Manual do usuário

AWS Secrets Manager

AWS Secrets Manager: Manual do usuário

Table of Contents

O que é AWS Secrets Manager?

AWS Secrets Manager ajuda você a gerenciar, recuperar e alternar credenciais de banco de dados, credenciais de aplicativos, OAuth tokens, chaves de API e outros segredos ao longo de seus ciclos de vida. Muitos AWS serviços armazenam e usam segredos no Secrets Manager.

O Secrets Manager ajuda você a melhorar seu procedimento de segurança, porque você não precisa mais de credenciais de codificação rígida no código-fonte da aplicação. Armazenar as credenciais no Secrets Manager evita um possível comprometimento por qualquer pessoa que possa inspecionar sua aplicação ou os componentes. Você substitui credenciais de codificação rígida com uma chamada de runtime ao serviço Secrets para recuperar as credenciais dinamicamente quando necessário.

Com o Secrets Manager, é possível configurar uma programação de alternância automática para seus segredos. Isso permite que você substitua os segredos de longo prazo por outros de curto prazo, reduzindo significativamente o risco de comprometimento. Como as credenciais não são mais armazenadas na aplicação, a alternância das credenciais não exige mais a atualização das aplicações e a implantação de alterações nos clientes da aplicação.

Para outros tipos de segredos que podem ser usados em sua organização:

- AWS credenciais — AWS Identity and Access ManagementRecomendamos.
- Chaves de criptografia: recomendamos o AWS Key Management Service.
- Chaves SSH — Recomendamos o Amazon EC2 Instance Connect.
- Chaves privadas e certificados: recomendamos o AWS Certificate Manager.

Conceitos básicos do Secrets Manager

Se você é novo no Secrets Manager, comece com um dos tutoriais a seguir:

- the section called “Substituir segredos codificados ”
- the section called “Substituir credenciais de banco de dados codificadas ”
- the section called “Alternância de usuários alternados”
- the section called “Alternâncias de usuário único”

Outras tarefas que podem ser realizadas com segredos:

- Gerenciar segredos
- Controlar o acesso a seus segredos
- Obter segredos
- Alternar segredos
- Monitorar segredos
- Monitorar segredos para conformidade
- Crie segredos em AWS CloudFormation

Compatibilidade com padrões

AWS Secrets Manager foi submetido à auditoria de vários padrões e pode fazer parte de sua solução quando você precisar obter a certificação de conformidade. Para obter mais informações, consulte Validação de conformidade.

Preços

Ao usar o Secrets Manager, você paga somente pelo que usa, e não há taxas mínimas ou de configuração. Não há cobrança para segredos que são marcados para exclusão. Para obter a lista de preços atual completa, consulte Definição de preço do AWS Secrets Manager. Para monitorar seus custos, consulte the section called “Monitorar custos do Secrets Manager”.

Você pode usar o Chave gerenciada pela AWS `aws/secretsmanager` que o Secrets Manager cria para criptografar seus segredos gratuitamente. Se você criar suas próprias chaves KMS para criptografar seus segredos, AWS cobrará a taxa atual AWS KMS . Para obter mais informações, consulte AWS Key Management Service Preço.

Quando você ativa a rotação automática (exceto a rotação gerenciada), o Secrets Manager usa uma AWS Lambda função para girar o segredo, e você é cobrado pela função de rotação na taxa Lambda atual. Para obter mais informações, consulte AWS Lambda Preço.

Se você ativar AWS CloudTrail em sua conta, poderá obter registros das chamadas de API que o Secrets Manager envia. O Secrets Manager registra todos os eventos como eventos de gerenciamento. AWS CloudTrail armazena gratuitamente a primeira cópia de todos os eventos de gerenciamento. No entanto, se você habilitar as notificações, poderão ser cobradas taxas pelo armazenamento de logs do Amazon S3 e pelo uso do Amazon SNS. Além disso, se você configurar

trilhas adicionais, as cópias adicionais de eventos de gerenciamento poderão ter custos. Para obter mais informações, consulte Preços do AWS CloudTrail.

É possível usar tags de alocação de custos no Secrets Manager para rastrear e categorizar despesas associadas a segredos ou a projetos específicos. Para obter mais informações, consulte the section called “Marcação de segredos do ” este guia e Usando tags de alocação de AWS custos no Guia do AWS Billing usuário.

Acesso AWS Secrets Manager

É possível trabalhar com o Secrets Manager de qualquer uma das seguintes formas:

- Console do Secrets Manager
- Ferramentas da linha de comando
- AWS SDKs
- API de consulta HTTPS
- AWS Secrets Manager endpoints

Console do Secrets Manager

É possível gerenciar os seus segredos usando o console do Secrets Manager baseado no navegador e executar praticamente qualquer tarefa relacionada aos seus segredos pelo console.

Ferramentas da linha de comando

As ferramentas de linha de AWS comando permitem que você emita comandos na linha de comando do sistema para executar o Secrets Manager e outras AWS tarefas. Isso pode ser mais rápido e mais conveniente do que usar o console. As ferramentas de linha de comando podem ser úteis se você quiser criar scripts para realizar AWS tarefas.

Quando você insere comandos em um shell de comando, existe o risco de o histórico de comandos ser acessado ou de utilitários terem acesso aos seus parâmetros de comando. Consulte the section called “Mitigue os riscos de usar o AWS CLI para armazenar seus segredos AWS Secrets Manager”.

As ferramentas da linha de comando usam automaticamente o endpoint padrão para o serviço em uma AWS região. É possível especificar um endpoint diferente para suas solicitações de API. Consulte the section called “Endpoint do Secrets Manager”.

AWS fornece dois conjuntos de ferramentas de linha de comando:

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

AWS SDKs

Eles AWS SDKs consistem em bibliotecas e exemplos de código para várias linguagens e plataformas de programação. SDKs Isso inclui tarefas como assinar solicitações criptograficamente, gerenciar erros e repetir solicitações automaticamente. Para baixar e instalar qualquer um deles SDKs, consulte Ferramentas para Amazon Web Services.

Eles usam AWS SDKs automaticamente o endpoint padrão para o serviço em uma AWS região. É possível especificar um endpoint diferente para suas solicitações de API. Consulte the section called "Endpoint do Secrets Manager".

Para obter a documentação do SDK, consulte:

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

API de consulta HTTPS

A API de consulta HTTPS fornece acesso programático ao Secrets Manager e. AWS A API de consulta HTTPS permite executar solicitações HTTPS diretamente ao serviço.

Embora você possa fazer chamadas diretas para a API de consulta HTTPS do Secrets Manager, recomendamos que você use uma delas. SDKs O SDK executa muitas tarefas úteis que você precisaria executar manualmente. Por exemplo, eles assinam SDKs automaticamente suas solicitações e convertem as respostas em uma estrutura sintaticamente apropriada ao seu idioma.

Para fazer chamadas HTTPS para o Secrets Manager, você se conecta a ???.

AWS Secrets Manager endpoints

Para estabelecer conexão programaticamente com o Secrets Manager, você usa um endpoint, o URL do ponto de entrada do serviço. Os endpoints do Secrets Manager são endpoints de pilha dupla, o que significa que eles suportam tanto e. IPv4 IPv6

Em algumas regiões, o Secrets Information oferece endpoints compatíveis com Federal Information Processing Standard (FIPS) 140-2.

O Secrets Manager é compatível com o TLS 1.2 e 1.3. O Secrets Manager é compatível com o PQTLS em todas as regiões, exceto nas da China.

> **Note**
>
> O AWS SDK do Python e a AWS CLI tentativa de chamada IPv6 e, IPv4 em seguida, em sequência. Portanto, se você não tiver IPv6 ativado, pode levar algum tempo até que a chamada atinja o tempo limite e tente novamente. IPv4 Para contornar esse problema, você pode desabilitar IPv6 completamente ou migrar para o. IPv6

Veja a seguir os endpoints de serviço do Secrets Manager. Observe que a nomenclatura difere da convenção de nomenclatura típica de pilha dupla. Para obter informações sobre o uso do endereçamento de pilha dupla no Secrets Manager, consulte IPv4 e IPv6 acesso.

Nome da região	Região	Endpoint	Protocolo	
Leste dos EUA (Ohio)	us-east-2	secretsmanager.us-east-2.amazonaws.com secretsmanager-fips.us-east-2.amazonaws.com	HTTPS HTTPS	
Leste dos EUA (Norte da Virgínia)	us-east-1	secretsmanager.us-east-1.amazonaws.com secretsmanager-fips.us-east-1.amazonaws.com	HTTPS HTTPS	

Nome da região	Região	Endpoint	Protocolo
Oeste dos EUA (N. da Califórnia)	us-west-1	secretsmanager.us-west-1.amazonaws.com secretsmanager-fips.us-west-1.amazonaws.com	HTTPS HTTPS
Oeste dos EUA (Oregon)	us-west-2	secretsmanager.us-west-2.amazonaws.com secretsmanager-fips.us-west-2.amazonaws.com	HTTPS HTTPS
África (Cidade do Cabo)	af-south-1	secretsmanager.af-south-1.amazonaws.com	HTTPS
Ásia-Pacífico (Hong Kong)	ap-east-1	secretsmanager.ap-east-1.amazonaws.com	HTTPS
Ásia-Pacífico (Hyderabad)	ap-south-2	secretsmanager.ap-south-2.amazonaws.com	HTTPS
Ásia-Pacífico (Jacarta)	ap-southeast-3	secretsmanager.ap-southeast-3.amazonaws.com	HTTPS
Ásia-Pacífico (Malásia)	ap-southeast-5	secretsmanager.ap-southeast-5.amazonaws.com	HTTPS

Nome da região	Região	Endpoint	Protocolo	
Ásia-Pacífico (Melbourne)	ap-southeast-4	secretsmanager.ap-southeast-4.amazonaws.com	HTTPS	
Ásia-Pacífico (Mumbai)	ap-south-1	secretsmanager.ap-south-1.amazonaws.com	HTTPS	
Ásia-Pacífico (Nova Zelândia)	ap-southeast-6	secretsmanager.ap-southeast-6.amazonaws.com	HTTPS	
Ásia-Pacífico (Osaka)	ap-northeast-3	secretsmanager.ap-northeast-3.amazonaws.com	HTTPS	
Ásia-Pacífico (Seul)	ap-northeast-2	secretsmanager.ap-northeast-2.amazonaws.com	HTTPS	
Ásia-Pacífico (Singapura)	ap-southeast-1	secretsmanager.ap-southeast-1.amazonaws.com	HTTPS	
Ásia-Pacífico (Sydney)	ap-southeast-2	secretsmanager.ap-southeast-2.amazonaws.com	HTTPS	
Ásia-Pacífico (Taipei)	ap-east-2	secretsmanager.ap-east-2.amazonaws.com	HTTPS	

Nome da região	Região	Endpoint	Protocolo
Ásia-Pacífico (Tailândia)	ap-southeast-7	secretsmanager.ap-southeast-7.amazonaws.com	HTTPS
Ásia-Pacífico (Tóquio)	ap-northeast-1	secretsmanager.ap-northeast-1.amazonaws.com	HTTPS
Canadá (Central)	ca-central-1	secretsmanager.ca-central-1.amazonaws.com secretsmanager-fips.ca-central-1.amazonaws.com	HTTPS HTTPS
Oeste do Canadá (Calgary)	ca-west-1	secretsmanager.ca-west-1.amazonaws.com secretsmanager-fips.ca-west-1.amazonaws.com	HTTPS HTTPS
Europa (Frankfurt)	eu-central-1	secretsmanager.eu-central-1.amazonaws.com	HTTPS
Europa (Irlanda)	eu-west-1	secretsmanager.eu-west-1.amazonaws.com	HTTPS
Europa (Londres)	eu-west-2	secretsmanager.eu-west-2.amazonaws.com	HTTPS
Europa (Milão)	eu-south-1	secretsmanager.eu-south-1.amazonaws.com	HTTPS
Europa (Paris)	eu-west-3	secretsmanager.eu-west-3.amazonaws.com	HTTPS

Nome da região	Região	Endpoint	Protocolo
Europa (Espanha)	eu-south-2	secretsmanager.eu-south-2.amazonaws.com	HTTPS
Europa (Estocolmo)	eu-north-1	secretsmanager.eu-north-1.amazonaws.com	HTTPS
Europa (Zurique)	eu-central-2	secretsmanager.eu-central-2.amazonaws.com	HTTPS
Israel (Tel Aviv)	il-central-1	secretsmanager.il-central-1.amazonaws.com	HTTPS
México (Central)	mx-central-1	secretsmanager.mx-central-1.amazonaws.com	HTTPS
Oriente Médio (Barém)	me-south-1	secretsmanager.me-south-1.amazonaws.com	HTTPS
Oriente Médio (Emirados Árabes Unidos)	me-central-1	secretsmanager.me-central-1.amazonaws.com	HTTPS
América do Sul (São Paulo)	sa-east-1	secretsmanager.sa-east-1.amazonaws.com	HTTPS
AWS GovCloud (Leste dos EUA)	us-gov-east-1	secretsmanager.us-gov-east-1.amazonaws.com secretsmanager-fips.us-gov-east-1.amazonaws.com	HTTPS HTTPS

Nome da região	Região	Endpoint	Protocolo	
AWS GovCloud (Oeste dos EUA)	us-gov-west-1	secretsmanager.us-gov-west-1.amazonaws.com secretsmanager-fips.us-gov-west-1.amazonaws.com	HTTPS HTTPS	

AWS Secrets Manager melhores práticas

O Secrets Manager oferece uma série de atributos de segurança a serem considerados no desenvolvimento e na implementação das suas próprias políticas de segurança. As melhores práticas a seguir são diretrizes gerais e não representam uma solução completa de segurança. Como essas práticas recomendadas podem não ser adequadas ou suficientes para o seu ambiente, trate-as como considerações úteis em vez de prescrições.

Considere as práticas recomendadas a seguir para armazenar e gerenciar segredos:

- Armazene credenciais e outras informações confidenciais em AWS Secrets Manager
- Encontre segredos desprotegidos em seu código
- Escolha uma chave de criptografia para o seu segredo
- Use o armazenamento em cache para recuperar segredos
- Alternar segredos do
- Reduza os riscos decorrentes do uso da CLI
- Limite o acesso aos segredos
- Replique os segredos
- Monitorar segredos
- Execute sua infraestrutura em redes privadas

Armazene credenciais e outras informações confidenciais em AWS Secrets Manager

O Secrets Manager ajuda a melhorar sua postura de segurança e conformidade, além de reduzir o risco de acesso não autorizado às informações confidenciais. O Secrets Manager criptografa segredos em repouso usando chaves de criptografia que você possui e armazena em AWS Key Management Service (AWS KMS). Quando você recupera um segredo, o Secrets Manager decodifica o segredo e o transmite com segurança via TLS para seu ambiente local. Para obter mais informações, consulte Criar segredos.

Encontre segredos desprotegidos em seu código

CodeGuru O Reviewer se integra ao Secrets Manager para usar um detector de segredos que encontra segredos desprotegidos em seu código. O detector de segredos pesquisa senhas diretamente codificadas, cadeias de conexão de banco de dados, nomes de usuário e muito mais. Para obter mais informações, consulte the section called “ CodeGuru Revisor da Amazon”.

O Amazon Q pode varrer sua base de código em busca de vulnerabilidades de segurança e problemas de qualidade de código para melhorar a postura de suas aplicações durante todo o ciclo de desenvolvimento. Para obter mais informações, consulte Varredura do seu código com o Amazon Q no Guia do usuário do Amazon Q Developer.

Escolha uma chave de criptografia para o seu segredo

Na maioria dos casos, recomendamos usar a chave `aws/secretsmanager` AWS gerenciada para criptografar segredos. Não há custo para seu uso.

Para poder acessar um segredo de outra conta ou aplicar uma política de chaves à chave de criptografia, use uma chave gerenciada pelo cliente para criptografar o segredo.

- Na política de chaves, atribua o valor `secretsmanager.<region>.amazonaws.com` à chave de condição `kms:ViaService`. Isso limita o uso da chave somente às solicitações do Secrets Manager.
- Para limitar ainda mais o uso da chave somente a solicitações do Secrets Manager com o contexto correto, use chaves ou valores no contexto de criptografia do Secrets Manager como condição para usar a chave do KMS, criando:
  - Um operador de condição de string em uma política do IAM ou política de chave
  - Uma restrição de concessão em uma concessão

Para obter mais informações, consulte the section called “Criptografia e descriptografia de segredos”.

Use o armazenamento em cache para recuperar segredos

Para usar seus segredos com mais eficiência, recomendamos que você use um dos seguintes componentes de cache com suporte do Secrets Manager para armazenar seus segredos em cache e atualizá-los somente quando necessário:

- Java com armazenamento em cache no lado do cliente
- Python com armazenamento em cache no lado do cliente
- .NET com armazenamento em cache no lado do cliente
- Go com armazenamento em cache no lado do cliente
- Rust com armazenamento em cache no lado do cliente
- AWS Parâmetros e segredos da extensão Lambda
- the section called “Amazon EKS”
- Use the section called “Agente do Secrets Manager” para padronizar o consumo de segredos do Secrets Manager em ambientes como AWS Lambda Amazon Elastic Container Service, Amazon Elastic Kubernetes Service e Amazon Elastic Compute Cloud.

Alternar segredos do

Se você não alterar o segredos por um longo período, eles se tornam mais propensos a ser comprometidos. Com o Secrets Manager, é possível configurar alternância automática com intervalos a partir de quatro horas. O Secrets Manager oferece duas estratégias de alternância: Usuário único e Usuários alternados. Para obter mais informações, consulte Alternar segredos .

Reduza os riscos decorrentes do uso da CLI

Ao usar as AWS operações AWS CLI para invocar, você insere esses comandos em um shell de comando. A maioria dos shells de comando oferece atributos que podem comprometer seus segredos, como o registro log e a capacidade de ver o último comando digitado. Antes de usar a AWS CLI para inserir informações confidenciais, certifique-se sobre the section called “Mitigue os riscos de usar o AWS CLI para armazenar seus segredos AWS Secrets Manager”.

Limite o acesso aos segredos

Em declarações de política do IAM que controlem o acesso aos seus segredos, use o princípio de acesso de privilégio mínimo. É possível usar políticas e perfis do IAM, políticas de recursos e controle de acesso por atributo (ABAC). Para obter mais informações, consulte the section called “Autenticação e controle de acesso”.

Tópicos

- Bloqueie o acesso amplo aos segredos

- Tenha cuidado com as condições de endereço IP nas políticas
- Limite solicitações com condições do endpoint da VPC

Bloqueie o acesso amplo aos segredos

Em políticas de identidade que permitem a ação `PutResourcePolicy`, recomendamos usar `BlockPublicPolicy: true`. Essa condição significa que os usuários só poderão anexar uma política de recursos a um segredo se a política não permitir acesso amplo.

O Secrets Manager usa o raciocínio automatizado Zelkova para analisar políticas de recursos para acesso amplo. Para obter mais informações sobre Zelkova, consulte Como AWS usa o raciocínio automatizado para ajudar você a obter segurança em grande escala no AWS Blog de Segurança.

O exemplo a seguir mostra como usar `BlockPublicPolicy`.

JSON

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

Tenha cuidado com as condições de endereço IP nas políticas

Tome cuidado ao especificar os operadores de condição do endereço IP ou a chave de condição `aws:SourceIp` em uma declaração de política que permite ou nega acesso ao Secrets Manager. Por exemplo, se você anexar uma política que restringe AWS ações às solicitações do intervalo de endereços IP da sua rede corporativa a um segredo, suas solicitações como usuário do IAM

que invoca a solicitação da rede corporativa funcionarão conforme o esperado. No entanto, se você permitir que outros serviços acessem o segredo em seu nome, como ao ativar a rotação com uma função Lambda, essa função chamará as operações do Secrets Manager a partir de um espaço AWS de endereço interno. As solicitações afetadas pela política com o filtro de endereços IP falharão.

Além disso, a chave de condição `aws:sourceIP` é menos efetiva quando a solicitação se origina em um endpoint da Amazon VPC. Para restringir solicitações a um endpoint da VPC específico, use the section called “Limite solicitações com condições do endpoint da VPC”.

Limite solicitações com condições do endpoint da VPC

Para conceder ou negar acesso a solicitações de uma determinada VPC ou de um endpoint da VPC, use `aws:SourceVpc` para limitar o acesso a solicitações da VPC especificada ou `aws:SourceVpce` para limitar o acesso a solicitações do endpoint da VPC especificado. Consulte the section called “Exemplo: permissões e VPCs”.

- `aws:SourceVpc` limita o acesso a solicitações da VPC especificado.
- `aws:SourceVpce` limita o acesso a solicitações do endpoint da VPC especificado.

Se você usar essas chaves de condição em uma declaração de política de recurso que permite ou nega o acesso a segredos do Secrets Manager, poderá inadvertidamente negar acesso a serviços que usam o Secrets Manager para acessar segredos em seu nome. Somente alguns AWS serviços podem ser executados com um endpoint em sua VPC. Se você restringir as solicitações de um segredo a uma VPC ou a um endpoint da VPC, poderão ocorrer falhas nas chamadas para o Secrets Manager de um serviço não configurado para o serviço.

Consulte the section called “Endpoints da VPC (AWS PrivateLink)”.

Replique os segredos

O Secrets Manager pode replicar automaticamente seus segredos em várias AWS regiões para atender aos seus requisitos de resiliência ou recuperação de desastres. Para obter mais informações, consulte Replicação multirregional.

Monitorar segredos

O Secrets Manager permite auditar e monitorar segredos por meio da integração com serviços de AWS registro, monitoramento e notificação. Para obter mais informações, consulte:

- the section called “Faça login com AWS CloudTrail ”
- the section called “Monitor com CloudWatch”
- the section called “Monitorar segredos para conformidade”
- the section called “Monitorar custos do Secrets Manager”
- the section called “Detecte ameaças com GuardDuty”

Execute sua infraestrutura em redes privadas

Sempre que possível, recomendamos que você execute o máximo da infraestrutura em redes privadas que não sejam acessíveis pela Internet pública. É possível estabelecer uma conexão privada entre a sua VPC e o Secrets Manager criando um interface VPC endpoint (Endpoint da VPC de interface). Para obter mais informações, consulte the section called “Endpoints da VPC (AWS PrivateLink)”.

AWS Secrets Manager tutoriais

Tópicos

Encontre segredos desprotegidos em seu código com o Amazon Reviewer CodeGuru

O Amazon CodeGuru Reviewer é um serviço que usa análise de programas e aprendizado de máquina para detectar possíveis defeitos difíceis de serem encontrados pelos desenvolvedores e oferece sugestões para melhorar seu código Java e Python. CodeGuru O Reviewer se integra ao Secrets Manager para encontrar segredos desprotegidos em seu código. Para saber os tipos de segredos que ele pode encontrar, consulte Tipos de segredos detectados pelo CodeGuru revisor no Guia do usuário do Amazon CodeGuru Reviewer.

Depois de encontrar segredos codificados, realize ações para substituí-los:

- the section called “Substituir credenciais de banco de dados codificadas ”
- the section called “Substituir segredos codificados ”

Mova segredos codificados para AWS Secrets Manager

Caso você tenha segredos de texto simples no código, recomendamos alterná-los e armazená-los no Secrets Manager. Transferir o segredo para o Secrets Manager resolverá o problema de deixar o segredo visível para qualquer pessoa que visualizar o código, porque, a partir de então, seu código recuperará o segredo diretamente do Secrets Manager. Alternar o segredo revogará o segredo codificado atual para que não seja mais válido.

Para obter segredos de credenciais de banco de dados, consulte Mova as credenciais codificadas do banco de dados para AWS Secrets Manager.

Antes de começar, é necessário determinar quem precisa acessar o segredo. Recomendamos usar dois perfis do IAM para gerenciar a permissão para seu segredo:

- Um perfil que gerencia os segredos da organização. Para obter mais informações, consulte the section called “Permissões de administrador do Secrets Manager”. Você criará e alternará o segredo usando esse perfil.
- Uma função que pode usar o segredo em tempo de execução, por exemplo, neste tutorial que você usa*RoleToRetrieveSecretAtRuntime*. Seu código assume esse perfil para recuperar o segredo. Neste tutorial, você concede ao perfil apenas a permissão para recuperar um valor de segredo e concede permissão usando a política de recursos do segredo. Para conhecer as alternativas, consulte the section called “Próximas etapas”.

Etapas:

- Etapa 1: criar o segredo
- Etapa 2: atualizar o código
- Etapa 3: atualizar o segredo
- Próximas etapas

Etapa 1: criar o segredo

A primeira etapa é copiar o segredo codificado existente para o Secrets Manager. Se o segredo estiver relacionado a um AWS recurso, armazene-o na mesma região do recurso. Caso contrário, armazene-o na região com a menor latência para seu caso de uso.

Para criar um segredo (console)

1. Abra o console do Secrets Manager em https://console.aws.amazon.com/secretsmanager/.
2. Selecione Armazenar um novo segredo.
3. Na página Choose secret type (Selecionar tipo de segredo), faça o seguinte:
    a. Em Tipo de segredo, escolha Outro tipo de segredo.
    b. Digite seu segredo como pares de chave-valor ou em texto não criptografado. Alguns exemplos:

API key

Insira como key/value pares:

ClientID : *my_client_id*

ClientSecret : *wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY*

OAuth token

Insira como texto sem formatação:

AKIAI44QH8DHBEXAMPLE

Digital certificate

Insira como texto sem formatação:

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

Private key

Insira como texto sem formatação:

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

c. Em Encryption key (Chave de criptografia), escolha aws/secretsmanager para usar a Chave gerenciada pela AWS para Secrets Manager. Não há custo para o uso dessa chave. Você também pode usar sua própria chave gerenciada pelo cliente, por exemplo, para acessar o segredo de outra Conta da AWS. Para obter mais informações sobre os custos do uso de uma chave gerenciada pelo cliente, consulte Preços.

d. Escolha Próximo.

4. Na página Choose secret type (Selecionar tipo de segredo), faça o seguinte:

a. Insira um Secret name (Nome de segredo) descritivo e uma Description (Descrição).

b. Em Resource permissions (Permissões do recurso), escolha Edit permissions (Editar permissões). Cole a política a seguir, que *RoleToRetrieveSecretAtRuntime* permite recuperar o segredo, e escolha Salvar.

JSON

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

c. Na parte inferior da página, selecione a opção Próximo.

5. Na página Configure rotation (Configurar alternância), mantenha a alternância ligada. Escolha Próximo.
6. Na página Review (Revisar), revise os detalhes do segredo e escolha Store (Armazenar).

Etapa 2: atualizar o código

Seu código deve assumir a função *RoleToRetrieveSecretAtRuntime* do IAM para poder recuperar o segredo. Para obter mais informações, consulte Mudar para uma função do IAM (AWS API).

Em seguida, atualize o código para recuperar o segredo do Secrets Manager usando o código de exemplo fornecido pelo Secrets Manager.

Como encontrar o código de exemplo

1. Abra o console do Secrets Manager em https://console.aws.amazon.com/secretsmanager/.
2. Na página Secrets (Segredos), escolha o segredo.

3. Role para baixo até Sample code (Código de exemplo). Escolha a linguagem de programação e copie o trecho de código.

Em sua aplicação, remova o segredo codificado e cole o trecho de código. Conforme o idioma do código, talvez seja necessário adicionar uma chamada ao perfil ou método no trecho de código.

Teste se sua aplicação funciona conforme o esperado com o segredo no lugar do segredo codificado.

Etapa 3: atualizar o segredo

A última etapa é revogar e atualizar o segredo codificado. Consulte a origem do segredo para encontrar instruções para revogá-lo e atualizá-lo. Por exemplo, talvez seja necessário desativar o segredo atual e gerar um novo segredo.

Para atualizar o segredo com o novo valor

1. Abra o console do Secrets Manager em https://console.aws.amazon.com/secretsmanager/.
2. Escolha Secrets (Segredos) e escolha o segredo.
3. Na página Secret details (Detalhes do segredo), role para baixo e escolha Retrieve secret value (Recuperar valor do segredo) e depois Edit (Editar).
4. Atualize o segredo e escolha Save (Salvar).

Em seguida, teste se a aplicação funciona conforme o esperado com o novo segredo.

Próximas etapas

Depois de remover um segredo codificado de seu código, há algumas ideias a serem consideradas em seguida:

- Para encontrar segredos codificados em seus aplicativos Java e Python, recomendamos o Amazon Reviewer. CodeGuru
- É possível melhorar a perfomance e reduzir custos armazenando segredos em cache. Para obter mais informações, consulte Obter segredos.
- Para segredos acessados de várias regiões, você pode replicar seu segredo para melhorar a latência. Para obter mais informações, consulte Replicação multirregional.

- Neste tutorial, você concedeu *RoleToRetrieveSecretAtRuntime* somente a permissão para recuperar o valor secreto. Para conceder mais permissões ao perfil, por exemplo, para obter metadados sobre o segredo ou para visualizar uma lista de segredos, consulte the section called "Políticas baseadas em recursos".
- Neste tutorial, você concedeu permissão *RoleToRetrieveSecretAtRuntime* usando a política de recursos do segredo. Para saber outras formas de conceder permissão, consulte the section called "Políticas baseadas em identidade".

Mova as credenciais codificadas do banco de dados para AWS Secrets Manager

Caso você tenha credenciais de banco de dados em texto simples no código, recomendamos mover as credencias para o Secrets Manager e alterná-las imediatamente. Mover as credenciais para o Secrets Manager resolverá o problema de deixar as credenciais visíveis para qualquer pessoa que visualizar o código, porque, a partir de então, seu código recuperará as credenciais diretamente do Secrets Manager. Alternar o segredo atualizará a senha e revogará a senha codificada atual para que não seja mais válida.

Para bancos de dados do Amazon RDS, Amazon Redshift e Amazon DocumentDB, use as etapas desta página para mover credenciais codificadas para o Secrets Manager. Para outros tipos de credenciais e outros segredos, consulte the section called "Substituir segredos codificados ".

Antes de começar, é necessário determinar quem precisa acessar o segredo. Recomendamos usar dois perfis do IAM para gerenciar a permissão para seu segredo:

- Um perfil que gerencia os segredos da organização. Para obter mais informações, consulte the section called "Permissões de administrador do Secrets Manager". Você criará e alternará o segredo usando esse perfil.
- Uma função que pode usar as credenciais em tempo de execução, *RoleToRetrieveSecretAtRuntime* neste tutorial. Seu código assume esse perfil para recuperar o segredo.

Etapas:

- Etapa 1: criar o segredo
- Etapa 2: atualizar o código

Etapa 1: criar o segredo

A primeira etapa é copiar as credenciais codificadas existentes para o Secrets Manager. Para obter a latência mais baixa, armazene o segredo na mesma região do banco de dados.

Como criar um segredo

1. Abra o console do Secrets Manager em https://console.aws.amazon.com/secretsmanager/.
2. Selecione Armazenar um novo segredo.
3. Na página Choose secret type (Selecionar tipo de segredo), faça o seguinte:
   a. Em Secret type (Tipo de segredo), escolha o tipo de credenciais de banco de dados a armazenar:
      - Bancos de dados do Amazon RDS
      - Amazon DocumentDB database (Bancos de dados do Amazon DocumentDB)
      - Data warehouse do Amazon Redshift.
      - Para outros tipos de segredos, consulte Replace hardcoded secrets (Substituir segredos codificados).
   b. Em Credenciais, insira as credenciais codificadas existentes para o banco de dados.
   c. Em Encryption key (Chave de criptografia), escolha aws/secretsmanager para usar a Chave gerenciada pela AWS para Secrets Manager. Não há custo para o uso dessa chave. Você também pode usar sua própria chave gerenciada pelo cliente, por exemplo, para acessar o segredo de outra Conta da AWS. Para obter mais informações sobre os custos do uso de uma chave gerenciada pelo cliente, consulte Preços.
   d. Em Database (Banco de dados), escolha seu banco de dados.
   e. Escolha Próximo.
4. Na página Configure secret (Configurar segredo), faça o seguinte:
   a. Insira um Secret name (Nome de segredo) descritivo e uma Description (Descrição).
   b. Em Resource permissions (Permissões do recurso), escolha Edit permissions (Editar permissões). Cole a política a seguir, que *RoleToRetrieveSecretAtRuntime* permite recuperar o segredo, e escolha Salvar.

JSON

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

c. Na parte inferior da página, selecione a opção Próximo.

5. Na página Configure rotation (Configurar alternância), mantenha a alternância desligada por enquanto. Você a ativará depois. Escolha Próximo.
6. Na página Review (Revisar), revise os detalhes do segredo e escolha Store (Armazenar).

Etapa 2: atualizar o código

Seu código deve assumir a função *RoleToRetrieveSecretAtRuntime* do IAM para poder recuperar o segredo. Para obter mais informações, consulte Mudar para uma função do IAM (AWS API).

Em seguida, atualize o código para recuperar o segredo do Secrets Manager usando o código de exemplo fornecido pelo Secrets Manager.

Como encontrar o código de exemplo

1. Abra o console do Secrets Manager em https://console.aws.amazon.com/secretsmanager/.
2. Na página Secrets (Segredos), escolha o segredo.
3. Role para baixo até Sample code (Código de exemplo). Escolha a linguagem e copie o trecho de código.

Em sua aplicação, remova as credenciais codificadas e cole o trecho de código. Conforme o idioma do código, talvez seja necessário adicionar uma chamada ao perfil ou método no trecho de código.

Teste se sua aplicação funciona conforme o esperado com o segredo no lugar das credenciais codificadas.

Etapa 3: alternar o segredo

A última etapa é revogar as credenciais codificadas alternando o segredo. Alternância é o processo de atualizar periodicamente um segredo. Quando o Secrets Manager alterna um segredo, você atualiza as credenciais tanto no segredo como no banco de dados. O Secrets Manager pode alternar automaticamente um segredo para você em uma programação que você estabelecer.

Parte da configuração da alternância é garantir que a função de alternância do Lambda possa acessar o Secrets Manager e seu banco de dados. Quando você ativa a alternância automática, o Secrets Manager cria a função de alternância do Lambda na mesma VPC do banco de dados para acessar o banco de dados pela rede. A função de alternância do Lambda também deve conseguir fazer chamadas ao Secrets Manager para atualizar o segredo. Recomendamos que você crie um endpoint do Secrets Manager na VPC para que as chamadas do Lambda para o Secrets Manager não saiam da infraestrutura. AWS Para instruções, consulte the section called “Endpoints da VPC (AWS PrivateLink)”.

Para ativar a alternância

1. Abra o console do Secrets Manager em https://console.aws.amazon.com/secretsmanager/.
2. Na página Secrets (Segredos), escolha o segredo.
3. Na página Secret details (Detalhes do segredo), na seção Rotation configuration (Configuração da alternância), escolha Edit rotation (Editar alternância).
4. Na caixa de diálogo Edit rotation configuration (Editar configuração da alternância), siga estas etapas:
    a. Ative a Automatic rotation (Alternância automática).
    b. Em Rotation schedule (Programação de alternância), insira a sua programação no fuso horário UTC.
    c. Escolha Rotate immediately when the secret is stored (Alternar imediatamente quando o segredo for armazenado) para alternar o segredo assim que suas alterações forem salvas.

d. Em Rotation function (Função de alternância), escolha Create a new Lambda function (Criar uma nova função Lambda) e insira um nome para a nova função. O Secrets Manager adiciona "SecretsManager" no início do nome da função.

e. Em Estratégia de alternância, escolha Usuário único.

f. Escolha Salvar.

Para verificar se o segredo foi alternado

1. Abra o console do Secrets Manager em https://console.aws.amazon.com/secretsmanager/.
2. Escolha Secrets (Segredos) e escolha o segredo.
3. Na página Secret details (Detalhes do segredo), role para baixo e escolha Retrieve secret value (Recuperar valor do segredo).

   Se o valor do segredo foi alterado, a alternância funcionou. Se o valor secreto não mudou, você precisa Solucionar problemas de alternância examinar os CloudWatch registros da função de rotação.

Teste se a aplicação funciona conforme o esperado com o segredo alternado.

Próximas etapas

Depois de remover um segredo codificado de seu código, há algumas ideias a serem consideradas em seguida:

- É possível melhorar a perfomance e reduzir custos armazenando segredos em cache. Para obter mais informações, consulte Obter segredos.
- É possível escolher uma programação de alternância diferente. Para obter mais informações, consulte the section called “Programação de alternância”.
- Para encontrar segredos codificados em seus aplicativos Java e Python, recomendamos o Amazon Reviewer. CodeGuru

Configure a rotação alternada de usuários para AWS Secrets Manager

Neste tutorial, você aprenderá como configurar a alternância de usuários alternados para um segredo que contenha credenciais de banco de dados. Alternância de usuários alternados é uma estratégia de alternância em que o Secrets Manager clona o usuário e, em seguida, alterna quais credenciais do usuário são atualizadas. Essa estratégia é uma boa opção se você precisar de alta disponibilidade para seu segredo, porque um dos usuários alternados tem credenciais atuais para o banco de dados enquanto o outro está sendo atualizado. Para obter mais informações, consulte the section called “Usuários alternados”.

Para configurar a alternância de usuários alternados, você precisa de dois segredos:

- Um segredo com as credenciais que você deseja alternar.
- Um segundo segredo que tem credenciais de administrador.

  Esse usuário tem permissões para clonar o primeiro usuário e alterar a senha do primeiro usuário. Neste tutorial, o Amazon RDS cria esse segredo para um usuário administrador. O Amazon RDS também gerencia a alternância da senha do administrador. Para obter mais informações, consulte the section called “Alternância gerenciada”.

A primeira parte deste tutorial está configurando um ambiente realista. Para mostrar como funciona a alternância, este tutorial usa um exemplo de banco de dados MySQL do Amazon RDS. Por segurança, o banco de dados está em uma VPC que não permite acesso de entrada à internet. Para se conectar ao banco de dados do computador local pela Internet, você usa um bastion host, um servidor na VPC que pode se conectar ao banco de dados, mas que também permite conexões SSH pela internet. O bastion host neste tutorial é uma instância do Amazon EC2, e os grupos de segurança da instância impedem outros tipos de conexões.

Depois de concluir o tutorial, recomendamos limpar os recursos do tutorial. Não os use em um ambiente de produção.

A rotação do Secrets Manager usa uma AWS Lambda função para atualizar o segredo e o banco de dados. Para obter mais informações sobre os custos do uso de uma função do Lambda, consulte Preços.

Tutorial:

- Permissões

Permissões

Para os pré-requisitos do tutorial, você precisa de permissões administrativas para sua Conta da AWS. Em uma configuração de produção, é uma prática recomendada usar funções diferentes para cada uma das etapas. Por exemplo, uma função com permissões de administrador de banco de dados criaria o banco de dados do Amazon RDS, e uma função com permissões de administrador de rede configuraria a VPC e os grupos de segurança. Para as etapas do tutorial, recomendamos que você continue usando a mesma identidade.

Para obter mais informações sobre como configurar permissões em um ambiente de produção, consulte the section called “Autenticação e controle de acesso”.

Pré-requisitos

Para este tutorial, você precisa do seguinte:

Pré-requisito A: Amazon VPC

Nesta etapa, crie uma VPC na qual pode ser iniciado um banco de dados do Amazon RDS e uma instância do Amazon EC2. Em uma etapa posterior, você usará seu computador para se conectar pela Internet ao bastion e depois ao banco de dados, portanto, precisará permitir que o tráfego saia da VPC. Para fazer isso, o Amazon VPC conecta um gateway da Internet à VPC e adiciona uma rota

na tabela de rotas de modo que o tráfego com destino para fora da VPC seja enviado para o gateway da Internet.

Dentro da VPC, você cria um endpoint do Secrets Manager e um endpoint do Amazon RDS. Quando você configura a alternância automática em uma etapa posterior, o Secrets Manager cria uma função de alternância do Lambda dentro da VPC para que ela possa acessar o banco de dados. A função de alternância do Lambda também chama o Secrets Manager para atualizar o segredo e chama o Amazon RDS para obter as informações de conexão do banco de dados. Ao criar endpoints na VPC, você garante que as chamadas da função Lambda para o Secrets Manager e o Amazon RDS não saiam da infraestrutura. AWS Em vez disso, elas são encaminhadas para os endpoints dentro da VPC.

Para criar uma VPC

1. Abra o console da Amazon VPC em https://console.aws.amazon.com/vpc/.
2. Escolha Criar VPC.
3. Na página Create VPC (Criar VPC), escolha VPC and more (VPC e muito mais).
4. Em Name tag auto-generation (Geração automática de tags de nome), em Auto-generate (Gerar automaticamente), insira **`SecretsManagerTutorial`**.
5. Em DNS options (Opções de DNS), escolha **`Enable DNS hostnames`** e **`Enable DNS resolution`**.
6. Escolha Criar VPC.

Para criar um endpoint do Secrets Manager na VPC

1. No console do Amazon VPC, em Endpoints, escolha Create Endpoint (Criar endpoint).
2. Em Endpoint settings (Configurações de endpoint), em API Name (Nome da API), insira **`SecretsManagerTutorialEndpoint`**.
3. Em Services (Serviços), insira **`secretsmanager`** para filtrar a lista e, em seguida, selecione o endpoint do Secrets Manager na sua Região da AWS. Por exemplo, no Leste dos EUA (Norte da Virgínia), escolha `com.amazonaws.us-east-1.secretsmanager`.
4. Em VPC, escolha **`vpc**** (SecretsManagerTutorial)`**.
5. Em Sub-redes, selecione todas as Availability Zones (Zonas de disponibilidade) e, em seguida, para cada uma, escolha um Subnet ID (ID de sub-rede) para incluir.
6. Em IP address type (Tipo de endereço IP), escolha **`IPv4`**.

7. Em Security groups (Grupos de segurança), escolha o grupo de segurança padrão.
8. Em Policy (Política), selecione **`Full access`**.
9. Escolha Criar endpoint.

Para criar um endpoint do Amazon RDS na VPC

1. No console do Amazon VPC, em Endpoints, escolha Create Endpoint (Criar endpoint).
2. Em Endpoint settings (Configurações de endpoint), em API Name (Nome da API), insira **`RDSTutorialEndpoint`**.
3. Em Services (Serviços), insira **`rds`** para filtrar a lista e, em seguida, selecione o endpoint do Amazon RDS na sua Região da AWS. Por exemplo, no Leste dos EUA (Norte da Virgínia), escolha `com.amazonaws.us-east-1.rds`.
4. Em VPC, escolha **`vpc**** (SecretsManagerTutorial)`**.
5. Em Sub-redes, selecione todas as Availability Zones (Zonas de disponibilidade) e, em seguida, para cada uma, escolha um Subnet ID (ID de sub-rede) para incluir.
6. Em IP address type (Tipo de endereço IP), escolha **`IPv4`**.
7. Em Security groups (Grupos de segurança), escolha o grupo de segurança padrão.
8. Em Policy (Política), selecione **`Full access`**.
9. Escolha Criar endpoint.

Pré-requisito B: instância do Amazon EC2

O banco de dados do Amazon RDS que você criar em uma etapa posterior estará na VPC, portanto, para acessá-lo, você precisa de um bastion host. O bastion host também está na VPC, mas em uma etapa posterior, você configura um grupo de segurança de modo que seu computador local se conecte ao bastion host com SSH.

Para criar uma instância do EC2 para um bastion host

1. Abra o console do Amazon EC2 em https://console.aws.amazon.com/ec2/.
2. Escolha Instances (Instâncias) e, em seguida, escolha Launch Instances (Iniciar instâncias).
3. Em Name and tags (Nome e etiquetas), em Name (Nome), insira **`SecretsManagerTutorialInstance`**.

4. Em Application and OS Images (Imagens do aplicativo e do sistema operacional), mantenha o padrão **`Amazon Linux 2 AMI (HMV) Kernel 5.10`**.
5. Em Instance type (Tipo de instância), mantenha o padrão **`t2.micro`**.
6. Em Key pair (Par de chaves), escolha Create key pair (Criar par de chaves).

   Na caixa de diálogo Create Key Pair (Criar par de chaves), em Key pair name (Nome do par de chaves), insira **`SecretsManagerTutorialKeyPair`** e selecione Create key pair (Criar par de chaves).

   O download do par de chaves será realizado automaticamente.
7. Em Network settings (Configurações da rede), escolha Edit (Editar), e faça o seguinte:
   a. Em VPC, escolha **`vpc-**** SecretsManagerTutorial`**.
   b. Em Auto-assign Public IP (Atribuir IP público automaticamente), selecione **`Enable`**.
   c. Em Firewall, escolha Select existing security group (Selecionar grupo de segurança existente).
   d. Em Common security groups (Grupos de segurança comuns), escolha **`default`**.
8. Escolha Iniciar instância.

Pré-requisito C: banco de dados do Amazon RDS e um segredo do Secrets Manager para as credenciais do administrador

Nesta etapa, você cria um banco de dados do Amazon RDS MySQL e o configura de modo que o Amazon RDS crie um segredo para conter as credenciais do administrador. Em seguida, o Amazon RDS gerencia automaticamente a alternância do segredo do administrador para você. Para obter mais informações, consulte Alternância gerenciada.

Como parte da criação do seu banco de dados, você especifica o bastion host que criou na etapa anterior. Em seguida, o Amazon RDS configura grupos de segurança para que o banco de dados e a instância possam se acessar. Você adiciona uma regra ao grupo de segurança anexado à instância para permitir que seu computador local também se conecte a ela.

Para criar um banco de dados Amazon RDS com um segredo do Secrets Manager que contém as credenciais de administrador

1. No console do Amazon RDS, escolha Create database (Criar banco de dados).

2. Na seção Engine options (Opções de mecanismo), em Engine type (Tipo de mecanismo), escolha **`MySQL`**.
3. Na seção Templates (Modelos), escolha **`Free tier`**.
4. Na seção Settings (Configurações), faça o seguinte:
   a. Para DB instance identifier (Identificador de instância de banco de dados), insira **`SecretsManagerTutorial`**.
   b. Em Configurações de credenciais, selecione Gerenciar credenciais mestras em. AWS Secrets Manager
5. Na seção Connectivity (Conectividade), em Computer resource (Recurso do computador), escolha Connect to an EC2 computer resource (Conectar a um recurso de computador do EC2) e, em seguida, em EC2 Instance (Instância do EC2), escolha **`SecretsManagerTutorialInstance`**.
6. Selecione Criar banco de dados.

Pré-requisito D: Permita que seu computador local se conecte à instância do EC2

Nesta etapa, você configura a instância do EC2 criada no Pré-requisito B para permitir que seu computador local se conecte a ela. Para fazer isso, você edita o grupo de segurança que o Amazon RDS adicionou no Pré-requisito C para incluir uma regra que permite que o endereço IP do seu computador se conecte ao SSH. A regra permite que seu computador local (identificado pelo seu endereço IP atual) se conecte ao bastion host usando SSH pela Internet.

Para permitir que seu computador local se conecte à instância do EC2

1. Abra o console do Amazon EC2 em https://console.aws.amazon.com/ec2/.
2. Na instância EC2 SecretsManagerTutorialInstance, na guia Segurança, em Grupos de segurança, escolha**`sg-*** (ec2-rds-X)`**.
3. Em Input Rules (Regras de entrada), selecione Edit inbound rules (Editar regras de entrada).
4. Selecione Add Rule (Adicionar regra) e, na regra, e faça o seguinte:
   a. Em Type (Tipo), escolha **`SSH`**.
   b. Em Source type (Tipo de fonte), selecione **`My IP`**.

Etapa 1: criar um usuário do banco de dados do Amazon RDS

Primeiro, você precisa de um usuário cujas credenciais serão armazenadas no segredo. Para criar o usuário, faça login no banco de dados do Amazon RDS com credenciais de administrador. Para simplificar, no tutorial, você cria um usuário com permissão total para um banco de dados. Em um ambiente de produção, isso não é típico. Recomendamos que você siga o princípio de privilégio mínimo.

Para se conectar ao banco de dados, você usa uma ferramenta cliente de MySQL. Neste tutorial, você usa o MySQL Workbench, uma aplicação baseada em GUI. Baixe o MySQL Workbench em Download MySQL Workbench (Baixar MySQL Workbench).

Para se conectar ao banco de dados, crie uma configuração de conexão no MySQL Workbench. Para a configuração, você precisa de algumas informações do Amazon EC2 e do Amazon RDS.

Para criar uma conexão de banco de dados no MySQL Workbench

1. No MySQL Workbench, ao lado de MySQL Connections (Conexões do MySQL), escolha o botão (+).
2. Na caixa de diálogo Setup New Connection (Configurar uma nova conexão), faça o seguinte:
   a. Em Connection Name (Nome da conexão), insira **`SecretsManagerTutorial`**.
   b. Em Connection Method (Método de conexão), escolha **`Standard TCP/IP over SSH`**.
   c. Na guia Parameters (Parâmetros), faça o seguinte:
      i. Em SSH Hostname (Nome do host de SSH), insira o endereço IP público da instância do Amazon EC2.

         Você pode encontrar o endereço IP no console do Amazon EC2 escolhendo a instância. SecretsManagerTutorialInstance Copie o endereço IP em IPv4 DNS público.
      ii. Em SSH Username (Nome de usuário do SSH), insira **`ec2-user`**.
      iii. Para Arquivo de chave SSH, escolha o arquivo de par de chaves SecretsManagerTutorialKeyPair.pem que você baixou no pré-requisito anterior.
      iv. Em MySQL Hostname (Nome do host do MySQL), insira o endereço do endpoint do Amazon RDS.

         É possível encontrar o endereço do endpoint no console do Amazon RDS escolhendo a instância do banco de dados secretsmanagertutorialdb. Copie o endereço em Endpoint.

v. Em Username (Nome do usuário), insira **admin**.

d. Escolha OK.

Para recuperar a senha do administrador

1. No Console do Amazon RDS, navegue até o seu banco de dados.
2. Na guia Configuration (Configuração), em Master Credentials ARN (ARN das credenciais principais), escolha Manage in Secrets Manager (Gerenciar no Secrets Manager).

   O Console do Secrets Manager é aberto.
3. Na página de detalhes do seu segredo, escolha Retrieve secret value (Recuperar valor do segredo).
4. A senha aparece na seção Secret value (Valor do segredo).

Para criar um usuário de banco de dados

1. No MySQL Workbench, escolha a conexão. SecretsManagerTutorial
2. Digite a senha de administrador que você recuperou do segredo.
3. Em MySQL Workbench, na janela Query (Consultar), insira os seguintes comandos (incluindo uma senha forte) e, depois, escolha Execute (Executar). A função de alternância testa o segredo atualizado usando SELECT, portanto, **appuser** deve ter esse privilégio, no mínimo.

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

   Na janela Output (Saída), você verá os comandos com êxito.

Etapa 2: criar um segredo para as credenciais do usuário

Em seguida, crie um segredo para armazenar as credenciais do usuário que acabou de criar. Este é o segredo que você estará alternando. Você ativa a alternância automática e, para indicar a estratégia de usuários alternados, você escolhe um segredo de superusuário separado que tenha permissão para alterar a senha do primeiro usuário.

1. Abra o console do Secrets Manager em https://console.aws.amazon.com/secretsmanager/.

2. Selecione Armazenar um novo segredo.
3. Na página Choose secret type (Selecionar tipo de segredo), faça o seguinte:
   a. Em Secret type (Tipo de segredo), escolha Credentials for Amazon RDS database (Credenciais para o banco de dados do Amazon RDS).
   b. Em Credentials (Credenciais), insira o nome do usuário **`appuser`** e a senha inserida para o usuário do banco de dados que você criou usando o MySQL Workbench.
   c. Em Database (Banco de dados), escolha secretsmanagertutorialdb.
   d. Escolha Próximo.
4. Na página Configure secret (Configurar segredo) , em Secret name (Nome de segredo), insira **`SecretsManagerTutorialAppuser`** e, depois, escolha Next (Próximo).
5. Na página Configure rotation (Configurar alternância), faça o seguinte:
   a. Ative a Automatic rotation (Alternância automática).
   b. Em Rotation schedule (Programação da alternância), defina uma programação de Days (Dias):**2** dias com Duration (Duração): **2h**. Mantenha Rotate immediately (Alternar imediatamente) selecionado.
   c. Em Rotation function (Função de alternância), escolha Create a rotation function (Criar uma função de alternância), e, em seguida, para nome da função, insira **`tutorial-alternating-users-rotation`**.
   d. Em Estratégia de alternância, escolha Usuários alternados e, em Segredo da credencial do administrador, escolha o segredo com o nome rds!cluster... e cuja Descrição inclui o nome do banco de dados que você criou neste tutorial **`secretsmanagertutorial`**, por exemplo, `Secret associated with primary RDS DB instance: arn:aws:rds:`*`Region`*`:`*`AccountId`*`:db:secretsmanagertutorial`.
   e. Escolha Próximo.
6. Na página Review (Análise), escolha Store (Armazenar).

   O Secrets Manager retorna à página de detalhes do segredo. Na parte superior da página, é possível ver o status da configuração de alternância. O Secrets Manager usa CloudFormation para criar recursos como a função de rotação Lambda e uma função de execução que executa a função Lambda. Quando CloudFormation terminar, o banner muda para Segredo programado para rotação. A primeira alternância está completa.

Etapa 3: testar o segredo alternado

Agora que o segredo está alternado, é possível verificar se o segredo contém credenciais válidas. A senha no segredo mudou a partir das credenciais originais.

Para recuperar a nova senha do segredo

1. Abra o console do Secrets Manager em https://console.aws.amazon.com/secretsmanager/.
2. Escolha Secrets (Segredos) e, em seguida, escolha o segredo **`SecretsManagerTutorialAppuser`**.
3. Na página Secret details (Detalhes do segredo), role para baixo e escolha Retrieve secret value (Recuperar valor do segredo).
4. Na tabela Key/value (Chave/valor), copie o Secret value (Valor do segredo) para **`password`**.

Para testar as credenciais

1. No MySQL Workbench, clique com o botão direito do mouse na conexão SecretsManagerTutoriale escolha Editar conexão.
2. Na caixa de diálogo Manage Server Connections (Gerenciar conexões do servidor), em Username (Nome do usuário), insira **`appuser`** e, depois, escolha Close (Fechar).
3. De volta ao MySQL Workbench, escolha a conexão. SecretsManagerTutorial
4. Na caixa de diálogo Open SSH Connection (Abrir conexão de SSH), em Password (Senha), cole a senha que você recuperou do segredo e escolha OK.

   Se as credenciais forem válidas, o MySQL Workbench será aberto na página de design do banco de dados.

Isso mostra que a alternância secreta é bem-sucedida. As credenciais no segredo foram atualizadas, e ele é uma senha válida para se conectar ao banco de dados.

Etapa 4: limpar os recursos

Se você quiser tentar outra estratégia de alternância, a single user rotation (alternância de usuários alternados), pule a limpeza de recursos e vá para the section called “Alternâncias de usuário único”.

Ou então, para evitar possíveis cobranças e remover a instância do EC2 que tem acesso à Internet, exclua os seguintes recursos criados neste tutorial e seus pré-requisitos:

- Instância de bancos de dados do Amazon RDS. Para obter instruções, consulte Deleting a DB instance (Como excluir uma instância de banco de dados) no Amazon RDS User Guide (Guia do usuário do Amazon RDS).
- Instância do Amazon EC2. Para obter instruções, consulte Encerrar uma instância, no Guia do usuário do Amazon EC2.
- Segredo `SecretsManagerTutorialAppuser` do Secrets Manager. Para instruções, consulte the section called “Excluir um segredo”.
- Endpoint do Secrets Manager. Para obter instruções, consulte Delete a VPC endpoint (Excluir um endpoint da VPC) no AWS PrivateLink Guide (Guia da).
- VPC endpoint. Para obter instruções, consulte Delete your VPC (Excluir sua VPC) no AWS PrivateLink Guide (Guia da).

Próximas etapas

- Saiba como recuperar segredos em suas aplicações.
- Saiba mais sobre outras programações de alternância.

Configure a rotação de um único usuário para AWS Secrets Manager

Neste tutorial, você aprenderá como configurar a alternância de usuário único para um segredo que contenha credenciais de banco de dados. Alternância de usuário único é uma estratégia de alternância na qual o Secrets Manager atualiza as credenciais de um usuário no segredo e no banco de dados. Para obter mais informações, consulte the section called “Usuário único”.

Depois de concluir o tutorial, recomendamos limpar os recursos do tutorial. Não os use em um ambiente de produção.

A rotação do Secrets Manager usa uma AWS Lambda função para atualizar o segredo e o banco de dados. Para obter mais informações sobre os custos do uso de uma função do Lambda, consulte Preços.

Sumário

Permissões

Para os pré-requisitos do tutorial, você precisa de permissões administrativas para sua Conta da AWS. Em uma configuração de produção, é uma prática recomendada usar funções diferentes para cada uma das etapas. Por exemplo, uma função com permissões de administrador de banco de dados criaria o banco de dados do Amazon RDS, e uma função com permissões de administrador de rede configuraria a VPC e os grupos de segurança. Para as etapas do tutorial, recomendamos que você continue usando a mesma identidade.

Para obter mais informações sobre como configurar permissões em um ambiente de produção, consulte the section called “Autenticação e controle de acesso”.

Pré-requisitos

O pré-requisito para este tutorial é the section called “Alternância de usuários alternados”. Não limpe os recursos no final do primeiro tutorial. Depois desse tutorial, você tem um ambiente realista com um banco de dados do Amazon RDS e um segredo do Secrets Manager que contém credenciais de administrador para o banco de dados. Você também tem um segundo segredo que contém credenciais para um usuário do banco de dados, mas você não usa esse segredo neste tutorial.

Você também tem uma conexão configurada no MySQL Workbench para se conectar ao banco de dados com as credenciais de administrador.

Etapa 1: criar um usuário do banco de dados do Amazon RDS

Primeiro, você precisa de um usuário cujas credenciais serão armazenadas no segredo. Para criar o usuário, faça login no banco de dados do Amazon RDS com credenciais de administrador que são armazenadas em um segredo. Para simplificar, no tutorial, você cria um usuário com permissão total para um banco de dados. Em um ambiente de produção, isso não é típico. Recomendamos que você siga o princípio de privilégio mínimo.

Para recuperar a senha do administrador

1. No Console do Amazon RDS, navegue até o seu banco de dados.
2. Na guia Configuration (Configuração), em Master Credentials ARN (ARN das credenciais principais), escolha Manage in Secrets Manager (Gerenciar no Secrets Manager).

   O Console do Secrets Manager é aberto.
3. Na página de detalhes do seu segredo, escolha Retrieve secret value (Recuperar valor do segredo).
4. A senha aparece na seção Secret value (Valor do segredo).

Para criar um usuário de banco de dados

1. No MySQL Workbench, clique com o botão direito do mouse na conexão SecretsManagerTutoriale escolha Editar conexão.
2. Na caixa de diálogo Manage Server Connections (Gerenciar conexões do servidor), em Username (Nome do usuário), insira **`admin`** e, depois, escolha Close (Fechar).
3. De volta ao MySQL Workbench, escolha a conexão. SecretsManagerTutorial
4. Digite a senha de administrador que você recuperou do segredo.
5. Em MySQL Workbench, na janela Query (Consultar), insira os seguintes comandos (incluindo uma senha forte) e, depois, escolha Execute (Executar). A função de alternância testa o segredo atualizado usando SELECT, portanto, **`dbuser`** deve ter esse privilégio, no mínimo.

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

   Na janela Output (Saída), você verá os comandos com êxito.

Etapa 2: criar um segredo para as credenciais do usuário do banco de dados

Em seguida, crie um segredo para armazenar as credenciais do usuário que acabou de criar e ative a alternância automática, incluindo uma alternância imediata. O Secrets Manager alterna o segredo, o que significa que a senha é gerada programaticamente, ou seja, nenhum ser humano viu essa

nova senha. O início imediato da alternância também pode ajudar a determinar se a alternância está configurada corretamente.

1. Abra o console do Secrets Manager em https://console.aws.amazon.com/secretsmanager/.
2. Selecione Armazenar um novo segredo.
3. Na página Choose secret type (Selecionar tipo de segredo), faça o seguinte:
   a. Em Secret type (Tipo de segredo), escolha Credentials for Amazon RDS database (Credenciais para o banco de dados do Amazon RDS).
   b. Em Credentials (Credenciais), insira o nome do usuário **`dbuser`** e a senha inserida para o usuário do banco de dados que você criou usando o MySQL Workbench.
   c. Em Database (Banco de dados), escolha secretsmanagertutorialdb.
   d. Escolha Próximo.
4. Na página Configure secret (Configurar segredo) , em Secret name (Nome de segredo), insira **`SecretsManagerTutorialDbuser`** e, depois, escolha Next (Próximo).
5. Na página Configure rotation (Configurar alternância), faça o seguinte:
   a. Ative a Automatic rotation (Alternância automática).
   b. Em Rotation schedule (Programação da alternância), defina uma programação de Days (Dias):**2** dias com Duration (Duração): **2h**. Mantenha Rotate immediately (Alternar imediatamente) selecionado.
   c. Em Rotation function (Função de alternância), escolha Create a rotation function (Criar uma função de alternância), e, em seguida, para nome da função, insira **`tutorial-single-user-rotation`**.
   d. Em Estratégia de alternância, escolha Usuário único.
   e. Escolha Próximo.
6. Na página Review (Análise), escolha Store (Armazenar).

   O Secrets Manager retorna à página de detalhes do segredo. Na parte superior da página, é possível ver o status da configuração de alternância. O Secrets Manager usa CloudFormation para criar recursos como a função de rotação Lambda e uma função de execução que executa a função Lambda. Quando CloudFormation terminar, o banner muda para Segredo programado para rotação. A primeira alternância está completa.

Etapa 3: testar a senha alternada

Após a primeira alternância de segredo, que pode levar alguns segundos, é possível verificar se o segredo ainda contém credenciais válidas. A senha no segredo mudou a partir das credenciais originais.

Para recuperar a nova senha do segredo

1. Abra o console do Secrets Manager em https://console.aws.amazon.com/secretsmanager/.
2. Escolha Secrets (Segredos) e, em seguida, escolha o segredo **`SecretsManagerTutorialDbuser`**.
3. Na página Secret details (Detalhes do segredo), role para baixo e escolha Retrieve secret value (Recuperar valor do segredo).
4. Na tabela Key/value (Chave/valor), copie o Secret value (Valor do segredo) para **`password`**.

Para testar as credenciais

1. No MySQL Workbench, clique com o botão direito do mouse na conexão SecretsManagerTutoriale escolha Editar conexão.
2. Na caixa de diálogo Manage Server Connections (Gerenciar conexões do servidor), em Username (Nome do usuário), insira **`dbuser`** e, depois, escolha Close (Fechar).
3. De volta ao MySQL Workbench, escolha a conexão. SecretsManagerTutorial
4. Na caixa de diálogo Open SSH Connection (Abrir conexão de SSH), em Password (Senha), cole a senha que você recuperou do segredo e escolha OK.

   Se as credenciais forem válidas, o MySQL Workbench será aberto na página de design do banco de dados.

Etapa 4: Limpar os recursos

Para evitar possíveis cobranças, exclua o segredo que você criou neste tutorial. Para instruções, consulte the section called “Excluir um segredo”.

Para limpar os recursos criados no tutorial anterior, consulte the section called “Etapa 4: limpar os recursos”.

Próximas etapas

- Saiba como recuperar segredos em suas aplicações. Consulte Obter segredos.
- Saiba mais sobre outras programações de alternância. Consulte the section called “Programação de alternância”.

Crie um AWS Secrets Manager segredo

Um segredo pode ser uma senha, um conjunto de credenciais, como nome de usuário e senha, um OAuth token ou outras informações secretas que você armazena de forma criptografada no Secrets Manager.

> **Tip**
>
> Para credenciais de usuário administrador do Amazon RDS e do Amazon Redshift, recomendamos o uso de segredos gerenciados. Você cria o segredo gerenciado por meio do serviço de gerenciamento e, em seguida, pode usar a alternância gerenciada.

Quando você usa o console para armazenar credenciais de banco de dados para um banco de dados de origem replicado para outras regiões, o segredo contém informações de conexão para o banco de dados de origem. Se você replicar o segredo, as réplicas serão cópias do segredo de origem e conterão as mesmas informações de conexão. Você pode adicionar mais key/value pares ao segredo para obter informações sobre a conexão regional.

Para criar um segredo, você precisa das permissões concedidas pela política SecretsManagerReadWrite gerenciada.

O Secrets Manager gera uma entrada de CloudTrail registro quando você cria um segredo. Para obter mais informações, consulte the section called “Faça login com AWS CloudTrail ”.

Para criar um segredo (console)

1. Abra o console do Secrets Manager em https://console.aws.amazon.com/secretsmanager/.
2. Selecione Armazenar um novo segredo.
3. Na página Selecionar tipo de segredo, faça o seguinte:

    a. Para Secret type (Tipo de segredo), siga um destes procedimentos:

    - Para armazenar credenciais de banco de dados, escolha o tipo de credenciais de banco de dados a armazenar. Em seguida, escolha o Banco de dados e insira as Credenciais.
    - Para armazenar chaves de APIs, tokens de acesso e credenciais que não sejam para bancos de dados, escolha Outro tipo de segredo.

Em Key/value pairs (Pares de chave/valor), ou insira seu segredo no JSON Key/value (Chave/valor), ou escolha a guia Plaintext (Texto simples) e insira o segredo em qualquer formato. É possível armazenar até 65536 bytes no segredo. Alguns exemplos:

API key

Insira como key/value pares:

`ClientID` : *`my_client_id`*

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

OAuth token

Insira como texto sem formatação:

`AKIAI44QH8DHBEXAMPLE`

Digital certificate

Insira como texto sem formatação:

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

Private key

Insira como texto sem formatação:

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

- Para armazenar segredos externos gerenciados de um parceiro do Secrets Manager, escolha Segredo do parceiro. Em seguida, escolha o parceiro e forneça os detalhes que identificam o segredo do parceiro. Para obter detalhes, consulte Usando segredos externos AWS Secrets Manager gerenciados para gerenciar segredos de terceiros.

b. Em Chave de criptografia, escolha a AWS KMS key que o Secrets Manager usa para criptografar o valor secreto. Para obter mais informações, consulte Criptografia e descriptografia de segredos.

- Na maioria dos casos, escolha aws/secretsmanager para usar o for Secrets Chave gerenciada pela AWS Manager. Não há custo para o uso dessa chave.
- Se você precisar acessar o segredo de outra pessoa Conta da AWS ou se quiser usar sua própria chave KMS para poder alterná-la ou aplicar uma política de chaves a ela, escolha uma chave gerenciada pelo cliente na lista ou escolha Adicionar nova chave para criar uma. Para obter mais informações sobre os custos do uso de uma chave gerenciada pelo cliente, consulte Preços.

  É necessário ter the section called “Permissões para a chave do KMS”. Para obter informações sobre o acesso entre contas, consulte the section called “Acesso entre contas”.

c. Escolha Próximo.

4. Na página Configure secret (Configurar segredo), faça o seguinte:

   a. Insira um Secret name (Nome de segredo) descritivo e uma Description (Descrição). Os nomes dos segredos podem conter de 1 a 512 caracteres alfanuméricos e os caracteres /_+ =.@-.

   b. (Opcional) Se você criou um segredo externo, insira os metadados exigidos pelo parceiro do Secrets Manager que detém o segredo.

   c. (Opcional) Na seção Tags, adicione tags ao segredo. Para conhecer as estratégias de marcação, consulte the section called “Marcação de segredos do ”. Não armazene informações sigilosas em tags porque elas não são criptografadas.

   d. (Opcional) Em Resource permissions (Permissões do recurso), para adicionar uma política de recursos ao segredo, escolha Edit permissions (Editar permissões). Para obter mais informações, consulte the section called “Políticas baseadas em recursos”.

   e. (Opcional) Em Replicar segredo, para replicar seu segredo para outro Região da AWS, escolha Replicar segredo. É possível replicar seu segredo agora ou voltar e replicá-lo mais tarde. Para obter mais informações, consulte Replicação multirregional.

   f. Escolha Próximo.

5. (Opcional) Na página Configure rotation (Configurar alternância), habilite alternância automática para os segredos. Você também pode manter a alternância desabilitada por enquanto e habilitá-la mais tarde. Para obter mais informações, consulte Alternar segredos . Escolha Próximo.

6. Na página Review (Revisar), revise os detalhes do segredo e escolha Store (Armazenar).

O Secrets Manager retorna para a lista de segredos. Se o segredo não aparecer, escolha Refresh (Atualizar).

AWS CLI

Quando você insere comandos em um shell de comando, existe o risco de o histórico de comandos ser acessado ou de utilitários terem acesso aos seus parâmetros de comando. Consulte the section called “Mitigue os riscos de usar o AWS CLI para armazenar seus segredos AWS Secrets Manager”.

Example Crie um segredo com base nas credenciais do banco de dados em um arquivo JSON

O exemplo `create-secret` a seguir cria um segredo com base em credenciais em um arquivo. Para obter mais informações, consulte Carregando AWS CLI parâmetros de um arquivo no Guia AWS CLI do usuário.

Para que o Secrets Manager possa alternar o segredo, é necessário se certificar de que o JSON corresponde a Estrutura JSON de um segredo.

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

Conteúdo 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 Criar um segredo

O seguinte exemplo de `create-secret` cria um segredo com dois pares de chave/valor.

```
aws secretsmanager create-secret \
    --name MyTestSecret \
```

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

Example Criar um segredo

O exemplo de `create-secret` a seguir cria um segredo com duas tags.

```
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

Para criar um segredo usando um dos AWS SDKs, use a `CreateSecret`ação. Para obter mais informações, consulte the section called “AWS SDKs”.

O que há em um segredo do Secrets Manager?

No Secrets Manager, um segredo são informações de segredos, isto é, o valor do segredo mais alguns metadados sobre o segredo. Um valor do segredo pode ser uma string ou um binário.

Para armazenar vários valores de strings em um segredo, recomendamos usar uma string de texto JSON com pares de chave-valor, por exemplo:

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

Para segredos de bancos de dados, se você quiser ativar a alternância automática, o segredo deverá conter as informações da conexão do banco de dados na estrutura JSON correta. Para obter mais informações, consulte the section called “Estrutura JSON de um segredo”.

Metadados

Os metadados de um segredo incluem:

- Um nome do recurso da Amazon (ARN) com o seguinte formato:

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

  O Secrets Manager inclui seis caracteres aleatórios ao final do nome do segredo para ajudar a garantir que o ARN do segredo seja único. Se o segredo original for excluído e um novo segredo for criado com o mesmo nome, os dois segredos serão diferentes ARNs por causa desses caracteres. Os usuários com acesso ao segredo antigo não obtêm acesso automático ao novo segredo porque ARNs são diferentes.
- O nome do segredo, uma descrição, uma política de recursos e tags.
- O ARN de uma chave de criptografia, e AWS KMS key que o Secrets Manager usa para criptografar e descriptografar o valor secreto. O Secrets Manager sempre armazena o texto do segredo de forma criptografada e criptografa o segredo em trânsito. Consulte the section called “Criptografia e descriptografia de segredos”.
- Informações sobre como alternar o segredo, se você configurar a alternância. Consulte Alternar segredos .

O Secrets Manager usa políticas de permissões do IAM para garantir que apenas usuários autorizados possam acessar ou modificar um segredo. Consulte Autenticação e controle de acesso para AWS Secrets Manager.

Um segredo tem versões que contêm cópias do valor do segredo criptografado. Quando você altera o valor do segredo, ou o segredo é alternado, o Secrets Manager cria uma nova versão. Consulte the section called “Versões do segredo”.

Você pode usar um segredo em vários Regiões da AWS replicando-o. Quando você replica um segredo, você cria uma cópia do segredo primário ou original, chamado de segredo de réplica. O segredo de réplica permanece vinculado ao segredo primário. Consulte Replicação multirregional.

Consulte Gerenciar segredos.

Versões do segredo

Um segredo tem versões que contêm cópias do valor do segredo criptografado. Quando você altera o valor do segredo, ou o segredo é alternado, o Secrets Manager cria uma nova versão.

O Secrets Manager não armazena um histórico linear de segredos com versões. Em vez disso, ele rastreia três versões específicas, rotulando-as:

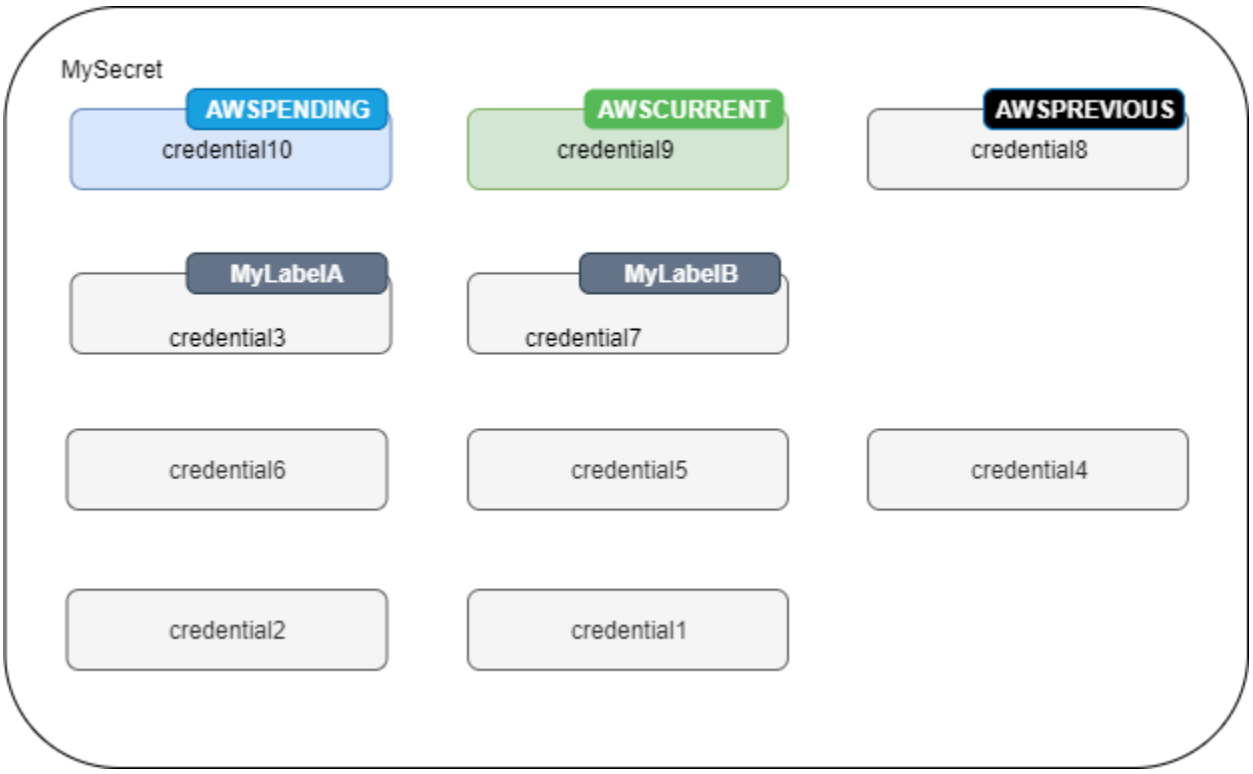
- A versão atual: `AWSCURRENT`
- A versão anterior: `AWSPREVIOUS`
- A versão pendente (durante a alternância): `AWSPENDING`

Um segredo sempre tem uma versão rotulada `AWSCURRENT`, e o Secrets Manager retorna essa versão por padrão quando você recupera o valor do segredo.

Você também pode rotular versões com suas próprias etiquetas `update-secret-version-stage`ligando para AWS CLI o. É possível anexar até 20 rótulos a versões em um segredo. Duas versões do segredo não podem ter o mesmo rótulo de preparação. As versões podem ter vários rótulos.

O Secrets Manager nunca remove as versões rotuladas, mas as versões sem rótulos são consideradas obsoletas. O Secrets Manager remove versões obsoletas quando há mais de 100. O Secrets Manager não remove versões criadas há menos de 24 horas.

A figura a seguir mostra um segredo que tem versões AWS rotuladas e versões rotuladas pelo cliente. As versões sem rótulos são consideradas obsoletas e serão removidas pelo Secrets Manager em algum momento no future.



Estrutura JSON de segredos AWS Secrets Manager

É possível armazenar qualquer texto ou binário em um segredo do Secrets Manager, até o tamanho máximo de 65.536 bytes.

Se você usar the section called “Função do Lambda de alternância”, um segredo deverá conter campos JSON específicos que a função de alternância espera. Por exemplo, para um segredo que contenha credenciais do banco de dados, a função de alternância se conecta ao banco de dados para atualizar as credenciais, portanto, o segredo deverá conter as informações de conexão do banco de dados.

Se você usar o console para editar a alternância de um segredo de banco de dados, o segredo deverá conter pares de valores-chave JSON específicos que identifiquem o banco de dados. O Secrets Manager usa esses campos para consultar o banco de dados e encontrar a VPC correta na qual armazenar uma função de alternância.

Os nomes de chaves no JSON diferenciam maiúsculas de minúsculas.

Tópicos

- Credenciais do Amazon RDS e do Aurora

- Credenciais do Amazon Redshift
- Credenciais do Amazon Redshift sem servidor
- Credenciais do Amazon DocumentDB
- Estrutura de segredo do Amazon Timestream para InfluxDB
- ElastiCache Credenciais da Amazon
- Credenciais do Active Directory

Credenciais do Amazon RDS e do Aurora

Para usar os modelos de função de alternância fornecidos pelo Secrets Manager, use a estrutura JSON a seguir. Você pode adicionar mais key/value pares, por exemplo, para conter informações de conexão para bancos de dados de réplica em outras regiões.

DB2

Para instâncias do Db2 do Amazon RDS, como os usuários não podem alterar suas próprias senhas, é necessário fornecer credenciais de administrador em outro segredo.

```
{
  "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 “Usuários alternados”.>",
  "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 “Usuários alternados”.>",
  "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 “Usuários alternados”.>",
  "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 “Usuários alternados”.>",
  "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 “Usuários alternados”.>",
  "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.>"
}
```

Credenciais do Amazon Redshift

Para usar os modelos de função de alternância fornecidos pelo Secrets Manager, use a estrutura JSON a seguir. Você pode adicionar mais key/value pares, por exemplo, para conter informações de conexão para bancos de dados de réplica em outras regiões.

```
{
```

```
  "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 “Usuários alternados”.>"
}
```

Credenciais do Amazon Redshift sem servidor

Para usar os modelos de função de alternância fornecidos pelo Secrets Manager, use a estrutura JSON a seguir. Você pode adicionar mais key/value pares, por exemplo, para conter informações de conexão para bancos de dados de réplica em outras regiões.

```
{
  "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 “Usuários alternados”.>"
}
```

Credenciais do Amazon DocumentDB

Para usar os modelos de função de alternância fornecidos pelo Secrets Manager, use a estrutura JSON a seguir. Você pode adicionar mais key/value pares, por exemplo, para conter informações de conexão para bancos de dados de réplica em outras regiões.

```
{
  "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 “Usuários alternados”.>",
  "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.>"
}
```

Estrutura de segredo do Amazon Timestream para InfluxDB

Para alternar os segredos do Timestream, é possível usar os modelos de alternância the section called “Amazon Timestream para InfluxDB”.

Para obter mais informações, consulte Como o Amazon Timestream para InfluxDB usa segredos no Guia do desenvolvedor do Amazon Timestream.

Os segredos do Timestream devem estar na estrutura JSON correta para poderem usar os modelos de alternância. Para obter mais informações, consulte O que há no segredo no Guia do desenvolvedor do Amazon Timestream.

ElastiCache Credenciais da Amazon

O exemplo a seguir mostra a estrutura JSON de um segredo que armazena ElastiCache credenciais.

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

Para obter mais informações, consulte Rotação automática de senhas para usuários no Guia do ElastiCache usuário da Amazon.

Credenciais do Active Directory

AWS Directory Service usa segredos para armazenar credenciais do Active Directory. Para obter mais informações, consulte Associar perfeitamente uma instância Linux do Amazon EC2

ao seu Active Directory AD gerenciado no Guia de administração do AWS Directory Service . A associação perfeita ao domínio requer os nomes de chave nos exemplos a seguir. Se você não usar a associação de domínio perfeita, poderá alterar os nomes das chaves no segredo usando variáveis de ambiente, conforme descrito no código do modelo da função de alternância.

Para alternar segredos do Active Directory, é possível usar os modelos de alternância do Active Directory.

Active Directory credential

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

Se você quiser alternar o segredo, inclua o ID do diretório do domínio.

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

Se o segredo for usado em conjunto com um segredo que contém um keytab, você inclui o segredo 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

Para obter informações sobre o uso de arquivos keytab para autenticação de contas do Active Directory no Amazon EC2, consulte Implantação e configuração da autenticação do Active Directory com o SQL Server 2017 no 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
}
```

Gerencie segredos com AWS Secrets Manager

Tópicos

Atualizar o valor de um AWS Secrets Manager segredo

Para atualizar o valor da sua chave secreta, você pode usar o console, a CLI ou um SDK. Quando você atualiza o valor do segredo, o Secrets Manager cria uma nova versão do segredo com uma rótulo de preparação `AWSCURRENT`. Você ainda pode acessar a versão antiga, que tem o rótulo da `AWSPREVIOUS`. Você também pode adicionar seus próprios rótulos. Para obter mais informações, consulte Secrets Manager versioning (Versionamento do Secrets Manager).

Para atualizar o valor do segredo (console)

1. Abra o console do Secrets Manager em https://console.aws.amazon.com/secretsmanager/.
2. Na lista de segredos, escolha o segredo.
3. Na página de detalhes do segredo, na guia Visão geral, na seção Valor do segredo, escolha Recuperar o valor do segredo e, em seguida, Editar.

AWS CLI

Para atualizar o valor do segredo (AWS CLI)

- Quando você insere comandos em um shell de comando, existe o risco de o histórico de comandos ser acessado ou de utilitários terem acesso aos seus parâmetros de comando. Consulte the section called “Mitigue os riscos de usar o AWS CLI para armazenar seus segredos AWS Secrets Manager”.

  O seguinte `put-secret-value` cria uma nova versão de um segredo com dois pares de chave/valor.

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

  O seguinte `put-secret-value` cria uma nova versão com um rótulo de teste personalizado. A nova versão terá os rótulos `MyLabel` e `AWSCURRENT`.

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

AWS SDK

Recomendamos que você evite chamar `PutSecretValue` ou `UpdateSecret` em uma frequência sustentada de mais de uma vez a cada dez minutos. Quando você chama `PutSecretValue` ou `UpdateSecret` para atualizar o valor do segredo, o Secrets Manager cria uma nova versão do segredo. O Secrets Manager remove versões sem rótulo quando há mais de 100, mas não remove versões criadas há menos de 24 horas. Se você atualizar o valor do segredo mais de uma vez a cada dez minutos, criará mais versões do que o Secrets Manager remove e atingirá a cota de versões de segredos.

Para atualizar o valor de um segredo, utilize as seguintes ações: `UpdateSecret` ou `PutSecretValue`. Para obter mais informações, consulte the section called “AWS SDKs”.

Gere uma senha com o Secrets Manager

Um padrão comum para usar o Secrets Manager é gerar uma senha no Secrets Manager e depois usar essa senha em seu banco de dados ou serviço. Isso pode ser feito usando um dos métodos a seguir:

- CloudFormation — VejaCloudFormation.
- AWS CLI — Veja `get-random-password`.
- AWS SDKs — Veja `GetRandomPassword`.

Reverter um segredo para uma versão anterior

É possível reverter um segredo para uma versão anterior movendo os rótulos anexados às versões secretas usando a AWS CLI. Para obter informações sobre como o Secrets Manager armazena versões de segredos, consulte the section called “Versões do segredo”.

O `update-secret-version-stage`exemplo a seguir move o rótulo AWSCURRENT de teste para a versão anterior de um segredo, o que reverte o segredo para a versão anterior. Para encontrar o ID da versão anterior, use `list-secret-version-ids` ou visualize as versões no console do Secrets Manager.

Neste exemplo, a versão com o rótulo é a1b2c3d4-5678-90ab-cdef- e a versão com o AWSCURRENT rótulo é a1b2c3d4-5678-90ab-cdef-. EXAMPLE11111 AWSPREVIOUS EXAMPLE22222 Neste exemplo, você move o AWSCURRENT rótulo da versão 11111 para 22222. Como o AWSCURRENT rótulo é removido de uma versão, move `update-secret-version-stage` automaticamente o AWSPREVIOUS rótulo para essa versão (11111). O efeito é que as AWSPREVIOUS versões AWSCURRENT e são trocadas.

```
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
```

Troque a chave de criptografia por um AWS Secrets Manager segredo

O Secrets Manager usa criptografia de envelope com AWS KMS chaves e chaves de dados para proteger cada valor secreto. Para cada segredo, é possível escolher qual chave KMS usar. Você pode usar o Chave gerenciada pela AWS aws/secretsmanager, ou você pode usar uma chave gerenciada pelo cliente. Na maioria dos casos, recomendamos usar a aws/secretsmanager, e não há custo para usá-la. Se você precisar acessar o segredo de outra pessoa Conta da AWS ou se quiser usar sua própria chave KMS para poder alterná-la ou aplicar uma política de chaves a ela, use a. chave gerenciada pelo clienteÉ necessário ter the section called “Permissões para a chave do KMS”. Para obter mais informações sobre os custos do uso de uma chave gerenciada pelo cliente, consulte Preços.

É possível alterar a chave de criptografia de um segredo. Por exemplo, se você quiser acessar o segredo de outra conta e o segredo estiver atualmente criptografado usando a chave AWS gerenciadaaws/secretsmanager, você pode mudar para uma chave gerenciada pelo cliente.

> **Tip**
>
> Se você quiser girar seu chave gerenciada pelo cliente, recomendamos usar a rotação AWS KMS automática de chaves. Para obter mais informações, consulte AWS KMS Chaves giratórias.

Quando você altera a chave de criptografia, o Secrets Manager criptografa novamente as versões `AWSCURRENT`, `AWSPENDING` e `AWSPREVIOUS` com a nova chave. Para evitar que você fique bloqueado sem o segredo, o Secrets Manager mantém todas as versões existentes criptografadas com a chave anterior. Isso significa que é possível descriptografar as versões `AWSCURRENT`, `AWSPENDING` e `AWSPREVIOUS` com a chave anterior ou com a nova chave. Se você não tiver a permissão `kms:Decrypt` para a chave anterior, ao alterar a chave de criptografia, o Secrets Manager não poderá descriptografar as versões do segredo para recriptografá-las. Nesse caso, as versões existentes não serão criptografadas novamente.

Para fazer com que `AWSCURRENT` só possa ser descriptografado pela nova chave de criptografia, crie uma nova versão do segredo com a nova chave. Em seguida, para poder decifrar a versão do segredo `AWSCURRENT`, será necessário ter permissão para a nova chave.

Se você desativar a chave de criptografia anterior, não poderá descriptografar nenhuma versão secreta, exceto `AWSCURRENT`, `AWSPENDING` e `AWSPREVIOUS`. Se você tiver outras versões secretas rotuladas às quais deseja manter o acesso, precisará recriar essas versões com a nova chave de criptografia usando o the section called “AWS CLI”.

Para alterar a chave de criptografia de um segredo (console)

1. Abra o console do Secrets Manager em https://console.aws.amazon.com/secretsmanager/.
2. Na lista de segredos, escolha o segredo.
3. Na página de detalhes do segredo, na seção Secrets details (Detalhes dos segredos), selecione Actions (Ações) e, em seguida, selecione Edit encryption key (Editar chave de criptografia).

AWS CLI

Se você alterar a chave de criptografia de um segredo e, em seguida, desativar a chave de criptografia anterior, você não conseguirá descriptografar nenhuma versão do segredo, exceto `AWSCURRENT`, `AWSPENDING` e `AWSPREVIOUS`. Se você tiver outras versões secretas rotuladas às quais deseja manter o acesso, precisará recriar essas versões com a nova chave de criptografia usando o the section called “AWS CLI”.

Para alterar a chave de criptografia de um segredo, consulte (AWS CLI)

1. O exemplo de `update-secret` a seguir atualiza a chave do KMS usada para criptografar o valor do segredo. A chave do KMS precisa estar na mesma do segredo.

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

2. (Opcional) Se você tiver versões de segredos com rótulos personalizados, para recriptografá-las usando a nova chave, será necessário recriar essas versões.

   Quando você insere comandos em um shell de comando, existe o risco de o histórico de comandos ser acessado ou de utilitários terem acesso aos seus parâmetros de comando. Consulte the section called “Mitigue os riscos de usar o AWS CLI para armazenar seus segredos AWS Secrets Manager”.

   a. Obtenha o valor da versão secreta.

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

Anote o valor do segredo.

b. Crie uma nova versão com esse valor.

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

Modificar um AWS Secrets Manager segredo

Dependendo de quem criou o segredo, você pode modificar os metadados de um segredo depois que ele for criado. Para segredos criados por outros serviços, talvez seja necessário usar o outro serviço para atualizar ou alternar o segredo.

Para determinar quem gerencia um segredo, você pode revisar o nome do segredo. Os segredos gerenciados por outros serviços são prefixados com o ID do respectivo serviço. Ou, no AWS CLI, chame describe-secret e revise o campo. `OwningService` Para obter mais informações, consulte Segredos gerenciados por outros serviços.

Para os segredos que você gerencia, é possível modificar a descrição, a política baseada em recursos, a chave de criptografia e as etiquetas. Você também pode alterar o valor do segredo criptografado; entretanto, recomendamos usar alternância para atualizar os valores dos segredo que contêm credenciais. A alternância atualiza o segredo no Secrets Manager e as credenciais no banco de dados ou serviço. Isso mantém o segredo sincronizado automaticamente para que, quando solicitarem um valor de segredo, os clientes sempre obtenham um conjunto de credenciais que funcionam. Para obter mais informações, consulte Alternar segredos .

O Secrets Manager gera uma entrada de CloudTrail registro quando você modifica um segredo. Para obter mais informações, consulte the section called “Faça login com AWS CloudTrail ”.

Para atualizar um segredo que você gerencia (console)

1. Abra o console do Secrets Manager em https://console.aws.amazon.com/secretsmanager/.

2. Na lista de segredos, escolha o segredo.
3. Na página de detalhes do segredo, siga um destes procedimentos:

   Observe que você não pode alterar o nome ou o ARN de um segredo.

   - Para atualizar a descrição, na seção Secrets details (Detalhes dos segredos), escolha Actions (Ações) e, em seguida, Edit description (Editar descrição).
   - Para atualizar a chave de criptografia, consulte the section called “Altere a chave de criptografia de um segredo”.
   - Para atualizar etiquetas, na guia Etiquetas, escolha Editar etiquetas. Consulte the section called “Marcação de segredos do ”.
   - Para atualizar o valor do segredo, consulte the section called “Atualize o valor do segredo”.
   - Para atualizar permissões do seu segredo, na guia Visão geral, escolha Editar permissões. Consulte the section called “Políticas baseadas em recursos”.

   - Para atualizar a alternância do seu segredo, na guia Alternância, escolha Editar alternância. Consulte Alternar segredos .
   - Para replicar seu segredo para outras regiões, consulte Replicação multirregional.
   - Se seu segredo tiver réplicas, você pode trocar a chave de criptografia de uma réplica. Na guia Replicação, selecione o botão de opção para a réplica e, em seguida, no menu Ações, selecione Editar chave de criptografia. Consulte the section called “Criptografia e descriptografia de segredos”.
   - Para alterar um segredo de modo que ele seja gerenciado por outro serviço, você precisa recriar o segredo no respectivo serviço. Consulte Segredos gerenciados por outros serviços.

AWS CLI

Example Atualizar descrição do segredo

O exemplo de `update-secret` a seguir retorna a descrição de um segredo.

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

AWS SDK

Recomendamos que você evite chamar `PutSecretValue` ou `UpdateSecret` em uma frequência sustentada de mais de uma vez a cada dez minutos. Quando você chama `PutSecretValue` ou `UpdateSecret` para atualizar o valor do segredo, o Secrets Manager cria uma nova versão do segredo. O Secrets Manager remove versões sem rótulo quando há mais de 100, mas não remove versões criadas há menos de 24 horas. Se você atualizar o valor do segredo mais de uma vez a cada dez minutos, criará mais versões do que o Secrets Manager remove e atingirá a cota de versões de segredos.

Para atualizar um segredo, use as seguintes ações: `UpdateSecret` ou `ReplicateSecretToRegions`. Para obter mais informações, consulte the section called “AWS SDKs”.

Encontre segredos em AWS Secrets Manager

Quando você pesquisa por segredos sem um filtro, o Secrets Manager faz correspondências com base em palavras-chave com o nome do segredo, descrição, chave de tag e valor da tag. Pesquisar sem filtros não diferencia maiúsculas de minúsculas e ignora caracteres especiais, como espaço, /, _, =, #, e usa apenas números e letras. Quando você pesquisa sem um filtro, o Secrets Manager analisa a string de pesquisa para convertê-la em palavras separadas. As palavras são separadas por qualquer alteração de maiúsculas para minúsculas, de letra para número ou de pontuação. number/letter Por exemplo, inserir o termo de pesquisa `credsDatabase#892` gera a pesquisa por `creds`, `Database` e 892 no nome, descrição e chave de tag e valor.

O Secrets Manager gera uma entrada de CloudTrail registro quando você lista segredos. Para obter mais informações, consulte the section called “Faça login com AWS CloudTrail ”.

O Secrets Manager é um serviço regional e apenas segredos na região selecionada são retornados.

Filtros de pesquisa

Se você não usar nenhum filtro, o Secrets Manager dividirá a string de pesquisa em palavras e, em seguida, pesquisará correspondências em todos os atributos. Essa pesquisa não diferencia maiúsculas de minúsculas. Por exemplo, pesquisar por **`My_Secret`** encontra correspondência com segredos com a palavra my ou secret no nome, na descrição ou nas tags.

É possível aplicar os seguintes filtros para sua pesquisa:

Nome

Encontra correspondências com o início dos nomes de segredo; diferencia maiúsculas de minúsculas. Por exemplo, Name: **`Data`** retorna um segredo nomeado como `DatabaseSecret`, mas não `databaseSecret` ou `MyData`.

Description

Encontra correspondência nas palavras em descrições dos segredos, não diferencia letras maiúsculas de minúsculas. Por exemplo, Description: **`My Description`** encontra correspondências entre segredos com as seguintes descrições:

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

Gerenciado por

Encontra segredos gerenciados por serviços externos à AWS, por exemplo:

- 1Password
- Akeyless
- CyberArk
- HashiCorp

Serviço de propriedade

Faz correspondências com o início do prefixo do ID do serviço de gerenciamento, não diferencia maiúsculas de minúsculas. Por exemplo, **`my-ser`** faz a correspondência com segredos gerenciados por serviços com os prefixos `my-serv` e `my-service`. Para obter mais informações, consulte Segredos gerenciados por outros serviços.

Segredos replicados

É possível filtrar segredos primários, réplicas de segredo ou segredos que não estão replicados.

Chaves de tags

Encontra correspondências com o início das chaves de tag; diferencia maiúsculas de minúsculas. Por exemplo, Tag key: **`Prod`** retorna segredos com as tags `Production` e `Prod1`, mas não segredos com as tags `prod` ou `1 Prod`.

Valores de tags

Encontra correspondências com o início dos valores de tag; diferencia maiúsculas de minúsculas. Por exemplo, Tag value: **`Prod`** retorna segredos com as tags `Production` e `Prod1`, mas não segredos com valor de tag `prod` ou `1 Prod`.

AWS CLI

Example Listar os segredos em sua conta

O exemplo de `list-secrets` a seguir mostra uma lista dos segredos em sua conta.

```
aws secretsmanager list-secrets
```

Example Filtrar a lista de segredos em sua conta

O exemplo de `list-secrets` a seguir obtém uma lista dos segredos em sua conta que têm Test no nome. A filtragem por nome diferencia maiúsculas de minúsculas.

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

Example Encontre segredos que são gerenciados por outros AWS serviços

O exemplo de `list-secrets` a seguir obtém uma lista de segredos gerenciados por um serviço. Você especifica o ID do serviço. Para obter mais informações, consulte Segredos gerenciados por outros serviços.

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

AWS SDK

Para descobrir segredos usando um dos AWS SDKs, use `ListSecrets`. Para obter mais informações, consulte the section called “AWS SDKs”.

Excluir um AWS Secrets Manager segredo

Devido à natureza crítica dos segredos, dificulta AWS Secrets Manager intencionalmente a exclusão de um segredo. O Secrets Manager não exclui segredos de imediato. Em vez disso, o Secrets Manager torna os segredos imediatamente inacessíveis e programados para exclusão após uma janela de recuperação de, no mínimo, sete dias. Até que a janela de recuperação termine, é possível recuperar um segredo excluído anteriormente. Não há cobrança para segredos que você marcou para exclusão.

Você não pode excluir um segredo primário se ele for replicado para outras regiões. Primeiramente, exclua as réplicas e, em seguida, exclua o segredo primário. Quando você exclui uma réplica, ela é excluída imediatamente.

Não é possível excluir diretamente uma versão de um segredo. Em vez disso, você remove todos os rótulos de teste da versão usando o AWS CLI ou AWS SDK. Isso marca a versão como defasada e permite que o Secrets Manager exclua automaticamente a versão em segundo plano.

Se você não sabe se um aplicativo ainda usa um segredo, você pode criar um CloudWatch alarme da Amazon para alertá-lo sobre qualquer tentativa de acessar um segredo durante a janela de recuperação. Para obter mais informações, consulte Monitore quando AWS Secrets Manager os segredos programados para exclusão são acessados.

Para excluir um segredo, você precisa ter permissões do `secretsmanager:ListSecrets` e do `secretsmanager:DeleteSecret`.

O Secrets Manager gera uma entrada de CloudTrail registro quando você exclui um segredo. Para obter mais informações, consulte the section called “Faça login com AWS CloudTrail ”.

Para excluir um segredo (console)

1. Abra o console do Secrets Manager em https://console.aws.amazon.com/secretsmanager/.
2. Na lista de segredos, escolha o segredo que você quer excluir.
3. Na seção Secrets details (Detalhes dos segredos), escolha Actions (Ações) e, em seguida, escolha Delete secret (Excluir segredo).
4. Na caixa de diálogo Disable secret and schedule deletion (Desabilitar segredo e programar exclusão), em Waiting period (Período de espera), insira o número de dias de espera antes que a exclusão se torne permanente. O Secrets Manager anexa um campo denominado `DeletionDate` e o define como a data e hora atual e soma o número de dias especificado para a janela de recuperação.

5. Escolha Schedule deletion.

Para visualizar segredos excluídos

1. Abra o console do Secrets Manager em https://console.aws.amazon.com/secretsmanager/.
2. Na página Secrets (Segredos), escolha Preferences (Preferências) ( ).
3. Na caixa de diálogo Preferências, selecione Mostrar segredos programados para exclusão e, em seguida, escolha Salvar.

Para excluir um segredo de réplica

1. Abra o console do Secrets Manager em https://console.aws.amazon.com/secretsmanager/.
2. Selecione o segredo primário.
3. Na seção Replicate Secret (Replicar segredo), escolha o segredo de réplica.
4. No menu Actions (Ações), escolha Delete Replica (Excluir réplica).

AWS CLI

Example Excluir um segredo

O exemplo de `delete-secret` a seguir exclui um segredo. Você pode recuperar o segredo `restore-secret`até a data e a hora no campo de DeletionDate resposta. Para excluir um segredo que está replicado em outras regiões, primeiro remova suas réplicas com `remove-regions-from-replication` e então chame `delete-secret`.

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

Example Excluir um segredo imediatamente

O exemplo de `delete-secret` a seguir exclui imediatamente um segredo sem uma janela de recuperação. Não é possível recuperar esse segredo.

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

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

Example Excluir um segredo de réplica

O exemplo de `remove-regions-from-replication` a seguir exclui um segredo de réplica em eu-west-3. Para excluir um segredo primário que está replicado em outras regiões, primeiro remova as réplicas e então chame `delete-secret`.

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

AWS SDK

Para excluir um segredo, use o comando `DeleteSecret`. Para excluir uma versão de um segredo, use o comando `UpdateSecretVersionStage`. Para excluir uma réplica, use o comando `StopReplicationToReplica`. Para obter mais informações, consulte the section called “AWS SDKs”.

Restaurar um AWS Secrets Manager segredo

O Secrets Manager considera um segredo programado para exclusão como defasado e não pode mais ser acessado diretamente. Depois que a janela de recuperação passar, o Secrets Manager excluirá o segredo permanentemente. Depois que o Secrets Manager apaga o segredo, não é possível recuperá-lo. Antes do final da janela de recuperação, você pode recuperar o segredo e torná-lo acessível novamente. Isso remove o campo `DeletionDate`, que cancela a exclusão permanente programada.

Para restaurar um segredo e os metadados no console, é necessário ter as permissões `secretsmanager:ListSecrets` e `secretsmanager:RestoreSecret`.

O Secrets Manager gera uma entrada de CloudTrail registro quando você restaura um segredo. Para obter mais informações, consulte the section called “Faça login com AWS CloudTrail ”.

Para restaurar um segredo (console)

1. Abra o console do Secrets Manager em https://console.aws.amazon.com/secretsmanager/.
2. Na lista de segredos, escolha o segredo que você quer restaurar.

Se os segredos excluídos não constarem da lista de segredos, escolha Preferences (Preferências)
().
Na caixa de diálogo Preferências, selecione Mostrar segredos programados para exclusão e, em seguida, escolha Salvar.

3. Na página Secret details (Detalhes do segredo), escolha Cancel deletion (Cancelar exclusão).
4. Na caixa de diálogo Cancel secret deletion (Cancelar exclusão do segredo), selecione Cancel deletion (Cancelar exclusão).

AWS CLI

Example Restaurar um segredo excluído anteriormente

O exemplo de `restore-secret` a seguir restaura um segredo que estava previamente programado para exclusão.

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

AWS SDK

Para restaurar um segredo marcado para exclusão, use o comando `RestoreSecret`. Para obter mais informações, consulte the section called “AWS SDKs”.

Marcando segredos em AWS Secrets Manager

Em AWS Secrets Manager, você pode atribuir metadados aos seus segredos usando tags. Uma tag é um par de chave-valor definido por você para um segredo. As tags ajudam você a gerenciar AWS recursos e organizar dados, incluindo informações de faturamento.

Com as tags, é possível:

- Gerenciar, pesquisar e filtrar segredos e outros recursos na sua conta da AWS .
- Controle de acesso a segredos com base em tags vinculadas
- Acompanhe e categorize as despesas associadas a segredos ou projetos específicos

Para obter mais informações sobre o uso de tags para controlar o acesso, consulte the section called “Controle o acesso a segredos usando tags”.

Para saber mais sobre tags de alocação de custos, consulte Como usar tags de alocação de AWS custos no Guia do AWS Billing usuário.

Para obter informações sobre cotas de tags e restrições de nomenclatura, consulte Service Quotas para aplicação de tags no Guia de referência geral da AWS . As tags diferenciam letras maiúsculas de minúsculas.

O Secrets Manager gera uma entrada de CloudTrail registro quando você marca ou desmarca um segredo. Para obter mais informações, consulte the section called “Faça login com AWS CloudTrail ”.

> **Tip**
>
> Use um esquema de marcação consistente em todos os seus AWS recursos. Para ver as práticas recomendadas, consulte o whitepaper Práticas recomendadas de aplicação de tags.

Revisar conceitos básicos de tags

Você pode encontrar segredos por meio de tags no console AWS CLI, SDKs e. AWS também fornece a ferramenta Resource Groups para criar um console personalizado que consolida e organiza seus recursos com base em suas tags. Para encontrar segredos com uma etiqueta específica, consulte the section called “Localizar segredos”.

Você pode usar o console Secrets Manager ou AWS CLI a API Secrets Manager para:

- Criar um segredo com tags
- Adicionar tags a um segredo
- Listar as tags dos seus segredos
- Remover tags de um segredo

É possível usar tags para categorizar os seus segredos. Por exemplo, é possível categorizá-las por finalidade, proprietário ou ambiente. Como você define a chave e o valor para cada marca, é possível criar um conjunto de categorias personalizado para atender às suas necessidades específicas. Aqui estão alguns exemplos de tags:

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

Monitorar custos usando tags

Você pode usar tags para categorizar e monitorar seus AWS custos. Quando você aplica tags aos seus AWS recursos, incluindo segredos, seu relatório de alocação de AWS custos inclui o uso e os custos agregados por tags. É possível aplicar tags que representem categorias de negócios (como centros de custos, nomes de aplicações ou proprietários) para organizar seus custos de vários serviços. Para obter mais informações, consulte Usar etiquetas de alocação de custos para relatórios de faturamento personalizados no Manual do usuário do AWS Billing .

Compreender as restrições de tags

As restrições a seguir se aplicam a tags.

Restrições básicas

- O número máximo de tags por recurso (segredo) é 50.
- As chaves e os valores de marcas diferenciam maiúsculas de minúsculas.
- Não é possível alterar nem editar as tags de um segredo excluído.

Restrições de chaves de marcas

- Cada chave de tag deve ser exclusiva. Se uma tag for adicionada com uma chave que já estiver em uso, a nova tag substituirá o par de chave-valor existente.
- Você não pode iniciar uma chave de tag com `aws:` porque esse prefixo é reservado para uso por AWS. AWS cria tags que começam com esse prefixo em seu nome, mas você não pode editá-las nem excluí-las.
- As chaves de tag devem ter entre 1 e 128 caracteres Unicode.
- As chaves de tag devem conter os seguintes caracteres: letras Unicode, dígitos, espaço em branco e os seguintes caracteres especiais: `_ . / = + - @`.

Restrições de valor das tags

- Os valores das tags devem ter entre 0 e 255 caracteres Unicode.
- Os valores das tags podem estar em branco. Caso contrário, eles devem conter os seguintes caracteres: letras Unicode, dígitos, espaço em branco e qualquer um dos seguintes caracteres especiais: `_ . / = + - @`.

Aplicação de tags a segredos por meio do console do Secrets Manager

É possível gerenciar tags para seus segredos usando o console do Secrets Manager.

Para acessar os recursos de aplicação de tags, faça o seguinte:

1. Abra o console do Secrets Manager.
2. Na barra de navegação, escolha sua região preferida.
3. Na página Segredos, selecione um segredo.

Para visualizar as tags de um segredo

- Na página Detalhes do segredo, selecione a guia Tags.

Para criar um segredo com uma tag

- Siga as etapas em Criar segredos.

Para adicionar ou editar tags em um segredo

1. Na página Detalhes do segredo, escolha a guia Tags e, em seguida, escolha Editar tags.
2. Insira a chave da tag no campo Chave. Opcionalmente, insira um valor para a tag no campo Valor.
3. Escolha Salvar. A tag nova ou atualizada aparecerá na lista de tags.

> **Note**
>
> Se o botão Salvar não estiver habilitado, a chave ou o valor da tag podem não atender às restrições da tag. Para obter mais informações, consulte Compreender as restrições de tags.

Para remover uma tag de um segredo

1. Na página Detalhes do segredo, escolha a guia Tags e, em seguida, escolha o ícone Remover ao lado da tag que você deseja remover.
2. Escolha Salvar para confirmar a remoção ou selecione Desfazer para cancelar.

Marque segredos usando o AWS CLI

AWS CLI exemplos

Example Adicionar um tag a um segredo

O exemplo de `tag-resource` a seguir mostra como anexar um tag com uma sintaxe abreviada.

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

Example Adicionar várias tags a um segredo

O exemplo de `tag-resource` a seguir anexa duas tags de chave/valor a um segredo.

```
aws secretsmanager tag-resource \
            --secret-id MyTestSecret \
```

```
            --tags '[{"Key": "FirstTag", "Value": "FirstValue"}, {"Key": "SecondTag",
 "Value": "SecondValue"}]'
```

Example Remover tags de um segredo

O exemplo de `untag-resource` a seguir remove duas tags de um segredo. Para cada tag, tanto a chave quanto o valor são removidos.

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

Aplicação de tags a segredos por meio da API do Secrets Manager

É possível adicionar, listar e remover tags usando a API do Secrets Manager. Para obter exemplos, consulte a seguinte documentação:

- ListSecrets: Use `ListSecrets` para ver as tags aplicadas a um segredo
- TagResource: adicionar tags a um segredo
- Untag: remove tags de um segredo

Marque segredos usando o AWS SDK do Secrets Manager

Para alterar as tags do seu segredo, use as operações de API a seguir:

- ListSecrets: Use `ListSecrets` para ver as tags aplicadas a um segredo
- `TagResource`: adicionar tags a um segredo
- `UntagResource`: remove tags de um segredo

Para obter mais informações sobre como usar o SDK, consulte the section called “AWS SDKs”.

Replique AWS Secrets Manager segredos em todas as regiões

Você pode replicar seus segredos em vários Regiões da AWS para oferecer suporte a aplicativos espalhados por essas regiões e atender aos requisitos regionais de acesso e baixa latência. Se precisar futuramente, é possível promover um segredo de réplica a um segredo autônomo e depois configurá-lo para replicação de modo independente. O Secrets Manager replica todos os dados de segredos e metadados criptografados, como tags, políticas de recursos em todas as regiões especificadas.

O ARN de um segredo replicado é o mesmo do segredo primário, exceto pela região, por exemplo:

- Segredo primário:
  `arn:aws:secretsmanager:Region1:123456789012:secret:MySecret-a1b2c3`
- Segredo de réplica:
  `arn:aws:secretsmanager:Region2:123456789012:secret:MySecret-a1b2c3`

Para obter informações sobre preços de segredos de réplicas, consulte Definição de preços de AWS Secrets Manager.

Quando você armazena credenciais de banco de dados para um banco de dados de origem replicado para outras regiões, o segredo contém informações de conexão para o banco de dados de origem. Se você replicar o segredo, as réplicas serão cópias do segredo de origem e conterão as mesmas informações de conexão. Você pode adicionar mais key/value pares ao segredo para obter informações sobre a conexão regional.

Se você habilitar a alternância para seu segredo primário, o Secrets Manager alternará o segredo na região primária e o novo valor do segredo se propagará para todos os segredos de réplica associados. Você não precisa gerenciar a alternância individualmente para todos os segredos de réplica.

Você pode replicar segredos em todas as AWS regiões habilitadas. No entanto, se você usar o Secrets Manager em AWS regiões especiais, como AWS GovCloud (US) ou regiões da China, só poderá configurar segredos e réplicas dentro dessas AWS regiões especializadas. Você não pode replicar um segredo em suas AWS regiões habilitadas para uma região especializada ou replicar segredos de uma região especializada para uma região comercial.

Antes que possa replicar um segredo para outra região, é necessário habilitar essa região. Para obter mais informações, consulte Gerenciar regiões da AWS .

É possível usar um segredo em várias regiões sem replicá-lo chamando o endpoint do Secrets Manager na região onde o segredo está armazenado. Para uma lista de endpoints , consulte the section called “Endpoint do Secrets Manager”. Para usar a replicação para melhorar a resiliência de sua carga de trabalho, consulte Arquitetura de recuperação de desastres (DR) em AWS, Parte I: Estratégias para recuperação na nuvem.

O Secrets Manager gera uma entrada de CloudTrail registro quando você replica um segredo. Para obter mais informações, consulte the section called “Faça login com AWS CloudTrail ”.

Para replicar um segredo para outras regiões (console)

1. Abra o console do Secrets Manager em https://console.aws.amazon.com/secretsmanager/.
2. Na lista de segredos, escolha o segredo.
3. Na página de detalhes do segredo, na guia Replicação, siga um destes procedimentos:
   - Se seu segredo não for replicado, selecione Replicate secret (Replicar segredo).
   - Se seu segredo for replicado, na seção Replicate secret (Replicar segredo), selecione Add Region (Adicionar região).
4. Na caixa de diálogo Add replica regions (Adicionar regiões para replicação), faça o seguinte:
   a. Em Região da AWS , escolha a região na qual deseja replicar o segredo.
   b. (Opcional) Para Encryption key (Chave de criptografia), escolha uma chave do KMS para criptografar o segredo. A chave deve ser criada na mesma região da réplica.
   c. (Opcional) Para adicionar outra região, selecione Add more regions (Adicionar mais regiões).
   d. Selecione Replicate (Replicar).

   Você retorna à página de detalhes do segredo. Na seção Replicate Secret (Replicar segredo), o Replication status (Status de replicação) é exibido para cada região.

AWS CLI

Example Replicar um segredo para outra região

O exemplo de `replicate-secret-to-regions` a seguir replica um segredo para eu-west-3. A réplica é criptografada com a chave AWS aws/secretsmanager gerenciada.

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

Example Criar um segredo e replicá-lo

O exemplo a seguir cria um segredo e o replica para eu-west-3. A réplica é criptografada com o. Chave gerenciada pela 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

Para replicar um segredo, use o comando `ReplicateSecretToRegions`. Para obter mais informações, consulte the section called “AWS SDKs”.

Promova uma réplica secreta em uma cópia secreta autônoma em AWS Secrets Manager

Um segredo de réplica é um segredo que é replicado de um primário em outro. Região da AWS Ele tem o mesmo valor e metadados do segredo primário, mas pode ser criptografado com uma chave diferente do KMS. Uma réplica de segredo não pode ser atualizada independentemente de seu segredo primário, exceto pela sua chave de criptografia. Promover uma réplica de segredo desconecta a réplica do segredo primário e transforma a réplica do segredo em um segredo autônomo. Alterações no segredo primário não serão replicadas para o segredo autônomo.

Pode ser útil promover uma réplica de segredo para um segredo autônomo como uma solução de recuperação de desastres se o segredo primário ficar indisponível. Ou talvez promover uma réplica a um segredo autônomo se você quiser ativar a alternância para a réplica.

Se promover uma réplica, certifique-se de atualizar as aplicações correspondentes para usar o segredo autônomo.

O Secrets Manager gera uma entrada de CloudTrail registro quando você promove um segredo. Para obter mais informações, consulte the section called “Faça login com AWS CloudTrail ”.

Para promover um segredo de réplica (console)

1. Faça login no Secrets Manager em https://console.aws.amazon.com/secretsmanager/.
2. Navegue até a região da réplica.
3. Na página Secrets (Segredos), selecione o segredo de réplica.
4. Na página de detalhes da réplica de segredo, escolha Promote to standalone secret (Promover a segredo autônomo).
5. Na caixa de diálogo Promote replica to standalone secret (Promover réplica a segredo autônomo), insira a região e escolha Promote replica (Promover réplica).

AWS CLI

Example Promover um segredo de réplica a um segredo primário

O exemplo de `stop-replication-to-replica` a seguir remove o link entre um segredo de réplica e o primário. O segredo de réplica é promovido a um segredo primário na região da réplica. É necessário chamar `stop-replication-to-replica` diretamente da região da réplica.

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

AWS SDK

Para promover uma réplica a um segredo autônomo, use o comando `StopReplicationToReplica`. É necessário chamar esse comando da região da réplica de segredo. Para obter mais informações, consulte the section called “AWS SDKs”.

Evite a AWS Secrets Manager replicação

Como os segredos podem ser replicados usando `ReplicateSecretToRegions` ou quando são criados usando `CreateSecret`, se você quiser impedir que os usuários repliquem segredos, recomendamos que você evite ações que contenham o parâmetro `AddReplicaRegions`. É possível usar uma instrução `Condition` em suas políticas de permissão para permitir somente ações que não adicionem regiões de réplica. Veja os exemplos de políticas a seguir para ver as instruções de condição que podem ser usadas.

Example Impedir a permissão de replicação

O exemplo de política a seguir mostra como permitir todas as ações que não adicionem regiões de réplica. Isso impede que os usuários repliquem segredos por meio de `ReplicateSecretToRegions` e `CreateSecret`.

JSON

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

Example Permitir permissão de replicação somente para regiões específicas

A política a seguir mostra como permitir tudo o que segue:

- Criar segredos sem replicação
- Criar segredos com replicação para regiões somente nos Estados Unidos e no Canadá

- Replicar segredos para regiões somente nos Estados Unidos e no Canadá

JSON

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

Solucionar problemas de AWS Secrets Manager replicação

AWS Secrets Manager a replicação pode falhar por vários motivos. Para verificar por que um segredo não foi replicado, é possível executar um dos procedimentos a seguir:

- Chame a operação de API `DescribeSecret`.
- Revise AWS CloudTrail os eventos

Quando ocorrer falha na replicação:

- Se não houver versões de segredos utilizáveis, o Secrets Manager removerá o segredo da região da réplica.

- Se houver versões de segredos replicadas com sucesso, elas permanecerão na região da réplica até que você as remova explicitamente usando a operação de API `RemoveRegionsFromReplication`.

As seções a seguir descrevem alguns motivos comuns para as falhas de replicação.

Existe um segredo com o mesmo nome na região selecionada

Para resolver esse problema, é possível substituir o segredo com nome duplicado na região da réplica. Repita a replicação e, na caixa de diálogo Tentar replicação novamente, escolha Substituir.

Não há permissões disponíveis na chave do KMS para concluir a replicação

O Secrets Manager primeiro descriptografa o segredo antes de criptografá-lo novamente com a nova chave do KMS na região da réplica. Se você não tiver permissão do `kms:Decrypt` para a chave de criptografia na região primária, você encontrará esse erro. Para criptografar o segredo replicado com uma chave do KMS diferente de `aws/secretsmanager`, você precisa de `kms:GenerateDataKey` e `kms:Encrypt` para a chave. Consulte the section called “Permissões para a chave do KMS”.

A chave do KMS foi desativada ou não foi encontrada

Se a chave de criptografia na região primária for desativada ou excluída, o Secrets Manager não poderá replicar o segredo. Esse erro pode ocorrer mesmo se você tiver alterado a chave de criptografia, se o segredo tiver versões personalizadas rotuladas que foram criptografadas com a chave de criptografia desativada ou excluída. Para obter informações sobre como o Secrets Manager faz criptografia, consulte the section called “Criptografia e descriptografia de segredos”. Para contornar esse problema, é possível recriar as versões secretas para que o Secrets Manager as criptografe com a chave de criptografia atual. Para obter mais informações, consulte Altere a chave de criptografia para um secredo. Em seguida, tente novamente a replicação.

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

Você não habilitou a região onde a replicação ocorre

Para obter informações sobre como habilitar uma região, consulte Managing AWS Regions. no Guia de referência de gerenciamento de contas da AWS .

Obtenha segredos de AWS Secrets Manager

O Secrets Manager gera uma entrada de CloudTrail registro quando você recupera um segredo. Para obter mais informações, consulte the section called “Faça login com AWS CloudTrail ”.

É possível recuperar valores de segredos usando:

- Obter um valor de segredo do Secrets Manager usando Java
- Obtenha um segredo do Secrets Manager usando Python
- Obtenção de um valor de segredo do Secrets Manager usando .NET
- Obter um valor de segredo do Secrets Manager usando Go
- Obter um valor de segredo do Secrets Manager usando Rust
- Use AWS Secrets Manager segredos no Amazon Elastic Kubernetes Service
- Use AWS Secrets Manager segredos em AWS Lambda funções
- Usando o AWS Secrets Manager agente
- Obtenha um valor secreto do Secrets Manager usando o SDK para C++ AWS
- Obtenha um valor secreto do Secrets Manager usando o JavaScript AWS SDK
- Obtenha um valor secreto do Secrets Manager usando o Kotlin SDK AWS
- Obtenha um valor secreto do Secrets Manager usando o AWS SDK do PHP
- Obtenha um valor secreto do Secrets Manager usando o SDK Ruby AWS
- Obtenha um valor secreto usando o AWS CLI
- Obtenha um valor secreto usando o AWS console
- Use AWS Secrets Manager segredos em AWS Batch
- Obtenha um AWS Secrets Manager segredo em um CloudFormation recurso
- Use AWS Secrets Manager segredos em GitHub empregos
- Use AWS Secrets Manager em GitLab
- Use AWS Secrets Manager segredos em AWS IoT Greengrass
- Use AWS Secrets Manager segredos no Parameter Store

Obter um valor de segredo do Secrets Manager usando Java

Nos aplicativos, você pode recuperar seus segredos ligando `GetSecretValue` ou `BatchGetSecretValue` em qualquer um dos AWS SDKs. No entanto, recomendamos armazenar

em cache seus valores de segredos usando o cache do lado do cliente. Armazenar segredos em cache melhora a velocidade e reduz os seus custos.

Para se conectar a um banco de dados usando as credenciais em um segredo, é possível usar os drivers de conexão de SQL do Secrets Manager, que encapsulam o driver JDBC básico. Isso também usa o cache do lado do cliente, para reduzir o custo de chamar o Secrets Manager. APIs

Tópicos

Obtenha um segredo do Secrets Manager usando Java com armazenamento em cache no lado do cliente

Ao recuperar um segredo, você pode usar o componente de cache baseado em Java do Secrets Manager para armazená-lo em cache para uso futuro. Recuperar um segredo armazenado em cache é mais rápido do que recuperá-lo do Secrets Manager. Como há um custo para ligar para o Secrets Manager APIs, usar um cache pode reduzir seus custos. Para ver todas as formas pelas quais é possível recuperar segredos, consulte Obter segredos.

A política de caches é a do menos usado recentemente (LRU). Assim, quando o cache precisar descartar um segredo, escolherá o segredo menos usado recentemente. Por padrão, o cache atualiza segredos a cada hora. É possível configurar com qual frequência o segredo será atualizado no cache e será possível se conectar à recuperação do segredo para adicionar mais funcionalidades.

O cache não obriga a coleta de resíduos depois que as referências de cache são liberadas. A implementação do cache não inclui a invalidação do cache. A implementação do cache é centrada em torno do próprio cache, portanto, não é reforçada ou tem ênfase em segurança. Se você necessitar de mais segurança, como criptografia de itens no cache, use as interfaces e os métodos abstratos fornecidos.

Para usar o componente, é necessário ter o seguinte:

- Um ambiente de desenvolvimento Java 8 ou superior. Consulte Java SE Downloads (Downloads do Java SE) no site da Oracle.

Para baixar o código-fonte, consulte o componente cliente de cache baseado em Java do Secrets Manager em. GitHub

Para adicionar o componente ao seu projeto, inclua a seguinte dependência no arquivo do Maven pom.xml. Para obter mais informações sobre o Maven, consulte o Getting Started Guide (Guia de primeiros passos) no site do projeto Maven do Apache.

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

Permissões obrigatórias:

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

Para obter mais informações, consulte Referência de permissões.

Referência

- SecretCache
- SecretCacheConfiguration
- SecretCacheHook

Example Recuperar segredos

O exemplo de código a seguir mostra uma função Lambda que recupera uma string do segredo. Ele segue a prática recomendada de instanciar o cache fora do manipulador de funções para que ele não continue chamando a API se você chamar a função Lambda novamente.

```
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

Um cache na memória para segredos solicitados no Secrets Manager. Você usa the section called "getSecretString" ou the section called "getSecretBinary" para recuperar um segredo do cache. É possível definir as configurações de cache executando-as em um objeto the section called "SecretCacheConfiguration" no construtor.

Para obter mais informações, incluindo exemplos, consulte the section called "Java com armazenamento em cache no lado do cliente".

Construtores

`public SecretCache()`

Construtor padrão para um objeto `SecretCache`.

`public SecretCache(AWSSecretsManagerClientBuilder builder)`

Constrói um novo cache usando um cliente do Secrets Manager criado usando o `AWSSecretsManagerClientBuilder` fornecido. Use esse construtor para personalizar o cliente do Secrets Manager, por exemplo, para usar uma região ou endpoint específico.

`public SecretCache(AWSSecretsManager client)`

Constrói um novo cache de segredo usando o `AWSSecretsManagerClient` fornecido. Use esse construtor para personalizar o cliente do Secrets Manager, por exemplo, para usar uma região ou endpoint específico.

`public SecretCache(SecretCacheConfiguration config)`

Constrói um novo cache de segredo usando a `the section called "SecretCacheConfiguration"` fornecida.

Métodos

getSecretString

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

Recupera um segredo de string do Secrets Manager. Retorna um `String`.

getSecretBinary

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

Recupera um segredo de binário do Secrets Manager. Retorna um `ByteBuffer`.

refreshNow

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

Impõe a atualização do cache. Retorna `true` se a atualização for concluída sem erro; caso contrário, `false`.

feche

```
public void close()
```

Fecha o cache.

SecretCacheConfiguration

Opções de configuração de cache para um the section called “SecretCache”, como o tamanho máximo do cache e a vida útil (TTL) para segredos armazenados em cache.

Construtor

```
public SecretCacheConfiguration
```

Construtor padrão para um objeto `SecretCacheConfiguration`.

Métodos

getClient

```
public AWSSecretsManager getClient()
```

Ele retorna o `AWSSecretsManagerClient` de onde o cache recupera segredos.

setClient

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

Configura o cliente do `AWSSecretsManagerClient` de onde o cache recupera segredos.

getCacheHook

```
public SecretCacheHook getCacheHook()
```

Retorna a interface do the section called “SecretCacheHook” usada para conectar atualizações de cache.

setCacheHook

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

Define a interface do the section called “SecretCacheHook” usada para conectar atualizações de cache.

getMaxCacheTamanho

```
public int getMaxCacheSize()
```

Retorna o tamanho máximo do cache. O padrão é de 1.024 segredos.

setMaxCacheTamanho

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

Define o tamanho máximo do cache. O padrão é de 1.024 segredos.

getCacheItemTTL

```
public long getCacheItemTTL()
```

Retorna o TTL em milissegundos para os itens armazenados em cache. Quando um segredo armazenado em cache excede esse TTL, o cache recupera uma nova cópia do segredo do `AWSSecretsManagerClient`. O padrão é 1 hora em milissegundos.

O cache atualizará o segredo de forma síncrona quando ele for solicitado após o TTL. Se a atualização síncrona falhar, o cache retornará o segredo obsoleto.

setCacheItemTTL

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

Define o TTL em milissegundos para os itens armazenados em cache. Quando um segredo armazenado em cache excede esse TTL, o cache recupera uma nova cópia do segredo do `AWSSecretsManagerClient`. O padrão é 1 hora em milissegundos.

getVersionStage

```
public String getVersionStage()
```

Retorna a versão dos segredos que você deseja armazenar em cache. Para obter mais informações, consulte Secret versions (Versões de segredos). O padrão é `"AWSCURRENT"`.

setVersionStage

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

Define a versão dos segredos que você deseja armazenar em cache. Para obter mais informações, consulte Secret versions (Versões de segredos). O padrão é `"AWSCURRENT"`.

SecretCacheConfiguration Com o cliente

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

Define o `AWSSecretsManagerClient` de onde os segredos serão recuperados. Retorna o objeto da `SecretCacheConfiguration` atualizado com a nova configuração.

SecretCacheConfiguration withCacheHook

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

Define a interface usada para conectar o cache na memória. Retorna o objeto da `SecretCacheConfiguration` atualizado com a nova configuração.

SecretCacheConfiguration withMaxCacheTamanho

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

Define o tamanho máximo do cache. Retorna o objeto da `SecretCacheConfiguration` atualizado com a nova configuração.

SecretCacheConfiguration withCacheItemTTL

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

Define o TTL em milissegundos para os itens armazenados em cache. Quando um segredo armazenado em cache excede esse TTL, o cache recupera uma nova cópia do segredo do `AWSSecretsManagerClient`. O padrão é 1 hora em milissegundos. Retorna o objeto da `SecretCacheConfiguration` atualizado com a nova configuração.

SecretCacheConfiguration withVersionStage

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

Define a versão dos segredos que você deseja armazenar em cache. Para obter mais informações, consulte Secret versions (Versões de segredos). Retorna o objeto da `SecretCacheConfiguration` atualizado com a nova configuração.

SecretCacheHook

Uma interface para conectar a um the section called “SecretCache” para executar ações nos segredos sendo armazenados no cache.

put

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

Prepare o objeto para o armazenamento no cache.

Retorna o objeto a ser armazenado no cache.

obter

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

Derive o objeto do objeto sendo armazenado em cache.

Retorna o objeto a ser retornado do cache

Conecte-se a um banco de dados SQL usando JDBC com credenciais em um segredo AWS Secrets Manager

Em aplicativos Java, você pode usar os drivers de conexão SQL do Secrets Manager para se conectar aos bancos de dados MySQL, PostgreSQL, Oracle, MSSQLServer Db2 e Redshift usando credenciais armazenadas no Secrets Manager. Cada driver empacota o driver JDBC base para que você possa usar chamadas JDBC para acessar o seu banco de dados. No entanto, em

vez de passar um nome de usuário e senha para a conexão, você fornece o ID de um segredo. O driver chama o Secrets Manager para recuperar o valor do segredo e usa as credenciais no segredo para se conectar ao banco de dados. O driver também armazena em cache as credenciais usando a biblioteca em cache no lado do cliente de Java. Assim, as conexões futuras não exigem uma chamada para o Secrets Manager. Por padrão, o cache é atualizado a cada hora e também quando o segredo é alternado. Para configurar o cache, consulte the section called "SecretCacheConfiguration".

Você pode baixar o código-fonte em GitHub.

Para usar os drivers de conexão SQL do Secrets Manager:

- A sua aplicação deve estar em Java 8 ou superior.
- O seu segredo deve ser um dos seguintes:
  - Um segredo de banco de dados na estrutura JSON esperada. Para verificar o formato, no console do Secrets Manager, veja o seu segredo e escolha Retrieve secret value (Recuperar o valor do segredo). Como alternativa, no AWS CLI, ligue get-secret-value.
  - Um segredo gerenciado pelo Amazon RDS. Para esse tipo de segredo, é necessário especificar um endpoint e uma porta ao estabelecer a conexão.
  - Um segredo gerenciado pelo Amazon Redshift. Para esse tipo de segredo, é necessário especificar um endpoint e uma porta ao estabelecer a conexão.

Se o banco de dados for replicado para outras regiões, para se conectar a uma réplica de banco de dados em outra região, especifique o endpoint regional e a porta ao criar a conexão. Você pode armazenar informações de conexão regional no segredo como key/value pares extras, nos parâmetros do SSM Parameter Store ou na sua configuração de código.

Para adicionar o driver ao seu projeto, no arquivo de compilação do Maven `pom.xml`, adicione a seguinte dependência para o driver. Para obter mais informações, consulte Secrets Manager SQL Connection Library (Biblioteca de conexões SQL do Secrets Manager) no site Maven Central Repository.

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

O driver usa a cadeia de fornecedores de credenciais padrão. Se você executar o driver no Amazon EKS, ele poderá retirar as credenciais do nó em que está sendo executado, em vez do perfil da conta de serviço. Para resolver isso, adicione a versão 1 de `com.amazonaws:aws-java-sdk-sts` ao seu arquivo de projeto Gradle ou Maven como uma dependência.

Para definir uma URL de endpoint de AWS PrivateLink DNS e uma região no `secretsmanager.properties` arquivo:

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

Para substituir a região primária, defina a variável de ambiente `AWS_SECRET_JDBC_REGION` ou faça a seguinte alteração no arquivo `secretsmanager.properties`:

```
drivers.region = region
```

Permissões obrigatórias:

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

Para obter mais informações, consulte Referência de permissões.

Exemplos:

- Estabeleça uma conexão com um banco de dados
- Estabelecer uma conexão especificando o endpoint e a porta
- Use o grupo de conexões c3p0 para estabelecer uma conexão
- Usar o agrupamento de conexões c3p0 para estabelecer uma conexão especificando o endpoint e a porta

Estabeleça uma conexão com um banco de dados

O exemplo a seguir mostra como estabelecer uma conexão com um banco de dados usando as credenciais e informações sobre a conexão em um segredo. Assim que você tiver uma conexão, será possível usar chamadas JDBC para acessar o banco de dados. Para obter mais informações, consulte JDBC Basics (Conceitos básicos de JDBC) no site da documentação de 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);
```

Estabelecer uma conexão especificando o endpoint e a porta

O exemplo a seguir mostra como estabelecer uma conexão com um banco de dados usando as credenciais em um segredo com um endpoint e uma porta que você especifica.

Os segredos gerenciados pelo Amazon RDS não incluem o endpoint e a porta do banco de dados. Para se conectar a um banco de dados usando credenciais principais em um segredo gerenciado pelo Amazon RDS, você as especifica em seu código.

Segredos que são replicados para outras regiões podem melhorar a latência da conexão com o banco de dados regional, mas não contêm informações diferentes de conexão em relação ao segredo da origem. Cada réplica é uma cópia do segredo de origem. Para armazenar informações de conexão regional no segredo, adicione mais key/value pares para o endpoint e informações de porta para as regiões.

Assim que você tiver uma conexão, será possível usar chamadas JDBC para acessar o banco de dados. Para obter mais informações, consulte JDBC Basics (Conceitos básicos de JDBC) no site da documentação de 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);
```

Use o grupo de conexões c3p0 para estabelecer uma conexão

O exemplo a seguir mostra como estabelecer um pool de conexões com um arquivo `c3p0.properties` que usa o driver para recuperar credenciais e informações de conexão do segredo. Em `user` e `jdbcUrl`, insira o ID do segredo para configurar o grupo de conexões. Em seguida, você pode recuperar conexões do grupo e usá-las como qualquer outra conexão de banco de dados. Para obter mais informações, consulte JDBC Basics (Conceitos básicos de JDBC) no site da documentação de Java.

Para obter mais informações sobre c3p0, consulte c3p0 no site 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
```

Usar o agrupamento de conexões c3p0 para estabelecer uma conexão especificando o endpoint e a porta

O exemplo a seguir mostra como estabelecer um pool de conexões com um arquivo `c3p0.properties` que use o driver para recuperar credenciais em um segredo com um endpoint e uma porta que você especifica. Em seguida, você pode recuperar conexões do grupo e usá-las como qualquer outra conexão de banco de dados. Para obter mais informações, consulte JDBC Basics (Conceitos básicos de JDBC) no site da documentação de Java.

Os segredos gerenciados pelo Amazon RDS não incluem o endpoint e a porta do banco de dados. Para se conectar a um banco de dados usando credenciais principais em um segredo gerenciado pelo Amazon RDS, você as especifica em seu código.

Segredos que são replicados para outras regiões podem melhorar a latência da conexão com o banco de dados regional, mas não contêm informações diferentes de conexão em relação ao segredo da origem. Cada réplica é uma cópia do segredo de origem. Para armazenar informações de conexão regional no segredo, adicione mais key/value pares para o endpoint e informações de porta para as regiões.

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
```

Obtenha um valor secreto do Secrets Manager usando o Java AWS SDK

Nos aplicativos, você pode recuperar seus segredos ligando `GetSecretValue` ou `BatchGetSecretValue` em qualquer um dos AWS SDKs. No entanto, recomendamos armazenar em cache seus valores de segredos usando o cache do lado do cliente. Armazenar segredos em cache melhora a velocidade e reduz os seus custos.

- Se você armazenar credenciais de banco de dados no segredo, use os drivers de conexão SQL do Secrets Manager para se conectar a um banco de dados usando as credenciais no segredo.

- Para outros tipos de segredos, use o componente de armazenamento em cache baseado em Java do Secrets Manager ou chame o SDK diretamente com `GetSecretValue` ou `BatchGetSecretValue`.

Os exemplos de código a seguir mostram como usar o `GetSecretValue`.

Permissões obrigatórias: `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);
        }
    }
}
```

Obtenha um segredo do Secrets Manager usando Python

Nos aplicativos, você pode recuperar seus segredos ligando `GetSecretValue` ou `BatchGetSecretValue` em qualquer um dos AWS SDKs. No entanto, recomendamos armazenar em cache seus valores de segredos usando o cache do lado do cliente. Armazenar segredos em cache melhora a velocidade e reduz os seus custos.

Tópicos

- Obtenha um segredo do Secrets Manager usando Python com armazenamento em cache no lado do cliente
- Obtenha um valor secreto do Secrets Manager usando o Python SDK AWS
- Obtenha um lote de valores secretos do Secrets Manager usando o Python SDK AWS

Obtenha um segredo do Secrets Manager usando Python com armazenamento em cache no lado do cliente

Ao recuperar um segredo, é possível usar o componente de cache baseado em Python do Secrets Manager para armazená-lo em cache para uso futuro. Recuperar um segredo armazenado em cache é mais rápido do que recuperá-lo do Secrets Manager. Como há um custo para ligar para o Secrets Manager APIs, usar um cache pode reduzir seus custos. Para ver todas as formas pelas quais é possível recuperar segredos, consulte Obter segredos.

A política de caches é a do menos usado recentemente (LRU). Assim, quando o cache precisar descartar um segredo, escolherá o segredo menos usado recentemente. Por padrão, o cache atualiza segredos a cada hora. É possível configurar com qual frequência o segredo será atualizado no cache e será possível se conectar à recuperação do segredo para adicionar mais funcionalidades.

O cache não obriga a coleta de resíduos depois que as referências de cache são liberadas. A implementação do cache não inclui a invalidação do cache. A implementação do cache é centrada em torno do próprio cache, portanto, não é reforçada ou tem ênfase em segurança. Se você necessitar de mais segurança, como criptografia de itens no cache, use as interfaces e os métodos abstratos fornecidos.

Para usar o componente, é necessário ter o seguinte:

- Python 3.6 ou superior.
- botocore 1.12 ou superior. Consulte AWS SDK para Python e Botocore.
- setuptools_scm 3.2 ou superior. Veja https://pypi. org/project/setuptools-fraude/.

Para baixar o código-fonte, consulte o componente cliente de cache baseado em Python do Secrets Manager em. GitHub

Para instalar o componente, use o seguinte comando.

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

Permissões obrigatórias:

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

Para obter mais informações, consulte Referência de permissões.

Referência

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

Example Recuperar segredos

O exemplo a seguir mostra como obter o valor secreto de um segredo chamado*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

Um cache na memória para segredos recuperados no Secrets Manager. Você usa the section called “get_secret_string” ou the section called “get_secret_binary” para recuperar um segredo do cache. É possível definir as configurações de cache executando-as em um objeto the section called “SecretCacheConfig” no construtor.

Para obter mais informações, incluindo exemplos, consulte the section called “Python com armazenamento em cache no lado do cliente”.

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

Estes são os métodos disponíveis:

- get_secret_string
- get_secret_binary

get_secret_string

Recupera o valor da string do segredo.

Sintaxe da solicitação

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

Parâmetros

- `secret_id` (string): [obrigatório] o nome ou o ARN do segredo.
- `version_stage` (string): a versão dos segredos que você deseja recuperar. Para obter mais informações, consulte versões de segredos. O padrão é 'AWSCURRENT'.

Tipo de retorno

string

get_secret_binary

Recupera o valor do binário do segredo.

Sintaxe da solicitação

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

Parâmetros

- `secret_id` (string): [obrigatório] o nome ou o ARN do segredo.
- `version_stage` (string): a versão dos segredos que você deseja recuperar. Para obter mais informações, consulte versões de segredos. O padrão é 'AWSCURRENT'.

Tipo de retorno

String codificada em base64

SecretCacheConfig

Opções de configuração de cache para um the section called “SecretCache”, como o tamanho máximo do cache e a vida útil (TTL) para segredos armazenados em cache.

Parâmetros

`max_cache_size` (int)

O tamanho máximo do cache. O padrão é de `1024` segredos.

`exception_retry_delay_base` (int)

O número de segundos a aguardar ao encontrar uma exceção e antes de repetir a solicitação. O padrão é `1`.

`exception_retry_growth_factor` (int)pur

O fator de crescimento a ser usado para calcular o tempo de espera entre novas tentativas de solicitações com falha. O padrão é `2`.

`exception_retry_delay_max` (int)

O tempo máximo em segundos a aguardar entre solicitações com falha. O padrão é `3600`.

`default_version_stage` (str)

A versão de segredos que você deseja armazenar em cache. Para obter mais informações, consulte Secret versions (Versões de segredos). O padrão é `'AWSCURRENT'`.

`secret_refresh_interval` (int)

O número de segundos a aguardar entre a atualização de informações de segredos armazenados em cache. O padrão é `3600`.

`secret_cache_hook` (SecretCacheHook)

Uma implementação da classe abstrata de `SecretCacheHook`. O valor padrão é `None`.

SecretCacheHook

Uma interface para conectar a um the section called “SecretCache” para executar ações nos segredos sendo armazenados no cache.

Estes são os métodos disponíveis:

- put
- obter

put

Prepara o objeto para armazenamento no cache.

Sintaxe da solicitação

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

Parâmetros

- `obj` (objeto): [obrigatório] o segredo ou objeto que contém o segredo.

Tipo de retorno

objeto

obter

Deriva o objeto do objeto armazenado em cache.

Sintaxe da solicitação

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

Parâmetros

- `obj` (objeto): [obrigatório] o segredo ou objeto que contém o segredo.

Tipo de retorno

objeto

@InjectSecretString

Este decorador espera uma string de ID de segredo e the section called “SecretCache” como o primeiro e o segundo argumentos. O decorador retorna o valor da string do segredo. O segredo deve conter uma string.

```
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

Este decorador espera uma string de ID de segredo e the section called “SecretCache” como o primeiro e o segundo argumentos. Os argumentos restantes mapeiam parâmetros da função empacotada para chaves JSON no segredo. O segredo deve conter uma string na estrutura JSON.

Para um segredo que contém esse JSON:

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

O exemplo a seguir mostra como extrair os valores JSON para `username` e `password` do segredo.

```
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')
```

Obtenha um valor secreto do Secrets Manager usando o Python SDK AWS

Nos aplicativos, você pode recuperar seus segredos ligando `GetSecretValue` ou `BatchGetSecretValue` em qualquer um dos AWS SDKs. No entanto, recomendamos armazenar em cache seus valores de segredos usando o cache do lado do cliente. Armazenar segredos em cache melhora a velocidade e reduz os seus custos.

Para aplicações Python, use o componente de cache baseado em Python do Secrets Manager ou chame o SDK diretamente com `get_secret_value` ou `batch_get_secret_value`.

Os exemplos de código a seguir mostram como usar o `GetSecretValue`.

Permissões obrigatórias: `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
```

Obtenha um lote de valores secretos do Secrets Manager usando o Python SDK AWS

O exemplo de código a seguir mostra como obter um lote de valores de segredo do Secrets Manager.

Permissões obrigatórias:

- `secretsmanager:BatchGetSecretValue`
- Permissão `secretsmanager:GetSecretValue` para cada segredo que deseja recuperar.
- Se você usa filtros, também deve ter `secretsmanager:ListSecrets`.

Para um exemplo de política de permissões, consulte the section called “Exemplo: permissão para recuperar um grupo de valores de segredos em um lote”.

> **Important**
>
> Se você tiver uma política de VPCE que nega permissão para recuperar um segredo individual no grupo que você está recuperando, `BatchGetSecretValue` não retornará nenhum valor de segredo e retornará um erro.

```
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
```

Obtenção de um valor de segredo do Secrets Manager usando .NET

Nos aplicativos, você pode recuperar seus segredos ligando `GetSecretValue` ou `BatchGetSecretValue` em qualquer um dos AWS SDKs. No entanto, recomendamos armazenar em cache seus valores de segredos usando o cache do lado do cliente. Armazenar segredos em cache melhora a velocidade e reduz os seus custos.

Tópicos

- Obtenha um segredo do Secrets Manager usando .NET com armazenamento em cache no lado do cliente
- Obtenha um valor secreto do Secrets Manager usando o SDK para .NET

Obtenha um segredo do Secrets Manager usando .NET com armazenamento em cache no lado do cliente

Ao recuperar um segredo, você pode usar o componente de cache baseado em .NET do Secrets Manager para armazená-lo em cache para uso futuro. Recuperar um segredo armazenado em cache é mais rápido do que recuperá-lo do Secrets Manager. Como há um custo para ligar para o Secrets Manager APIs, usar um cache pode reduzir seus custos. Para ver todas as formas pelas quais é possível recuperar segredos, consulte Obter segredos.

A política de caches é a do menos usado recentemente (LRU). Assim, quando o cache precisar descartar um segredo, escolherá o segredo menos usado recentemente. Por padrão, o cache atualiza segredos a cada hora. É possível configurar com qual frequência o segredo será atualizado no cache e será possível se conectar à recuperação do segredo para adicionar mais funcionalidades.

O cache não obriga a coleta de resíduos depois que as referências de cache são liberadas. A implementação do cache não inclui a invalidação do cache. A implementação do cache é centrada em torno do próprio cache, portanto, não é reforçada ou tem ênfase em segurança. Se você necessitar de mais segurança, como criptografia de itens no cache, use as interfaces e os métodos abstratos fornecidos.

Para usar o componente, é necessário ter o seguinte:

- .NET Framework 4.6.2 ou superior, ou .NET Standard 2.0 ou superior. Consulte Download .NET (Baixe .NET) no site da Microsoft .NET.
- O AWS SDK para.NET. Consulte the section called “AWS SDKs”.

Para baixar o código-fonte, consulte Cliente de cache para.NET on GitHub.

Para usar o cache, primeiro instancie-o e, em seguida, recupere o seu segredo usando `GetSecretString` ou `GetSecretBinary`. Em recuperações sucessivas, o cache retorna a cópia armazenada em cache do segredo.

Para obter o pacote de cache

- Execute um destes procedimentos:
  - Execute o seguinte comando CLI do .NET em seu diretório de projeto.

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

- Adicione a seguinte referência de pacote ao seu arquivo `.csproj`.

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

Permissões obrigatórias:

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

Para obter mais informações, consulte Referência de permissões.

Referência

- SecretsManagerCache
- SecretCacheConfiguration
- ISecretCacheHook

Example Recuperar segredos

O exemplo de código a seguir mostra um método que recupera um segredo chamado*MySecret*.

```
using Amazon.SecretsManager.Extensions.Caching;

namespace LambdaExample
{
    public class CachingExample
    {
        private const string MySecretName ="MySecret";

        private SecretsManagerCache cache = new SecretsManagerCache();

        public async Task<Response>  FunctionHandlerAsync(string input, ILambdaContext
 context)
        {
            string MySecret = await cache.GetSecretString(MySecretName);

            // Use the secret, return success
```

```
        }
    }
}
```

Example Configuração da duração da atualização do cache de vida útil (TTL)

O exemplo de código a seguir mostra um método que recupera um segredo chamado *MySecret* e define a duração da atualização do cache TTL como 24 horas.

```
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

Um cache na memória para segredos solicitados no Secrets Manager. Você usa the section called "GetSecretString" ou the section called "GetSecretBinary" para recuperar um segredo do cache. É possível definir as configurações de cache executando-as em um objeto the section called "SecretCacheConfiguration" no construtor.

Para obter mais informações, incluindo exemplos, consulte the section called ".NET com armazenamento em cache no lado do cliente".

Construtores

`public SecretsManagerCache()`

Construtor padrão para um objeto `SecretsManagerCache`.

`public SecretsManagerCache(IAmazonSecretsManager secretsManager)`

Constrói um novo cache usando um cliente do Secrets Manager criado usando o AmazonSecretsManagerClient fornecido. Use esse construtor para personalizar o cliente do Secrets Manager, por exemplo, para usar uma região ou endpoint específico.

Parâmetros

secretsManager

O AmazonSecretsManagerClientpara recuperar segredos de.

`public SecretsManagerCache(SecretCacheConfiguration config)`

Constrói um novo cache de segredo usando a the section called "SecretCacheConfiguration" fornecida. Use esse construtor para configurar o cache, por exemplo, o número de segredos a serem armazenados em cache e com que frequência ele é atualizado.

Parâmetros

config

Uma the section called "SecretCacheConfiguration" que contém informações de configuração para o cache.

`public SecretsManagerCache(IAmazonSecretsManager secretsManager, SecretCacheConfiguration config)`

Constrói um novo cache usando um cliente Secrets Manager criado usando o fornecido AmazonSecretsManagerCliente umthe section called "SecretCacheConfiguration". Use este construtor para personalizar o cliente do Secrets Manager, por exemplo, para usar uma região ou endpoint específico, assim como configurar o cache, por exemplo, o número de segredos a serem armazenados em cache e com que frequência ele será atualizado.

Parâmetros

secretsManager

O AmazonSecretsManagerClientpara recuperar segredos de.

config

Uma the section called “SecretCacheConfiguration” que contém informações de configuração para o cache.

Métodos

GetSecretString

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

Recupera um segredo de string do Secrets Manager.

Parâmetros

secretId

O ARN ou o nome do segredo a ser recuperado.

GetSecretBinary

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

Recupera um segredo de binário do Secrets Manager.

Parâmetros

secretId

O ARN ou o nome do segredo a ser recuperado.

RefreshNowAsync

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

Solicita o valor do segredo do Secrets Manager e atualiza o cache com quaisquer alterações. Se não houver entrada de cache existente, ele criará uma nova. Se a atualização for bem-sucedida, ele retornará `true`.

Parâmetros

secretId

O ARN ou o nome do segredo a ser recuperado.

GetCachedSecret

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

Retorna a entrada de cache para o segredo especificado, se ele existir no cache. Caso contrário, ele recupera o segredo do Secrets Manager e cria uma nova entrada de cache.

Parâmetros

secretId

O ARN ou o nome do segredo a ser recuperado.

SecretCacheConfiguration

Opções de configuração de cache para um the section called “SecretsManagerCache”, como o tamanho máximo do cache e a vida útil (TTL) para segredos armazenados em cache.

Propriedades

CacheItemTTL

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

O TTL de um item de cache em milissegundos. O padrão é de `3600000` minutos ou 1 hora. O máximo é de `4294967295` milissegundos, que é aproximadamente 49,7 dias.

MaxCacheSize

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

O tamanho máximo do cache. O padrão é de 1.024 segredos. O máximo é 65.535.

VersionStage

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

A versão de segredos que você deseja armazenar em cache. Para obter mais informações, consulte Secret versions (Versões de segredos). O padrão é `"AWSCURRENT"`.

Cliente

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

O AmazonSecretsManagerClientpara recuperar segredos de. Se for `null`, o cache instancia um novo cliente. O padrão é `null`.

CacheHook

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

O the section called “ISecretCacheHook”.

ISecretCacheHook

Uma interface para conectar a um the section called “SecretsManagerCache” para executar ações nos segredos sendo armazenados no cache.

Métodos

Put

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

Prepare o objeto para o armazenamento no cache.

Retorna o objeto a ser armazenado no cache.

Obtenção

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

Derive o objeto do objeto sendo armazenado em cache.

Retorna o objeto a ser retornado do cache

Obtenha um valor secreto do Secrets Manager usando o SDK para .NET

Nos aplicativos, você pode recuperar seus segredos ligando `GetSecretValue` ou `BatchGetSecretValue` em qualquer um dos AWS SDKs. No entanto, recomendamos armazenar em cache seus valores de segredos usando o cache do lado do cliente. Armazenar segredos em cache melhora a velocidade e reduz os seus custos.

Para aplicações .NET, use o componente de cache baseado em .NET do Secrets Manager ou chame o SDK diretamente com `GetSecretValue` ou `BatchGetSecretValue`.

Os exemplos de código a seguir mostram como usar o `GetSecretValue`.

Permissões obrigatórias: `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;
            }
        }
    }
```

Obter um valor de segredo do Secrets Manager usando Go

Nos aplicativos, você pode recuperar seus segredos ligando `GetSecretValue` ou `BatchGetSecretValue` em qualquer um dos AWS SDKs. No entanto, recomendamos armazenar em cache seus valores de segredos usando o cache do lado do cliente. Armazenar segredos em cache melhora a velocidade e reduz os seus custos.

Tópicos

- Obter um segredo do Secrets Manager usando Go com armazenamento em cache no lado do cliente
- Obtenha um valor secreto do Secrets Manager usando o Go AWS SDK

Obter um segredo do Secrets Manager usando Go com armazenamento em cache no lado do cliente

Ao recuperar um segredo, você pode usar o componente de cache baseado em Go do Secrets Manager para armazená-lo em cache para uso futuro. Recuperar um segredo armazenado em cache é mais rápido do que recuperá-lo do Secrets Manager. Como há um custo para ligar para o Secrets Manager APIs, usar um cache pode reduzir seus custos. Para ver todas as formas pelas quais é possível recuperar segredos, consulte Obter segredos.

A política de caches é a do menos usado recentemente (LRU). Assim, quando o cache precisar descartar um segredo, escolherá o segredo menos usado recentemente. Por padrão, o cache atualiza segredos a cada hora. É possível configurar com qual frequência o segredo será atualizado no cache e será possível se conectar à recuperação do segredo para adicionar mais funcionalidades.

O cache não obriga a coleta de resíduos depois que as referências de cache são liberadas. A implementação do cache não inclui a invalidação do cache. A implementação do cache é centrada em torno do próprio cache, portanto, não é reforçada ou tem ênfase em segurança. Se você necessitar de mais segurança, como criptografia de itens no cache, use as interfaces e os métodos abstratos fornecidos.

Para usar o componente, é necessário ter o seguinte:

- AWS SDK para Go. Consulte the section called “AWS SDKs”.

Para baixar o código-fonte, consulte Cliente de cache do Secrets Manager Go ativado GitHub.

Para configurar um ambiente de desenvolvimento Go, consulte Golang Getting Started (Conceitos básicos de Golang) no site da Go Programming Language.

Permissões obrigatórias:

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

Para obter mais informações, consulte Referência de permissões.

Referência

- tipo Cache

- digitar CacheConfig
- digitar CacheHook

Example Recuperar segredos

O exemplo de código a seguir mostra uma função Lambda que recupera um segredo.

```
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)
}
```

tipo Cache

Um cache na memória para segredos solicitados no Secrets Manager. Você usa the section called "GetSecretString" ou the section called "GetSecretBinary" para recuperar um segredo do cache.

O exemplo a seguir mostra como definir as configurações de 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 },
)
```

Para obter mais informações, incluindo exemplos, consulte the section called “Go com armazenamento em cache no lado do cliente”.

Métodos

Novo

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

O New constrói um cache do segredo usando opções funcionais. Caso contrário, usa opções padrão. Inicializa um SecretsManager cliente a partir de uma nova sessão. Inicializa CacheConfig com os valores padrão. Inicializa o cache LRU com um tamanho máximo padrão.

GetSecretString

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

GetSecretString obtém o valor da string secreta do cache para determinado ID secreto. Retorna a string do segredo e um erro caso a operação falhe.

GetSecretStringWithStage

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

GetSecretStringWithStage obtém o valor da string secreta do cache para determinado ID secreto e estágio de versão. Retorna a string do segredo e um erro caso a operação falhe.

GetSecretBinary

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

GetSecretBinary obtém o valor binário secreto do cache para determinado ID secreto. Caso a operação falhe, ele retorna o binário do segredo e um erro.

GetSecretBinaryWithStage

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

GetSecretBinaryWithStage obtém o valor binário secreto do cache para determinado ID secreto e estágio de versão. Caso a operação falhe, ele retorna o binário do segredo e um erro.

digitar CacheConfig

Opções de configuração de cache para o Cache, como o tamanho máximo do cache, a etapa de versão padrão e a vida útil (TTL) para segredos armazenados em 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
}
```

digitar CacheHook

Uma interface para conectar a um Cache para executar ações no segredo que está sendo armazenado no cache.

Métodos

Put

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

Prepara o objeto para armazenamento no cache.

Obtenção

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

Deriva o objeto do objeto armazenado em cache.

Obtenha um valor secreto do Secrets Manager usando o Go AWS SDK

Nos aplicativos, você pode recuperar seus segredos ligando `GetSecretValue` ou `BatchGetSecretValue` em qualquer um dos AWS SDKs. No entanto, recomendamos armazenar em cache seus valores de segredos usando o cache do lado do cliente. Armazenar segredos em cache melhora a velocidade e reduz os seus custos.

Para aplicações Go, use o componente de cache baseado em Go do Secrets Manager ou chame o SDK diretamente com `GetSecretValue` ou `BatchGetSecretValue`.

O exemplo de código a seguir mostra como obter um valor de segredo do Secrets Manager.

Permissões obrigatórias: `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.
  }
```

Obter um valor de segredo do Secrets Manager usando Rust

Nos aplicativos, você pode recuperar seus segredos ligando `GetSecretValue` ou `BatchGetSecretValue` em qualquer um dos AWS SDKs. No entanto, recomendamos armazenar em cache seus valores de segredos usando o cache do lado do cliente. Armazenar segredos em cache melhora a velocidade e reduz os seus custos.

Tópicos

- Obtenha um segredo do Secrets Manager usando Rust com armazenamento em cache no lado do cliente
- Obtenha um valor secreto do Secrets Manager usando o Rust SDK AWS

Obtenha um segredo do Secrets Manager usando Rust com armazenamento em cache no lado do cliente

Ao recuperar um segredo, é possível usar o componente de cache baseado em Rust do Secrets Manager para armazená-lo em cache para uso futuro. Recuperar um segredo armazenado em cache é mais rápido do que recuperá-lo do Secrets Manager. Como há um custo para ligar para o Secrets Manager APIs, usar um cache pode reduzir seus custos. Para ver todas as formas pelas quais é possível recuperar segredos, consulte Obter segredos.

A política de cache é a do primeiro a entrar, primeiro a sair (FIFO). Assim, quando o cache precisar descartar um segredo, usará o segredo mais antigo. Por padrão, o cache atualiza segredos a cada hora. É possível configurar as seguintes opções:

- `max_size`: o número máximo de segredos em cache a serem mantidos antes da remoção dos segredos que não foram acessados recentemente.
- `ttl`: a duração pela qual um item armazenado em cache é considerado válido antes de exigir uma atualização do estado do segredo.

A implementação do cache não inclui a invalidação do cache. A implementação do cache é centrada em torno do próprio cache, portanto, não é reforçada ou tem ênfase em segurança. Se você precisar de segurança adicional, como criptografar itens no cache, use as características fornecidas para modificar o cache.

Para usar o componente, é necessário ter um ambiente de desenvolvimento do Rust 2021 com `tokio`. Para obter mais informações, consulte Conceitos básicos no site Rust Programming Language.

Para baixar o código-fonte, consulte o componente cliente de cache baseado em Rust do Secrets Manager em. GitHub

Para instalar o componente de armazenamento em cache, use o seguinte comando.

```
cargo add aws_secretsmanager_caching
```

Permissões obrigatórias:

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

Para obter mais informações, consulte Referência de permissões.

Example Recuperar segredos

O exemplo a seguir mostra como obter o valor secreto de um segredo chamado*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 Instanciação do cache com uma configuração personalizada e um cliente personalizado

O exemplo a seguir mostra como configurar o cache e, em seguida, obter o valor secreto de um segredo chamado*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
```

Obtenha um valor secreto do Secrets Manager usando o Rust SDK AWS

Nos aplicativos, você pode recuperar seus segredos ligando `GetSecretValue` ou `BatchGetSecretValue` em qualquer um dos AWS SDKs. No entanto, recomendamos armazenar em cache seus valores de segredos usando o cache do lado do cliente. Armazenar segredos em cache melhora a velocidade e reduz os seus custos.

Para aplicativos Rust, use o componente de cache baseado em Rust do Secrets Manager ou chame o SDK diretamente com ou. GetSecretValue BatchGetSecretValue

O exemplo de código a seguir mostra como obter um valor de segredo do Secrets Manager.

Permissões obrigatórias: `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(())
}
```

Use AWS Secrets Manager segredos no Amazon Elastic Kubernetes Service

Para mostrar segredos do AWS Secrets Manager (ASCP) como arquivos montados nos pods do Amazon EKS, você pode usar os AWS segredos e o provedor de configuração do driver CSI do Kubernetes Secrets Store. O ASCP funciona com o Amazon Elastic Kubernetes Service 1.17+ executando um grupo de nós do Amazon EC2. AWS Fargate grupos de nós não são suportados. Com o ASCP, é possível armazenar e gerenciar segredos no Secrets Manager e, em seguida, recuperá-los por meio de workloads executadas no Amazon EKS. Se seu segredo contiver vários pares de chave-valor no formato JSON, será possível escolher quais montará no Amazon EKS. O ASCP usa a sintaxe JMESPath para consultar os pares de chave/valor no seu segredo. O ASCP também funciona com parâmetros do Parameter Store. O ASCP oferece dois métodos de autenticação com o Amazon EKS. A primeira abordagem usa perfis do IAM para contas de serviço (IRSA). A segunda abordagem usa a Identidade de Pods. Cada abordagem tem seus benefícios e casos de uso.

ASCP com perfis do IAM para contas de serviço (IRSA)

O ASCP com funções do IAM para contas de serviço (IRSA) permite que você monte segredos AWS Secrets Manager como arquivos em seus pods do Amazon EKS. Essa abordagem é adequada quando:

- É necessário montar os segredos como arquivos nos seus pods.
- Você está usando a versão 1.17 ou posterior do Amazon EKS com grupos de nós do Amazon EC2.
- Você deseja recuperar pares de chave-valor específicos de segredos formatados em JSON.

Para obter mais informações, consulte the section called “Integrar o ASCP com IRSA para o Amazon EKS”.

ASCP com Identidade de Pods

ASCP com Identidade de Pods do EKS

O método ASCP com Identidade de Pods aumenta a segurança e simplifica a configuração para acessar os segredos no Amazon EKS. Essa abordagem é benéfica quando:

- É necessário um gerenciamento de permissões mais granular no nível do pod.

- Você está usando a versão 1.24 ou posterior do Amazon EKS.
- Você quer performance e escalabilidade aprimorados.

Para obter mais informações, consulte the section called “Integrar o ASCP com a Identidade de Pods para Amazon EKS”.

Como escolher a abordagem correta

Considere os seguintes fatores ao decidir entre o ASCP com IRSA e o ASCP com Identidade de Pods:

- Amazon EKSversion: O Pod Identity requer o Amazon EKS 1.24+, enquanto o driver CSI funciona com o Amazon EKS 1.17+.
- Requisitos de segurança: a Identidade de Pods oferece um controle mais granular no nível do pod.
- Performance: a Identidade de Pods geralmente tem melhor performance em ambientes de alta escala.
- Complexidade: a Identidade de Pods simplifica a configuração ao eliminar a necessidade de contas de serviço separadas.

Escolha o método que melhor se alinhe aos seus requisitos específicos e ao ambiente do Amazon EKS.

Instalar o ASCP para Amazon EKS

Esta seção explica como instalar o AWS Secrets and Configuration Provider para o Amazon EKS. Com o ASCP, você pode montar segredos do Secrets Manager e parâmetros AWS Systems Manager como arquivos nos pods do Amazon EKS.

Pré-requisitos

- Um cluster do Amazon EKS
  - Versão 1.24 ou posterior da Identidade de Pods
  - Versão 1.17 ou posterior do IRSA
- O AWS CLI instalado e configurado
- kubectl instalado e configurado para o cluster do Amazon EKS
- Helm (versão 3.0 ou superior)

Instalar e configurar o ASCP

O ASCP está disponível GitHub no repositório secrets-store-csi-provider-aws. O repositório também contém arquivos YAML de exemplo para criar e montar um segredo.

Durante a instalação, é possível configurar o ASCP para usar um endpoint do FIPS. Para uma lista de endpoints , consulte the section called “Endpoint do Secrets Manager”.

Para instalar o ASCP como um complemento do EKS

1. Instalar `eksctl` (instruções de instalação)
2. Execute o comando a seguir para instalar o complemento com a configuração padrão:

```
eksctl create addon --cluster <your_cluster> --name aws-secrets-store-csi-driver-
provider
```

Se você quiser configurar o complemento, execute o seguinte comando de instalação:

```
aws eks create-addon --cluster-name <your_cluster> --addon-name aws-secrets-store-
csi-driver-provider --configuration-values 'file://path/to/config.yaml'
```

O arquivo de configuração pode ser um arquivo YAML ou JSON. Para ver o esquema de configuração do complemento:

a. Execute o comando a seguir e observe a versão mais recente do complemento:

```
aws eks describe-addon-versions --addon-name aws-secrets-store-csi-driver-
provider
```

b. Execute o comando a seguir para ver o esquema de configuração do complemento, `<version>` substituindo-o pela versão da etapa anterior:

```
aws eks describe-addon-configuration --addon-name aws-secrets-store-csi-driver-
provider --addon-version <version>
```

Para instalar o ASCP usando o Helm

1. Para garantir que o repositório esteja apontando para os gráficos mais recentes, use `helm repo update`.

2. Instale o chart. Veja a seguir um exemplo do comando `helm install`:

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

   a. Para usar um endpoint do FIPS, adicione o sinalizador a seguir: `--set useFipsEndpoint=true`

   b. Para configurar o controle de utilização, adicione o sinalizador a seguir: `--set-json 'k8sThrottlingParams={"qps": "number of queries per second", "burst": "number of queries per second"}'`

   c. Se o driver CSI do Secrets Store já estiver instalado em seu cluster, adicione o sinalizador a seguir: `--set secrets-store-csi-driver.install=false`. Isso ignorará a instalação do driver CSI do Secrets Store como uma dependência.

Para instalar usando o YAML no repositório

- Use os seguintes comandos.

```
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
```

Verificar as instalações

Para verificar as instalações do cluster do EKS, do driver CSI do Secrets Store e do plug-in do ASCP, siga estas etapas:

1. Verifique o cluster do EKS:

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

   Este comando deve retornar informações sobre o cluster.

2. Verifique a instalação do driver CSI do Secrets Store:

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

É necessário ver pods em execução com nomes como `csi-secrets-store-secrets-store-csi-driver-xxx`.

3. Verifique a instalação do plug-in do ASCP:

   YAML installation

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

   Resultado do exemplo:

   ```
   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
   ```

   Resultado do exemplo:

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

   É necessário ver pods no estado `Running`.

Depois de executar esses comandos, se tudo estiver configurado corretamente, será necessário ver todos os componentes funcionando sem erros. Se você encontrar algum problema, talvez seja necessário solucionar verificando os logs dos pods específicos que estão com problemas.

Solução de problemas

1. Para verificar os logs do provedor do ASCP, execute:

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

2. Verifique o status de todos os pods no namespace `kube-system`.

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

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

Todos os pods relacionados ao driver CSI e ao ASCP devem estar no estado “Running”.

3. Verifique a versão do driver CSI:

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

Este comando deve retornar informações sobre o driver CSI instalado.

Recursos adicionais do

Para obter mais informações sobre o uso do ASCP com o Amazon EKS, consulte os seguintes recursos:

- Uso da Identidade de Pods com o Amazon EKS
- AWS Driver CSI da Secrets Store ativado GitHub

Use AWS segredos e o provedor de configuração CSI com o Pod Identity para Amazon EKS

A integração do AWS Secrets and Configuration Provider com o Pod Identity Agent for Amazon Elastic Kubernetes Service fornece segurança aprimorada, configuração simplificada e desempenho aprimorado para aplicativos executados no Amazon EKS. O Pod Identity simplifica a autenticação do IAM para o Amazon EKS ao recuperar segredos do Secrets Manager ou AWS Systems Manager parâmetros do Parameter Store.

A Identidade de Pods do Amazon EKS simplifica o processo de configuração de permissões do IAM para aplicações do Kubernetes, possibilitando que as permissões sejam configuradas diretamente por meio das interfaces do Amazon EKS, reduzindo o número de etapas e eliminando a necessidade

de alternar entre os serviços do Amazon EKS e do IAM. A Identidade de Pods permite o uso de um único perfil do IAM em vários clusters sem atualizar as políticas de confiança, e oferece suporte a tags de sessão de perfis para um controle de acesso mais granular. Essa abordagem não apenas simplifica o gerenciamento de políticas, permitindo a reutilização de políticas de permissão em todas as funções, mas também aumenta a segurança ao permitir o acesso a AWS recursos com base nas tags correspondentes.

Como funciona

1. A Identidade de Pods atribui um perfil do IAM ao pod.
2. O ASCP usa essa função para se autenticar com. Serviços da AWS
3. Se autorizado, o ASCP recupera os segredos solicitados e os disponibiliza para o pod.

Para obter mais informações, consulte Como a Identidade de Pods do Amazon EKS funciona no Guia do usuário do Amazon EKS.

Pré-requisitos

> **Important**
>
> A Identidade de Pods é compatível somente com o Amazon EKS na nuvem. Ela não é compatível com o Amazon EKS Anywhere, o Serviço Red Hat OpenShift na AWS ou com clusters autogerenciados do Kubernetes em instâncias do Amazon EC2.

- Cluster do Amazon EKS (versão 1.24 ou posterior)
- Acesso a um AWS CLI cluster Amazon EKS via `kubectl`
- Acesso a dois Contas da AWS (para acesso entre contas)

Instalar o agente da Identidade de Pods do Amazon EKS

Para usar a Identidade de Pods com o cluster, é necessário instalar o complemento do agente da Identidade de Pods do Amazon EKS.

Para instalar o complemento agente da Identidade de Pods

- Instale o complemento do agente de Identidade de Pods no seu cluster.

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

Configurar o ASCP com a Identidade de Pods

1. Crie uma política de permissões que conceda as permissões `secretsmanager:GetSecretValue` e `secretsmanager:DescribeSecret` aos segredos que o pod precisa acessar. Para visualizar um exemplo de política, consulte the section called "Exemplo: permissão para ler e descrever segredos individuais".
2. Crie um perfil do IAM que possa ser assumido pela entidade principal do serviço do Amazon EKS para a Identidade de Pods:

   JSON

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

   Anexe a política do IAM ao perfil.

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

3. Crie uma associação de Identidade de Pods. Para obter um exemplo, consulte Criar uma associação de Identidade de Pods no Guia do usuário do Amazon EKS
4. Crie a `SecretProviderClass` que especifica quais segredos devem ser montados no pod:

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

   A principal diferença no `SecretProviderClass` entre o IRSA e a Identidade de Pods é o parâmetro opcional `usePodIdentity`. É um campo opcional que determina a abordagem de autenticação. Quando não especificado, o padrão é usar os perfis do IAM para contas de serviço (IRSA).

   - Para usar a Identidade de Pods do EKS, use qualquer um destes valores: `"true", "True", "TRUE", "t", "T"`.
   - Para usar explicitamente o IRSA, defina para qualquer um destes valores: `"false", "False", "FALSE", "f", or "F"`.
5. Implante o pod que monta os segredos em `/mnt/secrets-store`:

   ```
   kubectl apply -f https://raw.githubusercontent.com/aws/secrets-store-csi-driver-provider-aws/main/examples/ExampleDeployment-PodIdentity.yaml
   ```
6. Se você usa um cluster privado do Amazon EKS, certifique-se de que a VPC na qual o cluster está tenha um AWS STS endpoint. Para obter informações sobre a criação de um endpoint, consulte Endpoints da VPC da interface no Guia do usuário do AWS Identity and Access Management .

Verificar a montagem do segredo

Para verificar se o segredo foi montado corretamente, execute o comando a seguir.

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

Para configurar a Identidade de Pods do Amazon EKS para acessar os segredos no Secrets Manager

1. Crie uma política de permissões que conceda as permissões `secretsmanager:GetSecretValue` e `secretsmanager:DescribeSecret` aos segredos

que o pod precisa acessar. Para visualizar um exemplo de política, consulte the section called "Exemplo: permissão para ler e descrever segredos individuais".

2. Crie um segredo no Secrets Manager, caso ainda não tenha um.

Solução de problemas

É possível visualizar a maioria dos erros ao descrever a implantação do pod.

Para ver mensagens de erro para o contêiner

1. Obtenha uma lista de nomes de pods com o comando a seguir. Se você não estiver usando o namespace padrão, use `-n` *NAMESPACE*.

```
kubectl get pods
```

2. Para descrever o pod, no comando a seguir, *PODID* use o ID do pod dos pods que você encontrou na etapa anterior. Se você não estiver usando o namespace padrão, use `-n` *NAMESPACE*.

```
kubectl describe pod/PODID
```

Para ver erros para o ASCP

- Para encontrar mais informações nos registros do provedor, no comando a seguir, *PODID* use o ID do csi-secrets-store-provider-aws Pod.

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

Use AWS segredos e CSI do provedor de configuração com funções do IAM para contas de serviço (IRSA)

Tópicos

- Pré-requisitos
- Configurar o controle de acesso
- Identificar quais segredos montar

- Solução de problemas

Pré-requisitos

- Cluster do Amazon EKS (versão 1.17 ou posterior)
- Acesso a um AWS CLI cluster Amazon EKS via `kubectl`

Configurar o controle de acesso

O ASCP recupera a Identidade de Pods do Amazon EKS e a substitui por um perfil do IAM. Você define permissões em uma política do IAM para esse perfil do IAM. Quando o ASCP assume o perfil do IAM, ele obtém acesso aos segredos que você autorizou. Outros contêineres não podem acessar os segredos, a menos que você também os associe ao perfil do IAM.

Para conceder ao seu pod do Amazon EKS acesso aos segredos no Secrets Manager

1. Crie uma política de permissões que conceda as permissões `secretsmanager:GetSecretValue` e `secretsmanager:DescribeSecret` aos segredos que o pod precisa acessar. Para visualizar um exemplo de política, consulte the section called “Exemplo: permissão para ler e descrever segredos individuais”.
2. Crie um provedor IAM OIDC Connect (OIDC) para o cluster, se você ainda não tiver um. Para obter mais informações, consulte Criação de um provedor IAM OIDC para o seu cluster no Guia do Usuário do Amazon EKS.
3. Crie um perfil do IAM para conta de serviço e anexe a política a ele. Para obter mais informações, consulte Criação de um perfil do IAM para uma conta de serviço no Guia do usuário do Amazon EKS.
4. Se você usa um cluster privado do Amazon EKS, certifique-se de que a VPC na qual o cluster está tenha um AWS STS endpoint. Para obter informações sobre a criação de um endpoint, consulte Endpoints da VPC da interface no Guia do usuário do AWS Identity and Access Management .

Identificar quais segredos montar

Para determinar quais segredos o ASCP monta no Amazon EKS como arquivos no sistema de arquivos, você cria um arquivo YAML the section called “SecretProviderClass”. O `SecretProviderClass` lista os segredos a serem montados e o nome do arquivo no qual montá-

los. O`SecretProviderClass` deve estar no mesmo namespace que o pod do Amazon EKS ao qual ele faz referência.

Montagem dos segredos como arquivos

As instruções a seguir mostram como montar segredos como arquivos usando exemplos de arquivos YAML .yaml e ExampleSecretProviderClass.yaml. ExampleDeployment

Para montar segredos no Amazon EKS

1. Aplique o `SecretProviderClass` ao pod:

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

2. Implante o pod:

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

3. O ASCP monta os arquivos.

Solução de problemas

É possível visualizar a maioria dos erros ao descrever a implantação do pod.

Para ver mensagens de erro para o contêiner

1. Obtenha uma lista de nomes de pods com o comando a seguir. Se você não estiver usando o namespace padrão, use `-n` *`nameSpace`*.

```
kubectl get pods
```

2. Para descrever o pod, no comando a seguir, *`podId`* use o ID do pod dos pods que você encontrou na etapa anterior. Se você não estiver usando o namespace padrão, use `-n` *`nameSpace`*.

```
kubectl describe pod/podId
```

Para ver erros para o ASCP

- Para encontrar mais informações nos registros do provedor, no comando a seguir, *podId* use o ID do csi-secrets-store-provider-aws Pod.

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

- Verifique se a CRD do **SecretProviderClass** está instalada:

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

  Este comando deve retornar informações sobre a definição do recurso personalizado `SecretProviderClass`.

- Verifique se o SecretProviderClass objeto foi criado.

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

AWS Segredos e exemplos de código do provedor de configuração

Exemplos de autenticação do ASCP e controle de acesso

Exemplo: política do IAM que permite que o serviço da Identidade de Pods do Amazon EKS (pods.eks.amazonaws.com) assuma o perfil e marque a sessão:

JSON

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

```
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

SecretProviderClass

Você usa o YAML para descrever quais segredos devem ser montados no Amazon EKS usando o ASCP. Para obter exemplos, consulte the section called “SecretProviderClass uso”.

SecretProviderClass Estrutura YAML

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

O campo de parâmetros contém os detalhes da solicitação de montagem:

region

(Opcional) O Região da AWS do segredo. Se você não usar esse campo, o ASCP procurará a Região a partir da anotação no nó. Essa pesquisa adiciona sobrecarga às solicitações de montagem, portanto, recomendamos que você forneça a região para clusters que usam um grande número de pods.

Se você também especificar `failoverRegion`, o ASCP tentará recuperar o segredo de ambas as regiões. Se qualquer uma das regiões retornar um erro 4xx (p. ex., para um problema de autenticação), o ASCP não montará nenhum segredo. Se o segredo for recuperado com êxito de `region`, o ASCP montará o valor desse segredo. Se o segredo não for recuperado com êxito de

`region`, mas for recuperado com êxito de `failoverRegion`, o ASCP montará o valor desse segredo.

failoverRegion

(Opcional) Se você incluir esse campo, o ASCP tentará recuperar o segredo das regiões definidas em `region` e nesse campo. Se qualquer uma das regiões retornar um erro 4xx (p. ex., para um problema de autenticação), o ASCP não montará nenhum segredo. Se o segredo for recuperado com êxito de `region`, o ASCP montará o valor desse segredo. Se o segredo não for recuperado com êxito de `region`, mas for recuperado com êxito de `failoverRegion`, o ASCP montará o valor desse segredo. Para obter um exemplo de como usar esse campo, consulte Failover multirregiões.

pathTranslation

(Opcional) Um só caractere de substituição para usar se o nome de arquivo no Amazon EKS vá conter o caractere separador de caminho, como barra (/) no Linux. O ASCP não poderá criar um arquivo montado que contenha um caractere separador de caminho. Em vez disso, o ASCP substituirá o caractere separador de caminho por outro caractere. Se você não usar esse campo, o caractere substituto será um sublinhado (_), portanto, por exemplo,`My/Path/Secret` será montado como `My_Path_Secret`.

Para impedir a substituição de caracteres, digite a string`False`.

usePodIdentity

(Opcional) Determina a abordagem de autenticação. Quando não especificado, o padrão é Perfis do IAM para contas de serviço (IRSA).

- Para usar a Identidade de Pods do EKS, use qualquer um destes valores: `"true"`, `"True"`, `"TRUE"`, `"t"` ou `"T"`.
- Para usar explicitamente o IRSA, defina qualquer um destes valores: `"false"`, `"False"`, `"FALSE"`, `"f"` ou `"F""=`.

preferredAddressType

(Opcional) Especifica o tipo de endereço IP preferencial para comunicação de endpoints do atendente da Identidade de Pods. O campo só será aplicável ao usar o recurso de Identidade de Pods do EKS, e será ignorado ao usar perfis do IAM para contas de serviço. Os valores não diferenciam maiúsculas de minúsculas. Os valores válidos são:

- `"ipv4"`, `"IPv4"` “, ou `"IPV4"` — Forçar o uso do IPv4 endpoint do Pod Identity Agent
- `"ipv6"`,`"IPv6"`, ou `"IPV6"` — Forçar o uso do IPv6 endpoint do Pod Identity Agent

- não especificado — Use a seleção automática de endpoint, testando primeiro o IPv4 endpoint e retornando ao IPv6 endpoint se falhar IPv4

objects

Uma string contendo uma declaração do YAML dos segredos a serem montados. Recomendamos o uso de uma string com várias linhas ou um caractere pipe (|) no YAML.

objectName

Obrigatório. Especifica o nome do parâmetro ou segredo a ser buscado. Para o Secrets Manager, este é o parâmetro `SecretId` e pode ser o nome amigável ou o ARN completo do segredo. Para o SSM Parameter Store, esse é o `Name` do parâmetro e pode ser o nome ou o ARN completo do parâmetro.

objectType

Necessário se você não usar um ARN do Secrets Manager para`objectName`. Pode ser `secretsmanager` ou `ssmparameter`.

objectAlias

(Opcional) O nome do arquivo do segredo no pod do Amazon EKS. Se você não especificar esse campo, o `objectName` aparece como o nome do arquivo.

filePermission

(Opcional) A string octal de 4 dígitos que especifica a permissão do arquivo para montar o segredo. Se você não especificar esse campo, ele será padronizado como `"0644"`.

objectVersion

(Opcional) O ID da versão do segredo. Não recomendado porque é necessário atualizar o ID da versão sempre que atualizar o segredo. Por padrão, a versão mais recente é usada. Se você incluir um `failoverRegion`, esse campo representará o `objectVersion` primário.

objectVersionLabel

(Opcional) O alias da versão. O padrão é a versão mais recente AWSCURRENT. Para obter mais informações, consulte the section called “Versões do segredo”. Se você incluir um `failoverRegion`, esse campo representará o `objectVersionLabel` primário.

jmesPath

(Opcional) Um mapa das chaves no segredo para os arquivos a serem montados no Amazon EKS. Para usar esse campo, o valor do segredo deve estar no formato JSON. Se você usar esse campo, deverá incluir os subcampos `path` e `objectAlias`.

caminho

Uma chave de um par de chave-valor no JSON do valor do segredo. Se o campo contiver um hífen, use aspas simples para delimitá-lo, por exemplo: `path: '"hyphenated-path"'`

objectAlias

O nome do arquivo a ser montado no pod Amazon EKS. Se o campo contiver um hífen, use aspas simples para delimitá-lo, por exemplo: `objectAlias: '"hyphenated-alias"'`

filePermission

(Opcional) A string octal de 4 dígitos que especifica a permissão do arquivo para montar o segredo. Se você não especificar esse campo, ele será padronizado como a permissão de arquivo do objeto pai.

failoverObject

(Opcional) Se você especificar esse campo, o ASCP tentará recuperar o segredo especificado no `objectName` primário e o segredo especificado no subcampo `objectName` de `failoverObject`. Se qualquer um deles retornar um erro 4xx (p. ex., para um problema de autenticação), o ASCP não montará nenhum segredo. Se o segredo for recuperado com êxito do `objectName` primário, o ASCP montará o valor desse segredo. Se o segredo não for recuperado com êxito do `objectName` primário, mas for recuperado com êxito do `objectName` do failover, o ASCP montará o valor desse segredo. Se incluir esse campo, será necessário incluir o campo `objectAlias`. Para obter um exemplo de como usar esse campo, consulte Failover para um segredo diferente.

Normalmente, você usa esse campo quando o segredo de failover não for uma réplica. Para obter um exemplo de como especificar uma réplica, consulte Failover multirregiões.

objectName

O nome ou ARN completo do segredo de failover. Se você usar um ARN, a região no ARN deverá corresponder ao campo `failoverRegion`.

objectVersion

(Opcional) O ID da versão do segredo. Deve corresponder ao `objectVersion` primário. Não recomendado porque é necessário atualizar o ID da versão sempre que atualizar o segredo. Por padrão, a versão mais recente é usada.

objectVersionLabel

(Opcional) O alias da versão. O padrão é a versão mais recente AWSCURRENT. Para obter mais informações, consulte the section called “Versões do segredo”.

Crie uma SecretProviderClass configuração básica para montar segredos em seus pods do Amazon EKS.

Pod Identity

SecretProviderClass para usar um segredo no mesmo cluster do 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 uso

Use esses exemplos para criar SecretProviderClass configurações para diferentes cenários.

Exemplo: montar segredos por nome ou ARN

Este exemplo mostra como montar três tipos diferentes de segredos:

- Um segredo especificado por um ARN completo
- Um segredo especificado pelo nome
- Uma versão específica de um segredo

```
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"
```

Exemplo: montar pares de chave-valor a partir de um segredo

Este exemplo mostra como montar pares de chave-valor específicos de um segredo formatado em 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
```

Exemplo: montar segredos por permissão de arquivo

Este exemplo mostra como montar um segredo com uma permissão específica de arquivo

```
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"
```

Exemplo: exemplos de configuração de failover

Estes exemplos mostram como configurar o failover para segredos.

Failover multirregiões

Este exemplo mostra como configurar o failover automático para um segredo replicado em várias regiões:

```
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"
```

Failover para um segredo diferente

Este exemplo mostra como configurar o failover para um segredo diferente (não uma réplica):

```
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"
```

Recursos adicionais do

Para obter mais informações sobre o uso do ASCP com o Amazon EKS, consulte os seguintes recursos:

- Uso da Identidade de Pods com o Amazon EKS
- Usando AWS segredos e provedor de configuração
- AWS Driver CSI da Secrets Store ativado GitHub

Use AWS Secrets Manager segredos em AWS Lambda funções

AWS Lambda é um serviço de computação sem servidor que permite executar código sem provisionar ou gerenciar servidores. O Parameter Store, um recurso do AWS Systems Manager, fornece armazenamento seguro e hierárquico para gerenciamento de dados de configuração e gerenciamento de segredos. Você pode usar a extensão Lambda de AWS parâmetros e segredos para recuperar e armazenar em cache AWS Secrets Manager segredos e parâmetros do Parameter Store em funções do Lambda sem usar um SDK. Para obter informações detalhadas sobre o uso

dessa extensão, consulte Uso de segredos do Secrets Manager em funções do Lambda no Guia do desenvolvedor do Lambda.

Uso de segredos do Secrets Manager com o Lambda

O Guia do desenvolvedor do Lambda fornece instruções abrangentes para usar os segredos do Secrets Manager em funções do Lambda. Para começar:

1. Siga o step-by-step tutorial em Usar segredos do Secrets Manager nas funções do Lambda, que inclui:
   - Criação de uma função do Lambda com seu runtime preferido (Python, Node.js, Java)
   - Adicionando a extensão Lambda de AWS parâmetros e segredos como uma camada
   - Configuração das permissões necessárias
   - Escrita de código para recuperação de segredos da extensão
   - Teste da sua função
2. Saiba mais sobre variáveis de ambiente para configuração do comportamento da extensão, incluindo configurações de cache e tempos limite
3. Práticas recomendadas para trabalho com alternância de segredos

Uso do Secrets Manager e do Lambda em uma VPC

Se sua função do Lambda for executada em uma VPC, será necessário criar um endpoint da VPC para que a extensão possa fazer chamadas para o Secrets Manager. Para obter mais informações, consulte the section called “Endpoints da VPC (AWS PrivateLink)”.

Usando a extensão Lambda de AWS parâmetros e segredos

A extensão pode recuperar segredos do Secrets Manager e parâmetros do Parameter Store. Para obter informações detalhadas sobre o uso dos parâmetros do Parameter Store com a extensão, consulte Uso de parâmetros do Parameter Store em funções do Lambda no Guia do usuário do AWS Systems Manager .

A documentação do Systems Manager inclui:

- Explicação detalhada de como a extensão funciona com o Parameter Store
- Instruções para a adição da extensão a uma função do Lambda
- Variáveis de ambiente para configuração da extensão

- Comandos de exemplo para recuperação de parâmetros
- Lista completa de extensões ARNs para todas as arquiteturas e regiões suportadas

Usando o AWS Secrets Manager agente

Como o Agente do Secrets Manager funciona

O AWS Secrets Manager Agent é um serviço HTTP do lado do cliente que ajuda você a padronizar como você consome segredos do Secrets Manager em seus ambientes computacionais. É possível usá-lo com serviços a seguir:

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

O Agente do Secrets Manager recupera e armazena segredos em cache na memória, permitindo que suas aplicações obtenham segredos do localhost em vez de fazer chamadas diretas para o Secrets Manager. O Agente do Secrets Manager só pode ler segredos, ele não pode modificá-los.

> **Important**
>
> O Secrets Manager Agent usa as AWS credenciais do seu ambiente para chamar o Secrets Manager. Isso inclui proteção contra falsificação de solicitações do lado do servidor (SSRF) para ajudar a melhorar a segurança dos segredos. O Agente do Secrets Manager usa a troca de chaves pós-quântica ML-KEM como a troca de chaves de maior prioridade por padrão.

Noções básicas sobre o armazenamento em cache do Agente do Secrets Manager

O Agente do Secrets Manager usa um cache em memória, ele é redefinido quando o Agente do Secrets Manager é reiniciado. Ele atualiza periodicamente os valores de segredos em cache com base no seguinte:

- A frequência de atualização padrão (TTL) é de 300 segundos

- É possível modificar o TTL usando um arquivo de configuração
- A atualização ocorre quando você solicita um segredo após a expiração do TTL

> **Note**
>
> O Agente do Secrets Manager não inclui a invalidação do cache. Se um segredo for alternado antes que a entrada do cache expire, o Agente do Secrets Manager poderá retornar um valor de segredo obsoleto.

O Agente do Secrets Manager retorna valores de segredos no mesmo formato da resposta de `GetSecretValue`. Os valores de segredos não são criptografados no cache.

Tópicos

Criação do Agente do Secrets Manager

Antes de começar, verifique se você tem as ferramentas de desenvolvimento padrão e as ferramentas do Rust instaladas em sua plataforma.

> **Note**
>
> Atualmente, criar o agente com o recurso `fips` ativado no macOS requer a solução alternativa a seguir:
>
> - Crie uma variável de ambiente chamada `SDKROOT` que é definida como o resultado da execução de `xcrun --show-sdk-path`

RPM-based systems

Para criar em sistemas baseados em RPM

1. Use o script `install` fornecido no repositório.

   O script gera um token SSRF aleatório no startup e o armazena no arquivo `/var/run/awssmatoken`. O token pode ser lido pelo grupo `awssmatokenreader` criado pelo script de instalação.

2. Para permitir que sua aplicação leia o arquivo de token, você precisa adicionar ao grupo `awssmatokenreader` a conta de usuário na qual sua aplicação é executada. Por exemplo, você pode conceder permissões para que seu aplicativo leia o arquivo de token com o seguinte comando usermod, onde *<APP_USER>* está o ID do usuário sob o qual seu aplicativo é executado.

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

   Instalação das ferramentas de desenvolvimento

   Em sistemas baseados em RPM AL2023, como, instale o grupo Ferramentas de Desenvolvimento:

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

3. Instalação do Rust

   Siga as instruções em Instalação do Rust na documentação do Rust.

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

4. Criação do agente

   Crie o Agente do Secrets Manager usando o comando cargo build:

   ```
   cargo build --release
   ```

   Você encontrará o executável em `target/release/aws_secretsmanager_agent`.

Debian-based systems

Para criar em sistemas baseados em Debian

1. Instalação das ferramentas de desenvolvimento

   Em sistemas baseados no Debian, como o Ubuntu, instale o pacote build-essential.

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

2. Instalação do Rust

   Siga as instruções em Instalação do Rust na documentação do Rust.

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

3. Criação do agente

   Crie o Agente do Secrets Manager usando o comando cargo build:

   ```
   cargo build --release
   ```

   Você encontrará o executável em `target/release/aws_secretsmanager_agent`.

Windows

Para criar no Windows

1. Configuração do ambiente de desenvolvimento

   Siga as instruções em Configuração do seu ambiente de desenvolvimento no Windows para Rust na documentação do Microsoft Windows.

2. Criação do agente

   Crie o Agente do Secrets Manager usando o comando cargo build:

   ```
   cargo build --release
   ```

Você encontrará o executável em `target/release/aws_secretsmanager_agent.exe`.

Cross-compile natively

Para compilar de forma cruzada

1. Instalação das ferramentas de compilação cruzada

   Em distribuições em que o pacote mingw-w64 está disponível, como o Ubuntu, instale o conjunto de ferramentas de compilação cruzada.

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

2. Adição de destinos de criação do Rust

   Instale o destino de compilação do Windows GNU:

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

3. Criação no Windows

   Faça a compilação cruzada do agente para Windows:

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

   Você encontrará o executável em `target/x86_64-pc-windows-gnu/release/aws_secretsmanager_agent.exe`.

Cross compile with Rust cross

Para fazer a compilação cruzada usando o Rust cross

Se as ferramentas de compilação cruzada não estiverem disponíveis nativamente no sistema, será possível usar o projeto Rust cross. Para obter mais informações, consulte https://github.com/cross-rs/cross.

> **Important**
>
> Recomendamos 32 GB de espaço em disco para o ambiente de compilação.

1. Configurar o Docker

   Instalação e configuração do Docker

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

2. Configuração das permissões do Docker

   Adicione seuusuário ao grupo do docker:

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

3. Criação no Windows

   Instale o cross e crie o executável:

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

Instalação do Agente do Secrets Manager

Escolha seu ambiente de computação entre as opções de instalação a seguir.

Amazon EC2

Para instalar o Agente do Secrets Manager no Amazon EC2

1. Navegação até o diretório de configurações

   Mude para o diretório de configurações:

```
cd aws_secretsmanager_agent/configuration
```

2. Execução do script de instalação

   Execute o script `install` fornecido no repositório.

   O script gera um token SSRF aleatório no startup e o armazena no arquivo `/var/run/awssmatoken`. O token pode ser lido pelo grupo `awssmatokenreader` criado pelo script de instalação.

3. Configuração de permissões de aplicações

   Adicione a conta de usuário na qual sua aplicação é executada ao grupo `awssmatokenreader`:

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

   Substitua pelo ID de usuário *APP_USER* com o qual seu aplicativo é executado.

Container Sidecar

É possível executar o Agente do Secrets Manager como um contêiner auxiliar junto com sua aplicação usando o Docker. Então, sua aplicação pode recuperar segredos do servidor HTTP local fornecido pelo Agente do Secrets Manager. Para obter informações sobre o Docker, consulte a documentação do Docker.

Para criar um contêiner auxiliar para o Agente do Secrets Manager

1. Criação do Dockerfile do agente

   Crie um Dockerfile para o contêiner auxiliar do Agente do 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. Criação do Dockerfile da aplicação

   Crie um Dockerfile para sua aplicação cliente.

3. Criação do arquivo Docker Compose

   Crie um arquivo Docker Compose para executar ambos os contêineres com uma interface de rede compartilhada:

> **Important**
>
> Você deve carregar AWS as credenciais e o token SSRF para que o aplicativo possa usar o Secrets Manager Agent. Para o Amazon EKS e o Amazon ECS, consulte:
>
> - Gerenciamento de acesso no Guia do usuário do Amazon EKS
> - Perfil do IAM da tarefa do Amazon ECS no Guia do desenvolvedor do 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. Cópia de binário do agente

   Copie o binário do `secrets-manager-agent` para o mesmo diretório que contém seus arquivos Dockerfile e Docker Compose.

5. Criação e execução dos contêineres

   Crie e execute os contêineres usando o Docker Compose:

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

6. Próximas etapas

   Agora é possível usar o Agente do Secrets Manager para recuperar segredos do seu contêiner cliente. Para obter mais informações, consulte the section called “Recuperação de segredos com o Agente do Secrets Manager”.

Lambda

É possível empacotar o Agente do Secrets Manager como uma extensão do LAmbda. Em seguida, é possível adicioná-lo à sua função do Lambda como uma camada e chamar o Agente do Secrets Manager a partir da sua função da Lambda para obter segredos.

As instruções a seguir mostram como obter um nome secreto MyTestusando o script `secrets-manager-agent-extension.sh` de exemplo https://github.com/aws/aws-secretsmanager-agentpara instalar o Secrets Manager Agent como uma extensão Lambda.

Para criar uma extensão do Lambda para o Agente do Secrets Manager

1. Empacotamento da camada do agente

   Na raiz do pacote de código do Agente do Secrets Manager, execute os comandos a seguir:

   ```
   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. Configurar o token de SSRF

   A configuração padrão do agente definirá automaticamente o token SSRF com o valor definido nas variáveis predefinidas `AWS_SESSION_TOKEN` ou de `AWS_CONTAINER_AUTHORIZATION_TOKEN` ambiente (a última variável para funções Lambda com habilitada). SnapStart Como alternativa, é possível definir a variável de ambiente `AWS_TOKEN` com um valor arbitrário para sua função do Lambda, pois essa variável tem precedência sobre as outras duas. Se você optar por usar a variável de ambiente `AWS_TOKEN`, deverá definir essa variável de ambiente com uma chamada a `lambda:UpdateFunctionConfiguration`.

3. Vinculação da camada à função

   Vincule a versão da camada à sua função do Lambda.

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

4. Atualizar um código de função

Atualize sua função do Lambda para fazer uma consulta a `http://localhost:2773/secretsmanager/get?secretId=MyTest` com o valor de cabeçalho `X-Aws-codes-Secrets-Token` definido como o valor do token de SSRF proveniente de uma das variáveis de ambiente mencionadas acima para recuperar o segredo. Certifique-se de implementar a lógica de repetição no código da aplicação para acomodar atrasos na inicialização e no registro da extensão do Lambda.

5. Testar a função

   Invoque a função do Lambda para verificar se o segredo está sendo buscado corretamente.

Recuperação de segredos com o Agente do Secrets Manager

Para recuperar um segredo, chame o endpoint local do Agente do Secrets Manager e incluia o nome ou ARN do segredo como um parâmetro de consulta. Por padrão, o Agente do Secrets Manager recupera a versão `AWSCURRENT` do segredo. Para recuperar uma versão diferente, use o parâmetro versionStage ou versionId.

> **Important**
>
> Para ajudar a proteger o Agente do Secrets Manager, é necessário incluir um cabeçalho de token SSRF como parte de cada solicitação: `X-Aws-Parameters-Secrets-Token`. O Agente do Secrets Manager nega solicitações que não tenham esse cabeçalho ou que tenham um token SSRF inválido. É possível personalizar o nome do cabeçalho SSRF em the section called “Opções de configuração”.

Permissões obrigatórias

O Secrets Manager Agent usa o AWS SDK para Rust, que usa a cadeia de fornecedores de AWS credenciais. A identidade dessas credenciais do IAM determina as permissões que o Agente do Secrets Manager tem para recuperar segredos.

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

Para obter mais informações sobre permissões, consulte the section called “Referência de permissões”.

> **Important**
>
> Depois que o valor do segredo é inserido no Agente do Secrets Manager, qualquer usuário com acesso ao ambiente computacional e ao token SSRF pode acessar o segredo a partir do cache do Agente do Secrets Manager. Para obter mais informações, consulte the section called “Considerações sobre segurança”.

Exemplo de solicitações

curl

Example Exemplo: obtenção de um segredo usando curl

O exemplo de curl a seguir mostra como obter um segredo do Agente do Secrets Manager. O exemplo depende da presença do SSRF em um arquivo, que é onde ele é armazenado pelo script de instalação.

```
curl -v -H \\
    "X-Aws-Parameters-Secrets-Token: $(/var/run/awssmatoken)" \\
    'http://localhost:2773/secretsmanager/get?secretId=YOUR_SECRET_ID' \\
    echo
```

Python

Example Exemplo: obtenção de um segredo usando Python

O exemplo de Python a seguir mostra como obter um segredo do Agente do Secrets Manager. O exemplo depende da presença do SSRF em um arquivo, que é onde ele é armazenado pelo script de instalação.

```
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}")
```

Noções básicas sobre o parâmetro **`refreshNow`**

O Agente do Secrets Manager usa um cache em memória para armazenar valores de segredos, que são atualizados periodicamente. Por padrão, essa atualização ocorre quando você solicita um segredo após a expiração da vida útil (TTL), normalmente a cada 300 segundos. No entanto, essa abordagem às vezes pode resultar em valores de segredo obsoletos, especialmente se um segredo for alternado antes que a entrada do cache expire.

Para resolver essa limitação, o Agente do Secrets Manager oferece suporte a um parâmetro chamado `refreshNow` na URL. Você pode usar esse parâmetro para forçar uma atualização imediata do valor de um segredo, ignorando o cache e garantindo que você tenha o máximo up-to-date de informações.

Comportamento padrão (sem **`refreshNow`**)

- Usa valores em cache até que o TTL expire

- Atualiza segredos somente após o TTL (padrão de 300 segundos)
- Pode retornar valores obsoletos se os segredos forem alternados antes que o cache expire

Comportamento com **`refreshNow=true`**

- Ignora completamente o cache
- Recupera o valor do segredo mais recente diretamente do Secrets Manager
- Atualiza o cache com o novo valor e redefine o TTL
- Garante que você sempre obtenha o valor secreto mais atual

Atualização forçada de um valor de segredo

> **Important**
>
> O valor padrão de `refreshNow` é `false`. Quando definido como `true`, substitui o TTL especificado no arquivo de configuração do Agente do Secrets Manager e faz uma chamada de API para o Secrets Manager.

curl

Example Exemplo: atualização forçada de um segredo usando curl

O exemplo de curl a seguir mostra como forçar o Agente do Secrets Manager a atualizar o segredo. O exemplo depende da presença do SSRF em um arquivo, que é onde ele é armazenado pelo script de instalação.

```
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 Exemplo: atualização forçada de um segredo usando Python

O exemplo de Python a seguir mostra como obter um segredo do Agente do Secrets Manager. O exemplo depende da presença do SSRF em um arquivo, que é onde ele é armazenado pelo script de instalação.

```
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}")
```

Configurar o Agente do Secrets Manager

Para alterar a configuração do Agente do Secrets Manager, crie um arquivo de configuração TOML e, em seguida, chame `./aws_secretsmanager_agent --config config.toml`.

Opções de configuração

`log_level`

O nível de detalhes relatado nos logs do Agente do Secrets Manager: DEBUG, INFO, WARN, ERROR ou NONE. O padrão é INFO.

`log_to_file`

Se deve enviar o log para um arquivo ou para stdout/stderr: `true` ou `false`. O padrão é `true`.

`http_port`

A porta do servidor de HTTP local, no intervalo de 1024 a 65535. O padrão é 2773.

`region`

A AWS região a ser usada para solicitações. Se nenhuma região for especificada, o Agente do Secrets Manager determinará a região a partir do SDK. Para obter mais informações, consulte Especificação das suas credenciais e região padrão no Guia do desenvolvedor do SDK da AWS para Rust.

`ttl_seconds`

O TTL, em segundos, para os itens em cache, no intervalo de 0 a 3600. O padrão é de 300. 0 indica que não há armazenamento em cache.

`cache_size`

O número máximo de segredos que podem ser armazenados no cache, no intervalo de 1 a 1000. O padrão é 1000.

`ssrf_headers`

Uma lista de nomes de cabeçalhos que o Agente do Secrets Manager verifica para o token de SSRF. O padrão é “X-Aws-Parameters-Secrets-Token,”. X-Vault-Token

`ssrf_env_variables`

Uma lista de nomes de variáveis de ambiente que o Agente do Secrets Manager verifica em ordem sequencial para o token de SSRF. A variável de ambiente pode conter o token ou uma referência ao arquivo de token, como em: `AWS_TOKEN=file:///var/run/awssmatoken`. O padrão é "AWS_TOKEN, AWS_SESSION _TOKEN, AWS_CONTAINER _AUTHORIZATION_TOKEN”.

path_prefix

O prefixo do URI usado para determinar se a solicitação é baseada em caminho. O padrão é "/v1/".

max_conn

O número máximo de conexões de clientes HTTP que o Agente do Secrets Manager permite, na faixa de 1 a 1000. O padrão é 800.

Recursos opcionais

O Agente do Secrets Manager pode ser criado com recursos opcionais passando o sinalizador `--features` para `cargo build`. Os recursos disponíveis são:

Recursos de compilação

prefer-post-quantum

Torna `X25519MLKEM768` o algoritmo de troca de chaves de maior prioridade. Caso contrário, ele está disponível, mas não é de maior prioridade. `X25519MLKEM768`é um algoritmo híbrido de troca de post-quantum-secure chaves.

fips

Restringe os conjuntos de cifras usados pelo agente somente a cifras aprovadas pelo FIPS.

Registro em log

Log local

O Agente do Secrets Manager registra erros em log localmente no arquivo `logs/secrets_manager_agent.log` ou em stdout/stderr, dependendo da variável de configuração `log_to_file`. Quando sua aplicação chama o Agente do Secrets Manager para obter um segredo, essas chamadas aparecem no log local. Eles não aparecem nos CloudTrail registros.

Alternância de logs

O Agente do Secrets Manager e armazena até cinco arquivos de log no total e cria um novo arquivo de log quando o arquivo atinge 10 MB.

AWS registro de serviços

O registro não vai para o Secrets Manager, CloudTrail, ou CloudWatch. As solicitações para obter segredos do Agente do Secrets Manager não aparecem nesses logs. Quando o Secrets Manager Agent faz uma chamada para o Secrets Manager para obter um segredo, essa chamada é gravada CloudTrail com uma string de agente de usuário contendo`aws-secrets-manager-agent`.

É possível configurar as opções de log em the section called “Opções de configuração”.

Considerações sobre segurança

Domínio de confiança

Para uma arquitetura de agente, o domínio de confiança é onde o endpoint do agente e o token SSRF estão acessíveis, o que geralmente é o host inteiro. O domínio de confiança do Agente do Secrets Manager deve corresponder ao domínio em que as credenciais do Secrets Manager estão disponíveis para manter a mesma postura de segurança. Por exemplo, no Amazon EC2, o domínio de confiança do Agente do Secrets Manager seria o mesmo que o domínio das credenciais ao usar funções para o Amazon EC2.

Important

Aplicativos preocupados com a segurança que ainda não estão usando uma solução de agente com as credenciais do Secrets Manager bloqueadas no aplicativo devem considerar o uso de soluções específicas do idioma AWS SDKs ou de armazenamento em cache. Para obter mais informações, consulte Obtenção de segredos.

Obtenha um valor secreto do Secrets Manager usando o SDK para C++ AWS

Para aplicativos C++, chame o SDK diretamente com GetSecretValueou. BatchGetSecretValue

O exemplo de código a seguir mostra como obter um valor de segredo do Secrets Manager.

Permissões obrigatórias: `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();
}
```

Obtenha um valor secreto do Secrets Manager usando o JavaScript AWS SDK

Para JavaScript aplicativos, chame o SDK diretamente com `getSecretValue`ou `batchGetSecretValue`.

O exemplo de código a seguir mostra como obter um valor de segredo do Secrets Manager.

Permissões obrigatórias: `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;
  }
};
```

Obtenha um valor secreto do Secrets Manager usando o Kotlin SDK AWS

Para aplicativos Kotlin, chame o SDK diretamente com ou. GetSecretValueBatchGetSecretValue

O exemplo de código a seguir mostra como obter um valor de segredo do Secrets Manager.

Permissões obrigatórias: `secretsmanager:GetSecretValue`

```
suspend fun getValue(secretName: String?) {
    val valueRequest =
        GetSecretValueRequest {
            secretId = secretName
        }

    SecretsManagerClient.fromEnvironment { region = "us-east-1" }.use { secretsClient -
>
        val response = secretsClient.getSecretValue(valueRequest)
        val secret = response.secretString
        println("The secret value is $secret")
    }
}
```

Obtenha um valor secreto do Secrets Manager usando o AWS SDK do PHP

Para aplicações PHP, chame o SDK diretamente com `GetSecretValue` ou `BatchGetSecretValue`.

O exemplo de código a seguir mostra como obter um valor de segredo do Secrets Manager.

Permissões obrigatórias: `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
```

Obtenha um valor secreto do Secrets Manager usando o SDK Ruby AWS

Para aplicações Ruby, chame o SDK diretamente com `get_secret_value` ou `batch_get_secret_value`.

O exemplo de código a seguir mostra como obter um valor de segredo do Secrets Manager.

Permissões obrigatórias: `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
```

Obtenha um valor secreto usando o AWS CLI

Permissões obrigatórias: `secretsmanager:GetSecretValue`

Example Recuperar o valor secreto criptografado de um segredo

O exemplo de `get-secret-value` a seguir obtém o valor atual do segredo.

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

Example Recuperar o valor secreto anterior

O exemplo de `get-secret-value` a seguir recupera o valor secreto anterior.

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

Obtenha um grupo de segredos em um lote usando o AWS CLI

Permissões obrigatórias:

- `secretsmanager:BatchGetSecretValue`
- Permissão `secretsmanager:GetSecretValue` para cada segredo que deseja recuperar.
- Se você usa filtros, também deve ter `secretsmanager:ListSecrets`.

Para um exemplo de política de permissões, consulte the section called “Exemplo: permissão para recuperar um grupo de valores de segredos em um lote”.

> **Important**
>
> Se você tiver uma política de VPCE que nega permissão para recuperar um segredo individual no grupo que você está recuperando, `BatchGetSecretValue` não retornará nenhum valor de segredo e retornará um erro.

Example Recuperar o valor do segredo de um grupo de segredos listados por nome

O exemplo de `batch-get-secret-value` a seguir obtém o valor do segredo para três segredos.

```
aws secretsmanager batch-get-secret-value \
          --secret-id-list MySecret1 MySecret2 MySecret3
```

Example Recuperar o valor do segredo de um grupo de segredos selecionados pelo filtro

O `batch-get-secret-value` de exemplo a seguir obtém o valor dos segredos que têm uma tag chamada “Test”.

```
aws secretsmanager batch-get-secret-value \
          --filters Key="tag-key",Values="Test"
```

Obtenha um valor secreto usando o AWS console

Para recuperar um segredo (console)

1. Abra o console do Secrets Manager em https://console.aws.amazon.com/secretsmanager/.
2. Na lista de segredos, escolha o segredo que você deseja recuperar.
3. Na seção Valor secreto, selecione Recuperar valor secreto.

   Secrets Manager exibe a versão atual (`AWSCURRENT`) do segredo. Para ver outras versões do segredo, como `AWSPREVIOUS` versões com rótulos personalizados, use o the section called "AWS CLI".

Use AWS Secrets Manager segredos em AWS Batch

AWS Batch ajuda você a executar cargas de trabalho de computação em lote no Nuvem AWS. Com AWS Batch, você pode injetar dados confidenciais em seus trabalhos armazenando seus dados confidenciais em AWS Secrets Manager segredos e, em seguida, referenciando-os na definição de seu trabalho. Para mais informações, consulte Especificar dados sigilosos usando segredos do Secrets Manager.

Obtenha um AWS Secrets Manager segredo em um CloudFormation recurso

Com CloudFormation isso, você pode recuperar um segredo para usar em outro CloudFormation recurso. Um cenário comum é primeiro criar um segredo com uma senha gerada pelo Secrets Manager e, em seguida, recuperar o nome de usuário e a senha do segredo para usar como credenciais para um novo banco de dados. Para obter informações sobre como criar segredos com CloudFormation, consulteCloudFormation.

Para recuperar um segredo em um CloudFormation modelo, você usa uma referência dinâmica. Quando você cria a pilha, a referência dinâmica extrai o valor secreto para o CloudFormation recurso, para que você não precise codificar as informações secretas. Em vez disso, é necessário fazer referência ao segredo pelo nome ou ARN. É possível usar uma referência dinâmica para um segredo em qualquer propriedade do recurso. Você não pode usar uma referência dinâmica para um segredo em metadados do recurso, por exemplo, `AWS::CloudFormation::Init`, pois isso tornaria o valor do segredo visível no console.

Uma referência dinâmica para um segredo tem o seguinte padrão:

```
{{resolve:secretsmanager:secret-id:SecretString:json-key:version-stage:version-id}}
```

secret-id

O nome ou ARN completo do segredo. Para acessar um segredo em sua AWS conta, você pode usar o nome secreto. Para acessar um segredo em uma AWS conta diferente, use o ARN do segredo.

json-key (opcional)

O nome da chave do par de chave-valor cujo valor você deseja recuperar. Se você não especificar um`json-key`, CloudFormation recuperará todo o texto secreto. Esse segmento não pode incluir o caractere de dois pontos (`:`).

version-stage (opcional)

A versão do segredo a ser usado. O Secrets Manager usa rótulos de preparação para acompanhar diferentes versões durante o processo de alternância. Se você usar `version-stage`, não especifique `version-id`. Se você não especificar `version-stage` ou `version-id`, o padrão é a versão `AWSCURRENT`. Esse segmento não pode incluir o caractere de dois pontos (`:`).

version-id (opcional)

O identificador exclusivo da versão do segredo a usar. Se você especificar `version-id`, não especifique `version-stage`. Se você não especificar `version-stage` ou `version-id`, o padrão é a versão `AWSCURRENT`. Esse segmento não pode incluir o caractere de dois pontos (`:`).

Para obter mais informações, consulte Uso de referências dinâmicas para especificar segredos do Secrets Manager.

> **Note**
>
> Não crie uma referência dinâmica usando uma barra invertida (`\`) como valor final. CloudFormation não consigo resolver essas referências, o que causa uma falha no recurso.

Use AWS Secrets Manager segredos em GitHub empregos

Para usar um segredo em um GitHub trabalho, você pode usar uma GitHub ação para recuperar segredos AWS Secrets Manager e adicioná-los como variáveis de ambiente mascaradas em seu GitHub fluxo de trabalho. Para obter mais informações sobre GitHub ações, consulte Entendendo GitHub ações nos GitHub documentos.

Quando você adiciona um segredo ao seu GitHub ambiente, ele fica disponível para todas as outras etapas do seu GitHub trabalho. Siga as orientações em Fortalecimento de segurança para GitHub ações para ajudar a evitar que segredos em seu ambiente sejam usados indevidamente.

É possível definir a cadeia de caracteres inteira no valor do segredo como o valor da variável de ambiente ou, caso a cadeia de caracteres seja JSON, pode analisar o JSON a fim de definir variáveis de ambiente individuais para cada par chave-valor JSON. Se o valor do segredo for um binário, a ação o converterá em uma cadeia de caracteres.

Para visualizar as variáveis de ambiente criadas utilizando seus segredos, ative o log de depuração. Para obter mais informações, consulte Habilitando o registro de depuração no GitHub Docs.

Para usar as variáveis de ambiente criadas a partir de seus segredos, consulte Variáveis de ambiente na GitHub documentação.

Pré-requisitos

Para usar essa ação, primeiro você precisa configurar AWS as credenciais e defini-las Região da AWS em seu GitHub ambiente usando a `configure-aws-credentials` etapa. Siga as instruções em Configurar ações de AWS credenciais para que GitHub as ações assumam a função diretamente usando o provedor GitHub OIDC. Isso permite que você use credenciais de curta duração e evite armazenar chaves de acesso adicionais externas ao Secrets Manager.

O perfil do IAM que a ação assume deve ter as seguintes permissões:

- `GetSecretValue` sobre os segredos que você deseja recuperar.
- `ListSecrets` em todos os segredos.
- (Opcional) `Decrypt` KMS key se os segredos estiverem criptografados com um chave gerenciada pelo cliente.

Para obter mais informações, consulte the section called “Autenticação e controle de acesso”.

Usage

Para usar a ação, adicione uma etapa ao fluxo de trabalho que use a sintaxe a seguir.

```
- 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
```

Parâmetros

`secret-ids`

ARNS, nomes e prefixos de nomes dos segredos.

Para definir o nome da variável de ambiente, insira-o antes do ID do segredo e acrescente uma vírgula. Por exemplo, o `ENV_VAR_1, secretId` cria uma variável de ambiente chamada ENV_VAR_1 utilizando o `secretId` do segredo. O nome da variável de ambiente pode consistir em letras maiúsculas, números e sublinhados.

Para usar um prefixo, digite pelo menos três caracteres seguidos de um asterisco. Por exemplo, `dev*` corresponde a todos os segredos com um nome que começa em dev. O número máximo de segredos correspondentes que podem ser recuperados é 100. Se você definir o nome da variável e o prefixo corresponder a diversos segredos, a ação falhará.

`name-transformation`

Por padrão, a etapa cria cada nome de variável de ambiente utilizando o nome do segredo, o qual é transformado para incluir apenas letras maiúsculas, números e sublinhados, bem como não começar com um número. Para as letras no nome, é possível configurar a etapa para usar letras minúsculas com `lowercase` ou não alterar as maiúsculas e minúsculas das letras com `none`. O valor padrão é `uppercase`.

`parse-json-secrets`

(Opcional) Por padrão, a ação define o valor da variável de ambiente para a cadeia de caracteres JSON inteira no valor do segredo. Defina `parse-json-secrets` como `true` para criar variáveis de ambiente para cada par chave-valor no JSON.

Observe que, se o JSON usar chaves que diferenciam maiúsculas de minúsculas, como “nome” e “Nome”, a ação apresentará conflitos de nomes duplicados. Nesse caso, defina os `parse-json-secrets` como `false` e analise o valor do segredo JSON separadamente.

Nomeação de variáveis de ambiente

As variáveis de ambiente criadas pela ação são nomeadas da mesma forma que os segredos dos quais elas vêm. As variáveis de ambiente têm requisitos de nomenclatura mais rígidos do que os segredos, portanto, a ação transforma os nomes de segredos para atender a esses requisitos. Por exemplo, a ação transforma letras minúsculas em letras maiúsculas. Se você analisar o JSON do segredo, o nome da variável de ambiente incluirá tanto o nome do segredo quanto o nome da chave JSON, por exemplo, `MYSECRET_KEYNAME`. É possível configurar a ação para não transformar letras minúsculas.

Se duas variáveis de ambiente terminarem com o mesmo nome, a ação falhará. Nesse caso, você deverá especificar os nomes que deseja usar para as variáveis de ambiente como aliases.

Exemplos de quando os nomes podem entrar em conflito:

- Um segredo chamado "MySecret" e um segredo chamado “mysecret” se tornariam variáveis de ambiente chamadas “MYSECRET”.
- Um segredo chamado "secret_keyname" e um segredo analisado em JSON chamado "Secret" com uma chave chamada "keyname" se tornariam variáveis de ambiente chamadas "SECRET_KEYNAME".

É possível definir o nome da variável de ambiente especificando um alias, conforme mostrado no exemplo a seguir, que cria uma variável chamada `ENV_VAR_NAME`.

```
secret-ids: |
  ENV_VAR_NAME, secretId2
```

Aliases em branco

- Se você definir `parse-json-secrets: true` e inserir um alias em branco, seguido por uma vírgula e depois pelo ID do segredo, a ação nomeará a variável de ambiente da mesma forma que as chaves JSON analisadas. Os nomes das variáveis não incluem o nome do segredo.

Se o segredo não contiver um JSON válido, a ação criará uma variável de ambiente e a nomeará da mesma forma que o nome do segredo.

- Se você definir `parse-json-secrets: false` e inserir um alias em branco, seguido por uma vírgula e depois pelo ID do segredo, a ação nomeará as variáveis de ambiente como se você não especificasse um alias.

Veja a seguir um exemplo de um alias em branco.

```
,secret2
```

Exemplos

Example 1. Obtenha segredos por nome e por ARN

O exemplo a seguir cria variáveis de ambiente para segredos identificados por nome e por 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
```

Variáveis de ambiente criadas:

```
EXAMPLESECRETNAME: secretValue1
TEST1: secretValue2
_0_TEST_SECRET: secretValue3
_PROD_EXAMPLE_SECRET: secretValue4
SECRET_ALIAS_1: secretValue5
SECRET_ALIAS_2: secretValue6
SECRET2: secretValue7
```

Example 2. Obtenha todos os segredos que começam com determinado prefixo

O exemplo a seguir cria variáveis de ambiente para todos os segredos com nomes que começam com*beta*.

```
- name: Get Secret Names by Prefix
  uses: 2
  with:
    secret-ids: |
      beta*    # Retrieves all secrets that start with 'beta'
```

Variáveis de ambiente criadas:

```
BETASECRETNAME: secretValue1
BETATEST: secretValue2
BETA_NEWSECRET: secretValue3
```

Example 3. Análise do JSON no segredo

O exemplo a seguir cria variáveis de ambiente ao analisar o JSON no segredo.

```
- 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
```

O segredo `test/secret` tem o seguinte valor de segredo.

```
{
  "api_user": "user",
  "api_key": "key",
  "config": {
    "active": "true"
  }
}
```

O segredo `secret2` tem o seguinte valor de segredo.

```
{
  "myusername": "alejandro_rosalez",
  "mypassword": "EXAMPLE_PASSWORD"
}
```

Variáveis de ambiente criadas:

```
TEST_SECRET_API_USER: "user"
TEST_SECRET_API_KEY: "key"
TEST_SECRET_CONFIG_ACTIVE: "true"
MYUSERNAME: "alejandro_rosalez"
MYPASSWORD: "EXAMPLE_PASSWORD"
```

Example 4 Use letras minúsculas para nomes de variáveis de ambiente

O exemplo a seguir cria uma variável de ambiente com um nome em minúsculas.

```
- name: Get secrets
  uses: aws-actions/aws-secretsmanager-get-secrets@v2
  with:
    secret-ids: exampleSecretName
    name-transformation: lowercase
```

Variável de ambiente criada:

```
examplesecretname: secretValue
```

Use AWS Secrets Manager em GitLab

AWS Secrets Manager se integra com GitLab. Você pode aproveitar os segredos do Secrets Manager para proteger suas GitLab credenciais para que elas não sejam mais codificadas. GitLab Em vez disso, o GitLab Runner recupera esses segredos do Secrets Manager quando seu aplicativo executa um trabalho nos pipelines de GitLab CI/CD.

Para usar essa integração, você criará um provedor de identidade OpenID Connect (OIDC) no IAM AWS Identity and Access Management e uma função do IAM. Isso permite que o GitLab Runner acesse seu segredo do Secrets Manager. Para obter mais informações sobre GitLab CI/CD e OIDC, consulte a documentação. GitLab

Considerações

Se você estiver usando uma GitLab instância não pública, não poderá usar essa integração com o Secrets Manager. Em vez disso, consulte a GitLab documentação de instâncias não públicas.

Pré-requisitos

Para integrar o Secrets Manager com GitLab, preencha os seguintes pré-requisitos:

1. Crie um AWS Secrets Manager segredo

   Você precisará de um segredo do Secrets Manager que será recuperado em seu GitLab trabalho e eliminará a necessidade de codificar essas credenciais. Você precisará do ID secreto do Secrets Manager ao configurar seu GitLab pipeline. Consulte Crie um AWS Secrets Manager segredo para obter mais informações.
2. Crie GitLab seu provedor OIDC no console do IAM.

   Nesta etapa, você criará GitLab seu provedor OIDC no console do IAM. Para obter mais informações, consulte Criar um provedor de identidade e documentação do OpenID Connect (OIDC). GitLab

   Ao criar o provedor OIDC no console do IAM, use as configurações a seguir:

   a. Defina o `provider URL` para sua GitLab instância. Por exemplo, .**`gitlab.example.com`**

   b. Configure `audience` ou `aud` como **`sts.amazonaws.com`**.
3. Criar uma política e um perfil do IAM

   Será necessário criar uma política e um perfil do IAM. Essa função é assumida por GitLab with AWS Security Token Service (STS). Para obter mais informações, consulte Criação de um perfil usando políticas de confiança personalizadas.

   a. No console do IAM, use as configurações a seguir ao criar o perfil do IAM:

   - Defina `Trusted entity type` como **`Web identity`**.
   - Defina `Group` como **`your GitLab group`**.
   - `Identity provider`Defina com o mesmo URL do provedor (a GitLab instância) que você usou na etapa 2.

- Defina `Audience` como o mesmo público que você usou na etapa 2.

b. Veja a seguir um exemplo de uma política de confiança que GitLab permite assumir funções. Sua política de confiança deve listar seu Conta da AWS GitLab URL e o caminho do projeto.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:oidc-
provider/gitlab.example.com"
      },
      "Condition": {
        "StringEquals": {
          "gitlab.example.com:aud": [
            "sts.amazon.com"
          ]
        },
        "StringLike": {
          "gitlab.example.com:sub": [
            "project_path:mygroup/project-*:ref_type:branch-*:ref:main*"
          ]
        }
      }
    }
  ]
}
```

c. Você também precisará criar uma política do IAM para permitir o GitLab acesso AWS Secrets Manager a. É possível adicionar essa política à sua política de confiança. Para obter mais informações, consulte Criação de políticas de IAM.

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

```
      "Action": "secretsmanager:GetSecretValue",
      "Resource": "arn:aws:secretsmanager:us-
east-1:111122223333:secret:your-secret"
    }
  ]
}
```

Integrando com AWS Secrets Manager GitLab

Depois de concluir os pré-requisitos, você pode configurar GitLab para usar o Secrets Manager para proteger suas credenciais.

Configurar o GitLab pipeline para usar o Secrets Manager

Você precisará atualizar seu arquivo de configuração de GitLab CI/CD com as seguintes informações:

- O público do token definido como STS.
- O ID do segredo do Secrets Manager.
- A função do IAM que você deseja que o GitLab Runner assuma ao executar trabalhos no GitLab pipeline.
- O Região da AWS local onde o segredo está armazenado.

GitLab busca o segredo do Secrets Manager e armazena o valor em um arquivo temporário. O caminho para esse arquivo é armazenado em uma CI/CD variável, semelhante às variáveis CI/CD do tipo de arquivo.

Veja a seguir um trecho do arquivo YAML para um arquivo de configuração GitLab CI/CD:

```
variables:
  AWS_REGION: us-east-1
  AWS_ROLE_ARN: 'arn:aws:iam::111122223333:role/gitlab-role'
job:
  id_tokens:
    AWS_ID_TOKEN:
      aud: 'sts.amazonaws.com'
  secrets:
    DATABASE_PASSWORD:
      aws_secrets_manager:
```

```
        secret_id: "arn:aws:secretsmanager:us-east-1:111122223333:secret:secret-name"
```

Para obter mais informações, consulte a documentação de integração do GitLab Secrets Manager.

Opcionalmente, você pode testar sua configuração do OIDC em. GitLab Consulte a GitLab documentação para testar a configuração do OIDC para obter mais informações.

Solução de problemas

O seguinte pode ajudá-lo a solucionar problemas comuns que você pode encontrar ao integrar o Secrets Manager com o. GitLab

GitLab Problemas de tubulação

Se você tiver problemas no GitLab pipeline, verifique o seguinte:

- Seu arquivo YAML está formatado corretamente. Para obter mais informações, consulte a documentação da GitLab.
- Seu GitLab funil está assumindo a função correta, tem as permissões apropriadas e acesso ao AWS Secrets Manager segredo correto.

Recursos adicionais do

Os recursos a seguir podem ajudá-lo a solucionar problemas com GitLab e AWS Secrets Manager:

- GitLab Solução de problemas do OIDC
- GitLab Depurando o pipeline de CI/CD
- Solução de problemas

Use AWS Secrets Manager segredos em AWS IoT Greengrass

AWS IoT Greengrass é um software que estende os recursos da nuvem para dispositivos locais. Isso permite que os dispositivos coletem e analisem dados mais próximos da fonte de informações, reajam de maneira autônoma a eventos locais e se comuniquem com segurança uns com os outros em redes locais.

AWS IoT Greengrass permite que você se autentique com serviços e aplicativos de AWS IoT Greengrass dispositivos sem codificar senhas, tokens ou outros segredos. Você pode usar AWS

Secrets Manager para armazenar e gerenciar com segurança seus segredos na nuvem. AWS IoT Greengrass estende o Secrets Manager aos dispositivos AWS IoT Greengrass principais, para que seus conectores e funções Lambda possam usar segredos locais para interagir com serviços e aplicativos.

Para integrar um segredo em um AWS IoT Greengrass grupo, você cria um recurso de grupo que faz referência ao segredo do Secrets Manager. Esse recurso de segredo faz referência ao segredo da nuvem usando o ARN associado. Para saber como criar, gerenciar e usar recursos secretos, consulte Como trabalhar com recursos secretos no Guia do AWS IoT desenvolvedor.

Para implantar segredos no AWS IoT Greengrass núcleo, consulte Implantar segredos no AWS IoT Greengrass núcleo.

Use AWS Secrets Manager segredos no Parameter Store

AWS O Systems Manager Parameter Store fornece armazenamento seguro e hierárquico para gerenciamento de dados de configuração e gerenciamento de segredos. Você pode armazenar dados, como senhas, strings de banco de dados e códigos de licença como valores de parâmetro. No entanto, o repositório de parâmetros não fornece serviços de rotação automática para os segredos armazenados. Em vez disso, o Parameter Store permite que você armazene o segredo no Secrets Manager e faça referência ao segredo como um parâmetro do Parameter Store.

Quando você configura o Parameter Store com o Secrets Manager, o Parameter Store do `secret-id` requer uma barra (/) antes da string do nome.

Para obter mais informações, consulte Como referenciar AWS Secrets Manager segredos dos parâmetros do Parameter Store no Guia do AWS Systems Manager usuário.

Gire segredos AWS Secrets Manager

Alternância é o processo de atualizar periodicamente um segredo. Quando o Secrets Manager alterna um segredo, ele atualiza as credenciais tanto no segredo quanto no banco de dados ou serviço. No Secrets Manager, você pode configurar alternância automática para seus segredos. Existem duas formas de alternância:

- Alternância gerenciada: para a maioria dos segredos gerenciados, você usa a alternância gerenciada, na qual o serviço configura e gerencia a alternância para você. A alternância gerenciada não usa uma função do Lambda.
- O Rotate Secrets Manager gerenciou segredos externos— Para segredos mantidos por parceiros do Secrets Manager, você usa a rotação gerenciada de segredos externos para atualizar o segredo no sistema do parceiro. Isso não requer uma função Lambda.
- the section called “Função do Lambda de alternância”: para outros tipos de segredos, a alternância do Secrets Manager usa uma função do Lambda para atualizar o segredo e o banco de dados ou serviço.

Rotação gerenciada para AWS Secrets Manager segredos

Alguns serviços oferecem alternância gerenciada, na qual o serviço configura e gerencia a alternância para você. Com a rotação gerenciada, você não usa uma AWS Lambda função para atualizar o segredo e as credenciais no banco de dados.

Os seguintes serviços oferecem alternância gerenciada:

- O Amazon Aurora oferece alternância gerenciada para credenciais de usuário primário. Para obter mais informações, consulte Gerenciamento de senhas com Amazon Aurora e o AWS Secrets Manager no Guia do usuário do Amazon Aurora.
- O Amazon ECS Service Connect oferece alternância gerenciada para certificados TLS do Autoridade de Certificação Privada da AWS . Para obter mais informações, consulte TLS com o Service Connect no Guia do desenvolvedor do serviço Amazon Elastic Container.
- O Amazon RDS oferece alternância gerenciada para credenciais de usuário primário. Para obter mais informações, consulte Gerenciamento de senhas com Amazon RDS e o AWS Secrets Manager no Guia do usuário do Amazon RDS.

- O Amazon DocumentDB oferece rotação gerenciada para credenciais de usuário mestre. Para obter mais informações, consulte Gerenciamento de senhas com o Amazon DocumentDB e AWS Secrets Manager no Guia do usuário do Amazon DocumentDB.
- O Amazon Redshift oferece alternância gerenciada para senhas de administradores. Para obter mais informações, consulte Gerenciamento de senhas de administrador do Amazon Redshift usando o Guia de gerenciamento AWS Secrets Manager de clusters do Amazon Redshift.
- segredos externos gerenciados oferecem rotação gerenciada para segredos mantidos por parceiros do Secrets Manager. Para obter mais informações, consulte Usando segredos externos AWS Secrets Manager gerenciados para gerenciar segredos de terceiros.

> **Tip**
>
> Para todos os outros tipos de segredos, consulte the section called “Função do Lambda de alternância”.

A rotação de segredos gerenciados normalmente é concluída em um minuto. Durante a rotação, novas conexões que recuperam o segredo podem obter a versão anterior das credenciais. Em aplicações, é altamente recomendado que você siga a prática recomendada de utilizar um usuário de banco de dados criado com os privilégios mínimos necessários para sua aplicação, em vez de usar o usuário principal. Para usuários de aplicativos, para maior disponibilidade, você pode usar a Estratégia de rotação de usuários alternados.

Para segredos mantidos por parceiros do Secrets Manager,

Para alterar a programação de alternância gerenciada

1. Abra o segredo gerenciado no console do Secrets Manager. Você pode seguir um link do serviço de gerenciamento ou pesquisar o segredo no console do Secrets Manager.
2. Em Rotation schedule (Programação da alternância), insira sua programação no fuso horário UTC no Schedule expression builder (Desenvolvedor de expressão programada) ou como uma Schedule expression (Expressão programada). O Secrets Manager armazena sua programação como uma expressão `rate()` ou `cron()`. A janela de alternância começa automaticamente à 0h, a menos que você especifique um horário de início. É possível alternar um segredo com intervalos a partir de quatro horas. Para obter mais informações, consulte Programação de alternância.

3. (Opcional) Em Window duration (Duração da janela), escolha a duração da janela em que deseja que o Secrets Manager alterne o seu segredo, por exemplo, **3h**, por uma janela de três horas. A janela não pode se estender até a próxima janela de alternância. Se você não especificar Window duration (Duração da janela) para uma programação de alternância em horas, a janela será automaticamente encerrada após uma hora. Para uma programação de alternância em dias, a janela terminará automaticamente no final do dia.
4. Escolha Salvar.

Para alterar o cronograma de alternância gerenciada (AWS CLI)

- Chame `rotate-secret`. O exemplo a seguir alterna o segredo entre 16h e 18h UTC no 1.º e no 15.º dias do mês. Para obter mais informações, consulte Programação de alternância.

```
aws secretsmanager rotate-secret \
    --secret-id MySecret \
    --rotation-rules \
        "{\"ScheduleExpression\": \"cron(0 16 1,15 * ? *)\", \"Duration\": \"2h\"}"
```

O Rotate Secrets Manager gerenciou segredos externos

O Secrets Manager fez parceria com fornecedores de software selecionados para oferecer segredos externos gerenciados. Esse recurso ajuda os clientes a gerenciar o ciclo de vida secreto ao lidar com as rotações automaticamente. Com segredos externos gerenciados, os clientes não precisam mais manter uma lógica de rotação específica para cada segredo armazenado com parceiros diferentes. Isso será tratado pelo Secrets Manager.

Para ver a lista de parceiros integrados ao Secrets Manager, consulte Parceiros de segredos externos gerenciados.

Configurar a rotação no console

Para configurar a rotação de um segredo externo gerenciado existente, criado especificando o tipo e o valor do segredo conforme especificado pelos respectivos parceiros de integração, use as seguintes etapas:

1. Abra o console do Secrets Manager.
2. Selecione seu segredo externo gerenciado na lista.

3. Escolha a guia Configuração.
4. Na seção Configuração de rotação, escolha Editar rotação.
5. Ative a Automatic rotation (Alternância automática).
6. Em Metadados de rotação, adicione todos os metadados específicos do parceiro necessários para a rotação:

   Siga as diretrizes fornecidas pelo seu parceiro de integração para outros metadados necessários
7. Em Permissões de serviço para rotação secreta, selecione ou crie uma função do IAM para rotação:
   - Escolha Criar uma nova função para criar automaticamente uma função com as permissões necessárias
   - Ou selecione uma função existente com as permissões apropriadas para seu parceiro

   Por padrão, as permissões são atribuídas ao parceiro individual na região em que o segredo foi criado.
8. Defina seu cronograma de rotação (por exemplo, gire automaticamente a cada 30 dias).
9. Escolha Salvar para aplicar a configuração de rotação.

Os dois principais campos de metadados configurados durante esse processo são:

Campo	Description
ExternalSecretRotationMetadata	Metadados específicos do parceiro necessários para rotação, como a versão da API para Salesforce
ExternalSecretRotationRoleArn	O ARN da função do IAM usada para rotação, com permissões definidas para o parceiro de integração

Para obter mais informações sobre esses campos, consulte Usando segredos externos gerenciados pelo Secrets Manager para gerenciar segredos de terceiros.

Configurar a rotação usando a CLI

Execute o comando a seguir para configurar a rotação de um segredo do Salesforce. Esse comando especifica o ID secreto, o ARN da função do IAM para rotação, o cronograma de rotação e quaisquer metadados específicos do parceiro necessários para o processo de rotação.

```
aws secretsmanager rotate-secret \
            --secret-id SampleSecret \
            --external-secret-rotation-role-arn arn:aws:iam::123412341234:role/xyz \
            --rotation-rules AutomaticallyAfterDays=1 \
            --external-secret-rotation-metadata
 '[{"Key":"apiVersion","Value":"v65.0"}]'
```

Função do Lambda de alternância

Para muitos tipos de segredos, o Secrets Manager usa uma AWS Lambda função para atualizar o segredo e o banco de dados ou serviço. Para obter mais informações sobre os custos do uso de uma função do Lambda, consulte Preços.

Para alguns Segredos gerenciados por outros serviços, você usa alternância gerenciada. Para usar Alternância gerenciada, primeiro você cria o segredo por meio do serviço de gerenciamento.

Durante a alternância, o Secrets Manager registra eventos de logs que indicam o estado de alternância. Para obter mais informações, consulte the section called “Faça login com AWS CloudTrail ”.

Para alternar um segredo, o Secrets Manager chama uma função do Lambda de acordo com a programação de alternância que você configurou. Se você também atualizar manualmente seu valor secreto enquanto a alternância automática estiver configurada, o Secrets Manager considerará essa alternância válida ao calcular a próxima data de alternância.

Durante a alternância, o Secrets Manager chama a mesma função várias vezes, cada vez com parâmetros diferentes. O Secrets Manager invoca a função com a seguinte estrutura de parâmetros de solicitação JSON:

```
{
    "Step" : "request.type",
    "SecretId" : "string",
    "ClientRequestToken" : "string",
    "RotationToken" : "string"
}
```

Parâmetros:

- Etapa: a etapa de alternância: `create_secret`, `set_secret`, `test_secret` ou `finish_secret`. Para obter mais informações, consulte the section called “Quatro etapas em uma função de alternância”.
- SecretId— O ARN do segredo para rotacionar.
- ClientRequestToken— Um identificador exclusivo para a nova versão do segredo. Esse valor ajuda a garantir a idempotência. Para obter mais informações, consulte PutSecretValue: ClientRequestToken na Referência AWS Secrets Manager da API.
- RotationToken— Um identificador exclusivo que indica a origem da solicitação. Necessário para alternância de segredos usando um perfil assumido ou alternância entre contas, na qual você alterna um segredo em uma conta usando uma função de alternância do Lambda em outra conta. Em ambos os casos, a função de alternância assume um perfil do IAM para chamar o Secrets Manager e, em seguida, o Secrets Manager usa o token de alternância para validar a identidade do perfil do IAM.

Se alguma etapa de alternância falhar, o Secrets Manager tentará novamente todo o processo de alternância várias vezes.

Tópicos

- Configure a alternância automática para segredos do Amazon RDS, do Amazon Aurora, do Amazon Redshift ou do Amazon DocumentDB
- Configure a rotação automática para segredos que não sejam do banco de dados AWS Secrets Manager
- Configure a rotação automática usando o AWS CLI
- Estratégias de alternância da função do Lambda
- Função de alternância do Lambda
- AWS Secrets Manager modelos de função de rotação
- Permissões da função de execução da função de rotação Lambda para AWS Secrets Manager
- Acesso à rede para função AWS Lambda de rotação
- Solucionar problemas de rotação AWS Secrets Manager

Configure a alternância automática para segredos do Amazon RDS, do Amazon Aurora, do Amazon Redshift ou do Amazon DocumentDB

Este tutorial descreve como configurar the section called “Função do Lambda de alternância” para segredos de banco de dados. Alternância é o processo de atualizar periodicamente um segredo. Quando o Secrets Manager alterna um segredo, você atualiza as credenciais tanto no segredo como no banco de dados. No Secrets Manager, é possível configurar a alternância automática para seus segredos de banco de dados.

Para configurar a alternância usando o console, você precisa primeiro escolher uma estratégia de alternância. Em seguida, você configura o segredo para a alternância, o qual cria uma função de alternância do Lambda, caso você ainda não tenha uma. O console também define permissões para a função de execução da função do Lambda. A última etapa é garantir que a função de alternância do Lambda possa acessar o Secrets Manager e seu banco de dados utilizando a rede.

> **Warning**
>
> Para ativar a alternância automática, é necessário ter permissão para criar um perfil de execução do IAM para a função de alternância do Lambda e anexar uma política de permissão a ela. Ambas as permissões `iam:CreateRole` e `iam:AttachRolePolicy` são necessárias. Conceder essas permissões permite que uma identidade conceda a ela mesma quaisquer permissões.

Etapas:

- Etapa 1: escolher uma estratégia de alternância e (opcionalmente) criar um segredo de superusuário
- Etapa 2: configurar a alternância e criar uma função de alternância
- Etapa 3: (Opcional) Defina condições de permissões adicionais na função de alternância
- Etapa 4: configurar acesso à rede para a função de alternância
- Próximas etapas

Etapa 1: escolher uma estratégia de alternância e (opcionalmente) criar um segredo de superusuário

Para obter informações sobre as estratégias oferecidas pelo Secrets Manager, consulte the section called “Estratégias de alternância da função do Lambda”.

Se você escolher a estratégia de usuários alternados, deverá Criar segredos e armazenar nele as credenciais de superusuário do banco de dados. É necessário um segredo com credenciais de superusuário porque a alternância clona o primeiro usuário e a maioria dos usuários não tem essa permissão. Observe que o Amazon RDS Proxy não oferece suporte à estratégia de usuários alternados.

Etapa 2: configurar a alternância e criar uma função de alternância

Para ativar a alternância para um segredo do Amazon RDS, do Amazon DocumentDB ou do Amazon Redshift

1. Abra o console do Secrets Manager em https://console.aws.amazon.com/secretsmanager/.
2. Na página Secrets (Segredos), escolha o segredo.
3. Na página Secret details (Detalhes do segredo), na seção Rotation configuration (Configuração da alternância), escolha Edit rotation (Editar alternância).
4. Na caixa de diálogo Edit rotation configuration (Editar configuração da alternância), siga estas etapas:

    a. Ative a Automatic rotation (Alternância automática).

    b. Em Rotation schedule (Programação da alternância), insira sua programação no fuso horário UTC no Schedule expression builder (Desenvolvedor de expressão programada) ou como uma Schedule expression (Expressão programada). O Secrets Manager armazena sua programação como uma expressão `rate()` ou `cron()`. A janela de alternância começa automaticamente à 0h, a menos que você especifique um horário de início. É possível alternar um segredo com intervalos a partir de quatro horas. Para obter mais informações, consulte Programação de alternância.

    c. (Opcional) Em Window duration (Duração da janela), escolha a duração da janela em que deseja que o Secrets Manager alterne o seu segredo, por exemplo, **3h**, por uma janela de três horas. A janela não pode se estender até a próxima janela de alternância. Se você não especificar Window duration (Duração da janela) para uma programação de alternância em

horas, a janela será automaticamente encerrada após uma hora. Para uma programação de alternância em dias, a janela terminará automaticamente no final do dia.

d. (Opcional) Escolha Rotate immediately when the secret is stored (Alternar imediatamente quando o segredo for armazenado) para alternar o seu segredo assim que as suas alterações forem salvas. Se você desmarcar a caixa de seleção, a primeira alternância começará no cronograma definido.

   Se a alternância falhar, por exemplo, porque as etapas 3 e 4 ainda não foram concluídas, o Secrets Manager repetirá o processo de alternância várias vezes.

e. Em Rotation function (Função de alternância), execute uma das ações a seguir:

   - Selecione Criar uma nova função do Lambda e insira um nome para sua nova função. O Secrets Manager adiciona `SecretsManager` ao início do nome da função. O Secrets Manager cria a função com base no modelo apropriado e define as permissões necessárias para a função de execução do Lambda.
   - Escolha Usar uma função do Lambda existente para reutilizar uma função de alternância usada para outro segredo. As funções de alternância listadas em Recommended VPC configurations (Configurações de VPC recomendadas) têm a mesma VPC e o mesmo grupo de segurança que o banco de dados, o que ajuda a função a acessar o banco de dados.

f. Em Estratégia de alternância, escolha a estratégia de Usuário único ou de Usuários alternados. Para obter mais informações, consulte the section called “Etapa 1: escolher uma estratégia de alternância e (opcionalmente) criar um segredo de superusuário”.

5. Escolha Salvar.

Etapa 3: (Opcional) Defina condições de permissões adicionais na função de alternância

Na política de recursos para sua função de alternância, recomendamos incluir a chave de contexto `aws:SourceAccount` para ajudar a evitar que o Lambda seja usado como um confused deputy. Para alguns AWS serviços, para evitar o cenário confuso de substituto, AWS recomenda que você use as chaves de condição `aws:SourceArn` e as chaves de condição `aws:SourceAccount` globais. No entanto, se você incluir a condição `aws:SourceArn` em sua política de função de alternância, a função de alternância só poderá ser usada para alternar o segredo especificado por esse ARN. Recomendamos que inclua apenas a chave de contexto `aws:SourceAccount`, de modo que possa usar a função de alternância para vários segredos.

Para atualizar sua política de recursos da função de alternância

1. No console do Secrets Manager, escolha seu segredo e, em seguida, na página de detalhes, em Rotation configuration (Configuração de alternância), escolha a função de alternância do Lambda. Abra o console do Lambda.
2. Siga as instruções em Usar políticas baseadas em recursos para o Lambda para adicionar uma condição `aws:sourceAccount`.

```
"Condition": {
    "StringEquals": {
        "AWS:SourceAccount": "123456789012"
    }
},
```

Se o segredo for criptografado com uma chave KMS diferente da Chave gerenciada pela AWS `aws/secretsmanager`, o Secrets Manager concede permissão à função de execução do Lambda para usar a chave. Você pode usar o contexto de criptografia SecretARN para limitar o uso da função de descriptografia, de modo que a função de alternância tenha acesso apenas para descriptografar o segredo que é responsável pela alternância.

Para atualizar o papel de execução da função de alternância

1. Na função de alternância do Lambda, escolha Configuração e, em Função de execução, escolha o Nome da função.
2. Siga as instruções em Modifying a role permissions policy (Modificar uma política de permissões de função) para adicionar uma condição `kms:EncryptionContext:SecretARN`.

```
"Condition": {
    "StringEquals": {
        "kms:EncryptionContext:SecretARN": "SecretARN"
    }
},
```

Etapa 4: configurar acesso à rede para a função de alternância

Para obter mais informações, consulte the section called “Acesso à rede para função AWS Lambda de rotação”.

Próximas etapas

Consulte the section called “Solucionar problemas de alternância”.

Configure a rotação automática para segredos que não sejam do banco de dados AWS Secrets Manager

Este tutorial descreve como configurar the section called “Função do Lambda de alternância” para segredos não de banco de dados. Alternância é o processo de atualizar periodicamente um segredo. Ao alternar um segredo, você atualiza as credenciais no segredo e no banco de dados ou serviço para o qual o segredo se destina.

Para obter informações sobre segredos do banco de dados, consulte Alternância automática para segredos de banco de dados (console).

> **Warning**
>
> Para ativar a alternância automática, é necessário ter permissão para criar um perfil de execução do IAM para a função de alternância do Lambda e anexar uma política de permissão a ela. Ambas as permissões `iam:CreateRole` e `iam:AttachRolePolicy` são necessárias. Conceder essas permissões permite que uma identidade conceda a ela mesma quaisquer permissões.

Etapas:

- Etapa 1: criar uma função de alternância genérica
- Etapa 2: programar o código da função de alternância
- Etapa 3: configurar o segredo para alternância
- Etapa 4: permitir que a função de alternância acesse o Secrets Manager e seu banco de dados ou serviço
- Etapa 5: permitir que o Secrets Manager invoque a função de alternância
- Etapa 6: configurar acesso à rede para a função de alternância
- Próximas etapas

Etapa 1: criar uma função de alternância genérica

Para começar, crie uma função de alternância do Lambda. Ela não terá o código dentro dela para alternar seu segredo, então você escreverá isso em uma etapa posterior. Para obter informações sobre como uma função de alternância funciona, consulte the section called “Função de alternância do Lambda”.

Nas regiões suportadas, você pode usar AWS Serverless Application Repository para criar a função a partir de um modelo. Para obter uma lista de regiões compatíveis, consulte AWS Serverless Application Repository FAQs. Em outras regiões, você cria a função do zero e copia o código do modelo na função.

Para criar uma função de alternância genérica

1. Para determinar se AWS Serverless Application Repository é compatível com sua região, consulte AWS Serverless Application Repository endpoints e cotas na Referência AWS geral.
2. Execute um destes procedimentos:
   - Se AWS Serverless Application Repository houver suporte em sua região:
     a. No console do Lamba, escolha Aplicações e, em seguida, Criar aplicação.
     b. Na página Criar aplicação, escolha a guia Aplicativo com tecnologia sem servidor.
     c. Na caixa de pesquisa, em Aplicações públicas, insira **`SecretsManagerRotationTemplate`**.
     d. Selecione Mostrar aplicações que criam perfis do IAM ou políticas de recursos personalizados.
     e. Selecione o bloco SecretsManagerRotationTemplate.
     f. Na página Analisar, configurar e implantar, no quadro Configurações da aplicação, preencha os campos obrigatórios.
        - Em endpoint, insira o endpoint da sua região, incluindo **`https://`**. Para uma lista de endpoints , consulte the section called “Endpoint do Secrets Manager”.
        - Para colocar a função Lambda em uma VPC, inclua Ids e. vpcSecurityGroup vpcSubnetIds
     g. Escolha Implantar.
   - Se AWS Serverless Application Repository não for compatível com sua região:
     a. No console do Lambda, selecione Funções e, em seguida, Criar função.

b. Na página Create function (Criar função), faça o seguinte:

   i. Escolha Criar do zero.

   ii. Em Function name (Nome da função), insira um nome para sua função de alternância.

   iii. Em Runtime, escolha Python 3.10.

   iv. Escolha a opção Criar função.

Etapa 2: programar o código da função de alternância

Nesta etapa, você escreve o código que atualiza o segredo e o serviço ou banco de dados para o qual o segredo se destina. Para obter informações sobre o que uma função de alternância faz, incluindo dicas sobre como escrever sua própria função de alternância, consulte the section called "Função de alternância do Lambda". É possível utilizar Modelos de função de alternância como referência.

Etapa 3: configurar o segredo para alternância

Nesta etapa, você define uma programação de alternância para seu segredo e conecta a função de alternância para o segredo.

Para configurar a alternância e criar uma função de alternância em branco

1. Abra o console do Secrets Manager em https://console.aws.amazon.com/secretsmanager/.
2. Na página Secrets (Segredos), escolha o segredo.
3. Na página Secret details (Detalhes do segredo), na seção Rotation configuration (Configuração da alternância), escolha Edit rotation (Editar alternância). Na caixa de diálogo Edit rotation configuration (Editar configuração da alternância), siga estas etapas:

   a. Ative a Automatic rotation (Alternância automática).

   b. Em Rotation schedule (Programação da alternância), insira sua programação no fuso horário UTC no Schedule expression builder (Desenvolvedor de expressão programada) ou como uma Schedule expression (Expressão programada). O Secrets Manager armazena sua programação como uma expressão `rate()` ou `cron()`. A janela de alternância começa automaticamente à 0h, a menos que você especifique um horário de início. É possível alternar um segredo com intervalos a partir de quatro horas. Para obter mais informações, consulte Programação de alternância.

c. (Opcional) Em Window duration (Duração da janela), escolha a duração da janela em que deseja que o Secrets Manager alterne o seu segredo, por exemplo, **3h**, por uma janela de três horas. A janela não pode se estender até a próxima janela de alternância. Se você não especificar Window duration (Duração da janela) para uma programação de alternância em horas, a janela será automaticamente encerrada após uma hora. Para uma programação de alternância em dias, a janela terminará automaticamente no final do dia.

d. (Opcional) Escolha Rotate immediately when the secret is stored (Alternar imediatamente quando o segredo for armazenado) para alternar o seu segredo assim que as suas alterações forem salvas. Se você desmarcar a caixa de seleção, a primeira alternância começará no cronograma definido.

e. Em Função de alternância, escolha a função do Lambda criada na etapa 1.

f. Escolha Salvar.

Etapa 4: permitir que a função de alternância acesse o Secrets Manager e seu banco de dados ou serviço

A função de alternância do Lambda necessita de permissão para acessar o segredo no Secrets Manager e precisa de permissão para acessar seu banco de dados ou serviço. Nesta etapa, você concede essas permissões à função de execução do Lambda. Se o segredo for criptografado com uma chave KMS diferente da Chave gerenciada pela AWS `aws/secretsmanager`, será necessário conceder à função de execução do Lambda permissão para usar a chave. É possível usar o contexto de criptografia SecretARN para limitar o uso da função de descriptografia, para que a função de alternância tenha acesso apenas para descriptografar o segredo que é responsável pela alternância. Para obter exemplos de políticas, consulte Permissões para alternância.

Para obter instruções, consulte Função de execução do Lambda no Guia do desenvolvedor do AWS Lambda .

Etapa 5: permitir que o Secrets Manager invoque a função de alternância

Para permitir que o Secrets Manager invoque a função de alternância na programação de alternância que você configurou, você precisa conceder a permissão `lambda:InvokeFunction` à entidade principal do serviço Secrets Manager na política de recursos da função do Lambda.

Na política de recursos para sua função de alternância, recomendamos incluir a chave de contexto `aws:SourceAccount` para ajudar a evitar que o Lambda seja usado como um confused deputy. Para alguns AWS serviços, para evitar o cenário confuso de substituto, AWS

recomenda que você use as chaves de condição `aws:SourceArn` e as chaves de condição `aws:SourceAccount` globais. No entanto, se você incluir a condição `aws:SourceArn` em sua política de função de alternância, a função de alternância só poderá ser usada para alternar o segredo especificado por esse ARN. Recomendamos que inclua apenas a chave de contexto `aws:SourceAccount`, de modo que possa usar a função de alternância para vários segredos.

Para anexar uma política de recursos a uma função do Lambda, consulte Usar políticas baseadas em recursos para o Lambda.

A política a seguir permite que o Secrets Manager invoque uma função do 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": "arn:aws:lambda:us-east-1:123456789012:function:function-
name"
    }
    ]
}
```

Etapa 6: configurar acesso à rede para a função de alternância

Nesta etapa, você permite que a função de alternância se conecte ao Secrets Manager e ao serviço ou banco de dados ao qual o segredo se destina. A função de alternância deve ser capaz de acessar ambos para ser capaz de alternar o segredo. Consulte the section called “Acesso à rede para função AWS Lambda de rotação”.

Próximas etapas

Ao configurar a alternância na Etapa 3, você define uma programação para a alternância do segredo. Se a alternância falhar quando estiver programada, o Secrets Manager tentará a alternância várias vezes. Você também pode iniciar uma alternância imediatamente seguindo as instruções em Alternar um segredo imediatamente.

Se a alternância falhar, consulte Solucionar problemas de alternância.

Configure a rotação automática usando o AWS CLI

Este tutorial descreve como configurar the section called “Função do Lambda de alternância” usando AWS CLI o. Ao alternar um segredo, você atualiza as credenciais no segredo e no banco de dados ou serviço para o qual o segredo se destina.

Você também pode usar o console para configurar a alternância. Para obter informações sobre segredos do banco de dados, consulte Alternância automática para segredos de banco de dados (console). Para todos os outros tipos de segredos, consulte Alternância automática para segredos não de banco de dados (console).

Para configurar a rotação usando o AWS CLI, se você estiver rotacionando um segredo de banco de dados, primeiro você precisa escolher uma estratégia de rotação. Se você escolher a estratégia de usuários alternados, deverá armazenar um segredo separado com credenciais para um superusuário do banco de dados. Em seguida, você programa o código da função de alternância. O Secrets Manager fornece modelos nos quais você pode basear sua função. Em seguida, você cria uma função Lambda com seu código e define permissões para a função Lambda e para a função de execução do Lambda. A próxima etapa é garantir que a função do Lambda possa acessar o Secrets Manager e seu banco de dados ou serviço utilizando a rede. Por fim, você configura o segredo para a alternância.

Etapas:

Pré-requisito para segredos de banco de dados: escolha uma estratégia de alternância

Para obter informações sobre as estratégias oferecidas pelo Secrets Manager, consulte the section called “Estratégias de alternância da função do Lambda”.

Opção 1: estratégia de usuário único

Se você escolher a estratégia de usuário único, poderá continuar com a Etapa 1.

Opção 2: estratégia de usuários alternados

Se você escolher a estratégia de usuários alternados, deverá:

- Criar um segredo e armazenar nele as credenciais de superusuário do banco de dados. Você precisa de um segredo com credenciais de superusuário porque a rotação da alternância de usuários clona o primeiro usuário e a maioria dos usuários não tem essa permissão.
- Adicionar o ARN do segredo do superusuário ao segredo original. Para obter mais informações, consulte the section called “Estrutura JSON de um segredo”.

Observe que o Amazon RDS Proxy não oferece suporte à estratégia de usuários alternados.

Etapa 1: escrever o código da função de alternância

Para alternar um segredo, você precisa de uma função de alternância. Uma função de alternância corresponde a uma função Lambda a qual o Secrets Manager chama para alternar o segredo. Para obter mais informações, consulte the section called “Função do Lambda de alternância”. Nesta etapa, você escreve o código que atualiza o segredo e o serviço ou banco de dados para o qual o segredo se destina.

O Secrets Manager fornece modelos para segredos de banco de dados do Amazon RDS, Amazon Aurora, Amazon Redshift e Amazon DocumentDB em Modelos de função de alternância.

Para escrever o código da função de alternância

1. Execute um destes procedimentos:
   - Verifique a lista de modelos de função de alternância. Se houver um que corresponda à sua estratégia de serviço e alternância, copie o código.
   - Para outros tipos de segredos, você escreve sua própria função de alternância. Para instruções, consulte the section called “Função de alternância do Lambda”.

2. Salve o arquivo em um arquivo ZIP *my-function.zip* junto com todas as dependências necessárias.

Etapa 2: criar a função do Lambda

Nesta etapa, você cria a função do Lambda usando o arquivo ZIP criado na Etapa 1. Você também define o perfil de execução do Lambda, que é o perfil que o Lambda assume quando a função é invocada.

Para criar uma função de alternância e uma função de execução do Lambda

1. Crie uma política de confiança para a função de execução do Lambda e salve-a como um arquivo JSON. Para obter mais informações e exemplos, consulte the section called "Permissões para alternância". A política deve:
   - Permitir que a função chame as operações do Secrets Manager no segredo.
   - Permita que o perfil chame o serviço para o qual o segredo se destina, por exemplo, para criar uma nova senha.
2. Crie o perfil de execução do Lambda e aplique a política de confiança criada na etapa anterior chamando `iam create-role`.

```
aws iam create-role \
    --role-name rotation-lambda-role \
    --assume-role-policy-document file://trust-policy.json
```

3. Crie a função Lambda do arquivo ZIP chamando `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. Defina uma política de recursos na função Lambda para permitir que o Secrets Manager a invoque chamando `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
```

Etapa 3: configurar o acesso à rede

Para obter mais informações, consulte the section called “Acesso à rede para função AWS Lambda de rotação”.

Etapa 4: configurar o segredo para alternância

Para ativar a alternância automática do seu segredo, determine `rotate-secret`. Você pode definir uma programação de alternância com uma expressão de programação `cron()` ou `rate()` e pode definir a duração da janela de alternância. Para obter mais informações, consulte the section called “Programação de alternância”.

```
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\"}"
```

Próximas etapas

Consulte the section called “Solucionar problemas de alternância”.

Estratégias de alternância da função do Lambda

Em the section called “Função do Lambda de alternância”, para segredos de bancos de dados, o Secrets Manager oferece duas estratégias de alternância.

Estratégia de alternância: usuário único

Essa estratégia atualiza as credenciais de um único usuário em um segredo. Para instâncias do Db2 do Amazon RDS, como os usuários não podem alterar suas próprias senhas, é necessário fornecer credenciais de administrador em outro segredo. Essa é a estratégia de alternância mais simples, e é apropriada para a maioria dos casos de uso. Em particular, recomendamos que você use essa estratégia para obter credenciais para usuários únicos (ad hoc) ou interativos.

Quando o segredo é alternado, as conexões de banco de dados abertas não são descartadas. Enquanto a alternância acontece, há um curto período entre a alteração da senha no banco de dados e a atualização do segredo. Durante esse período, há um baixo risco de o banco de dados recusar chamadas que usam as credenciais alternadas. É possível mitigar esse risco com uma estratégia de nova tentativa apropriada. Após a alternância, as novas conexões usam as novas credenciais.

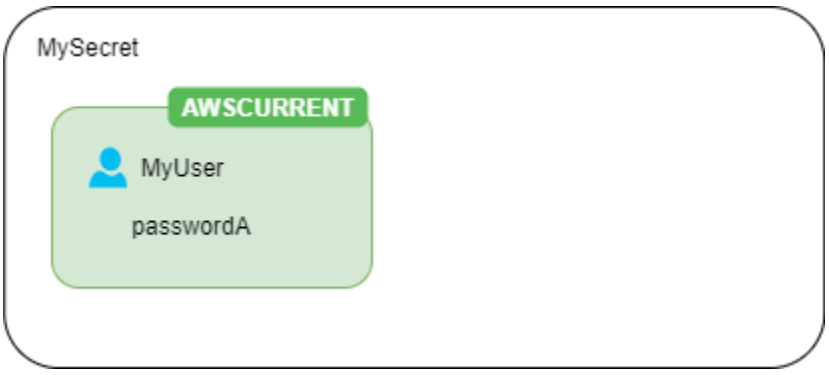
Estratégia de alternância: usuários alternados

Essa estratégia atualiza as credenciais de dois usuários em um segredo. Você cria o primeiro usuário e, durante a primeira alternância, a função de alternância o clona para criar o segundo usuário. Toda vez que o segredo é alternado, a função de alternância alterna a senha do usuário que ela atualiza. Como a maioria dos usuários não tem permissão para se clonar, é necessário fornecer as credenciais para um `superuser` em outro segredo. Recomendamos o uso da estratégia de alternância de usuário único quando os usuários clonados em seu banco de dados não tiverem as mesmas permissões que o usuário original, e para credenciais de usuários únicos (ad hoc) ou interativos.

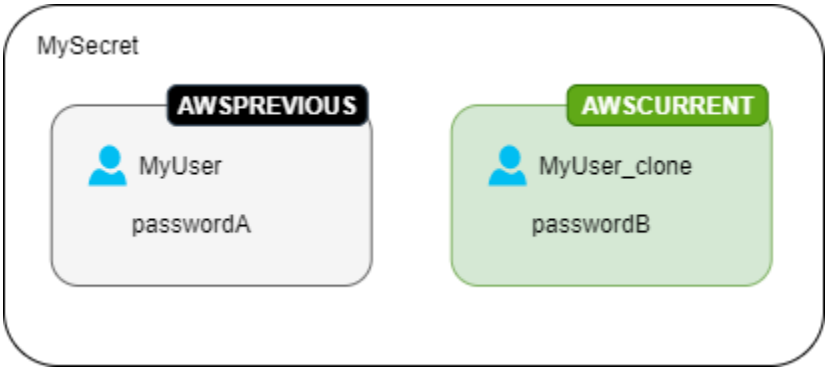
Essa estratégia é apropriada para bancos de dados com modelos de permissão em que uma função possui as tabelas de banco de dados e uma segunda função tem permissão para acessar as tabelas de banco de dados. Ela também é apropriada para aplicações que requerem alta disponibilidade. Se uma aplicação recuperar o segredo durante a alternância, a aplicação ainda obterá um conjunto válido de credenciais. Após a alternância, as credenciais do `user` e do `user_clone` são válidas. Há ainda menos chances de aplicações serem recusadas durante esse tipo de alternância em comparação com a alternância de usuário único. Se o banco de dados estiver hospedado em um farm de servidores em que a alteração de senha demora algum tempo para se propagar para todos os servidores, existe o risco de o banco de dados recusar chamadas que usam as novas credenciais. É possível mitigar esse risco com uma estratégia de nova tentativa apropriada.

O Secrets Manager cria o usuário clonado com as mesmas permissões do usuário original. Se você alterar as permissões do usuário original após a criação do clone, também deverá alterar as permissões do usuário clonado.

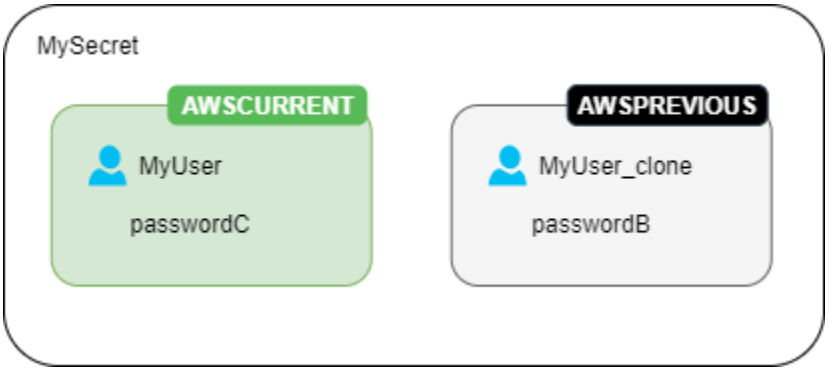
Por exemplo, se você criar um segredo com as credenciais de um usuário do banco de dados, o segredo conterá uma versão com essas credenciais.



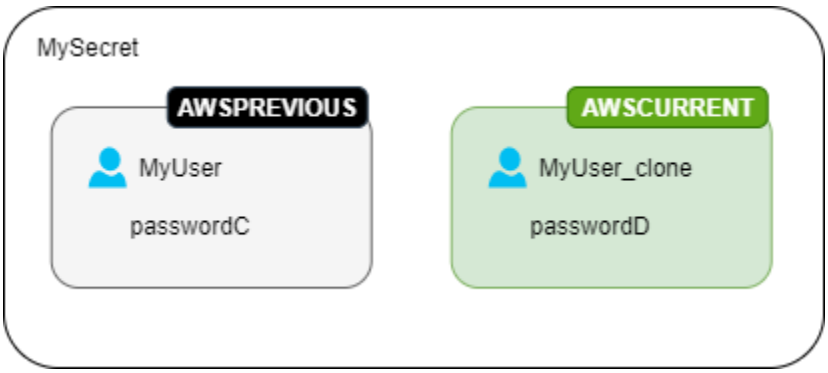
Primeira alternância: a função de alternância cria um clone do seu usuário com uma senha gerada e essas credenciais se tornam a versão do segredo atual.



Segunda alternância: a função de alternância atualiza a senha do usuário original.



Terceira alternância: a função de alternância atualiza a senha do usuário clonado.



Função de alternância do Lambda

Emthe section called “Função do Lambda de alternância”, uma AWS Lambda função gira o segredo. AWS Secrets Manager usa rótulos de teste para identificar versões secretas durante a rotação.

Se AWS Secrets Manager não fornecer um modelo de função de rotação para seu tipo de segredo, você pode criar uma função de rotação personalizada. Siga estas diretrizes ao escrever sua função de alternância:

Práticas recomendadas para funções de alternância personalizadas

- Use o modelo de alternância genérico como ponto de partida.
- Tenha cuidado com as instruções de depuração ou registro em log. Eles podem gravar informações no Amazon CloudWatch Logs. Certifique-se de que os logs não contenham informações confidenciais.

  Para exemplos de instruções de log, consulte o código-fonte dos the section called “Modelos de função de alternância”.
- Por segurança, permite AWS Secrets Manager apenas que uma função de rotação Lambda gire o segredo diretamente. A função de alternância não pode chamar uma outra função do Lambda para alternar o segredo.
- Para orientações sobre depuração, consulte Teste e depuração de aplicações com tecnologia sem servidor.
- Se você usar binários e bibliotecas externas, por exemplo, para se conectar a um recurso, será responsável pela aplicação de patches e atualizações.
- Package sua função de rotação e quaisquer dependências em um arquivo ZIP, como *my-function.zip*.

> **Warning**
>
> Definir o parâmetro de simultaneidade provisionado como um valor menor que 10 pode causar limitação devido à insuficiência de threads de execução para a função do Lambda. Para obter mais informações, consulte Noções básicas sobre simultaneidade reservada e simultaneidade provisionada no Guia do desenvolvedor do AWS Lambda AWS Lambda .

Quatro etapas em uma função de alternância

Tópicos

`createSecret`: criar uma nova versão do segredo

O método `createSecret` primeiro verifica se existe um segredo chamando `get_secret_value` com o `ClientRequestToken` aprovado. Se não houver segredo, ele cria um novo segredo com `create_secret` e o token como `VersionId`. Em seguida, ele gera um novo valor de segredo com `get_random_password`. Depois ele chama `put_secret_value` para armazená-lo com o rótulo de preparação `AWSPENDING`. Armazenar o novo valor do segredo em `AWSPENDING` ajuda a garantir a idempotência. Se a alternância falhar por qualquer motivo, será possível consultar esse valor de segredo em chamadas subsequentes. Consulte Como torno minha função do Lambda idempotente?.

Dicas para escrever sua própria função de alternância

- Certifique-se de que o novo valor do segredo inclua apenas caracteres válidos para o banco de dados ou serviço. Exclua caracteres usando o parâmetro `ExcludeCharacters`.
- Ao testar sua função, use o AWS CLI para ver os estágios da versão: chame `describe-secret` e examine `VersionIdsToStages`.
- Para o Amazon RDS MySQL, o Secrets Manager cria um usuário clonado com um nome de no máximo 16 caracteres no rodízio de usuários em alternância. É possível modificar a função de rodízio para permitir nomes de usuário mais longos. A versão 5.7 e superior do MySQL é compatível com nomes de usuário de até 32 caracteres. No entanto, o Secrets Manager

acrescenta “_clone” (seis caracteres) ao final do nome de usuário, portanto, é necessário manter o nome de usuário com no máximo 26 caracteres.

setSecret: alterar as credenciais no banco de dados ou serviço

O método `setSecret` altera a credencial no banco de dados ou serviço para corresponder ao novo valor do segredo na versão `AWSPENDING` do segredo.

Dicas para escrever sua própria função de alternância

- Se você transmitir instruções para um serviço que interprete instruções, como um banco de dados, use a parametrização de consulta. Para obter mais informações, consulte a Folha de dicas de parametrização de consultas no site da OWASP.
- A função de alternância é um representante privilegiado que tem autorização para acessar e modificar as credenciais do cliente no segredo do Secrets Manager e no recurso de destino. Para evitar um possível ataque “confused deputy”, você precisa garantir que um invasor não possa usar a função para acessar outros recursos. Antes de atualizar a credencial:
  - Verifique se a credencial na versão `AWSCURRENT` do segredo é válida. Se a credencial `AWSCURRENT` não for válida, abandone a tentativa de alternância.
  - Verifique se os valores dos segredos `AWSCURRENT` e `AWSPENDING` são para o mesmo recurso. Para obter um nome de usuário e uma senha, verifique se os nomes de usuário `AWSCURRENT` e `AWSPENDING` são os mesmos.
  - Verifique se o recurso do serviço de destino é o mesmo. Para um banco de dados, verifique se os nomes de host `AWSCURRENT` e `AWSPENDING` são os mesmos.
- Em casos raros, talvez você queira personalizar a função de alternância existente para um banco de dados. Por exemplo, com o rodízio de usuários em alternância, o Secrets Manager cria o usuário clonado copiando os parâmetros de configuração de runtime do primeiro usuário. Se você quiser incluir mais atributos ou alterar quais são concedidos ao usuário clonado, é necessário atualizar o código na função `set_secret`.

testSecret: testar a nova versão do segredo

Em seguida, a função de alternância do Lambda testará a versão `AWSPENDING` do segredo usando-a para acessar o banco de dados ou serviço. As funções de alternância baseadas em Modelos de função de alternância testam o novo segredo usando o acesso de leitura.

finishSecret: concluir a alternância

Por fim, a função de alternância do Lambda move o rótulo `AWSCURRENT` da versão anterior do segredo para a atual, o que também remove o rótulo `AWSPENDING` na mesma chamada de API. O Secrets Manager adiciona o rótulo de preparação prévia `AWSPREVIOUS` na versão anterior, para que você retenha a última versão boa conhecida do segredo.

O método finish_secret usa `update_secret_version_stage` para mover o rótulo de preparação `AWSCURRENT` da versão do segredo anterior para a nova versão do segredo. O Secrets Manager adiciona automaticamente o rótulo de preparação `AWSPREVIOUS` à versão anterior para reter a última versão válida do segredo.

Dicas para escrever sua própria função de alternância

- Não remova `AWSPENDING` antes desse ponto, e não remova-o por meio de uma chamada de API distinta. Isso pode indicar ao Secrets Manager que a alternância não foi concluída com êxito. O Secrets Manager adiciona o rótulo de preparação prévia `AWSPREVIOUS` na versão anterior, para que você retenha a última versão boa conhecida do segredo.

Quando a alternância for bem-sucedida, talvez o rótulo de preparação de `AWSPENDING` seja anexado à mesma versão da versão `AWSCURRENT` ou talvez não seja anexado a nenhuma versão. Se o rótulo de preparação de `AWSPENDING` estiver presente, mas não estiver anexado à mesma versão de `AWSCURRENT`, qualquer invocação posterior de alternância vai pressupor que uma solicitação de alternância anterior ainda está em andamento e retornará um erro. Quando a alternância não for bem-sucedida, o rótulo de preparação de `AWSPENDING` poderá ser anexado a uma versão vazia de segredo. Para obter mais informações, consulte Solucionar problemas de alternância.

AWS Secrets Manager modelos de função de rotação

AWS Secrets Manager fornece um conjunto de modelos de função de rotação que ajudam a automatizar o gerenciamento seguro de credenciais para vários sistemas e serviços de banco de dados. Os modelos são funções ready-to-use Lambda que implementam as melhores práticas para rotação de credenciais, ajudando você a manter sua postura de segurança sem intervenção manual.

Os modelos oferecem suporte a duas estratégias principais de alternância:

- Alternância de usuário único, que atualiza as credenciais de um único usuário.
- Alternância de usuários alternados, que mantém dois usuários separados para ajudar a eliminar o tempo de inatividade durante mudanças de credenciais.

O Secrets Manager também fornece um modelo genérico que serve como ponto de partida para qualquer tipo de segredo.

Para usar os modelos, consulte:

- Alternância automática para segredos de banco de dados (console)
- Alternância automática para segredos não de banco de dados (console)

Para escrever sua própria função de alternância, consulte Escrever uma função de alternância.

Modelos

- Amazon RDS e Amazon Aurora
  - Usuário único do Db2 do Amazon RDS
  - Usuários em alternância do Db2 do Amazon RDS
  - Usuário único do MariaDB do Amazon RDS
  - Usuários alternados do MariaDB do Amazon RDS
  - Usuário único do Amazon RDS e do Amazon Aurora MySQL
  - Usuários em alternância do Amazon RDS e do Amazon Aurora MySQL
  - Usuário único do Oracle do Amazon RDS
  - Usuários alternados do Oracle do Amazon RDS
  - Usuário único do Amazon RDS e do Amazon Aurora PostgreSQL
  - Usuários em alternância do Amazon RDS e do Amazon Aurora PostgreSQL
  - Amazon RDS Microsoft (usuário SQLServer único)
  - Amazon RDS Microsoft (usuários SQLServer alternados)
- Amazon DocumentDB (compatível com MongoDB)
  - Usuário único do Amazon DocumentDB
  - Usuários alternados do Amazon DocumentDB
- banco de dados de origem
  - Usuário único do Amazon Redshift
  - Usuários alternados do Amazon Redshift
- Amazon Timestream para InfluxDB
  - Usuário único do Amazon Timestream para InfluxDB
  - Usuário alternados do Amazon Timestream para InfluxDB

- Amazon ElastiCache
- Active Directory
  - Credenciais do Active Directory
  - Atributos do Active Directory
- Outros tipos de segredo

Amazon RDS e Amazon Aurora

Usuário único do Db2 do Amazon RDS

- Nome do modelo: SecretsManager RDSDb2 RotationSingleUser
- Estratégia de alternância: Estratégia de alternância: usuário único.
- Estrutura da **`SecretString`**: the section called “Credenciais do Amazon RDS e do Aurora”.
- Código fonte: https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerRDSDb2RotationSingleUser/lambda _function.py
- Dependência: python-ibmdb

Usuários em alternância do Db2 do Amazon RDS

- Nome do modelo: SecretsManager RDSDb2 RotationMultiUser
- Estratégia de alternância: the section called “Usuários alternados”.
- Estrutura da **`SecretString`**: the section called “Credenciais do Amazon RDS e do Aurora”.
- Código fonte: https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerRDSDb2RotationMultiUser/lambda _function.py
- Dependência: python-ibmdb

Usuário único do MariaDB do Amazon RDS

- Nome do modelo: SecretsManager RDSMaria DBRotation SingleUser
- Estratégia de alternância: Estratégia de alternância: usuário único.
- Estrutura da **`SecretString`**: the section called “Credenciais do Amazon RDS e do Aurora”.
- Código fonte: https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerRDSMariaDBRotationSingleUser/lambda _function.py

- Dependência: PyMy SQL 1.0.2. Se você usar a senha sha256 para autenticação, PyMy SQL [rsa]. Para obter informações sobre o uso de pacotes com código compilado em um runtime do Lambda, consulte Como faço para adicionar pacotes Python com binários compilados ao meu pacote de implantação e tornar o pacote compatível com o Lambda? no Centro de Conhecimentos da AWS .

Usuários alternados do MariaDB do Amazon RDS

- Nome do modelo: SecretsManager RDSMaria DBRotation MultiUser
- Estratégia de alternância: the section called “Usuários alternados”.
- Estrutura da **`SecretString`**: the section called “Credenciais do Amazon RDS e do Aurora”.
- Código fonte: https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerRDSMariaDBRotationMultiUser/lambda _function.py
- Dependência: PyMy SQL 1.0.2. Se você usar a senha sha256 para autenticação, PyMy SQL [rsa]. Para obter informações sobre o uso de pacotes com código compilado em um runtime do Lambda, consulte Como faço para adicionar pacotes Python com binários compilados ao meu pacote de implantação e tornar o pacote compatível com o Lambda? no Centro de Conhecimentos da AWS .

Usuário único do Amazon RDS e do Amazon Aurora MySQL

- Nome do modelo: SecretsManager RDSMy SQLRotation SingleUser
- Estratégia de alternância: the section called “Usuário único”.
- Estrutura esperada da **`SecretString`**: the section called “Credenciais do Amazon RDS e do Aurora”.
- Código fonte: https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerRDSMySQLRotationSingleUser/lambda _function.py
- Dependência: PyMy SQL 1.0.2. Se você usar a senha sha256 para autenticação, PyMy SQL [rsa]. Para obter informações sobre o uso de pacotes com código compilado em um runtime do Lambda, consulte Como faço para adicionar pacotes Python com binários compilados ao meu pacote de implantação e tornar o pacote compatível com o Lambda? no Centro de Conhecimentos da AWS .

Usuários em alternância do Amazon RDS e do Amazon Aurora MySQL

- Nome do modelo: SecretsManager RDSMy SQLRotation MultiUser
- Estratégia de alternância: the section called “Usuários alternados”.

- Estrutura esperada da **`SecretString`**: the section called “Credenciais do Amazon RDS e do Aurora”.
- Código fonte: https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerRDSMySQLRotationMultiUser/lambda _function.py
- Dependência: PyMy SQL 1.0.2. Se você usar a senha sha256 para autenticação, PyMy SQL [rsa]. Para obter informações sobre o uso de pacotes com código compilado em um runtime do Lambda, consulte Como faço para adicionar pacotes Python com binários compilados ao meu pacote de implantação e tornar o pacote compatível com o Lambda? no Centro de Conhecimentos da AWS .

Usuário único do Oracle do Amazon RDS

- Nome do modelo: SecretsManager RDSOracle RotationSingleUser
- Estratégia de alternância: the section called “Usuário único”.
- Estrutura esperada da **`SecretString`**: the section called “Credenciais do Amazon RDS e do Aurora”.
- Código fonte: https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerRDSOracleRotationSingleUser/lambda _function.py
- Dependência: python-oracledb 2.4.1

Usuários alternados do Oracle do Amazon RDS

- Nome do modelo: SecretsManager RDSOracle RotationMultiUser
- Estratégia de alternância: the section called “Usuários alternados”.
- Estrutura esperada da **`SecretString`**: the section called “Credenciais do Amazon RDS e do Aurora”.
- Código fonte: https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerRDSOracleRotationMultiUser/lambda _function.py
- Dependência: python-oracledb 2.4.1

Usuário único do Amazon RDS e do Amazon Aurora PostgreSQL

- Nome do modelo: SecretsManager RDSPostgre SQLRotation SingleUser
- Estratégia de alternância: Estratégia de alternância: usuário único.

- Estrutura esperada da **`SecretString`**: the section called “Credenciais do Amazon RDS e do Aurora”.
- Código fonte: https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerRDSPostgreSQLRotationSingleUser/lambda _function.py
- Dependência: PyGre SQL 5.2.5

Usuários em alternância do Amazon RDS e do Amazon Aurora PostgreSQL

- Nome do modelo: SecretsManager RDSPostgre SQLRotation MultiUser
- Estratégia de alternância: the section called “Usuários alternados”.
- Estrutura esperada da **`SecretString`**: the section called “Credenciais do Amazon RDS e do Aurora”.
- Código fonte: https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerRDSPostgreSQLRotationMultiUser/lambda _function.py
- Dependência: PyGre SQL 5.2.5

Amazon RDS Microsoft (usuário SQLServer único)

- Nome do modelo: SecretsManager RDSSQLServer RotationSingleUser
- Estratégia de alternância: the section called “Usuário único”.
- Estrutura esperada da **`SecretString`**: the section called “Credenciais do Amazon RDS e do Aurora”.
- Código fonte: https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerRDSSQLServerRotationSingleUser/lambda _function.py
- Dependência: Pymssql 2.2.2

Amazon RDS Microsoft (usuários SQLServer alternados)

- Nome do modelo: SecretsManager RDSSQLServer RotationMultiUser
- Estratégia de alternância: the section called “Usuários alternados”.
- Estrutura esperada da **`SecretString`**: the section called “Credenciais do Amazon RDS e do Aurora”.
- Código fonte: https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerRDSSQLServerRotationMultiUser/lambda _function.py

- Dependência: Pymssql 2.2.2

Amazon DocumentDB (compatível com MongoDB)

Usuário único do Amazon DocumentDB

- Nome do modelo: SecretsManagerMongo DBRotation SingleUser
- Estratégia de alternância: the section called “Usuário único”.
- Estrutura esperada da **`SecretString`**: the section called “Credenciais do Amazon DocumentDB”.
- Código fonte: https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerMongoDBRotationSingleUser/lambda _function.py
- Dependência: 4.2.0 PyMongo

Usuários alternados do Amazon DocumentDB

- Nome do modelo: SecretsManagerMongo DBRotation MultiUser
- Estratégia de alternância: the section called “Usuários alternados”.
- Estrutura esperada da **`SecretString`**: the section called “Credenciais do Amazon DocumentDB”.
- Código fonte: https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerMongoDBRotationMultiUser/lambda _function.py
- Dependência: 4.2.0 PyMongo

banco de dados de origem

Usuário único do Amazon Redshift

- Nome do modelo: SecretsManagerRedshiftRotationSingleUser
- Estratégia de alternância: the section called “Usuário único”.
- Estrutura esperada da **`SecretString`**: the section called “Credenciais do Amazon Redshift”.
- Código fonte: https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerRedshiftRotationSingleUser/lambda _function.py
- Dependência: PyGre SQL 5.2.5

Usuários alternados do Amazon Redshift

- Nome do modelo: SecretsManagerRedshiftRotationMultiUser
- Estratégia de alternância: the section called “Usuários alternados”.
- Estrutura esperada da **`SecretString`**: the section called “Credenciais do Amazon Redshift”.
- Código fonte: https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/ master/SecretsManagerRedshiftRotationMultiUser/lambda _function.py
- Dependência: PyGre SQL 5.2.5

Amazon Timestream para InfluxDB

Para usar esses modelos, consulte Como o Amazon Timestream para InfluxDB usa segredos no Guia do desenvolvedor do Amazon Timestream.

Usuário único do Amazon Timestream para InfluxDB

- Nome do modelo: SecretsManager Influx DBRotation SingleUser
- Estrutura esperada da **`SecretString`**: the section called “Estrutura de segredo do Amazon Timestream para InfluxDB”.
- Código fonte: https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/ master/SecretsManagerInfluxDBRotationSingleUser/lambda _function.py
- Dependência: cliente python do InfluxDB 2.0

Usuário alternados do Amazon Timestream para InfluxDB

- Nome do modelo: SecretsManagerInflux DBRotation MultiUser
- Estrutura esperada da **`SecretString`**: the section called “Estrutura de segredo do Amazon Timestream para InfluxDB”.
- Código fonte: https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/ master/SecretsManagerInfluxDBRotationMultiUser/lambda _function.py
- Dependência: cliente python do InfluxDB 2.0

Amazon ElastiCache

Para usar esse modelo, consulte Rotação automática de senhas para usuários no Guia do ElastiCache usuário da Amazon.

- Nome do modelo: SecretsManagerElasticacheUserRotation
- Estrutura esperada da **`SecretString`**: the section called “ElastiCache Credenciais da Amazon”.
- Código fonte: https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerElasticacheUserRotation/lambda _function.py

Active Directory

Credenciais do Active Directory

- Nome do modelo: SecretsManagerActiveDirectoryRotationSingleUser
- Estrutura esperada da **`SecretString`**: the section called “Credenciais do Active Directory”.
- Código fonte: https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerActiveDirectoryRotationSingleUser/lambda _function.py

Atributos do Active Directory

- Nome do modelo: SecretsManagerActiveDirectoryAndKeytabRotationSingleUser
- Estrutura esperada da **`SecretString`**: the section called “Credenciais do Active Directory”.
- Código fonte: https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerActiveDirectoryAndKeytabRotationSingleUser/lambda _function.py
- Dependências: msktutil

Outros tipos de segredo

O Secrets Manager fornece esse modelo como ponto de partida para você criar uma função de alternância para qualquer tipo de segredo.

- Nome do modelo: SecretsManagerRotationTemplate
- Código fonte: https://github.com/aws-samples/ aws-secrets-manager-rotation- lambdas/tree/master/SecretsManagerRotationTemplate/lambda _function.py

Permissões da função de execução da função de rotação Lambda para AWS Secrets Manager

Em the section called “Função do Lambda de alternância”, quando o Secrets Manager usa uma função do Lambda para alternar um segredo, o Lamba assume um perfil de execução do IAM e fornece essas credenciais ao código da função do Lambda. Para obter instruções sobre como configurar a alternância automática, consulte:

- Alternância automática para segredos de banco de dados (console)
- Alternância automática para segredos não de banco de dados (console)
- Alternância automática (AWS CLI)

Os exemplos a seguir mostram políticas em linha para funções de execução da função de alternância do Lambda. Para criar uma função de execução e anexá-la a uma política de permissões, consulte Perfil de execução do AWS Lambda.

Exemplos:

- Política para uma função de execução da função de alternância do Lambda
- Declaração de política para chaves gerenciadas pelo cliente
- Declaração de política para a estratégia de usuários alternados

Política para uma função de execução da função de alternância do Lambda

A política de exemplo a seguir permite que a função de alternância:

- Execute as operações do Secrets Manager para*SecretARN*.
- Criar uma nova senha.
- Definir a configuração necessária se seu banco de dados ou serviço for executado em uma VPC. Consulte Configuring a Lambda function to access resources in a VPC (Configurar uma função do Lambda para acessar recursos em uma VPC).

JSON

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "secretsmanager:UpdateSecretVersionStage"
            ],
            "Resource": "arn:aws:secretsmanager:us-
east-1:123456789012:secret:secretName-AbCdEf"
    },
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetRandomPassword"
            ],
            "Resource": "*"
        },
        {
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterface",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DetachNetworkInterface"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

Declaração de política para chaves gerenciadas pelo cliente

Se o segredo for criptografado com uma chave KMS diferente da Chave gerenciada pela AWS `aws/secretsmanager`, será necessário conceder à função de execução do Lambda permissão para usar a chave. Você pode usar o contexto de criptografia SecretARN para limitar o uso da função de descriptografia, para que a função de alternância tenha acesso apenas para descriptografar o segredo que é responsável pela rotação. O exemplo a seguir mostra uma instrução a ser adicionada à política de função de execução para descriptografar o segredo usando a chave KMS.

```
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey"
            ],
            "Resource": "KMSKeyARN",
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:SecretARN": "SecretARN"
                }
            }
        }
```

Para usar a função de alternância para vários segredos criptografados com uma chave gerenciada pelo cliente, adicione uma declaração como o exemplo a seguir para permitir que o perfil de execução decifre o segredo.

```
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey"
            ],
            "Resource": "KMSKeyARN",
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:SecretARN": [
                        "arn1",
                        "arn2"
                    ]
                }
            }
        }
```

Declaração de política para a estratégia de usuários alternados

Para obter informações sobre a estratégia de alternância de usuários alternados, consulte the section called “Estratégias de alternância da função do Lambda”.

Para um segredo que contém credenciais do Amazon RDS, se você estiver usando a estratégia de usuários alternados e o segredo do superusuário for gerenciado pelo Amazon RDS, você também deverá permitir que a função de rotação chame somente leitura no APIs Amazon RDS para que ela possa obter as informações de conexão do banco de dados. Recomendamos que você anexe a política AWS gerenciada da Amazon RDSRead OnlyAccess.

O exemplo de política a seguir permite que a função:

- Execute as operações do Secrets Manager para*SecretARN*.
- Recupere as credenciais no segredo do superusuário. O Secrets Manager usa as credenciais no segredo do superusuário para atualizar as credenciais no segredo que está sendo alternado.
- Criar uma nova senha.
- Definir a configuração necessária se seu banco de dados ou serviço for executado em uma VPC. Para obter mais informações, consulte Configuração de uma função do Lambda para acessar recursos em uma VPC.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "secretsmanager:UpdateSecretVersionStage"
            ],
            "Resource": "arn:aws:secretsmanager:us-
east-1:123456789012:secret:secretName-AbCdEf"
    },
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource": "arn:aws:secretsmanager:us-
east-1:123456789012:secret:secretName-AbCdEf"
```

```
    },
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetRandomPassword"
            ],
            "Resource": "*"
        },
        {
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterface",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DetachNetworkInterface"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

Acesso à rede para função AWS Lambda de rotação

Em the section called “Função do Lambda de alternância”, quando o Secrets Manager usa uma função do Lambda para alternar um segredo, a função de alternância do Lambda deve ser capaz de acessar o segredo. Se seu segredo contiver credenciais, a função do Lambda também deverá poder acessar a fonte dessas credenciais, como um banco de dados ou serviço.

Para acessar um segredo

A função de alternância do Lambda deve ser capaz de acessar um endpoint do Secrets Manager. Se sua função do Lambda puder acessar a Internet, é possível usar um endpoint público. Para localizar um endpoint, consulte the section called “Endpoint do Secrets Manager”.

Se a função do Lambda for executada em uma VPC que não tem acesso à Internet, recomendamos que você configure endpoints privados de serviço do Secrets Manager dentro de sua VPC. Assim, sua VPC pode interceptar solicitações endereçadas ao endpoint regional público e redirecioná-las para o endpoint privado. Para obter mais informações, consulte Endpoints da VPC (AWS PrivateLink).

Como alternativa, é possível habilitar a função do Lambda para acessar um endpoint público do Secrets Manager adicionando um gateway NAT ou um gateway da Internet à VPC, o que permite que o tráfego da VPC alcance o endpoint público. Isso expõe a VPC a um risco maior, pois um endereço IP do gateway pode ser atacado a partir da Internet pública.

(Opcional) para acessar o banco de dados ou o serviço

Para segredos como chaves de API, não há banco de dados ou serviço de origem que você precise atualizar junto com o segredo.

Se seu banco de dados ou serviço estiver sendo executado em uma EC2 instância da Amazon em uma VPC, recomendamos que você configure sua função Lambda para ser executada na mesma VPC. Assim, a função de alternância pode se comunicar diretamente com seu serviço. Para obter mais informações, consulte Configuring VPC access (Configurar o acesso à VPC).

Para permitir que a função do Lambda acesse o banco de dados ou serviço, certifique-se de que os grupos de segurança anexados à sua função de alternância do Lambda permitam conexões de saída com o banco de dados ou serviço. Você também deve garantir que os grupos de segurança anexados ao seu banco de dados ou serviço permitam conexões de entrada a partir da função de alternância do Lambda.

Solucionar problemas de rotação AWS Secrets Manager

Em muitos serviços, o Secrets Manager usa uma função do Lambda para alternar segredos. Para obter mais informações, consulte the section called “Função do Lambda de alternância”. A função de alternância do Lambda interage com o banco de dados ou serviço para o qual o segredo se destina, bem como o Secrets Manager. Quando a rotação não funciona da maneira esperada, você deve primeiro verificar os CloudWatch registros.

> **Note**
>
> Alguns serviços podem gerenciar segredos para você, incluindo o gerenciamento da alternância automática. Para obter mais informações, consulte the section called “Alternância gerenciada”.

Tópicos

- Como solucionar falhas de rotação secreta em funções AWS Lambda

- Nenhuma atividade após “Found credentials in environment variables” (Credenciais encontradas em variáveis de ambiente)
- Nenhuma atividade após “createSecret”
- Erro: “O acesso ao KMS não é permitido”
- Erro: “Key is missing from secret JSON” (A chave do segredo JSON está ausente)
- Erro: “setSecret: Unable to log into database” (setSecret: não é possível fazer login no banco de dados)
- Erro: “Não é possível importar o módulo ‘lambda_function’"
- Atualize uma função de alternância existente do Python 3.7 para 3.9
- Atualize uma função de alternância existente de Python 3.9 para 3.10
- AWS Lambda rotação secreta com PutSecretValue falha
- Erro: “Erro ao executar lambda <arn> durante a <a rotation> etapa”

Como solucionar falhas de rotação secreta em funções AWS Lambda

Se estiver enfrentando falhas de alternância de segredo nas funções do Lambda, use as etapas a seguir para solucionar o problema.

Possíveis causas

- Execuções simultâneas insuficientes para a função do Lambda
- Condições de corrida devido a várias chamadas de API durante a alternância
- Lógica incorreta da função do Lambda
- Problemas de rede entre a função do Lambda e o banco de dados

Etapas gerais de solução de problemas

1. Analise CloudWatch os registros:
    - Procure por mensagens de erro específicas ou comportamentos inesperados nos logs das funções do Lambda
    - Verifique se todas as etapas de alternância (CreateSecret, SetSecret, TestSecret, FinishSecret) estão sendo tentadas
2. Analise as chamadas de API durante a alternância:

- Evite fazer chamadas de API mutantes no segredo durante a alternância do Lambda
- Certifique-se de que não haja condições de corrida entre as chamadas RotateSecret e PutSecretValue

3. Verifique a lógica da função do Lambda:
   - Confirme se você está usando o código de AWS amostra mais recente para rotação secreta
   - Se estiver usando código personalizado, revise-o para verificar o tratamento adequado de todas as etapas da alternância
4. Verifique a configuração da rede:
   - Verifique se as regras do grupo de segurança permitem que a função do Lambda acesse o banco de dados
   - Garanta o acesso adequado ao endpoint da VPC ou ao endpoint público para o Secrets Manager
5. Teste as versões do segredo:
   - Verifique se a AWSCURRENT versão do segredo permite acesso ao banco de dados
   - Verifique se AWSPREVIOUS nossas AWSPENDING versões são válidas
6. Limpe as alternâncias pendentes:
   - Se a rotação falhar consistentemente, limpe a etiqueta AWSPENDING de preparação e tente novamente a rotação
7. Verifique as configurações de simultaneidade do Lambda:
   - Verifique se as configurações de simultaneidade são apropriadas para sua workload
   - Se você suspeitar de problemas de simultaneidade, consulte a seção “Solução de falhas de alternância relacionadas à simultaneidade”

Nenhuma atividade após “Found credentials in environment variables” (Credenciais encontradas em variáveis de ambiente)

Se não houver atividade após "Credenciais encontradas em variáveis de ambiente" e a duração da tarefa for longa, por exemplo, o tempo limite padrão do Lambda de 30 mil milissegundos, a função do Lambda pode estar expirando ao tentar alcançar o endpoint do Secrets Manager.

A função de alternância do Lambda deve ser capaz de acessar um endpoint do Secrets Manager. Se sua função do Lambda puder acessar a Internet, é possível usar um endpoint público. Para localizar um endpoint, consulte the section called “Endpoint do Secrets Manager”.

Se a função do Lambda for executada em uma VPC que não tem acesso à Internet, recomendamos que você configure endpoints privados de serviço do Secrets Manager dentro de sua VPC. Assim, sua VPC pode interceptar solicitações endereçadas ao endpoint regional público e redirecioná-las para o endpoint privado. Para obter mais informações, consulte Endpoints da VPC (AWS PrivateLink).

Como alternativa, é possível habilitar a função do Lambda para acessar um endpoint público do Secrets Manager adicionando um gateway NAT ou um gateway da Internet à VPC, o que permite que o tráfego da VPC alcance o endpoint público. Isso expõe a VPC a um risco maior, pois um endereço IP do gateway pode ser atacado a partir da Internet pública.

Nenhuma atividade após “createSecret”

A seguir, estão os problemas que podem fazer com que a alternância pare após createSecret:

A rede VPC ACLs não permite entrada e saída de tráfego HTTPS.

> Para obter mais informações, consulte Controle o tráfego para sub-redes usando a rede ACLs no Guia do usuário da Amazon VPC.

A configuração de tempo limite da função do Lambda é muito curta para executar a tarefa.

> Para obter mais informações, consulte Configurar as opções da função do Lambda no Guia do desenvolvedor do AWS Lambda .

O endpoint VPC do Secrets Manager não permite que a VPC entre CIDRs nos grupos de segurança atribuídos.

> Para obter mais informações, consulte Controle o tráfego para recursos usando grupos de segurança no Guia do usuário da Amazon VPC.

A política de endpoint da VPC do Secrets Manager não permite que o Lambda use o endpoint da VPC.

> Para obter mais informações, consulte the section called “Endpoints da VPC (AWS PrivateLink)”.

O segredo usa a alternância de usuários alternados, o segredo do superusuário é gerenciado pelo Amazon RDS e a função do Lambda não pode acessar a API do RDS.

Para rodízio de usuários em alternância em que o segredo do superusuário é gerenciado por outro produto da AWS, a função de rodízio do Lambda deve ser capaz de chamar o endpoint do serviço para obter as informações de conexão do banco de dados. Recomendamos que você configure um endpoint da VPC para o serviço de banco de dados. Para obter mais informações, consulte:

- API do Amazon RDS e endpoints da VPC de interface no Guia do usuário do Amazon RDS.
- Trabalhar com endpoints da VPC no Guia de gerenciamento do Amazon Redshift.

Erro: “O acesso ao KMS não é permitido”

Se você vir `ClientError: An error occurred (AccessDeniedException) when calling the GetSecretValue operation: Access to KMS is not allowed`, a função de alternância não tem permissão para descriptografar o segredo usando a chave KMS usada para criptografar o segredo. Pode haver uma condição na política de permissões que limita o contexto de criptografia a um segredo específico. Para obter informações sobre as permissões necessárias, consulte the section called “Declaração de política para chaves gerenciadas pelo cliente”.

Erro: “Key is missing from secret JSON” (A chave do segredo JSON está ausente)

Uma função de alternância do Lambda requer que o valor do segredo esteja em uma estrutura JSON específica. Se você vir esse erro, o JSON pode estar sem uma chave que a função de alternância tentou acessar. Para obter informações sobre a estrutura JSON para cada tipo de segredo, consulte the section called “Estrutura JSON de um segredo”.

Erro: “setSecret: Unable to log into database” (setSecret: não é possível fazer login no banco de dados)

A seguir, estão os problemas que podem causar esse erro:

A função de alternância não pode acessar o banco de dados.

Se a duração da tarefa for longa, por exemplo, mais de 5 milissegundos, a função de alternância do Lambda pode não conseguir acessar o banco de dados utilizando a rede.

Se seu banco de dados ou serviço estiver sendo executado em uma instância do Amazon EC2 em uma VPC, é recomendável que você configure a função do Lambda para ser executada na

mesma VPC. Assim, a função de alternância pode se comunicar diretamente com seu serviço. Para obter mais informações, consulte Configuring VPC access (Configurar o acesso à VPC).

Para permitir que a função do Lambda acesse o banco de dados ou serviço, certifique-se de que os grupos de segurança anexados à sua função de alternância do Lambda permitam conexões de saída com o banco de dados ou serviço. Você também deve garantir que os grupos de segurança anexados ao seu banco de dados ou serviço permitam conexões de entrada a partir da função de alternância do Lambda.

As credenciais no segredo estão incorretas.

Se a duração da tarefa for curta, a função de alternância do Lambda pode não conseguir autenticar com as credenciais no segredo. Verifique as credenciais fazendo login manualmente com as informações nas `AWSPREVIOUS` versões `AWSCURRENT` e do segredo usando o AWS CLI comando `get-secret-value`.

O banco de dados usa **`scram-sha-256`** para criptografar senhas.

Se seu banco de dados for Aurora PostgreSQL versão 13 ou posterior e usar `scram-sha-256` para criptografar senhas, mas a função de alternância usar `libpq` versão 9 ou anterior que não seja compatível com `scram-sha-256`, a função de alternância não poderá estabelecer conexão com o banco de dados.

Para determinar quais usuários do banco de dados usam criptografia **`scram-sha-256`**

- Consulte Checking for users with non-SCRAM passwords (Verificação de usuários com senhas não SCRAM) na postem de blog SCRAM Authentication in RDS for PostgreSQL 13 (Autenticação SCRAM no RDS para PostgreSQL 13).

Para determinar a versão que sua função de alternância **`libpq`** usa

1. Em um computador baseado em Linux, no console do Lambda, acesse sua função de alternância e baixe o pacote de implantação. Descompacte o arquivo zip em um diretório de trabalho.
2. Usando uma linha de comando, no diretório de trabalho, execute:

   ```
   readelf -a libpq.so.5 | grep RUNPATH
   ```
3. Se você visualizar a string *`PostgreSQL-9.4.x`* ou qualquer versão principal inferior à versão 10, a função de alternância não será compatível com `scram-sha-256`.
   - Saída para uma função de alternância incompatível com `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]
```

- Saída para uma função de alternância compatível com `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]
```

- Saída para uma função de alternância compatível com `scram-sha-256`:

```
0x000000000000001d (RUNPATH) Library runpath: [/local/
p4clients/pkgbuild- a1b2c /workspace/build/PostgreSQL/
PostgreSQL-14.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]
```

- Saída para uma função de alternância compatível com `scram-sha-256`:

```
0x000000000000001d (RUNPATH) Library runpath: [/local/p4clients/
pkgbuild- a1b2c/workspace/build/PostgreSQL/PostgreSQL-
14.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**
>
> Se tiver configurado a alternância automática de segredos antes de 30 de dezembro de 2021, sua função de alternância incluiu uma versão anterior de `libpq` que não oferece

suporte a `scram-sha-256`. Para oferecer suporte a `scram-sha-256`, você precisa recriar sua função de alternância.

O banco de dados requer SSL/TLS acesso.

Se seu banco de dados exigir uma SSL/TLS conexão, mas a função de rotação usar uma conexão não criptografada, a função de rotação não poderá se conectar ao banco de dados. As funções de rodízio do Amazon RDS (exceto Oracle e Db2) e o Amazon DocumentDB usam automaticamente o Secure Socket Layer (SSL) ou o Transport Layer Security (TLS) para se conectar ao seu banco de dados, se ele estiver disponível. Caso contrário, utilizarão uma conexão não criptografada.

> **Note**
>
> Se você configurou a rotação secreta automática antes de 20 de dezembro de 2021, sua função de rotação pode ser baseada em um modelo anterior que não era compatívelSSL/TLS. To support connections that use SSL/TLS. Você precisará recriar sua função de rotação.

Para determinar quando a sua função de alternância foi criada

1. No console do Secrets Manager https://console.aws.amazon.com/secretsmanager/, abra seu segredo. Na seção Configuração de alternância, em Função de alternância do Lambda, você verá o ARN da função do Lambda, por exemplo, `arn:aws:lambda:`*`aws-region`*`:`*`123456789012`*`:function:`*`SecretsManagerMyRotationFunction`*. Copie o nome da função do final do ARN. Neste exemplo, é *`SecretsManagerMyRotationFunction`* .
2. No AWS Lambda console https://console.aws.amazon.com/lambda/, em Funções, cole o nome da função Lambda na caixa de pesquisa, escolha Enter e escolha a função Lambda.
3. Na página de detalhes da função, na guia Configuration (Configuração), em Tags (Etiquetas), copie o valor ao lado da chave aws:cloudformation:stack-name.
4. No AWS CloudFormation console https://console.aws.amazon.com/cloudformation, em Stacks, cole o valor da chave na caixa de pesquisa e escolha Enter.

5. A lista de filtros de pilhas para que somente a pilha que criou a função de alternância do Lambda apareça. Na coluna Created date (Data da criação), visualize a data em que a pilha foi criada. Esta é a data em que a função de alternância do Lambda foi criada.

Erro: “Não é possível importar o módulo ‘lambda_function’"

É possível receber esse erro se estiver executando uma função do Lambda anterior que foi atualizada automaticamente do Python 3.7 para uma versão mais recente do Python. Para solucionar o erro, é possível alterar a versão da função do Lambda de volta para Python 3.7 e, em seguida, para the section called “Atualize uma função de alternância existente do Python 3.7 para 3.9”. Para obter mais informações, consulte Por que a alternância da minha função do Lambda do Secrets Manager falhou com o erro “módulo pg não encontrado”? no AWS re:Post.

Atualize uma função de alternância existente do Python 3.7 para 3.9

Algumas funções de alternância criadas antes de novembro de 2022 usavam o Python 3.7. O AWS SDK para Python deixou de oferecer suporte ao Python 3.7 em dezembro de 2023. Para obter mais informações, consulte Atualizações AWS SDKs e ferramentas da política de suporte do Python. Para mudar para uma nova função de alternância que usa o Python 3.9, é possível adicionar uma propriedade de runtime a uma função de alternância existente ou recriá-la.

Para descobrir quais funções de alternância do Lambda usam o Python 3.7

1. Faça login no Console de gerenciamento da AWS e abra o AWS Lambda console em https://console.aws.amazon.com/lambda/.
2. Na lista de Funções, filtre por **`SecretsManager`**.
3. Na lista filtrada de funções, em Runtime, busque o Python 3.7.

Para atualizar para o Python 3.9:

- Opção 1: Recriar a função de rotação usando CloudFormation
- Opção 2: atualizar o tempo de execução da função de rotação existente usando CloudFormation
- Opção 3: Para AWS CDK usuários, atualize a biblioteca CDK

Opção 1: Recriar a função de rotação usando CloudFormation

Quando você usa o console do Secrets Manager para ativar a rotação, o Secrets Manager usa CloudFormation para criar os recursos necessários, incluindo a função de rotação do Lambda. Se

você usou o console para ativar a rotação ou criou a função de rotação usando uma CloudFormation pilha, poderá usar a mesma CloudFormation pilha para recriar a função de rotação com um novo nome. A nova função usa a versão mais recente do Python.

Para encontrar a CloudFormation pilha que criou a função de rotação

- Na página de detalhes da função do Lambda, na guia Configuração, escolha Tags. Visualize o ARN próximo à aws:cloudformation:stack-id.

  O nome da pilha está incorporado no ARN, conforme exibido no exemplo a seguir.

  - ARN: `arn:aws:cloudformation:us-west-2:408736277230:stack/`**`SecretsManagerRDSMySQLRotationSingleUser5c2-SecretRotationScheduleHostedRotationLambda`**`-3CUDHZMDMBO8/79fc9050-2eef-11ed-`
  - Nome da pilha: **`SecretsManagerRDSMySQLRotationSingleUser5c2-SecretRotationScheduleHostedRotationLambda`**

Para recriar uma função de alternância (CloudFormation)

1. Em CloudFormation, pesquise a pilha pelo nome e escolha Atualizar.

   Se aparecer uma caixa de diálogo recomendando a atualização da pilha raiz, escolha Ir para a pilha raiz e, em seguida, escolha Atualizar.
2. Na página Atualizar pilha, em Preparar modelo, escolha Editar no Application Composer e, em Editar modelo no Application Composer, escolha o botão Editar no Application Composer.
3. No Application Composer, faça o seguinte:

   a. No código do modelo, em `SecretRotationScheduleHostedRotationLambda`, substitua o valor por `"functionName": "SecretsManagerTestRotationRDS"` com um novo nome de função. Por exemplo, em JSON, **`"functionName": "SecretsManagerTestRotationRDSupdated"`**.

   b. Escolha Atualizar modelo.

   c. Na caixa de diálogo Prosseguir para CloudFormation, escolha Confirmar e prosseguir para CloudFormation.
4. Continue com o fluxo de trabalho da CloudFormation pilha e escolha Enviar.

Opção 2: atualizar o tempo de execução da função de rotação existente usando CloudFormation

Quando você usa o console do Secrets Manager para ativar a rotação, o Secrets Manager usa CloudFormation para criar os recursos necessários, incluindo a função de rotação do Lambda. Se você usou o console para ativar a rotação ou criou a função de rotação usando uma CloudFormation pilha, poderá usar a mesma CloudFormation pilha para atualizar o tempo de execução da função de rotação.

Para encontrar a CloudFormation pilha que criou a função de rotação

- Na página de detalhes da função do Lambda, na guia Configuração, escolha Tags. Visualize o ARN próximo à aws:cloudformation:stack-id.

  O nome da pilha está incorporado no ARN, conforme exibido no exemplo a seguir.

  - ARN: `arn:aws:cloudformation:us-west-2:408736277230:stack/`**`SecretsManagerRDSMySQLRotationSingleUser5c2-SecretRotationScheduleHostedRotationLambda`**`-3CUDHZMDMBO8/79fc9050-2eef-11ed-`
  - Nome da pilha: **`SecretsManagerRDSMySQLRotationSingleUser5c2-SecretRotationScheduleHostedRotationLambda`**

Para atualizar o runtime de uma função de alternância (CloudFormation)

1. Em CloudFormation, pesquise a pilha pelo nome e escolha Atualizar.

   Se aparecer uma caixa de diálogo recomendando a atualização da pilha raiz, escolha Ir para a pilha raiz e, em seguida, escolha Atualizar.
2. Na página Atualizar pilha, em Preparar modelo, escolha Editar no Application Composer e, em Editar modelo no Application Composer, escolha o botão Editar no Application Composer.
3. No Application Composer, faça o seguinte:

   a. No modelo JSON, para `SecretRotationScheduleHostedRotationLambda`, em `Properties`, em `Parameters`, adicione **`"runtime": "python3.9"`**.

   b. Escolha Atualizar modelo.

   c. Na caixa de diálogo Prosseguir para CloudFormation, escolha Confirmar e prosseguir para CloudFormation.
4. Continue com o fluxo de trabalho da CloudFormation pilha e escolha Enviar.

Opção 3: Para AWS CDK usuários, atualize a biblioteca CDK

Se você usou a versão AWS CDK anterior à v2.94.0 para configurar a rotação do seu segredo, você pode atualizar a função Lambda fazendo o upgrade para a v2.94.0 ou posterior. Para obter mais informações, consulte o Guia do desenvolvedor do AWS Cloud Development Kit (AWS CDK) v2.

Atualize uma função de alternância existente de Python 3.9 para 3.10

O Secrets Manager está fazendo a transição de Python 3.9 para 3.10 para funções de alternância do Lambda. Para mudar para uma nova função de alternância que use o Python 3.10, é necessário seguir o caminho de atualização com base no método de implantação. Use os procedimentos a seguir para atualizar a versão do Python e as dependências subjacentes.

Para descobrir quais funções de rotação do Lambda, use o Python 3.9

1. Faça login no Console de gerenciamento da AWS e abra o AWS Lambda console em https://console.aws.amazon.com/lambda/.
2. Na lista de Funções, filtre por **`SecretsManager`**.
3. Na lista filtrada de funções, em Runtime, busque por **`Python 3.9`**.

Atualização de caminhos por método de implantação

As funções de rotação do Lambda identificadas nessa lista podem ser implantadas por meio do console, AWS Serverless Application Repository aplicativos ou transformações do Secrets Manager. CloudFormation Cada uma dessas estratégias de implantação tem um caminho de atualização distinto.

Use um dos procedimentos a seguir para atualizar suas funções de alternância do Lambda, dependendo de como sua função foi implantada.

AWS Secrets Manager console-deployed functions

Uma nova função Lambda deve ser implantada por meio do AWS Secrets Manager console, pois você não pode atualizar manualmente as dependências das funções Lambda existentes.

Use o procedimento a seguir para atualizar as funções implantadas no AWS Secrets Manager console.

1. Abra o console do Secrets Manager em https://console.aws.amazon.com/secretsmanager/.

2. Em AWS Secrets Manager, selecione Segredos. Selecione o segredo que usa a função do Lambda que deseja atualizar.
3. Navegue até a guia Alternâncias e selecione a opção Atualizar configurações de alternância.
4. Em Funções de alternância, escolha Criar uma nova função e insira um nome para a função de alternância do Lambda.

    a. (Opcional) Depois que a atualização for concluída, será possível testar a função do Lambda atualizada para confirmar se ela funciona conforme o esperado. Na guia Alternância, selecione Alternar segredo imediatamente para iniciar uma alternância imediata.

    b. (Opcional) Você pode visualizar seus registros de funções e a versão do Python usada em tempo de execução na Amazon. CloudWatch Para obter mais informações, consulte Visualização de CloudWatch registros de funções Lambda no Guia do AWS Lambda desenvolvedor.
5. Depois que a nova função de alternância estiver configurada, será possível excluir a função de alternância antiga.

AWS Serverless Application Repository deployments

O procedimento a seguir mostra como atualizar as AWS Serverless Application Repository implantações. As funções Lambda implantadas por meio AWS Serverless Application Repository de um banner informando `This function belongs to an application. Click here to manage it.` que inclui um link para o aplicativo Lambda ao qual a função pertence.

> **Important**
>
> AWS Serverless Application Repository a disponibilidade Região da AWS depende.

Use o procedimento a seguir para atualizar as funções AWS Serverless Application Repository implantadas.

1. Abra o AWS Lambda console em https://console.aws.amazon.com/lambda/.
2. Navegue até a guia Configurações da função do Lambda que precisa ser atualizada.

- Você precisará das seguintes informações sobre sua função ao atualizar o AWS Serverless Application Repository aplicativo implantado. É possível encontrar estas informações no console do Lambda.

  - Nome da aplicação do Lambda
    - O nome da aplicação do Lambda pode ser encontrado usando o link no banner. Por exemplo, o banner indica `serverlessrepo-`*`SecretsManagerRedshiftRotationSingleUser`*. Neste exemplo, o nome é `SecretsManagerRedshiftRotationSingleUser`.
  - Nome da função de alternância do Lambda
  - Ponto final do Secrets Manager
    - O endpoint pode ser encontrado nas guias Configurações e Variáveis de ambiente atribuídas à variável SECRETS_MANAGER_ENDPOINT.

3. Para atualizar o Python, será necessário atualizar a versão semântica da aplicação sem servidor. Consulte Atualização de aplicações no Guia do desenvolvedor do AWS Serverless Application Repository .

Custom Lambda rotation functions

Se você criou funções personalizadas de alternância do Lambda, precisará atualizar as dependências e os runtimes de cada pacote para essas funções. Para obter mais informações, consulte Atualização do runtime da função do Lambda para a versão mais recente.

AWS::SecretsManager-2024-09-16 transform macro

Se a função do Lambda for implantada por meio dessa transformação, a atualização das pilhas usando o modelo existente permitirá que você use o runtime atualizado do Lambda.

Use o procedimento a seguir para atualizar a CloudFormation pilha usando o modelo existente.

1. Abra o CloudFormation console em https://console.aws.amazon.com/cloudformation.
2. Na página Pilhas, selecione a pilha que você deseja atualizar.
3. Escolha Atualizar no painel de detalhes da pilha.
4. Em Escolha um método de atualização de modelo, selecione Atualização direta.
5. Na página Especificar modelo, selecione Usar modelo existente.
6. Deixe todas as outras opções nos valores padrão e escolha Atualizar pilha.

Se você tiver problemas para atualizar a pilha, consulte Determinação da causa de uma falha na pilha no Guia do usuário do CloudFormation .

AWS::SecretsManager-2020-07-23 transform macro

Recomendamos que você migre para a versão de transformação mais recente se estiver usando o `AWS::SecretsManager-2020-07-23`. Consulte Apresentando uma versão aprimorada da AWS Secrets Manager transformação:: AWS: SecretsManager -2024-09-16 no blog de AWS segurança para obter mais informações. Se você continuar usando o `AWS::SecretsManager-2020-07-23`, poderá encontrar um erro de incompatibilidade entre sua versão de runtime e os artefatos do código da função do Lambda. Para obter mais informações, consulte AWS:SecretsManager:: RotationSchedule HostedRotationLambda na Referência do CloudFormation modelo.

Se você tiver problemas para atualizar a pilha, consulte Determinação da causa de uma falha na pilha no Guia do usuário do CloudFormation .

Verificação da atualização do Python

Para verificar a atualização do Python, abra o console Lambda (https://console.aws.amazon.com/lambda/) e acesse a página Function. Selecione a função que você atualizou. Na seção Fonte do código, revise os arquivos incluídos no diretório e verifique se a versão do arquivo .so do Python é `3.10`.

AWS Lambda rotação secreta com **`PutSecretValue`** falha

Se você usa uma função assumida ou uma rotação entre contas com o Secrets Manager e encontra um RotationFailed evento CloudTrail com a mensagem: A versão secreta pendente do *`VERSION_ID`* Secret não *`SECRET_ARN`* foi criada pelo *`LAMBDA_ARN.`* Lambda Remova `AWSPENDING` o rótulo de teste e reinicie a rotação, então você precisa atualizar sua função Lambda para usar o parâmetro. `RotationToken`

Atualização da função de alternância Lambda para incluir **`RotationToken`**

1. Baixe o código da função do Lambda

   - Abra o console do Lambda
   - Selecione Funções no painel de navegação
   - Selecione sua função de alternância de segredos do Lambda em Nome da função

- Em Download, escolha um dentre Códigos de função .zip, Arquivo AWS SAM , Ambos
- Escolha OK para salvar a função em sua máquina local.

2. Editar o `Lambda_handler`

   Inclua o parâmetro rotation_token na etapa create_secret para alternância entre contas:

```
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. Edição do código `create_secret`

   Revise a função `create_secret` para aceitar e usar o parâmetro `rotation_token`:

```
# 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. Transferência do código atualizado da função do Lambda

   Depois de atualizar o código da função do Lambda, transfira-o para alternar seu segredo.

Erro: “Erro ao executar lambda *<arn>* durante a *<a rotation>* etapa”

Se você estiver enfrentando falhas intermitentes de alternância de segredos com sua função do Lambda presa em um loop de conjuntos, por exemplo, entre CreateSecret e SetSecret, o problema pode estar relacionado às configurações de simultaneidade.

Etapas da solução de problemas de simultaneidade

> **Warning**
>
> Definir o parâmetro de simultaneidade provisionado como um valor menor que 10 pode causar limitação devido à insuficiência de threads de execução para a função do Lambda. Para obter mais informações, consulte Noções básicas sobre simultaneidade reservada e simultaneidade provisionada no Guia do desenvolvedor do AWS Lambda AWS Lambda .

1. Verifique e ajuste as configurações de simultaneidade do Lambda:
   - Verifique se `reserved_concurrent_executions` não está definido como muito baixo (por exemplo, 1)
   - Se estiver usando simultaneidade reservada, defina para pelo menos 10
   - Considere o uso de simultaneidade sem reservas para obter mais flexibilidade
2. Para simultaneidade provisionada:
   - Não defina explicitamente o parâmetro de simultaneidade provisionado (por exemplo, no Terraform).
   - Se você precisar configurá-lo, use um valor de pelo menos 10.
   - Teste minuciosamente para garantir que o valor escolhido funcione para seu caso de uso.
3. Monitore e ajuste a simultaneidade:

- Calcule a simultaneidade usando esta fórmula: Concorrência = (média de solicitações por segundo) * (duração média da solicitação em segundos). Para obter mais informações, consulte Estimativa de simultaneidade reservada.
- Observe e registre os valores durante as alternâncias para determinar as configurações de simultaneidade apropriadas.
- Tenha cuidado ao definir valores baixos de simultaneidade. Eles podem causar controle de utilização se não houver threads de execução suficientes disponíveis.

Para obter mais informações sobre como configurar a simultaneidade do Lambda, consulte Configurando a simultaneidade reservada e Configurando a simultaneidade provisionada no Guia do desenvolvedor. AWS Lambda

Programação de alternância

O Secrets Manager faz a alternância do seu segredo em uma programação durante uma janela de alternância definida por você. Para definir a programação e a janela, use uma expressão cron() ou rate() junto com a duração da janela. O Secrets Manager alterna seu segredo a qualquer momento durante a janela de alternância. É possível alternar um segredo com intervalos a partir de quatro horas em uma janela de alternância.

Para ativar a alternância, consulte:

- the section called “Alternância gerenciada”
- the section called “Alternância automática para segredos de banco de dados (console)”
- the section called “Alternância automática para segredos não de banco de dados (console)”

Os horários de alternância do Secrets Manager usam o fuso horário UTC.

Janelas de alternância

A janela de alternância do Secrets Manager é semelhante a uma janela de manutenção. Você define a janela de alternância quando deseja alternar seu segredo, e o Secrets Manager alterna seu segredo em algum momento durante a janela de alternância.

As janelas de alternância do Secrets Manager sempre começam na hora. Para uma programação de alternância que use uma expressão `rate()` em dias, a janela de alternância começa à meia-noite. É

possível definir a hora de início da janela de alternância usando uma expressão `cron()`. Para obter exemplos, consulte the section called “Expressão cron”.

Por padrão, a janela de alternância fecha após uma hora para uma programação de alternância em horas, e no final do dia para uma programação de alternância em dias.

Para alterar o comprimento da janela de alternância, defina a Duração de janela. É possível definir a janela de alternância tão pequena quanto uma hora. A janela de alternância não pode se estender até a próxima janela de alternância. Em outras palavras, para um cronograma de alternância em horas, confirme se a janela de alternância é menor ou igual ao número de horas entre as alternâncias. Para um cronograma de alternância em dias, confirme se o horário inicial somado à duração da janela é menor ou igual a 24 horas.

Expressões rate

As expressões de taxa do Secrets Manager têm o seguinte formato, onde *`Value`* é um número inteiro positivo e *`Unit`* pode ser`hour`,`hours`,`day`, ou`days`:

```
rate(Value Unit)
```

É possível alternar um segredo com intervalos a partir de quatro horas. O período máximo de alternância é de 999 dias. Exemplos:

- `rate(4 hours)` significa que o segredo é alternado a cada 4 horas.
- `rate(1 day)` significa que o segredo é alternado a cada dia.
- `rate(10 days)` significa que o segredo é alternado a cada 10 dias.

Expressão cron

As expressões cron do Secrets Manager têm o formato a seguir:

```
cron(Minutes Hours Day-of-month Month Day-of-week Year)
```

Uma expressão cron que inclui incrementos de redefinições de horas a cada dia. Por exemplo, `cron(0 4/12 * * ? *)` significa 4h, 16h e, no dia seguinte, 4h, 16h. Os horários de alternância do Secrets Manager usam o fuso horário UTC.

Exemplo de programação	Expressão
A cada 8 horas a partir da meia-noite.	`cron(0 /8 * * ? *)`
A cada 8 horas a partir das 8h.	`cron(0 8/8 * * ? *)`
A cada 10 horas a partir das 2h. As janelas de alternância começarão às 2h, 12h e 22h, e no dia seguinte, às 2h, 12h e 22h.	`cron(0 2/10 * * ? *)`
Todos os dias às 10h.	`cron(0 10 * * ? *)`
Todos os sábados às 18h.	`cron(0 18 ? * SAT *)`
O primeiro dia de cada mês, às 8h.	`cron(0 8 1 * ? *)`
A cada três meses, no primeiro domingo, à 1 hora da manhã.	`cron(0 1 ? 1/3 SUN#1 *)`
O último dia de cada mês, às 17h.	`cron(0 17 L * ? *)`
De segunda-feira a sexta-feira, às 8h.	`cron(0 8 ? * MON-FRI *)`
Primeiro e 15.º dia de cada mês às 16h.	`cron(0 16 1,15 * ? *)`
Primeiro domingo de cada mês à 0h.	`cron(0 0 ? * SUN#1 *)`
A partir de janeiro, a cada 11 meses na primeira segunda-feira à meia-noite.	`cron(0 0 ? 1/11 2#1 *)`

Requisitos de expressão cron no Secrets Manager

O Secrets Manager tem algumas restrições quanto ao que pode ser usado em expressões cron. Uma expressão cron para o Secrets Manager deve ter 0 no campo de minutos, pois as janelas de alternância do Secrets Manager começam na hora indicada. É necessário que um * esteja no campo de ano, porque o Secrets Manager não oferece suporte a cronogramas de alternância com mais de um ano de intervalo. A tabela a seguir mostra as opções que você pode utilizar.

Campos	Valores	Curingas
Minutos	Deve ser 0	Nenhum
Horas	0–23	Use / (barra) para especificar incrementos. Por exemplo, `2/10` significa a cada 10 horas a partir das 2h. É possível alternar um segredo com intervalos a partir de quatro horas.
D ay-of-month	1–31	Use , (vírgula) para incluir valores adicionais. Por exemplo, `1,15` significa o 1.º e o 15.º dia do mês. Use - (traço) para especificar um intervalo. Por exemplo, `1-15` significa do dia 1 ao dia 15 do mês. Use * (asterisco) para incluir todos os valores no campo. Por exemplo, `*` significa todos os dias do mês. O curinga ? (interrogação) especifica um ou outro. Não é possível especificar os campos `Day-of-month` e `Day-of-week` na mesma expressão cron. Se você especificar um valor em um dos campos, deverá usar um ? (ponto de interrogação) no outro.

Campos	Valores	Curingas
		Use / (barra) para especificar incrementos. Por exemplo, `1/2` significa a cada 2 dias a partir do dia 1, ou seja, dia 1, 3, 5 e assim por diante. Use L para especificar o último dia do mês. Use ***DAY***L para especificar o último dia nomeado do mês. Por exemplo, `SUNL` significa o último domingo do mês.
Mês	1-12 ou JAN-DEZ	Use , (vírgula) para incluir valores adicionais. Por exemplo, `JAN,APR,JUL,OCT` significa janeiro, abril, julho e outubro. Use - (traço) para especificar um intervalo. Por exemplo, `1-3` significa do mês 1 ao mês 3 do ano. Use * (asterisco) para incluir todos os valores no campo. Por exemplo, `*` significa todos os meses. Use / (barra) para especificar incrementos. Por exemplo, `1/3` significa a cada 3 meses, começando no mês 1, ou seja, mês 1, 4, 7 e 10.

Campos	Valores	Curingas
D ay-of-week	1-7 ou DOM-SÁB	Use # para especificar o dia da semana em um mês. Por exemplo, `TUE#3` significa a terceira terça-feira do mês. Use , (vírgula) para incluir valores adicionais. Por exemplo, `1,4` significa o 1.º e o 4.º dia da semana. Use - (traço) para especificar um intervalo. Por exemplo, `1-4` significa do dia 1 ao dia 4 da semana. Use * (asterisco) para incluir todos os valores no campo. Por exemplo, `*` significa todos os dias da semana. O curinga ? (interrogação) especifica um ou outro. Não é possível especificar os campos `Day-of-month` e `Day-of-week` na mesma expressão cron. Se você especificar um valor em um dos campos, deverá usar um ? (ponto de interrogação) no outro. Use / (barra) para especific ar incrementos. Por exemplo, `1/2` significa cada 2.º dia da semana, começando no

Campos	Valores	Curingas
		primeiro dia, ou seja, dias 1, 3, 5 e 7. Use L para especificar o último dia da semana.
Ano	Deve ser *	Nenhum

Gire um AWS Secrets Manager segredo imediatamente

Você só pode alternar um segredo que tenha a alternância configurada. Para determinar se um segredo foi configurado para alternância, no console, visualize o segredo e role para baixo até a seção Rotation configuration (Configuração de alternância). Se o Rotation status (Estado de alternância) for Enabled (Habilitado), o segredo será configurado para alternância. Se não, consulte Alternar segredos .

Para alternar um segredo imediatamente (console)

1. Abra o console do Secrets Manager em https://console.aws.amazon.com/secretsmanager/.
2. Escolha seu segredo.
3. Na página de detalhes do segredo, em Rotation configuration (Configuração de alternância), escolha Rotate secret immediately (Alternar o segredo imediatamente).
4. Na caixa de diálogo Rotate secret (Alternar segredo), escolha Rotate (Alternar).

AWS CLI

Example Alternar um segredo imediatamente

O exemplo de rotate-secret a seguir inicia uma alternância imediata. O segredo já deve ter a alternância configurada.

```
$ aws secretsmanager rotate-secret \
    --secret-id MyTestSecret
```

Encontrar segredos que não são alternados

Você pode usar AWS Config para avaliar seus segredos para ver se eles estão alternando de acordo com seus padrões. Você define seus requisitos internos de segurança e conformidade para segredos usando AWS Config regras. Em seguida, AWS Config pode identificar segredos que não estão de acordo com suas regras. Você também pode rastrear alterações em metadados de segredos, a configuração de alternância, a chave KMS usada para criptografia de segredos, a função de alternância do Lambda e as tags associadas a um segredo.

Se você tiver segredos em vários Contas da AWS e Regiões da AWS em sua organização, poderá agregar esses dados de configuração e conformidade. Para obter mais informações, consulte Agregação de dados de várias contas e regiões.

Para avaliar se os segredos estão sendo alternados

1. Siga as instruções em Como avaliar seus recursos com AWS Config regras e escolha uma das seguintes regras:

   - `secretsmanager-rotation-enabled-check`: verifica se a alternância está configurada para segredos armazenados no Secrets Manager.
   - `secretsmanager-scheduled-rotation-success-check`: verifica se a última alternância bem-sucedida está dentro da frequência de alternância configurada. A frequência mínima para a verificação é diária.
   - `secretsmanager-secret-periodic-rotation`: verifica se os segredos foram alternados dentro do número de dias especificado.

2. Opcionalmente, configure AWS Config para notificá-lo quando os segredos não estiverem em conformidade. Para obter mais informações, consulte Notificações AWS Config enviadas para um tópico do Amazon SNS.

Cancelar a alternância automática no Secrets Manager

Se você configurou a alternância automática para um segredo e deseja parar de alterná-lo, poderá cancelar a alternância.

Para cancelar a alternância automática

1. Abra o console do Secrets Manager em https://console.aws.amazon.com/secretsmanager/.

2. Escolha seu segredo.
3. Na página de detalhes do segredo, em Configuração da alternância, escolha Editar alternância.
4. Na caixa de diálogo Editar configuração de alternância, desative a Alternância automática e escolha Salvar.

   O Secrets Manager retém as informações de configuração de alternância para que você possa usá-las no futuro, caso decida reativar a alternância.

AWS Secrets Manager segredos gerenciados por outros AWS serviços

Muitos AWS serviços armazenam e usam segredos em AWS Secrets Manager. Em alguns casos, esses segredos são segredos gerenciados, o que significa que o serviço que os criou ajuda a gerenciá-los. Por exemplo, alguns segredos gerenciados incluem alternância gerenciada para que você não precise configurar a alternância por conta própria. O serviço de gerenciamento também pode impedir você de atualizar segredos ou excluí-los sem um período de recuperação, o que ajuda a evitar interrupções, uma vez que o serviço de gerenciamento depende do segredo.

> **Note**
>
> Os segredos gerenciados só podem ser criados pelo AWS serviço que os gerencia.

Os segredos gerenciados usam uma convenção de nomenclatura que inclui o ID do serviço de gerenciamento para ajudar a identificá-los.

```
Secret name: ServiceID!MySecret
Secret ARN : arn:aws:us-east-1:ServiceID!MySecret-a1b2c3
```

IDs para serviços que gerenciam segredos

- `appflow` – the section called “Amazon AppFlow”
- `databrew` – the section called “AWS Glue DataBrew”
- `datasync` – the section called “AWS DataSync”
- `directconnect` – the section called “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 Serviço de computação paralela”
- `rds` – the section called “Amazon RDS”
- `redshift` – the section called “banco de dados de origem”

- `sqlworkbench` – the section called “Editor de Consultas V2 do Amazon Redshift”

Para encontrar segredos gerenciados por outros AWS serviços, consulte Localizar segredos gerenciados.

Serviços da AWS que usam AWS Secrets Manager segredos

Obtenha informações sobre como cada um dos Serviços da AWS a seguir se integra ao Secrets Manager.

- Como AWS App Runner usa AWS Secrets Manager
- Como o AWS App2Container usa AWS Secrets Manager
- Como AWS AppConfig usa AWS Secrets Manager
- Como a Amazon AppFlow usa AWS Secrets Manager
- Como AWS AppSync usa AWS Secrets Manager
- Como o Amazon Athena usa AWS Secrets Manager
- Como o Amazon Aurora usa AWS Secrets Manager
- Como AWS CodeBuild usa AWS Secrets Manager
- Como o Amazon Data Firehose usa AWS Secrets Manager
- Como AWS DataSync usa AWS Secrets Manager
- Como a Amazon DataZone usa AWS Secrets Manager
- Como AWS Direct Connect usa AWS Secrets Manager
- Como AWS Directory Service usa AWS Secrets Manager
- Como o Amazon DocumentDB (com compatibilidade com o MongoDB) usa AWS Secrets Manager
- Como AWS Elastic Beanstalk usa AWS Secrets Manager
- Como o Amazon Elastic Container Registry usa AWS Secrets Manager
- Amazon Elastic Container Service
- Como a Amazon ElastiCache usa AWS Secrets Manager
- Como AWS Elemental Live usa AWS Secrets Manager
- Como AWS Elemental MediaConnect usa AWS Secrets Manager
- Como AWS Elemental MediaConvert usa AWS Secrets Manager
- Como AWS Elemental MediaLive usa AWS Secrets Manager
- Como AWS Elemental MediaPackage usa AWS Secrets Manager

- Como AWS Elemental MediaTailor usa AWS Secrets Manager
- Como o Amazon EMR usa o Secrets Manager
- Como a Amazon EventBridge usa AWS Secrets Manager
- Como a Amazon FSx usa AWS Secrets Manager segredos
- Como AWS Glue DataBrew usa AWS Secrets Manager
- Como o AWS Glue Studio usa AWS Secrets Manager
- Como AWS IoT SiteWise usa AWS Secrets Manager
- Como a Amazon Kendra usa AWS Secrets Manager
- Como o Amazon Kinesis Video Streams usa AWS Secrets Manager
- Como AWS Launch Wizard usa AWS Secrets Manager
- Como o Amazon Lookout for Metrics usa AWS Secrets Manager
- Como o Amazon Managed Grafana usa AWS Secrets Manager
- Como AWS Managed Services usa AWS Secrets Manager
- Como o Amazon Managed Streaming for Apache Kafka usa AWS Secrets Manager
- Como o Amazon Managed Workflows para Apache Airflow usa AWS Secrets Manager
- AWS Marketplace
- Como AWS Migration Hub usa AWS Secrets Manager
- Como AWS Panorama usa o Secrets Manager
- Como o serviço de computação AWS paralela usa AWS Secrets Manager
- Como AWS ParallelCluster usa AWS Secrets Manager
- Como o Amazon Q usa o Secrets Manager
- Como Amazon OpenSearch Ingestion usa o Secrets Manager
- Como AWS OpsWorks for Chef Automate usa AWS Secrets Manager
- Como o Amazon Quick usa AWS Secrets Manager
- Como o Amazon RDS usa AWS Secrets Manager
- Como o Amazon Redshift usa AWS Secrets Manager
- Editor de Consultas V2 do Amazon Redshift
- Como a Amazon SageMaker AI usa AWS Secrets Manager
- Como AWS Schema Conversion Tool usa AWS Secrets Manager
- Como o Amazon Timestream para InfluxDB usa AWS Secrets Manager

- Como AWS Toolkit for JetBrains usa AWS Secrets Manager
- Como AWS Transfer Family usa AWS Secrets Manager segredos
- Como AWS Wickr usa AWS Secrets Manager segredos

Como AWS App Runner usa AWS Secrets Manager

AWS App Runner é um AWS serviço que fornece uma maneira rápida, simples e econômica de implantar a partir do código-fonte ou de uma imagem de contêiner diretamente em um aplicativo web escalável e seguro na AWS nuvem. Você não precisa aprender novas tecnologias, decidir qual serviço de computação usar ou saber como provisionar e configurar AWS recursos.

Com o App Runner, é possível fazer referência a segredos e configurações como variáveis de ambiente em seu serviço ao criar um serviço ou atualizar a configuração do serviço. Para obter mais informações, consulte Referencing environment variables (Fazer referência a variáveis de ambiente) e Managing environment variables (Gerenciar variáveis de ambiente) no Guia do desenvolvedor do AWS App Runner .

Como o AWS App2Container usa AWS Secrets Manager

AWS App2Container é uma ferramenta de linha de comando para ajudá-lo a elevar e deslocar aplicativos executados em seus datacenters locais ou em máquinas virtuais, para que sejam executados em contêineres gerenciados pelo Amazon ECS, Amazon EKS ou. AWS App Runner

O App2Container usa o Secrets Manager para gerenciar as credenciais para conectar sua máquina de trabalho a servidores de aplicações para executar comandos remotos. Para obter mais informações, consulte Gerenciar segredos do AWS App2Container no Guia do usuário do AWS App2Container.

Como AWS AppConfig usa AWS Secrets Manager

AWS AppConfig é um recurso AWS Systems Manager que você pode usar para criar, gerenciar e implantar rapidamente configurações de aplicativos. Uma configuração pode conter dados de credenciais ou outras informações confidenciais armazenadas no Secrets Manager. Ao criar um perfil de configuração de formato livre, é possível escolher o Secrets Manager como fonte de seus dados de configuração. Para obter mais informações, consulte Criar um perfil de configuração de formato livre no Guia do usuário do AWS AppConfig . Para obter informações sobre como AWS AppConfig lidar com segredos que têm a rotação automática ativada, consulte a rotação de chaves do Secrets Manager no Guia AWS AppConfig do Usuário.

Como a Amazon AppFlow usa AWS Secrets Manager

AppFlow A Amazon é um serviço de integração totalmente gerenciado que permite que você troque dados com segurança entre aplicativos de software como serviço (SaaS), como o Salesforce, e, como o Amazon Simple Storage Service (Amazon S3) e Serviços da AWS o Amazon Redshift.

Na Amazon AppFlow, ao configurar um aplicativo SaaS como origem ou destino, você cria uma conexão. Isso inclui informações necessárias para se conectar às aplicações SaaS, como tokens de autenticação, nomes de usuário e senhas. A Amazon AppFlow armazena seus dados de conexão em um segredo gerenciado pelo Secrets Manager com o prefixo`appflow`. O custo de armazenar o segredo está incluído na cobrança da Amazon AppFlow. Para obter mais informações, consulte Proteção de dados na Amazon AppFlow no Guia AppFlow do usuário da Amazon.

Como AWS AppSync usa AWS Secrets Manager

AWS AppSync fornece uma interface GraphQL robusta e escalável para desenvolvedores de aplicativos combinarem dados de várias fontes, incluindo Amazon DynamoDB e HTTP. AWS Lambda APIs

AWS AppSync usa as credenciais em um segredo do Secrets Manager para se conectar ao Amazon RDS e ao Aurora. Para obter mais informações, consulte Tutorial: Aurora Serverless (Tutorial: Aurora Serverless) no Guia do desenvolvedor do AWS AppSync .

Como o Amazon Athena usa AWS Secrets Manager

O Amazon Athena é um serviço de consultas interativas que facilita a análise de dados diretamente no Amazon Simple Storage Service (Amazon S3) usando SQL padrão.

Os conectores da fonte de dados do Amazon Athena podem usar o atributo de consulta federada do Athena com segredos do Secrets Manager para consultar dados. Para obter mais informações, consulte Usar a consulta federada do Amazon Athena no Guia do usuário do Amazon Athena.

Como o Amazon Aurora usa AWS Secrets Manager

O Amazon Aurora é um mecanismo de banco de dados relacional totalmente gerenciado compatível com o MySQL e o PostgreSQL.

Para gerenciar as credenciais do usuário primário do Aurora, o Aurora pode criar um segredo gerenciado para você. Haverá uma cobrança por esse segredo. O Aurora também gerencia a

alternância dessas credenciais. Para obter mais informações, consulte Gerenciamento de senhas com Amazon Aurora e o AWS Secrets Manager no Guia do usuário do Amazon Aurora.

Para outras credenciais da Aurora, consulte Criar segredos.

Ao chamar a API de dados do Amazon RDS, é possível passar credenciais para o banco de dados usando um segredo no Secrets Manager. Para obter mais informações, consulte Usar a tecnologia sem servidor da API Data for Aurora no Guia do usuário do Amazon Aurora.

Ao usar o editor de consultas do Amazon RDS para se conectar a um banco de dados, é possível armazenar credenciais para o banco de dados no Secrets Manager. Para obter mais informações, consulte Usar o editor de consultas no Guia do usuário do Amazon RDS.

Como AWS CodeBuild usa AWS Secrets Manager

AWS CodeBuild é um serviço de criação totalmente gerenciado na nuvem. CodeBuild compila seu código-fonte, executa testes de unidade e produz artefatos prontos para implantação.

É possível armazenar as credenciais de registro privadas usando o Secrets Manager. Para obter mais informações, consulte Registro privado com AWS Secrets Manager amostra CodeBuild no Guia do AWS CodeBuild usuário.

Como o Amazon Data Firehose usa AWS Secrets Manager

É possível usar o Amazon Data Firehose para fornecer dados de streaming em tempo real para vários destinos de streaming. Quando o destino exige uma chave ou credenciais, o Firehose recupera um segredo do Secrets Manager em tempo de execução para se conectar ao destino. Para obter mais informações, consulte Autenticar com o Amazon Data Firehose AWS Secrets Manager no Guia do desenvolvedor do Amazon Data Firehose.

Como AWS DataSync usa AWS Secrets Manager

AWS DataSync é um serviço de transferência de dados on-line que simplifica, automatiza e acelera a movimentação de dados entre sistemas e serviços de armazenamento.

Alguns dos sistemas de armazenamento suportados pelo DataSync exigem credenciais para ler e gravar dados. DataSync usa o Secrets Manager para armazenar ou acessar as credenciais de armazenamento. Você pode configurar DataSync para criar segredos em seu nome ou fornecer um segredo personalizado. Os segredos gerenciados pelo serviço começam com o prefixo `aws-datasync`. Você é cobrado apenas pelo uso de segredos que você cria fora do DataSync. Consulte Fornecimento de credenciais para locais de armazenamento no Guia do usuário do AWS DataSync .

Como a Amazon DataZone usa AWS Secrets Manager

DataZone A Amazon é um serviço de gerenciamento de dados que permite catalogar, descobrir, controlar, compartilhar e analisar seus dados. Você pode usar ativos de dados de tabelas e visualizações de um cluster do Amazon Redshift que é rastreado usando um trabalho. Crawler do AWS Glue Para se conectar ao Amazon Redshift, você fornece as DataZone credenciais da Amazon em um segredo do Secrets Manager. Para obter mais informações, consulte Criar uma fonte de dados para um banco de dados do Amazon Redshift usando uma nova AWS Glue conexão no Guia DataZone do usuário da Amazon.

Como AWS Direct Connect usa AWS Secrets Manager

Direct Connect conecta sua rede interna a um Direct Connect local por meio de um cabo de fibra óptica Ethernet padrão. Com essa conexão, você pode criar interfaces virtuais diretamente para o público Serviços da AWS.

Direct Connect armazena um nome de chave de associação de conectividade e um par de chaves de associação de conectividade (par CKN/CAK) em um segredo gerenciado com o prefixo. `directconnect` O custo do segredo está incluído na cobrança do Direct Connect. Para atualizar o segredo, você deve usar Direct Connect em vez do Secrets Manager. Para obter mais informações, consulte Associar a MACsec CKN/CAK a a um LAG no Guia do Direct Connect usuário.

Como AWS Directory Service usa AWS Secrets Manager

Directory Service fornece várias maneiras de usar o Microsoft Active Directory (AD) com outros AWS serviços. É possível unir uma instância do Amazon EC2 ao seu diretório usando segredos para credenciais. Para obter mais informações, no Guia do usuário do Direct Connect , consulte:

- Associe perfeitamente uma instância Linux EC2 ao seu diretório gerenciado AWS do Microsoft AD
- Ingressar perfeitamente uma instância do EC2 do Linux ao seu diretório do AD Connector
- Ingressar perfeitamente uma instância do EC2 do Linux ao seu diretório do Simple AD

Como o Amazon DocumentDB (com compatibilidade com o MongoDB) usa AWS Secrets Manager

O Amazon DocumentDB (compatível com MongoDB) é um serviço de banco de dados de documentos totalmente gerenciado que oferece suporte a workloads do MongoDB. O Amazon

DocumentDB se integra ao Secrets Manager para gerenciar senhas de usuário primárias para seus clusters, aprimorando a segurança e simplificando o gerenciamento de credenciais.

O Amazon DocumentDB gera a senha, a armazena no Secrets Manager e gerencia as configurações do segredo. Por padrão, o Amazon DocumentDB alterna o segredo a cada sete dias, mas é possível modificar o cronograma de alternância, se necessário. Ao criar ou modificar um cluster do Amazon DocumentDB, é possível especificar que ele gerencie a senha do usuário principal no Secrets Manager. Para obter mais informações, consulte Gerenciamento de senhas com o Amazon DocumentDB e o Secrets Manager no Guia do desenvolvedor do Amazon DocumentDB.

Como AWS Elastic Beanstalk usa AWS Secrets Manager

Com AWS Elastic Beanstalk, você pode implantar e gerenciar rapidamente aplicativos na AWS nuvem sem precisar aprender sobre a infraestrutura que executa esses aplicativos. O Elastic Beanstalk pode iniciar ambientes do Docker compilando uma imagem descrita em um Dockerfile ou usando uma imagem do Docker remota. Para se autenticar com o registro on-line que hospeda o repositório privado, o Elastic Beanstalk usa um segredo do Secrets Manager. Para obter mais informações, consulte Configuração do Docker no AWS Elastic Beanstalk Guia do desenvolvedor.

Como o Amazon Elastic Container Registry usa AWS Secrets Manager

O Amazon Elastic Container Registry (Amazon ECR) é AWS um serviço gerenciado de registro de imagens de contêineres que é seguro, escalável e confiável. É possível usar a CLI do Docker, ou seu cliente preferido, para enviar e extrair imagens dos seus repositórios. Para cada registro upstream que contém imagens que você deseja armazenar em cache em seu registro privado do Amazon ECR, é necessário criar uma regra de cache de pull-through. Para registros de upstream que exigem autenticação, é necessário armazenar as credenciais em um segredo do Secrets Manager. É possível criar o segredo do Secrets Manager nos consoles Amazon ECR ou Secrets Manager. Para obter mais informações, consulte Criação de uma regra de cache de pull-through no Guia do usuário do Amazon ECR.

Amazon Elastic Container Service

O Amazon Elastic Container Service (Amazon ECS) é um serviço totalmente gerenciado de orquestração de contêineres ajuda a implantar, gerenciar e dimensionar facilmente aplicações conteinerizadas. É possível injetar dados confidenciais em seus contêineres fazendo referência aos segredos do Secrets Manager. Para obter mais informações, consulte as seguintes páginas no Guia do desenvolvedor do Amazon Elastic Container Service:

- Tutorial: Especificar dados sigilosos usando segredos do Secrets Manager
- Recuperar segredos programaticamente por meio da sua aplicação
- Recuperar segredos por meio de variáveis de ambiente
- Recuperar segredos para a configuração de registro em log

O Amazon ECS oferece suporte FSx a volumes do Windows File Server para contêineres. O Amazon ECS usa as credenciais armazenadas em um segredo do Secrets Manager para ingressar no domínio do Active Directory e anexar o sistema FSx de arquivos do Windows File Server. Para obter mais informações, consulte Tutorial: Uso FSx para sistemas de arquivos do Windows File Server com o Amazon ECS e FSx para volumes do Windows File Server no Amazon Elastic Container Service Developer Guide.

Você pode referenciar imagens de contêiner em registros privados AWS que não exijam autenticação usando um segredo do Secrets Manager com as credenciais do registro. Para obter mais informações, consulte Autenticação de registro privado para tarefas no Guia do desenvolvedor do Amazon Elastic Container Service.

Quando você usa o Amazon ECS Service Connect, o Amazon ECS usa segredos gerenciados do Secrets Manager para armazenar certificados Autoridade de Certificação Privada da AWS TLS. O custo de armazenamento do segredo está incluído na cobrança pelo Amazon ECS. Para atualizar o segredo, é necessário usar o Amazon ECS em vez do Secrets Manager. Para obter mais informações, consulte TLS com o Service Connect no Guia do desenvolvedor do serviço Amazon Elastic Container.

Como a Amazon ElastiCache usa AWS Secrets Manager

Em ElastiCache você pode usar um recurso chamado Controle de Acesso Baseado em Funções (RBAC) para proteger o cluster. É possível armazenar essas credenciais no Secrets Manager. O Secrets Manager fornece um modelo de alternância para esse tipo de segredo. Para obter mais informações, consulte Rotação automática de senhas para usuários no Guia do ElastiCache usuário da Amazon.

Como AWS Elemental Live usa AWS Secrets Manager

AWS Elemental Live é um serviço de vídeo em tempo real que permite criar saídas ao vivo para transmissão e entrega de streaming.

AWS Elemental Live usa um ARN secreto para obter um segredo que contém uma chave de criptografia do Secrets Manager. O Elemental Live usa a chave de criptografia encrypt/decrypt do vídeo. Para obter mais informações, consulte Como a entrega de AWS Elemental Live para MediaConnect funciona em tempo de execução no Guia do usuário do Elemental Live.

Como AWS Elemental MediaConnect usa AWS Secrets Manager

AWS Elemental MediaConnect é um serviço que torna mais fácil para emissoras e outros provedores de vídeo premium ingerir vídeo ao vivo de forma confiável Nuvem AWS e distribuí-lo para vários destinos dentro ou fora do. Nuvem AWS

É possível usar criptografia de chave estática para proteger suas fontes, saídas e direitos e armazenar sua chave de criptografia no AWS Secrets Manager. Para obter mais informações, consulte Criptografia de chave estática no AWS Elemental MediaConnect no Guia do usuário do AWS Elemental MediaConnect .

Como AWS Elemental MediaConvert usa AWS Secrets Manager

AWS Elemental MediaConvert é um serviço de processamento de vídeo baseado em arquivos que fornece processamento de vídeo escalável para proprietários e distribuidores de conteúdo com bibliotecas de mídia de qualquer tamanho. Para MediaConvert codificar as marcas d'água da Kantar, você usa o Secrets Manager para armazenar suas credenciais da Kantar. Para obter mais informações, consulte Usando o Kantar para marcas d'água de áudio nas AWS Elemental MediaConvert saídas no Guia do AWS Elemental MediaConvert usuário.

Como AWS Elemental MediaLive usa AWS Secrets Manager

AWS Elemental MediaLive é um serviço de vídeo em tempo real que permite criar saídas ao vivo para transmissão e entrega de streaming. Se sua organização usa AWS Elemental Link dispositivos com AWS Elemental MediaLive ou AWS Elemental MediaConnect, você deve implantar o dispositivo e configurá-lo. Para obter mais informações, consulte Configurando MediaLive como uma entidade confiável no Guia do MediaLive usuário.

Como AWS Elemental MediaPackage usa AWS Secrets Manager

AWS Elemental MediaPackage é um serviço de embalagem e originação de just-in-time vídeos executado no Nuvem AWS. Com MediaPackage, você pode fornecer streams de vídeo altamente seguros, escaláveis e confiáveis para uma ampla variedade de dispositivos de reprodução e redes

de distribuição de conteúdo (). CDNs Para obter mais informações, consulte Acesso do Secrets Manager para autorização de CDN no Guia do usuário do AWS Elemental MediaPackage .

Como AWS Elemental MediaTailor usa AWS Secrets Manager

AWS Elemental MediaTailor é um serviço escalável de inserção de anúncios e montagem de canais executado no. Nuvem AWS

MediaTailor suporta a autenticação de token de acesso do Secrets Manager em seus locais de origem. Com a autenticação do token de acesso do Secrets Manager, MediaTailor usa um segredo do Secrets Manager para autenticar solicitações em sua origem. Para obter mais informações, consulte Configurando a autenticação do token de AWS Secrets Manager acesso no Guia do AWS Elemental MediaTailor usuário.

Como o Amazon EMR usa o Secrets Manager

O Amazon EMR é uma plataforma que simplifica a execução de estruturas de big data, como Apache Hadoop e Apache Spark, para processar e analisar grandes quantidades de dados. AWS Ao usar essas estruturas e projetos de código aberto relacionados, como Apache Hive e Apache Pig, é possível processar dados para análise e workloads de business intelligence. Você também pode usar o Amazon EMR para transformar e mover grandes quantidades de dados para dentro e para fora de outros bancos de dados e bancos de AWS dados, como o Amazon S3 e o Amazon DynamoDB.

Como o Amazon EMR executado no Amazon EC2 usa o Secrets Manager

Ao criar um cluster no Amazon EMR, é possível fornecer dados de configuração de aplicações usando um segredo no Secrets Manager. Para obter mais informações, consulte Armazenar dados de configuração confidenciais no Secrets Manager no Guia de gerenciamento do Amazon EMR.

Além disso, quando você cria um bloco de anotações do EMR, é possível armazenar as credenciais de registro privadas baseadas no Git usando o Secrets Manager. Para obter mais informações, consulte Add a Git-based Repository to Amazon EMR (Adicionar um repositório baseado em Git ao Amazon EMR) no Guia de gerenciamento do Amazon EMR.

Como o EMR Serverless usa o Secrets Manager

O EMR Serverless fornece um ambiente de runtime com tecnologia sem servidor para simplificar a operação de aplicações de análise para que você não precise configurar, otimizar, proteger ou operar clusters.

Você pode armazenar seus dados AWS Secrets Manager e depois usar o ID secreto em suas configurações do EMR Serverless. Dessa forma, você não passa dados de configuração confidenciais em texto simples e os expõe ao externo APIs.

Para obter mais informações, consulte Secrets Manager para proteção de dados com o EMR Serverless no Guia do usuário do Amazon EMR Serverless.

Como a Amazon EventBridge usa AWS Secrets Manager

EventBridge A Amazon é um serviço de barramento de eventos sem servidor que você pode usar para conectar seus aplicativos a dados de várias fontes.

Quando você cria um destino de EventBridge API da Amazon, EventBridge armazena a conexão para ele em um segredo gerenciado do Secrets Manager com o prefixo`events`. O custo de armazenamento do segredo está incluído na cobrança pelo uso de um destino de API. Para atualizar o segredo, você deve usar EventBridge em vez do Secrets Manager. Para obter mais informações, consulte Destinos de API no Guia EventBridge do usuário da Amazon.

Como a Amazon FSx usa AWS Secrets Manager segredos

O Amazon FSx para Windows File Server fornece servidores de arquivos Microsoft Windows totalmente gerenciados, apoiados por um sistema de arquivos Windows totalmente nativo. Ao criar ou gerenciar compartilhamentos de arquivos, você pode passar credenciais de um AWS Secrets Manager segredo. Para obter mais informações, consulte Compartilhamentos de arquivos e Migração de configurações de compartilhamento de arquivos para a Amazon FSx no Guia do usuário do Amazon FSx para Windows File Server.

Como AWS Glue DataBrew usa AWS Secrets Manager

AWS Glue DataBrew é uma ferramenta visual de preparação de dados que você pode usar para limpar e normalizar dados sem escrever nenhum código. Em DataBrew, um conjunto de etapas de transformação de dados é chamado de receita. AWS Glue DataBrew fornece as etapas de `DETERMINISTIC_DECRYPT`DETERMINISTIC_ENCRYPT, e `CRYPTOGRAPHIC_HASH`receita para realizar transformações em informações de identificação pessoal (PII) em um conjunto de dados, que usa uma chave de criptografia armazenada em um segredo do Secrets Manager. Se você usar o segredo DataBrew padrão para armazenar a chave de criptografia, DataBrew cria um segredo gerenciado com o prefixo`databrew`. O custo de armazenar o segredo está incluído na taxa de uso DataBrew. Se você criar um novo segredo para armazenar a chave de criptografia, DataBrew cria um segredo com o prefixo`AwsGlueDataBrew`. Haverá uma cobrança por esse segredo.

Como o AWS Glue Studio usa AWS Secrets Manager

AWS Glue Studio é uma interface gráfica que facilita a criação, execução e monitoramento de trabalhos de extração, transformação e carregamento (ETL) em AWS Glue. Você pode usar o Amazon OpenSearch Service como um armazenamento de dados para suas tarefas de extração, transformação e carregamento (ETL) configurando o Elasticsearch Spark Connector no. AWS Glue Studio Para se conectar ao OpenSearch cluster, você pode usar um segredo no Secrets Manager. Para obter mais informações, consulte Tutorial: Usando o conector Glue para Elasticsearch do AWS no Guia do desenvolvedor do AWS Glue .

Como AWS IoT SiteWise usa AWS Secrets Manager

AWS IoT SiteWise é um serviço gerenciado que permite coletar, modelar, analisar e visualizar dados de equipamentos industriais em grande escala. Você pode usar o AWS IoT SiteWise console para criar um gateway. Em seguida, adicione origens dos dados, servidores locais ou equipamentos industriais conectados a gateways. Se a sua origem exigir autenticação, use um segredo para autenticar. Para obter mais informações, consulte Configuring data source authentication (Configurar a autenticação da fonte de dados) Guia do usuário do AWS IoT SiteWise .

Como a Amazon Kendra usa AWS Secrets Manager

O Amazon Kendra é um serviço de pesquisa altamente preciso e inteligente que habilita seus usuários a pesquisarem dados não estruturados e estruturados usando processamento de linguagem natural e algoritmos de pesquisa avançados.

É possível indexar documentos armazenados em um banco de dados especificando um segredo que contenha credenciais para o banco de dados. Para obter mais informações, consulteUsing a database data source (Usar uma fonte de dados de banco de dados) no Guia do usuário do Amazon Kendra.

Como o Amazon Kinesis Video Streams usa AWS Secrets Manager

É possível usar o Amazon Kinesis Video Streams para se conectar a câmeras IP nas dependências do cliente, gravar e armazenar localmente vídeos das câmeras e transmitir vídeos para a nuvem para fins de armazenamento, reprodução e processamento analítico de longo prazo. Para gravar e carregar mídia de câmeras IP, é necessário implantar o Kinesis Video Streams Edge Agent no AWS IoT Greengrass. Você armazena as credenciais necessárias para acessar os arquivos de mídia que são transmitidos para a câmera em um segredo do Secrets Manager. Para obter mais informações,

consulte Implantar o Amazon Kinesis Video Streams Edge Agent no AWS IoT Greengrass no Guia do desenvolvedor do Amazon Kinesis Video Streams.

Como AWS Launch Wizard usa AWS Secrets Manager

AWS Launch Wizard for Active Directory é um serviço que aplica as melhores práticas de Nuvem AWS aplicativos para orientá-lo na configuração de uma nova infraestrutura do Active Directory ou na adição de controladores de domínio a uma infraestrutura existente, no local Nuvem AWS ou no local.

AWS Launch Wizard exige que as credenciais do administrador de domínio sejam adicionadas ao Secrets Manager para unir seus controladores de domínio ao Active Directory. Para obter mais informações, consulte Set up for AWS Launch Wizard for Active Directory (Configuração do para Active Directory) no Guia do usuário do AWS Launch Wizard .

Como o Amazon Lookout for Metrics usa AWS Secrets Manager

O Amazon Lookout for Metrics é um serviço que localiza anomalias nos seus dados, determina suas causas raiz e permite tomar medidas rapidamente. O Amazon Redshift ou o Amazon RDS podem ser usados como fontes de dados para um detector do Lookout for Metrics. Para configurar a fonte de dados, você usa um segredo que contém a senha do banco de dados. Para obter mais informações, consulte Using Amazon RDS with Lookout for Metrics (Uso do Amazon RDS com o Lookout for Metrics) e Using Amazon Redshift with Lookout for Metrics (Uso do Amazon Redshift com o Lookout for Metrics) no Guia do desenvolvedor do Amazon Lookout for Metrics.

Como o Amazon Managed Grafana usa AWS Secrets Manager

O Amazon Managed Grafana é um serviço de visualização de dados totalmente gerenciado e seguro que pode ser usado para consultar, correlacionar e visualizar instantaneamente métricas operacionais, logs e rastreamentos de várias fontes. Ao usar o Amazon Redshift como fonte de dados, você pode fornecer credenciais do Amazon Redshift usando um segredo. AWS Secrets Manager Para obter mais informações, consulte Configurando Amazon Redshift no Guia de ususário do Amazon Managed Grafana.

Como AWS Managed Services usa AWS Secrets Manager

AWS Managed Services é um serviço corporativo que fornece gerenciamento contínuo de sua AWS infraestrutura. O modo AMS Self-Service Provisioning (SSP) fornece acesso total aos recursos nativos AWS service (Serviço da AWS) e de API nas contas gerenciadas pelo AMS. Para obter

informações sobre como solicitar acesso ao Secrets Manager no AMS, consulte AWS Secrets Manager (provisionamento de autoatendimento do AMS) no Guia avançado do usuário do AMS.

Como o Amazon Managed Streaming for Apache Kafka usa AWS Secrets Manager

O Amazon Managed Streaming for Apache Kafka (Amazon MSK) é um serviço totalmente gerenciado que o habilita a criar e executar aplicações que usam o Apache Kafka para processar dados de transmissões. É possível controlar o acesso aos seus clusters do Amazon MSK usando nomes de usuário e senhas armazenados e protegidos com o AWS Secrets Manager. Para obter mais informações, consulte Autenticação de nome de usuário e senha com AWS Secrets Manager no Guia do desenvolvedor do Amazon Managed Streaming for Apache Kafka.

Como o Amazon Managed Workflows para Apache Airflow usa AWS Secrets Manager

O Amazon Managed Workflows for Apache Airflow é um serviço gerenciado de orquestração para o Apache Airflow que facilita a configuração e a operação de pipelines de dados na nuvem em grande escala. end-to-end

É possível configurar uma conexão do Apache Airflow usando um segredo do Secrets Manager. Para obter mais informações, consulte Configurando uma conexão do Apache Airflow usando um segredo do Secrets Manager e Usando uma chave AWS Secrets Manager secreta para uma variável do Apache Airflow no Guia do usuário do Amazon Managed Workflows for Apache Airflow.

AWS Marketplace

Quando você usa o AWS Marketplace Quick Launch, AWS Marketplace distribui seu software junto com a chave de licença. AWS Marketplace armazena a chave de licença em sua conta como um segredo gerenciado pelo Secrets Manager. O custo de armazenar o segredo está incluído nas cobranças de AWS Marketplace. Para atualizar o segredo, você deve usar AWS Marketplace em vez do Secrets Manager. Para obter mais informações, consulte Configuração rápida no Guia do vendedor do AWS Marketplace .

Como AWS Migration Hub usa AWS Secrets Manager

AWS Migration Hub fornece um único local para rastrear as tarefas de migração em várias AWS ferramentas e soluções de parceiros.

AWS Migration Hub O Orchestrator simplifica e automatiza a migração de servidores e aplicativos corporativos para o. AWS O Migration Hub Orchestrator usa um segredo para as informações de conexão com o servidor de origem. Para obter mais informações, no Guia do usuário do AWS Migration Hub Orchestrator, consulte:

- Migre NetWeaver aplicativos SAP para AWS
- Redefinir a hospedagem de aplicações no Amazon EC2

O Migration Hub Strategy Recommendations oferece recomendações de estratégia de migração e modernização para caminhos de transformação viáveis para suas aplicações. O Strategy Recommendations pode analisar bancos de dados do SQL Server, usando um segredo para as informações de conexão. Para obter mais informações, consulte Análise de banco de dados do Strategy Recommendations.

Como AWS Panorama usa o Secrets Manager

AWS Panorama é um serviço que traz visão computacional para sua rede de câmeras local. Você usa AWS Panorama para registrar um equipamento, atualizar seu software e implantar aplicativos nele. Quando você registra um stream de vídeo como fonte de dados para sua aplicação, se o stream estiver protegido por senha, o AWS Panorama armazena as credenciais dele em um segredo do Secrets Manager. Para obter mais informações, consulte Gerenciar fluxo de câmeras no AWS Panorama no AWS Panorama Guia do desenvolvedor .

Como o serviço de computação AWS paralela usa AWS Secrets Manager

AWS O Serviço de Computação Paralela (AWS PCS) é um serviço gerenciado que facilita a execução e a escalabilidade de cargas de trabalho de computação de alto desempenho (HPC) e de aprendizado de máquina distribuído. AWS

Para se conectar ao agendador de tarefas do cluster, o AWS PCS cria um segredo gerenciado com o prefixo `pcs` para armazenar a chave do agendador. O custo de armazenar o segredo está incluído na cobrança do AWS PCS. AWS O PCS exclui automaticamente o segredo quando você exclui seu cluster AWS PCS. Para obter mais informações, consulte Trabalhando com segredos de cluster no AWS PCS no Guia do usuário do AWS PCS.

> **Important**
>
> Não modifique nem exclua os segredos do cluster AWS PCS.

Como AWS ParallelCluster usa AWS Secrets Manager

AWS ParallelCluster é uma ferramenta de gerenciamento de cluster de código aberto que você pode usar para implantar e gerenciar clusters de computação de alto desempenho (HPC) no Nuvem AWS. Você pode criar um ambiente de vários usuários que inclua um AWS ParallelCluster que esteja integrado a um Microsoft AD AWS gerenciado (Active Directory). O AWS ParallelCluster usa um segredo do Secrets Manager para validar logins no Active Directory. Para obter mais informações, consulte Integrar o Active Directory no Guia do usuário do AWS ParallelCluster .

Como o Amazon Q usa o Secrets Manager

Para autenticar o Amazon Q para acessar sua fonte de dados, você fornece suas credenciais de acesso à fonte de dados para o Amazon Q usando um segredo do Secrets Manager. Se você usa o console, pode optar por criar um novo segredo ou usar um segredo já existente. Para obter mais informações, consulte Conceitos: autenticação no Guia do Amazon Q Developer.

Como Amazon OpenSearch Ingestion usa o Secrets Manager

Amazon OpenSearch Ingestioné um coletor de dados totalmente gerenciado e sem servidor que transmite registros, métricas e dados de rastreamento em tempo real para domínios e coleções do Amazon OpenSearch Service. OpenSearch Serverless Você pode usar OpenSearch Ingestion pipelines com o Secrets Manager para gerenciar suas credenciais com segurança. Para obter mais informações, consulte:

- Usando um OpenSearch Ingestion pipeline com Atlassian Services
- Usando um OpenSearch Ingestion pipeline com o Amazon DocumentDB
- Usando um OpenSearch Ingestion pipeline com Confluent Cloud Kafka
- Usando um OpenSearch Ingestion pipeline com Kafka
- Migração de dados de clusters autogerenciados usando OpenSearch Amazon OpenSearch Ingestion

Como AWS OpsWorks for Chef Automate usa AWS Secrets Manager

OpsWorks é um serviço de gerenciamento de configuração que ajuda você a configurar e operar aplicativos em uma empresa em nuvem usando o OpsWorks Puppet Enterprise ou AWS OpsWorks for Chef Automate.

Quando você cria um novo servidor em AWS OpsWorks CM, o OpsWorks CM armazena informações do servidor em um segredo gerenciado do Secrets Manager com o prefixo`opsworks-cm`. O custo do segredo está incluído na cobrança pelo OpsWorks. Para obter mais informações, consulte Integration with AWS Secrets Manager (Integração com o) no Guia do usuário do OpsWorks .

Como o Amazon Quick usa AWS Secrets Manager

O Amazon Quick é um serviço de inteligência de negócios (BI) em escala de nuvem que você pode usar para análises, visualização de dados e relatórios. Você pode usar várias fontes de dados no Quick. Se você armazenar as credenciais do banco de dados nos segredos do Secrets Manager, o Quick poderá usar esses segredos para se conectar aos bancos de dados. Para obter mais informações, consulte Uso de AWS Secrets Manager segredos no lugar de credenciais de banco de dados no Amazon Quick no Guia do usuário do Amazon Quick.

Como o Amazon RDS usa AWS Secrets Manager

O Amazon Relational Database Service (Amazon RDS) é um serviço da Web que facilita a configuração, a operação e escalabilidade de um banco de dados relacional na Nuvem AWS.

Para gerenciar credenciais de usuário primário do Amazon Relational Database Service (Amazon RDS), incluindo o Aurora, o Amazon RDS pode criar um segredo gerenciado para você. Haverá uma cobrança por esse segredo. O Amazon RDS também gerencia a alternância dessas credenciais. Para obter mais informações, consulte Gerenciamento de senhas com Amazon RDS e o AWS Secrets Manager no Guia do usuário do Amazon RDS.

Para outras credenciais do Amazon RDS, consulte Criar segredos.

Ao usar o editor de consultas do Amazon RDS para se conectar a um banco de dados, é possível armazenar credenciais para o banco de dados no Secrets Manager. Para obter mais informações, consulte Usar o editor de consultas no Guia do usuário do Amazon RDS.

Como o Amazon Redshift usa AWS Secrets Manager

O Amazon Redshift é um serviço de data warehouse totalmente gerenciado e em escala de petabytes na Nuvem .

Para gerenciar as credenciais de administrador do Amazon Redshift, o Amazon Redshift pode criar um segredo gerenciado para você. Haverá uma cobrança por esse segredo. O Amazon

Redshift também gerencia a alternância dessas credenciais. Para obter mais informações, consulte Gerenciamento de senhas de administrador do Amazon Redshift usando o Guia de gerenciamento AWS Secrets Manager de clusters do Amazon Redshift.

Para outras credenciais do Amazon Redshift, consulte Criar segredos.

Ao chamar a API de dados do Amazon Redshift, é possível passar credenciais para o cluster usando um segredo no Secrets Manager. Para obter mais informações, consulte Uso da API de dados do Amazon Redshift.

Quando você usa o editor de consultas do Amazon Redshift para se conectar a um banco de dados, o Amazon Redshift pode armazenar suas credenciais em um segredo do Secrets Manager com o prefixo `redshiftqueryeditor`. Haverá uma cobrança por esse segredo. Para obter mais informações, consulte Consultar um banco de dados usando o editor de consultas no Guia de gerenciamento do Amazon Redshift.

Para o editor de consultas v2, consulte the section called “Editor de Consultas V2 do Amazon Redshift”.

Editor de Consultas V2 do Amazon Redshift

O editor de consultas v2 do Amazon Redshift é uma aplicação de cliente SQL baseada na Web que pode ser usada para criar e executar consultas no data warehouse do Amazon Redshift. Quando você usa o Editor de Consultas v2 do Amazon Redshift para se conectar a um banco de dados, o Amazon Redshift pode armazenar suas credenciais em um segredo gerenciado do Secrets Manager com o prefixo `sqlworkbench`. O custo de armazenamento do segredo está incluído na cobrança pelo uso do Amazon Redshift. Para atualizar o segredo, é necessário usar o Amazon Redshift em vez do Secrets Manager. Para obter mais informações, consulte Trabalhar com o editor de consultas v2 no Guia de gerenciamento do Amazon Redshift.

Para a versão anterior do editor de consultas, consulte the section called “banco de dados de origem”.

Como a Amazon SageMaker AI usa AWS Secrets Manager

SageMaker A IA é um serviço de aprendizado de máquina totalmente gerenciado. Com a SageMaker IA, cientistas de dados e desenvolvedores podem criar e treinar modelos de aprendizado de máquina com rapidez e facilidade e, em seguida, implantá-los diretamente em um ambiente hospedado pronto para produção. O serviço oferece uma instância de notebook de autoria Jupyter integrado para

facilitar o acesso a fontes de dados para fins de exploração e análise, sem necessidade de gerenciar servidores.

É possível associar repositórios do Git a suas instâncias do caderno Jupyter para salvar seus cadernos de anotações em um ambiente de controle de origem que persista mesmo se você interromper ou excluir sua instância de bloco de anotações. É possível gerenciar suas credenciais de repositórios privados usando o Secrets Manager. Para obter mais informações, consulte Associar repositórios Git às instâncias do Amazon SageMaker Notebook no Amazon SageMaker AI Developer Guide.

Para importar dados do Databricks, o Data Wrangler armazena sua URL do JDBC no Secrets Manager. Para obter mais informações, consulte Importar dados do Databricks (JDBC).

Para importar dados do snowflake, o Data Wrangler armazena suas credenciais em um segredo do Secrets Manager. Para obter mais informações, consulte Importar dados do Snowflake.

Como AWS Schema Conversion Tool usa AWS Secrets Manager

Você pode usar o AWS Schema Conversion Tool (AWS SCT) para converter seu esquema de banco de dados existente de um mecanismo de banco de dados para outro. É possível converter o esquema OLTP relacional ou o esquema de data warehouse. Seu esquema convertido é adequado para um Amazon Relational Database Service (Amazon RDS) MySQL, MariaDB, Oracle, SQL Server, PostgreSQL DB, um cluster Amazon Aurora DB ou um cluster Amazon Redshift. O esquema convertido também pode ser usado com um banco de dados em uma instância do Amazon Elastic Compute Cloud ou armazenado como dados em um bucket do S3.

Quando você converte um esquema de banco de dados, AWS SCT pode usar as credenciais do banco de dados que você armazena. AWS Secrets Manager Para obter mais informações, consulte Usando AWS Secrets Manager na interface AWS SCT do usuário no Guia do AWS Schema Conversion Tool usuário.

Como o Amazon Timestream para InfluxDB usa AWS Secrets Manager

O Timestream for InfluxDB é um mecanismo gerenciado de banco de dados de séries temporais que facilita a execução de bancos de dados do InfluxDB para aplicativos de séries temporais em AWS tempo real usando código aberto. APIs Com o Timestream para InfluxDB, é possível configurar, operar e escalar workloads de séries temporais que podem responder a consultas com tempo de resposta de consulta de um dígito de milissegundos.

Quando você cria um banco de dados do Timestream para o InfluxDB, o Timestream cria automaticamente um segredo para armazenar as credenciais de administrador. Para obter mais informações, consulte Como o Timestream para InfluxDB usa segredos no Guia do desenvolvedor do Timestream.

Como AWS Toolkit for JetBrains usa AWS Secrets Manager

O AWS Toolkit for JetBrains é um plugin de código aberto para os ambientes de desenvolvimento integrados (IDEs) de JetBrains. Esse toolkit facilita o desenvolvimento, a depuração e a implantação de aplicações sem servidor para desenvolvedores que usam AWS. Ao conectar a um cluster do Amazon Redshift usando o toolkit, é possível se autenticar usando um segredo do Secrets Manager. Para obter mais informações, consulte Accessing Amazon Redshift clusters (Acesso a clusters do Amazon Redshift) no Guia do usuário do AWS Toolkit for JetBrains .

Como AWS Transfer Family usa AWS Secrets Manager segredos

AWS Transfer Family é um serviço de transferência segura que permite transferir arquivos para dentro e para fora dos serviços de AWS armazenamento.

O Transfer Family agora oferece suporte ao uso da autenticação básica para servidores que usam o protocolo Applicability Statement 2 (AS2). É possível criar um novo segredo do Secrets Manager ou escolher um segredo existente para suas credenciais. Para obter mais informações, consulte Autenticação básica para AS2 conectores no Guia do AWS Transfer Family usuário.

Para autenticar usuários do Transfer Family, você pode usar AWS Secrets Manager como provedor de identidade. Para obter mais informações, consulte Trabalhando com provedores de identidade personalizados no Guia do AWS Transfer Family usuário e no artigo do blog Habilitar a autenticação por senha para AWS Transfer Family uso AWS Secrets Manager.

É possível usar a decodificação Pretty Good Privacy (PGP) com os arquivos que o Transfer Family processa com fluxos de trabalho. Para usar a descriptografia em uma etapa do fluxo de trabalho, você fornece uma chave PGP que gerencia no Secrets Manager. Para obter mais informações, consulte Gerar e gerenciar chaves PGP no Guia do usuário do AWS Transfer Family .

Como AWS Wickr usa AWS Secrets Manager segredos

AWS Wickr é um serviço end-to-end criptografado que ajuda organizações e agências governamentais a se comunicarem com segurança por meio one-to-one de mensagens em grupo, chamadas de voz e vídeo, compartilhamento de arquivos, compartilhamento de tela e muito mais.

É possível automatizar fluxos de trabalho usando bots de retenção de dados Wickr. Se o bot tiver acesso Serviços da AWS, você deverá criar um segredo do Secrets Manager para armazenar as credenciais do bot. Para obter mais informações, consulte Início do bot de retenção de dados no Guia de administração do AWS Wickr .

Usando segredos externos AWS Secrets Manager gerenciados para gerenciar segredos de terceiros

Segredos externos gerenciados são um novo tipo de segredo AWS Secrets Manager que permite armazenar e alternar automaticamente as credenciais dos parceiros de integração. Esse recurso elimina a necessidade de criar e manter AWS Lambda funções personalizadas para rotatividade de segredos de parceiros de integração. Para obter uma lista completa de todos os parceiros integrados, consulte Parceiros de integração.

Quando você cria aplicativos AWS, suas cargas de trabalho geralmente precisam interagir com aplicativos de terceiros por meio de credenciais seguras, como chaves de API, OAuth tokens ou pares de credenciais. Anteriormente, era necessário desenvolver abordagens personalizadas para proteger e gerenciar essas credenciais, incluindo a criação de funções Lambda de rotação complexas que eram exclusivas para cada aplicativo e exigiam manutenção contínua.

Segredos externos gerenciados fornecem uma abordagem padronizada para armazenar credenciais de terceiros em um formato predefinido prescrito por cada parceiro. O recurso inclui rotação automática que é ativada (por padrão no console) durante a criação de segredos, transparência total e controles de usuário para fluxos de trabalho de gerenciamento de segredos e o conjunto completo de recursos oferecido pelo Secrets Manager, incluindo gerenciamento refinado de permissões, observabilidade, governança, conformidade, recuperação de desastres e controles de monitoramento.

Recursos principais do

Os segredos externos gerenciados oferecem vários recursos importantes que simplificam o gerenciamento de credenciais de terceiros:

- A rotação gerenciada sem Lambda elimina a sobrecarga de criar e gerenciar funções de rotação personalizadas. Quando você cria um externo, a rotação é ativada automaticamente sem nenhuma função Lambda implantada em sua conta.
- Os formatos secretos predefinidos garantem que os segredos possam ser associados adequadamente ao parceiro de integração e incluam os metadados necessários para a rotação. Cada parceiro define o formato necessário.
- O ecossistema integrado de parceiros fornece suporte para vários parceiros por meio de um processo de integração padronizado. Os parceiros se integram diretamente ao Secrets

Manager para oferecer orientação programática para criação de segredos e recursos de rotação gerenciada.

- A auditabilidade completa mantém a transparência total por meio do AWS CloudTrail registro de todas as atividades de rotação, atualizações secretas de valores e operações de gerenciamento.

Segredos externos gerenciados Parceiros

O Secrets Manager se integra nativamente a aplicativos de terceiros para alternar os segredos mantidos pelo parceiro. Cada parceiro define os campos de metadados e valores secretos necessários para alternar os segredos.

O valor secreto contém campos que são necessários para se conectar com seu cliente terceirizado e são armazenados durante a CreateSecretchamada. Os metadados de rotação contêm os campos que são usados para atualizar o segredo durante a rotação e são usados na RotateSecretchamada. Esses campos serão definidos pelo parceiro de integração para permitir fluxos de rotação gerenciados.

Para que a rotação funcione corretamente, você deve fornecer ao Secrets Manager permissões específicas para gerenciar o ciclo de vida secreto. Para obter mais informações, consulte Segurança e permissões

Os tópicos a seguir incluem uma descrição de cada um dos campos de metadados necessários para alternar o segredo, bem como uma descrição de cada um dos campos obrigatórios no segredo do Secrets Manager para alternar.

Tópicos

Parceiro de integração	Tipo de segredo
Salesforce	SalesforceClientSecret
BigID	Grande IDClient segredo
Snowflake	SnowflakeKeyPairAuthentication

Segredo do cliente Salesforce

Campos de valor secreto

A seguir estão os campos que devem estar contidos no segredo do Secrets Manager:

```
{
  "consumerKey": "client ID",
  "consumerSecret": "client secret",
  "baseUri": "https://domain.my.salesforce.com",
  "appId": "app ID",
  "consumerId": "consumer ID"
}
```

consumerKey

A chave do consumidor, também conhecida como ID do cliente, é o identificador de credencial para as credenciais OAuth 2.0. Você pode recuperar a chave do consumidor diretamente das configurações do Salesforce External Client App Manager. OAuth

consumerSecret

O segredo do consumidor, também conhecido como segredo do cliente, é a senha privada usada com a chave do consumidor para autenticar usando o fluxo de credenciais do cliente OAuth 2.0. Você pode recuperar o segredo do consumidor diretamente das configurações do Salesforce External Client App Manager. OAuth

baseUri

O URI base é o URL base da sua organização do Salesforce usado para interagir com o Salesforce. APIs Isso assume a forma do seguinte exemplo:`https://domainName.my.salesforce.com`.

appId

O ID do aplicativo é o identificador do seu aplicativo cliente externo Salesforce (ECA). Você pode recuperar isso chamando o endpoint de uso do Salesforce. OAuth Ele deve começar com `0x` e conter somente caracteres alfanuméricos. Esse campo se refere ao external_client_app_identifier no guia de rotação do Salesforce.

ID do consumidor

O ID do consumidor é o identificador do seu consumidor do Salesforce External Client Application (ECA). Você pode recuperar isso chamando o endpoint Salesforce OAuth Credentials by App ID. Esse campo se refere ao consumer_id no guia de rotação do Salesforce.

Campos de metadados secretos

A seguir estão os campos de metadados necessários para alternar um segredo mantido pela Salesforce.

```
{
  "apiVersion": "v65.0",
  "adminSecretArn": "arn:aws:secretsmanager:us-
east-1:111122223333:secret:SalesforceClientSecret"
}
```

apiVersion

A versão da API do Salesforce é a versão da API da sua organização do Salesforce. A versão deve ser pelo menos v65.0. Ele deve estar no formato em *vXX.X* que *X* há um caractere numérico.

adminSecretArn

(Opcional) O ARN do segredo do administrador é o Amazon Resource Name (ARN) do segredo que contém as OAuth credenciais administrativas que serão usadas para alternar esse segredo do cliente Salesforce. No mínimo, o segredo do administrador deve conter um valor ConsumerKey e ConsumerSecret dentro da estrutura secreta. É um campo opcional e, se omitido, durante a rotação, o Secrets Manager usará OAuth as credenciais desse segredo para se autenticar no Salesforce.

Fluxo de uso

Os clientes que armazenam segredos do Salesforce AWS Secrets Manager têm a opção de alternar um segredo com as credenciais armazenadas no mesmo segredo ou usar as credenciais no segredo do administrador para rotação. Você pode criar seu segredo usando a CreateSecretchamada com o valor secreto contendo os campos mencionados acima e o tipo de segredo como SalesforceClientSecret. As configurações de rotação podem ser definidas usando

uma RotateSecretchamada. Essa chamada requer a especificação dos campos de metadados, como no exemplo acima. Se você optar por uma rotação usando credenciais no mesmo segredo, poderá pular o campo. adminSecretArn Além disso, os clientes devem fornecer um ARN de função na RotateSecretchamada que conceda ao serviço as permissões necessárias para alternar o segredo. Para obter um exemplo de política de permissões, consulte Segurança e permissões.

Para clientes que optam por alternar seus segredos usando um conjunto separado de credenciais (armazenado em um segredo administrativo), certifique-se de criar o segredo administrativo AWS Secrets Manager seguindo exatamente as mesmas etapas do segredo do consumidor. Você deve fornecer o ARN desse segredo administrativo nos metadados de rotação em uma RotateSecretchamada para seu segredo de consumidor.

A lógica de rotação segue a orientação fornecida pela Salesforce.

Token de atualização do Big ID

Campos de valor secreto

A seguir estão os campos que devem estar contidos no segredo do Secrets Manager:

```
{
  "hostname": "Host Name",
  "refreshToken": "Refresh Token"
}
```

hostname

Esse é o nome do host em que sua instância BigID está hospedada. Você deve inserir o nome de domínio totalmente qualificado da sua instância.

Token de atualização

O token de atualização do usuário JWT gerado no console BigID via Administração → Gerenciamento de acesso → Selecionar usuário → Gerar token → Salvar

Fluxo de uso

Você pode criar seu segredo usando a CreateSecretchamada com o valor secreto contendo os campos mencionados acima e o tipo de segredo como Big IDClient Secret. As configurações de rotação podem ser definidas usando uma RotateSecretchamada. Você também deve fornecer um

ARN de função na RotateSecretchamada que conceda ao serviço as permissões necessárias para alternar o segredo. Por exemplo, de uma política de permissões, consulte Segurança e permissões. Observe que o campo de metadados de rotação pode ser deixado vazio para esse parceiro.

Par de chaves Snowflake

Campos de valor secreto

A seguir estão os campos que devem estar contidos no segredo do Secrets Manager:

```
{
  "account": "Your Account Identifier",
  "user": "Your user name",
  "privateKey": "Your private Key",
  "publicKey": "Your public Key",
  "passphrase": "Your Passphrase"
}
```

usuário

O nome de usuário do Snowflake associado a essa autenticação de par de chaves. Esse usuário deve estar configurado no Snowflake para aceitar a autenticação por par de chaves, e a chave pública deve ser atribuída ao perfil desse usuário.

account

Seu identificador de conta do Snowflake usado para estabelecer a conexão. Isso pode ser extraído do URL do Snowflake (a parte antes de .snowflakecomputing.com)

privateKey

A chave privada RSA no formato PEM usada para autenticação. Os BEGIN/END marcadores são opcionais.

Chave pública

A contraparte da chave pública no formato PEM correspondente à chave privada. Os BEGIN/END marcadores são opcionais.

frase secreta

(Opcional) Esse campo se refere à frase secreta usada para descriptografar a chave privada criptografada.

Campos de metadados secretos

A seguir estão os campos de metadados do Snowflake:

```
{
  "cryptographicAlgorithm": "Your Cryptographic algorithm",
  "encryptPrivateKey": "True/False"
}
```

Algoritmo criptográfico

(Opcional) Isso se refere ao algoritmo usado para geração de chaves. Você tem uma escolha de 3 algoritmos:`RS256 | RS384 | RS512`. Esse campo é opcional e o algoritmo padrão escolhido é RS256.

encryptPrivateKey

(Opcional) Esse campo pode ser usado para escolher se você deseja criptografar sua chave privada. Ele é "false" por padrão. A senha para criptografia é gerada aleatoriamente.

Fluxo de uso

Você pode criar seu segredo usando a CreateSecretchamada com o valor secreto contendo os campos mencionados acima e o tipo de segredo como SnowflakeKeyPairAuthentication. As configurações de rotação podem ser definidas usando uma RotateSecretchamada. Opcionalmente, você pode fornecer os campos de metadados secretos com base em sua necessidade. Você também deve fornecer um ARN de função na RotateSecretchamada que conceda ao serviço as permissões necessárias para alternar o segredo. Por exemplo, de uma política de permissões, consulte Segurança e permissões. Observe que o campo de metadados de rotação pode ser deixado vazio para esse parceiro.

Segurança e permissões

Segredos externos gerenciados não exigem que você compartilhe privilégios de administrador de suas contas de aplicativos de terceiros com. AWS Em vez disso, o processo de rotação usa credenciais e metadados que você fornece para fazer chamadas de API autorizadas para o aplicativo de terceiros para atualizações e validação de credenciais.

Os segredos externos gerenciados mantêm os mesmos padrões de segurança dos outros tipos de segredos do Secrets Manager. Os valores secretos são criptografados em repouso usando

suas chaves KMS e em trânsito usando TLS. O acesso aos segredos é controlado por meio de políticas do IAM e políticas baseadas em recursos. Ao usar uma chave gerenciada pelo cliente para criptografar seu segredo, você precisará atualizar a política do IAM da função de rotação e a política de confiança da CMK para fornecer as permissões necessárias para garantir a rotação bem-sucedida.

Para que a rotação funcione corretamente, você deve fornecer ao Secrets Manager permissões específicas para gerenciar o ciclo de vida secreto. Essas permissões podem ter como escopo os segredos individuais e seguir o princípio do menor privilégio. A função de rotação que você fornece é validada durante a configuração e usada exclusivamente para operações de rotação.

Você pode restringir a entrada de IP para seu recurso externo permitindo somente os intervalos de AWS IP para o EC2 na região em que seu segredo existe. Essa lista de intervalos de IP pode mudar, então você deve atualizar suas regras de entrada periodicamente.

AWS Secrets Manager também oferece soluções de toque único para criar a política do IAM com as permissões necessárias para gerenciar o segredo ao criar o segredo por meio do console do Secrets Manager. As permissões para essa função são reduzidas para cada parceiro de integração em cada região.

Exemplo de política de permissões:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowRotationAccess",
      "Action": [
        "secretsmanager:DescribeSecret",
        "secretsmanager:GetSecretValue",
        "secretsmanager:PutSecretValue",
        "secretsmanager:UpdateSecretVersionStage"
      ],
      "Resource": "*",
      "Effect": "Allow",
      "Condition": {
        "StringEquals": {
          "secretsmanager:resource/Type": "SalesforceClientSecret"
        }
      }
    },
```

```
    {
      "Sid": "AllowPasswordGenerationAccess",
      "Action": [
        "secretsmanager:GetRandomPassword"
      ],
      "Resource": "*",
      "Effect": "Allow"
    }
  ]
}
```

Observação: a lista de tipos de segredos que estão disponíveis para SecretsManager:resource/type pode ser encontrada em Parceiros de Integração.

Exemplo de política de confiança:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "SecretsManagerPrincipalAccess",
      "Effect": "Allow",
      "Principal": {
        "Service": "secretsmanager.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "111122223333"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:secretsmanager:us-east-1:111122223333:secret:*"
        }
      }
    }
  ]
}
```

Monitore e solucione problemas de segredos externos gerenciados

Segredos externos gerenciados fornecem recursos abrangentes de monitoramento por meio de AWS CloudTrail registros e CloudWatch métricas da Amazon. Todas as atividades de rotação são

registradas com informações detalhadas sobre sucesso, falha e quaisquer erros encontrados durante o processo.

Problemas comuns no fluxo de trabalho de rotação incluem uma configuração incorreta das permissões da função ou do valor secreto. A não definição desses campos no formato especificado pelos parceiros de integração pode causar falhas de rotação, pois o serviço não conseguirá acessar o segredo ou se conectar ao cliente do parceiro de integração para atualizar o segredo. Outros problemas podem ser problemas de conectividade de rede, expiração de credenciais ou disponibilidade de serviços de parceiros. O serviço de rotação gerenciada inclui lógica de repetição e tratamento de erros para maximizar a confiabilidade

Você pode monitorar cronogramas de rotação, taxas de sucesso e métricas de desempenho por meio da Amazon CloudWatch. Você pode configurar alarmes personalizados por meio do Event Bridge para alertá-lo sobre falhas de rotação ou outros problemas que exijam atenção.

Migrando segredos existentes

Você tem a opção de migrar seus segredos de parceiros existentes para segredos externos gerenciados. Isso pode ser feito com uma UpdateSecretchamada. Você deve atualizar o valor secreto e os metadados conforme mencionado no guia. Se você já tiver uma lógica de rotação personalizada configurada para esses segredos, primeiro cancele a rotação usando uma CancelRotateSecretchamada.

Limitações e considerações

Segredos externos gerenciados não oferecem suporte a segredos efêmeros com vida útil inferior a quatro horas. Os segredos associados aos certificados de infraestrutura de chave pública também não são suportados.

Os segredos externos gerenciados são suportados somente por parceiros que se integraram ao. AWS Secrets Manager Para obter uma lista completa, consulte Parceiros de integração. Não vê seu parceiro na lista? Diga a eles que se inscrevam no AWS Secrets Manager

Se você atualizar ou alternar valores secretos diretamente do serviço de cliente parceiro fora do mecanismo de rotação do Secrets Manager, a sincronização entre os sistemas poderá ser interrompida. Embora o Secrets Manager forneça avisos de console e prevenção programática para atualizações manuais de valores secretos, você ainda pode modificar valores diretamente em seu aplicativo de terceiros. Para restabelecer a sincronização após out-of-band as atualizações,

você deve atualizar o valor secreto para refletir o segredo correto e, em seguida, invocar a RotateSecretAPI para garantir rotações bem-sucedidas contínuas.

Crie AWS Secrets Manager segredos em AWS CloudFormation

Você pode criar segredos em uma CloudFormation pilha usando o `AWS::SecretsManager::Secret` recurso em um CloudFormation modelo, conforme mostrado emCriar um segredo.

Para criar um segredo de administrador para Amazon RDS ou Aurora, recomendamos que você use `ManageMasterUserPassword` em `AWS::RDS::DBCluster`. Em seguida, o Amazon RDS cria o segredo e gerencia a alternância para você. Para obter mais informações, consulte Alternância gerenciada.

Para credenciais do Amazon Redshift e do Amazon DocumentDB, primeiro, crie um segredo com uma senha gerada pelo Secrets Manager e, em seguida, usar uma referência dinâmica para recuperar o nome de usuário e a senha do segredo para usar como credenciais para um novo banco de dados. Em seguida, use o recurso `AWS::SecretsManager::SecretTargetAttachment` para adicionar detalhes sobre o banco de dados ao segredo de que o Secrets Manager precisa para alternar o segredo. Por fim, para ativar a alternância automática, use o recurso `AWS::SecretsManager::RotationSchedule` e forneça uma função de rotação e um cronograma. Veja os exemplos a seguir:

- Criar um segredo com credenciais do Amazon Redshift
- Criar um segredo com credenciais do Amazon DocumentDB

Para anexar uma política de recursos ao seu segredo, use o recurso `AWS::SecretsManager::ResourcePolicy`.

Para obter informações sobre a criação de recursos com CloudFormation, consulte Aprenda os conceitos básicos do modelo no Guia do CloudFormation usuário. Você também pode usar o AWS Cloud Development Kit (AWS CDK) Para obter mais informações, consulte Biblioteca de construções do AWS Secrets Manager.

Crie um AWS Secrets Manager segredo com CloudFormation

Este exemplo cria um segredo denominado **`CloudFormationCreatedSecret-a1b2c3d4e5f6`**. O valor do segredo é o JSON seguinte, com uma senha de 32 caracteres gerada na criação do segredo.

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

Este exemplo usa o seguinte CloudFormation recurso:

- `AWS::SecretsManager::Secret`

Para obter informações sobre a criação de recursos com CloudFormation, consulte Aprenda os conceitos básicos do modelo no Guia do CloudFormation usuário.

JSON

```
{
    "Resources": {
        "CloudFormationCreatedSecret": {
            "Type": "AWS::SecretsManager::Secret",
            "Properties": {
                "Description": "Simple secret created by CloudFormation.",
                "GenerateSecretString": {
                    "SecretStringTemplate": "{\"username\": \"saanvi\"}",
                    "GenerateStringKey": "password",
                    "PasswordLength": 32
                }
            }
        }
    }
}
```

YAML

```
Resources:
  CloudFormationCreatedSecret:
```

```
    Type: 'AWS::SecretsManager::Secret'
    Properties:
      Description: Simple secret created by CloudFormation.
      GenerateSecretString:
        SecretStringTemplate: '{"username": "saanvi"}'
        GenerateStringKey: password
        PasswordLength: 32
```

Crie um AWS Secrets Manager segredo com rotação automática e uma instância de banco de dados MySQL do Amazon RDS com CloudFormation

Para criar um segredo de administrador para Amazon RDS ou Aurora, recomendamos que você use `ManageMasterUserPassword`, conforme mostrado no exemplo Criar um segredo do Secrets Manager para uma senha mestre em `AWS::RDS::DBCluster`. Em seguida, o Amazon RDS cria o segredo e gerencia a alternância para você. Para obter mais informações, consulte Alternância gerenciada.

Crie um AWS Secrets Manager segredo e um cluster do Amazon Redshift com CloudFormation

Para criar um segredo do administrador do Amazon Redshift, recomendamos usar os exemplos em `AWS::Redshift::Cluster` e `AWS::RedshiftServerless::Namespace`.

Crie um AWS Secrets Manager segredo e uma instância do Amazon DocumentDB com CloudFormation

Este exemplo cria um segredo e uma instância do Amazon DocumentDB usando as credenciais no segredo como usuário e senha. O segredo tem uma política baseada em recursos anexada que define quem pode acessar o segredo. O modelo também cria uma função de alternância do Lambda a partir de Modelos de função de alternância e configura o segredo para alternar automaticamente entre às 8h e 10h UTC do primeiro dia de cada mês. Como uma prática recomendada de segurança, a instância está em uma Amazon VPC.

Este exemplo usa os seguintes CloudFormation recursos para o Secrets Manager:

- AWS::SecretsManager::Secret
- AWS::SecretsManager::SecretTargetAttachment
- AWS::SecretsManager::RotationSchedule

Para obter informações sobre a criação de recursos com CloudFormation, consulte Aprenda os conceitos básicos do modelo no Guia do CloudFormation usuário.

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 * ? *)
```

Como o Secrets Manager usa AWS CloudFormation

Quando você usa o console para ativar a rotação, o Secrets Manager usa AWS CloudFormation para criar recursos para rotação. Se você criar uma nova função de rotação durante esse processo, CloudFormation cria uma `AWS::Serverless::Function`com base na apropriadaModelos de função de alternância. Em seguida, CloudFormation define o `RotationSchedule`, que define a função de rotação e as regras de rotação para o segredo. Você pode ver a CloudFormation pilha escolhendo Exibir pilha no banner depois de ativar a rotação automática.

Para obter informações sobre como ativar a rotação automática, consulte Alternar segredos .

Crie AWS Secrets Manager segredos em AWS Cloud Development Kit (AWS CDK)

Para criar, gerenciar e recuperar segredos em uma aplicação CDK, é possível usar a AWS Secrets Manager Construct Library, que contém construções `ResourcePolicy`, `RotationSchedule`, `Secret`, `SecretRotation` e `SecretTargetAttachment`.

Uma prática recomendada para usar segredos em aplicações CDK é primeiro criar o segredo usando o console ou a CLI e, em seguida, importar o segredo para sua aplicação CDK.

Para ver exemplos, consulte:

- Criar um segredo
- Importar um segredo
- Recuperar segredos
- Conceder permissão para usar o segredo
- Alternar um segredo
- Alternar um segredo de banco de dados
- Replicar um segredo para outras regiões

Para obter mais informações sobre o CDK, consulte o Guia do desenvolvedor do AWS Cloud Development Kit (AWS CDK) v2.

Monitore AWS Secrets Manager segredos

AWS fornece ferramentas de monitoramento para monitorar os segredos do Secrets Manager, relatar quando algo está errado e realizar ações automáticas quando apropriado. É possível usar os logs se precisar investigar qualquer uso ou alteração inesperada e, em seguida, reverter as alterações indesejadas. Você também pode estabelecer verificações automatizadas para o uso inadequado de segredos e qualquer tentativa de exclusão de segredos.

Tópicos

AWS Secrets Manager Registre eventos com AWS CloudTrail

AWS CloudTrail registra todas as chamadas de API para o Secrets Manager como eventos, incluindo chamadas do console do Secrets Manager, bem como vários outros eventos para rotação e exclusão de versões secretas. Para obter uma lista das entradas de log dos registros do Secrets Manager, consulte CloudTrail entradas.

Você pode usar o CloudTrail console para ver os últimos 90 dias de eventos gravados. Para um registro contínuo de eventos em sua AWS conta, incluindo eventos para o Secrets Manager, crie uma trilha para que CloudTrail entregue os arquivos de log para um bucket do Amazon S3. Consulte Criação de uma trilha para sua AWS conta. Você também pode configurar CloudTrail para receber arquivos de CloudTrail log de vários Contas da AWSRegiões da AWSe.

Você pode configurar outros AWS serviços para analisar e agir com base nos dados coletados nos CloudTrail registros. Veja as integrações de AWS serviços com CloudTrail registros. Você também pode receber notificações ao CloudTrail publicar novos arquivos de log em seu bucket do Amazon S3. Consulte Configuração de notificações do Amazon SNS para. CloudTrail

Para recuperar eventos do Secrets Manager dos CloudTrail registros (console)

1. Abra o CloudTrail console em https://console.aws.amazon.com/cloudtrail/.
2. Certifique-se de que o console esteja apontando para a região em que os eventos ocorreram. O console mostra apenas os eventos que ocorreram na região selecionada. Escolha a região na lista suspensa no canto superior direito do console.
3. No painel de navegação à esquerda, escolha Histórico de eventos.
4. Escolha Filtrar critérios e and/or um intervalo de tempo para ajudá-lo a encontrar o evento que você está procurando. Por exemplo:

    a. Para ver todos os eventos do Secrets Manager, em Pesquisar atributos, escolha Origem do evento. Em seguida, em Enter envent source (Inserir origem do evento), escolha **`secretsmanager.amazonaws.com`**.

    b. Para ver todos os eventos para um segredo, em Pesquisar atributos, escolha Nome do recurso. Em seguida, em Insira um nome de recurso, insira o nome do segredo.
5. Para ver outros detalhes, escolha a seta de expansão ao lado do evento. Para ver todas as informações disponíveis, escolha View evento (Visualizar evento).

AWS CLI

Example Recupere eventos do Secrets Manager dos registros CloudTrail

O exemplo de `lookup-events` a seguir procura eventos do Secrets Manager.

```
aws cloudtrail lookup-events \
    --region us-east-1 \
    --lookup-attributes
 AttributeKey=EventSource,AttributeValue=secretsmanager.amazonaws.com
```

AWS CloudTrail entradas para Secrets Manager

AWS Secrets Manager grava entradas em seu AWS CloudTrail log para todas as operações do Secrets Manager e para outros eventos relacionados à rotação e exclusão. Para obter informações sobre como agir nesses eventos, consulte Combine eventos do Secrets Manager com EventBridge.

Tipos de entrada de log

- Entradas de log para operações do Secrets Manager

- Entradas de log para exclusão
- Entradas de log para replicação
- Entradas de log para alternância

Entradas de log para operações do Secrets Manager

Os eventos que são gerados por chamadas para as operações do Secrets Manager têm `"detail-type": ["AWS API Call via CloudTrail"]`.

> **Note**
>
> Antes de fevereiro de 2024, algumas operações do Secrets Manager relataram eventos que continham "aRN" em vez de "arn" para o ARN do segredo. Para obter mais informações, consulte AWS re:Post.

A seguir estão as CloudTrail entradas geradas quando você ou um serviço chamam as operações do Secrets Manager por meio da API, SDK ou CLI.

BatchGetSecretValue

Gerado pela BatchGetSecretValueoperação. Para obter informações sobre a recuperação de segredos, consulte Obter segredos.

CancelRotateSecret

Gerado pela CancelRotateSecretoperação. Para obter informações sobre alternância, consulte Alternar segredos .

CreateSecret

Gerado pela CreateSecretoperação. Para obter informações sobre a criação de segredos, consulte Gerenciar segredos.

DeleteResourcePolicy

Gerado pela DeleteResourcePolicyoperação. Para obter mais informações sobre permissões, consulte the section called “Autenticação e controle de acesso”.

DeleteSecret

Gerado pela DeleteSecretoperação. Para obter informações sobre como excluir segredos, consulte the section called “Excluir um segredo”.

DescribeSecret

Gerado pela DescribeSecretoperação.

GetRandomPassword

Gerado pela GetRandomPasswordoperação.

GetResourcePolicy

Gerado pela GetResourcePolicyoperação. Para obter mais informações sobre permissões, consulte the section called “Autenticação e controle de acesso”.

GetSecretValue

Gerado pelas BatchGetSecretValueoperações GetSecretValuee. Para obter informações sobre a recuperação de segredos, consulte Obter segredos.

ListSecrets

Gerado pela ListSecretsoperação. Para obter informações sobre a listagem de segredos, consulte the section called “Localizar segredos”.

ListSecretVersionIds

Gerado pela ListSecretVersionIdsoperação.

PutResourcePolicy

Gerado pela PutResourcePolicyoperação. Para obter mais informações sobre permissões, consulte the section called “Autenticação e controle de acesso”.

PutSecretValue

Gerado pela PutSecretValueoperação. Para obter informações sobre como atualizar um segredo, consulte the section called “Modificar um segredo”.

RemoveRegionsFromReplication

Gerado pela RemoveRegionsFromReplicationoperação. Para obter informações sobre a replicação de um segredo, consulte Replicação multirregional.

ReplicateSecretToRegions

Gerado pela ReplicateSecretToRegionsoperação. Para obter informações sobre a replicação de um segredo, consulte Replicação multirregional.

RestoreSecret

Gerado pela RestoreSecretoperação. Para obter informações sobre como restaurar um segredo excluído, consulte the section called “Restaurar um segredo”.

RotateSecret

Gerado pela RotateSecretoperação. Para obter informações sobre alternância, consulte Alternar segredos .

StopReplicationToReplica

Gerado pela StopReplicationToReplicaoperação. Para obter informações sobre a replicação de um segredo, consulte Replicação multirregional.

TagResource

Gerado pela TagResourceoperação. Para obter informações sobre como marcar um segredo, consulte the section called “Marcação de segredos do ”.

UntagResource

Gerado pela UntagResourceoperação. Para obter informações sobre como desmarcar um segredo, consulte the section called “Marcação de segredos do ”.

UpdateSecret

Gerado pela UpdateSecretoperação. Para obter informações sobre como atualizar um segredo, consulte the section called “Modificar um segredo”.

UpdateSecretVersionStage

Gerado pela UpdateSecretVersionStageoperação. Para obter informações sobre estágios de versão, consulte the section called “Versões do segredo”.

ValidateResourcePolicy

Gerado pela ValidateResourcePolicyoperação. Para obter mais informações sobre permissões, consulte the section called “Autenticação e controle de acesso”.

Entradas de log para exclusão

Além dos eventos das operações do Secrets Manager, o Secrets Manager gera os eventos a seguir relacionados à exclusão. Esses eventos têm `"detail-type": ["AWS Service Event via CloudTrail"]`.

CancelSecretVersionDelete

Gerado pelo serviço do Secrets Manager. Se você chamar `DeleteSecret` em um segredo que tenha versões e, posteriormente, chamar `RestoreSecret`, o Secrets Manager registra esse evento para cada versão de segredo que foi restaurada. Para obter informações sobre como restaurar um segredo excluído, consulte the section called “Restaurar um segredo”.

EndSecretVersionDelete

Gerado pelo serviço do Secrets Manager quando uma versão do segredo é excluída. Para obter mais informações, consulte the section called “Excluir um segredo”.

StartSecretVersionDelete

Gerado pelo serviço do Secrets Manager quando ele começa a exclusão da versão de um segredo. Para obter informações sobre como excluir segredos, consulte the section called “Excluir um segredo”.

SecretVersionDeletion

Gerado pelo serviço do Secrets Manager quando ele começa a exclusão da versão de um segredo. Para obter mais informações, consulte Secret versions (Versões de segredos).

Entradas de log para replicação

Além dos eventos das operações do Secrets Manager, o Secrets Manager gera os eventos a seguir relacionados à replicação. Esses eventos têm `"detail-type": ["AWS Service Event via CloudTrail"]`.

ReplicationFailed

Gerado pelo serviço do Secrets Manager quando a replicação falha. Para obter informações sobre a replicação de um segredo, consulte Replicação multirregional.

ReplicationStarted

Gerado pelo serviço Secrets Manager quando ele começa a replicar um segredo. Para obter informações sobre a replicação de um segredo, consulte Replicação multirregional.

ReplicationSucceeded

Gerado pelo serviço do Secrets Manager quando um segredo é replicado com sucesso. Para obter informações sobre a replicação de um segredo, consulte Replicação multirregional.

Entradas de log para alternância

Além dos eventos das operações do Secrets Manager, o Secrets Manager gera os eventos a seguir relacionados à alternância. Esses eventos têm `"detail-type": ["AWS Service Event via CloudTrail"]`.

RotationStarted

Gerado pelo serviço Secrets Manager quando ele começa a alternar um segredo. Para obter informações sobre alternância, consulte Alternar segredos .

RotationAbandoned

Gerado pelo serviço do Secrets Manager quando ele abandona uma tentativa de alternância e remove o rótulo `AWSPENDING` de uma versão existente de um segredo. O Secrets Manager abandona a alternância quando você cria uma nova versão de um segredo durante a alternância. Para obter informações sobre alternância, consulte Alternar segredos .

RotationFailed

Gerado pelo serviço do Secrets Manager quando a alternância falha. Para obter informações sobre alternância, consulte the section called “Solucionar problemas de alternância”.

RotationSucceeded

Gerado pelo serviço do Secrets Manager quando um segredo é alternado com êxito. Para obter informações sobre alternância, consulte Alternar segredos .

TestRotationStarted

Gerado pelo serviço do Secrets Manager quando ele começa a testar a alternância de um segredo que não está programado para alternância imediata. Para obter informações sobre alternância, consulte Alternar segredos .

TestRotationSucceeded

Gerado pelo serviço do Secrets Manager quando ele testa com êxito a alternância de um segredo que não está programado para alternância imediata. Para obter informações sobre alternância, consulte Alternar segredos .

TestRotationFailed

Gerado pelo serviço do Secrets Manager quando ele testa a alternância de um segredo que não está programado para alternância imediata e a alternância falhou. Para obter informações sobre alternância, consulte the section called “Solucionar problemas de alternância”.

Monitore AWS Secrets Manager com a Amazon CloudWatch

Usando a Amazon CloudWatch, você pode monitorar AWS serviços e criar alarmes para avisar quando as métricas mudam. CloudWatch mantém essas estatísticas por 15 meses, para que você possa acessar informações históricas e ter uma melhor perspectiva sobre o desempenho de seu aplicativo ou serviço web. Pois AWS Secrets Manager, você pode monitorar o número de segredos em sua conta, incluindo segredos marcados para exclusão e chamadas de API para o Secrets Manager, incluindo chamadas feitas por meio do console. Para obter informações sobre como monitorar métricas, consulte Usar CloudWatch métricas no Guia do CloudWatch usuário.

Para encontrar métricas do Secrets Manager

1. No CloudWatch console, em Métricas, escolha Todas as métricas.
2. Na caixa de pesquisa Métricas, insira `secret`.
3. Faça o seguinte:
   - Para monitorar o número de segredos em sua conta, escolha AWS/eSecretsManager, em seguida, selecione SecretCount. Essa métrica é publicada de hora em hora.
   - Para monitorar chamadas de API para o Secrets Manager, incluindo chamadas feitas por meio do console, escolha Uso > Por AWS recurso e selecione as chamadas de API a serem monitoradas. Para obter uma lista do Secrets Manager APIs, consulte Operações do Secrets Manager.
4. Faça o seguinte:
   - Para criar um gráfico da métrica, consulte Representação gráfica de métricas no Guia do CloudWatch usuário da Amazon.

- Para detectar anomalias, consulte Usando a detecção de CloudWatch anomalias no Guia do usuário da Amazon CloudWatch .
- Para obter estatísticas de uma métrica, consulte Obter estatísticas de uma métrica no Guia CloudWatch do usuário da Amazon.

CloudWatch alarmes

Você pode criar um CloudWatch alarme que envia uma mensagem do Amazon SNS quando o valor de uma métrica muda e faz com que o alarme mude de estado. É possível definir um alarme em `ResourceCount` de métrica do Secrets Manager, que é o número de segredos em sua conta. Também é possível definir alarmes. Um alarme observa uma métrica ao longo de um período especificado por você e realiza ações com base no valor da métrica relativo a um determinado limite ao longo de vários períodos. Os alarmes invocam ações somente para mudanças de estado sustentadas. CloudWatch os alarmes não invocam ações simplesmente porque estão em um determinado estado; o estado deve ter sido alterado e mantido por um determinado número de períodos.

Para obter mais informações, consulte Usando CloudWatch alarmes da Amazon e Criar um CloudWatch alarme com base na detecção de anomalias no Guia do CloudWatch usuário.

Você também pode definir alarmes que observam determinados limites e enviam notificações ou realizam ações quando esses limites são atingidos. Para obter mais informações, consulte o Guia CloudWatch do usuário da Amazon.

Combine AWS Secrets Manager eventos com a Amazon EventBridge

Na Amazon EventBridge, você pode combinar eventos do Secrets Manager a partir de entradas de CloudTrail registro. Você pode configurar EventBridge regras que procurem esses eventos e, em seguida, enviem novos eventos gerados a um alvo para agir. Para obter uma lista das CloudTrail entradas que o Secrets Manager registra, consulteCloudTrail entradas. Para obter instruções de configuração EventBridge, consulte Introdução EventBridge no Guia do EventBridge usuário.

Corresponder todas as alterações com um segredo especificado

> **Note**
>
> Como alguns eventos do Secrets Manager devolvem o ARN do segredo com uma capitalização diferente, em padrões de eventos que correspondem a mais de uma ação, para especificar um segredo por ARN, talvez seja necessário incluir ambas as chaves `arn` e `aRN`. Para obter mais informações, consulte AWS re:Post.

O exemplo a seguir mostra um padrão de EventBridge evento que corresponde às entradas de registro para alterações em um segredo.

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

Corresponder eventos quando um valor do segredo é alternado

O exemplo a seguir mostra um padrão de EventBridge evento que corresponde às entradas de CloudTrail registro para alterações de valores secretos que ocorrem a partir de atualizações manuais ou rotação automática. Como alguns desses eventos são provenientes de operações do Secrets Manager e outros são gerados pelo serviço do Secrets Manager, você deve incluir o `detail-type` para ambos.

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

Monitore quando AWS Secrets Manager os segredos programados para exclusão são acessados

Você pode usar uma combinação de AWS CloudTrail Amazon CloudWatch Logs e Amazon Simple Notification Service (Amazon SNS) para criar um alarme que notifique você sobre qualquer tentativa de acessar uma exclusão secreta pendente. Se você receber uma notificação de um alarme, talvez queira cancelar a exclusão do segredo para ter mais tempo para decidir se deseja realmente excluí-lo. Sua investigação talvez resulte na restauração do segredo porque ainda precisa dele. Como alternativa, talvez seja necessário atualizar o usuário com detalhes do novo segredo a ser usado.

Os procedimentos a seguir explicam como receber uma notificação quando uma solicitação para a `GetSecretValue` operação resulta em uma mensagem de erro específica gravada em seus arquivos de CloudTrail log. É possível executar outras operações de API no segredo sem acionar o alarme. Esse CloudWatch alarme detecta um uso que pode indicar que uma pessoa ou aplicativo está usando credenciais desatualizadas.

Antes de iniciar esses procedimentos, você deve ativar a conta Região da AWS e CloudTrail na qual pretende monitorar as solicitações de AWS Secrets Manager API. Para obter instruções, consulte Criar uma trilha pela primeira vez no AWS CloudTrail Manual do usuário.

Etapa 1: configurar a entrega do arquivo de CloudTrail log para o CloudWatch Logs

Você deve configurar a entrega de seus arquivos de CloudTrail log para o CloudWatch Logs. Você faz isso para que CloudWatch os Logs possam monitorá-los em busca de solicitações da API Secrets Manager para recuperar um segredo pendente de exclusão.

Para configurar a entrega do arquivo de CloudTrail log para o CloudWatch Logs

1. Abra o CloudTrail console em https://console.aws.amazon.com/cloudtrail/.
2. Na barra de navegação superior, escolha a opção Região da AWS para monitorar segredos.

3. No painel de navegação esquerdo, escolha Trilhas e, em seguida, escolha o nome da trilha a ser configurada. CloudWatch
4. Na página Configuração de trilhas, role para baixo até a seção CloudWatch Registros e escolha o ícone de edição (✏️).
5. Em New or existing log group, digite um nome para o grupo de log, como **`CloudTrail/MyCloudWatchLogGroup`**.
6. Para a função do IAM, você pode usar a função padrão chamada CloudTrail_ CloudWatchLogs _Role. Essa função tem uma política de função padrão com as permissões necessárias para entregar CloudTrail eventos ao grupo de registros.
7. Escolha Continue para salvar suas configurações.
8. Em AWS CloudTrail Entregará CloudTrail eventos associados à atividade da API em sua conta na página do grupo de CloudWatch registros de registros, escolha Permitir.

Etapa 2: criar o CloudWatch alarme

Para receber uma notificação quando uma operação `GetSecretValue` da API Secrets Manager solicitar o acesso a uma exclusão secreta pendente, você deve criar um CloudWatch alarme e configurar a notificação.

Para criar um CloudWatch alarme

1. Faça login no CloudWatch console em https://console.aws.amazon.com/cloudwatch/.
2. Na barra de navegação superior, escolha a AWS região em que você deseja monitorar os segredos.
3. No painel de navegação esquerdo, selecione Logs.
4. Na lista de grupos de registros, marque a caixa de seleção ao lado do grupo de registros que você criou no procedimento anterior, como CloudTrail/MyCloudWatchLogGroup. Escolha Create Metric Filter (Criar filtro de métrica).
5. Para Filter Pattern, digite ou cole o seguinte:

   ```
   { $.eventName = "GetSecretValue" && $.errorMessage = "*secret because it was marked
    for deletion*" }
   ```

   Escolha Assign Metric (Atribuir métrica).

6. Na página Create Metric Filter and Assign a Metric (Criar filtro de métrica e atribuir uma métrica), faça o seguinte:

   a. Em Metric Namespace (Namespace da métrica), digite **`CloudTrailLogMetrics`**.
   b. Para Metric Name (Nome da métrica), digite **`AttemptsToAccessDeletedSecrets`**.
   c. Escolha Show advanced metric settings, em seguida, se necessário, para Metric Value, digite **1**.
   d. Selecione Create Filter (Criar filtro).
7. Na caixa de filtro, selecione Create Alarm (Criar alarme).
8. Na janela Create Alarm (Criar alarme), faça o seguinte:

   a. Para Name (Nome), digite **`AttemptsToAccessDeletedSecretsAlarm`**.
   b. Em Whenever: (Sempre:), em is: (é:), escolha **>=** e digite **1**.
   c. Ao lado de Send notification to:, siga um destes procedimentos:

      - Para criar e usar um novo tópico do Amazon SNS, escolha New list (Nova lista) e digite o nome do novo tópico. Para Lista de e-mails:, digite pelo menos um endereço de e-mail. É possível digitar mais de um endereço de e-mail, basta separá-los com vírgulas.
      - Para usar um tópico existente do Amazon SNS, escolha o nome do tópico a ser usado. Se a lista não existir, escolha Select list (Selecionar lista).

   d. Escolha Criar Alarm.

Etapa 3: testar o CloudWatch alarme

Para testar seu alarme, crie um segredo e, em seguida, programe sua exclusão. Em seguida, tente recuperar o valor do segredo. Em breve, você receberá um e-mail no endereço configurado no alarme. Ele alerta sobre o uso de um segredo com a exclusão programada.

Monitore AWS Secrets Manager segredos para verificar a conformidade usando AWS Config

Você pode usar AWS Config para avaliar seus segredos para ver se eles estão em conformidade com seus padrões. Você define seus requisitos internos de segurança e conformidade para segredos usando AWS Config regras. Em seguida, AWS Config pode identificar segredos que não estão de acordo com suas regras. Você também pode rastrear alterações em metadados de segredos,

a configuração de alternância, a chave KMS usada para criptografia de segredos, a função de alternância do Lambda e as tags associadas a um segredo.

Você pode configurar AWS Config para notificá-lo sobre alterações. Para obter mais informações, consulte Notificações AWS Config enviadas para um tópico do Amazon SNS.

Se você tiver segredos em vários Contas da AWS e Regiões da AWS em sua organização, poderá agregar esses dados de configuração e conformidade. Para obter mais informações, consulte Agregação de dados de várias contas e regiões.

Para avaliar se os segredos estão em conformidade

- Siga as instruções em Avaliação de seus recursos com AWS Config regras e escolha uma das seguintes regras:
  - `secretsmanager-secret-unused`: verifica se os segredos foram acessados dentro do número de dias especificado.
  - `secretsmanager-using-cmk`— Verifica se os segredos são criptografados usando a chave gerenciada pelo cliente Chave gerenciada pela AWS `aws/secretsmanager` ou uma chave gerenciada pelo cliente que você criou AWS KMS.
  - `secretsmanager-rotation-enabled-check`: verifica se a alternância está configurada para segredos armazenados no Secrets Manager.
  - `secretsmanager-scheduled-rotation-success-check`: verifica se a última alternância bem-sucedida está dentro da frequência de alternância configurada. A frequência mínima para a verificação é diária.
  - `secretsmanager-secret-periodic-rotation`: verifica se os segredos foram alternados dentro do número de dias especificado.

Monitorar custos do Secrets Manager

Você pode usar CloudWatch a Amazon para monitorar as AWS Secrets Manager cobranças estimadas. Para obter mais informações, consulte Criação de um alarme de cobrança para monitorar suas AWS cobranças estimadas no Guia do CloudWatch usuário.

Outra opção para monitorar seus custos é a Detecção de Anomalias de AWS Custos. Para obter mais informações, consulte Detecção de gastos incomuns com a Detecção de anomalias de custo da AWS no Guia do usuário do gerenciamento de custos da AWS .

Para obter informações sobre como monitorar o uso do Secrets Manager, consulte the section called “Monitor com CloudWatch” e the section called “Faça login com AWS CloudTrail ”.

Para obter informações sobre AWS Secrets Manager preços, consultethe section called “Preços”.

Detecte ameaças com a Amazon GuardDuty

GuardDuty A Amazon é um serviço de detecção de ameaças que ajuda você a proteger suas contas, contêineres, cargas de trabalho e os dados do seu AWS ambiente. Usando modelos de aprendizado de máquina (ML) e recursos de detecção de anomalias e ameaças, monitora GuardDuty continuamente diferentes fontes de log para identificar e priorizar possíveis riscos de segurança e atividades maliciosas em seu ambiente. Por exemplo, GuardDuty detectará ameaças em potencial, como acesso incomum ou suspeito a segredos e exfiltração de credenciais, caso detecte credenciais que foram criadas exclusivamente para uma instância da Amazon por meio de uma função de lançamento de EC2 instância, mas estão sendo usadas em outra conta interna. AWS Para obter mais informações, consulte o Guia GuardDuty do usuário da Amazon.

Outro exemplo de caso de uso para detecção é o comportamento anômalo. Por exemplo, se AWS Secrets Manager normalmente recebe`create-secret`,, `get-secret-value``describe-secret`, e `list-secrets` chamadas de uma entidade usando o Java SDK e, em seguida, uma entidade diferente começa a chamar `batch-get-secret-value` e `get-secret-value` usar o de fora AWS CLI da VPN, GuardDuty pode relatar uma descoberta de que a segunda entidade está invocando de forma anômala. APIs Para obter mais informações, consulte o tipo de descoberta GuardDuty do IAM CredentialAccess:IAMUser/AnomalousBehavior.

Validação de conformidade para AWS Secrets Manager

Sua responsabilidade de conformidade ao usar o Secrets Manager é determinada pela confidencialidade de seus dados, pelos objetivos de conformidade de sua empresa e pelas leis e regulamentos aplicáveis. AWS fornece os seguintes recursos para ajudar na conformidade:

- Guias de início rápido de segurança e compatibilidade: esses guias de implantação abordam as considerações de arquitetura e fornecem etapas para implantação de ambientes de linha de base focados em compatibilidade e segurança na AWS.
- Documento técnico sobre arquitetura para segurança e conformidade com a HIPAA — Este whitepaper descreve como as empresas podem usar para criar aplicativos compatíveis com a HIPAA. AWS
- AWS Recursos de https://aws.amazon.com/compliance/resources/ de conformidade — Essa coleção de pastas de trabalho e guias pode ser aplicada ao seu setor e local.
- O AWS Config avalia até que ponto suas configurações de recursos atendem adequadamente às práticas internas e às diretrizes e regulamentações do setor. Para obter mais informações, consulte the section called “Monitorar segredos para conformidade”.
- AWS Security Hub CSPMfornece uma visão abrangente do seu estado de segurança interno AWS que ajuda você a verificar sua conformidade com os padrões e as melhores práticas do setor de segurança. Para obter informações sobre como usar o Security Hub CSPM para avaliar os recursos do Secrets Manager, consulte AWS Secrets Manager os controles no Guia do AWS Security Hub CSPM Usuário.
- O IAM Access Analyzer analisa políticas, incluindo instruções de condição em uma política, que permitem que uma entidade externa acesse um segredo. Para obter mais informações, consulte Pré-visualização de acesso com Access Analyzer.
- O AWS Systems Manager fornece runbooks predefinidos para o Secrets Manager. Para obter mais informações, consulte Referência do runbook Systems Manager Automation para o Secrets Manager.
- Você pode baixar relatórios de auditoria de terceiros usando AWS Artifact. Para obter mais informações, consulte Baixar relatórios em AWS Artifact .

Padrões de conformidade

AWS Secrets Manager passou por uma auditoria para os seguintes padrões e pode fazer parte de sua solução quando você precisar obter a certificação de conformidade.

- AWS HIPAA — expandiu seu programa de conformidade com a Lei de Portabilidade e Responsabilidade de Seguros de Saúde (HIPAA) para incluí-lo como um serviço qualificado pela HIPAA. AWS Secrets Manager Se você tiver um Acordo de Associado Comercial (BAA) assinado com AWS, você pode usar o Secrets Manager para ajudar a criar seus aplicativos compatíveis com HIPAA. AWS oferece um whitepaper com foco na HIPAA para clientes interessados em saber mais sobre como eles podem aproveitar o processamento e o armazenamento AWS de informações de saúde. Para obter mais informações, consulte Conformidade com a HIPAA.
- Organização participante do PCI — AWS Secrets Manager tem um Atestado de Conformidade para o Padrão de Segurança de Dados (DSS) do Setor de Cartões de Pagamento (PCI), versão 3.2, no nível 1 do provedor de serviços. Os clientes que usam AWS produtos e serviços para armazenar, processar ou transmitir dados do titular do cartão podem usá-los AWS Secrets Manager enquanto gerenciam sua própria certificação de conformidade com o PCI DSS. Para obter mais informações sobre o PCI DSS, incluindo como solicitar uma cópia do PCI AWS Compliance Package, consulte PCI DSS Nível 1.
- ISO — AWS Secrets Manager concluiu com sucesso a certificação de conformidade para ISO/IEC 27001, ISO/IEC 27017, ISO/IEC 27018 e ISO 9001. Para obter mais informações, consulte ISO 27001, ISO 27017, ISO 27018 e ISO 9001.
- AICPA SOC: os relatórios de Controles do Sistema e da Organização (CSO) são relatórios de exames de terceiros independentes que demonstram como o Secrets Manager obtém os principais controles e objetivos de compatibilidade. O objetivo desses relatórios é ajudar você e seus auditores a entender os AWS controles estabelecidos para apoiar as operações e a conformidade. Para obter mais informações, consulte Conformidade com o SOC.
- FedRAMP: o Federal Risk and Authorization Management Program (FedRAMP – Programa federal de gerenciamento de autorização e risco) é um programa do governo que disponibiliza uma abordagem padronizada para avaliação de segurança, autorização e monitoramento contínuo de produtos e serviços na nuvem. O Programa FedRAMP também fornece autorizações provisórias para serviços e regiões East/West para GovCloud consumir dados governamentais ou regulamentados. Para obter mais informações, consulte Conformidade com o FedRAMP.
- Departamento de Defesa — O Guia de Requisitos de Segurança de Computação em Nuvem (SRG) do Departamento de Defesa (DoD) fornece um processo padronizado de avaliação e autorização para que os provedores de serviços em nuvem (CSPs) obtenham uma autorização

provisória do DoD, para que possam atender aos clientes do DoD. Para obter mais informações, consulte Recursos de SRG do DoD

- IRAP: o Information Security Registered Assessors Program (IRAP) permite que os clientes do governo australiano validem se os controles apropriados estão em vigor e determinem o modelo de responsabilidade correto para o cumprimento dos requisitos do Manual de Segurança da Informação (ISM) do governo australiano produzido pelo Australian Cyber Security Centre (ACSC). Para obter mais informações, consulte Recursos do IRAP.
- OSPAR — A Amazon Web Services (AWS) obteve o certificado Outsourced Service Provider's Audit Report (OSPAR). AWS o alinhamento com as Diretrizes da Associação de Bancos de Cingapura (ABS) sobre objetivos e procedimentos de controle para provedores de serviços terceirizados (Diretrizes ABS) demonstra aos clientes o AWS compromisso dos clientes em atender às altas expectativas dos provedores de serviços em nuvem estabelecidas pelo setor de serviços financeiros em Cingapura. Para obter mais informações, consulte Recursos do OSPAR.

Segurança em AWS Secrets Manager

A segurança AWS é a maior prioridade. Como AWS cliente, você se beneficia de uma arquitetura de data center e rede criada para atender aos requisitos das organizações mais sensíveis à segurança.

Você e AWS compartilhamos a responsabilidade pela segurança. O modelo de responsabilidade compartilhada descreve a segurança da nuvem e a segurança na nuvem:

- Segurança da nuvem — AWS é responsável por proteger a infraestrutura que executa AWS os serviços na AWS nuvem. AWS também fornece serviços que você pode usar com segurança. Auditores de terceiros testam e verificam regularmente a eficácia da nossa segurança como parte dos Programas de conformidade da AWS. Para saber mais sobre os programas de conformidade aplicáveis AWS Secrets Manager, consulte AWS Serviços no escopo por programa de conformidade.
- Segurança na nuvem — Seu AWS serviço determina sua responsabilidade. Você também é responsável por outros fatores, incluindo a confidencialidade de seus dados, os requisitos da empresa e as leis e regulamentos aplicáveis.

Para obter mais recursos, consulte Pilar de segurança: AWS Well-Architected Framework.

Tópicos

Mitigue os riscos de usar o AWS CLI para armazenar seus segredos AWS Secrets Manager

Ao usar o AWS Command Line Interface (AWS CLI) para invocar AWS operações, você insere esses comandos em um shell de comando. Por exemplo, você pode usar o prompt de comando do Windows ou o Windows PowerShell, ou o shell Bash ou Z, entre outros. Muitos desses shells de comando incluem funcionalidade desenvolvida para aumentar a produtividade. Mas essa funcionalidade pode ser usada para comprometer seus segredos. Por exemplo, na maioria dos shells, é possível usar a tecla de seta para cima para ver o último comando inserido. O recurso histórico de comandos pode ser explorado por qualquer pessoa que acesse sua sessão não segura. Além disso, outros utilitários que funcionam em segundo plano podem ter acesso aos parâmetros de seus comandos, com o objetivo de ajudar você a realizar as tarefas com mais eficiência. Para mitigar esses riscos, siga estas etapas:

- Sempre bloqueie seu computador ao sair do console.
- Desinstale ou desabilite os utilitários do console que você não precisa ou não usa mais.
- Verifique se o shell ou o programa de acesso remoto, caso esteja usando um, não registra em log os comandos digitados.
- Use técnicas para passar parâmetros não capturados pelo histórico de comandos do shell. O exemplo a seguir mostra como você pode digitar o texto secreto em um arquivo de texto e, em seguida, passar o arquivo para o AWS Secrets Manager comando e destruí-lo imediatamente. Isso significa que o histórico típico do shell não captura o texto do segredo.

  O exemplo a seguir mostra os comandos típicos do Linux, mas seu shell pode exigir comandos um pouco diferentes:

```
$ 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.
```

Depois de executar esses comandos, será possível usar as setas para cima e para baixo para percorrer o histórico de comandos e ver se o texto do segredo está sendo exibido em alguma linha.

> **Important**
>
> Por padrão, não é possível executar uma técnica equivalente no Windows, a menos que você reduza primeiro o tamanho do buffer do histórico de comandos para 1.

Para configurar o prompt de comando do Windows para ter apenas 1 buffer de histórico de comando de 1 comando

1. Abra um prompt de comando como administrador (Executar como administrador).
2. Escolha o ícone no canto superior esquerdo e escolha Propriedades.
3. Na guia Options, defina Buffer Size e Number of Buffers como **1**, e escolha OK.
4. Sempre que precisar digitar um comando que você não deseja armazenar no histórico, insira outro comando imediatamente depois dele, como:

   ```
   echo.
   ```

   Isso garante que você descarrega o comando sensível.

Para o shell do prompt de comando do Windows, você pode baixar a SysInternals SDeleteferramenta e usar comandos semelhantes aos seguintes:

```
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.
```

Autenticação e controle de acesso para AWS Secrets Manager

O Secrets Manager usa o AWS Identity and Access Management (IAM) para proteger o acesso a segredos. O IAM fornece autenticação e controle de acesso. Autenticação verifica a identidade das solicitações das pessoas. O Secrets Manager usa um processo de login com senhas, chaves de acesso e tokens de autenticação multifator (MFA) para verificar a identidade dos usuários. Consulte Como fazer login em AWS. Controle de acesso garante que apenas pessoas aprovadas possam executar operações em recursos da AWS , como segredos. O Secrets Manager usa políticas para definir quem tem acesso a quais recursos e quais ações a identidade pode executar nesses recursos. Consulte Políticas e permissões no IAM.

Tópicos

Referência de permissões para AWS Secrets Manager

A referência de permissões do Secrets Manager está disponível em Ações, recursos e chaves de condição do AWS Secrets Manager na Referência de autorização do serviço.

Permissões de administrador do Secrets Manager

Para conceder permissões de administrador ao Secrets Manager, siga as instruções de Adicionar e remover permissões de identidade do IAM e anexe as seguintes políticas:

- SecretsManagerReadWrite
- IAMFullAccess

Recomendamos que você não conceda permissões de administrador a usuários finais. Embora isso permita que os usuários criem e gerenciem os segredos, a permissão necessária para habilitar a alternância (IAMFullAccess) concede permissões significativas que não são adequadas para usuários finais.

Permissões para acessar segredos

Ao usar as políticas de permissão do IAM, é possível controlar quais usuários ou serviços têm acesso aos seus segredos. Uma política de permissões descreve quem pode executar quais ações em quais recursos. Você pode:

- the section called “Políticas baseadas em identidade”
- the section called “Políticas baseadas em recursos”

Permissões para funções de alternância do Lambda

O Secrets Manager usa AWS Lambda funções para alternar segredos. A função do Lambda deve ter acesso ao segredo e também ao banco de dados ou serviço para o qual o segredo contém credenciais. Consulte Permissões para alternância.

Permissões para chaves de criptografia

O Secrets Manager usa chaves AWS Key Management Service (AWS KMS) para criptografar segredos. O tem Chave gerenciada pela AWS `aws/secretsmanager` automaticamente as

permissões corretas. Se você usar uma chave do KMS diferente, o Secrets Manager necessitará de permissões para essa chave. Consulte the section called “Permissões para a chave do KMS”.

Permissões para replicação

Ao usar as políticas de permissão do IAM, é possível controlar quais usuários ou serviços podem replicar seus segredos para outras regiões. Consulte the section called “Impedir a replicação”.

Políticas baseadas em identidade

É possível anexar políticas de permissões a identidades do IAM: usuários, grupos de usuários e funções. Em uma política baseada em identidades, especifique que segredos a identidade pode acessar e que ações a identidade pode executar nos segredos. Para obter mais informações, consulte Adicionar e remover permissões de identidade do IAM.

É possível conceder permissões a um perfil que representa uma aplicação ou um usuário em outro serviço. Por exemplo, uma aplicação em execução em uma instância do Amazon EC2 pode precisar de acesso a um banco de dados. É possível criar um perfil do IAM anexado ao perfil de instância do EC2 e usar uma política de permissões para conceder ao perfil acesso ao segredo que contém credenciais para o banco de dados. Para obter mais informações, consulte Uso de um perfil do IAM para conceder permissões a aplicações em execução em instâncias do Amazon EC2. Outros serviços aos quais é possível anexar perfis incluem o Amazon Redshift, o AWS Lambda e o Amazon ECS.

Você também pode conceder permissões a usuários autenticados por um sistema de identidade diferente do IAM. Por exemplo, é possível associar perfis do IAM a usuários de aplicações móveis que fazem login usando o Amazon Cognito. O perfil concede à aplicação credenciais temporárias com as permissões na política de permissões da função. Em seguida, é possível usar uma política de permissões para conceder ao perfil acesso ao segredo. Para obter mais informações, consulte Provedores de identidade e federação.

É possível usar políticas baseadas em identidades para:

- Conceder um acesso de identidade a vários segredos.
- Controlar quem pode criar novos segredos e quem pode acessar segredos que ainda não foram criados.
- Conceder a um grupo do IAM acesso a segredos.

Exemplos:

- Exemplo: permissão para recuperar valores de segredos individuais
- Exemplo: permissão para ler e descrever segredos individuais
- Exemplo: permissão para recuperar um grupo de valores de segredos em um lote
- Exemplo: curingas
- Exemplo: Permissão para criar segredos
- Exemplo: negar uma AWS KMS chave específica para criptografar segredos

Exemplo: permissão para recuperar valores de segredos individuais

Para conceder permissão para recuperar valores de segredos, é possível anexar políticas a segredos ou identidades. Para obter ajuda na determinação do tipo de política a ser usado, consulte Políticas baseadas em identidades e políticas baseadas em recursos. Para obter informações sobre como anexar uma política, consulte the section called “Políticas baseadas em recursos” e the section called “Políticas baseadas em identidade”.

Esse exemplo é útil quando você deseja conceder acesso a um grupo do IAM. Para conceder permissão para recuperar um grupo de segredos em uma chamada de API em lote, consulte the section called “Exemplo: permissão para recuperar um grupo de valores de segredos em um lote”.

Example Ler um segredo criptografado usando uma chave gerenciada pelo cliente

Se um segredo for criptografado usando uma chave gerenciada pelo cliente, será possível conceder acesso para ler o segredo anexando a política a seguir a uma identidade. \

JSON

```
{
"Version":"2012-10-17",
"Statement": [
{
  "Effect": "Allow",
  "Action": "secretsmanager:GetSecretValue",
  "Resource": "arn:aws:secretsmanager:us-east-1:123456789012:secret:secretName-
AbCdEf"
},
{
  "Effect": "Allow",
```

```
  "Action": "kms:Decrypt",
  "Resource": "arn:aws:kms:us-east-1:123456789012:key/key-id"
}
]
}
```

Exemplo: permissão para ler e descrever segredos individuais

Example Ler e descrever um segredo

É possível conceder acesso a um segredo anexando a seguinte política a uma identidade.

JSON

```
{
"Version":"2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "secretsmanager:GetSecretValue",
      "secretsmanager:DescribeSecret"
      ],
      "Resource": "arn:aws:secretsmanager:us-
east-1:123456789012:secret:secretName-AbCdEf"
    }
]
}
```

Exemplo: permissão para recuperar um grupo de valores de segredos em um lote

Example Ler um grupo de segredos em um lote

É possível conceder acesso para recuperar um grupo de segredos em uma chamada de API em lote anexando a seguinte política a uma identidade. A política restringe o chamador para que ele só possa recuperar os segredos especificados por*SecretARN1*,, e *SecretARN2SecretARN3*, mesmo que a chamada em lote inclua outros segredos. Se o chamador também solicitar outros segredos na chamada de API em lote, o Secrets Manager não os retornará. Para obter mais informações, consulte BatchGetSecretValue. .

JSON

```
{
"Version":"2012-10-17",
"Statement": [
{
  "Effect": "Allow",
  "Action": [
    "secretsmanager:BatchGetSecretValue",
    "secretsmanager:ListSecrets"
  ],
  "Resource": "*"
},
{
  "Effect": "Allow",
  "Action": [
    "secretsmanager:GetSecretValue"
  ],
  "Resource": [
    "arn:aws:secretsmanager:us-east-1:123456789012:secret:secretName1-AbCdEf",
    "arn:aws:secretsmanager:us-east-1:123456789012:secret:secretName2-AbCdEf",
    "arn:aws:secretsmanager:us-east-1:123456789012:secret:secretName3-AbCdEf"
    ]
}
]
}
```

Exemplo: curingas

É possível usar curingas para incluir um conjunto de valores em um elemento de política.

Example Acessar todos os segredos em um caminho

A política a seguir concede acesso para recuperar todos os segredos com um nome que começa com "*TestEnv/*".

JSON

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

```
    "Statement": {
        "Effect": "Allow",
        "Action": "secretsmanager:GetSecretValue",
        "Resource": "arn:aws:secretsmanager:us-
east-1:123456789012:secret:TestEnv/*"
    }
}
```

Example Acessar metadados em todos os segredos

A política a seguir concede `DescribeSecret` e permissões, começando com `List`: `ListSecrets` e `ListSecretVersionIds`.

JSON

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

Example Correspondência de nome de segredo

A política a seguir concede todas as permissões do Secrets Manager para um segredo por nome. Para usar essa política, consulte the section called “Políticas baseadas em identidade”.

Para corresponder um nome de segredo, crie o ARN para o segredo reunindo a região, o ID da conta, o nome do segredo e o caractere curinga (?) para corresponder caracteres aleatórios individuais. O Secrets Manager anexa seis caracteres aleatórios a nomes de segredos como parte do ARN para que você possa usar esse curinga para corresponder a esses caracteres. Se você usar a sintaxe `"another_secret_name-*"`, o Secrets Manager corresponderá não apenas ao segredo previsto com os seis caracteres aleatórios, mas também a `"another_secret_name-<anything-here>a1b2c3"`.

Como é possível prever todas as partes do ARN de um segredo, exceto os seis caracteres aleatórios, o uso da sintaxe `'??????'` do caractere curinga permite que você conceda com segurança permissões a um segredo que ainda não existe. Mas observe que, se você excluir o segredo e recriá-lo com o mesmo nome, o usuário receberá automaticamente permissão para o novo segredo, mesmo que os 6 caracteres tenham sido alterados.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "secretsmanager:*",
            "Resource": [
                "arn:aws:secretsmanager:us-
east-1:123456789012:secret:a_specific_secret_name-a1b2c3",
                "arn:aws:secretsmanager:us-
east-1:123456789012:secret:another_secret_name-??????"
            ]
        }
    ]
}
```

Exemplo: Permissão para criar segredos

Para conceder permissões a um usuário para criar um segredo, recomendamos anexar uma política de permissões a um grupo do IAM ao qual o usuário pertence. Consulte Grupos de usuários do IAM.

Example Criar segredos

A política a seguir concede permissão para a criação de segredos e a visualização de uma lista de segredos. Para usar essa política, consulte the section called “Políticas baseadas em identidade”.

JSON

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

```
  "Effect": "Allow",
  "Action": [
    "secretsmanager:CreateSecret",
    "secretsmanager:ListSecrets"
  ],
  "Resource": "*"
}
]
}
```

Exemplo: negar uma AWS KMS chave específica para criptografar segredos

> **Important**
>
> Para negar uma chave gerenciada pelo cliente, recomendamos que você restrinja o acesso usando uma política de chaves ou uma concessão de chaves. Para obter mais informações, consulte Autenticação e controle de acesso para o AWS KMS no Guia do desenvolvedor do AWS Key Management Service .

Example Negar a chave AWS gerenciada **`aws/secretsmanager`**

A política a seguir nega o uso do Chave gerenciada pela AWS `aws/secretsmanager` para criar ou atualizar segredos. Essa política exige que os segredos sejam criptografados usando uma chave gerenciada pelo cliente. A política inclui duas instruções:

1. A primeira declaração,`Sid: "RequireCustomerManagedKeysOnSecrets"`, nega solicitações para criar ou atualizar segredos usando o. Chave gerenciada pela AWS `aws/secretsmanager`
2. A segunda declaração,`Sid: "RequireKmsKeyIdParameterOnCreate"`, nega solicitações para criar segredos que não incluam uma chave KMS, porque o Secrets Manager usaria por padrão o. Chave gerenciada pela 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"
            }
        }
    }
]
}
```

Políticas baseadas em recursos

Em uma política baseada em recursos, especifique quem pode acessar o segredo e as ações que essa pessoa pode executar no segredo. É possível usar políticas baseadas em recursos para:

- Conceder acesso a vários usuários e funções a um único segredo.
- Conceda acesso a usuários ou funções em outras AWS contas.

Quando você anexa uma política baseada em recursos a um segredo no console, o Secrets Manager usa o mecanismo de raciocínio automatizado Zelkova e a API `ValidateResourcePolicy` para impedir que você conceda acesso a seus segredos a uma ampla gama de entidades principais do IAM. Você também pode chamar a API `PutResourcePolicy` com o parâmetro `BlockPublicPolicy` da CLI ou do SDK.

Important

A validação da política de recursos e o parâmetro `BlockPublicPolicy` ajudam a proteger seus recursos, impedindo que o acesso público seja concedido por meio de políticas de recursos diretamente vinculadas aos segredos. Além de usar esses atributos, inspecione com cuidado as seguintes políticas para garantir que elas não concedam acesso público:

- Políticas baseadas em identidade vinculadas aos AWS diretores associados (por exemplo, funções do IAM)
- Políticas baseadas em recursos anexadas aos AWS recursos associados (por exemplo, chaves AWS Key Management Service (AWS KMS))

Para revisar as permissões de seus segredos, consulte Determinação de quem tem permissões para seus segredos do .

Para visualizar, alterar ou excluir a política de recursos de um segredo (console)

1. Abra o console do Secrets Manager em https://console.aws.amazon.com/secretsmanager/.
2. Na lista de segredos, escolha o segredo.
3. Na página de detalhes do segredo, na guia de Visão geral da seção Permissões de recursos, escolha Editar permissões.
4. No campo do código, siga um dos procedimentos a seguir e escolha Save (Salvar):
   - Para anexar ou modificar uma política de recursos, insira a política.
   - Para excluir a política, limpe o campo do código.

AWS CLI

Example Recuperar uma política de recursos

O exemplo de `get-resource-policy` a seguir recupera a política baseada em recurso anexada a um segredo.

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

Example Excluir uma política de recurso

O exemplo de `delete-resource-policy` a seguir exclui a política baseada em recurso anexada a um segredo.

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

Example Adicionar uma política de recurso

O exemplo de `put-resource-policy` a seguir adiciona uma política de permissões a um segredo, verificando primeiro se a política não fornece acesso amplo ao segredo. A política é lida de um arquivo. Para obter mais informações, consulte Carregando AWS CLI parâmetros de um arquivo no Guia AWS CLI do usuário.

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

Conteúdo 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

Para recuperar a política anexada a um segredo, use `GetResourcePolicy`.

Para excluir a política anexada a um segredo, use `DeleteResourcePolicy`.

Para anexar uma política a um segredo, use `PutResourcePolicy`. Se já houver uma política anexada, o comando a substituirá pela nova política. O documento da política deve estar formatado como texto JSON estruturado. Consulte Estrutura de documento de política JSON.

Para obter mais informações, consulte the section called “AWS SDKs”.

Exemplos

Exemplos:

- Exemplo: permissão para recuperar valores de segredos individuais
- Exemplo: permissões e VPCs
- Exemplo: entidade principal de serviço

Exemplo: permissão para recuperar valores de segredos individuais

Para conceder permissão para recuperar valores de segredos, é possível anexar políticas a segredos ou identidades. Para obter ajuda na determinação do tipo de política a ser usado, consulte Políticas baseadas em identidades e políticas baseadas em recursos. Para obter informações sobre como anexar uma política, consulte the section called “Políticas baseadas em recursos” e the section called “Políticas baseadas em identidade”.

Esse exemplo é útil quando você quer conceder acesso a um único segredo a vários usuários ou funções. Para conceder permissão para recuperar um grupo de segredos em uma chamada de API em lote, consulte the section called “Exemplo: permissão para recuperar um grupo de valores de segredos em um lote”.

Example Ler um segredo

É possível conceder acesso a um segredo anexando a seguinte política ao segredo.

JSON

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/EC2RoleToAccessSecrets"
            },
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "*"
        }
    ]
}
```

Exemplo: permissões e VPCs

Se você precisar acessar o Secrets Manager de uma VPC, verifique se as solicitações para o Secrets Manager vêm da VPC ao incluir uma condição nas políticas de permissões. Para obter mais informações, consulte Limite solicitações com condições do endpoint da VPC e the section called "Endpoints da VPC (AWS PrivateLink)".

Certifique-se de que as solicitações para acessar o segredo de outros AWS serviços também venham da VPC, caso contrário, essa política negará o acesso a elas.

Example Exigir que as solicitações sejam enviadas por meio de um endpoint da VPC

A política a seguir permite que um usuário execute operações do Secrets Manager somente quando a solicitação é fornecida por meio do endpoint do VPC *vpce-1234a5678b9012c*.

JSON

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

```
      "aws:sourceVpce": "vpce-12345678"
    }
  }
}
]
}
```

Example Exigir que as solicitações sejam provenientes de uma VPC

A política a seguir permite que os comandos criem e gerenciem segredos somente quando eles são provenientes da *vpc-12345678*. Além disso, a política permite operações que usam o acesso ao valor criptografado do segredo somente quando as solicitações são recebidas de vpc-2b2b2b2b. Será possível usar uma política como essa se uma aplicação for executada em uma VPC, mas você usa uma segunda VPC isolada para funções de gerenciamento.

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"
      }
    }
  }
  ]
  }
```

Exemplo: entidade principal de serviço

Se a política de recursos anexada ao seu segredo incluir um diretor de AWS serviço, recomendamos que você use as chaves de condição SourceAccount globais aws: SourceArn e aws:. O ARN e os valores da conta só são incluídos no contexto de autorização quando uma solicitação chega ao Secrets Manager diretamente de outro serviço da AWS . Essa combinação de condições evita um potencial confused deputy scenario (cenário de substituto confuso).

Se um ARN de recurso incluir caracteres que não sejam permitidos em uma política de recurso, você não poderá usar esse ARN de recurso no valor da chave de condição `aws:SourceArn`. Em vez disso, use a chave de condição `aws:SourceAccount`. Para obter mais informações, consulte os requisitos do IAM.

Os diretores de serviços normalmente não são usados como diretores em uma política anexada a um segredo, mas alguns AWS serviços exigem isso. Para obter informações sobre as políticas de recursos que um serviço exige que você anexe a um segredo, consulte a documentação do serviço.

Example Permitir que um serviço acesse um segredo usando uma entidade principal de serviço

JSON

```
{
"Version":"2012-10-17",
"Statement": [
{
  "Effect": "Allow",
  "Principal": {
    "Service": [
      "s3.amazonaws.com"
    ]
  },
  "Action": "secretsmanager:GetSecretValue",
  "Resource": "*",
  "Condition": {
    "ArnLike": {
      "aws:sourceArn": "arn:aws:s3::123456789012:*"
    },
    "StringEquals": {
      "aws:sourceAccount": "123456789012"
    }
  }

}
]
}
```

Controle o acesso a segredos usando o controle de acesso por atributo (ABAC)

Controle de acesso por atributo (ABAC) é uma estratégia de autorização que define permissões com base em atributos ou características do usuário, dos dados ou do ambiente, como o departamento, a unidade de negócios ou outros fatores que podem afetar o resultado da autorização. Em AWS, esses atributos são chamados de tags.

O uso de tags para controlar permissões é útil em ambientes que estão crescendo rapidamente e ajuda em situações em que o gerenciamento de políticas se torna um problema. As regras ABAC são avaliadas dinamicamente no runtime, o que significa que o acesso dos usuários às aplicações

e aos dados e o tipo de operações permitidas mudam automaticamente com base nos fatores contextuais da política. Por exemplo, se um usuário muda de departamento, o acesso é ajustado automaticamente sem a necessidade de atualizar as permissões ou solicitar novas funções. Para obter mais informações, consulte: Para AWS que serve o ABAC? , Defina permissões para acessar segredos com base em tags. e escale suas necessidades de autorização para o Secrets Manager usando o ABAC com o IAM Identity Center.

Exemplo: permitir que uma identidade acesse segredos que tenham tags específicas

A política a seguir permite o `DescribeSecret` acesso a segredos com uma tag com a chave *`ServerName`* e o valor*`ServerABC`*. Se você anexar essa política a uma identidade, a identidade terá permissão para qualquer segredo com essa tag na conta.

JSON

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

Exemplo: permitir o acesso somente a identidades com tags que correspondam às tags dos segredos

A política a seguir permite que qualquer identidade na conta `GetSecretValue` acesse qualquer segredo na conta em que a tag *`AccessProject`* da identidade tenha o mesmo valor que a tag *`AccessProject`* do segredo.

JSON

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

AWS política gerenciada para AWS Secrets Manager

Uma política AWS gerenciada é uma política autônoma criada e administrada por AWS. AWS as políticas gerenciadas são projetadas para fornecer permissões para muitos casos de uso comuns, para que você possa começar a atribuir permissões a usuários, grupos e funções.

Lembre-se de que as políticas AWS gerenciadas podem não conceder permissões de privilégio mínimo para seus casos de uso específicos porque estão disponíveis para uso de todos os AWS clientes. Recomendamos que você reduza ainda mais as permissões definindo as políticas gerenciadas pelo cliente que são específicas para seus casos de uso.

Você não pode alterar as permissões definidas nas políticas AWS gerenciadas. Se AWS atualizar as permissões definidas em uma política AWS gerenciada, a atualização afetará todas as identidades principais (usuários, grupos e funções) às quais a política está anexada. AWS é mais provável que atualize uma política AWS gerenciada quando uma nova AWS service (Serviço da AWS) é lançada ou novas operações de API são disponibilizadas para serviços existentes.

Para saber mais, consulte AWS Políticas gerenciadas pela no Guia do usuário do IAM.

AWS política gerenciada: SecretsManagerReadWrite

Essa política fornece read/write acesso a recursos do Amazon RDS AWS Secrets Manager, Amazon Redshift e Amazon DocumentDB, incluindo permissão para descrever recursos do Amazon RDS, Amazon Redshift e Amazon DocumentDB, além de permissão para AWS KMS usar para criptografar e descriptografar segredos. Essa política também fornece permissão para criar conjuntos de AWS CloudFormation alterações, obter modelos de rotação de um bucket do Amazon S3 gerenciado por AWS, listar AWS Lambda funções e descrever o Amazon EC2. VPCs Essas permissões são exigidas pelo console para configurar a alternância com as funções de rotação existentes.

Para criar novas funções de rotação, você também deve ter permissão para criar AWS CloudFormation pilhas e funções de AWS Lambda execução. Você pode atribuir a política gerenciada de IAMFullacesso. Consulte Permissões para alternância.

Detalhes das permissões

Esta política inclui as seguintes permissões.

- `secretsmanager`: permite que entidades principais realizem todas as ações do Secrets Manager.
- `cloudformation`— Permite que os diretores criem CloudFormation pilhas. Isso é necessário para que os diretores que usam o console para ativar a rotação possam criar funções CloudFormation de rotação do Lambda por meio de pilhas. Para obter mais informações, consulte the section called “Como o Secrets Manager usa CloudFormation”.
- `ec2`— Permite que os diretores descrevam o Amazon VPCs EC2. Isso é necessário para que as entidades principais que usam o console possam criar funções de alternância na mesma VPC do banco de dados das credenciais que estão armazenando em um segredo.
- `kms`— Permite que os diretores usem AWS KMS chaves para operações criptográficas. Isso é necessário para que o Secrets Manager possa criptografar e descriptografar segredos. Para obter mais informações, consulte the section called “Criptografia e descriptografia de segredos”.
- `lambda`: permite que as entidades principais listem as funções de alternância do Lambda. Isso é necessário para que as entidades principais que usam o console possam escolher as funções de alternância existentes.
- `rds`: permite que as entidades principais descrevam clusters e instâncias no Amazon RDS. Isso é necessário para que as entidades principais que usam o console possam escolher clusters ou instâncias do Amazon RDS.

- `redshift`: permite que as entidades principais descrevam clusters no Amazon Redshift. Isso é necessário para que as entidades principais que usam o console possam escolher clusters do Amazon Redshift.
- `redshift-serverless`: permite que as entidades principais descrevam namespaces no Amazon Redshift sem servidor. Isso é necessário para que as entidades principais que usam o console possam namespaces do Amazon Redshift sem servidor.
- `docdb-elastic`: permite que as entidades principais descrevam clusters elásticos no Amazon DocumentDB. Isso é necessário para que as entidades principais que usam o console possam escolher clusters elásticos do Amazon DocumentDB.
- `tag`: permite que as entidades principais obtenham todos os recursos marcados na conta.
- `serverlessrepo`— Permite que os diretores criem conjuntos de CloudFormation mudanças. Isso é necessário para que as entidades principais que usam o console possam criar funções de alternância do Lambda. Para obter mais informações, consulte the section called “Como o Secrets Manager usa CloudFormation”.
- `s3`— Permite que os diretores obtenham objetos de um bucket do Amazon S3 que é gerenciado pelo. AWS Esse bucket contém Modelos de função de alternância do Lambda. Essa permissão é necessária para que as entidades principais que usam o console possam criar funções de alternância do Lambda com base nos modelos no bucket. Para obter mais informações, consulte the section called “Como o Secrets Manager usa CloudFormation”.

Para ver a política, consulte o documento de política SecretsManagerReadWrite JSON.

AWS política gerenciada: AWSSecrets ManagerClientReadOnlyAccess

Essa política fornece acesso somente para leitura aos AWS Secrets Manager segredos dos aplicativos cliente. Ele permite que os diretores recuperem valores secretos e descrevam metadados secretos, junto com as AWS KMS permissões necessárias para descriptografar segredos criptografados com chaves gerenciadas pelo cliente.

Detalhes das permissões

Esta política inclui as seguintes permissões.

- `secretsmanager`— Permite que os diretores recuperem valores secretos e descrevam metadados secretos.

- `kms`— Permite que os diretores decifrem segredos usando chaves. AWS KMS Essa permissão tem como escopo as chaves usadas pelo Secrets Manager por meio de condições específicas do serviço.

Para visualizar mais detalhes sobre a política, inclusive a versão mais recente do documento de política JSON, consulte AWSSecretsManagerClientReadOnlyAccess no AWS Managed Policy Reference Guide.

Atualizações do Secrets Manager para políticas AWS gerenciadas

Veja detalhes sobre as atualizações das políticas AWS gerenciadas do Secrets Manager.

Alteração	Descrição	Data	Versão
AWSSecretsManagerClientReadOnlyAccess: Nova política gerenciada	O Secrets Manager criou uma nova política gerenciada para fornecer acesso somente de leitura aos segredos para aplicativos clientes. Essa política permite recuperar valores secretos e descrever metadados secretos, com as AWS KMS permissões necessárias para descriptografar segredos.	5 de novembro de 2025	v1
SecretsManagerReadWrite – atualização para uma política existente	Essa política foi atualizada para permitir a descrição do acesso ao Amazon Redshift sem servidor de forma que os usuários do console	12 de março de 2024	v5

Alteração	Descrição	Data	Versão
	possam escolher um namespace do Amazon Redshift sem servidor ao criarem um segredo do Amazon Redshift.		
SecretsManagerReadWrite – atualização para uma política existente	Essa política foi atualizada para permitir descrever o acesso aos clusters elásticos do Amazon DocumentDB para que os usuários do console possam escolher um cluster elástico ao criar um segredo do Amazon DocumentDB.	12 de setembro de 2023	v4

Alteração	Descrição	Data	Versão
SecretsManagerReadWrite – atualização para uma política existente	Essa política foi atualizada para permitir descrever o acesso ao Amazon Redshift para que os usuários do console possam escolher um cluster do Amazon Redshift ao criarem um segredo do Amazon Redshift. A atualização também adicionou novas permissões para permitir acesso de leitura a um bucket do Amazon S3 gerenciado pela empresa AWS que armazena os modelos de função de rotação do Lambda.	24 de junho de 2020	v3
SecretsManagerReadWrite – atualização para uma política existente	Essa política foi atualizada para permitir descrever o acesso aos clusters do Amazon RDS para que os usuários do console possam escolher um cluster ao criar um segredo do Amazon RDS.	3 de maio de 2018	v2 da

Alteração	Descrição	Data	Versão
SecretsManagerReadWrite – Nova política	O Secrets Manager criou uma política para conceder as permissões necessárias para usar o console com todo o read/write acesso ao Secrets Manager.	04 de abril de 2018	v1

Determine quem tem permissões para seus AWS Secrets Manager segredos

Por padrão, as identidades do IAM não têm permissão para acessar segredos. Ao autorizar o acesso a um segredo, o Secrets Manager avalia a política baseada em recursos anexada ao segredo e todas as políticas baseadas em identidades anexadas ao usuário ou à função do IAM que está enviando a solicitação. Para fazer isso, o Secrets Manager usa um processo semelhante ao descrito em Determinar se uma solicitação é permitida ou negada no Manual do usuário do IAM.

Quando várias políticas se aplicarem a uma solicitação, o Secrets Manager usará uma hierarquia para controlar as permissões:

1. Se uma declaração em qualquer política com uma `deny` explícita corresponder à ação de solicitação e ao recurso:

   A `deny` explícita substituirá todo o resto e bloqueará a ação.
2. Se não houver qualquer `deny` explícita, mas uma declaração com uma `allow` explícita corresponder à ação de solicitação e ao recurso:

   A `allow` explícita concederá à ação de solicitação acesso aos recursos da declaração.

   Se a identidade e o segredo estiverem em duas contas diferentes, deverá haver uma `allow` na política de recursos para o segredo e na política anexada à identidade, caso contrário, AWS negará a solicitação. Para obter mais informações, consulte Acesso entre contas.
3. Se não houver qualquer declaração com uma `allow` explícita que corresponda à ação de solicitação e ao recurso:

AWS nega a solicitação por padrão, o que é chamado de negação implícita.

Para visualizar a política baseada em recursos de um segredo

- Execute um destes procedimentos:
  - Abra o console do Secrets Manager em https://console.aws.amazon.com/secretsmanager/. Na página de detalhes do segredo, na seção Resource permissions (Permissões de recursos), escolha Edit permissions (Editar permissões).
  - Use o AWS CLI para ligar `get-resource-policy`ou o AWS SDK para ligar `GetResourcePolicy`.

Para determinar quem tem acesso por meio de políticas baseadas em identidades

- Use o simulador de políticas do IAM. Consulte Testar políticas do IAM com o simulador de políticas do IAM

Acesse AWS Secrets Manager segredos de uma conta diferente

Para permitir que os usuários de uma conta acessem segredos em outra conta (acesso entre contas), é necessário permitir o acesso em uma política de recursos e em uma política de identidade. Isso é diferente de conceder acesso a identidades na mesma conta do segredo.

A permissão entre contas é efetiva apenas nas seguintes operações:

- CancelRotateSecret
- DeleteResourcePolicy
- DeleteSecret
- DescribeSecret
- GetRandomPassword
- GetResourcePolicy
- GetSecretValue
- ListSecretVersionIds
- PutResourcePolicy
- PutSecretValue

- RemoveRegionsFromReplication
- ReplicateSecretToRegions
- RestoreSecret
- RotateSecret
- StopReplicationToReplica
- TagResource
- UntagResource
- UpdateSecret
- UpdateSecretVersionStage
- ValidateResourcePolicy

Você pode usar o `BlockPublicPolicy` parâmetro com a PutResourcePolicyação para ajudar a proteger seus recursos, impedindo que o acesso público seja concedido por meio das políticas de recursos diretamente associadas aos seus segredos. Você também pode usar o analisador de acesso do IAM para verificar o acesso entre contas.

Você também deve permitir que a identidade use a chave do KMS com a qual o segredo está criptografado. Isso ocorre porque você não pode usar o Chave gerenciada pela AWS (`aws/secretsmanager`) para acesso entre contas. Em vez disso, você precisa criptografar o segredo com uma chave do KMS que criar e, em seguida, anexar uma política de chave a ele. Existe uma cobrança pela criação de chaves do KMS. Para alterar a chave de criptografia de um segredo, consulte the section called “Modificar um segredo”.

> **Important**
>
> Políticas baseadas em recursos que concedem a permissão `secretsmanager:PutResourcePolicy` dão às entidades principais, mesmo aquelas em outras contas, a capacidade de modificar suas políticas baseadas em recursos. Essa permissão permite que as entidades principais escalem as permissões existentes, como obter acesso administrativo total aos segredos. Recomendamos que você aplique o princípio do acesso de privilégio mínimo às suas políticas. Para obter mais informações, consulte Políticas baseadas em recursos.

Os exemplos de políticas a seguir supõem que você tenha um segredo e uma chave de criptografia na Conta1 e uma identidade na Conta2, à qual você quer permitir acesso ao valor do segredo.

Etapa 1: anexar uma política de recursos ao segredo na Conta1

- A política a seguir permite que *Account2* a *ApplicationRole* entrada secreta acesse a entrada secreta*Account1*. Para usar essa política, consulte the section called “Políticas baseadas em recursos”.

  JSON

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

Etapa 2: adicionar uma instrução à política de chaves para a chave KMS na Conta1

- A instrução de política de chave a seguir permite que o *ApplicationRole* na *Account2* use a chave do KMS na *Account1* para descriptografar o segredo na *Account1*. Para usar essa declaração, adicione-a à política de chaves para sua chave do KMS. Para obter mais informações, consulte Alterar uma política de chaves.

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

```
  ],
  "Resource": "*"
}
```

Etapa 3: anexar uma política de identidade à identidade na Conta2

- A política a seguir permite que o *ApplicationRole* na *Account2* acesse o segredo na *Account1* e descriptografe o valor do segredo usando a chave de criptografia, que também está na *Account1*. Para usar essa política, consulte the section called "Políticas baseadas em identidade". É possível encontrar o ARN do seu segredo no console do Secrets Manager na página de detalhes do segredo em ARN do segredo. Como alternativa, é possível chamar `describe-secret`.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "arn:aws:secretsmanager:us-
east-1:123456789012:secret:secretName-AbCdEf"
        },
        {
            "Effect": "Allow",
            "Action": "kms:Decrypt",
            "Resource": "arn:aws:kms:us-
east-1:123456789012:key/EncryptionKey"
        }
    ]
}
```

Acesso a segredos a partir de um ambiente on-premises

Você pode usar o AWS Identity and Access Management Roles Anywhere para obter credenciais de segurança temporárias no IAM para cargas de trabalho, como servidores, contêineres e aplicativos executados fora do. AWS Suas cargas de trabalho podem usar as mesmas políticas e funções do

IAM que você usa com AWS aplicativos para acessar AWS recursos. Com o IAM Roles Anywhere, é possível usar o Secrets Manager para armazenar e gerenciar credenciais que podem ser acessadas por recursos no AWS , bem como por dispositivos on-premises, como servidores de aplicações. Para obter mais informações, consulte o Guia do usuário do IAM Roles Anywhere.

Proteção de dados em AWS Secrets Manager

O modelo de responsabilidade AWS compartilhada modelo se aplica à proteção de dados em AWS Secrets Manager. Conforme descrito neste modelo, AWS é responsável por proteger a infraestrutura global que executa todos os Nuvem AWS. Você é responsável por manter o controle sobre seu conteúdo hospedado nessa infraestrutura. Esse conteúdo inclui as tarefas de configuração e gerenciamento de segurança dos Serviços da AWS que você usa. Para obter mais informações sobre a privacidade de dados, consulte as Perguntas Frequentes sobre Privacidade de Dados.. Para saber mais sobre a proteção de dados na Europa, consulte a postagem do blog AWS Shared Responsibility Model and RGPD no Blog de segurança da AWS .

Para fins de proteção de dados, recomendamos que você proteja Conta da AWS as credenciais e configure contas de usuário individuais com AWS Identity and Access Management (IAM). Dessa maneira, cada usuário receberá apenas as permissões necessárias para cumprir suas obrigações de trabalho. Recomendamos também que você proteja seus dados das seguintes formas:

- Use uma autenticação multifator (MFA) com cada conta.
- Use SSL/TLS para se comunicar com AWS os recursos. O Secrets Manager é compatível com o TLS 1.2 e 1.3 em todas as regiões. O Secrets Manager também é compatível com um protocolo híbrido de criptografia de rede com opção de troca de chave pós-quântica para TLS (PQTLS).
- Assine suas solicitações programáticas para o Secrets Manager usando um ID da chave de acesso e uma chave de acesso secreta associados a uma entidade principal do IAM. Você também pode usar o AWS Security Token Service (AWS STS) para gerar credenciais de segurança temporárias para assinar solicitações.
- Configure a API e o registro de atividades do usuário com AWS CloudTrail. Consulte the section called “Faça login com AWS CloudTrail ”.
- Se você precisar de módulos criptográficos validados pelo FIPS 140-2 ao acessar AWS por meio de uma interface de linha de comando ou de uma API, use um endpoint FIPS. Consulte the section called “Endpoint do Secrets Manager”.
- Se você usar o AWS CLI para acessar o Secrets Manager,the section called “Mitigue os riscos de usar o AWS CLI para armazenar seus segredos AWS Secrets Manager”.

Criptografia em repouso

O Secrets Manager usa criptografia via AWS Key Management Service (AWS KMS) para proteger a confidencialidade dos dados em repouso. AWS KMS fornece um serviço de criptografia e armazenamento de chaves usado por muitos AWS serviços. Cada segredo no Secrets Manager é criptografado com uma chave de dados exclusiva. Cada chave de dados é protegida por uma chave KMS. É possível optar por usar a criptografia padrão com a Chave gerenciada pela AWS do Secrets Manager para a conta ou pode criar sua própria chave gerenciada pelo cliente no AWS KMS. O uso de uma chave gerenciada pelo cliente oferece controles de autorização mais granulares sobre as atividades da chave KMS. Para obter mais informações, consulte the section called “Criptografia e descriptografia de segredos”.

Criptografia em trânsito

O Secrets Manager fornece endpoints seguros e privados para criptografar dados em trânsito. Os endpoints seguros e privados permitem proteger AWS a integridade das solicitações de API ao Secrets Manager. AWS exige que as chamadas de API sejam assinadas pelo chamador usando certificados X.509, uma chave de acesso secreta and/or do Secrets Manager. Esse requisito é indicado no Processo de assinatura do Signature versão 4 (Sigv4).

Se você usar o AWS Command Line Interface (AWS CLI) ou qualquer um dos AWS SDKs para fazer chamadas AWS, você configura a chave de acesso a ser usada. Em seguida, essas ferramentas usam automaticamente a chave de acesso para assinar as solicitações para você. Consulte the section called “Mitigue os riscos de usar o AWS CLI para armazenar seus segredos AWS Secrets Manager”.

Privacidade do tráfego entre redes

AWS oferece opções para manter a privacidade ao rotear o tráfego por meio de rotas de rede conhecidas e privadas.

Tráfego entre clientes de serviço e on-premises e as aplicações

Você tem duas opções de conectividade entre sua rede privada e AWS Secrets Manager:

- Uma conexão AWS Site-to-Site VPN. Para obter mais informações, consulte O que é AWS VPN Site-to-Site?
- Uma conexão AWS Direct Connect. Para obter mais informações, consulte O que é o AWS Direct Connect?

Tráfego entre AWS recursos na mesma região

Se você quiser proteger o tráfego entre o Secrets Manager e os clientes da API AWS, configure um AWS PrivateLinkpara acessar de forma privada os endpoints da API do Secrets Manager.

Gerenciamento das chaves de criptografia

Quando o Secrets Manager precisa criptografar uma nova versão dos dados secretos protegidos, o Secrets Manager envia uma solicitação AWS KMS para gerar uma nova chave de dados a partir da chave KMS. O Secrets Manager usa essa chave de dados para criptografia de envelope. O Secrets Manager armazena a chave de dados criptografada com o segredo criptografado. Quando o segredo precisa ser descriptografado, o Secrets Manager solicita AWS KMS a decodificação da chave de dados. Em seguida, o Secrets Manager usa a chave de dados descriptografada para descriptografar o segredo criptografado. O Secrets Manager nunca armazena a chave de dados em formato não criptografado e remove a chave da memória o mais rápido possível. Para obter mais informações, consulte the section called “Criptografia e descriptografia de segredos”.

Criptografia e decodificação secretas em AWS Secrets Manager

O Secrets Manager usa criptografia envelopada com chaves do AWS KMS e chaves de dados para proteger o valor de cada segredo. Sempre que o valor do segredo muda, o Secrets Manager solicita uma nova chave de dados do AWS KMS para protegê-lo. A chave de dados é criptografada em uma chave do KMS e é armazenada nos metadados do segredo. Para descriptografar o segredo, o Secrets Manager primeiro descriptografa a chave de dados criptografada usando a chave KMS em. AWS KMS

O Secrets Manager não usa a chave do KMS para criptografar o valor do segredo diretamente. Em vez disso, ele usa a chave do KMS para gerar e criptografar uma chave de dados simétrica de Padrão de criptografia avançada (AES) de 256 bits e usa a chave de dados para criptografar o valor do segredo. O Secrets Manager usa a chave de dados de texto simples para criptografar o valor secreto externo e AWS KMS, em seguida, o remove da memória. Ele armazena a cópia criptografada da chave de dados nos metadados do segredo.

Tópicos

- Escolhendo uma AWS KMS chave
- O que é criptografado?
- Processos de criptografia e descriptografia

- Permissões para a chave do KMS
- Como o Secrets Manager usa a chave do KMS
- Política chave do Chave gerenciada pela AWS (aws/secretsmanager)
- Contexto de criptografia do Secrets Manager
- Monitore a interação do Secrets Manager com AWS KMS

Escolhendo uma AWS KMS chave

Ao criar um segredo, você pode escolher qualquer chave de criptografia simétrica gerenciada pelo cliente na região Conta da AWS e, ou você pode usar o Chave gerenciada pela AWS for Secrets Manager (`aws/secretsmanager`). Se você escolher o Chave gerenciada pela AWS `aws/secretsmanager` e ele ainda não existe, o Secrets Manager o cria e o associa ao segredo. É possível usar a mesma chave do KMS ou diferentes chaves do KMS para cada segredo na sua conta. Talvez você queira usar chaves KMS diferentes para definir permissões personalizadas nas chaves para um grupo de segredos ou se quiser auditar operações específicas para essas chaves. O Secrets Manager só oferece suporte a chaves de criptografia simétricas do KMS. Se você usar uma chave KMS em um armazenamento de chaves externas, as operações criptográficas na chave KMS podem levar mais tempo e ter menos confiabilidade e durabilidade, pois a solicitação precisa sair da AWS.

Para obter informações sobre alterar a chave de criptografia de um segredo, consulte the section called “Altere a chave de criptografia de um segredo”.

Quando você altera a chave de criptografia, o Secrets Manager criptografa novamente as versões `AWSCURRENT`, `AWSPENDING` e `AWSPREVIOUS` com a nova chave. Para evitar que você fique bloqueado sem o segredo, o Secrets Manager mantém todas as versões existentes criptografadas com a chave anterior. Isso significa que é possível descriptografar as versões `AWSCURRENT`, `AWSPENDING` e `AWSPREVIOUS` com a chave anterior ou com a nova chave. Se você não tiver a permissão `kms:Decrypt` para a chave anterior, ao alterar a chave de criptografia, o Secrets Manager não poderá descriptografar as versões do segredo para recriptografá-las. Nesse caso, as versões existentes não serão criptografadas novamente.

Para fazer com que `AWSCURRENT` só possa ser descriptografado pela nova chave de criptografia, crie uma nova versão do segredo com a nova chave. Em seguida, para poder decifrar a versão do segredo `AWSCURRENT`, será necessário ter permissão para a nova chave.

Você pode negar a permissão Chave gerenciada pela AWS `aws/secretsmanager` e exigir que os segredos sejam criptografados com uma chave gerenciada pelo cliente. Para obter mais informações, consulte the section called “Exemplo: negar uma AWS KMS chave específica para criptografar segredos”.

Para encontrar a chave KMS associada a um segredo, visualize o segredo no console ou ligue para ListSecretsou DescribeSecret. Quando o segredo é associado ao Chave gerenciada pela AWS for Secrets Manager (`aws/secretsmanager`), essas operações não retornam um identificador de chave KMS.

O que é criptografado?

O Secrets Manager criptografa o valor do segredo, mas não criptografa o seguinte:

- Nome e descrição do segredo
- Configurações de alternância
- ARN da chave do KMS associada ao segredo
- Qualquer AWS etiqueta anexada

Processos de criptografia e descriptografia

Para criptografar o valor de um segredo, o Secrets Manager usa o processo descrito a seguir.

1. O Secrets Manager chama a AWS KMS GenerateDataKeyoperação com o ID da chave KMS do segredo e uma solicitação de uma chave simétrica AES de 256 bits. AWS KMS retorna uma chave de dados em texto simples e uma cópia dessa chave de dados criptografada sob a chave KMS.
2. O Secrets Manager usa a chave de dados de texto simples e o algoritmo Advanced Encryption Standard (AES) para criptografar o valor secreto externo. AWS KMS Ele remove a chave de texto simples da memória o mais rápido possível após o uso.
3. O Secrets Manager armazena a chave de dados criptografada nos metadados do segredo para que fique disponível para descriptografar o valor do segredo. No entanto, nenhum dos Secrets Manager APIs retorna o segredo criptografado ou a chave de dados criptografada.

Para descriptografar um valor do segredo criptografado:

1. O Secrets Manager chama a operação AWS KMS Decrypt e passa a chave de dados criptografada.

2. AWS KMS usa a chave KMS do segredo para descriptografar a chave de dados. Ele gera a chave de dados de texto simples.
3. O Secrets Manager usa a chave de dados de texto simples para descriptografar o valor do segredo. Então, ele remove a chave de dados da memória o mais rápido possível.

Permissões para a chave do KMS

Quando o Secrets Manager usa uma chave do KMS em operações de criptografia, ele atua em nome do usuário que está acessando ou atualizando o valor do segredo. É possível conceder permissões em uma política do IAM ou em uma política de chave. As seguintes operações do Secrets Manager exigem AWS KMS permissões.

- CreateSecret
- GetSecretValue
- PutSecretValue
- UpdateSecret
- ReplicateSecretToRegions

Para permitir que a chave KMS seja usada somente para solicitações originadas no Secrets Manager, na política de permissões, você pode usar a chave kms: ViaService condition com o valor. `secretsmanager.<Region>.amazonaws.com`

Também é possível usar as chaves ou valores no contexto de criptografia como condição para usar a chave do KMS em operações de criptografia. Por exemplo, é possível usar um operador de condição de string em um IAM ou documento de política de chaves ou usar uma restrição de concessão em uma concessão. A propagação da concessão da chave do KMS pode demorar até cinco minutos. Para obter mais informações, consulte CreateGrant.

Como o Secrets Manager usa a chave do KMS

O Secrets Manager chama as seguintes AWS KMS operações com sua chave KMS.

GenerateDataKey

O Secrets Manager chama a AWS KMS GenerateDataKeyoperação em resposta às seguintes operações do Secrets Manager.

- CreateSecret— Se o novo segredo incluir um valor secreto, o Secrets Manager solicitará uma nova chave de dados para criptografá-lo.
- PutSecretValue— O Secrets Manager solicita uma nova chave de dados para criptografar o valor secreto especificado.
- ReplicateSecretToRegions— Para criptografar o segredo replicado, o Secrets Manager solicita uma chave de dados para a chave KMS na região da réplica.
- UpdateSecret— Se você alterar o valor secreto ou a chave KMS, o Secrets Manager solicitará uma nova chave de dados para criptografar o novo valor secreto.

A RotateSecretoperação não chama`GenerateDataKey`, porque não altera o valor secreto. No entanto, se o `RotateSecret` invoca a função do Lambda que muda o valor do segredo, sua chamada para a operação `PutSecretValue` acionará uma solicitação `GenerateDataKey`.

Decrypt

O Secrets Manager chama a operação Decrypt em resposta às seguintes operações do Secrets Manager.

- GetSecretValuee BatchGetSecretValue— Secrets Manager decifra o valor secreto antes de devolvê-lo ao chamador. Para descriptografar um valor secreto criptografado, o Secrets Manager chama a operação AWS KMS Decrypt para descriptografar a chave de dados criptografada no segredo. Ele usa a chave de dados de texto simples para descriptografar o valor do segredo criptografado. Para comandos em lote, o Secrets Manager pode reutilizar a chave descriptografada; portanto, nem todas as chamadas resultam em uma solicitação `Decrypt`.
- PutSecretValuee UpdateSecret— A maioria das `UpdateSecret` solicitações de `PutSecretValue` e não aciona uma `Decrypt` operação. No entanto, quando uma solicitação `PutSecretValue` ou `UpdateSecret` tenta alterar o valor do segredo em uma versão existente de um segredo, o Secrets Manager descriptografa o valor do segredo existente e o compara com o valor do segredo da solicitação para confirmar se são iguais. Essa ação garante que as operações do Secrets Manager sejam idempotentes. Para descriptografar um valor secreto criptografado, o Secrets Manager chama a operação AWS KMS Decrypt para descriptografar a chave de dados criptografada no segredo. Ele usa a chave de dados de texto simples para descriptografar o valor do segredo criptografado.
- ReplicateSecretToRegions— O Secrets Manager primeiro descriptografa o valor secreto na região primária antes de criptografar novamente o valor secreto com a chave KMS na região de réplica.

Encrypt

O Secrets Manager chama a operação Encrypt em resposta às seguintes operações do Secrets Manager:

- UpdateSecret— Se você alterar a chave KMS, o Secrets Manager criptografará novamente a chave de dados que protege as versões `AWSCURRENTAWSPREVIOUS`, e `AWSPENDING` secretas com a nova chave.

DescribeKey

O Secrets Manager chama a DescribeKeyoperação para determinar se a chave KMS deve ser listada ao criar ou editar um segredo no console do Secrets Manager.

Validar o acesso à chave do KMS

Ao estabelecer ou alterar a chave do KMS associada ao segredo, o Secrets Manager chama as operações `GenerateDataKey` e `Decrypt` com a chave do KMS especificada. Essas chamadas confirmam que o autor da chamada tem permissão para usar a chave do KMS para essas operações. O Secrets Manager descarta os resultados dessas operações; ele não os usa em qualquer operação de criptografia.

É possível identificar essas chamadas de validação, pois o valor da chave `SecretVersionId` com contexto de criptografia nessas solicitações é `RequestToValidateKeyAccess`.

> **Note**
>
> No passado, as chamadas de validação do Secrets Manager não incluíam um contexto de criptografia. Você pode encontrar chamadas sem contexto de criptografia em AWS CloudTrail registros mais antigos.

Política chave do Chave gerenciada pela AWS (**`aws/secretsmanager`**)

A política de chaves do Chave gerenciada pela AWS for Secrets Manager (`aws/secretsmanager`) dá aos usuários permissão para usar a chave KMS para operações específicas somente quando o Secrets Manager faz a solicitação em nome do usuário. A política de chaves não permite que os usuários utilizem a chave do KMS diretamente.

Essa política de chaves, como as políticas de todas as Chaves gerenciadas pela AWS, é estabelecida pelo serviço. Não é possível alterar a política de chaves, mas é possível visualizá-la a qualquer momento. Para obter mais detalhes, consulte Visualizar uma política de chaves.

As declarações de política na política de chaves têm os seguintes efeitos:

- Permita que os usuários da conta usem a chave do KMS para operações de criptografia somente quando a solicitação for proveniete do Secrets Manager em seu nome. A chave de condição `kms:ViaService` impõe essa restrição.
- Permite que a AWS conta crie políticas do IAM que permitem aos usuários visualizar as propriedades da chave KMS e revogar concessões.
- Embora o Secrets Manager não use concessões para obter acesso à chave do KMS, a política também permite que o Secrets Manager crie concessões para a chave do KMS em nome do usuário e permite que a conta revogue qualquer concessão que permite que o Secrets Manager use a chave do KMS. Esses são elementos padrão do documento de política para um Chave gerenciada pela AWS.

A seguir está uma política fundamental para um exemplo Chave gerenciada pela AWS para o 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": "*"
    }
  ]
```

```
}
```

Contexto de criptografia do Secrets Manager

Um contexto de criptografia é um conjunto de pares de chave-valor que contêm dados arbitrários não secretos. Quando você inclui um contexto de criptografia em uma solicitação para criptografar dados, vincula AWS KMS criptograficamente o contexto de criptografia aos dados criptografados. Para descriptografar os dados, é necessário passar o mesmo contexto de criptografia.

Em suas solicitações GenerateDataKeye Decrypt para, o Secrets AWS KMS Manager usa um contexto de criptografia com dois pares de nome e valor que identificam o segredo e sua versão, conforme mostrado no exemplo a seguir. Os nomes não variam, mas os valores de contexto de criptografia combinados serão diferentes para cada valor de segredo.

```
"encryptionContext": {
    "SecretARN": "arn:aws:secretsmanager:us-east-2:111122223333:secret:test-secret-
a1b2c3",
    "SecretVersionId": "EXAMPLE1-90ab-cdef-fedc-ba987SECRET1"
}
```

Você pode usar o contexto de criptografia para identificar essas operações criptográficas em registros e registros de auditoria, como AWS CloudTrailAmazon CloudWatch Logs, e como condição para autorização em políticas e concessões.

O contexto de criptografia do Secrets Manager consiste em dois pares de nome e valor.

- SecretARN: o primeiro par de nome e valor identifica o segredo. A chave é `SecretARN`. O valor é o nome de recurso da Amazon (ARN) do segredo.

  ```
  "SecretARN": "ARN of an Secrets Manager secret"
  ```

  Por exemplo, se o ARN do segredo fosse `arn:aws:secretsmanager:us-east-2:111122223333:secret:test-secret-a1b2c3`, o contexto de criptografia incluiria o seguinte par.

  ```
  "SecretARN": "arn:aws:secretsmanager:us-east-2:111122223333:secret:test-secret-
  a1b2c3"
  ```

- SecretVersionId— O segundo par nome-valor identifica a versão do segredo. A chave é `SecretVersionId`. O valor é o ID da versão.

  ```
  "SecretVersionId": "<version-id>"
  ```

  Por exemplo, se o ID de versão do segredo fosse `EXAMPLE1-90ab-cdef-fedc-ba987SECRET1`, o contexto de criptografia incluiria o seguinte par.

  ```
  "SecretVersionId": "EXAMPLE1-90ab-cdef-fedc-ba987SECRET1"
  ```

Quando você estabelece ou altera a chave KMS de um segredo, o Secrets Manager envia GenerateDataKeye descriptografa solicitações para AWS KMS validar se o chamador tem permissão para usar a chave KMS para essas operações. Ele descarta as respostas; não as usa no valor do segredo.

Nessas solicitações de validação, o valor do `SecretARN` é o ARN real do segredo, mas o valor `SecretVersionId` é `RequestToValidateKeyAccess`, como mostrado no seguinte exemplo de contexto de criptografia. Esse valor especial ajuda você a identificar solicitações de validação em logs e trilhas de auditoria.

```
"encryptionContext": {
    "SecretARN": "arn:aws:secretsmanager:us-east-2:111122223333:secret:test-secret-
a1b2c3",
    "SecretVersionId": "RequestToValidateKeyAccess"
}
```

> **Note**
>
> No passado, as solicitações de validação do Secrets Manager não incluíam um contexto de criptografia. Você pode encontrar chamadas sem contexto de criptografia em AWS CloudTrail registros mais antigos.

Monitore a interação do Secrets Manager com AWS KMS

Você pode usar o AWS CloudTrail Amazon CloudWatch Logs para rastrear as solicitações que o Secrets Manager envia AWS KMS em seu nome. Para obter mais informações sobre o monitoramento do uso de segredos, consulte Monitorar segredos.

GenerateDataKey

Quando você cria ou altera o valor secreto em um segredo, o Secrets Manager envia uma GenerateDataKeysolicitação AWS KMS que especifica a chave KMS do segredo.

O evento que registra a operação `GenerateDataKey` é semelhante ao evento de exemplo a seguir. A solicitação foi invocada por `secretsmanager.amazonaws.com`. Os parâmetros incluem o nome do recurso da Amazon (ARN) da chave do KMS do segredo, um especificador de chaves que requer uma chave de 256 bits e o contexto de criptografia que identifica o segredo e a versão.


```
{
    "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

Quando você obtém ou altera o valor secreto de um segredo, o Secrets Manager envia uma solicitação de descriptografia para descriptografar AWS KMS a chave de dados criptografada. Para comandos em lote, o Secrets Manager pode reutilizar a chave descriptografada; portanto, nem todas as chamadas resultam em uma solicitação `Decrypt`.

O evento que registra a operação `Decrypt` é semelhante ao evento de exemplo a seguir. O usuário é o principal da sua AWS conta que está acessando a tabela. Os parâmetros incluem a chave da tabela criptografada (como um blob de texto cifrado) e o contexto de criptografia que identifica a tabela e a conta. AWS AWS KMS deriva o ID da chave KMS do texto cifrado.

```
{
    "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

Quando você altera a chave KMS associada a um segredo, o Secrets Manager envia uma solicitação Encrypt para AWS KMS recriptografar as versões`AWSCURRENT`,`AWSPREVIOUS`, e `AWSPENDING` secretas com a nova chave. Quando você replica um segredo para outra região, o Secrets Manager também envia uma solicitação Encrypt para o AWS KMS.

O evento que registra a operação `Encrypt` é semelhante ao evento de exemplo a seguir. O usuário é o principal da sua AWS conta que está acessando a tabela.


```
{
    "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"
}
```

Segurança da infraestrutura em AWS Secrets Manager

Como serviço gerenciado, AWS Secrets Manager é protegido pela segurança de rede AWS global. Para obter informações sobre serviços AWS de segurança e como AWS proteger a infraestrutura, consulte AWS Cloud Security. Para projetar seu AWS ambiente usando as melhores práticas de segurança de infraestrutura, consulte Proteção de infraestrutura no Security Pillar AWS Well-Architected Framework.

O acesso ao Secrets Manager pela rede é AWS publicado APIs usando TLS. O Secrets Manager APIs pode ser chamado de qualquer local da rede. No entanto, o Secrets Manager é compatível com políticas de acesso com base em recursos, que podem incluir restrições com base no endereço IP de origem. Você também pode usar as políticas de recursos do Secrets Manager para controlar o acesso a segredos de endpoints específicos de nuvem privada virtual (VPC) ou específicos. VPCs Efetivamente, isso isola o acesso à rede a um determinado segredo somente da VPC específica dentro da AWS rede. Para obter mais informações, consulte the section called “Endpoints da VPC (AWS PrivateLink)”.

Usando um AWS Secrets Manager VPC endpoint

Sempre que possível, recomendamos que você execute o máximo da infraestrutura em redes privadas que não sejam acessíveis pela Internet pública. É possível estabelecer uma conexão privada entre a sua VPC e o Secrets Manager criando um interface VPC endpoint (Endpoint da VPC de interface). Os endpoints de interface são alimentados por AWS PrivateLinkuma tecnologia que permite acessar o Secrets Manager de forma privada APIs sem um gateway de internet, dispositivo NAT, conexão VPN ou conexão. Direct Connect As instâncias em sua VPC não precisam de endereços IP públicos para se comunicar com o Secrets Manager. APIs O tráfego entre sua VPC e o Secrets Manager não sai da AWS rede. Para obter mais informações, consulte Endpoints da VPC da interface (AWS PrivateLink) no Manual do Usuário do Amazon VPC.

Quando o Secrets Manager alterna um segredo usando uma função de alternância do Lambda, por exemplo, um segredo que contém credenciais de banco de dados, a função do Lambda faz solicitações para o banco de dados e para o Secrets Manager. Quando você ativa a alternância automática usando o console, o Secrets Manager cria a função do Lambda na mesma VPC do banco de dados. Recomendamos que você crie um endpoint do Secrets Manager na mesma VPC para que as solicitações da função de alternância do Lambda para o Secrets Manager não saiam da rede da Amazon.

Se você habilitar o DNS privado para o endpoint, poderá fazer solicitações de API para o Secrets Manager usando seu nome DNS padrão para a região, por exemplo, `secretsmanager.us-east-1.amazonaws.com`. Para mais informações, consulte Acessar um serviço por um endpoint de interface no Guia do usuário da Amazon VPC.

É possível verificar se as solicitações para o Secrets Manager vêm do acesso à VPC ao incluir uma condição nas políticas de permissões. Para obter mais informações, consulte the section called "Exemplo: permissões e VPCs".

Você pode usar AWS CloudTrail registros para auditar o uso de segredos por meio do VPC endpoint.

Para criar um endpoint da VPC do Secrets Manager

1. Consulte Criação de um endpoint de interface no Guia do usuário da Amazon VPC. Use um dos nomes de serviço a seguir:
   - `com.amazonaws.region.secretsmanager`
   - `com.amazonaws.region.secretsmanager-fips`
2. Para controlar o acesso ao endpoint, consulte Controle o acesso aos endpoints da VPC usando políticas de endpoint.
3. Para usar IPv6 um endereçamento de pilha dupla, consulte. IPv4 e IPv6 acesso

Criar uma política de endpoint para o endpoint de interface

Uma política de endpoint é um recurso do IAM que pode ser anexado ao endpoint de interface. A política de endpoint padrão permite acesso total ao Secrets Manager por meio de endpoints de interface. Para controlar o acesso permitido ao Secrets Manager pela VPC, anexe uma política de endpoint personalizada ao endpoint da interface.

Uma política de endpoint especifica as seguintes informações:

- As entidades principais que podem realizar ações (Contas da AWS, usuários do IAM e perfis do IAM).
- As ações que podem ser realizadas.
- Os recursos nos quais as ações podem ser executadas.

Para obter mais informações, consulte Controlar o acesso aos serviços usando políticas de endpoint no Guia do AWS PrivateLink .

Exemplo: política de endpoint da VPC para ações do Secrets Manager

Veja a seguir um exemplo de uma política de endpoint personalizado. Quando você vincula essa política ao endpoint da sua interface, ela concede acesso às ações do Secrets Manager listadas no segredo especificado.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "Allow all users to use GetSecretValue and DescribeSecret on the
 specified secret.",
      "Effect": "Allow",
      "Principal": "*",
      "Action": [
        "secretsmanager:GetSecretValue",
        "secretsmanager:DescribeSecret"
      ],
      "Resource": "arn:aws:secretsmanager:us-
east-1:111122223333:secret:secretName-AbCdEf"
    }
  ]
}
```

Sub-redes compartilhadas

Você não pode criar, descrever, modificar ou excluir endpoints da VPC em sub-redes que são compartilhadas com você. No entanto, é possível usar os endpoints da VPC em sub-redes que são

compartilhadas com você. Para obter informações sobre o compartilhamento de VPC, consulte Compartilhar sua VPC com outras contas no Guia do usuário do Amazon Virtual Private Cloud.

Controlar o acesso à API com políticas do IAM

Se você usa políticas do IAM para controlar o acesso Serviços da AWS com base em endereços IP, talvez seja necessário atualizar suas políticas para incluir intervalos de IPv6 endereços. Este guia explica as diferenças entre IPv4 e IPv6 e descreve como atualizar suas políticas do IAM para oferecer suporte a ambos os protocolos. A implementação dessas mudanças ajuda você a manter o acesso seguro aos seus AWS recursos e, ao mesmo tempo, oferecer suporte IPv6.

O que é IPv6?

IPv6 é o padrão IP de próxima geração destinado a ser substituído eventualmente IPv4. A versão anterior, IPv4, usa um esquema de endereçamento de 32 bits para suportar 4,3 bilhões de dispositivos. IPv6 em vez disso, usa endereçamento de 128 bits para suportar aproximadamente 340 trilhões de trilhões de trilhões (ou 2 até a 128ª potência) de dispositivos.

Para obter mais informações, consulte a página da IPv6Web sobre VPC.

Estes são exemplos de IPv6 endereços:

```
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.
```

Políticas de pilha dupla (IPv4 e) do IAM IPv6

Você pode usar políticas do IAM para controlar o acesso ao Secrets Manager APIs e impedir que endereços IP fora do intervalo configurado acessem o Secrets Manager APIs.

O gerente de segredos. O endpoint de pilha dupla {region} .amazonaws.com para Secrets Manager é compatível com e. APIs IPv6 IPv4

Se você precisar oferecer suporte a ambos IPv6, IPv4 atualize suas políticas de filtragem de endereços IP para lidar com IPv6 endereços. Caso contrário, talvez você não consiga se conectar ao Secrets Manager IPv6.

Quem deve fazer essa alteração?

Essa alteração afetará você caso use o endereçamento duplo com políticas que contenham `aws:sourceIp`. O endereçamento duplo significa que a rede oferece suporte tanto para IPv4 quanto IPv6.

Se você usa endereçamento duplo, atualize suas políticas do IAM que atualmente usam endereços de IPv4 formato para incluir endereços de IPv6 formato.

Quem não deve fazer essa alteração?

Essa alteração não afetará você se você usar apenas IPv4 redes.

Adicionando IPv6 a uma política do IAM

As políticas do IAM usam a chave de condição `aws:SourceIp` para controlar o acesso a partir de endereços IP específicos. Se sua rede usa endereçamento duplo (IPv4 e IPv6), atualize suas políticas do IAM para incluir intervalos de IPv6 endereços.

No elemento `Condition` das suas políticas, use os operadores `IpAddress` e `NotIpAddress` para condições de endereço IP. Não use operadores de string, pois eles não conseguem lidar com os vários formatos de IPv6 endereço válidos.

Estes exemplos usam `aws:SourceIp`. Para VPCs, use `aws:VpcSourceIp` em vez disso.

A seguir está a política de referência de negação de acesso AWS com base na política de referência de IP de origem do Guia do usuário do IAM. O `NotIpAddress` `Condition` elemento a lista dois intervalos de IPv4 endereços `192.0.2.0/24` e`203.0.113.0/24`, aos quais será negado o acesso à API.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Deny",
        "Action": "*",
        "Resource": "*",
        "Condition": {
            "NotIpAddress": {
```

```
                "aws:SourceIp": [
                    "192.0.2.0/24",
                    "203.0.113.0/24"
                ]
            },
            "Bool": {
                "aws:ViaAWSService": "false"
            }
        }
    }
}
```

Para atualizar essa política, altere o `Condition` elemento para incluir os intervalos de IPv6 endereços `2001:DB8:1234:5678::/64` `2001:cdba:3257:8593::/64` e.

> **Note**
>
> Não remova os IPv4 endereços existentes. Eles são necessários para retrocompatibilidade.

```
"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"
                }
            }
```

Para atualizar essa política para uma VPC, use `aws:VpcSourceIp` em vez 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"
            }
        }
```

Verificando o suporte do seu cliente IPv6

Se você usar o endpoint secretsmanager.{region}.amazonaws.com, verifique se é possível se conectar a ele. As etapas a seguir descrevem como realizar a verificação.

Este exemplo usa Linux e curl versão 8.6.0 e usa o AWS Secrets Manager serviço que IPv6 habilitou endpoints localizados no endpoint amazonaws.com.

> **Note**
>
> O secretsmanager.{region}.amazonaws.com difere da convenção de nomenclatura típica de pilha dupla. Para obter uma lista completa de endpoints do Secrets Manager, consulte AWS Secrets Manager endpoints.
> Mude Região da AWS para a mesma região em que seu serviço está localizado. Neste exemplo, usamos o endpoint `us-east-1`, ou seja, Leste dos EUA (Norte da Virgínia).

1. Determine se o endpoint é resolvido com um IPv6 endereço usando o comando a seguir`dig`.

```
$ dig +short AAAA secretsmanager.us-east-1.amazonaws.com

> 2600:1f18:e2f:4e05:1a8a:948e:7c08:c1c3
```

2. Determine se a rede do cliente pode fazer uma IPv6 conexão usando o `curl` comando a seguir. Um código de resposta 404 significa uma conexão bem-sucedida, enquanto um código de resposta 0 significa falha da conexão.

```
$ 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
```

Se um IP remoto foi identificado e o código de resposta não`0`, uma conexão de rede foi estabelecida com sucesso com o endpoint usando IPv6. O IP remoto deve ser um IPv6 endereço porque o sistema operacional deve selecionar o protocolo válido para o cliente.

Se o IP remoto estiver em branco ou o código de resposta estiver`0`, a rede do cliente ou o caminho da rede até o endpoint será somente IPv4 -. É possível verificar isso com o seguinte comando do `curl`:

```
$ 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
```

Resiliência em AWS Secrets Manager

AWS constrói a infraestrutura global em torno de zonas Regiões da AWS de disponibilidade. Regiões da AWS fornecem várias zonas de disponibilidade fisicamente separadas e isoladas, que se conectam a redes de baixa latência, alta taxa de transferência e altamente redundantes. Com as zonas de disponibilidade, é possível projetar e operar aplicações e bancos de dados que automaticamente executam o failover entre as zonas sem interrupção. As zonas de disponibilidade permitem que você seja mais altamente disponível, tolerante a falhas e escalável do que uma ou várias infraestruturas de data center tradicionais.

Para obter mais informações sobre resiliência e recuperação de desastres, consulte Reliability Pillar — Well-Architected AWS Framework.

Para obter mais informações sobre zonas de disponibilidade Regiões da AWS e zonas de disponibilidade, consulte Infraestrutura AWS global.

TLS pós-quântico

O Secrets Manager é compatível com uma opção de troca pós-quântica híbrida de chaves para o protocolo Transport Layer Security (TLS) de criptografia de rede. É possível usar essa opção de TLS ao se conectar a endpoints de API do Secrets Manager. Estamos oferecendo esse recurso antes da

padronização dos algoritmos pós-quânticos, permitindo que você comece a testar o efeito desses protocolos de troca de chaves em chamadas do Secrets Manager. Esses recursos opcionais de troca de chaves pós-quânticas híbridas são pelo menos tão seguros quanto a criptografia TLS que usamos atualmente e provavelmente fornecerão benefícios adicionais de segurança. No entanto, eles afetam a latência e o throughput em comparação com os protocolos clássicos de troca de chaves em uso atualmente. O Agente do Secrets Manager usa a troca de chaves pós-quântica ML-KEM como a troca de chaves de maior prioridade por padrão.

Para proteger os dados criptografados hoje contra possíveis ataques futuros, AWS está participando com a comunidade criptográfica no desenvolvimento de algoritmos quânticos resistentes ou pós-quânticos. Implementamos pacotes de criptografia de troca pós-quântica híbrida de chaves em endpoints do Secrets Manager. Esses conjuntos de cifras híbridas, que combinam elementos clássicos e pós-quânticos, garantem que sua conexão TLS seja pelo menos tão forte quanto seria com pacotes de criptografia clássica. No entanto, como as características de desempenho e os requisitos de largura de banda dos pacotes de criptografia híbrida são diferentes dos mecanismos clássicos de troca de chaves, recomendamos testá-los em suas chamadas de API.

O Secrets Manager é compatível com o PQTLS em todas as regiões, exceto nas da China.

Configurar o TLS pós-quântico híbrido

1. Adicione o cliente AWS Common Runtime às suas dependências do Maven. Recomendamos usar a versão mais recente disponível. Por exemplo, esta instrução adiciona a versão 2.20.0.

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>aws-crt-client</artifactId>
  <version>2.20.0</version>
</dependency>
```

2. Adicione o AWS SDK for Java 2.x ao seu projeto e inicialize-o. Habilite os pacotes de cifra pós-quântica híbrida em seu cliente HTTP.

```
SdkAsyncHttpClient awsCrtHttpClient = AwsCrtAsyncHttpClient.builder()
            .postQuantumTlsEnabled(true)
            .build();
```

3. Crie o cliente assíncrono do Secrets Manager.

```
SecretsManagerAsyncClient SecretsManagerAsync = SecretsManagerAsyncClient.builder()
```

```
            .httpClient(awsCrtHttpClient)
            .build();
```

Agora quando você chama operações de API do Secrets Manager, suas chamadas são transmitidas ao endpoint do Secrets Manager ao usar o TLS pós-quântico híbrido.

Para obter mais informações sobre o uso de TLS pós-quântico híbrido, consulte:

- AWS SDK for Java 2.x Guia do desenvolvedor e a publicação AWS SDK for Java 2.x lançada no blog.
- Apresentação do s2n-tls, uma nova implementação de código aberto do TLS e Uso do s2n-tls.
- Post-Quantum Cryptography (Criptografia pós-quântica) no National Institute for Standards and Technology (NIST).
- Hybrid Post-Quantum Key Encapsulation Methods (PQ KEM – Métodos pós-quânticos híbridos de encapsulamento de chave) para Transport Layer Security 1.2 (TLS).

O TLS pós-quântico para Secrets Manager está disponível em todos, exceto na Regiões da AWS China.

Solução de problemas AWS Secrets Manager

Use estas informações para ajudá-lo a diagnosticar e corrigir problemas que você venha a encontrar ao trabalhar com o Secrets Manager.

Para problemas relacionados à alternância, consulte the section called “Solucionar problemas de alternância”.

Tópicos

Mensagens de "Acesso negado"

Ao fazer uma chamada de API, como GetSecretValue ou CreateSecret para o Secrets Manager, você precisa ter permissões do IAM para fazer essa chamada. Quando você usa o console, ele faz as mesmas chamadas de API em seu nome, então você também precisa ter permissões do IAM. Um administrador pode conceder permissões anexando uma política do IAM ao seu usuário do IAM ou a um grupo do qual você faz parte. Se as declarações de política que concedem essas permissões incluírem quaisquer condições, como time-of-day restrições de endereço IP, você também deverá atender a esses requisitos ao enviar a solicitação. Para obter informações sobre como visualizar ou modificar políticas para um usuário, grupo ou perfil do IAM, consulte Trabalho com políticas no Manual do usuário do IAM. Para obter mais informações sobre as permissões necessárias para o Secrets Manager, consulte the section called “Autenticação e controle de acesso”.

Se você estiver assinando solicitações de API manualmente, sem usar o AWS SDKs, verifique se assinou corretamente a solicitação.

"Access denied" (Acesso negado) para credenciais de segurança temporárias

Verifique se o usuário ou o perfil do IAM que você está usando para fazer a solicitação tem as permissões corretas. As permissões de credenciais de segurança temporárias são originárias de um usuário ou perfil do IAM. Isso significa que as permissões são limitadas às concedidas ao usuário ou perfil do IAM. Para obter mais informações sobre como as permissões de credenciais de segurança temporárias são determinadas, consulte Controlar permissões para credenciais de segurança temporárias no Manual do usuário do IAM.

Verifique se suas solicitações são assinadas corretamente e bem-formada. Para obter detalhes, consulte a documentação do kit de ferramentas do SDK escolhido ou Como usar credenciais de segurança temporárias para solicitar acesso aos AWS recursos no Guia do usuário do IAM.

Verifique se suas credenciais de segurança temporárias não expiraram. Para obter mais informações, consulte Solicitação de credenciais de segurança temporárias no Manual do usuário do IAM.

Para obter mais informações sobre as permissões necessárias para o Secrets Manager, consulte the section called “Autenticação e controle de acesso”.

As alterações que faço nem sempre ficam imediatamente visíveis.

O Secret Manager usa um modelo de computação distribuído chamado consistência eventual. Qualquer alteração que você fizer no Secrets Manager (ou em outros AWS serviços) leva tempo para se tornar visível em todos os endpoints possíveis. Parte do atraso resulta do tempo necessário para enviar os dados de um servidor para outro, de uma zona de replicação para outra e de uma região para outra em todo o mundo. O Secrets Manager também usa armazenamento em cache para melhorar a performance. Porém, em alguns casos, isso pode aumentar o tempo. A alteração talvez não fique visível enquanto os dados armazenados em cache anteriormente não atingirem o tempo limite.

Projete seus aplicações globais levando em conta esses possíveis atrasos. Além disso, garanta o funcionamento esperado, mesmo quando uma alteração feita em um local não fique imediatamente visível em outro.

Para obter mais informações sobre como alguns outros AWS serviços são afetados pela consistência eventual, consulte:

- Gerenciamento da consistência de dados no Guia do desenvolvedor do banco de dados do Amazon Redshift
- Modelo de consistência de dados do Amazon S3 no Manual do usuário do Amazon Simple Storage Service
- Garantir consistência ao usar o Amazon S3 e o Amazon EMR para fluxos de trabalho de ETL no blog sobre big data da AWS
- Consistência final do Amazon EC2 na Referência de API do Amazon EC2

“Cannot generate a data key with an asymmetric KMS key” (Não é possível gerar uma chave de dados com uma chave do KMS assimétrica) ao criar um segredo

O Secrets Manager usa uma chave de criptografia simétrica do KMS associada a um segredo para gerar uma chave de dados para cada valor de segredo. Não é possível usar uma chave do KMS assimétrica. Verifique se você está usando uma chave de criptografia do KMS simétrica em vez de uma chave do KMS assimétrica. Para obter instruções, consulte Identificação de chaves do KMS assimétricas.

Uma operação AWS do SDK AWS CLI ou não consegue encontrar meu segredo em um ARN parcial

Em muitos casos, o Secrets Manager pode encontrar seu segredo com parte de um ARN em vez do ARN completo. No entanto, se o nome do seu segredo terminar com um hífen seguido de seis caracteres, talvez o Secrets Manager não consiga encontrar o segredo exclusivamente com base em uma parte de um ARN. Em vez disso, recomendamos que você use o ARN completo ou o nome do segredo.

Mais detalhes

O Secrets Manager inclui seis caracteres aleatórios ao final do nome do segredo para ajudar a garantir que o ARN do segredo seja único. Se o segredo original for excluído e um novo segredo for criado com o mesmo nome, os dois segredos serão diferentes ARNs por causa desses caracteres. Os usuários com acesso ao segredo antigo não obtêm acesso automático ao novo segredo porque ARNs são diferentes.

O Secrets Manager cria um ARN para um segredo com a região, conta, nome do segredo e, em seguida, um hífen e mais seis caracteres, da seguinte forma:

```
arn:aws:secretsmanager:us-east-2:111122223333:secret:SecretName-abcdef
```

Se o nome do segredo terminar com um hífen e seis caracteres, usar apenas parte do ARN pode dar a impressão para o Secrets Manager de que você está especificando um ARN completo. Por exemplo, é possível ter um segredo nomeado `MySecret-abcdef` com o ARN

`arn:aws:secretsmanager:us-east-2:111122223333:secret:MySecret-abcdef-nutBrk`

Se você chamar a seguinte operação, que usa apenas parte do ARN do segredo, talvez o Secrets Manager não encontre o segredo.

```
$ aws secretsmanager describe-secret --secret-id arn:aws:secretsmanager:us-
east-2:111122223333:secret:MySecret-abcdef
```

Esse segredo é gerenciado por um AWS serviço e você deve usar esse serviço para atualizá-lo.

Se você encontrar essa mensagem ao tentar modificar um segredo, o segredo só poderá ser atualizado usando o serviço de gerenciamento listado na mensagem. Para obter mais informações, consulte Segredos gerenciados por outros serviços.

Para determinar quem gerencia um segredo, é possível revisar o nome do segredo. Os segredos gerenciados por outros serviços são prefixados com o ID do respectivo serviço. Ou, no AWS CLI, chame describe-secret e revise o campo. `OwningService`

A importação do módulo Python falha ao usar **`Transform: AWS::SecretsManager-2024-09-16`**

Se você estiver usando Transform: `AWS::SecretsManager-2024-09-16` e encontrar falhas na importação do módulo Python quando sua função do Lambda de alternância é executada, o problema provavelmente é causado por um valor incompatível do `Runtime`. Com essa versão de transformação, o AWS CloudFormation gerencia a versão, o código e os arquivos de objetos compartilhados do runtime para você. Não é necessário implementar isso você mesmo.

AWS Secrets Manager cotas

As leituras do Secrets Manager APIs têm cotas de TPS altas e os planos de controle APIs que são chamados com menos frequência têm cotas de TPS mais baixas. Recomendamos que você evite chamar `PutSecretValue` ou `UpdateSecret` em uma frequência sustentada de mais de uma vez a cada dez minutos. Quando você chama `PutSecretValue` ou `UpdateSecret` para atualizar o valor do segredo, o Secrets Manager cria uma nova versão do segredo. O Secrets Manager remove versões sem rótulo quando há mais de 100, mas não remove versões criadas há menos de 24 horas. Se você atualizar o valor do segredo mais de uma vez a cada dez minutos, criará mais versões do que o Secrets Manager remove e atingirá a cota de versões de segredos.

É possível operar várias regiões na sua conta, e cada cota é específica para cada região.

Quando um aplicativo em um Conta da AWS usa um segredo de propriedade de uma conta diferente, isso é conhecido como solicitação entre contas. Nas solicitações entre contas, o Secrets Manager limita a conta da identidade que faz as solicitações, não a conta que possui o segredo. Por exemplo, se um aplicação na conta A usar um segredo na conta B, o uso do segredo será aplicado somente às cotas da conta A.

Cotas do Secrets Manager

Nome	Padrão	Ajustá	Description
Taxa combinada de DeleteResourcePolicy, GetResourcePolicy PutResourcePolicy, e solicitações de ValidateResourcePolicy API	Cada região compatível: 50 por segundo	Não	O máximo de transações por segundo para DeleteResourcePolicy, GetResourcePolicy PutResourcePolicy, e solicitações de ValidateResourcePolicy API combinadas.
Taxa combinada de PutSecretValue, RemoveRegionsFromReplication, ReplicateSecretToRegion, StopReplicationToReplica UpdateSecret, e	Cada região compatível: 50 por segundo	Não	O máximo de transações por segundo para PutSecretValue, RemoveRegionsFromReplication, Replicate

Nome	Padrão	Ajustá	Description
solicitações de UpdateSecretVersionStage API			SecretToRegion, StopReplicationToReplica UpdateSecret, e solicitações de UpdateSecretVersionStage API combinadas.
Taxa combinada de solicitações de RestoreSecret API	Cada região compatível: 50 por segundo	Não	O máximo de transações por segundo para solicitações de RestoreSecret API.
Taxa combinada de solicitações RotateSecret e de CancelRotateSecret API	Cada região compatível: 50 por segundo	Não	O máximo de transações por segundo RotateSecret e solicitações de CancelRotateSecret API combinadas.
Taxa combinada de solicitações TagResource e de UntagResource API	Cada região compatível: 50 por segundo	Não	O máximo de transações por segundo TagResource e solicitações de UntagResource API combinadas.
Taxa de solicitações de BatchGetSecretValue API	Cada região compatível: 100 por segundo	Não	O máximo de transações por segundo para solicitações de BatchGetSecretValue API.
Taxa de solicitações de CreateSecret API	Cada região compatível: 50 por segundo	Não	O máximo de transações por segundo para solicitações de CreateSecret API.

Nome	Padrão	Ajustá	Description
Taxa de solicitações de DeleteSecret API	Cada região compatível: 50 por segundo	Não	O máximo de transações por segundo para solicitações de DeleteSecret API.
Taxa de solicitações de DescribeSecret API	Cada região suportada: 40.000 por segundo	Não	O máximo de transações por segundo para solicitações de DescribeSecret API.
Taxa de solicitações de GetRandomPassword API	Cada região compatível: 50 por segundo	Não	O máximo de transações por segundo para solicitações de GetRandomPassword API.
Taxa de solicitações de GetSecretValue API	Cada região compatível: 10.000 por segundo	Não	O máximo de transações por segundo para solicitações de GetSecretValue API.
Taxa de solicitações de ListSecretVersionIds API	Cada região compatível: 50 por segundo	Não	O máximo de transações por segundo para solicitações de ListSecretVersionIds API.
Taxa de solicitações de ListSecrets API	Cada região compatível: 100 por segundo	Não	O máximo de transações por segundo para solicitações de ListSecrets API.
Comprimento de política baseada em recurso	Cada região compatível: 20.480	Não	O número máximo de caracteres em uma política de permissões baseada em recurso anexada a um segredo.

Nome	Padrão	Ajustá	Description
Tamanho de valor do segredo	Cada região compatível: 65.536 Bytes	Não	O tamanho máximo de um valor do segredo criptografado. Se o valor do segredo for uma string, esse tamanho será referente ao número de caracteres permitidos no valor do segredo.
Segredos	Cada região compatível: 500.000	Não	O número máximo de segredos em cada AWS região dessa AWS conta.
Preparação prévia de rótulos anexados em todas as versões de um segredo	Cada região compatível: 20	Não	Número máximo de rótulos de preparação anexados entre todas as versões de um segredo.
Versões por segredo	Cada região compatível: 100	Não	O número máximo de versões de um segredo.

Adicionar novas tentativas à aplicação

Seu AWS cliente pode ver as chamadas para o Secrets Manager falharem devido a problemas inesperados no lado do cliente. Ou as chamadas podem falhar devido à limitação da taxa pelo Secrets Manager. Quando você excede uma cota de solicitação de API, o Secrets Manager controla a utilização da solicitação. Ele rejeita uma solicitação que do contrário seria válida e retorna um erro de throttling. Para ambos os tipos de falhas, recomendamos que você tente a chamada novamente após um breve período de espera. Isso é chamado de estratégia de recuo e repetição.

Se ocorrerem os seguintes erros, adicione novas tentativas ao código da aplicação:

Erros e exceções transitórias

- `RequestTimeout`
- `RequestTimeoutException`

- `PriorRequestNotComplete`
- `ConnectionError`
- `HTTPClientError`

Erros e exceções de controle de utilização e limitação no lado do serviço

- `Throttling`
- `ThrottlingException`
- `ThrottledException`
- `RequestThrottledException`
- `TooManyRequestsException`
- `ProvisionedThroughputExceededException`
- `TransactionInProgressException`
- `RequestLimitExceeded`
- `BandwidthLimitExceeded`
- `LimitExceededException`
- `RequestThrottled`
- `SlowDown`

Para obter mais informações, além do código de exemplo, sobre novas tentativas, recuo exponencial e jitter, consulte os seguintes recursos:

- Recuo exponencial e jitter
- Tempos limite esgotado, novas tentativas e recuo com jitter
- Tentativas de erro e recuo exponencial. AWS

Histórico do documento

A tabela a seguir descreve as mudanças importantes na documentação desde a última versão do AWS Secrets Manager. Para receber notificações sobre atualizações dessa documentação, é possível inscrever-se em um feed RSS.

Alteração	Descrição	Data
Nova política AWS gerenciada	O Secrets Manager lançou uma nova política gerenciada `AWSSecretsManagerClientReadOnlyAccess` que fornece acesso somente de leitura aos segredos para aplicativos clientes. Para obter informações, consulte Atualizações do Secrets Manager para políticas AWS gerenciadas.	5 de novembro de 2025
Adição de suporte a tags de alocação de custos	O Secrets Manager agora oferece suporte a tags de alocação de custos, permitindo que os clientes categorizem e acompanhem os custos por departamento, equipe ou aplicação. Para obter mais informações, consulte Usando tags de alocação de custos com AWS Secrets Manager.	27 de maio de 2025
Suporte adicional IPv6 e de pilha dupla	O Secrets Manager agora oferece suporte a endpoints de pilha dupla. Consulte IPv4 e IPv6 acesse para obter mais informações.	20 de dezembro de 2024

Alteração do Secrets Manager para política AWS gerenciada	A política gerenciada pela `SecretsManagerReadWrite` agora inclui a permissão `redshift-serverless` . Para obter mais informações, consulte a política AWS gerenciada para AWS Secrets Manager	12 de março de 2024

Atualizações anteriores

A tabela a seguir descreve mudanças importantes em cada versão do Guia do AWS Secrets Manager usuário antes de fevereiro de 2024.

Alteração	Descrição	Data
Disponibilidade geral	Este é o lançamento público inicial do Secrets Manager.	4 de abril de 2018

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.