Data Encryption Workshop

Melhores práticas

Edição	01
Data	2024-09-14

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Marcas registadas e permissões

e outras marcas registadas da Huawei são marcas registadas da Huawei Technologies Co., Ltd. Todos as outras marcas registadas e os nomes registados mencionados neste documento são propriedade dos seus respectivos detentores.

Aviso

Os produtos, os serviços e as funcionalidades adquiridos são estipulados pelo contrato estabelecido entre a Huawei Cloud e o cliente. Os produtos, os serviços e as funcionalidades descritos neste documento, no todo ou em parte, podem não estar dentro do âmbito de aquisição ou do âmbito de uso. Salvo especificação em contrário no contrato, todas as declarações, informações e recomendações neste documento são fornecidas "TAL COMO ESTÃO" sem garantias ou representações de qualquer tipo, sejam expressas ou implícitas.

As informações contidas neste documento estão sujeitas a alterações sem aviso prévio. Foram feitos todos os esforços na preparação deste documento para assegurar a exatidão do conteúdo, mas todas as declarações, informações e recomendações contidas neste documento não constituem uma garantia de qualquer tipo, expressa ou implícita.

Huawei Cloud Computing Technologies Co., Ltd.

Endereço: Huawei Cloud Data Center, Rua Jiaoxinggong
Avenida Qianzhong
Novo Distrito de Gui'an
Guizhou 550029
República Popular da China

Site: https://www.huaweicloud.com/intl/pt-br/

Índice

1 Serviço de gerenciamento de chaves

1.1 Uso do KMS para criptografar dados off-line

1.1.1 Criptografia ou descriptografia de pequenos volumes de dados

Cenário

Você pode usar ferramentas on-line no console do Key Management Service (KMS) ou chamar as APIs do KMS necessárias para criptografar ou descriptografar diretamente dados de tamanho pequeno com uma CMK, como senhas, certificados ou números de telefone.

Restrições

Atualmente, um máximo de 4 KB de dados podem ser criptografados ou descriptografados dessa maneira.

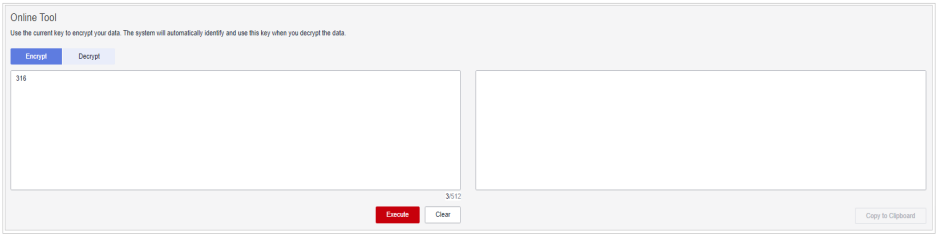
Criptografia e descriptografia usando ferramentas on-line

- **Criptografar dados**

Passo 1 Clique no alias de uma chave personalizada para exibir seus detalhes e vá para a ferramenta on-line para criptografia e descriptografia de dados.

Passo 2 Clique em **Encrypt**. Na caixa de texto à esquerda, insira os dados a serem criptografados. Para obter detalhes, consulte **Figura 1-1**.

Figura 1-1 Criptografia de dados



Passo 3 Clique em **Execute**. Texto cifrado dos dados é exibido na caixa de texto à direita.

NOTA

- Use a CMK atual para criptografar os dados.
- Você pode clicar em **Clear** para limpar os dados inseridos.
- Você pode clicar em **Copy to Clipboard** para copiar o texto cifrado e salvá-lo em um arquivo local.

- **Descriptografar dados**

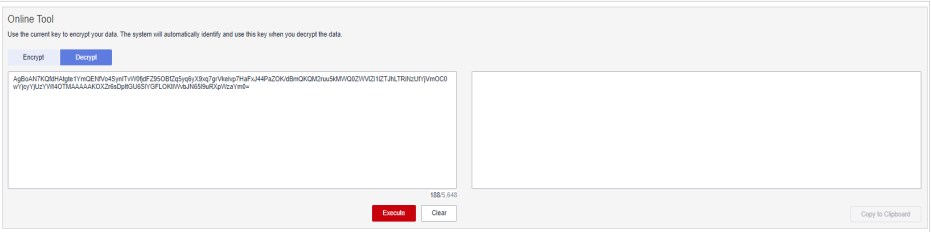
Passo 4 Você pode clicar em qualquer chave não padrão no status **Enabled** para acessar a página de criptografia e descriptografia da ferramenta on-line.

Passo 5 Clique em **Decrypt**. Na caixa de texto à esquerda, insira os dados a serem descriptografados. Para obter detalhes, consulte **Figura 1-2**.

NOTA

- A ferramenta identificará a CMK de criptografia original e a usará para descriptografar os dados.
- No entanto, se a CMK tiver sido excluída, a descriptografia falhará.

Figura 1-2 Descriptografia de dados



Passo 6 Clique em **Execute**. O texto não criptografado dos dados é exibido na caixa de texto à direita.

NOTA

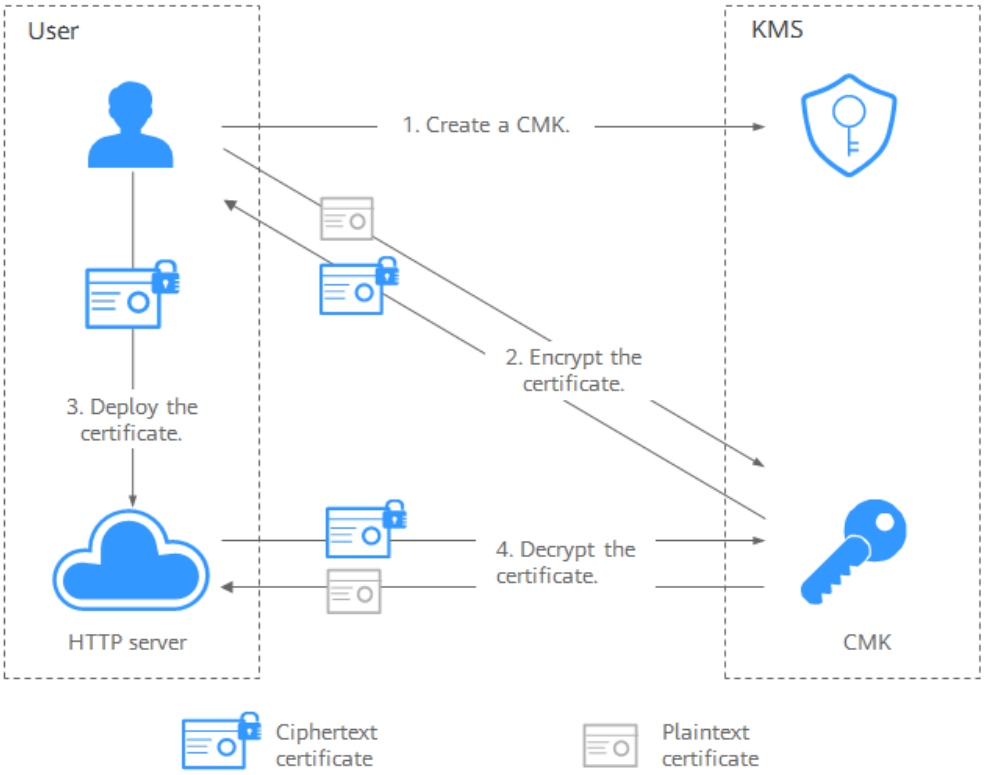
- Você pode clicar em **Copy to Clipboard** para copiar o texto não criptografado e salvá-lo em um arquivo local.

----Fim

Chamada de APIs para criptografia e descriptografia

A **Figura 1-3** mostra um exemplo sobre como chamar APIs do KMS para criptografar e descriptografar um certificado HTTPS.

Figura 1-3 Criptografar e descriptografar um certificado HTTPS



O procedimento é o seguinte:

1. Crie uma CMK no KMS.
2. Chame a interface **encrypt-data** do KMS e use a CMK para criptografar o certificado de texto não criptografado.
3. Implemente o certificado em um servidor.
4. O servidor usa a interface **decrypt-data** do KMS para descriptografar o certificado de texto cifrado.

> **NOTA**
>
> Se você inserir e criptografar texto no console, o texto será codificado no formato Base64 antes de ser transferido para o back-end para criptografia. O resultado da descriptografia retornado via API estará no formato Base64. Texto criptografado via API não pode ser descriptografado no console, ou caracteres ilegíveis serão retornados.

1.1.2 Criptografia ou descriptografia de uma grande quantidade de dados

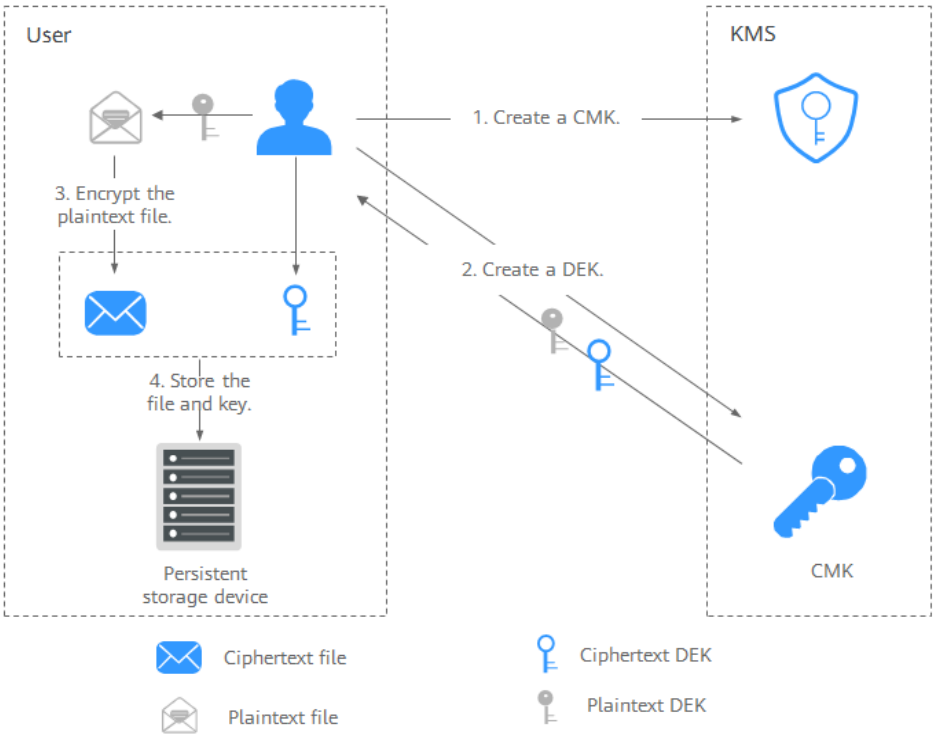
Cenário

Se você quiser criptografar ou descriptografar grandes volumes de dados, como imagens, vídeos e arquivos de banco de dados, poderá usar a criptografia de envelope, que permite criptografar e descriptografar arquivos sem precisar transferir uma grande quantidade de dados pela rede.

Processos de criptografia e descriptografia

- Criptografia de dados de grande porte

  Figura 1-4 Criptografar um arquivo local

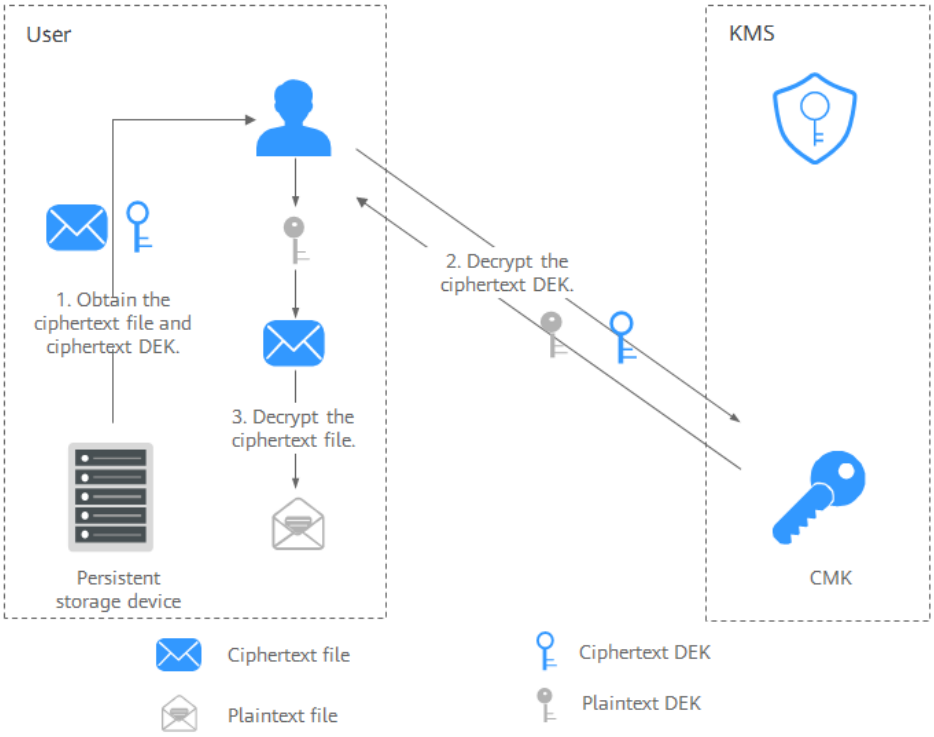


  O processo é o seguinte:

  a. Crie uma CMK no KMS.
  b. Chame a API **create-datakey** do KMS para criar uma DEK. Uma DEK de texto não criptografado e uma DEK de texto cifrado serão geradas. A DEK de texto cifrado é gerada quando você usa uma CMK para criptografar a DEK de texto não criptografado.
  c. Use a DEK de texto não criptografado para criptografar um arquivo de texto não criptografado, gerando um arquivo de texto cifrado.
  d. Armazene a DEK de texto cifrado e o arquivo de texto cifrado juntos em um dispositivo de armazenamento permanente ou um serviço de armazenamento.

- Descriptografia de dados de grande porte

Figura 1-5 Descriptografar um arquivo local



O processo é o seguinte:

a. Leia a DEK de texto cifrado e o arquivo de texto cifrado do dispositivo de armazenamento permanente ou do serviço de armazenamento.

b. Chame a API **decrypt-datakey** do KMS e use a CMK correspondente (aquela usada para criptografar a DEK) para descriptografar a DEK de texto cifrado. Então você obtém a DEK de texto não criptografado.

   Se a CMK for excluída, a descriptografia falhará. Mantenha corretamente suas CMKs.

c. Use a DEK de texto não criptografado para descriptografar o arquivo de texto cifrado.

APIs relacionadas à criptografia de envelope

Você pode usar as seguintes APIs para criptografar e descriptografar dados.

API	Descrição
Criação de uma DEK	Criar uma DEK.
Criptografia de uma DEK	Criptografar uma DEK com a chave principal especificada.
Descriptografia de uma DEK	Descriptografar uma DEK com a chave principal especificada.

Criptografia de um arquivo local

1. Crie uma CMK no console de gerenciamento. Para obter detalhes, consulte **Criação de uma CMK**.
2. Prepare informações básicas de autenticação.
   - **ACCESS_KEY**: chave de acesso do ID da Huawei
   - **SECRET_ACCESS_KEY**: chave de acesso secreta da Huawei ID
   - **PROJECT_ID**: ID do projeto de um site da HUAWEI CLOUD. Para obter detalhes, consulte **Projeto**.
   - **KMS_ENDPOINT**: ponto de extremidade para acessar o KMS. Para obter detalhes, consulte **Pontos de extremidade**.
   - Haverá riscos de segurança se a AK/SK usada para autenticação for gravada diretamente no código. Criptografe a AK/SK no arquivo de configuração ou nas variáveis de ambiente para armazenamento.
   - Neste exemplo, a AK/SK armazenada nas variáveis de ambiente é usada para autenticação de identidade. Configure as variáveis de ambiente **HUAWEICLOUD_SDK_AK** e **HUAWEICLOUD_SDK_SK** primeiro no ambiente local.
3. Criptografe um arquivo local.

   O código de exemplo é o seguinte.
   - CMK é o ID da chave criada no console de gerenciamento da HUAWEI CLOUD.
   - O arquivo de dados de texto não criptografado é **FirstPlainFile.jpg**.
   - O arquivo de dados gerado após a criptografia é **SecondEncryptFile.jpg**.

```
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.kms.v1.KmsClient;
import com.huaweicloud.sdk.kms.v1.model.CreateDatakeyRequest;
import com.huaweicloud.sdk.kms.v1.model.CreateDatakeyRequestBody;
import com.huaweicloud.sdk.kms.v1.model.CreateDatakeyResponse;
import com.huaweicloud.sdk.kms.v1.model.DecryptDatakeyRequest;
import com.huaweicloud.sdk.kms.v1.model.DecryptDatakeyRequestBody;

import javax.crypto.Cipher;
import javax.crypto.spec.GCMParameterSpec;
import javax.crypto.spec.SecretKeySpec;
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.nio.file.Files;
import java.security.SecureRandom;

/**
* Use a DEK to encrypt and decrypt files.
* To enable the assert syntax, add -ea to enable VM_OPTIONS.
 */
public class FileStreamEncryptionExample {

    private static final String ACCESS_KEY =
System.getenv("HUAWEICLOUD_SDK_AK");
    private static final String SECRET_ACCESS_KEY =
System.getenv("HUAWEICLOUD_SDK_SK");
    private static final String PROJECT_ID = "<ProjectID>";
    private static final String KMS_ENDPOINT = "<KmsEndpoint>";

//Version of the KMS interface. Currently, the value is fixed to v1.0.
    private static final String KMS_INTERFACE_VERSION = "v1.0";
```



```
    /**
     * AES algorithm flags:
     * - AES_KEY_BIT_LENGTH: bit length of the AES256 key
      * - AES_KEY_BYTE_LENGTH: byte length of the AES256 key
     * - AES_ALG: AES256 algorithm. In this example, the Group mode is
GCM and the padding mode is PKCS5Padding.
     * - AES_FLAG: AES algorithm flag
     * - GCM_TAG_LENGTH: GCM tag length
     * - GCM_IV_LENGTH: length of the GCM initial vector
     */
    private static final String AES_KEY_BIT_LENGTH = "256";
    private static final String AES_KEY_BYTE_LENGTH = "32";
    private static final String AES_ALG = "AES/GCM/PKCS5Padding";
    private static final String AES_FLAG = "AES";
    private static final int GCM_TAG_LENGTH = 16;
    private static final int GCM_IV_LENGTH = 12;

    public static void main(final String[] args) {
        // ID of the CMK you created on the HUAWEI CLOUD management
console
        final String keyId = args[0];

        encryptFile(keyId);
    }

    /**
     * Using a DEK to encrypt and decrypt a file
     *
     * @param keyId: user CMK ID
     */
    static void encryptFile(String keyId) {
        // 1. Prepare the authentication information for accessing
HUAWEI CLOUD.
        final BasicCredentials auth = new
BasicCredentials().withAk(ACCESS_KEY).withSk(SECRET_ACCESS_KEY)
                .withProjectId(PROJECT_ID);

        // 2. Initialize the SDK and transfer the authentication
information and the address for the KMS to access the client.
        final KmsClient kmsClient =
KmsClient.newBuilder().withCredential(auth).withEndpoint(KMS_ENDPOINT).bu
ild();

        // 3. Assemble the request message for creating a DEK.
        final CreateDatakeyRequest createDatakeyRequest = new
CreateDatakeyRequest().withVersionId(KMS_INTERFACE_VERSION)
                .withBody(new
CreateDatakeyRequestBody().withKeyId(keyId).withDatakeyLength(AES_KEY_BIT
_LENGTH));

        // 4. Create a DEK.
        final CreateDatakeyResponse createDatakeyResponse =
kmsClient.createDatakey(createDatakeyRequest);

        // 5. Receive the created DEK information.
        // It is recommended that the ciphertext key and key ID be
stored locally so that the plaintext key can be easily obtained for data
decryption.
        // The plaintext key should be used immediately after being
created. Before using it, convert the hexadecimal plaintext key to a
byte array.
        final String cipherText = createDatakeyResponse.getCipherText();
        final byte[] plainKey =
hexToBytes(createDatakeyResponse.getPlainText());

        // 6. Prepare the file to be encrypted.
        // inFile: file to be encrypted
        // outEncryptFile: file generated after encryption
```



```
        final File inFile = new File("FirstPlainFile.jpg");
        final File outEncryptFile = new File("SecondEncryptFile.jpg");

        // 7. If the AES algorithm is used for encryption, you can
create an initial vector.
        final byte[] iv = new byte[GCM_IV_LENGTH];
        final SecureRandom secureRandom = new SecureRandom();
        secureRandom.nextBytes(iv);

        // 8. Encrypt the file and store the encrypted file.
        doFileFinal(Cipher.ENCRYPT_MODE, inFile, outEncryptFile,
plainKey, iv);

    }

    /**
     * Encrypting and decrypting a file
     *
     * @param cipherMode: Encryption mode. It can be Cipher.ENCRYPT_MODE
or Cipher.DECRYPT_MODE.
     * @param infile: file to be encrypted or decrypted
     * @param outFile: file generated after encryption and decryption
     * @param keyPlain: plaintext key
     * @param iv: initial vector
     */
    static void doFileFinal(int cipherMode, File infile, File outFile,
byte[] keyPlain, byte[] iv) {

        try (BufferedInputStream bis = new BufferedInputStream(new
FileInputStream(infile));
             BufferedOutputStream bos = new BufferedOutputStream(new
FileOutputStream(outFile))) {
            final byte[] bytIn = new byte[(int) infile.length()];
            final int fileLength = bis.read(bytIn);

            assert fileLength > 0;

            final SecretKeySpec secretKeySpec = new
SecretKeySpec(keyPlain, AES_FLAG);
            final Cipher cipher = Cipher.getInstance(AES_ALG);
            final GCMParameterSpec gcmParameterSpec = new
GCMParameterSpec(GCM_TAG_LENGTH * Byte.SIZE, iv);
            cipher.init(cipherMode, secretKeySpec, gcmParameterSpec);
            final byte[] bytOut = cipher.doFinal(bytIn);
            bos.write(bytOut);
        } catch (Exception e) {
            throw new RuntimeException(e.getMessage());
        }
    }

}
```

Descriptografia de um arquivo local

1. Prepare informações básicas de autenticação.
   - **ACCESS_KEY**: chave de acesso do ID da Huawei
   - **SECRET_ACCESS_KEY**: chave de acesso secreta da Huawei ID
   - **PROJECT_ID**: ID do projeto de um site da HUAWEI CLOUD. Para obter detalhes, consulte **Projeto**.
   - **KMS_ENDPOINT**: ponto de extremidade para acessar o KMS. Para obter detalhes, consulte **Pontos de extremidade**.
   - Haverá riscos de segurança se a AK/SK usada para autenticação for gravada diretamente no código. Criptografe a AK/SK no arquivo de configuração ou nas variáveis de ambiente para armazenamento.

- Neste exemplo, a AK/SK armazenada nas variáveis de ambiente é usada para autenticação de identidade. Configure as variáveis de ambiente **HUAWEICLOUD_SDK_AK** e **HUAWEICLOUD_SDK_SK** primeiro no ambiente local.

2. Descriptografe um arquivo local.

   O código de exemplo é o seguinte.

   - CMK é o ID da chave criada no console de gerenciamento da HUAWEI CLOUD.
   - O arquivo de dados gerado após a criptografia é **SecondEncryptFile.jpg**.
   - O arquivo de dados gerado após a criptografia e descriptografia é **ThirdDecryptFile.jpg**.

```
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.kms.v1.KmsClient;
import com.huaweicloud.sdk.kms.v1.model.CreateDatakeyRequest;
import com.huaweicloud.sdk.kms.v1.model.CreateDatakeyRequestBody;
import com.huaweicloud.sdk.kms.v1.model.CreateDatakeyResponse;
import com.huaweicloud.sdk.kms.v1.model.DecryptDatakeyRequest;
import com.huaweicloud.sdk.kms.v1.model.DecryptDatakeyRequestBody;

import javax.crypto.Cipher;
import javax.crypto.spec.GCMParameterSpec;
import javax.crypto.spec.SecretKeySpec;
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.nio.file.Files;
import java.security.SecureRandom;

/**
* Use a DEK to encrypt and decrypt files.
* To enable the assert syntax, add -ea to enable VM_OPTIONS.
 */
public class FileStreamEncryptionExample {

    private static final String ACCESS_KEY =
System.getenv("HUAWEICLOUD_SDK_AK");
    private static final String SECRET_ACCESS_KEY =
System.getenv("HUAWEICLOUD_SDK_SK");
    private static final String PROJECT_ID = "<ProjectID>";
    private static final String KMS_ENDPOINT = "<KmsEndpoint>";


//Version of the KMS interface. Currently, the value is fixed to v1.0.
    private static final String KMS_INTERFACE_VERSION = "v1.0";

    /**
     * AES algorithm flags:
     * - AES_KEY_BIT_LENGTH: bit length of the AES256 key
      * - AES_KEY_BYTE_LENGTH: byte length of the AES256 key
     * - AES_ALG: AES256 algorithm. In this example, the Group mode is
GCM and the padding mode is PKCS5Padding.
     * - AES_FLAG: AES algorithm flag
     * - GCM_TAG_LENGTH: GCM tag length
     * - GCM_IV_LENGTH: length of the GCM initial vector
     */
    private static final String AES_KEY_BIT_LENGTH = "256";
    private static final String AES_KEY_BYTE_LENGTH = "32";
    private static final String AES_ALG = "AES/GCM/PKCS5Padding";
    private static final String AES_FLAG = "AES";
    private static final int GCM_TAG_LENGTH = 16;
    private static final int GCM_IV_LENGTH = 12;

    public static void main(final String[] args) {
```



```
        // ID of the CMK you created on the HUAWEI CLOUD management
console
        final String keyId = args[0];
        // // Returned ciphertext DEK after DEK creation
        final String cipherText = args[1];

        decryptFile(keyId, cipherText);
    }

    /**
     * Using a DEK to encrypt and decrypt a file
     *
     * @param keyId: user CMK ID
     * @param cipherText: ciphertext data key
     */
    static void decryptFile(String keyId, String cipherText) {
        // 1. Prepare the authentication information for accessing
HUAWEI CLOUD.
        final BasicCredentials auth = new
BasicCredentials().withAk(ACCESS_KEY).withSk(SECRET_ACCESS_KEY)
                .withProjectId(PROJECT_ID);

        // 2. Initialize the SDK and transfer the authentication
information and the address for the KMS to access the client.
        final KmsClient kmsClient =
KmsClient.newBuilder().withCredential(auth).withEndpoint(KMS_ENDPOINT).bu
ild();

        // 3. Prepare the file to be encrypted.
        // inFile: file to be encrypted
        // outEncryptFile: file generated after encryption
        // outDecryptFile: file generated after encryption and decryption
        final File inFile = new File("FirstPlainFile.jpg");
        final File outEncryptFile = new File("SecondEncryptFile.jpg");
        final File outDecryptFile = new File("ThirdDecryptFile.jpg");

        // 4. Use the same initial vector for AES encryption and
decryption.
        final byte[] iv = new byte[GCM_IV_LENGTH];

        // 5. Assemble the request message for decrypting the DEK.
cipherText is the ciphertext DEK returned after DEK creation.
        final DecryptDatakeyRequest decryptDatakeyRequest = new
DecryptDatakeyRequest()
                .withVersionId(KMS_INTERFACE_VERSION).withBody(new
DecryptDatakeyRequestBody()
                        .withKeyId(keyId).withCipherText(cipherText).with
DatakeyCipherLength(AES_KEY_BYTE_LENGTH));

        // 6. Decrypt the DEK and convert the returned hexadecimal
plaintext key into a byte array.
        final byte[] decryptDataKey =
hexToBytes(kmsClient.decryptDatakey(decryptDatakeyRequest).getDataKey());

        // 7. Decrypt the file and store the decrypted file.
// iv at the end of the statement is the initial vector created in the
encryption example.
        doFileFinal(Cipher.DECRYPT_MODE, outEncryptFile, outDecryptFile,
decryptDataKey, iv);

        // 8. Compare the original file with the decrypted file.
        assert
getFileSha256Sum(inFile).equals(getFileSha256Sum(outDecryptFile));

    }

    /**
     * Encrypting and decrypting a file
     *
```



```
     * @param cipherMode: Encryption mode. It can be Cipher.ENCRYPT_MODE
or Cipher.DECRYPT_MODE.
     * @param infile: file to be encrypted or decrypted
     * @param outFile: file generated after encryption and decryption
     * @param keyPlain: plaintext key
     * @param iv: initial vector
     */
    static void doFileFinal(int cipherMode, File infile, File outFile,
byte[] keyPlain, byte[] iv) {

        try (BufferedInputStream bis = new BufferedInputStream(new
FileInputStream(infile));
             BufferedOutputStream bos = new BufferedOutputStream(new
FileOutputStream(outFile))) {
            final byte[] bytIn = new byte[(int) infile.length()];
            final int fileLength = bis.read(bytIn);

            assert fileLength > 0;

            final SecretKeySpec secretKeySpec = new
SecretKeySpec(keyPlain, AES_FLAG);
            final Cipher cipher = Cipher.getInstance(AES_ALG);
            final GCMParameterSpec gcmParameterSpec = new
GCMParameterSpec(GCM_TAG_LENGTH * Byte.SIZE, iv);
            cipher.init(cipherMode, secretKeySpec, gcmParameterSpec);
            final byte[] bytOut = cipher.doFinal(bytIn);
            bos.write(bytOut);
        } catch (Exception e) {
            throw new RuntimeException(e.getMessage());
        }
    }

    /**
* Converting a hexadecimal string to a byte array
     *
     * @param hexString: a hexadecimal string
     * @return: byte array
     */
    static byte[] hexToBytes(String hexString) {
        final int stringLength = hexString.length();
        assert stringLength > 0;
        final byte[] result = new byte[stringLength / 2];
        int j = 0;
        for (int i = 0; i < stringLength; i += 2) {
            result[j++] = (byte) Integer.parseInt(hexString.substring(i,
i + 2), 16);
        }
        return result;
    }

    /**
     * Calculate the SHA256 digest of the file.
     *
     * @param file
     * @return SHA256 digest
     */
     static String getFileSha256Sum(File file) {
        int length;
        MessageDigest sha256;
        byte[] buffer = new byte[1024];
        try {
            sha256 = MessageDigest.getInstance("SHA-256");
        } catch (NoSuchAlgorithmException e) {
            throw new RuntimeException(e.getMessage());
        }
        try (FileInputStream inputStream = new FileInputStream(file)) {
            while ((length = inputStream.read(buffer)) != -1) {
                sha256.update(buffer, 0, length);
            }
```



```
            return new BigInteger(1, sha256.digest()).toString(16);
        } catch (IOException e) {
            throw new RuntimeException(e.getMessage());
        }
    }
}
```

1.2 Uso do KMS para criptografar e descriptografar dados para serviços em nuvem

1.2.1 Visão geral

O KMS é um serviço de nuvem seguro, confiável e fácil de usar que ajuda os usuários a criar, gerenciar e proteger chaves de maneira centralizada.

Depois que seus serviços em nuvem estiverem integrados ao KMS, para criptografar dados na nuvem, basta selecionar uma CMK gerenciada pelo KMS para criptografia.

Você pode selecionar uma Chave principal padrão (DMK) criada automaticamente por um serviço de nuvem por meio do KMS ou uma chave criada ou importada para o KMS. Para obter detalhes, consulte **Diferenças entre uma CMK e uma Chave mestra padrão**.

Tabela 1-1 Serviços de nuvem que usam criptografia do KMS

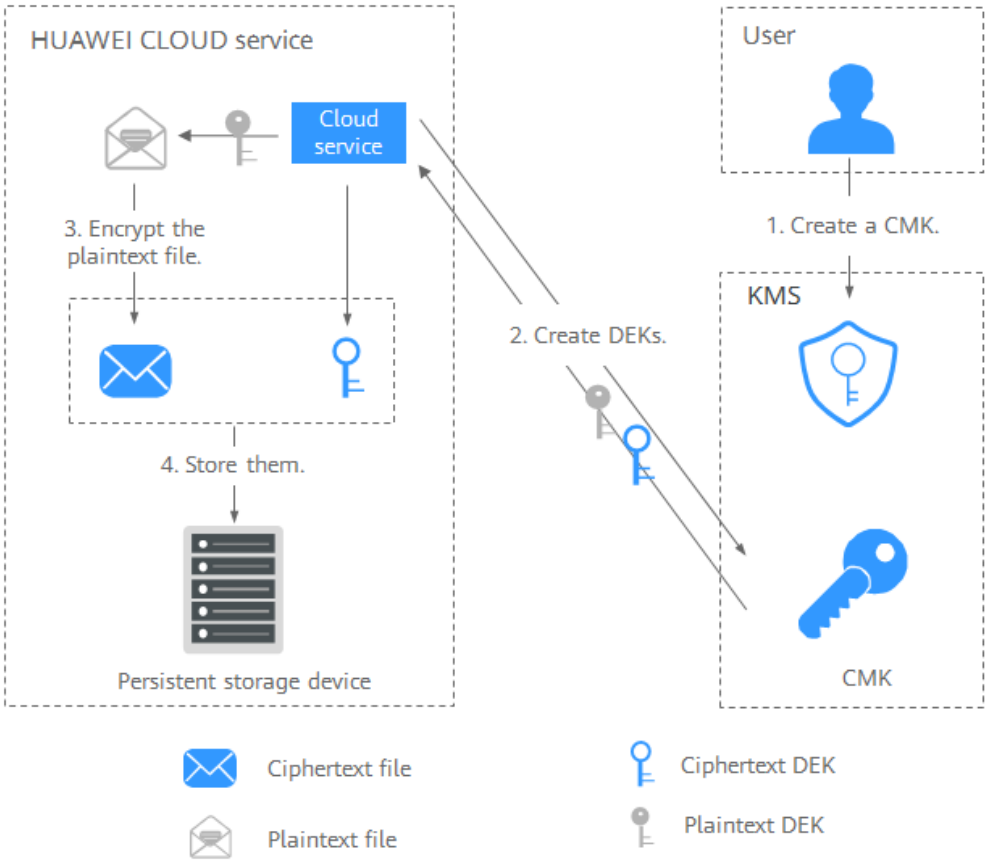
Categoria	Serviço	Modo de criptografia
Computação	Elastic Cloud Server (ECS)	Você pode criptografar uma imagem ou um disco do EVS no ECS. ● Ao criar um ECS, se você selecionar uma imagem criptografada, o disco do sistema do ECS criado terá automaticamente a criptografia ativada, com seu modo de criptografia igual ao modo de criptografia da imagem. ● Ao criar um ECS, você pode criptografar discos de dados adicionados.
	Image Management Service (IMS)	**Criptografia de dados no IMS**
Armazenamento	Object Storage Service (OBS)	**Criptografia de dados no OBS**
	Elastic Volume Service (EVS)	**Criptografia de dados no EVS**
	Volume Backup Service (VBS)	O VBS geralmente cria backups on-line para um único disco do EVS (sistema ou disco de dados) do servidor. Se for criptografado, seus dados de backup serão armazenados em modo criptografado.

Categoria	Serviço	Modo de criptografia
	Cloud Server Backup Service (CSBS)	O CSBS cria principalmente backups de consistência on-line para todos os discos do EVS do servidor. Os backups do CSBS também serão exibidos na página do VBS. Se for criptografado, seus dados de backup serão armazenados em modo criptografado.
Banco de dados	RDS for MySQL	**Criptografia de uma instância de banco de dados do RDS**
	RDS for PostgreSQL	
	RDS for SQL Server	
	Document Database Service (DDS)	**Criptografia de uma instância de banco de dados do DDS**

Processo de criptografia

Os serviços da HUAWEI CLOUD usam a tecnologia de criptografia de envelope e chamam as APIs do KMS para criptografar os recursos do serviço. Suas CMKs estão sob seu próprio gerenciamento. Com a sua concessão, os serviços da HUAWEI CLOUD usam uma CMK específica sua para criptografar dados.

Figura 1-6 Como a Huawei Cloud usa o KMS para criptografia



O processo de criptografia é o seguinte:

1. Crie uma CMK no KMS.
2. Um serviço da HUAWEI CLOUD chama a API **create-datakey** do KMS para criar uma DEK. Uma DEK de texto não criptografado e uma DEK de texto cifrado são geradas.

   NOTA

   As DEKs de texto cifrado são geradas quando você usa uma CMK para criptografar as DEKs de texto não criptografado.
3. O serviço da HUAWEI CLOUD usa a DEK de texto não criptografado para criptografar um arquivo de texto não criptografado, gerando um arquivo de texto cifrado.
4. O serviço da HUAWEI CLOUD salva a DEK de texto cifrado e o arquivo de texto cifrado juntos em um dispositivo de armazenamento permanente ou um serviço de armazenamento.

NOTA

Quando os usuários baixam os dados do serviço da HUAWEI CLOUD, o serviço usa a CMK especificada pelo KMS para descriptografar o texto cifrado de DEK, usa a DEK descriptografada para descriptografar dados, e, em seguida, fornece os dados descriptografados para os usuários baixarem.

1.2.2 Criptografia de dados no ECS

Visão geral

O KMS oferece suporte à criptografia de um clique para ECS. As imagens e os discos de dados do ECS podem ser criptografados.

- Ao criar um ECS, se você selecionar uma imagem criptografada, o disco do sistema do ECS criado terá automaticamente a criptografia ativada, com seu modo de criptografia igual ao modo de criptografia da imagem.
- Ao criar um ECS, você pode criptografar discos de dados adicionados.

Para obter detalhes sobre como criptografar uma imagem, consulte **Criptografia de dados no IMS**.

Para obter detalhes sobre como criptografar um disco de dados, consulte **Criptografia de dados no EVS**.

1.2.3 Criptografia de dados no OBS

Visão geral

Depois que a criptografia do lado do servidor é ativada, os dados de um objeto carregado no Object Storage Service (OBS) são criptografados no servidor antes de serem armazenados. Quando o objeto é baixado, os dados são descriptografados no servidor primeiro.

O KMS usa um módulo de segurança de hardware (HSM) de terceiros para proteger chaves, permitindo que você crie e gerencie chaves de criptografia facilmente. As chaves não são exibidas em texto não criptografado fora dos HSMs, o que impede a divulgação das chaves. Com o KMS, todas as operações em chaves são controladas e registradas, e os registros de uso de todas as chaves podem ser fornecidos para atender aos requisitos de conformidade regulatória.

A criptografia do lado do servidor com chaves gerenciadas pelo KMS (SSE-KMS) pode ser implementada para os objetos a serem carregados. Você precisa criar uma chave usando o KMS ou usar a chave padrão fornecida pelo KMS. Em seguida, você pode usar a chave para criptografar o objeto no servidor ao carregar o objeto no OBS.

Carregamento de arquivos no modo de criptografia no lado do servidor (no console)

Passo 1 Na lista de buckets no console do OBS, clique em um bucket para acessar a página **Overview**.

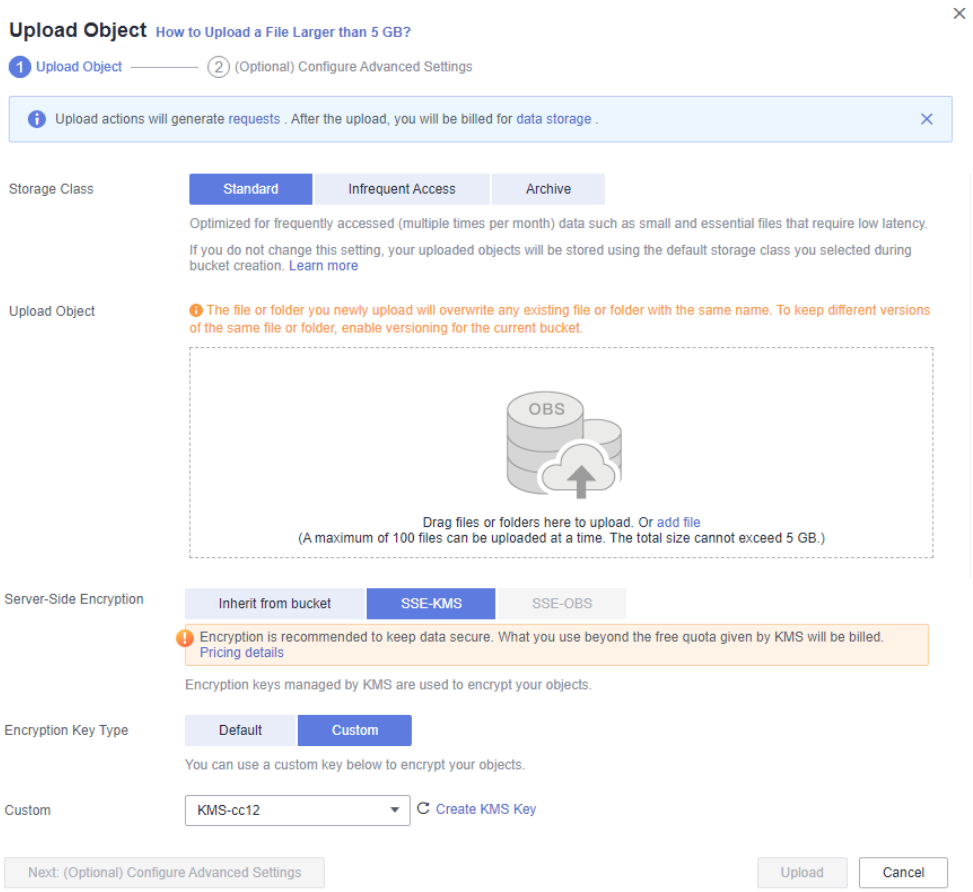
Passo 2 Na árvore de navegação à esquerda, escolha **Objects**.

Passo 3 Clique em **Upload Object**. A caixa de diálogo **Upload Object** é exibida.

Passo 4 Clique em **Add File**, selecione o arquivo a ser carregado e clique em **Open**.

Passo 5 Defina **Server-Side Encryption** como **SSE-KMS**, selecione a chave padrão ou uma chave personalizada e clique em **Upload**.

Figura 1-7 Criptografia de um objeto a ser carregado



Nome da chave: nome da chave personalizada. A chave é criada no DEW e é usada para proteção criptografada de dados. O OBS fornece uma chave padrão **obs/default**. Você pode usar a chave padrão ou criar uma chave no DEW.

NOTA

Geralmente, a chave AES é usada quando o SSE-KMS é usado. As chaves de criptografia SM4 podem ser usadas somente na região CN North-Ulanqab1.

Passo 6 Depois de carregar o objeto, clique nele para exibir seu status de criptografia.

NOTA

- O status de criptografia do objeto não pode ser alterado.
- Uma chave em uso não pode ser excluída. Caso contrário, o objeto criptografado com essa chave não pode ser baixado.

----Fim

Carregamento de arquivos no modo de criptografia do lado do servidor (por meio de uma API)

Você pode chamar a API necessária do OBS para carregar um arquivo no modo de SSE-KMS. Para obter detalhes, consulte *Referência de API do Object Storage Service*.

1.2.4 Criptografia de dados no EVS

Visão geral

Caso seus serviços exijam criptografia para os dados armazenados em discos no Elastic Volume Service (EVS), o EVS fornece a função de criptografia. Você pode criptografar discos do EVS recém-criados. As chaves usadas por discos do EVS criptografados são fornecidas pelo KMS do DEW, seguras e convenientes. Portanto, você não precisa estabelecer e manter a infraestrutura de gerenciamento de chaves.

A criptografia de disco é usada apenas para discos de dados. A criptografia de disco do sistema depende da imagem. Para mais detalhes, consulte **Criptografia de dados no IMS**.

Quem pode usar a função de criptografia de disco?

- Os administradores de segurança (usuários com direitos de Administrador de segurança) podem conceder os direitos de acesso do KMS ao EVS para usar a criptografia de disco.
- Quando um usuário comum que não tem os direitos de Administrador de segurança precisa usar o recurso de criptografia de disco, a condição varia dependendo se o usuário é o primeiro na região ou projeto atual a usar esse recurso.
  - Se o usuário for o primeiro, ele deverá entrar em contato com um usuário que tenha os direitos de Administrador de segurança para conceder os direitos de acesso do KMS ao EVS. Em seguida, o usuário pode usar o recurso de criptografia de disco.
  - Se o usuário não é o primeiro, o usuário pode usar a função de criptografia de disco diretamente.

Do ponto de vista de um locatário, desde que os direitos de acesso do KMS tenham sido concedidos ao EVS em uma região, todos os usuários na mesma região podem usar diretamente o recurso de criptografia de disco.

Se houver vários projetos na região atual, os direitos de acesso do KMS precisam ser concedidos a cada projeto nessa região.

Chaves usadas para criptografia de disco do EVS

As chaves fornecidas pelo KMS para criptografia de disco incluem uma Chave mestra padrão e Chaves mestras do cliente (CMKs).

- Chave mestra padrão: uma chave que é criada automaticamente pelo EVS por meio do KMS e denominada **evs/default**.

  A Chave mestra padrão não pode ser desativada e não oferece suporte à exclusão agendada.
- CMKs: chaves criadas pelos usuários. Você pode usar CMKs existentes ou criar uma. Para obter detalhes, consulte **Criação de uma CMK**.

Se os discos forem criptografados usando uma CMK, que é desativada ou agendada para exclusão, os discos não poderão mais ser lidos ou gravados, e os dados nesses discos talvez nunca sejam restaurados. Consulte **Tabela 1-2** para obter mais informações.

Tabela 1-2 Impacto em discos criptografados depois que uma CMK se torna indisponível

Status da CMK	Impacto em discos criptografados	Método de restauração
Disabled	• Se um disco criptografado for anexado a um ECS, o disco ainda poderá ser usado, mas as operações normais de leitura/gravação não serão garantidas permanentemente. • Se um disco criptografado for desanexado, a reanexação do disco falhará.	Ative a CMK. Para obter detalhes, consulte **Ativação de um ou mais CMKs**.
Pending deletion		Cancele a exclusão programada para a CMK. Para obter detalhes, consulte **Cancelamento da exclusão programada de um ou mais CMKs**.
Deleted		Os dados nos discos nunca podem ser restaurados.

AVISO

Você será cobrado pelas CMKs que usar. Se forem usadas chaves básicas, verifique se o saldo de sua conta é suficiente. Se forem usadas chaves profissionais, renove seu pedido em tempo hábil. Caso contrário, seus serviços podem ser interrompidos e seus dados podem nunca ser restaurados, pois os discos criptografados se tornam ilegíveis e não graváveis.

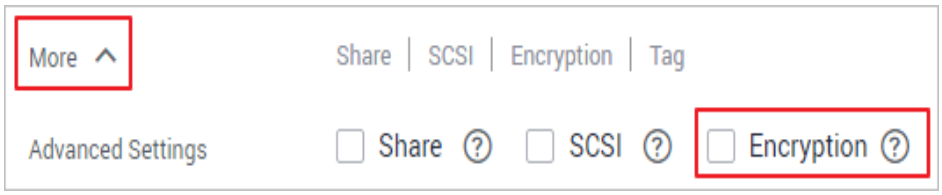
Uso do KMS para criptografar um disco (no console)

Passo 1 No console de gerenciamento do EVS, clique em **Buy Disk**.

Passo 2 Selecione a caixa de seleção **Encryption**.

1. Clique em **More**. A caixa de seleção **Encryption** é exibida.

Figura 1-8 Mais



2. Crie uma agência.

   Selecione **Encrypt**. Se o EVS não estiver autorizado a acessar o KMS, a caixa de diálogo **Create Agency** será exibida. Nesse caso, clique em **Yes** para autorizá-lo. Após a autorização, o EVS pode obter chaves do KMS para criptografar e descriptografar discos.

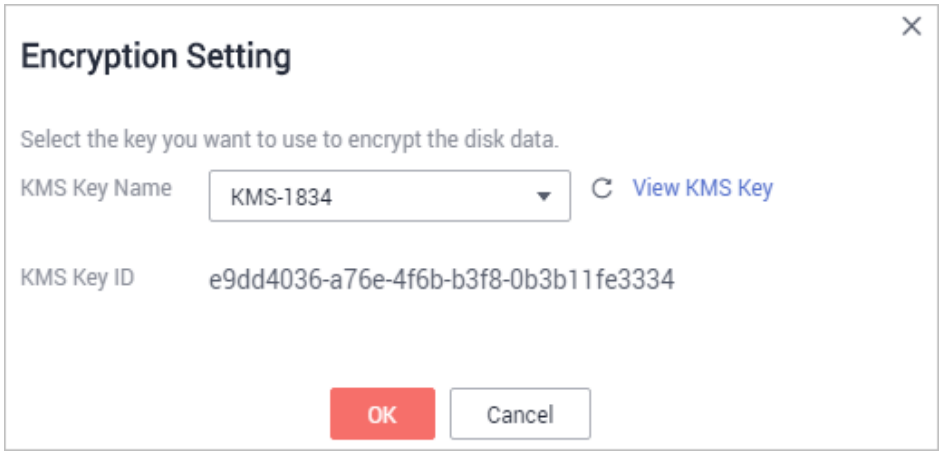
   NOTA

   Antes de usar a função de criptografia de disco, os direitos de acesso do KMS precisam ser concedidos ao EVS. Se você tem o direito de conceder, conceda os direitos de acesso do KMS diretamente ao EVS. Se você não tiver o direito, entre em contato com um usuário com os direitos de Administrador de segurança para conceder os direitos de acesso do KMS ao EVS e repita as operações anteriores.

3. Defina parâmetros de criptografia.

   Selecione **Encrypt**. Se a autorização for bem-sucedida, a caixa de diálogo de configuração de criptografia será exibida.

Figura 1-9 Configurações de criptografia



Selecione um dos seguintes tipos de chaves na lista suspensa **KMS Key Name**:

- Chave mestra padrão. Depois que os direitos de acesso do KMS forem concedidos ao EVS, o sistema criará automaticamente uma Chave mestra padrão chamada **evs/default**.
- Uma CMK existente ou nova. Para obter detalhes sobre como criar uma, consulte **Criação de uma CMK**.

Passo 3 Configure outros parâmetros para o disco. Para obter detalhes sobre os parâmetros, consulte **Comprar um disco do EVS**.

----Fim

Uso do KMS para criptografar um disco (por meio de uma API)

Você pode chamar a API necessária do EVS para comprar um disco do EVS criptografado. Para obter detalhes, consulte *Referência de API do Elastic Volume Service*.

1.2.5 Criptografia de dados no IMS

Você pode criar uma imagem criptografada no Image Management Service (IMS) para armazenar dados com segurança.

Restrições

- O DEW deve ser ativado.
- Uma imagem criptografada não pode ser compartilhada com outros usuários.
- Uma imagem criptografada não pode ser publicada no Marketplace.
- Se um ECS tiver um disco do sistema criptografado, a imagem privada criada usando o ECS também será criptografada.
- A chave usada para criptografar uma imagem não pode ser alterada.
- Se a chave usada para criptografar uma imagem for desativada ou excluída, a imagem não estará disponível.
- O disco do sistema de um ECS criado usando uma imagem criptografada também é criptografado, e sua chave é a mesma que a chave de imagem.

Uso do KMS para criptografar uma imagem privada (no console)

Você pode criar uma imagem criptografada usando um ECS criptografado ou um arquivo de imagem externo.

- Crie uma imagem criptografada usando um ECS criptografado.

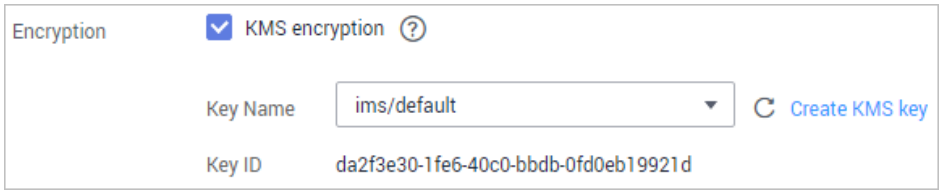
  Quando você usa um ECS para criar uma imagem privada, se o disco do sistema do ECS estiver criptografado, a imagem privada criada usando o ECS também será criptografada. A chave usada para criptografar a imagem é aquela usada para criar o disco do sistema.
- Crie uma imagem criptografada usando um arquivo externo de imagem.

  Quando você usa um arquivo de imagem externo que foi carregado em um bucket do OBS para criar uma imagem privada, pode selecionar a criptografia do KMS ao registrar a imagem para criptografá-la.

  Ao fazer upload de um arquivo de imagem, você pode selecionar **KMS encryption** e usar uma chave fornecida pelo KMS para criptografar o arquivo carregado, conforme mostrado em **Figura 1-10**.

  a. No console de gerenciamento do IMS, clique em **Create Private Image**.
  b. Defina **Type** como **System disk image**.
  c. Defina **Source** como **Image File**.
  d. Selecione **KMS encryption**.

  Figura 1-10 Criptografia de dados no IMS



Selecione um dos seguintes tipos de chaves na lista suspensa **Key Name**:

- Chave principal padrão **ims/default** criada pelo KMS
- Uma CMK existente ou nova. Para obter detalhes sobre como criar uma, consulte **Criação de uma CMK**.

e. Configure outros parâmetros. Para obter detalhes sobre os parâmetros, consulte **Registro de uma imagem**.

Uso do KMS para criptografar uma imagem privada (por meio de uma API)

Você pode chamar a API necessária do IMS para criptografar o arquivo de imagem. Para obter detalhes, consulte *Referência de API do Image Management Service*.

1.2.6 Criptografia de uma instância de banco de dados do RDS

Visão geral

O Relational Database Service (RDS) suporta mecanismos MySQL e PostgreSQL.

Depois que a criptografia for ativada, os dados do disco serão criptografados e armazenados no servidor quando você criar uma instância de banco de dados ou expandir a capacidade do disco. Quando você baixa objetos criptografados, os dados criptografados serão descriptografados no servidor e exibidos em texto não criptografado.

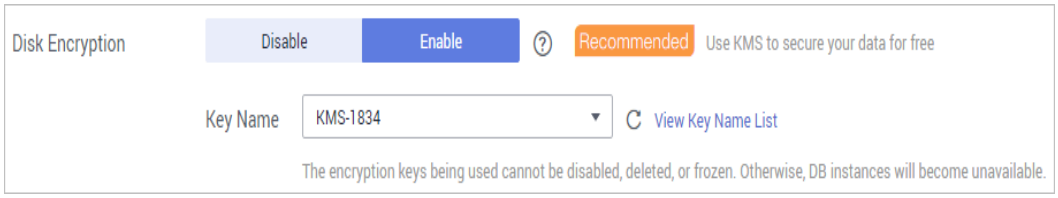
Restrições

- O direito de KMS Administrator deve ser concedido ao usuário na região do RDS usando o Identity and Access Management (IAM). Para obter detalhes sobre como atribuir permissões a grupos de usuários, consulte. "Como gerenciar grupos de usuários e conceder permissões a eles?" no *Guia de usuário do Identity and Access Management*.
- Para usar uma chave definida pelo usuário para criptografar objetos a serem carregados, crie uma chave usando DEW. Para obter detalhes, consulte **Criação de uma CMK**.
- Depois que a função de criptografia de disco estiver ativada, você não poderá desativá-la ou alterar a chave após a criação de uma instância de banco de dados. Os dados de backup armazenados no OBS não serão criptografados.
- Depois que uma instância de banco de dados do RDS for criada, não ative nem exclua a chave que está sendo usada. Caso contrário, RDS ficará indisponível e os dados não poderão ser restaurados.
- Se você ampliar uma instância de BD com discos criptografados, o espaço de armazenamento expandido será criptografado usando a chave de criptografia original.

Uso do KMS para criptografar uma instância de BD (no console)

Quando um usuário compra uma instância de banco de dados do Relational Database Service (RDS), ele pode selecionar **Disk encryption** e usar a chave fornecida pelo KMS para criptografar o disco da instância de banco de dados. Para obter mais informações, consulte **Comprar uma instância de banco de dados do MySQL** e **Comprar uma instância de banco de dados do PostgreSQL**.

Figura 1-11 Criptografia de dados no RDS



Uso do KMS para criptografar uma instância de banco de dados (por meio de uma API)

Você também pode chamar a API necessária do RDS para comprar instâncias de banco de dados criptografadas. Para obter detalhes, consulte *Referência de API do Relational Database Service*.

1.2.7 Criptografia de uma instância de banco de dados do DDS

Visão geral

Depois que a criptografia for ativada, os dados do disco serão criptografados e armazenados no servidor quando você criar uma instância de banco de dados ou expandir a capacidade do disco. Quando você baixa objetos criptografados, os dados criptografados serão descriptografados no servidor e exibidos em texto não criptografado.

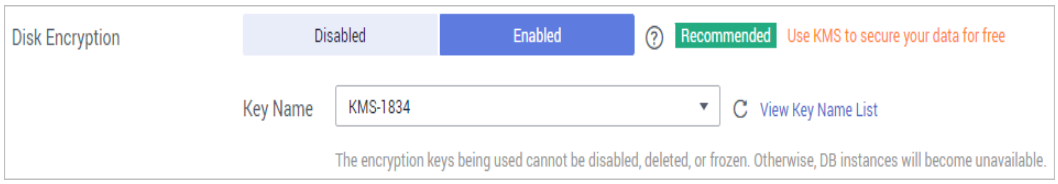
Restrições

- O direito de KMS Administrator deve ser adicionado na região do RDS usando o IAM. Para obter detalhes sobre como atribuir permissões a grupos de usuários, consulte. "Como gerenciar grupos de usuários e conceder permissões a eles?" no *Guia de usuário do Identity and Access Management*.
- Para usar uma chave definida pelo usuário para criptografar objetos a serem carregados, crie uma chave usando DEW. Para obter detalhes, consulte **Criação de uma CMK**.
- Depois que a função de criptografia de disco estiver ativada, você não poderá desativá-la ou alterar a chave após a criação de uma instância de banco de dados. Os dados de backup armazenados no OBS não serão criptografados.
- Depois que uma instância de banco de dados do DDS (Document Database Service) for criada, não desative ou exclua a chave que está sendo usada. Caso contrário, o DDS não estará disponível e os dados não poderão ser restaurados.
- Se você ampliar uma instância de BD com discos criptografados, o espaço de armazenamento expandido será criptografado usando a chave de criptografia original.

Uso do KMS para criptografar uma instância de BD (no console)

Ao comprar uma instância de banco de dados no DDS, você pode definir **Disk Encryption** como **Enable** e usar a chave fornecida pelo KMS para criptografar o disco da instância de banco de dados. Para obter mais informações, consulte **Compra de uma instância de cluster**.

Figura 1-12 Criptografia de dados no DDS



Uso do KMS para criptografar uma instância de banco de dados (por meio de uma API)

Você também pode chamar a API necessária do DDS para comprar instâncias de banco de dados criptografadas. Para obter detalhes, consulte *Referência de API do Document Database Service*.

1.3 Uso do SDK de criptografia para criptografar e descriptografar arquivos locais

Você pode usar certos algoritmos para criptografar seus arquivos, protegendo-os contra violação ou adulteração.

O **SDK de criptografia** é uma biblioteca de senhas de cliente que pode criptografar e descriptografar fluxos de dados e arquivos. Você pode facilmente criptografar e descriptografar grandes quantidades de dados simplesmente chamando APIs. Ele permite que você se concentre no desenvolvimento das principais funções de suas aplicações sem se distrair com os processos de criptografia e descriptografia de dados.

NOTA

Para mais informações, visite. Para obter detalhes, consulte **Detalhes**.

Cenário

Se arquivos e imagens grandes forem enviados ao KMS por meio de HTTPS para criptografia, um grande número de recursos de rede será consumido e a criptografia será lenta. Esta seção descreve como criptografar rapidamente uma grande quantidade de dados.

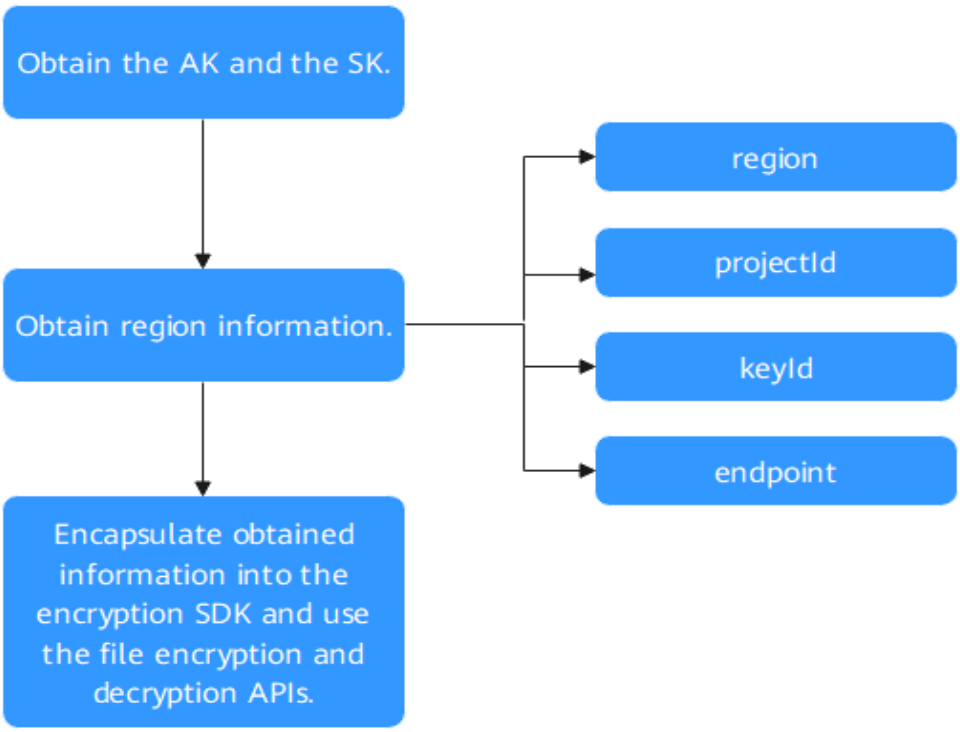
Solução

O SDK de criptografia executa a criptografia de envelope em fluxos de arquivos, segmento por segmento.

Os dados são criptografados no SDK usando a DEK gerada pelo KMS. A criptografia segmentada de arquivos na memória garante a segurança e a correção da criptografia de arquivos, porque não requer transferência de arquivos pela rede.

O SDK carrega um arquivo na memória e o processa segmento por segmento. O próximo segmento não será lido antes que a criptografia ou descriptografia do segmento atual seja concluída.

Processo



Procedimento

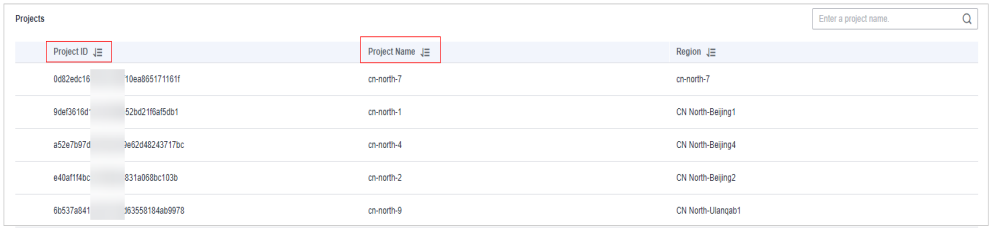
Passo 1 Obtenha o AK e a SK.

- ACCESS_KEY: chave de acesso da conta da Huawei. Para obter detalhes, consulte **Como obter uma chave de acesso (AK/SK)?**
- SECRET_ACCESS_KEY: chave de acesso de segredo da conta da Huawei. Para obter detalhes, consulte **Como obter uma chave de acesso (AK/SK)?**
- Haverá riscos de segurança se a AK/SK usada para autenticação for gravada diretamente no código. Criptografe a AK/SK no arquivo de configuração ou nas variáveis de ambiente para armazenamento.
- Neste exemplo, a AK/SK armazenada nas variáveis de ambiente é usada para autenticação de identidade. Configure as variáveis de ambiente **HUAWEICLOUD_SDK_AK** e **HUAWEICLOUD_SDK_SK** primeiro no ambiente local.

Passo 2 Obtenha informações da região.

1. **Faça logon no console de gerenciamento**.
2. Passe o mouse sobre o nome de usuário no canto superior direito e escolha **My Credentials** na lista suspensa.
3. Obtenha o **Project ID** e o **Project Name**.

Figura 1-13 Obtenção do ID e do nome do projeto



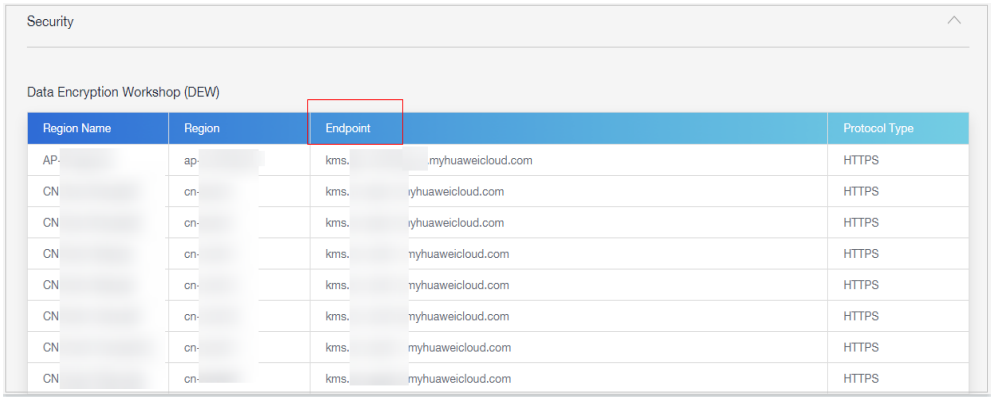
4. Clique em ☰. Escolha **Security & Compliance** > **Data Encryption Workshop**.
5. Obtenha o ID da CMK (**KEYID**) a ser usado na região atual.

Figura 1-14 Obtenção do ID da CMK

6. Obtenha o ponto de extremidade (**ENDPOINT**) exigido pela região atual.

   Um ponto de extremidade é o **request address** para chamar uma API. Os pontos de extremidade variam de acordo com os serviços e as regiões. Para obter os pontos de extremidade de todos os serviços, consulte **Regiões e pontos de extremidade**.

Figura 1-15 Obtenção de um ponto de extremidade



Passo 3 Criptografe e descriptografe um arquivo.

```
public class KmsEncryptFileExample {

    private static final String ACCESS_KEY = System.getenv("HUAWEICLOUD_SDK_AK");
    private static final String SECRET_ACCESS_KEY =
System.getenv("HUAWEICLOUD_SDK_SK");
    private static final String PROJECT_ID = "<projectId>";
    private static final String REGION = "<region>";
    private static final String KEYID = "<keyId>";
    public static final String ENDPOINT = "<endpoint>";

    public static void main(String[] args) throws IOException {
        // Source file path
        String encryptFileInPutPath = args[0];
        // Path of the encrypted ciphertext file
        String encryptFileOutPutPath = args[1];
        // Path of the decrypted file
        String decryptFileOutPutPath = args[2];
        // Encryption context
```



```
        Map<String, String> encryptContextMap = new HashMap<>();
        encryptContextMap.put("encryption", "context");
        encryptContextMap.put("simple", "test");
        encryptContextMap.put("caching", "encrypt");
        // Construct the encryption configuration
        HuaweiConfig config = HuaweiConfig.builder().buildSk(SECRET_ACCESS_KEY)
                .buildAk(ACCESS_KEY)
                .buildKmsConfig(Collections.singletonList(new KMSConfig(REGION,
KEYID, PROJECT_ID, ENDPOINT)))
                .buildCryptoAlgorithm(CryptoAlgorithm.AES_256_GCM_NOPADDING)
                .build();
        HuaweiCrypto huaweiCrypto = new HuaweiCrypto(config);
        // Set the key ring.
        huaweiCrypto.withKeyring(new
KmsKeyringFactory().getKeyring(KeyringTypeEnum.KMS_MULTI_REGION.getType()));
        // Encrypt the file.
        encryptFile(encryptContextMap, huaweiCrypto, encryptFileInPutPath,
encryptFileOutPutPath);
        // Decrypt the file.
        decryptFile(huaweiCrypto, encryptFileOutPutPath, decryptFileOutPutPath);
    }

    private static void encryptFile(Map<String, String> encryptContextMap,
HuaweiCrypto huaweiCrypto,
                                    String encryptFileInPutPath, String
encryptFileOutPutPath) throws IOException {
        // fileInputStream: input stream corresponding to the encrypted file
        FileInputStream fileInputStream = new
FileInputStream(encryptFileInPutPath);
        // fileOutputStream: output stream corresponding to the source file
        FileOutputStream fileOutputStream = new
FileOutputStream(encryptFileOutPutPath);
        // Encryption
        huaweiCrypto.encrypt(fileInputStream, fileOutputStream,
encryptContextMap);
        fileInputStream.close();
        fileOutputStream.close();
    }

    private static void decryptFile(HuaweiCrypto huaweiCrypto, String
decryptFileInPutPath, String decryptFileOutPutPath) throws IOException {
        // in: input stream corresponding to the source file
        FileInputStream fileInputStream = new
FileInputStream(decryptFileInPutPath);
        // out: output stream corresponding to the encrypted file
        FileOutputStream fileOutputStream = new
FileOutputStream(decryptFileOutPutPath);
        // Decryption
        huaweiCrypto.decrypt(fileInputStream, fileOutputStream);
        fileInputStream.close();
        fileOutputStream.close();
    }
}
```

----Fim

1.4 Criptografia e descriptografia de dados por meio de DR entre regiões

Cenário

Se ocorrer uma falha durante a criptografia ou descriptografia em uma região, você poderá usar o KMS para implementar a criptografia e a descriptografia de DR entre regiões, garantindo a continuidade do serviço.

Solução

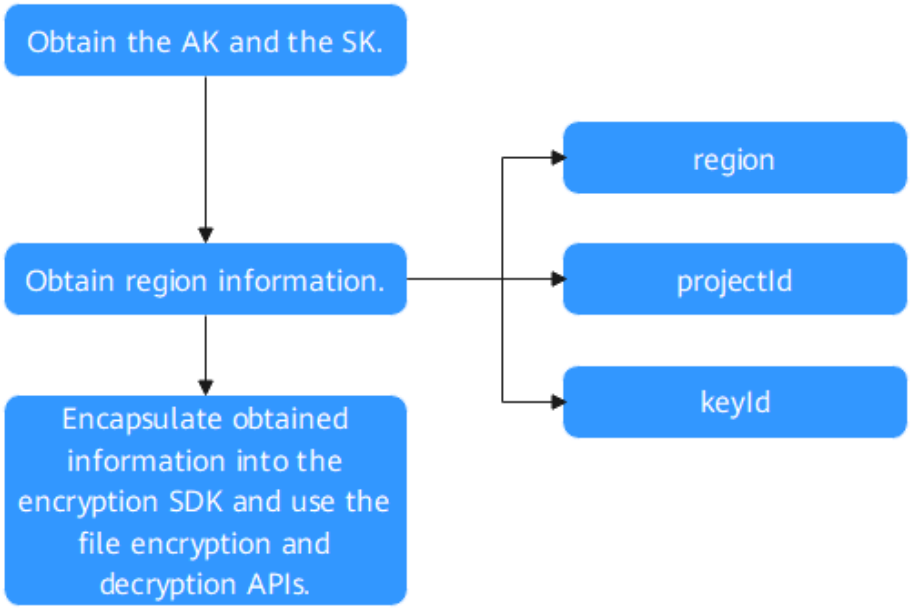
Se o KMS estiver defeituoso em uma ou várias regiões, a criptografia e a descriptografia podem ser concluídas, desde que uma chave no chaveiro esteja disponível.

Um chaveiro entre regiões pode usar as CMKs de várias regiões para criptografar uma parte dos dados e gerar texto cifrado de dados exclusivo. Para descriptografar os dados, basta usar um chaveiro que contenha uma ou mais CMKs disponíveis que foram usadas para criptografar os dados.

NOTA

Para obter mais informações, consulte **Detalhes**.

Processo



Procedimento

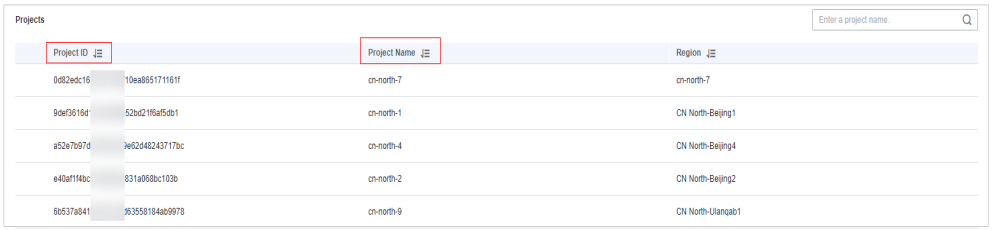
Passo 1 Obtenha o AK e a SK.

- ACCESS_KEY: chave de acesso da conta da Huawei. Para obter detalhes, consulte **Como obter uma chave de acesso (AK/SK)?**
- SECRET_ACCESS_KEY: chave de acesso de segredo da conta da Huawei. Para obter detalhes, consulte **Como obter uma chave de acesso (AK/SK)?**
- Haverá riscos de segurança se a AK/SK usada para autenticação for gravada diretamente no código. Criptografe a AK/SK no arquivo de configuração ou nas variáveis de ambiente para armazenamento.
- Neste exemplo, a AK/SK armazenada nas variáveis de ambiente é usada para autenticação de identidade. Configure as variáveis de ambiente **HUAWEICLOUD_SDK_AK** e **HUAWEICLOUD_SDK_SK** primeiro no ambiente local.

Passo 2 Obtenha informações da região.

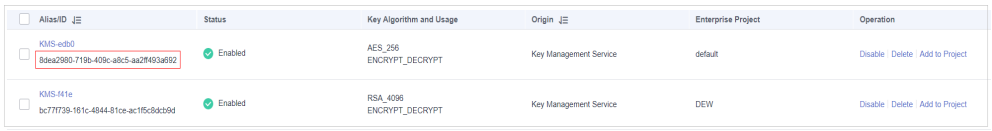
1. **Faça logon no console de gerenciamento**.
2. Passe o mouse sobre o nome de usuário no canto superior direito e escolha **My Credentials** na lista suspensa.
3. Obtenha o **Project ID** e o **Project Name**.

Figura 1-16 Obtenção do ID e do nome do projeto



4. Clique em ☰. Escolha **Security & Compliance** > **Data Encryption Workshop**.
5. Obtenha o ID da CMK (**KEYID**) a ser usado na região atual.

Figura 1-17 Obtenção do ID da CMK



Passo 3 Use o chaveiro para criptografia e descriptografia.

```
public class KmsEncryptionExample {
    private static final String ACCESS_KEY = System.getenv("HUAWEICLOUD_SDK_AK");
    private static final String SECRET_ACCESS_KEY =
System.getenv("HUAWEICLOUD_SDK_SK");

    private static final String PROJECT_ID_1 = "<projectId1>";
    private static final String REGION_1 = "<region1>";
    private static final String KEYID_1 = "<keyId1>";

    public static final String PROJECT_ID_2 = "<projectId2>";
    public static final String REGION_2 = "<region2>";
    public static final String KEYID_2 = "<keyId2>";

    // Data to be encrypted
    private static final String PLAIN_TEXT = "Hello World!";

    public static void main(String[] args) {
        // CMK list
        List<KMSConfig> kmsConfigList = new ArrayList<>();
        kmsConfigList.add(new KMSConfig(REGION_1, KEYID_1, PROJECT_ID_1));
        kmsConfigList.add(new KMSConfig(REGION_2, KEYID_2, PROJECT_ID_2));
        // Construct encryption-related information.
        HuaweiConfig multiConfig =
HuaweiConfig.builder().buildSk(SECRET_ACCESS_KEY)
                .buildAk(ACCESS_KEY)
                .buildKmsConfig(kmsConfigList)
                .buildCryptoAlgorithm(CryptoAlgorithm.AES_256_GCM_NOPADDING)
                .build();
        // Select a key ring.
        KMSKeyring keyring = new
KmsKeyringFactory().getKeyring(KeyringTypeEnum.KMS_MULTI_REGION.getType());
        HuaweiCrypto huaweiCrypto = new
HuaweiCrypto(multiConfig).withKeyring(keyring);
        // Encryption context
        Map<String, String> encryptContextMap = new HashMap<>();
```



```
        encryptContextMap.put("key", "value");
        encryptContextMap.put("context", "encrypt");
        // Encryption
        CryptoResult<byte[]> encryptResult = huaweiCrypto.encrypt(new
EncryptRequest(encryptContextMap, PLAIN_TEXT.getBytes(StandardCharsets.UTF_8)));
        // Decryption
        CryptoResult<byte[]> decryptResult =
huaweiCrypto.decrypt(encryptResult.getResult());
        Assert.assertEquals(PLAIN_TEXT, new String(decryptResult.getResult()));
    }
}
```

----Fim

1.5 Uso do KMS para proteger a integridade dos arquivos

Cenário

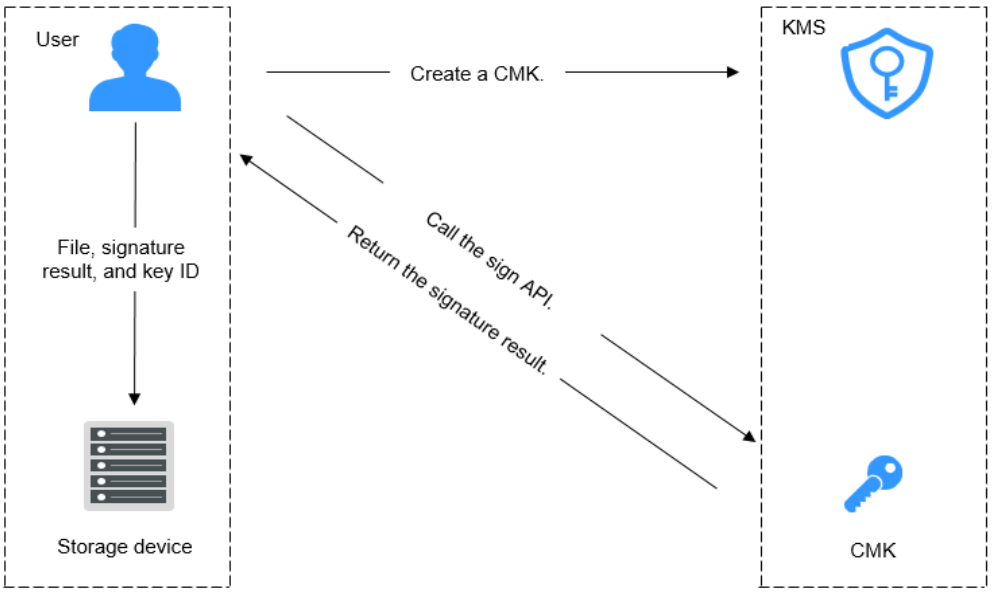
Quando uma grande quantidade de arquivos (como imagens, apólices de seguro eletrônicas e arquivos importantes) precisar ser transmitida ou armazenada com segurança, você poderá usar o KMS para assinar o resumo do arquivo. Quando os arquivos são usados novamente, você pode recalcular o resumo para verificação de assinatura. Certifique-se de que os arquivos não sejam adulterados durante a transmissão ou armazenamento.

Solução

Crie uma CMK no KMS.

Calcule o resumo do arquivo e chame a API de assinatura do KMS para assinar o resumo. O resultado da assinatura do resumo é obtido. Transmita ou armazene o resultado da assinatura do resumo, o ID da chave e o arquivo juntos. A figura a seguir mostra o processo de assinatura.

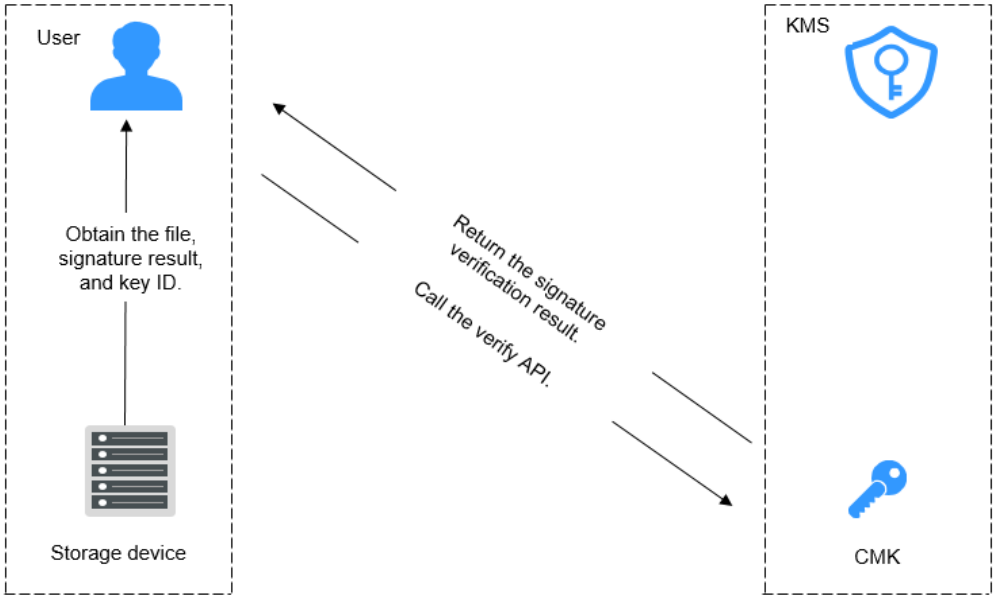
Figura 1-18 Processo de assinatura



Antes de usar um arquivo, você precisa verificar a integridade do arquivo para garantir que o arquivo não seja adulterado.

Recalcule o resumo do arquivo e chame a API de verificação do KMS com o valor da assinatura para verificar a assinatura do resumo. O resultado da verificação da assinatura é obtido. Se a assinatura for verificada, o arquivo não foi adulterado. A figura a seguir mostra o processo de verificação de assinatura.

Figura 1-19 Processo de verificação de assinatura.



Procedimento

Passo 1 Obtenha o AK e a SK.

- ACCESS_KEY: chave de acesso da conta da Huawei. Para obter detalhes, consulte **Como obter uma chave de acesso (AK/SK)?**
- SECRET_ACCESS_KEY: chave de acesso de segredo da conta da Huawei. Para obter detalhes, consulte **Como obter uma chave de acesso (AK/SK)?**
- Haverá riscos de segurança se a AK/SK usada para autenticação for gravada diretamente no código. Criptografe a AK/SK no arquivo de configuração ou nas variáveis de ambiente para armazenamento.
- Neste exemplo, a AK/SK armazenada nas variáveis de ambiente é usada para autenticação de identidade. Configure as variáveis de ambiente **HUAWEICLOUD_SDK_AK** e **HUAWEICLOUD_SDK_SK** primeiro no ambiente local.

Passo 2 Obtenha informações da região.

- **Faça logon no console de gerenciamento**.

Passo 3 Use o KMS para assinar o arquivo e verificar a assinatura.

```
public class FileStreamSignVerifyExample {

    /**
     * Basic authentication information:
     * - ACCESS_KEY: access key of the Huawei Cloud account
```



```
     * - SECRET_ACCESS_KEY: secret access key of the Huawei Cloud account, which
is sensitive information. Store this in ciphertext.
     * - IAM_ENDPOINT: endpoint for accessing IAM. For details, see Regions and
Endpoints.
     * - KMS_REGION_ID: regions supported by KMS. For details, see Regions and
Endpoints.
     * - KMS_ENDPOINT: endpoint for accessing KMS. For details, see Regions and
Endpoints.
     */
    private static final String ACCESS_KEY = System.getenv("HUAWEICLOUD_SDK_AK");
    private static final String SECRET_ACCESS_KEY =
System.getenv("HUAWEICLOUD_SDK_SK");
    private static final String IAM_ENDPOINT = "https://<IamEndpoint>";
    private static final String KMS_REGION_ID = "<RegionId>";
    private static final String KMS_ENDPOINT = "https://<KmsEndpoint>";

    public static void main(String[] args) {
        // CMK ID. Select a key whose usage contains SIGN_VERIFY.
        final String keyId = args[0];

        signAndVerifyFile(keyId);
    }

    /**
     * Use KMS to sign the file and verify the signature.
     *
     * @param keyId: CMK ID
     */
    static void signAndVerifyFile(String keyId) {
        // 1. Prepare the authentication information for accessing HUAWEI CLOUD.
        final BasicCredentials auth = new BasicCredentials()
                .withIamEndpoint(IAM_ENDPOINT).withAk(ACCESS_KEY).withSk(SECRET_AC
CESS_KEY);

        // 2. Initialize the SDK and transfer the authentication information and
the address for the KMS to access the client.
        final KmsClient kmsClient = KmsClient.newBuilder()
                .withRegion(new Region(KMS_REGION_ID,
KMS_ENDPOINT)).withCredential(auth).build();

        // 3. Prepare the file to be signed.
        // inFile File to be signed
        final File inFile = new File("FirstSignFile.iso");
        final String fileSha256Sum = getFileSha256Sum(inFile);

        // 4. Calculate the digest and select a proper signature algorithm based
on the key type.
        final SignRequest signRequest = new SignRequest().withBody(
                new
SignRequestBody().withKeyId(keyId).withSigningAlgorithm(SignRequestBody.SigningAlg
orithmEnum.RSASSA_PSS_SHA_256)
                        .withMessageType(SignRequestBody.MessageTypeEnum.DIGEST).w
ithMessage(fileSha256Sum));

        final SignResponse signResponse = kmsClient.sign(signRequest);

        // 5. Verify the digest.
        final ValidateSignatureRequest validateSignatureRequest = new
ValidateSignatureRequest().withBody(
                new
VerifyRequestBody().withKeyId(keyId).withMessage(fileSha256Sum).withSignature(sign
Response.getSignature())
                        .withSigningAlgorithm(VerifyRequestBody.SigningAlgorithmEn
um.RSASSA_PSS_SHA_256)
                        .withMessageType(VerifyRequestBody.MessageTypeEnum.DIGEST)
);
        final ValidateSignatureResponse validateSignatureResponse =
kmsClient.validateSignature(validateSignatureRequest);
```



```
        // 6. Compare the digest result.
        assert
validateSignatureResponse.getSignatureValid().equalsIgnoreCase("true");

    }

    /**
     * Calculate the SHA256 digest of the file.
     *
     * @param file
     * @return SHA256 digest in Base64 format
     */
    static String getFileSha256Sum(File file) {
        int length;
        MessageDigest sha256;
        byte[] buffer = new byte[1024];
        try {
            sha256 = MessageDigest.getInstance("SHA-256");
        } catch (NoSuchAlgorithmException e) {
            throw new RuntimeException(e.getMessage());
        }
        try (FileInputStream inputStream = new FileInputStream(file)) {
            while ((length = inputStream.read(buffer)) != -1) {
                sha256.update(buffer, 0, length);
            }
            return Base64.getEncoder().encodeToString(sha256.digest());
        } catch (IOException e) {
            throw new RuntimeException(e.getMessage());
        }
    }

}
```

----Fim

2 Serviço de gerenciamento de segredo em nuvem

2.1 Uso do CSMS para alterar senhas de contas de banco de dados codificadas rigidamente

Geralmente, os segredos usadospara acesso estão incorporados nas aplicações. Para atualizar um segredo, você precisa criar um novo segredo e gastar tempo atualizando suas aplicações. Se você tiver várias aplicações usando o mesmo segredo, terá que atualizar todas elas, ou as aplicações que você se esqueceu de atualizar não poderão usar o segredo para logon.

Uma ferramenta de gerenciamento de segredos fácil de usar, eficaz e segura será útil.

O Cloud Secret Management Service (CSMS) tem as seguintes vantagens:

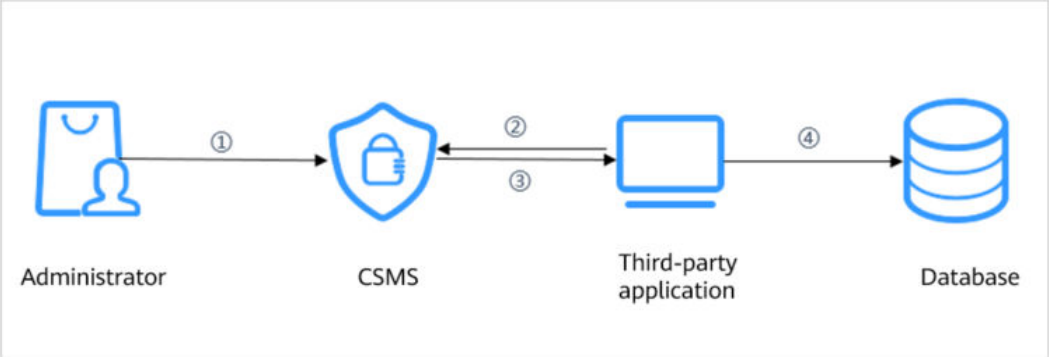
- Você pode hospedar seus segredos em vez de usar segredos codificados rigidamente, melhorando a segurança de dados e ativos.
- Seus serviços não são afetados quando você **faz a rotação manual** dos segredos.
- O acesso seguro ao SDK permite que você chame seus segredos dinamicamente.
- Você pode armazenar muitos tipos de segredos. Você pode armazenar contas de serviço, senhas e informações de banco de dados, incluindo, mas não limitado a, nomes de banco de dados, endereços IP e números de porta.

Fazer logon em um banco de dados usando segredos

Você pode criar um segredo e fazer logon no seu banco de dados chamando o segredo por meio de uma API.

Verifique se sua conta tem a permissão KMS Administrator ou KMS CMKFullAccess. Para obter detalhes, consulte **Gerenciamento de permissões do DEW**.

Figura 2-1 Processo de logon baseado em segredo



O processo é o seguinte:

Passo 1 Crie um segredo no **console** ou por meio de uma **API** para armazenar informações do banco de dados (como endereço, porta e senha do banco de dados).

Passo 2 Use uma aplicação para acessar o banco de dados. CSMS irá consultar o segredo criado em **1**.

Passo 3 O CSMS recupera e descriptografa o texto cifrado de segredo e retorna com segurança as informações armazenadas no segredo para a aplicação por meio da API de gerenciamento de segredo.

Passo 4 A aplicação obtém o segredo de texto não criptografado descriptografado e o usa para acessar o banco de dados.

----Fim

APIs de criação e consulta de segredos

Você pode chamar as seguintes APIs para criar segredos, salvar seu conteúdo e consultar informações de segredos.

API	Descrição
Criação de um segredo	Essa API é usada para criar um segredo e armazenar o valor de segredo na versão de segredo inicial.
Consulta de um segredo	Essa API é usada para consultar um segredo.

Criação e consulta de segredos por meio de APIs

1. Prepare informações básicas de autenticação.
   - **ACCESS_KEY**: chave de acesso da conta de Huawei
   - **SECRET_ACCESS_KEY**: chave de acesso de segredo da conta de Huawei
   - **PROJECT_ID**: ID do projeto de um site da Huawei Cloud. Para obter detalhes, consulte **Projeto**.
   - **CSMS_ENDPOINT**: ponto de extremidade para acessar o CSMS. Para obter detalhes, consulte **Pontos de extremidade**.

- Haverá riscos de segurança se a AK/SK usada para autenticação for gravada diretamente no código. Criptografe a AK/SK no arquivo de configuração ou nas variáveis de ambiente para armazenamento.
- Neste exemplo, a AK/SK armazenada nas variáveis de ambiente é usada para autenticação de identidade. Configure as variáveis de ambiente **HUAWEICLOUD_SDK_AK** e **HUAWEICLOUD_SDK_SK** primeiro no ambiente local.

2. Crie e consulte informações de segredos.

   Nome de segredo: **secretName**

   Valor de segredo: **secretString**

   Valor da versão de segredo: **LATEST_SECRET**

   Versão de segredo: **versionId**

```
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.csms.v1.CsmsClient;
import com.huaweicloud.sdk.csms.v1.model.CreateSecretRequest;
import com.huaweicloud.sdk.csms.v1.model.CreateSecretRequestBody;
import com.huaweicloud.sdk.csms.v1.model.CreateSecretResponse;
import com.huaweicloud.sdk.csms.v1.model.ShowSecretVersionRequest;
import com.huaweicloud.sdk.csms.v1.model.ShowSecretVersionResponse;

public class CsmsCreateSecretExample {
    /**
     * Basic authentication information:
     * - ACCESS_KEY: Access key of the Huawei account
     * - SECRET_ACCESS_KEY: Secret access key of the Huawei account
     * - PROJECT_ID: Huawei Cloud project ID. For details, see https://
support.huaweicloud.com/intl/pt-br/productdesc-iam/iam_01_0023.html
* - CSMS_ENDPOINT: endpoint address for accessing CSMS. For details, see https://
support.huaweicloud.com/intl/pt-br/api-dew/dew_02_0052.html.
     * - There will be security risks if the AK/SK used for authentication is
directly written into code. Encrypt the AK/SK in the configuration file or
environment variables for storage.
     * - In this example, the AK/SK stored in the environment variables are used
for identity authentication. Configure the environment variables
HUAWEICLOUD_SDK_AK and HUAWEICLOUD_SDK_SK in the local environment first.
     */
    private static final String ACCESS_KEY = System.getenv("HUAWEICLOUD_SDK_AK");
    private static final String SECRET_ACCESS_KEY =
System.getenv("HUAWEICLOUD_SDK_SK");
    private static final String PROJECT_ID = "<ProjectID>";
    private static final String CSMS_ENDPOINT = "<CsmsEndpoint>";

    //Version ID used to query the latest secret version details
    private static final String LATEST_SECRET = "latest";

    public static void main(String[] args) {
        String secretName = args[0];
        String secretString = args[1];

        //Create a secret.
        createSecret(secretName, secretString);

        //Query the content of the new secret based on the secret version latest
or v1.
        ShowSecretVersionResponse latestVersion = showSecretVersion(secretName,
LATEST_SECRET);
        ShowSecretVersionResponse firstVersion = showSecretVersion(secretName,
"v1");

        assert latestVersion.equals(firstVersion);
        assert
latestVersion.getVersion().getSecretString().equalsIgnoreCase(secretString);
    }
```



```
    /**
     * Create a secret.
     * @param secretName
     * @param secretString
     */
    private static void createSecret(String secretName, String secretString) {
        CreateSecretRequest secret = new CreateSecretRequest().withBody(
                new
CreateSecretRequestBody().withName(secretName).withSecretString(secretString));

        CsmsClient csmsClient = getCsmsClient();

        CreateSecretResponse createdSecret = csmsClient.createSecret(secret);

        System.out.printf("Created secret success, secret detail:%s",
createdSecret);
    }
    /**
     * Query secret version details based on the secret version ID.
     * @param secretName
     * @param versionId
     * @return
     */
    private static ShowSecretVersionResponse showSecretVersion(String secretName,
String versionId) {
        ShowSecretVersionRequest showSecretVersionRequest = new
ShowSecretVersionRequest().withSecretName(secretName)
                .withVersionId(versionId);

        CsmsClient csmsClient = getCsmsClient();

        ShowSecretVersionResponse version =
csmsClient.showSecretVersion(showSecretVersionRequest);

        System.out.printf("Query secret success. version id:%s",
version.getVersion().getVersionMetadata().getId());

        return version;
    }

    /**
     * Obtain the CSMS client.
     * @return
     */
    private static CsmsClient getCsmsClient() {
        BasicCredentials auth = new BasicCredentials()
                .withAk(ACCESS_KEY)
                .withSk(SECRET_ACCESS_KEY)
                .withProjectId(PROJECT_ID);

        return
CsmsClient.newBuilder().withCredential(auth).withEndpoint(CSMS_ENDPOINT).build();
    }
}
```

Obtenção da conta do banco de dados por meio de uma aplicação

1. Obtenha a declaração de dependência do SDK do CSMS.

   Exemplo:

   ```
   <dependency>
               <groupId>mysql</groupId>
               <artifactId>mysql-connector-java</artifactId>
               <version>XXX</version>
           </dependency>
           <dependency>
               <groupId>com.google.code.gson</groupId>
               <artifactId>gson</artifactId>
   ```



```
            <version>2.8.9</version>
        </dependency>
        <dependency>
            <groupId>com.huaweicloud.sdk</groupId>
            <artifactId>huaweicloud-sdk-csms</artifactId>
            <version>3.0.79</version>
        </dependency>
```

2. Estabeleça uma conexão com o banco de dados e obtenha a conta.

   Exemplo:

```
import com.google.gson.Gson;
import com.google.gson.JsonObject;
import com.huaweicloud.sdk.csms.v1.model.ShowSecretVersionRequest;
import com.huaweicloud.sdk.csms.v1.model.ShowSecretVersionResponse;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

 // Obtain the specified database account based on the secret information.
    public static Connection getMySQLConnectionBySecret(String secretName,
String jdbcUrl) throws ClassNotFoundException, SQLException{
        Class.forName(MYSQL_JDBC_DRIVER);
        ShowSecretVersionResponse latestVersionValue =
getCsmsClient().showSecretVersion(new
ShowSecretVersionRequest().withSecretName(secretName).withVersionId("latest"))
;
        String secretString =
latestVersionValue.getVersion().getSecretString();
        JsonObject jsonObject = new Gson().fromJson(secretString,
JsonObject.class);
        return DriverManager.getConnection(jdbcUrl,
jsonObject.get("username").getAsString(),
jsonObject.get("password").getAsString());
    }
```

2.2 Uso do CSMS para evitar vazamento de AK e SK

O CSMS é um serviço de hospedagem de credenciais seguro, confiável e fácil de usar. Os usuários ou aplicações podem usar o CSMS para criar, recuperar, atualizar e excluir credenciais de maneira unificada durante todo o ciclo de vida das credenciais. O CSMS pode ajudá-lo a eliminar os riscos incorridos pela codificação rígida, configuração de texto simples e abuso de permissão.

Cenário

Os segredos de aplicação são armazenados e podem ser acessados temporariamente para evitar vazamentos de AK e SK.

Como funciona

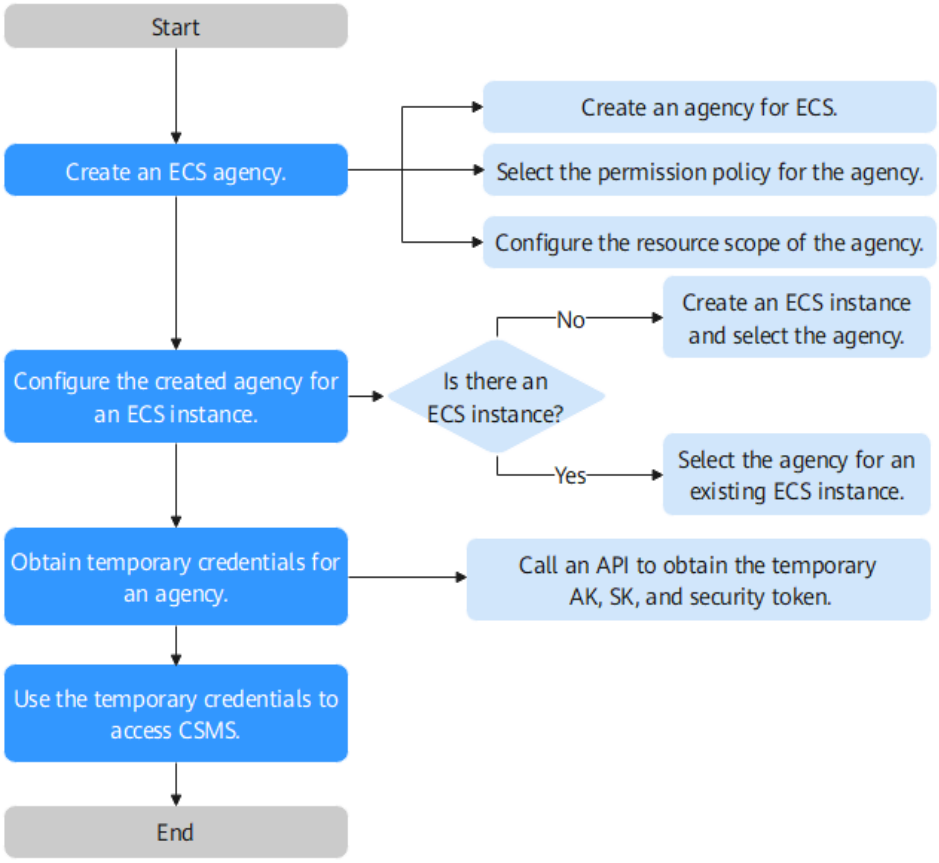
Você pode usar o Identity and Access Management (IAM) para obter chaves de acesso temporárias para o Elastic Cloud Server (ECS) para proteger AKs e SKs.

Os segredos de acesso podem ser classificados em segredos permanentes e segredos temporários com base em seus períodos de validade. Os segredos de acesso permanente incluem nomes de usuário e senhas. As chaves de acesso temporárias têm um período de validade mais curto, são atualizadas com frequência e, portanto, são mais seguras. Você pode atribuir uma agência do IAM a uma instância do ECS para que as aplicações na instância do ECS possam usar o AK, a SK e o token de segurança temporários para acessar o CSMS. As

chaves de acesso temporárias são obtidas dinamicamente sempre que são necessárias. Elas também podem ser armazenadas em cache na memória e atualizadas periodicamente.

Fluxo do processo

Figura 2-2 Processo de configuração da agência do ECS



Restrição

Somente o administrador ou um usuário do IAM com a permissão do ECS pode configurar uma agência para uma instância do ECS.

Procedimento

Passo 1 Crie uma agência do ECS no IAM.

1. **Faça logon no console de gerenciamento**.
2. Clique em ☰ à esquerda da página e escolha **Management & Governance** > **Identity and Access Management**. A página **Users** é exibida.
3. No painel de navegação, escolha **Agencies**.
4. Clique em **Create Agency** no canto superior direito.
5. Configure os parâmetros na caixa de diálogo **Create Agency**. Para obter mais informações, consulte **Parâmetros da agência**.

Figura 2-3 Criação de uma agência

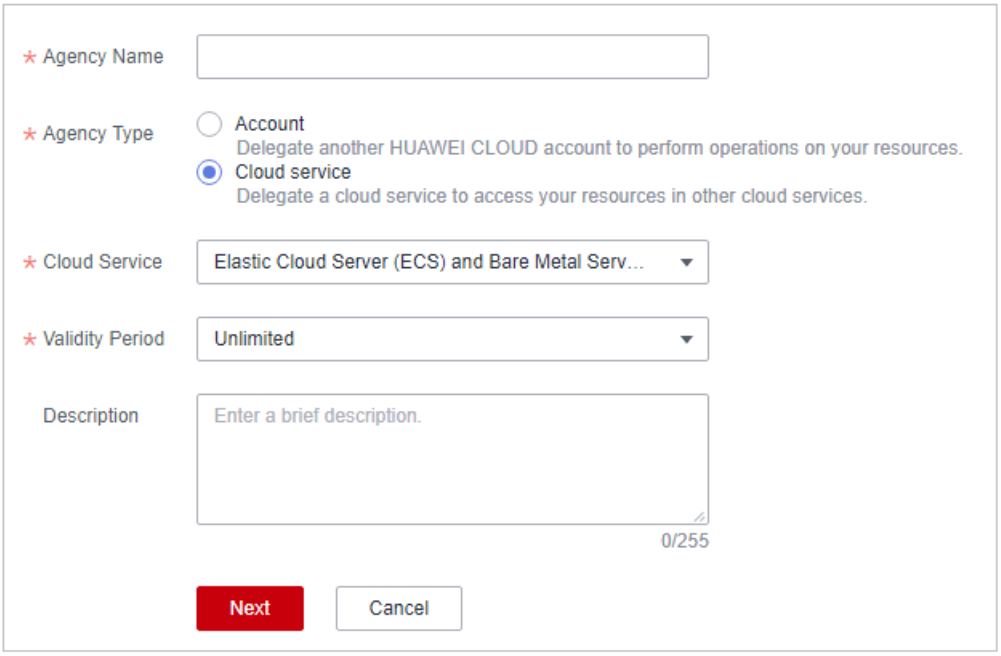


Tabela 2-1 Parâmetros da agência

Nome do parâmetro	Descrição
Agency Name	Insira um nome de agência. Exemplo: **ECS_TO_CSMS**
Agency Type	Selecione **Cloud service**.
Cloud Service	Selecione **Elastic Cloud Server (ECS) and Bare Metal Server (BMS)**.
Validity Period	Selecione uma duração. O valor pode ser **Unlimited**, **1 day** ou **Custom**.
Description	(Opcional) Insira a descrição da agência.

6. Clique em **Next** para acessar a página de autorização.
7. Clique em **Create Policy** no canto superior direito. Se você já tiver uma política, pule esta etapa.
   a. Configure parâmetros de política. Para obter mais informações, consulte **Parâmetros de política**.

Figura 2-4 Criação de uma política

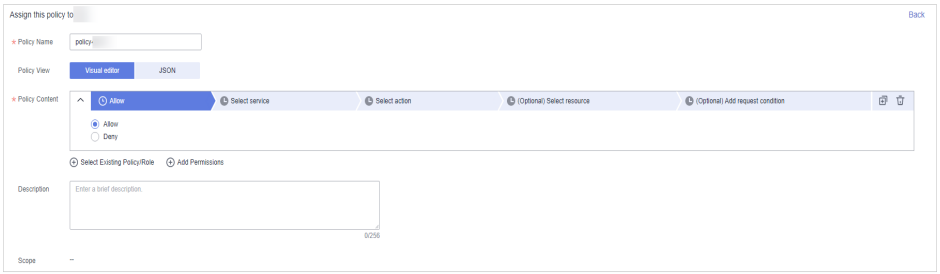


Tabela 2-2 Parâmetros de política

Nome do parâmetro	Descrição
Policy Name	Insira um nome de política.
Policy View	Selecione **Visual editor**.
Policy Content	■ **Allow**: selecione **Allow**. ■ **Select service**: selecione **Cloud Secret Management Service (CSMS)**. ■ **Select action**: selecione as permissões de leitura e gravação conforme necessário. ■ **(Optional) Select resource**: selecione o escopo dos recursos. ○ **Specific**: acesse segredos específicos. **NOTA** Você pode selecionar **Specify resource path** e, em seguida, clique em **Add Resource Path** para especificar um segredo acessível. ○ **All**: acesse todos os segredos. ■ **(Optional) Add request condition**: clique em **Add Request Condition**, selecione uma chave de condição e um operador e insira os valores conforme necessário.
Description	(Opcional) Insira a descrição da política.

8. Selecione uma política para a agência. Clique em **Next**.
9. Selecione um escopo e clique em **OK**.
   - **All resources**: os usuários do IAM poderão usar todos os recursos, incluindo aqueles em projetos empresariais, projetos específicos de região e serviços globais em sua conta com base nas permissões atribuídas.
   - **Enterprise projects**: as permissões selecionadas serão aplicadas aos recursos nos projetos empresariais selecionados.
   - **Region-specific projects**: as permissões selecionadas serão aplicadas aos recursos nos projetos específicos da região que você selecionar.

Passo 2 Atribua uma agência (por exemplo, **ECS_TO_CSMS**) a uma instância do ECS.

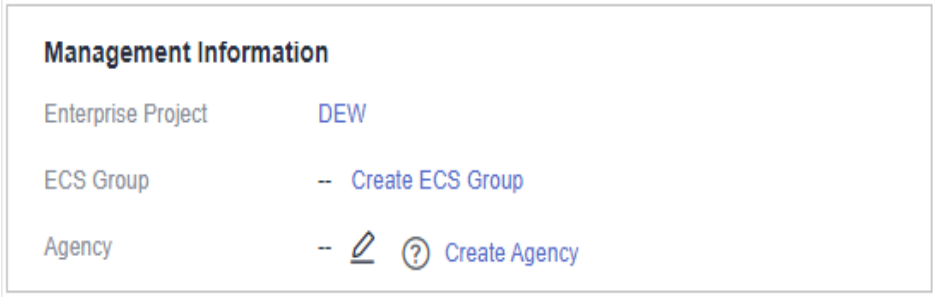
- Para criar uma instância do ECS, execute as operações descritas em **Criação de um ECS**. Na **Etapa 3: configurar as definições avançadas**, selecione a nova agência (por exemplo, **ECS_TO_CSMS**).
- Para usar uma instância do ECS existente, execute as seguintes etapas:

  a. Clique em ☰ à esquerda da página e escolha **Management & Governance** > **Identity and Access Management**. Acesse a página do ECS.

  b. Clique no nome de uma instância do ECS para acessar a página **Summary**.

  c. Na área **Management Information**, clique em ✎ e selecione uma agência (por exemplo, **ECS_TO_CSMS**).

  Figura 2-5 Seleção de uma agência

Passo 3 Em uma aplicação em execução na instância do ECS, chame uma API para obter os segredos da agência temporária, incluindo o AK, a SK e o token de segurança temporários, para acessar o CSMS.

1. Obtenha o AK e a SK temporários (no diretório **Security Key**). Para obter detalhes, consulte **Obtenção de metadados**.
   - URI

     /openstack/latest/securitykey
   - Método

     Solicitação GET
   - Os seguintes dados são retornados:

```
{
      "credential":{
          "access": "LDHZK30XXXXXXXXXXXV",
          "secret":"gyqcdzVXXXXXXXXXXXXXXXXXXXXXXXXXMl6",
          "securitytoken": "El9FI2C65qXXXXXXXXXXXXXXXXXXXXXnkcaoV",
          "expires_at": "2022-07-14T12:09:24.147000Z"
                  }
}
```

   NOTA

   - Extraia os valores de **access**, **secret** e **securitytoken** para acessar o CSMS.
   - O ECS rotaciona automaticamente os segredos temporários para garantir que eles sejam seguros e válidos.
2. Use o AK, a SK e o token de segurança temporários para acessar o CSMS.
   - Um exemplo da lista de segredos é o seguinte. Para obter detalhes, consulte **APIs de gerenciamento de segredos**.
   - Prepare informações básicas de autenticação.

- ACCESS_KEY: chave de acesso da conta da Huawei. Para obter detalhes, consulte **Como obter uma chave de acesso (AK/SK)?**
- SECRET_ACCESS_KEY: chave de acesso de segredo da conta da Huawei. Para obter detalhes, consulte **Como obter uma chave de acesso (AK/SK)?**
- Haverá riscos de segurança se a AK/SK usada para autenticação for gravada diretamente no código. Criptografe a AK/SK no arquivo de configuração ou nas variáveis de ambiente para armazenamento.
- Neste exemplo, a AK/SK armazenada nas variáveis de ambiente é usada para autenticação de identidade. Configure as variáveis de ambiente **HUAWEICLOUD_SDK_AK** e **HUAWEICLOUD_SDK_SK** primeiro no ambiente local.

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.csms.v1.region.CsmsRegion;
import com.huaweicloud.sdk.csms.v1.*;
import com.huaweicloud.sdk.csms.v1.model.*;

public class ListSecretsSolution {
    public static void main(String[] args) {
     * Basic authentication information:
     * - ACCESS_KEY: Access key of the Huawei account
     * - SECRET_ACCESS_KEY: Secret access key of the Huawei account
     * - PROJECT_ID: Huawei Cloud project ID. For details, see https://
support.huaweicloud.com/intl/pt-br/productdesc-iam/iam_01_0023.html
* - CSMS_ENDPOINT: endpoint address for accessing CSMS. For details, see
https://support.huaweicloud.com/intl/pt-br/api-dew/dew_02_0052.html.
     * - There will be security risks if the AK/SK used for
authentication is directly written into code. Encrypt the AK/SK in the
configuration file or environment variables for storage.
     * - In this example, the AK/SK stored in the environment variables
are used for identity authentication. Configure the environment
variables HUAWEICLOUD_SDK_AK and HUAWEICLOUD_SDK_SK in the local
environment first.
     */
    private static final String ACCESS_KEY =
System.getenv("HUAWEICLOUD_SDK_AK");
    private static final String SECRET_ACCESS_KEY =
System.getenv("HUAWEICLOUD_SDK_SK");
    String securitytoken = "<YOUR SecurityToken>";

        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");

        ICredential auth = new BasicCredentials()
                .withAk(ak)
                .withSk(sk)
                .withSecurityToken(securitytoken);

         CsmsClient client = CsmsClient.newBuilder()
                 .withCredential(auth)
                 .withRegion(CsmsRegion.valueOf("cn-north-1"))
                 .build();
         ListSecretsRequest request = new ListSecretsRequest();
         try {
             ListSecretsResponse response = client.listSecrets(request);
             System.out.println(response.toString());
         } catch (ConnectionException e) {
             e.getMessage();
         } catch (RequestTimeoutException e) {
             e.getMessage();
```



```
            } catch (ServiceResponseException e) {
                e.getMessage();
                System.out.println(e.getHttpStatusCode());
                System.out.println(e.getErrorCode());
                System.out.println(e.getErrorMsg());
            }
        }
}
```

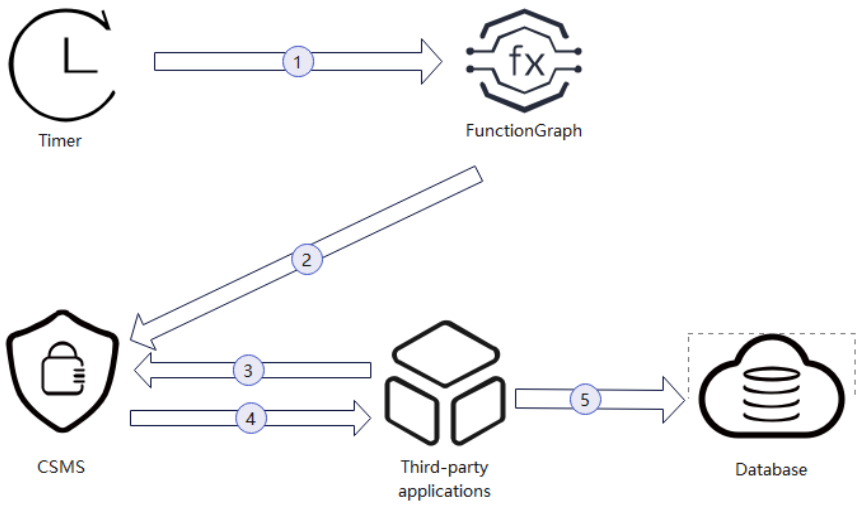
----Fim

2.3 Uso do CSMS para rotacionar automaticamente as senhas de segurança

Esta seção descreve como usar o FunctionGraph e o CSMS para gerar e rotacionar senhas seguras e fortes periodicamente, de modo que senhas seguras e em conformidade possam ser geradas, hospedadas e rotacionadas automaticamente.

Processo

Figura 2-6 Rotação de senha



O processo é o seguinte:

1. Quando um temporizador expira, um evento de acionamento agendado é publicado.
2. Depois de receber o evento, o FunctionGraph substitui o espaço reservado no modelo de segredo por uma nova senha aleatória e armazena a senha no segredo, que é considerada uma nova versão do segredo.
3. As aplicações chamam periodicamente APIs ou SDKs para obter a versão de segredo mais recente.
4. O CSMS recupera e descriptografa o texto cifrado de segredo e retorna com segurança as informações armazenadas no segredo para a aplicação por meio da API de gerenciamento de segredo.

5. Depois de receber o segredo descriptografado, as aplicações usam a nova senha para acesso futuro usando-a para atualizar o objeto de destino (como o banco de dados ou o servidor).

Restrições

- O CSMS está disponível na região.
- O FunctionGraph está disponível na região.

Criação de uma agência

Passo 1 **Faça logon no console de gerenciamento**.

Passo 2 Clique em ☰ à esquerda e escolha **Management & Governance** > **Identity and Access Management** para acessar a página **Users**.

Passo 3 No painel de navegação à esquerda, escolha **Agencies**.

Passo 4 Clique em **Create Agency** e configure os parâmetros conforme mostrado em **Figura 2-7**. **Tabela 2-3** descreve os parâmetros.

Figura 2-7 Criação de uma agência

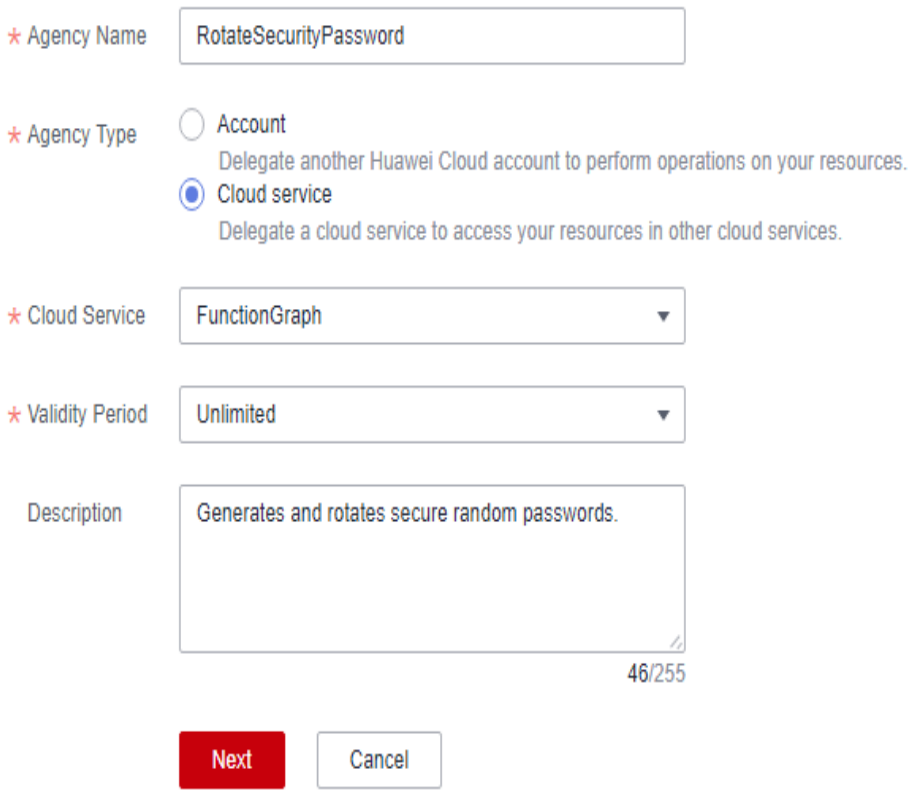


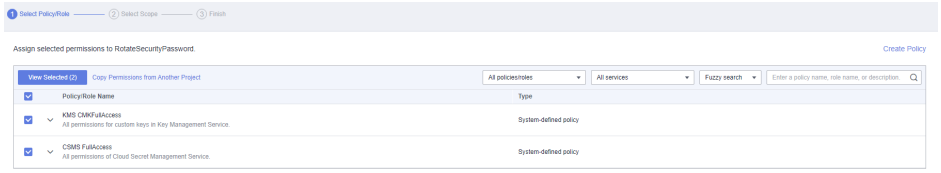
Tabela 2-3 Parâmetros da agência

Parâmetro	Descrição
Agency Name	Defina este parâmetro conforme necessário.
Agency Type	Selecione **Cloud service**.
Cloud Service	Escolha **FunctionGraph**.
Validity Period	Defina este parâmetro com base no cenário de aplicação da função. Se a função precisar ser executada por um longo tempo, escolha **Unlimited**.
Description	Defina este parâmetro conforme necessário.

Passo 5 Clique em **Next**.

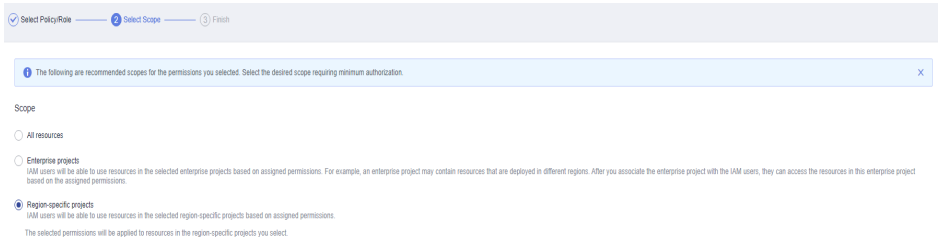
Passo 6 Selecione **CSMS FullAccess** e **KMS CMKFullAccess**.

Figura 2-8 Seleção de permissões



Passo 7 Clique em **Next** e selecione um escopo com base em seus requisitos de serviço.

Figura 2-9 Seleção de um escopo



Passo 8 Clique em **OK**.

----Fim

Criação de uma função de rotação de senha

Passo 1 **Faça logon no console de gerenciamento**.

Passo 2 Clique em ☰ à esquerda e escolha **Compute** > **FunctionGraph**.

Passo 3 Clique em **Create Function** no canto superior direito e configure os parâmetros conforme mostrado em **Figura 2-10**. **Tabela 2-4** descreve os parâmetros.

Figura 2-10 Criação de uma função

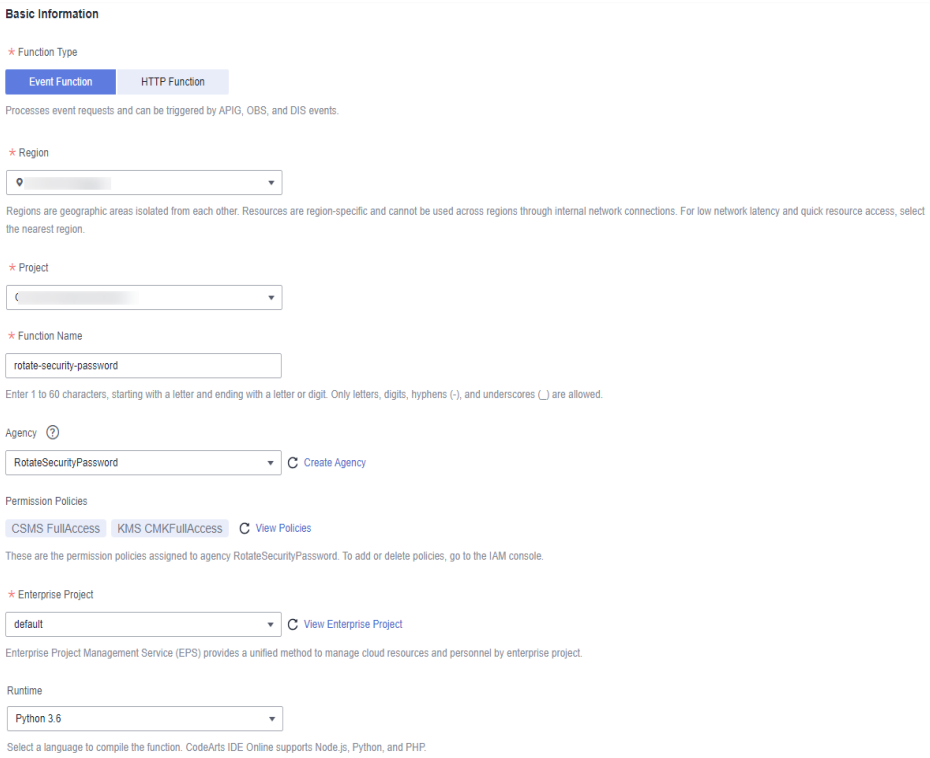


Tabela 2-4 Parâmetros básicos

Parâmetro	Descrição
Region	Selecione a região onde a função está implementada.
Project	Selecione o projeto no qual a função será implementada.
Function Name	Insira um nome de função personalizado.
Agency	Digite o nome definido em **Criação de uma agência**.
Enterprise Project	Se você ativou projetos empresariais, selecione um para adicionar uma função a ele. Se você não ativou projetos empresariais, esse parâmetro não será exibido. Pule este passo. Para obter detalhes sobre como ativar um projeto empresarial, consulte **Ativação da central empresarial**.
Runtime	Selecione uma linguagem para escrever a função. Atualmente, o Python é suportado. **NOTA** Apenas Python 3.6, 3.9 e 3.10 são suportados.

Passo 4 Clique em **Create Function**.

Passo 5 Escolha a guia **Configuration**. No painel de navegação à esquerda, escolha **Environment Variables**. Clique em **Add** e adicione variáveis de ambiente na linha de configuração de variáveis. **Tabela 2-4** descreve os parâmetros. Em seguida, clique em **Save**.

Tabela 2-5 Variáveis ambientais

Parâmetro	Descrição	Exemplo
region	Nome do projeto que é baseado na região, por exemplo, se a região é CN North-Beijing4, o nome do projeto deve ser **cn-north-4**. Para obter sua região, clique em seu nome de usuário no canto superior direito da página e escolha **My Credentials** na lista suspensa.	cn-north-4
secret_name	Nome do segredo a ser rotacionado **NOTA** Um segredo deve ter sido criado. Para obter detalhes, consulte **Criação de um segredo**.	rds-functionGraph-rotate
secret_content	O modelo de segredo que é especificado usando chaves ({}), por exemplo, o modelo de segredo é **{"password":"password_placeholder"}**. Nesse caso, **password_placeholder** é o espaço reservado, que será substituído por uma senha segura após a execução da função. O novo conteúdo será salvo em segredo após a substituição. **NOTA** Se existirem vários espaços reservados em um modelo, mais de uma senha será gerada e substituída em sequência.	{"password":"password_placeholder"}
password_length	Comprimento da senha. O valor varia de **8** a **128**. O valor padrão é **16**.	16
password_format	Formato de senha. O valor padrão é **2**. Os valores possíveis são os seguintes: 1. Dígitos e letras 2. Dígitos, letras e caracteres especiais (~!@#%^*-_=+?) 3. Apenas dígitos 4. Apenas letras	2

Passo 6 Escolha a guia **Code**, adicione a seguinte função de rotação de senha à janela de edição e clique em **Deploy**.

```
# -*- coding:utf-8 -*-
import json
import secrets
import string
import requests
import inspect

def handler (event, context):
    global secret_content
    global password_length
    global password_format
```



```
    global kms_endpoint
    global region
    global secret_name
    global headers
    region = context.getUserData('region')
    secret_name = context.getUserData('secret_name')
    password_length = 16 if context.getUserData('password_length') is None else
int(context.getUserData('password_length'))
    password_format = 2 if context.getUserData('password_format') is None else
int(context.getUserData('password_format'))
    secret_content = context.getUserData('secret_content')
    headers = {
        'Content-Type': 'application/json',
        'x-Auth-Token': context.getToken()
    }
    try:
        new_content = replace_old_content(secret_content)
        # check region, if pass, return kms endpoint
        kms_endpoint = check_region(region)
        return update(context, new_content)
    except Exception as e:
        print("ERROR: %s" % e)
        return 'FAILED'

# replace "password_placeholder" in secret_content by new password
def replace_old_content(content):
    while content.find("password_placeholder") != -1:
        password = generate_password()
        while password.find("password_placeholder") != -1:
            password = generate_password()
        content = content.replace("password_placeholder", password, 1)
    return content

def generate_password():
    special_chars = "~!@#%^*-_=+?"
    # password format(default is 2):
    # 1.support letters and digits; 2.support letters, digits and special chars(~!
@#%^*-_=+?);
    # 3.only support digits; 4.only support letters
    format_mapping = {
        1: string.ascii_letters + string.digits,
        2: string.ascii_letters + string.digits + special_chars,
        3: string.digits,
        4: string.ascii_letters
    }
    if password_length < 8 or password_length > 128:
        raise Exception("invalid password_length: %s, the password length range
must be between 8-128." % password_length)
    try:
        support_chars = format_mapping[password_format]
        password = ''.join([secrets.choice(support_chars) for _ in
range(password_length)])
        return password
    except:
        raise Exception("invalid password_format: %s." % password_format)

def check_region(region):
    endpoint_mapping = {
        'cn-north-1': 'cn-north-1.myhuaweicloud.com',
        'cn-north-2': 'cn-north-2.myhuaweicloud.com',
        'cn-north-4': 'cn-north-4.myhuaweicloud.com',
        'cn-north-7': 'cn-north-7.myhuaweicloud.com',
        'cn-north-9': 'cn-north-9.myhuaweicloud.com',
        'cn-east-2': 'cn-east-2.myhuaweicloud.com',
        'cn-east-3': 'cn-east-3.myhuaweicloud.com',
        'cn-south-1': 'cn-south-1.myhuaweicloud.com',
        'cn-south-2': 'cn-south-2.myhuaweicloud.com',
        'cn-southwest-2': 'cn-southwest-2.myhuaweicloud.com',
        'ap-southeast-1': 'ap-southeast-1.myhuaweicloud.com',
```



```
        'ap-southeast-2': 'ap-southeast-2.myhuaweicloud.com',
        'ap-southeast-3': 'ap-southeast-3.myhuaweicloud.com',
        'af-south-1': 'af-south-1.myhuaweicloud.com',
        'la-north-2': 'la-north-2.myhuaweicloud.com',
        'la-south-2': 'la-south-2.myhuaweicloud.com',
        'na-mexico-1': 'na-mexico-1.myhuaweicloud.com',
        'sa-brazil-1': 'sa-brazil-1.myhuaweicloud.com'
    }
    try:
        endpoint = endpoint_mapping[region]
        kms_endpoint = '%s.%s' % ('kms', endpoint)
        return kms_endpoint
    except:
        raise Exception("invalid region: %s" % region)

def check_csms_resp(resp):
    if resp.status_code in (200, 201, 204):
        return
    caller_function_name = inspect.stack()[1].function
    json_resp = json.loads(resp.text)
    if 'error_msg' in json_resp:
        error_message = 'function:%s , reason: %s' % (
            caller_function_name, json_resp['error_msg'])
        raise Exception(error_message)
    error_message = 'function:%s , reason: %s' % (
        caller_function_name, resp.text)
    raise Exception(error_message)

def update(context, new_content):
    project_id = context.getProjectID()
    url = 'https://%s/v1/%s/secrets/%s/versions' % (kms_endpoint, project_id,
secret_name)
    payload = {'secret_string': new_content}
    payload = json.dumps(payload)
    resp = requests.post(url, headers=headers, data=payload)
    check_csms_resp(resp)
    return 'SUCCESS'
```

----Fim

Depuração

Depure o FunctionGraph criado. Para obter detalhes, consulte **Depuração on-line**.

Criação de um acionador

Crie um acionador. Para obter detalhes, consulte **Uso de um acionador de temporizador**.

Visualização de um segredo

Passo 1 **Faça logon no console de gerenciamento**.

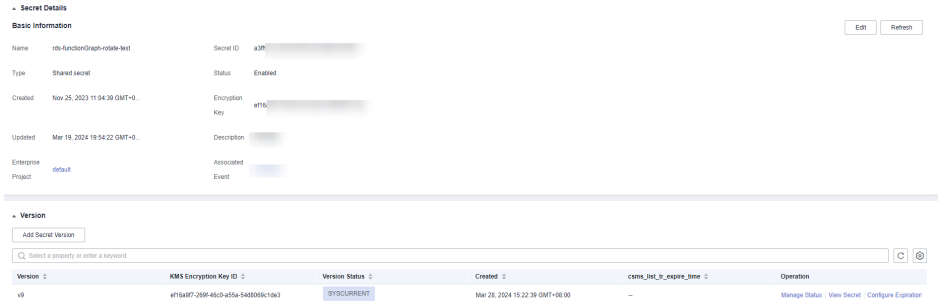
Passo 2 Clique em ☰ à esquerda, escolha **Security & Compliance** > **Data Encryption Workshop**, a página de gerenciamento de chaves é exibida.

Passo 3 No painel de navegação, escolha **Cloud Secret Management Service**.

Passo 4 Pesquise o segredo criado em **Criação de uma função de rotação de senha**.

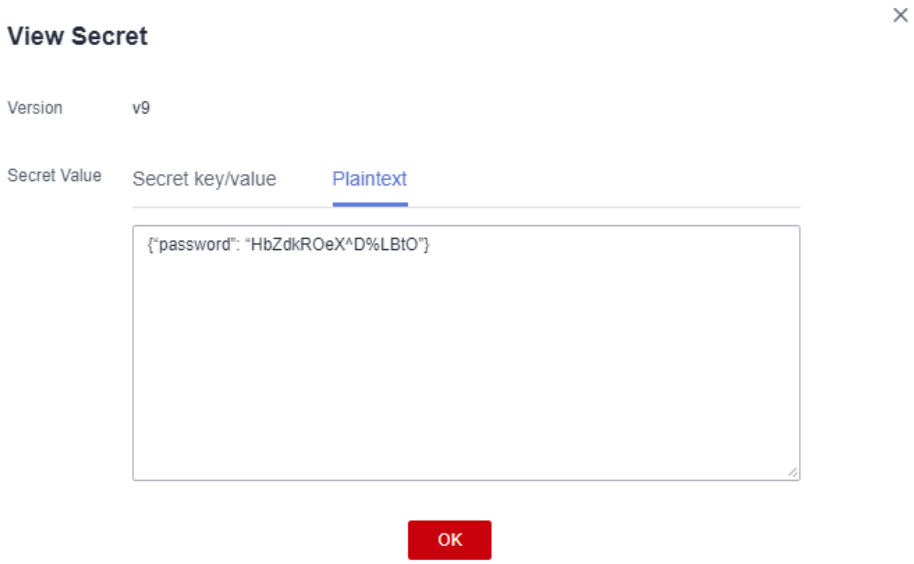
Passo 5 Clique no segredo para ver seus detalhes, incluindo versões atuais e históricas.

Figura 2-11 Visualização de detalhes de segredos



Passo 6 Na **Version**, localize o segredo e clique em **View Secret** na coluna **Operation** para exibir a senha.

Figura 2-12 Visualização de um valor de segredo



----Fim

2.4 Rotação de segredos

2.4.1 Visão geral

Se você não tiver atualizado os segredos por um longo período, informações importantes (como senhas importantes, tokens, certificados, chaves SSH e chaves de API) protegidas por esses segredos estarão expostas a riscos de vazamento. A rotação periódica de segredos melhora a segurança das informações de texto não criptografado protegidas.

A HUAWEI CLOUD oferece duas políticas de rotação de segredo:

- **Rotação de um segredo para um usuário**
- **Rotação de um segredo para dois usuários**

Você pode selecionar uma política de rotação conforme necessário.

Procedimento

1. O administrador adiciona uma versão de segredo e atualiza seu conteúdo no **console** ou por meio de uma **API**.
2. A aplicação chama uma API do CSMS para obter a versão de segredo mais recente ou o segredo de um status de versão especificado e, em seguida, rotaciona o segredo.
3. Repita regularmente os passos **1** e **2** para rotacionar segredos.

2.4.2 Rotação de um segredo para um usuário

Visão geral

Você pode atualizar as informações de um usuário em um segredo.

Esta é a política de rotação de segredo mais usada.

Exemplo:

- Para o acesso ao banco de dados, uma conexão de banco de dados não é excluída durante a rotação de segredos. Após a rotação, novas conexões usam os novos segredos.
- Um usuário pode criar uma conta, por exemplo, usando um endereço de e-mail como nome de usuário. Geralmente, os usuários podem alterar senhas conforme necessário, mas não podem criar outros usuários ou alterar nomes de usuário.
- Os usuários temporários são criados no momento em que são necessários.
- As senhas são inseridas pelos usuários, não recuperadas do CSMS pelas aplicações. Esses usuários não precisam alterar seus nomes de usuário ou senhas.

Restrições

Verifique se sua conta tem a permissão KMS Administrator ou KMS CMKFullAccess. Para obter detalhes, consulte **Gerenciamento de permissões do DEW**.

APIs de rotação de segredos

Você pode chamar as seguintes APIs para rotacionar segredos localmente.

API	Descrição
Criação de uma versão de segredo	Criar uma versão de segredo.
Consulta da versão e do valor do segredo	Consultar as informações sobre uma versão de segredo especificada e o valor de segredo de texto não criptografado na versão.

Exemplo de rotação de segredo de conta única

1. Crie um segredo no console da HUAWEI CLOUD. Para obter detalhes, consulte **Criação de um segredo**.
2. Prepare informações básicas de autenticação.
   - **ACCESS_KEY**: chave de acesso da conta de Huawei

- **SECRET_ACCESS_KEY**: chave de acesso de segredo da conta de Huawei
- **PROJECT_ID**: ID do projeto de um site da HUAWEI CLOUD. Para obter detalhes, consulte **Projeto**.
- **CSMS_ENDPOINT**: ponto de extremidade para acessar o CSMS. Para obter detalhes, consulte **Pontos de extremidade**.
- Haverá riscos de segurança se a AK/SK usada para autenticação for gravada diretamente no código. Criptografe a AK/SK no arquivo de configuração ou nas variáveis de ambiente para armazenamento.
- Neste exemplo, a AK/SK armazenada nas variáveis de ambiente é usada para autenticação de identidade. Configure as variáveis de ambiente **HUAWEICLOUD_SDK_AK** e **HUAWEICLOUD_SDK_SK** primeiro no ambiente local.

3. Rotação do segredo de um usuário.

   No código de exemplo,

   - **secretName** indica o nome do segredo criado no console da Huawei Cloud.
   - **secretString** indica o valor salvo no segredo criado no console da Huawei Cloud.
   - **versionId** indica o ID de segredo gerado automaticamente depois que um segredo é criado no console da Huawei Cloud.
   - **LATEST_VERSION** indica a versão de segredo.

```
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.csms.v1.CsmsClient;
import com.huaweicloud.sdk.csms.v1.model.CreateSecretVersionRequest;
import com.huaweicloud.sdk.csms.v1.model.CreateSecretVersionRequestBody;
import com.huaweicloud.sdk.csms.v1.model.CreateSecretVersionResponse;
import com.huaweicloud.sdk.csms.v1.model.ShowSecretVersionRequest;
import com.huaweicloud.sdk.csms.v1.model.ShowSecretVersionResponse;

public class CsmsSingleAccountExample {
    /**
     * Basic authentication information:
     * - ACCESS_KEY: Access key of the Huawei account
     * - SECRET_ACCESS_KEY: Secret access key of the Huawei account
     * - PROJECT_ID: Huawei Cloud project ID. For details, see https://
support.huaweicloud.com/intl/pt-br/productdesc-iam/iam_01_0023.html
* - CSMS_ENDPOINT: endpoint address for accessing CSMS. For details, see
https://support.huaweicloud.com/intl/pt-br/api-dew/dew_02_0052.html.
     * - There will be security risks if the AK/SK used for
authentication is directly written into code. Encrypt the AK/SK in the
configuration file or environment variables for storage.
     * - In this example, the AK/SK stored in the environment variables
are used for identity authentication. Configure the environment
variables HUAWEICLOUD_SDK_AK and HUAWEICLOUD_SDK_SK in the local
environment first.
     */
    private static final String ACCESS_KEY =
System.getenv("HUAWEICLOUD_SDK_AK");
    private static final String SECRET_ACCESS_KEY =
System.getenv("HUAWEICLOUD_SDK_SK");
    private static final String PROJECT_ID = "<ProjectID>";
    private static final String CSMS_ENDPOINT = "<CsmsEndpoint>";
    //Version ID used to query the latest secret version details.
    private static final String LATEST_VERSION = "latest";
    public static void main(String[] args) {
        String secretName = args[0];
        String secretString = args[1];
        singleAccountRotation(secretName, secretString);
    }

    /**
     * Example: secret rotation for a single account
```



```
     *
     * @param secretName   Secret name
     * @param secretString Secret content
     */
    private static void singleAccountRotation(String secretName, String
secretString) {
        //Host the new secret in CSMS.
        createNewSecretVersion(secretName, secretString);
        //Query the secret of the latest version based on the secret
version latest.
        ShowSecretVersionResponse secretResponseByLatest =
showSecretVersionDetail(secretName, LATEST_VERSION);
        assert
secretResponseByLatest.getVersion().getSecretString().equals(secretString
);
    }

    /**
     * Example of creating a secret
     * If a secret version is added without version status, the program
points the SYSCURRENT version status to the version by default.
     *
     * @param secretName   Secret name
     * @param secretString Secret content
     * @return
     */
    private static CreateSecretVersionResponse
createNewSecretVersion(String secretName, String secretString) {
        CsmsClient csmsClient = getCsmsClient();

        CreateSecretVersionRequest createSecretVersionRequest = new
CreateSecretVersionRequest()
                .withSecretName(secretName)
                .withBody(new
CreateSecretVersionRequestBody().withSecretString(secretString));

        CreateSecretVersionResponse secretVersion =
csmsClient.createSecretVersion(createSecretVersionRequest);

        System.out.printf("Created new version success, version id:%s",
secretVersion.getVersionMetadata().getId());
        return secretVersion;
    }
    /**
     * Query the secret of a specified version.
     *
     * @param secretName Secret name
     * @param versionId  Secret_version_ID
     * @return
     */
    private static ShowSecretVersionResponse
showSecretVersionDetail(String secretName, String versionId) {
        ShowSecretVersionRequest showSecretVersionRequest = new
ShowSecretVersionRequest().withSecretName(secretName)
                .withVersionId(versionId);
        CsmsClient csmsClient = getCsmsClient();
        ShowSecretVersionResponse secretDetail =
csmsClient.showSecretVersion(showSecretVersionRequest);
        System.out.printf("Query latest version success. version id:%s",
secretDetail.getVersion().getVersionMetadata().getId());
        return secretDetail;
    }
    /**
     * Obtain the CSMS client.
     *
     * @return
     */
    private static CsmsClient getCsmsClient() {
        BasicCredentials auth = new BasicCredentials()
```



```
                .withAk(ACCESS_KEY)
                .withSk(SECRET_ACCESS_KEY)
                .withProjectId(PROJECT_ID);
        return
CsmsClient.newBuilder().withCredential(auth).withEndpoint(CSMS_ENDPOINT).
build();
    }
}
```

2.4.3 Rotação de um segredo para dois usuários

Visão geral

Você pode atualizar as informações de dois usuários em um segredo.

Por exemplo, se a sua aplicação exigir alta disponibilidade e você executar a rotação de usuário único, poderá não conseguir acessar a aplicação ao alterar a senha do usuário e atualizar o conteúdo de segredo. Nesse caso, você precisa usar a política de rotação de segredos para vários usuários.

Você deve ter uma conta, por exemplo, **Admin**, que tenha permissão para criar e alterar as senhas de **user1** e **user2**. Primeiro, crie um segredo, crie e salve a conta e a senha de **user1** e registre-as como **user1/password1**. Em seguida, crie **user2** adicionando a versão de segredo **v2** e salve a senha de **user2** como **user2/password2**. Se você alterar a senha de **user1**, será necessário adicionar a versão de segredo **v3** e registrá-la como **user1/password3**. Quando a versão de segredo **v3** está sendo criada, a aplicação usa a versão de segredo **v2** existente para obter informações. Quando a **v3** estiver pronta, a tag de armazenamento temporário será alterada e a aplicação poderá usar a **v3** para obter informações.

Restrições

Verifique se sua conta tem a permissão KMS Administrator ou KMS CMKFullAccess. Para obter detalhes, consulte **Gerenciamento de permissões do DEW**.

APIs de rotação de segredos

Você pode chamar as seguintes APIs para rotacionar segredos localmente.

API	Descrição
Criação de uma versão de segredo	Criar uma versão de segredo.
Consulta da versão e do valor do segredo	Consultar as informações sobre uma versão de segredo especificada e o valor de segredo de texto não criptografado na versão.
Atualização do status da versão de um segredo	Atualizar o status da versão de um segredo.
Consulta do status de uma versão de segredo	Consultar o status de uma versão de segredo especificada.

Exemplo de rotação de segredo de conta dupla

1. Crie um segredo no console da HUAWEI CLOUD como administrador. Para obter detalhes, consulte **Criação de um segredo**.
2. Prepare informações básicas de autenticação.
   - **ACCESS_KEY**: chave de acesso da conta de Huawei
   - **SECRET_ACCESS_KEY**: chave de acesso de segredo da conta de Huawei
   - **PROJECT_ID**: ID do projeto de um site da HUAWEI CLOUD. Para obter detalhes, consulte **Projeto**.
   - **CSMS_ENDPOINT**: ponto de extremidade para acessar o CSMS. Para obter detalhes, consulte **Pontos de extremidade**.
   - Haverá riscos de segurança se a AK/SK usada para autenticação for gravada diretamente no código. Criptografe a AK/SK no arquivo de configuração ou nas variáveis de ambiente para armazenamento.
   - Neste exemplo, a AK/SK armazenada nas variáveis de ambiente é usada para autenticação de identidade. Configure as variáveis de ambiente **HUAWEICLOUD_SDK_AK** e **HUAWEICLOUD_SDK_SK** primeiro no ambiente local.
3. Rotacione o segredo para dois usuários.

   No código de exemplo,
   - **secretName** indica o nome do segredo criado no console da Huawei Cloud.
   - **secretString** indica o valor salvo no segredo criado no console da Huawei Cloud.
   - **versionId** indica o ID de segredo gerado automaticamente depois que um segredo é criado no console da Huawei Cloud.
   - **LATEST_VERSION** indica a versão de segredo.

```
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.csms.v1.CsmsClient;
import com.huaweicloud.sdk.csms.v1.model.CreateSecretVersionRequest;
import com.huaweicloud.sdk.csms.v1.model.CreateSecretVersionRequestBody;
import com.huaweicloud.sdk.csms.v1.model.CreateSecretVersionResponse;
import com.huaweicloud.sdk.csms.v1.model.ListSecretStageRequest;
import com.huaweicloud.sdk.csms.v1.model.ListSecretStageResponse;
import com.huaweicloud.sdk.csms.v1.model.ShowSecretVersionRequest;
import com.huaweicloud.sdk.csms.v1.model.ShowSecretVersionResponse;
import com.huaweicloud.sdk.csms.v1.model.UpdateSecretStageRequest;
import com.huaweicloud.sdk.csms.v1.model.UpdateSecretStageRequestBody;

import java.util.Collections;
import java.util.List;

public class CsmsDualAccountExample {
    /**
     * Basic authentication information:
     * - ACCESS_KEY: Access key of the Huawei account
     * - SECRET_ACCESS_KEY: Secret access key of the Huawei account
     * - PROJECT_ID: Huawei Cloud project ID. For details, see https://
support.huaweicloud.com/intl/pt-br/productdesc-iam/iam_01_0023.html
* - CSMS_ENDPOINT: endpoint address for accessing CSMS. For details, see
https://support.huaweicloud.com/intl/pt-br/api-dew/dew_02_0052.html.
     * - There will be security risks if the AK/SK used for
authentication is directly written into code. Encrypt the AK/SK in the
configuration file or environment variables for storage;
     * - In this example, the AK/SK stored in the environment variables
are used for identity authentication. Configure the environment
variables HUAWEICLOUD_SDK_AK and HUAWEICLOUD_SDK_SK in the local
environment first.
     */
```



```
    private static final String ACCESS_KEY =
System.getenv("HUAWEICLOUD_SDK_AK");
    private static final String SECRET_ACCESS_KEY =
System.getenv("HUAWEICLOUD_SDK_SK");
    private static final String PROJECT_ID = "<ProjectID>";
    private static final String CSMS_ENDPOINT = "<CsmsEndpoint>";

    //Version ID used to query the latest secret version details.
    private static final String LATEST_VERSION = "latest";
    private static final String HW_CURRENT_STAGE = "SYSCURRENT";
    private static final String HW_PENDING_STAGE = "HW_PENDING";

    public static void main(String[] args) {
        String secretName = args[0];
        String secretString = args[1];

        dualAccountRotation(secretName, secretString);
    }

    /**
     * Example: secret rotation for two accounts
     *
     * @param secretName
     * @param secretString
     */
    private static void dualAccountRotation(String secretName, String
secretString) {
        // Create a new secret version with a custom version status.
Assume that the secret content is the account password of service A.
        CreateSecretVersionResponse newSecretVersion =
createNewSecretVersionWithStage(secretName,
                secretString,
Collections.singletonList(HW_PENDING_STAGE));
        String versionId = newSecretVersion.getVersionMetadata().getId();

        // Before rotation, check whether the pending state is modified.
        ListSecretStageResponse listSecretStageResponse =
showSecretStage(secretName, HW_PENDING_STAGE);
        assert
listSecretStageResponse.getStage().getVersionId().equals(versionId);

        // After the account and password of service A are updated, the
version status of the new secret is updated to SYSCURRENT. The old
secret is still stored in the service and cannot be accessed by
specifying latest.
        updateSecretStage(secretName, versionId, HW_CURRENT_STAGE);

        // Query the secret of the latest version by specifying latest
to complete the dual-account secret rotation.
        ShowSecretVersionResponse secretResponse =
showVersionDetail(secretName, LATEST_VERSION);

        assert
secretResponse.getVersion().getSecretString().equals(secretString);
    }

    /**
     * Example of creating a secret
     * If you add a secret version by customizing the version status,
the program will not point the SYSCURRENT version status to this version.
     *
     * @param secretName
     * @param newSecretVersionText
     * @param stageList
     * @return
     */
    private static CreateSecretVersionResponse
createNewSecretVersionWithStage(String secretName,
```



```
      String newSecretVersionText,

      List<String> stageList) {
        CsmsClient csmsClient = getCsmsClient();

        //Host the new secret in CSMS.
        CreateSecretVersionRequest createSecretVersionRequest = new
CreateSecretVersionRequest()
                .withSecretName(secretName)
                .withBody(new CreateSecretVersionRequestBody()
                        .withSecretString(newSecretVersionText)
                        .withVersionStages(stageList));

        CreateSecretVersionResponse secretVersion =
csmsClient.createSecretVersion(createSecretVersionRequest);

        System.out.printf("Created new version success, version id: %s",
secretVersion.getVersionMetadata().getId());

        return secretVersion;
    }

    /**
     * Point the input version state to the specified secret version.
     * @param secretName
     * @param versionId
     * @param newStageName
     */
    private static void updateSecretStage(String secretName, String
versionId, String newStageName) {
        UpdateSecretStageRequest updateSecretStageRequest = new
UpdateSecretStageRequest().withSecretName(secretName)
                .withStageName(newStageName).withBody(new
UpdateSecretStageRequestBody().withVersionId(versionId));

        CsmsClient csmsClient = getCsmsClient();

        csmsClient.updateSecretStage(updateSecretStageRequest);

        System.out.printf("Version stage update success. version id:%s,
new stage name:%s", versionId, newStageName);
    }

    /**
     * Query the secret of a specified version.
     * @param secretName
     * @param versionId
     * @return
     */
    private static ShowSecretVersionResponse showVersionDetail(String
secretName, String versionId) {
        ShowSecretVersionRequest showSecretVersionRequest = new
ShowSecretVersionRequest().withSecretName(secretName)
                .withVersionId(versionId);

        CsmsClient csmsClient = getCsmsClient();
        ShowSecretVersionResponse secretDetail =
csmsClient.showSecretVersion(showSecretVersionRequest);

        System.out.printf("Query latest version success. version id:%s",
 secretDetail.getVersion().getVersionMetadata().getId());
        return secretDetail;
    }

    /**
     * Query the status of a specified version.
     * @param secretName
     * @param stageName
     * @return
```



```
     */
    private static ListSecretStageResponse showSecretStage(String
secretName, String stageName) {
        ShowSecretStageRequest showSecretStageRequest = new
ShowSecretStageRequest()
                .withSecretName(secretName).withStageName(stageName);
        CsmsClient csmsClient = getCsmsClient();
        return csmsClient.showSecretStage(showSecretStageRequest);
    }

    /**
     * Obtain the CSMS client.
     *
     * @return
     */
    private static CsmsClient getCsmsClient() {
        BasicCredentials auth = new BasicCredentials()
                .withAk(ACCESS_KEY)
                .withSk(SECRET_ACCESS_KEY)
                .withProjectId(PROJECT_ID);
        return
CsmsClient.newBuilder().withCredential(auth).withEndpoint(CSMS_ENDPOINT).
build();
    }
}
```

2.4.4 Rotação de segredos de IAM usando FunctionGraph

Cenário

Esta seção descreve como rotacionar segredos do IAM pelo KMS usando um modelo do FunctionGraph.

Restrições

- Apenas as contas de membros do IAM podem ser rotacionadas. As contas principais do IAM não podem ser rotacionadas.
- A conta de membro do IAM a ser rotacionada deve ter pelo menos um grupo de segredos.
- O CSMS está disponível na região.

Procedimento

Crie uma agência.

Passo 1 **Faça logon no console de gerenciamento**.

Passo 2 Clique em  no canto superior esquerdo do console de gerenciamento e selecione uma região ou projeto.

Passo 3 Clique em  no canto superior esquerdo da página e escolha **Management & Governance** > **Identity and Access Management**.

Passo 4 No painel de navegação à esquerda, escolha **Agencies**. Clique em **Create Agency** no canto superior direito.

Passo 5 Configure os parâmetros, conforme mostrado na figura a seguir. A **Tabela 2-6** lista os parâmetros.

Figura 2-13 Criação de uma agência

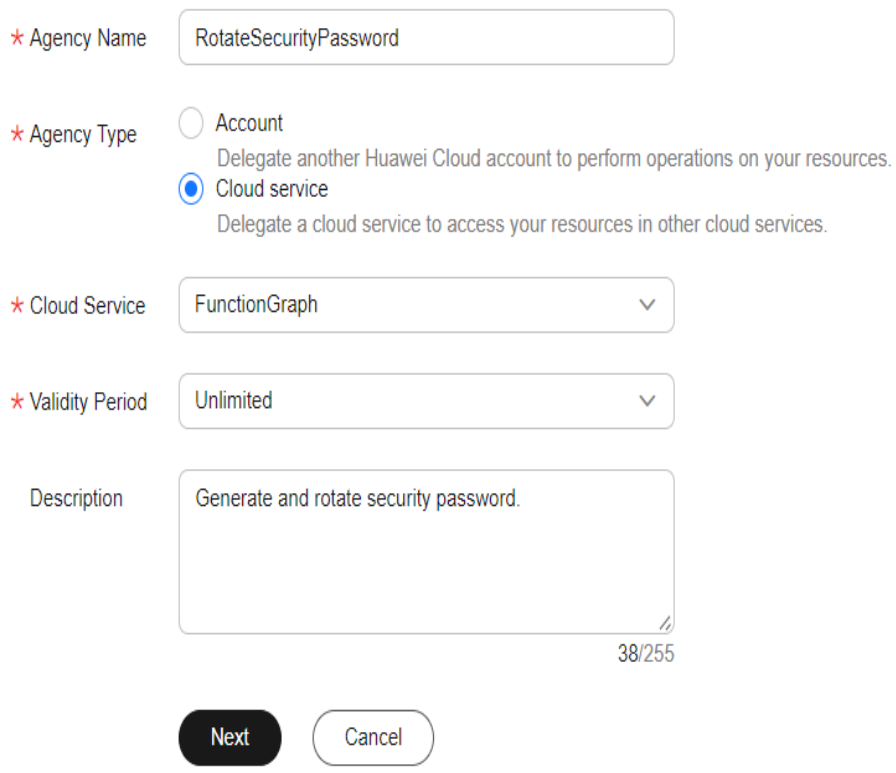


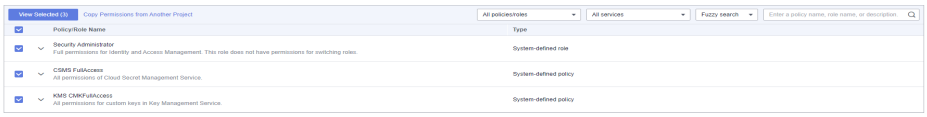
Tabela 2-6 Parâmetros para criar uma agência

Parâmetro	Descrição
Agency Name	Insira um nome de agência personalizado, por exemplo, **test**.
Agency Type	Selecione **Cloud service**.
Cloud Service	Escolha **FunctionGraph**.
Validity Period	Defina este parâmetro com base no cenário real. Se a função precisar ser executada por um longo tempo, escolha **Unlimited**.
Description	Defina este parâmetro conforme necessário.

Passo 6 Clique em **Next**.

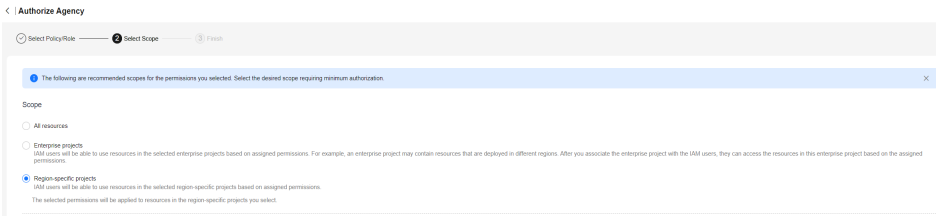
Passo 7 Selecione **Security Administrator**, **CSMS FullAccess** e **KMS CMKFullAccess**, conforme mostrado na figura a seguir.

Figura 2-14 Atribuição de permissões



Passo 8 Clique em **Next** e escolha um escopo de autorização, conforme mostrado na figura a seguir.

Figura 2-15 Escopo de autorização



Passo 9 Clique em **OK**.

----Fim

Crie um modelo de função e uma função.

Passo 1 **Faça logon no console de gerenciamento**.

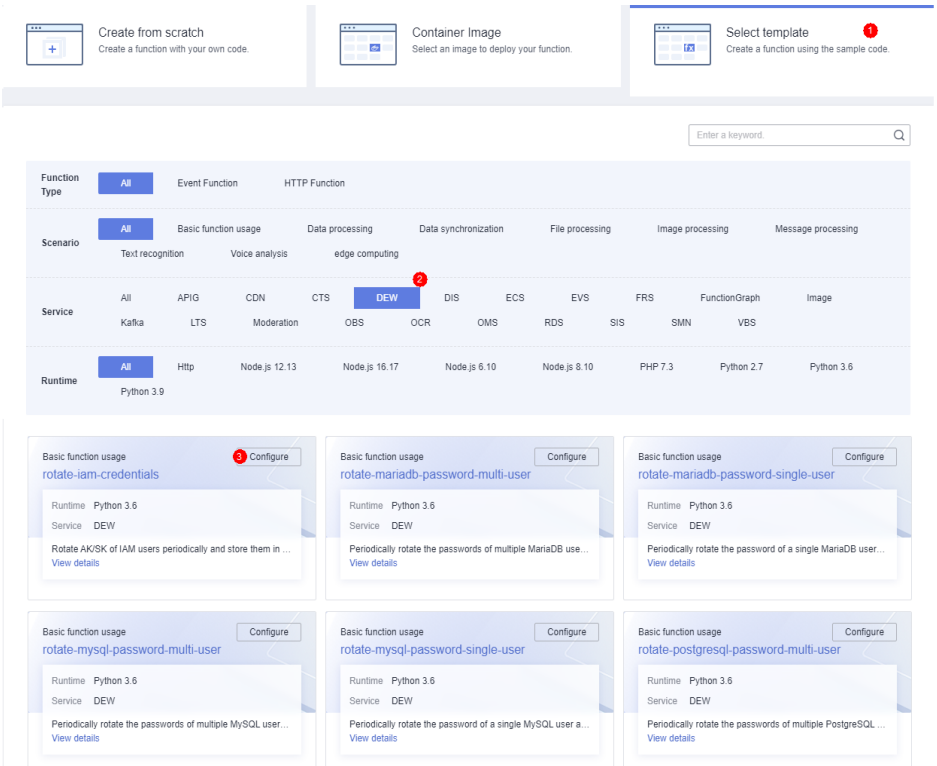
Passo 2 Clique em no canto superior esquerdo do console de gerenciamento e selecione uma região ou projeto.

Passo 3 Clique em no canto superior esquerdo da página e escolha **Compute** > **FunctionGraph**.

Passo 4 Clique em **Create Function** no canto superior direito.

Passo 5 Clique em **Select template**. Na área **Service**, clique em **DEW**, localize **rotate-iam-credentials** abaixo e clique em **Configure** no canto superior direito da caixa, conforme mostrado na figura a seguir.

Figura 2-16 Criação de uma função



Passo 6 Configure os parâmetros, que são descritos em **Tabela 2-7**.

Figura 2-17 Informação básica da função

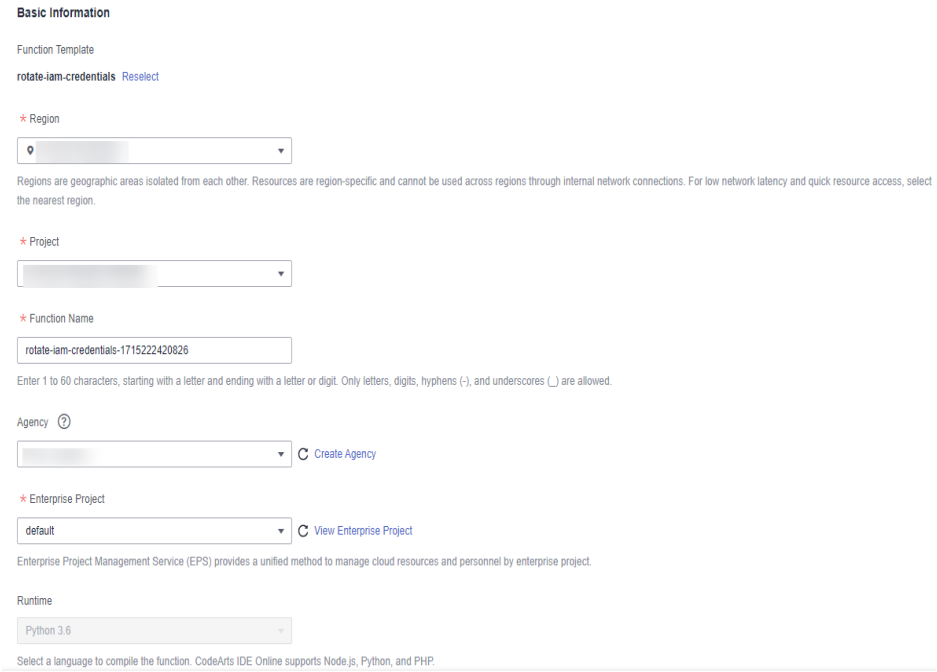


Tabela 2-7 Parâmetros básicos

Parâmetro	Descrição
Region	Selecione a região em que a função deve ser implementada.
Project	Selecione o projeto no qual a função será implementada.
Function Name	Insira um nome de função personalizado.
Agency	Selecione uma agência.
Enterprise Project	Se você ativou projetos empresariais, selecione um para adicionar uma função a ele. Se você não tiver ativado um projeto empresarial, esse parâmetro ficará indisponível. Pule este passo.

Passo 7 Configure as variáveis de ambiente, que são descritas em **Tabela 2-8**.

Figura 2-18 Variáveis ambientais

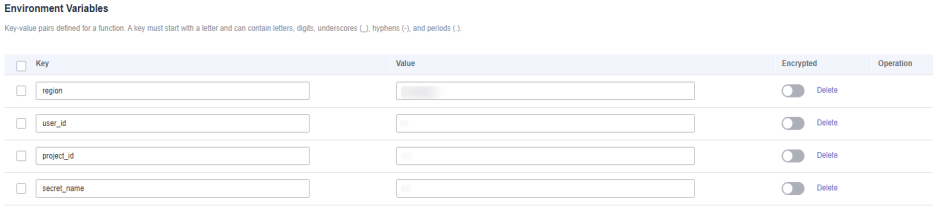


Tabela 2-8 Variáveis ambientais

Parâmetro	Descrição
region	Projeto, por exemplo, se a região é CN North-Beijing4, o nome do projeto deve ser **cn-north-4**. **NOTA** Para obter o projeto, passe o mouse sobre o nome de usuário e escolha **My Credentials** na lista suspensa. Os projetos são exibidos.
user_id	Insira o ID do usuário do IAM a ser rotacionado. **NOTA** Para obter o ID de usuário do IAM, passe o mouse sobre o nome de usuário e escolha **My Credentials** na lista suspensa. O ID é exibido.
project_id	Insira o ID do projeto. **NOTA** Para obter o ID de usuário do IAM, passe o mouse sobre o nome de usuário e escolha **My Credentials** na lista suspensa. Os IDs dos projetos são exibidos.
secret_name	Digite o nome do segredo a ser criado, que não pode ser o mesmo que um nome de segredo existente.

Passo 8 Configure as variáveis de acionamento, que são descritas em **Tabela 2-9**.

Figura 2-19 Variáveis de acionamento

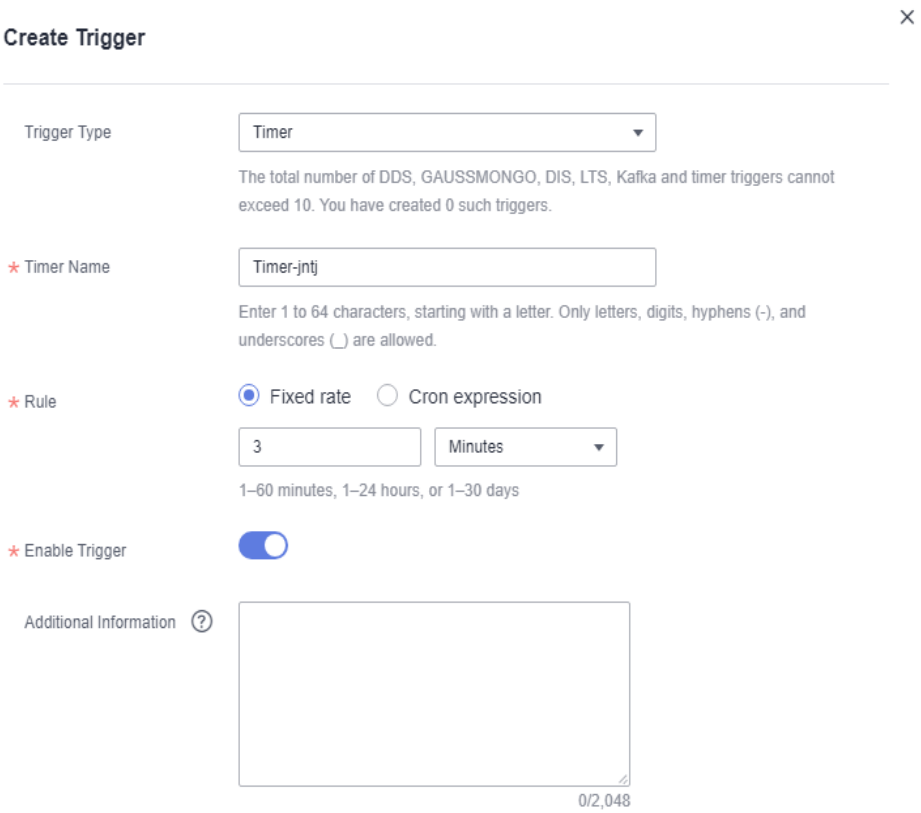


Tabela 2-9 Variáveis de acionamento

Parâmetro	Descrição
Timer Name	Personalizado
Rule	Configurado conforme necessário
Enable Trigger	Configurado conforme necessário

Passo 9 Clique em **Create Function**.

----Fim

Depure.

Para obter detalhes sobre como depurar um fluxo de trabalho de função, consulte **Depuração on-line**.

Veja o segredo.

Passo 1 **Faça logon no console de gerenciamento**.

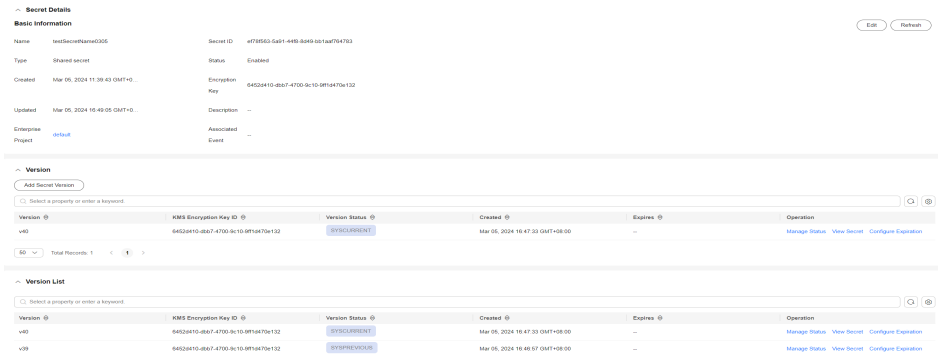
Passo 2 Clique em no canto superior esquerdo do console de gerenciamento e selecione uma região ou um projeto.

Passo 3 Clique em ☰ à esquerda, escolha **Security & Compliance** > **Data Encryption Workshop**, a página de gerenciamento de chaves é exibida.

Passo 4 No painel de navegação, escolha **Cloud Secret Management Service**.

Passo 5 Consulte o segredo especificado na **Etapa 7**. Clique nele para ver os detalhes. O segredo válido atual, segredos históricos e outras informações são exibidos.

Figura 2-20 Detalhes de segredos



Passo 6 Localize a versão mais recente do segredo, clique em **View Secret** na coluna **Operation** para verificar a AK/SK.

----Fim

3 Geral

3.1 Repetição de solicitações do DEW com falha usando retirada exponencial

Cenário

Se você receber uma mensagem de erro ao chamar uma API, poderá usar a retirada exponencial para repetir a solicitação.

Como funciona

Se erros consecutivos (como erros de limitação de tráfego) forem relatados pelo lado do serviço, o acesso contínuo continuará causando conflitos. A retirada exponencial pode ajudá-lo a evitar tais erros.

Restrições

A conta atual tem uma chave ativada.

Exemplo

1. Prepare informações básicas de autenticação.
   - ACCESS_KEY: chave de acesso da conta da Huawei. Para obter detalhes, consulte **Como obter uma chave de acesso (AK/SK)?**
   - SECRET_ACCESS_KEY: chave de acesso de segredo da conta da Huawei. Para obter detalhes, consulte **Como obter uma chave de acesso (AK/SK)?**
   - PROJECT_ID: ID do projeto do site. Para obter detalhes, consulte **Obtenção de um ID de projeto**.
   - KMS_ENDPOINT: ponto de extremidade para acessar o KMS. Para obter detalhes, consulte **Pontos de extremidade**.
   - Haverá riscos de segurança se a AK/SK usada para autenticação for gravada diretamente no código. Criptografe a AK/SK no arquivo de configuração ou nas variáveis de ambiente para armazenamento.

– Neste exemplo, a AK/SK armazenada nas variáveis de ambiente é usada para autenticação de identidade. Configure as variáveis de ambiente **HUAWEICLOUD_SDK_AK** e **HUAWEICLOUD_SDK_SK** primeiro no ambiente local.

2. Código para retirada exponencial:

```
import com.huaweicloud.sdk.core.auth.BasicCredentials;
 import com.huaweicloud.sdk.core.auth.ICredential;
 import com.huaweicloud.sdk.core.exception.ClientRequestException;
 import com.huaweicloud.sdk.kms.v2.model.EncryptDataRequest;
 import com.huaweicloud.sdk.kms.v2.model.EncryptDataRequestBody;
 import com.huaweicloud.sdk.kms.v2.KmsClient;

 public class KmsEncryptExample {

     private static final String ACCESS_KEY =
System.getenv("HUAWEICLOUD_SDK_AK");

     private static final String SECRET_ACCESS_KEY =
System.getenv("HUAWEICLOUD_SDK_SK");

     private static final String KMS_ENDPOINT = "xxxx";

     private static final String KEY_ID = "xxxx";

     private static final String PROJECT_ID = "xxxx";


     private static KmsClient KmsClientInit() {
         ICredential auth = new BasicCredentials()
                 .withAk(ACCESS_KEY)
                 .withSk(SECRET_ACCESS_KEY)
                 .withProjectId(PROJECT_ID);
         return KmsClient.newBuilder()
                 .withCredential(auth)
                 .withEndpoint(KMS_ENDPOINT)
                 .build();
     }

     public static long getWaitTime(int retryCount) {
         long initialDelay = 200L;
         return (long) (Math.pow(2, retryCount) * initialDelay);
     }

     public static void encryptData(KmsClient client, String plaintext) {
         EncryptDataRequest request = new EncryptDataRequest().withBody(
                 new EncryptDataRequestBody()
                         .withKeyId(KEY_ID)
                         .withPlainText(plaintext));
         client.encryptData(request);
     }

     public static void main(String[] args) {
         int maxRetryTimes = 6;
         String plaintext = "plaintext";
         String errorMsg = "The throttling threshold has been reached";

         KmsClient client = KmsClientInit();
         for (int i = 0; i < maxRetryTimes; i++) {
             try {
                 encryptData(client, plaintext);
                 return;
             } catch (ClientRequestException e) {
                 if (e.getErrorMsg().contains(errorMsg)) {
                     try {
                         Thread.sleep(getWaitTime(i));
                     } catch (InterruptedException ex) {
                         throw new RuntimeException(ex);
                     }
```



```
                    }
                }
            }
        }
}
```


A Histórico de alterações

Lançado em	Descrição
28/03/2024	Este é o 8º lançamento oficial. Adição de "Uso do CSMS para rotacionar automaticamente as senhas de segurança".
15/03/2023	Este é o 7º lançamento oficial. Adição de "Uso do SDK de criptografia para criptografar e descriptografar arquivos locais". Adição de "Criptografia e descriptografia de dados por meio de DR entre regiões".
03/03/2023	Este é o 6º lançamento oficial. Adição da seção "Criptografia de dados no ECS".
13/12/2022	Este é o 5º lançamento oficial. Adição de "Repetição de solicitações do DEW com falha usando retirada exponencial".
11/11/2022	Este é o 4º lançamento oficial. ● Adição de "Uso do CSMS para evitar vazamento de AK e SK". ● Alteração de "Uso do CSMS para aplicações para fazer logon em bancos de dados" para "Uso do CSMS para alterar senhas de contas de banco de dados codificadas rigidamente".
14/03/2022	Este é o 3º lançamento oficial. Adição das seções "Uso do CSMS para aplicações para fazer logon em bancos de dados" e "Rotação de segredos".
30/09/2021	Este é o 2º lançamento oficial. Adição de "Criptografia ou descriptografia de uma grande quantidade de dados".
30/06/2021	Este é o 1º lançamento oficial.

