Guide de l’utilisateur

AWS Secrets Manager

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

AWS Secrets Manager: Guide de l’utilisateur

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

Les marques commerciales et la présentation commerciale d’Amazon ne peuvent pas être utilisées en relation avec un produit ou un service extérieur à Amazon, d’une manière susceptible d’entraîner une confusion chez les clients, ou d’une manière qui dénigre ou discrédite Amazon. Toutes les autres marques commerciales qui ne sont pas la propriété d’Amazon appartiennent à leurs propriétaires respectifs, qui peuvent ou non être affiliés ou connectés à Amazon, ou sponsorisés par Amazon.

Table of Contents

Qu'est-ce que Secrets Manager ? 1
Démarrer avec Secrets Manager 1
Conformité aux normes 2
Tarification 2
Accès à Secrets Manager 4
Console Secrets Manager 4
Outils de ligne de commande 4
AWS SDKs 5
API de requête HTTPS 5
Points de terminaison Secrets Manager 6
Bonnes pratiques 11
Stockez les informations d'identification et autres informations sensibles dans AWS Secrets Manager 11
Trouvez des secrets non protégés dans votre code 12
Choisissez une clé de chiffrement pour votre secret 12
Utiliser la mise en cache pour récupérer des secrets 12
Rotation de vos secrets 13
Atténuer les risques liés à l'utilisation de la CLI 13
Limitez l'accès aux secrets 13
Condition `BlockPublicPolicy` 14
Faites preuve de prudence en ce qui concerne les conditions d'adresse IP dans les politiques 15
Limitez les demandes en fonction des conditions du point de terminaison VPC 15
Répliquez les secrets 16
Surveillance des secrets 16
Gérez votre infrastructure sur des réseaux privés 16
Didacticiels 17
CodeGuru Réviseur Amazon 17
Remplacer les secrets codés en dur 17
Étape 1 : créer le secret 18
Étape 2 : mettre à jour votre code 20
Étape 3 : mettre à jour le secret 21
Étapes suivantes 21
Remplacer les informations d'identification de bases de données codées en dur 22

Étape 1 : créer le secret 23
Étape 2 : mettre à jour votre code 24
Étape 3 : effectuer la rotation du secret 25
Étapes suivantes 26
Rotation des utilisateurs en alternance 27
Autorisations 28
Prérequis 28
Étape 1 : créer un utilisateur de base de données Amazon RDS 32
Étape 2 : créez un secret pour les informations d'identification de l'utilisateur 35
Étape 3 : Test de rotation du secret 36
Étape 4 : Nettoyer les ressources 37
Étapes suivantes 38
Rotation utilisateur unique 38
Autorisations 38
Prérequis 39
Étape 1 : créer un utilisateur de base de données Amazon RDS 39
Étape 2 : créez un secret pour les informations d'identification de l'utilisateur 40
Étape 3 : tester le mot de passe ayant subi la rotation 41
Étape 4 : Nettoyer les ressources 42
Étapes suivantes 42
Création de secrets 43
AWS CLI 46
AWS SDK 47
Qu'y a-t-il dans un secret 47
Métadonnées 47
Versions secrètes 48
Structure JSON d'un secret 50
Informations d'identification Amazon RDS et Aurora 51
Informations d'identification Amazon Redshift 53
Informations d'identification Amazon Redshift Serverless 54
Informations d'identification Amazon DocumentDB 54
Structure secrète d'Amazon Timestream pour InfluxDB 55
ElastiCache Informations d'identification Amazon 55
Informations d'identification Active Directory 55
Gérez les secrets 58
Mise à jour d'une valeur de secret 58

AWS CLI 59
AWS SDK 59
Générez un mot de passe avec Secrets Manager 60
Rétablirez un secret à une version précédente 60
Modification de clé de chiffrement d'un secret 60
AWS CLI 62
Modification d'un secret 63
AWS CLI 64
AWS SDK 65
Recherche de secrets 65
Filtres de recherche 65
AWS CLI 67
AWS SDK 67
Suppression d'un secret 67
AWS CLI 69
AWS SDK 70
Restaurer un secret 70
AWS CLI 71
AWS SDK 71
Étiqueter les secrets 72
Bases des balises 72
Suivi des coûts à l'aide d'étiquettes 73
Comprendre les restrictions liées aux balises 73
Marquer des secrets dans la console 74
AWS CLI 75
« Hello, World! » 76
SDK 76
Réplication multirégionale 78
AWS CLI 80
AWS SDK 80
Promouvoir un secret de réplica en un secret autonome 80
AWS CLI 81
AWS SDK 82
Empêcher la réplication 82
Résoudre les problèmes de réplication 83
Un secret avec le même nom existe dans la région sélectionnée 84

Aucune autorisation disponible sur la clé KMS pour terminer la réplication 84
La clé KMS est désactivée ou introuvable 84
Vous n'avez pas activé la région dans laquelle la réplication se produit 85
Obtenez des secrets 86
Java 86
Java avec mise en cache côté client 87
Connexion JDBC avec informations d'identification dans un secret 94
AWS SDK Java 104
Python 106
Python avec mise en cache côté client 106
AWS SDK Python 112
Obtenez un lot de valeurs secrètes 114
.NET 116
.NET avec mise en cache côté client 116
SDK pour .NET 123
Go 126
Optez pour la mise en cache côté client 126
Go AWS SDK 130
Rust 132
Rust avec mise en cache côté client 132
Rust 135
Amazon EKS 135
ASCP avec rôles IAM pour les comptes de service (IRSA) 136
ASCP avec Pod Identity 136
Choix de la bonne approche 136
Installation d'ASCP pour Amazon EKS 137
Intégrer ASCP à Pod Identity pour Amazon EKS 140
Intégrer ASCP à IRSA pour Amazon EKS 145
Exemples d'ASCP 147
AWS Lambda 156
Découvrez des secrets avec Lambda 156
Intégration de Parameter Store 157
Secrets Manager 157
Comment fonctionne l'agent Secrets Manager 157
Comprendre la mise en cache des agents Secrets Manager 158
Créer l'agent Secrets Manager 159

Installer l'agent Secrets Manager 163
Récupérez des secrets avec l'agent Secrets Manager 167
Comprendre le `refreshNow` paramètre 170
Options de configuration 172
Fonctionnalités optionnelles 173
Journalisation 174
Considérations sur la sécurité 175
C++ 175
JavaScript 176
Kotlin 177
PHP 178
Ruby 179
AWS CLI 180
Obtenez un groupe de secrets par lots à l'aide du AWS CLI 180
AWS console 181
AWS Batch 182
AWS CloudFormation 182
GitHub emplois 183
Prérequis 184
Utilisation 185
Désignation des variables d'environnement 186
Exemples 187
AWS IoT Greengrass 189
Parameter Store 190
Rotation des secrets 191
Rotation gérée 191
Rotation par fonction Lambda 193
Rotation automatique pour les secrets de base de données (console) 195
Rotation automatique pour les secrets non liés à la base de données (console) 199
Rotation automatique (AWS CLI) 204
Stratégies de rotation des fonctions Lambda 208
Fonctions de rotation Lambda 210
Modèles de fonctions de rotation 214
Autorisations de rotation 222
Accès au réseau pour la fonction AWS Lambda de rotation 227
Résolution des problèmes de rotation 228

Horaires de rotation 243
Fenêtres rotatives 244
Expressions de fréquence 244
Expressions Cron 245
Mettre immédiatement un secret en rotation 250
AWS CLI 250
Trouvez des secrets qui ne sont pas alternés 251
Annuler la rotation automatique 251
Secrets gérés par d'autres services 253
Services utilisant des secrets 255
App Runner 257
AWS App2Container 257
AWS AppConfig 257
Amazon AppFlow 258
AWS AppSync 258
Amazon Athena 258
Amazon Aurora 259
AWS CodeBuild 259
Amazon Data Firehose 259
AWS DataSync 260
Amazon DataZone 260
AWS Direct Connect 260
AWS Directory Service 261
Amazon DocumentDB 261
AWS Elastic Beanstalk 261
Amazon Elastic Container Registry 262
Amazon Elastic Container Service 262
Amazon ElastiCache 263
AWS Elemental Live 263
AWS Elemental MediaConnect 263
AWS Elemental MediaConvert 264
AWS Elemental MediaLive 264
AWS Elemental MediaPackage 264
AWS Elemental MediaTailor 264
Amazon EMR 265
EMR activé EC2 265

EMR sans serveur ... 265
Amazon EventBridge ... 266
Amazon FSx ... 266
AWS Glue DataBrew ... 266
AWS Glue Studio ... 267
AWS IoT SiteWise ... 267
Amazon Kendra ... 267
Amazon Kinesis Video Streams ... 267
AWS Launch Wizard ... 268
Amazon Lookout for Metrics ... 268
Amazon Managed Grafana ... 268
AWS Managed Services ... 269
Amazon Managed Streaming for Apache Kafka ... 269
Amazon Managed Workflows for Apache Airflow ... 269
AWS Marketplace ... 270
AWS Migration Hub ... 270
AWS Panorama ... 270
AWS Service de calcul parallèle ... 271
AWS ParallelCluster ... 271
Amazon Q ... 271
Amazon OpenSearch Ingestion ... 272
AWS OpsWorks for Chef Automate ... 272
Amazon QuickSight ... 272
Amazon RDS ... 273
Amazon Redshift ... 273
Éditeur de requête Amazon Redshift v2 ... 274
Amazon SageMaker AI ... 274
AWS SCT ... 275
Amazon Timestream pour InfluxDB ... 275
AWS Toolkit for JetBrains ... 276
AWS Transfer Family ... 276
AWS Wickr ... 276
AWS CloudFormation ... 278
Créer un secret ... 279
JSON ... 279
YAML ... 279

Créez un secret avec les informations d'identification Amazon RDS avec rotation automatique ... 280
Créez un secret avec les informations d'identification Amazon Redshift ... 280
Créez un secret avec les informations d'identification Amazon DocumentDB ... 280
JSON ... 281
YAML ... 285
Comment utilise Secrets Manager AWS CloudFormation ... 288
AWS CDK ... 289
Surveillance des secrets ... 290
Connectez-vous avec AWS CloudTrail ... 290
AWS CLI ... 291
CloudTrail entrées ... 291
Moniteur avec CloudWatch ... 297
CloudWatch alarmes ... 298
Associez les événements de Secrets Manager à EventBridge ... 298
Faire correspondre toutes les modifications à un secret spécifié ... 299
Faire correspondre les événements lorsqu'une valeur secrète change ... 299
Surveillance des secrets planifiés pour une suppression ... 300
Étape 1 : Configuration de la livraison du fichier CloudTrail journal à CloudWatch Logs ... 300
Étape 2 : Création de l' CloudWatchalarme ... 301
Étape 3 : Testez l' CloudWatchalarme ... 303
Surveillez les secrets à des fins de conformité ... 303
Surveillez les coûts de Secrets Manager ... 304
Détectez les menaces avec GuardDuty ... 304
Validation de conformité ... 306
Normes de conformité ... 306
Sécurité ... 309
Réduisez les risques liés AWS CLI à l'utilisation du pour stocker vos AWS Secrets Manager secrets ... 310
Authentification et contrôle d'accès ... 312
Référence des autorisations ... 313
Secrets Manager ... 313
Autorisations d'accès aux secrets ... 313
Autorisations pour les fonctions de rotation Lambda ... 313
Autorisations pour les clés de chiffrement ... 314
Autorisations pour la réplication ... 314

Politiques basées sur l'identité 314
Politiques basées sur les ressources 321
Contrôlez l'accès aux secrets à l'aide de balises 329
AWS politiques gérées 330
Déterminer qui a les autorisations pour vos secrets 335
Accès intercomptes 336
Accès sur site 339
Protection des données dans Secrets Manager 340
Chiffrement au repos 341
Chiffrement en transit 341
Confidentialité du trafic inter-réseaux 341
Gestion des clés de chiffrement 342
Chiffrement et déchiffrement de secret 342
Choisir une AWS KMS clé 343
Qu'est-ce qui est chiffré ? 344
Processus de chiffrement et déchiffrement 344
Autorisations pour la clé KMS 345
Comment Secrets Manager utilise votre clé KMS 345
Stratégie de clé de Clé gérée par AWS (`aws/secretsmanager`) 347
Contexte de chiffrement de Secrets Manager 350
Surveillez l'interaction de Secrets Manager avec AWS KMS 352
Sécurité de l'infrastructure 356
Points de terminaison d'un VPC (AWS PrivateLink) 356
Sous-réseaux partagés 357
IPv4 et IPv6 accès 358
Qu'est-ce que c'est IPv6 ? 358
Utilisation de politiques à double pile 358
Ajouter IPv6 à une politique 359
Vérifier le support de vos clients IPv6 361
Résilience 362
TLS post-quantique 362
Résolution des problèmes 365
Messages « Accès refusé » 365
« Access denied » (« Accès refusé ») pour les informations d'identification de sécurité temporaires 366
Les modifications que j'apporte ne sont pas toujours visibles immédiatement. 366

« Cannot generate a data key with an asymmetric KMS key » (« Impossible de générer une clé de données avec une clé KMS asymétrique ») lors de la création d'un secret 367
Une opération AWS CLI ou AWS SDK ne trouve pas mon secret à partir d'un ARN partiel 367
Ce secret est géré par un AWS service, et vous devez utiliser ce service pour le mettre à jour. 368
L'importation du module Python échoue lors de l'utilisation `Transform: AWS::SecretsManager-2024-09-16` 368
Quotas 370
Quotas de Secrets Manager 370
Ajouter de nouvelles tentatives à votre application 373
Historique du document 376
Mises à jour antérieures 377
..... ccclxxviii

Qu'est-ce que c'est AWS Secrets Manager ?

AWS Secrets Manager vous aide à gérer, récupérer et faire pivoter les informations d'identification des bases de données, les informations d'identification des applications, les OAuth jetons, les clés d'API et d'autres secrets tout au long de leur cycle de vie. Plusieurs AWS services utilisent et stockent des secrets dans Secrets Manager.

Secrets Manager vous aide à améliorer votre posture de sécurité, car vous n'avez plus besoin d'informations d'identification codées en dur dans le code source des applications. Le stockage des informations d'identification dans Secrets Manager permet d'éviter toute compromission possible par quiconque peut inspecter votre application ou ses composants. Vous remplacez les informations d'identification codées en dur par un appel au service Secrets Manager au moment de l'exécution pour récupérer les informations d'identification de manière dynamique lorsque vous en avez besoin.

Avec Secrets Manager, vous pouvez planifier une rotation automatique de vos secrets. Cela vous permet de remplacer les secrets à long terme par ceux à court terme, ce qui réduit considérablement le risque de mise en danger. Les informations d'identification n'étant plus stockées dans l'application, la rotation des informations d'identification ne nécessite plus la mise à jour de vos applications et le déploiement des modifications sur les clients de l'application.

Pour d'autres types de secrets que vous pourriez avoir dans votre organisation :

- AWS informations d'identification — Nous vous recommandons AWS Identity and Access Management.
- Clés de chiffrement : nous vous recommandons AWS Key Management Service.
- Clés SSH : nous vous recommandons Amazon EC2 Instance Connect.
- Clés et certificats privés : nous vous recommandons AWS Certificate Manager.

Démarrer avec Secrets Manager

Si vous utilisez Secrets Manager pour la première fois, commencez avec l'un des didacticiels suivants :

- the section called “Remplacer les secrets codés en dur ”
- the section called “Remplacer les informations d'identification de bases de données codées en dur ”

- the section called “Rotation des utilisateurs en alternance”
- the section called “Rotation utilisateur unique”

Autres tâches que vous pouvez effectuer avec des secrets :

- Gérez les secrets
- Contrôler l'accès à vos secrets
- Obtenez des secrets
- Rotation des secrets
- Surveillance des secrets
- Surveillez les secrets à des fins de conformité
- Créez des secrets dans AWS CloudFormation

Conformité aux normes

AWS Secrets Manager a fait l'objet d'un audit visant à vérifier sa conformité aux multiples normes et peut être utilisé dans le cadre de votre solution quand vous avez besoin d'obtenir une certification de conformité. Pour de plus amples informations, veuillez consulter Validation de conformité.

Tarification

Lorsque vous utilisez Secrets Manager, vous ne payez que pour ce que vous utilisez, sans frais minimum ni frais d'installation. Il n'y a pas de frais pour les secrets marqués pour suppression. Pour obtenir la liste de prix actuelle complète, consultez Tarification AWS Secrets Manager. Pour suivre vos coûts, consultezthe section called “Surveillez les coûts de Secrets Manager”.

Vous pouvez utiliser le Clé gérée par AWS `aws/secretsmanager` plugin que Secrets Manager crée pour chiffrer vos secrets gratuitement. Si vous créez vos propres clés KMS pour chiffrer vos secrets AWS , applique le AWS KMS tarif actuel. Pour plus d'informations, consultez Tarification d'AWS Key Management Service.

Lorsque vous activez la rotation automatique (à l'exception de la rotation gérée), Secrets Manager utilise une AWS Lambda fonction pour faire tourner le secret, et la fonction de rotation vous est facturée au tarif Lambda en vigueur. Pour plus d'informations, consultez Tarification d'AWS Lambda.

Si vous activez AWS CloudTrail sur votre compte, vous pouvez obtenir les journaux des appels d'API effectués par Secrets Manager. Secrets Manager consigne tous les événements en tant qu'événements de gestion. AWS CloudTrail stocke gratuitement la première copie de tous les événements de gestion. Cependant, des frais peuvent vous être facturés pour le stockage des journaux Amazon S3 et pour Amazon SNS si vous activez la notification. En outre, si vous configurez des journaux de suivi supplémentaires, les copies supplémentaires d'événements de gestion peuvent entraîner des coûts. Pour en savoir plus, consultez Pricing AWS CloudTrail (Tarification).

Vous pouvez utiliser les balises de répartition des coûts dans Secrets Manager pour suivre et classer les dépenses associées à des secrets ou à des projets spécifiques. Pour plus d'informations, reportez-vous the section called “Étiqueter les secrets ” à ce guide et à l'utilisation des balises de répartition des AWS coûts dans le guide de AWS Billing l'utilisateur.

Accès AWS Secrets Manager

Vous pouvez utiliser Secrets Manager de l'une des façons suivantes :

- Console Secrets Manager
- Outils de ligne de commande
- AWS SDKs
- API de requête HTTPS
- AWS Secrets Manager points de terminaison

Console Secrets Manager

Vous pouvez gérer vos secrets à l'aide de la console Secrets Manager basée sur un navigateur et effectuer presque toutes les tâches liées à vos secrets à l'aide de la console.

Outils de ligne de commande

Les outils de ligne de AWS commande vous permettent d'émettre des commandes sur la ligne de commande de votre système pour exécuter Secrets Manager et d'autres AWS tâches. Ceci peut être plus rapide et plus pratique qu'utiliser la console. Les outils de ligne de commande peuvent être utiles si vous souhaitez créer des scripts pour effectuer des AWS tâches.

Lorsque saisissez des commandes dans un shell de commande, il existe un risque d'accès à l'historique des commandes ou aux paramètres de vos commandes. Consultez the section called “Réduisez les risques liés AWS CLI à l'utilisation du pour stocker vos AWS Secrets Manager secrets”.

Les outils de ligne de commande utilisent automatiquement le point de terminaison par défaut pour le service dans une AWS région. Vous pouvez spécifier un point de terminaison différent pour vos requêtes d'API. Consultez the section called “Points de terminaison Secrets Manager”.

AWS fournit deux ensembles d'outils de ligne de commande :

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

AWS SDKs

Ils se AWS SDKs composent de bibliothèques et d'exemples de code pour différents langages de programmation et plateformes. Elles SDKs incluent des tâches telles que la signature cryptographique des demandes, la gestion des erreurs et le renouvellement automatique des demandes. Pour télécharger et installer l'un des SDKs, consultez la section Outils pour Amazon Web Services.

Ils utilisent AWS SDKs automatiquement le point de terminaison par défaut pour le service dans une AWS région. Vous pouvez spécifier un point de terminaison différent pour vos requêtes d'API. Consultez the section called "Points de terminaison Secrets Manager".

Pour obtenir la documentation relative aux kits SDK, consultez :

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

API de requête HTTPS

L'API de requête HTTPS vous donne un accès programmatique à Secrets Manager et AWS. L'API de requête HTTPS vous permet d'envoyer des demandes HTTPS directement au service.

Bien que vous puissiez passer des appels directs à l'API de requête HTTPS de Secrets Manager, nous vous recommandons d'utiliser l'une d'entre elles à la SDKs place. Le kit SDK effectue de nombreuses tâches utiles que vous devez sinon effectuer manuellement. Par exemple, ils

signent SDKs automatiquement vos demandes et convertissent les réponses dans une structure syntaxiquement adaptée à votre langue.

Pour passer des appels HTTPS à Secrets Manager, vous devez vous connecter à un point de terminaison ???.

AWS Secrets Manager points de terminaison

Pour vous connecter par programmation à Secrets Manager, vous devez utiliser un point de terminaison, l'URL du point d'entrée du service. Les points de terminaison Secrets Manager sont des points de terminaison à double pile, ce qui signifie qu'ils prennent en charge les deux. IPv4 IPv6

Secrets Manager propose des points de terminaison compatibles avec la Federal Information Processing Standard, (FIPS) 140-2 (Norme fédérale de traitement de l'information) dans certaines régions.

Secrets Manager prend en charge TLS 1.2 et 1.3. Secrets Manager prend en charge le protocole PQTLS dans toutes les régions à l'exception de la Chine.

> **Note**
>
> Le AWS SDK Python et la AWS CLI tentative d'appel, IPv6 puis IPv4 en séquence, donc si vous ne l'avez pas IPv6 activé, cela peut prendre un certain temps avant que l'appel expire et que vous réessayez avec. IPv4 Pour contourner ce problème, vous pouvez désactiver IPv6 complètement ou migrer vers IPv6.

Les points de terminaison de service pour Secrets Manager sont les suivants. Notez que la dénomination diffère de la convention de dénomination classique à double pile. Pour plus d'informations sur l'utilisation de l'adressage à double pile dans Secrets Manager, consultezIPv4 et IPv6 accès.

Nom de la région	Région	Point de terminaison	Protocole	
US East (Ohio)	us-east-2	secretsmanager.us-east-2.amazonaws.com	HTTPS	
		secretsmanager-fips.us-east-2.amazonaws.com	HTTPS	

Nom de la région	Région	Point de terminaison	Protocole
US East (N. Virginia)	us-east-1	secretsmanager.us-east-1.amazonaws.com secretsmanager-fips.us-east-1.amazonaws.com	HTTPS HTTPS
USA Ouest (Californie du Nord)	us-west-1	secretsmanager.us-west-1.amazonaws.com secretsmanager-fips.us-west-1.amazonaws.com	HTTPS HTTPS
USA Ouest (Oregon)	us-west-2	secretsmanager.us-west-2.amazonaws.com secretsmanager-fips.us-west-2.amazonaws.com	HTTPS HTTPS
Afrique (Le Cap)	af-south-1	secretsmanager.af-south-1.amazonaws.com	HTTPS
Asie-Pacifique (Hong Kong)	ap-east-1	secretsmanager.ap-east-1.amazonaws.com	HTTPS
Asie-Pacifique (Hyderabad)	ap-south-2	secretsmanager.ap-south-2.amazonaws.com	HTTPS
Asie-Pacifique (Jakarta)	ap-southeast-3	secretsmanager.ap-southeast-3.amazonaws.com	HTTPS
Asie-Pacifique (Malaisie)	ap-southeast-5	secretsmanager.ap-southeast-5.amazonaws.com	HTTPS

Nom de la région	Région	Point de terminaison	Protocole	
Asie-Pacifique (Melbourne)	ap-southeast-4	secretsmanager.ap-southeast-4.amazonaws.com	HTTPS	
Asia Pacific (Mumbai)	ap-south-1	secretsmanager.ap-south-1.amazonaws.com	HTTPS	
Asie-Pacifique (Osaka)	ap-northeast-3	secretsmanager.ap-northeast-3.amazonaws.com	HTTPS	
Asia Pacific (Seoul)	ap-northeast-2	secretsmanager.ap-northeast-2.amazonaws.com	HTTPS	
Asie-Pacifique (Singapour)	ap-southeast-1	secretsmanager.ap-southeast-1.amazonaws.com	HTTPS	
Asie-Pacifique (Sydney)	ap-southeast-2	secretsmanager.ap-southeast-2.amazonaws.com	HTTPS	
Asie-Pacifique (Taipei)	ap-est-2	secretsmanager.ap-east-2.amazonaws.com	HTTPS	
Asie-Pacifique (Thaïlande)	ap-sud-est 7	secretsmanager.ap-southeast-7.amazonaws.com	HTTPS	

Nom de la région	Région	Point de terminaison	Protocole
Asie-Pacifique (Tokyo)	ap-northeast-1	secretsmanager.ap-northeast-1.amazonaws.com	HTTPS
Canada (Centre)	ca-central-1	secretsmanager.ca-central-1.amazonaws.com secretsmanager-fips.ca-central-1.amazonaws.com	HTTPS HTTPS
Canada Ouest (Calgary)	ca-west-1	secretsmanager.ca-west-1.amazonaws.com secretsmanager-fips.ca-west-1.amazonaws.com	HTTPS HTTPS
Europe (Francfort)	eu-central-1	secretsmanager.eu-central-1.amazonaws.com	HTTPS
Europe (Irlande)	eu-west-1	secretsmanager.eu-west-1.amazonaws.com	HTTPS
Europe (Londres)	eu-west-2	secretsmanager.eu-west-2.amazonaws.com	HTTPS
Europe (Milan)	eu-south-1	secretsmanager.eu-south-1.amazonaws.com	HTTPS
Europe (Paris)	eu-west-3	secretsmanager.eu-west-3.amazonaws.com	HTTPS
Europe (Espagne)	eu-south-2	secretsmanager.eu-south-2.amazonaws.com	HTTPS
Europe (Stockholm)	eu-north-1	secretsmanager.eu-north-1.amazonaws.com	HTTPS

Nom de la région	Région	Point de terminaison	Protocole
Europe (Zurich)	eu-central-2	secretsmanager.eu-central-2.amazonaws.com	HTTPS
Israël (Tel Aviv)	il-central-1	secretsmanager.il-central-1.amazonaws.com	HTTPS
Mexique (centre)	mx-central-1	secretsmanager.mx-central-1.amazonaws.com	HTTPS
Moyen-Orient (Bahreïn)	me-south-1	secretsmanager.me-south-1.amazonaws.com	HTTPS
Moyen-Orient (EAU)	me-central-1	secretsmanager.me-central-1.amazonaws.com	HTTPS
Amérique du Sud (São Paulo)	sa-east-1	secretsmanager.sa-east-1.amazonaws.com	HTTPS
AWS GovCloud (USA Est)	us-gov-east-1	secretsmanager.us-gov-east-1.amazonaws.com secretsmanager-fips.us-gov-east-1.amazonaws.com	HTTPS HTTPS
AWS GovCloud (US-Ouest)	us-gov-west-1	secretsmanager.us-gov-west-1.amazonaws.com secretsmanager-fips.us-gov-west-1.amazonaws.com	HTTPS HTTPS

AWS Secrets Manager meilleures pratiques

Secrets Manager fournit un certain nombre de fonctionnalités de sécurité à prendre en compte lors de l'élaboration et de la mise en œuvre de vos propres politiques de sécurité. Les bonnes pratiques suivantes doivent être considérées comme des instructions générales et ne représentent pas une solution de sécurité complète. Étant donné que ces bonnes pratiques peuvent ne pas être appropriées ou suffisantes pour votre environnement, considérez-les comme des remarques utiles plutôt que comme des recommandations.

Tenez compte des meilleures pratiques suivantes en matière de stockage et de gestion des secrets :

- Stockez les informations d'identification et autres informations sensibles dans AWS Secrets Manager
- Trouvez des secrets non protégés dans votre code
- Choisissez une clé de chiffrement pour votre secret
- Utiliser la mise en cache pour récupérer des secrets
- Rotation de vos secrets
- Atténuer les risques liés à l'utilisation de la CLI
- Limitez l'accès aux secrets
- Répliquez les secrets
- Surveillance des secrets
- Gérez votre infrastructure sur des réseaux privés

Stockez les informations d'identification et autres informations sensibles dans AWS Secrets Manager

Secrets Manager peut vous aider à améliorer votre niveau de sécurité et votre conformité, et à réduire le risque d'accès non autorisé à vos informations sensibles. Secrets Manager chiffre les secrets au repos à l'aide des clés de chiffrement que vous possédez et dans lesquelles vous les stockez AWS Key Management Service (AWS KMS). Lorsque vous récupérez un secret, Secrets Manager le déchiffre et le transmet de manière sécurisée via TLS à votre environnement local. Pour de plus amples informations, veuillez consulter Création de secrets.

Trouvez des secrets non protégés dans votre code

CodeGuru Reviewer s'intègre à Secrets Manager pour utiliser un détecteur de secrets qui détecte les secrets non protégés dans votre code. Le détecteur de secrets recherche les mots de passe codés en dur, les chaînes de connexion à la base de données, les noms d'utilisateur, etc. Pour de plus amples informations, veuillez consulter the section called " CodeGuru Réviseur Amazon".

Amazon Q peut analyser votre base de code pour détecter les failles de sécurité et les problèmes de qualité du code afin d'améliorer la posture de vos applications tout au long du cycle de développement. Pour plus d'informations, consultez la section Numérisation de votre code avec Amazon Q dans le manuel Amazon Q Developer User Guide.

Choisissez une clé de chiffrement pour votre secret

Dans la plupart des cas, nous recommandons d'utiliser la clé `aws/secretsmanager` AWS gérée pour chiffrer les secrets. Son utilisation est gratuite.

Pour accéder à un secret depuis un autre compte ou pour appliquer une politique de clé à la clé de chiffrement, utilisez une clé gérée par le client pour chiffrer le secret.

- Dans la politique clé, attribuez la valeur `secretsmanager.<region>.amazonaws.com` à la clé de `kms:ViaService`condition. Cela limite l'utilisation de la clé aux seules demandes provenant de Secrets Manager.
- Pour limiter davantage l'utilisation de la clé aux seules demandes émanant de Secrets Manager présentant le contexte approprié, utilisez des clés ou des valeurs dans le contexte de chiffrement de Secrets Manager comme condition d'utilisation de la clé KMS en créant :
  - Un opérateur de condition de chaîne dans une politique IAM ou une politique clé
  - Une contrainte d'octroi dans un octroi

Pour de plus amples informations, veuillez consulter the section called "Chiffrement et déchiffrement de secret".

Utiliser la mise en cache pour récupérer des secrets

Pour utiliser vos secrets le plus efficacement possible, nous vous recommandons d'utiliser l'un des composants de mise en cache pris en charge par Secrets Manager suivants pour les mettre en cache et les mettre à jour uniquement lorsque cela est nécessaire :

- Java avec mise en cache côté client
- Python avec mise en cache côté client
- .NET avec mise en cache côté client
- Optez pour la mise en cache côté client
- Rust avec mise en cache côté client
- AWS Paramètres et secrets de l'extension Lambda
- the section called "Amazon EKS"
- the section called "Secrets Manager"À utiliser pour standardiser la consommation des secrets issus de Secrets Manager dans des environnements tels qu' AWS Lambda Amazon Elastic Container Service, Amazon Elastic Kubernetes Service et Amazon Elastic Compute Cloud.

Rotation de vos secrets

Si vous ne modifiez pas vos secrets pendant une longue période, ceux-ci risquent davantage d'être compromis. Avec Secrets Manager, vous pouvez configurer une rotation automatique toutes les quatre heures. Secrets Manager propose deux stratégies de rotation : Utilisateur unique etUtilisateurs en alternance. Pour de plus amples informations, veuillez consulter Rotation des secrets.

Atténuer les risques liés à l'utilisation de la CLI

Lorsque vous utilisez les AWS opérations AWS CLI pour appeler, vous entrez ces commandes dans un interpréteur de commandes. La plupart des interfaces de commande proposent des fonctionnalités susceptibles de compromettre vos secrets, telles que la journalisation et la possibilité de voir la dernière commande saisie. Avant d'utiliser le AWS CLI pour saisir des informations sensibles, assurez-vous de le fairethe section called "Réduisez les risques liés AWS CLI à l'utilisation du pour stocker vos AWS Secrets Manager secrets".

Limitez l'accès aux secrets

Dans les déclarations de politique IAM qui contrôlent l'accès à vos secrets, utilisez le principe de l'accès le moins privilégié. Vous pouvez utiliser les rôles et politiques IAM, les politiques de ressources et le contrôle d'accès basé sur les attributs (ABAC). Pour de plus amples informations, veuillez consulter the section called "Authentification et contrôle d'accès".

Rubriques

- Bloquez l'accès généralisé aux secrets
- Faites preuve de prudence en ce qui concerne les conditions d'adresse IP dans les politiques
- Limitez les demandes en fonction des conditions du point de terminaison VPC

Bloquez l'accès généralisé aux secrets

Dans les politiques d'identité qui autorisent l'action `PutResourcePolicy`, nous vous recommandons d'utiliser `BlockPublicPolicy: true`. Cette condition signifie que les utilisateurs ne peuvent attacher une politique de ressource à un secret que si la politique n'autorise pas un accès étendu.

Secrets Manager utilise le raisonnement automatisé Zelkova pour analyser les politiques de ressources en vue d'un large accès. Pour plus d'informations sur Zelkova, consultez l'article Comment AWS utilise le raisonnement automatique pour vous aider à renforcer la sécurité à grande échelle sur le blog sur la AWS sécurité.

L'exemple suivant montre comment utiliser `BlockPublicPolicy`.

JSON

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

Faites preuve de prudence en ce qui concerne les conditions d'adresse IP dans les politiques

Cependant, soyez prudent lorsque vous indiquez les opérateurs de condition d'adresse IP ou la clé de condition `aws:SourceIp` dans cette déclaration de politique qui autorise ou refuse l'accès à Secrets Manager. Par exemple, si vous associez une politique qui restreint les AWS actions aux demandes provenant de la plage d'adresses IP de votre réseau d'entreprise à un secret, vos demandes en tant qu'utilisateur IAM invoquant la demande depuis le réseau d'entreprise fonctionnent comme prévu. Toutefois, si vous autorisez d'autres services à accéder au secret en votre nom, par exemple lorsque vous activez la rotation avec une fonction Lambda, cette fonction appelle les opérations du Secrets Manager depuis un espace d' AWS adressage interne. Les demandes affectées par la politique avec le filtre d'adresses IP échouent.

Par ailleurs, la clé de condition `aws:sourceIP` devient moins efficace lorsque la demande provient d'un point de terminaison d'un VPC Amazon. Pour restreindre les demandes à un point de terminaison d'un VPC spécifique, utilisez the section called “Limitez les demandes en fonction des conditions du point de terminaison VPC”.

Limitez les demandes en fonction des conditions du point de terminaison VPC

Pour accorder ou refuser l'accès à des requêtes à partir d'un VPC ou d'un point de terminaison d'un VPC particulier, utilisez `aws:SourceVpc` pour limiter l'accès aux requêtes à partir du VPC spécifié ou `aws:SourceVpce` pour limiter l'accès aux requêtes à partir du point de terminaison d'un VPC spécifié. Consultez the section called “Exemple : autorisations et VPCs”.

- `aws:SourceVpc` limite l'accès aux requêtes à partir du VPC spécifié.
- `aws:SourceVpce` limite l'accès aux requêtes à partir du point de terminaison d'un VPC spécifié.

Si vous utilisez ces clés de condition dans une déclaration de politique de ressource qui autorise ou interdit l'accès aux secrets de Secrets Manager, vous risquez de refuser l'accès aux services qui utilisent Secrets Manager pour accéder aux secrets en votre nom. Seuls certains AWS services peuvent être exécutés avec un point de terminaison au sein de votre VPC. Si vous limitez les demandes de secret à un point de terminaison d'un VPC ou à un VPC, les appels à Secrets Manager à partir d'un service non configuré pour ce service peuvent échouer.

Consultez the section called “Points de terminaison d’un VPC (AWS PrivateLink)”.

Répliquez les secrets

Secrets Manager peut automatiquement répliquer vos secrets AWS dans plusieurs régions afin de répondre à vos exigences en matière de résilience ou de reprise après sinistre. Pour de plus amples informations, veuillez consulter Réplication multirégionale.

Surveillance des secrets

Secrets Manager vous permet d'auditer et de surveiller les secrets grâce à l'intégration aux services de AWS journalisation, de surveillance et de notification. Pour plus d'informations, consultez :

- the section called “Connectez-vous avec AWS CloudTrail ”
- the section called “Moniteur avec CloudWatch”
- the section called “Surveillez les secrets à des fins de conformité”
- the section called “Surveillez les coûts de Secrets Manager”
- the section called “Détectez les menaces avec GuardDuty”

Gérez votre infrastructure sur des réseaux privés

Nous recommandons d'exécuter la majeure partie de votre infrastructure sur des réseaux privés non accessibles à partir de l'Internet public. Vous pouvez établir une connexion privée entre votre VPC et Secrets Manager en créant un point de terminaison d'un VPC d'interface. Pour de plus amples informations, veuillez consulter the section called “Points de terminaison d’un VPC (AWS PrivateLink)”.

AWS Secrets Manager tutoriels

Rubriques

- Trouvez des secrets non protégés dans votre code avec Amazon Reviewer CodeGuru
- Déplacez les secrets codés en dur vers AWS Secrets Manager
- Déplacez les informations d'identification codées en dur vers AWS Secrets Manager
- Configurer une rotation des utilisateurs en alternance pour AWS Secrets Manager
- Configurer la rotation utilisateur unique pour AWS Secrets Manager

Trouvez des secrets non protégés dans votre code avec Amazon Reviewer CodeGuru

Amazon CodeGuru Reviewer est un service qui utilise l'analyse des programmes et l'apprentissage automatique pour détecter les défauts potentiels difficiles à détecter pour les développeurs et propose des suggestions pour améliorer votre code Java et Python. CodeGuru Reviewer s'intègre à Secrets Manager pour détecter les secrets non protégés dans votre code. Pour connaître les types de secrets qu'il peut trouver, consultez la section Types de secrets détectés par CodeGuru Reviewer dans le guide de l'utilisateur Amazon CodeGuru Reviewer.

Une fois que vous avez trouvé des secrets codés en dur, prenez des mesures pour les remplacer :

- the section called “Remplacer les informations d'identification de bases de données codées en dur ”
- the section called “Remplacer les secrets codés en dur ”

Déplacez les secrets codés en dur vers AWS Secrets Manager

Si vous avez des secrets en texte brut dans votre code, nous vous recommandons de les faire tourner et les stocker dans Secrets Manager. Le déplacement du secret vers Secrets Manager résout le problème de la visibilité du secret par toute personne qui voit le code, car à l'avenir, votre code récupère le secret directement depuis Secrets Manager. La rotation du secret révoque le secret actuellement codé en dur pour qu'il ne soit plus valide.

Pour les secrets des informations d'identification de la base de données, consultez Déplacez les informations d'identification codées en dur vers AWS Secrets Manager.

Avant de commencer, déterminez qui a besoin d'accéder au secret. Nous vous recommandons d'utiliser deux rôles IAM pour gérer les autorisations d'accès à votre secret :

- Un rôle qui gère les secrets de votre organisation. Pour de plus amples informations, veuillez consulter the section called "Secrets Manager". Vous allez créer et faire tourner le secret à l'aide de ce rôle.
- Rôle qui peut utiliser le secret lors de l'exécution, par exemple dans ce didacticiel que vous utilisez`RoleToRetrieveSecretAtRuntime`. Votre code assume ce rôle pour récupérer le secret. Dans ce tutoriel, vous accordez au rôle uniquement l'autorisation de récupérer une valeur secrète unique, et vous accordez l'autorisation à l'aide de la politique de ressources du secret. Pour d'autres alternatives, consultez la page the section called "Étapes suivantes".

Étapes :

- Étape 1 : créer le secret
- Étape 2 : mettre à jour votre code
- Étape 3 : mettre à jour le secret
- Étapes suivantes

Étape 1 : créer le secret

La première étape consiste à copier le secret codé en dur existant dans dans Secrets Manager. Si le secret est lié à une AWS ressource, stockez-le dans la même région que la ressource. Sinon, stockez-le dans la région où la latence est la plus faible pour votre cas d'utilisation.

Pour créer des secrets (console)

1. Ouvrez la console Secrets Manager à l'adresse https://console.aws.amazon.com/secretsmanager/.
2. Choisissez Store a new secret (Stocker un nouveau secret).
3. Sur la page Choose secret type (Choisir un type de secret), procédez comme suit :
    a. Pour Secret type (Type de secret), choisissez Other type of secret (Autre type de secret).
    b. Entrez votre secret sous forme de Paires clé-valeur ou texte brut. Voici quelques exemples :

API key

Entrez par key/value paires :

`ClientID` : *`my_client_id`*

`ClientSecret` : *`wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`*

OAuth token

Entrez en texte brut :

`AKIAI44QH8DHBEXAMPLE`

Digital certificate

Entrez en texte brut :

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

Private key

Entrez en texte brut :

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

c. Pour Encryption key (clé de cryptage), choisissez aws/secretsmanager pour utiliser la Clé gérée par AWS pour Secrets Manager. L'utilisation de cette clé n'entraîne aucun coût. Vous pouvez également utiliser votre propre clé gérée par le client, par exemple pour accéder au secret d'un autre Compte AWS. Pour plus d'informations sur les coûts d'utilisation d'une clé gérée par le client, consultez Tarification.

d. Choisissez Suivant.

4. Sur la page Choose secret type (Choisir un type de secret), procédez comme suit :

a. Saisissez un Secret name (Nom de secret) descriptif et une Description.

b. Dans Resource permissions (Autorisations des ressources), choisissez Edit permissions (Modifier les autorisations). Collez la politique suivante, qui

RoleToRetrieveSecretAtRuntime permet de récupérer le secret, puis choisissez Enregistrer.

JSON

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

c. Au bas de la page, sélectionnez Next.

5. Sur la page Configure rotation (Configurer la rotation), maintenez la rotation désactivée. Choisissez Suivant.
6. Dans la page Review (Révision), passez en revue vos paramètres, puis choisissez Store (Stocker).

Étape 2 : mettre à jour votre code

Votre code doit assumer le rôle IAM *RoleToRetrieveSecretAtRuntime* pour pouvoir récupérer le secret. Pour plus d'informations, consultez la section Passage à un rôle IAM (AWS API).

Ensuite, mettez à jour votre code pour récupérer le secret à partir de Secrets Manager à l'aide de l'exemple de code fourni par Secrets Manager.

Pour trouver l'exemple de code

1. Ouvrez la console Secrets Manager à l'adresse https://console.aws.amazon.com/secretsmanager/.
2. Sur la page Secrets, choisissez votre secret.

3. Faites défiler la page vers le bas jusqu'à Sample code (Exemple de code). Choisissez votre langue de programmation, puis copiez l'extrait de code.

Dans votre application, supprimez le secret codé en dur et collez l'extrait de code. Selon la langue de votre code, vous devrez peut-être ajouter un appel à la fonction ou à la méthode dans l'extrait de code.

Vérifiez que votre application fonctionne comme prévu avec le secret à la place du secret codé en dur.

Étape 3 : mettre à jour le secret

La dernière étape consiste à révoquer et mettre à jour le secret codé en dur. Reportez-vous à la source du secret pour obtenir des instructions pour révoquer et mettre à jour le secret. Par exemple, vous devrez peut-être désactiver le secret actuel et générer un nouveau secret.

Pour mettre à jour le secret avec la nouvelle valeur

1. Ouvrez la console Secrets Manager à l'adresse https://console.aws.amazon.com/secretsmanager/.
2. Choisissez Secrets, puis choisissez le secret.
3. Sur la page Secret details (Détails secrets), faites défiler l'écran vers le bas et choisissez Retrieve secret value (Récupération d'une valeur de secret), puis choisissez Edit (Modifier).
4. Mettez à jour le secret, puis choisissez Save (Enregistrer).

Ensuite, vérifiez que votre application fonctionne comme prévu avec le nouveau secret.

Étapes suivantes

Après avoir supprimé un secret codé en dur de votre code, voici quelques idées à considérer pour la suite :

- Pour trouver des secrets codés en dur dans vos applications Java et Python, nous vous recommandons Amazon CodeGuru Reviewer.
- Vous pouvez améliorer les performances et réduire les coûts en mettant en cache des secrets. Pour de plus amples informations, veuillez consulter Obtenez des secrets.

- Pour connaître les secrets auxquels vous accédez depuis plusieurs régions, envisagez de répliquer votre secret pour améliorer la latence. Pour de plus amples informations, veuillez consulter Réplication multirégionale.
- Dans ce didacticiel, vous *RoleToRetrieveSecretAtRuntime* n'avez accordé que l'autorisation de récupérer la valeur secrète. Pour accorder plus d'autorisations au rôle, par exemple pour obtenir des métadonnées sur le secret ou afficher une liste de secrets, consultez the section called "Politiques basées sur les ressources".
- Dans ce didacticiel, vous avez accordé l'autorisation en *RoleToRetrieveSecretAtRuntime* utilisant la politique de ressources du secret. Pour découvrir d'autres moyens d'accorder une autorisation, consultez the section called "Politiques basées sur l'identité".

Déplacez les informations d'identification codées en dur vers AWS Secrets Manager

Si vous avez des informations d'identification de base de données en texte brut dans votre code, nous vous recommandons de les déplacer vers Secrets Manager, puis de les faire tourner immédiatement. Le déplacement des informations d'identification vers Secrets Manager résout le problème de la visibilité des informations d'identification par toute personne qui voit le code, car à l'avenir, votre code récupère les informations d'identification directement depuis Secrets Manager. La rotation du secret met à jour le mot de passe, puis révoque le mot de passe actuellement codé en dur pour qu'il ne soit plus valide.

Pour les bases de données Amazon RDS, Amazon Redshift et Amazon DocumentDB, suivez les étapes de cette page pour déplacer les informations d'identification codées en dur vers Secrets Manager. Pour d'autres types d'informations d'identification et d'autres secrets, consultez the section called "Remplacer les secrets codés en dur ".

Avant de commencer, déterminez qui a besoin d'accéder au secret. Nous vous recommandons d'utiliser deux rôles IAM pour gérer les autorisations d'accès à votre secret :

- Un rôle qui gère les secrets de votre organisation. Pour de plus amples informations, veuillez consulter the section called "Secrets Manager". Vous allez créer et faire tourner le secret à l'aide de ce rôle.
- *RoleToRetrieveSecretAtRuntime*Dans ce didacticiel, un rôle qui peut utiliser les informations d'identification lors de l'exécution. Votre code assume ce rôle pour récupérer le secret.

Étapes :

- Étape 1 : créer le secret
- Étape 2 : mettre à jour votre code
- Étape 3 : effectuer la rotation du secret
- Étapes suivantes

Étape 1 : créer le secret

La première étape consiste à copier les informations d'identification codées en dur existantes dans un secret dans Secrets Manager. Pour la latence la plus faible, stockez le secret dans la même région que la base de données.

Pour créer un secret

1. Ouvrez la console Secrets Manager à l'adresse https://console.aws.amazon.com/secretsmanager/.
2. Choisissez Store a new secret (Stocker un nouveau secret).
3. Sur la page Choose secret type (Choisir un type de secret), procédez comme suit :

    a. Pour Secret type (Type de secret), choisissez le type d'informations d'identification de base de données à stocker :

    - Bases de données Amazon RDS
    - Base de données Amazon DocumentDB
    - Entrepôt de données Amazon Redshift.
    - Pour d'autres types de secrets, consultez Remplacer les secrets codés en dur.

    b. Pour Informations d'identification, saisissez les informations d'identification codées en dur existantes de la base de données.

    c. Pour Encryption key (clé de cryptage), choisissez aws/secretsmanager pour utiliser la Clé gérée par AWS pour Secrets Manager. L'utilisation de cette clé n'entraîne aucun coût. Vous pouvez également utiliser votre propre clé gérée par le client, par exemple pour accéder au secret d'un autre Compte AWS. Pour plus d'informations sur les coûts d'utilisation d'une clé gérée par le client, consultez Tarification.

    d. Pour Database (Base de données), choisissez votre base de données.

    e. Choisissez Suivant.

4. Sur la page Configure secret (Configurer le secret), procédez comme suit :

   a. Saisissez un Secret name (Nom de secret) descriptif et une Description.

   b. Dans Resource permissions (Autorisations des ressources), choisissez Edit permissions (Modifier les autorisations). Collez la politique suivante, qui *RoleToRetrieveSecretAtRuntime* permet de récupérer le secret, puis choisissez Enregistrer.

      JSON

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

   c. Au bas de la page, sélectionnez Next.

5. Sur la page Configure rotation (Configurer la rotation), maintenez la rotation désactivée pour le moment. Vous l'activerez plus tard. Choisissez Suivant.

6. Dans la page Review (Révision), passez en revue vos paramètres, puis choisissez Store (Stocker).

Étape 2 : mettre à jour votre code

Votre code doit assumer le rôle IAM *RoleToRetrieveSecretAtRuntime* pour pouvoir récupérer le secret. Pour plus d'informations, consultez la section Passage à un rôle IAM (AWS API).

Ensuite, mettez à jour votre code pour récupérer le secret à partir de Secrets Manager à l'aide de l'exemple de code fourni par Secrets Manager.

Pour trouver l'exemple de code

1. Ouvrez la console Secrets Manager à l'adresse https://console.aws.amazon.com/secretsmanager/.
2. Sur la page Secrets, choisissez votre secret.
3. Faites défiler la page vers le bas jusqu'à Sample code (Exemple de code). Choisissez votre langue, puis copiez l'extrait de code.

Dans votre application, supprimez les informations d'identification codées en dur et collez l'extrait de code. Selon la langue de votre code, vous devrez peut-être ajouter un appel à la fonction ou à la méthode dans l'extrait de code.

Vérifiez que votre application fonctionne comme prévu avec le secret à la place des informations d'identification codées en dur.

Étape 3 : effectuer la rotation du secret

La dernière étape consiste à révoquer les informations d'identification codées en dur en faisant tourner le secret. La Rotation est le processus de mise à jour périodique d'un secret. Lorsque vous effectuez une rotation de secret, vous mettez à jour les informations d'identification dans le secret et la base de données. Secrets Manager peut faire tourner automatiquement un secret pour vous selon une planification que vous définissez.

Une partie de la configuration de la rotation consiste à s'assurer que la fonction de rotation Lambda peut accéder à la fois à Secrets Manager et à votre base de données. Lorsque vous activez la rotation automatique, Secrets Manager crée la fonction de rotation Lambda dans le même VPC que votre base de données afin qu'elle ait un accès réseau à la base de données. La fonction de rotation Lambda doit également pouvoir appeler Secrets Manager pour mettre à jour le secret. Nous vous recommandons de créer un point de terminaison Secrets Manager dans le VPC afin que les appels de Lambda à Secrets Manager ne quittent pas l'infrastructure. AWS Pour obtenir des instructions, veuillez consulter the section called “Points de terminaison d’un VPC (AWS PrivateLink)”.

Pour activer la rotation

1. Ouvrez la console Secrets Manager à l'adresse https://console.aws.amazon.com/secretsmanager/.
2. Sur la page Secrets, choisissez votre secret.

3. Sur la page Secret details (Détails du secret), dans la section Rotation configuration (Configuration de la rotation), choisissez Edit rotation (Modifier la rotation).
4. Dans la boîte de dialogue Edit rotation configuration (Modifier la configuration de la rotation), suivez la procédure suivante :

   a. Activez Automatic rotation (Rotation automatique).

   b. Sous Rotation schedule (Planification de rotation), saisissez votre planification dans le fuseau horaire UTC.

   c. Choisissez Rotate immediately when the secret is stored (Effectuer immédiatement une rotation lorsque le secret est stocké) pour effectuer une rotation de votre secret lorsque vous enregistrez vos modifications.

   d. Sous Rotation function (Fonction de rotation), choisissez Create a new Lambda function (Créer une fonction Lambda) et saisissez le nom de votre nouvelle fonction. Secrets Manager ajoute « SecretsManager » au début du nom de votre fonction.

   e. Pour Stratégie de rotation, choisissez Utilisateur unique.

   f. Choisissez Enregistrer.

Pour vérifier que le secret a tourné

1. Ouvrez la console Secrets Manager à l'adresse https://console.aws.amazon.com/secretsmanager/.
2. Choisissez Secrets, puis choisissez le secret.
3. Sur la page Secret details (Détails secrets), faites défiler l'écran vers le bas et choisissez Retrieve secret value (Récupération d'une valeur de secret).

   Si la valeur secrète a changé, la rotation a réussi. Si la valeur secrète n'a pas changé, vous devez consulter Résolution des problèmes de rotation les CloudWatch journaux de la fonction de rotation.

Vérifiez que votre application fonctionne comme prévu avec le secret provenant de la rotation.

Étapes suivantes

Après avoir supprimé un secret codé en dur de votre code, voici quelques idées à considérer pour la suite :

- Vous pouvez améliorer les performances et réduire les coûts en mettant en cache des secrets. Pour de plus amples informations, veuillez consulter Obtenez des secrets.
- Vous pouvez choisir une planification de rotation différente. Pour de plus amples informations, veuillez consulter the section called "Horaires de rotation".
- Pour trouver des secrets codés en dur dans vos applications Java et Python, nous vous recommandons Amazon CodeGuru Reviewer.

Configurer une rotation des utilisateurs en alternance pour AWS Secrets Manager

Dans ce tutoriel, vous apprendrez comment configurer une rotation des utilisateurs en alternance pour un secret contenant des informations d'identification de base de données. La rotation des utilisateurs en alternance est une stratégie de rotation dans laquelle Secrets Manager clone l'utilisateur, puis alterne les informations d'identification de l'utilisateur mises à jour. Cette stratégie est un bon choix si vous avez besoin d'une haute disponibilité pour votre secret, car l'un des utilisateurs en alternance dispose d'informations d'identification actuelles pour la base de données pendant que l'autre est en cours de mise à jour. Pour de plus amples informations, veuillez consulter the section called "Utilisateurs en alternance".

Pour configurer la rotation des utilisateurs en alternance, deux secrets sont nécessaires :

- Un secret avec les informations d'identification sur lequel vous souhaitez effectuer la rotation.
- Un deuxième secret qui possède des informations d'identification d'administrateur.

  Cet utilisateur est autorisé à cloner le premier utilisateur et à modifier le mot de passe du premier utilisateur. Dans ce didacticiel, vous demandez à Amazon RDS de créer ce secret pour un utilisateur administrateur. Amazon RDS gère également la rotation des mots de passe d'administrateur. Pour de plus amples informations, veuillez consulter the section called "Rotation gérée".

La première partie de ce didacticiel consiste à mettre en place un environnement réaliste. Pour vous montrer comment fonctionne la rotation, ce tutoriel utilise un exemple de base de données Amazon RDS MySQL. Pour des raisons de sécurité, la base de données se trouve dans un VPC qui restreint l'accès à Internet. Pour vous connecter à la base de données depuis votre ordinateur local via Internet, vous utilisez un hôte bastion, un serveur du VPC qui peut se connecter à la base de données, mais qui autorise également les connexions SSH depuis Internet. Dans ce didacticiel,

l'hôte bastion est une EC2 instance Amazon, et les groupes de sécurité associés à cette instance empêchent d'autres types de connexions.

Une fois le didacticiel terminé, nous vous recommandons d'en nettoyer les ressources. Ne les utilisez pas dans un environnement de production.

La rotation de Secrets Manager utilise une AWS Lambda fonction pour mettre à jour le secret et la base de données. Pour plus d'informations sur les coûts d'utilisation d'une fonction Lambda, consultez Tarification.

Didacticiel :

- Autorisations
- Prérequis
- Étape 1 : créer un utilisateur de base de données Amazon RDS
- Étape 2 : créez un secret pour les informations d'identification de l'utilisateur
- Étape 3 : Test de rotation du secret
- Étape 4 : Nettoyer les ressources
- Étapes suivantes

Autorisations

Pour les prérequis du tutoriel, vous devez disposer d'autorisations administratives sur votre Compte AWS. Dans un contexte de production, une bonne pratique est d'utiliser différents rôles pour chacune des étapes. Par exemple, un rôle doté d'autorisations d'administrateur de base de données créerait la base de données Amazon RDS, et un rôle doté d'autorisations d'administrateur réseau configurerait le VPC et les groupes de sécurité. Pour les étapes du tutoriel, nous vous recommandons de continuer à utiliser la même identité.

Pour plus d'informations sur la façon de configurer les autorisations dans un environnement de production, consultez the section called “Authentification et contrôle d’accès”.

Prérequis

Pour ce didacticiel, vous avez besoin des éléments suivants :

- Prérequis A : Amazon VPC
- Prérequis B : instance Amazon EC2

- Prérequis C : base de données Amazon RDS et secret Secrets Manager pour les informations d’identification d'administrateur
- Prérequis D : Autoriser votre ordinateur local à se connecter à l'instance EC2

Prérequis A : Amazon VPC

Au cours de cette étape, vous créez un VPC dans lequel vous pouvez lancer une base de données Amazon RDS et une instance Amazon EC2. Dans une étape ultérieure, vous utiliserez votre ordinateur pour vous connecter via Internet au bastion, puis à la base de données. Vous devez donc autoriser le trafic sortant du VPC. Pour ce faire, Amazon VPC attache une passerelle Internet au VPC et ajoute une route dans la table de routage afin que le trafic destiné à l'extérieur du VPC soit envoyé vers la passerelle Internet.

Dans le VPC, vous créez un point de terminaison Secrets Manager et un point de terminaison Amazon RDS. Lorsque vous configurez la rotation automatique lors de l'étape ultérieure, Secrets Manager crée une fonction de rotation Lambda dans le VPC afin qu'il ait accès à la base de données. La fonction de rotation Lambda appelle également Secrets Manager pour mettre à jour le secret, et appelle Amazon RDS pour obtenir les informations de connexion à la base de données. En créant des points de terminaison au sein du VPC, vous vous assurez que les appels de la fonction Lambda vers Secrets Manager et Amazon RDS ne quittent pas l'infrastructure. AWS Au lieu de cela, ils sont acheminés vers les points de terminaison dans le VPC.

Pour créer un VPC

1. Ouvrez la console Amazon VPC à l’adresse https://console.aws.amazon.com/vpc/.
2. Sélectionnez Create VPC (Créer un VPC).
3. Sur la page Create VPC (Créer un VPC), choisissez VPC and more (VPC et plus).
4. Sous Name tag auto-generation (Génération automatique de balises nominatives), sous Auto-generate (Génération automatique), saisissez **`SecretsManagerTutorial`**.
5. Pour les options DNS, choisissez à la fois **`Enable DNS hostnames`** et **`Enable DNS resolution`**.
6. Sélectionnez Create VPC (Créer un VPC).

Pour créer un point de terminaison Secrets Manager au sein du VPC

1. Dans la console Amazon VPC, sous Endpoints (Points de terminaison), choisissez Create Endpoint (Créer un point de terminaison).
2. Dans Endpoint settings (Paramètres du point de terminaison), dans le champ Name (Nom), saisissez **`SecretsManagerTutorialEndpoint`**.
3. Sous Services, saisissez **`secretsmanager`** pour filtrer la liste, puis sélectionnez le point de terminaison Secrets Manager dans votre Région AWS. Par exemple, dans l'Est des États-Unis (Virginie du Nord), choisissez `com.amazonaws.us-east-1.secretsmanager`.
4. Pour VPC, choisissez **`vpc**** (SecretsManagerTutorial)`**.
5. Pour Subnets (Sous-réseaux), sélectionnez toutes les Availability Zones (Zones de disponibilité), puis pour chacune d'elle, choisissez un Subnet ID (ID de sous-réseau) à inclure.
6. Pour IP address type (Type d'adresse IP), choisissez **`IPv4`**.
7. Pour Security Groups (Groupes de sécurité), choisissez le groupe de sécurité par défaut.
8. Pour Policy type (Type de politique), choisissez **`Full access`**.
9. Choisissez Créer un point de terminaison.

Pour créer un point de terminaison Amazon RDS au sein du VPC

1. Dans la console Amazon VPC, sous Endpoints (Points de terminaison), choisissez Create Endpoint (Créer un point de terminaison).
2. Dans Endpoint settings (Paramètres du point de terminaison), dans le champ Name (Nom), saisissez **`RDSTutorialEndpoint`**.
3. Sous Services, saisissez **`rds`** pour filtrer la liste, puis sélectionnez le point de terminaison Amazon RDS dans votre Région AWS. Par exemple, dans l'Est des États-Unis (Virginie du Nord), choisissez `com.amazonaws.us-east-1.rds`.
4. Pour VPC, choisissez **`vpc**** (SecretsManagerTutorial)`**.
5. Pour Subnets (Sous-réseaux), sélectionnez toutes les Availability Zones (Zones de disponibilité), puis pour chacune d'elle, choisissez un Subnet ID (ID de sous-réseau) à inclure.
6. Pour IP address type (Type d'adresse IP), choisissez **`IPv4`**.
7. Pour Security Groups (Groupes de sécurité), choisissez le groupe de sécurité par défaut.
8. Pour Policy type (Type de politique), choisissez **`Full access`**.
9. Choisissez Créer un point de terminaison.

Prérequis B : instance Amazon EC2

La base de données Amazon RDS que vous créerez ultérieurement se trouvera dans le VPC. Pour y accéder, vous aurez besoin d'un hôte bastion. L'hôte bastion se trouve également dans le VPC, mais dans une étape ultérieure, vous allez configurer un groupe de sécurité pour permettre à l'ordinateur local de connecter l'hôte bastion avec SSH.

Pour créer une EC2 instance pour un hôte bastion

1. Ouvrez la EC2 console Amazon à l'adresse https://console.aws.amazon.com/ec2/.
2. Choisissez Instances, puis choisissez Launch Instances (Lancer les instances).
3. Sous Name and tags (Nom et identifications), pour Name (Nom), saisissez **`SecretsManagerTutorialInstance`**.
4. Sous Images de l'application et du système d'exploitation, conservez la valeur par défaut**`Amazon Linux 2 AMI (HMV) Kernel 5.10`**.
5. Sous Type d'instance, conservez la valeur par défaut **`t2.micro`**.
6. Sous Paire de clés, choisissez Créer une paire de clés.

   Dans la boîte de dialogue Créer une paire de clés, pour Nom de la paire de clés, entrez **`SecretsManagerTutorialKeyPair`**, puis choisissez Créer une paire de clés.

   La paire de clése est automatiquement téléchargée.
7. Sous Network settings (Paramètres réseau), choisissez Edit (Modifier), puis procédez comme suit :

   a. Pour VPC, choisissez **`vpc-**** SecretsManagerTutorial`**.

   b. Pour Auto-assign Public IP (Attribuer automatiquement l'adresse IP publique), choisissez **`Enable`**.

   c. Pour Firewall (pare-feu, choisissez Select existing security group (Choisir le groupe de sécurité existant).

   d. Pour Groupes de sécurité communs, sélectionnez **`default`**.
8. Sélectionnez Launch instance (Lancer une instance).

Prérequis C : base de données Amazon RDS et secret Secrets Manager pour les informations d’identification d'administrateur

Au cours de cette étape, vous allez créer une base de données MySQL Amazon RDS et la configurer afin que Amazon RDS créer un secret pour contenir les informations d’identification d’administrateur. Amazon RDS gère ensuite automatiquement la rotation du secret d'administration pour vous. Pour de plus amples informations, veuillez consulter Rotation gérée.

Dans le cadre de la création de votre base de données, vous devez indiquer l'hôte bastion que vous avez créé à l'étape précédente. Amazon RDS configure ensuite des groupes de sécurité afin que la base de données et l'instance puissent accéder l'une à l'autre. Vous ajoutez une règle au groupe de sécurité associé à l’instance pour permettre à votre ordinateur local de s'y connecter également.

Pour créer une base de données Amazon RDS avec un secret Secrets Manager contenant les informations d’identification de l'administrateur

1. Dans la console Amazon RDS, choisissez Créer une base de données.
2. Dans la section Options de moteur, pour Type de moteur, choisissez **`MySQL`**.
3. Dans la section Templates (Modèles), choisissez .
4. Dans la section Settings (Paramètres), procédez comme suit :
    a. Pour l'identifiant de l'instance DB, saisissez **`SecretsManagerTutorial`**.
    b. Sous Paramètres des informations d'identification, sélectionnez Gérer les informations d'identification principales dans AWS Secrets Manager.
5. Dans la section Connectivité, pour Ressource informatique, choisissez Se connecter à une ressource EC2 informatique, puis pour EC2 Instance, choisissez**`SecretsManagerTutorialInstance`**.
6. Choisissez Créer une base de données.

Prérequis D : Autoriser votre ordinateur local à se connecter à l'instance EC2

Au cours de cette étape, vous configurez l' EC2 instance que vous avez créée dans Prereq B pour permettre à votre ordinateur local de s'y connecter. Pour ce faire, vous devez modifier le groupe de sécurité qu'Amazon RDS a ajouté dans le prérequis C afin d'inclure une règle permettant à l'adresse IP de votre ordinateur de se connecter via SSH. La règle permet à votre ordinateur local (identifié par votre adresse IP actuelle) de se connecter à l'hôte bastion via SSH sur Internet.

Pour autoriser votre ordinateur local à se connecter à l' EC2 instance

1. Ouvrez la EC2 console Amazon à l'adresse https://console.aws.amazon.com/ec2/.
2. Sur l' EC2 instance SecretsManagerTutorialInstance, sous l'onglet Sécurité, sous Groupes de sécurité, sélectionnez**sg-*** (ec2-rds-X)**.
3. Sous l'onglet Input rules (Règles entrantes), sélectionnez Edit inbound rules (Modifier les règles entrantes).
4. Choisissez Ajouter une règle et procédez comme suit pour la règle :
   a. Dans le champ Type, sélectionnez **SSH**.
   b. Pour Type de source, choisissez **My IP**.

Étape 1 : créer un utilisateur de base de données Amazon RDS

Tout d'abord, vous avez besoin d'un utilisateur dont les informations d'identification seront stockées dans le secret. Pour créer l'utilisateur, connectez-vous à la base de données Amazon RDS à l'aide des informations d'identification d'administrateur. Pour des raisons de simplicité, dans le didacticiel, vous allez créer un utilisateur avec des autorisations complètes sur une base de données. Dans un environnement de production, ce n'est pas courant et nous vous recommandons de respecter le principe du moindre privilège.

Pour vous connecter à la base de données, vous utilisez un outil client MySQL. Dans ce tutoriel, vous utiliserez MySQL Workbench, une application basée sur une interface graphique. Téléchargez et installez MySQL Workbench depuis Télécharger MySQL Workbench.

Pour vous connecter à la base de données, créez une configuration de connexion dans MySQL Workbench. Pour la configuration, vous avez besoin de certaines informations provenant d'Amazon EC2 et d'Amazon RDS.

Pour créer une connexion à une base de données dans MySQL Workbench

1. Dans MySQL Workbench, à côté de MySQL Connections (Connexions MySQL), choisissez le bouton (+).
2. Dans la boîte de dialogue Setup New Connection (Configurer une nouvelle connexion), effectuez les tâches suivantes :
   a. Pour Connection Name (Nom de connexion), saisissez **SecretsManagerTutorial**.

b. Pour Connection Method (Méthode de connexion), choisissez **`Standard TCP/IP over SSH`**.

c. Sur l'onglet Parameters (Paramètres), procédez comme suit :

i. Pour le nom d'hôte SSH, entrez l'adresse IP publique de l'instance Amazon EC2 .

Vous pouvez trouver l'adresse IP sur la EC2 console Amazon en choisissant l'instance SecretsManagerTutorialInstance. Copiez l'adresse IP sous IPv4 DNS public.

ii. Pour SSH Username (Nom d'utilisateur SSH), saisissez **`ec2-user`**.

iii. Pour le fichier de clés SSH, choisissez le fichier de paire de clés SecretsManagerTutorialKeyPair.pem que vous avez téléchargé dans le prérequis précédent.

iv. Pour MySQL Hostname (Nom d'hôte MySQL), saisissez l'adresse du point de terminaison Amazon RDS.

Vous pouvez trouver l'adresse du point de terminaison sur la console Amazon RDS en choisissant l'instance de base de données secretsmanagertutorialdb. Copiez l'adresse sous Endpoint (Point de terminaison).

v. Pour Username (Nom d'utilisateur), saisissez **`admin`**.

d. Choisissez OK.

Pour récupérer le mot de passe administrateur

1. Dans la console Amazon RDS, accédez à votre base de données.
2. Dans l'onglet Configuration, sous Master Credentials ARN, choisissez Gérer dans Secrets Manager.

   La console Secrets Manager s'ouvre.
3. Sur la page de détails de votre secret, sélectionnez Retrieve secret value (Récupérer la valeur du secret).
4. Le mot de passe apparaît dans la section Valeur secrète.

Pour créer un utilisateur de base de données

1. Dans MySQL Workbench, choisissez la connexion SecretsManagerTutorial.
2. Saisissez le mot de passe administrateur que vous avez récupéré dans le secret.

3. Dans le MySQL Workbench, dans la fenêtre Query (Requête), saisissez les commandes suivantes (y compris un mot de passe fort), puis choisissez Execute (Exécuter). La fonction de rotation teste le secret mis à jour à l'aide de SELECT, de sorte qu'ils **appuser** doivent avoir ce privilège au minimum.

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

   Dans la fenêtre Output (Sortie), vous voyez que les commandes sont réussies.

Étape 2 : créez un secret pour les informations d'identification de l'utilisateur

Ensuite, vous créez un secret pour stocker les informations d'identification de l'utilisateur que vous venez de créer. C'est le secret sur lequel vous allez effectuer la rotation. Vous activez la rotation automatique et, pour indiquer la stratégie des utilisateurs en alternance, vous choisissez un secret de super-utilisateur distinct autorisé à modifier le mot de passe du premier utilisateur.

1. Ouvrez la console Secrets Manager à l'adresse https://console.aws.amazon.com/secretsmanager/.
2. Choisissez Store a new secret (Stocker un nouveau secret).
3. Sur la page Choose secret type (Choisir un type de secret), procédez comme suit :
   a. Pour Secret type (Type de secret), choisissez Credentials for Amazon RDS database (Informations d'identification pour une base de données Amazon RDS).
   b. Pour Credentials (Informations d'identification), saisissez le nom d'utilisateur **appuser** et le mot de passe que vous avez entrés pour l'utilisateur de base de données que vous avez créé à l'aide de MySQL Workbench.
   c. Pour Database (Base de données), choisissez secretsmanagertutorialdb.
   d. Choisissez Next (Suivant).
4. Sur la page Configure secret (Configurer le secret), pour Secret name (Nom du secret), saisissez **SecretsManagerTutorialAppuser** puis choisissez Next (Suivant).
5. Sur la page Configure rotation (Configuration de la rotation), procédez comme suit :
   a. Activez Automatic rotation (Rotation automatique).

b. Pour Rotation schedule (Calendrier de rotation), définissez un calendrier de Days (Jours) : **2** Jours avec une Duration (Durée) :**2h**. Gardez Rotate immediately (Rotation immédiate) sélectionné.

c. Pour Rotation function (Fonction de rotation), choisissez Create a rotation function (Créer une fonction de rotation), puis pour le nom de la fonction, saisissez **`tutorial-alternating-users-rotation`**.

d. Pour la Stratégie de rotation, choisissez Utilisateurs en alternance, puis sous Informations d'identification d'administrateur du secret, choisissez le secret nommé rds !cluster... dont la description inclut le nom de la base de données que vous avez créée dans ce didacticiel **`secretsmanagertutorial`**, par exemple `Secret associated with primary RDS DB instance: arn:aws:rds:`*`Region`*`:`*`AccountId`*`:db:secretsmanagertutorial`.

e. Choisissez Next (Suivant).

6. Sur la page Review (Vérification), choisissez Store (Stocker).

   Secrets Manager revient à la page des détails du secret. En haut de la page, vous pouvez voir l'état de la configuration de rotation. Secrets Manager permet CloudFormation de créer des ressources telles que la fonction de rotation Lambda et un rôle d'exécution qui exécute la fonction Lambda. Lorsque vous CloudFormation avez terminé, la bannière devient « Secret », dont la rotation est planifiée. La première rotation est configurée.

Étape 3 : Test de rotation du secret

Maintenant que la rotation du secret a été effectuée, vous pouvez vérifier que le secret contient des informations d'dentification valides. Le mot de passe dans le secret a changé par rapport aux informations d'identification d'origine.

Pour récupérer le nouveau mot de passe du secret

1. Ouvrez la console Secrets Manager à l'adresse https://console.aws.amazon.com/secretsmanager/.
2. Choisissez Secrets, puis choisissez le secret **`SecretsManagerTutorialAppuser`**.
3. Sur la page Secret details (Détails secrets), faites défiler l'écran vers le bas et choisissez Retrieve secret value (Récupération d'une valeur de secret).
4. Dans le tableau Key/value (Clé/Valeur), copiez la Secret value (Valeur de secret) pour **`password`**.

Pour tester les informations d'identification

1. Dans MySQL Workbench, cliquez avec le bouton droit sur la connexion, SecretsManagerTutorialpuis choisissez Modifier la connexion.
2. Dans la boite de dialogue Manage Server Connections (Gérer les connexions aux serveurs), pour Username (Nom d'utilisateur), saisissez **`appuser`**, puis choisissez Close (Fermer).
3. De retour dans MySQL Workbench, choisissez la connexion SecretsManagerTutorial.
4. Dans la boite de dialogue Open SSH Connection (Connexion SSH ouverte), pour Password (Mot de passe), collez le mot de passe que vous avez récupéré dans le secret, puis choisissez OK..

   Si les informations d'identification sont valides, MySQL Workbench s'ouvre sur la page de conception de la base de données.

Cela montre que la rotation du secret est réussie. Les informations d'identification du secret ont été mises à jour et c'est un mot de passe valide pour se connecter à la base de données.

Étape 4 : Nettoyer les ressources

Si vous voulez essayer une autre stratégie de rotation, rotation des utilisateurs en simple, sautez le nettoyage des ressources et accédez à the section called “Rotation utilisateur unique”.

Dans le cas contraire, pour éviter d'éventuels frais et pour supprimer l' EC2 instance ayant accès à Internet, supprimez les ressources suivantes que vous avez créées dans ce didacticiel ainsi que leurs prérequis :

- Instance de base de données Amazon RDS. Pour obtenir des instructions, consultez la section Supprimer une instance DB dans le Guide de l'utilisateur Amazon RDS.
- EC2 Instance Amazon. Pour obtenir des instructions, consultez la section Résiliation d'une instance dans le guide de EC2 l'utilisateur Amazon.
- Secret `SecretsManagerTutorialAppuser` de Secrets Manager. Pour obtenir des instructions, consultez the section called “Suppression d'un secret”.
- Point de terminaison Secrets Manager. Pour plus d'informations, consultez Supprimer un point de terminaison d'un VPC dans le Guide AWS PrivateLink .
- Point de terminaison d'un VPC. Pour obtenir plus d’instructions, consultez la section Supprimer votre VPC dans le Guide AWS PrivateLink .

Étapes suivantes

- Découvrez comment récupérer les secrets de vos applications.
- Découvrez d'autres calendriers de rotation.

Configurer la rotation utilisateur unique pour AWS Secrets Manager

Dans ce didacticiel, vous apprendrez comment configurer la rotation d'un utilisateur unique pour un secret contenant les informations d'identification d'administrateur de base de données. La rotation utilisateur unique est une stratégie de rotation dans laquelle Secrets Manager met à jour les informations d'identification d'un utilisateur dans le secret et dans la base de données. Pour de plus amples informations, veuillez consulter the section called "Utilisateur unique".

Une fois le didacticiel terminé, nous vous recommandons d'en nettoyer les ressources. Ne les utilisez pas dans un environnement de production.

La rotation de Secrets Manager utilise une AWS Lambda fonction pour mettre à jour le secret et la base de données. Pour plus d'informations sur les coûts d'utilisation d'une fonction Lambda, consultez Tarification.

Table des matières

- Autorisations
- Prérequis
- Étape 1 : créer un utilisateur de base de données Amazon RDS
- Étape 2 : créez un secret pour les informations d'identification de l'utilisateur
- Étape 3 : tester le mot de passe ayant subi la rotation
- Étape 4 : Nettoyer les ressources
- Étapes suivantes

Autorisations

Pour les prérequis du tutoriel, vous devez disposer d'autorisations administratives sur votre Compte AWS. Dans un contexte de production, une bonne pratique est d'utiliser différents rôles pour chacune des étapes. Par exemple, un rôle doté d'autorisations d'administrateur de base de

données créerait la base de données Amazon RDS, et un rôle doté d'autorisations d'administrateur réseau configurerait le VPC et les groupes de sécurité. Pour les étapes du tutoriel, nous vous recommandons de continuer à utiliser la même identité.

Pour plus d'informations sur la façon de configurer les autorisations dans un environnement de production, consultez the section called "Authentification et contrôle d'accès".

Prérequis

La condition préalable à ce tutoriel est : the section called "Rotation des utilisateurs en alternance". Ne nettoyez pas les ressources à la fin du premier tutoriel. Après ce didacticiel, vous disposez d'un environnement réaliste avec une base de données Amazon RDS et un secret Secrets Manager qui contient des informations d'identification admin pour la base de données. Vous disposez également d'un second secret qui contient les informations d'identification d'un utilisateur de base de données, mais vous n'utiliserez pas ce secret dans ce didacticiel.

Vous disposez également d'une connexion configurée dans MySQL Workbench pour vous connecter à la base de données avec les informations d'identification d'administrateur.

Étape 1 : créer un utilisateur de base de données Amazon RDS

Tout d'abord, vous avez besoin d'un utilisateur dont les informations d'identification seront stockées dans le secret. Pour créer l'utilisateur, connectez-vous à la base de données Amazon RDS à l'aide des informations d'identification d'administrateur stockées dans un secret. Pour des raisons de simplicité, dans le didacticiel, vous allez créer un utilisateur avec des autorisations complètes sur une base de données. Dans un environnement de production, ce n'est pas courant et nous vous recommandons de respecter le principe du moindre privilège.

Pour récupérer le mot de passe administrateur

1. Dans la console Amazon RDS, accédez à votre base de données.
2. Dans l'onglet Configuration, sous Master Credentials ARN, choisissez Gérer dans Secrets Manager.

   La console Secrets Manager s'ouvre.
3. Sur la page de détails de votre secret, sélectionnez Retrieve secret value (Récupérer la valeur du secret).
4. Le mot de passe apparaît dans la section Valeur secrète.

Pour créer un utilisateur de base de données

1. Dans MySQL Workbench, cliquez avec le bouton droit sur la connexion, SecretsManagerTutorialpuis choisissez Modifier la connexion.
2. Dans la boite de dialogue Manage Server Connections (Gérer les connexions aux serveurs), pour Username (Nom d'utilisateur), saisissez **admin**, puis choisissez Close (Fermer).
3. De retour dans MySQL Workbench, choisissez la connexion SecretsManagerTutorial.
4. Saisissez le mot de passe administrateur que vous avez récupéré dans le secret.
5. Dans le MySQL Workbench, dans la fenêtre Query (Requête), saisissez les commandes suivantes (y compris un mot de passe fort), puis choisissez Execute (Exécuter). La fonction de rotation teste le secret mis à jour à l'aide de SELECT. **dbuser** Ils doivent donc avoir ce privilège au minimum.

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

   Dans la fenêtre Output (Sortie), vous voyez que les commandes sont réussies.

Étape 2 : créez un secret pour les informations d'identification de l'utilisateur

Ensuite, vous créez un secret pour stocker les informations d'identification de l'utilisateur que vous venez de créer, et vous activez la rotation automatique, y compris une rotation immédiate. Secrets Manager fait pivoter le secret, ce qui signifie que le mot de passe est généré par programmation. Aucun humain n'a vu ce nouveau mot de passe. Le fait que la rotation commence immédiatement peut également vous aider à déterminer si la rotation est correctement configurée.

1. Ouvrez la console Secrets Manager à l'adresse https://console.aws.amazon.com/secretsmanager/.
2. Choisissez Store a new secret (Stocker un nouveau secret).
3. Sur la page Choose secret type (Choisir un type de secret), procédez comme suit :

   a. Pour Secret type (Type de secret), choisissez Credentials for Amazon RDS database (Informations d'identification pour une base de données Amazon RDS).

   b. Pour Credentials (Informations d'identification), saisissez le nom d'utilisateur **dbuser** et le mot de passe que vous avez entrés pour l'utilisateur de base de données que vous avez créé à l'aide de MySQL Workbench.

c. Pour Database (Base de données), choisissez secretsmanagertutorialdb.

d. Choisissez Next (Suivant).

4. Sur la page Configure secret (Configurer le secret), pour Secret name (Nom du secret), saisissez **`SecretsManagerTutorialDbuser`** puis choisissez Next (Suivant).

5. Sur la page Configure rotation (Configuration de la rotation), procédez comme suit :

   a. Activez Automatic rotation (Rotation automatique).

   b. Pour Rotation schedule (Calendrier de rotation), définissez un calendrier de Days (Jours) : **2** Jours avec une Duration (Durée) :**2h**. Gardez Rotate immediately (Rotation immédiate) sélectionné.

   c. Pour Rotation function (Fonction de rotation), choisissez Create a rotation function (Créer une fonction de rotation), puis pour le nom de la fonction, saisissez **`tutorial-single-user-rotation`**.

   d. Pour Stratégie de rotation, choisissez Utilisateur unique.

   e. Choisissez Next (Suivant).

6. Sur la page Review (Vérification), choisissez Store (Stocker).

   Secrets Manager revient à la page des détails du secret. En haut de la page, vous pouvez voir l'état de la configuration de rotation. Secrets Manager permet CloudFormation de créer des ressources telles que la fonction de rotation Lambda et un rôle d'exécution qui exécute la fonction Lambda. Lorsque vous CloudFormation avez terminé, la bannière devient « Secret », dont la rotation est planifiée. La première rotation est configurée.

Étape 3 : tester le mot de passe ayant subi la rotation

Après la première rotation du secret, qui peut prendre quelques secondes, vous pouvez vérifier que le secret contient toujours des informations d'identification valides. Le mot de passe dans le secret a changé par rapport aux informations d'identification d'origine.

Pour récupérer le nouveau mot de passe du secret

1. Ouvrez la console Secrets Manager à l'adresse https://console.aws.amazon.com/secretsmanager/.

2. Choisissez Secrets, puis choisissez le secret **`SecretsManagerTutorialDbuser`**.

3. Sur la page Secret details (Détails secrets), faites défiler l'écran vers le bas et choisissez Retrieve secret value (Récupération d'une valeur de secret).
4. Dans le tableau Key/value (Clé/Valeur), copiez la Secret value (Valeur de secret) pour **`password`**.

Pour tester les informations d'identification

1. Dans MySQL Workbench, cliquez avec le bouton droit sur la connexion, SecretsManagerTutorialpuis choisissez Modifier la connexion.
2. Dans la boite de dialogue Manage Server Connections (Gérer les connexions aux serveurs), pour Username (Nom d'utilisateur), saisissez **`dbuser`**, puis choisissez Close (Fermer).
3. De retour dans MySQL Workbench, choisissez la connexion SecretsManagerTutorial.
4. Dans la boite de dialogue Open SSH Connection (Connexion SSH ouverte), pour Password (Mot de passe), collez le mot de passe que vous avez récupéré dans le secret, puis choisissez OK..

   Si les informations d'identification sont valides, MySQL Workbench s'ouvre sur la page de conception de la base de données.

Étape 4 : Nettoyer les ressources

Pour éviter des frais potentiels, supprimez le secret que vous avez créé dans ce tutoriel. Pour obtenir des instructions, consultez the section called “Suppression d'un secret”.

Pour nettoyer les ressources créées dans le didacticiel précédent, consultez the section called “Étape 4 : Nettoyer les ressources”.

Étapes suivantes

- Découvrez comment récupérer les secrets de vos applications. Voir Obtenez des secrets.
- Découvrez d'autres calendriers de rotation. Voir the section called “Horaires de rotation”.

Créez un AWS Secrets Manager secret

Un secret peut être un mot de passe, un ensemble d'informations d'identification telles qu'un nom d'utilisateur et un mot de passe, un OAuth jeton ou d'autres informations secrètes que vous stockez sous une forme chiffrée dans Secrets Manager.

> **Tip**
>
> Pour gérer les informations d'identification des utilisateurs pour Amazon RDS et Amazon Redshift, nous vous recommandons d'utiliser des secrets gérés. Vous créez le secret géré à l'aide du service de gestion, puis vous pouvez utiliser la rotation gérée.

Lorsque vous utilisez la console pour stocker les informations d'identification de base de données pour une base de données source répliquée vers d'autres régions, le secret contient des informations de connexion pour la base de données source. Si vous répliquez ensuite le secret, les réplicas sont des copies du secret source et contiennent les mêmes informations de connexion. Vous pouvez ajouter des paires clé-valeur supplémentaires au secret pour obtenir des informations de connexion régionale.

Pour créer un secret, vous avez besoin des autorisations accordées par la stratégie SecretsManagerReadWrite gérée.

Secrets Manager génère une entrée de CloudTrail journal lorsque vous créez un secret. Pour de plus amples informations, veuillez consulter the section called “Connectez-vous avec AWS CloudTrail ”.

Pour créer des secrets (console)

1. Ouvrez la console Secrets Manager à l'adresse https://console.aws.amazon.com/secretsmanager/.
2. Choisissez Store a new secret (Stocker un nouveau secret).
3. Sur la page Choose secret type (Choisir un type de secret), procédez comme suit :

    a. Pour Secret type (Type de secret), effectuez l'une des opérations suivantes :

    - Pour stocker les informations d'identification de base de données, choisissez le type d'informations d'identification de base de données à stocker. Choisissez ensuite la base de données, puis entrez les informations d'identification.

- Pour stocker des clés API, des jetons d'accès et des informations d'identification qui ne sont pas destinées aux bases de données, choisissez Autre type de secret.

  Dans Paires clé/valeur, vous devez soit saisir votre secret dans les paires Clé/valeur JSON, soit choisir l'onglet Plaintext et saisir le secret dans n'importe quel format. Vous pouvez stocker jusqu'à 65 536 octets dans le secret. Voici quelques exemples :

  API key

  Entrez sous la forme de paires clé-valeur :

  `ClientID` : *`my_client_id`*

  `ClientSecret` : *`wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`*

  OAuth token

  Entrez en texte brut :

  `AKIAI44QH8DHBEXAMPLE`

  Digital certificate

  Entrez en texte brut :

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

  Private key

  Entrez en texte brut :

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

b. Pour Clé de chiffrement, choisissez la AWS KMS key que Secrets Manager utilise dans le chiffrement de la valeur du secret. Pour de plus amples informations, veuillez consulter Chiffrement et déchiffrement de secret.

- Dans la plupart des cas, choisissez aws/secretsmanager afin d'utiliser la Clé gérée par AWS pour Secrets Manager. L'utilisation de cette clé n'entraîne aucun coût.

- Si vous avez besoin d'accéder au secret à partir d'un autre Compte AWS, ou si vous souhaitez utiliser votre propre clé KMS afin de pouvoir la faire pivoter ou lui appliquer une politique de clé, choisissez une clé gérée par le client dans la liste ou choisissez Add new key (Ajout d'une nouvelle clé) pour en créer une. Pour plus d'informations sur les coûts d'utilisation d'une clé gérée par le client, consultez Tarification.

  Vous devez disposer de the section called "Autorisations pour la clé KMS". Pour plus d'informations sur l'accès intercompte, consultez the section called "Accès intercomptes".

c. Choisissez Suivant.

4. Sur la page Configure secret (Configurer le secret), procédez comme suit :

   a. Saisissez un Secret name (Nom de secret) descriptif et une Description. Les noms de secrets peuvent contenir de 1 à 512 caractères alphanumériques et /_+ =.@-.

   b. (Facultatif) Dans la section Tags (Balises), vous pouvez ajouter une ou plusieurs balises à votre secret. Pour les stratégies de balisage, consultez the section called "Étiqueter les secrets ". Ne stockez pas les informations sensibles dans des balises car elles ne sont pas chiffrées.

   c. (Facultatif) Dans Resource permissions (Permission de la ressource), pour ajouter une stratégie de ressources à votre secret, choisissez Edit permissions (Modification des autorisations). Pour de plus amples informations, veuillez consulter the section called "Politiques basées sur les ressources".

   d. (Facultatif) Dans Replicate secret (Réplication du secret), pour répliquer votre secret vers un autre Région AWS, choisissez Replicate secret (Répliquer le secret). Vous pouvez reproduire votre secret maintenant ou revenir et le répliquer ultérieurement. Pour de plus amples informations, veuillez consulter Réplication multirégionale.

   e. Choisissez Suivant.

5. (Facultatif) Dans la page Configure rotation (Configurer la rotation), vous pouvez activer la rotation automatique. Vous pouvez également garder la rotation désactivée pour l'instant, puis l'activer plus tard. Pour de plus amples informations, veuillez consulter Rotation des secrets. Choisissez Suivant.

6. Dans la page Review (Révision), passez en revue vos paramètres, puis choisissez Store (Stocker).

   Secrets Manager revient à la liste des secrets. Si votre nouveau secret n'apparaît pas, choisissez le bouton d'actualisation.

AWS CLI

Lorsque saisissez des commandes dans un shell de commande, il existe un risque d'accès à l'historique des commandes ou aux paramètres de vos commandes. Consultez the section called "Réduisez les risques liés AWS CLI à l'utilisation du pour stocker vos AWS Secrets Manager secrets".

Example Création d'un secret à partir des informations d'identification de base de données d'un fichier JSON

L'exemple suivant `create-secret` crée un secret à partir des informations d'identification d'un fichier. Pour de plus amples informations, consultez la section sur le Chargement de AWS CLI paramètres (français non garanti) dans le fichier Guide de AWS CLI l'utilisateur.

Pour que Secrets Manager puisse réaliser une rotation du secret, vous devez vous assurer que JSON correspond à Structure JSON d'un secret.

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

Contenus de mycreds.json :

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

Example Créer un secret

L'exemple suivant `create-secret` crée un secret avec deux paires clé-valeur.

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

Example Créer un secret

L'`create-secret`exemple suivant crée un secret avec deux balises.

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

AWS SDK

Pour créer un secret à l'aide de l'un des AWS SDKs, utilisez l'`CreateSecret`action. Pour de plus amples informations, veuillez consulter the section called "AWS SDKs".

Que contient le secret d'un Secrets Manager ?

Dans Secrets Manager, un secret se compose d'informations secrètes, de la valeur de secret et de métadonnées sur le secret. Une valeur de secret peut être une chaîne ou binaire.

Pour stocker plusieurs valeurs de chaîne dans un secret, nous vous recommandons d'utiliser une chaîne de texte JSON avec des paires clé-valeur, par exemple :

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

Pour les secrets de base de données, si vous souhaitez activer la rotation automatique, le secret doit contenir les informations de connexion à la base de données dans la structure JSON appropriée. Pour de plus amples informations, veuillez consulter the section called "Structure JSON d'un secret".

Métadonnées

Les métadonnées d'un secret comprennent :

- Un Amazon Resource Name (ARN) avec le format suivant :

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

  Secrets Manager inclut six caractères aléatoires à la fin du nom du secret pour garantir que l'ARN secret est unique. Si le secret d'origine est supprimé, puis qu'un nouveau secret est créé avec le même nom, les deux secrets sont différents en ARNs raison de ces caractères. Les utilisateurs ayant accès à l'ancien secret n'ont pas automatiquement accès au nouveau secret car il ARNs est différent.

- Le nom du secret, une description, une politique de ressources et des balises.
- L'ARN d'une clé de chiffrement, AWS KMS key que Secrets Manager utilise pour chiffrer et déchiffrer la valeur secrète. Secrets Manager stocke toujours le texte du secret sous une forme chiffrée et chiffre le secret en transit. Voir the section called "Chiffrement et déchiffrement de secret".
- Des informations sur la façon d'effectuer une rotation du secret, dans le cas où vous configurez la rotation. Voir Rotation des secrets.

Secrets Manager utilise des politiques d'autorisation IAM pour s'assurer que seuls les utilisateurs autorisés peuvent accéder à un secret ou le modifier. Voir Authentification et contrôle d'accès pour AWS Secrets Manager.

Un secret possède des versions qui contiennent des copies de la valeur secrète cryptée. Lorsque vous modifiez la valeur du secret ou que ce dernier est tourné, Secrets Manager crée une nouvelle version. Voir the section called "Versions secrètes".

Vous pouvez utiliser un secret sur plusieurs Régions AWS en le répliquant. Lorsque vous répliquez un secret, vous créez une copie du secret primaire appelé secret de réplica ou de l'original. Le secret de réplica reste lié au secret primaire. Voir Réplication multirégionale.

Voir Gérez les secrets.

Versions secrètes

Un secret possède des versions qui contiennent des copies de la valeur secrète cryptée. Lorsque vous modifiez la valeur du secret ou que ce dernier est tourné, Secrets Manager crée une nouvelle version.

Secrets Manager ne stocke pas d'historique linéaire des secrets avec les versions. Au lieu de cela, il garde une trace de trois versions spécifiques en les étiquetant :

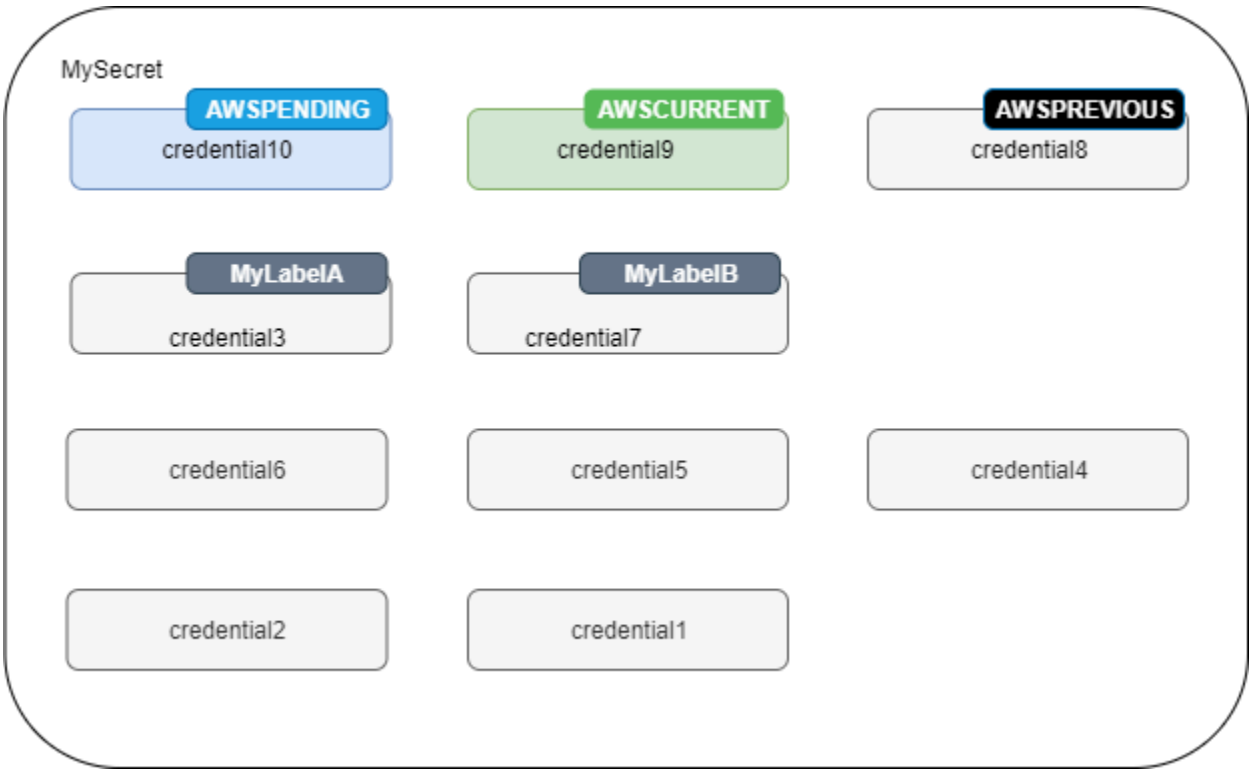
- La version actuelle — `AWSCURRENT`
- La version précédente — `AWSPREVIOUS`
- La version en attente (pendant la rotation) — `AWSPENDING`

Un secret possède toujours une version étiquetée `AWSCURRENT` et Secrets Manager renvoie cette version par défaut lorsque vous récupérez la valeur du secret.

Vous pouvez également étiqueter les versions avec vos propres étiquettes `update-secret-version-stage`en appelant le AWS CLI. Vous pouvez attacher jusqu'à 20 étiquettes aux versions d'un secret. Deux versions d'un secret ne peuvent pas avoir la même étiquette intermédiaire. Les versions peuvent avoir plusieurs étiquettes.

Secrets Manager ne supprime jamais les versions étiquetées, mais les versions non étiquetées sont considérées comme obsolètes. Secrets Manager supprime les versions obsolètes lorsqu'il y en a plus de 100. Secrets Manager ne supprime pas les versions créées il y a moins de 24 heures.

La figure suivante montre un secret qui a AWS étiqueté des versions et des versions étiquetées par le client. Les versions sans étiquette sont considérées comme obsolètes et seront supprimées ultérieurement par Secrets Manager.



Structure JSON des AWS Secrets Manager secrets

Vous pouvez stocker n'importe quel texte ou binaire dans un secret de Secrets Manager jusqu'à une taille maximale de 65 536 octets.

Si vous l'utilisezthe section called "Rotation par fonction Lambda", un secret doit contenir des champs JSON spécifiques attendus par la fonction de rotation. Par exemple, pour un secret contenant des informations d'identification de base de données, la fonction de rotation se connecte à la base de données pour mettre à jour les informations d'identification. Le secret doit donc contenir les informations de connexion à la base de données.

Si vous utilisez la console pour modifier la rotation d'un secret de base de données, celui-ci doit contenir des paires clé-valeur JSON spécifiques identifiant la base de données. Secrets Manager utilise ces champs pour interroger la base de données afin de trouver le VPC approprié dans lequel stocker une fonction de rotation.

Les noms de clés JSON distinguent les majuscules et minuscules.

Rubriques

- Informations d'identification Amazon RDS et Aurora
- Informations d'identification Amazon Redshift

- Informations d'identification Amazon Redshift Serverless
- Informations d'identification Amazon DocumentDB
- Structure secrète d'Amazon Timestream pour InfluxDB
- ElastiCache Informations d'identification Amazon
- Informations d'identification Active Directory

Informations d'identification Amazon RDS et Aurora

Pour utiliser les modèles de fonctions de rotation fournis par Secrets Manager, utilisez la structure JSON suivante. Vous pouvez ajouter d'autres paires clé/valeur, par exemple pour contenir des informations de connexion pour les répliques de bases de données dans d'autres régions.

DB2

Pour les instances Amazon RDS Db2, étant donné que les utilisateurs ne peuvent pas modifier leurs propres mots de passe, vous devez fournir des informations d'identification d'administrateur dans un secret distinct.

```
{
  "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 "Utilisateurs en alternance".>",
  "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 "Utilisateurs en alternance".>",
  "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 "Utilisateurs en alternance".>",
  "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 "Utilisateurs en alternance".>",
  "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 "Utilisateurs en alternance".>",
  "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.>"
}
```

Informations d'identification Amazon Redshift

Pour utiliser les modèles de fonctions de rotation fournis par Secrets Manager, utilisez la structure JSON suivante. Vous pouvez ajouter d'autres paires clé/valeur, par exemple pour contenir des informations de connexion pour les répliques de bases de données dans d'autres régions.

```
{
```

```
  "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 "Utilisateurs en alternance".>"
}
```

Informations d'identification Amazon Redshift Serverless

Pour utiliser les modèles de fonctions de rotation fournis par Secrets Manager, utilisez la structure JSON suivante. Vous pouvez ajouter d'autres paires clé/valeur, par exemple pour contenir des informations de connexion pour les répliques de bases de données dans d'autres régions.

```
{
  "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 "Utilisateurs en alternance".>"
}
```

Informations d'identification Amazon DocumentDB

Pour utiliser les modèles de fonctions de rotation fournis par Secrets Manager, utilisez la structure JSON suivante. Vous pouvez ajouter d'autres paires clé/valeur, par exemple pour contenir des informations de connexion pour les répliques de bases de données dans d'autres régions.

```
{
  "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 "Utilisateurs en alternance".>",
  "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.>"
}
```

Structure secrète d'Amazon Timestream pour InfluxDB

Pour faire pivoter les secrets de Timestream, vous pouvez utiliser les modèles de the section called "Amazon Timestream pour InfluxDB" rotation.

Pour plus d'informations, consultez Comment Amazon Timestream pour InfluxDB utilise les secrets dans le manuel du développeur Amazon Timestream.

Les secrets Timestream doivent être dans la bonne structure JSON pour pouvoir utiliser les modèles de rotation. Pour plus d'informations, consultez What's in the secret du manuel Amazon Timestream Developer Guide.

ElastiCache Informations d'identification Amazon

L'exemple suivant montre la structure JSON d'un secret qui stocke les ElastiCache informations d'identification.

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

Pour plus d'informations, consultez Rotation automatique des mots de passe pour les utilisateurs dans le guide de ElastiCache l'utilisateur Amazon.

Informations d'identification Active Directory

AWS Directory Service utilise des secrets pour stocker les informations d'identification Active Directory. Pour plus d'informations, consultez Joindre sans difficulté une instance Amazon EC2 Linux

à votre annuaire Active Directory géré dans le guide d'AWS Directory Service administration. Une jointure de domaine fluide nécessite les noms de clé présentés dans les exemples suivants. Si vous n'utilisez pas une jointure de domaine fluide, vous pouvez modifier le nom des clés du secret à l'aide de variables d'environnement, comme décrit dans le code du modèle de fonction de rotation.

Pour faire pivoter les secrets Active Directory, vous pouvez utiliser les modèles de rotation Active Directory.

Active Directory credential

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

Si vous souhaitez modifier le secret, vous devez inclure l'ID du répertoire de domaines.

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

Si le secret est utilisé conjointement avec un secret contenant un keytab, vous incluez le secret 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

Pour plus d'informations sur l'utilisation de fichiers keytab pour s'authentifier auprès de comptes Active Directory sur Amazon EC2, consultez Déploiement et configuration de l'authentification Active Directory avec SQL Server 2017 sur Amazon Linux 2.

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

Gérez les secrets avec AWS Secrets Manager

Rubriques

- Mettre à jour la valeur d'un AWS Secrets Manager secret
- Générez un mot de passe avec Secrets Manager
- Rétablirez un secret à une version précédente
- Modifier la clé de chiffrement d'un AWS Secrets Manager secret
- Modifier un AWS Secrets Manager secret
- Trouvez des secrets dans AWS Secrets Manager
- Supprimer un AWS Secrets Manager secret
- Restaurer un AWS Secrets Manager secret
- Marquer des secrets dans AWS Secrets Manager

Mettre à jour la valeur d'un AWS Secrets Manager secret

Pour mettre à jour la valeur de votre secret, vous pouvez utiliser la console, la CLI ou un kit SDK. Lorsque vous mettez à jour la valeur du secret, Secrets Manager crée une nouvelle version du secret avec l'étiquette intermédiaire `AWSCURRENT`. Vous pouvez toujours accéder à l'ancienne version avec l'étiquette `AWSPREVIOUS`. Vous pouvez également ajouter vos propres étiquettes. Pour plus d'informations, consultez la section relative à la gestion des versions Secrets Manager.

Pour mettre à jour la valeur d'un secret (console)

1. Ouvrez la console Secrets Manager à l'adresse https://console.aws.amazon.com/secretsmanager/.
2. Dans la liste des secrets, choisissez le secret.
3. Sur la page des détails du secret, sous l'onglet Présentation, dans la section Valeur du secret), choisissez Récupérer la valeur du secret, puis choisissez Modifier.

AWS CLI

Pour mettre à jour la valeur d'un secret (AWS CLI)

- Lorsque saisissez des commandes dans un shell de commande, il existe un risque d'accès à l'historique des commandes ou aux paramètres de vos commandes. Consultez the section called "Réduisez les risques liés AWS CLI à l'utilisation du pour stocker vos AWS Secrets Manager secrets".

  Ce qui suit `put-secret-value` crée une nouvelle version d'un secret avec deux paires clé-valeur.

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

  La commande `put-secret-value` suivante crée une nouvelle version avec une étiquette intermédiaire personnalisée. La nouvelle version aura les étiquettes `MyLabel` et `AWSCURRENT`.

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

AWS SDK

Nous vous recommandons d'éviter d'appeler `PutSecretValue` ou `UpdateSecret` à un rythme soutenu, soit plus d'une fois toutes les 10 minutes. Lorsque vous appelez `PutSecretValue` ou `UpdateSecret` pour mettre à jour la valeur sécrète, Secrets Manager crée une nouvelle version du secret. Secrets Manager supprime les versions non marquées lorsqu'il y en a plus de 100, mais il ne supprime pas les versions créées il y a moins de 24 heures. Si vous mettez à jour la valeur secrète plus d'une fois toutes les 10 minutes, vous créez plus de versions que Secrets Manager ne peut en supprimer, et vous atteindrez ainsi le quota pour les versions secrètes.

Pour mettre à jour la valeur d'un secret, utilisez l'une des actions suivantes : `UpdateSecret` ou `PutSecretValue`. Pour de plus amples informations, veuillez consulter the section called "AWS SDKs".

Générez un mot de passe avec Secrets Manager

Un modèle courant d'utilisation de Secrets Manager consiste à générer un mot de passe dans Secrets Manager, puis à utiliser ce mot de passe dans votre base de données ou votre service. Pour ce faire, utilisez les méthodes suivantes :

- AWS CloudFormation — VoirAWS CloudFormation.
- AWS CLI — Voir `get-random-password`.
- AWS SDKs — Voir `GetRandomPassword`.

Rétablirez un secret à une version précédente

Vous pouvez rétablir une version précédente d'un secret en déplaçant les étiquettes associées aux versions secrètes à l'aide du AWS CLI. Pour plus d'informations sur le mode de stockage des versions de Secrets Manager, consultez la sectionthe section called “Versions secrètes”.

L'`update-secret-version-stage`exemple suivant déplace l' AWSCURRENT étiquette intermédiaire vers la version précédente d'un secret, ce qui ramène le secret à la version précédente. Pour trouver l'ID de la version précédente, utilisez `list-secret-version-ids`ou consultez les versions dans la console Secrets Manager.

Dans cet exemple, la version portant le AWSCURRENT libellé est a1b2c3d4-5678-90ab-cdef- et la version portant le libellé est a1b2c3d4-5678-90ab-cdef-. EXAMPLE11111 AWSPREVIOUS EXAMPLE22222 Dans cet exemple, vous déplacez l' AWSCURRENT étiquette de la version 11111 vers la version 22222. Comme l' AWSCURRENT étiquette est supprimée d'une version, elle est `update-secret-version-stage` automatiquement déplacée AWSPREVIOUS vers cette version (11111). L'effet est que les AWSPREVIOUS versions AWSCURRENT et sont échangées.

```
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
```

Modifier la clé de chiffrement d'un AWS Secrets Manager secret

Secrets Manager utilise le chiffrement des enveloppes avec des AWS KMS clés et des clés de données pour protéger chaque valeur secrète. Pour chaque secret, vous pouvez sélectionner la clé

KMS à utiliser. Vous pouvez utiliser la Clé gérée par AWS aws/secretsmanager, ou vous pouvez utiliser une clé gérée par le client. Dans la plupart des cas, nous vous recommandons de l'utiliseraws/secretsmanager, et son utilisation est gratuite. Si vous devez accéder au secret depuis un autre Compte AWS utilisateur, ou si vous souhaitez utiliser votre propre clé KMS afin de pouvoir la faire pivoter ou lui appliquer une politique clé, utilisez un clé gérée par le client. Vous devez disposer de the section called "Autorisations pour la clé KMS". Pour plus d'informations sur les coûts d'utilisation d'une clé gérée par le client, consultez Tarification.

Vous pouvez modifier la clé de chiffrement de votre secret. Par exemple, si vous souhaitez accéder au secret depuis un autre compte et que le secret est actuellement chiffré à l'aide de la clé AWS géréeaws/secretsmanager, vous pouvez passer à un clé gérée par le client.

> **Tip**
>
> Si vous souhaitez faire pivoter votre clé gérée par le client, nous vous recommandons d'utiliser la rotation AWS KMS automatique des touches. Pour plus d'informations, reportez-vous à la section Rotation AWS KMS des touches.

Lorsque vous modifiez la clé de chiffrement, Secrets Manager le chiffre à nouveau `AWSCURRENT` et `AWSPENDING` les `AWSPREVIOUS` versions utilisant la nouvelle clé. Pour ne pas vous empêcher d'accéder au secret, Secrets Manager crypte toutes les versions existantes avec la clé précédente. Cela signifie que vous pouvez déchiffrer `AWSCURRENT` `AWSPENDING` les `AWSPREVIOUS` versions avec la clé précédente ou la nouvelle clé. Si vous n'êtes pas `kms:Decrypt` autorisé à accéder à la clé précédente, lorsque vous modifiez la clé de chiffrement, Secrets Manager ne peut pas déchiffrer les versions secrètes pour les rechiffrer. Dans ce cas, les versions existantes ne sont pas rechiffrées.

Pour qu'il ne `AWSCURRENT` puisse être déchiffré que par la nouvelle clé de chiffrement, créez une nouvelle version du secret avec la nouvelle clé. Ensuite, pour pouvoir déchiffrer la version `AWSCURRENT` secrète, vous devez avoir l'autorisation d'utiliser la nouvelle clé.

Si vous désactivez la clé de chiffrement précédente, vous ne pouvez déchiffrer aucune version de secret à l'exception de `AWSCURRENT`, `AWSPENDING` et `AWSPREVIOUS`. Si vous avez d'autres versions de secret étiquetées auxquelles vous souhaitez conserver l'accès, vous devez recréer ces versions avec la nouvelle clé de chiffrement à l'aide de l'the section called "AWS CLI".

Pour modifier la clé de chiffrement d'un secret (console)

1. Ouvrez la console Secrets Manager à l'adresse https://console.aws.amazon.com/secretsmanager/.
2. Dans la liste des secrets, choisissez le secret.
3. Sur la page des détails du secret, dans la section Détails des secrets, cliquez sur Actions, puis sur Modifier la clé de chiffrement.

AWS CLI

Si vous remplacez la clé de chiffrement par un secret, puis que vous désactivez la clé de chiffrement précédente, vous ne pouvez déchiffrer aucune version de secret à l'exception de `AWSCURRENT`, `AWSPENDING` et `AWSPREVIOUS`. Si vous avez d'autres versions de secret étiquetées auxquelles vous souhaitez conserver l'accès, vous devez recréer ces versions avec la nouvelle clé de chiffrement à l'aide de l'the section called "AWS CLI".

Pour modifier la clé de chiffrement d'un secret (AWS CLI)

1. L'exemple suivant `update-secret` met à jour la clé KMS utilisée pour chiffrer la valeur secrète. Les clés KMS doivent être situées dans la même région que le secret.

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

2. (Facultatif) Si vous avez des versions de secret dotées d'étiquettes personnalisées, vous devez recréer ces versions pour les rechiffrer à l'aide de la nouvelle clé.

   Lorsque saisissez des commandes dans un shell de commande, il existe un risque d'accès à l'historique des commandes ou aux paramètres de vos commandes. Voir the section called "Réduisez les risques liés AWS CLI à l'utilisation du pour stocker vos AWS Secrets Manager secrets".

   a. Obtenez la valeur de la version de secret.

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

Notez la valeur de secret.

b. Créez une nouvelle version avec cette valeur.

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

Modifier un AWS Secrets Manager secret

Vous pouvez modifier les métadonnées d'un secret après sa création, en fonction de la personne qui a créé le secret. Pour les secrets créés par d'autres services, vous devrez peut-être utiliser l'autre service pour le mettre à jour ou le faire tourner.

Pour déterminer qui gère un secret, vous pouvez vérifier le nom du secret. Les secrets gérés par d'autres services sont préfixés par l'ID de ce service. Ou, dans le AWS CLI, appelez describe-secret, puis passez en revue le champ. `OwningService` Pour de plus amples informations, veuillez consulter Secrets gérés par d'autres services.

Pour les secrets que vous gérez, vous pouvez modifier la description, la politique basée sur les ressources, la clé de chiffrement et les balises. Vous pouvez également modifier la valeur du secret chiffré, mais nous vous conseillons d'utiliser la rotation pour mettre à jour les valeurs du secret qui contiennent des informations d'identification. La rotation met à jour le secret dans Secrets Manager et les informations d'identification dans la base de données ou le service. Les secrets restent ainsi automatiquement synchronisés afin que, lorsque les clients demandent une valeur de secret, ils récupèrent toujours un ensemble d'informations d'identification actives. Pour de plus amples informations, veuillez consulter Rotation des secrets.

Secrets Manager génère une entrée de CloudTrail journal lorsque vous modifiez un secret. Pour de plus amples informations, veuillez consulter the section called “Connectez-vous avec AWS CloudTrail ”.

Pour mettre à jour un secret que vous gérez (console)

1. Ouvrez la console Secrets Manager à l'adresse https://console.aws.amazon.com/secretsmanager/.
2. Dans la liste des secrets, choisissez le secret.

3. Sur la page des détails du secret, effectuez l'une des actions suivantes :

   Notez que vous ne pouvez pas modifier le nom ou l'ARN d'un secret.

   - Pour mettre à jour la description, dans la section Secrets details (Détails du secret), choisissez Actions, puis Edit description (Modifier la description).
   - Pour mettre à jour la clé de chiffrement, consultez la section the section called "Modification de clé de chiffrement d'un secret".
   - Pour mettre à jour les balises, sous l'onglet Balises, choisissez Modifier les balises. Consultez the section called "Étiqueter les secrets ".
   - Pour mettre à jour la valeur d'un secret, consultez la section the section called "Mise à jour d'une valeur de secret".
   - Pour mettre à jour les autorisations pour votre secret, sous l'onglet Présentation, choisissez Modifier les autorisations. Consultez the section called "Politiques basées sur les ressources".
   - Pour mettre à jour la rotation de votre secret, sous l'onglet Rotation, choisissez Modifier la rotation. Consultez Rotation des secrets.
   - Pour répliquer votre secret vers d'autres régions, consultez Réplication multirégionale.
   - Si votre secret contient des réplicas, vous pouvez modifier la clé de chiffrement d'un réplica. Sous l'onglet Réplication, sélectionnez le bouton radio pour la réplica, puis dans le menu Actions, choisissez Editer la clé de chiffrement. Consultez the section called "Chiffrement et déchiffrement de secret".
   - Pour modifier un secret afin qu'il soit géré par un autre service, vous devez recréer le secret dans ce service. Consultez Secrets gérés par d'autres services.

AWS CLI

Example Mettre à jour la description du secret

L'exemple suivant `update-secret` met à jour la description d'un secret.

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

AWS SDK

Nous vous recommandons d'éviter d'appeler `PutSecretValue` ou `UpdateSecret` à un rythme soutenu, soit plus d'une fois toutes les 10 minutes. Lorsque vous appelez `PutSecretValue` ou `UpdateSecret` pour mettre à jour la valeur sécrète, Secrets Manager crée une nouvelle version du secret. Secrets Manager supprime les versions non marquées lorsqu'il y en a plus de 100, mais il ne supprime pas les versions créées il y a moins de 24 heures. Si vous mettez à jour la valeur secrète plus d'une fois toutes les 10 minutes, vous créez plus de versions que Secrets Manager ne peut en supprimer, et vous atteindrez ainsi le quota pour les versions secrètes.

Pour mettre à jour un secret, utilisez l'une des actions suivantes : `UpdateSecret` ou `ReplicateSecretToRegions`. Pour de plus amples informations, veuillez consulter the section called “AWS SDKs”.

Trouvez des secrets dans AWS Secrets Manager

Lorsque vous recherchez des secrets sans filtre, Secrets Manager fait correspondre les mots-clés dans le nom, la description, la clé et la valeur de l'identification du secret. La recherche sans filtre n'est pas sensible à la casse et ignore les caractères spéciaux, tels que l'espace, /, _, =, #, et utilise uniquement les chiffres et les lettres. Lorsque vous effectuez une recherche sans filtre, Secrets Manager analyse la chaîne de recherche pour la convertir en mots distincts. Les mots sont séparés par tout changement de majuscule à minuscule, de lettre à chiffre ou de number/letter ponctuation. Par exemple, saisir le terme de recherche `credsDatabase#892` permet de rechercher `creds`, `Database` et 892 dans le nom, la description, la clé et la valeur d'étiquette.

Secrets Manager génère une entrée de CloudTrail journal lorsque vous listez des secrets. Pour de plus amples informations, veuillez consulter the section called “Connectez-vous avec AWS CloudTrail”.

Secrets Manager est un service régional et seuls les secrets de la région sélectionnée sont retournés.

Filtres de recherche

Si vous n'utilisez aucun filtre, Secrets Manager divise la chaîne de recherche en mots, puis recherche des correspondances dans tous les attributs. Cette recherche ne fait pas la distinction majuscules/minuscules. Par exemple, la recherche de **`My_Secret`** correspond à des secrets avec le mot « mon » ou « secret » dans le nom, la description ou les balises.

Vous pouvez appliquer les filtres suivants à votre recherche :

Nom

Correspond au début des noms de secrets, sensible à la casse. Par exemple, Name: **`Data`** renvoie un secret nommé `DatabaseSecret`, mais pas `databaseSecret` ou `MyData`.

Description

Correspond aux mots dans les descriptions des secrets, non sensibles à la casse. Par exemple, Description: **`My Description`** correspond aux secrets avec les descriptions suivantes :

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

Géré par

Trouve les secrets gérés par des services extérieurs à AWS, par exemple, CyberArk ou HashiCorp.

Posséder un service

Correspond au début du préfixe d'identification du service de gestion, non sensibles à la casse. Par exemple, **`my-ser`** fait correspondre les secrets gérés par les services avec les préfixes `my-serv` et `my-service`. Pour de plus amples informations, veuillez consulter Secrets gérés par d'autres services.

Secrets répliqués

Vous pouvez filtrer les secrets principaux, les secrets répliqués ou les secrets qui ne sont pas répliqués.

Tag Keys (Clés d'identification)

Correspond au début des clés d'identification, sensible à la casse. Par exemple, Tag key: **`Prod`** renvoie les secrets avec l'identification `Production` et `Prod1`, mais pas de secrets avec l'identification `prod` ou `1 Prod`.

Tag Value (Valeur d'identification)

Correspond au début des valeurs d'identification, sensible à la casse. Par exemple, Tag value: **`Prod`** renvoie les secrets avec l'identification `Production` et `Prod1`, mais pas de secrets avec la valeur d'identification `prod` ou `1 Prod`.

AWS CLI

Example Répertorie les secrets de votre compte

L'exemple suivant `list-secrets` permet d'obtenir la liste des secrets de votre compte.

```
aws secretsmanager list-secrets
```

Example Filtrer la liste des secrets de votre compte

L'exemple suivant `list-secrets` permet d'obtenir la liste des secrets de votre compte dont le nom contient l'élément Test. Le filtrage par nom est sensible à la casse.

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

Example Trouvez des secrets gérés par d'autres AWS services

L'exemple suivant `list-secrets` a une liste de secrets gérés par un service. Vous spécifiez le service par ID. Pour de plus amples informations, veuillez consulter Secrets gérés par d'autres services.

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

AWS SDK

Pour trouver des secrets à l'aide de l'un des AWS SDKs, utilisez `ListSecrets`. Pour de plus amples informations, veuillez consulter the section called "AWS SDKs".

Supprimer un AWS Secrets Manager secret

En raison de la nature critique des secrets, il est difficile de les supprimer AWS Secrets Manager intentionnellement. Secrets Manager ne supprime pas immédiatement les secrets. Au lieu de cela, Secrets Manager les rend immédiatement indisponibles et leur suppression est programmée après une plage de récupération de sept jours minimum. Tant que la plage de récupération n'est pas

terminée, vous pouvez récupérer un secret que vous avez précédemment supprimé. Il n'y a pas de frais pour les secrets que vous avez marqués pour suppression.

Vous ne pouvez pas supprimer un secret principal s'il est répliqué vers d'autres régions. Supprimez d'abord les réplicas, puis le secret principal. Lorsque vous supprimez un réplica, il est immédiatement supprimé.

Vous ne pouvez pas supprimer directement une version d'un secret. Au lieu de cela, vous supprimez toutes les étiquettes de mise en scène de la version à l'aide du AWS SDK AWS CLI or. Cela marque la version comme obsolète et permet à Secrets Manager de supprimer automatiquement la version en arrière-plan.

Si vous ne savez pas si une application utilise toujours un secret, vous pouvez créer une CloudWatch alarme Amazon pour vous avertir de toute tentative d'accès à un secret pendant la fenêtre de restauration. Pour de plus amples informations, veuillez consulter Surveiller l'accès aux AWS Secrets Manager secrets dont la suppression est prévue.

Pour supprimer un secret, vous devez disposer des autorisations `secretsmanager:ListSecrets` et `secretsmanager:DeleteSecret`.

Secrets Manager génère une entrée de CloudTrail journal lorsque vous supprimez un secret. Pour de plus amples informations, veuillez consulter the section called “Connectez-vous avec AWS CloudTrail ”.

Pour supprimer un secret (console)

1. Ouvrez la console Secrets Manager à l'adresse https://console.aws.amazon.com/secretsmanager/.
2. Dans la liste des secrets, choisissez le nom du secret que vous souhaitez supprimer.
3. Dans la section Secrets details (Détails du secret), choisissez Actions, puis Edit description (Modifier la description).
4. Dans la boîte de dialogue Disable secret and schedule deletion (Désactiver le secret et planifier la suppression), dans Waiting period (Période d'attente), saisissez le nombre de jours d'attente avant que la suppression soit définitive. Secrets Manager attache un champ appelé `DeletionDate` et le définit sur la date et l'heure actuelles, auxquelles est ajouté le nombre de jours spécifié pour la plage de récupération.
5. Choisissez Schedule deletion (Planifier la suppression).

Pour afficher les secrets supprimés

1. Ouvrez la console Secrets Manager à l'adresse https://console.aws.amazon.com/secretsmanager/.
2. Dans la page Secrets, choisissez Preferences (Préférences)(⚙)
3. Dans la boîte de dialogue Préférences, sélectionnez Afficher les secrets désactivés, puis Enregistrer.

Pour supprimer un secret répliqué

1. Ouvrez la console Secrets Manager à l'adresse https://console.aws.amazon.com/secretsmanager/.
2. Choisissez le secret principal.
3. Dans Replicate Secret (Réplication du secret), choisissez le secret de réplica.
4. Dans le menu Actions, choisissez Delete Replica (Supprimer le réplica).

AWS CLI

Example Suppression d'un secret

L'exemple suivant `delete-secret` supprime un secret répliqué. Vous pouvez récupérer le secret `restore-secret`jusqu'à la date et à l'heure indiquées dans le champ de DeletionDate réponse. Pour supprimer un secret répliqué dans d'autres régions, supprimez d'abord ses répliques avec `remove-regions-from-replication`, puis appelez `delete-secret`.

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

Example Suppression immédiate d'un secret

L'exemple suivant `delete-secret` supprime immédiatement le secret, sans plage de récupération. Il n'est pas possible de récupérer ce secret.

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

```
    --force-delete-without-recovery
```

Example Supprimer un secret répliqué

L'exemple suivant `remove-regions-from-replication` supprime un secret répliqué dans eu-west-3. Pour supprimer un secret principal répliqué dans d'autres régions, supprimez d'abord les répliques, puis appelez `delete-secret`.

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

AWS SDK

Pour supprimer un fichier, utilisez la commande `DeleteSecret`. Pour supprimer une version de secret, utilisez la commande `UpdateSecretVersionStage`. Pour supprimer un réplica, utilisez la commande `StopReplicationToReplica`. Pour de plus amples informations, veuillez consulter the section called “AWS SDKs”.

Restaurer un AWS Secrets Manager secret

Secrets Manager considère qu'un secret dont la suppression est programmée est obsolète et n'est plus directement accessible. Au terme de la plage de récupération, Secrets Manager supprime définitivement le secret. Une fois que Secrets Manager supprime le secret, vous ne pouvez pas récupérer ce dernier. Avant la fin de la plage de récupération, vous pouvez récupérer le secret et le rendre à nouveau accessible. Cette opération supprime le champ `DeletionDate`, ce qui annule la suppression définitive programmée.

Pour restaurer un secret et les métadonnées dans la console, vous devez disposer des autorisations `secretsmanager:ListSecrets` et `secretsmanager:RestoreSecret`.

Secrets Manager génère une entrée de CloudTrail journal lorsque vous restaurez un secret. Pour de plus amples informations, veuillez consulter the section called “Connectez-vous avec AWS CloudTrail ”.

Pour restaurer un secret (console)

1. Ouvrez la console Secrets Manager à l'adresse https://console.aws.amazon.com/secretsmanager/.

2. Dans la liste des secrets, sélectionnez le nom du secret que vous souhaitez modifier.

   Si les secrets supprimés n'apparaissent pas dans la liste, sélectionnez Préférences( )
   Dans la boîte de dialogue Préférences, sélectionnez Afficher les secrets désactivés, puis Enregistrer.
3. Dans la page Secret details (Détails du secret), sélectionnez Cancel deletion (Annuler la suppression).
4. Dans la boîte de dialogue Cancel secret deletion (Annuler la suppression du secret), sélectionnez Cancel deletion (Annuler la suppression).

AWS CLI

Example Restauration d'un secret précédemment supprimé

L'exemple suivant `restore-secret` restaure un secret dont la suppression était précédemment planifiée.

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

AWS SDK

Pour restaurer un secret marqué pour suppression, utilisez la commande`RestoreSecret`. Pour de plus amples informations, veuillez consulter the section called “AWS SDKs”.

Marquer des secrets dans AWS Secrets Manager

Dans AWS Secrets Manager, vous pouvez attribuer des métadonnées à vos secrets à l'aide de balises. Une balise est une paire nom-valeur que vous définissez pour un secret. Les balises vous aident à gérer les AWS ressources et à organiser les données, y compris les informations de facturation.

Les balises vous permettent de :

- Gestion, recherche et filtrage des secrets et autres ressources dans votre AWS compte
- Contrôler l'accès aux secrets en fonction de leurs balises jointes
- Suivez et catégorisez les dépenses associées à des secrets ou à des projets spécifiques

Pour plus d'informations sur l'utilisation des balises pour contrôler l'accès, consultezthe section called "Contrôlez l'accès aux secrets à l'aide de balises".

Pour en savoir plus sur les balises de répartition des coûts, veuillez consulter Utilisation des balises de répartition des AWS coûts dans le Guide de AWS Billing l'utilisateur.

Pour plus d'informations sur les quotas de balises et les restrictions d'attribution de noms, consultez Service Quotas pour le balisage dans le Guide de référence AWS général. Les balises sont sensibles à la casse.

Secrets Manager génère une entrée de CloudTrail journal lorsque vous étiquetez un secret ou supprimez son étiquette. Pour de plus amples informations, veuillez consulter the section called "Connectez-vous avec AWS CloudTrail ".

> **Tip**
>
> Utilisez un schéma de balisage cohérent pour toutes vos AWS ressources. Pour connaître les meilleures pratiques, consultez le livre blanc sur les meilleures pratiques en matière de balisage.

Bases des balises

Vous pouvez trouver des secrets par tags dans la console, AWS CLI, et SDKs. AWS fournit également l'outil Resource Groups pour créer une console personnalisée qui consolide et organise

vos ressources en fonction de leurs identifications. Pour rechercher des secrets avec une balise spécifique, consultez the section called "Recherche de secrets".

Vous pouvez utiliser la console Secrets Manager ou AWS CLI l'API Secrets Manager pour :

- Création d'un secret à l'aide d'identifications
- Ajouter des identifications à un secret
- Répertoriez les tags de vos secrets
- Supprimer les identifications d'un secret

Vous pouvez utiliser des identifications pour classer vos secrets par catégorie. Par exemple, vous pouvez classer les secrets par objectif, propriétaire ou environnement. Dans la mesure où vous avez défini la clé et la valeur de chaque balise, vous pouvez créer un ensemble personnalisé de catégories répondant à vos besoins spécifiques. Voici quelques exemples de balises :

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

Suivi des coûts à l'aide d'étiquettes

Vous pouvez utiliser les balises pour classer vos AWS coûts par catégorie et en assurer le suivi. Lorsque vous appliquez des identifications à vos AWS ressources, notamment aux secrets, votre rapport de ventilation des AWS coûts comprend l'utilisation et les coûts regroupés par identifications. Vous pouvez appliquer des balises associées à des catégories métier (telles que les centres de coûts, les noms d'applications ou les propriétaires) pour organiser les coûts relatifs à divers services. Pour plus d'informations, consultez Utilisation des identifications de répartition des coûts pour les rapports de facturation personnalisés dans le Guide de l'utilisateur AWS Billing .

Comprendre les restrictions liées aux balises

Les restrictions suivantes s'appliquent aux balises :

Restrictions de base

- Le nombre maximal de balises par ressource (secret) est de 50.
- Les clés et valeurs d'étiquette sont sensibles à la casse.
- Vous ne pouvez pas modifier ou modifier les balises d'un secret supprimé.

Restrictions relatives aux clés de balise

- Chaque clé de balise doit être unique. Si vous ajoutez une balise avec une clé qui est déjà en cours d'utilisation, la nouvelle balise remplacera la paire clé-valeur existante.
- Vous ne pouvez pas commencer une clé de balise par, `aws:` car ce préfixe est réservé à l'usage de AWS. AWS crée en votre des balises qui commencent par ce préfixe, mais vous ne pouvez ni les modifier ni les supprimer.
- Les clés de balise doivent comporter entre 1 et 128 caractères Unicode.
- Les clés de balise doivent comporter les caractères suivants : lettres Unicode, chiffres, espaces et les caractères spéciaux suivants : `_ . / = + - @`.

Restrictions relatives à la valeur de balise

- Les valeurs de balise doivent comporter entre 0 et 255 caractères Unicode.
- Les valeurs de balise peuvent être vides. Si tel n'est pas le cas, elles doivent être composées des caractères suivants : lettres Unicode, chiffres, espaces et les caractères spéciaux suivants : `_ . / = + - @`.

Balisage des secrets à l'aide de la console Secrets Manager

Vous pouvez gérer les balises associées à vos secrets à l'aide de la console Secrets Manager.

Pour accéder aux fonctionnalités de balisage, procédez comme suit :

1. Ouvrez la console Secrets Manager.
2. Dans la barre de navigation, choisissez votre région préférée.
3. Sur la page Secrets, sélectionnez un secret.

Pour afficher les balises d'un secret

- Sur la page Détails du secret, choisissez l'onglet Tags.

Pour créer des secrets à l'aide d'une balise

- Suivez les étapes de Création de secrets.

Pour ajouter ou changer les balises d'un secret

1. Sur la page Détails du secret, choisissez l'onglet Tags, puis sélectionnez Modifier les tags.
2. Entrez la clé du tag dans le champ Clé. Entrez éventuellement une valeur de balise dans le champ Valeur.
3. Choisissez Enregistrer. La balise nouvelle ou mise à jour apparaît dans la liste des balises.

> **Note**
>
> Si le bouton Enregistrer n'est pas activé, il est possible que la clé ou la valeur du tag ne respecte pas les restrictions du tag. Pour de plus amples informations, veuillez consulter Comprendre les restrictions liées aux balises.

Pour supprimer une balise d'un secret

1. Sur la page Détails du secret, choisissez l'onglet Tags, puis cliquez sur l'icône Supprimer à côté du tag que vous souhaitez supprimer.
2. Choisissez Enregistrer pour confirmer la suppression ou sélectionnez Annuler pour annuler.

Marquez les secrets à l'aide du AWS CLI

AWS CLI exemples

Example Ajouter une balise à un secret

L'exemple suivant `tag-resource` montre comment associer une identification à l'aide d'une syntaxe abrégée.

```
aws secretsmanager tag-resource \
```

```
            --secret-id MyTestSecret \
            --tags Key=FirstTag,Value=FirstValue
```

Example Ajouter plusieurs identifications à un secret

L'exemple suivant `tag-resource` associe deux balises clé-valeur à un secret.

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

Example Supprimer les identifications d'un secret

L'exemple suivant `untag-resource` montre comment supprimer deux identifications d'un secret. Pour chaque identification, la clé et la valeur sont toutes les deux supprimées.

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

Balisage des secrets à l'aide de l'API Secrets Manager

Vous pouvez ajouter, répertorier et supprimer des balises à l'aide de l'API Secrets Manager. Pour obtenir des exemples, consultez la documentation suivante :

- ListSecrets: permet `ListSecrets` de visualiser les balises appliquées à un secret
- TagResource: Ajouter des tags à un secret
- Supprimer les identifications : Supprimer les identifications d'un secret

Balisage des secrets à l'aide du AWS SDK Secrets Manager

Pour modifier les balises de votre secret, utilisez les opérations d'API suivantes :

- ListSecrets: permet `ListSecrets` de visualiser les balises appliquées à un secret
- `TagResource`: Ajouter des tags à un secret
- `UntagResource`: Supprimer les identifications d'un secret

Pour plus d'informations sur l'utilisation du kit SDK, consultez the section called "AWS SDKs".

Reproduisez les AWS Secrets Manager secrets d'une région à l'autre

Vous pouvez reproduire vos secrets en plusieurs exemplaires Régions AWS pour prendre en charge des applications réparties dans ces régions afin de répondre aux exigences régionales en matière d'accès et de faible latence. Si vous en avez besoin ultérieurement, vous pouvez transformer un secret de réplique en un secret autonome, puis le configurer pour une réplication indépendante. Secrets Manager réplique les données et métadonnées chiffrées du secret, telles que les identifications et les politiques de ressources, dans les régions spécifiées.

L'ARN d'un secret répliqué est identique au secret principal à l'exception de la région, par exemple :

- Secret principal :`arn:aws:secretsmanager:Region1:123456789012:secret:MySecret-a1b2c3`
- Secret de réplica : `arn:aws:secretsmanager:Region2:123456789012:secret:MySecret-a1b2c3`

Pour obtenir des informations sur la tarification des secrets de réplica, veuillez consulter la Tarification d'AWS Secrets Manager.

Lorsque vous stockez les informations d'identification de base de données pour une base de données source répliquée vers d'autres régions, le secret contient des informations de connexion pour la base de données source. Si vous répliquez ensuite le secret, les réplicas sont des copies du secret source et contiennent les mêmes informations de connexion. Vous pouvez ajouter des key/value paires supplémentaires au secret pour obtenir des informations de connexion régionales.

Si vous activez la rotation pour votre secret principal, Secrets Manager effectue une rotation du secret dans la région principale et la nouvelle valeur du secret se propage à tous les secrets répliqués associés. Vous n'avez pas à gérer la rotation individuellement pour tous les secrets répliqués.

Vous pouvez reproduire les secrets dans toutes vos AWS régions activées. Toutefois, si vous utilisez Secrets Manager dans des AWS régions spéciales telles que AWS GovCloud (US) la Chine, vous ne pouvez configurer les secrets et les répliques que dans ces AWS régions spécialisées. Vous ne pouvez pas répliquer un secret de vos AWS régions activées vers une région spécialisée, ni reproduire un secret d'une région spécialisée vers une région commerciale.

Avant de pouvoir répliquer un secret vers une autre région, vous devez activer cette région. Pour plus d'informations, consultez Gestion des régions AWS.

Il est possible d'utiliser un secret dans plusieurs régions sans le répliquer en appelant le point de terminaison Secrets Manager dans la région où le secret est stocké. Pour obtenir la liste des points de terminaison , consultez the section called "Points de terminaison Secrets Manager". Pour utiliser la réplication afin d'améliorer la résilience de votre charge de travail, consultez Disaster Recovery Architecture on AWS, Part I : Strategies for Recovery in the Cloud.

Secrets Manager génère une entrée de CloudTrail journal lorsque vous répliquez un secret. Pour de plus amples informations, veuillez consulter the section called "Connectez-vous avec AWS CloudTrail ".

Pour répliquer un secret vers d'autres régions (console)

1. Ouvrez la console Secrets Manager à l'adresse https://console.aws.amazon.com/secretsmanager/.
2. Dans la liste des secrets, choisissez le secret.
3. Sur la page Détails du secret, sous l'onglet Réplication procédez de l'une des manières suivantes :
   - Si votre secret n'est pas répliqué, choisissez Replicate secret (Répliquer le secret).
   - Si votre secret est répliqué, dans la section Replicate secret (Répliquer le secret), choisissez Add Region (Ajouter une région).
4. Dans la boîte de dialogue Add replica regions, procédez comme suit :
   a. Pour AWS Région, choisissez la région dans laquelle vous souhaitez répliquer le secret.
   b. (Facultatif) Pour Clé de chiffrement, choisissez une clé KMS avec laquelle chiffrer le secret. La clé doit se trouver dans la région de la réplique.
   c. (Facultatif) Pour ajouter une autre région, choisissez Add more regions (Ajouter des régions supplémentaires).
   d. Choisissez Replicate (Répliquer).

   Vous revenez à la page des détails du secret. Dans la section Replicate secret (Répliquer le secret), l'état de réplication s'affiche pour chaque région.

AWS CLI

Example Réplication d'un secret vers une autre région

L'exemple suivant `replicate-secret-to-regions` réplique un secret vers la zone eu-west-3. La réplique est chiffrée à l'aide de la clé AWS géréeaws/secretsmanager.

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

Example Créez un secret et dupliquez-le

L'exemple suivant crée un secret et le réplique dans eu-west-3. La réplique est cryptée avec le Clé gérée par AWS aws/secretsmanager.

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

AWS SDK

Pour répliquer un secret, utilisez la commande `ReplicateSecretToRegions`. Pour de plus amples informations, veuillez consulter the section called "AWS SDKs".

Transformez une réplique de secret en secret autonome dans AWS Secrets Manager

Un secret de réplique est un secret qui est répliqué d'un secret principal vers un autre Région AWS. Il possède la même valeur et les mêmes métadonnées de secret que le principal, mais il peut être chiffré avec une clé KMS différente. Un secret répliqué ne peut pas être mis à jour indépendamment de son secret principal, sauf pour sa clé de chiffrement. Promouvoir un secret répliqué déconnecte le secret répliqué du secret principal et fait du secret répliqué un secret autonome. Les modifications apportées au secret principal ne seront pas répliquées au secret autonome.

Vous pouvez promouvoir un secret répliqué à un secret autonome comme une solution de reprise après sinistre si le secret principal devient indisponible. Vous pouvez également promouvoir un réplica vers un secret autonome si vous souhaitez activer la rotation pour le réplica.

Si vous promouvez un réplica, veillez à mettre à jour les applications correspondantes pour utiliser le secret autonome.

Secrets Manager génère une entrée de CloudTrail journal lorsque vous promouvez un secret. Pour de plus amples informations, veuillez consulter the section called "Connectez-vous avec AWS CloudTrail ".

Pour promouvoir un secret répliqué (console)

1. Connectez-vous au Secrets Manager à l'adresse https://console.aws.amazon.com/secretsmanager/.
2. Accédez à la région de réplique.
3. Dans la page Secrets, choisissez le secret répliqué.
4. Dans la page sur les détails du secret répliqué, choisissez Promote to standalone secret (Promouvoir en un secret autonome).
5. Dans la boîte de dialogue Promote replica to standalone secret (Promouvoir un secret répliqué en un secret autonome), saisissez la région, puis choisissez Promote replica (Promouvoir la réplique).

AWS CLI

Example Promouvoir un secret répliqué en secret principal

L'exemple suivant `stop-replication-to-replica` supprime le lien entre un secret de réplique et le secret principal. Le secret répliqué est promu en secret principal dans la région de la réplique. Vous devez appeler `stop-replication-to-replica` depuis la région où se trouve la réplique.

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

AWS SDK

Pour promouvoir un réplica vers un secret autonome, utilisez la commande `StopReplicationToReplica`. Vous devez appeler cette commande depuis la région du secret répliqué. Pour de plus amples informations, veuillez consulter the section called "AWS SDKs".

Empêcher AWS Secrets Manager la réplication

Étant donné que les secrets peuvent être répliqués à l'aide de `ReplicateSecretToRegions`ou lorsqu'ils sont créés à l'aide de ceux-ci `CreateSecret`, si vous souhaitez empêcher les utilisateurs de répliquer des secrets, nous vous recommandons d'empêcher les actions contenant le `AddReplicaRegions` paramètre. Vous pouvez utiliser une `Condition` déclaration dans vos politiques d'autorisation pour autoriser uniquement les actions qui n'ajoutent pas de zones de réplication. Consultez les exemples de politique suivants pour les déclarations de condition que vous pouvez utiliser.

Example Empêcher l'autorisation de réplication

L'exemple de politique suivant montre comment autoriser toutes les actions qui n'ajoutent pas de régions de réplication. Cela empêche les utilisateurs de répliquer des secrets à la fois via `ReplicateSecretToRegions` et`CreateSecret`.

JSON

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

Example Autoriser la réplication uniquement pour des régions spécifiques

La politique suivante indique comment autoriser tous les éléments suivants :

- Créez des secrets sans réplication
- Créez des secrets en les répliquant uniquement dans les régions des États-Unis d'Amérique et du Canada
- Répliquez les secrets dans les régions des États-Unis d'Amérique et du Canada uniquement

JSON

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

Résoudre les problèmes de réplication AWS Secrets Manager

AWS Secrets Manager l'échec de la réplication peut se produire pour diverses raisons. Pour vérifier les raisons de l'échec de la réplication d'un secret, vous pouvez effectuer l'une des opérations suivantes :

- Appelez l'opération de `DescribeSecret` l'API
- Passez en revue AWS CloudTrail les événements

En cas d'échec de la réplication :

- S'il n'existe aucune version secrète utilisable, Secrets Manager supprime le secret de la région de réplication.
- Si des versions secrètes ont été répliquées avec succès, elles restent dans la région de réplication jusqu'à ce que vous les supprimiez explicitement à l'aide de l'opération `RemoveRegionsFromReplication` d'API.

Les sections suivantes décrivent certaines des raisons les plus courantes des erreurs de réplication.

Un secret avec le même nom existe dans la région sélectionnée

Pour résoudre ce problème, vous pouvez écraser le secret du nom dupliqué dans la région de réplica. Réessayer la réplication, puis dans la boîte de dialogue Retry replication (Réessayer la réplication), choisissez Overwrite (Écraser).

Aucune autorisation disponible sur la clé KMS pour terminer la réplication

Secrets Manager décrypte d'abord le secret avant de le crypter de nouveau avec la nouvelle clé KMS de la région de réplication. Si `kms:Decrypt` ne vous autorise pas à accéder à la clé de chiffrement dans la région principale, vous rencontrerez cette erreur. Pour chiffrer le secret répliqué avec une clé KMS autre que `aws/secretsmanager`, vous avez besoin de `kms:GenerateDataKey` et de `kms:Encrypt` pour la clé. Consultez the section called “Autorisations pour la clé KMS”.

La clé KMS est désactivée ou introuvable

Secrets Manager ne peut pas répliquer le secret si la clé de chiffrement de la région principale est désactivée ou supprimée. Si le secret contient des versions étiquetées personnalisées chiffrées avec la clé de chiffrement désactivée ou supprimée, cette erreur peut se produire même si vous avez modifié la clé de chiffrement. Pour plus d'informations sur le mode de chiffrement de Secrets Manager, consultez la section the section called “Chiffrement et déchiffrement de secret”. Pour contourner ce problème, vous pouvez recréer les versions de secret afin que Secrets Manager les chiffre avec la clé de chiffrement actuelle. Pour de plus amples informations, consultez la section

Modification de la clé de chiffrement d'un secret (français non garanti). Réessayez ensuite la réplication.

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

Vous n'avez pas activé la région dans laquelle la réplication se produit

Pour des informations sur comment activer une région, consultez Gestion des régions AWS . dans le Guide de référence de la gestion des comptes AWS .

Obtenez des secrets auprès de AWS Secrets Manager

Secrets Manager génère une entrée de CloudTrail journal lorsque vous récupérez un secret. Pour de plus amples informations, veuillez consulter the section called "Connectez-vous avec AWS CloudTrail ".

Vous pouvez récupérer des valeurs secrètes en utilisant :

- Obtenir une valeur secrète de Secrets Manager à l'aide de Java
- Obtenir une valeur secrète de Secrets Manager en utilisant Python
- Obtenez une valeur secrète de Secrets Manager en utilisant .NET
- Obtenez une valeur secrète de Secrets Manager à l'aide de Go
- Obtenez une valeur secrète de Secrets Manager en utilisant Rust
- Utiliser des AWS Secrets Manager secrets dans Amazon Elastic Kubernetes Service
- Utiliser des AWS Secrets Manager secrets dans les AWS Lambda fonctions
- Utilisation de l' AWS Secrets Manager agent
- Obtenez une valeur secrète de Secrets Manager à l'aide du AWS SDK C++
- Obtenez une valeur secrète de Secrets Manager à l'aide du JavaScript AWS SDK
- Obtenez une valeur secrète de Secrets Manager à l'aide du SDK Kotlin AWS
- Obtenez une valeur secrète de Secrets Manager à l'aide du AWS SDK PHP
- Obtenez une valeur secrète de Secrets Manager à l'aide du AWS SDK Ruby
- Obtenez une valeur secrète à l'aide du AWS CLI
- Obtenir une valeur secrète à l'aide de la AWS console
- Utilisez AWS Secrets Manager des secrets dans AWS Batch
- Trouver un AWS Secrets Manager secret dans une AWS CloudFormation ressource
- Utilisez AWS Secrets Manager les secrets dans les GitHub emplois
- Utilisez AWS Secrets Manager des secrets dans AWS IoT Greengrass
- Utiliser des AWS Secrets Manager secrets dans Parameter Store

Obtenir une valeur secrète de Secrets Manager à l'aide de Java

Dans les applications, vous pouvez récupérer vos secrets en appelant `GetSecretValue` ou `BatchGetSecretValue` dans l'un des AWS SDKs. Cependant, nous vous recommandons de

mettre en cache vos valeurs de secret à l'aide de la mise en cache côté client. La mise en cache des secrets améliore la vitesse et réduit vos coûts.

Pour vous connecter à une base de données à l'aide des informations d'identification contenues dans un code secret, vous pouvez utiliser les pilotes de connexion SQL Secrets Manager, qui encapsulent le pilote JDBC de base. Cela utilise également la mise en cache côté client, ce qui permet de réduire le coût d'appel de Secrets Manager. APIs

Rubriques

- Obtenez une valeur secrète de Secrets Manager à l'aide de Java avec mise en cache côté client
- Connectez-vous à une base de données SQL à l'aide de JDBC avec des informations d'identification enregistrées dans un secret AWS Secrets Manager
- Obtenez une valeur secrète de Secrets Manager à l'aide du AWS SDK Java

Obtenez une valeur secrète de Secrets Manager à l'aide de Java avec mise en cache côté client

Lorsque vous récupérez un secret, vous pouvez utiliser le composant de mise en cache basé sur Java de Secrets Manager pour le mettre en cache en vue d'une utilisation future. Il est plus rapide de récupérer un secret mis en cache que de le récupérer à partir de Secrets Manager. L'appel de Secrets Manager étant payant APIs, l'utilisation d'un cache peut vous permettre de réduire vos coûts. Pour connaître toutes les manières dont vous pouvez récupérer des secrets, consultezObtenez des secrets.

La politique de cache est la moins récemment utilisée (LRU). Ainsi, lorsque le cache doit supprimer un secret, il supprime le secret le moins récemment utilisé. Par défaut, le cache actualise les secrets toutes les heures. Vous pouvez configurer la fréquence d'actualisation du secret dans le cache et utiliser un hook pour la récupération du secret afin d'ajouter plus de fonctionnalités.

Le cache ne force pas le récupérateur de mémoire une fois que les références du cache sont libérées. L'implémentation du cache n'inclut pas l'invalidation du cache. L'implémentation du cache se concentre sur le cache lui-même et n'est pas renforcée ou ciblée sur le plan de la sécurité. Si vous avez besoin d'une sécurité supplémentaire, telle que le chiffrement d'éléments dans le cache, utilisez les interfaces et les méthodes abstraites fournies.

Pour pouvoir utiliser le composant, vous devez disposer des éléments suivants :

- Environnement de développement Java 8 ou une version ultérieure. Consultez Java SE Downloads sur le site web d'Oracle.

Pour télécharger le code source, consultez le composant client de mise en cache basé sur Java de Secrets Manager sur. GitHub

Pour ajouter le composant à votre projet, dans votre fichier Maven pom.xml, intégrez la dépendance suivante. Pour plus d'informations sur Maven, consultez le Getting Started Guide sur le site web Apache Maven Project.

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

Autorisations requises :

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

Pour de plus amples informations, veuillez consulter Référence des autorisations .

Référence

- SecretCache
- SecretCacheConfiguration
- SecretCacheHook

Example Récupérer un secret

L'exemple de code suivant montre une fonction Lambda qui récupère une chaîne de secret. Il respecte les bonnes pratiques consistant à instancier le cache en dehors du gestionnaire de fonctions, de sorte qu'il ne continue pas d'appeler l'API si vous appelez à nouveau la fonction Lambda.

```
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

Un cache en mémoire pour les secrets demandés à Secrets Manager. Vous utilisez the section called "getSecretString" ou the section called "getSecretBinary" pour récupérer un secret du cache. Vous pouvez configurer les paramètres de cache en transmettant un objet the section called "SecretCacheConfiguration" dans le constructeur.

Pour plus d'informations, ainsi que pour voir des exemples, consultez the section called "Java avec mise en cache côté client".

Constructeurs

`public SecretCache()`

Constructeur par défaut d'un objet `SecretCache`.

`public SecretCache(AWSSecretsManagerClientBuilder builder)`

Crée un cache à l'aide d'un client Secrets Manager créé à l'aide du `AWSSecretsManagerClientBuilder` fourni. Utilisez ce constructeur pour personnaliser le client Secrets Manager, par exemple pour utiliser une région ou un point de terminaison spécifique.

`public SecretCache(AWSSecretsManager client)`

Crée un cache de secret à l'aide du `AWSSecretsManagerClient` fourni. Utilisez ce constructeur pour personnaliser le client Secrets Manager, par exemple pour utiliser une région ou un point de terminaison spécifique.

`public SecretCache(SecretCacheConfiguration config)`

Crée un cache de secret à l'aide du the section called "SecretCacheConfiguration" fourni.

Méthodes

getSecretString

`public String getSecretString(final String secretId)`

Récupère un secret de chaîne dans Secrets Manager. Retourne un `String`.

getSecretBinary

`public ByteBuffer getSecretBinary(final String secretId)`

Récupère un secret binaire dans Secrets Manager. Retourne un `ByteBuffer`.

refreshNow

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

Force l'actualisation du cache. Renvoie `true` si l'actualisation s'est terminée sans erreur, sinon `false`.

close

`public void close()`

Ferme le cache.

SecretCacheConfiguration

Options de configuration du cache pour un the section called "SecretCache", telles que la taille maximale du cache et la durée de vie (TTL) pour les secrets mis en cache.

Constructeur

`public SecretCacheConfiguration`

Constructeur par défaut d'un objet `SecretCacheConfiguration`.

Méthodes

getClient

`public AWSSecretsManager getClient()`

Renvoie le `AWSSecretsManagerClient` dont le cache récupère les secrets.

setClient

`public void setClient(AWSSecretsManager client)`

Définit le client `AWSSecretsManagerClient` à partir desquels le cache récupère des secrets.

getCacheHook

`public SecretCacheHook getCacheHook()`

Renvoie l'interface the section called "SecretCacheHook" utilisée pour raccorder les mises à jour du cache.

setCacheHook

`public void setCacheHook(SecretCacheHook cacheHook)`

Définit l'interface the section called "SecretCacheHook" utilisée pour raccorder les mises à jour du cache.

getMaxCacheTaille

`public int getMaxCacheSize()`

Renvoie la taille maximale du cache. La valeur par défaut est de 1 024 secrets.

setMaxCacheTaille

`public void setMaxCacheSize(int maxCacheSize)`

Définit la taille maximale du cache. La valeur par défaut est de 1 024 secrets.

getCacheItemTTL

```
public long getCacheItemTTL()
```

Renvoie la durée de vie (TTL) en millisecondes pour les éléments mis en cache. Lorsqu'un secret mis en cache dépasse cette durée de vie, le cache récupère une nouvelle copie du secret à partir du `AWSSecretsManagerClient`. La valeur par défaut est de 1 heure en millisecondes.

Le cache actualise le secret de manière synchrone lorsque le secret est demandé après la durée de vie. Si l'actualisation synchrone échoue, le cache renvoie le secret obsolète.

setCacheItemTTL

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

Définit la durée de vie (TTL) en millisecondes pour les éléments mis en cache. Lorsqu'un secret mis en cache dépasse cette durée de vie, le cache récupère une nouvelle copie du secret à partir du `AWSSecretsManagerClient`. La valeur par défaut est de 1 heure en millisecondes.

getVersionStage

```
public String getVersionStage()
```

Renvoie la version des secrets que vous souhaitez mettre en cache. Pour plus d'informations, consultez Versions de secret. La valeur par défaut est `"AWSCURRENT"`.

setVersionStage

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

Définit la version des secrets que vous souhaitez mettre en cache. Pour plus d'informations, consultez Versions de secret. L'argument par défaut est `"AWSCURRENT"`.

SecretCacheConfiguration Avec le client

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

Définit le `AWSSecretsManagerClient` pour en récupérer des secrets. Renvoie l'objet `SecretCacheConfiguration` mis à jour avec le nouveau paramètre.

SecretCacheConfiguration withCacheHook

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

Définit l'interface utilisée pour raccorder le cache en mémoire. Renvoie l'objet `SecretCacheConfiguration` mis à jour avec le nouveau paramètre.

SecretCacheConfiguration withMaxCacheTaille

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

Définit la taille maximale du cache. Renvoie l'objet `SecretCacheConfiguration` mis à jour avec le nouveau paramètre.

SecretCacheConfiguration withCacheItemTTL

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

Définit la durée de vie (TTL) en millisecondes pour les éléments mis en cache. Lorsqu'un secret mis en cache dépasse cette durée de vie, le cache récupère une nouvelle copie du secret à partir du `AWSSecretsManagerClient`. La valeur par défaut est de 1 heure en millisecondes. Renvoie l'objet `SecretCacheConfiguration` mis à jour avec le nouveau paramètre.

SecretCacheConfiguration withVersionStage

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

Définit la version des secrets que vous souhaitez mettre en cache. Pour plus d'informations, consultez Versions de secret. Renvoie l'objet `SecretCacheConfiguration` mis à jour avec le nouveau paramètre.

SecretCacheHook

Une interface pour se connecter à un the section called "SecretCache" pour effectuer des actions sur les secrets stockés dans le cache.

put

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

Préparez l'objet en vue de son stockage dans le cache.

Renvoie l'objet à stocker dans le cache.

get

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

Dérivez l'objet de l'objet mis en cache.

Renvoie l'objet à renvoyer à partir du cache

Connectez-vous à une base de données SQL à l'aide de JDBC avec des informations d'identification enregistrées dans un secret AWS Secrets Manager

Dans les applications Java, vous pouvez utiliser les pilotes de connexion SQL Secrets Manager pour vous connecter aux bases de données MySQL, PostgreSQL, Oracle MSSQLServer, Db2 et Redshift à l'aide des informations d'identification stockées dans Secrets Manager. Chaque pilote encapsule le pilote JDBC de base, vous pouvez donc utiliser des appels JDBC pour accéder à votre base de données. Cependant, au lieu de transmettre un nom d'utilisateur et un mot de passe pour la connexion, vous fournissez l'ID d'un secret. Le pilote appelle Secrets Manager pour récupérer la valeur du secret, puis utilise les informations d'identification dans le secret pour se connecter à la base de données. Le pilote met également en cache les informations d'identification à l'aide de la bibliothèque de mise en cache côté client Java, de sorte que les futures connexions ne nécessitent pas d'effectuer d'appel à Secrets Manager. Par défaut, le cache est actualisé toutes les heures et également lors de la rotation du secret. Pour configurer le cache, consultez the section called "SecretCacheConfiguration".

Vous pouvez télécharger le code source à partir de GitHub.

Pour utiliser les pilotes de connexion SQL Secrets Manager :

- Votre application doit être en Java 8 ou une version ultérieure.
- Votre secret doit être parmi les suivants :
  - Un secret de base de données dans la structure JSON attendue. Pour vérifier le format, dans la console Secrets Manager, affichez votre secret et sélectionnez Retrieve secret value (Récupérer la valeur du secret). Sinon, dans le AWS CLI, appelez get-secret-value.
  - Un secret géré par Amazon RDS. Pour ce type de secret, vous devez spécifier un point de terminaison et un port lorsque vous établissez la connexion.
  - Un secret géré par Amazon Redshift. Pour ce type de secret, vous devez spécifier un point de terminaison et un port lorsque vous établissez la connexion.

Si votre base de données est répliquée vers d'autres régions, pour vous connecter à une base de données de réplica dans une autre région, vous spécifiez le point de terminaison et le port régionaux lorsque vous créez la connexion. Vous pouvez stocker des informations de connexion régionale dans

le secret sous forme de paires clé/valeur supplémentaires, dans les paramètres SSM Parameter Store ou dans la configuration de votre code.

Pour ajouter le pilote à votre projet, dans votre fichier de création Maven `pom.xml`, ajoutez la dépendance suivante pour le pilote. Pour plus d'informations, consultez Secrets Manager SQL Connection Library (français non garanti) sur le site web de Maven Central Repository.

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

Le pilote utilise la chaîne de fournisseurs d'informations d'identification par défaut. Si vous exécutez le pilote sur Amazon EKS, il est possible qu'il récupère les informations d'identification du nœud sur lequel il s'exécute au lieu du rôle de compte de service. Pour résoudre ce problème, ajoutez la version 1 de `com.amazonaws:aws-java-sdk-sts` à votre fichier de projet Gradle ou Maven en tant que dépendance.

Pour définir une URL de point de terminaison AWS PrivateLink DNS et une région dans le `secretsmanager.properties` fichier :

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

Pour remplacer la région principale, définissez la variable d'environnement `AWS_SECRET_JDBC_REGION` ou apportez les modifications suivantes au fichier `secretsmanager.properties` :

```
drivers.region = region
```

Autorisations requises :

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

Pour de plus amples informations, veuillez consulter Référence des autorisations .

Exemples :

- Établir une connexion à une base de données
- Établir une connexion en spécifiant le point de terminaison et le port
- Utiliser le regroupement de connexions c3p0 pour établir une connexion
- Utiliser le regroupement de connexions c3p0 pour établir une connexion en spécifiant le point de terminaison et le port

Établir une connexion à une base de données

L'exemple suivant montre comment établir une connexion à une base de données à l'aide des informations d'identification et de connexion dans un secret. Une fois la connexion établie, vous pouvez utiliser les appels JDBC pour accéder à la base de données. Pour plus d'informations, consultez JDBC Basics sur le site web dédié à la documentation Java.

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);
```

Établir une connexion en spécifiant le point de terminaison et le port

L'exemple suivant montre comment établir une connexion avec une base de données en utilisant les informations d'identification dans un secret avec un point de terminaison et un port que vous devez spécifier.

Les secrets gérés par Amazon RDS n'incluent pas le point de terminaison et le port de la base de données. Pour vous connecter à une base de données à l'aide d'informations d'identification principales dans un secret géré par Amazon RDS, vous devez les spécifier dans votre code.

Les secrets répliqués vers d'autres régions peuvent améliorer la latence de la connexion à la base de données régionale, mais ils ne contiennent pas d'informations de connexion différentes de celles du secret source. Chaque réplica est une copie du secret source. Pour stocker les informations de connexion régionale dans le secret, ajoutez d'autres paires clé-valeur pour le point de terminaison et les informations de port pour les régions.

Une fois la connexion établie, vous pouvez utiliser les appels JDBC pour accéder à la base de données. Pour plus d'informations, consultez JDBC Basics sur le site web dédié à la documentation Java.

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);
```

Utiliser le regroupement de connexions c3p0 pour établir une connexion

L'exemple suivant montre comment établir un groupement de connexions à l'aide d'un `c3p0.properties` fichier qui se sert du pilote pour récupérer les informations d'identification et de connexion à partir du secret. Pour `user` et `jdbcUrl`, saisissez l'ID secret pour configurer le groupe de connexions. Vous pouvez ensuite récupérer les connexions du groupe et les utiliser comme n'importe quelle autre connexion de base de données. Pour plus d'informations, consultez JDBC Basics sur le site web dédié à la documentation Java.

Pour plus d'informations sur c3p0, consultez c3p0 sur le site web Machinery For Change.

MySQL

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

PostgreSQL

```
c3p0.user=secretId
```

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

Oracle

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

MSSQLServer

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

Db2

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

Redshift

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

Utiliser le regroupement de connexions c3p0 pour établir une connexion en spécifiant le point de terminaison et le port

L'exemple suivant montre comment établir un pool de connexions avec un `c3p0.properties` fichier qui utilise le pilote pour récupérer des informations d'identification dans un secret avec un point de terminaison et un port que vous spécifiez. Vous pouvez ensuite récupérer les connexions du groupe et les utiliser comme n'importe quelle autre connexion de base de données. Pour plus d'informations, consultez JDBC Basics sur le site web dédié à la documentation Java.

Les secrets gérés par Amazon RDS n'incluent pas le point de terminaison et le port de la base de données. Pour vous connecter à une base de données à l'aide d'informations d'identification principales dans un secret géré par Amazon RDS, vous devez les spécifier dans votre code.

Les secrets répliqués vers d'autres régions peuvent améliorer la latence de la connexion à la base de données régionale, mais ils ne contiennent pas d'informations de connexion différentes de celles du secret source. Chaque réplica est une copie du secret source. Pour stocker les informations de connexion régionale dans le secret, ajoutez d'autres paires clé-valeur pour le point de terminaison et les informations de port pour les régions.

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
```

Obtenez une valeur secrète de Secrets Manager à l'aide du AWS SDK Java

Dans les applications, vous pouvez récupérer vos secrets en appelant `GetSecretValue` ou `BatchGetSecretValue` dans l'un des AWS SDKs. Cependant, nous vous recommandons de mettre en cache vos valeurs de secret à l'aide de la mise en cache côté client. La mise en cache des secrets améliore la vitesse et réduit vos coûts.

- Si vous stockez les informations d'identification de la base de données dans le secret, utilisez les pilotes de connexion Secrets Manager SQL pour vous connecter à une base de données à l'aide des informations d'identification que contient le secret.
- Pour les autres types de secrets, utilisez le composant de mise en cache Java de Secrets Manager ou appelez le SDK directement avec ou. `GetSecretValue` `BatchGetSecretValue`

Les exemples de code suivants illustrent comment utiliser `GetSecretValue`.

Autorisations requises : `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);
        }
    }
```

```
}
```

Obtenir une valeur secrète de Secrets Manager en utilisant Python

Dans les applications, vous pouvez récupérer vos secrets en appelant `GetSecretValue` ou `BatchGetSecretValue` dans l'un des AWS SDKs. Cependant, nous vous recommandons de mettre en cache vos valeurs de secret à l'aide de la mise en cache côté client. La mise en cache des secrets améliore la vitesse et réduit vos coûts.

Rubriques

- Obtenez une valeur secrète de Secrets Manager en utilisant Python avec mise en cache côté client
- Obtenez une valeur secrète de Secrets Manager à l'aide du AWS SDK Python
- Obtenez un lot de valeurs secrètes de Secrets Manager à l'aide du AWS SDK Python

Obtenez une valeur secrète de Secrets Manager en utilisant Python avec mise en cache côté client

Lorsque vous récupérez un secret, vous pouvez utiliser le composant de mise en cache basé sur Python de Secrets Manager pour le mettre en cache en vue d'une utilisation future. Il est plus rapide de récupérer un secret mis en cache que de le récupérer à partir de Secrets Manager. L'appel de Secrets Manager étant payant APIs, l'utilisation d'un cache peut réduire vos coûts. Pour connaître toutes les manières dont vous pouvez récupérer des secrets, consultezObtenez des secrets.

La politique de cache est la moins récemment utilisée (LRU). Ainsi, lorsque le cache doit supprimer un secret, il supprime le secret le moins récemment utilisé. Par défaut, le cache actualise les secrets toutes les heures. Vous pouvez configurer la fréquence d'actualisation du secret dans le cache et utiliser un hook pour la récupération du secret afin d'ajouter plus de fonctionnalités.

Le cache ne force pas le récupérateur de mémoire une fois que les références du cache sont libérées. L'implémentation du cache n'inclut pas l'invalidation du cache. L'implémentation du cache se concentre sur le cache lui-même et n'est pas renforcée ou ciblée sur le plan de la sécurité. Si vous avez besoin d'une sécurité supplémentaire, telle que le chiffrement d'éléments dans le cache, utilisez les interfaces et les méthodes abstraites fournies.

Pour pouvoir utiliser le composant, vous devez disposer des éléments suivants :

- Python 3.6 ou version ultérieure.

- botocore 1.12 ou version ultérieure. Consultez les sections Kit SDK AWS pour Python et Botocore.
- setuptools_scm 3.2 ou version ultérieure. Voir https://pypi. org/project/setuptools-scm/.

Pour télécharger le code source, voir le composant client de mise en cache basé sur Python de Secrets Manager sur. GitHub

Pour installer le composant, utilisez la commande suivante.

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

Autorisations requises :

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

Pour de plus amples informations, veuillez consulter Référence des autorisations .

Référence

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

Example Récupérer un secret

L'exemple suivant montre comment obtenir la valeur secrète d'un secret nommé*mysecret*.

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

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

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

SecretCache

Un cache en mémoire pour les secrets récupérés dans Secrets Manager. Vous utilisez the section called "get_secret_string" ou the section called "get_secret_binary" pour récupérer un secret du cache. Vous pouvez configurer les paramètres de cache en transmettant un objet the section called "SecretCacheConfig" dans le constructeur.

Pour plus d'informations, ainsi que pour voir des exemples, consultez the section called "Python avec mise en cache côté client".

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

Les méthodes disponibles sont les suivantes :

- get_secret_string
- get_secret_binary

get_secret_string

Récupère la valeur de chaîne secrète.

Syntaxe de demande

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

Paramètres

- `secret_id`(chaîne) : [Obligatoire] Le nom ou l'ARN du secret.
- `version_stage`(chaîne) : version des secrets que vous souhaitez récupérer. Pour plus d'informations, consultez la section Versions secrètes. La valeur par défaut est « AWSCURRENT ».

Type de retour

chaîne

get_secret_binary

Récupère la valeur binaire secrète.

Syntaxe de demande

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

Paramètres

- `secret_id`(chaîne) : [Obligatoire] Le nom ou l'ARN du secret.
- `version_stage`(chaîne) : version des secrets que vous souhaitez récupérer. Pour plus d'informations, consultez la section Versions secrètes. La valeur par défaut est « AWSCURRENT ».

Type de retour

Chaîne base64-encoded

SecretCacheConfig

Options de configuration du cache pour un the section called "SecretCache", telles que la taille maximale du cache et la durée de vie (TTL) pour les secrets mis en cache.

Paramètres

`max_cache_size` (int)

Taille de cache maximale. La valeur par défaut est `1024` secrets.

`exception_retry_delay_base` (int)

Le nombre de secondes à attendre après la rencontre d'une exception avant d'effectuer une nouvelle demande. L'argument par défaut est 1.

`exception_retry_growth_factor` (int)pur

Facteur de croissance à utiliser pour calculer le temps d'attente entre les tentatives de demandes qui ont échoué. L'argument par défaut est 2.

`exception_retry_delay_max` (int)

Durée maximale d'attente en secondes entre les demandes qui ont échoué. L'argument par défaut est `3600`.

`default_version_stage` (str)

La version des secrets que vous souhaitez mettre en cache. Pour plus d'informations, consultez Versions de secret. L'argument par défaut est `'AWSCURRENT'`.

`secret_refresh_interval` (int)

Nombre de secondes à attendre entre l'actualisation des informations secrètes mises en cache. L'argument par défaut est `3600`.

`secret_cache_hook` (SecretCacheHook)

Implémentation de la classe abstraite `SecretCacheHook`. La valeur par défaut est `None`.

SecretCacheHook

Une interface permettant d'utiliser un hook pour un the section called "SecretCache" et effectuer des actions sur les secrets stockés dans le cache.

Les méthodes disponibles sont les suivantes :

- put
- get

put

Prépare l'objet en vue de son stockage dans le cache.

Syntaxe de demande

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

Paramètres

- `obj` (objet) -- [Obligatoire] Secret ou objet qui contient le secret.

Type de retour

objet

get

Dérive l'objet de l'objet mis en cache.

Syntaxe de demande

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

Paramètres

- `obj`(objet) : [Obligatoire] Le secret ou l'objet contenant le secret.

Type de retour

objet

@InjectSecretString

Ce décorateur attend une chaîne d'identité secrète et the section called "SecretCache" comme premier et deuxième arguments. Le décorateur renvoie la valeur de chaîne secrète. Le secret doit contenir une chaîne.

```
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

Ce décorateur attend une chaîne d'identité secrète et the section called "SecretCache" comme premier et deuxième arguments. Les arguments restants mappent les paramètres de la fonction encapsulée aux clés JSON du secret. Le secret doit contenir une chaîne dans la structure JSON.

Pour un secret contenant ce JSON :

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

L'exemple suivant montre comment extraire du secret les valeurs JSON pour `username` et `password`.

```
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')
```

Obtenez une valeur secrète de Secrets Manager à l'aide du AWS SDK Python

Dans les applications, vous pouvez récupérer vos secrets en appelant `GetSecretValue` ou `BatchGetSecretValue` dans l'un des AWS SDKs. Cependant, nous vous recommandons de mettre en cache vos valeurs de secret à l'aide de la mise en cache côté client. La mise en cache des secrets améliore la vitesse et réduit vos coûts.

Pour les applications Python, utilisez le composant de mise en cache basé sur Python de Secrets Manager ou appelez le SDK directement avec `get_secret_value` ou `batch_get_secret_value`.

Les exemples de code suivants illustrent comment utiliser `GetSecretValue`.

Autorisations requises : `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
```

Obtenez un lot de valeurs secrètes de Secrets Manager à l'aide du AWS SDK Python

L'exemple de code suivant montre comment obtenir un lot de valeurs secrètes de Secrets Manager.

Autorisations requises :

- `secretsmanager:BatchGetSecretValue`
- `secretsmanager:GetSecretValue`autorisation pour chaque secret que vous souhaitez récupérer.
- Si vous utilisez des filtres, vous devez également avoir `secretsmanager:ListSecrets`.

Pour obtenir un exemple de politique d'autorisations, consultez the section called “Exemple : autorisation de récupérer un groupe de valeurs secrètes dans un lot”.

> **Important**
>
> Si vous avez une politique VPCE qui refuse l'autorisation de récupérer un secret individuel dans le groupe que vous recherchez, `BatchGetSecretValue` ne renverra aucune valeur de secret et renverra une erreur.

```
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
```

Obtenez une valeur secrète de Secrets Manager en utilisant .NET

Dans les applications, vous pouvez récupérer vos secrets en appelant `GetSecretValue` ou `BatchGetSecretValue` dans l'un des AWS SDKs. Cependant, nous vous recommandons de mettre en cache vos valeurs de secret à l'aide de la mise en cache côté client. La mise en cache des secrets améliore la vitesse et réduit vos coûts.

Rubriques

- Obtenez une valeur secrète de Secrets Manager en utilisant .NET avec mise en cache côté client
- Obtenez une valeur secrète de Secrets Manager à l'aide du SDK pour .NET

Obtenez une valeur secrète de Secrets Manager en utilisant .NET avec mise en cache côté client

Lorsque vous récupérez un secret, vous pouvez utiliser le composant de mise en cache basé sur .NET de Secrets Manager pour le mettre en cache en vue d'une utilisation future. Il est plus rapide de récupérer un secret mis en cache que de le récupérer à partir de Secrets Manager. L'appel de Secrets Manager étant payant APIs, l'utilisation d'un cache peut réduire vos coûts. Pour connaître toutes les manières dont vous pouvez récupérer des secrets, consultezObtenez des secrets.

La politique de cache est la moins récemment utilisée (LRU). Ainsi, lorsque le cache doit supprimer un secret, il supprime le secret le moins récemment utilisé. Par défaut, le cache actualise les secrets toutes les heures. Vous pouvez configurer la fréquence d'actualisation du secret dans le cache et utiliser un hook pour la récupération du secret afin d'ajouter plus de fonctionnalités.

Le cache ne force pas le récupérateur de mémoire une fois que les références du cache sont libérées. L'implémentation du cache n'inclut pas l'invalidation du cache. L'implémentation du cache se concentre sur le cache lui-même et n'est pas renforcée ou ciblée sur le plan de la sécurité. Si vous avez besoin d'une sécurité supplémentaire, telle que le chiffrement d'éléments dans le cache, utilisez les interfaces et les méthodes abstraites fournies.

Pour pouvoir utiliser le composant, vous devez disposer des éléments suivants :

- Framework .NET 4.6.2 ou version ultérieure, ou .NET Standard 2.0 ou version ultérieure. Consultez Télécharger .NET sur le site web Microsoft .NET.
- Le AWS SDK pour .NET. Consultez the section called "AWS SDKs".

Pour télécharger le code source, consultez la section Client de mise en cache pour .NET sur GitHub.

Pour utiliser le cache, instanciez-le d'abord, puis récupérez votre secret en utilisant `GetSecretString` ou `GetSecretBinary`. Lors de récupérations successives, le cache renvoie la copie en cache du secret.

Pour obtenir le package de mise en cache

- Effectuez l'une des actions suivantes :
  - Exécutez la commande .NET CLI suivante dans le répertoire de votre projet.

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

  - Ajoutez la référence de package suivante à votre fichier `.csproj`.

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

Autorisations requises :

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

Pour de plus amples informations, veuillez consulter Référence des autorisations .

Référence

- SecretsManagerCache
- SecretCacheConfiguration
- ISecretCacheHook

Example Récupérer un secret

L'exemple de code suivant montre une méthode qui récupère un secret nomméMySecret.

```
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 Configuration de la durée d'actualisation du cache de durée de vie (TTL)

L'exemple de code suivant montre une méthode qui récupère un secret nommé *MySecret* et définit la durée d'actualisation du cache TTL à 24 heures.

```
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

Un cache en mémoire pour les secrets demandés à Secrets Manager. Vous utilisez the section called "GetSecretString" ou the section called "GetSecretBinary" pour récupérer un secret du cache. Vous pouvez configurer les paramètres de cache en transmettant un objet the section called "SecretCacheConfiguration" dans le constructeur.

Pour plus d'informations, ainsi que pour voir des exemples, consultez the section called ".NET avec mise en cache côté client".

Constructeurs

`public SecretsManagerCache()`

Constructeur par défaut d'un objet `SecretsManagerCache`.

`public SecretsManagerCache(IAmazonSecretsManager secretsManager)`

Crée un cache à l'aide d'un client Secrets Manager créé à l'aide du AmazonSecretsManagerClient fourni. Utilisez ce constructeur pour personnaliser le client Secrets Manager, par exemple, pour utiliser une région ou un point de terminaison spécifique.

Paramètres

secretsManager

Le AmazonSecretsManagerClientpour récupérer des secrets.

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

Crée un cache de secret à l'aide du the section called "SecretCacheConfiguration" fourni. Utilisez ce constructeur pour configurer le cache, par exemple, le nombre de secrets à mettre en cache et la fréquence d'actualisation.

Paramètres

config

Une the section called "SecretCacheConfiguration" qui contient des informations de configuration pour le cache.

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

Construit un nouveau cache à l'aide d'un client Secrets Manager créé à l'aide des options fournies AmazonSecretsManagerClientet a. the section called "SecretCacheConfiguration" Utilisez ce constructeur pour personnaliser le client Secrets Manager, par exemple, pour utiliser une région ou un point de terminaison spécifique, ainsi que pour configurer le cache, par exemple le nombre de secrets à mettre en cache et la fréquence d'actualisation.

Paramètres

secretsManager

Le AmazonSecretsManagerClientpour récupérer des secrets.

config

Une the section called "SecretCacheConfiguration" qui contient des informations de configuration pour le cache.

Méthodes

GetSecretString

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

Récupère un secret de chaîne dans Secrets Manager.

Paramètres

secretId

ARN ou nom du secret à récupérer.

GetSecretBinary

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

Récupère un secret binaire dans Secrets Manager.

Paramètres

secretId

ARN ou nom du secret à récupérer.

RefreshNowAsync

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

Demande la valeur secrète à Secrets Manager et met à jour le cache avec toutes les modifications. Crée une entrée de cache s'il n'en existe pas déjà une. Renvoie `true` si l'actualisation est réussie.

Paramètres

secretId

ARN ou nom du secret à récupérer.

GetCachedSecret

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

Renvoie l'entrée de cache pour le secret spécifié s'il existe dans le cache. Sinon, récupère le secret à partir de Secrets Manager et crée une entrée de cache.

Paramètres

secretId

ARN ou nom du secret à récupérer.

SecretCacheConfiguration

Options de configuration du cache pour un the section called "SecretsManagerCache", telles que la taille maximale du cache et la durée de vie (TTL) pour les secrets mis en cache.

Propriétés

CacheItemTTL

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

Durée de vie (TTL) d'un élément de cache en millisecondes. La valeur par défaut est de `3600000` ms ou 1 heure. La valeur maximale est de `4294967295` ms, soit environ 49,7 jours.

MaxCacheSize

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

Taille de cache maximale. La valeur par défaut est de 1 024 secrets. Le valeur maximale est de 65 535.

VersionStage

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

La version des secrets que vous souhaitez mettre en cache. Pour plus d'informations, consultez Versions de secret. La valeur par défaut est `"AWSCURRENT"`.

Client

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

Le AmazonSecretsManagerClientpour récupérer des secrets. Si la valeur est `null`, le cache instancie un nouveau client. La valeur par défaut est `null`.

CacheHook

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

Une the section called "ISecretCacheHook".

ISecretCacheHook

Une interface permettant d'utiliser un hook pour un the section called "SecretsManagerCache" et effectuer des actions sur les secrets stockés dans le cache.

Méthodes

Put

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

Préparez l'objet en vue de son stockage dans le cache.

Renvoie l'objet à stocker dans le cache.

Get

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

Dérivez l'objet de l'objet mis en cache.

Renvoie l'objet à renvoyer à partir du cache

Obtenez une valeur secrète de Secrets Manager à l'aide du SDK pour .NET

Dans les applications, vous pouvez récupérer vos secrets en appelant `GetSecretValue` ou `BatchGetSecretValue` dans l'un des AWS SDKs. Cependant, nous vous recommandons de mettre en cache vos valeurs de secret à l'aide de la mise en cache côté client. La mise en cache des secrets améliore la vitesse et réduit vos coûts.

Pour les applications .NET, utilisez le composant de mise en cache basé sur .NET de Secrets Manager ou appelez le SDK directement avec `GetSecretValue` ou `BatchGetSecretValue`.

Les exemples de code suivants illustrent comment utiliser `GetSecretValue`.

Autorisations requises : `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;
            }
        }
    }
```

Obtenez une valeur secrète de Secrets Manager à l'aide de Go

Dans les applications, vous pouvez récupérer vos secrets en appelant `GetSecretValue` ou `BatchGetSecretValue` dans l'un des AWS SDKs. Cependant, nous vous recommandons de mettre en cache vos valeurs de secret à l'aide de la mise en cache côté client. La mise en cache des secrets améliore la vitesse et réduit vos coûts.

Rubriques

- Obtenez une valeur secrète de Secrets Manager à l'aide de Go avec la mise en cache côté client
- Obtenez une valeur secrète de Secrets Manager à l'aide du AWS SDK Go

Obtenez une valeur secrète de Secrets Manager à l'aide de Go avec la mise en cache côté client

Lorsque vous récupérez un secret, vous pouvez utiliser le composant de mise en cache basé sur Go de Secrets Manager pour le mettre en cache en vue d'une utilisation future. Il est plus rapide de récupérer un secret mis en cache que de le récupérer à partir de Secrets Manager. L'appel de Secrets Manager étant payant APIs, l'utilisation d'un cache peut réduire vos coûts. Pour connaître toutes les manières dont vous pouvez récupérer des secrets, consultezObtenez des secrets.

La politique de cache est la moins récemment utilisée (LRU). Ainsi, lorsque le cache doit supprimer un secret, il supprime le secret le moins récemment utilisé. Par défaut, le cache actualise les secrets toutes les heures. Vous pouvez configurer la fréquence d'actualisation du secret dans le cache et utiliser un hook pour la récupération du secret afin d'ajouter plus de fonctionnalités.

Le cache ne force pas le récupérateur de mémoire une fois que les références du cache sont libérées. L'implémentation du cache n'inclut pas l'invalidation du cache. L'implémentation du cache se concentre sur le cache lui-même et n'est pas renforcée ou ciblée sur le plan de la sécurité. Si vous

avez besoin d'une sécurité supplémentaire, telle que le chiffrement d'éléments dans le cache, utilisez les interfaces et les méthodes abstraites fournies.

Pour pouvoir utiliser le composant, vous devez disposer des éléments suivants :

- AWS SDK pour Go. Consultez the section called "AWS SDKs".

Pour télécharger le code source, consultez la section Client de mise en cache de Secrets Manager Go activé GitHub.

Pour configurer un environnement de développement Go, consultez Mise en route sur Golang sur le site web Go Programming Language.

Autorisations requises :

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

Pour de plus amples informations, veuillez consulter Référence des autorisations .

Référence

- type Cache
- type CacheConfig
- type CacheHook

Example Récupérer un secret

L'exemple de code suivant montre une fonction Lambda qui récupère un secret.

```
package main

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

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

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

  // Use the secret, return success
}

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

type Cache

Un cache en mémoire pour les secrets demandés à Secrets Manager. Vous utilisez the section called "GetSecretString" ou the section called "GetSecretBinary" pour récupérer un secret du cache.

L'exemple suivant montre comment configurer les paramètres du cache.

```
// 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 },
)
```

Pour plus d'informations, ainsi que pour voir des exemples, consultez the section called "Optez pour la mise en cache côté client".

Méthodes

New

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

New construit un cache secret à l'aide d'options fonctionnelles, sinon il utilise les valeurs par défaut. Initialise un SecretsManager client à partir d'une nouvelle session. S'initialise CacheConfig aux valeurs par défaut. Initialise le cache LRU avec une taille maximale par défaut.

GetSecretString

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

GetSecretString obtient la valeur de la chaîne secrète du cache pour un ID secret donné. Renvoie la chaîne secrète et une erreur en cas d'échec de l'opération.

GetSecretStringWithStage

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

GetSecretStringWithStage obtient la valeur de la chaîne secrète du cache pour un ID secret et une étape de version donnés. Renvoie la chaîne secrète et une erreur en cas d'échec de l'opération.

GetSecretBinary

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

GetSecretBinary obtient la valeur binaire secrète du cache pour un identifiant secret donné. Renvoie le binaire secret et une erreur si l'opération a échoué.

GetSecretBinaryWithStage

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

GetSecretBinaryWithStage obtient la valeur binaire secrète du cache pour un ID secret et une étape de version donnés. Renvoie le binaire secret et une erreur si l'opération a échoué.

type CacheConfig

Options de configuration du cache pour un Cache, telles que la taille maximale du cache, l'étape de version par défaut et la durée de vie (TTL) pour les secrets mis en cache.

```
type CacheConfig struct {
```

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

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

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

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

type CacheHook

Une interface pour se connecter à un Cache pour effectuer des actions sur le secret stocké dans le cache.

Méthodes

Put

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

Prépare l'objet en vue de son stockage dans le cache.

Get

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

Dérive l'objet de l'objet mis en cache.

Obtenez une valeur secrète de Secrets Manager à l'aide du AWS SDK Go

Dans les applications, vous pouvez récupérer vos secrets en appelant `GetSecretValue` ou `BatchGetSecretValue` dans l'un des AWS SDKs. Cependant, nous vous recommandons de mettre en cache vos valeurs de secret à l'aide de la mise en cache côté client. La mise en cache des secrets améliore la vitesse et réduit vos coûts.

Pour les applications Go, utilisez le composant de mise en cache basé sur Go de Secrets Manager appelez le SDK directement avec `GetSecretValue` ou `BatchGetSecretValue`.

L'exemple de code suivant montre comment obtenir une valeur de Secrets Secrets Manager.

Autorisations requises : `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.
  }
```

Obtenez une valeur secrète de Secrets Manager en utilisant Rust

Dans les applications, vous pouvez récupérer vos secrets en appelant `GetSecretValue` ou `BatchGetSecretValue` dans l'un des AWS SDKs. Cependant, nous vous recommandons de mettre en cache vos valeurs de secret à l'aide de la mise en cache côté client. La mise en cache des secrets améliore la vitesse et réduit vos coûts.

Rubriques

- Obtenez une valeur secrète de Secrets Manager en utilisant Rust avec la mise en cache côté client
- Obtenez une valeur secrète de Secrets Manager à l'aide du AWS SDK Rust

Obtenez une valeur secrète de Secrets Manager en utilisant Rust avec la mise en cache côté client

Lorsque vous récupérez un secret, vous pouvez utiliser le composant de mise en cache basé sur Rust de Secrets Manager pour le mettre en cache pour une utilisation future. Il est plus rapide de récupérer un secret mis en cache que de le récupérer à partir de Secrets Manager. L'appel de Secrets Manager étant payant APIs, l'utilisation d'un cache peut réduire vos coûts. Pour connaître toutes les manières dont vous pouvez récupérer des secrets, consultezObtenez des secrets.

La politique du cache est le premier entré, premier sorti (FIFO). Ainsi, lorsque le cache doit supprimer un secret, il supprime le secret le plus ancien. Par défaut, le cache actualise les secrets toutes les heures. Vous pouvez configurer les éléments suivants :

- `max_size`— Le nombre maximum de secrets mis en cache à conserver avant d'expulser des secrets qui n'ont pas été consultés récemment.
- `ttl`— La durée pendant laquelle un élément mis en cache est considéré comme valide avant de nécessiter une actualisation de l'état secret.

L'implémentation du cache n'inclut pas l'invalidation du cache. L'implémentation du cache se concentre sur le cache lui-même et n'est pas renforcée ou ciblée sur le plan de la sécurité. Si vous avez besoin d'une sécurité supplémentaire, telle que le chiffrement des éléments du cache, utilisez les caractéristiques fournies pour modifier le cache.

Pour utiliser le composant, vous devez disposer d'un environnement de développement Rust 2021 avec`tokio`. Pour plus d'informations, consultez Getting started sur le site Web du langage de programmation Rust.

Pour télécharger le code source, voir le composant client de mise en cache basé sur Rust de Secrets Manager sur. GitHub

Pour installer le composant de mise en cache, utilisez la commande suivante.

```
cargo add aws_secretsmanager_caching
```

Autorisations requises :

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

Pour de plus amples informations, veuillez consulter Référence des autorisations .

Example Récupérer un secret

L'exemple suivant montre comment obtenir la valeur secrète d'un secret nommé*`MyTest`*.

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

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

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

// Your code here
```

Example Instanciation du cache avec une configuration personnalisée et un client personnalisé

L'exemple suivant montre comment configurer le cache puis obtenir la valeur secrète d'un secret nommé*MyTest*.

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

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

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

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

// Your code here
```

Obtenez une valeur secrète de Secrets Manager à l'aide du AWS SDK Rust

Dans les applications, vous pouvez récupérer vos secrets en appelant `GetSecretValue` ou `BatchGetSecretValue` dans l'un des AWS SDKs. Cependant, nous vous recommandons de mettre en cache vos valeurs de secret à l'aide de la mise en cache côté client. La mise en cache des secrets améliore la vitesse et réduit vos coûts.

Pour les applications Rust, utilisez le composant de mise en cache basé sur Rust de Secrets Manager ou appelez le SDK directement avec ou. GetSecretValue BatchGetSecretValue

L'exemple de code suivant montre comment obtenir une valeur de Secrets Secrets Manager.

Autorisations requises : `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(())
}
```

Utiliser des AWS Secrets Manager secrets dans Amazon Elastic Kubernetes Service

Pour afficher les secrets AWS Secrets Manager (ASCP) sous forme de fichiers montés dans des pods Amazon EKS, vous pouvez utiliser le fournisseur de AWS secrets et de configuration pour le pilote CSI Kubernetes Secrets Store. L'ASCP fonctionne avec Amazon Elastic Kubernetes Service 1.17+ exécutant un groupe de nœuds Amazon. EC2 AWS Fargate les groupes de nœuds ne sont pas pris en charge. Avec l'ASCP, vous pouvez stocker et gérer vos secrets dans Secrets Manager, puis les récupérer via vos applications exécutées sur Amazon EKS. Si votre secret contient plusieurs paires clé-valeur au format JSON, vous pouvez choisir celles à monter dans Amazon EKS. L'ASCP utilise JMESPath la syntaxe pour interroger les paires clé-valeur de votre secret. L'ASCP fonctionne également avec les paramètres du Parameter Store. L'ASCP propose deux méthodes d'authentification avec Amazon EKS. La première approche utilise les rôles IAM pour les comptes de service (IRSA). La deuxième approche utilise Pod Identities. Chaque approche a ses avantages et ses cas d'utilisation.

ASCP avec rôles IAM pour les comptes de service (IRSA)

L'ASCP avec rôles IAM pour les comptes de service (IRSA) vous permet de monter des secrets AWS Secrets Manager sous forme de fichiers dans vos pods Amazon EKS. Cette approche convient lorsque :

- Vous devez monter les secrets sous forme de fichiers dans vos pods.
- Vous utilisez Amazon EKS version 1.17 ou ultérieure avec des groupes de EC2 nœuds Amazon.
- Vous souhaitez récupérer des paires clé-valeur spécifiques à partir de secrets au format JSON.

Pour de plus amples informations, veuillez consulter the section called "Intégrer ASCP à IRSA pour Amazon EKS".

ASCP avec Pod Identity

ASCP avec EKS Pod Identity

La méthode ASCP avec Pod Identity améliore la sécurité et simplifie la configuration pour accéder aux secrets dans Amazon EKS. Cette approche est bénéfique lorsque :

- Vous avez besoin d'une gestion des autorisations plus précise au niveau du Pod.
- Vous utilisez Amazon EKS version 1.24 ou ultérieure.
- Vous souhaitez améliorer les performances et l'évolutivité.

Pour de plus amples informations, veuillez consulter the section called "Intégrer ASCP à Pod Identity pour Amazon EKS".

Choix de la bonne approche

Tenez compte des facteurs suivants lorsque vous choisissez entre ASCP avec IRSA et ASCP avec Pod Identity :

- Amazon EKSversion : Pod Identity nécessite Amazon EKS 1.24+, tandis que le pilote CSI fonctionne avec Amazon EKS 1.17+.
- Exigences de sécurité : Pod Identity offre un contrôle plus granulaire au niveau du Pod.
- Performances : Pod Identity fonctionne généralement mieux dans les environnements à grande échelle.

- Complexité : Pod Identity simplifie la configuration en éliminant le besoin de comptes de service distincts.

Choisissez la méthode qui correspond le mieux à vos exigences spécifiques et à l'environnement Amazon EKS.

Installation d'ASCP pour Amazon EKS

Cette section explique comment installer le fournisseur de AWS secrets et de configuration pour Amazon EKS. Avec ASCP, vous pouvez monter des secrets à partir de Secrets Manager et des paramètres AWS Systems Manager sous forme de fichiers dans Amazon EKS Pods.

Prérequis

- Un cluster Amazon EKS
  - Version 1.24 ou ultérieure pour Pod Identity
  - Version 1.17 ou ultérieure pour IRSA
- Le AWS CLI installé et configuré
- kubectl installé et configuré pour votre cluster Amazon EKS
- Helm (version 3.0 ou ultérieure)

Installation et configuration de l'ASCP

L'ASCP est disponible GitHub dans le référentiel secrets-store-csi-provider-aws. Le référentiel contient également des exemples de fichiers YAML pour créer et monter un secret.

Pendant l'installation, vous pouvez configurer l'ASCP pour utiliser un point de terminaison FIPS. Pour obtenir la liste des points de terminaison , consultez the section called “Points de terminaison Secrets Manager”.

Pour installer l'ASCP à l'aide de Helm

1. Pour vous assurer que le dépôt pointe vers les derniers graphiques, utilisez `helm repo update.`
2. Ajoutez le graphique des pilotes CSI de Secrets Store.

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

3. Installez le chart. Pour configurer la régulation, ajoutez l'indicateur suivant : `--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. Ajoutez le graphique ASCP.

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

5. Installez le chart. Pour utiliser un point de terminaison FIPS, ajoutez l'indicateur suivant : `--set useFipsEndpoint=true`

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

Pour installer en utilisant le code YAML dans le dépôt

- Utilisez les commandes suivantes.

```
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
```

Vérifiez les installations

Pour vérifier les installations de votre cluster EKS, du pilote CSI Secrets Store et du plug-in ASCP, procédez comme suit :

1. Vérifiez le cluster EKS :

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

Cette commande doit renvoyer des informations sur votre cluster.

2. Vérifiez l'installation du pilote CSI de Secrets Store :

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

Vous devriez voir des pods fonctionner avec des noms tels que`csi-secrets-store-secrets-store-csi-driver-xxx`.

3. Vérifiez l'installation du plugin ASCP :

YAML installation

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

Exemple de sortie :

```
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
```

Exemple de sortie :

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

Vous devriez voir Pods dans l'`Running`État.

Après avoir exécuté ces commandes, si tout est correctement configuré, vous devriez voir tous les composants fonctionner sans aucune erreur. Si vous rencontrez des problèmes, vous devrez peut-être les résoudre en consultant les journaux des pods spécifiques qui rencontrent des problèmes.

Résolution des problèmes

1. Pour consulter les journaux du fournisseur ASCP, exécutez :

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

2. Vérifiez l'état de tous les pods de l'espace de `kube-system` noms :

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

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

   Tous les pods liés au pilote CSI et à l'ASCP doivent être dans l'état « En cours d'exécution ».

3. Vérifiez la version du pilote CSI :

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

   Cette commande doit renvoyer des informations sur le pilote CSI installé.

Ressources supplémentaires

Pour plus d'informations sur l'utilisation d'ASCP avec Amazon EKS, consultez les ressources suivantes :

- Utilisation de Pod Identity avec Amazon EKS
- AWS Pilote CSI Secrets Store activé GitHub

Utiliser AWS les secrets et le fournisseur de configuration CSI avec Pod Identity pour Amazon EKS

L'intégration du fournisseur de AWS secrets et de configuration à l'agent Pod Identity pour Amazon Elastic Kubernetes Service améliore la sécurité, simplifie la configuration et améliore les performances des applications exécutées sur Amazon EKS. Pod Identity simplifie l'authentification

IAM pour Amazon EKS lors de la récupération de secrets depuis Secrets Manager ou de paramètres depuis AWS Systems Manager Parameter Store.

Amazon EKS Pod Identity rationalise le processus de configuration des autorisations IAM pour les applications Kubernetes en permettant de configurer les autorisations directement via les interfaces Amazon EKS, en réduisant le nombre d'étapes et en éliminant le besoin de basculer entre les services Amazon EKS et IAM. Pod Identity permet d'utiliser un rôle IAM unique sur plusieurs clusters sans mettre à jour les politiques de confiance et prend en charge les balises de session de rôle pour un contrôle d'accès plus précis. Cette approche simplifie non seulement la gestion des politiques en permettant la réutilisation des politiques d'autorisation entre les rôles, mais elle améliore également la sécurité en autorisant l'accès aux AWS ressources en fonction des balises correspondantes.

Comment ça marche

1. Pod Identity attribue un rôle IAM au Pod.
2. ASCP utilise ce rôle pour s'authentifier auprès de. Services AWS
3. S'il est autorisé, l'ASCP récupère les secrets demandés et les met à la disposition du Pod.

Pour plus d'informations, consultez Comprendre le fonctionnement d'Amazon EKS Pod Identity dans le guide de l'utilisateur Amazon EKS.

Prérequis

> **Important**
>
> Pod Identity n'est pris en charge que pour Amazon EKS dans le cloud. Il n'est pas pris en charge pour Amazon EKS Anywhere Red Hat OpenShift Service on AWS, ni pour les clusters Kubernetes autogérés sur les instances Amazon. EC2

- Cluster Amazon EKS (version 1.24 ou ultérieure)
- Accès à un AWS CLI cluster Amazon EKS via `kubectl`
- Accès à deux Comptes AWS (pour un accès entre comptes)

Installation de l'agent d'identité Amazon EKS Pod

Pour utiliser Pod Identity avec votre cluster, vous devez installer le module complémentaire Amazon EKS Pod Identity Agent.

Pour installer l'agent Pod Identity

- Installez le module complémentaire Pod Identity Agent sur votre cluster :

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

Configurer ASCP avec Pod Identity

1. Créez une politique d'autorisation qui accorde `secretsmanager:GetSecretValue` une `secretsmanager:DescribeSecret` autorisation aux secrets auxquels le Pod a besoin pour accéder. Pour un exemple de politique, consultez the section called “Exemple : autorisation de lire et de décrire des secrets individuels”.
2. Créez un rôle IAM qui peut être assumé par le principal de service Amazon EKS pour Pod Identity :

   JSON

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

```
}
```

Associez la politique IAM au rôle :

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

3. Créez une association Pod Identity. Par exemple, consultez Créer une association Pod Identity dans le guide de l'utilisateur Amazon EKS
4. Créez le `SecretProviderClass` qui spécifie les secrets à monter dans le Pod :

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

La principale différence SecretProviderClass entre IRSA et Pod Identity réside dans le paramètre `usePodIdentity` optionnel. Il s'agit d'un champ facultatif qui détermine l'approche d'authentification. Lorsqu'il n'est pas spécifié, il utilise par défaut les rôles IAM pour les comptes de service (IRSA).

- Pour utiliser EKS Pod Identity, utilisez l'une des valeurs suivantes :`"true", "True", "TRUE", "t", "T"`.
- Pour utiliser IRSA de manière explicite, définissez l'une des valeurs suivantes :`"false", "False", "FALSE", "f", or "F"`.

5. Déployez le pod qui contient les secrets sous `/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. Si vous utilisez un cluster Amazon EKS privé, assurez-vous que le VPC dans lequel se trouve le cluster possède un AWS STS point de terminaison. Pour plus d'informations sur la création d'un point de terminaison, consultez la section Points de terminaison VPC d'interface dans le guide de l'AWS Identity and Access Management utilisateur.

Vérifiez le montage secret

Pour vérifier que le secret est correctement monté, exécutez la commande suivante :

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

Pour configurer Amazon EKS Pod Identity afin d'accéder aux secrets dans Secrets Manager

1. Créez une politique d'autorisation qui accorde `secretsmanager:GetSecretValue` une `secretsmanager:DescribeSecret` autorisation aux secrets auxquels le Pod a besoin pour accéder. Pour un exemple de politique, consultez the section called “Exemple : autorisation de lire et de décrire des secrets individuels”.
2. Créez un secret dans Secrets Manager, si vous n'en avez pas déjà un.

Dépannage

Vous pouvez visualiser la plupart des erreurs en décrivant le déploiement du Pod.

Pour afficher les messages d'erreur pour votre conteneur

1. Obtenez une liste des noms de Pod à l'aide de la commande suivante. Si vous n'utilisez pas l'espace de noms par défaut, utilisez `-n` *NAMESPACE*.

   ```
   kubectl get pods
   ```

2. Pour décrire le Pod, dans la commande suivante, *PODID* utilisez l'ID du Pod indiqué dans les Pods que vous avez trouvés à l'étape précédente. Si vous n'utilisez pas l'espace de noms par défaut, utilisez `-n` *NAMESPACE*.

   ```
   kubectl describe pod/PODID
   ```

Pour voir les erreurs pour l'ASCP

- Pour obtenir plus d'informations dans les journaux du fournisseur, utilisez la commande suivante pour *PODID* utiliser l'ID du module csi-secrets-store-provider-aws.

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

Utiliser AWS les secrets et le fournisseur de configuration CSI avec des rôles IAM pour les comptes de service (IRSA)

Rubriques

- Prérequis
- Configurer le contrôle d'accès
- Identifier les secrets à monter
- Dépannage

Prérequis

- Cluster Amazon EKS (version 1.17 ou ultérieure)
- Accès à un AWS CLI cluster Amazon EKS via `kubectl`

Configurer le contrôle d'accès

L'ASCP récupère l'identité du pod Amazon EKS et l'échange contre un rôle IAM. Vous définissez des autorisations dans une politique IAM pour ce rôle IAM. Lorsque l'ASCP assume le rôle IAM, il a accès aux secrets que vous avez autorisés. Les autres conteneurs ne peuvent pas accéder aux secrets sauf si vous les associez également au rôle IAM.

Pour autoriser votre Amazon EKS Pod à accéder aux secrets dans Secrets Manager

1. Créez une politique d'autorisation qui accorde `secretsmanager:GetSecretValue` une `secretsmanager:DescribeSecret` autorisation aux secrets auxquels le Pod a besoin pour accéder. Pour un exemple de politique, consultez the section called “Exemple : autorisation de lire et de décrire des secrets individuels”.
2. Créez un fournisseur OpenID Connect (OIDC) IAM pour le cluster si vous n'en avez pas déjà un. Pour plus d'informations, consultez la section Créer un fournisseur IAM OIDC pour votre cluster dans le guide de l'utilisateur Amazon EKS.
3. Créez un rôle IAM pour le compte de service et associez-y la politique. Pour plus d'informations, consultez la section Créer un rôle IAM pour un compte de service dans le guide de l'utilisateur Amazon EKS.
4. Si vous utilisez un cluster Amazon EKS privé, assurez-vous que le VPC dans lequel se trouve le cluster possède un AWS STS point de terminaison. Pour plus d'informations sur la création d'un

point de terminaison, consultez la section Points de terminaison VPC d'interface dans le guide de l'AWS Identity and Access Management utilisateur.

Identifier les secrets à monter

Pour déterminer quels secrets l'ASCP monte dans Amazon EKS en tant que fichiers du système de fichiers, vous devez créer un fichier YAML the section called "SecretProviderClass". La `SecretProviderClass` liste répertorie les secrets à monter et le nom du fichier sous lequel les monter. `SecretProviderClass`Il doit se trouver dans le même espace de noms que le pod Amazon EKS auquel il fait référence.

Montez les secrets sous forme de fichiers

Les instructions suivantes montrent comment monter des secrets sous forme de fichiers à l'aide d'exemples de fichiers YAML .yaml et ExampleSecretProviderClass.yaml. ExampleDeployment

Pour monter des secrets dans Amazon EKS

1. Appliquez le `SecretProviderClass` sur le pod :

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

2. Déployez votre Pod :

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

3. L'ASCP monte les fichiers.

Dépannage

Vous pouvez visualiser la plupart des erreurs en décrivant le déploiement du Pod.

Pour afficher les messages d'erreur pour votre conteneur

1. Obtenez une liste des noms de Pod à l'aide de la commande suivante. Si vous n'utilisez pas l'espace de noms par défaut, utilisez `-n` *`nameSpace`*.

```
kubectl get pods
```

2. Pour décrire le Pod, dans la commande suivante, *podId* utilisez l'ID du Pod indiqué dans les Pods que vous avez trouvés à l'étape précédente. Si vous n'utilisez pas l'espace de noms par défaut, utilisez `-n` *nameSpace*.

```
kubectl describe pod/podId
```

Pour voir les erreurs pour l'ASCP

- Pour obtenir plus d'informations dans les journaux du fournisseur, utilisez la commande suivante pour *podId* utiliser l'ID du module csi-secrets-store-provider-aws.

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

- Vérifiez que le **`SecretProviderClass`** CRD est installé :

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

Cette commande doit renvoyer des informations sur la définition de ressource `SecretProviderClass` personnalisée.

- Vérifiez que l' SecretProviderClass objet a été créé.

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

AWS Exemples de code de secret et de fournisseur de configuration

Exemples d'authentification ASCP et de contrôle d'accès

Exemple : politique IAM autorisant le service Amazon EKS Pod Identity (pods.eks.amazonaws.com) à assumer le rôle et à étiqueter la session :

JSON

```
{
```

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

SecretProviderClass

Vous utilisez YAML pour décrire les secrets à monter dans Amazon EKS à l'aide de l'ASCP. Pour obtenir des exemples, consultez the section called "SecretProviderClass utilisation".

SecretProviderClass Structure YAML

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

Le champ des paramètres contient les détails de la demande de montage :

region

(Facultatif) Le nom Région AWS du secret. Si vous n'utilisez pas ce champ, ASCP va rechercher la région à partir de l'annotation sur le nœud. Cette recherche alourdit les demandes de montage.

Nous vous recommandons donc de fournir la région pour les clusters utilisant un grand nombre de pods.

Si vous spécifiez également `failoverRegion`, l'ASCP essaie de récupérer le secret des deux régions. Si l'une des régions renvoie une erreur 4xx, notamment en raison d'un problème d'authentification, l'ASCP ne monte aucun secret. Si le secret est récupéré avec succès de `region`, l'ASCP monte cette valeur secrète. Si le secret n'est pas récupéré correctement de `region`, mais qu'il l'est avec succès de `failoverRegion`, l'ASCP monte cette valeur secrète.

failoverRegion

(Facultatif) Si vous incluez ce champ, l'ASCP va essayer de récupérer le secret à partir des régions définies dans `region` et dans ce champ. Si l'une des régions renvoie une erreur 4xx, notamment en raison d'un problème d'authentification, l'ASCP ne monte aucun secret. Si le secret est récupéré avec succès de `region`, l'ASCP monte cette valeur secrète. Si le secret n'est pas récupéré correctement de `region`, mais qu'il l'est avec succès de `failoverRegion`, l'ASCP monte cette valeur secrète. Pour accéder à un exemple sur la façon de procéder, consultez Basculement secret multirégional.

pathTranslation

(Facultatif) Un caractère de substitution unique à utiliser si le nom de fichier dans Amazon EKS contient un caractère séparateur de chemin, tel que la barre oblique (/) sur Linux. L'ASCP ne peut pas créer un fichier monté possédant un caractère de séparation de chemin. Par contre, l'ASCP va remplacer le caractère séparateur de chemin par un autre caractère. Si vous n'utilisez pas ce champ, la valeur par défaut est le trait de soulignement (_). Par exemple, `My/Path/Secret` se monte en tant que `My_Path_Secret`.

Pour empêcher la substitution de caractères, entrez la chaîne `False`.

usePodIdentity

(Facultatif) Détermine l'approche d'authentification. Lorsqu'il n'est pas spécifié, la valeur par défaut est IAM Roles for Service Accounts (IRSA) (IRSA).

- Pour utiliser EKS Pod Identity, utilisez l'une des valeurs suivantes : `"true"` «`"True"`,`"TRUE"`,`"t"`, ou`"T"`.
- Pour utiliser IRSA de manière explicite, définissez l'une des valeurs suivantes :`"false"`,`"False"`, `"FALSE""f"`, ou `"F"` « = ».

preferredAddressType

(Facultatif) Spécifie le type d'adresse IP préféré pour les communications entre les terminaux du Pod Identity Agent. Le champ n'est applicable que lors de l'utilisation de la fonctionnalité EKS Pod Identity et sera ignoré lors de l'utilisation des rôles IAM pour les comptes de service. Les valeurs ne distinguent pas les majuscules et minuscules. Les valeurs valides sont :

- `"ipv4"`, `"IPv4"` «, ou `"IPV4"` — Forcer l'utilisation du point de IPv4 terminaison Pod Identity Agent
- `"ipv6""IPv6"`, ou `"IPV6"` — Forcer l'utilisation du point de IPv6 terminaison Pod Identity Agent
- non spécifié — Utilisez la sélection automatique du point de terminaison, en essayant d'abord le point de IPv4 terminaison et en revenant au IPv6 point de terminaison en cas d' IPv4 échec

objects

Chaîne contenant une déclaration YAML des secrets à monter. Nous vous recommandons d'utiliser une chaîne YAML multi-ligne ou un caractère pipe (|).

objectName

Obligatoire. Spécifie le nom du secret ou du paramètre à récupérer. Pour Secrets Manager, il s'agit du `SecretId`paramètre et il peut s'agir du nom convivial ou de l'ARN complet du secret. Pour le magasin de paramètres SSM, il s'agit `Name`du paramètre et il peut s'agir du nom ou de l'ARN complet du paramètre.

objectType

Obligatoire si vous n'utilisez pas d'ARN Secrets Manager pour `objectName`. Peut avoir la valeur `secretsmanager` ou `ssmparameter`.

objectAlias

(Facultatif) Le nom de fichier du secret dans le Pod Amazon EKS. Si vous ne spécifiez pas ce champ,`objectName` apparaît en tant que nom de fichier.

Autorisation du fichier

(Facultatif) Chaîne octale à 4 chiffres qui indique l'autorisation du fichier à utiliser pour le montage du secret. Si vous ne spécifiez pas ce champ, il sera défini par défaut sur`"0644"`.

objectVersion

(Facultatif) ID de version du secret. Déconseillé, car vous devez mettre à jour l'identifiant de la version à chaque fois que vous mettez le secret à jour. La version la plus récente est utilisée

par défaut. Si vous incluez un `failoverRegion`, ce champ représente le champ principal `objectVersion`.

objectVersionLabel

(Facultatif) Alias de la version. La version par défaut est la version la plus récente AWSCURRENT. Pour de plus amples informations, veuillez consulter the section called "Versions secrètes". Si vous incluez un `failoverRegion`, ce champ représente le champ principal `objectVersionLabel`.

jmesPath

(Facultatif) Carte des clés dans le secret des fichiers à monter dans Amazon EKS. Pour utiliser ce champ, votre valeur secrète doit être au format JSON. Si vous utilisez ce champ, vous devez inclure les sous-champs `path` et `objectAlias`.

path

Une clé issue d'une paire clé-valeur dans le JSON de la valeur secrète. Si le champ contient un trait d'union, utilisez des guillemets simples pour l'ignorer, par exemple : `path: '"hyphenated-path"'`

objectAlias

Nom du fichier à monter dans le module Amazon EKS. Si le champ contient un trait d'union, utilisez des guillemets simples pour l'ignorer, par exemple : `objectAlias: '"hyphenated-alias"'`

Autorisation du fichier

(Facultatif) Chaîne octale à 4 chiffres qui indique l'autorisation du fichier à utiliser pour le montage du secret. Si vous ne spécifiez pas ce champ, l'autorisation de fichier de l'objet parent sera utilisée par défaut.

failoverObject

(Facultatif) Si vous spécifiez ce champ, l'ASCP essaie de récupérer à la fois le secret spécifié dans le champ principal `objectName` et le secret spécifié dans le sous-champ `failoverObject` `objectName`. Si l'un ou l'autre renvoie une erreur 4xx, notamment pour un problème d'authentification, l'ASCP ne monte aucun secret. Si le secret est récupéré avec succès à partir du `objectName` principal, l'ASCP monte cette valeur secrète. Si le secret n'est pas récupéré correctement depuis le `objectName` principal, mais qu'il est récupéré correctement depuis le basculement `objectName`, l'ASCP monte cette valeur secrète. Si vous

incluez ce champ, vous devez inclure le champ `objectAlias`. Pour accéder à un exemple sur la façon de procéder, consultez Basculement vers un autre secret.

Vous allez généralement utiliser ce champ lorsque le secret de basculement n'est pas une réplique. Pour obtenir un exemple de spécification des capacités, consultez Basculement secret multirégional.

objectName

Nom ou ARN complet du secret de basculement. Si vous utilisez un ARN, la région de l'ARN doit correspondre au champ `failoverRegion`.

objectVersion

(Facultatif) ID de version du secret. Doit correspondre au principal `objectVersion`. Déconseillé, car vous devez mettre à jour l'identifiant de la version à chaque fois que vous mettez le secret à jour. La version la plus récente est utilisée par défaut.

objectVersionLabel

(Facultatif) Alias de la version. La version par défaut est la version la plus récente AWSCURRENT. Pour de plus amples informations, veuillez consulter the section called "Versions secrètes".

Créez une SecretProviderClass configuration de base pour monter des secrets dans vos pods Amazon EKS.

Pod Identity

SecretProviderClass pour utiliser un secret dans le même cluster Amazon EKS :

```
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 utilisation

Utilisez ces exemples pour créer des SecretProviderClass configurations pour différents scénarios.

Exemple : Monter des secrets par nom ou ARN

Cet exemple montre comment monter trois types de secrets différents :

- Un secret spécifié par un ARN complet
- Un secret spécifié par son nom
- Une version spécifique d'un secret

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

Exemple : monter des paires clé-valeur à partir d'un secret

Cet exemple montre comment monter des paires clé-valeur spécifiques à partir d'un secret au format JSON :

```
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
```

Exemple : monter des secrets par autorisation de fichier

Cet exemple montre comment monter un secret avec une autorisation de fichier spécifique

```
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"
```

Exemple : exemples de configuration de basculement

Ces exemples montrent comment configurer le basculement pour les secrets.

Basculement secret multirégional

Cet exemple montre comment configurer le basculement automatique pour un secret répliqué dans plusieurs régions :

```
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"
```

Basculement vers un autre secret

Cet exemple montre comment configurer le basculement vers un autre secret (pas une réplique) :

```
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"
```

Ressources supplémentaires

Pour plus d'informations sur l'utilisation d'ASCP avec Amazon EKS, consultez les ressources suivantes :

- Utilisation de Pod Identity avec Amazon EKS
- Utilisation des AWS secrets et du fournisseur de configuration
- AWS Pilote CSI Secrets Store activé GitHub

Utiliser des AWS Secrets Manager secrets dans les AWS Lambda fonctions

AWS Lambda est un service informatique sans serveur qui vous permet d'exécuter du code sans devoir approvisionner ou gérer des serveurs. Parameter Store, une des fonctionnalités de AWS Systems Manager, offre un stockage sécurisé et hiérarchique des données pour la gestion des données de configuration et des codes secrets. Vous pouvez utiliser l'extension Lambda AWS Parameters and Secrets pour récupérer et mettre en cache les AWS Secrets Manager secrets et les paramètres de Parameter Store dans les fonctions Lambda sans utiliser de kit SDK. Pour des informations détaillées sur l'utilisation de cette extension, consultez la section Utiliser les secrets du Gestionnaire de secrets dans les fonctions Lambda dans le Guide du développeur Lambda.

Utilisation des secrets Secrets Manager avec Lambda

Le guide du développeur Lambda fournit des instructions complètes sur l'utilisation des secrets de Secrets Manager dans les fonctions Lambda. Mise en route :

1. Suivez le step-by-step didacticiel de la section Utiliser les secrets du Gestionnaire de secrets dans les fonctions Lambda, qui inclut :
   - Création d'une fonction Lambda avec votre environnement d'exécution préféré (Python, Node.js, Java)
   - Ajout de l'extension Lambda AWS Parameters and Secrets en tant que couche
   - Configuration des autorisations nécessaires
   - Écrire du code pour récupérer les secrets de l'extension
   - Test de votre fonction
2. Découvrez les variables d'environnement permettant de configurer le comportement de l'extension, notamment les paramètres du cache et les délais d'expiration

3. Découvrez les bonnes pratiques d'utilisation de la rotation secrète

Utilisation de Secrets Manager et Lambda dans un VPC

Si votre fonction Lambda s'exécute dans un VPC, vous devez créer un point de terminaison de VPC afin que l'extension puisse appeler Secrets Manager. Pour de plus amples informations, veuillez consulter the section called "Points de terminaison d'un VPC (AWS PrivateLink)".

Utilisation de l'extension Lambda AWS Parameters and Secrets

L'extension peut récupérer les secrets de Secrets Manager et les paramètres du Parameter Store. Pour obtenir des informations détaillées sur l'utilisation des paramètres du magasin de paramètres avec l'extension, consultez la section Utilisation des paramètres du magasin de paramètres dans les fonctions Lambda du guide de l'AWS Systems Manager utilisateur.

La documentation de Systems Manager inclut :

- Explication détaillée du fonctionnement de l'extension avec Parameter Store
- Instructions pour ajouter l'extension à une fonction Lambda
- Variables d'environnement pour configurer l'extension
- Exemples de commandes pour récupérer les paramètres
- Liste complète des extensions ARNs pour toutes les architectures et régions prises en charge

Utilisation de l' AWS Secrets Manager agent

Comment fonctionne l'agent Secrets Manager

L' AWS Secrets Manager Agent est un service HTTP côté client qui vous aide à normaliser la façon dont vous consommez les secrets de Secrets Manager dans vos environnements informatiques. Vous pouvez l'utiliser avec les services suivants :

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

L'agent Secrets Manager récupère et met en cache les secrets en mémoire, ce qui permet à vos applications d'obtenir des secrets auprès de localhost au lieu de passer des appels directs à Secrets Manager. L'agent Secrets Manager peut uniquement lire les secrets, il ne peut pas les modifier.

> **Important**
>
> L'agent Secrets Manager utilise les AWS informations d'identification de votre environnement pour appeler Secrets Manager. Il inclut une protection contre la falsification des requêtes côté serveur (SSRF) afin d'améliorer la sécurité des secrets. L'agent Secrets Manager utilise l'échange de clés ML-KEM post-quantique comme échange de clés le plus prioritaire par défaut.

Comprendre la mise en cache des agents Secrets Manager

L'agent Secrets Manager utilise un cache en mémoire qui se réinitialise au redémarrage de l'agent Secrets Manager. Il actualise régulièrement les valeurs secrètes mises en cache en fonction des éléments suivants :

- La fréquence de rafraîchissement par défaut (TTL) est de 300 secondes
- Vous pouvez modifier le TTL à l'aide d'un fichier de configuration
- L'actualisation a lieu lorsque vous demandez un secret après l'expiration du TTL

> **Note**
>
> L'agent Secrets Manager n'inclut pas l'invalidation du cache. Si un secret change avant l'expiration de l'entrée du cache, l'agent Secrets Manager peut renvoyer une valeur secrète périmée.

L'agent Secrets Manager renvoie les valeurs secrètes dans le même format que la réponse de`GetSecretValue`. Les valeurs secrètes ne sont pas chiffrées dans le cache.

Rubriques

- Créer l'agent Secrets Manager
- Installer l'agent Secrets Manager

- Récupérez des secrets avec l'agent Secrets Manager
- Comprendre le refreshNow paramètre
- Configuration de l'agent Secrets Manager
- Fonctionnalités optionnelles
- Journalisation
- Considérations sur la sécurité

Créer l'agent Secrets Manager

Avant de commencer, assurez-vous que les outils de développement standard et les outils Rust sont installés pour votre plateforme.

> **Note**
>
> La création de l'agent avec la `fips` fonctionnalité activée sur macOS nécessite actuellement la solution de contournement suivante :
>
> - Créez une variable d'environnement appelée `SDKROOT` qui est définie sur le résultat de l'exécution `xcrun --show-sdk-path`

RPM-based systems

Pour s'appuyer sur des systèmes basés sur le RPM

1. Utilisez le `install` script fourni dans le référentiel.

   Le script génère un jeton SSRF aléatoire au démarrage et le stocke dans le fichier`/var/run/awssmatoken`. Le jeton est lisible par le `awssmatokenreader` groupe créé par le script d'installation.
2. Pour permettre à votre application de lire le fichier de jetons, vous devez ajouter au `awssmatokenreader` groupe le compte utilisateur sous lequel votre application s'exécute. Par exemple, vous pouvez autoriser votre application à lire le fichier de jetons à l'aide de la commande usermod suivante, où *<APP_USER>* est l'ID utilisateur sous lequel votre application s'exécute.

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

Installer les outils de développement

Sur les systèmes basés sur le RPM tels que AL2 023, installez le groupe d'outils de développement :

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

3. Installer Rust

   Suivez les instructions de la section Installer Rust dans la documentation Rust :

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

4. Création de l'agent

   Créez l'agent Secrets Manager à l'aide de la commande cargo build :

   ```
   cargo build --release
   ```

   Vous trouverez l'exécutable sous `target/release/aws_secretsmanager_agent`.

Debian-based systems

Pour s'appuyer sur des systèmes Debian

1. Installer les outils de développement

   Sur les systèmes Debian tels qu'Ubuntu, installez le package build-essential :

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

2. Installer Rust

   Suivez les instructions de la section Installer Rust dans la documentation Rust :

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

3. Création de l'agent

   Créez l'agent Secrets Manager à l'aide de la commande cargo build :

   ```
   cargo build --release
   ```

   Vous trouverez l'exécutable sous `target/release/aws_secretsmanager_agent`.

Windows

Pour créer sous Windows

1. Configuration de l'environnement de développement

   Suivez les instructions de la section Configurer votre environnement de développement sous Windows pour Rust dans la documentation Microsoft Windows.

2. Création de l'agent

   Créez l'agent Secrets Manager à l'aide de la commande cargo build :

   ```
   cargo build --release
   ```

   Vous trouverez l'exécutable sous `target/release/aws_secretsmanager_agent.exe`.

Cross-compile natively

Pour effectuer une compilation croisée en mode natif

1. Installation d'outils de compilation croisée

   Sur les distributions où le package mingw-w64 est disponible, comme Ubuntu, installez la chaîne d'outils de compilation croisée :

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

2. Ajouter des cibles de build Rust

Installez la cible de compilation Windows GNU :

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

3. Génération pour Windows

   Compilez l'agent pour Windows de manière croisée :

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

   Vous trouverez le fichier exécutable à l'adresse`target/x86_64-pc-windows-gnu/release/aws_secretsmanager_agent.exe`.

Cross compile with Rust cross

Pour effectuer une compilation croisée à l'aide de Rust cross

Si les outils de compilation croisée ne sont pas disponibles nativement sur le système, vous pouvez utiliser le projet croisé Rust. Pour plus d'informations, voir https://github.com/cross-rs/croix.

> **Important**
>
> Nous recommandons 32 Go d'espace disque pour l'environnement de génération.

1. Configuration de Docker

   Installer et configurer Docker :

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

2. Configuration des autorisations Docker

   Ajoutez votre utilisateur au groupe de dockers :

   ```
   # Give ourselves permission to run the docker images without sudo
   ```

```
sudo usermod -aG docker $USER
newgrp docker
```

3. Génération pour Windows

   Installez cross et compilez l'exécutable :

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

Installer l'agent Secrets Manager

Choisissez votre environnement informatique parmi les options d'installation suivantes.

Amazon EC2

Pour installer l'agent Secrets Manager sur Amazon EC2

1. Accédez au répertoire de configuration

   Accédez au répertoire de configuration :

   ```
   cd aws_secretsmanager_agent/configuration
   ```

2. Exécutez le script d'installation

   Exécutez le `install` script fourni dans le référentiel.

   Le script génère un jeton SSRF aléatoire au démarrage et le stocke dans le fichier`/var/run/awssmatoken`. Le jeton est lisible par le `awssmatokenreader` groupe créé par le script d'installation.

3. Configuration des autorisations d'application

   Ajoutez au `awssmatokenreader` groupe le compte utilisateur sous lequel votre application s'exécute :

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

   Remplacez *APP_USER* par l'ID utilisateur sous lequel votre application s'exécute.

Container Sidecar

Vous pouvez exécuter l'agent Secrets Manager en tant que conteneur annexe à votre application à l'aide de Docker. Votre application peut ensuite récupérer les secrets depuis le serveur HTTP local fourni par l'agent Secrets Manager. Pour obtenir des informations sur Docker, veuillez consulter la documentation Docker.

Pour créer un conteneur annexe pour l'agent Secrets Manager

1. Créer un agent Dockerfile

   Créez un Dockerfile pour le conteneur annexe de l'agent Secrets Manager :

   ```
   # 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. Créer un fichier Dockerfile d'application

   Créez un Dockerfile pour votre application cliente.

3. Créer un fichier Docker Compose

   Créez un fichier Docker Compose pour exécuter les deux conteneurs avec une interface réseau partagée :

   > **Important**
   >
   > Vous devez charger les AWS informations d'identification et le jeton SSRF pour que l'application puisse utiliser l'agent Secrets Manager. Pour Amazon EKS et Amazon ECS, consultez les rubriques suivantes :

- Gérez l'accès dans le guide de l'utilisateur Amazon EKS
- Rôle IAM de la tâche Amazon ECS dans le manuel du développeur Amazon ECS

```
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. Agent de copie binaire

   Copiez le `secrets-manager-agent` fichier binaire dans le même répertoire que celui qui contient vos fichiers Dockerfiles et Docker Compose.

5. Créez et gérez des conteneurs

   Créez et exécutez les conteneurs à l'aide de Docker Compose :

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

6. Étapes suivantes

   Vous pouvez désormais utiliser l'agent Secrets Manager pour récupérer les secrets de votre conteneur client. Pour de plus amples informations, veuillez consulter the section called "Récupérez des secrets avec l'agent Secrets Manager".

Lambda

Vous pouvez empaqueter l'agent Secrets Manager sous forme d'extension Lambda. Vous pouvez ensuite l'ajouter à votre fonction Lambda sous forme de couche et appeler l'agent Secrets Manager depuis votre fonction Lambda pour obtenir des secrets.

Les instructions suivantes montrent comment obtenir un nom de secret à l'aide MyTestde l'exemple de script https://github.com/aws/aws-secretsmanager-agentpour installer l'agent Secrets Manager `secrets-manager-agent-extension.sh` en tant qu'extension Lambda.

Pour créer une extension Lambda pour l'agent Secrets Manager

1. Package de la couche d'agent

   À partir de la racine du package de code Secrets Manager, exécutez les commandes suivantes :

```
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. Configurer le jeton SSRF

   La configuration par défaut de l'agent définira automatiquement le jeton SSRF à la valeur définie dans les variables prédéfinies `AWS_SESSION_TOKEN` ou d'`AWS_CONTAINER_AUTHORIZATION_TOKEN`environnement (cette dernière variable est activée pour les fonctions Lambda). SnapStart Vous pouvez également définir la variable d'`AWS_TOKEN`environnement avec une valeur arbitraire pour votre fonction Lambda, car cette variable a priorité sur les deux autres. Si vous choisissez d'utiliser la variable d'`AWS_TOKEN`environnement, vous devez la définir par un `lambda:UpdateFunctionConfiguration` appel.

3. Attacher une couche à la fonction

   Associez la version de couche à votre fonction Lambda :

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

4. Mettre à jour le code de la fonction

   Mettez à jour votre fonction Lambda pour effectuer `http://localhost:2773/secretsmanager/get?secretId=MyTest` une requête avec la valeur d'`X-Aws-codes-Secrets-Token`en-tête définie sur la valeur du jeton SSRF provenant de l'une des variables d'environnement mentionnées ci-dessus afin de récupérer le secret. Veillez à implémenter une logique de nouvelle tentative dans le code de votre application afin de prendre en compte les délais d'initialisation et d'enregistrement de l'extension Lambda.

5. Tester la fonction

   Appelez la fonction Lambda pour vérifier que le secret est correctement extrait.

Récupérez des secrets avec l'agent Secrets Manager

Pour récupérer un secret, appelez le point de terminaison local de l'agent Secrets Manager avec le nom du secret ou l'ARN comme paramètre de requête. Par défaut, l'agent Secrets Manager récupère la `AWSCURRENT` version du secret. Pour récupérer une version différente, utilisez le paramètre VersionStage ou VersionId.

> **Important**
>
> Pour protéger l'agent Secrets Manager, vous devez inclure un en-tête de jeton SSRF dans chaque demande :`X-Aws-Parameters-Secrets-Token`. L'agent Secrets Manager refuse les demandes qui ne contiennent pas cet en-tête ou qui contiennent un jeton SSRF non valide. Vous pouvez personnaliser le nom de l'en-tête SSRF dans lethe section called "Options de configuration".

Autorisations requises

L'agent Secrets Manager utilise le AWS SDK pour Rust, qui utilise la chaîne de fournisseurs AWS d'informations d'identification. L'identité de ces informations d'identification IAM détermine les autorisations dont dispose l'agent Secrets Manager pour récupérer les secrets.

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

Pour plus d'informations sur les autorisations, consultez the section called "Référence des autorisations ".

> **Important**
>
> Une fois la valeur secrète saisie dans l'agent Secrets Manager, tout utilisateur ayant accès à l'environnement informatique et au jeton SSRF peut accéder au secret depuis le cache de l'agent Secrets Manager. Pour de plus amples informations, veuillez consulter the section called "Considérations sur la sécurité".

Exemples de demandes

curl

Example Exemple — Obtenir un secret en utilisant curl

L'exemple de curl suivant montre comment obtenir un secret de l'agent Secrets Manager. L'exemple repose sur la présence du SSRF dans un fichier, où il est stocké par le script d'installation.

```
curl -v -H \\
    "X-Aws-Parameters-Secrets-Token: $(/var/run/awssmatoken)" \\
    'http://localhost:2773/secretsmanager/get?secretId=YOUR_SECRET_ID' \\
    echo
```

Python

Example Exemple — Obtenir un secret en utilisant Python

L'exemple Python suivant montre comment obtenir un secret à partir de l'agent Secrets Manager. L'exemple repose sur la présence du SSRF dans un fichier, où il est stocké par le script d'installation.

```
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}")
```

Comprendre le **`refreshNow`** paramètre

L'agent Secrets Manager utilise un cache en mémoire pour stocker les valeurs secrètes, qu'il actualise régulièrement. Par défaut, cette actualisation a lieu lorsque vous demandez un secret après l'expiration du délai de vie (TTL), généralement toutes les 300 secondes. Cependant, cette approche peut parfois entraîner des valeurs secrètes périmées, en particulier si un secret change avant l'expiration de l'entrée du cache.

Pour pallier cette limitation, l'agent Secrets Manager prend en charge un paramètre appelé `refreshNow` dans l'URL. Vous pouvez utiliser ce paramètre pour forcer l'actualisation immédiate de la valeur d'un secret, en contournant le cache et en vous assurant de disposer du maximum up-to-date d'informations.

Comportement par défaut (sans**`refreshNow`**)

- Utilise les valeurs mises en cache jusqu'à l'expiration du TTL
- Actualise les secrets uniquement après TTL (300 secondes par défaut)
- Peut renvoyer des valeurs périmées si les secrets changent avant l'expiration du cache

Comportement avec **`refreshNow=true`**

- Contourne complètement le cache
- Récupère la dernière valeur secrète directement depuis Secrets Manager
- Met à jour le cache avec la nouvelle valeur et réinitialise le TTL
- Garantit que vous obtenez toujours la valeur secrète la plus récente

Renouvellement forcé une valeur de secret

> **Important**
>
> La valeur par défaut de `refreshNow` est `false`. Lorsqu'il est défini sur`true`, il remplace le TTL spécifié dans le fichier de configuration de l'agent Secrets Manager et envoie un appel d'API à Secrets Manager.

curl

Example Exemple — Actualisation forcée d'un secret à l'aide de curl

L'exemple de curl suivant montre comment forcer l'agent Secrets Manager à actualiser le secret. L'exemple repose sur la présence du SSRF dans un fichier, où il est stocké par le script d'installation.

```
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 Exemple — Actualisation forcée d'un secret à l'aide de Python

L'exemple Python suivant montre comment obtenir un secret à partir de l'agent Secrets Manager. L'exemple repose sur la présence du SSRF dans un fichier, où il est stocké par le script d'installation.

```
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}")
```

Configuration de l'agent Secrets Manager

Pour modifier la configuration de l'agent Secrets Manager, créez un fichier de configuration TOML, puis appelez `./aws_secretsmanager_agent --config config.toml`.

Options de configuration

`log_level`

Niveau de détail indiqué dans les journaux pour l'agent Secrets Manager : DEBUG, INFO, WARN, ERROR ou NONE. La valeur par défaut est INFO.

`log_to_file`

S'il faut se connecter à un fichier ou à stdout/stderr : ou. `true` `false` L'argument par défaut est `true`.

`http_port`

Port du serveur HTTP local, compris entre 1024 et 65535. La valeur par défaut est 2773.

`region`

AWS Région à utiliser pour les demandes. Si aucune région n'est spécifiée, l'agent Secrets Manager détermine la région à partir du SDK. Pour plus d'informations, consultez Spécifier vos informations d'identification et la région par défaut dans le guide du développeur du AWS SDK pour Rust.

`ttl_seconds`

Le TTL en secondes pour les éléments mis en cache, compris entre 0 et 3 600. La valeur par défaut est 300. Zéro signifie qu'aucune mise en cache n'est appliquée.

`cache_size`

Le nombre maximum de secrets pouvant être stockés dans le cache, compris entre 1 et 1 000. La valeur par défaut est 1000.

`ssrf_headers`

Une liste de noms d'en-têtes que l'agent Secrets Manager vérifie pour le jeton SSRF. La valeur par défaut est « X-Aws-Parameters-Secrets-Token ». X-Vault-Token

`ssrf_env_variables`

Une liste de noms de variables d'environnement que l'agent Secrets Manager vérifie dans l'ordre séquentiel pour le jeton SSRF. La variable d'environnement peut contenir le jeton ou une référence au fichier du jeton comme dans :`AWS_TOKEN=file:///var/run/awssmatoken`. La valeur par défaut est «AWS_TOKEN, AWS_SESSION _TOKEN, AWS_CONTAINER _AUTHORIZATION_TOKEN ».

`path_prefix`

Le préfixe d'URI utilisé pour déterminer si la demande est une demande basée sur le chemin. La valeur par défaut est « /v1/ ».

`max_conn`

Le nombre maximum de connexions depuis des clients HTTP autorisées par l'agent Secrets Manager, compris entre 1 et 1 000. La valeur par défaut est 800.

Fonctionnalités optionnelles

L'agent Secrets Manager peut être créé avec des fonctionnalités optionnelles en passant le `--features` drapeau à`cargo build`. Les fonctionnalités disponibles sont les suivantes :

Fonctions de génération

`prefer-post-quantum`

Génère `X25519MLKEM768` l'algorithme d'échange de clés ayant la priorité la plus élevée. Dans le cas contraire, il est disponible mais n'est pas prioritaire. `X25519MLKEM768`est un algorithme hybride d'échange de post-quantum-secure clés.

`fips`

Limite les suites de chiffrement utilisées par l'agent aux seuls chiffrements approuvés par FIPS.

Journalisation

Journalisation locale

L'agent Secrets Manager enregistre les erreurs localement dans le fichier `logs/secrets_manager_agent.log` ou dans stdout/stderr en fonction de la variable de configuration. `log_to_file` Lorsque votre application appelle l'agent Secrets Manager pour obtenir un secret, ces appels apparaissent dans le journal local. Elles n'apparaissent pas dans les CloudTrail journaux.

Renouvellement des journaux

L'agent Secrets Manager crée un nouveau fichier journal lorsque le fichier atteint 10 Mo, et il stocke jusqu'à cinq fichiers journaux au total.

AWS journalisation des services

Le journal n'est pas envoyé à Secrets Manager CloudTrail, ou CloudWatch. Les demandes d'obtention de secrets de la part de l'agent Secrets Manager n'apparaissent pas dans ces journaux. Lorsque l'agent Secrets Manager appelle Secrets Manager pour obtenir un secret, cet appel est enregistré CloudTrail avec une chaîne d'agent utilisateur contenant`aws-secrets-manager-agent`.

Vous pouvez configurer les options de journalisation dans lethe section called “Options de configuration”.

Considérations sur la sécurité

Domaine de confiance

Pour une architecture d'agent, le domaine de confiance correspond à l'endroit où le point de terminaison de l'agent et le jeton SSRF sont accessibles, c'est-à-dire généralement l'ensemble de l'hôte. Le domaine de confiance de l'agent Secrets Manager doit correspondre au domaine dans lequel les informations d'identification du Secrets Manager sont disponibles afin de maintenir le même niveau de sécurité. Par exemple, sur Amazon, EC2 le domaine de confiance de l'agent Secrets Manager serait le même que celui des informations d'identification lors de l'utilisation de rôles pour Amazon EC2.

Important

Les applications soucieuses de sécurité qui n'utilisent pas déjà une solution d'agent avec les informations d'identification de Secrets Manager verrouillées sur l'application devraient envisager d'utiliser des solutions spécifiques à la langue AWS SDKs ou de mise en cache. Pour plus d'informations, voir Obtenir des secrets.

Obtenez une valeur secrète de Secrets Manager à l'aide du AWS SDK C++

Pour les applications C++, appelez le SDK directement avec GetSecretValueou BatchGetSecretValue.

L'exemple de code suivant montre comment obtenir une valeur de Secrets Secrets Manager.

Autorisations requises : `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();
}
```

Obtenez une valeur secrète de Secrets Manager à l'aide du JavaScript AWS SDK

Pour JavaScript les applications, appelez le SDK directement avec `getSecretValue`ou `batchGetSecretValue`.

L'exemple de code suivant montre comment obtenir une valeur de Secrets Secrets Manager.

Autorisations requises : `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;
  }
};
```

Obtenez une valeur secrète de Secrets Manager à l'aide du SDK Kotlin AWS

Pour les applications Kotlin, appelez le SDK directement avec GetSecretValueou. BatchGetSecretValue

L'exemple de code suivant montre comment obtenir une valeur de Secrets Secrets Manager.

Autorisations requises : `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")
    }
}
```

Obtenez une valeur secrète de Secrets Manager à l'aide du AWS SDK PHP

Pour les applications PHP, appelez directement le SDK avec `GetSecretValue` ou `BatchGetSecretValue`.

L'exemple de code suivant montre comment obtenir une valeur de Secrets Secrets Manager.

Autorisations requises : `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
```

Obtenez une valeur secrète de Secrets Manager à l'aide du AWS SDK Ruby

Pour les applications Ruby, appelez directement le SDK avec `get_secret_value` ou `batch_get_secret_value`.

L'exemple de code suivant montre comment obtenir une valeur de Secrets Secrets Manager.

Autorisations requises : `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
```

Obtenez une valeur secrète à l'aide du AWS CLI

Autorisations requises : `secretsmanager:GetSecretValue`

Example Récupération de la valeur secrète cryptée d'un secret

L'exemple suivant `get-secret-value` récupère la valeur actuelle du secret.

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

Example Récupération de la valeur secrète précédente

L'exemple suivant `get-secret-value` récupère la valeur précédente du secret.

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

Obtenez un groupe de secrets par lots à l'aide du AWS CLI

Autorisations requises :

- `secretsmanager:BatchGetSecretValue`
- `secretsmanager:GetSecretValue`autorisation pour chaque secret que vous souhaitez récupérer.
- Si vous utilisez des filtres, vous devez également avoir `secretsmanager:ListSecrets`.

Pour obtenir un exemple de politique d'autorisations, consultez the section called "Exemple : autorisation de récupérer un groupe de valeurs secrètes dans un lot".

> **Important**
>
> Si vous avez une politique VPCE qui refuse l'autorisation de récupérer un secret individuel dans le groupe que vous recherchez, `BatchGetSecretValue` ne renverra aucune valeur de secret et renverra une erreur.

Example Récupération de la valeur de secret d'un groupe de secrets listés par nom

L'exemple `batch-get-secret-value` suivant récupère la valeur du secret pour trois secrets.

```
aws secretsmanager batch-get-secret-value \
          --secret-id-list MySecret1 MySecret2 MySecret3
```

Example Récupération de la valeur de secret d'un groupe de secrets sélectionnés par filtre

L'exemple `batch-get-secret-value` suivant récupère la valeur de secret pour les secrets dotés d'une balise nommée « Test ».

```
aws secretsmanager batch-get-secret-value \
          --filters Key="tag-key",Values="Test"
```

Obtenir une valeur secrète à l'aide de la AWS console

Pour récupérer un secret (console)

1. Ouvrez la console Secrets Manager à l'adresse https://console.aws.amazon.com/secretsmanager/.
2. Dans la liste des secrets, sélectionnez celui que vous souhaitez récupérer.

3. Dans la section Secret value (Valeur du secret), choisissez Retrieve secret value (Récupérer la valeur du secret).

   Secrets Manager affiche la version actuelle (`AWSCURRENT`) du secret. Pour afficher les autres versions du secret, telles que les versions étiquetées personnalisées ou `AWSPREVIOUS`, utilisez the section called "AWS CLI".

Utilisez AWS Secrets Manager des secrets dans AWS Batch

AWS Batch vous permet d'exécuter des charges de travail de calcul par lots sur le AWS Cloud. Avec AWS Batch, vous pouvez injecter des données sensibles dans vos tâches en stockant vos données sensibles en AWS Secrets Manager secret, puis en les référençant dans votre définition de tâche. Pour plus d'informations, consultez Spécification de données sensibles à l'aide de Secrets Manager.

Trouver un AWS Secrets Manager secret dans une AWS CloudFormation ressource

Avec AWS CloudFormation, vous pouvez récupérer un secret à utiliser dans une autre AWS CloudFormation ressource. Un scénario habituel consiste à créer d'abord un secret avec un mot de passe généré par Secrets Manager, à récupérer le nom d'utilisateur et le mot de passe du secret pour les utiliser comme des informations d'identification dans une nouvelle base de données. Pour plus d'informations sur la création de secrets avec AWS CloudFormation, consultezAWS CloudFormation.

Pour récupérer un secret dans un AWS CloudFormation modèle, vous devez utiliser une référence dynamique. Lorsque vous créez la pile, la référence dynamique introduit la valeur secrète dans la AWS CloudFormation ressource, de sorte que vous n'avez pas à coder en dur les informations secrètes. Au lieu de cela, vous vous référez au secret par son nom ou son ARN. Vous pouvez utiliser une référence dynamique pour un secret dans n'importe quelle propriété de ressource. Vous ne pouvez pas utiliser de référence dynamique pour un secret dans les métadonnées des ressources, à l'instar de `AWS::CloudFormation::Init`, car cela rendrait la valeur secrète visible dans la console.

Une référence dynamique pour un secret présente le modèle suivant :

```
{{resolve:secretsmanager:secret-id:SecretString:json-key:version-stage:version-id}}
```

secret-id

Nom ou ARN du secret. Pour accéder à un secret de votre AWS compte, vous pouvez utiliser le nom du secret. Pour accéder à un secret d'un autre AWS compte, utilisez l'ARN du secret.

json-key (Facultatif)

Le nom de clé de la paire clé-valeur dont vous voulez récupérer la valeur. Si vous ne spécifiez pas de`json-key`, AWS CloudFormation récupère l'intégralité du texte secret. Ce segment peut ne pas inclure le caractère deux-points (`:`).

version-stage (Facultatif)

La version du secret à utiliser. Secrets Manager utilise des étiquettes intermédiaires pour assurer le suivi des différentes versions pendant le processus de rotation. Si vous utilisez `version-stage`, ne spécifiez pas `version-id`. Si vous ne spécifiez ni `version-stage` ni `version-id`, alors la version par défaut est la version `AWSCURRENT`. Ce segment peut ne pas inclure le caractère deux-points (`:`).

version-id (Facultatif)

L'identifiant unique de la version du secret à utiliser. Si vous spécifiez `version-id`, ne spécifiez pas `version-stage`. Si vous ne spécifiez ni `version-stage` ni `version-id`, alors la version par défaut est la version `AWSCURRENT`. Ce segment peut ne pas inclure le caractère deux-points (`:`).

Pour de plus amples informations, veuillez consulter Utilisation de références dynamiques pour spécifier les secrets Secrets Manager.

> **Note**
>
> Ne créez pas de référence dynamique en utilisant une barre oblique inversée (`\`) comme valeur finale. AWS CloudFormation ne peut pas résoudre ces références, ce qui entraîne une défaillance des ressources.

Utilisez AWS Secrets Manager les secrets dans les GitHub emplois

Pour utiliser un secret dans une GitHub tâche, vous pouvez utiliser une GitHub action pour récupérer des secrets AWS Secrets Manager et les ajouter en tant que variables d'environnement masquées

dans votre GitHub flux de travail. Pour plus d'informations sur GitHub les actions, voir Comprendre GitHub les actions dans la GitHub documentation.

Lorsque vous ajoutez un secret à votre GitHub environnement, il est accessible à toutes les autres étapes de votre GitHub travail. Suivez les instructions de la section Renforcement de la sécurité pour les GitHub actions afin d'éviter que les secrets de votre environnement ne soient utilisés à mauvais escient.

Vous pouvez définir la chaîne complète de la valeur du secret comme valeur de variable d'environnement. Sinon, si la chaîne est au format JSON, vous pouvez analyser le fichier JSON pour définir des variables d'environnement individuelles pour chaque paire clé-valeur JSON. Si la valeur du secret est binaire, l'action la convertit en chaîne.

Pour afficher les variables d'environnement créées à partir de vos secrets, activez la journalisation du débogage. Pour plus d'informations, voir Activation de la journalisation du débogage dans la GitHub documentation.

Pour utiliser les variables d'environnement créées à partir de vos secrets, consultez la section Variables d'environnement dans la GitHub documentation.

Prérequis

Pour utiliser cette action, vous devez d'abord configurer les AWS informations d'identification et les définir Région AWS dans votre GitHub environnement à l'aide de l'`configure-aws-credentials`étape. Suivez les instructions de la section Configurer l'action des AWS informations d'identification pour que GitHub les actions assument le rôle directement à l'aide du fournisseur GitHub OIDC. Cela vous permet d'utiliser des informations d'identification de courte durée et d'éviter de stocker des clés d'accès supplémentaires en dehors de Secrets Manager.

Le rôle IAM assumé par l'action doit avoir les autorisations suivantes :

- `GetSecretValue` sur les secrets que vous souhaitez récupérer ;
- `ListSecrets` sur tous les secrets ;
- (Facultatif) `Decrypt` KMS key si les secrets sont chiffrés avec un clé gérée par le client.

Pour de plus amples informations, veuillez consulter the section called “Authentification et contrôle d'accès”.

Utilisation

Pour utiliser l'action, ajoutez une étape à votre flux qui utilise la syntaxe suivante.

```
- 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
```

Paramètres

`secret-ids`

ARN, noms et préfixes de nom du secret.

Pour définir le nom de la variable d'environnement, saisissez-le avant l'ID du secret, suivi d'une virgule. Par exemple, `ENV_VAR_1, secretId` crée une variable d'environnement nommée ENV_VAR_1 à partir du `secretId` du secret. Le nom de la variable d'environnement peut contenir des lettres majuscules, des chiffres et des traits de soulignement.

Pour utiliser un préfixe, saisissez au moins trois caractères suivis d'un astérisque. Par exemple, `dev*` correspond à tous les secrets dont le nom commence par dev. Le nombre maximum de secrets correspondants pouvant être récupérés est de 100. Si vous définissez le nom de la variable et que le préfixe correspond à plusieurs secrets, l'action échoue.

`name-transformation`

Par défaut, l'étape crée chaque nom de variable d'environnement à partir du nom du secret, transformé de façon à inclure uniquement des lettres majuscules, des chiffres et des traits de soulignement, et à ne pas commencer par un chiffre. Pour les lettres du nom, vous pouvez configurer l'étape pour utiliser des lettres minuscules avec `lowercase` ou pour ne pas changer les majuscules des lettres avec. `none` La valeur par défaut est `uppercase`.

`parse-json-secrets`

(Facultatif) Par défaut, l'action définit la valeur de la variable d'environnement sur la chaîne JSON entière de la valeur du secret. Définissez sur `parse-json-secrets` pour `true` créer des variables d'environnement pour chaque paire clé-valeur dans le JSON.

Notez que si le fichier JSON utilise des clés sensibles à la casse, par exemple « nom » et « Nom », l'action rencontrera des conflits de noms dupliqués. Dans ce cas, définissez `parse-json-secrets` sur `false` et analysez la valeur du secret JSON séparément.

Désignation des variables d'environnement

Les variables d'environnement créées par l'action portent le même nom que les secrets dont elles proviennent. Les variables d'environnement ont des exigences de dénomination plus strictes que les secrets. L'action transforme donc les noms des secrets pour répondre à ces exigences. Par exemple, l'action transforme les lettres minuscules en lettres majuscules. Si vous analysez le JSON du secret, le nom de la variable d'environnement inclut à la fois le nom du secret et le nom de la clé JSON, par exemple`MYSECRET_KEYNAME`. Vous pouvez configurer l'action pour ne pas transformer les lettres minuscules.

Si deux variables d'environnement finissent par porter le même nom, l'action échoue. Dans ce cas, vous devez spécifier les noms que vous souhaitez utiliser pour les variables d'environnement sous forme d'alias.

Exemples de situations dans lesquelles les noms peuvent entrer en conflit :

- Un secret nommé « MySecret » et un secret nommé « mysecret » deviendraient tous deux des variables d'environnement nommées « MYSECRET ».
- Un secret nommé « Secret_keyname » et un secret analysé en JSON nommé « Secret » avec une clé nommée « keyname » deviendraient tous deux des variables d'environnement nommées « SECRET_KEYNAME ».

Vous pouvez définir le nom de la variable d'environnement en spécifiant un alias, comme illustré dans l'exemple suivant, qui crée une variable nommée`ENV_VAR_NAME`.

```
secret-ids: |
  ENV_VAR_NAME, secretId2
```

Alias vides

- Si vous définissez `parse-json-secrets: true` et entrez un alias vide, suivi d'une virgule puis de l'ID secret, l'action nomme la variable d'environnement de la même manière que les clés JSON analysées. Les noms des variables n'incluent pas le nom du secret.

Si le secret ne contient pas de code JSON valide, l'action crée une variable d'environnement et lui donne le même nom que le secret.

- Si vous définissez `parse-json-secrets: false` et entrez un alias vide, suivi d'une virgule et de l'ID secret, l'action nomme les variables d'environnement comme si vous n'aviez pas spécifié d'alias.

L'exemple suivant montre un alias vide.

```
,secret2
```

Exemples

Example 1 Obtenir des secrets par leur nom et par leur ARN

L'exemple suivant crée des variables d'environnement pour les secrets identifiés par leur nom et par leur ARN.

```
- 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
```

Variables d'environnement créées :

```
EXAMPLESECRETNAME: secretValue1
TEST1: secretValue2
_0_TEST_SECRET: secretValue3
_PROD_EXAMPLE_SECRET: secretValue4
SECRET_ALIAS_1: secretValue5
SECRET_ALIAS_2: secretValue6
SECRET2: secretValue7
```

Example 2 Obtenir tous les secrets qui commencent par un préfixe

L'exemple suivant crée des variables d'environnement pour tous les secrets dont le nom commence par*beta*.

```
- name: Get Secret Names by Prefix
  uses: 2
  with:
    secret-ids: |
      beta*    # Retrieves all secrets that start with 'beta'
```

Variables d'environnement créées :

```
BETASECRETNAME: secretValue1
BETATEST: secretValue2
BETA_NEWSECRET: secretValue3
```

Example 3 Analyser le fichier JSON dans le secret

L'exemple suivant crée des variables d'environnement en analysant le fichier JSON dans le secret.

```
- 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
```

Le secret `test/secret` a la valeur de secret suivante.

```
{
  "api_user": "user",
  "api_key": "key",
  "config": {
    "active": "true"
  }
}
```

Le secret `secret2` a la valeur de secret suivante.

```
{
  "myusername": "alejandro_rosalez",
  "mypassword": "EXAMPLE_PASSWORD"
}
```

Variables d'environnement créées :

```
TEST_SECRET_API_USER: "user"
TEST_SECRET_API_KEY: "key"
TEST_SECRET_CONFIG_ACTIVE: "true"
MYUSERNAME: "alejandro_rosalez"
MYPASSWORD: "EXAMPLE_PASSWORD"
```

Example 4 Utilisez des lettres minuscules pour les noms des variables d'environnement

L'exemple suivant crée une variable d'environnement dont le nom est en minuscules.

```
- name: Get secrets
  uses: aws-actions/aws-secretsmanager-get-secrets@v2
  with:
    secret-ids: exampleSecretName
    name-transformation: lowercase
```

Variable d'environnement créée :

```
examplesecretname: secretValue
```

Utilisez AWS Secrets Manager des secrets dans AWS IoT Greengrass

AWS IoT Greengrass est un logiciel qui étend les fonctionnalités du cloud aux appareils locaux. Il permet aux appareils de collecter et d'analyser les données plus près de la source des informations, de réagir de manière autonome aux événements locaux et de communiquer en toute sécurité sur les réseaux locaux.

AWS IoT Greengrass vous permet de vous authentifier auprès de services et d'applications à partir d' AWS IoT Greengrass appareils sans avoir à coder en dur des mots de passe, des jetons ou d'autres secrets. Vous pouvez l'utiliser AWS Secrets Manager pour stocker et gérer vos secrets en

toute sécurité dans le cloud. AWS IoT Greengrass étend Secrets Manager aux appareils AWS IoT Greengrass principaux, afin que vos connecteurs et fonctions Lambda puissent utiliser des secrets locaux pour interagir avec les services et les applications.

Pour intégrer un secret dans un AWS IoT Greengrass groupe, vous devez créer une ressource de groupe qui fait référence au secret du Secrets Manager. Cette ressource secrète fait référence au secret de Cloud en utilisant l'ARN associé. Pour savoir comment créer, gérer et utiliser des ressources secrètes, consultez la section Travailler avec des ressources secrètes dans le guide du AWS IoT développeur.

Pour déployer des secrets sur le AWS IoT Greengrass cœur, voir Déployer des secrets sur le AWS IoT Greengrass cœur.

Utiliser des AWS Secrets Manager secrets dans Parameter Store

AWS Systems Manager Parameter Store fournit un stockage hiérarchique sécurisé pour la gestion des données de configuration et la gestion des secrets. Vous pouvez stocker des données telles que des mots de passe, des chaînes de base de données et des codes de licence en tant que valeurs de paramètres. Toutefois, le Parameter Store ne fournit pas de service de rotation automatique pour les secrets stockés. Au lieu de cela, le Parameter Store vous permet de stocker votre secret dans Secrets Manager et de référencer le secret sous forme de paramètre.

Lorsque vous configurez le Parameter Store avec Secrets Manager, le Parameter Store `secret-id` requiert une barre oblique (/) avant la chaîne de nom.

Pour plus d'informations, consultez la section Référencement de AWS Secrets Manager secrets à partir des paramètres du magasin de paramètres dans le guide de AWS Systems Manager l'utilisateur.

Faire pivoter AWS Secrets Manager les secrets

La Rotation est le processus de mise à jour périodique d'un secret. Lorsque vous effectuez une rotation de secret, vous mettez à jour les informations d'identification dans le secret et dans la base de données ou le service. Dans Secrets Manager, vous pouvez définir la rotation automatique de vos secrets. Il existe deux formes de rotation :

- Rotation gérée— Pour la plupart des secrets gérés, vous utilisez la rotation gérée, dans le cadre de laquelle le service configure et gère la rotation pour vous. La rotation gérée n'utilise pas de fonction Lambda.
- the section called "Rotation par fonction Lambda"— Pour les autres types de secrets, la rotation de Secrets Manager utilise une fonction Lambda pour mettre à jour le secret et la base de données ou le service.

Rotation gérée pour les AWS Secrets Manager secrets

Certains services proposent une rotation gérée, ce qui permet au service de configurer et de gérer la rotation pour vous. Avec la rotation gérée, vous n'utilisez aucune AWS Lambda fonction pour mettre à jour le secret et les informations d'identification dans la base de données.

Les services suivants proposent une rotation gérée :

- Amazon Aurora propose une rotation gérée pour les informations d'identification des utilisateurs principaux. Pour plus d'informations, consultez Gestion d'un cluster de base de données Amazon Aurora AWS Secrets Manager (français non garanti) dans le Manuel de l'utilisateur Amazon Aurora (français non garanti).
- Amazon ECS Service Connect propose une rotation gérée pour les certificats AWS Private Certificate Authority TLS. Pour plus d'informations, consultez TLS with Service Connect dans le manuel Amazon Elastic Container Service Developer Guide.
- Amazon RDS propose une rotation gérée pour les informations d'identification des utilisateurs principaux. Pour plus informations, consultez Gestion des mots de passe avec Amazon RDS et AWS Secrets Manager (français non garanti) dans le Guide de l'utilisateur Amazon RDS (français non garanti).
- Amazon Redshift propose une rotation gérée pour les mots de passe des administrateurs. Pour plus d'informations, consultez la section Gestion des mots de passe administrateur d'Amazon

Redshift à l'aide d' AWS Secrets Manager (français non garanti) du Guide de gestion Amazon Redshift.

> **Tip**
>
> Pour tout autre type de secrets, consultez the section called "Rotation par fonction Lambda".

La rotation des secrets gérés nécessite généralement moins d'une minute. Pendant la rotation, il est possible que les nouvelles connexions récupérant le secret obtiennent la version précédente des informations d'identification. Dans les applications, nous vous recommandons vivement d'avoir recours à un utilisateur de base de données doté des privilèges minimum requis pour votre application plutôt qu'un utilisateur principal. Pour les utilisateurs d'application, vous pouvez utiliser la stratégie de rotation des utilisateurs en alternance pour une disponibilité maximale.

Pour modifier le calendrier de rotation gérée

1. Ouvrez le secret géré dans la console Secrets Manager. Vous pouvez suivre un lien depuis le service de gestion ou rechercher le secret dans la console Secrets Manager.
2. Sous Rotation schedule (Planification de la rotation), saisissez votre planification dans le fuseau horaire UTC dans le Schedule expression builder (Générateur d'expressions de planification) ou sous la forme d'une Schedule expression (Expression de planification). Secrets Manager stocke votre planification sous la forme d'une expression `rate()` ou `cron()`. La fenêtre de rotation démarre automatiquement à minuit, sauf si vous spécifiez une Start time (Heure de début). Vous pouvez effectuer la rotation d'un secret toutes les quatre heures. Pour de plus amples informations, veuillez consulter Horaires de rotation.
3. (Facultatif) Pour Window duration (Durée de la fenêtre), choisissez la durée de la fenêtre pendant laquelle vous souhaitez que Secrets Manager effectue une rotation de votre secret, par exemple, **3h** pour une fenêtre de trois heures. La fenêtre ne doit pas s'étendre jusqu'à la fenêtre de rotation suivante. Si vous ne spécifiez pas la durée de la fenêtre, la fenêtre se ferme automatiquement au bout d'une heure si le programme de rotation est défini en heures. Pour un programme de rotation en jours, la fenêtre se ferme automatiquement à la fin de la journée.
4. Choisissez Enregistrer.

Pour modifier le programme de la rotation gérée (AWS CLI)

- Appelez `rotate-secret`. L'exemple suivant réalise une rotation du secret entre 16 h et 18 h UTC les 1ers et 15e jours du mois. Pour de plus amples informations, veuillez consulter Horaires de rotation.

```
aws secretsmanager rotate-secret \
    --secret-id MySecret \
    --rotation-rules \
        "{\"ScheduleExpression\": \"cron(0 16 1,15 * ? *)\", \"Duration\": \"2h\"}"
```

Rotation par fonction Lambda

Pour de nombreux types de secrets, Secrets Manager utilise une AWS Lambda fonction pour mettre à jour le secret et la base de données ou le service. Pour plus d'informations sur les coûts d'utilisation d'une fonction Lambda, consultez Tarification.

Pour certains Secrets gérés par d'autres services, vous utilisez la rotation gérée. Pour utiliser Rotation gérée, vous devez d'abord créer le secret via le service de gestion.

Pendant la rotation, Secrets Manager enregistre les événements qui indiquent l'état de rotation. Pour de plus amples informations, veuillez consulter the section called “Connectez-vous avec AWS CloudTrail ”.

Pour faire pivoter un secret, Secrets Manager appelle une fonction Lambda selon le calendrier de rotation que vous avez défini. Si vous mettez manuellement à jour votre valeur secrète alors que la rotation automatique est configurée, Secrets Manager considère qu'il s'agit d'une rotation valide lorsqu'il calcule la date de rotation suivante.

Au cours de la rotation, Secrets Manager appelle la même fonction plusieurs fois, avec des paramètres différents à chaque fois. Secrets Manager appelle la fonction avec la structure de demande JSON de paramètres suivante :

```
{
    "Step" : "request.type",
    "SecretId" : "string",
    "ClientRequestToken" : "string",
    "RotationToken" : "string"
}
```

Paramètres :

- Étape — L'étape de rotation : `create_secret`set_secret,`test_secret`, ou`finish_secret`. Pour de plus amples informations, veuillez consulter the section called "Quatre étapes d'une fonction de rotation".
- SecretId— L'ARN du secret à faire pivoter.
- ClientRequestToken— Un identifiant unique pour la nouvelle version du secret. Cette valeur permet de garantir l'idempuissance. Pour plus d'informations, voir PutSecretValue: ClientRequestToken dans la référence de AWS Secrets Manager l'API.
- RotationToken— Un identifiant unique qui indique la source de la demande. Nécessaire pour une rotation secrète utilisant un rôle assumé ou une rotation entre comptes, dans laquelle vous faites pivoter un secret dans un compte en utilisant une fonction de rotation Lambda dans un autre compte. Dans les deux cas, la fonction de rotation assume un rôle IAM pour appeler Secrets Manager, puis Secrets Manager utilise le jeton de rotation pour valider l'identité du rôle IAM.

Si une étape de la rotation échoue, Secrets Manager retente l'intégralité du processus de rotation plusieurs fois.

Rubriques

- Configurer la rotation automatique pour les secrets Amazon RDS, Amazon Aurora, Amazon Redshift ou Amazon DocumentDB
- Configurer la rotation automatique pour les secrets non liés à la base de données AWS Secrets Manager
- Configurer la rotation automatique à l'aide de l' AWS CLI
- Stratégies de rotation des fonctions Lambda
- Fonctions de rotation Lambda
- AWS Secrets Manager Modèles de fonctions de rotation
- Autorisations de rôle d'exécution de la fonction de rotation Lambda pour AWS Secrets Manager
- Accès au réseau pour la fonction AWS Lambda de rotation
- Résoudre AWS Secrets Manager la rotation d'

Configurer la rotation automatique pour les secrets Amazon RDS, Amazon Aurora, Amazon Redshift ou Amazon DocumentDB

Ce didacticiel explique comment configurer les secrets the section called "Rotation par fonction Lambda" de base de données. La Rotation est le processus de mise à jour périodique d'un secret. Lorsque vous effectuez une rotation de secret, vous mettez à jour les informations d'identification dans le secret et la base de données. Dans Secrets Manager, vous pouvez définir la rotation automatique de vos secrets de base de données.

Pour configurer la rotation à l'aide de la console, vous devez d'abord choisir une stratégie de rotation. Vous configurez ensuite le secret pour la rotation, ce qui crée une fonction de rotation Lambda si vous n'en avez pas déjà une. La console définit également les autorisations pour le rôle d'exécution de la fonction Lambda. La dernière étape consiste à s'assurer que la fonction de rotation Lambda peut accéder à la fois à Secrets Manager et à votre base de données via le réseau.

> **Warning**
>
> Pour activer la rotation automatique, vous devez être autorisé à créer un rôle d'exécution IAM pour la fonction de rotation Lambda et à y associer une politique d'autorisation. Vous avez besoin des deux autorisations `iam:CreateRole` et `iam:AttachRolePolicy`. L'octroi de ces autorisations permet à une identité de s'octroyer toutes les autorisations.

Étapes :

- Étape 1 : choisir une stratégie de rotation et (éventuellement) créer un secret de super-utilisateur
- Étape 2 : configurer la rotation et créer une fonction de rotation
- Étape 3 : (facultative) définir des conditions d'autorisation supplémentaires pour la fonction de rotation
- Étape 4 : configurer l'accès au réseau pour la fonction de rotation
- Étapes suivantes

Étape 1 : choisir une stratégie de rotation et (éventuellement) créer un secret de super-utilisateur

Pour plus d'informations sur les stratégies proposées par Secrets Manager, consultezthe section called "Stratégies de rotation des fonctions Lambda".

Si vous choisissez la stratégie des utilisateurs en alternance, vous devez Création de secrets et y stocker les informations d'identification du super-utilisateur de la base de données. Vous avez besoin d'un code secret avec les informations d'identification du super-utilisateur, car la rotation clone le premier utilisateur, et la plupart des utilisateurs ne disposent pas de cette autorisation. Notez qu'Amazon RDS Proxy ne prend pas en charge la stratégie des utilisateurs alternatifs.

Étape 2 : configurer la rotation et créer une fonction de rotation

Pour activer la rotation d'un secret Amazon RDS, Amazon DocumentDB ou Amazon Redshift

1. Ouvrez la console Secrets Manager à l'adresse https://console.aws.amazon.com/secretsmanager/.
2. Sur la page Secrets, choisissez votre secret.
3. Sur la page Secret details (Détails du secret), dans la section Rotation configuration (Configuration de la rotation), choisissez Edit rotation (Modifier la rotation).
4. Dans la boîte de dialogue Edit rotation configuration (Modifier la configuration de la rotation), suivez la procédure suivante :

   a. Activez Automatic rotation (Rotation automatique).

   b. Sous Rotation schedule (Planification de la rotation), saisissez votre planification dans le fuseau horaire UTC dans le Schedule expression builder (Générateur d'expressions de planification) ou sous la forme d'une Schedule expression (Expression de planification). Secrets Manager stocke votre planification sous la forme d'une expression `rate()` ou `cron()`. La fenêtre de rotation démarre automatiquement à minuit, sauf si vous spécifiez une Start time (Heure de début). Vous pouvez effectuer la rotation d'un secret toutes les quatre heures. Pour de plus amples informations, veuillez consulter Horaires de rotation.

   c. (Facultatif) Pour Window duration (Durée de la fenêtre), choisissez la durée de la fenêtre pendant laquelle vous souhaitez que Secrets Manager effectue une rotation de votre secret, par exemple, **3h** pour une fenêtre de trois heures. La fenêtre ne doit pas s'étendre jusqu'à la fenêtre de rotation suivante. Si vous ne spécifiez pas la durée de la fenêtre, la fenêtre se ferme automatiquement au bout d'une heure si le programme de rotation est défini en heures. Pour un programme de rotation en jours, la fenêtre se ferme automatiquement à la fin de la journée.

   d. (Facultatif) Sélectionnez Rotate immediately when the secret is stored (Effectuer immédiatement une rotation lorsque le secret est stocké) pour effectuer une rotation de votre secret lorsque vous enregistrez vos modifications. Si vous décochez la case, la première rotation commencera selon la planification que vous avez définie.

Si la rotation échoue, par exemple parce que les étapes 3 et 4 ne sont pas encore terminées, Secrets Manager tente à nouveau le processus de rotation plusieurs fois.

e. Sous Rotation function (Fonction de rotation), effectuez l'une des actions suivantes :

- Sélectionnez Create a new Lambda function (Créer une fonction Lambda) et saisissez le nom de votre fonction. Secrets Manager ajoute `SecretsManager` au début du nom de votre fonction. Secrets Manager crée la fonction en se basant sur le modèle approprié et définit les autorisations nécessaires pour le rôle d'exécution Lambda.
- Choisissez Use an existing Lambda function (Utiliser une fonction Lambda existante) pour réutiliser une fonction de rotation que vous avez utilisée pour un autre secret. Les fonctions de rotation répertoriées sous Recommended VPC configurations (Configurations de VPC recommandées) ont le même VPC et le même groupe de sécurité que la base de données, ce qui permet à la fonction d'accéder à la base de données.

f. Pour la Stratégie de rotation, choisissez la stratégie Utilisateur unique ou Utilisateurs en alternance. Pour de plus amples informations, veuillez consulter the section called "Étape 1 : choisir une stratégie de rotation et (éventuellement) créer un secret de super-utilisateur".

5. Choisissez Enregistrer.

Étape 3 : (facultative) définir des conditions d'autorisation supplémentaires pour la fonction de rotation

Dans la stratégie de ressources de votre fonction de rotation, nous vous recommandons d'inclure la clé de contexte `aws:SourceAccount` afin d'éviter que Lambda ne soit utilisé comme un adjoint confus. Pour certains AWS services, afin d'éviter toute confusion dans le scénario adjoint, il est AWS recommandé d'utiliser à la fois les clés de condition `aws:SourceArn`et les clés de condition `aws:SourceAccount`globale. Cependant, si vous incluez la condition `aws:SourceArn` dans votre stratégie de fonction de rotation Lambda, la fonction de rotation ne peut être utilisée que pour effectuer la rotation du secret spécifié par cet ARN. Nous vous recommandons d'inclure uniquement la clé de contexte `aws:SourceAccount` afin de pouvoir utiliser la fonction de rotation pour plusieurs secrets.

Pour mettre à jour la stratégie de ressources de votre fonction de rotation

1. Dans la console Secrets Manager, choisissez votre secret, puis sur la page des détails, sous Rotation configuration (Configuration de la rotation), choisissez la fonction de rotation Lambda. La console Lambda s'ouvre.
2. Suivez les instructions de la rubrique Utilisation de stratégies basées sur les ressources pour Lambda pour ajouter une condition `aws:sourceAccount`.

```
"Condition": {
    "StringEquals": {
        "AWS:SourceAccount": "123456789012"
    }
},
```

Si le secret est chiffré avec une clé KMS autre que la Clé gérée par AWS `aws/secretsmanager`, Secrets Manager accorde au rôle d'exécution Lambda l'autorisation d'utiliser la clé. Vous pouvez utiliser le contexte de chiffrement SecretARN pour limiter l'utilisation de la fonction de déchiffrement, de sorte que le rôle de la fonction de rotation n'ait accès que pour déchiffrer le secret qu'il est chargé de faire pivoter.

Pour mettre à jour le rôle d'exécution de votre fonction de rotation

1. Dans la fonction de rotation Lambda, choisissez Configuration, puis sous Rôle d'exécution, choisissez le Nom du rôle.
2. Suivez les instructions de la rubrique Modification de la stratégie d'autorisations d'un rôle pour ajouter une condition `kms:EncryptionContext:SecretARN`.

```
"Condition": {
    "StringEquals": {
        "kms:EncryptionContext:SecretARN": "SecretARN"
    }
},
```

Étape 4 : configurer l'accès au réseau pour la fonction de rotation

Pour de plus amples informations, veuillez consulter the section called “Accès au réseau pour la fonction AWS Lambda de rotation”.

Étapes suivantes

Voir the section called "Résolution des problèmes de rotation".

Configurer la rotation automatique pour les secrets non liés à la base de données AWS Secrets Manager

Ce didacticiel explique comment configurer les secrets autres que the section called "Rotation par fonction Lambda" ceux de base de données. La Rotation est le processus de mise à jour périodique d'un secret. Lorsque vous effectuez une rotation de secret, vous mettez à jour les informations d'identification dans le secret et dans la base de données ou le service que le secret concerne.

Pour les secrets de base de données, consultez Rotation automatique pour les secrets de base de données (console).

> **Warning**
>
> Pour activer la rotation automatique, vous devez être autorisé à créer un rôle d'exécution IAM pour la fonction de rotation Lambda et à y associer une politique d'autorisation. Vous avez besoin des deux autorisations `iam:CreateRole` et `iam:AttachRolePolicy`. L'octroi de ces autorisations permet à une identité de s'octroyer toutes les autorisations.

Étapes :

- Étape 1 : Création d'une fonction de rotation générique
- Étape 2 : écrire le code de la fonction de rotation
- Étape 3 : Configuration du secret pour la rotation
- Étape 4 : Autoriser la fonction de rotation à accéder à Secrets Manager et à votre base de données ou service
- Étape 5 : Autoriser Secrets Manager à invoquer la fonction de rotation
- Étape 6 : Configuration de l'accès au réseau pour la fonction de rotation
- Étapes suivantes

Étape 1 : Création d'une fonction de rotation générique

Pour commencer, créez une fonction de rotation Lambda. Il ne contiendra pas le code nécessaire pour faire pivoter votre secret, vous l'écrirez donc ultérieurement. Pour plus d'informations sur

le fonctionnement d'une fonction de rotation, consultezthe section called "Fonctions de rotation Lambda".

Dans les régions prises en charge, vous pouvez l'utiliser AWS Serverless Application Repository pour créer la fonction à partir d'un modèle. Pour obtenir une liste des régions prises en charge, consultez AWS Serverless Application Repository FAQs. Dans d'autres régions, vous créez la fonction à partir de zéro et vous copiez le code du modèle dans la fonction.

Pour créer une fonction de rotation générique

1. Pour déterminer s'il AWS Serverless Application Repository est pris en charge dans votre région, consultez la section AWS Serverless Application Repository Points de terminaison et quotas dans le Guide de référence AWS général.
2. Effectuez l'une des actions suivantes :
   - S'il AWS Serverless Application Repository est pris en charge dans votre région :
     a. Dans la console Lambda, choisissez Applications, puis Create application.
     b. Sur la page Créer une application, choisissez l'onglet Application sans serveur.
     c. Dans le champ de recherche sous Applications publiques, entrez**`SecretsManagerRotationTemplate`**.
     d. Sélectionnez Afficher les applications qui créent des rôles IAM ou des politiques de ressources personnalisés.
     e. Choisissez le SecretsManagerRotationTemplatecarreau.
     f. Sur la page Révision, configuration et déploiement, dans la vignette Paramètres de l'application, renseignez les champs obligatoires.
        - Pour point de terminaison, entrez le point de terminaison de votre région, y compris**`https://`**. Pour obtenir la liste des points de terminaison , consultez the section called "Points de terminaison Secrets Manager".
        - Pour placer la fonction Lambda dans un VPC, incluez les identifiants et. vpcSecurityGroup vpcSubnetIds
     g. Choisissez Déployer.
   - S'il AWS Serverless Application Repository n'est pas pris en charge dans votre région :
     a. Dans la console Lambda, choisissez Functions, puis Create function.
     b. Sur la page Create function (Créer une fonction), procédez de la façon suivante :

i. Choisissez Créer à partir de zéro.

ii. Dans Function name (Nom de la fonction), saisissez un nom pour votre fonction de rotation.

iii. Pour Runtime (Environnement d'exécution), choisissez Python 3.9.

iv. Choisissez Créer une fonction.

Étape 2 : écrire le code de la fonction de rotation

Au cours de cette étape, vous écrivez le code qui met à jour le secret et le service ou la base de données auxquels le secret est destiné. Pour plus d'informations sur le fonctionnement d'une fonction de rotation, y compris des conseils sur l'écriture de votre propre fonction de rotation, consultezthe section called "Fonctions de rotation Lambda". Vous pouvez également utiliser le Modèles de fonctions de rotation comme référence.

Étape 3 : Configuration du secret pour la rotation

Au cours de cette étape, vous définissez un calendrier de rotation pour votre secret et connectez la fonction de rotation au secret.

Pour configurer la rotation et créer une fonction de rotation vide

1. Ouvrez la console Secrets Manager à l'adresse https://console.aws.amazon.com/secretsmanager/.
2. Sur la page Secrets, choisissez votre secret.
3. Sur la page Secret details (Détails du secret), dans la section Rotation configuration (Configuration de la rotation), choisissez Edit rotation (Modifier la rotation). Dans la boîte de dialogue Edit rotation configuration (Modifier la configuration de la rotation), suivez la procédure suivante :

   a. Activez Automatic rotation (Rotation automatique).

   b. Sous Rotation schedule (Planification de la rotation), saisissez votre planification dans le fuseau horaire UTC dans le Schedule expression builder (Générateur d'expressions de planification) ou sous la forme d'une Schedule expression (Expression de planification). Secrets Manager stocke votre planification sous la forme d'une expression `rate()` ou `cron()`. La fenêtre de rotation démarre automatiquement à minuit, sauf si vous spécifiez

une Start time (Heure de début). Vous pouvez effectuer la rotation d'un secret toutes les quatre heures. Pour de plus amples informations, veuillez consulter Horaires de rotation.

c. (Facultatif) Pour Window duration (Durée de la fenêtre), choisissez la durée de la fenêtre pendant laquelle vous souhaitez que Secrets Manager effectue une rotation de votre secret, par exemple, **3h** pour une fenêtre de trois heures. La fenêtre ne doit pas s'étendre jusqu'à la fenêtre de rotation suivante. Si vous ne spécifiez pas la durée de la fenêtre, la fenêtre se ferme automatiquement au bout d'une heure si le programme de rotation est défini en heures. Pour un programme de rotation en jours, la fenêtre se ferme automatiquement à la fin de la journée.

d. (Facultatif) Sélectionnez Rotate immediately when the secret is stored (Effectuer immédiatement une rotation lorsque le secret est stocké) pour effectuer une rotation de votre secret lorsque vous enregistrez vos modifications. Si vous décochez la case, la première rotation commencera selon la planification que vous avez définie.

e. Sous Fonction de rotation, choisissez la fonction Lambda que vous avez créée à l'étape 1.

f. Choisissez Enregistrer.

Étape 4 : Autoriser la fonction de rotation à accéder à Secrets Manager et à votre base de données ou service

La fonction de rotation Lambda a besoin d'une autorisation pour accéder au secret dans Secrets Manager, ainsi que pour accéder à votre base de données ou à votre service. Au cours de cette étape, vous accordez ces autorisations au rôle d'exécution Lambda. Si le secret est chiffré avec une clé KMS autre que la Clé gérée par AWS `aws/secretsmanager`, vous devez accorder au rôle d'exécution Lambda l'autorisation d'utiliser la clé. Vous pouvez utiliser le contexte de chiffrement SecretARN pour limiter l'utilisation de la fonction de déchiffrement, de sorte que le rôle de la fonction de rotation n'ait accès que pour déchiffrer le secret qu'il est chargé de faire pivoter. Pour obtenir des exemples de stratégie, consultez Autorisations de rotation.

Pour obtenir des instructions, consultez Rôle d'exécution Lambda dans le Guide du développeur AWS Lambda .

Étape 5 : Autoriser Secrets Manager à invoquer la fonction de rotation

Pour permettre à Secrets Manager d'invoquer la fonction de rotation selon le calendrier de rotation que vous avez défini, vous devez `lambda:InvokeFunction` autoriser le responsable du service Secrets Manager dans la politique de ressources de la fonction Lambda.

Dans la stratégie de ressources de votre fonction de rotation, nous vous recommandons d'inclure la clé de contexte `aws:SourceAccount` afin d'éviter que Lambda ne soit utilisé comme un adjoint confus. Pour certains AWS services, afin d'éviter toute confusion dans le scénario adjoint, il est AWS recommandé d'utiliser à la fois les clés de condition `aws:SourceArn`et les clés de condition `aws:SourceAccount`globale. Cependant, si vous incluez la condition `aws:SourceArn` dans votre stratégie de fonction de rotation Lambda, la fonction de rotation ne peut être utilisée que pour effectuer la rotation du secret spécifié par cet ARN. Nous vous recommandons d'inclure uniquement la clé de contexte `aws:SourceAccount` afin de pouvoir utiliser la fonction de rotation pour plusieurs secrets.

Pour attacher une stratégie de ressources à une fonction Lambda, consultez Utilisation de stratégies basées sur les ressources pour Lambda.

La politique suivante permet à Secrets Manager d'appeler une fonction Lambda.

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"
    }
    ]
}
```

Étape 6 : Configuration de l'accès au réseau pour la fonction de rotation

Au cours de cette étape, vous autorisez la fonction de rotation à se connecter à la fois à Secrets Manager et au service ou à la base de données auxquels le secret est destiné. La fonction de rotation doit avoir accès aux deux pour pouvoir faire pivoter le secret. Consultez the section called "Accès au réseau pour la fonction AWS Lambda de rotation".

Étapes suivantes

Lorsque vous avez configuré la rotation à l'étape 3, vous définissez un calendrier pour la rotation du secret. Si la rotation échoue lorsqu'elle est planifiée, Secrets Manager essaiera de la faire plusieurs fois. Vous pouvez également démarrer une rotation immédiatement en suivant les instructions figurant dansMettre immédiatement un secret en rotation.

Si la rotation échoue, voirRésolution des problèmes de rotation.

Configurer la rotation automatique à l'aide de l' AWS CLI

Ce didacticiel explique comment procéder à la configuration the section called "Rotation par fonction Lambda" à l'aide du AWS CLI. Lorsque vous effectuez une rotation de secret, vous mettez à jour les informations d'identification dans le secret et dans la base de données ou le service que le secret concerne.

Vous pouvez également configurer la rotation à l'aide de la console. Pour les secrets de base de données, consultez Rotation automatique pour les secrets de base de données (console). Pour tout autre type de secrets, consultez Rotation automatique pour les secrets non liés à la base de données (console).

Pour configurer la rotation à l'aide de l' AWS CLI, si vous effectuez la rotation d'un secret de base de données, vous devez d'abord choisir une stratégie de rotation. Si vous choisissez la stratégie des utilisateurs en alternance, vous devez stocker un secret distinct avec les informations d'identification d'un super-utilisateur de la base de données. Ensuite, vous écrivez le code de la fonction de rotation. Secrets Manager fournit des modèles de base pour écrire votre fonction. Vous créez ensuite une fonction Lambda avec votre code et définissez des autorisations pour la fonction Lambda et le rôle d'exécution Lambda. L'étape suivante consiste à s'assurer que la fonction Lambda peut accéder à la fois à Secrets Manager et à votre base de données ou à votre service via le réseau. Enfin, vous configurez le secret de la rotation.

Étapes :

- Prérequis pour les secrets de base de données : Choisissez une stratégie de rotation
- Étape 1 : écrire le code de la fonction de rotation
- Étape 2 : création de la fonction Lambda
- Étape 3 : configurer l'accès au réseau
- Étape 4 : configurer le secret pour la rotation
- Étapes suivantes

Prérequis pour les secrets de base de données : Choisissez une stratégie de rotation

Pour plus d'informations sur les stratégies proposées par Secrets Manager, consultezthe section called "Stratégies de rotation des fonctions Lambda".

Option 1 : stratégie utilisateur unique

Si vous choisissez la stratégie utilisateur unique, vous pouvez passer à l'étape 1.

Option 2 : stratégie d'utilisation en alternance

Si vous optez pour la stratégie d'alternance des utilisateurs, vous devez :

- Créez un secret et stockez-y les informations d'identification du superutilisateur de la base de données. Vous avez besoin d'un code secret avec des informations d'identification de super-utilisateur, car la rotation des utilisateurs en alternance clone le premier utilisateur, et la plupart des utilisateurs ne disposent pas de cette autorisation.
- Ajoutez l'ARN du secret du superutilisateur au secret d'origine. Pour de plus amples informations, veuillez consulter the section called "Structure JSON d'un secret".

Notez qu'Amazon RDS Proxy ne prend pas en charge la stratégie des utilisateurs alternatifs.

Étape 1 : écrire le code de la fonction de rotation

Pour effectuer la rotation d'un secret, vous avez besoin d'une fonction de rotation. Une fonction de rotation est une fonction Lambda que Secrets Manager appelle pour effectuer la rotation de votre secret. Pour de plus amples informations, veuillez consulter the section called "Rotation par fonction Lambda". Dans cette étape, vous écrivez le code qui met à jour le secret et le service ou la base de données que le secret concerne.

Secrets Manager fournit des modèles pour les secrets de base de données Amazon RDS, Amazon Aurora, Amazon Redshift et Amazon DocumentDB dans. Modèles de fonctions de rotation

Pour écrire le code de la fonction de rotation

1. Effectuez l'une des actions suivantes :
   - Consultez la liste des modèles de fonctions de rotation. S'il y en a un qui correspond à votre stratégie de service et de rotation, copiez-le.
   - Pour d'autres types de secrets, vous écrivez votre propre fonction de rotation. Pour obtenir des instructions, veuillez consulter the section called "Fonctions de rotation Lambda".
2. Enregistrez le fichier dans un fichier ZIP *my-function.zip* avec toutes les dépendances requises.

Étape 2 : création de la fonction Lambda

Dans cette étape, vous créez la fonction Lambda à l'aide du fichier ZIP que vous avez créé à l'étape 1. Vous définissez également le rôle d'exécution Lambda, qui est le rôle assumé par Lambda lorsque la fonction est invoquée.

Pour créer une fonction de rotation Lambda et un rôle d'exécution

1. Créez une stratégie d'approbation pour le rôle d'exécution Lambda et enregistrez-la dans un fichier JSON. Pour obtenir des exemples et de plus amples informations, veuillez consulterthe section called "Autorisations de rotation". La stratégie doit :
   - autoriser le rôle à appeler les opérations Secrets Manager sur le secret ;
   - autoriser le rôle à appeler le service auquel le secret est destiné, par exemple pour créer un nouveau mot de passe.
2. Créez le rôle d'exécution Lambda et appliquez la politique d'approbation que vous avez créée à l'étape précédente en appelant. `iam create-role`

```
aws iam create-role \
    --role-name rotation-lambda-role \
    --assume-role-policy-document file://trust-policy.json
```

3. Créez la fonction Lambda à partir du fichier ZIP en appelant `lambda create-function`.

```
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. Définissez une stratégie de ressources sur la fonction Lambda afin de permettre à Secrets Manager de l'invoquer en appelant `lambda add-permission`.

```
aws lambda add-permission \
  --function-name my-rotation-function \
  --action lambda:InvokeFunction \
  --statement-id SecretsManager \
  --principal secretsmanager.amazonaws.com \
  --source-account 123456789012
```

Étape 3 : configurer l'accès au réseau

Pour de plus amples informations, veuillez consulter the section called “Accès au réseau pour la fonction AWS Lambda de rotation”.

Étape 4 : configurer le secret pour la rotation

Pour activer la rotation automatique de votre secret, appelez `rotate-secret`. Vous pouvez définir une planification de rotation avec une expression de planification `cron()` ou `rate()`, et définir la durée d'une fenêtre de rotation. Pour de plus amples informations, veuillez consulter the section called “Horaires de rotation”.

```
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\"}"
```

Étapes suivantes

Consultez the section called “Résolution des problèmes de rotation”.

Stratégies de rotation des fonctions Lambda

En the section called "Rotation par fonction Lambda" ce qui concerne les secrets de base de données, Secrets Manager propose deux stratégies de rotation.

Stratégie de rotation : utilisateur unique

Cette stratégie met à jour les informations d'identification d'un utilisateur dans un seul secret. Pour les instances Amazon RDS Db2, étant donné que les utilisateurs ne peuvent pas modifier leurs propres mots de passe, vous devez fournir des informations d'identification d'administrateur dans un secret distinct. Il s'agit de la stratégie de rotation la plus simple, qui convient à la plupart des cas d'utilisation. Nous vous recommandons en particulier d'utiliser cette stratégie pour les informations d'identification des utilisateurs ponctuels (ad hoc) ou interactifs.

Lorsque le secret effectue une rotation, les connexions de base de données ouvertes ne sont pas supprimées. Pendant la rotation, peu de temps s'écoule entre le moment où le mot de passe dans la base de données change et le moment où le secret est mis à jour. Pendant cette durée, le risque que la base de données refuse les appels utilisant les informations d'identification mises en rotation est faible. Pour pallier ce risque, vous pouvez utiliser une stratégie de nouvelle tentative appropriée. Après la rotation, les nouvelles connexions utilisent les nouvelles informations d'identification.

Stratégie de rotation : utilisateurs en alternance

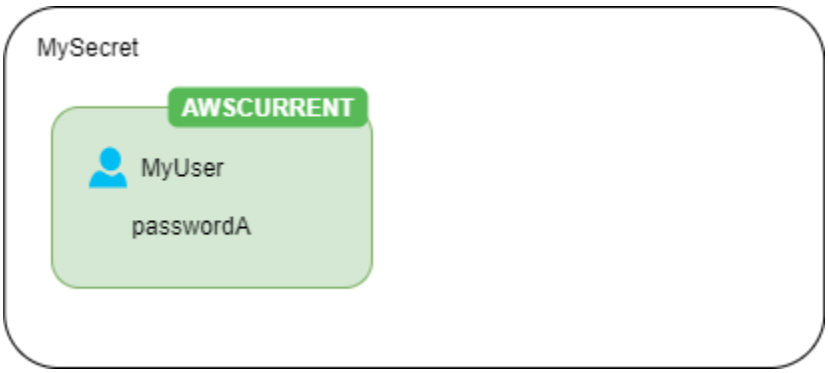
Cette stratégie met à jour les informations d'identification de deux utilisateurs dans un seul secret. Vous créez le premier utilisateur et lors de la première rotation, la fonction de rotation le clone pour créer le second utilisateur. Chaque fois que le secret effectue une rotation, la fonction de rotation alterne l'utilisateur dont elle met à jour le mot de passe. Comme la plupart des utilisateurs n'ont pas l'autorisation de se cloner eux-mêmes, vous devez fournir les informations d'identification pour un `superuser` dans un autre secret. Nous recommandons d'utiliser la stratégie de rotation d'utilisateur unique lorsque les utilisateurs clonés de votre base de données ne disposent pas des mêmes autorisations que l'utilisateur d'origine, ainsi que pour les informations d'identification des utilisateurs ponctuels (ad hoc) ou interactifs.

Cette stratégie est appropriée pour les bases de données avec modèles d'autorisations où un rôle possède les tables de base de données et un second rôle est autorisé à accéder aux tables de la base de données. Elle convient également aux applications qui requièrent une haute disponibilité. Si une application récupère le secret pendant la rotation, elle obtient quand même un ensemble d'informations d'identification valide. Après la rotation, les informations d'identification `user` et

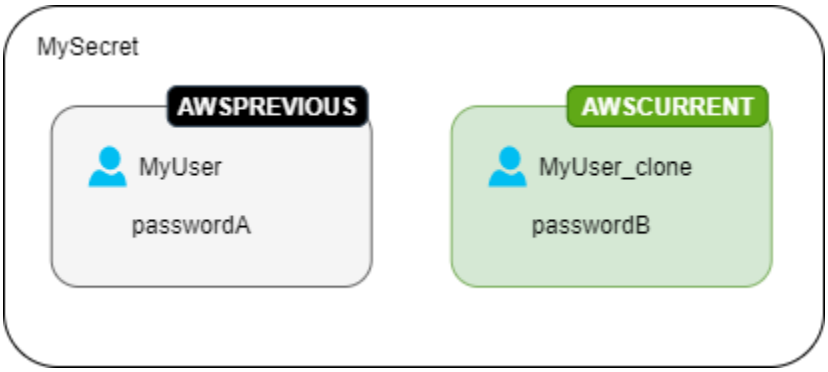
`user_clone` sont valides. Il y a même moins de chances que les applications obtiennent un refus pendant ce type de rotation qu'avec la rotation utilisateur unique. Si la base de données est hébergée sur une batterie de serveurs où la propagation de la modification du mot de passe à tous les serveurs prend du temps, la base de données risque de refuser les appels utilisant les nouvelles informations d'identification. Pour pallier ce risque, vous pouvez utiliser une stratégie de nouvelle tentative appropriée.

Secrets Manager crée l'utilisateur cloné avec les mêmes autorisations que l'utilisateur d'origine. Si vous modifiez les autorisations de l'utilisateur d'origine après la création du clone, vous devez également modifier les autorisations de l'utilisateur cloné.

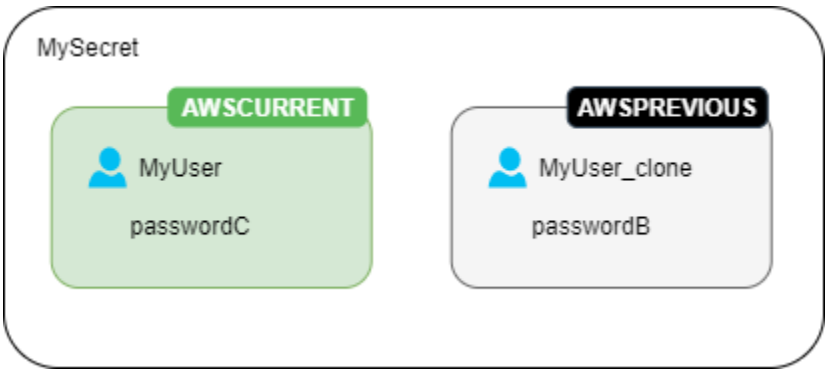
Par exemple, si vous créez un secret avec les informations d'identification d'un utilisateur de base de données, le secret contient une version avec ces informations d'identification.



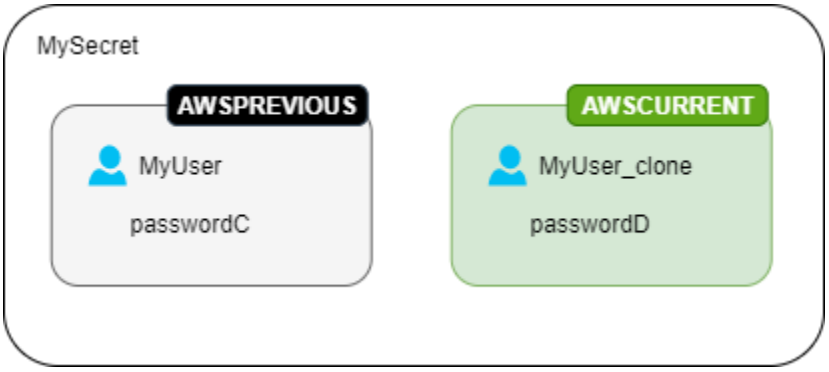
Première rotation — La fonction de rotation crée un clone de votre utilisateur avec un mot de passe généré, et ces informations d'identification deviennent la version secrète actuelle.



Deuxième rotation — La fonction de rotation met à jour le mot de passe de l'utilisateur d'origine.



Troisième rotation — La fonction de rotation met à jour le mot de passe de l'utilisateur cloné.



Fonctions de rotation Lambda

Dansthe section called "Rotation par fonction Lambda", une AWS Lambda fonction fait pivoter le secret. AWS Secrets Manager utilise des étiquettes de mise en scène pour identifier les versions secrètes lors de la rotation.

S'il AWS Secrets Manager ne fournit pas de modèle de fonction de rotation pour votre type secret, vous pouvez créer une fonction de rotation personnalisée. Suivez ces instructions lorsque vous écrivez votre fonction de rotation :

Bonnes pratiques pour les fonctions de rotation personnalisées

- Utilisez le modèle de rotation générique comme point de départ.
- Soyez prudent avec les instructions de débogage ou de journalisation. Ils peuvent écrire des informations sur Amazon CloudWatch Logs. Assurez-vous que les journaux ne contiennent pas d'informations sensibles.

  Pour des exemples d'instructions de journal, consultez le code the section called "Modèles de fonctions de rotation" source.

- Pour des raisons de sécurité, AWS Secrets Manager seule une fonction de rotation Lambda permet à une fonction de rotation Lambda de faire pivoter directement le secret. La fonction de rotation ne peut pas appeler une autre fonction Lambda pour faire pivoter le secret.
- Pour obtenir des conseils de débogage, consultez la section Tester et déboguer des applications sans serveur.
- Si vous utilisez des binaires et des bibliothèques externes, par exemple pour vous connecter à une ressource, il vous incombe de les corriger et de les mettre à jour.
- Package votre fonction de rotation et toutes les dépendances dans un fichier ZIP, tel que*my-function.zip*.

> **Warning**
>
> La définition du paramètre de simultanéité provisionné sur une valeur inférieure à 10 peut entraîner un ralentissement dû à un nombre insuffisant de threads d'exécution pour la fonction Lambda. Pour plus d'informations, consultez la section Comprendre la simultanéité réservée et la simultanéité provisionnée dans le Guide du AWS Lambda AWS Lambda développeur.

Quatre étapes d'une fonction de rotation

Rubriques

- createSecret: crée une nouvelle version du secret
- setSecret: modifiez les informations d'identification dans la base de données ou le service
- testSecret: Testez la nouvelle version secrète
- finishSecret: Terminez la rotation

`createSecret`: crée une nouvelle version du secret

La méthode vérifie d'`createSecret`abord si un secret existe en appelant `get_secret_value`avec le mot transmis`ClientRequestToken`. S'il n'y a pas de secret, il en crée un nouveau avec `create_secret`le jeton comme`VersionId`. Il génère ensuite une nouvelle valeur secrète avec `get_random_password`. Ensuite, il appelle `put_secret_value`pour le stocker avec l'étiquette de mise en scène`AWSPENDING`. Le stockage de la nouvelle valeur de secret dans `AWSPENDING` permet de garantir l'idempotence. Si la rotation échoue pour une raison quelconque, vous pouvez

vous référer à cette valeur de secret lors des appels suivants. Veuillez consulter Comment puis-je rendre ma fonction Lambda idempotente ?.

Conseils pour écrire votre propre fonction de rotation

- Assurez-vous que la nouvelle valeur secrète inclut uniquement des caractères valides pour la base de données ou le service. Excluez des caractères à l'aide du paramètre `ExcludeCharacters`.
- Lorsque vous testez votre fonction, utilisez le AWS CLI pour voir les étapes de version : call `describe-secret`and look at`VersionIdsToStages`.
- Pour Amazon RDS MySQL, lors de la rotation alternée des utilisateurs, Secrets Manager crée un utilisateur cloné dont le nom ne dépasse pas 16 caractères. Vous pouvez modifier la fonction de rotation pour autoriser des noms d'utilisateur plus longs. MySQL version 5.7 et versions ultérieures prend en charge les noms d'utilisateur jusqu'à 32 caractères, mais Secrets Manager ajoute « _clone » (six caractères) à la fin du nom d'utilisateur. Vous devez donc limiter le nom d'utilisateur à 26 caractères maximum.

setSecret: modifiez les informations d'identification dans la base de données ou le service

La méthode `setSecret` modifie les informations d'identification dans la base de données ou le service pour qu'elles correspondent à la nouvelle valeur secrète de la `AWSPENDING` version du secret.

Conseils pour écrire votre propre fonction de rotation

- Si vous transmettez des instructions à un service qui interprète des instructions, comme une base de données, utilisez le paramétrage des requêtes. Pour plus d'informations, consultez le aide-mémoire sur le paramétrage des requêtes sur le site Web de l'OWASP.
- La fonction de rotation est un adjoint privilégié autorisé à accéder aux informations d'identification des clients et à les modifier à la fois dans le secret Secrets Manager et dans la ressource cible. Pour éviter une éventuelle attaque d'adjoint confus, vous devez vous assurer que les pirates ne peuvent pas utiliser la fonction afin d'accéder à d'autres ressources. Avant de mettre à jour les informations d'identification :
  - Vérifiez que les informations d'identification figurant dans la version `AWSCURRENT` du secret sont valides. Si les informations d'identification `AWSCURRENT` ne sont pas valides, abandonnez la tentative de rotation.

  - Vérifiez que les valeurs de secret `AWSPENDING` et `AWSCURRENT` concernent la même ressource. Pour un nom d'utilisateur et un mot de passe, vérifiez que les noms d'utilisateur `AWSCURRENT` et `AWSPENDING` sont identiques.
  - Vérifiez également que la ressource du service de destination est identique. Pour une base de données, vérifiez que les noms d'hôte `AWSCURRENT` et `AWSPENDING` sont identiques.
- Dans de rares cas, vous souhaiterez peut-être personnaliser une fonction de rotation existante pour une base de données. Par exemple, avec une rotation des utilisateurs en alternance, Secrets Manager crée l'utilisateur cloné en copiant les paramètres de configuration d'exécution du premier utilisateur. Si vous souhaitez inclure d'autres attributs ou modifier ceux qui sont accordés à l'utilisateur cloné, vous devez mettre à jour le code de la fonction `set_secret`.

testSecret: Testez la nouvelle version secrète

Ensuite, la fonction de rotation Lambda teste la version `AWSPENDING` du secret en l'utilisant pour accéder à la base de données ou au service. Fonctions de rotation basées sur le test Modèles de fonctions de rotation du nouveau secret en utilisant l'accès en lecture.

finishSecret: Terminez la rotation

Enfin, la fonction de rotation Lambda déplace l'étiquette `AWSCURRENT` de la version secrète précédente vers cette version, qui supprime également l'étiquette `AWSPENDING` dans le même appel d'API. Secrets Manager ajoute l'étiquette intermédiaire `AWSPREVIOUS` à la version précédente, de sorte que vous conservez la dernière bonne version connue du secret.

La méthode finish_secret permet `update_secret_version_stage`de déplacer l'étiquette intermédiaire `AWSCURRENT` de la version secrète précédente vers la nouvelle version secrète. Secrets Manager ajoute automatiquement l'étiquette intermédiaire `AWSPREVIOUS` à la version précédente, de sorte que vous conservez la dernière bonne version connue du secret.

Conseils pour écrire votre propre fonction de rotation

- Ne le supprimez pas `AWSPENDING` avant ce point, et ne le supprimez pas en utilisant un appel d'API distinct, car cela peut indiquer à Secrets Manager que la rotation ne s'est pas terminée correctement. Secrets Manager ajoute l'étiquette intermédiaire `AWSPREVIOUS` à la version précédente, de sorte que vous conservez la dernière bonne version connue du secret.

Lorsque la rotation est réussie, l'étiquette `AWSPENDING` de transit peut être attachée à la même version que la version `AWSCURRENT`, ou elle peut ne pas être attachée à aucune version. Si l'étiquette

de transit AWSPENDING est présente, mais n'est pas attachée à la même version AWSCURRENT, toute invocation ultérieure à la rotation suppose qu'une demande de rotation précédente est toujours en cours et renvoie une erreur. Lorsque la rotation échoue, l'étiquette AWSPENDING intermédiaire peut être attachée à une version secrète vide. Pour de plus amples informations, veuillez consulter Résolution des problèmes de rotation.

AWS Secrets Manager Modèles de fonctions de rotation

AWS Secrets Manager fournit un ensemble de modèles de fonctions de rotation qui aident à automatiser la gestion sécurisée des informations d'identification pour différents systèmes et services de base de données. Les modèles sont des fonctions ready-to-use Lambda qui mettent en œuvre les meilleures pratiques en matière de rotation des informations d'identification, vous aidant ainsi à maintenir votre niveau de sécurité sans intervention manuelle.

Les modèles prennent en charge deux stratégies de rotation principales :

- Rotation mono-utilisateur qui met à jour les informations d'identification d'un seul utilisateur.
- Rotation des utilisateurs alternés qui permet de maintenir deux utilisateurs distincts afin d'éliminer les temps d'arrêt lors des changements d'informations d'identification.

Secrets Manager fournit également un modèle générique qui sert de point de départ pour tout type de secret.

Pour utiliser les modèles, consultez :

- Rotation automatique pour les secrets de base de données (console)
- Rotation automatique pour les secrets non liés à la base de données (console)

Pour écrire votre propre fonction de rotation, voir Écrire une fonction de rotation.

Modèles

- Amazon RDS et Amazon Aurora
  - Amazon RDS Db2 pour un utilisateur unique
  - Amazon RDS Db2 pour des utilisateurs en alternance
  - Amazon RDS MariaDB pour un utilisateur unique
  - Amazon RDS MariaDB pour des utilisateurs en alternance
  - Amazon RDS et Amazon Aurora MySQL pour un utilisateur unique

  - Utilisateurs en alternance Amazon RDS et Amazon Aurora MySQL
  - Amazon RDS Oracle pour un utilisateur unique
  - Amazon RDS Oracle pour des utilisateurs en alternance
  - Amazon RDS et Amazon Aurora PostgreSQL pour un utilisateur unique
  - Amazon RDS et Amazon Aurora PostgreSQL pour des utilisateurs en alternance
  - Amazon RDS Microsoft pour SQLServer un utilisateur unique
  - Amazon RDS Microsoft pour des utilisateurs SQLServer en alternance
- Amazon DocumentDB (compatible avec MongoDB)
  - Amazon DocumentDB pour un utilisateur unique
  - Amazon DocumentDB pour des utilisateurs en alternance
- Amazon Redshift
  - Amazon Redshift pour un utilisateur unique
  - Amazon Redshift pour des utilisateurs en alternance
- Amazon Timestream pour InfluxDB
  - Amazon Timestream pour InfluxDB en mode mono-utilisateur
  - Amazon Timestream pour les utilisateurs alternatifs d'InfluxDB
- Amazon ElastiCache
- Active Directory
  - Authentification Active Directory
  - Tab Active Directory
- Autres types de secrets

Amazon RDS et Amazon Aurora

Amazon RDS Db2 pour un utilisateur unique

- Nom du modèle : SecretsManager RDSDb2 RotationSingleUser
- Stratégie de rotation : Stratégie de rotation : utilisateur unique.
- Structure **`SecretString`** : the section called “Informations d'identification Amazon RDS et Aurora”.
- Code source : https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerRDSDb2RotationSingleUser/lambda _function.py

- Dépendance : python-ibmdb

Amazon RDS Db2 pour des utilisateurs en alternance

- Nom du modèle : SecretsManager RDSDb2 RotationMultiUser
- Stratégie de rotation : the section called "Utilisateurs en alternance".
- Structure **`SecretString`** : the section called "Informations d'identification Amazon RDS et Aurora".
- Code source : https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerRDSDb2RotationMultiUser/lambda _function.py
- Dépendance : python-ibmdb

Amazon RDS MariaDB pour un utilisateur unique

- Nom du modèle : SecretsManager RDSMaria DBRotation SingleUser
- Stratégie de rotation : Stratégie de rotation : utilisateur unique.
- Structure **`SecretString`** : the section called "Informations d'identification Amazon RDS et Aurora".
- Code source : https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerRDSMariaDBRotationSingleUser/lambda _function.py
- Dépendance : PyMy SQL 1.0.2. Si vous utilisez le mot de passe sha256 pour l'authentification, PyMy SQL [rsa]. Pour plus d'informations sur l'utilisation de packages avec du code compilé dans un environnement d'exécution Lambda, voir Comment ajouter des packages Python avec des fichiers binaires compilés à mon package de déploiement et rendre le package compatible avec Lambda ? dans le AWS Knowledge Center.

Amazon RDS MariaDB pour des utilisateurs en alternance

- Nom du modèle : SecretsManager RDSMaria DBRotation MultiUser
- Stratégie de rotation : the section called "Utilisateurs en alternance".
- Structure **`SecretString`** : the section called "Informations d'identification Amazon RDS et Aurora".
- Code source : https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerRDSMariaDBRotationMultiUser/lambda _function.py

- Dépendance : PyMy SQL 1.0.2. Si vous utilisez le mot de passe sha256 pour l'authentification, PyMy SQL [rsa]. Pour plus d'informations sur l'utilisation de packages avec du code compilé dans un environnement d'exécution Lambda, voir Comment ajouter des packages Python avec des fichiers binaires compilés à mon package de déploiement et rendre le package compatible avec Lambda ? dans le AWS Knowledge Center.

Amazon RDS et Amazon Aurora MySQL pour un utilisateur unique

- Nom du modèle : SecretsManager RDSMy SQLRotation SingleUser
- Stratégies de rotation : the section called "Utilisateur unique".
- **`SecretString`**Structure attendue : the section called "Informations d'identification Amazon RDS et Aurora".
- Code source : https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerRDSMySQLRotationSingleUser/lambda _function.py
- Dépendance : PyMy SQL 1.0.2. Si vous utilisez le mot de passe sha256 pour l'authentification, PyMy SQL [rsa]. Pour plus d'informations sur l'utilisation de packages avec du code compilé dans un environnement d'exécution Lambda, voir Comment ajouter des packages Python avec des fichiers binaires compilés à mon package de déploiement et rendre le package compatible avec Lambda ? dans le AWS Knowledge Center.

Utilisateurs en alternance Amazon RDS et Amazon Aurora MySQL

- Nom du modèle : SecretsManager RDSMy SQLRotation MultiUser
- Stratégies de rotation : the section called "Utilisateurs en alternance".
- **`SecretString`**Structure attendue : the section called "Informations d'identification Amazon RDS et Aurora".
- Code source : https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerRDSMySQLRotationMultiUser/lambda _function.py
- Dépendance : PyMy SQL 1.0.2. Si vous utilisez le mot de passe sha256 pour l'authentification, PyMy SQL [rsa]. Pour plus d'informations sur l'utilisation de packages avec du code compilé dans un environnement d'exécution Lambda, voir Comment ajouter des packages Python avec des fichiers binaires compilés à mon package de déploiement et rendre le package compatible avec Lambda ? dans le AWS Knowledge Center.

Amazon RDS Oracle pour un utilisateur unique

- Nom du modèle : SecretsManager RDSOracle RotationSingleUser
- Stratégies de rotation : the section called "Utilisateur unique".
- **`SecretString`**Structure attendue : the section called "Informations d'identification Amazon RDS et Aurora".
- Code source : https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerRDSOracleRotationSingleUser/lambda _function.py
- Dépendance : python-oracledb 2.4.1

Amazon RDS Oracle pour des utilisateurs en alternance

- Nom du modèle : SecretsManager RDSOracle RotationMultiUser
- Stratégies de rotation : the section called "Utilisateurs en alternance".
- **`SecretString`**Structure attendue : the section called "Informations d'identification Amazon RDS et Aurora".
- Code source : https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerRDSOracleRotationMultiUser/lambda _function.py
- Dépendance : python-oracledb 2.4.1

Amazon RDS et Amazon Aurora PostgreSQL pour un utilisateur unique

- Nom du modèle : SecretsManager RDSPostgre SQLRotation SingleUser
- Stratégies de rotation : Stratégie de rotation : utilisateur unique.
- **`SecretString`**Structure attendue : the section called "Informations d'identification Amazon RDS et Aurora".
- Code source : https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerRDSPostgreSQLRotationSingleUser/lambda _function.py
- Dépendance : PyGre SQL 5.0.7

Amazon RDS et Amazon Aurora PostgreSQL pour des utilisateurs en alternance

- Nom du modèle : SecretsManager RDSPostgre SQLRotation MultiUser
- Stratégies de rotation : the section called "Utilisateurs en alternance".

- **`SecretString`**Structure attendue : the section called "Informations d'identification Amazon RDS et Aurora".
- Code source : https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerRDSPostgreSQLRotationMultiUser/lambda _function.py
- Dépendance : PyGre SQL 5.0.7

Amazon RDS Microsoft pour SQLServer un utilisateur unique

- Nom du modèle : SecretsManager RDSSQLServer RotationSingleUser
- Stratégies de rotation : the section called "Utilisateur unique".
- **`SecretString`**Structure attendue : the section called "Informations d'identification Amazon RDS et Aurora".
- Code source : https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerRDSSQLServerRotationSingleUser/lambda _function.py
- Dépendance : Pymssql 2.2.2

Amazon RDS Microsoft pour des utilisateurs SQLServer en alternance

- Nom du modèle : SecretsManager RDSSQLServer RotationMultiUser
- Stratégies de rotation : the section called "Utilisateurs en alternance".
- **`SecretString`**Structure attendue : the section called "Informations d'identification Amazon RDS et Aurora".
- Code source : https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerRDSSQLServerRotationMultiUser/lambda _function.py
- Dépendance : Pymssql 2.2.2

Amazon DocumentDB (compatible avec MongoDB)

Amazon DocumentDB pour un utilisateur unique

- Nom du modèle : SecretsManagerMongo DBRotation SingleUser
- Stratégies de rotation : the section called "Utilisateur unique".
- **`SecretString`**Structure attendue : the section called "Informations d'identification Amazon DocumentDB".

- Code source : https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerMongoDBRotationSingleUser/lambda _function.py
- Dépendance : Pymongo 3.2

Amazon DocumentDB pour des utilisateurs en alternance

- Nom du modèle : SecretsManagerMongo DBRotation MultiUser
- Stratégies de rotation : the section called "Utilisateurs en alternance".
- **`SecretString`**Structure attendue : the section called "Informations d'identification Amazon DocumentDB".
- Code source : https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerMongoDBRotationMultiUser/lambda _function.py
- Dépendance : Pymongo 3.2

Amazon Redshift

Amazon Redshift pour un utilisateur unique

- Nom du modèle : SecretsManagerRedshiftRotationSingleUser
- Stratégies de rotation : the section called "Utilisateur unique".
- **`SecretString`**Structure attendue : the section called "Informations d'identification Amazon Redshift".
- Code source : https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerRedshiftRotationSingleUser/lambda _function.py
- Dépendance : PyGre SQL 5.0.7

Amazon Redshift pour des utilisateurs en alternance

- Nom du modèle : SecretsManagerRedshiftRotationMultiUser
- Stratégies de rotation : the section called "Utilisateurs en alternance".
- **`SecretString`**Structure attendue : the section called "Informations d'identification Amazon Redshift".
- Code source : https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerRedshiftRotationMultiUser/lambda _function.py

- Dépendance : PyGre SQL 5.0.7

Amazon Timestream pour InfluxDB

Pour utiliser ces modèles, consultez Comment Amazon Timestream pour InfluxDB utilise les secrets dans le manuel du développeur Amazon Timestream.

Amazon Timestream pour InfluxDB en mode mono-utilisateur

- Nom du modèle : SecretsManager Influx DBRotation SingleUser
- **`SecretString`**Structure attendue : the section called "Structure secrète d'Amazon Timestream pour InfluxDB".
- Code source : https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerInfluxDBRotationSingleUser/lambda _function.py
- Dépendance : client python InfluxDB 2.0

Amazon Timestream pour les utilisateurs alternatifs d'InfluxDB

- Nom du modèle : SecretsManagerInflux DBRotation MultiUser
- **`SecretString`**Structure attendue : the section called "Structure secrète d'Amazon Timestream pour InfluxDB".
- Code source : https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerInfluxDBRotationMultiUser/lambda _function.py
- Dépendance : client python InfluxDB 2.0

Amazon ElastiCache

Pour utiliser ce modèle, consultez Rotation automatique des mots de passe pour les utilisateurs dans le guide de ElastiCache l'utilisateur Amazon.

- Nom du modèle : SecretsManagerElasticacheUserRotation
- **`SecretString`**Structure attendue : the section called " ElastiCache Informations d'identification Amazon".
- Code source : https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerElasticacheUserRotation/lambda _function.py

Active Directory

Authentification Active Directory

- Nom du modèle : SecretsManagerActiveDirectoryRotationSingleUser
- **`SecretString`**Structure attendue : the section called "Informations d'identification Active Directory".
- Code source : https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerActiveDirectoryRotationSingleUser/lambda _function.py

Tab Active Directory

- Nom du modèle : SecretsManagerActiveDirectoryAndKeytabRotationSingleUser
- **`SecretString`**Structure attendue : the section called "Informations d'identification Active Directory".
- Code source : https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerActiveDirectoryAndKeytabRotationSingleUser/lambda _function.py
- Dépendances : msktutil

Autres types de secrets

Secrets Manager fournit ce modèle comme point de départ pour créer une fonction de rotation pour tout type de secret.

- Nom du modèle : SecretsManagerRotationTemplate
- Code source : https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerRotationTemplate/lambda _function.py

Autorisations de rôle d'exécution de la fonction de rotation Lambda pour AWS Secrets Manager

En effetthe section called "Rotation par fonction Lambda", lorsque Secrets Manager utilise une fonction Lambda pour faire pivoter un secret, Lambda assume un rôle d'exécution IAM et fournit ces informations d'identification au code de la fonction Lambda. Pour obtenir des instructions sur la configuration de la rotation automatique, voir :

- Rotation automatique pour les secrets de base de données (console)
- Rotation automatique pour les secrets non liés à la base de données (console)
- Rotation automatique (AWS CLI)

Les exemples suivants présentent des stratégies en ligne pour les rôles d'exécution de fonctions de rotation Lambda. Pour créer un rôle d'exécution et attacher une politique d'autorisations, consultez Rôle d'exécution AWS Lambda.

Exemples :

- Stratégie pour un rôle d'exécution de fonction de rotation Lambda
- Déclaration de stratégie pour une clé gérée par le client
- Déclaration de stratégie pour une stratégie d'utilisateurs en alternance

Stratégie pour un rôle d'exécution de fonction de rotation Lambda

L'exemple de stratégie suivant autorise la fonction de rotation à :

- Exécutez les opérations de Secrets Manager pour*SecretARN*.
- Créer un nouveau mot de passe.
- Mettre en place la configuration requise si votre base de données ou votre service s'exécute dans un VPC. Consultez Configuration d'une fonction Lambda pour accéder aux ressources d'un VPC.

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"
        }
    ]
}
```

Déclaration de stratégie pour une clé gérée par le client

Si le secret est chiffré avec une clé KMS autre que la Clé gérée par AWS `aws/secretsmanager`, vous devez accorder au rôle d'exécution Lambda l'autorisation d'utiliser la clé. Vous pouvez utiliser le contexte de chiffrement SecretARN pour limiter l'utilisation de la fonction de déchiffrement, de sorte que le rôle de la fonction de rotation n'ait accès que pour déchiffrer le secret qu'il est chargé de faire pivoter. L'exemple suivant montre une instruction à ajouter à la stratégie de rôle d'exécution pour déchiffrer le secret à l'aide de la clé KMS.

```
    {
        "Effect": "Allow",
        "Action": [
            "kms:Decrypt",
            "kms:DescribeKey",
            "kms:GenerateDataKey"
        ],
        "Resource": "KMSKeyARN"
        "Condition": {
            "StringEquals": {
                "kms:EncryptionContext:SecretARN": "SecretARN"
            }
```

```
        }
    }
```

Pour utiliser la fonction de rotation pour plusieurs secrets chiffrés à l'aide d'une clé gérée par le client, ajoutez une instruction comme dans l'exemple suivant pour autoriser le rôle d'exécution à déchiffrer le secret.

```
    {
        "Effect": "Allow",
        "Action": [
            "kms:Decrypt",
            "kms:DescribeKey",
            "kms:GenerateDataKey"
        ],
        "Resource": "KMSKeyARN"
        "Condition": {
            "StringEquals": {
                "kms:EncryptionContext:SecretARN": [
                    "arn1",
                    "arn2"
                ]
            }
        }
    }
```

Déclaration de stratégie pour une stratégie d'utilisateurs en alternance

Pour plus d'informations sur la stratégie de rotation des utilisateurs en alternance, consultez the section called “Stratégies de rotation des fonctions Lambda”.

Pour un secret contenant des informations d'identification Amazon RDS, si vous utilisez la stratégie des utilisateurs alternatifs et que le secret du superutilisateur est géré par Amazon RDS, vous devez également autoriser la fonction de rotation à appeler en lecture seule APIs sur Amazon RDS afin qu'elle puisse obtenir les informations de connexion à la base de données. Nous vous recommandons de joindre la politique AWS gérée Amazon RDSRead OnlyAccess.

L'exemple de politique suivant autorise la fonction à :

- Exécutez les opérations de Secrets Manager pour*SecretARN*.

- Récupérez les informations d'identification dans le secret du super-utilisateur. Secrets Manager utilise les informations d'identification du secret du super-utilisateur pour mettre à jour les informations d'identification dans le secret faisant l'objet d'une rotation.
- Créer un nouveau mot de passe.
- Mettre en place la configuration requise si votre base de données ou votre service s'exécute dans un VPC. Pour de plus amples informations, veuillez consulter Configuration d'une fonction Lambda pour accéder aux ressources d'accès dans un VPC.

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"
        }
    ]
}
```

Accès au réseau pour la fonction AWS Lambda de rotation

En the section called "Rotation par fonction Lambda" effet, lorsque Secrets Manager utilise une fonction Lambda pour faire pivoter un secret, la fonction de rotation Lambda doit pouvoir accéder au secret. Si votre secret contient des informations d'identification, la fonction Lambda doit également être en mesure d'accéder à la source de ces informations d'identification, telle qu'une base de données ou un service.

Pour accéder à un secret

- Votre fonction de rotation Lambda doit être en mesure d'accéder à un point de terminaison Secrets Manager. Si votre fonction Lambda peut accéder à Internet, vous pouvez utiliser un point de terminaison public. Pour trouver un point de terminaison, consultez the section called "Points de terminaison Secrets Manager".

- Si votre fonction Lambda s'exécute dans un VPC qui ne dispose pas d'un accès Internet, nous vous recommandons de configurer les points de terminaison privés du service Secrets Manager dans votre VPC. Votre VPC peut ensuite intercepter les demandes adressées au point de terminaison régional public et les rediriger vers le point de terminaison privé. Pour de plus amples informations, veuillez consulter Points de terminaison d'un VPC (AWS PrivateLink).

- Vous pouvez accorder à votre fonction Lambda un accès au point de terminaison public Secrets Manager en ajoutant une Passerelle NAT ou une passerelle Internet à votre VPC, ce qui permet au trafic de votre VPC d'atteindre le point de terminaison public. En procédant ainsi, votre VPC est exposé à certains risques dans la mesure où une adresse IP pour la passerelle peut être attaquée depuis le réseau Internet public.

(Facultatif) Pour accéder à la base de données ou au service

Pour les secrets tels que les clés d'API, il n'y a pas de base de données ou de service source à mettre à jour en même temps que le secret.

Si votre base de données ou votre service s'exécute sur une EC2 instance Amazon dans un VPC, nous vous recommandons de configurer votre fonction Lambda pour qu'elle s'exécute dans le même VPC. La fonction de rotation peut alors communiquer directement avec votre service. Pour plus d'informations, consultez Configuring VPC access (Configuration de l'accès VPC).

Pour permettre à la fonction Lambda d'accéder à la base de données ou au service, vous devez vous assurer que les groupes de sécurité attachés à votre fonction de rotation Lambda autorisent les connexions sortantes vers la base de données ou le service. Vous devez également vous assurer que les groupes de sécurité attachés à votre base de données ou service autorisent les connexions entrantes depuis la fonction de rotation Lambda.

Résoudre AWS Secrets Manager la rotation d'

Pour un grand nombre de services, Secrets Manager utilise une fonction Lambda afin d'effectuer la rotation des secrets. Pour de plus amples informations, veuillez consulter the section called “Rotation par fonction Lambda”. La fonction de rotation Lambda interagit avec la base de données ou le service auquel le secret est destiné, ainsi qu'avec Secrets Manager. Lorsque la rotation ne fonctionne pas comme prévu, vous devez avant toute chose vérifier les problèmes de CloudWatch rotation.

> **Note**
>
> Certains services peuvent effectuer la gestion des secrets à votre place, notamment la gestion de la rotation automatique. Pour de plus amples informations, veuillez consulter the section called “Rotation gérée”.

Rubriques

- Comment résoudre les défaillances de rotation secrètes dans les fonctions AWS Lambda
- Aucune activité après « Found credentials in environment variables » (Informations d'identification trouvées dans les variables d'environnement)
- Aucune activité après « createSecret »
- Erreur : « L'accès à KMS n'est pas autorisé »

- Erreur : « Key is missing from secret JSON » (Le fichier JSON secret ne contient pas de clé)
- Erreur : « SetSecret: Unable to log into database » (SetSecret : connexion à la base de données impossible)
- Erreur : « Impossible d'importer le module "lambda_function" »
- Mise à jour d'une fonction de rotation existante depuis Python 3.7 vers 3.9
- AWS Lambda rotation secrète en cas d'PutSecretValueéchec
- Erreur : « Erreur lors de l'exécution de Lambda <arn> pendant l'<a rotation>étape »

Comment résoudre les défaillances de rotation secrètes dans les fonctions AWS Lambda

Si vous rencontrez des échecs de rotation secrets avec vos fonctions Lambda, suivez les étapes ci-dessous pour résoudre le problème.

Causes possibles :

- Exécutions simultanées insuffisantes pour votre fonction Lambda
- Conditions de course dues aux multiples appels d'API pendant la rotation
- Logique de la fonction Lambda incorrecte
- Problèmes de mise en réseau entre la fonction Lambda et la base de données

Étapes de dépannage généraux

1. Analyser CloudWatch les journaux :
   - Recherchez les messages d'erreur spécifiques ou les comportements inattendus dans les journaux des fonctions Lambda
   - Vérifiez que toutes les étapes de rotation (CreateSecretSetSecret,TestSecret,,FinishSecret) sont tentées
2. Passez en revue les appels d'API pendant la rotation :
   - Évitez d'effectuer des appels d'API mutants sur le secret pendant la rotation Lambda
   - Assurez-vous qu'il n'y a aucune condition de race entre RotateSecret et les PutSecretValue appels
3. Vérifiez la logique de la fonction Lambda :

- Vérifiez que vous utilisez le dernier AWS exemple de code pour la rotation secrète
- Si vous utilisez un code personnalisé, consultez-le pour gérer correctement toutes les étapes de rotation

4. Vérifiez la configuration du réseau :

   - Vérifiez que les règles du groupe de sécurité autorisent la fonction Lambda à accéder à la base de données
   - Garantir un accès approprié au point de terminaison VPC ou au point de terminaison public pour Secrets Manager

5. Testez les versions secrètes :

   - Vérifiez que la AWSCURRENT version du secret autorise l'accès à la base de données
   - Vérifiez si AWSPREVIOUS nos AWSPENDING versions sont valides

6. Effacer les rotations en attente :

   - Si la rotation échoue systématiquement, effacez le libellé de la AWSPENDING mise en scène et réessayez la rotation

7. Vérifiez les paramètres de simultanéité Lambda :

   - Vérifiez que les paramètres de simultanéité sont adaptés à votre charge de travail
   - Si vous suspectez des problèmes de simultanéité, consultez la section « Résolution des problèmes de rotation liés à la simultanéité »

Aucune activité après « Found credentials in environment variables » (Informations d'identification trouvées dans les variables d'environnement)

Si rien ne se passe après « Found credentials in environment variables » (Informations d'identification trouvées dans les variables d'environnement) et que la durée de la tâche est longue, par exemple si elle dépasse le délai Lambda par défaut de 30 000 ms, il se peut que la fonction Lambda expire alors qu'elle essaie d'atteindre le point de terminaison Secrets Manager.

Votre fonction de rotation Lambda doit être en mesure d'accéder à un point de terminaison Secrets Manager. Si votre fonction Lambda peut accéder à Internet, vous pouvez utiliser un point de terminaison public. Pour trouver un point de terminaison, consultez the section called “Points de terminaison Secrets Manager”.

Si votre fonction Lambda s'exécute dans un VPC qui ne dispose pas d'un accès Internet, nous vous recommandons de configurer les points de terminaison privés du service Secrets Manager dans votre VPC. Votre VPC peut ensuite intercepter les demandes adressées au point de terminaison régional public et les rediriger vers le point de terminaison privé. Pour de plus amples informations, veuillez consulter Points de terminaison d'un VPC (AWS PrivateLink).

Vous pouvez accorder à votre fonction Lambda un accès au point de terminaison public Secrets Manager en ajoutant une Passerelle NAT ou une passerelle Internet à votre VPC, ce qui permet au trafic de votre VPC d'atteindre le point de terminaison public. En procédant ainsi, votre VPC est exposé à certains risques dans la mesure où une adresse IP pour la passerelle peut être attaquée depuis le réseau Internet public.

Aucune activité après « createSecret »

Les problèmes suivants peuvent entraîner l'arrêt de la rotation après createSecret :

Le réseau VPC n'autorise ACLs pas le trafic HTTPS entrant et sortant.

> Pour plus d'informations, consultez la section Contrôler le trafic vers les sous-réseaux à l'aide du réseau ACLs dans le guide de l'utilisateur Amazon VPC.

La configuration du délai d'expiration de la fonction Lambda est trop courte pour effectuer la tâche.

> Pour plus d'informations, consultez Configuration des options de fonction Lambda dans le Guide du développeur AWS Lambda .

Le point de terminaison d'un VPC Secrets Manager n'autorise pas le VPC à l'entrée dans CIDRs les groupes de sécurité attribués.

> Pour plus d'informations, consultez la rubrique Contrôler le trafic vers les ressources à l'aide de groupes de sécurité dans le Guide de l'utilisateur Amazon VPC.

La stratégie de point de terminaison d'un VPC Secrets Manager n'autorise pas Lambda à utiliser le point de terminaison d'un VPC.

> Pour de plus amples informations, veuillez consulter the section called "Points de terminaison d'un VPC (AWS PrivateLink)".

Le secret utilise la rotation des utilisateurs en alternance, le secret du superutilisateur est géré par Amazon RDS et la fonction Lambda ne peut pas accéder à l'API RDS.

> Pour la rotation des utilisateurs en alternance lorsque le secret du superutilisateur est géré par un autre AWS service, la fonction de rotation Lambda doit pouvoir appeler le point de

terminaison du service pour obtenir les informations de connexion à la base de données. Nous vous recommandons de configurer un point de terminaison d'un VPC pour le service de base de données. Pour plus d'informations, consultez :

- API Amazon RDS et points de terminaison d'un VPC d'interface dans le Guide de l'utilisateur Amazon RDS.
- Utilisation des points de terminaison VPC dans le Guide de gestion Amazon Redshift.

Erreur : « L'accès à KMS n'est pas autorisé »

Si vous voyez `ClientError: An error occurred (AccessDeniedException) when calling the GetSecretValue operation: Access to KMS is not allowed`, la fonction de rotation n'est pas autorisée à déchiffrer le secret à l'aide de la clé KMS qui a été utilisée pour le chiffrer. La stratégie d'autorisation peut contenir une condition qui limite le contexte de chiffrement à un secret spécifique. Pour plus d'informations sur l'autorisation requise, veuillez consulter la rubrique the section called “Déclaration de stratégie pour une clé gérée par le client”.

Erreur : « Key is missing from secret JSON » (Le fichier JSON secret ne contient pas de clé)

Une fonction de rotation Lambda nécessite que la valeur de secret se trouve dans une structure JSON spécifique. Si vous voyez cette erreur, il se peut que le fichier JSON ne dispose pas d'une clé à laquelle la fonction de rotation a essayé d'accéder. Pour plus d'informations sur la structure JSON de chaque type de secret, consultez the section called “Structure JSON d'un secret”.

Erreur : « SetSecret: Unable to log into database » (SetSecret : connexion à la base de données impossible)

Les problèmes suivants peuvent entraîner cette erreur :

La fonction de rotation ne peut pas accéder à la base de données.

Si la durée de la tâche est longue, par exemple plus de 5 000 ms, il est possible que la fonction de rotation Lambda ne parvienne pas à accéder à la base de données via le réseau.

Si votre base de données ou votre service est en cours d'exécution sur une EC2 instance Amazon dans un VPC, nous vous recommandons de configurer votre fonction Lambda de sorte qu'elle s'exécute dans le même VPC. La fonction de rotation peut alors communiquer directement avec

votre service. Pour plus d'informations, consultez Configuring VPC access (Configuration de l'accès VPC).

Pour permettre à la fonction Lambda d'accéder à la base de données ou au service, vous devez vous assurer que les groupes de sécurité attachés à votre fonction de rotation Lambda autorisent les connexions sortantes vers la base de données ou le service. Vous devez également vous assurer que les groupes de sécurité attachés à votre base de données ou service autorisent les connexions entrantes depuis la fonction de rotation Lambda.

Les informations d'identification contenues dans le secret sont incorrectes.

Si la durée de la tâche est courte, il se peut que la fonction de rotation Lambda ne parvienne pas à s'authentifier à l'aide des informations d'identification contenues dans le secret. Vérifiez les informations d'identification en vous connectant manuellement avec les informations contenues dans les `AWSPREVIOUS` versions `AWSCURRENT` et du secret à l'aide de la AWS CLI commande `get-secret-value`.

La base de données utilise **`scram-sha-256`** pour crypter les mots de passe.

Si votre base de données est Aurora PostgreSQL version 13 ou une version plus récente et utilise `scram-sha-256` pour crypter les mots de passe, mais que la fonction de rotation utilise la `libpq` version 9 ou une version moins récente qui ne prend pas en charge `scram-sha-256`, la fonction de rotation ne peut pas se connecter à la base de données.

Pour déterminer quels utilisateurs de base de données utilisent le chiffrement **`scram-sha-256`**

- Consultez la section Vérification de la présence d'utilisateurs utilisant des mots de passe (français non garanti) dans le journal SCRAM Authentication (Authentification SCRAM) dans RDS pour PostgreSQL 13 (français non garanti).

Pour déterminer quelle version de **`libpq`** votre fonction de rotation utilise

1. Sur un ordinateur Linux, sur la console Lambda, accédez à votre fonction de rotation et téléchargez le package de déploiement. Décompressez le fichier zip dans un répertoire de travail.
2. Dans le répertoire de travail d'une ligne de commande, exécutez :

   ```
   readelf -a libpq.so.5 | grep RUNPATH
   ```
3. Si vous voyez la chaîne *`PostgreSQL-9.4.x`*, ou toute version majeure inférieure à 10, cela signifie que la fonction de rotation n'est pas prise en charge `scram-sha-256`.

- Sortie pour une fonction de rotation qui ne prend pas en charge `scram-sha-256` :

  ```
  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]
  ```

- Sortie pour une fonction de rotation prenant en charge `scram-sha-256` :

  ```
  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**
>
> Si vous avez configuré la rotation automatique des secrets avant le 30 décembre 2021, votre fonction de rotation intégrée à une version antérieure `libpq` ne prend pas en charge`scram-sha-256`. Afin de prendre en charge `scram-sha-256`, vous devez recréer votre fonction de rotation.

La base de données nécessite un accès SSL/TLS.

Si votre base de données nécessite une connexion SSL/TLS, mais que la fonction de rotation utilise une connexion non chiffrée, la fonction de rotation ne peut pas se connecter à la base de données. Les fonctions de rotation pour Amazon RDS (sauf Oracle et Db2) et Amazon DocumentDB utilisent automatiquement le protocole SSL (Secure Socket Layer) ou le protocole TLS (Transport Layer Security) pour se connecter à votre base de données, si elle est disponible. Sinon, elles utilisent une connexion non chiffrée.

> **Note**
>
> Si vous avez configuré la rotation automatique des secrets avant le 20 décembre 2021, votre fonction de rotation peut être basée sur un modèle antérieur qui ne prenait pas en chargeSSL/TLS. To support connections that use SSL/TLS. Vous devez donc recréer votre fonction de rotation.

Pour déterminer quand votre fonction de rotation a été créée

1. Dans la console Secrets Manager https://console.aws.amazon.com/secretsmanager/, ouvrez votre secret. Dans la section Rotation configuration (Configuration de la rotation), sous Lambda rotation function (Fonction de rotation Lambda), vous pouvez voir Lambda function ARN (ARN de la fonction Lambda), par exemple, `arn:aws:lambda:`*`aws-region`*`:`*`123456789012`*`:function:`*`SecretsManagerMyRotationFunction`* . Copiez le nom de la fonction à partir de la fin de l'ARN. Dans l'exemple présent, il s'agit de *`SecretsManagerMyRotationFunction`* .
2. Dans la AWS Lambda console https://console.aws.amazon.com/lambda/, sous Fonctions, collez le nom de votre fonction Lambda dans la zone de recherche, appuyez sur Entrée, puis choisissez la fonction Lambda.
3. Dans la page des détails de la fonction, dans l'onglet Configuration, sous Tags (Balises), copiez la valeur à côté de la clé aws:cloudformation:stack-name.
4. Dans la AWS CloudFormation console https://console.aws.amazon.com/cloudformation, sous Stacks (Piles), collez la valeur clé dans la zone de recherche, puis appuyez sur Entrée.
5. La liste des piles est filtrée afin que seule la pile qui a créé la fonction de rotation Lambda apparaisse. Dans la colonne Created date (Date de création), affichez la date à laquelle la pile a été créée. Il s'agit de la date à laquelle la fonction de rotation Lambda a été créée.

Erreur : « Impossible d'importer le module "lambda_function" »

Cette erreur peut s'afficher si vous exécutez une fonction Lambda antérieure qui a été automatiquement mise à niveau de Python 3.7 vers une version plus récente de Python. Pour résoudre l'erreur, vous pouvez rétablir la version de la fonction Lambda vers Python 3.7, puis the section called “Mise à jour d'une fonction de rotation existante depuis Python 3.7 vers 3.9”. Pour plus d'informations, veuillez consulter la rubrique Pourquoi la rotation de ma fonction Lambda Secrets Manager a-t-elle échoué avec le message d'erreur « module pg introuvable » ? dans AWS re:Post.

Mise à jour d'une fonction de rotation existante depuis Python 3.7 vers 3.9

Certaines fonctions de rotation créées avant novembre 2022 utilisaient Python 3.7. Le AWS SDK pour Python a cessé de prendre en charge Python 3.7 en décembre 2023. Pour plus d'informations, consultez les mises à jour de la politique de support de Python pour AWS SDKs et les outils. Pour passer à une nouvelle fonction de rotation utilisant Python 3.9, vous pouvez ajouter une propriété d'exécution à une fonction de rotation existante ou recréer la fonction de rotation.

Pour rechercher quelles fonctions de rotation Lambda utilisent Python 3.7

1. Connectez-vous à la AWS Lambda console AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/lambda/.
2. Dans la liste des fonctions, filtrez pour **`SecretsManager`**.
3. Dans la liste filtrée des fonctions, sous Exécution, recherchez Python 3.7.

Pour effectuer la mise à jour vers Python 3.9 :

- Option 1 : recréer la fonction de rotation en utilisant AWS CloudFormation
- Option 2 : mettre à jour l'exécution de la fonction de rotation existante à l'aide de AWS CloudFormation
- Option 3 : pour les AWS CDK utilisateurs, mettez à jour la bibliothèque CDK

Option 1 : recréer la fonction de rotation en utilisant AWS CloudFormation

Lorsque vous utilisez la console Secrets Manager pour activer la rotation, Secrets Manager utilise AWS CloudFormation pour créer les ressources nécessaires, y compris la fonction de rotation Lambda. Si vous avez utilisé la console pour activer la rotation ou si vous avez créé la fonction de rotation à l'aide d'une AWS CloudFormation pile, vous pouvez utiliser la même AWS CloudFormation pile pour recréer la fonction de rotation sous un nouveau nom. La nouvelle fonction utilise la version la plus récente de Python.

Pour trouver la AWS CloudFormation pile qui a créé la fonction de rotation

- Dans la page des détails de la fonction Lambda, sous l'onglet Configuration, choisissez Balises. Afficher l'ARN en regard de aws:cloudformation:stack-id.

  Le nom de la pile est intégré dans l'ARN, comme illustré dans l'exemple suivant.

- ARN : `arn:aws:cloudformation:us-west-2:408736277230:stack/`**`SecretsManagerRDSMySQLRotationSingleUser5c2-SecretRotationScheduleHostedRotationLambda`**`-3CUDHZMDMBO8/79fc9050-2eef-11ed-`
- Nom de la pile : **`SecretsManagerRDSMySQLRotationSingleUser5c2-SecretRotationScheduleHostedRotationLambda`**

Pour recréer une fonction de rotation (AWS CloudFormation)

1. Dans AWS CloudFormation, recherchez la pile par son nom, puis choisissez Mettre à jour.

   Si une boîte de dialogue vous recommande de mettre à jour la pile racine s'affiche, choisissez Accéder à la pile racine, puis choisissez Mettre à jour.
2. Sur la page Mettre à jour la pile, sous Préparer le modèle, choisissez Modifier dans Application Composer, puis sous Modifier le modèle dans Application Composer, cliquez sur le bouton Modifier dans Application Composer.
3. Dans Application Composer, procédez comme suit :

   a. Dans le code du modèle, dans`SecretRotationScheduleHostedRotationLambda`, remplacez la valeur pour `"functionName": "SecretsManagerTestRotationRDS"` par un nouveau nom de fonction, par exemple en JSON, **`"functionName": "SecretsManagerTestRotationRDSupdated"`**

   b. Choisissez Mettre à jour le modèle.

   c. Dans la AWS CloudFormation boîte de dialogue Continuer à, choisissez Confirmer et passez à AWS CloudFormation.
4. Continuez à suivre le flux de travail de la AWS CloudFormation pile, puis choisissez Envoyer.

Option 2 : mettre à jour l'exécution de la fonction de rotation existante à l'aide de AWS CloudFormation

Lorsque vous utilisez la console Secrets Manager pour activer la rotation, Secrets Manager utilise AWS CloudFormation pour créer les ressources nécessaires, y compris la fonction de rotation Lambda. Si vous avez utilisé la console pour activer la rotation ou si vous avez créé la fonction de rotation à l'aide d'une AWS CloudFormation pile, vous pouvez utiliser la même AWS CloudFormation pile pour mettre à jour l'exécution de la fonction de rotation.

Pour trouver la AWS CloudFormation pile qui a créé la fonction de rotation

- Dans la page des détails de la fonction Lambda, sous l'onglet Configuration, choisissez Balises. Afficher l'ARN en regard de aws:cloudformation:stack-id.

  Le nom de la pile est intégré dans l'ARN, comme illustré dans l'exemple suivant.

  - ARN : `arn:aws:cloudformation:us-west-2:408736277230:stack/`**`SecretsManagerRDSMySQLRotationSingleUser5c2-SecretRotationScheduleHostedRotationLambda`**`-3CUDHZMDMBO8/79fc9050-2eef-11ed-`
  - Nom de la pile : **`SecretsManagerRDSMySQLRotationSingleUser5c2-SecretRotationScheduleHostedRotationLambda`**

Pour mettre à jour l'exécution d'une fonction de rotation (AWS CloudFormation)

1. Dans AWS CloudFormation, recherchez la pile par son nom, puis choisissez Mettre à jour.

   Si une boîte de dialogue vous recommande de mettre à jour la pile racine s'affiche, choisissez Accéder à la pile racine, puis choisissez Mettre à jour.
2. Sur la page Mettre à jour la pile, sous Préparer le modèle, choisissez Modifier dans Application Composer, puis sous Modifier le modèle dans Application Composer, cliquez sur le bouton Modifier dans Application Composer.
3. Dans Application Composer, procédez comme suit :

   a. Dans le modèle JSON, pour le`SecretRotationScheduleHostedRotationLambda`, under`Properties`, under`Parameters`, add**`"runtime": "python3.9"`**.

   b. Choisissez Mettre à jour le modèle.

   c. Dans la AWS CloudFormation boîte de dialogue Continuer à, choisissez Confirmer et passez à AWS CloudFormation.
4. Continuez à suivre le flux de travail de la AWS CloudFormation pile, puis choisissez Envoyer.

Option 3 : pour les AWS CDK utilisateurs, mettez à jour la bibliothèque CDK

Si vous avez utilisé la version v2.94.0 AWS CDK avant la version v2.94.0 pour configurer la rotation de votre secret, vous pouvez mettre à jour la fonction Lambda en passant à la version v2.94.0 ou ultérieure. Pour plus d'informations, consultez le Guide du développeur AWS Cloud Development Kit (AWS CDK) v2.

AWS Lambda rotation secrète en cas d'**`PutSecretValue`**échec

Si vous utilisez un rôle assumé ou une rotation entre comptes avec Secrets Manager et que vous trouvez un RotationFailed événement contenant le message suivant : La version *`VERSION_ID`* secrète en attente de Secret n'*`SECRET_ARN`* a pas été créée par *`LAMBDA_ARN.`* Lambda. Supprimez `AWSPENDING` le libellé intermédiaire et relancez la rotation, vous devez mettre à jour votre fonction Lambda pour utiliser le paramètre. CloudTrail `RotationToken`

Mettre à jour la fonction de rotation Lambda pour inclure **`RotationToken`**

1. Téléchargez le code de la fonction Lambda
   - Ouvrez la console Lambda
   - Dans le volet de navigation, choisissez Fonctions
   - Sélectionnez votre fonction de rotation secrète Lambda pour le nom de la fonction
   - Pour le téléchargement, choisissez l'une des options suivantes : code de fonction .zip, AWS SAM fichier, les deux
   - Cliquez sur OK pour enregistrer la fonction sur votre ordinateur local.
2. Modifier `Lambda_handler`

   Incluez le paramètre rotation_token dans l'étape create_secret pour la rotation entre comptes :

```
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. Modifier le `create_secret` code

   Révisez la `create_secret` fonction pour accepter et utiliser le `rotation_token` paramètre :

```
# 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. Téléchargez le code de la fonction Lambda mis à jour

   Après avoir mis à jour le code de votre fonction Lambda, téléchargez-le pour faire pivoter votre secret.

Erreur : « Erreur lors de l'exécution de Lambda *<arn>* pendant l'*<a rotation>*étape »

Si vous rencontrez des échecs de rotation secrète intermittents lorsque votre fonction Lambda est bloquée dans une boucle d'ensembles, par exemple entre CreateSecret etSetSecret, le problème peut être lié aux paramètres de simultanéité.

Étapes de résolution des problèmes de simultanéité

> **Warning**
>
> La définition du paramètre de simultanéité provisionné sur une valeur inférieure à 10 peut entraîner un ralentissement dû à un nombre insuffisant de threads d'exécution pour la fonction Lambda. Pour plus d'informations, consultez la section Comprendre la simultanéité réservée et la simultanéité provisionnée dans le Guide du AWS Lambda AWS Lambda développeur.

1. Vérifiez et ajustez les paramètres de simultanéité Lambda :
   - Vérifiez que ce `reserved_concurrent_executions` paramètre n'est pas trop bas (par exemple, 1)
   - Si vous utilisez la simultanéité réservée, réglez-la sur au moins 10
   - Envisagez d'utiliser la simultanéité sans réserve pour plus de flexibilité
2. Pour la simultanéité provisionnée :
   - Ne définissez pas explicitement le paramètre de simultanéité provisionné (par exemple, dans Terraform).
   - Si vous devez le définir, utilisez une valeur d'au moins 10.
   - Effectuez des tests approfondis pour vous assurer que la valeur choisie convient à votre cas d'utilisation.
3. Surveillez et réglez la simultanéité :
   - Calculez la simultanéité à l'aide de cette formule : Concurrence = (nombre moyen de demandes par seconde) * (durée moyenne des demandes en secondes). Pour plus d'informations, voir Estimation de la simultanéité réservée.
   - Observez et enregistrez les valeurs pendant les rotations afin de déterminer les paramètres de simultanéité appropriés.
   - Soyez prudent lorsque vous définissez de faibles valeurs de simultanéité. Ils peuvent provoquer un ralentissement s'il n'y a pas suffisamment de threads d'exécution disponibles.

Pour plus d'informations sur la configuration de la simultanéité Lambda, consultez les sections Configuration de la simultanéité réservée et Configuration de la simultanéité provisionnée dans le Guide du développeur. AWS Lambda

Horaires de rotation

Secrets Manager fait pivoter votre secret selon un calendrier pendant une période de rotation que vous avez définie. Pour définir le calendrier et la fenêtre, vous utilisez une expression cron () ou rate () ainsi qu'une durée de fenêtre. Secrets Manager effectue une rotation de votre secret à tout moment lors d'une fenêtre de rotation. Vous pouvez alterner un secret toutes les quatre heures au cours d'une période de rotation d'une heure seulement.

Pour activer la rotation, consultez :

- the section called "Rotation gérée"
- the section called "Rotation automatique pour les secrets de base de données (console)"
- the section called "Rotation automatique pour les secrets non liés à la base de données (console)"

Les planifications de rotation de Secrets Manager utilisent le fuseau horaire UTC.

Fenêtres rotatives

Une fenêtre de rotation de Secrets Manager est similaire à une fenêtre de maintenance. Vous définissez la fenêtre de rotation lorsque vous souhaitez faire pivoter votre secret, et Secrets Manager fait pivoter votre secret à un moment donné pendant la fenêtre de rotation.

Les fenêtres de rotation de Secrets Manager démarrent toujours à l'heure. Pour un calendrier de rotation qui utilise une `rate()` expression en jours, la fenêtre de rotation commence à minuit. Vous pouvez définir l'heure de début de la fenêtre de rotation à l'aide d'une `cron()` expression. Pour obtenir des exemples, consultez the section called "Expressions Cron".

Par défaut, la fenêtre de rotation se ferme au bout d'une heure pour un calendrier de rotation en heures, et à la fin de la journée pour un calendrier de rotation en jours.

Pour modifier la longueur de la fenêtre de rotation, définissez la durée de la fenêtre. Vous pouvez définir une fenêtre de rotation aussi petite qu'une heure. La fenêtre de rotation ne doit pas s'étendre jusqu'à la fenêtre de rotation suivante. En d'autres termes, pour un calendrier de rotation en heures, vérifiez que la fenêtre de rotation est inférieure ou égale au nombre d'heures entre les rotations. Pour un calendrier de rotation en jours, vérifiez que l'heure de début et la durée de la fenêtre sont inférieures ou égales à 24 heures.

Expressions de fréquence

Les expressions tarifaires de Secrets Manager ont le format suivant, où *`Value`* est un entier positif et *`Unit`* peut être `hourhours`,`day`, ou `days` :

```
rate(Value Unit)
```

Vous pouvez effectuer la rotation d'un secret toutes les quatre heures. La période de rotation maximale est de 999 jours. Exemples :

- `rate(4 hours)` signifie que la rotation d'un secret est effectuée toutes les quatre heures.
- `rate(1 day)` signifie que la rotation d'un secret est effectuée tous les jours.

- `rate(10 days)` signifie que la rotation d'un secret est effectuée tous les 10 jours.

Expressions Cron

Les expressions cron de Secrets Manager ont le format suivant :

```
cron(Minutes Hours Day-of-month Month Day-of-week Year)
```

Une expression cron qui inclut des incréments d'heures est réinitialisée chaque jour. Par exemple, `cron(0 4/12 * * ? *)` signifie 4 h 00, 16 h 00, puis 4 h 00, 16 h 00 le jour suivant. Les planifications de rotation de Secrets Manager utilisent le fuseau horaire UTC.

Exemple de programme	Expression
Toutes les huit heures à partir de minuit.	`cron(0 /8 * * ? *)`
Toutes les huit heures à partir de 8 h 00.	`cron(0 8/8 * * ? *)`
Toutes les dix heures, à partir de 2 h 00. Les fenêtres de rotation débuteront à 2 h 00, 12 h00 et 22 h 00, puis le jour suivant à 2 h 00, 12 h 00 et 22 h 00.	`cron(0 2/10 * * ? *)`
Tous les jours à 10 h 00.	`cron(0 10 * * ? *)`
Tous les samedis à 18 h 00.	`cron(0 18 ? * SAT *)`
Le premier jour de chaque mois à 8 h 00.	`cron(0 8 1 * ? *)`
Le premier dimanche de tous les trois mois à 1h00.	`cron(0 1 ? 1/3 SUN#1 *)`
Le dernier jour de chaque mois à 17 h 00.	`cron(0 17 L * ? *)`
Du lundi au vendredi à 8 h 00.	`cron(0 8 ? * MON-FRI *)`
Le premier et le 15e jour de chaque mois à 16 h 00.	`cron(0 16 1,15 * ? *)`

Exemple de programme	Expression
Le premier dimanche de chaque mois à minuit.	`cron(0 0 ? * SUN#1 *)`
À partir de janvier, tous les 11 mois le premier lundi à minuit.	`cron(0 0 ? 1/11 2#1 *)`

Exigences en matière d'expression cron dans Secrets Manager

Secrets Manager impose certaines restrictions quant à ce que vous pouvez utiliser pour les expressions cron. Une expression cron dans Secrets Manager doit indiquer 0 dans le champ des minutes, car les fenêtres de rotation de Secrets Manager s'ouvrent toutes les heures. Elle doit afficher * dans le champ de l'année, car Secrets Manager ne prend pas en charge les calendriers de rotation espacés de plus d'un an. Le tableau suivant répertorie les options que vous pouvez utiliser.

Champs	Valeurs	Caractères génériques
Minutes	Doit être 0	Aucun
Heures	0 – 23	Utilisez / (barre oblique) pour spécifier les incrément s. Par exemple, `2/10` cela signifie toutes les 10 heures à partir de 2 h 00. Vous pouvez effectuer la rotation d'un secret toutes les quatre heures.
D ay-of-month	1 – 31	Utilisez , (virgule) pour inclure des valeurs supplémentaires. Par exemple, `1,15` signifie le premier et le 15e jour du mois. Utilisez - (tiret) pour spécifier une plage. Par exemple, `1-15` renvoie aux journées du 1er au 15 du mois.

Champs	Valeurs	Caractères génériques
		Utilisez le caractère générique * (astérisque) pour inclure toutes les valeurs du champ. Par exemple, `*` signifie tous les jours du mois. Le caractère générique ? (point d'interrogation) indique l'un ou l'autre. Vous ne pouvez pas spécifier les champs `Day-of-month` et `Day-of-week` de la même expression cron. Si vous spécifiez une valeur dans l'un de ces champs, vous devez utiliser un signe ? (point d'interrogation) dans l'autre. Utilisez / (barre oblique) pour spécifier les incréments. Par exemple, `1/2` signifie tous les deux jours à partir du jour 1, en d'autres termes, les jours 1, 3, 5, et ainsi de suite. Utilisez L pour spécifier le dernier jour du mois. Utilisez ***DAY***L pour spécifier le dernier jour du mois. Par exemple, `SUNL` signifie le dernier dimanche du mois.

Champs	Valeurs	Caractères génériques
Mois	1–12 ou JAN–DEC	Utilisez , (virgule) pour inclure des valeurs supplémentaires. Par exemple, `JAN,APR,JUL,OCT` signifie janvier, avril, juillet et octobre. Utilisez - (tiret) pour spécifier une plage. Par exemple, `1-3` signifie du 1er au 3e mois de l'année. Utilisez le caractère générique * (astérisque) pour inclure toutes les valeurs du champ. Par exemple, `*` signifie tous les mois. Utilisez / (barre oblique) pour spécifier les incréments. Par exemple, `1/3` signifie tous les trois mois, à partir du premier mois, c'est-à-dire les mois 1, 4, 7 et 10.

Champs	Valeurs	Caractères génériques
D ay-of-week	1–7 ou SUN–SAT	Utilisez # pour spécifier le jour de la semaine dans un mois. Par exemple, `TUE#3` signifie le troisième mardi du mois. Utilisez , (virgule) pour inclure des valeurs supplémentaires. Par exemple, `1,4` signifie le premier et le quatrième jour de la semaine. Utilisez - (tiret) pour spécifier une plage. Par exemple, `1-4` signifie du 1er au 4e jour de la semaine. Utilisez le caractère générique * (astérisque) pour inclure toutes les valeurs du champ. Par exemple, `*` signifie tous les jours de la semaine. Le caractère générique ? (point d'interrogation) indique l'un ou l'autre. Vous ne pouvez pas spécifier les champs `Day-of-month` et `Day-of-week` de la même expression cron. Si vous spécifiez une valeur dans l'un de ces champs, vous devez utiliser un signe ? (point d'interrogation) dans l'autre. Utilisez / (barre oblique) pour spécifier les incréments. Par

Champs	Valeurs	Caractères génériques
		exemple, 1/2 signifie tous les deux jours de la semaine, à partir du premier jour, donc les jours 1, 3, 5 et 7. Utilisez L pour spécifier le dernier jour de la semaine.
Année	Doit être *	Aucun

Faites immédiatement pivoter un AWS Secrets Manager secret

Vous ne pouvez procéder à la rotation que d'un secret pour lequel la rotation a été configurée. Pour déterminer si un secret a été configuré pour la rotation, dans la console, affichez le secret et faites défiler jusqu'à la section Rotation configuration (Configuration de la rotation). Si Rotation status (Statut de la rotation) est défini sur Enabled (Activé), le secret est configuré pour la rotation. Si ce n'est pas le cas, voyez Rotation des secrets.

Pour mettre immédiatement un secret en rotation (console)

1. Ouvrez la console Secrets Manager à l'adresse https://console.aws.amazon.com/secretsmanager/.
2. Choisissez votre secret.
3. Sur la page des détails du secret, sous Configuration de la rotation, choisissez Mettre immédiatement le secret en rotation.
4. Dans Rotation de secret, choisissez Mettre en rotation.

AWS CLI

Example Mettre immédiatement un secret en rotation

L'exemple suivant rotate-secret lance une rotation immédiate. La rotation du secret doit déjà être configurée à l'heure actuelle.

```
$ aws secretsmanager rotate-secret \
```

```
    --secret-id MyTestSecret
```

Trouvez des secrets qui ne sont pas alternés

Vous pouvez l'utiliser AWS Config pour évaluer vos secrets afin de voir s'ils sont renouvelés conformément à vos normes. Vous définissez vos exigences internes en matière de sécurité et de conformité pour les secrets à l'aide de AWS Config règles. AWS Config Vous pouvez ensuite identifier les secrets qui ne sont pas conformes à vos règles. Vous pouvez également suivre les modifications apportées aux métadonnées des secrets, à la configuration de la rotation, à la clé KMS utilisée pour le chiffrement des secrets, à la fonction de rotation Lambda et aux identifications associées à un secret.

Si vous avez des secrets dans plusieurs entités Comptes AWS et Régions AWS au sein de votre organisation, vous pouvez agréger ces données de configuration et de conformité. Pour plus d'informations, consultez la section Agrégation de données multicomptes et multirégions.

Pour déterminer si les secrets changent

1. Suivez les instructions relatives à l'évaluation de vos ressources à l'aide de AWS Config règles et choisissez l'une des règles suivantes :

   - `secretsmanager-rotation-enabled-check` : vérifie si la rotation est configurée pour les secrets stockés dans Secrets Manager.
   - `secretsmanager-scheduled-rotation-success-check` : vérifie si la dernière rotation réussie se situe dans les limites de la fréquence de rotation configurée. La fréquence minimale de contrôle se fait de façon quotidienne.
   - `secretsmanager-secret-periodic-rotation` : vérifie si les secrets ont fait l'objet d'une rotation dans le nombre de jours spécifié.
2. Configurez éventuellement AWS Config pour vous avertir lorsque les secrets ne sont pas conformes. Pour plus d'informations, consultez la rubrique Notifications AWS Config envoyées à un Amazon SNS.

Annuler la rotation automatique dans Secrets Manager

Si vous avez configuré la rotation automatique pour un secret et que vous souhaitez arrêter de le faire pivoter, vous pouvez annuler la rotation.

Pour annuler la rotation automatique

1. Ouvrez la console Secrets Manager à l'adresse https://console.aws.amazon.com/secretsmanager/.
2. Choisissez votre secret.
3. Sur la page des détails secrets, sous Configuration de la rotation, choisissez Modifier la rotation.
4. Dans la boîte de dialogue Modifier la configuration de rotation, désactivez la rotation automatique, puis choisissez Enregistrer.

   Secrets Manager conserve les informations de configuration de rotation afin que vous puissiez les utiliser à l'avenir si vous décidez de réactiver la rotation.

AWS Secrets Manager secrets gérés par d'autres AWS services

De nombreux AWS services stockent et utilisent des secrets dans AWS Secrets Manager. Dans certains cas, ces secrets sont des secrets gérés, ce qui signifie que le service qui les a créés aide à les gérer. Par exemple, certains secrets gérés incluent la rotation gérée, de sorte que vous n'avez pas à configurer la rotation vous-même. Le service de gestion peut également vous empêcher de mettre à jour des secrets ou de les supprimer sans période de récupération, ce qui permet d'éviter des pannes car le service de gestion dépend du secret.

> **Note**
>
> Les secrets gérés ne peuvent être créés que par le AWS service qui les gère.

Les secrets gérés utilisent une convention de dénomination qui inclut l'ID du service de gestion pour aider à les identifier.

```
Secret name: ServiceID!MySecret
Secret ARN : arn:aws:us-east-1:ServiceID!MySecret-a1b2c3
```

IDs pour les services qui gèrent les secrets

- `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 Service de calcul parallèle”
- `rds` – the section called “Amazon RDS”
- `redshift` – the section called “Amazon Redshift”

- `sqlworkbench` – the section called "Éditeur de requête Amazon Redshift v2"

Pour rechercher des secrets gérés par d'autres AWS services, voir Rechercher des secrets gérés.

Pour obtenir la liste complète des services utilisant des secrets, consultezServices utilisant des secrets.

Services AWS qui utilisent des AWS Secrets Manager secrets

Découvrez comment chacun des éléments suivants s' Services AWS intègre à Secrets Manager.

- Comment AWS App Runner utilise AWS Secrets Manager
- Comment AWS App2Container utilise AWS Secrets Manager
- Comment AWS AppConfig utilise AWS Secrets Manager
- Comment Amazon AppFlow utilise AWS Secrets Manager
- Comment AWS AppSync utilise AWS Secrets Manager
- Comment Amazon Athena utilise AWS Secrets Manager
- Comment Amazon Aurora utilise AWS Secrets Manager
- Comment AWS CodeBuild utilise AWS Secrets Manager
- Comment Amazon Data Firehose utilise AWS Secrets Manager
- Comment AWS DataSync utilise AWS Secrets Manager
- Comment Amazon DataZone utilise AWS Secrets Manager
- Comment AWS Direct Connect utilise AWS Secrets Manager
- Comment AWS Directory Service utilise AWS Secrets Manager
- Comment Amazon DocumentDB (avec compatibilité MongoDB) utilise AWS Secrets Manager
- Comment AWS Elastic Beanstalk utilise AWS Secrets Manager
- Comment Amazon Elastic Container Registry utilise AWS Secrets Manager
- Amazon Elastic Container Service
- Comment Amazon ElastiCache utilise AWS Secrets Manager
- Comment AWS Elemental Live utilise AWS Secrets Manager
- Comment AWS Elemental MediaConnect utilise AWS Secrets Manager
- Comment AWS Elemental MediaConvert utilise AWS Secrets Manager
- Comment AWS Elemental MediaLive utilise AWS Secrets Manager
- Comment AWS Elemental MediaPackage utilise AWS Secrets Manager
- Comment AWS Elemental MediaTailor utilise AWS Secrets Manager
- Comment Amazon EMR utilise Secrets Manager

- Comment Amazon EventBridge utilise AWS Secrets Manager
- Comment Amazon FSx utilise les AWS Secrets Manager secrets
- Comment AWS Glue DataBrew utilise AWS Secrets Manager
- Comment AWS Glue Studio utilise AWS Secrets Manager
- Comment AWS IoT SiteWise utilise AWS Secrets Manager
- Comment Amazon Kendra utilise AWS Secrets Manager
- Comment Amazon Kinesis Video Streams utilise AWS Secrets Manager
- Comment AWS Launch Wizard utilise AWS Secrets Manager
- Comment Amazon Lookout for Metrics utilise AWS Secrets Manager
- Comment Amazon Managed Grafana utilise AWS Secrets Manager
- Comment AWS Managed Services utilise AWS Secrets Manager
- Comment Amazon Managed Streaming for Apache Kafka utilise AWS Secrets Manager
- Comment Amazon Managed Workflows pour Apache Airflow utilise AWS Secrets Manager
- AWS Marketplace
- Comment AWS Migration Hub utilise AWS Secrets Manager
- Comment AWS Panorama utilise Secrets Manager
- Comment le AWS Parallel Computing Service utilise AWS Secrets Manager
- Comment AWS ParallelCluster utilise AWS Secrets Manager
- Comment Amazon Q utilise Secrets Manager
- Comment Amazon OpenSearch Ingestion utilise Secrets Manager
- Comment AWS OpsWorks for Chef Automate utilise AWS Secrets Manager
- Comment Amazon QuickSight utilise AWS Secrets Manager
- Comment Amazon RDS utilise AWS Secrets Manager
- Comment Amazon Redshift utilise AWS Secrets Manager
- Éditeur de requête Amazon Redshift v2
- Comment Amazon SageMaker AI utilise AWS Secrets Manager
- Comment AWS Schema Conversion Tool utilise AWS Secrets Manager
- Comment utilise Amazon Timestream pour InfluxDB AWS Secrets Manager
- Comment AWS Toolkit for JetBrains utilise AWS Secrets Manager

- Comment AWS Transfer Family utilise les AWS Secrets Manager secrets
- Comment AWS Wickr utilise les AWS Secrets Manager secrets

Comment AWS App Runner utilise AWS Secrets Manager

AWS App Runner est un AWS service qui fournit un moyen rapide, simple et économique de déployer à partir du code source ou d'une image de conteneur directement vers une application Web évolutive et sécurisée dans le AWS cloud. Vous n'avez pas besoin d'apprendre de nouvelles technologies, de choisir le service informatique à utiliser ou de savoir comment approvisionner et configurer les AWS ressources.

Avec App Runner, vous pouvez référencer des secrets et des configurations en tant que variables d'environnement dans votre service lorsque vous créez un service ou mettez à jour la configuration du service. Pour plus d'informations, consultez les sections Référencement des variables d'environnement (français non garanti) et Gestion des variables d'environnement (français non garanti) dans le AWS App Runner Guide du développeur.

Comment AWS App2Container utilise AWS Secrets Manager

AWS App2Container est un outil de ligne de commande qui vous aide à transférer et à déplacer les applications qui s'exécutent dans vos centres de données locaux ou sur des machines virtuelles, afin qu'elles s'exécutent dans des conteneurs gérés par Amazon ECS, Amazon EKS ou AWS App Runner.

App2Container utilise Secrets Manager pour gérer les informations d'identification qui permettent de connecter votre machine de travail à des serveurs d'applications, afin d'exécuter des commandes à distance. Pour plus d'informations, consultez la section Gérer les secrets pour AWS App2Container dans le guide de l'utilisateur d'AWS App2Container.

Comment AWS AppConfig utilise AWS Secrets Manager

AWS AppConfig est une fonctionnalité AWS Systems Manager que vous pouvez utiliser pour créer, gérer et déployer rapidement des configurations d'applications. Une configuration peut contenir des données d'informations d'identification ou d'autres informations sensibles stockées dans Secrets Manager. Lorsque vous créez un profil de configuration libre, vous pouvez choisir Secrets Manager en tant que source de vos données de configuration. Pour plus d'informations, consultez la section Création d'un profil de configuration libre dans le Guide de l'utilisateur AWS AppConfig . Pour plus

d'informations sur la AWS AppConfig gestion des secrets dont la rotation automatique est activée, consultez la section Rotation des clés de Secrets Manager dans le guide de AWS AppConfig l'utilisateur.

Comment Amazon AppFlow utilise AWS Secrets Manager

Amazon AppFlow est un service d'intégration entièrement géré qui vous permet d'échanger des données en toute sécurité entre des applications SaaS (Software as a Service), telles que Salesforce Services AWS, et Amazon Simple Storage Service (Amazon S3) et Amazon Redshift.

Dans Amazon AppFlow, lorsque vous configurez une application SaaS en tant que source ou destination, vous créez une connexion. Cela inclut les informations requises pour une connection aux applications SaaS, telles que les jetons d'authentification, les noms d'utilisateur et les mots de passe. Amazon AppFlow stocke vos données de connexion dans un secret géré par Secrets Manager avec le préfixe`appflow`. Le coût de stockage du secret est inclus dans les frais d'Amazon AppFlow. Pour plus d'informations, consultez la section Protection des données sur Amazon AppFlow dans le guide de AppFlow l'utilisateur Amazon.

Comment AWS AppSync utilise AWS Secrets Manager

AWS AppSync fournit une interface GraphQL robuste et évolutive permettant aux développeurs d'applications de combiner des données provenant de plusieurs sources, notamment Amazon DynamoDB AWS Lambda et HTTP. APIs

AWS AppSync utilise les informations d'identification contenues dans un secret de Secrets Manager pour se connecter à Amazon RDS et Aurora. Pour plus d'informations, consultez le Tutoriel : Aurora sans serveur (français non garanti) dans le Guide du développeur AWS AppSync .

Comment Amazon Athena utilise AWS Secrets Manager

Amazon Athena est un service de requêtes interactif qui facilite l'analyse de données directe dans Amazon Simple Storage Service (Amazon S3) via la syntaxe SQL standard.

Les connecteurs de source de données Amazon Athena peuvent utiliser la fonction de requête fédérée Athena avec des secrets de Secrets Manager pour réaliser des requêtes de données. Pour de plus amples d'informations, consultez la section Utilisation de la requête fédérée Amazon Athena dans le Guide de l'utilisateur Amazon Athena.

Comment Amazon Aurora utilise AWS Secrets Manager

Amazon Aurora est un moteur de base de données relationnelle entièrement géré compatible avec MySQL et PostgreSQL.

Pour gérer les informations d'identification de l'utilisateur principal pour Aurora, Aurora peut créer un secret géré pour vous. Ce secret vous est facturé. Aurora gère également la rotation de ces informations d'identification. Pour plus d'informations, consultez Gestion d'un cluster de base de données Amazon Aurora AWS Secrets Manager (français non garanti) dans le Manuel de l'utilisateur Amazon Aurora (français non garanti).

Pour d'autres informations d'identification Aurora, consultezCréation de secrets.

Lorsque vous appelez l'API de données Amazon RDS, vous pouvez transmettre des informations d'identification pour la base de données en utilisant un secret dans Secrets Manager. Pour de plus amples informations, veuillez consulter Utilisation de l'API Data pour Aurora Serverless dans le Guide de l'utilisateur Amazon Aurora.

Lorsque vous utilisez l'éditeur de requêtes Amazon RDS pour vous connecter à une base de données, vous pouvez stocker les informations d'identification de la base de données dans Secrets Manager. Pour plus d'informations, consultez Utilisation de l'éditeur de requête dans le Guide de l'utilisateur Amazon RDS.

Comment AWS CodeBuild utilise AWS Secrets Manager

AWS CodeBuild est un service de création entièrement géré dans le cloud. CodeBuild compile votre code source, exécute des tests unitaires et produit des artefacts prêts à être déployés.

Vous pouvez stocker vos informations d'identification de registre privé à l'aide de Secrets Manager. Pour plus d'informations, voir Registre privé avec AWS Secrets Manager exemple CodeBuild dans le Guide de AWS CodeBuild l'utilisateur.

Comment Amazon Data Firehose utilise AWS Secrets Manager

Vous pouvez utiliser Amazon Data Firehose pour diffuser des données de streaming en temps réel vers différentes destinations de streaming. Lorsque la destination nécessite des informations d'identification ou une clé, Firehose récupère un secret auprès de Secrets Manager lors de l'exécution pour se connecter à la destination. Pour plus d'informations, consultez Authenticate with

AWS Secrets Manager dans Amazon Data Firehose dans le manuel du développeur Amazon Data Firehose.

Comment AWS DataSync utilise AWS Secrets Manager

AWS DataSync est un service de transfert de données en ligne qui simplifie, automatise et accélère le transfert des données entre les systèmes et les services de stockage.

Certains des systèmes de stockage pris en charge par DataSync nécessitent des informations d'identification pour lire et écrire des données. DataSync utilise Secrets Manager pour stocker ou accéder aux informations d'identification de stockage. Vous pouvez configurer DataSync pour créer des secrets en votre nom ou vous pouvez fournir un secret personnalisé. Les secrets gérés par le service commencent par le préfixe. `aws-datasync` Vous n'êtes facturé que pour l'utilisation de secrets que vous créez en dehors de DataSync. Consultez la section Fourniture d'informations d'identification pour les emplacements de stockage dans le guide de AWS DataSync l'utilisateur.

Comment Amazon DataZone utilise AWS Secrets Manager

Amazon DataZone est un service de gestion des données qui vous permet de cataloguer, de découvrir, de gérer, de partager et d'analyser vos données. Vous pouvez utiliser les actifs de données issus de tables et de vues d'un cluster Amazon Redshift analysé à l'aide d'une tâche. AWS Glue crawler Pour vous connecter à Amazon Redshift, vous devez fournir des DataZone informations d'identification Amazon dans un secret Secrets Manager. Pour plus d'informations, consultez la section Créer une source de données pour une base de données Amazon Redshift à l'aide d'une nouvelle AWS Glue connexion dans le guide de DataZone l'utilisateur Amazon.

Comment AWS Direct Connect utilise AWS Secrets Manager

AWS Direct Connect relie votre réseau interne à un AWS Direct Connect emplacement via un câble à fibre optique Ethernet standard. Grâce à cette connexion, vous pouvez créer des interfaces virtuelles directement destinées au public Services AWS.

AWS Direct Connect stocke un nom de clé d'association de connectivité et une paire de clés d'association de connectivité (paire CKN/CAK) dans un secret géré avec le préfixe. `directconnect` Le coût du secret est inclus dans les frais pour AWS Direct Connect. Pour mettre à jour le secret, vous devez utiliser AWS Direct Connect plutôt que Secrets Manager. Pour plus d'informations, voir Associer un MACsec CKN/CAK à un LAG dans le guide de l'AWS Direct Connect utilisateur.

Comment AWS Directory Service utilise AWS Secrets Manager

AWS Directory Service propose plusieurs manières d'utiliser Microsoft Active Directory (AD) avec d'autres AWS services. Vous pouvez joindre une EC2 instance Amazon à votre répertoire en utilisant des secrets comme informations d'identification. Pour plus d'informations, dans le Guide de l'utilisateur d'AWS Direct Connect , consultez :

- Joignez facilement une EC2 instance Linux à votre répertoire Microsoft AD AWS géré
- Associez facilement une EC2 instance Linux à votre répertoire AD Connector
- Associez facilement une EC2 instance Linux à votre répertoire Simple AD

Comment Amazon DocumentDB (avec compatibilité MongoDB) utilise AWS Secrets Manager

Amazon DocumentDB (compatible avec MongoDB) est un service de base de données de documents entièrement géré qui prend en charge les charges de travail MongoDB. Amazon DocumentDB s'intègre à Secrets Manager pour gérer les mots de passe des utilisateurs principaux de vos clusters, améliorer la sécurité et simplifier la gestion des informations d'identification.

Amazon DocumentDB génère le mot de passe, le stocke dans Secrets Manager et gère les paramètres secrets. Par défaut, Amazon DocumentDB fait pivoter le secret tous les sept jours, mais vous pouvez modifier le calendrier de rotation si nécessaire. Lorsque vous créez ou modifiez un cluster Amazon DocumentDB, vous pouvez spécifier qu'il doit gérer le mot de passe de l'utilisateur principal dans Secrets Manager. Pour plus d'informations, consultez la section Gestion des mots de passe avec Amazon DocumentDB et Secrets Manager dans le manuel Amazon DocumentDB Developer Guide.

Comment AWS Elastic Beanstalk utilise AWS Secrets Manager

Vous pouvez ainsi déployer et gérer rapidement des applications dans le AWS cloud sans avoir à vous renseigner sur l'infrastructure qui exécute ces applications. AWS Elastic Beanstalk Elastic Beanstalk peut lancer des environnements Docker en créant une image décrite dans un Dockerfile ou en extrayant une image Docker à distance. Pour s'authentifier auprès du registre en ligne qui héberge le référentiel privé, Elastic Beanstalk utilise un secret Secrets Manager. Pour plus d'informations, consultez la rubrique Configuration Docker du Guide du développeur AWS Elastic Beanstalk .

Comment Amazon Elastic Container Registry utilise AWS Secrets Manager

Amazon Elastic Container Registry (Amazon ECR) est AWS un service géré de registre d'images de conteneurs sécurisé, évolutif et fiable. Vous pouvez utiliser la CLI Docker pour transmettre et extraire les images des référentiels. Pour chaque registre en amont contenant des images que vous souhaitez mettre en cache dans votre registre privé Amazon ECR, vous devez créer une règle de mise en par extraction. Pour de registres en amont qui nécessitent une authentification, vous devez stocker les informations d'identification dans un secret Secrets Manager. Vous pouvez créer le secret Secrets Manager dans les consoles Amazon ECR ou Secrets Manager. Pour plus d'informations, consultez la section Création d'une règle de cache d'extraction dans le guide de l'utilisateur Amazon ECR.

Amazon Elastic Container Service

Amazon Elastic Container Service (Amazon ECS) est un service d'orchestration de conteneurs entièrement géré qui vous permet de déployer, de gérer et de dimensionner aisément des applications conteneurisées. Vous pouvez injecter des données sensibles dans vos conteneurs en faisant référence aux secrets de Secrets Manager. Pour plus d'informations, consultez les pages suivantes dans le Guide du développeur Amazon Elastic Container Service :

- Didacticiel : spécification des données sensibles à l'aide des secrets de Secrets Manager
- Récupérer des secrets de manière programmatique par le biais de votre application
- Récupérer des secrets par le biais de variables d'environnement
- Récupérer les secrets pour la configuration de la journalisation

Amazon ECS prend en charge FSx les volumes de serveur de fichiers Windows pour les conteneurs. Amazon ECS utilise les informations d'identification stockées dans un secret de Secrets Manager pour rejoindre le domaine Active Directory et joindre le FSx système de fichiers Windows File Server. Pour plus d'informations, consultez Tutoriel : Utilisation FSx pour les systèmes de fichiers Windows File Server avec Amazon ECS et FSx pour les volumes Windows File Server dans le manuel Amazon Elastic Container Service Developer Guide.

Vous pouvez référencer des images de conteneurs dans des registres privés autres AWS que ceux qui nécessitent une authentification en utilisant un secret Secrets Manager avec les informations

d'identification du registre. Pour plus d'informations, consultez la section Authentification du registre privé pour les tâches dans le Guide du développeur Amazon Elastic Container Service.

Lorsque vous utilisez Amazon ECS Service Connect, Amazon ECS utilise les secrets gérés par Secrets Manager pour stocker AWS Private Certificate Authority les certificats TLS. Le coût du stockage du secret est inclus dans les frais d'Amazon ECS. Pour mettre à jour le secret, vous devez utiliser Amazon ECS plutôt que Secrets Manager. Pour plus d'informations, consultez TLS with Service Connect dans le manuel Amazon Elastic Container Service Developer Guide.

Comment Amazon ElastiCache utilise AWS Secrets Manager

ElastiCache Vous pouvez utiliser une fonctionnalité appelée contrôle d'accès basé sur les rôles (RBAC) pour sécuriser le cluster. Vous pouvez stocker ces informations d'identification dans Secrets Manager. Le gestionnaire de secrets fournit un modèle de rotation pour ce type de secret. Pour plus d'informations, consultez Rotation automatique des mots de passe pour les utilisateurs dans le guide de ElastiCache l'utilisateur Amazon.

Comment AWS Elemental Live utilise AWS Secrets Manager

AWS Elemental Live est un service vidéo en temps réel qui vous permet de créer des sorties en direct pour la diffusion et la diffusion en continu.

AWS Elemental Live utilise un ARN secret pour obtenir un secret contenant une clé de chiffrement auprès de Secrets Manager. Elemental Live utilise la clé de cryptage de encrypt/decrypt la vidéo. Pour plus d'informations, consultez la section Fonctionnement de la livraison de AWS Elemental Live à MediaConnect lors de l'exécution dans le guide de l'utilisateur d'Elemental Live.

Comment AWS Elemental MediaConnect utilise AWS Secrets Manager

AWS Elemental MediaConnect est un service qui permet aux diffuseurs et autres fournisseurs de vidéos haut de gamme d'ingérer de manière fiable des vidéos en direct dans le AWS Cloud et de les distribuer vers plusieurs destinations, à l'intérieur ou à l'extérieur du AWS Cloud.

Vous pouvez utiliser le chiffrement à clé statique pour protéger vos sources, sorties et droits, et vous stockez votre clé de chiffrement dans AWS Secrets Manager. Pour plus d'informations, consultez la

section Chiffrement par clé statique AWS Elemental MediaConnect dans le guide de AWS Elemental MediaConnect l'utilisateur.

Comment AWS Elemental MediaConvert utilise AWS Secrets Manager

AWS Elemental MediaConvert est un service de traitement vidéo basé sur des fichiers qui fournit un traitement vidéo évolutif aux propriétaires de contenu et aux distributeurs disposant de bibliothèques multimédias de toutes tailles. Pour encoder MediaConvert les filigranes Kantar, vous devez utiliser Secrets Manager pour stocker vos informations d'identification Kantar. Pour plus d'informations, consultez la section Utilisation de Kantar pour le filigrane audio dans les AWS Elemental MediaConvert sorties dans le guide de l'AWS Elemental MediaConvert utilisateur.

Comment AWS Elemental MediaLive utilise AWS Secrets Manager

AWS Elemental MediaLive est un service vidéo en temps réel qui vous permet de créer des sorties en direct pour la diffusion et la diffusion en continu. Si votre entreprise utilise des AWS Elemental Link appareils avec AWS Elemental MediaLive ou AWS Elemental MediaConnect, vous devez déployer l'appareil et le configurer. Pour plus d'informations, consultez la section Configuration en MediaLive tant qu'entité de confiance dans le Guide de MediaLive l'utilisateur.

Comment AWS Elemental MediaPackage utilise AWS Secrets Manager

AWS Elemental MediaPackage est un service de packaging et de création de just-in-time vidéos qui fonctionne dans le AWS Cloud. Vous pouvez ainsi diffuser des flux vidéo hautement sécurisés, évolutifs et fiables vers une grande variété d'appareils de lecture et de réseaux de diffusion de contenu (CDNs). MediaPackage Pour plus d'informations, consultez la section Accès à Secrets Manager pour l'autorisation du CDN dans le Guide de l'AWS Elemental MediaPackage utilisateur.

Comment AWS Elemental MediaTailor utilise AWS Secrets Manager

AWS Elemental MediaTailor est un service évolutif d'insertion de publicités et d'assemblage de chaînes qui fonctionne dans le AWS Cloud.

MediaTailor prend en charge l'authentification par jeton d'accès de Secrets Manager à vos emplacements sources. L'authentification par jeton d'accès Secrets Manager MediaTailor utilise un secret Secrets Manager pour authentifier les demandes adressées à votre origine. Pour plus d'informations, consultez la section Configuration de AWS Secrets Manager l'authentification par jeton d'accès dans le Guide de AWS Elemental MediaTailor l'utilisateur.

Comment Amazon EMR utilise Secrets Manager

Amazon EMR est une plateforme qui simplifie l'exécution de frameworks de mégadonnées, tels qu'Apache Hadoop et Apache Spark, AWS pour traiter et analyser de grandes quantités de données. Lorsque vous utilisez ces infrastructures et des projets open source connexes, tels qu'Apache Hive et Apache Pig, vous pouvez traiter des données pour analyse et pour des charges de travail business intelligence. Vous pouvez également utiliser Amazon EMR pour transformer et déplacer de grandes quantités de données vers et depuis d'autres magasins de données et bases de AWS données, tels qu'Amazon S3 et Amazon DynamoDB.

Comment Amazon EMR exécuté sur Amazon EC2 utilise Secrets Manager

Lorsque vous créez un cluster dans Amazon EMR, vous pouvez fournir des données de configuration de l'application au cluster avec un secret dans Secrets Manager. Pour de plus amples informations, veuillez consulter Stockage de données de configuration sensible dans Secrets Manager dans le guide de gestion Amazon EMR.

De plus, lorsque vous créez un bloc-note EMR, vous pouvez stocker vos informations d'identification de registre privé basés sur GIT à l'aide de Secrets Manager. Pour plus d'information, consultez la section Ajout d'un référentiel basé sur GIT à Amazon EMR (français non garanti) dans le Guide de gestion Amazon EMR.

Comment EMR sans serveur utilise Secrets Manager

EMR sans serveur fournit un environnement d'exécution sans serveur pour simplifier le fonctionnement des applications d'analyse afin que vous n'ayez pas à configurer, optimiser, sécuriser ou exploiter des clusters.

Vous pouvez stocker vos données AWS Secrets Manager puis utiliser l'identifiant secret dans vos configurations EMR Serverless. Ainsi, vous ne transmettez pas de données de configuration sensibles en texte brut et ne les exposez pas à des données externes APIs.

Pour plus d'informations, consultez Secrets Manager pour la protection des données avec EMR sans serveur dans le Guide de l'utilisateur d'Amazon EMR sans serveur.

Comment Amazon EventBridge utilise AWS Secrets Manager

Amazon EventBridge est un service de bus d'événements sans serveur que vous pouvez utiliser pour connecter vos applications à des données provenant de diverses sources.

Lorsque vous créez une destination d' EventBridge API Amazon, la EventBridge connexion correspondante est enregistrée dans un secret géré par Secrets Manager avec le préfixe`events`. Le coût de stockage du secret est inclus dans les frais d'utilisation d'une destination d'API. Pour mettre à jour le secret, vous devez utiliser EventBridge plutôt que Secrets Manager. Pour plus d'informations, consultez la section Destinations des API dans le guide de EventBridge l'utilisateur Amazon.

Comment Amazon FSx utilise les AWS Secrets Manager secrets

Amazon FSx pour Windows File Server fournit des serveurs de fichiers Microsoft Windows entièrement gérés, soutenus par un système de fichiers Windows entièrement natif. Lorsque vous créez ou gérez des partages de fichiers, vous pouvez transmettre des informations d'identification à partir d'un AWS Secrets Manager secret. Pour plus d'informations, consultez les sections Partage de fichiers et Migration des configurations de partage de fichiers vers Amazon FSx dans le guide de l'utilisateur du serveur de fichiers Amazon FSx pour Windows.

Comment AWS Glue DataBrew utilise AWS Secrets Manager

AWS Glue DataBrew est un outil visuel de préparation des données que vous pouvez utiliser pour nettoyer et normaliser les données sans écrire de code. Dans DataBrew, un ensemble d'étapes de transformation de données est appelé recette. AWS Glue DataBrew fournit les `DETERMINISTIC_DECRYPT`étapes et `DETERMINISTIC_ENCRYPT`les `CRYPTOGRAPHIC_HASH`recettes pour effectuer des transformations sur les informations personnelles identifiables (PII) d'un ensemble de données, qui utilisent une clé de chiffrement stockée dans un secret du Secrets Manager. Si vous utilisez le secret DataBrew par défaut pour stocker la clé de chiffrement, DataBrew crée un secret géré avec le préfixe`databrew`. Le coût de conservation du secret est inclus dans les frais d'utilisation DataBrew. Si vous créez un nouveau secret pour stocker la clé de chiffrement, DataBrew crée un secret avec le préfixe`AwsGlueDataBrew`. Ce secret vous est facturé.

Comment AWS Glue Studio utilise AWS Secrets Manager

AWS Glue Studio est une interface graphique qui facilite la création, l'exécution et le suivi de tâches d'extraction, de transformation et de chargement (ETL) dans AWS Glue. Vous pouvez utiliser Amazon OpenSearch Service comme magasin de données pour vos tâches d'extraction, de transformation et de chargement (ETL) en configurant le connecteur Elasticsearch Spark dans. AWS Glue Studio Pour vous connecter au OpenSearch cluster, vous pouvez utiliser un secret dans Secrets Manager. Pour plus d'informations, consultez Tutoriel : Utilisation du connecteur AWS Glue pour Elasticsearch dans le manuel du AWS Glue développeur.

Comment AWS IoT SiteWise utilise AWS Secrets Manager

AWS IoT SiteWise est un service géré qui vous permet de collecter, de modéliser, d'analyser et de visualiser des données provenant d'équipements industriels à grande échelle. Vous pouvez utiliser la AWS IoT SiteWise console pour créer une passerelle. Ajoutez ensuite des sources de données, des serveurs locaux ou des équipements industriels qui sont connectés à des passerelles. Si votre source requiert une authentification, utilisez un secret pour vous authentifier. Pour plus d'informations, consultez la section Configuration de l'authentification de la source de données (français non granti) dans le Guide de l'utilisateur AWS IoT SiteWise .

Comment Amazon Kendra utilise AWS Secrets Manager

Amazon Kendra est un service de recherche hautement précis et intelligent qui permet à vos utilisateurs de rechercher des données non structurées et structurées à l'aide du traitement du langage naturel et d'algorithmes de recherche avancée.

Vous pouvez indexer les documents stockés dans une base de données en spécifiant un secret contenant des informations d'identification pour la base de données. Pour plus d'informations, consultez Utilisation d'une source de données de base de données dans le Guide de l'utilisateur Amazon Kendra.

Comment Amazon Kinesis Video Streams utilise AWS Secrets Manager

Vous pouvez utiliser Amazon Kinesis Video Streams pour vous connecter à des caméras IP dans les locaux du client, enregistrer et stocker localement des vidéos provenant des caméras, et diffuser des

vidéos vers le cloud à des fins de stockage, de lecture et de traitement analytique à long terme. Pour enregistrer et charger du contenu multimédia à partir de caméras IP, déployez l'agent Kinesis Video Streams Edge sur AWS IoT Greengrass. Vous stockez les informations d'identification requises pour accéder aux fichiers multimédia qui sont diffusés vers la caméra dans un secret Secrets Manager. Pour plus d'informations, consultez la section Déployez l'agent Amazon Kinesis Video Streams Edge sur AWS IoT Greengrass du Manuel du développeur Amazon Kinesis Video Streams.

Comment AWS Launch Wizard utilise AWS Secrets Manager

AWS Launch Wizard for Active Directory est un service qui applique les meilleures pratiques en matière d' AWS Cloud applications pour vous aider à configurer une nouvelle infrastructure Active Directory ou à ajouter des contrôleurs de domaine à une infrastructure existante, que ce soit sur site AWS Cloud ou sur site.

AWS Launch Wizard nécessite que les informations d'identification de l'administrateur de domaine soient ajoutées à Secrets Manager pour joindre vos contrôleurs de domaine à Active Directory. Pour plus d'informations, voir Configuration AWS Launch Wizard pour Active Directory dans le Guide de l'AWS Launch Wizard utilisateur.

Comment Amazon Lookout for Metrics utilise AWS Secrets Manager

Amazon Lookout for Metrics est un service qui détecte des anomalies dans vos données, détermine leurs causes d'origine et vous permet de prendre rapidement des mesures. Vous pouvez utiliser Amazon Redshift ou Amazon RDS comme source de données pour un détecteur Lookout for Metrics. Pour configurer la source de données, vous utilisez un secret contenant le mot de passe de la base de données. Pour plus d'informations, consultez les sections Utiliser Amazon RDS avec Lookout for Metrics (français non garanti) et Utiliser Amazon Redshift avec Lookout for Metrics (français non garanti) dans le Guide du développeur Amazon Lookout for Metrics (français non garanti).

Comment Amazon Managed Grafana utilise AWS Secrets Manager

Amazon Managed Grafana est un service de visualisation de données entièrement géré et sécurisé que vous pouvez utiliser pour interroger, corréler et visualiser instantanément des métriques

opérationnelles, des journaux et des suivis provenant de plusieurs sources. Lorsque vous utilisez Amazon Redshift comme source de données, vous pouvez fournir des informations d'identification Amazon Redshift à l'aide d'un secret AWS Secrets Manager . Pour plus d'informations, veuillez consulter la rubrique Configuration d'Amazon Redshift dans le Guide de l'utilisateur Amazon Managed Grafana.

Comment AWS Managed Services utilise AWS Secrets Manager

AWS Managed Services est un service d'entreprise qui assure la gestion continue de votre AWS infrastructure. Le mode AMS Self-Service Provisioning (SSP) fournit un accès complet aux fonctionnalités natives Service AWS et API des comptes gérés par AMS. Pour plus d'informations sur la procédure à suivre pour demander l'accès à Secrets Manager dans AMS, consultez la section AWS Secrets Manager (Approvisionnement en libre-service d'AMS) dans le Guide de l'utilisateur avancé d'AMS.

Comment Amazon Managed Streaming for Apache Kafka utilise AWS Secrets Manager

Amazon Managed Streaming for Apache Kafka (Amazon MSK) est un service entièrement géré qui vous permet de créer et d'exécuter des applications utilisant Apache Kafka pour traiter des données de diffusion. Vous pouvez contrôler l'accès à vos clusters Amazon MSK à l'aide de noms d'utilisateur et mots de passe stockés et sécurisés à l'aide d' AWS Secrets Manager. Pour plus d'informations, consultez Authentification par nom d'utilisateur et mot de passe avec AWS Secrets Managerdans le Guide du développeur Amazon Managed Streaming for Apache Kafka.

Comment Amazon Managed Workflows pour Apache Airflow utilise AWS Secrets Manager

Amazon Managed Workflows for Apache Airflow est un service d'orchestration géré pour Apache Airflow qui facilite la configuration et l'exploitation de pipelines de end-to-end données dans le cloud à grande échelle.

Vous pouvez configurer une connexion Apache Airflow à l'aide d'un secret de Secrets Manager. Pour plus d'informations, consultez Configuration d'une connexion Apache Airflow à l'aide d'un secret Secrets Manager et Utilisation d'une clé secrète AWS Secrets Manager pour une variable Apache Airflow dans le guide de l'utilisateur Amazon Managed Workflows for Apache Airflow.

AWS Marketplace

Lorsque vous utilisez AWS Marketplace Quick Launch, AWS Marketplace distribue votre logiciel avec la clé de licence. AWS Marketplace stocke la clé de licence dans votre compte en tant que secret géré par Secrets Manager. Le coût de conservation du secret est inclus dans les frais de AWS Marketplace. Pour mettre à jour le secret, vous devez utiliser AWS Marketplace plutôt que Secrets Manager. Pour plus d'informations, consultez Configuration Quick Launch dans le Guide du vendeur AWS Marketplace .

Comment AWS Migration Hub utilise AWS Secrets Manager

AWS Migration Hub fournit un emplacement unique pour suivre les tâches de migration à travers plusieurs AWS outils et solutions partenaires.

AWS Migration Hub Orchestrator simplifie et automatise la migration des serveurs et des applications d'entreprise vers. AWS Orchestrateur Migration Hub utilise un secret pour les informations de connexion à votre serveur source. Pour plus d'informations, consultez dans le Guide de l'utilisateur AWS Migration Hub Orchestrateur (français non-garanti) :

- Migrez NetWeaver les applications SAP vers AWS
- Réhébergez des applications sur Amazon EC2

Migration Hub Strategy Recommendations offre des recommandations de stratégies de migration et de modernisation de chemins de transformation viables pour vos applications. Les recommandations de stratégies peuvent analyser des bases de données SQL Server en utilisant un secret pour les informations de connexion. Pour plus d'informations, consultez Analyse de la base de données des recommandations de stratégies.

Comment AWS Panorama utilise Secrets Manager

AWS Panorama est un service qui intègre la vision par ordinateur à votre réseau de caméras sur site. Vous l'utilisez AWS Panorama pour enregistrer une appliance, mettre à jour son logiciel et y déployer des applications. Lorsque vous enregistrez un flux vidéo en tant que source de données pour votre application, si le flux est protégé par un mot de passe, AWS Panorama stocke ses informations d'identification dans un secret du Gestionnaire de Secrets. Pour plus d'informations, consultez la section Gestion des flux de données des caméras dans AWS Panorama dans le Guide du développeur AWS Panorama .

Comment le AWS Parallel Computing Service utilise AWS Secrets Manager

AWS Le Parallel Computing Service (AWS PCS) est un service géré qui facilite l'exécution et le dimensionnement des charges de travail liées au calcul haute performance (HPC) et à l'apprentissage automatique distribué. AWS

Pour se connecter au planificateur de tâches du cluster, AWS PCS crée un secret géré avec le préfixe `pcs` pour stocker la clé du planificateur. Le coût de stockage du secret est inclus dans le prix du AWS PCS. AWS PCS supprime automatiquement le secret lorsque vous supprimez votre cluster AWS PCS. Pour plus d'informations, consultez la section Utilisation des secrets de cluster dans AWS PCS dans le Guide de l'utilisateur de AWS PCS.

> **Important**
>
> Ne modifiez ni ne supprimez les secrets du cluster AWS PCS.

Comment AWS ParallelCluster utilise AWS Secrets Manager

AWS ParallelCluster est un outil de gestion de clusters open source que vous pouvez utiliser pour déployer et gérer des clusters de calcul haute performance (HPC) dans le AWS Cloud. Vous pouvez créer un environnement multi-utilisateurs qui inclut ou AWS ParallelCluster qui est intégré à un Microsoft AD AWS géré (Active Directory). AWS ParallelCluster Utilise un secret Secrets Manager pour valider les connexions à Active Directory. Pour plus d'informations, consultez la section Intégration d'Active Directory (français non garanti) du Guide de l'utilisateur AWS ParallelCluster .

Comment Amazon Q utilise Secrets Manager

Pour authentifier Amazon Q afin d'accéder à votre source de données, vous fournissez vos informations d'accès à la source de données à Amazon Q à l'aide d'un secret Secrets Manager. Si vous utilisez la console, vous pouvez choisir de créer un nouveau secret ou d'utiliser un secret existant. Pour plus d'informations, consultez Concepts — Authentification dans le guide du développeur Amazon Q.

Comment Amazon OpenSearch Ingestion utilise Secrets Manager

Amazon OpenSearch Ingestionest un collecteur de données sans serveur entièrement géré qui diffuse des journaux, des métriques et des données de suivi en temps réel vers les domaines et les OpenSearch Serverless collections Amazon OpenSearch Service. Vous pouvez utiliser des OpenSearch Ingestion pipelines avec Secrets Manager pour gérer vos informations d'identification en toute sécurité. Pour plus d'informations, consultez :

- Utilisation d'un OpenSearch Ingestion pipeline avec Atlassian Services
- Utilisation d'un OpenSearch Ingestion pipeline avec Amazon DocumentDB
- Utilisation d'un OpenSearch Ingestion pipeline avec Confluent Cloud Kafka
- Utilisation d'un OpenSearch Ingestion pipeline avec Kafka
- Migration de données à partir de clusters autogérés OpenSearch à l'aide de Amazon OpenSearch Ingestion

Comment AWS OpsWorks for Chef Automate utilise AWS Secrets Manager

AWS OpsWorks est un service de gestion de configuration qui vous aide à configurer et à exploiter des applications dans une entreprise cloud en utilisant OpsWorks pour Puppet Enterprise ou AWS OpsWorks for Chef Automate.

Lorsque vous créez un nouveau serveur dans AWS OpsWorks CM, OpsWorks CM stocke les informations relatives au serveur dans un secret géré par Secrets Manager avec le préfixe`opsworks-cm`. Le coût du secret est inclus dans le prix de AWS OpsWorks. Pour plus d'informations, consultez Intégrer Intégration avec AWS Secrets Manager dans le Guide de l'utilisateur AWS OpsWorks .

Comment Amazon QuickSight utilise AWS Secrets Manager

Amazon QuickSight est un service de business intelligence (BI) à l'échelle du cloud que vous pouvez utiliser à des fins d'analyse, de visualisation des données et de reporting. Vous pouvez utiliser diverses sources de données dans QuickSight. Si vous stockez les informations d'identification de base de données dans les secrets de Secrets Manager, QuickSight vous pouvez utiliser ces secrets pour vous connecter aux bases de données. Pour plus d'informations, consultez la section Utilisation

de AWS Secrets Manager secrets à la place des informations d'identification de base de données QuickSight dans Amazon dans le guide de QuickSight l'utilisateur Amazon.

Comment Amazon RDS utilise AWS Secrets Manager

Amazon Relational Database Service (Amazon RDS) est un service web qui facilite la configuration, l'exploitation et la mise à l'échelle d'une base de données relationnelle dans le AWS Cloud.

Pour gérer les informations d'identification de l'utilisateur principal pour Amazon Relational Database Service (Amazon RDS), y compris Aurora, Amazon RDS peut créer un secret géré pour vous. Ce secret vous est facturé. Amazon RDS gère également la rotation pour ces informations d'identification. Pour plus informations, consultez Gestion des mots de passe avec Amazon RDS et AWS Secrets Manager (français non garanti) dans le Guide de l'utilisateur Amazon RDS (français non garanti).

Pour d'autres informations d'identification sur Amazon RDS, consultez Création de secrets.

Lorsque vous utilisez l'éditeur de requêtes Amazon RDS pour vous connecter à une base de données, vous pouvez stocker les informations d'identification de la base de données dans Secrets Manager. Pour plus d'informations, consultez Utilisation de l'éditeur de requête dans le Guide de l'utilisateur Amazon RDS.

Comment Amazon Redshift utilise AWS Secrets Manager

Amazon Redshift est un service d'entrepôt des données entièrement géré dans le cloud.

Pour gérer les informations d'identification d'administrateur pour Amazon Redshift, Amazon Redshift peut créer un secret géré pour vous. Ce secret vous est facturé. Amazon Redshift gère également la rotation de ces informations d'identification. Pour plus d'informations, consultez la section Gestion des mots de passe administrateur d'Amazon Redshift à l'aide d' AWS Secrets Manager (français non garanti) du Guide de gestion Amazon Redshift.

Pour les autres informations d'identification Amazon Redshift, consultez la section Création de secrets.

Lorsque vous appelez l'API de données, Amazon Redshift vous pouvez transmettre les informations d'identification pour le cluster en utilisant un secret dans Secrets Manager. Pour plus d'informations, consultez la section Utilisation de l'API de données Amazon Redshift.

Lorsque vous utilisez la fonction éditeur de requêtes Amazon Redshift pour vous connecter à une base de données, Amazon Redshift peut stocker vos informations d'identification dans un secret de Secrets Manager avec le préfixe `redshiftqueryeditor`. Ce secret vous est facturé. Pour plus d'informations, veuillez consulter la rubrique Interrogation d'une base de données à l'aide de l'éditeur de requête dans le Guide de gestion Amazon Redshift.

Pour l'éditeur de requête v2, consultez the section called "Éditeur de requête Amazon Redshift v2".

Éditeur de requête Amazon Redshift v2

L'éditeur de requêtes v2 Amazon Redshift est une application client SQL basée sur le web que vous pouvez utiliser pour créer et exécuter des requêtes sur votre entrepôt des données Amazon Redshift. Lorsque vous utilisez l'éditeur de requêtes Amazon Redshift v2 pour vous connecter à une base de données, Amazon Redshift peut stocker vos informations d'identification dans un secret géré par Secrets Manager avec le préfixe. `sqlworkbench` Le coût du stockage du secret est inclus dans les frais d'utilisation pour Amazon Redshift. Pour mettre à jour le secret, vous devez utiliser Amazon Redshift plutôt que Secrets Manager. Pour plus d'informations, veuillez consulter la rubrique Utilisation de l'éditeur de requête v2 dans le Guide de gestion Amazon Redshift.

Pour accéder à l'éditeur de requêtes précédent, consultez the section called "Amazon Redshift".

Comment Amazon SageMaker AI utilise AWS Secrets Manager

SageMaker L'IA est un service d'apprentissage automatique entièrement géré. Grâce à l' SageMaker IA, les data scientists et les développeurs peuvent rapidement et facilement créer et entraîner des modèles d'apprentissage automatique, puis les déployer directement dans un environnement hébergé prêt pour la production. Ce service fournit une instance de bloc-notes intégré à Jupyter pour accéder facilement à vos sources de données à des fins d'exploration et d'analyse. Ainsi, vous n'avez pas de serveur à gérer.

Vous pouvez associer des référentiels Git à vos instances de bloc-notes Jupyter pour enregistrer vos blocs-notes dans un environnement source de contrôle qui persiste même si vous arrêtez ou supprimez votre instance de bloc-notes. Vous pouvez gérer vos informations d'identification de référentiels privés à l'aide de Secrets Manager. Pour plus d'informations, consultez Associer des référentiels Git à des instances Amazon SageMaker Notebook dans le manuel Amazon SageMaker AI Developer Guide.

Pour importer des données depuis Databricks, Data Wrangler stocke votre URL JDBC dans Secrets Manager. Pour plus d'informations, consultez Importation des données depuis Databricks (JDBC).

Pour importer des données depuis Snowflake, Data Wrangler stocke vos informations d'identification dans un secret de Secrets Manager. Pour plus d'informations, consultez Importation des données depuis Snowflake.

Comment AWS Schema Conversion Tool utilise AWS Secrets Manager

Vous pouvez utiliser le AWS Schema Conversion Tool (AWS SCT) pour convertir votre schéma de base de données existant d'un moteur de base de données à un autre. Vous pouvez convertir le schéma OLTP relationnel ou le schéma d'entrepôt de données. Votre schéma converti convient à une base de données Amazon Relational Database Service (Amazon RDS) MySQL, MariaDB, Oracle, SQL Server, PostgreSQL, à un cluster de base de données Amazon Aurora ou à un cluster Amazon Redshift. Le schéma converti peut également être utilisé avec une base de données sur une instance Amazon Elastic Compute Cloud ou stocké sous forme de données dans un compartiment S3.

Lorsque vous convertissez un schéma de base de données, vous AWS SCT pouvez utiliser les informations d'identification de base de données que vous y stockez AWS Secrets Manager. Pour plus d'informations, consultez la section Utilisation AWS Secrets Manager dans l'interface AWS SCT utilisateur du Guide de AWS Schema Conversion Tool l'utilisateur.

Comment utilise Amazon Timestream pour InfluxDB AWS Secrets Manager

Timestream for InfluxDB est un moteur de base de données de séries chronologiques géré qui vous permet d'exécuter facilement des bases de données InfluxDB pour des applications de séries chronologiques en temps réel utilisant l'open source. AWS APIs Avec Timestream for InfluxDB, vous pouvez configurer, exploiter et dimensionner des charges de travail chronologiques capables de répondre à des requêtes avec un temps de réponse à un chiffre en millisecondes.

Lorsque vous créez une base de données Timestream pour InfluxDB, Timestream crée automatiquement un secret pour stocker les informations d'identification de l'administrateur. Pour plus d'informations, consultez Comment Timestream pour InfluxDB utilise les secrets dans le guide du développeur de Timestream.

Comment AWS Toolkit for JetBrains utilise AWS Secrets Manager

AWS Toolkit for JetBrains Il s'agit d'un plugin open source pour les environnements de développement intégrés (IDEs) de JetBrains. Cette boîte à outils facilite aux développeurs le développement, le débogage et le déploiement d'applications sans serveur qui utilisent AWS. Lorsque vous vous connectez à un cluster Amazon Redshift à l'aide de la boîte à outils, vous pouvez vous authentifier à l'aide d'un secret de Secrets Manager. Pour plus d'informations, consultez la section Accès aux clusters Amazon Redshift dans le Guide de l'utilisateur AWS Toolkit for JetBrains .

Comment AWS Transfer Family utilise les AWS Secrets Manager secrets

AWS Transfer Family est un service de transfert sécurisé qui vous permet de transférer des fichiers vers et depuis les services de AWS stockage.

Transfer Family prend désormais en charge l'utilisation de l'authentification de base pour les serveurs utilisant le protocole Applicability Statement 2 (AS2). Vous pouvez créer un nouveau secret de Secrets Manager ou choisir un secret existant pour vos informations d'identification. Pour plus d'informations, consultez la section Authentification de base pour les AS2 connecteurs dans le guide de AWS Transfer Family l'utilisateur.

Pour authentifier les utilisateurs de Transfer Family, vous pouvez l'utiliser AWS Secrets Manager en tant que fournisseur d'identité. Pour plus d'informations, consultez la section Travailler avec des fournisseurs d'identité personnalisés dans le guide de l'AWS Transfer Family utilisateur et dans l'article de blog Activer l'authentification par mot de passe pour AWS Transfer Family l'utilisation AWS Secrets Manager.

Vous pouvez utiliser le décryptage Pretty Good Privacy (PGP) avec les fichiers que Transfer Family traite avec des flux de travail. Pour utiliser le décryptage dans une étape du flux de travail, vous devez fournir une clé PGP que vous gérez dans Secrets Manager. Pour plus d'informations, consultez Générer et gérer des clés PGP dans le Guide de l'utilisateur AWS Transfer Family .

Comment AWS Wickr utilise les AWS Secrets Manager secrets

AWS Wickr est un service end-to-end crypté qui aide les organisations et les agences gouvernementales à communiquer en toute sécurité par le biais one-to-one de la messagerie de groupe, des appels vocaux et vidéo, du partage de fichiers, du partage d'écran, etc. Vous pouvez

automatiser les flux de travail à l'aide des robots de conservation des données Wickr. Si le bot y a accès Services AWS, vous devez créer un secret Secrets Manager pour stocker les informations d'identification du bot. Pour plus d'informations, voir Démarrer le bot de conservation des données dans le Guide d'AWS Wickr administration.

Créez des AWS Secrets Manager secrets dans AWS CloudFormation

Vous pouvez créer des secrets dans une CloudFormation pile en utilisant la `AWS::SecretsManager::Secret` ressource d'un CloudFormation modèle, comme indiqué dansCréer un secret.

Pour créer un secret d'administration pour Amazon RDS ou Aurora, nous vous recommandons d'utiliser `ManageMasterUserPassword` dans `AWS::RDS::DBCluster`. Ensuite, Amazon RDS crée le secret et gère la rotation pour vous. Pour de plus amples informations, veuillez consulter Rotation gérée.

Pour les informations d'identification Amazon Redshift et Amazon DocumentDB, créez d'abord un secret avec un mot de passe généré par Secrets Manager, puis utilisez une référence dynamique pour récupérer le nom d'utilisateur et le mot de passe du secret à utiliser comme informations d'identification pour une nouvelle base de données. Utilisez ensuite la ressource `AWS::SecretsManager::SecretTargetAttachment` pour ajouter des détails relatifs à la base de données au secret dont Secrets Manager a besoin pour effectuer la rotation du secret. Enfin, pour activer la rotation automatique, utilisez la ressource `AWS::SecretsManager::RotationSchedule` et fournissez une fonction de rotation et un calendrier. Voir les exemples suivants :

- Créez un secret avec les informations d'identification Amazon Redshift
- Créez un secret avec les informations d'identification Amazon DocumentDB

Pour attacher une politique de ressources à votre secret, utilisez la ressource `AWS::SecretsManager::ResourcePolicy`.

Pour plus d'informations sur la création de ressources avec AWS CloudFormation, consultez la section Apprenez les principes de base des modèles dans le guide de AWS CloudFormation l'utilisateur. Vous pouvez également utiliser AWS Cloud Development Kit (AWS CDK). Pour de plus amples informations, consultez AWS Secrets Manager Construct Library.

Créez un AWS Secrets Manager secret avec AWS CloudFormation

Dans cet exemple, un service nommé **CloudFormationCreatedSecret-*a1b2c3d4e5f6*** est créé. La valeur secrète est le JSON suivant, avec un mot de passe de 32 caractères qui est généré lors de la création du secret.

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

Cet exemple utilise la CloudFormation ressource suivante :

- AWS::SecretsManager::Secret

Pour plus d'informations sur la création de ressources avec AWS CloudFormation, consultez la section Apprenez les principes de base des modèles dans le guide de AWS CloudFormation l'utilisateur.

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
```

Créez un AWS Secrets Manager secret avec rotation automatique et une instance de base de données MySQL Amazon RDS avec AWS CloudFormation

Pour créer un secret d'administration pour Amazon RDS ou Aurora, nous vous recommandons d'utiliser `ManageMasterUserPassword`, comme indiqué dans l'exemple Create a Secrets Manager secret for a master password (Créer un secret Secrets Manager pour un mot de passe principal) dans `AWS::RDS::DBCluster`. Esnuite, Amazon RDS crée le secret et gère la rotation pour vous. Pour de plus amples informations, veuillez consulter Rotation gérée.

Créez un AWS Secrets Manager secret et un cluster Amazon Redshift avec AWS CloudFormation

Pour créer un secret d'administration pour Amazon Redshift, nous vous recommandons d'utiliser les exemples figurant sur `AWS::Redshift::Cluster`et. `AWS::RedshiftServerless::Namespace`

Créez un AWS Secrets Manager secret et une instance Amazon DocumentDB avec AWS CloudFormation

Cet exemple crée un secret et une instance Amazon DocumentDB en utilisant les informations d'identification dans le secret comme utilisateur et mot de passe. Une politique basée sur les ressources est attachée au secret ; elle définit qui peut accéder au secret. Le modèle crée également une fonction de rotation Lambda à partir du Modèles de fonctions de rotation et configure le secret pour qu'il effectue une rotation automatiquement entre 8 h 00 et 10 h 00 UTC le premier jour de chaque mois. En tant que bonne pratique de sécurité, l'instance se trouve dans un Amazon VPC.

Cet exemple utilise les CloudFormation ressources suivantes pour Secrets Manager :

- AWS::SecretsManager::Secret
- AWS::SecretsManager::SecretTargetAttachment
- AWS::SecretsManager::RotationSchedule

Pour plus d'informations sur la création de ressources avec AWS CloudFormation, consultez la section Apprenez les principes de base des modèles dans le guide de AWS CloudFormation l'utilisateur.

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 * ? *)
```

Comment utilise Secrets Manager AWS CloudFormation

Lorsque vous utilisez la console pour activer la rotation, Secrets Manager crée AWS CloudFormation des ressources pour la rotation. Si vous créez une nouvelle fonction de rotation au cours de ce processus, AWS CloudFormation créez une `AWS::Serverless::Function`fonction basée sur la fonction appropriéeModèles de fonctions de rotation. AWS CloudFormation Définit ensuite le `RotationSchedule`, qui définit la fonction de rotation et les règles de rotation pour le secret. Vous pouvez afficher la AWS CloudFormation pile en choisissant Afficher la pile dans la bannière après avoir activé la rotation automatique.

Pour plus d'informations sur l'activation de la rotation automatique, veuillez consulter Rotation des secrets.

Créez des AWS Secrets Manager secrets dans AWS Cloud Development Kit (AWS CDK)

Pour créer, gérer et récupérer des secrets dans une application CDK, vous pouvez utiliser la bibliothèque de construction AWS Secrets Manager, qui contient des constructions `ResourcePolicy`, `RotationSchedule`, `Secret`, `SecretRotation` et `SecretTargetAtt`

Une bonne pratique pour utiliser des secrets dans les applications CDK consiste à créer d'abord le secret à l'aide de la console ou de la CLI, puis à l'importer dans votre application CDK.

Pour obtenir des exemples, veuillez consulter :

- Créer un secret
- Importer un secret
- Récupérer un secret
- Accorder l'autorisation d'utiliser le secret
- Réaliser une rotation d'un secret
- Réaliser une rotation d'un secret de base de données
- Répliquer un secret vers d'autres régions

Pour plus d'informations sur le CDK, veuillez consulter le Guide du développeur AWS Cloud Development Kit (AWS CDK) v2.

Surveillez AWS Secrets Manager les secrets

AWS fournit des outils de surveillance permettant de surveiller les secrets de Secrets Manager, de signaler tout problème et de prendre des mesures automatiques le cas échéant. Vous pouvez utiliser les journaux pour examiner tout utilisation ou changement inattendu, et ensuite annuler les modifications non désirées. Vous pouvez aussi définir des contrôles automatisés pour une utilisation inappropriée des secrets et pour toute tentative de suppression de ces derniers.

Rubriques

- Enregistrez AWS Secrets Manager les événements avec AWS CloudTrail
- Surveillez AWS Secrets Manager avec Amazon CloudWatch
- Associez AWS Secrets Manager des événements à Amazon EventBridge
- Surveiller l'accès aux AWS Secrets Manager secrets dont la suppression est prévue
- Surveillez AWS Secrets Manager les secrets pour garantir la conformité en utilisant AWS Config
- Surveillez les coûts de Secrets Manager
- Détectez les menaces avec Amazon GuardDuty

Enregistrez AWS Secrets Manager les événements avec AWS CloudTrail

AWS CloudTrail enregistre tous les appels d'API pour Secrets Manager sous forme d'événements, y compris les appels depuis la console Secrets Manager, ainsi que plusieurs autres événements relatifs à la rotation et à la suppression de versions secrètes. Pour obtenir la liste des entrées du journal dans les enregistrements de Secrets Manager, consultezCloudTrail entrées.

Vous pouvez utiliser la CloudTrail console pour consulter les événements enregistrés au cours des 90 derniers jours. Pour un enregistrement continu des événements de votre AWS compte, y compris des événements relatifs à Secrets Manager, créez un journal qui CloudTrail transmettra les fichiers journaux à un compartiment Amazon S3. Consultez la section Création d'un historique pour votre AWS compte. Vous pouvez également configurer CloudTrail pour recevoir des fichiers CloudTrail journaux provenant de plusieurs Comptes AWS et Régions AWS.

Vous pouvez configurer d'autres AWS services pour analyser davantage les données collectées dans les CloudTrail journaux et agir en conséquence. Découvrez les intégrations de AWS services avec les CloudTrail journaux. Vous pouvez également recevoir des notifications lorsque vous CloudTrail

publiez de nouveaux fichiers journaux dans votre compartiment Amazon S3. Consultez Configuration des notifications Amazon SNS pour. CloudTrail

Pour récupérer les événements de Secrets Manager à partir CloudTrail des journaux (console)

1. Ouvrez la CloudTrail console à l'adresse https://console.aws.amazon.com/cloudtrail/.
2. Assurez-vous que la console pointe vers la région où vos événements se sont produits. La console affiche uniquement les événements survenus dans la région sélectionnée. Choisissez la région dans la liste déroulante située dans le coin supérieur droit de la console.
3. Dans le volet de navigation de gauche, sélectionnez Event history (Historique des événements).
4. Choisissez les critères de filtre et/ou une plage de temps pour vous aider à trouver l'événement que vous recherchez. Par exemple :

    a. Pour voir tous les événements de Secrets Manager, pour les attributs de recherche, sélectionnez Source de l'événement. Ensuite, pour Enter event source (Entrer source d'événement), sélectionnez **`secretsmanager.amazonaws.com`**.

    b. Pour voir tous les événements liés à un secret, dans la zone Attributs de recherche, sélectionnez Nom de la ressource. Ensuite, pour Entrez un nom de ressource, entrez le nom du secret.
5. Pour plus de détails, cliquez sur la flèche d'extension à côté de l'événement. Pour voir toutes les informations disponibles, sélectionnez View event (Afficher l'événement).

AWS CLI

Example Récupérer les événements de Secrets Manager à partir CloudTrail des journaux

L'exemple suivant `lookup-events` recherche les événements de Secrets Manager.

```
aws cloudtrail lookup-events \
    --region us-east-1 \
    --lookup-attributes
 AttributeKey=EventSource,AttributeValue=secretsmanager.amazonaws.com
```

AWS CloudTrail entrées pour Secrets Manager

AWS Secrets Manager écrit des entrées dans votre AWS CloudTrail journal pour toutes les opérations de Secrets Manager et pour les autres événements liés à la rotation et à la suppression.

Pour plus d'informations sur les mesures à prendre face à ces événements, consultez Associez les événements de Secrets Manager à EventBridge.

Types d'entrée de journal

- Entrées de journal pour les opérations Secrets Manager
- Entrées du journal marquées pour suppression
- Entrées de journal pour la réplication
- Entrées du journal pour la rotation

Entrées de journal pour les opérations Secrets Manager

Les événements générés par des appels aux opérations de Secrets Manager ont `"detail-type": ["AWS API Call via CloudTrail"]`.

> **Note**
>
> Avant février 2024, certaines opérations de Secrets Manager signalaient des événements contenant « ARn » au lieu de « »arn« pour l'ARN secret. Pour obtenir plus d'informations, consultez re:Post AWS.

Les CloudTrail entrées suivantes sont générées lorsque vous ou un service appelez les opérations de Secrets Manager via l'API, le SDK ou la CLI.

BatchGetSecretValue

Généré par l'BatchGetSecretValueopération. Pour plus d'informations sur la récupération de secrets, veuillez consulter Obtenez des secrets.

CancelRotateSecret

Généré par l'CancelRotateSecretopération. Pour plus d'informations sur la rotation, veuillez consulter Rotation des secrets.

CreateSecret

Généré par l'CreateSecretopération. Pour plus d'informations sur la création de secrets, veuillez consulter Gérez les secrets.

DeleteResourcePolicy

Généré par l'DeleteResourcePolicyopération. Pour plus d'informations sur les autorisations, veuillez consulter the section called “Authentification et contrôle d'accès”.

DeleteSecret

Généré par l'DeleteSecretopération. Pour plus d'informations sur la suppression de secrets, veuillez consulter the section called “Suppression d'un secret”.

DescribeSecret

Généré par l'DescribeSecretopération.

GetRandomPassword

Généré par l'GetRandomPasswordopération.

GetResourcePolicy

Généré par l'GetResourcePolicyopération. Pour plus d'informations sur les autorisations, veuillez consulter the section called “Authentification et contrôle d'accès”.

GetSecretValue

Généré par les BatchGetSecretValueopérations GetSecretValueet. Pour plus d'informations sur la récupération de secrets, veuillez consulter Obtenez des secrets.

ListSecrets

Généré par l'ListSecretsopération. Pour plus d'informations sur l'établissement d'une liste de secrets, veuillez consulter the section called “Recherche de secrets”.

ListSecretVersionIds

Généré par l'ListSecretVersionIdsopération.

PutResourcePolicy

Généré par l'PutResourcePolicyopération. Pour plus d'informations sur les autorisations, veuillez consulter the section called “Authentification et contrôle d'accès”.

PutSecretValue

Généré par l'PutSecretValueopération. Pour plus d'informations sur la mise à jour d'un secret, veuillez consulter the section called “Modification d'un secret”.

RemoveRegionsFromReplication

Généré par l'RemoveRegionsFromReplicationopération. Pour plus d'informations sur la réplication d'un secret, veuillez consulter Réplication multirégionale.

ReplicateSecretToRegions

Généré par l'ReplicateSecretToRegionsopération. Pour plus d'informations sur la réplication d'un secret, veuillez consulter Réplication multirégionale.

RestoreSecret

Généré par l'RestoreSecretopération. Pour plus d'informations sur la restauration d'un secret supprimé, veuillez consulter the section called “Restaurer un secret”.

RotateSecret

Généré par l'RotateSecretopération. Pour plus d'informations sur la rotation, veuillez consulter Rotation des secrets.

StopReplicationToReplica

Généré par l'StopReplicationToReplicaopération. Pour plus d'informations sur la réplication d'un secret, veuillez consulter Réplication multirégionale.

TagResource

Généré par l'TagResourceopération. Pour plus d'informations sur l'étiquetage d'un secret, veuillez consulter the section called “Étiqueter les secrets ”.

UntagResource

Généré par l'UntagResourceopération. Pour plus d'informations sur la suppression du balisage d'un secret, veuillez consulter the section called “Étiqueter les secrets ”.

UpdateSecret

Généré par l'UpdateSecretopération. Pour plus d'informations sur la mise à jour d'un secret, veuillez consulter the section called “Modification d'un secret”.

UpdateSecretVersionStage

Généré par l'UpdateSecretVersionStageopération. Pour plus d'informations sur les étapes des versions, veuillez consulter the section called “Versions secrètes”.

ValidateResourcePolicy

Généré par l'ValidateResourcePolicyopération. Pour plus d'informations sur les autorisations, veuillez consulter the section called “Authentification et contrôle d'accès”.

Entrées du journal marquées pour suppression

Outre les événements relatifs aux opérations de Secrets Manager, Secrets Manager génère les événements suivants relatifs à la suppression. Ces événements ont `"detail-type": ["AWS Service Event via CloudTrail"]`.

CancelSecretVersionDelete

Généré par le service Secrets Manager. Si vous appelez `DeleteSecret` sur un secret qui a des versions, puis que vous appelez `RestoreSecret` plus tard, Secrets Manager journalise cet événement pour chaque version de secret restaurée. Pour plus d'informations sur la restauration d'un secret supprimé, veuillez consulter the section called “Restaurer un secret”.

EndSecretVersionDelete

Généré par le service Secrets Manager lorsqu'une version de secret est supprimée. Pour de plus amples informations, veuillez consulter the section called “Suppression d'un secret”.

StartSecretVersionDelete

Généré par le service Secrets Manager lorsque ce dernier lance la suppression d'une version de secret. Pour plus d'informations sur la suppression de secrets, veuillez consulter the section called “Suppression d'un secret”.

SecretVersionDeletion

Généré par le service Secrets Manager lorsque ce dernier supprime une version de secret obsolète. Pour plus d'informations, consultez Versions de secret.

Entrées de journal pour la réplication

Outre les événements liés aux opérations de Secrets Manager, Secrets Manager génère les événements suivants liés à la réplication. Ces événements ont `"detail-type": ["AWS Service Event via CloudTrail"]`.

ReplicationFailed

Généré par le service Secrets Manager en cas d'échec de la réplication. Pour plus d'informations sur la réplication d'un secret, veuillez consulter Réplication multirégionale.

ReplicationStarted

Généré par le service Secrets Manager lorsque Secrets Manager lance la réplication d'un secret. Pour plus d'informations sur la réplication d'un secret, veuillez consulter Réplication multirégionale.

ReplicationSucceeded

Généré par le service Secrets Manager lorsque la réplication d'un secret est réussie. Pour plus d'informations sur la réplication d'un secret, veuillez consulter Réplication multirégionale.

Entrées du journal pour la rotation

Outre les événements relatifs aux opérations de Secrets Manager, Secrets Manager génère les événements suivants relatifs à la rotation. Ces événements ont `"detail-type": ["AWS Service Event via CloudTrail"]`.

RotationStarted

Généré par le service Secrets Manager lorsque Secrets Manager lance la rotation d'un secret. Pour plus d'informations sur la rotation, veuillez consulter Rotation des secrets.

RotationAbandoned

Généré par le service Secrets Manager lorsque Secrets Manager abandonne une tentative de rotation et supprime l'étiquette `AWSPENDING` d'une version existante d'un secret. Secrets Manager abandonne la rotation lorsque vous créez une nouvelle version d'un secret pendant la rotation. Pour plus d'informations sur la rotation, veuillez consulter Rotation des secrets.

RotationFailed

Généré par le service Secrets Manager en cas d'échec de la rotation. Pour plus d'informations sur la rotation, veuillez consulter the section called “Résolution des problèmes de rotation”.

RotationSucceeded

Généré par le service Secrets Manager lorsque la rotation d'un secret est réussie. Pour plus d'informations sur la rotation, veuillez consulter Rotation des secrets.

TestRotationStarted

Généré par le service Secrets Manager lorsque Secrets Manager commence à tester la rotation d'un secret dont la rotation immédiate n'est pas planifiée. Pour plus d'informations sur la rotation, veuillez consulter Rotation des secrets.

TestRotationSucceeded

Généré par le service Secrets Manager lorsque Secrets Manager teste avec succès la rotation d'un secret dont la rotation immédiate n'est pas planifiée. Pour plus d'informations sur la rotation, veuillez consulter Rotation des secrets.

TestRotationFailed

Généré par le service Secrets Manager lorsque Secrets Manager teste la rotation d'un secret dont la rotation immédiate n'est pas planifiée et que la rotation a échoué. Pour plus d'informations sur la rotation, veuillez consulter the section called "Résolution des problèmes de rotation".

Surveillez AWS Secrets Manager avec Amazon CloudWatch

À l'aide d'Amazon CloudWatch, vous pouvez surveiller les AWS services et créer des alarmes pour vous informer lorsque les indicateurs changent. CloudWatch conserve ces statistiques pendant 15 mois, afin que vous puissiez accéder aux informations historiques et avoir une meilleure idée des performances de votre application ou service Web. En AWS Secrets Manager effet, vous pouvez contrôler le nombre de secrets de votre compte, y compris les secrets marqués pour suppression, et les appels d'API vers Secrets Manager, y compris les appels passés via la console. Pour plus d'informations sur la façon de surveiller les métriques, voir Utiliser CloudWatch les métriques dans le Guide de CloudWatch l'utilisateur.

Pour trouver les statistiques de Secrets Manager

1. Sur la CloudWatch console, sous Métriques, sélectionnez Toutes les métriques.
2. Dans le champ de recherche de métriques, entrez`secret`.
3. Procédez comme suit :
   - Pour contrôler le nombre de secrets de votre compte, choisissez AWS/SecretsManager, puis sélectionnez SecretCount. Cette métrique est publiée toutes les heures.
   - Pour surveiller les appels d'API à Secrets Manager, y compris les appels effectués via la console, choisissez Utilisation > Par AWS ressource, puis sélectionnez les appels d'API

à surveiller. Pour une liste des Secrets Manager APIs, consultez la section Opérations de Secrets Manager.

4. Procédez comme suit :
   - Pour créer un graphique de la métrique, consultez la section Représentation graphique des métriques dans le guide de l' CloudWatch utilisateur Amazon.
   - Pour détecter les anomalies, consultez la section Utilisation de la détection des CloudWatch anomalies dans le guide de CloudWatch l'utilisateur Amazon.
   - Pour obtenir les statistiques d'une métrique, consultez la section Obtenir les statistiques d'une métrique dans le guide de CloudWatch l'utilisateur Amazon.

CloudWatch alarmes

Vous pouvez créer une CloudWatch alarme qui envoie un message Amazon SNS lorsque la valeur d'une métrique change et provoque le changement d'état de l'alarme. Vous pouvez définir une alarme sur la métrique Secrets Manager`ResourceCount`, qui correspond au nombre de secrets de votre compte. Vous pouvez également définir des alarmes sur Une alarme surveille une métrique sur une période que vous spécifiez et exécute des actions en fonction de la valeur de la métrique par rapport à un seuil donné sur un certain nombre de périodes. Les alarmes déclenchent des actions uniquement pour les changements d'état prolongés. CloudWatch les alarmes n'appellent pas d'actions simplement parce qu'elles sont dans un état particulier ; l'état doit avoir changé et être maintenu pendant un certain nombre de périodes.

Pour plus d'informations, consultez les sections Utilisation des CloudWatch alarmes Amazon et Création CloudWatch d'une alarme basée sur la détection d'anomalies dans le guide de l'CloudWatch utilisateur.

Vous pouvez également définir des alarmes qui surveillent certains seuils et envoient des notifications ou prennent des mesures lorsque ces seuils sont atteints. Pour plus d'informations, consultez le guide de CloudWatch l'utilisateur Amazon.

Associez AWS Secrets Manager des événements à Amazon EventBridge

Dans Amazon EventBridge, vous pouvez associer les événements de Secrets Manager aux entrées du CloudTrail journal. Vous pouvez configurer EventBridge des règles qui recherchent ces événements, puis envoient les nouveaux événements générés à une cible pour qu'elle prenne

des mesures. Pour obtenir la liste des CloudTrail entrées enregistrées par Secrets Manager, consultezCloudTrail entrées. Pour obtenir des instructions de configuration EventBridge, reportez-vous à la section Getting started with EventBridge dans le guide de EventBridge l'utilisateur.

Faire correspondre toutes les modifications à un secret spécifié

> **Note**
>
> Étant donné que certains événements de Secrets Manager renvoient l'ARN du secret avec des majuscules différentes, dans les modèles d'événements correspondant à plusieurs actions, pour spécifier un secret par ARN, vous devrez peut-être inclure à la fois les clés `arn` et `aRN`. Pour obtenir plus d'informations, consultez re:Post AWS.

L'exemple suivant montre un modèle d' EventBridge événement qui correspond aux entrées du journal pour les modifications apportées à un secret.

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

Faire correspondre les événements lorsqu'une valeur secrète change

L'exemple suivant montre un modèle d' EventBridge événement qui correspond aux entrées du CloudTrail journal pour les modifications de valeurs secrètes résultant de mises à jour manuelles ou de rotation automatique. Étant donné que certains de ces événements proviennent d'opérations de Secrets Manager et que d'autres sont générés par le service Secrets Manager, vous devez inclure les `detail-type` pour les deux.

```
{
```

```
    "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"]
    }
}
```

Surveiller l'accès aux AWS Secrets Manager secrets dont la suppression est prévue

Vous pouvez utiliser à la AWS CloudTrail fois Amazon CloudWatch Logs et Amazon Simple Notification Service (Amazon SNS) pour créer une alarme qui vous avertira de toute tentative d'accès à un secret en attente de suppression. Si vous recevez une notification d'une alarme, vous pouvez annuler la suppression du secret afin de disposer de plus de temps pour déterminer si vous souhaitez vraiment le supprimer. Votre investigation peut aboutir à la restauration du secret car vous avez encore besoin du secret. Sinon, vous devrez peut-être mettre à jour l'utilisateur avec les détails du nouveau secret à utiliser.

Les procédures suivantes expliquent comment recevoir une notification lorsqu'une demande d'`GetSecretValue`opération entraîne l'enregistrement d'un message d'erreur spécifique dans vos fichiers CloudTrail journaux. Les autres opérations d'API peuvent être effectuées sur le secret sans déclencher l'alarme. Cette CloudWatch alarme détecte une utilisation qui pourrait indiquer qu'une personne ou une application utilise des informations d'identification périmées.

Avant de commencer ces procédures, vous devez activer le compte Région AWS and CloudTrail dans lequel vous souhaitez surveiller les demandes AWS Secrets Manager d'API. Pour de plus amples informations, consultez Création d'un journal d'activité pour la première fois dans le Guide de l'utilisateur AWS CloudTrail .

Étape 1 : Configuration de la livraison du fichier CloudTrail journal à CloudWatch Logs

Vous devez configurer la livraison de vos fichiers CloudTrail CloudWatch journaux à Logs. Vous procédez ainsi pour que CloudWatch Logs puisse les surveiller afin de détecter les demandes d'API Secrets Manager visant à récupérer un secret en attente de suppression.

Pour configurer la livraison des fichiers CloudTrail journaux à CloudWatch Logs

1. Ouvrez la CloudTrail console à l'adresse https://console.aws.amazon.com/cloudtrail/.
2. Dans la barre de navigation supérieure, choisissez l'option Région AWS pour surveiller les secrets.
3. Dans le volet de navigation de gauche, choisissez Pistes, puis choisissez le nom de la piste pour laquelle vous souhaitez effectuer la configuration CloudWatch.
4. Sur la page Configuration des sentiers, faites défiler la page jusqu'à la section CloudWatch Logs, puis cliquez sur l'icône de modification (✎).
5. Dans New or existing log group (Groupe de journaux nouveau ou existant), saisissez un nom pour le groupe de journaux, comme **`CloudTrail/MyCloudWatchLogGroup`**.
6. Pour le rôle IAM, vous pouvez utiliser le rôle par défaut nommé CloudTrail_ CloudWatchLogs _Role. Ce rôle dispose d'une politique de rôle par défaut avec les autorisations requises pour transmettre CloudTrail des événements au groupe de journaux.
7. Choisissez Continue (Continuer) pour enregistrer votre configuration.
8. Sur la AWS CloudTrail page du groupe de CloudWatch journalisation des CloudTrail événements associés à l'activité des API de votre compte, sélectionnez Autoriser.

Étape 2 : Création de l' CloudWatchalarme

Pour recevoir une notification lorsqu'une opération de `GetSecretValue` l'API Secrets Manager demande l'accès à un secret en attente de suppression, vous devez créer une CloudWatch alarme et configurer une notification.

Pour créer une CloudWatch alarme

1. Connectez-vous à la CloudWatch console à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Dans la barre de navigation supérieure, choisissez la AWS région dans laquelle vous souhaitez surveiller les secrets.
3. Dans le panneau de navigation de gauche, choisissez Logs (Journaux).
4. Dans la liste des groupes de journaux, cochez la case à côté du groupe de journaux que vous avez créé lors de la procédure précédente, tel que CloudTrail/MyCloudWatchLogGroup. Ensuite, choisissez Create Metric Filter (Créer un filtre de métrique).

5. Pour Filter Pattern (Modèle de filtre), entrez ou collez ce qui suit :

```
{ $.eventName = "GetSecretValue" && $.errorMessage = "*secret because it was marked
 for deletion*" }
```

   Choisissez Assign Metric (Affecter une métrique).

6. Sur la page Create Metric Filter and Assign a Metric (Créer un filtre de métrique et affecter une métrique), procédez comme suit :

   a. Dans Metric Namespace (Espace de noms de la métrique), saisissez **`CloudTrailLogMetrics`**.

   b. Dans Metric Name (Nom de la métrique), saisissez **`AttemptsToAccessDeletedSecrets`**.

   c. Choisissez Show advanced metric settings (Afficher les paramètres de métriques avancés), puis, si nécessaire, pour Metric Value (Valeur métrique), saisissez **1**.

   d. Choisissez Create Filter (Créer un filtre).

7. Dans la zone de filtre, choisissez Create Alarm (Créer une alarme).

8. Dans la fenêtre Create Alarm (Créer une alarme), procédez comme suit :

   a. Pour Name (Nom), tapez **`AttemptsToAccessDeletedSecretsAlarm`**.

   b. Sous Whenever (Lorsque), pour is (est :), choisissez **>=** et saisissez **1**.

   c. En regard de Send notification to (Envoyer une notification à), effectuez l'une des opérations suivantes :

      - Pour créer et utiliser une nouvelle rubrique Amazon SNS, choisissez New list (Nouvelle liste), puis saisissez le nom d'une nouvelle rubrique. Pour Email list (Liste des adresses e-mail), tapez au moins une adresse e-mail. Vous pouvez taper plusieurs adresses e-mail en les séparant par des virgules.
      - Pour utiliser une rubrique Amazon SNS existante, choisissez le nom de la rubrique à utiliser. Si aucune liste n'existe, choisissez Select list (Sélectionner la liste).

   d. Sélectionnez Create Alarm (Créer une alerte).

Étape 3 : Testez l' CloudWatchalarme

Pour tester votre alarme, créez un secret, puis planifiez sa suppression. Essayez ensuite de récupérer la valeur du secret. Vous recevez rapidement un e-mail à l'adresse que vous avez configurée dans l'alarme. Il vous avertit de l'utilisation d'un secret dont la suppression est programmée.

Surveillez AWS Secrets Manager les secrets pour garantir la conformité en utilisant AWS Config

Vous pouvez l'utiliser AWS Config pour évaluer vos secrets pour voir s'ils sont conformes à vos normes. Vous définissez vos exigences internes en matière de sécurité et de conformité pour les secrets à l'aide de AWS Config règles. AWS Config Vous pouvez ensuite identifier les secrets qui ne sont pas conformes à vos règles. Vous pouvez également suivre les modifications apportées aux métadonnées secrètes, à la configuration de rotation, à la clé KMS utilisée pour le chiffrement secret, à la fonction de rotation Lambda et aux balises associées à un secret.

Vous pouvez configurer AWS Config pour vous informer des modifications. Pour plus d'informations, consultez la rubrique Notifications AWS Config envoyées à un Amazon SNS.

Si vous avez des secrets dans plusieurs entités Comptes AWS et Régions AWS au sein de votre organisation, vous pouvez agréger ces données de configuration et de conformité. Pour plus d'informations, consultez la section Agrégation de données multicomptes et multirégions.

Pour évaluer si les secrets sont conformes

- Suivez les instructions de la section Évaluation de vos ressources à l'aide de AWS Config règles, puis choisissez l'une des règles suivantes :
  - `secretsmanager-secret-unused` : vérifie si les secrets ont été consultés dans le nombre de jours spécifié.
  - `secretsmanager-using-cmk`— Vérifie si les secrets sont chiffrés à l'aide de la clé que vous avez créée Clé gérée par AWS `aws/secretsmanager` ou d'une clé gérée par le client dans laquelle vous l'avez créée AWS KMS.
  - `secretsmanager-rotation-enabled-check` : vérifie si la rotation est configurée pour les secrets stockés dans Secrets Manager.

- `secretsmanager-scheduled-rotation-success-check` : vérifie si la dernière rotation réussie se situe dans les limites de la fréquence de rotation configurée. La fréquence minimale de contrôle se fait de façon quotidienne.
- `secretsmanager-secret-periodic-rotation` : vérifie si les secrets ont fait l'objet d'une rotation dans le nombre de jours spécifié.

Surveillez les coûts de Secrets Manager

Vous pouvez utiliser Amazon CloudWatch pour surveiller les AWS Secrets Manager frais estimés. Pour plus d'informations, consultez la section Création d'une alarme de facturation pour surveiller vos AWS frais estimés dans le Guide de CloudWatch l'utilisateur.

Une autre option pour surveiller vos coûts est la détection AWS des anomalies de coûts. Pour plus d'informations, consultez la section Détection des dépenses inhabituelles grâce à la détection des anomalies de AWS coût dans le guide de l'utilisateur de AWS Cost Management.

Pour plus d'informations sur le suivi de votre utilisation de Secrets Manager, reportez-vous the section called “Moniteur avec CloudWatch” aux sections etthe section called “Connectez-vous avec AWS CloudTrail ”.

Pour plus d'informations sur la AWS Secrets Manager tarification, consultezthe section called “Tarification”.

Détectez les menaces avec Amazon GuardDuty

Amazon GuardDuty est un service de détection des menaces qui vous aide à protéger vos comptes, vos conteneurs, vos charges de travail et les données de votre AWS environnement. En utilisant des modèles d'apprentissage automatique (ML) et des fonctionnalités de détection des anomalies et des menaces, vous surveillez GuardDuty en permanence différentes sources de journaux afin d'identifier et de hiérarchiser les risques de sécurité potentiels et les activités malveillantes dans votre environnement. Par exemple, GuardDuty détectera les menaces potentielles telles que l'accès inhabituel ou suspect à des secrets et l'exfiltration d'informations d'identification au cas où il détecterait des informations d'identification créées exclusivement pour une EC2 instance Amazon via un rôle de lancement d'instance mais utilisées à partir d'un autre compte interne. AWS Pour plus d'informations, consultez le guide de GuardDuty l'utilisateur Amazon.

Un autre exemple de cas d'utilisation pour la détection est le comportement anormal. Par exemple, si vous AWS Secrets Manager recevez généralement `create-secret` des `list-secrets`

appels d'une entité utilisant le SDK Java, puis qu'une autre entité commence à appeler `batch-get-secret-value` et à `get-secret-value` utiliser le AWS CLI depuis l'extérieur du VPN, GuardDuty vous pouvez signaler une découverte selon laquelle la deuxième entité invoque de manière anormale. `get-secret-value` `describe-secret` APIs Pour plus d'informations, voir Type de recherche GuardDuty IAM CredentialAccess :IAMUser/AnomalousBehavior.

Validation de conformité pour AWS Secrets Manager

Lorsque vous utilisez Secrets Manager, votre responsabilité en matière de conformité dépend de la sensibilité de vos données, des objectifs de conformité de votre entreprise et des lois et réglementations applicables. AWS fournit les ressources suivantes pour faciliter la mise en conformité :

- Guides démarrage rapide de la sécurité et de la conformité. Ces guides de déploiement traitent des considérations architecturales et fournissent des étapes pour déployer des environnements de base axés sur la sécurité et la conformité sur AWS.
- Livre blanc sur l'architecture pour la sécurité et la conformité HIPAA — Ce livre blanc décrit comment les entreprises peuvent créer des applications conformes à la loi HIPAA. AWS
- AWS Ressources de https://aws.amazon.com/compliance/resources/ de conformité — Cette collection de classeurs et de guides peut s'appliquer à votre secteur d'activité et à votre région.
- AWS Config permet d'évaluer la conformité des configurations de vos ressources aux pratiques internes, règlementations et directives du secteur. Pour de plus amples informations, veuillez consulter the section called “Surveillez les secrets à des fins de conformité”.
- AWS Security Hubfournit une vue complète de l'état de votre sécurité interne AWS qui vous aide à vérifier votre conformité aux normes et aux meilleures pratiques du secteur de la sécurité. Pour plus d'informations sur l'utilisation de Security Hub pour évaluer les ressources Secrets Manager, consultez Contrôles d'AWS Secrets Manager dans le Guide de l'utilisateur AWS Security Hub .
- IAM Access Analyzer analyse les politiques, y compris les déclarations de condition dans une politique, qui permettent à une entité externe d'accéder à un secret. Pour plus d'informations, consultez Prévisualisation de l'accès avec Access Analyzer.
- AWS Systems Manager fournit des runbooks prédéfinis pour Secrets Manager. Pour plus d'informations, consultez Référence des runbooks Systems Manager Automation pour Secrets Manager.
- Vous pouvez télécharger des rapports d'audit tiers à l'aide de AWS Artifact. Pour plus d'informations, voir Téléchargement de rapports dans AWS Artifact .

Normes de conformité

AWS Secrets Manager a fait l'objet d'un audit pour les normes suivantes et peut faire partie de votre solution lorsque vous devez obtenir une certification de conformité.

- HIPAA — AWS a élargi son programme de conformité à la loi HIPAA (Health Insurance Portability and Accountability Act) pour l'inclure en AWS Secrets Manager tant que service éligible à la HIPAA. Si vous avez signé un accord de partenariat commercial (BAA) avec AWS, vous pouvez utiliser Secrets Manager pour créer vos applications conformes à la loi HIPAA. AWS propose un livre blanc axé sur la loi HIPAA aux clients qui souhaitent en savoir plus sur la manière dont ils peuvent tirer parti AWS pour le traitement et le stockage des informations de santé. Pour de plus amples informations, consultez Conformité à la loi HIPAA.
- Organisation participante à la norme PCI — AWS Secrets Manager possède une attestation de conformité à la norme de sécurité des données (DSS) de l'industrie des cartes de paiement (PCI) version 3.2 au niveau 1 du fournisseur de services. Les clients qui utilisent AWS des produits et services pour stocker, traiter ou transmettre les données des titulaires de cartes peuvent les utiliser pour AWS Secrets Manager gérer leur propre certification de conformité à la norme PCI DSS. Pour plus d'informations sur la norme PCI DSS, notamment sur la manière de demander une copie du Package de AWS conformité PCI, consultez la section PCI DSS niveau 1.
- ISO — AWS Secrets Manager a obtenu avec succès les certifications de conformité pour ISO/IEC 27001, ISO/IEC 27017, ISO/IEC 27018 et ISO 9001. Pour plus d'informations, consultez ISO 27001, ISO 27017, ISO 27018 et ISO 9001.
- Les rapports SOC — System and Organization Control (SOC) de l'AICPA sont des rapports d'examen indépendants réalisés par des tiers qui montrent comment Secrets Manager atteint les principaux contrôles et objectifs de conformité. L'objectif de ces rapports est de vous aider, ainsi que vos auditeurs, à comprendre les AWS contrôles établis pour soutenir les opérations et la conformité. Pour plus d'informations, consultez Conformité SOC.
- FedRAMP — Le programme fédéral de gestion des risques et des autorisations (FedRAMP) est un programme gouvernemental qui fournit une approche standardisée en matière d'évaluation de la sécurité, d'autorisation et de surveillance continue des produits et services cloud. Le programme FedRAMP fournit également des autorisations provisoires pour les services et les régions pour l'Est/Ouest et pour la consommation de données gouvernementales ou GovCloud réglementées. Pour plus d'informations, consultez Conformité au programme FedRAMP.
- Département de la défense — Le guide des exigences de sécurité du cloud computing (SRG) du ministère de la Défense (DoD) fournit un processus d'évaluation et d'autorisation standardisé permettant aux fournisseurs de services cloud (CSPs) d'obtenir une autorisation provisoire du DoD, afin qu'ils puissent servir les clients du DoD. Pour plus d'informations, consultez Resources DoD SRG
- IRAP — Le programme d'évaluateurs enregistrés en matière de sécurité de l'information (IRAP) permet aux clients du gouvernement australien de valider que les contrôles appropriés sont

en place et de déterminer le modèle de responsabilité approprié pour répondre aux exigences du manuel de sécurité de l'information (ISM) du gouvernement australien produit par le Centre australien de cybersécurité (ACSC). Pour plus d'informations, consultez Ressources IRAP.

- OSPAR — Amazon Web Services (AWS) a obtenu l'attestation OSPAR (Outsourced Service Provider's Audit Report). AWS l'alignement sur les directives de l'Association des banques de Singapour (ABS) sur les objectifs et procédures de contrôle pour les fournisseurs de services externalisés (directives ABS) démontre aux AWS clients l'engagement des clients à répondre aux attentes élevées du secteur des services financiers à Singapour à l'égard des fournisseurs de services cloud. Pour plus d'informations, consultez Ressources OSPAR.

Sécurité dans AWS Secrets Manager

La sécurité AWS est la priorité absolue. En tant que AWS client, vous bénéficiez d'un centre de données et d'une architecture réseau conçus pour répondre aux exigences des entreprises les plus sensibles en matière de sécurité.

Vous et AWS partagez la responsabilité de la sécurité. Le modèle de responsabilité partagée décrit ceci en tant que sécurité du cloud et sécurité dans le cloud :

- Sécurité du cloud : AWS est chargée de protéger l'infrastructure qui exécute les AWS services dans le AWS cloud. AWS vous fournit également des services que vous pouvez utiliser en toute sécurité. Des auditeurs tiers testent et vérifient régulièrement l'efficacité de notre sécurité dans le cadre des programmes de conformitéAWS. Pour en savoir plus sur les programmes de conformité qui s'appliquent à AWS Secrets Manager, consultez la section AWS Services concernés par programme de conformité.
- Sécurité dans le cloud : votre AWS service détermine votre responsabilité. Vous êtes également responsable d'autres facteurs, y compris de la sensibilité de vos données, des exigences de votre entreprise, ainsi que de la législation et de la réglementation applicables.

Pour plus de ressources, consultez Security Pillar — AWS Well-Architected Framework.

Rubriques

- Réduisez les risques liés AWS CLI à l'utilisation du pour stocker vos AWS Secrets Manager secrets
- Authentification et contrôle d'accès pour AWS Secrets Manager
- Protection des données dans AWS Secrets Manager
- Chiffrement et déchiffrement secrets dans AWS Secrets Manager
- Sécurité de l'infrastructure dans AWS Secrets Manager
- Utilisation d'un point de AWS Secrets Manager terminaison VPC
- Contrôlez l'accès aux API avec des politiques IAM
- Résilience dans AWS Secrets Manager
- TLS post-quantique

Réduisez les risques liés AWS CLI à l'utilisation du pour stocker vos AWS Secrets Manager secrets

Lorsque vous utilisez le AWS Command Line Interface (AWS CLI) pour appeler AWS des opérations, vous entrez ces commandes dans un interpréteur de commandes. Par exemple, vous pouvez utiliser l'invite de commande Windows ou Windows PowerShell, ou le shell Bash ou Z, entre autres. Bon nombre de ces shells de commande incluent des fonctionnalités conçues pour accroître la productivité. Toutefois, ces fonctionnalités peuvent être utilisées pour compromettre vos secrets. Par exemple, dans la plupart des shells, vous pouvez utiliser la touche de direction vers le haut pour afficher la dernière commande entrée. La fonction historique des commandes peut être exploitée par quiconque accède à votre session non sécurisée. En outre, d'autres utilitaires qui fonctionnent en arrière-plan peuvent avoir accès à vos paramètres de commande, dans le but de vous aider à exécuter plus efficacement vos tâches. Pour atténuer ces risques, veillez à prendre les mesures suivantes :

- Verrouillez toujours votre ordinateur lorsque vous vous éloignez de la console.
- Désinstallez ou désactivez les utilitaires de console dont vous n'avez pas besoin ou que vous n'utilisez plus.
- Assurez-vous que le shell ou le programme d'accès distant, si vous en utilisez un, ne consignent pas les commandes saisies.
- Utilisez des techniques pour transmettre des paramètres sans qu'ils soient capturés par l'historique des commandes du shell. L'exemple suivant montre comment vous pouvez taper le texte secret dans un fichier texte, puis transmettre le fichier à la AWS Secrets Manager commande et le détruire immédiatement. Cela signifie que l'historique shell type ne capture pas le texte secret.

  L'exemple suivant montre des commandes Linux classiques, mais votre shell peut avoir besoin de commandes légèrement différentes :

```
$ 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.
```

Une fois que vous avez exécuté ces commandes, vous devez pouvoir utiliser les flèches vers le haut et vers le bas pour faire défiler l'historique des commandes et constater que le texte secret ne s'affiche sur aucune ligne.

> **Important**
>
> Par défaut, vous ne pouvez pas exécuter une technique équivalente dans Windows, à moins de réduire au préalable la taille du tampon de l'historique des commandes à 1.

Pour configurer l'invite de commande Windows afin d'avoir uniquement 1 tampon d'historique des commandes avec 1 commande

1. Ouvrez une fenêtre d'invite de commande d'administrateur (Run as administrator (Exécuter en tant qu'administrateur)).
2. Cliquez sur l'icône en haut à gauche, puis sur Propriétés.
3. Dans l'onglet Options, définissez Buffer Size (Taille de la mémoire tampon) et Number of Buffers (Nombre de mémoires tampon) sur **1**, puis choisissez OK.
4. Chaque fois que vous devez saisir une commande que vous ne voulez pas consigner dans l'historique, faites-la suivre immédiatement par une autre commande, telle que :

   ```
   echo.
   ```

   Cela vous garantit de vider la commande sensible.

Pour le shell d'invite de commandes Windows, vous pouvez télécharger l'SysInternals SDeleteoutil, puis utiliser des commandes similaires aux suivantes :

```
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.
```

Authentification et contrôle d'accès pour AWS Secrets Manager

Secrets ManagerAWS Identity and Access Management (IAM)pour sécuriser l'accès aux secrets. IAM fournit une authentification et un contrôle d'accès. Authentification Vérifie l'identité des personnes qui émettent des demandes. Secrets Manager utilise un processus de connexion avec des mots de passe, des clés d'accès et des jetons d'authentification multi-facteurs (MFA) pour vérifier l'identité des utilisateurs. Voir Se connecter à AWS. Contrôle d'accès vérifie que seules les personnes autorisées peuvent effectuer des opérations sur AWS les ressources, telles que des secrets. Secrets Manager utilise des politiques pour définir qui a accès aux ressources, et quelles actions l'identité peut entreprendre sur ces ressources. Consultez Autorisations et stratégies dans IAM.

Rubriques

- Référence des autorisations pour AWS Secrets Manager
- Secrets Manager
- Autorisations d'accès aux secrets
- Autorisations pour les fonctions de rotation Lambda
- Autorisations pour les clés de chiffrement
- Autorisations pour la réplication
- Politiques basées sur l'identité
- Politiques basées sur les ressources
- Contrôlez l'accès aux secrets à l'aide du contrôle d'accès basé sur les attributs (ABAC)
- AWS politique gérée pour AWS Secrets Manager
- Déterminez qui est autorisé à accéder à vos AWS Secrets Manager secrets

- Accédez aux AWS Secrets Manager secrets depuis un autre compte
- Accédez aux secrets depuis un environnement sur site

Référence des autorisations pour AWS Secrets Manager

La référence des autorisations pour Secrets Manager est disponible dans Actions, ressources et clés de condition AWS Secrets Manager dans la référence d'autorisation du service.

Secrets Manager

Pour accorder des autorisations d'administrateur Secrets Manager, suivez les instructions dans Ajout et suppression d'autorisations d'identité IAMet joignez les politiques suivantes :

- SecretsManagerReadWrite
- IAMFullAccess

Il est déconseillé d'accorder des autorisations d'administrateur aux utilisateurs finaux. Bien que cela permet à vos utilisateurs de créer et de gérer leurs secrets, l'autorisation requise pour activer la rotation (IAMFullAccess) accorde des autorisations essentielles qui ne sont pas appropriées pour les utilisateurs finaux.

Autorisations d'accès aux secrets

Grâce aux stratégies d'autorisation IAM, vous pouvez vérifier quels utilisateurs ou services ont accès à vos secrets. Une stratégie d'autorisation décrit qui peut effectuer quelles actions sur quelles ressources. Vous pouvez :

- the section called “Politiques basées sur l'identité”
- the section called “Politiques basées sur les ressources”

Autorisations pour les fonctions de rotation Lambda

Secrets Manager utilise des AWS Lambda fonctions pour faire pivoter les secrets. La fonction Lambda doit avoir accès au secret et à la base de données ou au service pour lequel le secret contient des informations d'identification. Voir Autorisations de rotation.

Autorisations pour les clés de chiffrement

Secrets Manager utilise des clés AWS Key Management Service (AWS KMS) pour chiffrer les secrets. Clé gérée par AWS `aws/secretsmanager`Dispose automatiquement des autorisations appropriées. Si vous utilisez une autre clé KMS, des autorisations sont requises par Secrets Manager pour cette dernière. Consultez the section called "Autorisations pour la clé KMS".

Autorisations pour la réplication

En utilisant les politiques d'autorisation IAM, vous contrôlez quels utilisateurs ou services peuvent répliquer vos secrets dans d'autres régions. Consultez the section called "Empêcher la réplication".

Politiques basées sur l'identité

Attachez les stratégies d'autorisation aux identitésIAM : utilisateurs, groupes et rôles. Dans le cadre d'une stratégie basée sur les identités, vous spécifiez à quels secrets l'identité peut accéder et les actions que l'identité peut effectuer sur les secrets. Pour de plus amples informations, consulter Ajout et suppression d'autorisations d'identité IAM.

Vous pouvez accorder des autorisations à un rôle qui représente une application ou un utilisateur dans un autre service. Par exemple, une application exécutée sur une EC2 instance Amazon peut avoir besoin d'accéder à une base de données. Vous pouvez créer un rôle IAM attaché au profil d' EC2 instance, puis utiliser une politique d'autorisation pour accorder au rôle l'accès au secret contenant les informations d'identification de la base de données. Pour plus d'informations, consultez Utiliser un rôle IAM pour accorder des autorisations aux applications exécutées sur des EC2 instances Amazon. Les autres services auxquels vous pouvez attacher des rôles incluent Amazon Redshift, AWS Lambda, et Amazon ECS.

Vous pouvez aussi accorder des autorisations aux utilisateurs authentifiés par un système d'identité autre qu'IAM. Par exemple, vous pouvez associer des rôles IAM aux utilisateurs d'applications mobiles qui se connectent avec Amazon Cognito. Le rôle accorde aux informations d'identification temporaires de l'application les autorisations figurant dans la stratégie d'autorisation du rôle. Vous pouvez ensuite utiliser une stratégie d'autorisations pour accorder au rôle l'accès au secret. Pour de plus amples informations, consultez Fournisseurs d'identité et fédération.

Vous devez utiliser des stratégies basées sur les identités pour :

- Accorder à une identité l'accès à plusieurs secrets.

- Contrôler qui peut créer de nouveaux secrets et qui peut accéder aux secrets qui n'ont pas encore été créés.
- Accorder à un groupe IAM l'accès aux secrets.

Exemples :

- Exemple : Autorisation pour récupérer des valeurs de secrets individuels
- Exemple : autorisation de lire et de décrire des secrets individuels
- Exemple : autorisation de récupérer un groupe de valeurs secrètes dans un lot
- Exemple : Caractères génériques
- Exemple : Autorisation de créer des secrets
- Exemple : refuser une AWS KMS clé spécifique pour chiffrer des secrets

Exemple : Autorisation pour récupérer des valeurs de secrets individuels

Pour accorder l'autorisation de récupérer des valeurs de secrets, vous pouvez attacher des stratégies à des secrets ou des identités. Pour savoir comment déterminer le type de stratégie à utiliser, consultez Stratégies basées sur l'identité et Stratégies basées sur une ressource. Pour plus d'informations sur la façon d'attacher une stratégie, consultez the section called “Politiques basées sur les ressources” et the section called “Politiques basées sur l’identité”.

Cet exemple est utile lorsque vous souhaitez accorder un accès à un groupe IAM. Pour autoriser la récupération d'un groupe de secrets lors d'un appel d'API par lots, consultez the section called “Exemple : autorisation de récupérer un groupe de valeurs secrètes dans un lot”.

Example Lire un secret chiffré à l'aide d'une clé gérée par le client

Si un secret est chiffré à l'aide d'une clé gérée par le client, vous pouvez autoriser l'accès à la lecture du secret en attachant la politique suivante à une identité. \

JSON

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

```
  "Effect": "Allow",
  "Action": "secretsmanager:GetSecretValue",
  "Resource": "SecretARN"
},
{
  "Effect": "Allow",
  "Action": "kms:Decrypt",
  "Resource": "KMSKeyARN"
}
]
}
```

Exemple : autorisation de lire et de décrire des secrets individuels

Example Lisez et décrivez un secret

Vous pouvez accorder l'accès à un secret en attachant la stratégie suivante à une identité.

JSON

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

Exemple : autorisation de récupérer un groupe de valeurs secrètes dans un lot

Example Lire un groupe de secrets par lots

Vous pouvez accorder l'accès pour récupérer un groupe de secrets dans un appel d'API par lots en attachant la stratégie suivante à une identité. La politique restreint l'appelant afin qu'il ne puisse

récupérer que les secrets spécifiés par*SecretARN1*, et *SecretARN2SecretARN3*, même si l'appel par lots inclut d'autres secrets. Si l'appelant demande également d'autres secrets lors de l'appel d'API par lots, Secrets Manager ne les renverra pas. Pour plus d'informations, consultez BatchGetSecretValue. .

JSON

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

Exemple : Caractères génériques

Vous pouvez utiliser des caractères génériques pour inclure un ensemble de valeurs dans un élément de stratégie.

Example Accédez à tous les secrets d'un chemin

La politique suivante autorise l'accès à la récupération de tous les secrets dont le nom commence par « *TestEnv/* ».

JSON

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

Example Accédez aux métadonnées de tous les secrets

La stratégie suivante donne `DescribeSecret` et les autorisations commençant par `List`: `ListSecrets` et `ListSecretVersionIds`.

JSON

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

Example Nom du secret correspondant

La stratégie suivante accorde toutes les autorisations Secrets Manager pour un secret par nom. Pour utiliser cette stratégie, consultez the section called "Politiques basées sur l'identité".

Pour faire correspondre un nom du secret, créez l'ARN pour le secret en regroupant la région, l'ID de compte, le nom du secret et le caractère générique (?) pour faire correspondre les caractères aléatoires individuels. Secrets Manager ajoute six caractères aléatoires aux noms

du secret dans leur ARN, de à ce que vous puissiez utiliser ce caractère générique pour faire correspondre ces caractères. Si vous utilisez la syntaxe `"another_secret_name-*"`, Secrets Manager met en correspondance le secret souhaité avec les 6 caractères aléatoires ainsi que `"another_secret_name-<anything-here>a1b2c3"`.

Comme il est possible de prévoir toutes les parties de l'ARN d'un secret, à l'exception des 6 caractères aléatoires, l'utilisation de la syntaxe `'??????'` de caractère générique permet d'accorder en toute sécurité des autorisations à un secret qui n'existe pas encore. Soyez conscient toutefois, que si vous supprimez le secret, puis le recréez avec le même nom, l'utilisateur reçoit automatiquement l'autorisation sur le nouveau secret, même si les 6 caractères ont changé.

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

Exemple : Autorisation de créer des secrets

Pour accorder à un utilisateur l'autorisation de créer un secret, il est conseillé d'attacher une stratégie d'autorisations à un groupe IAM auquel l'utilisateur appartient. Voir Groupes d'utilisateurs IAM.

Example Créez des secrets

La stratégie suivante autorise la création de secrets et l'affichage d'une liste de secrets. Pour utiliser cette stratégie, consultez the section called “Politiques basées sur l'identité”.

JSON

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

Exemple : refuser une AWS KMS clé spécifique pour chiffrer des secrets

> **Important**
>
> Pour refuser une clé gérée par le client, nous vous recommandons de restreindre l'accès au moyen d'une politique de clés ou d'une attribution de clés. Pour plus d'informations, consultez la section Authentification et contrôle d'accès AWS KMS dans le guide du AWS Key Management Service développeur.

Example Refuser la clé AWS gérée **`aws/secretsmanager`**

La politique suivante interdit l'utilisation du Clé gérée par AWS `aws/secretsmanager` pour créer ou mettre à jour des secrets. Cette politique exige que les secrets soient chiffrés à l'aide d'une clé gérée par le client. La stratégie comprend deux déclarations :

1. La première déclaration`Sid: "RequireCustomerManagedKeysOnSecrets"`, refuse les demandes de création ou de mise à jour de secrets à l'aide du Clé gérée par AWS `aws/secretsmanager`.
2. La deuxième déclaration`Sid: "RequireKmsKeyIdParameterOnCreate"`, refuse les demandes de création de secrets qui n'incluent pas de clé KMS, car Secrets Manager utiliserait par défaut le Clé gérée par AWS `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"
          }
      }
  }
]
}
```

Politiques basées sur les ressources

Dans la stratégie basée sur une ressource, vous spécifiez les personnes qui peuvent accéder au secret et les actions qu'elles peuvent exécuter sur celui-ci. Vous pouvez utiliser des stratégies basées sur une ressource pour :

- Accorder l'accès à un seul secret.à plusieurs utilisateurs ou rôles.

- Accordez l'accès à des utilisateurs ou à des rôles dans d'autres AWS comptes.

Lorsque vous attachez une stratégie basée sur une ressource à un secret dans la console, Secrets Manager utilise le moteur de raisonnement automatisé Zelkova et l'API `ValidateResourcePolicy` pour éviter que vous n'accordiez un accès à vos secrets à un grand nombre de mandataires IAM. Vous pouvez également appeler l'API `PutResourcePolicy` avec le paramètre `BlockPublicPolicy` à partir de l'interface de ligne de commande CLI ou SDK.

Important

La validation des politiques de ressources et le `BlockPublicPolicy` paramètre aident à protéger vos ressources en empêchant l'accès public par le biais des politiques de ressources directement associées à vos secrets. Outre l'utilisation de ces fonctionnalités, examinez attentivement les politiques suivantes pour vous assurer qu'elles n'accordent pas d'accès public :

- Politiques basées sur l'identité associées aux AWS principaux associés (par exemple, les rôles IAM)
- Politiques basées sur les AWS ressources associées (par exemple, AWS Key Management Service (AWS KMS) clés)

Pour vérifier les autorisations relatives à vos secrets, consultezDéterminer qui a les autorisations pour vos secrets .

Pour afficher, modifier ou supprimer la stratégie de ressources d'un secret (console)

1. Ouvrez la console Secrets Manager à l'adresse https://console.aws.amazon.com/secretsmanager/.
2. Dans la liste des secrets, choisissez le secret.
3. Sur la page des détails secrets, sous l'onglet Présentation, dans la section Autorisations d'accès aux ressources, choisissez Modifier des autorisations.
4. Dans le champ de code, suivez une de ces procédures, puis sélectionnez Enregistrer :
   - Pour attacher ou modifier une stratégie de ressource, entrez la stratégie.
   - Pour supprimer la stratégie, désactivez le champ de code.

AWS CLI

Example Récupérer une politique de ressources

L'exemple suivant `get-resource-policy` récupère la politique basée sur les ressources associée à un secret.

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

Example Supprimer une politique de ressources

L'exemple suivant `delete-resource-policy` supprime la politique basée sur les ressources associée à un secret.

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

Example Ajouter une politique de ressources

L'exemple suivant `put-resource-policy` ajoute une politique d'autorisations à un secret, en vérifiant d'abord que la politique ne fournit pas un accès étendu au secret. La politique est lue à partir d'un fichier. Pour plus d'informations, consultez la section Chargement de AWS CLI paramètres depuis un fichier dans le Guide de AWS CLI l'utilisateur.

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

Contenu de `mypolicy.json` :

JSON

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

```
                "AWS": "arn:aws:iam::123456789012:role/MyRole"
            },
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "*"
        }
    ]
}
```

AWS SDK

Pour récupérer la stratégie attachée au secret, utilisez `GetResourcePolicy` .

Pour supprimer une stratégie attachée à un secret, utilisez `DeleteResourcePolicy`.

Pour attacher une stratégie à un secret, utilisez `PutResourcePolicy`. Si une stratégie est déjà attachée, la commande la remplace par la nouvelle stratégie. La stratégie doit avoir le format de texte structuré JSON. Consultez Structure d'un document de stratégie JSON.

Pour de plus amples informations, veuillez consulter the section called “AWS SDKs”.

Exemples

Exemples :

- Exemple : Autorisation pour récupérer des valeurs de secrets individuels
- Exemple : autorisations et VPCs
- Exemple : Principal du service

Exemple : Autorisation pour récupérer des valeurs de secrets individuels

Pour accorder l'autorisation de récupérer des valeurs de secrets, vous pouvez attacher des stratégies à des secrets ou des identités. Pour savoir comment déterminer le type de stratégie à utiliser, consultez Stratégies basées sur l'identité et Stratégies basées sur une ressource. Pour plus d'informations sur la façon d'attacher une stratégie, consultez the section called “Politiques basées sur les ressources” et the section called “Politiques basées sur l'identité”.

Cet exemple est utile lorsque vous souhaitez accorder à plusieurs utilisateurs ou rôles l'accès à un secret unique. Pour autoriser la récupération d'un groupe de secrets lors d'un appel d'API par lots, consultez the section called “Exemple : autorisation de récupérer un groupe de valeurs secrètes dans un lot”.

Example Lisez un secret

Vous pouvez accorder à un secret l'accès en attachant la stratégie suivante au secret.

JSON

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

Exemple : autorisations et VPCs

Si vous devez accéder à Secrets Manager à partir d'un VPC, vous devez vérifier que les demandes adressées à Secrets Manager proviennent du VPC en incluant une condition dans vos stratégies d'autorisations. Pour de plus amples informations, consultez Limitez les demandes en fonction des conditions du point de terminaison VPC et the section called “Points de terminaison d’un VPC (AWS PrivateLink)”.

Assurez-vous que les demandes d'accès au secret provenant d'autres AWS services proviennent également du VPC, sinon cette politique leur refusera l'accès.

Example Exiger que les demandes passent par un point de terminaison VPC

La stratégie de secret suivante autorise un utilisateur à effectuer des opérations dans Secrets Manager uniquement lorsque la requête provient du point de terminaison d'un VPC spécifié *vpce-1234a5678b9012c*.

JSON

```
{
```

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

Example Exiger que les demandes proviennent d'un VPC

La stratégie de secret suivante autorise les commandes à créer et gérer les secrets uniquement lorsqu'ils proviennent de *vpc-12345678*. En outre, la stratégie autorise les opérations qui utilisent l'accès à la valeur chiffrée du secret uniquement lorsque les demandes proviennent de vpc-2b2b2b2b. Vous pouvez utiliser une stratégie comme celle-ci si vous exécutez une application dans un VPC, mais que vous utilisez un second VPC isolé pour les fonctions de gestion.

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"
      }
    }
  }
  ]
  }
```

Exemple : Principal du service

Si la politique de ressources attachée à votre secret inclut un principal de AWS service, nous vous recommandons d'utiliser les clés de condition SourceAccount globales aws : SourceArn et aws :. Les valeurs ARN et compte sont incluses dans le contexte d'autorisation uniquement lorsqu'une demande arrive à Secrets Manager depuis un autre service AWS . Cette combinaison de conditions permet d'éviter un possible scénario du député confus.

Si un ARN de ressources inclut des caractères non autorisés dans une politique de ressource, vous ne pouvez pas utiliser cet ARN de ressource dans la valeur de la clé de condition `aws:SourceArn`.

Utilisez à la place la clé de condition `aws:SourceAccount`. Pour plus d'informations, consultez les conditions IAM.

Les principaux de service ne sont généralement pas utilisés comme principes dans une politique attachée à un secret, mais certains AWS services l'exigent. Pour plus d'informations sur les politiques de ressources qu'un service vous demande d'attacher à un secret, consultez la documentation du service.

Example Autoriser un service à accéder à un secret à l'aide d'un principal de service

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"
    }
  }

}
]
}
```

Contrôlez l'accès aux secrets à l'aide du contrôle d'accès basé sur les attributs (ABAC)

Le contrôle d'accès basé sur les attributs (ABAC) est une stratégie d'autorisation qui définit les autorisations en fonction des attributs ou des caractéristiques de l'utilisateur, des données ou de l'environnement, tels que le département, l'unité commerciale ou d'autres facteurs susceptibles d'affecter le résultat de l'autorisation. Dans AWS, ces attributs sont appelés balises.

Le fait d'utiliser des balises pour contrôler les autorisations est utile dans les environnements qui connaissent une croissance rapide et pour les cas où la gestion des stratégies devient fastidieuse. Les règles ABAC sont évaluées dynamiquement lors de l'exécution, ce qui signifie que l'accès des utilisateurs aux applications et aux données ainsi que le type d'opérations autorisées changent automatiquement en fonction des facteurs contextuels de la politique. Par exemple, si un utilisateur change de service, l'accès est automatiquement ajusté sans qu'il soit nécessaire de mettre à jour les autorisations ou de demander de nouveaux rôles. Pour plus d'informations, voir : À quoi sert ABAC ? AWS , Définissez les autorisations d'accès aux secrets en fonction des balises. , et adaptez vos besoins d'autorisation pour Secrets Manager à l'aide d'ABAC avec IAM Identity Center.

Exemple : autoriser une identité à accéder à des secrets dotés de balises spécifiques

La politique suivante autorise `DescribeSecret` l'accès aux secrets dotés d'une balise contenant la clé *`ServerName`* et la valeur*`ServerABC`*. Si vous associez cette politique à une identité, celle-ci est autorisée à accéder à tous les secrets associés à cette étiquette dans le compte.

JSON

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

Exemple : autoriser l'accès uniquement aux identités dont les tags correspondent aux tags des secrets

La politique suivante permet à toutes les identités du compte `GetSecretValue` d'accéder à tous les secrets du compte lorsque le *`AccessProject`* tag d'identité a la même valeur que le *`AccessProject`* tag du secret.

JSON

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

AWS politique gérée pour AWS Secrets Manager

Une politique AWS gérée est une politique autonome créée et administrée par AWS. AWS les politiques gérées sont conçues pour fournir des autorisations pour de nombreux cas d'utilisation courants afin que vous puissiez commencer à attribuer des autorisations aux utilisateurs, aux groupes et aux rôles.

N'oubliez pas que les politiques AWS gérées peuvent ne pas accorder d'autorisations de moindre privilège pour vos cas d'utilisation spécifiques, car elles sont accessibles à tous les AWS clients. Nous vous recommandons de réduire encore les autorisations en définissant des politiques gérées par le client qui sont propres à vos cas d'utilisation.

Vous ne pouvez pas modifier les autorisations définies dans les politiques AWS gérées. Si les autorisations définies dans une politique AWS gérée sont AWS mises à jour, la mise à jour affecte toutes les identités principales (utilisateurs, groupes et rôles) auxquelles la politique est attachée. AWS est le plus susceptible de mettre à jour une politique AWS gérée lorsqu'une nouvelle politique Service AWS est lancée ou lorsque de nouvelles opérations d'API sont disponibles pour les services existants.

Pour plus d'informations, consultez Politiques gérées par AWS dans le Guide de l'utilisateur IAM.

AWS politique gérée : SecretsManagerReadWrite

Cette politique fournit read/write l'accès aux ressources Amazon RDS AWS Secrets Manager, Amazon Redshift et Amazon DocumentDB, y compris l'autorisation de les décrire, ainsi que l'autorisation de les AWS KMS utiliser pour chiffrer et déchiffrer des secrets. Cette politique autorise également la création d'ensembles de AWS CloudFormation modifications, l'obtention de modèles de rotation à partir d'un compartiment Amazon S3 géré par AWS, la liste des AWS Lambda fonctions et la description d'Amazon EC2 VPCs. Ces autorisations sont requises par la console pour configurer la rotation avec les fonctions de rotation existantes.

Pour créer de nouvelles fonctions de rotation, vous devez également être autorisé à créer des AWS CloudFormation piles et des rôles AWS Lambda d'exécution. Vous pouvez attribuer la politique de gestion des IAMFullaccès. Consultez Autorisations de rotation.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes.

- `secretsmanager` : permet aux principaux d'effectuer toutes les actions de Secrets Manager.
- `cloudformation`— Permet aux principaux de créer des AWS CloudFormation piles. Cela est nécessaire pour que les directeurs utilisant la console pour activer la rotation puissent créer des fonctions AWS CloudFormation de rotation Lambda par le biais de piles. Pour de plus amples informations, veuillez consulter the section called "Comment utilise Secrets Manager AWS CloudFormation".
- `ec2`— Permet aux directeurs de décrire Amazon EC2 VPCs. Cela est nécessaire pour que les principaux utilisateurs de la console puissent créer des fonctions de rotation dans le même VPC que la base de données contenant les informations d'identification qu'ils stockent dans un secret.
- `kms`— Permet aux principaux d'utiliser des AWS KMS clés pour les opérations cryptographiques. Cela est nécessaire pour que Secrets Manager puisse chiffrer et déchiffrer les secrets. Pour de

plus amples informations, veuillez consulter the section called "Chiffrement et déchiffrement de secret".

- `lambda` : permet aux principaux de répertorier les fonctions de rotation Lambda. Cela est nécessaire pour que les principaux utilisateurs de la console puissent choisir les fonctions de rotation existantes.
- `rds` : permet aux principaux de décrire des clusters et des instances dans Amazon RDS. Cela est nécessaire pour que les principaux utilisateurs de la console puissent choisir des clusters ou des instances Amazon RDS.
- `redshift` : permet aux principaux de décrire les clusters dans Amazon Redshift. Cela est nécessaire pour que les principaux utilisateurs de la console puissent choisir des clusters Amazon Redshift.
- `redshift-serverless`— Permet aux principaux de décrire les espaces de noms dans Amazon Redshift Serverless. Cela est nécessaire pour que les principaux utilisant la console puissent choisir les espaces de noms Amazon Redshift Serverless.
- `docdb-elastic` : permet aux principaux de décrire les clusters élastiques dans Amazon DocumentDB. Cela est nécessaire pour que les principaux utilisateurs de la console puissent choisir les clusters élastiques Amazon DocumentDB.
- `tag` : permet aux principaux d'accéder à toutes les ressources du compte qui sont étiquetées.
- `serverlessrepo`— Permet aux directeurs de créer des ensembles de AWS CloudFormation modifications. Cela est nécessaire pour que les principaux utilisateurs de la console puissent créer des fonctions de rotation Lambda. Pour de plus amples informations, veuillez consulter the section called "Comment utilise Secrets Manager AWS CloudFormation".
- `s3`— Permet aux principaux d'obtenir des objets depuis un compartiment Amazon S3 géré par AWS. Ce compartiment contient Modèles de fonctions de rotation Lambda. Cette autorisation est requise pour que les principaux utilisateurs de la console puissent créer des fonctions de rotation Lambda basées sur les modèles du compartiment. Pour de plus amples informations, veuillez consulter the section called "Comment utilise Secrets Manager AWS CloudFormation".

Pour consulter la politique, consultez le document de politique SecretsManagerReadWrite JSON.

Mises à jour des politiques AWS gérées par Secrets Manager

Consultez les détails des mises à jour des politiques AWS gérées pour Secrets Manager.

Modification	Description	Date	Version
SecretsManagerReadWrite – Mise à jour d'une politique existante	Cette politique a été mise à jour pour autoriser la description de l'accès à Amazon Redshift Serverless afin que les utilisateurs de la console puissent choisir un espace de noms Amazon Redshift Serverless lorsqu'ils créent un secret Amazon Redshift.	12 mars 2024	v5
SecretsManagerReadWrite – Mise à jour d'une politique existante	Cette politique a été mise à jour pour autoriser la description de l'accès aux clusters élastiques Amazon DocumentDB afin que les utilisateurs de la console puissent choisir un cluster élastique lorsqu'ils créent un secret Amazon DocumentDB.	12 septembre 2023	v4
SecretsManagerReadWrite – Mise à jour d'une politique existante	Cette politique a été mise à jour pour autoriser la description de l'accès à Amazon Redshift afin que les utilisate	24 juin 2020	v3

Modification	Description	Date	Version
	urs de la console puissent choisir un cluster Amazon Redshift lorsqu'il s créent un secret Amazon Redshift. La mise à jour a également ajouté de nouvelles autorisat ions pour autoriser l'accès en lecture à un compartiment Amazon S3 géré par AWS lequel sont stockés les modèles de fonctions de rotation Lambda.		
SecretsManagerReadWrite – Mise à jour d'une politique existante	Cette politique a été mise à jour pour autoriser la descripti on de l'accès aux clusters Amazon RDS afin que les utilisate urs de la console puissent choisir un cluster lorsqu'il s créent un secret Amazon RDS.	3 mai 2018	v2

Modification	Description	Date	Version
SecretsManagerReadWrite : nouvelle politique	Secrets Manager a créé une politique pour accorder les autorisations nécessaires à l'utilisation de la console avec un read/write accès complet à Secrets Manager.	04 avril 2018	v1

Déterminez qui est autorisé à accéder à vos AWS Secrets Manager secrets

Par défaut, les identités IAM ne disposent pas de l'autorisation d'accéder à des secrets. Lorsqu'il autorise l'accès à un secret, Secrets Manager évalue la stratégie basée sur une ressource attachée au secret, ainsi que toutes celles basées sur l'identité au rôle ou à l'utilisateur IAM envoyant la demande. Pour ce faire, Secrets Manager utilise un processus similaire à celui décrit dans la rubrique Identification d'une demande autorisée ou refusée dans le Guide de l'utilisateur IAM.

Lorsque plusieurs stratégies s'appliquent à une demande, Secrets Manager utilise une hiérarchie pour contrôler les autorisations :

1. Si une instruction dans une politique avec un `deny` explicite correspond à l'action de la demande et à la ressource :

   Le `deny` explicite remplace tout le reste et bloque l'action.
2. S'il n'y a pas de `deny` explicite, mais une instruction avec un `allow` explicite qui correspond à l'action de la demande et à la ressource :

   Le `allow` explicite accorde à l'action de la demande l'accès aux ressources de l'instruction.

   Si l'identité et le secret se trouvent dans deux comptes différents, il doit y avoir une `allow` politique de ressources pour le secret et une politique attachée à l'identité, sinon la AWS demande sera refusée. Pour de plus amples informations, veuillez consulter Accès intercomptes.
3. S'il n'y a pas d'instruction avec un `allow` explicite qui correspond à l'action de la demande et à la ressource :

AWS refuse la demande par défaut, ce que l'on appelle un refus implicite.

Pour afficher la stratégie basée sur une ressource pour un secret

- Effectuez l'une des actions suivantes :
  - Ouvrez la console Secrets Manager à l'adresse https://console.aws.amazon.com/secretsmanager/. Dans la page des détails de votre secret, dans la section Resource permissions (Autorisations d'accès aux ressources), sélectionnez Edit permissions (Modifier des autorisations).
  - Utilisez le AWS CLI pour appeler `get-resource-policy`ou le AWS SDK pour appeler `GetResourcePolicy`.

Pour déterminer qui a accès par le biais des stratégies basées sur l'identité

- Utilisez le simulateur de stratégie IAM. Consultez Test des stratégies IAM avec le simulateur de stratégies IAM

Accédez aux AWS Secrets Manager secrets depuis un autre compte

Pour autoriser les utilisateurs d'un compte à accéder aux secrets d'un autre compte (accès entre comptes), vous devez autoriser l'accès dans une stratégie de ressource et dans une stratégie d'identité. Cela diffère de l'octroi d'accès aux identités dans le même compte que le secret.

L'autorisation inter-comptes n'est effective que pour les opérations suivantes :

- CancelRotateSecret
- DeleteResourcePolicy
- DeleteSecret
- DescribeSecret
- GetRandomPassword
- GetResourcePolicy
- GetSecretValue
- ListSecretVersionIds
- PutResourcePolicy

- PutSecretValue
- RemoveRegionsFromReplication
- ReplicateSecretToRegions
- RestoreSecret
- RotateSecret
- StopReplicationToReplica
- TagResource
- UntagResource
- UpdateSecret
- UpdateSecretVersionStage
- ValidateResourcePolicy

Vous pouvez utiliser le `BlockPublicPolicy` paramètre avec l'PutResourcePolicyaction pour protéger vos ressources en empêchant l'accès public par le biais des politiques de ressources directement associées à vos secrets. Vous pouvez également utiliser IAM Access Analyzer pour vérifier l'accès entre comptes.

Vous devez aussi autoriser l'identité à utiliser la clé KMS avec laquelle le secret est chiffré. Cela est dû au fait que vous ne pouvez pas utiliser le Clé gérée par AWS (`aws/secretsmanager`) pour un accès entre comptes. Au lieu de cela, vous devez chiffrer votre secret avec une clé KMS que vous créez, puis y attacher une stratégie de clé. La création de clés KMS engendre des frais. Pour modifier la clé de chiffrement d'un secret, consultez the section called "Modification d'un secret".

> **Important**
>
> Les politiques basées sur les ressources qui `secretsmanager:PutResourcePolicy` octroient des autorisations donnent aux principaux, même à ceux d'autres comptes, la possibilité de modifier vos politiques basées sur les ressources. Cette autorisation permet aux principaux d'augmenter les autorisations existantes, par exemple en obtenant un accès administratif complet aux secrets. Nous vous recommandons d'appliquer le principe de l'accès le moins privilégié à vos politiques. Pour de plus amples informations, veuillez consulter Politiques basées sur les ressources.

Les exemples de stratégies suivants partent du principe que vous disposez d'un secret et d'une clé de chiffrement dans le Compte1, et d'une identité dans le Compte2 qui doit être autorisée à accéder à la valeur de secret.

Étape 1 : attacher une stratégie de ressources au secret dans Account1

- La politique suivante permet *ApplicationRole* *Account2* d'accéder au secret dans*Account1*. Pour utiliser cette stratégie, consultez the section called "Politiques basées sur les ressources".

  JSON

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

Étape 2 : ajouter une déclaration à la stratégie de clé pour la clé KMS dans Account1

- La déclaration de politique clé suivante permet *Account2* à *ApplicationRole* in d'utiliser la clé KMS *Account1* pour déchiffrer le secret dans*Account1*. Pour utiliser cette déclaration, ajoutez-la à la stratégie de clé de votre clé KMS. Consultez Modification d'une stratégie de clé pour de plus amples informations.

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

```
    "kms:DescribeKey"
  ],
  "Resource": "*"
}
```

Étape 3 : attacher une stratégie d'identité à l'identité dans Account2

- La politique suivante permet *ApplicationRole Account2* à l'entrée d'accéder au secret *Account1* et de déchiffrer la valeur du secret en utilisant la clé de chiffrement qui se trouve également dans*Account1*. Pour utiliser cette stratégie, consultez the section called "Politiques basées sur l'identité". Vous pouvez trouver l'ARN de votre secret dans la console Secrets Manager sur la page de détails du secret sous ARN du secret. Vous pouvez aussi appeler `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"
    }
  ]
}
```

Accédez aux secrets depuis un environnement sur site

Vous pouvez utiliser AWS Identity and Access Management Roles Anywhere pour obtenir des informations d'identification de sécurité temporaires dans IAM pour les charges de travail telles que les serveurs, les conteneurs et les applications qui s'exécutent en dehors de. AWS Vos charges de travail peuvent utiliser les mêmes politiques IAM et les mêmes rôles IAM que ceux que vous utilisez

avec les AWS applications pour accéder aux ressources. AWS Avec Rôles Anywhere IAM, vous pouvez utiliser Secrets Manager pour stocker et gérer les informations d'identification auxquelles peuvent accéder les ressources dans AWS , ainsi que les appareils sur site tels que des serveurs d'applications. Pour plus d'informations, veuillez consulter le Guide de l'utilisateur Rôles Anywhere IAM.

Protection des données dans AWS Secrets Manager

Le modèle de responsabilité AWS partagée de s'applique à la protection des données dans AWS Secrets Manager. Comme décrit dans ce modèle, AWS est chargé de protéger l'infrastructure mondiale qui gère tous les AWS Cloud. La gestion du contrôle de votre contenu hébergé sur cette infrastructure relève de votre responsabilité. Ce contenu comprend les tâches de configuration et de gestion de la sécurité des Services AWS que vous utilisez. Pour en savoir plus sur la confidentialité des données, consultez Questions fréquentes (FAQ) sur la confidentialité des données. Pour en savoir plus sur la protection des données en Europe, consultez le billet de blog Modèle de responsabilité partagée AWS et RGPD (Règlement général sur la protection des données) sur le Blog de sécuritéAWS .

Pour des raisons de protection des données, nous vous recommandons de protéger les Compte AWS informations d'identification et de configurer des comptes utilisateur individuels avec AWS Identity and Access Management (IAM). Ainsi, chaque utilisateur se voit attribuer uniquement les autorisations nécessaires pour exécuter ses tâches. Nous vous recommandons également de sécuriser vos données comme indiqué ci-dessous :

- Utilisez l'authentification multifactorielle (MFA) avec chaque compte.
- Utilisez le protocole SSL/TLS pour communiquer avec les ressources. AWS Secrets Manager prend en charge TLS 1.2 et 1.3 dans toutes les régions. Secrets Manager prend également en charge une option d'échange de clés post-quantiques hybrides pour le protocole de chiffrement réseau TLS (Transport Layer Security).
- Signez vos demandes programmatiques à Secrets Manager en utilisant un ID de clé d'accès et une clé d'accès secrète associée à un principal IAM. Vous pouvez également utiliser AWS Security Token Service (AWS STS) pour générer des informations d'identification de sécurité temporaires et signer les demandes.
- Configurez l'API et la journalisation de l'activité des utilisateurs avec AWS CloudTrail. Consultez the section called “Connectez-vous avec AWS CloudTrail ”.

- Si vous avez besoin de modules cryptographiques validés par la norme FIPS 140-2 pour accéder AWS via une interface de ligne de commande ou une API, utilisez un point de terminaison FIPS. Consultez the section called "Points de terminaison Secrets Manager".
- Si vous utilisez le AWS CLI pour accéder à Secrets Manager,the section called "Réduisez les risques liés AWS CLI à l'utilisation du pour stocker vos AWS Secrets Manager secrets".

Chiffrement au repos

Secrets Manager utilise le chiffrement via AWS Key Management Service (AWS KMS) pour protéger la confidentialité des données au repos. AWS KMS fournit un service de stockage et de chiffrement des clés utilisé par de nombreux AWS services. Chaque secret de Secrets Manager est chiffré à l'aide d'une clé de données unique. Chaque clé de données est protégée par une clé KMS. Vous pouvez utiliser le chiffrement par défaut avec la Clé gérée par AWS Secrets Manager pour le compte ou créer votre propre clé gérée par le client dans AWS KMS. En utilisant une clé gérée par le client, vous disposez de contrôles d'autorisation plus précis sur les activités de vos clés KMS. Pour de plus amples informations, veuillez consulter the section called "Chiffrement et déchiffrement de secret".

Chiffrement en transit

Secrets Manager fournit des points de terminaison sécurisés et privés pour le chiffrement des données en transit. Les points de terminaison sécurisés et privés permettent AWS de protéger l'intégrité des demandes d'API adressées à Secrets Manager. AWS exige que les appels d'API soient signés par l'appelant à l'aide de certificats X.509 et/ou d'une clé d'accès secrète de Secrets Manager. Cette exigence est énoncée dans Processus de signature Signature version 4 (Sigv4).

Si vous utilisez le AWS Command Line Interface (AWS CLI) ou l'un des AWS SDKs pour passer des appels AWS, vous configurez la clé d'accès à utiliser. Ces outils utilisent ensuite automatiquement la clé d'accès pour signer les demandes pour vous. Consultez the section called "Réduisez les risques liés AWS CLI à l'utilisation du pour stocker vos AWS Secrets Manager secrets".

Confidentialité du trafic inter-réseaux

AWS propose des options pour préserver la confidentialité lors du routage du trafic via des itinéraires réseau connus et privés.

Trafic entre les clients de service et sur site et les applications

> Vous disposez de deux options de connectivité entre votre réseau privé et AWS Secrets Manager :

- Une connexion AWS Site-to-Site VPN. Pour plus d'informations, consultez Qu'est-ce qu'un VPN de AWS site à site ?
- Une connexion AWS Direct Connect. Pour de plus amples informations, veuillez consulter Description de AWS Direct Connect

Trafic entre les AWS ressources d'une même région

Si vous souhaitez sécuriser le trafic entre Secrets Manager et les clients de l'API AWS, configurez un AWS PrivateLinkpour accéder de manière privée aux points de terminaison de l'API Secrets Manager.

Gestion des clés de chiffrement

Lorsque Secrets Manager doit chiffrer une nouvelle version des données secrètes protégées, Secrets Manager envoie une demande pour générer une nouvelle clé de données AWS KMS à partir de la clé KMS. Secrets Manager utilise cette clé de données pour le chiffrement d'enveloppe. Secrets Manager stocke la clé de données chiffrée avec le secret chiffré. Lorsque le secret doit être déchiffré, Secrets Manager demande de déchiffrer la clé AWS KMS de données. Secrets Manager utilise alors la clé de données déchiffrée pour déchiffrer le secret chiffré. Secrets Manager ne stocke jamais la clé de données sous forme non chiffrée et supprime la clé de la mémoire dès que possible. Pour de plus amples informations, veuillez consulter the section called “Chiffrement et déchiffrement de secret”.

Chiffrement et déchiffrement secrets dans AWS Secrets Manager

Secrets Manager utilise le chiffrement des enveloppes avec des AWS KMS clés et des clés de données pour protéger chaque valeur secrète. Chaque fois que la valeur secrète d'un secret change, Secrets Manager demande une nouvelle clé de données à AWS KMS pour le protéger. La clé de données est chiffrée sous une clé KMS et ensuite stockée dans les métadonnées du secret. Pour déchiffrer le secret, Secrets Manager déchiffre d'abord la clé de données chiffrée à l'aide de la clé KMS dans. AWS KMS

Secrets Manager n'utilise pas la clé KMS pour chiffrer la valeur du secret directement. Au lieu de cela, il utilise la clé KMS pour générer et chiffrer une symétrique AES (Advanced Encryption Standard) 256 bits Clé de données et utilise la clé de données pour chiffrer la valeur du secret. Secrets Manager utilise la clé de données en texte brut pour chiffrer la valeur secrète en dehors de AWS KMS, puis la supprime de la mémoire. Il stocke la copie chiffrée de la clé de données dans les métadonnées du secret.

Rubriques

- Choisir une AWS KMS clé
- Qu'est-ce qui est chiffré ?
- Processus de chiffrement et déchiffrement
- Autorisations pour la clé KMS
- Comment Secrets Manager utilise votre clé KMS
- Stratégie de clé de Clé gérée par AWS (aws/secretsmanager)
- Contexte de chiffrement de Secrets Manager
- Surveillez l'interaction de Secrets Manager avec AWS KMS

Choisir une AWS KMS clé

Lorsque vous créez un secret, vous pouvez choisir n'importe quelle clé de chiffrement symétrique gérée par le client dans la région Compte AWS et, ou vous pouvez utiliser le Clé gérée par AWS for Secrets Manager (`aws/secretsmanager`). Si vous choisissez le Clé gérée par AWS `aws/secretsmanager` et qu'il n'existe pas encore, Secrets Manager le crée et l'associe au secret. Vous pouvez utiliser la même clé KMS ou d'autres clés KMS pour chaque secret dans votre compte. Vous pouvez utiliser différentes clés KMS pour définir des autorisations personnalisées sur les clés d'un groupe de secrets, ou si vous souhaitez contrôler des opérations spécifiques pour ces clés. Secrets Manager ne prend en charge que les clés KMS de chiffrement symétriques. Si vous utilisez une clé KMS dans une boutique de clés externe, les opérations cryptographiques sur la clé KMS peuvent prendre plus de temps et être moins fiables et durables, car la demande doit être transmise à l'extérieur d' AWS.

Pour plus d'informations sur la modification de la clé de chiffrement d'un secret, consultez la section the section called “Modification de clé de chiffrement d'un secret”.

Lorsque vous modifiez la clé de chiffrement, Secrets Manager le chiffre à nouveau `AWSCURRENT` et `AWSPENDING` les `AWSPREVIOUS` versions utilisant la nouvelle clé. Pour ne pas vous empêcher d'accéder au secret, Secrets Manager crypte toutes les versions existantes avec la clé précédente. Cela signifie que vous pouvez déchiffrer `AWSCURRENT` `AWSPENDING` les `AWSPREVIOUS` versions avec la clé précédente ou la nouvelle clé. Si vous n'êtes pas `kms:Decrypt` autorisé à accéder à la clé précédente, lorsque vous modifiez la clé de chiffrement, Secrets Manager ne peut pas déchiffrer les versions secrètes pour les rechiffrer. Dans ce cas, les versions existantes ne sont pas rechiffrées.

Pour qu'il ne `AWSCURRENT` puisse être déchiffré que par la nouvelle clé de chiffrement, créez une nouvelle version du secret avec la nouvelle clé. Ensuite, pour pouvoir déchiffrer la version `AWSCURRENT` secrète, vous devez avoir l'autorisation d'utiliser la nouvelle clé.

Vous pouvez refuser l'autorisation Clé gérée par AWS `aws/secretsmanager` et exiger que les secrets soient chiffrés à l'aide d'une clé gérée par le client. Pour de plus amples informations, veuillez consulter the section called “Exemple : refuser une AWS KMS clé spécifique pour chiffrer des secrets”.

Pour trouver la clé KMS associée à un secret, consultez le secret dans la console ou appelez ListSecretsou DescribeSecret. Lorsque le secret est associé au Clé gérée par AWS for Secrets Manager (`aws/secretsmanager`), ces opérations ne renvoient pas d'identifiant de clé KMS.

Qu'est-ce qui est chiffré ?

Secrets Manager chiffre la valeur secrète, mais il ne chiffre pas les éléments suivants :

- Le nom et la description du secret
- Paramètres de rotation
- L'ARN de la clé KMS associée au secret
- Toutes les AWS étiquettes jointes

Processus de chiffrement et déchiffrement

Pour chiffrer la valeur de secret dans un secret, Secrets Manager utilise le processus suivant.

1. Secrets Manager appelle l' AWS KMS GenerateDataKeyopération en indiquant l'ID de la clé KMS associée au secret et en demandant une clé symétrique AES 256 bits. AWS KMS renvoie une clé de données en texte brut et une copie de cette clé de données chiffrée sous la clé KMS.
2. Secrets Manager utilise la clé de données en texte brut et l'algorithme Advanced Encryption Standard (AES) pour chiffrer la valeur secrète en dehors de. AWS KMS Il supprime la clé en texte brut de la mémoire dès que possible après l'avoir utilisée.
3. Secrets Manager stocke la clé de données chiffrée dans les métadonnées du secret afin qu'elle soit disponible pour déchiffrer la valeur du secret. Cependant, aucun des Secrets Manager ne APIs renvoie le secret chiffré ou la clé de données chiffrée.

Pour déchiffrer une valeur de secret chiffrée :

1. Secrets Manager lance l'opération de AWS KMS déchiffrement et transmet la clé de données chiffrée.
2. AWS KMS utilise la clé KMS comme secret pour déchiffrer la clé de données. Il renvoie la clé de données en texte brut.
3. Secrets Manager utilise la clé de données en texte brut pour déchiffrer la valeur du secret. Puis il supprime la clé de données de la mémoire dès que possible.

Autorisations pour la clé KMS

Lorsque Secrets Manager utilise une clé KMS dans les opérations de chiffrement, il agit au nom de l'utilisateur qui accède à la valeur de secret ou la met à jour. Vous pouvez accorder ces autorisations dans une politique IAM ou une stratégie de clé. Les opérations suivantes de Secrets Manager nécessitent AWS KMS des autorisations.

- CreateSecret
- GetSecretValue
- PutSecretValue
- UpdateSecret
- ReplicateSecretToRegions

Pour autoriser l'utilisation de la clé KMS uniquement pour les demandes provenant de Secrets Manager, dans la politique d'autorisations, vous pouvez utiliser la clé de ViaService condition kms : avec la `secretsmanager.<Region>.amazonaws.com` valeur.

Vous pouvez également utiliser les clés ou les valeurs du contexte de chiffrement comme condition d'utilisation de la clé KMS pour les opérations de chiffrement. Par exemple, vous pouvez utiliser un opérateur de condition de chaîne dans un document de stratégie IAM ou de clé, ou utiliser une contrainte d'octroi dans un octroi. La propagation des autorisations de clés KMS peut prendre jusqu'à cinq minutes. Pour de plus amples informations, veuillez consulter CreateGrant.

Comment Secrets Manager utilise votre clé KMS

Secrets Manager effectue les AWS KMS opérations suivantes à l'aide de votre clé KMS.

GenerateDataKey

Secrets Manager appelle l' AWS KMS GenerateDataKeyopération en réponse aux opérations suivantes du Gestionnaire de secrets.

- CreateSecret— Si le nouveau secret inclut une valeur secrète, Secrets Manager demande une nouvelle clé de données pour le chiffrer.
- PutSecretValue— Secrets Manager demande une nouvelle clé de données pour chiffrer la valeur secrète spécifiée.
- ReplicateSecretToRegions— Pour chiffrer le secret répliqué, Secrets Manager demande une clé de données pour la clé KMS dans la région de réplication.
- UpdateSecret— Si vous modifiez la valeur secrète ou la clé KMS, Secrets Manager demande une nouvelle clé de données pour chiffrer la nouvelle valeur secrète.

L'RotateSecretopération n'appelle pas`GenerateDataKey`, car elle ne modifie pas la valeur secrète. Toutefois, si `RotateSecret` invoque une fonction de rotation Lambda qui modifie la valeur du secret, son appel à l'opération `PutSecretValue` déclenche une requête `GenerateDataKey`.

Decrypt

Secrets Manager appelle l'opération Decrypt en réponse aux opérations Secrets Manager.

- GetSecretValueet BatchGetSecretValue— Secrets Manager déchiffre la valeur secrète avant de la renvoyer à l'appelant. Pour déchiffrer une valeur secrète chiffrée, Secrets Manager appelle l'opération AWS KMS Decrypt pour déchiffrer la clé de données chiffrée contenue dans le secret. Ensuite, il utilise la clé de données en texte brut pour déchiffrer la valeur du secret chiffrée. Pour les commandes par lots, Secrets Manager peut réutiliser la clé déchiffrée, de sorte que tous les appels n'aboutissent pas à une demande `Decrypt`.
- PutSecretValueet UpdateSecret— La plupart des `UpdateSecret` demandes `PutSecretValue` et ne déclenchent aucune `Decrypt` opération. Toutefois, lorsqu'une demande `PutSecretValue` ou `UpdateSecret` tente de modifier la valeur de secret dans une version existante d'un secret, Secrets Manager déchiffre la valeur du secret existante et la compare à la valeur du secret dans la demande afin de confirmer qu'elles sont identiques. Cette action permet de s'assurer que les opérations de Secrets Manager sont idempotentes. Pour déchiffrer une valeur secrète chiffrée, Secrets Manager appelle l'opération AWS KMS Decrypt pour déchiffrer la clé de données chiffrée contenue dans le secret. Ensuite, il utilise la clé de données en texte brut pour déchiffrer la valeur du secret chiffrée.

- ReplicateSecretToRegions— Secrets Manager déchiffre d'abord la valeur secrète dans la région principale avant de la rechiffrer avec la clé KMS dans la région de réplication.

Encrypt

Secrets Manager appelle l'opération Encrypt en réponse aux opérations suivantes de Secrets Manager :

- UpdateSecret— Si vous modifiez la clé KMS, Secrets Manager chiffre à nouveau la clé de données qui protège les versions `AWSCURRENT`AWSPREVIOUS, et `AWSPENDING` secrète avec la nouvelle clé.

DescribeKey

Secrets Manager lance l'DescribeKeyopération pour déterminer s'il convient de répertorier la clé KMS lorsque vous créez ou modifiez un secret dans la console Secrets Manager.

Validation de l'accès à la clé KMS

Lorsque vous établissez ou modifiez la clé KMS associée au secret, `GenerateDataKey` appelle les opérations Secrets Manager et `Decrypt` avec la clé KMS spécifiée. Ces appels confirment que l'appelant a l'autorisation d'utiliser la clé KMS pour ces opérations. Secrets Manager ignore les résultats de ces opérations ; il ne les utilise pas dans les opérations de chiffrement.

Vous pouvez identifier ces appels de validation, car la valeur de la clé `SecretVersionId` de contexte de chiffrement dans ces requêtes est `RequestToValidateKeyAccess`.

> **Note**
>
> Par le passé, les appels de validation Secrets Manager n'incluaient pas de contexte de chiffrement. Il est possible que vous trouviez des appels sans contexte de chiffrement dans les anciens AWS CloudTrail journaux.

Stratégie de clé de Clé gérée par AWS (**`aws/secretsmanager`**)

La politique clé du Clé gérée par AWS for Secrets Manager (`aws/secretsmanager`) autorise les utilisateurs à utiliser la clé KMS pour des opérations spécifiques uniquement lorsque Secrets Manager fait la demande au nom de l'utilisateur. La politique de clé n'autorise pas les utilisateurs à utiliser la clé KMS directement.

Cette politique de clé, comme les politiques de toutes les Clés gérées par AWS, est établie par le service. Vous ne pouvez pas modifier la politique de clé, mais vous pouvez l'afficher à tout moment. Pour plus d'informations, consultez Affichage d'une stratégie de clé.

Les instructions de politique de la politique de clé ont l'effet suivant :

- Elles autorisent les utilisateurs du compte à utiliser la clé KMS pour les opérations de chiffrement uniquement lorsque la demande vient de Secrets Manager en leur nom. La clé de condition `kms:ViaService` applique cette restriction.
- Permet au AWS compte de créer des politiques IAM qui permettent aux utilisateurs de consulter les propriétés des clés KMS et de révoquer les autorisations.
- Bien que Secrets Manager n'utilise pas d'octrois pour obtenir l'accès à la clé KMS, la stratégie autorise également Secrets Manager à créer des octrois pour la clé KMS au nom de l'utilisateur et autorise le compte à révoquer tous les octrois qui autorisent Secrets Manager à utiliser la clé KMS. Ce sont des éléments standard du document de politique pour un Clé gérée par AWS.

Voici une politique clé à titre d'exemple Clé gérée par AWS pour Secrets Manager.

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": "*"
    }
```

```
  ]
}
```

Contexte de chiffrement de Secrets Manager

Un contexte de chiffrement est un ensemble de paires clé-valeur contenant des données arbitraires non secrètes. Lorsque vous incluez un contexte de chiffrement dans une demande de chiffrement de données, lie AWS KMS cryptographiquement le contexte de chiffrement aux données chiffrées. Pour déchiffrer les données, vous devez transmettre le même contexte de chiffrement.

Dans ses requêtes GenerateDataKeyet Decrypt à AWS KMS, Secrets Manager utilise un contexte de chiffrement avec deux paires nom-valeur qui identifient le secret et sa version, comme illustré dans l'exemple suivant. Les noms ne varient pas, mais les valeurs de contexte de chiffrement combinées sont différentes pour chaque valeur de secret.

```
"encryptionContext": {
    "SecretARN": "arn:aws:secretsmanager:us-east-2:111122223333:secret:test-secret-
a1b2c3",
    "SecretVersionId": "EXAMPLE1-90ab-cdef-fedc-ba987SECRET1"
}
```

Vous pouvez utiliser le contexte de chiffrement pour identifier ces opérations cryptographiques dans les enregistrements et journaux d'audit, tels que AWS CloudTrailAmazon CloudWatch Logs, et comme condition d'autorisation dans les politiques et les autorisations.

Le contexte de chiffrement Secrets Manager se compose de deux paires nom-valeur.

- SecretARN – La première paire nom–valeur identifie le secret. La clé est `SecretARN`. La valeur est l'Amazon Resource Name (ARN) du secret.

  ```
  "SecretARN": "ARN of an Secrets Manager secret"
  ```

  Par exemple si l'ARN du secret est `arn:aws:secretsmanager:us-east-2:111122223333:secret:test-secret-a1b2c3`, le contexte de chiffrement inclura la paire suivante.

  ```
  "SecretARN": "arn:aws:secretsmanager:us-east-2:111122223333:secret:test-secret-
  a1b2c3"
  ```

- SecretVersionId— La deuxième paire nom-valeur identifie la version du secret. La clé est `SecretVersionId`. La valeur est l'ID de version.

  ```
  "SecretVersionId": "<version-id>"
  ```

  Par exemple si l'ID de version du secret est `EXAMPLE1-90ab-cdef-fedc-ba987SECRET1`, le contexte de chiffrement inclura la paire suivante.

  ```
  "SecretVersionId": "EXAMPLE1-90ab-cdef-fedc-ba987SECRET1"
  ```

Lorsque vous définissez ou modifiez la clé KMS d'un secret, Secrets Manager envoie GenerateDataKeyet déchiffre des demandes AWS KMS pour valider que l'appelant est autorisé à utiliser la clé KMS pour ces opérations. Il ignore les réponses ; il ne les utilise pas sur la valeur du secret.

Dans le cadre de ces requêtes de validation, la valeur de `SecretARN` est l'ARN réel du secret, mais la valeur de `SecretVersionId` est `RequestToValidateKeyAccess`, comme illustré dans l'exemple de contexte de chiffrement suivant. Cette valeur spéciale vous permet d'identifier les requêtes de validation dans les journaux et les pistes d'audit.

```
"encryptionContext": {
    "SecretARN": "arn:aws:secretsmanager:us-east-2:111122223333:secret:test-secret-
a1b2c3",
    "SecretVersionId": "RequestToValidateKeyAccess"
}
```

> **Note**
>
> Par le passé, les demandes de validation Secrets Manager n'incluaient pas de contexte de chiffrement. Il est possible que vous trouviez des appels sans contexte de chiffrement dans les anciens AWS CloudTrail journaux.

Surveillez l'interaction de Secrets Manager avec AWS KMS

Vous pouvez utiliser AWS CloudTrail Amazon CloudWatch Logs pour suivre les demandes que Secrets Manager envoie AWS KMS en votre nom. Pour plus d'informations sur la surveillance de l'utilisation des secrets, voir Surveillance des secrets.

GenerateDataKey

Lorsque vous créez ou modifiez la valeur secrète d'un secret, Secrets Manager envoie une GenerateDataKeydemande AWS KMS indiquant la clé KMS du secret.

L'événement qui enregistre l'opération `GenerateDataKey` est similaire à l'exemple d'événement suivant. La requête est appelée par `secretsmanager.amazonaws.com`. Les paramètres incluent le nom Amazon Resource Name (ARN) de la clé KMS du secret, un spécificateur de clé qui nécessite une clé de 256 bits, et le contexte de chiffrement qui identifie le secret et la version.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AROAIGDTESTANDEXAMPLE:user01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/user01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2018-05-31T23:23:41Z"
            }
        },
        "invokedBy": "secretsmanager.amazonaws.com"
    },
    "eventTime": "2018-05-31T23:23:41Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "secretsmanager.amazonaws.com",
    "userAgent": "secretsmanager.amazonaws.com",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "keySpec": "AES_256",
```

```
        "encryptionContext": {
            "SecretARN": "arn:aws:secretsmanager:us-east-2:111122223333:secret:test-
secret-a1b2c3",
            "SecretVersionId": "EXAMPLE1-90ab-cdef-fedc-ba987SECRET1"
        }
    },
    "responseElements": null,
    "requestID": "a7d4dd6f-6529-11e8-9881-67744a270888",
    "eventID": "af7476b6-62d7-42c2-bc02-5ce86c21ed36",
    "readOnly": true,
    "resources": [
        {
            "ARN": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "accountId": "111122223333",
            "type": "AWS::KMS::Key"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

Decrypt

Lorsque vous obtenez ou modifiez la valeur secrète d'un secret, Secrets Manager envoie une demande de déchiffrement AWS KMS pour déchiffrer la clé de données chiffrée. Pour les commandes par lots, Secrets Manager peut réutiliser la clé déchiffrée, de sorte que tous les appels n'aboutissent pas à une demande `Decrypt`.

L'événement qui enregistre l'opération `Decrypt` est similaire à l'exemple d'événement suivant. L'utilisateur est le principal de votre AWS compte qui accède à la table. Les paramètres incluent la clé de table chiffrée (sous forme de blob de texte chiffré) et le contexte de chiffrement qui identifie la table et le compte. AWS AWS KMS déduit l'ID de la clé KMS à partir du texte chiffré.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AROAIGDTESTANDEXAMPLE:user01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/user01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
```

```
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2018-05-31T23:36:09Z"
            }
        },
        "invokedBy": "secretsmanager.amazonaws.com"
    },
    "eventTime": "2018-05-31T23:36:09Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "secretsmanager.amazonaws.com",
    "userAgent": "secretsmanager.amazonaws.com",
    "requestParameters": {
        "encryptionContext": {
            "SecretARN": "arn:aws:secretsmanager:us-east-2:111122223333:secret:test-
secret-a1b2c3",
            "SecretVersionId": "EXAMPLE1-90ab-cdef-fedc-ba987SECRET1"
        }
    },
    "responseElements": null,
    "requestID": "658c6a08-652b-11e8-a6d4-ffee2046048a",
    "eventID": "f333ec5c-7fc1-46b1-b985-cbda13719611",
    "readOnly": true,
    "resources": [
        {
            "ARN": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "accountId": "111122223333",
            "type": "AWS::KMS::Key"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```


Encrypt

Lorsque vous modifiez la clé KMS associée à un secret, Secrets Manager envoie une demande de chiffrement AWS KMS pour rechiffrer les versions `AWSCURRENT`, `AWSPREVIOUS`, et `AWSPENDING` secrète avec la nouvelle clé. Lorsque vous répliquez un secret vers une autre région, Secrets Manager envoie également une demande Encrypt à AWS KMS.

L'événement qui enregistre l'opération `Encrypt` est similaire à l'exemple d'événement suivant. L'utilisateur est le principal de votre AWS compte qui accède à la table.


```
{
    "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"
}
```

Sécurité de l'infrastructure dans AWS Secrets Manager

En tant que service géré, AWS Secrets Manager il est protégé par la sécurité du réseau AWS mondial. Pour plus d'informations sur les services AWS de sécurité et sur la manière dont AWS l'infrastructure est protégée, consultez la section Sécurité du AWS cloud. Pour concevoir votre AWS environnement en utilisant les meilleures pratiques en matière de sécurité de l'infrastructure, consultez la section Protection de l'infrastructure dans le cadre AWS bien architecturé du pilier de sécurité.

L'accès à Secrets Manager via le réseau se fait via une AWS publication à APIs l'aide du protocole TLS. Les Secrets Manager APIs sont appelables depuis n'importe quel emplacement du réseau. Cependant, Secrets Manager prend en charge les stratégies d'accès basées sur les ressources, ce qui peut inclure des restrictions en fonction de l'adresse IP source. Vous pouvez également utiliser les politiques de ressources de Secrets Manager pour contrôler l'accès aux secrets à partir de points de terminaison de cloud privé virtuel (VPC) spécifiques ou spécifiques. VPCs En fait, cela isole l'accès réseau à un secret donné uniquement du VPC spécifique au sein AWS du réseau. Pour de plus amples informations, veuillez consulter the section called “Points de terminaison d’un VPC (AWS PrivateLink)”.

Utilisation d'un point de AWS Secrets Manager terminaison VPC

Nous recommandons d'exécuter la majeure partie de votre infrastructure sur des réseaux privés non accessibles à partir de l'Internet public. Vous pouvez établir une connexion privée entre votre VPC et Secrets Manager en créant un point de terminaison d'un VPC d'interface. Les points de terminaison de l'interface sont alimentés par AWS PrivateLinkune technologie qui vous permet d'accéder en privé à Secrets Manager APIs sans passerelle Internet, périphérique NAT, connexion VPN ou AWS Direct Connect connexion. Les instances de votre VPC n'ont pas besoin d'adresses IP publiques pour communiquer avec Secrets Manager. APIs Le trafic entre votre VPC et Secrets Manager ne quitte

pas le réseau AWS . Pour de plus amples informations, consultez Points de terminaison VPC (AWS PrivateLink) dans le Guide de l'utilisateur Amazon VPC.

Lorsque Secrets Manager effectue une rotation d'un secret à l'aide d'une fonction de rotation Lambda,par exemple un secret contenant des informations d'identification de base de données, la fonction Lambda envoie des requêtes à la fois à la base de données et à Secrets Manager. Lorsque vous activez la rotation automatique en utilisant la console, Secrets Manager crée la fonction Lambda dans le même VPC que votre base de données. Nous vous recommandons de créer un point de terminaison Secrets Manager dans le même VPC afin que les demandes de la fonction de rotation Lambda vers Secrets Manager ne quittent pas le réseau Amazon.

Si vous activez le DNS privé pour le point de terminaison, vous pouvez faire des demandes d'API à Secrets Manager en utilisant son nom DNS par défaut pour la Région, par exemple `secretsmanager.us-east-1.amazonaws.com`. Pour plus d'informations, consultez Accès à un service via un point de terminaison d'interface dans le Guide de l'utilisateur Amazon VPC.

Vous pouvez vérifier que les demandes adressées à Secrets Manager proviennent du VPC en incluant une condition dans vos stratégies d'autorisations. Pour de plus amples informations, veuillez consulter the section called "Exemple : autorisations et VPCs".

Vous pouvez utiliser AWS CloudTrail les journaux pour vérifier votre utilisation des secrets via le point de terminaison VPC.

Création d'un point de terminaison d'un VPC Secrets Manager

1. Consultez la section Création d'un point de terminaison d'interface dans le guide de l'utilisateur Amazon VPC. Utilisez le nom du service :`com.amazonaws.region.secretsmanager`.
2. Pour contrôler l'accès au point de terminaison, consultez la section Contrôler l'accès aux points de terminaison VPC à l'aide de politiques de point de terminaison.
3. Pour utiliser un IPv6 adressage à double pile, voirIPv4 et IPv6 accès.

Sous-réseaux partagés

Vous ne pouvez pas créer, décrire, modifier ou supprimer des points de terminaison d'un VPC dans des sous-réseaux qui sont partagés avec vous. Toutefois, vous pouvez utiliser les points de terminaison d'un VPC dans les sous-réseaux qui sont partagés avec vous. Pour plus d'informations sur le partage de VPC, consultez la section Partager votre VPC avec d'autres comptes du Guide de l'utilisateur d'Amazon Virtual Private Cloud.

Contrôlez l'accès aux API avec des politiques IAM

Si vous utilisez des politiques IAM pour contrôler l'accès en Services AWS fonction des adresses IP, vous devrez peut-être mettre à jour vos politiques pour inclure les plages d' IPv6 adresses. Ce guide explique les différences entre IPv4 et IPv6 et décrit comment mettre à jour vos politiques IAM pour prendre en charge les deux protocoles. La mise en œuvre de ces modifications vous permet de conserver un accès sécurisé à vos AWS ressources tout en assurant le support IPv6.

Qu'est-ce que c'est IPv6 ?

IPv6 est la norme IP de prochaine génération destinée à être remplacée à terme IPv4. La version précédente utilise un schéma d'adressage 32 bits pour prendre en charge 4,3 milliards d'appareils. IPv4 IPv6 utilise plutôt un adressage 128 bits pour prendre en charge environ 340 billions de billions de milliards de milliards de milliards de dollars (soit 2 appareils à la 128e puissance).

Pour plus d'informations, consultez la IPv6page Web VPC.

Voici des exemples d' IPv6 adresses :

```
2001:cdba:0000:0000:0000:0000:3257:9652 # This is a full, unabbreviated IPv6 address.
2001:cdba:0:0:0:0:3257:9652             # The same address with leading zeros in each
 group omitted
2001:cdba::3257:965                     # A compressed version of the same address.
```

Politiques IAM à double pile (IPv4 et IPv6)

Vous pouvez utiliser les politiques IAM pour contrôler l'accès à Secrets Manager APIs et empêcher les adresses IP situées en dehors de la plage configurée d'accéder à Secrets Manager APIs.

Le responsable des secrets. Le point de terminaison à double pile {region} .amazonaws.com pour Secrets Manager APIs prend en charge les deux et. IPv6 IPv4

Si vous devez prendre en charge IPv4 les deux IPv6, mettez à jour vos politiques de filtrage des adresses IP pour gérer IPv6 les adresses. Dans le cas contraire, il se peut que vous ne puissiez pas vous connecter à Secrets Manager IPv6.

Qui doit effectuer ce changement ?

Cette modification vous concerne si vous utilisez le double adressage avec des politiques contenant`aws:sourceIp`. Le double adressage signifie que le réseau prend en charge à la fois IPv4 et IPv6.

Si vous utilisez le double adressage, mettez à jour vos politiques IAM qui utilisent actuellement des adresses de IPv4 format pour inclure les adresses de IPv6 format.

Qui ne devrait pas effectuer ce changement ?

Cette modification ne vous concerne pas si vous utilisez uniquement des IPv4 réseaux.

Ajout IPv6 à une politique IAM

Les politiques IAM utilisent la clé de `aws:SourceIp` condition pour contrôler l'accès à partir d'adresses IP spécifiques. Si votre réseau utilise le double adressage (IPv4 et IPv6), mettez à jour vos politiques IAM pour inclure les plages d' IPv6 adresses.

Dans l'`Condition`élément de vos politiques, utilisez les `NotIpAddress` opérateurs `IpAddress` et pour les conditions d'adresse IP. N'utilisez pas d'opérateurs de chaîne, car ils ne peuvent pas gérer les différents formats d' IPv6 adresse valides.

Ces exemples utilisent`aws:SourceIp`. Pour VPCs, utilisez `aws:VpcSourceIp` plutôt.

Ce qui suit indique le refus d'accès AWS basé sur la politique de référence IP source du guide de l'utilisateur IAM. L'`Condition`élément `NotIpAddress` in the to répertorie deux plages d' IPv4 adresses, `192.0.2.0/24` et`203.0.113.0/24`, auxquelles l'accès à l'API sera refusé.

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"
```

```
            }
        }
    }
}
```

Pour mettre à jour cette politique, modifiez l'`Condition`élément afin d'inclure les plages d' IPv6adresses `2001:DB8:1234:5678::/64` et`2001:cdba:3257:8593::/64`.

> **Note**
>
> Ne supprimez pas les IPv4 adresses existantes. Ils sont nécessaires pour assurer la rétrocompatibilité.

```
"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"
                }
            }
```

Pour mettre à jour cette politique pour un VPC, utilisez `aws:VpcSourceIp` au lieu de : `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"
            }
        }
```

Vérifier le support de vos clients IPv6

Si vous utilisez le secretsmanager. Point de terminaison {region} .amazonaws.com, vérifiez que vous pouvez vous y connecter. Les étapes suivantes décrivent comment effectuer la vérification.

Cet exemple utilise Linux et la version 8.6.0 de curl et utilise le AWS Secrets Manager service qui a IPv6 activé les points de terminaison situés sur le point de terminaison amazonaws.com.

> **Note**
>
> Le responsable des secrets. {region} .amazonaws.com diffère de la convention de dénomination classique à double pile. Pour une liste complète des points de terminaison de Secrets Manager, consultezAWS Secrets Manager points de terminaison. Remplacez-le Région AWS par la même région que celle où se trouve votre service. Dans cet exemple, nous utilisons le point de `us-east-1` terminaison de l'est des États-Unis (Virginie du Nord).

1. Déterminez si le point de terminaison est résolu avec une IPv6 adresse à l'aide de la `dig` commande suivante.

   ```
   $ dig +short AAAA secretsmanager.us-east-1.amazonaws.com

   > 2600:1f18:e2f:4e05:1a8a:948e:7c08:c1c3
   ```

2. Déterminez si le réseau client peut établir une IPv6 connexion à l'aide de la `curl` commande suivante. Un code de réponse 404 signifie que la connexion a réussi, tandis qu'un code de réponse 0 signifie que la connexion a échoué.

   ```
   $ 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
   ```

Si une adresse IP distante a été identifiée mais que le code de réponse ne l'est pas`0`, une connexion réseau a été établie avec succès avec le terminal à l'aide de IPv6. L'adresse IP distante doit être une IPv6 adresse car le système d'exploitation doit sélectionner le protocole valide pour le client.

Si l'adresse IP distante est vide ou si le code de réponse l'est`0`, le réseau client ou le chemin réseau vers le point de IPv4 terminaison est uniquement disponible. Vous pouvez vérifier cette configuration à l'aide de la `curl` commande suivante.

```
$ 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
```

Résilience dans AWS Secrets Manager

AWS construit l'infrastructure mondiale autour Régions AWS des zones de disponibilité. Régions AWS fournissent plusieurs zones de disponibilité physiquement séparées et isolées, connectées à un réseau à faible latence, à haut débit et hautement redondant. Avec les zones de disponibilité, vous pouvez concevoir et exploiter des applications et des bases de données qui basculent automatiquement d'une zone à l'autre sans interruption. Les zones de disponibilité vous permettent d'être plus hautement disponibles, plus tolérants aux pannes et plus évolutifs que les infrastructures traditionnelles à un ou plusieurs centres de données.

Pour plus d'informations sur la résilience et la reprise après sinistre, reportez-vous à Reliability Pillar — AWS Well-Architected Framework.

Pour plus d'informations sur les zones de disponibilité Régions AWS et les zones de disponibilité, consultez la section Infrastructure AWS globale.

TLS post-quantique

Secrets Manager prend également en charge une option d'échange de clés post-quantiques hybrides pour le protocole de chiffrement réseau TLS (Transport Layer Security). Vous pouvez utiliser cette option TLS lorsque vous vous connectez aux points de terminaison de l'API Secrets Manager. Nous offrons cette fonctionnalité avant la standardisation des algorithmes post-quantiques afin que vous puissiez commencer à tester l'effet de ces protocoles d'échange de clés sur les appels Secrets

Manager. Ces fonctionnalités d'échange de clés post-quantiques hybrides optionnelles sont au moins aussi sécurisées que le chiffrement TLS que nous utilisons aujourd'hui et sont susceptibles d'offrir des avantages supplémentaires en matière de sécurité. Cependant, elles affectent la latence et le débit par rapport aux protocoles d'échange de clés classiques utilisés aujourd'hui. L'agent Secrets Manager utilise l'échange de clés ML-KEM post-quantique comme échange de clés ayant la priorité la plus élevée par défaut.

Afin de protéger les données chiffrées aujourd'hui contre d'éventuelles attaques futures, AWS participe avec la communauté cryptographique au développement d'algorithmes résistants aux quanta ou post-quantiques. Nous avons mis en place des suites hybrides de chiffrement d'échange de clés post-quantiques dans les points de terminaison Secrets Manager. Ces suites de chiffrement hybrides, qui combinent des éléments classiques et post-quantiques, garantissent que votre connexion TLS est au moins aussi forte qu'avec les suites de chiffrement classiques. Cependant, étant donné que les caractéristiques de performance et les exigences de bande passante des suites de chiffrement hybrides sont différentes de celles des mécanismes d'échange de clés classiques, nous vous recommandons de les tester sur vos appels d'API.

Secrets Manager prend en charge le protocole PQTLS dans toutes les régions, à l'exception de la Chine.

Configurer le TLS post-quantique hybride

1. Ajoutez le client de moteur d'exécution AWS commun à vos dépendances Maven. Nous vous recommandons d'utiliser la dernière version disponible. Par exemple, cette instruction ajoute la version 2.20.0.

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>aws-crt-client</artifactId>
  <version>2.20.0</version>
</dependency>
```

2. Ajoutez le AWS kit SDK for Java 2.x à votre projet et initialisez-le. Activez les suites de chiffrement post-quantique hybrides sur votre client HTTP.

```
SdkAsyncHttpClient awsCrtHttpClient = AwsCrtAsyncHttpClient.builder()
            .postQuantumTlsEnabled(true)
            .build();
```

3. Créez le client asynchrone Secrets Manager.

```
SecretsManagerAsyncClient SecretsManagerAsync = SecretsManagerAsyncClient.builder()
            .httpClient(awsCrtHttpClient)
            .build();
```

Lorsque vous appelez des opérations d'API Secrets Manager, vos appels sont transmis au point de terminaison Secrets Manager à l'aide de TLS post-quantique hybride.

Pour de plus amples informations sur l'utilisation de TLS post-quantique hybride dans, consultez :

- AWS SDK for Java 2.x Guide du développeur et article de blog AWS SDK for Java 2.x publié.
- Présentation de s2n-tls, une nouvelle implémentation Open Source de TLS et utilisation de s2n-tls.
- Cryptographie post-quantique à l'Institut américain des normes et de la technologie (NIST).
- Méthodes d'encapsulation de clés post-quantiques hybrides (PQ KEM) pour Transport Layer Security 1.2 (TLS).

Le TLS post-quantique pour Secrets Manager est disponible dans toutes les sauf pour Régions AWS la Chine.

Résolution des problèmes AWS Secrets Manager

Utilisez ces informations pour identifier et résoudre les problèmes courants que vous pouvez rencontrer lors de l'utilisation des rôles Secrets Manager.

Pour les problèmes liés à la rotation, consultez the section called "Résolution des problèmes de rotation".

Rubriques

- Messages « Accès refusé »
- « Access denied » (« Accès refusé ») pour les informations d'identification de sécurité temporaires
- Les modifications que j'apporte ne sont pas toujours visibles immédiatement.
- « Cannot generate a data key with an asymmetric KMS key » (« Impossible de générer une clé de données avec une clé KMS asymétrique ») lors de la création d'un secret
- Une opération AWS CLI ou AWS SDK ne trouve pas mon secret à partir d'un ARN partiel
- Ce secret est géré par un AWS service, et vous devez utiliser ce service pour le mettre à jour.
- L'importation du module Python échoue lors de l'utilisation Transform: AWS::SecretsManager-2024-09-16

Messages « Accès refusé »

Lorsque vous effectuez un appel d'API tel que GetSecretValue ou CreateSecret vers Secrets Manager, vous devez disposer des autorisations IAM pour effectuer cet appel. Lorsque vous utilisez la console, celle-ci effectue les mêmes appels d'API en votre nom. Vous devez donc également disposer des autorisations IAM. Un administrateur peut accorder des autorisations en attachant une politique IAM à votre utilisateur IAM ou à un groupe dont vous êtes membre. Si les déclarations de politique qui accordent ces autorisations incluent des conditions, telles que time-of-day des restrictions d'adresse IP, vous devez également respecter ces exigences lorsque vous envoyez la demande. Pour plus d'informations sur l'affichage ou la modification de politiques pour un utilisateur, un groupe ou un rôle IAM, consultez Utilisation de politiques dans le Guide de l'utilisateur IAM. Pour plus d'informations sur les autorisations requises pour Secrets Manager, veuillez consulter the section called "Authentification et contrôle d'accès".

Si vous signez des demandes d'API manuellement, sans utiliser le AWS SDKs, vérifiez que vous avez correctement signé la demande.

« Access denied » (« Accès refusé ») pour les informations d'identification de sécurité temporaires

Vérifiez que l'utilisateur ou le rôle IAM que vous utilisez pour effectuer la demande dispose des autorisations appropriées. Les autorisations pour les informations d'identification de sécurité temporaires sont dérivées d'un utilisateur ou d'un rôle IAM. Cela signifie que les autorisations sont limitées à celles qui sont accordées à l'utilisateur ou au rôle IAM. Pour plus d'informations sur la manière dont les autorisations pour les informations d'identification de sécurité temporaires sont déterminées, consultez Contrôle des autorisations affectées aux informations d'identification de sécurité temporaires dans le Guide de l'utilisateur IAM.

Vérifiez que vos demandes sont signées correctement et que la demande est correctement formée. Pour plus de détails, consultez la documentation du kit de développement logiciel correspondant au SDK de votre choix ou l'utilisation d'informations d'identification de sécurité temporaires pour demander l'accès aux AWS ressources dans le guide de l'utilisateur IAM.

Vérifiez que vos informations d'identification de sécurité temporaires ne sont pas arrivées à expiration. Pour plus d'informations, consultez Obtention d'informations d'identification temporaires de sécurité dans le Guide de l'utilisateur IAM.

Pour plus d'informations sur les autorisations requises pour Secrets Manager, veuillez consulter the section called “Authentification et contrôle d’accès”.

Les modifications que j'apporte ne sont pas toujours visibles immédiatement.

Secrets Manager utilise un modèle de calcul distribué appelé cohérence éventuelle (eventual consistency). Toute modification que vous apportez dans Secrets Manager (ou dans d'autres AWS services) met du temps à être visible depuis tous les points de terminaison possibles. Une partie du retard s'explique par le temps requis pour envoyer les données d'un serveur à un autre, d'une zone de réplication à une autre et d'une région à une autre dans le monde entier. Secrets Manager utilise également la mise en cache pour améliorer les performances mais, dans certains cas, cela peut ralentir le processus. La modification peut ne pas être visible tant que les données mises en cache précédemment n'arrivent pas à expiration.

Concevez vos applications globales de sorte qu'elles tiennent compte de ces retards potentiels. Assurez-vous également qu'elles fonctionnent comme prévu, même lorsqu'une modification effectuée à un emplacement n'est pas visible instantanément à un autre.

Pour plus d'informations sur la manière dont certains autres AWS services sont affectés par une éventuelle cohérence, voir :

- Gestion de la cohérence des données dans le Guide du développeur de base de données Amazon Redshift
- Modèle de cohérence de données Amazon S3 dans le Guide de l'utilisateur Amazon Simple Storage Service.
- Assurer la cohérence lors de l'utilisation d'Amazon S3 et Amazon EMR pour les flux de travail ETL dans le blog AWS sur les Big Data
- Cohérence EC2 éventuelle d'Amazon dans la référence EC2 d'API Amazon

« Cannot generate a data key with an asymmetric KMS key » (« Impossible de générer une clé de données avec une clé KMS asymétrique ») lors de la création d'un secret

Secrets Manager utilise une clé KMS de chiffrement symétrique associée à un secret pour générer une clé de données pour chaque valeur de secret. Vous ne pouvez pas utiliser de clés KMS asymétriques. Assurez-vous d'utiliser une clé KMS de chiffrement symétrique à la place d'une clé KMS asymétrique. Pour obtenir des instructions, consultez Identification des clés KMS asymétriques.

Une opération AWS CLI ou AWS SDK ne trouve pas mon secret à partir d'un ARN partiel

Dans de nombreux cas, Secrets Manager peut trouver votre secret à partir d'une partie d'un ARN plutôt que de l'ARN complet. Toutefois, si le nom de votre secret se termine par un trait d'union suivi de six caractères, Secrets Manager risque de ne pas pouvoir trouver le secret à partir d'une partie seulement d'un ARN. Nous vous recommandons plutôt d'utiliser l'ARN complet ou le nom du secret.

Plus d'informations

Secrets Manager inclut six caractères aléatoires à la fin du nom du secret pour garantir que l'ARN secret est unique. Si le secret d'origine est supprimé, puis qu'un nouveau secret est créé avec le

même nom, les deux secrets sont différents en ARNs raison de ces caractères. Les utilisateurs ayant accès à l'ancien secret n'ont pas automatiquement accès au nouveau secret car il ARNs est différent.

Secrets Manager construit un ARN pour un secret avec région, compte, nom de secret, puis un trait d'union et six autres caractères, comme suit :

```
arn:aws:secretsmanager:us-east-2:111122223333:secret:SecretName-abcdef
```

Si le nom de votre secret se termine par un trait d'union et six caractères, l'utilisation d'une partie seulement de l'ARN peut donner l'impression à Secrets Manager que vous spécifiez un ARN complet. Par exemple, vous pourriez avoir un secret nommé `MySecret-abcdef` avec l'ARN

`arn:aws:secretsmanager:us-east-2:111122223333:secret:MySecret-abcdef-nutBrk`

Si vous appelez l'opération suivante, qui n'utilise qu'une partie de l'ARN du secret, Secrets Manager risque de ne pas trouver le secret.

```
$ aws secretsmanager describe-secret --secret-id arn:aws:secretsmanager:us-
east-2:111122223333:secret:MySecret-abcdef
```

Ce secret est géré par un AWS service, et vous devez utiliser ce service pour le mettre à jour.

Si ce message s'affiche alors que vous essayez de modifier un secret, celui-ci ne peut être mis à jour qu'à l'aide du service de gestion indiqué dans le message. Pour de plus amples informations, veuillez consulter Secrets gérés par d'autres services.

Pour déterminer qui gère un secret, vous pouvez vérifier le nom du secret. Les secrets gérés par d'autres services sont préfixés par l'ID de ce service. Ou, dans le AWS CLI, appelez describe-secret, puis passez en revue le champ. `OwningService`

L'importation du module Python échoue lors de l'utilisation `Transform: AWS::SecretsManager-2024-09-16`

Si vous utilisez Transform : `AWS::SecretsManager-2024-09-16` et que vous rencontrez des échecs d'importation du module Python lors de l'exécution de votre fonction Lambda de rotation, le problème est probablement dû à une valeur incompatible`Runtime`. Avec cette version de

transformation, AWS CloudFormation gère pour vous la version d'exécution, le code et les fichiers d'objets partagés. Vous n'avez pas besoin de les gérer vous-même.

AWS Secrets Manager quotas

Les quotas de lecture de Secrets Manager APIs sont élevés, tandis APIs que les plans de contrôle les moins fréquemment appelés ont des quotas TPS inférieurs. Nous vous recommandons d'éviter d'appeler `PutSecretValue` ou `UpdateSecret` à un rythme soutenu, soit plus d'une fois toutes les 10 minutes. Lorsque vous appelez `PutSecretValue` ou `UpdateSecret` pour mettre à jour la valeur sécrète, Secrets Manager crée une nouvelle version du secret. Secrets Manager supprime les versions non marquées lorsqu'il y en a plus de 100, mais il ne supprime pas les versions créées il y a moins de 24 heures. Si vous mettez à jour la valeur secrète plus d'une fois toutes les 10 minutes, vous créez plus de versions que Secrets Manager ne peut en supprimer, et vous atteindrez ainsi le quota pour les versions secrètes.

Vous pouvez exploiter plusieurs régions dans votre compte, et chaque quota est spécifique à une région bien déterminée.

Lorsqu'une application Compte AWS utilise un secret appartenant à un autre compte, on parle de demande entre comptes. Pour les demandes croisées entre comptes, Secrets Manager limite le compte de l'identité qui effectue les demandes, et non pas le compte qui possède la clé KMS. Par exemple, si un utilisateur du compte A utilise un secret dans le compte B, l'utilisation du secret s'applique uniquement aux quotas du compte A.

Quotas de Secrets Manager

Nom	Par défaut	Ajusta	Description
Taux combiné de demandes DeleteResourcePolicy GetResourcePolicy, PutResourcePolicy, et ValidateResourcePolicy d'API	Chaque région prise en charge : 50 par seconde	Non	Le nombre maximum de transactions par seconde pour DeleteResourcePolicy, GetResourcePolicy PutResourcePolicy, et les demandes ValidateResourcePolicy d'API combinées.
Taux combiné de demandes PutSecretValue RemoveRegionsFromReplication, ReplicateSecretToRegion, StopRepli	Chaque région prise en charge : 50 par seconde	Non	Le nombre maximum de transactions par seconde pour PutSecretValue,

Nom	Par défaut	Ajusta	Description
cationToReplica, UpdateSecret, et UpdateSecretVersionStage d'API			RemoveRegionsFromReplication, ReplicateSecretToRegion, StopReplicationToReplica UpdateSecret, et les demandes UpdateSecretVersionStage d'API combinées.
Taux combiné de demandes RestoreSecret d'API	Chaque région prise en charge : 50 par seconde	Non	Le nombre maximal de transactions par seconde pour les demandes RestoreSecret d'API.
Taux combiné de demandes d' CancelRotateSecret API RotateSecret et de demandes d'API	Chaque région prise en charge : 50 par seconde	Non	Le nombre maximum de transactions par seconde pour RotateSecret les demandes CancelRotateSecret d'API combinées.
Taux combiné de demandes d' UntagResource API TagResource et de demandes d'API	Chaque région prise en charge : 50 par seconde	Non	Le nombre maximum de transactions par seconde pour TagResource les demandes UntagResource d'API combinées.
Taux de demandes BatchGetSecretValue d'API	Chaque région prise en charge : 100 par seconde	Non	Le nombre maximal de transactions par seconde pour les demandes BatchGetSecretValue d'API.

Nom	Par défaut	Ajusta	Description
Taux de demandes CreateSecret d'API	Chaque région prise en charge : 50 par seconde	Non	Le nombre maximal de transactions par seconde pour les demandes CreateSecret d'API.
Taux de demandes DeleteSecret d'API	Chaque région prise en charge : 50 par seconde	Non	Le nombre maximal de transactions par seconde pour les demandes DeleteSecret d'API.
Taux de demandes DescribeSecret d'API	Chaque région prise en charge : 40 000 par seconde	Non	Le nombre maximal de transactions par seconde pour les demandes DescribeSecret d'API.
Taux de demandes GetRandom Password d'API	Chaque région prise en charge : 50 par seconde	Non	Le nombre maximal de transactions par seconde pour les demandes GetRandomPassword d'API.
Taux de demandes GetSecretValue d'API	Chaque Région prise en charge : 10 000 par seconde	Non	Le nombre maximal de transactions par seconde pour les demandes GetSecretValue d'API.
Taux de demandes ListSecretVersionI ds d'API	Chaque région prise en charge : 50 par seconde	Non	Le nombre maximal de transactions par seconde pour les demandes ListSecretVersionIds d'API.
Taux de demandes ListSecrets d'API	Chaque région prise en charge : 100 par seconde	Non	Le nombre maximal de transactions par seconde pour les demandes ListSecrets d'API.

Nom	Par défaut	Ajusta	Description
Taille des politiques basées sur une ressource	Chaque région prise en charge : 20 480	Non	Nombre maximum de caractères attaché à un secret dans une stratégie d'autorisation basée sur une ressource.
Taille de la valeur secrète	Chaque région prise en charge : 65 536 octets	Non	Taille maximale d'une valeur secrète cryptée. Si la valeur secrète est une chaîne de caractère s, il s'agit du nombre de caractères autorisés dans la valeur secrète.
Secrets	Chaque région prise en charge : 500 000	Non	Le nombre maximum de secrets dans chaque AWS région de ce AWS compte.
Étiquettes intermédiaires attachées à toutes les versions d'un secret	Chaque Région prise en charge : 20	Non	Nombre maximum d'étiquettes de mise à disposition attachées à toutes les versions d'un secret.
Versions par secret	Chaque Région prise en charge : 100	Non	Nombre maximum de versions d'un secret.

Ajouter de nouvelles tentatives à votre application

Votre AWS client risque de voir ses appels à Secrets Manager échouer en raison de problèmes imprévus côté client. Les appels peuvent aussi échouer en raison de la limitation de taux de la part de Secrets Manager Si vous dépassez le quota de demandes d'API, Secrets Manager limite la demande. La requête pourtant valide a été rejetée et signale qu'il s'agit d'une erreur throttling. Pour

les deux types d'échecs, nous vous recommandons de réessayer l'appel après une courte attente. C'est ce qu'on appelle une stratégie de backoff et de nouvelle tentative.

Si vous rencontrez les erreurs suivantes, vous pouvez ajouter des nouvelles tentatives à votre code d'application :

Erreurs transitoires et exceptions

- `RequestTimeout`
- `RequestTimeoutException`
- `PriorRequestNotComplete`
- `ConnectionError`
- `HTTPClientError`

Limitation côté service et erreurs de limitation et exceptions

- `Throttling`
- `ThrottlingException`
- `ThrottledException`
- `RequestThrottledException`
- `TooManyRequestsException`
- `ProvisionedThroughputExceededException`
- `TransactionInProgressException`
- `RequestLimitExceeded`
- `BandwidthLimitExceeded`
- `LimitExceededException`
- `RequestThrottled`
- `SlowDown`

Pour plus d'informations, ainsi que des exemples de code, sur les nouvelles tentatives, le backoff exponentiel et l'instabilité, consultez les ressources suivantes :

- Backoff exponentiel et Jitter
- Délais d'attente, tentatives et backoff avec instabilité

- Rétentatives d'erreur et ralentissement exponentiel. AWS

Historique du document

Le tableau suivant décrit les modifications importantes apportées à la documentation depuis la dernière version d' AWS Secrets Manager. Pour recevoir les notifications de mise à jour de cette documentation, abonnez-vous à un flux RSS.

Modification	Description	Date
Ajout de la prise en charge des balises de répartition des coûts	Secrets Manager prend désormais en charge les balises de répartition des coûts, ce qui permet aux clients de classer et de suivre les coûts par département, équipe ou application. Pour plus d'informations, consultez Utilisation des étiquettes d'allocation des coûts avec AWS Secrets Manager.	27 mai 2025
Support supplémentaire IPv6 et double pile	Secrets Manager prend désormais en charge les points de terminaison Dual-Stack. Voir IPv4 et IPv6 accéder pour plus d'informations.	20 décembre 2024
Passage de Secrets Manager à une politique AWS gérée	La politique `SecretsManagerReadWrite` gérée inclut désormais les `redshift-serverless` autorisations. Pour plus d'informations, consultez la section Politique AWS gérée pour AWS Secrets Manager	12 mars 2024

Mises à jour antérieures

Le tableau suivant décrit les modifications importantes apportées dans chaque version du Guide de AWS Secrets Manager l'utilisateur avant février 2024.

Modification	Description	Date
Disponibilité générale	Il s'agit de la première version publique d'Secrets Manager.	4 avril 2018

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.