用户指南

AWS Secrets Manager

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

AWS Secrets Manager: 用户指南

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

Amazon 的商标和商业外观不得用于任何非 Amazon 的商品或服务，也不得以任何可能引起客户混淆、贬低或诋毁 Amazon 的方式使用。所有非 Amazon 拥有的其他商标均为各自所有者的财产，这些所有者可能附属于 Amazon、与 Amazon 有关联或由 Amazon 赞助，也可能不是如此。

Table of Contents

什么是 Secrets Manager？ 1
Secrets Manager 入门 1
符合标准 2
定价 2
访问 Secrets Manager 3
Secrets Manager 控制台 3
命令行工具 3
AWS SDKs 3
HTTPS 查询 API 4
Secrets Manager 端点 4
最佳实践 10
将凭证和其他敏感信息存储在 AWS Secrets Manager 10
查找代码中不受保护的密钥 10
为您的密钥选择一个加密密钥 11
使用缓存来检索密钥 11
轮换您的 密钥 12
降低使用 CLI 的风险 12
限制对密钥的访问 12
`BlockPublicPolicy` 条件 12
谨慎使用策略中的 IP 地址条件 13
使用 VPC 端点条件限制请求 13
复制密钥 14
监控密钥 14
在私有网络上运行基础设施 14
教程 15
Amazon CodeGuru Reviewer 15
替换硬编码的密钥 15
第 1 步：创建密钥 16
第 2 步：更新代码 18
第 3 步：更新密钥 18
后续步骤 19
替换硬编码的数据库凭证 19
第 1 步：创建密钥 20
第 2 步：更新代码 21

步骤 3：轮换秘密 21
后续步骤 23
交替用户轮换 23
权限 24
先决条件 24
步骤 1：创建 Amazon RDS 数据库用户 27
步骤 2：为用户凭证创建秘密 29
步骤 3：测试已轮换的秘密 30
步骤 4：清理资源 31
后续步骤 31
单用户轮换 32
权限 32
先决条件 32
步骤 1：创建 Amazon RDS 数据库用户 33
步骤 2：为数据库用户凭证创建密钥 34
步骤 3：测试轮换的密码 35
步骤 4：清理资源 35
后续步骤 35
创建密钥 36
AWS CLI 38
AWS SDK 39
密钥的内容 39
Metadata 40
密钥版本 40
密钥的 JSON 结构 41
Amazon RDS 和 Aurora 凭证 42
Amazon Redshift 凭证 44
Amazon Redshift Serverless 凭证 45
Amazon DocumentDB 凭证 45
Amazon Timestream for InfluxDB 密钥结构 46
Amazon ElastiCache 凭证 46
Active Directory 凭证 46
管理密钥 49
更新秘密值 49
AWS CLI 49
AWS SDK 50

使用 Secrets Manager 生成密码 50
将密钥回滚到以前的版本 50
更改秘密的加密密钥 51
AWS CLI 52
修改密钥 53
AWS CLI 54
AWS SDK 54
查找密钥 54
搜索筛选条件 55
AWS CLI 56
AWS SDK 57
删除密钥 57
AWS CLI 58
AWS SDK 59
恢复密钥 59
AWS CLI 60
AWS SDK 60
标记 密钥 61
查看标签基本知识 61
使用标签跟踪成本 62
了解标签限制 62
在控制台中标记机密 63
AWS CLI 64
API 64
SDK 65
多区域复制 66
AWS CLI 67
AWS SDK 68
将副本密钥升级为独立密钥 68
AWS CLI 68
AWS SDK 69
防止复制 69
排查 复制问题 70
选定区域中存在名称相同的密钥。 71
KMS 密钥上没有可用的权限来完成复制。 71
KMS 密钥已禁用或者未找到 71

您尚未启用要复制的区域。 ... 71
获取密钥 ... 72
Java ... 73
Java 与客户端缓存 ... 73
使用密钥中的凭证进行 JDBC 连接 ... 79
Java AWS 开发工具 ... 88
Python ... 90
Python 与客户端缓存 ... 90
Python AWS SD ... 96
获取批量密钥值 ... 97
.NET ... 99
.NET 与客户端缓存 ... 99
适用于 .NET 的 SDK ... 106
Go ... 109
Go 与客户端缓存 ... 109
Go AWS SDK ... 113
Rust ... 114
Rust 与客户端缓存 ... 114
Rust ... 117
Amazon EKS ... 117
基于服务账户的 IAM 角色（IRSA）的 ASCP ... 117
基于容器组身份的 ASCP ... 118
选择正确的方法 ... 118
安装适用于 Amazon EKS 的 ASCP ... 118
将 ASCP 与适用于 Amazon EKS 的容器组身份集成 ... 122
将 ASCP 与 IRSA 集成 Amazon EKS ... 125
ASCP 示例 ... 128
AWS Lambda ... 135
Lambda 入门 ... 135
参数存储集成 ... 136
Secrets Manager 代理 ... 136
Secrets Manager 代理是如何运作的 ... 136
了解 Secrets Manager 代理缓存 ... 137
构建 Secrets Manager 代理 ... 138
安装 Secrets Manager 代理 ... 141
使用 Secrets Manager 代理检索密钥 ... 146

了解refreshNow参数 148
配置选项 150
可选功能 151
日志记录 151
安全性注意事项 152
C++ 152
JavaScript 153
Kotlin 154
PHP 155
Ruby 156
AWS CLI 157
使用批量获取一组机密 AWS CLI 157
AWS 控制台 158
AWS Batch 159
AWS CloudFormation 159
GitHub 工作 160
先决条件 160
使用量 161
环境变量命名 162
示例 163
GitLab 165
注意事项 165
先决条件 165
AWS Secrets Manager 与集成 GitLab 166
故障排除 167
AWS IoT Greengrass 167
Parameter Store 168
轮换 密钥 169
托管轮换 169
通过 Lambda 函数进行轮换 170
自动轮换数据库密钥（控制台） 172
自动轮换非数据库密钥（控制台） 175
自动轮换（AWS CLI） 179
Lambda 函数轮换策略 182
Lambda 轮换函数 183
轮换函数模板 185

轮换权限 193
AWS Lambda 轮换功能的网络接入 197
轮换问题排查 198
轮换计划 211
轮换时段 211
Rate 表达式 212
Cron 表达式 212
立即轮换密钥 217
AWS CLI 217
查找未轮换的密钥 217
取消自动轮换 218
由其他服务管理的密钥 219
使用密钥的服务 220
App Runner 222
AWS App2Container 222
AWS AppConfig 222
Amazon AppFlow 222
AWS AppSync 223
Amazon Athena 223
Amazon Aurora 223
AWS CodeBuild 223
Amazon Data Firehose 224
AWS DataSync 224
Amazon DataZone 224
AWS Direct Connect 224
AWS Directory Service 225
Amazon DocumentDB 225
AWS Elastic Beanstalk 225
Amazon Elastic Container Registry 225
Amazon Elastic Container Service 226
Amazon ElastiCache 226
AWS Elemental Live 226
AWS Elemental MediaConnect 227
AWS Elemental MediaConvert 227
AWS Elemental MediaLive 227
AWS Elemental MediaPackage 227

AWS Elemental MediaTailor 227
Amazon EMR 228
EMR 已开启 EC2 228
EMR Serverless 228
Amazon EventBridge 228
Amazon FSx 229
AWS Glue DataBrew 229
AWS Glue Studio 229
AWS IoT SiteWise 229
Amazon Kendra 229
Amazon Kinesis Video Streams 230
AWS Launch Wizard 230
Amazon Lookout for Metrics 230
Amazon Managed Grafana 230
AWS Managed Services 231
Amazon Managed Streaming for Apache Kafka 231
Amazon Managed Workflows for Apache Airflow 231
AWS Marketplace 231
AWS Migration Hub 231
AWS Panorama 232
AWS 并行计算服务 232
AWS ParallelCluster 232
Amazon Q 233
Amazon OpenSearch Ingestion 233
AWS OpsWorks for Chef Automate 233
Amazon QuickSight 233
Amazon RDS 234
Amazon Redshift 234
Amazon Redshift 查询编辑器 v2 234
亚马逊 SageMaker AI 235
AWS SCT 235
Amazon Timestream for InfluxDB 236
AWS Toolkit for JetBrains 236
AWS Transfer Family 236
AWS Wickr 236
AWS CloudFormation 238

创建密钥 238
JSON 239
YAML 239
使用自动轮换的 Amazon RDS 凭证创建秘密 239
使用 Amazon Redshift 凭证创建密钥 240
使用 Amazon DocumentDB 凭证创建密钥 240
JSON 240
YAML 245
Secrets Manager 的使用方式 AWS CloudFormation 247
AWS CDK 248
监控密钥 249
使用登录 AWS CloudTrail 249
AWS CLI 250
CloudTrail 条目 250
使用监视器 CloudWatch 255
CloudWatch 警报 255
将 Secrets Manager 活动与 EventBridge 256
将所有更改与指定密钥匹配 256
在轮换密钥值时匹配事件 256
监控计划删除的密钥 257
步骤 1：配置 CloudTrail 日志文件传输到 CloudWatch 日志 257
步骤 2：创建 CloudWatch警报 258
第 3 步：测试 CloudWatch警报 259
监控密钥的合规性 259
监控 Secrets Manager 成本 260
使用以下方法检测威胁 GuardDuty 260
合规性验证 261
合规性标准 261
安全性 263
降低使用存储 AWS Secrets Manager 密钥 AWS CLI 的风险 263
身份验证和访问控制 265
权限参考 266
Secrets Manager 管理员权限 266
访问密钥的权限 266
Lambda 轮换函数的权限 266
加密密钥权限 267

复制权限 267
基于身份的策略 267
基于资源的策略 274
使用标签控制对密钥的访问 280
AWS 托管策略 282
确定谁有权限访问您的 密钥 285
跨账户访问 286
本地访问 289
Secrets Manager 中的数据保护 289
静态加密 290
传输中加密 290
互联网络流量隐私 290
加密密钥管理 291
密钥加密和解密 291
选择一把 AWS KMS 钥匙 291
什么是加密？ 292
加密和解密流程 292
KMS 密钥的权限 293
Secrets Manager 如何使用您的 KMS 密钥 293
AWS 托管式密钥 (`aws/secretsmanager`) 的密钥策略 295
Secrets Manager 加密上下文 297
监控 Secrets Manager 的互动 AWS KMS 299
基础结构安全性 303
VPC 端点（AWS PrivateLink） 303
共享子网 304
IPv4 和 IPv6 访问权限 304
什么是 IPv6？ 304
使用双栈策略 305
IPv6 添加到策略中 305
验证您的客户端支持 IPv6 307
恢复能力 308
后量子 TLS 308
故障排除 311
“访问被拒绝”消息 311
对于临时安全凭证的“拒绝访问” 311
并非始终立即显示我所做的更改。 312

在创建秘密时收到“Cannot generate a data key with an asymmetric KMS key”（无法使用非对称 KMS 密钥生成数据密钥） 312
AWS CLI 或 S AWS DK 操作无法从部分 ARN 中找到我的秘密 312
此密钥由 AWS 服务管理，您必须使用该服务对其进行更新。 313
使用时 Python 模块导入失败 `Transform: AWS::SecretsManager-2024-09-16` 313
限额 314
Secrets Manager 配额 314
将重试添加到您的应用程序 316
文档历史记录 318
早期更新 318
..... cccxix

什么是 AWS Secrets Manager？

AWS Secrets Manager 借助，您可以在数据库凭证、应用程序凭证、 OAuth 令牌、API 密钥和其他密钥的整个生命周期内对其进行管理、检索和轮换。许多 AWS 服务将密钥存储在 Secrets Manager 中。

Secrets Manager 无需应用程序源代码中的硬编码凭证，因此可帮助您改进安全状况。将凭证存储在 Secrets Manager 中有助于避免检查您应用程序或组件的任何人泄露这些凭证。您可以将硬编码凭证替换为对 Secrets Manager 服务的运行时系统调用，以便在需要时动态检索凭证。

使用 Secrets Manager，您可以为密钥配置自动轮换计划。这样，您就可以将长期密钥替换为短期密钥，从而显著降低泄露风险。由于凭证不再与应用程序存储在一起，所以轮换凭证不再需要更新应用程序并将更改部署到应用程序客户端。

对于您的组织可能拥有的其他类型的密钥：

- AWS 凭证 — 我们推荐AWS Identity and Access Management。
- 加密密钥 – 建议使用 AWS Key Management Service。
- SSH 密钥 — 建议使用 Amazon Instance EC2 Connect。
- 私有密钥和证书 – 建议使用 AWS Certificate Manager。

Secrets Manager 入门

如果不熟悉 Secrets Manager，请从以下教程之一开始：

- the section called “替换硬编码的密钥 ”
- the section called “替换硬编码的数据库凭证 ”
- the section called “交替用户轮换”
- the section called “单用户轮换”

可使用密钥执行的其他任务：

- 管理密钥
- 控制对密钥的访问
- 获取密钥
- 轮换 密钥

- 监控密钥
- 监控密钥的合规性
- 在中创建密钥 AWS CloudFormation

符合标准

AWS Secrets Manager 通过了审核，符合多项标准，可用于需要获取合规性认证的解决方案中。有关更多信息，请参阅 合规性验证。

定价

使用 Secrets Manager 时，仅按实际使用量收费，无最低费用或设置费用。标记为已删除的密钥不收取任何费用。有关当前完整定价列表，请参阅 AWS Secrets Manager 定价。要监控成本，请参阅 the section called “监控 Secrets Manager 成本”。

您可以使用 Secrets Manager 创建的免费加密您的密钥。 AWS 托管式密钥 `aws/secretsmanager`如果您创建自己的 KMS 密钥来加密您的机密，将按当前费 AWS KMS 率向您 AWS 收费。有关更多信息，请参阅AWS Key Management Service 定价。

当您开启自动轮换（托管轮换除外）时，Secrets Manager 会使用 AWS Lambda 函数来轮换密钥，并且按当前 Lambda 费率向您收取轮换函数的费用。有关更多信息，请参阅AWS Lambda 定价。

如果 AWS CloudTrail 在您的账户上启用了，则可以获取 Secrets Manager 进行的 API 调用的日志。Secrets Manager 将所有事件记录为管理事件。 AWS CloudTrail 免费存储所有管理事件的第一个副本。但是，如果启用通知，可能会对 Amazon S3 的日志存储和 Amazon SNS 产生费用。此外，如果您设置了其他跟踪，管理事件的其他副本可能会产生费用。有关更多信息，请参阅AWS CloudTrail 定价。

您可以在 Secrets Manager 中使用成本分配标签来跟踪和分类与特定密钥或项目相关的费用。有关更多信息，请参阅the section called “标记 密钥”本指南和 AWS Billing 用户指南中的使用 AWS 成本分配标签。

访问权限 AWS Secrets Manager

您可以通过以下任何方式使用 Secrets Manager：

- Secrets Manager 控制台
- 命令行工具
- AWS SDKs
- HTTPS 查询 API
- AWS Secrets Manager 端点

Secrets Manager 控制台

您可以使用基于浏览器的 Secrets Manager 控制台管理密钥，并可使用该控制台执行与密钥相关的几乎所有任务。

命令行工具

AWS 命令行工具允许您在系统命令行中发出命令以执行 Secrets Manager 和其他 AWS 任务。与使用控制台相比，此方法更快、更方便。如果要生成脚本来执行 AWS 任务，则命令行工具可能很有用。

当您在命令 shell 中输入命令时，存在访问命令历史记录或实用程序可以访问您命令参数的风险。请参阅the section called “降低使用存储 AWS Secrets Manager 密钥 AWS CLI 的风险”。

命令行工具会自动使用 AWS 区域中服务的默认终端节点。您可以为 API 请求指定其他端点。请参阅the section called “Secrets Manager 端点”。

AWS 提供了两组命令行工具：

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

AWS SDKs

AWS SDKs 由适用于各种编程语言和平台的库和示例代码组成。 SDKs 包括对请求进行加密签名、管理错误和自动重试请求等任务。要下载并安装其中任何一个 SDKs，请参阅 Amazon Web Services 工具。

在某个 AWS 区域中 AWS SDKs 自动使用服务的默认终端节点。您可以为 API 请求指定其他端点。请参阅the section called “Secrets Manager 端点”。

要获得开发工具包文档，请参阅：

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

HTTPS 查询 API

HTTPS 查询 API 允许你以编程方式访问 Secrets Manager 和 AWS。HTTPS 查询 API 允许您直接向服务发出 HTTPS 请求。

尽管你可以直接调用 Secrets Manager HTTPS 查询 API，但我们建议你 SDKs 改用其中一个。开发工具包可以执行许多您必须手动执行的有用任务。例如， SDKs 会自动对您的请求进行签名，并将响应转换为语法上适合您的语言的结构。

要对 Secrets Manager 进行 HTTPS 调用，您需要连接到 ???。

AWS Secrets Manager 端点

要以编程方式连接到 Secrets Manager，您可以使用端点，即服务入口点的 URL。Secrets Manager 端点是双栈端点，这意味着它们同时支持 IPv4 和。 IPv6

Secrets Manager 在某些区域提供支持美国联邦信息处理标准 (FIPS) 140-2 的端点。

Secrets Manager 支持 TLS 1.2 和 1.3。Secrets Manager 在除中国区域之外的所有区域都支持 PQTLS。

> **Note**
>
> Python AWS SDK IPv6 以及按顺序 AWS CLI 尝试调用，因此，如果您未 IPv6 启用，则可能需要一段时间才能调用超时并重试。 IPv4 IPv4要解决此问题，您可以 IPv6 完全禁用或迁移到 IPv6。

以下是 Secrets Manager 的服务端点。请注意，命名与典型的双堆栈命名约定不同。有关在 Secrets Manager 中使用双栈寻址的信息，请参阅IPv4 和 IPv6 访问权限。

区域名称	区域	端点	协议
美国东部（俄亥俄州）	us-east-2	secretsmanager.us-east-2.amazonaws.com	HTTPS
		secretsmanager-fips.us-east-2.amazonaws.com	HTTPS
美国东部（弗吉尼亚州北部）	us-east-1	secretsmanager.us-east-1.amazonaws.com	HTTPS
		secretsmanager-fips.us-east-1.amazonaws.com	HTTPS
美国西部（加利福尼亚北部）	us-west-1	secretsmanager.us-west-1.amazonaws.com	HTTPS
		secretsmanager-fips.us-west-1.amazonaws.com	HTTPS
美国西部（俄勒冈州）	us-west-2	secretsmanager.us-west-2.amazonaws.com	HTTPS
		secretsmanager-fips.us-west-2.amazonaws.com	HTTPS
非洲（开普敦）	af-south-1	secretsmanager.af-south-1.amazonaws.com	HTTPS

区域名称	区域	端点	协议
亚太地区（香港）	ap-east-1	secretsmanager.ap-east-1.amazonaws.com	HTTPS
亚太地区（海得拉巴）	ap-south-2	secretsmanager.ap-south-2.amazonaws.com	HTTPS
亚太地区（雅加达）	ap-southeast-3	secretsmanager.ap-southeast-3.amazonaws.com	HTTPS
亚太地区（马来西亚）	ap-southeast-5	secretsmanager.ap-southeast-5.amazonaws.com	HTTPS
亚太地区（墨尔本）	ap-southeast-4	secretsmanager.ap-southeast-4.amazonaws.com	HTTPS
亚太地区（孟买）	ap-south-1	secretsmanager.ap-south-1.amazonaws.com	HTTPS
亚太地区（大阪）	ap-northeast-3	secretsmanager.ap-northeast-3.amazonaws.com	HTTPS
亚太地区（首尔）	ap-northeast-2	secretsmanager.ap-northeast-2.amazonaws.com	HTTPS
亚太地区（新加坡）	ap-southeast-1	secretsmanager.ap-southeast-1.amazonaws.com	HTTPS
亚太地区（悉尼）	ap-southeast-2	secretsmanager.ap-southeast-2.amazonaws.com	HTTPS

区域名称	区域	端点	协议
亚太地区（台北）	ap-east-2	secretsmanager.ap-east-2.amazonaws.com	HTTPS
亚太地区（泰国）	ap-southeast-7	secretsmanager.ap-southeast-7.amazonaws.com	HTTPS
亚太地区（东京）	ap-northeast-1	secretsmanager.ap-northeast-1.amazonaws.com	HTTPS
加拿大（中部）	ca-central-1	secretsmanager.ca-central-1.amazonaws.com secretsmanager-fips.ca-central-1.amazonaws.com	HTTPS HTTPS
加拿大西部（卡尔加里）	ca-west-1	secretsmanager.ca-west-1.amazonaws.com secretsmanager-fips.ca-west-1.amazonaws.com	HTTPS HTTPS
欧洲地区（法兰克福）	eu-central-1	secretsmanager.eu-central-1.amazonaws.com	HTTPS
欧洲地区（爱尔兰）	eu-west-1	secretsmanager.eu-west-1.amazonaws.com	HTTPS
欧洲地区（伦敦）	eu-west-2	secretsmanager.eu-west-2.amazonaws.com	HTTPS
欧洲地区（米兰）	eu-south-1	secretsmanager.eu-south-1.amazonaws.com	HTTPS
欧洲地区（巴黎）	eu-west-3	secretsmanager.eu-west-3.amazonaws.com	HTTPS

区域名称	区域	端点	协议
欧洲（西班牙）	eu-south-2	secretsmanager.eu-south-2.amazonaws.com	HTTPS
欧洲地区（斯德哥尔摩）	eu-north-1	secretsmanager.eu-north-1.amazonaws.com	HTTPS
欧洲（苏黎世）	eu-central-2	secretsmanager.eu-central-2.amazonaws.com	HTTPS
以色列（特拉维夫）	il-central-1	secretsmanager.il-central-1.amazonaws.com	HTTPS
墨西哥（中部）	mx-central-1	secretsmanager.mx-central-1.amazonaws.com	HTTPS
中东（巴林）	me-south-1	secretsmanager.me-south-1.amazonaws.com	HTTPS
中东（阿联酋）	me-central-1	secretsmanager.me-central-1.amazonaws.com	HTTPS
南美洲（圣保罗）	sa-east-1	secretsmanager.sa-east-1.amazonaws.com	HTTPS
AWS GovCloud（美国东部）	us-gov-east-1	secretsmanager.us-gov-east-1.amazonaws.com secretsmanager-fips.us-gov-east-1.amazonaws.com	HTTPS HTTPS

区域名称	区域	端点	协议	
AWS GovCloud（美国西部）	us-gov-west-1	secretsmanager.us-gov-west-1.amazonaws.com secretsmanager-fips.us-gov-west-1.amazonaws.com	HTTPS HTTPS	

AWS Secrets Manager 最佳实践

Secrets Manager 提供了在您开发和实施自己的安全策略时需要考虑的大量安全功能。以下最佳实操是一般准则，并不代表完整的安全解决方案。这些最佳实操可能不适合您的环境或不满足您的环境要求，请将其视为有用的考虑因素而不是惯例。

请考虑下面存储和管理密钥的最佳实践：

- 将凭证和其他敏感信息存储在 AWS Secrets Manager
- 查找代码中不受保护的密钥
- 为您的密钥选择一个加密密钥
- 使用缓存来检索密钥
- 轮换您的 密钥
- 降低使用 CLI 的风险
- 限制对密钥的访问
- 复制密钥
- 监控密钥
- 在私有网络上运行基础设施

将凭证和其他敏感信息存储在 AWS Secrets Manager

Secrets Manager 可帮助您改善安全状况和合规性，并降低未经授权访问您的敏感信息的风险。Secrets Manager 使用您拥有并存储在 AWS Key Management Service (AWS KMS) 中的加密密钥对静态密钥进行加密。当您检索密钥时，Secrets Manager 会解密密钥并通过 TLS 将其安全地传输到您的本地环境。有关更多信息，请参阅 创建密钥。

查找代码中不受保护的密钥

CodeGuru Reviewer 与 Secrets Manager 集成，使用秘密探测器在你的代码中查找未受保护的机密。密钥检测器会搜索硬编码密码、数据库连接字符串、用户名等。有关更多信息，请参阅 the section called “Amazon CodeGuru Reviewer”。

Amazon Q 可以扫描您的代码库以查找安全漏洞和代码质量问题，从而改善整个开发周期内应用程序的状况。有关更多信息，请参阅《Amazon Q 开发者版用户指南》中的 Scanning your code with Amazon Q。

为您的密钥选择一个加密密钥

在大多数情况下，我们建议使用`aws/secretsmanager` AWS 托管密钥来加密机密。使用它不产生任何费用。

为了能够从其他账户访问密钥或将密钥政策应用于加密密钥，请使用客户自主管理型密钥对密钥进行加密。

- 在密钥策略中，为`kms:ViaService`条件密钥分配值`secretsmanager.<region>.amazonaws.com`。这会将密钥的使用限制为仅限来自 Secrets Manager 的请求。
- 为了进一步将密钥的使用限制为仅来自 Secrets Manager 且具有正确上下文的请求，请通过创建以下项将 Secrets Manager 加密上下文中的键或值用作使用 KMS 密钥的条件：
  - IAM 策略或密钥政策中的字符串条件运算符
  - 在授权中创建授权约束

有关更多信息，请参阅 the section called “密钥加密和解密”。

使用缓存来检索密钥

为了最有效地使用您的密钥，我们建议您使用以下受支持的 Secrets Manager 缓存组件之一来缓存您的密钥并仅在需要时更新它们：

- Java 与客户端缓存
- Python 与客户端缓存
- .NET 与客户端缓存
- Go 与客户端缓存
- Rust 与客户端缓存
- AWS 参数和密钥 Lambda 扩展
- the section called “Amazon EKS”
- 用于the section called “Secrets Manager 代理”在亚马逊弹性容器服务 AWS Lambda、亚马逊弹性容器服务、亚马逊弹性 Kubernetes Service 和亚马逊弹性计算云等环境中对 Secrets Manager 密钥的使用进行标准化。

轮换您的 密钥

如果您很长一段时间未更改密钥，密钥泄露的可能性就会增大。使用 Secrets Manager，您可以设置每四小时自动轮换一次。Secrets Manager 提供了两种轮换策略：单用户 和 交替用户。有关更多信息，请参阅 轮换 密钥。

降低使用 CLI 的风险

使用调用 AWS 操作时，可以在命令 shell 中输入这些命令。 AWS CLI 大多数命令 shell 都提供了可能会泄露您的密钥的功能，例如日志记录和查看上次输入的命令的功能。在使用 AWS CLI 输入敏感信息之前，请务必 the section called “降低使用存储 AWS Secrets Manager 密钥 AWS CLI 的风险”。

限制对密钥的访问

在控制对您的密钥的访问的 IAM 策略语句中，使用最低权限访问原则。您可以使用 IAM 角色和策略、资源策略和基于属性的访问权限控制（ABAC）。有关更多信息，请参阅 the section called “身份验证和访问控制”。

主题

- 阻止对密钥的广泛访问
- 谨慎使用策略中的 IP 地址条件
- 使用 VPC 端点条件限制请求

阻止对密钥的广泛访问

在允许操作 `PutResourcePolicy` 的身份策略中，我们建议您使用 `BlockPublicPolicy: true`。这种情况意味着只有在策略不允许广泛访问的情况下，用户才能将资源策略附加到密钥。

Secrets Manager 使用 Zelkova 自动推理来分析资源策略，以确定是否存在宽泛访问权限问题。有关 Zelkova 的更多信息，请参阅安全博客上的 “如何 AWS 使用自动推理来帮助您实现大规模 AWS 安全”。

以下示例显示了如何使用 `BlockPublicPolicy`。

JSON

```
{
```

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

谨慎使用策略中的 IP 地址条件

在允许或拒绝访问 Secrets Manager 的策略语句中指定 IP 地址条件运算符或 `aws:SourceIp` 条件键时，请务必小心。例如，如果您将限制对来自公司网络 IP 地址范围的请求 AWS 执行操作的策略附加到密钥，则您作为 IAM 用户从公司网络调用请求的请求将按预期工作。但是，如果您允许其他服务代表您访问密钥，例如使用 Lambda 函数启用轮换，则该函数会从 AWS内部地址空间调用 Secrets Manager 操作。受该策略影响并使用 IP 地址筛选器的请求将会失败。

此外，当请求来自 Amazon VPC 终端节点时，`aws:sourceIP` 条件键也不起作用。要限制对特定 VPC 终端节点的请求，请使用 the section called “使用 VPC 端点条件限制请求”。

使用 VPC 端点条件限制请求

要允许或拒绝对来自特定 VPC 或 VPC 终端节点的请求的访问，请使用 `aws:SourceVpc` 来限制对来自指定 VPC 的请求的访问，或 `aws:SourceVpce` 来限制对来自指定 VPC 终端节点的请求的访问。请参阅the section called “示例：权限和 VPCs”。

- `aws:SourceVpc` 将访问限制为来自指定 VPC 的请求。
- `aws:SourceVpce` 将访问限制为来自指定 VPC 终端节点的请求。

如果在允许或拒绝访问 Secrets Manager 密钥的资源策略语句中使用这些条件键，可能会无意中拒绝访问代表您使用 Secrets Manager 访问密钥的服务。只有部分 AWS 服务可以在您的 VPC 内使用终端节点运行。如果将密钥的请求限制为 VPC 或 VPC 终端节点，则从未针对该服务配置的服务中调用 Secrets Manager 可能会失败。

请参阅the section called “VPC 端点（AWS PrivateLink）”。

复制密钥

Secrets Manager 可以自动将您的密钥复制到多个 AWS 区域，以满足您的弹性或灾难恢复要求。有关更多信息，请参阅 多区域复制。

监控密钥

Secrets Manager 使您能够通过与 AWS 日志、监控和通知服务集成来审核和监控密钥。有关更多信息，请参阅：

- the section called “使用登录 AWS CloudTrail ”
- the section called “使用监视器 CloudWatch”
- the section called “监控密钥的合规性”
- the section called “监控 Secrets Manager 成本”
- the section called “使用以下方法检测威胁 GuardDuty”

在私有网络上运行基础设施

我们建议您在无法从私有网络访问的专用网络上运行尽可能多的基础设施。您可以通过创建接口 VPC 终端节点在 VPC 与 Secrets Manager 之间建立私有连接。有关更多信息，请参阅 the section called “VPC 端点（AWS PrivateLink）”。

AWS Secrets Manager 教程

主题

- 使用 Amazon CodeGuru Reviewer 查找代码中未受保护的机密
- 将硬编码的机密移至 AWS Secrets Manager
- 将硬编码的数据库凭据移至 AWS Secrets Manager
- 为 AWS Secrets Manager设置交替用户轮换
- 为 AWS Secrets Manager设置单用户轮换

使用 Amazon CodeGuru Reviewer 查找代码中未受保护的机密

Amazon CodeGuru Reviewer 是一项使用程序分析和机器学习来检测开发人员难以发现的潜在缺陷的服务，并提供改进您的 Java 和 Python 代码的建议。 CodeGuru Reviewer 与 Secrets Manager 集成，可在你的代码中查找未受保护的机密。有关它能找到的机密类型，请参阅 Amazon Reviewer 用户指南中的 CodeGuru Reviewer 检测到的 CodeGuru 机密类型。

找到硬编码的密钥后，请立即将其替换：

- the section called “替换硬编码的数据库凭证 ”
- the section called “替换硬编码的密钥 ”

将硬编码的机密移至 AWS Secrets Manager

如果代码中存在明文密钥，我们建议将其轮换并存储到 Secrets Manager 中。将密钥移动到 Secrets Manager 后，您的代码将直接从 Secrets Manager 中检索密钥，从而解决了任何看到代码的人会看到密钥的问题。轮换密钥会吊销当前硬编码的密钥，使其不再有效。

关于数据库凭证密钥，请参见将硬编码的数据库凭据移至 AWS Secrets Manager。

在开始之前，您需要确定谁需要访问该密钥。我们建议使用两个 IAM 角色来管理密钥的权限：

- 负责管理组织中的密钥的角色。有关更多信息，请参阅 the section called “Secrets Manager 管理员权限”。您将使用此角色创建和轮换密钥。
- 可以在运行时使用密钥的角色，例如在本教程中使用的角色*RoleToRetrieveSecretAtRuntime*。您的代码将代入此角色以检索密钥。在本教程中，您将

向该角色仅授予检索一个密钥值的权限，并您将使用密钥的资源策略授予权限。有关其他替代方法，请参阅the section called “后续步骤”。

步骤：

- 第 1 步：创建密钥
- 第 2 步：更新代码
- 第 3 步：更新密钥
- 后续步骤

第 1 步：创建密钥

第一步是将现有硬编码的密钥复制到 Secrets Manager 中的密钥中。如果密钥与 AWS 资源相关，请将其存储在与资源相同的区域。否则，请将其存储在对于您的使用场景而言延迟最低的区域中。

创建密钥（控制台）

1. 打开 Secrets Manager 控制台，网址为https://console.aws.amazon.com/secretsmanager/。
2. 选择存储新密钥。
3. 在 Choose secret type（选择密钥类型）页面上，执行以下操作：
   a. 对于密钥类型，请选择其他密钥类型。
   b. 以 Key/value pairs（键值对）或者 Plaintext（明文）格式输入密钥。一些示例：

      API key

      key/value 成对输入：

      ClientID：*my_client_id*

      ClientSecret：*wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY*

      OAuth token

      输入明文：

      AKIAI44QH8DHBEXAMPLE

Digital certificate

输入明文：

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

Private key

输入明文：

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

c. 对于 Encryption key（加密密钥），选择 aws/secretsmanager 使用 Secrets Manager 的 AWS 托管式密钥 。使用此密钥不产生任何费用。例如，您还可以使用自己的客户管理型密钥来访问来自其他 AWS 账户的密钥。有关使用客户托管密钥的成本的信息，请参阅 定价。

d. 选择下一步。

4. 在 Choose secret type（选择密钥类型）页面上，执行以下操作：

a. 输入一个描述性的 Secret name（密钥名称）和 Description（说明）。

b. 在 Resource permissions（资源权限）中，选择 Edit permissions（编辑权限）。粘贴以下*RoleToRetrieveSecretAtRuntime*允许检索密钥的策略，然后选择保存。

JSON

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

```
    ]
}
```

c. 在页面底部，选择 Next。

5. 在 Configure rotation（配置轮换）页面上，将轮换禁用。选择下一步。
6. 在 Review (审核) 页上，审核您的密钥详细信息，然后选择 Store (存储)。

第 2 步：更新代码

您的代码必须担任 IAM 角色*RoleToRetrieveSecretAtRuntime*才能检索密钥。有关更多信息，请参阅切换到 IAM 角色 (AWS API)。

然后，您可以使用 Secrets Manager 提供的示例代码更新您的代码，以检索 Secrets Manager 中的密钥。

查找示例代码

1. 打开 Secrets Manager 控制台，网址为https://console.aws.amazon.com/secretsmanager/。
2. 在密钥列表页上，选择您的密钥。
3. 向下滚动到 Sample code（示例代码）。选择您的编程语言，然后复制代码片段。

移除应用程序中的硬编码密钥并粘贴此代码片段。根据代码语言的不同，您可能需要在片段中添加对函数或方法的调用。

使用密钥代替硬编码密钥，测试您的应用程序是否符合预期。

第 3 步：更新密钥

最后一步是吊销并更新硬编码的密钥。请参阅密钥的来源以查找吊销和更新密钥的说明。例如，您可能需要停用当前密钥并生成一个新密钥。

用新值更新密钥

1. 打开 Secrets Manager 控制台，网址为https://console.aws.amazon.com/secretsmanager/。
2. 选择 Secrets（密钥），然后选择该密钥。
3. 在 Secret details（密钥详细信息）页面上，向下滚动并选择 Retrieve secret value（检索密钥值），然后选择 Edit（编辑）。
4. 更新密钥然后选择 Save（保存）。

然后，测试您的应用程序按照预期那样在使用新密钥。

后续步骤

从代码中移除硬编码的密钥后，接下来需要注意以下事项：

- 要在你的 Java 和 Python 应用程序中查找硬编码的机密，我们建议使用 Amazon CodeGuru Reviewer。
- 您可以通过缓存密钥来提高性能并降低成本。有关更多信息，请参阅 获取密钥。
- 对于从多个区域访问的密钥，请考虑复制密钥以减少延迟。有关更多信息，请参阅 多区域复制。
- 在本教程中，您*RoleToRetrieveSecretAtRuntime*仅授予了检索密钥值的权限。要向角色授予更多权限（例如获取有关密钥的元数据或查看密钥列表），请参阅the section called “基于资源的策略”。
- 在本教程中，您使用密钥的资源策略授予了权限。*RoleToRetrieveSecretAtRuntime*有关授予权限的其他方法，请参阅the section called “基于身份的策略”。

将硬编码的数据库凭据移至 AWS Secrets Manager

如果代码中存在明文数据库凭证，我们建议您将凭证移动到 Secrets Manager，然后立即将其轮换。将凭证移动到 Secrets Manager 后，您的代码将直接从 Secrets Manager 中检索凭证，从而解决了任何看到代码的人会看到凭证的问题。轮换密钥会更新密码，然后吊销当前硬编码的密码，使其不再有效。

对于 Amazon RDS、Amazon Redshift 和 Amazon DocumentDB 数据库，请使用本页中的步骤将硬编码的凭证移动到 Secrets Manager。对于其他类型的凭证和其他密钥，请参阅the section called “替换硬编码的密钥 ”。

在开始之前，您需要确定谁需要访问该密钥。我们建议使用两个 IAM 角色来管理密钥的权限：

- 负责管理组织中的密钥的角色。有关更多信息，请参阅 the section called “Secrets Manager 管理员权限”。您将使用此角色创建和轮换密钥。
- 在本教程中，一个可以在运行时使用凭据*RoleToRetrieveSecretAtRuntime*的角色。您的代码将代入此角色以检索密钥。

步骤：

- 第 1 步：创建密钥
- 第 2 步：更新代码

- 步骤 3：轮换秘密
- 后续步骤

第 1 步：创建密钥

第一步是将现有硬编码的凭证复制到 Secrets Manager 中的密钥中。为了实现低延迟，可将密钥存储在与数据库相同的区域中。

创建密钥

1. 打开 Secrets Manager 控制台，网址为https://console.aws.amazon.com/secretsmanager/。
2. 选择存储新密钥。
3. 在 Choose secret type（选择密钥类型）页面上，执行以下操作：
   a. 对于密钥类型，选择要存储的数据库凭证类型：
      - Amazon RDS 数据库
      - Amazon DocumentDB 数据库
      - Amazon Redshift 数据仓库。
      - 有关其他类型的密钥，请参阅替换硬编码的密钥。
   b. 对于凭证，请输入数据库现有的硬编码凭证。
   c. 对于 Encryption key（加密密钥），选择 aws/secretsmanager 使用 Secrets Manager 的 AWS 托管式密钥 。使用此密钥不产生任何费用。例如，您还可以使用自己的客户管理型密钥来访问来自其他 AWS 账户的密钥。有关使用客户托管密钥的成本的信息，请参阅 定价。
   d. 对于 Database（数据库），请选择您的数据库。
   e. 选择下一步。
4. 在 Configure secret（配置密钥）页面上，执行以下操作：
   a. 输入一个描述性的 Secret name（密钥名称）和 Description（说明）。
   b. 在 Resource permissions（资源权限）中，选择 Edit permissions（编辑权限）。粘贴以下*RoleToRetrieveSecretAtRuntime*允许检索密钥的策略，然后选择保存。

      JSON

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

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

c. 在页面底部，选择 Next。

5. 在 Configure rotation（配置轮换）页面上，暂时将轮换禁用。稍后您会将其启用。选择下一步。
6. 在 Review (审核) 页上，审核您的密钥详细信息，然后选择 Store (存储)。

第 2 步：更新代码

您的代码必须担任 IAM 角色*RoleToRetrieveSecretAtRuntime*才能检索密钥。有关更多信息，请参阅切换到 IAM 角色 (AWS API)。

然后，您可以使用 Secrets Manager 提供的示例代码更新您的代码，以检索 Secrets Manager 中的密钥。

查找示例代码

1. 打开 Secrets Manager 控制台，网址为https://console.aws.amazon.com/secretsmanager/。
2. 在密钥列表页上，选择您的密钥。
3. 向下滚动到 Sample code（示例代码）。选择您的语言，然后复制代码片段。

移除应用程序中的硬编码凭证并粘贴此代码片段。根据代码语言的不同，您可能需要在片段中添加对函数或方法的调用。

使用密钥代替硬编码凭证，测试您的应用程序是否符合预期。

步骤 3：轮换秘密

最后一步是通过轮换密钥来吊销硬编码的凭证。Rotation 是定期更新密钥的过程。轮换密钥时，您会同时更新密钥和数据库中的凭证。Secrets Manager 可以按照您设定的计划自动为您轮换密钥。

设置轮换包括确保 Lambda 轮换函数可以访问 Secrets Manager 和您的数据库。启用自动轮换后，Secrets Manager 会与您的数据库相同的 VPC 中创建 Lambda 轮换函数，以确保它拥有数据库的网络访问权限。Lambda 轮换函数还必须能够调用 Secrets Manager 以更新密钥。我们建议您在 VPC 中创建一个 Secrets Manager 终端节点，这样从 Lambda 到 Secrets Manager 的调用就不会离开基础架构 AWS 。有关说明，请参阅the section called “VPC 端点（AWS PrivateLink）”。

启用轮换

1. 打开 Secrets Manager 控制台，网址为https://console.aws.amazon.com/secretsmanager/。
2. 在密钥列表页上，选择您的密钥。
3. 在 Secret details (密钥详细信息) 页上的 Rotation configuration (轮换配置) 部分中，选择 Edit rotation (编辑轮换)。
4. 在编辑轮换配置对话框中，执行以下操作：
   a. 启用 Automatic rotation（自动轮换）。
   b. 在 Rotation schedule（轮换计划）下，以 UTC 时区格式输入您的计划。
   c. 选择 Rotate immediately when the secret is stored（在存储密钥时立即轮换），以在保存更改时轮换密钥。
   d. 在 Rotation function（轮换函数）下，选择 Create a new Lambda function（创建新的 Lambda 函数），然后为新函数输入一个名称。Secrets Manager 将 "SecretsManager" 添加到您的函数名称的开头。
   e. 对于轮换策略，选择单用户。
   f. 选择保存。

检查密钥是否已轮换

1. 打开 Secrets Manager 控制台，网址为https://console.aws.amazon.com/secretsmanager/。
2. 选择 Secrets（密钥），然后选择该密钥。
3. 在 Secret details（秘密详细信息）页面上，向下滚动并选择 Retrieve secret value（检索秘密值）。

   如果密钥值改变，则说明轮换已经成功。如果密钥值没有更改，则需要轮换问题排查查看轮换功能的 CloudWatch 日志。

测试您的应用程序按照预期那样在使用轮换后的密钥。

后续步骤

从代码中移除硬编码的密钥后，接下来需要注意以下事项：

- 您可以通过缓存密钥来提高性能并降低成本。有关更多信息，请参阅 获取密钥。
- 您可以选择不同的轮换计划。有关更多信息，请参阅 the section called “轮换计划”。
- 要在你的 Java 和 Python 应用程序中查找硬编码的机密，我们建议使用 Amazon CodeGuru Reviewer。

为 AWS Secrets Manager设置交替用户轮换

在本教程中，您将学习如何为包含数据库凭证的秘密设置交替用户轮换。Alternating users rotation（交替用户轮换）是一种轮换策略，在该策略中，Secrets Manager 将克隆用户，然后替换被更新的那些用户凭证。如果您需要为密钥实现高可用性，则此策略是一个不错的选择，因为其中一个交替用户拥有数据库的最新凭证，而另一个则正在更新。有关更多信息，请参阅 the section called “交替用户”。

要设置交替用户轮换，您需要两个秘密：

- 其中一个秘密包含您想轮换的凭证。
- 具有管理员凭证的第二个密钥。

  此用户有权克隆第一个用户并更改第一个用户的密码。在本教程中，您将让 Amazon RDS 为管理员用户创建此密钥。Amazon RDS 还会管理管理员密码轮换。有关更多信息，请参阅 the section called “托管轮换”。

本教程的第一部分内容是介绍如何设置真实环境。为了向您展示轮换的工作原理，本教程使用了一个示例 Amazon RDS MySQL 数据库。为了安全起见，数据库位于限制入站互联网访问的 VPC 中。要通过互联网从本地电脑连接到数据库，请使用堡垒主机，它是 VPC 中可以连接到数据库的服务器，但也允许从互联网进行 SSH 连接。本教程中的堡垒主机是 Amazon EC2 实例，该实例的安全组阻止其他类型的连接。

完成本教程后，我们建议您清理教程中的资源。请勿在生产环境中使用它们。

Secrets Manager 轮换使用 AWS Lambda 函数来更新密钥和数据库。有关使用 Lambda 函数的成本的信息，请参阅 定价。

教程：

- 权限
- 先决条件
- 步骤 1：创建 Amazon RDS 数据库用户
- 步骤 2：为用户凭证创建秘密
- 步骤 3：测试已轮换的秘密
- 步骤 4：清理资源
- 后续步骤

权限

本教程的先决条件为，您需要对 AWS 账户的管理权限。在生产环境中，最佳实践是为每个步骤使用不同的角色。例如，具有数据库管理员权限的角色将创建 Amazon RDS 数据库，而具有网络管理员权限的角色将设置 VPC 和安全组。在执行教程步骤时，我们建议您继续使用相同身份。

有关如何在生产环境中设置权限的信息，请参阅 the section called “身份验证和访问控制”。

先决条件

在此教程中，您需要以下内容：

- 先决条件 A：Amazon VPC
- 先决条件 B：Amazon 实例 EC2
- 先决条件 C：Amazon RDS 数据库和管理员凭证的 Secrets Manager 密钥
- 前提条件 D：允许您的本地计算机连接到实例 EC2

先决条件 A：Amazon VPC

在此步骤中，您将创建一个 VPC，您可以在其中启动 Amazon RDS 数据库和亚马逊 EC2实例。在后续步骤中，您将使用计算机通过互联网连接到堡垒机，然后连接到数据库，因此您需要允许来自 VPC 的流量。为此，Amazon VPC 会将互联网网关连接到 VPC 并在路由表中添加路由，以将发往 VPC 外部的流量发送到互联网网关。

在 VPC 中，您可以创建一个 Secrets Manager 端点和一个 Amazon RDS 端点。在稍后的步骤中设置自动轮换时，Secrets Manager 会在 VPC 内创建 Lambda 轮换函数，以便它可以访问数据库。Lambda 轮换函数还会调用 Secrets Manager 来更新密钥，并调用 Amazon RDS 来获取数据库连

接信息。通过在 VPC 内创建终端节点，您可以确保从 Lambda 函数对 Secrets Manager 和 Amazon RDS 的调用不会离开 AWS 基础设施。相反，这些调用将被路由到 VPC 内的端点。

创建 VPC

1. 打开位于 https://console.aws.amazon.com/vpc/ 的 Amazon VPC 控制台。
2. 选择创建 VPC。
3. 在 Create VPC（创建 VPC）页面上，选择 VPC and more（VPC 等）。
4. 在 Name tag auto-generation（名称标签自动生成）下的 Auto-generate（自动生成）下，输入 **`SecretsManagerTutorial`**。
5. 对于 DNS options（DNS 选项），请同时选择 **`Enable DNS hostnames`** 和 **`Enable DNS resolution`**。
6. 选择创建 VPC。

在 VPC 内创建 Secrets Manager 端点

1. 在 Amazon VPC 控制台的 Endpoints（端点）下，选择 Create Endpoint（创建端点）。
2. 在 Endpoint settings（端点设置）下，为 Name（名称）输入 **`SecretsManagerTutorialEndpoint`**。
3. 在 Services（服务）下，输入 **`secretsmanager`** 以筛选列表，然后在您的 AWS 区域中选择 Secrets Manager 端点。例如，在美国东部（弗吉尼亚北部），选择 `com.amazonaws.us-east-1.secretsmanager`。
4. 对于 VPC，选择 **`vpc**** (SecretsManagerTutorial)`**。
5. 对于 Subnets（子网），选择所有 Availability Zones（可用性区域），然后对于每个区域，选择要包含的 Subnet ID（子网 ID）。
6. 对于 IP address type（IP 地址类型），选择 **`IPv4`**。
7. 对于 Security groups（安全组），选择默认安全组。
8. 对于 Policy（策略），选择 **`Full access`**。
9. 选择创建端点。

在 VPC 内创建 Amazon RDS 端点

1. 在 Amazon VPC 控制台的 Endpoints（端点）下，选择 Create Endpoint（创建端点）。
2. 在 Endpoint settings（端点设置）下，为 Name（名称）输入 **`RDSTutorialEndpoint`**。

3. 在 Services（服务）下，输入 **rds** 以筛选列表，然后在您的 AWS 区域中选择 Amazon RDS 端点。例如，在美国东部（弗吉尼亚北部），选择 `com.amazonaws.us-east-1.rds`。
4. 对于 VPC，选择 **vpc**** (SecretsManagerTutorial)**。
5. 对于 Subnets（子网），选择所有 Availability Zones（可用性区域），然后对于每个区域，选择要包含的 Subnet ID（子网 ID）。
6. 对于 IP address type（IP 地址类型），选择 **IPv4**。
7. 对于 Security groups（安全组），选择默认安全组。
8. 对于 Policy（策略），选择 **Full access**。
9. 选择创建端点。

先决条件 B：Amazon 实例 EC2

您在后续步骤中创建的 Amazon RDS 数据库将位于 VPC 中，因此要访问它，您需要堡垒主机。该堡垒主机也位于 VPC 中，但在稍后步骤中，您将配置一个安全组，以允许本地计算机使用 SSH 连接到堡垒主机。

为堡垒主机创建 EC2 实例

1. 打开亚马逊 EC2 控制台，网址为https://console.aws.amazon.com/ec2/。
2. 选择 Instances（实例），然后选择 Launch Instances（启动实例）。
3. 在 Name and tags（名称和标签）下，对于 Name（名称），输入 **SecretsManagerTutorialInstance**。
4. 在 Application and OS Images（应用程序和操作系统映像）下，保留默认值 **Amazon Linux 2 AMI (HMV) Kernel 5.10**。
5. 在 Instance type（实例类型）下，保留默认值 **t2.micro**。
6. 在 Key pair（密钥对）下，选择 Create key pair（创建密钥对）。

   在 Create key pair（创建密钥对）对话框中，对于 Key pair name（密钥对名称），输入 **SecretsManagerTutorialKeyPair**，然后选择 Create key pair（创建密钥对）。

   此时会自动下载密钥对。
7. 在 Network settings（网络设置）下，选择 Edit（编辑），然后执行以下操作：

   a. 对于 VPC，选择 **vpc-**** SecretsManagerTutorial**。

   b. 对于 Auto-assign Public IP（自动分配公有 IP），选择 **Enable**。

c. 对于 Firewall（防火墙），选择 Select existing security group（选择现有安全组）。

d. 对于 Common security groups（常见安全组），选择 **default**。

8. 选择启动实例。

先决条件 C：Amazon RDS 数据库和管理员凭证的 Secrets Manager 密钥

在此步骤中，您将创建一个 Amazon RDS MySQL 数据库并对其进行配置，以便 Amazon RDS 创建包含管理员凭证的密钥。然后，Amazon RDS 会自动为您管理管理员密钥的轮换。有关更多信息，请参阅 托管轮换。

在创建数据库过程中，请指定您在上一步中创建的堡垒主机。然后，Amazon RDS 会设置安全组，以便数据库和实例能够相互访问。您可向连接到实例的安全组添加规则，以允许您的本地计算机也连接到该实例。

使用包含管理员凭证的 Secrets Manager 密钥创建 Amazon RDS 数据库

1. 在 Amazon RDS 控制台中，选择 Create database（创建数据库）。
2. 在 Engine options（引擎选项）部分，为 Engine type（引擎类型）选择 **MySQL**。
3. 在 Templates（模板）部分，选择 **Free tier**。
4. 在 Settings（设置）部分，执行以下操作：

   a. 对于 DB instance identifier（数据库实例标识符），输入 **SecretsManagerTutorial**。

   b. 在 “凭据设置” 下，选择中的管理主凭证。 AWS Secrets Manager

5. 在 “连接” 部分中，对于 “计算机资源”，选择 “连接到 EC2 计算机资源”，然后在 “EC2 实例” 中选择**SecretsManagerTutorialInstance**。
6. 选择创建数据库。

前提条件 D：允许您的本地计算机连接到实例 EC2

在此步骤中，您将在 Prereq B 中创建的 EC2 实例配置为允许您的本地计算机连接到该实例。为此，您将编辑 Amazon RDS 在“先决条件 C”中添加的安全组，使其包含允许您的计算机的 IP 地址与 SSH 连接的规则。该规则允许本地计算机（通过当前 IP 地址识别）通过 Internet 使用 SSH 连接到堡垒主机。

允许您的本地计算机连接到实 EC2 例

1. 打开亚马逊 EC2 控制台，网址为https://console.aws.amazon.com/ec2/。

2. 在 EC2 实例上 SecretsManagerTutorialInstance，在安全选项卡上的安全组下，选择**sg-*** (ec2-rds-X)**。
3. 在 Input rules（输入规则）下，选择Edit inbound rules（编辑入站规则）。
4. 选择 Add rule（添加规则），然后对该规则执行以下操作：
   a. 对于类型，选择 **SSH**。
   b. 对于 Source type（源类型），选择 **My IP**。

步骤 1：创建 Amazon RDS 数据库用户

首先，您需要一个用户，其凭证将被存储在秘密中。要创建用户，请使用管理员凭证登录 Amazon RDS 数据库。为简单起见，在本教程中，您将创建具有数据库完全权限的用户。在生产环境中，这并不常见，建议您遵循最低权限原则。

要连接到数据库，请使用 MySQL 客户端工具。在本教程中，您将使用基于 GUI 的应用程序 MySQL Workbench。要安装 MySQL Workbench，请参阅下载 MySQL Workbench。

要连接到数据库，请在 MySQL Workbench 中创建连接配置。要进行配置，您需要从 Amazon EC2 和 Amazon RDS 获得一些信息。

在 MySQL Workbench 中创建数据库连接

1. 在 MySQL Workbench 中，选择 MySQL Connections（MySQL 连接）旁边的 (+) 按钮。
2. 在 Setup New Connection（设置新连接）对话框中，执行以下操作：
   a. 对于 Connection Name（连接名称），输入 **SecretsManagerTutorial**。
   b. 对于 Connection Method（连接方法），选择 **Standard TCP/IP over SSH**。
   c. 在 Parameters（参数）选项卡上，执行以下操作：
      i. 对于 SSH 主机名，请输入 Amazon EC2 实例的公有 IP 地址。

         您可以通过选择实例在 Amazon EC2 控制台上找到 IP 地址SecretsManagerTutorialInstance。复制公有 IPv4 DNS 下的 IP 地址。
      ii. 对于 SSH Username（SSH 用户名），输入 **ec2-user**。
      iii. 对于 SSH Keyfile，请选择您在前面的先决条件中下载的密钥对文件 SecretsManagerTutorialKeyPair.pem。
      iv. 对于 MySQL Hostname（MySQL 主机名），输入 Amazon RDS 端点地址。

您可以在 Amazon RDS 控制台上通过选择数据库实例 secretsmanagertutorialdb 查找端点地址。复制 Endpoint（端点）下的地址。

v. 对于 Username（用户名），输入 **admin**。

d. 选择确定。

检索管理员密码

1. 在 Amazon RDS 控制台中，导航到您的数据库。
2. 在 Configuration（配置）选项卡的 Master Credentials ARN（主凭证 ARN）下，选择 Manage in Secrets Manager（在 Secrets Manager 中管理）。

   此时将打开 Secrets Manager 控制台。
3. 在密钥详细信息页面上，选择 Retrieve secret value（检索密钥值）。
4. 密码显示在 Secret value（密钥值）部分中。

创建数据库用户

1. 在 MySQL 工作台中，选择连接SecretsManagerTutorial。
2. 输入从密钥中检索到的管理员密码。
3. 在 MySQL Workbench 中，在 Query（查询）窗口中，输入以下命令（包括强密码），然后选择 Execute（执行）。轮换函数使用 SELECT 测试更新的密钥，因此 **appuser** 必须至少具有该权限。

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

   在 Output（输出）窗口中，您会看到这些命令执行成功。

步骤 2：为用户凭证创建秘密

接下来，您将创建秘密，用于存储您刚创建的用户凭证。这是您将要轮换的秘密。启用自动轮换，要指示交替用户策略，您应选择一个单独的超级用户秘密，它应有权限更改第一个用户的密码。

1. 打开 Secrets Manager 控制台，网址为https://console.aws.amazon.com/secretsmanager/。

2. 选择存储新密钥。
3. 在 Choose secret type（选择密钥类型）页面上，执行以下操作：
   a. 对于 Secret type（秘密类型），选择 Credentials for Amazon RDS database（Amazon RDS 数据库凭证）。
   b. 对于 Credentials（凭证），输入用户名 **appuser**，以及您为使用 MySQL Workbench 创建的数据库用户输入的密码。
   c. 对于 Database（数据库），选择 secretsmanagertutorialdb。
   d. 选择下一步。
4. 在 Configure secret（配置密钥）页面上，对于 Secret name（密钥名称），输入 **SecretsManagerTutorialAppuser**，然后选择 Next（下一步）。
5. 在 Configure rotation（配置轮换）页面上，执行以下操作：
   a. 启用 Automatic rotation（自动轮换）。
   b. 对于 Rotation schedule（轮换计划），设置计划 Days（天数）：**2** 天，以及 Duration（持续时间）：**2h**。使 Rotate immediately（立即轮换）处于已选择状态。
   c. 对于 Rotation function（轮换函数），选择 Create a rotation function（创建轮换函数），然后对于函数名称，输入 **tutorial-alternating-users-rotation**。
   d. 对于轮换策略，选择交替用户，然后在管理员凭证密钥下，选择名为 rds!cluster...，并且描述包含您在本教程 **secretsmanagertutorial** 中所创建数据库的名称的密钥，例如 `Secret associated with primary RDS DB instance: arn:aws:rds:`*`Region`*`:`*`AccountId`*`:db:secretsmanagertutorial`。
   e. 选择下一步。
6. 在 Review（检查）页面上，选择 Store（存储）。

   Secrets Manager 会返回到密钥详细信息页面。您可以在该页面顶部查看轮换配置状态。Secrets CloudFormation Manager 用于创建资源，例如 Lambda 轮换函数和运行 Lambda 函数的执行角色。 CloudFormation 完成后，横幅变为预定轮换的 Secret。第一次轮换已完成。

步骤 3：测试已轮换的秘密

在密钥轮换后，您可以检查该密钥是否包含有效凭证。秘密中的密码已从原始凭证发生更改。

从秘密中检索新密码

1. 打开 Secrets Manager 控制台，网址为https://console.aws.amazon.com/secretsmanager/。

2. 选择 Secrets（秘密），然后选择秘密 **`SecretsManagerTutorialAppuser`**。
3. 在 Secret details（秘密详细信息）页面上，向下滚动并选择 Retrieve secret value（检索秘密值）。
4. 在 Key/value（键/值）表中，为 **`password`** 复制 Secret value（秘密值）。

测试凭证

1. 在 MySQL 工作台中，右键单击该连接 SecretsManagerTutorial，然后选择 “编辑连接”。
2. 在 Manage Server Connections（管理服务器连接）对话框中，对于 Username（用户名），输入 **`appuser`**，然后选择 Close（关闭）。
3. 回到 MySQL 工作台，选择连接SecretsManagerTutorial。
4. 在 Open SSH Connection（打开 SSH 连接）对话框中，对于 Password（密码），粘贴您从秘密中检索到的密码，然后选择 OK（确定）。

   如果凭证有效，则 MySQL Workbench 将打开至数据库的设计页面。

这表明秘密轮换是成功的。秘密中的凭证已更新，它是用于连接到数据库的有效密码。

步骤 4：清理资源

如果您想尝试另一种轮换策略单用户轮换，请跳过清理资源，然后转到 the section called “单用户轮换”。

否则，为了避免可能产生的费用并移除可以访问 Internet 的 EC2 实例，请删除您在本教程中创建的以下资源及其先决条件：

- Amazon RDS 数据库实例。有关说明，请参阅《Amazon RDS 用户指南》中的删除数据库实例。
- 亚马逊 EC2 实例。有关说明，请参阅 Amazon EC2 用户指南中的终止实例。
- Secrets Manager 秘密 `SecretsManagerTutorialAppuser`。有关说明，请参阅the section called “删除密钥”。
- Secrets Manager 端点。有关说明，请参阅《AWS PrivateLink 指南》中的删除 VPC 端点。
- VPC 端点。有关说明，请参阅《AWS PrivateLink 指南》中的删除 VPC。

后续步骤

- 了解如何在您的应用程序中检索密钥。

- 了解其他轮换计划。

为 AWS Secrets Manager设置单用户轮换

在本教程中，您将学习如何为包含数据库凭证的密钥设置单用户轮换。单用户轮换是一种轮换策略，在该策略中，Secrets Manager 将同时在密钥和数据库中更新用户的凭证。有关更多信息，请参阅 the section called “单用户”。

完成本教程后，我们建议您清理教程中的资源。请勿在生产环境中使用它们。

Secrets Manager 轮换使用 AWS Lambda 函数来更新密钥和数据库。有关使用 Lambda 函数的成本的信息，请参阅 定价。

目录

- 权限
- 先决条件
- 步骤 1：创建 Amazon RDS 数据库用户
- 步骤 2：为数据库用户凭证创建密钥
- 步骤 3：测试轮换的密码
- 步骤 4：清理资源
- 后续步骤

权限

本教程的先决条件为，您需要对 AWS 账户的管理权限。在生产环境中，最佳实践是为每个步骤使用不同的角色。例如，具有数据库管理员权限的角色将创建 Amazon RDS 数据库，而具有网络管理员权限的角色将设置 VPC 和安全组。在执行教程步骤时，我们建议您继续使用相同身份。

有关如何在生产环境中设置权限的信息，请参阅 the section called “身份验证和访问控制”。

先决条件

本教程的先决条件是 the section called “交替用户轮换”。在第一个教程结束时，请不要清理资源。在该教程之后，您将拥有一个现实环境，其中包含一个 Amazon RDS 数据库和一个内含数据库管理员凭证的 Secrets Manager 密钥。您还有另一个密钥包含数据库用户的凭证，但您在本教程中不使用该密钥。

您还在 MySQL Workbench 中配置了一条连接，可以使用管理员凭证连接到数据库。

步骤 1：创建 Amazon RDS 数据库用户

首先，您需要一个用户，其凭证将被存储在秘密中。要创建用户，请使用存储在密钥中的管理员凭证登录 Amazon RDS 数据库。为简单起见，在本教程中，您将创建具有数据库完全权限的用户。在生产环境中，这并不常见，建议您遵循最低权限原则。

检索管理员密码

1. 在 Amazon RDS 控制台中，导航到您的数据库。
2. 在 Configuration（配置）选项卡的 Master Credentials ARN（主凭证 ARN）下，选择 Manage in Secrets Manager（在 Secrets Manager 中管理）。

   此时将打开 Secrets Manager 控制台。
3. 在密钥详细信息页面上，选择 Retrieve secret value（检索密钥值）。
4. 密码显示在 Secret value（密钥值）部分中。

创建数据库用户

1. 在 MySQL Workbench 中，右键单击该连接，SecretsManagerTutorial然后选择 “编辑连接”。
2. 在 Manage Server Connections（管理服务器连接）对话框中，对于 Username（用户名），输入 **admin**，然后选择 Close（关闭）。
3. 回到 MySQL 工作台，选择连接SecretsManagerTutorial。
4. 输入从密钥中检索到的管理员密码。
5. 在 MySQL Workbench 中，在 Query（查询）窗口中，输入以下命令（包括强密码），然后选择 Execute（执行）。轮换函数使用 SELECT 测试更新的密钥，因此 **dbuser** 必须至少具有该权限。

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

   在 Output（输出）窗口中，您会看到这些命令执行成功。

步骤 2：为数据库用户凭证创建密钥

接下来，您将创建一个密钥用于存储您刚创建的用户的凭证，并且将启用自动轮换（包括立即轮换）。Secrets Manager 轮换密钥，这意味着密码是以编程方式生成的，没有人看到这个新密码。立即开始轮换也可以帮助您确定轮换设置是否正确。

1. 打开 Secrets Manager 控制台，网址为https://console.aws.amazon.com/secretsmanager/。
2. 选择存储新密钥。
3. 在 Choose secret type（选择密钥类型）页面上，执行以下操作：
   a. 对于 Secret type（秘密类型），选择 Credentials for Amazon RDS database（Amazon RDS 数据库凭证）。
   b. 对于 Credentials（凭证），输入用户名 **dbuser**，以及您为使用 MySQL Workbench 创建的数据库用户输入的密码。
   c. 对于 Database（数据库），选择 secretsmanagertutorialdb。
   d. 选择下一步。
4. 在 Configure secret（配置密钥）页面上，对于 Secret name（密钥名称），输入 **SecretsManagerTutorialDbuser**，然后选择 Next（下一步）。
5. 在 Configure rotation（配置轮换）页面上，执行以下操作：
   a. 启用 Automatic rotation（自动轮换）。
   b. 对于 Rotation schedule（轮换计划），设置计划 Days（天数）：**2** 天，以及 Duration（持续时间）：**2h**。使 Rotate immediately（立即轮换）处于已选择状态。
   c. 对于 Rotation function（轮换函数），选择 Create a rotation function（创建轮换函数），然后对于函数名称，输入 **tutorial-single-user-rotation**。
   d. 对于轮换策略，选择单用户。
   e. 选择下一步。
6. 在 Review（检查）页面上，选择 Store（存储）。

   Secrets Manager 会返回到密钥详细信息页面。您可以在该页面顶部查看轮换配置状态。Secrets CloudFormation Manager 用于创建资源，例如 Lambda 轮换函数和运行 Lambda 函数的执行角色。 CloudFormation 完成后，横幅变为预定轮换的 Secret。第一次轮换已完成。

步骤 3：测试轮换的密码

在第一次密钥轮换（可能需要几秒钟）之后，您可以检查秘密是否仍包含有效凭证。秘密中的密码已从原始凭证发生更改。

从秘密中检索新密码

1. 打开 Secrets Manager 控制台，网址为https://console.aws.amazon.com/secretsmanager/。
2. 选择 Secrets（秘密），然后选择秘密 **SecretsManagerTutorialDbuser**。
3. 在 Secret details（秘密详细信息）页面上，向下滚动并选择 Retrieve secret value（检索秘密值）。
4. 在 Key/value（键/值）表中，为 **password** 复制 Secret value（秘密值）。

测试凭证

1. 在 MySQL Workbench 中，右键单击该连接，SecretsManagerTutorial然后选择 “编辑连接”。
2. 在 Manage Server Connections（管理服务器连接）对话框中，对于 Username（用户名），输入 **dbuser**，然后选择 Close（关闭）。
3. 回到 MySQL 工作台，选择连接SecretsManagerTutorial。
4. 在 Open SSH Connection（打开 SSH 连接）对话框中，对于 Password（密码），粘贴您从秘密中检索到的密码，然后选择 OK（确定）。

   如果凭证有效，则 MySQL Workbench 将打开至数据库的设计页面。

步骤 4：清理资源

为避免潜在费用，请删除您在本教程中创建的秘密。有关说明，请参阅the section called “删除密钥”。

要清理前面教程中创建的资源，请参阅 the section called “步骤 4：清理资源”。

后续步骤

- 了解如何在您的应用程序中检索秘密。请参阅获取密钥。
- 了解其他轮换计划。请参阅the section called “轮换计划”。

创建密 AWS Secrets Manager 钥

秘密可以是密码、一组凭证（例如，用户名和密码）、 OAuth 令牌或以加密形式存储在 Secrets Manager 中的其他秘密信息。

> **Tip**
>
> 对于 Amazon RDS 和 Amazon Redshift 管理员用户凭证，我们建议您使用托管密钥。您可以通过管理服务创建托管密钥，然后可以使用托管轮换。

当使用控制台存储复制到其他区域的源数据库的数据库凭证时，密钥将包含源数据库的连接信息。然后复制密钥时，副本将是源密钥的副本，并且包含相同的连接信息。您可以在密钥中添加其他键值对以记录区域连接信息。

要创建密钥，您需要SecretsManagerReadWrite 托管策略授予权限。

当您创建密钥时，Secrets Manager 会生成 CloudTrail 日志条目。有关更多信息，请参阅 the section called “使用登录 AWS CloudTrail ”。

创建密钥（控制台）

1. 打开 Secrets Manager 控制台，网址为https://console.aws.amazon.com/secretsmanager/。
2. 选择存储新密钥。
3. 在 Choose secret type（选择密钥类型）页面上，执行以下操作：

   a. 对于 Secret type（密钥类型），执行以下操作之一：

      - 要存储数据库凭证，请选择要存储的数据库凭证类型。然后选择数据库，并输入凭证。
      - 要存储非用于数据库的 API 密钥、访问令牌和凭证，请选择其他类型的密钥。

        在键值对中，请以 JSON 键值对格式输入密钥，或者选择明文选项卡，然后以任何格式输入密钥。您可以在密钥中存储最多 65536 个字节。一些示例：

        API key

        输入键/值对：

        ClientID：*my_client_id*

ClientSecret：*wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY*

OAuth token

输入明文：

AKIAI44QH8DHBEXAMPLE

Digital certificate

输入明文：

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

Private key

输入明文：

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

b. 对于 Encryption key（加密密钥） AWS KMS key ，选择 Secrets Manager 用于加密密钥值的。有关更多信息，请参阅 密钥加密和解密。

- 在大多数情况下，请选择 aws/secretsmanager 以将 AWS 托管式密钥 用于 Secrets Manager。使用此密钥不产生任何费用。
- 如果您需要从另一个访问秘密 AWS 账户，或者如果您想使用自己的 KMS 密钥，以便轮换密钥或对其应用密钥策略，请从列表中选择客户托管密钥，或者选择 Add New Key（添加新密钥）以创建密钥。有关使用客户托管密钥的成本的信息，请参阅 定价。

  您必须具有 the section called “KMS 密钥的权限”。有关跨账户访问的更多信息，请参阅the section called “跨账户访问”。

c. 选择下一步。

4. 在 Configure secret（配置密钥）页面上，执行以下操作：

   a. 输入一个描述性的 Secret name（密钥名称）和 Description（说明）。密钥名称可以包含 1-512 个字母数字和 /_+=.@- 字符。

b. （可选）在标签部分中，在您的密钥中添加一个或多个标签。有关标记策略，请参阅 the section called “标记 密钥”。请不要将敏感信息存储在标签中，因为它们未加密。

c. （可选）在资源权限，要将资源策略添加到您的密钥中，请选择编辑权限。有关更多信息，请参阅 the section called “基于资源的策略”。

d. （可选）在 Replicate sec ret（复制密钥）中，要将密钥复制到另一个 AWS 区域中，请选择复制密钥 您可以现在复制密钥，也可以回头再复制。有关更多信息，请参阅 多区域复制。

e. 选择下一步。

5. （可选）在 Configure rotation（配置轮换）页面上，您可以启用自动轮换。您也可以现在保持关闭轮换，然后稍后将其打开。有关更多信息，请参阅 轮换 密钥。选择下一步。

6. 在 Review (审核) 页上，审核您的密钥详细信息，然后选择 Store (存储)。

   Secrets Manager 将返回到密钥列表。如果您的新密钥未显示，请选择 Refresh（刷新）按钮。

AWS CLI

当您在命令 shell 中输入命令时，存在访问命令历史记录或实用程序可以访问您命令参数的风险。请参阅the section called “降低使用存储 AWS Secrets Manager 密钥 AWS CLI 的风险”。

Example 根据 JSON 文件中的数据库凭证创建密钥

以下 `create-secret` 示例将根据文件中的凭证创建密钥。有关更多信息，请参阅《 AWS CLI 用户指南》中的从文件加载 AWS CLI 参数。

要使 Secrets Manager 能够轮换密钥，您必须确保 JSON 符合密钥的 JSON 结构。

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

mycreds.json 的内容：

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

```
}
```

Example 创建密钥

以下 `create-secret` 示例将创建包含两个键值对的密钥。

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

Example 创建密钥

以下`create-secret`示例创建了一个带有两个标签的密钥。

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

AWS SDK

要使用其中一个来创建密钥 AWS SDKs，请使用`CreateSecret`操作。有关更多信息，请参阅 the section called “AWS SDKs”。

Secrets Manager 密钥中有什么？

在 Secrets Manager 中，密钥由密钥信息、密钥值和密钥元数据组成。密钥值可以是字符串或二进制值。

为了将多个字符串值存储在一个密钥中，我们建议您使用带有键值对的 JSON 文本字符串，例如：

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

```
  "dbname"     : "MyDatabase",
  "engine"     : "mysql"
}
```

对于数据库密钥，如果要启用自动轮换，密钥必须包含正确的 JSON 结构的数据库连接信息。有关更多信息，请参阅 the section called “密钥的 JSON 结构”。

Metadata

密钥元数据包括：

- 具有以下格式的 Amazon Resource Name (ARN)。

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

  Secrets Manager 会在密钥名称末尾添加六个随机字符，以帮助确保密钥 ARN 的唯一性。如果删除了原始密钥，然后使用相同的名称创建了新密钥，则 ARNs 由于这些字符，这两个密钥会有所不同。有权访问旧密钥的用户不会自动获得对新密钥的访问权限，因为两 ARNs 者不同。
- 密钥的名称、说明、资源策略和标签。
- 加密密钥的 ARN，Secrets Manager 使用它来加密和解密密钥值。 AWS KMS key Secrets Manager 始终以加密形式存储密钥文本，并在传输过程中加密密钥。请参阅the section called “密钥加密和解密”。
- 如果设置了轮转，有关如何轮转密钥的信息。请参阅轮换 密钥。

Secrets Manager 使用 IAM 权限策略来确保只有授权用户可以访问或修改密钥。请参阅的身份验证和访问控制 AWS Secrets Manager。

密钥有包含加密密钥值副本的版本。更改密钥值或轮换密钥时，Secrets Manager 会创建一个新版本。请参阅the section called “密钥版本”。

您可以通过复制多个密钥 AWS 区域 来使用该密钥。复制密钥时，您可以创建原始或主密钥称为副本密钥。副本密钥保持链接到主密钥上。请参阅 多区域复制。

请参阅 管理密钥。

密钥版本

密钥有包含加密密钥值副本的版本。更改密钥值或轮换密钥时，Secrets Manager 会创建一个新版本。

Secrets Manager 不会存储带有版本的线性密钥历史记录。而是通过标记来跟踪这三个特定版本：

- 当前版本 — `AWSCURRENT`
- 之前的版本—— `AWSPREVIOUS`
- 待定版本（轮换期间）— `AWSPENDING`

密钥始终有一个标记为 `AWSCURRENT` 的版本，Secrets Manager 会在您检索密钥值时默认返回该版本。

您也可以通过调`update-secret-version-stage`用自己的标签来为版本添加标签 AWS CLI。您最多可以为一个密钥附加 20 个版本标签。密钥的两个版本不能具有相同的暂存标注。版本可以有多个标签。

Secrets Manager 从不移除带标签的版本，但未标记的版本将被视为已弃用。如果版本超过 100 个，Secrets Manager 会移除已弃用的版本。Secrets Manager 不会移除 24 小时前创建的版本。

下图显示了一个 AWS 标有版本和客户标签版本的密钥。无标签的版本将被视为已弃用，Secrets Manager 将在某个未来的时间将其移除。

AWS Secrets Manager 密钥的 JSON 结构

您可以在 Secrets Manager 密钥中存储任何文本或二进制文件，最大大小为 65,536 字节。

如果使用 the section called “通过 Lambda 函数进行轮换”，则密钥必须包含轮换函数所需的特定 JSON 字段。例如，对于包含数据库凭证的密钥，轮换函数会连接到数据库以更新凭证，因此该密钥必须包含数据库连接信息。

如果使用控制台编辑数据库密钥的轮换，则该密钥必须包含标识数据库的特定 JSON 键值对。Secrets Manager 使用这些字段查询数据库，以查找存储轮换函数的正确 VPC。

JSON 键名称区分大小写。

主题

- Amazon RDS 和 Aurora 凭证
- Amazon Redshift 凭证
- Amazon Redshift Serverless 凭证

- Amazon DocumentDB 凭证
- Amazon Timestream for InfluxDB 密钥结构
- Amazon ElastiCache 凭证
- Active Directory 凭证

Amazon RDS 和 Aurora 凭证

要使用 Secrets Manager 提供的轮换函数模板，请使用以下 JSON 结构。您可以添加更多键/值对，例如包含其他区域中的副本数据库的连接信息。

DB2

对于 Amazon RDS Db2 实例，由于用户无法更改自己的密码，因此您必须在单独的秘密中提供管理员凭证。

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

MariaDB

```
{
  "engine": "mariadb",
  "host": "<instance host name/resolvable DNS name>",
  "username": "<username>",
  "password": "<password>",
  "dbname": "<database name. If not specified, defaults to None>",
  "port": <TCP port number. If not specified, defaults to 3306>,
  "masterarn": "<optional: ARN of the elevated secret. Required for the the section
 called “####”.>",
```

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

MySQL

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

Oracle

```
{
  "engine": "oracle",
  "host": "<instance host name/resolvable DNS name>",
  "username": "<username>",
  "password": "<password>",
  "dbname": "<database name>",
  "port": <TCP port number. If not specified, defaults to 1521>,
  "masterarn": "<optional: ARN of the elevated secret. Required for the the section
 called “####”.>",
  "dbInstanceIdentifier": <optional: ID of the instance. Alternately, use
 dbClusterIdentifier. Required for configuring rotation in the console.>",
  "dbClusterIdentifier": <optional: ID of the cluster. Alternately, use
 dbInstanceIdentifier. Required for configuring rotation in the console.>"
}
```

Postgres

```
{
```

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

SQLServer

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

Amazon Redshift 凭证

要使用 Secrets Manager 提供的轮换函数模板，请使用以下 JSON 结构。您可以添加更多键/值对，例如包含其他区域中的副本数据库的连接信息。

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

```
  "dbClusterIdentifier": "<optional: database ID. Required for configuring rotation in
 the console.>"
  "port": <optional: TCP port number. If not specified, defaults to 5439>
  "masterarn": "<optional: ARN of the elevated secret. Required for the the section
 called “####”.>"
}
```

Amazon Redshift Serverless 凭证

要使用 Secrets Manager 提供的轮换函数模板，请使用以下 JSON 结构。您可以添加更多键/值对，例如包含其他区域中的副本数据库的连接信息。

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

Amazon DocumentDB 凭证

要使用 Secrets Manager 提供的轮换函数模板，请使用以下 JSON 结构。您可以添加更多键/值对，例如包含其他区域中的副本数据库的连接信息。

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

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

Amazon Timestream for InfluxDB 密钥结构

要轮换 Timestream 密钥，您可以使用 the section called “Amazon Timestream for InfluxDB” 轮换模板。

有关更多信息，请参阅《Amazon Timestream 开发人员指南》中的 Amazon Timestream for InfluxDB 如何使用密钥。

Timestream 密钥必须采用正确的 JSON 结构才能使用轮换模板。有关更多信息，请参阅《Amazon Timestream 开发人员指南》中的密钥的内容。

Amazon ElastiCache 凭证

以下示例显示了存储 ElastiCache 凭证的密钥的 JSON 结构。

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

有关更多信息，请参阅 Amazon 用户指南中的自动轮换 ElastiCache 用户密码。

Active Directory 凭证

AWS Directory Service 使用密钥存储活动目录凭证。有关更多信息，请参阅《AWS Directory Service 管理指南》中的将 Amazon EC2 Linux 实例无缝加入您的托管 AD Active Directory。无缝加入域名需要以下示例中的键名称。如果不使用无缝域加入，则可以使用环境变量更改密钥中键的名称，如轮换函数模板代码中所述。

要轮换 Active Directory 密钥，您可以使用 Active Directory 轮换模板。

Active Directory credential

```
{
  "awsSeamlessDomainUsername": "<username>",
```

```
  "awsSeamlessDomainPassword": "<password>"
}
```

如果要轮换密钥，请包括域目录 ID。

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

如果将该密钥与包含密钥表的密钥一起使用，则将该密钥表密钥包括在内。 ARNs

```
{
  "awsSeamlessDomainDirectoryId": "d-12345abc6e",
  "awsSeamlessDomainUsername": "<username>",
  "awsSeamlessDomainPassword": "<password>",
  "directoryServiceSecretVersion": 1,
  "schemaVersion": "1.0",
  "keytabArns": [
    "<ARN of child keytab secret 1>,
    "<ARN of child keytab secret 2>,
    "<ARN of child keytab secret 3>,
  ],
  "lastModifiedDateTime": "2021-07-19 17:06:58"
}
```

Active Directory keytab

有关使用密钥表文件对亚马逊上的活动目录账户进行身份验证的信息 EC2，请参阅在 Amazon Linux 2 上使用 SQL Server 2017 部署和配置活动目录身份验证。

```
{
  "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
}
```

使用管理密钥 AWS Secrets Manager

主题

- 更新 AWS Secrets Manager 密钥的值
- 使用 Secrets Manager 生成密码
- 将密钥回滚到以前的版本
- 更改密钥的加密 AWS Secrets Manager 密钥
- 修改密 AWS Secrets Manager 钥
- 在里面寻找秘密 AWS Secrets Manager
- 删除密 AWS Secrets Manager 钥
- 恢复密 AWS Secrets Manager 钥
- 在中标记机密 AWS Secrets Manager

更新 AWS Secrets Manager 密钥的值

要更新您的秘密值，可以使用控制台、CLI 或 SDK。当您更新秘密值时，Secrets Manager 会使用暂存标签 `AWSCURRENT` 创建秘密的新版本。您仍然可以访问带有标签 `AWSPREVIOUS` 的旧版本。您也可以添加自己的标签。有关更多信息，请参阅 Secrets Manager 版本控制。

更新秘密值（控制台）

1. 打开 Secrets Manager 控制台，网址为https://console.aws.amazon.com/secretsmanager/。
2. 从密钥列表上，选择您的密钥。
3. 进入密钥详细信息页面后，在概述选项卡中的密钥值部分，选择检索秘密值，然后选择编辑。

AWS CLI

更新秘密值 (AWS CLI)

- 当您在命令 shell 中输入命令时，存在访问命令历史记录或实用程序可以访问您命令参数的风险。请参阅 the section called “降低使用存储 AWS Secrets Manager 密钥 AWS CLI 的风险”。

  以下 `put-secret-value` 将创建包含两个键值对的新版本密钥。

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

以下 put-secret-value 创建了一个带有自定义暂存标签的新版本。新版本将带有标签 MyLabel 和 AWSCURRENT。

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

AWS SDK

我们建议您避免以超过每 10 分钟一次的速率持续调用 PutSecretValue 或 UpdateSecret。如果调用 PutSecretValue 或 UpdateSecret 更新密钥值，Secrets Manager 将创建密钥的新版本。当版本超过 100 个时，Secrets Manager 会删除未标记的版本，但不会删除 24 小时内创建的版本。如果每 10 分钟更新一次密钥值，则创建的版本多于 Secrets Manager 删除的版本，将达到密钥版本的配额。

要更新秘密值，请使用以下操作：UpdateSecret 或 PutSecretValue。有关更多信息，请参阅 the section called “AWS SDKs”。

使用 Secrets Manager 生成密码

使用 Secrets Manager 的一种常见模式是在 Secrets Manager 中生成密码，然后在数据库或服务中使用该密码。您可以使用以下方法执行此操作：

- AWS CloudFormation — 请参阅AWS CloudFormation。
- AWS CLI — 请参阅get-random-password。
- AWS SDKs — 请参阅GetRandomPassword。

将密钥回滚到以前的版本

您可以通过使用 AWS CLI移动附加到密钥版本的标签，将密钥还原为先前版本。有关 Secrets Manager 如何存储密钥版本的信息，请参阅 the section called “密钥版本”。

以下update-secret-version-stage示例将 AWSCURRENT 暂存标签移动到密钥的先前版本，这样做会将密钥还原为先前的版本。要查找以前版本的 ID，请使用 list-secret-version-ids 或在 Secrets Manager 控制台中查看版本。

对于此示例，带有标签的版本为 a1b2c3d4-5678-90ab-cdef-带 AWSCURRENT 标签的版本为 a1b2c3d4-5678-90ab-cdef-。EXAMPLE11111 AWSPREVIOUS EXAMPLE22222在此示例中，您将 AWSCURRENT 标签从版本 11111 移动到 22222。由于 AWSCURRENT 标签已从版本中移除，因此update-secret-version-stage会自动将 AWSPREVIOUS 标签移动刀该版本（11111）。结果是交换 AWSPREVIOUS 了 AWSCURRENT 和。

```
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
```

更改密钥的加密 AWS Secrets Manager 密钥

Secrets Manager 使用带有 AWS KMS 密钥和数据密钥的信封加密来保护每个密钥值。对于每个秘密，您可以选择要使用的 KMS 密钥。您可以使用客户管理的密钥 AWS 托管式密钥 aws/secretsmanager，也可以使用客户管理的密钥。在大多数情况下，我们建议使用aws/secretsmanager，而且使用它不收取任何费用。如果您需要从其他人访问密钥 AWS 账户，或者您想使用自己的 KMS 密钥以便轮换密钥或对其应用密钥策略，请使用 客户托管式密钥。您必须具有 the section called “KMS 密钥的权限”。有关使用客户托管密钥的成本的信息，请参阅 定价。

您可以更改秘密的加密密钥。例如，如果您想从其他账户访问密钥，并且该密钥当前已使用 AWS 托管密钥进行加密aws/secretsmanager，则可以切换到 客户托管式密钥。

> **Tip**
>
> 如果您想轮换 客户托管式密钥，我们建议您使用 AWS KMS 自动密钥轮换。有关更多信息，请参阅轮换 AWS KMS 密钥。

当您更改加密密钥时，Secrets Manager 会使用新密钥重新加密 AWSCURRENT、AWSPENDING 和 AWSPREVIOUS 版本。为了避免将您锁定在密钥之外，Secrets Manager 会使用以前的密钥加密所有现有版本。这意味着您可以使用以前的密钥或新密钥解密 AWSCURRENT、AWSPENDING 和 AWSPREVIOUS 版本。如果没有先前密钥的 kms:Decrypt 权限，则当您更改加密密钥时，Secrets Manager 无法解密密钥版本来重新加密它们。在这种情况下，现有版本不会被重新加密。

为了使只能通过新加密密钥解密 AWSCURRENT，请使用新密钥创建新版本的密钥。然后，为了能够解密 AWSCURRENT 密钥版本，您必须拥有新密钥的权限。

如果停用以前的加密密钥，则除了 AWSCURRENT、AWSPENDING 和 AWSPREVIOUS 之外，您将无法解密任何秘密版本。如果想要保留对其他标有标签的秘密版本的访问权限，则需要使用 the section called "AWS CLI" 通过新的加密密钥重新创建这些版本。

更改秘密的加密密钥（控制台）

1. 打开 Secrets Manager 控制台，网址为https://console.aws.amazon.com/secretsmanager/。
2. 从密钥列表上，选择您的密钥。
3. 在秘密详细信息页面上的秘密详细信息部分中，选择操作，然后选择编辑加密密钥。

AWS CLI

如果更改秘密的加密密钥，然后停用了以前的加密密钥，则除了 AWSCURRENT、AWSPENDING 和 AWSPREVIOUS 之外，您将无法解密任何秘密版本。如果想要保留对其他标有标签的秘密版本的访问权限，则需要使用 the section called "AWS CLI" 通过新的加密密钥重新创建这些版本。

更改秘密的加密密钥 (AWS CLI)

1. 以下 `update-secret` 示例将更新用于加密密钥值的 KMS 密钥。该 KMS 密钥必须与加密密钥位于同一区域中。

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

2. （可选）如果您的秘密版本带有自定义标签，则要使用新密钥对其重新加密，则必须重新创建这些版本。

   当您在命令 shell 中输入命令时，存在访问命令历史记录或实用程序可以访问您命令参数的风险。请参阅the section called "降低使用存储 AWS Secrets Manager 密钥 AWS CLI 的风险"。

   a. 获取秘密版本的值。

      ```
      aws secretsmanager get-secret-value \
            --secret-id MyTestSecret \
      ```

```
    --version-stage MyCustomLabel
```

记下秘密值。

b. 创建具有该值的新版本。

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

修改密 AWS Secrets Manager 钥

您可以在创建密钥后修改其元数据，具体取决于密钥的创建者。对于由其他服务创建的密钥，您可能需要使用其他服务来更新或轮换它。

要确定谁管理密钥，您可以查看密钥名称。由其他服务管理的密钥以该服务的 ID 作为前缀。或者，在中 AWS CLI，调用 d escribe-secret，然后查看该字段。`OwningService`有关更多信息，请参阅 由其他服务管理的密钥。

对于您管理的密钥，您可以修改密钥的描述、基于资源的策略、加密密钥和标记。您还可以更改加密密钥值信息，但我们建议您轮换更新包含凭证的密钥值。轮换会更新 Secrets Manager 中的密钥以及数据库或服务上的凭据。这会自动保持同步这些密钥，以便在客户端请求密钥值时，始终检索一组正常工作的凭证。有关更多信息，请参阅 轮换 密钥。

当你修改密钥时，Secrets Manager 会生成一个 CloudTrail 日志条目。有关更多信息，请参阅 the section called “使用登录 AWS CloudTrail ”。

更新您管理的密钥（控制台）

1. 打开 Secrets Manager 控制台，网址为https://console.aws.amazon.com/secretsmanager/。
2. 从密钥列表上，选择您的密钥。
3. 在密钥详细信息页面上，执行以下操作之一：

   请注意，您无法更改 ARN 或密钥的名称。

   - 要更新描述，在密钥详细信息部分中，选择操作，然后选择编辑描述。
   - 要更新加密密钥，请参阅 the section called “更改秘密的加密密钥”。
   - 要更新标签，请在标签选项卡中，选择编辑标签。请参阅 the section called “标记 密钥”。

- 要更新秘密值，请参阅 the section called “更新秘密值”。
- 要更新密钥的权限，请在概述选项卡中选择编辑权限。请参阅 the section called “基于资源的策略”。
- 要更新密钥的轮换，请在轮换选项中选择编辑轮换。请参阅 轮换 密钥。
- 要将您的密钥复制到其他区域，请参阅 多区域复制。
- 如果您的密钥有副本，则您可以更改副本的加密密钥。在复制选项卡中，选择副本对应的单选按钮，然后在操作菜单中，选择编辑加密密钥。请参阅 the section called “密钥加密和解密”。
- 若要更改密钥以使其由其他服务管理，您需要在该服务中重新创建密钥。请参阅 由其他服务管理的密钥。

AWS CLI

Example 更新密钥说明

以下 `update-secret` 示例将更新密钥的描述。

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

AWS SDK

我们建议您避免以超过每 10 分钟一次的速率持续调用 `PutSecretValue` 或 `UpdateSecret`。如果调用 `PutSecretValue` 或 `UpdateSecret` 更新密钥值，Secrets Manager 将创建密钥的新版本。当版本超过 100 个时，Secrets Manager 会删除未标记的版本，但不会删除 24 小时内创建的版本。如果每 10 分钟更新一次密钥值，则创建的版本多于 Secrets Manager 删除的版本，将达到密钥版本的配额。

要更新秘密，请使用以下操作：`UpdateSecret` 或 `ReplicateSecretToRegions`。有关更多信息，请参阅 the section called “AWS SDKs”。

在里面寻找秘密 AWS Secrets Manager

如果搜索密钥时未设置筛选条件，Secrets Manager 会匹配密钥名称、描述、标签键和标签值中的关键字。未设置筛选条件的搜索不区分大小写，忽略空格、/、_、=、# 等特殊字符，并且仅使用数字和

字母进行搜索。在不使用筛选条件的情况下进行搜索时，Secrets Manager 会分析搜索字符串以将其转换为单独的单词。通过从大写到小写、从字母到数字或从 number/letter 标点符号到标点符号的任何变化来分隔单词。例如，输入搜索词 `credsDatabase#892` 将会搜索名称、描述和标签键和值中的 `creds`、`Database` 和 892。

当你列出密钥时，Secrets Manager 会生成一个 CloudTrail 日志条目。有关更多信息，请参阅 the section called “使用登录 AWS CloudTrail ”。

Secrets Manager 是一种区域服务，仅返回选定区域内的密钥。

搜索筛选条件

如果不使用任何筛选条件，Secrets Manager 会将搜索字符串分成单词，然后在所有属性中搜索匹配项。此搜索不区分大小写。例如，搜索 **`My_Secret`** 会匹配名称、描述或标签中带有 my 或 secret 字样的密钥。

您可以将以下筛选条件应用到搜索：

名称

匹配密钥名称的开头；区分大小写。例如，名称：**`Data`** 会返回名为 `DatabaseSecret` 的密钥，不返回名为 `databaseSecret` 或 `MyData` 的密钥。

描述

匹配密钥描述中的单词，不区分大小写。例如，描述：**`My Description`** 会匹配具有以下描述的密钥：

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

管理者

查找由外部服务管理的机密 AWS，例如：

- 1密码
- Akeyless
- CyberArk

- HashiCorp

拥有服务

匹配管理服务 ID 前缀的开头，不区分大小写。例如，**my-ser** 将服务管理的密钥与前缀 `my-serv` 和 `my-service` 进行匹配。有关更多信息，请参阅 由其他服务管理的密钥。

已复制的密钥

您可以筛选主密钥、副本密钥或未复制的密钥。

标签键

匹配标签键的开头；区分大小写。例如，标签键：**Prod** 会返回带标签 `Production` 和 `Prod1` 的密钥，不返回带标签 `prod` 或 `1 Prod` 的密钥。

标签值

匹配标签值的开头；区分大小写。例如，标签值：**Prod** 会返回带标签 `Production` 和 `Prod1` 的密钥，不返回带标签值 `prod` 或 `1 Prod` 的密钥。

AWS CLI

Example 列出您账户中的密钥

以下 `list-secrets` 示例获取了您账户中的密钥列表。

```
aws secretsmanager list-secrets
```

Example 筛选您账户中的密钥列表

以下 `list-secrets` 示例将获取您的账户中名称包含 Test 的密钥列表。按名称筛选区分大小写。

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

Example 查找由其他 AWS 服务管理的机密

以下 `list-secrets` 示例获取服务管理的密钥列表。按 ID 指定服务。有关更多信息，请参阅 由其他服务管理的密钥。

```
aws secretsmanager list-secrets \
```

```
    --filters Key="owning-service",Values="<service ID prefix>"
```

AWS SDK

要使用其中一个来查找机密 AWS SDKs，请使用`ListSecrets`。有关更多信息，请参阅 the section called “AWS SDKs”。

删除密 AWS Secrets Manager 钥

由于机密的关键性， AWS Secrets Manager 故意使删除机密变得困难。Secrets Manager 不会立即删除密钥。而是Secrets Manager 会立即使这些密钥无法访问，并计划在恢复时段（最少 为 7 天）后删除这些密钥。在恢复时段结束后，您才能恢复以前删除的密钥。标记为已删除的密钥不收取任何费用。

如果已将主密钥复制到其他区域，则无法将其删除。首先删除副本，然后删除主密钥。在您删除副本时，该副本会被立即删除。

您无法直接删除某个密钥版本，相反，您可以使用 AWS CLI 或 AWS SDK 从版本中移除所有暂存标签。这会将该版本标记为已弃用，并允许 Secrets Manager 在后台自动删除该版本。

如果您不知道应用程序是否仍在使用密钥，则可以创建一个 Amazon CloudWatch 警报，提醒您在恢复时段内有人尝试访问密钥。有关更多信息，请参阅 监控计划删除的 AWS Secrets Manager 密钥何时被访问。

要删除密钥，您必须具有 `secretsmanager:ListSecrets` 和 `secretsmanager:DeleteSecret` 权限。

当您删除密钥时，Secrets Manager 会生成一个 CloudTrail 日志条目。有关更多信息，请参阅 the section called “使用登录 AWS CloudTrail ”。

删除密钥 (控制台)

1. 打开 Secrets Manager 控制台，网址为https://console.aws.amazon.com/secretsmanager/。
2. 在密钥列表中，选择要删除的密钥。
3. 在密钥详细信息部分中，选择操作，然后选择编辑描述。
4. 在禁用密钥和计划删除对话框中，在等待时间下，输入永久删除之前等待的天数。Secrets Manager 附加一个名为 `DeletionDate` 的字段，并将其设置为当前日期和时间加上为恢复时段指定的天数。
5. 选择计划删除。

查看已删除的密钥

1. 打开 Secrets Manager 控制台，网址为https://console.aws.amazon.com/secretsmanager/。
2. 在密钥页面上，选择偏好
   ( )。
3. 在“首选项”对话框中，选择显示计划删除的密钥，然后选择保存。

删除副本密钥

1. 打开 Secrets Manager 控制台，网址为https://console.aws.amazon.com/secretsmanager/。
2. 选择主密钥。
3. 在复制密钥密钥部分，选择副本密钥。
4. 从 Actions (操作) 菜单中选择 Delete Stack (删除副本)。

AWS CLI

Example 删除密钥

以下 `delete-secret` 示例将删除密钥。您可以在 DeletionDate 响应字段中的日期和时间`restore-secret`之前恢复密钥。要删除复制到其他区域的密钥，请先使用 `remove-regions-from-replication` 删除其副本，然后调用 `delete-secret`。

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

Example 立即删除密钥

以下 `delete-secret` 示例将立即删除密钥而没有恢复时段。您无法恢复此密钥。

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

Example 删除副本密钥

以下 `remove-regions-from-replication` 示例将删除 eu-west-3 中的副本密钥。要删除复制到其他区域的主密钥，请先删除副本，然后调用 `delete-secret`。

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

AWS SDK

要删除密钥，请使用 `DeleteSecret` 命令。要删除密钥版本，请使用 `UpdateSecretVersionStage` 命令。要删除副本，请使用 `StopReplicationToReplica` 命令。有关更多信息，请参阅 the section called “AWS SDKs”。

恢复密 AWS Secrets Manager 钥

Secrets Manager 将计划删除的密钥视为已弃用，而不再直接访问该密钥。在恢复时段过后，Secrets Manager 将永久删除该密钥。在 Secrets Manager 删除密钥后，您无法恢复该密钥。在恢复时段结束之前，您可以恢复密钥并再次使其可进行访问。这会删除 `DeletionDate` 字段，从而取消计划的永久删除。

要在控制台中恢复密钥和元数据，您必须具有 `secretsmanager:ListSecrets` 和 `secretsmanager:RestoreSecret` 权限：

当您恢复密钥时，Secrets Manager 会生成一个 CloudTrail 日志条目。有关更多信息，请参阅 the section called “使用登录 AWS CloudTrail ”。

要恢复密钥（控制台）

1. 打开 Secrets Manager 控制台，网址为https://console.aws.amazon.com/secretsmanager/。
2. 在密钥列表中，选择要恢复的密钥名称。

   如果密钥列表中未显示删除的密钥，请选择偏好
   ( )。
   在“首选项”对话框中，选择显示计划删除的密钥，然后选择保存。
3. 在密钥详细信息部分中，选择取消删除。
4. 在取消密钥删除确认对话框中，选择取消删除。

AWS CLI

Example 恢复之前删除的密钥

以下 `restore-secret` 示例恢复了先前计划删除的密钥。

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

AWS SDK

要恢复标记删除的密钥，请使用 `RestoreSecret` 命令。有关更多信息，请参阅 the section called “AWS SDKs”。

在中标记机密 AWS Secrets Manager

在中 AWS Secrets Manager，您可以使用标签将元数据分配给密钥。标签是由您为密钥定义的键/值对。标签可帮助您管理 AWS 资源和整理数据，包括账单信息。

借助标签，您可以：

- 管理、搜索和筛选 AWS 账户中的密钥和其他资源
- 根据附加标签控制对密钥的访问
- 跟踪和分类与特定机密或项目相关的费用

有关使用标签控制访问的更多信息，请参阅the section called “使用标签控制对密钥的访问”。

要了解有关成本分配标签的信息，请参阅 AWS Billing 用户指南中的使用 AWS 成本分配标签。

有关标签配额和命名限制的信息，请参阅AWS 一般参考指南中的标记的服务配额。标签区分大小写。

当您标记或取消标记密钥时，Secrets Manager 会生成 CloudTrail 日志条目。有关更多信息，请参阅 the section called “使用登录 AWS CloudTrail ”。

> **Tip**
> 对所有 AWS 资源使用一致的标记方案。有关最佳实践，请参阅《标记最佳实践》白皮书。

查看标签基本知识

您可以在控制台中按标签查找密钥 AWS CLI、和 SDKs。 AWS 还提供了 Res our ce Groups 工具以便您创建自定义控制台，从而根据资源的标签整合与整理资源。要查找带有特定标签的密钥，请参阅 the section called “查找密钥”。

您可以使用 Secrets Manager 控制台 AWS CLI、或 Secrets Manager API 来执行以下操作：

- 创建带有标签的密钥
- 向密钥添加标签
- 列出您的秘密的标签
- 从密钥中删除标签

您可以使用标签对密钥进行分类。例如，您可以按用途、所有者或环境对密钥进行分类。由于您定义每个标签的键和值，因此您可以创建一组自定义类别来满足您的特定需求。以下几个标签示例：

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

使用标签跟踪成本

您可以使用标签对 AWS 成本进行分类和跟踪。当您将标签应用于 AWS 资源（包括密钥）时，您的 AWS 成本分配报告将包括按标签汇总的使用率和成本。您可以设置代表业务类别（例如成本中心、应用程序名称或所有者）的标签，以便整理多种服务的成本。有关更多信息，请参阅 AWS Billing 用户指南中的对自定义账单报告使用成本分配标签。

了解标签限制

以下限制适用于标签。

基本限制

- 每个资源（密钥）的最大标签数是 50。
- 标签键和值区分大小写。
- 您无法更改或编辑已删除的密钥的标签。

标签键限制

- 每个标签键必须是唯一的。如果您添加的标签具有已使用的键，则您的新标签将覆盖现有键值对。
- 标签键不能以开头，`aws:`因为此前缀将预留以供使用 AWS。 AWS 代表您创建以此前缀开头的标签，但您不能编辑或删除这些标签。
- 标签键的长度必须介于 1 和 128 个 Unicode 字符之间。
- 标签键必须包含以下字符：Unicode 字母、数字、空格和以下特殊字符：`_ . / = + - @`。

标签值限制

- 标签值的长度必须介于 0 和 255 个 Unicode 字符之间。
- 标签值可以为空。另外，它们必须包含以下字符：Unicode 字母、数字、空格和以下任意特殊字符：`_ . / = + - @`。

使用 Secrets Manager 控制台标记密钥

您可以使用 Secrets Manag er 控制台管理密钥的标签。

要访问标签功能，请执行以下操作：

1. 打开 Secrets Manager 控制台。
2. 在导航栏中，选择您的首选区域。
3. 在密钥页面上，选择一个密钥。

查看密钥的标签

- 在密钥详细信息页面上，选择标签选项卡。

使用标签创建密钥

- 按照创建密钥中的步骤操作。

添加或编辑密钥的标签

1. 在密钥详细信息页面上，选择标签选项卡，然后选择编辑标签。
2. 在密钥字段中输入标签密钥。或者，在 “值” 字段中输入标签值。
3. 选择保存。新的或更新的标签出现在标签列表中。

> **Note**
>
> 如果未启用 Save 按钮，则标签键或值可能不满足标签限制。有关更多信息，请参阅 了解标签限制。

从密钥中删除标签

1. 在密钥详细信息页面上，选择标签选项卡，然后选择要移除的标签旁边的删除图标。
2. 选择 “保存” 以确认删除，或选择 “撤消” 以取消。

使用标记机密 AWS CLI

AWS CLI 示例

Example 向密钥添加标签

以下 `tag-resource` 示例说明了如何使用速记语法附加标签。

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

Example 向密钥添加多个标签

以下 `tag-resource` 示例将向密钥附加两个键值标签。

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

Example 从密钥中删除标签

以下 `untag-resource` 示例将从密钥中删除两个标签。对于每个标签，键和值都会被删除。

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

使用 Secrets Manager API 标记密钥

您可以使用 Secrets Manager API 添加、列出和删除标签。有关示例，请参阅以下文档：

- ListSecrets: `ListSecrets` 用于查看应用于密钥的标签

- TagResource: 为密钥添加标签
- 取消标签：从密钥中删除标签

使用 Secrets Manager AWS 开发工具包标记密钥

要更改您的秘密的标签，请使用以下 API 操作：

- ListSecrets: `ListSecrets` 用于查看应用于密钥的标签
- `TagResource`: 为密钥添加标签
- `UntagResource`: 从密钥中删除标签

有关如何使用开发工具包的更多信息，请参阅the section called “AWS SDKs”。

跨区域复制 AWS Secrets Manager 密钥

您可以将您的密钥分成多个复制 AWS 区域 ，以支持分布在这些地区的应用程序，从而满足区域访问和低延迟要求。如果以后需要，您可以将副本密钥升级为独立密钥，然后将其设置为独立复制。Secrets Manager 可以跨指定区域复制加密密钥数据和元数据，例如标签和资源策略。

除区域外，副本密钥的 ARN 与主密钥相同，例如：

- 主密钥：`arn:aws:secretsmanager:Region1:123456789012:secret:MySecret-a1b2c3`
- 副本密钥：`arn:aws:secretsmanager:Region2:123456789012:secret:MySecret-a1b2c3`

有关副本密钥的定价信息，请参阅 AWS Secrets Manager 定价。

当存储复制到其他区域的源数据库的数据库凭证时，密钥将包含源数据库的连接信息。然后复制密钥时，副本将是源密钥的副本，并且包含相同的连接信息。您可以向密钥中添加其他 key/value 配对以获取区域连接信息。

如果您为主密钥启用轮换，Secrets Manager 将在主区域中进行密钥轮换，新的密钥值会传播到所有关联的副本密钥。您无需单独管理所有副本密钥的轮换。

您可以在所有已启用的 AWS 区域中复制密钥。但是，如果您在特殊 AWS 区域（例如 AWS GovCloud (US) 或中国区域）使用 Secrets Manager，则只能在这些 AWS 特殊区域内配置密钥和副本。您不能将已启用 AWS 区域中的密钥复制到专门区域，也不能将机密从专业区域复制到商业区域。

在将密钥复制到另一个区域之前，您必须启用该区域。有关更多信息，请参阅管理 AWS 区域。

通过调用存储密钥的区域中的 Secrets Manager 端点，则无需复制密钥即可在多个区域中使用密钥。有关 终端节点的列表，请参阅the section called “Secrets Manager 端点”。要使用复制来提高工作负载的弹性，请参阅第一部分：云端恢复策略中的灾难恢复 (DR) 架构。 AWS

当您复制密钥时，Secrets Manager 会生成一个 CloudTrail 日志条目。有关更多信息，请参阅 the section called “使用登录 AWS CloudTrail ”。

要将密钥复制到其他地区（控制台）

1. 打开 Secrets Manager 控制台，网址为https://console.aws.amazon.com/secretsmanager/。
2. 从密钥列表上，选择您的密钥。

3. 在密钥详细信息页面的复制选项卡中，执行以下任意一项操作：
   - 如果未复制密钥，请选择 Replicate secret（复制密钥）。
   - 如果已复制密钥，请在 Replicate secret（复制密钥）部分，选择 Add Region（添加区域）。
4. 在 Add replica regions 对话框中，执行以下操作：
   a. 针对 AWS 区域，请选择要将密钥复制粘贴的区域。
   b. （可选）对于加密密钥中，选择用来加密密钥的 KMS 密钥。密钥必须位于副本区域中。
   c. （可选）要添加其他区域，请选择 Add more regions（添加更多区域）。
   d. 选择 Replicate（复制）。

   此时会返回到密钥详细信息页面。Replicate secret（复制密钥）部分会显示每个区域的 Replication status（复制状态）。

AWS CLI

Example 将密钥复制到其他区域

以下 `replicate-secret-to-regions` 示例将密钥复制到 eu-west-3。副本使用 AWS 托管密钥加密aws/secretsmanager。

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

Example 创建密钥并复制它

以下示例创建一个密钥并将其复制到 eu-west-3。副本使用加密 AWS 托管式密钥 aws/secretsmanager。

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

AWS SDK

要复制密钥，请使用 `ReplicateSecretToRegions` 命令。有关更多信息，请参阅 the section called “AWS SDKs”。

在中将副本密钥提升为独立密钥 AWS Secrets Manager

副本密钥是从主密钥复制到另一个密钥 AWS 区域。它具有与主密钥相同的密钥值和元数据，但对它的加密可以使用不同的 KMS 密钥。不能独立于主密钥更新副本密钥，但其加密密钥除外。副本密钥升级会断开副本密钥与主密钥的连接，并使副本机密成为独立密钥。对主密钥的更改不会复制到独立密钥。

在主密钥不可用的情况下，您可能希望将副本密钥升级为独立密钥，以此作为灾难恢复解决方案。或者，如果要为副本密钥启用轮换，您可能需要将副本密钥升级为独立密钥。

如您升级副本密钥，请务必更新相应的应用程序来使用独立密钥。

当你提升密钥时，Secrets Manager 会生成一个 CloudTrail 日志条目。有关更多信息，请参阅 the section called “使用登录 AWS CloudTrail ”。

升级副本密钥（控制台）

1. 登录 Secrets Manager，网址为https://console.aws.amazon.com/secretsmanager/。
2. 导航至副本区域。
3. 在 Secrets（密钥）列表页上，选择副本密钥。
4. 在副本密钥详细信息页面上，选择 Promote to standalone secret（升级为独立密钥）。
5. 在 Promote replica to standalone secret（将副本升级为独立密钥）对话框中，输入区域，然后选择 Promote replica（提升副本）。

AWS CLI

Example 将副本密钥提升为主密钥

以下 `stop-replication-to-replica` 示例将删除副本密钥与主密钥之间的链接。副本密钥在副本区域中被提升为主密钥。您必须从副本区域内调用 `stop-replication-to-replica`。

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

AWS SDK

要将副本密钥升级为独立密钥，请使用 `StopReplicationToReplica` 命令。您必须从副本密钥区域调用此命令。有关更多信息，请参阅 the section called “AWS SDKs”。

防止 AWS Secrets Manager 复制

由于密钥可以使用 `ReplicateSecretToRegions` 进行复制或在使用 `CreateSecret` 创建时进行复制，因此如果您想防止用户复制密钥，我们建议您阻止包含 `AddReplicaRegions` 参数的操作。您可以在权限策略中使用 `Condition` 语句，以仅允许不添加副本区域的操作。有关您可以使用的条件语句，请参阅以下策略示例。

Example 防止复制权限

以下策略示例演示了如何允许所有不添加副本区域的操作。这可以防止用户同时通过 `ReplicateSecretToRegions` 和 `CreateSecret` 复制密钥。

JSON

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

Example 仅允许向特定区域提供复制权限

以下策略演示了如何允许以下所有操作：

- 创建密钥而不复制

- 创建密钥并复制到仅位于美国和加拿大的区域
- 仅将密钥复制到位于美国和加拿大的区域

JSON

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

排查 AWS Secrets Manager 复制问题

AWS Secrets Manager 复制可能会因各种原因而失败。要检查密钥复制失败的原因，您可以执行下列操作之一：

- 调用 `DescribeSecret` API 操作
- 查看 AWS CloudTrail 活动

复制失败时：

- 如果没有可用的密钥版本，Secrets Manager 会将该密钥从副本区域中移除。

- 如果成功复制了密钥版本，则它们将保留在副本区域中，直到您使用 `RemoveRegionsFromReplication` API 操作将其明确移除。

以下各节将介绍复制失败的一些常见原因。

选定区域中存在名称相同的密钥。

为此解决此问题，可以覆盖副本区域中的重复名称密钥。重试复制，然后在重试复制对话框中，选择覆盖。

KMS 密钥上没有可用的权限来完成复制。

Secrets Manager 首先解密密密钥，然后使用副本区域中的新 KMS 密钥重新加密。如果您在主区域中没有加密密钥的 `kms:Decrypt` 权限，则会遇到此错误。要使用 `aws/secretsmanager` 以外的 KMS 密钥对复制的秘密进行加密，您需要对密钥进行 `kms:GenerateDataKey` 和 `kms:Encrypt`。请参阅the section called “KMS 密钥的权限”。

KMS 密钥已禁用或者未找到

如果主区域中的加密密钥被禁用或删除，则 Secrets Manager 将无法复制该秘密。即使您更改了加密密钥，如果秘密具有使用已禁用或删除的加密密钥加密的自定义标签版本，也可能发生此错误。有关 Secrets Manager 如何加密的信息，请参阅 the section called “密钥加密和解密”。要解决此问题，您可以重新创建秘密版本，以便 Secrets Manager 使用当前加密密钥对其进行加密。有关更多信息，请参阅更改秘密的加密密钥。然后重试复制。

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

您尚未启用要复制的区域。

有关如何启用区域的信息，请参阅 AWS 账户管理参考指南中的管理 AWS 区域。

从中获取秘密 AWS Secrets Manager

当你检索密钥时，Secrets Manager 会生成一个 CloudTrail 日志条目。有关更多信息，请参阅 the section called “使用登录 AWS CloudTrail ”。

您可以使用以下方式检索密钥值：

- 使用 Java 获取 Secrets Manager 密钥值
- 使用 Python 获取 Secrets Manager 密钥值
- 使用获取 Secrets Manager 密钥值 .NET
- 使用 Go 获取 Secrets Manager 密钥值
- 使用 Rust 获取 Secrets Manager 密钥值
- 在亚马逊 Elastic Kubernetes Service 中使用 AWS Secrets Manager 密钥
- 在 AWS Lambda 函数中使用 AWS Secrets Manager 密钥
- 使用代 AWS Secrets Manager 理
- 使用 C++ AWS 软件开发工具包获取 Secrets Manager 密钥值
- 使用 JavaScript AWS SDK 获取 Secrets Manager 密钥值
- 使用 Kotlin AWS SDK 获取 Secrets Manager 的密钥值
- 使用 PHP AWS 开发工具包获取 Secrets Manager 密钥值
- 使用 Ruby AWS SDK 获取 Secrets Manager 密钥值
- 使用获取密钥值 AWS CLI
- 使用 AWS 控制台获取密钥值
- 在中使用 AWS Secrets Manager 秘密 AWS Batch
- 在 AWS CloudFormation 资源中获取 AWS Secrets Manager 秘密
- 在 GitHub 工作中使用 AWS Secrets Manager 秘密
- 用 AWS Secrets Manager 于 GitLab
- 在中使用 AWS Secrets Manager 秘密 AWS IoT Greengrass
- 在参数存储中使用 AWS Secrets Manager 密钥

使用 Java 获取 Secrets Manager 密钥值

在应用程序中，您可以通过调用`GetSecretValue`或`BatchGetSecretValue`在任一应用程序中检索您的秘密 AWS SDKs。不过，我们建议您通过使用客户端缓存来缓存您的密钥值。缓存密钥可以提高速度并降低成本。

要使用密钥中的凭证连接到数据库，您可以使用 Secrets Manager SQL 连接驱动程序，它封装了基本 JDBC 驱动程序。它还使用客户端缓存，因此可以降低调用 Secrets Manager APIs 的成本。

主题

- 使用 Java 和客户端缓存获取 Secrets Manager 密钥值
- 使用 JDBC 和 AWS Secrets Manager 密钥中的凭证连接到 SQL 数据库
- 使用 Java AWS SDK 获取 Secrets Manager 密钥值

使用 Java 和客户端缓存获取 Secrets Manager 密钥值

在检索密钥时，您可以使用 Secrets Manager 基于 Java 的缓存组件来缓存密钥，以备将来使用。检索已缓存密钥比从 Secrets Manager 中检索密钥的速度要快。由于调用 Secrets Manager 需要付费 APIs，因此使用缓存可以降低成本。有关检索密钥的所有方法，请参阅 获取密钥。

缓存策略为“最近最少使用 (LRU)”，因此当缓存必须丢弃某个密钥时，它会丢弃最近使用最少的密钥。原定设置下，缓存会每小时刷新一次秘密。您可以配置在缓存中刷新密钥的频率，也可以挂钩到密钥检索中以添加更多功能。

一旦释放缓存引用，缓存便不会进行强制垃圾回收。缓存实施不包括缓存失效。缓存实现侧重于缓存本身，而不是侧重加强安全性或以安全性为重点。如果您需要额外的安全性（例如加密缓存中的项目），请使用提供的接口和抽象方法。

要使用该组件，您必须满足以下条件：

- Java 8 或更高版本的开发环境。请参阅 Oracle 网站上的 Java SE 下载。

要下载源代码，请参阅上的 S ecrets Manager 基于 Java 的缓存客户端组件。 GitHub

要将该组件添加到您的项目中，请在 Maven pom.xml 文件中包括以下依赖项。有关 Maven 的更多信息，请参阅 Apache Maven Project 网站上的《入门指南》。

```
<dependency>
```

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

所需权限：

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

有关更多信息，请参阅 权限参考。

参考

- SecretCache
- SecretCacheConfiguration
- SecretCacheHook

Example 检索密钥

以下代码示例显示了检索密钥字符串的 Lambda 函数。它遵循在函数处理程序之外实例化缓存的最佳实践，因此如果您再次调用该 Lambda 函数，它不会继续调用该 API。

```
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

适用于从 Secrets Manager 请求的密钥的内存中缓存。您使用 the section called “getSecretString” 或 the section called “getSecretBinary” 从缓存中检索密钥。您可以通过传入构造函数中的 the section called “SecretCacheConfiguration” 对象来配置缓存设置。

有关包括示例在内的更多信息，请参阅 the section called “Java 与客户端缓存”。

构造函数

`public SecretCache()`

适用于 `SecretCache` 对象的默认构造函数。

`public SecretCache(AWSSecretsManagerClientBuilder builder)`

使用 Secrets Manager 客户端（使用提供的 `AWSSecretsManagerClientBuilder` 创建）构造新缓存。使用此构造函数自定义 Secrets Manager 客户端，例如使用某一特定区域或端点。

`public SecretCache(AWSSecretsManager client)`

请使用提供的 `AWSSecretsManagerClient` 构造新密钥缓存。使用此构造函数自定义 Secrets Manager 客户端，例如使用某一特定区域或端点。

`public SecretCache(SecretCacheConfiguration config)`

请使用提供的 `the section called “SecretCacheConfiguration”` 构造新密钥缓存。

方法

getSecretString

`public String getSecretString(final String secretId)`

从 Secrets Manager 中检索字符串密钥。返回 `String`。

getSecretBinary

`public ByteBuffer getSecretBinary(final String secretId)`

从 Secrets Manager 中检索二进制密钥。返回 `ByteBuffer`。

refreshNow

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

强制刷新缓存。如果刷新完成没有错误，将返回 `true`，否则将返回 `false`。

close

```
public void close()
```

关闭缓存。

SecretCacheConfiguration

适用于 the section called “SecretCache” 的缓存配置选项，例如最大缓存大小和已缓存密钥的生存时间 (TTL)。

构造函数

```
public SecretCacheConfiguration
```

适用于 `SecretCacheConfiguration` 对象的默认构造函数。

方法

getClient

```
public AWSSecretsManager getClient()
```

返回缓存从中检索密钥的 `AWSSecretsManagerClient`。

setClient

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

设置缓存从中检索密钥的 `AWSSecretsManagerClient` 客户端。

getCacheHook

```
public SecretCacheHook getCacheHook()
```

返回用于挂钩缓存更新的 the section called “SecretCacheHook” 接口。

setCacheHook

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

设置用于挂钩缓存更新的 the section called “SecretCacheHook” 接口。

getMaxCache大小

```
public int getMaxCacheSize()
```

返回最大缓存大小。默认值为 1024 个密钥。

setMaxCache大小

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

设置最大缓存大小。默认值为 1024 个密钥。

getCacheItemTTL

```
public long getCacheItemTTL()
```

返回已缓存项目的 TTL（以毫秒为单位）。当已缓存密钥超过此 TTL 时，缓存将从 `AWSSecretsManagerClient` 中检索该密钥的新副本。默认值为 1 小时（以毫秒为单位）。

在 TTL 之后请求密钥时，缓存将同步刷新密钥。如果同步刷新失败，缓存将返回过时密钥。

setCacheItemTTL

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

为已缓存项目设置 TTL（以毫秒为单位）。当已缓存密钥超过此 TTL 时，缓存将从 `AWSSecretsManagerClient` 中检索该密钥的新副本。默认值为 1 小时（以毫秒为单位）。

getVersionStage

```
public String getVersionStage()
```

返回您要缓存的密钥的版本。有关更多信息，请参阅密钥版本。默认值为 `"AWSCURRENT"`。

setVersionStage

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

设置您要缓存的密钥的版本。有关更多信息，请参阅密钥版本。默认值为 `"AWSCURRENT"`。

SecretCacheConfiguration 与客户一起

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

设置 `AWSSecretsManagerClient` 以从中检索密钥。返回具有新设置的更新后的 `SecretCacheConfiguration` 对象。

SecretCacheConfiguration withCacheHook

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

设置用于挂钩内存中缓存的接口。返回具有新设置的更新后的 `SecretCacheConfiguration` 对象。

SecretCacheConfiguration withMaxCache大小

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

设置最大缓存大小。返回具有新设置的更新后的 `SecretCacheConfiguration` 对象。

SecretCacheConfiguration withCacheItemTTL

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

为已缓存项目设置 TTL（以毫秒为单位）。当已缓存密钥超过此 TTL 时，缓存将从 `AWSSecretsManagerClient` 中检索该密钥的新副本。默认值为 1 小时（以毫秒为单位）。返回具有新设置的更新后的 `SecretCacheConfiguration` 对象。

SecretCacheConfiguration withVersionStage

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

设置您要缓存的密钥的版本。有关更多信息，请参阅密钥版本。返回具有新设置的更新后的 `SecretCacheConfiguration` 对象。

SecretCacheHook

用于挂钩到 the section called “SecretCache” 中以便对存储于缓存中的密钥执行操作的接口。

put

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

准备对象以存储到缓存中。

返回要存储在缓存中的对象。

get

`Object get(final Object cachedObject)`

从已缓存对象派生对象。

返回要从缓存中返回的对象

使用 JDBC 和 AWS Secrets Manager 密钥中的凭证连接到 SQL 数据库

在 Java 应用程序中，你可以使用 Secrets Manager SQL Connection 驱动程序使用存储在 Secrets Manager 中的凭据连接到 MySQL、PostgreSQL MSSQLServer、Oracle、Db2 和 Redshift 数据库。每个驱动程序都会包装基本 JDBC 驱动程序，因此您可以使用 JDBC 调用来访问数据库。但是，您不必为连接传递用户名和密码，而是提供密钥的 ID。驱动程序将调用 Secrets Manager 来检索密钥值，然后使用密钥中的凭证连接到数据库。驱动程序还将使用 Java 客户端缓存库来缓存凭证，这样未来的连接就不需要调用 Secrets Manager。默认情况下，缓存会每小时刷新一次，此外在轮换密钥时也会刷新。要配置缓存，请参阅the section called “SecretCacheConfiguration”。

您可以从 GitHub中下载源代码。

要使用 Secrets Manager SQL 连接驱动程序：

- 您的应用程序必须处于 Java 8 或更高版本中。
- 您的密钥必须为以下之一：
  - 预期 JSON 结构中的数据库密钥。要检查格式，请在 Secrets Manager 控制台中查看密钥并选择 Retrieve secret value（检索密钥值）。或者，在通话 AWS CLI中get-secret-value。
  - Amazon RDS 托管密钥。对于此类密钥，必须在建立连接时指定端点和端口。
  - Amazon Redshift 托管密钥。对于此类密钥，必须在建立连接时指定端点和端口。

如果您的数据库复制到其他区域，要连接到另一个区域中的副本数据库，请在创建连接时指定区域端点和端口。您可以在作为额外键值对的密钥中、SSM 参数存储参数或代码配置中存储区域连接信息。

要将驱动程序添加到项目中，请在 Maven 构建文件 `pom.xml` 中为该驱动程序添加以下依赖项。有关更多信息，请参阅 Maven Central 存储库网站上的 Secrets Manager SQL 连接库。

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

驱动程序使用默认凭证提供程序链。如果您在 Amazon EKS 上运行驱动程序，它可能会获取正在运行的节点的凭证，而不会获取服务账户角色。要解决此问题，请将 `com.amazonaws:aws-java-sdk-sts` 的版本 1 作为依赖项添加到 Gradle 或 Maven 项目文件。

要在`secretsmanager.properties`文件中设置 AWS PrivateLink DNS 终端节点 URL 和区域，请执行以下操作：

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

要覆盖主区域，请设置 `AWS_SECRET_JDBC_REGION` 环境变量或对 `secretsmanager.properties` 文件进行以下更改：

```
drivers.region = region
```

所需权限：

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

有关更多信息，请参阅 权限参考。

示例：

- 建立与数据库的连接
- 通过指定端点和端口建立连接
- 使用 c3p0 连接池建立连接
- 使用 c3p0 连接池通过指定端点和端口来建立连接

建立与数据库的连接

下面的示例演示了如何使用密钥中的凭证和连接信息建立与数据库的连接。建立连接后，您可使用 JDBC 调用来访问数据库。有关更多信息，请参阅 Java 文档网站上的 JDBC 基础知识。

MySQL

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

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

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

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

PostgreSQL

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

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

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

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

Oracle

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

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

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

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

MSSQLServer

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

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

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

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

Db2

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

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

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

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

Redshift

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

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

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

```
info.put( "user", "secretId" );

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

通过指定端点和端口建立连接

以下示例演示了如何使用密钥中的凭证以及指定的端点和端口建立与数据库的连接。

Amazon RDS 托管密钥不包括数据库的端点和端口。要使用由 Amazon RDS 管理的密钥中的主凭证连接到数据库，请在代码中指定这些凭证。

复制到其他区域的密钥可以降低连接到区域数据库的延迟，但复制的密钥将不包含与源密钥不同的连接信息。每个副本都与源密钥相同。要将区域连接信息存储到密钥中，请添加更多键值对来存储区域的端点和端口信息。

建立连接后，您可使用 JDBC 调用来访问数据库。有关更多信息，请参阅 Java 文档网站上的 JDBC 基础知识。

MySQL

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

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

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

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

PostgreSQL

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

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

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

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

Oracle

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

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

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

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

MSSQLServer

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

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

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

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

Db2

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

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

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

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

Redshift

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

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

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

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

使用 c3p0 连接池建立连接

以下示例演示了如何使用 `c3p0.properties` 文件建立连接池，该文件使用驱动程序从密钥中检索凭证和连接信息。对于 `user` 和 `jdbcUrl`，请输入密钥 ID 以配置连接池。然后，您可以从该池中检索连

接，并将这些连接用作任何其他数据库连接。有关更多信息，请参阅 Java 文档网站上的 JDBC 基础知识。

有关 c3p0 的更多信息，请参阅 Machinery For Change 网站上的 c3p0。

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
```

使用 c3p0 连接池通过指定端点和端口来建立连接

以下示例演示了如何使用 `c3p0.properties` 文件建立连接池，该文件使用驱动程序通过指定的端点和端口检索密钥中的凭证。然后，您可以从该池中检索连接，并将这些连接用作任何其他数据库连接。有关更多信息，请参阅 Java 文档网站上的 JDBC 基础知识。

Amazon RDS 托管密钥不包括数据库的端点和端口。要使用由 Amazon RDS 管理的密钥中的主凭证连接到数据库，请在代码中指定这些凭证。

复制到其他区域的密钥可以降低连接到区域数据库的延迟，但复制的密钥将不包含与源密钥不同的连接信息。每个副本都与源密钥相同。要将区域连接信息存储到密钥中，请添加更多键值对来存储区域的端点和端口信息。

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
```

使用 Java AWS SDK 获取 Secrets Manager 密钥值

在应用程序中，您可以通过调用`GetSecretValue`或`BatchGetSecretValue`在任一应用程序中检索您的秘密 AWS SDKs。不过，我们建议您通过使用客户端缓存来缓存您的密钥值。缓存密钥可以提高速度并降低成本。

- 如果您将数据库凭证存储在密钥中，请使用 Secrets Manager SQL 连接驱动程序借助密钥中的凭证连接到数据库。
- 对于其他类型的密钥，使用 Secrets Manager 基于 Java 的缓存组件，或直接使用 `GetSecretValue` 或 `BatchGetSecretValue` 调用 SDK。

以下代码示例演示如何使用 `GetSecretValue`。

所需权限：`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);
        }
    }
}
```

使用 Python 获取 Secrets Manager 密钥值

在应用程序中，您可以通过调用`GetSecretValue`或`BatchGetSecretValue`在任一应用程序中检索您的秘密 AWS SDKs。不过，我们建议您通过使用客户端缓存来缓存您的密钥值。缓存密钥可以提高速度并降低成本。

主题

- 使用 Python 和客户端缓存获取 Secrets Manager 密钥值
- 使用 Python AWS SDK 获取 Secrets Manager 密钥值
- 使用 Python AWS SDK 获取一批 Secrets Manager 密钥值

使用 Python 和客户端缓存获取 Secrets Manager 密钥值

在检索密钥时，您可以使用 Secrets Manager 基于 Python 的缓存组件来缓存密钥，以备将来使用。检索已缓存密钥比从 Secrets Manager 中检索密钥的速度要快。由于调用 Secrets Manager 需要付费 APIs，因此使用缓存可以降低成本。有关检索密钥的所有方法，请参阅 获取密钥。

缓存策略为“最近最少使用 (LRU)”，因此当缓存必须丢弃某个密钥时，它会丢弃最近使用最少的密钥。原定设置下，缓存会每小时刷新一次秘密。您可以配置在缓存中刷新密钥的频率，也可以挂钩到密钥检索中以添加更多功能。

一旦释放缓存引用，缓存便不会进行强制垃圾回收。缓存实施不包括缓存失效。缓存实现侧重于缓存本身，而不是侧重加强安全性或以安全性为重点。如果您需要额外的安全性（例如加密缓存中的项目），请使用提供的接口和抽象方法。

要使用该组件，您必须满足以下条件：

- Python 3.6 或更高版本。
- botocore 1.12 或更高版本。请参阅适用于 Python 和 Botocore 的AWS SDK。
- setuptools_scm 3.2 或更高版本。见 https://pypi。 org/project/setuptools-scm/。

要下载源代码，请参阅上的 S ecrets Manager 基于 Python 的缓存客户端组件。 GitHub

要安装组件，请使用以下命令。

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

所需权限：

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

有关更多信息，请参阅 权限参考。

参考

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

Example 检索密钥

以下示例说明如何获取名为的密钥的机密值*mysecret*。

```
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

适用于从 Secrets Manager 检索的密钥的内存中缓存。您使用 the section called “get_secret_string” 或 the section called “get_secret_binary” 从缓存中检索密钥。您可以通过传入构造函数中的 the section called “SecretCacheConfig” 对象来配置缓存设置。

有关包括示例在内的更多信息，请参阅 the section called “Python 与客户端缓存”。

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

以下是可用方法：

- get_secret_string
- get_secret_binary

get_secret_string

检索密钥字符串值。

请求语法

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

参数

- `secret_id`（字符串）：[必填] 密钥的名称或 ARN。
- `version_stage`（字符串）：您要检索的机密版本。有关更多信息，请参阅 secret versions。默认值为“AWSCURRENT”。

返回类型

字符串

get_secret_binary

检索密钥二进制值。

请求语法

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

```
)
```

参数

- `secret_id`（字符串）：[必填] 密钥的名称或 ARN。
- `version_stage`（字符串）：您要检索的机密版本。有关更多信息，请参阅 secret versions。默认值为“AWSCURRENT”。

返回类型

base64 编码的字符串

SecretCacheConfig

适用于 the section called “SecretCache” 的缓存配置选项，例如最大缓存大小和已缓存密钥的存活时间 (TTL)。

参数

`max_cache_size` (int)

最大缓存大小。默认值为 `1024` 个密钥。

`exception_retry_delay_base` (int)

遇到异常后重试请求之前需要等待的秒数。默认值为 `1`。

`exception_retry_growth_factor` (int)pur

用于计算重试失败请求之间等待时间的增长系数。默认值为 `2`。

`exception_retry_delay_max` (int)

在失败请求之间需要等待的最长时间（以秒为单位）。默认值为 `3600`。

`default_version_stage` (str)

您要缓存的密钥的版本。有关更多信息，请参阅密钥版本。默认值为 `'AWSCURRENT'`。

`secret_refresh_interval` (int)

刷新已缓存密钥信息之间需要等待的秒数。默认值为 `3600`。

`secret_cache_hook` (SecretCacheHook)

`SecretCacheHook` 抽象类的实施。默认值为 `None`。

SecretCacheHook

用于挂钩到 the section called “SecretCache” 中以便对存储于缓存中的密钥执行操作的接口。

以下是可用方法：

- put
- get

put

使对象为存储在缓存中做好准备。

请求语法

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

参数

- obj (对象) -- [必需] 密钥或包含密钥的对象。

返回类型

object

get

从已缓存对象派生对象。

请求语法

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

参数

- obj（对象）：[必填] 包含密钥的秘密或对象。

返回类型

object

@InjectSecretString

此装饰器需要一个密钥 ID 字符串和 the section called “SecretCache” 作为前两个参数。该装饰器将返回密钥字符串值。密钥必须包含一个字符串。

```
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

此装饰器需要一个密钥 ID 字符串和 the section called “SecretCache” 作为前两个参数。其余自变量将已包装函数中的参数映射到密钥中的 JSON 键。密钥必须包含一个 JSON 结构的字符串。

对于包含此 JSON 的密钥：

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

下面的示例演示了如何从密钥中提取 `username` 和 `password` 的 JSON 值。

```
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')
```

使用 Python AWS SDK 获取 Secrets Manager 密钥值

在应用程序中，您可以通过调用GetSecretValue或BatchGetSecretValue在任一应用程序中检索您的秘密 AWS SDKs。不过，我们建议您通过使用客户端缓存来缓存您的密钥值。缓存密钥可以提高速度并降低成本。

对于 Python 应用程序，请使用 Secrets Manager 基于 Python 的缓存组件或直接使用 get_secret_value 或 batch_get_secret_value 调用 SDK。

以下代码示例演示如何使用 GetSecretValue。

所需权限：secretsmanager:GetSecretValue

```
"""
Purpose

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

from get_secret_value import GetSecretWrapper

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


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

    :param secret_name: Name of the secret to retrieve.
    :type secret_name: str
    """
    try:
        # Validate secret_name
        if not secret_name:
            raise ValueError("Secret name must be provided.")
        # Retrieve the secret by name
        client = boto3.client("secretsmanager")
        wrapper = GetSecretWrapper(client)
```

```
        secret = wrapper.get_secret(secret_name)
        # Note: Secrets should not be logged.
        return secret
    except Exception as e:
        logging.error(f"Error retrieving secret: {e}")
        raise

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


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

        :param secret_name: The name of the secret fetched.
        :type secret_name: str
        """
        try:
            get_secret_value_response = self.client.get_secret_value(
                SecretId=secret_name
            )
            logging.info("Secret retrieved successfully.")
            return get_secret_value_response["SecretString"]
        except self.client.exceptions.ResourceNotFoundException:
            msg = f"The requested secret {secret_name} was not found."
            logger.info(msg)
            return msg
        except Exception as e:
            logger.error(f"An unknown error occurred: {str(e)}.")
            raise
```

使用 Python AWS SDK 获取一批 Secrets Manager 密钥值

以下代码示例演示了如何获取批量 Secrets Manager 密钥值。

所需权限：

- `secretsmanager:BatchGetSecretValue`
- 对要检索的每个密钥拥有 `secretsmanager:GetSecretValue` 权限。
- 如果您使用筛选器，则还必须拥有 `secretsmanager:ListSecrets`。

有关权限策略的示例，请参阅 the section called “示例：批量检索一组密钥值的权限”。

> **Important**
>
> 如果您的 VPCE 策略拒绝在您正在检索的群组中检索单个秘密的权限，则 `BatchGetSecretValue` 不会返回任何秘密值，并且会返回错误。

```
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
```

使用获取 Secrets Manager 密钥值 .NET

在应用程序中，您可以通过调用`GetSecretValue`或`BatchGetSecretValue`在任一应用程序中检索您的秘密 AWS SDKs。不过，我们建议您通过使用客户端缓存来缓存您的密钥值。缓存密钥可以提高速度并降低成本。

主题

- 使用 .NET 和客户端缓存获取 Secrets Manager 密钥值
- 使用获取 Secrets Manager 的密钥值 适用于 .NET 的 SDK

使用 .NET 和客户端缓存获取 Secrets Manager 密钥值

在检索密钥时，您可以使用 Secrets Manager 基于 .NET 的缓存组件来缓存密钥，以备将来使用。检索已缓存密钥比从 Secrets Manager 中检索密钥的速度要快。由于调用 Secrets Manager 需要付费 APIs，因此使用缓存可以降低成本。有关检索密钥的所有方法，请参阅 获取密钥。

缓存策略为“最近最少使用 (LRU)”，因此当缓存必须丢弃某个密钥时，它会丢弃最近使用最少的密钥。原定设置下，缓存会每小时刷新一次秘密。您可以配置在缓存中刷新密钥的频率，也可以挂钩到密钥检索中以添加更多功能。

一旦释放缓存引用，缓存便不会进行强制垃圾回收。缓存实施不包括缓存失效。缓存实现侧重于缓存本身，而不是侧重加强安全性或以安全性为重点。如果您需要额外的安全性（例如加密缓存中的项目），请使用提供的接口和抽象方法。

要使用该组件，您必须满足以下条件：

- .NET Framework 4.6.2 或更高版本，或者 .NET Standard 2.0 或更高版本。请参阅 Microsoft .NET 网站上的下载 .NET。
- 适用于.NET 的 AWS SDK。请参阅 the section called “AWS SDKs”。

要下载源代码，请参阅上的 “.NET 缓存客户端” GitHub。

要使用缓存，请先对其进行实例化，然后使用 `GetSecretString` 或 `GetSecretBinary` 检索密钥。在连续检索时，缓存将返回密钥的已缓存副本。

获取缓存包

- 请执行以下操作之一：
  - 在您的项目目录中运行下列 .NET CLI 命令。

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

  - 将下列软件包引用添加到您的 `.csproj` 文件中。

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

所需权限：

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

有关更多信息，请参阅 权限参考。

参考

- SecretsManagerCache
- SecretCacheConfiguration
- ISecretCacheHook

Example 检索密钥

以下代码示例显示了一种检索名为*MySecret*的密钥的方法。

```
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 配置生存时间 (TTL) 缓存刷新持续时间

以下代码示例显示了一种检索名为的密钥*MySecret*并将 TTL 缓存刷新持续时间设置为 24 小时的方法。

```
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

适用于从 Secrets Manager 请求的密钥的内存中缓存。您使用 the section called “GetSecretString” 或 the section called “GetSecretBinary” 从缓存中检索密钥。您可以通过传入构造函数中的 the section called “SecretCacheConfiguration” 对象来配置缓存设置。

有关包括示例在内的更多信息，请参阅 the section called “.NET 与客户端缓存”。

构造函数

`public SecretsManagerCache()`

适用于 `SecretsManagerCache` 对象的默认构造函数。

`public SecretsManagerCache(IAmazonSecretsManager secretsManager)`

使用 Secrets Manager 客户端（使用提供的 AmazonSecretsManagerClient 创建）构造新缓存。使用此构造函数可自定义 Secrets Manager 客户端，例如使用某一特定区域或终端节点。

参数

secretsManager

AmazonSecretsManagerClient要从中检索机密。

`public SecretsManagerCache(SecretCacheConfiguration config)`

使用提供的 the section called “SecretCacheConfiguration” 构造新密钥缓存。使用此构造函数来配置缓存，例如要缓存的密钥数量及其刷新频率。

参数

config

一个 the section called “SecretCacheConfiguration”，其中包含缓存的配置信息。

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

使用使用提供的AmazonSecretsManagerClient和创建的 Secrets Manager 客户端构造新的缓存。the section called “SecretCacheConfiguration”使用此构造函数可自定义 Secrets Manager 客户端，例如使用某一特定区域或终端节点以及配置缓存，例如要缓存的密钥数量及其刷新频率。

参数

secretsManager

AmazonSecretsManagerClient要从中检索机密。

config

一个 the section called “SecretCacheConfiguration”，其中包含缓存的配置信息。

方法

GetSecretString

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

从 Secrets Manager 中检索字符串密钥。

参数

secretId

要检索的密钥的 ARN 或名称。

GetSecretBinary

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

从 Secrets Manager 中检索二进制密钥。

参数

secretId

要检索的密钥的 ARN 或名称。

RefreshNowAsync

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

请从 Secrets Manager 请求密钥值，并使用任何更改更新缓存。如果没有现有的缓存条目，请创建一个新缓存条目。如果刷新成功，则返回 `true`。

参数

secretId

要检索的密钥的 ARN 或名称。

GetCachedSecret

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

返回指定密钥的缓存条目（如果缓存中存在该密钥）。否则，从 Secret Manager 中该检索密钥，并创建一个新缓存条目。

参数

secretId

要检索的密钥的 ARN 或名称。

SecretCacheConfiguration

适用于 the section called “SecretsManagerCache” 的缓存配置选项，例如最大缓存大小和已缓存密钥的存活时间 (TTL)。

属性

CacheItemTTL

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

缓存项目的 TTL（以毫秒为单位）。默认值为 3600000 毫秒或 1 小时。最大值为 4294967295 ms，约为 49.7 天。

MaxCacheSize

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

最大缓存大小。默认值为 1024 个密钥。最大值为 65535。

VersionStage

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

您要缓存的密钥的版本。有关更多信息，请参阅密钥版本。默认值为 "AWSCURRENT"。

客户端

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

AmazonSecretsManagerClient要从中检索机密。如果是 null，缓存将实例化一个新客户端。默认为 null。

CacheHook

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

一个 the section called “ISecretCacheHook”。

ISecretCacheHook

用于挂钩到 the section called “SecretsManagerCache” 中以便对存储于缓存中的密钥执行操作的接口。

方法

Put

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

准备对象以存储到缓存中。

返回要存储在缓存中的对象。

获取

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

从已缓存对象派生对象。

返回要从缓存中返回的对象

使用获取 Secrets Manager 的密钥值 适用于 .NET 的 SDK

在应用程序中，您可以通过调用`GetSecretValue`或`BatchGetSecretValue`在任一应用程序中检索您的秘密 AWS SDKs。不过，我们建议您通过使用客户端缓存来缓存您的密钥值。缓存密钥可以提高速度并降低成本。

对于 .NET 应用程序，请使用 Secrets Manager 基于 .NET 的缓存组件或直接使用 `GetSecretValue` 或 `BatchGetSecretValue` 调用 SDK。

以下代码示例演示如何使用 `GetSecretValue`。

所需权限：`secretsmanager:GetSecretValue`

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.SecretsManager;
using Amazon.SecretsManager.Model;

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

        IAmazonSecretsManager client = new AmazonSecretsManagerClient();

        var response = await GetSecretAsync(client, secretName);
```

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

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

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

            GetSecretValueResponse response = null;

            // For the sake of simplicity, this example handles only the most
            // general SecretsManager exception.
            try
            {
                response = await client.GetSecretValueAsync(request);
            }
            catch (AmazonSecretsManagerException e)
```

```
            {
                Console.WriteLine($"Error: {e.Message}");
            }

            return response;
        }

        /// <summary>
        /// Decodes the secret returned by the call to GetSecretValueAsync and
        /// returns it to the calling program.
        /// </summary>
        /// <param name="response">A GetSecretValueResponse object containing
        /// the requested secret value returned by GetSecretValueAsync.</param>
        /// <returns>A string representing the decoded secret value.</returns>
        public static string DecodeString(GetSecretValueResponse response)
        {
            // Decrypts secret using the associated AWS Key Management Service
            // Customer Master Key (CMK.) Depending on whether the secret is a
            // string or binary value, one of these fields will be populated.
            if (response.SecretString is not null)
            {
                var secret = response.SecretString;
                return secret;
            }
            else if (response.SecretBinary is not null)
            {
                var memoryStream = response.SecretBinary;
                StreamReader reader = new StreamReader(memoryStream);
                string decodedBinarySecret =
System.Text.Encoding.UTF8.GetString(Convert.FromBase64String(reader.ReadToEnd()));
                return decodedBinarySecret;
            }
            else
            {
                return string.Empty;
            }
        }
    }
```

使用 Go 获取 Secrets Manager 密钥值

在应用程序中，您可以通过调用`GetSecretValue`或`BatchGetSecretValue`在任一应用程序中检索您的秘密 AWS SDKs。不过，我们建议您通过使用客户端缓存来缓存您的密钥值。缓存密钥可以提高速度并降低成本。

主题

- 使用 Go 和客户端缓存获取 Secrets Manager 密钥值
- 使用 Go AWS SDK 获取 Secrets Manager 密钥值

使用 Go 和客户端缓存获取 Secrets Manager 密钥值

在检索密钥时，您可以使用 Secrets Manager 基于 Go 的缓存组件来缓存密钥，以备将来使用。检索已缓存密钥比从 Secrets Manager 中检索密钥的速度要快。由于调用 Secrets Manager 需要付费 APIs，因此使用缓存可以降低成本。有关检索密钥的所有方法，请参阅 获取密钥。

缓存策略为“最近最少使用 (LRU)”，因此当缓存必须丢弃某个密钥时，它会丢弃最近使用最少的密钥。原定设置下，缓存会每小时刷新一次秘密。您可以配置在缓存中刷新密钥的频率，也可以挂钩到密钥检索中以添加更多功能。

一旦释放缓存引用，缓存便不会进行强制垃圾回收。缓存实施不包括缓存失效。缓存实现侧重于缓存本身，而不是侧重加强安全性或以安全性为重点。如果您需要额外的安全性（例如加密缓存中的项目），请使用提供的接口和抽象方法。

要使用该组件，您必须满足以下条件：

- AWS 适用于 Go 的 SDK。请参阅 the section called “AWS SDKs”。

要下载源代码，请参阅 S ecrets Manager Go 缓存客户端 GitHub。

要设置 Go 开发环境，请参阅 Go Programming Language 网站上的 Golang 入门。

所需权限：

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

有关更多信息，请参阅 权限参考。

参考

- type Cache
- 键入 CacheConfig
- 键入 CacheHook

Example 检索密钥

以下代码示例显示了检索密钥的 Lambda 函数。

```
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

适用于从 Secrets Manager 请求的密钥的内存中缓存。您使用 the section called “GetSecretString” 或 the section called “GetSecretBinary” 从缓存中检索密钥。

下面的示例演示了如何配置缓存设置。

```
// 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 },
)
```

有关包括示例在内的更多信息，请参阅 the section called “Go 与客户端缓存”。

方法

New

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

New 使用功能选项构造密钥缓存，否则将使用默认值。从新会话初始化 SecretsManager 客户端。初始化 CacheConfig 为默认值。使用默认最大大小初始化 LRU 缓存。

GetSecretString

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

GetSecretString 从缓存中获取给定密钥 ID 的秘密字符串值。返回密钥字符串，如果操作失败则返回错误。

GetSecretStringWithStage

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

GetSecretStringWithStage 从缓存中获取给定密钥 ID 和版本阶段的秘密字符串值。返回密钥字符串，如果操作失败则返回错误。

GetSecretBinary

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

GetSecretBinary 从缓存中获取给定密钥 ID 的秘密二进制值。返回密钥二进制值，如果操作失败则返回错误。

GetSecretBinaryWithStage

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

GetSecretBinaryWithStage 从缓存中获取给定密钥 ID 和版本阶段的秘密二进制值。返回密钥二进制值，如果操作失败则返回错误。

键入 CacheConfig

适用于缓存的缓存配置选项，例如最大缓存大小、默认版本阶段，以及已缓存密钥的存活时间 (TTL)。

```
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
}
```

键入 CacheHook

用于挂钩到缓存中以便对存储于缓存中的密钥执行操作的接口。

方法

Put

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

使对象为存储在缓存中做好准备。

获取

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

从已缓存对象派生对象。

使用 Go AWS SDK 获取 Secrets Manager 密钥值

在应用程序中，您可以通过调用`GetSecretValue`或`BatchGetSecretValue`在任一应用程序中检索您的秘密 AWS SDKs。不过，我们建议您通过使用客户端缓存来缓存您的密钥值。缓存密钥可以提高速度并降低成本。

对于 Go 应用程序，请使用 Secrets Manager 基于 Go 的缓存组件或直接使用 `GetSecretValue` 或 `BatchGetSecretValue` 调用 SDK。

以下代码示例展示了如何获取 Secrets Manager 密钥值。

所需权限：`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.
  }
```

使用 Rust 获取 Secrets Manager 密钥值

在应用程序中，您可以通过调用`GetSecretValue`或`BatchGetSecretValue`在任一应用程序中检索您的秘密 AWS SDKs。不过，我们建议您通过使用客户端缓存来缓存您的密钥值。缓存密钥可以提高速度并降低成本。

主题

- 使用 Rust 和客户端缓存获取 Secrets Manager 密钥值
- 使用 Rust AWS SDK 获取 Secrets Manager 的密钥值

使用 Rust 和客户端缓存获取 Secrets Manager 密钥值

在检索密钥时，您可以使用 Secrets Manager 基于 Rust 的缓存组件来缓存密钥，以备将来使用。检索已缓存密钥比从 Secrets Manager 中检索密钥的速度要快。由于调用 Secrets Manager 需要付费 APIs，因此使用缓存可以降低成本。有关检索密钥的所有方法，请参阅 获取密钥。

缓存策略是先进先出（FIFO），因此当缓存必须丢弃一个密钥时，它会丢弃最旧的密钥。原定设置下，缓存会每小时刷新一次秘密。您可以配置以下内容：

- `max_size` – 在驱逐最近未访问的密钥之前要维护的最大缓存密钥数。
- `ttl` – 在需要刷新密钥状态之前缓存的项目被视为有效的持续时间。

缓存实施不包括缓存失效。缓存实现侧重于缓存本身，而不是侧重加强安全性或以安全性为重点。如果需要额外安全性（例如加密缓存中的项目），请使用提供的特性来修改缓存。

要使用该组件，您必须拥有一个带有 `tokio` 的 Rust 2021 开发环境。有关更多信息，请参阅 Rust 编程语言网站上的入门。

要下载源代码，请参阅上的 S ecrets Manager 基于 Rust 的缓存客户端组件。 GitHub

要安装缓存组件，请使用以下命令。

```
cargo add aws_secretsmanager_caching
```

所需权限：

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

有关更多信息，请参阅 权限参考。

Example 检索密钥

以下示例说明如何获取名为的密钥的机密值*`MyTest`*。

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

let client = match SecretsManagerCachingClient::default(
    NonZeroUsize::new(10).unwrap(),
    Duration::from_secs(60),
)
.await
{
    Ok(c) => c,
```

```
    Err(_) => panic!("Handle this error"),
};

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

// Your code here
```

Example 使用自定义配置和自定义客户端实例化缓存

以下示例说明如何配置缓存，然后获取名为的密钥的密钥值*MyTest*。

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

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

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

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

// Your code here
```

使用 Rust AWS SDK 获取 Secrets Manager 的密钥值

在应用程序中，您可以通过调用`GetSecretValue`或`BatchGetSecretValue`在任一应用程序中检索您的秘密 AWS SDKs。不过，我们建议您通过使用客户端缓存来缓存您的密钥值。缓存密钥可以提高速度并降低成本。

对于 Rust 应用程序，请使用基于 Secrets Manager 的 Rust 缓存组件，或者使用或直接调用 SDK。GetSecretValue BatchGetSecretValue

以下代码示例展示了如何获取 Secrets Manager 密钥值。

所需权限：`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(())
}
```

在亚马逊 Elastic Kubernetes Service 中使用 AWS Secrets Manager 密钥

要将来自 AWS Secrets Manager (ASCP) 的密钥显示为挂载在 Amazon EKS Pod 中的文件，你可以使用 Kubernetes S AWS ecrets Store CSI 驱动程序的密钥和配置提供程序。ASCP 可与运行亚马逊节点组的亚马逊 Elastic Kubernetes Service 1.17+ 配合使用。 EC2 AWS Fargate 不支持节点组。使用 ASCP，您可以在 Secrets Manager 中存储并管理密钥，然后通过 Amazon EKS 上运行的工作负载检索。如果您的密钥包含多个 JSON 格式的键值对，则可以选择在 Amazon EKS 中挂载哪些键值对。ASCP 可使用 JMESPath 语法来查询密钥中的键/值对。ASCP 还适用于 Parameter Store 参数。ASCP 提供两种通过 Amazon EKS 进行身份验证的方法。第一种方法是使用服务账户的 IAM 角色（IRSA），第二种方法是使用容器组身份。每种方法都有其优势和用例。

基于服务账户的 IAM 角色（IRSA）的 ASCP

具有 IAM 服务账户角色的 ASCP (IRSA) 允许您将密钥 AWS Secrets Manager 作为文件挂载到 Amazon EKS Pod 中。这种方法适用于以下情况：

- 你需要将密钥作为文件挂载到 Pod 中。

- 您正在使用带有亚马逊 EC2 节点组的 Amazon EKS 1.17 或更高版本。
- 你想从 JSON 格式的密钥中检索特定的键值对。

有关更多信息，请参阅 the section called “将 ASCP 与 IRSA 集成 Amazon EKS”。

基于容器组身份的 ASCP

基于 EKS 容器组身份的 ASCP

带有 Pod 身份的 ASCP 方法可增强安全性并简化在 Amazon EKS 中访问密钥的配置。这种方法在以下情况下非常有用：

- 需要在容器组（pod）级别进行更精细的权限管理时。
- 使用的是 Amazon EKS 版本 1.24 或更高版本时。
- 需要提高性能和可扩展性时。

有关更多信息，请参阅 the section called “将 ASCP 与适用于 Amazon EKS 的容器组身份集成”。

选择正确的方法

在基于 IRSA 的 ASCP 和基于容器组身份的 ASCP 之间做选择时，需考虑以下因素：

- 亚马逊 EKSversion：Pod Identity 需要亚马逊 EKS 1.24+，而 CSI 驱动程序适用于亚马逊 EKS 1.17+。
- 安全要求：容器组身份可在容器组（pod）级别提供更精细的控制。
- 性能：容器组身份通常会在大规模环境中展现更出色的性能。
- 复杂性：容器组身份无需单独的服务账户，可以简化设置。

选择最符合您的具体要求和 Amazon EKS 环境的方法。

安装适用于 Amazon EKS 的 ASCP

本节介绍如何安装适用于 Amazon EKS 的 AWS 密钥和配置提供程序。使用 ASCP，您可以将 Secrets Manager 中的密钥和来自的参数 AWS Systems Manager 作为文件挂载到 Amazon EKS Pods 中。

先决条件

- Amazon EKS 集群

  - 容器组身份版本 1.24 或更高版本
  - IRSA 版本 1.17 或更高版本
- 已 AWS CLI 安装并配置的
- 已为 Amazon EKS 集群安装并配置 kubectl
- Helm（版本 3.0 或更高版本）

安装和配置 ASCP

ASCP 可在 secrets-store-csi-provider-aws 存储 GitHub 库中找到。回购还包含用于创建和装载密钥的 YAML 文件示例。

安装过程中，您可以将 ASCP 配置为使用 FIPS 端点。有关 终端节点的列表，请参阅the section called “Secrets Manager 端点”。

使用 Helm 安装 ASCP

1. 为确保存储库指向最新图表，请使用 `helm repo update.`
2. 安装图表。以下是该`helm install`命令的示例：

```
helm install -n kube-system secrets-provider-aws aws-secrets-manager/secrets-store-
csi-driver-provider-aws
```

   a. 要使用 FIPS 端点，请添加以下标志：`--set useFipsEndpoint=true`
   b. 要配置节流，请添加以下标志：`--set-json 'k8sThrottlingParams={"qps": "number of queries per second", "burst": "number of queries per second"}'`
   c. 如果您的集群上已经安装了 Secrets Store CSI 驱动程序，请添加以下标志：`--set secrets-store-csi-driver.install=false`。这将跳过将 Secrets Store CSI 驱动程序作为依赖项进行安装。

在存储库中使用 YAML 进行安装

- 使用以下命令。

```
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
```

验证安装情况

要验证 EKS 集群、Secrets Store CSI 驱动程序和 ASCP 插件的安装情况，请按照以下步骤操作：

1. 验证 EKS 集群的安装情况：

   ```
   eksctl get cluster --name clusterName
   ```

   该命令应返回有关集群的信息。

2. 验证 Secrets Store CSI 驱动程序的安装情况：

   ```
   kubectl get pods -n kube-system -l app=secrets-store-csi-driver
   ```

   您应该看到正在运行的容器组（pod），其名称如下：`csi-secrets-store-secrets-store-csi-driver-xxx`。

3. 验证 ASCP 插件的安装情况：

   YAML installation

   ```
   $ kubectl get pods -n kube-system -l app=csi-secrets-store-provider-aws
   ```

   输出示例：

   ```
   NAME                                      READY   STATUS    RESTARTS   AGE
   csi-secrets-store-provider-aws-12345       1/1     Running   0          2m
   ```

   Helm installation

   ```
   $  kubectl get pods -n kube-system -l app=secrets-store-csi-driver-provider-aws
   ```

   输出示例：

```
NAME                                                    READY   STATUS    RESTARTS
 AGE
secrets-provider-aws-secrets-store-csi-driver-provider-67890       1/1
 Running   0          2m
```

您应该看到处于 Running 状态的容器组（pod）。

运行这些命令后，如果一切设置正确，您应该会看到所有组件都在运行，且没有任何错误。如果遇到任何问题，可能需要通过查看出现问题的特定容器组（pod）的日志来进行故障排除。

故障排除

1. 要查看 ASCP 提供者的日志，请运行：

   ```
   kubectl logs -n kube-system -l app=csi-secrets-store-provider-aws
   ```

2. 检查`kube-system`命名空间中所有 pod 的状态：

   ```
   kubectl -n kube-system get pods
   ```

   ```
   kubectl -n kube-system logs pod/PODID
   ```

   所有与 CSI 驱动程序和 ASCP 相关的容器组（pod）都应处于“正在运行”状态。

3. 检查 CSI 驱动程序版本：

   ```
   kubectl get csidriver secrets-store.csi.k8s.io -o yaml
   ```

   该命令应返回有关已安装的 CSI 驱动程序的信息。

其他资源

有关将 ASCP 与 Amazon EKS 结合使用的更多信息，请参阅以下资源：

- 将容器组身份与 Amazon EKS 结合使用
- AWS 秘密商店 CSI 驱动程序开启 GitHub

在 Amazon EKS 中使用带有 Pod 身份的 AWS 密钥和配置提供商 CSI

AWS 机密和配置提供程序与适用于 Amazon Elastic Kubernetes Service 的 Pod 身份代理集成，为在 Amazon EKS 上运行的应用程序提供了增强的安全性、简化的配置和更高的性能。Pod Identity 在从 Secrets Manager 检索密钥或从 Parameter Stor AWS Systems Manager e 检索参数时简化了 Amazon EKS 的

Amazon EKS 容器组身份通过直接在 Amazon EKS 接口设置权限，减少了操作步骤，且无需在 Amazon EKS 和 IAM 服务之间切换，从而简化了为 Kubernetes 应用程序配置 IAM 权限的过程。容器组身份允许在多个集群中共用一个 IAM 角色而无需更新信任策略，并支持角色会话标签以实现更精细的访问控制。这种方法不仅允许跨角色重复使用权限策略，简化了策略管理，而且还允许根据匹配的标签访问 AWS 资源，增强了安全性。

工作方式

1. 容器组身份会为容器组（pod）分配 IAM 角色。
2. ASCP 使用此角色进行身份验证。 AWS 服务
3. 如果获得授权，ASCP 会检索所请求的密钥并将其提供给 Pod。

有关更多信息，请参阅《Amazon EKS 用户指南》中的了解 Amazon EKS 容器组身份的工作原理。

先决条件

> **Important**
>
> 仅云中的 Amazon EKS 支持容器组身份。Amazon EK S Anywhere 或亚马逊实例上的自行管理的 Kubernetes 集群不支持该功能。AWS 云端 Red Hat OpenShift 服务 EC2

- Amazon EKS 集群（版本 1.24 或更高版本）
- 通过以下方式访问 AWS CLI 和 Amazon EKS 集群 `kubectl`
- 访问两个 AWS 账户 （用于跨账户访问）

安装 Amazon EKS 容器组身份代理

要将容器组身份与集群结合使用，必须安装 Amazon EKS 容器组身份代理附加组件。

安装容器组身份代理

- 在集群上安装 Pod Identity Agent 插件：

```
eksctl create addon \
  --name eks-pod-identity-agent \
  --cluster clusterName \
  --region region
```

通过容器组身份设置 ASCP

1. 创建权限策略，向 Pod 需要访问的机密授予`secretsmanager:GetSecretValue`和`secretsmanager:DescribeSecret`权限。有关策略示例，请参阅 the section called “示例：读取和描述个人密钥的权限”。
2. 创建可由容器组身份的 Amazon EKS 服务主体担任的 IAM 角色：

   JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Principal": {
          "Service": "pods.eks.amazonaws.com"
        },
        "Action": [
          "sts:AssumeRole",
          "sts:TagSession"
        ]
      }
    ]
  }
```

   将 IAM 策略附加到该角色：

```
aws iam attach-role-policy \
  --role-name MY_ROLE \
```

```
  --policy-arn POLICY_ARN
```

3. 创建容器组身份关联。有关示例，请参阅《Amazon EKS 用户指南》中的创建容器组身份关联
4. 创建指定`SecretProviderClass`要在 Pod 中挂载哪些密钥的：

```
kubectl apply -f https://raw.githubusercontent.com/aws/secrets-store-csi-driver-provider-aws/main/examples/ExampleSecretProviderClass-PodIdentity.yaml
```

   IRSA 和 Pod Identity SecretProviderClass 之间的主要区别在于可选参数`usePodIdentity`。这是一个可选字段，用于确定身份验证方法。如果未指定，则默认对服务账户（IRSA）使用 IAM 角色。

   - 要使用 EKS 容器组身份，请使用以下任意值：`"true"`, `"True"`, `"TRUE"`, `"t"`, `"T"`。
   - 要明确使用 IRSA，请将其设置为以下任意值：`"false"`, `"False"`, `"FALSE"`, `"f"`, or `"F"`。
5. 将装载机密的 Pod 部署到以下`/mnt/secrets-store`位置：

```
kubectl apply -f kubectl apply -f https://raw.githubusercontent.com/aws/secrets-store-csi-driver-provider-aws/main/examples/ExampleDeployment-PodIdentity.yaml
```

6. 如果您使用私有 Amazon EKS 集群，请确保集群所在的 VPC 具有 AWS STS 终端节点。有关创建端点的信息，请参阅《AWS Identity and Access Management 用户指南》中的接口 VPC 端点。

验证密钥的挂载情况

要验证密钥是否已正确挂载，请运行以下命令：

```
kubectl exec -it $(kubectl get pods | awk '/pod-identity-deployment/{print $1}' | head
 -1) -- cat /mnt/secrets-store/MySecret
```

设置 Amazon EKS Pod Identity 以访问 Secrets Manager 中的密钥

1. 创建权限策略，向 Pod 需要访问的机密授予`secretsmanager:GetSecretValue`和`secretsmanager:DescribeSecret`权限。有关策略示例，请参阅 the section called “示例：读取和描述个人密钥的权限”。
2. 如果你还没有密钥，请在 Secrets Manager 中创建密钥。

故障排除

您可以通过描述容器组（pod）部署来查看大多数错误。

查看容器的错误消息

1. 使用以下命令获取容器组（pod）名称列表。如果您没有使用默认命名空间，请使用 `-n NAMESPACE`。

```
kubectl get pods
```

2. 要描述 Pod，请在以下命令中*PODID*使用您在上一步中找到的 Pod 中的 Pod ID。如果没有使用默认命名空间，请使用 `-n NAMESPACE`。

```
kubectl describe pod/PODID
```

查看 ASCP 的错误

- 要在提供者日志中查找更多信息，请在以下命令中*PODID*使用 csi-secrets-store-provider-aws Pod 的 ID。

```
kubectl -n kube-system get pods
kubectl -n kube-system logs pod/PODID
```

将 AWS 密钥和配置提供商 CSI 与服务账户 IAM 角色配合使用 (IRSA)

主题

- 先决条件
- 设置访问控制
- 确定要挂载的密钥
- 故障排除

先决条件

- 亚马逊 EKS 集群（版本 1.17 或更高版本）
- 通过以下方式访问 AWS CLI 和 Amazon EKS 集群 `kubectl`

设置访问控制

ASCP 检索 Amazon EKS Pod 身份并将其兑换 IAM 角色。您可以在 IAM 策略中为该 IAM 角色设置权限。当 ASCP 代入 IAM 角色时，它可以访问您授权的密钥。除非将其与 IAM 角色关联，否则其他容器无法访问密钥。

授予你的 Amazon EKS Pod 访问 Secrets Manager 中的密钥的权限

1. 创建权限策略，向 Pod 需要访问的机密授予`secretsmanager:GetSecretValue`和`secretsmanager:DescribeSecret`权限。有关策略示例，请参阅the section called “示例：读取和描述个人密钥的权限”。
2. 为集群创建 IAM OpenID Connect (OIDC) 提供商（如果还没有）。有关更多信息，请参阅《Amazon EKS 用户指南》中的为集群创建 IAM OIDC 提供商。
3. 为服务账户创建一个 IAM 角色并将策略附加到该角色。有关更多信息，请参阅《Amazon EKS 用户指南》中的为服务账户创建 IAM 角色。
4. 如果您使用私有 Amazon EKS 集群，请确保集群所在的 VPC 具有 AWS STS 终端节点。有关创建端点的信息，请参阅《AWS Identity and Access Management 用户指南》中的接口 VPC 端点。

确定要挂载的密钥

要确定 ASCP 将哪些密钥作为文件系统上的文件挂载在 Amazon EKS 中，您需要创建一个 the section called “SecretProviderClass” YAML 文件。`SecretProviderClass` 列出了要挂载的密钥以及要挂载这些密钥的文件名。`SecretProviderClass`必须与它引用的 Amazon EKS Pod 位于同一个命名空间中。

将机密作为文件挂载

以下说明说明如何使用示例 YAML 文件.yaml 和 ExampleSecretProviderClass.yaml 将密钥挂载为文件。 ExampleDeployment

在 Amazon EKS 中挂载密钥

1. 将`SecretProviderClass`应用于 Pod：

```
kubectl apply -f ExampleSecretProviderClass.yaml
```

2. 部署你的 Pod：

```
kubectl apply -f ExampleDeployment.yaml
```

3. ASCP 会挂载文件。

故障排除

您可以通过描述 Pod 部署来查看大多数错误。

查看容器的错误消息

1. 使用以下命令获取 Pod 名称列表。如果您没有使用默认命名空间，请使用 `-n nameSpace`。

```
kubectl get pods
```

2. 要描述 Pod，请在以下命令中*podId*使用您在上一步中找到的 Pod 中的 Pod ID。如果没有使用默认命名空间，请使用 `-n nameSpace`。

```
kubectl describe pod/podId
```

查看 ASCP 的错误

- 要在提供者日志中查找更多信息，请在以下命令中*podId*使用 csi-secrets-store-provider-aws Pod 的 ID。

```
kubectl -n kube-system get pods
kubectl -n kube-system logs Pod/podId
```

- 验证 C **SecretProviderClass** RD 是否已安装：

```
kubectl get crd secretproviderclasses.secrets-store.csi.x-k8s.io
```

此命令应返回有关`SecretProviderClass`自定义资源定义的信息。

- 验证 SecretProviderClass 对象是否已创建。

```
kubectl get secretproviderclass SecretProviderClassName -o yaml
```

AWS 机密和配置提供程序代码示例

ASCP 身份验证和访问控制示例

示例：允许 Amazon EKS 容器组身份服务（pods.eks.amazonaws.com）担任角色并标记会话的 IAM 策略：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "pods.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

SecretProviderClass

您可以使用 YAML 描述要使用 ASCP 在 Amazon EKS 中挂载哪些密钥。有关示例，请参阅 the section called “SecretProviderClass 用法”。

SecretProviderClass YAML 结构

```
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
   name: name
spec:
  provider: aws
  parameters:
    region:
```

```
    failoverRegion:
    pathTranslation:
    usePodIdentity:
    preferredAddressType:
    objects:
```

参数字段包含挂载请求的详细信息：

区域

（可选）机密 AWS 区域 中的一个。如果不使用此字段，ASCP 将从节点上的注释中查找 “区域”。查找会增加挂载请求的开销，因此我们建议为使用大量容器组（pod）的集群提供区域。

如果您还指定 `failoverRegion`，ASCP 会尝试从两个区域检索密钥。如果任一区域返回 4xx 错误（例如身份验证问题），ASCP 都不会挂载任何一个密钥。如果成功从 `region` 中检索到密钥，则 ASCP 会挂载该密钥值。如果未成功从 `region` 中检索到密钥，但已成功从 `failoverRegion` 中检索到密钥，则 ASCP 会挂载该密钥值。

failoverRegion

（可选）如果您包含此字段，ASCP 会尝试从 `region` 中定义的区域和此字段检索密钥。如果任一区域返回 4xx 错误（例如身份验证问题），ASCP 都不会挂载任何一个密钥。如果成功从 `region` 中检索到密钥，则 ASCP 会挂载该密钥值。如果未成功从 `region` 中检索到密钥，但已成功从 `failoverRegion` 中检索到密钥，则 ASCP 会挂载该密钥值。有关如何使用此字段的示例，请参阅 多区域密钥故障转移。

pathTranslation

（可选）如果 Amazon EKS 中的文件名包含路径分隔符则要使用的单个替换字符，例如 Linux 上的斜杠 (/)。ASCP 无法创建包含路径分隔符的挂载文件。相反，ASCP 使用不同的字符替换路径分隔符。如果不使用此字段，替换字符为下划线 (_)，因此，例如 `My/Path/Secret` 挂载为 `My_Path_Secret`。

要防止字符替换，请输入字符串 `False`。

usePodIdentity

（可选）确定身份验证方法。如果未指定，则默认为服务账户（IRSA）的 IAM 角色。

- 要使用 EKS 容器组身份，请使用以下任意值：`"true"`、`"True"`、`"TRUE"`、`"t"` 或 `"T"`。
- 要明确使用 IRSA，请设置为以下任意值：`"false"`、`"False"`、`"FALSE"`、`"f"` 或 `"F"`。

preferredAddressType

（可选）指定容器组身份代理端点通信的首选 IP 地址类型。该字段仅在使用 EKS 容器组身份功能时适用，使用服务账户的 IAM 角色时将忽略。值不区分大小写。有效值为：

- `"ipv4"`、`"IPv4"` “或 `"IPV4"` — 强制使用 Pod Identity Agent IPv4 端点
- `"ipv6""IPv6"`、或 `"IPV6"` — 强制使用 Pod Identity Agent IPv6 端点
- 未指定 — 使用 auto 端点选择，先尝试 IPv4 端点，如果 IPv4 失败则回退到 IPv6 端点

objects

包含要挂载密钥的 YAML 声明字符串。我们建议使用 YAML 多行字符串或竖线 (|) 字符。

objectName

必需。指定要获取的密钥或参数的名称。对于 Secrets Manager，这是 `SecretId` 参数，可以是密钥的友好名称，也可以是完整 ARN。对于 SSM Parameter Store，这是参数`Name`的名称，可以是参数的名称或完整 ARN。

objectType

如果不将 Secrets Manager ARN 用于 `objectName`，需要这个操作 可以是 `secretsmanager` 或 `ssmparameter`。

objectAlias

（可选）Amazon EKS 容器组（pod）中密钥的文件名。如果不指定此字段，则 `objectName` 作为文件名显示。

文件权限

（可选）4 位八进制字符串，它指定用于装载密钥的文件权限。如果您未指定此字段，则默认为`"0644"`。

ObjectVersion

（可选）密钥的版本 ID。不推荐，因为每次更新密钥时都必须更新版本 ID。默认情况下，使用最新版本。如果包括 `failoverRegion`，则此字段表示主 `objectVersion`。

objectVersionLabel

（可选）版本的别名。默认为最新版本 AWSCURRENT。有关更多信息，请参阅 the section called “密钥版本”。如果包括 `failoverRegion`，则此字段表示主 `objectVersionLabel`。

JMESPath

（可选）密钥中的键映射到要在 Amazon EKS 中挂载的文件。要使用此字段，密钥值必须采用 JSON 格式。如果使用此字段，必须包含子字段 `path` 和 `objectAlias`。

path

来自密钥值的 JSON 中键值对的密钥。如果该字段包含连字符，请使用单引号对其进行转义，例如：`path: '"hyphenated-path"'`

objectAlias

要挂载到 Amazon EKS 容器组（pod）中的文件名。如果该字段包含连字符，请使用单引号对其进行转义，例如：`objectAlias: '"hyphenated-alias"'`

文件权限

（可选）4 位八进制字符串，它指定用于装载密钥的文件权限。如果未指定此字段，则默认为父对象的文件权限。

failoverObject

（可选）如果您指定此字段，ASCP 会尝试检索主 `objectName` 中指定的密钥和 `failoverObject` `objectName` 子字段中指定的密钥。如果任何一个返回 4xx 错误（例如身份验证问题），ASCP 都不会挂载任何一个密钥。如果成功从主 `objectName` 中检索到密钥，则 ASCP 会挂载该密钥值。如果未成功从主 `objectName` 中检索到密钥，但已成功从失效转移 `objectName` 中检索到密钥，则 ASCP 会挂载该密钥值。如果包含此字段，责必须包含字段 `objectAlias`。有关如何使用此字段的示例，请参阅 故障转移到不同的密钥。

当失效转移密钥不是副本时，通常使用此字段。有关如何指定副本的示例，请参阅 多区域密钥故障转移。

objectName

失效转移密钥的名称或完整 ARN。如果使用 ARN，则 ARN 中的区域必须与字段 `failoverRegion` 匹配。

ObjectVersion

（可选）密钥的版本 ID。必须与主 `objectVersion` 匹配。不推荐，因为每次更新密钥时都必须更新版本 ID。默认情况下，使用最新版本。

objectVersionLabel

（可选）版本的别名。默认为最新版本 AWSCURRENT。有关更多信息，请参阅 the section called “密钥版本”。

创建基本 SecretProviderClass 配置以在您的 Amazon EKS 容器中挂载密钥。

Pod Identity

SecretProviderClass 要在同一 Amazon EKS 集群中使用密钥，请执行以下操作：

```
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: aws-secrets-manager
spec:
  provider: aws
  parameters:
    objects: |
      - objectName: "mySecret"
        objectType: "secretsmanager"
    usePodIdentity: "true"
```

IRSA

```
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: deployment-aws-secrets
spec:
  provider: aws
  parameters:
    objects: |
        - objectName: "MySecret"
          objectType: "secretsmanager"
```

SecretProviderClass 用法

使用这些示例为不同的场景创建 SecretProviderClass 配置。

示例：按名称或 ARN 挂载密钥

此示例说明如何挂载三种不同类型的密钥：

- 由完整 ARN 指定的密钥
- 按名称指定的机密
- 密钥的特定版本

```
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: aws-secrets
spec:
  provider: aws
  parameters:
    objects: |
      - objectName: "arn:aws:secretsmanager:us-east-2:777788889999:secret:MySecret2-
d4e5f6"
      - objectName: "MySecret3"
        objectType: "secretsmanager"
      - objectName: "MySecret4"
        objectType: "secretsmanager"
        objectVersionLabel: "AWSCURRENT"
```

示例：从密钥挂载键值对

此示例说明如何从 JSON 格式的密钥中挂载特定的键值对：

```
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: aws-secrets
spec:
  provider: aws
  parameters:
    objects: |
      - objectName: "arn:aws:secretsmanager:us-east-2:777788889999:secret:MySecret-
a1b2c3"
        jmesPath:
            - path: username
              objectAlias: dbusername
            - path: password
              objectAlias: dbpassword
```

示例：通过文件权限挂载密钥

此示例说明如何使用特定文件权限挂载密钥

```
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
```

```
metadata:
  name: aws-secrets
spec:
  provider: aws
  parameters:
    objects: |
      - objectName: "mySecret"
        objectType: "secretsmanager"
        filePermission: "0600"
        jmesPath:
            - path: username
              objectAlias: dbusername
              filePermission: "0400"
```

示例：失效转移配置示例

这些示例显示了如何为密钥配置故障转移。

多区域密钥故障转移

此示例说明如何为跨多个区域复制的密钥配置自动故障转移：

```
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"
```

故障转移到不同的密钥

此示例说明如何将故障转移配置为不同的密钥（不是副本）：

```
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: aws-secrets
spec:
```

```
  provider: aws
  parameters:
    region: us-east-1
    failoverRegion: us-east-2
    objects: |
      - objectName: "arn:aws:secretsmanager:us-east-1:777788889999:secret:MySecret-
a1b2c3"
        objectAlias: "MyMountedSecret"
        failoverObject:
          - objectName: "arn:aws:secretsmanager:us-
east-2:777788889999:secret:MyFailoverSecret-d4e5f6"
```

其他资源

有关将 ASCP 与 Amazon EKS 结合使用的更多信息，请参阅以下资源：

- 将容器组身份与 Amazon EKS 结合使用
- 使用 AWS 密钥和配置提供程序
- AWS 秘密商店 CSI 驱动程序开启 GitHub

在 AWS Lambda 函数中使用 AWS Secrets Manager 密钥

AWS Lambda 是一项无服务器计算服务，可使您无需预置或管理服务器即可运行代码。Parameter Store（ AWS Systems Manager的一项功能）可提供安全的分层存储，用于配置数据管理和密钥管理。您可以使用 AWS 参数和密钥 Lambda 扩展程序来检索和缓存 Lambda 函数中的 AWS Secrets Manager 密钥，而无需使用开发工具包。有关使用此扩展程序的详细信息，请参阅 Lambda 开发者指南中的在 Lambda 函数中使用 Secrets Manager 密钥。

Secrets Manager 密钥搭配 Lambda

Lambda 开发者指南提供了在 Lambda 函数中使用 Secrets Manager 密钥的全面说明。开始使用：

1. 按照在 Lambda 函数中使用 Secrets Manager 密钥中的 step-by-step教程进行操作，其中包括：
   - 使用你首选的运行时创建 Lambda 函数（Python、Node.js、Java）
   - 将 AWS 参数和密钥 Lambda 扩展添加为层
   - 配置必要的权限
   - 编写代码以从扩展程序中检索机密
   - 测试函数

2. 了解用于配置扩展程序行为的环境变量，包括缓存设置和超时
3. 了解密钥轮换的最佳实践

在 VPC 中使用 Secrets Manager 和 Lambda

如果 Lambda 函数在 VPC 中运行，您需要创建一个 VPC 端点才能调用 Secrets Manager。有关更多信息，请参阅 the section called “VPC 端点（AWS PrivateLink）”。

使用 AWS 参数和密钥 Lambda 扩展

该扩展可以检索 Secrets Manager 密钥和 Parameter Store 参数。有关在扩展中使用参数存储参数的详细信息，请参阅AWS Systems Manager 用户指南中的在 Lambda 函数中使用参数存储参数。

Systems Manager 文档包括：

- 详细说明该扩展程序如何与参数存储配合使用
- 有关将扩展添加到 Lambda 函数的说明
- 用于配置扩展的环境变量
- 用于检索参数的示例命令
- 所有支持的架构和区域 ARNs 的扩展完整列表

使用代 AWS Secrets Manager 理

Secrets Manager 代理是如何运作的

AWS Secrets Manager 代理是一项客户端 HTTP 服务，可帮助您标准化在计算环境中使用 Secrets Manager 中的密钥的方式。您可以将它用于以下服务：

- AWS Lambda
- Amazon Elastic Container Service
- Amazon Elastic Kubernetes Service
- Amazon Elastic Compute Cloud

Secrets Manager Agent 检索密钥并将其缓存在内存中，允许您的应用程序从本地主机获取密钥，而不必直接调用 Secrets Manager。Secrets Manager Agent 只能读取密钥，而无法修改密钥。

> **Important**
>
> Secrets Manager 代理使用您环境中的 AWS 凭证来调用 Secrets Manager。它包括针对服务器端请求伪造（SSRF）的保护，以帮助提高密钥安全性。默认情况下，Secrets Manager Agent 使用后量子 ML-KEM 密钥交换作为优先级最高的密钥交换。

了解 Secrets Manager 代理缓存

Secrets Manager 代理使用内存缓存，该缓存会在 Secrets Manager 代理重启时重置。它会根据以下内容定期刷新缓存的密钥值：

- 默认刷新频率 (TTL) 为 300 秒
- 您可以使用配置文件修改 TTL
- 当你在 TTL 到期后请求密钥时，就会发生刷新

> **Note**
>
> Secrets Manager 代理不包括缓存失效。如果密钥在缓存条目过期之前轮换，则 Secrets Manager 代理可能会返回过时的密钥值。

Secrets Manager 代理返回的密钥值与 `GetSecretValue` 的响应格式相同。密钥值在缓存中未进行加密。

主题

- 构建 Secrets Manager 代理
- 安装 Secrets Manager 代理
- 使用 Secrets Manager 代理检索密钥
- 了解refreshNow参数
- 配置 Secrets Manager 代理
- 可选功能
- 日志记录
- 安全性注意事项

构建 Secrets Manager 代理

在开始之前，请确保您已为您的平台安装标准开发工具和 Rust 工具。

> **Note**
>
> 目前，在 macOS 上启用该`fips`功能的情况下构建代理需要以下解决方法：
>
> - 创建一个名为的环境变量`SDKROOT`，该变量设置为运行结果 `xcrun --show-sdk-path`

RPM-based systems

在基于 RPM 的系统上构建

1. 使用存储库中提供的 `install` 脚本。

   该脚本在启动时生成一个随机的 SSRF 令牌并将其存储在文件 `/var/run/awssmatoken` 中。安装脚本创建的 `awssmatokenreader` 组可以读取该令牌。

2. 要允许您的应用程序读取令牌文件，您需要将应用程序在其下运行的用户账户添加到 `awssmatokenreader` 组。例如，您可以使用以下 usermod 命令授予应用程序读取令牌文件的权限，其中*<APP_USER>*是应用程序在其下运行的用户 ID。

   ```
   sudo usermod -aG awssmatokenreader <APP_USER>
   ```

   安装开发工具

   在基于 RPM 的系统（如 AL2 023）上，安装 “开发工具” 组：

   ```
   sudo yum -y groupinstall "Development Tools"
   ```

3. 安装 Rust

   按照 Rust 文档中安装 Rus t 中的说明进行操作：

   ```
   curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh # Follow the on-
   screen instructions
   . "$HOME/.cargo/env"
   ```

4. 构建代理

使用 cargo build 命令构建 Secrets Manager 代理：

```
cargo build --release
```

您将在 `target/release/aws_secretsmanager_agent` 下找到可执行文件。

Debian-based systems

在基于 Debian 的系统上构建

1. 安装开发工具

   在基于 Debian 的系统（例如 Ubuntu）上，安装构建必需包：

   ```
   sudo apt install build-essential
   ```

2. 安装 Rust

   按照 Rust 文档中安装 Rus t 中的说明进行操作：

   ```
   curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh # Follow the on-
   screen instructions
   . "$HOME/.cargo/env"
   ```

3. 构建代理

   使用 cargo build 命令构建 Secrets Manager 代理：

   ```
   cargo build --release
   ```

   您将在 `target/release/aws_secretsmanager_agent` 下找到可执行文件。

Windows

在 Windows 上构建

1. 设置开发环境

   请按照 Microsoft Windows 文档中的 Set up your dev environment on Windows fo r Rus t

2. 构建代理

   使用 cargo build 命令构建 Secrets Manager 代理：

   ```
   cargo build --release
   ```

   您将在 `target/release/aws_secretsmanager_agent.exe` 下找到可执行文件。

Cross-compile natively

本机交叉编译

1. 安装交叉编译工具

   在 mingw-w64 包可用的发行版（例如 Ubuntu）上，请安装交叉编译工具链：

   ```
   # Install the cross compile tool chain
   sudo add-apt-repository universe
   sudo apt install -y mingw-w64
   ```

2. 添加 Rust 构建目标

   安装 Windows GNU 编译目标：

   ```
   rustup target add x86_64-pc-windows-gnu
   ```

3. 为 Windows 构建

   交叉编译适用于 Windows 的代理：

   ```
   cargo build --release --target x86_64-pc-windows-gnu
   ```

   您将在 `target/x86_64-pc-windows-gnu/release/aws_secretsmanager_agent.exe` 处找到可执行文件。

Cross compile with Rust cross

使用 Rust 交叉进行交叉编译

如果系统本身没有交叉编译工具，则可以使用 Rust 交叉项目。有关更多信息，请参阅 c https://github.com/cross-rs/ross。

> **Important**
>
> 我们建议为构建环境提供 32GB 的磁盘空间。

1. 设置 Docker

   安装和配置 Docker：

   ```
   # Install and start docker
   sudo yum -y install docker
   sudo systemctl start docker
   sudo systemctl enable docker # Make docker start after reboot
   ```

2. 配置 Docker 权限

   将您的用户添加到 docker 群组：

   ```
   # Give ourselves permission to run the docker images without sudo
   sudo usermod -aG docker $USER
   newgrp docker
   ```

3. 为 Windows 构建

   安装 cross 并生成可执行文件：

   ```
   # Install cross and cross compile the executable
   cargo install cross
   cross build --release --target x86_64-pc-windows-gnu
   ```

安装 Secrets Manager 代理

从以下安装选项中选择您的计算环境。

Amazon EC2

在亚马逊上安装 Secrets Manager 代理 EC2

1. 导航到配置目录

   切换到配置目录：

   ```
   cd aws_secretsmanager_agent/configuration
   ```

2. 运行安装脚本

   运行存储库中提供的 install `install` 脚本。

   该脚本在启动时生成一个随机的 SSRF 令牌并将其存储在文件 `/var/run/awssmatoken` 中。安装脚本创建的 `awssmatokenreader` 组可以读取该令牌。

3. 配置应用程序权限

   将运行应用程序的用户帐户添加到`awssmatokenreader`群组中：

   ```
   sudo usermod -aG awssmatokenreader APP_USER
   ```

   *APP_USER*替换为运行应用程序时使用的用户 ID。

Container Sidecar

您可以使用 Docker 将 Secrets Manager 代理作为附加容器与应用程序一起运行。然后，您的应用程序可以从 Secrets Manager 代理提供的本地 HTTP 服务器检索密钥。有关 Docker 的信息，请参阅 Docker 文档。

为 Secrets Manager 代理创建附加容器

1. 创建代理 Dockerfile

   为 Secrets Manager 代理附加容器创建 Dockerfile：

   ```
   # Use the latest Debian image as the base
   FROM debian:latest

   # Set the working directory inside the container
   WORKDIR /app
   ```

```
# Copy the Secrets Manager Agent binary to the container
COPY secrets-manager-agent .

# Install any necessary dependencies
RUN apt-get update && apt-get install -y ca-certificates

# Set the entry point to run the Secrets Manager Agent binary
ENTRYPOINT ["./secrets-manager-agent"]
```

2. 创建应用程序 Dockerfile

   为您的客户端应用程序创建一个 Dockerfile。

3. 创建 Docker Compose 文件

   创建 Docker Compose 文件，使用共享网络接口运行两个容器：

> **Important**
>
> 您必须加载 AWS 凭证和 SSRF 令牌，应用程序才能使用 Secrets Manager 代理。对于 Amazon EKS 和 Amazon ECS，请参阅以下内容：
>
> - 在 Amazon EKS 用户指南中@@ 管理访问权限
> - 亚马逊 ECS 开发者指南中的 Amazon ECS 任务 IAM 角色

```
version: '3'
services:
    client-application:
    container_name: client-application
    build:
        context: .
        dockerfile: Dockerfile.client
    command: tail -f /dev/null  # Keep the container running


    secrets-manager-agent:
    container_name: secrets-manager-agent
    build:
        context: .
        dockerfile: Dockerfile.agent
```

```
    network_mode: "container:client-application"  # Attach to the client-
application container's network
    depends_on:
        - client-application
```

4. 复制座席班次

   将 `secrets-manager-agent` 二进制文件复制到包含您的 Dockerfile 和 Docker Compose 文件的同一个目录中。

5. 构建并运行容器

   使用 Docker Compose 构建并运行容器：

```
docker-compose up --build
```

6. 后续步骤

   您现在可使用 Secrets Manager 代理从您的客户端容器中检索密钥。有关更多信息，请参阅 the section called “使用 Secrets Manager 代理检索密钥”。

Lambda

您可以将 S ecrets Manager 代理打包为 Lambda 扩展。然后，您可以将其作为层添加到 Lambda 函数中，并从 Lambda 函数调用 Secrets Manager 代理来获取密钥。

以下说明说明如何使用`secrets-manager-agent-extension.sh`中的示例脚本将 Secret MyTests Manager 代理作为 Lambda 扩展进行安装，https://github.com/aws/aws-secretsmanager-agent从而获取名为的密钥。

为 Secrets Manager 代理创建 Lambda 扩展

1. Package 代理层

   从 Secrets Manager 代理代码包的根目录运行以下命令：

```
AWS_ACCOUNT_ID=AWS_ACCOUNT_ID
LAMBDA_ARN=LAMBDA_ARN

# Build the release binary
cargo build --release --target=x86_64-unknown-linux-gnu

# Copy the release binary into the `bin` folder
```

```
mkdir -p ./bin
cp ./target/x86_64-unknown-linux-gnu/release/aws_secretsmanager_agent ./bin/
secrets-manager-agent

# Copy the `secrets-manager-agent-extension.sh` example script into the
 `extensions` folder.
mkdir -p ./extensions
cp aws_secretsmanager_agent/examples/example-lambda-extension/secrets-manager-
agent-extension.sh ./extensions

# Zip the extension shell script and the binary
zip secrets-manager-agent-extension.zip bin/* extensions/*

# Publish the layer version
LAYER_VERSION_ARN=$(aws lambda publish-layer-version \
    --layer-name secrets-manager-agent-extension \
    --zip-file "fileb://secrets-manager-agent-extension.zip" | jq -r
 '.LayerVersionArn')
```

2. 配置 SSRF 令牌

   代理的默认配置会自动将 SSRF 令牌设置为在预设变量`AWS_SESSION_TOKEN`或`AWS_CONTAINER_AUTHORIZATION_TOKEN`环境变量中设置的值（后一个变量适用于启用的 Lambda 函数）。 SnapStart 或者，您可以改用您的 Lambda 函数的任意值定义`AWS_TOKEN`环境变量，因为该变量优先于其他两个变量。如果您选择使用`AWS_TOKEN`环境变量，则必须通过`lambda:UpdateFunctionConfiguration`调用来设置该环境变量。

3. 将图层附加到函数

   将层版本附加到您的 Lambda 函数：

```
# Attach the layer version to the Lambda function
aws lambda update-function-configuration \
    --function-name $LAMBDA_ARN \
    --layers "$LAYER_VERSION_ARN"
```

4. 更新函数代码

   更新您的 Lambda 函数以`http://localhost:2773/secretsmanager/get?secretId=MyTest`使用`X-Aws-codes-Secrets-Token`标头值设置为来自上述环境变量之一的 SSRF 令牌的值进行查询，以检索密钥。务必在应用程序代码中实现重试逻辑，以适应 Lambda 扩展初始化和注册中的延迟。

5. 测试此函数

   调用 Lambda 函数以验证是否已正确获取密钥。

使用 Secrets Manager 代理检索密钥

要检索密钥，请使用密钥名称或 ARN 作为查询参数调用本地 Secrets Manager 代理端点。默认情况下，Secrets Manager 代理会检索密钥的 `AWSCURRENT` 版本。要检索其他版本，请使用 versionStage 或 versionId 参数。

> **Important**
>
> 为了帮助保护 Secrets Manager 代理，您必须在每个请求中包含 SSRF 令牌标头：`X-Aws-Parameters-Secrets-Token`。Secrets Manager 代理会拒绝没有此标头或具有无效 SSRF 令牌的请求。您可以在 the section called “配置选项” 中自定义 SSRF 标头名称。

所需的权限

Secrets Manager 代理使用 AWS 适用于 Rust 的 SDK，该 SDK 使用AWS 凭证提供程序链。这些 IAM 凭证的身份决定了 Secrets Manager 代理检索密钥的权限。

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

有关权限的更多信息，请参阅 the section called “权限参考”。

> **Important**
>
> 将密钥值拉入 Secrets Manager 代理后，任何有权访问计算环境和 SSRF 令牌的用户都可以从 Secrets Manager 代理缓存中访问密钥。有关更多信息，请参阅 the section called “安全性注意事项”。

示例请求

curl

Example 示例-使用 curl 获取秘密

以下 curl 示例展示了如何从 Secrets Manager 代理获取密钥。该示例依赖于文件中存在的 SSRF，该文件是安装脚本存储示例的位置。

```
curl -v -H \\
    "X-Aws-Parameters-Secrets-Token: $(/var/run/awssmatoken)" \\
    'http://localhost:2773/secretsmanager/get?secretId=YOUR_SECRET_ID' \\
    echo
```

Python

Example 示例 — 使用 Python 获取秘密

以下 Python 示例展示了如何从 Secrets Manager 代理获取密钥。该示例依赖于文件中存在的 SSRF，该文件是安装脚本存储示例的位置。

```
import requests
import json

# Function that fetches the secret from Secrets Manager Agent for the provided
 secret id.
def get_secret():
    # Construct the URL for the GET request
    url = f"http://localhost:2773/secretsmanager/get?secretId=YOUR_SECRET_ID"

    # Get the SSRF token from the token file
    with open('/var/run/awssmatoken') as fp:
        token = fp.read()

    headers = {
        "X-Aws-Parameters-Secrets-Token": token.strip()
    }

    try:
        # Send the GET request with headers
        response = requests.get(url, headers=headers)

        # Check if the request was successful
```

```
        if response.status_code == 200:
            # Return the secret value
            return response.text
        else:
            # Handle error cases
            raise Exception(f"Status code {response.status_code} - {response.text}")

    except Exception as e:
        # Handle network errors
        raise Exception(f"Error: {e}")
```

了解refreshNow参数

Secrets Manager 代理使用内存缓存来存储密钥值，并且会定期刷新密钥值。默认情况下，当您在生存时间 (TTL) 到期后请求密钥时，通常每 300 秒刷新一次。但是，这种方法有时会导致密钥值过时，特别是如果密钥在缓存条目过期之前轮换。

为了解决此限制，Secrets Manager Agent 支持在 URL `refreshNow` 中调用的参数。您可以使用此参数强制立即刷新密钥的值，绕过缓存并确保您拥有最多的 up-to-date信息。

默认行为（没有**refreshNow**）

- 在 TTL 到期之前使用缓存值
- 仅在 TTL 之后才刷新密钥（默认为 300 秒）
- 如果密钥在缓存过期之前轮换，则可能会返回陈旧值

行为与 **refreshNow=true**

- 完全绕过缓存
- 直接从 Secrets Manager 中检索最新的密钥值
- 使用新值更新缓存并重置 TTL
- 确保您始终获得最新的秘密值

强制刷新密钥值

> **Important**
>
> `refreshNow` 的默认值为 `false`。设置为时`true`，它将覆盖 Secrets Manager 代理配置文件中指定的 TTL，并对 Secrets Manager 进行 API 调用。

curl

Example 示例 — 使用 curl 强制刷新密钥

以下 curl 示例展示了如何强制 Secrets Manager 代理刷新密钥。该示例依赖于文件中存在的 SSRF，该文件是安装脚本存储示例的位置。

```
curl -v -H \\
"X-Aws-Parameters-Secrets-Token: $(/var/run/awssmatoken)" \\
'http://localhost:2773/secretsmanager/get?secretId=YOUR_SECRET_ID&refreshNow=true' \
\
echo
```

Python

Example 示例-使用 Python 强制刷新密钥

以下 Python 示例展示了如何从 Secrets Manager 代理获取密钥。该示例依赖于文件中存在的 SSRF，该文件是安装脚本存储示例的位置。

```
import requests
import json

# Function that fetches the secret from Secrets Manager Agent for the provided
 secret id.
def get_secret():
    # Construct the URL for the GET request
    url = f"http://localhost:2773/secretsmanager/get?
secretId=YOUR_SECRET_ID&refreshNow=true"

    # Get the SSRF token from the token file
    with open('/var/run/awssmatoken') as fp:
        token = fp.read()

    headers = {
        "X-Aws-Parameters-Secrets-Token": token.strip()
    }

    try:
        # Send the GET request with headers
        response = requests.get(url, headers=headers)

        # Check if the request was successful
```

```
        if response.status_code == 200:
            # Return the secret value
            return response.text
        else:
            # Handle error cases
            raise Exception(f"Status code {response.status_code} - {response.text}")

    except Exception as e:
        # Handle network errors
        raise Exception(f"Error: {e}")
```

配置 Secrets Manager 代理

要更改 Secrets Manager 代理的配置，请创建一个 TOML 配置文件，然后调用 `./aws_secretsmanager_agent --config config.toml`。

配置选项

`log_level`

Secrets Manager 代理日志中报告的详细程度：DEBUG、INFO、WARN、ERROR 或 NONE。默认值为 INFO。

`log_to_file`

是登录文件还是 stdout/stderr：或。`true` `false`默认值为 `true`。

`http_port`

本地 HTTP 服务器的端口，范围在 1024 到 65535 之间。默认值为 2773。

`region`

用于请求的 AWS 区域。如果未指定区域，则 Secrets Manager 代理会根据 SDK 确定区域。有关更多信息，请参阅《AWS SDK for Rust 开发人员指南》中的 Specify your credentials and default Region。

`ttl_seconds`

缓存项目的 TTL（以秒为单位），范围在 0 到 3600 之间。默认值为 300。0 表示没有缓存。

`cache_size`

缓存中可以存储的最大密钥数量，范围在 1 到 1000 之间。默认值为 1000。

ssrf_headers

Secrets Manager 代理检查 SSRF 令牌的标头名称列表。默认值为 “X-Aws-Parameters-Token”。X-Vault-Token

ssrf_env_variables

Secrets Manager 代理按顺序检查 SSRF 令牌的环境变量名称列表。环境变量可以包含令牌或对令牌文件的引用，如下所示：`AWS_TOKEN=file:///var/run/awssmatoken`。默认为 “AWS_TOKEN、 AWS_SESSION _TOKEN、_AUTHORIZATION AWS_CONTAINER _TOKEN”。

path_prefix

用于确定请求是否为基于路径的请求的 URI 前缀。默认值为“/v1/”。

max_conn

Secrets Manager 代理允许的来自 HTTP 客户端的最大连接数，范围在 1 到 1000 之间。默认值为 800。

可选功能

通过将`--features`标志传递给，可以构建 Secrets Manager Agent，使用可选功能`cargo build`。可用功能有：

生成功能

prefer-post-quantum

生成优先级最高`X25519MLKEM768`的密钥交换算法。否则，它可用，但不是最高优先级。`X25519MLKEM768`是一种混合 post-quantum-secure密钥交换算法。

fips

将代理使用的密码套件限制为 FIPS 批准的密码。

日志记录

本地日志

Secrets Manager 代理会在本地将错误记录到 stdout/stderr 中`logs/secrets_manager_agent.log`，具体取决于配置变量。`log_to_file`当应用程序调用 Secrets Manager 代理来获取密钥时，这些调用会显示在本地日志中。它们不会显示在 CloudTrail 日志中。

日志轮换

当文件达到 10 MB 时，Secrets Manager 代理会创建一个新的日志文件，并且总共最多存储五个日志文件。

AWS 服务日志

日志不会转到 Secrets Manager CloudWatch。 CloudTrail从 Secrets Manager 代理获取密钥的请求不会出现在这些日志中。当 Secrets Manager 代理调用 Secrets Manager 来获取密钥时，该调用会记录在 CloudTrail Secrets Manager 代理字符串中`aws-secrets-manager-agent`。

您可以在中配置日志记录选项the section called “配置选项”。

安全性注意事项

信任域

对于代理架构，信任域是代理终端节点和 SSRF 令牌可访问的位置，通常是整个主机。为了保持相同的安全状况，Secrets Manager 代理的信任域应与 Secrets Manager 凭证可用的域相匹配。例如，在亚马逊 EC2 上，使用 Amazon 角色时，Secrets Manager 代理的信任域将与凭证的域相同 EC2。

> **Important**
>
> 如果应用程序具有安全意识并且尚未使用将 Secrets Manager 凭证锁定到应用程序的代理解决方案，则应考虑使用特定于语言 AWS SDKs 或缓存的解决方案。有关更多信息，请参阅获取密钥。

使用 C++ AWS 软件开发工具包获取 Secrets Manager 密钥值

对于 C++ 应用程序，请使用GetSecretValue或直接调用 SDK BatchGetSecretValue。

以下代码示例展示了如何获取 Secrets Manager 密钥值。

所需权限：`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();
}
```

使用 JavaScript AWS SDK 获取 Secrets Manager 密钥值

对于 JavaScript 应用程序，请使用`getSecretValue`或直接调用 SDK `batchGetSecretValue`。

以下代码示例展示了如何获取 Secrets Manager 密钥值。

所需权限：`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;
  }
};
```

使用 Kotlin AWS SDK 获取 Secrets Manager 的密钥值

对于 Kotlin 应用程序，请使用GetSecretValue或BatchGetSecretValue直接调用 SDK。

以下代码示例展示了如何获取 Secrets Manager 密钥值。

所需权限：`secretsmanager:GetSecretValue`

```
suspend fun getValue(secretName: String?) {
    val valueRequest =
        GetSecretValueRequest {
            secretId = secretName
        }

    SecretsManagerClient.fromEnvironment { region = "us-east-1" }.use { secretsClient -
>
        val response = secretsClient.getSecretValue(valueRequest)
        val secret = response.secretString
        println("The secret value is $secret")
    }
}
```

使用 PHP AWS 开发工具包获取 Secrets Manager 密钥值

对于 PHP 应用程序，请直接使用 `GetSecretValue` 或 `BatchGetSecretValue` 调用 SDK。

以下代码示例展示了如何获取 Secrets Manager 密钥值。

所需权限：`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
```

使用 Ruby AWS SDK 获取 Secrets Manager 密钥值

对于 Ruby 应用程序，请直接使用 get_secret_value 或 batch_get_secret_value 调用 SDK。

以下代码示例展示了如何获取 Secrets Manager 密钥值。

所需权限：`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
```

使用获取密钥值 AWS CLI

所需权限：`secretsmanager:GetSecretValue`

Example 检索密钥的加密密钥值

以下 `get-secret-value` 示例获取当前密钥值。

```
aws secretsmanager get-secret-value \
    --secret-id MyTestSecret
```

Example 检索之前的密钥值

以下 `get-secret-value` 示例获取之前的密钥值。

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

使用批量获取一组机密 AWS CLI

所需权限：

- `secretsmanager:BatchGetSecretValue`
- 对要检索的每个密钥拥有 `secretsmanager:GetSecretValue` 权限。
- 如果您使用筛选器，则还必须拥有 `secretsmanager:ListSecrets`。

有关权限策略的示例，请参阅 the section called “示例：批量检索一组密钥值的权限”。

> **Important**
> 如果您的 VPCE 策略拒绝在您正在检索的群组中检索单个秘密的权限，则 `BatchGetSecretValue` 不会返回任何秘密值，并且会返回错误。

Example 检索按名称列出的一组秘密的秘密值

以下 `batch-get-secret-value` 示例获取三个秘密的秘密值。

```
aws secretsmanager batch-get-secret-value \
          --secret-id-list MySecret1 MySecret2 MySecret3
```

Example 检索筛选器选择的一组秘密的秘密值

以下 `batch-get-secret-value` 示例获取具有名为“Test”的标签的秘密的秘密值。

```
aws secretsmanager batch-get-secret-value \
          --filters Key="tag-key",Values="Test"
```

使用 AWS 控制台获取密钥值

检索密钥（控制台）

1. 打开 Secrets Manager 控制台，网址为https://console.aws.amazon.com/secretsmanager/。
2. 在秘密列表中，选择要检索的秘密。
3. 在密钥值部分中，选择检索密钥值。

   Secrets Manager 显示秘密的当前版本 (`AWSCURRENT`)。要查看秘密的其他版本，例如 `AWSPREVIOUS` 或带有自定义标签的版本，请使用 the section called “AWS CLI”。

在中使用 AWS Secrets Manager 秘密 AWS Batch

AWS Batch 可帮助您在上运行批量计算工作负载 AWS Cloud。使用 AWS Batch，您可以将敏感数据注入作业，方法是将敏感数据存储在 AWS Secrets Manager 机密中，然后在作业定义中引用它们。有关更多信息，请参阅使用 Secrets Manager 指定敏感数据。

在 AWS CloudFormation 资源中获取 AWS Secrets Manager 秘密

使用 AWS CloudFormation，您可以检索密钥以在其他 AWS CloudFormation 资源中使用。常见场景是首先使用 Secret Manager 生成的密码创建密钥，然后从该密钥中检索用户名和密码，以用作新数据库的凭证。有关使用创建密钥的信息 AWS CloudFormation，请参阅AWS CloudFormation。

要检索 AWS CloudFormation 模板中的密钥，请使用动态引用。创建堆栈时，动态引用会将密钥值提取到 AWS CloudFormation 资源中，因此您不必对机密信息进行硬编码。相反，您可以通过名称或 ARN 来引用密钥。您可以在任何资源属性中使用对密钥的动态引用。您不能在资源元数据（例如 `AWS::CloudFormation::Init`）中使用对密钥的动态引用，因为那样会使密钥值在控制台中可见。

密钥的动态引用模式如下：

```
{{resolve:secretsmanager:secret-id:SecretString:json-key:version-stage:version-id}}
```

secret-id

密钥的名称或 ARN。要访问您 AWS 账户中的密钥，您可以使用该密钥名称。要访问其他 AWS 账户中的密钥，请使用该密钥的 ARN。

json-key（可选）

要检索其值的键值对的键名称。如果未指定`json-key`，则 AWS CloudFormation 检索整个机密文本。此分段不得包含冒号字符（：）。

version-stage（可选）

要使用的密钥的版本。Secrets Manager 在轮换过程中使用暂存标注来跟踪不同的版本。如果您使用 `version-stage`，则不要指定 `version-id`。如果您既未指定 `version-stage`，也未指定 `version-id`，则原定设置将为 `AWSCURRENT` 版本。此分段不得包含冒号字符（：）。

version-id（可选）

要使用的密钥版本的唯一标识符。如果指定 `version-id`，则不要指定 `version-stage`。如果您既未指定 `version-stage`，也未指定 `version-id`，则原定设置将为 `AWSCURRENT` 版本。此分段不得包含冒号字符（：）。

有关更多信息，请参阅使用动态引用指定 Secrets Manager 秘密。

> **Note**
>
> 不要使用反斜杠(\)作为最终值来创建动态引用。 AWS CloudFormation 无法解析这些引用，这会导致资源故障。

在 GitHub 工作中使用 AWS Secrets Manager 秘密

要在 GitHub 作业中使用密钥，您可以使用 GitHub 操作从 AWS Secrets Manager 中检索密钥并将其作为屏蔽的环境变量添加到 GitHub 工作流程中。有关 GitHub 操作的更多信息，请参阅了解GitHub 文档中的 GitHub 操作。

当你向 GitHub 环境中添加密钥时，该密钥可用于 GitHub 工作中的所有其他步骤。按照操作安全强化中的指导进行 GitHub 操作，以帮助防止环境中的密钥被滥用。

您可以将密钥值中的整个字符串设置为环境变量值，或者如果字符串为 JSON，则可以解析 JSON，以为每个 JSON 键值对设置单独的环境变量。如果密钥值是二进制，则操作会将密钥值转换为字符串。

若要查看从密钥创建的环境变量，请启用调试日志记录。有关更多信息，请参阅GitHub 文档中的启用调试日志记录。

要使用根据您的密钥创建的环境变量，请参阅GitHub 文档中的环境变量。

先决条件

要使用此操作，您首先需要配置 AWS 凭证，然后使用`configure-aws-credentials`步骤 AWS 区域 在您的 GitHub 环境中进行设置。按照配置 AWS 凭证操作中的说明进行 GitHub 操作，以便使用 GitHub OIDC 提供程序直接担任角色。这让您能够使用短期凭证，避免在 Secrets Manager 之外存储额外的访问密钥。

操作承担的 IAM 角色必须具有下列权限：

- 您要检索的密钥的 `GetSecretValue` 权限。
- 所有密钥的 `ListSecrets` 权限。
- （可选）`Decrypt`上 KMS key 是否使用加密密钥 客户托管式密钥。

有关更多信息，请参阅 the section called “身份验证和访问控制”。

使用量

若要使用该操作，请在工作流程中添加一个使用以下语法的步骤。

```
- 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
```

参数

`secret-ids`

密钥 ARN、名称和名称前缀。

若要设置环境变量名称，请在密钥 ID 前输入该名称，然后输入逗号。例如，`ENV_VAR_1, secretId` 从密钥 `secretId` 中创建名为 ENV_VAR_1 的环境变量。环境变量的名称可以包含小写字母、数字和下划线。

若要使用前缀，请输入至少三个字符，然后输入星号。例如，`dev*` 匹配名称以 dev 开头的所有密钥。最多可以检索 100 个匹配密钥。如果您设置了变量名称，并且前缀与多个密钥匹配，则操作将失败。

`name-transformation`

默认情况下，该步骤从密钥名称创建每个环境变量名称，并转换为仅包含大写字母、数字和下划线，防止名称以数字开头。对于名称中的字母，您可以通过 `lowercase` 将步骤配置为使用小写字母，或者通过 none 配置为不改变字母的大小写。默认值为 `uppercase`。

`parse-json-secrets`

（可选）默认情况下，该操作将环境变量值设置为密钥值中的整个 JSON 字符串。将 `parse-json-secrets` 设置为 `true`，以为 JSON 中的每个键值对创建环境变量。

请注意，如果 JSON 使用区分大小写的密钥（例如“name”和“Name”），则该操作将出现重复的名称冲突。在这种情况下，请将 `parse-json-secrets` 设置为 `false` 并单独解析 JSON 密钥值。

环境变量命名

操作创建的环境变量的名称与它们来源的密钥相同。环境变量的命名要求比密钥的命名要求更严格，因此操作会转换密钥名称，以满足这些要求。例如，该操作将把小写字母转换为大写字母。如果解析密钥的 JSON，则环境变量名称将同时包含密钥名称和 JSON 键名称，例如 `MYSECRET_KEYNAME`。您可以将操作配置为不转换小写字母。

如果两个环境变量具有相同的名称，则操作将失败。在这种情况下，您必须将要用于环境变量的名称指定为别名。

名称可能冲突的示例：

- 名为 “” MySecret 的密钥和名为 “mysecret” 的密钥都将成为名为 “MYSECRET” 的环境变量。
- 名为“Secret_keyname”的密钥以及名为“Secret”且具有“keyname”键的 JSON 解析密钥都将成为名为“SECRET_KEYNAME”的环境变量。

您可以通过指定别名来设置环境变量名称，如以下示例所示，它会创建一个名为 `ENV_VAR_NAME` 的变量。

```
secret-ids: |
  ENV_VAR_NAME, secretId2
```

空白别名

- 如果您设置 `parse-json-secrets: true` 并输入空白别名，后跟逗号，然后是密钥 ID，则操作会将环境变量命名为与解析后的 JSON 键相同。变量名称不包含密钥名称。

  如果密钥不包含有效的 JSON，则操作会创建一个环境变量并将其命名为与密钥名称相同。
- 如果您设置 `parse-json-secrets: false` 并输入空白别名，后跟逗号，然后是密钥 ID，则操作将命名环境变量，就好像您没有指定别名一样。

以下示例显示了一个空白别名。

```
,secret2
```

示例

Example 1 按名称和 ARN 获取密钥

下列示例为按名称和 ARN 标识的密钥创建环境变量。

```
- 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
```

已创建的环境变量：

```
EXAMPLESECRETNAME: secretValue1
TEST1: secretValue2
_0_TEST_SECRET: secretValue3
_PROD_EXAMPLE_SECRET: secretValue4
SECRET_ALIAS_1: secretValue5
SECRET_ALIAS_2: secretValue6
SECRET2: secretValue7
```

Example 2 获取所有以前缀开头的密钥

以下示例为所有名称以开头的密钥创建环境变量*beta*。

```
- name: Get Secret Names by Prefix
  uses: 2
  with:
    secret-ids: |
      beta*    # Retrieves all secrets that start with 'beta'
```

已创建的环境变量：

```
BETASECRETNAME: secretValue1
BETATEST: secretValue2
```

```
BETA_NEWSECRET: secretValue3
```

Example 3 在密钥中解析 JSON

下列示例通过解析密钥中的 JSON 来创建环境变量。

```
- 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
```

密钥 `test/secret` 具有下列密钥值。

```
{
  "api_user": "user",
  "api_key": "key",
  "config": {
    "active": "true"
  }
}
```

密钥 `secret2` 具有下列密钥值。

```
{
  "myusername": "alejandro_rosalez",
  "mypassword": "EXAMPLE_PASSWORD"
}
```

已创建的环境变量：

```
TEST_SECRET_API_USER: "user"
TEST_SECRET_API_KEY: "key"
TEST_SECRET_CONFIG_ACTIVE: "true"
MYUSERNAME: "alejandro_rosalez"
MYPASSWORD: "EXAMPLE_PASSWORD"
```

Example 4 对环境变量名称使用小写字母

以下示例将创建一个具有小写名称的环境变量。

```
- name: Get secrets
  uses: aws-actions/aws-secretsmanager-get-secrets@v2
  with:
    secret-ids: exampleSecretName
    name-transformation: lowercase
```

已创建的环境变量：

```
examplesecretname: secretValue
```

用 AWS Secrets Manager 于 GitLab

AWS Secrets Manager 与。集成 GitLab。你可以利用 Secrets Manager 的密钥来保护你的 GitLab凭证，这样它们就不会再被硬编码了。 GitLab相反，当你的应用程序在 GitLab CI/CD 管GitLab 道中运行作业时，Runner 会从 Secrets Manager 中检索这些机密。

要使用此集成，您需要在 IAM 中创建一个 OpenID Connect (OIDC) 身份提供商 AWS Identity and Access Management 和一个 IAM 角色。这允许 GitLab Runner 访问你的 Secrets Manager 密钥。有关 C GitLab I/CD 和 OIDC 的更多信息，请参阅文档。GitLab

注意事项

如果您使用的是非公共 GitLab 实例，则无法使用此 Secrets Manager 集成。相反，请参阅非公共实例的GitLab 文档。

先决条件

要将 Secrets Manager 与集成 GitLab，请完成以下先决条件：

1. 创建密 AWS Secrets Manager 钥

   你需要一个 Secrets Manager 密钥，该密钥将在你的 GitLab 工作中检索，无需对这些凭据进行硬编码。在配置 GitLab 管道时，你需要 Secrets Manager 的密钥 ID。请参阅创建密 AWS Secrets Manager 钥了解更多信息。
2. 在 IAM 控制台中创建 GitLab 您的 OIDC 提供商。

   在此步骤中，您将在 IAM 控制台中创建 GitLab 您的 OIDC 提供商。有关更多信息，请参阅创建 OpenID Connect (OIDC) 身份提供商和文档。GitLab

   在 IAM 控制台中创建 OIDC 提供商时，请使用以下配置：

a. 将设置`provider URL`为您的 GitLab 实例。例如 **`gitlab.example.com`**。

b. 将`audience`或设置`aud`为**`sts.amazonaws.com`**。

3. 创建 IAM 角色和策略

   您需要创建 IAM 角色和策略。此角色由 with GitLab AWS Security Token Service (STS) 担任。有关更多信息，请参阅使用自定义信任策略创建角色。

   a. 在 IAM 控制台中，创建 IAM 角色时使用以下设置：

      - 将 `Trusted entity type` 设置为 **`Web identity`**。
      - 将 `Group` 设置为 **`your GitLab group`**。
      - 设置`Identity provider`为您在步骤 2 中使用的提供商 URL（GitLab 实例）。
      - 设置`Audience`为您在步骤 2 中使用的受众群体。

   b. 您还需要创建一个 IAM 策略以允许 GitLab 访问 AWS Secrets Manager。您可以将此策略添加到您的信任策略中。有关更多信息，请参阅创建 IAM 策略。

AWS Secrets Manager 与集成 GitLab

完成先决条件后，您可以配置为使用 Sec GitLab rets Manager 来保护您的凭证。

将 GitLab 管道配置为使用 Secrets Manager

您需要使用以下信息更新 GitLab CI/CD 配置文件：

- 代币的受众设置为 STS。
- Secrets Manager 的密钥 ID。
- 您希望 GitLab Runner 在 GitLab管道中执行任务时扮演的 IAM 角色。
- 密钥的存储位置。 AWS 区域

GitLab 从 Secrets Manager 中获取密钥并将该值存储在临时文件中。此文件的路径存储在 CI/CD 变量中，类似于文件类型 CI/CD 变量。

以下是 GitLab CI/CD 配置文件的 YAML 文件片段：

```
variables:
```

```
  AWS_REGION: us-east-1
  AWS_ROLE_ARN: 'arn:aws:iam::111122223333:role/gitlab-role'
job:
  id_tokens:
    AWS_ID_TOKEN:
      aud: 'sts.amazonaws.com'
  secrets:
    DATABASE_PASSWORD:
      aws_secrets_manager:
        secret_id: "arn:aws:secretsmanager:us-east-1:111122223333:secret:secret-name"
```

有关更多信息，请参阅 S GitLabecrets Manager 集成文档。

或者，您也可以在中测试 OIDC 配置。 GitLab有关更多信息，请参阅测试 OIDC 配置的GitLab 文档。

故障排除

以下内容可以帮助您解决在将 Secrets Manager 与集成时可能遇到的常见问题 GitLab。

GitLab 管道问题

如果您遇到 GitLab 管道问题，请确保以下几点：

- 您的 YAML 文件格式正确。有关更多信息，请参阅 GitLab 文档。
- 您的 GitLab 管道扮演了正确的角色，拥有相应的权限，并且可以访问正确的 AWS Secrets Manager 密钥。

其他资源

以下资源可以帮助您解决 GitLab 和的问题 AWS Secrets Manager：

- GitLab OIDC 疑难解答
- 调试 GitLab CI/CD 管道
- 故障排除

在中使用 AWS Secrets Manager 秘密 AWS IoT Greengrass

AWS IoT Greengrass 是将云功能扩展到本地设备的软件。这使得设备可以更靠近信息源来收集和分析数据，自主响应本地事件，同时在本地网络上彼此安全地通信。

AWS IoT Greengrass 允许您使用 AWS IoT Greengrass 设备上的服务和应用程序进行身份验证，而无需对密码、令牌或其他机密进行硬编码。您可以使用在云端安全 AWS Secrets Manager 地存储和管理您的密钥。 AWS IoT Greengrass 将 Secrets Manager 扩展到 AWS IoT Greengrass 核心设备，因此您的连接器和 Lambda 函数可以使用本地密钥与服务和应用程序进行交互。

要将密钥集成到 AWS IoT Greengrass 群组中，您需要创建一个引用 Secrets Manager 密钥的群组资源。此密钥资源使用关联 ARN 引用云密钥。要了解如何创建、管理和使用秘密资源，请参阅 AWS IoT 开发者指南中的使用机密资源。

要将密钥部署到 AWS IoT Greengrass 核心，请参阅将密钥部署到 AWS IoT Greengrass 核心。

在参数存储中使用 AWS Secrets Manager 密钥

AWS Systems Manager Parameter Store 为配置数据管理和密钥管理提供安全的分层存储。也可以将密码、数据库字符串和许可证代码等数据存储为参数值。不过，Parameter Store 不会为存储的密钥提供自动轮换服务。相反，Parameter Store 允许您在 Secrets Manager 中存储密钥，然后以 Parameter Store 参数形式引用该密钥。

使用 Secrets Manager 配置 Parameter Store 时，`secret-id` Parameter Store 需要在名称字符串之前使用正斜杠 (/)。

有关更多信息，请参阅AWS Systems Manager 用户指南中的通过参数存储参数引用 AWS Secrets Manager 密钥。

轮换 AWS Secrets Manager 秘密

Rotation 是定期更新密钥的过程。当轮换密钥时，会同时更新密钥和数据库或服务中的凭据。在 Secrets Manager 中，您可以为密钥设置自动轮换。轮换有两种形式：

- 托管轮换 – 对于大多数托管密钥，可以使用托管轮换，其中服务为您配置和管理轮换。托管轮换不使用 Lambda 函数。
- the section called “通过 Lambda 函数进行轮换” – 对于其他类型的密钥，Secrets Manager 轮换使用 Lambda 函数来更新密钥和数据库或服务。

AWS Secrets Manager 密钥的托管轮换

部分服务提供托管轮换，即服务为您配置和管理轮换。使用托管轮换，您无需使用 AWS Lambda 函数来更新数据库中的密钥和凭据。

以下服务提供托管轮换：

- Amazon Aurora 为主用户凭证提供托管轮换。有关更多信息，请参阅《Amazon Aurora 用户指南》中的使用 Amazon Aurora 和 AWS Secrets Manager管理密码。
- Amazon ECS S ervice Connect 为 AWS Private Certificate Authority TLS 证书提供托管轮换。有关更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》中的支持 Service Connect 的 TLS。
- Amazon RDS 为主用户凭证提供托管轮换。有关更多信息，请参阅《Amazon RDS 用户指南》中的使用 Amazon RDS 和 AWS Secrets Manager管理密码。
- Amazon Redshift 为管理员密码提供托管轮换。有关更多信息，请参阅《Amazon Redshift 管理指南》中的使用 AWS Secrets Manager管理 Amazon Redshift 管理员密码。

> **Tip**
>
> 有关所有其他类型的密钥，请参阅 the section called “通过 Lambda 函数进行轮换”。

托管秘密的轮换通常会在一分钟内完成。在轮换期间，检索秘密的新连接可能会获得先前版本的凭证。在应用程序中，我们强烈建议您遵循使用以您的应用程序所需的最低权限创建的数据库用户的最佳实践，而不是使用主用户。对于应用程序用户，为了获得最高可用性，可以使用交替用户轮换策略。

更改托管轮换的计划

1. 在 Secrets Manager 控制台中打开托管密钥。您可以访问管理服务中的链接，也可以在 Secrets Manager 控制台中搜索密钥。
2. 在 Rotation schedule（轮换计划）下，在 Schedule expression builder（计划表达式生成器）或 Schedule expression（计划表达式）中，以 UTC 时区格式输入您的计划。Secrets Manager 会将您的计划存储为 `rate()` 或 `cron()` 表达式。轮换时段将自动从午夜开始，除非您指定 Start time（开始时间）。您可以每四小时轮换一次密钥。有关更多信息，请参阅 轮换计划。
3. （可选）对于 Window duration（时段持续时间），选择您希望 Secrets Manager 在其间轮换密钥的时段长度，例如 **3h** 表示三个小时的时段。该时段不得延伸到下一个轮换时段。如果未指定 Window duration（时段持续时间），则对于以小时为单位的轮换计划，时段将在一小时后自动关闭。对于以天为单位的轮换计划，时段将在一天结束时自动关闭。
4. 选择保存。

更改托管轮换的计划 (AWS CLI)

- 调用 `rotate-secret`。以下示例在每月 1 日和 15 日 UTC 16:00 至 18:00 之间轮换密钥。有关更多信息，请参阅 轮换计划。

```
aws secretsmanager rotate-secret \
    --secret-id MySecret \
    --rotation-rules \
        "{\"ScheduleExpression\": \"cron(0 16 1,15 * ? *)\", \"Duration\": \"2h\"}"
```

通过 Lambda 函数进行轮换

对于许多类型的密钥，Secrets Manager 使用 AWS Lambda 函数来更新密钥以及数据库或服务。有关使用 Lambda 函数的成本的信息，请参阅 定价。

对于某些 由其他服务管理的密钥，可使用托管轮换。要使用 托管轮换，请首先通过管理服务来创建密钥。

在轮换期间，Secrets Manager 会录入指示轮换状态的事件。有关更多信息，请参阅 the section called “使用登录 AWS CloudTrail ”。

要轮换密钥，Secrets Manager 会根据你设置的轮换计划调用 Lambda 函数。如果在设置自动轮换时也手动更新密钥值，则 Secrets Manager 在计算下一次轮换日期时会认为这是有效的轮换。

在轮换过程中，Secrets Manager 调用几次同一函数，每次使用不同的参数。Secrets Manager 调用具有以下 JSON 请求参数结构的函数：

```
{
    "Step" : "request.type",
    "SecretId" : "string",
    "ClientRequestToken" : "string",
    "RotationToken" : "string"
}
```

参数：

- 步骤 – 轮换步骤：`create_secret`、`set_secret`、`test_secret` 或 `finish_secret`。有关更多信息，请参阅 the section called “轮换函数的四个步骤”。
- SecretId— 轮换秘密的 ARN。
- ClientRequestToken— 新版本密钥的唯一标识符。此值有助于确保幂等性。有关更多信息，请参阅《AWS Secrets Manager API 参考》 ClientRequestToken中的 PutSecretValue：。
- RotationToken— 表示请求来源的唯一标识符。这是使用代入角色或跨账户轮换的密钥轮换所必需的，在这种轮换中，您可以通过在另一个账户中使用 Lambda 轮换函数来轮换一个账户中的密钥。在这两种情况下，轮换函数都假设一个 IAM 角色来调用 Secrets Manager，然后 Secrets Manager 使用轮换令牌来验证 IAM 角色身份。

如果任何轮换步骤失败，Secrets Manager 会多次重试整个轮换过程。

主题

- 为 Amazon RDS、Amazon Aurora、Amazon Redshift 或 Amazon DocumentDB 密钥设置自动轮换
- 为非数据库 AWS Secrets Manager 密钥设置自动轮换
- 使用设置自动轮换 AWS CLI
- Lambda 函数轮换策略
- Lambda 轮换函数
- AWS Secrets Manager 轮换函数模板
- Lambda 轮换函数的执行角色权限 AWS Secrets Manager
- AWS Lambda 轮换功能的网络接入
- AWS Secrets Manager 轮换问题排查

为 Amazon RDS、Amazon Aurora、Amazon Redshift 或 Amazon DocumentDB 密钥设置自动轮换

本教程介绍了如何为数据库密钥设置 the section called “通过 Lambda 函数进行轮换”。Rotation 是定期更新密钥的过程。轮换密钥时，您会同时更新密钥和数据库中的凭证。在 Secrets Manager 中，您可以为数据库密钥设置自动轮换。

要使用控制台设置轮换，您需要先选择轮换策略。然后配置密钥以进行轮换，如果您还没有 Lambda 轮换函数，这将创建一个 Lambda 轮换函数。控制台还会为 Lambda 函数执行角色设置权限。最后一步是确保 Lambda 轮换函数可以通过网络访问 Secrets Manager 和数据库。

> **Warning**
>
> 要启用自动轮换，您必须有权为 Lambda 轮换函数创建 IAM 执行角色并向其附加权限策略。您需要拥有 `iam:CreateRole` 和 `iam:AttachRolePolicy` 两个权限。授予这些权限允许身份授予自己任何权限。

步骤：

- 步骤 1：选择轮换策略并（可选）创建超级用户密钥
- 步骤 2：配置轮换并创建轮换函数
- 第 3 步：（可选）为轮换函数设置额外的权限条件
- 步骤 4：为轮换函数设置网络访问
- 后续步骤

步骤 1：选择轮换策略并（可选）创建超级用户密钥

有关 Secrets Manager 提供的策略的信息，请参阅 the section called “Lambda 函数轮换策略”。

如果选择 alternating users strategy（交替用户策略），您必须 创建密钥 并在其中存储数据库超级用户凭证。您需要一个包含超级用户凭证的密钥，因为轮换会克隆第一个用户，而大多数用户没有该权限。请注意，Amazon RDS 代理不支持交替用户策略。

步骤 2：配置轮换并创建轮换函数

为 Amazon RDS、Amazon DocumentDB 或 Amazon Redshift 密钥启用轮换

1. 打开 Secrets Manager 控制台，网址为https://console.aws.amazon.com/secretsmanager/。

2. 在密钥列表页上，选择您的密钥。
3. 在 Secret details (密钥详细信息) 页上的 Rotation configuration (轮换配置) 部分中，选择 Edit rotation (编辑轮换)。
4. 在编辑轮换配置对话框中，执行以下操作：
   a. 启用 Automatic rotation（自动轮换）。
   b. 在 Rotation schedule（轮换计划）下，在 Schedule expression builder（计划表达式生成器）或 Schedule expression（计划表达式）中，以 UTC 时区格式输入您的计划。Secrets Manager 会将您的计划存储为 `rate()` 或 `cron()` 表达式。轮换时段将自动从午夜开始，除非您指定 Start time（开始时间）。您可以每四小时轮换一次密钥。有关更多信息，请参阅 轮换计划。
   c. （可选）对于 Window duration（时段持续时间），选择您希望 Secrets Manager 在其间轮换密钥的时段长度，例如 **3h** 表示三个小时的时段。该时段不得延伸到下一个轮换时段。如果未指定 Window duration（时段持续时间），则对于以小时为单位的轮换计划，时段将在一小时后自动关闭。对于以天为单位的轮换计划，时段将在一天结束时自动关闭。
   d. （可选）请选择 Rotate immediately when the secret is stored（在存储密钥时立即轮换），以在保存更改时轮换密钥。如果您清除该复选框，则第一次轮换将按照您设置的计划开始。

      如果轮换失败，例如因为步骤 3 和 4 尚未完成，Secrets Manager 会多次重试轮换过程。
   e. 在 Rotation function（轮换函数）下，执行以下操作之一：
      - 选择 Create a new Lambda function（创建新的 Lambda 函数），然后输入新函数的名称。Secrets Manager 会将 `SecretsManager` 添加到函数名称的开头。Secrets Manager 会基于相应的模板创建函数并为 Lambda 执行角色设置必要的权限。
      - 选择 Use an existing Lambda function（使用现有 Lambda 函数），以重复使用用于另一个密钥的轮换函数。在 Recommended VPC configurations（建议的 VPC 配置）下列出的轮换函数，与数据库具有相同的 VPC 和安全组，有助于函数访问数据库。
   f. 对于轮换策略，选择单用户或交替用户策略。有关更多信息，请参阅 the section called “步骤 1：选择轮换策略并（可选）创建超级用户密钥”。
5. 选择 Save。

第 3 步：（可选）为轮换函数设置额外的权限条件

我们建议您在轮换函数的资源策略中包括上下文密钥 `aws:SourceAccount`，以防止 Lambda 被用作混淆代理。对于某些 AWS 服务，为了避免混淆副手的情况， AWS 建议您同时使用`aws:SourceArn`和`aws:SourceAccount`全局条件键。但如果轮换函数策略中包括

aws:SourceArn 条件，则轮换函数只能用于轮换该 ARN 指定的密钥。我们建议您仅在其中包括上下文键 aws:SourceAccount，以便对多个密钥使用轮换函数。

更新轮换函数资源策略

1. 在 Secrets Manager 控制台中选择您的密钥，然后在详细信息页面中的 Rotation configuration（轮换配置）下，选择 Lambda 轮换函数。Lambda 控制台将打开。
2. 按照 Using resource-based policies for Lambda（将基于资源的策略用于 Lambda）中的说明添加 aws:sourceAccount 条件。

```
"Condition": {
    "StringEquals": {
        "AWS:SourceAccount": "123456789012"
    }
},
```

如果密钥使用 AWS 托管式密钥 aws/secretsmanager 以外的 KMS 密钥进行加密，则 Secrets Manager 会向 Lambda 执行角色授予使用该密钥的权限。您可以使用 SecretARN 加密上下文 来限制解密函数的使用，从而确保轮换函数角色只能解密其负责轮换的密钥。

更新轮换函数执行角色

1. 从 Lambda 轮换函数中选择配置，然后在执行角色下，选择角色名称。
2. 按照 修改角色权限策略 中的说明添加 kms:EncryptionContext:SecretARN 条件。

```
"Condition": {
    "StringEquals": {
        "kms:EncryptionContext:SecretARN": "SecretARN"
    }
},
```

步骤 4：为轮换函数设置网络访问

有关更多信息，请参阅 the section called “AWS Lambda 轮换功能的网络接入”。

后续步骤

请参阅the section called “轮换问题排查”。

为非数据库 AWS Secrets Manager 密钥设置自动轮换

本教程介绍了如何为非数据库密钥设置 the section called “通过 Lambda 函数进行轮换”。Rotation 是定期更新密钥的过程。轮换密钥时，会同时更新密钥以及拥有密钥的数据库或服务中的凭证。

有关数据库密钥的信息，请参阅 自动轮换数据库密钥（控制台）。

> **Warning**
>
> 要启用自动轮换，您必须有权为 Lambda 轮换函数创建 IAM 执行角色并向其附加权限策略。您需要拥有 `iam:CreateRole` 和 `iam:AttachRolePolicy` 两个权限。授予这些权限允许身份授予自己任何权限。

步骤：

- 步骤 1：创建通用轮换函数
- 步骤 2：编写轮换函数代码
- 步骤 3：配置密钥以进行轮换
- 步骤 4：允许轮换函数访问 Secrets Manager 以及您的数据库或服务
- 步骤 5：允许 Secrets Manager 调用轮换函数
- 步骤 6：为轮换函数设置网络访问
- 后续步骤

步骤 1：创建通用轮换函数

首先，创建一个 Lambda 轮换函数。它不包含用于轮换您的密钥的代码，因此您将在后面的步骤中编写该代码。有关轮换函数如何工作的信息，请参阅 the section called “Lambda 轮换函数”。

在支持的区域中 AWS Serverless Application Repository ，您可以使用从模板创建函数。有关受支持的 区域的列表，请参阅AWS Serverless Application Repository FAQs。在其他区域，您将从头开始创建函数并将模板代码复制到函数中。

创建通用轮换函数

1. 要确定您所在的地区 AWS Serverless Application Repository 是否支持，请参阅《AWS 一般参考》中的AWS Serverless Application Repository 终端节点和配额。

2. 请执行以下操作之一：
   - 如果您 AWS Serverless Application Repository 所在的地区支持：
     a. 在 Lambda 控制台中，选择应用程序，然后选择创建应用程序。
     b. 在创建应用程序页面上，选择无服务器应用程序选项卡。
     c. 在公用应用程序下的搜索框中，输入 **SecretsManagerRotationTemplate**。
     d. 选择显示创建自定义 IAM 角色或资源策略的应用程序。
     e. 选择 SecretsManagerRotationTemplate 磁贴。
     f. 在查看、配置和部署页面上的应用程序设置磁贴中，填写必填字段。
        - 对于端点，输入您所在区域的端点，包括 **https://**。有关 终端节点的列表，请参阅the section called “Secrets Manager 端点”。
        - 要将 Lambda 函数放在 VPC 中，请添加 vpcSecurityGroupID 和。vpcSubnetIds
     g. 选择部署。
   - 如果您所在的地区 AWS Serverless Application Repository 不支持：
     a. 在 Lambda 控制台中，选择函数，然后选择创建函数。
     b. 在 Create function (创建函数) 页面上，执行以下操作：
        i. 选择从头开始创作。
        ii. 在 Function name（函数名称）中，输入轮换函数的名称。
        iii. 对于 Runtime（运行时），选择 Python 3.9。
        iv. 选择创建函数。

步骤 2：编写轮换函数代码

在此步骤中，您将编写用于更新密钥以及该密钥所针对的服务或数据库的代码。有关轮换函数作用的信息（包括编写自己的轮换函数的提示），请参阅 the section called “Lambda 轮换函数”。您可以使用轮换函数模板 作为参考。

步骤 3：配置密钥以进行轮换

在此步骤中，您将为密钥设置轮换计划，并将轮换函数连接到密钥。

配置轮换并创建空轮换函数

1. 打开 Secrets Manager 控制台，网址为https://console.aws.amazon.com/secretsmanager/。

2. 在密钥列表页上，选择您的密钥。
3. 在 Secret details (密钥详细信息) 页上的 Rotation configuration (轮换配置) 部分中，选择 Edit rotation (编辑轮换)。在编辑轮换配置对话框中，执行以下操作：
   a. 启用 Automatic rotation（自动轮换）。
   b. 在 Rotation schedule（轮换计划）下，在 Schedule expression builder（计划表达式生成器）或 Schedule expression（计划表达式）中，以 UTC 时区格式输入您的计划。Secrets Manager 会将您的计划存储为 `rate()` 或 `cron()` 表达式。轮换时段将自动从午夜开始，除非您指定 Start time（开始时间）。您可以每四小时轮换一次密钥。有关更多信息，请参阅 轮换计划。
   c. （可选）对于 Window duration（时段持续时间），选择您希望 Secrets Manager 在其间轮换密钥的时段长度，例如 **3h** 表示三个小时的时段。该时段不得延伸到下一个轮换时段。如果未指定 Window duration（时段持续时间），则对于以小时为单位的轮换计划，时段将在一小时后自动关闭。对于以天为单位的轮换计划，时段将在一天结束时自动关闭。
   d. （可选）请选择 Rotate immediately when the secret is stored（在存储密钥时立即轮换），以在保存更改时轮换密钥。如果您清除该复选框，则第一次轮换将按照您设置的计划开始。
   e. 在轮换函数下，选择在步骤 1 中创建的 Lambda 函数。
   f. 选择保存。

步骤 4：允许轮换函数访问 Secrets Manager 以及您的数据库或服务

Lambda 轮换函数需要权限才能访问 Secrets Manager 中的密钥，并且需要权限才能访问您的数据库或服务。在此步骤中，您将向 Lambda 执行角色授予这些权限。如果密钥使用 AWS 托管式密钥 `aws/secretsmanager` 以外的 KMS 密钥进行加密，则您需要向 Lambda 执行角色授予使用该密钥的权限。您可以使用 SecretARN 加密上下文 来限制解密函数的使用，从而确保轮换函数角色只能解密其负责轮换的密钥。有关策略示例，请参阅 轮换权限。

有关说明，请参阅《AWS Lambda 开发人员指南》中的 Lambda 执行角色。

步骤 5：允许 Secrets Manager 调用轮换函数

要允许 Secrets Manager 按照您设置的轮换计划调用轮换函数，您需要在 Lambda 函数的资源策略中向 Secrets Manager 服务主体授予 `lambda:InvokeFunction` 权限。

我们建议您在轮换函数的资源策略中包括上下文密钥 `aws:SourceAccount`，以防止 Lambda 被用作混淆代理。对于某些 AWS 服务，为了避免混淆副手的情况， AWS 建议您同时使用`aws:SourceArn`和`aws:SourceAccount`全局条件键。但如果轮换函数策略中包括

`aws:SourceArn` 条件，则轮换函数只能用于轮换该 ARN 指定的密钥。我们建议您仅在其中包括上下文键 `aws:SourceAccount`，以便对多个密钥使用轮换函数。

要将资源策略附加到 Lambda 函数，请参阅将基于资源的策略用于 Lambda。

以下策略允许 Secrets Manager 调用 Lambda 函数。

JSON

```
{
    "Version": "2012-10-17",
    "Id": "default",
    "Statement": [
    {
        "Effect": "Allow",
        "Principal": {
            "Service": "secretsmanager.amazonaws.com"
            },
        "Action": "lambda:InvokeFunction",
        "Condition": {
            "StringEquals": {
                "AWS:SourceAccount": "123456789012"
            }
        },
        "Resource": "arn:aws:lambda:us-east-1:123456789012:function:function-
name"
    }
    ]
}
```

步骤 6：为轮换函数设置网络访问

在此步骤中，您将允许轮换功能同时连接到 Secrets Manager 以及该密钥所针对的服务或数据库。旋转函数必须能够访问两者才能轮换密钥。请参阅the section called “AWS Lambda 轮换功能的网络接入”。

后续步骤

当在步骤 3 中配置轮换时，您会设置一个轮换密钥的计划。如果轮换在计划时失败，Secrets Manager 将多次尝试轮换。您也可以按照 立即轮换密钥 中的说明立即开始轮换。

如果轮换失败，请参阅 轮换问题排查。

使用设置自动轮换 AWS CLI

本教程介绍了如何the section called “通过 Lambda 函数进行轮换”使用设置 AWS CLI。轮换密钥时，会同时更新密钥以及拥有密钥的数据库或服务中的凭证。

您也可以使用控制台设置轮换。有关数据库密钥的信息，请参阅 自动轮换数据库密钥（控制台）。有关所有其他类型的密钥，请参阅 自动轮换非数据库密钥（控制台）。

要使用设置轮换 AWS CLI，如果要轮换数据库密钥，则首先需要选择轮换策略。如果选择 alternating users strategy（交替用户策略），您必须存储一个单独密钥，其中包含数据库超级用户凭证。接下来，编写轮换函数代码。Secrets Manager 会提供模板，您可以基于该模板创建函数。然后，使用代码创建 Lambda 函数，并为 Lambda 函数和 Lambda 执行角色设置权限。下一步是确保 Lambda 函数可以通过网络访问 Secrets Manager 和数据库或服务。最后，配置密钥以进行轮换。

步骤：

- 数据库密钥的先决条件：选择轮换策略
- 步骤 1：编写轮换函数代码
- 第 2 步：创建 Lambda 函数
- 步骤 3：设置网络访问
- 步骤 4：配置要轮换的密钥
- 后续步骤

数据库密钥的先决条件：选择轮换策略

有关 Secrets Manager 提供的策略的信息，请参阅 the section called “Lambda 函数轮换策略”。

选项 1：单用户策略

如果选择单用户策略，则可以继续执行步骤 1。

选项 2：交替用户策略

如果选择交替用户策略，则必须：

- 创建一个密钥并在其中存储数据库超级用户凭证。您需要一个包含超级用户凭证的密钥，因为交替用户轮换会克隆第一个用户，而大多数用户没有该权限。

- 将超级用户密钥的 ARN 添加到原始密钥。有关更多信息，请参阅 the section called “密钥的 JSON 结构”。

请注意，Amazon RDS 代理不支持交替用户策略。

步骤 1：编写轮换函数代码

要轮换密钥，您需要轮换函数。轮换函数是 Secrets Manager 为轮换密钥而调用的 Lambda 函数。有关更多信息，请参阅 the section called “通过 Lambda 函数进行轮换”。在此步骤中，您将编写用于更新密钥以及该密钥所针对的服务或数据库的代码。

Secrets Manager 在 轮换函数模板 中提供了 Amazon RDS、Amazon Aurora、Amazon Redshift 和 Amazon DocumentDB 数据库密钥的模板。

编写轮换函数代码

1. 请执行以下操作之一：
   - 查看轮换函数模板列表。如果有与您的服务和轮换策略匹配的模板，请复制代码。
   - 对于其他类型的密钥，请编写您自己的轮换函数。有关说明，请参阅the section called “Lambda 轮换函数”。
2. 将该文件*my-function.zip*连同任何所需的依赖项一起保存在 ZIP 文件中。

第 2 步：创建 Lambda 函数

在此步骤中，您将使用在步骤 1 中创建的 ZIP 文件创建 Lambda 函数。您还将设置 Lambda 执行角色，即调用函数时 Lambda 代入的角色。

创建 Lambda 轮换函数和执行角色

1. 为 Lambda 执行角色创建信任策略并将其另存为 JSON 文件。有关示例和更多信息，请参阅 the section called “轮换权限”。该策略必须：
   - 允许角色对密钥调用 Secrets Manager 操作。
   - 允许角色调用密钥所针对的服务，例如创建新密码。
2. 通过调用 `iam create-role` 来创建 Lambda 执行角色并应用在上一步中创建的信任策略。

```
aws iam create-role \
    --role-name rotation-lambda-role \
```

```
    --assume-role-policy-document file://trust-policy.json
```

3. 通过调用 `lambda create-function` 从 ZIP 文件创建 Lambda 函数。

```
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. 在 Lambda 函数上设置资源策略，以允许 Secrets Manager 通过调用 `lambda add-permission` 来调用该资源策略。

```
aws lambda add-permission \
  --function-name my-rotation-function \
  --action lambda:InvokeFunction \
  --statement-id SecretsManager \
  --principal secretsmanager.amazonaws.com \
  --source-account 123456789012
```

步骤 3：设置网络访问

有关更多信息，请参阅 the section called “AWS Lambda 轮换功能的网络接入”。

步骤 4：配置要轮换的密钥

要为密钥开启自动轮换功能，请调用 `rotate-secret`。您可以使用 `cron()` 或 `rate()` 计划表达式设置轮换计划，也可以设置轮换时段持续时间。有关更多信息，请参阅 the section called “轮换计划”。

```
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\"}"
```

后续步骤

请参阅the section called “轮换问题排查”。

Lambda 函数轮换策略

对于 the section called “通过 Lambda 函数进行轮换”，对于数据库密钥，Secrets Manager 提供了两种轮换策略。

轮换策略：单用户

此策略在一个密钥中更新一个用户的凭证。对于 Amazon RDS Db2 实例，由于用户无法更改自己的密码，因此您必须在单独的秘密中提供管理员凭证。这是最简单的轮换策略，适用于大多数使用场景。具体而言，建议您为一次性（临时）用户或交互式用户的凭证使用此策略。

轮换密钥时，不会删除打开的数据库连接。在进行轮换时，在数据库中的密码更改后一小段时间，相应的密码才会更新。在此期间，数据库有较低的风险拒绝使用轮换凭证的调用。您可以使用适当的重试策略来降低风险。轮换后，新连接将使用新凭证。

轮换策略：交替用户

此策略在一个密钥中更新两个用户的凭证。您创建第一个用户，然后在第一次轮换期间，轮换函数将进行克隆以创建第二个用户。每次轮换密钥时，轮换函数都会交替更新其更新的用户密码。由于大多数用户无权克隆自己，因此您必须在另一个密钥中为 `superuser` 提供凭证。如果数据库中的克隆用户与原始用户具有的权限不同，或者涉及一次性（临时）用户或交互式用户的凭证，我们建议使用单用户轮换策略。

此策略适用于具有权限模型的数据库，其中一个角色拥有数据库表，而另一个角色具有访问数据库表的权限。其也适用于需要高可用性的应用程序。如果应用程序在轮换期间检索密钥，则该应用程序仍会获得一组有效的凭证。轮换后，`user` 和 `user_clone` 凭证均有效。在这种类型的轮换期间，应用程序获得拒绝的可能性甚至比单用户轮换获得拒绝的可能性更小。如果数据库托管在服务器场中，密码更改需要时间传播到所有服务器，则存在数据库拒绝使用新凭证的调用的风险。您可以使用适当的重试策略来降低风险。

Secrets Manager 将创建权限与原始用户相同的克隆用户。如果您在创建克隆用户后更改了原始用户的权限，则还必须更改克隆用户的权限。

例如，假设您使用某个数据库用户的凭证创建了一个密钥，则该密钥包含一个带有这些凭证的版本。

第一次轮换 — 轮换功能使用生成的密码创建您的用户的克隆，这些凭据将成为当前的密钥版本。

第二次轮换-轮换功能更新原始用户的密码。

第三轮轮换-轮换功能更新克隆用户的密码。

Lambda 轮换函数

在中the section called “通过 Lambda 函数进行轮换”， AWS Lambda 函数轮换密钥。 AWS Secrets Manager 在轮换期间使用暂存标签来识别秘密版本。

如果 AWS Secrets Manager 没有为您的密钥类型提供轮换函数模板，则可以创建自定义轮换函数。编写旋转函数时，请遵循以下准则：

自定义旋转函数的最佳实践

- 使用通用旋转模板作为起点。
- 谨慎使用调试或记录语句。他们可以将信息写入 Amazon CloudWatch 日志。确保日志不包含敏感信息。

  有关日志语句的示例，请参阅the section called “轮换函数模板”源代码。
- 为了安全起见， AWS Secrets Manager 仅允许 Lambda 轮换函数直接轮换密钥。轮换函数无法调用其他 Lambda 函数来轮换密钥。
- 有关调试指导，请参阅测试和调试无服务器应用程序。
- 例如，如果您使用外部二进制文件和库来连接资源，则需要对其进行修补和更新。
- Package 将您的旋转函数和所有依赖项打包到 ZIP 文件中，例如*my-function.zip*。

> **Warning**
>
> 由于 Lambda 函数的执行线程不足，将预配置的并发参数设置为低于 10 的值可能会导致限制。有关更多信息，请参阅《开发人员指南》中的 “了解预留并发和预配置并发” AWS Lambda AWS Lambda 。

轮换函数的四个步骤

主题

- createSecret：创建密钥的新版本

- setSecret：更改数据库或服务中的凭证
- testSecret：测试新的密钥版本
- finishSecret：完成轮换

`createSecret`：创建密钥的新版本

方法 `createSecret` 首先通过使用传入的 `ClientRequestToken` 调用 `get_secret_value` 来检查密钥是否存在。如果没有密钥，它会使用 `create_secret` 创建一个新密钥，并将令牌作为 `VersionId`。然后它使用 `get_random_password` 生成一个新的密钥值。接下来，它调用 `put_secret_value` 以将其与暂存标签 `AWSPENDING` 一起存储。将新的密钥值存储在 `AWSPENDING` 中有助于确保幂等性。如果由于任何原因轮换失败，您可以在后续调用中引用该密钥值。请参阅如何使我的 Lambda 函数具有幂等性。

编写自己的轮换函数的技巧

- 确保新的密钥值仅包含对数据库或服务有效的字符。使用 `ExcludeCharacters` 参数排除字符。
- 在测试函数时，使用查看版本阶段：调用`describe-secret`并查看`VersionIdsToStages`。AWS CLI
- 对于 Amazon RDS MySQL，在交替用户轮换中，Secrets Manager 会创建一个名称不超过 16 个字符的克隆用户。您可以修改轮换函数以允许使用更长的用户名。MySQL 5.7 及更高版本支持最多 32 个字符的用户名，但 Secrets Manager 会在用户名末尾附加“_clone”（六个字符），因此用户名最多必须保持在 26 个字符以内。

setSecret：更改数据库或服务中的凭证

方法 `setSecret` 更改数据库或服务中的凭证以匹配密钥的 `AWSPENDING` 版本中的新密钥值。

编写自己的轮换函数的技巧

- 如果将语句传递给解释语句的服务（如数据库），请使用查询参数化。有关更多信息，请参阅 OWASP 网站上的 Query Parameterization Cheat Sheet。
- 轮换函数作为特权代理，有权访问和修改 Secrets Manager 密钥和目标资源中的客户凭证。为防范潜在的混淆代理攻击，您需要确保攻击者无法使用该函数访问其他资源。在更新凭证之前：
  - 检查密钥 `AWSCURRENT` 版本中的凭证是否有效。如果 `AWSCURRENT` 凭证无效，请放弃轮换尝试。
  - 检查 `AWSCURRENT` 和 `AWSPENDING` 密钥值是否适用于同一资源。对于用户名和密码，检查 `AWSCURRENT` 和 `AWSPENDING` 用户名是否相同。

- 检查目标服务资源是否相同。对于数据库，检查 AWSCURRENT 和 AWSPENDING 主机名是否相同。
- 在极少数情况下，您可能希望为数据库自定义现有轮换功能。例如，对于交替用户轮换，Secrets Manager 通过复制第一个用户的运行时配置参数来创建克隆用户。如果要包含更多属性，或更改授予克隆用户的属性，则需要更新 set_secret 函数中的代码。

testSecret：测试新的密钥版本

然后，Lambda 轮换函数将使用该密钥来访问数据库或服务，从而测试密钥的 AWSPENDING 版本。基于 轮换函数模板 测试的轮换函数使用读取访问权限测试新的密钥。

finishSecret：完成轮换

最后，Lambda 轮换函数将标签 AWSCURRENT 从之前的密钥版本移动到此版本，这将同时在同一 API 调用中移除 AWSPENDING 标签。Secrets Manager 添加 AWSPREVIOUS 暂存标注到以前的版本，以便您保留密钥的上次已知良好的版本。

方法 finish_secret 使用 update_secret_version_stage 将暂存标签 AWSCURRENT 从早期密钥版本移动到新的密钥版本。Secrets Manager 会将 AWSPREVIOUS 暂存标签自动添加到早期版本，以便您保留上次已知良好的密钥版本。

编写自己的轮换函数的技巧

- 在此AWSPENDING之前不要将其删除，也不要使用单独的 API 调用将其删除，因为这可能会向 Secrets Manager 表明轮换未成功完成。Secrets Manager 添加 AWSPREVIOUS 暂存标注到以前的版本，以便您保留密钥的上次已知良好的版本。

成功轮换后，AWSPENDING 暂存标签可能附加到与 AWSCURRENT 版本相同的版本，也可能未附加到任何版本。如果 AWSPENDING 暂存标签存在但未附加到与 AWSCURRENT 相同的版本，则以后对轮换的任何调用都假定先前的轮换请求仍在进行中并返回错误。轮换不成功时，AWSPENDING 暂存标签可能会附加到空密钥版本。有关更多信息，请参阅 轮换问题排查。

AWS Secrets Manager 轮换函数模板

AWS Secrets Manager 提供了一组轮换函数模板，可帮助自动对各种数据库系统和服务的凭据进行安全管理。这些模板是 ready-to-use Lambda 函数，它们实现了证书轮换的最佳实践，可帮助您在无需人工干预的情况下保持安全状态。

这些模板支持两种主要的轮换策略：

- 单用户轮换，用于更新单个用户的凭证。
- 交替用户轮换，可保留两个独立的用户，以帮助消除凭据更改期间的停机时间。

Secrets Manager 还提供了一个通用模板，可以作为任何类型密钥的起点。

若要使用模板，请参阅：

- 自动轮换数据库密钥（控制台）
- 自动轮换非数据库密钥（控制台）

要编写自己的轮换函数，请参阅 Write a rotation function。

模板

- Amazon RDS 和 Amazon Aurora
  - Amazon RDS Db2 单用户
  - Amazon RDS Db2 交替用户
  - Amazon RDS MariaDB 单用户
  - Amazon RDS MariaDB 交替用户
  - Amazon RDS 和 Amazon Aurora MySQL 单用户
  - Amazon RDS 和 Amazon Aurora MySQL 交替用户
  - Amazon RDS Oracle 单用户
  - Amazon RDS Oracle 交替用户
  - Amazon RDS 和 Amazon Aurora PostgreSQL 单用户
  - Amazon RDS 和 Amazon Aurora PostgreSQL 交替用户
  - Amazon RDS Micros SQLServer o RDS
  - Amazon RDS Microsoft SQLServer 交替用户
- Amazon DocumentDB（与 MongoDB 兼容）
  - Amazon DocumentDB 单个用户
  - Amazon DocumentDB 交替用户
- Amazon Redshift
  - Amazon Redshift 单用户
  - Amazon Redshift 交替用户
- Amazon Timestream for InfluxDB

  - Amazon Timestream for InfluxDB 单用户
  - Amazon Timestream for InfluxDB 交替用户
- Amazon ElastiCache
- Active Directory
  - Active Directory 凭证
  - Active Directory keytab
- 其他密钥类型

Amazon RDS 和 Amazon Aurora

Amazon RDS Db2 单用户

- 模板名称： SecretsManagerRDSDb2RotationSingleUser
- 轮换策略： 轮换策略：单用户。
- **`SecretString`** 结构：the section called “Amazon RDS 和 Aurora 凭证”。
- 源代码：https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSDb2RotationSingleUser/lambda _function.py
- 依赖关系：python-ibmdb

Amazon RDS Db2 交替用户

- 模板名称： SecretsManagerRDSDb2RotationMultiUser
- 轮换策略： the section called “交替用户”。
- **`SecretString`** 结构：the section called “Amazon RDS 和 Aurora 凭证”。
- 源代码：https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSDb2RotationMultiUser/lambda _function.py
- 依赖关系：python-ibmdb

Amazon RDS MariaDB 单用户

- 模板名称： SecretsManagerRDSMariaDBRotationSingleUser
- 轮换策略： 轮换策略：单用户。
- **`SecretString`** 结构：the section called “Amazon RDS 和 Aurora 凭证”。

- 源代码：https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSMariaDBRotationSingleUser/lambda _function.py
- 依赖关系：PyMySQL 1.0.2。如果使用 sha256 密码进行身份验证，则为 PyMy SQL [rsa]。有关在 Lambda 运行时中使用含编译代码的包的信息，请参阅 AWS 知识中心 中的如何将含已编译二进制文件的 Python 包添加到我的部署包，并使该包与 Lambda 兼容？。

Amazon RDS MariaDB 交替用户

- 模板名称： SecretsManagerRDSMariaDBRotationMultiUser
- 轮换策略： the section called “交替用户”。
- **`SecretString`** 结构：the section called “Amazon RDS 和 Aurora 凭证”。
- 源代码：https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSMariaDBRotationMultiUser/lambda _function.py
- 依赖关系：PyMySQL 1.0.2。如果使用 sha256 密码进行身份验证，则为 PyMy SQL [rsa]。有关在 Lambda 运行时中使用含编译代码的包的信息，请参阅 AWS 知识中心 中的如何将含已编译二进制文件的 Python 包添加到我的部署包，并使该包与 Lambda 兼容？。

Amazon RDS 和 Amazon Aurora MySQL 单用户

- 模板名称： SecretsManagerRDSMySQLRotationSingleUser
- 轮换策略： the section called “单用户”。
- 预期的 **`SecretString`** 结构：the section called “Amazon RDS 和 Aurora 凭证”。
- 源代码：https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSMySQLRotationSingleUser/lambda _function.py
- 依赖关系：PyMySQL 1.0.2。如果使用 sha256 密码进行身份验证，则为 PyMy SQL [rsa]。有关在 Lambda 运行时中使用含编译代码的包的信息，请参阅 AWS 知识中心 中的如何将含已编译二进制文件的 Python 包添加到我的部署包，并使该包与 Lambda 兼容？。

Amazon RDS 和 Amazon Aurora MySQL 交替用户

- 模板名称： SecretsManagerRDSMySQLRotationMultiUser
- 轮换策略： the section called “交替用户”。
- 预期的 **`SecretString`** 结构：the section called “Amazon RDS 和 Aurora 凭证”。

- 源代码：https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSMySQLRotationMultiUser/lambda _function.py
- 依赖关系：PyMySQL 1.0.2。如果使用 sha256 密码进行身份验证，则为 PyMy SQL [rsa]。有关在 Lambda 运行时中使用含编译代码的包的信息，请参阅 AWS 知识中心 中的如何将含已编译二进制文件的 Python 包添加到我的部署包，并使该包与 Lambda 兼容？。

Amazon RDS Oracle 单用户

- 模板名称： SecretsManagerRDSOracleRotationSingleUser
- 轮换策略： the section called “单用户”。
- 预期的 **`SecretString`** 结构：the section called “Amazon RDS 和 Aurora 凭证”。
- 源代码：https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSOracleRotationSingleUser/lambda _function.py
- 依赖项：python-oracledb 2.4.1

Amazon RDS Oracle 交替用户

- 模板名称： SecretsManagerRDSOracleRotationMultiUser
- 轮换策略： the section called “交替用户”。
- 预期的 **`SecretString`** 结构：the section called “Amazon RDS 和 Aurora 凭证”。
- 源代码：https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSOracleRotationMultiUser/lambda _function.py
- 依赖项：python-oracledb 2.4.1

Amazon RDS 和 Amazon Aurora PostgreSQL 单用户

- 模板名称： SecretsManagerRDSPostgreSQLRotationSingleUser
- 轮换策略： 轮换策略：单用户。
- 预期的 **`SecretString`** 结构：the section called “Amazon RDS 和 Aurora 凭证”。
- 源代码：https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSPostgreSQLRotationSingleUser/lambda _function.py
- 依赖关系：PyGreSQL 5.0.7

Amazon RDS 和 Amazon Aurora PostgreSQL 交替用户

- 模板名称： SecretsManagerRDSPostgreSQLRotationMultiUser
- 轮换策略： the section called “交替用户”。
- 预期的 **`SecretString`** 结构：the section called “Amazon RDS 和 Aurora 凭证”。
- 源代码：https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSPostgreSQLRotationMultiUser/lambda _function.py
- 依赖关系：PyGreSQL 5.0.7

Amazon RDS Micros SQLServer o RDS

- 模板名称： SecretsManagerRDSSQLServerRotationSingleUser
- 轮换策略： the section called “单用户”。
- 预期的 **`SecretString`** 结构：the section called “Amazon RDS 和 Aurora 凭证”。
- 源代码：https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSSQLServerRotationSingleUser/lambda _function.py
- 依赖关系：Pymssql 2.2.2

Amazon RDS Microsoft SQLServer 交替用户

- 模板名称： SecretsManagerRDSSQLServerRotationMultiUser
- 轮换策略： the section called “交替用户”。
- 预期的 **`SecretString`** 结构：the section called “Amazon RDS 和 Aurora 凭证”。
- 源代码：https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSSQLServerRotationMultiUser/lambda _function.py
- 依赖关系：Pymssql 2.2.2

Amazon DocumentDB（与 MongoDB 兼容）

Amazon DocumentDB 单个用户

- 模板名称： SecretsManagerMongoDBRotationSingleUser
- 轮换策略： the section called “单用户”。

- 预期的 **`SecretString`** 结构：the section called “Amazon DocumentDB 凭证”。
- 源代码：https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerMongoDBRotationSingleUser/lambda _function.py
- 依赖关系：Pymongo 3.2

Amazon DocumentDB 交替用户

- 模板名称： SecretsManagerMongoDBRotationMultiUser
- 轮换策略： the section called “交替用户”。
- 预期的 **`SecretString`** 结构：the section called “Amazon DocumentDB 凭证”。
- 源代码：https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerMongoDBRotationMultiUser/lambda _function.py
- 依赖关系：Pymongo 3.2

Amazon Redshift

Amazon Redshift 单用户

- 模板名称： SecretsManagerRedshiftRotationSingleUser
- 轮换策略： the section called “单用户”。
- 预期的 **`SecretString`** 结构：the section called “Amazon Redshift 凭证”。
- 源代码：https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRedshiftRotationSingleUser/lambda _function.py
- 依赖关系：PyGreSQL 5.0.7

Amazon Redshift 交替用户

- 模板名称： SecretsManagerRedshiftRotationMultiUser
- 轮换策略： the section called “交替用户”。
- 预期的 **`SecretString`** 结构：the section called “Amazon Redshift 凭证”。
- 源代码：https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRedshiftRotationMultiUser/lambda _function.py
- 依赖关系：PyGreSQL 5.0.7

Amazon Timestream for InfluxDB

要使用这些模板，请参阅《Amazon Timestream 开发人员指南》中的 Amazon Timestream for InfluxDB 如何使用密钥。

Amazon Timestream for InfluxDB 单用户

- 模板名称： SecretsManagerInflux DBRotation SingleUser
- 预期的 **`SecretString`** 结构：the section called “Amazon Timestream for InfluxDB 密钥结构”。
- 源代码：https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerInfluxDBRotationSingleUser/lambda _function.py
- 依赖项：InfluxDB 2.0 python 客户端

Amazon Timestream for InfluxDB 交替用户

- 模板名称： SecretsManagerInfluxDBRotationMultiUser
- 预期的 **`SecretString`** 结构：the section called “Amazon Timestream for InfluxDB 密钥结构”。
- 源代码：https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerInfluxDBRotationMultiUser/lambda _function.py
- 依赖项：InfluxDB 2.0 python 客户端

Amazon ElastiCache

要使用此模板，请参阅 Amazon 用户指南中的自动轮换 ElastiCache 用户密码。

- 模板名称： SecretsManagerElasticacheUserRotation
- 预期的 **`SecretString`** 结构：the section called “Amazon ElastiCache 凭证”。
- 源代码：https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerElasticacheUserRotation/lambda _function.py

Active Directory

Active Directory 凭证

- 模板名称： SecretsManagerActiveDirectoryRotationSingleUser

- 预期的 **`SecretString`** 结构：the section called “Active Directory 凭证”。
- 源代码：https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerActiveDirectoryRotationSingleUser/lambda _function.py

Active Directory keytab

- 模板名称： SecretsManagerActiveDirectoryAndKeytabRotationSingleUser
- 预期的 **`SecretString`** 结构：the section called “Active Directory 凭证”。
- 源代码：https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerActiveDirectoryAndKeytabRotationSingleUser/lambda _function.py
- 依赖项：msktutil

其他密钥类型

Secrets Manager 提供此模板作为您为任何类型密钥创建轮换函数的起点。

- 模板名称： SecretsManagerRotationTemplate
- 源代码：https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRotationTemplate/lambda _function.py

Lambda 轮换函数的执行角色权限 AWS Secrets Manager

对于 the section called “通过 Lambda 函数进行轮换”，当 Secrets Manager 使用 Lambda 函数轮换密钥时，Lambda 将代入 IAM 执行角色并将这些凭证提供给 Lambda 函数代码。有关如何设置自动轮换的说明，请参阅：

- 自动轮换数据库密钥（控制台）
- 自动轮换非数据库密钥（控制台）
- 自动轮换（AWS CLI）

以下示例显示了适用于 Lambda 轮换函数执行角色的内联策略。要创建执行角色并附加权限策略，请参阅 AWS Lambda 执行角色。

示例：

- 适用于 Lambda 轮换函数执行角色的策略

- 适用于客户托管密钥的策略语句
- 适用于交替用户策略的策略语句

适用于 Lambda 轮换函数执行角色的策略

以下示例策略允许轮换函数：

- 运行 Secrets Manager 的操作*SecretARN*。
- 创建新密码。
- 如果数据库或服务在 VPC 中运行，则设置所需的配置。请参阅配置 Lambda 函数以访问 VPC 中的资源。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "secretsmanager:UpdateSecretVersionStage"
            ],
            "Resource": "arn:aws:secretsmanager:us-
east-1:123456789012:secret:secretName-AbCdEf"
    },
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetRandomPassword"
            ],
            "Resource": "*"
        },
        {
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterface",
                "ec2:DescribeNetworkInterfaces",
```

```
                "ec2:DetachNetworkInterface"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

适用于客户托管密钥的策略语句

如果密钥使用 AWS 托管式密钥 `aws/secretsmanager` 以外的 KMS 密钥进行加密，则您需要向 Lambda 执行角色授予使用该密钥的权限。您可以使用 SecretARN 加密上下文 来限制解密函数的使用，从而确保轮换函数角色只能解密其负责轮换的密钥。以下示例演示了要添加到执行角色策略中，以使用 KMS 密钥将密钥解密的语句。

```
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey"
            ],
            "Resource": "KMSKeyARN"
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:SecretARN": "SecretARN"
                }
            }
        }
```

要对使用客户托管密钥加密的多个密钥使用轮换函数，请添加如下示例所示的语句以允许执行角色解密该密钥。

```
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey"
            ],
            "Resource": "KMSKeyARN"
```

```
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:SecretARN": [
                        "arn1",
                        "arn2"
                    ]
                }
            }
        }
```

适用于交替用户策略的策略语句

有关交替用户轮换策略的信息，请参阅 the section called “Lambda 函数轮换策略”。

对于包含 Amazon RDS 凭证的密钥，如果您使用的是交替用户策略，并且超级用户密钥由 Amazon RDS 管理，则还必须允许轮换函数在 APIs Amazon RDS 上以只读方式调用，这样它才能获取数据库的连接信息。我们建议您附上 AWS 托管政策 Amazon RDSRead OnlyAccess。

以下示例策略允许函数：

- 运行 Secrets Manager 的操作*SecretARN*。
- 在超级用户密钥中检索凭证。Secrets Manager 会使用超级用户密钥中的凭证更新轮换密钥中的凭证。
- 创建新密码。
- 如果数据库或服务在 VPC 中运行，则设置所需的配置。有关更多信息，请参阅配置 Lambda 函数以访问 VPC 中的资源。

JSON

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

```
            ],
            "Resource": "arn:aws:secretsmanager:us-
east-1:123456789012:secret:secretName-AbCdEf"
    },
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource": "arn:aws:secretsmanager:us-
east-1:123456789012:secret:secretName-AbCdEf"
    },
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetRandomPassword"
            ],
            "Resource": "*"
        },
        {
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterface",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DetachNetworkInterface"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

AWS Lambda 轮换功能的网络接入

对于 the section called “通过 Lambda 函数进行轮换”，当 Secrets Manager 使用 Lambda 函数轮换密钥时，Lambda 轮换函数必须能够访问该密钥。如果您的密钥包含凭证，则 Lambda 函数还必须能够访问这些凭证的来源，例如数据库或服务。

访问密钥

Lambda 轮换功能必须能够访问 Secrets Manager 端点。如果您的 Lambda 函数可以访问互联网，则可以使用公共终端节点。若要查找端点，请参阅 the section called “Secrets Manager 端点”。

如果您的 Lambda 函数在不具备互联网访问权限的 VPC 中运行，我们建议您在 VPC 内配置 Secrets Manager 服务私有终端节点。然后，您的 VPC 可以拦截发往公共区域终端节点的请求并将其重定向到私有终端节点。有关更多信息，请参阅 VPC 端点（AWS PrivateLink）。

或者，您可以通过向 VPC 添加 NAT 网关或互联网网关（这将允许来自您 VPC 的流量访问公有端点），允许 Lambda 函数访问 Secrets Manager 公有端点。这会使 VPC 面临一定的风险，因为网关的 IP 地址可能会受到来自公有 Internet 的攻击。

（可选）访问数据库或服务

对于诸如 API 密钥之类的密钥，无需随密钥更新源数据库或服务。

如果您的数据库或服务在 VPC 中的 Amazon EC2 实例上运行，我们建议您将您的 Lambda 函数配置为在同一 VPC 中运行。然后轮换功能可以直接与您的服务通信。有关更多信息，请参阅配置 VPC 访问。

要允许 Lambda 函数访问数据库或服务，您必须确保附加到 Lambda 轮换函数的安全组允许与数据库或服务的出站连接。您还必须确保附加到数据库或服务的安全组允许来自 Lambda 轮换函数进行入站连接。

AWS Secrets Manager 轮换问题排查

对于许多服务，Secrets Manager 使用 Lambda 函数来轮换密钥。有关更多信息，请参阅 the section called “通过 Lambda 函数进行轮换”。Lambda 轮换函数与拥有密钥的数据库或服务以及 Secrets Manager 交互。在轮换没有按预期方式运行时，应先检查日 CloudWatch 志。

> **Note**
>
> 某些服务可以为您管理密钥，包括管理自动轮换。有关更多信息，请参阅 the section called “托管轮换”。

主题

- 如何对函数中的密钥轮换失败进行 AWS Lambda 故障排除
- “在环境变量中找到凭证”之后没有活动
- “createSecret”之后没有活动
- 错误：“不允许访问 KMS”
- 错误：“Key is missing from secret JSON（密钥 JSON 中缺少密钥）”

- 错误：“setSecret: Unable to log into database（setSecret：无法登录数据库）”
- 错误：“无法导入模块‘lambda_function’”
- 将现有的轮换函数版本从 Python 3.7 升级到 Python 3.9
- AWS Lambda 秘密轮换PutSecretValue失败
- 错误：“<arn>在<a rotation>步骤中执行 lambda 时出错”

如何对函数中的密钥轮换失败进行 AWS Lambda 故障排除

如果您的 Lambda 函数遇到密钥轮换失败，请按照以下步骤排查并解决问题。

可能的原因

- Lambda 函数并发执行不足
- 由于轮换期间有多个 API 调用而导致的竞赛条件
- Lambda 函数逻辑不正确
- Lambda 函数和数据库之间的网络问题

一般故障排除步骤

1. 分析 CloudWatch 日志：
   - 在 Lambda 函数日志中查找特定的错误消息或意外行为
   - 验证是否正在尝试所有旋转步骤（CreateSecretSetSecretTestSecret、、、FinishSecret）
2. 在轮换期间查看 API 调用：
   - 避免在 Lambda 轮换期间对密钥进行变更 API 调用
   - 确保RotateSecret和PutSecretValue通话之间没有竞争条件
3. 验证 Lambda 函数逻辑：
   - 确认你使用的是最新的密钥轮换 AWS 示例代码
   - 如果使用自定义代码，请查看它以正确处理所有轮换步骤
4. 检查网络配置：
   - 验证安全组规则是否允许 Lambda 函数访问数据库
   - 确保 Secrets Manager 能够正确访问 VPC 终端节点或公有终端节点
5. 测试密钥版本：

- 验证密钥的 AWSCURRENT 版本是否允许访问数据库
- 检查 AWSPREVIOUS 或 AWSPENDING 版本是否有效

6. 清除待处理的轮换：
   - 如果轮换一直失败，请清除 AWSPENDING 暂存标签并重试轮换
7. 检查 Lambda 并发设置：
   - 验证并发设置是否适合您的工作负载
   - 如果您怀疑存在并发问题，请参阅 “解决与并发相关的轮换故障” 部分

“在环境变量中找到凭证”之后没有活动

如果“在环境变量中找到凭证”之后没有活动，并且任务持续时间很长，例如默认 Lambda 超时为 30000 毫秒，则 Lambda 函数可能会在尝试访问 Secrets Manager 端点时超时。

Lambda 轮换功能必须能够访问 Secrets Manager 端点。如果您的 Lambda 函数可以访问互联网，则可以使用公共终端节点。若要查找端点，请参阅 the section called “Secrets Manager 端点”。

如果您的 Lambda 函数在不具备互联网访问权限的 VPC 中运行，我们建议您在 VPC 内配置 Secrets Manager 服务私有终端节点。然后，您的 VPC 可以拦截发往公共区域终端节点的请求并将其重定向到私有终端节点。有关更多信息，请参阅 VPC 端点（AWS PrivateLink）。

或者，您可以通过向 VPC 添加 NAT 网关或互联网网关（这将允许来自您 VPC 的流量访问公有端点），允许 Lambda 函数访问 Secrets Manager 公有端点。这会使 VPC 面临一定的风险，因为网关的 IP 地址可能会受到来自公有 Internet 的攻击。

“createSecret”之后没有活动

以下问题可能会在 createSecret 之后导致轮换停止：

VPC 网络 ACLs 不允许 HTTPS 流量进出。

有关更多信息，请参阅 Amazon VPC 用户指南 ACLs中的使用网络控制子网的流量。

Lambda 函数超时配置过短，无法执行任务。

有关更多信息，请参阅《AWS Lambda 开发人员指南》中的配置 Lambda 函数选项。

Secrets Manager VPC CIDRs 端点不允许在分配的安全组入口处使用 VPC。

有关更多信息，请参阅《Amazon VPC 用户指南》中的使用安全组控制到资源的流量。

Secrets Manager VPC 端点策略不允许 Lambda 使用 VPC 端点。

有关更多信息，请参阅 the section called “VPC 端点（AWS PrivateLink）”。

该密钥使用交替用户轮换策略，超级用户密钥由 Amazon RDS 管理，并且 Lambda 函数无法访问 RDS API。

对于超级用户秘密由其他 AWS 服务管理的交替用户轮换策略，Lambda 轮换函数必须能够调用服务端点以获取数据库连接信息。我们建议您为数据库服务配置 VPC 端点。有关更多信息，请参阅：

- 《Amazon RDS 用户指南》中的 Amazon RDS API 和接口 VPC 端点。
- 《Amazon Redshift 管理指南》中的使用 VPC 端点。

错误：“不允许访问 KMS”

如果您看到 `ClientError: An error occurred (AccessDeniedException) when calling the GetSecretValue operation: Access to KMS is not allowed`，则轮换函数无权使用密钥加密所用的 KMS 密钥来将密钥解密。权限策略中可能存在将加密上下文限定为特定密钥的条件。有关所需权限的信息，请参阅 the section called “适用于客户托管密钥的策略语句”。

错误：“Key is missing from secret JSON（密钥 JSON 中缺少密钥）”

Lambda 轮换函数要求密钥值采用特定的 JSON 结构。如果显示此错误，则 JSON 可能缺少轮换函数尝试访问的密钥。有关每种密钥类型的 JSON 结构的信息，请参阅 the section called “密钥的 JSON 结构”。

错误：“setSecret: Unable to log into database（setSecret：无法登录数据库）”

以下问题可能导致此错误：

轮换函数无法访问数据库。

如果任务持续时间过长，例如超过 5000 毫秒，则 Lambda 轮换函数可能无法通过网络访问数据库。

如果数据库或服务在 VPC 中的 Amazon EC2 实例上运行，建议将 Lambda 函数配置为在同一 VPC 中运行。然后轮换功能可以直接与您的服务通信。有关更多信息，请参阅配置 VPC 访问。

要允许 Lambda 函数访问数据库或服务，您必须确保附加到 Lambda 轮换函数的安全组允许与数据库或服务的出站连接。您还必须确保附加到数据库或服务的安全组允许来自 Lambda 轮换函数进行入站连接。

密钥中的凭证有误。

如果任务持续时间过短，则 Lambda 轮换函数可能无法使用密钥中的凭证进行身份验证。使用 AWS CLI 命令使用密钥AWSCURRENT和AWSPREVIOUS版本中的信息手动登录，检查凭据get-secret-value。

数据库使用 **scram-sha-256** 加密密码。

如果您的数据库是 Aurora PostgreSQL 版本 13 或更高版本，并且使用 scram-sha-256 加密密码，但轮换函数使用不支持 scram-sha-256 的 libpq 版本 9 或更旧版本，则轮换函数无法连接到数据库。

确定哪些数据库用户使用 **scram-sha-256** 加密

- 请参阅博客 RDS for PostgreSQL 13 中的 SCRAM 身份验证中的检查使用非 SCRAM 密码的用户。

确定您的轮换函数使用哪个 **libpq** 版本

1. 在基于 Linux 的计算机上，在 Lambda 控制台上导航到您的轮换函数并下载部署包。将 zip 文件解压缩到工作目录中。
2. 在命令行上，在工作目录中运行：

   ```
   readelf -a libpq.so.5 | grep RUNPATH
   ```

3. 如果看到字符串 *PostgreSQL-9.4.x* 或任何低于 10 的主要版本，则轮换函数不支持 scram-sha-256。
   - 不支持 scram-sha-256 的轮换函数的输出：

     ```
     0x000000000000001d (RUNPATH) Library runpath: [/
     local/p4clients/pkgbuild-a1b2c/workspace/build/
     PostgreSQL/PostgreSQL-9.4.x_client_only.123456.0/AL2_x86_64/
     DEV.STD.PTHREAD/build/private/tmp/brazil-path/build.libfarm/lib:/
     local/p4clients/pkgbuild-a1b2c/workspace/src/PostgreSQL/build/
     private/install/lib]
     ```

   - 支持 scram-sha-256 的轮换函数的输出：

     ```
     0x000000000000001d (RUNPATH) Library runpath: [/
     local/p4clients/pkgbuild-a1b2c/workspace/build/
     PostgreSQL/PostgreSQL-10.x_client_only.123456.0/AL2_x86_64/
     ```

```
DEV.STD.PTHREAD/build/private/tmp/brazil-path/build.libfarm/lib:/
local/p4clients/pkgbuild-a1b2c/workspace/src/PostgreSQL/build/
private/install/lib]
```

> **Note**
>
> 如果您在 2021 年 12 月 30 日之前设置了自动密钥轮换，则您的轮换函数捆绑了`libpq`不支持的`scram-sha-256`早期版本。为了支持 `scram-sha-256`，您需要重新创建您的轮换函数。

数据库需要 SSL/TLS 访问权限。

如果您的数据库需要 SSL/TLS 连接，但轮换函数使用了未加密的连接，则轮换函数无法连接到数据库。适用于 Amazon RDS（Oracle 和 Db2 除外）和 Amazon DocumentDB 的轮换函数将自动使用安全套接字层 (SSL) 或传输层安全性协议 (TLS) 来连接到数据库（如果可用）。否则，他们将使用未加密的连接。

> **Note**
>
> 如果您在 2021 年 12 月 20 日之前设置了自动密钥轮换，则您的轮换函数可能基于不支持的SSL/TLS. To support connections that use SSL/TLS较早模板。

确定您的轮换函数的创建时间

1. 在 Secrets Manager 控制台中 https://console.aws.amazon.com/secretsmanager/，打开您的密钥。在 Rotation configuration（轮换配置）中的 Lambda rotation function（Lambda 轮换函数）下，您将看到 Lambda function ARN（Lambda 函数 ARN），例如，`arn:aws:lambda:aws-region:123456789012:function:SecretsManagerMyRotationFunction` 。在此示例 `SecretsManagerMyRotationFunction` 中，从 ARN 末尾复制函数名称。
2. 在 AWS Lambda 控制台中的 Functions（函数）下 https://console.aws.amazon.com/lambda/，将您的 Lambda 函数名称粘贴到搜索框中，按 Enter 键，然后选择相应的 Lambda 函数。
3. 在函数详细信息页面中，在 Configuration（配置）选项卡上的 Tags（标签）下，复制键 aws:cloudformation:stack-name 旁边的值。

4. 在 AWS CloudFormation 控制台 https://console.aws.amazon.com/cloudformation 中的堆栈下，将键值粘贴到搜索框中，然后选择 Enter 键。
5. 堆栈列表将进行筛选，以便只显示创建 Lambda 轮换函数的堆栈。在 Created date（创建日期）列中，查看堆栈的创建日期。这是 Lambda 轮换函数的创建日期。

错误："无法导入模块'lambda_function'"

如果您运行的是早期版本的 Lambda 函数，且该函数是从 Python 3.7 自动升级到更新版本的 Python 的，则可能会遇到此错误。要解决此错误，您可以将 Lambda 函数版本改回 Python 3.7，然后 the section called "将现有的轮换函数版本从 Python 3.7 升级到 Python 3.9"。要了解更多信息，请参阅 AWS re:Post 中的 为什么我的 Secrets Manager Lambda 函数轮换失败并出现"找不到 pg 模块"错误？。

将现有的轮换函数版本从 Python 3.7 升级到 Python 3.9

2022 年 11 月之前创建的部分轮换函数使用 Python 3.7。适用于 Python 的 AWS SDK 已于 2023 年 12 月停止支持 Python 3.7。有关更多信息，请参阅 Python 支持策略更新 AWS SDKs 和工具。要切换到使用 Python 3.9 的新轮换函数，可以在现有轮换函数中添加运行时系统属性或重新创建轮换函数。

查找使用 Python 3.7 的 Lambda 轮换函数

1. 登录 AWS Management Console 并打开 AWS Lambda 控制台，网址为https://console.aws.amazon.com/lambda/。
2. 在函数列表中，筛选 **`SecretsManager`**。
3. 在筛选后的函数列表中，在运行时系统下，查找 Python 3.7。

升级到 Python 3.9：

- 选项 1：使用 AWS CloudFormation重新创建轮换函数
- 选项 2：使用更新现有轮换函数的运行时系统 AWS CloudFormation
- 选项 3： AWS CDK 用户升级 CDK 库

选项 1：使用 AWS CloudFormation重新创建轮换函数

当您使用 Secrets Manager 控制台开启轮换功能时，Secrets Manager 会使用 AWS CloudFormation 创建必要的资源，包括 Lambda 轮换函数。如果您使用控制台开启轮换功能，或者使用 AWS

CloudFormation 堆栈创建了轮换函数，则可以使用相同的 AWS CloudFormation 堆栈重新创建具有新名称的轮换函数。新函数将使用最新版本的 Python。

查找创建了轮换函数的 AWS CloudFormation 堆栈

- 在 Lambda 函数的详细信息页面的配置选项卡上，选择标签。查看 aws:cloudformation:stack-id 旁的 ARN。

  堆栈名称已嵌入在 ARN 中，如下例所示。

  - ARN：`arn:aws:cloudformation:us-west-2:408736277230:stack/`**`SecretsManagerRDSMySQLRotationSingleUser5c2-SecretRotationScheduleHostedRotationLambda`**`-3CUDHZMDMBO8/79fc9050-2eef-11ed-`
  - 堆栈名称：**`SecretsManagerRDSMySQLRotationSingleUser5c2-SecretRotationScheduleHostedRotationLambda`**

重新创建轮换函数（AWS CloudFormation）

1. 在中 AWS CloudFormation，按名称搜索堆栈，然后选择更新。

   如果出现建议您更新根堆栈的对话框，请选择转到根堆栈，然后选择更新。
2. 在更新堆栈页面上的准备模板下，选择在应用程序编辑器中编辑，然后在在应用程序编辑器中编辑模板下，选择在应用程序编辑器中编辑按钮。
3. 在应用程序编辑器中，执行以下操作：

   a. 在模板代码中，在 `SecretRotationScheduleHostedRotationLambda` 中将 `"functionName": "SecretsManagerTestRotationRDS"` 的值替换为新的函数名称，例如在 JSON 中为 **`"functionName": "SecretsManagerTestRotationRDSupdated"`**

   b. 选择更新模板。

   c. 在继续使用 AWS CloudFormation 对话框中，选择确认并继续使用 AWS CloudFormation。
4. 继续完成 AWS CloudFormation 堆栈工作流，然后选择提交。

选项 2：使用更新现有轮换函数的运行时系统 AWS CloudFormation

当您使用 Secrets Manager 控制台开启轮换功能时，Secrets Manager 会使用 AWS CloudFormation 创建必要的资源，包括 Lambda 轮换函数。如果您使用控制台开启轮换功能，或者使用 AWS

CloudFormation 堆栈创建了轮换函数，则可以使用相同的 AWS CloudFormation 堆栈更新轮换函数的运行时系统。

查找创建了轮换函数的 AWS CloudFormation 堆栈

- 在 Lambda 函数的详细信息页面的配置选项卡上，选择标签。查看 aws:cloudformation:stack-id 旁的 ARN。

  堆栈名称已嵌入在 ARN 中，如下例所示。

  - ARN：`arn:aws:cloudformation:us-west-2:408736277230:stack/`**`SecretsManagerRDSMySQLRotationSingleUser5c2-SecretRotationScheduleHostedRotationLambda`**`-3CUDHZMDMBO8/79fc9050-2eef-11ed-`
  - 堆栈名称：**`SecretsManagerRDSMySQLRotationSingleUser5c2-SecretRotationScheduleHostedRotationLambda`**

更新轮换函数的运行时系统（AWS CloudFormation）

1. 在中 AWS CloudFormation，按名称搜索堆栈，然后选择更新。

   如果出现建议您更新根堆栈的对话框，请选择转到根堆栈，然后选择更新。
2. 在更新堆栈页面上的准备模板下，选择在应用程序编辑器中编辑，然后在在应用程序编辑器中编辑模板下，选择在应用程序编辑器中编辑按钮。
3. 在应用程序编辑器中，执行以下操作：

   a. 在模板 JSON 中，对于 `SecretRotationScheduleHostedRotationLambda`，在 `Properties` 下的 `Parameters` 下方，添加 **`"runtime": "python3.9"`**。

   b. 选择更新模板。

   c. 在继续使用 AWS CloudFormation 对话框中，选择确认并继续使用 AWS CloudFormation。
4. 继续完成 AWS CloudFormation 堆栈工作流，然后选择提交。

选项 3： AWS CDK 用户升级 CDK 库

如果您使用 v2.94.0 AWS CDK 之前版本的设置了密钥轮换，则可以通过升级到 v2.94.0 或更高版本来更新 Lambda 函数。有关更多信息，请参见 AWS Cloud Development Kit (AWS CDK) v2 开发人员指南。

AWS Lambda 秘密轮换**PutSecretValue**失败

如果您在 Secrets Manager 中使用代入角色或跨账户轮换，并且在中发现了一个 CloudTrail 带有以下消息RotationFailed的事件：*LAMBDA_ARN*.Lambda 未创建 Secret 的待处理密钥*SECRET_ARN* 版本*VERSION_ID* 移除暂存标签并重启轮换，则需要更新 Lambda 函数才能使用AWSPENDING该参数。RotationToken

更新 Lambda 旋转函数以包括 **RotationToken**

1. 下载 Lambda 函数代码
   - 打开 Lambda 控制台
   - 在导航窗格中，选择函数
   - 为函数名称选择您的 Lambda 密钥轮换函数
   - 在 “下载” 中，选择函数代码.zip、AWS SAM 文件、两者中的一个
   - 选择 “确定” 将该功能保存在本地计算机上。
2. 编辑 Lambda_handler

   在跨账户轮换的 create_secret 步骤中加入 rotation_token 参数：

```
def lambda_handler(event, context):
    """Secrets Manager Rotation Template

    This is a template for creating an AWS Secrets Manager rotation lambda

    Args:
        event (dict): Lambda dictionary of event parameters. These keys must
 include the following:
            - SecretId: The secret ARN or identifier
            - ClientRequestToken: The ClientRequestToken of the secret version
            - Step: The rotation step (one of createSecret, setSecret, testSecret,
 or finishSecret)
            - RotationToken: the rotation token to put as parameter for
 PutSecretValue call

        context (LambdaContext): The Lambda runtime information

    Raises:
        ResourceNotFoundException: If the secret with the specified arn and stage
 does not exist
```

```
        ValueError: If the secret is not properly configured for rotation

        KeyError: If the event parameters do not contain the expected keys

    """
    arn = event['SecretId']
    token = event['ClientRequestToken']
    step = event['Step']
    # Add the rotation token
    rotation_token = event['RotationToken']

    # Setup the client
    service_client = boto3.client('secretsmanager',
endpoint_url=os.environ['SECRETS_MANAGER_ENDPOINT'])

    # Make sure the version is staged correctly
    metadata = service_client.describe_secret(SecretId=arn)
    if not metadata['RotationEnabled']:
        logger.error("Secret %s is not enabled for rotation" % arn)
        raise ValueError("Secret %s is not enabled for rotation" % arn)
    versions = metadata['VersionIdsToStages']
    if token not in versions:
        logger.error("Secret version %s has no stage for rotation of secret %s." %
(token, arn))
        raise ValueError("Secret version %s has no stage for rotation of secret
%s." % (token, arn))
    if "AWSCURRENT" in versions[token]:
        logger.info("Secret version %s already set as AWSCURRENT for secret %s." %
(token, arn))
        return
    elif "AWSPENDING" not in versions[token]:
        logger.error("Secret version %s not set as AWSPENDING for rotation of
secret %s." % (token, arn))
        raise ValueError("Secret version %s not set as AWSPENDING for rotation of
secret %s." % (token, arn))
    # Use rotation_token
    if step == "createSecret":
        create_secret(service_client, arn, token, rotation_token)

    elif step == "setSecret":
        set_secret(service_client, arn, token)

    elif step == "testSecret":
```

```
        test_secret(service_client, arn, token)

    elif step == "finishSecret":
        finish_secret(service_client, arn, token)

    else:
        raise ValueError("Invalid step parameter")
```

3. 编辑create_secret代码

   修改create_secret函数以接受并使用以下rotation_token参数：

```
# Add rotation_token to the function
def create_secret(service_client, arn, token, rotation_token):
"""Create the secret

This method first checks for the existence of a secret for the passed in token. If
 one does not exist, it will generate a
new secret and put it with the passed in token.

Args:
service_client (client): The secrets manager service client

arn (string): The secret ARN or other identifier

token (string): The ClientRequestToken associated with the secret version

rotation_token (string): the rotation token to put as parameter for PutSecretValue
 call

Raises:
ResourceNotFoundException: If the secret with the specified arn and stage does not
 exist

"""
# Make sure the current secret exists
service_client.get_secret_value(SecretId=arn, VersionStage="AWSCURRENT")

# Now try to get the secret version, if that fails, put a new secret
try:
service_client.get_secret_value(SecretId=arn, VersionId=token,
 VersionStage="AWSPENDING")
```

```
logger.info("createSecret: Successfully retrieved secret for %s." % arn)
except service_client.exceptions.ResourceNotFoundException:
# Get exclude characters from environment variable
exclude_characters = os.environ['EXCLUDE_CHARACTERS'] if 'EXCLUDE_CHARACTERS' in
 os.environ else '/@"\'\\'
# Generate a random password
passwd = service_client.get_random_password(ExcludeCharacters=exclude_characters)

# Put the secret, using rotation_token
service_client.put_secret_value(SecretId=arn, ClientRequestToken=token,
 SecretString=passwd['RandomPassword'], VersionStages=['AWSPENDING'],
 RotationToken=rotation_token)
logger.info("createSecret: Successfully put secret for ARN %s and version %s." %
 (arn, token))
```

4. 上传更新的 Lambda 函数代码

   更新您的 Lambda 函数代码后，将其上传以轮换您的密钥。

错误："*<arn>*在*<a rotation>*步骤中执行 lambda 时出错"

如果您遇到间歇性的密钥轮换失败，您的 Lambda 函数陷入了集合循环，例如CreateSecret介于SetSecret和之间，则问题可能与并发设置有关。

并发故障排除步骤

> **Warning**
>
> 由于 Lambda 函数的执行线程不足，将预配置的并发参数设置为低于 10 的值可能会导致限制。有关更多信息，请参阅《开发人员指南》中的了解预留并发和预配置并发。 AWS Lambda AWS Lambda

1. 检查并调整 Lambda 并发设置：
   - 验证`reserved_concurrent_executions`是否设置得太低（例如，1）
   - 如果使用预留并发，请将其设置为至少 10
   - 考虑使用未预留的并发以获得更大的灵活性
2. 对于预配置并发：

- 不要显式设置预配置的并发参数（例如，在 Terraform 中）。
- 如果必须进行设置，请使用至少 10 的值。
- 彻底测试以确保所选值适用于您的用例。

3. 监控和调整并发度：

- 使用以下公式计算并发度：并发度 =（每秒平均请求数）*（以秒为单位的平均请求持续时间）。有关更多信息，请参阅估算预留并发量。
- 观察并记录轮换期间的值，以确定适当的并发设置。
- 设置低并发值时要小心。如果没有足够的可用执行线程，它们可能会导致限制。

有关配置 Lambda 并发的更多信息，请参阅开发人员指南中的配置预留并发和配置预配置并发。 AWS Lambda

轮换计划

Secrets Manager 会在您设置的轮换时段内按照计划轮换您的密钥。要设置计划和时段，请使用 cron() 或 rate() 表达式以及时段持续时间。Secrets Manager 将在轮换时段内随时轮换密钥。您可以在短至一小时的轮换时段内每四小时轮换一次密钥。

要启用轮换，请参阅：

- the section called “托管轮换”
- the section called “自动轮换数据库密钥（控制台）”
- the section called “自动轮换非数据库密钥（控制台）”

Secrets Manager 轮换计划使用 UTC 时区。

轮换时段

Secrets Manager 轮换时段与维护时段类似。当您想要轮换密钥时，可以设置轮换时段，Secrets Manager 会在轮换时段内的某个时间轮换您的密钥。

Secrets Manager 轮换时段总是按小时开始。对于使用 `rate()` 表达式（以天为单位）的轮换计划，轮换时段从午夜开始。您可以使用 `cron()` 表达式设置轮换时段的开始时间。有关示例，请参阅 the section called “Cron 表达式”。

默认情况下，对于按小时计算的轮换计划，轮换时段在一小时后关闭；对于按天计算的轮换计划，轮换时段在一天结束时关闭。

要更改轮换时段的长度，请设置时段持续时间。您可以将轮换时段设置为短至一小时。该轮换时段不得延伸到下一个轮换时段。换句话说，对于以小时为单位的轮换计划，请确认轮换时段小于或等于轮换之间的小时数。对于以天为单位的轮换计划，请确认起始小时数加上时段持续时间小于或等于 24 小时。

Rate 表达式

Secrets Manager 速率表达式采用以下格式，其中*Value*为正整数`hour`，*Unit*可以是`hours`day、、或`days`：

```
rate(Value Unit)
```

您可以每四小时轮换一次密钥。最长轮换周期为 999 天。示例：

- `rate(4 hours)` 意味着密钥每四小时轮换一次。
- `rate(1 day)` 意味着密钥每天轮换一次。
- `rate(10 days)` 意味着密钥每 10 天轮换一次。

Cron 表达式

Secrets Manager cron 表达式具有以下格式：

```
cron(Minutes Hours Day-of-month Month Day-of-week Year)
```

包含小时增量的 cron 表达式每天都会重置。例如，`cron(0 4/12 * * ? *)` 表示凌晨 4:00、下午 4:00，然后是第二天凌晨 4:00、下午 4:00。Secrets Manager 轮换计划使用 UTC 时区。

示例计划	Expression
每八小时一次，从午夜开始。	`cron(0 /8 * * ? *)`
每八小时一次，从早上 8:00 开始。	`cron(0 8/8 * * ? *)`
每十小时一次，从凌晨 2:00 开始。 轮换时段将从 2:00、12:00 和 22:00 开始，然后在第二天的 2:00、12:00 和 22:00 进行。	`cron(0 2/10 * * ? *)`

示例计划	Expression
每天上午 10:00。	`cron(0 10 * * ? *)`
每星期六下午 6:00。	`cron(0 18 ? * SAT *)`
每月第 1 天上午 8:00。	`cron(0 8 1 * ? *)`
每三个月第一个星期日凌晨 1:00。	`cron(0 1 ? 1/3 SUN#1 *)`
每月最后一天下午 5:00。	`cron(0 17 L * ? *)`
星期一到星期五上午 8:00。	`cron(0 8 ? * MON-FRI *)`
每月第 1 天和第 15 天下午 4:00。	`cron(0 16 1,15 * ? *)`
每月第一个星期日午夜。	`cron(0 0 ? * SUN#1 *)`
从一月份开始，每 11 个月的第一个星期一的午夜。	`cron(0 0 ? 1/11 2#1 *)`

Secrets Manager 中的 Cron 表达式要求

Secrets Manager 对可以用于 cron 表达式的内容有一些限制。Secrets Manager 的 cron 表达式的分钟字段必须填写 0，因为 Secrets Manager 轮换时段在整点开始。其年份字段必须填写 *，因为 Secrets Manager 不支持相隔一年以上的轮换计划。下表显示了可以使用的选项。

字段	值	通配符
Minutes	必须为 0	无
小时	0–23	使用 /（正斜杠）指定增量。例如，`2/10` 意味着从凌晨 2:00 开始每 10 小时一次。您可以每四小时轮换一次密钥。
D ay-of-month	1-31	使用 ,（逗号）包含其他值。例如，`1,15` 当前当月的第 1 天和第 15 天。

字段	值	通配符
		使用 - (短划线) 指定范围。例如，1–15 表示当月的第 1 天到第 15 天。 使用 * (星号) 包含该字段中的所有值。例如，* 表示当月的每一天。 ? (问号) 通配符用于指定一个或另一个。您无法在同一 cron 表达式中为 Day-of-month 和 Day-of-week 字段同时指定值。如果您在其中一个字段中指定了值，则必须在另一个字段中使用 ? (问号)。 使用 / (正斜杠) 指定增量。例如，1/2 表示从第 1 天开始每两天一次，换句话说，第 1 天、第 3 天、第 5 天，依此类推。 使用 L 指定当月的最后一天。 使用 ***DAY***L 指定该月中最后一个命名的日期。例如，SUNL 表示当月的最后一个星期日。

字段	值	通配符
月份	1-12 或 JAN-DEC	使用 , (逗号) 包含其他值。例如 , `JAN,APR,JUL,OCT` 表示一月、四月、七月和十月。 使用 - (短划线) 指定范围。例如 , `1-3` 表示一年的第 1 个月至第 3 个月。 使用 * (星号) 包含该字段中的所有值。例如 , `*` 表示每个月。 使用 / (正斜杠) 指定增量。例如 , `1/3` 表示每三个月一次 , 从第 1 个月开始 , 即第 1、4、7 和 10 个月。

字段	值	通配符
D ay-of-week	1-7 或 SUN-SAT	使用 # 指定某个月内一周的星期几。例如，TUE#3 表示该月的第三个星期二。 使用 ,（逗号）包含其他值。例如，1,4 表示一周的第一天和第四天。 使用 -（短划线）指定范围。例如，1–4 表示一周的第 1 天到第 4 天。 使用 *（星号）包含该字段中的所有值。例如，* 表示一周的每一天。 ?（问号）通配符用于指定一个或另一个。您无法在同一 cron 表达式中为 Day-of-month 和 Day-of-week 字段同时指定值。如果您在其中一个字段中指定了值，则必须在另一个字段中使用 ?（问号）。 使用 /（正斜杠）指定增量。例如，1/2 表示一周的每隔一天，从第一天开始，即第 1 天、第 3 天、第 5 天和第 7 天。 使用 L 指定一周的最后一天。
Year	必须是 *	无

立即轮换 AWS Secrets Manager 密钥

您只能轮换已配置了轮换的密钥。要确定密钥是否已配置为轮换，请在控制台中查看密钥并向下滚动到 Rotation configuration（轮换配置）部分。如果 Rotation status（轮换状态）为 Enabled（启用），则密钥配置为轮换。如果不是，请参阅轮换 密钥。

要立即轮换密码（控制台）

1. 打开 Secrets Manager 控制台，网址为https://console.aws.amazon.com/secretsmanager/。
2. 选择您的密钥。
3. 在密钥详细信息页面上，在旋转配置下方，选择立即轮换密钥。
4. 在轮换密钥对话框中，选择轮换。

AWS CLI

Example 立即轮换密钥

以下 rotate-secret 示例将立即开始轮换。密钥必须已配置轮换。

```
$ aws secretsmanager rotate-secret \
    --secret-id MyTestSecret
```

查找未轮换的密钥

您可以使用 AWS Config 来评估您的密钥，以查看它们是否符合您的标准。您可以使用 AWS Config 规则定义对机密的内部安全和合规性要求。然后 AWS Config 可以识别不符合你规则的秘密。您还可以跟踪对密钥元数据、轮换配置、用于密钥加密的 KMS 密钥、Lambda 轮换函数以及与密钥关联的标签等进行的更改。

如果您的组织中有多个 AWS 账户 密钥，则可以聚合该配置和合规性数据。 AWS 区域 有关更多信息，请参阅 Multi-account Multi-Region data aggregation。

评测密钥是否正在轮换

1. 按照使用AWS Config 规则评估资源中的说明进行操作，并从以下规则中进行选择：
   - `secretsmanager-rotation-enabled-check` — 检查是否为存储在 Secrets Manager 中的密钥配置了轮换。

- `secretsmanager-scheduled-rotation-success-check`— 检查上次成功的轮换是否在配置的轮换频率内。检查的最低频率为每天。
- `secretsmanager-secret-periodic-rotation` — 检查是否已在指定的天数内轮换了密钥。

2. （可选）配置 AWS Config 为在密钥不合规时通知您。有关更多信息，请参阅AWS Config 发送至 Amazon SNS 主题的通知。

在 Secrets Manager 中取消自动轮换

如果为密钥配置了自动轮换，并且想要停止轮换，则可以取消轮换。

取消自动轮换

1. 打开 Secrets Manager 控制台，网址为https://console.aws.amazon.com/secretsmanager/。
2. 选择您的密钥。
3. 在密钥详细信息页上的轮换配置下，选择编辑轮换。
4. 在编辑轮换配置对话框中，关闭自动轮换，然后选择保存。

   Secrets Manager 会保留轮换配置信息，以便您在将来决定重新启用轮换时可以使用它。

AWS Secrets Manager 由其他 AWS 服务管理的机密

许多 AWS 服务在中存储和使用机密 AWS Secrets Manager。在某些情况下，这些密钥是托管密钥，这意味着创建这些密钥的服务可以帮助管理它们。例如，一些托管密钥包括托管轮换，因此您不必自己配置轮换。托管服务还可能限制您在没有恢复期的情况下更新密钥或删除它们，这有助于防止中断，因为托管服务依赖于密钥。

> **Note**
>
> 托管密钥只能由管理这些密钥的 AWS 服务创建。

托管密钥使用包括管理服务 ID 的命名约定来帮助识别它们。

```
Secret name: ServiceID!MySecret
Secret ARN : arn:aws:us-east-1:ServiceID!MySecret-a1b2c3
```

IDs 适用于管理机密的服务

- `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”
- `pcs` – the section called “AWS 并行计算服务”
- `rds` – the section called “Amazon RDS”
- `redshift` – the section called “Amazon Redshift”
- `sqlworkbench` – the section called “Amazon Redshift 查询编辑器 v2”

要查找由其他 AWS 服务管理的密钥，请参阅查找托管密钥。

有关使用密钥的服务的完整列表，请参阅 使用密钥的服务。

AWS 服务 那些使用 AWS Secrets Manager 秘密的

获取有关以下每个 AWS 服务 如何与 Secrets Manager 集成的信息。

- 如何 AWS App Runner 使用 AWS Secrets Manager
- AWS App2Container 如何使用 AWS Secrets Manager
- 如何 AWS AppConfig 使用 AWS Secrets Manager
- 亚马逊如何 AppFlow 使用 AWS Secrets Manager
- 如何 AWS AppSync 使用 AWS Secrets Manager
- Amazon Athena 如何使用 AWS Secrets Manager
- 亚马逊 Aurora 是如何使用的 AWS Secrets Manager
- 如何 AWS CodeBuild 使用 AWS Secrets Manager
- Amazon Data Firehose 如何使用 AWS Secrets Manager
- 如何 AWS DataSync 使用 AWS Secrets Manager
- 亚马逊如何 DataZone 使用 AWS Secrets Manager
- 如何 AWS Direct Connect 使用 AWS Secrets Manager
- 如何 AWS Directory Service 使用 AWS Secrets Manager
- Amazon DocumentDB（with MongoDB compatibility）如何使用 AWS Secrets Manager
- 如何 AWS Elastic Beanstalk 使用 AWS Secrets Manager
- Amazon 弹性容器注册表的使用方式 AWS Secrets Manager
- Amazon Elastic Container Service
- 亚马逊如何 ElastiCache 使用 AWS Secrets Manager
- 如何 AWS Elemental Live 使用 AWS Secrets Manager
- 如何 AWS Elemental MediaConnect 使用 AWS Secrets Manager
- 如何 AWS Elemental MediaConvert 使用 AWS Secrets Manager
- 如何 AWS Elemental MediaLive 使用 AWS Secrets Manager
- 如何 AWS Elemental MediaPackage 使用 AWS Secrets Manager
- 如何 AWS Elemental MediaTailor 使用 AWS Secrets Manager
- Amazon EMR 如何使用 Secrets Manager
- 亚马逊如何 EventBridge 使用 AWS Secrets Manager

- 亚马逊如何 FSx 使用 AWS Secrets Manager 机密
- 如何 AWS Glue DataBrew 使用 AWS Secrets Manager
- AWS Glue Studio 如何使用 AWS Secrets Manager
- 如何 AWS IoT SiteWise 使用 AWS Secrets Manager
- 亚马逊 Kendra 是如何使用的 AWS Secrets Manager
- Amazon Kinesis Video Streams 如何使用 AWS Secrets Manager
- 如何 AWS Launch Wizard 使用 AWS Secrets Manager
- Amazon Lookout for Metrics 如何使用 AWS Secrets Manager
- Amazon Managed Grafana 如何使用 AWS Secrets Manager
- 如何 AWS Managed Services 使用 AWS Secrets Manager
- Amazon Managed Streaming for Apache Kafka 如何使用 AWS Secrets Manager
- Apache Airflow 的亚马逊托管工作流程是如何使用的 AWS Secrets Manager
- AWS Marketplace
- 如何 AWS Migration Hub 使用 AWS Secrets Manager
- 如何 AWS Panorama 使用 Secrets Manager
- AWS 并行计算服务如何使用 AWS Secrets Manager
- 如何 AWS ParallelCluster 使用 AWS Secrets Manager
- Amazon Q 如何使用 Secrets Manager
- 如何Amazon OpenSearchIngestion使用 Secrets Manager
- 如何 AWS OpsWorks for Chef Automate 使用 AWS Secrets Manager
- 亚马逊如何 QuickSight 使用 AWS Secrets Manager
- Amazon RDS 的使用方式 AWS Secrets Manager
- 亚马逊 Redshift 的使用方式 AWS Secrets Manager
- Amazon Redshift 查询编辑器 v2
- 亚马逊 A SageMaker I 是如何使用的 AWS Secrets Manager
- 如何 AWS Schema Conversion Tool 使用 AWS Secrets Manager
- Amazon Timestream for InfluxDB 如何使用 AWS Secrets Manager
- 如何 AWS Toolkit for JetBrains 使用 AWS Secrets Manager
- 如何 AWS Transfer Family 使用 AWS Secrets Manager 秘密

- 如何 AWS Wickr使用 AWS Secrets Manager 秘密

如何 AWS App Runner 使用 AWS Secrets Manager

AWS App Runner 是一项 AWS 服务，它提供了一种快速、简单且经济实惠的方式，可将源代码或容器映像直接部署到 AWS 云中可扩展且安全的 Web 应用程序。您无需学习新技术、决定使用哪种计算服务，也不需要知道如何预置和配置 AWS 资源。

使用 App Runner，您可以在创建服务或更新服务的配置时将密钥和配置引用为服务中的环境变量。有关更多信息，请参阅《AWS App Runner 开发人员指南》中的引用环境变量和管理环境变量。

AWS App2Container 如何使用 AWS Secrets Manager

AWS App2Container 是一款命令行工具，可帮助您提升和转移在本地数据中心或虚拟机上运行的应用程序，使其在由 Amazon ECS、Amazon EKS 或管理的容器中运行 AWS App Runner。

App2Container 使用 Secrets Manager 来管理用于将工件计算机连接到应用程序服务器的凭证，以便运行远程命令。有关更多信息，请参阅 App2Container 用户指南中的管理 AWS App2Contain AWS er 的密钥。

如何 AWS AppConfig 使用 AWS Secrets Manager

AWS AppConfig 是一 AWS Systems Manager 项可用于创建、管理和快速部署应用程序配置的功能。配置可包含存储在 Secrets Manager 中的凭证数据或其他敏感信息。当您创建自由格式配置文件时，可以选择 Secrets Manager 作为配置数据的来源。有关更多信息，请参阅《AWS AppConfig 用户指南》中的创建自由格式配置文件。有关如何 AWS AppConfig 处理开启自动轮换功能的密钥的信息，请参阅《AWS AppConfig 用户指南》中的 S ecrets Manager 密钥轮换。

亚马逊如何 AppFlow 使用 AWS Secrets Manager

亚马逊 AppFlow 是一项完全托管的集成服务，使您能够在软件即服务（SaaS）应用程序（例如 Salesforce）与亚马逊简单存储服务（Amazon S3）和 AWS 服务 Amazon Redshift等应用程序之间安全地交换数据。

在 Amazon 中 AppFlow，当您将一个 SaaS 应用程序配置为源或目标时，您就创建了一个连接。这包括连接到 SaaS 应用程序所需的信息，例如身份验证令牌、用户名和密码。亚马逊将您的连接数据 AppFlow 存储在带前缀的 Secrets Manager 托管密钥中`appflow`。存储密钥的费用已包含在亚马逊的费用中 AppFlow。有关更多信息，请参阅亚马逊 AppFlow 用户指南 AppFlow中的亚马逊数据保护。

如何 AWS AppSync 使用 AWS Secrets Manager

AWS AppSync 为应用程序开发人员提供了一个强大、可扩展的 GraphQL 接口，用于合并来自多个来源的数据，包括 Amazon DynamoD AWS Lambda B 和 HTTP。 APIs

AWS AppSync 使用 Secrets Manager 密钥中的凭证连接亚马逊 RDS 和 Aurora。有关更多信息，请参阅《AWS AppSync 开发人员指南》中的教程：Aurora Serverless。

Amazon Athena 如何使用 AWS Secrets Manager

Amazon Athena 是一种交互式查询服务，让您能够轻松使用标准 SQL 直接分析 Amazon Simple Storage Service（Amazon S3）中的数据。

Amazon Athena 数据源连接器可以将 Athena 联合查询功能与 Secrets Manager 密钥结合使用，从而查询数据。有关更多信息，请参阅《Amazon Athena 用户指南》中的使用 Amazon Athena 联合查询。

亚马逊 Aurora 是如何使用的 AWS Secrets Manager

Amazon Aurora 是一个与 MySQL 和 PostgreSQL 兼容的完全托管式的关系数据库引擎。

要管理 Aurora 的主用户凭证，Aurora 可以为您创建托管密钥。您需要为此密钥支付费用。Aurora 还管理这些凭证的轮换。有关更多信息，请参阅《Amazon Aurora 用户指南》中的使用 Amazon Aurora 和 AWS Secrets Manager管理密码。

有关其他 Aurora 凭证的信息，请参阅 创建密钥。

调用 Amazon RDS Data API 时，您可以使用 Secrets Manager 中的密钥传递数据库的凭证。有关更多信息，请参阅《Amazon Aurora 用户指南》中的使用 Aurora Serverless 数据 API。

使用 Amazon RDS 查询编辑器连接到数据库时，可以将数据库的凭证存储在 Secrets Manager 中。有关更多信息，请参阅 Amazon RDS 用户指南中的使用查询编辑器。

如何 AWS CodeBuild 使用 AWS Secrets Manager

AWS CodeBuild 是云端完全托管的生成服务。 CodeBuild 编译您的源代码，运行单元测试，并生成准备部署的工件。

您可以使用 Secrets Manager 存储私有注册表凭证。有关更多信息，请参阅《AWS CodeBuild 用户指南》 CodeBuild中的私有注册表及其 AWS Secrets Manager 示例。

Amazon Data Firehose 如何使用 AWS Secrets Manager

您可以使用 Amazon Data Firehose 将实时流媒体数据传输到各种流式传输目标。当目标需要凭证或密钥时，Firehose 会在运行时从 Secrets Manager 检索密钥以连接到目标。有关更多信息，请参阅《Amazon Data Firehose 开发人员指南》中的 Authenticate with AWS Secrets Manager in Amazon Data Firehose。

如何 AWS DataSync 使用 AWS Secrets Manager

AWS DataSync 是一项在线数据传输服务，可简化、自动化和加速存储系统和服务之间的数据移动。

支持的某些存储系统 DataSync 需要凭据才能读取和写入数据。 DataSync 使用 Secrets Manager 存储或访问存储凭证。您可以配置 DataSync 为代表您创建密钥，也可以提供自定义密钥。服务管理的密钥以前缀开头`aws-datasync`。您只需为使用在外部创建的密钥而付费 DataSync。请参阅《AWS DataSync 用户指南》中的 “提供存储位置凭证”。

亚马逊如何 DataZone 使用 AWS Secrets Manager

Amazon DataZone 是一项数据管理服务，可让您对数据进行分类、发现、管理、共享和分析。您可以使用来自使用任务抓取的 Amazon Redshift 集群中的表和视图中的数据资产。 AWS Glue 爬网程序要连接亚马逊 Redshift，您需要在 Secrets Manager 密钥中提供亚马逊 DataZone凭证。有关更多信息，请参阅亚马逊 DataZone 用户指南中的使用新 AWS Glue 连接为 Amazon Redshift 数据库创建数据源。

如何 AWS Direct Connect 使用 AWS Secrets Manager

AWS Direct Connect 通过标准的以太网光纤电缆将您的内部网络链接到某个 AWS Direct Connect 位置。通过此连接，您可以直接创建面向公共 AWS 服务的虚拟接口。

AWS Direct Connect 将连接关联密钥名称和连接关联密钥对（CKN/CAK 对）存储在带有前缀的托管密钥中。`directconnect`密钥的成本包含在 AWS Direct Connect的费用中。要更新密钥，必须使用 AWS Direct Connect 而不是 Secrets Manager。有关更多信息，请参阅AWS Direct Connect 用户指南中的将 MACsec CKN/CAK 与 LAG 关联。

如何 AWS Directory Service 使用 AWS Secrets Manager

AWS Directory Service 提供了多种将 Microsoft Active Directory (AD) 与其他 AWS 服务一起使用的方法。您可以使用凭证密码将 Amazon EC2 实例加入您的目录。有关更多信息，请参阅《AWS Direct Connect 用户指南》中的以下内容：

- 将 Linux EC2 实例无缝加入你的微软 AD AWS 托管目录
- 将 Linux EC2 实例无缝加入你的 AD Connector 目录
- 将 Linux EC2 实例无缝加入你的 Simple AD 目录

Amazon DocumentDB（with MongoDB compatibility）如何使用 AWS Secrets Manager

亚马逊 DocumentDB（兼容 MongoDB）是一项完全托管的文档数据库服务，支持 MongoDB 工作负载。Amazon DocumentDB 与 Secrets Manager 集成，可以管理集群的主用户密码，从而增强安全性并简化凭据管理。

Amazon DocumentDB 生成密码，将其存储在 Secrets Manager 中，然后管理密钥设置。默认情况下，Amazon DocumentDB 每七天轮换一次密钥，但您可以根据需要修改轮换计划。创建或修改 Amazon DocumentDB 集群时，您可以在 Secrets Manager 中指定它应管理主用户密码。有关更多信息，请参阅亚马逊 DocumentDB 开发者指南中的使用亚马逊 DocumentDB 进行密码管理和 S ec rets Manager。

如何 AWS Elastic Beanstalk 使用 AWS Secrets Manager

借 AWS Elastic Beanstalk助，您可以快速部署和管理 AWS 云端应用程序，而不必了解运行这些应用程序的基础架构。Elastic Beanstalk 可通过生成 Dockerfile 中描述的映像或提取远程 Docker 映像来启动 Docker 环境。为了向托管私有存储库的在线注册表进行身份验证，Elastic Beanstalk 使用 Secrets Manager 密钥。有关更多信息，请参阅《AWS Elastic Beanstalk 开发人员指南》中的 Docker配置。

Amazon 弹性容器注册表的使用方式 AWS Secrets Manager

Amazon Elastic Container Registry (Amazon ECR) AWS 是一项安全、可扩展且可靠的托管容器镜像注册服务。您可以使用 Docker CLI 或首选客户端从存储库推送和提取镜像。对于包含您要在 Amazon ECR 私有注册表中缓存的映像的每个上游注册表，您必须创建缓存提取规则。对于需要身份验证的上游注册表，您必须以 Secrets Manager 密钥存储凭证。您可以在 Amazon ECR 或 Secrets Manager 控

制台中创建 Secrets Manager 密钥。有关更多信息，请参阅《Amazon ECR 用户指南》中的 Creating a pull through cache rule。

Amazon Elastic Container Service

Amazon Elastic Container Service (Amazon ECS) 是一种完全托管式的容器编排服务，可以帮助您轻松部署、管理和扩展容器化应用程序。您可以通过引用 Secrets Manager 密钥将敏感数据注入容器。有关更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》中的以下页面：

- 教程：使用 Secrets Manager 密钥指定敏感数据
- 通过应用程序以编程方式检索密钥
- 通过环境变量检索密钥
- 检索日志记录配置的密钥

Amazon ECS FSx 支持容器的 Windows 文件服务器卷。Amazon ECS 使用存储在 Secrets Manager 密钥中的凭证来加入活动目录并附加 FSx适用于 Windows 文件服务器的文件系统。有关更多信息，请参阅《亚马逊弹性容器服务开发人员指南》中的教程：用 FSx FSx 于 Amazon ECS 上的 Windows 文件服务器文件系统和用于 Windows 文件服务器卷。

您可以通过使用带有注册表凭据的 Secrets Manager 密钥来引用需要身份验证的私有注册表中的容器镜像。 AWS 有关更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》中的任务的私有注册表身份验证。

当你使用 Amazon ECS Service Connect 时，Amazon ECS 使用 Secrets Manager 托管密钥来存储 AWS Private Certificate Authority TLS 证书。存储密钥的成本包含在 Amazon ECS 的费用中。要更新此密钥，您必须使用 Amazon ECS 而非 Secrets Manager。有关更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》中的支持 Service Connect 的 TLS。

亚马逊如何 ElastiCache 使用 AWS Secrets Manager

在中， ElastiCache 您可以使用名为基于角色的访问控制 (RBAC) 的功能来保护集群。您可以将这些凭证存储在 Secrets Manager 中。Secrets Manager 为这种类型的密钥提供轮换模板。有关更多信息，请参阅 Amazon 用户指南中的自动轮换 ElastiCache 用户密码。

如何 AWS Elemental Live 使用 AWS Secrets Manager

AWS Elemental Live 是一项实时视频服务，可让您为广播和流媒体传输创建实时输出。

AWS Elemental Live 使用秘密 ARN 从 Secrets Manager 获取包含加密密钥的密钥。Elemental Live 使用视频 encrypt/decrypt 的加密密钥。有关更多信息，请参阅 Elemental Live 用户指南中的从 AWS Elemental Live 到的交付在运行时 MediaConnect 的工作原理。

如何 AWS Elemental MediaConnect 使用 AWS Secrets Manager

AWS Elemental MediaConnect 是一项服务，使广播公司和其他优质视频提供商可以轻松可靠地将直播视频摄入其中， AWS Cloud 并将其分发到内部或外部的多个目的地。 AWS Cloud

您可以使用静态密钥加密来保护源、输出和授权，并将加密密钥存储在 AWS Secrets Manager中。有关更多信息，请参阅《AWS Elemental MediaConnect 用户指南》中的 Static key encryption in AWS Elemental MediaConnect。

如何 AWS Elemental MediaConvert 使用 AWS Secrets Manager

AWS Elemental MediaConvert 是一种基于文件的视频处理服务，可为拥有任何规模媒体库的内容所有者和发行商提供可扩展的视频处理。 MediaConvert 要使用凯度水印进行编码，你可以使用 Secrets Manager 来存储你的凯度凭证。有关更多信息，请参阅《用户指南》中的 “使用Kantar在 AWS Elemental MediaConvert 输出中添加音频水印”AWS Elemental MediaConvert 。

如何 AWS Elemental MediaLive 使用 AWS Secrets Manager

AWS Elemental MediaLive 是一项实时视频服务，可让您为广播和流媒体传输创建实时输出。如果您的组织使用带有 AWS Elemental MediaLive 或的 AWS Elemental Link 设备 AWS Elemental MediaConnect，则必须部署设备并配置设备。有关更多信息，请参阅《MediaLive 用户指南》中的设置 MediaLive 为可信实体。

如何 AWS Elemental MediaPackage 使用 AWS Secrets Manager

AWS Elemental MediaPackage 是一项在中运行的 just-in-time视频打包和创作 AWS Cloud服务。借 MediaPackage助，您可以向各种播放设备和内容交付网络提供高度安全、可扩展和可靠的视频流（CDNs）。有关更多信息，请参阅《AWS Elemental MediaPackage 用户指南》中的 Secrets Manager access for CDN authorization。

如何 AWS Elemental MediaTailor 使用 AWS Secrets Manager

AWS Elemental MediaTailor 是一项可扩展的广告插入和频道组装服务，运行在 AWS Cloud.

MediaTailor 支持对你的来源位置进行 Secrets Manager 访问令牌身份验证。通过 Secrets Manager 访问令牌身份验证， MediaTailor 使用 Secrets Manager 密钥对发往您的来源的请求进行身份验证。有关更多信息，请参阅《AWS Elemental MediaTailor 用户指南》中的配置 AWS Secrets Manager 访问令牌身份验证。

Amazon EMR 如何使用 Secrets Manager

亚马逊 EMR 是一个平台，可简化大数据框架（例如 Apache Hadoop 和 Apache Spark）的运行，AWS 以处理和分析大量数据。使用这些框架和相关的开源项目（如 Apache Hive 和 Apache Pig）時，您可以处理用于分析的数据和商业智能工作负载。您还可以使用 Amazon EMR 将大量数据转换进出其他 AWS 数据存储和数据库，例如 Amazon S3 和 Amazon DynamoDB。

在亚马逊上运行的亚马逊 EMR 如何 EC2 使用 Secrets Manager

在 Amazon EMR 中创建集群时，您可以使用 Secrets Manager 中的密钥向集群提供应用程序配置数据。有关更多信息，请参阅 Amazon EMR 管理指南 中的在 Secrets Manager 中存储敏感配置数据。

此外，当您创建 EMR Notebook 时，可以使用 Secrets Manager 存储基于 Git 的私有注册表凭证。有关更多信息，请参阅《Amazon EMR 管理指南》中的将基于 Git 的存储库添加到 Amazon EMR。

EMR Serverless 如何使用密 Secrets Manager

EMR Serverless 提供无服务器运行时环境以简化分析应用程序的操作，因此您无需配置、优化、保护或操作集群。

您可以将数据存储在 EMR Serverless 配置中， AWS Secrets Manager 然后在 EMR Serverless 配置中使用该密钥 ID。这样，您就不会以纯文本形式传递敏感的配置数据并将其暴露给外部 APIs。

有关更多信息，请参阅 Amazon EMR Serverless 用户指南中的使用 EMR Serverless 进行数据保护的 Secrets Manager。

亚马逊如何 EventBridge 使用 AWS Secrets Manager

Amazon EventBridge 是一项无服务器事件总线服务，可用于将应用程序与来自各种来源的数据连接起来。

当您创建 Amazon EventBridge API 目标时，会将其连接 EventBridge 存储在带有前缀的 Secrets Manager 托管密钥中`events`。存储此密钥的成本包含在使用 API 目标的费用中。要更新密钥，必须使用 EventBridge 而不是 Secrets Manager。有关更多信息，请参阅 Amazon EventBridge 用户指南中的 API 目的地。

亚马逊如何 FSx 使用 AWS Secrets Manager 机密

亚马逊 FSx 版 Windows 文件服务器提供完全托管的微软 Windows 文件服务器，由完全原生 Windows 文件系统提供支持。在创建或管理文件共享时，您可以传递来自 AWS Secrets Manager 密钥的证书。有关更多信息，请参阅《亚马逊 Windows 文件服务器用户指南》 FSx中的文件共享和将文件共享配置迁移到亚马逊 FSx 。

如何 AWS Glue DataBrew 使用 AWS Secrets Manager

AWS Glue DataBrew 是一款可视化数据准备工具，无需编写任何代码即可使用它来清理和标准化数据。在中 DataBrew，一组数据转换步骤称为配方。 AWS Glue DataBrew 提供了`DETERMINISTIC_DECRYPTDETERMINISTIC_ENCRYPT`、和`CRYPTOGRAPHIC_HASH`配方步骤，用于对数据集中的个人身份信息 (PII) 执行转换，这些信息使用存储在 Secrets Manager 密钥中的加密密钥。如果您使用 DataBrew 默认密钥来存储加密密钥，则 DataBrew 会创建一个带有前缀的托管密钥`databrew`。存储密钥的费用包含在使用费用中 DataBrew。如果您创建新密钥来存储加密密钥，则 DataBrew 会创建一个带有前缀的密钥`AwsGlueDataBrew`。您需要为此密钥支付费用。

AWS Glue Studio 如何使用 AWS Secrets Manager

AWS Glue Studio 是一个图形界面，可以轻松地在中创建、运行和监控提取、转换和加载 (ETL) 作业。 AWS Glue通过在中配置 Elasticsearch Spark Connector，您可以将亚马逊 OpenSearch 服务用作提取、转换和加载 (ETL) 任务的数据存储。 AWS Glue Studio要连接到 OpenSearch 集群，你可以在 Secrets Manager 中使用密钥。有关更多信息，请参阅《AWS Glue 开发者指南》中的教程：使用适用于 Elasticsearch 的 AWS Glue 连接器。

如何 AWS IoT SiteWise 使用 AWS Secrets Manager

AWS IoT SiteWise 是一项托管服务，可让您大规模收集、建模、分析和可视化来自工业设备的数据。您可以使用 AWS IoT SiteWise 控制台创建网关。然后添加连接到网关的数据源、本地服务器或工业设备。如果您的源要求身份验证，请使用一个密钥来进行身份验证。有关更多信息，请参阅《AWS IoT SiteWise 用户指南》中的配置数据来源身份验证。

亚马逊 Kendra 是如何使用的 AWS Secrets Manager

Amazon Kendra 是一项高度准确且智能的搜索服务，可让用户使用自然语言处理和高级搜索算法搜索非结构化和结构化数据。

通过指定包含数据库凭证的密钥，您可以为存储在数据库中的文档建立索引。有关更多信息，请参阅 Amazon Kendra 用户指南中的使用数据库数据源。

Amazon Kinesis Video Streams 如何使用 AWS Secrets Manager

您可以使用 Amazon Kinesis Video Streams 连接到客户场所的 IP 摄像机，在本地录制和存储来自摄像机的视频，并将视频流式传输到云端进行长期存储、回放和分析处理。要录制和上传来自 IP 摄像机的媒体，请将 Kinesis Video Streams Edge Agent 部署到 AWS IoT Greengrass。您可以将访问流式传输到摄像机的媒体文件所需的凭证存储在 Secrets Manager 密钥中。有关更多信息，请参阅《Amazon Kinesis Video Streams 开发人员指南》中的将 Amazon Kinesis Video Streams Edge Agent 部署到 AWS IoT Greengrass。

如何 AWS Launch Wizard 使用 AWS Secrets Manager

AWS Launch Wizard for Active Directory 是一项应用 AWS Cloud 应用程序最佳实践的服务，可指导您在本地或内部设置新的 Active Directory 基础架构， AWS Cloud 或者向现有基础架构中添加域控制器。

AWS Launch Wizard 需要将域管理员凭据添加到 Secrets Manager 才能将您的域控制器加入活动目录。有关更多信息，请参阅《AWS Launch Wizard 用户指南》中的为 Active Directory AWS Launch Wizard 进行设置。

Amazon Lookout for Metrics 如何使用 AWS Secrets Manager

Amazon Lookout for Metrics 是一项可查找数据中的异常情况，确定其根本原因，并使您能够快速采取措施的服务。您可以将 Amazon Redshift 或 Amazon RDS 作为 Lookout for Metrics 检测器的数据源。要配置数据源，您可以使用包含数据库密码的密钥。有关更多信息，请参阅《Amazon Lookout for Metrics 开发人员指南》中的将 Amazon RDS 与 Lookout for Metrics 结合使用和将 Amazon Redshift 与 Lookout for Metrics 结合使用。

Amazon Managed Grafana 如何使用 AWS Secrets Manager

Amazon Managed Grafana 是一种安全的完全托管式数据可视化服务，您可以使用该服务即时查询、关联和可视化来自多个来源的运行指标、日志和跟踪。当您将 Amazon Redshift 作为数据来源时，您可以通过使用 AWS Secrets Manager 密钥提供 Amazon Redshift 凭证。有关更多信息，请参阅《Amazon Managed Grafana 用户指南》中的 配置 Amazon Redshift。

如何 AWS Managed Services 使用 AWS Secrets Manager

AWS Managed Services 是一项企业服务，可为您的 AWS 基础架构提供持续管理。AMS 自助服务预置（SSP）模式提供对 AMS 托管账户中本机 AWS 服务 服务和 API 功能的完全访问权限。有关如何在 AMS 中请求 Secrets Manager 访问权限的信息，请参阅《AMS 高级用户指南》中的AWS Secrets Manager（AMS 自助服务预置）。

Amazon Managed Streaming for Apache Kafka 如何使用 AWS Secrets Manager

Amazon Managed Streaming for Apache Kafka（Amazon MSK）是一项完全托管式服务，让您能够构建并运行使用 Apache Kafka 来处理串流数据的应用程序。您可以使用 AWS Secrets Manager来存储和保护用户名和密码，从而控制使用这些用户名和密码访问 Amazon MSK 集群的权限。有关更多信息，请参阅 Amazon Managed Streaming for Apache Kafka 开发人员指南中的使用 AWS Secrets Manager进行用户名和密码身份验证。

Apache Airflow 的亚马逊托管工作流程是如何使用的 AWS Secrets Manager

适用于 Apache Airflow 的亚马逊托管工作流程是 Apache Airflow 的托管编排服务，它可以更轻松地在云中大规模设置和操作 end-to-end数据管道。

您可以使用 Secrets Manager 密钥配置 Apache Airflow 连接。有关更多信息，请参阅《适用于 Apache 气流的亚马逊托管工作流程用户指南》中的 AWS Secrets Manager “使用 Secrets Manager 密钥配置 Apache 气流连接” 和 “使用密钥获取 Ap ache 气流变量”。

AWS Marketplace

当您使用 AWS Marketplace Quick Launch 时， AWS Marketplace 会将您的软件与许可证密钥一起分发。 AWS Marketplace 将许可证密钥作为 Secrets Manager 托管密钥存储在你的账户中。存储密钥的费用包含在的费用中 AWS Marketplace。要更新此密钥，您必须使用 AWS Marketplace 而非 Secrets Manager。有关更多信息，请参阅《AWS Marketplace 卖家指南》中的配置 Quick Launch。

如何 AWS Migration Hub 使用 AWS Secrets Manager

AWS Migration Hub 提供单一位置来跟踪跨多个 AWS 工具和合作伙伴解决方案的迁移任务。

AWS Migration Hub Orchestrator 可简化并自动将服务器和企业应用程序迁移到。 AWS Migration Hub Orchestrator 使用密钥来获取源服务器的连接信息。有关更多信息，请参阅《AWS Migration Hub 编排工具用户指南》中的以下内容：

- 将 SAP NetWeaver 应用程序迁移到 AWS
- 在 Amazon 上重新托管应用程序 EC2

Migration Hub Strategy Recommendations 为可行的应用程序转型路径提供了迁移和现代化策略建议。Strategy Recommendations 可以使用密钥来获取连接信息，从而分析 SQL Server 数据库。有关更多信息，请参阅 Strategy Recommendations 数据库分析。

如何 AWS Panorama 使用 Secrets Manager

AWS Panorama 是一项将计算机视觉引入您的本地摄像机网络的服务。您 AWS Panorama 用于注册设备、更新其软件以及向其部署应用程序。当您将视频流注册为应用程序的数据源时，如果该视频流受密码保护，则会将其凭据 AWS Panorama 存储在 Secrets Manager 密钥中。有关更多信息，请参阅《AWS Panorama 开发人员指南》中的管理 AWS Panorama 中的摄像机视频流。

AWS 并行计算服务如何使用 AWS Secrets Manager

AWS 并行计算服务 (AWS PCS) 是一项托管服务，可以更轻松地运行和扩展高性能计算 (HPC) 和分布式机器学习工作负 AWS载。

要连接到集群作业调度程序， AWS PCS 会创建一个带有前缀的托管密钥pcs来存储调度程序密钥。存储密钥的费用包含在 AWS PCS 费用中。 AWS 当您删除 PCS 集群时， AWS PCS 会自动删除该密钥。有关更多信息，请参阅 PCS 用户指南中的在 AWS PCS 中AWS 使用集群密钥。

> **Important**
>
> 请勿修改或删除 AWS PCS 集群密钥。

如何 AWS ParallelCluster 使用 AWS Secrets Manager

AWS ParallelCluster 是一款开源集群管理工具，可用于在中部署和管理高性能计算 (HPC) 集群。AWS Cloud你可以创建一个多用户环境，其中包括与 AWS 托管 Microsoft AD（活动目录）集成的环

境。 AWS ParallelCluster AWS ParallelCluster 使用 Secrets Manager 密钥来验证 Active Directory 的登录信息。有关更多信息，请参阅《AWS ParallelCluster 用户指南》中的集成 Active Directory。

Amazon Q 如何使用 Secrets Manager

要验证 Amazon Q 是否能够访问您的数据来源，您需要使用 Secrets Manager 密钥向 Amazon Q 提供数据来源访问凭证。如果使用控制台，您可以选择创建新密钥，也可以使用现有密钥。有关更多信息，请参阅 Amazon Q 开发者指南中的概念-身份验证。

如何Amazon OpenSearchIngestion使用 Secrets Manager

Amazon OpenSearch Ingestion是一款完全托管的无服务器数据收集器，可将实时日志、指标和跟踪数据流式传输到 Amazon Serv OpenSearch ice 域和 OpenSearch Serverless集合。您可以使用带有 Secrets Manager 的 OpenSearchIngestion管道来安全地管理您的凭证。有关更多信息，请参阅：

- 将 OpenSearch Ingestion管道与 Atlassian Services
- 在 Amazon DocumentDB 中使用 OpenSearch Ingestion管道
- 将 OpenSearch Ingestion管道与 Confluent Cloud Kafka
- 将 OpenSearch Ingestion管道与 Kafka
- 使用从自管 OpenSearch 集群迁移数据 Amazon OpenSearch Ingestion

如何 AWS OpsWorks for Chef Automate 使用 AWS Secrets Manager

OpsWorks 是一项配置管理服务，可通过使用 OpsWorks Puppet Enterprise 或 AWS OpsWorks for Chef Automate，来帮助您在云企业中配置和操作应用程序。

当您在中创建新服务器时 AWS OpsWorks CM， OpsWorks CM 会将服务器的信息存储在 Secrets Manager 托管密钥中，前缀为前缀`opsworks-cm`。秘密的费用已包含在费用中 OpsWorks。有关更多信息，请参阅 OpsWorks 用户指南中的集成 AWS Secrets Manager。

亚马逊如何 QuickSight 使用 AWS Secrets Manager

Amazon QuickSight 是一项云规模的商业智能 (BI) 服务，可用于分析、数据可视化和报告。您可以在中使用各种数据源 QuickSight。如果您将数据库凭据存储在 Secrets Manager 密钥中，则 QuickSight

可以使用这些密钥连接到数据库。有关更多信息，请参阅《亚马逊 QuickSight 用户指南》中的在亚马逊 QuickSight中使用 AWS Secrets Manager 密码代替数据库凭证。

Amazon RDS 的使用方式 AWS Secrets Manager

Amazon Relational Database Service (Amazon RDS) 是一项 Web 服务，让用户能够在 AWS Cloud云中更轻松地设置、操作和扩展关系数据库。

要管理包括 Aurora 在内的 Amazon Relational Database Service（Amazon RDS）的主用户凭证，Amazon RDS 可以为您创建托管密钥。您需要为此密钥支付费用。Amazon RDS 还管理这些凭证的轮换。有关更多信息，请参阅《Amazon RDS 用户指南》中的使用 Amazon RDS 和 AWS Secrets Manager管理密码。

有关其他 Amazon RDS 凭证，请参阅 创建密钥。

使用 Amazon RDS 查询编辑器连接到数据库时，可以将数据库的凭证存储在 Secrets Manager 中。有关更多信息，请参阅 Amazon RDS 用户指南中的使用查询编辑器。

亚马逊 Redshift 的使用方式 AWS Secrets Manager

Amazon Redshift 是云中一种完全托管的 PB 级数据仓库服务。

要管理 Amazon Redshift 的管理员凭证，Amazon Redshift 可以为您创建托管密钥。您需要为此密钥支付费用。Amazon Redshift 还管理这些凭证的轮换。有关更多信息，请参阅《Amazon Redshift 管理指南》中的使用 AWS Secrets Manager管理 Amazon Redshift 管理员密码。

有关其他 Amazon Redshift 凭证，请参阅 创建密钥。

调用 Amazon Redshift Data API 时，您可以使用 Secrets Manager 中的密钥传递集群的凭证。有关更多信息，请参阅使用 Amazon Redshift Data API。

使用 Amazon Redshift 查询编辑器连接到数据库时，Amazon Redshift 可将您的凭证存储在带有前缀 `redshiftqueryeditor` 的 Secrets Manager 密钥中。您需要为此密钥支付费用。有关更多信息，请参阅《Amazon Redshift 管理指南》中的 使用查询编辑器查询数据库。

有关查询编辑器 v2，请参阅 the section called “Amazon Redshift 查询编辑器 v2”。

Amazon Redshift 查询编辑器 v2

Amazon Redshift 查询编辑器 v2 是一个单独的基于 Web 的 SQL 客户端应用程序，用于在 Amazon Redshift 数据仓库上创作和运行查询。使用 Amazon Redshift 查询编辑器 V2 连接到数据库

时，Amazon Redshift 可将您的凭证存储在带有前缀 `sqlworkbench` 的 Secrets Manager 托管密钥中。存储此密钥的成本包含在使用 Amazon Redshift 的费用中。要更新此密钥，您必须使用 Amazon Redshift 而非 Secrets Manager。有关更多信息，请参阅《Amazon Redshift 管理指南》中的使用查询编辑器 v2。

有关之前的查询编辑器，请参阅 the section called “Amazon Redshift”。

亚马逊 A SageMaker I 是如何使用的 AWS Secrets Manager

SageMaker AI 是一项完全托管的机器学习服务。借助 SageMaker 人工智能，数据科学家和开发人员可以快速轻松地构建和训练机器学习模型，然后将其直接部署到生产就绪的托管环境中。它提供了一个集成的 Jupyter 编写 Notebook 实例，供您轻松访问数据源以便进行探索和分析，因此您无需管理服务器。

您可以将 Git 存储库关联到 Jupyter notebook 实例，以将笔记本保存到即使您停止或删除笔记本电脑实例仍可持久保存的源代码控制环境中。您可以使用 Secrets Manager 管理私有存储库凭证。有关更多信息，请参阅《亚马逊 A SageMaker I 开发者指南》中的 “将 Git 存储库与亚马逊 SageMaker 笔记本实例关联”。

要从 Databricks 导入数据，Data Wrangler 会将您的 JDBC URL 存储在 Secrets Manager 中。有关更多信息，请参阅从 Databricks（JDBC）导入数据。

要从 Snowflake 导入数据，Data Wrangler 会将您的凭证存储在某个 Secrets Manager 密钥中。有关更多信息，请参阅从 Snowflake 导入数据。

如何 AWS Schema Conversion Tool 使用 AWS Secrets Manager

您可以使用 AWS Schema Conversion Tool (AWS SCT) 将现有数据库架构从一个数据库引擎转换为另一个数据库引擎。您可以转换关系 OLTP 架构或数据仓库架构。转换后的 Schema 适用于 Amazon Relational Database Service（Amazon RDS）MySQL、MariaDB、Oracle、SQL Server、PostgreSQL 数据库、Amazon Aurora 数据库集群或Amazon Redshift 集群。转换后的架构也可用于 Amazon Elastic Compute Cloud 实例上的数据库，或作为数据存储在 S3 存储桶中。

转换数据库架构时， AWS SCT 可以使用存储在中的数据库凭据 AWS Secrets Manager。有关更多信息，请参阅《用户指南》 AWS Secrets Manager 中的在 AWS SCT 用户界面中AWS Schema Conversion Tool 使用。

Amazon Timestream for InfluxDB 如何使用 AWS Secrets Manager

InfluxDB 的 Timestream 是一个托管的时间序列数据库引擎，可让您轻松地使用开源为实时时间序列应用程序运行 InfluxDB 数据库。 AWS APIs借助 Timestream for InfluxDB，您可以设置、操作和扩展时间序列工作负载，这些工作负载可以以个位数毫秒的查询响应时间来应答查询。

当您创建 Timestream for InfluxDB 数据库时，Timestream 会自动创建一个用于存储管理员凭证的密钥。有关更多信息，请参阅《Timestream 开发人员指南》中的 How Timestream for InfluxDB uses secrets。

如何 AWS Toolkit for JetBrains 使用 AWS Secrets Manager

AWS Toolkit for JetBrains 是用于集成开发环境 (IDEs) 的开源插件 JetBrains。此工具包使开发人员能够更轻松地开发、调试和部署使用 AWS的无服务器应用程序。使用此工具包连接到 Amazon Redshift 集群时，您可以使用 Secrets Manager 密钥进行身份验证。有关更多信息，请参阅《AWS Toolkit for JetBrains 用户指南》中的访问 Amazon Redshift 集群。

如何 AWS Transfer Family 使用 AWS Secrets Manager 秘密

AWS Transfer Family 是一种安全的传输服务，使您能够将文件传入和传出 AWS 存储服务。

Transfer Family 现在支持对使用适用性声明 2 (AS2) 协议的服务器使用基本身份验证。您可以创建新的 Secrets Manager 密钥，也可以选择现有密钥作为凭证。有关更多信息，请参阅《AWS Transfer Family 用户指南》中的 AS2 连接器基本身份验证。

要对 Transfer Family 用户进行身份验证，您可以 AWS Secrets Manager 将其用作身份提供者。有关更多信息，请参阅《AWS Transfer Family 用户指南》中的使用自定义身份提供商和博客文章启用密码身份验证以供 AWS Transfer Family 使用 AWS Secrets Manager。

您可以对 Transfer Family 通过工作流程处理的文件使用 Pretty Good Privacy (PGP) 解密。要在工作流程步骤中使用解密，您需要提供在 Secrets Manager 中管理的 PGP 密钥。有关更多信息，请参阅《AWS Transfer Family 用户指南》中的生成和管理 PGP 密钥。

如何 AWS Wickr使用 AWS Secrets Manager 秘密

AWS Wickr 是一项 end-to-end加密服务，可帮助组织和政府机构通过群组消息、语音 one-to-one和视频通话、文件共享、屏幕共享等进行安全通信。您可以使用 Wickr 数据留存机器人实现工作流的自动

化。如果机器人可以访问 AWS 服务，那么你应该创建一个 Secrets Manager 密钥来存储机器人凭证。有关更多信息，请参阅《AWS Wickr 管理指南》中的启动数据保留机器人。

在中创建 AWS Secrets Manager 密钥 AWS CloudFormation

您可以使用 CloudFormation 模板中的 `AWS::SecretsManager::Secret`资源在 CloudFormation 堆栈中创建密钥，如所示创建密钥。

要为 Amazon RDS 或 Aurora 创建管理员密钥，建议您使用 `AWS::RDS::DBCluster` 中的 `ManageMasterUserPassword`。然后，Amazon RDS 为您创建密钥并管理轮换。有关更多信息，请参阅 托管轮换。

对于 Amazon Redshift 和 Amazon DocumentDB 凭证，请首先使用 Secret Manager 生成的密码创建密钥，然后使用动态引用从该密钥中检索用户名和密码，以用作新数据库的凭证。接下来，使用 `AWS::SecretsManager::SecretTargetAttachment` 资源将有关数据库的详细信息添加到 Secrets Manager 需要轮换密钥的密钥。最后，要启用自动轮换，请使用 `AWS::SecretsManager::RotationSchedule` 资源并提供轮换函数和计划。请参阅以下示例：

- 使用 Amazon Redshift 凭证创建密钥
- 使用 Amazon DocumentDB 凭证创建密钥

要将资源策略附加到您的秘密，请使用 `AWS::SecretsManager::ResourcePolicy` 资源。

有关使用创建资源的信息 AWS CloudFormation，请参阅《 AWS CloudFormation 用户指南》中的 “学习模板基础知识”。您也可以使用 AWS Cloud Development Kit (AWS CDK)。有关更多信息，请参阅 AWS Secrets Manager 构建库。

使用创建 AWS Secrets Manager 密钥 AWS CloudFormation

此示例将创建一个名为 **`CloudFormationCreatedSecret-`*`a1b2c3d4e5f6`*** 的密钥。密码值是下面的 JSON，其中包含一个 32 个字符的密码，该密码是在创建密钥时生成的。

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

此示例使用以下 CloudFormation 资源：

- `AWS::SecretsManager::Secret`

有关使用创建资源的信息 AWS CloudFormation，请参阅《 AWS CloudFormation 用户指南》中的 “学习模板基础知识”。

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
```

使用自动轮换功能创建 AWS Secrets Manager 密钥和一个 Amazon RDS MySQL 数据库实例 AWS CloudFormation

要为 Amazon RDS 或 Aurora 创建管理员密钥，建议您使用 `ManageMasterUserPassword`，如 `AWS::RDS::DBCluster` 中的示例为主密码创建 Secrets Manager 密钥所示。然后，Amazon RDS 为您创建密钥并管理轮换。有关更多信息，请参阅 托管轮换。

使用创建 AWS Secrets Manager 密钥和亚马逊 Redshift 集群 AWS CloudFormation

要为 Amazon Redshift 创建管理员密钥，我们建议您使用 `AWS::Redshift::Cluster` 和 `AWS::RedshiftServerless::Namespace` 上的示例。

使用创建 AWS Secrets Manager 密钥和亚马逊文档数据库实例 AWS CloudFormation

此示例将创建一个秘密，并使用该秘密中的凭证作为用户和密码，创建一个 Amazon DocumentDB 实例。该密钥已附加用于指定谁可以访问密钥的基于资源的策略。该模板还可以从 轮换函数模板 创建 Lambda 轮换函数，并将秘密配置为协调世界时每月第一天上午 8:00 到 10:00 之间自动轮换。作为安全最佳实践，该实例位于 Amazon VPC 中。

此示例使用了 Secrets Manager 的以下 CloudFormation 资源：

- `AWS::SecretsManager::Secret`
- `AWS::SecretsManager::SecretTargetAttachment`
- `AWS::SecretsManager::RotationSchedule`

有关使用创建资源的信息 AWS CloudFormation，请参阅《 AWS CloudFormation 用户指南》中的 “学习模板基础知识”。

JSON

```
{
   "AWSTemplateFormatVersion":"2010-09-09",
   "Transform":"AWS::SecretsManager-2020-07-23",
   "Resources":{
      "TestVPC":{
         "Type":"AWS::EC2::VPC",
         "Properties":{
            "CidrBlock":"10.0.0.0/16",
            "EnableDnsHostnames":true,
            "EnableDnsSupport":true
         }
      },
      "TestSubnet01":{
```

```
            "Type":"AWS::EC2::Subnet",
            "Properties":{
               "CidrBlock":"10.0.96.0/19",
               "AvailabilityZone":{
                  "Fn::Select":[
                     "0",
                     {
                        "Fn::GetAZs":{
                           "Ref":"AWS::Region"
                        }
                     }
                  ]
               },
               "VpcId":{
                  "Ref":"TestVPC"
               }
            }
         },
         "TestSubnet02":{
            "Type":"AWS::EC2::Subnet",
            "Properties":{
               "CidrBlock":"10.0.128.0/19",
               "AvailabilityZone":{
                  "Fn::Select":[
                     "1",
                     {
                        "Fn::GetAZs":{
                           "Ref":"AWS::Region"
                        }
                     }
                  ]
               },
               "VpcId":{
                  "Ref":"TestVPC"
               }
            }
         },
         "SecretsManagerVPCEndpoint":{
            "Type":"AWS::EC2::VPCEndpoint",
            "Properties":{
               "SubnetIds":[
                  {
                     "Ref":"TestSubnet01"
                  },
```

```
                {
                   "Ref":"TestSubnet02"
                }
             ],
             "SecurityGroupIds":[
                {
                   "Fn::GetAtt":[
                      "TestVPC",
                      "DefaultSecurityGroup"
                   ]
                }
             ],
             "VpcEndpointType":"Interface",
             "ServiceName":{
                "Fn::Sub":"com.amazonaws.${AWS::Region}.secretsmanager"
             },
             "PrivateDnsEnabled":true,
             "VpcId":{
                "Ref":"TestVPC"
             }
          }
       },
       "MyDocDBClusterRotationSecret":{
          "Type":"AWS::SecretsManager::Secret",
          "Properties":{
             "GenerateSecretString":{
                "SecretStringTemplate":"{\"username\": \"someadmin\",\"ssl\": true}",
                "GenerateStringKey":"password",
                "PasswordLength":16,
                "ExcludeCharacters":"\"@/\\"
             },
             "Tags":[
                {
                   "Key":"AppName",
                   "Value":"MyApp"
                }
             ]
          }
       },
       "MyDocDBCluster":{
          "Type":"AWS::DocDB::DBCluster",
          "Properties":{
             "DBSubnetGroupName":{
                "Ref":"MyDBSubnetGroup"
```

```
            },
            "MasterUsername":{
               "Fn::Sub":"{{resolve:secretsmanager:
${MyDocDBClusterRotationSecret}::username}}"
            },
            "MasterUserPassword":{
               "Fn::Sub":"{{resolve:secretsmanager:
${MyDocDBClusterRotationSecret}::password}}"
            },
            "VpcSecurityGroupIds":[
               {
                  "Fn::GetAtt":[
                     "TestVPC",
                     "DefaultSecurityGroup"
                  ]
               }
            ]
         }
      },
      "DocDBInstance":{
         "Type":"AWS::DocDB::DBInstance",
         "Properties":{
            "DBClusterIdentifier":{
               "Ref":"MyDocDBCluster"
            },
            "DBInstanceClass":"db.r5.large"
         }
      },
      "MyDBSubnetGroup":{
         "Type":"AWS::DocDB::DBSubnetGroup",
         "Properties":{
            "DBSubnetGroupDescription":"",
            "SubnetIds":[
               {
                  "Ref":"TestSubnet01"
               },
               {
                  "Ref":"TestSubnet02"
               }
            ]
         }
      },
      "SecretDocDBClusterAttachment":{
         "Type":"AWS::SecretsManager::SecretTargetAttachment",
```

```
         "Properties":{
            "SecretId":{
               "Ref":"MyDocDBClusterRotationSecret"
            },
            "TargetId":{
               "Ref":"MyDocDBCluster"
            },
            "TargetType":"AWS::DocDB::DBCluster"
         }
      },
      "MySecretRotationSchedule":{
         "Type":"AWS::SecretsManager::RotationSchedule",
         "DependsOn":"SecretDocDBClusterAttachment",
         "Properties":{
            "SecretId":{
               "Ref":"MyDocDBClusterRotationSecret"
            },
            "HostedRotationLambda":{
               "RotationType":"MongoDBSingleUser",
               "RotationLambdaName":"MongoDBSingleUser",
               "VpcSecurityGroupIds":{
                  "Fn::GetAtt":[
                     "TestVPC",
                     "DefaultSecurityGroup"
                  ]
               },
               "VpcSubnetIds":{
                  "Fn::Join":[
                     ",",
                     [
                        {
                           "Ref":"TestSubnet01"
                        },
                        {
                           "Ref":"TestSubnet02"
                        }
                     ]
                  ]
               }
            },
            "RotationRules":{
              "Duration": "2h",
              "ScheduleExpression": "cron(0 8 1 * ? *)"
            }
```

```
        }
      }
    }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::SecretsManager-2020-07-23
Resources:
  TestVPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsHostnames: true
      EnableDnsSupport: true
  TestSubnet01:
    Type: AWS::EC2::Subnet
    Properties:
      CidrBlock: 10.0.96.0/19
      AvailabilityZone: !Select
        - '0'
        - !GetAZs
          Ref: AWS::Region
      VpcId: !Ref TestVPC
  TestSubnet02:
    Type: AWS::EC2::Subnet
    Properties:
      CidrBlock: 10.0.128.0/19
      AvailabilityZone: !Select
        - '1'
        - !GetAZs
          Ref: AWS::Region
      VpcId: !Ref TestVPC
  SecretsManagerVPCEndpoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      SubnetIds:
        - !Ref TestSubnet01
        - !Ref TestSubnet02
      SecurityGroupIds:
        - !GetAtt TestVPC.DefaultSecurityGroup
      VpcEndpointType: Interface
```

```
      ServiceName: !Sub com.amazonaws.${AWS::Region}.secretsmanager
      PrivateDnsEnabled: true
      VpcId: !Ref TestVPC
  MyDocDBClusterRotationSecret:
    Type: AWS::SecretsManager::Secret
    Properties:
      GenerateSecretString:
        SecretStringTemplate: '{"username": "someadmin","ssl": true}'
        GenerateStringKey: password
        PasswordLength: 16
        ExcludeCharacters: '"@/\'
      Tags:
        - Key: AppName
          Value: MyApp
  MyDocDBCluster:
    Type: AWS::DocDB::DBCluster
    Properties:
      DBSubnetGroupName: !Ref MyDBSubnetGroup
      MasterUsername: !Sub '{{resolve:secretsmanager:
${MyDocDBClusterRotationSecret}::username}}'
      MasterUserPassword: !Sub '{{resolve:secretsmanager:
${MyDocDBClusterRotationSecret}::password}}'
      VpcSecurityGroupIds:
        - !GetAtt TestVPC.DefaultSecurityGroup
  DocDBInstance:
    Type: AWS::DocDB::DBInstance
    Properties:
      DBClusterIdentifier: !Ref MyDocDBCluster
      DBInstanceClass: db.r5.large
  MyDBSubnetGroup:
    Type: AWS::DocDB::DBSubnetGroup
    Properties:
      DBSubnetGroupDescription: ''
      SubnetIds:
        - !Ref TestSubnet01
        - !Ref TestSubnet02
  SecretDocDBClusterAttachment:
    Type: AWS::SecretsManager::SecretTargetAttachment
    Properties:
      SecretId: !Ref MyDocDBClusterRotationSecret
      TargetId: !Ref MyDocDBCluster
      TargetType: AWS::DocDB::DBCluster
  MySecretRotationSchedule:
    Type: AWS::SecretsManager::RotationSchedule
```

```
    DependsOn: SecretDocDBClusterAttachment
    Properties:
      SecretId: !Ref MyDocDBClusterRotationSecret
      HostedRotationLambda:
        RotationType: MongoDBSingleUser
        RotationLambdaName: MongoDBSingleUser
        VpcSecurityGroupIds: !GetAtt TestVPC.DefaultSecurityGroup
        VpcSubnetIds: !Join
          - ','
          - - !Ref TestSubnet01
            - !Ref TestSubnet02
      RotationRules:
        Duration: 2h
        ScheduleExpression: cron(0 8 1 * ? *)
```

Secrets Manager 的使用方式 AWS CloudFormation

当你使用控制台开启轮换时，Secrets Manager 会使用 AWS CloudFormation 创建轮换资源。如果在该过程中创建了新的旋转函数，则会`AWS::Serverless::Function`根据相应的函数 AWS CloudFormation 创建一个轮换函数模板。然后 AWS CloudFormation 设置 `RotationSchedule`，它为密钥设置轮换函数和轮换规则。开启自动旋转后， AWS CloudFormation 您可以通过在横幅中选择 “查看堆栈” 来查看堆栈。

有关启用自动轮换功能的信息，请参阅 轮换 密钥。

在中创建 AWS Secrets Manager 密钥 AWS Cloud Development Kit (AWS CDK)

要在 CDK 应用程序中创建、管理和检索密钥，您可以使用 AWS Secrets Manager 构造库，其中包含了 `ResourcePolicy`、`RotationSchedule`、`Secret`、`SecretRotation` 和 `SecretTargetAttachment` 构造。

在 CDK 应用程序中使用密钥的一个好做法是先使用控制台或 CLI 创建密钥，然后将密钥导入到 CDK 应用程序中。

有关示例，请参阅：

- 创建密钥
- 导入密钥
- 检索密钥
- 授予使用密钥的权限
- 轮换密钥
- 轮换数据库密钥
- 将密钥复制到其他区域

有关 CDK 的更多信息，请参阅 AWS Cloud Development Kit (AWS CDK) v2 开发人员指南。

监控 AWS Secrets Manager 机密

AWS 提供监控工具，用于监视 Secrets Manager 的密钥，在出现问题时进行报告，并在适当时自动采取措施。如果您需要调查任何意外的使用或更改，您可以使用日志，并回滚不需要的更改。您还可以设置自动检查不当使用密钥的机制和任何尝试删除密钥的机制。

主题

- 使用记录 AWS Secrets Manager 事件 AWS CloudTrail
- AWS Secrets Manager 使用 Amazon 进行监控 CloudWatch
- 将赛 AWS Secrets Manager 事与 Amazon 进行匹配 EventBridge
- 监控计划删除的 AWS Secrets Manager 密钥何时被访问
- 使用以下方法监控 AWS Secrets Manager 密钥的合规性 AWS Config
- 监控 Secrets Manager 成本
- 使用 Amazon 检测威胁 GuardDuty

使用记录 AWS Secrets Manager 事件 AWS CloudTrail

AWS CloudTrail 将 Secrets Manager 的所有 API 调用记录为事件，包括来自 Secrets Manager 控制台的调用，以及其他几个用于轮换和删除密钥版本的事件。有关 Secrets Manager 记录中的日志条目列表，请参阅 CloudTrail 条目。

您可以使用 CloudTrail 控制台查看最近 90 天记录的事件。要持续记录您的 AWS 账户中的事件，包括 Secrets Manager 的事件，请创建一个跟踪，以便将日志文件 CloudTrail 传输到 Amazon S3 存储桶。请参阅为您的 AWS 账户创建跟踪。您也可以配置 CloudTrail 为接收来自多个 AWS 账户和的 CloudTrail 日志文件AWS 区域。

您可以配置其他 AWS 服务，以进一步分析和处理 CloudTrail 日志中收集的数据。查看与 CloudTrail 日志的AWS 服务集成。当您向 Amazon S3 存储桶 CloudTrail 发布新的日志文件时，您还可以收到通知。有关信息，请参阅配置 Amazon SNS 通知。 CloudTrail

从 CloudTrail 日志中检索 Secrets Manager 事件（控制台）

1. 打开 CloudTrail 控制台，网址为https://console.aws.amazon.com/cloudtrail/。
2. 确保控制台指向发生事件的区域。控制台仅显示在所选区域中发生的那些事件。从控制台右上角的下拉列表中选择区域。

3. 在左侧导航窗格中，选择 Event history (事件历史记录)。
4. 选择 Filter (筛选) 条件和/或 Time range (时间范围) 以帮助您查找您在寻找的事件。例如：
   a. 要查看所有 Secrets Manager 事件，对于查找属性，请选择事件源。然后，对于 Enter event source (输入事件源)，选择 **`secretsmanager.amazonaws.com`**。
   b. 要查看密钥的所有事件，对于查找属性，请选择资源名称。然后，对于输入资源名称，输入密钥的名称。
5. 要查看更多详细信息，请选择事件旁边的展开箭头。要查看所有可用信息，请选择 View event (查看事件)。

AWS CLI

Example 从 CloudTrail 日志中检索 Secrets Manager 事件

以下 `lookup-events` 示例将查找 Secrets Manager 事件。

```
aws cloudtrail lookup-events \
    --region us-east-1 \
    --lookup-attributes
 AttributeKey=EventSource,AttributeValue=secretsmanager.amazonaws.com
```

AWS CloudTrail Secrets Manager 的参赛作品

AWS Secrets Manager 将所有 Secrets Manager 操作以及其他与轮换和删除相关的事件的条目写入您的 AWS CloudTrail 日志。有关对这些事件执行操作的信息，请参阅 将 Secrets Manager 活动与 EventBridge。

日志条目类型

- Secrets Manager 操作的日志条目
- 有关删除操作的日志条目
- 可用于复制的日志条目
- 有关轮换操作的日志条目

Secrets Manager 操作的日志条目

通过调用 Secrets Manager 操作生成的事件具有 `"detail-type": ["AWS API Call via CloudTrail"]`。

> **Note**
>
> 在 2024 年 2 月之前，一些 Secrets Manager 操作报告的事件中包含 “arN” 而不是”arn“为了秘密的 ARN。有关更多信息，请参阅 AWS re:Post。

以下是您或服务通过 API、SDK 或 CLI 调用 Secrets Manager 操作时生成的 CloudTrail 条目。

BatchGetSecretValue

由BatchGetSecretValue操作生成。有关检索密钥的信息，请参阅 获取密钥。

CancelRotateSecret

由CancelRotateSecret操作生成。有关轮换的更多信息，请参阅 轮换 密钥。

CreateSecret

由CreateSecret操作生成。有关创建密钥的信息，请参阅 管理密钥。

DeleteResourcePolicy

由DeleteResourcePolicy操作生成。有关权限的信息，请参阅 the section called “身份验证和访问控制”。

DeleteSecret

由DeleteSecret操作生成。有关删除密钥的信息，请参阅 the section called “删除密钥”。

DescribeSecret

由DescribeSecret操作生成。

GetRandomPassword

由GetRandomPassword操作生成。

GetResourcePolicy

由GetResourcePolicy操作生成。有关权限的信息，请参阅 the section called “身份验证和访问控制”。

GetSecretValue

由GetSecretValue和BatchGetSecretValue操作生成。有关检索密钥的信息，请参阅 获取密钥。

ListSecrets

由ListSecrets操作生成。有关列出密钥的信息，请参阅 the section called “查找密钥”。

ListSecretVersionIds

由ListSecretVersionIds操作生成。

PutResourcePolicy

由PutResourcePolicy操作生成。有关权限的信息，请参阅 the section called “身份验证和访问控制”。

PutSecretValue

由PutSecretValue操作生成。有关更新密钥的信息，请参阅 the section called “修改密钥”。

RemoveRegionsFromReplication

由RemoveRegionsFromReplication操作生成。有关复制密钥的更多信息，请参阅 多区域复制。

ReplicateSecretToRegions

由ReplicateSecretToRegions操作生成。有关复制密钥的更多信息，请参阅 多区域复制。

RestoreSecret

由RestoreSecret操作生成。有关还原已删除密钥的信息，请参阅 the section called “恢复密钥”。

RotateSecret

由RotateSecret操作生成。有关轮换的更多信息，请参阅 轮换 密钥。

StopReplicationToReplica

由StopReplicationToReplica操作生成。有关复制密钥的更多信息，请参阅 多区域复制。

TagResource

由TagResource操作生成。有关标记密钥的信息，请参阅 the section called “标记 密钥”。

UntagResource

由UntagResource操作生成。有关取消密钥标签的信息，请参阅 the section called “标记 密钥”。

UpdateSecret

由UpdateSecret操作生成。有关更新密钥的信息，请参阅 the section called “修改密钥”。

UpdateSecretVersionStage

由UpdateSecretVersionStage操作生成。有关版本阶段的更多信息，请参阅 the section called “密钥版本”。

ValidateResourcePolicy

由ValidateResourcePolicy操作生成。有关权限的信息，请参阅 the section called “身份验证和访问控制”。

有关删除操作的日志条目

除了生成与 Secrets Manager 操作有关的事件外，Secrets Manager 还会生成以下与删除有关的事件。这些事件具有 `"detail-type": ["AWS Service Event via CloudTrail"]`。

CancelSecretVersionDelete

由 Secrets Manager 服务生成。如果在具有版本的密钥上调用 `DeleteSecret`，然后再调用 `RestoreSecret`，则 Secrets Manager 会为还原的每个密钥版本记录此事件。有关还原已删除密钥的信息，请参阅 the section called “恢复密钥”。

EndSecretVersionDelete

在删除密钥版本时由 Secrets Manager 服务生成。有关更多信息，请参阅 the section called “删除密钥”。

StartSecretVersionDelete

在 Secrets Manager 开始删除密钥版本时由 Secrets Manager 服务生成。有关删除密钥的信息，请参阅 the section called “删除密钥”。

SecretVersionDeletion

在 Secrets Manager 删除已弃用的秘密版本时由 Secrets Manager 服务生成。有关更多信息，请参阅密钥版本。

可用于复制的日志条目

除了生成与 Secrets Manager 操作有关的事件外，Secrets Manager 还会生成以下与复制有关的事件。这些事件具有 `"detail-type": ["AWS Service Event via CloudTrail"]`。

ReplicationFailed

在复制失败时由 Secrets Manager 服务生成。有关复制密钥的更多信息，请参阅 多区域复制。

ReplicationStarted

在 Secrets Manager 开始复制密钥时由 Secrets Manager 服务生成。有关复制密钥的更多信息，请参阅 多区域复制。

ReplicationSucceeded

在成功复制密钥时由 Secrets Manager 服务生成。有关复制密钥的更多信息，请参阅 多区域复制。

有关轮换操作的日志条目

除了生成与 Secrets Manager 操作有关的事件外，Secrets Manager 还会生成以下与轮换有关的事件。这些事件具有 `"detail-type": ["AWS Service Event via CloudTrail"]`。

RotationStarted

在 Secrets Manager 开始轮换密钥时由 Secrets Manager 服务生成。有关轮换的更多信息，请参阅 轮换 密钥。

RotationAbandoned

在 Secrets Manager 放弃轮换尝试并从密钥的某个现有版本删除 `AWSPENDING` 标签时由 Secrets Manager 服务生成。当您在轮换期间创建密钥的新版本时，Secrets Manager 会放弃轮换。有关轮换的更多信息，请参阅 轮换 密钥。

RotationFailed

在轮换失败时由 Secrets Manager 服务生成。有关轮换的更多信息，请参阅 the section called “轮换问题排查”。

RotationSucceeded

在成功轮换密钥时由 Secrets Manager 服务生成。有关轮换的更多信息，请参阅 轮换 密钥。

TestRotationStarted

在开始测试轮换某个尚未计划立即轮换的密钥时由 Secrets Manager 服务生成。有关轮换的更多信息，请参阅 轮换 密钥。

TestRotationSucceeded

在成功测试轮换某个尚未计划立即轮换的密钥时由 Secrets Manager 服务生成。有关轮换的更多信息，请参阅 轮换 密钥。

TestRotationFailed

在测试轮换某个尚未计划立即轮换的密钥但轮换失败时由 Secrets Manager 服务生成。有关轮换的更多信息，请参阅 the section called “轮换问题排查”。

AWS Secrets Manager 使用 Amazon 进行监控 CloudWatch

使用 Amazon CloudWatch，您可以监控 AWS 服务并创建警报，以便在指标发生变化时通知您。CloudWatch 将这些统计数据保存 15 个月，因此您可以访问历史信息并更好地了解 Web 应用程序或服务的性能。对于 AWS Secrets Manager，您可以监控账户中的密钥数量，包括标记为删除的密钥，以及对 Secrets Manager 的 API 调用，包括通过控制台进行的调用。有关如何监控指标的信息，请参阅CloudWatch 用户指南中的使用 CloudWatch 指标。

查找 Secrets Manager 指标

1. 在 CloudWatch 控制台的指标下，选择所有指标。
2. 在指标搜索框中，输入 `secret`。
3. 执行以下操作：
   - 要监控您账户中的密钥数量，请选择 AWS/SecretsManager，然后选择SecretCount。此指标每小时发布一次。
   - 要监控对 Secrets Manager 的 API 调用，包括通过控制台进行的调用，请选择 “使用情况” > “按 AWS 资源”，然后选择要监控的 API 调用。有关 Secrets Manager 的列表 APIs，请参阅 S ecrets Manager 的操作。
4. 执行以下操作：
   - 要创建指标图表，请参阅 Amazon CloudWatch 用户指南中的绘制指标图表。
   - 要检测异常，请参阅 A ma CloudWatch zon 用户指南中的使用 CloudWatch 异常检测。
   - 要获取指标的统计数据，请参阅 Amazon CloudWatch 用户指南中的获取指标的统计信息。

CloudWatch 警报

您可以创建一个 CloudWatch 警报，当指标值发生变化并导致警报状态发生变化时，该警报会发送 Amazon SNS 消息。您可以对 Secrets Manager 指标 `ResourceCount` 设置警报，该指标是您账户中的密钥数量。您可以设置警报来按照您指定的时间段监控某个指标，并根据该指标在若干时间段相对于给定阈值的值执行操作。警报仅针对持续的状态变化调用操作。 CloudWatch 警报不会仅仅因为它们处于特定状态就调用操作；该状态必须已更改并保持了指定的时间段。

有关更多信息，请参阅CloudWatch 用户指南中的使用 Amazon CloudWatch CloudWatch 警报和基于异常检测创建警报。

还可以设置特定阈值监视警报，在达到对应阈值时发送通知或采取行动。有关更多信息，请参阅 Amazon CloudWatch 用户指南。

将赛 AWS Secrets Manager 事与 Amazon 进行匹配 EventBridge

在亚马逊中 EventBridge，您可以匹配 CloudTrail 日志条目中的 Secrets Manager 事件。您可以配置 EventBridge 规则来查找这些事件，然后将新生成的事件发送到目标以采取行动。有关 Secrets Manager 记录的 CloudTrail 条目列表，请参阅CloudTrail 条目。有关设置说明 EventBridge，请参阅《EventBridge 用户指南》 EventBridge中的 “入门”。

将所有更改与指定密钥匹配

> **Note**
>
> 由于某些 Secrets Manager 事件返回的密钥 ARN 大小写不同，所以在匹配多个操作的事件模式中，您可能需要同时包含密钥 `arn` 和 `aRN` 才能通过 ARN 指定密钥。有关更多信息，请参阅 AWS re:Post。

以下示例显示了一种 EventBridge 事件模式，该模式与密钥更改的日志条目相匹配。

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

在轮换密钥值时匹配事件

以下示例显示了一种 EventBridge 事件模式，该模式与因手动更新或自动轮换而发生的机密值更改的 CloudTrail 日志条目相匹配。由于有些事件来自 Secrets Manager 操作，有些则由 Secrets Manager 服务生成，因此两者都必须包含 `detail-type`。

```
{
```

```
    "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"]
    }
}
```

监控计划删除的 AWS Secrets Manager 密钥何时被访问

您可以结合使用 Amazon CloudWatch 日志和亚马逊简单通知服务 (Amazon SNS) Simple Notification Service 来创建警报，通知您任何试图访问待删除的密钥的行为。 AWS CloudTrail如果您收到告警的通知，您可能需要取消删除密钥，以给自己更多时间来确定是否确实要将其删除。您的调查可能会导致密钥被恢复，因为您仍然需要它。或者，您可能需要使用所用的新密钥的详细信息来更新用户。

以下过程说明了当请求`GetSecretValue`操作导致将特定错误消息写入 CloudTrail 日志文件时，如何收到通知。可以对密钥执行其他 API 操作而不会触发告警。此 CloudWatch 警报检测到可能表明某人或应用程序使用过期凭证的使用情况。

在开始这些过程之前，您必须在要监控 AWS Secrets Manager API 请求的 AWS 区域 和账户 CloudTrail 中开启。有关说明，请参阅 AWS CloudTrail 《用户指南》中的首次创建跟踪。

步骤 1：配置 CloudTrail 日志文件传输到 CloudWatch 日志

您必须配置将 CloudTrail 日志文件传送到 CloudWatch 日志。这样做是为了让 CloudWatch Logs 可以监控它们是否有 Secrets Manager API 请求以检索待删除的密钥。

配置 CloudTrail 日志文件传输到 CloudWatch 日志

1. 打开 CloudTrail 控制台，网址为https://console.aws.amazon.com/cloudtrail/。
2. 在顶部导航栏上，选择 AWS 区域 要监控密钥。
3. 在左侧导航窗格中，选择 Tra il s，然后选择要为其配置的跟踪的名称 CloudWatch。
4. 在 T rails 配置页面上，向下滚动到 “CloudWatch 日志” 部分，然后选择编辑图标 ( )。
5. 对于 New or existing log group (新的或现有的日志组)，键入日志组的名称，例如 **`CloudTrail/MyCloudWatchLogGroup`**。

6. 对于 IAM 角色，您可以使用名为 CloudTrail_ CloudWatchLogs _Role 的默认角色。此角色具有默认角色策略，该策略具有向日志组传送 CloudTrail 事件所需的权限。
7. 选择 Continue (继续) 以保存您的配置。
8. 在AWS CloudTrail 将与您账户中 API 活动关联 CloudTrail 的事件发送到 CloudWatch 日志日志组页面上，选择允许。

步骤 2：创建 CloudWatch警报

要在 Secrets Manager `GetSecretValue` API 操作请求访问待删除的密钥时收到通知，您必须创建 CloudWatch 警报并配置通知。

创建 CloudWatch 警报

1. 登录 CloudWatch 控制台，网址为https://console.aws.amazon.com/cloudwatch/。
2. 在顶部导航栏上，选择要监控密钥的 AWS 区域。
3. 在左侧导航窗格中，选择 Logs。
4. 在日志组列表中，选中您在上一个过程中创建的日志组旁边的复选框，例如 CloudTrail/MyCloudWatchLogGroup。选择创建指标筛选器。
5. 对于筛选模式，键入或粘贴以下内容：

   ```
   { $.eventName = "GetSecretValue" && $.errorMessage = "*secret because it was marked
    for deletion*" }
   ```

   选择 Assign Metric (分配指标)。
6. 在 Create Metric Filter and Assign a Metric 页面上，执行以下操作：
   a. 对于 Metric Namespace (指标命名空间)，键入 **`CloudTrailLogMetrics`**。
   b. 对于 Metric Name（指标名称），键入 **`AttemptsToAccessDeletedSecrets`**。
   c. 选择显示高级指标设置，如有必要，再为 Metric Value (指标值) 键入 **1**。
   d. 选择 Create Filter。
7. 在筛选器框中，选择 Create Alarm。
8. 在 Create Alarm 窗口中，执行以下操作：
   a. 对于名称，键入 **`AttemptsToAccessDeletedSecretsAlarm`**。
   b. 在 Whenever: (每当:) 下，为 is: (是:) 选择 **>=** 并键入 **1**。

c. 在 Send notification to: 旁，执行以下操作之一：

- 要创建并使用新的 Amazon SNS 主题，请选择新建列表，然后键入新主题的名称。对于 Email list:，键入至少一个电子邮件地址。您可以键入多个电子邮件地址，并使用逗号将它们隔开。
- 要使用现有 Amazon SNS 主题，请选择要使用的主题的名称。如果列表不存在，请选择 Select list (选择列表)。

d. 选择创建警报。

第 3 步：测试 CloudWatch警报

要测试告警，请创建密钥，并计划将其删除。然后，尝试检索密钥值。您在告警中配置的地址很快会收到一封电子邮件。它提醒您注意计划删除的密钥的使用情况。

使用以下方法监控 AWS Secrets Manager 密钥的合规性 AWS Config

您可以使用 AWS Config 来评估您的密钥，以查看它们是否符合您的标准。您可以使用 AWS Config 规则定义对机密的内部安全和合规性要求。然后 AWS Config 可以识别不符合你规则的秘密。您还可以跟踪对密钥元数据、轮换配置、用于密钥加密的 KMS 密钥、Lambda 轮换函数以及与密钥关联的标签等进行的更改。

您可以配置 AWS Config 为将更改通知您。有关更多信息，请参阅AWS Config 发送至 Amazon SNS 主题的通知。

如果您的组织中有多个 AWS 账户 密钥，则可以聚合该配置和合规性数据。 AWS 区域 有关更多信息，请参阅 Multi-account Multi-Region data aggregation。

评测密钥是否合规

- 按照使用AWS Config 规则评估资源中的说明进行操作，然后选择以下规则之一：
  - `secretsmanager-secret-unused` — 检查是否已在指定的天数内访问了密钥。
  - `secretsmanager-using-cmk`— 检查密钥是否使用您在中创建的客户托管密钥 AWS 托管式密钥 `aws/secretsmanager`或客户管理的密钥进行加密 AWS KMS。
  - `secretsmanager-rotation-enabled-check` — 检查是否为存储在 Secrets Manager 中的密钥配置了轮换。

- `secretsmanager-scheduled-rotation-success-check`— 检查上次成功的轮换是否在配置的轮换频率内。检查的最低频率为每天。
- `secretsmanager-secret-periodic-rotation` — 检查是否已在指定的天数内轮换了密钥。

监控 Secrets Manager 成本

您可以使用 Amazon CloudWatch 来监控预估 AWS Secrets Manager 费用。有关更多信息，请参阅CloudWatch 用户指南中的创建账单警报以监控您的预估 AWS 费用。

监控成本的另一种选择是 AWS 成本异常检测。有关更多信息，请参阅《AWS 成本管理用户指南》中的 Detecting unusual spend with AWS Cost Anomaly Detection。

有关监控 Secrets Manager 使用情况的信息，请参阅 the section called “使用监视器 CloudWatch” 和 the section called “使用登录 AWS CloudTrail ”。

有关 AWS Secrets Manager 定价的信息，请参阅the section called “定价”。

使用 Amazon 检测威胁 GuardDuty

Amazon GuardDuty 是一项威胁检测服务，可帮助您保护您的账户、容器、工作负载和 AWS 环境中的数据。通过使用机器学习 (ML) 模型以及异常和威胁检测功能， GuardDuty 持续监控不同的日志源，以识别环境中的潜在安全风险和恶意活动并确定其优先级。例如， GuardDuty 如果它检测到通过 EC2 实例启动角色专为 Amazon 实例创建但正在从其中的其他账户使用的证书，则会检测到潜在威胁，例如对机密的异常或可疑访问以及凭证泄露。 AWS有关更多信息，请参阅 Amazon GuardDuty 用户指南。

检测的另一个示例使用场景是异常行为。例如，如果 AWS Secrets Manager 通常使用 Java SDK 从实体`list-secrets`接收、、和调用，然后另一个实体开始 AWS CLI 从 VPN 外部调用`batch-get-secret-value`和`get-secret-value`使用，则 GuardDuty 可以报告发现第二个实体正在异常调用。`create-secret` `get-secret-value` `describe-secret` APIs有关更多信息，请参阅 GuardDuty IAM 查找类型 CredentialAccess：IAMUser/AnomalousBehavior。

合规性验证 AWS Secrets Manager

您在使用 Secrets Manager 时的合规责任取决于您的数据的敏感度、贵公司的合规目标以及适用的法律和法规。 AWS 提供了以下资源来帮助实现合规性：

- 安全性与合规性快速入门指南 - 这些部署指南讨论了架构注意事项，并提供了在 AWS上部署基于安全性和合规性的基准环境的步骤。
- HIPAA 安全与合规架构白皮书 — 本白皮书描述了公司如何使用来 AWS 创建符合 HIPAA 标准的应用程序。
- AWS 合AWS 规资源 — 此工作簿和指南集可能适用于您的行业和所在地区。
- AWS Config 会评估资源配置符合内部实践、行业指南和法规的情况。有关更多信息，请参阅 the section called “监控密钥的合规性”。
- AWS Security Hub提供了您的安全状态的全面视图 AWS ，可帮助您检查自己是否符合安全行业标准和最佳实践。有关使用 Security Hub 评估 Secrets Manager 资源的信息，请参阅《AWS Security Hub 用户指南》中的 AWS Secrets Manager 控件。
- IAM Access Analyzer 会分析允许外部实体访问密钥的策略，包括策略中的条件语句。有关更多信息，请参阅使用 Access Analyzer 预览访问权限。
- AWS Systems Manager 为 Secrets Manager 提供了预定义的运行手册。有关更多信息，请参阅适用于 Secrets Manager 的 Systems Manager 自动化运行手册参考。
- 您可以使用下载第三方审计报告 AWS Artifact。有关更多信息，请参阅中的 “下载报告” 中的 “ AWS Artifact。

合规性标准

AWS Secrets Manager 已经过以下标准的审计，当您需要获得合规性认证时，可以成为您的解决方案的一部分。

- HIPAA — AWS 已扩大其《健康保险流通与责任法案》（HIPAA）合规计划，将其列 AWS Secrets Manager 为符合HIPAA资格的服务。如果您与签订了商业伙伴协议 (BAA) AWS，则可以使用 Secrets Manager 来帮助构建符合 HIPAA 标准的应用程序。 AWS 为有兴趣进一步了解如何利用健康信息处理和存储的客户提供了一份以 HIPAA AWS 为重点的白皮书。有关更多信息，请参阅 HIPAA 合规性。
- PCI 参与组织 — AWS Secrets Manager 拥有服务提供商级别 1 的支付卡行业 (PCI) 数据安全标准 (DSS) 3.2 版合规认证。使用 AWS 产品和服务存储、处理或传输持卡人数据的客户可以在管理自己

的 PCI DSS 合规性认证时使用 AWS Secrets Manager 。有关 PCI DSS 的更多信息，包括如何申请 PCI Compliance Package 的副本，请参阅 AWS PCI DSS 第 1 级。

- ISO — AWS Secrets Manager 已成功完成 ISO/IEC 27001, ISO/IEC 27017, ISO/IEC 27018 和 ISO 9001 的合规性认证。有关更多信息，请参阅 ISO 27001、ISO 27017、ISO 27018、ISO 9001。
- AICPA SOC – 系统和组织控制（SOC）报告是独立的第三方检查报告，用于说明 Secrets Manager 如何实现关键合规性控制和目标。这些报告的目的是帮助您和您的审计师了解为支持运营和合规而建立的 AWS 控制措施。有关更多信息，请参阅 SOC 合规性。
- FedRAMP – 联邦风险与授权管理计划（FedRAMP）是一项政府层面的计划，它提供一种标准化方法来对云产品和云服务进行安全性评测、授权以及持续监控。FedRAMP计划还为东西方 GovCloud 服务和地区以及使用政府或监管数据提供了临时授权。有关更多信息，请参阅 FedRAMP 合规性。
- 国防部 — 国防部 (DoD) 云计算安全要求指南 (SRG) 为云服务提供商 (CSPs) 提供了标准化的评估和授权流程，以获得国防部的临时授权，以便他们能够为国防部客户提供服务。有关更多信息，请参阅 DoD SRG 资源。
- IRAP – 通过信息安全注册评估员计划（IRAP），澳大利亚政府客户能够验证适当的控制措施是否到位，并确定适当的责任模式，满足澳大利亚网络安全中心编制的《澳大利亚政府信息安全手册》（ISM）的要求（ACSC）。有关更多信息，请参阅 IRAP 资源。
- OSPAR — Amazon Web Services (AWS) 获得了外包服务提供商的审计报告 (OSPAR) 认证。 AWS 与新加坡银行协会（ABS）的《外包服务提供商控制目标和程序指南》（ABS 指南）保持一致，这向客户表明，他们 AWS 致力于满足新加坡金融服务行业对云服务提供商设定的高期望。有关更多信息，请参阅 OSPAR 资源。

安全性 AWS Secrets Manager

安全性 AWS 是重中之重。作为 AWS 客户，您可以受益于专为满足大多数安全敏感型组织的要求而构建的数据中心和网络架构。

您和您 AWS 共同承担安全责任。责任共担模型将其描述为云的安全性和云中的安全性：

- 云安全 — AWS 负责保护在 AWS 云中运行 AWS 服务的基础架构。 AWS 还为您提供可以安全使用的服务。作为 AWS 合规性计划的一部分，第三方审核人员将定期测试和验证安全性的有效性。要了解适用的合规计划 AWS Secrets Manager，请参阅按合规计划划分的范围内的AWS 服务。
- 云端安全 — 您的 AWS 服务决定您的责任。您还需要对其他因素负责，包括您的数据的敏感性、您公司的要求以及适用的法律法规。

有关更多资源，请参阅安全支柱 — Well-Archit AWS ected 框架。

主题

- 降低使用存储 AWS Secrets Manager 密钥 AWS CLI 的风险
- 的身份验证和访问控制 AWS Secrets Manager
- 中的数据保护 AWS Secrets Manager
- 中的秘密加密和解密 AWS Secrets Manager
- 中的基础设施安全 AWS Secrets Manager
- 使用 AWS Secrets Manager VPC 终端节点
- 使用 IAM 策略控制 API 访问
- 灵活性 AWS Secrets Manager
- 后量子 TLS

降低使用存储 AWS Secrets Manager 密钥 AWS CLI 的风险

使用 AWS Command Line Interface (AWS CLI) 调用 AWS 操作时，可以在命令外壳中输入这些命令。例如，你可以使用 Windows 命令提示符或 Windows PowerShell，或者使用 Bash 或 Z shell 等。其中的很多命令 Shell 包含旨在提高工作效率的功能。但其他人可能会使用该功能窃取您的密钥。例如，在大多数 Shell 中，您可以使用上箭头键查看最后输入的命令。访问不受保护的会话的任何人可能会利用命令历史记录功能。另外，在后台工作的其他实用程序可能有权访问您的命令参数，这些参数旨在帮助您更高效地执行任务。为了减轻此类风险，请确保执行以下步骤：

- 在离开计算机前始终记得锁定计算机。
- 卸载或禁用不需要或不再使用的控制台实用程序。
- 确保 Shell 或远程访问程序（如果您在使用其中之一）不会记录键入的命令。
- 使用一些方法传递 Shell 命令历史记录未捕获的参数。以下示例说明如何将密文键入文本到文本文件中，然后将该文件传递给 AWS Secrets Manager 命令并立即销毁该文件。这意味着典型的 Shell 历史记录不会捕获密钥文本。

  以下示例显示典型的 Linux 命令（但您的 shell 可能需要稍有不同的命令）：

```
$ 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.
```

运行这些命令后，在使用向上和向下箭头滚动命令历史记录时就不会在任何一行中看到密钥文本。

> **Important**
>
> 默认情况下，不能在 Windows 中使用这类技术，除非您先将命令历史记录的缓冲区大小减小为 1。

将 Windows 命令提示符配置为 1 条命令只有 1 条命令的历史记录缓冲区

1. 以管理员身份打开命令提示符（以管理员身份运行）。
2. 选择左上角的图标，然后选择 “属性”。
3. 在选项选项卡上，将缓冲区大小和缓冲区数均设置为 **1**，然后选择确定。
4. 每次您必须键入不希望保留在历史记录中的命令时，请在紧靠该命令后面键入另一条命令，例如：

```
echo.
```

这可以确保您刷新敏感命令。

对于 Windows 命令提示符外壳，你可以下载该SysInternals SDelete工具，然后使用类似于以下内容的命令：

```
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.
```

的身份验证和访问控制 AWS Secrets Manager

Secrets Manager 用AWS Identity and Access Management (IAM) 来保护密钥的访问权限。IAM 提供了身份验证和访问控制。身份验证确认个人请求的身份。Secrets Manager 使用密码、访问密钥登录过程和多重身份验证（MFA）令牌来验证用户身份。请参阅登录 AWS。访问控制确保只有获得批准的个人才能对密钥等 AWS 资源执行操作。Secrets Manager 使用策略来定义谁有权访问哪些资源，以及身份可以对这些资源执行哪些操作。参见 IAM 中的策略和权限。

主题

- 的权限参考 AWS Secrets Manager
- Secrets Manager 管理员权限
- 访问密钥的权限
- Lambda 轮换函数的权限
- 加密密钥权限

- 复制权限
- 基于身份的策略
- 基于资源的策略
- 使用基于属性的访问权限控制（ABAC）控制对密钥的访问
- AWS 的托管策略 AWS Secrets Manager
- 确定谁有权访问你的 AWS Secrets Manager 秘密
- 从其他账户访问 AWS Secrets Manager 密钥
- 从本地环境访问密钥

的权限参考 AWS Secrets Manager

Secrets Manager 的权限参考可在服务授权参考中的 Actions, resources, and condition keys for AWS Secrets Manager 中找到。

Secrets Manager 管理员权限

如果要授予 Secrets Manager 管理员权限，请根据添加和删除 IAM 身份权限和下列策略进行操作：

- SecretsManagerReadWrite
- IAMFullAccess

我们建议您不要向最终用户授予管理员权限。尽管这样用户可以创建和管理密钥，但启用轮换所需的权限 (IAMFullAccess) 会授予不适合最终用户的重要权限。

访问密钥的权限

通过采用 IAM 权限策略，您可以控制哪些用户或服务有权访问您的密钥。权限策略描述了哪些人可以对哪些资源执行哪些操作。您可以：

- the section called “基于身份的策略”
- the section called “基于资源的策略”

Lambda 轮换函数的权限

Secrets Manager 使用 AWS Lambda 函数来轮换密钥。Lambda 函数必须具有对密钥以及密钥包含凭据的数据库或服务的访问权限。请参阅轮换权限。

加密密钥权限

Secrets Manager 使用 AWS Key Management Service (AWS KMS) 密钥来加密机密。 AWS 托管式密钥 `aws/secretsmanager`自动具有正确的权限。如果您使用不同的 KMS 密钥，Secrets Manager 需要对该密钥的权限。请参阅the section called “KMS 密钥的权限”。

复制权限

通过使用 IAM 权限策略，您可以控制哪些用户或服务可以将您的密钥复制到其他区域。请参阅the section called “防止复制”。

基于身份的策略

将权限策略附加到 IAM 身份、用户、用户组和角色。在基于身份的策略中，您指定该身份可以访问哪些密钥以及该身份可以对密钥执行哪些操作。有关更多信息，请参阅添加和删除 IAM 身份权限。

您可以向代表其他服务中的应用程序或用户的角色授予权限。例如，在 Amazon EC2 实例上运行的应用程序可能需要访问数据库。您可以创建附加到 EC2 实例配置文件的 IAM 角色，然后使用权限策略向该角色授予访问包含数据库证书的密钥的权限。有关更多信息，请参阅使用 IAM 角色向在 Amazon EC2 实例上运行的应用程序授予权限。您可以附加角色的其他服务包括 Amazon Redshift、AWS Lambda 和 Amazon ECS。

您还可以通过除 IAM 以外的其他身份系统验证的用户授予权限。例如，您可以将 IAM 角色与使用 Amazon Cognito 登录的移动应用程序用户关联。角色向应用程序授予具有角色权限策略中权限的临时凭据。然后，您可以使用权限策略授予角色对密钥的访问权限。有关更多信息，请参阅身份提供者和联合身份验证。

您必须使用基于身份的策略来：

- 授予对多个密钥的身份访问权限。
- 控制哪些人可以创建新密钥，哪些人可以访问尚未创建的密钥。
- 授予 IAM 组对密钥的访问权限。

示例：

- 示例：检索单个秘密值的权限
- 示例：读取和描述个人密钥的权限
- 示例：批量检索一组密钥值的权限

- 示例：通配符
- 示例：创建密钥的权限
- 示例：拒绝使用特定 AWS KMS 密钥来加密机密

示例：检索单个秘密值的权限

要授予检索密钥值的权限，您可以将策略附加到密钥或身份上。要帮助确定使用的策略类型，请参阅基于身份的策略和基于资源的策略。有关如何附加策略的信息，请参阅 the section called “基于资源的策略” 和 the section called “基于身份的策略”。

当您希望授予对 IAM 组的访问权限时，此示例非常有用。要授予在批处理 API 调用中检索一组秘密的权限，请参阅 the section called “示例：批量检索一组密钥值的权限”。

Example 读取使用客户自主管理型密钥加密的密钥

如果使用客户管理的密钥对密钥进行加密，则可以通过将以下策略附加到身份来授予读取该密钥的访问权限。

JSON

```
{
"Version": "2012-10-17",
"Statement": [
{
  "Effect": "Allow",
  "Action": "secretsmanager:GetSecretValue",
  "Resource": "arn:aws:secretsmanager:us-east-1:123456789012:secret:secretName-AbCdEf"
},
{
  "Effect": "Allow",
  "Action": "kms:Decrypt",
  "Resource": "arn:aws:kms:us-east-1:123456789012:key/key-id"
}
]
}
```

示例：读取和描述个人密钥的权限

Example 读取和描述一个密钥

通过将以下策略附加到身份，您可以授予密钥的访问权限。

JSON

```
{
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "secretsmanager:GetSecretValue",
      "secretsmanager:DescribeSecret"
      ],
      "Resource": "arn:aws:secretsmanager:us-
east-1:123456789012:secret:secretName-AbCdEf"
    }
]
}
```

示例：批量检索一组密钥值的权限

Example 批量读取一组密钥

您可以通过将以下策略附加到身份来授予在批处理 API 调用中检索一组秘密的访问权限。该策略限制了调用方，因此即使批量调用包含其他密钥 *SecretARN1SecretARN2*，他们也只能检索*SecretARN3*、和指定的密钥。如果调用方还在批处理 API 调用中请求其他秘密，则 Secrets Manager 将不会返回它们。有关更多信息，请参阅BatchGetSecretValue。 。

JSON

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

```
  "Action": [
    "secretsmanager:BatchGetSecretValue",
    "secretsmanager:ListSecrets"
  ],
  "Resource": "*"
},
{
  "Effect": "Allow",
  "Action": [
    "secretsmanager:GetSecretValue"
  ],
  "Resource": [
    "arn:aws:secretsmanager:us-east-1:123456789012:secret:secretName1-AbCdEf",
    "arn:aws:secretsmanager:us-east-1:123456789012:secret:secretName2-AbCdEf",
    "arn:aws:secretsmanager:us-east-1:123456789012:secret:secretName3-AbCdEf"
    ]
}
]
}
```

示例：通配符

您可以使用通配符在策略元素中包含一组值。

Example 访问路径中的所有密钥

以下策略授予检索名称以 “*TestEnv/*” 开头的所有密钥的权限。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "secretsmanager:GetSecretValue",
        "Resource": "arn:aws:secretsmanager:us-
east-1:123456789012:secret:TestEnv/*"
    }
}
```

Example 访问所有密钥的元数据

以下策略授予 `DescribeSecret` 和权限开头为 `List`：`ListSecrets` 和 `ListSecretVersionIds`。

JSON

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

Example 匹配密钥名称

以下策略按名称授予密钥的所有 Secrets Manager 权限。要使用该策略，请参阅 the section called “基于身份的策略”。

要匹配密钥名称，可以通过将区域、账户 ID、机密名称和通配符 (?) 放在一起来匹配单个随机字符，从而为密钥创建 ARN。Secrets Manager 会将六个随机字符附加到密钥名称作为 ARN 的一部分，因此您可以使用此通配符来匹配这些字符。如果使用 `"another_secret_name-*"` 语法，Secrets Manager 不仅匹配具有 6 个随机字符的预期密钥，而且还匹配 `"another_secret_name-<anything-here>a1b2c3"`。

因为除了 6 个随机字符外，您可以预测密钥的所有 ARN 部分，所以使用通配符 `'??????'` 语法，您能够安全地将权限授予给尚不存在的密钥。但要注意，如果删除密钥并以相同名称重新创建，即使 6 个字符发生变化，用户也会自动获得新密钥的权限。

JSON

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

```
        {
            "Effect": "Allow",
            "Action": "secretsmanager:*",
            "Resource": [
                "arn:aws:secretsmanager:us-
east-1:123456789012:secret:a_specific_secret_name-a1b2c3",
                "arn:aws:secretsmanager:us-
east-1:123456789012:secret:another_secret_name-??????"
            ]
        }
    ]
}
```

示例：创建密钥的权限

要为用户授予权限创建密钥，我们建议您将权限策略附加到用户所属的 IAM 组。请参阅 IAM 用户组。

Example 创建密钥

以下策略授予创建密钥和查看密钥列表的权限。要使用该策略，请参阅 the section called “基于身份的策略”。

JSON

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

示例：拒绝使用特定 AWS KMS 密钥来加密机密

> **Important**
>
> 要拒绝客户自主管理型密钥，我们建议您使用密钥政策或密钥授予来限制访问权限。有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的 Authentication and access control for AWS KMS。

Example 拒绝 AWS 托管密钥 **aws/secretsmanager**

以下策略拒绝使用来创建或更新密钥。 AWS 托管式密钥 `aws/secretsmanager`此政策要求使用客户管理的密钥对机密进行加密。该策略包含两条语句：

1. 第一条语句拒绝使用创建或更新密钥的请求 AWS 托管式密钥 `aws/secretsmanager`。`Sid: "RequireCustomerManagedKeysOnSecrets"`
2. 第二条语句拒绝创建不包含 KMS 密钥的密钥的请求，因为 Secrets Manager 会默认使用 AWS 托管式密钥 `aws/secretsmanager`。`Sid: "RequireKmsKeyIdParameterOnCreate"`

JSON

```
{
"Version": "2012-10-17",
"Statement": [
  {
      "Sid": "RequireCustomerManagedKeysOnSecrets",
      "Effect": "Deny",
      "Action": [
          "secretsmanager:CreateSecret",
          "secretsmanager:UpdateSecret"
      ],
      "Resource": "*",
      "Condition": {
          "StringLikeIfExists": {
              "secretsmanager:KmsKeyArn": "<key_ARN_of_the_AWS_managed_key>"
          }
      }
  },
  {
```

```
        "Sid": "RequireKmsKeyIdParameterOnCreate",
        "Effect": "Deny",
        "Action": "secretsmanager:CreateSecret",
        "Resource": "*",
        "Condition": {
            "Null": {
                "secretsmanager:KmsKeyArn": "true"
            }
        }
    }
]
}
```

基于资源的策略

在基于资源的策略中，您可以指定谁可以访问密钥，以及他们可以对密钥执行哪些操作。您可以使用基于资源的策略来：

- 为多个用户或角色授予单个密钥的访问权限。
- 向其他 AWS 账户中的用户或角色授予访问权限。

当您将基于资源的策略附加到控制台中的密钥时，Secrets Manager 使用自动推理引擎 Zelkova 和 API `ValidateResourcePolicy`，防止您向各种 IAM 委托人授予对您的密钥的访问权限。您也可以调用带有来自 CLI 或 SDK `BlockPublicPolicy` 参数的 `PutResourcePolicy` API。

> **Important**
>
> 资源策略验证和 `BlockPublicPolicy` 参数通过阻止利用直接附加到您的密钥的资源策略授予公共访问权限，来帮助保护您的资源。除了使用这些功能之外，还要仔细检查以下策略，来确认它们不授予公有访问权限：
>
> - 附加到关联 AWS 委托人（例如，IAM 角色）的基于身份的策略
> - 附加到关联资源的基于 AWS 资源的策略（例如， AWS Key Management Service (AWS KMS) 密钥）
>
> 要查看对您的密钥的权限，请参阅 确定谁有权限访问您的 密钥。

查看、更改或删除密钥的资源策略（控制台）

1. 打开 Secrets Manager 控制台，网址为https://console.aws.amazon.com/secretsmanager/。
2. 从密钥列表上，选择您的密钥。
3. 进入密钥详细信息页面后，在概述选项卡的资源权限部分中，选择编辑权限。
4. 在代码字段中，执行以下操作之一，然后选择保存：
   - 要附加或修改资源策略，输入该策略。
   - 要删除策略，清除代码字段。

AWS CLI

Example 检索资源策略

以下 `get-resource-policy` 示例将检索附加到密钥的基于资源的策略。

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

Example 删除资源策略

以下 `delete-resource-policy` 示例将删除附加到密钥的基于资源的策略。

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

Example 添加资源策略

以下 `put-resource-policy` 示例将向密钥添加权限策略，首先检查该策略是否不提供对该密钥的广泛访问权限。该策略是从文件中读取的。有关更多信息，请参阅 AWS CLI 用户指南中的从文件加载 AWS CLI 参数。

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

`mypolicy.json` 的内容：

JSON

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

AWS SDK

要检索附加到密钥的策略，请使用 GetResourcePolicy。

要删除附加到密钥的策略，请使用 DeleteResourcePolicy。

要将策略附加到密钥，请使用 PutResourcePolicy。如果已经附加了策略，命令会将其替换为新策略。策略必须格式化为 JSON 结构化文本。请参阅 JSON 策略文档结构。

有关更多信息，请参阅 the section called “AWS SDKs”。

示例

示例：

- 示例：检索单个秘密值的权限
- 示例：权限和 VPCs
- 示例：服务主体

示例：检索单个秘密值的权限

要授予检索密钥值的权限，您可以将策略附加到密钥或身份上。要帮助确定使用的策略类型，请参阅基于身份的策略和基于资源的策略。有关如何附加策略的信息，请参阅 the section called “基于资源的策略” 和 the section called “基于身份的策略”。

当您希望向多个用户或角色授予单个密钥的访问权限时，此示例非常有用。要授予在批处理 API 调用中检索一组秘密的权限，请参阅 the section called “示例：批量检索一组密钥值的权限”。

Example 读取一个密钥

通过将以下策略附加到密钥，你可以授予密钥的访问权限。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/EC2RoleToAccessSecrets"
            },
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "*"
        }
    ]
}
```

示例：权限和 VPCs

如果您需要在 VPC 内部访问 Secrets Manager，您可以通过在权限策略中包含条件来确保对 Secrets Manager 的请求来自 VPC。有关更多信息，请参阅使用 VPC 端点条件限制请求和the section called “VPC 端点（AWS PrivateLink）”。

确保从其他 AWS 服务访问密钥的请求也来自 VPC，否则此策略将拒绝他们访问。

Example 要求请求通过 VPC 端点

以下策略仅允许用户在请求通过 VPC 终端节点时执行 Secrets Manager 操作 *vpce-1234a5678b9012c*。

JSON

```
{
"Id": "example-policy-1",
```

```
"Version": "2012-10-17",
"Statement": [
{
  "Sid": "RestrictGetSecretValueoperation",
  "Effect": "Deny",
  "Principal": "*",
  "Action": "secretsmanager:GetSecretValue",
  "Resource": "*",
  "Condition": {
    "StringNotEquals": {
      "aws:sourceVpce": "vpce-12345678"
    }
  }
}
]
}
```

Example 要求请求来自 VPC

以下示例密钥策略仅允许来自 *vpc-12345678* 的命令创建和管理密钥。此外，只有在请求来自于 vpc-2b2b2b2b 时，该策略才允许使用访问密钥的加密值的操作。如果您在一个 VPC 中运行应用程序，但使用第二个隔离的 VPC 以提供管理功能，则可能会使用此类策略。

JSON

```
{
"Id": "example-policy-2",
"Version": "2012-10-17",
"Statement": [
{
  "Sid": "AllowAdministrativeActionsfromONLYvpc-12345678",
  "Effect": "Deny",
  "Principal": "*",
  "Action": [
    "secretsmanager:Create*",
    "secretsmanager:Put*",
    "secretsmanager:Update*",
    "secretsmanager:Delete*",
    "secretsmanager:Restore*",
    "secretsmanager:RotateSecret",
    "secretsmanager:CancelRotate*",
```

```
    "secretsmanager:TagResource",
    "secretsmanager:UntagResource"
  ],
  "Resource": "*",
  "Condition": {
    "StringNotEquals": {
      "aws:sourceVpc": "vpc-12345678"
    }
  }
},
{
  "Sid": "AllowSecretValueAccessfromONLYvpc-2b2b2b2b",
  "Effect": "Deny",
  "Principal": "*",
  "Action": [
    "secretsmanager:GetSecretValue"
  ],
  "Resource": "*",
  "Condition": {
    "StringNotEquals": {
      "aws:sourceVpc": "vpc-2b2b2b2b"
    }
  }
}
]
}
```

示例：服务主体

如果附加到您的密钥的资源策略包含AWS 服务委托人，我们建议您使用 aws: SourceArn 和 aw s: SourceAccount 全局条件密钥。仅当请求是从另一项 AWS 服务发送到 Secrets Manager 时，ARN 和账户值才会包含在授权上下文中。这种条件的组合避免了潜在的混淆代理情况。

如果资源 ARN 包含资源策略中不允许使用的字符，则不能在 `aws:SourceArn` 条件键的值中使用该资源 ARN 。改为使用 `aws:SourceAccount` 条件键。有关更多信息，请参阅 IAM 要求。

在附加到机密的策略中，服务委托人通常不用作委托人，但有些 AWS 服务需要它。有关服务要求您附加到密钥的资源策略的信息，请参阅该服务的文档。

Example 允许服务使用服务主体访问密钥

JSON

```
{
"Version": "2012-10-17",
"Statement": [
{
  "Effect": "Allow",
  "Principal": {
    "Service": [
      "s3.amazonaws.com"
    ]
  },
  "Action": "secretsmanager:GetSecretValue",
  "Resource": "*",
  "Condition": {
    "ArnLike": {
      "aws:sourceArn": "arn:aws:s3::123456789012:*"
    },
    "StringEquals": {
      "aws:sourceAccount": "123456789012"
    }
  }

}
]
}
```

使用基于属性的访问权限控制（ABAC）控制对密钥的访问

基于属性的访问权限控制（ABAC）是一种授权策略，它根据用户、数据或环境（例如部门、业务部门或可能影响授权结果的其他因素）的属性或特征来定义权限。在中 AWS，这些属性称为标签。

使用标签控制权限在快速增长的环境中非常有用，并在策略管理变得繁琐的情况下可以提供帮助。ABAC 规则是在运行时动态评估的，这意味着用户对应用程序和数据的访问以及允许的操作类型会根据策略中的上下文因素自动改变。例如，如果用户更改部门，则访问权限会自动调整，而无需更新权限或请求新角色。有关更多信息，请参阅：ABAC 有什么用 AWS？ ，根据标签定义访问密钥的权限。，然后使用带有 IAM 身份中心的 ABAC 来扩展 Secrets Manager 的授权需求。

示例：允许身份访问具有特定标签的密钥

以下策略允许DescribeSecret访问带有密钥*ServerName*和值的标签的机密*ServerABC*。如果将此策略附加到某个身份，则该身份有权访问账户中带有该标签的任何密钥。

JSON

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

示例：仅允许访问标签与密钥的标签匹配的身份

以下策略允许账户中的任何身份对账户中身份的 *AccessProject* 标签与密钥的 *AccessProject* 标签具有相同值的任何密钥具有 GetSecretValue 访问权限。

JSON

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

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

AWS 的托管策略 AWS Secrets Manager

AWS 托管策略是由创建和管理的独立策略 AWS。 AWS 托管策略旨在为许多常见用例提供权限，以便您可以开始为用户、组和角色分配权限。

请记住， AWS 托管策略可能不会为您的特定用例授予最低权限权限，因为它们可供所有 AWS 客户使用。我们建议通过定义特定于您的使用场景的客户管理型策略来进一步减少权限。

您无法更改 AWS 托管策略中定义的权限。如果 AWS 更新 AWS 托管策略中定义的权限，则更新会影响该策略所关联的所有委托人身份（用户、组和角色）。 AWS 最有可能在启动新的 API 或现有服务可以使用新 AWS 服务 的 API 操作时更新 AWS 托管策略。

有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管策略。

AWS 托管策略： SecretsManagerReadWrite

本政策提供 read/write 访问权限 AWS Secrets Manager，包括描述亚马逊 RDS、Amazon Redshift 和 Amazon DocumentDB 资源的权限，以及 AWS KMS 用于加密和解密密密钥的权限。该策略还允许创建 AWS CloudFormation 变更集、从由管理的 Amazon S3 存储桶获取轮换模板 AWS、列出 AWS Lambda 函数和描述 Amazon EC2 VPCs。控制台需要这些权限才能使用现有的轮换函数设置轮换。

要创建新的轮换函数，您还必须拥有创建 AWS CloudFormation 堆栈和 AWS Lambda 执行角色的权限。您可以分配IAMFull访问管理策略。请参阅轮换权限。

权限详细信息

该策略包含以下权限。

- `secretsmanager` – 允许主体执行所有 Secrets Manager 操作。
- `cloudformation`— 允许委托人创建 AWS CloudFormation 堆栈。这是必需的，以便使用控制台开启轮换功能的委托人可以通过堆栈创建 Lambda 轮换函数 AWS CloudFormation 。有关更多信息，请参阅 the section called “Secrets Manager 的使用方式 AWS CloudFormation”。
- `ec2`— 允许校长描述亚马逊 EC2 VPCs。这是必需的条件，这样使用控制台的主体才能在与其存储在密钥中的凭证数据库相同的 VPC 中创建轮换函数。

- `kms`— 允许委托人使用 AWS KMS 密钥进行加密操作。这是必需的条件，这样 Secrets Manager 才能加密和解密密钥。有关更多信息，请参阅 the section called “密钥加密和解密”。
- `lambda` – 允许主体列出 Lambda 轮换函数。这是必需的条件，以便使用控制台的主体可以选择现有的轮换函数。
- `rds` – 允许主体描述 Amazon RDS 中的集群和实例。这是必需的条件，以便使用控制台的主体可以选择 Amazon RDS 集群或实例。
- `redshift` – 允许主体描述 Amazon Redshift 中的集群。这是必需的条件，以便使用控制台的主体可以选择 Amazon Redshift 集群。
- `redshift-serverless` – 允许主体描述 Amazon Redshift Serverless 中的命名空间。这是必需的，以便使用控制台的主体可以选择 Amazon Redshift Serverless 命名空间。
- `docdb-elastic` – 允许主体描述 Amazon DocumentDB 中的弹性集群。这是必需的条件，以便使用控制台的主体可以选择 Amazon DocumentDB 弹性集群。
- `tag` – 允许主体获取账户中所有已标记的资源。
- `serverlessrepo`— 允许委托人创建 AWS CloudFormation 更改集。这是必需的条件，以便使用控制台的主体可以创建 Lambda 轮换函数。有关更多信息，请参阅 the section called “Secrets Manager 的使用方式 AWS CloudFormation”。
- `s3`— 允许委托人从由 AWS管理的 Amazon S3 存储桶中获取对象。此存储桶包含 Lambda 轮换函数模板。此权限是必需的，这样使用控制台的主体才能根据存储桶中的模板创建 Lambda 轮换函数。有关更多信息，请参阅 the section called “Secrets Manager 的使用方式 AWS CloudFormation”。

要查看该政策，请参阅 SecretsManagerReadWrite JSON 策略文档。

Secrets Manager 对 AWS 托管策略的更新

查看有关 Secrets Manager AWS 托管策略更新的详细信息。

更改	描述	日期	版本
SecretsManagerReadWrite – 对现有策略的更新	此策略已更新，允许描述访问 Amazon Redshift Serverless 的权限，以便控制台用户可在创建 Amazon Redshift 密钥时选择 Amazon Redshift	2024 年 3 月 12 日	v5

更改	描述	日期	版本
	Serverless 命名空间。		
SecretsManagerReadWrite – 对现有策略的更新	此策略已更新，允许描述访问 Amazon DocumentDB 弹性集群的权限，以便控制台用户可在创建 Amazon DocumentDB 密钥时选择弹性集群。	2023 年 9 月 12 日	v4
SecretsManagerReadWrite – 对现有策略的更新	此策略已更新，允许描述访问 Amazon Redshift 的权限，以便控制台用户可在创建 Amazon Redshift 密钥时选择 Amazon Redshift 集群。此更新还增加了新的权限，允许对存储 Lambda 轮换函数模板 AWS 的 Amazon S3 存储桶进行读取。	2020 年 6 月 24 日	v3
SecretsManagerReadWrite – 对现有策略的更新	此策略已更新，允许描述访问 Amazon RDS 集群的权限，以便控制台用户可在创建 Amazon RDS 密钥时选择集群。	2018 年 5 月 3 日	v2

更改	描述	日期	版本
SecretsManagerReadWrite：新策略	Secrets Manager 创建了一个策略，用于授予使用控制台所需的权限，并拥有对 Secrets Manager 的所有 read/write 访问权限。	2018 年 04 月 4 日	v1

确定谁有权访问你的 AWS Secrets Manager 秘密

预设情况下，IAM 身份无权限访问密钥。授权访问密钥时，Secrets Manager 会评估密钥基于资源的策略以及发送请求的 IAM 用户或角色的所有身份策略。为此，Secrets Manager 使用与 IAM 用户指南中确定请求是允许还是拒绝中描述过程类似的过程。

多个策略应用于请求时，Secrets Manager 会使用层次结构控制权限：

1. 如果任何策略中具有显式表达式的语句与请求操作和资源 `deny` 匹配：

   显式表达式 `deny` 覆盖其他所有内容并阻止操作。

2. 如果没有显式表达式 `deny`，但带有显式表达式 `allow` 与请求操作和资源匹配：

   显式表达式 `allow` 授予请求中的操作访问语句中资源的权限。

   如果身份和密钥位于两个不同的账户中，则该密钥的资源策略和附加到身份的策略`allow`中都必须有，否则会 AWS 拒绝请求。有关更多信息，请参阅 跨账户访问。

3. 如果没有带显式表达式 `allow` 与请求操作和资源相匹配：

   AWS 默认情况下拒绝请求，这称为隐式拒绝。

查看密钥基于资源的策略

- 请执行以下操作之一：
  - 打开 Secrets Manager 控制台，网址为https://console.aws.amazon.com/secretsmanager/。在您的密钥详细信息页面中，在资源权限部分，选择编辑权限。

- 使用 to AWS CLI call `get-resource-policy`或 AWS SDK 进行通话`GetResourcePolicy`。

确定哪些人可以通过基于身份的策略进行访问

- 使用 IAM policy simulator。参见用 IAM policy simulator 测试 IAM 策略

从其他账户访问 AWS Secrets Manager 密钥

一个账户中的用户可以访问另一个账户中的密钥（跨账户访问），您必须允许在资源策略和身份策略中进行访问。这与授予密钥所在账户中的身份访问权限不同。

跨账户权限仅对以下操作有效：

- CancelRotateSecret
- DeleteResourcePolicy
- DeleteSecret
- DescribeSecret
- GetRandomPassword
- GetResourcePolicy
- GetSecretValue
- ListSecretVersionIds
- PutResourcePolicy
- PutSecretValue
- RemoveRegionsFromReplication
- ReplicateSecretToRegions
- RestoreSecret
- RotateSecret
- StopReplicationToReplica
- TagResource
- UntagResource
- UpdateSecret

- UpdateSecretVersionStage
- ValidateResourcePolicy

您可以将`BlockPublicPolicy`参数与PutResourcePolicy操作配合使用，防止通过直接附加到您的密钥的资源策略授予公共访问权限，从而帮助保护您的资源。您也可以使用 IAM 访问分析器来验证跨账户访问权限。

您还必须允许身份使用密钥加密的 KMS 密钥。这是因为您不能使用 AWS 托管式密钥 (`aws/secretsmanager`) 进行跨账户访问。相反，您必须使用您创建的 KMS 密钥加密密钥，然后随附密钥策略。创建 KMS 密钥需支付费用。要更改密钥的加密密钥，请参阅 the section called “修改密钥”。

> **Important**
>
> 授予`secretsmanager:PutResourcePolicy`权限的基于资源的策略允许委托人（即使是其他账户中的委托人）修改基于资源的策略。此权限允许委托人升级现有权限，例如获得对机密的完全管理访问权限。我们建议您在策略中采用最低权限访问的原则。有关更多信息，请参阅基于资源的策略。

下列示例策略假定您在 Account1 中有密钥和加密密钥，而在 Account2 的身份希望有访问密钥值的权限。

步骤 1：将资源策略附加到 Account1 中的密钥

- 以下策略允许用户*Account2*访问*ApplicationRole*中的密钥*Account1*。要使用该策略，请参阅 the section called “基于资源的策略”。

  JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/ApplicationRole"
            },
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "*"
```

```
        }
    ]
}
```

步骤 2：将语句添加到 Account1 中 KMS 密钥的密钥策略中

- 以下密钥政策语句允许 *Account2* 中的 *ApplicationRole* 使用 *Account1* 中的 KMS 密钥来解密 *Account1* 中的密钥。要使用此语句，请将其添加到 KMS 密钥的密钥策略中。有关更多信息，请参阅更改密钥政策。

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

步骤 3：将身份策略附加到 Account2 中的身份

- 以下策略允许 *Account2* 中的 *ApplicationRole* 访问 *Account1* 中的密钥，并通过使用同样位于 *Account1* 中的加密密钥来解密密钥值。要使用该策略，请参阅 the section called “基于身份的策略”。您可以在 Secrets Manager 控制台的密钥详细信息页面的密钥 ARN 下方找到您的密钥 ARN。此外，您也可以调用 `describe-secret`。

  JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "arn:aws:secretsmanager:us-
east-1:123456789012:secret:secretName-AbCdEf"
```

```
        },
        {
            "Effect": "Allow",
            "Action": "kms:Decrypt",
            "Resource": "arn:aws:kms:us-
east-1:123456789012:key/EncryptionKey"
        }
    ]
}
```

从本地环境访问密钥

您可以使用 Ro AWS Identity and Access Management les Anywhere 在 IAM 中为在外部运行的服务器、容器和应用程序等工作负载获取临时安全证书 AWS。您的工作负载可以使用与 AWS 应用程序相同的 IAM 策略和 IAM 角色来访问 AWS 资源。借助 IAM Roles Anywhere，您可以使用 Secrets Manager 来存储和管理凭证，以便 AWS 中和应用程序服务器等本地设备上的资源能够访问。有关更多信息，请参阅 IAM Roles Anywhere 用户指南。

中的数据保护 AWS Secrets Manager

分 AWS 担责任模型适用于中的数据保护 AWS Secrets Manager。如本模型所述 AWS ，负责保护运行所有内容的全球基础架构 AWS Cloud。您负责维护对托管在此基础架构上的内容的控制。此内容包括您所使用的 AWS 服务 的安全配置和管理任务。有关数据隐私的更多信息，请参阅数据隐私常见问题。有关欧洲数据保护的信息，请参阅 AWS Security Blog 上的 AWS Shared Responsibility Model and GDPR 博客文章。

出于数据保护目的，我们建议您保护 AWS 账户 凭证并使用 AWS Identity and Access Management (IAM) 设置个人用户账户。这仅向每个用户授予履行其工作职责所需的权限。我们还建议您通过以下方式保护数据：

- 对每个账户使用 multi-factor authentication（MFA）。
- 使用 SSL/TLS 与资源通信。 AWS Secrets 在所有区域支持 TLS 1.2 和 1.3。Secrets Manager 还支持对传输层安全（TLS）网络加密协议使用混合后量子密钥交换选项（PQTLS）。
- 使用访问密钥 ID 和与 IAM 主体关联的秘密访问密钥来对以编程方式向 Secrets Manager 发出的请求进行签名。或者，您可以使用 AWS Security Token Service（AWS STS）生成临时安全凭证来对请求进行签名。
- 使用设置 API 和用户活动日志 AWS CloudTrail。请参阅 the section called “使用登录 AWS CloudTrail ”。

- 如果您在 AWS 通过命令行界面或 API 进行访问时需要经过 FIPS 140-2 验证的加密模块，请使用 FIPS 端点。请参阅 the section called “Secrets Manager 端点”。
- 如果你使用 AWS CLI 来访问 Secrets Manager，the section called “降低使用存储 AWS Secrets Manager 密钥 AWS CLI 的风险”.

静态加密

Secrets Manager 使用通过 AWS Key Management Service (AWS KMS) 进行加密来保护静态数据的机密性。 AWS KMS 提供许多服务使用的密钥存储和加密 AWS 服务。Secrets Manager 中的每个密钥都使用唯一的数据密钥加密。每个数据密钥都由一个 KMS 密钥保护。您可以选择对账户使用 Secrets Manager AWS 托管式密钥 的默认加密，也可以在 AWS KMS中创建自己的客户托管密钥。使用客户托管密钥可让您对 KMS 密钥活动进行更精细的授权控制。有关更多信息，请参阅 the section called “密钥加密和解密”。

传输中加密

Secrets Manager 为传输中的加密数据提供安全的私有终端节点。安全和私有端点 AWS 允许保护向 Secrets Manager 发出的 API 请求的完整性。 AWS 要求调用者使用 X.509 证书和/或 Secrets Manager 私有访问密钥对 API 调用进行签名。签名版本 4 签名流程 (Sigv4) 中阐述了此要求。

如果您使用 AWS Command Line Interface (AWS CLI) 或其中任何一个 AWS SDKs 来拨打电话 AWS，则需要配置要使用的访问密钥。然后，这些工具会自动使用访问密钥为您签署请求。请参阅 the section called “降低使用存储 AWS Secrets Manager 密钥 AWS CLI 的风险”。

互联网络流量隐私

AWS 在通过已知和专用网络路由路由路由流量时，提供了维护隐私的选项。

服务与本地客户端和应用之间的流量

您的私有网络和以下两种连接方式可供选择 AWS Secrets Manager：

- 一个 AWS Site-to-Site VPN 连接。有关更多信息，请参阅什么是 AWS 点对点 VPN？
- AWS Direct Connect 连接。有关更多信息，请参阅什么是 AWS Direct Connect？

同一区域内 AWS 资源之间的流量

如果你想在中保护 Secrets Manager 和 API 客户端之间的流量 AWS，请设置一个AWS PrivateLink以私密方式访问 Secrets Manager API 端点。

加密密钥管理

当 Secrets Manager 需要加密受保护机密数据的新版本时，Secrets Manager 会向发送请求，要求从 KMS 密钥生成新的数据密钥。 AWS KMS Secrets Manager 使用此数据密钥进行信封加密。Secrets Manager 将加密的数据密钥与加密的密钥储存在一起。当需要解密密密钥时，Secrets Manager 会要求 AWS KMS 解密数据密钥。然后，Secrets Manager 使用解密的数据密钥来解密加密的密钥。Secrets Manager 从不以未加密的形式存储数据密钥，并尽快从内存中删除密钥。有关更多信息，请参阅 the section called “密钥加密和解密”。

中的秘密加密和解密 AWS Secrets Manager

Secrets Manager 使用带有 AWS KMS 密钥和数据密钥的信封加密来保护每个密钥值。每当密钥中的密钥值更改时，Secrets Manager 都会从 AWS KMS 请求新数据密钥以为其提供保护。然后，用 KMS 密钥加密数据密钥，存储在密钥元数据中。要解密密密钥，Secrets Manager 首先使用中的 KMS 密钥对加密的数据密钥进行解密。 AWS KMS

Secrets Manager 不使用 KMS 直接对密钥值加密。相反，它使用 KMS 密钥来生成和加密 256 位高级加密标准 (AES) 对称数据密钥，并使用数据密钥对密钥值加密。Secrets Manager 使用纯文本数据密钥对外部的密钥值进行加密 AWS KMS，然后将其从内存中删除。它将数据密钥的加密副本存储在密钥的元数据中。

主题

- 选择一把 AWS KMS 钥匙
- 什么是加密？
- 加密和解密流程
- KMS 密钥的权限
- Secrets Manager 如何使用您的 KMS 密钥
- AWS 托管式密钥 (aws/secretsmanager) 的密钥策略
- Secrets Manager 加密上下文
- 监控 Secrets Manager 的互动 AWS KMS

选择一把 AWS KMS 钥匙

创建密钥时，您可以选择 AWS 账户 和区域中的任何对称加密客户托管密钥，也可以使用 for Secrets Manager (`aws/secretsmanager`)。 AWS 托管式密钥 如果你选择但它还不存在 AWS 托管式密钥 `aws/secretsmanager`，则 Secrets Manager 会创建它并将其与密钥关联。可对您账户中的每个密

钥使用相同的 KMS 密钥或不同的 KMS 密钥。您可能希望使用不同的 KMS 密钥为一组密钥设置密钥的自定义权限，或者如果您希望审计这些密钥的特定操作。Secrets Manager 仅支持对称加密 KMS 密钥。如果您在外部密钥存储中使用 KMS 密钥，则对 KMS 密钥的加密操作可能需要更长时间，而且可靠性和持久性会降低，因为请求必须在 AWS外部传输。

有关更改秘密的加密密钥的信息，请参阅 the section called “更改秘密的加密密钥”。

当您更改加密密钥时，Secrets Manager 会使用新密钥重新加密 `AWSCURRENT`、`AWSPENDING` 和 `AWSPREVIOUS` 版本。为了避免将您锁定在密钥之外，Secrets Manager 会使用以前的密钥加密所有现有版本。这意味着您可以使用以前的密钥或新密钥解密 `AWSCURRENT`、`AWSPENDING` 和 `AWSPREVIOUS` 版本。如果没有先前密钥的 `kms:Decrypt` 权限，则当您更改加密密钥时，Secrets Manager 无法解密密钥版本来重新加密它们。在这种情况下，现有版本不会被重新加密。

为了使只能通过新加密密钥解密 `AWSCURRENT`，请使用新密钥创建新版本的密钥。然后，为了能够解密 `AWSCURRENT` 密钥版本，您必须拥有新密钥的权限。

您可以拒绝访问权限 AWS 托管式密钥 `aws/secretsmanager`并要求使用客户托管密钥对机密进行加密。有关更多信息，请参阅 the section called “示例：拒绝使用特定 AWS KMS 密钥来加密机密”。

要查找与密钥关联的 KMS 密钥，请在控制台中查看密钥或致电ListSecrets或DescribeSecret。当密钥与 for Secrets Manager (`aws/secretsmanager`) 关联时，这些操作不会返回 KMS 密钥标识符。AWS 托管式密钥

什么是加密？

Secrets Manager 会加密密钥值，但不对以下项进行加密：

- 密钥名称和描述
- 轮换设置
- 与密钥关联的 KMS 密钥 ARN
- 任何附带的 AWS 标签

加密和解密流程

为了对密钥中的密钥值加密，Secrets Manager 使用以下过程。

1. Secrets Manager 使用密钥的 KMS 密钥的 ID 调用该 AWS KMS GenerateDataKey操作，并请求提供 256 位 AES 对称密钥。 AWS KMS 返回一个纯文本数据密钥和使用 KMS 密钥加密的数据密钥的副本。

2. Secrets Manager 使用纯文本数据密钥和高级加密标准 (AES) 算法对外部的密钥值进行加密。 AWS KMS然后，它将尽快从内存中删除明文密钥。
3. Secrets Manager 将加密的数据密钥存储在密钥的元数据中，使其可用于解密密钥值。但是，所有 Secrets Manager 都不会 APIs 返回加密的密钥或加密的数据密钥。

对已加密的密钥值解密：

1. Secrets Manager 调用 AWS KMS 解密操作并传入加密的数据密钥。
2. AWS KMS 使用密钥的 KMS 密钥来解密数据密钥。它将返回明文数据密钥。
3. Secrets Manager 使用该明文数据密钥来解密密钥值。然后，它会尽快从内存中删除数据密钥。

KMS 密钥的权限

当 Secrets Manager 在加密操作中使用 KMS 密钥时，它会代表创建或更新密钥值的用户执行操作。您可以在 IAM policy 或密钥政策中提供这些权限。以下 Secrets Manager 操作需要 AWS KMS 权限。

- CreateSecret
- GetSecretValue
- PutSecretValue
- UpdateSecret
- ReplicateSecretToRegions

要允许 KMS 密钥仅用于源自 Secrets Manager 的请求，可以在权限策略中使用带有`secretsmanager.<Region>.amazonaws.com`值的 k m ViaService s: 条件密钥。

您还可以在加密上下文中将密钥或值用作将 KMS 密钥用于加密操作的条件。例如，可在 IAM 或密钥策略文档中使用字符串条件运算符，或在授权中使用授权约束。KMS 密钥授权传播可能需要长达五分钟的时间。有关更多信息，请参阅 CreateGrant。

Secrets Manager 如何使用您的 KMS 密钥

Secrets Manager 使用你的 KMS 密钥调用以下 AWS KMS 操作。

GenerateDataKey

Secrets Manager 调用该 AWS KMS GenerateDataKey操作是为了响应以下 Secrets Manager 操作。

- CreateSecret— 如果新密钥包含密钥值，则 Secrets Manager 会请求新的数据密钥对其进行加密。
- PutSecretValue— Secrets Manager 请求新的数据密钥来加密指定的密钥值。
- ReplicateSecretToRegions— 要加密复制的密钥，Secrets Manager 在副本区域中请求一个 KMS 密钥的数据密钥。
- UpdateSecret— 如果您更改了密钥值或 KMS 密钥，Secrets Manager 会请求新的数据密钥来加密新的密钥值。

该RotateSecret操作不会调用`GenerateDataKey`，因为它不会更改密钥值。但是，如果 `RotateSecret` 调用更改了秘密值的 Lambda 轮换函数，则其调用 `PutSecretValue` 操作时将触发 `GenerateDataKey` 请求。

Decrypt

Secrets Manager 调用 Decrypt 操作来响应以下 Secrets Manager 操作。

- GetSecretValue和 BatchGetSecretValue— Secrets Manager 在将密钥值返回给调用者之前对其进行解密。要解密加密的密钥值，Secrets Manager 会调用 AWS KMS 解密操作来解密密密钥中的加密数据密钥。然后，它使用明文数据密钥来对已加密密钥值解密。对于批处理命令，Secrets Manager 可以重复使用解密后的密钥，因此并非所有调用都会产生 `Decrypt` 请求。
- PutSecretValue和 UpdateSecret— 大多数`PutSecretValue`和`UpdateSecret`请求不会触发`Decrypt`操作。但是，当 `PutSecretValue` 或 `UpdateSecret` 请求尝试更改现有密钥版本中的密钥值时，Secrets Manager 将对现有密钥值解密并将其与请求中的密钥值比较，以确认两者是否相同。此操作可确保 Secrets Manager 操作为幂等操作。要解密加密的密钥值，Secrets Manager 会调用 AWS KMS 解密操作来解密密密钥中的加密数据密钥。然后，它使用明文数据密钥来对已加密密钥值解密。
- ReplicateSecretToRegions— Secrets Manager 首先解密主区域中的密钥值，然后在副本区域中使用 KMS 密钥重新加密密钥值。

Encrypt

Secrets Manager 调用 Encrypt 操作来响应以下 Secrets Manager 操作：

- UpdateSecret— 如果您更改 KMS 密钥，Secrets Manager 会使用新密钥重新加密保护`AWSCURRENTAWSPREVIOUS`、和`AWSPENDING`机密版本的数据密钥。

DescribeKey

当你在 Secrets Manager 控制台中创建或编辑密钥时，Secrets Manager 调用该DescribeKey操作来确定是列出 KMS 密钥。

验证对 KMS 密钥的访问

当您建立或更改与密钥关联的 KMS 密钥时，Secrets Manager 将用指定的 KMS 调用 `GenerateDataKey` 和 `Decrypt` 操作。这些调用确认调用方是否有权将该 KMS 密钥用于这些操作。Secrets Manager 将放弃这些操作的结果；它不在任何加密操作中使用这些结果。

您可以识别这些验证调用，因为这些请求中 `SecretVersionId` 密钥加密上下文的值为 `RequestToValidateKeyAccess`。

> **Note**
>
> 过去，Secrets Manager 验证调用不包含加密上下文。您可能会在较早的 AWS CloudTrail 日志中发现没有加密上下文的呼叫。

AWS 托管式密钥 (`aws/secretsmanager`) 的密钥策略

仅当 Secrets Manager 代表用户发出请求时，for Secrets Manager (`aws/secretsmanager`) 的密钥策略才允许用户使用 KMS 密钥进行指定操作。 AWS 托管式密钥 密钥策略不允许任何用户直接使用 KMS 密钥。

此密钥策略与所有 AWS 托管式密钥 策略类似，均由该服务来建立。您无法更改密钥策略，但可以随时查看。有关详细信息，请参阅查看密钥策略。

密钥策略中的策略语句具有以下影响：

- 仅当 Secrets Manager 代表账户中的用户发出请求时，才允许这些用户使用 KMS 密钥执行加密操作。`kms:ViaService` 条件密钥可强制实施此限制。
- 允许该 AWS 账户创建 IAM 策略，允许用户查看 KMS 密钥属性和撤消授权。
- 尽管 Secrets Manager 不使用授权来获取 KMS 密钥的访问权限，但该策略还允许 Secrets Manager 代表用户创建 KMS 密钥授权，并允许帐户撤销任何允许 Secrets Manager 使用 KMS 密钥的授权。这些是政策文件的标准要素 AWS 托管式密钥.

以下是 Secrets Manager 示例 AWS 托管式密钥 的关键策略。

JSON

```
{
  "Id": "auto-secretsmanager-2",
```

```
 "Version": "2012-10-17",
 "Statement": [
   {
     "Sid": "Allow access through AWS Secrets Manager for all principals in the
account that are authorized to use AWS Secrets Manager",
     "Effect": "Allow",
     "Principal": {
       "AWS": [
         "*"
       ]
     },
     "Action": [
       "kms:Encrypt",
       "kms:Decrypt",
       "kms:ReEncrypt*",
       "kms:CreateGrant",
       "kms:DescribeKey"
     ],
     "Resource": "*",
     "Condition": {
       "StringEquals": {
         "kms:CallerAccount": "111122223333",
         "kms:ViaService": "secretsmanager.us-west-2.amazonaws.com"
       }
     }
   },
   {
     "Sid": "Allow access through AWS Secrets Manager for all principals in the
account that are authorized to use AWS Secrets Manager",
     "Effect": "Allow",
     "Principal": {
       "AWS": [
         "*"
       ]
     },
     "Action": "kms:GenerateDataKey*",
     "Resource": "*",
     "Condition": {
       "StringEquals": {
         "kms:CallerAccount": "111122223333"
       },
       "StringLike": {
         "kms:ViaService": "secretsmanager.us-west-2.amazonaws.com"
       }
```

```
      }
    },
    {
      "Sid": "Allow direct access to key metadata to the account",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::111122223333:root"
        ]
      },
      "Action": [
        "kms:Describe*",
        "kms:Get*",
        "kms:List*",
        "kms:RevokeGrant"
      ],
      "Resource": "*"
    }
  ]
}
```

Secrets Manager 加密上下文

加密上下文是一组包含任意非机密数据的密钥值对。当您在加密数据的请求中包含加密上下文时，会以加密 AWS KMS 方式将加密上下文绑定到加密数据。要解密数据，您必须传入相同的加密上下文。

在对的请求GenerateDataKey和解密请求中 AWS KMS，Secrets Manager 使用具有两个名称-值对的加密上下文，用于标识密钥及其版本，如以下示例所示。名称不会变化，但与其组合的加密上下文会因每个密钥值而异。

```
"encryptionContext": {
    "SecretARN": "arn:aws:secretsmanager:us-east-2:111122223333:secret:test-secret-
a1b2c3",
    "SecretVersionId": "EXAMPLE1-90ab-cdef-fedc-ba987SECRET1"
}
```

您可以使用加密上下文在审计记录和日志（例如和 Amazon CloudWatch Logs）中识别这些加密操作，并作为策略和授权中的授权条件。AWS CloudTrail

Secrets Manager 加密上下文包含两个名称–值对。

- SecretARN – 第一个名称–值对标识密钥。键是 `SecretARN`。该值是密钥的 Amazon Resource Name (ARN)。

  ```
  "SecretARN": "ARN of an Secrets Manager secret"
  ```

  例如，如果密钥的 ARN 是 `arn:aws:secretsmanager:us-east-2:111122223333:secret:test-secret-a1b2c3`，加密上下文将包括以下对。

  ```
  "SecretARN": "arn:aws:secretsmanager:us-east-2:111122223333:secret:test-secret-a1b2c3"
  ```

- SecretVersionId— 第二个名称-值对标识密钥的版本。键是 `SecretVersionId`。该值为版本 ID。

  ```
  "SecretVersionId": "<version-id>"
  ```

  例如，如果密钥的版本 ID 是 `EXAMPLE1-90ab-cdef-fedc-ba987SECRET1`，加密上下文将包括以下对。

  ```
  "SecretVersionId": "EXAMPLE1-90ab-cdef-fedc-ba987SECRET1"
  ```

当您建立或更改密钥的 KMS 密钥时，Secrets Manager 会向发送GenerateDataKey和解密请求，AWS KMS 以验证调用者是否有权使用 KMS 密钥进行这些操作。它将放弃响应，并且不对密钥值使用这些响应。

在这些验证请求中，`SecretARN` 的值是密钥的实际 ARN，但 `SecretVersionId` 值为 `RequestToValidateKeyAccess`，如以下加密上下文示例中所示。此特殊值可帮助您在日志和审核跟踪中标识验证请求。

```
"encryptionContext": {
    "SecretARN": "arn:aws:secretsmanager:us-east-2:111122223333:secret:test-secret-a1b2c3",
    "SecretVersionId": "RequestToValidateKeyAccess"
}
```

> **Note**
>
> 在过去，Secrets Manager 验证请求不包含加密上下文。您可能会在较早的 AWS CloudTrail 日志中发现没有加密上下文的呼叫。

监控 Secrets Manager 的互动 AWS KMS

您可以使用 AWS CloudTrail 和 Amazon CloudWatch Logs 来跟踪 Secrets Manager AWS KMS 代表您发送的请求。有关监测密钥使用的更多信息，请参阅 监控密钥。

GenerateDataKey

当您在密钥中创建或更改密钥值时，Secrets Manager 会向发送一个GenerateDataKey请求 AWS KMS ，指定该密钥的 KMS 密钥。

记录 `GenerateDataKey` 操作的事件与以下示例事件类似。该请求由 `secretsmanager.amazonaws.com` 调用。参数包括密钥的 KMS 密钥的 Amazon Resource Name (ARN)、需要 256 位密钥的密钥说明符以及标识密钥和版本的加密上下文。

```
{
    "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

当您获取或更改密钥的密钥值时，Secrets Manager 会向发送解密请求 AWS KMS 以解密加密的数据密钥。对于批处理命令，Secrets Manager 可以重复使用解密后的密钥，因此并非所有调用都会产生 `Decrypt` 请求。

记录 `Decrypt` 操作的事件与以下示例事件类似。用户是您 AWS 账户中访问表格的委托人。这些参数包括加密的表密钥（作为密文 blob）以及标识表和账户的加密上下文。 AWS AWS KMS 从密文中获取 KMS 密钥的 ID。

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AROAIGDTESTANDEXAMPLE:user01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/user01",
        "accountId": "111122223333",
```

```
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2018-05-31T23:36:09Z"
            }
        },
        "invokedBy": "secretsmanager.amazonaws.com"
    },
    "eventTime": "2018-05-31T23:36:09Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "secretsmanager.amazonaws.com",
    "userAgent": "secretsmanager.amazonaws.com",
    "requestParameters": {
        "encryptionContext": {
            "SecretARN": "arn:aws:secretsmanager:us-east-2:111122223333:secret:test-
secret-a1b2c3",
            "SecretVersionId": "EXAMPLE1-90ab-cdef-fedc-ba987SECRET1"
        }
    },
    "responseElements": null,
    "requestID": "658c6a08-652b-11e8-a6d4-ffee2046048a",
    "eventID": "f333ec5c-7fc1-46b1-b985-cbda13719611",
    "readOnly": true,
    "resources": [
        {
            "ARN": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "accountId": "111122223333",
            "type": "AWS::KMS::Key"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```


Encrypt

当您更改与密钥关联的 KMS 密钥时，Secrets Manager 会向发送加密请求，要求使用新密钥重新加`AWSPENDING`密`AWSCURRENT``AWSPREVIOUS`、和密钥版本。 AWS KMS 当您将密钥复制到另一个区域时，Secrets Manager 还会向 AWS KMS 发送 Encrypt 请求。

记录 Encrypt 操作的事件与以下示例事件类似。用户是您 AWS 账户中访问表格的委托人。


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AROAIGDTESTANDEXAMPLE:user01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/user01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "attributes": {
                "creationDate": "2023-06-09T18:11:34Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "secretsmanager.amazonaws.com"
    },
    "eventTime": "2023-06-09T18:11:34Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Encrypt",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "secretsmanager.amazonaws.com",
    "userAgent": "secretsmanager.amazonaws.com",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-east-2:111122223333:key/EXAMPLE1-f1c8-4dce-8777-
aa071ddefdcc",
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "encryptionContext": {
            "SecretARN": "arn:aws:secretsmanager:us-
east-2:111122223333:secret:ChangeKeyTest-5yKnKS",
            "SecretVersionId": "EXAMPLE1-5c55-4d7c-9277-1b79a5e8bc50"
        }
    },
    "responseElements": null,
    "requestID": "129bd54c-1975-4c00-9b03-f79f90e61d60",
    "eventID": "f7d9ff39-15ab-47d8-b94c-56586de4ab68",
    "readOnly": true,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:111122223333:key/EXAMPLE1-f1c8-4dce-8777-
aa071ddefdcc"
```

```
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

中的基础设施安全 AWS Secrets Manager

作为一项托管服务 AWS Secrets Manager ，受 AWS 全球网络安全的保护。有关 AWS 安全服务以及如何 AWS 保护基础设施的信息，请参阅AWS 云安全。要使用基础设施安全的最佳实践来设计您的 AWS 环境，请参阅 S AWS ecurity Pillar Well-Architected Fram ework 中的基础设施保护。

通过网络访问 Secrets Manager 是通过 APIs 使用 TLS AWS 发布的。Secret APIs s Manager 可以从任何网络位置调用。然而，Secrets Manager 支持基于资源的访问策略，其中可以包含基于源 IP 地址的限制。您还可以使用 Secrets Manager 资源策略来控制对来自特定虚拟私有云 (VPC) 终端节点或特定终端节点的机密的访问 VPCs。实际上，这可以将对给定密钥的网络访问与 AWS 网络中的特定 VPC 隔离开来。有关更多信息，请参阅 the section called “VPC 端点（AWS PrivateLink）”。

使用 AWS Secrets Manager VPC 终端节点

我们建议您在无法从私有网络访问的专用网络上运行尽可能多的基础设施。您可以通过创建接口 VPC 终端节点在 VPC 与 Secrets Manager 之间建立私有连接。接口端点由一项技术提供支持 AWS PrivateLink，该技术使您 APIs 无需互联网网关、NAT 设备、VPN 连接或 AWS Direct Connect 连接即可私密访问 Secrets Manager。您的 VPC 中的实例不需要公有 IP 地址即可与 Secrets Manager 通信 APIs。您的 VPC 与 Secrets Manager 之间的流量不会离开 AWS 网络。有关更多信息，请参阅《Amazon VPC 用户指南》中的接口 VPC 端点 (AWS PrivateLink)。

当 Secrets Manager 使用 Lambda 轮换函数轮换密钥（例如包含数据库凭证的密钥）时，Lambda 函数将同时向数据库和 Secrets Manager 发出请求。在使用控制台启用自动轮换后，Secrets Manager 将在与您的数据库相同的 VPC 中创建 Lambda 函数。建议您在同一 VPC 中创建 Secrets Manager 端点，以便从 Lambda 轮换函数向 Secrets Manager 发出的请求不会传出 Amazon 网络。

如果为端点启用私有 DNS，则可以使用其原定设置的 DNS 名称用作区域名，向 Secrets Manager 发送 API 请求，例如 `secretsmanager.us-east-1.amazonaws.com`。有关更多信息，请参阅《Amazon VPC 用户指南》中的通过接口端点访问服务。

您可以通过在权限策略中包含条件来确保对 Secrets Manager 的请求来自 VPC 访问。有关更多信息，请参阅 the section called “示例：权限和 VPCs”。

您可以通过 VPC 终端节点使用 AWS CloudTrail 日志来审核您对密钥的使用情况。

为 Secrets Manager 创建 VPC 端点

1. 请参阅《Amazon VPC 用户指南》中的 Creating an interface endpoint。使用服务名称:`com.amazonaws.region.secretsmanager`。
2. 要控制对端点的访问，请参阅 Control access to VPC endpoints using endpoint policies。
3. 要使用 IPv6 和双栈寻址，请参阅IPv4 和 IPv6 访问权限。

共享子网

您无法在与您共享的子网中创建、描述、修改或删除 VPC 端点。但是，您可以在与您共享的子网中使用 VPC 端点。有关 VPC 共享的信息，请参阅《Amazon Virtual Private Cloud 用户指南》中的与其他账户共享 VPC。

使用 IAM 策略控制 API 访问

如果您使用 IAM 策略 AWS 服务 根据 IP 地址控制访问权限，则可能需要更新策略以包括 IPv6 地址范围。本指南解释了 IPv4 和之间的区别， IPv6 并介绍了如何更新您的 IAM 策略以支持这两个协议。实施这些更改有助于您在提供支持的同时保持对 AWS 资源的安全访问 IPv6。

什么是 IPv6？

IPv6 是旨在最终取代的下一代 IP 标准 IPv4。之前的版本使用 32 位寻址方案来支持 43 亿台设备。IPv4 IPv6 而是使用 128 位寻址来支持大约 340 万亿亿亿美元（或第 128 功率的 2 倍）设备。

有关更多信息，请参阅 VPC IPv6 网页。

以下是 IPv6 地址的示例：

```
2001:cdba:0000:0000:0000:0000:3257:9652 # This is a full, unabbreviated IPv6 address.
2001:cdba:0:0:0:0:3257:9652             # The same address with leading zeros in each
 group omitted
2001:cdba::3257:965                     # A compressed version of the same address.
```

IAM 双栈（IPv4 和 IPv6）策略

您可以使用 IAM 策略来控制对 Secrets Manager 的访问权限， APIs 并阻止配置范围之外的 IP 地址访问 Secrets Manager APIs。

秘密经理。 {region} .amazonaws.com 适用于 Secrets Manager 的双栈终端节点同时支持和。 APIs IPv6 IPv4

如果您需要同时支持 IPv4 和 IPv6，请更新您的 IP 地址筛选策略以处理 IPv6 地址。否则，你可能无法通过连接到 Secrets Manager IPv6。

谁应该做出这种改变？

如果您将双寻址与包含以下内容的策略一起使用，则此更改会影响您`aws:sourceIp`。双寻址意味着网络同时支持 IPv4 和 IPv6。

如果您使用双地址，请更新当前使用 IPv4格式地址的 IAM 策略以包含 IPv6 格式地址。

谁不应该做出这种改变？

如果您只使用 IPv4 网络，则此更改不会影响您。

添加 IPv6 到 IAM 策略中

IAM 策略使用`aws:SourceIp`条件密钥来控制来自特定 IP 地址的访问。如果您的网络使用双寻址（IPv4 和 IPv6），请更新您的 IAM 策略以包含 IPv6 地址范围。

在策略`Condition`元素中，使用`IpAddress`和`NotIpAddress`运算符表示 IP 地址条件。不要使用字符串运算符，因为它们无法处理各种有效 IPv6 的地址格式。

这些示例使用`aws:SourceIp`. 对于 VPCs，请`aws:VpcSourceIp`改用。

以下是 IAM 用户指南中 AWS 基于源 IP 参考策略拒绝访问的内容。`Condition`元素`NotIpAddress`中的 to 列出了两个 IPv4 地址范围，`192.0.2.0/24`和`203.0.113.0/24`，这两个地址范围将被拒绝访问 API。

JSON

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

```
        "Action": "*",
        "Resource": "*",
        "Condition": {
            "NotIpAddress": {
                "aws:SourceIp": [
                    "192.0.2.0/24",
                    "203.0.113.0/24"
                ]
            },
            "Bool": {
                "aws:ViaAWSService": "false"
            }
        }
    }
}
```

要更新此政策，请将`Condition`元素更改为包括 IPv6地址范围`2001:DB8:1234:5678::/64`和`2001:cdba:3257:8593::/64`。

> **Note**
>
> 不要移除现有 IPv4 地址。它们是向后兼容所必需的。

```
"Condition": {
                "NotIpAddress": {
                    "aws:SourceIp": [
                        "192.0.2.0/24", <<DO NOT REMOVE existing IPv4 address>>
                        "203.0.113.0/24", <<DO NOT REMOVE existing IPv4 address>>
                        "2001:DB8:1234:5678::/64", <<New IPv6 IP address>>
                        "2001:cdba:3257:8593::/64" <<New IPv6 IP address>>
                    ]
                },
                "Bool": {
                    "aws:ViaAWSService": "false"
                }
            }
```

要更新 VPC 的此政策，请`aws:VpcSourceIp`改用`aws:SourceIp`：

```
"Condition": {
```

```
                "NotIpAddress": {
                    "aws:VpcSourceIp": [
                        "10.0.2.0/24", <<DO NOT REMOVE existing IPv4 address>>
                        "10.0.113.0/24", <<DO NOT REMOVE existing IPv4 address>>
                        "fc00:DB8:1234:5678::/64", <<New IPv6 IP address>>
                        "fc00:cdba:3257:8593::/64" <<New IPv6 IP address>>
                    ]
                },
                "Bool": {
                    "aws:ViaAWSService": "false"
                }
            }
```

验证您的客户端支持 IPv6

如果你使用密钥管理器。{region}.amazonaws.com 终端节点，请确认您可以连接到该终端节点。下面的步骤介绍了如何执行验证。

此示例使用 Linux 和 curl 版本 8.6.0，并使用已 IPv6 启用位于 amazonaws.com 端点的终端节点的AWS Secrets Manager 服务。

> **Note**
>
> 秘密经理。{region}.amazonaws.com 不同于典型的双栈命名惯例。有关 Secrets Manager 端点的完整列表，请参阅AWS Secrets Manager 端点。
> 将更改 AWS 区域 为您的服务所在的相同区域。在此示例中，我们使用的是美国东部（弗吉尼亚州北部）- `us-east-1` 端点。

1. 使用以下`dig`命令确定端点是否使用 IPv6 地址进行解析。

```
$ dig +short AAAA secretsmanager.us-east-1.amazonaws.com

> 2600:1f18:e2f:4e05:1a8a:948e:7c08:c1c3
```

2. 使用以下`curl`命令确定客户端网络能否建立 IPv6 连接。404 响应代码表示连接成功，而 0 响应代码表示连接失败。

```
$ curl --ipv6 -o /dev/null --silent -w "\nremote ip: %{remote_ip}\nresponse code:
 %{response_code}\n" https://secretsmanager.us-east-1.amazonaws.com
```

```
> remote ip: 2600:1f18:e2f:4e05:1a8a:948e:7c08:c1c3
> response code: 404
```

如果识别了远程 IP 但未识别响应码**0**，则使用成功地与端点建立了网络连接 IPv6。远程 IP 应该是一个 IPv6 地址，因为操作系统应选择对客户端有效的协议。

如果远程 IP 为空或响应码为`0`，则客户端网络或终端的网络路径 IPv4仅为-only。您可以使用以下 `curl` 命令验证此配置。

```
$ curl -o /dev/null --silent -w "\nremote ip: %{remote_ip}\nresponse code:
 %{response_code}\n" https://secretsmanager.us-east-1.amazonaws.com

> remote ip: 3.123.154.250
> response code: 404
```

灵活性 AWS Secrets Manager

AWS 围绕 AWS 区域 可用区构建全球基础架构。 AWS 区域 提供多个物理隔离和隔离的可用区，这些可用区可连接低延迟、高吞吐量和高度冗余的网络。利用可用区，您可以设计和操作在可用区之间无中断地自动实现失效转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区为您提供了更高的可用性、容错功能和可扩展性。

有关弹性和灾难恢复的更多信息，请参阅可靠性支柱 — Well-Architect AWS ed 框架。

有关 AWS 区域 和可用区的更多信息，请参阅AWS 全球基础设施。

后量子 TLS

Secrets Manager 支持对传输层安全 (TLS) 网络加密协议使用混合后量子密钥交换选项。当您连接到 Secrets Manager API 终端节点时，可以使用此 TLS 选项。我们在标准化后量子算法之前提供了此功能，因此您可以开始测试这些密钥交换协议对 Secrets Manager 调用产生的影响。这些混合后量子密钥交换功能是可选的，至少与我们目前使用的 TLS 加密一样安全，并且有可能会提供额外的安全优势。不过，与目前使用的传统密钥交换协议相比，它们会影响延迟和吞吐量性能。默认情况下，Secrets Manager Agent 使用后量子 ML-KEM 密钥交换作为优先级最高的密钥交换。

为了保护今天加密的数据，让这些数据在未来免受可能的攻击， AWS 正在积极参与密码社区，一起开发抗量子密码算法或后量子算法。我们已经在 Secrets Manager 端点中实施了混合后量子密钥交换密

码套件。这些混合密码套件将传统加密算法与后量子算法相结合，可确保 TLS 连接至少与传统密码套件一样强大。不过，由于混合密码套件的性能特征及带宽要求与传统密钥交换机制的性能特征及带宽要求有所不同，我们建议您针对 API 调用开展测试。

Secrets Manager 在除中国区域之外的所有区域都支持 PQTLS。

配置混合后量子 TLS

1. 将 AWS 公共运行时客户端添加到您的 Maven 依赖项中。我们建议您使用最新可用版本。例如，以下语句将添加版本 2.20.0。

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>aws-crt-client</artifactId>
  <version>2.20.0</version>
</dependency>
```

2. 将 AWS SDK for Java 2.x 添加到项目并对其进行初始化。在 HTTP 客户端上启用混合后量子密码套件。

```
SdkAsyncHttpClient awsCrtHttpClient = AwsCrtAsyncHttpClient.builder()
            .postQuantumTlsEnabled(true)
            .build();
```

3. 创建 Secrets Manager 异步客户端。

```
SecretsManagerAsyncClient SecretsManagerAsync = SecretsManagerAsyncClient.builder()
            .httpClient(awsCrtHttpClient)
            .build();
```

   现在，当您调用 Secrets Manager API 操作时，您的调用将通过混合后量子 TLS 传输到 Secrets Manager 端点。

有关使用混合后量子 TLS 的更多信息，请参阅：

- AWS SDK for Java 2.x 开发者指南和AWS SDK for Java 2.x 已发布的博客文章。
- s2n-tls 简介，新的开源 TLS 实施和使用 s2n-tls。
- 美国国家标准和技术研究院 (NIST) 的后量子密码学。
- 适合传输层安全 1.2 (TLS) 的混合后量子密钥封装方法 (PQ KEM)。

Secrets Manager 的后量子 TLS 在所有中都可用， AWS 区域 但中国除外。

故障排除 AWS Secrets Manager

使用此处的信息可帮助您诊断和修复您在使用 Secrets Manager 时可能遇到的问题。

有关轮换的问题，请参阅 the section called “轮换问题排查”。

主题

- “访问被拒绝”消息
- 对于临时安全凭证的“拒绝访问”
- 并非始终立即显示我所做的更改。
- 在创建秘密时收到“Cannot generate a data key with an asymmetric KMS key”（无法使用非对称 KMS 密钥生成数据密钥）
- AWS CLI 或 S AWS DK 操作无法从部分 ARN 中找到我的秘密
- 此密钥由 AWS 服务管理，您必须使用该服务对其进行更新。
- 使用时 Python 模块导入失败 Transform: AWS::SecretsManager-2024-09-16

“访问被拒绝”消息

当你向 Secrets Manager 进行诸如 GetSecretValue 或 CreateSecret 之类的 API 调用时，你必须具有 IAM 权限才能进行该调用。当您使用控制台时，控制台会代表您进行相同的 API 调用，因此您还必须具有 IAM 权限。管理员可以通过将 IAM 策略附加到您的 IAM 用户或您所属的组来授予权限。如果授予这些权限的政策声明包含任何条件，例如 time-of-day或 IP 地址限制，则您在发送请求时也必须满足这些要求。有关查看或修改适用于 IAM 用户、组或角色的策略的信息，请参阅《IAM 用户指南》中的使用策略。有关 Secrets Manager 所需权限的信息，请参阅 the section called “身份验证和访问控制”。

如果您在不使用 API 请求的情况下手动签名 AWS SDKs，请验证您是否正确签署了请求。

对于临时安全凭证的“拒绝访问”

请确认用于发出请求的 IAM 用户或角色具有正确的权限。临时安全凭证的权限来自于 IAM 用户或角色。这意味着，权限仅限于为 IAM 用户或角色授予的权限。有关临时安全凭证权限的确定方式的更多信息，请参阅 IAM 用户指南中的控制临时安全凭证的权限。

确认已正确对请求进行签名，并且请求格式正确无误。有关详细信息，请参阅所选软件开发工具包的工具包文档，或者 IAM 用户指南中的使用临时安全证书请求 AWS 资源访问权限。

验证您的临时安全凭证没有过期。有关更多信息，请参阅《IAM 用户指南》中的请求临时安全凭证。

有关 Secrets Manager 所需权限的信息，请参阅 the section called “身份验证和访问控制”。

并非始终立即显示我所做的更改。

Secrets Manager 使用名为最终一致性的分布式计算模型。你在 Secrets Manager（或其他 AWS 服务）中所做的任何更改都需要一段时间才能从所有可能的端点中看见。它在服务器与服务器之间、复制区域与复制区域之间，以及全球的区域与区域之间发送数据需要时间，这会造成一定的延迟。Secrets Manager 也使用缓存来提高性能，但在某些情况下，这可能会增加时间。在之前缓存的数据超时之前，更改可能不可见。

在设计全球应用程序时应考虑到这些可能的延迟。此外，确保应用程序可以按预期工作，即使在一个位置进行的更改不能立即在其他位置可见。

有关最终一致性如何影响其他一些 AWS 服务的更多信息，请参阅：

- Amazon Redshift 数据库开发人员指南中的管理数据一致性
- Amazon Simple Storage Service 用户指南中的 Amazon S3 数据一致性模型
- AWS 大数据博客中的 Ensuring Consistency When Using Amazon S3 and Amazon EMR for ETL Workflows
- 亚马逊 EC2 API 参考中的亚马逊 EC2最终一致性

在创建秘密时收到“Cannot generate a data key with an asymmetric KMS key”（无法使用非对称 KMS 密钥生成数据密钥）

Secrets Manager 使用与密钥关联的对称加密 KMS 密钥来为每个密钥值生成数据密钥。不能使用非对称 KMS 密钥。确认您使用的是对称加密 KMS 密钥，而不是非对称 KMS 密钥。有关说明，请参阅识别非对称 KMS 密钥。

AWS CLI 或 S AWS DK 操作无法从部分 ARN 中找到我的秘密

在许多情况下，Secrets Manager 可以从不完整的 ARN 中找到密钥，而无需完整的 ARN。但如果密钥名称以连字符后跟六个字符结尾，Secrets Manager 可能无法仅从一部分 ARN 中找到密钥。我们建议您改用完整的 ARN 或密钥名称。

更多详细信息

Secrets Manager 会在密钥名称末尾添加六个随机字符，以帮助确保密钥 ARN 的唯一性。如果删除了原始密钥，然后使用相同的名称创建了新密钥，则 ARNs 由于这些字符，这两个密钥会有所不同。有权访问旧密钥的用户不会自动获得对新密钥的访问权限，因为两 ARNs 者不同。

Secrets Manager 可为密钥构建 ARN，其中包含区域、账户、密钥名称，后跟连字符和六个字符，如下所示：

```
arn:aws:secretsmanager:us-east-2:111122223333:secret:SecretName-abcdef
```

如果密钥名称以连字符和六个字符结尾，当您仅使用一部分 ARN 时，在 Secrets Manager 看来您似乎指定了完整的 ARN。例如，您可能具有一个名为 `MySecret-abcdef` 的密钥，其 ARN 如下：

```
arn:aws:secretsmanager:us-east-2:111122223333:secret:MySecret-abcdef-nutBrk
```

当您调用以下操作（此操作仅使用一部分密钥 ARN）时，Secrets Manager 可能会找不到密钥。

```
$ aws secretsmanager describe-secret --secret-id arn:aws:secretsmanager:us-
east-2:111122223333:secret:MySecret-abcdef
```

此密钥由 AWS 服务管理，您必须使用该服务对其进行更新。

如果您在尝试修改密钥时遇到此消息，则只能使用消息中列出的管理服务来更新密钥。有关更多信息，请参阅 由其他服务管理的密钥。

要确定谁管理密钥，您可以查看密钥名称。由其他服务管理的密钥以该服务的 ID 作为前缀。或者，在中 AWS CLI，调用 d escribe-secret，然后查看该字段。`OwningService`

使用时 Python 模块导入失败 `Transform: AWS::SecretsManager-2024-09-16`

如果您使用的是 Transform:，`AWS::SecretsManager-2024-09-16`并且在轮换 Lambda 函数运行时遇到 Python 模块导入失败，则问题可能是由不兼容`Runtime`的值引起的。使用此转换版本， AWS CloudFormation 可以为您管理运行时版本、代码和共享目标文件。你不需要自己管理这些。

AWS Secrets Manager 配额

Secrets Manager 读取 APIs 的 TPS 配额很高，而调用频率较低的控制平面 APIs 的 TPS 配额较低。我们建议您避免以超过每 10 分钟一次的速率持续调用 `PutSecretValue` 或 `UpdateSecret`。如果调用 `PutSecretValue` 或 `UpdateSecret` 更新密钥值，Secrets Manager 将创建密钥的新版本。当版本超过 100 个时，Secrets Manager 会删除未标记的版本，但不会删除 24 小时内创建的版本。如果每 10 分钟更新一次密钥值，则创建的版本多于 Secrets Manager 删除的版本，将达到密钥版本的配额。

您可以在您的账户中运行多个区域，并且每个限额是特定于每个区域的。

当一个应用程序 AWS 账户 使用另一个账户拥有的密钥时，它被称为跨账户请求。对于跨账户请求，Secrets Manager 将限制发出请求的身份的账户，而不是拥有该秘密的账户。例如，如果账户 A 中的某一身份使用账户 B 中的某一秘密，则该秘密的使用将仅应用于账户 A 中的配额。

Secrets Manager 配额

名称	默认值	可调整	描述
DeleteResourcePolicy、 GetResourcePolicy PutResourcePolicy、和 ValidateResourcePolicy API 请求的总速率	每个受支持的区域：每秒 50 个	否	DeleteResourcePolicy、GetResourcePolicy、PutResourcePolicy和 ValidateResourcePolicy API 请求的每秒最大交易量。
PutSecretValue、、、 RemoveRegionsFromReplication Replicate SecretToRegion StopReplicationToReplica UpdateSecret、和 UpdateSecretVersionStage API 请求的合并速率	每个受支持的区域：每秒 50 个	否	PutSecretValue、、、RemoveRegionsFromReplication Replicate SecretToRegion StopReplicationToReplica UpdateSecret、和 UpdateSecretVersionStage API 请求的每秒最大交易量。

名称	默认值	可调整	描述
RestoreSecret API 请求的合并速率	每个受支持的区域：每秒 50 个	否	RestoreSecret API 请求每秒的最大交易量。
和 CancelRotateSecret API 请求 RotateSecret 的合并速率	每个受支持的区域：每秒 50 个	否	RotateSecret 和 CancelRotateSecret API 请求的每秒最大交易总和。
和 UntagResource API 请求 TagResource 的合并速率	每个受支持的区域：每秒 50 个	否	TagResource 和 UntagResource API 请求的每秒最大交易总和。
BatchGetSecretValue API 请求的速率	每个受支持的区域：每秒 100 个	否	BatchGetSecretValue API 请求每秒的最大交易量。
CreateSecret API 请求的速率	每个受支持的区域：每秒 50 个	否	CreateSecret API 请求每秒的最大交易量。
DeleteSecret API 请求的速率	每个受支持的区域：每秒 50 个	否	DeleteSecret API 请求每秒的最大交易量。
DescribeSecret API 请求的速率	每个支持的区域：每秒 40,000	否	DescribeSecret API 请求每秒的最大交易量。
GetRandomPassword API 请求的速率	每个受支持的区域：每秒 50 个	否	GetRandomPassword API 请求每秒的最大交易量。
GetSecretValue API 请求的速率	每个受支持的区域：每秒 1 万个	否	GetSecretValue API 请求每秒的最大交易量。
ListSecretVersionIds API 请求的速率	每个受支持的区域：每秒 50 个	否	ListSecretVersionIds API 请求每秒的最大交易量。

名称	默认值	可调整	描述
ListSecrets API 请求的速率	每个受支持的区域：每秒 100 个	否	ListSecrets API 请求每秒的最大交易量。
基于资源的策略长度	每个受支持的区域：2.048 万个	否	附加到秘密的基于资源的权限策略的最大字符数量。
秘密值大小	每个受支持的区域：6.5536 万字节	否	加密秘密值的最大大小。如果秘密值是字符串，则它是该秘密值中允许的字符数。
密文	每个受支持的区域：500 万个	否	此 AWS 账户每个 AWS 区域的最大密钥数量。
所有密钥版本中附上的标签数	每个受支持的区域：20 个	否	秘密的所有版本附加的暂存标签的最大数量。
每个密钥的版本数	每个受支持的区域：100 个	否	秘密的版本的最大数量。

将重试添加到您的应用程序

由于 AWS 客户端出现意外问题，您的客户可能会看到对 Secrets Manager 的调用失败。或者，调用可能由于 Secrets Manager 存在速率限制而失败。当您超过 API 请求配额时，Secrets Manager 会限制请求。它拒绝了一个原本有效的请求并返回 throttling 错误消息。对于两种失败，我们建议您在短暂的等待时间后重试呼叫。这被称为退避和重试策略。

如果您遇到以下错误，您可能需要将重试添加到您的应用程序代码：

瞬时错误和异常

- `RequestTimeout`
- `RequestTimeoutException`

- PriorRequestNotComplete
- ConnectionError
- HTTPClientError

服务端节流和限制错误与异常

- Throttling
- ThrottlingException
- ThrottledException
- RequestThrottledException
- TooManyRequestsException
- ProvisionedThroughputExceededException
- TransactionInProgressException
- RequestLimitExceeded
- BandwidthLimitExceeded
- LimitExceededException
- RequestThrottled
- SlowDown

有关重试、指数回退和抖动的详细信息以及示例代码，请参阅以下资源：

- 指数回退和抖动
- 超时、重试和回退并抖动
- 中出现错误重试和指数退缩。 AWS

文档历史记录

下表列出了自上一次发布以来对文档所做的重要更改 AWS Secrets Manager。要获得本文档的更新通知，您可以订阅 RSS 源。

变更	说明	日期
添加了对成本分配标签的支持	Secrets Manager 现在支持成本分配标签，允许客户按部门、团队或应用程序对成本进行分类和跟踪。有关更多信息，请参阅使用成本分配标签 AWS Secrets Manager。	2025 年 5 月 27 日
新 IPv6 增双栈支持	Secrets Manager 现支持双堆栈端点。有关更多信息 IPv4 ，请参阅并 IPv6 访问。	2024 年 12 月 20 日
Secrets Manager 改为 AWS 托管式策略	`SecretsManagerReadWrite` 托管式策略现在包括 `redshift-serverless` 权限。有关更多信息，请参阅AWS 托管式策略 AWS Secrets Manager	2024 年 3 月 12 日

早期更新

下表描述了在 2024 年 2 月前每次发布《 AWS Secrets Manager 用户指南》时进行的重要更改。

更改	描述	日期
正式发布	这是 Secrets Manager 的最初公开发布版。	2018 年 4 月 4 日

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。