aws

使用者指南

AWS Secrets Manager

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

AWS Secrets Manager: 使用者指南

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

Amazon 的商標和商業外觀不得用於任何非 Amazon 的產品或服務，也不能以任何可能造成客戶混淆、任何貶低或使 Amazon 名譽受損的方式使用 Amazon 的商標和商業外觀。所有其他非 Amazon 擁有的商標均為其各自擁有者的財產，這些擁有者可能附屬於 Amazon，或與 Amazon 有合作關係，亦或受到 Amazon 贊助。

Table of Contents

什麼是 Secrets Manager？ 1
Secrets Manager 入門 1
符合標準 2
定價 2
存取 Secrets Manager 3
Secrets Manager 主控台 3
命令列工具 3
AWS SDKs 3
HTTPS 查詢 API 4
Secrets Manager 端點 4
最佳實務 9
在 中存放登入資料和其他敏感資訊 AWS Secrets Manager 9
在程式碼中尋找未受保護的秘密 9
為您的秘密選擇加密金鑰 10
使用快取來擷取秘密 10
輪換 秘密 11
降低使用 CLI 的風險 11
限制對秘密的存取 11
BlockPublicPolicy 條件 11
請謹慎處理政策中的 IP 地址條件 12
限制具有 VPC 端點條件的請求 12
複寫秘密 13
監控秘密 13
在私有網路上執行您的基礎設施 13
教學課程 14
Amazon CodeGuru Reviewer 14
替換已寫死機密 14
步驟 1：建立機密 15
步驟 2：更新您的程式碼 17
步驟 3：更新機密 17
後續步驟 18
替換寫死資料庫憑證 18
步驟 1：建立機密 19
步驟 2：更新您的程式碼 20

步驟 3：輪換機密 20
後續步驟 22
交替使用者輪換 22
許可 23
先決條件 23
步驟 1：建立 Amazon RDS 資料庫使用者 26
步驟 2：為使用者的憑證建立秘密 28
步驟 3：測試輪換的秘密 29
步驟 4：清除資源 30
後續步驟 31
單一使用者輪換 31
許可 31
先決條件 32
步驟 1：建立 Amazon RDS 資料庫使用者 32
步驟 2：為資料庫使用者憑證建立秘密 33
步驟 3：測試輪換密碼 34
步驟 4：清除資源 34
後續步驟 34
建立秘密 36
AWS CLI 38
AWS 開發套件 39
秘密中的內容 39
中繼資料 40
秘密版本 40
機密的 JSON 結構 42
Amazon RDS 和 Aurora 登入資料 42
Amazon Redshift 登入資料 45
Amazon Redshift Serverless 登入資料 45
Amazon DocumentDB 登入資料 46
InfluxDB 秘密結構的 Amazon Timestream 46
Amazon ElastiCache 登入資料 46
Active Directory 登入資料 47
管理秘密 49
更新秘密值 49
AWS CLI 49
AWS 開發套件 50

使用 Secrets Manager 產生密碼 50
將秘密復原至先前的版本 50
變更秘密的加密金鑰 51
AWS CLI 52
修改秘密 53
AWS CLI 54
AWS 開發套件 54
查找秘密 54
搜尋篩選條件 55
AWS CLI 56
AWS 開發套件 56
刪除秘密 57
AWS CLI 58
AWS 開發套件 59
還原秘密 59
AWS CLI 59
AWS 開發套件 60
標籤 秘密 61
檢閱標籤基本概念 61
使用標記追蹤成本 62
了解標籤限制 62
在 主控台中標記秘密 63
AWS CLI 64
API 64
SDK 65
多區域複寫 66
AWS CLI 67
AWS 開發套件 68
將複本秘密提升為獨立秘密 68
AWS CLI 68
AWS 開發套件 69
防止複寫 69
針對複寫進行疑難排解 70
選取的區域中存在具有相同名稱的秘密。 71
KMS 金鑰上沒有可用的許可來完成複寫 71
KMS 金鑰已停用或找不到 KMS 金鑰 71

尚未啟用要進行複寫的區域 71
取得秘密 72
Java 72
Java 搭配用戶端快取 73
在秘密中使用登入資料進行 JDBC 連線 79
Java AWS 開發套件 88
Python 90
Python 搭配用戶端快取 90
Python AWS SDK 96
取得一批秘密值 98
.NET 99
具有用戶端快取的 .NET 99
適用於 .NET 的 SDK 106
Go 108
使用用戶端快取 109
Go AWS 開發套件 112
Rust 114
具有用戶端快取的 Rust 114
Rust 116
Amazon EKS 117
具有服務帳戶 IAM 角色的 ASCP (IRSA) 117
具有 Pod 身分的 ASCP 117
選擇正確的方法 118
安裝適用於 Amazon EKS 的 ASCP 118
將 ASCP 與 Amazon EKS 的 Pod Identity 整合 121
將 ASCP 與適用於 Amazon EKS 的 IRSA 整合 125
ASCP 範例 128
AWS Lambda 135
使用 Lambda 取得秘密 135
參數存放區整合 136
Secrets Manager 代理程式 136
Secrets Manager 代理程式的運作方式 136
了解 Secrets Manager 代理程式快取 137
建置 Secrets Manager 代理程式 138
安裝 Secrets Manager 代理程式 141
使用 Secrets Manager Agent 擷取秘密 146

了解 refreshNow 參數 148
組態選項 150
選用功能 151
日誌 151
安全考量 152
C++ 152
JavaScript 153
Kotlin 154
PHP 155
Ruby 156
AWS CLI 157
使用 取得批次中的一組秘密 AWS CLI 157
AWS 主控台 158
AWS Batch 159
AWS CloudFormation 159
GitHub 工作 160
先決條件 160
用量 161
環境變數命名 162
範例 163
AWS IoT Greengrass 165
參數存放區 165
輪換 秘密 167
受管輪換 167
依 Lambda 函數輪換 168
資料庫秘密的自動輪換 (主控台) 170
非資料庫秘密的自動輪換 （主控台） 173
自動輪換 (AWS CLI) 177
Lambda 函數輪換策略 180
Lambda 輪換函數 182
輪換函數範本 185
輪換的許可 192
AWS Lambda 輪換函數的網路存取 196
輪換疑難排解 197
輪換排程 210
輪換時段 211

Rate 運算式 211
Cron 表達式 211
立即輪換秘密 216
AWS CLI 216
尋找未輪換的秘密 216
取消自動輪換 217
由其他服務管理的秘密 218
使用秘密的服務 219
App Runner 221
AWS App2Container 221
AWS AppConfig 221
Amazon AppFlow 221
AWS AppSync 222
Amazon Athena 222
Amazon Aurora 222
AWS CodeBuild 222
Amazon Data Firehose 223
AWS DataSync 223
Amazon DataZone 223
AWS Direct Connect 223
AWS Directory Service 224
Amazon DocumentDB 224
AWS Elastic Beanstalk 224
Amazon Elastic Container Registry 224
Amazon Elastic Container Service 225
Amazon ElastiCache 225
AWS Elemental Live 226
AWS Elemental MediaConnect 226
AWS Elemental MediaConvert 226
AWS Elemental MediaLive 226
AWS Elemental MediaPackage 226
AWS Elemental MediaTailor 227
Amazon EMR 227
EC2 上的 EMR 227
EMR Serverless 227
Amazon EventBridge 227

Amazon FSx 228
AWS Glue DataBrew 228
AWS Glue Studio 228
AWS IoT SiteWise 228
Amazon Kendra 229
Amazon Kinesis Video Streams 229
AWS Launch Wizard 229
Amazon Lookout for Metrics 229
Amazon Managed Grafana 230
AWS Managed Services 230
Amazon Managed Streaming for Apache Kafka 230
Amazon Managed Workflows for Apache Airflow 230
AWS Marketplace 230
AWS Migration Hub 231
AWS Panorama 231
AWS 平行運算服務 231
AWS ParallelCluster 232
Amazon Q 232
Amazon OpenSearch Ingestion 232
AWS OpsWorks for Chef Automate 232
Amazon QuickSight 233
Amazon RDS 233
Amazon Redshift 233
Amazon Redshift 查詢編輯器第 2 版 234
Amazon SageMaker AI 234
AWS SCT 234
InfluxDB 的 Amazon Timestream 235
AWS Toolkit for JetBrains 235
AWS Transfer Family 235
AWS Wickr 236
AWS CloudFormation 237
建立秘密 237
JSON 238
YAML 238
使用具有自動輪換功能的 Amazon RDS 憑證建立金鑰 238
使用 Amazon Redshift 憑證建立秘密 239

使用 Amazon DocumentDB 憑證建立秘密 ... 239
JSON ... 239
YAML ... 244
Secrets Manager 如何使用 AWS CloudFormation ... 246
AWS CDK ... 247
監控秘密 ... 248
使用 記錄 AWS CloudTrail ... 248
AWS CLI ... 249
CloudTrail 事件 ... 249
使用 CloudWatch 監控 ... 254
CloudWatch 警示 ... 254
使用 EventBridge 比對 Secrets Manager 事件 ... 255
比對指定機密的所有變更 ... 255
機密值輪換時比對事件 ... 256
監控排定刪除的秘密 ... 256
步驟 1：設定 CloudTrail 日誌檔案交付至 CloudWatch Logs ... 256
步驟 2：建立 CloudWatch 警示 ... 257
步驟 3：測試 CloudWatch 警示 ... 258
監控秘密的合規性 ... 258
監控 Secrets Manager 成本 ... 259
使用 GuardDuty 偵測威脅 ... 259
法規遵循驗證 ... 261
合規標準 ... 261
安全 ... 263
降低使用 AWS CLI 存放 AWS Secrets Manager 秘密的風險 ... 263
身分驗證與存取控制 ... 265
許可參考 ... 266
Secrets Manager 管理員許可 ... 266
存取秘密的許可 ... 266
Lambda 輪換函數的許可 ... 267
用於加密金鑰的許可 ... 267
複寫的許可 ... 267
身分型政策 ... 267
資源型政策 ... 274
使用標籤控制對秘密的存取 ... 280
AWS 受管政策 ... 282

判斷誰有存取 秘密的許可 285
跨帳戶存取權 286
內部部署存取 289
Secrets Manager 中的資料保護 289
靜態加密 290
傳輸中加密 290
網際網路流量隱私權 290
加密金鑰管理 290
秘密加密和解密 291
選擇 AWS KMS 金鑰 291
會加密哪些資料？ 292
加密和解密程序 292
KMS 金鑰的許可 293
Secrets Manager 如何使用您的 KMS 金鑰 293
AWS 受管金鑰 (aws/secretsmanager) 的金鑰政策 295
Secrets Manager 加密內容 297
監控 Secrets Manager 與 的互動 AWS KMS 298
基礎架構安全 303
VPC 端點 (AWS PrivateLink) 303
共用子網路 304
IPv4 和 IPv6 存取 304
什麼是 IPv6？ 304
使用雙堆疊政策 304
將 IPv6 新增至政策 305
驗證您的用戶端支援 IPv6 307
恢復能力 308
後量子 TLS 308
疑難排解 310
「拒絕存取」訊息 310
暫時安全憑證的「存取遭拒」 310
我所做的變更不一定都會立即顯示。 311
在建立秘密時收到「無法使用非對稱 KMS 金鑰產生資料金鑰」的訊息 311
AWS CLI 或 AWS SDK 操作無法從部分 ARN 找到我的秘密 311
此秘密由 AWS 服務管理，您必須使用該服務來更新它。 312
使用 時，Python 模組匯入失敗 Transform: AWS::SecretsManager-2024-09-16 312
配額 313

Secrets Manager 配額 313
將重試新增至應用程式 315
文件歷史紀錄 317
舊版更新 317
..... cccxviii

什麼是 AWS Secrets Manager？

AWS Secrets Manager 可協助您管理、擷取和輪換資料庫登入資料、應用程式登入資料、OAuth 權杖、API 金鑰，以及在其整個生命週期中的其他秘密。許多 AWS 服務會在 Secrets Manager 中存放和使用秘密。

Secrets Manager 可協助您改善安全狀態，因為應用程式原始程式碼中不再需要硬式編碼憑證。將憑證儲存在 Secrets Manager 中有助於避免任何可以檢查您的應用程式或元件的人可能造成的危害。您可以將硬式編碼憑證取代為對 Secrets Manager 服務的執行期呼叫，以在需要時動態擷取憑證。

使用 Secrets Manager，您可以為機密設定自動輪換排程。這可讓您以短期秘密取代長期秘密，進而大幅降低洩漏風險。由於憑證不再與應用程式一起存放，因此輪換憑證不再需要更新您的應用程式並將變更部署至應用程式用戶端。

對於您的組織中可能具有的其他類型機密：

- AWS 登入資料 – 我們建議使用 AWS Identity and Access Management。
- 加密金鑰 – 建議使用 AWS Key Management Service。
- SSH 金鑰 – 建議使用 Amazon EC2 Instance Connect。
- 私有金鑰和憑證 – 建議使用 AWS Certificate Manager。

Secrets Manager 入門

如果您是 Secrets Manager 的新手，請從下列其中一個教學課程開始：

- the section called “替換已寫死機密 ”
- the section called “替換寫死資料庫憑證 ”
- the section called “交替使用者輪換”
- the section called “單一使用者輪換”

您可以使用機密執行的其他任務：

- 管理秘密
- 控制對機密的存取
- 取得秘密

- 輪換 秘密
- 監控秘密
- 監控秘密的合規性
- 在 中建立秘密 AWS CloudFormation

符合標準

AWS Secrets Manager 已針對多個標準進行稽核，並且可以在您需要取得合規認證時成為解決方案的一部分。如需詳細資訊，請參閱法規遵循驗證。

定價

當您使用 Secrets Manager 時，將按使用量付費，而沒有最低收費或設定費用。不會針對已標示為刪除的機密收取任何費用。如需目前完整定價清單，請參閱 AWS Secrets Manager 定價。若要監控您的成本，請參閱 the section called “監控 Secrets Manager 成本”。

您可以使用 Secrets Manager AWS 受管金鑰 `aws/secretsmanager`建立的 免費加密秘密。如果您建立自己的 KMS 金鑰來加密秘密， 會以目前的 AWS KMS 費率向您 AWS 收費。如需詳細資訊，請參閱AWS Key Management Service 定價。

當您開啟自動輪換 (受管輪換除外） 時，Secrets Manager 會使用 AWS Lambda 函數來輪換秘密，而且會以目前的 Lambda 速率向您收取輪換函數的費用。如需詳細資訊，請參閱AWS Lambda 定價。

如果您在 帳戶 AWS CloudTrail 上啟用 ，您可以取得 Secrets Manager 傳送的 API 呼叫日誌。Secrets Manager 會將所有事件記錄為管理事件。 免費 AWS CloudTrail 存放所有管理事件的第一個副本。不過，對於用來儲存日誌的 Amazon S3 及 Amazon SNS (如果您啟用通知)，您可能需要付費。此外，如果您設定額外的線索，則管理事件的額外副本可能會產生成本。如需詳細資訊，請參閱 AWS CloudTrail 定價。

您可以使用 Secrets Manager 中的成本分配標籤來追蹤和分類與特定秘密或專案相關聯的費用。如需詳細資訊，請參閱本指南the section called “標籤 秘密”中的 和 AWS Billing 《 使用者指南》中的使用 AWS 成本分配標籤。

存取 AWS Secrets Manager

您可透過以下方式來使用 Secrets Manager：

- Secrets Manager 主控台
- 命令列工具
- AWS SDKs
- HTTPS 查詢 API
- AWS Secrets Manager 端點

Secrets Manager 主控台

您可以使用瀏覽器型 Secrets Manager 主控台管理您的秘密，並使用主控台執行幾乎任何與您秘密相關的任務。

命令列工具

AWS 命令列工具可讓您在系統命令列發出命令，以執行 Secrets Manager 和其他 AWS 任務。與使用主控台相較，此方法更快速也更便利。如果您想要建置指令碼來執行 AWS 任務，命令列工具會很有用。

在命令 shell 中輸入命令時，存在命令歷史記錄被存取或公用程式存取命令參數的風險。請參閱 the section called “降低使用 AWS CLI 存放 AWS Secrets Manager 秘密的風險”。

命令列工具會自動為 AWS 區域中的服務使用預設端點。您可以為 API 請求指定其他端點。請參閱 the section called “Secrets Manager 端點”。

AWS 提供兩組命令列工具：

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

AWS SDKs

AWS SDKs 由適用於各種程式設計語言和平台的程式庫和範本程式碼組成。開發套件的工作諸如以密碼演算法簽署請求、管理錯誤以及自動重試請求。若要下載並安裝任何 SDK，請參閱適用於 Amazon Web Services 的工具。

AWS SDKs 會自動為 AWS 區域中的服務使用預設端點。您可以為 API 請求指定其他端點。請參閱 the section called “Secrets Manager 端點”。

有關 SDK 文件，請參閱：

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

HTTPS 查詢 API

HTTPS 查詢 API 可讓您以程式設計方式存取 Secrets Manager 和 AWS。HTTPS 查詢 API 可讓您直接將 HTTPS 請求發給該服務。

雖然您可以直接呼叫 Secrets Manager HTTPS 查詢 API，但建議您改為使用其中一個 SDK。SDK 可執行您原本必須手動執行的許多實用任務。例如，SDK 會自動簽署您的請求，並將回應轉換為語法上適合您語言的結構。

若要對 Secrets Manager 進行 HTTPS 呼叫，請連接至 ???。

AWS Secrets Manager 端點

若要以程式設計方式連接至 Secrets Manager，請使用端點，也就是服務的進入點 URL。Secrets Manager 端點是雙堆疊端點，這表示它們同時支援 IPv4 和 IPv6。

Secrets Manager 在部分區域提供支援聯邦資訊處理標準 (FIPS) 140-2 的端點。

Secrets Manager 支援 TLS 1.2 和 1.3。Secrets Manager 支援所有區域中的 PQTLS，唯中國地區除外。

> **Note**
>
> Python AWS SDK 和 AWS CLI 會嘗試依序呼叫 IPv6，然後呼叫 IPv4，因此，如果您未啟用 IPv6，可能需要一些時間才能呼叫逾時，並使用 IPv4 重試。若要解決這個問題，您可以完全停用 IPv6 或移轉至 IPv6。

以下是 Secrets Manager 的服務端點。請注意，命名與典型的雙堆疊命名慣例不同。如需在 Secrets Manager 中使用雙堆疊定址的詳細資訊，請參閱 IPv4 和 IPv6 存取。

區域名稱	區域	端點	通訊協定
美國東部 (俄亥俄)	us-east-2	secretsmanager.us-east-2.amazonaws.com	HTTPS
		secretsmanager-fips.us-east-2.amazonaws.com	HTTPS
美國東部 (維吉尼亞北部)	us-east-1	secretsmanager.us-east-1.amazonaws.com	HTTPS
		secretsmanager-fips.us-east-1.amazonaws.com	HTTPS
美國西部 (加利佛尼亞北部)	us-west-1	secretsmanager.us-west-1.amazonaws.com	HTTPS
		secretsmanager-fips.us-west-1.amazonaws.com	HTTPS
美國西部 (奧勒岡)	us-west-2	secretsmanager.us-west-2.amazonaws.com	HTTPS
		secretsmanager-fips.us-west-2.amazonaws.com	HTTPS
Africa (Cape Town)	af-south-1	secretsmanager.af-south-1.amazonaws.com	HTTPS

區域名稱	區域	端點	通訊協定
亞太區域 (香港)	ap-east-1	secretsmanager.ap-east-1.amazonaws.com	HTTPS
亞太區域 (海德拉巴)	ap-south-2	secretsmanager.ap-south-2.amazonaws.com	HTTPS
亞太區域 (雅加達)	ap-southeast-3	secretsmanager.ap-southeast-3.amazonaws.com	HTTPS
亞太地區 (馬來西亞)	ap-southeast-5	secretsmanager.ap-southeast-5.amazonaws.com	HTTPS
亞太區域 (墨爾本)	ap-southeast-4	secretsmanager.ap-southeast-4.amazonaws.com	HTTPS
亞太區域 (孟買)	ap-south-1	secretsmanager.ap-south-1.amazonaws.com	HTTPS
亞太區域 (大阪)	ap-northeast-3	secretsmanager.ap-northeast-3.amazonaws.com	HTTPS
亞太區域 (首爾)	ap-northeast-2	secretsmanager.ap-northeast-2.amazonaws.com	HTTPS
亞太區域 (新加坡)	ap-southeast-1	secretsmanager.ap-southeast-1.amazonaws.com	HTTPS
亞太區域 (雪梨)	ap-southeast-2	secretsmanager.ap-southeast-2.amazonaws.com	HTTPS

區域名稱	區域	端點	通訊協定
亞太區域（台北）	ap-east-2	secretsmanager.ap-east-2.amazonaws.com	HTTPS
亞太區域（泰國）	ap-southeast-7	secretsmanager.ap-southeast-7.amazonaws.com	HTTPS
亞太區域 (東京)	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 (卡加利)	ca-west-1	secretsmanager.ca-west-1.amazonaws.com secretsmanager-fips.ca-west-1.amazonaws.com	HTTPS HTTPS
歐洲 (法蘭克福)	eu-central-1	secretsmanager.eu-central-1.amazonaws.com	HTTPS
歐洲 (愛爾蘭)	eu-west-1	secretsmanager.eu-west-1.amazonaws.com	HTTPS
歐洲 (倫敦)	eu-west-2	secretsmanager.eu-west-2.amazonaws.com	HTTPS
歐洲 (米蘭)	eu-south-1	secretsmanager.eu-south-1.amazonaws.com	HTTPS
歐洲 (巴黎)	eu-west-3	secretsmanager.eu-west-3.amazonaws.com	HTTPS
歐洲 (西班牙)	eu-south-2	secretsmanager.eu-south-2.amazonaws.com	HTTPS

區域名稱	區域	端點	通訊協定
歐洲 (斯德哥爾摩)	eu-north-1	secretsmanager.eu-north-1.amazonaws.com	HTTPS
歐洲 (蘇黎世)	eu-central-2	secretsmanager.eu-central-2.amazonaws.com	HTTPS
以色列 (特拉維夫)	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
中東 (巴林)	me-south-1	secretsmanager.me-south-1.amazonaws.com	HTTPS
中東 (阿拉伯聯合大公國)	me-central-1	secretsmanager.me-central-1.amazonaws.com	HTTPS
南美洲 (聖保羅)	sa-east-1	secretsmanager.sa-east-1.amazonaws.com	HTTPS
AWS GovCloud（美國東部）	us-gov-east-1	secretsmanager.us-gov-east-1.amazonaws.com secretsmanager-fips.us-gov-east-1.amazonaws.com	HTTPS HTTPS
AWS GovCloud（美國西部）	us-gov-west-1	secretsmanager.us-gov-west-1.amazonaws.com secretsmanager-fips.us-gov-west-1.amazonaws.com	HTTPS HTTPS

AWS Secrets Manager 最佳實務

Secrets Manager 提供許多安全功能，供您在開發和實作自己的安全政策時考慮。以下最佳實務為一般準則，並不代表完整的安全解決方案。這些最佳實務可能不適用或無法滿足您的環境需求，因此請將其視為實用建議就好，而不要當作是指示。

請考慮下列儲存和管理秘密的最佳實務：

- 在 中存放登入資料和其他敏感資訊 AWS Secrets Manager
- 在程式碼中尋找未受保護的秘密
- 為您的秘密選擇加密金鑰
- 使用快取來擷取秘密
- 輪換 秘密
- 降低使用 CLI 的風險
- 限制對秘密的存取
- 複寫秘密
- 監控秘密
- 在私有網路上執行您的基礎設施

在 中存放登入資料和其他敏感資訊 AWS Secrets Manager

Secrets Manager 可協助改善您的安全狀態和合規，並降低未經授權存取敏感資訊的風險。Secrets Manager 會使用您擁有並存放在 AWS Key Management Service () 中的加密金鑰來加密靜態秘密AWS KMS。當您擷取秘密時，Secrets Manager 會解密秘密，並透過 TLS 將其安全地傳輸至您的本機環境。如需詳細資訊，請參閱建立秘密。

在程式碼中尋找未受保護的秘密

CodeGuru Reviewer 與 Secrets Manager 整合，以使用在程式碼中尋找未受保護秘密的秘密偵測器。秘密偵測器會搜尋硬式編碼密碼、資料庫連線字串、使用者名稱等。如需詳細資訊，請參閱the section called “Amazon CodeGuru Reviewer”。

Amazon Q 可以掃描您的程式碼庫是否有安全漏洞和程式碼品質問題，以改善整個開發週期的應用程式狀態。如需詳細資訊，請參閱《Amazon Q 開發人員使用者指南》中的使用 Amazon Q 掃描程式碼。

為您的秘密選擇加密金鑰

在大多數情況下，我們建議使用 aws/secretsmanager AWS 受管金鑰來加密秘密。使用它無需付費。

若要能夠從另一個帳戶存取秘密，或將金鑰政策套用至加密金鑰，請使用客戶受管金鑰來加密秘密。

- 在金鑰政策中，將值指派給secretsmanager.<region>.amazonawskms:ViaService條件金鑰。這會將金鑰的使用限制為僅來自 Secrets Manager 的請求。
- 若要進一步限制金鑰的使用僅限於具有正確內容的 Secrets Manager 請求，請在 Secrets Manager 加密內容中使用金鑰或值，作為使用 KMS 金鑰的條件，方法是建立：
  - IAM 政策或金鑰政策中的字串條件運算子
  - 授權中的授予限制條件

如需詳細資訊，請參閱the section called “秘密加密和解密”。

使用快取來擷取秘密

若要以最有效率的方式使用您的秘密，建議您使用下列其中一個支援的 Secrets Manager 快取元件來快取秘密，並只在需要時才更新它們：

- Java 搭配用戶端快取
- Python 搭配用戶端快取
- 具有用戶端快取的 .NET
- 使用用戶端快取
- 具有用戶端快取的 Rust
- AWS 參數和秘密 Lambda 延伸模組
- the section called “Amazon EKS”
- 使用 the section called “Secrets Manager 代理程式” 將 Secrets Manager 在 Amazon Elastic Container Service AWS Lambda、Amazon Elastic Kubernetes Service 和 Amazon Elastic Compute Cloud 等環境中的秘密使用標準化。

輪換 秘密

如果您長時間未變更您的秘密，秘密就更有可能遭到盜用。使用 Secrets Manager，您可以設定自動輪換的頻率為每四小時一次。Secrets Manager 提供兩種輪換策略： 單一使用者和 交替使用者。如需詳細資訊，請參閱輪換 秘密。

降低使用 CLI 的風險

當您使用 AWS CLI 叫用 AWS 操作時，您可以在命令 shell 中輸入這些命令。大多數命令 shell 提供的功能可能會危害您的秘密，例如記錄和查看上次輸入的命令的能力。在您使用 AWS CLI 輸入敏感資訊之前，請務必輸入 the section called “降低使用 AWS CLI 存放 AWS Secrets Manager 秘密的風險”。

限制對秘密的存取

在控制秘密存取的 IAM 政策陳述式中，使用最低權限存取的原則。您可以使用 IAM 角色和政策、資源政策和屬性型存取控制 (ABAC)。如需詳細資訊，請參閱the section called “身分驗證與存取控制”。

主題

- 封鎖對秘密的廣泛存取
- 請謹慎處理政策中的 IP 地址條件
- 限制具有 VPC 端點條件的請求

封鎖對秘密的廣泛存取

在允許 `PutResourcePolicy`操作的身分政策中，我們建議您使用 `BlockPublicPolicy: true`。此條件意味著只有在政策不允許廣泛存取時，使用者才能將資源政策連接到秘密。

Secrets Manager 使用 Zelkova 自動推理來分析廣泛存取的資源政策。如需 Zelkova 的詳細資訊，請參閱 安全部落格上的 AWS 如何使用自動化推理來協助您大規模實現 AWS 安全。

下列範例示範如何使用 `BlockPublicPolicy`。

JSON

```
{
```

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

請謹慎處理政策中的 IP 地址條件

在允許或拒絕對 Secrets Manager 之存取權的同一政策陳述式中，指定 IP 地址條件運算子或 `aws:SourceIp` 條件索引鍵時，請謹慎小心。例如，如果您將限制 AWS 動作從公司網路 IP 地址範圍請求的政策連接到秘密，則以 IAM 使用者身分從公司網路叫用請求的請求會如預期運作。不過，如果您讓其他 服務代表您存取秘密，例如當您使用 Lambda 函數啟用輪換時，該函數會從 AWS內部地址空間呼叫 Secrets Manager 操作。使用 IP 地址篩選條件時受到政策影響的請求會失敗。

此外，當請求來自 Amazon VPC 端點時，`aws:sourceIP` 條件金鑰較無效。若要將請求限制為特定 VPC 端點，請使用 the section called “限制具有 VPC 端點條件的請求”。

限制具有 VPC 端點條件的請求

若要允許或拒絕對特定 VPC 或 VPC 端點所發出之請求的存取權，請使用 `aws:SourceVpc` 來限制對指定 VPC 所發出之請求的存取權，或使用 `aws:SourceVpce` 來限制對指定 VPC 端點所發出之請求的存取權。請參閱 the section called “範例：許可和 VPC”。

- `aws:SourceVpc` 會限制對指定 VPC 所發出之請求的存取權。
- `aws:SourceVpce` 會限制對指定 VPC 端點所發出之請求的存取權。

如果您在允許或拒絕對 Secrets Manager 秘密之存取權的資源政策陳述式中使用這些條件金鑰，則您可能會不小心拒絕對使用 Secrets Manager 之服務的存取權，而這些服務會代表您存取秘密。只有部分 AWS 服務可以使用 VPC 中的端點執行。如果您將秘密的請求限制為來自 VPC 或 VPC 端點，則從未設定的服務呼叫 Secrets Manager 可能會失敗。

請參閱 the section called “VPC 端點 (AWS PrivateLink)”。

複寫秘密

Secrets Manager 可以自動將您的秘密複寫到多個 AWS 區域，以符合您的復原能力或災難復原需求。如需詳細資訊，請參閱多區域複寫。

監控秘密

Secrets Manager 可讓您透過與 AWS 記錄、監控和通知服務的整合來稽核和監控秘密。如需詳細資訊，請參閱：

- the section called “使用 記錄 AWS CloudTrail ”
- the section called “使用 CloudWatch 監控”
- the section called “監控秘密的合規性”
- the section called “監控 Secrets Manager 成本”
- the section called “使用 GuardDuty 偵測威脅”

在私有網路上執行您的基礎設施

我們建議您在無法從公有網際網路存取的私有網路上儘可能執行基礎設施。您可以建立介面 VPC 端點，以在您的 VPC 與 Secrets Manager 之間建立私有連線。如需詳細資訊，請參閱the section called “VPC 端點 (AWS PrivateLink)”。

AWS Secrets Manager 教學課程

主題

- 透過 Amazon CodeGuru Reviewer 在您的程式碼中尋找未受保護的機密
- 將硬式編碼秘密移至 AWS Secrets Manager
- 將硬式編碼資料庫登入資料移至 AWS Secrets Manager
- 為 AWS Secrets Manager設定交替使用者輪換
- 為 AWS Secrets Manager設定單一使用者輪換

透過 Amazon CodeGuru Reviewer 在您的程式碼中尋找未受保護的機密

Amazon CodeGuru Reviewer 是一項服務，可使用程式分析和機器學習來偵測開發人員難以找到的潛在缺陷，並提供改善 Java 和 Python 程式碼的建議。CodeGuru Reviewer 與機密管理員整合，在您的程式碼中找到未受保護的機密。如需該服務可以找到的機密類型，請參閱《Amazon CodeGuru Reviewer 使用者指南》中的 CodeGuru Reviewer 偵測的機密類型。

一旦您找到已寫死機密，請採取措施來替換它們：

- the section called “替換寫死資料庫憑證 ”
- the section called “替換已寫死機密 ”

將硬式編碼秘密移至 AWS Secrets Manager

如果您的程式碼中有純文字機密，我們建議您輪換它們並將它們儲存在機密管理員中。將機密移動到機密管理員可解決機密顯示問題，讓任何看到程式碼的人無法看見，因為再下一步，您的程式碼就能直接從機密管理員擷取機密。輪換機密將撤消目前已寫死機密，使其不再有效。

如需資料庫憑證機密，請參閱 將硬式編碼資料庫登入資料移至 AWS Secrets Manager。

在開始之前，您需要決定需要存取該機密的人。我們建議使用兩個 IAM 角色來管理您的機密權限：

- 管理組織中機密的角色。如需相關資訊，請參閱 the section called “Secrets Manager 管理員許可”。您將使用此角色建立並輪換機密。

- 可在執行階段使用機密的角色，例如在本教學中使用 *RoleToRetrieveSecretAtRuntime*。您的程式碼擔任此角色以擷取機密。在本教學中，您僅授予該角色擷取一個機密值的權限，並透過使用機密的資源策略授予權限。如需其他替代方案，請參閱 the section called “後續步驟”。

步驟：

- 步驟 1：建立機密
- 步驟 2：更新您的程式碼
- 步驟 3：更新機密
- 後續步驟

步驟 1：建立機密

第一步是將現有的已寫死機密複製到機密管理員中。如果秘密與 AWS 資源相關，請將秘密存放在與資源相同的區域中。否則，請將其存放在對您的使用案例具有最低延遲的區域。

若要建立機密 (控制台)

1. 前往以下位置開啟機密管理員控制台：https://console.aws.amazon.com/secretsmanager/。
2. 選擇 Store a new secret (存放新機密)。
3. 在 Choose secret type (選擇秘密類型) 頁面上，執行下列動作：
   a. 針對 Secret type (機密類型)，選擇 Other type of secret (其他機密類型)。
   b. 將您的機密輸入為 Key/value pairs (金鑰/值對) 或 Plaintext (純文字)。一些範例：

      API key

      輸入 做為鍵/值對：

      ClientID：*my_client_id*

      ClientSecret：*wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY*

      OAuth token

      輸入 做為純文字：

      AKIAI44QH8DHBEXAMPLE

Digital certificate

輸入 做為純文字：

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

Private key

輸入 做為純文字：

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

c. 如需 Encryption key (加密金鑰)，請選擇 aws/secretsmanager 使用用於機密管理員的 AWS 受管金鑰 。使用此金鑰無需任何成本。您也可以使用自己的顧客託管金鑰，例如從另一個 AWS 帳戶存取機密。如需有關使用客戶受管金鑰的成本的資訊，請參閱 定價。

d. 選擇下一步。

4. 在 Choose secret type (選擇秘密類型) 頁面上，執行下列動作：

a. 輸入描述性的 Secret name (機密名稱) 和 Description (描述)。

b. 在 Resource permissions (資源許可) 中，選擇 Edit permissions (編輯許可)。貼上以下政策，此政策允許 *RoleToRetrieveSecretAtRuntime* 擷取機密，然後選擇 Save (儲存)。

JSON

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

```
  ]
}
```

c. 請選擇頁面最下方的 Next (下一頁)。

5. 在 Configure rotation (設定輪換) 頁面上，保持輪換關閉。選擇下一步。
6. 在 Review (檢閱) 頁面上，檢閱機密詳細資訊，然後選擇 Store (存放)。

步驟 2：更新您的程式碼

您的程式碼必須擔任 IAM 角色 *RoleToRetrieveSecretAtRuntime* 以便能夠擷取到機密。如需詳細資訊，請參閱切換到 IAM 角色 (AWS API)。

接下來，您可以使用機密管理員提供的範本程式碼更新您的程式碼，以便從機密管理員中擷取機密。

若要尋找範本程式碼

1. 前往以下位置開啟機密管理員控制台：https://console.aws.amazon.com/secretsmanager/。
2. 在 Secrets (機密) 頁面中，選擇機密。
3. 向下捲動至 Sample code (範本程式碼)。選擇您的程式設計語言，然後複製程式碼片段。

在您的應用程式中，刪除已寫死機密並貼上程式碼片段。根據您的程式碼語言，您可能需要向程式碼片段中的函數或方法新增調用。

測試您的應用程式是否按預期運作，並使用機密代替已寫死機密。

步驟 3：更新機密

最後一個步驟是撤消並更新已寫死機密。請參閱機密的來源以尋找撤消和更新機密的指示。例如，您可能需要停用目前機密並產生新機密。

使用新值更新機密

1. 開啟位於的機密管理員控制台 https://console.aws.amazon.com/secretsmanager/。
2. 選擇 Secrets (機密)，然後選擇該機密。
3. 在 Secret details (機密詳細資訊) 頁面，向下捲動並選擇 Retrieve secret value (擷取機密值)，然後選擇 Edit (編輯)。
4. 更新機密，然後選擇 Save (儲存)。

接下來，測試您的應用程式在新機密下是否能按預期運作。

後續步驟

從程式碼中刪除已寫死機密後，接下來需要思考一些構想：

- 若要在 Java 和 Python 應用程式中尋找已寫死機密，我們推薦《Amazon CodeGuru Reviewer》。
- 您可以透過快取機密來提高效能並降低成本。如需詳細資訊，請參閱取得秘密。
- 如須從多個區域存取的機密，請考慮複製您的機密以改善延遲。如需詳細資訊，請參閱多區域複寫。
- 在此教學中，您授予 *RoleToRetrieveSecretAtRuntime* 僅擷取機密值的權限。若要向角色授予更多許可 (例如獲取有關機密的中繼資料或檢視機密清單)，請參閱 the section called “資源型政策”。
- 在此教學中，您透過使用機密的資源策略授予 *RoleToRetrieveSecretAtRuntime* 權限。如需授予權限的其他方式，請參閱 the section called “身分型政策”。

將硬式編碼資料庫登入資料移至 AWS Secrets Manager

如果您的程式碼中有純文字資料庫憑證，我們建議您將憑證移動到機密管理員，然後立即輪換它們。將憑證移動到機密管理員可解決憑證顯示問題，讓任何看到程式碼的人無法看見，因為再下一步，您的程式碼就能直接從機密管理員擷取憑證。輪換機密會更新密碼，然後撤銷目前的已寫死密碼，使其不再有效。

如需 Amazon RDS、Amazon Redshift 和 Amazon DocumentDB 資料庫，請使用此頁面中的步驟將寫死憑證移動到機密管理員。如需其他類型的憑證和其他機密，請參閱 the section called “替換已寫死機密 ”。

在開始之前，您需要決定需要存取該機密的人。我們建議使用兩個 IAM 角色來管理您的機密權限：

- 管理組織中機密的角色。如需相關資訊，請參閱 the section called “Secrets Manager 管理員許可”。您將使用此角色建立並輪換機密。
- 可在執行階段使用機密的角色，本教學中的 *RoleToRetrieveSecretAtRuntime*。您的程式碼擔任此角色以擷取機密。

步驟：

- 步驟 1：建立機密
- 步驟 2：更新您的程式碼

- 步驟 3：輪換機密
- 後續步驟

步驟 1：建立機密

第一步是將現有的寫死憑證複製到機密管理員中的機密中。如需實現最低延遲，請將機密儲存在與資料庫相同的區域中。

若要建立機密

1. 請開啟位於 https://console.aws.amazon.com/secretsmanager/ 的機密管理員控制台。
2. 選擇 Store a new secret (存放新機密)。
3. 在 Choose secret type (選擇秘密類型) 頁面上，執行下列動作：
   a. 針對 Secret type (機密類型)，選擇要存放的資料庫憑證的類型：
      - Amazon RDS 資料庫
      - Amazon DocumentDB 資料庫
      - Amazon Redshift 資料倉儲。
      - 如需其他類型的機密，請參閱《替換已寫死機密》。
   b. 如需憑證，請輸入資料庫的現有寫死憑證。
   c. 如需 Encryption key (加密金鑰)，請選擇 aws/secretsmanager 使用用於機密管理員的 AWS 受管金鑰 。使用此金鑰無需任何成本。您也可以使用自己的顧客託管金鑰，例如從另一個 AWS 帳戶存取機密。如需有關使用客戶受管金鑰的成本的資訊，請參閱 定價。
   d. 如需資料庫，請選擇您的資料庫。
   e. 選擇下一步。
4. 在 Configure secret (設定秘密) 頁面上，執行下列動作：
   a. 輸入描述性的 Secret name (機密名稱) 和 Description (描述)。
   b. 在 Resource permissions (資源許可) 中，選擇 Edit permissions (編輯許可)。貼上以下政策，此政策允許 *RoleToRetrieveSecretAtRuntime* 擷取機密，然後選擇 Save (儲存)。

      JSON

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

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

c. 請選擇頁面最下方的 Next (下一頁)。

5. 在 Configure rotation (設定輪換) 頁面上，暫時將輪換關閉。您稍後再打開。選擇 Next (下一步)。
6. 在 Review (檢閱) 頁面上，檢閱機密詳細資訊，然後選擇 Store (存放)。

步驟 2：更新您的程式碼

您的程式碼必須擔任 IAM 角色 *RoleToRetrieveSecretAtRuntime* 以便能夠擷取到機密。如需詳細資訊，請參閱切換到 IAM 角色 (AWS API)。

接下來，您可以使用機密管理員提供的範本程式碼更新您的程式碼，以便從機密管理員中擷取機密。

若要尋找範本程式碼

1. 前往以下位置開啟機密管理員控制台：https://console.aws.amazon.com/secretsmanager/。
2. 在 Secrets (機密) 頁面中，選擇機密。
3. 向下捲動至 Sample code (範本程式碼)。選擇您的語言，然後複製程式碼片段。

在您的應用程式中，刪除寫死憑證並貼上程式碼片段。根據您的程式碼語言，您可能需要向程式碼片段中的函數或方法新增調用。

測試您的應用程式是否按預期運作，並使用機密代替寫死憑證。

步驟 3：輪換機密

最後一步是透過輪換機密撤消寫死憑證。輪換是定期更新機密的過程。當您輪換機密時，會更新機密和資料庫中的憑證。機密管理員可以在您設定的時程上自動為您輪換機密。

設定輪換的一部分是確保 Lambda 輪換函數可以存取機密管理員和您的資料庫。在您開啟自動輪換後，機密管理員將在與資料庫相同的 VPC 中建立 Lambda 輪換函數，好讓它有網路可以存取資料庫。Lambda 輪換函數還必須能夠調用機密管理員來更新機密。我們建議您在 VPC 中建立 Secrets Manager 端點，以便從 Lambda 到 Secrets Manager 的呼叫不會離開 AWS 基礎設施。如需說明，請參閱the section called “VPC 端點 (AWS PrivateLink)”。

若要打開輪換功能

1. 請開啟位於 https://console.aws.amazon.com/secretsmanager/ 的機密管理員控制台。
2. 在 Secrets (機密) 頁面中，選擇機密。
3. 在 Secret details (機密詳細資訊) 頁面的 Rotation configuration (輪換組態) 區段中，選擇 Edit rotation (編輯輪換)。
4. 在 Edit rotation configuration (編輯輪換組態) 對話方塊中，執行以下動作：
   a. 開啟 Automatic rotation (自動輪換)。
   b. 在 Rotation schedule (輪換排程) 下，請以 UTC 時區輸入您的排程。
   c. 選擇 Rotate immediately when the secret is stored (儲存機密時立即輪換) 以在儲存變更時輪換您的機密。
   d. 在 Rotation function (輪換函數) 下，選擇 Create a new Lambda function (建立 Lambda 函數) 並輸入您的新函數名稱。機密管理員會將「SecretsManager」新增到函數名稱的開頭。
   e. 對於輪換策略，選擇單一使用者。
   f. 選擇儲存。

若要檢查機密是否輪換

1. 請開啟位於 https://console.aws.amazon.com/secretsmanager/ 的機密管理員控制台。
2. 選擇 Secrets (機密)，然後選擇該機密。
3. 在 Secret details (機密詳細資訊) 頁面，向下捲動並選擇 Retrieve secret value (擷取機密值)。

   如果機密值更改好了，則輪換成功。如果機密值沒有更改，您則需要透過 CloudWatch Logs 查看輪換函數來 輪換疑難排解。

測試您的應用程式在輪換的機密下是否能按預期運作。

後續步驟

從程式碼中刪除已寫死機密後，接下來需要思考一些構想：

- 您可以透過快取機密來提高效能並降低成本。如需詳細資訊，請參閱取得秘密。
- 您可以選擇不同的輪換計畫。如需詳細資訊，請參閱the section called “輪換排程”。
- 若要在 Java 和 Python 應用程式中尋找已寫死機密，我們推薦《Amazon CodeGuru Reviewer》。

為 AWS Secrets Manager設定交替使用者輪換

在此教學中，您將學習如何為包含資料庫憑證的秘密設定交替使用者輪換。交替使用者輪換是一種輪換策略，在該策略中，Secrets Manager 會複製使用者，然後交替使用更新的使用者憑證。如果您需要秘密高可用性，此策略是不錯的選擇，因為其中一個交替使用者具有資料庫的最新憑證，而另一個則正在更新。如需詳細資訊，請參閱the section called “交替使用者”。

若要設定交替使用者輪換，您需要兩個秘密：

- 一個秘密包含您要輪換的憑證。
- 具有管理員憑證的第二個秘密。

  此使用者具有複製第一個使用者和變更第一個使用者密碼的許可。在本教學中，您有 Amazon RDS 為管理員使用者建立此秘密。Amazon RDS 也管理管理員密碼輪換。如需詳細資訊，請參閱the section called “受管輪換”。

本教學的第一部分是設定一個真實的環境。為了向您展示輪換的運作方式，此教學使用了一個範例 Amazon RDS MySQL 資料庫。為安全起見，資料庫位於限制傳入網際網路存取的 VPC 中。若要透過網際網路從本機電腦連線至資料庫，請使用堡壘主機，這是 VPC 中可連線至資料庫的伺服器，而且允許從網際網路進行 SSH 連線。此教學中的堡壘主機是 Amazon EC2 執行個體，該執行個體的安全群組阻止其他類型的連線。

完成本教學後，建議您清除本教學的資源。請勿在生產環境中使用。

Secrets Manager 輪換使用 AWS Lambda 函數來更新秘密和資料庫。如需有關使用 Lambda 函數成本的資訊，請參閱 定價。

教學：

- 許可

- 先決條件
- 步驟 1：建立 Amazon RDS 資料庫使用者
- 步驟 2：為使用者的憑證建立秘密
- 步驟 3：測試輪換的秘密
- 步驟 4：清除資源
- 後續步驟

許可

對於教學必備條件，您需要 AWS 帳戶的管理許可。在生產設定中，最佳實務是針對每個步驟使用不同的角色。例如，具有資料庫管理員許可的角色將會建立 Amazon RDS 資料庫，具有網路管理員許可的角色將設定 VPC 和安全群組。對於教學步驟，我們建議您繼續使用相同的身分。

如需如何在生產環境中設定許可的詳細資訊，請參閱 the section called “身分驗證與存取控制”。

先決條件

在此教學課程中，您需執行下列項目：

- 先決條件 A：Amazon VPC
- 先決條件 B：Amazon EC2 執行個體
- 先決條件 C：管理員憑證的 Amazon RDS 資料庫和 Secrets Manager 秘密
- 先決條件 D：允許您的本機電腦連線到 EC2 執行個體

先決條件 A：Amazon VPC

在此步驟中，您會建立 VPC，可在其中啟動 Amazon RDS 資料庫和 Amazon EC2 執行個體。在稍後的步驟中，您會使用電腦透過網際網路連線到堡壘，然後連線到資料庫，因此您需要允許流量傳出 VPC。為此，Amazon VPC 會將網際網路閘道連接至 VPC，並在路由表中新增路由，以便將目的地為 VPC 外的流量傳送至網際網路閘道。

在 VPC 中，您會建立 Secrets Manager 端點和 Amazon RDS 端點。在稍後的步驟中設定自動輪換時，Secrets Manager 會在 VPC 內建立 Lambda 輪換函數，以便能夠存取資料庫。Lambda 輪換函數也會呼叫 Secrets Manager 來更新秘密，並呼叫 Amazon RDS 來取得資料庫連線資訊。透過在 VPC 中建立端點，您可以確保從 Lambda 函數對 Secrets Manager 和 Amazon RDS 的呼叫不會離開 AWS 基礎設施。而是路由至 VPC 內的端點。

建立 VPC

1. 在 https://console.aws.amazon.com/vpc/ 開啟 Amazon VPC 主控台。
2. 選擇建立 VPC。
3. 在 Create VPC (建立 VPC) 頁面上，選擇 VPC and more (VPC 和更多)。
4. 在 Name tag auto-generation (自動產生名稱標籤) 的 Auto-generate (自動產生) 中，輸入 **`SecretsManagerTutorial`**。
5. 對於 DNS options (DNS 選項)，請選擇 **`Enable DNS hostnames`** 和 **`Enable DNS resolution`**。
6. 選擇建立 VPC。

在 VPC 內建立 Secrets Manager 端點

1. 在 Amazon VPC 主控台的 Endpoints (端點) 中，選擇 Create Endpoint (建立端點)。
2. 在 Endpoint settings (端點設定) 中，針對 Name (名稱) 輸入 **`SecretsManagerTutorialEndpoint`**。
3. 在 Services (服務) 中，輸入 **`secretsmanager`** 以篩選清單，然後在 AWS 區域中選取 Secrets Manager 端點。例如，在美國東部 (維吉尼亞北部) 中，選擇 `com.amazonaws.us-east-1.secretsmanager`。
4. 對於 VPC，請選擇 **`vpc**** (SecretsManagerTutorial)`**。
5. 對於 Subnets (子網路)，選取所有 Availability Zones (可用區域)，然後針對每個選項選擇一個要包含的 Subnet ID (子網路 ID)。
6. 對於 IP address type (IP 地址類型)，請選擇 **`IPv4`**。
7. 對於 Security Groups (安全群組)，請選擇預設的安全群組。
8. 對於 Policy (政策)，請選擇 **`Full access`**。
9. 選擇建立端點。

在 VPC 內建立 Amazon RDS 端點

1. 在 Amazon VPC 主控台的 Endpoints (端點) 中，選擇 Create Endpoint (建立端點)。
2. 在 Endpoint settings (端點設定) 中，針對 Name (名稱) 輸入 **`RDSTutorialEndpoint`**。
3. 在 Services (服務) 中，輸入 **`rds`** 以篩選清單，然後在 AWS 區域中選取 Amazon RDS 端點。例如，在美國東部 (維吉尼亞北部) 中，選擇 `com.amazonaws.us-east-1.rds`。

4. 對於 VPC，請選擇 **vpc**** **(SecretsManagerTutorial)**。
5. 對於 Subnets (子網路)，選取所有 Availability Zones (可用區域)，然後針對每個選項選擇一個要包含的 Subnet ID (子網路 ID)。
6. 對於 IP address type (IP 地址類型)，請選擇 **IPv4**。
7. 對於 Security Groups (安全群組)，請選擇預設的安全群組。
8. 對於 Policy (政策)，請選擇 **Full access**。
9. 選擇建立端點。

先決條件 B：Amazon EC2 執行個體

您在稍後步驟中建立的 Amazon RDS 資料庫會位於 VPC 中，因此若要存取，您需有堡壘主機。堡壘主機也位於 VPC 中，但在稍後的步驟中，您會設定安全群組，允許您的本機電腦使用 SSH 連線到堡壘主機。

為堡壘主機建立 EC2 執行個體

1. 前往 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。
2. 選擇 Instances (執行個體)，然後選擇 Launch Instances (啟動執行個體)。
3. 在 Name and tags (名稱與標籤) 下，對於 Name (名稱)，輸入 **SecretsManagerTutorialInstance**。
4. 在 Application and OS Images (應用程式和作業系統映像) 中，保留預設值 **Amazon Linux 2 AMI (HMV) Kernel 5.10**。
5. 在 Instance type (執行個體類型) 中，保留預設值 **t2.micro**。
6. 在 Key pair (金鑰對) 中，選擇 Create key pair (建立金鑰對)。

   在 Create Key Pair (建立金鑰對) 對話方塊中，針對 Key pair name (金鑰對名稱) 輸入 **SecretsManagerTutorialKeyPair**，然後選擇 Create key pair (建立金鑰對)。

   系統會自動下載該金鑰對。
7. 在 Network settings (網路設定) 中，選擇 Edit (編輯)，接著執行下列動作：

   a. 對於 VPC，請選擇 **vpc-**** SecretsManagerTutorial**。
   b. 在 Auto-assign Public IP (自動指派公有 IP) 中，選擇 **Enable**。
   c. 對於 Firewall (防火牆)，請選擇 Select existing security group (選取現有的安全群組)。
   d. 對於 Common security groups (常見安全群組)，請選擇 **default**。

8. 選擇啟動執行個體。

先決條件 C：管理員憑證的 Amazon RDS 資料庫和 Secrets Manager 秘密

在此步驟中，您會建立 Amazon RDS MySQL 資料庫並加以設定，以便 Amazon RDS 建立秘密以加入管理憑證。Amazon RDS 接著會自動為您管理輪換管理員秘密。如需詳細資訊，請參閱受管輪換。

在建立資料庫時，您會指定在上一個步驟中建立的堡壘主機。Amazon RDS 接著會設定安全群組，以便資料庫和執行個體可彼此存取。您會將規則新增至連接執行個體的安全群組，允許本機電腦也連線到該執行個體。

使用包含管理員憑證的 Secrets Manager 秘密建立 Amazon RDS 資料庫

1. 在 Amazon RDS 主控台中，選擇 Create database (建立資料庫)。
2. 在 Engine options (引擎選項) 區段中，針對 Engine type (引擎類型) 選擇 **MySQL**。
3. 在 Templates (範本) 區段中，選擇 **Free tier**。
4. 在 Settings (設定) 區段中，執行下列動作：
   a. 對於 DB instance identifier (資料庫執行個體識別符)，請輸入 **SecretsManagerTutorial**。
   b. 在登入資料設定下，選取管理主要登入資料 AWS Secrets Manager。
5. 在 Connectivity (連線) 區段中，針對 Computer resource (電腦資源)，選擇 Connect to an EC2 computer resource (連線到 EC2 電腦資源)，然後針對 EC2 Instance (EC2 執行個體)，選擇 **SecretsManagerTutorialInstance**。
6. 選擇建立資料庫。

先決條件 D：允許您的本機電腦連線到 EC2 執行個體

在此步驟中，您會將先決條件 B 中建立的 EC2 執行個體，設定為允許本機電腦連線。為此，您會編輯 Amazon RDS 在先決條件 C 中新增的安全群組，加入規則，允許電腦 IP 地址使用 SSH 連線。此規則允許您的本機電腦 (以您目前的 IP 地址識別) 透過網際網路使用 SSH 連線到堡壘主機。

允許您的本機電腦連線到 EC2 執行個體

1. 前往 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。
2. 在 EC2 執行個體 SecretsManagerTutorialInstance Security (安全) 索引標籤的 Security groups (安全群組) 中，選擇 **sg-*** (ec2-rds-X)**。

3. 在 Inbound Rules (傳入規則) 中，選擇 Edit inbound rules (編輯傳入規則)。
4. 選擇 Add rule (新增規則)，然後針對該規則執行下列動作：
   a. 針對 Type (類型)，選擇 **SSH**。
   b. 針對 Source type (來源類型)，選擇 **My IP**。

步驟 1：建立 Amazon RDS 資料庫使用者

首先，您需要其憑證存放在秘密中的使用者。若要建立使用者，請使用管理員憑證登入 Amazon RDS 資料庫。為簡單起見，在本教學中，您會建立具有資料庫完整許可的使用者。在生產環境中，這並不典型，建議您遵守最低權限原則。

若要連線至資料庫，請使用 MySQL 用戶端工具。在此教學中，您將使用 MySQL Workbench，這是一個 GUI 型應用程式。若要安裝 MySQL Workbench，請參閱 Download MySQL Workbench (下載 MySQL Workbench)。

若要連線至資料庫，請在 MySQL Workbench 中建立連線組態。對於組態，您需要 Amazon EC2 和 Amazon RDS 提供的部分資訊。

在 MySQL Workbench 中建立資料庫連線

1. 在 MySQL Workbench 的 MySQL Connections (MySQL 連線) 旁邊，選擇 (+) 按鈕。
2. 在 Setup New Connection (設定新連線) 對話方塊中，請執行下列動作：
   a. 對於 Connection Name (連線名稱)，請輸入 **SecretsManagerTutorial**。
   b. 對於 Connection Method (連線方法)，請選擇 **Standard TCP/IP over SSH**。
   c. 在 Parameters (參數) 索引標籤上，請執行下列動作：
      i. 對於 SSH Hostname (SSH 主機名稱)，請輸入 Amazon EC2 執行個體的公有 IP 地址。

         您可以在 Amazon EC2 主控台上，透過選擇 SecretsManagerTutorialInstance 執行個體來尋找 IP 地址。在 Public IPv4 DNS (公有 IPv4 DNS) 下複製 IP 地址。
      ii. 對於 SSH Username (SSH 使用者名稱)，請輸入 **ec2-user**。
      iii. 對於 SSH Keyfile (SSH 金鑰檔案)，請選擇您在之前的先決條件中下載的 SecretsManagerTutorialKeyPair.pem 金鑰對檔案。
      iv. 對於 MySQL Hostname (MySQL 主機名稱)，請輸入 Amazon RDS 端點地址。

您可以在 Amazon RDS 主控台上，透過選擇 secretsmanagertutorialdb 資料庫執行個體來尋找端點地址。在 Endpoint (端點) 下複製地址。

v. 對於 User name (使用者名稱)，請輸入 **admin**。

d. 選擇確定。

擷取管理員密碼

1. 在 Amazon RDS 主控台中，導覽至您的資料庫。
2. 在 Configuration (組態) 索引標籤的 Master Credentials ARN (主要憑證 ARN) 中，選擇 Manage in Secrets Manager (在 Secrets Manager 中管理)。

   Secrets Manager 主控台隨即開啟。
3. 在秘密詳細資訊頁面上，選擇 Retrieve secret value (擷取秘密值)。
4. 密碼會顯示在 Secret value (秘密值) 區段。

建立資料庫使用者

1. 在 MySQL Workbench 中，選擇 SecretsManagerTutorial 連線。
2. 輸入您從秘密中擷取的管理員密碼。
3. 在 MySQL Workbench 的 Query (查詢) 視窗中，輸入下列命令 (包括一個強密碼)，然後選擇 Execute (執行)。輪換函數使用 SELECT 測試更新的秘密，因此 必須至少**appuser**具有該權限。

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

   在 Output (輸出) 視窗中，您會看到命令已成功。

步驟 2：為使用者的憑證建立秘密

接著，您可以建立一個秘密來存放剛建立使用者的憑證。這就是您將要輪換的秘密。開啟自動輪換，並指示交替使用者策略，您可以選擇單獨的進階使用者秘密，該秘密具有變更第一個使用者密碼的許可。

1. 前往以下位置開啟機密管理員控制台：https://console.aws.amazon.com/secretsmanager/。
2. 選擇 Store a new secret (存放新機密)。

3. 在 Choose secret type (選擇秘密類型) 頁面上，執行下列動作：

   a. 對於 Secret type (秘密類型)，請選擇 Credentials for Amazon RDS database (Amazon RDS 資料庫的憑證)。

   b. 對於 Credentials (憑證)，請輸入使用者名稱 **appuser** 以及您為使用 MySQL Workbench 建立的資料庫使用者輸入的密碼。

   c. 對於 Database (資料庫)，請選擇 secretsmanagertutorialdb。

   d. 選擇下一步。

4. 在 Configure secret (設定秘密) 頁面上，對於 Secret name (秘密名稱)，請輸入 **SecretsManagerTutorialAppuser**，然後選擇 Next (下一步)。

5. 在 Configure rotation (設定輪換) 頁面上，執行下列動作：

   a. 開啟 Automatic rotation (自動輪換)。

   b. 對於 Rotation schedule (輪換排程)，將排程設定為 Days (天數) ：**2** Duration (持續時間) ：**2h**。保持選取 Rotate immediately (立即輪換)。

   c. 對於 Rotation function (輪換函數)，請選擇 Create a rotation function (建立輪換函數)，然後對於函數名稱，請輸入 **tutorial-alternating-users-rotation**。

   d. 對於輪換策略，選擇交替使用者，然後在管理員憑證秘密之下，選擇名為 rds!cluster... 的秘密，其具有描述，其中包括您在本教學 **secretsmanagertutorial** 中所建立資料庫的名稱，例如 `Secret associated with primary RDS DB instance: arn:aws:rds:`*`Region`*`:`*`AccountId`*`:db:secretsmanagertutorial`。

   e. 選擇下一步。

6. 在 Review (檢閱) 頁面，選擇 Store (存放)。

   Secrets Manager 會返回秘密詳細資訊頁面。在頁面頂端，您可以看到輪換組態狀態。Secrets Manager 會使用 CloudFormation 建立資源，如 Lambda 輪換函數和執行 Lambda 函數的執行角色。在 CloudFormation 完成後，橫幅將變更為 Secret scheduled for rotation (排程輪換的秘密)。第一次輪換完成。

步驟 3：測試輪換的秘密

既然已輪換秘密，您可以檢查秘密是否包含有效的新憑證。秘密中的密碼已變更為原始憑證。

從秘密中擷取新密碼

1. 前往以下位置開啟機密管理員控制台：https://console.aws.amazon.com/secretsmanager/。

2. 選擇 Secrets (秘密)，然後選擇秘密 **SecretsManagerTutorialAppuser**。
3. 在 Secret details (秘密詳細資訊) 頁面，向下捲動並選擇 Retrieve secret value (擷取秘密值)。
4. 在 Key/value (鍵/值) 中，複製 **password** 的 Secret value (秘密值)。

測試憑證

1. 在 MySQL Workbench 中，在 SecretsManagerTutorial 連線上按一下滑鼠右鍵，然後選擇 Edit Connection (編輯連線)。
2. 在 Manage Server Connections (管理伺服器連線) 對話方塊，對於 Username (使用者名稱)，請輸入 **appuser**，然後選擇 Close (關閉)。
3. 返回 MySQL Workbench 中，選擇 SecretsManagerTutorial 連線。
4. 在 Open SSH Connection (開啟 SSH 連線) 對話方塊中，對於 Password (密碼)，貼上從秘密擷取的密碼，然後選擇 OK (確定)。

   如果憑證有效，則 MySQL Workbench 會開啟資料庫的設計頁面。

這表明秘密輪換是成功的。秘密中的憑證已更新，並且是連線至資料庫的有效密碼。

步驟 4：清除資源

如果您想要嘗試另一種輪換策略單一使用者輪換，請跳過清除資源並前往 the section called “單一使用者輪換”。

否則，為避免潛在的費用，以及移除可存取網際網路的 EC2 執行個體，請刪除您在此教學中建立的下列資源及其先決條件：

- Amazon RDS 資料庫執行個體。如需說明，請參閱《Amazon RDS 使用者指南》中的刪除資料庫執行個體。
- Amazon EC2 執行個體。如需說明，請參閱《Amazon EC2 使用者指南》中的終止執行個體。
- Secrets Manager 秘密 `SecretsManagerTutorialAppuser`。如需說明，請參閱the section called “刪除秘密”。
- Secrets Manager 端點。如需說明，請參閱《AWS PrivateLink 指南》中的刪除 VPC 端點。
- VPC 端點。如需說明，請參閱《AWS PrivateLink 指南》中刪除您的 VPC。

後續步驟

- 瞭解如何在應用程式中擷取秘密。
- 瞭解其他輪換排程。

為 AWS Secrets Manager設定單一使用者輪換

在此教學中，您會學習如何為包含資料庫憑證的秘密，設定單一使用者輪換。單一使用者輪換是一種輪換策略，使用此策略，Secrets Manager 會在秘密和資料庫中更新使用者的憑證。如需詳細資訊，請參閱the section called “單一使用者”。

完成本教學後，建議您清除本教學的資源。請勿在生產環境中使用。

Secrets Manager 輪換使用 AWS Lambda 函數來更新秘密和資料庫。如需有關使用 Lambda 函數成本的資訊，請參閱 定價。

內容

- 許可
- 先決條件
- 步驟 1：建立 Amazon RDS 資料庫使用者
- 步驟 2：為資料庫使用者憑證建立秘密
- 步驟 3：測試輪換密碼
- 步驟 4：清除資源
- 後續步驟

許可

對於教學必備條件，您需要 AWS 帳戶的管理許可。在生產設定中，最佳實務是針對每個步驟使用不同的角色。例如，具有資料庫管理員許可的角色將會建立 Amazon RDS 資料庫，具有網路管理員許可的角色將設定 VPC 和安全群組。對於教學步驟，我們建議您繼續使用相同的身分。

如需如何在生產環境中設定許可的詳細資訊，請參閱 the section called “身分驗證與存取控制”。

先決條件

此教學的先決條件是 the section called “交替使用者輪換”。請勿在第一個教學結束時清除資源。在該教學之後，您會擁有一個真實環境，具有 Amazon RDS 資料庫和包含資料庫管理員憑證的 Secrets Manager 秘密。您也有第二個秘密，其中包含資料庫使用者的憑證，但在本教學中不會使用該秘密。

此外，您還在 MySQL Workbench 中設定了一個連線，以便使用管理員憑證連線至資料庫。

步驟 1：建立 Amazon RDS 資料庫使用者

首先，您需要其憑證存放在秘密中的使用者。若要建立使用者，請使用存放在秘密中的管理員憑證，登入 Amazon RDS 資料庫。為簡單起見，在本教學中，您會建立具有資料庫完整許可的使用者。在生產環境中，這並不典型，建議您遵守最低權限原則。

擷取管理員密碼

1. 在 Amazon RDS 主控台中，導覽至您的資料庫。
2. 在 Configuration (組態) 索引標籤的 Master Credentials ARN (主要憑證 ARN) 中，選擇 Manage in Secrets Manager (在 Secrets Manager 中管理)。

   Secrets Manager 主控台隨即開啟。
3. 在秘密詳細資訊頁面上，選擇 Retrieve secret value (擷取秘密值)。
4. 密碼會顯示在 Secret value (秘密值) 區段。

建立資料庫使用者

1. 在 MySQL Workbench 中，在 SecretsManagerTutorial 連線上按一下滑鼠右鍵，然後選擇 Edit Connection (編輯連線)。
2. 在 Manage Server Connections (管理伺服器連線) 對話方塊，對於 Username (使用者名稱)，請輸入 **admin**，然後選擇 Close (關閉)。
3. 返回 MySQL Workbench 中，選擇 SecretsManagerTutorial 連線。
4. 輸入您從秘密中擷取的管理員密碼。
5. 在 MySQL Workbench 的 Query (查詢) 視窗中，輸入下列命令 (包括一個強密碼)，然後選擇 Execute (執行)。輪換函數使用 SELECT 測試更新的秘密，因此 必須至少**dbuser**具有該權限。

```
CREATE USER 'dbuser'@'%' IDENTIFIED BY 'EXAMPLE-PASSWORD';
```

```
GRANT SELECT ON myDB . * TO 'dbuser'@'%';
```

在 Output (輸出) 視窗中，您會看到命令已成功。

步驟 2：為資料庫使用者憑證建立秘密

接著，您會建立秘密，存放剛剛所建立使用者的憑證，然後開啟自動輪換 (包括立即輪換)。Secrets Manager 會輪換秘密，這表示會以程式設計方式產生密碼，沒有人會看到此新密碼。立即開始輪換也可協助您判斷是否正確設定輪換。

1. 前往以下位置開啟機密管理員控制台：https://console.aws.amazon.com/secretsmanager/。
2. 選擇 Store a new secret (存放新機密)。
3. 在 Choose secret type (選擇秘密類型) 頁面上，執行下列動作：
   a. 對於 Secret type (秘密類型)，請選擇 Credentials for Amazon RDS database (Amazon RDS 資料庫的憑證)。
   b. 對於 Credentials (憑證)，請輸入使用者名稱 **dbuser** 以及您為使用 MySQL Workbench 建立的資料庫使用者輸入的密碼。
   c. 對於 Database (資料庫)，請選擇 secretsmanagertutorialdb。
   d. 選擇下一步。
4. 在 Configure secret (設定秘密) 頁面上，對於 Secret name (秘密名稱)，請輸入 **SecretsManagerTutorialDbuser**，然後選擇 Next (下一步)。
5. 在 Configure rotation (設定輪換) 頁面上，執行下列動作：
   a. 開啟 Automatic rotation (自動輪換)。
   b. 對於 Rotation schedule (輪換排程)，將排程設定為 Days (天數) ：**2** Duration (持續時間) ：**2h**。保持選取 Rotate immediately (立即輪換)。
   c. 對於 Rotation function (輪換函數)，請選擇 Create a rotation function (建立輪換函數)，然後對於函數名稱，請輸入 **tutorial-single-user-rotation**。
   d. 對於輪換策略，選擇單一使用者。
   e. 選擇下一步。
6. 在 Review (檢閱) 頁面，選擇 Store (存放)。

Secrets Manager 會返回秘密詳細資訊頁面。在頁面頂端，您可以看到輪換組態狀態。Secrets Manager 會使用 CloudFormation 建立資源，如 Lambda 輪換函數和執行 Lambda 函數的執行角

色。在 CloudFormation 完成後，橫幅將變更為 Secret scheduled for rotation (排程輪換的秘密)。第一次輪換完成。

步驟 3：測試輪換密碼

在第一次秘密輪換之後，這可能需要幾秒鐘，您可以檢查秘密是否仍包含有效憑證。秘密中的密碼已變更為原始憑證。

從秘密中擷取新密碼

1. 前往以下位置開啟機密管理員控制台：https://console.aws.amazon.com/secretsmanager/。
2. 選擇 Secrets (秘密)，然後選擇秘密 **SecretsManagerTutorialDbuser**。
3. 在 Secret details (秘密詳細資訊) 頁面，向下捲動並選擇 Retrieve secret value (擷取秘密值)。
4. 在 Key/value (鍵/值) 中，複製 **password** 的 Secret value (秘密值)。

測試憑證

1. 在 MySQL Workbench 中，在 SecretsManagerTutorial 連線上按一下滑鼠右鍵，然後選擇 Edit Connection (編輯連線)。
2. 在 Manage Server Connections (管理伺服器連線) 對話方塊，對於 Username (使用者名稱)，請輸入 **dbuser**，然後選擇 Close (關閉)。
3. 返回 MySQL Workbench 中，選擇 SecretsManagerTutorial 連線。
4. 在 Open SSH Connection (開啟 SSH 連線) 對話方塊中，對於 Password (密碼)，貼上從秘密擷取的密碼，然後選擇 OK (確定)。

   如果憑證有效，則 MySQL Workbench 會開啟資料庫的設計頁面。

步驟 4：清除資源

為避免潛在的費用，請刪除您在此教學中建立的秘密。如需說明，請參閱the section called “刪除秘密”。

若要清除在上一個教學中建立的資源，請參閱 the section called “步驟 4：清除資源”。

後續步驟

- 瞭解如何在應用程式中擷取秘密。請參閱 取得秘密。

- 瞭解其他輪換排程。請參閱 the section called “輪換排程”。

建立 AWS Secrets Manager 秘密

秘密可以是以加密形式存放在 Secrets Manager 中的一個密碼、一組憑證，例如使用者名稱和密碼、OAuth 字符或其他秘密資訊。

> **Tip**
>
> 對於 Amazon RDS 和 Amazon Redshift 管理員使用者憑證，我們建議您使用 受管秘密。您可以透過 管理服務建立 受管秘密，然後使用 受管輪換。

當您使用 主控台存放複寫至其他 區域的來源資料庫的資料庫登入資料時，秘密會包含來源資料庫的連線資訊。如果隨後複製機密，則複本是來源機密的副本，並包含相同的連線資訊。您可以將其他金鑰/值對新增到區域連線資訊的機密。

若要建立秘密，您需要 SecretsManagerReadWrite 受管政策授予的許可。

建立機密時，Secrets Manager 會產生 CloudTrail 日誌項目。如需詳細資訊，請參閱the section called “使用 記錄 AWS CloudTrail ”。

若要建立秘密 (主控台)

1. 前往以下位置開啟機密管理員控制台：https://console.aws.amazon.com/secretsmanager/。
2. 選擇 Store a new secret (存放新機密)。
3. 在 Choose secret type (選擇秘密類型) 頁面上，執行下列動作：

   a. 針對 Secret Type (秘密類型)，執行下列其中一項操作：

      - 若要存放資料庫登入資料，請選擇要存放的資料庫登入資料類型。然後選擇資料庫，然後輸入登入資料。
      - 若要存放 API 金鑰、存取權杖、非資料庫的登入資料，請選擇其他類型的秘密。

        在鍵值對中，您可以在 JSON 鍵值對中輸入秘密，也可以先選擇純文字索引標籤，再以任何格式輸入秘密。秘密當中最多可以存放 65536 個位元組。一些範例：

        API key

        輸入 做為鍵/值對：

        `ClientID`：*`my_client_id`*

ClientSecret : *wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY*

OAuth token

以純文字輸入：

AKIAI44QH8DHBEXAMPLE

Digital certificate

以純文字輸入：

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

Private key

以純文字輸入：

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

b. 針對加密金鑰，選擇 AWS KMS key Secrets Manager 用來加密秘密值的 。如需詳細資訊，請參閱秘密加密和解密。

- 大多數情況下，選擇 aws/secretsmanager 以將 AWS 受管金鑰 用於 Secrets Manager。使用此金鑰無需任何成本。
- 如果您需要從另一個 存取秘密 AWS 帳戶，或者如果您想要使用自己的 KMS 金鑰來輪換秘密或套用金鑰政策，請從清單中選擇客戶受管金鑰，或選擇新增金鑰來建立秘密。如需有關使用客戶受管金鑰的成本的資訊，請參閱 定價。

  您必須擁有the section called “KMS 金鑰的許可”。如需跨帳戶存取權的詳細資訊，請參閱the section called “跨帳戶存取權”。

c. 選擇下一步。

4. 在 Configure secret (設定秘密) 頁面上，執行下列動作：

a. 輸入描述性的 Secret name (機密名稱) 和 Description (描述)。秘密名稱可包含 1-512 個英數字元和 /_+=.@- 字元。

b. (選用) 在 Tags (標籤) 區段，將標籤新增到秘密。如需標記策略，請參閱 the section called "標籤 秘密"。請勿在標籤中存放敏感資訊，因為標籤並未加密。

c. (選用) 若要將資源政策新增至秘密，請在 Resource permissions (資源使用權限) 中選擇 Edit permissions (編輯許可)。如需詳細資訊，請參閱the section called "資源型政策"。

d. （選用）在複寫秘密中，若要將秘密複寫至另一個秘密 AWS 區域，請選擇複寫秘密。您可以立即複寫秘密，也可以稍後返回複寫。如需詳細資訊，請參閱多區域複寫。

e. 選擇下一步。

5. (選用) 在 Configure rotation (設定輪換) 頁面上，可開啟自動輪換。您也可以暫時關閉輪換，稍後再將其開啟。如需詳細資訊，請參閱輪換 秘密。選擇下一步。

6. 在 Review (檢閱) 頁面上，檢閱機密詳細資訊，然後選擇 Store (存放)。

   Secrets Manager 會傳回秘密清單。如果您的新秘密沒有顯示，請選擇重新整理按鈕。

AWS CLI

在命令 shell 中輸入命令時，存在命令歷史記錄被存取或公用程式存取命令參數的風險。請參閱 the section called "降低使用 AWS CLI 存放 AWS Secrets Manager 秘密的風險"。

Example 從 JSON 檔案中的資料庫登入資料建立秘密

下列 `create-secret` 範例會透過檔案中的憑證建立機密。如需詳細資訊，請參閱 AWS CLI 《使用者指南》中的從檔案載入 AWS CLI 參數。

為了使 Secrets Manager 能夠輪換秘密，您必須確保 JSON 與 機密的 JSON 結構 相符。

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

mycreds.json 的內容：

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

```
}
```

Example 建立秘密

下列 create-secret 範例會建立具有兩個金鑰值對的機密。

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

Example 建立秘密

下列create-secret範例會建立具有兩個標籤的秘密。

```
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 開發套件

若要使用其中一個 AWS SDKs建立秘密，請使用 CreateSecret動作。如需詳細資訊，請參閱the section called “AWS SDKs”。

Secrets Manager 秘密中有什麼內容？

在 Secrets Manager 中，秘密由秘密資訊、秘密值，以及關於秘密的中繼資料組成。秘密值可以為字串或二進位。

若要將多個字串值儲存在一個秘密中，建議您使用 JSON 文字字串搭配鍵值對，例如：

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

```
  "engine"     : "mysql"
}
```

對於資料庫秘密，如果您想要開啟自動輪換，則秘密必須包含資料庫的連線資訊，其為正確的 JSON 結構。如需詳細資訊，請參閱the section called “機密的 JSON 結構”。

中繼資料

秘密的中繼資料包括：

- Amazon Resource Name (ARN) 具有以下格式：

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

  Secrets Manager 會在秘密名稱末尾包含六個隨機字元，協助確保秘密 ARN 是唯一。如果刪除原始秘密，然後使用相同的名稱建立新秘密，則這兩個秘密會因為這些字元而具有不同的 ARN。具有舊秘密存取權的使用者不會自動取得新秘密的存取權，因為 ARN 不同。
- 秘密的名稱、描述、資源政策和標籤。
- 加密金鑰的 ARN， AWS KMS key Secrets Manager 用來加密和解密秘密值的 。Secrets Manager 一律會以加密形式存放秘密文字，並且一律加密傳輸中的秘密。請參閱 the section called “秘密加密和解密”。
- 關於如何輪換秘密的資訊 (如果設定輪換)。請參閱 輪換 秘密。

Secrets Manager 使用 IAM 許可政策，以確保只有授權的使用者才能存取或修改秘密。請參閱 的身分驗證和存取控制 AWS Secrets Manager。

秘密的版本會保留加密秘密值的副本。在變更秘密值或輪換秘密後，Secrets Manager 會建立新的版本。請參閱 the section called “秘密版本”。

您可以透過複寫秘密 AWS 區域 來跨多個秘密。在複寫秘密時，您會建立原始或主要秘密 (稱為複本秘密) 的副本。複本秘密會保持與主要秘密的連結。請參閱 多區域複寫。

請參閱 管理秘密。

秘密版本

秘密的版本會保留加密秘密值的副本。在變更秘密值或輪換秘密後，Secrets Manager 會建立新的版本。

Secrets Manager 不會儲存帶有版本的秘密線性歷史記錄，相反地，它會標記三個特定版本來追蹤它們：

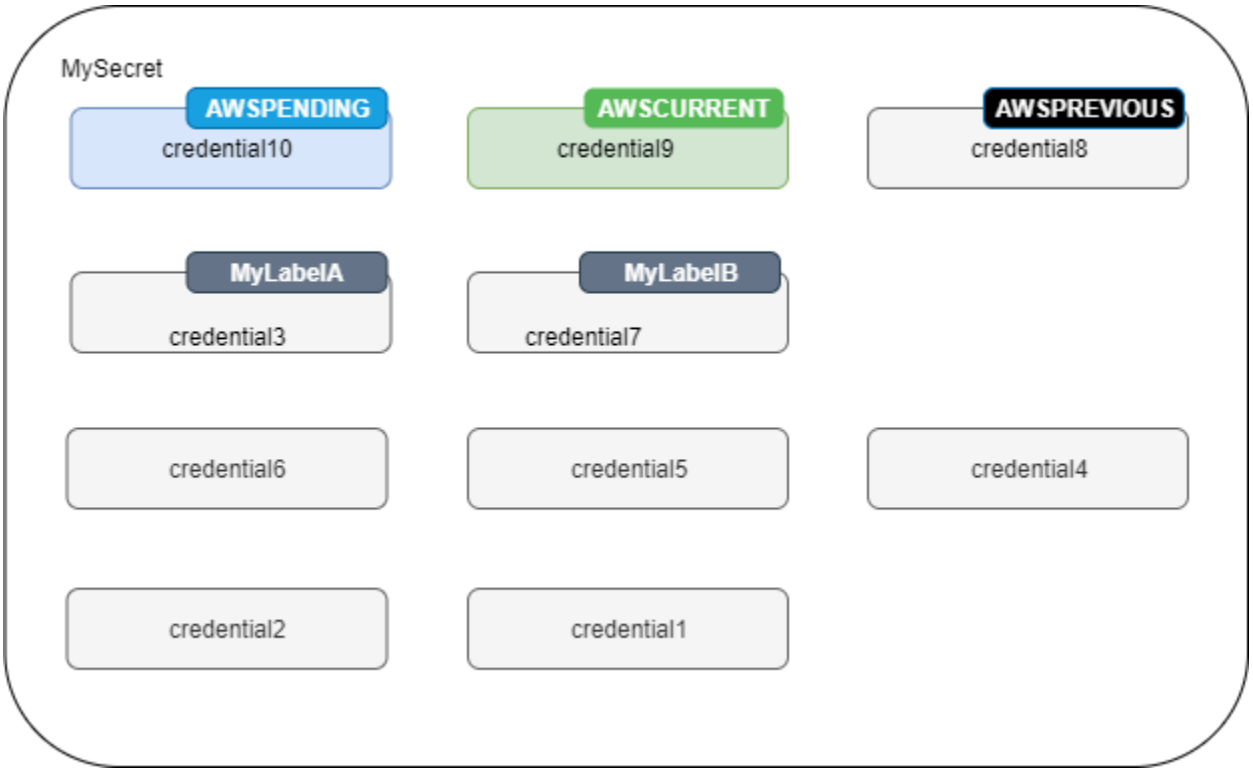
- 目前版本 – AWSCURRENT
- 舊版 – AWSPREVIOUS
- 待定版本 （輪換期間） – AWSPENDING

秘密始終有一個標記為 AWSCURRENT 的版本，當您擷取秘密值時，Secrets Manager 會依預設傳回該版本。

您也可以在 update-secret-version-stage中呼叫 ，以使用您自己的標籤來標記版本 AWS CLI。您最多可以將 20 個標籤連接至秘密中的版本。秘密的兩個版本不能擁有相同的預備標籤。一個版本可以有多個標籤。

Secrets Manager 永遠不會移除已標記版本，但會將未標記版本視為已遭取代。如果版本超過 100 個，Secrets Manager 會移除已棄用版本。Secrets Manager 不會刪除建立時間不到 24 小時的版本。

下圖顯示具有 AWS 標籤版本和客戶標籤版本的秘密。未標記版本將被視為已遭取代，並將在未來某個時間點被 Secrets Manager 移除。



AWS Secrets Manager 秘密的 JSON 結構

您可以在 Secrets Manager 秘密中存放任何文字或二進位，大小上限為 65，536 個位元組。

如果您使用 the section called “依 Lambda 函數輪換”，秘密必須包含輪換函數預期的特定 JSON 欄位。例如，對於包含資料庫登入資料的秘密，輪換函數會連線至資料庫以更新登入資料，因此秘密必須包含資料庫連線資訊。

如果您使用 主控台編輯資料庫秘密的輪換，則秘密必須包含可識別資料庫的特定 JSON 鍵/值對。Secrets Manager 使用這些欄位來查詢資料庫，以尋找要存放輪換函數的正確 VPC。

JSON 金鑰名稱區分大小寫。

主題

- Amazon RDS 和 Aurora 登入資料
- Amazon Redshift 登入資料
- Amazon Redshift Serverless 登入資料
- Amazon DocumentDB 登入資料
- InfluxDB 秘密結構的 Amazon Timestream
- Amazon ElastiCache 登入資料
- Active Directory 登入資料

Amazon RDS 和 Aurora 登入資料

若要使用 Secrets Manager 提供的輪換函數範本，請使用下列 JSON 結構。您可以新增更多鍵/值對，例如包含其他區域中複本資料庫的連線資訊。

DB2

對於 Amazon RDS Db2 執行個體，因為使用者無法變更自己的密碼，因此必須使用單獨的密碼來提供管理員登入資料。

```
{
  "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 登入資料

若要使用 Secrets Manager 提供的輪換函數範本，請使用下列 JSON 結構。您可以新增更多鍵/值對，例如包含其他區域中複本資料庫的連線資訊。

```
{
  "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 登入資料

若要使用 Secrets Manager 提供的輪換函數範本，請使用下列 JSON 結構。您可以新增更多鍵/值對，例如包含其他區域中複本資料庫的連線資訊。

```
{
  "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 登入資料

若要使用 Secrets Manager 提供的輪換函數範本，請使用下列 JSON 結構。您可以新增更多鍵/值對，例如包含其他區域中複本資料庫的連線資訊。

```
{
  "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.>"
}
```

InfluxDB 秘密結構的 Amazon Timestream

若要輪換 Timestream 秘密，您可以使用the section called “InfluxDB 的 Amazon Timestream”輪換範本。

如需詳細資訊，請參閱《Amazon Timestream 開發人員指南》中的 Amazon Timestream for InfluxDB 如何使用秘密。

Timestream 秘密必須位於正確的 JSON 結構，才能使用輪換範本。如需詳細資訊，請參閱《Amazon Timestream 開發人員指南》中的秘密內容。

Amazon ElastiCache 登入資料

下列範例顯示存放 ElastiCache 憑證之秘密的 JSON 結構。

```
{
```

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

如需詳細資訊，請參閱《Amazon ElastiCache 使用者指南》中的自動輪換使用者的密碼。

Active Directory 登入資料

AWS Directory Service 使用秘密來存放 Active Directory 登入資料。如需詳細資訊，請參閱 AWS Directory Service 管理指南中的無縫加入 Amazon EC2 Linux 執行個體到您的 Managed AD Active Directory。無縫網域聯結需要下列範例中的金鑰名稱。如果您不使用無縫網域聯結，可以使用環境變數變更秘密中的金鑰名稱，如輪換函數範本程式碼所述。

若要輪換 Active Directory 秘密，您可以使用 Active Directory 輪換範本。

Active Directory credential

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

如果您想要輪換秘密，請包含網域目錄 ID。

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

如果秘密與包含 keytab 的秘密搭配使用，您可以包含 keytab 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

如需使用 keytab 檔案對 Amazon EC2 上的 Active Directory 帳戶進行身分驗證的資訊，請參閱在 Amazon Linux 2 上使用 SQL Server 2017 部署和設定 Active Directory 身分驗證。

```
{
  "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 管理秘密 AWS Secrets Manager

主題

- 更新 AWS Secrets Manager 秘密的值
- 使用 Secrets Manager 產生密碼
- 將秘密復原至先前的版本
- 變更 AWS Secrets Manager 秘密的加密金鑰
- 修改 AWS Secrets Manager 秘密
- 在 中尋找秘密 AWS Secrets Manager
- 刪除 AWS Secrets Manager 秘密
- 還原 AWS Secrets Manager 秘密
- 在 中標記秘密 AWS Secrets Manager

更新 AWS Secrets Manager 秘密的值

若要更新秘密的值，您可以使用主控台、CLI 或軟體開發套件。在更新秘密值時，Secrets Manager 會使用預備標籤 AWSCURRENT 建立新的秘密版本。您仍然可以存取具有標籤 AWSPREVIOUS 的舊版本。您還可以新增自己的標籤。如需詳細資訊，請參閱 Secrets Manager 版本控制。

若要更秘密值 (主控台)

1. 於 https://console.aws.amazon.com/secretsmanager/ 開啟 Secrets Manager 主控台。
2. 從秘密清單中選擇秘密。
3. 在秘密詳細資訊頁面的概觀分頁上，在秘密值區段中，選擇擷取秘密值，然後選擇編輯。

AWS CLI

若要更新秘密值 (AWS CLI)

- 在命令 shell 中輸入命令時，存在命令歷史記錄被存取或公用程式存取命令參數的風險。請參閱 the section called “降低使用 AWS CLI 存放 AWS Secrets Manager 秘密的風險”。

  下列 `put-secret-value` 會建立具有兩個金鑰值對之新版本的機密。

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

以下 put-secret-value 會建立具有自訂預備標籤的新版本。新版本將具有標籤 MyLabel 和 AWSCURRENT。

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

AWS 開發套件

建議您避免以超過每 10 分鐘一次的持續速率呼叫 PutSecretValue 或 UpdateSecret。在呼叫 PutSecretValue 和 UpdateSecret 更新機密值時，機密管理員會建立機密的新版本。當版本超過 100 個時，Secrets Manager 會移除未標記的版本，但不會移除建立時間不到 24 小時的版本。如果以超過每 10 分鐘一次的速率更新機密值，您建立的版本比機密管理員移除的版本更多，且您將達到機密版本的配額。

若要更新秘密值，請使用以下動作：UpdateSecret 或 PutSecretValue。如需詳細資訊，請參閱the section called “AWS SDKs”。

使用 Secrets Manager 產生密碼

使用 Secrets Manager 的常見模式是在 Secrets Manager 中產生密碼，然後在資料庫或服務中使用該密碼。您可以使用下列方式執行這項操作：

- AWS CloudFormation – 請參閱 AWS CloudFormation。
- AWS CLI – 請參閱 get-random-password。
- AWS SDKs – 請參閱 GetRandomPassword。

將秘密復原至先前的版本

您可以使用 移動連接至秘密版本的標籤，將秘密還原至先前的版本 AWS CLI。如需 Secrets Manager 如何存放秘密版本的資訊，請參閱 the section called “秘密版本”。

下列update-secret-version-stage範例會將 AWSCURRENT 預備標籤移至舊版的秘密，將秘密還原至舊版。若要尋找舊版的 ID，請使用 list-secret-version-ids或檢視 Secrets Manager 主控台中的版本。

在此範例中，具有 AWSCURRENT 標籤的版本為 a1b2c3d4-5678-90ab-cdef-EXAMPLE111111，具有 AWSPREVIOUS 標籤的版本為 a1b2c3d4-5678-90ab-cdef-EXAMPLE22222。在此範例中，您將 AWSCURRENT 標籤從 11111 版移至 22222。由於 AWSCURRENT 標籤已從版本中移除，update-secret-version-stage會自動將 AWSPREVIOUS 標籤移至該版本 (11111)。效果是交換 AWSCURRENT 和 AWSPREVIOUS 版本。

```
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
```

變更 AWS Secrets Manager 秘密的加密金鑰

Secrets Manager 使用信封加密搭配 AWS KMS 金鑰和資料金鑰來保護每個秘密值。對於每個秘密，您可以選擇要使用的 KMS 金鑰。您可以使用 AWS 受管金鑰 aws/secretsmanager，也可以使用客戶受管金鑰。對於大多數情況，我們建議使用 aws/secretsmanager，而且使用它無需付費。如果您需要從另一個 存取秘密 AWS 帳戶，或者如果您想要使用自己的 KMS 金鑰來輪換秘密或套用金鑰政策，請使用 客戶受管金鑰。您必須擁有the section called “KMS 金鑰的許可”。如需有關使用客戶受管金鑰的成本的資訊，請參閱 定價。

您可以變更秘密的加密金鑰。例如，如果您想要從另一個帳戶存取秘密，而且秘密目前是使用 AWS 受管金鑰 加密aws/secretsmanager，您可以切換到 客戶受管金鑰。

> **Tip**
>
> 如果您想要輪換 客戶受管金鑰，建議您使用 AWS KMS 自動金鑰輪換。如需詳細資訊，請參閱輪換 AWS KMS 金鑰。

當您變更加密金鑰時，Secrets Manager 會使用新金鑰重新加密 AWSPENDING、 AWSCURRENT和AWSPREVIOUS版本。為了避免將您鎖定在秘密之外，Secrets Manager 會保留所有現有版本，並使用先前的金鑰加密。這表示您可以使用先前的金鑰或新的金鑰來解密 AWSCURRENTAWSPENDING、 和AWSPREVIOUS版本。如果您沒有前一個金鑰的kms:Decrypt許可，當您變更加密金鑰時，Secrets Manager 無法解密秘密版本以重新加密它們。在此情況下，現有的版本不會重新加密。

若要讓它AWSCURRENT只能由新的加密金鑰解密，請使用新金鑰建立新的秘密版本。若要能夠解密AWSCURRENT秘密版本，您必須擁有新金鑰的許可。

如果停用先前的加密金鑰，則除了 AWSCURRENT、AWSPENDING 和 AWSPREVIOUS 之外，您將無法解密任何秘密版本。如果您有其他要保留存取的標記秘密版本，則需要使用 the section called “AWS CLI”，以新的加密金鑰重新建立這些版本。

若要變更秘密的加密金鑰 (主控台)

1. 於 https://console.aws.amazon.com/secretsmanager/ 開啟 Secrets Manager 主控台。
2. 從秘密清單中選擇秘密。
3. 在秘密詳細資訊頁面上，在秘密詳細資訊區段中，選擇動作，然後選擇編輯加密金鑰。

AWS CLI

如果您變更秘密的加密金鑰，然後停用先前的加密金鑰，則除了 AWSCURRENT、AWSPENDING 和 AWSPREVIOUS 之外，您將無法解密任何秘密版本。如果您有其他要保留存取的標記秘密版本，則需要使用 the section called “AWS CLI”，以新的加密金鑰重新建立這些版本。

若要變更秘密的加密金鑰 (AWS CLI)

1. 下列 `update-secret` 範例會更新用於加密機密值的 KMS 金鑰。KMS 金鑰必須位於與機密相同的區域。

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

2. (選擇性) 如果您有具有自訂標籤的秘密版本，若要使用新金鑰重新加密，則您必須重新建立這些版本。

   在命令 shell 中輸入命令時，存在命令歷史記錄被存取或公用程式存取命令參數的風險。請參閱 the section called “降低使用 AWS CLI 存放 AWS Secrets Manager 秘密的風險”。

   a. 取得秘密版本的值。

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

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

記下秘密值。

b. 使用該值建立新版本。

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

修改 AWS Secrets Manager 秘密

您可以在建立秘密後修改中繼資料，具體取決於建立秘密的人員。對於由其他服務建立的秘密，您可能需要使用其他服務來更新或輪換它。

若要決定管理秘密的人員，您可以檢閱秘密名稱。由其他服務管理的秘密以該服務的 ID 為字首。或者，在 中 AWS CLI呼叫 describe-secret，然後檢閱 欄位 `OwningService`。如需詳細資訊，請參閱由其他服務管理的秘密。

對於您管理的秘密，您可以修改說明、資源型政策、加密金鑰和標籤。雖然我們建議您使用輪換來更新含有憑證的秘密值，但您也可以變更加密的秘密值。輪換會同時更新 Secrets Manager 中的秘密以及資料庫或服務上的憑證。這可讓秘密自動同步，以便在用戶端請求秘密值時，隨時都能擷取運作中的一組憑證。如需詳細資訊，請參閱輪換 秘密。

修改機密時，Secrets Manager 會產生 CloudTrail 日誌項目。如需詳細資訊，請參閱the section called “使用 記錄 AWS CloudTrail ”。

若要更新您管理的秘密 (主控台)

1. 於 https://console.aws.amazon.com/secretsmanager/ 開啟 Secrets Manager 主控台。
2. 從秘密清單中選擇秘密。
3. 在秘密詳細資訊頁面上，執行下列動作：

   請注意，您無法變更秘密的名稱或 ARN。

   - 若要更新描述，請在 Secrets details (秘密詳細資訊) 區段，選擇 Actions (動作)，然後選擇 Edit description (編輯描述)。
   - 若要更新加密金鑰，請參閱 the section called “變更秘密的加密金鑰”。

- 若要更新標籤，在標籤分頁上，選擇編輯標籤。請參閱 the section called “標籤 秘密”。
- 若要更新秘密值，請參閱 the section called “更新秘密值”。
- 若要更新密秘的許可，在概觀分頁上，選擇編輯許可。請參閱 the section called “資源型政策”。

- 若要更新機密的輪換，在輪換分頁上，選擇編輯輪換。請參閱 輪換 秘密。
- 若要將您的秘密複寫到其他區域，請參閱多區域複寫。
- 如果您的秘密有複本，您可以變更複本的加密金鑰。在複寫分頁上，選取複本的選項按鈕，然後在動作選單中，選擇編輯加密金鑰 。請參閱 the section called “秘密加密和解密”。
- 若要變更機密，使其由其他服務管理，您需要在該服務中重新建立機密。請參閱 由其他服務管理的秘密。

AWS CLI

Example 更新機密描述

下列 `update-secret` 範例會更新機密的描述。

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

AWS 開發套件

建議您避免以超過每 10 分鐘一次的持續速率呼叫 PutSecretValue 或 UpdateSecret。在呼叫 PutSecretValue 和 UpdateSecret 更新機密值時，機密管理員會建立機密的新版本。當版本超過 100 個時，Secrets Manager 會移除未標記的版本，但不會移除建立時間不到 24 小時的版本。如果以超過每 10 分鐘一次的速率更新機密值，您建立的版本比機密管理員移除的版本更多，且您將達到機密版本的配額。

若要更新秘密，請使用以下動作：`UpdateSecret` 或 `ReplicateSecretToRegions`。如需詳細資訊，請參閱the section called “AWS SDKs”。

在 中尋找秘密 AWS Secrets Manager

當您在不使用篩選條件的情況下搜尋秘密時，Secrets Manager 會比對秘密名稱、描述、標籤金鑰和標籤值中的關鍵字。不使用篩選條件進行搜尋並不區分大小寫，而且會忽略特殊字元，例如空

格、/、_、=、#，而且只使用數字和字母。在沒有篩選條件的情況下進行搜尋時，Secrets Manager 會分析搜尋字串以將其轉換為單獨的單詞。單詞透過從大寫到小寫、從字母到數字，或從數字/字母到標點符號的任何變化來分隔。例如，輸入搜尋詞 `credsDatabase#892` 以在名稱、描述、標籤索引鍵和值中搜尋`creds`、`Database` 和 `892`。

列出機密時，Secrets Manager 會產生 CloudTrail 日誌項目。如需詳細資訊，請參閱the section called “使用 記錄 AWS CloudTrail ”。

Secrets Manager 是一項區域服務，僅傳回所選區域中的秘密。

搜尋篩選條件

如果您不使用任何篩選條件，Secrets Manager 會將搜尋字串分成單字，然後搜尋相符項目的所有屬性。此搜尋不區分大小寫。例如，搜尋 會將秘密與名稱、描述或標籤中的 my 或 secret 一詞**`My_Secret`**相符。

您可以將下列篩選條件套用至您的搜尋：

名稱

比對秘密名稱的開頭；區分大小寫。例如：Name: **`Data`** 會傳回名為 `DatabaseSecret` 的秘密，而不是 `databaseSecret` 或者 `MyData`。

描述

比對秘密描述中的字詞，不區分大小寫。例如：Description: **`My Description`** 會將秘密與下列說明進行比對：

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

管理者

尋找由 外部服務管理的秘密 AWS，例如 CyberArk 或 HashiCorp。

擁有服務

比對管理服務 ID 前綴的開頭，不區分大小寫。例如，**`my-ser`** 使用前綴 `my-serv` 和 `my-service` 比對由服務管理的機密。如需詳細資訊，請參閱由其他服務管理的秘密。

已複寫的秘密

您可以篩選主要秘密、複本秘密或未複寫的秘密。

標籤金鑰

比對標籤金鑰的開頭；區分大小寫。例如：Tag key: **Prod** 返回帶有標籤 Production 和 Prod1 的秘密，而不是帶有標籤 prod 或者 1 Prod 的秘密。

標籤值

比對標籤值的開頭；區分大小寫。例如：Tag value: **Prod** 返回帶有標籤 Production 和 Prod1 的秘密，而不是帶有標籤值 prod 或者 1 Prod 的秘密。

AWS CLI

Example 列出帳戶中的機密

下列 list-secrets 範例會取得您帳戶中的機密清單。

```
aws secretsmanager list-secrets
```

Example 篩選帳戶中的機密清單

下列 list-secrets 範例會取得您帳戶中名稱包含 Test 的機密清單。依名稱篩選區分大小寫。

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

Example 尋找由其他 AWS 服務管理的秘密

下列 list-secrets 範例會取得由服務管理的秘密清單。您可以依 ID 指定服務。如需詳細資訊，請參閱由其他服務管理的秘密。

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

AWS 開發套件

若要使用其中一個 AWS SDKs 尋找秘密，請使用 ListSecrets。如需詳細資訊，請參閱the section called “AWS SDKs”。

刪除 AWS Secrets Manager 秘密

由於秘密的關鍵性質， AWS Secrets Manager 刻意讓刪除秘密變得困難。Secrets Manager 不會立即刪除秘密。反之，Secrets Manager 會立即使秘密無法存取，並排定在最少七天的復原時段後刪除。在復原時段結束之前，您都可以恢復之前刪除的秘密。不會針對您已標示為刪除的秘密收取任何費用。

如果主要秘密複寫到其他區域，則無法刪除它。先刪除複本，然後再刪除主要秘密。當您刪除複本時，會立即刪除它。

您無法直接刪除某個版本的秘密。反之，您可以使用 AWS CLI 或 AWS SDK 從版本中移除所有預備標籤。這會將版本標示為已棄用，然後 Secrets Manager 可在背景自動刪除該版本。

如果您不知道應用程式是否仍在使用秘密，則可以建立 Amazon CloudWatch 警示，來提醒您任何在復原時段期間嘗試存取秘密的情況。如需詳細資訊，請參閱 監控何時存取排程刪除的 AWS Secrets Manager 秘密。

若要刪除秘密，您必須擁有 `secretsmanager:ListSecrets` 和 `secretsmanager:DeleteSecret` 許可。

刪除機密時，Secrets Manager 會產生 CloudTrail 日誌項目。如需詳細資訊，請參閱the section called “使用 記錄 AWS CloudTrail ”。

若要刪除秘密 (主控台)

1. 前往以下位置開啟 Secrets Manager 主控台：https://console.aws.amazon.com/secretsmanager/。
2. 在秘密清單中，選擇您要刪除的秘密。
3. 在 Secrets details (秘密詳細資訊) 區段，選擇 Actions (動作)，然後選擇 Delete description (刪除描述)。
4. 在 Disable secret and schedule deletion (停用秘密和排程刪除) 對話方塊中，於 Waiting period (等待期) 下，輸入永久刪除前要等待的天數。Secrets Manager 會連接稱為 `DeletionDate` 的欄位，並將欄位設為目前的日期和時間，加上復原時段的指定天數。
5. 選擇 Schedule deletion (排定刪除)。

若要檢視已刪除的秘密

1. 前往以下位置開啟機密管理員控制台：https://console.aws.amazon.com/secretsmanager/。

2. 在 Secrets (秘密) 頁面上，選擇 Preferences (偏好設定) ( )。
3. 在「偏好設定」對話方塊中，選取顯示排程刪除的機密，然後選擇儲存。

若要刪除複本秘密

1. 前往以下位置開啟機密管理員控制台：https://console.aws.amazon.com/secretsmanager/。
2. 選擇主要秘密。
3. 在 Replicate Secret (複寫秘密) 區段中，選擇複本秘密。
4. 從 Actions (動作) 選單中，選擇 Delete Replica (刪除複本)。

AWS CLI

Example 刪除秘密

下列 `delete-secret` 範例會刪除機密。您可以在 DeletionDate 回應欄位中的日期和時間之前使用 `restore-secret` 復原機密。若要刪除複寫至其他區域的機密，請先使用 `remove-regions-from-replication` 移除複本，然後呼叫 `delete-secret`。

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

Example 立即刪除機密

下列 `delete-secret` 範例會在沒有復原時段的情況下立即刪除機密。您無法復原此機密。

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

Example 刪除複本秘密

下列 `remove-regions-from-replication` 範例會刪除 eu-west-3 中的複本機密。若要刪除複寫至其他區域的主要機密，請先刪除複本，然後呼叫 `delete-secret`。

```
aws secretsmanager remove-regions-from-replication \
    --secret-id MyTestSecret \
```

```
    --remove-replica-regions eu-west-3
```

AWS 開發套件

若要刪除秘密，請使用 DeleteSecret 命令。若要刪除秘密的某個版本，請使用 UpdateSecretVersionStage 命令。若要刪除複本，請使用 StopReplicationToReplica 命令。如需詳細資訊，請參閱the section called “AWS SDKs”。

還原 AWS Secrets Manager 秘密

Secrets Manager 會將排定刪除的秘密視為已棄用且您不再能夠直接存取。復原時段過後，Secrets Manager 會永久刪除秘密。一旦 Secrets Manager 刪除秘密，您就無法將其復原。復原時段結束之前，您可以復原秘密讓它再度成為可存取。這會移除 DeletionDate 欄位，因而取消排定的永久刪除。

若要在主控台中還原秘密和中繼資料，您必須擁有 secretsmanager:ListSecrets 和 secretsmanager:RestoreSecret 許可。

還原機密時，Secrets Manager 會產生 CloudTrail 日誌項目。如需詳細資訊，請參閱the section called “使用 記錄 AWS CloudTrail ”。

若要還原秘密 (主控台)

1. 前往以下位置開啟 Secrets Manager 主控台：https://console.aws.amazon.com/secretsmanager/。
2. 在秘密清單中，選擇您想要還原的秘密。

   如果秘密清單中沒有出現刪除的秘密，請選擇 Preferences (偏好設定) ( )。
   在「偏好設定」對話方塊中，選取顯示排程刪除的機密，然後選擇儲存。
3. 在 Secret details (秘密詳細資訊) 頁面中，選擇 Cancel deletion (取消刪除)。
4. 在 Cancel secret deletion (取消秘密刪除) 對話方塊中，選擇 Cancel deletion (取消刪除)。

AWS CLI

Example 還原先前刪除的機密

下列 restore-secret 範例會還原先前排程刪除的機密。

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

AWS 開發套件

若要還原標記為刪除的秘密，請使用 `RestoreSecret` 命令。如需詳細資訊，請參閱the section called “AWS SDKs”。

在 中標記秘密 AWS Secrets Manager

在 中 AWS Secrets Manager，您可以使用標籤將中繼資料指派給秘密。標籤是您為秘密定義的鍵/值對。標籤可協助您管理 AWS 資源和組織資料，包括帳單資訊。

使用標籤，您可以：

- 管理、搜尋和篩選帳戶中 AWS 的秘密和其他資源
- 根據連接的標籤控制對秘密的存取
- 追蹤和分類與特定秘密或專案相關的費用

如需使用標籤控制存取的詳細資訊，請參閱 the section called “使用標籤控制對秘密的存取”。

若要了解成本分配標籤，請參閱 AWS Billing 《 使用者指南》中的使用 AWS 成本分配標籤。

如需有關標籤配額和命名限制的資訊，請參閱《 AWS 一般參考指南》中的標記的服務配額。標籤會區分大小寫。

標記或取消標記機密時，Secrets Manager 會產生 CloudTrail 日誌項目。如需詳細資訊，請參閱the section called “使用 記錄 AWS CloudTrail ”。

> **Tip**
>
> 在您的所有 AWS 資源中使用一致的標記機制。如需最佳實務，請參閱標記最佳實務白皮書。

檢閱標籤基本概念

您可以在 主控台中依標籤找到秘密 AWS CLI，而 SDKs。 AWS 也提供 資源群組工具來建立自訂主控台，以根據資源的標籤來合併和整理您的資源。若要尋找具有特定標籤的秘密，請參閱 the section called “查找秘密”。

您可以使用 Secrets Manager 主控台 AWS CLI或 Secrets Manager API 來：

- 使用標籤建立秘密
- 將標籤新增至秘密
- 列出秘密的標籤

- 從機密移除標籤

您可以使用標籤來分類秘密。例如，您可以依用途、擁有者或環境來分類秘密。由於您定義了每個標籤的金鑰和值，您可以建立一組自訂的類別，以符合您的特定需求。以下是數個標籤的範例：

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

使用標記追蹤成本

您可以使用標籤來分類和追蹤您的 AWS 成本。當您將標籤套用至 AWS 資源時，包括秘密， AWS 成本分配報告會包含依標籤彙總的用量和成本。您可以套用代表業務類別 (例如成本中心、應用程式名稱或擁有者) 的標籤，來整理多個服務中的成本。如需詳細資訊，請參閱《AWS Billing 使用者指南》中的將成本分配標籤用於自訂帳單報告。

了解標籤限制

下列限制適用於標籤。

基本限制

- 每個資源 （秘密） 的標籤數目上限為 50。
- 標籤金鑰與值皆區分大小寫。
- 您無法變更或編輯已刪除秘密的標籤。

標籤鍵限制

- 每個標籤鍵都必須是唯一的。如果您新增具有已使用金鑰的標籤，則新的標籤會覆寫現有金鑰值對。
- 您無法以 啟動標籤金鑰，`aws:`因為此字首保留給 使用 AWS。 會代表您 AWS 建立以此字首開頭的標籤，但您無法編輯或刪除它們。
- 標籤鍵的長度必須介於 1 到 128 個 Unicode 字元之間。
- 標籤鍵必須包含下列字元：Unicode 字母、數字、空格以及下列特殊字元：`_ . / = + - @`。

標籤值限制

- 標籤值的長度必須介於 0 到 255 個 Unicode 字元之間。
- 標籤值可以空白。否則，它們必須包含下列字元：Unicode 字母、數字、空格以及下列任何特殊字元：_ . / = + - @。

使用 Secrets Manager 主控台標記秘密

您可以使用 Secrets Manager 主控台來管理秘密的標籤。

若要存取標記功能，請執行下列動作：

1. 開啟 Secrets Manager 主控台。
2. 在導覽列中，選擇您偏好的區域。
3. 在秘密頁面上，選取秘密。

檢視秘密的標籤

- 在秘密詳細資訊頁面上，選擇標籤索引標籤。

使用標籤建立秘密

- 請遵循 建立秘密 中的步驟。

新增或編輯秘密的標籤

1. 在秘密詳細資訊頁面上，選擇標籤索引標籤，然後選擇編輯標籤。
2. 在金鑰欄位中輸入標籤金鑰。或者，在值欄位中輸入標籤值。
3. 選擇儲存。新的或更新的標籤會出現在標籤清單中。

> **Note**
>
> 如果未啟用儲存按鈕，則標籤索引鍵或值可能不符合標籤限制。如需詳細資訊，請參閱了解標籤限制。

從秘密中移除標籤

1. 在秘密詳細資訊頁面上，選擇標籤索引標籤，然後選擇您要移除之標籤旁的移除圖示。
2. 選擇儲存以確認移除，或選擇復原以取消。

使用 標記秘密 AWS CLI

AWS CLI 範例

Example 將標籤新增至機密

下列 tag-resource 範例顯示如何使用速記語法連接標籤。

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

Example 將多個標籤新增至機密

下列 tag-resource 範例會將兩個金鑰值標籤連接至機密。

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

Example 從機密移除標籤

下列 untag-resource 範例會從機密中移除兩個標籤。對於每個標籤，金鑰和值都會移除。

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

使用 Secrets Manager API 標記秘密

您可以使用 Secrets Manager API 新增、列出和移除標籤。如需範例，請參閱下列文件：

- ListSecrets：使用 ListSecrets 來檢視套用至秘密的標籤

- TagResource：將標籤新增至秘密
- 取消標記：從秘密中移除標籤

使用 Secrets Manager AWS SDK 標記秘密

若要變更秘密的標籤，請使用下列 API 操作：

- ListSecrets：使用 `ListSecrets`來檢視套用至秘密的標籤
- `TagResource`：將標籤新增至秘密
- `UntagResource`：從秘密中移除標籤

如需使用 SDK 的詳細資訊，請參閱 the section called “AWS SDKs”。

跨區域複寫 AWS Secrets Manager 秘密

您可以在多個 中複寫秘密 AWS 區域 ，以支援分散在這些區域中的應用程式，以滿足區域存取和低延遲需求。如果您稍後需要，您可以將複本秘密提升為獨立，然後將其單獨設定為複寫。機密管理員會複寫已加密的機密資料和中繼資料，例如跨越指定區域的標籤和資源政策。

複寫秘密的 ARN 與主要秘密的 ARN 基本相同，唯一差異在於 Region 部分，示例如下：

- 主要機密：`arn:aws:secretsmanager:Region1:123456789012:secret:MySecret-a1b2c3`
- 複本秘密：`arn:aws:secretsmanager:Region2:123456789012:secret:MySecret-a1b2c3`

如需複本機密的定價資訊，請參閱 AWS Secrets Manager 定價。

當您針對可複製到其他區域的來源資料庫儲存資料庫憑證時，機密會包含來源資料庫的連線資訊。如果隨後複製機密，則複本是來源機密的副本，並包含相同的連線資訊。您可以將其他金鑰/值對新增到區域連線資訊的機密。

如果您對主要機密開啟輪換，則機密管理員會在主要區域中輪換機密，而新的機密值會傳播至所有相關聯的複本機密。您不必單獨管理所有複本機密的輪換。

您可以將秘密複寫到所有已啟用 AWS 的區域。不過，如果您在 AWS GovCloud (US) 或 中國 AWS 區域等特殊區域使用 Secrets Manager，您只能在這些特殊 AWS 區域內設定秘密和複本。您無法將已啟用 區域中的秘密複寫 AWS 至特定區域，或將秘密從特定區域複寫至商業區域。

在您可以將機密複寫到另一個區域之前，必須先啟用該區域。如需詳細資訊，請參閱管理 AWS 區域。

透過調用儲存機密之區域中的機密管理員端點，您可以在無需複製的情況下跨多個區域使用機密。如需端點清單，請參閱 the section called “Secrets Manager 端點”。若要使用複寫來改善工作負載的彈性，請參閱開啟災難復原 (DR) 架構 AWS，第 I 部分：雲端中的復原策略。

複寫機密時，Secrets Manager 會產生 CloudTrail 日誌項目。如需詳細資訊，請參閱the section called “使用 記錄 AWS CloudTrail ”。

若要將機密複寫到其他區域 (控制台)

1. 前往以下位置開啟機密管理員控制台：https://console.aws.amazon.com/secretsmanager/。
2. 從秘密清單中選擇秘密。

3. 在秘密詳細資訊頁面的複寫分頁上，執行以下其中一項操作：
   - 如果尚未複寫您的機密，請選擇 Replicate secret (複寫機密)。
   - 如果已複寫您的機密，請在 Replicate secret (複寫機密) 區段中選擇 Add Region (新增區域)。
4. 在 Add replica regions (新增複寫區域) 對話方塊中，執行下列操作：
   a. 針對 AWS Region (AWS 區域)，選擇您要複寫機密的目標 Region (區域)。
   b. (選用) 針對 Encryption key (加密金鑰)，選擇要用於將機密加密的 KMS 金鑰。金鑰必須在複本區域中。
   c. (選用) 若要新增另一個區域，請選擇 Add more regions (新增其他區域)。
   d. 選擇 Replicate (複寫)。

   返回機密詳細資訊頁面。在 Replicate Secret (複寫機密) 區段中，會顯示每個區域的 Replication Status (複寫狀態)。

AWS CLI

Example 將機密複寫至其他區域

下列 `replicate-secret-to-regions` 範例會將機密複寫至 eu-west-3。複本會使用 AWS 受管金鑰 加密aws/secretsmanager。

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

Example 建立秘密並進行複寫

下列範例會建立秘密並將其複寫至 eu-west-3。複本會使用 加密 AWS 受管金鑰 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 開發套件

若要複寫機密，請使用 `ReplicateSecretToRegions` 命令。如需詳細資訊，請參閱the section called “AWS SDKs”。

在 中將複本秘密提升為獨立秘密 AWS Secrets Manager

複本秘密是從另一個主要 中複寫的秘密 AWS 區域。它具有與主要秘密相同的秘密值和中繼資料，但可以使用不同的 KMS 金鑰加密。複本秘密不能獨立於其主要秘密更新，但其加密金鑰除外。升級複本秘密會中斷複本秘密與主要秘密的連線，並將複本秘密設為獨立秘密。主要秘密的任何變更均不會再複寫到獨立秘密。

您可以在主要秘密無法使用時，將複本秘密提升為獨立的秘密，以此作為災難復原解決方案。或者，如果您要開啟複本的輪換功能，您可以將複本提升為獨立的秘密。

如果要提升複本，請務必更新對應的應用程式，以便使用獨立的秘密。

提升機密時，Secrets Manager 會產生 CloudTrail 日誌項目。如需詳細資訊，請參閱the section called “使用 記錄 AWS CloudTrail ”。

若要提升複本秘密 (主控台)

1. 登入 Secrets Manager，網址為 https://console.aws.amazon.com/secretsmanager/。
2. 導覽至複本區域。
3. 在 Secrets (機密) 頁面上，選擇複本機密。
4. 在複本機密詳細資訊頁面上，選擇 Promote to standalone secret (提升為獨立機密)。
5. 在 Promote replica to standalone secret (將複本提升為獨立機密) 對話方塊中，輸入 Region (區域)，然後選擇 Promote replica (提升複本)。

AWS CLI

Example 將複本機密提升為主要機密

下列 `stop-replication-to-replica` 範例會移除複本機密至主要機密之間的連結。複本機密會提升為複本區域中的主要機密。您必須從複本區域內呼叫 `stop-replication-to-replica`。

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

AWS 開發套件

若要將複本秘密提升為獨立的秘密，請使用 StopReplicationToReplica 命令。您必須從複本秘密區域呼叫此命令。如需詳細資訊，請參閱the section called “AWS SDKs”。

防止 AWS Secrets Manager 複寫

由於秘密可以使用 ReplicateSecretToRegions或使用 建立時複寫CreateSecret，如果您想要防止使用者複寫秘密，我們建議您防止包含 AddReplicaRegions 參數的動作。您可以在許可政策中使用Condition陳述式，僅允許不新增複本區域的動作。如需您可以使用的條件陳述式，請參閱下列政策範例。

Example 防止複寫許可

下列政策範例示範如何允許未新增複本區域的所有動作。這可防止使用者透過 ReplicateSecretToRegions和 複寫秘密CreateSecret。

JSON

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

Example 僅允許對特定區域的複寫許可

下列政策說明如何允許下列所有項目：

- 建立無需複寫的秘密

- 僅在美國和加拿大使用複寫功能建立秘密至 區域
- 僅在美國和加拿大將秘密複寫至 區域

JSON

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

對 AWS Secrets Manager 複寫進行故障診斷

AWS Secrets Manager 複寫可能會因為各種原因而失敗。若要檢查秘密無法複寫的原因，您可以執行下列其中一項操作：

- 呼叫 `DescribeSecret` API 操作
- 檢閱 AWS CloudTrail 事件

當複寫失敗時：

- 如果沒有可用的秘密版本，Secrets Manager 會從複本區域移除秘密。

- 如果成功複寫秘密版本，它們會保留在複本區域中，直到您使用 `RemoveRegionsFromReplication` API 操作明確移除它們為止。

下列各節說明複寫失敗的一些常見原因。

選取的區域中存在具有相同名稱的秘密。

若要解決此問題，您可以覆寫複本區域中的重複名稱機密。重試複寫，接著在重試複寫對話方塊中，選擇覆寫。

KMS 金鑰上沒有可用的許可來完成複寫

Secrets Manager 會先解密秘密，然後再使用複本區域中的新 KMS 金鑰重新加密。如果您沒有主要區域中的加密金鑰的 `kms:Decrypt` 許可，則會遇到此錯誤。要使用 KMS 密鑰以外的密鑰加密複製的秘密`aws/secretsmanager`，你需要 `kms:GenerateDataKey` 和 `kms:Encrypt` 密鑰。請參閱 the section called “KMS 金鑰的許可”。

KMS 金鑰已停用或找不到 KMS 金鑰

如果主要區域中的加密金鑰已停用或已刪除，Secret Manager 就無法複寫秘密。如果秘密具有使用已停用或已刪除的加密金鑰進行加密的自訂標記版本，則即便您已變更加密金鑰，仍可能發生此錯誤。如需 Secrets Manager 如何加密的相關資訊，請參閱 the section called “秘密加密和解密”。若要解決這個問題，您可以重新建立秘密版本，讓 Secrets Manager 使用目前的加密金鑰進行加密。如需詳細資訊，請參閱變更秘密的加密金鑰。然後重試複寫。

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

尚未啟用要進行複寫的區域

如需如何啟用區域的相關資訊，請參閱《AWS 帳戶管理參考指南》中的管理 AWS 區域。

從 取得秘密 AWS Secrets Manager

擷取機密時，Secrets Manager 會產生 CloudTrail 日誌項目。如需詳細資訊，請參閱the section called “使用 記錄 AWS CloudTrail ”。

您可以使用下列方式擷取秘密值：

- 使用 Java 取得 Secrets Manager 秘密值
- 使用 Python 取得 Secrets Manager 秘密值
- 使用 取得 Secrets Manager 秘密值 .NET
- 使用 Go 取得 Secrets Manager 秘密值
- 使用 Rust 取得 Secrets Manager 秘密值
- 在 Amazon Elastic Kubernetes Service 中使用 AWS Secrets Manager 秘密
- 在 AWS Lambda 函數中使用 AWS Secrets Manager 秘密
- 使用 AWS Secrets Manager 代理程式
- 使用 C++ AWS SDK 取得 Secrets Manager 秘密值
- 使用 JavaScript AWS 開發套件取得 Secrets Manager 秘密值
- 使用 Kotlin AWS SDK 取得 Secrets Manager 秘密值
- 使用 PHP AWS SDK 取得 Secrets Manager 秘密值
- 使用 Ruby AWS SDK 取得 Secrets Manager 秘密值
- 使用 取得秘密值 AWS CLI
- 使用 AWS 主控台取得秘密值
- 在 中使用 AWS Secrets Manager 秘密 AWS Batch
- 在 資源中 AWS CloudFormation 取得 AWS Secrets Manager 秘密
- 在 GitHub 任務中使用 AWS Secrets Manager 秘密
- 在 中使用 AWS Secrets Manager 秘密 AWS IoT Greengrass
- 在參數存放區中使用 AWS Secrets Manager 秘密

使用 Java 取得 Secrets Manager 秘密值

在應用程式中，您可以呼叫 `GetSecretValue`或任何 AWS SDKs`BatchGetSecretValue`中的 來擷取秘密。不過，建議您使用用戶端快取來快取您的秘密值。快取秘密可提高速度並降低成本。

若要使用秘密中的登入資料連線到資料庫，您可以使用 Secrets Manager SQL Connection 驅動程式，以包裝基本 JDBC 驅動程式。這也使用用戶端快取，因此可以降低呼叫 Secrets Manager APIs的成本。

主題

- 使用 Java 搭配用戶端快取取得 Secrets Manager 秘密值
- 使用 JDBC 在 AWS Secrets Manager 秘密中使用登入資料連線到 SQL 資料庫
- 使用 Java AWS 開發套件取得 Secrets Manager 秘密值

使用 Java 搭配用戶端快取取得 Secrets Manager 秘密值

擷取秘密時，您可以使用 Secrets Manager Java 型快取元件進行快取以供將來使用。擷取快取的秘密比從 Secrets Manager 中擷取要快。由於呼叫 Secrets Manager API 需要花費成本，因此使用快取可以降低成本。如需您可以擷取機密的所有方法，請參閱 取得秘密。

快取政策是「最近最少使用的 (LRU)」，因此當快取必須丟棄秘密時，其會丟棄最近最少使用的秘密。預設情況下，快取每小時重新整理一次秘密。您可以設定在快取中重新整理機密的頻率，並可以掛接到機密擷取以新增更多功能。

一旦釋放快取參考，快取不會強制執行垃圾回收。快取實作不包括快取失效。快取實作著重於快取本身，而不是強化或著重於安全性。如果您需要額外的安全性，例如加密快取中的項目，請使用提供的介面和抽象方法。

若要使用元件，您必須擁有下列項目：

- Java 8 或更高版本的開發環境。請參閱 Oracle 網站上的 Java SE 下載。

若要下載開放程式碼，請參閱 Secrets Manager Java 型快取用戶端元件 (在 GitHub 上)。

要將元件新增至您的專案中，請在 Maven pom.xml 檔案中包含以下相依性。如需 Maven 的詳細資訊，請參閱 Apache Maven 專案網站上的入門指南。

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

必要許可：

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

如需詳細資訊，請參閱許可參考。

參考資料

- SecretCache
- SecretCacheConfiguration
- SecretCacheHook

Example 擷取秘密

下列程式碼範例顯示擷取秘密字串的 Lambda 函數。它遵循將函數處理常式之外的快取執行個體化的最佳實務，所以如果您再次呼叫 Lambda 函數，則不會繼續呼叫 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

從 Secrets Manager 請求的秘密記憶體內快取。您使用 the section called “getSecretString” 或 the section called “getSecretBinary” 從快取中擷取秘密。您可以透過在建構函式的 the section called “SecretCacheConfiguration” 物件中傳遞，設定快取設定。

如需包含範例的詳細資訊，請參閱 the section called “Java 搭配用戶端快取”。

建構函式

`public SecretCache()`

SecretCache 物件的預設建構函式。

`public SecretCache(AWSSecretsManagerClientBuilder builder)`

使用以提供的 `AWSSecretsManagerClientBuilder` 建立的 Secrets Manager 用戶端建構新快取。使用此建構函數自訂 Secrets Manager 用戶端，例如使用特定區域或端點。

`public SecretCache(AWSSecretsManager client)`

使用提供的 `AWSSecretsManagerClient` 建構新的秘密快取。使用此建構函數自訂 Secrets Manager 用戶端，例如使用特定區域或端點。

`public SecretCache(SecretCacheConfiguration config)`

使用提供的 `the section called “SecretCacheConfiguration”` 建構新的秘密快取。

方法

getSecretString

`public String getSecretString(final String secretId)`

從 Secrets Manager 中擷取字串秘密。傳回 `String`。

getSecretBinary

`public ByteBuffer getSecretBinary(final String secretId)`

從 Secrets Manager 中擷取二進位秘密。傳回 `ByteBuffer`。

refreshNow

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

強制重新整理快取。如果重新整理完成且沒有錯誤，則會傳回 `true`，否則會傳回 `false`。

關閉

```
public void close()
```

關閉快取。

SecretCacheConfiguration

the section called “SecretCache” 的快取組態選項，例如最大快取大小和快取秘密的存留時間 (TTL)。

建構函數

```
public SecretCacheConfiguration
```

`SecretCacheConfiguration` 物件的預設建構函式。

方法

getClient

```
public AWSSecretsManager getClient()
```

傳回 `AWSSecretsManagerClient`，快取從中擷取秘密。

setClient

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

傳回 `AWSSecretsManagerClient` 用戶端，快取從中擷取秘密。

getCacheHook

```
public SecretCacheHook getCacheHook()
```

傳回用於與快取更新掛鉤的 the section called “SecretCacheHook” 介面。

setCacheHook

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

設定用於與快取更新掛鉤的 the section called “SecretCacheHook” 介面。

getMaxCacheSize

```
public int getMaxCacheSize()
```

傳回最大快取大小。預設值為 1024 個秘密。

setMaxCacheSize

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

設定最大快取大小。預設值為 1024 個秘密。

getCacheItemTTL

```
public long getCacheItemTTL()
```

傳回快取項目的 TTL (以毫秒為單位)。當快取的秘密超過此 TTL 時，快取會從 AWSSecretsManagerClient 擷取秘密的新複本。預設值為 1 小時 (以毫秒為單位)。

當在 TTL 之後請求秘密時，快取會同步重新整理秘密。如果同步重新整理失敗，則快取會傳回過時的秘密。

setCacheItemTTL

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

設定快取項目的 TTL (以毫秒為單位)。當快取的秘密超過此 TTL 時，快取會從 AWSSecretsManagerClient 擷取秘密的新複本。預設值為 1 小時 (以毫秒為單位)。

getVersionStage

```
public String getVersionStage()
```

傳回要快取的秘密版本。如需詳細資訊，請參閱秘密版本。預設為 "AWSCURRENT"。

setVersionStage

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

設定要快取的秘密版本。如需詳細資訊，請參閱秘密版本。預設為 "AWSCURRENT"。

SecretCacheConfiguration withClient

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

設定 AWSSecretsManagerClient，以從中擷取秘密。使用新設定傳回已更新的 SecretCacheConfiguration 物件。

SecretCacheConfiguration withCacheHook

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

設定用於與記憶體內快取掛鉤的介面。使用新設定傳回已更新的 SecretCacheConfiguration 物件。

SecretCacheConfiguration withMaxCacheSize

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

設定最大快取大小。使用新設定傳回已更新的 SecretCacheConfiguration 物件。

SecretCacheConfiguration withCacheItemTTL

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

設定快取項目的 TTL (以毫秒為單位)。當快取的秘密超過此 TTL 時，快取會從 AWSSecretsManagerClient 擷取秘密的新複本。預設值為 1 小時 (以毫秒為單位)。使用新設定傳回已更新的 SecretCacheConfiguration 物件。

SecretCacheConfiguration withVersionStage

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

設定要快取的秘密版本。如需詳細資訊，請參閱秘密版本。使用新設定傳回已更新的 SecretCacheConfiguration 物件。

SecretCacheHook

用於掛接到 the section called “SecretCache”，以對存放在快取中的秘密執行動作的介面。

put

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

準備要存放在快取中的物件。

傳回要存放在快取中的物件。

get

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

從快取的物件中衍生物件。

傳回要從快取傳回的物件

使用 JDBC 在 AWS Secrets Manager 秘密中使用登入資料連線到 SQL 資料庫

在 Java 應用程式中，您可以使用 Secrets Manager SQL Connection 驅動程式，使用存放在 Secrets Manager 中的登入資料連線至 MySQL、PostgreSQL、Oracle、MSSQLServer、Db2 和 Redshift 資料庫。每個驅動程式都會包裝基本 JDBC 驅動程式，因此您可以使用 JDBC 呼叫來存取資料庫。但是，並非傳遞用於連線的使用者名稱和密碼，而是提供機密的 ID。該驅動程式調用機密管理員擷取機密值，然後使用機密中的憑證連線至資料庫。驅動程式還使用 Java 用戶端快取庫快取憑證，因此未來的連線不需要呼叫 Secrets Manager。預設快取每小時重新整理一次機密，並在輪換機密時重新整理一次。若要設定快取，請參閱 the section called “SecretCacheConfiguration”。

您可以從 GitHub 下載開放原始碼。

使用機密管理員 SQL 連線驅動程式：

- 您的應用程式必須使用 Java 8 或更高版本。
- 您的機密必須為下列之一：
  - 預期 JSON 結構中的資料庫機密。若要查看該格式，請在機密管理員控制台中檢視您的機密，然後選擇 Retrieve secret value (擷取機密值)。或者，在 中 AWS CLI呼叫 get-secret-value。
  - Amazon RDS 受管機密。對於此類型的機密，您必須在建立連線時指定端點和連接埠。
  - Amazon Redshift 受管秘密。對於此類型的機密，您必須在建立連線時指定端點和連接埠。

如果要將您的資料庫複製到其他區域，以連線到另一個區域中的複副本資料庫，請在建立連線時指定區域端點和端口。您可以將區域連線資訊作為額外的金鑰/值對、SSM 參數儲存參數或程式碼設定中儲存至機密中。

若要將驅動程式新增至您的專案中，請在 Maven 建置檔案 `pom.xml` 中，為驅動程式新增以下相依性。如需詳細資訊，請參閱 Maven Central Repository 網站上的 Secrets Manager SQL 連線庫。

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

此驅動程式使用預設憑證供應商鏈。如果您在 Amazon EKS 上執行驅動程式，則其可能會取得正在執行之節點的憑證，而不是服務帳戶角色。為瞭解決此問題，請將 com.amazonaws:aws-java-sdk-sts 的版本 1 新增至 Gradle 或 Maven 專案檔案作為相依性。

若要在 secretsmanager.properties 檔案中設定 AWS PrivateLink DNS 端點 URL 和區域：

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

若要覆寫主要區域，請設定 AWS_SECRET_JDBC_REGION 環境變數或對 secretsmanager.properties 檔案進行下列變更：

```
drivers.region = region
```

必要許可：

- secretsmanager:DescribeSecret
- secretsmanager:GetSecretValue

如需詳細資訊，請參閱許可參考。

範例：

- 建立與資料庫的連線
- 透過指定端點和連接埠來建立連線
- 使用 c3p0 連線集區建立連線
- 使用 c3p0 連線集區透過指定端點和連接埠來建立連線

建立與資料庫的連線

以下範例顯示如何使用機密中的憑證和連線資訊建立與資料庫的連線。連線後，您就可使用 JDBC 呼叫來存取資料庫。如需詳細資訊，請參閱 Java 文件網站上的 JDBC 基礎知識。

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);
```

透過指定端點和連接埠來建立連線

下列範例顯示如何使用機密中的憑證與您指定的端點和連接埠建立資料庫的連線。

Amazon RDS 受管機密不包括資料庫的端點和連接埠。若要使用 Amazon RDS 管理之機密中的主要憑證連線至資料庫，請在程式碼中指定這些憑證。

複製到其他區域的機密可以改善區域資料庫連線的延遲，但其不包含與來源機密不同的連線資訊。每個複本都是來源機密的副本。若要在機密中儲存區域連線資訊，請為區域的端點和端口資訊新增更多金鑰/值對。

連線後，您就可使用 JDBC 呼叫來存取資料庫。如需詳細資訊，請參閱 Java 文件網站上的 JDBC 基礎知識。

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 連線集區建立連線

下列範例顯示如何使用 `c3p0.properties` 檔案建立連線集區，此檔案使用驅動程式透過機密擷取憑證和連線資訊。對於 `user` 和 `jdbcUrl`，輸入機密 ID 以設定連線集區。然後，您可以從集區中擷取連線並將其用作任何其他資料庫連線。如需詳細資訊，請參閱 Java 文件網站上的 JDBC 基礎知識。

如需 c3p0 的詳細資訊，請參閱 Machinery For Change 網站上的 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 連線集區透過指定端點和連接埠來建立連線

下列範例示範如何使用 驅動程式，在您指定的端點和連接埠的秘密中擷取登入資料`c3p0.properties`的檔案，來建立連線集區。然後，您可以從集區中擷取連線並將其用作任何其他資料庫連線。如需詳細資訊，請參閱 Java 文件網站上的 JDBC 基礎知識。

Amazon RDS 受管機密不包括資料庫的端點和連接埠。若要使用 Amazon RDS 管理之機密中的主要憑證連線至資料庫，請在程式碼中指定這些憑證。

複製到其他區域的機密可以改善區域資料庫連線的延遲，但其不包含與來源機密不同的連線資訊。每個複本都是來源機密的副本。若要在機密中儲存區域連線資訊，請為區域的端點和端口資訊新增更多金鑰/值對。

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 開發套件取得 Secrets Manager 秘密值

在應用程式中，您可以呼叫 GetSecretValue或任何 AWS SDKsBatchGetSecretValue中的 來擷取秘密。不過，建議您使用用戶端快取來快取您的秘密值。快取秘密可提高速度並降低成本。

- 如果將資料庫憑證存放在秘密中，則請使用 Secrets Manager SQL 連線驅動程式，以使用秘密中的憑證連線至資料庫。
- 對於其他類型的秘密，請使用 Secrets Manager Java 型快取元件，或直接透過 GetSecretValue或 呼叫 SDKBatchGetSecretValue。

下列程式碼範例示範如何使用 GetSecretValue。

必要許可：secretsmanager:GetSecretValue

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

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
```

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

                Usage:
                    <secretName>\s

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

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

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

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

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

            GetSecretValueResponse valueResponse =
 secretsClient.getSecretValue(valueRequest);
            String secret = valueResponse.secretString();
```

```
            System.out.println(secret);

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

使用 Python 取得 Secrets Manager 秘密值

在應用程式中，您可以呼叫 `GetSecretValue`或任何 AWS SDKs`BatchGetSecretValue`中的 來擷取秘密。不過，建議您使用用戶端快取來快取您的秘密值。快取秘密可提高速度並降低成本。

主題

- 使用 Python 搭配用戶端快取取得 Secrets Manager 秘密值
- 使用 Python AWS SDK 取得 Secrets Manager 秘密值
- 使用 Python AWS SDK 取得一批 Secrets Manager 秘密值

使用 Python 搭配用戶端快取取得 Secrets Manager 秘密值

擷取秘密時，您可以使用 Secrets Manager Python 型快取元件進行快取以供將來使用。擷取快取的秘密比從 Secrets Manager 中擷取要快。由於呼叫 Secrets Manager API 需要花費成本，因此使用快取可以降低成本。如需您可以擷取機密的所有方法，請參閱 取得秘密。

快取政策是「最近最少使用的 (LRU)」，因此當快取必須丟棄秘密時，其會丟棄最近最少使用的秘密。預設情況下，快取每小時重新整理一次秘密。您可以設定在快取中重新整理機密的頻率，並可以掛接到機密擷取以新增更多功能。

一旦釋放快取參考，快取不會強制執行垃圾回收。快取實作不包括快取失效。快取實作著重於快取本身，而不是強化或著重於安全性。如果您需要額外的安全性，例如加密快取中的項目，請使用提供的介面和抽象方法。

若要使用元件，您必須擁有下列項目：

- Python 3.6 或更高版本。
- botocore 1.12 或更高版本。請參閱 AWS SDK for Python 和 Botocore。
- setuptools_scm 3.2 或更高版本。請參閱 https://pypi.org/project/setuptools-scm/。

若要下載開放程式碼，請參閱 Secrets Manager Python 型快取用戶端元件 (在 GitHub 上)。

若要安裝元件，請使用下列命令。

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

必要許可：

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

如需詳細資訊，請參閱許可參考。

參考資料

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

Example 擷取秘密

以下範例顯示如何為名為 *mysecret* 的秘密取得秘密值。

```
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

從 Secrets Manager 擷取的秘密記憶體內快取。您使用 the section called “get_secret_string” 或 the section called “get_secret_binary” 從快取中擷取秘密。您可以透過在建構函式的 the section called “SecretCacheConfig” 物件中傳遞，設定快取設定。

如需包含範例的詳細資訊，請參閱 the section called “Python 搭配用戶端快取”。

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

可用的方法如下：

- get_secret_string
- get_secret_binary

get_secret_string

擷取秘密字串值。

請求語法

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

參數

- `secret_id` (字串)：【必要】 秘密的名稱或 ARN。
- `version_stage` (字串)：您要擷取的秘密版本。如需詳細資訊，請參閱秘密版本。預設值為 'AWSCURRENT'。

傳回類型

string

get_secret_binary

擷取秘密二進位值。

請求語法

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

參數

- secret_id (字串)：【必要】 秘密的名稱或 ARN。
- version_stage (字串)：您要擷取的秘密版本。如需詳細資訊，請參閱秘密版本。預設值為 'AWSCURRENT'。

傳回類型

base64-encoded 字串

SecretCacheConfig

the section called “SecretCache” 的快取組態選項，例如最大快取大小和快取秘密的存留時間 (TTL)。

參數

max_cache_size (int)

最大快取大小。預設值為 1024 個秘密。

exception_retry_delay_base (int)

重試請求前遇到異常之後等待的秒數。預設值為 1。

exception_retry_growth_factor (int)pur

用於計算失敗請求重試之間等待時間的增長係數。預設值為 2。

exception_retry_delay_max (int)

在失敗請求之間等待的時間上限 (以秒為單位)。預設值為 3600。

default_version_stage (str)

要快取的秘密版本。如需詳細資訊，請參閱秘密版本。預設為 'AWSCURRENT'。

secret_refresh_interval (int)

重新整理快取秘密資訊之間的等待秒數。預設值為 3600。

`secret_cache_hook` (SecretCacheHook)

`SecretCacheHook` 摘要類別的實作。預設值為 `None`。

SecretCacheHook

用於掛接到 the section called “SecretCache”，以對存放在快取中的秘密執行動作的介面。

可用的方法如下：

- put
- get

put

準備要存放在快取中的物件。

請求語法

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

參數

- `obj` (物件) – [必需] 包含秘密的秘密或物件。

傳回類型

object

get

從快取的物件中衍生物件。

請求語法

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

參數

- obj (物件)：【必要】 包含秘密的秘密或物件。

傳回類型

object

@InjectSecretString

此裝飾項目需要秘密 ID 字串和 the section called “SecretCache” 作為第一個和第二個引數。裝飾項目傳回秘密字串值。秘密必須包含字串。

```
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

此裝飾項目需要秘密 ID 字串和 the section called “SecretCache” 作為第一個和第二個引數。剩餘的引數將參數從包裝函數映射到秘密中的 JSON 金鑰。秘密必須包含 JSON 結構中的字串。

對於包含此 JSON 的秘密：

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

下列範例顯示如何從秘密擷取 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 取得 Secrets Manager 秘密值

在應用程式中，您可以呼叫 GetSecretValue或任何 AWS SDKsBatchGetSecretValue中的 來擷取秘密。不過，建議您使用用戶端快取來快取您的秘密值。快取秘密可提高速度並降低成本。

對於 Python 應用程式，請使用 Secrets Manager Python 型快取元件或使用 get_secret_value 或 batch_get_secret_value 直接呼叫 SDK。

下列程式碼範例示範如何使用 GetSecretValue。

必要許可：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 取得一批 Secrets Manager 秘密值

下列程式碼範例示範如何取得一批 Secrets Manager 秘密值。

必要許可：

- `secretsmanager:BatchGetSecretValue`
- `secretsmanager:GetSecretValue` 您要擷取的每個秘密的 許可。
- 如果您使用過濾器，則還必須擁有 `secretsmanager:ListSecrets`。

如需許可政策範例，請參閱 the section called “範例：擷取批次中一組秘密值的許可”。

> **Important**
>
> 如果您的 VPCE 原則拒絕擷取您要擷取之群組中個別密碼的權限，則 `BatchGetSecretValue` 不會傳回任何秘密值，而且會傳回錯誤。

```
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 取得 Secrets Manager 秘密值 .NET

在應用程式中，您可以透過呼叫 GetSecretValue或任何 AWS SDKsBatchGetSecretValue中的來擷取秘密。不過，建議您使用用戶端快取來快取您的秘密值。快取秘密可提高速度並降低成本。

主題

- 使用具有用戶端快取的 .NET 取得 Secrets Manager 秘密值
- 使用 取得 Secrets Manager 秘密值 適用於 .NET 的 SDK

使用具有用戶端快取的 .NET 取得 Secrets Manager 秘密值

擷取秘密時，您可以使用 Secrets Manager .NET 型快取元件進行快取以供將來使用。擷取快取的秘密比從 Secrets Manager 中擷取要快。由於呼叫 Secrets Manager API 需要花費成本，因此使用快取可以降低成本。如需您可以擷取機密的所有方法，請參閱 取得秘密。

快取政策是「最近最少使用的 (LRU)」，因此當快取必須丟棄秘密時，其會丟棄最近最少使用的秘密。預設情況下，快取每小時重新整理一次秘密。您可以設定在快取中重新整理機密的頻率，並可以掛接到機密擷取以新增更多功能。

一旦釋放快取參考，快取不會強制執行垃圾回收。快取實作不包括快取失效。快取實作著重於快取本身，而不是強化或著重於安全性。如果您需要額外的安全性，例如加密快取中的項目，請使用提供的介面和抽象方法。

若要使用元件，您必須擁有下列項目：

- .NET Framework 4.6.2 或更新版本，或 .NET Standard 2.0 或更新版本。請參閱 Microsoft 網站上的下載 .NET。
- 適用於 .NET 的 AWS SDK。請參閱 the section called “AWS SDKs”。

若要下載開放程式碼，請參閱 .NET 快取用戶端 (在 GitHub 上)。

若要使用快取，請首先讓其成為執行個體，然後使用 `GetSecretString` 或 `GetSecretBinary` 擷取。在連續擷取時，快取會傳回秘密的快取複本。

若要取得快取套件

- 執行以下任意一項：
  - 在您的專案目錄中，執行下列 .NET CLI 命令。

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

  - 將以下套件參考新增至您的 `.csproj` 檔案中。

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

必要許可：

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

如需詳細資訊，請參閱許可參考。

參考資料

- SecretsManagerCache
- SecretCacheConfiguration
- ISecretCacheHook

Example 擷取秘密

下列程式碼範例顯示擷取名為 *MySecret* 之秘密的方式。

```
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 設定存留時間 (TTL) 快取重新整理持續時間

下列程式碼範例說明擷取名為 *MySecret* 的秘密，並將 TTL 快取重新整理持續時間設定為 24 小時的方式。

```
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

從 Secrets Manager 請求的秘密記憶體內快取。您使用 the section called “GetSecretString” 或 the section called “GetSecretBinary” 從快取中擷取秘密。您可以透過在建構函式的 the section called “SecretCacheConfiguration” 物件中傳遞，設定快取設定。

如需包含範例的詳細資訊，請參閱 the section called “具有用戶端快取的 .NET”。

建構函式

`public SecretsManagerCache()`

SecretsManagerCache 物件的預設建構函式。

`public SecretsManagerCache(IAmazonSecretsManager secretsManager)`

使用以提供的 AmazonSecretsManagerClient 建立的 Secrets Manager 用戶端建構新快取。使用此建構函式可以自訂 Secrets Manager 用戶端，例如使用特定區域或端點。

參數

secretsManager

從中擷取秘密的 AmazonSecretsManagerClient。

`public SecretsManagerCache(SecretCacheConfiguration config)`

使用提供的 the section called “SecretCacheConfiguration” 建構新的秘密快取。使用此建構函式設定快取，例如要快取的秘密數量及其重新整理的頻率。

參數

config

包含快取組態資訊的 the section called “SecretCacheConfiguration”。

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

使用以提供的 AmazonSecretsManagerClient 和 the section called “SecretCacheConfiguration” 建立的 Secrets Manager 用戶端建構新快取。使用此建構函式可以自訂 SSecrets Manager 用戶端，例如，使用特定區域或端點並設定快取，例如要快取的秘密數量及其重新整理的頻率。

參數

secretsManager

從中擷取秘密的 AmazonSecretsManagerClient。

config

包含快取組態資訊的 the section called “SecretCacheConfiguration”。

方法

GetSecretString

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

從 Secrets Manager 中擷取字串秘密。

參數

secretId

要擷取之秘密的 ARN 或名稱。

GetSecretBinary

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

從 Secrets Manager 中擷取二進位秘密。

參數

secretId

要擷取之秘密的 ARN 或名稱。

RefreshNowAsync

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

請求 Secrets Manager 的秘密值，並使用任何變更來更新快取。如果沒有現有的快取項目，則請建立一個新的。如果重新整理成功，則會傳回 `true`。

參數

secretId

要擷取之秘密的 ARN 或名稱。

GetCachedSecret

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

如果存在於快取中，則傳回指定秘密的快取項目。否則，從 Secrets Manager 中擷取秘密並建立新的快取項目。

參數

secretId

要擷取之秘密的 ARN 或名稱。

SecretCacheConfiguration

the section called “SecretsManagerCache” 的快取組態選項，例如最大快取大小和快取秘密的存留時間 (TTL)。

屬性

CacheItemTTL

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

快取項目的 TTL (以毫秒為單位)。預設值為 3600000 毫秒或 1 小時。上限為 4294967295 毫秒，大約為 49.7 天。

MaxCacheSize

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

最大快取大小。預設值為 1024 個秘密。最多 65,535 個。

VersionStage

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

要快取的秘密版本。如需詳細資訊，請參閱秘密版本。預設為 "AWSCURRENT"。

用戶端

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

從中擷取秘密的 AmazonSecretsManagerClient。如果是 null，則快取會執行個體化一個新的用戶端。預設值為 null。

CacheHook

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

the section called “ISecretCacheHook”。

ISecretCacheHook

用於掛接到 the section called “SecretsManagerCache”，以對存放在快取中的秘密執行動作的介面。

方法

放置

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

準備要存放在快取中的物件。

傳回要存放在快取中的物件。

取得

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

從快取的物件中衍生物件。

傳回要從快取傳回的物件

使用 取得 Secrets Manager 秘密值 適用於 .NET 的 SDK

在應用程式中，您可以透過呼叫 GetSecretValue或任何 AWS SDKsBatchGetSecretValue中的來擷取秘密。不過，建議您使用用戶端快取來快取您的秘密值。快取秘密可提高速度並降低成本。

對於 .NET 應用程式，請使用 Secrets Manager .NET 型快取元件或使用 GetSecretValue 或 BatchGetSecretValue 直接呼叫 SDK。

下列程式碼範例示範如何使用 GetSecretValue。

必要許可：secretsmanager:GetSecretValue

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

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

        IAmazonSecretsManager client = new AmazonSecretsManagerClient();

        var response = await GetSecretAsync(client, secretName);

        if (response is not null)
        {
```

```
                secret = DecodeString(response);

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

        /// <summary>
        /// Retrieves the secret value given the name of the secret to
        /// retrieve.
        /// </summary>
        /// <param name="client">The client object used to retrieve the secret
        /// value for the given secret name.</param>
        /// <param name="secretName">The name of the secret value to retrieve.</param>
        /// <returns>The 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 取得 Secrets Manager 秘密值

在應用程式中，您可以呼叫 GetSecretValue或任何 AWS SDKsBatchGetSecretValue中的 來擷取秘密。不過，建議您使用用戶端快取來快取您的秘密值。快取秘密可提高速度並降低成本。

主題

- 使用 Go 搭配用戶端快取取得 Secrets Manager 秘密值
- 使用 Go AWS 開發套件取得 Secrets Manager 秘密值

使用 Go 搭配用戶端快取取得 Secrets Manager 秘密值

擷取秘密時，您可以使用 Secrets Manager Go 型快取元件進行快取以供將來使用。擷取快取的秘密比從 Secrets Manager 中擷取要快。由於呼叫 Secrets Manager API 需要花費成本，因此使用快取可以降低成本。如需您可以擷取機密的所有方法，請參閱 取得秘密。

快取政策是「最近最少使用的 (LRU)」，因此當快取必須丟棄秘密時，其會丟棄最近最少使用的秘密。預設情況下，快取每小時重新整理一次秘密。您可以設定在快取中重新整理機密的頻率，並可以掛接到機密擷取以新增更多功能。

一旦釋放快取參考，快取不會強制執行垃圾回收。快取實作不包括快取失效。快取實作著重於快取本身，而不是強化或著重於安全性。如果您需要額外的安全性，例如加密快取中的項目，請使用提供的介面和抽象方法。

若要使用元件，您必須擁有下列項目：

- AWS 適用於 Go 的 SDK。請參閱 the section called “AWS SDKs”。

若要下載開放程式碼，請參閱 Secrets Manager Go 快取用戶端 (在 GitHub 上)。

若要設定 Go 開發環境，請參閱 Go 程式設計語言網站上的 Golang 入門。

必要許可：

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

如需詳細資訊，請參閱許可參考。

參考資料

- 輸入 Cache
- 輸入 CacheConfig
- 輸入 CacheHook

Example 擷取秘密

下列程式碼範例顯示擷取秘密的 Lambda 函數。

```
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)
}
```

輸入 Cache

從 Secrets Manager 請求的秘密記憶體內快取。您使用 the section called “GetSecretString” 或 the section called “GetSecretBinary” 從快取中擷取秘密。

下列範例顯示如何設定快取設定。

```
// 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 },
)
```

如需包含範例的詳細資訊，請參閱 the section called “使用用戶端快取”。

方法

新增

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

使用功能選項建構新秘密快取，否則使用預設值。從新的工作階段初始化 SecretsManager 用戶端。將 CacheConfig 初始化為預設值。使用預設的最大大小起始 LRU 快取。

GetSecretString

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

GetSecretString 從快取中取得秘密字串值，用於指定秘密 ID。如果操作失敗，傳回秘密字串和錯誤。

GetSecretStringWithStage

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

GetSecretStringWithStage 從快取中取得秘密字串值，用於指定秘密 ID 和版本階段。如果操作失敗，傳回秘密字串和錯誤。

GetSecretBinary

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

GetSecretBinary 從快取中取得秘密二進位值，用於指定秘密 ID。如果操作失敗，則傳回秘密二進位和錯誤。

GetSecretBinaryWithStage

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

GetSecretBinaryWithStage 從快取中取得秘密二進位值，用於指定秘密 ID 和版本階段。如果操作失敗，則傳回秘密二進位和錯誤。

輸入 CacheConfig

快取的快取組態選項，例如最大快取大小、預設版本階段和快取秘密的存留時間 (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
}
```

輸入 CacheHook

用於掛接到快取，以對存放在快取中的秘密執行動作的介面。

方法

放置

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

準備要存放在快取中的物件。

取得

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

從快取的物件中衍生物件。

使用 Go AWS 開發套件取得 Secrets Manager 秘密值

在應用程式中，您可以呼叫 `GetSecretValue`或任何 AWS SDKs`BatchGetSecretValue`中的 來擷取秘密。不過，建議您使用用戶端快取來快取您的秘密值。快取秘密可提高速度並降低成本。

對於 Go 應用程式，請使用 Secrets Manager Go 型快取元件或使用 GetSecretValue 或 BatchGetSecretValue 直接呼叫 SDK。

下列程式碼範例顯示如何取得 Secrets Manager 秘密值。

必要許可：secretsmanager:GetSecretValue

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

 import (
   "context"
   "log"

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

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

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

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

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

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

```
      log.Fatal(err.Error())
    }

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

    // Your code goes here.
  }
```

使用 Rust 取得 Secrets Manager 秘密值

在應用程式中，您可以呼叫 `GetSecretValue`或任何 AWS SDKs`BatchGetSecretValue`中的 來擷取秘密。不過，建議您使用用戶端快取來快取您的秘密值。快取秘密可提高速度並降低成本。

主題

- 使用 Rust 搭配用戶端快取取得 Secrets Manager 秘密值
- 使用 Rust AWS 開發套件取得 Secrets Manager 秘密值

使用 Rust 搭配用戶端快取取得 Secrets Manager 秘密值

擷取秘密時，您可以使用 Secrets Manager Rust 型快取元件來快取秘密以供日後使用。擷取快取的秘密比從 Secrets Manager 中擷取要快。由於呼叫 Secrets Manager API 需要花費成本，因此使用快取可以降低成本。如需您可以擷取機密的所有方法，請參閱 取得秘密。

快取政策是先進先出 (FIFO)，因此當快取必須捨棄秘密時，會捨棄最舊的秘密。預設情況下，快取每小時重新整理一次秘密。您可以設定下列項目：

- `max_size` – 在移出最近尚未存取的秘密之前，要維護的快取秘密數量上限。
- `ttl` – 在需要重新整理秘密狀態之前，快取項目視為有效的持續時間。

快取實作不包括快取失效。快取實作著重於快取本身，而不是強化或著重於安全性。如果您需要額外的安全性，例如加密快取中的項目，請使用提供的特徵來修改快取。

若要使用 元件，您必須具有具有 的 Rust 2021 開發環境`tokio`。如需詳細資訊，請參閱 Rust 程式設計語言網站上的入門。

若要下載原始程式碼，請參閱 GitHub 上的 Secrets Manager Rust 型快取用戶端元件。

若要安裝快取元件，請使用下列命令。

```
cargo add aws_secretsmanager_caching
```

必要許可：

- secretsmanager:DescribeSecret
- secretsmanager:GetSecretValue

如需詳細資訊，請參閱許可參考。

Example 擷取秘密

下列範例示範如何取得名為 *MyTest* 之秘密的秘密值。

```
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 使用自訂組態和自訂用戶端執行個體化快取

下列範例示範如何設定快取，然後取得名為 *MyTest* 之秘密的秘密值。

```
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 開發套件取得 Secrets Manager 秘密值

在應用程式中，您可以呼叫 `GetSecretValue`或任何 AWS SDKs`BatchGetSecretValue`中的 來擷取秘密。不過，建議您使用用戶端快取來快取您的秘密值。快取秘密可提高速度並降低成本。

對於 Rust 應用程式，請使用 Secrets Manager Rust 型快取元件，或直接使用 GetSecretValue 或 BatchGetSecretValue 呼叫 SDK。

下列程式碼範例顯示如何取得 Secrets Manager 秘密值。

必要許可：`secretsmanager:GetSecretValue`

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

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

```
    Ok(())
}
```

在 Amazon Elastic Kubernetes Service 中使用 AWS Secrets Manager 秘密

若要將來自 AWS Secrets Manager (ASCP) 的秘密顯示為掛載在 Amazon EKS Pod 中的檔案，您可以使用 AWS Kubernetes Secrets Store CSI Driver 的 Secrets and Configuration Provider。ASCP 可與執行 Amazon EC2 節點群組的 Amazon Elastic Kubernetes Service 1.17+ 搭配使用。不支援 AWS Fargate 節點群組。透過 ASCP，您可以在 Secrets Manager 中存放和管理您的秘密，然後透過在 Amazon EKS 上執行的工作負載擷取這些秘密。如果您的秘密包含 JSON 格式的多個鍵/值對，您可以選擇要在 Amazon EKS 中掛載的鍵/值對。ASCP 使用JMESPath語法來查詢秘密中的鍵/值對。ASCP 也可與 Parameter Store 參數搭配使用。ASCP 提供兩種使用 Amazon EKS 進行身分驗證的方法。第一個方法使用服務帳戶的 IAM 角色 (IRSA)。第二個方法使用 Pod 身分。每種方法都有其優點和使用案例。

具有服務帳戶 IAM 角色的 ASCP (IRSA)

具有服務帳戶 IAM 角色 (IRSA) 的 ASCP 可讓您將來自 的秘密掛載 AWS Secrets Manager 為 Amazon EKS Pod 中的檔案。此方法適用於下列情況：

- 您需要將秘密掛載為 Pod 中的檔案。
- 您使用 Amazon EKS 1.17 版或更新版本搭配 Amazon EC2 節點群組。
- 您想要從 JSON 格式的秘密擷取特定的鍵/值對。

如需詳細資訊，請參閱the section called “將 ASCP 與適用於 Amazon EKS 的 IRSA 整合”。

具有 Pod 身分的 ASCP

具有 EKS Pod 身分的 ASCP

具有 Pod Identity 的 ASCP 方法可增強安全性，並簡化在 Amazon EKS 中存取秘密的組態。此方法在下列情況下非常有用：

- 您需要在 Pod 層級進行更精細的許可管理。
- 您使用的是 Amazon EKS 1.24 版或更新版本。
- 您想要改善效能和可擴展性。

如需詳細資訊，請參閱the section called “將 ASCP 與 Amazon EKS 的 Pod Identity 整合”。

選擇正確的方法

在具有 IRSA 的 ASCP 與具有 Pod 身分的 ASCP 之間進行決策時，請考慮下列因素：

- Amazon EKSversion：Pod Identity 需要 Amazon EKS 1.24+，而 CSI 驅動程式可與 Amazon EKS 1.17+ 搭配使用。
- 安全需求：Pod Identity 在 Pod 層級提供更精細的控制。
- 效能：Pod Identity 通常在大規模環境中表現更佳。
- 複雜性：Pod Identity 透過消除對個別服務帳戶的需求，簡化設定。

選擇最適合您特定需求和 Amazon EKS 環境的方法。

安裝適用於 Amazon EKS 的 ASCP

本節說明如何安裝 Amazon EKS 的 AWS Secrets and Configuration Provider。使用 ASCP，您可以從 Secrets Manager 將秘密掛載，並從 將參數掛載 AWS Systems Manager 為 Amazon EKS Pod 中的檔案。

先決條件

- Amazon EKS 叢集
  - 適用於 Pod Identity 的 1.24 版或更新版本
  - 適用於 IRSA 的 1.17 版或更新版本
- AWS CLI 已安裝和設定的
- 為您的 Amazon EKS 叢集安裝和設定 kubectl
- Helm (3.0 版或更新版本)

安裝和設定 ASCP

您可在 secrets-store-csi-provider-aws 儲存庫中的 GitHub 取得 ASCP。儲存庫還包含用於建立和掛載秘密的範例 YAML 檔案。

在安裝期間，您可以將 ASCP 設定為使用 FIPS 端點。如需端點清單，請參閱 the section called “Secrets Manager 端點”。

使用 Helm 安裝 ASCP

1. 若要確保儲存庫指向最新的圖表，請使用 `helm repo update.`
2. 新增 Secrets Store CSI 驅動程式圖表。

```
helm repo add secrets-store-csi-driver https://kubernetes-sigs.github.io/secrets-store-csi-driver/charts
```

3. 安裝圖表。若要設定限流，請新增下列旗標： `--set-json 'k8sThrottlingParams={"qps": "number of queries per second", "burst": "number of queries per second"}'`

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

4. 新增 ASCP 圖表。

```
helm repo add aws-secrets-manager https://aws.github.io/secrets-store-csi-driver-provider-aws
```

5. 安裝圖表。若要使用 FIPS 端點，請新增下列旗標： `--set useFipsEndpoint=true`

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

使用儲存庫中的 YAML 安裝

- 使用下列命令。

```
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
```

驗證安裝

若要驗證 EKS 叢集、Secrets Store CSI 驅動程式和 ASCP 外掛程式的安裝，請遵循下列步驟：

1. 驗證 EKS 叢集：

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

此命令應會傳回叢集的相關資訊。

2. 驗證 Secrets Store CSI 驅動程式安裝：

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

您應該會看到執行的名稱為 的 Podcsi-secrets-store-secrets-store-csi-driver-xxx。

3. 驗證 ASCP 外掛程式安裝：

YAML installation

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

輸出範例：

```
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
```

輸出範例：

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

您應該會看到處於 Running 狀態的 Pod。

執行這些命令之後，如果一切設定正確，您應該會看到所有元件在執行中沒有任何錯誤。如果您遇到任何問題，您可能需要檢查有問題的特定 Pod 日誌來進行故障診斷。

故障診斷

1. 若要檢查 ASCP 提供者的日誌，請執行：

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

2. 檢查kube-system命名空間中所有 Pod 的狀態：

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

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

   所有與 CSI 驅動程式和 ASCP 相關的 Pod 都應處於「執行中」狀態。

3. 檢查 CSI 驅動程式版本：

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

   此命令應會傳回已安裝 CSI 驅動程式的相關資訊。

其他資源

如需搭配 Amazon EKS 使用 ASCP 的詳細資訊，請參閱下列資源：

- 搭配 Amazon EKS 使用 Pod 身分
- AWS Secrets 在 GitHub 上存放 CSI 驅動程式

使用 AWS 秘密和組態提供者 CSI 搭配 Amazon EKS 的 Pod 身分

AWS Secrets and Configuration Provider 與 Pod Identity Agent for Amazon Elastic Kubernetes Service 整合，可為在 Amazon EKS 上執行的應用程式提供增強的安全性、簡化組態並改善效能。從 Secrets Manager 或從 AWS Systems Manager 參數存放區擷取秘密或參數時，Pod Identity 可簡化 Amazon EKS 的 IAM 身分驗證。

Amazon EKS Pod Identity 透過允許直接透過 Amazon EKS 介面設定許可、減少步驟數量，並消除在 Amazon EKS 和 IAM 服務之間切換的需求，簡化設定 Kubernetes 應用程式 IAM 許可的程序。Pod

Identity 可讓您跨多個叢集使用單一 IAM 角色，而無需更新信任政策，並支援角色工作階段標籤，以實現更精細的存取控制。這種方法不僅允許跨角色重複使用許可政策來簡化政策管理，還允許根據相符標籤存取 AWS 資源來增強安全性。

運作方式

1. Pod Identity 會將 IAM 角色指派給 Pod。
2. ASCP 使用此角色進行身分驗證 AWS 服務。
3. 如果獲得授權，ASCP 會擷取請求的秘密，並將其提供給 Pod。

如需詳細資訊，請參閱《Amazon EKS 使用者指南》中的了解 Amazon EKS Pod Identity 的運作方式。

先決條件

> **Important**
>
> Pod Identity 僅支援雲端中的 Amazon EKS。Amazon EC2 執行個體上的 Amazon EKS AnywhereRed Hat OpenShift Service on AWS、 或自我管理 Kubernetes 叢集不支援此功能。

- Amazon EKS 叢集 (1.24 版或更新版本）
- 透過 存取 AWS CLI 和 Amazon EKS 叢集 `kubectl`
- 存取兩個 AWS 帳戶 （用於跨帳戶存取）

安裝 Amazon EKS Pod Identity Agent

若要在叢集中使用 Pod Identity，您必須安裝 Amazon EKS Pod Identity Agent 附加元件。

安裝 Pod Identity Agent

- 在叢集上安裝 Pod Identity Agent 附加元件：

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

使用 Pod 身分設定 ASCP

1. 建立許可政策，將 Pod 需要存取的秘密授予 `secretsmanager:GetSecretValue`和 `secretsmanager:DescribeSecret`許可。如需政策範例，請參閱 the section called “範例：讀取和描述個別秘密的許可”。
2. 建立可由 Amazon EKS 服務主體擔任的 Pod Identity IAM 角色：

   JSON

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

   將 IAM 政策連接至角色：

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

3. 建立 Pod Identity 關聯。如需範例，請參閱《Amazon EKS 使用者指南》中的建立 Pod 身分關聯
4. 建立 `SecretProviderClass`，指定要掛載在 Pod 中的秘密：

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

   SecretProviderClass 在 IRSA 和 Pod Identity 之間的主要差異是選用參數 `usePodIdentity`。這是選用欄位，可決定身分驗證方法。未指定時，它會預設為使用服務帳戶 (IRSA) 的 IAM 角色。

- 若要使用 EKS Pod Identity，請使用下列任何值："true", "True", "TRUE", "t", "T"。
- 若要明確使用 IRSA，請將 設定為下列任何值："false", "False", "FALSE", "f", or "F"。

5. 在 下部署掛載秘密的 Pod/mnt/secrets-store：

```
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 叢集，請確定叢集所在的 VPC 具有 AWS STS 端點。如需有關建立端點的資訊，請參閱AWS Identity and Access Management 《 使用者指南》中的介面 VPC 端點。

驗證秘密掛載

若要驗證秘密是否已正確掛載，請執行下列命令：

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

設定 Amazon EKS Pod Identity 以存取 Secrets Manager 中的秘密

1. 建立許可政策，將 Pod 需要存取的秘密授予 secretsmanager:GetSecretValue和 secretsmanager:DescribeSecret許可。如需政策範例，請參閱 the section called “範例：讀取和描述個別秘密的許可”。
2. 如果您還沒有秘密，請在 Secrets Manager 中建立秘密。

疑難排解

您可以透過描述 Pod 部署來檢視大多數錯誤。

若要查看容器的錯誤訊息

1. 使用下列命令取得 Pod 名稱清單。如果不使用預設命名空間，請使用 -n *NAMESPACE*。

```
kubectl get pods
```

2. 若要描述 Pod，請在下列命令中，針對 *PODID* 使用您在上一個步驟中找到的 Pod 中的 Pod ID。如果不使用預設命名空間，請使用 `-n` *NAMESPACE*。

```
kubectl describe pod/PODID
```

若要查看 ASCP 的錯誤

- 若要在提供者日誌中尋找更多資訊，請在下列命令中，針對 *PODID* 使用 csi-secrets-store-provider-aws Pod 的 ID。

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

使用 AWS Secrets and Configuration Provider CSI 搭配服務帳戶 (IRSA) 的 IAM 角色

主題

- 先決條件
- 設定存取控制
- 識別要掛載的機密
- 疑難排解

先決條件

- Amazon EKS 叢集 (1.17 版或更新版本)
- 透過 存取 AWS CLI 和 Amazon EKS 叢集 `kubectl`

設定存取控制

ASCP 會擷取 Amazon EKS Pod Identity，並將其交換為 IAM 角色。您可以在該 IAM 角色的 IAM 政策中設定許可。當 ASCP 擔任 IAM 角色時，它會存取您授權的秘密。除非您也將其與 IAM 角色建立關聯，否則其他容器無法存取秘密。

授予 Amazon EKS Pod 對 Secrets Manager 中秘密的存取權

1. 建立許可政策，將 Pod 需要存取的秘密授予 `secretsmanager:GetSecretValue`和 `secretsmanager:DescribeSecret` 許可。如需政策範例，請參閱 the section called “範例：讀取和描述個別秘密的許可”。
2. 如果尚未建立，請為叢集建立 IAM OpenID Connect (OIDC) 提供者。如需詳細資訊，請參閱《Amazon EKS 使用者指南》中的為您的叢集建立 IAM OIDC 提供者。
3. 為服務帳戶建立 IAM 角色，並將政策連接至該角色。如需詳細資訊，請參閱《Amazon EKS 使用者指南》中的為服務帳戶建立 IAM 角色。
4. 如果您使用私有 Amazon EKS 叢集，請確定叢集所在的 VPC 具有 AWS STS 端點。如需有關建立端點的資訊，請參閱AWS Identity and Access Management 《 使用者指南》中的介面 VPC 端點。

識別要掛載的機密

若要確定 ASCP 在 Amazon EKS 中掛載哪些機密作為檔案系統上的檔案，請建立 the section called “SecretProviderClass” YAML 檔案。`SecretProviderClass` 會列出要掛載的秘密，以及要掛載的檔案名稱。`SecretProviderClass` 必須與參考的 Amazon EKS Pod 位於相同的命名空間。

將秘密掛載為檔案

下列指示說明如何使用範例 YAML 檔案 ExampleSecretProviderClass.yaml 和 ExampleDeployment.yaml 將秘密掛載為檔案。

在 Amazon EKS 中掛載秘密

1. 將 `SecretProviderClass`套用至 Pod：

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

2. 部署您的 Pod：

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

3. ASCP 掛載檔案。

疑難排解

您可以透過描述 Pod 部署來檢視大多數錯誤。

若要查看容器的錯誤訊息

1. 使用以下命令取得 Pod 名稱清單。如果不使用預設命名空間，請使用 `-n` *`nameSpace`*。

```
kubectl get pods
```

2. 若要描述 Pod，請在下列命令中，針對 *`podId`*，使用您在上一個步驟中找到的 Pod 中的 Pod ID。如果不使用預設命名空間，請使用 `-n` *`nameSpace`*。

```
kubectl describe pod/podId
```

若要查看 ASCP 的錯誤

- 若要在提供者日誌中尋找詳細資訊，請在下列命令中，針對 *`podId`*，使用 csi-secrets-store-provider-aws Pod 的 ID。

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

- 確認已安裝 **`SecretProviderClass`** CRD：

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

此命令應會傳回`SecretProviderClass`自訂資源定義的相關資訊。

- 確認 SecretProviderClass 物件已建立。

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

AWS 秘密和組態提供者程式碼範例

ASCP 身分驗證和存取控制範例

範例：允許 Amazon EKS Pod Identity Service (pods.eks.amazonaws.com：//) 擔任角色並標記工作階段的 IAM 政策：

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 來描述要使用 ASCP 在 Amazon EKS 中裝載的機密。如需範例，請參閱 the section called “SecretProviderClass 用量”。

SecretProviderClass YAML 結構

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

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

參數欄位包含掛載請求的詳細資訊：

region

（選用）秘密 AWS 區域 的 。如果不使用此欄位，ASCP 會從節點上的註釋尋找區域。此查詢會增加掛載請求的額外負荷，因此建議您為使用大量 Pod 的叢集提供 區域。

如果您也指定 `failoverRegion`，則 ASCP 會嘗試從這兩個區域擷取機密。如果任一區域傳回 4xx 錯誤 (例如身分驗證問題)，則 ASCP 不會掛載任一機密。如果已成功從 `region` 擷取機密，則 ASCP 會掛載該機密值。如果未成功從 `region` 擷取機密，但已成功從 `failoverRegion` 擷取機密，則 ASCP 會掛載該機密值。

failoverRegion

(選用) 如果您包含此欄位，則 ASCP 會嘗試從 `region` 中定義的區域和此欄位擷取機密。如果任一區域傳回 4xx 錯誤 (例如身分驗證問題)，則 ASCP 不會掛載任一機密。如果已成功從 `region` 擷取機密，則 ASCP 會掛載該機密值。如果未成功從 `region` 擷取機密，但已成功從 `failoverRegion` 擷取機密，則 ASCP 會掛載該機密值。如需如何使用此欄位的範例，請參閱 多區域秘密容錯移轉。

pathTranslation

(選用) 如果 Amazon EKS 中的檔案名稱將包含路徑分隔符號字元，例如 Linux 上的斜線 (/)，則要使用的單一替代字元。ASCP 無法建立包含路徑分隔符號字元的掛載檔案。相反地，ASCP 會將路徑分隔符號字元取代為其他字元。如果不使用此欄位，則取代字元為底線 (_)，例如 `My/Path/Secret` 掛載為 `My_Path_Secret`。

若要避免發生字元取代的情況，請輸入字串 `False`。

usePodIdentity

（選用）決定身分驗證方法。未指定時，它會預設為服務帳戶 (IRSA) (IRSA) 的 IAM 角色。

- 若要使用 EKS Pod 身分，請使用下列任何值：`"true""`、`"True"`、`"TRUE"`、 `"t"`或 `"T"`。
- 若要明確使用 IRSA，請將 設定為下列任何值：`"false"`、`"False"`、`"f"`、 `"FALSE"`或 `"F""=`。

preferredAddressType

（選用）指定 Pod Identity Agent 端點通訊的慣用 IP 地址類型。欄位僅適用於使用 EKS Pod 身分功能時，並在使用服務帳戶的 IAM 角色時忽略。值不區分大小寫。有效的 值如下：

- `"ipv4"`、`"IPv4""` 或 `"IPV4"` – 強制使用 Pod Identity Agent IPv4 端點
- `"ipv6"`、 `"IPv6"`或 `"IPV6"` – 強制使用 Pod Identity Agent IPv6 端點
- 未指定 – 使用自動端點選擇、先嘗試 IPv4 端點，並在 IPv4 失敗時IPv4IPv6 端點

objects

包含待掛載秘密的 YAML 宣告的字串。建議使用 YAML 多行字串或分隔號 (|) 字元。

objectName

必要。指定要擷取的秘密或參數名稱。對於 Secrets Manager，這是 `SecretId` 參數，可以是秘密的易記名稱或完整 ARN。對於 SSM 參數存放區，這是 參數`Name`的 ，可以是 參數的名稱或完整 ARN。

objectType

如果未針對 `objectName` 使用 Secrets Manager ARN，其則為必要欄位。可以是 `secretsmanager` 或 `ssmparameter`。

objectAlias

（選用）Amazon EKS Pod 中秘密的檔案名稱。如果您未指定此欄位，`objectName` 會顯示為檔案名稱。

filePermission

（選用）指定用來掛載秘密的檔案許可的 4 位數八進位字串。如果您未指定此欄位，則會預設為 `"0644"`。

objectVersion

(選用) 秘密的版本 ID。不建議，因為每次更新機密時都必須更新版本 ID。依預設，會使用最新版本。如果您包含 `failoverRegion`，則此欄位代表主要 `objectVersion`。

objectVersionLabel

(選用) 版本的別名。預設值為最新版本的 AWSCURRENT。如需詳細資訊，請參閱the section called “秘密版本”。如果您包含 `failoverRegion`，則此欄位代表主要 `objectVersionLabel`。

jmesPath

(選用) 要掛載在 Amazon EKS 中之檔案的秘密金鑰映射。若要使用此欄位，您的秘密值必須是 JSON 格式。如果使用此欄位，您必須包含 path 和 objectAlias。

路徑

來自秘密值 JSON 中金鑰值對的金鑰。如果欄位包含連字號，請使用單引號將其逸出，例如：path: '"hyphenated-path"'。

objectAlias

要掛載在 Amazon EKS Pod 中的檔案名稱。如果欄位包含連字號，請使用單引號將其逸出，例如：objectAlias: '"hyphenated-alias"'。

filePermission

（選用）指定用來掛載秘密的檔案許可的 4 位數八進位字串。如果您未指定此欄位，則會預設為父物件的檔案許可。

failoverObject

(選用) 如果您指定此欄位，則 ASCP 會嘗試擷取在主要 objectName 中指定的機密和在 failoverObject objectName 子欄位中指定的機密。如果任一項傳回 4xx 錯誤 (例如身分驗證問題)，則 ASCP 不會掛載任一機密。如果已成功從主要 objectName 擷取機密，則 ASCP 會掛載該機密值。如果未成功從主要 objectName 擷取機密，但已成功從容錯移轉 objectName 擷取機密，則 ASCP 會掛載該機密值。如果包含此欄位，則您必須包含欄位 objectAlias。如需如何使用此欄位的範例，請參閱 容錯移轉至不同的秘密。

當容錯移轉機密不是複本時，您通常會使用此欄位。如需如何指定複本的範例，請參閱 多區域秘密容錯移轉。

objectName

容錯移轉機密的名稱或完整 ARN。如果您使用 ARN，則 ARN 中的區域必須與欄位 failoverRegion 相符。

objectVersion

(選用) 秘密的版本 ID。必須與主要 objectVersion 相符。不建議，因為每次更新機密時都必須更新版本 ID。依預設，會使用最新版本。

objectVersionLabel

(選用) 版本的別名。預設值為最新版本的 AWSCURRENT。如需詳細資訊，請參閱the section called “秘密版本”。

建立基本 SecretProviderClass 組態，以在 Amazon EKS Pod 中掛載秘密。

Pod Identity

SecretProviderClass 可在相同的 Amazon EKS 叢集中使用秘密：

```
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 用量

使用這些範例為不同的案例建立 SecretProviderClass 組態。

範例：依名稱或 ARN 掛載機密

此範例說明如何掛載三種不同類型的秘密：

- 完整 ARN 指定的秘密
- 名稱指定的秘密
- 秘密的特定版本

```
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"
```

範例：從秘密掛載金鑰/值對

此範例示範如何從 JSON 格式的秘密掛載特定的鍵/值對：

```
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
```

範例：依檔案許可掛載秘密

此範例示範如何使用特定檔案許可掛載秘密

```
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"
```

範例：容錯移轉組態範例

這些範例示範如何設定秘密的容錯移轉。

多區域秘密容錯移轉

此範例說明如何為跨多個區域複寫的秘密設定自動容錯移轉：

```
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"
```

容錯移轉至不同的秘密

此範例說明如何設定容錯移轉至不同的秘密（非複本）：

```
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"
```

其他資源

如需搭配 Amazon EKS 使用 ASCP 的詳細資訊，請參閱下列資源：

- 搭配 Amazon EKS 使用 Pod Identity
- 使用 AWS 秘密和組態提供者
- AWS Secrets 在 GitHub 上存放 CSI 驅動程式

在 AWS Lambda 函數中使用 AWS Secrets Manager 秘密

AWS Lambda 是一種無伺服器運算服務，可讓您執行程式碼，而無需佈建或管理伺服器。參數存放區是 的一項功能 AWS Systems Manager，可為組態資料管理和秘密管理提供安全的階層式儲存。您可以使用 AWS Parameters and Secrets Lambda Extension 來擷取和快取 Lambda 函數中的 AWS Secrets Manager 秘密和參數存放區參數，而無需使用 SDK。如需使用此延伸模組的詳細資訊，請參閱《Lambda 開發人員指南》中的在 Lambda 函數中使用 Secrets Manager 秘密。

搭配 Lambda 使用 Secrets Manager 秘密

Lambda 開發人員指南提供在 Lambda 函數中使用 Secrets Manager 秘密的完整說明。開始使用：

1. 遵循 Lambda 函數中的使用 Secrets Manager 秘密step-by-step教學課程，其中包括：
   - 使用您偏好的執行時間建立 Lambda 函數 (Python、Node.js、Java)
   - 新增 AWS 參數和秘密 Lambda 延伸做為 layer
   - 設定必要的許可
   - 撰寫程式碼以從延伸模組擷取秘密
   - 測試您的 函數

2. 了解用於設定延伸模組行為的環境變數，包括快取設定和逾時
3. 了解使用秘密輪換的最佳實務

在 VPC 中使用 Secrets Manager 和 Lambda

如果您的 Lambda 函數在 VPC 中執行，您需要建立 VPC 端點，以便延伸模組可以呼叫 Secrets Manager。如需詳細資訊，請參閱the section called “VPC 端點 (AWS PrivateLink)”。

使用 AWS 參數和秘密 Lambda 延伸模組

延伸模組可以擷取 Secrets Manager 秘密和 Parameter Store 參數。如需搭配 延伸模組使用參數存放區參數的詳細資訊，請參閱AWS Systems Manager 《 使用者指南》中的在 Lambda 函數中使用參數存放區參數。

Systems Manager 文件包括：

- 延伸模組如何與參數存放區搭配使用的詳細解釋
- 將擴充功能新增至 Lambda 函數的指示
- 用於設定延伸模組的環境變數
- 用於擷取參數的範例命令
- 所有支援的架構和區域的完整延伸 ARNs 清單

使用 AWS Secrets Manager 代理程式

Secrets Manager 代理程式的運作方式

AWS Secrets Manager 代理程式是一種用戶端 HTTP 服務，可協助您標準化在運算環境中從 Secrets Manager 取用秘密的方式。您可以搭配下列服務使用：

- AWS Lambda
- Amazon Elastic Container Service
- Amazon Elastic Kubernetes Service
- Amazon Elastic Compute Cloud

Secrets Manager 代理程式會擷取和快取記憶體中的秘密，讓您的應用程式從 localhost 取得秘密，而不是直接呼叫 Secrets Manager。Secrets Manager Agent 只能讀取秘密，無法修改秘密。

Important

Secrets Manager 代理程式會使用您環境中的 AWS 登入資料來呼叫 Secrets Manager。它包含對伺服器端請求偽造 (SSRF) 的保護，以協助改善秘密安全性。Secrets Manager 代理程式預設使用後量子 ML-KEM 金鑰交換作為最高優先順序金鑰交換。

了解 Secrets Manager 代理程式快取

Secrets Manager 代理程式使用記憶體內快取，會在 Secrets Manager 代理程式重新啟動時重設。它會根據下列項目定期重新整理快取的秘密值：

- 預設重新整理頻率 (TTL) 為 300 秒
- 您可以使用組態檔案修改 TTL
- 當您在 TTL 過期後請求秘密時，會發生重新整理

Note

Secrets Manager 代理程式不包含快取失效。如果秘密在快取項目過期之前輪換，Secrets Manager Agent 可能會傳回過時的秘密值。

Secrets Manager Agent 會以與 回應相同的格式傳回秘密值`GetSecretValue`。不會在快取中加密秘密值。

主題

- 建置 Secrets Manager 代理程式
- 安裝 Secrets Manager 代理程式
- 使用 Secrets Manager Agent 擷取秘密
- 了解 refreshNow 參數
- 設定 Secrets Manager 代理程式
- 選用功能
- 日誌
- 安全考量

建置 Secrets Manager 代理程式

開始之前，請確定您已為平台安裝標準開發工具和 Rust 工具。

> **Note**
>
> 在 macOS 上建立已啟用 `fips`功能的代理程式，目前需要以下解決方法：
>
> - 建立名為 的環境變數`SDKROOT`，其設定為執行的結果 `xcrun --show-sdk-path`

RPM-based systems

在以 RPM 為基礎的系統上建置

1. 使用 儲存庫中提供的`install`指令碼。

   指令碼會在啟動時產生隨機 SSRF 字符，並將其存放在 檔案 中`/var/run/awssmatoken`。安裝指令碼建立的 `awssmatokenreader` 群組可讀取權杖。

2. 若要允許應用程式讀取權杖檔案，您需要將應用程式執行所在的使用者帳戶新增至`awssmatokenreader`群組。例如，您可以使用下列 usermod 命令授予應用程式讀取權杖檔案的許可，其中 *<APP_USER>* 是應用程式執行所在的使用者 ID。

   ```
   sudo usermod -aG awssmatokenreader <APP_USER>
   ```

   安裝開發工具

   在 AL2023 等以 RPM 為基礎的系統上，安裝 開發工具 群組：

   ```
   sudo yum -y groupinstall "Development Tools"
   ```

3. 安裝 Rust

   請遵循 Rust 文件中的安裝 Rust 說明：

   ```
   curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh # Follow the on-screen instructions
   . "$HOME/.cargo/env"
   ```

4. 建置代理程式

使用貨運建置命令建置 Secrets Manager 代理程式：

```
cargo build --release
```

您可以在 下找到可執行檔`target/release/aws_secretsmanager_agent`。

Debian-based systems

在 Debian 型系統上建置

1. 安裝開發工具

   在 Ubuntu 等 Debian 系統上，安裝 build-essential 套件：

   ```
   sudo apt install build-essential
   ```

2. 安裝 Rust

   請遵循 Rust 文件中的安裝 Rust 說明：

   ```
   curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh # Follow the on-
   screen instructions
   . "$HOME/.cargo/env"
   ```

3. 建置代理程式

   使用貨運建置命令建置 Secrets Manager 代理程式：

   ```
   cargo build --release
   ```

   您可以在 下找到可執行檔`target/release/aws_secretsmanager_agent`。

Windows

在 Windows 上建置

1. 設定開發環境

   請遵循 Microsoft Windows 文件中在 Windows for Rust 上設定開發環境的指示。

2. 建置代理程式

   使用貨運建置命令建置 Secrets Manager 代理程式：

   ```
   cargo build --release
   ```

   您可以在 下找到可執行檔target/release/aws_secretsmanager_agent.exe。

Cross-compile natively

以原生方式跨編譯

1. 安裝跨編譯工具

   在提供 mingw-w64 套件的分佈上，例如 Ubuntu，安裝跨編譯工具鏈：

   ```
   # Install the cross compile tool chain
   sudo add-apt-repository universe
   sudo apt install -y mingw-w64
   ```

2. 新增 Rust 建置目標

   安裝 Windows GNU 建置目標：

   ```
   rustup target add x86_64-pc-windows-gnu
   ```

3. 適用於 Windows 的建置

   跨編譯適用於 Windows 的代理程式：

   ```
   cargo build --release --target x86_64-pc-windows-gnu
   ```

   您可以在 找到可執行檔target/x86_64-pc-windows-gnu/release/
   aws_secretsmanager_agent.exe。

Cross compile with Rust cross

使用 Rust cross 交叉交叉編譯

如果系統無法原生使用跨編譯工具，您可以使用 Rust 跨專案。如需詳細資訊，請參閱 https：//https://github.com/cross-rs/cross。

> **Important**
>
> 我們建議在建置環境中使用 32GB 的磁碟空間。

1. 設定 Docker

   安裝和設定 Docker：

   ```
   # Install and start docker
   sudo yum -y install docker
   sudo systemctl start docker
   sudo systemctl enable docker # Make docker start after reboot
   ```

2. 設定 Docker 許可

   將您的使用者新增至 Docker 群組：

   ```
   # Give ourselves permission to run the docker images without sudo
   sudo usermod -aG docker $USER
   newgrp docker
   ```

3. 適用於 Windows 的建置

   安裝 Cross 並建置可執行檔：

   ```
   # Install cross and cross compile the executable
   cargo install cross
   cross build --release --target x86_64-pc-windows-gnu
   ```

安裝 Secrets Manager 代理程式

從下列安裝選項中選擇您的運算環境。

Amazon EC2

在 Amazon EC2 上安裝 Secrets Manager 代理程式

1. 導覽至組態目錄

   變更為組態目錄：

   ```
   cd aws_secretsmanager_agent/configuration
   ```

2. 執行安裝指令碼

   執行 儲存庫中提供的`install`指令碼。

   指令碼會在啟動時產生隨機 SSRF 字符，並將其存放在 檔案 中`/var/run/awssmatoken`。安裝指令碼建立的 `awssmatokenreader` 群組可讀取權杖。

3. 設定應用程式許可

   將應用程式執行所在的使用者帳戶新增至 `awssmatokenreader`群組：

   ```
   sudo usermod -aG awssmatokenreader APP_USER
   ```

   以應用程式執行所在的使用者 ID 取代 *APP_USER*。

Container Sidecar

您可以使用 Docker 將 Secrets Manager Agent 作為附屬容器與應用程式一起執行。然後，您的應用程式可以從 Secrets Manager Agent 提供的本機 HTTP 伺服器擷取秘密。如需 Docker 的相關資訊，請參閱 Docker 文件。

為 Secrets Manager Agent 建立附屬容器

1. 建立代理程式 Dockerfile

   為 Secrets Manager Agent 附屬容器建立 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. 建立應用程式 Dockerfile

   為您的用戶端應用程式建立 Dockerfile。

3. 建立 Docker Compose 檔案

   建立 Docker Compose 檔案，以使用共用網路介面執行兩個容器：

> **Important**
>
> 您必須載入 AWS 登入資料和 SSRF 字符，應用程式才能使用 Secrets Manager Agent。如需 Amazon EKS 和 Amazon ECS，請參閱下列內容：
>
> - 《Amazon EKS 使用者指南》中的管理存取權
> - 《Amazon ECS 開發人員指南》中的 Amazon ECS 任務 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. 複製代理程式二進位

   將`secrets-manager-agent`二進位檔複製到包含 Dockerfiles 和 Docker Compose 檔案的相同目錄。

5. 建置和執行容器

   使用 Docker Compose 建置和執行容器：

   ```
   docker-compose up --build
   ```

6. 後續步驟

   您現在可以使用 Secrets Manager Agent 從用戶端容器擷取秘密。如需詳細資訊，請參閱the section called “使用 Secrets Manager Agent 擷取秘密”。

Lambda

您可以將 Secrets Manager 代理程式封裝為 Lambda 延伸模組。然後，您可以將它做為 layer 新增至 Lambda 函數，並從 Lambda 函數呼叫 Secrets Manager Agent 以取得秘密。

下列指示說明如何使用 https://github.com/aws/aws-secretsmanager-agent：// `secrets-manager-agent-extension.sh`中的範例指令碼來安裝 Secrets Manager Agent 做為 Lambda 延伸模組，以取得名為 MyTest 的秘密。

為 Secrets Manager Agent 建立 Lambda 擴充功能

1. 封裝代理程式層

   從 Secrets Manager Agent 程式碼套件的根目錄，執行下列命令：

   ```
   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. 設定 SSRF 字符

   代理程式的預設組態會自動將 SSRF 字符設定為預設`AWS_SESSION_TOKEN`或`AWS_CONTAINER_AUTHORIZATION_TOKEN`環境變數中設定的值（啟用 SnapStart 的 Lambda 函數的後一個變數）。或者，您可以使用 Lambda 函數的任意值來定義`AWS_TOKEN`環境變數，因為此變數優先於其他兩個變數。如果您選擇使用`AWS_TOKEN`環境變數，則必須使用 `lambda:UpdateFunctionConfiguration`呼叫來設定該環境變數。

3. 將 layer 連接至函數

   將 layer 版本連接至 Lambda 函數：

```
# Attach the layer version to the Lambda function
aws lambda update-function-configuration \
    --function-name $LAMBDA_ARN \
    --layers "$LAYER_VERSION_ARN"
```

4. 更新函數程式碼

   更新您的 Lambda 函數，以查詢`http://localhost:2773/secretsmanager/get?secretId=MyTestX-Aws-codes-Secrets-Token`標頭值設定為從上述環境變數取得的

SSRF 字符值，以擷取秘密。請務必在應用程式程式碼中實作重試邏輯，以因應 Lambda 延伸模組的初始化和註冊延遲。

5. 測試函數

   叫用 Lambda 函數來驗證是否正確擷取秘密。

使用 Secrets Manager Agent 擷取秘密

若要擷取秘密，請使用秘密名稱或 ARN 做為查詢參數來呼叫本機 Secrets Manager Agent 端點。根據預設，Secrets Manager Agent 會擷取秘密的`AWSCURRENT`版本。若要擷取不同的版本，請使用 versionStage 或 versionId 參數。

> **Important**
>
> 為了協助保護 Secrets Manager Agent，您必須在每個請求中包含 SSRF 字符標頭：`X-Aws-Parameters-Secrets-Token`。Secrets Manager 代理程式拒絕沒有此標頭或具有無效 SSRF 字符的請求。您可以在 中自訂 SSRF 標頭名稱the section called “組態選項”。

所需的許可

Secrets Manager 代理程式使用適用於 Rust 的 AWS SDK，它使用AWS 登入資料提供者鏈結。這些 IAM 登入資料的身分決定 Secrets Manager Agent 擷取秘密的許可。

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

如需許可的詳細資訊，請參閱「the section called “許可參考”」。

> **Important**
>
> 將秘密值提取至 Secrets Manager Agent 後，任何可存取運算環境和 SSRF 字符的使用者都可以從 Secrets Manager Agent 快取存取秘密。如需詳細資訊，請參閱the section called “安全考量”。

範例請求

curl

Example 範例 – 使用 curl 取得秘密

下列 curl 範例示範如何從 Secrets Manager Agent 取得秘密。此範例倚賴 SSRF 存在於 檔案中，也就是由安裝指令碼存放的位置。

```
curl -v -H \\
    "X-Aws-Parameters-Secrets-Token: $(/var/run/awssmatoken)" \\
    'http://localhost:2773/secretsmanager/get?secretId=YOUR_SECRET_ID' \\
    echo
```

Python

Example 範例 – 使用 Python 取得秘密

下列 Python 範例示範如何從 Secrets Manager Agent 取得秘密。此範例倚賴 SSRF 存在於 檔案中，也就是由安裝指令碼存放的位置。

```
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` 參數

Secrets Manager Agent 使用記憶體內快取來存放秘密值，它會定期重新整理。根據預設，當您在存留時間 (TTL) 過期後請求秘密時，通常會每 300 秒重新整理一次。不過，這種方法有時會導致秘密值過時，特別是當秘密在快取項目過期之前輪換時。

為了解決此限制，Secrets Manager Agent 支援 URL `refreshNow`中稱為 的參數。您可以使用此參數強制立即重新整理秘密的值、略過快取，並確保您擁有up-to-date資訊。

預設行為 （ 不含 **`refreshNow`**)

- 使用快取的值，直到 TTL 過期
- 僅在 TTL 之後重新整理秘密 （ 預設 300 秒 ）
- 如果秘密在快取過期之前輪換，可能會傳回過時的值

使用 的行為 **`refreshNow=true`**

- 完全略過快取
- 直接從 Secrets Manager 擷取最新的秘密值
- 使用新值更新快取並重設 TTL
- 確保您一律取得最新的秘密值

強制重新整理秘密值

> **Important**
>
> `refreshNow` 的預設值為 `false`。設定為 時`true`，它會覆寫 Secrets Manager Agent 組態檔案中指定的 TTL，並對 Secrets Manager 進行 API 呼叫。

curl

Example 範例 – 使用 curl 強制重新整理秘密

下列 curl 範例顯示如何強制 Secrets Manager Agent 重新整理秘密。此範例倚賴 SSRF 存在於 檔案中，也就是由安裝指令碼存放的位置。

```
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 範例 – 使用 Python 強制重新整理秘密

下列 Python 範例示範如何從 Secrets Manager Agent 取得秘密。此範例倚賴 SSRF 存在於 檔案中，也就是由安裝指令碼存放的位置。

```
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}")
```

設定 Secrets Manager 代理程式

若要變更 Secrets Manager Agent 的組態，請建立 TOML 組態檔案，然後呼叫 `./aws_secretsmanager_agent --config config.toml`。

組態選項

`log_level`

Secrets Manager 代理程式日誌中報告的詳細資訊層級：DEBUG、INFO、WARN、 ERROR 或 NONE。預設值為 INFO。

`log_to_file`

是否要登入檔案或 stdout/stderr： `true`或 `false`。預設值為 `true`。

`http_port`

本機 HTTP 伺服器的連接埠，範圍介於 1024 到 65535 之間。預設值為 2773。

`region`

AWS 用於請求的區域。如果未指定區域，Secrets Manager Agent 會從 SDK 判斷區域。如需詳細資訊，請參閱《適用於 AWS Rust 的 SDK 開發人員指南》中的指定您的登入資料和預設區域。

`ttl_seconds`

快取項目的 TTL，以秒為單位，範圍介於 0 到 3600。預設值為 300。0 表示沒有快取。

`cache_size`

快取中可存放的秘密數目上限，範圍介於 1 到 1000。預設為 1000。

ssrf_headers

Secrets Manager 代理程式會檢查 SSRF 字符的標頭名稱清單。預設值為「X-Aws-Parameters-Secrets-Token、X-Vault-Token」。

ssrf_env_variables

Secrets Manager 代理程式會依序檢查 SSRF 字符的環境變數名稱清單。環境變數可以包含權杖或權杖檔案的參考，如 所示`AWS_TOKEN=file:///var/run/awssmatoken`。預設值為「AWS_TOKEN、AWS_SESSION_TOKEN、AWS_CONTAINER_AUTHORIZATION_TOKEN」。

path_prefix

用來判斷請求是否為路徑型請求的 URI 字首。預設值為 "/v1/"。

max_conn

Secrets Manager Agent 允許的最大 HTTP 用戶端連線數，範圍介於 1 到 1000。預設值為 800。

選用功能

Secrets Manager Agent 可以透過將 `--features`旗標傳遞至 ，以選用功能建置`cargo build`。可用的功能包括：

建構功能

prefer-post-quantum

建立最高優先順序`X25519MLKEM768`的金鑰交換演算法。否則，它可用，但不是最高優先順序。`X25519MLKEM768` 是混合式、post-quantum-secure金鑰交換演算法。

fips

將代理程式使用的密碼套件限制為僅 FIPS 核准的密碼。

日誌

本機記錄

Secrets Manager Agent 會根據`log_to_file`組態變數，在本機將錯誤記錄到 檔案`logs/secrets_manager_agent.log`或 stdout/stderr。當您的應用程式呼叫 Secrets Manager 代理程式以取得秘密時，這些呼叫會出現在本機日誌中。它們不會出現在 CloudTrail 日誌中。

日誌輪換

Secrets Manager 代理程式會在檔案達到 10 MB 時建立新的日誌檔案，並總共存放最多五個日誌檔案。

AWS 服務記錄

日誌不會移至 Secrets Manager、CloudTrail 或 CloudWatch。從 Secrets Manager Agent 取得秘密的請求不會出現在這些日誌中。當 Secrets Manager 代理程式呼叫 Secrets Manager 以取得秘密時，該呼叫會以包含 的使用者代理程式字串記錄在 CloudTrail 中`aws-secrets-manager-agent`。

您可以在 中設定記錄選項the section called “組態選項”。

安全考量

信任網域

對於代理程式架構，信任網域是可以存取代理程式端點和 SSRF 字符的地方，通常是整個主機。Secrets Manager 代理程式的信任網域應與可使用 Secrets Manager 登入資料的網域相符，以維持相同的安全狀態。例如，在 Amazon EC2 上，Secrets Manager Agent 的信任網域與使用 Amazon EC2 角色時的登入資料網域相同。

> **Important**
>
> 尚未使用代理程式解決方案且 Secrets Manager 登入資料鎖定至應用程式的安全意識應用程式，應考慮使用語言特定的 AWS SDKs 或快取解決方案。如需詳細資訊，請參閱取得秘密。

使用 C++ AWS SDK 取得 Secrets Manager 秘密值

對於 C++ 應用程式，請直接使用 GetSecretValue 或 BatchGetSecretValue 呼叫 SDK。

下列程式碼範例顯示如何取得 Secrets Manager 秘密值。

必要許可：`secretsmanager:GetSecretValue`

```
//! Retrieve an AWS Secrets Manager encrypted secret.
```

```
/*!
  \param secretID: The ID for the secret.
  \return bool: Function succeeded.
 */
bool AwsDoc::SecretsManager::getSecretValue(const Aws::String &secretID,
                                            const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SecretsManager::SecretsManagerClient
 secretsManagerClient(clientConfiguration);

    Aws::SecretsManager::Model::GetSecretValueRequest request;
    request.SetSecretId(secretID);

    Aws::SecretsManager::Model::GetSecretValueOutcome getSecretValueOutcome =
 secretsManagerClient.GetSecretValue(
            request);
    if (getSecretValueOutcome.IsSuccess()) {
        std::cout << "Secret is: "
                  << getSecretValueOutcome.GetResult().GetSecretString() << std::endl;
    }
    else {
        std::cerr << "Failed with Error: " << getSecretValueOutcome.GetError()
                  << std::endl;
    }

    return getSecretValueOutcome.IsSuccess();
}
```

使用 JavaScript AWS 開發套件取得 Secrets Manager 秘密值

對於 JavaScript 應用程序，請使用 getSecretValue 或 batchGetSecretValue 直接呼叫 SDK。

下列程式碼範例顯示如何取得 Secrets Manager 秘密值。

必要許可：secretsmanager:GetSecretValue

```
import {
  GetSecretValueCommand,
  SecretsManagerClient,
} from "@aws-sdk/client-secrets-manager";

export const getSecretValue = async (secretName = "SECRET_NAME") => {
```

```
  const client = new SecretsManagerClient();
  const response = await client.send(
    new GetSecretValueCommand({
      SecretId: secretName,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '584eb612-f8b0-48c9-855e-6d246461b604',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   ARN: 'arn:aws:secretsmanager:us-east-1:xxxxxxxxxxxx:secret:binary-
secret-3873048-xxxxxx',
  //   CreatedDate: 2023-08-08T19:29:51.294Z,
  //   Name: 'binary-secret-3873048',
  //   SecretBinary: Uint8Array(11) [
  //      98, 105, 110, 97, 114,
  //     121,  32, 100, 97, 116,
  //      97
  //   ],
  //   VersionId: '712083f4-0d26-415e-8044-16735142cd6a',
  //   VersionStages: [ 'AWSCURRENT' ]
  // }

  if (response.SecretString) {
    return response.SecretString;
  }

  if (response.SecretBinary) {
    return response.SecretBinary;
  }
};
```

使用 Kotlin AWS SDK 取得 Secrets Manager 秘密值

對於 Kotlin 應用程式，請直接使用 GetSecretValue 或 BatchGetSecretValue 呼叫 SDK。

下列程式碼範例顯示如何取得 Secrets Manager 秘密值。

必要許可：secretsmanager:GetSecretValue

```
suspend fun getValue(secretName: String?) {
    val valueRequest =
        GetSecretValueRequest {
            secretId = secretName
        }

    SecretsManagerClient { region = "us-east-1" }.use { secretsClient ->
        val response = secretsClient.getSecretValue(valueRequest)
        val secret = response.secretString
        println("The secret value is $secret")
    }
}
```

使用 PHP AWS SDK 取得 Secrets Manager 秘密值

對於 PHP 應用程序，請使用 GetSecretValue 或 BatchGetSecretValue 直接呼叫 SDK。

下列程式碼範例顯示如何取得 Secrets Manager 秘密值。

必要許可：secretsmanager:GetSecretValue

```
<?php

  /**
    * Use this code snippet in your app.
    *
    * If you need more information about configurations or implementing the sample
 code, visit the AWS docs:
    * https://aws.amazon.com/developer/language/php/
    */

  require 'vendor/autoload.php';

  use Aws\SecretsManager\SecretsManagerClient;
  use Aws\Exception\AwsException;

  /**
    * This code expects that you have AWS credentials set up per:
    * https://<<{{DocsDomain}}>>/sdk-for-php/v3/developer-guide/guide_credentials.html
    */
```

```
  // Create a Secrets Manager Client
  $client = new SecretsManagerClient([
      'profile' => 'default',
      'version' => '2017-10-17',
      'region' => '<<{{MyRegionName}}>>',
  ]);

  $secret_name = '<<{{MySecretName}}>>';

  try {
      $result = $client->getSecretValue([
          'SecretId' => $secret_name,
      ]);
  } catch (AwsException $e) {
      // For a list of exceptions thrown, see
      // https://<<{{DocsDomain}}>>/secretsmanager/latest/apireference/
API_GetSecretValue.html
      throw $e;
  }

  // Decrypts secret using the associated KMS key.
  $secret = $result['SecretString'];

  // Your code goes here
```

使用 Ruby AWS SDK 取得 Secrets Manager 秘密值

對於 Ruby 應用程序，請使用 get_secret_value 或 batch_get_secret_value 直接呼叫 SDK。

下列程式碼範例顯示如何取得 Secrets Manager 秘密值。

必要許可：secretsmanager:GetSecretValue

```
  # Use this code snippet in your app.
  # If you need more information about configurations or implementing the sample code,
 visit the AWS docs:
  # https://aws.amazon.com/developer/language/ruby/

  require 'aws-sdk-secretsmanager'
```

```
  def get_secret
    client = Aws::SecretsManager::Client.new(region: '<<{{MyRegionName}}>>')

    begin
      get_secret_value_response = client.get_secret_value(secret_id:
 '<<{{MySecretName}}>>')
    rescue StandardError => e
      # For a list of exceptions thrown, see
      # https://<<{{DocsDomain}}>>/secretsmanager/latest/apireference/
API_GetSecretValue.html
      raise e
    end

    secret = get_secret_value_response.secret_string
    # Your code goes here.
  end
```

使用 取得秘密值 AWS CLI

必要許可：`secretsmanager:GetSecretValue`

Example 擷取機密的加密機密值

下列 `get-secret-value` 範例會取得目前機密值。

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

Example 擷取先前的機密值

下列 `get-secret-value` 範例會取得先前的機密值。

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

使用 取得批次中的一組秘密 AWS CLI

必要許可：

- `secretsmanager:BatchGetSecretValue`

- `secretsmanager:GetSecretValue` 您要擷取的每個秘密的 許可。
- 如果您使用過濾器，則還必須擁有 `secretsmanager:ListSecrets`。

如需許可政策範例，請參閱 the section called “範例：擷取批次中一組秘密值的許可”。

> **Important**
>
> 如果您的 VPCE 原則拒絕擷取您要擷取之群組中個別密碼的權限，則 `BatchGetSecretValue` 不會傳回任何秘密值，而且會傳回錯誤。

Example 擷取按名稱列出的密碼群組的密碼值

下列 `batch-get-secret-value` 範例會為三個秘密取得目前機密值。

```
aws secretsmanager batch-get-secret-value \
          --secret-id-list MySecret1 MySecret2 MySecret3
```

Example 擷取按篩選條件列出的密碼群組的密碼值

下列 `batch-get-secret-value` 範例會取得具有名為「Test」之標籤之密碼的密碼值。

```
aws secretsmanager batch-get-secret-value \
          --filters Key="tag-key",Values="Test"
```

使用 AWS 主控台取得秘密值

若要擷取秘密 (主控台)

1. 前往以下位置開啟 Secrets Manager 主控台：https://console.aws.amazon.com/secretsmanager/。
2. 在秘密清單中，選擇您想要擷取的秘密。
3. 在 Secret value (秘密值) 區段，選擇 Retrieve secret value (擷取秘密值)。

   Secrets Manager 會顯示秘密的目前版本 (AWSCURRENT)。若要查看秘密的其他版本，例如 AWSPREVIOUS 或自訂標記版本，請使用 the section called “AWS CLI”。

在 中使用 AWS Secrets Manager 秘密 AWS Batch

AWS Batch 可協助您在 上執行批次運算工作負載 AWS 雲端。使用 AWS Batch，您可以將敏感資料存放在 AWS Secrets Manager 秘密中，然後在任務定義中參考，藉此將敏感資料注入任務。如需詳細資訊，請參閱使用 Secrets Manager 指定敏感資料。

在 資源中 AWS CloudFormation 取得 AWS Secrets Manager 秘密

使用 AWS CloudFormation，您可以擷取要在其他 AWS CloudFormation 資源中使用的秘密。常見的案例是首先使用 Secrets Manager 產生的密碼建立秘密，然後從秘密中擷取使用者名稱和密碼，以用作新資料庫的憑證。如需使用 建立秘密的資訊 AWS CloudFormation，請參閱 AWS CloudFormation。

若要擷取 AWS CloudFormation 範本中的秘密，請使用動態參考。當您建立堆疊時，動態參考會將秘密值提取至 AWS CloudFormation 資源，因此您不需要硬式編碼秘密資訊。相對地，您就必須依名稱或 ARN 來參考秘密。您可以在任何資源屬性中將動態參考用於機密。您不能在資源中繼資料中將動態參考用於機密 (例如 `AWS::CloudFormation::Init`)，因為這會使機密值在主控台中可見。

秘密的動態參考具有下列模式：

```
{{resolve:secretsmanager:secret-id:SecretString:json-key:version-stage:version-id}}
```

secret-id

秘密的名稱或 ARN。若要存取 AWS 帳戶中的秘密，您可以使用秘密名稱。若要存取不同 AWS 帳戶中的秘密，請使用秘密的 ARN。

json-key (選用)

您要擷取值的鍵值組的索引鍵名稱。如果您未指定 `json-key`， 會 AWS CloudFormation 擷取整個秘密文字。此區段可能不可包含冒號字元 (`:`)。

version-stage (選用)

要使用的秘密版本。Secrets Manager 在輪換程序期間使用預備標籤來追蹤不同版本。如果您使用 `version-stage`，請不要指定 `version-id`。如果您未指定 `version-stage` 或 `version-id`，則預設為 `AWSCURRENT` 版本。此區段可能不可包含冒號字元 (`:`)。

version-id (選用)

您要使用的秘密版本的唯一識別碼。如果您指定 `version-id`，則請不要指定 `version-stage`。如果您未指定 `version-stage` 或 `version-id`，則預設為 `AWSCURRENT` 版本。此區段可能不可包含冒號字元 (`:`)。

如需詳細資訊，請參閱使用動態參考指定 Secrets Manager 秘密。

> **Note**
>
> 請勿使用反斜線(\)作為最終值來建立動態參考。 AWS CloudFormation 無法解析這些參考，這會導致資源失敗。

在 GitHub 任務中使用 AWS Secrets Manager 秘密

若要在 GitHub 任務中使用秘密，您可以使用 GitHub 動作從 擷取秘密， AWS Secrets Manager 並將其新增為 GitHub 工作流程中的遮罩環境變數。如需 GitHub Actions 的詳細資訊，請參閱 GitHub Docs (GitHub 文件) 中的 Understanding GitHub Actions (瞭解 GitHub Actions)。

將秘密新增至 GitHub 環境時，GitHub 工作中的所有其他步驟都可以使用該秘密。請遵循 Security hardening for GitHub Actions (GitHub Actions 的安全性強化) 中的指引，協助防止您環境中的秘密遭到濫用。

您可以將秘密值的整個字串設定為環境變數值，或者如果字串為 JSON，您可以剖析 JSON，為每個 JSON 索引鍵值組設定個別環境變數。如果秘密值是二進位，此動作會將其轉換為字串。

若要檢視以您秘密建立的環境變數，請開啟偵錯記錄。如需詳細資訊，請參閱 GitHub Docs (GitHub 文件) 中的 Enabling debug logging (啟用偵錯記錄)。

若要使用以秘密建立而成的環境變數，請參閱 GitHub 文件中的環境變數。

先決條件

若要使用此動作，您首先需要設定 AWS 登入資料，並使用 `configure-aws-credentials` 步驟 AWS 區域 在 GitHub 環境中設定 。依照針對 GitHub 動作設定 AWS 憑證動作的指示操作，使用 GitHub OIDC 供應商直接擔任角色。這可讓您使用短期憑證，避免將其他存取金鑰儲存在 Secrets Manager 之外。

此動作擔任的 IAM 角色必須擁有下列許可：

- 對您要擷取的秘密有 GetSecretValue。
- 對所有秘密有 ListSecrets。
- KMS key 如果秘密使用 加密，Decrypt則為 （ 選用 ） 客戶受管金鑰。

如需詳細資訊，請參閱the section called “身分驗證與存取控制”。

用量

若要使用此動作，請新增步驟至使用下列語法的工作流程。

```
- 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
```

參數

secret-ids

秘密 ARN、名稱和名稱字首。

若要設定環境變數名稱，請在秘密 ID 前輸入名稱，然後加上英文逗號。例如，ENV_VAR_1, secretId 會以秘密 secretId 建立名為 ENV_VAR_1 的環境變數。環境變數名稱可包含大寫字母、數字和底線。

若要使用字首，請輸入至少三個字元，然後加上星號。例如，dev* 會符合名稱以 dev 開頭的所有秘密。可擷取的相符秘密數上限為 100。如果您設定變數名稱，且字首與多個秘密相符，則動作會失敗。

name-transformation

此步驟預設會以秘密名稱建立每個環境變數名稱，環境變數名稱已轉換為只包含大寫字母、數字和底線，因此不會以數字開頭。對於名稱中的字母，您可以將步驟設定為使用小寫字母搭配 lowercase或 ，不要變更字母搭配 的案例none。預設值為 uppercase。

`parse-json-secrets`

(選用) 此動作預設會將環境變數值設定為秘密值的整個 JSON 字串。將 `parse-json-secrets`設定為 `true`，以為 JSON 中的每個鍵/值對建立環境變數。

請注意，如果 JSON 使用區分大小寫的索引鍵 (例如 "name" 和 "Name")，動作會有重複名稱衝突。在此情況下，請將 `parse-json-secrets` 設定為 `false`，並分別剖析 JSON 秘密值。

環境變數命名

由 動作建立的環境變數，其名稱會與其來源的秘密相同。環境變數的命名要求比秘密更嚴格，因此 動作會轉換秘密名稱以符合這些要求。例如，此動作會將小寫字母轉換為大寫字母。如果您剖析秘密的 JSON，環境變數名稱會同時包含秘密名稱和 JSON 金鑰名稱，例如 `MYSECRET_KEYNAME`。您可以設定動作，使其不會轉換小寫字母。

如果兩個環境變數最終會以相同的名稱出現，動作會失敗。在此情況下，您必須將要用於環境變數的名稱指定為別名。

名稱可能衝突的範例：

- 名為 "MySecret" 的秘密和名為 "mysecret" 的秘密都會成為名為 "MYSECRET" 的環境變數。
- 名為 "Secret_keyname" 的秘密和名為 "Secret" 的 JSON 剖析秘密，以及名為 "keyname" 的金鑰，都會成為名為 "SECRET_KEYNAME" 的環境變數。

您可以指定別名來設定環境變數名稱，如下列範例所示，這會建立名為 的變數`ENV_VAR_NAME`。

```
secret-ids: |
  ENV_VAR_NAME, secretId2
```

空白別名

- 如果您設定`parse-json-secrets: true`並輸入空白別名，後面接著逗號和秘密 ID，動作會命名環境變數，與剖析的 JSON 金鑰相同。變數名稱不包含秘密名稱。

  如果秘密不包含有效的 JSON，則動作會建立一個環境變數，並將其命名為與秘密名稱相同。
- 如果您設定`parse-json-secrets: false`並輸入空白別名，後面接著逗號和秘密 ID，則動作會將環境變數命名為您未指定別名。

下列範例顯示空白別名。

```
,secret2
```

範例

Example 1 依名稱和 ARN 取得秘密

以下範例會為依名稱和 ARN 識別的秘密，建立環境變數。

```
- 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
```

建立的環境變數：

```
EXAMPLESECRETNAME: secretValue1
TEST1: secretValue2
_0_TEST_SECRET: secretValue3
_PROD_EXAMPLE_SECRET: secretValue4
SECRET_ALIAS_1: secretValue5
SECRET_ALIAS_2: secretValue6
SECRET2: secretValue7
```

Example 2 取得以字首開頭的所有秘密

以下範例會為名稱以 *beta* 開頭的所有秘密，建立環境變數。

```
- name: Get Secret Names by Prefix
  uses: 2
  with:
    secret-ids: |
      beta*    # Retrieves all secrets that start with 'beta'
```

建立的環境變數：

```
BETASECRETNAME: secretValue1
BETATEST: secretValue2
BETA_NEWSECRET: secretValue3
```

Example 3 剖析秘密中的 JSON

以下範例會剖析秘密中的 JSON，藉此建立環境變數。

```
- 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
```

秘密 `test/secret` 具有以下秘密值。

```
{
  "api_user": "user",
  "api_key": "key",
  "config": {
    "active": "true"
  }
}
```

秘密 `secret2` 具有以下秘密值。

```
{
  "myusername": "alejandro_rosalez",
  "mypassword": "EXAMPLE_PASSWORD"
}
```

建立的環境變數：

```
TEST_SECRET_API_USER: "user"
TEST_SECRET_API_KEY: "key"
TEST_SECRET_CONFIG_ACTIVE: "true"
MYUSERNAME: "alejandro_rosalez"
```

```
MYPASSWORD: "EXAMPLE_PASSWORD"
```

Example 4 針對環境變數名稱使用小寫字母

下列範例會建立名稱為小寫的環境變數。

```
- name: Get secrets
  uses: aws-actions/aws-secretsmanager-get-secrets@v2
  with:
    secret-ids: exampleSecretName
    name-transformation: lowercase
```

環境變數已建立：

```
examplesecretname: secretValue
```

在 中使用 AWS Secrets Manager 秘密 AWS IoT Greengrass

AWS IoT Greengrass 是將雲端功能擴展至本機裝置的軟體。這能讓裝置收集與分析更接近資訊來源的資料、自主回應本機裝置，在本機網路上安全地互相通訊。

AWS IoT Greengrass 可讓您使用 AWS IoT Greengrass 裝置的服務和應用程式進行身分驗證，而不需要硬式編碼密碼、字符或其他秘密。您可以使用 在 cloud. AWS IoT Greengrass extends Secrets Manager 中 AWS Secrets Manager 安全地存放和管理秘密， AWS IoT Greengrass 以便您的連接器和 Lambda 函數可以使用本機秘密與服務和應用程式互動。

若要將秘密整合到 AWS IoT Greengrass 群組中，您可以建立參考 Secrets Manager 秘密的群組資源。此私密資源會使用相關聯的 ARN 來參考雲端私密。若要了解如何建立、管理和使用秘密資源，請參閱《 AWS IoT 開發人員指南》中的使用秘密資源。

若要將秘密部署至 AWS IoT Greengrass 核心，請參閱將秘密部署至 AWS IoT Greengrass 核心。

在參數存放區中使用 AWS Secrets Manager 秘密

AWS Systems Manager 參數存放區為組態資料管理和秘密管理提供安全的階層式儲存。您可以存放密碼、資料庫字串和授權碼之類的資料做為參數值。不過，參數存放區不會為存放的秘密提供自動輪換服務。反之，參數存放區可讓您將秘密存放在 Secrets Manager 中，然後將該秘密當作參數存放區參數來參考。

當您使用 Secrets Manager 設定參數存放區時，`secret-id` 參數存放區需要在名稱字串之前加上正斜線 (/)。

如需詳細資訊，請參閱AWS Systems Manager 《 使用者指南》中的從參數存放區參數參考 AWS Secrets Manager 秘密。

輪換 AWS Secrets Manager 秘密

輪換是定期更新秘密的過程。當您輪換秘密時，會更新秘密和資料庫或服務中的憑證。在 Secrets Manager 中，您可以為秘密設定自動輪換。有兩種輪換形式：

- 受管輪換 – 對於大多數受管秘密，您可以使用 受管輪換，服務會為您設定和管理輪換。受管輪換不使用 Lambda 函數。
- the section called “依 Lambda 函數輪換” – 對於其他類型的秘密，Secrets Manager 輪換會使用 Lambda 函數來更新秘密和資料庫或服務。

AWS Secrets Manager 秘密的受管輪換

部分服務提供受管輪換，服務會在其中為您設定和管理輪換。使用受管輪換時，您不會使用 AWS Lambda 函數來更新資料庫中的秘密和登入資料。

下列服務提供受管輪換：

- Amazon Aurora 為主要使用者憑證提供受管輪換。如需詳細資訊，請參閱《Amazon Aurora 使用者指南》中的使用 Amazon Aurora 和 AWS Secrets Manager進行密碼管理。
- Amazon ECS Service Connect 為 TLS AWS Private Certificate Authority 憑證提供受管輪換。如需詳細資訊，請參閱《Amazon Elastic Container Service 開發人員指南》中的 TLS with Service Connect。
- Amazon RDS 為主要使用者憑證提供受管輪換。如需詳細資訊，請參閱《Amazon RDS 使用者指南》中的使用 Amazon RDS 和 AWS Secrets Manager進行密碼管理。
- Amazon Redshift 為管理員密碼提供受管輪換。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的使用 AWS Secrets Manager管理 Amazon Redshift 管理員密碼。

> **Tip**
>
> 如需所有其他類型的機密，請參閱 the section called “依 Lambda 函數輪換”。

受管秘密的輪換通常會在一分鐘內完成。在輪換期間，擷取秘密的新連線可能會取得舊版的憑證。在應用程式中，我們強烈建議您遵循最佳實務，使用以應用程式要求的最低權限建立的資料庫使用者，而非使用主使用者。對於應用程式使用者，為取得最高的可用性，您可以使用交替使用者輪換策略。

變更受管輪換的排程

1. 在 Secrets Manager 主控台中開啟受管機密。您可以遵循來自管理服務的連結，或在 Secrets Manager 主控台中搜尋機密。
2. 在 Rotation schedule (輪換排程) 中，在 Schedule expression builder (排程表達式建置器)，或以 Schedule expression (排程表達式) 形式，輸入 UTC 時區的排程。Secrets Manager 會將您的排程儲存為 `rate( )` 或 `cron( )` 表達式。輪換時段會自動在午夜時開始，除非您指定 Start time (開始時間)。您可以每四小時輪換一次秘密。如需詳細資訊，請參閱輪換排程。
3. (選用) 對於 Window duration (時段持續時間)，選擇您想要 Secrets Manager 輪換秘密的時段長度，例如，三個小時時段 **3h**。時段不得延伸到下一個輪換時段。如果您未指定 Window duration (時段持續時間)，則對於以小時為單位的輪換排程，時段會在一小時後自動關閉。對於以天為單位的輪換排程，時段會在一天結束時自動關閉。
4. 選擇儲存。

變更受管輪換的排程 (AWS CLI)

- 呼叫 `rotate-secret`。下列範例會在每個月的第 1 天和第 15 天 16:00 和 18:00 (UTC) 之間進行。如需詳細資訊，請參閱輪換排程。

```
aws secretsmanager rotate-secret \
    --secret-id MySecret \
    --rotation-rules \
        "{\"ScheduleExpression\": \"cron(0 16 1,15 * ? *)\", \"Duration\": \"2h\"}"
```

依 Lambda 函數輪換

對於許多類型的秘密，Secrets Manager 會使用 AWS Lambda 函數來更新秘密和資料庫或服務。如需有關使用 Lambda 函數成本的資訊，請參閱 定價。

對於部分 由其他服務管理的秘密，您可以使用受管輪換。若要使用 受管輪換，您可以先透過管理服務建立機密。

輪換期間，Secrets Manager 會記錄表示輪換狀態的事件。如需詳細資訊，請參閱the section called “使用 記錄 AWS CloudTrail ”。

若要輪換秘密，Secrets Manager 會根據您設定的輪換排程呼叫 Lambda 函數。如果您在設定自動輪換時也手動更新機密值，則 Secrets Manager 會在計算下一個輪替日期時將其視為有效輪換。

輪換期間，Secrets Manager 會多次呼叫相同的函數，且每次使用不同的參數。Secrets Manager 使用參數的下列 JSON 請求結構來叫用函數：

```
{
    "Step" : "request.type",
    "SecretId" : "string",
    "ClientRequestToken" : "string",
    "RotationToken" : "string"
}
```

參數：

- 步驟 – 輪換步驟：`create_secret`、`test_secret`、 `set_secret`或 `finish_secret`。如需詳細資訊，請參閱the section called “輪換函數中的四個步驟”。
- SecretId – 要輪換之秘密的 ARN。
- ClientRequestToken – 新版本秘密的唯一識別符。此值有助於確保冪等性。如需詳細資訊，請參閱 AWS Secrets Manager API 參考中的 PutSecretValue： ClientRequestToken。
- RotationToken – 指出請求來源的唯一識別符。使用擔任的角色或跨帳戶輪換進行秘密輪換時需要，您可以在其中使用另一個帳戶中的 Lambda 輪換函數輪換一個帳戶中的秘密。在這兩種情況下，輪換函數會擔任 IAM 角色來呼叫 Secrets Manager，然後 Secrets Manager 會使用輪換字符來驗證 IAM 角色身分。

如果任何輪換步驟失敗，Secrets Manager 會多次重試整個輪換過程。

主題

- 設定 Amazon RDS、Amazon Aurora、Amazon Redshift 或 Amazon DocumentDB 秘密的自動輪換
- 設定非資料庫 AWS Secrets Manager 秘密的自動輪換
- 使用 設定自動輪換 AWS CLI
- Lambda 函數輪換策略
- Lambda 輪換函數
- AWS Secrets Manager 輪換函數範本
- 的 Lambda 輪換函數執行角色許可 AWS Secrets Manager
- AWS Lambda 輪換函數的網路存取
- 對 AWS Secrets Manager 輪換進行故障診斷

設定 Amazon RDS、Amazon Aurora、Amazon Redshift 或 Amazon DocumentDB 秘密的自動輪換

本教學課程說明如何設定the section called “依 Lambda 函數輪換”資料庫秘密。輪換是定期更新機密的過程。當您輪換機密時，會更新機密和資料庫中的憑證。在 Secrets Manager 中，您可以為資料庫秘密設定自動輪換。

若要使用主控台設定輪換，您必須先選擇輪換策略。接著設定秘密進行輪換，如果您還沒有 Lambda 輪換函數，這會建立一個。主控台也會為 Lambda 函數執行角色設定許可。最後一步是確保 Lambda 輪換函數可以透過網路，存取 Secrets Manager 和您的資料庫。

> **Warning**
>
> 若要開啟自動輪換，您必須擁有為 Lambda 輪換函數建立 IAM 執行角色的許可，並將許可政策連接至該角色。您同時需要 `iam:CreateRole` 和 `iam:AttachRolePolicy` 許可。授予這些許可可讓身分授予自己任何許可。

步驟：

- 步驟 1：選擇輪換策略並 (選擇性) 建立超級使用者秘密
- 步驟 2：設定輪換並建立輪換函數
- 步驟 3：(選用) 對輪換函數設定其他許可條件
- 步驟 4：為輪換函數設定網路存取
- 後續步驟

步驟 1：選擇輪換策略並 (選擇性) 建立超級使用者秘密

如需 Secrets Manager 所提供策略的相關資訊，請參閱 the section called “Lambda 函數輪換策略”。

如果您選擇交替使用者策略，必須建立秘密，並在其中儲存資料庫超級使用者憑證。您需要具有超級使用者憑證的秘密，因為輪換會複製第一個使用者，而大多數使用者沒有該許可。請注意，Amazon RDS Proxy 不支援交替使用者策略。

步驟 2：設定輪換並建立輪換函數

若要為 Amazon RDS、Amazon DocumentDB 或 Amazon Redshift 秘密開啟輪換

1. 前往以下位置開啟機密管理員控制台：https://console.aws.amazon.com/secretsmanager/。

2. 在 Secrets (機密) 頁面中，選擇機密。
3. 在 Secret details (機密詳細資訊) 頁面的 Rotation configuration (輪換組態) 區段中，選擇 Edit rotation (編輯輪換)。
4. 在 Edit rotation configuration (編輯輪換組態) 對話方塊中，執行以下動作：

   a. 開啟 Automatic rotation (自動輪換)。

   b. 在 Rotation schedule (輪換排程) 中，在 Schedule expression builder (排程表達式建置器)，或以 Schedule expression (排程表達式) 形式，輸入 UTC 時區的排程。Secrets Manager 會將您的排程儲存為 `rate( )` 或 `cron( )` 表達式。輪換時段會自動在午夜時開始，除非您指定 Start time (開始時間)。您可以每四小時輪換一次秘密。如需詳細資訊，請參閱輪換排程。

   c. (選用) 對於 Window duration (時段持續時間)，選擇您想要 Secrets Manager 輪換秘密的時段長度，例如，三個小時時段 **3h**。時段不得延伸到下一個輪換時段。如果您未指定 Window duration (時段持續時間)，則對於以小時為單位的輪換排程，時段會在一小時後自動關閉。對於以天為單位的輪換排程，時段會在一天結束時自動關閉。

   d. (選用) 選擇 Rotate immediately when the secret is stored (存放秘密時立即輪換) 以在儲存變更時輪換您的秘密。如果清除核取方塊，則第一次輪換將按照您設定的排程開始。

      如果輪換失敗，例如因為步驟 3 和 4 尚未完成，Secrets Manager 會多次重試輪換流程。

   e. 在 Rotation function (輪換函數) 下，請執行下列其中一項：

      - 選擇 Create a new Lambda function (新建 Lambda 函數)，然後輸入新函數的名稱。Secrets Manager 會將 `SecretsManager` 新增到函數名稱的開頭。Secrets Manager 會根據適當的範本建立函數，並為 Lambda 執行角色設定必要的許可。
      - 選擇 Use an existing Lambda function (使用現有的 Lambda 函數)，重複使用您用於其他秘密的輪換函數。Recommended VPC configurations (建議的 VPC 組態) 下列出的輪換函數，與資料庫具有相同的 VPC 和安全群組，有助於函數存取資料庫。

   f. 對於輪換策略，選擇單一使用者或交替使用者策略。如需詳細資訊，請參閱the section called "步驟 1：選擇輪換策略並 (選擇性) 建立超級使用者秘密"。

5. 選擇 Save (儲存)。

步驟 3：(選用) 對輪換函數設定其他許可條件

在輪換函數的資源政策中，我們建議您包含內容金鑰 `aws:SourceAccount`，協助防止 Lambda 被當作混淆代理人。對於某些 AWS 服務，為避免混淆代理人案例， AWS 建議您同時使用 `aws:SourceArn`和 `aws:SourceAccount`全域條件金鑰。但是，如果在您的輪換函數政策中包

含 aws:SourceArn 條件，則輪換函數只能用於輪換該 ARN 指定的秘密。建議您僅包含內容金鑰 aws:SourceAccount，以便可以將輪換函數用於多個秘密。

若要更新輪換函數資源政策

1. 在 Secrets Manager 主控台中，選擇您的秘密，然後在詳細資訊頁面的 Rotation configuration (輪換組態) 之下，選擇 Lambda 輪換函數。Lambda 主控台開啟。
2. 遵循將資源型政策用於 Lambda 中的指示，新增 aws:sourceAccount 條件。

```
"Condition": {
    "StringEquals": {
        "AWS:SourceAccount": "123456789012"
    }
},
```

如果使用 KMS 金鑰而不是 AWS 受管金鑰 aws/secretsmanager 為秘密加密，Secrets Manager 便會授予 Lambda 執行角色使用該金鑰的許可。您可以透過 SecretARN 加密內容來限制使用解密函數，從而使輪換函數角色僅有權解密其負責輪換的秘密。

更新輪換函數執行角色

1. 在 Lambda 輪換函數中選擇組態，然後在執行角色下選擇角色名稱。
2. 按照修改角色許可政策中的指示新增 kms:EncryptionContext:SecretARN 條件。

```
"Condition": {
    "StringEquals": {
        "kms:EncryptionContext:SecretARN": "SecretARN"
    }
},
```

步驟 4：為輪換函數設定網路存取

如需詳細資訊，請參閱the section called “AWS Lambda 輪換函數的網路存取”。

後續步驟

請參閱the section called “輪換疑難排解”。

設定非資料庫 AWS Secrets Manager 秘密的自動輪換

本教學課程說明如何the section called “依 Lambda 函數輪換”設定非資料庫秘密。輪換是定期更新機密的過程。當您輪換秘密時，會更新秘密以及該秘密所針對資料庫或服務中的憑證。

如需資料庫秘密，請參閱資料庫秘密的自動輪換 (主控台)。

> **Warning**
>
> 若要開啟自動輪換，您必須擁有許可，才能為 Lambda 輪換函數建立 IAM 執行角色，並將許可政策連接至該角色。您同時需要 `iam:CreateRole` 和 `iam:AttachRolePolicy` 許可。授予這些許可可讓身分授予自己任何許可。

步驟：

- 步驟 1：建立一般輪換函數
- 步驟 2：撰寫輪換函數程式碼
- 步驟 3：設定輪換的秘密
- 步驟 4：允許輪換函數存取 Secrets Manager 和您的資料庫或服務
- 步驟 5：允許 Secrets Manager 叫用輪換函數
- 步驟 6：設定輪換函數的網路存取
- 後續步驟

步驟 1：建立一般輪換函數

若要開始，請建立 Lambda 輪換函數。其中不會包含用來輪換秘密的程式碼，因此您將在後續步驟中寫入該程式碼。如需輪換函數如何運作的詳細資訊，請參閱 the section called “Lambda 輪換函數”。

在支援的區域中，您可以使用 從範本 AWS Serverless Application Repository 建立函數。如需支援的區域清單，請參閱AWS Serverless Application Repository FAQs。在其他區域中，您會從頭開始建立函數，並將範本程式碼複製到函數中。

建立一般輪換函數

1. 若要判斷您的區域是否 AWS Serverless Application Repository 支援 ，請參閱AWS Serverless Application Repository 《 一般參考》中的端點和配額。 AWS

2. 執行以下任意一項：
   - 如果您的 區域支援 AWS Serverless Application Repository ：
     a. 在 Lambda 主控台中，選擇應用程式，然後選擇建立應用程式。
     b. 在建立應用程式頁面上，選擇無伺服器應用程式索引標籤。
     c. 在公有應用程式下的搜尋方塊中，輸入 **SecretsManagerRotationTemplate**。
     d. 選取顯示建立自訂 IAM 角色或資源政策的應用程式。
     e. 選擇 SecretsManagerRotationTemplate 圖磚。
     f. 在檢閱、設定和部署頁面上，在應用程式設定圖磚中填入必要欄位。
        - 針對端點，輸入您區域的端點，包括 **https://**。如需端點清單，請參閱 the section called “Secrets Manager 端點”。
        - 若要將 Lambda 函數放入 VPC，請包含 vpcSecurityGroupIds 和 vpcSubnetIds。
     g. 選擇部署。
   - 如果您的區域 AWS Serverless Application Repository 不支援 ：
     a. 在 Lambda 主控台中，選擇函數，然後選擇建立函數。
     b. 在 Create function (建立函數) 頁面上，執行下列動作：
        i. 選擇從頭開始撰寫。
        ii. 針對 Function name (函數名稱)，輸入您輪換函數的名稱。
        iii. 針對 Runtime (執行階段)，選擇 Python 3.9。
        iv. 選擇 Create function (建立函數)。

步驟 2：撰寫輪換函數程式碼

在此步驟中，您會撰寫更新秘密的程式碼，以及秘密所要的服務或資料庫。如需輪換函數功能的相關資訊，包括撰寫自有輪換函數的秘訣，請參閱 the section called “Lambda 輪換函數”。您也可以使用 輪換函數範本做為參考。

步驟 3：設定輪換的秘密

在此步驟中，您會為秘密設定輪換排程，並將輪換函數連接至秘密。

若要設定輪換並建立空白輪換函數

1. 前往以下位置開啟機密管理員控制台：https://console.aws.amazon.com/secretsmanager/。

2. 在 Secrets (機密) 頁面中，選擇機密。
3. 在 Secret details (機密詳細資訊) 頁面的 Rotation configuration (輪換組態) 區段中，選擇 Edit rotation (編輯輪換)。在 Edit rotation configuration (編輯輪換組態) 對話方塊中，執行以下動作：
   a. 開啟 Automatic rotation (自動輪換)。
   b. 在 Rotation schedule (輪換排程) 中，在 Schedule expression builder (排程表達式建置器)，或以 Schedule expression (排程表達式) 形式，輸入 UTC 時區的排程。Secrets Manager 會將您的排程儲存為 `rate()` 或 `cron()` 表達式。輪換時段會自動在午夜時開始，除非您指定 Start time (開始時間)。您可以每四小時輪換一次秘密。如需詳細資訊，請參閱輪換排程。
   c. (選用) 對於 Window duration (時段持續時間)，選擇您想要 Secrets Manager 輪換秘密的時段長度，例如，三個小時時段 **3h**。時段不得延伸到下一個輪換時段。如果您未指定 Window duration (時段持續時間)，則對於以小時為單位的輪換排程，時段會在一小時後自動關閉。對於以天為單位的輪換排程，時段會在一天結束時自動關閉。
   d. (選用) 選擇 Rotate immediately when the secret is stored (存放秘密時立即輪換) 以在儲存變更時輪換您的秘密。如果清除核取方塊，則第一次輪換將按照您設定的排程開始。
   e. 在輪換函數下，選擇您在步驟 1 中建立的 Lambda 函數。
   f. 選擇儲存。

步驟 4：允許輪換函數存取 Secrets Manager 和您的資料庫或服務

Lambda 輪換函數需要許可以存取 Secrets Manager 中的秘密，且需要許可以存取您的資料庫或服務。在此步驟中，您會將這些許可授予 Lambda 執行角色。如果使用 KMS 金鑰為秘密加密，而不是 AWS 受管金鑰 `aws/secretsmanager`，那麼您需要將使用該金鑰的許可授予 Lambda 執行角色。您可以透過 SecretARN 加密內容來限制使用解密函數，從而使輪換函數角色僅有權解密其負責輪換的秘密。如需政策範例，請參閱輪換的許可。

如需指示，請參閱《AWS Lambda 開發人員指南》中的 Lambda 執行角色。

步驟 5：允許 Secrets Manager 叫用輪換函數

若要允許 Secrets Manager 根據您設定的輪換排程叫用輪換函數，您需要在 Lambda 函數的資源政策中授予`lambda:InvokeFunction`許可給 Secrets Manager 服務主體。

在輪換函數的資源政策中，我們建議您包含內容金鑰 `aws:SourceAccount`，協助防止 Lambda 被當作混淆代理人。對於某些 AWS 服務，為避免混淆代理人案例， AWS 建議您同時使用 `aws:SourceArn`和 `aws:SourceAccount`全域條件索引鍵。但是，如果在您的輪換函數政策中包

含 aws:SourceArn 條件，則輪換函數只能用於輪換該 ARN 指定的秘密。建議您僅包含內容金鑰 aws:SourceAccount，以便可以將輪換函數用於多個秘密。

若要將資源政策連接到 Lambda 函數，請參閱將資源型政策用於 Lambda。

下列政策允許 Secrets Manager 叫用 Lambda 函數。

JSON

```
{
    "Version": "2012-10-17",
    "Id": "default",
    "Statement": [
    {
        "Effect": "Allow",
        "Principal": {
            "Service": "secretsmanager.amazonaws.com"
            },
        "Action": "lambda:InvokeFunction",
        "Condition": {
            "StringEquals": {
                "AWS:SourceAccount": "123456789012"
            }
        },
        "Resource": "LambdaRotationFunctionARN"
    }
    ]
}
```

步驟 6：設定輪換函數的網路存取

在此步驟中，您可以允許 輪換函數同時連線到 Secrets Manager 和秘密所在的服務或資料庫。輪換函數必須能夠存取兩者，才能輪換秘密。請參閱 the section called “AWS Lambda 輪換函數的網路存取”。

後續步驟

當您在步驟 3 中設定輪換時，您可以設定輪換秘密的排程。如果輪換在排程時失敗，Secrets Manager 會多次嘗試輪換。您也可以依照 中的指示，立即開始輪換立即輪換秘密。

如果輪換失敗，請參閱 輪換疑難排解。

使用 設定自動輪換 AWS CLI

本教學說明如何the section called “依 Lambda 函數輪換”使用 設定 AWS CLI。當您輪換秘密時，會更新秘密以及該秘密所針對資料庫或服務中的憑證。

您也可以使用 主控台設定輪換。如需資料庫秘密，請參閱資料庫秘密的自動輪換 (主控台)。如需所有其他類型的機密，請參閱 非資料庫秘密的自動輪換 （ 主控台 ）。

若要使用 設定輪換 AWS CLI，如果您要輪換資料庫秘密，您必須先選擇輪換策略。如果您選擇交替使用者策略，必須儲存具有資料庫超級使用者憑證的另外秘密。接下來，您會撰寫輪換函數程式碼。Secrets Manager 提供範本，您可以用來撰寫函數。您接著可以使用程式碼建立 Lambda 函數，並為 Lambda 函數和 Lambda 執行角色設定許可。下一個步驟是確保 Lambda 函數可以透過網路存取 Secrets Manager 和資料庫或服務。最後，您可以設定秘密以進行輪換。

步驟：

- 資料庫秘密的先決條件：選擇輪換策略
- 步驟 1：撰寫輪換函數程式碼
- 步驟 2：建立 Lambda 函數
- 步驟 3：設定網路存取
- 步驟 4：設定輪換的秘密
- 後續步驟

資料庫秘密的先決條件：選擇輪換策略

如需 Secrets Manager 所提供策略的相關資訊，請參閱 the section called “Lambda 函數輪換策略”。

選項 1：單一使用者策略

如果您選擇單一使用者策略，您可以繼續進行步驟 1。

選項 2：交替使用者策略

如果您選擇交替使用者策略，您必須：

- 建立秘密，並將資料庫超級使用者登入資料存放在其中。您需要具有超級使用者登入資料的秘密，因為交替使用者輪換會複製第一個使用者，而且大多數使用者沒有該許可。
- 將超級使用者秘密的 ARN 新增至原始秘密。如需詳細資訊，請參閱the section called “機密的 JSON 結構”。

請注意，Amazon RDS Proxy 不支援交替使用者策略。

步驟 1：撰寫輪換函數程式碼

若要輪換秘密，您需要輪換函數。輪換函數是 Lambda 函數，Secrets Manager 會呼叫以輪換秘密。如需詳細資訊，請參閱the section called “依 Lambda 函數輪換”。在此步驟中，您會撰寫更新秘密的程式碼，以及秘密所要的服務或資料庫。

Secrets Manager 為 中的 Amazon RDS、Amazon Aurora、Amazon Redshift 和 Amazon DocumentDB 資料庫秘密提供範本輪換函數範本。

撰寫輪換函數程式碼

1. 執行以下任意一項：
   - 檢查輪換函數範本的清單。如果有符合您的服務和輪換策略，請複製程式碼。
   - 對於其他類型的秘密，您可以撰寫自己的輪換函數。如需說明，請參閱the section called “Lambda 輪換函數”。
2. 將檔案與所有必要的相依性一起儲存在 ZIP 檔案 *my-function.zip* 中。

步驟 2：建立 Lambda 函數

在此步驟中，您會使用您在步驟 1 中建立的 ZIP 檔案來建立 Lambda 函數。您也可以設定 Lambda 執行角色，這是 Lambda 在叫用函數時擔任的角色。

建立 Lambda 輪換函數和執行角色

1. 為 Lambda 執行角色建立信任政策，並將其儲存為 JSON 檔案。如需範例和詳細資訊，請參閱 the section called “輪換的許可”。該政策必須：
   - 允許角色對秘密呼叫 Secrets Manager 作業。
   - 例如，允許角色呼叫秘密的 服務，以建立新密碼。
2. 建立 Lambda 執行角色，並透過呼叫 來套用您在上一個步驟中建立的信任政策`iam create-role`。

```
aws iam create-role \
    --role-name rotation-lambda-role \
    --assume-role-policy-document file://trust-policy.json
```

3. 呼叫 `lambda create-function`，以 ZIP 檔案建立 Lambda 函數。

```
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. 對 Lambda 函數設定資源政策，允許 Secrets Manager 透過呼叫 `lambda add-permission` 來叫用該函數。

```
aws lambda add-permission \
  --function-name my-rotation-function \
  --action lambda:InvokeFunction \
  --statement-id SecretsManager \
  --principal secretsmanager.amazonaws.com \
  --source-account 123456789012
```

步驟 3：設定網路存取

如需詳細資訊，請參閱the section called “AWS Lambda 輪換函數的網路存取”。

步驟 4：設定輪換的秘密

若要為您的秘密開啟自動輪換功能，請呼叫 `rotate-secret`。您可以使用 `cron()` 或 `rate()` 排程表達式來設定輪換排程，也可以設定輪換時段時長。如需詳細資訊，請參閱the section called “輪換排程”。

```
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\"}"
```

後續步驟

請參閱the section called “輪換疑難排解”。

Lambda 函數輪換策略

對於 the section called “依 Lambda 函數輪換”，對於資料庫秘密，Secrets Manager 提供兩種輪換策略。

輪換策略：單一使用者

此策略會在一個秘密中更新一個使用者的憑證。對於 Amazon RDS Db2 執行個體，因為使用者無法變更自己的密碼，因此必須使用單獨的密碼來提供管理員登入資料。這是最簡單的輪換策略，適用於大多數使用案例。特別是，建議您將此策略用於一次性 (臨機操作) 或互動式使用者的憑證。

秘密輪換時，不會中斷開啟的資料庫連線。輪換正在進行時，資料庫中的密碼變更與更新秘密之間，有一小段時間落差。在此期間，資料庫有可能拒絕使用輪換後憑證的呼叫。您可以透過適當的重試策略降低這種風險。輪換之後，新的連線會使用新的憑證。

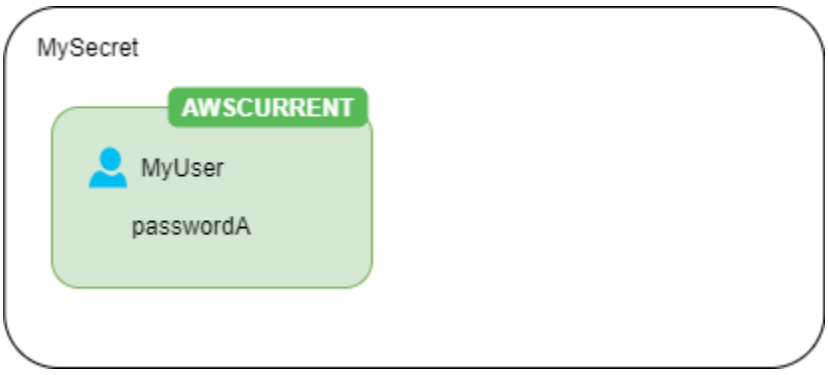
輪換策略：交替使用者

此策略會在一個秘密中更新兩個使用者的憑證。您可以建立第一個使用者，然後在第一次輪換期間，輪換函數會複製該使用者，以建立第二個使用者。每當秘密輪換時，輪換函數都會交替要更新的使用者密碼。由於大多數使用者沒有複製自身的許可，所以您必須提供其他秘密中 `superuser` 的憑證。在資料庫中複製的使用者沒有與原始使用者相同的許可時，建議使用單一使用者輪換策略，以及將其用於一次性 (臨機操作) 或互動式使用者的憑證。

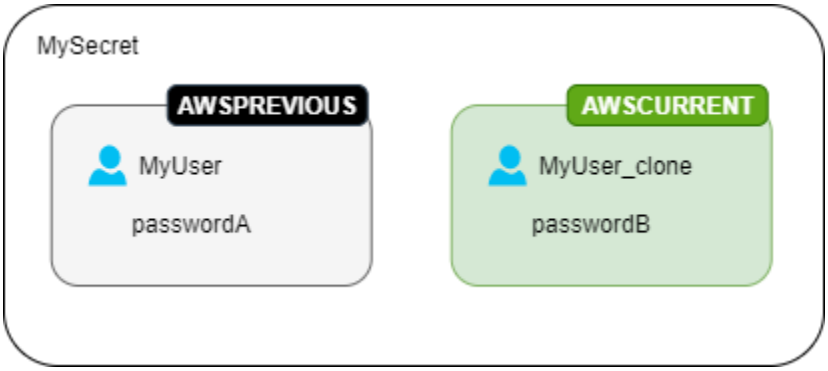
此策略適合具有許可模型的資料庫，其中一個角色擁有資料庫資料表，而第二個角色具有存取資料庫資料表的許可。這也適用於需要高可用性的應用程式。如果應用程式在輪換期間擷取秘密，應用程式仍會取得一組有效的憑證。輪換之後，`user` 和 `user_clone` 憑證都有效。在這種類型的輪換期間，應用程式遭到拒絕的機率比單一使用者輪換更低。如果資料庫託管於伺服器陣列，將密碼變更傳播到所有伺服器需要一段時間，則資料庫有可能拒絕使用新憑證的呼叫。您可以透過適當的重試策略降低這種風險。

Secrets Manager 會建立複製使用者，該使用者擁有與原始使用者相同的許可。如果在建立複製使用者後變更原始使用者的許可，您也必須變更複製使用者的許可。

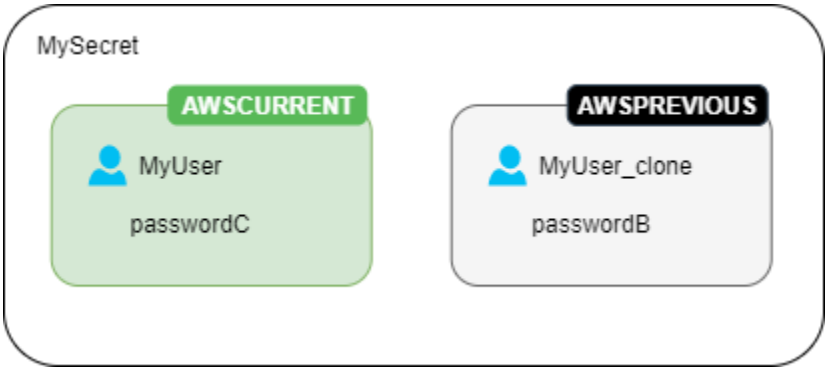
例如，如果您使用資料庫使用者的憑證建立秘密，則該秘密會包含一個具有這些憑證的版本。



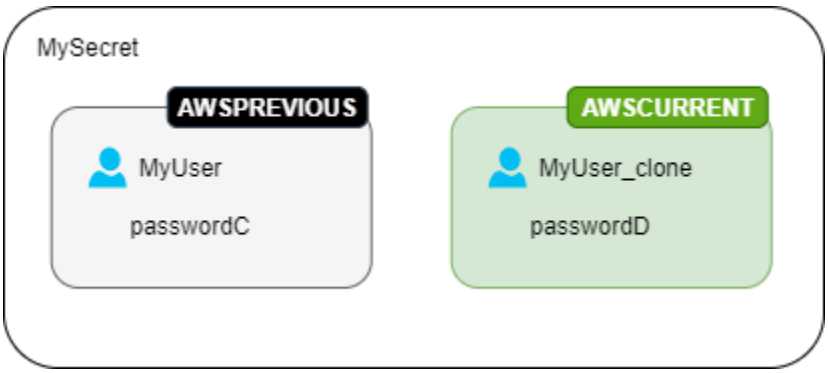
第一次輪換 – 輪換函數會使用產生的密碼來建立使用者的複製，而這些登入資料會成為目前的秘密版本。



第二次輪換 – 輪換函數會更新原始使用者的密碼。



第三次輪換 – 輪換函數會更新複製使用者的密碼。



Lambda 輪換函數

在 中the section called “依 Lambda 函數輪換”， AWS Lambda 函數會輪換秘密。 AWS Secrets Manager 會在輪換期間使用預備標籤來識別秘密版本。

如果 AWS Secrets Manager 未為您的秘密類型提供輪換函數範本，您可以建立自訂輪換函數。撰寫輪換函數時，請遵循下列準則：

自訂輪換函數的最佳實務

- 使用一般輪換範本做為起點。
- 請小心偵錯或記錄陳述式。他們可以將資訊寫入 Amazon CloudWatch Logs。確保日誌不包含敏感資訊。

  如需日誌陳述式範例，請參閱the section called “輪換函數範本”原始程式碼。
- 為了安全起見， AWS Secrets Manager 只允許 Lambda 輪換函數直接輪換秘密。輪換函數無法呼叫另一個 Lambda 函數來輪換秘密。
- 如需偵錯指引，請參閱測試和偵錯無伺服器應用程式。
- 如果您使用外部二進位檔和程式庫，例如連接到資源，則需負責修補和更新它們。
- 將您的輪換函數和任何相依性封裝在 ZIP 檔案中，例如 *my-function.zip*。

> **Warning**
>
> 由於 Lambda 函數的執行執行緒不足，將佈建並行參數設定為低於 10 的值可能會導致限流。如需詳細資訊，請參閱《 AWS Lambda 開發人員指南》中的 AWS Lambda 了解預留並行和佈建並行。

輪換函數中的四個步驟

主題

- createSecret：建立新的秘密版本
- setSecret：變更資料庫或服務中的登入資料
- testSecret：測試新的秘密版本
- finishSecret：完成輪換

createSecret：建立新的秘密版本

方法createSecret會先get_secret_value透過使用傳入 呼叫 來檢查秘密是否存在ClientRequestToken。如果沒有秘密，它會使用 和create_secret字符作為 建立新的秘密VersionId。然後，它會使用 產生新的秘密值get_random_password。接下來，它會呼叫 put_secret_value，將其與預備標籤 一起存放AWSPENDING。將新的秘密值儲存在 AWSPENDING 中，有助於確保等冪性。如果輪換因任何原因而失敗，您可以在後續呼叫中參考該秘密值。請參閱 How do I make my Lambda function idempotent (如何讓 Lambda 函數等冪)。

撰寫您自己的輪換函數的提示

- 確保新的秘密值僅包含對資料庫或服務有效的字元。使用 ExcludeCharacters 參數排除字元。
- 當您測試函數時，請使用 AWS CLI 查看版本階段：呼叫 describe-secret並查看 VersionIdsToStages。
- 對於 Amazon RDS MySQL，在交替使用者輪換時，Secrets Manager 會建立名稱不超過 16 個字元的複製使用者。您可以修改旋轉功能以允許更長的用戶名。MySQL 版本 5.7 及更高版本支持用戶名最多 32 個字符，但是 Secrets Manager 附加「_clone」（六個字符）到用戶名的末尾，所以你必須保持用戶名最多 26 個字符。

setSecret：變更資料庫或服務中的登入資料

方法會setSecret變更資料庫或服務中的登入資料，以符合秘密AWSPENDING版本中的新秘密值。

撰寫您自己的輪換函數的提示

- 如果您將陳述式傳遞給解譯陳述式的服務，例如資料庫，請使用查詢參數化。如需詳細資訊，請參閱 OWASP 網站上的查詢參數摘要。

- 輪換函數是具有特殊權限的代理人，有權存取和修改 Secrets Manager 秘密和目標資源中的客戶憑證。為了防止潛在的混淆代理人攻擊，您必須確保攻擊者無法使用該函數存取其他資源。在您更新憑證之前：
  - 查看 AWSCURRENT 版本秘密中的憑證是否有效。如果 AWSCURRENT 憑證無效，請放棄輪換嘗試。
  - 查看 AWSCURRENT 和 AWSPENDING 秘密值是否針對相同的資源。對於使用者名稱和密碼，請查看 AWSCURRENT 和 AWSPENDING 使用者名稱是否相同。
  - 檢查目的地服務資源是否相同。對於資料庫，請查看 AWSCURRENT 和 AWSPENDING 主機名稱是否相同。
- 在極少數情況下，您可能想要自訂資料庫的現有輪換函數。例如，使用者交替輪換時，Secrets Manager 會複製第一個使用者的執行期組態參數來建立複製的使用者。如果您想要包含更多屬性，或變更哪些屬性以授予複製的使用者，則需要更新 `set_secret` 函數中的程式碼。

testSecret：測試新的秘密版本

接下來，Lambda 輪換函數會使用 AWSPENDING 版本的秘密存取資料庫或服務，以進行測試。根據 輪換函數範本 建立的輪換函數會使用讀取權限測試新的秘密。

finishSecret：完成輪換

最後，Lambda 輪換函數會將標籤 AWSCURRENT 從先前的秘密版本移至此版本，這也會移除相同 API 呼叫中的 AWSPENDING 標籤。Secrets Manager 會將 AWSPREVIOUS 預備標籤新增至先前版本，以便您保留上一個已知良好的秘密版本。

方法finish_secret使用 將預備標籤AWSCURRENT從先前的秘密版本`update_secret_version_stage`移至新的秘密版本。Secrets Manager 會自動將 AWSPREVIOUS 預備標籤新增至先前版本，以便您保留上一個已知良好的秘密版本。

撰寫您自己的輪換函數的提示

- 在此AWSPENDING之前，請勿移除 ，也不要使用單獨的 API 呼叫來移除它，因為這可能會向 Secrets Manager 指出輪換未成功完成。Secrets Manager 會將 AWSPREVIOUS 預備標籤新增至先前版本，以便您保留上一個已知良好的秘密版本。

輪換成功時，AWSPENDING 預備標籤可能會連接至與 AWSCURRENT 版本相同的版本，或者可能未連接至任何版本。如果 AWSPENDING 預備標籤存在，但未連接至與 AWSCURRENT 相同的版本，則任何以後的輪換調用都會假定之前的輪換請求仍在進行中並傳回錯誤。輪換不成功時，AWSPENDING 預備標籤可能會連接至空的機密版本。如需詳細資訊，請參閱輪換疑難排解。

AWS Secrets Manager 輪換函數範本

AWS Secrets Manager 提供一組輪換函數範本，可協助自動化各種資料庫系統和服務的登入資料安全管理。範本是ready-to-use Lambda 函數，可實作登入資料輪換的最佳實務，協助您維持安全狀態，無需手動介入。

範本支援兩種主要輪換策略：

- 單一使用者輪換，更新單一使用者的登入資料。
- 交替使用者輪換，可維護兩個不同的使用者，以協助消除登入資料變更期間的停機時間。

Secrets Manager 也提供一般範本，做為任何類型的秘密的起點。

若要使用範本，請參閱：

- 資料庫秘密的自動輪換 (主控台)
- 非資料庫秘密的自動輪換 （ 主控台 ）

若要寫入您自己的輪換函數，請參閱寫入輪換函數。

範本

- Amazon RDS 和 Amazon Aurora
  - Amazon RDS Db2 單用戶
  - Amazon RDS Db2 交替用戶
  - Amazon RDS MariaDB 單一使用者
  - Amazon RDS MariaDB 交替使用者
  - Amazon RDS 和 Amazon Aurora MySQL 單一使用者
  - Amazon RDS 和 Amazon Aurora MySQL 交替使用者
  - Amazon RDS Oracle 單一使用者
  - Amazon RDS Oracle 交替使用者
  - Amazon RDS 和 Amazon Aurora PostgreSQL 單一使用者
  - Amazon RDS 和 Amazon Aurora PostgreSQL 交替使用者
  - Amazon RDS Microsoft SQLServer 單一使用者
  - Amazon RDS Microsoft SQLServer 交替使用者
- Amazon DocumentDB (with MongoDB compatibility)

  - Amazon DocumentDB 單一使用者
  - Amazon DocumentDB 交替使用者
- Amazon Redshift
  - Amazon Redshift 單一使用者
  - Amazon Redshift 交替使用者
- InfluxDB 的 Amazon Timestream
  - InfluxDB 單一使用者的 Amazon Timestream
  - InfluxDB 交替使用者的 Amazon Timestream
- Amazon ElastiCache
- Active Directory
  - Active Directory 登入資料
  - Active Directory keytab
- 其他類型的秘密

Amazon RDS 和 Amazon Aurora

Amazon RDS Db2 單用戶

- 範本名稱：SecretsManagerRDSDb2RotationSingleUser
- 輪換策略：輪換策略：單一使用者。
- **`SecretString`** 結構：the section called “Amazon RDS 和 Aurora 登入資料”。
- 原始碼：https：//https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSDb2RotationSingleUser/lambda_function.py
- 依賴關係：python-ibmdb

Amazon RDS Db2 交替用戶

- 範本名稱：SecretsManagerRDSDb2RotationMultiUser
- 輪換策略：the section called “交替使用者”。
- **`SecretString`** 結構：the section called “Amazon RDS 和 Aurora 登入資料”。
- 原始碼：https：//https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSDb2RotationMultiUser/lambda_function.py

- 依賴關係：python-ibmdb

Amazon RDS MariaDB 單一使用者

- 範本名稱：SecretsManagerRDSMariaDBRotationSingleUser
- 輪換策略：輪換策略：單一使用者。
- **SecretString** 結構：the section called “Amazon RDS 和 Aurora 登入資料”。
- 原始碼：https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSMariaDBRotationSingleUser/lambda_function.py
- 相依性：PyMySQL 1.0.2。如果您使用 sha256 密碼進行身分驗證，PyMySQL【rsa】。如需在 Lambda 執行時間中使用套件搭配編譯程式碼的資訊，請參閱 AWS 知識中心中的如何將具有編譯二進位檔的 Python 套件新增至部署套件，並讓套件與 Lambda 相容？。

Amazon RDS MariaDB 交替使用者

- 範本名稱：SecretsManagerRDSMariaDBRotationMultiUser
- 輪換策略：the section called “交替使用者”。
- **SecretString** 結構：the section called “Amazon RDS 和 Aurora 登入資料”。
- 原始碼：https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSMariaDBRotationMultiUser/lambda_function.py
- 相依性：PyMySQL 1.0.2。如果您使用 sha256 密碼進行身分驗證，PyMySQL【rsa】。如需在 Lambda 執行時間中使用套件搭配編譯程式碼的資訊，請參閱 AWS 知識中心中的如何將具有編譯二進位檔的 Python 套件新增至部署套件，並讓套件與 Lambda 相容？。

Amazon RDS 和 Amazon Aurora MySQL 單一使用者

- 範本名稱：SecretsManagerRDSMySQLRotationSingleUser
- 輪換策略：the section called “單一使用者”。
- 預期的 **SecretString** 結構：the section called “Amazon RDS 和 Aurora 登入資料”。
- 原始碼：https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSMySQLRotationSingleUser/lambda_function.py
- 相依性：PyMySQL 1.0.2。如果您使用 sha256 密碼進行身分驗證，PyMySQL【rsa】。如需在 Lambda 執行時間中搭配編譯程式碼使用套件的詳細資訊，請參閱 AWS 知識中心中的如何將具有編譯二進位檔的 Python 套件新增至部署套件，並讓套件與 Lambda 相容？。

Amazon RDS 和 Amazon Aurora MySQL 交替使用者

- 範本名稱：SecretsManagerRDSMySQLRotationMultiUser
- 輪換策略：the section called “交替使用者”。
- 預期的 **SecretString** 結構：the section called “Amazon RDS 和 Aurora 登入資料”。
- 原始碼：https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSMySQLRotationMultiUser/lambda_function.py
- 相依性：PyMySQL 1.0.2。如果您使用 sha256 密碼進行身分驗證，PyMySQL【rsa】。如需在 Lambda 執行時間中搭配編譯程式碼使用套件的資訊，請參閱 AWS 知識中心中的如何將具有編譯二進位檔的 Python 套件新增至部署套件，並讓套件與 Lambda 相容？。

Amazon RDS Oracle 單一使用者

- 範本名稱：SecretsManagerRDSOracleRotationSingleUser
- 輪換策略：the section called “單一使用者”。
- 預期的 **SecretString** 結構：the section called “Amazon RDS 和 Aurora 登入資料”。
- 原始碼：https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSOracleRotationSingleUser/lambda_function.py
- 相依性：python-oracledb 2.4.1

Amazon RDS Oracle 交替使用者

- 範本名稱：SecretsManagerRDSOracleRotationMultiUser
- 輪換策略：the section called “交替使用者”。
- 預期的 **SecretString** 結構：the section called “Amazon RDS 和 Aurora 登入資料”。
- 原始碼：https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSOracleRotationMultiUser/lambda_function.py
- 相依性：python-oracledb 2.4.1

Amazon RDS 和 Amazon Aurora PostgreSQL 單一使用者

- 範本名稱：SecretsManagerRDSPostgreSQLRotationSingleUser
- 輪換策略：輪換策略：單一使用者。
- 預期的 **SecretString** 結構：the section called “Amazon RDS 和 Aurora 登入資料”。

- 原始碼：https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSPostgreSQLRotationSingleUser/lambda_function.py
- 相依性：PyGreSQL 5.0.7

Amazon RDS 和 Amazon Aurora PostgreSQL 交替使用者

- 範本名稱：SecretsManagerRDSPostgreSQLRotationMultiUser
- 輪換策略：the section called “交替使用者”。
- 預期的 **`SecretString`** 結構：the section called “Amazon RDS 和 Aurora 登入資料”。
- 原始碼：https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSPostgreSQLRotationMultiUser/lambda_function.py
- 相依性：PyGreSQL 5.0.7

Amazon RDS Microsoft SQLServer 單一使用者

- 範本名稱：SecretsManagerRDSSQLServerRotationSingleUser
- 輪換策略：the section called “單一使用者”。
- 預期的 **`SecretString`** 結構：the section called “Amazon RDS 和 Aurora 登入資料”。
- 原始碼：https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSSQLServerRotationSingleUser/lambda_function.py
- 相依性：Pymssql 2.2.2

Amazon RDS Microsoft SQLServer 交替使用者

- 範本名稱：SecretsManagerRDSSQLServerRotationMultiUser
- 輪換策略：the section called “交替使用者”。
- 預期的 **`SecretString`** 結構：the section called “Amazon RDS 和 Aurora 登入資料”。
- 原始碼：https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRDSSQLServerRotationMultiUser/lambda_function.py
- 相依性：Pymssql 2.2.2

Amazon DocumentDB (with MongoDB compatibility)

Amazon DocumentDB 單一使用者

- 範本名稱：SecretsManagerMongoDBRotationSingleUser
- 輪換策略：the section called “單一使用者”。
- 預期的 **`SecretString`** 結構：the section called “Amazon DocumentDB 登入資料”。
- 原始碼：https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerMongoDBRotationSingleUser/lambda_function.py
- 相依性：Pymongo 3.2

Amazon DocumentDB 交替使用者

- 範本名稱：SecretsManagerMongoDBRotationMultiUser
- 輪換策略：the section called “交替使用者”。
- 預期的 **`SecretString`** 結構：the section called “Amazon DocumentDB 登入資料”。
- 原始碼：https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerMongoDBRotationMultiUser/lambda_function.py
- 相依性：Pymongo 3.2

Amazon Redshift

Amazon Redshift 單一使用者

- 範本名稱：SecretsManagerRedshiftRotationSingleUser
- 輪換策略：the section called “單一使用者”。
- 預期的 **`SecretString`** 結構：the section called “Amazon Redshift 登入資料”。
- 原始碼：https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRedshiftRotationSingleUser/lambda_function.py
- 相依性：PyGreSQL 5.0.7

Amazon Redshift 交替使用者

- 範本名稱：SecretsManagerRedshiftRotationMultiUser

- 輪換策略：the section called “交替使用者”。
- 預期的 **`SecretString`** 結構：the section called “Amazon Redshift 登入資料”。
- 原始碼：https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRedshiftRotationMultiUser/lambda_function.py
- 相依性：PyGreSQL 5.0.7

InfluxDB 的 Amazon Timestream

若要使用這些範本，請參閱《Amazon Timestream 開發人員指南》中的 Amazon Timestream for InfluxDB 如何使用秘密。

InfluxDB 單一使用者的 Amazon Timestream

- 範本名稱：SecretsManagerInfluxDBRotationSingleUser
- 預期的 **`SecretString`** 結構：the section called “InfluxDB 秘密結構的 Amazon Timestream”。
- 原始碼：https：//https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerInfluxDBRotationSingleUser/lambda_function.py
- 相依性：InfluxDB 2.0 python 用戶端

InfluxDB 交替使用者的 Amazon Timestream

- 範本名稱：SecretsManagerInfluxDBRotationMultiUser
- 預期的 **`SecretString`** 結構：the section called “InfluxDB 秘密結構的 Amazon Timestream”。
- 原始碼：https：//https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerInfluxDBRotationMultiUser/lambda_function.py
- 相依性：InfluxDB 2.0 python 用戶端

Amazon ElastiCache

若要使用此範本，請參閱《Amazon ElastiCache 使用者指南》中的自動輪換使用者的密碼。

- 範本名稱：SecretsManagerElasticacheUserRotation
- 預期的 **`SecretString`** 結構：the section called “Amazon ElastiCache 登入資料”。
- 原始碼：https：//https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerElasticacheUserRotation/lambda_function.py

Active Directory

Active Directory 登入資料

- 範本名稱：SecretsManagerActiveDirectoryRotationSingleUser
- 預期的 **`SecretString`** 結構：the section called “Active Directory 登入資料”。
- 原始碼：https：//https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerActiveDirectoryRotationSingleUser/lambda_function.py

Active Directory keytab

- 範本名稱：SecretsManagerActiveDirectoryAndKeytabRotationSingleUser
- 預期的 **`SecretString`** 結構：the section called “Active Directory 登入資料”。
- 原始碼：https：//https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerActiveDirectoryAndKeytabRotationSingleUser/lambda_function.py
- 相依性：msktutil

其他類型的秘密

Secrets Manager 提供此範本做為起點，供您為任何類型的秘密建立輪換函數。

- 範本名稱：SecretsManagerRotationTemplate
- 原始碼：https://github.com/aws-samples/aws-secrets-manager-rotation-lambdas/tree/master/SecretsManagerRotationTemplate/lambda_function.py

的 Lambda 輪換函數執行角色許可 AWS Secrets Manager

對於 the section called “依 Lambda 函數輪換”，當 Secrets Manager 使用 Lambda 函數來輪換秘密時，Lambda 會擔任 IAM 執行角色，並將這些登入資料提供給 Lambda 函數程式碼。如需如何設定自動輪換的指示，請參閱：

- 資料庫秘密的自動輪換 (主控台)
- 非資料庫秘密的自動輪換 （ 主控台 ）
- 自動輪換 (AWS CLI)

下列範例顯示 Lambda 輪換函數執行角色的內嵌政策。若要建立執行角色並附加許可政策，請參閱 AWS Lambda 執行角色。

範例：

- Lambda 輪換函數執行角色的政策
- 客戶管理金鑰的政策陳述式
- 交替使用者策略的政策陳述式

Lambda 輪換函數執行角色的政策

下列範例政策允許輪換函數：

- 為 *SecretARN* 執行 Secrets Manager 作業。
- 建立新密碼。
- 如果資料庫或服務在 VPC 內執行，則設定必要組態。請參閱設定 Lambda 函數以存取 VPC 中的資源。

JSON

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

```
        },
        {
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterface",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DetachNetworkInterface"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

客戶管理金鑰的政策陳述式

如果使用 KMS 金鑰為秘密加密，而不是 AWS 受管金鑰 `aws/secretsmanager`，那麼您需要將使用該金鑰的許可授予 Lambda 執行角色。您可以透過 SecretARN 加密內容來限制使用解密函數，從而使輪換函數角色僅有權解密其負責輪換的秘密。下列範例顯示要新增至執行角色政策，以使用 KMS 金鑰解密秘密的陳述式。

```
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey"
            ],
            "Resource": "KMSKeyARN"
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:SecretARN": "SecretARN"
                }
            }
        }
```

若要針對使用客戶受管金鑰加密的多組秘密使用輪換函數，請新增如下列範例所示的陳述式，以允許執行角色將秘密解密。

```
        {
            "Effect": "Allow",
```

```
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey"
            ],
            "Resource": "KMSKeyARN"
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:SecretARN": [
                        "arn1",
                        "arn2"
                    ]
                }
            }
        }
```

交替使用者策略的政策陳述式

如需交替使用者輪換策略的資訊，請參閱the section called “Lambda 函數輪換策略”。

對於包含 Amazon RDS 憑證的機密，如果您使用交替使用者策略，且超級使用者機密由 Amazon RDS 管理，則還必須允許輪換函數呼叫 Amazon RDS 上的唯讀 API，以便取得資料庫的連線資訊。建議您連接 AWS 受管政策 AmazonRDSReadOnlyAccess。

下列範例政策允許函數：

- 為 *SecretARN* 執行 Secrets Manager 作業。
- 擷取超級使用者秘密中的憑證。Secrets Manager 會使用超級使用者秘密中的憑證，更新輪換後秘密中的憑證。
- 建立新密碼。
- 如果資料庫或服務在 VPC 內執行，則設定必要組態。如需詳細資訊，請參閱設定 Lambda 函數以存取 VPC 中的資源。

JSON

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

```
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "secretsmanager:UpdateSecretVersionStage"
            ],
            "Resource": "SecretARN"
        },
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource": "SuperuserSecretARN"
        },
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetRandomPassword"
            ],
            "Resource": "*"
        },
        {
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterface",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DetachNetworkInterface"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

AWS Lambda 輪換函數的網路存取

對於 the section called “依 Lambda 函數輪換”，當 Secrets Manager 使用 Lambda 函數來輪換秘密時，Lambda 輪換函數必須能夠存取秘密。如果您的秘密包含憑證，則 Lambda 函數也必須能夠存取這些憑證的來源，例如資料庫或服務。

存取秘密

您的 Lambda 輪換函數必須能夠存取 Secrets Manager 服務端點。如果您的 Lambda 函數可以存取網際網路，那麼您可以使用公有端點。若要尋找端點，請參閱 the section called “Secrets Manager 端點”。

如果 Lambda 函數在無法存取網際網路的 VPC 中執行，建議您在 VPC 中設定 Secrets Manager 服務私有端點。然後，您的 VPC 可以攔截發送到公有區域端點的請求，並將其重新導向到私有端點。如需詳細資訊，請參閱VPC 端點 (AWS PrivateLink)。

或者，您可以啟用 Lambda 函數來存取 Secrets Manager 公有端點，方法是將 NAT 閘道或網際網路閘道新增至您的 VPC，以便來自 VPC 的流量到達公有端點。這將使 VPC 暴露在較高的風險下，因為 IP 地址 (用於閘道) 可能會遭到來自公有網際網路的攻擊。

(選用) 存取資料庫或服務

對於 API 金鑰等秘密，沒有需要與秘密一起更新的來源資料庫或服務。

如果資料庫或服務正在 VPC 中的 Amazon EC2 執行個體上執行，建議您將 Lambda 函數設定為在相同的 VPC 中執行。然後輪換函數就能直接與您的服務進行通訊。如需詳細資訊，請參閱設定 VPC 存取 。

若要允許 Lambda 函數存取資料庫或服務，您必須確定附加至 Lambda 輪換函數的安全群組允許連至資料庫或服務的傳出連線。此外，您必須確定附加至資料庫或服務的安全群組允許來自 Lambda 輪換函數的傳入連線。

對 AWS Secrets Manager 輪換進行故障診斷

對許多服務而言，Secrets Manager 使用 Lambda 函數來輪換秘密。如需詳細資訊，請參閱the section called “依 Lambda 函數輪換”。Lambda 輪換函數會與秘密適用的資料庫或服務，以及 Secrets Manager 互動。並未如您預期般輪換時，您應先查看 CloudWatch 日誌。

> **Note**
>
> 某些服務可以為您管理秘密，包括管理自動輪換。如需詳細資訊，請參閱the section called “受管輪換”。

主題

- 如何在 AWS Lambda 函數中對秘密輪換失敗進行疑難排解
- 「在環境變數中找到憑證」之後沒有活動
- "createSecret" 之後沒有任何活動
- 錯誤：「不允許存取 KMS」
- 錯誤：「秘密 JSON 缺少金鑰」
- 錯誤：「setSecret：無法登入資料庫」
- 錯誤：「無法匯入模組 'lambda_function'」
- 將現有的輪換函數從 Python 3.7 升級至 3.9
- AWS Lambda 秘密輪換PutSecretValue失敗
- 錯誤：「在 <a rotation> 步驟期間執行 lambda <arn> 時發生錯誤」

如何在 AWS Lambda 函數中對秘密輪換失敗進行疑難排解

如果您的 Lambda 函數發生秘密輪換失敗，請使用下列步驟進行故障診斷並解決問題。

可能原因

- Lambda 函數的並行執行不足
- 輪換期間多次 API 呼叫造成的競賽條件
- Lambda 函數邏輯不正確
- Lambda 函數與資料庫之間的聯網問題

一般故障診斷步驟

1. 分析 CloudWatch 日誌：
   - 在 Lambda 函數日誌中尋找特定錯誤訊息或非預期行為
   - 確認正在嘗試所有輪換步驟 (CreateSecret、SetSecret、TestSecret、FinishSecret)
2. 在輪換期間檢閱 API 呼叫：
   - 避免在 Lambda 輪換期間對秘密進行變動 API 呼叫
   - 確保 RotateSecret和 PutSecretValue呼叫之間沒有競爭條件
3. 驗證 Lambda 函數邏輯：
   - 確認您使用最新的 AWS 範例程式碼進行秘密輪換

- 如果使用自訂程式碼，請檢閱它以正確處理所有輪換步驟

4. 檢查網路組態：
   - 驗證安全群組規則允許 Lambda 函數存取資料庫
   - 確保 Secrets Manager 的適當 VPC 端點或公有端點存取
5. 測試秘密版本：
   - 確認秘密的 AWSCURRENT 版本允許資料庫存取
   - 檢查 AWSPREVIOUS 或 AWSPENDING 版本是否有效
6. 清除待定輪換：
   - 如果輪換持續失敗，請清除 AWSPENDING 預備標籤，然後重試輪換
7. 檢查 Lambda 並行設定：
   - 驗證並行設定是否適合您的工作負載
   - 如果您懷疑並行問題，請參閱「疑難排解並行相關輪換失敗」一節

「在環境變數中找到憑證」之後沒有活動

如果「在環境變數中找到憑證」之後沒有活動，且任務持續時間很長 (例如預設 Lambda 逾時為 30000 毫秒)，則 Lambda 函數可能會在嘗試連線 Secrets Manager 端點時逾時。

您的 Lambda 輪換函數必須能夠存取 Secrets Manager 服務端點。如果您的 Lambda 函數可以存取網際網路，那麼您可以使用公有端點。若要尋找端點，請參閱 the section called “Secrets Manager 端點”。

如果 Lambda 函數在無法存取網際網路的 VPC 中執行，建議您在 VPC 中設定 Secrets Manager 服務私有端點。然後，您的 VPC 可以攔截發送到公有區域端點的請求，並將其重新導向到私有端點。如需詳細資訊，請參閱VPC 端點 (AWS PrivateLink)。

或者，您可以啟用 Lambda 函數來存取 Secrets Manager 公有端點，方法是將 NAT 閘道或網際網路閘道新增至您的 VPC，以便來自 VPC 的流量到達公有端點。這將使 VPC 暴露在較高的風險下，因為 IP 地址 (用於閘道) 可能會遭到來自公有網際網路的攻擊。

"createSecret" 之後沒有任何活動

以下是可能導致輪換在 CreateSecret 之後停止的問題：

VPC 網路 ACL 不允許傳入和傳出 HTTPS 流量。

如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的使用網路 ACL 控制子網路的流量。

Lambda 函數逾時組態太短，無法執行任務。

如需詳細資訊，請參閱《AWS Lambda 開發人員指南》中的設定 Lambda 函數選項。

Secrets Manager VPC 端點不允許在所指派安全群組的輸入上使用 VPC CIDR。

如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的使用安全群組控制到資源的流量。

Secrets Manager VPC 端點政策不允許 Lambda 使用 VPC 端點。

如需詳細資訊，請參閱the section called “VPC 端點 (AWS PrivateLink)”。

此機密使用交替使用者輪換，超級使用者機密由 Amazon RDS 管理，而 Lambda 函數無法存取 RDS API。

對於由其他服務管理超級使用者秘密的 AWS交替使用者輪換，Lambda 輪換函數必須能夠呼叫服務端點以取得資料庫連線資訊。建議您為資料庫服務設定 VPC 端點。如需詳細資訊，請參閱：

- 在《Amazon RDS 使用者指南》中的 Amazon RDS API 和介面 VPC 端點。
- 在 Amazon Redshift 管理指南中使用 VPC 端點。

錯誤：「不允許存取 KMS」

如果顯示 `ClientError: An error occurred (AccessDeniedException) when calling the GetSecretValue operation: Access to KMS is not allowed`，則輪換函數沒有使用用於加密秘密的 KMS 金鑰來解密秘密的許可。許可政策可能包含一個將加密內容限制為特定秘密的條件。如需有關必要許可的資訊，請參閱the section called “客戶管理金鑰的政策陳述式”。

錯誤：「秘密 JSON 缺少金鑰」

Lambda 輪換函數需要秘密值位於特定的 JSON 結構。如果您看到此錯誤，JSON 可能缺少輪換函數先前嘗試存取的金鑰。如需每種秘密類型的 JSON 結構相關資訊，請參閱the section called “機密的 JSON 結構”。

錯誤：「setSecret：無法登入資料庫」

以下是可能導致此錯誤的問題：

輪換函數無法存取資料庫。

如果任務持續時間很長 (例如超過 5000 毫秒)，Lambda 輪換函數可能無法透過網路存取資料庫。

如果資料庫或服務正在 VPC 中的 Amazon EC2 執行個體上執行，建議您將 Lambda 函數設定為在相同的 VPC 中執行。然後輪換函數就能直接與您的服務進行通訊。如需詳細資訊，請參閱設定 VPC 存取 。

若要允許 Lambda 函數存取資料庫或服務，您必須確定附加至 Lambda 輪換函數的安全群組允許連至資料庫或服務的傳出連線。此外，您必須確定附加至資料庫或服務的安全群組允許來自 Lambda 輪換函數的傳入連線。

秘密中的憑證不正確。

如果任務持續時間很短，Lambda 輪換函數可能無法使用秘密中的憑證來驗證。使用 AWS CLI 命令 手動登入 AWSCURRENT和秘密AWSPREVIOUS版本中的資訊，以檢查登入資料get-secret-value。

資料庫使用 **scram-sha-256** 來加密機密。

如果資料庫是 Aurora PostgreSQL 13 版或更新版本，並且使用 scram-sha-256 加密密碼，但輪換函數使用不支援 scram-sha-256 的 libpq 9 版或更舊版本，則輪換函數無法連接至資料庫。

判斷哪些資料庫使用者使用 **scram-sha-256** 加密

- 請參閱 SCRAM Authentication in RDS for PostgreSQL 13 (RDS for PostgreSQL 13 中的 SCRAM 身分驗證) 中的 Checking for users with non-SCRAM passwords (檢查具有非 SCRAM 密碼的使用者)。

判斷輪換函數使用的 **libpq** 版本

1. 在 Linux 電腦上的 Lambda 主控台上，導覽至輪換函數並下載部署套件。將 zip 檔案解壓縮到工作目錄。
2. 在命令列，在工作目錄中執行：

   ```
   readelf -a libpq.so.5 | grep RUNPATH
   ```

3. 如果您看到字串 *PostgreSQL-9.4.x* 或任何小於 10 的主要版本，則輪換函數不支援 scram-sha-256。
   - 不支援 scram-sha-256 的輪換函數的輸出：

     ```
     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]
```

- 支援 scram-sha-256 的輪換函數的輸出：

```
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 日之前設定自動秘密輪換，輪換函數會綁定不支援 libpq的舊版 scram-sha-256。若要支援 scram-sha-256，您需要重新建立輪換函數。

資料庫需要 SSL/TLS 存取權限。

如果資料庫需要 SSL/TLS 連線，但輪換函數使用未加密的連線，則輪換函數無法連線至資料庫。Amazon RDS (Oracle 和 Db2 除外) 和 Amazon DocumentDB 的輪換函數自動使用 Secure Socket Layer (SSL) 或 Transport Layer Security (TLS) 來連線到您的資料庫 (如果有)。否則，他們會使用未加密的連線。

> **Note**
>
> 如果您在 2021 年 12 月 20 日之前設定自動秘密輪換，您的輪換函數可能基於不支援 SSL/TLS 的較早範本。若要支援使用 SSL/TLS 的連線，您需要重建輪換函數。

確定輪換函數的建立時間

1. 在 Secrets Manager 主控台 https://console.aws.amazon.com/secretsmanager/，開啟您的秘密。在 Rotation configuration (輪換組態) 區段，在 Lambda rotation function (Lambda 輪換函數) 下，您會看到 Lambda function ARN (Lambda 函數 ARN)，例如 arn:aws:lambda:*aws-region*:*123456789012*:function:*SecretsManagerMyRotationFunction* 。在此範例 *SecretsManagerMyRotationFunction* 中，從 ARN 末尾複製函數名稱。

2. 在 AWS Lambda 主控台 https://console.aws.amazon.com/lambda/ 的函數下，將 Lambda 函數名稱貼到搜尋方塊中，選擇 Enter，然後選擇 Lambda 函數。
3. 在函數詳細資訊頁面上，Configuration (組態) 欄標中，Tags (標籤) 下，複製 aws:cloudformation:stack-name 索引鍵旁的值。
4. 在 AWS CloudFormation 主控台 https://console.aws.amazon.com/cloudformation 的 Stacks 下，將索引鍵值貼到搜尋方塊中，然後選擇 Enter。
5. 篩選堆疊清單，以便只顯示建立 Lambda 輪換函數的堆疊。在 Created date (建立日期) 欄中，檢視堆疊建立的日期。這是建立 Lambda 輪換函數的日期。

錯誤：「無法匯入模組 'lambda_function'」

如果您執行的是先前的 Lambda 函數，由於函數版本已從 Python 3.7 自動升級到較新版本的 Python，則可能會收到此錯誤。若要解決錯誤，您可以將 Lambda 函數版本變更回 Python 3.7，然後 the section called “將現有的輪換函數從 Python 3.7 升級至 3.9”。如需詳細資訊，請參閱 AWS re:Post 中的為什麼我的 Secrets Manager Lambda 函數輪換失敗並顯示「找不到 pg 模組」錯誤？文章。

將現有的輪換函數從 Python 3.7 升級至 3.9

在 2022 年 11 月之前建立的一些輪換函數使用 Python 3.7。適用於 Python 的 AWS SDK 已於 2023 年 12 月停止支援 Python 3.7。如需詳細資訊，請參閱 Python 支援 AWS SDKs和工具的政策更新。若要切換為使用 Python 3.9 的新輪換函數，您可以將執行期屬性新增至現有的輪換函數或重新建立輪換函數。

若要查看哪些 Lambda 輪換函數使用 Python 3.7

1. 登入 AWS Management Console 並在 https：//https://console.aws.amazon.com/lambda/ 開啟 AWS Lambda 主控台。
2. 在函數清單中，篩選 **`SecretsManager`**。
3. 在篩選出的函數清單中，在執行期之下，尋找 Python 3.7。

若要升級至 Python 3.9：

- 方式 1：使用 AWS CloudFormation重新建立輪換函數
- 選項 2：使用 更新現有輪換函數的執行時間 AWS CloudFormation
- 選項 3：針對 AWS CDK 使用者，升級 CDK 程式庫

方式 1：使用 AWS CloudFormation重新建立輪換函數

當您使用 Secrets Manager 主控台開啟輪換時，Secrets Manager 會使用 AWS CloudFormation 來建立必要的資源，包括 Lambda 輪換函數。如果您使用主控台開啟輪換，或是使用 AWS CloudFormation 堆疊建立輪換函數，您可以使用相同的 AWS CloudFormation 堆疊以新名稱重新建立輪換函數。新函數使用了較新版本的 Python。

尋找建立輪換函數的 AWS CloudFormation 堆疊

- 在 Lambda 函數的詳細資訊頁面的組態分頁上，選擇標籤。檢視 aws:cloudformation:stack-id 旁邊的 ARN。

  堆疊名稱嵌入在 ARN 中，如以下範例所示。

  - ARN：`arn:aws:cloudformation:us-west-2:408736277230:stack/`**`SecretsManagerRDSMySQLRotationSingleUser5c2-SecretRotationScheduleHostedRotationLambda`**`-3CUDHZMDMBO8/79fc9050-2eef-11ed-`
  - 堆疊名稱：**`SecretsManagerRDSMySQLRotationSingleUser5c2-SecretRotationScheduleHostedRotationLambda`**

若要重新建立輪換函數 (AWS CloudFormation)

1. 在 中 AWS CloudFormation，依名稱搜尋堆疊，然後選擇更新。

   如果出現建議您更新根堆疊的對話方塊，請選擇前往根堆疊，然後選擇更新。
2. 在更新堆疊頁面的準備範本下，選擇應用程式編寫器中的編輯，然後在應用程式編寫器中的編輯範本下，選擇應用程式編寫器中的編輯按鈕。
3. 在應用程式編寫器中，執行下列動作：

   a. 在範本程式碼中，在 中`SecretRotationScheduleHostedRotationLambda`，將 的值取代`"functionName": "SecretsManagerTestRotationRDS"`為新的函數名稱，例如在 JSON 中，**`"functionName": "SecretsManagerTestRotationRDSupdated"`**

   b. 選擇更新範本。

   c. 在繼續 AWS CloudFormation對話方塊中，選擇確認並繼續 AWS CloudFormation。
4. 繼續執行 AWS CloudFormation 堆疊工作流程，然後選擇提交。

選項 2：使用 更新現有輪換函數的執行時間 AWS CloudFormation

當您使用 Secrets Manager 主控台開啟輪換時，Secrets Manager 會使用 AWS CloudFormation 來建立必要的資源，包括 Lambda 輪換函數。如果您使用主控台來開啟輪換，或是使用 AWS CloudFormation 堆疊建立輪換函數，您可以使用相同的 AWS CloudFormation 堆疊來更新輪換函數的執行時間。

尋找建立輪換函數的 AWS CloudFormation 堆疊

- 在 Lambda 函數的詳細資訊頁面的組態分頁上，選擇標籤。檢視 aws:cloudformation:stack-id 旁邊的 ARN。

  堆疊名稱嵌入在 ARN 中，如以下範例所示。

  - ARN：`arn:aws:cloudformation:us-west-2:408736277230:stack/`**`SecretsManagerRDSMySQLRotationSingleUser5c2-SecretRotationScheduleHostedRotationLambda`**`-3CUDHZMDMBO8/79fc9050-2eef-11ed-`
  - 堆疊名稱：**`SecretsManagerRDSMySQLRotationSingleUser5c2-SecretRotationScheduleHostedRotationLambda`**

若要更新輪換函數的執行期 (AWS CloudFormation)

1. 在 中 AWS CloudFormation，依名稱搜尋堆疊，然後選擇更新。

   如果出現建議您更新根堆疊的對話方塊，請選擇前往根堆疊，然後選擇更新。

2. 在更新堆疊頁面的準備範本下，選擇應用程式編寫器中的編輯，然後在應用程式編寫器中的編輯範本下，選擇應用程式編寫器中的編輯按鈕。
3. 在應用程式編寫器中，執行下列動作：

   a. 在範本 JSON 中，在 `SecretRotationScheduleHostedRotationLambda`下`Properties`，在下`Parameters`，新增 **`"runtime": "python3.9"`**。

   b. 選擇更新範本。

   c. 在繼續 AWS CloudFormation對話方塊中，選擇確認並繼續 AWS CloudFormation。

4. 繼續進行 AWS CloudFormation 堆疊工作流程，然後選擇提交。

選項 3：針對 AWS CDK 使用者，升級 CDK 程式庫

如果您使用 v2.94.0 AWS CDK 之前的 來設定秘密的輪換，您可以透過升級至 v2.94.0 或更新版本來更新 Lambda 函數。如需詳細資訊，請參閱《AWS Cloud Development Kit (AWS CDK) v2 開發人員指南》。

AWS Lambda 秘密輪換**PutSecretValue**失敗

如果您將擔任的角色或跨帳戶輪換與 Secrets Manager 搭配使用，並在 CloudTrail 中發現具有訊息RotationFailed的事件：Lambda LAMBDA_*ARN ######### VERSION_ID* for Secret *SECRET_ARN*。 移除AWSPENDING預備標籤並重新啟動輪換，然後您需要更新 Lambda 函數才能使用 RotationToken 參數。

更新 Lambda 輪換函數以包含 **RotationToken**

1. 下載 Lambda 函數程式碼
   - 開啟 Lambda 主控台
   - 在導覽窗格中，選擇函數
   - 選取函數名稱的 Lambda 秘密輪換函數
   - 針對下載，請選擇其中一個函數程式碼 .zip、AWS SAM 檔案、兩者
   - 選擇確定，將函數儲存在本機電腦上。
2. 編輯 Lambda_handler

   在 create_secret 步驟中包含 rotation_token 參數，以進行跨帳戶輪換：

```
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. 編輯create_secret程式碼

   修改create_secret函數以接受並使用 rotation_token 參數：

```
# 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. 上傳更新的 Lambda 函數程式碼

   更新 Lambda 函數程式碼後，請將其上傳以輪換秘密。

錯誤：「在 <a rotation> ###### lambda <arn> 時發生錯誤」

如果您在 Lambda 函數卡在集合迴圈中時遇到間歇性秘密輪換失敗，例如介於 CreateSecret和 之間SetSecret，問題可能與並行設定有關。

並行疑難排解步驟

> **Warning**
>
> 由於 Lambda 函數的執行執行緒不足，將佈建並行參數設定為小於 10 的值可能會導致限流。如需詳細資訊，請參閱《 AWS Lambda 開發人員指南》中的 AWS Lambda 了解預留並行和佈建並行。

1. 檢查和調整 Lambda 並行設定：
   - 確認 `reserved_concurrent_executions` 未設定太低（例如 1)
   - 如果使用預留並行，請將其設定為至少 10
   - 考慮使用未預留並行以提高靈活性
2. 對於佈建並行：
   - 請勿明確設定佈建並行參數（例如，在 Terraform 中）。
   - 如果您必須設定，請使用至少 10 的值。
   - 徹底測試，以確保所選的值適用於您的使用案例。
3. 監控和調整並行：
   - 使用此公式計算並行：並行 = （每秒平均請求數）*（以秒為單位的平均請求持續時間）。如需詳細資訊，請參閱預估預留並行。
   - 在輪換期間觀察並記錄值，以判斷適當的並行設定。
   - 設定低並行值時請小心。如果沒有足夠的可用執行緒，可能會導致限流。

如需設定 Lambda 並行的詳細資訊，請參閱《 AWS Lambda 開發人員指南》中的設定預留並行和設定佈建並行。

輪換排程

Secrets Manager 會在您設定的輪換時段內，依排程輪換您的秘密。若要設定排程和視窗，請使用 cron() 或 rate() 運算式以及視窗持續時間。Secrets Manager 在輪換時段的任何時間輪換您的機密。您可以在小至一小時的輪換時段內，每隔四個小時就可以輪換一次秘密。

若要開啟輪換，請參閱：

- the section called “受管輪換”
- the section called “資料庫秘密的自動輪換 (主控台)”
- the section called “非資料庫秘密的自動輪換（主控台）”

Secrets Manager 輪換排程使用 UTC 時區。

輪換時段

Secrets Manager 輪換時段類似於維護時段。當您想要輪換秘密時，您可以設定輪換時段，Secrets Manager 會在輪換時段的某個時間輪換秘密。

Secrets Manager 輪換時段一律從小時開始。對於使用天數`rate()`表達式的輪換排程，輪換時段會從午夜開始。您可以使用`cron()`表達式來設定輪換時段的開始時間。如需範例，請參閱 the section called “Cron 表達式”。

根據預設，輪換時段會在一小時後關閉，輪換排程則會在一天結束時關閉，以天為單位。

若要變更輪換時段的長度，請設定時段持續時間。您可以將輪換時段設定為小至一小時。輪換時段不得延伸到下一個輪換時段。換句話說，對於以小時為單位的輪換排程，請確認輪換時段小於或等於輪換之間的時數。對於以天為單位的輪換排程，請確認開始時間加上時段持續時間小於或等於 24 小時。

Rate 運算式

Secrets Manager rate 表達式的格式如下，*`Value`* 是正整數，*`Unit`* 可以是 `hour`、`hours`、`day` 或 `days`：

```
rate(Value Unit)
```

您可以每四小時輪換機密一次。最長輪換期間為 999 天。範例：

- `rate(4 hours)` 表示每四小時輪換機密一次。
- `rate(1 day)` 表示每天輪換機密一次。
- `rate(10 days)` 表示每 10 天輪換機密一次。

Cron 表達式

Secrets Manager cron 表達式的格式如下：

```
cron(Minutes Hours Day-of-month Month Day-of-week Year)
```

包含小時增量的 cron 表達式會每天重設。例如，`cron(0 4/12 * * ? *)` 表示凌晨 4:00 和下午 4:00，然後是第二天凌晨 4:00 和下午 4:00。Secrets Manager 輪換排程使用 UTC 時區。

範例排程	表達式
每隔八小時在午夜開始。	`cron(0 /8 * * ? *)`
每隔八小時在上午 8:00 開始。	`cron(0 8/8 * * ? *)`
每隔十小時在凌晨 2:00 開始。 輪換時段將在 2:00、12:00 和 22:00 開始，然後在第二天 2:00、12:00 和 22:00 開始。	`cron(0 2/10 * * ? *)`
每天上午 10:00。	`cron(0 10 * * ? *)`
每週六下午 6:00。	`cron(0 18 ? * SAT *)`
每個月第一天上午 8:00。	`cron(0 8 1 * ? *)`
每三個月第一個星期天的凌晨 1:00。	`cron(0 1 ? 1/3 SUN#1 *)`
每個月最後一天下午 5:00。	`cron(0 17 L * ? *)`
每週一至週五上午 8:00。	`cron(0 8 ? * MON-FRI *)`
每月的第一天和第十五天下午 4:00。	`cron(0 16 1,15 * ? *)`
每個月的第一個週日午夜。	`cron(0 0 ? * SUN#1 *)`
從 1 月開始，第一個星期一午夜每 11 個月一次。	`cron(0 0 ? 1/11 2#1 *)`

Secrets Manager 中的 cron 表達式要求

Secrets Manager 對您可以用於 cron 表達式的內容有一些限制。Secrets Manager 的 cron 表達式在分鐘欄位中必須有 0，因為 Secrets Manager 輪換時段會在整點時刻開始。在年份欄位中必須有 *，因為 Secrets Manager 不支援相隔一年以上的輪換排程。下列資料表顯示您可以使用的選項。

欄位	Values (數值)	萬用字元
分鐘	必須為 0	無

欄位	Values (數值)	萬用字元
小時	0–23	使用 / (正斜線) 指定增量。例如，2/10 表示從凌晨 2:00 開始每隔 10 小時一次。您可以每四小時輪換一次秘密。
月中的日	1–31	使用 , (逗號) 來包含其他值。例如，1,15 表示一個月的第 1 天和第 15 天。 使用 - (破折號) 指定範圍。例如，1–15 表示一個月的第 1 天至第 15 天。 使用 * (星號) 來包含欄位中的所有值。例如，* 表示一個月的每一天。 ? (問號) 萬用字元用於表示不限定任何一個。您無法在同一個 cron 表達式中指定 Day-of-month 和 Day-of-week 欄位。如果您在其中一個欄位指定了數值，就必須在另一個欄位中使用 ? (問號)。 使用 / (正斜線) 指定增量。例如，1/2 表示每兩天從第 1 天開始，換句話說，第 1 天、第 3 天、5 天等。 使用 L 指定一個月的最後一天。 使用 ***DAY***L 指定一個月的最後一個命名日期。例如，SUNL 表示一個月的最後一個週日。

欄位	Values (數值)	萬用字元
月	1-12 或 JAN-DEC	使用 , (逗號) 來包含其他值。例如，JAN,APR,JUL,OCT 表示 1 月、4 月、7 月 和 10 月。 使用 - (破折號) 指定範圍。例如，1–3 表示一年的第 1 個月至第 3 個月。 使用 * (星號) 來包含欄位中的所有值。例如，* 表示每個月。 使用 / (正斜線) 指定增量。例如，1/3 表示每第三個月，從第 1 個月開始，換句話說第 1 個月、第 4 個月、第 7 個月和第 10 個月。

欄位	Values (數值)	萬用字元
週中的日	1-7 或 SUN-SAT	使用 # 指定一個月內的星期幾。例如：TUE#3 表示當月的第三個週二。 使用 , (逗號) 來包含其他值。例如，1,4 表示一週的第 1 天和第 4 天。 使用 - (破折號) 指定範圍。例如，1–4 表示一週的第 1 天至第 4 天。 使用 * (星號) 來包含欄位中的所有值。例如，* 表示一週的每一天。 ? (問號) 萬用字元用於表示不限定任何一個。您無法在同一個 cron 表達式中指定 Day-of-month 和 Day-of-week 欄位。如果您在其中一個欄位指定了數值，就必須在另一個欄位中使用 ? (問號)。 使用 / (正斜線) 指定增量。例如，1/2 表示一週的每隔一天，從第一天開始，則第 1 天、第 3 天、第 5 天和第 7 天。 使用 L 指定一週的最後一天。
年	必須為 *	無

立即輪換 AWS Secrets Manager 秘密

您只能輪換已設定輪換的秘密。若要確定是否已設定秘密進行輪換，請在主控台中檢視該秘密，然後向下捲動至 Rotation configuration (輪換組態) 部分。如果 Rotation status (輪換狀態) 為 Enabled (已啟用)，則會設定秘密進行輪換。如果沒有，請參閱 輪換 秘密。

若要立即輪換秘密 (主控台)

1. 開啟位於的 Secrets Manager 主控台 https://console.aws.amazon.com/secretsmanager/。
2. 選擇您的秘密。
3. 在秘密詳細資訊頁面的 Rotation configuration (輪換組態) 中，選擇 Rotate secret immediately (立即輪換秘密)。
4. 在 Rotate secret (輪換秘密) 對話方塊中，選擇 Rotate (輪換)。

AWS CLI

Example 立即輪換秘密

下列 rotate-secret 範例會立即開始輪換。機密必須已設定輪換。

```
$ aws secretsmanager rotate-secret \
    --secret-id MyTestSecret
```

尋找未輪換的秘密

您可以使用 AWS Config 來評估秘密，以查看它們是否依照您的標準輪換。您可以使用 AWS Config 規則來定義秘密的內部安全和合規要求。然後， AWS Config 可以識別不符合您規則的秘密。您也可以追蹤秘密中繼資料、輪換組態、用於秘密加密的 KMS 金鑰、Lambda 輪換功能以及與秘密相關聯之標籤的變更。

如果您的組織中有多個 AWS 帳戶 和 AWS 區域 的秘密，您可以彙總該組態和合規資料。如需詳細資訊，請參閱多帳戶多區域資料彙總。

評估秘密是否輪換

1. 遵循使用 AWS Config 規則評估資源的指示，並從下列規則中選擇：

- secretsmanager-rotation-enabled-check – 檢查是否為 Secrets Manager 中存放的秘密設定了輪換。
- secretsmanager-scheduled-rotation-success-check – 檢查上次成功輪換是否在設定的輪換頻率範圍內。檢查的最低頻率為每日。
- secretsmanager-secret-periodic-rotation – 檢查是否在指定的天數內輪換秘密。

2. 或者，設定 AWS Config 以在秘密不合規時通知您。如需詳細資訊，請參閱AWS Config 傳送至 Amazon SNS 主題的通知。

在 Secrets Manager 中取消自動輪換

如果您已設定秘密的自動輪換，且想要停止輪換，您可以取消輪換。

取消自動輪換

1. 開啟位於的 Secrets Manager 主控台 https://console.aws.amazon.com/secretsmanager/。
2. 選擇您的秘密。
3. 在秘密詳細資訊頁面的輪換組態下，選擇編輯輪換。
4. 在編輯輪換組態對話方塊中，關閉自動輪換，然後選擇儲存。

   Secrets Manager 會保留輪換組態資訊，以便您日後決定重新開啟輪換時可以使用它。

AWS Secrets Manager 由其他服務管理的秘密 AWS

許多 AWS 服務存放和使用秘密 AWS Secrets Manager。在某些情況下，這些秘密是受管秘密，也就是說，建立秘密的服務有助於管理秘密。例如，有些受管秘密會包含受管輪換，因此您不必自行設定輪換。管理服務也可能會限制您在沒有復原期的情況下更新或刪除秘密，因為管理服務取決於秘密，因此這樣有助於避免外洩情況發生。

> **Note**
>
> 受管秘密只能由管理秘密 AWS 的服務建立。

受管秘密的命名慣例是在名稱中加入管理服務 ID ，以便識別。

```
Secret name: ServiceID!MySecret
Secret ARN : arn:aws:us-east-1:ServiceID!MySecret-a1b2c3
```

管理秘密的服務之 ID

- `appflow` – the section called “Amazon AppFlow”
- `databrew` – the section called “AWS Glue DataBrew”
- `datasync` – the section called “AWS DataSync”
- `directconnect` – the section called “AWS Direct Connect”
- `ecs-sc` – the section called “Amazon Elastic Container Service”
- `events` – the section called “Amazon EventBridge”
- `marketplace-deployment` – the section called “AWS Marketplace”
- `opsworks-cm` – the section called “AWS OpsWorks for Chef Automate”
- `pcs` – the section called “AWS 平行運算服務”
- `rds` – the section called “Amazon RDS”
- `redshift` – the section called “Amazon Redshift”
- `sqlworkbench` – the section called “Amazon Redshift 查詢編輯器第 2 版”

若要尋找由其他服務管理的秘密 AWS ，請參閱尋找受管秘密。

如需使用秘密的服務完整清單，請參閱使用秘密的服務。

AWS 服務 使用 AWS Secrets Manager 秘密

取得下列 AWS 服務 各項如何與 Secrets Manager 整合的相關資訊。

- AWS App Runner 如何使用 AWS Secrets Manager
- How AWS App2Container 使用 AWS Secrets Manager
- AWS AppConfig 如何使用 AWS Secrets Manager
- Amazon AppFlow 如何使用 AWS Secrets Manager
- AWS AppSync 如何使用 AWS Secrets Manager
- Amazon Athena 如何使用 AWS Secrets Manager
- Amazon Aurora 如何使用 AWS Secrets Manager
- AWS CodeBuild 如何使用 AWS Secrets Manager
- Amazon Data Firehose 如何使用 AWS Secrets Manager
- AWS DataSync 如何使用 AWS Secrets Manager
- Amazon DataZone 如何使用 AWS Secrets Manager
- AWS Direct Connect 如何使用 AWS Secrets Manager
- AWS Directory Service 如何使用 AWS Secrets Manager
- Amazon DocumentDB (with MongoDB compatibility) 如何使用 AWS Secrets Manager
- AWS Elastic Beanstalk 如何使用 AWS Secrets Manager
- Amazon Elastic Container Registry 如何使用 AWS Secrets Manager
- Amazon Elastic Container Service
- Amazon ElastiCache 如何使用 AWS Secrets Manager
- AWS Elemental Live 如何使用 AWS Secrets Manager
- AWS Elemental MediaConnect 如何使用 AWS Secrets Manager
- AWS Elemental MediaConvert 如何使用 AWS Secrets Manager
- AWS Elemental MediaLive 如何使用 AWS Secrets Manager
- AWS Elemental MediaPackage 如何使用 AWS Secrets Manager
- AWS Elemental MediaTailor 如何使用 AWS Secrets Manager
- Amazon EMR 如何使用 Secrets Manager
- Amazon EventBridge 如何使用 AWS Secrets Manager

- Amazon FSx 如何使用 AWS Secrets Manager 秘密
- AWS Glue DataBrew 如何使用 AWS Secrets Manager
- AWS Glue Studio 如何使用 AWS Secrets Manager
- AWS IoT SiteWise 如何使用 AWS Secrets Manager
- Amazon Kendra 如何使用 AWS Secrets Manager
- Amazon Kinesis Video Streams 如何使用 AWS Secrets Manager
- AWS Launch Wizard 如何使用 AWS Secrets Manager
- Amazon Lookout for Metrics 如何使用 AWS Secrets Manager
- Amazon Managed Grafana 如何使用 AWS Secrets Manager
- AWS Managed Services 如何使用 AWS Secrets Manager
- Amazon Managed Streaming for Apache Kafka 如何使用 AWS Secrets Manager
- Amazon Managed Workflows for Apache Airflow 如何使用 AWS Secrets Manager
- AWS Marketplace
- AWS Migration Hub 如何使用 AWS Secrets Manager
- AWS Panorama 如何使用 Secrets Manager
- AWS Parallel Computing Service 如何使用 AWS Secrets Manager
- AWS ParallelCluster 如何使用 AWS Secrets Manager
- Amazon Q 如何使用 Secrets Manager
- Amazon OpenSearch Ingestion如何使用 Secrets Manager
- AWS OpsWorks for Chef Automate 如何使用 AWS Secrets Manager
- Amazon QuickSight 如何使用 AWS Secrets Manager
- Amazon RDS 如何使用 AWS Secrets Manager
- Amazon Redshift 如何使用 AWS Secrets Manager
- Amazon Redshift 查詢編輯器第 2 版
- Amazon SageMaker AI 如何使用 AWS Secrets Manager
- AWS Schema Conversion Tool 如何使用 AWS Secrets Manager
- InfluxDB 的 Amazon Timestream 使用方式 AWS Secrets Manager
- AWS Toolkit for JetBrains 如何使用 AWS Secrets Manager
- AWS Transfer Family 如何使用 AWS Secrets Manager 秘密

- AWS Wickr如何使用 AWS Secrets Manager 秘密

AWS App Runner 如何使用 AWS Secrets Manager

AWS App Runner 是一種 AWS 服務，可提供快速、簡單且符合成本效益的方式，從原始碼或容器映像直接部署到 AWS 雲端中可擴展且安全的 Web 應用程式。您不需要學習新技術、決定要使用的運算服務，或知道如何佈建和設定 AWS 資源。

使用 App Runner，您可以在建立服務或更新服務的組態時，參考機密和組態作為服務中的環境變數。如需詳細資訊，請參閱《AWS App Runner 開發人員指南》中的參考環境變數和管理環境變數。

How AWS App2Container 使用 AWS Secrets Manager

AWS App2Container 是一種命令列工具，可協助您提升和轉移在內部部署資料中心或虛擬機器中執行的應用程式，使其在由 Amazon ECS、Amazon EKS 或 管理的容器中執行 AWS App Runner。

App2Container 使用機密管理員來管理用於將工作者機器連線到應用程式伺服器的憑證，以便執行遠端命令。如需詳細資訊，請參閱《AWS App2Container 使用者指南》中的管理 App2Container 的秘密。AWS App2Container

AWS AppConfig 如何使用 AWS Secrets Manager

AWS AppConfig 是 的一項功能 AWS Systems Manager ，可讓您用來建立、管理和快速部署應用程式組態。組態可以包含憑證資料或其他存放在 Secrets Manager 中的敏感資訊。建立自由格式組態設定檔時，您可以選擇 Secrets Manager 作為組態資料的來源。如需詳細資訊，請參閱《AWS AppConfig 使用者指南》中的 Creating a configuration and a configuration profile (建立組態和組態設定檔)。如需有關 AWS AppConfig 如何處理已開啟自動輪換的秘密的資訊，請參閱AWS AppConfig 《 使用者指南》中的 Secrets Manager 金鑰輪換。

Amazon AppFlow 如何使用 AWS Secrets Manager

Amazon AppFlow 是一種全受管整合服務，可讓您在軟體即服務 (SaaS) 應用程式之間安全地交換資料，例如 Salesforce 和 AWS 服務，例如 Amazon Simple Storage Service (Amazon S3) 和 Amazon Redshift。

在 Amazon AppFlow 中，當您將 SaaS 應用程式設定為來源或目的地時，您將建立連線。這包括連線至 SaaS 應用程式所需的資訊，如身分驗證權杖、使用者名稱和密碼。Amazon AppFlow 會將您的連線資料存放在字首為 的 Secrets Manager 受管秘密中`appflow`。存放秘密的成本包含在 Amazon

AppFlow 的費用中。如需詳細資訊，請參閱《Amazon AppFlow 使用者指南》中的 Amazon AppFlow 中的資料保護。

AWS AppSync 如何使用 AWS Secrets Manager

AWS AppSync 為應用程式開發人員提供強大且可擴展的 GraphQL 界面，以結合來自多個來源的資料，包括 Amazon DynamoDB AWS Lambda和 HTTP APIs。

AWS AppSync 使用 Secrets Manager 秘密中的登入資料來連線至 Amazon RDS 和 Aurora。如需詳細資訊，請參閱《AWS AppSync 開發人員指南》中的教學課程：Aurora Serverless。

Amazon Athena 如何使用 AWS Secrets Manager

Amazon Athena 是一種互動式查詢服務，可在 Amazon Simple Storage Service (Amazon S3) 中使用標準 SQL 輕鬆地直接分析資料。

Amazon Athena 資料來源連線器可以使用 Athena 聯合查詢功能和機密管理員機密來查詢資料。如需詳細資訊，請參閱《Amazon Athena 使用者指南》中的使用 Amazon Athena 聯合查詢。

Amazon Aurora 如何使用 AWS Secrets Manager

Amazon Aurora 為全受管關聯式資料庫引擎，可與 MySQL 和 PostgreSQL 相容。

若要管理 Aurora 的主要使用者登入資料，Aurora 可以為您建立受管秘密。您需要為該秘密付費。Aurora 也會管理這些登入資料的輪換。如需詳細資訊，請參閱《Amazon Aurora 使用者指南》中的使用 Amazon Aurora 和 AWS Secrets Manager進行密碼管理。

如需其他 Aurora 登入資料，請參閱 建立秘密。

當您呼叫 Amazon RDS 資料 API 時，您可以在機密管理員中使用機密來傳遞資料庫的憑證。如需詳細資訊，請參閱《Amazon Aurora 使用者指南》中的使用適用於 Aurora Serverless 的資料 API。

當您使用 Amazon RDS 查詢編輯器連線到資料庫時，可將資料庫的憑證存放在 Secrets Manager 中。如需詳細資訊，請參閱《Amazon RDS 使用者指南》中的使用查詢編輯器。

AWS CodeBuild 如何使用 AWS Secrets Manager

AWS CodeBuild 是雲端中全受管的建置服務。CodeBuild 可編譯原始碼、執行單元測試，並產生可立即部署的成品。

您可以使用機密管理員來存放私密登錄登入資料。如需詳細資訊，請參閱AWS CodeBuild 《 使用者指南》中的具有 CodeBuild AWS Secrets Manager 範例的私有登錄檔。

Amazon Data Firehose 如何使用 AWS Secrets Manager

您可以使用 Amazon Data Firehose 將即時串流資料交付至各種串流目的地。當目的地需要登入資料或金鑰時，Firehose 會在執行時間從 Secrets Manager 擷取秘密，以連線至目的地。如需詳細資訊，請參閱《Amazon Data Firehose 開發人員指南 AWS Secrets Manager 》中的在 Amazon Data Firehose中使用 驗證 。

AWS DataSync 如何使用 AWS Secrets Manager

AWS DataSync 是一種線上資料傳輸服務，可簡化、自動化和加速儲存系統與服務之間的資料移動。

DataSync 支援的某些儲存系統需要登入資料才能讀取和寫入資料。DataSync 使用 Secrets Manager 來存放或存取儲存憑證。您可以設定 DataSync 代表您建立秘密，也可以提供自訂秘密。服務管理的秘密開頭為字首 `aws-datasync`。您只需針對使用在 DataSync 外部建立的秘密付費。請參閱AWS DataSync 《 使用者指南》中的提供儲存位置的登入資料。

Amazon DataZone 如何使用 AWS Secrets Manager

Amazon DataZone 是一種資料管理服務，可讓您建立目錄、探索、管理、共享及分析資料。您可以從使用 AWS Glue 編目程式 任務爬取的 Amazon Redshift 叢集的資料表和檢視使用資料資產。如要連線至 Amazon Redshift，您必須以 Secrets Manager 秘密的形式提供 Amazon DataZone 憑證。如需詳細資訊，請參閱《Amazon DataZone 使用者指南》中的使用新 AWS Glue 連線建立 Amazon Redshift 資料庫的資料來源。 DataZone

AWS Direct Connect 如何使用 AWS Secrets Manager

AWS Direct Connect 會透過標準乙太網路光纖纜線將您的內部網路連結至 AWS Direct Connect 位置。透過此連線，您可以直接建立與公有 的虛擬介面 AWS 服務。

AWS Direct Connect 會將連線關聯金鑰名稱和連線關聯金鑰對 (CKN/CAK 對） 存放在字首為 的受管秘密中`directconnect`。秘密的成本包含在 AWS Direct Connect的費用中。若要更新秘密，您必須使用 AWS Direct Connect 而非 Secrets Manager。如需詳細資訊，請參閱《AWS Direct Connect 使用者指南》中的將 MACsec CKN/CAK 與 LAG 產生關聯。

AWS Directory Service 如何使用 AWS Secrets Manager

AWS Directory Service 提供多種將 Microsoft Active Directory (AD) 與其他 AWS 服務搭配使用的方式。您可以使用憑證的機密，將 Amazon EC2 執行個體加入目錄。如需詳細資訊，在《AWS Direct Connect 使用者指南》中，請參閱：

- 將 Linux EC2 執行個體無縫加入您的 AWS Managed Microsoft AD 目錄
- 將 Linux EC2 執行個體無縫加入您的 AD Connector 目錄
- 將 Linux EC2 執行個體無縫加入您的 Simple AD 目錄

Amazon DocumentDB (with MongoDB compatibility) 如何使用 AWS Secrets Manager

Amazon DocumentDB （與 MongoDB 相容） 是支援 MongoDB 工作負載的全受管文件資料庫服務。Amazon DocumentDB 與 Secrets Manager 整合，以管理叢集的主要使用者密碼，增強安全性並簡化憑證管理。

Amazon DocumentDB 會產生密碼、將它存放在 Secrets Manager 中，並管理秘密設定。根據預設，Amazon DocumentDB 每七天輪換一次秘密，但您可以視需要修改輪換排程。當您建立或修改 Amazon DocumentDB 叢集時，您可以指定它應該在 Secrets Manager 中管理主要使用者密碼。如需詳細資訊，請參閱《Amazon DocumentDB 開發人員指南》中的使用 Amazon DocumentDB 和 Secrets Manager 進行密碼管理。 Amazon DocumentDB

AWS Elastic Beanstalk 如何使用 AWS Secrets Manager

透過 AWS Elastic Beanstalk，您可以快速部署和管理 AWS 雲端中的應用程式，而無需了解執行這些應用程式的基礎設施。Elastic Beanstalk 能夠建置 Dockerfile 所描述的映像或提取遠端 Docker 映像檔，藉此啟動 Docker 環境。為使用託管私有儲存庫的線上登錄來驗證身分，Elastic Beanstalk 會使用 Secrets Manager 秘密。如需詳細資訊，請參閱《AWS Elastic Beanstalk 開發人員指南》中的Docker 組態。

Amazon Elastic Container Registry 如何使用 AWS Secrets Manager

Amazon Elastic Container Registry (Amazon ECR) 是一種安全、可擴展且可靠的 AWS 受管容器映像登錄服務。您可以使用 Docker CLI 或您偏好的用戶端來向您的儲存庫推送和提取映像。針對每個包含

要在 Amazon ECR 私有登錄檔中快取的映像的上游登錄檔，您必須建立一個提取快取規則。針對需要身分驗證的上游登錄檔，您必須將憑證儲存在 Secrets Manager 秘密中。您可以在 Amazon ECR 或 Secrets Manager 主控台中建立 Secrets Manager 秘密。如需詳細資訊，請參閱《Amazon ECR 使用者指南》中的建立提取快取規則。

Amazon Elastic Container Service

Amazon Elastic Container Service (Amazon ECS) 為全受管容器協同運作服務，可讓您輕鬆部署、管理和擴展容器化應用程式。您可以參考 Secrets Manager 秘密，將敏感資料插入容器中。如需詳細資訊，請參閱《Amazon Elastic Container Service 開發人員指南》中的下列頁面：

- 教學課程：使用 Secrets Manager 秘密指定敏感資料
- 透過應用程式以程式設計方式擷取秘密
- 透過環境變數擷取秘密
- 擷取記錄組態的秘密

Amazon ECS 支援 FSx for Windows File Server 容器磁碟區。Amazon ECS 使用儲存在 Secrets Manager 秘密中的憑證，用於網域加入 Active Directory 並連接 FSx for Windows File Server 檔案系統。如需詳細資訊，請參閱《Amazon Elastic Container Service 開發人員指南》中的教學課程：將 FSx for Windows File Server 檔案系統搭配 Amazon ECS 使用，以及 FSx for Windows File Server 磁碟區。

您可以在需要身分驗證的私有登錄檔中參考容器映像 AWS ，方法是使用 Secrets Manager 秘密搭配登錄檔憑證。如需詳細資訊，請參閱《Amazon Elastic Container Service 開發人員指南》中的任務的私有登錄檔身分驗證。

當您使用 Amazon ECS Service Connect 時，Amazon ECS 會使用 Secrets Manager 受管秘密來存放 AWS Private Certificate Authority TLS 憑證。存放秘密的成本會包含在 Amazon ECS 的費用中。若要更新秘密，您必須使用 Amazon ECS 而非 Secrets Manager。如需詳細資訊，請參閱《Amazon Elastic Container Service 開發人員指南》中的 TLS with Service Connect。

Amazon ElastiCache 如何使用 AWS Secrets Manager

在 ElastiCache 中，您可以使用稱為角色型存取控制 (RBAC) 的功能保護叢集安全。您可以將這些憑證儲存在 Secrets Manager 中。Secrets Manager 為此類型的機密提供了輪換範本。如需詳細資訊，請參閱《Amazon ElastiCache 使用者指南》中的自動輪換使用者的密碼。

AWS Elemental Live 如何使用 AWS Secrets Manager

AWS Elemental Live 是一種即時影片服務，可讓您建立廣播和串流交付的即時輸出。

AWS Elemental Live 使用秘密 ARN 從 Secrets Manager 取得包含加密金鑰的秘密。Elemental Live 會使用加密金鑰加密/解密視訊。如需詳細資訊，請參閱 Elemental Live 使用者指南中的從 交付 AWS Elemental Live 到 MediaConnect 在執行時間的運作方式。

AWS Elemental MediaConnect 如何使用 AWS Secrets Manager

AWS Elemental MediaConnect 是一項服務，可讓廣播者和其他高級影片供應商輕鬆地將即時影片可靠地擷取到 ， AWS 雲端 並將其分發到 內部或外部的多個目的地 AWS 雲端。

您可以使用靜態金鑰加密來保護來源、輸出和權利，並將加密金鑰存放在 AWS Secrets Manager。如需詳細資訊，請參閱AWS Elemental MediaConnect 《 使用者指南》中的 中的靜態金鑰加密 AWS Elemental MediaConnect。

AWS Elemental MediaConvert 如何使用 AWS Secrets Manager

AWS Elemental MediaConvert 是一種以檔案為基礎的影片處理服務，可為具有任何大小媒體庫的內容擁有者和經銷商提供可擴展的影片處理。若要使用 MediaConvert 對 Kantar 浮水印進行編碼，請使用機密管理員來存放您的 Kantar 憑證。如需詳細資訊，請參閱AWS Elemental MediaConvert 《 使用者指南》中的在 AWS Elemental MediaConvert 輸出中使用 Kantar 進行音訊浮水印。

AWS Elemental MediaLive 如何使用 AWS Secrets Manager

AWS Elemental MediaLive 是一種即時影片服務，可讓您建立廣播和串流交付的即時輸出。如果您的組織使用具有 AWS Elemental MediaLive 或 AWS Elemental Link 的裝置 AWS Elemental MediaConnect，您必須部署裝置並設定裝置。如需詳細資訊，請參閱 MediaLive 使用者指南中的將 MediaLive 設定為信任的實體。 MediaLive

AWS Elemental MediaPackage 如何使用 AWS Secrets Manager

AWS Elemental MediaPackage 是在 just-in-time影片封裝和起始服務 AWS 雲端。有了 MediaPackage，您可以將高度安全、可擴展且可靠的影片串流傳送到各種播放裝置和內容交付網路 (CDN)。如需詳細資訊，請參閱AWS Elemental MediaPackage 《 使用者指南》中的適用於 CDN 授權的 Secrets Manager 存取權。

AWS Elemental MediaTailor 如何使用 AWS Secrets Manager

AWS Elemental MediaTailor 是一種可擴展的廣告插入和頻道組合服務，可在 中執行 AWS 雲端。

MediaTailor 支援對您的來源位置進行機密管理員存取權杖身分驗證。透過機密管理員存取權杖身分驗證，MediaTailor 會使用機密管理員機密來對您原始伺服器的請求進行身分驗證。如需詳細資訊，請參閱AWS Elemental MediaTailor 《 使用者指南》中的設定 AWS Secrets Manager 存取權杖身分驗證。

Amazon EMR 如何使用 Secrets Manager

Amazon EMR 是一種平台，可簡化 Apache Hadoop 和 Apache Spark 等大數據架構的執行， AWS 以處理和分析大量資料。使用這些架構和相關的開放原始碼專案 (例如 Apache Hive 和 Apache Pig) 時，您可以處理資料以便用於分析和商業情資工作負載。您也可以使用 Amazon EMR 轉換大量資料，並將其移入和移出其他 AWS 資料存放區和資料庫，例如 Amazon S3 和 Amazon DynamoDB。

在 Amazon EC2 執行的 Amazon EMR 如何使用 Secrets Manager

您在 Amazon EMR 中建立叢集時，可以利用 Secrets Manager 中的秘密，將應用程式組態資料提供給叢集。如需詳細資訊，請參閱《Amazon EMR 管理指南》中的將敏感組態資料存放在 Secrets Manager。

此外，當您建立 EMR Notebook 時，可以使用 Secrets Manager 來存放私有 Git 型登錄憑證。如需詳細資訊，請參閱《Amazon EMR 管理指南》中的將 Git 型儲存庫新增至 Amazon EMR。

EMR Serverless 如何使用 Secrets Manager

EMR Serverless 提供無伺服器執行期環境，可簡化分析應用程式的作業，因此您不必設定、最佳化、保護或操作叢集。

您可以在 中存放資料， AWS Secrets Manager 然後在 EMR Serverless 組態中使用秘密 ID。如此一來，您就不會以純文字格式傳遞敏感的組態資料並將其暴露給外部 API。

如需詳細資訊，請參閱《Amazon EMR Serverless 使用指南》中的透過 EMR Serverless 使用 Secrets Manager 進行資料保護。

Amazon EventBridge 如何使用 AWS Secrets Manager

Amazon EventBridge 是無伺服器事件匯流排服務，可讓您用於將應用程式與來自各種來源的資料互相連線。

當您建立 Amazon EventBridge API 目的地時，EventBridge 會將其連線存放在字首為 的 Secrets Manager 受管秘密中`events`。存放秘密的成本包含在使用 API 目的地的費用中。若要更新秘密，必須使用 EventBridge 而不是 Secrets Manager。如需詳細資訊，請參閱《Amazon EventBridge 使用者指南》中的 API 目的地。

Amazon FSx 如何使用 AWS Secrets Manager 秘密

Amazon FSx for Windows File Server 提供全受管 Microsoft Windows 檔案伺服器，這些伺服器由完全原生的 Windows 檔案系統支援。當您建立或管理檔案共享時，您可以從 AWS Secrets Manager 秘密傳遞登入資料。如需詳細資訊，請參閱《Amazon FSx for Windows File Server 使用者指南》中的檔案共享和將檔案共享組態遷移至 Amazon FSx。

AWS Glue DataBrew 如何使用 AWS Secrets Manager

AWS Glue DataBrew 是一種視覺化資料準備工具，您可以用來清理和標準化資料，而無需撰寫任何程式碼。在 DataBrew 中，一組資料轉換步驟稱為配方。 AWS Glue DataBrew 提供`DETERMINISTIC_DECRYPT`、 `DETERMINISTIC_ENCRYPT`和 `CRYPTOGRAPHIC_HASH`配方步驟，以對資料集中的個人身分識別資訊 (PII) 執行轉換，該資料集使用存放在 Secrets Manager 秘密中的加密金鑰。如果您使用 DataBrew 預設秘密來存放加密金鑰，DataBrew 會建立字首為 的受管秘密`databrew`。存放秘密的成本包含在使用 DataBrew 的費用中。如果您建立新秘密來儲存加密金鑰，DatabreW 會建立一個字首為 `AwsGlueDataBrew` 的秘密。您需要為該秘密付費。

AWS Glue Studio 如何使用 AWS Secrets Manager

AWS Glue Studio 是一種圖形界面，可讓您輕鬆地在其中建立、執行和監控擷取、轉換和載入 (ETL) 任務 AWS Glue。您可以在其中設定 Elasticsearch Spark Connector，以使用 Amazon OpenSearch Service 做為擷取、轉換和載入 (ETL) 任務的資料存放區 AWS Glue Studio。若要連線到 OpenSearch 叢集，您可以在機密管理員中使用機密。如需詳細資訊，請參閱《 AWS Glue 開發人員指南》中的教學課程：使用 Elasticsearch 的 AWS Glue Connector。

AWS IoT SiteWise 如何使用 AWS Secrets Manager

AWS IoT SiteWise 是一項受管服務，可讓您大規模收集、建模、分析和視覺化工業設備的資料。您可以使用 AWS IoT SiteWise 主控台來建立閘道。然後新增連線到閘道的資料來源、本機伺服器或工業設備。如果來源需要身分驗證，請使用機密進行身分驗證。如需詳細資訊，請參閱《AWS IoT SiteWise 使用者指南》中的設定資料來源身分驗證。

Amazon Kendra 如何使用 AWS Secrets Manager

Amazon Kendra 是一項高度精確且智慧的搜尋服務，讓您的使用者能夠使用自然語言處理和進階搜尋演算法來搜尋非結構化和結構化的資料。

您可以透過指定包含資料庫憑證的機密來編制資料庫中存放的文件的索引。如需詳細資訊，請參閱《Amazon Kendra 使用者指南》中的使用資料庫資料來源。

Amazon Kinesis Video Streams 如何使用 AWS Secrets Manager

您可以使用 Amazon Kinesis Video Streams 連線至客戶據點的 IP 攝影機、在本機錄製及存放攝影機拍攝的影片，以及將影片串流到雲端以便長期儲存、播放及分析處理。若要從 IP 攝影機錄製及上傳媒體，您必須將 Kinesis Video Streams Edge Agent 部署至 AWS IoT Greengrass。存取串流至攝影機的媒體檔案所需的認證，會儲存於 Secret Manager 秘密中。如需詳細資訊，請參閱《Amazon Kinesis Video Streams 開發人員指南》中的將 Amazon Kinesis Video Streams Edge Agent 部署至 AWS IoT Greengrass。

AWS Launch Wizard 如何使用 AWS Secrets Manager

AWS Launch Wizard for Active Directory 是一項服務，可套用 AWS 雲端 應用程式最佳實務，引導您設定新的 Active Directory 基礎設施，或將網域控制站新增至 AWS 雲端 或內部部署中的現有基礎設施。

AWS Launch Wizard 需要將網域管理員登入資料新增至 Secrets Manager，才能將您的網域控制站加入 Active Directory。如需詳細資訊，請參閱AWS Launch Wizard 《 使用者指南》中的設定 AWS Launch Wizard for Active Directory。

Amazon Lookout for Metrics 如何使用 AWS Secrets Manager

Amazon Lookout for Metrics 是一項服務，可尋找資料中的異常、判斷其根本原因，並可讓您快速採取動作。您可以使用 Amazon Redshift 或 Amazon RDS 作為 Lookout for Metrics 偵測器的資料來源。若要設定資料來源，請使用包含資料庫密碼的機密。如需詳細資訊，請參閱《Amazon Lookout for Metrics 開發人員指南》中的使用 Amazon RDS 搭配 Lookout for Metrics 和使用 Amazon Redshift 搭配 Lookout for Metrics。

Amazon Managed Grafana 如何使用 AWS Secrets Manager

Amazon Managed Grafana 是一項完全受管的、安全的資料視覺化服務，可供您立即查詢、關聯及視覺化多個來源的運作指標、日誌和追蹤。在將 Amazon Redshift 用作資料來源時，您可以使用 AWS Secrets Manager 秘密來提供 Amazon Redshift 憑證。如需詳細資訊，請參閱《Amazon Managed Grafana 使用者指南》中的設定 Amazon Redshift。

AWS Managed Services 如何使用 AWS Secrets Manager

AWS Managed Services 是一種企業服務，可提供基礎設施 AWS 的持續管理。AMS 自助式佈建 (SSP) 模式提供 AMS 受管帳戶中原生 AWS 服務 和 API 功能的完整存取權。如需如何在 AMS 中要求存取 Secrets Manager 的相關資訊，請參閱《AMS 進階使用者指南》中的 AWS Secrets Manager (AMS 自助式佈建)。

Amazon Managed Streaming for Apache Kafka 如何使用 AWS Secrets Manager

Amazon Managed Streaming for Apache Kafka (Amazon MSK) 是一項全受管服務，可讓您建置和執行使用 Apache Kafka 處理串流資料的應用程式。您可以使用利用 AWS Secrets Manager所存放和保護的使用者名稱和密碼來控制 Amazon MSK 羣集的存取權。如需詳細資訊，請參閱 Amazon Managed Streaming for Apache Kafka 開發人員指南中的 AWS Secrets Manager的使用者名稱和密碼身分驗證。

Amazon Managed Workflows for Apache Airflow 如何使用 AWS Secrets Manager

Amazon Managed Workflows for Apache Airflow 是 Apache Airflow 的受管協同運作服務，可讓您更輕鬆地大規模在雲端中設定和操作end-to-end資料管道。

您可以使用機密管理員機密來設定 Apache Airflow 連線。如需詳細資訊，請參閱《Amazon Managed Workflows for Apache Airflow 使用者指南》中的使用 Secrets Manager 秘密在 中 AWS Secrets Manager 為 Apache Airflow 變數使用私密金鑰設定 Apache Airflow 連線。

AWS Marketplace

當您使用 AWS Marketplace Quick Launch 時， 會將您的軟體與授權金鑰一起 AWS Marketplace 分發。 會將授權金鑰 AWS Marketplace 存放在您的帳戶中，做為 Secrets Manager 受管秘密。儲存秘

密的成本會包含在 的費用中 AWS Marketplace。若要更新秘密，必須使用 AWS Marketplace 而不是 Secrets Manager。如需詳細資訊，請參閱AWS Marketplace 《賣方指南》中的使用進行快速組態設定。

AWS Migration Hub 如何使用 AWS Secrets Manager

AWS Migration Hub 提供單一位置來追蹤跨多個 AWS 工具和合作夥伴解決方案的遷移任務。

AWS Migration Hub Orchestrator 可簡化和自動化伺服器和企業應用程式的遷移 AWS。Migration Hub Orchestrator 使用機密作為與來源伺服器的連線資訊。如需詳細資訊，在《AWS Migration Hub Orchestrator 使用者指南》中，請參閱：

- 將 SAP NetWeaver 應用程式遷移至 AWS
- 在 Amazon EC2 上重新託管應用程式

Migration Hub 策略建議為您的應用程式的可行轉換路徑提供了遷移和現代化策略建議。策略建議可以使用連線資訊的機密來分析 SQL Server 資料庫。如需詳細資訊，請參閱策略建議資料庫分析。

AWS Panorama 如何使用 Secrets Manager

AWS Panorama 是一項將電腦視覺帶入現場部署攝影機網路的服務。您可以使用 AWS Panorama 註冊設備、更新其軟體，以及將應用程式部署到其中。當您將影片串流註冊為應用程式的資料來源時，如果串流受密碼保護， 會將其登入資料 AWS Panorama 存放在 Secrets Manager 秘密中。如需詳細資訊，請參閱《AWS Panorama 開發人員指南》中的管理 AWS Panorama中的攝影機串流。

AWS Parallel Computing Service 如何使用 AWS Secrets Manager

AWS 平行運算服務 (AWS PCS) 是一種受管服務，可讓您在 上更輕鬆地執行和擴展高效能運算 (HPC) 和分散式機器學習工作負載 AWS。

若要連線至叢集任務排程器， AWS PCS 會使用 字首建立受管秘密pcs，以存放排程器金鑰。儲存秘密的成本包含在 PCS 的費用 AWS 中。 AWS PCS 會在您刪除 AWS PCS 叢集時自動刪除秘密。如需詳細資訊，請參閱 AWS PCS 使用者指南中的在 PCS 中使用叢集秘密。 AWS

> **Important**
>
> 請勿修改或刪除 AWS PCS 叢集秘密。

AWS ParallelCluster 如何使用 AWS Secrets Manager

AWS ParallelCluster 是一種開放原始碼叢集管理工具，可用來在 中部署和管理高效能運算 (HPC) 叢集 AWS 雲端。您可以建立多個使用者環境，其中包含與 Managed Microsoft AD AWS ParallelCluster (Active Directory) 整合的 AWS 。 AWS ParallelCluster 使用 Secrets Manager 秘密來驗證 Active Directory 的登入。如需詳細資訊，請參閱《AWS ParallelCluster 使用者指南》中的整合 Active Directory。

Amazon Q 如何使用 Secrets Manager

若要驗證 Amazon Q 以存取您的資料來源，您可以使用 Secrets Manager 秘密將資料來源存取憑證提供給 Amazon Q。如果您使用 主控台，您可以選擇建立新的秘密或使用現有的秘密。如需詳細資訊，請參閱《Amazon Q 開發人員指南》中的概念 – 身分驗證。

Amazon OpenSearch Ingestion如何使用 Secrets Manager

Amazon OpenSearch Ingestion 是全受管、無伺服器資料收集器，可將即時日誌、指標和追蹤資料串流至 Amazon OpenSearch Service 網域和 OpenSearch Serverless集合。您可以使用 OpenSearch Ingestion管道搭配 Secrets Manager 來安全地管理您的登入資料。如需詳細資訊，請參閱：

- 搭配 使用 OpenSearch Ingestion管道 Atlassian Services
- 搭配 Amazon DocumentDB 使用 OpenSearch Ingestion管道
- 搭配 使用 OpenSearch Ingestion管道 Confluent Cloud Kafka
- 搭配 使用 OpenSearch Ingestion管道 Kafka
- 使用 OpenSearch 從自我管理Amazon的 OpenSearch 叢集遷移資料 Ingestion

AWS OpsWorks for Chef Automate 如何使用 AWS Secrets Manager

AWS OpsWorks 是一種組態管理服務，可協助您使用 OpsWorks for Puppet Enterprise 或 在雲端企業中設定和操作應用程式 AWS OpsWorks for Chef Automate。

當您在 中建立新伺服器時 AWS OpsWorks CM，OpsWorks CM 會將伺服器的資訊存放在字首為 的 Secrets Manager 受管秘密中`opsworks-cm`。秘密的成本包含在 的費用中 AWS OpsWorks。如需詳細資訊，請參閱《AWS OpsWorks 使用者指南》中的與 AWS Secrets Manager整合。

Amazon QuickSight 如何使用 AWS Secrets Manager

Amazon QuickSight 是雲端規模商業智慧 (BI) 服務，您可以用來進行分析、資料視覺化和報告。您可以在 QuickSight 中使用各種資料來源。如果您在 Secrets Manager 秘密中存放資料庫登入資料，QuickSight 可以使用這些秘密來連線至資料庫。如需詳細資訊，請參閱《Amazon QuickSight 使用者指南》中的使用 AWS Secrets Manager 秘密取代 Amazon QuickSight 中的資料庫登入資料。
Amazon QuickSight

Amazon RDS 如何使用 AWS Secrets Manager

Amazon Relational Database Service (Amazon RDS) 是一項 Web 服務，可以讓 AWS 雲端中關聯式資料庫的設定、操作和擴展更加簡單。

若要管理 Amazon Relational Database Service (Amazon RDS) 的主要使用者憑證，包括 Aurora，Amazon RDS 可以為您建立受管秘密。您需要為該秘密付費。Amazon RDS 也會管理這些憑證的輪換。如需詳細資訊，請參閱《Amazon RDS 使用者指南》中的使用 Amazon RDS 和 AWS Secrets Manager進行密碼管理。

若為其他 Amazon RDS 憑證，請參閱 建立秘密。

當您使用 Amazon RDS 查詢編輯器連線到資料庫時，可將資料庫的憑證存放在 Secrets Manager 中。如需詳細資訊，請參閱《Amazon RDS 使用者指南》中的使用查詢編輯器。

Amazon Redshift 如何使用 AWS Secrets Manager

Amazon Redshift 是一種在雲端中完全受管的 PB 級資料倉儲服務。

若要管理 Amazon Redshift 的管理員登入資料，Amazon Redshift 可以為您建立受管秘密。您需要為該秘密付費。Amazon Redshift 也會管理這些登入資料的輪換。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的使用 AWS Secrets Manager管理 Amazon Redshift 管理員密碼。

如需其他 Amazon Redshift 憑證的資訊，請參閱 建立秘密。

當您呼叫 Amazon Redshift 資料 API 時，您可以在機密管理員中使用機密來傳遞叢集的憑證。如需詳細資訊，請參閱使用 Amazon Redshift 資料 API。

當您使用 Amazon Redshift 查詢編輯器連線到資料庫時，Amazon Redshift 會將您的憑證存放在 Secrets Manager 秘密中，字首為 `redshiftqueryeditor`。您需要為該秘密付費。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的使用查詢編輯器來查詢資料庫。

如需查詢編輯器第 2 版，請參閱 the section called “Amazon Redshift 查詢編輯器第 2 版”。

Amazon Redshift 查詢編輯器第 2 版

Amazon Redshift 查詢編輯器第 2 版是一種網頁式 SQL 用戶端應用程式，可用來在 Amazon Redshift 資料倉儲上編寫和執行查詢。當您使用 Amazon Redshift 查詢編輯器 v2 連線到資料庫時，Amazon Redshift 可以將登入資料存放在字首為 的 Secrets Manager 受管秘密中`sqlworkbench`。存放秘密的成本包含在使用 Amazon Redshift 的費用中。若要更新秘密，必須使用 Amazon Redshift 而不是 Secrets Manager。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的使用查詢編輯器第 2 版。

如需先前的查詢編輯器，請參閱 the section called “Amazon Redshift”。

Amazon SageMaker AI 如何使用 AWS Secrets Manager

SageMaker AI 是一種全受管的機器學習服務。透過 SageMaker AI，資料科學家和開發人員可以快速輕鬆地建置和訓練機器學習模型，然後將它們直接部署到生產就緒的託管環境中。它提供一個整合式 Jupyter 編寫筆記本執行個體，可以方便地存取資料來源以進行探索和分析，因此您不必管理伺服器。

您可建立 Git 儲存器與 Jupyter 筆記本執行個體的關聯性，以節約即使停止或刪除筆記本執行個體，仍保留在來源控制環境中的筆記本。您可以使用機密管理員來管理私有儲存庫憑證。如需詳細資訊，請參閱《Amazon SageMaker AI 開發人員指南》中的將 Git 儲存庫與 Amazon SageMaker 筆記本執行個體建立關聯。 Amazon SageMaker

若要從 Databricks 匯入資料，Data Wrangler 會將您的 JDBC URL 存放在機密管理員中。如需詳細資訊，請參閱從 Databricks (JDBC) 匯入資料。

若要從 Snowflake 匯入資料，Data Wrangler 會將您的憑證存放在機密管理員機密中。如需詳細資訊，請參閱從 Snowflake 匯入資料。

AWS Schema Conversion Tool 如何使用 AWS Secrets Manager

您可以使用 AWS Schema Conversion Tool (AWS SCT) 將現有的資料庫結構描述從一個資料庫引擎轉換為另一個資料庫引擎。您可以轉換關聯式 OLTP 結構描述，或資料倉儲結構描述。轉換後的結構描述適用於 Amazon Relational Database Service (Amazon RDS) MySQL、MariaDB、Oracle、SQL Server、PostgreSQL 資料庫、Amazon Aurora 資料庫叢集或 Amazon Redshift 叢集。轉換後的結構描述也可以與 Amazon Elastic Compute Cloud 執行個體上的資料庫搭配使用，或儲存為 S3 儲存貯體上的資料。

當您轉換資料庫結構描述時， AWS SCT 可以使用您存放的資料庫登入資料 AWS Secrets Manager。如需詳細資訊，請參閱AWS Schema Conversion Tool 《 使用者指南AWS Secrets Manager 》中的在 AWS SCT 使用者介面中使用 。

InfluxDB 的 Amazon Timestream 使用方式 AWS Secrets Manager

Timestream for InfluxDB 是受管時間序列資料庫引擎，可讓您在使用開放原始碼 APIs AWS 的即時時間序列應用程式上，輕鬆地在 上執行 InfluxDB 資料庫。使用 Timestream for InfluxDB，您可以設定、操作和擴展時間序列工作負載，以單位數毫秒查詢回應時間回應查詢。

當您為 InfluxDB 資料庫建立 Timestream 時，Timestream 會自動建立秘密來存放管理員登入資料。如需詳細資訊，請參閱 Timestream 開發人員指南中的 Timestream for InfluxDB 如何使用秘密。

AWS Toolkit for JetBrains 如何使用 AWS Secrets Manager

AWS Toolkit for JetBrains 是 JetBrains 整合開發環境 (IDEs) 的開放原始碼外掛程式。工具組可讓開發人員輕鬆開發、偵錯和部署使用 AWS的無伺服器應用程式。使用工具組連線到 Amazon Redshift 叢集時，您可以使用機密管理員機密進行身分驗證。如需詳細資訊，請參閱《AWS Toolkit for JetBrains 使用者指南》中的存取 Amazon Redshift 叢集。

AWS Transfer Family 如何使用 AWS Secrets Manager 秘密

AWS Transfer Family 是一種安全傳輸服務，可讓您將檔案傳入和傳出 AWS 儲存服務。

Transfer Family 現在支援使用適用性聲明 2 (AS2) 通訊協定的伺服器使用基本驗證。您可以建立新的 Secrets Manager 秘密，或為您的憑證選擇現有秘密。如需詳細資訊，請參閱《AWS Transfer Family 使用者指南》中的適用於 AS2 連接器的身分驗證。

若要驗證 Transfer Family 使用者，您可以使用 AWS Secrets Manager 做為身分提供者。如需詳細資訊，請參閱AWS Transfer Family 《 使用者指南》中的使用自訂身分提供者和部落格文章啟用密碼身分驗證以 AWS Transfer Family 使用 AWS Secrets Manager 。

您可以搭配 Transfer Family 使用工作流程處理的檔案，使用 Pretty Good Privacy (PGP) 解密。若要在工作流程步驟中使用解密，請提供您在 Secrets Manager 中管理的 PGP 金鑰。如需詳細資訊，請參閱《AWS Transfer Family 使用者指南》中的 Generate and manage PGP keys (產生和管理 PGP 金鑰)。

AWS Wickr如何使用 AWS Secrets Manager 秘密

AWS Wickr 是一種end-to-end加密服務，可協助組織和政府機構透過one-to-one和群組傳訊、語音和視訊通話、檔案共用、螢幕共用等方式安全地通訊。您可以使用 Wickr 資料保留機器人自動執行工作流程。如果機器人可以存取 AWS 服務，則您應該建立 Secrets Manager 秘密來存放機器人登入資料。如需詳細資訊，請參閱《 AWS Wickr 管理指南》中的啟動資料保留機器人。

在 中建立 AWS Secrets Manager 秘密 AWS CloudFormation

您可以使用 CloudFormation 範本中的 `AWS::SecretsManager::Secret` 資源，在 CloudFormation 堆疊中建立秘密，如 建立秘密 中所示。

若要為 Amazon RDS 或 Aurora 建立管理員密碼，建議您在 `AWS::RDS::DBCluster` 中使用 `ManageMasterUserPassword`。Amazon RDS 接著會為您建立秘密並管理輪換。如需詳細資訊，請參閱受管輪換。

針對 Amazon Redshift 和 Amazon DocumentDB 憑證，首先使用 Secrets Manager 產生的密碼建立秘密，然後使用動態參考從秘密中擷取使用者名稱和密碼，當作新資料庫的憑證。接下來，使用 `AWS::SecretsManager::SecretTargetAttachment` 資源，將資料庫相關詳細資訊，新增至 Secrets Manager 必須輪換秘密的秘密。最後，若要開啟自動輪換，請使用 `AWS::SecretsManager::RotationSchedule` 資源並提供輪換函數和排程。請參閱以下範例：

- 使用 Amazon Redshift 憑證建立秘密
- 使用 Amazon DocumentDB 憑證建立秘密

使用 `AWS::SecretsManager::ResourcePolicy` 資源將資源政策連接至秘密。

如需使用 建立資源的相關資訊 AWS CloudFormation，請參閱 AWS CloudFormation 《 使用者指南》中的了解範本基本概念。您也可以使用 AWS Cloud Development Kit (AWS CDK)。如需詳細資訊，請參閱 AWS Secrets Manager 建構程式庫。

使用 建立 AWS Secrets Manager 秘密 AWS CloudFormation

此範例會建立名為 **CloudFormationCreatedSecret-*a1b2c3d4e5f6*** 的秘密。秘密值是以下 JSON，是建立秘密時產生的 32 個字元的密碼。

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

此範例將使用以下 CloudFormation 資源：

- `AWS::SecretsManager::Secret`

如需使用 建立資源的相關資訊 AWS CloudFormation，請參閱 AWS CloudFormation 《 使用者指南》中的了解範本基本概念。

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 建立具有自動輪換的 AWS Secrets Manager 秘密和具有 的 Amazon RDS MySQL 資料庫執行個體 AWS CloudFormation

若要為 Amazon RDS 或 Aurora 建立管理員密碼，建議您使用 `ManageMasterUserPassword`，如 `AWS::RDS::DBCluster` 中的範例「為主要密碼建立 Secrets Manager 秘密」所示。Amazon RDS 接著會為您建立秘密並管理輪換。如需詳細資訊，請參閱受管輪換。

使用 建立 AWS Secrets Manager 秘密和 Amazon Redshift 叢集 AWS CloudFormation

若要建立 Amazon Redshift 的管理秘密，建議您使用 AWS::Redshift::Cluster和 的範例AWS::RedshiftServerless::Namespace。

使用 建立 AWS Secrets Manager 秘密和 Amazon DocumentDB 執行個體 AWS CloudFormation

此範例會使用秘密中的憑證作為使用者和密碼，來建立秘密和 Amazon DocumentDB 執行個體。該秘密連接了資源型政策，可定義能夠存取秘密的人員。範本亦會從 輪換函數範本 建立 Lambda 輪換函數，並將秘密設定為每月第一天在上午 8:00 至 10:00 (UTC) 之間自動輪換。作為安全最佳實務，執行個體位於 Amazon VPC 中。

此範例將以下 CloudFormation 資源用於 Secrets Manager：

- AWS::SecretsManager::Secret
- AWS::SecretsManager::SecretTargetAttachment
- AWS::SecretsManager::RotationSchedule

如需使用 建立資源的相關資訊 AWS CloudFormation，請參閱 AWS CloudFormation 《 使用者指南》中的了解範本基本概念。

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 如何使用 AWS CloudFormation

當您使用主控台開啟輪換時，Secrets Manager 會使用 AWS CloudFormation 來建立輪換的資源。如果您在該過程中建立新的輪換函數，`AWS::Serverless::Function`會根據適當的 AWS CloudFormation 建立 輪換函數範本。然後 AWS CloudFormation 設定 `RotationSchedule`，這會設定秘密的輪換函數和輪換規則。您可以在開啟自動輪換之後，選擇橫幅中的檢視堆疊來檢視 AWS CloudFormation 堆疊。

如需瞭解開啟自動輪換功能的相關資訊，請參閱輪換 秘密。

在 中建立 AWS Secrets Manager 秘密 AWS Cloud Development Kit (AWS CDK)

要在 CDK 應用程式中建立、管理和擷取秘密，可以使用 AWS Secrets Manager 建構程式庫，其中包含 `ResourcePolicy`、`RotationSchedule`、`Secret`、`SecretRotation` 和 `SecretTargetAttachment` 建構模組。

在 CDK 應用程式中使用秘密的最佳實務是先使用主控台或 CLI 建立秘密，然後將秘密匯入您的 CDK 應用程式。

如需範例，請參閱：

- 建立秘密
- 匯入秘密
- 擷取秘密
- 授予秘密使用許可
- 輪換秘密
- 輪換資料庫秘密
- 將秘密複寫至其他區域

如需有關 CDK 的詳細資訊，請參閱《AWS Cloud Development Kit (AWS CDK) v2 開發人員指南》。

監控 AWS Secrets Manager 秘密

AWS 提供監控工具來監看 Secrets Manager 秘密、報告錯誤，以及適時採取自動動作。如果您需要針對任何未預期的使用或變更進行調查，則可使用此日誌，然後還原不想要的變更。您也可以針對不當使用秘密以及任何刪除秘密的嘗試設定自動檢查。

主題

- 使用 記錄 AWS Secrets Manager 事件 AWS CloudTrail
- AWS Secrets Manager 使用 Amazon CloudWatch 監控
- 使用 Amazon EventBridge 比對 AWS Secrets Manager 事件
- 監控何時存取排程刪除的 AWS Secrets Manager 秘密
- 使用 監控 AWS Secrets Manager 秘密的合規性 AWS Config
- 監控 Secrets Manager 成本
- 使用 Amazon GuardDuty 偵測威脅

使用 記錄 AWS Secrets Manager 事件 AWS CloudTrail

AWS CloudTrail 會將 Secrets Manager 的所有 API 呼叫記錄為事件，包括來自 Secrets Manager 主控台的呼叫，以及用於輪換和秘密版本刪除的數個其他事件。如需 Secrets Manager 記錄中日誌項目的清單，請參閱 CloudTrail 事件。

您可以使用 CloudTrail 主控台檢視過去 90 天所記錄的事件。若要持續記錄您 AWS 帳戶中的事件，包括 Secrets Manager 的事件，請建立追蹤，以便 CloudTrail 將日誌檔案交付至 Amazon S3 儲存貯體。請參閱為 AWS 您的帳戶建立追蹤。您也可以設定 CloudTrail，以接收來自多個 AWS 帳戶 和 AWS 區域 的日誌檔案。

您可以設定其他 AWS 服務，以進一步分析 CloudTrail 日誌中收集的資料並對其採取行動。請參閱 AWS 服務與 CloudTrail 日誌的整合。當 CloudTrail 發佈新的日誌檔案到您的 Amazon S3 儲存貯體時，您也會收到通知。請參閱設定 CloudTrail 的 Amazon SNS 通知。

若要從 CloudTrail 日誌中擷取 Secrets Manager 事件 (主控台)

1. 前往 https://console.aws.amazon.com/cloudtrail/ 開啟 CloudTrail 主控台。
2. 確保主控台指向事件發生的區域。主控台只會顯示所選區域中發生的事件。從主控台右上角的下拉式清單中選擇區域。

3. 在左側導覽窗格中，選擇 Event history (事件歷史記錄)。
4. 選擇 Filter (篩選) 標準和/或 Time range (時間範圍)，以協助找到您在尋找的事件。例如：
   a. 若要查看所有 Secrets Manager 事件，請針對查詢屬性選擇事件來源。然後，針對輸入事件來源，選擇 **secretsmanager.amazonaws.com**。
   b. 若要查看秘密的所有事件，請針對查詢屬性選擇資源名稱。然後，針對輸入資源名稱，輸入秘密的名稱。
5. 若要查看其他詳細資訊，請選擇事件旁的展開箭頭。若要查看所有可用的資訊，請選擇 View event (檢視事件)。

AWS CLI

Example 從 CloudTrail 日誌擷取 Secrets Manager 事件

下列 `lookup-events` 範例會查詢 Secrets Manager 事件。

```
aws cloudtrail lookup-events \
    --region us-east-1 \
    --lookup-attributes
 AttributeKey=EventSource,AttributeValue=secretsmanager.amazonaws.com
```

AWS CloudTrail Secrets Manager 的項目

AWS Secrets Manager 會將所有 Secrets Manager 操作和其他輪換和刪除相關事件的項目寫入 AWS CloudTrail 日誌。如需如何對這些事件採取動作的相關資訊，請參閱 使用 EventBridge 比對 Secrets Manager 事件。

日誌項目類型

- Secrets Manager 操作的日誌項目
- 要刪除的日誌項目
- 複寫的日誌項目
- 要輪換的日誌項目

Secrets Manager 操作的日誌項目

呼叫 Secrets Manager 操作所產生的事件具有 `"detail-type": ["AWS API Call via CloudTrail"]`。

> **Note**
>
> 在 2024 年 2 月之前，某些 Secrets Manager 操作報告了包含秘密 ARN 的「aRN」而非「arn」的事件。如需詳細資訊，請參閱 AWS re:Post。

以下是當您或服務透過 API、SDK 或 CLI 呼叫 Secrets Manager 操作時產生的 CloudTrail 項目。

BatchGetSecretValue

由 BatchGetSecretValue 操作產生。如需瞭解擷取秘密的相關資訊，請參閱 取得秘密。

CancelRotateSecret

由 CancelRotateSecret 操作產生。如需輪換的相關資訊，請參閱輪換 秘密。

CreateSecret

由 CreateSecret 操作產生。如需瞭解建立秘密的相關資訊，請參閱 管理秘密。

DeleteResourcePolicy

由 DeleteResourcePolicy 操作產生。如需許可的相關資訊，請參閱the section called “身分驗證與存取控制”。

DeleteSecret

由 DeleteSecret 操作產生。如需了解刪除秘密的相關資訊，請參閱the section called “刪除秘密”。

DescribeSecret

由 DescribeSecret 操作產生。

GetRandomPassword

由 GetRandomPassword 操作產生。

GetResourcePolicy

由 GetResourcePolicy 操作產生。如需許可的相關資訊，請參閱the section called “身分驗證與存取控制”。

GetSecretValue

由 GetSecretValue 和 BatchGetSecretValue 操作產生。如需瞭解擷取秘密的相關資訊，請參閱 取得秘密。

ListSecrets

由 ListSecrets 操作產生。如需瞭解列出秘密的相關資訊，請參閱 the section called “查找秘密”。

ListSecretVersionIds

由 ListSecretVersionIds 操作產生。

PutResourcePolicy

由 PutResourcePolicy 操作產生。如需許可的相關資訊，請參閱the section called “身分驗證與存取控制”。

PutSecretValue

由 PutSecretValue 操作產生。如需瞭解更新秘密的相關資訊，請參閱 the section called “修改秘密”。

RemoveRegionsFromReplication

由 RemoveRegionsFromReplication 操作產生。如需複寫秘密的相關資訊，請參閱多區域複寫。

ReplicateSecretToRegions

由 ReplicateSecretToRegions 操作產生。如需複寫秘密的相關資訊，請參閱多區域複寫。

RestoreSecret

由 RestoreSecret 操作產生。如需還原的相關資訊，請參閱the section called “還原秘密”。

RotateSecret

由 RotateSecret 操作產生。如需輪換的相關資訊，請參閱輪換 秘密。

StopReplicationToReplica

由 StopReplicationToReplica 操作產生。如需複寫秘密的相關資訊，請參閱多區域複寫。

TagResource

由 TagResource 操作產生。如需瞭解標記秘密的相關資訊，請參閱 the section called “標籤 秘密”。

UntagResource

由 UntagResource 操作產生。如需瞭解取消秘密標記的相關資訊，請參閱 the section called “標籤秘密”。

UpdateSecret

由 UpdateSecret 操作產生。如需了解更新秘密的相關資訊，請參閱the section called “修改秘密”。

UpdateSecretVersionStage

由 UpdateSecretVersionStage 操作產生。如需版本階段的相關資訊，請參閱the section called “秘密版本”。

ValidateResourcePolicy

由 ValidateResourcePolicy 操作產生。如需許可的相關資訊，請參閱the section called “身分驗證與存取控制”。

要刪除的日誌項目

除了 Secrets Manager 操作的事件之外，Secrets Manager 還會產生下列與刪除相關的事件。這些事件具有 `"detail-type": ["AWS Service Event via CloudTrail"]`。

CancelSecretVersionDelete

由 Secrets Manager 服務產生。如果您在具有多個版本的秘密上呼叫 `DeleteSecret`，然後呼叫 `RestoreSecret`，Secrets Manager 會針對所還原的各個密碼版本記錄這個事件。如需還原的相關資訊，請參閱the section called “還原秘密”。

EndSecretVersionDelete

刪除機密版本時由 Secrets Manager 服務產生。如需詳細資訊，請參閱the section called “刪除秘密”。

StartSecretVersionDelete

Secrets Manager 開始刪除機密版本時由 Secrets Manager 服務產生。如需瞭解刪除秘密的相關資訊，請參閱 the section called “刪除秘密”。

SecretVersionDeletion

Secrets Manager 刪除淘汰的秘密版本時由 Secrets Manager 服務所產生。如需詳細資訊，請參閱秘密版本。

複寫的日誌項目

除了 Secrets Manager 操作的事件之外，Secrets Manager 還會產生下列與複寫相關的事件。這些事件具有 `"detail-type": ["AWS Service Event via CloudTrail"]`。

ReplicationFailed

複寫失敗時由 Secrets Manager 服務產生。如需複寫秘密的相關資訊，請參閱多區域複寫。

ReplicationStarted

Secrets Manager 開始複寫秘密時由 Secrets Manager 服務產生。如需複寫秘密的相關資訊，請參閱多區域複寫。

ReplicationSucceeded

成功複寫秘密時由 Secrets Manager 服務產生。如需複寫秘密的相關資訊，請參閱多區域複寫。

要輪換的日誌項目

除了 Secrets Manager 操作的事件之外，Secrets Manager 還會產生下列與輪換相關的事件。這些事件具有 `"detail-type": ["AWS Service Event via CloudTrail"]`。

RotationStarted

Secrets Manager 開始輪換機密時由 Secrets Manager 服務產生。如需輪換的相關資訊，請參閱輪換 秘密。

RotationAbandoned

Secrets Manager 放棄嘗試輪換，並從現有的機密版本移除 `AWSPENDING` 標籤時由 Secrets Manager 服務產生。如果您在輪換期間建立新的秘密版本，Secrets Manager 便會放棄輪換。如需輪換的相關資訊，請參閱輪換 秘密。

RotationFailed

輪換失敗時由 Secrets Manager 服務產生。如需輪換的相關資訊，請參閱the section called “輪換疑難排解”。

RotationSucceeded

已成功輪換機密時由 Secrets Manager 服務產生。如需輪換的相關資訊，請參閱輪換 秘密。

TestRotationStarted

Secrets Manager 針對未排程立即輪換的機密開始測試輪換時由 Secrets Manager 服務產生。如需輪換的相關資訊，請參閱輪換 秘密。

TestRotationSucceeded

Secrets Manager 針對未排程立即輪換的機密成功測試輪換時由 Secrets Manager 服務產生。如需輪換的相關資訊，請參閱輪換 秘密。

TestRotationFailed

Secrets Manager 針對未排程立即輪換的機密測試輪換，且輪換失敗時由 Secrets Manager 服務產生。如需輪換的相關資訊，請參閱the section called “輪換疑難排解”。

AWS Secrets Manager 使用 Amazon CloudWatch 監控

使用 Amazon CloudWatch，您可以監控 AWS 服務並建立警示，讓您知道指標何時變更。CloudWatch 會保留這些統計資料 15 個月，因此您可以存取歷史資訊，並更清楚 Web 應用程式或服務的效能。對於 AWS Secrets Manager，您可以監控帳戶中的秘密數量，包括標記為刪除的秘密，以及對 Secrets Manager 的 API 呼叫，包括透過主控台進行的呼叫。如需如何監控指標的資訊，請參閱《CloudWatch 使用者指南》中的使用 CloudWatch 指標。 CloudWatch

尋找 Secrets Manager 指標

1. 在 CloudWatch 主控台的指標下，選擇所有指標。
2. 在指標搜尋方塊中，輸入 `secret`。
3. 請執行下列操作：
   - 若要監控帳戶中的秘密數量，請選擇 AWS/SecretsManager，然後選取 SecretCount。此指標每小時發佈一次。
   - 若要監控對 Secrets Manager 的 API 呼叫，包括透過主控台進行的呼叫，請選擇用量 > 依 AWS 資源，然後選擇要監控的 API 呼叫。如需 Secrets Manager APIs的清單，請參閱 Secrets Manager 操作。
4. 請執行下列操作：
   - 若要建立指標的圖形，請參閱《Amazon CloudWatch 使用者指南》中的繪製指標圖形。
   - 若要偵測異常，請參閱《Amazon CloudWatch 使用者指南》中的使用 CloudWatch 異常偵測。 Amazon CloudWatch
   - 若要取得指標的統計資料，請參閱《Amazon CloudWatch 使用者指南》中的取得指標的統計資料。

CloudWatch 警示

您可以建立 CloudWatch 警示，在指標值變更並導致警示變更狀態時傳送 Amazon SNS 訊息。您可以在 Secrets Manager 指標 上設定警示`ResourceCount`，這是您帳戶中的秘密數目。您也可以在 警示監看您指定期間內的指標，並根據在數個期間內相對於指定閾值的指標值執行動作。警示僅會針對持續

狀態變更調用動作。CloudWatch 警示不會只因處於特定狀態就調用動作，狀態必須已變更並已維持一段指定的時間。

如需詳細資訊，請參閱《CloudWatch 使用者指南》中的使用 Amazon CloudWatch 警示和根據異常偵測建立 CloudWatch 警示。 CloudWatch

您也可以設定留意特定閾值的警示，當滿足這些閾值時傳送通知或採取動作。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南。

使用 Amazon EventBridge 比對 AWS Secrets Manager 事件

在 Amazon EventBridge 中，您可以比對 CloudTrail 日誌項目中的 Secrets Manager 事件。您可以設定尋找這些事件的 EventBridge 規則，然後將新產生的事件傳送至目標以採取動作。如需 Secrets Manager 記錄的 CloudTrail 項目清單，請參閱 CloudTrail 事件。如需設定 EventBridge 的說明，請參閱《EventBridge 使用者指南》中的 EventBridge 入門。

比對指定機密的所有變更

> **Note**
>
> 由於部分 Secrets Manager 事件會以不同的大小寫傳回機密的 ARN，因此在比對多個動作的事件模式中，若要透過 ARN 指定機密，您可能需要同時包含金鑰 `arn` 和 `aRN`。如需詳細資訊，請參閱 AWS re:Post。

下列範例顯示用於比對機密變更之日誌項目的 EventBridge 事件模式。

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

機密值輪換時比對事件

下列範例顯示用於比對從手動更新或自動輪換發生之機密值變更的 CloudTrail 日誌項目的 EventBridge 事件模式。由於其中一些事件來自 Secrets Manager 操作，一些由 Secrets Manager 服務產生，因此您必須包含兩者的 `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"]
    }
}
```

監控何時存取排程刪除的 AWS Secrets Manager 秘密

您可以使用 Amazon CloudWatch Logs AWS CloudTrail和 Amazon Simple Notification Service (Amazon SNS) 的組合來建立警示，以通知您嘗試存取秘密待刪除。如果您收到警示的通知，您可以取消秘密的刪除，讓您有更多時間決定是否真的想要刪除該秘密。您調查到最後可能會還原秘密，因為您仍需要此秘密。或者，您可能需要為使用新秘密的使用者，提供最新的詳細資訊。

下列程序說明在對 `GetSecretValue` 操作提出請求而導致特定的錯誤訊息寫入 CloudTrail 日誌檔時，如何收到通知。可對秘密執行其他 API 操作，無需觸發警示。此 CloudWatch 警示會偵測可能指出使用過期登入資料之人員或應用程式的使用情況。

開始這些程序之前，您必須在您想要監控 AWS Secrets Manager API 請求的 AWS 區域 和 帳戶中開啟 CloudTrail。如需說明，請前往 AWS CloudTrail 使用者指南中的首次建立追蹤。

步驟 1：設定 CloudTrail 日誌檔案交付至 CloudWatch Logs

您必須將 CloudTrail 日誌檔設定為交付到 CloudWatch Logs。這樣做可讓 CloudWatch Logs 監控它們，以使 Secrets Manager API 請求擷取待刪除的秘密。

設定將 CloudTrail 日誌檔交付至 CloudWatch Logs

1. 前往 https://console.aws.amazon.com/cloudtrail/ 開啟 CloudTrail 主控台。

2. 在頂端導覽列上，選擇 AWS 區域 來監控秘密。
3. 在左側導覽窗格中，選擇 Trails (追蹤)，然後選擇要為 CloudWatch 設定的追蹤名稱。
4. 在 Trails Configuration (追蹤組態) 頁面，向下捲動到 CloudWatch Logs 部分，然後選擇編輯圖示 (✏)。
5. 在 New or existing log group (新建或現有的日誌群組) 中，輸入日誌群組的名稱，例如 **`CloudTrail/MyCloudWatchLogGroup`**。
6. 對於 IAM role (IAM 角色)，您可以使用名為 CloudTrail_CloudWatchLogs_Role 的預設角色。此角色有預設的角色政策，內含將 CloudTrail 事件交付至日誌群組所需的許可。
7. 選擇 Continue (繼續) 來儲存您的組態。
8. 在 AWS CloudTrail 將與帳戶中 API 活動相關的 CloudTrail 事件交付至 CloudWatch Logs 日誌群組頁面，選擇 Allow (允許)。

步驟 2：建立 CloudWatch 警示

若要在 Secrets Manager `GetSecretValue` API 操作請求存取待刪除的秘密時收到通知，您必須建立 CloudWatch 警示並設定通知。

建立 CloudWatch 警示

1. 前往 https://console.aws.amazon.com/cloudwatch/ 登入 CloudWatch 主控台。
2. 在頂端導覽列上，選擇您要監控秘密 AWS 的區域。
3. 在左側導覽窗格中，選擇 Logs (日誌)。
4. 在 Log Groups (日誌群組) 中，選取您在之前程序中建立的日誌群組旁的核取方塊，例如 CloudTrail/MyCloudWatchLogGroup。然後，選擇 Create Metric Filter (建立指標篩選條件)。
5. 在 Filter Pattern (篩選條件模式) 中，請輸入或貼上下列內容：

   ```
   { $.eventName = "GetSecretValue" && $.errorMessage = "*secret because it was marked
    for deletion*" }
   ```

   選擇 Assign Metric (指派指標)。
6. 在 Create Metric Filter and Assign a Metric (建立指標篩選條件並指定指標) 頁面上，請執行下列動作：

   a. 針對 Metric Namespace (指標命名空間)，輸入 **`CloudTrailLogMetrics`**。

b. 針對 Metric Name (指標名稱)，輸入 **`AttemptsToAccessDeletedSecrets`**。

c. 選擇 Show advanced metric settings (顯示進階指標設定)，然後在必要時於 Metric Value (指標值) 中輸入 **1**。

d. 選擇 Create Filter (建立篩選條件)。

7. 在篩選條件方塊中，選擇 Create Alarm (建立警示)。

8. 在 Create Alarm (建立警示) 視窗中，請執行下列動作：

a. 在 Name (名稱) 輸入 **`AttemptsToAccessDeletedSecretsAlarm`**。

b. 在 Whenever: (無論何時：) 的 is: (是) 中，選擇 **>=**，然後輸入 **1**。

c. 在 Send notification to: (傳送通知給：) 欄位旁，執行以下其中一項：

- 若要建立和使用新的 Amazon SNS 主題，請選擇 New list (新清單)，然後輸入新的主題名稱。對於 Email list: (電子郵件清單：) 欄位，請輸入至少一個電子郵件地址。您可以利用逗號分隔來輸入多個電子郵件地址。
- 若要使用現有的 Amazon SNS 主題，請選擇要使用的主題名稱。如果清單不存在，請選擇 Select list (選取清單)。

d. 選擇建立警示 。

步驟 3：測試 CloudWatch 警示

若要測試警示，請建立秘密，然後將其排定刪除。接著嘗試擷取秘密值。您很快就會在警示中設定的地址收到電子郵件。它會提醒您使用的秘密已排定要刪除。

使用 監控 AWS Secrets Manager 秘密的合規性 AWS Config

您可以使用 AWS Config 來評估秘密，以查看它們是否符合您的標準。您可以使用 AWS Config 規則來定義秘密的內部安全和合規要求。然後， AWS Config 可以識別不符合您規則的秘密。您也可以追蹤秘密中繼資料、輪換組態、用於秘密加密的 KMS 金鑰、Lambda 輪換函數以及與秘密相關聯的標籤的變更。

您可以設定 AWS Config 來通知您變更。如需詳細資訊，請參閱AWS Config 傳送至 Amazon SNS 主題的通知。

如果您的組織中有多個 AWS 帳戶 和 AWS 區域 的秘密，您可以彙總該組態和合規資料。如需詳細資訊，請參閱多帳戶多區域資料彙總。

評估秘密是否合規

- 遵循使用 AWS Config 規則評估資源的指示，然後選擇下列其中一個規則：
  - secretsmanager-secret-unused – 檢查是否在指定的天數內存取秘密。
  - secretsmanager-using-cmk — 檢查秘密是否使用 或您建立的客戶受管金鑰進行 AWS 受管金鑰 aws/secretsmanager加密 AWS KMS。
  - secretsmanager-rotation-enabled-check – 檢查是否為 Secrets Manager 中存放的秘密設定了輪換。
  - secretsmanager-scheduled-rotation-success-check – 檢查上次成功輪換是否在設定的輪換頻率範圍內。檢查的最低頻率為每日。
  - secretsmanager-secret-periodic-rotation – 檢查是否在指定的天數內輪換秘密。

監控 Secrets Manager 成本

您可以使用 Amazon CloudWatch 監控預估 AWS Secrets Manager 費用。如需詳細資訊，請參閱《CloudWatch 使用者指南》中的建立帳單警示以監控預估 AWS 費用。

另一種監控成本的選項是 AWS 成本異常偵測。如需詳細資訊，請參閱《 成本管理使用者指南》中的使用 AWS 成本異常偵測來偵測異常支出。 AWS

如需有關監控 Secrets Manager 用量的資訊，請參閱 the section called “使用 CloudWatch 監控”和 the section called “使用 記錄 AWS CloudTrail ”。

如需 AWS Secrets Manager 定價的相關資訊，請參閱 the section called “定價”。

使用 Amazon GuardDuty 偵測威脅

Amazon GuardDuty 是一種威脅偵測服務，可協助您使用 AWS 環境來保護您的帳戶、容器、工作負載和資料。透過使用機器學習 (ML) 模型和異常和威脅偵測功能，GuardDuty 會持續監控不同的日誌來源，以識別環境中的潛在安全風險和惡意活動，並排定其優先順序。例如，GuardDuty 將偵測潛在威脅，例如異常或可疑的秘密存取，以及憑證洩漏，以防它透過執行個體啟動角色偵測專門為 Amazon EC2 執行個體建立，但正從其中另一個帳戶使用中的登入資料 AWS。如需詳細資訊，請參閱 Amazon GuardDuty 使用者指南。

另一個用於偵測的範例使用案例是異常行為。例如，如果 AWS Secrets Manager 通常使用 Java 開發套件從實體取得 create-secret、describe-secret、 get-secret-value和 list-secrets

呼叫，然後不同的實體開始 AWS CLI 從 VPN 外部呼叫`batch-get-secret-value`和`get-secret-value`使用 ，GuardDuty 可以報告第二個實體異常呼叫 APIs 的調查結果。如需詳細資訊，請參閱 GuardDuty IAM 調查結果類型 CredentialAccess：IAMUser/AnomalousBehavior。

的合規驗證 AWS Secrets Manager

使用 Secrets Manager 時的合規責任取決於資料的敏感度、您公司的合規目標，以及適用的法律和法規。 AWS 提供下列資源以協助合規：

- 安全與合規快速入門指南：這些部署指南討論架構考量，並提供在 AWS上部署以安全及合規為重心之基準環境的步驟。
- HIPAA 安全與合規架構白皮書 – 此白皮書說明公司如何使用 AWS 來建立符合 HIPAA 規範的應用程式。
- AWS 合規資源 – 此工作手冊和指南的集合可能適用於您的產業和位置。
- AWS Config 可評定資源組態與內部實務、業界準則和法規的合規狀態。如需詳細資訊，請參閱the section called “監控秘密的合規性”。
- AWS Security Hub 提供 內安全狀態的完整檢視 AWS ，協助您檢查是否符合安全產業標準和最佳實務。如需有關使用 Security Hub 評估 Secrets Manager 資源的資訊，請參閱《AWS Security Hub 使用者指南》中的 AWS Secrets Manager 控制項。
- IAM Access Analyzer 會分析允許外部實體存取秘密的政策 (包括政策中的條件陳述式)。如需詳細資訊，請參閱使用 Access Analyzer 預覽存取。
- AWS Systems Manager 為 Secrets Manager 提供了預先定義的 Runbook。如需詳細資訊，請參閱《適用於 Secrets Manager 的 Systems Manager Automation Runbook 參考》。
- 您可以使用 下載第三方稽核報告 AWS Artifact。如需詳細資訊，請參閱在 中下載報告 AWS Artifact。

合規標準

AWS Secrets Manager 已針對下列標準進行稽核，而且當您需要取得合規認證時， 可以成為您解決方案的一部分。

- HIPAA – AWS 已將其健康保險流通與責任法案 (HIPAA) 合規計劃擴展為包含 AWS Secrets Manager 為符合 HIPAA 資格的服務。如果您與 簽訂了已執行的商業夥伴協議 (BAA) AWS，您可以使用 Secrets Manager 協助建置符合 HIPAA 規範的應用程式。 為有興趣進一步了解客戶 AWS 提供以 HIPAA 為重心的白皮書，讓他們了解如何利用這些白皮書 AWS 來處理和儲存健康資訊。如需詳細資訊，請參閱 HIPAA 合規。
- PCI 參與組織 – 在服務提供者層級 1 AWS Secrets Manager 具有支付卡產業 (PCI) 資料安全標準 (DSS) 3.2 版的合規證明。使用 AWS 產品和服務來存放、處理或傳輸持卡人資料的客戶，可以在管

理自己的 PCI DSS 合規認證 AWS Secrets Manager 時使用 。如需 PCI DSS 的詳細資訊，包括如何請求 AWS PCI 合規套件的副本，請參閱 PCI DSS 第 1 級。

- ISO – AWS Secrets Manager 已成功完成 ISO/IEC 27001、ISO/IEC 27017、ISO/IEC 27018 和 ISO 9001 的合規認證。如需詳細資訊，請參閱 ISO 27001、ISO 27017、ISO 27018 及 ISO 9001。
- AICPA SOC – 系統和組織控制 (SOC) 報告是獨立的第三方檢查報告，示範 Secrets Manager 如何實現關鍵合規控制和目標。這些報告的目的是協助您和稽核人員了解為支援操作和合規而建立的 AWS 控制項。如需詳細資訊，請參閱 SOC 合規。
- FedRAMP – 聯邦風險與授權管理計劃 (FedRAMP) 是全政府的計劃，提供標準化方法來評估、授權和持續監控雲端產品和服務。FedRAMP 計畫還針對東部/西部和 GovCloud 的服務和區域提供臨時授權，以使用政府或監管資料。如需詳細資訊，請參閱 FedRAMP 合規。
- 國防部 - 國防部 (DoD) 雲端運算安全要求指南 (SRG) 為雲端服務提供者 (CSPs) 提供標準化的評估和授權程序，以取得 DoD 臨時授權，以便其可為 DoD 客戶提供服務。如需詳細資訊，請參閱 DoD SRG 資源
- IRAP – 資訊安全註冊評估商計劃 (IRAP) 可讓澳洲政府客戶驗證是否已實施適當的控制，並判斷適當的責任模型，以解決澳洲網路安全中心 (ACSC) 產生的澳洲政府資訊安全手冊 (ISM) 的要求。如需詳細資訊，請參閱 IRAP 資源
- OSPAR – Amazon Web Services (AWS) 已達成委外服務提供者的稽核報告 (OSPAR) 認證。與新加坡銀行協會 (ABS) 委外服務提供者控制目標和程序準則 (ABS 準則） AWS 一致，證明了客戶 AWS 承諾滿足新加坡金融服務業對雲端服務提供者設定的高度期望。如需詳細資訊，請參閱 OSPAR 資源

中的安全性 AWS Secrets Manager

安全 AWS 是最高優先順序。身為 AWS 客戶，您可以受益於資料中心和網路架構，其建置旨在滿足最安全敏感組織的需求。

您和 AWS 共同承擔安全責任。共同責任模型將此描述為雲端本身的安全和雲端內部的安全：

- 雲端的安全性 – AWS 負責保護在 AWS Cloud 中執行 AWS 服務的基礎設施。 AWS 也為您提供可安全使用的服務。在 AWS 合規計劃中，第三方稽核員會定期測試並驗證我們的安全功效。若要了解適用的合規計劃 AWS Secrets Manager，請參閱AWS 合規計劃範圍內的服務。
- 雲端的安全性 – 您的 AWS 服務會決定您的責任。您也必須對其他因素負責，包括資料的機密性、您公司的要求和適用法律和法規。

如需更多資源，請參閱安全支柱 – AWS Well-Architected Framework。

主題

- 降低使用 AWS CLI 存放 AWS Secrets Manager 秘密的風險
- 的身分驗證和存取控制 AWS Secrets Manager
- 中的資料保護 AWS Secrets Manager
- 中的秘密加密和解密 AWS Secrets Manager
- 中的基礎設施安全性 AWS Secrets Manager
- 使用 AWS Secrets Manager VPC 端點
- 使用 IAM 政策控制 API 存取
- 中的彈性 AWS Secrets Manager
- 後量子 TLS

降低使用 AWS CLI 存放 AWS Secrets Manager 秘密的風險

當您使用 AWS Command Line Interface (AWS CLI) 叫用 AWS 操作時，您可以在命令 shell 中輸入這些命令。例如，您可以使用 Windows 命令提示字元或 Windows PowerShell，或 Bash 或 Z shell 等等。其中許多命令 Shell 包含旨在提高生產力的功能。但是，此功能可以用來洩露您的秘密。例如，在大多數 Shell 中，您可以使用向上鍵來查看最後輸入的命令。此「命令歷程記錄」功能可能被存取不安全工作階段的任何人濫用。此外，其他在背景執行的公用程式可能具有存取您命令參數的權限 (旨在協助您更有效率地執行任務)。為了降低這類風險，請確認您採取以下步驟：

- 當您離開主控台時，請務必鎖上電腦。
- 解除安裝或停用您不需要或不再使用的主控台公用程式。
- 確保 Shell 和遠端存取程式 (若您有使用) 不會記錄輸入的命令。
- 使用技術來傳遞不會被 Shell 命令歷程記錄擷取的參數。下列範例示範如何將秘密文字輸入文字檔案中，然後將檔案傳遞至 AWS Secrets Manager 命令並立即銷毀檔案。這表示典型 shell 歷史記錄不會擷取秘密文字。

  以下範例顯示典型 Linux 命令 (但您的 Shell 可能需要有些許不同的命令)：

```
$ 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.
```

在您執行這些命令後，應要能夠使用上下箭號來捲動命令歷程記錄，並可看到秘密文字未顯示於任何列。

> **Important**
>
> 在預設情況下，您必須先將命令歷史記錄緩衝區的大小降低至 1，否則無法在 Windows 中執行相同技術。

若要設定 Windows 命令提示字元為只有 1 個命令的 1 個歷程記錄緩衝區

1. 開啟管理員命令提示字元，或選擇 Run as administrator (以系統管理員身分執行)。
2. 選擇左上角的圖示，然後選擇屬性。

3. 在 Options (選項) 索引標籤上，將 Buffer Size (緩衝區大小) 及 Number of Buffers (緩衝區數量) 皆設定為 **1**，然後選擇 OK (確定)。
4. 當您需要輸入不想留存於歷史記錄中的命令時，請立即接續輸入另一個命令，例如：

   ```
   echo.
   ```

   這可確保您刷新敏感命令。

對於 Windows 命令提示字元 Shell，您可以下載 SysInternals SDelete 工具，然後使用類似以下的命令：

```
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.
```

的身分驗證和存取控制 AWS Secrets Manager

Secrets Manager 會使用 AWS Identity and Access Management (IAM) 來保護秘密的存取權。IAM 提供身分驗證與存取控制。身分驗證會驗證個人請求的身分。Secrets Manager 會使用登入程序、密碼、存取金鑰和多重要素驗證 (MFA) 字符來驗證使用者的身分。請參閱登入 AWS。存取控制可確保只有經核准的個人可對 AWS 資源 (例如秘密) 執行操作。Secrets Manager 使用政策來定義誰可以存取哪些資源，以及相應身分可以對那些資源採取哪些動作。請參閱 IAM 中的政策和許可。

主題

- 的許可參考 AWS Secrets Manager
- Secrets Manager 管理員許可
- 存取秘密的許可

- Lambda 輪換函數的許可
- 用於加密金鑰的許可
- 複寫的許可
- 身分型政策
- 資源型政策
- 使用屬性型存取控制 (ABAC) 控制對秘密的存取
- AWS 的 受管政策 AWS Secrets Manager
- 判斷誰有權存取您的 AWS Secrets Manager 秘密
- 從不同帳戶存取 AWS Secrets Manager 秘密
- 從內部部署環境存取秘密

的許可參考 AWS Secrets Manager

Secrets Manager 的許可參考可在服務授權參考中的 的動作、資源和條件索引鍵 AWS Secrets Manager中找到。

Secrets Manager 管理員許可

若要授予 Secrets Manager 管理員許可，請遵循新增與移除 IAM 身分許可中的說明，並連接下列政策：

- SecretsManagerReadWrite
- IAMFullAccess

建議您不要將管理員許可授予最終使用者。雖然這樣做可讓使用者建立及管理其秘密，但啟用輪換所需的許可 (IAMFullAccess) 會授予最終使用者不適用的重要許可。

存取秘密的許可

您可以利用 IAM 許可政策，藉此控制可以存取秘密的使用者或服務。許可政策描述哪些人可在哪些資源上執行哪些動作。您可以：

- the section called “身分型政策”
- the section called “資源型政策”

Lambda 輪換函數的許可

Secrets Manager 使用 AWS Lambda 函數來輪換秘密。Lambda 函數必須能夠存取秘密，以及該秘密包含其憑證的資料庫或服務。請參閱 輪換的許可。

用於加密金鑰的許可

Secrets Manager 使用 AWS Key Management Service (AWS KMS) 金鑰來加密秘密。 AWS 受管金鑰 `aws/secretsmanager` 會自動擁有正確的許可。如果使用不同的 KMS 金鑰，Secrets Manager 需要該金鑰的許可。請參閱 the section called “KMS 金鑰的許可”。

複寫的許可

透過使用 IAM 許可政策，您可以控制哪些使用者或服務可以將秘密複寫到其他區域。請參閱 the section called “防止複寫”。

身分型政策

將許可政策連接到 IAM 身分：使用者、使用者群組和角色。在身分型政策中，您指定該身分可以存取哪些秘密以及該身分可以對該秘密執行哪些動作。如需更多資訊，請參閱《新增和移除 IAM 身分許可》。

您可以向代表其他服務中應用程式或使用者的角色授予許可。例如，在 Amazon EC2 執行個體上執行的應用程式可能需要存取資料庫。您可以建立與 EC2 執行個體設定檔相連的 IAM 角色，然後使用許可政策授予角色存取含有資料庫憑證的機密。如需相關資訊，請參閱《利用 IAM 角色來授予許可給 Amazon EC2 執行個體上執行的應用程式》。其他您可以附加角色以加入 Amazon Redshift、AWS Lambda，和 Amazon ECS 的服務。

您也可以將許可授予經過 IAM 以外的身分系統驗證的使用者。例如，您可將 IAM 角色與使用 Amazon Cognito 登入的行動應用程式使用者建立關聯。角色將利用角色許可政策中的許可來授予應用程式暫時登入資料。然後，您可以使用許可政策將對秘密的存取權授予角色。如需相關資訊，請參閱《身分提供者和聯合》。

您可以使用身分型政策：

- 授予身分對多個秘密的存取權。
- 控制誰可以建立新秘密，以及誰可以存取尚未建立的秘密。
- 授予 IAM 群組對秘密的存取權。

範例：

- 範例：擷取每個秘密值的許可
- 範例：讀取和描述個別秘密的許可
- 範例：擷取批次中一組秘密值的許可
- 範例：萬用字元
- 範例：建立秘密的許可
- 範例：拒絕加密秘密的特定 AWS KMS 金鑰

範例：擷取每個秘密值的許可

若要授予擷取秘密值的許可，您可以將政策連接至秘密或身分。如需判斷要使用哪種政策的說明，請參閱身分型政策和資源型政策。如需連接政策的相關資訊，請參閱 the section called “資源型政策” 和 the section called “身分型政策”。

如果想要將存取權授予 IAM 群組，此範例很有用。若要授與批次 API 呼叫中擷取一組秘密的權限，請參閱the section called “範例：擷取批次中一組秘密值的許可”。

Example 讀取使用客戶受管金鑰加密的秘密

如果秘密是使用客戶受管金鑰加密，您可以透過將下列政策連接到身分來授予讀取秘密的存取權。 \

JSON

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

範例：讀取和描述個別秘密的許可

Example 讀取並描述一個秘密

您可以透過將下列政策連接至身分，授予對秘密的存取權。

JSON

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

範例：擷取批次中一組秘密值的許可

Example 讀取批次中的一組秘密

您可以透過將下列政策連接至身分，授予在 API 呼叫擷取一組秘密的存取權。此原則會限制呼叫者，以便他們只能擷取 *SecretARN1*、*SecretARN2* 和 *SecretARN3* 所指定的密碼，即使批次呼叫包含其他機密也是如此。如果呼叫者也要求批次 API 呼叫中的其他秘密，秘密管理員將不會傳回它們。如需詳細資訊，請參閱 BatchGetSecretValue。

JSON

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

```
    "secretsmanager:BatchGetSecretValue",
    "secretsmanager:ListSecrets"
  ],
  "Resource": "*"
},
{
  "Effect": "Allow",
  "Action": [
    "secretsmanager:GetSecretValue"
  ],
  "Resource": [
    "SecretARN1",
    "SecretARN2",
    "SecretARN3"
    ]
}
]
}
```

範例：萬用字元

您可以使用萬用字元在政策元素中包含一組值。

Example 存取路徑中的所有秘密

下列政策會授予存取權來擷取名稱開頭為 "*TestEnv/*" 的所有秘密。

JSON

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

Example 存取所有秘密上的中繼資料

下列政策授予 DescribeSecret 和開頭為 List (ListSecrets 及 ListSecretVersionIds) 的許可。

JSON

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

Example 比對秘密名稱

下列政策會依名稱授予秘密的所有 Secrets Manager 許可。若要使用此政策，請參閱 the section called “身分型政策”。

若要比對秘密名稱，您可以將區域、帳戶 ID、秘密名稱和萬用字元 (?) 放在一起，以此比對單個隨機字元，從而建立秘密的 ARN。Secrets Manager 會將六個隨機字元連接到秘密名稱作為其 ARN 的一部分，因此您可以使用此萬用字元來比對那些字元。如果您使用的是語法 "another_secret_name-*"，則 Secrets Manager 不僅會比對含有 6 個隨機字元的所需秘密，還會比對 "another_secret_name-<anything-here>a1b2c3"。

您可以預測 ARN 除了 6 個隨機字元以外的所有部分，因此使用萬元字元 '??????' 語法可讓您安全地將許可授予尚不存在的秘密。不過，請注意，如果您刪除秘密，然後以相同名稱將其重建，使用者會自動收到新秘密的許可，即使那 6 個字元已變更。

JSON

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

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

範例：建立秘密的許可

若要將建立秘密的許可授予使用者，建議您將許可政策連接至使用者所屬的 IAM 群組。請參閱 IAM 使用者群組。

Example 建立秘密

下列政策會授予建立秘密和檢視秘密清單的許可。若要使用此政策，請參閱 the section called “身分型政策”。

JSON

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

範例：拒絕加密秘密的特定 AWS KMS 金鑰

> **Important**
>
> 若要拒絕客戶受管金鑰，建議您使用金鑰政策或金鑰授予來限制存取。如需詳細資訊，請參閱《 AWS Key Management Service 開發人員指南》中的 的身分驗證和存取控制 AWS KMS。

Example 拒絕 AWS 受管金鑰 **aws/secretsmanager**

下列政策拒絕使用 AWS 受管金鑰 `aws/secretsmanager`來建立或更新秘密。此政策要求使用客戶受管金鑰加密秘密。政策包含兩個陳述式：

1. 第一個陳述式 `Sid: "RequireCustomerManagedKeysOnSecrets"`拒絕使用 建立或更新秘密的請求 AWS 受管金鑰 `aws/secretsmanager`。
2. 第二個陳述式 `Sid: "RequireKmsKeyIdParameterOnCreate"`拒絕建立不包含 KMS 金鑰之秘密的請求，因為 Secrets Manager 會預設為使用 AWS 受管金鑰 `aws/secretsmanager`。

JSON

```
{
"Version": "2012-10-17",
"Statement": [
  {
      "Sid": "RequireCustomerManagedKeysOnSecrets",
      "Effect": "Deny",
      "Action": [
          "secretsmanager:CreateSecret",
          "secretsmanager:UpdateSecret"
      ],
      "Resource": "*",
      "Condition": {
          "StringLikeIfExists": {
              "secretsmanager:KmsKeyArn": "<key_ARN_of_the_AWS_managed_key>"
          }
      }
  },
  {
      "Sid": "RequireKmsKeyIdParameterOnCreate",
      "Effect": "Deny",
```

```
      "Action": "secretsmanager:CreateSecret",
      "Resource": "*",
      "Condition": {
          "Null": {
              "secretsmanager:KmsKeyArn": "true"
          }
      }
  }
]
}
```

資源型政策

在資源型政策中，您可以指定能夠存取秘密的人員，以及他們可以對秘密執行的動作。您可以使用資源型政策：

- 將單一秘密的存取權授予多個使用者和角色。
- 將存取權授予其他 AWS 帳戶中的使用者或角色。

在將資源型政策連接至主控台中的秘密時，Secrets Manager 會使用自動推理引擎 Zelkova 和 API `ValidateResourcePolicy`，防止您將秘密的存取權授予各種 IAM 委託人。您也可以透過 CLI 或開發套件呼叫帶有 `BlockPublicPolicy` 參數的 `PutResourcePolicy` API。

> **Important**
>
> 資源政策驗證和 `BlockPublicPolicy` 參數可防止透過直接連接到秘密的資源政策授予公開存取權，以協助保護您的資源。除了使用這些功能之外，請仔細檢查下列政策，以確認它們未授予公有存取權：
>
> - 連接到相關聯 AWS 主體的身分型政策（例如 IAM 角色）
> - 連接至相關聯 AWS 資源的資源型政策（例如， AWS Key Management Service (AWS KMS) 金鑰）
>
> 若要檢閱秘密的許可，請參閱 判斷誰有存取 秘密的許可。

若要檢視、變更或刪除秘密的資源政策 (主控台)

1. 於 https://console.aws.amazon.com/secretsmanager/ 開啟 Secrets Manager 主控台。
2. 從秘密清單中選擇秘密。
3. 在秘密詳細資訊頁面的概觀分頁上，在資源許可區段中，選擇編輯許可。
4. 在程式碼欄位中，執行以下其中一項作業，然後選擇 Save (儲存)：
   - 若要連接或修改資源政策，請輸入該政策。
   - 若要刪除政策，請清除程式碼欄位。

AWS CLI

Example 擷取資源政策

下列 `get-resource-policy` 範例會擷取連接至機密的以資源為基礎的政策。

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

Example 刪除資源政策

下列 `delete-resource-policy` 範例會刪除連接至機密的以資源為基礎的政策。

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

Example 新增資源政策

下列 `put-resource-policy` 範例會將許可政策新增至機密，首先檢查政策是否不提供機密的廣泛存取權限。系統會從檔案讀取政策。如需詳細資訊，請參閱 AWS CLI 《使用者指南》中的從檔案載入 AWS CLI 參數。

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

`mypolicy.json` 的內容：

JSON

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

AWS 開發套件

若要擷取與秘密相連的政策，請使用 GetResourcePolicy。

若要刪除與秘密相連的政策，請使用 DeleteResourcePolicy。

若要將政策連接至秘密，請使用 PutResourcePolicy。如果原本就已連接政策，命令會以新政策取而代之。政策必須格式化為 JSON 結構化文字。請參閱 JSON 政策文件結構。

如需詳細資訊，請參閱the section called “AWS SDKs”。

範例

範例：

- 範例：擷取每個秘密值的許可
- 範例：許可和 VPC
- 範例：服務委託人

範例：擷取每個秘密值的許可

若要授予擷取秘密值的許可，您可以將政策連接至秘密或身分。如需判斷要使用哪種政策的說明，請參閱身分型政策和資源型政策。如需連接政策的相關資訊，請參閱 the section called “資源型政策” 和 the section called “身分型政策”。

如果想要將對單一秘密的存取權授予多個使用者或角色，此範例非常有用。若要授與批次 API 呼叫中擷取一組秘密的權限，請參閱the section called “範例：擷取批次中一組秘密值的許可”。

Example 讀取一個秘密

您可以透過將下列政策連接至秘密，授予對秘密的存取權。

JSON

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

範例：許可和 VPC

如果您需要從 VPC 中存取 Secrets Manager，則可以透過在許可政策中包含條件，來確保對 Secrets Manager 的請求都來自 VPC。如需詳細資訊，請參閱限制具有 VPC 端點條件的請求及the section called “VPC 端點 (AWS PrivateLink)”。

請確定從其他 AWS 服務存取秘密的請求也來自 VPC，否則此政策會拒絕他們存取。

Example 需要請求才能透過 VPC 端點

例如，以下政策允許使用者執行 Secrets Manager 操作，但僅限請求是透過 VPC 端點 *vpce-1234a5678b9012c* 提出的情況。

JSON

```
{
"Id": "example-policy-1",
```

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

Example 需要來自 VPC 的請求

以下政策只在命令來自 *vpc-12345678* 時，才允許這些命令建立和管理秘密。此外，此政策只在請求來自 vpc-2b2b2b2b 時，才允許存取秘密加密值的操作。如果您在某個 VPC 中執行應用程式，但您使用第二個隔離的 VPC 來執行管理功能，您可能會使用如下的政策。

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"
      }
    }
  }
  ]
  }
```

範例：服務委託人

如果秘密中連接的資源政策包含 AWS 服務委託人，建議您使用 aws:SourceArn 和 aws:SourceAccount 全域條件金鑰。ARN 和帳戶值只會在請求從另一個 AWS 服務來到 Secrets Manager 中時，包含在授權內容中。此條件組合會避免潛在的混淆代理人案例。

若資源 ARN 包含資源政策中不允許的字元，則您便不能將該資源 ARN 用於 `aws:SourceArn` 條件金鑰的值中。請改用 `aws:SourceAccount` 條件金鑰。如需更多資訊，請參閱《IAM 需求》。

服務主體通常不會用作附加至秘密的政策中的主體，但有些 AWS 服務需要它。如需服務要求您連接至秘密之資源政策的相關資訊，請參閱服務的說明文件。

Example 允許服務使用服務委託人存取秘密

JSON

```
{
"Version": "2012-10-17",
"Statement": [
{
  "Effect": "Allow",
  "Principal": {
    "Service": [
      "service-name.amazonaws.com"
    ]
  },
  "Action": "secretsmanager:GetSecretValue",
  "Resource": "*",
  "Condition": {
    "ArnLike": {
      "aws:sourceArn": "arn:aws:service-name::123456789012:*"
    },
    "StringEquals": {
      "aws:sourceAccount": "123456789012"
    }
  }

}
]
}
```

使用屬性型存取控制 (ABAC) 控制對秘密的存取

屬性型存取控制 (ABAC) 是一種授權策略，可根據使用者、資料或環境的屬性或特性定義許可，例如部門、業務單位或可能影響授權結果的其他因素。在 中 AWS，這些屬性稱為標籤。

使用標籤控制許可，在成長快速的環境中相當有幫助，也能在政策管理變得繁瑣時提供協助。ABAC 規則會在執行時間動態評估，這表示使用者對應用程式和資料的存取，以及允許操作的類型會根據政策中的內容因素自動變更。例如，如果使用者變更部門，則會自動調整存取權，而不需要更新許可或請求新角色。如需詳細資訊，請參閱：什麼是 ABAC AWS？、根據標籤定義存取秘密的許可，以及使用 ABAC 搭配 IAM Identity Center 擴展 Secrets Manager 的授權需求。

範例：允許身分存取具有特定標籤的秘密

下列政策允許DescribeSecret存取具有金鑰 *ServerName* 和值 *ServerABC* 之標籤的秘密。如果您將此政策連接至身分，則該身分具有帳戶中具有該標籤之任何秘密的許可。

JSON

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

範例：僅允許存取具有符合秘密標籤之標籤的身分

下列政策允許帳戶中的任何身分GetSecretValue存取帳戶中任何秘密，其中身分*AccessProject*標籤的值與秘密的*AccessProject*標籤相同。

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 受管政策 AWS Secrets Manager

AWS 受管政策是由 AWS AWS 受管政策建立和管理的獨立政策旨在為許多常用案例提供許可，以便您可以開始將許可指派給使用者、群組和角色。

請記住， AWS 受管政策可能不會授予特定使用案例的最低權限許可，因為這些許可可供所有 AWS 客戶使用。我們建議您定義使用案例專屬的客戶管理政策，以便進一步減少許可。

您無法變更 AWS 受管政策中定義的許可。如果 AWS 更新 AWS 受管政策中定義的許可，則更新會影響政策連接的所有主體身分（使用者、群組和角色）。當新的 AWS 服務 啟動或新的 API 操作可供現有 服務使用時， AWS 最有可能更新 AWS 受管政策。

如需詳細資訊，請參閱《IAM 使用者指南》中的 AWS 受管政策。

AWS 受管政策：SecretsManagerReadWrite

此政策提供讀取/寫入存取權 AWS Secrets Manager，包括描述 Amazon RDS、Amazon Redshift 和 Amazon DocumentDB 資源的許可，以及使用 AWS KMS 加密和解密秘密的許可。此政策也提供建立 AWS CloudFormation 變更集、從由 管理的 Amazon S3 儲存貯體取得輪換範本 AWS、列出 AWS Lambda 函數和描述 Amazon EC2 VPCs許可。控制台需要這些許可，才能使用現有的輪換函數來設定輪換。

若要建立新的輪換函數，您還必須具有建立 AWS CloudFormation 堆疊和 AWS Lambda 執行角色的許可。您可以指派 IAMFullAccess 受管政策。請參閱 輪換的許可。

許可詳細資訊

此政策包含以下許可。

- `secretsmanager` – 允許主體執行所有 Secrets Manager 動作。
- `cloudformation` – 允許主體建立 AWS CloudFormation 堆疊。這是必要的，以便使用主控台開啟輪換的主體可以透過 AWS CloudFormation 堆疊建立 Lambda 輪換函數。如需詳細資訊，請參閱the section called “Secrets Manager 如何使用 AWS CloudFormation”。

- ec2 – 允許主體描述 Amazon EC2 VPC。此為必要項目，如此使用主控台的主體才可以在與存放在秘密中的憑證之資料庫相同的 VPC 中建立輪換函數。
- kms – 允許主體使用 AWS KMS 金鑰進行密碼編譯操作。此為必要項目，如此 Secrets Manager 才可以加密和解密秘密。如需詳細資訊，請參閱the section called “秘密加密和解密”。
- lambda – 允許主體列出 Lambda 輪換函數。此為必要項目，如此使用主控台的主體才可以選擇現有的輪換函數。
- rds – 允許主體描述 Amazon RDS 中的叢集和執行個體。此為必要項目，如此使用主控台的主體才可以選擇 Amazon RDS 叢集或執行個體。
- redshift – 允許主體描述 Amazon Redshift 中的叢集。此為必要項目，如此使用主控台的主體才可以選擇 Amazon Redshift 叢集。
- redshift-serverless – 允許主體描述 Amazon Redshift Serverless 中的命名空間。這是必要的，以便使用主控台的主體可以選擇 Amazon Redshift Serverless 命名空間。
- docdb-elastic – 允許主體描述 Amazon DocumentDB 中的彈性叢集。此為必要項目，如此使用主控台的主體才可以選擇 Amazon DocumentDB 彈性叢集。
- tag – 允許主體取得帳戶中已標記的所有資源。
- serverlessrepo – 允許主體建立 AWS CloudFormation 變更集。此為必要項目，如此使用主控台的主體才可以建立 Lambda 輪換函數。如需詳細資訊，請參閱the section called “Secrets Manager 如何使用 AWS CloudFormation”。
- s3 – 允許主體從 管理的 Amazon S3 儲存貯體取得物件 AWS。此儲存貯體包含 Lambda 輪換函數範本。此為必要許可，如此使用主控台的主體才可以根據儲存貯體中的範本建立 Lambda 輪換函數。如需詳細資訊，請參閱the section called “Secrets Manager 如何使用 AWS CloudFormation”。

若要檢視政策，請參閱 SecretsManagerReadWrite JSON 政策文件。

Secrets Manager 更新 AWS 受管政策

檢視 Secrets Manager AWS 受管政策更新的詳細資訊。

變更	描述	日期	版本
SecretsManagerReadWrite – 更新現有政策	此政策已更新為允許描述對 Amazon Redshift Serverless 的存取，以便主控台使用者可以在建	2024 年 3 月 12 日	v5

變更	描述	日期	版本
	立 Amazon Redshift 秘密時選擇 Amazon Redshift Serverless 命名空間。		
SecretsManagerReadWrite – 更新現有政策	此政策已更新，允許描述 Amazon DocumentDB 彈性叢集的存取權，如此主控台使用者才可以在建立 Amazon DocumentDB 秘密時選擇彈性叢集。	2023 年 9 月 12 日	v4
SecretsManagerReadWrite – 更新現有政策	此政策已更新，允許描述 Amazon Redshift 的存取權，如此主控台使用者才可以在建立 Amazon Redshift 秘密時選擇 Amazon Redshift 叢集。更新也新增了新的許可，以允許讀取存取 管理的 Amazon S3 儲存貯體 AWS ，該儲存貯體存放 Lambda 輪換函數範本。	2020 年 6 月 24 日	v3
SecretsManagerReadWrite – 更新現有政策	此政策已更新，允許描述 Amazon RDS 叢集的存取權，如此主控台使用者才可以在建立 Amazon RDS 秘密時選擇叢集。	2018 年 5 月 3 日	v2

變更	描述	日期	版本
SecretsManagerReadWrite – 新政策	Secrets Manager 建立了一個政策來授予使用控制台所需的許可，這些許可具有 Secrets Manager 的所有讀取 / 寫入存取權。	2018 年 4 月 04 日	v1

判斷誰有權存取您的 AWS Secrets Manager 秘密

依預設，IAM 身分沒有存取秘密的許可。在授予秘密的存取權時，Secrets Manager 會評估與秘密相連的資源型政策，以及與 IAM 使用者或傳送請求的角色相連的所有身分型政策。為了這麼做，Secrets Manager 使用類似於 IAM 使用者指南中的判斷是否允許或拒絕請求中所述的程序。

多個政策套用到請求時，Secrets Manager 會使用階層來控制許可：

1. 如果任何政策中具有明確 deny 的陳述式與請求動作和資源相符：

   明確 deny 會覆寫其他所有內容並阻止該動作。
2. 如果沒有明確 deny，而是具有明確 allow 的陳述式與請求動作和資源相符：

   明確 allow 會授予請求中的動作對陳述式中資源的存取權。

   如果身分和秘密位於兩個不同的帳戶中，則秘密的資源政策和連接到身分的政策allow都必須有 ，否則 AWS 會拒絕請求。如需詳細資訊，請參閱跨帳戶存取權。
3. 如果沒有具有明確 allow 的陳述式與請求動作和資源相符：

   AWS 根據預設拒絕請求，這稱為隱含拒絕。

若要檢視秘密的資源型政策

- 執行以下任意一項：
  - 前往以下位置開啟 Secrets Manager 主控台：https://console.aws.amazon.com/secretsmanager/。進入秘密的秘密詳細資訊頁面，在 Resource permissions (資源使用權限) 區段選擇 Edit permissions (編輯許可)。

- 使用 AWS CLI 呼叫 `get-resource-policy`或 AWS SDK 呼叫 `GetResourcePolicy`。

透過身分型政策判斷誰擁有存取權

- 使用 IAM 政策模擬器。請參閱使用 IAM 政策模擬器測試 IAM 政策

從不同帳戶存取 AWS Secrets Manager 秘密

若要允許一個帳戶中的使用者存取另一個帳戶中的秘密 (跨帳戶存取)，您必須同時允許在資源政策和身分政策中的存取權。這與授予和秘密所在的同一帳戶中的身分存取權不同。

跨帳戶許可僅適用於下列操作：

- CancelRotateSecret
- DeleteResourcePolicy
- DeleteSecret
- DescribeSecret
- GetRandomPassword
- GetResourcePolicy
- GetSecretValue
- ListSecretVersionIds
- PutResourcePolicy
- PutSecretValue
- RemoveRegionsFromReplication
- ReplicateSecretToRegions
- RestoreSecret
- RotateSecret
- StopReplicationToReplica
- TagResource
- UntagResource
- UpdateSecret
- UpdateSecretVersionStage
- ValidateResourcePolicy

您可以搭配 PutResourcePolicy 動作使用 BlockPublicPolicy 參數，透過防止透過直接連接到秘密的資源政策授予公開存取，以協助保護您的資源。您也可以使用 IAM Access Analyzer 驗證跨帳戶存取。

您還必須允許身分使用秘密加密所用的 KMS 金鑰。這是因為您無法將 AWS 受管金鑰 (aws/secretsmanager) 用於跨帳戶存取。您必須使用建立的 KMS 金鑰來加密秘密，然後將金鑰政策連接至秘密。建立 KMS 金鑰需支付費用。若要變更秘密的加密金鑰，請參閱 the section called “修改秘密”。

> **Important**
>
> 授予secretsmanager:PutResourcePolicy許可的資源型政策可讓委託人，甚至是其他帳戶中的委託人，能夠修改以資源為基礎的政策。此許可可讓主體提升現有的許可，例如取得秘密的完整管理存取權。建議您將最低權限存取原則套用至您的政策。如需詳細資訊，請參閱資源型政策。

下列範例政策假設您在 Account1 中有秘密和加密金鑰，在 Account2 中有想要允許存取秘密值的身分。

步驟 1：將資源政策連接至 Account1 中的秘密

- 以下政策允許 *Account2* 中的 *ApplicationRole* 存取 *Account1* 中的秘密。若要使用此政策，請參閱 the section called “資源型政策”。

  JSON

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

步驟 2：在 Account1 中將陳述式新增至 KMS 金鑰的金鑰政策

- 下列金鑰政策陳述式允許 *Account2* 中的 *ApplicationRole* 使用 *Account1* 中的 KMS 金鑰來解密 *Account1* 中的秘密。若要使用此陳述式，請將其新增至 KMS 金鑰的金鑰政策。如需詳細資訊，請參閱變更金鑰政策。

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

步驟 3：將身分政策連接至 Account2 中的身分

- 以下政策允許 *Account2* 中的 *ApplicationRole* 存取 *Account1* 中的秘密，並透過使用 *Account1* 中也有的加密金鑰來解密秘密值。若要使用此政策，請參閱 the section called “身分型政策”。在 Secret ARN (秘密 ARN) 下的秘密詳細資訊頁面中，您可以從 Secrets Manager 主控台找到秘密的 ARN。或者，您也可以呼叫 describe-secret。

  JSON

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

```
  ]
}
```

從內部部署環境存取秘密

您可以使用 AWS Identity and Access Management Roles Anywhere 在 IAM 中取得臨時安全登入資料，例如在 外部執行的伺服器、容器和應用程式 AWS。您的工作負載可以使用與 AWS 應用程式搭配使用的相同 IAM 政策和 IAM 角色來存取 AWS 資源。借助 IAM Roles Anywhere，您可以透過 Secrets Manager 來儲存和管理可供 AWS 中的資源存取的憑證，以及應用程式伺服器等內部部署裝置存取的憑證。如需詳細資訊，請參閱《IAM Roles Anywhere 使用者指南》。

中的資料保護 AWS Secrets Manager

AWS 共同責任模型適用於 中的資料保護 AWS Secrets Manager。如此模型所述， AWS 負責保護執行所有 的全域基礎設施 AWS 雲端。您負責維護在此基礎設施上託管內容的控制權。此內容包括您所使用 AWS 服務 的安全組態和管理任務。如需有關資料隱私權的更多相關資訊，請參閱資料隱私權常見問答集。如需有關歐洲資料保護的相關資訊，請參閱 AWS 安全性部落格上的 AWS 共同的責任模型和 GDPR 部落格文章。

基於資料保護目的，我們建議您保護 AWS 帳戶 登入資料，並使用 AWS Identity and Access Management (IAM) 設定個別使用者帳戶。如此一來，每個使用者都只會獲得授予完成其任務所必須的許可。我們也建議您採用下列方式保護資料：

- 每個帳戶均要使用多重要素驗證 (MFA)。
- 使用 SSL/TLS 與 AWS 資源通訊。Secrets Manager 支援所有區域中的 TLS 1.2 和 1.3。Secrets Manager 也支援混合型適用於 TLS (PQTLS) 的後量子金鑰交換選項網路加密通訊協定。
- 使用存取金鑰 ID 和與 IAM 主體關聯的私密存取金鑰來簽署您對 Secrets Manager 的程式設計請求。或者，您可以使用 AWS Security Token Service (AWS STS) 來產生暫時性安全憑證以簽署請求。
- 使用 設定 API 和使用者活動記錄 AWS CloudTrail。請參閱 the section called “使用 記錄 AWS CloudTrail ”。
- 如果您在 AWS 透過命令列界面或 API 存取 時需要 FIPS 140-2 驗證的密碼編譯模組，請使用 FIPS 端點。請參閱 the section called “Secrets Manager 端點”。
- 如果您使用 AWS CLI 存取 Secrets Manager，則為 the section called “降低使用 AWS CLI 存放 AWS Secrets Manager 秘密的風險”。

靜態加密

Secrets Manager 透過 AWS Key Management Service (AWS KMS) 使用加密來保護靜態資料的機密性。 AWS KMS 提供許多 AWS 服務所使用的金鑰儲存和加密服務。Secrets Manager 中的每個秘密，都使用唯一資料金鑰加密。每個資料金鑰都由 KMS 金鑰保護。您可以選擇為該帳戶搭配 Secrets Manager AWS 受管金鑰 使用預設加密，或者可以在 AWS KMS中建立自己的客戶管理金鑰。使用客戶管理金鑰，可讓您更精細進行對 KMS 金鑰活動的授權控制。如需詳細資訊，請參閱the section called “秘密加密和解密”。

傳輸中加密

Secrets Manager 會提供安全且私有的端點，以供您加密傳輸中的資料。安全和私有端點允許 AWS 保護對 Secrets Manager 的 API 請求完整性。 AWS 要求呼叫者使用 X.509 憑證和/或 Secrets Manager Secret Access Key 簽署 API 呼叫。Signature 第 4 版簽署程序 (Sigv4) 規定了這項要求。

如果您使用 AWS Command Line Interface (AWS CLI) 或任何 AWS SDKs來呼叫 AWS，您可以設定要使用的存取金鑰。然後這些工具會自動使用存取金鑰來為您簽署請求。請參閱 the section called “降低使用 AWS CLI 存放 AWS Secrets Manager 秘密的風險”。

網際網路流量隱私權

AWS 透過已知和私有網路路由路由流量時， 提供維護隱私權的選項。

服務和內部部署用戶端與應用程式之間的流量。

您的私有網路與 之間有兩個連線選項 AWS Secrets Manager：

- An AWS Site-to-Site VPN 連線。如需詳細資訊，請參閱什麼是 AWS Site-to-Site VPN？
- AWS Direct Connect 連線。如需詳細資訊，請參閱什麼是 AWS Direct Connect？

相同區域中 AWS 資源之間的流量

如果您想要保護 Secrets Manager 和 API 用戶端之間的流量 AWS，請設定 AWS PrivateLink 以私下存取 Secrets Manager API 端點。

加密金鑰管理

當 Secrets Manager 需要加密受保護秘密資料的新版本時，Secrets Manager 會將請求傳送至 AWS KMS ，以從 KMS 金鑰產生新的資料金鑰。Secrets Manager 使用此資料金鑰進行信封加密。Secrets Manager 會將加密的資料金鑰與加密的秘密一起存放。當秘密需要解密時，Secrets Manager

AWS KMS 會要求解密資料金鑰。Secrets Manager 接著會使用已解密的資料金鑰來解密加密的秘密。Secrets Manager 絕不會以未加密的形式存放資料金鑰，而且會盡快將其從記憶體中移除。如需詳細資訊，請參閱the section called “秘密加密和解密”。

中的秘密加密和解密 AWS Secrets Manager

Secrets Manager 使用信封加密搭配 AWS KMS 金鑰和資料金鑰來保護每個秘密值。每當機密中的機密值變更，Secrets Manager 就會請求 AWS KMS 中的新資料金鑰來進行保護。資料金鑰將在 KMS 金鑰下加密，並會存放在秘密的中繼資料中。若要解密秘密，Secrets Manager 會先使用 中的 KMS 金鑰解密加密的資料金鑰 AWS KMS。

Secrets Manager 不會直接使用 KMS 金鑰來加密秘密值。相反地，它會使用 KMS 金鑰來產生和加密 256 位元的進階加密標準 (AES) 對稱資料金鑰，再使用資料金鑰來加密秘密值。Secrets Manager 使用純文字資料金鑰來加密 外部的秘密值 AWS KMS，然後將其從記憶體中移除。它將加密的資料金鑰副本存放在秘密的中繼資料中。

主題

- 選擇 AWS KMS 金鑰
- 會加密哪些資料？
- 加密和解密程序
- KMS 金鑰的許可
- Secrets Manager 如何使用您的 KMS 金鑰
- AWS 受管金鑰 (aws/secretsmanager) 的金鑰政策
- Secrets Manager 加密內容
- 監控 Secrets Manager 與 的互動 AWS KMS

選擇 AWS KMS 金鑰

建立秘密時，您可以選擇 AWS 帳戶 和 區域中的任何對稱加密客戶受管金鑰，也可以使用 AWS 受管金鑰 適用於 Secrets Manager 的 (`aws/secretsmanager`)。如果您選擇 AWS 受管金鑰 `aws/secretsmanager`且它尚未存在，Secrets Manager 會建立它並將其與秘密建立關聯。對於帳戶中的每個秘密，您可以使用相同的 KMS 金鑰或不同的 KMS 金鑰。建議您使用其他 KMS 金鑰，為一組秘密設定金鑰的自訂許可，或者如果您想要稽核這些金鑰的特定作業。Secrets Manager 只支援對稱加密 KMS 金鑰。如果您在外部金鑰存放區中使用 KMS 金鑰，KMS 金鑰上的密碼編譯作業可能需要更長的時間，且較不可靠和耐用，因為請求必須在 AWS外傳輸。

如需變更秘密的加密金鑰的資訊，請參閱 the section called “變更秘密的加密金鑰”。

當您變更加密金鑰時，Secrets Manager 會使用新金鑰重新加密 AWSPENDING、 AWSCURRENT和 AWSPREVIOUS版本。為了避免將您鎖定在秘密之外，Secrets Manager 會保留所有使用上一個金鑰加密的現有版本。這表示您可以使用先前的金鑰或新的金鑰來解密 AWSCURRENTAWSPENDING、 和 AWSPREVIOUS版本。如果您沒有前一個金鑰的kms:Decrypt許可，當您變更加密金鑰時，Secrets Manager 無法解密秘密版本以重新加密。在此情況下，現有的版本不會重新加密。

若要讓它AWSCURRENT只能由新的加密金鑰解密，請使用新金鑰建立新的秘密版本。然後，若要能夠解密AWSCURRENT秘密版本，您必須擁有新金鑰的許可。

您可以拒絕 的許可， AWS 受管金鑰 aws/secretsmanager並要求使用客戶受管金鑰加密秘密。如需詳細資訊，請參閱the section called “範例：拒絕加密秘密的特定 AWS KMS 金鑰”。

若要尋找與密碼相關聯的 KMS 金鑰，請在主控台中檢視密碼，或呼叫 ListSecrets 或 DescribeSecret。當秘密與 AWS 受管金鑰 適用於 Secrets Manager 的 (aws/secretsmanager) 相關聯時，這些操作不會傳回 KMS 金鑰識別符。

會加密哪些資料？

Secrets Manager 會加密機密值，但不會加密下列項目：

- 秘密名稱和說明
- 輪換設定
- 與秘密相關聯的 KMS 金鑰 ARN
- 任何連接的 AWS 標籤

加密和解密程序

若要加密秘密中的秘密值，Secrets Manager 使用以下程序。

1. Secrets Manager 會使用秘密的 KMS 金鑰 ID 和 256 位元 AES 對稱金鑰的請求來呼叫 AWS KMS GenerateDataKey 操作。 AWS KMS 會傳回純文字資料金鑰和以 KMS 金鑰加密的資料金鑰複本。
2. Secrets Manager 使用純文字資料金鑰和進階加密標準 (AES) 演算法來加密 外部的秘密值 AWS KMS。使用完畢後，它會盡快從記憶體中移除這些純文字金鑰。
3. Secrets Manager 將加密的資料金鑰存放在秘密的中繼資料，以便可以用來解密秘密值。不過，沒有 Secrets Manager API 會傳回加密的秘密或加密的資料金鑰。

若要解密加密秘密值：

1. Secrets Manager 會呼叫 AWS KMS Decrypt 操作並傳入加密的資料金鑰。
2. AWS KMS 使用秘密的 KMS 金鑰來解密資料金鑰。它會傳回純文字資料金鑰。
3. Secrets Manager 使用純文字資料金鑰來解密秘密值。接著，它會盡快從記憶體中移除資料金鑰。

KMS 金鑰的許可

Secrets Manager 在密碼編譯操作中使用 KMS 金鑰時，等於是代表正在存取或更新秘密值的使用者執行動作。您可以在 IAM 政策或金鑰政策中授予許可。下列 Secrets Manager 操作需要 AWS KMS 許可。

- CreateSecret
- GetSecretValue
- PutSecretValue
- UpdateSecret
- ReplicateSecretToRegions

若要允許僅將 KMS 金鑰用於源自 Secrets Manager 的請求，您可以在許可政策中使用 kms:ViaService 條件金鑰搭配 `secretsmanager.<Region>.amazonaws.com` 值。

您也可以使用加密內容中的金鑰或值作為條件金鑰，以便將 KMS 金鑰用於密碼編譯操作。例如，您可以在 IAM 或金鑰政策文件中使用字串條件運算子，或在授權中使用授權限制。KMS 金鑰授予傳播可能需時 5 分鐘。如需詳細資訊，請參閱 CreateGrant。

Secrets Manager 如何使用您的 KMS 金鑰

Secrets Manager 會使用 KMS 金鑰呼叫下列 AWS KMS 操作。

GenerateDataKey

Secrets Manager 會呼叫 AWS KMS GenerateDataKey 操作，以回應下列 Secrets Manager 操作。

- CreateSecret – 如果新的秘密包含秘密值，Secrets Manager 會請求新的資料金鑰來加密它。
- PutSecretValue– Secrets Manager 請求新的資料金鑰來加密指定的秘密值。
- ReplicateSecretToRegions – 為加密複製的秘密，Secrets Manager 會在複寫過程中，使用複本區域中的 KMS 金鑰重新加密資料金鑰。

- UpdateSecret – 如果變更了秘密值或 KMS 金輪，Secrets Manager 會請求新的資料金鑰來加密新的秘密值。

RotateSecret 操作不會呼叫 GenerateDataKey，因為它不會變更秘密值。不過，如果 RotateSecret 調用的 Lambda 函數變更了秘密值，其對 PutSecretValue 操作的呼叫會觸發 GenerateDataKey 請求。

解密

Secrets Manager 會呼叫解密操作來回應以下 Secrets Manager 操作。

- GetSecretValue 與 BatchGetSecretValue– Secrets Manager 在將秘密值傳回給呼叫者之前將其解密。若要解密加密的秘密值，Secrets Manager 會呼叫 AWS KMS 解密操作來解密秘密中的加密資料金鑰。接著，使用純文字資料金鑰來解密加密的秘密值。對於批次命令，Secrets Manager 可以重複使用解密的金鑰，因此並非所有呼叫都會產生 Decrypt 要求。
- PutSecretValue 和 UpdateSecret – 大多數 PutSecretValue 和 UpdateSecret 請求不會觸發 Decrypt 操作。不過，當 PutSecretValue 或 UpdateSecret 請求嘗試變更現有秘密版本中的秘密值時，Secrets Manager 會解密現有的秘密值並將其與請求中的秘密值進行比較，確認它們相同。這個動作可確保 Secrets Manager 操作為等冪操作。若要解密加密的秘密值，Secrets Manager 會呼叫 AWS KMS 解密操作來解密秘密中的加密資料金鑰。接著，使用純文字資料金鑰來解密加密的秘密值。
- ReplicateSecretToRegions –Secrets Manager 會先在主要區域解密秘密值，然後再使用複本區域中的新 KMS 金鑰重新加密秘密值。

加密

Secrets Manager 會呼叫 Encrypt 操作，回應以下 Secrets Manager 操作：

- UpdateSecret – 如果您變更 KMS 金鑰，Secrets Manager 會使用新金鑰，將保護 AWSCURRENT、AWSPREVIOUS 和 AWSPENDING 秘密版本的資料金鑰重新加密。

DescribeKey

Secrets Manager 會呼叫 DescribeKey 操作，決定您在 Secrets Manager 主控台建立或編輯密碼時，系統是否列出 KMS 金鑰。

驗證對 KMS 金鑰的存取

在建立或變更與秘密關聯的 KMS 金鑰後，Secrets Manager 會使用指定的 CMK 來呼叫 GenerateDataKey 和 Decrypt 操作。這些呼叫會確認呼叫者擁有使用 KMS 金鑰進行這些操作的許可。Secrets Manager 會捨棄這些操作的結果，不會在任何密碼編譯操作中使用它們。

您可以識別這些驗證呼叫，因為這些請求中 SecretVersionId 金鑰加密內容的值是 RequestToValidateKeyAccess。

> **Note**
>
> 在過去，Secrets Manager 驗證呼叫不包含加密內容。您可能會在較舊的 AWS CloudTrail 日誌中找到沒有加密內容的呼叫。

AWS 受管金鑰 (aws/secretsmanager) 的金鑰政策

適用於 AWS 受管金鑰 Secrets Manager 的 金鑰政策 (aws/secretsmanager) 僅在 Secrets Manager 代表使用者提出請求時，才授予使用者將 KMS 金鑰用於指定操作的許可。金鑰政策不允許任何使用者直接使用 KMS 金鑰。

此金鑰政策與所有 AWS 受管金鑰 的政策一樣，都是由服務建立。您無法變更金鑰政策，但可以隨時進行檢視。如需詳細資訊，請參閱檢視金鑰政策。

金鑰政策中的政策陳述式具有下列效果：

- 只在請求來自代表使用者的 Secrets Manager 時，才允許帳戶中的使用者將 KMS 金鑰用於密碼編譯操作。kms:ViaService 條件金鑰會強制實施此限制。
- 允許 AWS 帳戶建立 IAM 政策，允許使用者檢視 KMS 金鑰屬性並撤銷授予。
- 雖然 Secrets Manager 不會使用授權來取得 KMS 金鑰的存取權，但政策也允許 Secrets Manager 代表使用者為 KMS 金鑰建立授權，並允許帳戶撤銷任何授權 (該授權允許 Secrets Manager 使用 KMS 金鑰)。這些是 政策文件的標準元素 AWS 受管金鑰。

以下是 Secrets Manager 範例 AWS 受管金鑰 的金鑰政策。

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 加密內容

加密內容是一組金鑰/值對，其中包含任意的非秘密資料。當您在加密資料的請求中包含加密內容時，會以 AWS KMS 加密方式將加密內容繫結至加密的資料。若要解密資料，您必須傳遞相同的加密內容。

在其 GenerateDataKey 和 Decrypt 請求中 AWS KMS，Secrets Manager 會使用兩個名稱/值對的加密內容來識別秘密及其版本，如下列範例所示。名稱不會改變，但組合的加密內容值對於每個秘密值都是不同的。

```
"encryptionContext": {
    "SecretARN": "arn:aws:secretsmanager:us-east-2:111122223333:secret:test-secret-
a1b2c3",
    "SecretVersionId": "EXAMPLE1-90ab-cdef-fedc-ba987SECRET1"
}
```

您可以使用加密內容來識別稽核紀錄和日誌 (例如 AWS CloudTrail 和 Amazon CloudWatch Logs) 中的這些密碼編譯操作，以及在政策和授權中作為授權的條件。

Secrets Manager 加密內容包含兩個名稱值對。

- SecretARN – 第一個名稱值對會識別秘密。金鑰為 `SecretARN`。值是秘密的 Amazon Resource Name (ARN)。

  ```
  "SecretARN": "ARN of an Secrets Manager secret"
  ```

  例如，如果秘密的 ARN 是 `arn:aws:secretsmanager:us-east-2:111122223333:secret:test-secret-a1b2c3`，加密內容會包含下列對組。

```
"SecretARN": "arn:aws:secretsmanager:us-east-2:111122223333:secret:test-secret-
a1b2c3"
```

- SecretVersionId – 第二個名稱值對會識別秘密的版本。金鑰為 SecretVersionId。值為版本 ID。

```
"SecretVersionId": "<version-id>"
```

例如，如果秘密的版本 ID 是 EXAMPLE1-90ab-cdef-fedc-ba987SECRET1，加密內容會包含下列對組。

```
"SecretVersionId": "EXAMPLE1-90ab-cdef-fedc-ba987SECRET1"
```

當您建立或變更秘密的 KMS 金鑰時，Secrets Manager 會將 GenerateDataKey 和 Decrypt 請求傳送至 AWS KMS ，以驗證發起人是否具有對這些操作使用 KMS 金鑰的許可。它會捨棄回應；它不會將回應用於秘密值。

在這些驗證請求中，SecretARN 的值是秘密的實際 ARN，但 SecretVersionId 值是 RequestToValidateKeyAccess，如以下範例加密內容所示。這個特殊值可協助您識別日誌和稽核線索中的驗證請求。

```
"encryptionContext": {
    "SecretARN": "arn:aws:secretsmanager:us-east-2:111122223333:secret:test-secret-
a1b2c3",
    "SecretVersionId": "RequestToValidateKeyAccess"
}
```

> **Note**
>
> 在過去，Secrets Manager 驗證請求不包含加密內容。您可能會在較舊的 AWS CloudTrail 日誌中找到沒有加密內容的呼叫。

監控 Secrets Manager 與 的互動 AWS KMS

您可以使用 AWS CloudTrail 和 Amazon CloudWatch Logs 來追蹤 Secrets Manager AWS KMS 代表您傳送到 的請求。如需監控秘密使用情況的詳細資訊，請參閱 監控秘密。

GenerateDataKey

當您在秘密中建立或變更秘密值時，Secrets Manager 會將 GenerateDataKey 請求傳送至 AWS KMS ，以指定秘密的 KMS 金鑰。

記錄 GenerateDataKey 操作的事件類似於以下範例事件。請求由 secretsmanager.amazonaws.com 呼叫。參數包括秘密 KMS 金鑰的 Amazon Resource Name (ARN)、需要 256 位元金鑰的金鑰指標，以及識別秘密和版本的加密內容。

```
{
    "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"
}
```

解密

當您取得或變更秘密的秘密值時，Secrets Manager 會將解密請求傳送至 ， AWS KMS 以解密加密的資料金鑰。對於批次命令，Secrets Manager 可以重複使用解密的金鑰，因此並非所有呼叫都會產生 Decrypt 要求。

記錄 Decrypt 操作的事件類似於以下範例事件。使用者是您 AWS 帳戶中存取資料表的委託人。參數包括加密的資料表金鑰 （做為加密文字 Blob)，以及識別資料表和 AWS 帳戶的加密內容。 會從加密文字 AWS KMS 衍生 KMS 金鑰的 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"
}
```

加密

當您變更與秘密相關聯的 KMS 金鑰時，Secrets Manager 會傳送加密請求至 AWS KMS ，以使用新金鑰重新加密 AWSCURRENT、 AWSPREVIOUS和 AWSPENDING秘密版本。當您將秘密複寫到其他區域時，Secrets Manager 也會傳送 Encrypt 請求給 AWS KMS。

記錄 Encrypt 操作的事件類似於以下範例事件。使用者是您 AWS 帳戶中存取資料表的委託人。

```
{
    "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"
}
```

中的基礎設施安全性 AWS Secrets Manager

作為受管服務， AWS Secrets Manager 受到 AWS 全球網路安全的保護。如需 AWS 安全服務以及如何 AWS 保護基礎設施的相關資訊，請參閱AWS 雲端安全。若要使用基礎設施安全的最佳實務設計您的 AWS 環境，請參閱 Security Pillar AWS Well-Architected Framework 中的基礎設施保護。

透過網路存取 Secrets Manager 是透過使用 TLS 的AWS 已發佈 API 來進行。您可以從任何網路位置來呼叫 Secrets Manager API。但是，Secrets Manager 所支援的以資源為基礎存取政策可能包含與來源 IP 地址相關的限制。您也可以使用 Secrets Manager 資源政策來控制從特定虛擬私有雲端 (VPC) 端點或特定 VPC 存取機密。實際上，這只會隔離網路中特定 VPC 對指定秘密 AWS 的網路存取。如需詳細資訊，請參閱the section called “VPC 端點 (AWS PrivateLink)”。

使用 AWS Secrets Manager VPC 端點

我們建議您在無法從公有網際網路存取的私有網路上儘可能執行基礎設施。您可以建立介面 VPC 端點，以在您的 VPC 與 Secrets Manager 之間建立私有連線。介面端點採用 AWS PrivateLink技術，可讓您在沒有網際網路閘道、NAT 裝置、VPN 連線或 AWS Direct Connect 連線的情況下，私下存取 Secrets Manager APIs。VPC 中的執行個體不需要公有 IP 地址，即能與 Secrets Manager API 通訊。您的 VPC 與 Secrets Manager 之間的流量都會保持在 AWS 網路的範圍內。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的介面 VPC 端點 (AWS PrivateLink)。

Secrets Manager 使用 Lambda 輪換函數輪換秘密時，例如包含資料庫憑證的秘密，Lambda 函數會同時向資料庫和 Secrets Manager 發出請求。在您使用主控台開啟自動輪換後，Secrets Manager 將在與資料庫相同的 VPC 中建立 Lambda 函數。我們建議您在相同的 VPC 中建立 Secrets Manager 端點，以便從 Lambda 輪換函數到 Secrets Manager 的請求保持在 Amazon 網路的範圍內。

如果您為該端點啟用私有 DNS，您可以使用其區域的預設 DNS 名稱 (例如 `secretsmanager.us-east-1.amazonaws.com`)，向 Secrets Manager 發出 API 請求。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的透過介面端點存取服務。

您可以透過在許可政策中包含條件，確保對 Secrets Manager 的請求都來自 VPC 存取。如需詳細資訊，請參閱the section called “範例：許可和 VPC”。

您可以使用 AWS CloudTrail 日誌，透過 VPC 端點稽核秘密的使用。

為 Secrets Manager 建立 VPC 端點

1. 請參閱《Amazon VPC 使用者指南》中的建立介面端點。使用服務名稱：`com.amazonaws.region.secretsmanager`。

2. 若要控制對端點的存取，請參閱使用端點政策控制對 VPC 端點的存取。
3. 若要使用 IPv6 和雙堆疊定址，請參閱 IPv4 和 IPv6 存取。

共用子網路

無法在與您共用的子網路中建立、描述、修改或刪除 VPC 端點。不過，可以在與您共用的子網路中使用 VPC 端點。如需 VPC 共享的相關資訊，請參閱《Amazon Virtual Private Cloud 使用者指南》中的與其他帳戶共享 VPC。

使用 IAM 政策控制 API 存取

如果您使用 IAM 政策 AWS 服務 根據 IP 地址控制對 的存取，您可能需要更新政策以包含 IPv6 地址範圍。本指南說明 IPv4 和 IPv6 之間的差異，並說明如何更新 IAM 政策以支援這兩個通訊協定。實作這些變更可協助您在支援 IPv6 的同時維護對 AWS 資源的安全存取。

什麼是 IPv6？

IPv6 是新一代 IP 標準，旨在最終取代 IPv4。舊版 IPv4 使用 32 位元定址機制來支援 43 億部裝置。IPv6 改為使用 128 位元定址來支援大約 340 兆兆億（或 2 到第 128 個電源）的裝置。

如需詳細資訊，請參閱 VPC IPv6 網頁。

以下是 IPv6 地址的範例：

```
2001:cdba:0000:0000:0000:0000:3257:9652 # This is a full, unabbreviated IPv6 address.
2001:cdba:0:0:0:3257:9652               # The same address with leading zeros in each
 group omitted
2001:cdba::3257:965                     # A compressed version of the same address.
```

IAM 雙堆疊 (IPv4 和 IPv6) 政策

您可以使用 IAM 政策來控制對 Secrets Manager APIs存取，並防止設定範圍以外的 IP 地址存取 Secrets Manager APIs。

Secrets Manager API 的 Secretsmanager.{region}.amazonaws.com 雙堆疊端點支援 IPv6 和 IPv4。APIs

如果您需要同時支援 IPv4 和 IPv6，請更新 IP 地址篩選政策以處理 IPv6 地址。否則，您可能無法透過 IPv6 連線至 Secrets Manager。

誰應該進行此變更？

如果您使用雙定址搭配包含 的政策，此變更會影響您aws:sourceIp。雙重定址表示網路同時支援 IPv4 和 IPv6。

如果您使用雙定址，請更新目前使用 IPv4 格式地址的 IAM 政策，以包含 IPv6 格式地址。

誰不應進行此變更？

如果您只使用 IPv4 網路，此變更不會影響您。

將 IPv6 新增至 IAM 政策

IAM 政策使用 aws:SourceIp條件金鑰來控制來自特定 IP 地址的存取。如果您的網路使用雙定址 (IPv4 和 IPv6)，請更新您的 IAM 政策以包含 IPv6 地址範圍。

在政策的 Condition元素中，針對 IP 地址條件使用 IpAddress和 NotIpAddress運算子。請勿使用字串運算子，因為它們無法處理各種有效的 IPv6 地址格式。

這些範例使用 aws:SourceIp。對於 VPCs，請aws:VpcSourceIp改用 。

以下是 IAM 使用者指南中的 AWS 根據來源 IP 參考政策拒絕對 的存取。Condition 元素NotIpAddress中列出兩個 IPv4 地址範圍的 192.0.2.0/24和 203.0.113.0/24，這會被拒絕存取 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"
```

```
            }
        }
    }
}
```

若要更新此政策，請將 Condition元素變更為包含 IPv6 地址範圍2001:DB8:1234:5678::/64和2001:cdba:3257:8593::/64。

> **Note**
>
> 請勿移除現有的 IPv4 地址。需要它們才能回溯相容。

```
"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"
                }
            }
```

若要更新 VPC 的此政策，請使用 aws:VpcSourceIp而非 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"
                }
```

```
        }
```

驗證您的用戶端支援 IPv6

如果您使用 secretsmanager.{region}.amazonaws.com 端點，請確認您可以連線到它。下列步驟說明如何執行驗證。

此範例使用 Linux 和 curl 8.6.0 版，並使用 服務，該AWS Secrets Manager 服務具有位於 amazonaws.com 端點的已啟用 IPv6 的端點。

> **Note**
>
> secretsmanager.{region}.amazonaws.com 與典型的雙堆疊命名慣例https://docs.aws.amazon.com/general/latest/gr/rande.html#dual-stack-endpoints不同。如需 Secrets Manager 端點的完整清單，請參閱 AWS Secrets Manager 端點。
> 將 AWS 區域 變更為您的服務所在的相同區域。在此範例中，我們使用美國東部 （維吉尼亞北部） – us-east-1端點。

1. 使用以下dig命令判斷端點是否以 IPv6 地址解析。

```
$ dig +short AAAA secretsmanager.us-east-1.amazonaws.com

> 2600:1f18:e2f:4e05:1a8a:948e:7c08:c1c3
```

2. 使用以下curl命令判斷用戶端網路是否可以進行 IPv6 連線。404 回應代碼表示連線成功，而 0 回應代碼表示連線失敗。

```
$ 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
```

如果已識別遠端 IP，且回應代碼不是 0，則已成功使用 IPv6 與端點建立網路連線。遠端 IP 應該是 IPv6 地址，因為作業系統應該選取對用戶端有效的通訊協定。

如果遠端 IP 為空白或回應代碼為 0，則用戶端網路或端點的網路路徑為IPv4-only。您可以使用下列curl命令來驗證此組態。

```
$ 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
```

中的彈性 AWS Secrets Manager

AWS 會在 AWS 區域 和可用區域周圍建置全球基礎設施。 AWS 區域 提供多個實體隔離且隔離的可用區域，可連接低延遲、高輸送量和高度備援的網路。透過可用區域，您可以設計與操作的應用程式和資料庫，在可用區域之間自動容錯移轉而不會發生中斷。可用區域可讓您更有可用性、容錯能力和擴充能力，較單一或多個資料中心的傳統基礎設施還高。

如需彈性和災難復原的詳細資訊，請參閱可靠性支柱 - AWS Well-Architected Framework。

如需 AWS 區域 和可用區域的詳細資訊，請參閱 AWS 全球基礎設施。

後量子 TLS

Secrets Manager 支援適用於 Transport Layer Security (TLS) 網路加密通訊協定的混合式後量子金鑰交換選項。連線至 Secrets Manager API 端點時，您可使用此 TLS 選項。在後量子演算法標準化前，我們會提供此功能，以便您可以開始測試這些金鑰交換通訊協定對於 Secrets Manager 呼叫的影響。這些選用的混合式後量子金鑰交換功能至少與現今使用的 TLS 加密功能同樣安全，且還能提供其他安全優勢。不過，與現今使用的傳統金鑰交換通訊協定相較之下，這些功能會影響延遲和輸送量。Secrets Manager Agent 預設會使用後量子 ML-KEM 金鑰交換作為最高優先順序金鑰交換。

為了保護今天加密的資料免受潛在的未來攻擊， AWS 正與密碼編譯社群參與開發量子抗性或後量子演算法。我們已在 Secrets Manager 端點中實作混合式後量子金鑰交換密碼套件。這些混合式密碼套件結合傳統與後量子元素，能夠確保您的 TLS 連線至少與使用傳統密碼套件一樣堅強。然而，由於混合式密碼套件的效能特性和頻寬要求不同於傳統金鑰交換機制，我們建議您對 API 呼叫測試這些套件。

Secrets Manager 支援所有區域中的 PQTLS，唯中國地區除外。

設定混合式後量子 TLS

1. 將 AWS Common Runtime 用戶端新增至 Maven 相依性。我們建議使用最新的可用版本。例如，此陳述式將新增 2.20.0 版。

   ```
   <dependency>
   ```

```
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>aws-crt-client</artifactId>
  <version>2.20.0</version>
</dependency>
```

2. 將適用於 Java 的 AWS SDK 2.x 新增至您的專案並初始化它。在 HTTP 用戶端上啟用混合式後量子密碼套件。

```
SdkAsyncHttpClient awsCrtHttpClient = AwsCrtAsyncHttpClient.builder()
            .postQuantumTlsEnabled(true)
            .build();
```

3. 建立 Secrets Manager 非同步用戶端。

```
SecretsManagerAsyncClient SecretsManagerAsync = SecretsManagerAsyncClient.builder()
            .httpClient(awsCrtHttpClient)
            .build();
```

現在，當您呼叫 Secrets Manager API 操作時，系統會使用混合式後量子 TLS 將您的呼叫傳輸至 Secrets Manager 端點。

如需有關使用混合式後量子 TLS 的詳細資訊，請參閱：

- AWS SDK for Java 2.x 開發人員指南和AWS SDK for Java 2.x 發佈的部落格文章。
- s2n-tls 簡介 (全新開放原始碼 TLS 實作) 和使用 s2n-tls。
- 國家標準技術研究所 (NIST) 的後量子加密法。
- 用於 Transport Layer Security 1.2 (TLS) 的混合式後量子金鑰封裝法 (PQ KEM)。

Secrets Manager 的後量子 TLS 可用於中國 AWS 區域 以外的所有 。

故障診斷 AWS Secrets Manager

使用此處的資訊，來協助您針對在使用 Secrets Manager 時可能遇到的問題進行診斷與修復。

如需有關輪換的問題，請參閱 the section called “輪換疑難排解”。

主題

- 「拒絕存取」訊息
- 暫時安全憑證的「存取遭拒」
- 我所做的變更不一定都會立即顯示。
- 在建立秘密時收到「無法使用非對稱 KMS 金鑰產生資料金鑰」的訊息
- AWS CLI 或 AWS SDK 操作無法從部分 ARN 找到我的秘密
- 此秘密由 AWS 服務管理，您必須使用該服務來更新它。
- 使用 時，Python 模組匯入失敗 Transform: AWS::SecretsManager-2024-09-16

「拒絕存取」訊息

當您對 Secrets Manager 進行 GetSecretValue 或 CreateSecret 等 API 呼叫時，您必須擁有 IAM 許可才能進行該呼叫。當您使用 主控台時，主控台會代表您進行相同的 API 呼叫，因此您也必須擁有 IAM 許可。管理員可以透過將 IAM 政策連接至您的 IAM 使用者或您所屬的群組來授予許可。如果授予這些許可的政策內容中包含了任何條件，例如時刻或 IP 地址的限制，則當您傳送請求時，也必須滿足這些要求。關於檢視或修改 IAM 使用者、群組或角色的政策方面的相關資訊，請參閱 IAM 使用者指南中的政策的使用。如需有關 Secrets Manager 所需許可的詳細資訊，請參閱 the section called “身分驗證與存取控制”。

如果您是手動簽署 API 請求，而沒有使用 AWS 開發套件，請確認已正確簽署請求。

暫時安全憑證的「存取遭拒」

確認用來提出請求的 IAM 使用者或角色擁有正確許可。臨時安全登入資料的許可衍生自 IAM 使用者或角色。這表示能提供的許可僅限於授予 IAM 使用者或角色的許可。關於臨時安全登入資料許可的決定方式，詳細資訊請參閱 IAM 使用者指南中的控管臨時安全登入資料的許可。

確認您的請求已正確簽署且請求的格式也正確。如需詳細資訊，請參閱所選 SDK 的工具組文件，或《IAM 使用者指南》中的使用暫時安全登入資料請求存取 AWS 資源。

確認您的臨時安全憑證並未過期。如需詳細資訊，請參閱 IAM 使用者指南中的請求臨時安全憑證。

如需有關 Secrets Manager 所需許可的詳細資訊，請參閱 the section called “身分驗證與存取控制”。

我所做的變更不一定都會立即顯示。

Secrets Manager 使用稱為最終一致性的分散式運算模型。您在 Secrets Manager （或其他 AWS 服務）中所做的任何變更，都需要一些時間才能從所有可能的端點中顯示。部分延遲是由在伺服器之間、複寫區域之間和全球不同地區之間傳送資料所花費的時間造成。Secrets Manager 也會使用快取以提升效能，但在某些情況下，這可能會增加時間。直到先前快取的資料逾時後，才能看到變更。

設計您的全球應用程式以說明這些潛在的延遲。此外，確保它們如預期般運作，即使在某個位置所做的變更也不會立即顯示在另一個位置。

如需其他一些 AWS 服務如何受到最終一致性影響的詳細資訊，請參閱：

- Amazon Redshift 資料庫開發人員指南中的管理資料一致性
- Amazon Simple Storage Service 使用者指南中的 Amazon S3 資料一致性模式
- AWS 大數據部落格中的在使用 Amazon S3 和適用於 ETL 工作流程的 Amazon EMR 時確保一致性
- Amazon EC2 API 參考中的 Amazon EC2 最終一致性

在建立秘密時收到「無法使用非對稱 KMS 金鑰產生資料金鑰」的訊息

Secrets Manager 使用與秘密相關聯的對稱加密 KMS 金鑰，來為每個秘密值產生資料金鑰。您無法使用非對稱 KMS 金鑰。請確認您使用的是對稱加密 KMS 金鑰，而不是非對稱 KMS 金鑰。如需說明，請參閱標識非對稱 KMS 金鑰。

AWS CLI 或 AWS SDK 操作無法從部分 ARN 找到我的秘密

在許多情況下，Secrets Manager 可以從 ARN 的一部分而非完整的 ARN 中找到您的秘密。但是，如果秘密名稱以連字號結尾，且後面接著六個字元，Secrets Manager 可能無法僅從部分 ARN 中找到秘密。建議您改為使用完整的 ARN 或機密名稱。

更多詳細資訊

Secrets Manager 會在秘密名稱末尾包含六個隨機字元，協助確保秘密 ARN 是唯一。如果刪除原始秘密，然後使用相同的名稱建立新秘密，則這兩個秘密會因為這些字元而具有不同的 ARN。具有舊秘密存取權的使用者不會自動取得新秘密的存取權，因為 ARN 不同。

Secrets Manager 會為秘密建構一個 ARN，它包含區域、帳戶、秘密名稱、一個連字符和六個字元，如下所示：

```
arn:aws:secretsmanager:us-east-2:111122223333:secret:SecretName-abcdef
```

如果您的秘密名稱以連字符和六個字元結尾，則只有部分 ARN 會出現在 Secrets Manager 中，就好像您指定了一個完整的 ARN 一樣。例如，您可能擁有 ARN 為 `MySecret-abcdef` 的秘密。

```
arn:aws:secretsmanager:us-east-2:111122223333:secret:MySecret-abcdef-nutBrk
```

如果您呼叫以下操作，它只使用秘密 ARN 的一部分，則 Secrets Manager 可能找不到秘密。

```
$ aws secretsmanager describe-secret --secret-id arn:aws:secretsmanager:us-
east-2:111122223333:secret:MySecret-abcdef
```

此秘密由 AWS 服務管理，您必須使用該服務來更新它。

如果您在嘗試修改秘密時遇到此訊息，則只能使用訊息中列出的管理服務來更新秘密。如需詳細資訊，請參閱由其他服務管理的秘密。

若要決定管理秘密的人員，您可以檢閱秘密名稱。由其他服務管理的秘密以該服務的 ID 為字首。或者，在 中 AWS CLI呼叫 describe-secret，然後檢閱 欄位 `OwningService`。

使用 時，Python 模組匯入失敗 `Transform: AWS::SecretsManager-2024-09-16`

如果您使用的是 Transform： ，`AWS::SecretsManager-2024-09-16`並在輪換 Lambda 函數執行時遇到 Python 模組匯入失敗，問題可能是由不相容`Runtime`的值造成。在此轉換版本中， 會為您 AWS CloudFormation 管理執行時間版本、程式碼和共用物件檔案。您不需要自行管理這些項目。

AWS Secrets Manager 配額

Secrets Manager 已讀取的 API 擁有高 TPS 配額，而較少調用的控制平面 API 則擁有較低的 TPS 配額。建議您避免以超過每 10 分鐘一次的持續速率呼叫 `PutSecretValue` 或 `UpdateSecret`。在呼叫 `PutSecretValue` 和 `UpdateSecret` 更新機密值時，機密管理員會建立機密的新版本。當版本超過 100 個時，Secrets Manager 會移除未標記的版本，但不會移除建立時間不到 24 小時的版本。如果以超過每 10 分鐘一次的速率更新機密值，您建立的版本比機密管理員移除的版本更多，且您將達到機密版本的配額。

您可以在帳戶中經營多個區域，而且每個區域都有專屬的特定配額。

當一個 中的應用程式 AWS 帳戶 使用不同帳戶所擁有的秘密時，稱為跨帳戶請求。對於跨帳戶請求，機密管理員會調節發出請求的帳戶身分，而不是擁有機密的帳戶。例如，如果帳戶 A 中的身分使用帳戶 B 中的機密，使用該機密只會套用帳戶 A 中的配額。

Secrets Manager 配額

名稱	預設	可調整	描述
DeleteResourcePolicy、GetResourcePolicy、PutResourcePolicy 和 ValidateResourcePolicy 請求的合併速率	每個支援的區域：每秒 50 個	否	DeleteResourcePolicy、GetResourcePolicy、PutResourcePolicy 和 ValidateResourcePolicy API 合併請求的每秒交易數量上限。
PutSecretValue、RemoveRegionsFromReplication、ReplicateSecretToRegion、StopReplicationToReplica、UpdateSecret 和 UpdateSecretVersionStage API 請求的合併速率	每個支援的區域：每秒 50 個	否	PutSecretValue、RemoveRegionsFromReplication、ReplicateSecretToRegion、StopReplicationToReplica、UpdateSecret 和 UpdateSecretVersionStage API 合併請求的每秒交易數量上限。

名稱	預設	可調整	描述
RestoreSecret API 請求的合併速率	每個支援的區域：每秒 50 個	否	RestoreSecret API 請求的每秒交易數量上限。
RotateSecret 和 CancelRotateSecret API 請求的合併速率	每個支援的區域：每秒 50 個	否	RotateSecret 和 CancelRotateSecret API 合併請求的每秒交易數量上限。
TagResource 和 UntagResource API 請求的合併速率	每個支援的區域：每秒 50 個	否	TagResource 和 UntagResource API 合併請求的每秒交易數量上限。
BatchGetSecretValue API 請求的速率	每個支援的區域：每秒 100	否	BatchGetSecretValue API 請求的每秒交易數量上限。
CreateSecret API 請求的速率	每個支援的區域：每秒 50 個	否	CreateSecret API 請求的每秒交易數量上限。
DeleteSecret API 請求的速率	每個支援的區域：每秒 50 個	否	DeleteSecret API 請求的每秒交易數量上限。
DescribeSecret API 請求的速率	每個支援的區域：每秒 40，000 個	否	DescribeSecret API 請求的每秒交易數上限。
GetRandomPassword API 請求的速率	每個支援的區域：每秒 50 個	否	GetRandomPassword API 請求的每秒交易數量上限。
GetSecretValue API 請求的速率	每個支援的區域：每秒 1 萬個	否	GetSecretValue API 請求的每秒交易數上限。

名稱	預設	可調整	描述
ListSecretVersionIds API 請求的速率	每個支援的區域：每秒 50 個	否	ListSecretVersionIds API 請求的每秒交易數量上限。
ListSecrets API 請求的速率	每個支援的區域：每秒 100	否	ListSecrets API 請求的每秒交易數量上限。
以資源為基礎的政策長度	每個受支援的區域：20,480 個	否	連接至機密之以資源為基礎的許可政策中每秒交易數量上限。
機密值大小	每個受支援的區域：65,536 個位元組	否	加密之機密值的大小上限。如果機密值是字串，則這是機密值中允許的字元數。
秘密	每個受支援的區域：50 萬個	否	此 AWS 帳戶每個 AWS 區域中的秘密數量上限。
機密的所有版本附加的預備標籤	每個受支援的區域：20	否	機密的所有版本連接之預備標籤的數量上限。
每個機密的版本數	每個受支援的區域：100	否	機密的版本數上限。

將重試新增至應用程式

由於 AWS 用戶端發生意外問題，您的用戶端可能會看到呼叫 Secrets Manager 失敗。或者呼叫可能因機密管理員的速率限制而失敗。當您超過 API 請求配額時，機密管理員會調節請求。它會拒絕其他有效的請求，並傳回 throttling 錯誤。對於這兩種類型的失敗，我們建議您經過短暫的等待時間後重試呼叫。這就是所謂的退避和重試策略。

如果遇到下列錯誤，您可能會想要將重試新增到應用程式碼：

暫時性錯誤和例外狀況

- RequestTimeout
- RequestTimeoutException
- PriorRequestNotComplete
- ConnectionError
- HTTPClientError

服務端調節和限制錯誤與例外狀況

- Throttling
- ThrottlingException
- ThrottledException
- RequestThrottledException
- TooManyRequestsException
- ProvisionedThroughputExceededException
- TransactionInProgressException
- RequestLimitExceeded
- BandwidthLimitExceeded
- LimitExceededException
- RequestThrottled
- SlowDown

如需重試、指數退避和抖動的詳細資訊以及範例程式碼，請參閱下列資源：

- 指數退避和抖動
- 逾時、重試和退避 (具有抖動)
- 錯誤重試和指數退避。 AWS

文件歷史紀錄

下表說明文件自上次發行以來的重要變更 AWS Secrets Manager。如需有關此文件更新的通知，您可以訂閱 RSS 訂閱源。

變更	描述	日期
新增對成本分配標籤的支援	Secrets Manager 現在支援成本分配標籤，讓客戶能夠依部門、團隊或應用程式來分類和追蹤成本。如需詳細資訊，請參閱搭配 使用成本分配標籤 AWS Secrets Manager。	2025 年 5 月 27 日
新增 IPv6 和雙堆疊支援	Secrets Manager 現在支援雙堆疊端點。如需詳細資訊，請參閱 IPv4 和 IPv6 存取。	2024 年 12 月 20 日
Secrets Manager 變更為 AWS 受管政策	`SecretsManagerReadWrite` 受管政策現在包含 `redshift-serverless` 許可。如需詳細資訊，請參閱 AWS 的 受管政策 AWS Secrets Manager	2024 年 3 月 12 日

舊版更新

下表說明 AWS Secrets Manager 使用者指南 2024 年 2 月之前每個版本的重要變更。

變更	描述	日期
一般可用性	這是 Secrets Manager 的初始公開版本。	2018 年 4 月 4 日

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。