

Data Encryption Workshop 01

Prácticas recomendadas de

Edición 01
Fecha 2024-09-13



Copyright © Huawei Cloud Computing Technologies Co., Ltd. 2024. Todos los derechos reservados.

Quedan terminantemente prohibidas la reproducción y/o la divulgación totales y/o parciales del presente documento de cualquier forma y/o por cualquier medio sin la previa autorización por escrito de Huawei Cloud Computing Technologies Co., Ltd.

Marcas registradas y permisos



El logotipo HUAWEI y otras marcas registradas de Huawei pertenecen a Huawei Technologies Co., Ltd.

Todas las demás marcas registradas y los otros nombres comerciales mencionados en este documento son propiedad de sus respectivos titulares.

Aviso

Es posible que la totalidad o parte de los productos, las funcionalidades y/o los servicios que figuran en el presente documento no se encuentren dentro del alcance de un contrato vigente entre Huawei Cloud y el cliente. Las funcionalidades, los productos y los servicios adquiridos se limitan a los estipulados en el respectivo contrato. A menos que un contrato especifique lo contrario, ninguna de las afirmaciones, informaciones ni recomendaciones contenidas en el presente documento constituye garantía alguna, ni expresa ni implícita.

Huawei está permanentemente preocupada por la calidad de los contenidos de este documento; sin embargo, ninguna declaración, información ni recomendación aquí contenida constituye garantía alguna, ni expresa ni implícita. La información contenida en este documento se encuentra sujeta a cambios sin previo aviso.

Huawei Cloud Computing Technologies Co., Ltd.

Dirección: Huawei Cloud Data Center Jiaoxinggong Road
Avenida Qianzhong
Nuevo distrito de Gui'an
Gui Zhou, 550029
República Popular China

Sitio web: <https://www.huaweicloud.com/intl/es-us/>

Índice

1 Key Management Service.....	1
1.1 Uso de KMS para cifrar datos sin conexión.....	1
1.1.1 Encriptación o descifrado de pequeños volúmenes de datos.....	1
1.1.2 Encriptación o descifrado de una gran cantidad de datos.....	3
1.2 Uso de KMS para cifrar y descifrar datos para servicios en la nube.....	12
1.2.1 Descripción general.....	12
1.2.2 Encriptación de datos en ECS.....	14
1.2.3 Encriptación de datos en OBS.....	14
1.2.4 Encriptación de datos en EVS.....	16
1.2.5 Encriptación de datos en IMS.....	18
1.2.6 Encriptación de una instancia de BD de RDS.....	20
1.2.7 Encriptación de una instancia de BD de DDS.....	21
1.3 Uso del SDK de encriptación para cifrar y descifrar datos locales.....	22
1.4 Encriptación y descifrado de datos con recuperación ante desastres entre regiones.....	25
1.5 Uso de KMS para proteger la integridad de archivos.....	28
2 Cloud Secret Management Service.....	32
2.1 Uso de CSMS para cambiar las contraseñas de cuenta de base de datos codificadas de forma rígida.....	32
2.2 Uso de CSMS para evitar fugas de AK y SK.....	36
2.3 Uso de CSMS para rotar automáticamente contraseñas de seguridad.....	42
2.4 Rotación de secretos.....	49
2.4.1 Descripción general.....	49
2.4.2 Rotación de un secreto para un usuario.....	50
2.4.3 Rotación de un secreto para dos usuarios.....	53
2.4.4 Rotación de secretos de IAM con FunctionGraph.....	57
3 Generales.....	64
3.1 Reintento de solicitudes de DEW fallidas con retardo exponencial.....	64

1 Key Management Service

1.1 Uso de KMS para cifrar datos sin conexión

1.1.1 Encriptación o descifrado de pequeños volúmenes de datos

Escenario

Puede utilizar herramientas en línea en la consola del Key Management Service (KMS) o invocar a las API de KMS necesarias para cifrar o descifrar directamente datos de pequeño tamaño con una CMK, como contraseñas, certificados o números de teléfono.

Restricciones

Actualmente, un máximo de 4 KB de datos pueden ser cifrados o descifrados de esta manera.

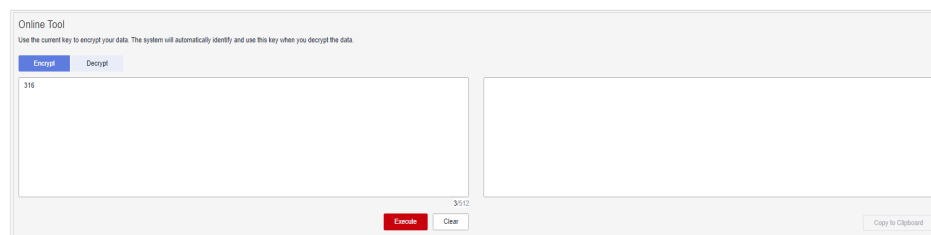
Encriptación y descifrado con herramientas en línea

- Encriptación de datos

Paso 1 Haga clic en el alias de una clave personalizada para ver sus detalles y vaya a la herramienta en línea para el cifrado y descifrado de datos.

Paso 2 Haga clic en **Encrypt**. En el cuadro de texto de la izquierda, introduzca los datos que se van a cifrar. Para más detalles, consulte [Figura 1-1](#).

Figura 1-1 Encriptación de datos



Paso 3 Haga clic en **Execute**. El texto cifrado de los datos se muestra en el cuadro de texto de la derecha.

 **NOTA**

- Utilice el CMK actual para cifrar los datos.
- Puede hacer clic en **Clear** para borrar los datos introducidos.
- Puede hacer clic en **Copy to Clipboard** para copiar el texto cifrado y guardarlo en un archivo local.

● **Descifrado de datos**

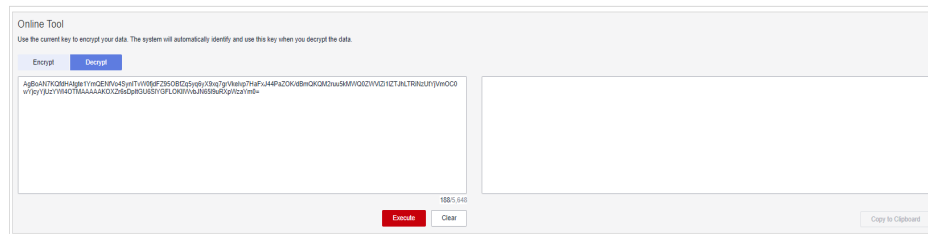
Paso 4 Puede hacer clic en cualquier clave no predeterminada en estado de **Enabled** para ir a la página de encriptación y descifrado de la herramienta en línea.

Paso 5 Haga clic en **Decrypt**. En el cuadro de texto de la izquierda, introduzca los datos que se van a descifrar. Para más detalles, consulte [Figura 1-2](#).

 **NOTA**

- La herramienta identificará el CMK de encriptación original y lo utilizará para descifrar los datos.
- Sin embargo, si el CMK se ha eliminado, el descifrado falla.

Figura 1-2 Desencriptación de datos



Paso 6 Haga clic en **Execute**. El texto sin formato de los datos se muestra en el cuadro de texto de la derecha.

 **NOTA**

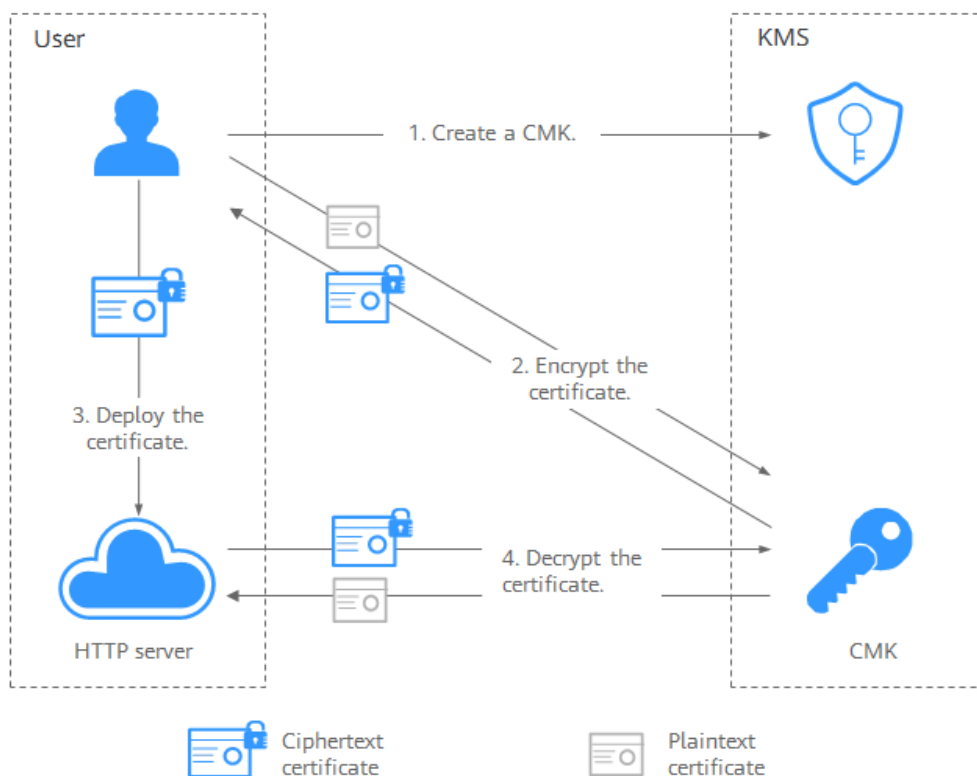
- Puede hacer clic en **Copy to Clipboard** para copiar el texto sin formato y guardarlo en un archivo local.

----Fin

Invocaciones a las API para encriptar y descifrar

[Figura 1-3](#) muestra un ejemplo sobre cómo invocar a las API de KMS para cifrar y descifrar un certificado de HTTPS.

Figura 1-3 Cifrado y descifrado de un certificado HTTPS



El procedimiento es el siguiente:

1. Crear una CMK en KMS.
2. Invoque a la interfaz **encrypt-data** de KMS y utilice la CMK para cifrar el certificado de texto plano.
3. Despliegue el certificado en un servidor.
4. El servidor utiliza la interfaz **decrypt-data** de KMS para descifrar el certificado de texto cifrado.

📖 NOTA

Si introduce y cifra texto en la consola, el texto se codificará en formato Base64 antes de transferirlo al backend para su encriptación. El resultado del descifrado devuelto mediante API estará en formato Base64. El texto cifrado mediante API no se puede descifrar en la consola, o se devolverán los caracteres ilegibles.

1.1.2 Encriptación o descifrado de una gran cantidad de datos

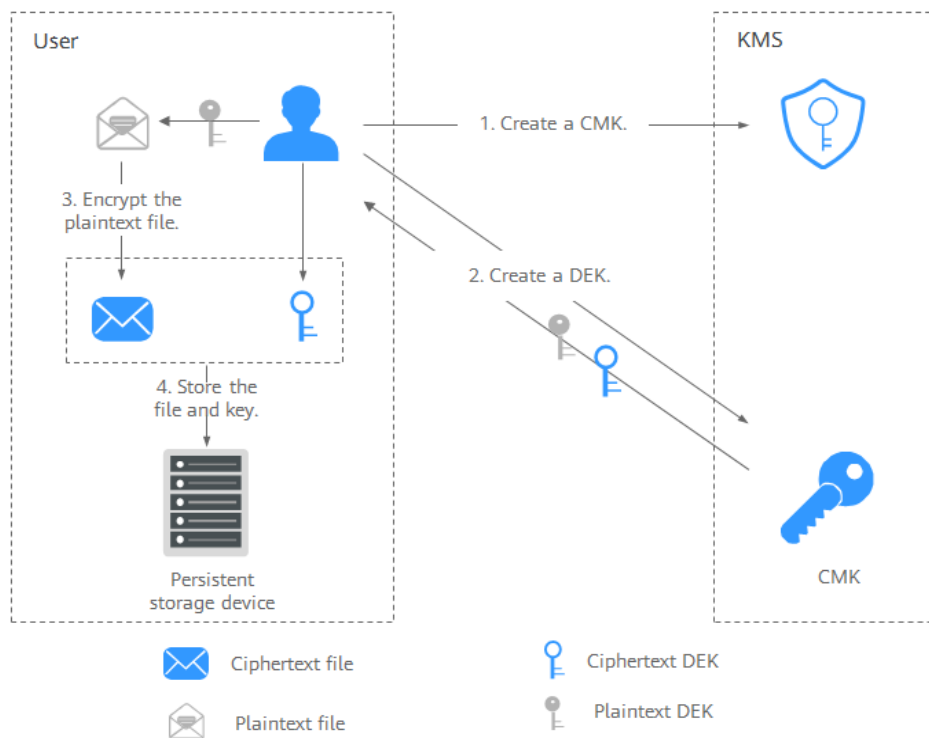
Escenario

Si desea encriptar o descifrar grandes volúmenes de datos, como imágenes, videos y archivos de bases de datos, puede utilizar la encriptación de sobres, que permite encriptar o descifrar archivos sin tener que transferir una gran cantidad de datos por la red.

Procesos de encriptación o descifrado

- Encriptación de datos de gran tamaño

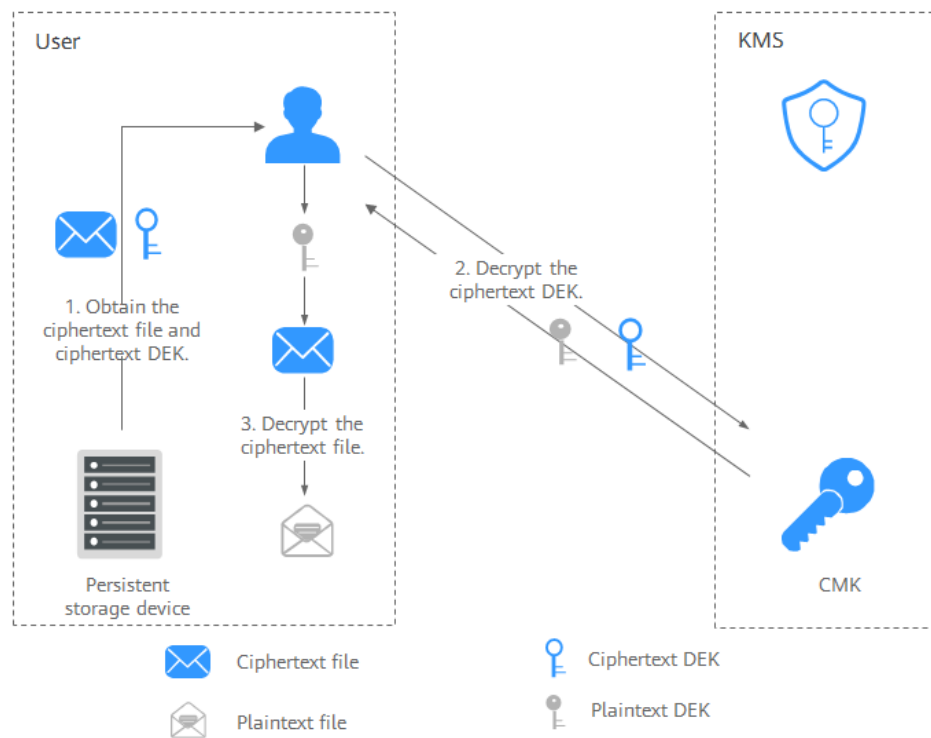
Figura 1-4 Cifrado de un archivo local



El proceso es el siguiente:

- Crear una CMK en KMS.
 - Invocar a la API **create-datakey** de KMS para crear una DEK. Se generará una DEK en texto plano y una DEK en texto encriptado. La DEK de texto encriptado se genera cuando se utiliza una CMK para encriptar la DEK de texto plano.
 - Utilice la DEK de texto plano para cifrar un archivo de texto plano, generando un archivo de texto cifrado.
 - Almacene la DEK de texto cifrado y el archivo de texto cifrado juntos en un dispositivo de almacenamiento permanente o en un servicio de almacenamiento.
- Descifrado de datos de gran tamaño

Figura 1-5 Descifrar un archivo local



El proceso es el siguiente:

- Lea la DEK de texto cifrado y el archivo de texto cifrado desde el dispositivo de almacenamiento permanente o servicio de almacenamiento.
- Invoque a la API de **decrypt-datakey** de KMS y use la CMK correspondiente (el utilizado para cifrar la DEK) para descifrar la DEK de texto cifrado. Luego obtiene la DEK de texto plano.
Si se elimina la CMK, el descifrado fallará. Mantenga sus CMK correctamente.
- Utilice la DEK de texto plano para descifrar el archivo de texto cifrado.

API relacionadas con la encriptación de sobre

Puede utilizar las siguientes API para cifrar y descifrar datos.

API	Descripción
Creación de una DEK	Crear una DEK.
Encriptación de una DEK	Cifrar una DEK con la clave principal especificada.
Descifrado de una DEK	Descifrar una DEK con la clave principal especificada.

Cifrado de un archivo local

1. Crear una CMK en la consola de gestión. Para obtener más detalles, consulte [Creación de una CMK](#).
2. Preparar la información básica de autenticación.
 - **ACCESS_KEY**: clave de acceso del ID de Huawei
 - **SECRET_ACCESS_KEY**: clave de acceso secreta del ID de Huawei
 - **PROJECT_ID**: ID de proyecto de un sitio de Huawei Cloud. Para obtener más detalles, véase [Proyecto](#).
 - **KMS_ENDPOINT**: punto de conexión para acceder a la KMS. Consulte [Puntos de conexión](#) para obtener más detalles.
 - Habrá riesgos de seguridad si la AK/SK utilizada para la autenticación se escribe directamente en el código. Cifre la AK/SK en el archivo de configuración o en las variables de entorno para su almacenamiento.
 - En este ejemplo, las AK/SK almacenadas en las variables de entorno se utilizan para la autenticación de identidad. Primero configure las variables de entorno **HUAWEICLOUD_SDK_AK** y **HUAWEICLOUD_SDK_SK** en el entorno local.
3. Cifrar un archivo local.

El código de ejemplo es el siguiente.

- CMK es el ID de la clave creada en la consola de gestión de Huawei Cloud.
- **FirstPlainFile.jpg** es el archivo de datos de texto plano.
- El archivo de datos generado tras la encriptación es **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());
    }
}
}
```

Descifrado de un archivo local

1. Preparar la información básica de autenticación.
 - **ACCESS_KEY**: clave de acceso del ID de Huawei
 - **SECRET_ACCESS_KEY**: clave de acceso secreta del ID de Huawei
 - **PROJECT_ID**: ID de proyecto de un sitio de Huawei Cloud. Para obtener más detalles, véase [Proyecto](#).
 - **KMS_ENDPOINT**: punto de conexión para acceder a la KMS. Consulte [Puntos de conexión](#) para obtener más detalles.
 - Habrá riesgos de seguridad si la AK/SK utilizada para la autenticación se escribe directamente en el código. Cifre la AK/SK en el archivo de configuración o en las variables de entorno para su almacenamiento.

- En este ejemplo, las AK/SK almacenadas en las variables de entorno se utilizan para la autenticación de identidad. Primero configure las variables de entorno **HUAWEICLOUD_SDK_AK** y **HUAWEICLOUD_SDK_SK** en el entorno local.

2. Descifrar un archivo local.

El código de ejemplo es el siguiente.

- CMK es el ID de la clave creada en la consola de gestión de Huawei Cloud.
- El archivo de datos generado tras la encriptación es **SecondEncryptFile.jpg**.
- El archivo de datos generado tras la encriptación y desencriptación es **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 de KMS para cifrar y descifrar datos para servicios en la nube

1.2.1 Descripción general

KMS es un servicio en la nube seguro, confiable y fácil de usar que ayuda a los usuarios a crear, gestionar y proteger claves de manera centralizada.

Una vez que sus servicios en la nube se integran con KMS, para encriptar los datos en la nube, solo necesita seleccionar una CMK gestionada por KMS para la encriptación.

Puede seleccionar una clave principal predeterminada (DMK) creada automáticamente por un servicio en la nube con KMS, o una clave que haya creado o importado a KMS. Para obtener más detalles, consulte [Diferencias entre una CMK y una clave principal predeterminada](#).

Tabla 1-1 Servicios en la nube que utilizan la encriptación de KMS

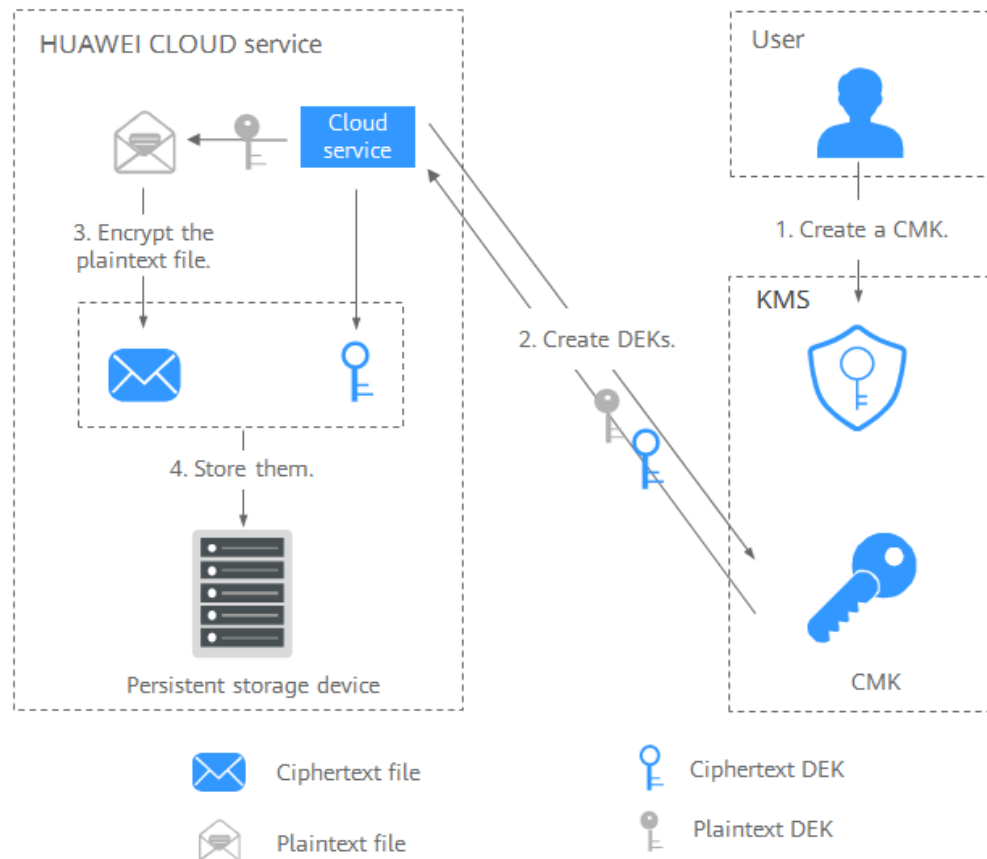
Categoría	Servicio	Modo de encriptación
Cómputo	Elastic Cloud Server (ECS)	Puede encriptar una imagen o un disco de EVS en ECS. <ul style="list-style-type: none"> ● Al crear un ECS, si selecciona una imagen cifrada, el disco del sistema del ECS creado automáticamente tendrá habilitado la encriptación, con el mismo modo de encriptación que la de la imagen. ● Al crear un ECS, puede cifrar los discos de datos agregados.
	Image Management Service (IMS)	Encriptación de datos en IMS
Almacenamiento	Object Storage Service (OBS)	Encriptación de datos en OBS
	Elastic Volume Service (EVS)	Encriptación de datos en EVS
	Volume Backup Service (VBS)	Por lo general, VBS crea copia de respaldo en línea para un único disco de EVS (sistema o disco de datos) del servidor. Si está encriptado, sus datos de respaldo se almacenarán en modo encriptado.

Categoría	Servicio	Modo de encriptación
	Cloud Server Backup Service (CSBS)	CSBS principalmente crea copia de respaldo de consistencia en línea para todos los discos de EVS del servidor. Las copias de respaldo de CSBS también se mostrarán en la página de VBS. Si está encriptado, sus datos de respaldo se almacenarán en modo encriptado.
Bases de datos	RDS for MySQL	Encriptación de una instancia de BD de RDS
	RDS for PostgreSQL	
	RDS for SQL Server	
	Document Database Service (DDS)	Encriptación de una instancia de BD de DDS

Proceso de encriptación

Los servicios de Huawei Cloud utilizan la tecnología de encriptación en sobres e invocan a las API de KMS para cifrar los recursos del servicio. Sus CMK están bajo su propia gestión. Con su concesión, los servicios de Huawei Cloud utilizan una CMK específica suya para cifrar datos.

Figura 1-6 Cómo Huawei Cloud utiliza KMS para encriptación



El proceso de encriptación es el siguiente:

1. Crear una CMK en KMS.
2. Un servicio de Huawei Cloud invoca a la API **create-datakey** del KMS para crear una DEK. Se generan una DEK de texto plano y una DEK de texto cifrado.

 **NOTA**

Las DEK de texto cifrado se generan cuando se utiliza una CMK para cifrar las DEK de texto plano.

3. El servicio de Huawei Cloud utiliza DEK de texto plano para cifrar un archivo de texto plano, generando un archivo de texto cifrado.
4. El servicio Huawei Cloud guarda la DEK del texto cifrado y el archivo de texto cifrado juntos en un dispositivo de almacenamiento permanente o un servicio de almacenamiento.

 **NOTA**

Cuando los usuarios descargan los datos del servicio Huawei Cloud, el servicio utiliza la CMK especificada por KMS para descifrar la DEK de texto cifrado, utiliza la DEK descifrada para descifrar los datos y, a continuación, proporciona los datos descifrados para que los usuarios los descarguen.

1.2.2 Encriptación de datos en ECS

Descripción general

KMS admite la encriptación con un solo clic para ECS. Se pueden cifrar las imágenes y los discos de datos del ECS.

- Al crear un ECS, si selecciona una imagen cifrada, el disco del sistema del ECS creado automáticamente tendrá habilitado la encriptación, con el mismo modo de encriptación que la de la imagen.
- Al crear un ECS, puede cifrar los discos de datos agregados.

Consulte [Encriptación de datos en IMS](#) para obtener detalles sobre cómo cifrar una imagen.

Consulte [Encriptación de datos en EVS](#) para obtener detalles sobre cómo cifrar un disco de datos.

1.2.3 Encriptación de datos en OBS

Descripción general

Una vez habilitado la encriptación del lado del servidor, los datos de un objeto cargados en Object Storage Service (OBS) se cifran en el servidor antes de almacenarse. Cuando se descarga el objeto, los datos se descifran primero en el servidor.

KMS utiliza un módulo de seguridad de hardware (HSM) de terceros para proteger las claves, lo que le permite crear y gestionar claves de encriptación fácilmente. Las claves no se muestran en texto plano fuera de los HSM, lo que impide su divulgación. Con KMS, todas las operaciones de las claves se controlan y registran, y se pueden proporcionar registros de uso de todas las claves para cumplir los requisitos normativos.

Se puede implementar la encriptación del lado del servidor con claves gestionadas por KMS (SSE-KMS) para los objetos que se van a cargar. Debe crear una clave usando KMS o usar la

clave predeterminada proporcionada por KMS. Luego, puede usar la clave para cifrar el objeto en el servidor al cargarlo en OBS.

Carga de datos en modo de encriptación del lado del servidor (en la consola)

- Paso 1** En la lista de buckets de la consola de OBS, haga clic en un bucket para ir a la página **Overview**.
- Paso 2** En el árbol de navegación de la izquierda, seleccione **Objects**.
- Paso 3** Haga clic en **Upload Object**. Aparece el cuadro de diálogo **Upload Object**.
- Paso 4** Haga clic en **Add File**. Seleccione el archivo que desea cargar y luego haga clic en **Open**.
- Paso 5** Configure **Server-Side Encryption** como **SSE-KMS**; seleccione la clave predeterminada o una personalizada y haga clic en **Upload**.

Figura 1-7 Encriptación de un objeto que se va a cargar

Upload Object [How to Upload a File Larger than 5 GB?](#) ×

1 Upload Object ——— 2 (Optional) Configure Advanced Settings

Upload actions will generate requests. After the upload, you will be billed for data storage. ×

Storage Class: **Standard** | Infrequent Access | Archive
Optimized for frequently accessed (multiple times per month) data such as small and essential files that require low latency. If you do not change this setting, your uploaded objects will be stored using the default storage class you selected during bucket creation. [Learn more](#)

Upload Object: ! The file or folder you newly upload will overwrite any existing file or folder with the same name. To keep different versions of the same file or folder, enable versioning for the current bucket.

Drag files or folders here to upload. Or [add file](#)
(A maximum of 100 files can be uploaded at a time. The total size cannot exceed 5 GB.)

Server-Side Encryption: **Inherit from bucket** | **SSE-KMS** | SSE-OBS
! Encryption is recommended to keep data secure. What you use beyond the free quota given by KMS will be billed. [Pricing details](#)
Encryption keys managed by KMS are used to encrypt your objects.

Encryption Key Type: **Default** | **Custom**
You can use a custom key below to encrypt your objects.

Custom: **KMS-cc12** [Create KMS Key](#)

Next: (Optional) Configure Advanced Settings Upload Cancel

Nombre de clave: nombre de la clave personalizada. La clave se crea en DEW y se utiliza para la protección cifrada de los datos. OBS proporciona **obs/default** como clave por defecto. Puede utilizar la clave predeterminada o crear una clave en DEW.

📖 NOTA

Generalmente, se elige la clave AES cuando se utiliza SSE-KMS. Las claves de encriptación de SM4 solo se pueden utilizar en la región CN North-Ulanqab1.

Paso 6 Después de cargar el objeto, haga clic en él para ver su estado de encriptación.

 **NOTA**

- No se puede cambiar el estado de encriptación del objeto.
- No se puede eliminar una clave en uso. De lo contrario, el objeto cifrado con esta clave no se puede descargar.

----Fin

Carga de datos en modo de encriptación del lado del servidor (con una API)

Puede invocar a la API requerida del OBS para cargar un archivo en modo SSE-KMS. Consulte la *Referencia de las API de Object Storage Service* para obtener más detalles.

1.2.4 Encriptación de datos en EVS

Descripción general

Cuando sus servicios requieran encriptar los datos almacenados en los discos en Elastic Volume Service (EVS), le proporciona EVS la función de encriptación. Puede encriptar los discos de EVS recién creados. Las claves utilizadas por los discos de EVS cifrados son proporcionadas por KMS de DEW, seguras y cómodas. Por lo tanto, no es necesario establecer y mantener la infraestructura de gestión de claves.

La encriptación de disco se utiliza solo para discos de datos. Encriptación del disco del sistema se basa en la imagen. Consulte [Encriptación de datos en IMS](#) para obtener más detalles.

Quién puede utilizar la función de encriptación de disco

- Los administradores de seguridad (usuarios con derechos de Security Administrator) pueden otorgar derechos de acceso del KMS a EVS para utilizar la encriptación de disco.
- Cuando un usuario común que no tiene derechos de Security Administrator necesita utilizar la función de encriptación de disco, la condición varía en función de si el usuario es el primero en la región o proyecto actual en utilizar esta función.
 - Si el usuario es el primero, debe ponerse en contacto con un usuario que tenga derechos de Security Administrator para otorgar los derechos de acceso del KMS a EVS. Luego, el usuario puede utilizar la función de encriptación de disco.
 - Si el usuario no es el primero, puede utilizar directamente la función de encriptación del disco.

Desde la perspectiva de un tenant, siempre que se hayan otorgado los derechos de acceso del KMS a EVS en una región, todos los usuarios de la misma región pueden utilizar directamente la función de encriptación de disco.

Si hay varios proyectos en la región actual, los derechos de acceso de KMS deben concederse a cada proyecto en esta región.

Claves utilizadas para la encriptación de disco de EVS

Las claves proporcionadas por KMS para la encriptación de disco incluyen Clave maestra predeterminada y Customer Master Keys (CMK).

- **Clave maestra predeterminada:** Una clave creada automáticamente por EVS a través de KMS y denominada **evs/default**.
La clave maestra predeterminada no se puede deshabilitar y no admite la eliminación programada.
- **CMK:** Claves creadas por los usuarios. Puede utilizar CMK existentes o crear una nueva. Para obtener más detalles, consulte [Creación de una CMK](#).

Si los discos se cifran mediante un CMK, que luego se deshabilita o se programa para su eliminación, los discos ya no se pueden leer ni escribir en, y es posible que los datos de estos discos nunca se restablezcan. Consulte [Tabla 1-2](#) para obtener más información.

Tabla 1-2 Impacto en los discos encriptados tras la indisponibilidad de una CMK

Estado de CMK	Impacto en los discos encriptados	Método de restauración
Desactivado	<ul style="list-style-type: none"> ● Si se conecta un disco encriptado a un ECS, el disco aún se puede usar, pero las operaciones normales de lectura/escritura no están garantizadas de forma permanente. ● Si se desconecta un disco encriptado, se producirá un error al volver a conectarlo. 	Habilite el CMK. Para obtener más información, consulte Habilitación de uno o más CMK .
Pendiente de eliminación		Cancele la eliminación programada para la CMK. Para obtener más información, consulte Cancelación de la eliminación programada de uno o más CMK .
Eliminado		Los datos de los discos nunca se pueden restaurar.

AVISO

Se le cobrará por las CMK que utilice. Si se utilizan claves básicas, asegúrese de que el saldo de su cuenta sea suficiente. Si se utilizan claves profesionales, renueve su pedido a tiempo. De lo contrario, sus servicios podrían verse interrumpidos y sus datos podrían no restablecerse nunca, ya que los discos encriptados se vuelven ilegibles y no se pueden escribir.

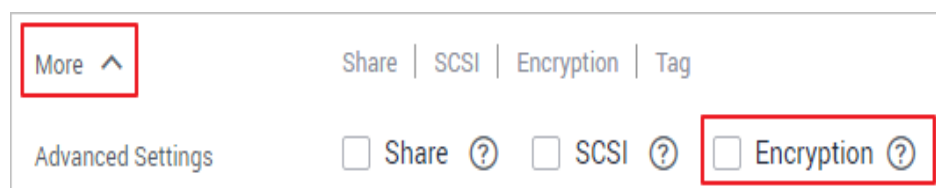
Uso de KMS para encriptar un disco (en la consola)

Paso 1 En la consola de gestión de EVS, haga clic en **Buy Disk**.

Paso 2 Seleccione la casilla de verificación **Encryption**.

1. Haga clic en **More**. Aparecerá la casilla de verificación **Encryption**.

Figura 1-8 Otro más



2. Cree una delegación.

Seleccione **Encrypt**. Si EVS no tiene autorización para acceder a KMS, aparece el cuadro de diálogo **Create Agency**. En este caso, haga clic en **Yes** para autorizarlo. Después de la autorización, EVS puede obtener claves de KMS para cifrar y descifrar discos.

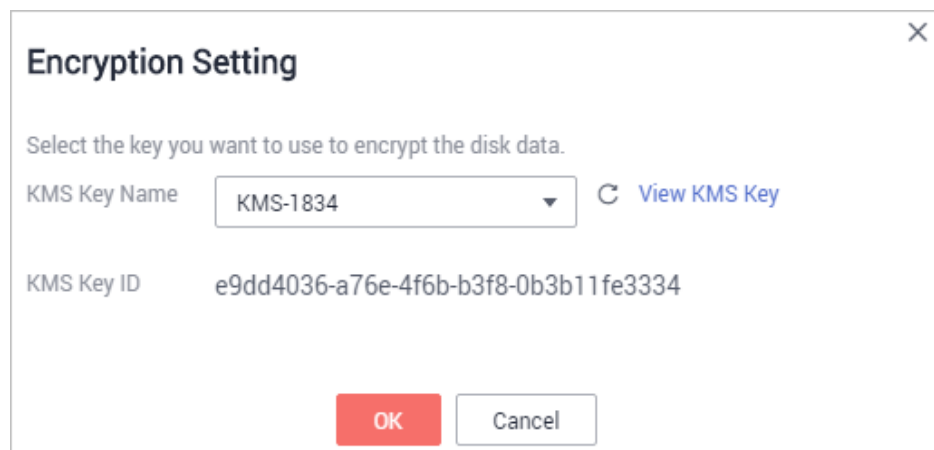
 **NOTA**

Antes de utilizar la función de encriptación de disco, los derechos de acceso del KMS deben concederse a EVS. Si tiene el derecho de concesión, otorgue los derechos de acceso del KMS a EVS directamente. Si no tiene el derecho, póngase en contacto con un usuario con derechos de Security Administrator para conceder los derechos de acceso de KMS a EVS y, a continuación, repita las operaciones anteriores.

3. Configure los parámetros de encriptación.

Seleccione **Encrypt**. Si la autorización se realiza correctamente, aparecerá el cuadro de diálogo de configuración de encriptación.

Figura 1-9 Configuración de encriptación



Seleccione uno de los siguientes tipos de claves en la lista desplegable **KMS Key Name**:

- Clave principal predeterminada. Una vez otorgados los derechos de acceso de KMS a EVS, el sistema crea automáticamente una clave principal predeterminada denominada **evs/default**.
- Una CMK existente o nueva. Para obtener más detalles sobre cómo crearla, consulte [Creación de una CMK](#).

Paso 3 Configure otros parámetros para el disco. Para obtener más detalles sobre los parámetros, consulte [Compra de un disco de EVS](#).

----Fin

Uso de KMS para encriptar un disco (con una API)

Puede invocar a la API requerida de EVS para comprar un disco de EVS cifrado. Para obtener más información, consulte la *Referencia de las API de Elastic Volume Service*.

1.2.5 Encriptación de datos en IMS

Puede crear una imagen encriptada en Image Management Service (IMS) para almacenar datos de forma segura.

Restricciones

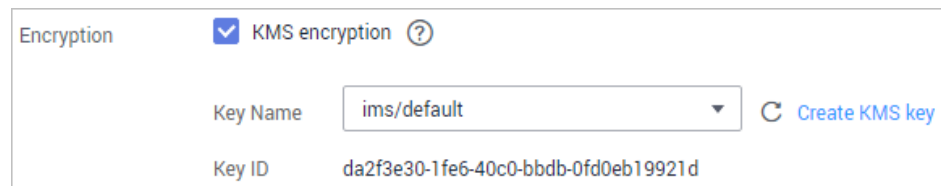
- DEW debe estar habilitado.
- No se puede compartir una imagen encriptada con otros usuarios.
- No se puede publicar una imagen encriptada en Marketplace.
- Si un ECS tiene un disco de sistema encriptado, la imagen privada creada con el ECS también se encripta.
- La clave utilizada para cifrar una imagen no se puede cambiar.
- Si la clave utilizada para cifrar una imagen está desactivada o eliminada, la imagen no estará disponible.
- El disco del sistema de un ECS creado con una imagen encriptada también está encriptado, y su clave es la misma que la clave de imagen.

Uso de KMS para encriptar una imagen privada (en la consola)

Puede crear una imagen encriptada con un ECS encriptado o con un archivo de imagen externo.

- Cree una imagen encriptada con un ECS encriptado.
Cuando se utiliza un ECS para crear una imagen privada, si el disco del sistema del ECS está cifrado, la imagen privada creada con este ECS también está encriptada. La clave utilizada para cifrar la imagen es la que se utiliza para crear el disco del sistema.
- Cree una imagen cifrada con un archivo de imagen externo.
Cuando utiliza un archivo de imagen externo que se ha cargado en un bucket de OBS para crear una imagen privada, puede seleccionar la encriptación de KMS al registrar la imagen para encriptarla.
Al cargar un archivo de imagen, puede seleccionar **KMS encryption** y utilizar una clave proporcionada por KMS para encriptar el archivo cargado, como se muestra en [Figura 1-10](#).
 - a. En la consola de gestión de IMS, haga clic en **Create Private Image**.
 - b. Configure **Type** como **System disk image**.
 - c. Configure **Source** como **Image File**.
 - d. Seleccione **KMS encryption**.

Figura 1-10 Encriptación de datos en IMS



Encryption	<input checked="" type="checkbox"/> KMS encryption ?
Key Name	ims/default ▼ ↻ Create KMS key
Key ID	da2f3e30-1fe6-40c0-bbdb-0fd0eb19921d

Seleccione uno de los siguientes tipos de claves en la lista desplegable **Key Name**:

- Clave principal predeterminada **ims/default** creada por KMS
 - Una CMK existente o nueva. Para obtener más detalles sobre cómo crearla, consulte [Creación de una CMK](#).
- e. Configure otros parámetros. Para obtener más detalles sobre los parámetros, consulte [Registro de una imagen](#).

Uso de KMS para encriptar una imagen privada (con una API)

Puede invocar a la API requerida de IMS para cifrar el archivo de imagen. Para obtener más información, consulta la *Referencia de las API de Image Management Service*.

1.2.6 Encriptación de una instancia de BD de RDS

Descripción general

Relational Database Service (RDS) soporta los motores de MySQL y PostgreSQL.

Una vez habilitada la encriptación, los datos del disco se cifrarán y almacenarán en el servidor cuando cree una instancia de BD o amplíe la capacidad del disco. Cuando descargue objetos cifrados, los datos cifrados se descifrarán en el servidor y se mostrarán en el texto plano.

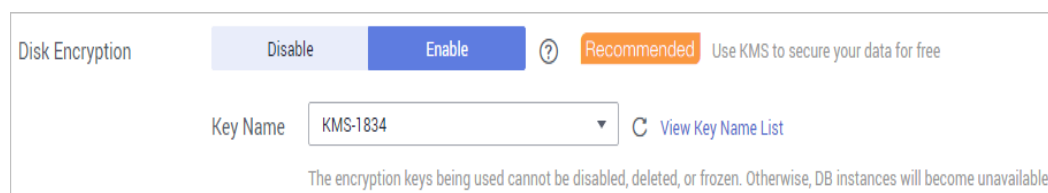
Restricciones

- El derecho de KMS Administrator debe otorgarse al usuario en la región de RDS con Identity and Access Management (IAM). Para obtener más detalles sobre cómo asignar permisos a grupos de usuarios, consulte "¿Cómo gestiono grupos de usuarios y les otorgo permisos?" en la *Guía de usuario de Identity and Access Management*.
- Para utilizar una clave definida por el usuario para encriptar los objetos que se van a cargar, cree una clave con DEW. Para obtener más detalles, consulte [Creación de una CMK](#).
- Una vez habilitada la función de encriptación del disco, no puede deshabilitarla ni cambiar la clave después de crear una instancia de BD. Los datos de respaldo almacenados en OBS no se cifrarán.
- Una vez creada una instancia de BD de RDS, no deshabilite ni elimine la clave que se está utilizando. De lo contrario, RDS no estará disponible y no se podrán restaurar los datos.
- Si escala verticalmente una instancia de base de datos con discos encriptados, el espacio de almacenamiento ampliado se cifrará con la clave de encriptación original.

Uso de KMS para encriptar una instancia de BD (en la consola)

Cuando un usuario compra una instancia de base de datos del Servicio de base de datos relacional (RDS), el usuario puede seleccionar **Disk encryption** y utilizar la clave proporcionada por KMS para cifrar el disco de la instancia de base de datos. Para obtener más información, consulte [Compra de una instancia de BD de MySQL](#) y [Compra de una instancia de BD de PostgreSQL](#).

Figura 1-11 Encriptación de datos en RDS



Uso de KMS para encriptar una instancia de BD (Con una API)

También puede invocar a la API requerida de RDS para adquirir instancias de BD cifradas. Consulte la *Referencia de las API de Relational Database Service* para obtener más detalles.

1.2.7 Encriptación de una instancia de BD de DDS

Descripción general

Una vez habilitada la encriptación, los datos del disco se cifrarán y almacenarán en el servidor cuando cree una instancia de BD o amplíe la capacidad del disco. Cuando descargue objetos cifrados, los datos cifrados se descifrarán en el servidor y se mostrarán en el texto plano.

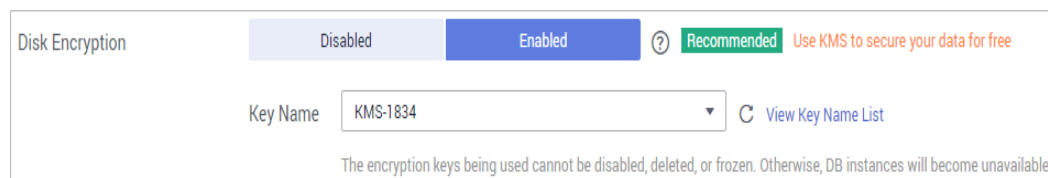
Restricciones

- Se debe agregar el derecho de KMS Administrator en la región de RDS con IAM. Para obtener más detalles sobre cómo asignar valores a grupos de usuarios, consulte "¿Cómo gestiono grupos de usuarios y les otorgo permisos?" en la *Guía de usuario de Identity and Access Management*.
- Para utilizar una clave definida por el usuario para encriptar los objetos que se van a cargar, cree una clave con DEW. Para obtener más detalles, consulte [Creación de una CMK](#).
- Una vez habilitada la función de encriptación del disco, no puede deshabilitarla ni cambiar la clave después de crear una instancia de BD. Los datos de respaldo almacenados en OBS no se cifrarán.
- Después de crear una instancia de BD de Document Database Service (DDS), no deshabilite ni elimine la clave que se está utilizando. De lo contrario, DDS se activará y no se podrán restaurar los datos.
- Si escala verticalmente una instancia de base de datos con discos encriptados, el espacio de almacenamiento ampliado se cifrará con la clave de encriptación original.

Uso de KMS para encriptar una instancia de BD (en la consola)

Al comprar una instancia de BD en DDS, puede configurar **Disk Encryption** en **Enable** y usar la clave proporcionada por KMS para cifrar el disco de la instancia de BD. Para obtener más información, consulte [Compra de una instancia de clúster](#).

Figura 1-12 Encriptación de datos en DDS



Uso de KMS para encriptar una instancia de BD (Con una API)

También puede invocar a la API requerida de DDS para comprar instancias de base de datos cifradas. Para obtener más información, consulta la *Referencia de las API de Document Database Service*.

1.3 Uso del SDK de encriptación para cifrar y descifrar datos locales

Puede utilizar determinados algoritmos para encriptar sus archivos, protegiéndolos de posibles violaciones o manipulaciones.

El **SDK de encriptación** es una biblioteca de contraseñas de cliente que puede cifrar y descifrar datos y flujos de archivos. Puede cifrar y descifrar grandes cantidades de datos simplemente invocando a las API. Le permite concentrarse en el desarrollo de las funciones principales de sus aplicaciones sin distraerse con los procesos de encriptación y descifrado de datos.

NOTA

Para obtener más detalles, consulte [Detalles](#).

Escenario

Si se envían archivos e imágenes de gran tamaño a KMS por HTTPS para su encriptación, se consumirá un gran número de recursos de red y la encriptación será lenta. En esta sección se describe cómo cifrar rápidamente una gran cantidad de datos.

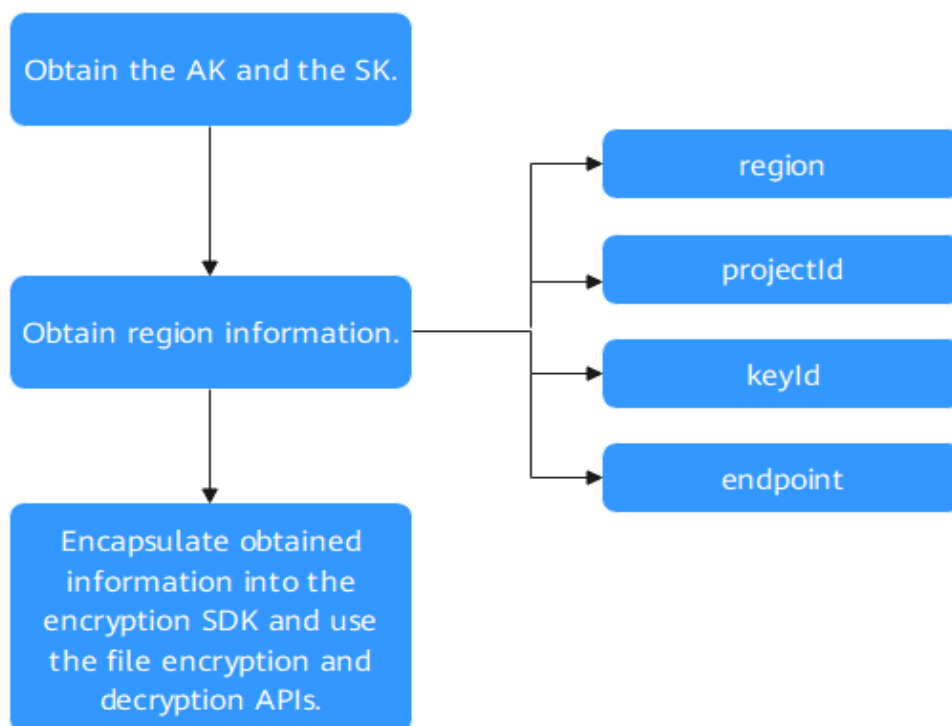
Solución

El SDK de encriptación realiza la encriptación en sobres en flujos de archivos segmento por segmento.

Los datos se cifran dentro del SDK utilizando la DEK generada por KMS. La encriptación segmentada de archivos en la memoria garantiza la seguridad y la corrección de la encriptación de archivos, ya que no requiere la transferencia de archivos por la red.

El SDK carga un archivo en la memoria y lo procesa segmento por segmento. El siguiente segmento no se leerá antes de que finalice la encriptación o el descifrado del segmento actual.

Proceso



Procedimiento

Paso 1 Obtenga las AK y SK.

- **ACCESS_KEY**: Clave de acceso de la cuenta de Huawei. Para obtener más detalles, consulte [¿Cómo obtengo una clave de acceso \(AK/SK\)?](#)
- **SECRET_ACCESS_KEY**: Clave de acceso secreta de la cuenta de Huawei. Para obtener más detalles, consulte [¿Cómo obtengo una clave de acceso \(AK/SK\)?](#)
- Habrá riesgos de seguridad si la AK/SK utilizada para la autenticación se escribe directamente en el código. Cifre la AK/SK en el archivo de configuración o en las variables de entorno para su almacenamiento.
- En este ejemplo, las AK/SK almacenadas en las variables de entorno se utilizan para la autenticación de identidad. Primero configure las variables de entorno **HUAWEICLOUD_SDK_AK** y **HUAWEICLOUD_SDK_SK** en el entorno local.

Paso 2 Obtenga información de la región.

1. **Inicie sesión en la consola de gestión.**
2. Pase el ratón por encima del nombre de usuario en la esquina superior derecha y seleccione **My Credentials** en la lista desplegable.
3. Obtenga los **Project ID** y **Project Name**.

Figura 1-13 Obtención del ID y el nombre del proyecto

Project ID	Project Name	Region
0d02edc16	10ea865171161f	cn-north-7
9de3619d	52bc216af5db1	cn-north-1
a52e7b97d	3e52448243717bc	cn-north-4
e40af14bc	031a966bc103b	cn-north-2
60537a041	363558184ab9978	cn-north-9


- Haga clic en . Elija **Security & Compliance > Data Encryption Workshop**.
- Obtenga el ID de la CMK (**KEYID**) que se utilizará en la región actual.

Figura 1-14 Obtención del ID de CMK

AliasID	Status	Key Algorithm and Usage	Origin	Enterprise Project	Operation
KMS-e60 88ea2960-719b-409c-ab3c-aad2f493a992	Enabled	AES_256 ENCRYPT_DECRYPT	Key Management Service	default	Disable Delete Add to Project
KMS-41e bc777738-161c-4844-81ce-ac1f5c9dd6d	Enabled	RSA_4096 ENCRYPT_DECRYPT	Key Management Service	DEW	Disable Delete Add to Project

- Obtenga el punto de conexión (**ENDPOINT**) requerido por la región actual.
Un punto de conexión es el **request address** para llamar a una API. Los puntos de conexión varían según los servicios y las regiones. Para ver los puntos de conexión de todos los servicios, consulte [Regiones y puntos de conexión](#).

Figura 1-15 Obtención de un punto de conexión

Region Name	Region	Endpoint	Protocol Type
AP	ap	kms.myhuaweicloud.com	HTTPS
CN	cn	kms.myhuaweicloud.com	HTTPS
CN	cn	kms.myhuaweicloud.com	HTTPS
CN	cn	kms.myhuaweicloud.com	HTTPS
CN	cn	kms.myhuaweicloud.com	HTTPS
CN	cn	kms.myhuaweicloud.com	HTTPS
CN	cn	kms.myhuaweicloud.com	HTTPS
CN	cn	kms.myhuaweicloud.com	HTTPS

Paso 3 Encriptar y descifrar un archivo.

```
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();
}
}
```

----Fin

1.4 Encriptación y descifrado de datos con recuperación ante desastres entre regiones

Escenario

Si se produce una falla durante la encriptación o el descifrado en una región, puede utilizar KMS para implementar la encriptación y el descifrado de recuperación ante desastres entre regiones, lo que garantiza la continuidad del servicio.

Solución

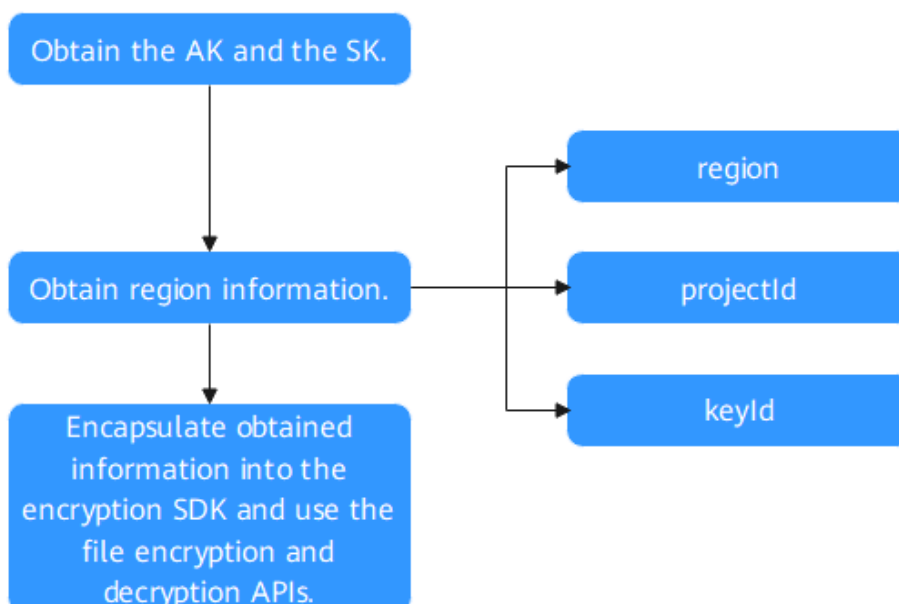
Si KMS presenta fallas en una o varias regiones, la encriptación y el descifrado se pueden completar siempre que haya disponible una clave en el anillo de claves.

Un anillo de claves entre regiones puede utilizar las CMK de varias regiones para cifrar un dato y generar texto cifrado de datos único. Para descifrar los datos, basta con utilizar un anillo de claves que contenga una o más CMK disponibles que se utilizaron para cifrar los datos.

📖 NOTA

Para obtener más información, consulte [Detalles](#).

Proceso



Procedimiento

Paso 1 Obtenga las AK y SK.

- **ACCESS_KEY**: Clave de acceso de la cuenta de Huawei. Para obtener más detalles, consulte [¿Cómo obtengo una clave de acceso \(AK/SK\)?](#)
- **SECRET_ACCESS_KEY**: Clave de acceso secreta de la cuenta de Huawei. Para obtener más detalles, consulte [¿Cómo obtengo una clave de acceso \(AK/SK\)?](#)
- Habrá riesgos de seguridad si la AK/SK utilizada para la autenticación se escribe directamente en el código. Cifre la AK/SK en el archivo de configuración o en las variables de entorno para su almacenamiento.
- En este ejemplo, las AK/SK almacenadas en las variables de entorno se utilizan para la autenticación de identidad. Primero configure las variables de entorno **HUAWEICLOUD_SDK_AK** y **HUAWEICLOUD_SDK_SK** en el entorno local.

Paso 2 Obtenga información de la región.

1. **Inicie sesión en la consola de gestión.**
2. Pase el ratón por encima del nombre de usuario en la esquina superior derecha y seleccione **My Credentials** en la lista desplegable.
3. Obtenga los **Project ID** y **Project Name**.

Figura 1-16 Obtención del ID y el nombre del proyecto

Project ID	Project Name	Region
0d22edc16	10ea865171161f	cn-north-7
9de3816d1	52bc219af5db1	cn-north-1
a52e7697d	7e62d48243717bc	cn-north-4
e40af114bc	831a098bc103b	cn-north-2
6d537a841	763558184ae9978	cn-north-9


4. Haga clic en . Elija **Security & Compliance > Data Encryption Workshop**.
5. Obtenga el ID de la CMK (**KEYID**) que se utilizará en la región actual.

Figura 1-17 Obtención del ID de CMK

AliasID	Status	Key Algorithm and Usage	Origin	Enterprise Project	Operation
KMS-4800 8de22900-719b-409c-ab0c-aad2f493a992	Enabled	AES_256 ENCRYPT_DECRYPT	Key Management Service	default	Disable Delete Add to Project
KMS-411e bc777738-161c-4844-81ce-ac1f5c9d026d	Enabled	RSA_4096 ENCRYPT_DECRYPT	Key Management Service	DEW	Disable Delete Add to Project

Paso 3 Utilice el anillo de claves para encriptar y desencriptar.

```
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>";

    private static final String PROJECT_ID_2 = "<projectId2>";
    private static final String REGION_2 = "<region2>";
    private 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()));
}
}

```

----Fin

1.5 Uso de KMS para proteger la integridad de archivos

Escenario

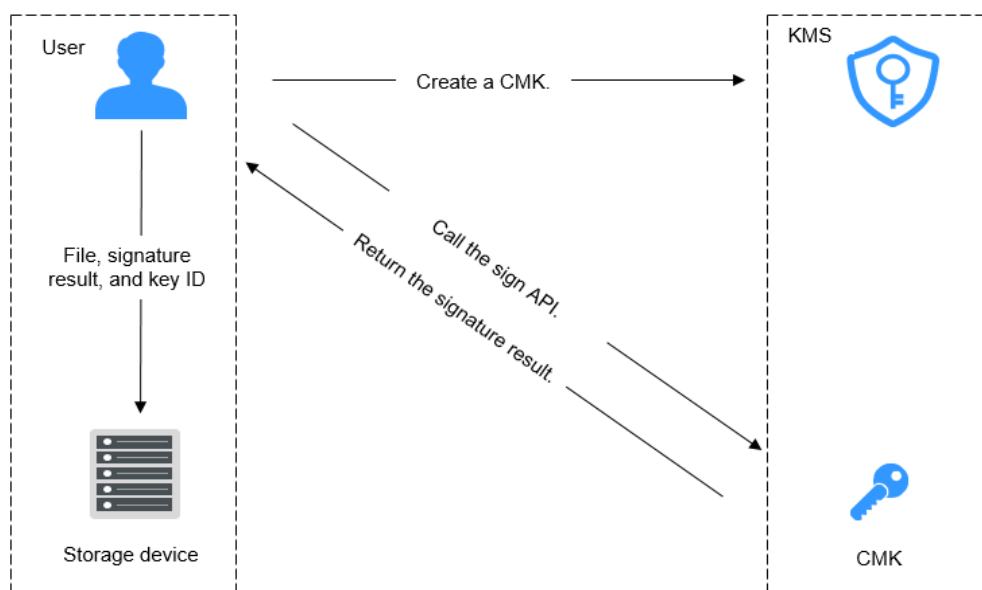
Cuando sea necesario transmitir o almacenar de forma segura una gran cantidad de archivos (como imágenes, pólizas de seguros electrónicas y archivos importantes), puede utilizar KMS para firmar el compendio de archivos. Cuando se vuelvan a utilizar los archivos, podrá volver a calcular el compendio para verificar la firma. Asegúrese de que los archivos no sean manipulados durante la transmisión o el almacenamiento.

Solución

Crear una CMK en KMS.

Realice el compendio del archivo e invoque a la API de signos de KMS para firmar el compendio. Se obtiene el resultado de firma del compendio. Transmita o almacene el resultado de la firma de compendio, el ID de clave y el archivo juntos. La siguiente figura muestra el proceso de firma.

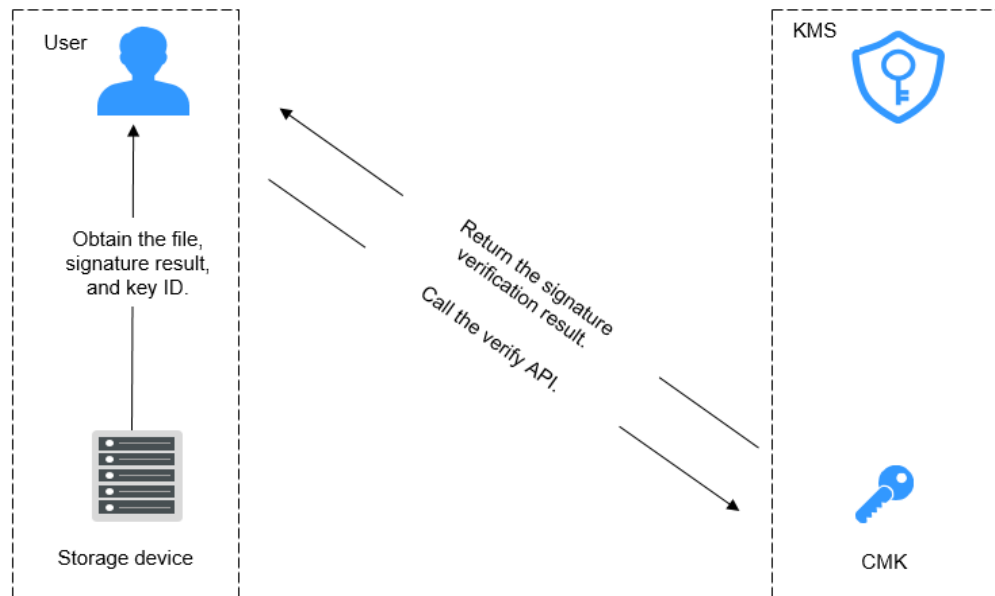
Figura 1-18 proceso de firma



Antes de utilizar un archivo, es necesario comprobar su integridad para asegurarse de que no ha sido manipulado.

Vuelva a calcular el compendio del archivo e invoque a la API de verificación de KMS con el valor de firma para verificar la firma del compendio. Se obtiene el resultado de la verificación de firmas. Si se verifica la firma, el archivo no ha sido manipulado. La siguiente figura muestra el proceso de verificación de firmas.

Figura 1-19 Proceso de verificación de firma.



Procedimiento

Paso 1 Obtenga las AK y SK.

- **ACCESS_KEY**: Clave de acceso de la cuenta de Huawei. Para obtener más detalles, consulte [¿Cómo obtengo una clave de acceso \(AK/SK\)?](#)
- **SECRET_ACCESS_KEY**: Clave de acceso secreta de la cuenta de Huawei. Para obtener más detalles, consulte [¿Cómo obtengo una clave de acceso \(AK/SK\)?](#)
- Habrá riesgos de seguridad si la AK/SK utilizada para la autenticación se escribe directamente en el código. Cifre la AK/SK en el archivo de configuración o en las variables de entorno para su almacenamiento.
- En este ejemplo, las AK/SK almacenadas en las variables de entorno se utilizan para la autenticación de identidad. Primero configure las variables de entorno **HUAWEICLOUD_SDK_AK** y **HUAWEICLOUD_SDK_SK** en el entorno local.

Paso 2 Obtenga información de la región.

- [Inicie sesión en la consola de gestión.](#)

Paso 3 Utilice KMS para firmar el archivo y verificar la firma.

```
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_ACCESS_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.SigningAlgorithmEnum.RSASSA_PSS_SHA_256)
            .withMessageType(SignRequestBody.MessageTypeEnum.DIGEST).withMessage(fileSha256Sum));
    final SignResponse signResponse = kmsClient.sign(signRequest);
    // 5. Verify the digest.
    final ValidateSignatureRequest validateSignatureRequest = new ValidateSignatureRequest().withBody(
        new VerifyRequestBody().withKeyId(keyId).withMessage(fileSha256Sum).withSignature(signResponse.getSignature())
            .withSigningAlgorithm(VerifyRequestBody.SigningAlgorithmEnum.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());
}
}
```

----Fin

2 Cloud Secret Management Service

2.1 Uso de CSMS para cambiar las contraseñas de cuenta de base de datos codificadas de forma rígida

Por lo general, los secretos utilizados para el acceso están integrados en las aplicaciones. Para actualizar un secreto, debe crear un nuevo secreto y dedicar tiempo a actualizar sus aplicaciones. Si tiene varias aplicaciones que utilizan el mismo secreto, debe actualizarlas todas; de lo contrario, las aplicaciones que olvidó actualizar no podrán utilizar el secreto para iniciar sesión.

Una herramienta para la gestión de secretos fácil de usar, eficaz y segura será útil.

Cloud Secret Management Service (CSMS) tiene las siguientes ventajas:

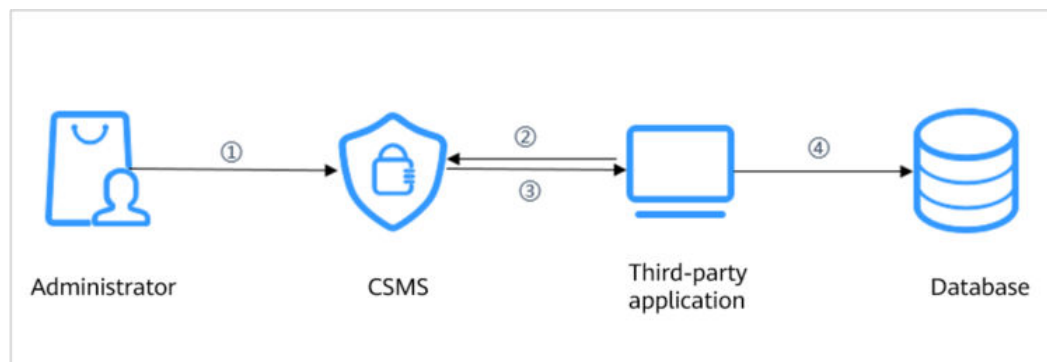
- Puede alojar sus secretos en lugar de utilizar secretos codificados de forma rígida, lo que mejora la seguridad de los datos y los activos.
- Sus servicios no se ven afectados cuando **gira manualmente** los secretos.
- El acceso de SDK seguro le permite invocar dinámicamente a sus secretos.
- Puede almacenar muchos tipos de secretos. Puede almacenar cuentas de servicio, contraseñas e información de bases de datos, incluidos, entre otros, nombres de bases de datos, direcciones IP y números de puerto.

Inicio de sesión en una base de datos con secretos

Puede crear un secreto e iniciar sesión en su base de datos invocando al secreto con una API.

Asegúrese de que su cuenta tenga permiso de KMS Administrator o de KMS CMKFullAccess. Para obtener más detalles, consulte [Gestión de permisos de DEW](#).

Figura 2-1 Proceso de inicio de sesión basado en secreto



El proceso es el siguiente:

- Paso 1** Cree un secreto en la [consola](#) o con una [API](#) para almacenar información de la base de datos (como la dirección de la base de datos, el puerto y la contraseña).
- Paso 2** Utilice una aplicación para acceder a la base de datos. CSMS consultará el secreto creado en [1](#).
- Paso 3** CSMS recupera y descifra el texto cifrado del secreto y devuelve de forma segura la información almacenada en el secreto a la aplicación a través de la API de gestión de secretos.
- Paso 4** La aplicación obtiene el secreto de texto plano descifrado y lo utiliza para acceder a la base de datos.

----Fin

API para crear y consultar secretos

Puede invocar a las siguientes API para crear secretos, guardar su contenido y consultar información secreta.

API	Descripción
Creación de un secreto	Esta API se utiliza para crear un secreto y almacenar el valor secreto en la versión secreta inicial.
Consulta de un secreto	Esta API se utiliza para consultar un secreto.

Creación y consulta de secretos con API

1. Prepare la información básica de autenticación.
 - **ACCESS_KEY**: Clave de acceso de la cuenta de Huawei
 - **SECRET_ACCESS_KEY**: Clave de acceso secreta de la cuenta de Huawei
 - **PROJECT_ID**: ID de proyecto de un sitio de Huawei Cloud. Para obtener más detalles, véase [Proyecto](#).
 - **CSMS_ENDPOINT**: punto de conexión para acceder al CSMS. Consulte [Puntos de conexión](#) para obtener más detalles.

- Habrá riesgos de seguridad si la AK/SK utilizada para la autenticación se escribe directamente en el código. Cifre la AK/SK en el archivo de configuración o en las variables de entorno para su almacenamiento.
- En este ejemplo, las AK/SK almacenadas en las variables de entorno se utilizan para la autenticación de identidad. Primero configure las variables de entorno **HUAWEICLOUD_SDK_AK** y **HUAWEICLOUD_SDK_SK** en el entorno local.

2. Crear y consultar información del secreto.

Nombre del secreto: **secretName**

Valor del secreto: **secretString**

Valor de la versión del secreto: **LATEST_SECRET**

Versión del secreto: **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/en-us/productdesc-iam/iam\_01\_0023.html
     * - CSMS_ENDPOINT: endpoint address for accessing CSMS. For details, see https://support.huaweicloud.com/intl/en-us/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();
}
}
```

Obtención de la cuenta de base de datos a través de una aplicación

1. Obtenga la declaración de dependencia del SDK de CSMS.

Ejemplo:

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

2. Establezca una conexión de base de datos y obtenga la cuenta.

Ejemplo:

```
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 de CSMS para evitar fugas de AK y SK

O CSMS é um serviço de hospedagem do segredo 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 do segredo. O CSMS pode ajudá-lo a eliminar os riscos incorridos pela codificação rígida, configuração de texto não criptografado e abuso de permissão.

Escenario

Los secretos de aplicación se almacenan y se puede acceder a ellos temporalmente para evitar fugas de AK y SK.

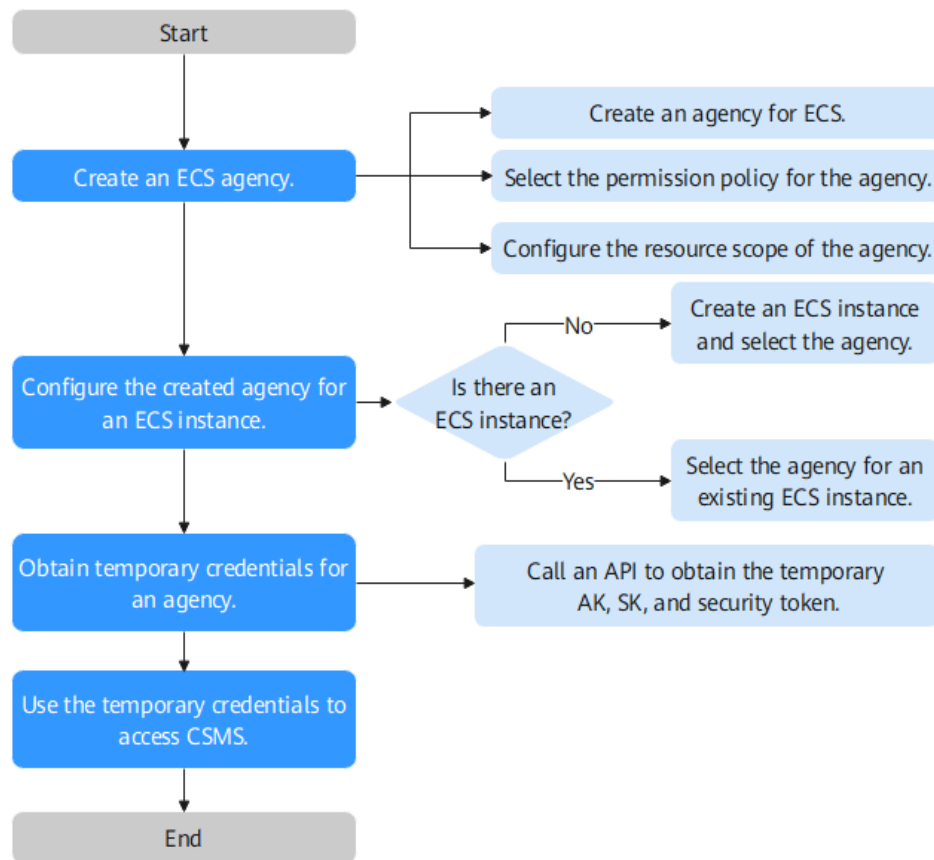
Cómo funciona

Puede utilizar Identity and Access Management (IAM) para obtener claves de acceso temporales para Elastic Cloud Server (ECS) con el fin de proteger las AK y SK.

Los secretos de acceso pueden clasificarse en secretos permanentes y secretos temporales según sus períodos de validez. Los secretos de acceso permanentes incluyen nombres de usuario y contraseñas. Las claves de acceso tienen un período de validez más corto, se actualizan con frecuencia, por lo que son más seguras. Puede asignar una delegación de IAM a una instancia de ECS, de modo que las aplicaciones de la instancia de ECS puedan utilizar las AK y SK temporales con el token de seguridad para acceder al CSMS. Las claves de acceso temporales se obtienen dinámicamente cada vez que se requieren. También pueden almacenarse en memoria caché y actualizarse periódicamente.

Flujo del proceso

Figura 2-2 Proceso de configuración de agencias de ECS



Restricción

Solo el administrador o un usuario de IAM con permiso de ECS puede configurar una delegación para una instancia de ECS.

Procedimiento

Paso 1 Cree una delegación de ECS en IAM.


1. **Inicie sesión en la consola de gestión.**
2. Haga clic en  a la izquierda de la página y elija **Management & Governance > Identity and Access Management**. Se aparecerá la página **Users**.
3. En el panel de navegación, seleccione **Agencies**.
4. Haga clic en **Create Agency** en la esquina superior derecha.
5. Configure los parámetros en el cuadro de diálogo **Create Agency**. Para obtener más información, consulte **Parámetros de delegación**.

Figura 2-3 Creación de una delegación

The screenshot shows a form for creating a delegation. It contains the following fields and options:

- * Agency Name:** A text input field.
- * Agency Type:** Two radio button options:
 - Account: Delegate another HUAWEI CLOUD account to perform operations on your resources.
 - Cloud service: Delegate a cloud service to access your resources in other cloud services.
- * Cloud Service:** A dropdown menu with the selected option "Elastic Cloud Server (ECS) and Bare Metal Serv...".
- * Validity Period:** A dropdown menu with the selected option "Unlimited".
- Description:** A text area with the placeholder "Enter a brief description." and a character count of "0/255".
- Buttons:** A red "Next" button and a white "Cancel" button.

Tabla 2-1 Parámetros de delegación

Nombre del parámetro	Descripción
Nombre de la delegación	Ingrese el nombre de una delegación. Ejemplo: ECS_TO_CSMS
Tipo de delegación	Seleccione Cloud service .
Servicio en la nube	Seleccione Elastic Cloud Server (ECS) and Bare Metal Server (BMS) .
Período de validez	Seleccione una duración. El valor puede ser Unlimited, 1 day o Custom .
Descripción	(Opcional) Introduzca la descripción de la delegación.

6. Haga clic en **Next** para ir a la página de autorización.
7. Haga clic en **Create Policy** en la esquina superior derecha. Si ya tiene una política, siga este paso.
 - a. Configure los parámetros de políticas. Para obtener más información, consulte [Parámetros de política](#).

Figura 2-4 Creación de una política

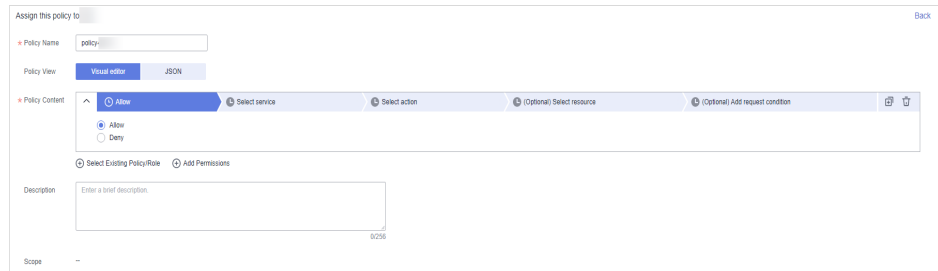


Tabla 2-2 Parámetros de política

Nombre del parámetro	Descripción
Nombre de política	Introduzca un nombre de política.
Policy View	Seleccione Visual editor .
Policy Content	<ul style="list-style-type: none"> ■ Allow: seleccione Allow. ■ Select service: seleccione Cloud Secret Management Service (CSMS). ■ Select action: seleccione los permisos de lectura y de escritura según sea necesario. ■ (Optional) Select resource: seleccione el ámbito de los recursos. <ul style="list-style-type: none"> ○ Specific: acceda a los secretos específicos. <p>NOTA Puede seleccionar Specify resource path y luego hacer clic en Add Resource Path para especificar un secreto accesible.</p> <ul style="list-style-type: none"> ○ All: acceda a todos los secretos. ■ (Optional) Add request condition: haga clic en Add Request Condition, seleccione una clave de condición y un operador e introduzca los valores necesarios.
Description	(Opcional) Introduzca la descripción de la política.

8. Seleccione una política para la delegación. Haga clic en **Next**.
9. Seleccione un ámbito y haga clic en **OK**.
 - **All resources**: Los usuarios de IAM podrán utilizar todos los recursos, incluidos los de proyectos empresariales, proyectos específicos de la región y servicios globales de su cuenta según los recursos asignados.
 - **Enterprise projects**: El valor seleccionado se aplicará a los recursos de los proyectos empresariales que seleccione.
 - **Region-specific projects**: Los permisos seleccionados se aplicarán a los recursos de los proyectos específicos de la región que seleccione.

Paso 2 Asigne una delegación (por ejemplo, **ECS_TO_CSMS**) a una instancia de ECS.



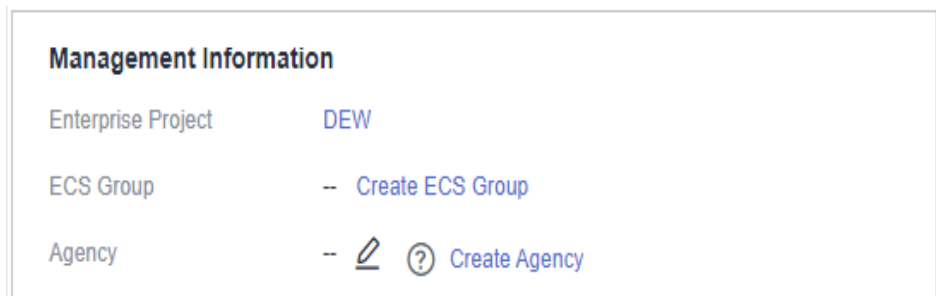
- Para crear una instancia de ECS, realice las operaciones descritas en [Creación de un ECS](#). En el [Paso 3: Configurar ajustes avanzados](#), seleccione la nueva delegación (por ejemplo, `ECS_TO_CSMS`).
- Para utilizar una instancia de ECS existente, lleve a cabo los siguientes pasos:
 - a. Haga clic en  a la izquierda de la página y elija **Management & Governance > Identity and Access Management**. Vaya a la página de ECS.
 - b. Haga clic en el nombre de una instancia de ECS para ir a la página **Summary**.
 - c. En el área **Management Information**, haga clic en  y seleccione una delegación (por ejemplo, `ECS_TO_CSMS`).

Figura 2-5 Selección de una delegación



Paso 3 En una aplicación que se ejecute en la instancia de ECS, invoque a una API para obtener los secretos temporales de la delegación, incluida las AK y SK temporales con el token de seguridad, para acceder al CSMS.

1. Obtenga las AK y SK temporales (en el directorio **Security Key**). Para obtener más detalles, consulte [Obtención de metadatos](#).

- URI
`/openstack/latest/securitykey`
- Método
Solicitud GET
- Se devuelven los siguientes datos:

```
{
  "credential": {
    "access": "LDHZK30XXXXXXXXXXXXXXXXV",
    "secret": "gyqcdzVXXXXXXXXXXXXXXXXXXXXXXXXM16",
    "securitytoken": "E19FI2C65qXXXXXXXXXXXXXXXXXXXXXXXXXnkcaoV",
    "expires_at": "2022-07-14T12:09:24.147000Z"
  }
}
```

 **NOTA**

- Extraiga los valores de **access**, **secret** y **securitytoken** para acceder al CSMS.
 - ECS gira automáticamente los secretos temporales para garantizar que sean seguros y válidos.
2. Utilice las AK y SK temporales con el token de seguridad para acceder al CSMS.
 - Un ejemplo de la lista de secretos es el siguiente. Para obtener más detalles, consulte [API de gestión de secretos](#).
 - Prepare la información básica de autenticación.

- **ACCESS_KEY**: Clave de acceso de la cuenta de Huawei. Para obtener más detalles, consulte [¿Cómo obtengo una clave de acceso \(AK/SK\)?](#)
- **SECRET_ACCESS_KEY**: Clave de acceso secreta de la cuenta de Huawei. Para obtener más detalles, consulte [¿Cómo obtengo una clave de acceso \(AK/SK\)?](#)
- Habrá riesgos de seguridad si la AK/SK utilizada para la autenticación se escribe directamente en el código. Cifre la AK/SK en el archivo de configuración o en las variables de entorno para su almacenamiento.
- En este ejemplo, las AK/SK almacenadas en las variables de entorno se utilizan para la autenticación de identidad. Primero configure las variables de entorno **HUAWEICLOUD_SDK_AK** y **HUAWEICLOUD_SDK_SK** en el entorno 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/en-us/productdesc-iam/iam\_01\_0023.html
        * - CSMS_ENDPOINT: endpoint address for accessing CSMS. For details, see https://support.huaweicloud.com/intl/en-us/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.getStatusCode());
        System.out.println(e.getErrorCode());
        System.out.println(e.getErrorMsg());
    }
}
}

```

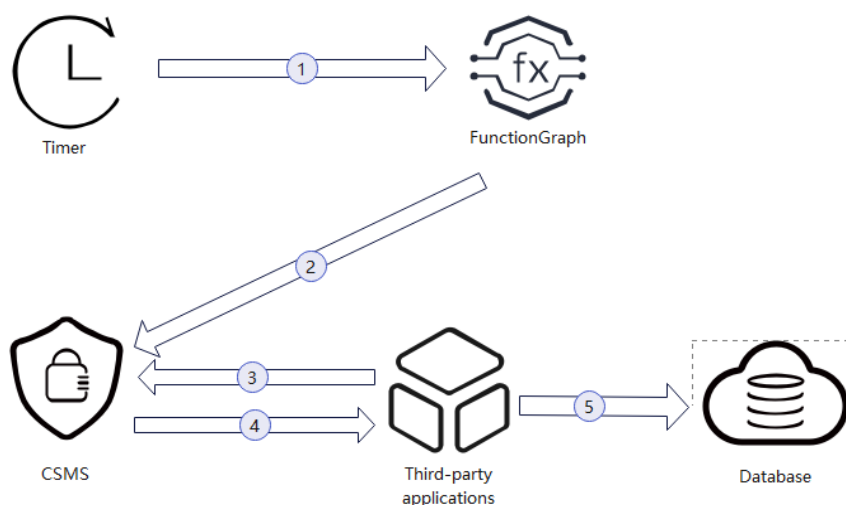
----Fin

2.3 Uso de CSMS para rotar automáticamente contraseñas de seguridad

En esta sección se describe cómo puede utilizar FunctionGraph y CSMS para generarse, alojarse y rotarse automáticamente las contraseñas seguras y conformes.

Proceso

Figura 2-6 Rotación de contraseñas



El proceso es el siguiente:

1. Cuando caduca un temporizador, se publica un evento activado programado.
2. Después de recibir el evento, FunctionGraph reemplaza el marcador de posición en la plantilla del secreto por una nueva contraseña aleatoria y la almacena en el secreto, que se considera una nueva versión del secreto.
3. Las aplicaciones invocan periódicamente a las API o los SDK para obtener la versión del secreto más reciente.
4. CSMS recupera y descifra el texto cifrado del secreto y devuelve de forma segura la información almacenada en el secreto a la aplicación a través de la API de gestión de secretos.


- Después de recibir el secreto descifrado, las aplicaciones utilizan la nueva contraseña para acceder en el futuro y la utilizan para actualizar el objeto de destino (como la base de datos o el servidor).

Restricciones

- El CSMS está disponible en la región.
- FunctionGraph está disponible en la región.

Creación de una delegación

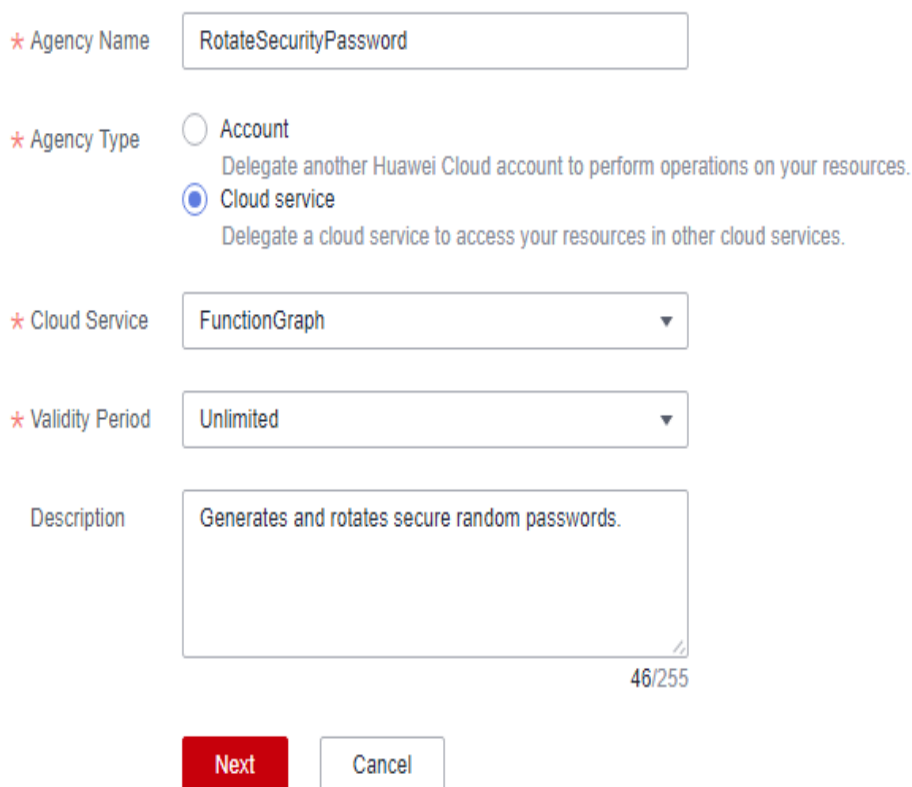
Paso 1 [Inicie sesión en la consola de gestión.](#)

Paso 2 Haga clic en  a la izquierda y seleccione **Management & Governance > Identity and Access Management** para acceder a la página **Users**.

Paso 3 En el panel de navegación situado a la izquierda, seleccione **Agencies**.

Paso 4 Haga clic en **Create Agency** y configure los parámetros como se muestra en [Figura 2-7](#). [Tabla 2-3](#) describe los parámetros.

Figura 2-7 Creación de una delegación



* Agency Name

* Agency Type Account
Delegate another Huawei Cloud account to perform operations on your resources.
 Cloud service
Delegate a cloud service to access your resources in other cloud services.

* Cloud Service

* Validity Period

Description

46/255

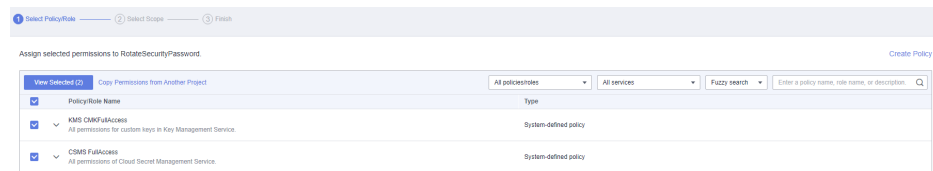
Tabla 2-3 Parámetros de delegación

Parámetro	Descripción
Agency Name	Configure este parámetro según sea necesario.
Agency Type	Seleccione Cloud service .
Cloud Service	Seleccione FunctionGraph .
Validity Period	Configure este parámetro según el escenario de aplicación de la función. Si la función debe ejecutarse durante mucho tiempo, seleccione Unlimited .
Description	Configure este parámetro según sea necesario.

Paso 5 Haga clic en **Next**.

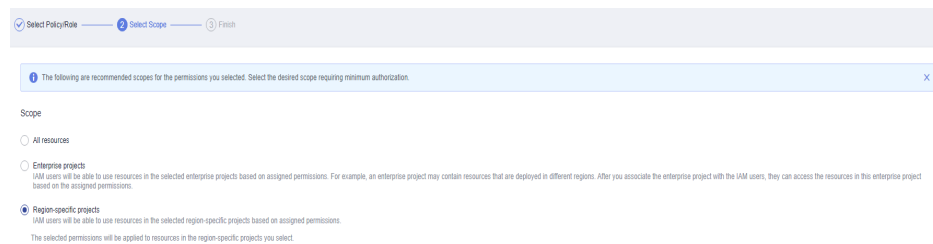
Paso 6 Seleccione **CSMS FullAccess** y **KMS CMKFullAccess**.

Figura 2-8 Selección de permisos



Paso 7 Haga clic en **Next** y seleccione un alcance basado en sus requerimientos de servicio.

Figura 2-9 Selección de un ámbito




Paso 8 Haga clic en **OK**.

----Fin

Creación de una función de rotación de contraseña

Paso 1 **Inicie sesión en la consola de gestión.**

Paso 2 Haga clic en  a la izquierda y seleccione **Compute > FunctionGraph**.

Paso 3 Haga clic en **Create Function** en la esquina superior derecha y configure los parámetros como se muestra en **Figura 2-10**. **Tabla 2-4** describe los parámetros.

Figura 2-10 Creación de una función

Basic Information

* Function Type

Event Function | HTTP Function

Processes event requests and can be triggered by APIG, OBS, and DIS events.

* Region

Region are geographic areas isolated from each other. Resources are region-specific and cannot be used across regions through internal network connections. For low network latency and quick resource access, select the nearest region.

* Project

Function Name

rotate-security-password

Enter 1 to 60 characters, starting with a letter and ending with a letter or digit. Only letters, digits, hyphens (-), and underscores (_) are allowed.

Agency

RotateSecurityPassword | Create Agency

Permission Policies

CSMS FullAccess | KMS CMKFullAccess | View Policies

These are the permission policies assigned to agency RotateSecurityPassword. To add or delete policies, go to the IAM console.

* Enterprise Project

default | View Enterprise Project

Enterprise Project Management Service (EPS) provides a unified method to manage cloud resources and personnel by enterprise project.

Runtime

Python 3.6

Select a language to compile the function. CodeArts IDE Online supports Node.js, Python, and PHP.

Tabla 2-4 Parámetros básicos

Parámetro	Descripción
Region	Seleccione la región donde se despliega la función.
Project	Seleccione el proyecto en el que se despliega la función.
Function Name	Introduzca un nombre de función personalizado.
Agency	Ingrese el nombre definido en Creación de una delegación .
Enterprise Project	Si habilitó los proyectos empresariales, seleccione uno para agregarle una función. Si no ha habilitado proyectos de empresa, este parámetro no se mostrará. Omita este paso. Para obtener más detalles sobre cómo habilitar un proyecto empresarial, consulte Habilitación del centro empresarial .
Runtime	Seleccione un idioma para escribir la función. Actualmente, se soporta Python. NOTA Solo se soportan Python de las versiones 3.6, 3.9 y 3.10.

Paso 4 Haga clic en **Create Function**.

Paso 5 Seleccione la pestaña **Configuration**. En el panel de navegación de la izquierda, seleccione **Environment Variables**. Haga clic en **Add** y agregue variables de entorno en la fila de configuración de variables. [Tabla 2-4](#) describe los parámetros. Luego, haga clic en **Save**.

Tabla 2-5 Variables del entorno

Parámetro	Descripción	Ejemplo
region	Nombre del proyecto basado en la región. Por ejemplo, si la región es CN North-Beijing4, el nombre del proyecto debe ser cn-north-4 . Para obtener su región, haga clic en su nombre de usuario en la esquina superior derecha de la página y seleccione My Credentials en la lista desplegable.	cn-north-4
secret_name	Nombre del secreto que se va a rotar NOTA Se debe haber creado un secreto. Para obtener más detalles, consulte Creación de un secreto .	rds-functionGraph-rotate
secret_content	Plantilla del secreto que se especifica con llaves ({}), por ejemplo, la plantilla del secreto es {"password":"password_placeholder"}. En este caso, password_placeholder es el marcador de posición, que será reemplazado por una contraseña segura una vez ejecutada la función. El nuevo contenido se guardará en secreto después del reemplazo. NOTA Si existen varios marcadores de posición en una plantilla, se generará y reemplazará más de una contraseña en secuencia.	{"password":"password_placeholder"}
password_length	Longitud de la contraseña. El valor oscila entre 8 y 128 . 16 es el valor predeterminado.	16
password_format	Formato de la contraseña. 2 es el valor predeterminado. Los valores posibles son los siguientes: 1. Dígitos y letras 2. Dígitos, letras y caracteres especiales (~!@#%^*-_+=?) 3. Solo dígitos 4. Solo letras	2

Paso 6 Seleccione la pestaña **Code**, agregue la siguiente función de rotación de contraseña a la ventana de edición y haga clic en **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'
```

----Fin

Depuración


Se utiliza en el FunctionGraph creado. Para obtener más detalles, consulte [Depuración online](#).

Creación de un activador

Crear un activador. Para obtener más detalles, consulte [Uso de un activador de temporizador](#).

Consulta de un secreto

Paso 1 [Inicie sesión en la consola de gestión](#).

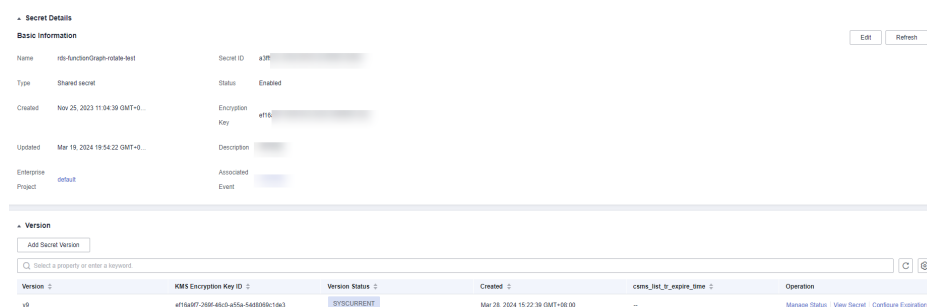
Paso 2 Haga clic en  a la izquierda, elija **Security & Compliance > Data Encryption Workshop**, se mostrará la página de gestión de claves.

Paso 3 En el panel de navegación, elija **Cloud Secret Management Service**.

Paso 4 Buscar el secreto creado en [Creación de una función de rotación de contraseña](#).

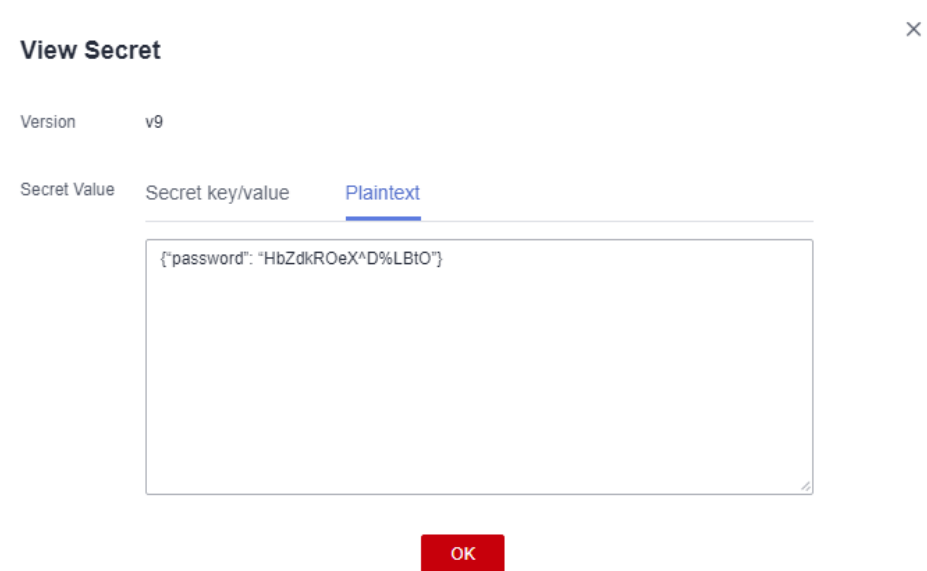
Paso 5 Haga clic en el secreto para ver sus detalles, incluidas las versiones actual e histórica.

Figura 2-11 Consulta de detalles de secretos



Paso 6 En **Version**, localice el secreto y haga clic en **View Secret** en la columna **Operation** para ver la contraseña.

Figura 2-12 Consulta de un valor del secreto



----Fin

2.4 Rotación de secretos

2.4.1 Descripción general

Si no ha actualizado los secretos durante mucho tiempo, la información importante (como contraseñas importantes, tokens, certificados, claves de SSH y claves de API) protegida por estos secretos está expuesta a riesgos de fuga. La rotación periódica de secretos mejora la seguridad de la información de texto plano protegida.

Huawei Cloud ofrece dos políticas de rotación de secreto:

- **Rotación de un secreto para un usuario**
- **Rotación de secreto para dos usuarios**

Puede seleccionar una política de rotación según sea necesario.

Procedimiento

1. El administrador utiliza una versión del secreto y actualiza su contenido en la [consola](#) o mediante una [API](#).
2. La aplicación invoca a una API de CSMS para obtener la última versión del secreto o el secreto de un estado de versión especificada y luego gira el secreto.
3. Repita regularmente los pasos [1](#) y [2](#) para girar secretos.

2.4.2 Rotación de un secreto para un usuario

Descripción general

Puede actualizar la información de un usuario en un secreto.

Esta política de rotación de secreto es la más utilizada.

Ejemplo:

- Para acceder a la base de datos, no se elimina una conexión de base de datos durante la rotación de secreto. Después de la rotación, las nuevas conexiones utilizan los nuevos secretos.
- Un usuario puede crear una cuenta, por ejemplo, utilizando una dirección de correo electrónico como nombre de usuario. Por lo general, los usuarios pueden cambiar las contraseñas según sea necesario, pero no pueden crear otros usuarios ni cambiar nombres de usuario.
- Los usuarios temporales se crean en el momento en que se necesitan.
- Las contraseñas son introducidas por los usuarios, no recuperadas de CSMS por las aplicaciones. Estos usuarios no necesitan cambiar sus nombres de usuario o contraseñas.

Restricciones

Asegúrese de que su cuenta tenga permiso de KMS Administrator o de KMS CMKFullAccess. Para obtener más detalles, consulte [Gestión de permisos de DEW](#).

API de rotación de secreto

Puede invocar a las siguientes API para rotar secretos localmente.

API	Descripción
Creación de una versión de secreto	Crear una versión de secreto.
Consulta de la versión y el valor de del secreto	La información sobre una versión del secreto específica y el valor del secreto de texto plano de la versión.

Ejemplo de rotación del secreto de la cuenta única

1. Cree un secreto en la consola de Huawei Cloud. Para obtener más detalles, consulte [Creación de un secreto](#).

2. Prepare la información básica de autenticación.
 - **ACCESS_KEY**: Clave de acceso de la cuenta de Huawei
 - **SECRET_ACCESS_KEY**: Clave de acceso secreta de la cuenta de Huawei
 - **PROJECT_ID**: ID de proyecto de un sitio de Huawei Cloud. Para obtener más detalles, véase [Proyecto](#).
 - **CSMS_ENDPOINT**: punto de conexión para acceder al CSMS. Consulte [Puntos de conexión](#) para obtener más detalles.
 - Habrá riesgos de seguridad si la AK/SK utilizada para la autenticación se escribe directamente en el código. Cifre la AK/SK en el archivo de configuración o en las variables de entorno para su almacenamiento.
 - En este ejemplo, las AK/SK almacenadas en las variables de entorno se utilizan para la autenticación de identidad. Primero configure las variables de entorno **HUAWEICLOUD_SDK_AK** y **HUAWEICLOUD_SDK_SK** en el entorno local.

3. Rotación del secreto de un usuario.

En el ejemplo de código,

- **secretName** indica el nombre del secreto creado en la consola de Huawei Cloud.
- **secretString** indica el valor guardado en el secreto creado en la consola de Huawei Cloud.
- **versionId** indica el ID del secreto generado automáticamente después de crear un secreto en la consola de Huawei Cloud.
- **LATEST_VERSION** indica la versión del secreto.

```
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/en-us/productdesc-iam/iam\_01\_0023.html
     * - CSMS_ENDPOINT: endpoint address for accessing CSMS. For details, see https://support.huaweicloud.com/intl/en-us/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 Rotación de un secreto para dos usuarios

Descripción general

Puede actualizar la información de dos usuarios en un secreto.

Por ejemplo, si su aplicación requiere alta disponibilidad y realiza la rotación de un solo usuario, es posible que no pueda acceder a la aplicación al cambiar la contraseña de usuario y actualizar el contenido secreto. En este caso, debe utilizar la política de rotación de secreto de multiusuarios.

Debe tener una cuenta. Por ejemplo, **Admin** que tenga permiso para crear y cambiar las contraseñas de **user1** y **user2**. Primero, cree un secreto, cree la cuenta y contraseña de **user1** y regístrelas como **user1/password1**. Luego, cree **user2** agregando la versión del secreto **v2** y guarde la contraseña de **user2** como **user2/password2**. Si cambia la contraseña de **user1**, deberá agregar una versión de secreto **v3** y registrarla como **user1/password3**. Cuando se crea la versión de secreto **v3**, la aplicación utiliza la versión de secreto **v2** existente para obtener información. Una vez que **v3** esté lista, se cambiará la etiqueta de almacenamiento temporal y la aplicación podrá utilizar **v3** para obtener información.

Restricciones

Asegúrese de que su cuenta tenga permiso de KMS Administrator o de KMS CMKFullAccess. Para obtener más detalles, consulte [Gestión de permisos de DEW](#).

API de rotación de secreto

Puede invocar a las siguientes API para rotar secretos localmente.

API	Descripción
Creación de una versión de secreto	Crear una versión de secreto.
Consulta de la versión y el valor de del secreto	La información sobre una versión del secreto específica y el valor del secreto de texto plano de la versión.
Actualización del estado de versión de un secreto	Actualizar el estado de versión de un secreto.
Consulta del estado de una versión del secreto	Consultar el estado de una versión del secreto especificada.

Ejemplo de rotación del secreto de las cuentas dobles

1. Cree un secreto en la consola de Huawei Cloud como administrador. Para obtener más detalles, consulte [Creación de un secreto](#).
2. Prepare la información básica de autenticación.
 - **ACCESS_KEY**: Clave de acceso de la cuenta de Huawei
 - **SECRET_ACCESS_KEY**: Clave de acceso secreta de la cuenta de Huawei
 - **PROJECT_ID**: ID de proyecto de un sitio de Huawei Cloud. Para obtener más detalles, véase [Proyecto](#).
 - **CSMS_ENDPOINT**: punto de conexión para acceder al CSMS. Consulte [Puntos de conexión](#) para obtener más detalles.
 - 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.
3. Rotate the secret for two users.

In the example code,

- **secretName** indicates the name of the secret created on the Huawei Cloud console.
- **secretString** indicates the value saved in the secret created on the Huawei Cloud console.
- **versionId** indicates the secret ID automatically generated after a secret is created on the Huawei Cloud console.
- **LATEST_VERSION** indicates the secret version.

```
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/en-us/productdesc-iam/iam\_01\_0023.html
     * - CSMS_ENDPOINT: endpoint address for accessing CSMS. For details, see https://support.huaweicloud.com/intl/en-us/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 Rotación de secretos de IAM con FunctionGraph

Escenario

Esta sección describe cómo rotar secretos de IAM con KMS mediante una plantilla de FunctionGraph.


Restricciones


- Solo se pueden rotar las cuentas de miembros de IAM. Las cuentas principales de IAM no se pueden rotar.
- La cuenta de miembro de IAM que se va a rotar debe tener al menos un grupo de secretos.
- El CSMS está disponible en la región.

Procedimiento

Creación de una delegación.

Paso 1 [Inicie sesión en la consola de gestión.](#)

Paso 2 Haga clic en  en la esquina superior izquierda de la consola de gestión y seleccione una región o proyecto.

Paso 3 Haga clic en  en la esquina superior izquierda de la página y elija **Management & Governance > Identity and Access Management**.

Paso 4 En el panel de navegación situado a la izquierda, seleccione **Agencies**. Haga clic en **Create Agency** en la esquina superior derecha.

Paso 5 Configure los parámetros, como se muestra en la siguiente figura. [Tabla 2-6](#) enumera los parámetros.

Figura 2-13 Creación de una delegación

* Agency Name

* Agency Type Account
Delegate another Huawei Cloud account to perform operations on your resources.

Cloud service
Delegate a cloud service to access your resources in other cloud services.

* Cloud Service

* Validity Period

Description
38/255

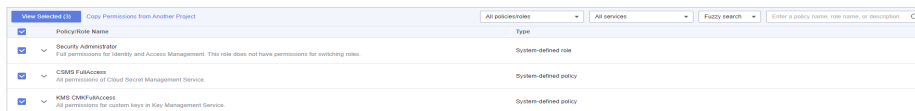
Tabla 2-6 Parámetros para crear una delegación

Parámetro	Descripción
Agency Name	Introduzca un nombre de delegación personalizado, por ejemplo, test .
Agency Type	Seleccione Cloud service .
Cloud Service	Seleccione FunctionGraph .
Validity Period	Configure este parámetro en función del escenario real. Si la función debe ejecutarse durante mucho tiempo, seleccione Unlimited .
Description	Configure este parámetro según sea necesario.

Paso 6 Haga clic en **Next**.

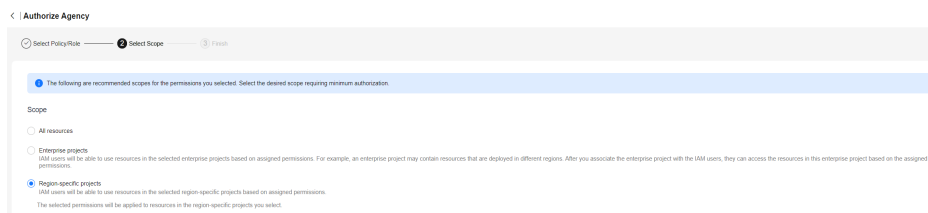
Paso 7 Seleccione **Security Administrator**, **CSMS FullAccess** y **KMS CMKFullAccess** como se muestra en la siguiente figura.

Figura 2-14 Asignación de permisos



Paso 8 Haga clic en **Next** y seleccione un ámbito de autorización, como se muestra en la siguiente figura.

Figura 2-15 Alcance de la autorización





Paso 9 Haga clic en **OK**.

----**Fin**

Creación de una plantilla de función y una función.

Paso 1 **Inicie sesión en la consola de gestión.**

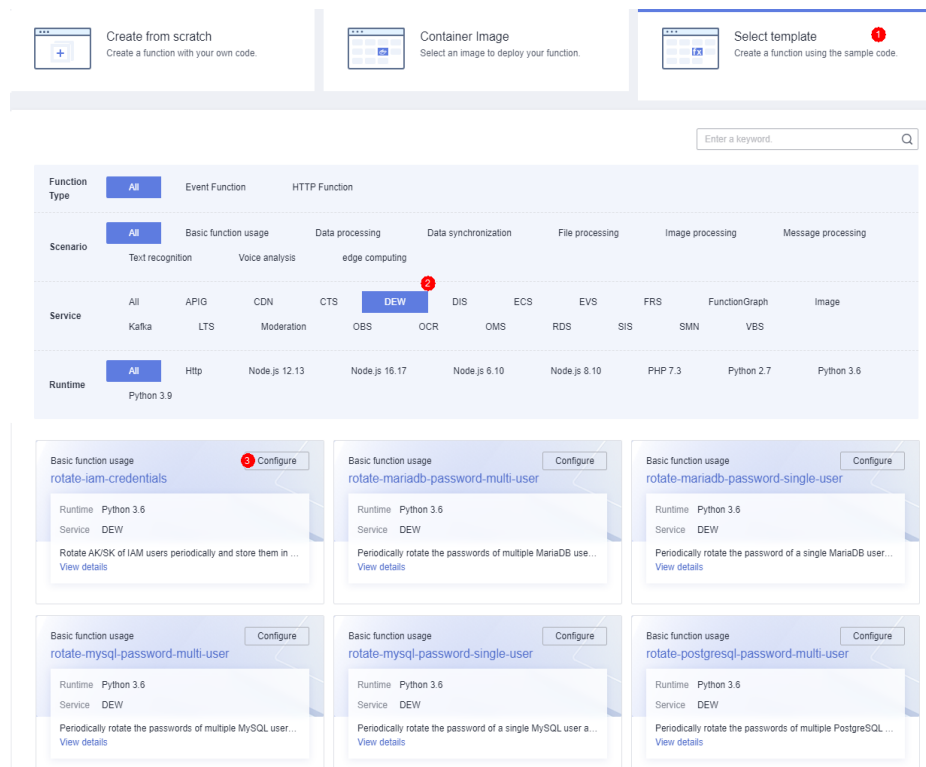
Paso 2 Haga clic en  en la esquina superior izquierda de la consola de gestión y seleccione una región o proyecto.

Paso 3 Haga clic en  en la esquina superior izquierda de la página y elija **Compute > FunctionGraph**.

Paso 4 Haga clic en **Create Function** en la esquina superior derecha.

Paso 5 Haga clic en **Select template**. En el área **Service**, haga clic en **DEW**, localice **rotate-iam-credentials** en la parte inferior y haga clic en **Configure** en la esquina superior derecha del cuadro, como se muestra en la siguiente figura.

Figura 2-16 Creación de una función



Paso 6 Configure los parámetros descritos en **Tabla 2-7**.

Figura 2-17 Información básica sobre funciones

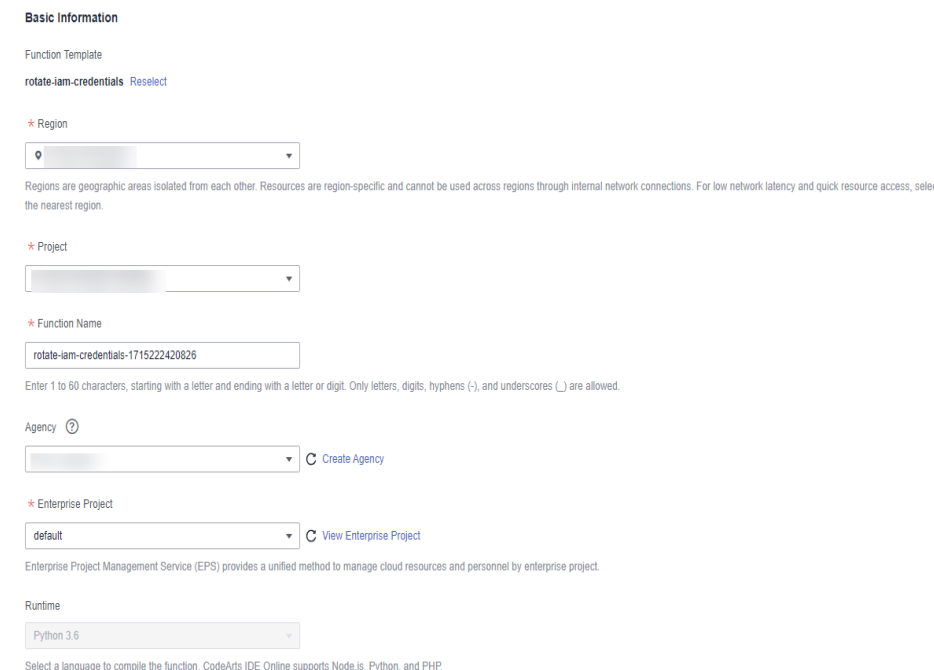


Tabla 2-7 Parámetros básicos

Parámetro	Descripción
Region	Seleccione la región donde se va a desplegar la función.
Project	Seleccione el proyecto en el que se va a desplegar la función.
Function Name	Introduzca un nombre de función personalizado.
Agency	Seleccione una delegación.
Enterprise Project	Si habilitó los proyectos empresariales, seleccione uno para agregarle una función. Si todavía no habilita un proyecto empresarial, este parámetro no estará disponible. Omite este paso.

Paso 7 Configure las variables de entorno descritas en [Tabla 2-8](#).

Figura 2-18 Variables del entorno

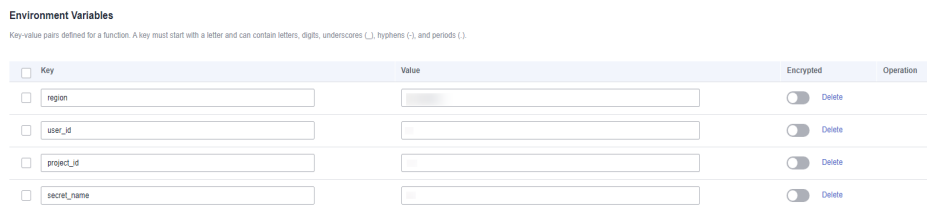


Tabla 2-8 Variables del entorno

Parámetro	Descripción
region	El proyecto. Por ejemplo, si la región es CN North-Beijing4, el nombre del proyecto debe ser cn-north-4 . NOTA Para obtener el proyecto, coloque el mouse sobre el nombre de usuario y seleccione Mis credenciales en la lista desplegable. Se muestran los proyectos.
user_id	Ingrese el ID del usuario de IAM que se va a rotar. NOTA Para obtener el ID de usuario de IAM, coloque el mouse sobre el nombre de usuario y seleccione Mis credenciales en la lista desplegable. Se muestra el ID.

Parámetro	Descripción
project_id	Introduzca el ID del proyecto. NOTA Para obtener el ID de usuario de IAM, coloque el mouse sobre el nombre de usuario y seleccione Mis credenciales en la lista desplegable. Se muestran los ID de los proyectos.
secret_name	Introduzca el nombre del secreto que se va a crear, que no puede coincidir con un nombre de secreto existente.

Paso 8 Configure las variables de activador descritas en [Tabla 2-9](#).

Figura 2-19 Variables de activador

Create Trigger
×

Trigger Type Timer

The total number of DDS, GAUSSMONGO, DIS, LTS, Kafka and timer triggers cannot exceed 10. You have created 0 such triggers.

* Timer Name Timer-jntj

Enter 1 to 64 characters, starting with a letter. Only letters, digits, hyphens (-), and underscores (_) are allowed.

* Rule
 Fixed rate Cron expression

3 Minutes

1-60 minutes, 1-24 hours, or 1-30 days

* Enable Trigger

Additional Information ?

0/2,048

Tabla 2-9 Variables de activador

Parámetro	Descripción
Timer Name	Personalizado
Rule	Configurado según sea necesario
Enable Trigger	Configurado según sea necesario

Paso 9 Haga clic en **Create Function**.


----Fin


Depuración.

Para obtener detalles sobre la depuración de un flujo de trabajo de función, consulte [Depuración online](#).

Consulta del secreto.

Paso 1 [Inicie sesión en la consola de gestión](#).

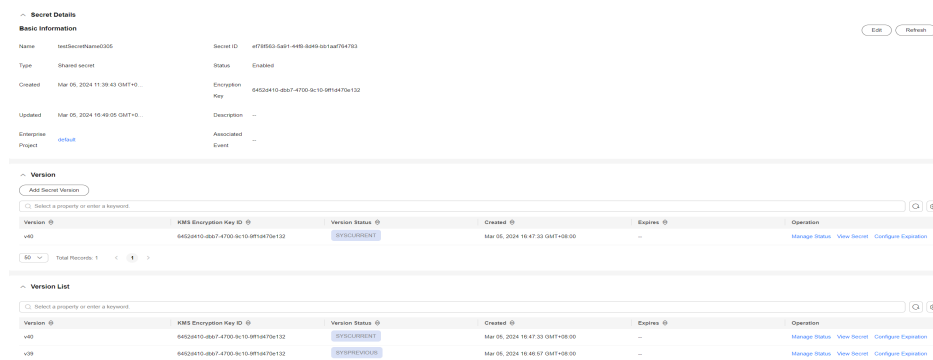
Paso 2 Haga clic en  en la esquina superior izquierda de la consola de gestión y seleccione una región o proyecto.

Paso 3 Haga clic en  a la izquierda, elija **Security & Compliance > Data Encryption Workshop**, se mostrará la página de gestión de claves.

Paso 4 En el panel de navegación, elija **Cloud Secret Management Service**.

Paso 5 Consulte el secreto especificado en el [Paso 7](#). Haga clic en él para ver los detalles. Se muestran el secreto válido actual, los secretos históricos y otras informaciones.

Figura 2-20 Detalles de secretos



Paso 6 Localice la versión más reciente del secreto, haga clic en **View Secret** en la columna **Operation** para verificar las AK/SK.

----Fin

3 Generales

3.1 Reintento de solicitudes de DEW fallidas con retardo exponencial

Escenario

Si recibe un mensaje de error al invocar a una API, puede utilizar el retroceso exponencial para volver a intentar la solicitud.

Cómo funciona

Si el lado del servicio informa de errores consecutivos (como errores de limitación de tráfico), el acceso continuo seguirá causando conflictos. El retroceso exponencial puede ayudarle a evitar este tipo de errores.

Restricciones

La cuenta actual tiene una clave habilitada.

Ejemplo

1. Prepare la información básica de autenticación.
 - **ACCESS_KEY**: Clave de acceso de la cuenta de Huawei. Para obtener más detalles, consulte [¿Cómo obtengo una clave de acceso \(AK/SK\)?](#)
 - **SECRET_ACCESS_KEY**: Clave de acceso secreta de la cuenta de Huawei. Para obtener más detalles, consulte [¿Cómo obtengo una clave de acceso \(AK/SK\)?](#)
 - **PROJECT_ID**: ID del proyecto del sitio. Para obtener más detalles, consulte [Obtención de un ID de proyecto](#).
 - **KMS_ENDPOINT**: El punto de conexión para acceder al KMS. Para obtener más detalles, consulte [Puntos de conexión](#).
 - Habrá riesgos de seguridad si la AK/SK utilizada para la autenticación se escribe directamente en el código. Cifre la AK/SK en el archivo de configuración o en las variables de entorno para su almacenamiento.

- En este ejemplo, las AK/SK almacenadas en las variables de entorno se utilizan para la autenticación de identidad. Primero configure las variables de entorno **HUAWEICLOUD_SDK_AK** y **HUAWEICLOUD_SDK_SK** en el entorno local.

2. Código para el retroceso 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);
                    }
                }
            }
        }
    }
}
```

```
}  
}  
}
```