Guía del usuario de Hooks

AWS CloudFormation

AWS CloudFormation: Guía del usuario de Hooks

Table of Contents

¿Qué son los AWS CloudFormation ganchos?

AWS CloudFormation Hooks es una función que puede utilizar para garantizar que sus CloudFormation recursos, pilas y conjuntos de cambios cumplan con las mejores prácticas de seguridad, operativas y de optimización de costes de su organización. CloudFormation Hooks también puede garantizar este mismo nivel de cumplimiento con tus API de control de nube de AWS recursos. Con CloudFormation Hooks, puedes proporcionar código que inspeccione de forma proactiva la configuración de tus AWS recursos antes del aprovisionamiento. Si se encuentran recursos que no cumplen con las normas, se produce un error en la operación y se impide el aprovisionamiento de los recursos, o AWS CloudFormation bien se emite una advertencia y se permite que la operación de aprovisionamiento continúe.

Puedes usar Hooks para hacer cumplir una variedad de requisitos y pautas. Por ejemplo, un Hook relacionado con la seguridad puede verificar que los grupos de seguridad cumplan con las reglas de tráfico entrante y saliente adecuadas para su Amazon Virtual Private Cloud (Amazon VPC). Un Hook relacionado con los costos puede restringir los entornos de desarrollo para que solo usen tipos de instancias más pequeñas de Amazon Elastic Compute Cloud (Amazon EC2). Un Hook diseñado para la disponibilidad de los datos puede imponer copias de seguridad automáticas para Amazon Relational Database Service (Amazon RDS).

CloudFormation Hooks es un tipo de extensión compatible en el AWS CloudFormation registro. El registro facilita la distribución y activación de Hooks tanto de forma pública como privada. Puede usar Hooks prediseñados o crear sus propios Hooks mediante la CloudFormation CLI.

Esta guía proporciona una descripción general de la estructura de AWS CloudFormation Hooks y guías para desarrollar, registrar, probar, administrar y publicar tus propios Hooks.

Creación y administración de AWS CloudFormation Hooks

AWS CloudFormation Los ganchos proporcionan un mecanismo para evaluar tus CloudFormation recursos antes de permitir la creación, modificación o eliminación de pilas. Esta función le ayuda a garantizar que sus CloudFormation recursos cumplan con las mejores prácticas de seguridad, operativas y de optimización de costes de su organización.

Para crear un Hook, tienes tres opciones.

- Guard Hook: evalúa los recursos mediante una AWS CloudFormation Guard regla.
- Lambda Hook: reenvía las solicitudes de evaluación de recursos a una función. AWS Lambda
- Hook personalizado: utiliza un controlador de Hook personalizado que se desarrolla manualmente.

Guard Hook

Para crear un gancho de protección, sigue estos pasos principales:

1. Escriba su lógica de evaluación de recursos como una regla de política de Guard utilizando el lenguaje específico de dominio (DSL) de Guard.
2. Guarde la regla de política Guard en un bucket de Amazon S3.
3. Ve a la CloudFormation consola y comienza a crear un Guard Hook.
4. Indique la ruta de Amazon S3 a su regla de guardia.
5. Elige los objetivos específicos que evaluará el Hook.
6. Elige las acciones de despliegue (crear, actualizar, eliminar) que invocarán tu Hook.
7. Elige cómo responde el Hook cuando no pasa la evaluación.
8. Cuando se complete la configuración, active el Hook para comenzar a aplicarlo.

Lambda Hook

Para crear un Hook Lambda, siga estos pasos principales:

1. Escriba la lógica de evaluación de recursos como una función Lambda.
2. Navegue hasta la CloudFormation consola y comience a crear un Hook Lambda.
3. Proporcione el nombre de recurso de Amazon (ARN) para la función Lambda.
4. Elija los objetivos específicos que evaluará el Hook.

5. Elige las acciones de despliegue (crear, actualizar, eliminar) que invocarán tu Hook.
6. Elige cómo responde el Hook cuando no pasa la evaluación.
7. Cuando se complete la configuración, active el Hook para comenzar a aplicarlo.

Custom Hook

Los Hooks personalizados son extensiones que se registran en el CloudFormation registro mediante la interfaz de línea de CloudFormation comandos (CFN-CLI).

Para crear un Hook personalizado, sigue estos pasos principales:

1. Inicie el proyecto: genere los archivos necesarios para desarrollar un Hook personalizado.
2. Modele el Hook: escriba un esquema que defina el Hook y los controladores que especifiquen las operaciones que pueden invocar el Hook.
3. Registra y activa el Hook: una vez que hayas creado un Hook, tendrás que registrarlo en la cuenta y la región en la que quieras usarlo y esto lo activará.

En los temas siguientes se proporciona más información sobre cómo crear y administrar Hooks.

Temas

AWS CloudFormation Conceptos de Hooks

La siguiente terminología y conceptos son fundamentales para su comprensión y uso de AWS CloudFormation Hooks.

Gancho

Un Hook contiene código que se invoca inmediatamente antes de CloudFormation crear, actualizar o eliminar pilas o recursos específicos. También se puede invocar durante una operación de creación de un conjunto de cambios. Hooks puede inspeccionar la plantilla, los recursos o el conjunto

de cambios que CloudFormation se van a aprovisionar. Además, los Hooks se pueden invocar inmediatamente antes de que la API de Cloud Control cree, actualice o elimine recursos específicos.

Si un Hook identifica alguna configuración que no cumpla con las pautas organizativas definidas en tu lógica de Hook, puedes elegir entre usar `WARN` los usuarios o `FAIL` CloudFormation impedir el aprovisionamiento del recurso.

Los Hooks tienen las siguientes características:

- Validación proactiva: reduce el riesgo, la sobrecarga operativa y los costos al identificar los recursos que no cumplen con las normas antes de crearlos, actualizarlos o eliminarlos.
- Aplicación automática: proporciona una aplicación interna Cuenta de AWS para evitar que los recursos que no cumplan con las normas sean aprovisionados por. CloudFormation

Modo de fallo

Tu lógica de Hook puede devolver el éxito o el fracaso. Una respuesta correcta permitirá que la operación continúe. Un error en un recurso que no cumpla con las normas puede provocar lo siguiente:

- `FAIL`— Detiene la operación de aprovisionamiento.
- `WARN`— Permite que el aprovisionamiento continúe con un mensaje de advertencia.

Crear Hooks en el `WARN` modo Hook es una forma eficaz de supervisar el comportamiento de los Hooks sin que ello afecte a las operaciones de apilado. En primer lugar, activa Hooks en `WARN` el modo Hooks para saber qué operaciones se verán afectadas. Una vez que hayas evaluado los posibles efectos, puedes cambiar el modo Hook al `FAIL` modo para empezar a evitar operaciones que no cumplan con las normas.

¡Gancho objetivos!

Los objetivos de Hook especifican las operaciones que evaluará un Hook. Estas pueden ser operaciones en:

- Recursos compatibles con CloudFormation (`RESOURCE`)
- Apile plantillas (`STACK`)
- Conjuntos de cambios (`CHANGE_SET`)

- Recursos compatibles con la API de Cloud Control (`CLOUD_CONTROL`)

Define uno o más objetivos que especifican las operaciones más amplias que evaluará el Hook. Por ejemplo, puedes crear una segmentación de Hook `RESOURCE` para segmentar todos los AWS recursos y `STACK` todas las plantillas de pila.

Segmenta las acciones

Las acciones objetivo definen las acciones específicas (`CREATEUPDATE`, o`DELETE`) que invocarán un Hook. Para `RESOURCESTACK`, y `CLOUD_CONTROL` los objetivos, son aplicables todas las acciones objetivo. En el caso de `CHANGE_SET` los objetivos, solo se aplica la `CREATE` acción.

Manipulador de ganchos

En el caso de los Hooks personalizados, este es el código que gestiona la evaluación. Está asociado a un punto de invocación objetivo y a una acción objetivo que marcan un punto exacto por el que corre un Hook. Escribes controladores que alojan la lógica para estos puntos específicos. Por ejemplo, un punto de invocación `PRE` objetivo con una acción `CREATE` objetivo crea un controlador `preCreate` Hook. El código del controlador Hook se ejecuta cuando un punto de invocación objetivo y un servicio coincidentes realizan una acción objetivo asociada.

Valores válidos: (`preCreate`| |`preUpdate`) `preDelete`

> **Important**
>
> Apila las operaciones que dan como resultado el estado de `UpdateCleanup` no invocan un Hook. Por ejemplo, en los dos escenarios siguientes, no se invoca el `preDelete` controlador del Hook:
>
> - la pila se actualiza después de eliminar un recurso de la plantilla.
> - se elimina un recurso con el tipo de actualización de reemplazo.

Límites de tiempo de espera y reintentos

Los Hooks tienen un límite de tiempo de espera de 30 segundos por invocación y están limitados a 3 reintentos. Si una invocación supera el tiempo de espera, devolvemos un mensaje de error que indica que se ha agotado el tiempo de espera de la ejecución de Hook. Tras el tercer reintento, CloudFormation marca la ejecución de Hook como fallida.

Enlaces de Guard

Para usar un AWS CloudFormation Guard Hook en tu cuenta, debes activar el Hook para la cuenta y la región en la que quieras usarlo. Al activar un Hook, podrás utilizarlo en las operaciones de apilamiento en la cuenta y la región en las que esté activado.

Cuando activas un Guard Hook, se CloudFormation crea una entrada en el registro de tu cuenta para el Hook activado como Hook privado. Esto te permite establecer cualquier propiedad de configuración que incluya el Hook. Las propiedades de configuración definen cómo se configura el Hook para una región determinada Cuenta de AWS .

Temas

AWS CLI comandos para trabajar con Guard Hooks

Los AWS CLI comandos para trabajar con Guard Hooks incluyen:

- activate-typepara iniciar el proceso de activación de un Guard Hook.
- set-type-configurationpara especificar los datos de configuración de un Hook en su cuenta.
- list-typespara incluir los Hooks de tu cuenta.
- describe-typepara obtener información detallada sobre un Hook específico o una versión específica de Hook, incluidos los datos de configuración actuales.
- deactivate-typepara eliminar un Hook previamente activado de tu cuenta.

Escribe las reglas de Guard para evaluar los recursos de Guard Hooks

AWS CloudFormation Guard es un lenguaje específico de dominio (DSL) de código abierto y de uso general que puede utilizar como autor. policy-as-code En este tema se explica cómo usar Guard para crear reglas de ejemplo que se pueden ejecutar en Guard Hook para evaluar CloudFormation y API

de control de nube de AWS operar automáticamente. También se centrará en los diferentes tipos de entradas disponibles para tus reglas de guardia en función del momento en que se ejecute tu garfio. Puedes configurar un Guard Hook para que funcione durante los siguientes tipos de operaciones:

- Operaciones de recursos
- Operaciones de apilado
- Operaciones del conjunto de cambios

Para obtener más información sobre cómo escribir reglas de Guard, consulte Escribir AWS CloudFormation Guard reglas

Temas

- Reglas de Guard para la operación de recursos
- Reglas de Stack Operation Guard
- Cambie las reglas establecidas de Operation Guard

Reglas de Guard para la operación de recursos

Cada vez que se crea, actualiza o elimina un recurso, se considera una operación de recurso. Por ejemplo, si ejecutas la actualización de una CloudFormation pila que crea un recurso nuevo, habrás completado una operación de recursos. Cuando creas, actualizas o eliminas un recurso mediante la API de Cloud Control, también se considera una operación de recursos. Puedes configurar tu Guard Hook según el objetivo `RESOURCE` y `CLOUD_CONTROL` las operaciones en la `TargetOperations` configuración de tu Hook. Cuando tu Guard Hook evalúa una operación de recursos, el motor Guard evalúa una entrada de recursos.

Temas

- Sintaxis de entrada de recursos de Guard
- Ejemplo de entrada de operación de recursos de Guard
- Proteja las reglas para los cambios en los recursos

Sintaxis de entrada de recursos de Guard

La entrada de recursos de Guard son los datos que están disponibles para que las reglas de Guard los evalúen.

El siguiente es un ejemplo de la forma de una entrada de recursos:

```
HookContext:
  AWSAccountID: String
  StackId: String
  HookTypeName: String
  HookTypeVersion: String
  InvocationPoint: [CREATE_PRE_PROVISION, UPDATE_PRE_PROVISION, DELETE_PRE_PROVISION]
  TargetName: String
  TargetType: RESOURCE
  TargetLogicalId: String
  ChangeSetId: String
Resources:
  {ResourceLogicalID}:
    ResourceType: {ResourceType}
    ResourceProperties:
        {ResourceProperties}
Previous:
  ResourceLogicalID:
    ResourceType: {ResourceType}
    ResourceProperties:
        {PreviousResourceProperties}
```

`HookContext`

`AWSAccountID`

El ID Cuenta de AWS que contiene el recurso que se está evaluando.

`StackId`

El ID de pila de la CloudFormation pila que forma parte de la operación de recursos. Está vacío si la persona que llama es Cloud Control API.

`HookTypeName`

El nombre del Hook que se está ejecutando.

`HookTypeVersion`

La versión del Hook que se está ejecutando.

`InvocationPoint`

El punto exacto de la lógica de aprovisionamiento en el que se ejecuta el Hook.

Valores válidos: (`CREATE_PRE_PROVISION`| `UPDATE_PRE_PROVISION` |`DELETE_PRE_PROVISION`)

`TargetName`

El tipo de objetivo que se está evaluando, por ejemplo,`AWS::S3::Bucket`.

`TargetType`

El tipo de objetivo que se está evaluando, por ejemplo`AWS::S3::Bucket`. En el caso de los recursos aprovisionados con la API de Cloud Control, este valor será`RESOURCE`.

`TargetLogicalId`

El `TargetLogicalId` del recurso que se está evaluando. Si el origen del Hook es CloudFormation, será el ID lógico (también conocido como nombre lógico) del recurso. Si el origen del Hook es la API de Cloud Control, será un valor construido.

`ChangeSetId`

El ID del conjunto de cambios que se ejecutó para provocar la invocación del Hook. Este valor está vacío si el cambio de recurso lo inició la API de Cloud Control o las `delete-stack` operaciones `create-stack`update-stack, o.

`Resources`

`ResourceLogicalID`

Cuando la operación se inicia con CloudFormation, `ResourceLogicalID` es el ID lógico del recurso de la CloudFormation plantilla.

Cuando la operación la inicia la API de Cloud Control, `ResourceLogicalID` es una combinación del tipo de recurso, el nombre, el ID de operación y el ID de solicitud.

`ResourceType`

El nombre del tipo del recurso (ejemplo:`AWS::S3::Bucket`).

`ResourceProperties`

Las propiedades propuestas del recurso que se está modificando. Cuando el Guard Hook se ejecute en función de los cambios en el CloudFormation recurso, todas las funciones, parámetros y transformaciones se resolverán por completo. Si se va a eliminar el recurso, este valor estará vacío.

`Previous`

`ResourceLogicalID`

Cuando la operación se inicia con CloudFormation, `ResourceLogicalID` es el identificador lógico del recurso de la CloudFormation plantilla.

Cuando la operación la inicia la API de Cloud Control, `ResourceLogicalID` es una combinación del tipo de recurso, el nombre, el ID de operación y el ID de solicitud.

`ResourceType`

El nombre del tipo del recurso (ejemplo:`AWS::S3::Bucket`).

`ResourceProperties`

Las propiedades actuales asociadas al recurso que se está modificando. Si se va a eliminar el recurso, este valor estará vacío.

Ejemplo de entrada de operación de recursos de Guard

El siguiente ejemplo de entrada muestra un Guard Hook que recibirá la definición del `AWS::S3::Bucket` recurso que se va a actualizar. Estos son los datos disponibles para que Guard los evalúe.

```
HookContext:
  AwsAccountId: "123456789012"
  StackId: "arn:aws:cloudformation:us-west-2:123456789012:stack/
MyStack/1a2345b6-0000-00a0-a123-00abc0abc000"
  HookTypeName: org::s3policy::hook
  HookTypeVersion: "00001"
  InvocationPoint: UPDATE_PRE_PROVISION
  TargetName: AWS::S3::Bucket
  TargetType: RESOURCE
  TargetLogicalId: MyS3Bucket
  ChangeSetId: ""
Resources:
  MyS3Bucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: amzn-s3-demo-bucket
      ObjectLockEnabled: true
Previous:
  MyS3Bucket:
```

```
    Type: AWS::S3::Bucket
    Properties:
      BucketName: amzn-s3-demo-bucket
      ObjectLockEnabled: false
```

Para ver todas las propiedades disponibles para el tipo de recurso, consulte AWS::S3::Bucket.

Proteja las reglas para los cambios en los recursos

Cuando un Guard Hook evalúa los cambios en los recursos, comienza por descargar todas las reglas configuradas con el Hook. Luego, estas reglas se evalúan en función de la entrada de recursos. The Hook fallará si alguna regla no pasa su evaluación. Si no hay errores, el Hook pasará.

El siguiente ejemplo es una regla de guardia que evalúa si la `ObjectLockEnabled` propiedad es `true` para algún tipo de `AWS::S3::Bucket` recurso.

```
let s3_buckets_default_lock_enabled = Resources.*[ Type == 'AWS::S3::Bucket']

rule S3_BUCKET_DEFAULT_LOCK_ENABLED when %s3_buckets_default_lock_enabled !empty {
  %s3_buckets_default_lock_enabled.Properties.ObjectLockEnabled exists
  %s3_buckets_default_lock_enabled.Properties.ObjectLockEnabled == true
  <<
    Violation: S3 Bucket ObjectLockEnabled must be set to true.
    Fix: Set the S3 property ObjectLockEnabled parameter to true.
  >>
}
```

Si esta regla se ejecuta en contra de la siguiente entrada, fallará ya que la `ObjectLockEnabled` propiedad no está establecida en. `true`

```
Resources:
  MyS3Bucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: amzn-s3-demo-bucket
      ObjectLockEnabled: false
```

Cuando esta regla se ejecute en contra de la siguiente entrada, pasará ya que `ObjectLockEnabled` está establecida en`true`.

```
Resources:
  MyS3Bucket:
```

```
    Type: AWS::S3::Bucket
    Properties:
      BucketName: amzn-s3-demo-bucket
      ObjectLockEnabled: true
```

Cuando se produce un error en un Hook, las reglas que hayan fallado se volverán a propagar a nuestra CloudFormation API de Cloud Control. Si se ha configurado un depósito de registro para el Guard Hook, allí se proporcionará información adicional sobre las reglas. Estos comentarios adicionales incluyen la `Fix` información `Violation` y la información.

Reglas de Stack Operation Guard

Cuando se crea, actualiza o elimina una CloudFormation pila, puedes configurar tu Guard Hook para que comience por evaluar la nueva plantilla y, posiblemente, impedir que la operación de apilamiento continúe. Puedes configurar tu Guard Hook para que se centre en `STACK` las operaciones en la `TargetOperations` configuración de tu Hook.

Temas

- Sintaxis de entrada de Guard Stack
- Ejemplo de entrada de operación de Guard Stack
- Reglas de protección para los cambios de pila

Sintaxis de entrada de Guard Stack

La entrada para las operaciones de Guard Stack proporciona la CloudFormation plantilla completa para que la evalúen las reglas de Guard.

El siguiente es un ejemplo de la forma de una entrada de pila:

```
HookContext:
  AWSAccountID: String
  StackId: String
  HookTypeName: String
  HookTypeVersion: String
  InvocationPoint: [CREATE_PRE_PROVISION, UPDATE_PRE_PROVISION, DELETE_PRE_PROVISION]
  TargetName: String
  TargetType:STACK
  ChangeSetId: String
{Proposed CloudFormation Template}
Previous:
```

```
    {CloudFormation Template}
```

`HookContext`

`AWSAccountID`

El ID del recurso Cuenta de AWS que contiene.

`StackId`

El ID de pila de la CloudFormation pila que forma parte de la operación de apilado.

`HookTypeName`

El nombre del Hook que se está ejecutando.

`HookTypeVersion`

La versión del Hook que se está ejecutando.

`InvocationPoint`

El punto exacto de la lógica de aprovisionamiento en el que se ejecuta el Hook.

Valores válidos: (`CREATE_PRE_PROVISION`| `UPDATE_PRE_PROVISION` |`DELETE_PRE_PROVISION`)

`TargetName`

El nombre de la pila que se está evaluando.

`TargetType`

Este valor será `STACK` cuando se ejecute como un Hook a nivel de pila.

`ChangeSetId`

El ID del conjunto de cambios que se ejecutó para provocar la invocación del Hook. Este valor está vacío si la operación de apilamiento se inició mediante una `delete-stack` operación `create-stackupdate-stack`, o.

`Proposed CloudFormation Template`

El valor completo CloudFormation de la plantilla que se pasó a CloudFormation `create-stack` nuestras `update-stack` operaciones. Esto incluye elementos como `ResourcesOutputs`, y`Properties`. Puede ser una cadena JSON o YAML, según lo que se haya proporcionado a CloudFormation.

En `delete-stack` las operaciones, este valor estará vacío.

`Previous`

La última CloudFormation plantilla implementada correctamente. Este valor está vacío si se va a crear o eliminar la pila.

En `delete-stack` las operaciones, este valor estará vacío.

> **Note**
>
> Las plantillas proporcionadas son las que se transfieren a las operaciones `create` o se `update` apilan. Al eliminar una pila, no se proporcionan valores de plantilla.

Ejemplo de entrada de operación de Guard Stack

El siguiente ejemplo de entrada muestra un Guard Hook que recibirá una plantilla completa y la plantilla implementada anteriormente. La plantilla de este ejemplo utiliza el formato JSON.

```
HookContext:
  AwsAccountId: 123456789012
  StackId: "arn:aws:cloudformation:us-west-2:123456789012:stack/
MyStack/1a2345b6-0000-00a0-a123-00abc0abc000"
  HookTypeName: org::templatechecker::hook
  HookTypeVersion: "00001"
  InvocationPoint: UPDATE_PRE_PROVISION
  TargetName: MyStack
  TargetType: CHANGE_SET
  TargetLogicalId: arn:aws:cloudformation:us-west-2:123456789012:changeSet/
SampleChangeSet/1a2345b6-0000-00a0-a123-00abc0abc000
  ChangeSetId: arn:aws:cloudformation:us-west-2:123456789012:changeSet/
SampleChangeSet/1a2345b6-0000-00a0-a123-00abc0abc000
Resources: {
   "S3Bucket": {
        "Type": "AWS::S3::Bucket",
        "Properties": {
           "BucketEncryption": {
               "ServerSideEncryptionConfiguration": [
                {"ServerSideEncryptionByDefault":
                    {"SSEAlgorithm": "aws:kms",
                      "KMSMasterKeyID": "KMS-KEY-ARN" },
                      "BucketKeyEnabled": true }
```

```
                ]
            }
        }
}
Previous: {
    "AWSTemplateFormatVersion": "2010-09-09",
    "Resources": {
        "S3Bucket": {
            "Type": "AWS::S3::Bucket",
            "Properties": {}
        }
    }
}
```

Reglas de protección para los cambios de pila

Cuando un Guard Hook evalúa los cambios en la pila, comienza por descargar todas las reglas configuradas con el Hook. Luego, estas reglas se evalúan en función de la entrada de recursos. The Hook fallará si alguna regla no pasa su evaluación. Si no hay errores, el Hook pasará.

El siguiente ejemplo es una regla de guardia que evalúa si hay algún tipo de `AWS::S3::Bucket` recurso que contenga una propiedad llamada`BucketEncryption`, con el valor `SSEAlgorithm` establecido en `aws:kms` o`AES256`.

```
let s3_buckets_s3_default_encryption = Resources.*[ Type == 'AWS::S3::Bucket']

rule S3_DEFAULT_ENCRYPTION_KMS when %s3_buckets_s3_default_encryption !empty {
  %s3_buckets_s3_default_encryption.Properties.BucketEncryption exists

 %s3_buckets_s3_default_encryption.Properties.BucketEncryption.ServerSideEncryptionConfiguratio
 in ["aws:kms","AES256"]
  <<
    Violation: S3 Bucket default encryption must be set.
    Fix: Set the S3 Bucket property
 BucketEncryption.ServerSideEncryptionConfiguration.ServerSideEncryptionByDefault.SSEAlgorithm
 to either "aws:kms" or "AES256"
  >>
}
```

Cuando la regla se ejecute en la siguiente plantilla, lo haráfail.

```
AWSTemplateFormatVersion: 2010-09-09
```

```
Description: S3 bucket without default encryption
Resources:
  EncryptedS3Bucket:
    Type: 'AWS::S3::Bucket'
    Properties:
      BucketName: !Sub 'encryptedbucket-${AWS::Region}-${AWS::AccountId}'
```

Cuando la regla se ejecute en la siguiente plantilla, lo harápass.

```
AWSTemplateFormatVersion: 2010-09-09
Description: S3 bucket with default encryption using SSE-KMS with an S3 Bucket Key
Resources:
  EncryptedS3Bucket:
    Type: 'AWS::S3::Bucket'
    Properties:
      BucketName: !Sub 'encryptedbucket-${AWS::Region}-${AWS::AccountId}'
      BucketEncryption:
        ServerSideEncryptionConfiguration:
          - ServerSideEncryptionByDefault:
              SSEAlgorithm: 'aws:kms'
              KMSMasterKeyID: KMS-KEY-ARN
            BucketKeyEnabled: true
```

Cambie las reglas establecidas de Operation Guard

Cuando se crea un conjunto de CloudFormation cambios, puede configurar su Guard Hook para que evalúe la plantilla y los cambios propuestos en el conjunto de cambios a fin de bloquear la ejecución del conjunto de cambios.

Temas

- Sintaxis de entrada del conjunto de cambios de Guard
- Ejemplo de entrada de operación de conjunto de cambios de Guard
- Regla de protección para las operaciones del conjunto de cambios

Sintaxis de entrada del conjunto de cambios de Guard

La entrada del conjunto de cambios de Guard son los datos que están disponibles para que las reglas de Guard los evalúen.

A continuación se muestra un ejemplo de la forma de una entrada de un conjunto de cambios:

```
HookContext:
  AWSAccountID: String
  StackId: String
  HookTypeName: String
  HookTypeVersion: String
  InvocationPoint: [CREATE_PRE_PROVISION, UPDATE_PRE_PROVISION, DELETE_PRE_PROVISION]
  TargetName: CHANGE_SET
  TargetType:CHANGE_SET
  TargetLogicalId:ChangeSet ID
  ChangeSetId: String
{Proposed CloudFormation Template}
Previous:
  {CloudFormation Template}
Changes: [{ResourceChange}]
```

La sintaxis del `ResourceChange` modelo es:

```
logicalResourceId: String
resourceType: String
acción: CREATE, UPDATE, DELETE
Número de línea: Number
Antes del contexto: JSON String
Después de Context: JSON String
```

`HookContext`

`AWSAccountID`

El ID del recurso Cuenta de AWS que contiene el recurso.

`StackId`

El ID de pila de la CloudFormation pila que forma parte de la operación de apilado.

`HookTypeName`

El nombre del Hook que se está ejecutando.

`HookTypeVersion`

La versión del Hook que se está ejecutando.

`InvocationPoint`

El punto exacto de la lógica de aprovisionamiento en el que se ejecuta el Hook.

Valores válidos: (`CREATE_PRE_PROVISION`| `UPDATE_PRE_PROVISION` |`DELETE_PRE_PROVISION`)

`TargetName`

El nombre de la pila que se está evaluando.

`TargetType`

Este valor será `CHANGE_SET` cuando se ejecute como un Hook a nivel de conjunto de cambios.

`TargetLogicalId`

Este valor será el ARN del conjunto de cambios.

`ChangeSetId`

El ID del conjunto de cambios que se ejecutó para provocar la invocación de Hook. Este valor está vacío si la operación de apilamiento se inició mediante una `delete-stack` operación `create-stackupdate-stack`, o.

`Proposed CloudFormation Template`

La CloudFormation plantilla completa que se proporcionó a una `create-change-set` operación. Puede ser una cadena JSON o YAML en función de lo que se haya proporcionado a CloudFormation.

`Previous`

La última CloudFormation plantilla implementada correctamente. Este valor está vacío si se va a crear o eliminar la pila.

`Changes`

El `Changes` modelo. Aquí se enumeran los cambios en los recursos.

Cambios

logicalResourceId

El nombre del recurso lógico del recurso modificado.

resourceType

El tipo de recurso que se cambiará.

acción

El tipo de operación que se está realizando en el recurso.

Valores válidos: (`CREATE`| `UPDATE` |`DELETE`)

Número de línea

El número de línea de la plantilla asociada al cambio.

Antes del contexto

Una cadena JSON de propiedades del recurso antes del cambio:

```
{"properties": {"property1": "value"}}
```

Después de Context

Una cadena JSON de propiedades del recurso tras el cambio:

```
{"properties": {"property1": "new value"}}
```

Ejemplo de entrada de operación de conjunto de cambios de Guard

El siguiente ejemplo de entrada muestra un Guard Hook que recibirá una plantilla completa, la plantilla implementada anteriormente y una lista de cambios en los recursos. La plantilla de este ejemplo utiliza el formato JSON.

```
HookContext:
  AwsAccountId: "00000000"
  StackId: MyStack
  HookTypeName: org::templatechecker::hook
  HookTypeVersion: "00001"
  InvocationPoint: UPDATE_PRE_PROVISION
  TargetName: my-example-stack
  TargetType:STACK
  TargetLogicalId: arn...:changeSet/change-set
  ChangeSetId: ""
Resources: {
    "S3Bucket": {
       "Type": "AWS::S3::Bucket",
       "Properties": {
```

```
            "BucketName": "amzn-s3-demo-bucket",
            "VersioningConfiguration":{
               "Status": "Enabled"
             }
          }
    }
Previous: {
    "AWSTemplateFormatVersion": "2010-09-09",
    "Resources": {
        "S3Bucket": {
            "Type": "AWS::S3::Bucket",
            "Properties": {
                "BucketName": "amzn-s3-demo-bucket",
                "VersioningConfiguration":{
                  "Status": "Suspended"
                }
            }
        }
    }
}
Changes: [
  {
    "logicalResourceId": "S3Bucket",
    "resourceType": "AWS::S3::Bucket",
    "action": "UPDATE",
    "lineNumber": 5,
    "beforeContext": "{\"Properties\":{\"VersioningConfiguration\":{\"Status\":
\"Suspended\"}}}",
    "afterContext": "{\"Properties\":{\"VersioningConfiguration\":{\"Status\":\"Enabled
\"}}}"
  }
]
```

Regla de protección para las operaciones del conjunto de cambios

El siguiente ejemplo es una regla de protección que evalúa los cambios en los buckets de Amazon S3 y garantiza que no estén `VersionConfiguration` deshabilitados.

```
let s3_buckets_changing = Changes[resourceType == 'AWS::S3::Bucket']

rule S3_VERSIONING_STAY_ENABLED when %s3_buckets_changing !empty {
    let afterContext = json_parse(%s3_buckets_changing.afterContext)
```

```
    when %afterContext.Properties.VersioningConfiguration.Status !empty {
        %afterContext.Properties.VersioningConfiguration.Status == 'Enabled'
    }
}
```

Prepárese para crear un gancho de protección

Antes de crear un Guard Hook, debe cumplir los siguientes requisitos previos:

- Debes haber creado ya una regla de guardia. Para obtener más información, consulte la Escribe reglas de protección para Hooks.
- El usuario o rol que crea el Hook debe tener permisos suficientes para activar los Hooks.
- Para usar el Guard Hook AWS CLI o un SDK para crear un Guard Hook, debes crear manualmente un rol de ejecución con permisos de IAM y una política de confianza que permita CloudFormation invocar un Guard Hook.

Crea un rol de ejecución para un Guard Hook

Un Hook usa un rol de ejecución para los permisos que necesita para invocar ese Hook en tu cuenta. Cuenta de AWS

Este rol se puede crear automáticamente si creas un Guard Hook desde el AWS Management Console; de lo contrario, debes crear este rol tú mismo.

En la siguiente sección, se muestra cómo configurar los permisos para crear tu Guard Hook.

Permisos necesarios

Siga las instrucciones sobre cómo crear un rol mediante políticas de confianza personalizadas de la Guía del usuario de IAM para crear un rol con una política de confianza personalizada.

Luego, completa los siguientes pasos para configurar tus permisos:

1. Adjunta la siguiente política de privilegios mínimos a la función de IAM que quieras usar para crear el Guard Hook.

   JSON

   ```
   {
   ```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket",
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::my-guard-output-bucket/*",
        "arn:aws:s3:::my-guard-rules-bucket"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::my-guard-output-bucket/*"
      ]
    }
  ]
}
```

2. Dale permiso a tu Hook para que asuma el rol añadiendo una política de confianza al rol. A continuación se muestra un ejemplo de política de confianza que puedes usar.

   JSON

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

```
  ]
}
```

Activa un Guard Hook en tu cuenta

En el siguiente tema, se muestra cómo activar un Guard Hook en tu cuenta para que puedas usarlo en la cuenta y la región en la que se activó.

Temas

Activa un Guard Hook (consola)

Para activar un Guard Hook para usarlo en tu cuenta

1. Inicia sesión AWS Management Console y abre la AWS CloudFormation consola en https://console.aws.amazon.com/cloudformation.
2. En la barra de navegación de la parte superior de la pantalla, elige el Región de AWS lugar donde quieres crear el Hook in.
3. Si aún no ha creado ninguna regla de Guard, cree la suya, guárdela en Amazon S3 y, a continuación, vuelva a este procedimiento. Consulte las reglas de ejemplo que aparecen en Escribe las reglas de Guard para evaluar los recursos de Guard Hooks el documento para empezar.

   Si ya ha creado su regla de guardia y la ha guardado en S3, continúe con el siguiente paso.

   > **Note**
   >
   > El objeto almacenado en S3 debe tener una de las siguientes extensiones de archivo: `.guard`, `.zip`, o `.tar.gz`.

4. Para el código fuente de Guard Hook, guarde sus reglas de Guard en S3 y haga lo siguiente:

   - Para el URI de S3, especifique la ruta de S3 al archivo de reglas o utilice el botón Examinar S3 para abrir un cuadro de diálogo en el que buscar y seleccionar el objeto de S3.

- (Opcional) Para la versión Object, si su bucket de S3 tiene habilitado el control de versiones, puede seleccionar una versión específica del objeto de S3.

  El Guard Hook descarga tus reglas de S3 cada vez que se invoca el Hook. Para evitar cambios o eliminaciones accidentales, te recomendamos que utilices una versión al configurar tu Guard Hook.

5. (Opcional) En el caso del informe de resultados de S3 bucket for Guard, especifique un bucket de S3 para almacenar el informe de resultados de Guard. Este informe contiene los resultados de las validaciones de las reglas de Guard.

   Para configurar el destino del informe de salida, elija una de las siguientes opciones:

   - Seleccione la casilla de verificación Usar el mismo grupo en el que están almacenadas las reglas de My Guard para usar el mismo grupo en el que se encuentran sus reglas de Guard.
   - Elija un nombre de bucket de S3 diferente para almacenar el informe de resultados de Guard.

6. (Opcional) Amplíe los parámetros de entrada de la regla de Guard y, a continuación, proporcione la siguiente información en Guardar los parámetros de entrada de la regla de Guard en S3:

   - Para el URI de S3, especifique la ruta de S3 a un archivo de parámetros o utilice el botón Examinar S3 para abrir un cuadro de diálogo en el que buscar y seleccionar el objeto de S3.
   - (Opcional) Para la versión Object, si su bucket de S3 tiene habilitado el control de versiones, puede seleccionar una versión específica del objeto de S3.

7. Elija Siguiente.

8. En Hook name, elige una de las siguientes opciones:

   - Proporcione un nombre breve y descriptivo que se añadirá después`Private::Guard::`. Por ejemplo, si escribes*`MyTestHook`*, el nombre completo de Hook pasa a ser`Private::Guard::`*`MyTestHook`*.
   - Introduce el nombre completo del Hook (también denominado alias) con este formato: *`Provider`*`::`*`ServiceName`*`::`*`HookName`*

9. En el caso de los objetivos de Hook, elige qué quieres evaluar:

   - Pilas: evalúa las plantillas de pilas cuando los usuarios crean, actualizan o eliminan pilas.
   - Recursos: evalúa los cambios en los recursos individuales cuando los usuarios actualizan las pilas.

- Conjuntos de cambios: evalúa las actualizaciones planificadas cuando los usuarios crean conjuntos de cambios.
- API de Cloud Control: evalúa las operaciones de creación, actualización o eliminación iniciadas por la API de Cloud Control.

10. En Acciones, elige qué acciones (crear, actualizar, eliminar) invocarán tu Hook.
11. En el modo Hook, elige cómo responderá el Hook cuando las reglas no pasen su evaluación:

    - Advertir: emite advertencias a los usuarios, pero permite que las acciones continúen. Esto resulta útil para validaciones o comprobaciones informativas no críticas.
    - Error: impide que la acción continúe. Esto resulta útil para aplicar políticas de seguridad o de cumplimiento estrictas.

12. Para la función de ejecución, elige la función de IAM que asumen los CloudFormation Hooks para recuperar tus reglas de Guard de S3 y, si lo prefieres, redactar un informe de resultados detallado de Guard. Puedes CloudFormation permitir que se cree automáticamente un rol de ejecución para ti o puedes especificar un rol que hayas creado.
13. Elija Siguiente.
14. (Opcional) Para los filtros Hook, haga lo siguiente:

    a. En el filtro de recursos, especifique qué tipos de recursos pueden invocar el Hook. Esto garantiza que el Hook solo se invoque para los recursos relevantes.

    b. En cuanto a los criterios de filtrado, elige la lógica para aplicar los filtros de nombre y rol de la pila:

    - Todos los nombres y roles de las pilas: The Hook solo se invocará cuando coincidan todos los filtros especificados.
    - Cualquier nombre y función de pila: se invocará el Hook si al menos uno de los filtros especificados coincide.

    > **Note**
    >
    > En el caso de las operaciones de la API de Cloud Control, se ignoran todos los filtros de nombres y roles de las pilas.

    c. En el caso de los nombres de pila, incluye o excluye pilas específicas de las invocaciones de Hook.

- En Include, especifique los nombres de las pilas que desee incluir. Úselo cuando desee segmentar un conjunto pequeño de pilas específicas. Solo las pilas especificadas en esta lista invocarán el Hook.
- En Exclude, especifique los nombres de las pilas que se van a excluir. Úselo cuando quiera invocar el Hook en la mayoría de las pilas, pero excluya algunas específicas. Todas las acumulaciones, excepto las que aparecen aquí, invocarán el Hook.

d. En el caso de las funciones de Stack, incluya o excluya pilas específicas de las invocaciones de Hook en función de sus funciones de IAM asociadas.

- En Include, especifique una o más funciones de IAM ARNs para centrarse en las pilas asociadas a estas funciones. Solo las operaciones de apilamiento iniciadas por estos roles invocarán el Hook.
- En Excluir, especifique una o más funciones de IAM ARNs para las pilas que desee excluir. El Hook se invocará en todas las pilas, excepto en las iniciadas por los roles especificados.

15. Elija Siguiente.
16. En la página Revisar y activar, revisa tus opciones. Para realizar cambios, elija Editar en la sección correspondiente.
17. Cuando estés listo para continuar, selecciona Activar Hook.

Activa un gancho de protección (AWS CLI)

Antes de continuar, confirma que has creado la regla de guardia y la función de ejecución que utilizarás con este Hook. Para obtener más información, consulte Escribe las reglas de Guard para evaluar los recursos de Guard Hooks y Crea un rol de ejecución para un Guard Hook.

Para activar un Guard Hook para usarlo en tu cuenta (AWS CLI)

1. Para empezar a activar un Hook, usa el siguiente activate-typecomando y reemplaza los marcadores de posición por tus valores específicos. Este comando autoriza al Hook a utilizar una función de ejecución específica suya. Cuenta de AWS

```
aws cloudformation activate-type --type HOOK \
  --type-name AWS::Hooks::GuardHook \
  --publisher-id aws-hooks \
  --type-name-alias Private::Guard::MyTestHook \
```

```
  --execution-role-arn arn:aws:iam::123456789012:role/my-execution-role \
  --region us-west-2
```

2. Para terminar de activar el Hook, debes configurarlo mediante un archivo de configuración JSON.

   Usa el cat comando para crear un archivo JSON con la siguiente estructura. Para obtener más información, consulte Referencia a la sintaxis del esquema de la configuración del enlace.

```
$ cat > config.json
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE",
        "CHANGE_SET"
      ],
      "FailureMode": "WARN",
      "Properties": {
        "ruleLocation": "s3://amzn-s3-demo-bucket/MyGuardRules.guard",
        "logBucket": "amzn-s3-demo-logging-bucket"
      },
      "TargetFilters": {
        "Actions": [
          "CREATE",
          "UPDATE",
          "DELETE"
        ]
      }
    }
  }
}
```

   - `HookInvocationStatus`: `ENABLED` Configúrelo en para habilitar el Hook.
   - `TargetOperations`: especifique las operaciones que evaluará el Hook.
   - `FailureMode`: se establece en `FAIL` o `WARN`.
   - `ruleLocation`: Sustitúyalo por el URI de S3 en el que está almacenada la regla. El objeto almacenado en S3 debe tener una de las siguientes extensiones de archivo: `.guard.zip`, `y.tar.gz`.

- `logBucket`: (Opcional) Especifique el nombre de un bucket de S3 para los informes JSON de Guard.
- `TargetFilters`: especifique los tipos de acciones que invocarán el Hook.

3. Usa el siguiente set-type-configurationcomando, junto con el archivo JSON que has creado, para aplicar la configuración. Sustituya los marcadores de posición por sus valores específicos.

```
aws cloudformation set-type-configuration \
  --configuration file://config.json \
  --type-arn "arn:aws:cloudformation:us-west-2:123456789012:type/hook/MyTestHook" \
  --region us-west-2
```

Recursos relacionados

Proporcionamos ejemplos de plantillas que puedes usar para entender cómo declarar un Guard Hook en una plantilla de CloudFormation pila. Para obtener más información, consulte AWS::CloudFormation::GuardHook en la Guía del usuario de AWS CloudFormation .

Consulta los registros de los Guard Hooks de tu cuenta

Al activar un Guard Hook, puede especificar un bucket de Amazon S3 como destino del informe de resultados de Hook. Una vez activado, el Hook almacena automáticamente los resultados de las validaciones de las reglas de Guard en el depósito especificado. A continuación, podrá ver estos resultados en la consola de Amazon S3.

Ver los registros de Guard Hook en la consola Amazon S3

Para ver el archivo de registro de salida de Guard Hook

1. Inicie sesión en. https://console.aws.amazon.com/s3/
2. En la barra de navegación de la parte superior de la pantalla, elija la Región de AWS.
3. Elige Buckets.
4. Elija el depósito que seleccionó para su informe de salida de Guard.
5. Elija el archivo de registro del informe de salida de validación deseado.
6. Elija si desea descargar el archivo o abrirlo para verlo.

Elimina Guard Hooks de tu cuenta

Cuando ya no necesites un Guard Hook activado, utiliza los siguientes procedimientos para eliminarlo de tu cuenta.

Para desactivar temporalmente un Hook en lugar de eliminarlo, consultaDesactivar y activar AWS CloudFormation Hooks.

Temas

- Elimina un Guard Hook de tu cuenta (consola)
- Elimina un Guard Hook de tu cuenta (AWS CLI)

Elimina un Guard Hook de tu cuenta (consola)

Para eliminar un Guard Hook de tu cuenta

1. Inicia sesión AWS Management Console y abre la AWS CloudFormation consola en https://console.aws.amazon.com/cloudformation.
2. En la barra de navegación de la parte superior de la pantalla, selecciona la Región de AWS ubicación del Hook.
3. En el panel de navegación, selecciona Hooks.
4. En la página Hooks, busca el Guard Hook que deseas eliminar.
5. Selecciona la casilla de verificación situada junto a tu Hook y selecciona Eliminar.
6. Cuando se te pida confirmación, escribe el nombre del Hook para confirmar la eliminación del Hook especificado y, a continuación, selecciona Eliminar.

Elimina un Guard Hook de tu cuenta (AWS CLI)

> **Note**
>
> Antes de poder eliminar el gancho, primero debes deshabilitarlo. Para obtener más información, consulte Desactiva y activa un Hook en tu cuenta (AWS CLI).

Usa el siguiente deactivate-typecomando para desactivar un Hook, lo que lo eliminará de tu cuenta. Sustituye los marcadores de posición por tus valores específicos.

```
aws cloudformation deactivate-type \
  --type-arn "arn:aws:cloudformation:us-west-2:123456789012:type/hook/MyTestHook" \
  --region us-west-2
```

Enlaces de Lambda

Para usar un AWS Lambda Hook en tu cuenta, primero debes activar el Hook para la cuenta y la región en la que quieras usarlo. Al activar un Hook, podrás utilizarlo en las operaciones de apilamiento en la cuenta y la región en las que esté activado.

Cuando activas un Hook Lambda, CloudFormation crea una entrada en el registro de tu cuenta para el Hook activado como Hook privado. Esto le permite establecer cualquier propiedad de configuración que incluya el Hook. Las propiedades de configuración definen cómo se configura el Hook para una región determinada Cuenta de AWS .

Temas

- AWS CLI comandos para trabajar con Lambda Hooks
- Cree funciones Lambda para evaluar los recursos de Lambda Hooks
- Prepárese para crear un Hook Lambda
- Active un Lambda Hook en su cuenta
- Ver los registros de los Lambda Hooks de su cuenta
- Elimine Lambda Hooks de su cuenta

AWS CLI comandos para trabajar con Lambda Hooks

Los AWS CLI comandos para trabajar con Lambda Hooks incluyen:

- activate-typepara iniciar el proceso de activación de una Lambda Hook.
- set-type-configurationpara especificar los datos de configuración de un Hook en su cuenta.
- list-typespara incluir los Hooks de tu cuenta.
- describe-typepara obtener información detallada sobre un Hook específico o una versión específica de Hook, incluidos los datos de configuración actuales.
- deactivate-typepara eliminar un Hook previamente activado de tu cuenta.

Cree funciones Lambda para evaluar los recursos de Lambda Hooks

AWS CloudFormation Lambda Hooks le permite evaluar CloudFormation y realizar API de control de nube de AWS operaciones con su propio código personalizado. Su Hook puede impedir que se lleve a cabo una operación o emitir una advertencia a la persona que llama y permitir que la operación continúe. Al crear un Hook Lambda, puede configurarlo para interceptar y evaluar las siguientes operaciones: CloudFormation

- Operaciones de recursos
- Operaciones de apilamiento
- Operaciones del conjunto de cambios

Temas

Desarrollo de un gancho Lambda

Cuando Hooks invoque tu Lambda, esperará hasta 30 segundos para que la Lambda evalúe la entrada. La Lambda devolverá una respuesta JSON que indica si el Hook se realizó correctamente o no.

Temas

Solicita una entrada

La entrada que se pasa a la función Lambda depende de la operación de destino de Hook (ejemplos: pila, recurso o conjunto de cambios).

Entrada de respuesta

Para poder comunicar a Hooks si tu solicitud se ha realizado correctamente o no, tu función Lambda debe devolver una respuesta JSON.

El siguiente es un ejemplo de la forma de la respuesta que espera Hooks:

```
{
  "HookStatus": "SUCCESS" or "FAILED" or "IN_PROGRESS",
  "errorCode": "NonCompliant" or "InternalFailure"
  "message": String,
  "clientRequestToken": String
  "CallbackContext": None,
  "callbackDelaySeconds": Integer,
}
```

HookStatus

El estado del Hook. Este campo es obligatorio.

Valores válidos: (`SUCCESS`| `FAILED` |`IN_PROGRESS`)

> **Note**
>
> Un Hook puede volver `IN_PROGRESS` 3 veces. Si no se devuelve ningún resultado, el Hook fallará. En el caso de un Lambda Hook, esto significa que la función Lambda se puede invocar hasta 3 veces.

errorCode

Muestra si la operación se evaluó y se determinó que no era válida, o si se produjeron errores en el Hook que impidieron la evaluación. Este campo es obligatorio si el Hook falla.

Valores válidos: (`NonCompliant`|`InternalFailure`)

message

El mensaje a la persona que llama en el que se indica por qué el Hook tuvo éxito o falló.

> **Note**
>
> Al evaluar CloudFormation las operaciones, este campo se trunca a 4096 caracteres.

Al evaluar las operaciones de la API de Cloud Control, este campo se trunca a 1024 caracteres.

clientRequestToken

El token de solicitud que se proporcionó como entrada a la solicitud de Hook. Este campo es obligatorio.

CallbackContext

Si lo indica, pasa un contexto adicional que `hookStatus` se `IN_PROGRESS` proporciona como entrada cuando se vuelve a invocar la función Lambda.

callbackDelaySeconds

Cuánto tiempo debe esperar Hooks para volver a invocar este Hook.

Ejemplos

El siguiente es un ejemplo de una respuesta exitosa:

```
{
  "hookStatus": "SUCCESS",
  "message": "compliant",
  "clientRequestToken": "123avjdjk31"
}
```

El siguiente es un ejemplo de una respuesta fallida:

```
{
  "hookStatus": "FAILED",
  "errorCode": "NonCompliant",
  "message": "S3 Bucket Versioning must be enabled.",
  "clientRequestToken": "123avjdjk31"
 }
```

Evaluación de las operaciones de recursos con Lambda Hooks

Cada vez que se crea, actualiza o elimina un recurso, se considera una operación de recurso. Por ejemplo, si ejecutas la actualización de una CloudFormation pila que crea un recurso nuevo, habrás

completado una operación de recursos. Cuando creas, actualizas o eliminas un recurso mediante la API de Cloud Control, también se considera una operación de recursos. Puede configurar su CloudFormation Lambda Hook para el destino `RESOURCE` y `CLOUD_CONTROL` las operaciones en la configuración de Hook`TargetOperations`.

> **Note**
>
> El controlador `delete` Hook solo se invoca cuando se elimina un recurso mediante un activador de operaciones de la API `delete-resource` de Cloud Control o. CloudFormation `delete-stack`

Temas

- Sintaxis de entrada de recursos de Lambda Hook
- Ejemplo de entrada de cambio de recursos de Lambda Hook
- Ejemplo de función Lambda para operaciones de recursos

Sintaxis de entrada de recursos de Lambda Hook

Cuando se invoque su Lambda para una operación de recursos, recibirá una entrada JSON que contiene las propiedades del recurso, las propiedades propuestas y el contexto en torno a la invocación de Hook.

El siguiente es un ejemplo de la forma de la entrada JSON:

```
{
    "awsAccountId": String,
    "stackId": String,
    "changeSetId": String,
    "hookTypeName": String,
    "hookTypeVersion": String,
    "hookModel": {
        "LambdaFunction": String
    },
    "actionInvocationPoint": "CREATE_PRE_PROVISION" or "UPDATE_PRE_PROVISION" or
 "DELETE_PRE_PROVISION"
    "requestData": {
        "targetName": String,
        "targetType": String,
```

```
        "targetLogicalId": String,
        "targetModel": {
            "resourceProperties": {...},
            "previousResourceProperties": {...}
        }
    },
    "requestContext": {
        "invocación": 1,
        "CallbackContext": null
    }
}
```

`awsAccountId`

El ID Cuenta de AWS que contiene el recurso que se está evaluando.

`stackId`

El ID de pila de la CloudFormation pila de la que forma parte esta operación. Este campo está vacío si la persona que llama es Cloud Control API.

`changeSetId`

El ID del conjunto de cambios que inició la invocación de Hook. Este valor está vacío si el cambio de recurso lo inició la API de Cloud Control o las `delete-stack` operaciones `create-stackupdate-stack`, o.

`hookTypeName`

El nombre del Hook que se está ejecutando.

`hookTypeVersion`

La versión del Hook que se está ejecutando.

`hookModel`

`LambdaFunction`

El ARN de Lambda actual invocado por el Hook.

`actionInvocationPoint`

El punto exacto de la lógica de aprovisionamiento en el que se ejecuta el Hook.

Valores válidos: (`CREATE_PRE_PROVISION`| `UPDATE_PRE_PROVISION` |`DELETE_PRE_PROVISION`)

`requestData`

`targetName`

El tipo de objetivo que se está evaluando, por ejemplo,`AWS::S3::Bucket`.

`targetType`

El tipo de objetivo que se está evaluando, por ejemplo`AWS::S3::Bucket`. En el caso de los recursos aprovisionados con la API de Cloud Control, este valor será`RESOURCE`.

`targetLogicalId`

El ID lógico del recurso que se está evaluando. Si el origen de la invocación de Hook es CloudFormation, será el ID de recurso lógico definido en la CloudFormation plantilla. Si el origen de esta invocación de Hook es la API de Cloud Control, será un valor generado.

`targetModel`

`resourceProperties`

Las propiedades propuestas del recurso que se está modificando. Si se elimina el recurso, este valor estará vacío.

`previousResourceProperties`

Las propiedades que están asociadas actualmente al recurso que se está modificando. Si se está creando el recurso, este valor estará vacío.

`requestContext`

invocación

El intento actual de ejecutar el Hook.

CallbackContext

Si el Hook se configuró en y `callbackContext` se devolvió`IN_PROGRESS`, estará aquí después de volver a invocarlo.

Ejemplo de entrada de cambio de recursos de Lambda Hook

El siguiente ejemplo de entrada muestra un Hook Lambda que recibirá la definición del `AWS::DynamoDB::Table` recurso que `ProvisionedThroughput` se va a actualizar, donde el valor `ReadCapacityUnits` de cambia de 3 a 10. Estos son los datos de los que dispone Lambda para su evaluación.

```
{
    "awsAccountId": "123456789012",
    "stackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/
MyStack/1a2345b6-0000-00a0-a123-00abc0abc000",
    "hookTypeName": "my::lambda::resourcehookfunction",
    "hookTypeVersion": "00000008",
    "hookModel": {
        "LambdaFunction": "arn:aws:lambda:us-west-2:123456789012:function:MyFunction"
    },
    "actionInvocationPoint": "UPDATE_PRE_PROVISION",
    "requestData": {
        "targetName": "AWS::DynamoDB::Table",
        "targetType": "AWS::DynamoDB::Table",
        "targetLogicalId": "DDBTable",
        "targetModel": {
            "resourceProperties": {
                "AttributeDefinitions": [
                    {
                        "AttributeType": "S",
                        "AttributeName": "Album"
                    },
                    {
                        "AttributeType": "S",
                        "AttributeName": "Artist"
                    }
                ],
                "ProvisionedThroughput": {
                    "WriteCapacityUnits": 5,
                    "ReadCapacityUnits": 10
                },
                "KeySchema": [
                    {
                        "KeyType": "HASH",
                        "AttributeName": "Album"
                    },
                    {
                        "KeyType": "RANGE",
                        "AttributeName": "Artist"
                    }
                ]
            },
            "previousResourceProperties": {
                "AttributeDefinitions": [
```

```
                    {
                        "AttributeType": "S",
                        "AttributeName": "Album"
                    },
                    {
                        "AttributeType": "S",
                        "AttributeName": "Artist"
                    }
                ],
                "ProvisionedThroughput": {
                    "WriteCapacityUnits": 5,
                    "ReadCapacityUnits": 5
                },
                "KeySchema": [
                    {
                        "KeyType": "HASH",
                        "AttributeName": "Album"
                    },
                    {
                        "KeyType": "RANGE",
                        "AttributeName": "Artist"
                    }
                ]
            }
        }
    },
    "requestContext": {
        "invocation": 1,
        "callbackContext": null
    }
}
```

Para ver todas las propiedades disponibles para el tipo de recurso, consulte AWS::DynamoDB::Table.

Ejemplo de función Lambda para operaciones de recursos

La siguiente es una función sencilla que produce errores en cualquier actualización de recursos de DynamoDB, que intenta establecer `ReadCapacity` el valor `ProvisionedThroughput` de en un valor superior a 10. Si el Hook funciona correctamente, la persona que llama verá el mensaje «ReadCapacity está correctamente configurado». Si la solicitud no pasa la validación, el Hook fallará con el estado «ReadCapacity no puede ser más de 10».

Node.js

```
export const handler = async (event, context) => {
    var targetModel = event?.requestData?.targetModel;
    var targetName = event?.requestData?.targetName;
    var response = {
        "hookStatus": "SUCCESS",
        "message": "ReadCapacity is correctly configured.",
        "clientRequestToken": event.clientRequestToken
    };

    if (targetName == "AWS::DynamoDB::Table") {
        var readCapacity =
 targetModel?.resourceProperties?.ProvisionedThroughput?.ReadCapacityUnits;
        if (readCapacity > 10) {
            response.hookStatus = "FAILED";
            response.errorCode = "NonCompliant";
            response.message = "ReadCapacity must be cannot be more than 10.";
        }
    }
    return response;
};
```

Python

```
import json

def lambda_handler(event, context):
    # Using dict.get() for safe access to nested dictionary values
    request_data = event.get('requestData', {})
    target_model = request_data.get('targetModel', {})
    target_name = request_data.get('targetName', '')

    response = {
        "hookStatus": "SUCCESS",
        "message": "ReadCapacity is correctly configured.",
        "clientRequestToken": event.get('clientRequestToken')
    }

    if target_name == "AWS::DynamoDB::Table":
        # Safely navigate nested dictionary
        resource_properties = target_model.get('resourceProperties', {})
```

```
        provisioned_throughput = resource_properties.get('ProvisionedThroughput',
{})
        read_capacity = provisioned_throughput.get('ReadCapacityUnits')

        if read_capacity and read_capacity > 10:
            response['hookStatus'] = "FAILED"
            response['errorCode'] = "NonCompliant"
            response['message'] = "ReadCapacity must be cannot be more than 10."

    return response
```

Evaluación de las operaciones de apilamiento con Lambda Hooks

Cada vez que cree, actualice o elimine una pila con una plantilla nueva, puede configurar su CloudFormation Lambda Hook para que comience por evaluar la nueva plantilla y, potencialmente, bloquear la operación de apilamiento para que no continúe. Puede configurar su CloudFormation Lambda Hook para que se dirija a `STACK` las operaciones en la configuración de Hook`TargetOperations`.

Temas

- Sintaxis de entrada de la pila Lambda Hook
- Ejemplo de entrada de cambio de pila de Lambda Hook
- Ejemplo de función Lambda para operaciones de apilamiento

Sintaxis de entrada de la pila Lambda Hook

Cuando se invoque su Lambda para una operación de apilamiento, recibirá una solicitud JSON que contiene el contexto de invocación de Hook y el contexto de solicitud. `actionInvocationPoint` Debido al tamaño de las CloudFormation plantillas y al tamaño de entrada limitado que aceptan las funciones de Lambda, las plantillas reales se almacenan en un objeto de Amazon S3. La entrada `requestData` incluye una URL resignada de Amazon S3 a otro objeto, que contiene la versión de la plantilla actual y la anterior.

El siguiente es un ejemplo de la forma de la entrada JSON:

```
{
    "clientRequesttoken": String,
    "awsAccountId": String,
    "stackID": String,
```

```
    "changeSetId": String,
    "hookTypeName": String,
    "hookTypeVersion": String,
    "hookModel": {
        "LambdaFunction":String
    },
    "actionInvocationPoint": "CREATE_PRE_PROVISION" or "UPDATE_PRE_PROVISION" or
 "DELETE_PRE_PROVISION"
    "requestData": {
        "targetName": "STACK",
        "targetType": "STACK",
        "targetLogicalId": String,
        "payload": String (S3 Presigned URL)
    },
    "requestContext": {
        "invocation": Integer,
        "callbackContext": String
    }
}
```

`clientRequesttoken`

El token de solicitud que se proporcionó como entrada a la solicitud de Hook. Este campo es obligatorio.

`awsAccountId`

El ID del elemento Cuenta de AWS que contiene la pila que se está evaluando.

`stackID`

El ID de pila de la CloudFormation pila.

`changeSetId`

El ID del conjunto de cambios que inició la invocación de Hook. Este valor está vacío si el cambio de pila lo inició la API de Cloud Control o las `delete-stack` operaciones `create-stackupdate-stack`, o.

`hookTypeName`

El nombre del Hook que se está ejecutando.

`hookTypeVersion`

La versión del Hook que se está ejecutando.

`hookModel`

`LambdaFunction`

El ARN de Lambda actual invocado por el Hook.

`actionInvocationPoint`

El punto exacto de la lógica de aprovisionamiento en el que se ejecuta el Hook.

Valores válidos: (`CREATE_PRE_PROVISION`| `UPDATE_PRE_PROVISION` |`DELETE_PRE_PROVISION`)

`requestData`

`targetName`

Este valor será `STACK`.

`targetType`

Este valor será `STACK`.

`targetLogicalId`

El nombre de la pila.

`payload`

La URL prefirmada de Amazon S3 que contiene un objeto JSON con las definiciones de plantilla actuales y anteriores.

`requestContext`

Si se vuelve a invocar el Hook, se establecerá este objeto.

`invocation`

El intento actual de ejecutar el Hook.

`callbackContext`

Si el Hook estaba activado `IN_PROGRESS` y `callbackContext` se devolvió, estará aquí al volver a invocarlo.

La `payload` propiedad de los datos de la solicitud es una URL que el código debe recuperar. Una vez que haya recibido la URL, obtendrás un objeto con el siguiente esquema:

```
{
    "template": String,
    "previousTemplate": String
}
```

`template`

La CloudFormation plantilla completa que se proporcionó a `create-stack` o`update-stack`. Puede ser una cadena JSON o YAML en función de lo que se haya proporcionado a CloudFormation.

En `delete-stack` las operaciones, este valor estará vacío.

`previousTemplate`

La CloudFormation plantilla anterior. Puede ser una cadena JSON o YAML en función de lo que se le CloudFormation haya proporcionado.

En `delete-stack` las operaciones, este valor estará vacío.

Ejemplo de entrada de cambio de pila de Lambda Hook

El siguiente es un ejemplo de entrada de cambio de pila. The Hook está evaluando un cambio que lo actualiza `ObjectLockEnabled` a verdadero y añade una cola de Amazon SQS:

```
{
    "clientRequestToken": "f8da6d11-b23f-48f4-814c-0fb6a667f50e",
    "awsAccountId": "123456789012",
    "stackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/
MyStack/1a2345b6-0000-00a0-a123-00abc0abc000",
    "changeSetId": null,
    "hookTypeName": "my::lambda::stackhook",
    "hookTypeVersion": "00000008",
    "hookModel": {
        "LambdaFunction": "arn:aws:lambda:us-west-2:123456789012:function:MyFunction"
    },
    "actionInvocationPoint": "UPDATE_PRE_PROVISION",
    "requestData": {
        "targetName": "STACK",
        "targetType": "STACK",
        "targetLogicalId": "my-cloudformation-stack",
        "payload": "https://s3......"
```

```
    },
    "requestContext": {
        "invocation": 1,
        "callbackContext": null
    }
}
```

Este es un ejemplo `payload` de: `requestData`

```
{
    "template": "{\"Resources\":{\"S3Bucket\":{\"Type\":\"AWS::S3::Bucket\",
\"Properties\":{\"ObjectLockEnabled\":true}},\"SQSQueue\":{\"Type\":\"AWS::SQS::Queue
\",\"Properties\":{\"QueueName\":\"NewQueue\"}}}}",
    "previousTemplate": "{\"Resources\":{\"S3Bucket\":{\"Type\":\"AWS::S3::Bucket\",
\"Properties\":{\"ObjectLockEnabled\":false}}}}"
}
```

Ejemplo de función Lambda para operaciones de apilamiento

El siguiente ejemplo es una función sencilla que descarga la carga útil de la operación de apilado, analiza el JSON de la plantilla y devuelve la información. `SUCCESS`

Node.js

```
export const handler = async (event, context) => {
    var targetType = event?.requestData?.targetType;
    var payloadUrl = event?.requestData?.payload;

    var response = {
        "hookStatus": "SUCCESS",
        "message": "Stack update is compliant",
        "clientRequestToken": event.clientRequestToken
    };
    try {
        const templateHookPayloadRequest = await fetch(payloadUrl);
        const templateHookPayload = await templateHookPayloadRequest.json()
        if (templateHookPayload.template)  {
            // Do something with the template templateHookPayload.template
            // JSON or YAML
        }
        if (templateHookPayload.previousTemplate) {
            // Do something with the template templateHookPayload.previousTemplate
            // JSON or YAML
```

```
        }
    } catch (error) {
        console.log(error);
        response.hookStatus = "FAILED";
        response.message = "Failed to evaluate stack operation.";
        response.errorCode = "InternalFailure";
    }
    return response;
};
```

Python

Para usar Python, tendrás que importar la `requests` biblioteca. Para ello, tendrá que incluir la biblioteca en el paquete de despliegue al crear la función Lambda. Para obtener más información, consulte Creación de un paquete de despliegue en formato.zip con dependencias en la AWS Lambda Guía para desarrolladores.

```
import json
import requests

def lamnbda_handler(event, context):
    # Safely access nested dictionary values
    request_data = event.get('requestData', {})
    target_type = request_data.get('targetType')
    payload_url = request_data.get('payload')

    response = {
        "hookStatus": "SUCCESS",
        "message": "Stack update is compliant",
        "clientRequestToken": event.get('clientRequestToken')
    }

    try:
        # Fetch the payload
        template_hook_payload_request = requests.get(payload_url)
        template_hook_payload_request.raise_for_status()  # Raise an exception for
 bad responses
        template_hook_payload = template_hook_payload_request.json()

        if 'template' in template_hook_payload:
            # Do something with the template template_hook_payload['template']
            # JSON or YAML
            pass
```

```
        if 'previousTemplate' in template_hook_payload:
            # Do something with the template
template_hook_payload['previousTemplate']
            # JSON or YAML
            pass

    except Exception as error:
        print(error)
        response['hookStatus'] = "FAILED"
        response['message'] = "Failed to evaluate stack operation."
        response['errorCode'] = "InternalFailure"

    return response
```

Evaluación de las operaciones de los conjuntos de cambios con Lambda Hooks

Cada vez que cree un conjunto de cambios, puede configurar su CloudFormation Lambda Hook para que evalúe primero el nuevo conjunto de cambios y, potencialmente, bloquee su ejecución. Puede configurar su CloudFormation Lambda Hook para que se dirija a `CHANGE_SET` las operaciones en la configuración de Hook`TargetOperations`.

Temas

- Sintaxis de entrada del conjunto de cambios de Lambda Hook
- Ejemplo de entrada de cambio de conjunto de cambios de Lambda Hook
- Ejemplo de función Lambda para operaciones de conjuntos de cambios

Sintaxis de entrada del conjunto de cambios de Lambda Hook

La entrada para las operaciones del conjunto de cambios es similar a la de las operaciones de apilamiento, pero la carga útil del conjunto de cambios `requestData` también incluye una lista de los cambios de recursos introducidos por el conjunto de cambios.

A continuación se muestra un ejemplo de la forma de la entrada JSON:

```
{
    "clientRequesttoken": String,
    "awsAccountId": String,
    "stackID": String,
```

```
    "changeSetId": String,
    "hookTypeName": String,
    "hookTypeVersion": String,
    "hookModel": {
        "LambdaFunction":String
    },
    "requestData": {
        "targetName": "CHANGE_SET",
        "targetType": "CHANGE_SET",
        "targetLogicalId": String,
        "payload": String (S3 Presigned URL)
    },
    "requestContext": {
        "invocation": Integer,
        "callbackContext": String
    }
}
```

`clientRequesttoken`

El token de solicitud que se proporcionó como entrada a la solicitud de Hook. Este campo es obligatorio.

`awsAccountId`

El ID del elemento Cuenta de AWS que contiene la pila que se está evaluando.

`stackID`

El ID de pila de la CloudFormation pila.

`changeSetId`

El ID del conjunto de cambios que inició la invocación de Hook.

`hookTypeName`

El nombre del Hook que se está ejecutando.

`hookTypeVersion`

La versión del Hook que se está ejecutando.

`hookModel`

`LambdaFunction`

El ARN de Lambda actual invocado por el Hook.

`requestData`

`targetName`

Este valor será `CHANGE_SET`.

`targetType`

Este valor será `CHANGE_SET`.

`targetLogicalId`

El conjunto de cambios ARN..

`payload`

La URL prefirmada de Amazon S3 que contiene un objeto JSON con la plantilla actual, así como una lista de los cambios introducidos por este conjunto de cambios.

`requestContext`

Si se vuelve a invocar el Hook, se establecerá este objeto.

`invocation`

El intento actual de ejecutar el Hook.

`callbackContext`

Si el Hook estaba activado `IN_PROGRESS` y `callbackContext` se devolvió, estará aquí al volver a invocarlo.

La `payload` propiedad de los datos de la solicitud es una URL que el código debe recuperar. Una vez que haya recibido la URL, obtendrás un objeto con el siguiente esquema:

```
{
    "template": String,
    "changedResources": [
        {
            "action": String,
            "beforeContext": JSON String,
            "afterContext": JSON String,
            "lineNumber": Integer,
            "logicalResourceId": String,
            "resourceType": String
```

```
        }
    ]
}
```

`template`

La CloudFormation plantilla completa que se proporcionó a `create-stack` o`update-stack`. Puede ser una cadena JSON o YAML en función de lo que se haya proporcionado a CloudFormation.

`changedResources`

Una lista de los recursos modificados.

`action`

El tipo de cambio aplicado al recurso.

Valores válidos: (`CREATE`| `UPDATE` |`DELETE`)

`beforeContext`

Una cadena JSON de las propiedades del recurso antes del cambio. Este valor es nulo cuando se crea el recurso. Todos los valores booleanos y numéricos de esta cadena JSON son CADENAS.

`afterContext`

Una cadena JSON de las propiedades de los recursos si se ejecuta este conjunto de cambios. Este valor es nulo cuando se elimina el recurso. Todos los valores booleanos y numéricos de esta cadena JSON son CADENAS.

`lineNumber`

El número de línea de la plantilla que provocó este cambio. Si la acción es, `DELETE` este valor será nulo.

`logicalResourceId`

El identificador de recurso lógico del recurso que se va a cambiar.

`resourceType`

El tipo de recurso que se va a cambiar.

Ejemplo de entrada de cambio de conjunto de cambios de Lambda Hook

A continuación se muestra un ejemplo de entrada de cambio de conjunto. En el siguiente ejemplo, puede ver los cambios introducidos por el conjunto de cambios. El primer cambio consiste en eliminar una cola llamada`CoolQueue`. El segundo cambio consiste en añadir una nueva cola llamada. `NewCoolQueue` El último cambio es una actualización de`DynamoDBTable`.

```
{
    "clientRequestToken": "f8da6d11-b23f-48f4-814c-0fb6a667f50e",
    "awsAccountId": "123456789012",
    "stackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/
MyStack/1a2345b6-0000-00a0-a123-00abc0abc000",
    "changeSetId": "arn:aws:cloudformation:us-west-2:123456789012:changeSet/
SampleChangeSet/1a2345b6-0000-00a0-a123-00abc0abc000",
    "hookTypeName": "my::lambda::changesethook",
    "hookTypeVersion": "00000008",
    "hookModel": {
        "LambdaFunction": "arn:aws:lambda:us-west-2:123456789012:function:MyFunction"
    },
    "actionInvocationPoint": "CREATE_PRE_PROVISION",
    "requestData": {
        "targetName": "CHANGE_SET",
        "targetType": "CHANGE_SET",
        "targetLogicalId": "arn:aws:cloudformation:us-west-2:123456789012:changeSet/
SampleChangeSet/1a2345b6-0000-00a0-a123-00abc0abc000",
        "payload": "https://s3......"
    },
    "requestContext": {
        "invocation": 1,
        "callbackContext": null
    }
}
```

Este es un ejemplo `payload` de`requestData.payload`:

```
{
  template: 'Resources:\n' +
    '  DynamoDBTable:\n' +
    '    Type: AWS::DynamoDB::Table\n' +
    '    Properties:\n' +
    '      AttributeDefinitions:\n' +
    '        - AttributeName: "PK"\n' +
    '          AttributeType: "S"\n' +
```

```
    '      BillingMode: "PAY_PER_REQUEST"\n' +
    '      KeySchema:\n' +
    '        - AttributeName: "PK"\n' +
    '          KeyType: "HASH"\n' +
    '      PointInTimeRecoverySpecification:\n' +
    '        PointInTimeRecoveryEnabled: false\n' +
    '  NewSQSQueue:\n' +
    '    Type: AWS::SQS::Queue\n' +
    '    Properties:\n' +
    '      QueueName: "NewCoolQueue"',
  changedResources: [
    {
      logicalResourceId: 'SQSQueue',
      resourceType: 'AWS::SQS::Queue',
      action: 'DELETE',
      lineNumber: null,
      beforeContext: '{"Properties":{"QueueName":"CoolQueue"}}',
      afterContext: null
    },
    {
      logicalResourceId: 'NewSQSQueue',
      resourceType: 'AWS::SQS::Queue',
      action: 'CREATE',
      lineNumber: 14,
      beforeContext: null,
      afterContext: '{"Properties":{"QueueName":"NewCoolQueue"}}'
    },
    {
      logicalResourceId: 'DynamoDBTable',
      resourceType: 'AWS::DynamoDB::Table',
      action: 'UPDATE',
      lineNumber: 2,
      beforeContext: '{"Properties":
{"BillingMode":"PAY_PER_REQUEST","AttributeDefinitions":
[{"AttributeType":"S","AttributeName":"PK"}],"KeySchema":
[{"KeyType":"HASH","AttributeName":"PK"}]}}',
      afterContext: '{"Properties":
{"BillingMode":"PAY_PER_REQUEST","PointInTimeRecoverySpecification":
{"PointInTimeRecoveryEnabled":"false"},"AttributeDefinitions":
[{"AttributeType":"S","AttributeName":"PK"}],"KeySchema":
[{"KeyType":"HASH","AttributeName":"PK"}]}}'
    }
  ]
```

```
}
```

Ejemplo de función Lambda para operaciones de conjuntos de cambios

El siguiente ejemplo es una función sencilla que descarga la carga útil de la operación del conjunto de cambios, recorre cada cambio y, a continuación, imprime las propiedades anteriores y posteriores antes de devolver un. SUCCESS

Node.js

```
export const handler = async (event, context) => {
    var payloadUrl = event?.requestData?.payload;
    var response = {
        "hookStatus": "SUCCESS",
        "message": "Change set changes are compliant",
        "clientRequestToken": event.clientRequestToken
    };
    try {
        const changeSetHookPayloadRequest = await fetch(payloadUrl);
        const changeSetHookPayload = await changeSetHookPayloadRequest.json();
        const changes = changeSetHookPayload.changedResources || [];
        for(const change of changes) {
            var beforeContext = {};
            var afterContext = {};
            if(change.beforeContext) {
                beforeContext = JSON.parse(change.beforeContext);
            }
            if(change.afterContext) {
                afterContext = JSON.parse(change.afterContext);
            }
            console.log(beforeContext)
            console.log(afterContext)
            // Evaluate Change here
        }
    } catch (error) {
        console.log(error);
        response.hookStatus = "FAILED";
        response.message = "Failed to evaluate change set operation.";
        response.errorCode = "InternalFailure";
    }
    return response;
};
```

Python

Para usar Python, tendrás que importar la `requests` biblioteca. Para ello, tendrá que incluir la biblioteca en el paquete de despliegue al crear la función Lambda. Para obtener más información, consulte Creación de un paquete de despliegue en formato.zip con dependencias en la AWS Lambda Guía para desarrolladores.

```
import json
import requests

def lambda_handler(event, context):
    payload_url = event.get('requestData', {}).get('payload')
    response = {
        "hookStatus": "SUCCESS",
        "message": "Change set changes are compliant",
        "clientRequestToken": event.get('clientRequestToken')
    }

    try:
        change_set_hook_payload_request = requests.get(payload_url)
        change_set_hook_payload_request.raise_for_status()  # Raises an HTTPError
 for bad responses
        change_set_hook_payload = change_set_hook_payload_request.json()

        changes = change_set_hook_payload.get('changedResources', [])

        for change in changes:
            before_context = {}
            after_context = {}

            if change.get('beforeContext'):
                before_context = json.loads(change['beforeContext'])

            if change.get('afterContext'):
                after_context = json.loads(change['afterContext'])

            print(before_context)
            print(after_context)
            # Evaluate Change here

    except requests.RequestException as error:
        print(error)
        response['hookStatus'] = "FAILED"
```

```
        response['message'] = "Failed to evaluate change set operation."
        response['errorCode'] = "InternalFailure"
    except json.JSONDecodeError as error:
        print(error)
        response['hookStatus'] = "FAILED"
        response['message'] = "Failed to parse JSON payload."
        response['errorCode'] = "InternalFailure"

    return response
```

Prepárese para crear un Hook Lambda

Antes de crear un Hook Lambda, debe cumplir los siguientes requisitos previos:

- Debe haber creado ya una función Lambda. Para obtener más información, consulte la Creación de funciones Lambda para Hooks.
- El usuario o rol que crea el Hook debe tener permisos suficientes para activar los Hooks.
- Para usar el AWS CLI o un SDK para crear un enlace de Lambda, debe crear manualmente un rol de ejecución con permisos de IAM y una política de confianza que permita CloudFormation invocar un enlace de Lambda.

Crear un rol de ejecución para un Hook Lambda

Un Hook usa un rol de ejecución para los permisos que necesita para invocar ese Hook en tu cuenta. Cuenta de AWS

Este rol se puede crear automáticamente si crea un Lambda Hook desde el AWS Management Console; de lo contrario, debe crear este rol usted mismo.

En la siguiente sección, se muestra cómo configurar los permisos para crear su Lambda Hook.

Permisos necesarios

Siga las instrucciones sobre cómo crear un rol mediante políticas de confianza personalizadas de la Guía del usuario de IAM para crear un rol con una política de confianza personalizada.

A continuación, complete los siguientes pasos para configurar los permisos:

1. Adjunte la siguiente política de privilegios mínimos a la función de IAM que desee utilizar para crear el Lambda Hook.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "lambda:InvokeFunction",
      "Resource": "arn:aws:lambda:us-west-2:123456789012:function:MyFunction"
    }
  ]
}
```

2. Dale permiso a tu Hook para que asuma la función añadiendo una política de confianza a la función. A continuación se muestra un ejemplo de política de confianza que puedes usar.

   JSON

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

Active un Lambda Hook en su cuenta

En el siguiente tema, se muestra cómo activar un Lambda Hook en su cuenta para que pueda utilizarse en la cuenta y la región en las que se activó.

Temas

Activar un Lambda Hook (consola)

Para activar un Lambda Hook para usarlo en su cuenta

1. Inicie sesión en AWS Management Console https://console.aws.amazon.com/cloudformation y abra la AWS CloudFormation consola.
2. En la barra de navegación de la parte superior de la pantalla, elige el Región de AWS lugar donde quieres crear el Hook in.
3. Si no ha creado una función Lambda para el Hook, haga lo siguiente:

   - Abra la página de Funciones en la consola de Lambda.
   - Cree la función Lambda que utilizará con este Hook y, a continuación, vuelva a este procedimiento. Para obtener más información, consulte Cree funciones Lambda para evaluar los recursos de Lambda Hooks.

   Si ya ha creado la función Lambda, continúe con el paso siguiente.
4. En el panel de navegación de la izquierda, selecciona Hooks.
5. En Nombre del gancho, elige una de las siguientes opciones:

   - Proporcione un nombre breve y descriptivo que se añadirá después`Private::Lambda::`. Por ejemplo, si escribes*`MyTestHook`*, el nombre completo de Hook pasa a ser`Private::Lambda::`*`MyTestHook`*.
   - Introduce el nombre completo del Hook (también denominado alias) con este formato: *`Provider`*`::`*`ServiceName`*`::`*`HookName`*
6. Para la función Lambda, proporcione la función Lambda que se utilizará con este Hook. Puede usar:

   - El nombre completo del recurso de Amazon (ARN) sin sufijo.
   - Un ARN cualificado con una versión o un sufijo de alias.
7. En el caso de los objetivos de Hook, elige qué quieres evaluar:

- Pilas: evalúa las plantillas de pilas cuando los usuarios crean, actualizan o eliminan pilas.
- Recursos: evalúa los cambios en los recursos individuales cuando los usuarios actualizan las pilas.
- Conjuntos de cambios: evalúa las actualizaciones planificadas cuando los usuarios crean conjuntos de cambios.
- API de Cloud Control: evalúa las operaciones de creación, actualización o eliminación iniciadas por la API de Cloud Control.

8. En Acciones, elige qué acciones (crear, actualizar, eliminar) invocarán tu Hook.
9. Para el modo Hook, elija cómo responde el Hook cuando la función Lambda invocada por el Hook devuelva una `FAILED` respuesta:

   - Advertir: emite advertencias a los usuarios, pero permite que las acciones continúen. Esto resulta útil para validaciones o comprobaciones informativas no críticas.
   - Error: impide que la acción continúe. Esto resulta útil para aplicar políticas de seguridad o de cumplimiento estrictas.

10. En la función de ejecución, elija la función de IAM que asume el Hook para invocar la función Lambda. Puedes CloudFormation permitir que se cree automáticamente un rol de ejecución para ti o puedes especificar un rol que hayas creado.
11. Elija Siguiente.
12. (Opcional) Para los filtros Hook, haga lo siguiente:

    a. En el filtro de recursos, especifique qué tipos de recursos pueden invocar el Hook. Esto garantiza que el Hook solo se invoque para los recursos relevantes.

    b. En cuanto a los criterios de filtrado, elige la lógica para aplicar los filtros de nombre y rol de la pila:

       - Todos los nombres y roles de las pilas: The Hook solo se invocará cuando coincidan todos los filtros especificados.
       - Cualquier nombre y función de pila: se invocará el Hook si al menos uno de los filtros especificados coincide.

> **Note**
>
> En el caso de las operaciones de la API de Cloud Control, se ignoran todos los filtros de nombres y roles de las pilas.

c. En el caso de los nombres de pila, incluye o excluye pilas específicas de las invocaciones de Hook.

- En Include, especifique los nombres de las pilas que desee incluir. Úselo cuando desee segmentar un conjunto pequeño de pilas específicas. Solo las pilas especificadas en esta lista invocarán el Hook.
- En Exclude, especifique los nombres de las pilas que se van a excluir. Úselo cuando quiera invocar el Hook en la mayoría de las pilas, pero excluya algunas específicas. Todas las acumulaciones, excepto las que aparecen aquí, invocarán el Hook.

d. En el caso de las funciones de Stack, incluya o excluya pilas específicas de las invocaciones de Hook en función de sus funciones de IAM asociadas.

- En Include, especifique una o más funciones de IAM ARNs para centrarse en las pilas asociadas a estas funciones. Solo las operaciones de apilamiento iniciadas por estos roles invocarán el Hook.
- En Excluir, especifique una o más funciones de IAM ARNs para las pilas que desee excluir. El Hook se invocará en todas las pilas, excepto en las iniciadas por los roles especificados.

13. Elija Siguiente.
14. En la página Revisar y activar, revisa tus opciones. Para realizar cambios, elija Editar en la sección correspondiente.
15. Cuando estés listo para continuar, selecciona Activar Hook.

Activar un gancho Lambda ()AWS CLI

Antes de continuar, confirme que ha creado la función Lambda y el rol de ejecución que utilizará con este Hook. Para obtener más información, consulte Cree funciones Lambda para evaluar los recursos de Lambda Hooks y Crear un rol de ejecución para un Hook Lambda.

Para activar un Lambda Hook para usarlo en su cuenta ()AWS CLI

1. Para empezar a activar un Hook, utilice el siguiente activate-typecomando y sustituya los marcadores de posición por sus valores específicos. Este comando autoriza al Hook a utilizar una función de ejecución específica suya. Cuenta de AWS

```
aws cloudformation activate-type --type HOOK \
  --type-name AWS::Hooks::LambdaHook \
  --publisher-id aws-hooks \
  --execution-role-arn arn:aws:iam::123456789012:role/my-execution-role \
  --type-name-alias Private::Lambda::MyTestHook \
  --region us-west-2
```

2. Para terminar de activar el Hook, debes configurarlo mediante un archivo de configuración JSON.

   Usa el cat comando para crear un archivo JSON con la siguiente estructura. Para obtener más información, consulte Referencia a la sintaxis del esquema de la configuración del enlace.

```
$ cat > config.json
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "CLOUD_CONTROL"
      ],
      "FailureMode": "WARN",
      "Properties": {
        "LambdaFunction": "arn:aws:lambda:us-
west-2:123456789012:function:MyFunction"
      },
      "TargetFilters": {
        "Actions": [
          "CREATE",
          "UPDATE",
          "DELETE"
        ]
      }
    }
  }
}
```

- `HookInvocationStatus`: `ENABLED` Configúrelo en para habilitar el Hook.
- `TargetOperations`: especifique las operaciones que evaluará el Hook.
- `FailureMode`: se establece en `FAIL` o `WARN`.
- `LambdaFunction`: especifique el ARN de la función Lambda.
- `TargetFilters`: especifique los tipos de acciones que invocarán el Hook.

3. Usa el siguiente set-type-configurationcomando, junto con el archivo JSON que has creado, para aplicar la configuración. Sustituya los marcadores de posición por sus valores específicos.

```
aws cloudformation set-type-configuration \
  --configuration file://config.json \
  --type-arn "arn:aws:cloudformation:us-west-2:123456789012:type/hook/MyTestHook" \
  --region us-west-2
```

Recursos relacionados

Proporcionamos ejemplos de plantillas que puede utilizar para entender cómo declarar un Hook de Lambda en una plantilla de CloudFormation pila. Para obtener más información, consulte AWS::CloudFormation::LambdaHook en la Guía del usuario de AWS CloudFormation .

Ver los registros de los Lambda Hooks de su cuenta

Si utiliza un Lambda Hook, el archivo de registro del informe de resultados de validación se encuentra en la consola de Lambda.

Ver los registros de Lambda Hook en la consola de Lambda

Para ver el archivo de registro de resultados de Lambda Hook

1. Inicie sesión en la consola Lambda.
2. En la barra de navegación de la parte superior de la pantalla, elija la Región de AWS.
3. Elija Funciones.
4. Elija la función Lambda deseada.
5. Elija la pestaña Prueba.
6. Elija CloudWatch Logs Live Trail
7. Elija el menú desplegable y seleccione los grupos de registros que desee ver.

8. Elija Iniciar. El registro se mostrará en la ventana CloudWatch Logs Live Trail. Elija Ver en columnas o Ver en texto plano según sus preferencias.

    - Puede añadir más filtros a los resultados añadiéndolos en el campo Añadir patrón de filtro. Este campo le permite filtrar los resultados para incluir solo los eventos que coincidan con el patrón especificado.

Para obtener más información sobre la visualización de registros de funciones de Lambda, consulte Visualización de CloudWatch registros de funciones de Lambda.

Elimine Lambda Hooks de su cuenta

Cuando ya no necesite un Lambda Hook activado, utilice los siguientes procedimientos para eliminarlo de su cuenta.

Para deshabilitar temporalmente un Hook en lugar de eliminarlo, consulteDesactivar y activar AWS CloudFormation Hooks.

Temas

- Eliminar un Lambda Hook de su cuenta (consola)
- Eliminar un Lambda Hook de su cuenta ()AWS CLI

Eliminar un Lambda Hook de su cuenta (consola)

Para eliminar un Lambda Hook de su cuenta

1. Inicie sesión en AWS Management Console https://console.aws.amazon.com/cloudformation y abra la AWS CloudFormation consola.
2. En la barra de navegación de la parte superior de la pantalla, selecciona la Región de AWS ubicación del Hook.
3. En el panel de navegación, selecciona Hooks.
4. En la página Hooks, busca el Hook Lambda que deseas eliminar.
5. Selecciona la casilla de verificación situada junto a tu Hook y selecciona Eliminar.
6. Cuando se te pida confirmación, escribe el nombre del Hook para confirmar la eliminación del Hook especificado y, a continuación, selecciona Eliminar.

Eliminar un Lambda Hook de su cuenta ()AWS CLI

> **Note**
>
> Antes de poder eliminar el Hook, primero debe deshabilitarlo. Para obtener más información, consulte Desactiva y activa un Hook en tu cuenta (AWS CLI).

Usa el siguiente deactivate-typecomando para desactivar un Hook, lo que lo eliminará de tu cuenta. Sustituye los marcadores de posición por tus valores específicos.

```
aws cloudformation deactivate-type \
  --type-arn "arn:aws:cloudformation:us-west-2:123456789012:type/hook/MyTestHook" \
  --region us-west-2
```

Desarrollo de Hooks personalizados mediante la CloudFormation CLI

Esta sección es para los clientes que desean desarrollar Hooks personalizados y registrarlos en el AWS CloudFormation Registro.

Hay tres pasos principales para desarrollar un Hook personalizado:

1. Iniciar

   Para desarrollar Hooks personalizados, debe configurar y usar la CloudFormation CLI. Para iniciar un proyecto de Hook y sus archivos necesarios, utilice el initcomando CloudFormation CLI y especifique que desea crear un Hook. Para obtener más información, consulte Iniciar un proyecto AWS CloudFormation Hooks personalizado.
2. Model

   Para modelar, crear y validar tu esquema de Hook, define el Hook, sus propiedades y sus atributos.

   La CloudFormation CLI crea funciones de controlador vacías que corresponden a un punto de invocación de Hook específico. Agrega tu propia lógica a estos controladores para controlar lo que ocurre durante la invocación de Hook en cada etapa del ciclo de vida objetivo. Para obtener más información, consulte Modelado de AWS CloudFormation ganchos personalizados.

3. Regístrese

   Para registrar un Hook, envía tu Hook para que se registre como una extensión privada o pública de terceros. Registra tu Hook con la `submit` operación. Para obtener más información, consulte Registrar un Hook personalizado con AWS CloudFormation.

   Las siguientes tareas están asociadas al registro de tu Hook:

   a. Publicar: los Hooks se publican en el registro.

   b. Configurar: los ganchos se configuran cuando la configuración de tipos se invoca contra las pilas.

   > **Note**
   >
   > Los ganchos se agotarán después de 30 segundos y se volverán a intentar hasta 3 veces. Para obtener más información, consulte Límites de tiempo de espera y reintentos.

Los siguientes temas lo guían a través del proceso de desarrollo, registro y publicación de Hooks personalizados con Python o Java.

Temas

- Requisitos previos para desarrollar Hooks personalizados AWS CloudFormation
- Iniciar un proyecto AWS CloudFormation Hooks personalizado
- Modelado de AWS CloudFormation ganchos personalizados
- Registrar un Hook personalizado con AWS CloudFormation
- Probando un Hook personalizado en tu Cuenta de AWS
- Actualización de un Hook personalizado
- Anular el registro de un Hook personalizado del registro CloudFormation
- Publicar ganchos para uso público
- Referencia de sintaxis del esquema para AWS CloudFormation Hooks

Requisitos previos para desarrollar Hooks personalizados AWS CloudFormation

Puedes desarrollar un Hook personalizado con Java o Python. Los siguientes son los requisitos previos para desarrollar Hooks personalizados:

Requisitos previos de Java

- Apache Maven
- JDK17

> **Note**
>
> Si pretende utilizar la interfaz de línea de CloudFormation comandos (CLI) para iniciar un proyecto de Hooks para Java, también debe instalar Python 3.8 o posterior. El complemento de Java para el se CloudFormation CLI puede instalar a través `pip` del administrador de paquetes de Python, que está distribuido con Python.

Para implementar los controladores de Hook en tu proyecto de Java Hooks, puedes descargar los archivos de ejemplo del controlador de Java Hook.

Requisitos previos de Python

- Python versión 3.8 o posterior.

Para implementar los controladores de Hook para tu proyecto de Python Hooks, puedes descargar los archivos de ejemplo del controlador de Python Hook.

Permisos para desarrollar Hooks

Además de los permisos CloudFormation `CreateUpdate`, y `Delete` stack, necesitarás acceder a las siguientes AWS CloudFormation operaciones. El acceso a estas operaciones se administra mediante la CloudFormation política de su IAM función.

- register-type
- list-types
- deregister-type

- set-type-configuration

Configura un entorno de desarrollo para Hooks

Para desarrollar Hooks, debes estar familiarizado con CloudFormation las plantillas y con Python o Java.

Para instalar CloudFormation CLI el y los complementos asociados:

1. Instala the CloudFormation CLI with`pip`, el administrador de paquetes de Python.

   ```
   pip3 install cloudformation-cli
   ```

2. Instale el complemento Python o Java para CloudFormation CLI.

   Python

   ```
   pip3 install cloudformation-cli-python-plugin
   ```

   Java

   ```
   pip3 install cloudformation-cli-java-plugin
   ```

Para actualizar el complemento CloudFormation CLI y el complemento, puede utilizar la opción de actualización.

Python

```
pip3 install --upgrade cloudformation-cli cloudformation-cli-python-plugin
```

Java

```
pip3 install --upgrade cloudformation-cli cloudformation-cli-java-plugin
```

Iniciar un proyecto AWS CloudFormation Hooks personalizado

El primer paso para crear tu proyecto Hooks personalizado es iniciar el proyecto. Puedes usar el CloudFormation CLI `init` comando para iniciar tu proyecto Hooks personalizado.

El `init` comando lanza un asistente que te guía a través de la configuración del proyecto, incluido un archivo de esquema de Hooks. Usa este archivo de esquema como punto de partida para definir la forma y la semántica de tus Hooks. Para obtener más información, consulte Sintaxis del esquema.

Para iniciar un proyecto de Hook:

1. Crea un directorio para el proyecto.

```
mkdir ~/mycompany-testing-mytesthook
```

2. Vaya al nuevo directorio.

```
cd ~/mycompany-testing-mytesthook
```

3. Utilice el CloudFormation CLI `init` comando para iniciar el proyecto.

```
cfn init
```

   El comando devuelve el resultado siguiente.

```
Initializing new project
```

4. El `init` comando lanza un asistente que le guiará a través de la configuración del proyecto. Cuando se le solicite, introduzca h para especificar un proyecto de Hooks.

```
Do you want to develop a new resource(r) a module(m) or a hook(h)?
```

```
h
```

5. Introduzca un nombre para el tipo de Hook.

```
What's the name of your hook type?
(Organization::Service::Hook)
```

```
MyCompany::Testing::MyTestHook
```

6. Si solo hay un complemento de idioma instalado, se selecciona de forma predeterminada. Si hay más de un complemento de idioma instalado, puede elegir el idioma que desee. Introduce una selección numérica para el idioma de tu elección.

```
Select a language for code generation:
[1] java
[2] python38
[3] python39
(enter an integer):
```

7. Configure el empaquetado en función del idioma de desarrollo elegido.

   Python

   (Opcional) Elija Docker para el empaquetado independiente de la plataforma. Si bien Docker no es obligatorio, es muy recomendable para facilitar el empaquetado.

```
Use docker for platform-independent packaging (Y/n)?
This is highly recommended unless you are experienced with cross-platform Python
 packaging.
```

   Java

   Establezca el nombre del paquete de Java y elija un modelo de generación de código. Puede usar el nombre de paquete predeterminado o crear uno nuevo.

```
Enter a package name (empty for default 'com.mycompany.testing.mytesthook'):
```

```
Choose codegen model - 1 (default) or 2 (guided-aws):
```

Resultados: Has iniciado correctamente el proyecto y has generado los archivos necesarios para desarrollar un Hook. El siguiente es un ejemplo de los directorios y archivos que componen un proyecto de Hooks para Python 3.8.

```
mycompany-testing-mytesthook.json
rpdk.log
```

```
README.md
requirements.txt
hook-role.yaml
template.yml
docs
    README.md
src
    __init__.py
    handlers.py
    models.py
    target_models
        aws_s3_bucket.py
```

> **Note**
>
> Los archivos del `src` directorio se crean en función de la selección de idioma. Hay algunos comentarios y ejemplos útiles en los archivos generados. Algunos archivos, por ejemplo`models.py`, se actualizan automáticamente en un paso posterior al ejecutar el `generate` comando para añadir código de tiempo de ejecución a los controladores.

Modelado de AWS CloudFormation ganchos personalizados

Modelar AWS CloudFormation Hooks personalizados implica crear un esquema que defina el Hook, sus propiedades y sus atributos. Al crear un proyecto Hook personalizado con el `cfn init` comando, se crea un ejemplo de esquema de Hook como un archivo JSON de texto con formato,. *`hook-name`*`.json`

Los puntos de invocación y las acciones objetivo especifican el punto exacto en el que se invoca el Hook. Los controladores de ganchos alojan una lógica personalizada ejecutable para estos puntos. Por ejemplo, una acción objetivo de la `CREATE` operación utiliza un `preCreate` controlador. El código escrito en el controlador se invocará cuando los objetivos y servicios de Hook realicen una acción coincidente. Los objetivos de los ganchos son el destino en el que se invocan los ganchos. Puede especificar objetivos como recursos AWS CloudFormation públicos, recursos privados o recursos personalizados. Los Hooks admiten un número ilimitado de objetivos de Hook.

El esquema contiene los permisos necesarios para el Hook. La creación del Hook requiere que especifiques los permisos para cada controlador de Hook. CloudFormation anima a los autores a redactar políticas que sigan el consejo de seguridad estándar de conceder el mínimo de privilegios o conceder solo los permisos necesarios para realizar una tarea. Determine lo que deben hacer los

usuarios (y las funciones) y, a continuación, elabore políticas que les permitan realizar únicamente esas tareas para las operaciones de Hook. CloudFormation usa estos permisos para analizar los permisos proporcionados por los usuarios de Hook. Estos permisos se transfieren a Hook. Los controladores de Hook utilizan estos permisos para acceder a AWS los recursos.

Puedes usar el siguiente archivo de esquema como punto de partida para definir tu Hook. Usa el esquema Hook para especificar qué controladores deseas implementar. Si decides no implementar un controlador específico, elimínalo de la sección de controladores del esquema de Hook. Para obtener más información sobre el esquema, consulte. Sintaxis del esquema

```
{
    "typeName":"MyCompany::Testing::MyTestHook",
    "description":"Verifies S3 bucket and SQS queues properties before create and
 update",
    "sourceUrl":"https://mycorp.com/my-repo.git",
    "documentationUrl":"https://mycorp.com/documentation",
    "typeConfiguration":{
        "properties":{
            "minBuckets":{
                "description":"Minimum number of compliant buckets",
                "type":"string"
            },
            "minQueues":{
                "description":"Minimum number of compliant queues",
                "type":"string"
            },
            "encryptionAlgorithm":{
                "description":"Encryption algorithm for SSE",
                "default":"AES256",
                "type":"string"
            }
        },
        "required":[

        ],
        "additionalProperties":false
    },
    "handlers":{
        "preCreate":{
            "targetNames":[
                "AWS::S3::Bucket",
                "AWS::SQS::Queue"
            ],
```

```
            "permissions":[

            ]
        },
        "preUpdate":{
            "targetNames":[
                "AWS::S3::Bucket",
                "AWS::SQS::Queue"
            ],
            "permissions":[

            ]
        },
        "preDelete":{
            "targetNames":[
                "AWS::S3::Bucket",
                "AWS::SQS::Queue"
            ],
            "permissions":[
                "s3:ListBucket",
                "s3:ListAllMyBuckets",
                "s3:GetEncryptionConfiguration",
                "sqs:ListQueues",
                "sqs:GetQueueAttributes",
                "sqs:GetQueueUrl"
            ]
        }
    },
    "additionalProperties":false
}
```

Temas

- Modelado de AWS CloudFormation Hooks personalizados con Java
- Modelado de AWS CloudFormation Hooks personalizados con Python

Modelado de AWS CloudFormation Hooks personalizados con Java

Modelar AWS CloudFormation Hooks personalizados implica crear un esquema que defina el Hook, sus propiedades y sus atributos. En este tutorial, se explica cómo modelar Hooks personalizados con Java.

Paso 1: Añadir las dependencias del proyecto

Los proyectos de Hooks basados en Java se basan en el `pom.xml` archivo de Maven como dependencia. Expanda la siguiente sección y copie el código fuente en el `pom.xml` archivo que se encuentra en la raíz del proyecto.

Enganche las dependencias del proyecto (pom.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<project
    xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/
maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.mycompany.testing.mytesthook</groupId>
    <artifactId>mycompany-testing-mytesthook-handler</artifactId>
    <name>mycompany-testing-mytesthook-handler</name>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>

    <properties>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
        <aws.java.sdk.version>2.16.1</aws.java.sdk.version>
        <checkstyle.version>8.36.2</checkstyle.version>
        <commons-io.version>2.8.0</commons-io.version>
        <jackson.version>2.11.3</jackson.version>
        <maven-checkstyle-plugin.version>3.1.1</maven-checkstyle-plugin.version>
        <mockito.version>3.6.0</mockito.version>
        <spotbugs.version>4.1.4</spotbugs.version>
        <spotless.version>2.5.0</spotless.version>
        <maven-javadoc-plugin.version>3.2.0</maven-javadoc-plugin.version>
        <maven-source-plugin.version>3.2.1</maven-source-plugin.version>
        <cfn.generate.args/>
    </properties>

    <dependencyManagement>
        <dependencies>
            <dependency>
```

```
                <groupId>software.amazon.awssdk</groupId>
                <artifactId>bom</artifactId>
                <version>2.16.1</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>

    <dependencies>
        <!-- https://mvnrepository.com/artifact/software.amazon.cloudformation/aws-
cloudformation-rpdk-java-plugin -->
        <dependency>
            <groupId>software.amazon.cloudformation</groupId>
            <artifactId>aws-cloudformation-rpdk-java-plugin</artifactId>
            <version>[2.0.0,3.0.0)</version>
        </dependency>

        <!-- AWS Java SDK v2 Dependencies -->
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>sdk-core</artifactId>
        </dependency>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>cloudformation</artifactId>
        </dependency>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>s3</artifactId>
        </dependency>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>utils</artifactId>
        </dependency>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>apache-client</artifactId>
        </dependency>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>sqs</artifactId>
        </dependency>
```

```
        <!-- Test dependency for Java Providers -->
        <dependency>
            <groupId>software.amazon.cloudformation</groupId>
            <artifactId>cloudformation-cli-java-plugin-testing-support</artifactId>
            <version>1.0.0</version>
        </dependency>

        <!-- https://mvnrepository.com/artifact/com.amazonaws/aws-java-sdk-s3 -->
        <dependency>
            <groupId>com.amazonaws</groupId>
            <artifactId>aws-java-sdk-s3</artifactId>
            <version>1.12.85</version>
        </dependency>

        <!-- https://mvnrepository.com/artifact/commons-io/commons-io -->
        <dependency>
            <groupId>commons-io</groupId>
            <artifactId>commons-io</artifactId>
            <version>${commons-io.version}</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.apache.commons/commons-lang3 -->
        <dependency>
            <groupId>org.apache.commons</groupId>
            <artifactId>commons-lang3</artifactId>
            <version>3.9</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.apache.commons/commons-collections4
 -->
        <dependency>
            <groupId>org.apache.commons</groupId>
            <artifactId>commons-collections4</artifactId>
            <version>4.4</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/com.google.guava/guava -->
        <dependency>
            <groupId>com.google.guava</groupId>
            <artifactId>guava</artifactId>
            <version>29.0-jre</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/com.amazonaws/aws-java-sdk-
cloudformation -->
        <dependency>
            <groupId>com.amazonaws</groupId>
            <artifactId>aws-java-sdk-cloudformation</artifactId>
```

```
            <version>1.11.555</version>
            <scope>test</scope>
        </dependency>

        <!-- https://mvnrepository.com/artifact/commons-codec/commons-codec -->
        <dependency>
            <groupId>commons-codec</groupId>
            <artifactId>commons-codec</artifactId>
            <version>1.14</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/software.amazon.cloudformation/aws-
cloudformation-resource-schema -->
        <dependency>
            <groupId>software.amazon.cloudformation</groupId>
            <artifactId>aws-cloudformation-resource-schema</artifactId>
            <version>[2.0.5, 3.0.0)</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.dataformat/
jackson-databind -->
        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-databind</artifactId>
            <version>${jackson.version}</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.dataformat/
jackson-dataformat-cbor -->
        <dependency>
            <groupId>com.fasterxml.jackson.dataformat</groupId>
            <artifactId>jackson-dataformat-cbor</artifactId>
            <version>${jackson.version}</version>
        </dependency>

        <dependency>
            <groupId>com.fasterxml.jackson.datatype</groupId>
            <artifactId>jackson-datatype-jsr310</artifactId>
            <version>${jackson.version}</version>
        </dependency>

        <!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.module/jackson-
modules-java8 -->
        <dependency>
            <groupId>com.fasterxml.jackson.module</groupId>
            <artifactId>jackson-modules-java8</artifactId>
            <version>${jackson.version}</version>
```

```
            <type>pom</type>
            <scope>runtime</scope>
        </dependency>

         <!-- https://mvnrepository.com/artifact/org.json/json -->
        <dependency>
            <groupId>org.json</groupId>
            <artifactId>json</artifactId>
            <version>20180813</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/com.amazonaws/aws-java-sdk-core -->
        <dependency>
            <groupId>com.amazonaws</groupId>
            <artifactId>aws-java-sdk-core</artifactId>
            <version>1.11.1034</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/com.amazonaws/aws-lambda-java-core -->
        <dependency>
            <groupId>com.amazonaws</groupId>
            <artifactId>aws-lambda-java-core</artifactId>
            <version>1.2.0</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/com.amazonaws/aws-lambda-java-log4j2 --
>
        <dependency>
            <groupId>com.amazonaws</groupId>
            <artifactId>aws-lambda-java-log4j2</artifactId>
            <version>1.2.0</version>
        </dependency>

        <!-- https://mvnrepository.com/artifact/com.google.code.gson/gson -->
        <dependency>
            <groupId>com.google.code.gson</groupId>
            <artifactId>gson</artifactId>
            <version>2.8.8</version>
        </dependency>


        <!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <version>1.18.4</version>
            <scope>provided</scope>
```

```
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.apache.logging.log4j/log4j-api -->
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-api</artifactId>
            <version>2.17.1</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.apache.logging.log4j/log4j-core --
>
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-core</artifactId>
            <version>2.17.1</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.apache.logging.log4j/log4j-slf4j-
impl -->
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-slf4j-impl</artifactId>
            <version>2.17.1</version>
        </dependency>

        <!-- https://mvnrepository.com/artifact/org.assertj/assertj-core -->
        <dependency>
            <groupId>org.assertj</groupId>
            <artifactId>assertj-core</artifactId>
            <version>3.12.2</version>
            <scope>test</scope>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter -->
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter</artifactId>
            <version>5.5.0-M1</version>
            <scope>test</scope>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.mockito/mockito-core -->
        <dependency>
            <groupId>org.mockito</groupId>
            <artifactId>mockito-core</artifactId>
            <version>3.6.0</version>
            <scope>test</scope>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.mockito/mockito-junit-jupiter -->
```

```
        <dependency>
            <groupId>org.mockito</groupId>
            <artifactId>mockito-junit-jupiter</artifactId>
            <version>3.6.0</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.8.1</version>
                <configuration>
                    <compilerArgs>
                        <arg>-Xlint:all,-options,-processing</arg>
                    </compilerArgs>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-shade-plugin</artifactId>
                <version>2.3</version>
                <configuration>
                    <createDependencyReducedPom>false</createDependencyReducedPom>
                    <filters>
                        <filter>
                            <artifact>*:*</artifact>
                            <excludes>
                                <exclude>**/Log4j2Plugins.dat</exclude>
                            </excludes>
                        </filter>
                    </filters>
                </configuration>
                <executions>
                    <execution>
                        <phase>package</phase>
                        <goals>
                            <goal>shade</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
```

```
            <plugin>
                <groupId>org.codehaus.mojo</groupId>
                <artifactId>exec-maven-plugin</artifactId>
                <version>1.6.0</version>
                <executions>
                    <execution>
                        <id>generate</id>
                        <phase>generate-sources</phase>
                        <goals>
                            <goal>exec</goal>
                        </goals>
                        <configuration>
                            <executable>cfn</executable>
                            <commandlineArgs>generate ${cfn.generate.args}</
commandlineArgs>
                            <workingDirectory>${project.basedir}</workingDirectory>
                        </configuration>
                    </execution>
                </executions>
            </plugin>
            <plugin>
                <groupId>org.codehaus.mojo</groupId>
                <artifactId>build-helper-maven-plugin</artifactId>
                <version>3.0.0</version>
                <executions>
                    <execution>
                        <id>add-source</id>
                        <phase>generate-sources</phase>
                        <goals>
                            <goal>add-source</goal>
                        </goals>
                        <configuration>
                            <sources>
                                <source>${project.basedir}/target/generated-sources/
rpdk</source>
                            </sources>
                        </configuration>
                    </execution>
                </executions>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-resources-plugin</artifactId>
                <version>2.4</version>
```

```
            </plugin>
            <plugin>
                <artifactId>maven-surefire-plugin</artifactId>
                <version>3.0.0-M3</version>
            </plugin>
            <plugin>
                <groupId>org.jacoco</groupId>
                <artifactId>jacoco-maven-plugin</artifactId>
                <version>0.8.4</version>
                <configuration>
                    <excludes>
                        <exclude>**/BaseHookConfiguration*</exclude>
                        <exclude>**/BaseHookHandler*</exclude>
                        <exclude>**/HookHandlerWrapper*</exclude>
                        <exclude>**/ResourceModel*</exclude>
                        <exclude>**/TypeConfigurationModel*</exclude>
                        <exclude>**/model/**/*</exclude>
                    </excludes>
                </configuration>
                <executions>
                    <execution>
                        <goals>
                            <goal>prepare-agent</goal>
                        </goals>
                    </execution>
                    <execution>
                        <id>report</id>
                        <phase>test</phase>
                        <goals>
                            <goal>report</goal>
                        </goals>
                    </execution>
                    <execution>
                        <id>jacoco-check</id>
                        <goals>
                            <goal>check</goal>
                        </goals>
                        <configuration>
                            <rules>
                                <rule>
                                    <element>PACKAGE</element>
                                    <limits>
                                        <limit>
                                            <counter>BRANCH</counter>
```

```
                                        <value>COVEREDRATIO</value>
                                        <minimum>0.8</minimum>
                                    </limit>
                                    <limit>
                                        <counter>INSTRUCTION</counter>
                                        <value>COVEREDRATIO</value>
                                        <minimum>0.8</minimum>
                                    </limit>
                                </limits>
                            </rule>
                        </rules>
                    </configuration>
                </execution>
            </executions>
        </plugin>
    </plugins>
    <resources>
        <resource>
            <directory>${project.basedir}</directory>
            <includes>
                <include>mycompany-testing-mytesthook.json</include>
            </includes>
        </resource>
        <resource>
            <directory>${project.basedir}/target/loaded-target-schemas</directory>
            <includes>
                <include>**/*.json</include>
            </includes>
        </resource>
    </resources>
  </build>
</project>
```

Paso 2: Genera el paquete del proyecto Hook

Genera tu paquete de proyectos Hook. CloudFormation CLICrea funciones de controlador vacías que corresponden a acciones específicas de Hook en el ciclo de vida objetivo, tal como se define en la especificación de Hook.

```
cfn generate
```

El comando devuelve el resultado siguiente.

```
Generated files for MyCompany::Testing::MyTestHook
```

> **Note**
>
> Asegúrese de que sus tiempos de ejecución de Lambda eviten el uso up-to-date de una versión obsoleta. Para obtener más información, consulte Actualización de los tiempos de ejecución de Lambda para tipos de recursos y Hooks.

Paso 3: Añadir controladores de Hook

Agrega tu propio código de tiempo de ejecución del controlador Hook a los controladores que decidas implementar. Por ejemplo, puedes añadir el siguiente código para el registro.

```
logger.log("Internal testing Hook triggered for target: " +
 request.getHookContext().getTargetName());
```

CloudFormation CLIGenera un objeto Java simple y antiguo (JavaPOJO). Los siguientes son ejemplos de resultados generados a partir de`AWS::S3::Bucket`.

Example WASS3 .java BucketTargetModel

```
package com.mycompany.testing.mytesthook.model.aws.s3.bucket;

import...


@Data
@NoArgsConstructor
@EqualsAndHashCode(callSuper = true)
@ToString(callSuper = true)
@JsonAutoDetect(fieldVisibility = Visibility.ANY, getterVisibility = Visibility.NONE,
 setterVisibility = Visibility.NONE)
public class AwsS3BucketTargetModel extends ResourceHookTargetModel<AwsS3Bucket> {

    @JsonIgnore
    private static final TypeReference<AwsS3Bucket> TARGET_REFERENCE =
        new TypeReference<AwsS3Bucket>() {};

    @JsonIgnore
```

```
    private static final TypeReference<AwsS3BucketTargetModel> MODEL_REFERENCE =
        new TypeReference<AwsS3BucketTargetModel>() {};

    @JsonIgnore
    public static final String TARGET_TYPE_NAME = "AWS::S3::Bucket";


    @JsonIgnore
    public TypeReference<AwsS3Bucket> getHookTargetTypeReference() {
        return TARGET_REFERENCE;
    }

    @JsonIgnore
    public TypeReference<AwsS3BucketTargetModel> getTargetModelTypeReference() {
        return MODEL_REFERENCE;
    }
}
```

Example AwsS3Bucket.java

```
package com.mycompany.testing.mytesthook.model.aws.s3.bucket;

import ...


@Data
@Builder
@AllArgsConstructor
@NoArgsConstructor
@EqualsAndHashCode(callSuper = true)
@ToString(callSuper = true)
@JsonAutoDetect(fieldVisibility = Visibility.ANY, getterVisibility = Visibility.NONE,
 setterVisibility = Visibility.NONE)
public class AwsS3Bucket extends ResourceHookTarget {
    @JsonIgnore
    public static final String TYPE_NAME = "AWS::S3::Bucket";

    @JsonIgnore
    public static final String IDENTIFIER_KEY_ID = "/properties/Id";

    @JsonProperty("InventoryConfigurations")
    private List<InventoryConfiguration> inventoryConfigurations;
```

```
    @JsonProperty("WebsiteConfiguration")
    private WebsiteConfiguration websiteConfiguration;

    @JsonProperty("DualStackDomainName")
    private String dualStackDomainName;

    @JsonProperty("AccessControl")
    private String accessControl;

    @JsonProperty("AnalyticsConfigurations")
    private List<AnalyticsConfiguration> analyticsConfigurations;

    @JsonProperty("AccelerateConfiguration")
    private AccelerateConfiguration accelerateConfiguration;

    @JsonProperty("PublicAccessBlockConfiguration")
    private PublicAccessBlockConfiguration publicAccessBlockConfiguration;

    @JsonProperty("BucketName")
    private String bucketName;

    @JsonProperty("RegionalDomainName")
    private String regionalDomainName;

    @JsonProperty("OwnershipControls")
    private OwnershipControls ownershipControls;

    @JsonProperty("ObjectLockConfiguration")
    private ObjectLockConfiguration objectLockConfiguration;

    @JsonProperty("ObjectLockEnabled")
    private Boolean objectLockEnabled;

    @JsonProperty("LoggingConfiguration")
    private LoggingConfiguration loggingConfiguration;

    @JsonProperty("ReplicationConfiguration")
    private ReplicationConfiguration replicationConfiguration;

    @JsonProperty("Tags")
    private List<Tag> tags;

    @JsonProperty("DomainName")
    private String domainName;
```

```
    @JsonProperty("BucketEncryption")
    private BucketEncryption bucketEncryption;

    @JsonProperty("WebsiteURL")
    private String websiteURL;

    @JsonProperty("NotificationConfiguration")
    private NotificationConfiguration notificationConfiguration;

    @JsonProperty("LifecycleConfiguration")
    private LifecycleConfiguration lifecycleConfiguration;

    @JsonProperty("VersioningConfiguration")
    private VersioningConfiguration versioningConfiguration;

    @JsonProperty("MetricsConfigurations")
    private List<MetricsConfiguration> metricsConfigurations;

    @JsonProperty("IntelligentTieringConfigurations")
    private List<IntelligentTieringConfiguration> intelligentTieringConfigurations;

    @JsonProperty("CorsConfiguration")
    private CorsConfiguration corsConfiguration;

    @JsonProperty("Id")
    private String id;

    @JsonProperty("Arn")
    private String arn;

    @JsonIgnore
    public JSONObject getPrimaryIdentifier() {
        final JSONObject identifier = new JSONObject();
        if (this.getId() != null) {
            identifier.put(IDENTIFIER_KEY_ID, this.getId());
        }

        // only return the identifier if it can be used, i.e. if all components are
present
        return identifier.length() == 1 ? identifier : null;
    }

    @JsonIgnore
```

```
    public List<JSONObject> getAdditionalIdentifiers() {
        final List<JSONObject> identifiers = new ArrayList<JSONObject>();
        // only return the identifiers if any can be used
        return identifiers.isEmpty() ? null : identifiers;
    }
}
```

Example BucketEncryption.java

```
package software.amazon.testing.mytesthook.model.aws.s3.bucket;

import ...


@Data
@Builder
@AllArgsConstructor
@NoArgsConstructor
@JsonAutoDetect(fieldVisibility = Visibility.ANY, getterVisibility = Visibility.NONE,
 setterVisibility = Visibility.NONE)
public class BucketEncryption {
    @JsonProperty("ServerSideEncryptionConfiguration")
    private List<ServerSideEncryptionRule> serverSideEncryptionConfiguration;

}
```

Example ServerSideEncryptionRule.java

```
package com.mycompany.testing.mytesthook.model.aws.s3.bucket;

import ...


@Data
@Builder
@AllArgsConstructor
@NoArgsConstructor
@JsonAutoDetect(fieldVisibility = Visibility.ANY, getterVisibility = Visibility.NONE,
 setterVisibility = Visibility.NONE)
public class ServerSideEncryptionRule {
    @JsonProperty("BucketKeyEnabled")
    private Boolean bucketKeyEnabled;
```

```
    @JsonProperty("ServerSideEncryptionByDefault")
    private ServerSideEncryptionByDefault serverSideEncryptionByDefault;

}
```

Example ServerSideEncryptionByDefault.java

```
package com.mycompany.testing.mytesthook.model.aws.s3.bucket;

import ...


@Data
@Builder
@AllArgsConstructor
@NoArgsConstructor
@JsonAutoDetect(fieldVisibility = Visibility.ANY, getterVisibility = Visibility.NONE,
 setterVisibility = Visibility.NONE)
public class ServerSideEncryptionByDefault {
    @JsonProperty("SSEAlgorithm")
    private String sSEAlgorithm;

    @JsonProperty("KMSMasterKeyID")
    private String kMSMasterKeyID;

}
```

Con lo POJOs generado, ahora puedes escribir los controladores que realmente implementan la funcionalidad del Hook. Para este ejemplo, implementa el punto de `preUpdate` invocación `preCreate` y para los controladores.

Paso 4: Implementa los controladores Hook

Temas

- Codificando el creador de API clientes
- Codificando el creador de API solicitudes
- Implementación del código auxiliar
- Implementación del controlador base
- Implementación del preCreate controlador
- Codificación del preCreate controlador

- Actualización de la preCreate prueba
- Implementación del preUpdate controlador
- Codificando el preUpdate controlador
- Actualización de la preUpdate prueba
- Implementación del preDelete controlador
- Codificación del preDelete controlador
- Actualización del preDelete controlador

Codificando el creador de API clientes

1. En su interiorIDE, abra el `ClientBuilder.java` archivo, que se encuentra en la `src/main/java/com/mycompany/testing/mytesthook` carpeta.
2. Sustituya todo el contenido del `ClientBuilder.java` archivo por el siguiente código.

   Example ClientBuilder.java

```
package com.awscommunity.kms.encryptionsettings;

import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.cloudformation.HookLambdaWrapper;

/**
 * Describes static HTTP clients (to consume less memory) for API calls that
 * this hook makes to a number of AWS services.
 */
public final class ClientBuilder {

    private ClientBuilder() {
    }

    /**
     * Create an HTTP client for Amazon EC2.
     *
     * @return Ec2Client An {@link Ec2Client} object.
     */
    public static Ec2Client getEc2Client() {
        return
 Ec2Client.builder().httpClient(HookLambdaWrapper.HTTP_CLIENT).build();
    }
```

```
}
```

Codificando el creador de API solicitudes

1. En suIDE, abra el `Translator.java` archivo, ubicado en la `src/main/java/com/mycompany/testing/mytesthook` carpeta.
2. Sustituya todo el contenido del `Translator.java` archivo por el siguiente código.

   Example Translator.java

```
package com.mycompany.testing.mytesthook;


import software.amazon.awssdk.services.s3.model.GetBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.sqs.model.ListQueuesRequest;
import software.amazon.cloudformation.proxy.hook.targetmodel.HookTargetModel;

/**
 * This class is a centralized placeholder for
 *  - api request construction
 *  - object translation to/from aws sdk
 */

public class Translator {

    static ListBucketsRequest translateToListBucketsRequest(final HookTargetModel
 targetModel) {
        return ListBucketsRequest.builder().build();
    }

    static ListQueuesRequest translateToListQueuesRequest(final String nextToken) {
        return ListQueuesRequest.builder().nextToken(nextToken).build();
    }

    static ListBucketsRequest createListBucketsRequest() {
        return ListBucketsRequest.builder().build();
    }

    static ListQueuesRequest createListQueuesRequest() {
        return createListQueuesRequest(null);
    }
```

```
    static ListQueuesRequest createListQueuesRequest(final String nextToken) {
        return ListQueuesRequest.builder().nextToken(nextToken).build();
    }

    static GetBucketEncryptionRequest createGetBucketEncryptionRequest(final String
 bucket) {
        return GetBucketEncryptionRequest.builder().bucket(bucket).build();
    }
}
```

Implementación del código auxiliar

1. En su IDE interior, abra el `AbstractTestBase.java` archivo, ubicado en la `src/main/java/com/mycompany/testing/mytesthook` carpeta.
2. Sustituya todo el contenido del `AbstractTestBase.java` archivo por el siguiente código.

   Example Translator.java

```
package com.mycompany.testing.mytesthook;

import com.google.common.collect.ImmutableMap;
import org.mockito.Mockito;
import software.amazon.awssdk.auth.credentials.AwsCredentialsProvider;
import software.amazon.awssdk.auth.credentials.AwsSessionCredentials;
import software.amazon.awssdk.auth.credentials.StaticCredentialsProvider;
import software.amazon.awssdk.awscore.AwsRequest;
import software.amazon.awssdk.awscore.AwsRequestOverrideConfiguration;
import software.amazon.awssdk.awscore.AwsResponse;
import software.amazon.awssdk.core.SdkClient;
import software.amazon.awssdk.core.pagination.sync.SdkIterable;
import software.amazon.cloudformation.proxy.AmazonWebServicesClientProxy;
import software.amazon.cloudformation.proxy.Credentials;
import software.amazon.cloudformation.proxy.LoggerProxy;
import software.amazon.cloudformation.proxy.OperationStatus;
import software.amazon.cloudformation.proxy.ProgressEvent;
import software.amazon.cloudformation.proxy.ProxyClient;
import software.amazon.cloudformation.proxy.hook.targetmodel.HookTargetModel;

import javax.annotation.Nonnull;
import java.time.Duration;
import java.util.concurrent.CompletableFuture;
```

```
import java.util.function.Function;
import java.util.function.Supplier;

import static org.assertj.core.api.Assertions.assertThat;

@lombok.Getter
public class AbstractTestBase {
    protected final AwsSessionCredentials awsSessionCredential;
    protected final AwsCredentialsProvider v2CredentialsProvider;
    protected final AwsRequestOverrideConfiguration configuration;
    protected final LoggerProxy loggerProxy;
    protected final Supplier<Long> awsLambdaRuntime = () ->
 Duration.ofMinutes(15).toMillis();
    protected final AmazonWebServicesClientProxy proxy;
    protected final Credentials mockCredentials =
        new Credentials("mockAccessId", "mockSecretKey", "mockSessionToken");

    @lombok.Setter
    private SdkClient serviceClient;

    protected AbstractTestBase() {
        loggerProxy = Mockito.mock(LoggerProxy.class);
        awsSessionCredential =
 AwsSessionCredentials.create(mockCredentials.getAccessKeyId(),
            mockCredentials.getSecretAccessKey(),
 mockCredentials.getSessionToken());
        v2CredentialsProvider =
 StaticCredentialsProvider.create(awsSessionCredential);
        configuration = AwsRequestOverrideConfiguration.builder()
            .credentialsProvider(v2CredentialsProvider)
            .build();
        proxy = new AmazonWebServicesClientProxy(
            loggerProxy,
            mockCredentials,
            awsLambdaRuntime
        ) {
            @Override
            public <ClientT> ProxyClient<ClientT> newProxy(@Nonnull
 Supplier<ClientT> client) {
                return new ProxyClient<ClientT>() {
                    @Override
                    public <RequestT extends AwsRequest, ResponseT extends
 AwsResponse>
                        ResponseT injectCredentialsAndInvokeV2(RequestT request,
```

```
                                                                    Function<RequestT,
ResponseT> requestFunction) {
                        return proxy.injectCredentialsAndInvokeV2(request,
requestFunction);
                    }

                    @Override
                    public <RequestT extends AwsRequest, ResponseT extends
AwsResponse> CompletableFuture<ResponseT>
                        injectCredentialsAndInvokeV2Async(RequestT request,
Function<RequestT, CompletableFuture<ResponseT>> requestFunction) {
                        return proxy.injectCredentialsAndInvokeV2Async(request,
requestFunction);
                    }

                    @Override
                    public <RequestT extends AwsRequest, ResponseT extends
AwsResponse, IterableT extends SdkIterable<ResponseT>>
                        IterableT
                        injectCredentialsAndInvokeIterableV2(RequestT request,
Function<RequestT, IterableT> requestFunction) {
                        return proxy.injectCredentialsAndInvokeIterableV2(request,
requestFunction);
                    }

                    @SuppressWarnings("unchecked")
                    @Override
                    public ClientT client() {
                        return (ClientT) serviceClient;
                    }
                };
            }
        };
    }

    protected void assertResponse(final ProgressEvent<HookTargetModel,
CallbackContext> response, final OperationStatus expectedStatus, final String
expectedMsg) {
        assertThat(response).isNotNull();
        assertThat(response.getStatus()).isEqualTo(expectedStatus);
        assertThat(response.getCallbackContext()).isNull();
        assertThat(response.getCallbackDelaySeconds()).isEqualTo(0);
        assertThat(response.getMessage()).isNotNull();
        assertThat(response.getMessage()).isEqualTo(expectedMsg);
```

```
    }

    protected HookTargetModel createHookTargetModel(final Object
 resourceProperties) {
        return HookTargetModel.of(ImmutableMap.of("ResourceProperties",
 resourceProperties));
    }

    protected HookTargetModel createHookTargetModel(final Object
 resourceProperties, final Object previousResourceProperties) {
        return HookTargetModel.of(
                ImmutableMap.of(
                    "ResourceProperties", resourceProperties,
                    "PreviousResourceProperties", previousResourceProperties
                )
        );
    }
}
```

Implementación del controlador base

1. En su interiorIDE, abra el `BaseHookHandlerStd.java` archivo, que se encuentra en la `src/main/java/com/mycompany/testing/mytesthook` carpeta.
2. Sustituya todo el contenido del `BaseHookHandlerStd.java` archivo por el siguiente código.

   Example Translator.java

```
package com.mycompany.testing.mytesthook;

import com.mycompany.testing.mytesthook.model.aws.s3.bucket.AwsS3Bucket;
import com.mycompany.testing.mytesthook.model.aws.sqs.queue.AwsSqsQueue;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.cloudformation.exceptions.UnsupportedTargetException;
import software.amazon.cloudformation.proxy.AmazonWebServicesClientProxy;
import software.amazon.cloudformation.proxy.Logger;
import software.amazon.cloudformation.proxy.ProgressEvent;
import software.amazon.cloudformation.proxy.ProxyClient;
import software.amazon.cloudformation.proxy.hook.HookHandlerRequest;
import software.amazon.cloudformation.proxy.hook.targetmodel.HookTargetModel;
```

```
public abstract class BaseHookHandlerStd extends BaseHookHandler<CallbackContext,
 TypeConfigurationModel> {
    public static final String HOOK_TYPE_NAME = "MyCompany::Testing::MyTestHook";

    protected Logger logger;

    @Override
    public ProgressEvent<HookTargetModel, CallbackContext> handleRequest(
            final AmazonWebServicesClientProxy proxy,
            final HookHandlerRequest request,
            final CallbackContext callbackContext,
            final Logger logger,
            final TypeConfigurationModel typeConfiguration
    ) {
        this.logger = logger;

        final String targetName = request.getHookContext().getTargetName();

        final ProgressEvent<HookTargetModel, CallbackContext> result;
        if (AwsS3Bucket.TYPE_NAME.equals(targetName)) {
            result = handleS3BucketRequest(
                    proxy,
                    request,
                    callbackContext != null ? callbackContext : new
 CallbackContext(),
                    proxy.newProxy(ClientBuilder::createS3Client),
                    typeConfiguration
            );
        } else if (AwsSqsQueue.TYPE_NAME.equals(targetName)) {
            result = handleSqsQueueRequest(
                    proxy,
                    request,
                    callbackContext != null ? callbackContext : new
 CallbackContext(),
                    proxy.newProxy(ClientBuilder::createSqsClient),
                    typeConfiguration
            );
        } else {
            throw new UnsupportedTargetException(targetName);
        }

        log(
            String.format(
```

```
                "Result for [%s] invocation for target [%s] returned status [%s]
 with message [%s]",
                request.getHookContext().getInvocationPoint(),
                targetName,
                result.getStatus(),
                result.getMessage()
            )
        );

        return result;
    }

    protected abstract ProgressEvent<HookTargetModel, CallbackContext>
 handleS3BucketRequest(
            final AmazonWebServicesClientProxy proxy,
            final HookHandlerRequest request,
            final CallbackContext callbackContext,
            final ProxyClient<S3Client> proxyClient,
            final TypeConfigurationModel typeConfiguration
    );

    protected abstract ProgressEvent<HookTargetModel, CallbackContext>
 handleSqsQueueRequest(
            final AmazonWebServicesClientProxy proxy,
            final HookHandlerRequest request,
            final CallbackContext callbackContext,
            final ProxyClient<SqsClient> proxyClient,
            final TypeConfigurationModel typeConfiguration
    );

    protected void log(final String message) {
        if (logger != null) {
            logger.log(message);
        } else {
            System.out.println(message);
        }
    }
}
```

Implementación del **preCreate** controlador

El `preCreate` controlador verifica la configuración de cifrado del lado del servidor para un recurso or. `AWS::S3::Bucket` `AWS::SQS::Queue`

- En el caso `AWS::S3::Bucket` de un recurso, el Hook solo pasará si se cumple lo siguiente:
  - Se ha establecido el cifrado del bucket de Amazon S3.
  - La clave de bucket de Amazon S3 está habilitada para el bucket.
  - El algoritmo de cifrado establecido para el bucket de Amazon S3 es el algoritmo correcto que se requiere.
  - El ID de AWS Key Management Service clave está establecido.
- En el caso `AWS::SQS::Queue` de un recurso, el Hook solo pasará si se cumple lo siguiente:
  - El ID de la AWS Key Management Service clave está establecido.

Codificación del **preCreate** controlador

1. En suIDE, abra el `PreCreateHookHandler.java` archivo, ubicado en la `src/main/java/software/mycompany/testing/mytesthook` carpeta.
2. Sustituya todo el contenido del `PreCreateHookHandler.java` archivo por el siguiente código.

```
package com.mycompany.testing.mytesthook;

import com.mycompany.testing.mytesthook.model.aws.s3.bucket.AwsS3Bucket;
import com.mycompany.testing.mytesthook.model.aws.s3.bucket.AwsS3BucketTargetModel;
import com.mycompany.testing.mytesthook.model.aws.s3.bucket.BucketEncryption;
import
 com.mycompany.testing.mytesthook.model.aws.s3.bucket.ServerSideEncryptionByDefault;
import
 com.mycompany.testing.mytesthook.model.aws.s3.bucket.ServerSideEncryptionRule;
import com.mycompany.testing.mytesthook.model.aws.sqs.queue.AwsSqsQueue;
import com.mycompany.testing.mytesthook.model.aws.sqs.queue.AwsSqsQueueTargetModel;
import org.apache.commons.collections.CollectionUtils;
import org.apache.commons.lang3.StringUtils;
import software.amazon.cloudformation.exceptions.UnsupportedTargetException;
import software.amazon.cloudformation.proxy.HandlerErrorCode;
import software.amazon.cloudformation.proxy.Logger;
import software.amazon.cloudformation.proxy.AmazonWebServicesClientProxy;
import software.amazon.cloudformation.proxy.hook.HookStatus;
import software.amazon.cloudformation.proxy.hook.HookProgressEvent;
```

```
import software.amazon.cloudformation.proxy.hook.HookHandlerRequest;
import
 software.amazon.cloudformation.proxy.hook.targetmodel.ResourceHookTargetModel;

import java.util.List;

public class PreCreateHookHandler extends BaseHookHandler<TypeConfigurationModel,
 CallbackContext> {

    @Override
    public HookProgressEvent<CallbackContext> handleRequest(
        final AmazonWebServicesClientProxy proxy,
        final HookHandlerRequest request,
        final CallbackContext callbackContext,
        final Logger logger,
        final TypeConfigurationModel typeConfiguration) {

        final String targetName = request.getHookContext().getTargetName();
        if ("AWS::S3::Bucket".equals(targetName)) {
            final ResourceHookTargetModel<AwsS3Bucket> targetModel =
 request.getHookContext().getTargetModel(AwsS3BucketTargetModel.class);

            final AwsS3Bucket bucket = targetModel.getResourceProperties();
            final String encryptionAlgorithm =
 typeConfiguration.getEncryptionAlgorithm();

            return validateS3BucketEncryption(bucket, encryptionAlgorithm);

        } else if ("AWS::SQS::Queue".equals(targetName)) {
            final ResourceHookTargetModel<AwsSqsQueue> targetModel =
 request.getHookContext().getTargetModel(AwsSqsQueueTargetModel.class);

            final AwsSqsQueue queue = targetModel.getResourceProperties();
            return validateSQSQueueEncryption(queue);
        } else {
            throw new UnsupportedTargetException(targetName);
        }
    }

    private HookProgressEvent<CallbackContext> validateS3BucketEncryption(final
 AwsS3Bucket bucket, final String requiredEncryptionAlgorithm) {
        HookStatus resultStatus = null;
        String resultMessage = null;
```

```
        if (bucket != null) {
            final BucketEncryption bucketEncryption = bucket.getBucketEncryption();
            if (bucketEncryption != null) {
                final List<ServerSideEncryptionRule> serverSideEncryptionRules =
bucketEncryption.getServerSideEncryptionConfiguration();
                if (CollectionUtils.isNotEmpty(serverSideEncryptionRules)) {
                    for (final ServerSideEncryptionRule rule :
serverSideEncryptionRules) {
                        final Boolean bucketKeyEnabled =
rule.getBucketKeyEnabled();
                        if (bucketKeyEnabled) {
                            final ServerSideEncryptionByDefault
serverSideEncryptionByDefault = rule.getServerSideEncryptionByDefault();

                            final String encryptionAlgorithm =
serverSideEncryptionByDefault.getSSEAlgorithm();
                            final String kmsKeyId =
serverSideEncryptionByDefault.getKMSMasterKeyID(); // "KMSMasterKeyID" is name of
the property for an AWS::S3::Bucket;

                            if (!StringUtils.equals(encryptionAlgorithm,
requiredEncryptionAlgorithm) && StringUtils.isBlank(kmsKeyId)) {
                                resultStatus = HookStatus.FAILED;
                                resultMessage = "KMS Key ID not set
and SSE Encryption Algorithm is incorrect for bucket with name: " +
bucket.getBucketName();
                            } else if (!StringUtils.equals(encryptionAlgorithm,
requiredEncryptionAlgorithm)) {
                                resultStatus = HookStatus.FAILED;
                                resultMessage = "SSE Encryption Algorithm is
incorrect for bucket with name: " + bucket.getBucketName();
                            } else if (StringUtils.isBlank(kmsKeyId)) {
                                resultStatus = HookStatus.FAILED;
                                resultMessage = "KMS Key ID not set for bucket with
name: " + bucket.getBucketName();
                            } else {
                                resultStatus = HookStatus.SUCCESS;
                                resultMessage = "Successfully invoked
PreCreateHookHandler for target: AWS::S3::Bucket";
                            }
                        } else {
                            resultStatus = HookStatus.FAILED;
                            resultMessage = "Bucket key not enabled for bucket with
name: " + bucket.getBucketName();
```

```
                        }

                        if (resultStatus == HookStatus.FAILED) {
                            break;
                        }
                    }
                } else {
                    resultStatus = HookStatus.FAILED;
                    resultMessage = "No SSE Encryption configurations for bucket
with name: " + bucket.getBucketName();
                }
            } else {
                resultStatus = HookStatus.FAILED;
                resultMessage = "Bucket Encryption not enabled for bucket with
name: " + bucket.getBucketName();
            }
        } else {
            resultStatus = HookStatus.FAILED;
            resultMessage = "Resource properties for S3 Bucket target model are
empty";
        }

        return HookProgressEvent.<CallbackContext>builder()
                .status(resultStatus)
                .message(resultMessage)
                .errorCode(resultStatus == HookStatus.FAILED ?
HandlerErrorCode.ResourceConflict : null)
                .build();
    }

    private HookProgressEvent<CallbackContext> validateSQSQueueEncryption(final
AwsSqsQueue queue) {
        if (queue == null) {
            return HookProgressEvent.<CallbackContext>builder()
                    .status(HookStatus.FAILED)
                    .message("Resource properties for SQS Queue target model are
empty")
                    .errorCode(HandlerErrorCode.ResourceConflict)
                    .build();
        }

        final String kmsKeyId = queue.getKmsMasterKeyId(); // "KmsMasterKeyId" is
name of the property for an AWS::SQS::Queue
        if (StringUtils.isBlank(kmsKeyId)) {
```

```
            return HookProgressEvent.<CallbackContext>builder()
                    .status(HookStatus.FAILED)
                    .message("Server side encryption turned off for queue with
 name: " + queue.getQueueName())
                    .errorCode(HandlerErrorCode.ResourceConflict)
                    .build();
        }

        return HookProgressEvent.<CallbackContext>builder()
                    .status(HookStatus.SUCCESS)
                    .message("Successfully invoked PreCreateHookHandler for target:
 AWS::SQS::Queue")
                    .build();
    }
}
```

Actualización de la **`preCreate`** prueba

1. En su carpetaIDE, abra el `PreCreateHandlerTest.java` archivo que se encuentra en la `src/test/java/software/mycompany/testing/mytesthook` carpeta.
2. Sustituya todo el contenido del `PreCreateHandlerTest.java` archivo por el siguiente código.

```
package com.mycompany.testing.mytesthook;

import com.google.common.collect.ImmutableMap;
import com.mycompany.testing.mytesthook.model.aws.s3.bucket.AwsS3Bucket;
import com.mycompany.testing.mytesthook.model.aws.s3.bucket.BucketEncryption;
import
 com.mycompany.testing.mytesthook.model.aws.s3.bucket.ServerSideEncryptionByDefault;
import
 com.mycompany.testing.mytesthook.model.aws.s3.bucket.ServerSideEncryptionRule;
import com.mycompany.testing.mytesthook.model.aws.sqs.queue.AwsSqsQueue;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import org.mockito.Mock;
import org.mockito.junit.jupiter.MockitoExtension;
import software.amazon.cloudformation.exceptions.UnsupportedTargetException;
import software.amazon.cloudformation.proxy.AmazonWebServicesClientProxy;
import software.amazon.cloudformation.proxy.HandlerErrorCode;
import software.amazon.cloudformation.proxy.Logger;
import software.amazon.cloudformation.proxy.hook.HookContext;
```

```
import software.amazon.cloudformation.proxy.hook.HookHandlerRequest;
import software.amazon.cloudformation.proxy.hook.HookProgressEvent;
import software.amazon.cloudformation.proxy.hook.HookStatus;
import software.amazon.cloudformation.proxy.hook.targetmodel.HookTargetModel;

import java.util.Collections;
import java.util.Map;

import static org.assertj.core.api.Assertions.assertThat;
import static org.assertj.core.api.Assertions.assertThatExceptionOfType;
import static org.mockito.Mockito.mock;

@ExtendWith(MockitoExtension.class)
public class PreCreateHookHandlerTest {

    @Mock
    private AmazonWebServicesClientProxy proxy;

    @Mock
    private Logger logger;

    @BeforeEach
    public void setup() {
        proxy = mock(AmazonWebServicesClientProxy.class);
        logger = mock(Logger.class);
    }

    @Test
    public void handleRequest_awsSqsQueueSuccess() {
        final PreCreateHookHandler handler = new PreCreateHookHandler();

        final AwsSqsQueue queue = buildSqsQueue("MyQueue", "KmsKey");
        final HookTargetModel targetModel = createHookTargetModel(queue);
        final TypeConfigurationModel typeConfiguration =
 TypeConfigurationModel.builder().encryptionAlgorithm("AES256").build();

        final HookHandlerRequest request = HookHandlerRequest.builder()

 .hookContext(HookContext.builder().targetName("AWS::SQS::Queue").targetModel(targetModel).
            .build();

        final HookProgressEvent<CallbackContext> response =
 handler.handleRequest(proxy, request, null, logger, typeConfiguration);
```

```
        assertResponse(response, HookStatus.SUCCESS, "Successfully invoked
PreCreateHookHandler for target: AWS::SQS::Queue");
    }

    @Test
    public void handleRequest_awsS3BucketSuccess() {
        final PreCreateHookHandler handler = new PreCreateHookHandler();

        final AwsS3Bucket bucket = buildAwsS3Bucket("amzn-s3-demo-bucket", true,
"AES256", "KmsKey");
        final HookTargetModel targetModel = createHookTargetModel(bucket);
        final TypeConfigurationModel typeConfiguration =
TypeConfigurationModel.builder().encryptionAlgorithm("AES256").build();

        final HookHandlerRequest request = HookHandlerRequest.builder()

.hookContext(HookContext.builder().targetName("AWS::S3::Bucket").targetModel(targetModel).
            .build();

        final HookProgressEvent<CallbackContext> response =
handler.handleRequest(proxy, request, null, logger, typeConfiguration);
        assertResponse(response, HookStatus.SUCCESS, "Successfully invoked
PreCreateHookHandler for target: AWS::S3::Bucket");
    }

    @Test
    public void handleRequest_awsS3BucketFail_bucketKeyNotEnabled() {
        final PreCreateHookHandler handler = new PreCreateHookHandler();

        final AwsS3Bucket bucket = buildAwsS3Bucket("amzn-s3-demo-bucket", false,
"AES256", "KmsKey");
        final HookTargetModel targetModel = createHookTargetModel(bucket);
        final TypeConfigurationModel typeConfiguration =
TypeConfigurationModel.builder().encryptionAlgorithm("AES256").build();

        final HookHandlerRequest request = HookHandlerRequest.builder()

.hookContext(HookContext.builder().targetName("AWS::S3::Bucket").targetModel(targetModel).
            .build();

        final HookProgressEvent<CallbackContext> response =
handler.handleRequest(proxy, request, null, logger, typeConfiguration);
        assertResponse(response, HookStatus.FAILED, "Bucket key not enabled for
bucket with name: amzn-s3-demo-bucket");
```

```
   }

   @Test
   public void handleRequest_awsS3BucketFail_incorrectSSEEncryptionAlgorithm() {
       final PreCreateHookHandler handler = new PreCreateHookHandler();

       final AwsS3Bucket bucket = buildAwsS3Bucket("amzn-s3-demo-bucket", true,
"SHA512", "KmsKey");
       final HookTargetModel targetModel = createHookTargetModel(bucket);
      final TypeConfigurationModel typeConfiguration =
TypeConfigurationModel.builder().encryptionAlgorithm("AES256").build();

       final HookHandlerRequest request = HookHandlerRequest.builder()

.hookContext(HookContext.builder().targetName("AWS::S3::Bucket").targetModel(targetModel).
           .build();

       final HookProgressEvent<CallbackContext> response =
handler.handleRequest(proxy, request, null, logger, typeConfiguration);
       assertResponse(response, HookStatus.FAILED, "SSE Encryption Algorithm is
incorrect for bucket with name: amzn-s3-demo-bucket");
   }

   @Test
   public void handleRequest_awsS3BucketFail_kmsKeyIdNotSet() {
       final PreCreateHookHandler handler = new PreCreateHookHandler();

       final AwsS3Bucket bucket = buildAwsS3Bucket("amzn-s3-demo-bucket", true,
"AES256", null);
       final HookTargetModel targetModel = createHookTargetModel(bucket);
       final TypeConfigurationModel typeConfiguration =
TypeConfigurationModel.builder().encryptionAlgorithm("AES256").build();

       final HookHandlerRequest request = HookHandlerRequest.builder()

.hookContext(HookContext.builder().targetName("AWS::S3::Bucket").targetModel(targetModel).
           .build();

       final HookProgressEvent<CallbackContext> response =
handler.handleRequest(proxy, request, null, logger, typeConfiguration);
       assertResponse(response, HookStatus.FAILED, "KMS Key ID not set for bucket
with name: amzn-s3-demo-bucket");
   }
```

```
   @Test
   public void handleRequest_awsSqsQueueFail_serverSideEncryptionOff() {
       final PreCreateHookHandler handler = new PreCreateHookHandler();

       final AwsSqsQueue queue = buildSqsQueue("MyQueue", null);
       final HookTargetModel targetModel = createHookTargetModel(queue);
       final TypeConfigurationModel typeConfiguration =
TypeConfigurationModel.builder().encryptionAlgorithm("AES256").build();

       final HookHandlerRequest request = HookHandlerRequest.builder()

.hookContext(HookContext.builder().targetName("AWS::SQS::Queue").targetModel(targetModel).
           .build();

       final HookProgressEvent<CallbackContext> response =
handler.handleRequest(proxy, request, null, logger, typeConfiguration);
       assertResponse(response, HookStatus.FAILED, "Server side encryption turned
off for queue with name: MyQueue");
   }

   @Test
   public void handleRequest_unsupportedTarget() {
       final PreCreateHookHandler handler = new PreCreateHookHandler();

       final Map<String, Object> unsupportedTarget =
ImmutableMap.of("ResourceName", "MyUnsupportedTarget");
       final HookTargetModel targetModel =
createHookTargetModel(unsupportedTarget);
       final TypeConfigurationModel typeConfiguration =
TypeConfigurationModel.builder().encryptionAlgorithm("AES256").build();

       final HookHandlerRequest request = HookHandlerRequest.builder()

.hookContext(HookContext.builder().targetName("AWS::Unsupported::Target").targetModel(targ
           .build();

       assertThatExceptionOfType(UnsupportedTargetException.class)
               .isThrownBy(() -> handler.handleRequest(proxy, request, null,
logger, typeConfiguration))
               .withMessageContaining("Unsupported target")
               .withMessageContaining("AWS::Unsupported::Target")
               .satisfies(e ->
assertThat(e.getErrorCode()).isEqualTo(HandlerErrorCode.InvalidRequest));
   }
```

```
   private void assertResponse(final HookProgressEvent<CallbackContext> response,
final HookStatus expectedStatus, final String expectedErrorMsg) {
       assertThat(response).isNotNull();
       assertThat(response.getStatus()).isEqualTo(expectedStatus);
       assertThat(response.getCallbackContext()).isNull();
       assertThat(response.getCallbackDelaySeconds()).isEqualTo(0);
       assertThat(response.getMessage()).isNotNull();
       assertThat(response.getMessage()).isEqualTo(expectedErrorMsg);
   }

   private HookTargetModel createHookTargetModel(final Object resourceProperties)
{
       return HookTargetModel.of(ImmutableMap.of("ResourceProperties",
resourceProperties));
   }

   @SuppressWarnings("SameParameterValue")
   private AwsSqsQueue buildSqsQueue(final String queueName, final String
kmsKeyId) {
       return AwsSqsQueue.builder()
               .queueName(queueName)
               .kmsMasterKeyId(kmsKeyId) // "KmsMasterKeyId" is name of the
property for an AWS::SQS::Queue
               .build();
   }

   @SuppressWarnings("SameParameterValue")
   private AwsS3Bucket buildAwsS3Bucket(
           final String bucketName,
           final Boolean bucketKeyEnabled,
           final String sseAlgorithm,
           final String kmsKeyId
   ) {
       return AwsS3Bucket.builder()
               .bucketName(bucketName)
               .bucketEncryption(
                   BucketEncryption.builder()
                       .serverSideEncryptionConfiguration(
                           Collections.singletonList(
                               ServerSideEncryptionRule.builder()
                                   .bucketKeyEnabled(bucketKeyEnabled)
                                   .serverSideEncryptionByDefault(
                                       ServerSideEncryptionByDefault.builder()
```

```
                                                .sSEAlgorithm(sseAlgorithm)
                                                .kMSMasterKeyID(kmsKeyId) //
 "KMSMasterKeyID" is name of the property for an AWS::S3::Bucket
                                                .build()
                                        ).build()
                                )
                            ).build()
                ).build();
    }
}
```

Implementación del **`preUpdate`** controlador

Implemente un `preUpdate` controlador, que se inicie antes de las operaciones de actualización para todos los objetivos especificados en el controlador. El `preUpdate` controlador logra lo siguiente:

- En el caso `AWS::S3::Bucket` de un recurso, el Hook solo pasará si se cumple lo siguiente:
  - El algoritmo de cifrado de buckets de un bucket de Amazon S3 no se ha modificado.

Codificando el **`preUpdate`** controlador

1. En suIDE, abra el `PreUpdateHookHandler.java` archivo, ubicado en la `src/main/java/software/mycompany/testing/mytesthook` carpeta.
2. Sustituya todo el contenido del `PreUpdateHookHandler.java` archivo por el siguiente código.

```
package com.mycompany.testing.mytesthook;

import com.mycompany.testing.mytesthook.model.aws.s3.bucket.AwsS3Bucket;
import com.mycompany.testing.mytesthook.model.aws.s3.bucket.AwsS3BucketTargetModel;
import
 com.mycompany.testing.mytesthook.model.aws.s3.bucket.ServerSideEncryptionRule;
import org.apache.commons.lang3.StringUtils;
import software.amazon.cloudformation.exceptions.UnsupportedTargetException;
import software.amazon.cloudformation.proxy.HandlerErrorCode;
import software.amazon.cloudformation.proxy.Logger;
import software.amazon.cloudformation.proxy.AmazonWebServicesClientProxy;
import software.amazon.cloudformation.proxy.hook.HookStatus;
import software.amazon.cloudformation.proxy.hook.HookProgressEvent;
import software.amazon.cloudformation.proxy.hook.HookHandlerRequest;
```

```
import
 software.amazon.cloudformation.proxy.hook.targetmodel.ResourceHookTargetModel;

import java.util.List;

public class PreUpdateHookHandler extends BaseHookHandler<TypeConfigurationModel,
 CallbackContext> {

    @Override
    public HookProgressEvent<CallbackContext> handleRequest(
        final AmazonWebServicesClientProxy proxy,
        final HookHandlerRequest request,
        final CallbackContext callbackContext,
        final Logger logger,
        final TypeConfigurationModel typeConfiguration) {

        final String targetName = request.getHookContext().getTargetName();
        if ("AWS::S3::Bucket".equals(targetName)) {
            final ResourceHookTargetModel<AwsS3Bucket> targetModel =
 request.getHookContext().getTargetModel(AwsS3BucketTargetModel.class);

            final AwsS3Bucket bucketProperties =
 targetModel.getResourceProperties();
            final AwsS3Bucket previousBucketProperties =
 targetModel.getPreviousResourceProperties();

            return validateBucketEncryptionRulesNotUpdated(bucketProperties,
 previousBucketProperties);
        } else {
            throw new UnsupportedTargetException(targetName);
        }
    }

    private HookProgressEvent<CallbackContext>
 validateBucketEncryptionRulesNotUpdated(final AwsS3Bucket resourceProperties,
 final AwsS3Bucket previousResourceProperties) {
        final List<ServerSideEncryptionRule> bucketEncryptionConfigs =
 resourceProperties.getBucketEncryption().getServerSideEncryptionConfiguration();
        final List<ServerSideEncryptionRule> previousBucketEncryptionConfigs =
 previousResourceProperties.getBucketEncryption().getServerSideEncryptionConfiguration();

        if (bucketEncryptionConfigs.size() !=
 previousBucketEncryptionConfigs.size()) {
            return HookProgressEvent.<CallbackContext>builder()
```

```
                    .status(HookStatus.FAILED)
                    .errorCode(HandlerErrorCode.NotUpdatable)
                    .message(
                        String.format(
                            "Current number of bucket encryption configs does not
 match previous. Current has %d configs while previously there were %d configs",
                            bucketEncryptionConfigs.size(),
                            previousBucketEncryptionConfigs.size()
                        )
                    ).build();
        }

        for (int i = 0; i < bucketEncryptionConfigs.size(); ++i) {
            final String currentEncryptionAlgorithm =
 bucketEncryptionConfigs.get(i).getServerSideEncryptionByDefault().getSSEAlgorithm();
            final String previousEncryptionAlgorithm =
 previousBucketEncryptionConfigs.get(i).getServerSideEncryptionByDefault().getSSEAlgorithm(

            if (!StringUtils.equals(currentEncryptionAlgorithm,
 previousEncryptionAlgorithm)) {
                return HookProgressEvent.<CallbackContext>builder()
                    .status(HookStatus.FAILED)
                    .errorCode(HandlerErrorCode.NotUpdatable)
                    .message(
                        String.format(
                            "Bucket Encryption algorithm can not be changed once
 set. The encryption algorithm was changed to '%s' from '%s'.",
                            currentEncryptionAlgorithm,
                            previousEncryptionAlgorithm
                        )
                    )
                    .build();
            }
        }

        return HookProgressEvent.<CallbackContext>builder()
                    .status(HookStatus.SUCCESS)
                    .message("Successfully invoked PreUpdateHookHandler for target:
 AWS::SQS::Queue")
                    .build();
    }
}
```

Actualización de la **`preUpdate`** prueba

1. En su carpetaIDE, abra el `PreUpdateHandlerTest.java` archivo de la `src/main/java/com/mycompany/testing/mytesthook` carpeta.
2. Sustituya todo el contenido del `PreUpdateHandlerTest.java` archivo por el siguiente código.

```
package com.mycompany.testing.mytesthook;

import com.google.common.collect.ImmutableMap;
import com.mycompany.testing.mytesthook.model.aws.s3.bucket.AwsS3Bucket;
import com.mycompany.testing.mytesthook.model.aws.s3.bucket.BucketEncryption;
import
 com.mycompany.testing.mytesthook.model.aws.s3.bucket.ServerSideEncryptionByDefault;
import
 com.mycompany.testing.mytesthook.model.aws.s3.bucket.ServerSideEncryptionRule;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import org.mockito.Mock;
import org.mockito.junit.jupiter.MockitoExtension;
import software.amazon.cloudformation.exceptions.UnsupportedTargetException;
import software.amazon.cloudformation.proxy.AmazonWebServicesClientProxy;
import software.amazon.cloudformation.proxy.HandlerErrorCode;
import software.amazon.cloudformation.proxy.Logger;
import software.amazon.cloudformation.proxy.hook.HookContext;
import software.amazon.cloudformation.proxy.hook.HookHandlerRequest;
import software.amazon.cloudformation.proxy.hook.HookProgressEvent;
import software.amazon.cloudformation.proxy.hook.HookStatus;
import software.amazon.cloudformation.proxy.hook.targetmodel.HookTargetModel;

import java.util.Arrays;
import java.util.stream.Stream;

import static org.assertj.core.api.Assertions.assertThat;
import static org.assertj.core.api.Assertions.assertThatExceptionOfType;
import static org.mockito.Mockito.mock;

@ExtendWith(MockitoExtension.class)
public class PreUpdateHookHandlerTest {

    @Mock
    private AmazonWebServicesClientProxy proxy;
```

```
    @Mock
    private Logger logger;

    @BeforeEach
    public void setup() {
        proxy = mock(AmazonWebServicesClientProxy.class);
        logger = mock(Logger.class);
    }

    @Test
    public void handleRequest_awsS3BucketSuccess() {
        final PreUpdateHookHandler handler = new PreUpdateHookHandler();

        final ServerSideEncryptionRule serverSideEncryptionRule =
 buildServerSideEncryptionRule("AES256");
        final AwsS3Bucket resourceProperties = buildAwsS3Bucket("amzn-s3-demo-
bucket", serverSideEncryptionRule);
        final AwsS3Bucket previousResourceProperties = buildAwsS3Bucket("amzn-s3-
demo-bucket", serverSideEncryptionRule);
        final HookTargetModel targetModel =
 createHookTargetModel(resourceProperties, previousResourceProperties);
        final TypeConfigurationModel typeConfiguration =
 TypeConfigurationModel.builder().encryptionAlgorithm("AES256").build();

        final HookHandlerRequest request = HookHandlerRequest.builder()

 .hookContext(HookContext.builder().targetName("AWS::S3::Bucket").targetModel(targetModel).
            .build();

        final HookProgressEvent<CallbackContext> response =
 handler.handleRequest(proxy, request, null, logger, typeConfiguration);
        assertResponse(response, HookStatus.SUCCESS, "Successfully invoked
 PreUpdateHookHandler for target: AWS::SQS::Queue");
    }

    @Test
    public void handleRequest_awsS3BucketFail_bucketEncryptionConfigsDontMatch() {
        final PreUpdateHookHandler handler = new PreUpdateHookHandler();

        final ServerSideEncryptionRule[] serverSideEncryptionRules =
 Stream.of("AES256", "SHA512", "AES32")
                .map(this::buildServerSideEncryptionRule)
                .toArray(ServerSideEncryptionRule[]::new);
```

```
        final AwsS3Bucket resourceProperties = buildAwsS3Bucket("amzn-s3-demo-
bucket", serverSideEncryptionRules[0]);
        final AwsS3Bucket previousResourceProperties = buildAwsS3Bucket("amzn-s3-
demo-bucket", serverSideEncryptionRules);
        final HookTargetModel targetModel =
 createHookTargetModel(resourceProperties, previousResourceProperties);
        final TypeConfigurationModel typeConfiguration =
 TypeConfigurationModel.builder().encryptionAlgorithm("AES256").build();

        final HookHandlerRequest request = HookHandlerRequest.builder()

 .hookContext(HookContext.builder().targetName("AWS::S3::Bucket").targetModel(targetModel).
            .build();

        final HookProgressEvent<CallbackContext> response =
 handler.handleRequest(proxy, request, null, logger, typeConfiguration);
        assertResponse(response, HookStatus.FAILED, "Current number of bucket
 encryption configs does not match previous. Current has 1 configs while previously
 there were 3 configs");
    }

    @Test
    public void
 handleRequest_awsS3BucketFail_bucketEncryptionAlgorithmDoesNotMatch() {
        final PreUpdateHookHandler handler = new PreUpdateHookHandler();

        final AwsS3Bucket resourceProperties = buildAwsS3Bucket("amzn-s3-demo-
bucket", buildServerSideEncryptionRule("SHA512"));
        final AwsS3Bucket previousResourceProperties = buildAwsS3Bucket("amzn-s3-
demo-bucket", buildServerSideEncryptionRule("AES256"));
        final HookTargetModel targetModel =
 createHookTargetModel(resourceProperties, previousResourceProperties);
        final TypeConfigurationModel typeConfiguration =
 TypeConfigurationModel.builder().encryptionAlgorithm("AES256").build();

        final HookHandlerRequest request = HookHandlerRequest.builder()

 .hookContext(HookContext.builder().targetName("AWS::S3::Bucket").targetModel(targetModel).
            .build();

        final HookProgressEvent<CallbackContext> response =
 handler.handleRequest(proxy, request, null, logger, typeConfiguration);
```

```
        assertResponse(response, HookStatus.FAILED, String.format("Bucket
Encryption algorithm can not be changed once set. The encryption algorithm was
changed to '%s' from '%s'.", "SHA512", "AES256"));
    }

    @Test
    public void handleRequest_unsupportedTarget() {
        final PreUpdateHookHandler handler = new PreUpdateHookHandler();

        final Object resourceProperties = ImmutableMap.of("FileSizeLimit", 256);
        final Object previousResourceProperties = ImmutableMap.of("FileSizeLimit",
512);
        final HookTargetModel targetModel =
createHookTargetModel(resourceProperties, previousResourceProperties);
        final TypeConfigurationModel typeConfiguration =
TypeConfigurationModel.builder().encryptionAlgorithm("AES256").build();

        final HookHandlerRequest request = HookHandlerRequest.builder()

.hookContext(HookContext.builder().targetName("AWS::Unsupported::Target").targetModel(targ
            .build();

        assertThatExceptionOfType(UnsupportedTargetException.class)
                .isThrownBy(() -> handler.handleRequest(proxy, request, null,
logger, typeConfiguration))
                .withMessageContaining("Unsupported target")
                .withMessageContaining("AWS::Unsupported::Target")
                .satisfies(e ->
assertThat(e.getErrorCode()).isEqualTo(HandlerErrorCode.InvalidRequest));
    }

    private void assertResponse(final HookProgressEvent<CallbackContext> response,
final HookStatus expectedStatus, final String expectedErrorMsg) {
        assertThat(response).isNotNull();
        assertThat(response.getStatus()).isEqualTo(expectedStatus);
        assertThat(response.getCallbackContext()).isNull();
        assertThat(response.getCallbackDelaySeconds()).isEqualTo(0);
        assertThat(response.getMessage()).isNotNull();
        assertThat(response.getMessage()).isEqualTo(expectedErrorMsg);
    }

    private HookTargetModel createHookTargetModel(final Object resourceProperties,
final Object previousResourceProperties) {
        return HookTargetModel.of(
```

```
                ImmutableMap.of(
                    "ResourceProperties", resourceProperties,
                    "PreviousResourceProperties", previousResourceProperties
                )
        );
    }

    @SuppressWarnings("SameParameterValue")
    private AwsS3Bucket buildAwsS3Bucket(
            final String bucketName,
            final ServerSideEncryptionRule ...serverSideEncryptionRules
            ) {
        return AwsS3Bucket.builder()
                .bucketName(bucketName)
                .bucketEncryption(
                    BucketEncryption.builder()
                        .serverSideEncryptionConfiguration(
                                Arrays.asList(serverSideEncryptionRules)
                        ).build()
                ).build();
    }

    private ServerSideEncryptionRule buildServerSideEncryptionRule(final String
 encryptionAlgorithm) {
        return ServerSideEncryptionRule.builder()
                .bucketKeyEnabled(true)
                .serverSideEncryptionByDefault(
                        ServerSideEncryptionByDefault.builder()
                                .sSEAlgorithm(encryptionAlgorithm)
                                .build()
                ).build();
    }
}
```

Implementación del **`preDelete`** controlador

Implemente un `preDelete` controlador, que se inicie antes de las operaciones de eliminación para todos los objetivos especificados en el controlador. El `preDelete` controlador logra lo siguiente:

- En el caso `AWS::S3::Bucket` de un recurso, el Hook solo pasará si se cumple lo siguiente:
  - Verifica que los recursos de quejas mínimos requeridos existan en la cuenta después de eliminar el recurso.

- La cantidad mínima requerida de recursos para quejas se establece en la configuración de tipos de Hook.

Codificación del **`preDelete`** controlador

1. En tuIDE, abre el `PreDeleteHookHandler.java` archivo de la `src/main/java/com/mycompany/testing/mytesthook` carpeta.
2. Sustituya todo el contenido del `PreDeleteHookHandler.java` archivo por el siguiente código.

```
package com.mycompany.testing.mytesthook;

import com.google.common.annotations.VisibleForTesting;
import com.mycompany.testing.mytesthook.model.aws.s3.bucket.AwsS3Bucket;
import com.mycompany.testing.mytesthook.model.aws.s3.bucket.AwsS3BucketTargetModel;
import com.mycompany.testing.mytesthook.model.aws.sqs.queue.AwsSqsQueue;
import com.mycompany.testing.mytesthook.model.aws.sqs.queue.AwsSqsQueueTargetModel;
import org.apache.commons.lang3.StringUtils;
import org.apache.commons.lang3.math.NumberUtils;
import software.amazon.awssdk.services.cloudformation.CloudFormationClient;
import
 software.amazon.awssdk.services.cloudformation.model.CloudFormationException;
import
 software.amazon.awssdk.services.cloudformation.model.DescribeStackResourceRequest;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.ListQueuesRequest;
import software.amazon.awssdk.services.sqs.model.ListQueuesResponse;
import software.amazon.awssdk.services.sqs.model.QueueAttributeName;
import software.amazon.awssdk.services.sqs.model.SqsException;
import software.amazon.cloudformation.exceptions.CfnGeneralServiceException;
import software.amazon.cloudformation.proxy.AmazonWebServicesClientProxy;
import software.amazon.cloudformation.proxy.HandlerErrorCode;
import software.amazon.cloudformation.proxy.OperationStatus;
import software.amazon.cloudformation.proxy.ProgressEvent;
import software.amazon.cloudformation.proxy.ProxyClient;
import software.amazon.cloudformation.proxy.hook.HookContext;
import software.amazon.cloudformation.proxy.hook.HookHandlerRequest;
import software.amazon.cloudformation.proxy.hook.targetmodel.HookTargetModel;
```

```
import
 software.amazon.cloudformation.proxy.hook.targetmodel.ResourceHookTargetModel;

import java.util.ArrayList;
import java.util.Collection;
import java.util.HashSet;
import java.util.List;
import java.util.Objects;
import java.util.stream.Collectors;

public class PreDeleteHookHandler extends BaseHookHandlerStd {

    private ProxyClient<S3Client> s3Client;
    private ProxyClient<SqsClient> sqsClient;

    @Override
    protected ProgressEvent<HookTargetModel, CallbackContext>
 handleS3BucketRequest(
            final AmazonWebServicesClientProxy proxy,
            final HookHandlerRequest request,
            final CallbackContext callbackContext,
            final ProxyClient<S3Client> proxyClient,
            final TypeConfigurationModel typeConfiguration
    ) {
        final HookContext hookContext = request.getHookContext();
        final String targetName = hookContext.getTargetName();
        if (!AwsS3Bucket.TYPE_NAME.equals(targetName)) {
            throw new RuntimeException(String.format("Request target type [%s] is
 not 'AWS::S3::Bucket'", targetName));
        }
        this.s3Client = proxyClient;

        final String encryptionAlgorithm =
 typeConfiguration.getEncryptionAlgorithm();
        final int minBuckets =
 NumberUtils.toInt(typeConfiguration.getMinBuckets());

        final ResourceHookTargetModel<AwsS3Bucket> targetModel =
 hookContext.getTargetModel(AwsS3BucketTargetModel.class);
        final List<String> buckets = listBuckets().stream()
                .filter(b -> !StringUtils.equals(b,
 targetModel.getResourceProperties().getBucketName()))
                .collect(Collectors.toList());
```

```
        final List<String> compliantBuckets = new ArrayList<>();
        for (final String bucket : buckets) {
            if (getBucketSSEAlgorithm(bucket).contains(encryptionAlgorithm)) {
                compliantBuckets.add(bucket);
            }

            if (compliantBuckets.size() >= minBuckets) {
                return ProgressEvent.<HookTargetModel, CallbackContext>builder()
                        .status(OperationStatus.SUCCESS)
                        .message("Successfully invoked PreDeleteHookHandler for
target: AWS::S3::Bucket")
                        .build();
            }
        }

        return ProgressEvent.<HookTargetModel, CallbackContext>builder()
                .status(OperationStatus.FAILED)
                .errorCode(HandlerErrorCode.NonCompliant)
                .message(String.format("Failed to meet minimum of [%d] encrypted
buckets.", minBuckets))
                .build();
    }

    @Override
    protected ProgressEvent<HookTargetModel, CallbackContext>
handleSqsQueueRequest(
            final AmazonWebServicesClientProxy proxy,
            final HookHandlerRequest request,
            final CallbackContext callbackContext,
            final ProxyClient<SqsClient> proxyClient,
            final TypeConfigurationModel typeConfiguration
    ) {
        final HookContext hookContext = request.getHookContext();
        final String targetName = hookContext.getTargetName();
        if (!AwsSqsQueue.TYPE_NAME.equals(targetName)) {
            throw new RuntimeException(String.format("Request target type [%s] is
not 'AWS::SQS::Queue'", targetName));
        }
        this.sqsClient = proxyClient;
        final int minQueues = NumberUtils.toInt(typeConfiguration.getMinQueues());

        final ResourceHookTargetModel<AwsSqsQueue> targetModel =
hookContext.getTargetModel(AwsSqsQueueTargetModel.class);
```

```
        final String queueName =
Objects.toString(targetModel.getResourceProperties().get("QueueName"), null);

        String targetQueueUrl = null;
        if (queueName != null) {
            try {
                targetQueueUrl = sqsClient.injectCredentialsAndInvokeV2(
                        GetQueueUrlRequest.builder().queueName(
                                queueName
                        ).build(),
                        sqsClient.client()::getQueueUrl
                ).queueUrl();
            } catch (SqsException e) {
                log(String.format("Error while calling GetQueueUrl API for queue
name [%s]: %s", queueName, e.getMessage()));
            }
        } else {
            log("Queue name is empty, attempting to get queue's physical ID");
            try {
                final ProxyClient<CloudFormationClient> cfnClient =
proxy.newProxy(ClientBuilder::createCloudFormationClient);
                targetQueueUrl = cfnClient.injectCredentialsAndInvokeV2(
                        DescribeStackResourceRequest.builder()
                                .stackName(hookContext.getTargetLogicalId())

.logicalResourceId(hookContext.getTargetLogicalId())
                                .build(),
                        cfnClient.client()::describeStackResource
                ).stackResourceDetail().physicalResourceId();
            } catch (CloudFormationException e) {
                log(String.format("Error while calling DescribeStackResource API
for queue name: %s", e.getMessage()));
            }
        }

        // Creating final variable for the filter lambda
        final String finalTargetQueueUrl = targetQueueUrl;

        final List<String> compliantQueues = new ArrayList<>();

        String nextToken = null;
        do {
            final ListQueuesRequest req =
Translator.createListQueuesRequest(nextToken);
```

```
            final ListQueuesResponse res =
sqsClient.injectCredentialsAndInvokeV2(req, sqsClient.client()::listQueues);
            final List<String> queueUrls = res.queueUrls().stream()
                    .filter(q -> !StringUtils.equals(q, finalTargetQueueUrl))
                    .collect(Collectors.toList());

            for (final String queueUrl : queueUrls) {
                if (isQueueEncrypted(queueUrl)) {
                    compliantQueues.add(queueUrl);
                }

                if (compliantQueues.size() >= minQueues) {
                    return ProgressEvent.<HookTargetModel,
CallbackContext>builder()
                            .status(OperationStatus.SUCCESS)
                            .message("Successfully invoked PreDeleteHookHandler for
target: AWS::SQS::Queue")
                            .build();
                }
                nextToken = res.nextToken();
            }
        } while (nextToken != null);

        return ProgressEvent.<HookTargetModel, CallbackContext>builder()
                .status(OperationStatus.FAILED)
                .errorCode(HandlerErrorCode.NonCompliant)
                .message(String.format("Failed to meet minimum of [%d] encrypted
queues.", minQueues))
                .build();
    }

    private List<String> listBuckets() {
        try {
            return
s3Client.injectCredentialsAndInvokeV2(Translator.createListBucketsRequest(),
s3Client.client()::listBuckets)
                    .buckets()
                    .stream()
                    .map(Bucket::name)
                    .collect(Collectors.toList());
        } catch (S3Exception e) {
            throw new CfnGeneralServiceException("Error while calling S3
ListBuckets API", e);
        }
```

```
    }

    @VisibleForTesting
    Collection<String> getBucketSSEAlgorithm(final String bucket) {
        try {
            return
 s3Client.injectCredentialsAndInvokeV2(Translator.createGetBucketEncryptionRequest(bucket),
 s3Client.client()::getBucketEncryption)
                    .serverSideEncryptionConfiguration()
                    .rules()
                    .stream()
                    .filter(r ->
 Objects.nonNull(r.applyServerSideEncryptionByDefault()))
                    .map(r ->
 r.applyServerSideEncryptionByDefault().sseAlgorithmAsString())
                    .collect(Collectors.toSet());
        } catch (S3Exception e) {
            return new HashSet<>();
        }
    }

    @VisibleForTesting
    boolean isQueueEncrypted(final String queueUrl) {
        try {
            final GetQueueAttributesRequest request =
 GetQueueAttributesRequest.builder()
                    .queueUrl(queueUrl)
                    .attributeNames(QueueAttributeName.KMS_MASTER_KEY_ID)
                    .build();
            final String kmsKeyId = sqsClient.injectCredentialsAndInvokeV2(request,
 sqsClient.client()::getQueueAttributes)
                    .attributes()
                    .get(QueueAttributeName.KMS_MASTER_KEY_ID);

            return StringUtils.isNotBlank(kmsKeyId);
        } catch (SqsException e) {
            throw new CfnGeneralServiceException("Error while calling SQS
 GetQueueAttributes API", e);
        }
    }
}
```

Actualización del **`preDelete`** controlador

1. En su carpetaIDE, abra el `PreDeleteHookHandler.java` archivo de la `src/main/java/com/mycompany/testing/mytesthook` carpeta.
2. Sustituya todo el contenido del `PreDeleteHookHandler.java` archivo por el siguiente código.

```
package com.mycompany.testing.mytesthook;

import com.google.common.collect.ImmutableList;
import com.google.common.collect.ImmutableMap;
import com.mycompany.testing.mytesthook.model.aws.s3.bucket.AwsS3Bucket;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import org.mockito.Mock;
import org.mockito.Mockito;
import org.mockito.junit.jupiter.MockitoExtension;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.GetBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.GetBucketEncryptionResponse;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionByDefault;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionConfiguration;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionRule;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesResponse;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;
import software.amazon.awssdk.services.sqs.model.ListQueuesRequest;
import software.amazon.awssdk.services.sqs.model.ListQueuesResponse;
import software.amazon.awssdk.services.sqs.model.QueueAttributeName;
import software.amazon.cloudformation.proxy.Logger;
import software.amazon.cloudformation.proxy.OperationStatus;
import software.amazon.cloudformation.proxy.ProgressEvent;
import software.amazon.cloudformation.proxy.hook.HookContext;
import software.amazon.cloudformation.proxy.hook.HookHandlerRequest;
import software.amazon.cloudformation.proxy.hook.targetmodel.HookTargetModel;

import java.util.Arrays;
```

```
import java.util.Collection;
import java.util.HashMap;
import java.util.List;
import java.util.stream.Collectors;

import static org.mockito.ArgumentMatchers.any;
import static org.mockito.Mockito.mock;
import static org.mockito.Mockito.never;
import static org.mockito.Mockito.times;
import static org.mockito.Mockito.verify;
import static org.mockito.Mockito.when;

@ExtendWith(MockitoExtension.class)
public class PreDeleteHookHandlerTest extends AbstractTestBase {

    @Mock private S3Client s3Client;
    @Mock private SqsClient sqsClient;
    @Mock private Logger logger;

    @BeforeEach
    public void setup() {
        s3Client = mock(S3Client.class);
        sqsClient = mock(SqsClient.class);
        logger = mock(Logger.class);
    }

    @Test
    public void handleRequest_awsS3BucketSuccess() {
        final PreDeleteHookHandler handler = Mockito.spy(new
 PreDeleteHookHandler());

        final List<Bucket> bucketList = ImmutableList.of(
                Bucket.builder().name("bucket1").build(),
                Bucket.builder().name("bucket2").build(),
                Bucket.builder().name("toBeDeletedBucket").build(),
                Bucket.builder().name("bucket3").build(),
                Bucket.builder().name("bucket4").build(),
                Bucket.builder().name("bucket5").build()
        );
        final ListBucketsResponse mockResponse =
 ListBucketsResponse.builder().buckets(bucketList).build();

 when(s3Client.listBuckets(any(ListBucketsRequest.class))).thenReturn(mockResponse);
        when(s3Client.getBucketEncryption(any(GetBucketEncryptionRequest.class)))
```

```
                .thenReturn(buildGetBucketEncryptionResponse("AES256"))
                .thenReturn(buildGetBucketEncryptionResponse("AES256", "aws:kms"))
                .thenThrow(S3Exception.builder().message("No Encrypt").build())
                .thenReturn(buildGetBucketEncryptionResponse("aws:kms"))
                .thenReturn(buildGetBucketEncryptionResponse("AES256"));
        setServiceClient(s3Client);

        final TypeConfigurationModel typeConfiguration =
TypeConfigurationModel.builder()
                .encryptionAlgorithm("AES256")
                .minBuckets("3")
                .build();

        final HookHandlerRequest request = HookHandlerRequest.builder()
                .hookContext(
                    HookContext.builder()
                        .targetName("AWS::S3::Bucket")
                        .targetModel(
                            createHookTargetModel(
                                AwsS3Bucket.builder()
                                    .bucketName("toBeDeletedBucket")
                                    .build()
                            )
                        )
                        .build())
                .build();

        final ProgressEvent<HookTargetModel, CallbackContext> response =
handler.handleRequest(proxy, request, null, logger, typeConfiguration);

        verify(s3Client,
times(5)).getBucketEncryption(any(GetBucketEncryptionRequest.class));
        verify(handler, never()).getBucketSSEAlgorithm("toBeDeletedBucket");

        assertResponse(response, OperationStatus.SUCCESS, "Successfully invoked
PreDeleteHookHandler for target: AWS::S3::Bucket");
    }


    @Test
    public void handleRequest_awsSqsQueueSuccess() {
        final PreDeleteHookHandler handler = Mockito.spy(new
PreDeleteHookHandler());
```

```
        final List<String> queueUrls = ImmutableList.of(
                "https://queue1.queue",
                "https://queue2.queue",
                "https://toBeDeletedQueue.queue",
                "https://queue3.queue",
                "https://queue4.queue",
                "https://queue5.queue"
        );

        when(sqsClient.getQueueUrl(any(GetQueueUrlRequest.class)))
                .thenReturn(GetQueueUrlResponse.builder().queueUrl("https://
toBeDeletedQueue.queue").build());
        when(sqsClient.listQueues(any(ListQueuesRequest.class)))

 .thenReturn(ListQueuesResponse.builder().queueUrls(queueUrls).build());
        when(sqsClient.getQueueAttributes(any(GetQueueAttributesRequest.class)))

 .thenReturn(GetQueueAttributesResponse.builder().attributes(ImmutableMap.of(QueueAttribute
 "kmsKeyId")).build())
                .thenReturn(GetQueueAttributesResponse.builder().attributes(new
 HashMap<>()).build())

 .thenReturn(GetQueueAttributesResponse.builder().attributes(ImmutableMap.of(QueueAttribute
 "kmsKeyId")).build())
                .thenReturn(GetQueueAttributesResponse.builder().attributes(new
 HashMap<>()).build())

 .thenReturn(GetQueueAttributesResponse.builder().attributes(ImmutableMap.of(QueueAttribute
 "kmsKeyId")).build());
        setServiceClient(sqsClient);

        final TypeConfigurationModel typeConfiguration =
 TypeConfigurationModel.builder()
                .minQueues("3")
                .build();

        final HookHandlerRequest request = HookHandlerRequest.builder()
                .hookContext(
                    HookContext.builder()
                        .targetName("AWS::SQS::Queue")
                        .targetModel(
                            createHookTargetModel(
                                ImmutableMap.of("QueueName", "toBeDeletedQueue")
                            )
```

```
                        )
                        .build())
                .build();

        final ProgressEvent<HookTargetModel, CallbackContext> response =
handler.handleRequest(proxy, request, null, logger, typeConfiguration);

        verify(sqsClient,
times(5)).getQueueAttributes(any(GetQueueAttributesRequest.class));
        verify(handler, never()).isQueueEncrypted("toBeDeletedQueue");

        assertResponse(response, OperationStatus.SUCCESS, "Successfully invoked
PreDeleteHookHandler for target: AWS::SQS::Queue");
    }

    @Test
    public void handleRequest_awsS3BucketFailed() {
        final PreDeleteHookHandler handler = Mockito.spy(new
PreDeleteHookHandler());

        final List<Bucket> bucketList = ImmutableList.of(
                Bucket.builder().name("bucket1").build(),
                Bucket.builder().name("bucket2").build(),
                Bucket.builder().name("toBeDeletedBucket").build(),
                Bucket.builder().name("bucket3").build(),
                Bucket.builder().name("bucket4").build(),
                Bucket.builder().name("bucket5").build()
        );
        final ListBucketsResponse mockResponse =
ListBucketsResponse.builder().buckets(bucketList).build();

when(s3Client.listBuckets(any(ListBucketsRequest.class))).thenReturn(mockResponse);
        when(s3Client.getBucketEncryption(any(GetBucketEncryptionRequest.class)))
                .thenReturn(buildGetBucketEncryptionResponse("AES256"))
                .thenReturn(buildGetBucketEncryptionResponse("AES256", "aws:kms"))
                .thenThrow(S3Exception.builder().message("No Encrypt").build())
                .thenReturn(buildGetBucketEncryptionResponse("aws:kms"))
                .thenReturn(buildGetBucketEncryptionResponse("AES256"));
        setServiceClient(s3Client);

        final TypeConfigurationModel typeConfiguration =
TypeConfigurationModel.builder()
                .encryptionAlgorithm("AES256")
                .minBuckets("10")
```

```
                .build();

        final HookHandlerRequest request = HookHandlerRequest.builder()
                .hookContext(
                    HookContext.builder()
                        .targetName("AWS::S3::Bucket")
                        .targetModel(
                            createHookTargetModel(
                                AwsS3Bucket.builder()
                                    .bucketName("toBeDeletedBucket")
                                    .build()
                            )
                        )
                        .build())
                .build();

        final ProgressEvent<HookTargetModel, CallbackContext> response =
handler.handleRequest(proxy, request, null, logger, typeConfiguration);

        verify(s3Client,
times(5)).getBucketEncryption(any(GetBucketEncryptionRequest.class));
        verify(handler, never()).getBucketSSEAlgorithm("toBeDeletedBucket");

        assertResponse(response, OperationStatus.FAILED, "Failed to meet minimum of
[10] encrypted buckets.");
    }


    @Test
    public void handleRequest_awsSqsQueueFailed() {
        final PreDeleteHookHandler handler = Mockito.spy(new
PreDeleteHookHandler());

        final List<String> queueUrls = ImmutableList.of(
                "https://queue1.queue",
                "https://queue2.queue",
                "https://toBeDeletedQueue.queue",
                "https://queue3.queue",
                "https://queue4.queue",
                "https://queue5.queue"
        );

        when(sqsClient.getQueueUrl(any(GetQueueUrlRequest.class)))
```

```
                .thenReturn(GetQueueUrlResponse.builder().queueUrl("https://
toBeDeletedQueue.queue").build());
        when(sqsClient.listQueues(any(ListQueuesRequest.class)))

 .thenReturn(ListQueuesResponse.builder().queueUrls(queueUrls).build());
        when(sqsClient.getQueueAttributes(any(GetQueueAttributesRequest.class)))

 .thenReturn(GetQueueAttributesResponse.builder().attributes(ImmutableMap.of(QueueAttribute
 "kmsKeyId")).build())
                .thenReturn(GetQueueAttributesResponse.builder().attributes(new
 HashMap<>()).build())

 .thenReturn(GetQueueAttributesResponse.builder().attributes(ImmutableMap.of(QueueAttribute
 "kmsKeyId")).build())
                .thenReturn(GetQueueAttributesResponse.builder().attributes(new
 HashMap<>()).build())

 .thenReturn(GetQueueAttributesResponse.builder().attributes(ImmutableMap.of(QueueAttribute
 "kmsKeyId")).build());
        setServiceClient(sqsClient);

        final TypeConfigurationModel typeConfiguration =
 TypeConfigurationModel.builder()
                .minQueues("10")
                .build();

        final HookHandlerRequest request = HookHandlerRequest.builder()
                .hookContext(
                    HookContext.builder()
                        .targetName("AWS::SQS::Queue")
                        .targetModel(
                            createHookTargetModel(
                                ImmutableMap.of("QueueName", "toBeDeletedQueue")
                            )
                        )
                        .build())
                .build();

        final ProgressEvent<HookTargetModel, CallbackContext> response =
 handler.handleRequest(proxy, request, null, logger, typeConfiguration);

        verify(sqsClient,
 times(5)).getQueueAttributes(any(GetQueueAttributesRequest.class));
        verify(handler, never()).isQueueEncrypted("toBeDeletedQueue");
```

```
        assertResponse(response, OperationStatus.FAILED, "Failed to meet minimum of
 [10] encrypted queues.");
    }

    private GetBucketEncryptionResponse buildGetBucketEncryptionResponse(final
 String ...sseAlgorithm) {
        return buildGetBucketEncryptionResponse(
                Arrays.stream(sseAlgorithm)
                    .map(a ->
 ServerSideEncryptionRule.builder().applyServerSideEncryptionByDefault(
                        ServerSideEncryptionByDefault.builder()
                            .sseAlgorithm(a)
                            .build()
                        ).build()
                    )
                    .collect(Collectors.toList())
        );
    }

    private GetBucketEncryptionResponse buildGetBucketEncryptionResponse(final
 Collection<ServerSideEncryptionRule> rules) {
        return GetBucketEncryptionResponse.builder()
                .serverSideEncryptionConfiguration(
                    ServerSideEncryptionConfiguration.builder().rules(
                        rules
                    ).build()
                ).build();
    }
}
```

Modelado de AWS CloudFormation Hooks personalizados con Python

Modelar AWS CloudFormation Hooks personalizados implica crear un esquema que defina el Hook, sus propiedades y sus atributos. Este tutorial te guía a través del modelado de Hooks personalizados con Python.

Paso 1: Genera el paquete del proyecto Hook

Genera tu paquete de proyectos Hook. CloudFormation CLICrea funciones de controlador vacías que corresponden a acciones específicas de Hook en el ciclo de vida objetivo, tal como se define en la especificación de Hook.

```
cfn generate
```

El comando devuelve el resultado siguiente.

```
Generated files for MyCompany::Testing::MyTestHook
```

> **Note**
>
> Asegúrese de que sus tiempos de ejecución de Lambda eviten el uso up-to-date de una versión obsoleta. Para obtener más información, consulte Actualización de los tiempos de ejecución de Lambda para tipos de recursos y Hooks.

Paso 2: Añadir controladores de Hook

Agrega tu propio código de tiempo de ejecución del controlador Hook a los controladores que decidas implementar. Por ejemplo, puedes añadir el siguiente código para el registro.

```
LOG.setLevel(logging.INFO)
LOG.info("Internal testing Hook triggered for target: " +
 request.hookContext.targetName);
```

CloudFormation CLIGenera el `src/models.py` archivo a partir delEsquema de configuración.

Example models.py

```
import sys
from dataclasses import dataclass
from inspect import getmembers, isclass
from typing import (
    AbstractSet,
    Any,
    Generic,
    Mapping,
    MutableMapping,
    Optional,
    Sequence,
    Type,
    TypeVar,
)
```

```
from cloudformation_cli_python_lib.interface import (
    BaseModel,
    BaseHookHandlerRequest,
)
from cloudformation_cli_python_lib.recast import recast_object
from cloudformation_cli_python_lib.utils import deserialize_list

T = TypeVar("T")


def set_or_none(value: Optional[Sequence[T]]) -> Optional[AbstractSet[T]]:
    if value:
        return set(value)
    return None


@dataclass
class HookHandlerRequest(BaseHookHandlerRequest):
    pass


@dataclass
class TypeConfigurationModel(BaseModel):
    limitSize: Optional[str]
    cidr: Optional[str]
    encryptionAlgorithm: Optional[str]

    @classmethod
    def _deserialize(
        cls: Type["_TypeConfigurationModel"],
        json_data: Optional[Mapping[str, Any]],
    ) -> Optional["_TypeConfigurationModel"]:
        if not json_data:
            return None
        return cls(
            limitSize=json_data.get("limitSize"),
            cidr=json_data.get("cidr"),
            encryptionAlgorithm=json_data.get("encryptionAlgorithm"),
        )


_TypeConfigurationModel = TypeConfigurationModel
```

Paso 3: Implementa los controladores de Hook

Con las clases de datos de Python generadas, puedes escribir los controladores que realmente implementan la funcionalidad del Hook. En este ejemplo, implementarás los puntos de `preDelete` invocación y `preCreatepreUpdate`, para los controladores.

Temas

Implementa el controlador preCreate

El `preCreate` controlador verifica la configuración de cifrado del lado del servidor para un recurso o. `AWS::S3::Bucket` `AWS::SQS::Queue`

- En el caso `AWS::S3::Bucket` de un recurso, el Hook solo pasará si se cumple lo siguiente.
  - Se ha establecido el cifrado del bucket de Amazon S3.
  - La clave de bucket de Amazon S3 está habilitada para el bucket.
  - El algoritmo de cifrado establecido para el bucket de Amazon S3 es el algoritmo correcto que se requiere.
  - El ID de AWS Key Management Service clave está establecido.
- En el caso `AWS::SQS::Queue` de un recurso, el Hook solo pasará si se cumple lo siguiente.
  - El ID de AWS Key Management Service clave está establecido.

Implemente el preUpdate controlador

Implemente un `preUpdate` controlador, que se inicie antes de las operaciones de actualización para todos los objetivos especificados en el controlador. El `preUpdate` controlador logra lo siguiente:

- En el caso `AWS::S3::Bucket` de un recurso, el Hook solo pasará si se cumple lo siguiente:
  - El algoritmo de cifrado de buckets de un bucket de Amazon S3 no se ha modificado.

Implemente el preDelete controlador

Implemente un `preDelete` controlador, que se inicie antes de las operaciones de eliminación para todos los objetivos especificados en el controlador. El `preDelete` controlador logra lo siguiente:

- En el caso `AWS::S3::Bucket` de un recurso, el Hook solo pasará si se cumple lo siguiente:
  - Verifica que existan en la cuenta los recursos conformes mínimos requeridos después de eliminar el recurso.
  - La cantidad mínima de recursos compatibles requerida se establece en la configuración del Hook.

Implementa un controlador de Hook

1. En su interiorIDE, abra el `handlers.py` archivo que se encuentra en la `src` carpeta.
2. Sustituya todo el contenido del `handlers.py` archivo por el siguiente código.

   Example handlers.py

```
import logging
from typing import Any, MutableMapping, Optional
import botocore

from cloudformation_cli_python_lib import (
    BaseHookHandlerRequest,
    HandlerErrorCode,
    Hook,
    HookInvocationPoint,
    OperationStatus,
    ProgressEvent,
    SessionProxy,
    exceptions,
)

from .models import HookHandlerRequest, TypeConfigurationModel

# Use this logger to forward log messages to CloudWatch Logs.
LOG = logging.getLogger(__name__)
TYPE_NAME = "MyCompany::Testing::MyTestHook"

LOG.setLevel(logging.INFO)
```

```
hook = Hook(TYPE_NAME, TypeConfigurationModel)
test_entrypoint = hook.test_entrypoint


def _validate_s3_bucket_encryption(
    bucket: MutableMapping[str, Any], required_encryption_algorithm: str
) -> ProgressEvent:
    status = None
    message = ""
    error_code = None

    if bucket:
        bucket_name = bucket.get("BucketName")

        bucket_encryption = bucket.get("BucketEncryption")
        if bucket_encryption:
            server_side_encryption_rules = bucket_encryption.get(
                "ServerSideEncryptionConfiguration"
            )
            if server_side_encryption_rules:
                for rule in server_side_encryption_rules:
                    bucket_key_enabled = rule.get("BucketKeyEnabled")
                    if bucket_key_enabled:
                        server_side_encryption_by_default = rule.get(
                            "ServerSideEncryptionByDefault"
                        )

                        encryption_algorithm =
 server_side_encryption_by_default.get(
                            "SSEAlgorithm"
                        )
                        kms_key_id = server_side_encryption_by_default.get(
                            "KMSMasterKeyID"
                        )  # "KMSMasterKeyID" is name of the property for an
 AWS::S3::Bucket

                        if encryption_algorithm == required_encryption_algorithm:
                            if encryption_algorithm == "aws:kms" and not
 kms_key_id:
                                status = OperationStatus.FAILED
                                message = f"KMS Key ID not set for bucket with
 name: f{bucket_name}"
                            else:
                                status = OperationStatus.SUCCESS
```

```
                                message = f"Successfully invoked
 PreCreateHookHandler for AWS::S3::Bucket with name: {bucket_name}"
                        else:
                            status = OperationStatus.FAILED
                            message = f"SSE Encryption Algorithm is incorrect for
 bucket with name: {bucket_name}"
                    else:
                        status = OperationStatus.FAILED
                        message = f"Bucket key not enabled for bucket with name:
 {bucket_name}"

                    if status == OperationStatus.FAILED:
                        break
            else:
                status = OperationStatus.FAILED
                message = f"No SSE Encryption configurations for bucket with name:
 {bucket_name}"
        else:
            status = OperationStatus.FAILED
            message = (
                f"Bucket Encryption not enabled for bucket with name:
 {bucket_name}"
            )
    else:
        status = OperationStatus.FAILED
        message = "Resource properties for S3 Bucket target model are empty"

    if status == OperationStatus.FAILED:
        error_code = HandlerErrorCode.NonCompliant

    return ProgressEvent(status=status, message=message, errorCode=error_code)


def _validate_sqs_queue_encryption(queue: MutableMapping[str, Any]) ->
 ProgressEvent:
    if not queue:
        return ProgressEvent(
            status=OperationStatus.FAILED,
            message="Resource properties for SQS Queue target model are empty",
            errorCode=HandlerErrorCode.NonCompliant,
        )
    queue_name = queue.get("QueueName")

    kms_key_id = queue.get(
```

```
        "KmsMasterKeyId"
    )  # "KmsMasterKeyId" is name of the property for an AWS::SQS::Queue
    if not kms_key_id:
        return ProgressEvent(
            status=OperationStatus.FAILED,
            message=f"Server side encryption turned off for queue with name:
 {queue_name}",
            errorCode=HandlerErrorCode.NonCompliant,
        )

    return ProgressEvent(
        status=OperationStatus.SUCCESS,
        message=f"Successfully invoked PreCreateHookHandler for
 targetAWS::SQS::Queue with name: {queue_name}",
    )


@hook.handler(HookInvocationPoint.CREATE_PRE_PROVISION)
def pre_create_handler(
    session: Optional[SessionProxy],
    request: HookHandlerRequest,
    callback_context: MutableMapping[str, Any],
    type_configuration: TypeConfigurationModel,
) -> ProgressEvent:
    target_name = request.hookContext.targetName
    if "AWS::S3::Bucket" == target_name:
        return _validate_s3_bucket_encryption(
            request.hookContext.targetModel.get("resourceProperties"),
            type_configuration.encryptionAlgorithm,
        )
    elif "AWS::SQS::Queue" == target_name:
        return _validate_sqs_queue_encryption(
            request.hookContext.targetModel.get("resourceProperties")
        )
    else:
        raise exceptions.InvalidRequest(f"Unknown target type: {target_name}")


def _validate_bucket_encryption_rules_not_updated(
    resource_properties, previous_resource_properties
) -> ProgressEvent:
    bucket_encryption_configs = resource_properties.get("BucketEncryption",
 {}).get(
        "ServerSideEncryptionConfiguration", []
```

```
    )
    previous_bucket_encryption_configs = previous_resource_properties.get(
        "BucketEncryption", {}
    ).get("ServerSideEncryptionConfiguration", [])

    if len(bucket_encryption_configs) != len(previous_bucket_encryption_configs):
        return ProgressEvent(
            status=OperationStatus.FAILED,
            message=f"Current number of bucket encryption configs does not
 match previous. Current has {str(len(bucket_encryption_configs))} configs while
 previously there were {str(len(previous_bucket_encryption_configs))} configs",
            errorCode=HandlerErrorCode.NonCompliant,
        )

    for i in range(len(bucket_encryption_configs)):
        current_encryption_algorithm = (
            bucket_encryption_configs[i]
            .get("ServerSideEncryptionByDefault", {})
            .get("SSEAlgorithm")
        )
        previous_encryption_algorithm = (
            previous_bucket_encryption_configs[i]
            .get("ServerSideEncryptionByDefault", {})
            .get("SSEAlgorithm")
        )

        if current_encryption_algorithm != previous_encryption_algorithm:
            return ProgressEvent(
                status=OperationStatus.FAILED,
                message=f"Bucket Encryption algorithm can not be changed once
 set. The encryption algorithm was changed to {current_encryption_algorithm} from
 {previous_encryption_algorithm}.",
                errorCode=HandlerErrorCode.NonCompliant,
            )

    return ProgressEvent(
        status=OperationStatus.SUCCESS,
        message="Successfully invoked PreUpdateHookHandler for target:
 AWS::SQS::Queue",
    )


def _validate_queue_encryption_not_disabled(
    resource_properties, previous_resource_properties
```

```
) -> ProgressEvent:
    if previous_resource_properties.get(
        "KmsMasterKeyId"
    ) and not resource_properties.get("KmsMasterKeyId"):
        return ProgressEvent(
            status=OperationStatus.FAILED,
            errorCode=HandlerErrorCode.NonCompliant,
            message="Queue encryption can not be disable",
        )
    else:
        return ProgressEvent(status=OperationStatus.SUCCESS)


@hook.handler(HookInvocationPoint.UPDATE_PRE_PROVISION)
def pre_update_handler(
    session: Optional[SessionProxy],
    request: BaseHookHandlerRequest,
    callback_context: MutableMapping[str, Any],
    type_configuration: MutableMapping[str, Any],
) -> ProgressEvent:
    target_name = request.hookContext.targetName
    if "AWS::S3::Bucket" == target_name:
        resource_properties =
 request.hookContext.targetModel.get("resourceProperties")
        previous_resource_properties = request.hookContext.targetModel.get(
            "previousResourceProperties"
        )

        return _validate_bucket_encryption_rules_not_updated(
            resource_properties, previous_resource_properties
        )
    elif "AWS::SQS::Queue" == target_name:
        resource_properties =
 request.hookContext.targetModel.get("resourceProperties")
        previous_resource_properties = request.hookContext.targetModel.get(
            "previousResourceProperties"
        )

        return _validate_queue_encryption_not_disabled(
            resource_properties, previous_resource_properties
        )
    else:
        raise exceptions.InvalidRequest(f"Unknown target type: {target_name}")
```

Continúe con el tema siguiente Registrar un Hook personalizado con AWS CloudFormation.

Registrar un Hook personalizado con AWS CloudFormation

Una vez que hayas creado un Hook personalizado, tendrás que registrarlo AWS CloudFormation para poder usarlo. En esta sección, aprenderás a empaquetar y registrar tu Hook para usarlo en tu Cuenta de AWS.

Package a Hook (Java)

Si has desarrollado tu Hook con Java, usa Maven para empaquetarlo.

En el directorio de tu proyecto Hook, ejecuta el siguiente comando para compilar tu Hook, ejecuta pruebas unitarias y empaqueta tu proyecto como un `JAR` archivo que puedas usar para enviar tu Hook al CloudFormation registro.

```
mvn clean package
```

Registra un Hook personalizado

Para registrar un Hook

1. (Opcional) Configure su Región de AWS nombre predeterminado para`us-west-2`, enviando el configureoperación.

   ```
   $ aws configure
   AWS Access Key ID [None]: <Your Access Key ID>
   AWS Secret Access Key [None]: <Your Secret Key>
   Default region name [None]: us-west-2
   Default output format [None]: json
   ```

2. (Opcional) El siguiente comando compila y empaqueta tu proyecto Hook sin registrarlo.

   ```
   $ cfn submit --dry-run
   ```

3. Registra tu Hook usando el CloudFormation CLI submitoperación.

   ```
   $ cfn submit --set-default
   ```

   El comando devuelve el siguiente comando:

```
{‘ProgressStatus’: ‘COMPLETE’}
```

Resultados: Has registrado correctamente tu Hook.

Verificación de que los Hooks estén accesibles en tu cuenta

Comprueba que tu Hook esté disponible en tu región Cuenta de AWS y en las regiones a las que lo has enviado.

1. Para verificar tu Hook, usa el list-typescomando para enumerar tu Hook recién registrado y devolver una descripción resumida del mismo.

   ```
   $ aws cloudformation list-types
   ```

   El comando devuelve el siguiente resultado y también te mostrará los Hooks disponibles públicamente que puedes activar en tu región Cuenta de AWS y en tu región.

   ```
   {
       "TypeSummaries": [
           {
               "Type": "HOOK",
               "TypeName": "MyCompany::Testing::MyTestHook",
               "DefaultVersionId": "00000001",
               "TypeArn": "arn:aws:cloudformation:us-west-2:ACCOUNT_ID/type/hook/
   MyCompany-Testing-MyTestHook",
               "LastUpdated": "2021-08-04T23:00:03.058000+00:00",
               "Description": "Verifies S3 bucket and SQS queues properties before
    creating or updating"
           }
       ]
   }
   ```

2. Recupera el `TypeArn` archivo de `list-type` salida de tu Hook y guárdalo.

   ```
   export HOOK_TYPE_ARN=arn:aws:cloudformation:us-west-2:ACCOUNT_ID/type/hook/
   MyCompany-Testing-MyTestHook
   ```

Para obtener información sobre cómo publicar Hooks para uso público, consulte Publicar ganchos para uso público.

Configurar Hooks

Una vez que hayas desarrollado y registrado tu Hook, puedes configurarlo en tu Hook Cuenta de AWS publicándolo en el registro.

- Para configurar un Hook en tu cuenta, usa el SetTypeConfiguration operación. Esta operación habilita las propiedades del enlace que se definen en la sección `properties` del esquema del enlace. En el siguiente ejemplo, la `minBuckets` propiedad se establece `1` en la configuración.

  > **Note**
  >
  > Al habilitar Hooks en tu cuenta, autorizas a un Hook a usar los permisos definidos por ti Cuenta de AWS. CloudFormation elimina los permisos no necesarios antes de pasarlos al Hook. CloudFormation recomienda a los clientes o usuarios de Hook que revisen los permisos de Hook y sepan qué permisos tienen los Hooks antes de habilitar los Hooks en su cuenta.

  Especifica los datos de configuración de tu extensión Hook registrada en la misma cuenta y Región de AWS.

```
$ aws cloudformation set-type-configuration --region us-west-2
  --configuration '{"CloudFormationConfiguration":{"HookConfiguration":
{"HookInvocationStatus":"ENABLED","FailureMode":"FAIL","Properties":{"minBuckets":
 "1","minQueues": "1", "encryptionAlgorithm": "aws:kms"}}}}'
  --type-arn $HOOK_TYPE_ARN
```

  > **Important**
  >
  > Para que tu Hook pueda inspeccionar de forma proactiva la configuración de tu pila, debes `HookInvocationStatus` establecer el valor `ENABLED` en la `HookConfiguration` sección una vez que el Hook se haya registrado y activado en tu cuenta.

Acceder a AWS APIs los controladores

Si tus Hooks utilizan un AWS API en alguno de sus controladores, el CFN - crea CLI automáticamente una plantilla de roles de IAM ejecución,. `hook-role.yaml` La `hook-role.yaml` plantilla se basa en los permisos especificados para cada controlador en la sección correspondiente al controlador del esquema de Hook. Si la `--role-arn` bandera no se utiliza durante el generateoperación, la función de esta pila se aprovisionará y se utilizará como función de ejecución del Hook.

Para obtener más información, consulta Acceder AWS APIs desde un tipo de recurso.

plantilla hook-role.yaml

> **Note**
>
> Si opta por crear su propio rol de ejecución, le recomendamos encarecidamente que practique el principio del privilegio mínimo al permitir publicar solo y. `hooks.cloudformation.amazonaws.com` `resources.cloudformation.amazonaws.com`

La siguiente plantilla utiliza IAM los SQS permisos Amazon S3 y Amazon.

```
AWSTemplateFormatVersion: 2010-09-09
Description: >
  This CloudFormation template creates a role assumed by CloudFormation during
  Hook operations on behalf of the customer.
Resources:
  ExecutionRole:
    Type: 'AWS::IAM::Role'
    Properties:
      MaxSessionDuration: 8400
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - resources.cloudformation.amazonaws.com
                - hooks.cloudformation.amazonaws.com
            Action: 'sts:AssumeRole'
```

```
            Condition:
              StringEquals:
                aws:SourceAccount: !Ref AWS::AccountId
              StringLike:
                aws:SourceArn: !Sub arn:${AWS::Partition}:cloudformation:
${AWS::Region}:${AWS::AccountId}:type/hook/MyCompany-Testing-MyTestHook/*
      Path: /
      Policies:
        - PolicyName: HookTypePolicy
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              - Effect: Allow
                Action:
                  - 's3:GetEncryptionConfiguration'
                  - 's3:ListBucket'
                  - 's3:ListAllMyBuckets'
                  - 'sqs:GetQueueAttributes'
                  - 'sqs:GetQueueUrl'
                  - 'sqs:ListQueues'
                Resource: '*'
Outputs:
  ExecutionRoleArn:
    Value: !GetAtt
      - ExecutionRole
      - Arn
```

Probando un Hook personalizado en tu Cuenta de AWS

Ahora que has codificado las funciones de tu controlador que corresponden a un punto de invocación, es hora de probar tu Hook personalizado en una CloudFormation pila.

El modo de fallo de Hook se establece `FAIL` si la CloudFormation plantilla no aprovisionó un bucket de S3 con lo siguiente:

- Se ha establecido el cifrado del bucket de Amazon S3.
- La clave de bucket de Amazon S3 está habilitada para el bucket.
- El algoritmo de cifrado establecido para el bucket de Amazon S3 es el algoritmo correcto que se requiere.
- El ID de AWS Key Management Service clave está establecido.

En el siguiente ejemplo, cree una plantilla llamada `my-failed-bucket-stack.yml` con un nombre de pila `my-hook-stack` que no supere la configuración de la pila y se detenga antes de aprovisionar los recursos.

Probar Hooks mediante el aprovisionamiento de una pila

Ejemplo 1: Para aprovisionar una pila

Aprovisione una pila que no cumpla con las normas

1. Cree una plantilla que especifique un bucket de S3. Por ejemplo, `my-failed-bucket-stack.yml`.

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  S3Bucket:
    Type: 'AWS::S3::Bucket'
    Properties: {}
```

2. Cree una pila y especifique la plantilla en AWS Command Line Interface (AWS CLI). En el siguiente ejemplo, especifique el nombre de la pila como `my-hook-stack` y el nombre de la plantilla como`my-failed-bucket-stack.yml`.

```
$ aws cloudformation create-stack \
  --stack-name my-hook-stack \
  --template-body file://my-failed-bucket-stack.yml
```

3. (Opcional) Para ver el progreso de la pila, especifique el nombre de la pila. En el siguiente ejemplo, especifique el nombre de la pila`my-hook-stack`.

```
$ aws cloudformation describe-stack-events \
  --stack-name my-hook-stack
```

   Utilice la `describe-stack-events` operación para ver el error de Hook al crear el depósito. A continuación se muestra un ejemplo del resultado del comando.

```
{
    "StackEvents": [
    ...
        {
```

```
            "StackId": "arn:aws:cloudformation:us-west-2:ACCOUNT_ID:stack/my-hook-
stack/2c693970-f57e-11eb-a0fb-061a2a83f0b9",
            "EventId": "S3Bucket-CREATE_FAILED-2021-08-04T23:47:03.305Z",
            "StackName": "my-hook-stack",
            "LogicalResourceId": "S3Bucket",
            "PhysicalResourceId": "",
            "ResourceType": "AWS::S3::Bucket",
            "Timestamp": "2021-08-04T23:47:03.305000+00:00",
            "ResourceStatus": "CREATE_FAILED",
            "ResourceStatusReason": "The following hook(s) failed:
 [MyCompany::Testing::MyTestHook]",
            "ResourceProperties": "{}",
            "ClientRequestToken": "Console-CreateStack-abe71ac2-ade4-
a762-0499-8d34d91d6a92"
        },
    ...
    ]
}
```

Resultados: la invocación de Hook falló en la configuración de la pila e impidió el aprovisionamiento del recurso.

Usa una CloudFormation plantilla para pasar la validación de Hook

1. Para crear una pila y pasar la validación de Hook, actualiza la plantilla para que tu recurso utilice un bucket de S3 cifrado. En este ejemplo, se utiliza la plantilla `my-encrypted-bucket-stack.yml`.

```
AWSTemplateFormatVersion: 2010-09-09
Description: |
  This CloudFormation template provisions an encrypted S3 Bucket
Resources:
  EncryptedS3Bucket:
    Type: 'AWS::S3::Bucket'
    Properties:
      BucketName: !Sub 'encryptedbucket-${AWS::Region}-${AWS::AccountId}'
      BucketEncryption:
        ServerSideEncryptionConfiguration:
          - ServerSideEncryptionByDefault:
              SSEAlgorithm: 'aws:kms'
              KMSMasterKeyID: !Ref EncryptionKey
            BucketKeyEnabled: true
```

```
  EncryptionKey:
    Type: 'AWS::KMS::Key'
    DeletionPolicy: Retain
    Properties:
      Description: KMS key used to encrypt the resource type artifacts
      EnableKeyRotation: true
      KeyPolicy:
        Version: 2012-10-17
        Statement:
          - Sid: Enable full access for owning account
            Effect: Allow
            Principal:
              AWS: !Ref 'AWS::AccountId'
            Action: 'kms:*'
            Resource: '*'
Outputs:
  EncryptedBucketName:
    Value: !Ref EncryptedS3Bucket
```

> **Note**
>
> Los ganchos no se invocarán para los recursos omitidos.

2. Crea una pila y especifica tu plantilla. En este ejemplo, el nombre de la pila es`my-encrypted-bucket-stack`.

```
$ aws cloudformation create-stack \
  --stack-name my-encrypted-bucket-stack \
  --template-body file://my-encrypted-bucket-stack.yml \
```

3. (Opcional) Para ver el progreso de la pila, especifique el nombre de la pila.

```
$ aws cloudformation describe-stack-events \
  --stack-name my-encrypted-bucket-stack
```

Usa el `describe-stack-events` comando para ver la respuesta. A continuación, se muestra un ejemplo del comando `describe-stack-events`.

```
{
    "StackEvents": [
    ...
```

```
        {
            "StackId": "arn:aws:cloudformation:us-west-2:ACCOUNT_ID:stack/my-
encrypted-bucket-stack/82a97150-f57a-11eb-8eb2-06a6bdcc7779",
            "EventId": "EncryptedS3Bucket-
CREATE_COMPLETE-2021-08-04T23:23:20.973Z",
            "StackName": "my-encrypted-bucket-stack",
            "LogicalResourceId": "EncryptedS3Bucket",
            "PhysicalResourceId": "encryptedbucket-us-west-2-ACCOUNT_ID",
            "ResourceType": "AWS::S3::Bucket",
            "Timestamp": "2021-08-04T23:23:20.973000+00:00",
            "ResourceStatus": "CREATE_COMPLETE",
            "ResourceProperties": "{\"BucketName\":\"encryptedbucket-us-
west-2-071617338693\",\"BucketEncryption\":{\"ServerSideEncryptionConfiguration\":
[{\"BucketKeyEnabled\":\"true\",\"ServerSideEncryptionByDefault\":{\"SSEAlgorithm
\":\"aws:kms\",\"KMSMasterKeyID\":\"ENCRYPTION_KEY_ARN\"}}]}}",
            "ClientRequestToken": "Console-CreateStack-39df35ac-ca00-
b7f6-5661-4e917478d075"
        },
        {
            "StackId": "arn:aws:cloudformation:us-west-2:ACCOUNT_ID:stack/my-
encrypted-bucket-stack/82a97150-f57a-11eb-8eb2-06a6bdcc7779",
            "EventId": "EncryptedS3Bucket-
CREATE_IN_PROGRESS-2021-08-04T23:22:59.410Z",
            "StackName": "my-encrypted-bucket-stack",
            "LogicalResourceId": "EncryptedS3Bucket",
            "PhysicalResourceId": "encryptedbucket-us-west-2-ACCOUNT_ID",
            "ResourceType": "AWS::S3::Bucket",
            "Timestamp": "2021-08-04T23:22:59.410000+00:00",
            "ResourceStatus": "CREATE_IN_PROGRESS",
            "ResourceStatusReason": "Resource creation Initiated",
            "ResourceProperties": "{\"BucketName\":\"encryptedbucket-us-
west-2-071617338693\",\"BucketEncryption\":{\"ServerSideEncryptionConfiguration\":
[{\"BucketKeyEnabled\":\"true\",\"ServerSideEncryptionByDefault\":{\"SSEAlgorithm
\":\"aws:kms\",\"KMSMasterKeyID\":\"ENCRYPTION_KEY_ARN\"}}]}}",
            "ClientRequestToken": "Console-CreateStack-39df35ac-ca00-
b7f6-5661-4e917478d075"
        },
        {
            "StackId": "arn:aws:cloudformation:us-west-2:ACCOUNT_ID:stack/my-
encrypted-bucket-stack/82a97150-f57a-11eb-8eb2-06a6bdcc7779",
            "EventId": "EncryptedS3Bucket-6516081f-c1f2-4bfe-a0f0-cefa28679994",
            "StackName": "my-encrypted-bucket-stack",
            "LogicalResourceId": "EncryptedS3Bucket",
            "PhysicalResourceId": "",
```

```
            "ResourceType": "AWS::S3::Bucket",
            "Timestamp": "2021-08-04T23:22:58.349000+00:00",
            "ResourceStatus": "CREATE_IN_PROGRESS",
            "ResourceStatusReason": "Hook invocations complete.  Resource creation
 initiated",
            "ClientRequestToken": "Console-CreateStack-39df35ac-ca00-
b7f6-5661-4e917478d075"
        },
    ...
    ]
}
```

Resultados: la pila se creó CloudFormation correctamente. La lógica de Hook verificó que el `AWS::S3::Bucket` recurso contenía cifrado del lado del servidor antes de aprovisionar el recurso.

Ejemplo 2: Para aprovisionar una pila

Aprovisione una pila que no cumpla con las normas

1. Cree una plantilla que especifique un bucket de S3. Por ejemplo, `aes256-bucket.yml`.

```
AWSTemplateFormatVersion: 2010-09-09
Description: |
  This CloudFormation template provisions an encrypted S3 Bucket
Resources:
  EncryptedS3Bucket:
    Type: 'AWS::S3::Bucket'
    Properties:
      BucketName: !Sub 'encryptedbucket-${AWS::Region}-${AWS::AccountId}'
      BucketEncryption:
        ServerSideEncryptionConfiguration:
          - ServerSideEncryptionByDefault:
              SSEAlgorithm: AES256
            BucketKeyEnabled: true
Outputs:
  EncryptedBucketName:
    Value: !Ref EncryptedS3Bucket
```

2. Cree una pila y especifique la plantilla en AWS CLI. En el siguiente ejemplo, especifique el nombre de la pila como `my-hook-stack` y el nombre de la plantilla como`aes256-bucket.yml`.

```
$ aws cloudformation create-stack \
  --stack-name my-hook-stack \
  --template-body file://aes256-bucket.yml
```

3. (Opcional) Para ver el progreso de la pila, especifique el nombre de la pila. En el siguiente ejemplo, especifique el nombre de la pila`my-hook-stack`.

```
$ aws cloudformation describe-stack-events \
  --stack-name my-hook-stack
```

Utilice la `describe-stack-events` operación para ver el error de Hook al crear el depósito. A continuación se muestra un ejemplo del resultado del comando.

```
{
    "StackEvents": [
    ...
        {
            "StackId": "arn:aws:cloudformation:us-west-2:ACCOUNT_ID:stack/my-hook-
stack/2c693970-f57e-11eb-a0fb-061a2a83f0b9",
            "EventId": "S3Bucket-CREATE_FAILED-2021-08-04T23:47:03.305Z",
            "StackName": "my-hook-stack",
            "LogicalResourceId": "S3Bucket",
            "PhysicalResourceId": "",
            "ResourceType": "AWS::S3::Bucket",
            "Timestamp": "2021-08-04T23:47:03.305000+00:00",
            "ResourceStatus": "CREATE_FAILED",
            "ResourceStatusReason": "The following hook(s) failed:
 [MyCompany::Testing::MyTestHook]",
            "ResourceProperties": "{}",
            "ClientRequestToken": "Console-CreateStack-abe71ac2-ade4-
a762-0499-8d34d91d6a92"
        },
    ...
    ]
}
```

Resultados: la invocación de Hook falló en la configuración de la pila e impidió el aprovisionamiento del recurso. Se produjo un error en la pila debido a que el cifrado del bucket de S3 estaba mal configurado. La configuración del tipo Hook es necesaria `aws:kms` mientras este bucket esté en uso`AES256`.

Usa una CloudFormation plantilla para pasar la validación de Hook

1. Para crear una pila y pasar la validación de Hook, actualiza la plantilla para que tu recurso utilice un bucket de S3 cifrado. En este ejemplo, se utiliza la plantilla `kms-bucket-and-queue.yml`.

```
AWSTemplateFormatVersion: 2010-09-09
Description: |
  This CloudFormation template provisions an encrypted S3 Bucket
Resources:
  EncryptedS3Bucket:
    Type: 'AWS::S3::Bucket'
    Properties:
      BucketName: !Sub 'encryptedbucket-${AWS::Region}-${AWS::AccountId}'
      BucketEncryption:
        ServerSideEncryptionConfiguration:
          - ServerSideEncryptionByDefault:
              SSEAlgorithm: 'aws:kms'
              KMSMasterKeyID: !Ref EncryptionKey
            BucketKeyEnabled: true
  EncryptedQueue:
    Type: 'AWS::SQS::Queue'
    Properties:
      QueueName: 'encryptedqueue-${AWS::Region}-${AWS::AccountId}'
      KmsMasterKeyId: !Ref EncryptionKey
  EncryptionKey:
    Type: 'AWS::KMS::Key'
    DeletionPolicy: Retain
    Properties:
      Description: KMS key used to encrypt the resource type artifacts
      EnableKeyRotation: true
      KeyPolicy:
        Version: 2012-10-17
        Statement:
          - Sid: Enable full access for owning account
            Effect: Allow
            Principal:
```

```
              AWS: !Ref 'AWS::AccountId'
            Action: 'kms:*'
            Resource: '*'
Outputs:
  EncryptedBucketName:
    Value: !Ref EncryptedS3Bucket
  EncryptedQueueName:
    Value: !Ref EncryptedQueue
```

> **Note**
>
> Los ganchos no se invocarán para los recursos omitidos.

2. Crea una pila y especifica tu plantilla. En este ejemplo, el nombre de la pila es`my-encrypted-bucket-stack`.

```
$ aws cloudformation create-stack \
  --stack-name my-encrypted-bucket-stack \
  --template-body file://kms-bucket-and-queue.yml
```

3. (Opcional) Para ver el progreso de la pila, especifique el nombre de la pila.

```
$ aws cloudformation describe-stack-events \
  --stack-name my-encrypted-bucket-stack
```

Usa el `describe-stack-events` comando para ver la respuesta. A continuación, se muestra un ejemplo del comando `describe-stack-events`.

```
{
    "StackEvents": [
    ...
        {
            "StackId": "arn:aws:cloudformation:us-west-2:ACCOUNT_ID:stack/my-
encrypted-bucket-stack/82a97150-f57a-11eb-8eb2-06a6bdcc7779",
            "EventId": "EncryptedS3Bucket-
CREATE_COMPLETE-2021-08-04T23:23:20.973Z",
            "StackName": "my-encrypted-bucket-stack",
            "LogicalResourceId": "EncryptedS3Bucket",
            "PhysicalResourceId": "encryptedbucket-us-west-2-ACCOUNT_ID",
            "ResourceType": "AWS::S3::Bucket",
            "Timestamp": "2021-08-04T23:23:20.973000+00:00",
```

```
            "ResourceStatus": "CREATE_COMPLETE",
            "ResourceProperties": "{\"BucketName\":\"encryptedbucket-us-
west-2-071617338693\",\"BucketEncryption\":{\"ServerSideEncryptionConfiguration\":
[{\"BucketKeyEnabled\":\"true\",\"ServerSideEncryptionByDefault\":{\"SSEAlgorithm
\":\"aws:kms\",\"KMSMasterKeyID\":\"ENCRYPTION_KEY_ARN\"}}]}}",
            "ClientRequestToken": "Console-CreateStack-39df35ac-ca00-
b7f6-5661-4e917478d075"
        },
        {
            "StackId": "arn:aws:cloudformation:us-west-2:ACCOUNT_ID:stack/my-
encrypted-bucket-stack/82a97150-f57a-11eb-8eb2-06a6bdcc7779",
            "EventId": "EncryptedS3Bucket-
CREATE_IN_PROGRESS-2021-08-04T23:22:59.410Z",
            "StackName": "my-encrypted-bucket-stack",
            "LogicalResourceId": "EncryptedS3Bucket",
            "PhysicalResourceId": "encryptedbucket-us-west-2-ACCOUNT_ID",
            "ResourceType": "AWS::S3::Bucket",
            "Timestamp": "2021-08-04T23:22:59.410000+00:00",
            "ResourceStatus": "CREATE_IN_PROGRESS",
            "ResourceStatusReason": "Resource creation Initiated",
            "ResourceProperties": "{\"BucketName\":\"encryptedbucket-us-
west-2-071617338693\",\"BucketEncryption\":{\"ServerSideEncryptionConfiguration\":
[{\"BucketKeyEnabled\":\"true\",\"ServerSideEncryptionByDefault\":{\"SSEAlgorithm
\":\"aws:kms\",\"KMSMasterKeyID\":\"ENCRYPTION_KEY_ARN\"}}]}}",
            "ClientRequestToken": "Console-CreateStack-39df35ac-ca00-
b7f6-5661-4e917478d075"
        },
        {
            "StackId": "arn:aws:cloudformation:us-west-2:ACCOUNT_ID:stack/my-
encrypted-bucket-stack/82a97150-f57a-11eb-8eb2-06a6bdcc7779",
            "EventId": "EncryptedS3Bucket-6516081f-c1f2-4bfe-a0f0-cefa28679994",
            "StackName": "my-encrypted-bucket-stack",
            "LogicalResourceId": "EncryptedS3Bucket",
            "PhysicalResourceId": "",
            "ResourceType": "AWS::S3::Bucket",
            "Timestamp": "2021-08-04T23:22:58.349000+00:00",
            "ResourceStatus": "CREATE_IN_PROGRESS",
            "ResourceStatusReason": "Hook invocations complete.  Resource creation
 initiated",
            "ClientRequestToken": "Console-CreateStack-39df35ac-ca00-
b7f6-5661-4e917478d075"
        },
    ...
    ]
```

```
}
```

Resultados: la pila se creó CloudFormation correctamente. La lógica de Hook verificó que el `AWS::S3::Bucket` recurso contenía cifrado del lado del servidor antes de aprovisionar el recurso.

Actualización de un Hook personalizado

La actualización de un Hook personalizado permite que las revisiones del Hook estén disponibles en el CloudFormation registro.

Para actualizar un Hook personalizado, envía tus revisiones al CloudFormation registro a través del CloudFormation CLI submitoperación.

```
$ cfn submit
```

Para especificar la versión predeterminada de tu Hook en tu cuenta, usa el set-type-default-versioncomando y especifica el tipo, el nombre del tipo y el ID de la versión.

```
$ aws cloudformation set-type-default-version \
    --type HOOK \
    --type-name MyCompany::Testing::MyTestHook \
    --version-id 00000003
```

Para recuperar información sobre las versiones de un Hook, usa list-type-versions.

```
$ aws cloudformation list-type-versions \
  --type HOOK \
  --type-name "MyCompany::Testing::MyTestHook"
```

Anular el registro de un Hook personalizado del registro CloudFormation

Al anular el registro de un Hook personalizado, se marca la extensión o versión de la extensión como tal `DEPRECATED` en el CloudFormation registro, lo que la elimina del uso activo. Una vez obsoleto, el Hook personalizado no se puede usar en ninguna operación. CloudFormation

> **Note**
>
> Antes de anular el registro del Hook, debes anular el registro de forma individual todas las versiones activas anteriores de esa extensión. Para obtener más información, consulte DeregisterType.

Para anular el registro de un Hook, usa el deregister-typeopera y especifica tu Hook. ARN

```
$ aws cloudformation deregister-type \
    --arn HOOK_TYPE_ARN
```

Este comando no produce ningún resultado.

Publicar ganchos para uso público

Para desarrollar un Hook público de terceros, desarrolla tu Hook como una extensión privada. Luego, en cada una de las extensiones Región de AWS en las que quieras que la extensión esté disponible públicamente:

1. Registra tu Hook como una extensión privada en el CloudFormation registro.
2. Prueba tu Hook para asegurarte de que cumple con todos los requisitos necesarios para ser publicado en el CloudFormation registro.
3. Publica tu Hook en el CloudFormation registro.

> **Note**
>
> Antes de publicar cualquier extensión en una región determinada, primero debes registrarte como editor de extensiones en esa región. Para hacerlo en varias regiones simultáneamente, consulte Publicar extensiones en varias regiones StackSets en la Guía del AWS CloudFormation CLI usuario.

Una vez que hayas desarrollado y registrado tu Hook, podrás ponerlo a disposición del público en CloudFormation general publicándolo en el CloudFormation registro, como una extensión pública de terceros.

Los Hooks públicos de terceros te permiten ofrecer a CloudFormation los usuarios la posibilidad de inspeccionar de forma proactiva la configuración de AWS los recursos antes de aprovisionarlos. Al igual que ocurre con los Hooks privados, los Hooks públicos reciben el mismo trato que cualquier Hook publicado por AWS ellos. CloudFormation

Todos los CloudFormation usuarios pueden ver los Hooks publicados en el registro Regiones de AWS en el que están publicados. A continuación, los usuarios pueden activar la extensión en su cuenta, de modo que esté disponible para su uso en sus plantillas. Para obtener más información, consulte Utilizar extensiones públicas de terceros del CloudFormation registro en la Guía del AWS CloudFormation usuario.

Probando un Hook personalizado para uso público

Para publicar tu Hook personalizado registrado, debe superar todos los requisitos de prueba definidos para él. La siguiente es una lista de requisitos necesarios antes de publicar tu Hook personalizado como una extensión de terceros.

Cada controlador y objetivo se prueban dos veces. Una vez para `SUCCESS` y otra para`FAILED`.

- Para un caso de `SUCCESS` respuesta:
  - El estado debe ser`SUCCESS`.
  - No debe devolver un código de error.
  - El retardo de devolución de llamada debe establecerse en `0` segundos, si se especifica.
- Para el caso `FAILED` de respuesta:
  - El estado debe ser`FAILED`.
  - Debe devolver un código de error.
  - Debe tener un mensaje de respuesta.
  - El retardo de devolución de llamada se debe establecer en `0` segundos, si se especifica.
- Para el caso `IN_PROGRESS` de respuesta:
  - No debe devolver un código de error.
  - `Result`el campo no debe configurarse como respuesta.

Especificación de los datos de entrada para su uso en las pruebas de contrato

De forma predeterminada, CloudFormation realiza pruebas de contrato utilizando las propiedades de entrada generadas a partir de los patrones que definas en tu esquema de Hook. Sin embargo, la

mayoría de los Hooks son lo suficientemente complejos como para que las propiedades de entrada para precrear o actualizar las pilas de aprovisionamiento requieran una comprensión del recurso que se está aprovisionando. Para solucionar este problema, puedes especificar la entrada que CloudFormation utilizan al realizar sus pruebas de contrato.

CloudFormation ofrece dos formas de especificar los datos de entrada para utilizarlos al realizar las pruebas de contrato:

- Anula el archivo

  El uso de un `overrides` archivo proporciona una forma ligera de especificar los datos de entrada para determinadas propiedades específicas CloudFormation para utilizarlos durante `preCreate` las pruebas `preUpdate` y `preDelete` las operaciones.
- Archivos de entrada

  También puede usar varios `input` archivos para especificar los datos de entrada de las pruebas de contrato si:
  - Desea o necesita especificar datos de entrada diferentes para las operaciones de creación, actualización y eliminación, o bien datos no válidos con los que realizar las pruebas.
  - Desea especificar varios conjuntos de datos de entrada diferentes.

Especificar los datos de entrada mediante un archivo de anulación

El siguiente es un ejemplo de los datos de entrada de Amazon S3 Hook utilizando el `overrides` archivo.

```
{
    "CREATE_PRE_PROVISION": {
        "AWS::S3::Bucket": {
            "resourceProperties": {
                "/BucketName": "encryptedbucket-us-west-2-contractor",
                "/BucketEncryption/ServerSideEncryptionConfiguration": [
                    {
                        "BucketKeyEnabled": true,
                        "ServerSideEncryptionByDefault": {
                            "KMSMasterKeyID": "KMS-KEY-ARN",
                            "SSEAlgorithm": "aws:kms"
                        }
                    }
                ]
```

```
            }
        },
        "AWS::SQS::Queue": {
            "resourceProperties": {
                "/QueueName": "MyQueueContract",
                "/KmsMasterKeyId": "hellocontract"
            }
        }
    },
    "UPDATE_PRE_PROVISION": {
        "AWS::S3::Bucket": {
            "resourceProperties": {
                "/BucketName": "encryptedbucket-us-west-2-contractor",
                "/BucketEncryption/ServerSideEncryptionConfiguration": [
                    {
                        "BucketKeyEnabled": true,
                        "ServerSideEncryptionByDefault": {
                            "KMSMasterKeyID": "KMS-KEY-ARN",
                            "SSEAlgorithm": "aws:kms"
                        }
                    }
                ]
            },
            "previousResourceProperties": {
                "/BucketName": "encryptedbucket-us-west-2-contractor",
                "/BucketEncryption/ServerSideEncryptionConfiguration": [
                    {
                        "BucketKeyEnabled": true,
                        "ServerSideEncryptionByDefault": {
                            "KMSMasterKeyID": "KMS-KEY-ARN",
                            "SSEAlgorithm": "aws:kms"
                        }
                    }
                ]
            }
        }
    },
    "INVALID_UPDATE_PRE_PROVISION": {
        "AWS::S3::Bucket": {
            "resourceProperties": {
                "/BucketName": "encryptedbucket-us-west-2-contractor",
                "/BucketEncryption/ServerSideEncryptionConfiguration": [
                    {
                        "BucketKeyEnabled": true,
```

```
                        "ServerSideEncryptionByDefault": {
                            "KMSMasterKeyID": "KMS-KEY-ARN",
                            "SSEAlgorithm": "AES256"
                        }
                    }
                ]
            },
            "previousResourceProperties": {
                "/BucketName": "encryptedbucket-us-west-2-contractor",
                "/BucketEncryption/ServerSideEncryptionConfiguration": [
                    {
                        "BucketKeyEnabled": true,
                        "ServerSideEncryptionByDefault": {
                            "KMSMasterKeyID": "KMS-KEY-ARN",
                            "SSEAlgorithm": "aws:kms"
                        }
                    }
                ]
            }
        }
    },
    "INVALID": {
        "AWS::SQS::Queue": {
            "resourceProperties": {
                "/QueueName": "MyQueueContract",
                "/KmsMasterKeyId": "KMS-KEY-ARN"
            }
        }
    }
}
```

Especificar los datos de entrada mediante archivos de entrada

Utilice `input` los archivos para especificar diferentes tipos de datos de entrada CloudFormation para su uso: `preCreate` entrada, `preUpdate` entrada y entrada no válida. Cada tipo de datos se especifica en un archivo independiente. También puede especificar varios conjuntos de datos de entrada para las pruebas de contrato.

Para especificar `input` los archivos CloudFormation que se van a utilizar en las pruebas por contrato, añade una `inputs` carpeta al directorio raíz de tu proyecto de Hooks. Luego agrega tus archivos de entrada.

Especifique el tipo de datos de entrada que contiene un archivo mediante las siguientes convenciones de nomenclatura, donde *n*es un número entero:

- `inputs_`*n*`_pre_create.json`: utilice archivos con `preCreate` controladores para especificar las entradas a fin de crear el recurso.
- `inputs_`*n*`_pre_update.json`: Utilice archivos con `preUpdate` controladores para especificar las entradas a fin de actualizar el recurso.
- `inputs_`*n*`_pre_delete.json`: Utilice archivos con `preDelete` controladores para especificar las entradas a fin de eliminar el recurso.
- `inputs_`*n*`_invalid.json`: Para especificar entradas no válidas para probarlas.

Para especificar varios conjuntos de datos de entrada para las pruebas de contrato, incremente el número entero en los nombres de los archivos para ordenar los conjuntos de datos de entrada. Por ejemplo, el primer conjunto de archivos de entrada debe tener el nombre `inputs_1_pre_create.jsoninputs_1_pre_update.json`, `yinputs_1_pre_invalid.json`. El siguiente conjunto se llamaría `inputs_2_pre_create.jsoninputs_2_pre_update.json`, `yinputs_2_pre_invalid.json`, y así sucesivamente.

Cada archivo de entrada es un JSON archivo que contiene solo las propiedades del recurso que se utilizarán en las pruebas.

El siguiente es un ejemplo de directorio `inputs` para Amazon S3 especificar los datos de entrada mediante archivos de entrada.

`inputs_1_pre_create.json`

El siguiente es un ejemplo de la prueba de `inputs_1_pre_create.json` contrato.

```
{
    "AWS::S3::Bucket": {
        "resourceProperties": {
            "AccessControl": "BucketOwnerFullControl",
            "AnalyticsConfigurations": [],
            "BucketEncryption": {
                "ServerSideEncryptionConfiguration": [
                    {
                        "BucketKeyEnabled": true,
                        "ServerSideEncryptionByDefault": {
```

```
                            "KMSMasterKeyID": "KMS-KEY-ARN",
                            "SSEAlgorithm": "aws:kms"
                        }
                    }
                ]
            },
            "BucketName": "encryptedbucket-us-west-2"
        }
    },
    "AWS::SQS::Queue": {
        "resourceProperties": {
            "QueueName": "MyQueue",
            "KmsMasterKeyId": "KMS-KEY-ARN"
        }
    }
}
```

`inputs_1_pre_update.json`

El siguiente es un ejemplo de la prueba de `inputs_1_pre_update.json` contrato.

```
{
    "AWS::S3::Bucket": {
        "resourceProperties": {
            "BucketEncryption": {
                "ServerSideEncryptionConfiguration": [
                    {
                        "BucketKeyEnabled": true,
                        "ServerSideEncryptionByDefault": {
                            "KMSMasterKeyID": "KMS-KEY-ARN",
                            "SSEAlgorithm": "aws:kms"
                        }
                    }
                ]
            },
            "BucketName": "encryptedbucket-us-west-2"
        },
        "previousResourceProperties": {
            "BucketEncryption": {
                "ServerSideEncryptionConfiguration": [
                    {
                        "BucketKeyEnabled": true,
                        "ServerSideEncryptionByDefault": {
                            "KMSMasterKeyID": "KMS-KEY-ARN",
```

```
                            "SSEAlgorithm": "aws:kms"
                        }
                    }
                ]
            },
            "BucketName": "encryptedbucket-us-west-2"
        }
    }
}
```

`inputs_1_invalid.json`

El siguiente es un ejemplo de la prueba de `inputs_1_invalid.json` contrato.

```
{
    "AWS::S3::Bucket": {
        "resourceProperties": {
            "AccessControl": "BucketOwnerFullControl",
            "AnalyticsConfigurations": [],
            "BucketEncryption": {
                "ServerSideEncryptionConfiguration": [
                    {
                        "ServerSideEncryptionByDefault": {
                            "SSEAlgorithm": "AES256"
                        }
                    }
                ]
            },
            "BucketName": "encryptedbucket-us-west-2"
        }
    },
    "AWS::SQS::Queue": {
        "resourceProperties": {
            "NotValid": "The property of this resource is not valid."
        }
    }
}
```

`inputs_1_invalid_pre_update.json`

El siguiente es un ejemplo de la prueba de `inputs_1_invalid_pre_update.json` contrato.

```
{
```

```
    "AWS::S3::Bucket": {
        "resourceProperties": {
            "BucketEncryption": {
                "ServerSideEncryptionConfiguration": [
                    {
                        "BucketKeyEnabled": true,
                        "ServerSideEncryptionByDefault": {
                            "KMSMasterKeyID": "KMS-KEY-ARN",
                            "SSEAlgorithm": "AES256"
                        }
                    }
                ]
            },
            "BucketName": "encryptedbucket-us-west-2"
        },
        "previousResourceProperties": {
            "BucketEncryption": {
                "ServerSideEncryptionConfiguration": [
                    {
                        "BucketKeyEnabled": true,
                        "ServerSideEncryptionByDefault": {
                            "KMSMasterKeyID": "KMS-KEY-ARN",
                            "SSEAlgorithm": "aws:kms"
                        }
                    }
                ]
            },
            "BucketName": "encryptedbucket-us-west-2"
        }
    }
}
```

Para obtener más información, consulte Publicar extensiones para que estén disponibles para el uso público en la Guía del AWS CloudFormation CLI usuario.

Referencia de sintaxis del esquema para AWS CloudFormation Hooks

En esta sección se describe la sintaxis del esquema que utilizas para desarrollar AWS CloudFormation Hooks.

Un Hook incluye una especificación de Hook representada por un esquema JSON y controladores de Hook. El primer paso para crear un Hook personalizado es modelar un esquema que defina el

Hook, sus propiedades y sus atributos. Al inicializar un proyecto Hook personalizado mediante el initcomando CloudFormation CLI, se crea un archivo de esquema Hook para usted. Usa este archivo de esquema como punto de partida para definir la forma y la semántica de tu Hook personalizado.

Sintaxis del esquema

El siguiente esquema es la estructura de un Hook.

```
{
"typeName": "string",
    "description": "string",
    "sourceUrl": "string",
    "documentationUrl": "string",
    "definitions": {
        "definitionName": {
          . . .
        }
    },
    "typeConfiguration": {
        "properties": {
             "propertyName": {
                "description": "string",
                "type": "string",
                 . . .
            },
        },
    "required": [
        "propertyName"
         . . .
            ],
    "additionalProperties": false
    },
    "handlers": {
        "preCreate": {
            "targetNames": [
            ],
            "permissions": [
            ]
        },
        "preUpdate": {
            "targetNames": [
            ],
            "permissions": [
```

```
            ]
        },
        "preDelete": {
            "targetNames": [
            ],
            "permissions": [
            ]
        }
    },
    "additionalProperties": false
}
```

`typeName`

El nombre exclusivo de tu Hook. Especifica un espacio de nombres de tres partes para tu Hook, con un patrón recomendado de. `Organization::Service::Hook`

> **Note**
>
> Los siguientes espacios de nombres de organización están reservados y no se pueden usar en los nombres de tipo Hook:
>
> - `Alexa`
> - `AMZN`
> - `Amazon`
> - `ASK`
> - `AWS`
> - `Custom`
> - `Dev`

Obligatorio: sí

Patrón: `^[a-zA-Z0-9]{2,64}::[a-zA-Z0-9]{2,64}::[a-zA-Z0-9]{2,64}$`

Mínimo: `10`

Máximo: `196`

`description`

Una breve descripción del Hook que se muestra en la CloudFormation consola.

Obligatorio: sí

`sourceUrl`

La URL del código fuente del Hook, si es pública.

Obligatorio: no

Máximo: `4096`

`documentationUrl`

La URL de una página que proporciona documentación detallada del Hook.

Obligatorio: sí

Patrón: `^https\:\/\/[0-9a-zA-Z]([-.\w]*[0-9a-zA-Z])(\:[0-9]*)*([\?/#].*)?$`

Máximo: `4096`

> **Note**
>
> Si bien el esquema de Hook debe incluir descripciones de propiedades completas y precisas, puedes usar la `documentationURL` propiedad para proporcionar a los usuarios más detalles, incluidos ejemplos, casos de uso y otra información detallada.

`definitions`

Usa el `definitions` bloque para proporcionar esquemas de propiedades de Hook compartidos.

Se considera una buena práctica utilizar la `definitions` sección para definir elementos de esquema que se puedan utilizar en varios puntos del esquema de tipo Hook. A continuación, puedes usar un puntero JSON para hacer referencia a ese elemento en los lugares correspondientes de tu esquema de tipo Hook.

Obligatorio: no

`typeConfiguration`

La definición de los datos de configuración de un Hook.

Obligatorio: sí

`properties`

Las propiedades del Hook. Todas las propiedades de un Hook deben expresarse en el esquema. Alinee las propiedades del esquema Hook con las propiedades de configuración del tipo Hook.

> **Note**
>
> No se permiten propiedades anidadas. En su lugar, defina las propiedades anidadas en el `definitions` elemento y utilice un `$ref` puntero para hacer referencia a ellas en la propiedad deseada.

Actualmente se admiten las siguientes propiedades:

- `default`— El valor por defecto de la propiedad.
- `description`— Una descripción de la propiedad.
- `pattern`— Un patrón de expresiones regulares que se utiliza para validar la entrada.
- `type`— El tipo de propiedad aceptado.

`additionalProperties`

`additionalProperties` se debe establecer en `false`. Todas las propiedades de un Hook deben expresarse en el esquema: no se permiten entradas arbitrarias.

Obligatorio: sí

Valores válidos: `false`

`handlers`

Los controladores especifican las operaciones que pueden iniciar el Hook definido en el esquema, como los puntos de invocación del Hook. Por ejemplo, se invoca un `preUpdate` controlador antes de las operaciones de actualización para todos los objetivos especificados en el controlador.

Valores válidos: `preCreate` | `preUpdate` | `preDelete`

> **Note**
>
> Debe especificarse al menos un valor para el controlador.

> **Important**
>
> Apila las operaciones que dan como resultado el estado de `UpdateCleanup` no invocan un Hook. Por ejemplo, en los dos escenarios siguientes, no se invoca el `preDelete` controlador del Hook:
>
> - la pila se actualiza después de eliminar un recurso de la plantilla.
> - se elimina un recurso con el tipo de actualización de reemplazo.

`targetNames`

Un conjunto de cadenas de nombres de tipos a los que apunta Hook. Por ejemplo, si un `preCreate` controlador tiene un `AWS::S3::Bucket` objetivo, el Hook se ejecuta para los buckets de Amazon S3 durante la fase de preaprovisionamiento.

- `TargetName`

  Especifique al menos un nombre de destino para cada controlador implementado.

  Patrón: `^[a-zA-Z0-9]{2,64}::[a-zA-Z0-9]{2,64}::[a-zA-Z0-9]{2,64}$`

  Mínimo: 1

  Obligatorio: sí

  > **Warning**
  >
  > Las referencias dinámicas de SSM SecureString y Secrets Manager no se resuelven antes de pasarlas a Hooks.

`permissions`

Una matriz de cadenas que especifica los AWS permisos necesarios para invocar el controlador.

Obligatorio: sí

`additionalProperties`

`additionalProperties` se debe establecer en `false`. Todas las propiedades de un Hook deben expresarse en el esquema: no se permiten entradas arbitrarias.

Obligatorio: sí

Valores válidos: `false`

Ejemplos de esquemas de Hooks

Ejemplo 1

Los tutoriales de Java y Python utilizan el siguiente ejemplo de código. El siguiente es un ejemplo de estructura para un Hook llamado. `mycompany-testing-mytesthook.json`

```
{
    "typeName":"MyCompany::Testing::MyTestHook",
    "description":"Verifies S3 bucket and SQS queues properties before create and
 update",
    "sourceUrl":"https://mycorp.com/my-repo.git",
    "documentationUrl":"https://mycorp.com/documentation",
    "typeConfiguration":{
        "properties":{
            "minBuckets":{
                "description":"Minimum number of compliant buckets",
                "type":"string"
            },
            "minQueues":{
                "description":"Minimum number of compliant queues",
                "type":"string"
            },
            "encryptionAlgorithm":{
                "description":"Encryption algorithm for SSE",
                "default":"AES256",
                "type":"string",
                "pattern": "[a-zA-Z]*[1-9]"
            }
        },
        "required":[

        ],
        "additionalProperties":false
    },
    "handlers":{
        "preCreate":{
            "targetNames":[
                "AWS::S3::Bucket",
                "AWS::SQS::Queue"
            ],
```

```
            "permissions":[

            ]
        },
        "preUpdate":{
            "targetNames":[
                "AWS::S3::Bucket",
                "AWS::SQS::Queue"
            ],
            "permissions":[

            ]
        },
        "preDelete":{
            "targetNames":[
                "AWS::S3::Bucket",
                "AWS::SQS::Queue"
            ],
            "permissions":[
                "s3:ListBucket",
                "s3:ListAllMyBuckets",
                "s3:GetEncryptionConfiguration",
                "sqs:ListQueues",
                "sqs:GetQueueAttributes",
                "sqs:GetQueueUrl"
            ]
        }
    },
    "additionalProperties":false
}
```

Ejemplo 2

El siguiente ejemplo es un esquema que usa `STACK` and `CHANGE_SET` for para apuntar `targetNames` a una plantilla de pila y a una operación de conjunto de cambios.

```
{
    "typeName":"MyCompany::Testing::MyTestHook",
    "description":"Verifies Stack and Change Set properties before create and update",
    "sourceUrl":"https://mycorp.com/my-repo.git",
    "documentationUrl":"https://mycorp.com/documentation",
    "typeConfiguration":{
        "properties":{
```

```
            "minBuckets":{
                "description":"Minimum number of compliant buckets",
                "type":"string"
            },
            "minQueues":{
                "description":"Minimum number of compliant queues",
                "type":"string"
            },
            "encryptionAlgorithm":{
                "description":"Encryption algorithm for SSE",
                "default":"AES256",
                "type":"string",
                "pattern": "[a-zA-Z]*[1-9]"
            }
        },
        "required":[
        ],
        "additionalProperties":false
    },
    "handlers":{
        "preCreate":{
            "targetNames":[
                "STACK",
                "CHANGE_SET"
            ],
            "permissions":[
            ]
        },
        "preUpdate":{
            "targetNames":[
                "STACK"
            ],
            "permissions":[
            ]
        },
        "preDelete":{
            "targetNames":[
                "STACK"
            ],
            "permissions":[

            ]
        }
    },
```

```
    "additionalProperties":false
}
```

Desactivar y activar AWS CloudFormation Hooks

En este tema se describe cómo deshabilitar y volver a activar un Hook para evitar temporalmente que esté activo en tu cuenta. Desactivar los Hooks puede resultar útil cuando necesitas investigar un problema sin que los Hooks interfieran.

Desactiva y activa un Hook en tu cuenta (consola)

Para deshabilitar un Hook en tu cuenta

1. Inicia sesión AWS Management Console y abre la AWS CloudFormation consola en https://console.aws.amazon.com/cloudformation.
2. En la barra de navegación de la parte superior de la pantalla, selecciona la Región de AWS ubicación del Hook.
3. En el panel de navegación, selecciona Hooks.
4. Elige el nombre del Hook que deseas deshabilitar.
5. En la página de detalles del Hook, a la derecha del nombre del Hook, pulsa el botón Desactivar.
6. Cuando se te pida confirmación, selecciona Desactivar gancho.

Para volver a activar un Hook previamente desactivado

1. Inicia sesión en AWS Management Console https://console.aws.amazon.com/cloudformation y abre la AWS CloudFormation consola.
2. En la barra de navegación de la parte superior de la pantalla, selecciona la Región de AWS ubicación del Hook.
3. En el panel de navegación, selecciona Hooks.
4. Elige el nombre del Hook que quieres activar.
5. En la página de detalles del Hook, a la derecha del nombre del Hook, selecciona el botón Activar.
6. Cuando se te pida confirmación, selecciona Activar Hook.

Desactiva y activa un Hook en tu cuenta (AWS CLI)

> **Important**
>
> Los AWS CLI comandos para deshabilitar y habilitar Hooks sustituyen toda la configuración de Hook por los valores especificados en la `--configuration` opción. Para evitar cambios imprevistos, debe incluir todos los ajustes existentes que desee conservar al ejecutar estos comandos. Para ver los datos de configuración actuales, utilice la describe-typecomando

Para deshabilitar un Hook

Usa lo siguiente set-type-configurationcomando y especifique `HookInvocationStatus` cómo `DISABLED` deshabilitar el Hook. Sustituya los marcadores de posición por sus valores específicos.

```
aws cloudformation set-type-configuration \
  --configuration "{"CloudFormationConfiguration":{"HookConfiguration":
{"HookInvocationStatus": "DISABLED", "FailureMode": "FAIL",
 "TargetOperations": ["STACK","RESOURCE","CHANGE_SET"], "Properties":{}}}}" \
  --type-arn "arn:aws:cloudformation:us-west-2:123456789012:type/hook/MyTestHook" \
  --region us-west-2
```

Para volver a activar un Hook previamente desactivado

Usa lo siguiente set-type-configurationcomando y especifique cómo `HookInvocationStatus` volver `ENABLED` a habilitar el Hook. Sustituya los marcadores de posición por sus valores específicos.

```
aws cloudformation set-type-configuration \
  --configuration "{"CloudFormationConfiguration":{"HookConfiguration":
{"HookInvocationStatus": "ENABLED", "FailureMode": "FAIL",
 "TargetOperations": ["STACK","RESOURCE","CHANGE_SET"], "Properties":{}}}}" \
  --type-arn "arn:aws:cloudformation:us-west-2:123456789012:type/hook/MyTestHook" \
  --region us-west-2
```

Para obtener más información, consulte Referencia a la sintaxis del esquema de la configuración del enlace.

Referencia a la sintaxis del esquema de la configuración del enlace

En esta sección se describe la sintaxis del esquema utilizada para configurar los Hooks. CloudFormation utiliza este esquema de configuración en tiempo de ejecución al invocar un Hook en un Cuenta de AWS.

Para que tu Hook pueda inspeccionar de forma proactiva la configuración de tu pila, configúrala una `ENABLED` vez `HookInvocationStatus` que el Hook se haya registrado y activado en tu cuenta.

Temas

- Propiedades del esquema de configuración de ganchos
- Ejemplos de configuración de Hook
- AWS CloudFormation Filtros de nivel de pila Hooks
- AWS CloudFormation Filtros de destino Hooks
- Uso de caracteres comodín con nombres de objetivos de Hook

> **Note**
>
> La cantidad máxima de datos que puede almacenar la configuración de un Hook es de 300 KB. Esto se suma a todas las restricciones impuestas al parámetro de `Configuration` solicitud de SetTypeConfigurationoperación.

Propiedades del esquema de configuración de ganchos

El siguiente esquema es la estructura de un esquema de configuración de Hook.

```
{
    "CloudFormationConfiguration": {
        "HookConfiguration": {
            "HookInvocationStatus": "ENABLED",
            "TargetOperations": ["STACK"],
            "FailureMode": "FAIL",
            "Properties": {
                ...
```

```
            }
        }
    }
}
```

`HookConfiguration`

La configuración de los Hooks permite activar o desactivar los Hooks a nivel de pila, los modos de error y los valores de las propiedades del Hook.

La configuración de Hook admite las siguientes propiedades.

`HookInvocationStatus`

Especifica si el Hook es `ENABLED` o`DISABLED`.

Valores válidos: `ENABLED` | `DISABLED`

`TargetOperations`

Especifica la lista de operaciones con las que se ejecuta el Hook. Para obtener más información, consulte ¡Gancho objetivos!.

Valores válidos: `STACK` | `RESOURCE` | `CHANGE_SET` | `CLOUD_CONTROL`

`TargetStacks`

Disponible por motivos de compatibilidad con versiones anteriores. Úselo *`HookInvocationStatus`* en su lugar.

Si el modo está configurado en`ALL`, el Hook se aplica a todas las pilas de tu cuenta durante una operación `CREATEUPDATE`, o con un `DELETE` recurso.

Si el modo está configurado en`NONE`, el Hook no se aplicará a las acumulaciones de tu cuenta.

Valores válidos: `ALL` | `NONE`

`FailureMode`

Este campo indica al servicio cómo tratar los errores de Hook.

- Si el modo está establecido en y el Hook falla, la configuración errónea detiene el aprovisionamiento de recursos y revierte la pila. `FAIL`

- Si el modo está establecido en `WARN` y el Hook falla, la configuración de advertencia permite que el aprovisionamiento continúe con un mensaje de advertencia.

Valores válidos: `FAIL` | `WARN`

`Properties`

Especifica las propiedades de ejecución de Hook. Deben coincidir con la forma de las propiedades admitidas por el esquema de Hooks.

Ejemplos de configuración de Hook

Para ver ejemplos de cómo configurar Hooks desde AWS CLI, consulte las siguientes secciones:

- Activa un gancho de protección (AWS CLI)
- Activar un gancho Lambda ()AWS CLI

AWS CloudFormation Filtros de nivel de pila Hooks

Puedes añadir filtros a nivel de pila a tus CloudFormation Hooks para segmentar pilas específicas en función de los nombres y roles de las pilas. Esto es útil en los casos en los que tienes varias pilas con los mismos tipos de recursos, pero el Hook está pensado para pilas específicas.

En esta sección se explica cómo funcionan estos filtros y se proporcionan ejemplos que puede seguir.

La estructura básica de una configuración de Hook sin filtrado a nivel de pila es la siguiente:

```
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "TargetFilters": {
        "Actions": [
          "CREATE",
```

```
            "UPDATE",
            "DELETE"
          ]
        }
      }
    }
}
```

Para obtener más información sobre la `HookConfiguration` sintaxis, consulte[Referencia a la sintaxis del esquema de la configuración del enlace](#).

Para usar filtros a nivel de pila, añada una `StackFilters` clave debajo`HookConfiguration`.

La `StackFilters` clave tiene un miembro obligatorio y dos miembros opcionales.

- `FilteringCriteria` (obligatorio)
- `StackNames` (opcional)
- `StackRoles` (opcional)

Las `StackRoles` propiedades `StackNames` o son opcionales. Sin embargo, debe especificar al menos una de estas propiedades.

Si creas un Hook dirigido a las operaciones de la API de Cloud Control, se ignorarán todos los filtros a nivel de pila.

`FilteringCriteria`

`FilteringCriteria`es un parámetro obligatorio que especifica el comportamiento de filtrado. Se puede establecer en `ALL` o`ANY`.

- `ALL`invoca el Hook si todos los filtros coinciden.
- `ANY`invoca el Hook si algún filtro coincide.

`StackNames`

Para especificar uno o más nombres de pila como filtros en tu configuración de Hooks, usa la siguiente estructura JSON:

```
"StackNames": {
```

```
  "Include": [
    "string"
  ],
  "Exclude": [
    "string"
  ]
}
```

Debe especificar uno de los siguientes:

- `Include`: Lista de nombres de pila que se van a incluir. Solo las pilas especificadas en esta lista invocarán el Hook.
  - Tipo: matriz de cadenas
  - Número máximo de artículos: 50
  - Artículos mínimos: 1
- `Exclude`: Lista de nombres de pilas que se van a excluir. Todas las pilas, excepto las que se muestran aquí, invocarán el Hook.
  - Tipo: matriz de cadenas
  - Número máximo de artículos: 50
  - Artículos mínimos: 1

El nombre de cada pila de las `Exclude` matrices `Include` y debe cumplir los siguientes requisitos de patrón y longitud:

- Patrón: `^[a-zA-Z][-a-zA-Z0-9]*$`
- Longitud máxima: 128

`StackNames`admiten nombres de pila concretos y la combinación completa de comodines. Para ver ejemplos de uso de caracteres comodín, consulte. Uso de caracteres comodín con nombres de objetivos de Hook

`StackRoles`

Para especificar una o más funciones de IAM como filtros en la configuración de Hook, utilice la siguiente estructura JSON:

```
"StackRoles": {
```

```
  "Include": [
    "string"
  ],
  "Exclude": [
    "string"
  ]
}
```

Debe especificar uno de los siguientes:

- `Include`: Lista de funciones de IAM ARNs para segmentar las pilas asociadas a estas funciones. Solo las operaciones de apilamiento iniciadas por estos roles invocarán el Hook.
  - Tipo: matriz de cadenas
  - Número máximo de artículos: 50
  - Artículos mínimos: 1
- `Exclude`: Lista de funciones de IAM ARNs para las pilas que quieres excluir. El Hook se invocará en todas las pilas, excepto en las iniciadas por los roles especificados.
  - Tipo: matriz de cadenas
  - Número máximo de objetos: 50
  - Artículos mínimos: 1

Cada función de pila de las `Exclude` matrices `Include` y debe cumplir los siguientes requisitos de patrón y longitud:

- Patrón: `arn:.+:iam::[0-9]{12}:role/.+`
- Longitud máxima: 256

`StackRoles`permitir caracteres comodín en las siguientes secciones de sintaxis del ARN:

- `partition`
- `account-id`
- `resource-id`

Para ver ejemplos de uso de caracteres comodín en las secciones de sintaxis del ARN, consulte. Uso de caracteres comodín con nombres de objetivos de Hook

`Include` y `Exclude`

Cada filtro (`StackNames`y`StackRoles`) tiene una `Include` lista y `Exclude` una lista. `StackNames`Como ejemplo, el Hook solo se invoca en las pilas especificadas en la `Include` lista. Si los nombres de las pilas solo se especifican en la `Exclude` lista, el gancho solo se invoca en las pilas que no están en la `Exclude` lista. Si `Exclude` se especifican ambos `Include` y, el Hook apunta a lo que está en la `Include` lista y no a lo que está en la `Exclude` lista.

Por ejemplo, supongamos que tienes cuatro pilas: A, B, C y D.

- `"Include": ["A","B"]`El Hook se invoca en A y B.
- `"Exclude": ["B"]`El Hook se invoca en A, C y D.
- `"Include": ["A","B","C"], "Exclude": ["A","D"]`El Hook se invoca en B y C.
- `"Include": ["A","B","C"], "Exclude": ["A”,"B","C"]`El Hook no se invoca en ninguna pila.

Ejemplos de filtros a nivel de pila

Esta sección proporciona ejemplos que puedes seguir para crear filtros a nivel de pila para AWS CloudFormation Hooks.

Ejemplo 1: Incluye pilas específicas

El siguiente ejemplo especifica una `Include` lista. El Hook solo se invoca en las pilas `stack-test-1` denominadas `stack-test-2` y`stack-test-3`.

```
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "StackFilters": {
        "FilteringCriteria": "ALL",
```

```
        "StackNames": {
          "Include": [
            "stack-test-1",
            "stack-test-2",
            "stack-test-3"
          ]
        }
      }
    }
  }
}
```

Ejemplo 2: Excluir pilas específicas

Si, en cambio, los nombres de las pilas se añaden a la `Exclude` lista, el Hook se invoca en cualquier pila que no tenga nombre`stack-test-1`, `stack-test-2` o`stack-test-3`.

```
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "StackFilters": {
        "FilteringCriteria": "ALL",
        "StackNames": {
          "Exclude": [
            "stack-test-1",
            "stack-test-2",
            "stack-test-3"
          ]
        }
      }
    }
  }
}
```

Ejemplo 3: Combinar incluir y excluir

Si `Include` no se especifican `Exclude` las listas, el Hook solo se invoca en las pilas `Include` que no están en la `Exclude` lista. En el siguiente ejemplo, el Hook solo se invoca en`stack-test-3`.

```
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "StackFilters": {
        "FilteringCriteria": "ALL",
        "StackNames": {
          "Include": [
            "stack-test-1",
            "stack-test-2",
            "stack-test-3"
          ],
          "Exclude": [
            "stack-test-1",
            "stack-test-2"
          ]
        }
      }
    }
  }
}
```

Ejemplo 4: Combinar nombres y roles de pila con **ALL** criterios

El siguiente Hook incluye tres nombres de pila y un rol de pila. Como `FilteringCriteria` se especifica comoALL, el Hook solo se invoca para las pilas que tienen un nombre de pila coincidente y una función de pila coincidentes.

```
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
```

```
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "StackFilters": {
        "FilteringCriteria": "ALL",
        "StackNames": {
          "Include": [
            "stack-test-1",
            "stack-test-2",
            "stack-test-3"
          ]
        },
        "StackRoles": {
          "Include": ["arn:aws:iam::123456789012:role/hook-role"]
        }
      }
    }
  }
}
```

Ejemplo 5: Combinar nombres y roles de pila con **ANY** criterios

El siguiente Hook incluye tres nombres de pila y un rol de pila. Como *`FilteringCriteria`* se especifica como*ANY*, el Hook se invoca para las pilas que tienen un nombre de pila coincidente o el rol de pila coincidente.

```
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "StackFilters": {
        "FilteringCriteria": "ANY",
        "StackNames": {
```

```
          "Include": [
            "stack-test-1",
            "stack-test-2",
            "stack-test-3"
          ]
        },
        "StackRoles": {
            "Include": ["arn:aws:iam::123456789012:role/hook-role"]
        }
      }
    }
  }
}
```

AWS CloudFormation Filtros de destino Hooks

En este tema se proporciona una guía sobre la configuración de los filtros de destino para AWS CloudFormation Hooks. Puedes usar filtros de destino para tener un control más detallado sobre cuándo y en qué recursos se invoca tu Hook. Puedes configurar filtros que van desde una segmentación simple por tipos de recursos hasta combinaciones más complejas de tipos de recursos, acciones y puntos de invocación.

Para especificar uno o más nombres de pila como filtros en tu configuración de Hooks, añade una `TargetFilters` clave a continuación. `HookConfiguration`

`TargetFilters`admite las siguientes propiedades.

`Actions`

Una matriz de cadenas que especifica las acciones a las que dirigirse. Para ver un ejemplo, consulta Ejemplo 1: filtro de destino básico.

Valores válidos: `CREATE` | `UPDATE` | `DELETE`

> **Note**
>
> Para `RESOURCESTACK`, y `CLOUD_CONTROL` los objetivos, son aplicables todas las acciones objetivo. En el caso de `CHANGE_SET` los objetivos, solo se aplica la `CREATE` acción. Para obtener más información, consulte ¡Gancho objetivos!.

`InvocationPoints`

Una matriz de cadenas que especifica los puntos de invocación al objetivo.

Valores válidos: `PRE_PROVISION`

`TargetNames`

Una matriz de cadenas que especifica los nombres de los tipos de recursos a los que se dirige, por ejemplo,`AWS::S3::Bucket`.

Los nombres de destino admiten nombres de destino concretos y la coincidencia completa de caracteres comodín. Para obtener más información, consulte Uso de caracteres comodín con nombres de objetivos de Hook.

Patrón: `^[a-zA-Z0-9]{2,64}::[a-zA-Z0-9]{2,64}::[a-zA-Z0-9]{2,64}$`

Máximo: `50`

`Targets`

Matriz de objetos que especifica la lista de objetivos que se van a utilizar para el filtrado de objetivos.

Cada objetivo de la matriz de objetivos tiene las siguientes propiedades.

`Actions`

La acción del objetivo especificado.

Valores válidos: `CREATE` | `UPDATE` | `DELETE`

`InvocationPoints`

El punto de invocación del objetivo especificado.

Valores válidos: `PRE_PROVISION`

`TargetNames`

El nombre del tipo de recurso al que se va a destinar.

> **Note**
>
> No puede incluir la matriz de `Targets` objetos y las `InvocationPoints` matrices `TargetNamesActions`, o al mismo tiempo. Si desea utilizar estos tres elementos`Targets`,

debe incluirlos en la matriz de `Targets` objetos. Para ver un ejemplo, consulta Ejemplo 2: Uso de la matriz de `Targets` objetos.

Ejemplos de filtros de destino

Esta sección proporciona ejemplos que puedes seguir para crear filtros de destino para AWS CloudFormation Hooks.

Ejemplo 1: filtro de destino básico

Para crear un filtro de destino básico que se centre en tipos de recursos específicos, utilice el `TargetFilters` objeto con la `Actions` matriz. La siguiente configuración del filtro de destino invocará el Hook en todas `Create` `Delete` las operaciones de destino especificadas (en este caso, tanto como `RESOURCE` las `STACK` operaciones). `Update`

```
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "TargetFilters": {
        "Actions": [
           "Create",
           "Update",
           "Delete"
        ]
      }
    }
  }
}
```

Ejemplo 2: Uso de la matriz de **`Targets`** objetos

Para filtros más avanzados, puedes usar la matriz de `Targets` objetos para enumerar combinaciones específicas de objetivos, acciones y puntos de invocación. La siguiente configuración

de filtro de destino invocará el Hook antes `CREATE` y `UPDATE` las acciones en los buckets S3 y las tablas de DynamoDB. Se aplica a ambas operaciones. `STACK` `RESOURCE`

```
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "TargetFilters": {
        "Targets": [
          {
             "TargetName": "AWS::S3::Bucket",
             "Action": "CREATE",
             "InvocationPoint": "PRE_PROVISION"
          },
          {
             "TargetName": "AWS::S3::Bucket",
             "Action": "UPDATE",
             "InvocationPoint": "PRE_PROVISION"
          },
          {
             "TargetName": "AWS::DynamoDB::Table",
             "Action": "CREATE",
             "InvocationPoint": "PRE_PROVISION"
          },
          {
             "TargetName": "AWS::DynamoDB::Table",
             "Action": "UPDATE",
             "InvocationPoint": "PRE_PROVISION"
          }
        ]
      }
    }
  }
}
```

Uso de caracteres comodín con nombres de objetivos de Hook

Puede utilizar caracteres comodín como parte del nombre del objetivo. Puedes usar caracteres comodín (*y?) en los nombres de los objetivos de Hook. El asterisco (*) representa cualquier combinación de caracteres. El signo de interrogación (?) representa cualquier carácter individual. Puede utilizar varios ? caracteres * y en un nombre de destino.

Example : ejemplos de caracteres comodín del nombre de destino en los esquemas de Hook

El siguiente ejemplo se dirige a todos los tipos de recursos compatibles con Amazon S3.

```
{
...
    "handlers": {
       "preCreate": {
            "targetNames": [
                "AWS::S3::*"
            ],
            "permissions": []
       }
    }
...
}
```

El siguiente ejemplo coincide con todos los tipos de recursos que tienen»Bucket"en el nombre.

```
{
...
    "handlers": {
       "preCreate": {
            "targetNames": [
                "AWS::*::Bucket*"
            ],
            "permissions": []
       }
    }
...
}
```

Esto `AWS::*::Bucket*` podría traducirse en cualquiera de los siguientes tipos de recursos concretos:

- AWS::Lightsail::Bucket
- AWS::S3::Bucket
- AWS::S3::BucketPolicy
- AWS::S3Outpost::Bucket
- AWS::S3Outpost::BucketPolicy

Example : Ejemplos de caracteres comodín en los nombres de destino en los esquemas de configuración de Hook

El siguiente ejemplo de configuración invoca el Hook para CREATE las operaciones en todos los tipos de recursos de Amazon S3 y para UPDATE las operaciones en todos los tipos de recursos de tablas con nombre asignado, como AWS::DynamobDB::Table oAWS::Glue::Table.

```
{
   "CloudFormationConfiguration": {
        "HookConfiguration": {
            "TargetStacks": "ALL",
            "FailureMode": "FAIL",
            "Properties": {},
            "TargetFilters":{
                 "Targets": [
                    {
                        "TargetName": "AWS::S3::*",
                        "Action": "CREATE",
                        "InvocationPoint": "PRE_PROVISION"
                    },
                    {
                        "TargetName": "AWS::*::Table",
                        "Action": "UPDATE",
                        "InvocationPoint": "PRE_PROVISION"
                    }
                ]
            }
        }
   }
}
```

El siguiente ejemplo de configuración invoca el Hook CREATE y UPDATE las operaciones en todos los tipos de recursos de Amazon S3 y también UPDATE las operaciones en todos los tipos de recursos de tablas con nombre asignado, como AWS::DynamobDB::Table oAWS::Glue::Table. CREATE

```
{
   "CloudFormationConfiguration": {
        "HookConfiguration": {
            "TargetStacks": "ALL",
            "FailureMode": "FAIL",
            "Properties": {},
            "TargetFilters":{
                "TargetNames": [
                    "AWS::S3::*",
                    "AWS::*::Table"
                ],
                "Actions": [
                    "CREATE",
                    "UPDATE"
                ],
                "InvocationPoints": [
                    "PRE_PROVISION"
                ]
            }
        }
   }
}
```

Example : pilas **`Include`** específicas

En los siguientes ejemplos se especifica una `Include` lista. El Hook solo se invoca si los nombres de las pilas comienzan por`stack-test-`.

```
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "StackFilters": {
        "FilteringCriteria": "ALL",
        "StackNames": {
          "Include": [
            "stack-test-*"
```

```
          ]
        }
      }
    }
  }
}
```

Example : **`Exclude`** pilas específicas

En los siguientes ejemplos se especifica una `Exclude` lista. El Hook se invoca en cualquier pila que no empiece por`stack-test-`.

```
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "StackFilters": {
        "FilteringCriteria": "ALL",
        "StackNames": {
          "Exclude": [
            "stack-test-*"
          ]
        }
      }
    }
  }
}
```

Example : Combinando **`Include`** y **`Exclude`** para pilas específicas

Si `Include` se especifican `Exclude` listas, el Hook solo se invoca en las pilas que coincidan con las `Include` que no coincidan en la `Exclude` lista. En el siguiente ejemplo, el Hook se invoca en todas las pilas que comiencen por, `stack-test-` excepto en las pilas denominadas`stack-test-1`, y`stack-test-2`. `stack-test-3`

```
{
```

```
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "StackFilters": {
        "FilteringCriteria": "ALL",
        "StackNames": {
          "Include": [
            "stack-test-*"
          ],
          "Exclude": [
            "stack-test-1",
            "stack-test-2",
            "stack-test-3"
          ]
        }
      }
    }
  }
}
```

Example : funciones específicas **`Include`**

En el siguiente ejemplo, se especifica una `Include` lista con dos patrones de caracteres comodín. La primera entrada ejecutará el Hook para cualquier rol que comience por un `partition` y`account-id`. `hook-role` La segunda entrada se ejecutará para cualquier rol en cualquiera de los roles a `partition` los que pertenezca `account-id``123456789012`.

```
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
```

```
      "StackFilters": {
        "FilteringCriteria": "ALL",
        "StackRoles": {
          "Include": [
            "arn:*:iam::*:role/hook-role*",
            "arn:*:iam::123456789012:role/*
          ]
        }
      }
    }
  }
}
```

Example : funciones **`Exclude`** específicas

En los ejemplos siguientes se especifica una `Exclude` lista con dos patrones de caracteres comodín. La primera entrada omitirá la ejecución de Hook cuando un rol tenga `exempt` en su nombre algún `partition` nombre. `account-id` La segunda entrada omitirá la ejecución de Hook cuando `account-id` `123456789012` se utilice un rol al que pertenezca con la operación de apilamiento.

```
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "StackFilters": {
        "FilteringCriteria": "ALL",
        "StackRoles": {
          "Exclude": [
            "arn:*:iam::*:role/*exempt*",
            "arn:*:iam::123456789012:role/*
          ]
        }
      }
    }
  }
```

```
}
```

Example : Combinación **`Include`** y **`Exclude`** para patrones de ARN de roles específicos

Si `Include` se especifican `Exclude` listas, el Hook solo se invoca en las pilas utilizadas con roles que coinciden con los `Include` que no coinciden en la `Exclude` lista. En el siguiente ejemplo, el Hook se invoca en las operaciones de apilamiento con cualquier `partition` `role` nombre y nombre, excepto si el rol pertenece a `account-id` `123456789012` él. `account-id`

```
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "StackFilters": {
        "FilteringCriteria": "ALL",
        "StackRoles": {
          "Include": [
            "arn:*:iam::*:role/*"
          ],
          "Exclude": [
            "arn:*:iam::123456789012:role/*"
          ]
        }
      }
    }
  }
}
```

Example : Combinar los nombres y roles de las pilas con todos los criterios

El siguiente Hook incluye un comodín con el nombre de la pila y un comodín con el nombre de la pila. Como `FilteringCriteria` se especifica como`ALL`, el Hook solo se invoca para las pilas que coincidan y coincidan`StackName`. `StackRoles`

```
{
  "CloudFormationConfiguration": {
```

```
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "StackFilters": {
        "FilteringCriteria": "ALL",
        "StackNames": {
          "Include": [
            "stack-test-*"
          ]
        },
        "StackRoles": {
          "Include": ["arn:*:iam::*:role/hook-role*"]
        }
      }
    }
  }
}
```

Example : Combinando **`StackNames`** y **`StackRoles`** con cualquier criterio

El siguiente Hook incluye un comodín con el nombre de la pila y un comodín con el nombre de la pila. Como `FilteringCriteria` se especifica como`ANY`, el Hook se invoca para la pila que tenga coincidencias `StackNames` o coincidencias. `StackRoles`

```
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "StackFilters": {
        "FilteringCriteria": "ANY",
        "StackNames": {
          "Include": [
```

```
            "stack-test-*"
          ]
        },
        "StackRoles": {
            "Include": ["arn:*:iam::*:role/hook-role*"]
        }
      }
    }
  }
}
```

Crea ganchos usando CloudFormation plantillas

Esta página proporciona enlaces a CloudFormation plantillas de muestra y temas de referencia técnica para Hooks.

Al usar CloudFormation plantillas para crear Hooks, puedes reutilizar tu plantilla para configurar tus Hooks de forma coherente y repetida. Este enfoque te permite definir tus Hooks una vez y luego aprovisionar los mismos Hooks una y otra vez en varias regiones Cuentas de AWS y regiones.

CloudFormation ofrece los siguientes tipos de recursos especializados para la creación de Guard y Lambda Hook.

Tarea	Solución	Enlaces
Crea un gancho de protección	Usa el tipo de `AWS::CloudFormation::GuardHook` recurso para crear y activar un garfio.	Plantilla de ejemplo Referencia técnica
Crear un gancho Lambda	Utilice el tipo `AWS::CloudFormation::LambdaHook` de recurso para crear y activar un Hook Lambda.	Plantilla de ejemplo Referencia técnica

CloudFormation también ofrece los siguientes tipos de recursos que puedes usar en tus plantillas de pila para crear Hooks personalizados.

Tarea	Solución	Enlaces
Registra un gancho	Usa el tipo de `AWS::CloudFormation::HookVersion` recurso para publicar una versión nueva o la primera de un Hook personalizado en el CloudFormation registro.	Plantillas de ejemplo Referencia técnica

Tarea	Solución	Enlaces
Establece la configuración del Hook	Usa el tipo de `AWS::CloudFormation::HookTypeConfig` recurso para especificar la configuración de un Hook personalizado.	Plantillas de ejemplo Referencia técnica
Establece la versión por defecto del Hook	Usa el tipo de `AWS::CloudFormation::HookDefaultVersion` recurso para especificar la versión predeterminada de un Hook personalizado.	Plantillas de ejemplo Referencia técnica
Registre su cuenta como editor	Utilice el tipo de `AWS::CloudFormation::Publisher` recurso para registrar su cuenta como publicador de extensiones públicas (Hooks, módulos y tipos de recursos) en el CloudFormation registro.	Referencia técnica
Publica un Hook públicamente	Usa el tipo de `AWS::CloudFormation::PublicTypeVersion` recurso para probar y publicar un Hook personalizado registrado como un Hook público de terceros.	Referencia técnica
Activa Hooks públicos de terceros	El tipo de `AWS::CloudFormation::TypeActivation` recurso funciona junto con el tipo de `AWS::CloudFormation::HookTypeConfig` recurso para activar un Hook público personalizado de terceros en tu cuenta.	Referencia técnica

Historial de documentos de la guía de usuario de AWS CloudFormation Hooks

En la siguiente tabla se describen los cambios importantes en la documentación desde la última versión de AWS CloudFormation Hooks. Para recibir notificaciones sobre las actualizaciones de esta documentación, puedes suscribirte a un RSS feed.

- Última actualización de la documentación: 8 de diciembre de 2023.

Cambio	Descripción	Fecha
Ganchos a nivel de pila	Ahora se admiten los Hooks a nivel de pila, lo que permite a los clientes utilizar los CloudFormation Hooks para evaluar nuevas plantillas y, potencialmente, impedir que las operaciones de apilamien to continúen.	13 de noviembre de 2024
API de control de nube de AWS Integración de Hooks	Los Hooks ahora están integrados con Cloud ControlAPI, lo que permite a los clientes usar CloudForm ation Hooks para inspeccionar proactivamente la configura ción de los recursos antes de aprovisionarlos. Si se encuentran recursos que no cumplen con las normas, el Hook falla en la operación e impide el aprovisionamiento de los recursos, o bien emite una advertencia y permite	13 de noviembre de 2024

	que la operación de aprovisio namiento continúe.	
AWS CloudFormation Guard Enlaces	AWS CloudFormation Guard es un lenguaje específico de dominio de código abierto y de uso general (DSL) que puede utilizar como autor. policy-as-code Guard Hooks puede evaluar el control API y CloudFormation las operacion es de la nube para inspeccio nar la configuración de los recursos antes del aprovisio namiento. Si se encuentra n recursos que no cumplen con las normas, el Hook falla en la operación e impide el aprovisionamiento de los recursos, o bien emite una advertencia y permite que la operación de aprovisio namiento continúe.	13 de noviembre de 2024
AWS Lambda Enlaces	AWS CloudFormation Los Lambda Hooks te permiten evaluar las API operacion es CloudFormation de Cloud Control comparándolas con tu propio código personalizado. Tu Hook puede impedir que se lleve a cabo una operación o emitir una advertencia a la persona que llama y permitir que la operación continúe.	13 de noviembre de 2024

Guía del usuario de Hooks	Versión inicial de la guía del usuario de AWS CloudFormation Hooks. Las actualizaciones incluyen una nueva introducción, un tutorial de introducción, conceptos y terminología, filtrado a nivel de pila y temas actualizados sobre los requisitos previos, la configuración y el desarrollo de CloudFormation Hooks.	8 de diciembre de 2023

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.