einschließen, umfassen:

- **Principal:** wem der Zugriff gewährt werden soll. Weitere Informationen finden Sie unter [Specify a principal \(Angaben eines Prinzipals\)](#) im IAM-Benutzerhandbuch. Wenn Sie eine Richtlinie an eine Identität anfügen, fügen Sie kein Principal-Element zur Richtlinie hinzu.
- **Action:** was sie tun können. Siehe [the section called “Secrets-Manager-Aktionen”](#).
- **Resource:** auf welche Geheimnisse sie zugreifen können. Siehe [the section called “Secrets-Manager-Ressourcen”](#).

Das Platzhalterzeichen (*) hat je nachdem, was Sie an die Richtlinie anhängen, eine andere Bedeutung:

- In einer Richtlinie, die einem Secret zugeordnet ist, bedeutet *, dass die Richtlinie auf dieses Secret angewendet wird.
- In einer Richtlinie, die einer Identität zugeordnet ist, bedeutet *, dass die Richtlinie für alle Ressourcen, einschließlich Secrets, im Konto gilt.

Informationen zum Anfügen einer Richtlinie an ein Secret finden Sie unter [the section called “Zuordnen einer Berechtigungsrichtlinie zu einem Secret”](#).

So können Sie eine Richtlinie an eine [the section called “Anhängen einer Berechtigungsrichtlinie an eine Identität”](#)-Identität anfügen.

Themen

- [Beispiel: Berechtigung zum Abrufen von einzelnen Secret-Werten](#)
- [Beispiel: Erlaubnis, einzelne Geheimnisse zu lesen und zu beschreiben](#)
- [Beispiel: Berechtigung zum Abrufen einer Gruppe geheimer Werte in einem Batch](#)
- [Beispiel: Platzhalter](#)
- [Beispiel: Berechtigung zum Erstellen von Secrets](#)
- [Beispiel: Verweigern Sie einen bestimmten AWS KMS Schlüssel zur Verschlüsselung von Geheimnissen](#)
- [Beispiel: Berechtigungen und VPCs](#)
- [Beispiel: Steuern des Zugriffs auf Secrets mit Tags](#)
- [Beispiel: Beschränken Sie den Zugriff auf Identitäten mit Tags, die mit den Tags der Secrets übereinstimmen](#)
- [Beispiel: Service-Prinzipal](#)

Beispiel: Berechtigung zum Abrufen von einzelnen Secret-Werten

Um die Berechtigung zum Abrufen von Secret-Werten zu erteilen, können Sie Secrets oder Identitäten Richtlinien zuordnen. Hilfe zum Festlegen des zu verwendenden Richtlinientyps finden Sie unter [Identity-based policies and resource-based policies \(Identitätsbasierte Richtlinien und ressourcenbasierte Richtlinien\)](#). Informationen zum Erstellen von Richtlinien finden Sie unter [the section called “Zuordnen einer Berechtigungsrichtlinie zu einem Secret”](#) und [the section called “Anhängen einer Berechtigungsrichtlinie an eine Identität”](#).

Die folgenden Beispiele zeigen zwei verschiedene Möglichkeiten, Zugriff auf ein Secret zu gewähren. Das erste Beispiel ist eine ressourcenbasierte Richtlinie, die Sie an ein Secret anfügen können. Dieses Beispiel ist nützlich, wenn Sie mehreren Benutzern oder Rollen Zugriff auf ein einzelnes Secret gewähren möchten. Das zweite Beispiel ist eine identitätsbasierte Richtlinie, die Sie einem Benutzer oder einer Rolle in IAM zuordnen können. Dieses Beispiel ist nützlich, wenn Sie Zugriff auf

eine IAM-Gruppe gewähren möchten. Informationen zum Gewähren der Berechtigung zum Abrufen einer Gruppe von Secrets in einem Batch-API-Aufruf finden Sie unter [the section called “Beispiel: Berechtigung zum Abrufen einer Gruppe geheimer Werte in einem Batch”](#).

Example Lesen Sie eines Secrets (Zuordnung zu einem Secret)

Sie können Zugriff auf ein Secret gewähren, indem Sie die folgende Richtlinie an das Secret anfügen. Informationen zur Verwendung dieser Richtlinie finden Sie unter [the section called “Zuordnen einer Berechtigungsrichtlinie zu einem Secret”](#).

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

Example Auslesen eines Secrets, das mit einem vom Kunden verwalteten Schlüssel verschlüsselt ist (Zuordnung zu Identität)

Wenn ein Secret mit einem vom Kunden verwalteten Schlüssel verschlüsselt ist, können Sie Zugriff zum Auslesen des Secrets gewähren, indem Sie die folgende Richtlinie an eine Identität anfügen. Informationen zur Verwendung dieser Richtlinie finden Sie unter [the section called “Anhängen einer Berechtigungsrichtlinie an eine Identität”](#).

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

```

    "Effect": "Allow",
    "Action": "kms:Decrypt",
    "Resource": "KMSKeyARN"
  }
]
}

```

Beispiel: Erlaubnis, einzelne Geheimnisse zu lesen und zu beschreiben

Example Lies und beschreibe ein Geheimnis (füge es einer Identität hinzu)

Sie können Zugriff auf ein Secret gewähren, indem Sie die folgende Richtlinie an eine Identität anfügen. Informationen zur Verwendung dieser Richtlinie finden Sie unter [the section called “Anhängen einer Berechtigungsrichtlinie an eine Identität”](#).

```

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

```

Beispiel: Berechtigung zum Abrufen einer Gruppe geheimer Werte in einem Batch

Example Lesen einer Gruppe von Secrets in einem Batch (Anhang an Identität)

Sie können den Zugriff auf den Abruf einer Gruppe von Secrets in einem Batch-API-Aufruf gewähren, indem Sie die folgende Richtlinie an eine Identität anhängen. Die Richtlinie schränkt den Aufrufer so ein, dass er nur die von *SecretARN1*, *SecretARN2* und *SecretARN3* angegebenen Secrets abrufen kann, auch wenn der Batch-Aufruf andere Secrets enthält. Wenn der Aufrufer im Batch-API-Aufruf auch andere Secrets anfordert, gibt Secrets Manager diese nicht zurück. Weitere Informationen finden Sie unter [BatchGetSecretValue](#). Informationen zur Verwendung dieser

Richtlinie finden Sie unter [the section called “Anhängen einer Berechtigungsrichtlinie an eine Identität”](#).

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

Beispiel: Platzhalter

Sie können Platzhalter verwenden, um einen Satz von Werten in ein Richtlinienelement aufzunehmen.

Example Zugriff auf alle Secrets in einem Pfad (Zuordnung zu Identität)

Die folgende Richtlinie gewährt Zugriff auf das Abrufen aller Geheimnisse, deren Name mit *TestEnv/* beginnt. Informationen zur Verwendung dieser Richtlinie finden Sie unter [the section called “Anhängen einer Berechtigungsrichtlinie an eine Identität”](#).

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

```

    "Effect": "Allow",
    "Action": "secretsmanager:GetSecretValue",
    "Resource": "arn:aws:secretsmanager:Region:AccountId:secret:TestEnv/*"
  }
}

```

Example Zugriff auf Metadaten in allen Secrets (Zuordnung zu Identität)

Die folgenden Richtlinien gewährt DescribeSecret und Berechtigungen beginnend mit List: ListSecrets und ListSecretVersionIds. Informationen zur Verwendung dieser Richtlinie finden Sie unter [the section called “Anhängen einer Berechtigungsrichtlinie an eine Identität”](#).

```

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

```

Example Zuordnen eines Secret-Namen (Zuordnung zu Identität)

Die folgende Richtlinie gewährt allen Secrets-Manager-Berechtigungen für ein Secret nach Namen. Informationen zur Verwendung dieser Richtlinie finden Sie unter [the section called “Anhängen einer Berechtigungsrichtlinie an eine Identität”](#).

Um einen Secret-Namen zuzuordnen, erstellen Sie den ARN für das Secret, indem Sie Region, Konto-ID, Secret-Namen und Platzhalter (?) zusammenstellen, um einzelne zufällige Zeichen zuzuordnen. Secrets Manager fügt sechs zufällige Zeichen zu geheimen Namen als Teil ihres ARN an, sodass Sie diesen Platzhalter verwenden können, um diese Zeichen zu vergleichen. Bei Verwendung der Syntax "another_secret_name-*" gleicht Secrets Manager nicht nur das vorgesehene Secret mit den 6 zufälligen Zeichen, sondern auch ""another_secret_name-<anything-here>a1b2c3"" ab.

Da Sie alle Teile des ARN eines Secret mit Ausnahme der 6 zufälligen Zeichen vorhersagen können, ermöglicht Ihnen das Platzhalterzeichen '?????' das sichere Erteilen von Berechtigungen für ein

noch nicht bestehendes Secret. Beachten Sie jedoch Folgendes: Wenn Sie das Secret löschen und mit demselben Namen neu erstellen, erhält der Benutzer automatisch die Berechtigung für das neue Secret, auch wenn die 6 Zeichen geändert wurden.

```
{
  "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-??????"
      ]
    }
  ]
}
```

Beispiel: Berechtigung zum Erstellen von Secrets

Um einem Benutzer Berechtigungen für die Erstellung eines Secrets zu gewähren, empfehlen wir Ihnen, eine Berechtigungsrichtlinie an eine IAM-Gruppe anzufügen, der der Benutzer angehört. Weitere Informationen zu [IAM-Benutzergruppen](#).

Example Erstellen von Secrets (Zuordnung zu Identität)

Die folgende Richtlinie erteilt die Berechtigung zum Erstellen von Secrets und zum Anzeigen einer Liste von Secrets. Informationen zur Verwendung dieser Richtlinie finden Sie unter [the section called "Anhängen einer Berechtigungsrichtlinie an eine Identität"](#).

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

```
]
}
```

Beispiel: Verweigern Sie einen bestimmten AWS KMS Schlüssel zur Verschlüsselung von Geheimnissen

Important

Um einem vom Kunden verwalteten Schlüssel zu verweigern, empfehlen wir Ihnen, den Zugriff mithilfe einer Schlüsselrichtlinie oder einer Schlüsselgewährung einzuschränken. Weitere Informationen finden Sie AWS KMS im AWS Key Management Service Entwicklerhandbuch unter [Authentifizierung und Zugriffskontrolle für](#).

Example Den AWS verwalteten Schlüssel ablehnen **aws/secretsmanager** (an die Identität anhängen)

Die folgende Richtlinie zeigt, wie die Verwendung des AWS verwalteten Schlüssels `aws/secretsmanager` für die Erstellung oder Aktualisierung von Geheimnissen verweigert werden kann. Das bedeutet, dass Geheimnisse mit einem vom Kunden verwalteten Schlüssel verschlüsselt werden müssen. Wenn der `aws/secretsmanager` Schlüssel existiert, müssen Sie auch seine Schlüssel-ID angeben. Sie fügen auch die leere Zeichenfolge hinzu, da Secrets Manager diese als AWS verwalteten Schlüssel `aws/secretsmanager` interpretiert. Die zweite Anweisung lehnt Anfragen zur Erstellung von Geheimnissen ab, die keinen KMS-Schlüssel enthalten, da Secrets Manager diesen als AWS verwalteten Schlüssel interpretiert. `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"
        }
    }
}
]
}

```

Beispiel: Berechtigungen und VPCs

Wenn Sie innerhalb einer VPC auf Secrets Manager zugreifen müssen, können Sie sicherstellen, dass Anforderungen an Secrets Manager von der VPC stammen, indem Sie eine Bedingung in Ihre Berechtigungsrichtlinien aufnehmen. Weitere Informationen finden Sie unter [VPC-Endpointbedingungen](#) und [VPC-Endpoint](#).

Stellen Sie sicher, dass Anfragen für den Zugriff auf das Secret von anderen AWS Diensten auch von der VPC kommen, da ihnen diese Richtlinie andernfalls den Zugriff verweigert.

Example Anforderungen müssen über einen VPC-Endpoint kommen (Zuordnung zu Secret)

Die folgende Richtlinie erlaubt es einem Benutzer z. B. nur dann, Secrets-Manager-Operationen auszuführen, wenn die Anforderung durch den angegebenen VPC-Endpoint *vpce-1234a5678b9012c* geleitet wird. Informationen zur Verwendung dieser Richtlinie finden Sie unter [the section called “Zuordnen einer Berechtigungsrichtlinie zu einem Secret”](#).

```

{
    "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 Anforderungen müssen über einen VPC kommen (Zuordnung zu Secret)

Die folgende Richtlinie lässt nur dann Befehle zur Erstellung und Verwaltung von Secrets zu, wenn sie von *vpce-12345678* stammen. Darüber hinaus erlaubt die Richtlinie Operationen, die auf den verschlüsselten Wert des Secrets zugreifen, nur dann, wenn die Anforderungen von *vpc-2b2b2b2b* stammen. Sie verwenden eine solche Richtlinie wie diese möglicherweise, wenn Sie eine Anwendung in einer VPC ausführen, aber eine zweite, isolierte VPC für die Verwaltungsfunktionen genutzt wird. Informationen zur Verwendung dieser Richtlinie finden Sie unter [the section called “Zuordnen einer Berechtigungsrichtlinie zu einem Secret”](#).

```

{
  "Id": "example-policy-2",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAdministrativeActionsfromONLYVpce-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"
    }
  }
}
]
}

```

Beispiel: Steuern des Zugriffs auf Secrets mit Tags

Sie können Tags verwenden, um den Zugriff auf Ihre Secrets zu steuern. Die Verwendung von Tags zur Kontrolle von Berechtigungen in Umgebungen, die schnell wachsen, ist hilfreich und unterstützt Sie in Situationen, in denen die Richtlinienverwaltung mühsam wird. Eine Strategie besteht darin, Tags Secrets zuzuordnen und dann Berechtigungen für eine Identität zu erteilen, wenn ein Secret ein bestimmtes Tag hat.

Example Gewähren von Zugriff auf Secrets mit einem bestimmten Tag (Zuordnung zu Identität)

Die folgende Richtlinie erlaubt geheime DescribeSecret Daten mit einem Tag mit dem Schlüssel "" und dem Wert *ServerName*"*ServerABC*". Informationen zur Verwendung dieser Richtlinie finden Sie unter [the section called "Anhängen einer Berechtigungsrichtlinie an eine Identität"](#).

```
{
```

```

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

```

Beispiel: Beschränken Sie den Zugriff auf Identitäten mit Tags, die mit den Tags der Secrets übereinstimmen

Eine Strategie besteht darin, Tags sowohl Secrets als auch IAM-Identitäten zuzuordnen. Anschließend erstellen Sie Berechtigungsrichtlinien, um Operationen zuzulassen, wenn das Tag der Identität mit dem Tag des Secret übereinstimmt. Ein vollständiges Tutorial finden Sie unter [Define permissions to access secrets based on tags \(Definieren Sie Berechtigungen für den Zugriff auf Secrets basierend auf Tags\)](#).

Die Verwendung von Tags zur Kontrolle von Berechtigungen in Umgebungen, die schnell wachsen, ist hilfreich und unterstützt Sie in Situationen, in denen die Richtlinienverwaltung mühsam wird. Weitere Informationen finden Sie unter [Was ist ABAC für AWS?](#)

Example Zugriff auf Rollen zulassen, die dieselben Tags wie Secrets haben (Zuordnung zu Secret)

Die folgende Richtlinie gewährt GetSecretValue Zugriff auf das Konto **123456789012**, sofern das Tag **AccessProject** den gleichen Wert für das Geheimnis und die Rolle aufweist. Informationen zur Verwendung dieser Richtlinie finden Sie unter [the section called “Zuordnen einer Berechtigungsrichtlinie zu einem Secret”](#).

```

{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {
      "AWS": "123456789012"
    }
  },

```

```

"Condition": {
  "StringEquals": {
    "aws:ResourceTag/AccessProject": "${ aws:PrincipalTag/AccessProject }"
  }
},
"Action": "secretsmanager:GetSecretValue",
"Resource": "*"
}
}

```

Beispiel: Service-Prinzipal

Wenn die mit Ihrem Secret verknüpfte Ressourcenrichtlinie einen [AWS Service Principal](#) beinhaltet, empfehlen wir Ihnen, die SourceAccount globalen [Bedingungsschlüssel aws: SourceArn](#) und [aws:](#) zu verwenden. Die ARN- und Kontowerte werden nur dann in den Autorisierungskontext aufgenommen, wenn eine Anforderung von einem anderen AWS -Service an Secrets Manager eingeht. Diese Kombination von Bedingungen vermeidet ein potenziell [verwirrtes Stellvertreterszenario](#).

Wenn ein Ressourcen-ARN Zeichen enthält, die in einer Ressourcenrichtlinie nicht zugelassen sind, können Sie diesen Ressourcen-ARN nicht im Wert des `aws: SourceArn`-Bedingungsschlüssel verwenden. Verwenden Sie stattdessen den Bedingungsschlüssel `aws: SourceAccount`. Weitere Informationen finden Sie unter [IAM-Anforderungen](#).

Dienstprinzipale werden normalerweise nicht als Prinzipale in einer Richtlinie verwendet, die an ein Geheimnis angehängt ist, aber für einige AWS Dienste ist dies erforderlich. Informationen zu Ressourcenrichtlinien, die ein Service an ein Geheimnis anhängen muss, finden Sie in der Dokumentation des Services.

Example Erlauben Sie einem Service den Zugriff auf ein Geheimnis mit einem Serviceprinzipal (an ein Geheimnis anhängen)

```

{
  "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"
      }
    }
  }
]
}
```

Berechtigungsreferenz für AWS Secrets Manager

Die Elemente, aus denen eine Berechtigungsrichtlinie besteht, finden Sie in der [JSON-Richtliniendokumentstruktur](#) und in der [Referenz der IAM-JSON-Richtlinienelemente](#).

Informationen zum Erstellen einer eigenen Berechtigungsrichtlinie finden Sie unter [the section called "Beispiele für Richtlinien für Berechtigungen"](#).

Die Spalte Resource types (Ressourcentypen) der Aktionstabelle gibt an, ob die Aktion Berechtigungen auf Ressourcenebene unterstützt. Wenn es keinen Wert für diese Spalte gibt, müssen Sie alle Ressourcen ("*") im Element Resource Ihrer Richtlinienanweisung angeben. Wenn die Spalte einen Ressourcentyp enthält, können Sie einen ARN dieses Typs in einer Anweisung mit dieser Aktion angeben. Wenn für die Aktion eine oder mehrere Ressourcen erforderlich sind, muss der Aufrufer die Erlaubnis haben, die Aktion mit diesen Ressourcen zu verwenden. Erforderliche Ressourcen sind in der Tabelle mit einem Sternchen (*) gekennzeichnet. Wenn Sie den Ressourcenzugriff mit dem Element Resource in einer IAM-Richtlinie einschränken, müssen Sie für jeden erforderlichen Ressourcentyp einen ARN oder ein Muster angeben. Einige Aktionen unterstützen mehrere Ressourcentypen. Wenn der Ressourcentyp optional ist (nicht als erforderlich angegeben), können Sie sich für einen der optionalen Ressourcentypen entscheiden.

Die Spalte Bedingungsschlüssel der Tabelle der Aktionen enthält Schlüssel, die Sie im Element Condition einer Richtlinienanweisung angeben können. Weitere Informationen zu den Bedingungsschlüsseln, die den Ressourcen für den Service zugeordnet sind, finden Sie in der Spalte Bedingungsschlüssel der Tabelle der Ressourcentypen.

Secrets-Manager-Aktionen

Aktionen	Beschreibung	Zugriffsberechtigungen	Ressourcentypen (*erforderlich)	Bedingungs-schlüssel	Abhängige Aktionen
BatchGetSecretValue	Gewährt die Berechtigung zum Abrufen und Entschlüsseln einer Liste von Geheimnissen	Auflisten			
CancelRotateSecret	Gewährt die Berechtigung zum Abbrechen einer laufenden Geheimnisrotation	Schreiben	Secret*	secretsmanager:SecretId secretsmanager:resource/AllowRotationLambdaAction secretsmanager:ResourceTag/tag-key aws:ResourceTag/\${TagKey} secretsmanager:SecretPrimaryRegion	

Aktionen	Beschreibung	Zugriffsbene	Ressourcentypen (*erforderlich)	Bedingungschlüssel	Abhängige Aktionen
CreateSecret	<p>Gewährt die Berechtigung zum Erstellen eines Geheimnisses, das verschlüsselte Daten speichert, die abgefragt und gedreht werden können</p>	<p>Schreiben</p>	<p>Secret*</p>	<p>secretsmanager:Name secretsmanager:Description secretsmanager:KmsKeyId aws:RequestTag/\${TagKey} aws:ResourceTag/\${TagKey} aws:TagKeys secretsmanager:ResourceTag/tag-key secretsmanager:AddReplicaRegions</p>	

Aktionen	Beschreibung	Zugriffsbene	Ressourcentypen (*erforderlich)	Bedingungschlüssel	Abhängige Aktionen
				secretsmanager:ForceOverwriteReplicaSecret	
DeleteResourcePolicy	Gewährt die Berechtigung zum Löschen der Ressourcenrichtlinie, die an ein Geheimnis angefügt ist	Berechtigungsverwaltung	Secret*	secretsmanager:SecretId secretsmanager:resource/AllowRotationLambdaArn secretsmanager:ResourceTag/tag-key aws:ResourceTag/\${TagKey} secretsmanager:SecretPrimaryRegion	

Aktionen	Beschreibung	Zugriffsbene	Ressourcentypen (*erforderlich)	Bedingungschlüssel	Abhängige Aktionen
DeleteSecret	Gewährt die Berechtigung zum Löschen eines Geheimnisses	Schreiben	Secret*		

Aktionen	Beschreibung	Zugriffsebene	Ressourcentypen (*erforderlich)	Bedingungschlüssel	Abhängige Aktionen
				secretsmanager:SecretId secretsmanager:resource/AllowRotationLambdaAction secretsmanager:RecoveryWindowInDays secretsmanager:ForceDeleteWithoutRecovery secretsmanager:ResourceTag/tag-key aws:ResourceTag/\${TagKey} secretsmanager:Sec	

Aktionen	Beschreibung	Zugriffsberechtigungen	Ressourcentypen (*erforderlich)	Bedingungen	Abhängige Aktionen
				secretPrimaryRegion	
DescribeSecret	Gewährt die Berechtigung zum Abrufen der Metadaten über ein Geheimnis, aber nicht die verschlüsselten Daten	Lesen	Secret*	secretPrimaryRegion secretsmanager:SecretId secretsmanager:resource/AllowRotationLambdaAction secretsmanager:ResourceTag/tag-key aws:ResourceTag/\${TagKey} secretsmanager:secretPrimaryRegion	
GetRandomPassword	Gewährt die Berechtigung zum Generieren einer zufälligen Zeichenfolge zur Verwendung bei der Passwörterstellung	Lesen			

Aktionen	Beschreibung	Zugriffsbene	Ressourcentypen (*erforderlich)	Bedingungschlüssel	Abhängige Aktionen
GetResourcePolicy	Gewährt die Berechtigung zum Abrufen der Ressourcennrichtlinie, die an ein Geheimnis angefügt ist	Lesen	Secret*	secretsmanager:SecretId secretsmanager:resource/AllowRotationLambdaAction secretsmanager:ResourceTag/tag-key aws:ResourceTag/\${TagKey} secretsmanager:SecretPrimaryRegion	
GetSecretValue	Gewährt die Berechtigung zum Abrufen und Entschlüsseln der verschlüsselten Daten	Lesen	Secret*		

Aktionen	Beschreibung	Zugriffsbene	Ressourcentypen (*erforderlich)	Bedingungschlüssel	Abhängige Aktionen
				secretsmanager:SecretId secretsmanager:VersionId secretsmanager:VersionStage secretsmanager:resource/AllowRotationLambdaAction secretsmanager:ResourceTag/tag-key aws:ResourceTag/\${TagKey} secretsmanager:SecretPrimaryRegion	

Aktionen	Beschreibung	Zugriffsbene	Ressourcentypen (*erforderlich)	Bedingungschlüssel	Abhängige Aktionen
ListSecretVersionIds	Gewährt die Berechtigung zum Auflisten der verfügbaren Versionen eines Geheimnisses	Lesen	Secret*	secretsmanager:SecretId secretsmanager:resource/AllowRotationLambdaAction secretsmanager:ResourceTag/tag-key aws:ResourceTag/\${TagKey} secretsmanager:SecretPrimaryRegion	
ListSecrets	Gewährt die Berechtigung zum Auflisten der verfügbaren Geheimnisse	Auflisten			
PutResourcePolicy	Gewährt die Berechtigung zum Anfügen einer Ressourcenrichtlinie an ein Geheimnis	Berechtigungsverwaltung	Secret*		

Aktionen	Beschreibung	Zugriffsebene	Ressourcentypen (*erforderlich)	Bedingungschlüssel	Abhängige Aktionen
				secretsmanager:SecretId secretsmanager:resource/AllowRotationLambdaAction secretsmanager:ResourceTag/tag-key aws:ResourceTag/\${TagKey} secretsmanager:BlockPublicPolicy secretsmanager:SecretPrimaryRegion	

Aktionen	Beschreibung	Zugriffsbene	Ressourcentypen (*erforderlich)	Bedingungschlüssel	Abhängige Aktionen
PutSecretValue	Gewährt die Berechtigung zum Erstellen einer neuen Version des Geheimnisses mit neuen verschlüsselten Daten	Schreiben	Secret*	secretsmanager:SecretId secretsmanager:resource/AllowRotationLambdaAction secretsmanager:ResourceTag/tag-key aws:ResourceTag/\${TagKey} secretsmanager:SecretPrimaryRegion	
RemoveRegionsFromReplication	Erteilt die Erlaubnis, Regionen aus der Replikation zu entfernen	Schreiben	Secret*		

Aktionen	Beschreibung	Zugriffsebene	Ressourcentypen (*erforderlich)	Bedingungschlüssel	Abhängige Aktionen
				secretsmanager:SecretId secretsmanager:resource/AllowRotationLambdaAction secretsmanager:ResourceTag/tag-key aws:ResourceTag/\${TagKey} secretsmanager:SecretPrimaryRegion	

Aktionen	Beschreibung	Zugriffsbene	Ressourcentypen (*erforderlich)	Bedingungschlüssel	Abhängige Aktionen
Replicate SecretToRegions	<p>Erteilt die Erlaubnis, einen vorhandenen geheimen Schlüssel in einen Schlüssel für mehrere Regionen umzuwandeln und mit der Replikation des Geheimnisses in eine Liste neuer Regionen zu beginnen</p>	Schreiben	Secret*	secretsmanager:SecretId secretsmanager:resource/AllowRotationLambdaAction secretsmanager:ResourceTag/tag-key aws:ResourceTag/\${TagKey} secretsmanager:SecretPrimaryRegion secretsmanager:AddReplicaRegions secretsmanager:For	

Aktionen	Beschreibung	Zugriffsbene	Ressourcentypen (*erforderlich)	Bedingungschlüssel	Abhängige Aktionen
				ceOverwriteReplicaSecret	
RestoreSecret	Gewährt die Berechtigung zum Abbrechen des Löschsens eines Geheimnisses	Schreiben	Secret*	secretsmanager:SecretId secretsmanager:resource/AllowRotationLambdaAction secretsmanager:ResourceTag/tag-key aws:ResourceTag/\${TagKey} secretsmanager:SecretPrimaryRegion	
RotateSecret	Gewährt die Berechtigung zum Starten der Drehung eines Geheimnisses	Schreiben	Secret*		

Aktionen	Beschreibung	Zugriffsebene	Ressourcentypen (*erforderlich)	Bedingungschlüssel	Abhängige Aktionen
				secretsmanager:SecretId secretsmanager:RotationLambdaARN secretsmanager:resource/AllowRotationLambdaArn secretsmanager:ResourceTag/tag-key aws:ResourceTag/\${TagKey} secretsmanager:SecretPrimaryRegion secretsmanager:Mod	

Aktionen	Beschreibung	Zugriffsberechtigungen	Ressourcentypen (*erforderlich)	Bedingungen	Abhängige Aktionen
				ifyRotationRules secretsmanager:RotateImmediately	
StopReplicationToRegion	<p>Gewährt die Berechtigung zum Entfernen des Geheimnisses aus der Replikation und zum Konvertieren des Geheimnisses in ein regionales Geheimnis in der Replikationsregion</p>	Schreiben	Secret*	secretsmanager:SecretId secretsmanager:resource/AllowRotationLambdaAction secretsmanager:ResourceTag/tag-key aws:ResourceTag/\${TagKey} secretsmanager:SecretPrimaryRegion	

Aktionen	Beschreibung	Zugriffsbene	Ressourcentypen (*erforderlich)	Bedingungschlüssel	Abhängige Aktionen
TagResource	Gewährt die Berechtigung zum Hinzufügen von Tags zu einem Geheimnis	Tagging	Secret*	secretsmanager:SecretId aws:RequestTag/\${TagKey} aws:TagKeys secretsmanager:resource/AllowRotationLambdaArn secretsmanager:ResourceTag/tag-key aws:ResourceTag/\${TagKey} secretsmanager:SecretPrimaryRegion	

Aktionen	Beschreibung	Zugriffsebene	Ressourcentypen (*erforderlich)	Bedingungs-schlüssel	Abhängige Aktionen
UntagResource	Gewährt die Berechtigung zum Entfernen von Tags aus einem Geheimnis	Tagging	Secret*	secretsmanager:SecretId aws:TagKeys secretsmanager:resource/AllowRotationLambdaAction secretsmanager:ResourceTag/tag-key aws:ResourceTag/\${TagKey} secretsmanager:SecretPrimaryRegion	

Aktionen	Beschreibung	Zugriffsbene	Ressourcentypen (*erforderlich)	Bedingungschlüssel	Abhängige Aktionen
UpdateSecret	Gewährt die Berechtigung zum Aktualisieren eines Geheimnisses mit neuen Metadaten oder mit einer neuen Version der verschlüsselten Daten	Schreiben	Secret*	secretsmanager:SecretId secretsmanager:Description secretsmanager:KmsKeyId secretsmanager:resource/AllowRotationLambdaAction secretsmanager:ResourceTag/tag-key aws:ResourceTag/\${TagKey} secretsmanager:Sec	

Aktionen	Beschreibung	Zugriffsbene	Ressourcentypen (*erforderlich)	Bedingungschlüssel	Abhängige Aktionen
				retPrimaryRegion	
UpdateSecretVersionStage	Gewährt die Berechtigung, eine Stufe von einem Geheimnis zum anderen zu verschieben	Schreiben	Secret*	secretsmanager:SecretId secretsmanager:VersionStage secretsmanager:resource/AllowRotationLambdaAction secretsmanager:ResourceTag/tag-key aws:ResourceTag/\${TagKey} secretsmanager:SecretPrimaryRegion	

Aktionen	Beschreibung	Zugriffsbene	Ressourcentypen (*erforderlich)	Bedingungschlüssel	Abhängige Aktionen
ValidateResourcePolicy	Erteilt die Berechtigung, eine Ressourcenrichtlinie zu überprüfen, bevor eine Richtlinie angehängt wird	Berechtigungsverwaltung	Secret*	secretsmanager:SecretId secretsmanager:resource/AllowRotationLambdaAction secretsmanager:ResourceTag/tag-key aws:ResourceTag/\${TagKey} secretsmanager:SecretPrimaryRegion	

Secrets-Manager-Ressourcen

Ressourcentypen	ARN	Bedingungsschlüssel
Secret	<code>arn:\${Partition}:secretsmanager:\${Region}:\${Account}:secret:\${SecretId}</code>	aws:RequestTag/\${TagKey} aws:ResourceTag/\${TagKey} aws:TagKeys secretsmanager:ResourceTag/tag-key secretsmanager:resource/AllowRotationLambdaArn

Secrets Manager konstruiert den letzten Teil des Geheimnis-ARN, indem ein Bindestrich und sechs zufällige alphanumerische Zeichen am Ende des Geheimnis-Namens angehängt werden. Beim Löschen eines Secrets und dem Erstellen eines anderen Secrets mit demselben Namen wird mit dieser Formatierung sichergestellt, dass Personen mit Berechtigungen für das ursprüngliche Secret nicht automatisch Zugriff auf das neue Secret erhalten, da Secrets Manager sechs neue zufällige Zeichen generiert.

Sie finden den ARN für ein Secret in der Secrets-Manager-Konsole auf der Seite mit den Secret-Details oder durch Aufruf von [DescribeSecret](#).

Bedingungsschlüssel

Wenn Sie Zeichenfolgebedingungen aus der folgenden Tabelle in Ihre Berechtigungsrichtlinie aufnehmen, müssen Anrufer von Secrets Manager den übereinstimmenden Parameter übergeben, sonst wird ihnen der Zugriff verweigert. Um zu vermeiden, dass Anrufern wegen eines fehlenden Parameters der Zugriff verweigert wird, fügen Sie `IfExists` zum Ende des Namens des Bedingungs-Operators hinzu, z. B. `StringLikeIfExists`. Weitere Informationen finden Sie unter [IAM-JSON-Richtlinienelemente: Bedingungs-Operatoren](#).

Bedingungsschlüssel	Beschreibung	Typ
aws:RequestTag/\${TagKey}	Filtert den Zugriff nach einem Schlüssel, der in der Anforderung vorhanden ist, die der Benutzer an den Secrets-Manager-Service sendet	String
aws:ResourceTag/\${TagKey}	Filtert den Zugriff basierend auf den Tags, die der Ressource zugeordnet sind.	String
aws:TagKeys	Filtert den Zugriff nach der Liste aller Tag-Schlüsselnamen, die in der Anforderung vorhanden sind, die der Benutzer an den Secrets-Manager-Service sendet	ArrayOfString
secretsmanager:AddReplicaRegions	Filtert den Zugriff nach der Liste der Regionen, in denen das Geheimnis repliziert werden soll	ArrayOfString
secretsmanager:BlockPublicPolicy	Filtert den Zugriff danach, ob die Ressourcenrichtlinie den allgemeinen Zugriff blockiert AWS-Konto	Bool
secretsmanager:Description	Filtert den Zugriff nach dem Beschreibungstext in der Anforderung	String
secretsmanager:ForceDeleteWithoutRecovery	Filtert den Zugriff basierend darauf, ob das Geheimnis sofort und ohne Wiederherstellungsfenster gelöscht werden soll	Bool
secretsmanager:ForceOverwriteReplicaSecret	Filtert den Zugriff danach, ob ein Secret mit demselben Namen in der Zielregion überschrieben werden soll	Bool

Bedingungsschlüssel	Beschreibung	Typ
secretsmanager:KmsKeyId	Filtert den Zugriff nach dem ARN des KMS-Schlüssels in der Anforderung	String
secretsmanager:ModifyRotationRules	Filtert den Zugriff danach, ob die Rotationsregeln des Secrets geändert werden sollen	Bool
secretsmanager:Name	Filtert den Zugriff nach dem Anzeigenamen des Geheimnisses in der Anforderung	String
secretsmanager:RecoveryWindowInDays	Filtert den Zugriff anhand der Anzahl der Tage, die Secrets Manager wartet, bevor das Geheimnis gelöscht werden kann	Numerischer Wert
secretsmanager:ResourceTag/tag-key	Filtert den Zugriff anhand eines Tag-Schlüssel-Wert-Paares	String
secretsmanager:RotateImmediately	Filtert den Zugriff danach, ob das Secret sofort rotiert werden soll	Bool
secretsmanager:RotationLambdaARN	Filtert den Zugriff durch den ARN der Lambda-Funktion für die Drehung in der Anforderung	ARN
secretsmanager:SecretId	Filtert den Zugriff durch den SecretID-Wert in der Anforderung	ARN
secretsmanager:SecretPrimaryRegion	Filtert den Zugriff nach der primären Region, in der das Geheimnis erstellt wurde	String

Bedingungschlüssel	Beschreibung	Typ
secretsmanager:VersionId	Filtert den Zugriff nach der eindeutigen Kennung der Geheimnisversion in der Anforderung	String
secretsmanager:VersionStage	Filtert den Zugriff nach der Liste von Versionsstufen in der Anforderung	String
secretsmanager:resource/AllowRotationLambdaArn	Filtert den Zugriff nach dem ARN der Lambda-Funktion für die Drehung, die dem Geheimnis zugeordnet ist	ARN

Blockieren Sie breiten Zugriff auf Geheimnisse mit **BlockPublicPolicy**-Bedingung

In Identitätsrichtlinien, welche die Aktion `PutResourcePolicy` zulassen, empfehlen wir `BlockPublicPolicy: true` zu verwenden. Diese Bedingung bedeutet, dass Benutzer eine Ressourcenrichtlinie nur an ein Geheimnis anhängen können, wenn die Richtlinie keinen breiten Zugriff zulässt.

Secrets Manager verwendet das Automated Reasoning Zelkova zur Analyse von Ressourcenrichtlinien für den breiten Zugriff. Weitere Informationen zu Zelkova finden Sie im [Sicherheits-Blog AWS unter So hilft Ihnen automatisiertes Denken dabei, Sicherheit im großen Maßstab zu erreichen](#). AWS

Im folgenden Beispiel wird gezeigt, wie `BlockPublicPolicy` verwendet wird.

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

```
        "secretsmanager:BlockPublicPolicy": "true"
    }
}
}
```

Bedingungen für die IP-Adresse

Bei Angabe der [Bedingungsoperatoren für IP-Adressen](#) oder des `aws:SourceIp`-Bedingungsschlüssels in einer Richtlinienanweisung, die den Zugriff auf Secrets Manager zulässt oder verweigert, ist Vorsicht geboten. Wenn Sie beispielsweise eine Richtlinie anhängen, die AWS Aktionen auf Anfragen aus dem IP-Adressbereich Ihres Unternehmensnetzwerks auf einen geheimen Wert beschränkt, funktionieren Ihre Anfragen als IAM-Benutzer, der die Anfrage vom Unternehmensnetzwerk aus aufruft, erwartungsgemäß. Wenn Sie jedoch anderen Diensten ermöglichen, in Ihrem Namen auf das Geheimnis zuzugreifen, z. B. wenn Sie die Rotation mit einer Lambda-Funktion aktivieren, ruft diese Funktion die Secrets Manager Manager-Operationen von einem AWS-internen Adressraum aus auf. Anfragen, die von der Richtlinie mit dem IP-Adressfilter betroffen sind, schlagen fehl.

Zudem wird der Bedingungsschlüssel `aws:sourceIP` weniger wirksam, wenn die Anfrage von einem Amazon VPC-Endpoint kommt. Um Anfragen auf einen bestimmten VPC-Endpoint zu beschränken, verwenden Sie [the section called "VPC-Endpointbedingungen"](#).

VPC-Endpointbedingungen

Um den Zugriff auf Anfragen von einer bestimmten VPC oder einem VPC-Endpoint zuzulassen oder zu verweigern, verwenden Sie `aws:SourceVpc`, um den Zugriff auf Anfragen von der angegebenen VPC zu beschränken, oder `aws:SourceVpce`, um den Zugriff auf Anfragen von dem angegebenen VPC-Endpoint zu beschränken. Siehe [the section called "Beispiel: Berechtigungen und VPCs"](#).

- `aws:SourceVpc` beschränkt den Zugriff auf Anforderungen von der angegebenen VPC.
- `aws:SourceVpce` beschränkt den Zugriff auf Anforderungen vom angegebenen VPC-Endpoint.

Wenn Sie diese Bedingungsschlüssel in einer Ressourcen-Richtlinienanweisung verwenden, die Zugriff auf Secrets-Manager-Secrets zulässt oder verweigert, verweigern Sie möglicherweise versehentlich den Zugriff auf Services, die Secrets Manager verwenden, um für Sie auf Secrets zuzugreifen. Nur einige AWS Dienste können mit einem Endpoint in Ihrer VPC ausgeführt werden. Wenn Sie Anforderungen für ein Secret auf eine VPC oder einen VPC-Endpoint beschränken,

können Aufrufe an Secrets Manager von einem Service, der nicht für den Service konfiguriert ist, fehlschlagen.

Siehe [VPC-Endpunkt](#).

Geheimnisse erstellen und verwalten mit AWS Secrets Manager

Ein Geheimnis kann ein Passwort, ein Satz von Anmeldeinformationen wie z. B. ein Benutzername und ein Passwort, ein OAuth-Token oder andere Geheiminformationen sein, die Sie in verschlüsselter Form in Secrets Manager speichern.

Themen

- [Erstellen Sie ein AWS Secrets Manager Datenbankgeheimnis](#)
- [JSON-Struktur von AWS Secrets Manager Geheimnissen](#)
- [Erstelle ein AWS Secrets Manager Geheimnis](#)
- [Aktualisieren Sie den Wert für ein AWS Secrets Manager-Secret](#)
- [Generieren Sie ein Passwort mit Secrets Manager](#)
- [Ein Geheimnis auf eine frühere Version zurücksetzen](#)
- [Ändern Sie den Verschlüsselungsschlüssel für ein AWS Secrets Manager Geheimnis](#)
- [Ein AWS Secrets Manager Geheimnis ändern](#)
- [Finde Geheimnisse in AWS Secrets Manager](#)
- [Ein AWS Secrets Manager Geheimnis löschen](#)
- [Ein AWS Secrets Manager Geheimnis wiederherstellen](#)
- [AWS Secrets Manager-Secrets markieren](#)

Erstellen Sie ein AWS Secrets Manager Datenbankgeheimnis

Nachdem Sie einen Benutzer in Amazon RDS, Amazon Aurora, Amazon Redshift oder Amazon DocumentDB erstellt haben, können Sie seine Anmeldeinformationen in Secrets Manager speichern, indem Sie diese Schritte ausführen. Wenn Sie das AWS CLI oder eines der SDKs zum Speichern des Geheimnisses verwenden, müssen Sie das Geheimnis in der [richtigen JSON-Struktur](#) angeben. Wenn Sie die Konsole zum Speichern eines Datenbank-Secrets verwenden, erstellt Secrets Manager es automatisch in der richtigen JSON-Struktur.

Tip

Für Amazon RDS- und Amazon Redshift Redshift-Administrator-Benutzeranmeldedaten empfehlen wir die Verwendung von [Managed Secrets](#). Sie erstellen das verwaltete Geheimnis über den Verwaltungsservice und können dann die [verwaltete](#) Rotation verwenden.

Wenn Sie die Datenbank Anmeldeinformation für eine Quelldatenbank speichern, die in anderen Regionen repliziert wird, enthält das Secret Verbindungsinformationen für die Quelldatenbank. Wenn Sie dann das Secret replizieren, sind die Replikat Kopien des Quellsecret und enthalten dieselben Verbindungsinformationen. Sie können dem Secret zusätzliche Schlüssel/Wert-Paare hinzufügen, um regionale Verbindungsinformationen aufzunehmen.

Um ein Geheimnis zu erstellen, benötigen Sie die SecretsManagerReadWrite [AWS verwaltete Richtlinien](#) vom.

Secrets Manager generiert einen CloudTrail Protokolleintrag, wenn Sie ein Geheimnis erstellen. Weitere Informationen finden Sie unter [the section called “Loggen Sie sich mit AWS CloudTrail”](#).

Ein Secret erstellen (Konsole)

1. Öffnen Sie die Secrets-Manager-Konsole unter <https://console.aws.amazon.com/secretsmanager/>.
2. Wählen Sie Store a new secret (Ein neues Secret speichern).
3. Führen Sie auf der Seite Choose secret type (Secret-Typ auswählen) die folgenden Schritte aus:
 - a. Wählen Sie als Secret-Typ den Typ der zu speichernden Datenbank-Anmeldeinformation aus:
 - Amazon RDS-Datenbank (beinhaltet Aurora)
 - Amazon-DocumentDB-Datenbank
 - Amazon Redshift Redshift-Datawarehouse
 - b. Geben Sie als Anmeldeinformation Ihre Anmeldeinformation für die Datenbank ein.
 - c. Wählen Sie unter Verschlüsselungsschlüssel den aus AWS KMS key , den Secrets Manager zum Verschlüsseln des geheimen Werts verwendet. Weitere Informationen finden Sie unter [Ver- und Entschlüsselung von Secrets](#).

- Wählen Sie in den meisten Fällen `aws/secretsmanager` aus, um den Von AWS verwalteter Schlüssel für Secrets Manager zu benutzen. Für die Verwendung dieses Schlüssels fallen keine Kosten an.
- Wenn Sie von einem anderen auf den geheimen Schlüssel zugreifen müssen oder wenn Sie Ihren eigenen KMS-Schlüssel verwenden möchten AWS-Konto, sodass Sie ihn rotieren oder eine Schlüsselrichtlinie darauf anwenden können, wählen Sie einen vom Kunden verwalteten Schlüssel aus der Liste aus oder klicken Sie auf Neuen Schlüssel hinzufügen, um einen Schlüssel zu erstellen. Informationen zu den Kosten der Verwendung eines vom Kunden verwalteten Schlüssels finden Sie unter [Preisgestaltung](#).

Sie müssen [the section called “Berechtigungen für den KMS-Schlüssel”](#) haben. Informationen zum kontoübergreifenden Zugriff finden Sie unter [the section called “Kontoübergreifender Zugriff”](#).

- d. Als Datenbank wählen Sie Ihre Datenbank aus.
 - e. Wählen Sie Weiter aus.
4. Führen Sie auf der Seite Configure secret (Secret konfigurieren) die folgenden Schritte aus:
 - a. Geben Sie einen beschreibenden Secret-Namen und eine Beschreibung ein. Secret-Namen müssen 1–512 Unicode-Zeichen enthalten.
 - b. (Optional) Im Abschnitt Tags können Sie Tags zu Ihrem Secret hinzufügen. Informationen zu Tagging-Strategien finden Sie unter [the section called “-Secrets markieren”](#). Speichern Sie keine sensiblen Daten in Tags, da sie nicht verschlüsselt sind.
 - c. (Optional) Um eine Ressourcenrichtlinie zu Ihrem Secret hinzuzufügen, wählen Sie unter Resource permissions (Ressourcenberechtigungen) die Option Edit permissions (Berechtigungen bearbeiten) aus. Weitere Informationen finden Sie unter [the section called “Zuordnen einer Berechtigungsrichtlinie zu einem Secret”](#).
 - d. (Optional) Wählen Sie unter Geheimnis replizieren die Option Geheimnis replizieren aus, um Ihr Geheimnis auf ein anderes AWS-Region zu replizieren. Sie können Ihr Secret jetzt replizieren oder zurückkommen und es später replizieren. Weitere Informationen finden Sie unter [Replizieren Sie Geheimnisse regionsübergreifend](#).
 - e. Wählen Sie Weiter.
 5. (Optional) Auf der Seite Rotation konfigurieren können Sie die automatische Rotation aktivieren. Sie können die Rotation auch vorerst ausschalten und später einschalten. Weitere Informationen finden Sie unter [Rotieren von -Geheimnissen](#). Wählen Sie Weiter.

6. Prüfen Sie auf der Seite Review (Prüfen) die Secret-Details und wählen Sie Store (Speichern).

Secrets Manager kehrt zur Liste der Secrets zurück. Wenn Ihr Secret nicht angezeigt wird, wählen Sie den Aktualisieren-Button aus.

AWS CLI

Wenn Sie Befehle in eine Befehls-Shell eingeben, besteht die Gefahr, dass auf den Befehlsverlauf zugegriffen wird oder Serviceprogramme Zugriff auf Ihre Befehlsparameter haben. Siehe [the section called “Reduzieren von Risiken durch die Verwendung der AWS CLI zur Speicherung Ihrer AWS Secrets Manager-Secrets”](#).

Example Ein Secret aus Anmeldeinformationen in einer JSON-Datei erstellen

Das folgende [create-secret](#)-Beispiel erstellt ein Secret anhand von Anmeldeinformationen in einer Datei. Weitere Informationen finden Sie im Benutzerhandbuch unter [Laden von AWS CLI Parametern aus einer Datei](#). AWS CLI

Damit Secrets Manager das Secret rotieren kann, müssen Sie sicherstellen, dass JSON mit [JSON-Struktur eines Secrets](#) übereinstimmt.

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

Inhalt von mycreds.json:

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

AWS SDK

Verwenden Sie die [CreateSecret](#)Aktion, um mithilfe eines der AWS SDKs ein Geheimnis zu erstellen. Weitere Informationen finden Sie unter [the section called “AWS SDKs”](#).

JSON-Struktur von AWS Secrets Manager Geheimnissen

Sie können jeden Text oder jede Binärdatei in Secrets Manager Secrets speichern. Wenn Sie die automatische Rotation für ein Secret von Secrets Manager aktivieren möchten, muss es in der richtigen JSON-Struktur vorliegen. Während der Rotation verwendet Secrets Manager die Informationen im Secret, um eine Verbindung mit der Quelle der Anmeldeinformationen herzustellen und die Anmeldeinformationen dort zu aktualisieren. Bei den JSON-Schlüsselnamen wird zwischen Groß- und Kleinschreibung unterschieden.

Bitte beachten Sie: Wenn Sie die Konsole zum Speichern eines Datenbank-Secrets verwenden, erstellt Secrets Manager es automatisch in der richtigen JSON-Struktur.

Sie können einem Secret weitere Schlüssel/Wert-Paare hinzufügen, damit es beispielsweise in einem Datenbank-Secret Verbindungsinformationen für Replikat-Datenbanken in anderen Regionen enthält.

Themen

- [Secret-Struktur von Amazon RDS Db2](#)
- [Amazon RDS MariaDB – Secret-Struktur](#)
- [Secret-Struktur von Amazon RDS und Amazon Aurora MySQL](#)
- [Amazon RDS Oracle – Secret-Struktur](#)
- [Secret-Struktur von Amazon RDS und Amazon Aurora PostgreSQL](#)
- [Amazon RDS Microsoft SQLServer – Secret-Struktur](#)
- [Amazon DocumentDB – Secret-Struktur](#)
- [Amazon Redshift – Secret-Struktur](#)
- [Geheime Struktur von Amazon Redshift Serverless](#)
- [ElastiCache Geheime Struktur von Amazon](#)
- [Geheime Active Directory-Strukturen](#)

Secret-Struktur von Amazon RDS Db2

Da Benutzer bei Amazon-RDS-Db2-Instances ihre eigenen Passwörter nicht ändern können, müssen Sie Administratoranmeldedaten in einem separaten Secrets angeben.

```
{
  "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 – Secret-Struktur

```

{
  "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>
}

```

Um das zu verwenden [the section called “Wechselnde Benutzer”](#), geben Sie das masterarn für das Geheimnis ein, das Admin- oder Superuser-Anmeldeinformationen enthält.

```

{
  "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>"
}

```

Secret-Struktur von Amazon RDS und Amazon Aurora MySQL

```

{
  "engine": "mysql",
  "host": "<instance host name/resolvable DNS name>",
  "username": "<username>",
  "password": "<password>",
  "dbname": "<database name. If not specified, defaults to None>",
  "port": <TCP port number. If not specified, defaults to 3306>
}

```

Um das zu verwendenthe section called [“Wechselnde Benutzer”](#), geben Sie das masterarn für das Geheimnis ein, das Admin- oder Superuser-Anmeldeinformationen enthält.

```
{
  "engine": "mysql",
  "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 3306></i>,
  "masterarn": "<i><the ARN of the elevated secret></i>"
}
```

Amazon RDS Oracle – Secret-Struktur

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

Um das zu verwendenthe section called [“Wechselnde Benutzer”](#), geben Sie das masterarn für das Geheimnis ein, das Admin- oder Superuser-Anmeldeinformationen enthält.

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

Secret-Struktur von Amazon RDS und Amazon Aurora PostgreSQL

```
{
  "engine": "postgres",
```

```
"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 'postgres'></i>",
"port": <i><TCP port number. If not specified, defaults to 5432></i>
}
```

Um das zu verwendende [the section called “Wechselnde Benutzer”](#), geben Sie das masterarn für das Geheimnis ein, das Admin- oder Superuser-Anmeldeinformationen enthält.

```
{
  "engine": "postgres",
  "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 'postgres'></i>",
  "port": <i><TCP port number. If not specified, defaults to 5432></i>,
  "masterarn": "<i><the ARN of the elevated secret></i>"
}
```

Amazon RDS Microsoft SQLServer – Secret-Struktur

```
{
  "engine": "sqlserver",
  "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 'master'></i>",
  "port": <i><TCP port number. If not specified, defaults to 1433></i>
}
```

Um das zu verwendende [the section called “Wechselnde Benutzer”](#), geben Sie das masterarn für das Geheimnis ein, das Admin- oder Superuser-Anmeldeinformationen enthält.

```
{
  "engine": "sqlserver",
  "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 'master'></i>",
  "port": <i><TCP port number. If not specified, defaults to 1433></i>,
  "masterarn": "<i><the ARN of the elevated secret></i>"
}
```



```
}
```

Amazon DocumentDB – Secret-Struktur

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

Um das zu verwendende [the section called “Wechselnde Benutzer”](#), geben Sie das `masterarn` für das Geheimnis ein, das Admin- oder Superuser-Anmeldeinformationen enthält.

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

Amazon Redshift – Secret-Struktur

```
{
  "engine": "redshift",
  "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 5439>
}
```

Um das zu verwendende [the section called “Wechselnde Benutzer”](#), geben Sie das `masterarn` für das Geheimnis ein, das Admin- oder Superuser-Anmeldeinformationen enthält.

```
{
  "engine": "redshift",
  "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 5439>,
  "masterarn": "<the ARN of the elevated secret>"
}
```

Geheime Struktur von Amazon Redshift Serverless

```
{
  "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>
}
```

Um die zu verwendende [the section called “Wechselnde Benutzer”](#), geben Sie das masterarn für das Geheimnis ein, das Admin- oder Superuser-Anmeldeinformationen enthält.

```
{
  "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>"
}
```

ElastiCache Geheime Struktur von Amazon

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

```
"user_arn": "ARN of the Amazon EC2 user"
}
```

Weitere Informationen finden Sie unter [Automatisches Rotieren von Passwörtern für Benutzer](#) im ElastiCache Amazon-Benutzerhandbuch.

Geheime Active Directory-Strukturen

AWS Directory Service verwendet Geheimnisse, um Active Directory-Anmeldeinformationen zu speichern. Weitere Informationen finden Sie unter [Nahtloses Hinzufügen einer Amazon EC2 EC2-Linux-Instance zu Ihrem Managed AD Active Directory](#) im AWS Directory Service Administratorhandbuch. Für einen nahtlosen Domänenbeitritt sind die Schlüsselnamen in den folgenden Beispielen erforderlich. Wenn Sie Seamless Domain Join nicht verwenden, können Sie die Namen der Schlüssel im Secret mithilfe von Umgebungsvariablen ändern, wie im Vorlagencode für die Rotationsfunktion beschrieben.

Um Active Directory-Geheimnisse zu rotieren, können Sie die [Active Directory-Rotationsvorlagen](#) verwenden.

Struktur der geheimen Active Directory-Anmeldeinformationen

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

Wenn Sie den geheimen Schlüssel rotieren möchten, geben Sie die Domänenverzeichnis-ID an.

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

Wenn das Geheimnis in Verbindung mit einem Geheimnis verwendet wird, das eine Schlüsseltabelle enthält, geben Sie die ARNs für geheime Keytab-Schlüssel an.

```
{
  "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"
}

```

Geheime Struktur der Active Directory-Schlüsseltabelle

Informationen zur Verwendung von Keytab-Dateien zur Authentifizierung bei Active Directory-Konten auf Amazon EC2 finden Sie unter [Bereitstellen und Konfigurieren der Active Directory-Authentifizierung mit SQL Server 2017 auf Amazon Linux 2](#).

```

{
  "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
}

```

Erstelle ein AWS Secrets Manager Geheimnis

Gehen Sie folgendermaßen vor, um API-Schlüssel, Zugriffs-Token und Anmeldeinformationen zu speichern, die nicht für Datenbanken vorgesehen sind, und andere Secrets in Secrets Manager. Wenn Sie für ein ElastiCache Amazon-Geheimnis die Rotation aktivieren möchten, müssen Sie das Geheimnis in der [erwarteten JSON-Struktur](#) speichern.

Um ein Geheimnis zu erstellen, benötigen Sie die von der gewährten Berechtigungen SecretsManagerReadWrite [AWS verwaltete Richtlinien](#).

Secrets Manager generiert einen CloudTrail Protokolleintrag, wenn Sie ein Geheimnis erstellen. Weitere Informationen finden Sie unter [the section called “Loggen Sie sich mit AWS CloudTrail”](#).

Ein Secret erstellen (Konsole)

1. Öffnen Sie die Secrets-Manager-Konsole unter <https://console.aws.amazon.com/secretsmanager/>.
2. Wählen Sie Store a new secret (Ein neues Secret speichern).
3. Führen Sie auf der Seite Choose secret type (Secret-Typ auswählen) die folgenden Schritte aus:
 - a. Als Secret-Typ wählen Sie Anderer Secret-Typ aus.
 - b. Geben Sie unter Schlüssel/Wertpaare entweder Ihr Secret in JSON-Schlüssel/Wertpaaren ein oder wählen Sie die Registerkarte Nur-Text und geben Sie das Secret in einem beliebigen Format ein. Sie können bis zu 65536 Bytes im Secret speichern. Hier einige Beispiele:

Schlüssel-Wert-Paare des API-Schlüssels:

ClientID : *my_client_id*

ClientSecret : *bPxRfiWJALRXUTNFEMI/K7MDENG/CY* BEISPIELS
CHLÜSSEL

Schlüssel-Wert-Paare von Anmeldeinformationen:

Username: *Saanvis*

Password : *EXAMPLE-PASSWORD*

Klartext für OAuth-Token:

AKIAI44QH8DHBEXAMPLE

Klartext für digitales Zertifikat:

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

Schlüssel-Wert-Paare des API-Schlüssels:

ClientID : *my_client_id*

ClientSecret : *bPxRfiWJALRXUTNFEMI/K7MDENG/CY* BEISPIELS
CHLÜSSEL

Klartext für privaten Schlüssel:

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

- c. Wählen Sie unter Verschlüsselungsschlüssel den aus AWS KMS key , den Secrets Manager zum Verschlüsseln des geheimen Werts verwendet. Weitere Informationen finden Sie unter [Ver- und Entschlüsselung von Secrets](#).
 - Wählen Sie in den meisten Fällen aws/secretsmanager aus, um den Von AWS verwalteter Schlüssel für Secrets Manager zu benutzen. Für die Verwendung dieses Schlüssels fallen keine Kosten an.
 - Wenn Sie von einem anderen auf den geheimen Schlüssel zugreifen müssen oder wenn Sie Ihren eigenen KMS-Schlüssel verwenden möchten AWS-Konto, sodass Sie ihn rotieren oder eine Schlüsselrichtlinie darauf anwenden können, wählen Sie einen vom Kunden verwalteten Schlüssel aus der Liste aus oder klicken Sie auf Neuen Schlüssel hinzufügen, um einen Schlüssel zu erstellen. Informationen zu den Kosten der Verwendung eines vom Kunden verwalteten Schlüssels finden Sie unter [Preisgestaltung](#).

Sie müssen [the section called "Berechtigungen für den KMS-Schlüssel"](#) haben. Informationen zum kontoübergreifenden Zugriff finden Sie unter [the section called "Kontoübergreifender Zugriff"](#).
 - d. Wählen Sie Weiter aus.
4. Führen Sie auf der Seite Configure secret (Secret konfigurieren) die folgenden Schritte aus:
 - a. Geben Sie einen beschreibenden Secret-Namen und eine Beschreibung ein. Secret-Namen müssen 1–512 Unicode-Zeichen enthalten.

- b. (Optional) Im Abschnitt Tags können Sie Tags zu Ihrem Secret hinzufügen. Informationen zu Tagging-Strategien finden Sie unter [the section called “-Secrets markieren”](#). Speichern Sie keine sensiblen Daten in Tags, da sie nicht verschlüsselt sind.
 - c. (Optional) Um eine Ressourcenrichtlinie zu Ihrem Secret hinzuzufügen, wählen Sie unter Resource permissions (Ressourcenberechtigungen) die Option Edit permissions (Berechtigungen bearbeiten) aus. Weitere Informationen finden Sie unter [the section called “Zuordnen einer Berechtigungsrichtlinie zu einem Secret”](#).
 - d. (Optional) Wählen Sie unter Geheimnis replizieren die Option Geheimnis replizieren aus, um Ihr Geheimnis auf ein anderes AWS-Region zu replizieren. Sie können Ihr Secret jetzt replizieren oder zurückkommen und es später replizieren. Weitere Informationen finden Sie unter [Replizieren Sie Geheimnisse regionsübergreifend](#).
 - e. Wählen Sie Weiter.
5. (Optional) Auf der Seite Rotation konfigurieren können Sie die automatische Rotation aktivieren. Sie können die Rotation auch vorerst ausschalten und später einschalten. Weitere Informationen finden Sie unter [Rotieren von -Geheimnissen](#). Wählen Sie Weiter.
 6. Prüfen Sie auf der Seite Review (Prüfen) die Secret-Details und wählen Sie Store (Speichern).

Secrets Manager kehrt zur Liste der Secrets zurück. Wenn Ihr Secret nicht angezeigt wird, wählen Sie den Aktualisieren-Button aus.

AWS CLI

Wenn Sie Befehle in eine Befehls-Shell eingeben, besteht die Gefahr, dass auf den Befehlsverlauf zugegriffen wird oder Serviceprogramme Zugriff auf Ihre Befehlsparameter haben. Siehe [the section called “Reduzieren von Risiken durch die Verwendung der AWS CLI zur Speicherung Ihrer AWS Secrets Manager-Secrets”](#).

Example Ein Secret erstellen

Das folgende [create-secret](#)-Beispiel erstellt ein Secret mit zwei Schlüssel-/Wert-Paaren.

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

Example Ein Secret aus Anmeldeinformationen in einer JSON-Datei erstellen

Das folgende [create-secret](#)-Beispiel erstellt ein Secret anhand von Anmeldeinformationen in einer Datei. Weitere Informationen finden Sie im Benutzerhandbuch unter [Laden von AWS CLI Parametern aus einer Datei](#). AWS CLI

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

Inhalt von mycreds.json:

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

AWS SDK

Verwenden Sie die [CreateSecret](#)Aktion, um mithilfe eines der AWS SDKs ein Geheimnis zu erstellen. Weitere Informationen finden Sie unter [the section called "AWS SDKs"](#).

Aktualisieren Sie den Wert für ein AWS Secrets Manager-Secret

Um den Wert Ihres Secrets zu aktualisieren, können Sie die Konsole, die CLI oder ein SDK verwenden. Wenn Sie den Secret-Wert aktualisieren, erstellt Secrets Manager eine neue Version des Secrets mit dem Staging-Label AWSCURRENT. Sie können immer noch auf die alte Version zugreifen, die das AWSPREVIOUS-Label trägt. Sie können auch Ihre eigenen Labels hinzufügen. Weitere Informationen finden Sie unter [Secrets-Manager-Versionsverwaltung](#).

Informationen zur Aktualisierung des Geheimwertes (Konsole)

1. Öffnen Sie die Secrets-Manager-Konsole unter <https://console.aws.amazon.com/secretsmanager/>.
2. Wählen Sie aus der Liste der Secrets Ihr Secret aus.
3. Wählen Sie auf der Seite zu den Secret-Details auf der Registerkarte Übersicht im Abschnitt Secret-Wert die Option Secret-Wert abrufen und dann Bearbeiten aus.

AWS CLI

Informationen zur Aktualisierung des Geheimwertes (AWS CLI)

- Wenn Sie Befehle in eine Befehls-Shell eingeben, besteht die Gefahr, dass auf den Befehlsverlauf zugegriffen wird oder Serviceprogramme Zugriff auf Ihre Befehlsparameter haben. Siehe [the section called “Reduzieren von Risiken durch die Verwendung der AWS CLI zur Speicherung Ihrer AWS Secrets Manager-Secrets”](#).

Der folgende [put-secret-value](#) erstellt eine neue Version eines Secrets mit zwei Schlüssel-/Wert-Paaren.

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

Mit dem folgenden [put-secret-value](#) wird eine neue Version mit einem benutzerdefinierten Staging-Label erstellt. Die neue Version wird die Labels MyLabel und AWSCURRENT haben.

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

AWS-SDK

Wir empfehlen, `PutSecretValue` oder `UpdateSecret` nicht dauerhaft mehr als einmal alle 10 Minuten aufzurufen. Wenn Sie `PutSecretValue` oder `UpdateSecret` aufrufen, um den Secret-Wert zu aktualisieren, erstellt Secrets Manager eine neue Version des Secrets. Secrets Manager entfernt Versionen ohne Label, wenn es mehr als 100 davon gibt. Versionen, die jünger als 24 Stunden sind, werden nicht entfernt. Wenn Sie den Secret-Wert mehr als einmal alle 10 Minuten aktualisieren, erstellen Sie mehr Versionen als Secrets Manager entfernt, und Sie erreichen das Kontingent für Secret-Versionen.

Gehen Sie wie folgt vor, um ein Secret-Wert zu aktualisieren: [UpdateSecret](#) oder [PutSecretValue](#). Weitere Informationen finden Sie unter [the section called “AWS SDKs”](#).

Generieren Sie ein Passwort mit Secrets Manager

Ein übliches Muster für die Verwendung von Secrets Manager besteht darin, ein Passwort in Secrets Manager zu generieren und dieses Passwort dann in Ihrer Datenbank oder Ihrem Dienst zu verwenden. Sie können dies mit den folgenden Methoden tun:

- AWS CloudFormation — Siehst du [AWS CloudFormation](#).
- AWS CLI — Siehst du [get-random-password](#).
- AWS SDKs — siehe [GetRandomPassword](#).

Ein Geheimnis auf eine frühere Version zurücksetzen

Sie können ein Geheimnis auf eine frühere Version zurücksetzen, indem Sie die Labels, die mit geheimen Versionen verknüpft sind, mithilfe von verschieben. AWS CLI Informationen darüber, wie Secrets Manager Versionen von Geheimnissen speichert, finden Sie unter [the section called "Geheime Versionen"](#).

Im folgenden [update-secret-version-stage](#) Beispiel wird das AWSCURRENT Staging-Label auf die vorherige Version eines Secrets verschoben, wodurch das Secret auf die vorherige Version zurückgesetzt wird. Um die ID für die vorherige Version zu finden, verwenden [list-secret-version-ids](#) oder sehen Sie sich die Versionen in der Secrets Manager Manager-Konsole an.

In diesem Beispiel ist die Version mit der Bezeichnung a1B2C3D4-5678-90AB-CDEF-Example11111 und die Version mit der AWSCURRENT Bezeichnung A1B2C3D4-5678-90AB-CDEF-Example22222. AWSPREVIOUS In diesem Beispiel verschieben Sie AWSCURRENT das Label von Version 11111 nach 22222. Da das AWSCURRENT Label aus einer Version entfernt wurde, wird das AWSPREVIOUS Label `update-secret-version-stage` automatisch in diese Version (11111) verschoben. Dies hat zur Folge, dass die AWSPREVIOUS Versionen AWSCURRENT und die Versionen vertauscht werden.

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

Ändern Sie den Verschlüsselungsschlüssel für ein AWS Secrets Manager Geheimnis

Secrets Manager verwendet eine [Umschlagverschlüsselung](#) mit AWS KMS Schlüsseln und Datenschlüsseln, um jeden geheimen Wert zu schützen. Für jedes Secret können Sie wählen, welcher KMS-Schlüssel verwendet werden soll. Sie können den Von AWS verwalteter Schlüssel `aws/secretsmanager` oder einen vom Kunden verwalteten Schlüssel verwenden. In den meisten Fällen empfehlen wir die Nutzung von `aws/secretsmanager` und es fallen keine Kosten für die Nutzung an. Wenn Sie von einem anderen AWS-Konto auf den geheimen Schlüssel zugreifen müssen oder wenn Sie Ihren eigenen KMS-Schlüssel verwenden möchten, sodass Sie ihn rotieren oder eine Schlüsselrichtlinie darauf anwenden können, verwenden Sie einen Kundenverwalteter Schlüssel. Sie müssen [the section called “Berechtigungen für den KMS-Schlüssel”](#) haben. Informationen zu den Kosten der Verwendung eines vom Kunden verwalteten Schlüssels finden Sie unter [Preisgestaltung](#).

Sie können den Verschlüsselungscode für Ihr Secret ändern. Wenn Sie beispielsweise [von einem anderen Konto aus auf das Geheimnis zugreifen](#) möchten und das Geheimnis derzeit mit dem AWS verwalteten Schlüssel verschlüsselt ist `aws/secretsmanager`, können Sie zu einem wechseln Kundenverwalteter Schlüssel.

Tip

Wenn Sie Ihren wechseln möchten Kundenverwalteter Schlüssel, empfehlen wir die AWS KMS automatische Schlüsselrotation. Weitere Informationen finden Sie unter [AWS KMS Tasten drehen](#).

Wenn Sie den Verschlüsselungsschlüssel ändern, verschlüsselt Secrets Manager `AWSCURRENTAWSPENDING`, und `AWSPREVIOUS` Versionen erneut mit dem neuen Schlüssel. Um zu verhindern, dass Sie aus dem Geheimnis ausgesperrt werden, speichert Secrets Manager alle vorhandenen Versionen mit dem vorherigen Schlüssel verschlüsselt. Das bedeutet `AWSCURRENTAWSPENDING`, dass Sie `AWSPREVIOUS` Versionen mit dem vorherigen Schlüssel oder dem neuen Schlüssel entschlüsseln können.

Damit es nur mit dem neuen Verschlüsselungsschlüssel entschlüsselt werden `AWSCURRENT` kann, erstellen Sie eine neue Version des Geheimnisses mit dem neuen Schlüssel. Um dann die `AWSCURRENT` geheime Version entschlüsseln zu können, benötigen Sie die Erlaubnis für den neuen Schlüssel.

Wenn Sie den vorherigen Verschlüsselungsschlüssel deaktivieren, können Sie keine Secret-Versionen außer `AWSCURRENT`, `AWSPENDING` und `AWSPREVIOUS` entschlüsseln. Wenn Sie über andere als Secret gekennzeichnete Versionen verfügen, auf die Sie weiterhin Zugriff haben möchten, müssen Sie diese Versionen mit dem neuen Verschlüsselungsschlüssel neu erstellen. Verwenden Sie dazu [the section called “AWS CLI”](#).

Informationen zum Ändern des Verschlüsselungsschlüssels für ein Secret (Konsole)

1. Öffnen Sie die Secrets-Manager-Konsole unter <https://console.aws.amazon.com/secretsmanager/>.
2. Wählen Sie aus der Liste der Secrets Ihr Secret aus.
3. Wählen Sie auf der Seite Secret details (Secret-Details) im Abschnitt Secrets details (Secret-Details) die Option Actions (Aktionen) und danach Edit encryption key (Verschlüsselungsschlüssel bearbeiten).

AWS CLI

Wenn Sie den Verschlüsselungsschlüssel für ein Secret ändern und dann den vorherigen Verschlüsselungsschlüssel deaktivieren, können Sie keine Secret-Versionen außer `AWSCURRENT`, `AWSPENDING` und `AWSPREVIOUS` entschlüsseln. Wenn Sie über andere als Secret gekennzeichnete Versionen verfügen, auf die Sie weiterhin Zugriff haben möchten, müssen Sie diese Versionen mit dem neuen Verschlüsselungsschlüssel neu erstellen. Verwenden Sie dazu [the section called “AWS CLI”](#).

Informationen zum Ändern des Verschlüsselungsschlüssels für ein Secret (AWS CLI)

1. Im folgenden [update-secret](#)-Beispiel wird der KMS-Schlüssel aktualisiert, der zum Verschlüsseln des Secret-Werts verwendet wird. Der KMS-Schlüssel muss sich in derselben Region wie das Secret befinden.

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

2. (Optional) Wenn Sie geheime Versionen mit benutzerdefinierten Bezeichnungen haben, müssen Sie diese Versionen neu erstellen, um sie mit dem neuen Schlüssel erneut zu verschlüsseln.

Wenn Sie Befehle in eine Befehls-Shell eingeben, besteht die Gefahr, dass auf den Befehlsverlauf zugegriffen wird oder Serviceprogramme Zugriff auf Ihre Befehlsparameter haben. Siehe [the section called “Reduzieren von Risiken durch die Verwendung der AWS CLI zur Speicherung Ihrer AWS Secrets Manager-Secrets”](#).

- a. Ermittelt den Wert der Secret-Version.

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

Notieren Sie sich den Secret-Wert.

- b. Erstellen Sie eine neue Version mit diesem Wert.

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

Ein AWS Secrets Manager Geheimnis ändern

Sie können die Metadaten eines Services nach dessen Erstellung ändern, je nachdem, wer das Geheimnis erstellt hat. Bei Geheimnissen, die von anderen Services erstellt wurden, müssen Sie möglicherweise den anderen Service nutzen, um sie zu aktualisieren oder zu rotieren.

Um festzustellen, wer ein Geheimnis verwaltet, können Sie den Namen des Geheimnisses überprüfen. Bei Geheimnissen, die von anderen Services verwaltet werden, wird die ID des jeweiligen Services vorangestellt. Oder rufen Sie im AWS CLI [describe-secret](#) auf und überprüfen Sie dann das Feld. `OwningService` Weitere Informationen finden Sie unter [Verwaltete Geheimnisse](#).

Für von Ihnen verwaltete Geheimnisse können Sie die Beschreibung, die ressourcenbasierte Richtlinie, den Verschlüsselungsschlüssel und die Tags ändern. Sie können auch den verschlüsselten Secret-Wert ändern. Wir empfehlen jedoch, Secret-Werte, die Anmeldeinformationen enthalten, durch Rotation zu aktualisieren. Durch die Rotation werden sowohl das Secret im Secrets Manager als auch die Anmeldeinformationen in der Datenbank oder im Service aktualisiert. Auf diese Weise werden die Secrets automatisch synchronisiert, damit sie immer einen Satz an

Anmeldeinformationen abrufen, wenn Clients einen Secret-Wert anfordern. Weitere Informationen finden Sie unter [Rotieren von -Geheimnissen](#).

Secrets Manager generiert einen CloudTrail Protokolleintrag, wenn Sie ein Geheimnis ändern. Weitere Informationen finden Sie unter [the section called “Loggen Sie sich mit AWS CloudTrail”](#).

So aktualisieren Sie ein von Ihnen verwaltetes Geheimnis (Konsole)

1. Öffnen Sie die Secrets-Manager-Konsole unter <https://console.aws.amazon.com/secretsmanager/>.
2. Wählen Sie aus der Liste der Secrets Ihr Secret aus.
3. Führen Sie auf der Seite zu den Secret Daten die folgenden Schritte aus:

Beachten Sie, dass Sie den Namen oder die ARN eines Secrets nicht ändern können.

- Zum Aktualisieren der Beschreibung wählen Sie im Bereich Secret-Details die Option Aktionen und danach Beschreibung bearbeiten.
- Informationen zum Aktualisieren des Verschlüsselungsschlüssels finden Sie unter [the section called “Ändern des Verschlüsselungsschlüssels für ein Secret”](#).
- Zum Aktualisieren von Tags wählen Sie in der Registerkarte Tags die Option Tags bearbeiten aus. Siehe [the section called “-Secrets markieren”](#).
- Um den Geheimwert zu aktualisieren, siehe [the section called “Aktualisieren eines Geheimniswertes”](#).
- Um die Berechtigungen für Ihr Secret zu aktualisieren, wählen Sie auf der Registerkarte Übersicht die Option Berechtigungen bearbeiten aus. Siehe [the section called “Zuordnen einer Berechtigungsrichtlinie zu einem Secret”](#).
- Um die Rotation für Ihr Secret zu aktualisieren, wählen Sie auf der Registerkarte Rotation die Option Rotation bearbeiten aus. Siehe [Rotieren von -Geheimnissen](#).
- Um Ihr Geheimnis in andere Regionen zu replizieren, siehe [Replizieren Sie Geheimnisse regionsübergreifend](#).
- Wenn Ihr Geheimnisse Replikate enthält, können Sie den Verschlüsselungsschlüssel für ein Replikat ändern. Wählen Sie auf der Registerkarte Replikation das Optionsfeld für das Replikat aus und wählen Sie dann im Menü Aktionen die Option Verschlüsselungsschlüssel bearbeiten aus. Siehe [the section called “Ver- und Entschlüsselung von Secrets”](#).
- Um ein Secret so zu ändern, dass es von einem anderen Service verwaltet wird, müssen Sie das Secret in diesem Service neuerstellen. Siehe [Verwaltete Geheimnisse](#).

AWS CLI

Example Secret-Beschreibung aktualisieren

Im folgenden [update-secret](#)-Beispiel wird die Beschreibung eines Secrets aktualisiert.

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

AWS SDK

Wir empfehlen, `PutSecretValue` oder `UpdateSecret` nicht dauerhaft mehr als einmal alle 10 Minuten aufzurufen. Wenn Sie `PutSecretValue` oder `UpdateSecret` aufrufen, um den Secret-Wert zu aktualisieren, erstellt Secrets Manager eine neue Version des Secrets. Secrets Manager entfernt Versionen ohne Label, wenn es mehr als 100 davon gibt. Versionen, die jünger als 24 Stunden sind, werden nicht entfernt. Wenn Sie den Secret-Wert mehr als einmal alle 10 Minuten aktualisieren, erstellen Sie mehr Versionen als Secrets Manager entfernt, und Sie erreichen das Kontingent für Secret-Versionen.

Gehen Sie wie folgt vor, um ein Secret zu aktualisieren: [UpdateSecret](#) oder [ReplicateSecretToRegions](#). Weitere Informationen finden Sie unter [the section called "AWS SDKs"](#).

Finde Geheimnisse in AWS Secrets Manager

Wenn Sie nach Geheimnissen ohne Filter suchen, stimmt Secrets Manager mit Schlüsselwörtern im geheimen Namen, der Beschreibung, dem Tag-Schlüssel und dem Tag-Wert überein. Bei der Suche ohne Filter wird die Groß-/Kleinschreibung nicht beachtet und Sonderzeichen wie Leerzeichen, /, _, =, # ignoriert und nur Zahlen und Buchstaben verwendet. Wenn Sie ohne Filter suchen, analysiert Secrets Manager die Suchzeichenfolge, um sie in separate Wörter zu konvertieren. Die Wörter sind durch jede Änderung von Groß- zu Kleinbuchstaben, von Buchstabe zu Zahl oder von Zahl/Buchstabe zu Satzzeichen getrennt. Wenn Sie beispielsweise den Suchbegriff `credsDatabase#892` eingeben, wird nach `creds`, `Database`, und `892` im Namen, in der Beschreibung und im Tag-Schlüssel und -Wert gesucht.

Secrets Manager generiert einen CloudTrail Protokolleintrag, wenn Sie Geheimnisse auflisten. Weitere Informationen finden Sie unter [the section called "Loggen Sie sich mit AWS CloudTrail"](#).

Sie können die folgenden Filter auf Ihre Suche anwenden:

Name

Stimmt mit dem Anfang geheimer Namen überein; Groß-/Kleinschreibung beachten. Name: **Data** gibt beispielsweise ein Geheimnis namens DatabaseSecret zurück, aber nicht databaseSecret oder MyData.

Beschreibung

Entspricht den Wörtern in geheimen Beschreibungen, wobei die Groß- und Kleinschreibung nicht beachtet wird. Beschreibung: **My Description** ordnet beispielsweise Secrets den folgenden Beschreibungen zu:

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

Verwaltet von

Findet Geheimnisse, die von Diensten außerhalb von AWS z. B. CyberArk oder verwaltet HashiCorp werden.

Besitzender Service

Entspricht dem Anfang des ID-Präfixes des Verwaltungsservices, wobei Groß- und Kleinschreibung nicht beachtet wird. **my-ser** entspricht beispielsweise von Services verwalteten Secrets mit dem Präfix my-serv und my-service. Weitere Informationen finden Sie unter [Verwaltete Geheimnisse](#).

Replizierte Objekte

Sie können nach primären Geheimnissen, Replikatgeheimnissen oder Geheimnissen filtern, die nicht repliziert werden.

Tag-Schlüssel

Entspricht dem Anfang von Tag-Schlüsseln; Groß- und Kleinschreibung wird beachtet. Tag-Schlüssel: **Prod** gibt beispielsweise Geheimnisse mit dem Tag Production und Prod1 zurück, aber keine Geheimnisse mit dem Tag prod oder 1 Prod.

Tag-Werte

Entspricht dem Anfang von Tag-Werten; Groß- und Kleinschreibung wird beachtet. Tag-Wert: **Prod** gibt beispielsweise Geheimnisse mit dem Tag Production und Prod1 zurück, aber keine Geheimnisse mit dem Tag-Wert prod oder 1 Prod.

Secrets Manager ist ein regionaler Service und es werden nur Secrets innerhalb der ausgewählten Region zurückgegeben.

AWS CLI

Example Auflisten der Secrets in Ihrem Konto

Das folgende [list-secrets](#)-Beispiel erhält eine Liste der Secrets in Ihrem Konto.

```
aws secretsmanager list-secrets
```

Example Filtern der Liste der Secrets in Ihrem Konto

Das folgende [list-secrets](#)-Beispiel erhält eine Liste der Secrets in Ihrem Konto, deren Name Test enthält. Bei dem Filtern nach Namen muss die Groß- und Kleinschreibung beachtet werden.

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

Example Findet Geheimnisse, die von anderen AWS Diensten verwaltet werden

Im folgenden [list-secrets](#)-Beispiel wird eine Liste von Secrets abgerufen, die von einem Service verwaltet werden. Geben Sie den Service anhand der ID an. Weitere Informationen finden Sie unter [Verwaltete Geheimnisse](#).

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

AWS SDK

Um mithilfe eines der AWS SDKs nach Geheimnissen zu suchen, verwenden Sie [ListSecrets](#). Weitere Informationen finden Sie unter [the section called "AWS SDKs"](#).

Ein AWS Secrets Manager Geheimnis löschen

Aufgrund der kritischen Natur von Geheimnissen wird das AWS Secrets Manager vorsätzliche Löschen eines Geheimnisses erschwert. Secrets Manager löscht Secrets nicht sofort. Stattdessen werden die Secrets von Secrets Manager sofort unzugänglich gemacht und zum Löschen nach einem Wiederherstellungsfenster von mindestens sieben Tagen vorgesehen. Vorher gelöschte Secrets können bis zum Ablauf des Wiederherstellungsfensters wiederhergestellt werden. Es fallen keine Gebühren für Secrets an, die Sie zum Löschen markiert haben.

Sie können ein primäres Geheimnis nicht löschen, wenn es in andere Regionen repliziert wird. Löschen Sie zuerst die Replikat und löschen Sie dann das primäre Geheimnis. Wenn Sie ein Replikat löschen, wird es sofort gelöscht.

Die Version eines Secrets kann nicht direkt gelöscht werden. Stattdessen entfernen Sie mithilfe des AWS SDK AWS CLI oder alle Staging-Labels aus der Version. Die Version wird auf diese Weise als veraltet markiert. Dies ermöglicht Secrets Manager, die Version automatisch im Hintergrund zu löschen.

Wenn Sie nicht wissen, ob eine Anwendung immer noch ein Geheimnis verwendet, können Sie einen CloudWatch Amazon-Alarm einrichten, der Sie über alle Versuche informiert, während des Wiederherstellungsfensters auf ein Geheimnis zuzugreifen. Weitere Informationen finden Sie unter [Überwachen Sie, wann auf AWS Secrets Manager Geheimnisse zugegriffen wird, die gelöscht werden sollen](#).

Zum Löschen eines Secrets benötigen Sie die Berechtigungen `secretsmanager:ListSecrets` und `secretsmanager>DeleteSecret`.


Secrets Manager generiert einen CloudTrail Protokolleintrag, wenn Sie ein Geheimnis löschen. Weitere Informationen finden Sie unter [the section called “Loggen Sie sich mit AWS CloudTrail”](#).

Ein Secret löschen (Konsole)

1. Öffnen Sie die Secrets-Manager-Konsole unter <https://console.aws.amazon.com/secretsmanager/>.
2. Wählen Sie in der Secret-Liste das Secret aus, das Sie löschen möchten.
3. Wählen Sie im Bereich Secret details (Secret-Details) die Option Actions (Aktionen) und danach Delete secret (Secret löschen) aus.

4. Geben Sie im Dialogfeld Disable secret and schedule deletion (Secret deaktivieren und Löschen planen) unter Waiting period (Wartezeit) die Anzahl der Tage ein, die gewartet werden soll. Secrets Manager fügt ein Feld mit dem Namen DeletionDate an und legt es auf das aktuelle Datum und die aktuelle Uhrzeit plus die für das Wiederherstellungsfenster angegebene Anzahl von Tagen fest.
5. Wählen Sie Schedule deletion.

Gelöschte Secrets anzeigen

1. Öffnen Sie die Secrets-Manager-Konsole unter <https://console.aws.amazon.com/secretsmanager/>.
2. Klicken Sie auf der Seite Secrets auf Preferences (Einstellungen)
).
3. Wählen Sie im Dialogfeld „Einstellungen“ die Option Deaktivierte Secrets anzeigen aus und wählen Sie dann Speichern.

Löschen eines Geheimnisreplikats

1. Öffnen Sie die Secrets-Manager-Konsole unter <https://console.aws.amazon.com/secretsmanager/>.
2. Wählen Sie das primäre Geheimnis.
3. Wählen Sie im Bereich Replicate Secret (Secret replizieren) das Secret-Replikat.
4. Wählen Sie im Menü Actions (Aktionen) die Option Delete Replica (Replikat löschen).

AWS CLI

Example Löschen eines Secrets

Im folgenden [delete-secret](#)-Beispiel wird ein Secret gelöscht. Sie können das Geheimnis [restore-secret](#) bis zu dem Datum und der Uhrzeit im DeletionDate Antwortfeld wiederherstellen. Um ein Secret zu löschen, das in andere Regionen repliziert wird, entfernen Sie zuerst die zugehörigen Replikate mit [remove-regions-from-replication](#) und rufen Sie dann [delete-secret](#) auf.

```
aws secretsmanager delete-secret \
```

```
--secret-id MyTestSecret \  
--recovery-window-in-days 7
```

Example Ein Secret sofort löschen

Das folgende [delete-secret](#)-Beispiel löscht ein Secret sofort und ohne ein Wiederherstellungsfenster. Sie können dieses Secret nicht wiederherstellen.

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

Example Löschen eines Secret-Replikats

Im folgenden [remove-regions-from-replication](#)-Beispiel wird ein Replikat-Secret in eu-west-3 gelöscht. Um ein primäres Secret zu löschen, das in andere Regionen repliziert wird, entfernen Sie zuerst die Replikate und rufen Sie dann [delete-secret](#) auf.

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

AWS SDK

Verwenden Sie zum Löschen eines Secrets den Befehl [DeleteSecret](#). Verwenden Sie zum Löschen einer Secret-Version den Befehl [UpdateSecretVersionStage](#). Verwenden Sie zum Löschen eines Replikats den Befehl [StopReplicationToReplica](#). Weitere Informationen finden Sie unter [the section called “AWS SDKs”](#).

Ein AWS Secrets Manager Geheimnis wiederherstellen

Secrets Manager betrachtet ein Secret, das zum Löschen vorgesehen ist, als veraltet, und auf dieses wird nicht mehr direkt zugegriffen. Nach Ablauf des Wiederherstellungsfensters löscht Secrets Manager das Secret endgültig. Sobald das Secret von Secrets Manager gelöscht wird, können Sie es nicht wiederherstellen. Vor Ablauf des Wiederherstellungsfensters können Sie das Secret wiederherstellen und wieder zugänglich machen. Dadurch wird das Feld `DeletionDate` entfernt, wodurch das vorgesehene endgültige Löschen aufgehoben wird.

Um ein Secret und die Metadaten mithilfe der Konsole wiederherzustellen, müssen Sie über die Berechtigungen `secretsmanager:ListSecrets` und `secretsmanager:RestoreSecret` verfügen:

Secrets Manager generiert einen CloudTrail Protokolleintrag, wenn Sie ein Geheimnis wiederherstellen. Weitere Informationen finden Sie unter [the section called “Loggen Sie sich mit AWS CloudTrail”](#).

Ein Secret wiederherstellen (Konsole)

1. Öffnen Sie die Secrets-Manager-Konsole unter <https://console.aws.amazon.com/secretsmanager/>.
2. Wählen Sie in der Secret-Liste das Secret aus, das Sie wiederherstellen möchten.

Wählen Sie Preferences (Einstellungen)



aus, wenn in Ihrer Secret-Liste keine gelöschten Secrets angezeigt werden. Wählen Sie im Dialogfeld „Einstellungen“ die Option Deaktivierte Secrets anzeigen aus und wählen Sie dann Speichern.

3. Wählen Sie im Bereich Secret details (Secret-Details) die Option Cancel deletion (Löschen aufheben).
4. Klicken Sie im Dialogfeld Cancel secret deletion (Löschen des Secrets aufheben) auf Cancel deletion (Löschen aufheben).

AWS CLI

Example Ein zuvor gelöscht Secret wiederherstellen

Das folgende [restore-secret](#)-Beispiel stellt ein Secret wieder her, dessen Löschung zuvor geplant war.

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

AWS SDK

Zum Wiederherstellen eines zum Löschen markierten Secrets verwenden Sie den Befehl [RestoreSecret](#). Weitere Informationen finden Sie unter [the section called “AWS SDKs”](#).

AWS Secrets Manager-Secrets markieren

In Secrets Manager ist ein Tag eine einfache Markierung, die aus einem vom Kunden definierten Schlüssel und einem optionalen Wert besteht. Sie können Tags verwenden, um die Verwaltung, Suche und Filterung von Geheimnissen und anderen Ressourcen in Ihrem AWS-Konto zu vereinfachen. Verwenden Sie beim Markieren Ihrer Geheimnisse ein Standardbenennungsschema für alle Ihre Ressourcen. Weitere Informationen finden Sie im Whitepaper [Bewährte Methoden für die Markierung](#).

Sie können den Zugriff auf ein Secret gewähren oder verweigern, indem Sie die Tags überprüfen, die dem Secret zugeordnet sind. Weitere Informationen finden Sie unter [the section called “Beispiel: Steuern des Zugriffs auf Secrets mit Tags”](#).

Sie können Secrets anhand von Tags in der Konsole, AWS CLI, und SDKs finden. AWS bietet auch das [Ressourcengruppen](#)-Tool zum Erstellen einer benutzerdefinierten Konsole, die Ihre Ressourcen basierend auf ihren Tags konsolidiert und organisiert. Um Secrets mit einem bestimmten Tag zu finden, lesen Sie [the section called “Finden von Geheimnissen”](#). Secrets Manager unterstützt keine tagbasierte Kostenzuweisung.

Speichern Sie niemals sensible Informationen für ein Secret in einem Tag.

Informationen zu Tag-Kontingenten und Benennungsbeschränkungen finden Sie unter [Service Quotas für Tagging](#) im allgemeinen AWS-Referenzleitfaden. Bei Tags muss die Groß- und Kleinschreibung beachtet werden.

Secrets Manager generiert einen CloudTrail-Protokolleintrag, wenn Sie ein Secret markieren oder die Markierung aufheben. Weitere Informationen finden Sie unter [the section called “Loggen Sie sich mit AWS CloudTrail”](#).

Die Tags für Ihr Secret (Konsole) ändern

1. Öffnen Sie die Secrets-Manager-Konsole unter <https://console.aws.amazon.com/secretsmanager/>.
2. Wählen Sie aus der Liste der Secrets Ihr Secret aus.
3. Wählen Sie auf der Seite zu den Secret-Details auf der Registerkarte Tags die Option Tags bearbeiten aus. Bei den Namen und Werten der Tag-Schlüssel wird zwischen Groß- und Kleinschreibung unterschieden, und die Tag-Schlüssel müssen eindeutig sein.

AWS CLI

Example Einem Secret ein Tag hinzufügen

Im folgenden [tag-resource](#)-Beispiel wird gezeigt, wie Sie ein Tag mit Abkürzungssyntax anfügen.

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

Example Einem Secret mehrere Tags hinzufügen

Das folgende [tag-resource](#)-Beispiel fügt zwei Schlüssel-/Wert-Tags an ein Secret an.

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

Example Tags von einem Secret entfernen

Im folgenden [untag-resource](#)-Beispiel werden zwei Tags aus einem Secret entfernt. Für jedes Tag werden sowohl der Schlüssel als auch der Wert entfernt.

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

AWS-SDK

Um Tags für Ihr Secret zu ändern, verwenden Sie [TagResource](#) oder [UntagResource](#). Weitere Informationen finden Sie unter [the section called “AWS SDKs”](#).

Replizieren Sie AWS Secrets Manager Geheimnisse in allen Regionen

Sie können Ihre Secrets in mehreren Regionen replizieren, um Anwendungen AWS-Regionen zu unterstützen, die über diese Regionen verteilt sind, um die Anforderungen an regionalen Zugriff und geringe Latenz zu erfüllen. Bei Bedarf können Sie ein geheimes Replikat zu [einem eigenständigen Replikatgeheimnis heraufstufen](#) und es dann unabhängig für die Replikation einrichten. Secrets Manager repliziert die verschlüsselten geheimen Daten und Metadaten wie Tags und Ressourcenrichtlinien in den angegebenen Regionen.

Der ARN für ein repliziertes Secret ist bis auf die Region mit dem primären Secret identisch, zum Beispiel:

- Primäres Secret: `arn:aws:secretsmanager:Region1:123456789012:secret:MySecret-a1b2c3`
- Replikat-Secret: `arn:aws:secretsmanager:Region2:123456789012:secret:MySecret-a1b2c3`

Preisinformationen für Replikat-Secrets finden Sie unter [AWS Secrets Manager -Preise](#).

Wenn Sie die Datenbank Anmeldeinformation für eine Quelldatenbank speichern, die in anderen Regionen repliziert wird, enthält das Secret Verbindungsinformationen für die Quelldatenbank. Wenn Sie dann das Secret replizieren, sind die Replikate Kopien des Quellsecret und enthalten dieselben Verbindungsinformationen. Sie können dem Secret zusätzliche Schlüssel/Wert-Paare hinzufügen, um regionale Verbindungsinformationen aufzunehmen.

Wenn Sie die Rotation für Ihr primäres Secret aktivieren, rotiert Secrets Manager das Secret in der primären Region, und der neue Geheimniswert wird an alle zugeordneten Replikat-Geheimnisse weitergegeben. Sie müssen die Drehung nicht für alle Geheimnis-Replikate einzeln verwalten.

Sie können geheime Daten in all Ihren aktivierten AWS Regionen replizieren. Wenn Sie Secrets Manager jedoch in speziellen AWS Regionen wie AWS GovCloud (US) oder Regionen in China verwenden, können Sie Secrets und Replicas nur in diesen speziellen AWS Regionen konfigurieren. Sie können ein Geheimnis in Ihren aktivierten AWS Regionen nicht in eine spezielle Region replizieren oder Geheimnisse aus einer speziellen Region in eine kommerzielle Region replizieren.

Bevor Sie ein Geheimnis in eine andere Region replizieren können, müssen Sie diese Region aktivieren. Weitere Informationen finden Sie unter [Verwalten von AWS -Regionen](#).

Sie können ein Secret in mehreren Regionen verwenden, ohne dass es repliziert wurde, indem Sie den Secrets-Manager-Endpunkt in der Region aufrufen, in der das Secret gespeichert ist. Eine Liste der Endpunkte finden Sie unter [the section called “Secrets-Manager-Endpunkte”](#). Informationen zur Verbesserung der Ausfallsicherheit Ihres Workloads mithilfe von Replikation finden Sie unter [Disaster Recovery \(DR\) -Architektur unter AWS Teil I: Strategien für die Wiederherstellung in der Cloud](#).

Secrets Manager generiert einen CloudTrail Protokolleintrag, wenn Sie ein Geheimnis replizieren. Weitere Informationen finden Sie unter [the section called “Loggen Sie sich mit AWS CloudTrail ”](#).

So replizieren Sie ein Secret in andere Regionen (Konsole)

1. Öffnen Sie die Secrets-Manager-Konsole unter <https://console.aws.amazon.com/secretsmanager/>.
2. Wählen Sie aus der Liste der Secrets Ihr Secret aus.
3. Führen Sie auf der Seite zu den Secret-Details auf der Registerkarte Replikation einen der folgenden Schritte aus:
 - Wenn Ihr Geheimnis nicht repliziert wird, wählen Sie Geheimnis in andere Regionen replizieren aus.
 - Wenn Ihr Geheimnis repliziert wird, wählen Sie im Abschnitt Geheimnis replizieren Region hinzufügen aus.
4. Führen Sie im Dialogfeld Replikatwert hinzufügen die folgenden Schritte aus:
 - a. Wählen Sie für AWS -Region die Region aus, in die Sie das Secret replizieren möchten.
 - b. (Optional) Wählen Sie für Verschlüsselungsschlüssel einen KMS-Schlüssel, mit dem Sie das Secret verschlüsseln möchten. Der Schlüssel muss in der Replikat-Region sein.
 - c. (Optional) Zum Hinzufügen einer weiteren Region wählen Sie Weitere Regionen hinzufügen aus.
 - d. Wählen Sie Replikat.

Sie kehren zur Details-Seite des Geheimnisses zurück. Wählen Sie im Abschnitt Geheimnis replizieren die Region mit dem Replikatsstatus aus.

AWS CLI

Example Secret in eine andere Region replizieren

Im folgenden [replicate-secret-to-regions](#)-Beispiel wird ein Secret in eu-west-3 repliziert. Das Replikat ist mit dem AWS verwalteten Schlüssel aws/secretsmanager verschlüsselt.

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

Example Erstellen Sie ein Geheimnis und replizieren Sie es

Das folgende [Beispiel](#) erstellt ein Geheimnis und repliziert es nach eu-west-3. Das Replikat ist mit dem verwalteten Schlüssel aws/secretsmanager verschlüsselt. AWS

```
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

Um ein Geheimnis zu replizieren, verwenden Sie den [ReplicateSecretToRegions](#)-Befehl. Weitere Informationen finden Sie unter [the section called “AWS SDKs”](#).

Hochstufen eines Secret-Replikats zu einem eigenständigen Secret in AWS Secrets Manager

Ein Replikat-Geheimnis ist ein Geheimnis, das von einem primären in eine andere AWS-Region repliziert wird. Es hat den gleichen geheimen Wert und die gleichen Metadaten wie die primäre, kann aber mit einem anderen KMS-Schlüssel verschlüsselt werden. Ein Replikatgeheimnis kann nicht unabhängig von seinem primären Geheimnis aktualisiert werden, mit Ausnahme seines Verschlüsselungsschlüssels. Das Anheben eines Replikatgeheimnisses trennt das Replikatgeheimnis vom primären Geheimnis und macht das Replikat zu einem eigenständigen Geheimnis. Änderungen am primären Geheimnis werden nicht auf das eigenständige Geheimnis repliziert.

Möglicherweise möchten Sie ein Replikat-Geheimnis als Notfallwiederherstellungslösung zu einem eigenständigen Geheimnis hochstufen, wenn das primäre Geheimnis nicht verfügbar ist. Oder Sie möchten ein Replikat zu einem eigenständigen Geheimnis hochstufen, wenn Sie die Drehung für das Replikat aktivieren möchten.

Wenn Sie ein Replikat heraufstufen, stellen Sie sicher, dass die entsprechenden Anwendungen aktualisiert werden, um das eigenständige Geheimnis zu verwenden.

Secrets Manager generiert einen CloudTrail-Protokolleintrag, wenn Sie ein Secret hochstufen. Weitere Informationen finden Sie unter [the section called “Loggen Sie sich mit AWS CloudTrail”](#).

So stufen Sie ein Replikatgeheimnis hoch (Konsole)

1. Melden Sie sich bei Secrets Manager an unter <https://console.aws.amazon.com/secretsmanager/>.
2. Navigieren Sie zur Replikat-Region.
3. Wählen Sie auf der Seite Secrets das Replikat-Secret aus.
4. Wählen Sie auf der Seite mit den Details zum Replikat-Secret Promote to standalone secret (Auf eigenständiges Secret heraufstufen).
5. Geben Sie im Dialogfeld Promote replica to standalone secret (Replikat zu eigenständigem Secret heraufstufen) die Region ein und wählen Sie dann Promote replica (Replikat heraufstufen) aus.

AWS CLI

Example Ein Replikat-Secret zu einem primären heraufstufen

Im folgenden [stop-replication-to-replica](#)-Beispiel wird die Verknüpfung zwischen einem Replikat-Secret und dem primären entfernt. Das Replikat-Secret wird in der Replikat-Region zum primären Secret heraufgestuft. Sie müssen [stop-replication-to-replica](#) innerhalb der Replikatregion aufrufen.

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

AWS-SDK

Verwenden Sie den [StopReplicationToReplica](#)-Befehl, um ein Replikat zu einem eigenständigen Geheimnis hochzustufen. Sie müssen diesen Befehl aus der Replikat-Geheimnis-Region aufrufen. Weitere Informationen finden Sie unter [the section called "AWS SDKs"](#).

AWS Secrets Manager Replikation verhindern

Da Geheimnisse mit [ReplicateSecretToRegions](#) oder bei ihrer Erstellung repliziert werden können, empfehlen wir Ihnen [CreateSecret](#), Aktionen zu verhindern, die den Parameter enthalten, wenn Sie verhindern möchten, dass Benutzer Geheimnisse replizieren. `AddReplicaRegions` Sie können in Ihren Berechtigungsrichtlinien eine `Condition` Erklärung verwenden, um nur Aktionen zuzulassen, bei denen keine Replikatbereiche hinzugefügt werden. In den folgenden Richtlinienbeispielen finden Sie Hinweise zu Bedingungen, die Sie verwenden können.

Example Sperren Sie die Replikationsberechtigung

Das folgende Richtlinienbeispiel zeigt, wie alle Aktionen zugelassen werden, bei denen keine Replikatbereiche hinzugefügt werden. Dadurch wird verhindert, dass Benutzer Geheimnisse sowohl `ReplicateSecretToRegions` mit als auch replizieren. `CreateSecret`

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

Example Erlauben Sie die Replikationsberechtigung nur für bestimmte Regionen

Die folgende Richtlinie zeigt, wie Sie Folgendes zulassen können:

- Geheimnisse ohne Replikation erstellen
- Erstellen Sie Geheimnisse mit Replikation nur in Regionen in den USA und Kanada
- Replizieren Sie Geheimnisse nur in Regionen in den USA und Kanada

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

Fehlerbehebung bei der AWS Secrets Manager Replikation

Nachfolgend sind einige Gründe für das Fehlschlagen der Replikation ausgeführt.

Ein Secret mit demselben Namen ist in der ausgewählten Region bereits vorhanden.

Zur Behebung dieses Problems können Sie das Secret mit dem doppelten Namen in der Replikatregion überschreiben. Wiederholen Sie die Replikation. Wählen Sie im Dialogfeld Replikation wiederholen die Option Überschreiben.

Keine Berechtigungen für den KMS-Schlüssel verfügbar, um die Replikation abzuschließen

Secrets Manager entschlüsselt zunächst das Secret, bevor die erneute Verschlüsselung mit dem neuen KMS-Schlüssel in der Replikatregion erfolgt. Dieser Fehler tritt auf, wenn Sie über keine `kms:Decrypt`-Berechtigung für den Verschlüsselungsschlüssel in der primären Region verfügen. Um das replizierte Secret mit einem anderen KMS-Schlüssel als `aws/secretsmanager` zu verschlüsseln, benötigen Sie `kms:GenerateDataKey` und `kms:Encrypt` zum Schlüssel. Siehe [the section called "Berechtigungen für den KMS-Schlüssel"](#).

Der KMS-Schlüssel wurde deaktiviert oder wurde nicht gefunden

Wenn der Verschlüsselungsschlüssel in der primären Region deaktiviert oder gelöscht ist, kann Secrets Manager das Secret nicht replizieren. Dieser Fehler kann auch dann auftreten, wenn Sie den Verschlüsselungsschlüssel geändert haben, wenn das Secret [Versionen mit benutzerdefinierter Bezeichnung](#) enthält, die mit dem deaktivierten oder gelöschten Verschlüsselungsschlüssel verschlüsselt wurden. Informationen darüber, wie Secrets Manager die Verschlüsselung durchführt, finden Sie unter [the section called "Ver- und Entschlüsselung von Secrets"](#). Um dieses Problem zu umgehen, können Sie die Secret-Versionen neu erstellen, sodass Secrets Manager sie mit dem aktuellen Verschlüsselungsschlüssel verschlüsselt. Weitere Informationen finden Sie unter [Ändern des Verschlüsselungsschlüssels für ein Secret](#). Versuchen Sie dann die Replikation erneut.

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

Sie haben die Region, in der die Replikation stattfindet, nicht aktiviert.

Informationen zum Aktivieren einer Region finden Sie unter [AWS Regionen verwalten](#), im Referenzhandbuch zur AWS Kontoverwaltung.

Hol dir Geheimnisse von AWS Secrets Manager

Secrets Manager generiert einen CloudTrail Protokolleintrag, wenn Sie ein Geheimnis abrufen. Weitere Informationen finden Sie unter [the section called “Loggen Sie sich mit AWS CloudTrail”](#).

Sie können geheime Werte wie folgt abrufen:

- [Holen Sie sich einen geheimen Secrets Manager Manager-Wert mit Java](#)
- [Holen Sie sich einen geheimen Secrets Manager Manager-Wert mit Python](#)
- [Rufen Sie mithilfe von .NET einen geheimen Wert von Secrets Manager ab](#)
- [Holen Sie sich mit Go einen geheimen Wert für Secrets Manager](#)
- [Rufen Sie mit dem AWS C++-SDK einen geheimen Wert von Secrets Manager ab](#)
- [Rufen Sie mithilfe des JavaScript AWS SDK einen geheimen Wert von Secrets Manager ab](#)
- [Holen Sie sich mit dem Kotlin AWS SDK einen geheimen Wert für Secrets Manager](#)
- [Holen Sie sich mit dem AWS PHP-SDK einen geheimen Wert für Secrets Manager](#)
- [Holen Sie sich mit dem Ruby AWS SDK einen geheimen Wert für Secrets Manager](#)
- [Holen Sie sich mit dem Rust AWS SDK einen geheimen Wert für Secrets Manager](#)
- [Holen Sie sich einen geheimen Wert mit dem AWS CLI](#)
- [Rufen Sie mithilfe der AWS Konsole einen geheimen Wert ab](#)
- [Benutze AWS Secrets Manager Geheimnisse in AWS Batch](#)
- [Holen Sie sich ein AWS Secrets Manager Geheimnis in einer AWS CloudFormation Ressource](#)
- [Verwenden Sie AWS Secrets Manager Geheimnisse in Amazon Elastic Kubernetes Service](#)
- [Verwenden Sie AWS Secrets Manager Geheimnisse in GitHub Jobs](#)
- [Verwenden von AWS Secrets Manager-Secrets in AWS IoT Greengrass](#)
- [Verwenden Sie AWS Secrets Manager Geheimnisse in AWS Lambda Funktionen](#)
- [Verwenden von AWS Secrets Manager-Secrets in Parameter Store](#)

Holen Sie sich einen geheimen Secrets Manager Manager-Wert mit Java

In Anwendungen können Sie Ihre Geheimnisse abrufen, indem Sie eines der SDKs aufrufen `GetSecretValue` oder `BatchGetSecretValue` in einem der AWS SDKs aufrufen. Wir empfehlen

jedoch, Ihre Secret-Werte mithilfe des clientseitigen Cachings zu speichern. Das Caching von Secrets verbessert die Geschwindigkeit und senkt Ihre Kosten.

Um mithilfe der Anmeldeinformationen in einem Secret eine Verbindung zu einer Datenbank herzustellen, können Sie die Secrets Manager SQL Connection-Treiber verwenden, die den JDBC-Basistreiber umschließen. Dabei wird auch clientseitiges Caching verwendet, sodass die Kosten für den Aufruf von Secrets Manager Manager-APIs gesenkt werden können.

Themen

- [Holen Sie sich einen geheimen Secrets Manager-Wert mithilfe von Java mit clientseitigem Caching](#)
- [Stellen Sie mithilfe von JDBC mit geheimen Anmeldeinformationen eine Connect zu einer SQL-Datenbank her AWS Secrets Manager](#)
- [Rufen Sie mit dem Java AWS SDK einen geheimen Wert von Secrets Manager ab](#)

Holen Sie sich einen geheimen Secrets Manager-Wert mithilfe von Java mit clientseitigem Caching

Wenn Sie ein Secret abrufen, können Sie die Java-basierte Caching-Komponente von Secrets Manager verwenden, um es für zukünftige Verwendung zu cachieren. Das Abrufen eines gecacheten Secrets ist schneller als das Abrufen aus Secrets Manager. Da für den Aufruf von Secrets-Manager-APIs Kosten anfallen, kann die Verwendung eines Caches Ihre Kosten senken. Alle Möglichkeiten, wie Sie Secrets abrufen können, finden Sie unter [Holen Sie sich Geheimnisse](#).

Die Cache-Richtlinie ist Least Recently Used (LRU). Wenn der Cache also ein Secret verwerfen muss, verwirft er das am wenigsten verwendete Secret. Standardmäßig aktualisiert der Cache Secrets jede Stunde. Sie können konfigurieren, [wie oft das Secret im Cache aktualisiert wird](#), und Sie können [den Secret-Abruf anbinden](#), um weitere Funktionalität hinzuzufügen.

Der Cache erzwingt keine Garbage Collection, sobald Cache-Referenzen freigegeben wurden. Die Cache-Implementierung beinhaltet keine Cache-Invalidierung. Die Cache-Implementierung konzentriert sich auf den Cache selbst und ist weder sicherheitsgehärtet noch fokussiert. Wenn Sie zusätzliche Sicherheit benötigen, z. B. das Verschlüsseln von Elementen im Cache, verwenden Sie die bereitgestellten Schnittstellen und abstrakten Methoden.

Zum Verwenden der Komponente ist Folgendes erforderlich:

- Eine Java-8- oder eine höhere Entwicklungsumgebung. Siehe [Java SE-Downloads](#) auf der Oracle-Website.

- Das AWS SDK 1.x für Java. Sie können beide Versionen des AWS SDK for Java in Ihren Projekten verwenden. Weitere Informationen finden Sie unter [Verwenden des SDK for Java 1.x und side-by-side 2.x](#).

Informationen zum Herunterladen des Quellcodes finden Sie unter [Secrets Manager Java-basierte Caching-Client-Komponente](#) auf GitHub

Um die Komponente Ihrem Projekt hinzuzufügen, fügen Sie in Ihrer Datei Maven pom.xml die folgende Abhängigkeit ein. Weitere Informationen zu Maven finden Sie im [Handbuch „Erste Schritte“](#) auf der Website von Apache Maven.

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

Erforderliche Berechtigungen:

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

Weitere Informationen finden Sie unter [Berechtigungsreferenz](#).

Referenz

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

Example Ein Secret abrufen

Das folgende Codebeispiel zeigt eine Lambda-Funktion, die eine Secret-Zeichenfolge abruft. Es folgt der [bewährten Methode](#), den Cache außerhalb des Funktionshandlers zu instanziiieren, ruft also die API nicht weiter auf, wenn Sie die Lambda-Funktion erneut aufrufen.

```
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

Ein In-Memory-Cache für von Secrets Manager angeforderte Secrets. Sie verwenden [the section called “getSecretString”](#) oder [the section called “getSecretBinary”](#), um ein Secret aus dem Cache abzurufen. Sie können die Cache-Einstellungen konfigurieren, indem Sie ein [the section called “SecretCacheConfiguration”](#)-Objekt im Konstruktor übergeben.

Weitere Informationen hierzu einschließlich Beispielen finden Sie unter [the section called “Java mit clientseitigem Caching”](#).

Konstruktoren

```
public SecretCache()
```

Standardkonstruktor für ein SecretCache-Objekt.

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

Konstruiert einen neuen Cache mit einem Secrets-Manager-Manager-Client, der mit dem bereitgestellten [AWSSecretsManagerClientBuilder](#) erstellt wurde. Verwenden Sie diesen Konstruktor, um den Secrets Manager Manager-Client anzupassen, z. B. um eine bestimmte Region oder einen bestimmten Endpunkt zu verwenden.

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

Konstruiert einen neuen Secret-Cache mit dem bereitgestellten [AWSSecretsManagerClient](#). Verwenden Sie diesen Konstruktor, um den Secrets Manager Manager-Client anzupassen, z. B. um eine bestimmte Region oder einen bestimmten Endpunkt zu verwenden.

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

Konstruiert einen neuen Secret-Cache mit dem bereitgestellten [the section called "SecretCacheConfiguration"](#).

Methoden

getSecretString

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

Ruft ein String-Secret von Secrets Manager ab. Gibt eine [String](#) zurück.

getSecretBinary

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

Ruft ein binäres Secret von Secrets Manager ab. Gibt eine [ByteBuffer](#) zurück.

refreshNow

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

Zwingt den Cache zur Aktualisierung. Gibt `true` zurück, wenn die Aktualisierung fehlerfrei abgeschlossen ist, sonst `false`.

close

```
public void close()
```

Schließt den Cache.

SecretCacheConfiguration

Cache-Konfigurationsoptionen für ein [the section called "SecretCache"](#), z. B. maximale Cachegröße und Time to Live (TTL) für gecachte Secrets.

Konstruktor

```
public SecretCacheConfiguration
```

Standardkonstruktor für ein `SecretCacheConfiguration`-Objekt.

Methoden

getClient

```
public AWSSecretsManager getClient()
```

Gibt den neuen [AWSSecretsManagerClient](#) zurück, von dem der Cache Secrets abrufft.

setClient

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

Legt den neuen [AWSSecretsManagerClient](#) fest, von dem der Cache Secrets abrufft.

getCacheHook

```
public SecretCacheHook getCacheHook()
```

Gibt die neue [the section called "SecretCacheHook"](#)-Schnittstelle zurück, die zum Anbinden von Cache-Updates verwendet wird.

setCacheHook

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

Legt die neue [the section called "SecretCacheHook"](#)-Schnittstelle fest, die zum Anbinden von Cache-Updates verwendet wird.

getMaxCacheGröße

```
public int getMaxCacheSize()
```

Gibt die maximale Cachegröße zurück. Der Standardwert ist 1 024 Secrets.

setMaxCacheGröße

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

Legt die maximale Cachegröße fest. Der Standardwert ist 1 024 Secrets.

getCacheItemTTL

```
public long getCacheItemTTL()
```

Gibt die TTL in Millisekunden für die gecachelten Elemente zurück. Wenn ein gecachetes Secret diese TTL überschreitet, ruft der Cache eine neue Kopie des Secrets aus dem [AWSecretsManagerClient](#) ab. Der Standardwert beträgt 1 Stunde in Millisekunden.

Der Cache aktualisiert das Secret synchron, wenn das Secret nach der TTL angefordert wird. Wenn die synchrone Aktualisierung fehlschlägt, gibt der Cache das veraltete Secrets zurück.

setCacheItemTTL

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

Legt die TTL in Millisekunden für die gecachelten Elemente fest. Wenn ein gecachetes Secret diese TTL überschreitet, ruft der Cache eine neue Kopie des Secrets aus dem [AWSecretsManagerClient](#) ab. Der Standardwert beträgt 1 Stunde in Millisekunden.

getVersionStage

```
public String getVersionStage()
```

Gibt die Version von Secrets zurück, die Sie cachen möchten. Weitere Informationen hierzu finden Sie unter [Secret-Versionen](#). Der Standardwert ist "AWSCURRENT".

setVersionStage

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

Legt die Version von Secrets fest, die Sie cachen möchten. Weitere Informationen hierzu finden Sie unter [Secret-Versionen](#). Der Standardwert ist "AWSCURRENT".

SecretCacheConfiguration mit dem Kunden

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

Legt den neuen [AWSecretsManagerClient](#) zum Abrufen von Secrets fest. Gibt das aktualisierte SecretCacheConfiguration-Objekt mit der neuen Einstellung zurück.

SecretCacheConfiguration mit CacheHook

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

Legt die Schnittstelle fest, die zum Anbinden des In-Memory-Cache verwendet wird. Gibt das aktualisierte `SecretCacheConfiguration`-Objekt mit der neuen Einstellung zurück.

SecretCacheConfiguration withMaxCacheGröße

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

Legt die maximale Cachegröße fest. Gibt das aktualisierte `SecretCacheConfiguration`-Objekt mit der neuen Einstellung zurück.

SecretCacheConfiguration withCacheItemTTL

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

Legt die TTL in Millisekunden für die gecacheten Elemente fest. Wenn ein gecachetes Secret diese TTL überschreitet, ruft der Cache eine neue Kopie des Secrets aus dem [AWS Secrets Manager Client](#) ab. Der Standardwert beträgt 1 Stunde in Millisekunden. Gibt das aktualisierte `SecretCacheConfiguration`-Objekt mit der neuen Einstellung zurück.

SecretCacheConfiguration withVersionStage

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

Legt die Version von Secrets fest, die Sie cachen möchten. Weitere Informationen hierzu finden Sie unter [Secret-Versionen](#). Gibt das aktualisierte `SecretCacheConfiguration`-Objekt mit der neuen Einstellung zurück.

SecretCacheHook

Eine Schnittstelle für das Anbinden eines [the section called "SecretCache"](#), um Aktionen mit dem im Cache gespeicherten Secret durchzuführen.

put

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

Bereitet das Objekt für das Speichern im Cache vor.

Gibt das Objekt zurück, das im Cache gespeichert werden soll.

get

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

Leitet das Objekt aus dem gecacheten Objekt ab.

Gibt das Objekt zurück, das vom Cache zurückgegeben werden soll.

Stellen Sie mithilfe von JDBC mit geheimen Anmeldeinformationen eine Connect zu einer SQL-Datenbank her AWS Secrets Manager

In Java-Anwendungen können Sie die Secrets Manager SQL Connection-Treiber verwenden, um mithilfe der in Secrets Manager gespeicherten Anmeldeinformationen eine Verbindung zu MySQL-, PostgreSQL-, Oracle-, MSSQLServer-, Db2- und Redshift-Datenbanken herzustellen. Jeder Treiber umschließt den JDBC-Basis-Treiber, sodass Sie JDBC-Aufrufe verwenden können, um auf Ihre Datenbank zuzugreifen. Anstatt jedoch einen Benutzernamen und ein Passwort für die Verbindung zu übergeben, geben Sie die ID eines Secrets an. Der Treiber ruft Secrets Manager auf, um den Secret-Wert abzurufen, und verwendet dann die Anmeldeinformationen im Secret, um eine Verbindung mit der Datenbank herzustellen. Der Treiber speichert die Anmeldeinformationen auch mit der [Java-clientseitigen Caching-Library](#), damit zukünftige Verbindungen keinen Anruf bei Secrets Manager erfordern. Standardmäßig wird der Cache jede Stunde aktualisiert sowie wenn ein Secret gedreht wird. Informationen zum Konfigurieren des Cache finden Sie unter [the section called "SecretCacheConfiguration"](#).

Sie können den Quellcode von [herunterladen](#). [GitHub](#)

So verwenden Sie die Secrets-Manager-SQL-Connection-Treiber:

- Ihre Anwendung muss sich in Java 8 oder höher befinden.
- Ihr Secret muss in einem der folgenden Formate vorliegen:
 - Ein [Datenbank-Secret in der erwarteten JSON-Struktur](#). Um das Format zu überprüfen, sehen Sie sich in der Secrets-Manager-Konsole Ihr Secret an und wählen Sie Retrieve secret value (Secret-Wert abrufen) aus. Alternativ rufen Sie im AWS CLI an [get-secret-value](#).
 - Ein von Amazon RDS [verwaltetes Secret](#). Für diese Art von Secret müssen Sie einen Endpunkt und einen Port angeben, wenn Sie die Verbindung herstellen.
 - Ein von Amazon Redshift [verwaltetes Geheimnis](#). Für diese Art von Secret müssen Sie einen Endpunkt und einen Port angeben, wenn Sie die Verbindung herstellen.

Wenn Ihre Datenbank in andere Regionen repliziert wird, geben Sie, um eine Verbindung mit einer Replikat-Datenbank in einer anderen Region herzustellen, beim Erstellen der Verbindung den regionalen Endpunkt und den Port an. Sie können regionale Verbindungsinformationen im

Secret als zusätzliche Schlüssel/Wert-Paare, in SSM-Parameterspeicher-Parametern oder in Ihrer Codekonfiguration speichern.

Um den Treiber zu Ihrem Projekt hinzuzufügen, fügen Sie in Ihrer Maven Build-Datei `pom.xml` die folgende Abhängigkeit für den Treiber hinzu. Weitere Informationen finden Sie unter [Secrets Manager SQL Connection Library](#) auf der Maven-Central-Repository-Website.

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

Der Treiber verwendet [die Standard-Anmeldeinformation des Providers](#). Wenn Sie den Treiber auf Amazon EKS ausführen, werden möglicherweise die Anmeldeinformationen des Knotens, auf dem er ausgeführt wird, anstelle der Dienstkontrolle abgerufen. Um dies zu beheben, fügen Sie Version 1 von `com.amazonaws:aws-java-sdk-sts` als Abhängigkeit zu Ihrer Gradle- oder Maven-Projektdatei hinzu.

So legen Sie eine AWS PrivateLink DNS-Endpunkt-URL und eine Region in der `secretsmanager.properties` Datei fest:

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

Um die primäre Region zu überschreiben, legen Sie die `AWS_SECRET_JDBC_REGION`-Umgebungsvariable fest oder nehmen Sie die folgende Änderung an der `secretsmanager.properties`-Datei vor:

```
drivers.region = region
```

Erforderliche Berechtigungen:

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

Weitere Informationen finden Sie unter [Berechtigungsreferenz](#).

Beispiele:

- [Verbindung zu einer Datenbank herstellen](#)
- [Verbindung herstellen, indem Sie den Endpunkt und den Port angeben](#)
- [c3p0-Verbindungs-Pooling verwenden, um eine Verbindung herzustellen](#)
- [c3p0-Verbindungspooling verwenden, indem Sie den Endpunkt und den Port angeben](#)

Verbindung zu einer Datenbank herstellen

Das folgende Beispiel zeigt, wie Sie mithilfe der Anmeldeinformationen und Verbindungsinformationen in einem Secret eine Verbindung zu einer Datenbank herstellen. Nachdem Sie über die Verbindung verfügen, können Sie JDBC-Aufrufe verwenden, um auf die Datenbank zuzugreifen. Weitere Informationen finden Sie unter [JDBC-Grundlagen](#) auf der Website der Java-Dokumentation.

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);
```

Verbindung herstellen, indem Sie den Endpunkt und den Port angeben

Das folgende Beispiel zeigt, wie Sie mithilfe der Anmeldeinformationen in einem Secret mit einem von Ihnen angegebenen Endpunkt und Port eine Verbindung zu einer Datenbank herstellen.

Die von [Amazon RDS verwalteten Secrets](#) beinhalten nicht den Endpunkt und den Port der Datenbank. Um mithilfe von Master-Anmeldeinformationen in einem von Amazon RDS verwalteten Secret eine Verbindung zu einer Datenbank herzustellen, geben Sie diese in Ihrem Code an.

[Secrets, die in anderen Regionen repliziert werden](#), können die Latenz für die Verbindung zur regionalen Datenbank verbessern, enthalten jedoch keine unterschiedlichen Verbindungsinformationen aus dem Quellsecret. Jedes Replikat ist eine Kopie des Quellsecret. Um regionale Verbindungsinformationen im Secret zu speichern, fügen Sie weitere Schlüssel/Wert-Paare für die Endpunkt- und Portinformationen für die Regionen hinzu.

Nachdem Sie über die Verbindung verfügen, können Sie JDBC-Aufrufe verwenden, um auf die Datenbank zuzugreifen. Weitere Informationen finden Sie unter [JDBC-Grundlagen](#) auf der Website der Java-Dokumentation.

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-Verbindungs-Pooling verwenden, um eine Verbindung herzustellen

Das folgende Beispiel zeigt, wie Sie einen Verbindungspool mit einer `c3p0.properties`-Datei herstellen, die den Treiber verwendet, um Anmeldeinformationen und Verbindungsinformationen aus dem Secret abzurufen. Geben Sie für `user` und `jdbcUrl` die Secret-ID ein, um den Verbindungspool zu konfigurieren. Dann können Sie Verbindungen aus dem Pool abrufen und sie wie andere Datenbankverbindungen verwenden. Weitere Informationen finden Sie unter [JDBC-Grundlagen](#) auf der Website der Java-Dokumentation.

Weitere Informationen zu c3p0 finden Sie unter [c3p0](#) auf der Website von Machinery For Change.

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-Verbindungspooling verwenden, indem Sie den Endpunkt und den Port angeben

Das folgende Beispiel zeigt, wie Sie einen Verbindungspool mit einer `c3p0.properties` Datei einrichten, die den Treiber verwendet, um Anmeldeinformationen in einem geheimen Ordner mit einem von Ihnen angegebenen Endpunkt und Port abzurufen. Dann können Sie Verbindungen aus dem Pool abrufen und sie wie andere Datenbankverbindungen verwenden. Weitere Informationen finden Sie unter [JDBC-Grundlagen](#) auf der Website der Java-Dokumentation.

Die von [Amazon RDS verwalteten Secrets](#) beinhalten nicht den Endpunkt und den Port der Datenbank. Um mithilfe von Master-Anmeldeinformationen in einem von Amazon RDS verwalteten Secret eine Verbindung zu einer Datenbank herzustellen, geben Sie diese in Ihrem Code an.

[Secrets, die in anderen Regionen repliziert werden](#), können die Latenz für die Verbindung zur regionalen Datenbank verbessern, enthalten jedoch keine unterschiedlichen Verbindungsinformationen aus dem Quellsecret. Jedes Replikat ist eine Kopie des Quellsecret. Um regionale Verbindungsinformationen im Secret zu speichern, fügen Sie weitere Schlüssel/Wert-Paare für die Endpunkt- und Portinformationen für die Regionen hinzu.

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
```

Rufen Sie mit dem Java AWS SDK einen geheimen Wert von Secrets Manager ab

In Anwendungen können Sie Ihre Geheimnisse abrufen, indem Sie eines der SDKs aufrufen `GetSecretValue` oder `BatchGetSecretValue` in einem der AWS SDKs aufrufen. Wir empfehlen jedoch, Ihre Secret-Werte mithilfe des clientseitigen Cachings zu speichern. Das Caching von Secrets verbessert die Geschwindigkeit und senkt Ihre Kosten.

- Wenn Sie Datenbankmeldeinformationen im Secret speichern, verwenden Sie die [Secrets-Manager-SQL-Verbindungstreiber](#), um mithilfe der Anmeldeinformationen im Secret eine Verbindung zu einer Datenbank herzustellen.
- Verwenden Sie für andere Arten von Geheimnissen die [Java-basierte Caching-Komponente von Secrets Manager](#) oder rufen Sie das SDK direkt mit oder auf [GetSecretValueBatchGetSecretValue](#)

Die folgenden Codebeispiele zeigen die Verwendung. `GetSecretValue`

Erforderliche Berechtigungen: `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);
    }
}
```

Holen Sie sich einen geheimen Secrets Manager Manager-Wert mit Python

In Anwendungen können Sie Ihre Geheimnisse abrufen, indem Sie eines der SDKs aufrufen `GetSecretValue` oder `BatchGetSecretValue` in einem der AWS SDKs aufrufen. Wir empfehlen jedoch, Ihre Secret-Werte mithilfe des clientseitigen Caching zu speichern. Das Caching von Secrets verbessert die Geschwindigkeit und senkt Ihre Kosten.

Themen

- [Holen Sie sich einen geheimen Secrets Manager-Wert mithilfe von Python mit clientseitigem Caching](#)
- [Rufen Sie mit dem AWS Python-SDK einen geheimen Wert von Secrets Manager ab](#)
- [Holen Sie sich mit dem AWS Python-SDK einen Stapel geheimer Werte von Secrets Manager](#)

Holen Sie sich einen geheimen Secrets Manager-Wert mithilfe von Python mit clientseitigem Caching

Wenn Sie ein Secret abrufen, können Sie die Python-basierte Caching-Komponente von Secrets Manager verwenden, um es für zukünftige Verwendung zu cachen. Das Abrufen eines gecacheten Secrets ist schneller als das Abrufen aus Secrets Manager. Da für den Aufruf von Secrets-Manager-Manager-APIs Kosten anfallen, kann die Verwendung eines Caches Ihre Kosten senken. Alle Möglichkeiten, wie Sie Secrets abrufen können, finden Sie unter [Holen Sie sich Geheimnisse](#).

Die Cache-Richtlinie ist Least Recently Used (LRU). Wenn der Cache also ein Secret verwerfen muss, verwirft er das am wenigsten verwendete Secret. Standardmäßig aktualisiert der Cache Secrets jede Stunde. Sie können konfigurieren, [wie oft das Secret im Cache aktualisiert wird](#), und Sie können [den Secret-Abruf anbinden](#), um weitere Funktionalität hinzuzufügen.

Der Cache erzwingt keine Garbage Collection, sobald Cache-Referenzen freigegeben wurden. Die Cache-Implementierung beinhaltet keine Cache-Invalidierung. Die Cache-Implementierung konzentriert sich auf den Cache selbst und ist weder sicherheitsgehärtet noch fokussiert. Wenn Sie

zusätzliche Sicherheit benötigen, z. B. das Verschlüsseln von Elementen im Cache, verwenden Sie die bereitgestellten Schnittstellen und abstrakten Methoden.

Zum Verwenden der Komponente ist Folgendes erforderlich:

- Python 3.6 oder höher.
- botocore 1.12 oder höher. Siehe [AWS -SDK für Python](#) und [Botocore](#).
- setuptools_scm 3.2 oder höher. Siehe <https://pypi.org/project/setuptools-scm/>.

Informationen zum Herunterladen des Quellcodes finden Sie unter [Secrets Manager Python-basierte Caching-Client-Komponente](#) auf GitHub

Mit dem folgenden Befehl können Sie die Komponente installieren.

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

Erforderliche Berechtigungen:

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

Weitere Informationen finden Sie unter [Berechtigungsreferenz](#).

Referenz

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

Example Ein Secret abrufen

Im folgenden Beispiel wird gezeigt, wie Sie den Secret-Wert eines Secrets mit dem Namen *mysecret* erhalten.

```
import botocore
import botocore.session
```

```
from aws_secretsmanager_caching import SecretCache, SecretCacheConfig

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

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

SecretCache

Ein In-Memory-Cache für aus Secrets Manager abgerufene Secrets. Sie verwenden [the section called “get_secret_string”](#) oder [the section called “get_secret_binary”](#), um ein Secret aus dem Cache abzurufen. Sie können die Cache-Einstellungen konfigurieren, indem Sie ein [the section called “SecretCacheConfig”](#)-Objekt im Konstruktor übergeben.

Weitere Informationen hierzu einschließlich Beispielen finden Sie unter [the section called “Python mit clientseitigem Caching”](#).

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

Die folgenden Methoden sind verfügbar:

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

get_secret_string

Ruft den Secret-String-Wert ab.

Erforderliche Syntax

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

Parameter

- `secret_id` (string) -- [Required] Der Name oder ARN des Secrets.

- `version_stage` (string) -- Die Version der Secrets, die Sie abrufen möchten. [Weitere Informationen finden Sie unter Geheime Versionen](#). Der Standardwert ist 'AWSCURRENT'.

Rückgabetypp

Zeichenfolge

`get_secret_binary`

Ruft den Secret-Binärwert ab.

Erforderliche Syntax

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

Parameter

- `secret_id` (string) -- [Required] Der Name oder ARN des Secrets.
- `version_stage` (string) -- Die Version der Secrets, die Sie abrufen möchten. Weitere Informationen finden Sie unter [Geheime Versionen](#). Der Standardwert ist 'AWSCURRENT'.

Rückgabetypp

[base64-kodierte](#) Zeichenfolge

SecretCacheConfig

Cache-Konfigurationsoptionen für ein [the section called "SecretCache"](#), z. B. maximale Cachegröße und Time to Live (TTL) für gecachte Secrets.

Parameter

`max_cache_size` (int)

Die maximale Cachegröße. Der Standardwert ist 1024 Secrets.

`exception_retry_delay_base` (int)

Die Anzahl der Sekunden, die gewartet werden soll, nachdem eine Ausnahme aufgetreten ist, bevor ein erneuter Versuch gestartet wird. Der Standardwert ist 1.

`exception_retry_growth_factor` (int)

Der Wachstumsfaktor, der für die Berechnung der Wartezeit zwischen Wiederholungen fehlgeschlagener Anfragen verwendet werden soll. Der Standardwert ist 2.

`exception_retry_delay_max` (int)

Maximale Wartezeit in Sekunden zwischen fehlgeschlagenen Anfragen. Der Standardwert ist 3600.

`default_version_stage` (str)

Die Version der Secrets, die Sie cachieren möchten. Weitere Informationen hierzu finden Sie unter [Secret-Versionen](#). Der Standardwert ist 'AWSCURRENT'.

`secret_refresh_interval` (int)

Die Anzahl der Sekunden, die zwischen den Aktualisierungen der gecachten Secret-Informationen gewartet wird. Der Standardwert ist 3600.

`secret_cache_hook` (SecretCacheHook)

Eine Implementierung der SecretCacheHook-abstrakten Klasse. Der Standardwert ist None.

SecretCacheHook

Eine Schnittstelle für das Anbinden eines [the section called "SecretCache"](#), um Aktionen mit dem im Cache gespeicherten Secret durchzuführen.

Die folgenden Methoden sind verfügbar:

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

put

Bereitet das Objekt für das Speichern im Cache vor.

Erforderliche Syntax

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

```
)
```

Parameter

- `obj (object)` -- [Required] Das Secret oder das Objekt, das das Secret enthält.

Rückgabotyp

`object`

get

Leitet das Objekt aus dem gecachten Objekt ab.

Erforderliche Syntax

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

Parameter

- `obj (object)` -- [Required] Das Secret oder das Objekt, das das Secret enthält.

Rückgabotyp

`object`

@InjectSecretString

Dieser Dekorator erwartet eine Secret-ID-Zeichenfolge und [the section called "SecretCache"](#) als erstes und zweites Argument. Der Dekorator gibt den Secret-Zeichenfolgewert zurück. Das Secret muss eine Zeichenfolge enthalten.

```
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

Dieser Dekorator erwartet eine Secret-ID-Zeichenfolge und [the section called "SecretCache"](#) als erstes und zweites Argument. Die verbleibenden Argumente ordnen die Parameter der umschlossenen Funktion den JSON-Schlüsseln im Secret zu. Das Secret muss eine Zeichenfolge in der JSON-Struktur enthalten.

Für ein Secret, das dieses JSON enthält:

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

Das folgende Beispiel zeigt, wie Sie die JSON-Werte für username und password aus dem Secret extrahieren.

```
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')
```

Rufen Sie mit dem AWS Python-SDK einen geheimen Wert von Secrets Manager ab

In Anwendungen können Sie Ihre Geheimnisse abrufen, indem Sie eines der SDKs aufrufen `GetSecretValue` oder `BatchGetSecretValue` in einem der AWS SDKs aufrufen. Wir empfehlen jedoch, Ihre Secret-Werte mithilfe des clientseitigen Cachings zu speichern. Das Caching von Secrets verbessert die Geschwindigkeit und senkt Ihre Kosten.

Verwenden Sie für Python-Anwendungen die [Secrets-Manager-Python-basierte Caching-Komponente](#) oder rufen Sie das SDK direkt mit [get_secret_value](#) oder [batch_get_secret_value](#) auf.

Die folgenden Codebeispiele zeigen die Verwendung `GetSecretValue`.

Erforderliche Berechtigungen: `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
```

Holen Sie sich mit dem AWS Python-SDK einen Stapel geheimer Werte von Secrets Manager

Das folgende Codebeispiel zeigt, wie Sie einen Stapel geheimer Werte von Secrets Manager abrufen.

Erforderliche Berechtigungen:

- `secretsmanager:BatchGetSecretValue`
- `secretsmanager:GetSecretValue` Erlaubnis für jedes Geheimnis, das Sie abrufen möchten.
- Wenn Sie Filter verwenden, müssen Sie auch `secretsmanager:ListSecrets` haben.

Ein Beispiel für eine Berechtigungsrichtlinie finden Sie unter [the section called “Beispiel: Berechtigung zum Abrufen einer Gruppe geheimer Werte in einem Batch”](#).

Important

Wenn Sie über eine VPCE-Richtlinie verfügen, die die Berechtigung zum Abrufen eines einzelnen Secrets in der abgerufenen Gruppe verweigert, gibt `BatchGetSecretValue` keine Secrets-Werte zurück, und es wird ein Fehler zurückgegeben.

```
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
```

Rufen Sie mithilfe von .NET einen geheimen Wert von Secrets Manager ab

In Anwendungen können Sie Ihre Geheimnisse abrufen, indem Sie eines der SDKs aufrufen `GetSecretValue` oder `BatchGetSecretValue` in einem der AWS SDKs aufrufen. Wir empfehlen jedoch, Ihre Secret-Werte mithilfe des clientseitigen Caching zu speichern. Das Caching von Secrets verbessert die Geschwindigkeit und senkt Ihre Kosten.

Themen

- [Rufen Sie mithilfe von .NET mit clientseitigem Caching einen geheimen Wert von Secrets Manager ab](#)
- [Rufen Sie mithilfe des .NET AWS SDK einen geheimen Wert von Secrets Manager ab](#)

Rufen Sie mithilfe von .NET mit clientseitigem Caching einen geheimen Wert von Secrets Manager ab

Wenn Sie ein Secret abrufen, können Sie die .NET-basierte Caching-Komponente von Secrets Manager verwenden, um es für zukünftige Verwendung zu cachieren. Das Abrufen eines gecacheten Secrets ist schneller als das Abrufen aus Secrets Manager. Da für den Aufruf von Secrets-Manager-APIs Kosten anfallen, kann die Verwendung eines Caches Ihre Kosten senken. Alle Möglichkeiten, wie Sie Secrets abrufen können, finden Sie unter [Holen Sie sich Geheimnisse](#).

Die Cache-Richtlinie ist Least Recently Used (LRU). Wenn der Cache also ein Secret verwerfen muss, verwirft er das am wenigsten verwendete Secret. Standardmäßig aktualisiert der Cache Secrets jede Stunde. Sie können konfigurieren, [wie oft das Secret im Cache aktualisiert wird](#), und Sie können [den Secret-Abruf anbinden](#), um weitere Funktionalität hinzuzufügen.

Der Cache erzwingt keine Garbage Collection, sobald Cache-Referenzen freigegeben wurden. Die Cache-Implementierung beinhaltet keine Cache-Invalidierung. Die Cache-Implementierung konzentriert sich auf den Cache selbst und ist weder sicherheitsgehärtet noch fokussiert. Wenn Sie zusätzliche Sicherheit benötigen, z. B. das Verschlüsseln von Elementen im Cache, verwenden Sie die bereitgestellten Schnittstellen und abstrakten Methoden.

Zum Verwenden der Komponente ist Folgendes erforderlich:

- .NET Framework ab Version 4.6.2 oder .NET Standard ab Version 2.0. Siehe [Download von .NET](#) auf der Microsoft-Website.
- Das AWS SDK for .NET. Siehe [the section called "AWS SDKs"](#).

Informationen zum Herunterladen des Quellcodes finden Sie unter [Caching-Client für .NET](#) auf GitHub.

Um den Cache zu verwenden, instanziiieren Sie ihn zuerst und rufen Sie dann Ihr Secret mit `GetSecretString` oder `GetSecretBinary` auf. Bei aufeinanderfolgenden Abrufen gibt der Cache die gecachte Kopie des Secrets zurück.

Tun Sie Folgendes, um das Caching-Paket zu erhalten

- Führen Sie eine der folgenden Aktionen aus:
 - Führen Sie den folgenden .NET-CLI-Befehl in Ihrem Projektverzeichnis aus.

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

- Fügen Sie die folgende Paketreferenz zu Ihrer `.csproj`-Datei hinzu.

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

Erforderliche Berechtigungen:

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

Weitere Informationen finden Sie unter [Berechtigungsreferenz](#).

Referenz

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

Example Ein Secret abrufen

Das folgende Codebeispiel zeigt eine Methode, die ein Geheimnis mit dem Namen abrufen. *MySecret*

```
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 Konfigurieren der Aktualisierungsdauer von Time to Live (TTL)-Caches

Das folgende Codebeispiel zeigt eine Methode, die ein Geheimnis mit dem Namen abrufen *MySecret* und die Aktualisierungsdauer des TTL-Caches auf 24 Stunden festlegt.

```
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

Ein In-Memory-Cache für von Secrets Manager angeforderte Secrets. Sie verwenden [the section called “GetSecretString”](#) oder [the section called “GetSecretBinary”](#), um ein Secret aus dem Cache abzurufen. Sie können die Cache-Einstellungen konfigurieren, indem Sie ein [the section called “SecretCacheConfiguration”](#)-Objekt im Konstruktor übergeben.

Weitere Informationen hierzu einschließlich Beispielen finden Sie unter [the section called “.NET mit clientseitigem Caching”](#).

Konstruktoren

```
public SecretsManagerCache()
```

Standardkonstruktor für ein `SecretsManagerCache`-Objekt.

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

Konstruiert einen neuen Cache mit einem Secrets Manager Manager-Client, der mit dem bereitgestellten [AmazonSecretsManagerClient](#) erstellt wurde. Verwenden Sie diesen Konstruktor, um den Secrets-Manager-Manager-Client anzupassen, z. B. um eine bestimmte Region oder einen bestimmten Endpunkt zu verwenden.

Parameter

`secretsManager`

Der [AmazonSecretsManagerClient](#), aus dem Geheimmnisse abgerufen werden sollen.

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

Konstruiert einen neuen Secret-Cache mit dem bereitgestellten [the section called "SecretCacheConfiguration"](#). Verwenden Sie diesen Konstruktor, um den Cache zu konfigurieren, z. B. die Anzahl der zu cachenden Secrets und wie oft er aktualisiert wird.

Parameter

`config`

Eine [the section called "SecretCacheConfiguration"](#), die Konfigurationsinformationen für den Cache enthält.

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

Konstruiert einen neuen Cache mit einem Secrets Manager Manager-Client, der mit dem bereitgestellten [AmazonSecretsManagerClient](#) und einem [the section called "SecretCacheConfiguration"](#) erstellt wurde. Verwenden Sie diesen Konstruktor, um den Secrets-Manager-Manager-Client anzupassen, z. B. um eine bestimmte Region oder einen bestimmten Endpunkt zu verwenden und den Cache zu konfigurieren, z. B. die Anzahl der zu cachenden Secrets und wie oft er aktualisiert wird.

Parameter

secretsManager

Der [AmazonSecretsManagerClient](#), aus dem Geheimnisse abgerufen werden sollen.

config

Eine [the section called "SecretCacheConfiguration"](#), die Konfigurationsinformationen für den Cache enthält.

Methoden

GetSecretString

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

Ruft ein String-Secret von Secrets Manager ab.

Parameter

secretId

Der ARN oder Name des abzurufenden Secrets.

GetSecretBinary

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

Ruft ein binäres Secret von Secrets Manager ab.

Parameter

secretId

Der ARN oder Name des abzurufenden Secrets.

RefreshNowAsync

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

Fordert den Secret-Wert von Secrets Manager an und aktualisiert den Cache mit allen Änderungen. Wenn kein Cache-Eintrag vorhanden ist, wird ein neuer erstellt. Gibt `true` zurück, wenn die Aktualisierung erfolgreich ist.

Parameter

secretId

Der ARN oder Name des abzurufenden Secrets.

GetCachedSecret

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

Gibt den Cache-Eintrag für das angegebene Secrets zurück, falls er im Cache vorhanden ist. Andernfalls wird das Secret aus Secrets Manager abgerufen und ein neuer Cache-Eintrag erstellt.

Parameter

secretId

Der ARN oder Name des abzurufenden Secrets.

SecretCacheConfiguration

Cache-Konfigurationsoptionen für einen [the section called "SecretsManagerCache"](#), z. B. maximale Cachegröße und Time to Live (TTL) für gecachete Secrets.

Eigenschaften

CacheItemTTL

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

Die TTL eines Cache-Elements in Millisekunden. Die Standardeinstellung ist 3600000 ms oder 1 Stunde. Das Maximum ist 4294967295 ms, was ungefähr 49,7 Tagen entspricht.

MaxCacheSize

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

Die maximale Cachegröße. Der Standardwert ist 1 024 Secrets. Der Höchstwert ist 65,535.

VersionStage

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

Die Version der Secrets, das Sie cachieren möchten. Weitere Informationen hierzu finden Sie unter [Secret-Versionen](#). Der Standardwert ist "AWSCURRENT".

Client

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

Das [AmazonSecretsManagerClient](#), aus dem Geheimnisse abgerufen werden sollen. Wenn er null ist, instanziiert der Cache einen neuen Client. Der Standardwert ist null.

CacheHook

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

Ein [the section called "Ich SecretCacheHook"](#)

Ich SecretCacheHook

Eine Schnittstelle für das Anbinden eines [the section called "SecretsManagerCache"](#), um Aktionen mit dem im Cache gespeicherten Secret durchzuführen.

Methoden

Put

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

Bereitet das Objekt für das Speichern im Cache vor.

Gibt das Objekt zurück, das im Cache gespeichert werden soll.

Get

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

Leitet das Objekt aus dem gecachten Objekt ab.

Gibt das Objekt zurück, das vom Cache zurückgegeben werden soll.

Rufen Sie mithilfe des .NET AWS SDK einen geheimen Wert von Secrets Manager ab

In Anwendungen können Sie Ihre Geheimnisse abrufen, indem Sie eines der SDKs aufrufen `GetSecretValue` oder `BatchGetSecretValue` in einem der AWS SDKs aufrufen. Wir empfehlen

jedoch, Ihre Secret-Werte mithilfe des clientseitigen Cachings zu speichern. Das Caching von Secrets verbessert die Geschwindigkeit und senkt Ihre Kosten.

Verwenden Sie für .NET-Anwendungen die [Secrets-Manager-.NET-basierte Caching-Komponente](#) oder rufen Sie das SDK direkt mit [GetSecretValue](#) oder [BatchGetSecretValue](#) auf.

Die folgenden Codebeispiele zeigen die Verwendung `GetSecretValue`.

Erforderliche Berechtigungen: `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;
    }
}
}
```

Holen Sie sich mit Go einen geheimen Wert für Secrets Manager

In Anwendungen können Sie Ihre Geheimnisse abrufen, indem Sie eines der SDKs aufrufen `GetSecretValue` oder `BatchGetSecretValue` in einem der AWS SDKs aufrufen. Wir empfehlen jedoch, Ihre Secret-Werte mithilfe des clientseitigen Caching zu speichern. Das Caching von Secrets verbessert die Geschwindigkeit und senkt Ihre Kosten.

Themen

- [Rufen Sie mithilfe von Go mit clientseitigem Caching einen geheimen Wert für Secrets Manager ab](#)

- [Holen Sie sich mit dem Go AWS SDK einen geheimen Wert für Secrets Manager](#)

Rufen Sie mithilfe von Go mit clientseitigem Caching einen geheimen Wert für Secrets Manager ab

Wenn Sie ein Secret abrufen, können Sie die Go-basierte Caching-Komponente von Secrets Manager verwenden, um es für zukünftige Verwendung zu cachieren. Das Abrufen eines gecacheten Secrets ist schneller als das Abrufen aus Secrets Manager. Da für den Aufruf von Secrets-Manager-APIs Kosten anfallen, kann die Verwendung eines Caches Ihre Kosten senken. Alle Möglichkeiten, wie Sie Secrets abrufen können, finden Sie unter [Holen Sie sich Geheimnisse](#).

Die Cache-Richtlinie ist Least Recently Used (LRU). Wenn der Cache also ein Secret verwerfen muss, verwirft er das am wenigsten verwendete Secret. Standardmäßig aktualisiert der Cache Secrets jede Stunde. Sie können konfigurieren, [wie oft das Secret im Cache aktualisiert wird](#), und Sie können [den Secret-Abruf anbinden](#), um weitere Funktionalität hinzuzufügen.

Der Cache erzwingt keine Garbage Collection, sobald Cache-Referenzen freigegeben wurden. Die Cache-Implementierung beinhaltet keine Cache-Invalidierung. Die Cache-Implementierung konzentriert sich auf den Cache selbst und ist weder sicherheitsgehärtet noch fokussiert. Wenn Sie zusätzliche Sicherheit benötigen, z. B. das Verschlüsseln von Elementen im Cache, verwenden Sie die bereitgestellten Schnittstellen und abstrakten Methoden.

Zum Verwenden der Komponente ist Folgendes erforderlich:

- AWS SDK for Go. Siehe [the section called "AWS SDKs"](#).

Um den Quellcode herunterzuladen, siehe [Secrets Manager Go Caching-Client](#) auf GitHub.

Informationen zum Einrichten einer Go-Entwicklungsumgebung finden Sie unter [Golang – Erste Schritte](#) auf der Website der Go-Programmiersprache.

Erforderliche Berechtigungen:

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

Weitere Informationen finden Sie unter [Berechtigungsreferenz](#).

Referenz

- [type Cache](#)
- [Typ CacheConfig](#)
- [Typ CacheHook](#)

Example Ein Secret abrufen

Das folgende Codebeispiel zeigt eine Lambda-Funktion, die ein Secret abruft.

```
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

Ein In-Memory-Cache für von Secrets Manager angeforderte Secrets. Sie verwenden [the section called "GetSecretString"](#) oder [the section called "GetSecretBinary"](#), um ein Secret aus dem Cache abzurufen.

Im folgenden Beispiel wird veranschaulicht, wie Sie die Cache-Einstellungen konfigurieren.

```
// 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 },
)
```

Weitere Informationen hierzu einschließlich Beispielen finden Sie unter [the section called “Entscheiden Sie sich für clientseitiges Caching”](#).

Methoden

Neu

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

„Neu“ konstruiert einen Secret-Cache mit Funktionsoptionen, verwendet ansonsten Standardwerte. Initialisiert einen SecretsManager Client aus einer neuen Sitzung. Initialisiert CacheConfig auf Standardwerte. Initialisiert LRU-Cache mit einer maximalen Standardgröße.

GetSecretString

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

GetSecretString ruft den geheimen Zeichenkettenwert aus dem Cache für die angegebene geheime ID ab. Gibt die Secret-Zeichenfolge und einen Fehler zurück, wenn der Vorgang fehlgeschlagen ist.

GetSecretStringWithStage

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

GetSecretStringWithStage ruft den geheimen Zeichenkettenwert aus dem Cache für die angegebene geheime ID und [Versionsstufe](#) ab. Gibt die Secret-Zeichenfolge und einen Fehler zurück, wenn der Vorgang fehlgeschlagen ist.

GetSecretBinary

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

GetSecretBinary ruft den geheimen Binärwert aus dem Cache für die angegebene geheime ID ab. Gibt den Secret-Binärwert und einen Fehler zurück, wenn der Vorgang fehlgeschlagen ist.

GetSecretBinaryWithStage

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

GetSecretBinaryWithStage ruft den geheimen Binärwert aus dem Cache für die angegebene geheime ID und [Versionsstufe](#) ab. Gibt den Secret-Binärwert und einen Fehler zurück, wenn der Vorgang fehlgeschlagen ist.

Typ CacheConfig

Cache-Konfigurationsoptionen für ein [Cache](#), z. B. maximale Cachegröße, Standard-[Versionsstufe](#) und Time to Live (TTL) für gecachte Secrets.

```
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  
}
```

Typ CacheHook

Eine Schnittstelle für das Anbinden eines [Cache](#), um Aktionen mit dem im Cache gespeicherten Secret durchzuführen.

Methoden

Put

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

Bereitet das Objekt für das Speichern im Cache vor.

Get

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

Leitet das Objekt aus dem gecacheten Objekt ab.

Holen Sie sich mit dem Go AWS SDK einen geheimen Wert für Secrets Manager

In Anwendungen können Sie Ihre Geheimnisse abrufen, indem Sie sie aufrufen `GetSecretValue` oder `BatchGetSecretValue` in einem der AWS SDKs aufrufen. Wir empfehlen jedoch, Ihre Secret-Werte mithilfe des clientseitigen Cachings zu speichern. Das Caching von Secrets verbessert die Geschwindigkeit und senkt Ihre Kosten.

Verwenden Sie für Go-Anwendungen die [Secrets-Manager-Go-basierte Caching-Komponente](#) oder rufen Sie das SDK direkt mit [GetSecretValue](#) oder [BatchGetSecretValue](#) auf.

Das folgende Codebeispiel veranschaulicht, wie Sie einen Secrets-Manager-Geheimniswert abrufen.

Erforderliche Berechtigungen: `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.
}
```

Rufen Sie mit dem AWS C++-SDK einen geheimen Wert von Secrets Manager ab

Rufen Sie das SDK für C++-Anwendungen direkt mit [GetSecretValue](#) oder auf [BatchGetSecretValue](#).

Das folgende Codebeispiel veranschaulicht, wie Sie einen Secrets-Manager-Geheimniswert abrufen.

Erforderliche Berechtigungen: `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();
}

```

Rufen Sie mithilfe des JavaScript AWS SDK einen geheimen Wert von Secrets Manager ab

Rufen Sie das SDK für JavaScript Anwendungen direkt mit [getSecretValue](#) oder auf [batchGetSecretValue](#).

Das folgende Codebeispiel veranschaulicht, wie Sie einen Secrets-Manager-Geheimniswert abrufen.

Erforderliche Berechtigungen: `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;
  }
};
```

Holen Sie sich mit dem Kotlin AWS SDK einen geheimen Wert für Secrets Manager

Rufen Sie das SDK für Kotlin-Anwendungen direkt mit [GetSecretValue](#) oder auf [BatchGetSecretValue](#).

Das folgende Codebeispiel veranschaulicht, wie Sie einen Secrets-Manager-Geheimniswert abrufen.

Erforderliche Berechtigungen: `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")
    }
}
```

Holen Sie sich mit dem AWS PHP-SDK einen geheimen Wert für Secrets Manager

Rufen Sie für PHP-Anwendungen das SDK direkt mit [GetSecretValue](#) oder [BatchGetSecretValue](#) auf.

Das folgende Codebeispiel veranschaulicht, wie Sie einen Secrets-Manager-Geheimniswert abrufen.

Erforderliche Berechtigungen: `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
```

Holen Sie sich mit dem Ruby AWS SDK einen geheimen Wert für Secrets Manager

Rufen Sie für Ruby-Anwendungen das SDK direkt mit [get_secret_value](#) oder [batch_get_secret_value](#) auf.

Das folgende Codebeispiel veranschaulicht, wie Sie einen Secrets-Manager-Geheimniswert abrufen.

Erforderliche Berechtigungen: `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
```

Holen Sie sich mit dem Rust AWS SDK einen geheimen Wert für Secrets Manager

Rufen Sie das SDK für Rust-Anwendungen direkt mit [GetSecretValue](#) oder auf [BatchGetSecretValue](#).

Das folgende Codebeispiel veranschaulicht, wie Sie einen Secrets-Manager-Geheimniswert abrufen.

Erforderliche Berechtigungen: `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(())  
}
```

Holen Sie sich einen geheimen Wert mit dem AWS CLI

Erforderliche Berechtigungen: `secretsmanager:GetSecretValue`

Example Den verschlüsselten Secret-Wert eines Secrets abrufen

Im folgenden [get-secret-value](#)-Beispiel wird der aktuelle Secret-Wert abgerufen.

```
aws secretsmanager get-secret-value \  
  --secret-id MyTestSecret
```

Example Den vorherigen Secret-Wert abrufen

Im folgenden [get-secret-value](#)-Beispiel wird der vorherige Secret-Wert abgerufen.

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

Holen Sie sich eine Gruppe von Geheimnissen in einem Batch mit dem AWS CLI

Erforderliche Berechtigungen:

- `secretsmanager:BatchGetSecretValue`
- `secretsmanager:GetSecretValue` Erlaubnis für jedes Geheimnis, das Sie abrufen möchten.
- Wenn Sie Filter verwenden, müssen Sie auch `secretsmanager:ListSecrets` haben.

Ein Beispiel für eine Berechtigungsrichtlinie finden Sie unter [the section called “Beispiel: Berechtigung zum Abrufen einer Gruppe geheimer Werte in einem Batch”](#).

⚠ Important

Wenn Sie über eine VPCE-Richtlinie verfügen, die die Berechtigung zum Abrufen eines einzelnen Secrets in der abgerufenen Gruppe verweigert, gibt `BatchGetSecretValue` keine Secrets-Werte zurück, und es wird ein Fehler zurückgegeben.

Example Ruft den Secrets-Wert für eine Gruppe von Secrets ab, die nach Namen aufgelistet sind

Im folgenden [batch-get-secret-value](#)-Beispiel wird der aktuelle Secret-Wert für drei Secrets abgerufen.

```
aws secretsmanager batch-get-secret-value \  
    --secret-id-list MySecret1 MySecret2 MySecret3
```

Example Ruft den Secrets-Wert für eine Gruppe von Secrets ab, die durch Filter ausgewählt sind

Im folgenden [batch-get-secret-value](#)-Beispiel wird der Secret-Wert für die Secrets abgerufen, die über ein Tag mit dem Namen „Test“ verfügen.

```
aws secretsmanager batch-get-secret-value \  
    --filters Key="tag-key",Values="Test"
```

Rufen Sie mithilfe der AWS Konsole einen geheimen Wert ab

Ein Secret abrufen (Konsole)

1. Öffnen Sie die Secrets-Manager-Konsole unter <https://console.aws.amazon.com/secretsmanager/>.
2. Wählen Sie in der Secret-Liste das Secret aus, das Sie abrufen möchten.
3. Wählen Sie im Bereich Secret value (Secret-Wert) die Option Retrieve secret value (Secret-Wert abrufen).

Secrets Manager zeigt die aktuelle Version (AWSCURRENT) des Secrets an. Um [andere Versionen](#) des Secrets anzuzeigen, z. B. AWSPREVIOUS oder Versionen mit benutzerdefinierter Bezeichnung, verwenden Sie [the section called "AWS CLI"](#).

Benutze AWS Secrets Manager Geheimnisse in AWS Batch

AWS Batch hilft Ihnen beim Ausführen von Batch-Computing-Workloads auf dem AWS Cloud. Mit können Sie vertrauliche Daten in Ihre Jobs einfügen AWS Batch, indem Sie Ihre sensiblen Daten geheim speichern und dann in AWS Secrets Manager Ihrer Jobdefinition darauf verweisen. Weitere Informationen finden Sie unter [Angeben sensibler Daten mit Secrets Manager](#).

Holen Sie sich ein AWS Secrets Manager Geheimnis in einer AWS CloudFormation Ressource

Mit können Sie ein Geheimnis abrufen AWS CloudFormation, um es in einer anderen AWS CloudFormation Ressource zu verwenden. Ein häufiges Szenario besteht darin, zuerst ein Secret mit einem von Secrets Manager generierten Passwort zu erstellen und dann den Benutzernamen und das Passwort aus dem Secret abzurufen, um sie als Anmeldeinformationen für eine neue Datenbank zu verwenden. Hinweise zum Erstellen von Geheimnissen mit AWS CloudFormation finden Sie unter [AWS CloudFormation](#).

Um ein Geheimnis in einer AWS CloudFormation Vorlage abzurufen, verwenden Sie eine dynamische Referenz. Wenn Sie den Stack erstellen, überträgt die dynamische Referenz den geheimen Wert in die AWS CloudFormation Ressource, sodass Sie die geheimen Informationen nicht fest codieren müssen. Sie können das Secret stattdessen anhand des Namens oder des ARN referenzieren. Sie können eine dynamische Referenz für ein Secret in jeder Ressourceneigenschaft verwenden. Sie können keine dynamische Referenz für ein Secret in Ressourcenmetadaten verwenden, z. B. [AWS::CloudFormation::Init](#), weil dadurch der Secret-Wert in der Konsole sichtbar würde.

Eine dynamische Referenz für ein Secret hat das folgende Muster:

```
{{resolve:secretsmanager:secret-id:SecretString:json-key:version-stage:version-id}}
```

`secret-id`

Der Name oder ARN des Secrets. Um auf ein Geheimnis in Ihrem AWS Konto zuzugreifen, können Sie den geheimen Namen verwenden. Um auf ein Geheimnis in einem anderen AWS Konto zuzugreifen, verwenden Sie den ARN des Geheimnisses.

json-key (Optional)

Der Schlüsselname des Schlüssel/Wert-Paares, dessen Wert Sie abrufen möchten. Wenn Sie kein `json-key` angeben, wird der gesamte geheime Text AWS CloudFormation abgerufen. Dieses Segment darf nicht das Doppelpunktzeichen (:) enthalten.

version-stage (Optional)

Die [Version](#) des zu verwendenden Secrets. Secrets Manager verwendet Staging-Markierungen, um während des Rotationsprozesses den Überblick über verschiedene Versionen zu behalten. Wenn Sie `version-stage` verwenden, geben Sie `version-id` nicht an. Wenn Sie weder `version-stage` noch `version-id` angeben, dann ist der Standard die `AWSCURRENT`-Version. Dieses Segment darf nicht das Doppelpunktzeichen (:) enthalten.

version-id (Optional)

Die eindeutige ID der Version des zu verwendenden Secrets. Wenn Sie `version-id` angeben, dürfen Sie `version-stage` nicht angeben. Wenn Sie weder `version-stage` noch `version-id` angeben, dann ist der Standard die `AWSCURRENT`-Version. Dieses Segment darf nicht das Doppelpunktzeichen (:) enthalten.

Weitere Informationen finden Sie unter [Verwenden dynamischer Referenzen, um Secrets-Manager-Geheimnisse anzugeben](#).

Note

Erstellen Sie keinen dynamischen Verweis mit einem umgekehrten Schrägstrich (\) als Endwert. AWS CloudFormation kann diese Verweise nicht auflösen, was zu einem Ressourcenausfall führt.

Verwenden Sie AWS Secrets Manager Geheimnisse in Amazon Elastic Kubernetes Service

Um Secrets aus Secrets Manager als in [Amazon EKS-Pods](#) gemountete Dateien anzuzeigen, können Sie den AWS Secrets and Configuration Provider (ASCP) für den [Kubernetes Secrets Store](#) CSI-Treiber verwenden. Das ASCP funktioniert mit Amazon Elastic Kubernetes Service (Amazon EKS) 1.17+, auf dem eine Amazon EC2 EC2-Knotengruppe ausgeführt wird. AWS Fargate Knotengruppen

werden nicht unterstützt. Mit dem ASCP können Sie Ihre Secrets in Secrets Manager speichern und verwalten und diese dann über Ihre auf Amazon EKS ausgeführten Workloads abrufen. Wenn Ihr Secret mehrere Schlüssel/Wert-Paare im JSON-Format enthält, können Sie auswählen, welche in Amazon EKS bereitgestellt werden sollen. Der ASCP verwendet [JMESPath-Syntax](#), um die Schlüssel/Wert-Paare in Ihrem Secret abzufragen. Der ASCP funktioniert auch mit [Parametern des Parameter-Speichers](#).

Wenn Sie einen privaten Amazon-EKS-Cluster verwenden, stellen Sie sicher, dass die VPC, in der sich der Cluster befindet, über einen Secrets-Manager-Endpunkt verfügt. Der Secrets-Store-CSI-Treiber verwendet den Endpunkt, um Aufrufe an Secrets Manager vorzunehmen. Informationen zum Erstellen eines Endpunkts in einer VPC finden Sie unter [VPC-Endpunkt](#).

Wenn Sie in Secrets Manager die automatische Rotation für Ihre Secrets verwenden, können Sie auch das Abstimmungsfeature für die Secrets-Store-CSI-Treiberrotation verwenden, um sicherzustellen, dass Sie das neueste Secret aus Secrets Manager abrufen. Weitere Informationen finden Sie unter [Automatische Rotation bereitgestellter Inhalte und synchronisierter Kubernetes Secrets](#).

Themen

- [Schritt 1: Einrichten der Zugriffssteuerung](#)
- [Schritt 2: Installieren und konfigurieren Sie den ASCP](#)
- [Schritt 3: Identifizieren Sie, welche Secrets bereitgestellt werden sollen](#)
- [Schritt 4: Mounten Sie die Geheimnisse als Dateien im Amazon EKS-Pod](#)
- [Fehlerbehebung](#)
- [SecretProviderClass](#)

Schritt 1: Einrichten der Zugriffssteuerung

Das ASCP ruft die Amazon EKS-Pod-Identität ab und tauscht sie gegen eine IAM-Rolle aus. Sie legen Berechtigungen in einer IAM-Richtlinie für diese IAM-Rolle fest. Wenn der ASCP die IAM-Rolle übernimmt, erhält er Zugriff auf die von Ihnen autorisierten Geheimnisse. Andere Container können nur auf die Secrets zugreifen, wenn Sie diese auch der IAM-Rolle zuordnen.

Wenn Aufrufe von ASCP zur Suche nach der Region und der IAM-Rolle, die dem Pod zugeordnet sind, von Kubernetes gedrosselt werden, können Sie die Drosselungsquoten wie in Schritt 2 gezeigt wie in Schritt 2 gezeigt ändern. `helm install`

So gewähren Sie Ihrem Amazon EKS-Pod Zugriff auf Secrets Manager

1. Erstellen Sie eine Berechtigungsrichtlinie, die Zugriff auf die Geheimnisse gewährt `secretsmanager:GetSecretValue`, auf die der Pod zugreifen muss. `secretsmanager:DescribeSecret` Eine Beispielrichtlinie finden Sie unter [the section called "Beispiel: Erlaubnis, einzelne Geheimnisse zu lesen und zu beschreiben"](#).
2. Erstellen Sie einen IAM OpenID Connect (OIDC)-Anbieter für den Cluster, wenn Sie noch keinen haben. Weitere Informationen finden Sie unter [Erstellen eines IAM-OIDC-Anbieters für Ihren Cluster](#) im Amazon EKS-Benutzerhandbuch.
3. Erstellen Sie eine [IAM-Rolle für ein Servicekonto](#) und fügen Sie die Richtlinie hinzu. Weitere Informationen finden Sie unter [Erstellen einer IAM-Rolle für ein Servicekonto](#) im Amazon EKS-Benutzerhandbuch.
4. Wenn Sie einen privaten Amazon EKS-Cluster verwenden, stellen Sie sicher, dass die VPC, in der sich der Cluster befindet, über einen AWS STS Endpunkt verfügt. Informationen zum Erstellen eines Endpunkts finden Sie unter [Interface VPC Endpoints](#) im AWS Identity and Access Management Benutzerhandbuch.

Schritt 2: Installieren und konfigurieren Sie den ASCP

Das ASCP ist GitHub im [secrets-store-csi-provider-aws-Repository](#) verfügbar. Das Repo enthält auch YAML-Beispieldateien zum Erstellen und Mounten eines Secrets.

Während der Installation können Sie das ASCP so konfigurieren, dass es einen FIPS-Endpunkt verwendet. Eine Liste der Endpunkte finden Sie unter [the section called "Secrets-Manager-Endpunkte"](#).

Um den ASCP mit Helm zu installieren

1. Um sicherzustellen, dass das Repo auf die neuesten Charts verweist, verwenden Sie `helm repo update`.
2. Fügen Sie das Secrets Store CSI-Treiberdiagramm hinzu.

```
helm repo add secrets-store-csi-driver https://kubernetes-sigs.github.io/secrets-store-csi-driver/charts
```

3. Installieren Sie das Diagramm. Um die Drosselung zu konfigurieren, fügen Sie das folgende Flag hinzu: `--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. Fügen Sie das ASCP-Diagramm hinzu.

```
helm repo add aws-secrets-manager https://aws.github.io/secrets-store-csi-driver-provider-aws
```

5. Installieren Sie das Diagramm. Um einen FIPS-Endpoint zu verwenden, fügen Sie das folgende Flag hinzu: `--set useFipsEndpoint=true`

```
helm install -n kube-system secrets-provider-aws aws-secrets-manager/secrets-store-csi-driver-provider-aws
```

Zur Installation mithilfe von YAML im Repo

- Verwenden Sie die folgenden Befehle.

```
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
```

Schritt 3: Identifizieren Sie, welche Secrets bereitgestellt werden sollen

Um zu bestimmen, welche Secrets der ASCP in Amazon EKS als Dateien im Dateisystem bereitstellt, erstellen Sie eine [the section called “SecretProviderClass”](#)-YAML-Datei. Die `SecretProviderClass` listet die Secrets auf, die gemountet werden sollen, und den Dateinamen, unter dem sie gemountet werden sollen. `SecretProviderClass` muss sich im gleichen Namespace wie der Amazon-EKS-Pod befinden, auf den verwiesen wird.

Die folgenden Beispiele zeigen, wie Sie `SecretProviderClass` verwenden, um die Secrets, die Sie mounten möchten, und die Benennung der im Amazon-EKS-Pod gemounteten Dateien zu beschreiben.

Beispiele:

- [Beispiel: Secrets nach Namen oder ARN mounten](#)
- [Beispiel: Schlüssel-/Wert-Paare aus einem Secret mounten](#)
- [Beispiel: Definieren einer Failover-Region für ein multiregionales Secret](#)
- [Beispiel: Ein Failover-Secret zum Mounten auswählen](#)

Beispiel: Secrets nach Namen oder ARN mounten

Das folgende Beispiel zeigt ein `SecretProviderClass`, das drei Dateien in Amazon EKS mountet:

1. Ein Secret, das durch den vollständigen ARN angegeben wird.
2. Ein durch den Namen spezifiziertes Secret.
3. Eine bestimmte Version eines Secrets.

```
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"
```

Beispiel: Schlüssel-/Wert-Paare aus einem Secret mounten

Das folgende Beispiel zeigt ein `SecretProviderClass`, das drei Dateien in Amazon EKS mountet:

1. Ein Secret, das durch den vollständigen ARN angegeben wird.
2. Das `username`-Schlüssel/Wert-Paar aus demselben Secret.
3. Das `password`-Schlüssel/Wert-Paar aus demselben Secret.

```
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
```

Beispiel: Definieren einer Failover-Region für ein multiregionales Secret

Um die Verfügbarkeit bei Verbindungsausfällen oder für Notfallwiederherstellungskonfigurationen zu gewährleisten, unterstützt der ASCP ein automatisches Failover-Feature zum Abrufen von Secrets aus einer sekundären Region.

Das folgende Beispiel zeigt ein `SecretProviderClass`, das ein Secret abrufen, das in mehrere Regionen repliziert wird. In diesem Beispiel versucht der ASCP, das Secret sowohl von `us-east-1` als auch `us-east-2` abzurufen. Wenn eine Region einen 4xx-Fehler zurückgibt, z. B. aufgrund eines Authentifizierungsproblems, mountet der ASCP keines der Secrets. Wenn das Secret erfolgreich von `us-east-1` abgerufen wurde, mountet der ASCP diesen Secret-Wert. Wenn das Secret nicht erfolgreich von `us-east-1` abgerufen wurde, aber erfolgreich von `us-east-2` abgerufen werden konnte, mountet der ASCP diesen Secret-Wert.

```
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"
```

Beispiel: Ein Failover-Secret zum Mounten auswählen

Das folgende Beispiel zeigt ein `SecretProviderClass`, das angibt, welches Secret im Falle eines Failovers gemountet werden soll. Das Failover-Secret ist kein Replikat. In diesem Beispiel versucht der ASCP, die zwei von `objectName` angegebenen Secrets abzurufen. Wenn eines von beiden einen 4xx-Fehler zurückgibt, z. B. aufgrund eines Authentifizierungsproblems, mountet der ASCP keines der Secrets. Wenn das Secret erfolgreich von `us-east-1` abgerufen wurde, mountet der ASCP diesen Secret-Wert. Wenn das Secret nicht erfolgreich von `us-east-1` abgerufen wurde, aber erfolgreich von `us-east-2` abgerufen werden konnte, mountet der ASCP diesen Secret-Wert. Die gemountete Datei in Amazon EKS heißt `MyMountedSecret`.

```
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"
```

Schritt 4: Mounten Sie die Geheimnisse als Dateien im Amazon EKS-Pod

So hängen Sie Geheimnisse in Amazon EKS ein

1. Wenden Sie das mit `SecretProviderClass` dem Befehl auf den Pod `ankubectl apply -f ExampleSecretProviderClass.yaml`.
2. Stellen Sie Ihren Pod mit dem Befehl `bereitkubectl apply -f ExampleDeployment.yaml`.
3. Das ASCP mountet die Dateien.

Fehlerbehebung

Sie können die meisten Fehler anzeigen, indem Sie die Pod-Bereitstellung beschreiben.

Fehlermeldungen für Ihren Container anzeigen

1. Erstellen Sie mit dem folgenden Befehl eine Liste der Pod-Namen. Wenn Sie nicht den Standard-Namespace verwenden, verwenden Sie `-n <NAMESPACE>`.

```
kubectl get pods
```

2. Um den Pod zu beschreiben, geben Sie im folgenden Befehl für `<PODID>` die Pod-ID aus den Pods an, die Sie im vorherigen Schritt gefunden haben. Wenn Sie nicht den Standard-Namespace verwenden, verwenden Sie `-n <NAMESPACE>`.

```
kubectl describe pod/<PODID>
```

Fehler für den ASCP anzeigen

- `<PODID>` Verwenden Sie im folgenden Befehl die ID des Pods `csi-secrets-store-provider-aws`, um weitere Informationen in den Anbieterprotokollen zu finden.

```
kubectl -n kube-system get pods  
kubectl -n kube-system logs pod/<PODID>
```

SecretProviderClass

Sie verwenden YAML, um zu beschreiben, welche Secrets [mithilfe von ASCP in Amazon EKS bereitgestellt werden sollen](#). Beispiele finden Sie unter [the section called “Secrets nach Namen oder ARN mounten”](#).

```
apiVersion: secrets-store.csi.x-k8s.io/v1  
kind: SecretProviderClass  
metadata:  
  name: <NAME>  
spec:  
  provider: aws  
  parameters:
```

```
region:  
failoverRegion:  
pathTranslation:  
objects:
```

Das Feld `parameters` enthält die Details der Mounting-Anfrage:

Region

(Optional) Der Name des AWS-Region Geheimnisses. Wenn Sie dieses Feld nicht verwenden, sucht der ASCP die Region aus der Anmerkung auf dem Knoten. Diese Suche steigert den Overhead von Mounting-Anfragen. Daher wird empfohlen, die Region für Cluster mit einer großen Anzahl von Pods anzugeben.

Wenn Sie auch `failoverRegion` angeben, versucht der ASCP, das Secret aus beiden Regionen abzurufen. Wenn eine Region einen 4xx-Fehler zurückgibt, z. B. aufgrund eines Authentifizierungsproblems, mountet der ASCP keines der Secrets. Wenn das Secret erfolgreich von `region` abgerufen wurde, mountet der ASCP diesen Secret-Wert. Wenn das Secret nicht erfolgreich von `region` abgerufen wurde, aber erfolgreich von `failoverRegion` abgerufen werden konnte, mountet der ASCP diesen Secret-Wert.

failoverRegion

(Optional) Wenn Sie dieses Feld angeben, versucht der ASCP, das Secret aus den Regionen abzurufen, die in `region` und diesem Feld definiert sind. Wenn eine Region einen 4xx-Fehler zurückgibt, z. B. aufgrund eines Authentifizierungsproblems, mountet der ASCP keines der Secrets. Wenn das Secret erfolgreich von `region` abgerufen wurde, mountet der ASCP diesen Secret-Wert. Wenn das Secret nicht erfolgreich von `region` abgerufen wurde, aber erfolgreich von `failoverRegion` abgerufen werden konnte, mountet der ASCP diesen Secret-Wert. Ein Beispiel für die Nutzung dieses Felds finden Sie unter [Definieren einer Failover-Region für ein multiregionales Secret](#).

pathTranslation (Pfadangabe)

(Optional) Ein einzelnes Ersetzungszeichen, das verwendet werden soll, wenn der Dateiname in Amazon EKS das Pfadtrennzeichen enthält, z. B. Schrägstrich (/) unter Linux. ASCP kann keine gemountete Datei erstellen, die ein Pfadtrennzeichen enthält. Stattdessen ersetzt ASCP das Pfadtrennzeichen durch ein anderes Zeichen. Wenn Sie dieses Feld nicht verwenden, ist das Ersatzzeichen ein Unterstrich (_), d. h. `My/Path/Secret` wird als `My_Path_Secret` gemountet.

Um die Zeichenersetzung zu verhindern, geben Sie die Zeichenfolge `False` ein.

objects (Objekte)

Eine Zeichenfolge, die eine YAML-Deklaration der bereitzustellenden Secrets enthält. Wir empfehlen, eine mehrzeilige YAML-Zeichenfolge oder ein Pipe-Zeichen (|) zu verwenden.

objectName (Objektname)

Der Name oder vollständige ARN des Secrets. Wenn Sie den ARN verwenden, können Sie `objectType` weglassen. Dieses Feld wird der Dateiname des Secrets im Amazon-EKS-Pod, es sei denn, Sie geben `objectAlias` an. Wenn Sie einen ARN verwenden, muss die Region im ARN mit dem Feld `region` übereinstimmen. Wenn Sie eine `failoverRegion` angeben, stellt dieses Feld den primären `objectName` dar.

objectType

Erforderlich, wenn Sie keinen Secrets Manager ARN für `objectName` verwenden. Kann `secretsmanager` oder `ssmparameter` sein.

objectAlias (Objektalias)

(Optional) Der Dateiname des Secrets im Amazon-EKS-Pod. Wenn Sie dieses Feld nicht angeben, wird `objectName` als Dateiname angezeigt.

objectVersion (Objektversion)

(Optional) Die Versions-ID des Secrets. Nicht empfohlen, da Sie jedes Mal, wenn Sie das Secret aktualisieren, die Versions-ID aktualisieren müssen. Standardmäßig wird die neueste Version verwendet. Wenn Sie eine `failoverRegion` angeben, stellt dieses Feld den primären `objectVersion` dar.

objectVersionLabel

(Optional) Der Alias für die Version. Die Standardversion ist die neueste Version `AWSCURRENT`. Weitere Informationen finden Sie unter [the section called “Geheime Versionen”](#). Wenn Sie eine `failoverRegion` angeben, stellt dieses Feld den primären `objectVersionLabel` dar.

jmesPath (jmes-Pfad)

(Optional) Eine Zuordnung der Schlüssel im Secret zu den Dateien, die in Amazon EKS bereitgestellt werden sollen. Um dieses Feld zu verwenden, muss Ihr Secret-Wert im JSON-Format vorliegen. Wenn Sie dieses Feld verwenden, müssen Sie die Unterfelder `path` und `objectAlias` angeben.

path (Pfad)

Ein Schlüssel aus einem Schlüssel/Wert-Paar im JSON des Secret-Werts. Wenn das Feld einen Bindestrich enthält, verwenden Sie einfache Anführungszeichen als Escape-Zeichen.

Beispiel: path: '"hyphenated-path"'

objectAlias

Der Dateiname, der im Amazon-EKS-Pod bereitgestellt werden soll. Wenn das Feld einen Bindestrich enthält, verwenden Sie einfache Anführungszeichen als Escape-Zeichen.

Beispiel: objectAlias: '"hyphenated-alias"'

failoverObject

(Optional) Wenn Sie dieses Feld angeben, versucht der ASCP, sowohl das im primären objectName angegebene Secret als auch das im failoverObject-objectName-Unterfeld angegebene Secret abzurufen. Wenn eines von beiden einen 4xx-Fehler zurückgibt, z. B. aufgrund eines Authentifizierungsproblems, mountet der ASCP keines der Secrets. Wenn das Secret erfolgreich vom primären objectName abgerufen wurde, mountet der ASCP diesen Secret-Wert. Wenn das Secret nicht erfolgreich vom primären objectName abgerufen wurde, aber erfolgreich vom Failover-objectName abgerufen werden konnte, mountet der ASCP diesen Secret-Wert. Wenn Sie dieses Feld angeben, müssen Sie auch das Feld objectAlias angeben. Ein Beispiel für die Nutzung dieses Felds finden Sie unter [Ein Failover-Secret zum Mounten auswählen](#).

In der Regel verwenden Sie dieses Feld, wenn es sich bei dem Failover-Secret nicht um ein Replikat handelt. Ein Beispiel dazu, wie Sie ein Replikat angeben, finden Sie unter [Definieren einer Failover-Region für ein multiregionales Secret](#).

objectName (Objektname)

Der Name oder vollständige ARN des Failover-Secrets. Wenn Sie einen ARN verwenden, muss die Region im ARN mit dem Feld failoverRegion übereinstimmen.

objectVersion (Objektversion)

(Optional) Die Versions-ID des Secrets. Muss mit der primären objectVersion übereinstimmen. Nicht empfohlen, da Sie jedes Mal, wenn Sie das Secret aktualisieren, die Versions-ID aktualisieren müssen. Standardmäßig wird die neueste Version verwendet.

objectVersionLabel

(Optional) Der Alias für die Version. Die Standardversion ist die neueste Version AWSCURRENT. Weitere Informationen finden Sie unter [the section called “Geheime Versionen”](#).

Verwenden Sie AWS Secrets Manager Geheimnisse in GitHub Jobs

Um ein Geheimnis in einem GitHub Job zu verwenden, können Sie eine GitHub Aktion verwenden, um Geheimnisse abzurufen AWS Secrets Manager und sie als maskierte [Umgebungsvariablen](#) in Ihrem GitHub Workflow hinzuzufügen. Weitere Informationen zu GitHub Aktionen finden Sie in der GitHub Dokumentation unter [Grundlegendes zu GitHub Aktionen](#).

Wenn Sie Ihrer GitHub Umgebung ein Geheimnis hinzufügen, steht es für alle anderen Schritte Ihres GitHub Jobs zur Verfügung. Folgen Sie den Anweisungen unter [Sicherheitshärtung für GitHub Maßnahmen](#), um zu verhindern, dass geheime Daten in Ihrer Umgebung missbraucht werden.

Sie können die gesamte Zeichenfolge im Secret-Wert als Umgebungsvariablenwert festlegen, oder wenn die Zeichenfolge JSON ist, können Sie den JSON-Code analysieren, um einzelne Umgebungsvariablen für jedes JSON-Schlüsselwertpaar festzulegen. Wenn der Secret-Wert ein Binärwert ist, wandelt die Aktion ihn in eine Zeichenfolge um.

Um die aus Ihren Secrets erstellten Umgebungsvariablen anzuzeigen, aktivieren Sie die Debug-Protokollierung. Weitere Informationen finden Sie in der Dokumentation unter [Aktivieren der Debug-Protokollierung](#). GitHub

Informationen zur Verwendung der aus Ihren Geheimnissen erstellten Umgebungsvariablen finden Sie in der GitHub Dokumentation unter [Umgebungsvariablen](#).

Voraussetzungen

Um diese Aktion verwenden zu können, müssen Sie zunächst AWS Anmeldeinformationen konfigurieren und diese AWS-Region in Ihrer GitHub Umgebung einrichten, indem Sie den `configure-aws-credentials` Schritt ausführen. Folgen Sie den Anweisungen unter [Aktion „AWS Anmeldeinformationen für GitHub Aktionen konfigurieren“](#), um die Rolle direkt über den GitHub OIDC-Anbieter zu übernehmen. Auf diese Weise können Sie Anmeldeinformationen mit kurzer Lebensdauer verwenden und vermeiden, dass zusätzliche Zugriffsschlüssel außerhalb von Secrets Manager gespeichert werden.

Die IAM-Rolle, die die Aktion annimmt, muss über die folgenden Berechtigungen verfügen:

- `GetSecretValue` auf den Secrets, die Sie abrufen möchten.
- `ListSecrets` auf allen Secrets.
- (Optional)`Decrypt`, KMS key ob die Geheimnisse mit einem verschlüsselt sind. Kundenverwalteter Schlüssel

Weitere Informationen finden Sie unter [Authentifizierung und Zugriffskontrolle](#).

Verwendung

Um die Aktion zu verwenden, fügen Sie Ihrem Workflow einen Schritt hinzu, der die folgende Syntax verwendet.

```
- 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
```

Parameter

`secret-ids`

Geheime ARNS, Namen und Namenspräfixe.

Um den Namen der Umgebungsvariablen festzulegen, geben Sie ihn vor der Secret-ID ein, gefolgt von einem Komma. Beispielsweise erstellt `ENV_VAR_1, secretId` eine Umgebungsvariable namens `ENV_VAR_1` aus der Secret-`secretId`. Die Namen von Umgebungsvariablen können Buchstaben, Zahlen und den Unterstriche enthalten.

Um ein Präfix zu verwenden, geben Sie mindestens drei Zeichen gefolgt von einem Sternchen ein. Zum Beispiel entspricht `dev*` allen Secrets mit einem Namen, der mit `dev` beginnt. Die maximale Anzahl übereinstimmender Secrets, die abgerufen werden können, ist 100. Wenn Sie den Variablennamen festlegen und das Präfix mit mehreren Secrets übereinstimmt, schlägt die Aktion fehl.

name-transformation

Standardmäßig erstellt der Schritt jeden Umgebungsvariablenamen aus dem Secret-Namen, der so umgewandelt wird, dass er nur Großbuchstaben, Zahlen und Unterstriche enthält, sodass er nicht mit einer Zahl beginnt. Für die Buchstaben im Namen können Sie den Schritt so konfigurieren, dass Kleinbuchstaben zusammen verwendet werden `lowercase` oder dass die Groß- und Kleinschreibung der Buchstaben mit `none` nicht geändert wird. Der Standardwert ist `uppercase`.

parse-json-secrets

(Optional) Standardmäßig legt die Aktion den Wert der Umgebungsvariablen auf die gesamte JSON-Zeichenfolge im Secret-Wert fest. Stellen Sie `parse-json-secrets` auf `true`, um Umgebungsvariablen für jedes Schlüssel-Wert-Paar im JSON zu erstellen.

Beachten Sie, dass die Aktion doppelte Namenskonflikte aufweist, wenn JSON Schlüssel wie „name“ und „Name“ verwendet, bei denen die Groß-/Kleinschreibung beachtet wird. Setzen Sie in diesem Fall `parse-json-secrets` auf `false` und analysieren Sie den JSON-Secret-Wert separat.

Benennung von Umgebungsvariablen

Die durch die Aktion erstellten Umgebungsvariablen haben den gleichen Namen wie die Geheimnisse, aus denen sie stammen. Für Umgebungsvariablen gelten strengere Benennungsanforderungen als für Geheimnisse, sodass die Aktion geheime Namen so transformiert, dass sie diese Anforderungen erfüllen. Die Aktion wandelt beispielsweise Kleinbuchstaben in Großbuchstaben um. Wenn Sie den JSON-Code des Geheimnisses analysieren, enthält der Name der Umgebungsvariablen beispielsweise sowohl den geheimen Namen als auch den JSON-Schlüsselnamen. `MYSECRET_KEYNAME` Sie können die Aktion so konfigurieren, dass Kleinbuchstaben nicht umgewandelt werden.

Wenn zwei Umgebungsvariablen am Ende denselben Namen haben würden, schlägt die Aktion fehl. In diesem Fall müssen Sie die Namen, die Sie für die Umgebungsvariablen verwenden möchten, als Aliase angeben.

Beispiele dafür, wann die Namen in Konflikt geraten könnten:

- Ein Geheimnis mit dem Namen "MySecret" und ein Geheimnis mit dem Namen „mysecret“ würden beide zu Umgebungsvariablen mit dem Namen „MYSECRET“ werden.

- Ein Geheimnis namens „Secret_KeyName“ und ein von JSON analysiertes Geheimnis namens „Secret“ mit einem Schlüssel namens „keyname“ würden beide zu Umgebungsvariablen mit dem Namen „SECRET_KEYNAME“ werden.

Sie können den Namen der Umgebungsvariablen festlegen, indem Sie einen Alias angeben, wie im folgenden Beispiel gezeigt, wodurch eine Variable mit dem Namen erstellt wird. ENV_VAR_NAME

```
secret-ids: |
  ENV_VAR_NAME, secretId2
```

Leere Aliase

- Wenn Sie einen leeren Alias, gefolgt von einem Komma und dann der geheimen ID, festlegen `parse-json-secrets: true` und eingeben, benennt die Aktion die Umgebungsvariable genauso wie die analysierten JSON-Schlüssel. Die Variablennamen enthalten nicht den geheimen Namen.

Wenn das Geheimnis kein gültiges JSON enthält, erstellt die Aktion eine Umgebungsvariable und benennt sie genauso wie den geheimen Namen.

- Wenn Sie einen leeren Alias, gefolgt von einem Komma und der geheimen ID, festlegen `parse-json-secrets: false` und eingeben, benennt die Aktion die Umgebungsvariablen so, als ob Sie keinen Alias angegeben hätten.

Das folgende Beispiel zeigt einen leeren Alias.

```
,secret2
```

Beispiele

Example 1 Erhalten Sie Secrets nach Namen und ARN

Im folgenden Beispiel werden Umgebungsvariablen für Secrets erstellt, die durch Name und ARN identifiziert werden.

```
- 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

```

Erstellte Umgebungsvariablen:

```

EXAMPLESECRETNAME: secretValue1
TEST1: secretValue2
_0_TEST_SECRET: secretValue3
_PROD_EXAMPLE_SECRET: secretValue4
SECRET_ALIAS_1: secretValue5
SECRET_ALIAS_2: secretValue6
SECRET2: secretValue7

```

Example 2 Erhalten Sie alle Secrets, die mit einem Präfix beginnen

Im folgenden Beispiel werden Umgebungsvariablen für alle Secrets erstellt, deren Namen mit *beta* beginnen.

```

- name: Get Secret Names by Prefix
  uses: 2
  with:
    secret-ids: |
      beta*    # Retrieves all secrets that start with 'beta'

```

Erstellte Umgebungsvariablen:

```

BETASECRETNAME: secretValue1
BETATEST: secretValue2
BETA_NEWSECRET: secretValue3

```

Example 3 Analysieren Sie JSON im Secret

Im folgenden Beispiel werden Umgebungsvariablen erstellt, indem das JSON im Secret analysiert wird.

```

- 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
```

Das Secret test/secret hat den folgenden Secret-Wert.

```
{
  "api_user": "user",
  "api_key": "key",
  "config": {
    "active": "true"
  }
}
```

Das Secret secret2 hat den folgenden Secret-Wert.

```
{
  "myusername": "alejandro_rosalez",
  "mypassword": "EXAMPLE_PASSWORD"
}
```

Erstellte Umgebungsvariablen:

```
TEST_SECRET_API_USER: "user"
TEST_SECRET_API_KEY: "key"
TEST_SECRET_CONFIG_ACTIVE: "true"
MYUSERNAME: "alejandro_rosalez"
MYPASSWORD: "EXAMPLE_PASSWORD"
```

Example 4 Verwenden Sie Kleinbuchstaben für Namen von Umgebungsvariablen

Im folgenden Beispiel wird eine Umgebungsvariable mit einem Namen in Kleinbuchstaben erstellt.

```
- name: Get secrets
uses: aws-actions/aws-secretsmanager-get-secrets@v2
with:
  secret-ids: exampleSecretName
  name-transformation: lowercase
```

Die Umgebungsvariable wurde erstellt:

```
examplesecretname: secretValue
```

Verwenden von AWS Secrets Manager-Secrets in AWS IoT Greengrass

AWS IoT Greengrass ist Software, mit der sich Cloud-Fähigkeiten auf lokale Geräte übertragen lassen. Dies ermöglicht es Geräten, Daten näher an der Informationsquelle zu erfassen und zu analysieren, selbstständig auf lokale Ereignisse zu reagieren und in lokalen Netzwerken sicher untereinander zu kommunizieren.

AWS IoT Greengrass ermöglicht Ihnen die Authentifizierung bei Services und Anwendungen aus Greengrass-Geräten, ohne Passwörter, Token oder andere Secrets fest kodieren zu müssen. AWS Secrets Manager ist ein Service, mit dem Sie Ihre Secrets sicher in der Cloud speichern und verwalten können. AWS IoT Greengrass erweitert Secrets Manager auf Greengrass-Core-Geräten, sodass Ihre Konnektoren und Lambda-Funktionen für die Interaktion mit Services und Anwendungen lokale Secrets verwenden können.

Um ein Secret in eine Greengrass-Gruppe zu integrieren, erstellen Sie eine Gruppen-Ressource, die auf das Secrets Manager-Secret verweist. Diese Secret-Ressource verweist über den zugehörigen ARN auf das Cloud-Secret. Informationen zum Erstellen, Verwalten und Verwenden geheimer Ressourcen finden Sie unter [Working with Secret Resources \(Arbeiten mit Secret-Ressourcen\)](#) im AWS IoT-Entwicklerhandbuch.

Informationen zum Bereitstellen von Secrets für AWS IoT Greengrass Core finden Sie unter [Bereitstellen von Secrets für AWS IoT Greengrass Core](#).

Verwenden Sie AWS Secrets Manager Geheimnisse in AWS Lambda Funktionen

Sie können die Lambda-Erweiterung AWS Parameters and Secrets verwenden, um AWS Secrets Manager Geheimnisse in Lambda-Funktionen abzurufen und zwischenspeichern, ohne ein SDK zu verwenden. Das Abrufen eines gecacheten Secrets ist schneller als das Abrufen aus Secrets Manager. Da für den Aufruf von Secrets-Manager-Manager-APIs Kosten anfallen, kann die Verwendung eines Caches Ihre Kosten senken. Die Erweiterung kann sowohl Secrets-

Manager-Secrets als auch Parameter-Store-Parameter abrufen. Weitere Informationen zum Parameterspeicher finden Sie unter [Parameter Store integration with Lambda extensions](#) (Integration des Parameterspeichers mit Lambda-Erweiterungen) im AWS Systems Manager -Benutzerhandbuch.

Eine Lambda-Erweiterung ist ein begleitender Prozess, der die Fähigkeiten einer Lambda-Funktion erweitert. Weitere Informationen finden Sie unter [Lambda-Erweiterungen](#) im Lambda-Entwicklerhandbuch. Informationen zur Verwendung der Erweiterung in einem Container-Image finden Sie unter [Arbeiten mit Lambda-Ebenen und Erweiterungen in Container-Images](#). Lambda protokolliert Ausführungsinformationen über die Erweiterung zusammen mit der Funktion mithilfe von Amazon CloudWatch Logs. Standardmäßig protokolliert die Erweiterung eine minimale Menge an CloudWatch Informationen unter. Um weitere Details zu protokollieren, setzen Sie die [Umgebungsvariable](#) `PARAMETERS_SECRETS_EXTENSION_LOG_LEVEL` auf `debug`.

Um den In-Memory-Cache für Parameter und Secrets bereitzustellen, stellt die Erweiterung der Lambda-Umgebung einen lokalen HTTP-Endpunkt, den Localhost-Port 2773, zur Verfügung. Sie können den Port konfigurieren, indem Sie die [Umgebungsvariable](#) auf `PARAMETERS_SECRETS_EXTENSION_HTTP_PORT` setzen.

Lambda instanziiert separate Instances, die der Gleichzeitigkeitsstufe entsprechen, die Ihre Funktion benötigt. Jede Instance ist isoliert und verwaltet ihren eigenen lokalen Cache Ihrer Konfigurationsdaten. Weitere Informationen zu Lambda-Instances und Gleichzeitigkeit finden Sie unter [Verwalten der reservierten Lambda-Gleichzeitigkeit](#) im Lambda-Entwicklerhandbuch.

Um die Erweiterung für ARM hinzuzufügen, müssen Sie die `arm64`-Architektur für Ihre Lambda-Funktion verwenden. Weitere Informationen finden Sie unter [Lambda-Befehlssatz-Architekturen](#) im Lambda-Entwicklerhandbuch. Die Erweiterung unterstützt ARM in den folgenden Regionen: Asien-Pazifik (Mumbai), USA Ost (Ohio), Europa (Irland), Europa (Frankfurt), Europa (Zürich), USA Ost (Nord-Virginia), Europa (London), Europa (Spanien) Asien-Pazifik (Tokio), USA West (Oregon), Asien-Pazifik (Singapur), Asien-Pazifik (Hyderabad) und Asien-Pazifik (Sydney).

Die Erweiterung verwendet einen AWS Client. Informationen zur Konfiguration des AWS Clients finden Sie in der [Einstellungsreferenz](#) im AWS SDK- und Tools-Referenzhandbuch. Wenn Ihre Lambda-Funktion in einer VPC ausgeführt wird, müssen Sie einen VPC-Endpunkt erstellen, damit die Erweiterung Secrets Manager aufrufen kann. Weitere Informationen finden Sie unter [VPC-Endpunkt](#).

Erforderliche Berechtigungen:

- Die [Lambda-Ausführungsrolle](#) muss über die `secretsmanager:GetSecretValue` Berechtigung für das Geheimnis verfügen.

- Wenn das Geheimnis mit einem vom Kunden verwalteten Schlüssel anstelle von verschlüsselt wird Von AWS verwalteter Schlüssel `aws/secretsmanager`, benötigt die Ausführungsrolle auch die `kms:Decrypt` Erlaubnis für den KMS-Schlüssel.

So verwenden Sie die Lambda-Erweiterung AWS Parameters and Secrets

1. Fügen Sie Ihrer Funktion die AWS Ebene mit dem Namen AWS Parameters and Secrets Lambda Extension hinzu. Anweisungen finden Sie unter [Hinzufügen von Ebenen zu Funktionen](#) im Lambda Developer Guide. Wenn Sie den Layer AWS CLI hinzufügen, benötigen Sie den ARN der Erweiterung. Eine Liste der ARNs finden Sie unter [AWS -Parameter and Secrets-Lambda-Erweiterung ARNs](#) im AWS Systems Manager -Benutzerhandbuch.
2. Erteilen Sie der Lambda-[Ausführungsrolle](#) Berechtigungen, um auf Secrets zugreifen zu können:
 - `secretsmanager:GetSecretValue`-Berechtigung für das Secret. Siehe [the section called "Beispiel: Berechtigung zum Abrufen von einzelnen Secret-Werten"](#).
 - (Optional) Wenn das Geheimnis mit einem vom Kunden verwalteten Schlüssel statt mit verschlüsselt wird Von AWS verwalteter Schlüssel `aws/secretsmanager`, benötigt die Ausführungsrolle auch die `kms:Decrypt` Erlaubnis für den KMS-Schlüssel.
 - Sie können die attributbasierte Zugriffskontrolle (ABAC) mit der Lambda-Rolle verwenden, um einen detaillierteren Zugriff auf Secrets im Konto zu ermöglichen. Weitere Informationen finden Sie unter [the section called "Beispiel: Steuern des Zugriffs auf Secrets mit Tags"](#) und [the section called "Beispiel: Beschränken Sie den Zugriff auf Identitäten mit Tags, die mit den Tags der Secrets übereinstimmen"](#).
3. Konfigurieren Sie den Cache mit Lambda-[Umgebungsvariablen](#).
4. Um Secrets aus dem Erweiterungs-Cache abzurufen, müssen Sie zuerst das `X-AWS-Parameters-Secrets-Token` zum Anforderungs-Header hinzufügen. Setzen Sie das Token auf `AWS_SESSION_TOKEN`, das von Lambda für alle laufenden Funktionen bereitgestellt wird. Die Verwendung dieses Headers zeigt an, dass sich der Anrufer in der Lambda-Umgebung befindet.

Das folgende Python-Beispiel zeigt, wie Sie den Header hinzufügen.

```
import os
headers = {"X-Aws-Parameters-Secrets-Token": os.environ.get('AWS_SESSION_TOKEN')}
```

5. Verwenden Sie eine der folgenden HTTP-GET-Anforderungen wie folgt, um ein Secret innerhalb der Lambda-Funktion abzurufen:

- Um ein Secret abzurufen, verwenden Sie für `secretId` den ARN oder den Namen des Secrets.

```
GET: /secretsmanager/get?secretId=secretId
```

- Um den vorherigen Secret-Wert oder eine bestimmte Version über die Staging-Bezeichnung, verwenden Sie für `secretId` den ARN oder den Namen des Secrets und für `versionStage` die Staging-Bezeichnung.

```
GET: /secretsmanager/get?secretId=secretId&versionStage=AWSPREVIOUS
```

- Um eine bestimmte Secret-Version anhand der ID abzurufen, verwenden Sie für `secretId` den ARN oder den Namen des Secrets und für `versionId` die Versions-ID.

```
GET: /secretsmanager/get?secretId=secretId&versionId=versionId
```

Example Secret abrufen (Python)

Das folgende Python-Beispiel zeigt, wie Sie ein Secret abrufen und das Ergebnis mit [`json.loads`](#) analysieren.

```
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 Parameter und Geheimnisse Umgebungsvariablen der Lambda-Erweiterung

Sie können die Erweiterung mit den folgenden Umgebungsvariablen konfigurieren.

Weitere Informationen zur Verwendung von Umgebungsvariablen finden Sie unter [Verwendung von Lambda-Umgebungsvariablen](#) im Lambda-Entwicklerhandbuch.

PARAMETERS_SECRETS_EXTENSION_CACHE_ENABLED

Legen Sie den Wert auf „true“ (wahr) fest, um Parameter und Secrets zwischenspeichern. Legen Sie den Wert auf „false“ (falsch) fest, um kein Caching durchzuführen. Der Standardwert ist „true“.

PARAMETERS_SECRETS_EXTENSION_CACHE_SIZE

Die maximale Anzahl von Secrets und Parametern, die zwischengespeichert werden sollen. Muss ein Wert zwischen 0 und 1 000 sein. Ein Wert von 0 bedeutet, dass keine Zwischenspeicherung stattfindet. Diese Variable wird ignoriert, wenn sowohl SSM_PARAMETER_STORE_TTL als auch SECRETS_MANAGER_TTL gleich 0 sind. Der Standardwert ist 1 000.

PARAMETERS_SECRETS_EXTENSION_HTTP_PORT

Der Port für den lokalen HTTP-Server. Der Standardwert ist 2 773.

PARAMETERS_SECRETS_EXTENSION_LOG_LEVEL

Die Protokollierungsstufe, die die Erweiterung bereitstellt: debug, info, warn, error, oder none. Setzen Sie dies auf debug, um die Cache-Konfiguration anzuzeigen. Der Standardwert ist info.

PARAMETERS_SECRETS_EXTENSION_MAX_CONNECTIONS

Maximale Anzahl von Verbindungen für HTTP-Clients, die die Erweiterung verwendet, um Anforderungen an Parameter Store oder Secrets Manager zu stellen. Dies ist eine Konfiguration pro Client. Der Standardwert ist 3.

SECRETS_MANAGER_TIMEOUT_MILLIS

Timeout für Anfragen an Secrets Manager in Millisekunden. Ein Wert von 0 bedeutet, dass es kein Timeout gegeben hat. Standard = 0.

SECRETS_MANAGER_TTL

TTL eines Secrets im Cache in Sekunden. Ein Wert von 0 bedeutet, dass keine Zwischenspeicherung stattfindet. Das Maximum beträgt 300 Sekunden. Diese Variable wird ignoriert, wenn PARAMETERS_SECRETS_CACHE_SIZE 0 ist. Der Standardwert beträgt 300 Sekunden.

SSM_PARAMETER_STORE_TIMEOUT_MILLIS

Timeout für Anfragen an den Parameterspeicher in Millisekunden. Ein Wert von 0 bedeutet, dass es kein Timeout gegeben hat. Standard = 0.

SSM_PARAMETER_STORE_TTL

TTL eines Parameters im Cache in Sekunden. Ein Wert von 0 bedeutet, dass keine Zwischenspeicherung stattfindet. Das Maximum beträgt 300 Sekunden. Diese Variable wird ignoriert, wenn `PARAMETERS_SECRETS_CACHE_SIZE` 0 ist. Der Standardwert beträgt 300 Sekunden.

Verwenden von AWS Secrets Manager-Secrets in Parameter Store

AWS Systems Manager Parameter Store ermöglicht eine sichere, hierarchische Speicherung für die Konfigurationsdatenverwaltung und die Verwaltung von Secrets. Sie können Daten wie Passwörter, Datenbankzeichenfolgen und Lizenzcodes als Parameterwerte speichern. Allerdings bietet Parameter Store keine automatischen Rotationsservices für gespeicherte Secrets. Stattdessen können Sie mit Parameter Store Ihr Secret in Secrets Manager speichern und dann als Parameter-Store-Parameter referenzieren.

Wenn Sie Parameter Store mit Secrets Manager konfigurieren, erfordert der `secret-id` Parameter Store einen Schrägstrich (`/`) vor der Name-Zeichenfolge.

Weitere Informationen finden Sie im Abschnitt [Referenzieren von AWS Secrets Manager Geheimnissen über Parameter-Store-Parameter](#) im AWS Systems Manager-Benutzerhandbuch.

AWS Secrets Manager Geheimnisse rotieren

Drehung ist der Prozess der periodischen Aktualisierung eines Secrets. Wenn Sie ein Secret rotieren, werden die Anmeldeinformationen sowohl im Secret als auch in der Datenbank oder im Service aktualisiert. Im Secrets Manager können Sie die automatische Rotation für Ihre Secrets einrichten. Es gibt zwei Formen der Rotation:

- [Verwaltete Rotation](#)— Für die meisten [verwalteten Geheimnisse](#) verwenden Sie die verwaltete Rotation, bei der der Service die Rotation für Sie konfiguriert und verwaltet. Die verwaltete Rotation verwendet keine Lambda-Funktion.
- [the section called “Rotation durch Lambda-Funktion”](#)— Bei anderen Arten von Geheimnissen verwendet die Secrets Manager Manager-Rotation eine Lambda-Funktion, um das Geheimnis und die Datenbank oder den Dienst zu aktualisieren.

Verwaltete Rotation von AWS Secrets Manager Geheimnissen

Einige Services bieten eine verwaltete Rotation an, bei der der Service die Rotation für Sie konfiguriert und verwaltet. Bei der verwalteten Rotation verwenden Sie keine AWS Lambda Funktion, um das Geheimnis und die Anmeldeinformationen in der Datenbank zu aktualisieren.

Die folgenden Services bieten eine verwaltete Rotation:

- Amazon Aurora bietet eine verwaltete Rotation für Master-Benutzeranmeldedaten. Weitere Informationen finden Sie unter [Passwortverwaltung mit Amazon Aurora und AWS Secrets Manager](#) im Amazon-Aurora-Benutzerhandbuch.
- Amazon ECS Service Connect bietet eine verwaltete Rotation für AWS Private Certificate Authority TLS-Zertifikate. Weitere Informationen finden Sie unter [TLS with Service Connect](#) im Amazon Elastic Container Service Developer Guide.
- Amazon RDS bietet eine verwaltete Rotation für Master-Benutzeranmeldedaten. Weitere Informationen finden Sie unter [Passwortverwaltung mit Amazon RDS und AWS Secrets Manager](#) im Amazon-RDS-Benutzerhandbuch.
- Amazon Redshift bietet eine verwaltete Rotation für Administrator Kennwörter. Weitere Informationen finden Sie unter [Verwaltung von Amazon Redshift-Administrator Kennwörtern mithilfe von AWS Secrets Manager](#) im Amazon Redshift Management Guide.

Tip

Informationen zu allen anderen Secret-Typen finden Sie unter [the section called “Rotation durch Lambda-Funktion”](#).

Die Rotation für verwaltete Geheimnisse ist in der Regel innerhalb einer Minute abgeschlossen. Während der Rotation erhalten neue Verbindungen, die das Geheimnis abrufen, möglicherweise die vorherige Version der Anmeldeinformationen. Bei Anwendungen wird dringend empfohlen, einen Datenbankbenutzer zu verwenden, der mit den Mindestberechtigungen erstellt wurde, die für Ihre Anwendung erforderlich sind, anstatt den Masterbenutzer zu verwenden. Für Anwendungsbenutzer können Sie für höchste Verfügbarkeit die [Rotationsstrategie für wechselnde Benutzer](#) verwenden.

Um den Zeitplan für die verwaltete Rotation zu ändern

1. Öffnen Sie das verwaltete Secret in der Secrets-Manager-Konsole. Sie können einem Link des Verwaltungsservices folgen oder in [der Secrets Manager-Konsole nach dem Secret suchen](#).
2. Geben Sie unter Rotation schedule (Drehungszeitplan) Ihren Zeitplan in der UTC-Zeitzone entweder im Schedule expression builder (Zeitplanausdruck-Generator) oder als Schedule expression (Zeitplanausdruck) ein. Secrets Manager speichert Ihren Zeitplan als `rate()`- oder `cron()`-Ausdruck. Das Rotationsfenster beginnt automatisch um Mitternacht, es sei denn, Sie geben eine Startzeit an. Sie können ein Secret bis zu alle vier Stunden rotieren. Weitere Informationen finden Sie unter [Rotationspläne](#).
3. (Optional) Wählen Sie für Dauer des Fensters die Länge des Fensters aus, in dem Secrets Manager Ihr Secret rotieren soll, z. B. **3h** für ein Drei-Stunden-Fenster. Das Fenster darf nicht in das nächste Rotationsfenster übergehen. Wenn Sie keine Fensterdauer angeben, wird das Fenster für einen Rotationsplan in Stunden automatisch nach einer Stunde geschlossen. Bei einem Rotationsplan in Tagen wird das Fenster am Ende des Tages automatisch geschlossen.
4. Wählen Sie Speichern.

So ändern Sie den Zeitplan für die verwaltete Rotation (AWS CLI)

- Rufen Sie die folgende Seite auf [rotate-secret](#). Im folgenden Beispiel rotiert das Secret am 1. und 15. Tag des Monats zwischen 16:00 Uhr und 18:00 Uhr UTC. Weitere Informationen finden Sie unter [Rotationspläne](#).

```
aws secretsmanager rotate-secret \
```

```
--secret-id MySecret \  
--rotation-rules "{\"ScheduleExpression\": \"cron(0 16 1,15 * ? *)\",  
\"Duration\": \"2h\"}"
```

Rotation durch Lambda-Funktion

Für viele Arten von Geheimnissen verwendet Secrets Manager eine AWS Lambda Funktion, um das Geheimnis und die Datenbank oder den Dienst zu aktualisieren. Hinweise zu den Kosten der Verwendung einer Lambda-Funktion finden Sie unter [Preisgestaltung](#).

Für einige [Verwaltete Geheimnisse](#) verwenden Sie die verwaltete Rotation. Um [Verwaltete Rotation](#) zu verwenden, erstellen Sie das Secret zunächst über den Verwaltungsservice.

Während der Drehung protokolliert Secrets Manager Ereignisse, die den Drehungszustand angeben. Weitere Informationen finden Sie unter [the section called “Loggen Sie sich mit AWS CloudTrail”](#).

Um ein Geheimnis zu rotieren, ruft Secrets Manager eine [Lambda-Funktion](#) gemäß dem von Ihnen eingerichteten Rotationsplan auf. Wenn Sie Ihren Secret-Wert auch manuell aktualisieren, während die automatische Rotation eingerichtet ist, berücksichtigt Secrets Manager diese Rotation bei der Berechnung des nächsten Rotationsdatums.

Während der Drehung ruft Secrets Manager dieselbe Funktion mehrmals auf – jeweils mit unterschiedlichen Parametern. Secrets Manager ruft die Funktion mit der folgenden JSON-Anforderungsstruktur von Parametern auf:

```
{  
  "Step" : "request.type",  
  "SecretId" : "string",  
  "ClientRequestToken" : "string"  
}
```

Wenn ein Schritt in der Rotation fehlschlägt, wiederholt Secrets Manager den gesamten Rotationsprozess mehrmals.

Themen

- [Automatische Rotation für Amazon RDS-, Amazon Aurora-, Amazon Redshift- oder Amazon DocumentDB DocumentDB-Geheimnisse einrichten](#)
- [Automatische Rotation für AWS Secrets Manager Nicht-Datenbankgeheimnisse einrichten](#)

- [Richten Sie die automatische Rotation ein mit dem AWS CLI](#)
- [Strategien zur Rotation von Lambda-Funktionen](#)
- [Lambda-Rotationsfunktionen](#)
- [AWS Secrets Manager Vorlagen für Rotationsfunktionen](#)
- [Rollenberechtigungen für die Ausführung der Lambda-Rotationsfunktion für AWS Secrets Manager](#)
- [Netzwerkzugriff für die Lambda-Rotationsfunktion](#)
- [Fehlerbehebung bei der AWS Secrets Manager Rotation](#)

Automatische Rotation für Amazon RDS-, Amazon Aurora-, Amazon Redshift- oder Amazon DocumentDB DocumentDB-Geheimnisse einrichten

In diesem Tutorial wird beschrieben, wie Sie Datenbankgeheimnisse einrichten [the section called “Rotation durch Lambda-Funktion”](#). Rotation ist der Prozess der periodischen Aktualisierung eines Secrets. Wenn Sie ein Secret drehen, werden die Anmeldeinformationen sowohl im Secret als auch in der Datenbank aktualisiert. Im Secrets Manager können Sie die automatische Drehung für Ihre Datenbank-Secrets einrichten.

Um die Rotation über die Konsole einzurichten, müssen Sie zunächst eine Rotationsstrategie auswählen. Dann konfigurieren Sie das Secret für die Drehung, wodurch eine Lambda-Drehungsfunktion erstellt wird, falls Sie noch keine haben. Die Konsole legt auch Berechtigungen für die Ausführungsrolle der Lambda-Funktion fest. Der letzte Schritt besteht darin, sicherzustellen, dass die Lambda-Drehungsfunktion sowohl auf Secrets Manager als auch auf Ihre Datenbank über das Netzwerk zugreifen kann.

Warning

Um die automatische Rotation zu aktivieren, benötigen Sie die Berechtigung, eine IAM-Ausführungsrolle für die Lambda-Rotationsfunktion zu erstellen und ihr eine Berechtigungsrichtlinie anzuhängen. Sie brauchen sowohl `iam:CreateRole` und `iam:AttachRolePolicy`-Berechtigungen. Durch die Gewährung dieser Berechtigungen kann sich eine Identität selbst alle Berechtigungen gewähren.

Schritte:

- [Schritt 1: Wählen Sie eine Drehungsstrategie und erstellen Sie \(optional\) ein Superuser-Secret](#)

- [Schritt 2: Konfigurieren Sie die Drehung und erstellen Sie eine Drehungsfunktion](#)
- [Schritt 3: \(Optional\) Zusätzliche Berechtigungsbedingungen für die Rotationsfunktion festlegen](#)
- [Schritt 4: Konfigurieren Sie den Netzwerkzugriff für die Drehungsfunktion](#)
- [Nächste Schritte](#)

Schritt 1: Wählen Sie eine Drehungsstrategie und erstellen Sie (optional) ein Superuser-Secret

Informationen zu den von Secrets Manager angebotenen Strategien finden Sie unter [the section called "Strategien zur Rotation von Lambda-Funktionen"](#).

Wenn Sie die Strategie für alternierende Benutzer wählen, müssen Sie [Erstellen eines Datenbank-Secrets](#) und Superuser-Anmeldeinformationen der Datenbank darin speichern. Sie benötigen ein Secret mit Superuser-Anmeldeinformationen, da die Drehung den ersten Benutzer kloniert und die meisten Benutzer nicht über diese Berechtigung verfügen. Beachten Sie, dass Amazon RDS Proxy die Strategie für wechselnde Benutzer nicht unterstützt.

Schritt 2: Konfigurieren Sie die Drehung und erstellen Sie eine Drehungsfunktion

Aktivieren Sie die Drehung für ein Secret von Amazon RDS, Amazon DocumentDB oder Amazon Redshift wie folgt:

1. Öffnen Sie die Secrets-Manager-Konsole unter <https://console.aws.amazon.com/secretsmanager/>.
2. Wählen Sie auf der Seite Secrets Ihr Secret aus.
3. Klicken Sie auf der Seite mit den Secret-Details im Abschnitt Rotation configuration (Rotationskonfiguration) auf Edit rotation (Rotation bearbeiten).
4. Führen Sie im Dialogfeld Edit rotation configuration (Rotationskonfiguration bearbeiten) die folgenden Schritte aus:
 - a. Schalten Sie die automatische Rotation ein.
 - b. Geben Sie unter Rotation schedule (Drehungszeitplan) Ihren Zeitplan in der UTC-Zeitzone entweder im Schedule expression builder (Zeitplanausdruck-Generator) oder als Schedule expression (Zeitplanausdruck) ein. Secrets Manager speichert Ihren Zeitplan als `rate()`- oder `cron()`-Ausdruck. Das Rotationsfenster beginnt automatisch um Mitternacht, es sei denn, Sie geben eine Startzeit an. Sie können ein Secret bis zu alle vier Stunden rotieren. Weitere Informationen finden Sie unter [Rotationspläne](#).

- c. (Optional) Wählen Sie für Dauer des Fensters die Länge des Fensters aus, in dem Secrets Manager Ihr Secret rotieren soll, z. B. **3h** für ein Drei-Stunden-Fenster. Das Fenster darf nicht in das nächste Rotationsfenster übergehen. Wenn Sie keine Fensterdauer angeben, wird das Fenster für einen Rotationsplan in Stunden automatisch nach einer Stunde geschlossen. Bei einem Rotationsplan in Tagen wird das Fenster am Ende des Tages automatisch geschlossen.
- d. (Optional) Wählen Sie Sofort rotieren, wenn das Secret gespeichert ist, um Ihr Secret zu rotieren, wenn Sie Ihre Änderungen speichern. Wenn Sie das Kontrollkästchen deaktivieren, beginnt die erste Rotation nach dem von Ihnen festgelegten Zeitplan.

Wenn die Drehung fehlschlägt, z. B. weil die Schritte 3 und 4 noch nicht abgeschlossen sind, wiederholt Secrets Manager den Drehungsvorgang mehrmals.

- e. Führen Sie unter Rotationsfunktion einen der folgenden Schritte aus:
 - Wählen Sie Create a new Lambda function (Neue Lambda-Funktion erstellen) aus und geben Sie einen Namen für Ihre neue Funktion ein. Secrets Manager fügt „SecretsManager“ am Anfang des Funktionsnamens hinzu. Secrets Manager erstellt die Funktion basierend auf der entsprechenden [Vorlage](#) und legt die erforderlichen [Berechtigungen](#) für die Lambda-Ausführungsrolle fest.
 - Wählen Sie Use an existing Lambda function (Bestehende Lambda-Funktion verwenden), um eine Drehungsfunktion wiederzuverwenden, die Sie für ein anderes Secret verwendet haben. Die Drehungsfunktionen, die unter Recommended VPC configurations (Empfohlene VPC-Konfigurationen) aufgeführt sind, haben dieselbe VPC und Sicherheitsgruppe wie die Datenbank, wodurch die Funktion auf die Datenbank zugreifen kann.
- f. Wählen Sie für die Rotationsstrategie die Strategie Einzelbenutzer oder Alternierende Benutzer aus. Weitere Informationen finden Sie unter [the section called “Schritt 1: Wählen Sie eine Drehungsstrategie und erstellen Sie \(optional\) ein Superuser-Secret”](#).

5. Wählen Sie Save (Speichern) aus.

Schritt 3: (Optional) Zusätzliche Berechtigungsbedingungen für die Rotationsfunktion festlegen

Wir empfehlen, in die Ressourcenrichtlinie für Ihre Drehungsfunktion den Kontextschlüssel [aws:SourceAccount](#) aufzunehmen, um zu verhindern, dass Lambda als [verwirrter Stellvertreter](#) verwendet wird. Für einige AWS Dienste AWS empfiehlt es sich, sowohl den Bedingungsschlüssel

als auch den [aws:SourceAccount](#) globalen Bedingungsschlüssel zu verwenden, um das verwirrte [aws:SourceArn](#) Deputy-Szenario zu vermeiden. Wenn Sie jedoch die `aws:SourceArn`-Bedingung in Ihre Drehungsfunktions-Richtlinie einschließen, kann die Drehungsfunktion nur verwendet werden, um das von diesem ARN angegebene Secret zu rotieren. Es wird empfohlen, nur den Kontextschlüssel `aws:SourceAccount` anzugeben, damit Sie die Drehungsfunktion für mehrere Geheimnisse verwenden können.

Aktualisieren Sie die Ressourcenrichtlinie Ihrer Drehungsfunktion wie folgt:

1. Wählen Sie in der Secrets-Manager-Konsole Ihr Secret aus und wählen Sie dann auf der Detailseite unter Rotation configuration (Drehungs-Konfiguration) die Lambda-Drehungsfunktion aus. Die Lambda-Konsole wird geöffnet.
2. Folgen Sie den Anweisungen unter [Verwenden von ressourcenbasierten Richtlinien für Lambda](#), um eine `aws:sourceAccount`-Bedingung hinzuzufügen.

```
"Condition": {
  "StringEquals": {
    "AWS:SourceAccount": "123456789012"
  }
},
```

Wenn das Secret mit einem anderen KMS-Schlüssel als Von AWS verwalteter Schlüssel `aws/secretsmanager` verschlüsselt ist, gewährt Secrets Manager der Lambda-Ausführungsrolle die Berechtigung, den Schlüssel zu verwenden. Sie können den [SecretARN-Verschlüsselungskontext](#) verwenden, um die Verwendung der Entschlüsselungsfunktion einzuschränken, sodass die Rolle der Rotationsfunktion nur Zugriff auf das Secret hat, für dessen Rotation diese verantwortlich ist.

So aktualisieren Sie Ihre Ausführungsrolle Ihrer Rotationsfunktion

1. Wählen Sie in der Lambda-Rotationsfunktion die Option Konfiguration und dann unter Ausführungsrolle den Rollennamen aus.
2. Folgen Sie den Anweisungen unter [Ändern einer Richtlinie für Rollenberechtigungen](#), um eine `kms:EncryptionContext:SecretARN`-Bedingung hinzuzufügen.

```
"Condition": {
  "StringEquals": {
    "kms:EncryptionContext:SecretARN": "SecretARN"
  }
}
```

```
},
```

Schritt 4: Konfigurieren Sie den Netzwerkzugriff für die Drehungsfunktion

Weitere Informationen finden Sie unter [the section called “Netzwerkzugriff für die Lambda-Rotationsfunktion”](#).

Nächste Schritte

Siehe [the section called “Fehlerbehebung bei der -Rotation”](#).

Automatische Rotation für AWS Secrets Manager Nicht-Datenbankgeheimnisse einrichten

In diesem Tutorial wird beschrieben, wie Sie geheime Daten einrichten, [the section called “Rotation durch Lambda-Funktion”](#) die keine Datenbanken sind. Rotation ist der Prozess der periodischen Aktualisierung eines Secrets. Wenn Sie ein Secret drehen, werden die Anmeldeinformationen sowohl im Secret als auch in der Datenbank oder im Service, für die bzw. den das Secret bestimmt ist, aktualisiert.

Informationen zu Datenbank-Secrets finden Sie unter [Automatische Rotierung für Datenbank-Secrets \(Konsole\)](#).

Warning

Um die automatische Rotation zu aktivieren, benötigen Sie die Berechtigung, eine IAM-Ausführungsrolle für die Lambda-Rotationsfunktion zu erstellen und ihr eine Berechtigungsrichtlinie anzuhängen. Sie brauchen sowohl `iam:CreateRole` und `iam:AttachRolePolicy`-Berechtigungen. Durch die Gewährung dieser Berechtigungen kann sich eine Identität selbst alle Berechtigungen gewähren.

Schritte:

- [Schritt 1: Erstellen Sie eine generische Rotationsfunktion](#)
- [Schritt 2: Schreiben Sie den Drehungsfunktionscode](#)
- [Schritt 3: Konfigurieren Sie das Geheimnis für die Rotation](#)

- [Schritt 4: Erlauben Sie der Rotationsfunktion den Zugriff auf Secrets Manager und Ihre Datenbank oder Ihren Dienst](#)
- [Schritt 5: Erlauben Sie Secrets Manager, die Rotationsfunktion aufzurufen](#)
- [Schritt 6: Richten Sie den Netzwerkzugriff für die Rotationsfunktion ein](#)
- [Nächste Schritte](#)

Schritt 1: Erstellen Sie eine generische Rotationsfunktion

Erstellen Sie zunächst eine Lambda-Rotationsfunktion. Sie wird nicht den Code enthalten, mit dem Sie Ihr Geheimnis rotieren können. Sie werden ihn also in einem späteren Schritt schreiben. Hinweise zur Funktionsweise einer Rotationsfunktion finden Sie unter [the section called “Lambda-Rotationsfunktionen”](#).

In unterstützten Regionen können Sie sie verwenden, AWS Serverless Application Repository um die Funktion aus einer Vorlage zu erstellen. Eine Liste der unterstützten Regionen finden Sie unter [AWS Serverless Application Repository Häufig gestellte Fragen](#). In anderen Regionen erstellen Sie die Funktion von Grund auf neu und kopieren den Vorlagencode in die Funktion.

Um eine generische Rotationsfunktion zu erstellen

1. Informationen darüber, ob sie in Ihrer Region unterstützt AWS Serverless Application Repository wird, finden Sie in der AWS allgemeinen Referenz unter [AWS Serverless Application Repository Endpunkte und Kontingente](#).
2. Führen Sie eine der folgenden Aktionen aus:
 - AWS Serverless Application Repository Was in Ihrer Region unterstützt wird:
 - a. Wählen Sie in der Lambda-Konsole Applications und anschließend Create Application aus.
 - b. Wählen Sie auf der Seite „Anwendung erstellen“ die Registerkarte Serverlose Anwendung aus.
 - c. Geben **SecretsManagerRotationTemplate** Sie im Suchfeld unter Öffentliche Anwendungen den Text ein.
 - d. Wählen Sie Apps anzeigen, die benutzerdefinierte IAM-Rollen oder Ressourcenrichtlinien erstellen.
 - e. Wählen Sie die SecretsManagerRotationTemplateKachel aus.

- f. Füllen Sie auf der Seite Überprüfen, konfigurieren und bereitstellen in der Kachel Anwendungseinstellungen die erforderlichen Felder aus.
 - Geben Sie als Endpunkt den Endpunkt für Ihre Region ein, einschließlich **https://**. Eine Liste der Endpunkte finden Sie unter [the section called “Secrets-Manager-Endpunkte”](#).
 - Um die Lambda-Funktion in eine VPC zu integrieren, schließen Sie die vpcSecurityGroupIDs und ein. vpcSubnetIds
- g. Wählen Sie Bereitstellen.
- Falls in Ihrer Region AWS Serverless Application Repository nicht unterstützt wird:
 - a. Wählen Sie in der Lambda-Konsole Funktionen und dann Funktion erstellen aus.
 - b. Gehen Sie auf der Seite Create function (Funktion erstellen) wie folgt vor:
 - i. Wählen Sie Von Grund auf neu schreiben aus.
 - ii. Geben Sie unter Function name (Funktionsname) einen Namen für Ihre Drehungsfunktion ein.
 - iii. Wählen Sie für Runtime (Laufzeit) die Option Python 3.9 aus.
 - iv. Wählen Sie Funktion erstellen.

Schritt 2: Schreiben Sie den Drehungsfunktionscode

In diesem Schritt schreiben Sie den Code, der das Geheimnis aktualisiert, und den Dienst oder die Datenbank, für den das Geheimnis bestimmt ist. Hinweise zur Funktionsweise einer Rotationsfunktion, einschließlich Tipps zum Schreiben Ihrer eigenen Rotationsfunktion, finden Sie unter [the section called “Lambda-Rotationsfunktionen”](#). Sie können die auch [Rotationsfunktionsvorlagen](#) als Referenz verwenden.

Schritt 3: Konfigurieren Sie das Geheimnis für die Rotation

In diesem Schritt legen Sie einen Rotationsplan für Ihr Geheimnis fest und verbinden die Rotationsfunktion mit dem Geheimnis.

Konfigurieren Sie die Drehung und erstellen Sie eine leere Drehungsfunktion wie folgt:

1. Öffnen Sie die Secrets-Manager-Konsole unter <https://console.aws.amazon.com/secretsmanager/>.

2. Wählen Sie auf der Seite Secrets Ihr Secret aus.
3. Klicken Sie auf der Seite mit den Secret-Details im Abschnitt Rotation configuration (Rotationskonfiguration) auf Edit rotation (Rotation bearbeiten). Führen Sie im Dialogfeld Edit rotation configuration (Rotationskonfiguration bearbeiten) die folgenden Schritte aus:
 - a. Schalten Sie die automatische Rotation ein.
 - b. Geben Sie unter Rotation schedule (Drehungszeitplan) Ihren Zeitplan in der UTC-Zeitzone entweder im Schedule expression builder (Zeitplanausdruck-Generator) oder als Schedule expression (Zeitplanausdruck) ein. Secrets Manager speichert Ihren Zeitplan als `rate()`- oder `cron()`-Ausdruck. Das Rotationsfenster beginnt automatisch um Mitternacht, es sei denn, Sie geben eine Startzeit an. Sie können ein Secret bis zu alle vier Stunden rotieren. Weitere Informationen finden Sie unter [Rotationspläne](#).
 - c. (Optional) Wählen Sie für Dauer des Fensters die Länge des Fensters aus, in dem Secrets Manager Ihr Secret rotieren soll, z. B. **3h** für ein Drei-Stunden-Fenster. Das Fenster darf nicht in das nächste Rotationsfenster übergehen. Wenn Sie keine Fensterdauer angeben, wird das Fenster für einen Rotationsplan in Stunden automatisch nach einer Stunde geschlossen. Bei einem Rotationsplan in Tagen wird das Fenster am Ende des Tages automatisch geschlossen.
 - d. (Optional) Wählen Sie Sofort rotieren, wenn das Secret gespeichert ist, um Ihr Secret zu rotieren, wenn Sie Ihre Änderungen speichern. Wenn Sie das Kontrollkästchen deaktivieren, beginnt die erste Rotation nach dem von Ihnen festgelegten Zeitplan.
 - e. Wählen Sie unter Rotationsfunktion die Lambda-Funktion aus, die Sie in Schritt 1 erstellt haben.
 - f. Wählen Sie Speichern.

Schritt 4: Erlauben Sie der Rotationsfunktion den Zugriff auf Secrets Manager und Ihre Datenbank oder Ihren Dienst

Die Lambda-Drehungsfunktion benötigt die Berechtigung, auf das Secret in Secrets Manager zuzugreifen, und sie benötigt die Berechtigung, auf Ihre Datenbank oder Ihren Service zuzugreifen. In diesem Schritt erteilen Sie diese Berechtigungen der Lambda-Ausführungsrolle. Wenn das Secret mit einem anderen KMS-Schlüssel als den Von AWS verwalteter Schlüssel `aws/secretsmanager` verschlüsselt wird, müssen Sie der Lambda-Ausführungsrolle die Berechtigung erteilen, den Schlüssel zu verwenden. Sie können den [SecretARN-Verschlüsselungskontext](#) verwenden, um die Verwendung der Entschlüsselungsfunktion einzuschränken, sodass die Rolle der Rotationsfunktion

nur Zugriff auf das Secret hat, für dessen Rotation diese verantwortlich ist. Richtlinienbeispiele finden Sie unter [Berechtigungen für Rotation](#).

Anweisungen dazu finden Sie unter [Lambda-Ausführungsrolle](#) im AWS Lambda - Entwicklerhandbuch.

Schritt 5: Erlauben Sie Secrets Manager, die Rotationsfunktion aufzurufen

Damit Secrets Manager die Rotationsfunktion auf dem von Ihnen eingerichteten Rotationsplan aufrufen kann, müssen Sie dem Secrets Manager Manager-Dienstprinzipal in der Ressourcenrichtlinie der Lambda-Funktion die `lambda:InvokeFunction` Erlaubnis erteilen.

Wir empfehlen, in die Ressourcenrichtlinie für Ihre Drehungsfunktion den Kontextschlüssel [aws:SourceAccount](#) aufzunehmen, um zu verhindern, dass Lambda als [verwirrter Stellvertreter](#) verwendet wird. Für einige AWS Dienste AWS empfiehlt es sich, sowohl den Bedingungsschlüssel als auch den [aws:SourceAccount](#) globalen Bedingungsschlüssel zu verwenden, um ein verwirrtes [aws:SourceArn](#) Deputy-Szenario zu vermeiden. Wenn Sie jedoch die `aws:SourceArn`-Bedingung in Ihre Drehungsfunktions-Richtlinie einschließen, kann die Drehungsfunktion nur verwendet werden, um das von diesem ARN angegebene Secret zu rotieren. Es wird empfohlen, nur den Kontextschlüssel `aws:SourceAccount` anzugeben, damit Sie die Drehungsfunktion für mehrere Geheimnisse verwenden können.

Informationen zum Anfügen einer Ressourcenrichtlinie an eine Lambda-Funktion finden Sie unter [Verwenden von ressourcenbasierten Richtlinien für Lambda](#).

Die folgende Richtlinie ermöglicht Secrets Manager, eine Lambda-Funktion aufzurufen.

```
{
  "Version": "2012-10-17",
  "Id": "default",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "secretsmanager.amazonaws.com"
      },
      "Action": "lambda:InvokeFunction",
      "Condition": {
        "StringEquals": {
          "AWS:SourceAccount": "123456789012"
        }
      }
    },
  ],
}
```

```
    "Resource": "LambdaRotationFunctionARN"
  }
]
}
```

Schritt 6: Richten Sie den Netzwerkzugriff für die Rotationsfunktion ein

In diesem Schritt erlauben Sie der Rotationsfunktion, eine Verbindung sowohl mit Secrets Manager als auch mit dem Dienst oder der Datenbank herzustellen, für den das Geheimnis bestimmt ist. Die Rotationsfunktion muss Zugriff auf beide haben, um das Geheimnis rotieren zu können. Siehe [the section called "Netzwerkzugriff für die Lambda-Rotationsfunktion"](#).

Nächste Schritte

Bei der Konfiguration der Rotation in Schritt 3 haben Sie einen Zeitplan für die Rotation des Secrets festgelegt. Wenn die Rotation zum geplanten Zeitpunkt fehlschlägt, versucht Secrets Manager die Rotation mehrmals. Sie können eine Rotation auch sofort starten, indem Sie den Anweisungen unter folgen [Secret sofort drehen](#).

Falls die Rotation fehlschlägt, finden Sie weitere Informationen unter [Fehlerbehebung bei der - Rotation](#).

Richten Sie die automatische Rotation ein mit dem AWS CLI

In diesem Tutorial wird beschrieben, wie Sie mit [the section called "Rotation durch Lambda-Funktion"](#) dem einrichten AWS CLI. Wenn Sie ein Secret drehen, werden die Anmeldeinformationen sowohl im Secret als auch in der Datenbank oder im Service, für die bzw. den das Secret bestimmt ist, aktualisiert.

Sie können die Rotation auch über die Konsole einrichten. Informationen zu Datenbank-Secrets finden Sie unter [Automatische Rotierung für Datenbank-Secrets \(Konsole\)](#). Informationen zu allen anderen Secret-Typen finden Sie unter [Automatische Rotation für Nicht-Datenbankgeheimnisse \(Konsole\)](#).

Um die Rotation mithilfe von einzurichten AWS CLI, müssen Sie, wenn Sie ein Datenbankgeheimnis rotieren, zunächst eine Rotationsstrategie wählen. Wenn Sie die Strategie für alternierende Benutzer wählen, müssen Sie ein separates Secret mit Anmeldeinformationen für einen Datenbank-Superuser speichern. Als Nächstes schreiben Sie den Drehungsfunktionscode. Secrets Manager bietet Vorlagen, auf denen Sie Ihre Funktion aufbauen können. Anschließend erstellen Sie eine Lambda-Funktion mit Ihrem Code und legen Berechtigungen sowohl für die Lambda-Funktion als auch für die

Lambda-Ausführungsrolle fest. Der nächste Schritt besteht darin, sicherzustellen, dass die Lambda-Funktion über das Netzwerk sowohl auf Secrets Manager als auch auf Ihre Datenbank oder Ihren Dienst zugreifen kann. Schließlich konfigurieren Sie das Secret für die Drehung.

Schritte:

- [Voraussetzung für Datenbankgeheimnisse: Wählen Sie eine Rotationsstrategie](#)
- [Schritt 1: Schreiben Sie den Code für die Rotationsfunktion](#)
- [Schritt 2: Lambda-Funktion erstellen](#)
- [Schritt 3: Netzwerkzugriff einrichten](#)
- [Schritt 4: Konfigurieren Sie das Geheimnis für die Rotation](#)
- [Nächste Schritte](#)

Voraussetzung für Datenbankgeheimnisse: Wählen Sie eine Rotationsstrategie

Informationen zu den von Secrets Manager angebotenen Strategien finden Sie unter [the section called “Strategien zur Rotation von Lambda-Funktionen”](#).

Option 1: Strategie für einen einzelnen Benutzer

Wenn Sie sich für die Einzelbenutzerstrategie entscheiden, können Sie mit Schritt 1 fortfahren.

Option 2: Strategie für wechselnde Benutzer

Wenn Sie sich für die Strategie mit wechselnden Benutzern entscheiden, müssen Sie:

- [Erstellen Sie ein Datenbankgeheimnis](#) und speichern Sie darin die Anmeldeinformationen für den Datenbank-Superuser. Sie benötigen einen geheimen Schlüssel mit Superuser-Anmeldeinformationen, da der erste Benutzer abwechselnd geklont wird und die meisten Benutzer nicht über diese Berechtigung verfügen.
- Fügen Sie den ARN des Superuser-Geheimnisses zum ursprünglichen Geheimnis hinzu. Weitere Informationen finden Sie unter [the section called “JSON-Struktur eines Secrets”](#).

Beachten Sie, dass Amazon RDS Proxy die Strategie für wechselnde Benutzer nicht unterstützt.

Schritt 1: Schreiben Sie den Code für die Rotationsfunktion

Um ein Secret zu drehen, benötigen Sie eine Drehungsfunktion. Eine Drehungsfunktion ist eine Lambda-Funktion, die Secrets Manager aufruft, um Ihr Secret zu drehen. Weitere Informationen

finden Sie unter [the section called “Rotation durch Lambda-Funktion”](#). In diesem Schritt schreiben Sie den Code, der das Geheimnis aktualisiert, und den Dienst oder die Datenbank, für den das Geheimnis bestimmt ist.

Secrets Manager bietet Vorlagen für Amazon RDS-, Amazon Aurora-, Amazon Redshift- und Amazon DocumentDB DocumentDB-Datenbankgeheimnisse in. [Rotationsfunktionsvorlagen](#)

Um den Code für die Rotationsfunktion zu schreiben

1. Führen Sie eine der folgenden Aktionen aus:
 - Überprüfen Sie die Liste der [Vorlagen für Rotationsfunktionen](#). Wenn es eine gibt, die Ihrer Service- und Rotationsstrategie entspricht, kopieren Sie den Code.
 - Für andere Arten von Geheimnissen schreiben Sie Ihre eigene Rotationsfunktion. Anweisungen finden Sie unter [the section called “Lambda-Rotationsfunktionen”](#).
2. Speichern Sie die Datei zusammen mit allen erforderlichen Abhängigkeiten in einer ZIP-Datei *my-function.zip*.

Schritt 2: Lambda-Funktion erstellen

In diesem Schritt erstellen Sie die Lambda-Funktion mithilfe der ZIP-Datei, die Sie in Schritt 1 erstellt haben. Sie legen auch die [Lambda-Ausführungsrolle](#) fest, die Lambda übernimmt, wenn die Funktion aufgerufen wird.

Erstellen Sie eine Lambda-Drehungsfunktion und Ausführungsrolle wie folgt:

1. Erstellen Sie eine Vertrauensrichtlinie für die Lambda-Ausführungsrolle und speichern Sie sie als JSON-Datei. Beispiele und weitere Informationen finden Sie unter. [the section called “Berechtigungen für Rotation”](#) Die Richtlinie muss:
 - Der Rolle erlauben, Secrets-Manager-Vorgänge für das Secret aufzurufen.
 - Erlauben Sie der Rolle, den Dienst aufzurufen, für den das Geheimnis bestimmt ist, z. B. um ein neues Passwort zu erstellen.
2. Erstellen Sie die Lambda-Ausführungsrolle und wenden Sie die Vertrauensrichtlinie an, die Sie im vorherigen Schritt erstellt haben, indem Sie aufrufen [iam create-role](#).

```
aws iam create-role \  
  --role-name rotation-lambda-role \  
  --assume-role-policy-document file://trust-policy.json
```

3. Die Lambda-Funktion aus der ZIP-Datei erstellen, indem sie [lambda create-function](#) aufruft.

```
aws lambda create-function \  
  --function-name my-rotation-function \  
  --runtime python3.7 \  
  --zip-file fileb://my-function.zip \  
  --handler .handler \  
  --role arn:aws:iam::123456789012:role/service-role/rotation-lambda-role
```

4. Eine Ressourcenrichtlinie für die Lambda-Funktion festlegen, damit Secrets Manager sie durch Aufrufen von [lambda add-permission](#) aufrufen kann.

```
aws lambda add-permission \  
  --function-name my-rotation-function \  
  --action lambda:InvokeFunction \  
  --statement-id SecretsManager \  
  --principal secretsmanager.amazonaws.com \  
  --source-account 123456789012
```

Schritt 3: Netzwerkzugriff einrichten

Weitere Informationen finden Sie unter [the section called “Netzwerkzugriff für die Lambda-Rotationsfunktion”](#).

Schritt 4: Konfigurieren Sie das Geheimnis für die Rotation

Um die automatische Drehung für Ihr Secret zu aktivieren, rufen Sie [rotate-secret](#) an. Sie können einen Drehungszeitplan mit einem `cron()`- oder `rate()`-Zeitplanausdruck festlegen, und Sie können die Dauer eines Drehungsfensters festlegen. Weitere Informationen finden Sie unter [the section called “Rotationspläne”](#).

```
aws secretsmanager rotate-secret \  
  --secret-id MySecret \  
  --rotation-lambda-arn arn:aws:lambda:Region:123456789012:function:my-rotation-  
function \  
  --rotation-rules "{\"ScheduleExpression\": \"cron(0 16 1,15 * ? *)\", \"Duration\":  
  \"2h\"}"
```

Nächste Schritte

Siehe [the section called “Fehlerbehebung bei der -Rotation”](#).

Strategien zur Rotation von Lambda-Funktionen

Denn für [the section called “Rotation durch Lambda-Funktion”](#) Datenbankgeheimnisse bietet Secrets Manager zwei Rotationsstrategien.

Rotationsstrategie: Einzelbenutzer

Diese Strategie aktualisiert die Anmeldeinformationen für einen Benutzer in einem Secret. Da Benutzer bei Amazon-RDS-Db2-Instances ihre eigenen Passwörter nicht ändern können, müssen Sie Administratoranmeldedaten in einem separaten Secrets angeben. Dies ist die einfachste Rotationsstrategie und eignet sich für die meisten Anwendungsfälle. Insbesondere empfehlen wir Ihnen, diese Strategie für Anmeldeinformationen für einmalige (Ad-hoc) oder interaktive Benutzer zu verwenden.

Wenn das Secret rotiert, werden offene Datenbankverbindungen nicht gelöscht. Während dieser Rotation gibt es einen kurzen Zeitraum zwischen dem Ändern des Passworts in der Datenbank und dem Zeitpunkt, an dem das Secret aktualisiert wird. Während dieser Zeit besteht ein geringes Risiko, dass die Datenbank Anrufe ablehnt, die die rotierten Anmeldeinformationen verwenden. Sie können dieses Risiko abschwächen indem Sie eine [angemessene Wiederholungsstrategie](#) nutzen. Nach der Rotation verwenden neue Verbindungen die neuen Anmeldeinformationen.

Rotationsstrategie: wechselnde Benutzer

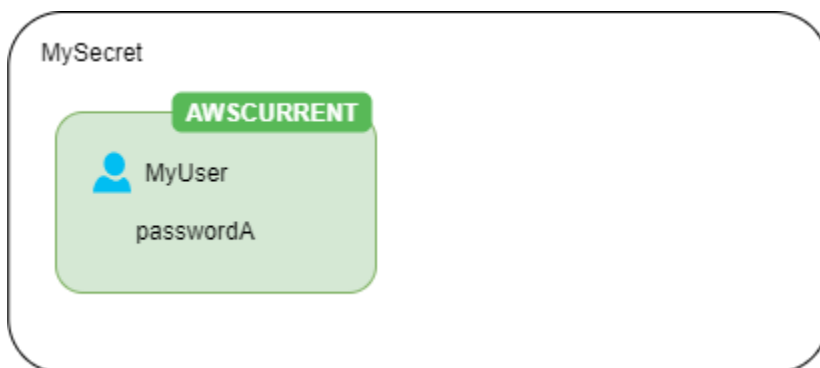
Diese Strategie aktualisiert die Anmeldeinformationen für zwei Benutzer in einem Secret. Sie erstellen den ersten Benutzer, und während der ersten Rotation kloniert die Rotationsfunktion ihn, um den zweiten Benutzer zu erstellen. Jedes Mal, wenn das Secret rotiert, wechselt die Rotationsfunktion ab, welches Benutzerkennwort sie aktualisiert. Da die meisten Benutzer keine Berechtigung haben, sich selbst zu klonen, müssen Sie die Anmeldeinformationen für einen `superuser` in einem anderen Secret angeben. Wir empfehlen die Verwendung der Einzelbenutzer-Drehungsstrategie, wenn geklonte Benutzer in Ihrer Datenbank nicht über dieselben Berechtigungen verfügen wie der ursprüngliche Benutzer, sowie für Anmeldeinformationen für einmalige (Ad-hoc-) oder interaktive Benutzer.

Diese Strategie ist für Datenbanken mit Berechtigungsmodellen geeignet, bei denen eine Rolle Eigentümer der Datenbanktabellen ist und eine zweite Rolle die Berechtigung zum Zugriff auf

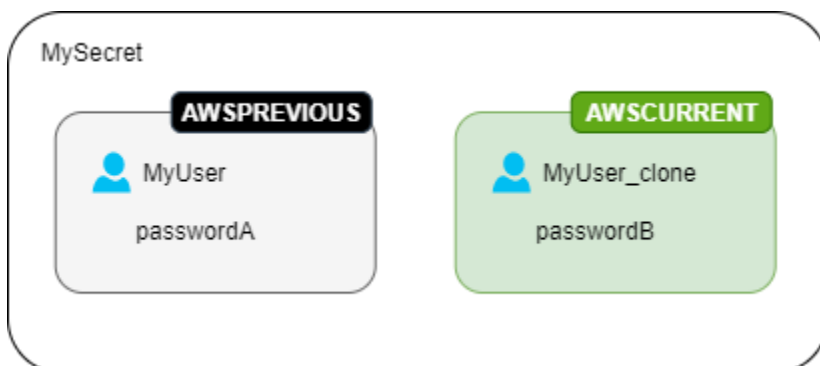
die Datenbanktabellen hat. Sie ist auch für Anwendungen geeignet, die eine hohe Verfügbarkeit erfordern. Wenn eine Anwendung das Secret während der Rotation abrufen, erhält die Anwendung dennoch einen gültigen Satz von Anmeldeinformationen. Nach der Rotation sind sowohl `user-` als auch `user_clone-`Anmeldeinformationen gültig. Die Wahrscheinlichkeit, dass Anwendungen bei dieser Art der Rotation abgelehnt werden, ist noch geringer als bei der Rotation durch einen Einzelbenutzer. Wenn die Datenbank in einer Serverfarm gehostet wird, bei der die Passwortänderung einige Zeit für die Weitergabe an alle Server benötigt, besteht die Gefahr, dass die Datenbank Aufrufe ablehnt, die die neuen Anmeldeinformationen verwenden. Sie können dieses Risiko abschwächen indem Sie eine [angemessene Wiederholungsstrategie](#) nutzen.

Secrets Manager erstellt den geklonten Benutzer mit denselben Berechtigungen wie die des ursprünglichen Benutzers. Wenn Sie nach der Erstellung des Klons die Berechtigungen des ursprünglichen Benutzers ändern, müssen Sie auch die Berechtigungen des geklonten Benutzers ändern.

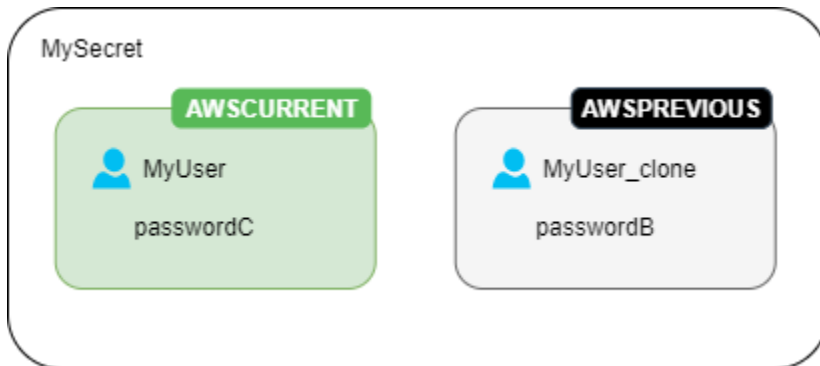
Wenn Sie beispielsweise ein Secret mit den Anmeldeinformationen eines Datenbankbenutzers erstellen, enthält das Secret eine Version mit diesen Anmeldeinformationen.



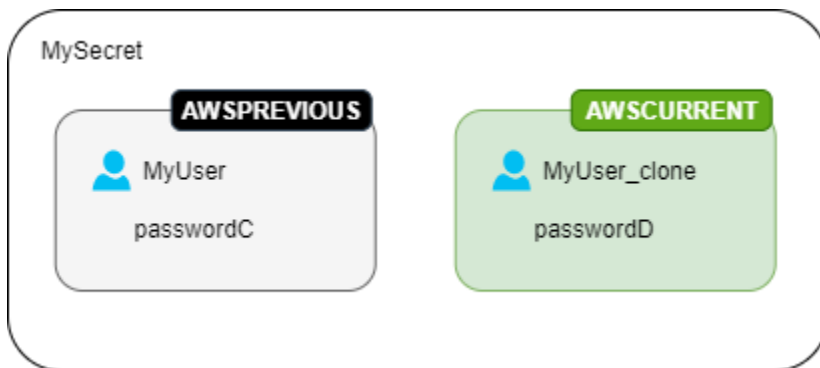
Erste Rotation – Die Rotationsfunktion erstellt einen Klon Ihres Benutzers mit einem generierten Passwort, und diese Anmeldedaten gelten als die aktuelle geheime Version.



Zweite Rotation – Die Rotationsfunktion aktualisiert das Passwort für den ursprünglichen Benutzer.



Zweite Rotation – Die Rotationsfunktion aktualisiert das Passwort für den geklonten Benutzer.



Lambda-Rotationsfunktionen

[the section called “Rotation durch Lambda-Funktion”](#) In einer Lambda-Funktion wird das Geheimnis rotiert. Secrets Manager verwendet [Staging-Bezeichnungen](#), um Secret-Versionen während der Drehung zu kennzeichnen.

Wenn Secrets Manager keine [Rotationsfunktionsvorlage](#) für Ihren Geheimtyp bereitstellt, können Sie eine Rotationsfunktion erstellen. Folgen Sie beim Schreiben einer Rotationsfunktion den Anweisungen für jeden Schritt.

Tipps zum Schreiben Ihrer eigenen Rotationsfunktion

- Verwenden Sie die [generische Rotationsvorlage](#) als Ausgangspunkt, um Ihre eigene Rotationsfunktion zu schreiben.
- Wenn Sie Ihre Funktion schreiben, sollten Sie bei den Debugging- oder Protokollierungsanweisungen vorsichtig sein. Diese Anweisungen können dazu führen, dass

Informationen in Ihrer Funktion an Amazon geschrieben werden. Sie müssen also sicherstellen CloudWatch, dass das Protokoll keine sensiblen Informationen enthält, die während der Entwicklung gesammelt wurden.

Beispiele für Protokollanweisungen finden Sie im [the section called “Rotationsfunktionsvorlagen”](#)-Quellcode.

- Aus Sicherheitsgründen erlaubt Secrets Manager nur eine Lambda-Rotationsfunktion, um das Geheimnis direkt zu rotieren. Die Rotationsfunktion kann keine zweite Lambda-Funktion aufrufen, um das Geheimnis zu rotieren.
- Vorschläge zum Debuggen finden Sie unter [Testen und Debuggen von Serverless Anwendungen](#).
- Wenn Sie externe Binärdateien und Bibliotheken verwenden, um beispielsweise eine Verbindung zu einer Ressource herzustellen, müssen Sie sich darum kümmern, diese zu patchen und zu behalten. up-to-date
- Speichern Sie Ihre Drehungsfunktion in einer ZIP-Datei *my-function.zip* zusammen mit allen erforderlichen Abhängigkeiten.

Vier Schritte in einer Rotationsfunktion

Themen

- [create_secret: Erstelle eine neue Version des Secrets](#)
- [set_secret: Ändern Sie die Anmeldeinformationen in der Datenbank oder im Service](#)
- [test_secret: Testen Sie die neue geheime Version](#)
- [finish_secret: Beende die Rotation](#)

create_secret: Erstelle eine neue Version des Secrets

Die Methode prüft `create_secret` zunächst, ob ein Geheimnis existiert, indem sie [get_secret_value](#) mit dem übergebenen aufruft. `ClientRequestToken` Wenn es kein Geheimnis gibt, erstellt sie ein neues Geheimnis mit [create_secret](#) und dem Token als `VersionId` Dann generiert es einen neuen geheimen Wert mit [get_random_password](#). Als Nächstes wird es aufgerufen [put_secret_value](#), um es mit dem Staging-Label `AWSPENDING` zu speichern. Das Speichern des neuen geheimen Werts in `AWSPENDING` trägt zur Sicherstellung der Idempotenz bei. Wenn die Drehung aus irgendeinem Grund fehlschlägt, können Sie in nachfolgenden Aufrufen auf diesen geheimen Wert verweisen. Weitere Informationen finden Sie unter [How do I make my Lambda function idempotent](#) (Wie mache ich meine Lambda-Funktion idempotent?).

Tipps zum Schreiben Ihrer eigenen Rotationsfunktion

- Stellen Sie sicher, dass der neue geheime Wert nur Zeichen enthält, die für die Datenbank oder den Dienst gültig sind. Schließen Sie Zeichen mithilfe des `ExcludeCharacters`-Parameters aus.
- Verwenden Sie beim Testen Ihrer Funktion die Versionsstufen, AWS CLI um sich die Versionsstufen anzusehen: aufrufen [describe-secret](#) und ansehen `VersionIdsToStages`.
- Für Amazon RDS MySQL erstellt Secrets Manager in abwechselnder Benutzerrotation einen geklonten Benutzer mit einem Namen, der nicht länger als 16 Zeichen ist. Sie können die Rotationsfunktion ändern, um längere Benutzernamen zuzulassen. MySQL Version 5.7 und höher unterstützt Benutzernamen mit bis zu 32 Zeichen. Secrets Manager fügt jedoch „_clone“ (sechs Zeichen) an das Ende des Benutzernamens an, sodass Sie den Benutzernamen auf maximal 26 Zeichen beschränken müssen.

set_secret: Ändern Sie die Anmeldeinformationen in der Datenbank oder im Service

Die Methode `set_secret` ändert die Anmeldeinformationen in der Datenbank oder im Dienst so, dass sie dem neuen geheimen Wert in der `AWSPENDING` Version des Geheimnisses entsprechen.

Tipps zum Schreiben Ihrer eigenen Rotationsfunktion

- Wenn Sie Anweisungen an einen Dienst übergeben, der Anweisungen interpretiert, z. B. an eine Datenbank, verwenden Sie die Abfrageparametrisierung. Weitere Informationen finden Sie im [Cheat Sheet zur Abfrageparametrisierung auf der OWASP-Website](#).
- Die Drehungsfunktion ist ein privilegierter Stellvertreter, der berechtigt ist, auf Kundenanmeldeinformationen sowohl im Secrets-Manager-Secret als auch in der Zielressource zuzugreifen und diese zu ändern. Um einen möglichen [verwirrten Stellvertreterangriff](#) zu verhindern, müssen Sie sicherstellen, dass ein Angreifer die Funktion nicht verwenden kann, um auf andere Ressourcen zuzugreifen. Bevor Sie die Anmeldeinformationen aktualisieren:
 - Überprüfen Sie, ob die Anmeldeinformationen in der `AWSCURRENT`-Version des Secrets gültig sind. Wenn die `AWSCURRENT`-Anmeldeinformationen nicht gültig sind, brechen Sie den Drehungsversuch ab.
 - Stellen Sie sicher, dass die geheimen Werte `AWSPENDING` und `AWSCURRENT` für dieselbe Ressource gelten. Überprüfen Sie für einen Benutzernamen und ein Passwort, ob die `AWSCURRENT`- und `AWSPENDING`-Benutzernamen identisch sind.
 - Stellen Sie sicher, dass die Ziel-Service-Ressource identisch ist. Überprüfen Sie bei einer Datenbank, ob die Hostnamen `AWSCURRENT` und `AWSPENDING` identisch sind.

- In seltenen Fällen möchten Sie möglicherweise eine vorhandene Rotationsfunktion für eine Datenbank anpassen. Bei abwechselnder Benutzerrotation erstellt Secrets Manager beispielsweise den geklonten Benutzer, indem er die [Laufzeitkonfigurationsparameter](#) des ersten Benutzers kopiert. Wenn Sie mehr Attribute hinzufügen oder ändern möchten, welche dem geklonten Benutzer gewährt werden, müssen Sie den Code in der `set_secret`-Funktion aktualisieren.

test_secret: Testen Sie die neue geheime Version

Als Nächstes testet die Lambda-Rotationsfunktion die `AWSPENDING`-Version des Secrets, indem es für den Zugriff auf die Datenbank oder den Service verwendet wird. Rotationsfunktionen basierend auf Folgendem: [Rotationsfunktionsvorlagen](#) testen das neue Secret mit Lesezugriff.

finish_secret: Beende die Rotation

Schließlich verschiebt die Lambda-Rotationsfunktion das Label `AWSCURRENT` von der vorherigen Secret-Version in diese Version, wodurch auch das `AWSPENDING`-Label im selben API-Aufruf entfernt wird. Secrets Manager fügt der vorherigen Version die `AWSPREVIOUS`-Staging-Bezeichnung hinzu, sodass Sie die letzte als funktionierend bekannte Version des Secret behalten.

Die Methode `finish_secret` verwendet [update_secret_version_stage](#), um das Staging-Label `AWSCURRENT` von der vorherigen geheimen Version in die neue geheime Version zu verschieben. Secrets Manager fügt der vorherigen Version die `AWSPREVIOUS`-Staging-Bezeichnung automatisch hinzu, sodass Sie die letzte als funktionierend bekannte Version des Secret behalten.

Tipps zum Schreiben Ihrer eigenen Rotationsfunktion

- Entfernen Sie es nicht `AWSPENDING` vor diesem Punkt und entfernen Sie es nicht mithilfe eines separaten API-Aufrufs, da dies Secrets Manager anzeigen kann, dass die Rotation nicht erfolgreich abgeschlossen wurde. Secrets Manager fügt der vorherigen Version die `AWSPREVIOUS`-Staging-Bezeichnung hinzu, sodass Sie die letzte als funktionierend bekannte Version des Secret behalten.

Wenn die Rotation erfolgreich ist, wird die `AWSPENDING`-Staging-Bezeichnung möglicherweise an dieselbe Version wie die `AWSCURRENT`-Version angehängt oder wird möglicherweise keiner Version zugeordnet. Wenn die `AWSPENDING`-Staging-Bezeichnung zwar vorhanden, aber nicht derselben Version zugeordnet ist wie `AWSCURRENT`, wird bei jedem späteren Aufruf der Rotation davon ausgegangen, dass eine vorherige Rotationsanforderung noch in Bearbeitung ist, und es wird ein Fehler zurückgegeben. Wenn die Rotation nicht erfolgreich ist, wird die `AWSPENDING`-Staging-

Bezeichnung möglicherweise an eine leere Secret-Version angehängt. Weitere Informationen finden Sie unter [Fehlerbehebung bei der -Rotation](#).

AWS Secrets Manager Vorlagen für Rotationsfunktionen

Denn [the section called “Rotation durch Lambda-Funktion”](#) Secrets Manager bietet eine Reihe von Vorlagen für Rotationsfunktionen. Informationen zur Verwendung der Vorlagen finden Sie unter:

- [Automatische Rotierung für Datenbank-Secrets \(Konsole\)](#)
- [Automatische Rotation für Nicht-Datenbankgeheimnisse \(Konsole\)](#)

Die Vorlagen unterstützen Python 3.9.

Informationen zum Schreiben Ihrer eigenen Rotationsfunktion finden [Sie unter Schreiben einer Rotationsfunktion](#).

Vorlagen

- [Amazon RDS und Amazon Aurora](#)
 - [Amazon RDS Db2: Einzelbenutzer](#)
 - [Amazon RDS Db2: Wechselnde Benutzer](#)
 - [Amazon RDS MariaDB – Einzelbenutzer](#)
 - [Amazon RDS MariaDB – wechselnde Benutzer](#)
 - [Amazon RDS und Amazon Aurora MySQL: Einzelbenutzer](#)
 - [Amazon RDS und Amazon Aurora MySQL: Wechselnde Benutzer](#)
 - [Amazon RDS Oracle – Einzelbenutzer](#)
 - [Amazon RDS Oracle – wechselnde Benutzer](#)
 - [Amazon RDS und Amazon Aurora PostgreSQL: Einzelbenutzer](#)
 - [Amazon RDS und Amazon Aurora PostgreSQL: Wechselnde Benutzer](#)
 - [Amazon RDS Microsoft SQLServer – Einzelbenutzer](#)
 - [Amazon RDS Microsoft SQLServer – wechselnde Benutzer](#)
- [Amazon DocumentDB \(mit MongoDB-Kompatibilität\)](#)
 - [Amazon-DocumentDB-Einzelbenutzer](#)
 - [Amazon DocumentDB – wechselnde Benutzer](#)
- [Amazon-Redshift](#)

- [Amazon Redshift – Einzelbenutzer](#)
- [Amazon Redshift – wechselnde Benutzer](#)
- [Amazon ElastiCache](#)
- [Active Directory](#)
 - [Active Directory-Anmeldeinformationen](#)
 - [Active Directory-Schlüsseltabelle](#)
- [Andere Arten von Secrets](#)

Amazon RDS und Amazon Aurora

Amazon RDS Db2: Einzelbenutzer

- Name der Vorlage: SecretsManager RdsDB2 RotationSingleUser
- Drehungsstrategie: [Rotationsstrategie: Einzelbenutzer](#).
- **SecretString**-Struktur: [the section called “Secret-Struktur von Amazon RDS Db2”](#).
- Quellcode: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/rdsDB2/lambda_function.py SecretsManager RotationSingleUser
- Abhängigkeit: [python-ibmdb](#)

Amazon RDS Db2: Wechselnde Benutzer

- Name der SecretsManager Vorlage: RdsDB2 RotationMultiUser
- Drehungsstrategie: [the section called “Wechselnde Benutzer”](#).
- **SecretString**-Struktur: [the section called “Secret-Struktur von Amazon RDS Db2”](#).
- Quellcode: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/rdsDB2/lambda_function.py SecretsManager RotationMultiUser
- Abhängigkeit: [python-ibmdb](#)

Amazon RDS MariaDB – Einzelbenutzer

- Name der SecretsManager Vorlage: rdsmariaDB RotationSingleUser
- Drehungsstrategie: [Rotationsstrategie: Einzelbenutzer](#).
- **SecretString**-Struktur: [the section called “Amazon RDS MariaDB – Secret-Struktur”](#).

- Quellcode: https://github.com/aws-samples/-lambdas/tree/master/rdsmariaDB/lambda_function.py aws-secrets-manager-rotation SecretsManager RotationSingleUser
- AbhängigkeitPyMy: SQL 1.0.2. Wenn Sie das SHA256-Passwort für die Authentifizierung verwenden, PyMy SQL [rsa]. Informationen zur Verwendung von Paketen mit kompiliertem Code in einer Lambda-Laufzeit finden Sie unter [Wie füge ich Python-Pakete mit kompilierten Binärdateien zu meinem Bereitstellungspaket hinzu und mache das Paket mit Lambda kompatibel?](#) im AWS Knowledge Center.

Amazon RDS MariaDB – wechselnde Benutzer

- Name der Vorlage: SecretsManager rdsmariaDB RotationMultiUser
- Drehungsstrategie: [the section called “Wechselnde Benutzer”](#).
- **SecretString**-Struktur: [the section called “Amazon RDS MariaDB – Secret-Struktur”](#).
- Quellcode: https://github.com/aws-samples/-lambdas/tree/master/rdsmariaDB/lambda_function.py aws-secrets-manager-rotation SecretsManager RotationMultiUser
- AbhängigkeitPyMy: SQL 1.0.2. Wenn Sie das SHA256-Passwort für die Authentifizierung verwenden, PyMy SQL [rsa]. Informationen zur Verwendung von Paketen mit kompiliertem Code in einer Lambda-Laufzeit finden Sie unter [Wie füge ich Python-Pakete mit kompilierten Binärdateien zu meinem Bereitstellungspaket hinzu und mache das Paket mit Lambda kompatibel?](#) im AWS Knowledge Center.

Amazon RDS und Amazon Aurora MySQL: Einzelbenutzer

- Name der Vorlage: SecretsManager RDSMySQL RotationSingleUser
- Drehungsstrategie: [the section called “Einzelbenutzer”](#).
- Erwartete **SecretString**-Struktur: [the section called “Secret-Struktur von Amazon RDS und Amazon Aurora MySQL”](#).
- Quellcode: https://github.com/aws-samples/aws-secrets-manager-rotation/-lambdas/tree/master/rdsmySQL/lambda_function.py SecretsManager RotationSingleUser
- AbhängigkeitPyMy: SQL 1.0.2. Wenn Sie das SHA256-Passwort für die Authentifizierung verwenden, PyMy SQL [rsa]. Informationen zur Verwendung von Paketen mit kompiliertem Code in einer Lambda-Laufzeit finden Sie unter [Wie füge ich Python-Pakete mit kompilierten Binärdateien zu meinem Bereitstellungspaket hinzu und mache das Paket mit Lambda kompatibel?](#) im AWS Knowledge Center.

Amazon RDS und Amazon Aurora MySQL: Wechselnde Benutzer

- Name der Vorlage: SecretsManager RDSMySQL RotationMultiUser
- Drehungsstrategie: [the section called “Wechselnde Benutzer”](#).
- Erwartete **SecretString**-Struktur: [the section called “Secret-Struktur von Amazon RDS und Amazon Aurora MySQL”](#).
- Quellcode: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/rdsmySQL/lambda_function.py SecretsManager RotationMultiUser
- AbhängigkeitPyMy: SQL 1.0.2. Wenn Sie das SHA256-Passwort für die Authentifizierung verwenden, PyMy SQL [rsa]. Informationen zur Verwendung von Paketen mit kompiliertem Code in einer Lambda-Laufzeit finden Sie unter [Wie füge ich Python-Pakete mit kompilierten Binärdateien zu meinem Bereitstellungspaket hinzu und mache das Paket mit Lambda kompatibel?](#) im AWS Knowledge Center.

Amazon RDS Oracle – Einzelbenutzer

- Name der Vorlage: SecretsManager RDS OracleRotationSingleUser
- Drehungsstrategie: [the section called “Einzelbenutzer”](#).
- Erwartete **SecretString**-Struktur: [the section called “Amazon RDS Oracle – Secret-Struktur”](#).
- Quellcode: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/RDS/lambda_function.py OracleRotationSingleUser
- Abhängigkeit: [python-oracledb](#) 2.0.1

Amazon RDS Oracle – wechselnde Benutzer

- Name der Vorlage: RDS SecretsManager OracleRotationMultiUser
- Drehungsstrategie: [the section called “Wechselnde Benutzer”](#).
- Erwartete **SecretString**-Struktur: [the section called “Amazon RDS Oracle – Secret-Struktur”](#).
- Quellcode: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/RDS/lambda_function.py OracleRotationMultiUser
- Abhängigkeit: [python-oracledb](#) 2.0.1

Amazon RDS und Amazon Aurora PostgreSQL: Einzelbenutzer

- Name der Vorlage: RDSPostgreSQL SecretsManager RotationSingleUser
- Drehungsstrategie: [Rotationsstrategie: Einzelbenutzer](#).
- Erwartete **SecretString**-Struktur: [the section called “Secret-Struktur von Amazon RDS und Amazon Aurora PostgreSQL”](#).
- Quellcode: https://github.com/aws-samples/-lambdas/tree/master/rdspostgresql/lambda_function.py aws-secrets-manager-rotation SecretsManager RotationSingleUser
- AbhängigkeitPyGre: SQL 5.0.7

Amazon RDS und Amazon Aurora PostgreSQL: Wechselnde Benutzer

- Name der Vorlage: RDSPostgreSQL SecretsManager RotationMultiUser
- Drehungsstrategie: [the section called “Wechselnde Benutzer”](#).
- Erwartete **SecretString**-Struktur: [the section called “Secret-Struktur von Amazon RDS und Amazon Aurora PostgreSQL”](#).
- Quellcode: https://github.com/aws-samples/-lambdas/tree/master/rdspostgresql/lambda_function.py aws-secrets-manager-rotation SecretsManager RotationMultiUser
- AbhängigkeitPyGre: SQL 5.0.7

Amazon RDS Microsoft SQLServer – Einzelbenutzer

- Name der Vorlage: RDSSQL SecretsManager ServerRotationSingleUser
- Drehungsstrategie: [the section called “Einzelbenutzer”](#).
- Erwartete **SecretString**-Struktur: [the section called “Amazon RDS Microsoft SQLServer – Secret-Struktur”](#).
- Quellcode: https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/RDSSQL/lambda_function.py SecretsManager -
- Abhängigkeit: Pymssql 2.2.2

Amazon RDS Microsoft SQLServer – wechselnde Benutzer

- Name der SecretsManager Vorlage: RDSSQL ServerRotationMultiUser
- Drehungsstrategie: [the section called “Wechselnde Benutzer”](#).

- Erwartete **SecretString**-Struktur: [the section called “Amazon RDS Microsoft SQLServer – Secret-Struktur”](#).
- Quellcode: https://github.com/aws-samples/aws-secrets-manager-rotation-SecretsManager-lambdas/tree/master/RDSSQL/lambda_function.py ServerRotationMultiUser
- Abhängigkeit: Pymssql 2.2.2

Amazon DocumentDB (mit MongoDB-Kompatibilität)

Amazon-DocumentDB-Einzelbenutzer

- Name SecretsManagerMongo der Vorlage: DB RotationSingleUser
- Drehungsstrategie: [the section called “Einzelbenutzer”](#).
- Erwartete **SecretString**-Struktur: [the section called “Amazon DocumentDB – Secret-Struktur”](#).
- Quellcode: https://github.com/aws-samples/aws-secrets-manager-rotation-SecretsManagerMongo-lambdas/tree/master/DB/lambda_function.py RotationSingleUser
- Abhängigkeit: Pymongo 3.2

Amazon DocumentDB – wechselnde Benutzer

- Name SecretsManagerMongo der Vorlage: DB RotationMultiUser
- Drehungsstrategie: [the section called “Wechselnde Benutzer”](#).
- Erwartete **SecretString**-Struktur: [the section called “Amazon DocumentDB – Secret-Struktur”](#).
- Quellcode: https://github.com/aws-samples/aws-secrets-manager-rotation-SecretsManagerMongo-lambdas/tree/master/DB/lambda_function.py RotationMultiUser
- Abhängigkeit: Pymongo 3.2

Amazon-Redshift

Amazon Redshift – Einzelbenutzer

- Name der Vorlage: SecretsManagerRedshiftRotationSingleUser
- Drehungsstrategie: [the section called “Einzelbenutzer”](#).
- Erwartete **SecretString** Struktur: [the section called “Amazon Redshift – Secret-Struktur”](#) oder [the section called “Geheime Struktur von Amazon Redshift Serverless”](#).

- Quellcode: [https://github.com/aws-samples/ aws-secrets-manager-rotation SecretsManagerRedshiftRotationSingleUser](https://github.com/aws-samples/aws-secrets-manager-rotation/SecretsManagerRedshiftRotationSingleUser) -lambdas/tree/master/ /lambda_function.py
- AbhängigkeitPyGre: SQL 5.0.7

Amazon Redshift – wechselnde Benutzer

- Name der Vorlage: SecretsManagerRedshiftRotationMultiUser
- Drehungsstrategie: [the section called “Wechselnde Benutzer”](#).
- Erwartete **SecretString** Struktur: [the section called “Amazon Redshift – Secret-Struktur”](#) oder [the section called “Geheime Struktur von Amazon Redshift Serverless”](#).
- Quellcode: [https://github.com/aws-samples/ aws-secrets-manager-rotation SecretsManagerRedshiftRotationMultiUser](https://github.com/aws-samples/aws-secrets-manager-rotation/SecretsManagerRedshiftRotationMultiUser) -lambdas/tree/master/ /lambda_function.py
- AbhängigkeitPyGre: SQL 5.0.7

Amazon ElastiCache

Informationen zur Verwendung dieser Vorlage finden Sie unter [Automatisches Rotieren von Passwörtern für Benutzer](#) im ElastiCache Amazon-Benutzerhandbuch.

- Name der Vorlage: SecretsManagerElasticacheUserRotation
- Erwartete **SecretString**-Struktur: [the section called “ElastiCache Geheime Struktur von Amazon”](#).
- Quellcode: [https://github.com/aws-samples/ aws-secrets-manager-rotation SecretsManagerElasticacheUserRotation](https://github.com/aws-samples/aws-secrets-manager-rotation/SecretsManagerElasticacheUserRotation) -lambdas/tree/master/ /lambda_function.py

Active Directory

Active Directory-Anmeldeinformationen

- Name der Vorlage: SecretsManagerActiveDirectoryRotationSingleUser
- Erwartete **SecretString**-Struktur: [the section called “Struktur der geheimen Active Directory-Anmeldeinformationen”](#).
- Quellcode: [https://github.com/aws-samples/ aws-secrets-manager-rotation SecretsManagerActiveDirectoryRotationSingleUser](https://github.com/aws-samples/aws-secrets-manager-rotation/SecretsManagerActiveDirectoryRotationSingleUser) -lambdas/tree/master/ /lambda_function.py

Active Directory-Schlüsseltabelle

- Name der Vorlage: SecretsManagerActiveDirectoryAndKeytabRotationSingleUser
- Erwartete **SecretString**-Struktur: [the section called “Geheime Active Directory-Strukturen”](#).
- Quellcode: [https://github.com/aws-samples/ aws-secrets-manager-rotation SecretsManagerActiveDirectoryAndKeytabRotationSingleUser -lambdas/tree/master/ / lambda_function.py](https://github.com/aws-samples/aws-secrets-manager-rotation/SecretsManagerActiveDirectoryAndKeytabRotationSingleUser-lambdas/tree/master/lambda_function.py)
- Abhängigkeiten: mskutil

Andere Arten von Secrets

Secrets Manager stellt diese Vorlage als Ausgangspunkt bereit, um eine Rotationsfunktion für jede Art von Geheimnis zu erstellen.

- Name der Vorlage: SecretsManagerRotationTemplate
- Quellcode: [https://github.com/aws-samples/ aws-secrets-manager-rotation SecretsManagerRotationTemplate -lambdas/tree/master/ /lambda_function.py](https://github.com/aws-samples/aws-secrets-manager-rotation/SecretsManagerRotationTemplate-lambdas/tree/master/lambda_function.py)

Rollenberechtigungen für die Ausführung der Lambda-Rotationsfunktion für AWS Secrets Manager

Denn wenn Secrets Manager eine Lambda-Funktion verwendet [the section called “Rotation durch Lambda-Funktion”](#), um ein Geheimnis zu rotieren, nimmt Lambda eine [IAM-Ausführungsrolle](#) an und stellt diese Anmeldeinformationen für den Lambda-Funktionscode bereit. Anweisungen zum Einrichten der automatischen Rotation finden Sie unter:

- [Automatische Rotierung für Datenbank-Secrets \(Konsole\)](#)
- [Automatische Rotation für Nicht-Datenbankgeheimnisse \(Konsole\)](#)
- [Automatische Drehung \(AWS CLI\)](#)

Die folgenden Beispiele zeigen Inline-Richtlinien für Ausführungsrollen von Lambda-Drehungsfunktionen. Informationen zum Erstellen einer Ausführungsrolle und Anfügen einer Berechtigungsrichtlinie finden Sie unter [AWS Lambda -Ausführungsrolle](#).

Beispiele:

- [Richtlinie für eine Lambda-Drehungsfunktion und Ausführungsrolle](#)
- [Richtlinienanweisung für einen kundenverwalteten Schlüssel](#)
- [Richtlinienanweisung für die Strategie für wechselnde Benutzer](#)

Richtlinie für eine Lambda-Drehungsfunktion und Ausführungsrolle

Die folgende Beispielrichtlinie erlaubt der Drehungsfunktion folgende Tätigkeiten:

- Ausführen von Secrets-Manager-Vorgängen für *SecretARN*.
- Erstellen eines Passworts.
- Einrichten der erforderlichen Konfiguration, wenn Ihre Datenbank oder Ihr Service in einer VPC ausgeführt wird. Siehe [Konfigurieren einer Lambda-Funktion für den Zugriff auf Ressourcen in einer VPC](#).

```
{
  "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"
  }
]
}

```

Richtlinienanweisung für einen kundenverwalteten Schlüssel

Wenn das Secret mit einem anderen KMS-Schlüssel als den Von AWS verwalteter Schlüssel `aws/secretsmanager` verschlüsselt wird, müssen Sie der Lambda-Ausführungsrolle die Berechtigung erteilen, den Schlüssel zu verwenden. Sie können den [SecretARN-Verschlüsselungskontext](#) verwenden, um die Verwendung der Entschlüsselungsfunktion einzuschränken, sodass die Rolle der Rotationsfunktion nur Zugriff auf das Secret hat, für dessen Rotation diese verantwortlich ist. Das folgende Beispiel zeigt eine Anweisung, die zur Ausführungsrollenrichtlinie hinzugefügt werden soll, um das Secret mithilfe des KMS-Schlüssels zu entschlüsseln.

```

{
  "Effect": "Allow",
  "Action": [
    "kms:Decrypt",
    "kms:DescribeKey",
    "kms:GenerateDataKey"
  ],
  "Resource": "KMSKeyARN"
  "Condition": {
    "StringEquals": {
      "kms:EncryptionContext:SecretARN": "SecretARN"
    }
  }
}

```

Um die Rotationsfunktion für mehrere Secrets zu verwenden, die mit einem vom Kunden verwalteten Schlüssel verschlüsselt sind, fügen Sie eine Anweisung wie im folgenden Beispiel hinzu, damit die Ausführungsrolle das Secret entschlüsseln kann.

```

{
  "Effect": "Allow",
  "Action": [
    "kms:Decrypt",
    "kms:DescribeKey",

```

```

        "kms:GenerateDataKey"
    ],
    "Resource": "KMSKeyARN"
    "Condition": {
        "StringEquals": {
            "kms:EncryptionContext:SecretARN": [
                "arn1",
                "arn2"
            ]
        }
    }
}

```

Richtlinienanweisung für die Strategie für wechselnde Benutzer

Weitere Informationen zu Drehungsstrategie für alternierende Benutzer finden Sie unter [the section called “Strategien zur Rotation von Lambda-Funktionen”](#).

Wenn Sie für ein Secret, das Amazon-RDS-Anmeldeinformationen enthält, die Strategie für wechselnde Benutzer verwenden und das Superuser-Geheimnis [von Amazon RDS verwaltet](#) wird, müssen Sie der Rotationsfunktion auch erlauben, schreibgeschützte APIs auf Amazon RDS aufzurufen, damit sie die Verbindungsinformationen für die Datenbank abrufen kann. Wir empfehlen Ihnen, die AWS verwaltete Richtlinie [AmazonRDS ReadOnlyAccess](#) beizufügen.

Die folgende Beispielrichtlinie erlaubt der Funktion folgende Tätigkeiten:

- Ausführen von Secrets-Manager-Vorgängen für *SecretARN*.
- Abrufen von Anmeldeinformationen im Superuser-Secret. Secrets Manager verwendet die Anmeldeinformationen im Superuser-Secret, um die Anmeldeinformationen im gedrehten Secret zu aktualisieren.
- Erstellen eines Passworts.
- Einrichten der erforderlichen Konfiguration, wenn Ihre Datenbank oder Ihr Service in einer VPC ausgeführt wird. Weitere Informationen finden Sie unter [Konfigurieren einer Lambda-Funktion für den Zugriff auf Ressourcen in einer VPC](#).

```

{
  "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"
}
]
}

```

Netzwerkzugriff für die Lambda-Rotationsfunktion

Denn wenn Secrets Manager eine Lambda-Funktion verwendet [the section called “Rotation durch Lambda-Funktion”](#), um ein Geheimnis zu rotieren, muss die Lambda-Rotationsfunktion auf das Geheimnis zugreifen können. Wenn Ihr Secret Anmeldeinformationen enthält, muss die Lambda-

Funktion auch auf die Quelle dieser Anmeldeinformationen zugreifen können, z. B. auf eine Datenbank oder einen Service.

Greifen Sie wie folgt auf ein Secret zu:

Die Lambda-Drehungsfunktion muss auf einen Secrets-Manager-Endpunkt zugreifen können. Wenn Ihre Lambda-Funktion auf das Internet zugreifen kann, können Sie einen öffentlichen Endpunkt verwenden. Informationen zum Suchen nach einem Endpunkt finden Sie unter [the section called “Secrets-Manager-Endpunkte”](#).

Wenn Ihre Lambda-Funktion in einer VPC ausgeführt wird, die keinen Internetzugang hat, empfehlen wir Ihnen, private Endpunkte des Secrets-Manager-Services in Ihrer VPC zu konfigurieren. Ihre VPC kann dann Anfragen abfangen, die an den öffentlichen regionalen Endpunkt gerichtet sind und sie an den privaten Endpunkt umleiten. Weitere Informationen finden Sie unter [VPC-Endpunkt](#).

Sie können alternativ Ihre Lambda-Funktion so konfigurieren, dass sie auf den öffentlichen Secrets-Manager-Endpunkt zugreifen kann, indem Sie ein [NAT-Gateway](#) oder ein [Internet-Gateway](#) zu Ihrer VPC hinzufügen. Auf diese Weise kann Datenverkehr von Ihrer VPC den öffentlichen Endpunkt erreichen. Für Ihre VPC ergibt sich dabei ein Risiko, da es eine IP-Adresse für das Gateway gibt, die aus dem öffentlichen Internet angegriffen werden kann.

(Optional) Greifen Sie auf die Datenbank oder den Service wie folgt zu:

Für Secrets wie API-Schlüssel gibt es keine Quelldatenbank oder -Service, den Sie zusammen mit dem Secret aktualisieren müssen.

Wenn Ihre Datenbank oder Ihr Service auf einer Amazon-EC2-Instance in einer VPC ausgeführt wird, empfehlen wir, dass Sie die Ausführung Ihrer Lambda-Funktion für dieselbe VPC konfigurieren. Dann kann die Drehungsfunktion direkt mit Ihrem Service kommunizieren. Weitere Informationen finden Sie unter [Konfigurieren des VPC-Zugriffs](#).

Um der Lambda-Funktion den Zugriff auf die Datenbank oder den Service zu ermöglichen, müssen Sie sicherstellen, dass die Sicherheitsgruppen, die an Ihre Lambda-Drehungsfunktion angeschlossen sind, ausgehende Verbindungen zur Datenbank oder zum Service zulassen. Sie müssen darüber hinaus sicherstellen, dass die Sicherheitsgruppen, die an Ihre Datenbank oder Ihren Service angefügt sind, eingehende Verbindungen von der Lambda-Drehungsfunktion zulassen.

Fehlerbehebung bei der AWS Secrets Manager Rotation

Für viele Services verwendet Secrets Manager eine Lambda-Funktion, um Secrets zu rotieren. Weitere Informationen finden Sie unter [the section called "Rotation durch Lambda-Funktion"](#). Die Lambda-Drehungsfunktion interagiert mit der Datenbank oder dem Service, für den das Secret bestimmt ist, sowie mit dem Secrets Manager. Wenn die Rotation nicht wie erwartet funktioniert, sollten Sie zunächst die CloudWatch Protokolle überprüfen.

Note

Einige Services können Services für Sie verwalten, einschließlich der Verwaltung der automatischen Rotation. Weitere Informationen finden Sie unter [the section called "Verwaltete Rotation"](#).

So zeigen Sie die CloudWatch Protokolle für Ihre Lambda-Funktion an

1. Öffnen Sie die Secrets-Manager-Konsole unter <https://console.aws.amazon.com/secretsmanager/>.
2. Wählen Sie Ihr Secret aus und dann auf der Detailseite unter Rotation configuration (Drehungs-Konfiguration) wählen Sie die Lambda-Drehungsfunktion aus. Die Lambda-Konsole wird geöffnet.
3. Wählen Sie auf der Registerkarte Monitor die Option Logs und anschließend View logs in CloudWatch aus.

Die CloudWatch Konsole wird geöffnet und zeigt die Protokolle für Ihre Funktion an.

Interpretieren Sie die Protokolle wie folgt:

- [Keine Aktivität nach „Anmeldeinformationen in Umgebungsvariablen gefunden“](#)
- [Keine Aktivität nach „createSecret“](#)
- [Fehler: „Zugriff auf KMS ist nicht zulässig“](#)
- [Fehler: „Key is missing from secret JSON“ \(Schlüssel fehlt in Secret-JSON\)](#)
- [Fehler: „setSecret: Unable to log into database“ \(setSecret: Anmeldung in Datenbank nicht möglich\)](#)
- [Fehler: „Modul ‚lambda_function‘ konnte nicht importiert werden“](#)

- [Eine bestehende Rotationsfunktion von Python 3.7 auf 3.9 aktualisieren](#)

Keine Aktivität nach „Anmeldeinformationen in Umgebungsvariablen gefunden“

Wenn nach „Anmeldeinformationen in Umgebungsvariablen gefunden“ keine Aktivität vorhanden ist und die Aufgabendauer lang ist, z. B. das standardmäßige Lambda-Timeout von 30 000 ms, kann es bei der Lambda-Funktion zu einem Timeout kommen, während versucht wird, den Secrets-Manager-Endpunkt zu erreichen.

Die Lambda-Drehungsfunktion muss auf einen Secrets-Manager-Endpunkt zugreifen können. Wenn Ihre Lambda-Funktion auf das Internet zugreifen kann, können Sie einen öffentlichen Endpunkt verwenden. Informationen zum Suchen nach einem Endpunkt finden Sie unter [the section called “Secrets-Manager-Endpunkte”](#).

Wenn Ihre Lambda-Funktion in einer VPC ausgeführt wird, die keinen Internetzugang hat, empfehlen wir Ihnen, private Endpunkte des Secrets-Manager-Services in Ihrer VPC zu konfigurieren. Ihre VPC kann dann Anfragen abfangen, die an den öffentlichen regionalen Endpunkt gerichtet sind und sie an den privaten Endpunkt umleiten. Weitere Informationen finden Sie unter [VPC-Endpunkt](#).

Sie können alternativ Ihre Lambda-Funktion so konfigurieren, dass sie auf den öffentlichen Secrets-Manager-Endpunkt zugreifen kann, indem Sie ein [NAT-Gateway](#) oder ein [Internet-Gateway](#) zu Ihrer VPC hinzufügen. Auf diese Weise kann Datenverkehr von Ihrer VPC den öffentlichen Endpunkt erreichen. Für Ihre VPC ergibt sich dabei ein Risiko, da es eine IP-Adresse für das Gateway gibt, die aus dem öffentlichen Internet angegriffen werden kann.

Keine Aktivität nach „createSecret“

Die folgenden Probleme können dazu führen, dass die Rotation nach createSecret gestoppt wird:

Die VPC-Netzwerk-ACLs lassen keinen ein- und ausgehenden HTTPS-Datenverkehr zu.

Weitere Informationen finden Sie unter [Kontrollieren des Datenverkehrs zu Subnetzen mithilfe von Netzwerk-ACLs](#) im Benutzerhandbuch von Amazon VPC.

Die Timeout-Konfiguration der Lambda-Funktion ist zu kurz, um die Aufgabe auszuführen.

Weitere Informationen finden Sie unter [Konfigurieren von Lambda-Funktionsoptionen](#) im AWS Lambda -Entwicklerhandbuch.

Der VPC-Endpunkt des Secrets Managers lässt die VPC-CIDRs beim Eintritt in die zugewiesenen Sicherheitsgruppen nicht zu.

Weitere Informationen finden Sie unter [Control traffic to resources using security groups](#) (Kontrollieren des Datenverkehrs zu Ressourcen mithilfe von Sicherheitsgruppen) im Benutzerhandbuch von Amazon VPC.

Der VPC-Endpunkt des Secrets Managers erlaubt Lambda nicht, den VPC-Endpunkt zu verwenden.

Weitere Informationen finden Sie unter [VPC-Endpunkt](#).

Das Secret verwendet die Rotation alternierender Benutzer, das Superuser-Secret wird von Amazon RDS verwaltet und die Lambda-Funktion kann nicht auf die RDS-API zugreifen.

Für eine [abwechselnde Benutzerrotation](#), bei der das Superuser-Geheimnis [von einem anderen AWS Dienst verwaltet](#) wird, muss die Lambda-Rotationsfunktion in der Lage sein, den Dienstendpunkt aufzurufen, um die Datenbankverbindungsinformationen abzurufen. Wir empfehlen die Konfiguration eines VPC-Endpunkts für den Datenbankservice. Weitere Informationen finden Sie hier:

- [Amazon RDS API und Schnittstellen-VPC-Endpunkte](#) im Amazon-RDS-Benutzerhandbuch.
- [Arbeiten mit VPC-Endpunkten](#) im Amazon-Redshift-Verwaltungshandbuch.

Fehler: „Zugriff auf KMS ist nicht zulässig“

Wenn Ihnen `ClientError: An error occurred (AccessDeniedException) when calling the GetSecretValue operation: Access to KMS is not allowed` angezeigt wird, ist die Rotationsfunktion nicht berechtigt, das Secret mit dem KMS-Schlüssel, der zur Verschlüsselung des Secrets verwendet wurde, zu entschlüsseln. Möglicherweise enthält die Berechtigungsrichtlinie eine Bedingung, die den Verschlüsselungskontext auf ein bestimmtes Secret beschränkt. Informationen zu den erforderlichen Berechtigungen finden Sie unter [the section called "Richtlinienanweisung für einen kundenverwalteten Schlüssel"](#).

Fehler: „Key is missing from secret JSON“ (Schlüssel fehlt in Secret-JSON)

Für eine Lambda-Rotationsfunktion muss sich der Secret-Wert in einer bestimmten JSON-Struktur befinden. Wenn Sie diesen Fehler sehen, fehlt dem JSON möglicherweise ein Schlüssel, auf den die Rotationsfunktion zugreifen wollte. Hinweise zur JSON-Struktur für die einzelnen Arten von Secrets finden Sie unter [the section called "JSON-Struktur eines Secrets"](#).

Fehler: „setSecret: Unable to log into database“ (setSecret: Anmeldung in Datenbank nicht möglich)

Die folgenden Probleme können diesen Fehler verursachen:

Die Rotationsfunktion kann nicht auf die Datenbank zugreifen.

Wenn die Aufgabendauer lang ist, z. B. über 5 000 ms, kann die Lambda-Rotationsfunktion möglicherweise nicht über das Netzwerk auf die Datenbank zugreifen.

Wenn Ihre Datenbank oder Ihr Service auf einer Amazon-EC2-Instance in einer VPC ausgeführt wird, empfehlen wir, dass Sie die Ausführung Ihrer Lambda-Funktion für dieselbe VPC konfigurieren. Dann kann die Drehungsfunktion direkt mit Ihrem Service kommunizieren. Weitere Informationen finden Sie unter [Konfigurieren des VPC-Zugriffs](#).

Um der Lambda-Funktion den Zugriff auf die Datenbank oder den Service zu ermöglichen, müssen Sie sicherstellen, dass die Sicherheitsgruppen, die an Ihre Lambda-Drehungsfunktion angeschlossen sind, ausgehende Verbindungen zur Datenbank oder zum Service zulassen. Sie müssen darüber hinaus sicherstellen, dass die Sicherheitsgruppen, die an Ihre Datenbank oder Ihren Service angefügt sind, eingehende Verbindungen von der Lambda-Drehungsfunktion zulassen.

Die Anmeldeinformationen im Secret sind falsch.

Wenn die Aufgabendauer kurz ist, kann sich die Lambda-Drehungsfunktion möglicherweise nicht mit den Anmeldeinformationen im Secret authentifizieren. Überprüfen Sie die Anmeldeinformationen, indem Sie sich mithilfe des Befehls manuell mit den Informationen in den `AWSPREVIOUS` Versionen `AWSCURRENT` und Versionen des Geheimnisses anmelden. AWS CLI [get-secret-value](#)

Die Datenbank verwendet **scram-sha-256** zum Verschlüsseln der Passwörter.

Wenn Ihre Datenbank Aurora PostgreSQL Version 13 oder höher ist und `scram-sha-256` zur Verschlüsselung verwendet, die Rotationsfunktion jedoch `libpq` Version 9 oder eine ältere Version verwendet, die `scram-sha-256` nicht unterstützt, kann die Rotationsfunktion keine Verbindung zur Datenbank herstellen.

Um festzustellen, welche Datenbankbenutzer **scram-sha-256**-Verschlüsselung verwenden

- Weitere Informationen finden Sie unter [Auf Benutzer prüfen, deren Passwörter nicht im SCRAM-Format sind](#) im Blog [SCRAM-Authentifizierung in RDS für PostgreSQL 13](#).

Um festzustellen, welche Version von **libpq** Ihre Rotationsfunktion verwendet

1. Navigieren Sie auf einem Linux-Computer in der Lambda-Konsole zu Ihrer Rotationsfunktion und laden Sie das Bereitstellungspaket herunter. Entpacken Sie die Zip-Datei in ein Arbeitsverzeichnis.
2. Führen Sie in einer Befehlszeile im Arbeitsverzeichnis Folgendes aus:

```
readelf -a libpq.so.5 | grep RUNPATH
```

3. Wenn Sie die Zeichenfolge *PostgreSQL-9.4.x* oder eine Hauptversion unter 10 sehen, unterstützt die Rotationsfunktion `scram-sha-256` nicht.

- Ausgabe für eine Rotationsfunktion, die `scram-sha-256` nicht unterstützt:

```
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]
```

- Ausgabe für eine Rotationsfunktion, die `scram-sha-256` unterstützt:

```
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

Wenn Sie die automatische Secret-Rotation vor dem 30. Dezember 2021 einrichten, wird Ihre in einer älteren Version von `libpq` verpackte Rotationsfunktion `scram-sha-256` nicht unterstützen. Um `scram-sha-256` zu unterstützen, müssen Sie [Ihre Rotationsfunktion neu erstellen](#).

Die Datenbank benötigt SSL/TLS-Zugriff.

Wenn Ihre Datenbank eine SSL/TLS-Verbindung benötigt, die Rotationsfunktion jedoch eine unverschlüsselte Verbindung verwendet, dann kann die Rotationsfunktion keine Verbindung zur Datenbank herstellen. Rotationsfunktionen für Amazon RDS (außer Oracle und Db2) und Amazon DocumentDB verwenden automatisch Secure Socket Layer (SSL) oder Transport Layer Security (TLS), um eine Verbindung zu Ihrer Datenbank herzustellen, wenn sie verfügbar ist. Andernfalls verwenden sie eine unverschlüsselte Verbindung.

Note

Wenn Sie die automatische Secret-Rotation vor dem 20. Dezember 2021 einrichten, basiert Ihre Rotationsfunktion möglicherweise auf einer älteren Vorlage, die SSL/TLS nicht unterstützt. Um Verbindungen zu unterstützen, die SSL/TLS verwenden, müssen Sie [Ihre Rotationsfunktion neu](#).

So bestimmen Sie, wann Ihre Rotationsfunktion erstellt wurde

1. Öffnen Sie die Secrets-Manager-Konsole unter <https://console.aws.amazon.com/secretsmanager/>, öffnen Sie dann Ihr Secret. Im Abschnitt Rotationskonfiguration sehen Sie unter Lambda-Rotationsfunktion den ARN der Lambda-Funktion, zum Beispiel `arn:aws:lambda:aws-region:123456789012:function:SecretsManagerMyRotationFunction`. Kopieren Sie in diesem Beispiel den Funktionsnamen vom Ende des ARN, z. B. `SecretsManagerMyRotationFunction`.
2. Fügen Sie in der AWS Lambda Konsole <https://console.aws.amazon.com/lambda/> unter Funktionen Ihren Lambda-Funktionsnamen in das Suchfeld ein, wählen Sie Enter und wählen Sie dann die Lambda-Funktion aus.
3. Kopieren Sie auf der Funktionsdetailseite auf der Registerkarte Konfiguration unter Tags, den Wert neben den Schlüssel `aws:cloudformation:stack-name`.
4. Fügen Sie in der AWS CloudFormation Konsole <https://console.aws.amazon.com/cloudformation> unter Stacks den Schlüsselwert in das Suchfeld ein und wählen Sie dann Enter.
5. Die Liste der Stacks wird so gefiltert, dass nur der Stack angezeigt wird, der die Lambda-Rotationsfunktion erstellt hat. In der Spalte Erstellungsdatum sehen Sie das Datum, an dem

der Stack erstellt wurde. Dies ist das Datum, an dem die Lambda-Rotationsfunktion erstellt wurde.

Fehler: „Modul ‚lambda_function‘ konnte nicht importiert werden“

Dieser Fehler kann auftreten, wenn Sie eine frühere Lambda-Funktion ausführen, die automatisch von Python 3.7 auf eine neuere Version von Python aktualisiert wurde. Um den Fehler zu beheben, können Sie die Version der Lambda-Funktion wieder auf Python 3.7 und dann auf [the section called „Eine bestehende Rotationsfunktion von Python 3.7 auf 3.9 aktualisieren“](#) ändern. Weitere Informationen finden Sie unter [Warum ist die Rotation meiner Secrets-Manager-Lambda-Funktion mit der Fehlermeldung „PG-Modul nicht gefunden“ fehlgeschlagen?](#) in AWS -re:Post.

Eine bestehende Rotationsfunktion von Python 3.7 auf 3.9 aktualisieren

Einige Rotationsfunktionen, die vor November 2022 erstellt wurden, verwendeten Python 3.7. Das AWS SDK für Python hat die Unterstützung von Python 3.7 im Dezember 2023 eingestellt. Weitere Informationen finden Sie unter [Aktualisierungen der Python-Supportrichtlinie für AWS SDKs und Tools](#). Um zu einer neuen Rotationsfunktion zu wechseln, die Python 3.9 verwendet, können Sie einer vorhandenen Rotationsfunktion eine Laufzeiteigenschaft hinzufügen oder die Rotationsfunktion neu erstellen.

So finden Sie heraus, welche Lambda-Rotationsfunktionen Python 3.7 verwenden

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die AWS Lambda Konsole unter <https://console.aws.amazon.com/lambda/>.
2. Filtern Sie in der Liste der Funktionen nach **SecretsManager**.
3. Suchen Sie in der gefilterten Liste der Funktionen unter Laufzeit nach Python 3.7.

Aktualisieren Sie auf Python 3.9 wie folgt:

- [Option 1: Erstellen Sie die Rotationsfunktion neu mit AWS CloudFormation](#)
- [Option 2: Aktualisieren Sie die Laufzeit für die bestehende Rotationsfunktion mit AWS CloudFormation](#)
- [Option 3: AWS CDK Benutzer müssen die CDK-Bibliothek aktualisieren](#)

Option 1: Erstellen Sie die Rotationsfunktion neu mit AWS CloudFormation

Wenn Sie die Secrets Manager-Konsole verwenden, um die Rotation zu aktivieren, erstellt Secrets Manager die erforderlichen Ressourcen, einschließlich der Lambda-Rotationsfunktion. AWS CloudFormation Wenn Sie die Konsole verwendet haben, um die Rotation zu aktivieren, oder wenn Sie die Rotationsfunktion mithilfe eines AWS CloudFormation Stacks erstellt haben, können Sie denselben AWS CloudFormation Stack verwenden, um die Rotationsfunktion mit einem neuen Namen neu zu erstellen. Die neue Funktion verwendet die neuere Version von Python.

Um den AWS CloudFormation Stapel zu finden, der die Rotationsfunktion erstellt hat

- Wählen Sie auf der Detailseite zur Lambda-Funktion auf der Registerkarte Konfiguration Tags aus. Zeigen Sie den ARN neben `aws:cloudformation:stack-id` an.

Der Stack-Name ist in den ARN eingebettet, wie im folgenden Beispiel gezeigt.

- ARN: `arn:aws:cloudformation:us-west-2:408736277230:stack/SecretsManagerRDSMySQLRotationSingleUser5c2-SecretRotationScheduleHostedRotationLambda-3CUDHZMDMB08/79fc9050-2eef-11ed-`
- Stackname: **SecretsManagerRDSMySQLRotationSingleUser5c2-SecretRotationScheduleHostedRotationLambda**

So erstellen Sie eine Rotationsfunktion neu (AWS CloudFormation)

1. Suchen Sie in AWS CloudFormation anhand des Namens nach dem Stapel und wählen Sie dann Aktualisieren aus.

Wenn ein Dialogfeld angezeigt wird, in dem empfohlen wird, den Root-Stack zu aktualisieren, wählen Sie Zum Root-Stack gehen und dann Aktualisieren aus.

2. Wählen Sie auf der Seite Stack aktualisieren die Option Vorlage im Designer bearbeiten und anschließend Im Designer anzeigen aus.
3. Ersetzen Sie im Designer im Vorlagencode, in `SecretRotationScheduleHostedRotationLambda`, den Wert für `"functionName"`: `"SecretsManagerTestRotationRDS"` durch einen neuen Funktionsnamen, z. B. in JSON, **`"functionName": "SecretsManagerTestRotationRDSupdated"`**
4. Fahren Sie mit dem AWS CloudFormation Stack-Workflow fort und wählen Sie dann Submit aus.

Option 2: Aktualisieren Sie die Laufzeit für die bestehende Rotationsfunktion mit AWS CloudFormation

Wenn Sie die Secrets Manager-Konsole verwenden, um die Rotation zu aktivieren, erstellt Secrets Manager die erforderlichen Ressourcen, einschließlich der Lambda-Rotationsfunktion. AWS CloudFormation Wenn Sie die Konsole zum Aktivieren der Rotation verwendet haben oder die Rotationsfunktion mithilfe eines AWS CloudFormation Stacks erstellt haben, können Sie denselben AWS CloudFormation Stack verwenden, um die Laufzeit für die Rotationsfunktion zu aktualisieren.

Um den AWS CloudFormation Stapel zu finden, der die Rotationsfunktion erstellt hat

- Wählen Sie auf der Detailseite zur Lambda-Funktion auf der Registerkarte Konfiguration Tags aus. Zeigen Sie den ARN neben `aws:cloudformation:stack-id` an.

Der Stack-Name ist in den ARN eingebettet, wie im folgenden Beispiel gezeigt.

- ARN: `arn:aws:cloudformation:us-west-2:408736277230:stack/SecretsManagerRDSMySQLRotationSingleUser5c2-SecretRotationScheduleHostedRotationLambda-3CUDHZMDMB08/79fc9050-2eef-11ed-`
- Stackname: **SecretsManagerRDSMySQLRotationSingleUser5c2-SecretRotationScheduleHostedRotationLambda**

So aktualisieren Sie die Laufzeit für eine Rotationsfunktion (AWS CloudFormation)

1. Suchen Sie in AWS CloudFormation anhand des Namens nach dem Stapel und wählen Sie dann Aktualisieren aus.

Wenn ein Dialogfeld angezeigt wird, in dem empfohlen wird, den Root-Stack zu aktualisieren, wählen Sie Zum Root-Stack gehen und dann Aktualisieren aus.

2. Wählen Sie auf der Seite Stack aktualisieren die Option Vorlage im Designer bearbeiten und anschließend Im Designer anzeigen aus.
3. Im Designer, in der JSON-Vorlage `SecretRotationScheduleHostedRotationLambda`, für `Properties`, unter `Parameters`, hinzufügen **"runtime": "python3.9"**
4. Fahren Sie mit dem AWS CloudFormation Stack-Workflow fort und wählen Sie dann Submit aus.

Option 3: AWS CDK Benutzer müssen die CDK-Bibliothek aktualisieren

Wenn Sie die AWS CDK frühere Version v2.94.0 verwendet haben, um die Rotation für Ihr Geheimnis einzurichten, können Sie die Lambda-Funktion aktualisieren, indem Sie ein Upgrade auf Version 2.94.0 oder höher durchführen. Weitere Informationen finden Sie im [AWS Cloud Development Kit \(AWS CDK\) -v2-Entwicklerhandbuch](#).

Drehe ein AWS Secrets Manager Geheimnis sofort

Sie können nur ein Secret rotieren, bei dem die Rotation aktiviert ist. Um festzustellen, ob ein Secret für die Drehung konfiguriert wurde, zeigen Sie in der Konsole das Secret an und scrollen Sie nach unten zum Abschnitt Rotation configuration (Rotationskonfiguration). Wenn der Rotation status (Drehungsstatus) Enabled (Aktiviert) lautet, ist das Secret für die Drehung konfiguriert. Falls nicht, siehe [Rotieren von -Geheimnissen](#).

So drehen Sie sofort ein Secret (Konsole)

1. Öffnen Sie die Secrets-Manager-Konsole unter <https://console.aws.amazon.com/secretsmanager/>.
2. Wählen Sie Ihr Secret aus.
3. Wählen Sie auf der Details-Seite des Secrets unter Drehungskonfiguration, Drehen Sie das Secret sofort aus.
4. Wählen Sie im Dialogfeld Secret drehen die Option Drehen.

AWS CLI

Example Secret sofort drehen

Im folgenden [rotate-secret](#)-Beispiel wird eine sofortige Rotation gestartet. Für das Secret muss die Rotation bereits konfiguriert sein.

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

Rotationspläne

Wenn Sie die automatische Rotation aktivieren, können Sie einen cron ()- oder rate ()-Ausdruck verwenden, um den Zeitplan für die Rotation Ihres Secrets festzulegen. Bei einem rate-Ausdruck

können Sie einen Rotationsplan erstellen, der sich in einem Zeitintervall von Stunden oder Tagen wiederholt. Bei einem cron-Ausdruck können Sie Rotationspläne erstellen, die detaillierter sind als ein Rotationsintervall. Rotationspläne von Secrets Manager verwenden die UTC-Zeitzone. Sie können ein Secret bis zu alle vier Stunden rotieren. Secrets Manager rotiert Ihr Secret während des Rotationsfensters.

Informationen zum Aktivieren der Drehung finden Sie unter:

- [the section called “Verwaltete Rotation”](#)
- [the section called “Automatische Rotierung für Datenbank-Secrets \(Konsole\)”](#)
- [the section called “Automatische Rotation für Nicht-Datenbankgeheimnisse \(Konsole\)”](#)

Rate-Ausdrücke

rate-Ausdrücke von Secrets Manager haben das folgende Format, bei dem *Value* (Wert) eine positive Ganzzahl ist und *Unit* (Einheit) hour, hours, day oder days sein kann:

```
rate(Value Unit)
```

Sie können ein Secret bis zu alle vier Stunden rotieren. Beispiele:

- `rate(4 hours)` bedeutet, dass das Secret alle vier Stunden rotiert wird.
- `rate(1 day)` bedeutet, dass das Secret jeden Tag rotiert wird.
- `rate(10 days)` bedeutet, dass das Secret alle zehn Tage rotiert wird.

Bei einer Rate in Stunden beginnt das Standard-Rotationsfenster um Mitternacht und schließt nach einer Stunde. Sie können die Dauer des Fensters festlegen, um das Rotationsfenster zu ändern. Das Rotationsfenster darf nicht in das nächste Rotationsfenster übergehen. Sie können dafür überprüfen, ob das Rotationsfenster kleiner oder gleich der Anzahl der Stunden zwischen den Rotationen ist.

Bei einer Rate in Tagen beginnt das Standard-Rotationsfenster um Mitternacht und schließt am Ende des Tages. Sie können die Dauer des Fensters festlegen, um das Rotationsfenster zu ändern. Das Rotationsfenster darf nicht in den nächsten UTC-Tag übergehen. Sie können dafür bestätigen, dass die Startstunde plus die Fensterdauer weniger als oder gleich 24 Stunden betragen.

Cron-Ausdrücke

cron-Ausdrücke haben das folgende Format.

```
cron(Minutes Hours Day-of-month Month Day-of-week Year)
```

Ein Cron-Ausdruck, der Stundenschritte enthält, wird jeden Tag zurückgesetzt. `cron(0 4/12 * * ? *)` bedeutet zum Beispiel 4:00 Uhr, 16:00 Uhr und dann am nächsten Tag 4:00 Uhr, 16:00 Uhr. Rotationspläne von Secrets Manager verwenden die UTC-Zeitzone.

Bei einem Plan in Stunden schließt das Standard-Rotationsfenster nach einer Stunde. Sie können die Dauer des Fensters festlegen, um das Rotationsfenster zu ändern. Das Rotationsfenster darf nicht in das nächste Rotationsfenster übergehen. Sie können ein Secret bis zu alle vier Stunden rotieren.

Beispiel für einen Zeitplan	Expression
Alle acht Stunden ab Mitternacht.	<code>cron(0 /8 * * ? *)</code>
Alle acht Stunden ab 8:00 Uhr.	<code>cron(0 8/8 * * ? *)</code>
Alle zehn Stunden ab 2:00 Uhr.	<code>cron(0 2/10 * * ? *)</code>
Die Rotationsfenster beginnen um 2:00 Uhr, 12:00 Uhr und 22:00 Uhr und dann am nächsten Tag um 2:00 Uhr, 12:00 Uhr und 22:00 Uhr.	
Täglich um 10:00 Uhr.	<code>cron(0 10 * * ? *)</code>
Jeden Samstag um 18:00 Uhr.	<code>cron(0 18 ? * SAT *)</code>
Ausführung jeden 1. Tag des Monats um 08:00 Uhr.	<code>cron(0 8 1 * ? *)</code>
Alle drei Monate am ersten Sonntag um 1:00 Uhr.	<code>cron(0 1 ? 1/3 SUN#1 *)</code>
5:00 Uhr jeden letzten Tag im Monat	<code>cron(0 17 L * ? *)</code>
Montag bis Freitag um 8:00 Uhr.	<code>cron(0 8 ? * MON-FRI *)</code>
Erster und 15. Tag eines jeden Monats um 16:00 Uhr.	<code>cron(0 16 1,15 * ? *)</code>

Beispiel für einen Zeitplan	Expression
Jeden ersten Sonntag im Monat um Mitternacht.	<code>cron(0 0 ? * SUN#1 *)</code>

Anforderungen an Cron-Ausdrücke in Secrets Manager

Secrets Manager hat einige Einschränkungen im Hinblick darauf, was Sie für Cron-Ausdrücke verwenden können. Ein cron-Ausdruck für Secrets Manager muss 0 im Feld „Minutes“ (Minuten) haben, da Secrets-Manager-Rotationsfenster zur vollen Stunde gestartet werden. Es muss *im Feld „Jahr“ haben, da Secrets Manager keine Rotationspläne unterstützt, die mehr als ein Jahr voneinander entfernt sind. Die folgende Tabelle zeigt die Optionen an, die Sie verwenden können.

Felder	Werte	Platzhalter
Minuten	Muss 0 sein	None
Stunden	0–23	Verwenden Sie / (Schrägstrich), um die Schrittweite anzugeben. 2/10 bedeutet zum Beispiel alle zehn Stunden ab 2:00 Uhr. Sie können ein Secret bis zu alle vier Stunden rotieren.
Day-of-month	1-31	Verwenden Sie , (Komma), um zusätzliche Werte hinzuzufügen. 1, 15 bedeutet zum Beispiel den 1. und 15. Tag des Monats. Verwenden Sie - (Bindestrich), um einen Bereich anzugeben. 1–15 bedeutet zum Beispiel die Tage 1 bis 15 des Monats.

Felder	Werte	Platzhalter
		<p>Verwenden Sie das Platzhalterzeichen * (Sternchen), um alle Werte im Feld einzubeziehen. * bedeutet zum Beispiel jeden Tag des Monats.</p> <p>Das Platzhalterzeichen ? (Fragezeichen) steht für einen Wert. Es ist nicht möglich, die Felder Day-of-month und Day-of-week im gleichen Cron-Ausdruck anzugeben. Wenn Sie einen Wert in einem der Felder angeben, müssen Sie in dem anderen Feld ein ? (Fragezeichen) eingeben.</p> <p>Verwenden Sie /(Schrägstrich), um die Schrittweite anzugeben. 1/2 bedeutet zum Beispiel alle zwei Tage ab Tag 1, also Tag 1, 3, 5 usw.</p> <p>Verwenden Sie L, um den letzten Tag des Monats anzugeben.</p> <p>Verwenden Sie DAYL, um den letzten genannten Tag des Monats anzugeben. SUNL bedeutet zum Beispiel den letzten Sonntag des Monats.</p>

Felder	Werte	Platzhalter
Monat	1–12 oder JAN-DEZ	<p>Verwenden Sie , (Komma), um zusätzliche Werte hinzuzufügen. JAN, APR, JUL, OCT bedeutet beispielsweise Januar, April, Juli und Oktober.</p> <p>Verwenden Sie - (Bindestrich), um einen Bereich anzugeben. 1–3 bedeutet zum Beispiel die Monate 1 bis 3 des Jahres.</p> <p>Verwenden Sie das Platzhalterzeichen * (Sternchen), um alle Werte im Feld einzubeziehen. * bedeutet zum Beispiel jeden Monat.</p> <p>Verwenden Sie /(Schrägstrich), um die Schrittweite anzugeben. 1/3 bedeutet zum Beispiel jeden dritten Monat, beginnend mit Monat 1, also Monat 1, 4, 7 und 10.</p>

Felder	Werte	Platzhalter
Day-of-week	1–7 oder SO-SA	<p>Verwenden Sie #, um den Wochentag innerhalb eines Monats anzugeben. Beispielsweise steht TUE#3 für den dritten Dienstag im Monat.</p> <p>Verwenden Sie , (Komma), um zusätzliche Werte hinzuzufügen. 1, 4 bedeutet zum Beispiel den ersten und vierten Tag der Woche.</p> <p>Verwenden Sie - (Bindestrich), um einen Bereich anzugeben. 1–4 bedeutet zum Beispiel die Tage 1 bis 4 der Woche.</p> <p>Verwenden Sie das Platzhalterzeichen * (Sternchen), um alle Werte im Feld einzubeziehen. * bedeutet zum Beispiel jeden Tag der Woche.</p> <p>Das Platzhalterzeichen ? (Fragezeichen) steht für einen Wert. Es ist nicht möglich, die Felder Day-of-month und Day-of-week im gleichen Cron-Ausdruck anzugeben. Wenn Sie einen Wert in einem der Felder angeben, müssen Sie in dem anderen Feld ein ? (Fragezeichen) eingeben.</p> <p>Verwenden Sie /(Schrägstrich), um die Schrittweite</p>

Felder	Werte	Platzhalter
		anzugeben. 1/2 bedeutet beispielsweise jeden zweiten Wochentag, beginnend mit dem ersten Tag, also Tag 1, 3, 5 und 7. Verwenden Sie L, um den letzten Tag der Woche anzugeben.
Jahr	Muss * sein.	None

Finde Geheimnisse, die nicht rotiert werden

Damit können Sie AWS Config Ihre Geheimnisse auswerten, um festzustellen, ob sie gemäß Ihren Standards rotieren. Sie definieren Ihre internen Sicherheits- und Compliance-Anforderungen für Geheimnisse mithilfe von AWS Config Regeln. Dann AWS Config können Sie Geheimnisse identifizieren, die nicht Ihren Regeln entsprechen. Sie können auch Änderungen an geheimen Metadaten, der Drehungskonfiguration, dem für die geheime Verschlüsselung verwendeten KMS-Schlüssel, der Lambda-Drehungsfunktion und mit einem Geheimnis verknüpften Tags nachverfolgen.

Wenn Sie in mehreren AWS-Konten und AWS-Regionen in Ihrer Organisation über Geheimnisse verfügen, können Sie diese Konfigurations- und Compliance-Daten zusammenfassen. Weitere Informationen finden Sie unter [Datenaggregation für mehrere Konten und mehrere Regionen](#).

Um zu beurteilen, ob geheime Daten rotieren

1. Folgen Sie den Anweisungen zur [Bewertung Ihrer Ressourcen mithilfe von AWS Config Regeln](#) und wählen Sie eine der folgenden Regeln aus:
 - [secretsmanager-rotation-enabled-check](#) – Prüft, ob die Rotation für in Secrets Manager gespeicherte Secrets konfiguriert ist.
 - [secretsmanager-scheduled-rotation-success-check](#) – Prüft, ob die letzte erfolgreiche Rotation innerhalb der konfigurierten Rotationsfrequenz liegt. Die Überprüfung muss mindestens täglich erfolgen.

- [secretsmanager-secret-periodic-rotation](#) – Prüft, ob Secrets innerhalb der letzten angegebenen Anzahl von Tagen rotiert wurden.
2. Sie können die Konfiguration optional so konfigurieren, AWS Config dass Sie benachrichtigt werden, wenn geheime Daten nicht den Vorschriften entsprechen. Weitere Informationen finden Sie unter [Benachrichtigungen, die AWS Config an ein Amazon SNS SNS-Thema gesendet](#) werden.

Automatische Rotation im Secrets Manager abbrechen

Wenn Sie die [automatische Rotation](#) für ein Geheimnis konfiguriert haben und die Rotation beenden möchten, können Sie die Rotation abbrechen.

Um die automatische Rotation abzubrechen

1. Öffnen Sie die Secrets-Manager-Konsole unter <https://console.aws.amazon.com/secretsmanager/>.
2. Wählen Sie Ihr Secret aus.
3. Wählen Sie auf der Seite mit den geheimen Details unter Rotationskonfiguration die Option Rotation bearbeiten aus.
4. Deaktivieren Sie im Dialogfeld „Rotationskonfiguration bearbeiten“ die Option „Automatische Rotation“ und wählen Sie dann „Speichern“.

Secrets Manager speichert die Informationen zur Rotationskonfiguration, sodass Sie sie in future verwenden können, wenn Sie die Rotation wieder einschalten möchten.

AWS Secrets Manager Geheimnisse, die von anderen AWS Diensten verwaltet werden

Viele AWS Dienste speichern und verwenden Geheimnisse in AWS Secrets Manager. In einigen Fällen handelt es sich dabei um verwaltete Secrets. Das bedeutet, dass der Service, der sie erstellt hat, bei deren Verwaltung hilft. Einige verwaltete Secrets beinhalten beispielsweise eine [verwaltete Rotation](#), sodass Sie die Rotation nicht selbst konfigurieren müssen. Der verwaltende Service hindert Sie möglicherweise auch daran, Secrets zu löschen oder sie ohne Wiederherstellungszeitraum zu aktualisieren. Dies trägt zur Verhinderung von Ausfällen bei, da der verwaltende Service vom Secret abhängt.

Verwaltete Secrets enthalten die ID des Verwaltungsservice im Namen, damit sie leichter identifiziert werden können.

```
Secret name: ServiceID!MySecret
Secret ARN : arn:aws:us-east-1:ServiceID!MySecret-a1b2c3
```

IDs für Services, die Secrets verwalten

- 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-Abfrage-Editor v2”](#)

Informationen zum Auffinden von Geheimnissen, die von anderen AWS Diensten verwaltet werden, finden [Sie unter Finden verwalteter Geheimnisse](#).

Eine vollständige Liste der Dienste, die geheime Daten verwenden, finden Sie unter [Dienste, die Geheimnisse verwenden](#).

AWS Dienste, die AWS Secrets Manager Geheimnisse verwenden

Informieren Sie sich darüber, wie sich die folgenden Funktionen AWS-Services in Secrets Manager integrieren lassen.

- [Wie AWS App Runner verwendet AWS Secrets Manager](#)
- [Wie verwendet AWS App2Container AWS Secrets Manager](#)
- [Wie verwendet AWS AppConfigAWS Secrets Manager](#)
- [So AppFlow nutzt Amazon AWS Secrets Manager](#)
- [Wie AWS AppSync verwendet AWS Secrets Manager](#)
- [So nutzt Amazon Athena AWS Secrets Manager](#)
- [So verwendet Amazon Aurora AWS Secrets Manager](#)
- [Wie AWS CodeBuild verwendet AWS Secrets Manager](#)
- [So verwendet Amazon Data Firehose AWS Secrets Manager](#)
- [Wie verwendet AWS DataSyncAWS Secrets Manager](#)
- [So DataZone nutzt Amazon AWS Secrets Manager](#)
- [Wie verwendet AWS Direct ConnectAWS Secrets Manager](#)
- [Wie AWS Directory Service verwendet AWS Secrets Manager](#)
- [So nutzt Amazon DocumentDB \(mit MongoDB-Kompatibilität\) AWS Secrets Manager](#)
- [Wie verwendet AWS Elastic BeanstalkAWS Secrets Manager](#)
- [So verwendet Amazon Elastic Container Registry AWS Secrets Manager](#)
- [Amazon Elastic Container Service](#)
- [So ElastiCache nutzt Amazon AWS Secrets Manager](#)
- [Wie AWS Elemental Live verwendet AWS Secrets Manager](#)
- [Wie verwendet AWS Elemental MediaConnectAWS Secrets Manager](#)
- [Wie AWS Elemental MediaConvert verwendet AWS Secrets Manager](#)
- [Wie verwendet AWS Elemental MediaLiveAWS Secrets Manager](#)
- [Wie AWS Elemental MediaPackage verwendet AWS Secrets Manager](#)

- [Wie verwendet AWS Elemental MediaTailor AWS Secrets Manager](#)
- [So verwendet Amazon EMR Secrets Manager](#)
- [So EventBridge nutzt Amazon AWS Secrets Manager](#)
- [Wie Amazon FSx Geheimnisse verwendet AWS Secrets Manager](#)
- [Wie AWS Glue DataBrew verwendet AWS Secrets Manager](#)
- [Wie verwendet AWS Glue Studio AWS Secrets Manager](#)
- [Wie verwendet AWS IoT SiteWise AWS Secrets Manager](#)
- [So verwendet Amazon Kendra AWS Secrets Manager](#)
- [So verwendet Amazon Kinesis Video Streams AWS Secrets Manager](#)
- [Wie verwendet AWS Launch Wizard AWS Secrets Manager](#)
- [So verwendet Amazon Lookout for Metrics AWS Secrets Manager](#)
- [So verwendet Amazon Managed Grafana AWS Secrets Manager](#)
- [Wie verwendet AWS Managed Services AWS Secrets Manager](#)
- [So verwendet Amazon Managed Streaming for Apache Kafka AWS Secrets Manager](#)
- [So verwendet Amazon Managed Workflows für Apache Airflow AWS Secrets Manager](#)
- [AWS Marketplace](#)
- [Wie AWS Migration Hub verwendet AWS Secrets Manager](#)
- [Wie AWS Panorama verwendet Secrets Manager](#)
- [Wie AWS ParallelCluster verwendet AWS Secrets Manager](#)
- [So verwendet Amazon Q Secrets Manager](#)
- [Wie AWS OpsWorks for Chef Automate verwendet AWS Secrets Manager](#)
- [So QuickSight nutzt Amazon AWS Secrets Manager](#)
- [Amazon RDS](#)
- [So verwendet Amazon Redshift AWS Secrets Manager](#)
- [Amazon-Redshift-Abfrage-Editor v2](#)
- [So SageMaker nutzt Amazon AWS Secrets Manager](#)
- [Wie verwendet AWS Schema Conversion Tool AWS Secrets Manager](#)
- [Wie AWS Toolkit for JetBrains verwendet AWS Secrets Manager](#)
- [Wie AWS Transfer Family benutzt AWS Secrets Manager Secrets](#)

- [Wie AWS Wickr Geheimnisse verwendet AWS Secrets Manager](#)

Wie AWS App Runner verwendet AWS Secrets Manager

AWS App Runner ist ein AWS Service, der eine schnelle, einfache und kostengünstige Möglichkeit bietet, Quellcode oder ein Container-Image direkt in einer skalierbaren und sicheren Webanwendung in der AWS Cloud bereitzustellen. Sie müssen sich nicht mit neuen Technologien vertraut machen, entscheiden, welchen Rechendienst Sie verwenden möchten, oder wissen, wie AWS Ressourcen bereitgestellt und konfiguriert werden.

Mit App Runner können Sie Secrets und Konfigurationen als Umgebungsvariablen in Ihrem Service referenzieren, wenn Sie einen Service erstellen oder die Konfiguration des Services aktualisieren. Weitere Informationen finden Sie unter [Referenzieren von Umgebungsvariablen](#) und [Verwalten von Umgebungsvariablen](#) im AWS App Runner -Entwicklerhandbuch.

Wie verwendet AWS App2Container AWS Secrets Manager

AWS App2Container ist ein Befehlszeilentool, mit dem Sie Anwendungen, die in Ihren lokalen Rechenzentren oder auf virtuellen Maschinen ausgeführt werden, nach oben verschieben können, sodass sie in Containern ausgeführt werden, die von Amazon ECS, Amazon EKS oder AWS App Runner verwaltet werden.

App2Container verwendet Secrets Manager, um die Anmeldeinformationen für die Verbindung Ihres Arbeitscomputers mit den Anwendungsservern zu verwalten, damit Sie Remote-Befehle ausführen können. Weitere Informationen finden Sie unter [Manage Secrets for AWS App2Container im App2Container-Benutzerhandbuch](#) AWS .

Wie verwendet AWS AppConfig AWS Secrets Manager

AWS AppConfig ist eine Funktion AWS Systems Manager , mit der Sie Anwendungskonfigurationen erstellen, verwalten und schnell bereitstellen können. Eine Konfiguration kann Anmeldeinformationen oder andere vertrauliche Informationen enthalten, die in Secrets Manager gespeichert sind. Wenn Sie ein Freiform-Konfigurationsprofil erstellen, können Sie Secrets Manager als Quelle Ihrer Konfigurationsdaten wählen. Weitere Informationen finden Sie unter [Erstellen eines Freiform-Konfigurationsprofils](#) im Benutzerhandbuch für AWS AppConfig . Informationen zum AWS AppConfig Umgang mit Geheimnissen, bei denen die automatische Rotation aktiviert ist, finden Sie unter [Secrets Manager Manager-Schlüsselrotation](#) im AWS AppConfig Benutzerhandbuch.

So AppFlow nutzt Amazon AWS Secrets Manager

Amazon AppFlow ist ein vollständig verwalteter Integrationservice, mit dem Sie sicher Daten zwischen SaaS-Anwendungen (Software as a Service) wie Salesforce und AWS-Services Amazon Simple Storage Service (Amazon S3) und Amazon Redshift austauschen können.

Wenn Sie in Amazon AppFlow eine SaaS-Anwendung als Quelle oder Ziel konfigurieren, stellen Sie eine Verbindung her. Diese umfasst Informationen, die für die Verbindung mit den SaaS-Anwendungen erforderlich sind, wie Authentifizierungstoken, Benutzernamen und Passwörter. Amazon AppFlow speichert Ihre Verbindungsdaten in einem Secrets Manager [Managed Secret](#) mit dem Präfix `appflow`. Die Kosten für die Aufbewahrung des Geheimnisses sind in der Gebühr für Amazon AppFlow enthalten. Weitere Informationen finden Sie unter [Datenschutz bei Amazon AppFlow](#) im AppFlow Amazon-Benutzerhandbuch.

Wie AWS AppSync verwendet AWS Secrets Manager

AWS AppSync bietet eine robuste, skalierbare GraphQL-Schnittstelle für Anwendungsentwickler, um Daten aus mehreren Quellen, einschließlich Amazon DynamoDB und HTTP-APIs AWS Lambda, zu kombinieren.

AWS AppSync verwendet den CLI-Befehl `aws rds execute-statement`, um eine Verbindung zu Amazon RDS herzustellen, wobei die Anmeldeinformationen in einem geheimen Ordner verwendet werden. Weitere Informationen finden Sie unter [Tutorial: Aurora Serverless](#) im AWS AppSync - Entwicklerhandbuch.

So nutzt Amazon Athena AWS Secrets Manager

Amazon Athena ist ein interaktiver Abfrageservice, der die direkte Analyse von Daten in Amazon Simple Storage Service (Amazon S3) mit Standard-SQL erleichtert.

Amazon-Athena-Datenquellen-Connectors können das Athena-Federated-Query-Feature mit Secrets Manager-Secrets verwenden, um Daten abzufragen. Weitere Informationen finden Sie unter [Verwenden der Verbundabfrage von Amazon Athena](#) im Amazon-Athena-Benutzerhandbuch.

So verwendet Amazon Aurora AWS Secrets Manager

Amazon Aurora ist eine vollständig verwaltete, mit MySQL und PostgreSQL kompatible relationale Datenbank-Engine.

Um die Master-Benutzeranmeldedaten für Aurora zu verwalten, kann Aurora ein [verwaltetes Geheimnis](#) für Sie erstellen. Ihnen werden Kosten für dieses Geheimnis berechnet. Aurora [verwaltet auch die Rotation](#) dieser Zugangsdaten. Weitere Informationen finden Sie unter [Passwortverwaltung mit Amazon Aurora und AWS Secrets Manager](#) im Amazon-Aurora-Benutzerhandbuch.

Weitere Aurora-Anmeldeinformationen finden Sie unter [the section called “Erstellen eines Datenbank-Secrets”](#).

Wenn Sie die Daten-API von Amazon RDS aufrufen, können Sie die Anmeldeinformationen für die Datenbank unter Verwendung eines Secrets in Secrets Manager übergeben. Weitere Informationen finden Sie unter [Verwenden der Daten-API für Aurora Serverless](#) im Amazon Aurora-Benutzerhandbuch.

Wenn Sie den Amazon-RDS-Abfrage-Editor verwenden, um eine Verbindung zu einer Datenbank herzustellen, können Sie Anmeldeinformationen für die Datenbank in Secrets Manager speichern. Weitere Informationen finden Sie unter [Verwenden des Abfrage-Editors](#) im Amazon-RDS-Benutzerhandbuch.

Wie AWS CodeBuild verwendet AWS Secrets Manager

AWS CodeBuild ist ein vollständig verwalteter Build-Service in der Cloud. CodeBuild kompiliert Ihren Quellcode, führt Unit-Tests durch und produziert Artefakte, die sofort einsatzbereit sind.

Sie können Ihre privaten Registry-Anmeldeinformationen mit Secrets Manager speichern. Weitere Informationen finden Sie unter [Private Registry mit AWS Secrets Manager Beispiel für CodeBuild](#) im AWS CodeBuild Benutzerhandbuch.

So verwendet Amazon Data Firehose AWS Secrets Manager

Sie können Amazon Data Firehose verwenden, um Streaming-Daten in Echtzeit an verschiedene Streaming-Ziele zu liefern. Wenn das Ziel Anmeldeinformationen oder Schlüssel benötigt, ruft Firehose zur Laufzeit ein Geheimnis von Secrets Manager ab, um eine Verbindung zum Ziel herzustellen. Weitere Informationen finden Sie unter [Authentifizieren mit AWS Secrets Manager in Amazon Data Firehose](#) im Amazon Data Firehose Developer Guide.

Wie verwendet AWS DataSync AWS Secrets Manager

AWS DataSync ist ein Online-Datenübertragungsdienst, der die Übertragung von Daten zwischen Speichersystemen und Diensten vereinfacht, automatisiert und beschleunigt. DataSync Discovery hilft Ihnen dabei, Ihre Migration zu zu AWS beschleunigen.

Um Informationen über ein lokales Speichersystem zu sammeln, verwendet DataSync Discovery die Anmeldeinformationen für die Verwaltungsschnittstelle des Speichersystems. DataSync speichert diese Anmeldeinformationen in einem von Secrets Manager [verwalteten geheimen Schlüssel](#) mit dem Präfix `dataasync`. Ihnen werden Kosten für dieses Geheimnis berechnet. Weitere Informationen finden Sie unter [Hinzufügen Ihres lokalen Speichersystems zu DataSync Discovery](#) im AWS DataSync Benutzerhandbuch.

So DataZone nutzt Amazon AWS Secrets Manager

Amazon DataZone ist ein Datenverwaltungsservice, mit dem Sie Ihre Daten katalogisieren, ermitteln, verwalten, teilen und analysieren können. Sie können Datenbestände aus Tabellen und Ansichten aus einem Amazon Redshift Redshift-Cluster verwenden, der mithilfe eines AWS-Glue-Crawler Jobs gecrawlt wird. Um eine Verbindung zu Amazon Redshift herzustellen, geben Sie DataZone Amazon-Anmeldeinformationen in einem Secrets Manager Manager-Geheimnis ein. Weitere Informationen finden Sie unter [Erstellen einer Datenquelle für eine Amazon Redshift Redshift-Datenbank mithilfe einer neuen AWS Glue Verbindung](#) im DataZone Amazon-Benutzerhandbuch.

Wie verwendet AWS Direct Connect AWS Secrets Manager

AWS Direct Connect verbindet Ihr internes Netzwerk über ein Standard-Ethernet-Glasfaserkabel mit einem AWS Direct Connect Standort. Mit dieser Verbindung können Sie virtuelle Schnittstellen direkt zur Öffentlichkeit erstellen. AWS-Services

AWS Direct Connect speichert einen Konnektivitätszuordnungsschlüsselnamen und ein Konnektivitätszuordnungsschlüsselpaar (CKN/CAK-Paar) in einem [verwalteten Geheimnis](#) mit dem Präfix `directconnect`. Die Kosten für das Secret sind in der Gebühr für enthalten. AWS Direct Connect Um den Secret zu aktualisieren, müssen Sie Secrets Manager AWS Direct Connect anstelle von Secrets Manager verwenden. Weitere Informationen finden Sie unter [Verknüpfen eines MACsec CKN/CAK mit einer LAG](#) im AWS Direct Connect -Benutzerhandbuch.

Wie AWS Directory Service verwendet AWS Secrets Manager

AWS Directory Service bietet mehrere Möglichkeiten, Microsoft Active Directory (AD) mit anderen AWS Diensten zu verwenden. Sie können eine Amazon-EC2-Instance mit Ihrem Verzeichnis verbinden, indem Sie Secrets für Anmeldeinformationen verwenden. Weitere Informationen finden Sie im AWS Direct Connect -Benutzerhandbuch unter:

- [Fügen Sie eine Linux EC2-Instance nahtlos zu Ihrem AWS verwalteten Microsoft AD-Verzeichnis hinzu](#)
- [Nahtloses Verbinden einer Linux-EC2-Instance mit Ihrem AD-Connector-Verzeichnis](#)
- [Nahtloses Verbinden einer Linux-EC2-Instance mit Ihrem Simple-AD-Verzeichnis](#)

So nutzt Amazon DocumentDB (mit MongoDB-Kompatibilität) AWS Secrets Manager

In Amazon DocumentDB authentifizieren sich Benutzer bei einem Cluster in Verbindung mit einem Passwort. Mit AWS Secrets Manager können Sie fest codierte Anmeldeinformationen im Code (einschließlich Passwörter) durch einen API-Aufruf an Secrets Manager ersetzen und das Secret programmgesteuert abrufen. Weitere Informationen finden Sie unter [the section called “Erstellen eines Datenbank-Secrets”](#) und [Verwalten von Amazon-DocumentDB-Benutzern](#) im Amazon-DocumentDB-Entwicklerhandbuch.

Wie verwendet AWS Elastic Beanstalk AWS Secrets Manager

Mit AWS Elastic Beanstalk können Sie Anwendungen schnell in der AWS Cloud bereitstellen und verwalten, ohne sich mit der Infrastruktur vertraut machen zu müssen, auf der diese Anwendungen ausgeführt werden. Elastic Beanstalk kann Docker-Umgebungen starten. Dazu wird ein in einem Dockerfile beschriebenes Image erstellt oder ein Remote-Docker-Image abgerufen. Zur Authentifizierung bei der Onlineregistrierung, die das private Repository hostet, verwendet Elastic Beanstalk ein Secrets-Manager-Secret. Weitere Informationen finden Sie unter [Docker configuration](#) im Entwicklerhandbuch für AWS Elastic Beanstalk .

So verwendet Amazon Elastic Container Registry AWS Secrets Manager

Amazon Elastic Container Registry (Amazon ECR) ist ein AWS verwalteter Container-Image-Registry-Service, der sicher, skalierbar und zuverlässig ist. Sie können die Docker-Befehlszeilenschnittstelle (CLI) oder Ihren bevorzugten Client verwenden, um Images in Ihre Repositories zu pushen und von dort abzuholen. Sie müssen für jede Upstream-Registrierung, die Images enthält, die Sie in Ihrer privaten Registrierung von Amazon ECR zwischenspeichern möchten, eine Pull-Through-Cache-Regel erstellen. Für Upstream-Registrierungen, für die eine Authentifizierung erforderlich ist, müssen Sie die Anmeldeinformationen in einem Secrets-Manager-Secret speichern. Sie können das Secrets-Manager-Secret entweder in der Amazon-ECR- oder in der Secrets-Manager-Konsole erstellen. Weitere Informationen finden Sie unter [Erstellen einer Pull-Through-Cache-Regel](#) im Amazon ECR-Benutzerhandbuch.

Amazon Elastic Container Service

Amazon Elastic Container Service (Amazon ECS) ist ein vollständig verwalteter Container-Orchestrierungsservice, mit dem Sie containerisierte Anwendungen einfach bereitstellen, verwalten und skalieren können. Sie können vertrauliche Daten in Ihre Container injizieren, indem Sie auf Secrets-Manager-Secrets verweisen. Weitere Informationen finden Sie auf den folgenden Seiten des Entwicklerhandbuchs von Amazon Elastic Container Service:

- [Tutorial: Angeben vertraulicher Daten mithilfe von Secrets-Manager-Secrets](#)
- [Programmgesteuertes Abrufen von Geheimnissen über Ihre Anwendung](#)
- [Abrufen von Secrets über Umgebungsvariablen](#)
- [Abrufen von Secrets für die Protokollierungskonfiguration](#)

Amazon ECS unterstützt FSx-for-Windows-File-Server-Volumes für Container. Amazon ECS verwendet die in einem Secrets-Manager-Secret gespeicherten Anmeldeinformationen, um der Active-Directory-Domain beizutreten und das FSx-for-Windows-File-Server-Dateisystem anzuhängen. Weitere Informationen finden Sie unter [Tutorial: Verwenden von FSx-for-Windows-File-Server-Dateisystemen mit Amazon ECS](#) und [FSx-for-Windows-File-Server-Volumes](#) im Entwicklerhandbuch von Amazon Elastic Container Service.

Sie können auf Container-Images in privaten Registern verweisen, für AWS die keine Authentifizierung erforderlich ist, indem Sie ein Secrets Manager Manager-Geheimnis mit den

Anmeldeinformationen der Registrierung verwenden. Weitere Informationen finden Sie unter [Private Registrierungsauthentifizierung für Aufgaben](#) im Entwicklerhandbuch von Amazon Elastic Container Service.

Wenn Sie Amazon ECS Service Connect verwenden, verwendet Amazon ECS [verwaltete Secrets von Secrets Manager](#) zum Speichern von AWS Private Certificate Authority TLS-Zertifikaten. Die Kosten für die Aufbewahrung des Geheimnisses sind in den Gebühren für Amazon ECS enthalten. Um den Secret zu aktualisieren, müssen Sie Amazon ECS anstelle von Secrets Manager verwenden. Weitere Informationen finden Sie unter [TLS with Service Connect](#) im Amazon Elastic Container Service Developer Guide.

So ElastiCache nutzt Amazon AWS Secrets Manager

Darin können ElastiCache Sie eine Funktion namens Role-Based Access Control (RBAC) verwenden, um den Cluster zu sichern. Sie können diese Anmeldeinformationen in Secrets Manager speichern. Secrets Manager bietet eine [Rotationsvorlage](#) für diese Art von Secret. Weitere Informationen finden Sie unter [Automatisches Rotieren von Passwörtern für Benutzer](#) im ElastiCache Amazon-Benutzerhandbuch.

Wie AWS Elemental Live verwendet AWS Secrets Manager

AWS Elemental Live ist ein Echtzeit-Videodienst, mit dem Sie Live-Ausgaben für die Übertragung und Streaming-Übertragung erstellen können.

AWS Elemental Live verwendet einen geheimen ARN, um ein Geheimnis, das einen Verschlüsselungsschlüssel enthält, von Secrets Manager abzurufen. Elemental Live verwendet den Verschlüsselungsschlüssel, um das Video zu verschlüsseln/zu entschlüsseln. Weitere Informationen finden Sie unter [So MediaConnect funktioniert AWS Elemental Live die Lieferung von nach zur Laufzeit im](#) Elemental Live-Benutzerhandbuch.

Wie verwendet AWS Elemental MediaConnect AWS Secrets Manager

AWS Elemental MediaConnect ist ein Dienst, der es Rundfunkanstalten und anderen Premium-Videoanbietern leicht macht, Live-Videos zuverlässig in den aufzunehmen AWS Cloud und an mehrere Ziele innerhalb oder außerhalb des Landes zu verteilen. AWS Cloud

Sie können die statische Schlüsselverschlüsselung verwenden, um Ihre Quellen, Ausgaben und Berechtigungen zu schützen, und Sie speichern Ihren Verschlüsselungsschlüssel in AWS Secrets Manager. Weitere Informationen finden Sie unter [Verschlüsselung mit statischem Schlüssel AWS Elemental MediaConnect im AWS Elemental MediaConnect Benutzerhandbuch](#).

Wie AWS Elemental MediaConvert verwendet AWS Secrets Manager

AWS Elemental MediaConvert ist ein dateibasierter Videoverarbeitungsdienst, der skalierbare Videoverarbeitung für Eigentümer und Vertreiber von Inhalten mit Medienbibliotheken jeder Größe bietet. Um Kantar-Wasserzeichen MediaConvert zu kodieren, verwenden Sie Secrets Manager, um Ihre Kantar-Anmeldeinformationen zu speichern. Weitere Informationen finden Sie im Benutzerhandbuch unter [Verwenden von Kantar für Audio-Wasserzeichen in AWS Elemental MediaConvert](#) Ausgängen. AWS Elemental MediaConvert

Wie verwendet AWS Elemental MediaLive AWS Secrets Manager

AWS Elemental MediaLive ist ein Echtzeit-Videodienst, mit dem Sie Live-Ausgaben für die Übertragung und Streaming-Übertragung erstellen können. Wenn Ihre Organisation AWS Elemental Link Geräte mit AWS Elemental MediaLive oder verwendet AWS Elemental MediaConnect, müssen Sie das Gerät bereitstellen und konfigurieren. Weitere Informationen finden Sie im MediaLive Benutzerhandbuch unter [Einrichtung MediaLive als vertrauenswürdige Entität](#).

Wie AWS Elemental MediaPackage verwendet AWS Secrets Manager

AWS Elemental MediaPackage ist ein Dienst zur Paketierung und Erstellung von just-in-time Videos, der in der AWS Cloud ausgeführt wird. Mit MediaPackage können Sie hochsichere, skalierbare und zuverlässige Videostreams für eine Vielzahl von Wiedergabegeräten und Content Delivery Networks (CDNs) bereitstellen. Weitere Informationen finden Sie unter [Secrets Manager Manager-Zugriff für die CDN-Autorisierung](#) im AWS Elemental MediaPackage Benutzerhandbuch.

Wie verwendet AWS Elemental MediaTailor AWS Secrets Manager

AWS Elemental MediaTailor ist ein skalierbarer Service zum Einfügen von Anzeigen und zum Zusammenstellen von Kanälen, der in der ausgeführt wird AWS Cloud.

MediaTailor unterstützt die Secrets Manager Manager-Zugriffstoken-Authentifizierung für Ihre Quellstandorte. Bei der Secrets Manager Manager-Zugriffstoken-Authentifizierung MediaTailor wird ein Secrets Manager Manager-Geheimnis verwendet, um Anfragen an Ihren Ursprung zu authentifizieren. Weitere Informationen finden Sie unter [Konfiguration der AWS Secrets Manager Zugriffstoken-Authentifizierung](#) im AWS Elemental MediaTailor Benutzerhandbuch.

So verwendet Amazon EMR Secrets Manager

Amazon EMR ist eine Plattform, die die Ausführung von Big-Data-Frameworks wie Apache Hadoop und Apache Spark vereinfacht, AWS um riesige Datenmengen zu verarbeiten und zu analysieren. Durch die Verwendung dieser Frameworks und verwandter Open-Source-Projekte, wie z. B. Apache Hive und Apache Pig, können Sie Daten zur Analyse und Business-Intelligence-Workloads verarbeiten. Sie können Amazon EMR auch verwenden, um große Datenmengen in und aus anderen AWS Datenspeichern und Datenbanken wie Amazon S3 und Amazon DynamoDB zu transformieren und zu verschieben.

So verwendet Amazon EMR, das in Amazon EC2 ausgeführt wird, Secrets Manager

Wenn Sie einen Cluster in Amazon EMR erstellen, können Sie dem Cluster Anwendungskonfigurationsdaten zur Verfügung stellen, indem Sie ein Secret in Secrets Manager verwenden. Weitere Informationen finden Sie unter [Speichern sensibler Konfigurationsdaten in Secrets Manager](#) im Amazon-EMR-Managementhandbuch.

Wenn Sie außerdem ein EMR-Notebook erstellen, können Sie mit Secrets Manager Ihre privaten Git-basierten Registrierungsinformationen speichern. Weitere Informationen finden Sie unter [Hinzufügen eines Git-basierten Repositorys zu Amazon EMR](#) im Amazon-EMR-Managementhandbuch.

So verwendet EMR Serverless Secrets Manager

EMR Serverless bietet eine serverlose Laufzeitumgebung, um den Betrieb von Analyseanwendungen zu vereinfachen, sodass Sie Cluster nicht konfigurieren, optimieren, sichern oder betreiben müssen.

Sie können Ihre Daten in Ihren EMR Serverless-Konfigurationen speichern AWS Secrets Manager und dann die geheime ID verwenden. Auf diese Weise geben Sie keine vertraulichen Konfigurationsdaten im Klartext weiter und stellen sie externen APIs zur Verfügung.

Weitere Informationen finden Sie unter [Secrets Manager für Datenschutz mit EMR Serverless](#) im Amazon EMR Serverless – Benutzerhandbuch.

So EventBridge nutzt Amazon AWS Secrets Manager

Amazon EventBridge ist ein serverloser Event-Bus-Service, mit dem Sie Ihre Anwendungen mit Daten aus einer Vielzahl von Quellen verbinden können.

Wenn Sie ein EventBridge Amazon-API-Ziel erstellen, EventBridge speichert es die Verbindung dafür in einem von Secrets Manager [verwalteten geheimen Schlüssel](#) mit dem Präfix `events.` Die Kosten für die Speicherung des Geheimnisses sind in der Gebühr für die Verwendung eines API-Ziels enthalten. Um den Secret zu aktualisieren, müssen Sie Secrets Manager EventBridge anstelle von Secrets Manager verwenden. Weitere Informationen finden Sie unter [API-Ziele](#) im EventBridge Amazon-Benutzerhandbuch.

Wie Amazon FSx Geheimnisse verwendet AWS Secrets Manager

Amazon FSx for Windows File Server bietet vollständig verwaltete Microsoft-Windows-Dateiserver, die von einem vollständig nativen Windows-Dateisystem unterstützt werden. Wenn Sie Dateifreigaben erstellen oder verwalten, können Sie Anmeldeinformationen aus einem AWS Secrets Manager geheimen Schlüssel weitergeben. Weitere Informationen finden Sie unter [Dateifreigaben](#) und [Migrieren von Dateifreigabekonfigurationen auf Amazon FSx](#) im Benutzerhandbuch zu Amazon FSx für Windows File Server.

Wie AWS Glue DataBrew verwendet AWS Secrets Manager

AWS Glue DataBrew ist ein Tool zur visuellen Datenvorbereitung, mit dem Sie Daten bereinigen und normalisieren können, ohne Code schreiben zu müssen. In DataBrew wird eine Reihe von Schritten zur Datentransformation als Rezept bezeichnet. AWS Glue DataBrew stellt die [DETERMINISTIC_DECRYPT/DETERMINISTIC_ENCRYPT](#),- und [CRYPTOGRAPHIC_HASH](#)Rezeptschritte zur Durchführung von Transformationen personenbezogener Daten (PII) in einem Datensatz bereit, die einen Verschlüsselungsschlüssel verwenden, der in einem Secrets Manager-Geheimnis gespeichert ist. Wenn Sie das DataBrew Standardgeheimnis zum Speichern des Verschlüsselungsschlüssels verwenden, DataBrew erstellt [es ein verwaltetes Geheimnis](#) mit dem Präfix `databrew.` Die Kosten für die Speicherung des Geheimnisses sind in der Nutzungsgebühr enthalten DataBrew. Wenn Sie ein neues Geheimnis zum Speichern des Verschlüsselungsschlüssels erstellen, DataBrew wird ein Geheimnis mit dem Präfix `databrew.` erstellt. Ihnen werden Kosten für dieses Geheimnis berechnet.

Wie verwendet AWS Glue Studio AWS Secrets Manager

AWS Glue Studio ist eine grafische Oberfläche, die das Erstellen, Ausführen und Überwachen von Extraktions-, Transformations- und Ladeaufträgen (ETL) auf einfache Weise ermöglicht AWS Glue. Sie können Amazon OpenSearch Service als Datenspeicher für Ihre ETL-Jobs (Extrahieren, Transformieren und Laden) verwenden, indem Sie den Elasticsearch Spark Connector in AWS Glue Studio konfigurieren. Um eine Verbindung zum OpenSearch Cluster herzustellen, können Sie ein Geheimnis in Secrets Manager verwenden. Weitere Informationen finden Sie unter [Tutorial: Using the AWS Glue Connector for Elasticsearch](#) im AWS Glue Developer Guide.

Wie verwendet AWS IoT SiteWise AWS Secrets Manager

AWS IoT SiteWise ist ein verwalteter Service, mit dem Sie Daten von Industrieanlagen in großem Maßstab sammeln, modellieren, analysieren und visualisieren können. Sie können die AWS IoT SiteWise Konsole verwenden, um ein Gateway zu erstellen. Fügen Sie dann Datenquellen, lokale Server oder Industrieanlagen hinzu, die mit Gateways verbunden sind. Wenn Ihre Quelle eine Authentifizierung erfordert, verwenden Sie ein Secret zum Authentifizieren. Weitere Informationen finden Sie unter [Konfigurieren der Datenquellenauthentifizierung](#) im Benutzerhandbuch für AWS IoT SiteWise .

So verwendet Amazon Kendra AWS Secrets Manager

Amazon Kendra ist ein hochgenauer und intelligenter Suchservice, der es Ihren Benutzern ermöglicht, unstrukturierte und strukturierte Daten mithilfe natürlicher Sprachverarbeitung und erweiterten Suchalgorithmen zu durchsuchen.

Sie können Dokumente indizieren, die in einer Datenbank gespeichert sind, indem Sie ein Secret angeben, das Anmeldeinformationen für die Datenbank enthält. Weitere Informationen finden Sie unter [Verwenden einer Datenbankdatenquelle](#) im Amazon-Kendra-Benutzerhandbuch.

So verwendet Amazon Kinesis Video Streams AWS Secrets Manager

Sie können Amazon Kinesis Video Streams verwenden, um eine Verbindung zu IP-Kameras an Kundenstandorten herzustellen, Videos von den Kameras lokal aufzuzeichnen und zu speichern und Videos zur Langzeitspeicherung, Wiedergabe und analytischen Verarbeitung in die Cloud zu

streamen. Um Medien von IP-Kameras aufzuzeichnen und hochzuladen, stellen Sie den Kinesis Video Streams Edge Agent in AWS IoT Greengrass bereit. Die Anmeldeinformationen, die für den Zugriff auf die Mediendateien erforderlich sind, die an die Kamera gestreamt werden, speichern Sie in einem Secrets-Manager-Secret. Weitere Informationen finden Sie unter [Deploy the Amazon Kinesis Video Streams Edge Agent to AWS IoT Greengrass](#) im Entwicklerhandbuch für Amazon Kinesis Video Streams.

Wie verwendet AWS Launch Wizard AWS Secrets Manager

AWS Launch Wizard for Active Directory ist ein Dienst, der bewährte AWS Cloud Anwendungsmethoden anwendet, um Sie bei der Einrichtung einer neuen Active Directory-Infrastruktur oder beim Hinzufügen von Domänencontrollern zu einer vorhandenen Infrastruktur, entweder vor Ort AWS Cloud oder vor Ort, zu unterstützen.

AWS Launch Wizard erfordert das Hinzufügen von Domänenadministratoranmeldedaten zu Secrets Manager, um Ihre Domänencontroller mit Active Directory zu verbinden. Weitere Informationen finden Sie unter [Einrichtung AWS Launch Wizard für Active Directory](#) im AWS Launch Wizard Benutzerhandbuch.

So verwendet Amazon Lookout for Metrics AWS Secrets Manager

Amazon Lookout for Metrics ist ein Service, der Anomalien in Ihren Daten findet, deren Ursachen bestimmt und es Ihnen ermöglicht, schnell Maßnahmen zu ergreifen. Sie können Amazon Redshift oder Amazon RDS als Datenquelle für einen Detektor für Lookout for Metrics verwenden. Um die Datenquelle zu konfigurieren, verwenden Sie ein Secret, das das Datenbankpasswort enthält. Weitere Informationen finden Sie unter [Verwenden von Amazon RDS mit Lookout für Metrics](#) und [Verwenden von Amazon Redshift mit Lookout für Metrics](#) im Entwicklerhandbuch für Amazon Lookout für Metrics.

So verwendet Amazon Managed Grafana AWS Secrets Manager

Amazon Managed Grafana ist ein vollständig verwalteter und sicherer Service zur Datenvisualisierung, mit dem Sie Betriebsmetriken, Protokolle und Ablaufverfolgungen aus mehreren Quellen sofort abfragen, korrelieren und visualisieren können. Wenn Sie Amazon Redshift als Datenquelle verwenden, können Sie Amazon Redshift-Anmeldeinformationen mithilfe eines AWS Secrets Manager Geheimnisses angeben. Weitere Informationen finden Sie unter [Konfigurieren von Amazon Redshift](#) im Benutzerhandbuch zu Amazon Managed Grafana.

Wie verwendet AWS Managed Services AWS Secrets Manager

AWS Managed Services ist ein Unternehmensservice, der die kontinuierliche Verwaltung Ihrer AWS Infrastruktur ermöglicht. Der AMS Self-Service Provisioning (SSP) -Modus bietet vollen Zugriff auf native Funktionen AWS-Service und API-Funktionen in von AMS verwalteten Konten. Informationen darüber, wie Sie Zugriff auf Secrets Manager in AMS anfordern können, finden Sie unter [AWS Secrets Manager \(AMS Self-Service Provisioning\)](#) im AMS-Advanced-Benutzerhandbuch.

So verwendet Amazon Managed Streaming for Apache Kafka AWS Secrets Manager

Amazon Managed Streaming for Apache Kafka (Amazon MSK) ist ein vollständig verwalteter Service, mit dem Sie Anwendungen erstellen und ausführen können, die Apache Kafka zum Verarbeiten von Streaming-Daten verwenden. Sie können den Zugriff auf Ihre Amazon-MSK-Cluster mit Benutzernamen und Passwörtern steuern, die mit AWS Secrets Manager gespeichert und gesichert wurden. Weitere Informationen finden Sie unter [Benutzername und Kennwortauthentifizierung mit AWS Secrets Manager](#) im Entwicklerhandbuch für Amazon Managed Streaming for Apache Kafka.

So verwendet Amazon Managed Workflows für Apache Airflow AWS Secrets Manager

Amazon Managed Workflows for Apache Airflow ist ein verwalteter Orchestrierungsservice für [Apache Airflow](#), der die Einrichtung und den Betrieb von end-to-end Daten-Pipelines in der Cloud in großem Umfang erleichtert.

Sie können eine Apache-Airflow-Verbindung mit einem Secrets-Manager-Secret konfigurieren. Weitere Informationen finden Sie unter [Konfiguration einer Apache Airflow-Verbindung mithilfe eines Secrets Manager-Geheimnisses](#) und [Verwenden eines geheimen Schlüssels AWS Secrets Manager für eine Apache Airflow-Variable](#) im Amazon Managed Workflows for Apache Airflow-Benutzerhandbuch.

AWS Marketplace

Wenn Sie AWS Marketplace Quick Launch verwenden, AWS Marketplace verteilt Ihre Software zusammen mit dem Lizenzschlüssel. AWS Marketplace speichert den Lizenzschlüssel in Ihrem Konto als von Secrets Manager [verwaltetes Geheimnis](#). Die Kosten für die Speicherung des Geheimnisses

sind in den Gebühren für enthalten AWS Marketplace. Um den Secret zu aktualisieren, müssen Sie Secrets Manager AWS Marketplace anstelle von Secrets Manager verwenden. Weitere Informationen finden Sie unter [Konfigurieren der Schnellkonfiguration](#) im AWS Marketplace -Verkäuferhandbuch.

Wie AWS Migration Hub verwendet AWS Secrets Manager

AWS Migration Hub bietet einen zentralen Ort, an dem Sie Migrationsaufgaben für mehrere AWS Tools und Partnerlösungen verfolgen können.

AWS Migration Hub Orchestrator vereinfacht und automatisiert die Migration von Servern und Unternehmensanwendungen zu. AWS Migration Hub Orchestrator verwendet ein Secret für die Verbindungsinformationen zu Ihrem Quellserver. Weitere Informationen finden Sie im AWS Migration Hub -Orchestrator-Benutzerhandbuch unter:

- [Migrieren Sie SAP-Anwendungen NetWeaver zu AWS](#)
- [Hostwechsel von Anwendungen auf Amazon EC2](#)

Migration Hub Strategy Recommendations bietet Empfehlungen für Migration und Modernisierungsstrategie für tragfähige Transformationspfade für Ihre Anwendungen. Strategy Recommendations kann SQL-Server-Datenbanken analysieren und dabei ein Secret für die Verbindungsinformationen verwenden. Weitere Informationen finden Sie unter [Datenbankanalyse mit Strategy Recommendation](#).

Wie AWS Panorama verwendet Secrets Manager

AWS Panorama ist ein Dienst, der Computer Vision in Ihr lokales Kameranetzwerk bringt. Sie werden verwendet, AWS Panorama um eine Appliance zu registrieren, ihre Software zu aktualisieren und Anwendungen darauf bereitzustellen. Wenn Sie einen Videostream als Datenquelle für Ihre Anwendung registrieren und der Stream passwortgeschützt ist, werden die Anmeldeinformationen dafür in einem Secrets Manager Secret AWS Panorama gespeichert. Weitere Informationen finden Sie unter [Verwalten von Kamerastreams in AWS Panorama](#) im AWS Panorama -Entwicklerhandbuch.

Wie AWS ParallelCluster verwendet AWS Secrets Manager

AWS ParallelCluster ist ein Open-Source-Clusterverwaltungstool, mit dem Sie HPC-Cluster (High Performance Computing) bereitstellen und verwalten können. AWS Cloud Sie können eine

Umgebung mit mehreren Benutzern erstellen, die eine enthält AWS ParallelCluster , die in ein AWS verwaltetes Microsoft AD (Active Directory) integriert ist. Der AWS ParallelCluster verwendet ein Secrets Manager Manager-Geheimnis zur Validierung von Anmeldungen bei Active Directory. Weitere Informationen finden Sie unter [Integrating Active Directory](#) im Benutzerhandbuch für AWS ParallelCluster .

So verwendet Amazon Q Secrets Manager

Um Amazon Q für den Zugriff auf Ihre Datenquelle zu authentifizieren, stellen Sie Amazon Q Ihre Zugangsdaten für die Datenquelle mithilfe eines Secrets Manager Manager-Geheimnisses zur Verfügung. Wenn Sie die Konsole verwenden, können Sie wählen, ob Sie ein neues Geheimnis erstellen oder ein vorhandenes verwenden möchten. Weitere Informationen finden Sie unter [Concepts — Authentication](#) im Amazon Q Developer Guide.

Wie AWS OpsWorks for Chef Automate verwendet AWS Secrets Manager

AWS OpsWorks ist ein Konfigurationsverwaltungsdienst, der Sie bei der Konfiguration und dem Betrieb von Anwendungen in einem Cloud-Unternehmen unterstützt, indem Sie OpsWorks für Puppet Enterprise oder AWS OpsWorks for Chef Automate.

Wenn Sie einen neuen Server in erstellen AWS OpsWorks CM, speichert OpsWorks CM Informationen für den Server in einem von Secrets Manager [verwalteten geheimen Schlüssel](#) mit dem Präfixopsworks-cm. Die Kosten für das Secret sind in der Gebühr für enthalten AWS OpsWorks. Weitere Informationen finden Sie unter [Integrieren mit AWS Secrets Manager](#) im AWS OpsWorks - Benutzerhandbuch.

So QuickSight nutzt Amazon AWS Secrets Manager

Amazon QuickSight ist ein Business Intelligence (BI) -Service auf Cloud-Ebene, den Sie für Analysen, Datenvisualisierung und Berichterstattung verwenden können. Sie können eine Vielzahl von Datenquellen in Amazon verwenden QuickSight. Wenn Sie Datenbankanmeldeinformationen in Secrets Manager Secrets speichern, QuickSight kann Amazon diese Secrets verwenden, um eine Verbindung zu den Datenbanken herzustellen. Weitere Informationen finden Sie unter [Verwenden von AWS Secrets Manager Geheimnissen anstelle von Datenbankanmeldedaten QuickSight in Amazon](#) im QuickSight Amazon-Benutzerhandbuch.

Amazon RDS

Amazon Relational Database Service (Amazon RDS) ist ein Webservice, der das Einrichten, Betreiben und Skalieren einer relationalen Datenbank in der AWS Cloud vereinfacht.

Um die Masterbenutzeranmeldedaten für Amazon Relational Database Service (Amazon RDS), einschließlich Aurora, zu verwalten, kann Amazon RDS ein [verwaltetes Geheimnis](#) für Sie erstellen. Ihnen werden Kosten für dieses Geheimnis berechnet. Amazon RDS [verwaltet auch die Rotation](#) dieser Anmeldeinformationen. Weitere Informationen finden Sie unter [Passwortverwaltung mit Amazon RDS und AWS Secrets Manager](#) im Amazon-RDS-Benutzerhandbuch.

Weitere Amazon-RDS-Anmeldeinformationen finden Sie unter [the section called “Erstellen eines Datenbank-Secrets”](#).

Wenn Sie den Amazon-RDS-Abfrage-Editor verwenden, um eine Verbindung zu einer Datenbank herzustellen, können Sie Anmeldeinformationen für die Datenbank in Secrets Manager speichern. Weitere Informationen finden Sie unter [Verwenden des Abfrage-Editors](#) im Amazon-RDS-Benutzerhandbuch.

So verwendet Amazon Redshift AWS Secrets Manager

Amazon Redshift ist ein vollständig verwalteter Data-Warehouse-Service in Petabytegröße in der Cloud.

Um Administratoranmeldedaten für Amazon Redshift zu verwalten, kann Amazon Redshift ein [verwaltetes Geheimnis](#) für Sie erstellen. Ihnen werden Kosten für dieses Geheimnis berechnet. Amazon Redshift [verwaltet auch die Rotation](#) für diese Anmeldeinformationen. Weitere Informationen finden Sie unter [Verwaltung von Amazon Redshift-Administratorkennwörtern mithilfe von AWS Secrets Manager](#) im Amazon Redshift Management Guide.

Weitere Amazon Redshift-Anmeldeinformationen finden Sie unter [the section called “Erstellen eines Datenbank-Secrets”](#).

Wenn Sie die Daten-API von Amazon Redshift aufrufen, können Sie die Anmeldeinformationen für den Cluster unter Verwendung eines Secrets in Secrets Manager übergeben. Weitere Informationen finden Sie unter [Verwenden der Amazon-Redshift-Daten-API](#).

Wenn Sie den Amazon-Redshift-Abfrage-Editor verwenden, um eine Verbindung zu einer Datenbank herzustellen, kann Amazon Redshift Ihre Anmeldeinformationen in einem Secrets-Manager-

Geheimnis mit dem Präfix `redshiftqueryeditor` speichern. Ihnen werden Kosten für dieses Geheimnis berechnet. Weitere Informationen finden Sie unter [Abfragen einer Datenbank mit dem Abfrage-Editor](#) im Amazon-Redshift-Verwaltungshandbuch.

Für den Abfrage-Editor v2, siehe [the section called “Amazon-Redshift-Abfrage-Editor v2”](#).

Amazon-Redshift-Abfrage-Editor v2

Der Amazon-Redshift-Abfrage-Editor v2 ist eine webbasierte SQL-Client-Anwendung, mit der Sie Abfragen in Ihrem Amazon-Redshift-Data-Warehouse erstellen und ausführen können. Wenn Sie den Amazon Redshift Query Editor v2 verwenden, um eine Verbindung zu einer Datenbank herzustellen, kann Amazon Redshift Ihre Anmeldeinformationen in einem Secrets Manager [Managed Secret mit dem Präfix](#) `sqlworkbench` speichern. Die Kosten für die Speicherung des Geheimnisses sind in der Gebühr für die Nutzung von Amazon Redshift enthalten. Um das Geheimnis zu aktualisieren, müssen Sie Amazon Redshift anstelle von Secrets Manager verwenden. Weitere Informationen finden Sie unter [Arbeiten mit dem Abfrage-Editor v2](#) im Amazon-Redshift-Verwaltungshandbuch.

Informationen zum vorherigen Abfrage-Editor finden Sie unter [the section called “Amazon-Redshift”](#).

So SageMaker nutzt Amazon AWS Secrets Manager

SageMaker ist ein vollständig verwalteter Service für maschinelles Lernen. Damit SageMaker können Datenwissenschaftler und Entwickler schnell und einfach Modelle für maschinelles Lernen erstellen und trainieren und sie dann direkt in einer produktionsbereiten gehosteten Umgebung bereitstellen. Er bietet eine integrierte Jupyter-Authoring-Notebook-Instance für den einfachen Zugriff auf Ihre Datenquellen zur Durchsuchung und Analyse, sodass Sie keine Server verwalten müssen.

Sie können Git-Repositorys mit Ihrer Jupyter-Notebook-Instance verknüpfen, um Ihre Notebooks in einer Quellkontrollumgebung zu speichern, die auch dann bestehen bleibt, wenn Sie Ihre Notebook-Instance stoppen oder löschen. Sie können Ihre privaten Repository-Anmeldeinformationen mit Secrets Manager verwalten. Weitere Informationen finden Sie unter [Verknüpfen von Git-Repositorys mit Amazon SageMaker Notebook-Instances](#) im Amazon SageMaker Developer Guide.

Um Daten aus Databricks zu importieren, speichert Data Wrangler Ihre JDBC-URL in Secrets Manager. Weitere Informationen finden Sie unter [Importieren von Daten aus Databricks \(JDBC\)](#).

Um Daten aus Snowflake zu importieren, speichert Data Wrangler Ihre Anmeldeinformationen in einem Secrets-Manager-Secret. Weitere Informationen finden Sie unter [Importieren von Daten aus Snowflake](#).

Wie verwendet AWS Schema Conversion Tool AWS Secrets Manager

Sie können das AWS Schema Conversion Tool (AWS SCT) verwenden, um Ihr vorhandenes Datenbankschema von einer Datenbank-Engine in eine andere zu konvertieren. Sie können relationale OLTP-Schemata sowie Data Warehouse-Schemata konvertieren. Ihr konvertiertes Schema ist für Amazon Relational Database Service (Amazon RDS) MySQL, MariaDB, Oracle, SQL Server, PostgreSQL DB, einen Amazon-Aurora-DB-Cluster oder einen Amazon-Redshift-Cluster geeignet. Das konvertierte Schema kann auch mit einer Datenbank auf einer Amazon Elastic Compute Cloud-Instance verwendet oder als Daten in einem S3-Bucket gespeichert werden.

Wenn Sie ein Datenbankschema konvertieren, AWS SCT können Sie Datenbankanmeldedaten verwenden, die Sie in speichern AWS Secrets Manager. Weitere Informationen finden Sie [AWS Secrets Manager im AWS SCT Benutzerhandbuch unter Verwendung in der AWS Schema Conversion Tool Benutzeroberfläche](#).

Wie AWS Toolkit for JetBrains verwendet AWS Secrets Manager

Das AWS Toolkit for JetBrains ist ein Open-Source-Plugin für die integrierten Entwicklungsumgebungen (IDEs) von JetBrains. Dieses Toolkit erleichtert Entwicklern das Entwickeln, Debuggen und Bereitstellen von Serverless-Anwendungen, die AWS verwenden. Wenn Sie sich mit dem Toolkit mit einem Amazon-Redshift-Cluster verbinden, können Sie sich mit einem Secrets-Manager-Secret authentifizieren. Weitere Informationen finden Sie unter [Zugriff auf Amazon-Redshift-Cluster](#) im AWS Toolkit for JetBrains -Benutzerhandbuch.

Wie AWS Transfer Family benutzt AWS Secrets Manager Secrets

AWS Transfer Family ist ein sicherer Übertragungsdienst, mit dem Sie Dateien in und aus AWS Speicherdiensten übertragen können.

Transfer Family unterstützt jetzt die Verwendung der Standardauthentifizierung für Server, die das Applicability Statement 2 (AS2)-Protokoll verwenden. Sie können ein neues Secrets-Manager-Secret erstellen oder ein vorhandenes Secret für Ihre Anmeldeinformationen auswählen. Weitere Informationen finden Sie unter [Standardauthentifizierung für AS2-Konnektoren](#) im Benutzerhandbuch von AWS Transfer Family .

Um Transfer Family Family-Benutzer zu authentifizieren, können Sie es AWS Secrets Manager als Identitätsanbieter verwenden. Weitere Informationen finden Sie im AWS Transfer Family

Benutzerhandbuch unter [Arbeiten mit benutzerdefinierten Identitätsanbietern](#) und im Blogartikel [Aktivieren der Kennwortauthentifizierung für die AWS Transfer Family Verwendung AWS Secrets Manager](#).

Sie können die PGP-Entschlüsselung (Pretty Good Privacy) für die Dateien verwenden, die Transfer Family mit Workflows verarbeitet. Um die Entschlüsselung in einem Workflow-Schritt zu verwenden, geben Sie einen PGP-Schlüssel an, den Sie in Secrets Manager verwalten. Weitere Informationen finden Sie unter [Generieren und Verwalten von PGP-Schlüsseln](#) im AWS Transfer Family -Benutzerhandbuch.

Wie AWS Wickr Geheimnisse verwendet AWS Secrets Manager

AWS Wickr ist ein end-to-end verschlüsselter Dienst, der Organisationen und Regierungsbehörden dabei hilft, sicher über one-to-one Gruppennachrichten, Sprach- und Videoanrufe, Dateifreigabe, Bildschirmübertragung und mehr zu kommunizieren. Sie können Workflows mithilfe von Wickr-Bots zur Datenaufbewahrung automatisieren. Wenn der Bot Zugriff darauf hat AWS-Services, sollten Sie ein Secrets Manager Manager-Geheimnis erstellen, um die Bot-Anmeldeinformationen zu speichern. Weitere Informationen finden Sie unter [Starten des Bots zur Datenaufbewahrung](#) im AWS -Wickr-Administrationshandbuch.

Verwenden eines AWS Secrets Manager-VPC-Endpunkts

Wir empfehlen, einen Großteil der Infrastruktur in privaten Netzwerken auszuführen, die vom öffentlichen Internet aus nicht zugänglich sind, sofern dies möglich ist. Sie können eine private Verbindung zwischen Ihrer VPC und Secrets Manager herstellen, indem Sie einen Schnittstellen-VPC-Endpunkt erstellen. Schnittstellen-Endpunkte werden mit [AWS PrivateLink](#) bereitgestellt, einer Technologie, die es Ihnen ermöglicht, ohne Internet-Gateway, NAT-Gerät, VPN-Verbindung oder AWS Direct Connect-Verbindung privat auf Secrets-Manager-APIs zuzugreifen. Die Instances in Ihrer VPC benötigen für die Kommunikation mit Secrets-Manager-APIs keine öffentlichen IP-Adressen. Der Datenverkehr zwischen Ihrer VPC und Secrets Manager verlässt das AWS-Netzwerk nicht. Weitere Informationen finden Sie unter [Schnittstellen-VPC-Endpunkte \(AWS PrivateLink\)](#) im Amazon-VPC-Benutzerhandbuch.

Wenn Secrets Manager [ein Secret mithilfe einer Lambda-Rotationsfunktion rotiert](#), beispielsweise ein Secret, das Datenbankanmeldeinformationen enthält, stellt die Lambda-Funktion Anfragen sowohl an die Datenbank als auch an Secrets Manager. Wenn Sie die [automatische Rotation in der Konsole aktivieren](#), erstellt Secrets Manager die Lambda-Funktion in derselben VPC wie Ihre Datenbank. Wir empfehlen Ihnen, einen Secrets-Manager-Endpunkt in derselben VPC zu erstellen, damit Anfragen von der Lambda-Rotationsfunktion an Secrets Manager das Amazon-Netzwerk nicht verlassen.

Wenn Sie einen privaten DNS für den Endpunkt aktivieren, können Sie mittels seines standardmäßigen DNS-Namen für die Region, beispielsweise `secretsmanager.us-east-1.amazonaws.com`, API-Anforderungen an Secrets Manager senden. Weitere Informationen finden Sie unter [Zugriff auf einen Service über einen Schnittstellenendpunkt](#) im Benutzerhandbuch für Amazon VPC.

Sie können sicherstellen, dass Anforderungen an Secrets Manager von der VPC stammen, indem Sie eine Bedingung in Ihre Berechtigungsrichtlinien aufnehmen. Weitere Informationen finden Sie unter [the section called "Beispiel: Berechtigungen und VPCs"](#).

Sie können anhand von AWS CloudTrail-Protokollen Ihre Verwendung von Geheimnissen durch den VPC-Endpunkt überwachen.

So erstellen Sie einen privaten Secrets-Manager-VPC-Endpunkt

1. Weitere Informationen finden Sie unter [Erstellen eines Schnittstellenendpunkts](#) im Amazon-VPC-Benutzerhandbuch. Verwenden Sie den Servicenamen: `com.amazonaws.region.secretsmanager`

2. Informationen zum Steuern des Zugriffs auf den Endpunkt finden Sie unter [Steuern des Zugriffs auf VPC-Endpunkte mithilfe von Endpunktrichtlinien](#).

Gemeinsam genutzte Subnetze

Sie können VPC-Endpunkte in Subnetzen, die mit Ihnen geteilt werden, nicht erstellen, beschreiben, ändern oder löschen. Sie können die VPC-Endpunkte jedoch in Subnetzen verwenden, die mit Ihnen geteilt werden. Informationen zur VPC-Freigabe finden Sie unter [Freigeben Ihrer VPC für andere Konten](#) im Benutzerhandbuch für Amazon Virtual Private Cloud.

Erstellen von AWS Secrets Manager-Secrets in AWS CloudFormation

Sie können Secrets in einem CloudFormation-Stack erstellen, indem Sie die [AWS::SecretsManager::Secret](#)-Ressource in einer CloudFormation-Vorlage verwenden, wie in [Ein Secret erstellen](#) gezeigt.

Um ein Admin-Secret für Amazon RDS oder Aurora zu erstellen, empfehlen wir Ihnen, `ManageMasterUserPassword` in [AWS::RDS::DBCluster](#) zu verwenden. Dann erstellt Amazon RDS das Secret und verwaltet die Rotation für Sie. Weitere Informationen finden Sie unter [Verwaltete Rotation](#).

Für Amazon-Redshift- und Amazon-DocumentDB-Anmeldeinformationen erstellen Sie zunächst ein Secret mit einem von Secrets Manager generierten Passwort und verwenden dann eine [dynamische Referenz](#), um den Benutzernamen und das Passwort aus dem Secret abzurufen und als Anmeldeinformationen für eine neue Datenbank zu verwenden. Als nächstes verwenden Sie die [AWS::SecretsManager::SecretTargetAttachment](#)-Ressource, um dem Secret Details über die Datenbank hinzuzufügen, die Secrets Manager benötigt, um das Secret zu rotieren. Verwenden Sie abschließend die [AWS::SecretsManager::RotationSchedule](#)-Ressource und geben Sie eine [Rotationsfunktion](#) und einen [Zeitplan](#) an, um die automatische Rotation zu aktivieren. Im Folgenden sind einige Beispiele aufgeführt:

- [Erstellen eines Secrets mit Amazon-Redshift-Anmeldeinformationen](#)
- [Erstellen eines Secrets mit Amazon-DocumentDB-Anmeldeinformationen](#)

Um eine Ressourcenrichtlinie an Ihr Geheimnis anzufügen, verwenden Sie die Ressource [AWS::SecretsManager::ResourcePolicy](#).

Weitere Informationen über die Erstellung von Ressourcen mit AWS CloudFormation finden Sie unter [Informationen über Vorlagengrundlagen](#) im AWS CloudFormation-Benutzerhandbuch. Sie können auch die AWS Cloud Development Kit (AWS CDK) verwenden. Weitere Informationen finden Sie unter [AWS Secrets Manager Construct Library](#).

Erstellen eines AWS Secrets Manager-Secrets mit AWS CloudFormation

In diesem Beispiel wird ein Secret mit dem Namen

CloudFormationCreatedSecret-*a1b2c3d4e5f6* erstellt. Der Secret-Wert ist der folgende JSON mit einem Passwort mit 32 Zeichen, das beim Erstellen des Secrets generiert wird.

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

In diesem Beispiel wird die folgende CloudFormation-Ressource verwendet:

- [AWS::SecretsManager::Secret](#)

Weitere Informationen über die Erstellung von Ressourcen mit AWS CloudFormation finden Sie unter [Informationen über Vorlagengrundlagen](#) im AWS CloudFormation-Benutzerhandbuch.

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
```

Erstellen eines AWS Secrets Manager-Secrets mit automatischer Drehung und einer MySQL-DB-Instance von Amazon RDS mit AWS CloudFormation

Um ein Admin-Secret für Amazon RDS oder Aurora zu erstellen, empfehlen wir die Verwendung von `ManageMasterUserPassword`, wie im Beispiel Erstellen eines Secrets-Manager-Secrets für ein Master-Passwort in [AWS::RDS::DBCluster](#) gezeigt. Dann erstellt Amazon RDS das Secret und verwaltet die Rotation für Sie. Weitere Informationen finden Sie unter [Verwaltete Rotation](#).

Erstellen Sie ein AWS Secrets Manager Geheimnis und einen Amazon Redshift Redshift-Cluster mit AWS CloudFormation

Um ein Administratorgeheimnis für Amazon Redshift zu erstellen, empfehlen wir Ihnen, die Beispiele auf [AWS::Redshift::Cluster](#) und [AWS::RedshiftServerless::Namespace](#) zu verwenden.

Erstellen Sie ein AWS Secrets Manager Geheimnis und eine Amazon DocumentDB DocumentDB-Instance mit AWS CloudFormation

In diesem Beispiel werden ein Geheimnis und eine Amazon-DocumentDB-Instance erstellt, wobei die Anmeldeinformationen im Geheimnis als Benutzer und Passwort verwendet werden. Das Secret verfügt über eine zugeordnete ressourcenbasierte Richtlinie, die angibt, wer Zugriff auf das Secret hat. Die Vorlage erstellt außerdem eine Lambda-Drehungsfunktion von den

[Rotationsfunktionsvorlagen](#) und konfiguriert das Geheimnis so, dass es am ersten Tag eines jeden Monats automatisch zwischen 8:00 und 10:00 Uhr UTC gedreht wird. Als bewährte Methode für Sicherheit befindet sich die Instance in einer Amazon VPC.

In diesem Beispiel werden die folgenden CloudFormation Ressourcen für Secrets Manager verwendet:

- [AWS::SecretsManager::Secret](#)
- [AWS::SecretsManager::SecretTargetAttachment](#)
- [AWS::SecretsManager::RotationSchedule](#)

Informationen zum Erstellen von Ressourcen mit AWS CloudFormation finden [Sie im AWS CloudFormation Benutzerhandbuch unter Grundlagen von Vorlagen lernen](#).

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 * ? *)

So verwendet Secrets Manager AWS CloudFormation

Wenn Sie die Konsole verwenden, um die Drehung zu aktivieren, verwendet Secrets Manager AWS CloudFormation, um Ressourcen für die Drehung zu erstellen. Wenn Sie während dieses Vorgangs eine neue Drehungsfunktion erstellen, erstellt AWS CloudFormation eine beliebige [AWS::Serverless::Function](#) basierend auf den entsprechenden [Rotationsfunktionsvorlagen](#). Dann legt AWS CloudFormation den [RotationSchedule](#) fest, das die Drehungsfunktion und die Drehungsregeln für das Secret festlegt. Sie können den AWS CloudFormation-Stack anzeigen, indem Sie View stack (Stack anzeigen) im Banner auswählen, nachdem Sie die automatische Drehung aktiviert haben.

Informationen zum Einschalten der automatischen Drehung finden Sie unter [Rotieren von - Geheimnissen](#).

Erstelle AWS Secrets Manager Geheimnisse in AWS Cloud Development Kit (AWS CDK)

Zum Erstellen, Verwalten und Abrufen von Secrets in einer CDK-Anwendung können Sie die [AWS Secrets Manager Construct Library](#) verwenden, die [ResourcePolicy](#)-, [RotationSchedule](#)-, [Secret](#)-, [SecretRotation](#)- und [SecretTargetAttachment](#)-Konstrukte enthält.

Eine bewährte Methode für die Verwendung von Geheimnissen in CDK-Anwendungen besteht darin, [das Geheimnis zunächst mithilfe der Konsole oder der CLI zu erstellen](#) und das Geheimnis dann in Ihre CDK-Anwendung zu importieren.

Beispiele finden Sie unter:

- [Ein Secret erstellen](#)
- [Ein Secret importieren](#)
- [Ein Secret abrufen](#)
- [Berechtigung zur Verwendung des Secrets gewähren](#)
- [Ein Secret rotieren](#)
- [Ein Datenbank-Secret rotieren](#)
- [Ein Secret in anderen Regionen replizieren](#)

Weitere Informationen zum CDK finden Sie im [AWS Cloud Development Kit \(AWS CDK\) -v2-Entwicklerhandbuch](#).

AWS Secrets Manager Geheimnisse überwachen

AWS bietet Überwachungstools, um Secrets Manager Manager-Geheimnisse zu überwachen, zu melden, wenn etwas nicht stimmt, und gegebenenfalls automatische Maßnahmen zu ergreifen. Sie können die Protokolle verwenden, wenn Sie unerwartete Nutzung oder Änderungen untersuchen müssen. Anschließend ist es möglich, unerwünschte Änderungen rückgängig zu machen. Darüber hinaus können Sie automatische Prüfungen auf unangemessene Nutzung von Secrets und alle Versuche zum Löschen von Secrets einrichten.

Themen

- [AWS Secrets Manager Ereignisse protokollieren mit AWS CloudTrail](#)
- [Überwachen Sie AWS Secrets Manager mit Amazon CloudWatch](#)
- [AWS Secrets Manager Ereignisse mit Amazon abgleichen EventBridge](#)
- [Überwachen Sie, wann auf AWS Secrets Manager Geheimnisse zugegriffen wird, die gelöscht werden sollen](#)
- [Überwachen Sie AWS Secrets Manager Geheimnisse auf Einhaltung der Vorschriften, indem Sie AWS Config](#)
- [Secrets Manager Manager-Kosten überwachen](#)

AWS Secrets Manager Ereignisse protokollieren mit AWS CloudTrail

AWS CloudTrail zeichnet alle API-Aufrufe für Secrets Manager als Ereignisse auf, einschließlich Aufrufe von der Secrets Manager Manager-Konsole sowie mehrere andere Ereignisse für die Rotation und das Löschen geheimer Versionen. Eine Liste der Protokolleinträge in Secrets Manager Manager-Datensätzen finden Sie unter [CloudTrail Einträge](#).

Sie können die CloudTrail Konsole verwenden, um die aufgezeichneten Ereignisse der letzten 90 Tage anzuzeigen. Für eine fortlaufende Aufzeichnung von Ereignissen in Ihrem AWS Konto, einschließlich Ereignissen für Secrets Manager, erstellen Sie einen Trail, sodass CloudTrail Protokolldateien an einen Amazon S3-Bucket gesendet werden. Weitere Informationen finden Sie unter [Einen Trail für Ihr AWS Konto erstellen](#). Sie können auch so konfigurieren CloudTrail , dass CloudTrail Protokolldateien von [mehreren AWS-Konten](#) und empfangen [AWS-Regionen](#) werden.

Sie können andere AWS Dienste so konfigurieren, dass sie die in den CloudTrail Protokollen gesammelten Daten weiter analysieren und darauf reagieren. Weitere Informationen finden Sie unter [AWS Serviceintegrationen mit CloudTrail Protokollen](#). Sie können auch Benachrichtigungen erhalten, wenn neue Protokolldateien in Ihrem Amazon S3 S3-Bucket CloudTrail veröffentlicht werden. Weitere Informationen finden Sie [unter Konfiguration von Amazon SNS SNS-Benachrichtigungen für CloudTrail](#).

Um Secrets Manager Manager-Ereignisse aus CloudTrail Protokollen abzurufen (Konsole)

1. Öffnen Sie die CloudTrail Konsole unter <https://console.aws.amazon.com/cloudtrail/>.
2. Stellen Sie sicher, dass die Konsole auf die Region zeigt, in der Ihre Ereignisse aufgetreten sind. In der Konsole werden nur die Ereignisse angezeigt, die in der ausgewählten Region aufgetreten sind. Wählen Sie die Region aus der Dropdownliste in der oberen rechten Ecke der Konsole aus.
3. Wählen Sie im Navigationsbereich links Event history (Ereignisverlauf) aus.
4. Wählen Sie Filter-Kriterien und/oder einen Time range (Zeitraum) aus, um das gesuchte Ereignis leichter zu finden. Beispielsweise:
 - a. Um alle Secrets Manager Manager-Ereignisse zu sehen, wählen Sie für Lookup-Attribute die Option Ereignisquelle. Wählen Sie für Enter event source (Ereignisquelle eingeben) die Option **secretsmanager.amazonaws.com** aus.
 - b. Um alle Ereignisse für ein Geheimnis anzuzeigen, wählen Sie für Lookup-Attribute die Option Ressourcennamen aus. Geben Sie dann unter Geben Sie einen Ressourcennamen ein den Namen des Geheimnisses ein.
5. Um weitere Details anzuzeigen, klicken Sie auf den Erweiterungspfeil neben dem Ereignis. Wenn Sie alle verfügbaren Informationen sehen möchten, wählen Sie View event (Ereignis anzeigen) aus.

AWS CLI

Example Secrets Manager Manager-Ereignisse aus CloudTrail Protokollen abrufen

Im folgenden [lookup-events](#)-Beispiel werden Secrets-Manager-Ereignisse gesucht.

```
aws cloudtrail lookup-events \  
  --region us-east-1 \  
  --lookup-attributes  
  AttributeKey=EventSource,AttributeValue=secretsmanager.amazonaws.com
```

AWS CloudTrail Einträge für Secrets Manager

AWS Secrets Manager schreibt Einträge in Ihr AWS CloudTrail Protokoll für alle Secrets Manager Manager-Operationen und für andere Ereignisse im Zusammenhang mit Rotation und Löschung. Informationen zum Ergreifen von Maßnahmen bei diesen Ereignissen finden Sie unter [Kombiniere Secrets Manager Manager-Ereignisse mit EventBridge](#).

Typen von Protokolleinträgen

- [Protokolleinträge für Secrets-Manager-Operationen](#)
- [Protokolleinträge zum Löschen](#)
- [Protokolleinträge für die Replikation](#)
- [Logeinträge für Rotation](#)

Protokolleinträge für Secrets-Manager-Operationen

Ereignisse, die durch Aufrufe von Secrets-Manager-Vorgängen generiert werden, haben "detail-type": ["AWS API Call via CloudTrail"].

Note

Vor Februar 2024 meldeten einige Secrets Manager Manager-Operationen Ereignisse, die „ARn“ statt „arn“ für den geheimen ARN enthielten. Weitere Informationen finden Sie unter [AWS re:Post](#).

Die folgenden CloudTrail Einträge werden generiert, wenn Sie oder ein Dienst Secrets Manager Manager-Operationen über die API, das SDK oder die CLI aufrufen.

BatchGetSecretValue

Durch den [BatchGetSecretValue](#)Vorgang generiert. Weitere Informationen zum Abrufen von Secrets finden Sie unter [Holen Sie sich Geheimnisse](#).

CancelRotateSecret

Durch die [CancelRotateSecret](#)Operation generiert. Weitere Informationen zur Rotation finden Sie unter [Rotieren von -Geheimnissen](#).

CreateSecret

Durch die [CreateSecret](#) Operation generiert. Weitere Informationen zum Erstellen von Secrets finden Sie unter [Erstellen und Verwalten von Secrets](#).

DeleteResourcePolicy

Durch die [DeleteResourcePolicy](#) Operation generiert. Weitere Informationen zu Berechtigungen finden Sie unter [Authentifizierung und Zugriffskontrolle](#).

DeleteSecret

Durch die [DeleteSecret](#) Operation generiert. Weitere Informationen zum Löschen von Secrets finden Sie unter [the section called "Löschen eines Secrets"](#).

DescribeSecret

Durch die [DescribeSecret](#) Operation generiert.

GetRandomPassword

Durch die [GetRandomPassword](#) Operation generiert.

GetResourcePolicy

Durch die [GetResourcePolicy](#) Operation generiert. Weitere Informationen zu Berechtigungen finden Sie unter [Authentifizierung und Zugriffskontrolle](#).

GetSecretValue

Generiert durch die [BatchGetSecretValue](#) Operationen [GetSecretValue](#) und. Weitere Informationen zum Abrufen von Secrets finden Sie unter [Holen Sie sich Geheimnisse](#).

ListSecrets

Wird durch die [ListSecrets](#) Operation generiert. Weitere Informationen zum Auflisten von Secrets finden Sie unter [the section called "Finden von Geheimnissen"](#).

ListSecretVersionIds

Durch die [ListSecretVersionIds](#) Operation generiert.

PutResourcePolicy

Durch die [PutResourcePolicy](#) Operation generiert. Weitere Informationen zu Berechtigungen finden Sie unter [Authentifizierung und Zugriffskontrolle](#).

PutSecretValue

Durch die [PutSecretValue](#) Operation generiert. Weitere Informationen zum Aktualisieren eines Secrets finden Sie unter [the section called “Ändern eines Secrets”](#).

RemoveRegionsFromReplication

Durch die [RemoveRegionsFromReplication](#) Operation generiert. Weitere Informationen zum Replizieren eines Secrets finden Sie unter [Replizieren Sie Geheimnisse regionsübergreifend](#).

ReplicateSecretToRegions

Durch die [ReplicateSecretToRegions](#) Operation generiert. Weitere Informationen zum Replizieren eines Secrets finden Sie unter [Replizieren Sie Geheimnisse regionsübergreifend](#).

RestoreSecret

Durch die [RestoreSecret](#) Operation generiert. Weitere Informationen zum Wiederherstellen eines gelöschten Secrets finden Sie unter [the section called “Wiederherstellen eines Secrets”](#).

RotateSecret

Durch die [RotateSecret](#) Operation generiert. Weitere Informationen zur Rotation finden Sie unter [Rotieren von -Geheimnissen](#).

StopReplicationToReplica

Durch die [StopReplicationToReplica](#) Operation generiert. Weitere Informationen zum Replizieren eines Secrets finden Sie unter [Replizieren Sie Geheimnisse regionsübergreifend](#).

TagResource

Durch die [TagResource](#) Operation generiert. Weitere Informationen zum Markieren eines Secrets finden Sie unter [the section called “-Secrets markieren”](#).

UntagResource

Durch die [UntagResource](#) Operation generiert. Weitere Informationen zum Aufheben der Markierung eines Secrets finden Sie unter [the section called “-Secrets markieren”](#).

UpdateSecret

Durch die [UpdateSecret](#) Operation generiert. Weitere Informationen zum Aktualisieren eines Secrets finden Sie unter [the section called “Ändern eines Secrets”](#).

UpdateSecretVersionStage

Durch die [UpdateSecretVersionStage](#) Operation generiert. Weitere Informationen zu Versionsstufen finden Sie unter [the section called "Geheime Versionen"](#).

ValidateResourcePolicy

Durch die [ValidateResourcePolicy](#) Operation generiert. Weitere Informationen zu Berechtigungen finden Sie unter [Authentifizierung und Zugriffskontrolle](#).

Protokolleinträge zum Löschen

Zusätzlich zu den Ereignissen für Secrets-Manager-Operationen generiert Secrets Manager die folgenden Ereignisse im Zusammenhang mit dem Löschen. Diese Ereignisse haben "detail-type": ["AWS Service Event via CloudTrail"].

CancelSecretVersionDelete

Generiert vom Secrets-Manager-Dienst. Wenn Sie `DeleteSecret` für ein Secret mit Versionen aufrufen und später `RestoreSecret` aufrufen, protokolliert Secrets Manager dieses Ereignis für jede wiederhergestellte Secret-Version. Weitere Informationen zum Wiederherstellen eines gelöschten Secrets finden Sie unter [the section called "Wiederherstellen eines Secrets"](#).

EndSecretVersionDelete

Wird generiert vom Secrets Manager, wenn eine Version eines Secrets gelöscht wurde. Weitere Informationen finden Sie unter [the section called "Löschen eines Secrets"](#).

StartSecretVersionDelete

Wird vom Secrets Manager generiert, wenn Secrets Manager mit dem Löschen einer Secret-Version beginnt. Weitere Informationen zum Löschen von Secrets finden Sie unter [the section called "Löschen eines Secrets"](#).

SecretVersionDeletion

Wird vom Secrets Manager generiert, wenn Secrets Manager eine veraltete Secret-Version löscht. Weitere Informationen hierzu finden Sie unter [Secret-Versionen](#).

Protokolleinträge für die Replikation

Zusätzlich zu den Ereignissen für Secrets-Manager-Operationen generiert Secrets Manager die folgenden Ereignisse im Zusammenhang mit der Replikation. Diese Ereignisse haben "detail-type": ["AWS Service Event via CloudTrail"].

ReplicationFailed

Wird vom Secrets-Manager-Service generiert, wenn die Replikation fehlschlägt. Weitere Informationen zum Replizieren eines Secrets finden Sie unter [Replizieren Sie Geheimnisse regionsübergreifend](#).

ReplicationStarted

Wird vom Secrets-Manager-Service generiert, wenn Secrets Manager mit der Replikation eines Secrets beginnt. Weitere Informationen zum Replizieren eines Secrets finden Sie unter [Replizieren Sie Geheimnisse regionsübergreifend](#).

ReplicationSucceeded

Wird von Secrets Manager generiert, wenn ein Secret erfolgreich repliziert wurde. Weitere Informationen zum Replizieren eines Secrets finden Sie unter [Replizieren Sie Geheimnisse regionsübergreifend](#).

Logeinträge für Rotation

Zusätzlich zu den Ereignissen für Secrets-Manager-Operationen generiert Secrets Manager die folgenden Ereignisse im Zusammenhang mit der Rotation. Diese Ereignisse haben "detail-type": ["AWS Service Event via CloudTrail"].

RotationStarted

Wird vom Secrets-Manager-Dienst generiert, wenn Secrets Manager mit der Rotation eines Secrets beginnt. Weitere Informationen zur Rotation finden Sie unter [Rotieren von -Geheimnissen](#).

RotationAbandoned

Wird generiert, wenn Secrets Manager einen Rotationsversuch abbricht und die AWSPENDING-Bezeichnung aus einer vorhandenen Version eines Secrets entfernt. Secrets Manager bricht die Rotation ab, wenn Sie während der Rotation eine neue Version eines Secrets erstellen. Weitere Informationen zur Drehung finden Sie unter [Rotieren von -Geheimnissen](#).

RotationFailed

Wird vom Secrets-Manager-Dienst generiert, wenn die Rotation fehlschlägt. Weitere Informationen zur Rotation finden Sie unter [the section called “Fehlerbehebung bei der -Rotation”](#).

RotationSucceeded

Wird generiert, wenn Secrets Manager ein Secret erfolgreich rotiert wurde. Weitere Informationen zur Rotation finden Sie unter [Rotieren von -Geheimnissen](#).

TestRotationStarted

Wird generiert, wenn Secrets Manager mit dem Testen der Rotation für ein Secret beginnt, das nicht für eine sofortige Rotation geplant ist. Weitere Informationen zur Rotation finden Sie unter [Rotieren von -Geheimnissen](#).

TestRotationSucceeded

Wird generiert, wenn Secrets Manager erfolgreich die Rotation für ein Secret testet, das nicht für eine sofortige Rotation geplant ist. Weitere Informationen zur Rotation finden Sie unter [Rotieren von -Geheimnissen](#).

TestRotationFailed

Wird generiert, wenn Secrets Manager die Rotation für ein Secret testet, das nicht für eine sofortige Rotation geplant ist, und die Rotation fehlgeschlagen ist. Weitere Informationen zur Rotation finden Sie unter [the section called “Fehlerbehebung bei der -Rotation”](#).

Überwachen Sie AWS Secrets Manager mit Amazon CloudWatch

Mit Amazon können Sie AWS Services überwachen und Alarme erstellen CloudWatch, um Sie darüber zu informieren, wenn sich Kennzahlen ändern. CloudWatch speichert diese Statistiken 15 Monate lang, sodass Sie auf historische Informationen zugreifen und sich einen besseren Überblick über die Leistung Ihrer Webanwendung oder Ihres Dienstes verschaffen können. Denn AWS Secrets Manager Sie können die Anzahl der Geheimnisse in Ihrem Konto überwachen, einschließlich der zum Löschen markierten Geheimnisse und API-Aufrufe an Secrets Manager, einschließlich der Aufrufe, die über die Konsole getätigt wurden. Informationen zur Überwachung von Metriken finden Sie unter [Verwenden von CloudWatch Metriken](#) im CloudWatch Benutzerhandbuch.

So finden Sie Secrets Manager Manager-Metriken

1. Wählen Sie auf der CloudWatch Konsole unter Metriken die Option Alle Metriken aus.

2. Geben Sie im Feld Metrik-Suche Folgendes einsecret.
3. Gehen Sie wie folgt vor:
 - Um die Anzahl der Geheimnisse in Ihrem Konto zu überwachen, wählen Sie AWS/SecretsManager und wählen Sie dann aus SecretCount. Diese Metrik wird stündlich veröffentlicht.
 - Um API-Aufrufe an Secrets Manager zu überwachen, einschließlich der Aufrufe, die über die Konsole getätigt wurden, wählen Sie „Verwendung“ > „Nach AWS Ressource“ und wählen Sie dann die API-Aufrufe aus, die überwacht werden sollen. Eine Liste der Secrets Manager Manager-APIs finden Sie unter [Secrets Manager Manager-Operationen](#).
4. Gehen Sie wie folgt vor:
 - Informationen zum Erstellen eines Diagramms der Metrik finden Sie unter [Metriken grafisch darstellen](#) im CloudWatch Amazon-Benutzerhandbuch.
 - Informationen zur Erkennung von Anomalien finden Sie unter [Verwenden der CloudWatch Anomalieerkennung](#) im CloudWatch Amazon-Benutzerhandbuch.
 - Informationen zum Abrufen von Statistiken für eine Metrik finden [Sie unter Statistiken für eine Metrik abrufen](#) im CloudWatch Amazon-Benutzerhandbuch.

CloudWatch Alarme

Sie können einen CloudWatch Alarm erstellen, der eine Amazon SNS SNS-Nachricht sendet, wenn sich der Wert einer Metrik ändert und der Alarm seinen Status ändert. Sie können einen Alarm für die Secrets Manager Manager-Metrik einrichten `ResourceCount`, die die Anzahl der Geheimnisse in Ihrem Konto angibt. Sie können auch Alarme für Ein Alarm überwacht eine Metrik über einen von Ihnen angegebenen Zeitraum und führt Aktionen auf der Grundlage des Werts der Metrik relativ zu einem bestimmten Schwellenwert über mehrere Zeiträume aus. Bei Alarmen werden nur Aktionen für anhaltende Statusänderungen ausgelöst. CloudWatch Alarme lösen keine Aktionen aus, nur weil sie sich in einem bestimmten Zustand befinden. Der Zustand muss sich geändert haben und für eine bestimmte Anzahl von Zeiträumen beibehalten worden sein.

Weitere Informationen finden Sie unter [Verwenden von CloudWatch Amazon-Alarmen](#) und [Erstellen eines CloudWatch Alarms auf der Grundlage von Anomalieerkennung](#) im CloudWatch Benutzerhandbuch.

Sie können auch Alarme einrichten, die auf bestimmte Grenzwerte achten und Benachrichtigungen senden oder Aktivitäten auslösen, wenn diese Grenzwerte erreicht werden. Weitere Informationen finden Sie im [CloudWatch Amazon-Benutzerhandbuch](#).

AWS Secrets Manager Ereignisse mit Amazon abgleichen EventBridge

In Amazon EventBridge können Sie Secrets Manager-Ereignisse anhand von CloudTrail Protokolleinträgen zuordnen. Sie können EventBridge Regeln konfigurieren, die nach diesen Ereignissen suchen, und dann neu generierte Ereignisse an ein Ziel senden, um Maßnahmen zu ergreifen. Eine Liste der CloudTrail Einträge, die Secrets Manager protokolliert, finden Sie unter [CloudTrail Einträge](#). Anweisungen zur Einrichtung EventBridge finden Sie unter [Erste Schritte EventBridge](#) im EventBridge Benutzerhandbuch.

Alle Änderungen mit einem bestimmten Secret abgleichen

Note

Da [einige Secrets Manager-Ereignisse](#) den ARN des Secrets mit einer anderen Großschreibung zurückgeben, müssen Sie in Ereignismustern, die mit mehr als einer Aktion übereinstimmen, zur Angabe eines Secrets nach ARN möglicherweise sowohl die Schlüssel `arn` als auch `aRN` angeben. Weitere Informationen finden Sie unter [AWS re:Post](#).

Das folgende Beispiel zeigt ein EventBridge Ereignismuster, das Protokolleinträgen für Änderungen an einem Geheimnis entspricht.

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

```
}
```

Ereignisse abgleichen, wenn ein Secret-Wert rotiert

Das folgende Beispiel zeigt ein EventBridge Ereignismuster, das CloudTrail Protokolleinträgen für Änderungen geheimer Werte entspricht, die durch manuelle Aktualisierungen oder automatische Rotation verursacht wurden. Da einige dieser Ereignisse aus Secrets-Manager-Vorgängen stammen und andere vom Secrets-Manager-Dienst generiert werden, müssen Sie den `detail-type` für beide angeben.

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

Überwachen Sie, wann auf AWS Secrets Manager Geheimnisse zugegriffen wird, die gelöscht werden sollen

Sie können eine Kombination aus Amazon CloudWatch Logs und Amazon Simple Notification Service (Amazon SNS) verwenden, um einen Alarm zu erstellen, der Sie über alle Versuche informiert, auf ein Geheimnis zuzugreifen, dessen Löschung noch aussteht. AWS CloudTrail Wenn Sie von einem solchen Alarm eine Benachrichtigung erhalten, können Sie den Löschvorgang für das Secret abbrechen, damit Sie mehr Zeit haben, um zu bestimmen, ob Sie es tatsächlich löschen möchten. Vielleicht führt Ihre Untersuchung zu einer Wiederherstellung des Secrets, da es tatsächlich noch benötigt wird. Alternativ müssen Sie den Benutzer möglicherweise mit Details des neuen zu verwendenden Secrets aktualisieren.

In den folgenden Verfahren wird erklärt, wie Sie eine Benachrichtigung erhalten, wenn eine Anfrage für den `GetSecretValue` Vorgang, der zu einer bestimmten Fehlermeldung führt, in Ihre CloudTrail Protokolldateien geschrieben wird. Andere API-Operationen können für das Geheimnis ausgeführt werden, ohne dass der Alarm ausgelöst wird. Dieser CloudWatch Alarm erkennt eine Nutzung,


die darauf hindeuten könnte, dass eine Person oder Anwendung veraltete Anmeldeinformationen verwendet.

Bevor Sie mit diesen Verfahren beginnen, müssen Sie das Konto AWS-Region und das Konto aktivieren, CloudTrail in dem Sie AWS Secrets Manager API-Anfragen überwachen möchten. Weitere Informationen finden Sie unter [Erstmaliges Erstellen eines Trails](#) im AWS CloudTrail - Benutzerhandbuch.

Schritt 1: Konfigurieren Sie die Übertragung der CloudTrail Protokolldatei in CloudWatch Logs

Sie müssen die Übermittlung Ihrer CloudTrail Protokolldateien an CloudWatch Logs konfigurieren. Sie tun dies, damit CloudWatch Logs sie auf Secrets Manager API-Anfragen zum Abrufen eines Geheimnisses hin überwachen kann, dessen Löschung noch aussteht.

Um die Übertragung von CloudTrail Protokolldateien an CloudWatch Logs zu konfigurieren

1. Öffnen Sie die CloudTrail Konsole unter <https://console.aws.amazon.com/cloudtrail/>.
2. Wählen Sie in der oberen Navigationsleiste die Option AWS-Region zum Überwachen von Geheimnissen aus.
3. Wählen Sie im linken Navigationsbereich Trails und dann den Namen des Trails aus, für den Sie die Konfiguration vornehmen möchten CloudWatch.
4. Scrollen Sie auf der Seite „Trails-Konfiguration“ nach unten zum Abschnitt „CloudWatch Logs“ und wählen Sie dann das Bearbeitungssymbol ).
5. Geben Sie in New or existing log group (Neue oder vorhandene Gruppe) den Namen der Protokollgruppe ein, z. B. **CloudTrail/MyCloudWatchLogGroup**.
6. Für die IAM-Rolle können Sie die Standardrolle mit dem Namen CloudTrail_ CloudWatchLogs _Role verwenden. Diese Rolle hat eine Standardrollenrichtlinie mit den erforderlichen Berechtigungen, um CloudTrail Ereignisse an die Protokollgruppe zu übermitteln.
7. Wählen Sie Continue (Weiter) aus, um die Konfigurationseinstellungen zu speichern.
8. Wählen AWS CloudTrail Sie auf der Gruppenseite CloudTrail Ereignisse im Zusammenhang mit API-Aktivitäten in Ihrem Konto an Ihre Protokollgruppe „ CloudWatch Logs“ übertragen die Option Zulassen aus.

Schritt 2: Erstellen Sie den CloudWatch Alarm

Um eine Benachrichtigung zu erhalten, wenn ein Secrets Manager `GetSecretValue` API-Vorgang den Zugriff auf ein Geheimnis anfordert, dessen Löschung noch aussteht, müssen Sie einen CloudWatch Alarm erstellen und eine Benachrichtigung konfigurieren.

Um einen CloudWatch Alarm zu erstellen

1. Melden Sie sich unter <https://console.aws.amazon.com/cloudwatch/> bei der CloudWatch Konsole an.
2. Wählen Sie in der oberen Navigationsleiste die AWS Region aus, in der Sie geheime Daten überwachen möchten.
3. Wählen Sie im linken Navigationsbereich Protokolle aus.
4. Aktivieren Sie in der Liste der Protokollgruppen das Kontrollkästchen neben der Protokollgruppe, die Sie im vorherigen Verfahren erstellt haben, z. B. `CloudTrail/MyCloudWatchLogGroup`. Wählen Sie anschließend `Metrikfilter erstellen`.
5. Geben Sie beim Filtermuster Folgendes an:

```
{ $.eventName = "GetSecretValue" && $.errorMessage = "*secret because it was marked for deletion*" }
```

Wählen Sie `Assign Metric`.

6. Gehen Sie auf der Seite `Metrikfilter erstellen` und eine Metrik zuweisen folgendermaßen vor:
 - a. Geben Sie für Namespace der Metrik den Wert **CloudTrailLogMetrics** ein.
 - b. Geben Sie für Metrikname den Wert **AttemptsToAccessDeletedSecrets** ein.
 - c. Wählen Sie `Erweiterte Metrikeinstellungen anzeigen` und geben Sie dann ggf. für Metrikwert **1** ein.
 - d. Wählen Sie `Filter erstellen`.
7. Wählen Sie im Filterfeld `Alarm erstellen`.
8. Führen Sie im Fenster `Alarm erstellen` die folgenden Schritte aus:
 - a. Geben Sie bei Name **AttemptsToAccessDeletedSecretsAlarm** ein.
 - b. `Whenever (Immer wenn)`: Wählen Sie für `is: (ist:) >=` aus und geben Sie **1** ein.
 - c. Gehen Sie neben `Benachrichtigung senden an`: folgendermaßen vor:

- Wählen Sie New List (Neue Liste) aus, um ein neues Amazon-SNS-Thema zu erstellen und zu verwenden und geben Sie dann einen neuen Namen für das Thema ein. Geben Sie unter E-Mail-Liste: mindestens eine E-Mail-Adresse ein. Sie können mehrere, durch Komma abgetrennte E-Mail-Adressen eingeben.
 - Wenn Sie ein vorhandenes Amazon-SNS-Thema verwenden möchten, wählen Sie den Namen des Themas. Wenn keine Liste vorhanden ist, wählen Sie Select list (Liste auswählen).
- d. Wählen Sie Alarm erstellen aus.

Schritt 3: Testen Sie den CloudWatch Alarm

Um Ihren Alarm zu testen, erstellen Sie ein Geheimnis und planen Sie es dann zum Löschen ein. Versuchen Sie dann, den Secret-Wert abzurufen. Sie erhalten in Kürze eine E-Mail unter der im Alarm konfigurierten Adresse. Darin werden Sie über die Verwendung eines Geheimnisses benachrichtigt, deren Löschung geplant ist.

Überwachen Sie AWS Secrets Manager Geheimnisse auf Einhaltung der Vorschriften, indem Sie AWS Config

Sie können sie verwenden AWS Config , um Ihre Geheimnisse daraufhin zu überprüfen, ob sie Ihren Standards entsprechen. Sie definieren Ihre internen Sicherheits- und Compliance-Anforderungen für Geheimnisse mithilfe von AWS Config Regeln. Dann AWS Config können Sie Geheimnisse identifizieren, die nicht Ihren Regeln entsprechen. Sie können auch Änderungen an geheimen Metadaten, der [Rotationskonfiguration](#), dem für die geheime Verschlüsselung verwendeten KMS-Schlüssel, der Lambda-Rotationsfunktion und den mit einem Geheimnis verknüpften Tags verfolgen.

Sie können so konfigurieren AWS Config , dass Sie über Änderungen informiert werden. Weitere Informationen finden Sie unter [Benachrichtigungen, die AWS Config an ein Amazon SNS SNS-Thema gesendet](#) werden.

Wenn Sie in mehreren Unternehmen AWS-Konten und AWS-Regionen in Ihrer Organisation über geheime Daten verfügen, können Sie diese Konfigurations- und Compliance-Daten zusammenfassen. Weitere Informationen finden Sie unter [Datenaggregation für mehrere Konten und mehrere Regionen](#).

Um zu beurteilen, ob geheime Daten den Vorschriften entsprechen

- Folgen Sie den Anweisungen zur [Bewertung Ihrer Ressourcen mithilfe von AWS Config Regeln](#) und wählen Sie eine der folgenden Regeln aus:
 - [secretsmanager-secret-unused](#) – Prüft, ob innerhalb der letzten angegebenen Anzahl von Tagen auf Secrets zugegriffen wurde.
 - [secretsmanager-using-cmk](#)— Prüft, ob Geheimnisse mit dem Von AWS verwalteter Schlüssel `aws/secretsmanager` oder einem vom Kunden verwalteten Schlüssel, in dem Sie sie erstellt haben, verschlüsselt sind AWS KMS.
 - [secretsmanager-rotation-enabled-check](#) – Prüft, ob die Rotation für in Secrets Manager gespeicherte Secrets konfiguriert ist.
 - [secretsmanager-scheduled-rotation-success-check](#) – Prüft, ob die letzte erfolgreiche Rotation innerhalb der konfigurierten Rotationsfrequenz liegt. Die Überprüfung muss mindestens täglich erfolgen.
 - [secretsmanager-secret-periodic-rotation](#) – Prüft, ob Secrets innerhalb der letzten angegebenen Anzahl von Tagen rotiert wurden.

Secrets Manager Manager-Kosten überwachen

Sie können Amazon verwenden CloudWatch , um die geschätzten AWS Secrets Manager Gebühren zu überwachen. Weitere Informationen finden Sie im CloudWatch Benutzerhandbuch unter [Erstellen eines Abrechnungsalarms zur Überwachung Ihrer geschätzten AWS Gebühren](#).

Eine weitere Option zur Überwachung Ihrer Kosten ist die Erkennung von AWS Kostenanomalien. Weitere Informationen finden Sie unter [Erkennung ungewöhnlicher Ausgaben mithilfe der Erkennung von AWS Kostenanomalien](#) im AWS Cost Management-Benutzerhandbuch.

Hinweise zur Überwachung Ihrer Secrets Manager Manager-Nutzung finden Sie unter [the section called “Überwachen Sie mit CloudWatch”](#) und [the section called “Loggen Sie sich mit AWS CloudTrail”](#).

Informationen zur AWS Secrets Manager Preisgestaltung finden Sie unter [the section called “Preisgestaltung”](#).

Konformitätsvalidierung für AWS Secrets Manager

Ihre Compliance-Verantwortung bei der Verwendung von Secrets Manager hängt von der Sensibilität Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. AWS stellt die folgenden Ressourcen zur Verfügung, die Sie bei der Einhaltung von Vorschriften unterstützen:

- [Schnellstartanleitungen für Sicherheit und Compliance](#) – In diesen Bereitstellungsleitfäden werden architektonische Überlegungen erörtert und Schritte für die Bereitstellung von sicherheits- und konformitätsorientierten Basisumgebungen auf AWS angegeben.
- Whitepaper „[Architecting for HIPAA Security and Compliance](#)“ — In diesem Whitepaper wird beschrieben, wie Unternehmen HIPAA-konforme Anwendungen erstellen können AWS .
- [AWS Compliance-Ressourcen](#) — Diese Sammlung von Arbeitsmappen und Leitfäden kann auf Ihre Branche und Ihren Standort zutreffen.
- AWS Config bewertet, zu welchem Grad die Konfiguration Ihrer Ressourcen den internen Vorgehensweisen, Branchenrichtlinien und Vorschriften entspricht. Weitere Informationen finden Sie unter [the section called “Überwachen Sie geheime Daten auf Einhaltung der Vorschriften”](#).
- [AWS Security Hub](#) bietet einen umfassenden Überblick über Ihren Sicherheitsstatus, sodass Sie überprüfen können AWS , ob Sie die Sicherheitsstandards und Best Practices der Branche einhalten. Informationen zur Verwendung von Security Hub zur Bewertung von Secrets Manager-Ressourcen finden Sie unter [AWS Secrets Manager -Steuerelemente](#) im AWS Security Hub - Benutzerhandbuch.
- IAM Access Analyzer analysiert Richtlinien, einschließlich Bedingungsanweisungen in einer Richtlinie, die einer externen Entität den Zugriff auf ein Secret erlauben. Weitere Informationen finden Sie unter [Vorschau des Zugriffs mit Access Analyzer](#).
- AWS Systems Manager stellt vordefinierte Runbooks für Secrets Manager bereit. Weitere Informationen finden Sie unter [Runbook-Referenz für Secrets Manager zur Systems-Manager-Automatisierung](#).
- Sie können Prüfberichte von Drittanbietern unter herunterladen AWS Artifact. Weitere Informationen finden Sie unter [Berichte herunterladen unter](#) .

Konformitätsstandards

AWS Secrets Manager wurde anhand der folgenden Standards geprüft und kann Teil Ihrer Lösung sein, wenn Sie eine Konformitätszertifizierung benötigen.

- [HIPAA — AWS hat sein Compliance-Programm zum Health Insurance Portability and Accountability Act \(HIPAA\) um eine HIPAA-fähige Dienstleistung erweitert. AWS Secrets Manager](#) Wenn Sie ein Business Associate Agreement (BAA) mit abgeschlossen haben AWS, können Sie Secrets Manager verwenden, um Ihre HIPAA-konformen Anwendungen zu erstellen. AWS bietet ein [Whitepaper mit Fokus auf HIPAA](#) für Kunden, die mehr darüber erfahren möchten, wie sie Gesundheitsinformationen AWS für die Verarbeitung und Speicherung von Gesundheitsinformationen nutzen können. Weitere Informationen finden Sie unter [HIPAA-Compliance](#).
- [An PIC teilnehmende Organisation — AWS Secrets Manager verfügt über eine Konformitätsbescheinigung für den Payment Card Industry \(PCI\) Data Security Standard \(DSS\) Version 3.2 auf Service Provider-Stufe 1.](#) Kunden, die AWS Produkte und Dienste zur Speicherung, Verarbeitung oder Übertragung von Karteninhaberdaten verwenden, können diese bei der Verwaltung ihrer eigenen PCI-DSS-Konformitätszertifizierung verwenden AWS Secrets Manager . Weitere Informationen zu PCI DSS, einschließlich der Möglichkeit, eine Kopie des AWS PCI Compliance Package anzufordern, finden Sie unter [PCI DSS Level 1](#).
- [ISO — AWS Secrets Manager hat die Konformitätszertifizierung für ISO/IEC 27001, ISO/IEC 27017, ISO/IEC 27018 und ISO 9001 erfolgreich abgeschlossen.](#) Weitere Informationen finden Sie unter [ISO 27001](#), [ISO 27017](#), [ISO 27018](#), [ISO 9001](#).
- [AICPA SOC — System and Organization Control \(SOC\) -Berichte sind unabhängige Prüfungsberichte von Drittanbietern, die belegen, wie Secrets Manager wichtige Compliance-Kontrollen und -Ziele erreicht.](#) Der Zweck dieser Berichte besteht darin, Ihnen und Ihren Prüfern zu helfen, die AWS Kontrollen zu verstehen, die zur Unterstützung von Betriebsabläufen und zur Einhaltung von Vorschriften eingeführt wurden. Weitere Informationen finden Sie unter [SOC-Compliance](#).
- [FedRAMP — Das Federal Risk and Authorization Management Program \(FedRAMP\) ist ein landesweites Programm, das einen standardisierten Ansatz für die Sicherheitsbewertung, Autorisierung und kontinuierliche Überwachung von Cloud-Produkten und -Diensten bietet.](#) Das FedRAMP-Programm sieht auch vorläufige Genehmigungen für Dienste und Regionen für Ost/West sowie für die Nutzung staatlicher oder regulierter GovCloud Daten vor. Weitere Informationen finden Sie unter [FedRAMP-Compliance](#).

- Verteidigungsministerium — Der Cloud Computing Security Requirements Guide (SRG) des Verteidigungsministeriums (DoD) bietet ein standardisiertes Bewertungs- und Autorisierungsverfahren für Cloud-Dienstleister (CSPs), um eine vorläufige DoD-Autorisierung zu erhalten, damit sie DoD-Kunden bedienen können. Weitere Informationen finden Sie in den [DoD SRG-Ressourcen](#).
- IRAP — Das Information Security Registered Assessors Program (IRAP) ermöglicht es australischen Regierungskunden, zu überprüfen, ob angemessene Kontrollen vorhanden sind, und das geeignete Verantwortungsmodell für die Erfüllung der Anforderungen des vom Australian Cyber Security Centre (ACSC) herausgegebenen Informationssicherheitshandbuch (ISM) der australischen Regierung zu bestimmen. Weitere Informationen finden Sie in den [IRAP-Ressourcen](#).
- OSPAR — Amazon Web Services (AWS) erhielt die Zertifizierung durch den Outsourced Service Provider's Audit Report (OSPAR). AWS Die Angleichung an die Richtlinien der Association of Banks in Singapore (ABS) über Kontrollziele und Verfahren für ausgelagerte Dienstleister (ABS-Richtlinien) zeigt den Kunden AWS, dass sie bereit sind, die hohen Erwartungen zu erfüllen, die die Finanzdienstleistungsbranche in Singapur an Cloud-Dienstleister stellt. Weitere Informationen finden Sie in den [OSPAR-Ressourcen](#).

Sicherheit in AWS Secrets Manager

Cloud-Sicherheit hat bei AWS höchste Priorität. Als AWS-Kunde profitieren Sie von einer Rechenzentrums- und Netzwerkarchitektur, die eingerichtet wurde, um die Anforderungen der anspruchsvollsten Organisationen in puncto Sicherheit zu erfüllen.

Sie und AWS teilen sich die Verantwortung für die Sicherheit. Im [Modell der übergreifenden Verantwortlichkeit](#) wird Folgendes mit „Sicherheit der Cloud“ bzw. „Sicherheit in der Cloud“ umschrieben:

- Sicherheit der Cloud selbst – AWS ist dafür verantwortlich, die Infrastruktur zu schützen, mit der AWS-Services in der AWS Cloud ausgeführt werden. AWS stellt Ihnen außerdem Services bereit, die Sie sicher nutzen können. Auditoren von Drittanbietern testen und überprüfen die Effektivität unserer Sicherheitsmaßnahmen im Rahmen der [AWS-Compliance-Programme](#) regelmäßig. Weitere Informationen zu den Compliance-Programmen für AWS Secrets Manager finden Sie unter [Durch das Compliance-Programm abgedeckte AWS-Services](#).
- Sicherheit in der Cloud — Ihr AWS-Service bestimmt Ihre Verantwortung. Sie sind auch für andere Faktoren verantwortlich, etwa für die Vertraulichkeit Ihrer Daten, für die Anforderungen Ihres Unternehmens und für die geltenden Gesetze und Vorschriften.

Weitere Ressourcen finden Sie unter [Sicherheitssäule – AWS Well-Architected Framework](#).

Themen

- [Reduzieren von Risiken durch die Verwendung der AWS CLI zur Speicherung Ihrer AWS Secrets Manager-Secrets](#)
- [Datenschutz in AWS Secrets Manager](#)
- [Geheime Verschlüsselung und Entschlüsselung in AWS Secrets Manager](#)
- [Sicherheit der Infrastruktur in AWS Secrets Manager](#)
- [Resilienz in AWS Secrets Manager](#)
- [Post-Quantum-TLS](#)

Reduzieren von Risiken durch die Verwendung der AWS CLI zur Speicherung Ihrer AWS Secrets Manager-Secrets

Wenn Sie die AWS Command Line Interface (AWS CLI) zum Aufrufen von AWS-Operationen verwenden, geben Sie diese Befehle in einer Befehls-Shell ein. Sie können beispielsweise die Windows-Eingabeaufforderung oder Windows PowerShell oder die Bash- oder Z-Shell verwenden. Viele dieser Befehlszeilen enthalten Funktionalität, die darauf ausgelegt ist, die Produktivität zu steigern. Diese Funktionalität kann jedoch dazu verwendet werden, Ihre Secrets zu kompromittieren. Zum Beispiel können Sie in den meisten Shells die Pfeiltaste nach oben verwenden, um den zuletzt eingegebenen Befehl zu sehen. Der Befehlsverlauf kann von jedem Benutzer ausgenutzt werden, der auf Ihre ungesicherte Sitzung zugreift. Andere Hintergrund-Dienstprogramme haben möglicherweise ebenfalls Zugriff auf Ihre Befehlsparameter mit der Absicht, dass Sie Ihre Aufgaben effizienter erledigen können. Um solche Risiken zu minimieren, stellen Sie sicher, dass Sie die folgenden Schritte durchführen:

- Sperren Sie Ihren Computer, wenn Sie die Konsole verlassen.
- Deinstallieren oder deaktivieren Sie Konsolenprogramme, die Sie nicht benötigen oder nicht mehr verwenden.
- Stellen Sie sicher, dass die Shell und das Remote-Zugriffsprogramm, falls Sie eines davon verwenden, keine eingetippten Befehle protokollieren.
- Verwenden Sie Techniken, um Parameter zu übergeben, die nicht vom Shell-Befehlsverlauf erfasst wurden. Das folgende Beispiel zeigt, wie Sie den Secret-Text in eine Textdatei eingeben können, die dann an den AWS Secrets Manager-Befehl übergeben und danach sofort zerstört wird. Dies bedeutet, dass der typische Shell-Verlauf den Secret-Text nicht erfasst.

Das folgende Beispiel zeigt typische Linux-Befehle (Ihre Shell benötigt jedoch möglicherweise etwas andere Befehle):

```
$ 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.
```

Nachdem Sie diese Befehle ausgeführt haben, sollten Sie in der Lage sein, mit den Aufwärts- und Abwärtspfeilen durch die Befehlshistorie zu scrollen und zu sehen, dass der Secret-Text in keiner Zeile angezeigt wird.

Important

Unter Windows gibt es standardmäßig keine gleichwertige Technik. Sie müssen erst die Größe des Befehlsverlaufspuffers auf 1 reduzieren.

So konfigurieren Sie die Windows-Eingabeaufforderung so, dass der Befehlsverlaufspuffer nur einen Befehl enthält

1. Öffnen Sie eine Administrator-Eingabeaufforderung (Run as administrator (Als Administrator ausführen)).
2. Wählen Sie das Symbol oben links und dann Eigenschaften aus.
3. Setzen Sie auf der Registerkarte Options (Optionen) die Werte für Buffer Size (Puffergröße) und Number of Buffers (Anzahl der Puffer) auf **1** und wählen Sie dann OK aus.
4. Wenn Sie einen Befehl eingeben, der nicht im Verlauf zu sehen sein soll, geben Sie direkt nach dem Befehl einen weiteren Befehl ein. Beispiel:

```
echo.
```

Dadurch wird sichergestellt, dass der sensible Befehl nicht mehr enthalten ist.

Für die Windows-Eingabeaufforderung können Sie das [SysInternals SDelete](#)-Tool herunterladen und dann folgende Befehle verwenden:

```
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.
```

Datenschutz in AWS Secrets Manager

Das [Modell der geteilten Verantwortung](#) von AWS gilt für den Datenschutz in AWS Secrets Manager. Wie in diesem Modell beschrieben, ist AWS verantwortlich für den Schutz der globalen Infrastruktur, in der die gesamte AWS Cloud ausgeführt wird. Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Dieser Inhalt enthält die Sicherheitskonfigurations- und Verwaltungsaufgaben für die von Ihnen verwendeten AWS-Services. Weitere Informationen zum Datenschutz finden Sie unter [Häufig gestellte Fragen zum Datenschutz](#). Informationen zum Datenschutz in Europa finden Sie im Blog-Beitrag [AWS-Modell der geteilten Verantwortung und die GDPR](#) im Blog zur AWS-Sicherheit.

Wir empfehlen aus Gründen des Datenschutzes, dass Sie AWS-Konto-Anmeldeinformationen schützen und die Benutzerkonten jeweils mit AWS Identity and Access Management (IAM) einrichten. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem sollten Sie die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto die [Multi-Faktor Authentifizierung \(MFA\)](#).
- Verwenden Sie SSL/TLS für die Kommunikation mit AWS-Ressourcen. Secrets Manager unterstützt TLS 1.2 und 1.3 in allen Regionen. Außerdem unterstützt Secrets Manager eine hybride [Post-Quantum-Schlüsselaustauschoption für das Netzwerkverschlüsselungsprotokoll TLS \(PQTLS\)](#).
- Signieren Sie programmgesteuerte Anfragen an Secrets Manager mit einer Zugriffsschlüssel-ID und einem geheimen Zugriffsschlüssel, die mit einem IAM-Prinzipal verknüpft sind. Alternativ können Sie mit [AWS Security Token Service](#) (AWS STS) temporäre Sicherheitsanmeldeinformationen erstellen, um die Anfragen zu signieren.

- Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit AWS CloudTrail ein. Siehe [the section called “Loggen Sie sich mit AWS CloudTrail”](#).
- Wenn Sie für den Zugriff auf AWS über eine Befehlszeilenschnittstelle oder über eine API FIPS 140-2-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Siehe [the section called “Secrets-Manager-Endpunkte”](#).
- Wenn Sie über die AWS CLI auf Secrets Manager zugreifen, [the section called “Reduzieren von Risiken durch die Verwendung der AWS CLI zur Speicherung Ihrer AWS Secrets Manager-Secrets”](#).

Verschlüsselung im Ruhezustand

Secrets Manager verwendet Verschlüsselung über AWS Key Management Service (AWS KMS), um die Vertraulichkeit von Daten im Ruhezustand zu schützen. AWS KMS bietet einen Service für die Schlüsselspeicherung und Verschlüsselung, der von vielen AWS-Services verwendet wird. Jedes Secret in Secrets Manager ist mit einem eindeutigen Datenschlüssel verschlüsselt. Jeder Datenschlüssel wird durch einen KMS-Schlüssel geschützt. Sie können die Standardverschlüsselung mit dem Secrets Manager Von AWS verwalteter Schlüssel für das Konto verwenden oder einen eigenen kundenverwalteten Schlüssel in AWS KMS erstellen. Durch die Verwendung eines vom Kunden verwalteten Schlüssels erhalten Sie detailliertere Autorisierungskontrollen für Ihre KMS-Schlüsselaktivitäten. Weitere Informationen finden Sie unter [the section called “Ver- und Entschlüsselung von Secrets”](#).

Verschlüsselung während der Übertragung

Secrets Manager bietet für die Verschlüsselung von Daten während der Übertragung sichere und private Endpunkte. Die sicheren und privaten Endpunkte ermöglichen es AWS, die Integrität von API-Anfragen an Secrets Manager zu schützen. AWS erfordert, dass API-Aufrufe vom Aufrufer mit X.509-Zertifikaten und/oder einem geheimen Secrets-Manager-Zugriffsschlüssel signiert sind. Diese Anforderung ist in der [Signaturversion 4 Signaturprozess](#) (Sigv4) festgelegt.

Wenn Sie das AWS Command Line Interface (AWS CLI) oder eine der AWS-SDKs verwenden, um Aufrufe von AWS durchzuführen, konfigurieren Sie den zu verwendenden Zugriffsschlüssel. Dann verwenden diese Tools automatisch den Zugriffsschlüssel, um die Anfragen für Sie zu signieren. Siehe [the section called “Reduzieren von Risiken durch die Verwendung der AWS CLI zur Speicherung Ihrer AWS Secrets Manager-Secrets”](#).

Datenschutz für den Datenverkehr zwischen Netzwerken

AWS bietet Optionen zur Wahrung der Privatsphäre beim Weiterleiten von Datenverkehr über bekannte und private Netzwerkrouuten.

Datenverkehr zwischen Service und On-Premises-Clients und -Anwendungen

Sie haben zwei Verbindungsoptionen zwischen Ihrem privaten Netzwerk und AWS Secrets Manager:

- Eine AWS-Site-to-Site-VPN-Verbindung. Weitere Informationen finden Sie unter [Was ist AWS Site-to-Site VPN?](#)
- Eine AWS-Direct-Connect-Verbindung. Weitere Informationen finden Sie unter [Was ist AWS Direct Connect?](#)

Datenverkehr zwischen AWS-Ressourcen in derselben Region

Wenn Sie den Datenverkehr zwischen Secrets Manager und API-Clients in AWS sichern wollen, richten Sie einen [AWS-PrivateLink](#) ein, um privat auf Secrets-Manager-API-Endpunkte zuzugreifen.

Verwaltung von Verschlüsselungsschlüsseln

Wenn Secrets Manager eine neue Version der geschützten Secret-Daten verschlüsseln muss, sendet Secrets Manager eine Anfrage an AWS KMS, um einen neuen Datenschlüssel aus dem KMS-Schlüssel zu generieren. Secrets Manager verwendet diesen Datenschlüssel für die [Envelope-Verschlüsselung](#). Secrets Manager speichert den verschlüsselten Datenschlüssel mit dem verschlüsselten Secret. Wenn das Geheimnis entschlüsselt werden muss, bittet Secrets Manager AWS KMS, den Datenschlüssel zu entschlüsseln. Anschließend verwendet Secrets Manager den entschlüsselten Datenschlüssel, um das verschlüsselte Secret zu entschlüsseln. Secrets Manager speichert den Datenschlüssel nie unverschlüsselt und entfernt ihn so schnell wie möglich aus dem Speicher. Weitere Informationen finden Sie unter [the section called “Ver- und Entschlüsselung von Secrets”](#).

Geheime Verschlüsselung und Entschlüsselung in AWS Secrets Manager

Secrets Manager verwendet eine [Umschlagverschlüsselung](#) mit AWS KMS [Schlüsseln](#) und [Datenschlüsseln](#), um jeden geheimen Wert zu schützen. Immer wenn sich der geheime Wert in

einem Geheimnis ändert, fordert Secrets Manager einen neuen Datenschlüssel von AWS KMS an, um ihn zu schützen. Der Datenschlüssel wird unter einem KMS-Schlüssel verschlüsselt und in den Metadaten des Secrets gespeichert. Um das Geheimnis zu entschlüsseln, entschlüsselt Secrets Manager zuerst den verschlüsselten Datenschlüssel mit dem KMS-Schlüssel in AWS KMS

Secrets Manager verwendet den KMS-Schlüssel nicht zum direkten Verschlüsseln des Secret-Werts. Stattdessen verwendet der Service den KMS-Schlüssel zum Generieren und Verschlüsseln eines symmetrischen 256-Bit Advanced Encryption Standard (AES)-[Datenschlüssels](#) und den Datenschlüssel zum Verschlüsseln des Secret-Werts. Secrets Manager verwendet den Klartext-Datenschlüssel, um den geheimen Wert außerhalb von zu verschlüsseln AWS KMS, und entfernt ihn dann aus dem Speicher. Die verschlüsselte Kopie des Datenschlüssels wird in den Metadaten des Geheimnisses gespeichert.

Themen

- [Auswahl eines Schlüssels AWS KMS](#)
- [Was ist verschlüsselt?](#)
- [Ver- und Entschlüsselungsprozesse](#)
- [Berechtigungen für den KMS-Schlüssel](#)
- [Wie Secrets Manager den KMS-Schlüssel verwendet](#)
- [Schlüsselrichtlinie des Von AWS verwalteter Schlüssel \(aws/secretsmanager\)](#)
- [Verschlüsselungskontext für Secrets Manager](#)
- [Überwachen Sie die Secrets Manager Manager-Interaktion mit AWS KMS](#)

Auswahl eines Schlüssels AWS KMS

Wenn Sie ein Geheimnis erstellen, können Sie einen beliebigen vom Kunden verwalteten symmetrischen Verschlüsselungsschlüssel in der Region AWS-Konto und wählen oder den Von AWS verwalteter Schlüssel for Secrets Manager (aws/secretsmanager) verwenden. Wenn Sie das auswählen Von AWS verwalteter Schlüssel aws/secretsmanager und es noch nicht existiert, erstellt Secrets Manager es und ordnet es dem Geheimnis zu. Sie können entweder denselben KMS-Schlüssel oder unterschiedliche KMS-Schlüssel für jedes Secret in Ihrem Konto verwenden. Möglicherweise möchten Sie verschiedene KMS-Schlüssel verwenden, um benutzerdefinierte Berechtigungen für die Schlüssel für eine Gruppe von Secrets festzulegen oder wenn Sie bestimmte Vorgänge für diese Schlüssel überprüfen möchten. Secrets Manager unterstützt nur [symmetrische KMS-Verschlüsselungsschlüssel](#). Wenn Sie einen KMS-Schlüssel in einem

[externen Schlüsselspeicher](#) verwenden, können kryptografische Operationen mit dem KMS-Schlüssel länger dauern und weniger zuverlässig und dauerhaft sein, da die Anforderung außerhalb von AWS gesendet werden muss.

Informationen zum Ändern des Verschlüsselungsschlüssels für ein Secret finden Sie unter [the section called “Ändern des Verschlüsselungsschlüssels für ein Secret”](#).

Wenn Sie den Verschlüsselungsschlüssel ändern, verschlüsselt Secrets Manager `AWSCURRENTAWSPENDING`, und `AWSPREVIOUS` Versionen erneut mit dem neuen Schlüssel. Um zu verhindern, dass Sie aus dem Geheimnis ausgesperrt werden, speichert Secrets Manager alle vorhandenen Versionen mit dem vorherigen Schlüssel verschlüsselt. Das bedeutet `AWSCURRENTAWSPENDING`, dass Sie `AWSPREVIOUS` Versionen mit dem vorherigen Schlüssel oder dem neuen Schlüssel entschlüsseln können.

Damit es nur mit dem neuen Verschlüsselungsschlüssel entschlüsselt werden `AWSCURRENT` kann, erstellen Sie eine neue Version des Geheimnisses mit dem neuen Schlüssel. Um dann die `AWSCURRENT` geheime Version entschlüsseln zu können, benötigen Sie die Erlaubnis für den neuen Schlüssel.

Sie können den Zugriff auf den verweigern Von AWS verwalteter Schlüssel `aws/secretsmanager` und verlangen, dass Geheimnisse mit einem vom Kunden verwalteten Schlüssel verschlüsselt werden. Weitere Informationen finden Sie unter [the section called “Beispiel: Verweigern Sie einen bestimmten AWS KMS Schlüssel zur Verschlüsselung von Geheimnissen”](#).

Um den KMS-Schlüssel zu finden, der einem Geheimnis zugeordnet ist, schauen Sie sich das Geheimnis in der Konsole an oder rufen Sie [ListSecrets](#) oder an [DescribeSecret](#). Wenn das Geheimnis mit dem Von AWS verwalteter Schlüssel for Secrets Manager (`aws/secretsmanager`) verknüpft ist, geben diese Operationen keinen KMS-Schlüsselbezeichner zurück.

Was ist verschlüsselt?

Secrets Manager verschlüsselt den geheimen Wert, aber Folgendes wird nicht verschlüsselt:

- Name und Beschreibung des Secrets
- Rotationseinstellungen
- ARN des mit dem Secret verknüpften KMS-Schlüssels
- Alle angehängten AWS Tags

Ver- und Entschlüsselungsprozesse

Um den Secret-Wert in einem Secret zu verschlüsseln, geht Secrets Manager wie folgt vor.

1. Secrets Manager ruft den AWS KMS [GenerateDataKey](#) Vorgang mit der ID des KMS-Schlüssels für das Geheimnis und einer Anforderung für einen symmetrischen 256-Bit-AES-Schlüssel auf. AWS KMS gibt einen Klartext-Datenschlüssel und eine Kopie dieses Datenschlüssels zurück, der unter dem KMS-Schlüssel verschlüsselt wurde.
2. Secrets Manager verwendet den Klartext-Datenschlüssel und den Advanced Encryption Standard (AES) -Algorithmus, um den geheimen Wert außerhalb von zu verschlüsseln. AWS KMS Der Klartextschlüssel wird direkt nach der Verwendung aus dem Speicher gelöscht.
3. Secrets Manager speichert den verschlüsselten Datenschlüssel in den Metadaten des Secrets, sodass er jederzeit zum Entschlüsseln des Secret-Werts verfügbar ist. Keine der Secrets Manager APIs gibt jedoch das verschlüsselte Secret oder den verschlüsselten Datenschlüssel zurück.

So wird ein verschlüsselter Geheimniswert entschlüsselt:

1. Secrets Manager ruft den AWS KMS [Decrypt-Vorgang](#) auf und übergibt den verschlüsselten Datenschlüssel.
2. AWS KMS verwendet den KMS-Schlüssel für das Geheimnis, um den Datenschlüssel zu entschlüsseln. gibt den Datenschlüssel in Klartext zurück.
3. Secrets Manager entschlüsselt den Secret-Wert mithilfe des Nur-Text-Datenschlüssels. Anschließend entfernt er den Datenschlüssel so schnell wie möglich aus dem Speicher.

Berechtigungen für den KMS-Schlüssel

Wenn Secrets Manager einen KMS-Schlüssel in kryptografischen Operationen verwendet, handelt er im Namen des Benutzers, der auf den Secret-Wert zugreift oder diesen aktualisiert. Sie können Berechtigungen in einer IAM-Richtlinie oder einer Schlüsselrichtlinie gewähren. Für die folgenden Secrets Manager Manager-Operationen sind AWS KMS Berechtigungen erforderlich.

- [CreateSecret](#)
- [GetSecretValue](#)
- [PutSecretValue](#)
- [UpdateSecret](#)

- [ReplicateSecretToRegions](#)

Damit der KMS-Schlüssel nur für Anfragen verwendet werden kann, die ihren Ursprung in Secrets Manager haben, können Sie in der Berechtigungsrichtlinie den [ViaService Bedingungsschlüssel kms:](#) mit dem `secretsmanager.<Region>.amazonaws.com` Wert verwenden.

Sie können auch die Schlüssel oder Werte im [Verschlüsselungskontext](#) als Bedingung für die Verwendung des KMS-Schlüssels für kryptografische Operationen verwenden.

Verwenden Sie beispielsweise einen [Zeichenfolgen-Bedingungsoperator](#) in einem IAM- oder Schlüsselrichtliniendokument oder eine [Erteilungseinschränkung](#) in einer Erteilung. Die Propagierung von Berechtigungen für KMS-Schlüssel kann bis zu fünf Minuten dauern. Weitere Informationen finden Sie unter [CreateGrant](#).

Wie Secrets Manager den KMS-Schlüssel verwendet

Secrets Manager ruft die folgenden AWS KMS Operationen mit Ihrem KMS-Schlüssel auf.

GenerateDataKey

Secrets Manager ruft die AWS KMS [GenerateDataKey](#) Operation als Antwort auf die folgenden Secrets Manager Manager-Operationen auf.

- [CreateSecret](#)— Wenn das neue Geheimnis einen geheimen Wert enthält, fordert Secrets Manager einen neuen Datenschlüssel an, um es zu verschlüsseln.
- [PutSecretValue](#)— Secrets Manager fordert einen neuen Datenschlüssel an, um den angegebenen geheimen Wert zu verschlüsseln.
- [ReplicateSecretToRegions](#)— Um das replizierte Geheimnis zu verschlüsseln, fordert Secrets Manager einen Datenschlüssel für den KMS-Schlüssel in der Replikatregion an.
- [UpdateSecret](#)— Wenn Sie den geheimen Wert oder den KMS-Schlüssel ändern, fordert Secrets Manager einen neuen Datenschlüssel an, um den neuen geheimen Wert zu verschlüsseln.

Die [RotateSecret](#) Operation wird nicht aufgerufen `GenerateDataKey`, da sie den geheimen Wert nicht ändert. Wenn `RotateSecret` jedoch eine Lambda-Rotationsfunktion aufruft, die den Secret-Wert ändert, löst sein Aufruf der Operation `PutSecretValue` eine `GenerateDataKey`-Anforderung aus.

Decrypt

Secrets Manager ruft die Operation [Decrypt](#) als Reaktion auf die folgenden Secrets-Manager-Operationen auf.

- [GetSecretValue](#) und [BatchGetSecretValue](#)— Secrets Manager entschlüsselt den geheimen Wert, bevor er an den Aufrufer zurückgegeben wird. Um einen verschlüsselten geheimen Wert zu entschlüsseln, ruft Secrets Manager die Operation AWS KMS [Decrypt](#) auf, um den verschlüsselten Datenschlüssel im Secret zu entschlüsseln. Dann wird der verschlüsselte Geheimniswert mithilfe des Klartext-Datenschlüssels entschlüsselt. Bei Batch-Befehlen kann Secrets Manager den entschlüsselten Schlüssel wiederverwenden, sodass nicht alle Aufrufe zu einer Decrypt-Anforderung führen.
- [PutSecretValue](#) und [UpdateSecret](#)— Die meisten PutSecretValue UpdateSecret AND-Anfragen lösen keine Operation aus. Decrypt Wenn jedoch eine PutSecretValue- oder UpdateSecret-Anforderung versucht, den Secret-Wert in einer vorhandenen Version eines Secrets zu ändern, entschlüsselt Secrets Manager den vorhandenen Secret-Wert und vergleicht ihn mit dem Secret-Wert in der Anforderung, um zu bestätigen, dass beide Werte identisch sind. Somit wird die Idempotenz von Secrets-Manager-Operationen sichergestellt. Um einen verschlüsselten geheimen Wert zu entschlüsseln, ruft Secrets Manager die Operation AWS KMS [Decrypt](#) auf, um den verschlüsselten Datenschlüssel im Secret zu entschlüsseln. Dann wird der verschlüsselte Geheimniswert mithilfe des Klartext-Datenschlüssels entschlüsselt.
- [ReplicateSecretToRegions](#)— Secrets Manager entschlüsselt zuerst den geheimen Wert in der primären Region, bevor er den geheimen Wert mit dem KMS-Schlüssel in der Replikatregion erneut verschlüsselt.

Encrypt

Secrets Manager ruft die [Verschlüsselungsoperation](#) als Reaktion auf die folgenden Secrets-Manager-Operationen auf:

- [UpdateSecret](#)— Wenn Sie den KMS-Schlüssel ändern, verschlüsselt Secrets Manager den Datenschlüssel, der die AWSPENDING geheimen Versionen AWSCURRENT, und schützt AWSPREVIOUS, erneut mit dem neuen Schlüssel.
- [ReplicateSecretToRegions](#)— Secrets Manager verschlüsselt den Datenschlüssel während der Replikation erneut mithilfe des KMS-Schlüssels in der Replikatregion.

DescribeKey

Secrets Manager ruft den [DescribeKey](#) Vorgang auf, um zu bestimmen, ob der KMS-Schlüssel aufgeführt werden soll, wenn Sie ein Geheimnis in der Secrets Manager-Konsole erstellen oder bearbeiten.

Validieren des Zugriffs auf den KMS-Schlüssel

Wenn Sie den einem Secret zugeordneten KMS-Schlüssel einrichten oder ändern, ruft Secrets Manager die Operationen `GenerateDataKey` und `Decrypt` mit dem angegebenen KMS-Schlüssel auf. Damit wird bestätigt, dass der Aufrufer über die Berechtigung zur Verwendung des KMS-Schlüssels für diese Operation verfügt. Secrets Manager verwirft die Ergebnisse dieser Operationen. Sie werden nicht für Entschlüsselungsoperationen verwendet.

Sie können diese Validierungsaufrufe daran erkennen, dass der `SecretVersionIdWert` [des - Verschlüsselungskontexts](#) in diesen Anforderungen `RequestToValidateKeyAccess` ist.

Note

Bisher enthielten Secrets-Manager-Validierungsaufrufe keinen Verschlüsselungskontext. Möglicherweise finden Sie in älteren AWS CloudTrail Protokollen Aufrufe ohne Verschlüsselungskontext.

Schlüsselrichtlinie des Von AWS verwalteter Schlüssel (**aws/secretsmanager**)

Die Schlüsselrichtlinie für den Von AWS verwalteter Schlüssel for Secrets Manager (`aws/secretsmanager`) gibt Benutzern nur dann die Erlaubnis, den KMS-Schlüssel für bestimmte Operationen zu verwenden, wenn Secrets Manager die Anfrage im Namen des Benutzers stellt. Die Schlüsselrichtlinie erlaubt es keinem Benutzer, den KMS-Schlüssel direkt zu verwenden.

Diese Schlüsselrichtlinie wird – wie die Richtlinien aller [Von AWS verwaltete Schlüssel](#) – vom Service eingerichtet. Sie können die Schlüsselrichtlinie nicht ändern, Sie können sie jedoch jederzeit anzeigen. Details dazu finden Sie unter [Anzeigen einer Schlüsselrichtlinie](#).

Die Richtlinienanweisungen in der Schlüsselrichtlinie haben folgende Wirkungen:

- Benutzer im Konto dürfen den KMS-Schlüssel für kryptografische Operationen nur verwenden, wenn die Anforderung von Secrets Manager in ihrem Namen ergeht. Der Bedingungsschlüssel `kms:ViaService` setzt diese Beschränkung durch.
- Ermöglicht dem AWS Konto, IAM-Richtlinien zu erstellen, die es Benutzern ermöglichen, die Eigenschaften von KMS-Schlüsseln einzusehen und Zuweisungen zu widerrufen.

- Secrets Manager verwendet zwar keine Erteilungen für den Zugriff auf KMS-Schlüssel, die Richtlinie ermöglicht es Secrets Manager jedoch, im Namen des Benutzers für den KMS-Schlüssel [Ertellungen zu erstellen](#), und erlaubt es dem Konto, [Ertellungen zu widerrufen](#), mit denen Secrets Manager den KMS-Schlüssel verwenden kann. Dies sind Standardkomponenten des Richtliniendokuments für einen Von AWS verwalteter Schlüssel.

Im Folgenden finden Sie eine wichtige Richtlinie für ein Beispiel Von AWS verwalteter Schlüssel für Secrets Manager.

```
{
  "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": "*"
}
]
}

```

Verschlüsselungskontext für Secrets Manager

Ein [Verschlüsselungskontext](#) ist eine Gruppe von Schlüssel/Wert-Paaren mit willkürlichen, nicht geheimen Daten. Wenn Sie einen Verschlüsselungskontext in eine Anforderung zur Verschlüsselung von Daten einbeziehen, wird der Verschlüsselungskontext AWS KMS kryptografisch an die verschlüsselten Daten gebunden. Zur Entschlüsselung der Daten müssen Sie denselben Verschlüsselungskontext übergeben.

Secrets Manager verwendet in seinen Anfragen [GenerateDataKey](#) und [Decrypt](#) an einen Verschlüsselungskontext mit zwei Name-Wert-Paaren, die das Geheimnis und seine Version identifizieren, wie im folgenden Beispiel gezeigt. AWS KMS Die Namen variieren nicht. In Kombination unterscheiden sich die Verschlüsselungskontextwerte jedoch für jeden Geheimniswert.

```
"encryptionContext": {
  "SecretARN": "arn:aws:secretsmanager:us-east-2:111122223333:secret:test-secret-
a1b2c3",
  "SecretVersionId": "EXAMPLE1-90ab-cdef-fedc-ba987SECRET1"
}
```

Sie können den Verschlüsselungskontext verwenden, um diese kryptografischen Vorgänge in Prüfaufzeichnungen und Protokollen wie Amazon CloudWatch Logs zu identifizieren [AWS CloudTrail](#) und als Voraussetzung für die Autorisierung in Richtlinien und Zuschüssen zu verwenden.

Der Secrets-Manager-Verschlüsselungskontext besteht aus zwei Name-Wert-Paaren.

- **SecretARN** – Das erste Name-Wert-Paar dient der Identifizierung des Secrets. Der Schlüssel lautet `SecretARN`. Der Wert ist der Amazon-Ressourcenname (ARN) des Geheimnisses.

```
"SecretARN": "ARN of an Secrets Manager secret"
```

Wenn der ARN des Geheimnisses beispielsweise `arn:aws:secretsmanager:us-east-2:111122223333:secret:test-secret-a1b2c3` lautet, würde der Verschlüsselungskontext das folgende Paar beinhalten.

```
"SecretARN": "arn:aws:secretsmanager:us-east-2:111122223333:secret:test-secret-
a1b2c3"
```

- **SecretVersionId**— Das zweite Name-Wert-Paar identifiziert die Version des Geheimnisses. Der Schlüssel lautet `SecretVersionId`. Der Wert ist die Versions-ID.

```
"SecretVersionId": "<version-id>"
```

Wenn die Versions-ID des Geheimnisses beispielsweise `EXAMPLE1-90ab-cdef-fedc-ba987SECRET1` lautet, würde der Verschlüsselungskontext das folgende Paar beinhalten.

```
"SecretVersionId": "EXAMPLE1-90ab-cdef-fedc-ba987SECRET1"
```

Wenn Sie den KMS-Schlüssel für ein Geheimnis einrichten oder ändern, sendet Secrets Manager Anfragen [GenerateDataKey](#) und [entschlüsselt](#), um AWS KMS zu überprüfen, ob der Anrufer berechtigt ist, den KMS-Schlüssel für diese Operationen zu verwenden. Die Antworten werden verworfen und nicht für den Geheimniswert verwendet.

In diesem Validierungsanfragen ist der Wert von `SecretARN` der tatsächliche ARN des Geheimnisses. Der Wert `SecretVersionId` ist jedoch `RequestToValidateKeyAccess`, wie im folgenden Beispielverschlüsselungskontext dargestellt. Mithilfe dieses speziellen Werts lassen sich Validierungsanfragen in Protokollen und Audit-Trails identifizieren.

```
"encryptionContext": {
  "SecretARN": "arn:aws:secretsmanager:us-east-2:111122223333:secret:test-secret-
a1b2c3",
  "SecretVersionId": "RequestToValidateKeyAccess"
}
```

Note

Bisher enthielten Secrets-Manager-Validierungsanforderungen keinen Verschlüsselungskontext. Möglicherweise finden Sie in älteren AWS CloudTrail Protokollen Anrufe ohne Verschlüsselungskontext.

Überwachen Sie die Secrets Manager Manager-Interaktion mit AWS KMS

Sie können Amazon AWS CloudTrail CloudWatch Logs verwenden, um die Anfragen zu verfolgen, an die Secrets Manager in AWS KMS Ihrem Namen sendet. Weitere Informationen über das Überwachen der Verwendung von Secrets finden Sie unter [Überwachung von Geheimnissen](#).

GenerateDataKey

Wenn Sie den geheimen Wert in einem Geheimnis erstellen oder ändern, sendet Secrets Manager eine [GenerateDataKey](#)Anfrage an AWS KMS , in der der KMS-Schlüssel für das Geheimnis angegeben wird.

Das Ereignis, das die `GenerateDataKey`-Operation aufzeichnet, ähnelt dem folgenden Beispielergebnis. Die Anforderung wird durch `secretsmanager.amazonaws.com` aufgerufen. Die Parameter enthalten den Amazon-Ressourcennamen (ARN) des KMS-Schlüssels für

das Secret, einen Schlüsselbezeichner, der einen 256-Bit-Schlüssel benötigt, und den [Verschlüsselungskontext](#), der das Secret und dessen Version identifiziert.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AROAIGDTESTANDEXAMPLE:user01",
    "arn": "arn:aws:sts::111122223333:assumed-role/Admin/user01",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2018-05-31T23:23:41Z"
      }
    }
  },
  "invokedBy": "secretsmanager.amazonaws.com",
},
"eventTime": "2018-05-31T23:23:41Z",
"eventSource": "kms.amazonaws.com",
"eventName": "GenerateDataKey",
"awsRegion": "us-east-2",
"sourceIPAddress": "secretsmanager.amazonaws.com",
"userAgent": "secretsmanager.amazonaws.com",
"requestParameters": {
  "keyId": "arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
  "keySpec": "AES_256",
  "encryptionContext": {
    "SecretARN": "arn:aws:secretsmanager:us-east-2:111122223333:secret:test-secret-a1b2c3",
    "SecretVersionId": "EXAMPLE1-90ab-cdef-fedc-ba987SECRET1"
  }
},
"responseElements": null,
"requestID": "a7d4dd6f-6529-11e8-9881-67744a270888",
"eventID": "af7476b6-62d7-42c2-bc02-5ce86c21ed36",
"readOnly": true,
"resources": [
  {
    "ARN": "arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
```



```

        "accountId": "111122223333",
        "type": "AWS::KMS::Key"
    }
],
"eventType": "AwsApiCall",
"recipientAccountId": "111122223333"
}

```

Decrypt

Wenn Sie den geheimen Wert eines Geheimnisses abrufen oder ändern, sendet Secrets Manager eine [Entschlüsselungsanforderung](#) an, AWS KMS um den verschlüsselten Datenschlüssel zu entschlüsseln. Bei Batch-Befehlen kann Secrets Manager den entschlüsselten Schlüssel wiederverwenden, sodass nicht alle Aufrufe zu einer Decrypt-Anforderung führen.

Das Ereignis, das die Decrypt-Operation aufzeichnet, ähnelt dem folgenden Beispielergebnis. Der Benutzer ist der Hauptbenutzer in Ihrem AWS Konto, der auf die Tabelle zugreift. Zu den Parametern gehören der verschlüsselte Tabellenschlüssel (als Chiffretext-Blob) und der [Verschlüsselungskontext](#), der die Tabelle und das Konto identifiziert. AWS KMS leitet die ID des KMS-Schlüssels aus dem Chiffretext ab.

```

{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AROAIIGDTESTANDEXAMPLE: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"
  }

```

Encrypt

Wenn Sie den KMS-Schlüssel ändern, der einem Geheimnis zugeordnet ist, sendet Secrets Manager eine [Verschlüsselungsanforderung](#) an, AWS KMS um die AWSPENDING geheimen Versionen AWSCURRENTAWSPREVIOUS, und mit dem neuen Schlüssel erneut zu verschlüsseln. Wenn Sie ein Secret in eine andere Region replizieren, sendet Secrets Manager außerdem eine [Verschlüsselungsanfrage](#) an AWS KMS.

Das Ereignis, das die Encrypt-Operation aufzeichnet, ähnelt dem folgenden Beispielergebnis. Der Benutzer ist der Hauptbenutzer in Ihrem AWS Konto, der auf die Tabelle zugreift.

```

{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AROAIQDTESTANDEXAMPLE: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"
}
```

Sicherheit der Infrastruktur in AWS Secrets Manager

Als verwalteter Service ist AWS Secrets Manager durch die globalen Verfahren zur Gewährleistung der Netzwerksicherheit von AWS geschützt. Informationen zu AWS-Sicherheitsdiensten und wie AWS die Infrastruktur schützt, finden Sie unter [AWS Cloud-Sicherheit](#). Informationen zum Entwerfen Ihrer AWS-Umgebung anhand der bewährten Methoden für die Infrastruktursicherheit finden Sie unter [Infrastrukturschutz](#) im Security Pillar AWS Well-Architected Framework.

Der Zugriff über das Netzwerk auf [AWS erfolgt durch veröffentlichte](#) -APIs. Diese APIs lassen sich von einem beliebigen Netzwerkstandort aus aufrufen. Secrets Manager unterstützt allerdings [ressourcenbasierte Zugriffsrichtlinien](#), die Einschränkungen auf der Grundlage der Quell-IP-Adresse enthalten können. Sie können auch ressourcenbasierte Richtlinien verwenden, um den Zugriff über [bestimmte Virtual Private Cloud \(VPC\)-Endpunkte](#) oder bestimmte VPCs zu steuern. Auf diese Weise wird der Netzwerkzugriff auf ein bestimmtes Geheimnis nur von der spezifischen VPC innerhalb des AWS Netzwerks isoliert. Weitere Informationen finden Sie unter [VPC-Endpunkt](#).

Resilienz in AWS Secrets Manager

AWS baut die globale Infrastruktur rund um AWS-Regionen Availability Zones auf. AWS-Regionen stellt mehrere physisch getrennte und isolierte Availability Zones bereit, die über Netzwerke mit niedriger Latenz, hohem Durchsatz und hoher Redundanz miteinander verbunden sind. Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Zonen ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones ermöglichen Ihnen eine höhere Verfügbarkeit, Fehlertoleranz und Skalierbarkeit als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Weitere Informationen zu Resilienz und Disaster Recovery finden Sie unter [Reliability Pillar — AWS Well-Architected Framework](#).

[Weitere Informationen zu AWS-Regionen und Availability Zones finden Sie unter AWS Globale Infrastruktur.](#)

Post-Quantum-TLS

Secrets Manager unterstützt eine hybride Post-Quantum-Schlüsselaustauschoption für das Transport Layer Security (TLS) Netzwerkverschlüsselungsprotokoll. Sie können diese TLS-Option verwenden, wenn Sie eine Verbindung zu Secrets-Manager-API-Endpunkten herstellen. Wir bieten dieses Feature an, bevor Post-Quantum-Algorithmen standardisiert werden, damit Sie damit beginnen

können, die Auswirkungen dieser Schlüsselaustauschprotokolle auf Secrets-Manager-Aufrufe zu testen. Diese optionalen Hybrid-Post-Quantum-Schlüsselaustauschfunktionen sind mindestens so sicher wie die heute verwendete TLS-Verschlüsselung und bieten wahrscheinlich zusätzliche Sicherheitsvorteile. Sie wirken sich jedoch auf Latenz und Durchsatz im Vergleich zu den klassischen Schlüsselaustauschprotokollen aus, die heute verwendet werden.

Um Daten zu schützen, die heute vor potenziellen zukünftigen Angriffen verschlüsselt werden, beteiligt AWS sich mit der kryptografischen Gemeinschaft an der Entwicklung von quantenresistenten oder Post-Quantum Algorithmen. Wir haben hybride Post-Quantum-Schlüsselaustausch-Cipher-Suites in Secrets-Manager-Endpunkten implementiert. Diese Hybrid-Cipher-Suites, die klassische und Post-Quantum-Elemente kombinieren, stellen sicher, dass Ihre TLS-Verbindung mindestens so stark ist wie bei klassischen Cipher-Suites. Da sich jedoch die Leistungseigenschaften und Bandbreitenanforderungen hybrider Cipher-Suites von denen klassischer Schlüsselaustauschmechanismen unterscheiden, empfehlen wir Ihnen, diese bei Ihren API-Aufrufen zu testen.

Secrets Manager unterstützt PQTLS in allen Regionen bis auf Regionen in China.

Konfigurieren von Hybrid-Post-Quantum-TLS

1. Fügen Sie den AWS-Common-Runtime-Client zu Ihren Maven-Abhängigkeiten hinzu. Wir empfehlen, die neueste verfügbare Version zu verwenden. Diese Anweisung fügt beispielsweise die Version 2.20.0 hinzu.

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>aws-crt-client</artifactId>
  <version>2.20.0</version>
</dependency>
```

2. Fügen Sie das AWS-SDK für Java 2.x zu Ihrem Projekt hinzu und initialisieren Sie es. Aktivieren Sie die hybriden Post-Quantum-Cipher-Suites auf Ihrem HTTP-Client.

```
SdkAsyncHttpClient awsCrtHttpClient = AwsCrtAsyncHttpClient.builder()
    .postQuantumTlsEnabled(true)
    .build();
```

3. Erstellen Sie den [asynchronen Secrets-Manager-Client](#).

```
SecretsManagerAsyncClient secretsManagerAsync = SecretsManagerAsyncClient.builder()
```

```
.httpClient(awsCrtHttpClient)
    .build();
```

Wenn Sie jetzt Secrets-Manager-API-Vorgänge aufrufen, werden Ihre Aufrufe über hybrides Post-Quantum-TLS an den Secrets-Manager-Endpunkt übertragen.

Weitere Informationen zur Verwendung von hybridem Post-Quantum-TLS finden Sie unter:

- [AWS SDK for Java 2.x-Entwicklerhandbuch](#) und der [AWS SDK for Java 2.x-veröffentlichte](#) Blog-Beitrag.
- [Einführung von s2n-tls, einer neuen Open-Source-TLS-Implementierung](#) und [Verwendung von s2n-tls](#).
- [Post-Quantum Cryptography](#) am National Institute for Standards and Technology (NIST).
- [Hybride Post-Quantum Key Encapsulation Methods \(PQ KEM\) für Transport Layer Security 1.2 \(TLS\)](#).

Post-Quantum-TLS für Secrets Manager ist in allen AWS-Regionen bis auf Regionen in China verfügbar.

Problembhebung AWS Secrets Manager

Verwenden Sie die hier aufgeführten Informationen, um Probleme zu diagnostizieren und zu beheben, die beim Arbeiten mit dem Secrets Manager auftreten können.

Informationen zu Problemen im Zusammenhang mit der Drehung finden Sie unter [the section called „Fehlerbehebung bei der -Rotation“](#).

Themen

- [Meldungen mit dem Hinweis „Zugriff verweigert“](#)
- [„Access denied“ \(Zugriff verweigert\) für temporäre Sicherheitsanmeldeinformationen](#)
- [Änderungen, die ich vornehme, sind nicht immer direkt sichtbar.](#)
- [„Cannot generate a data key with an asymmetric KMS key“ \(Kann keinen Datenschlüssel mit einem asymmetrischen KMS-Schlüssel generieren\) beim Erstellen eines Geheimnisses](#)
- [Eine AWS CLI oder AWS SDK-Operation kann mein Geheimnis nicht aus einem partiellen ARN finden](#)
- [Dieses Geheimnis wird von einem AWS Dienst verwaltet, und Sie müssen diesen Dienst verwenden, um es zu aktualisieren.](#)

Meldungen mit dem Hinweis „Zugriff verweigert“

Wenn Sie einen API-Aufruf wie GetSecretValue oder CreateSecret an Secrets Manager tätigen, benötigen Sie IAM-Berechtigungen, um diesen Aufruf tätigen zu können. Wenn Sie die Konsole verwenden, führt die Konsole dieselben API-Aufrufe in Ihrem Namen durch, sodass Sie auch über IAM-Berechtigungen verfügen müssen. Ein Administrator kann Berechtigungen gewähren, indem er Ihrem IAM-Benutzer oder einer Gruppe, der Sie angehören, eine IAM-Richtlinie anhängt. Wenn die Richtlinienenerklärungen, die diese Berechtigungen gewähren, Bedingungen enthalten, z. time-of-day B. Einschränkungen von IP-Adressen, müssen Sie diese Anforderungen auch erfüllen, wenn Sie die Anfrage senden. Weitere Informationen zum Anzeigen oder Ändern von Richtlinien für einen IAM-Benutzer, eine IAM-Gruppe oder eine IAM-Rolle finden Sie unter [Arbeiten mit Richtlinien](#) im IAM-Benutzerhandbuch. Informationen zu den für Secrets Manager erforderlichen Berechtigungen finden Sie unter [Authentifizierung und Zugriffskontrolle](#).

Wenn Sie API-Anfragen manuell ohne Verwendung der [AWS -SDKs](#) signieren, überprüfen Sie, ob Sie die [Anforderung korrekt signiert haben](#).

„Access denied“ (Zugriff verweigert) für temporäre Sicherheitsanmeldeinformationen

Stellen Sie sicher, dass der IAM-Benutzer oder die IAM-Rolle, die Sie zum Erstellen der Anforderung verwenden, über die entsprechenden Berechtigungen verfügt. Berechtigungen für temporäre Sicherheitsanmeldeinformationen werden von einem IAM-Benutzer oder einer IAM-Rolle abgeleitet. Dies bedeutet, dass sich die Berechtigungen auf diejenigen beschränken, die dem IAM-Benutzer oder der IAM-Rolle erteilt wurden. Weitere Informationen über die Berechtigungen für temporäre Sicherheitsanmeldeinformationen finden Sie unter [Kontrolle von Berechtigungen für temporäre Sicherheitsanmeldeinformationen](#) im IAM-Benutzerhandbuch.

Stellen Sie sicher, dass Ihre Anfragen korrekt signiert sind und die Anfrage richtig aufgebaut ist. Weitere Informationen finden Sie in der [Toolkit-Dokumentation](#) für das von Ihnen gewählte SDK oder [unter Verwenden temporärer Sicherheitsanmeldedaten zur Anforderung des Zugriffs auf AWS Ressourcen](#) im IAM-Benutzerhandbuch.

Stellen Sie sicher, dass die temporären Sicherheitsanmeldeinformationen nicht abgelaufen sind. Weitere Informationen finden Sie unter [Anfordern von temporären Sicherheitsanmeldeinformationen](#) im IAM-Benutzerhandbuch.

Informationen zu den für Secrets Manager erforderlichen Berechtigungen finden Sie unter [Authentifizierung und Zugriffskontrolle](#).

Änderungen, die ich vornehme, sind nicht immer direkt sichtbar.

Secrets Manager verwendet ein verteiltes Computing-Modell namens [letztendliche Konsistenz](#). Jede Änderung, die Sie in Secrets Manager (oder anderen AWS Diensten) vornehmen, dauert einige Zeit, bis sie von allen möglichen Endpunkten aus sichtbar wird. Die Verzögerung ergibt sich teilweise aus der Zeit, die erforderlich ist, um die Daten von Server zu Server, von Replikationszone zu Replikationszone und von einer Region der Welt in eine andere zu senden. Secrets Manager verwendet darüber hinaus Zwischenspeicherungen zur Verbesserung der Leistung, doch in einigen Fällen kann dies Zeit erfordern. Die Änderung ist möglicherweise erst sichtbar, wenn die Zeit für die vorher zwischengespeicherten Daten abgelaufen ist.

Entwickeln Sie Ihre globalen Anwendungen unter Berücksichtigung dieser potenziellen Verzögerungen. Stellen Sie darüber hinaus sicher, dass sie wie erwartet funktionieren, und zwar auch dann, wenn eine Änderung an einem Speicherort nicht sofort an einem anderen sichtbar ist.

Weitere Informationen darüber, wie sich die eventuelle Konsistenz auf einige andere AWS Dienste auswirkt, finden Sie unter:

- [Verwalten der Datenkonsistenz](#) im Datenbankentwicklerhandbuch zu Amazon Redshift
- [Amazon-S3-Datenkonsistenzmodell](#) im Benutzerhandbuch für Amazon Simple Storage Service
- [Sicherstellen der Konsistenz bei Verwendung von Amazon S3 und Amazon EMR für ETL-Workflows](#) im AWS Big Data Blog
- [Amazon EC2 Eventual Consistency](#) in der Amazon-EC2-API-Referenz

„Cannot generate a data key with an asymmetric KMS key“ (Kann keinen Datenschlüssel mit einem asymmetrischen KMS-Schlüssel generieren) beim Erstellen eines Geheimnisses

Secrets Manager verwendet einen [symmetrischen KMS-Verschlüsselungsschlüssel](#), der einem Secret zugeordnet ist, um einen Datenschlüssel für jeden Secret-Wert zu erzeugen. Sie können keinen asymmetrischen KMS-Schlüssel verwenden. Überprüfen Sie, ob Sie einen symmetrischen KMS-Verschlüsselungsschlüssel anstelle eines asymmetrischen KMS-Schlüssels verwenden. Detaillierte Anweisungen finden Sie unter [Identifizieren asymmetrischer KMS-Schlüssel](#).

Eine AWS CLI oder AWS SDK-Operation kann mein Geheimnis nicht aus einem partiellen ARN finden

In vielen Fällen kann Secrets Manager Ihr Geheimnis aus einem Teil eines ARN und nicht aus dem vollständigen ARN finden. Wenn der Name Ihres Geheimnisses jedoch mit einem Bindestrich gefolgt von sechs Zeichen endet, kann Secrets Manager das Geheimnis möglicherweise nicht nur aus einem Teil eines ARN finden. Stattdessen empfehlen wir Ihnen, den vollständigen ARN oder den Namen des Secrets zu verwenden.

Weitere Details

Secrets Manager enthält sechs zufällige Zeichen am Ende des Secret-Namens, um sicherzustellen, dass der Secret-ARN einzigartig ist. Wenn das ursprüngliche Secret gelöscht und anschließend ein neues Secret mit demselben Namen erstellt wird, haben die beiden Secrets aufgrund dieser Zeichen unterschiedliche ARNs. Benutzer mit Zugriff auf das alte Secret erhalten nicht automatisch Zugriff auf das neue Secret, da die ARNs unterschiedlich sind.

Secrets Manager erstellt einen ARN für ein Geheimnis mit Region, Konto, geheimen Namen und dann einem Bindestrich und sechs weiteren Zeichen wie folgt:

```
arn:aws:secretsmanager:us-east-2:111122223333:secret:SecretName-abcdef
```

Wenn Ihr geheimer Name mit einem Bindestrich und sechs Zeichen endet, kann Secrets Manager nur einen Teil des ARN verwenden, als würden Sie einen vollständigen ARN angeben. Zum Beispiel könnten Sie ein Geheimnis namens `MySecret-abcdef` mit dem ARN haben

```
arn:aws:secretsmanager:us-east-2:111122223333:secret:MySecret-abcdef-nutBrk
```

Wenn Sie den folgenden Vorgang aufrufen, der nur einen Teil des geheimen ARN verwendet, findet Secrets Manager das Geheimnis möglicherweise nicht.

```
$ aws secretsmanager describe-secret --secret-id arn:aws:secretsmanager:us-east-2:111122223333:secret:MySecret-abcdef
```

Dieses Geheimnis wird von einem AWS Dienst verwaltet, und Sie müssen diesen Dienst verwenden, um es zu aktualisieren.

Wenn Sie beim Versuch, ein Geheimnis zu ändern, auf diese Meldung stoßen, kann das Geheimnis nur mithilfe des in der Meldung aufgeführten Verwaltungsservices aktualisiert werden. Weitere Informationen finden Sie unter [Verwaltete Geheimnisse](#).

Um festzustellen, wer ein Geheimnis verwaltet, können Sie den Namen des Geheimnisses überprüfen. Bei Geheimnissen, die von anderen Services verwaltet werden, wird die ID des jeweiligen Services vorangestellt. Oder rufen Sie im AWS CLI [describe-secret](#) auf und überprüfen Sie dann das Feld `OwningService`

AWS Secrets Manager-Kontingente

Lese-APIs von Secrets Manager haben hohe TPS-Kontingente, und APIs der Steuerebene, die weniger häufig aufgerufen werden, haben niedrigere TPS-Kontingente. Wir empfehlen, `PutSecretValue` oder `UpdateSecret` nicht dauerhaft mehr als einmal alle 10 Minuten aufzurufen. Wenn Sie `PutSecretValue` oder `UpdateSecret` aufrufen, um den Secret-Wert zu aktualisieren, erstellt Secrets Manager eine neue Version des Secrets. Secrets Manager entfernt Versionen ohne Label, wenn es mehr als 100 davon gibt. Versionen, die jünger als 24 Stunden sind, werden nicht entfernt. Wenn Sie den Secret-Wert mehr als einmal alle 10 Minuten aktualisieren, erstellen Sie mehr Versionen als Secrets Manager entfernt, und Sie erreichen das Kontingent für Secret-Versionen.

Sie können mehrere Regionen in Ihrem Konto verwalten, und jeder Kontingent ist für jede Region spezifisch.

Wenn eine Anwendung in einem AWS-Konto einen Geheimnis in einem anderen Konto verwendet, wird dies als kontoübergreifende Anforderung bezeichnet. Bei kontoübergreifenden Anforderungen drosselt Secrets Manager das Konto der Identität, das die Anforderungen sendet, nicht das Konto, das das Geheimnis besitzt. Wenn beispielsweise eine Identität von Konto A ein Geheimnis in Konto B verwendet, gilt die Verwendung des Geheimnisses nur für die Kontingente in Konto A.

Secrets-Manager-Kontingente

Name	Standard	Anpas	Beschreibung
Kombinierte Rate von <code>DeleteResourcePolicy</code> -, <code>GetResourcePolicy</code> -, <code>PutResourcePolicy</code> - und <code>ValidateResourcePolicy</code> -API-Anforderungen	Jede unterstützte Region: 50 pro Sekunde	Nein	Die maximale Anzahl von Transaktionen pro Sekunde für <code>DeleteResourcePolicy</code> -, <code>GetResourcePolicy</code> -, <code>PutResourcePolicy</code> - und <code>ValidateResourcePolicy</code> -API-Anforderungen zusammengenommen.
Zusammengenommene Rate für <code>DescribeSecret</code> - und <code>GetSecretValue</code> -API-Anforderungen	Jede unterstützte Region: 10 000 pro Sekunde	Nein	Die maximale Anzahl an Transaktionen pro Sekunde für <code>DescribeS</code>

Name	Standard	Anpas	Beschreibung
			secret- und GetSecret Value-API-Anforderungen zusammengenommen.
Zusammengenommene Rate von PutSecretValue-, RemoveRegionsFromReplication-, ReplicateSecretToRegion-, StopReplicationToReplica-, UpdateSecret- und UpdateSecretVersionStage-API-Anforderungen	Jede unterstützte Region: 50 pro Sekunde	Nein	Die maximale Anzahl von Transaktionen pro Sekunde für die PutSecretValue-, RemoveRegionsFromReplication-, ReplicateSecretToRegion-, StopReplicationToReplica-, UpdateSecret- und UpdateSecretVersionStage-API-Anforderungen zusammengenommen.
Kombinierte Rate von RestoreSecret-API-Anforderungen	Jede unterstützte Region: 50 pro Sekunde	Nein	Die maximale Anzahl von Transaktionen pro Sekunde für RestoreSecret-API-Anforderungen.
Kombinierte Rate von RotateSecret- und CancelRotateSecret-API-Anforderungen	Jede unterstützte Region: 50 pro Sekunde	Nein	Die maximale Anzahl von Transaktionen pro Sekunde für RotateSecret- und CancelRotateSecret-API-Anforderungen zusammengenommen.

Name	Standard	Anpas	Beschreibung
Zusammengenommene Rate von TagResource- und UntagResource-API-Anforderungen	Jede unterstützte Region: 50 pro Sekunde	Nein	Die maximale Anzahl von Transaktionen pro Sekunde für die TagResource- und UntagResource-API-Anforderungen zusammengefasst.
Rate der BatchGetSecretValue-API-Anforderungen	Jede unterstützte Region: 100 pro Sekunde	Nein	Die maximale Anzahl von Transaktionen pro Sekunde für BatchGetSecretValue-API-Anforderungen.
Rate of CreateSecret-API-Anforderungen	Jede unterstützte Region: 50 pro Sekunde	Nein	Die maximale Anzahl von Transaktionen pro Sekunde für CreateSecret-API-Anforderungen.
Rate of DeleteSecret-API-Anforderungen	Jede unterstützte Region: 50 pro Sekunde	Nein	Die maximale Anzahl von Transaktionen pro Sekunde für DeleteSecret-API-Anforderungen.
Rate der GetRandomPassword-API-Anforderungen	Jede unterstützte Region: 50 pro Sekunde	Nein	Die maximale Anzahl von Transaktionen pro Sekunde für GetRandomPassword-API-Anforderungen.
Rate der ListSecretVersionIds-API-Anforderungen	Jede unterstützte Region: 50 pro Sekunde	Nein	Die maximale Anzahl von Transaktionen pro Sekunde für ListSecretVersionIds-API-Anforderungen.

Name	Standard	Anpas	Beschreibung
Rate der ListSecrets-API-Anforderungen	Jede unterstützte Region: 100 pro Sekunde	Nein	Die maximale Anzahl von Transaktionen pro Sekunde für ListSecrets-API-Anforderungen.
Ressourcenbasierte Richtlinienlänge	Jede unterstützte Region: 20 480	Nein	Die maximale Anzahl von Zeichen in einer ressourcenbasierten Berechtigungsrichtlinie, die an ein Secret angefügt ist.
Größe des Secret-Werts	Jede unterstützte Region: 65 536 Byte	Nein	Die maximale Größe eines verschlüsselten Secret-Werts. Wenn der Secret-Wert eine Zeichenfolge ist, ist dies die Anzahl der Zeichen, die im geheimen Wert zulässig sind.
Secrets	Jede unterstützte Region: 500 000	Nein	Die maximale Anzahl von Secrets in jeder AWS-Region von diesem AWS-Konto.
Staging-Bezeichnungen, die in allen Versionen eines Secrets angefügt sind	Jede unterstützte Region: 20	Nein	Die maximale Anzahl an Staging-Bezeichnungen, die in allen Versionen eines Secrets angefügt sind.
Versionen pro Secret	Jede unterstützte Region: 100	Nein	Die maximale Anzahl von Versionen eines Secrets.

Hinzufügen von Wiederholungen zu Ihrer Anwendung

Bei Ihrem AWS-Client kann es aufgrund unerwarteter Probleme auf Clientseite möglicherweise zu Fehlern bei Aufrufen von Secrets Manager kommen. Es kann auch sein, dass Aufrufe aufgrund der Ratenbegrenzung der Secrets Manager fehlschlagen. Wenn Sie ein API-Anforderungskontingent überschreiten, drosselt Secrets Manager die Anforderung. Es lehnt eine ansonsten gültige Anfrage ab und gibt einen throttling-Fehler zurück. Bei beiden Arten von Fehlern empfehlen wir Ihnen, den Anruf nach einer kurzen Wartezeit erneut zu versuchen. Das nennt man [Backoff- und Wiederholungsstrategie](#).

Wenn die folgenden Fehler auftreten, sollten Sie Ihrem Anwendungscode Wiederholungen hinzufügen:

Transiente Fehler und Ausnahmen

- `RequestTimeout`
- `RequestTimeoutException`
- `PriorRequestNotComplete`
- `ConnectionError`
- `HTTPClientError`

Serviceseitige Drosselung/Begrenzung von Fehlern und Ausnahmen

- `Throttling`
- `ThrottlingException`
- `ThrottledException`
- `RequestThrottledException`
- `TooManyRequestsException`
- `ProvisionedThroughputExceededException`
- `TransactionInProgressException`
- `RequestLimitExceeded`
- `BandwidthLimitExceeded`
- `LimitExceededException`
- `RequestThrottled`

- `SlowDown`

Weitere Informationen sowie Beispielcode zu Wiederholungen, exponentiellem Backoff und Jitter finden Sie in den folgenden Ressourcen:

- [Exponentielles Backoff und Jitter](#)
- [Timeouts, Wiederholungen und Backoff mit Jitter](#)
- [Wiederholversuche bei Fehlern und exponentielles Backoff in AWS.](#)

Dokumentverlauf

In der folgenden Tabelle werden die wichtigen Änderungen an der Dokumentation seit der letzten Version von beschrieben AWS Secrets Manager. Um Benachrichtigungen über Aktualisierungen dieser Dokumentation zu erhalten, können Sie einen RSS-Feed abonnieren.

Änderung	Beschreibung	Datum
Secrets Manager wechselt zur AWS verwalteten Richtlinie	Die <code>SecretsManagerReadWrite</code> verwaltete Richtlinie umfasst jetzt <code>redshift-serverless</code> Berechtigungen. Weitere Informationen finden Sie unter AWS Verwaltete Richtlinie für AWS Secrets Manager	12. März 2024

Frühere Aktualisierungen

In der folgenden Tabelle werden wichtige Änderungen beschrieben, die in den einzelnen Versionen des AWS Secrets Manager Benutzerhandbuchs vor Februar 2024 vorgenommen wurden.

Änderung	Beschreibung	Datum
Allgemeine Verfügbarkeit	Dies ist die erste öffentliche Version von Secrets Manager.	4. April 2018

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.