Guide de l'utilisateur de Hooks

AWS CloudFormation

AWS CloudFormation: Guide de l'utilisateur de Hooks

Table of Contents

Qu'est-ce que AWS CloudFormation Hooks ?

AWS CloudFormation Hooks est une fonctionnalité que vous pouvez utiliser pour vous assurer que vos CloudFormation ressources, vos piles et vos ensembles de modifications sont conformes aux meilleures pratiques de sécurité, d'exploitation et d'optimisation des coûts de votre organisation. CloudFormation Les Hooks peuvent également garantir ce même niveau de conformité avec vos AWS Cloud Control API ressources. Avec CloudFormation Hooks, vous pouvez fournir du code qui inspecte de manière proactive la configuration de vos AWS ressources avant le provisionnement. Si des ressources non conformes sont détectées, AWS CloudFormation l'opération échoue et empêche le provisionnement des ressources, ou émet un avertissement et autorise la poursuite de l'opération de provisionnement.

Vous pouvez utiliser les Hooks pour appliquer diverses exigences et directives. Par exemple, un Hook lié à la sécurité peut vérifier les groupes de sécurité conformément aux règles de trafic entrant et sortant appropriées pour votre Amazon Virtual Private Cloud (Amazon). VPC Un Hook lié aux coûts peut limiter les environnements de développement à n'utiliser que des types d'instances Amazon Elastic Compute Cloud (AmazonEC2) plus petits. Un Hook conçu pour la disponibilité des données peut imposer des sauvegardes automatiques pour Amazon Relational Database Service (RDSAmazon).

CloudFormation Les Hooks sont un type d'extension pris en charge dans le AWS CloudFormation registre. Le registre facilite la distribution et l'activation des Hooks, tant en public qu'en privé. Vous pouvez utiliser des Hooks prédéfinis ou créer vos propres Hooks à l'aide du CloudFormation CLI.

Ce guide fournit une vue d'ensemble de la structure des AWS CloudFormation Hooks, ainsi que des guides pour développer, enregistrer, tester, gérer et publier vos propres Hooks.

Création et gestion de AWS CloudFormation Hooks

AWS CloudFormation Les hooks fournissent un mécanisme permettant d'évaluer vos CloudFormation ressources avant d'autoriser la création, la modification ou la suppression de piles. Cette fonctionnalité vous permet de vous assurer que vos CloudFormation ressources sont conformes aux meilleures pratiques de votre organisation en matière de sécurité, d'exploitation et d'optimisation des coûts.

Pour créer un Hook, trois options s'offrent à vous.

- Guard Hook — Évalue les ressources à l'aide d'une AWS CloudFormation Guard règle.
- Lambda Hook — Transfère les demandes d'évaluation des ressources à une AWS Lambda fonction.
- Crochet personnalisé — Utilise un gestionnaire de crochet personnalisé que vous développez manuellement.

Guard Hook

Pour créer un crochet de protection, procédez comme suit :

1. Écrivez votre logique d'évaluation des ressources sous forme de règle de politique Guard en utilisant le langage spécifique au domaine Guard ()DSL.
2. Stockez la règle de politique Guard dans un compartiment Amazon S3.
3. Accédez à la CloudFormation console et commencez à créer un crochet de protection.
4. Indiquez le chemin Amazon S3 vers votre règle Guard.
5. Choisissez les cibles spécifiques que le Hook évaluera.
6. Choisissez les actions de déploiement (créer, mettre à jour, supprimer) qui appelleront votre Hook.
7. Choisissez la façon dont le Hook répond en cas d'échec de l'évaluation.
8. Lorsque la configuration est terminée, activez le Hook pour commencer l'application.

Lambda Hook

Pour créer un Lambda Hook, suivez les étapes principales suivantes :

1. Écrivez votre logique d'évaluation des ressources sous forme de fonction Lambda.

2. Accédez à la CloudFormation console et commencez à créer un Lambda Hook.
3. Indiquez le nom de ressource Amazon (ARN) pour votre fonction Lambda.
4. Choisissez les cibles spécifiques que le Hook évaluera.
5. Choisissez les actions de déploiement (créer, mettre à jour, supprimer) qui appelleront votre Hook.
6. Choisissez la façon dont le Hook répond en cas d'échec de l'évaluation.
7. Lorsque la configuration est terminée, activez le Hook pour commencer l'application.

Custom Hook

Les Hooks personnalisés sont des extensions que vous enregistrez dans le CloudFormation registre à l'aide de l'interface de ligne de CloudFormation commande (CFN-CLI).

Pour créer un Hook personnalisé, suivez les étapes principales suivantes :

1. Lancez le projet — Générez les fichiers nécessaires au développement d'un Hook personnalisé.
2. Modéliser le Hook — Écrivez un schéma qui définit le Hook et les gestionnaires qui spécifient les opérations qui peuvent appeler le Hook.
3. Enregistrez et activez le Hook — Après avoir créé un Hook, vous devez l'enregistrer dans le compte et dans la région où vous souhaitez l'utiliser pour l'activer.

Les rubriques suivantes fournissent des informations supplémentaires sur la création et la gestion des Hooks.

Rubriques

AWS CloudFormation Concepts de crochets

La terminologie et les concepts suivants sont essentiels à votre compréhension et à votre utilisation des AWS CloudFormation Hooks :

Crochet

Un Hook contient du code qui est invoqué immédiatement avant la CloudFormation création, la mise à jour ou la suppression de piles ou de ressources spécifiques. Il peut également être invoqué lors d'une opération de création d'un ensemble de modifications. Les Hooks peuvent inspecter le modèle, les ressources ou l'ensemble de modifications qui CloudFormation est sur le point d'être provisionné. De plus, les Hooks peuvent être invoqués immédiatement avant que le Cloud Control ne API crée, ne mette à jour ou ne supprime des ressources spécifiques.

Si un Hook identifie des configurations qui ne sont pas conformes aux directives organisationnelles définies dans votre logique Hook, vous pouvez choisir d'utiliser `WARN` les utilisateurs ou `FAIL` d' CloudFormation empêcher le provisionnement de la ressource.

Les crochets présentent les caractéristiques suivantes :

- Validation proactive : réduit les risques, les frais opérationnels et les coûts en identifiant les ressources non conformes avant leur création, leur mise à jour ou leur suppression.
- Application automatique : assure l'application dans votre système afin Compte AWS d'empêcher le provisionnement de ressources non conformes par. CloudFormation

Mode de défaillance

Votre logique Hook peut renvoyer un succès ou un échec. Une réponse positive permettra à l'opération de se poursuivre. Une défaillance due à des ressources non conformes peut avoir les conséquences suivantes :

- `FAIL`— Arrête l'opération de provisionnement.
- `WARN`— Permet de poursuivre le provisionnement avec un message d'avertissement.

La création de Hooks en `WARN` mode est un moyen efficace de surveiller le comportement des Hooks sans affecter les opérations de stack. Activez d'abord le `WARN` mode Hooks pour comprendre quelles

opérations seront affectées. Après avoir évalué les effets potentiels, vous pouvez passer en `FAIL` mode Hook pour commencer à empêcher les opérations non conformes.

Crochet et cibles

Les cibles Hook spécifient les opérations qu'un Hook évaluera. Il peut s'agir d'opérations sur :

- Ressources soutenues par CloudFormation (`Resources`)
- Modèles de pile (`STACK`)
- Ensembles de modifications (`CHANGE_SET`)
- Ressources prises en charge par le Cloud Control API (`CLOUD_CONTROL`)

Vous définissez une ou plusieurs cibles qui spécifient les opérations les plus larges que le Hook évaluera. Par exemple, vous pouvez créer un ciblage Hook `RESOURCES` pour cibler toutes les AWS ressources et `STACK` tous les modèles de stack. Vous pouvez également appliquer des filtres pour réduire la portée du Hook à l'opération exacte que vous souhaitez évaluer. Pour plus d'informations, consultez AWS CloudFormation Filtres de niveau Hooks Stack et AWS CloudFormation Filtres cibles Hooks.

Actions ciblées

L'action cible est le type d'opération qui déclenche un Hook. L'action peut être `CREATE`, `UPDATE`, ou`DELETE`.

> **Note**
>
> Lorsque vous utilisez `RESOURCE` les cibles`STACK`,, et `CLOUD_CONTROL` Hook, toutes les actions cibles sont applicables. Lorsque vous utilisez des cibles `CHANGE_SET` Hook, seule l'`CREATE`action est applicable.

Manipulateur de crochets

Pour les Hooks personnalisés, il s'agit du code qui gère l'évaluation. Il est associé à un point d'invocation cible et à une action cible qui marquent le point exact où un Hook s'exécute. Vous écrivez des gestionnaires qui hébergent la logique pour ces points spécifiques. Par exemple, un point d'invocation `PRE` cible avec une action `CREATE` cible crée un gestionnaire `preCreate` Hook. Le code

du gestionnaire Hook s'exécute lorsqu'un point d'appel cible et un service correspondants exécutent une action cible associée.

Valeurs valides : (`preCreate`| `preUpdate` |`preDelete`)

> **Important**
>
> Les opérations de pile qui se traduisent par le statut de `UpdateCleanup` n'invoquent pas de Hook. Par exemple, dans les deux scénarios suivants, le `preDelete` gestionnaire du Hook n'est pas invoqué :
>
> - la pile est mise à jour après la suppression d'une ressource du modèle.
> - une ressource dont le type de mise à jour est remplacé est supprimée.

Crochets de protection

Pour utiliser un AWS CloudFormation Guard Hook dans votre compte, vous devez activer le Hook pour le compte et la région dans lesquels vous souhaitez l'utiliser. L'activation d'un Hook le rend utilisable dans les opérations de stack du compte et de la région où il est activé.

Lorsque vous activez un Guard Hook, il CloudFormation crée une entrée dans le registre de votre compte pour le Hook activé en tant que Hook privé. Cela vous permet de définir toutes les propriétés de configuration incluses dans le Hook. Les propriétés de configuration définissent la manière dont le Hook est configuré pour une région Compte AWS et une région données.

Rubriques

AWS CLI commandes pour travailler avec Guard Hooks

Les AWS CLI commandes permettant d'utiliser les Guard Hooks sont les suivantes :

- activate-typepour démarrer le processus d'activation d'un Guard Hook.
- set-type-configurationpour spécifier les données de configuration d'un Hook dans votre compte.
- list-typespour répertorier les Hooks de votre compte.
- describe-typepour renvoyer des informations détaillées sur un Hook spécifique ou une version spécifique de Hook, y compris les données de configuration actuelles.
- deactivate-typepour supprimer un Hook précédemment activé de votre compte.

Rédiger des règles de garde pour évaluer les ressources pour Guard Hooks

AWS CloudFormation Guard est un langage open source et à usage général spécifique à un domaine (DSL) que vous pouvez utiliser pour créer. policy-as-code Cette rubrique explique comment utiliser Guard pour créer des exemples de règles qui peuvent être exécutées dans Guard Hook pour une évaluation CloudFormation et AWS Cloud Control API des opérations automatiques. Il se concentrera également sur les différents types d'entrées disponibles pour vos règles de garde en fonction du moment où votre crochet de garde fonctionne. Un Guard Hook peut être configuré pour s'exécuter lors des types d'opérations suivants :

- Opérations de ressources
- Opérations de pile
- Modifier les opérations du set

Pour plus d'informations sur la rédaction des règles Guard, voir AWS CloudFormation Guard Règles d'écriture

Rubriques

- Règles relatives à l'exploitation des ressources
- Règles de Stack Operation Guard
- Modifier les règles du jeu Operation Guard

Règles relatives à l'exploitation des ressources

Chaque fois que vous créez, mettez à jour ou supprimez une ressource, cela est considéré comme une opération de ressource. Par exemple, si vous exécutez la mise à jour d'une CloudFormation pile qui crée une nouvelle ressource, vous avez terminé une opération sur la ressource. Lorsque vous

créez, mettez à jour ou supprimez une ressource à l'aide de Cloud ControlAPI, cela est également considéré comme une opération de ressource. Vous pouvez configurer votre Guard Hook en fonction du `RESOURCE` ciblage et `CLOUD_CONTROL` des opérations dans la `TargetOperations` configuration de votre Hook. Lorsque votre Guard Hook évalue une opération de ressource, le moteur Guard évalue une entrée de ressource.

Rubriques

- Syntaxe d'entrée des ressources Guard
- Exemple d'entrée d'opération de ressource Guard
- Règles de protection en cas de modification des ressources

Syntaxe d'entrée des ressources Guard

Les ressources d'entrée du Guard sont les données mises à la disposition de vos règles Guard à des fins d'évaluation.

Voici un exemple de forme d'entrée de ressource :

```
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`

ID du Compte AWS contenant la ressource en cours d'évaluation.

`StackId`

L'ID de pile de la CloudFormation pile qui fait partie de l'opération de ressource. Ce champ est vide si l'appelant est Cloud ControlAPI.

`HookTypeName`

Le nom du Hook en cours d'exécution.

`HookTypeVersion`

Version du Hook en cours d'exécution.

`InvocationPoint`

Point exact de la logique de provisionnement où le Hook s'exécute.

Valeurs valides : (`CREATE_PRE_PROVISION`| `UPDATE_PRE_PROVISION` |`DELETE_PRE_PROVISION`)

`TargetName`

Nom de la ressource en cours d'évaluation.

`TargetType`

Type de ressource de la ressource en cours d'évaluation (exemple :`AWS::S3::Bucket`).

`TargetLogicalId`

Le `TargetLogicalId` de la ressource en cours d'évaluation. Si l'origine du Hook est CloudFormation, il s'agira de l'ID logique (également appelé nom logique) de la ressource. Si l'origine du Hook est Cloud ControlAPI, il s'agira d'une valeur construite.

`ChangeSetId`

L'ID de l'ensemble de modifications qui a été exécuté pour provoquer l'invocation du Hook. Cette valeur est vide si le changement de ressource a été initié par Cloud Control API ou par les `delete-stack` opérations `create-stackupdate-stack`, ou.

`Resources`

`ResourceLogicalID`

Lorsque l'opération est initiée par CloudFormation, `ResourceLogicalID` il s'agit de l'ID logique de la ressource dans le CloudFormation modèle.

Lorsque l'opération est initiée par Cloud ControlAPI, `ResourceLogicalID` il s'agit d'une combinaison du type de ressource, du nom, de l'ID de l'opération et de l'ID de demande.

`ResourceType`

Le nom du type de la ressource (exemple :`AWS::S3::Bucket`).

`ResourceProperties`

Les propriétés proposées pour la ressource en cours de modification. Lorsque le Guard Hook s'exécute contre les modifications CloudFormation des ressources, toutes les fonctions, tous les paramètres et toutes les transformations seront entièrement résolus. Si la ressource est supprimée, cette valeur sera vide.

`Previous`

`ResourceLogicalID`

Lorsque l'opération est initiée par CloudFormation, `ResourceLogicalID` il s'agit de l'ID logique de la ressource dans le CloudFormation modèle.

Lorsque l'opération est initiée par Cloud ControlAPI, `ResourceLogicalID` il s'agit d'une combinaison du type de ressource, du nom, de l'ID de l'opération et de l'ID de demande.

`ResourceType`

Le nom du type de la ressource (exemple :`AWS::S3::Bucket`).

`ResourceProperties`

Les propriétés actuelles associées à la ressource en cours de modification. Si la ressource est supprimée, cette valeur sera vide.

Exemple d'entrée d'opération de ressource Guard

L'exemple d'entrée suivant montre un Guard Hook qui recevra la définition de la `AWS::S3::Bucket` ressource en cours de modification. Il s'agit des données mises à la disposition de Guard à des fins d'évaluation.

```
HookContext:
  AwsAccountId: "00000000"
  StackId: ""
  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
```

Pour voir toutes les propriétés disponibles pour le type de ressource, consultez AWS: :S3 : :Bucket.

Règles de protection en cas de modification des ressources

Lorsqu'un Guard Hook évalue les modifications des ressources, il commence par télécharger toutes les règles configurées avec le Hook. Ces règles sont ensuite évaluées par rapport à l'entrée de ressources. Le Hook échouera si l'une des règles échoue lors de son évaluation. S'il n'y a aucun échec, le Hook réussira.

L'exemple suivant est une règle de protection qui évalue si la `ObjectLockEnabled` propriété est `true` destinée à un type de `AWS::S3::Bucket` ressource quelconque.

```
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.
  >>
}
```

Lorsque cette règle est exécutée à l'encontre de l'entrée suivante, elle échoue car la `ObjectLockEnabled` propriété n'est pas définie sur`true`.

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

Lorsque cette règle est exécutée à l'encontre de l'entrée suivante, elle est transmise car elle `ObjectLockEnabled` est définie sur`true`.

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

Lorsqu'un Hook échoue, les règles qui ont échoué sont CloudFormation repropagées vers Cloud ControlAPI. Si un bucket de journalisation a été configuré pour le Guard Hook, des informations supplémentaires sur les règles y seront fournies. Ces commentaires supplémentaires incluent les `Fix` informations `Violation` et.

Règles de Stack Operation Guard

Lorsqu'une CloudFormation pile est créée, mise à jour ou supprimée, vous pouvez configurer votre Guard Hook pour commencer par évaluer le nouveau modèle et éventuellement empêcher l'opération de pile de se poursuivre. Vous pouvez configurer votre Guard Hook pour cibler `STACK` les opérations dans la `TargetOperations` configuration de votre Hook.

Rubriques

- Syntaxe d'entrée Guard Stack
- Exemple d'entrée d'opération Guard Stack

- Règles de protection en cas de changement de pile

Syntaxe d'entrée Guard Stack

L'entrée pour les opérations Guard Stack fournit le CloudFormation modèle complet pour l'évaluation de vos règles Guard.

Voici un exemple de forme d'entrée de pile :

```
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`

L'ID du Compte AWS conteneur de la ressource.

`StackId`

L'ID de pile de la CloudFormation pile qui fait partie de l'opération de pile.

`HookTypeName`

Le nom du Hook en cours d'exécution.

`HookTypeVersion`

Version du Hook en cours d'exécution.

`InvocationPoint`

Point exact de la logique de provisionnement où le Hook s'exécute.

Valeurs valides : (`CREATE_PRE_PROVISION`| `UPDATE_PRE_PROVISION` |`DELETE_PRE_PROVISION`)

`TargetName`

Nom de la pile en cours d'évaluation.

`TargetType`

Cette valeur sera utilisée `STACK` lors de l'exécution en tant que Hook au niveau de la pile.

`ChangeSetId`

L'ID de l'ensemble de modifications qui a été exécuté pour provoquer l'invocation du Hook. Cette valeur est vide si l'opération de pile a été initiée par une `delete-stack` opération `create-stackupdate-stack`, ou.

`Proposed CloudFormation Template`

La valeur complète du CloudFormation modèle qui a été transmise à CloudFormation `create-stack` nos `update-stack` opérations. Cela inclut des éléments tels que `ResourcesOutputs`, et`Properties`. Il peut s'agir d'une YAML chaîne JSON ou d'une chaîne en fonction de ce qui a été fourni à CloudFormation.

Dans `delete-stack` les opérations, cette valeur sera vide.

`Previous`

Le dernier CloudFormation modèle déployé avec succès. Cette valeur est vide si la pile est créée ou supprimée.

Dans `delete-stack` les opérations, cette valeur sera vide.

> **Note**
>
> Les modèles fournis correspondent à ce qui est transmis aux opérations `create` ou aux opérations de `update` pile. Lors de la suppression d'une pile, aucune valeur de modèle n'est fournie.

Exemple d'entrée d'opération Guard Stack

L'exemple d'entrée suivant montre un Guard Hook qui recevra un modèle complet et le modèle précédemment déployé. Dans cet exemple, le modèle utilise le JSON format.

```
HookContext:
  AwsAccountId: 123456789012
  StackId: arn:aws:cloudformation:us-east-1:123456789012:stack/myteststack
  HookTypeName: org::templatechecker::hook
  HookTypeVersion: "00001"
  InvocationPoint: UPDATE_PRE_PROVISION
  TargetName: my-example-stack
  TargetType: CHANGE_SET
  TargetLogicalId: arn:aws:cloudformation:us-east-1:123456789012:changeSet/
SampleChangeSet/12a3b456-0e10-4ce0-9052-5d484a8c4e5b
  ChangeSetId: arn:aws:cloudformation:us-east-1:123456789012:changeSet/
SampleChangeSet/12a3b456-0e10-4ce0-9052-5d484a8c4e5b
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": {}
        }
    }
}
```

Règles de protection en cas de changement de pile

Lorsqu'un Guard Hook évalue les modifications de pile, il commence par télécharger toutes les règles configurées avec le Hook. Ces règles sont ensuite évaluées par rapport à l'entrée de ressources. Le Hook échouera si l'une des règles échoue lors de son évaluation. S'il n'y a aucun échec, le Hook réussira.

L'exemple suivant est une règle de protection qui évalue s'il existe des types de `AWS::S3::Bucket` ressources contenant une propriété appelée`BucketEncryption`, avec la valeur `SSEAlgorithm` définie sur `aws:kms` ou`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"
  >>
}
```

Lorsque la règle s'exécute sur le modèle suivant, elle le sera`fail`.

```
AWSTemplateFormatVersion: 2010-09-09
Description: S3 bucket without default encryption
Resources:
  EncryptedS3Bucket:
    Type: 'AWS::S3::Bucket'
    Properties:
      BucketName: !Sub 'encryptedbucket-${AWS::Region}-${AWS::AccountId}'
```

Lorsque la règle s'exécute sur le modèle suivant, elle le sera`pass`.

```
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
```

Modifier les règles du jeu Operation Guard

Lorsqu'un ensemble de CloudFormation modifications est créé, vous pouvez configurer votre Guard Hook pour évaluer le modèle et les modifications proposées dans l'ensemble de modifications afin de bloquer l'exécution de l'ensemble de modifications.

Rubriques

- Syntaxe d'entrée Guard Change Set
- Exemple d'entrée d'opération du kit Guard Change
- Règle de protection pour les opérations relatives aux ensembles de modifications

Syntaxe d'entrée Guard Change Set

Les données entrées dans le set de modifications de Guard sont les données mises à la disposition de vos règles Guard à des fins d'évaluation.

Voici un exemple de forme d'entrée d'un ensemble de modifications :

```
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 syntaxe `ResourceChange` du modèle est la suivante :

```
logicalResourceId: String
resourceType: String
```

```
action: CREATE, UPDATE, DELETE
lineNumber: Number
beforeContext: JSON String
afterContext: JSON String
```

`HookContext`

`AWSAccountID`

L'ID du Compte AWS conteneur de la ressource.

`StackId`

L'ID de pile de la CloudFormation pile qui fait partie de l'opération de pile.

`HookTypeName`

Le nom du Hook en cours d'exécution.

`HookTypeVersion`

Version du Hook en cours d'exécution.

`InvocationPoint`

Point exact de la logique de provisionnement où le Hook s'exécute.

Valeurs valides : (`CREATE_PRE_PROVISION`| `UPDATE_PRE_PROVISION` |`DELETE_PRE_PROVISION`)

`TargetName`

Nom de la pile en cours d'évaluation.

`TargetType`

Cette valeur sera utilisée `CHANGE_SET` lors de l'exécution en tant que Hook au niveau de l'ensemble de modifications.

`TargetLogicalId`

Cette valeur sera celle ARN de l'ensemble de modifications.

`ChangeSetId`

L'ID de l'ensemble de modifications qui a été exécuté pour provoquer l'invocation du Hook. Cette valeur est vide si l'opération de pile a été initiée par une `delete-stack` opération `create-stackupdate-stack`, ou.

`Proposed CloudFormation Template`

CloudFormation Modèle complet fourni à une `create-change-set` opération. Il peut s'agir d'une YAML chaîne JSON ou d'une chaîne en fonction de ce qui a été fourni à CloudFormation.

`Previous`

Le dernier CloudFormation modèle déployé avec succès. Cette valeur est vide si la pile est créée ou supprimée.

`Changes`

Le `Changes` modèle. Ceci répertorie les modifications apportées aux ressources.

Modifications

logicalResourceId

Le nom de ressource logique de la ressource modifiée.

resourceType

Type de ressource qui sera modifié.

action

Type d'opération effectuée sur la ressource.

Valeurs valides : (`CREATE`| `UPDATE` |`DELETE`)

lineNumber

Numéro de ligne du modèle associé à la modification.

beforeContext

JSONChaîne de propriétés de la ressource avant la modification :

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

afterContext

Une JSON chaîne de propriétés de la ressource après la modification :

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

Exemple d'entrée d'opération du kit Guard Change

L'exemple d'entrée suivant montre un Guard Hook qui recevra un modèle complet, le modèle précédemment déployé et une liste des modifications apportées aux ressources. Dans cet exemple, le modèle utilise le JSON format.

```
HookContext:
  AwsAccountId: "00000000"
  StackId: my-example-stack
  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
\"}}}"
  }
]
```

Règle de protection pour les opérations relatives aux ensembles de modifications

L'exemple suivant est une règle Guard qui évalue les modifications apportées aux compartiments Amazon S3 et garantit qu'elles ne `VersionConfiguration` sont pas désactivées.

```
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'
    }
}
```

Préparez-vous à créer un crochet de protection

Avant de créer un Guard Hook, vous devez remplir les conditions préalables suivantes :

- Vous devez déjà avoir créé une règle de garde. Pour plus d'informations, consultez le Règles de Write Guard pour les Hooks.
- L'utilisateur ou le rôle qui crée le Hook doit disposer des autorisations suffisantes pour activer les Hooks.
- Pour utiliser le AWS CLI ou an SDK pour créer un Guard Hook, vous devez créer manuellement un rôle d'exécution avec IAM des autorisations et une politique de confiance CloudFormation permettant d'invoquer un Guard Hook.

Création d'un rôle d'exécution pour un Guard Hook

Un Hook utilise un rôle d'exécution pour les autorisations dont il a besoin pour invoquer ce Hook dans votre Compte AWS.

Ce rôle peut être créé automatiquement si vous créez un Guard Hook à partir du AWS Management Console ; sinon, vous devez créer ce rôle vous-même.

La section suivante explique comment configurer les autorisations pour créer votre Guard Hook.

Autorisations nécessaires

Suivez les instructions de la section Créer un rôle à l'aide de politiques de confiance personnalisées dans le Guide de IAM l'utilisateur pour créer un rôle avec une politique de confiance personnalisée.

Effectuez ensuite les étapes suivantes pour configurer vos autorisations :

1. Associez la politique de privilèges minimaux suivante au IAM rôle que vous souhaitez utiliser pour créer le Guard Hook.

```
{
  "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. Donnez à votre Hook l'autorisation d'assumer le rôle en ajoutant une politique de confiance au rôle. Vous trouverez ci-dessous un exemple de politique de confiance que vous pouvez utiliser.

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

Activez un Guard Hook dans votre compte

La rubrique suivante explique comment activer un Guard Hook dans votre compte, afin de le rendre utilisable dans le compte et la région dans lesquels il a été activé.

Rubriques

- Activer un crochet de protection (console)
- Activer un crochet de protection (AWS CLI)
- Ressources connexes

Activer un crochet de protection (console)

Pour activer un Guard Hook à utiliser sur votre compte

1. Connectez-vous à la AWS CloudFormation console AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/cloudformation.

2. Dans la barre de navigation en haut de l'écran, choisissez l' Région AWS endroit où vous souhaitez créer le Hook in.
3. Si vous n'avez pas encore créé de règle Guard, créez votre règle Guard, stockez-la dans Amazon S3, puis revenez à cette procédure. Reportez-vous aux exemples de règles Rédiger des règles de garde pour évaluer les ressources pour Guard Hooks présentés pour commencer.

   Si vous avez déjà créé votre règle Guard et que vous l'avez stockée dans S3, passez à l'étape suivante.

   > **Note**
   >
   > L'objet stocké dans S3 doit avoir l'une des extensions de fichier suivantes : `.guard.zip`, ou `.tar.gz`.

4. Pour la source Guard Hook, stockez vos règles Guard dans S3, procédez comme suit :

   - Pour S3 URI, spécifiez le chemin S3 vers votre fichier de règles ou utilisez le bouton Parcourir S3 pour ouvrir une boîte de dialogue permettant de rechercher et de sélectionner l'objet S3.
   - (Facultatif) Pour la version de l'objet, si le contrôle de version de votre compartiment S3 est activé, vous pouvez sélectionner une version spécifique de l'objet S3.

     Le Guard Hook télécharge vos règles depuis S3 chaque fois que le Hook est invoqué. Pour éviter toute modification ou suppression accidentelle, nous vous recommandons d'utiliser une version lors de la configuration de votre Guard Hook.

5. (Facultatif) Pour le rapport de sortie du compartiment S3 pour Guard, spécifiez un compartiment S3 pour stocker le rapport de sortie Guard. Ce rapport contient les résultats de vos validations des règles Guard.

   Pour configurer la destination du rapport de sortie, choisissez l'une des options suivantes :

   - Cochez la case Utiliser le même compartiment dans lequel les règles de mon Guard sont stockées pour utiliser le même compartiment que celui dans lequel se trouvent vos règles Guard.
   - Choisissez un autre nom de compartiment S3 pour stocker le rapport de sortie Guard.

6. (Facultatif) Développez les paramètres d'entrée de la règle Guard, puis fournissez les informations suivantes sous Stocker les paramètres d'entrée de la règle Guard dans S3 :

- Pour S3 URI, spécifiez le chemin S3 vers un fichier de paramètres ou utilisez le bouton Parcourir S3 pour ouvrir une boîte de dialogue permettant de rechercher et de sélectionner l'objet S3.
- (Facultatif) Pour la version de l'objet, si le contrôle de version de votre compartiment S3 est activé, vous pouvez sélectionner une version spécifique de l'objet S3.

7. Choisissez Suivant.
8. Pour le nom du crochet, choisissez l'une des options suivantes :

   - Entrez un nom court et descriptif qui sera ajouté par la suite`Private::Guard::`.
     Par exemple, si vous entrez*`MyTestHook`*, le nom complet du Hook devient`Private::Guard::`*`MyTestHook`*.
   - Fournissez le nom complet du Hook (également appelé alias) en utilisant le format suivant : *`Provider`*`::`*`ServiceName`*`::`*`HookName`*

9. Pour les cibles Hook, choisissez les éléments à évaluer :

   - Piles : évalue les modèles de pile lorsque les utilisateurs créent, mettent à jour ou suppriment des piles.
   - Ressources — Évalue les modifications individuelles des ressources lorsque les utilisateurs mettent à jour les piles.
   - Ensembles de modifications : évalue les mises à jour planifiées lorsque les utilisateurs créent des ensembles de modifications.
   - Contrôle du cloud API — Évalue les opérations de création, de mise à jour ou de suppression initiées par le contrôle API du cloud.

10. Pour Actions, choisissez les actions (créer, mettre à jour, supprimer) qui appelleront votre Hook.
11. Pour le mode Hook, choisissez la façon dont le Hook répond lorsque l'évaluation des règles échoue :

    - Avertir : émet des avertissements à l'intention des utilisateurs, mais autorise la poursuite des actions. Cela est utile pour les validations non critiques ou les contrôles informatifs.
    - Echec — Empêche le déroulement de l'action. Cela est utile pour appliquer des politiques de conformité ou de sécurité strictes.

12. Pour le rôle d'exécution, choisissez le IAM rôle que les CloudFormation Hooks assument pour récupérer vos règles Guard depuis S3 et rédigez éventuellement un rapport de sortie détaillé de

Guard. Vous pouvez soit CloudFormation autoriser la création automatique d'un rôle d'exécution pour vous, soit spécifier un rôle que vous avez créé.

13. Choisissez Suivant.
14. (Facultatif) Pour les filtres Hook, procédez comme suit :

    a. Pour le filtre de ressources, spécifiez les types de ressources qui peuvent appeler le Hook. Cela garantit que le Hook n'est invoqué que pour les ressources pertinentes.

    b. Pour les critères de filtrage, choisissez la logique d'application des filtres de nom de pile et de rôle de pile :

    - Tous les noms de pile et tous les rôles de pile — Le Hook ne sera invoqué que lorsque tous les filtres spécifiés correspondent.
    - Tous les noms de pile et rôles de pile — Le Hook sera invoqué si au moins l'un des filtres spécifiés correspond.

    > **Note**
    >
    > Pour les API opérations Cloud Control, tous les filtres relatifs aux noms de pile et aux rôles de pile sont ignorés.

    c. Pour les noms de pile, incluez ou excluez des piles spécifiques des invocations Hook.

    - Pour Inclure, spécifiez les noms des piles à inclure. Utilisez-le lorsque vous souhaitez cibler un petit ensemble de piles spécifiques. Seules les piles spécifiées dans cette liste invoqueront le Hook.
    - Pour Exclure, spécifiez les noms des piles à exclure. Utilisez-le lorsque vous souhaitez invoquer le Hook sur la plupart des piles, mais en exclure quelques unes en particulier. Toutes les piles, à l'exception de celles répertoriées ici, invoqueront le Hook.

    d. Pour les rôles Stack, incluez ou excluez des piles spécifiques des invocations Hook en fonction de leurs rôles associés. IAM

    - Pour Inclure, spécifiez un ou plusieurs IAM rôles ARNs pour cibler les piles associées à ces rôles. Seules les opérations de stack initiées par ces rôles invoqueront le Hook.
    - Pour Exclure, spécifiez un ou plusieurs IAM rôles ARNs pour les piles que vous souhaitez exclure. Le Hook sera invoqué sur toutes les piles sauf celles initiées par les rôles spécifiés.

15. Choisissez Suivant.
16. Sur la page Vérifier et activer, passez en revue vos choix. Pour apporter des modifications, choisissez Modifier dans la section correspondante.
17. Lorsque vous êtes prêt à continuer, choisissez Activate Hook.

Activer un crochet de protection (AWS CLI)

Avant de continuer, vérifiez que vous avez créé la règle Guard et le rôle d'exécution que vous allez utiliser avec ce Hook. Pour plus d'informations, consultez Rédiger des règles de garde pour évaluer les ressources pour Guard Hooks et Création d'un rôle d'exécution pour un Guard Hook.

Pour activer un Guard Hook à utiliser sur votre compte (AWS CLI)

1. Pour commencer à activer un Hook, utilisez ce qui suit activate-typecommande, en remplaçant les espaces réservés par vos valeurs spécifiques. Cette commande autorise le Hook à utiliser un rôle d'exécution spécifié par votre Compte 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:aws:iam::123456789012:role/my-execution-role \
  --region us-west-2
```

2. Pour terminer l'activation du Hook, vous devez le configurer à l'aide d'un fichier JSON de configuration.

   Utilisez la cat commande pour créer un JSON fichier avec la structure suivante. Pour de plus amples informations, veuillez consulter Référence syntaxique du schéma de configuration Hook.

```
$ 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"
      }
    }
  }
}
```

- `HookInvocationStatus`: défini sur `ENABLED` pour activer le Hook.
- `TargetOperations`: Spécifiez les opérations sur lesquelles exécuter le Hook.
- `FailureMode` : Définissez sur `FAIL` ou `WARN`.
- `ruleLocation`: Remplacez-le par le S3 URI dans lequel votre règle est stockée. L'objet stocké dans S3 doit avoir l'une des extensions de fichier suivantes : `.guard.zip`, et `.tar.gz`.
- `logBucket`: (Facultatif) Spécifiez le nom d'un compartiment S3 pour les JSON rapports Guard.

3. Utilisez ce qui suit set-type-configurationcommande, avec le JSON fichier que vous avez créé, pour appliquer la configuration. Remplacez les espaces réservés par vos valeurs spécifiques.

```
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
```

Ressources connexes

Nous fournissons des exemples de modèles que vous pouvez utiliser pour comprendre comment déclarer un Guard Hook dans un modèle de CloudFormation pile. Pour plus d'informations, consultez .AWS::CloudFormation::GuardHook dans le guide de l'utilisateur AWS CloudFormation .

Afficher les journaux des Guard Hooks sur votre compte

Lorsque vous activez un Guard Hook, vous pouvez spécifier un compartiment Amazon S3 comme destination pour le rapport de sortie Hook. Une fois activé, le Hook stocke automatiquement les résultats de vos validations de règles Guard dans le compartiment spécifié. Vous pouvez ensuite consulter ces résultats dans la console Amazon S3.

Afficher les journaux Guard Hook dans la console Amazon S3

Pour consulter le fichier journal de sortie de Guard Hook

1. Connectez-vous au. https://console.aws.amazon.com/s3/
2. Dans la barre de navigation en haut de l'écran, choisissez votre Région AWS.
3. Choisissez Buckets.
4. Choisissez le compartiment que vous avez sélectionné pour votre rapport de sortie Guard.
5. Choisissez le fichier journal du rapport de sortie de validation souhaité.
6. Choisissez si vous souhaitez télécharger le fichier ou l'ouvrir pour l'afficher.

Supprimer Guard Hooks de votre compte

Lorsque vous n'avez plus besoin d'un Guard Hook activé, suivez les procédures suivantes pour le supprimer de votre compte.

Pour désactiver temporairement un Hook au lieu de le supprimer, consultezDésactiver et activer les AWS CloudFormation Hooks.

Rubriques

- Supprimer un Guard Hook dans votre compte (console)
- Supprimer un Guard Hook dans votre compte (AWS CLI)

Supprimer un Guard Hook dans votre compte (console)

Pour supprimer un Guard Hook de votre compte

1. Connectez-vous à la AWS CloudFormation console AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/cloudformation.
2. Dans la barre de navigation en haut de l'écran, choisissez l' Région AWS emplacement du crochet.
3. Dans le volet de navigation, choisissez Hooks.
4. Sur la page Hooks, trouvez le Guard Hook que vous souhaitez supprimer.
5. Cochez la case à côté de votre Hook et choisissez Supprimer.

6. Lorsque vous êtes invité à confirmer, saisissez le nom du crochet pour confirmer la suppression du crochet spécifié, puis choisissez Supprimer.

Supprimer un Guard Hook dans votre compte (AWS CLI)

> **Note**
>
> Avant de pouvoir supprimer le Hook, vous devez d'abord le désactiver. Pour de plus amples informations, veuillez consulter Désactiver et activer un Hook dans votre compte (AWS CLI).

Utilisez ce qui suit deactivate-typecommande pour désactiver un Hook, ce qui le supprime de votre compte. Remplacez les espaces réservés par vos valeurs spécifiques.

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

Crochets Lambda

Pour utiliser un AWS Lambda Hook dans votre compte, vous devez d'abord activer le Hook pour le compte et la région dans lesquels vous souhaitez l'utiliser. L'activation d'un Hook le rend utilisable dans les opérations de stack du compte et de la région où il est activé.

Lorsque vous activez un Lambda Hook, CloudFormation crée une entrée dans le registre de votre compte pour le Hook activé en tant que Hook privé. Cela vous permet de définir toutes les propriétés de configuration incluses dans le Hook. Les propriétés de configuration définissent la manière dont le Hook est configuré pour une région Compte AWS et une région données.

Rubriques

- AWS CLI commandes pour travailler avec les Hooks Lambda
- Créez des fonctions Lambda pour évaluer les ressources pour les Lambda Hooks
- Préparez-vous à créer un crochet Lambda
- Activez un Lambda Hook dans votre compte
- Afficher les journaux des Lambda Hooks dans votre compte
- Supprimer les Lambda Hooks de votre compte

AWS CLI commandes pour travailler avec les Hooks Lambda

Les AWS CLI commandes permettant d'utiliser les Hooks Lambda sont les suivantes :

- activate-typepour démarrer le processus d'activation d'un Lambda Hook.
- set-type-configurationpour spécifier les données de configuration d'un Hook dans votre compte.
- list-typespour répertorier les Hooks de votre compte.
- describe-typepour renvoyer des informations détaillées sur un Hook spécifique ou une version spécifique de Hook, y compris les données de configuration actuelles.
- deactivate-typepour supprimer un Hook précédemment activé de votre compte.

Créez des fonctions Lambda pour évaluer les ressources pour les Lambda Hooks

AWS CloudFormation Lambda Hooks vous permet d'évaluer CloudFormation et d'effectuer AWS Cloud Control API des opérations par rapport à votre propre code personnalisé. Votre Hook peut bloquer le déroulement d'une opération ou envoyer un avertissement à l'appelant et autoriser le déroulement de l'opération. Lorsque vous créez un Lambda Hook, vous pouvez le configurer pour intercepter et évaluer les opérations suivantes : CloudFormation

- Opérations de ressources
- Opérations de pile
- Modifier les opérations du set

Rubriques

- Développement d'un hook Lambda
- Évaluation des opérations sur les ressources avec les Lambda Hooks
- Évaluation des opérations de stack avec les Lambda Hooks
- Évaluation des opérations relatives aux ensembles de modifications à l'aide de Lambda Hooks

Développement d'un hook Lambda

Lorsque les Hooks invoquent votre Lambda, le Lambda attend jusqu'à 30 secondes pour évaluer l'entrée. Le Lambda renverra une JSON réponse indiquant si le Hook a réussi ou échoué.

Rubriques

Demande d'entrée

L'entrée transmise à votre fonction Lambda dépend de l'opération cible Hook (exemples : stack, resource ou change set).

Entrée de réponse

Afin de communiquer à Hooks si votre demande a réussi ou échoué, votre fonction Lambda doit renvoyer une JSON réponse.

Voici un exemple de forme de la réponse attendue par Hooks :

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

hookStatus

L'état du Hook. Ce champ est obligatoire.

Valeurs valides : (`SUCCESS`| `FAILED` |`IN_PROGRESS`)

> **Note**
>
> Un Hook peut revenir `IN_PROGRESS` 3 fois. Si aucun résultat n'est renvoyé, le Hook échouera. Pour un Lambda Hook, cela signifie que votre fonction Lambda peut être invoquée jusqu'à 3 fois.

errorCode

Indique si l'opération a été évaluée et jugée non valide, ou si des erreurs se sont produites dans le Hook, empêchant l'évaluation. Ce champ est obligatoire en cas d'échec du Hook.

Valeurs valides : (`NonCompliant`|`InternalFailure`)

message

Le message envoyé à l'appelant expliquant pourquoi le Hook a réussi ou échoué.

> **Note**
>
> Lors de l'évaluation CloudFormation des opérations, ce champ est tronqué à 4 096 caractères.
> Lors de l'évaluation API des opérations Cloud Control, ce champ est tronqué à 1024 caractères.

clientRequestToken

Le jeton de demande qui a été fourni en entrée de la demande Hook. Ce champ est obligatoire.

callbackContext

Si vous indiquez que `hookStatus` c'est le cas, `IN_PROGRESS` vous transmettez un contexte supplémentaire fourni en entrée lorsque la fonction Lambda est réinvoquée.

callbackDelaySeconds

Combien de temps les Hooks doivent-ils attendre pour invoquer à nouveau ce Hook ?

Exemples

Voici un exemple de réponse réussie :

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

Voici un exemple d'échec de réponse :

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

Évaluation des opérations sur les ressources avec les Lambda Hooks

Chaque fois que vous créez, mettez à jour ou supprimez une ressource, cela est considéré comme une opération de ressource. Par exemple, si vous exécutez la mise à jour d'une CloudFormation pile qui crée une nouvelle ressource, vous avez terminé une opération sur la ressource. Lorsque vous créez, mettez à jour ou supprimez une ressource à l'aide de Cloud ControlAPI, cela est également considéré comme une opération de ressource. Vous pouvez configurer votre CloudFormation Lambda Hook en fonction du ciblage `RESOURCE` et `CLOUD_CONTROL` des opérations dans la configuration du Hook`TargetOperations`.

> **Note**
>
> Le gestionnaire `delete` Hook n'est invoqué que lorsqu'une ressource est supprimée à l'aide d'un déclencheur d'opération depuis `cloud-control delete-resource` ou`cloudformation delete-stack`.

Rubriques

- Syntaxe d'entrée des ressources Lambda Hook
- Exemple d'entrée de modification de ressource Lambda Hook
- Exemple de fonction Lambda pour les opérations sur les ressources

Syntaxe d'entrée des ressources Lambda Hook

Lorsque votre Lambda est invoqué pour une opération sur une ressource, vous recevez une JSON entrée contenant les propriétés de la ressource, les propriétés proposées et le contexte de l'invocation de Hook.

Voici un exemple de forme de l'JSONentrée :

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

`awsAccountId`

ID du Compte AWS contenant la ressource en cours d'évaluation.

`stackId`

L'ID de pile de la CloudFormation pile dont cette opération fait partie. Ce champ est vide si l'appelant est Cloud ControlAPI.

`changeSetId`

L'ID de l'ensemble de modifications qui a initié l'invocation de Hook. Cette valeur est vide si le changement de ressource a été initié par Cloud Control API ou par les `delete-stack` opérations `create-stackupdate-stack`, ou.

`hookTypeName`

Le nom du Hook en cours d'exécution.

`hookTypeVersion`

Version du Hook en cours d'exécution.

`hookModel`

`LambdaFunction`

Le Lambda actuel ARN invoqué par le Hook.

`actionInvocationPoint`

Point exact de la logique de provisionnement où le Hook s'exécute.

Valeurs valides : (`CREATE_PRE_PROVISION`| `UPDATE_PRE_PROVISION` |`DELETE_PRE_PROVISION`)

`requestData`

`targetName`

Nom de la ressource cible en cours de création.

`targetType`

Le type de cible en cours de création, par exemple`AWS::S3::Bucket`.

`targetLogicalId`

ID logique de la ressource en cours d'évaluation. Si l'origine de l'invocation de Hook est CloudFormation, il s'agira de l'ID de ressource logique défini dans votre CloudFormation modèle. Si l'origine de cette invocation de Hook est Cloud ControlAPI, il s'agira d'une valeur construite.

`targetModel`

`resourceProperties`

Les propriétés proposées pour la ressource en cours de modification. Si la ressource est supprimée, cette valeur sera vide.

`previousResourceProperties`

Les propriétés actuellement associées à la ressource en cours de modification. Si la ressource est créée, cette valeur sera vide.

`requestContext`

invocation

La tentative actuelle d'exécution du Hook.

callbackContext

Si le Hook a été réglé sur `IN_PROGRESS` et `callbackContext` a été renvoyé, il sera là après sa révocation.

Exemple d'entrée de modification de ressource Lambda Hook

Dans l'exemple de saisie suivant, le Guard Hook recevra la définition de la `AWS::DynamoDB::Table` ressource en cours de modification. Les `ReadCapacityUnits` paramètres `ProvisionedThroughput` et sont mis à jour de 3 à 10.

Pour plus d'informations sur les propriétés disponibles pour la ressource, voir AWS: :DynamoDB : :Table.

```
{
    "awsAccountId": "123456789",
    "stackId": "arn:aws:cloudformation:eu-central-1:123456789:stack/test-
stack/123456abcd",
    "hookTypeName": "my::lambda::resourcehookfunction",
    "hookTypeVersion": "00000008",
    "hookModel": {
        "LambdaFunction": "arn:aws:lambda:eu-
central-1:123456789:function:resourcehookfunction"
    },
    "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
    }
}
```

Exemple de fonction Lambda pour les opérations sur les ressources

Voici un exemple de cibles Lambda Hook. `Node.js` Il s'agit d'une fonction simple qui échoue à toute mise à jour des ressources de DynamoDB, qui essaie de définir une valeur `ProvisionedThroughput ReadCapacity` supérieure à 10. Si le Hook réussit, le message « ReadCapacity est correctement configuré » s'affichera à l'attention de l'appelant. Si la demande échoue à la validation, le Hook échouera avec le statut « ReadCapacity ne peut pas être supérieur à 10 ».

```
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;
};
```

Évaluation des opérations de stack avec les Lambda Hooks

Chaque fois que vous créez, mettez à jour ou supprimez une pile avec un nouveau modèle, vous pouvez configurer votre CloudFormation Lambda Hook pour commencer par évaluer le nouveau modèle et éventuellement bloquer le déroulement de l'opération de pile. Vous pouvez configurer votre CloudFormation Lambda Hook pour cibler les `STACK` opérations dans la configuration Hook`TargetOperations`.

Rubriques

Syntaxe d'entrée Lambda Hook Stack

Lorsque votre Lambda est invoqué pour une opération de pile, vous recevez une JSON demande contenant le contexte d'invocation Hook et le contexte de la demande. `actionInvocationPoint` En raison de la taille des CloudFormation modèles et de la taille d'entrée limitée acceptée par les fonctions Lambda, les modèles réels sont stockés dans un objet Amazon S3. L'entrée `requestData` inclut un Amazon S3 affecté URL à un autre objet, qui contient la version actuelle et précédente du modèle.

Voici un exemple de forme de l'JSONentrée :

```
{
    "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`

Le jeton de demande qui a été fourni en entrée de la demande Hook. Ce champ est obligatoire.

`awsAccountId`

L'ID du Compte AWS contenant la pile en cours d'évaluation.

`stackID`

L'ID de pile de la CloudFormation pile.

`changeSetId`

L'ID de l'ensemble de modifications qui a initié l'invocation de Hook. Cette valeur est vide si le changement de pile a été initié par Cloud Control API ou par les `delete-stack` opérations `create-stack`update-stack, ou.

hookTypeName

Le nom du Hook en cours d'exécution.

`hookTypeVersion`

Version du Hook en cours d'exécution.

`hookModel`

`LambdaFunction`

Le Lambda actuel ARN invoqué par le Hook.

`actionInvocationPoint`

Point exact de la logique de provisionnement où le Hook s'exécute.

Valeurs valides : (`CREATE_PRE_PROVISION`| `UPDATE_PRE_PROVISION` |`DELETE_PRE_PROVISION`)

`requestData`

`targetName`

Cette valeur sera`STACK`.

`targetType`

Cette valeur sera`STACK`.

`targetLogicalId`

Nom de la pile.

`payload`

L'Amazon S3 présigné URL contient un JSON objet avec les définitions de modèles actuelles et précédentes.

`requestContext`

Si le Hook est réinvoqué, cet objet sera défini.

`invocation`

La tentative actuelle d'exécution du Hook.

`callbackContext`

Si le Hook a été réglé sur `IN_PROGRESS` et `callbackContext` a été renvoyé, il sera présent lors de sa révocation.

La `payload` propriété contenue dans les données de demande est URL celle que votre code doit récupérer. Une fois qu'il a reçu leURL, vous obtenez un objet avec le schéma suivant :

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

`template`

Le CloudFormation modèle complet qui a été fourni à `create-stack` ou`update-stack`. Il peut s'agir d'une YAML chaîne JSON ou d'une chaîne en fonction de ce qui a été fourni à CloudFormation.

Dans `delete-stack` les opérations, cette valeur sera vide.

`previousTemplate`

Le CloudFormation modèle précédent. Il peut s'agir d'une YAML chaîne JSON ou d'une chaîne en fonction de ce qui a été fourni à CloudFormation.

Dans `delete-stack` les opérations, cette valeur sera vide.

Exemple d'entrée de modification de la pile Lambda Hook

Voici un exemple d'entrée de changement de pile. The Hook évalue une modification qui met le à jour `ObjectLockEnabled` à true et ajoute une SQS file d'attente Amazon :

```
{
    "clientRequestToken": "f8da6d11-b23f-48f4-814c-0fb6a667f50e",
    "awsAccountId": "123456789",
    "stackId": "arn:aws:cloudformation:eu-central-1:123456789:stack/david-ddb-test-
stack/400b40f0-8e72-11ef-80ab-02f2902f0df1",
    "changeSetId": null,
    "hookTypeName": "my::lambda::stackhook",
    "hookTypeVersion": "00000008",
    "hookModel": {
        "LambdaFunction": "arn:aws:lambda:eu-
central-1:123456789:function:stackhookfunction"
    },
    "actionInvocationPoint": "UPDATE_PRE_PROVISION",
    "requestData": {
        "targetName": "STACK",
        "targetType": "STACK",
        "targetLogicalId": "my-cloudformation-stack",
        "payload": "https://s3......"
    },
    "requestContext": {
        "invocation": 1,
        "callbackContext": null
    }
}
```

Voici un exemple `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}}}}"
}
```

Exemple de fonction Lambda pour les opérations de stack

L'exemple suivant Lambda Hook cible. `Node.js` Il s'agit d'une fonction simple qui télécharge la charge utile de l'opération de pile, analyse le modèle et JSON renvoie. `SUCCESS`

```
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;
};
```

Évaluation des opérations relatives aux ensembles de modifications à l'aide de Lambda Hooks

Chaque fois que vous créez un ensemble de modifications, vous pouvez configurer votre CloudFormation Lambda Hook pour d'abord évaluer le nouvel ensemble de modifications et éventuellement bloquer son exécution. Vous pouvez configurer votre CloudFormation Lambda Hook pour cibler les `CHANGE_SET` opérations dans la configuration Hook`TargetOperations`.

Rubriques

Lambda Hook modifie la syntaxe d'entrée du set

L'entrée pour les opérations d'ensemble de modifications est similaire à celle des opérations de pile, mais la charge utile des opérations inclut `requestData` également une liste des modifications de ressources introduites par l'ensemble de modifications.

Voici un exemple de forme de l'JSONentrée :

```
{
    "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`

Le jeton de demande qui a été fourni en entrée de la demande Hook. Ce champ est obligatoire.

`awsAccountId`

ID du Compte AWS contenant la pile en cours d'évaluation.

`stackID`

L'ID de pile de la CloudFormation pile.

`changeSetId`

L'ID de l'ensemble de modifications qui a initié l'invocation de Hook.

`hookTypeName`

Le nom du Hook en cours d'exécution.

`hookTypeVersion`

Version du Hook en cours d'exécution.

`hookModel`

`LambdaFunction`

Le Lambda actuel ARN invoqué par le Hook.

`requestData`

`targetName`

Cette valeur sera`CHANGE_SET`.

`targetType`

Cette valeur sera`CHANGE_SET`.

`targetLogicalId`

Le kit de modificationsARN.

`payload`

L'Amazon S3 présigné URL contient un JSON objet avec le modèle actuel, ainsi qu'une liste des modifications introduites par cet ensemble de modifications.

`requestContext`

Si le Hook est réinvoqué, cet objet sera défini.

`invocation`

La tentative actuelle d'exécution du Hook.

`callbackContext`

Si le Hook a été réglé sur `IN_PROGRESS` et `callbackContext` a été renvoyé, il sera présent lors de sa révocation.

La `payload` propriété contenue dans les données de demande est URL celle que votre code doit récupérer. Une fois qu'il a reçu leURL, vous obtenez un objet avec le schéma suivant :

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

`template`

Le CloudFormation modèle complet qui a été fourni à `create-stack` ou`update-stack`. Il peut s'agir d'une YAML chaîne JSON ou d'une chaîne en fonction de ce qui a été fourni à CloudFormation.

`changedResources`

Liste des ressources modifiées.

`action`

Type de modification appliqué à la ressource.

Valeurs valides : (`CREATE`| `UPDATE` |`DELETE`)

`beforeContext`

JSONChaîne contenant les propriétés de la ressource avant la modification. Cette valeur est nulle lors de la création de la ressource. Toutes les valeurs booléennes et numériques de cette JSON chaîne sont. STRINGS

`afterContext`

JSONChaîne contenant les propriétés des ressources si cet ensemble de modifications est exécuté. Cette valeur est nulle lorsque la ressource est supprimée. Toutes les valeurs booléennes et numériques de cette JSON chaîne sont. STRINGS

`lineNumber`

Numéro de ligne du modèle à l'origine de cette modification. Si c'est le cas, `DELETE` cette valeur sera nulle.

`logicalResourceId`

ID de ressource logique de la ressource en cours de modification.

`resourceType`

Type de ressource en cours de modification.

Exemple : Lambda Hook change Set, change d'entrée

Voici un exemple d'entrée de modification d'ensemble de modifications. Dans l'exemple suivant, vous pouvez voir les modifications introduites par l'ensemble de modifications. La première modification consiste à supprimer une file d'attente appelée`CoolQueue`. La deuxième modification consiste à ajouter une nouvelle file d'attente appelée`NewCoolQueue`. La dernière modification est une mise à jour du`DynamoDBTable`.

```
{
    "clientRequestToken": "f8da6d11-b23f-48f4-814c-0fb6a667f50e",
    "awsAccountId": "123456789",
    "stackId": "arn:aws:cloudformation:eu-central-1:123456789:stack/david-ddb-test-
stack/400b40f0-8e72-11ef-80ab-02f2902f0df1",
    "changeSetId": "arn:aws:cloudformation:eu-central-1:123456789:changeSet/davids-
change-set/59ebd63c-7c89-4771-a576-74c3047c15c6",
    "hookTypeName": "my::lambda::changesethook",
    "hookTypeVersion": "00000008",
    "hookModel": {
        "LambdaFunction": "arn:aws:lambda:eu-
central-1:123456789:function:changesethookfunction"
    },
    "actionInvocationPoint": "CREATE_PRE_PROVISION",
    "requestData": {
        "targetName": "CHANGE_SET",
```

```
        "targetType": "CHANGE_SET",
        "targetLogicalId": "arn:aws:cloudformation:eu-central-1:123456789:changeSet/
davids-change-set/59ebd63c-7c89-4771-a576-74c3047c15c6",
        "payload": "https://s3......"
    },
    "requestContext": {
        "invocation": 1,
        "callbackContext": null
    }
}
```

Voici un exemple `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"}]}}'
    }
  ]
}
```

Exemple de fonction Lambda pour les opérations d'ensemble de modifications

L'exemple suivant Lambda Hook cible. `Node.js` Il s'agit d'une fonction simple qui télécharge la charge utile de l'opération d'ensemble de modifications, passe en revue chaque modification, puis imprime les propriétés avant et après avant de renvoyer un`SUCCESS`.

```
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;
};
```

Préparez-vous à créer un crochet Lambda

Avant de créer un Lambda Hook, vous devez remplir les conditions préalables suivantes :

- Vous devez déjà avoir créé une fonction Lambda. Pour plus d'informations, consultez le Création de fonctions Lambda pour les Hooks.
- L'utilisateur ou le rôle qui crée le Hook doit disposer des autorisations suffisantes pour activer les Hooks.
- Pour utiliser le AWS CLI ou an SDK pour créer un hook Lambda, vous devez créer manuellement un rôle d'exécution avec des IAM autorisations et une politique de confiance CloudFormation permettant d'invoquer un hook Lambda.

Création d'un rôle d'exécution pour un Lambda Hook

Un Hook utilise un rôle d'exécution pour les autorisations dont il a besoin pour invoquer ce Hook dans votre Compte AWS.

Ce rôle peut être créé automatiquement si vous créez un Lambda Hook à partir du AWS Management Console ; sinon, vous devez créer ce rôle vous-même.

La section suivante explique comment configurer les autorisations pour créer votre Lambda Hook.

Autorisations nécessaires

Suivez les instructions de la section Créer un rôle à l'aide de politiques de confiance personnalisées dans le Guide de IAM l'utilisateur pour créer un rôle avec une politique de confiance personnalisée.

Effectuez ensuite les étapes suivantes pour configurer vos autorisations :

1. Attachez la politique de privilèges minimaux suivante au IAM rôle que vous souhaitez utiliser pour créer le Lambda Hook.

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

2. Donnez à votre Hook l'autorisation d'assumer le rôle en ajoutant une politique de confiance au rôle. Vous trouverez ci-dessous un exemple de politique de confiance que vous pouvez utiliser.

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

Activez un Lambda Hook dans votre compte

La rubrique suivante explique comment activer un Lambda Hook dans votre compte, afin de le rendre utilisable dans le compte et la région dans lesquels il a été activé.

Rubriques

Activer un Lambda Hook (console)

Pour activer un Lambda Hook à utiliser sur votre compte

1. Connectez-vous à la AWS CloudFormation console AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/cloudformation.
2. Dans la barre de navigation en haut de l'écran, choisissez l' Région AWS endroit où vous souhaitez créer le Hook in.
3. Si vous n'avez pas créé de fonction Lambda pour le Hook, procédez comme suit :

   - Ouvrez la page Functions (Fonctions) sur la console Lambda.
   - Créez la fonction Lambda que vous utiliserez avec ce Hook, puis revenez à cette procédure. Pour de plus amples informations, veuillez consulter Créez des fonctions Lambda pour évaluer les ressources pour les Lambda Hooks.

   Si vous avez déjà créé votre fonction Lambda, passez à l'étape suivante.
4. Pour le nom du crochet, choisissez l'une des options suivantes :

   - Entrez un nom court et descriptif qui sera ajouté par la suite`Private::Lambda::`. Par exemple, si vous entrez*`MyTestHook`*, le nom complet du Hook devient`Private::Lambda::`*`MyTestHook`*.
   - Fournissez le nom complet du Hook (également appelé alias) en utilisant le format suivant : *`Provider`*`::`*`ServiceName`*`::`*`HookName`*
5. Pour la fonction Lambda, indiquez la fonction Lambda à utiliser avec ce Hook. Vous pouvez utiliser :

- Le nom complet de la ressource Amazon (ARN) sans suffixe.
- Un qualifié ARN avec un suffixe de version ou d'alias.

6. Pour les cibles Hook, choisissez les éléments à évaluer :

   - Piles : évalue les modèles de pile lorsque les utilisateurs créent, mettent à jour ou suppriment des piles.
   - Ressources — Évalue les modifications individuelles des ressources lorsque les utilisateurs mettent à jour les piles.
   - Ensembles de modifications : évalue les mises à jour planifiées lorsque les utilisateurs créent des ensembles de modifications.
   - Contrôle du cloud API — Évalue les opérations de création, de mise à jour ou de suppression initiées par le contrôle API du cloud.

7. Pour Actions, choisissez les actions (créer, mettre à jour, supprimer) qui appelleront votre Hook.
8. Pour le mode Hook, choisissez la façon dont le Hook répond lorsque la fonction Lambda invoquée par le Hook renvoie une `FAILED` réponse :

   - Avertir : émet des avertissements à l'intention des utilisateurs, mais autorise la poursuite des actions. Cela est utile pour les validations non critiques ou les contrôles informatifs.
   - Echec — Empêche le déroulement de l'action. Cela est utile pour appliquer des politiques de conformité ou de sécurité strictes.

9. Pour le rôle d'exécution, choisissez le IAM rôle que le Hook assume pour appeler votre fonction Lambda. Vous pouvez soit CloudFormation autoriser la création automatique d'un rôle d'exécution pour vous, soit spécifier un rôle que vous avez créé.
10. Choisissez Suivant.
11. (Facultatif) Pour les filtres Hook, procédez comme suit :

    a. Pour le filtre de ressources, spécifiez les types de ressources qui peuvent appeler le Hook. Cela garantit que le Hook n'est invoqué que pour les ressources pertinentes.
    b. Pour les critères de filtrage, choisissez la logique d'application des filtres de nom de pile et de rôle de pile :

       - Tous les noms de pile et tous les rôles de pile — Le Hook ne sera invoqué que lorsque tous les filtres spécifiés correspondent.
       - Tous les noms de pile et rôles de pile — Le Hook sera invoqué si au moins l'un des filtres spécifiés correspond.

> **Note**
>
> Pour les API opérations Cloud Control, tous les filtres relatifs aux noms de pile et aux rôles de pile sont ignorés.

c. Pour les noms de pile, incluez ou excluez des piles spécifiques des invocations Hook.

- Pour Inclure, spécifiez les noms des piles à inclure. Utilisez-le lorsque vous souhaitez cibler un petit ensemble de piles spécifiques. Seules les piles spécifiées dans cette liste invoqueront le Hook.
- Pour Exclure, spécifiez les noms des piles à exclure. Utilisez-le lorsque vous souhaitez invoquer le Hook sur la plupart des piles, mais en exclure quelques unes en particulier. Toutes les piles, à l'exception de celles répertoriées ici, invoqueront le Hook.

d. Pour les rôles Stack, incluez ou excluez des piles spécifiques des invocations Hook en fonction de leurs rôles associés. IAM

- Pour Inclure, spécifiez un ou plusieurs IAM rôles ARNs pour cibler les piles associées à ces rôles. Seules les opérations de stack initiées par ces rôles invoqueront le Hook.
- Pour Exclure, spécifiez un ou plusieurs IAM rôles ARNs pour les piles que vous souhaitez exclure. Le Hook sera invoqué sur toutes les piles sauf celles initiées par les rôles spécifiés.

12. Choisissez Suivant.
13. Sur la page Vérifier et activer, passez en revue vos choix. Pour apporter des modifications, choisissez Modifier dans la section correspondante.
14. Lorsque vous êtes prêt à continuer, choisissez Activate Hook.

Activer un crochet Lambda ()AWS CLI

Avant de continuer, vérifiez que vous avez créé la fonction Lambda et le rôle d'exécution que vous allez utiliser avec ce Hook. Pour plus d’informations, consultez Créez des fonctions Lambda pour évaluer les ressources pour les Lambda Hooks et Création d'un rôle d'exécution pour un Lambda Hook.

Pour activer un Lambda Hook à utiliser dans votre compte ()AWS CLI

1. Pour commencer à activer un Hook, utilisez ce qui suit activate-typecommande, en remplaçant les espaces réservés par vos valeurs spécifiques. Cette commande autorise le Hook à utiliser un rôle d'exécution spécifié par votre Compte AWS.

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

2. Pour terminer l'activation du Hook, vous devez le configurer à l'aide d'un fichier JSON de configuration.

   Utilisez la cat commande pour créer un JSON fichier avec la structure suivante. Pour de plus amples informations, veuillez consulter Référence syntaxique du schéma de configuration Hook.

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

   - `HookInvocationStatus`: défini sur `ENABLED` pour activer le Hook.
   - `TargetOperations`: Spécifiez les opérations sur lesquelles exécuter le Hook.
   - `FailureMode` : Définissez sur `FAIL` ou `WARN`.
   - `LambdaFunction`: Spécifiez ARN la fonction Lambda.

3. Utilisez ce qui suit set-type-configurationcommande, ainsi que le JSON fichier que vous avez créé, pour appliquer la configuration. Remplacez les espaces réservés par vos valeurs spécifiques.

```
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
```

Ressources connexes

Nous fournissons des exemples de modèles que vous pouvez utiliser pour comprendre comment déclarer un Lambda Hook dans un modèle de CloudFormation pile. Pour plus d'informations, consultez .AWS::CloudFormation::LambdaHook dans le guide de l'utilisateur AWS CloudFormation .

Afficher les journaux des Lambda Hooks dans votre compte

Lorsque vous utilisez un Lambda Hook, le fichier journal de votre rapport de sortie de validation se trouve dans la console Lambda.

Afficher les journaux Lambda Hook dans la console Lambda

Pour consulter le fichier journal de sortie du Lambda Hook

1. Connectez-vous à la console Lambda.
2. Dans la barre de navigation en haut de l'écran, choisissez votre Région AWS.
3. Choisissez Functions.
4. Choisissez la fonction Lambda souhaitée.
5. Choisissez l'onglet Test.
6. Choisissez CloudWatch Logs Live Trail
7. Choisissez le menu déroulant et sélectionnez les groupes de journaux que vous souhaitez consulter.
8. Sélectionnez Démarrer. Le journal s'affichera dans la fenêtre CloudWatch Logs Live Trail. Choisissez Afficher en colonnes ou Afficher en texte brut selon vos préférences.

- Vous pouvez ajouter d'autres filtres aux résultats en les ajoutant dans le champ Ajouter un modèle de filtre. Ce champ vous permet de filtrer les résultats pour n'inclure que les événements correspondant au modèle spécifié.

Pour plus d'informations sur l'affichage des journaux des fonctions Lambda, consultez la section Affichage CloudWatch des journaux des fonctions Lambda.

Supprimer les Lambda Hooks de votre compte

Lorsque vous n'avez plus besoin d'un Lambda Hook activé, suivez les procédures suivantes pour le supprimer de votre compte.

Pour désactiver temporairement un Hook au lieu de le supprimer, consultezDésactiver et activer les AWS CloudFormation Hooks.

Rubriques

- Supprimer un Lambda Hook dans votre compte (console)
- Supprimer un Lambda Hook dans votre compte ()AWS CLI

Supprimer un Lambda Hook dans votre compte (console)

Pour supprimer un Lambda Hook de votre compte

1. Connectez-vous à la AWS CloudFormation console AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/cloudformation.
2. Dans la barre de navigation en haut de l'écran, choisissez l' Région AWS emplacement du crochet.
3. Dans le volet de navigation, choisissez Hooks.
4. Sur la page Hooks, recherchez le hook Lambda que vous souhaitez supprimer.
5. Cochez la case à côté de votre Hook et choisissez Supprimer.
6. Lorsque vous êtes invité à confirmer, saisissez le nom du crochet pour confirmer la suppression du crochet spécifié, puis choisissez Supprimer.

Supprimer un Lambda Hook dans votre compte ()AWS CLI

> **Note**
>
> Avant de pouvoir supprimer le Hook, vous devez d'abord le désactiver. Pour de plus amples informations, veuillez consulter Désactiver et activer un Hook dans votre compte (AWS CLI).

Utilisez ce qui suit deactivate-typecommande pour désactiver un Hook, ce qui le supprime de votre compte. Remplacez les espaces réservés par vos valeurs spécifiques.

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

Développement de Hooks personnalisés à l'aide du CloudFormation CLI

Cette section est destinée aux clients qui souhaitent développer des Hooks personnalisés et les enregistrer dans le AWS CloudFormation registre.

Le développement d'un Hook personnalisé comporte trois étapes principales :

1. Initier

   Pour développer des Hooks personnalisés, vous devez configurer et utiliser le CloudFormation CLI. Pour lancer le projet d'un Hook et ses fichiers requis, utilisez le CloudFormation CLI initcommande et spécifiez que vous souhaitez créer un Hook. Pour de plus amples informations, veuillez consulter Lancer un projet AWS CloudFormation Hooks personnalisé.

2. Modèle

   Pour modéliser, créer et valider votre schéma Hook, définissez le Hook, ses propriétés et ses attributs.

   CloudFormation CLIcrée des fonctions de gestion vides qui correspondent à un point d'invocation Hook spécifique. Ajoutez votre propre logique à ces gestionnaires pour contrôler ce qui se passe lors de votre invocation de Hook à chaque étape du cycle de vie cible. Pour de plus amples informations, veuillez consulter Modélisation de AWS CloudFormation crochets personnalisés.

3. S'inscrire

   Pour enregistrer un Hook, soumettez votre Hook pour qu'il soit enregistré en tant qu'extension tierce privée ou publique. Enregistrez votre Hook lors de l'`submit`opération. Pour de plus amples informations, veuillez consulter Enregistrer un Hook personnalisé avec AWS CloudFormation.

   Les tâches suivantes sont associées à l'enregistrement de votre Hook :

   a. Publier — Les Hooks sont publiés dans le registre.

   b. Configurer — Les hooks sont configurés lorsque la configuration de type est invoquée contre des piles.

   > **Note**
   >
   > Les crochets expirent au bout de 30 secondes.

Les rubriques suivantes vous guident dans le processus de développement, d'enregistrement et de publication de Hooks personnalisés avec Python ou Java.

Rubriques

- Conditions préalables au développement de Hooks personnalisés AWS CloudFormation
- Lancer un projet AWS CloudFormation Hooks personnalisé
- Modélisation de AWS CloudFormation crochets personnalisés
- Enregistrer un Hook personnalisé avec AWS CloudFormation
- Tester un Hook personnalisé dans votre Compte AWS
- Mettre à jour un Hook personnalisé
- Désenregistrer un Hook personnalisé du registre CloudFormation
- Hooks de publication destinés à un usage public
- Référence syntaxique du schéma pour les AWS CloudFormation Hooks

Conditions préalables au développement de Hooks personnalisés AWS CloudFormation

Vous pouvez développer un Hook personnalisé avec Java ou Python. Les conditions requises pour développer des Hooks personnalisés sont les suivantes :

Prérequis pour Java

- Apache Maven
- JDK17

> **Note**
>
> Si vous avez l'intention d'utiliser l'interface de ligne de CloudFormation commande (CLI) pour lancer un projet Hooks pour Java, vous devez également installer Python 3.8 ou version ultérieure. Le plugin Java pour le CloudFormation CLI peut être installé via `pip` (le gestionnaire de paquets de Python), qui est distribué avec Python.

Pour implémenter des gestionnaires Hook pour votre projet Java Hooks, vous pouvez télécharger les fichiers d'exemple des gestionnaires Java Hook.

Prérequis pour Python

- Python version 3.8 ou ultérieure.

Pour implémenter des gestionnaires Hook pour votre projet Python Hooks, vous pouvez télécharger les fichiers d'exemple des gestionnaires Python Hook.

Autorisations pour développer des Hooks

Outre les autorisations CloudFormation `Create`,`Update`, et `Delete` stack, vous devez avoir accès aux AWS CloudFormation opérations suivantes. L'accès à ces opérations est géré par le biais de la CloudFormation politique de votre IAM rôle.

- register-type
- list-types
- deregister-type
- set-type-configuration

Configurer un environnement de développement pour les Hooks

Pour développer des Hooks, vous devez être familiarisé avec les CloudFormation modèles, qu'il s'agisse de Python ou de Java.

Pour installer CloudFormation CLI le et les plugins associés :

1. Installez le CloudFormation CLI with`pip`, le gestionnaire de paquets Python.

```
pip3 install cloudformation-cli
```

2. Installez le plugin Python ou Java pour CloudFormation CLI.

   Python

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

   Java

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

Pour mettre à jour le plugin CloudFormation CLI et le plugin, vous pouvez utiliser l'option de mise à niveau.

Python

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

Java

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

Lancer un projet AWS CloudFormation Hooks personnalisé

La première étape de la création de votre projet Hooks personnalisé consiste à lancer le projet. Vous pouvez utiliser la CloudFormation CLI `init` commande pour lancer votre projet Hooks personnalisé.

La `init` commande lance un assistant qui vous guide tout au long de la configuration du projet, y compris un fichier de schéma Hooks. Utilisez ce fichier de schéma comme point de départ pour définir la forme et la sémantique de vos Hooks. Pour de plus amples informations, veuillez consulter Syntaxe du schéma.

Pour lancer un projet Hook :

1. Créez un répertoire pour le projet.

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

2. Accédez au nouveau répertoire.

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

3. Utilisez la CloudFormation CLI `init` commande pour lancer le projet.

```
cfn init
```

La commande renvoie le résultat suivant.

```
Initializing new project
```

4. La `init` commande lance un assistant qui vous guide tout au long de la configuration du projet. Lorsque vous y êtes invité, entrez h pour spécifier un projet Hooks.

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

```
h
```

5. Entrez un nom pour votre type de Hook.

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

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

6. Si un seul plugin de langue est installé, il est sélectionné par défaut. Si plusieurs plug-ins linguistiques sont installés, vous pouvez choisir la langue de votre choix. Entrez une sélection de numéros pour la langue de votre choix.

```
Select a language for code generation:
[1] java
[2] python38
```

```
[3] python39
(enter an integer):
```

7. Configurez le packaging en fonction du langage de développement choisi.

   Python

   (Facultatif) Choisissez Docker pour un emballage indépendant de la plate-forme. Bien que Docker ne soit pas obligatoire, il est fortement recommandé pour faciliter l'emballage.

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

   Java

   Définissez le nom du package Java et choisissez un modèle de codegen. Vous pouvez utiliser le nom du package par défaut ou en créer un nouveau.

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

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

Résultats : Vous avez lancé le projet avec succès et avez généré les fichiers nécessaires au développement d'un Hook. Voici un exemple des répertoires et des fichiers qui constituent un projet Hooks pour 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**
>
> Les fichiers du `src` répertoire sont créés en fonction de la langue que vous avez sélectionnée. Les fichiers générés contiennent des commentaires et des exemples utiles. Certains fichiers, tels que`models.py`, sont automatiquement mis à jour ultérieurement lorsque vous exécutez la `generate` commande pour ajouter du code d'exécution pour vos gestionnaires.

Modélisation de AWS CloudFormation crochets personnalisés

La modélisation de AWS CloudFormation Hooks personnalisés implique la création d'un schéma qui définit le Hook, ses propriétés et ses attributs. Lorsque vous créez votre projet Hook personnalisé à l'aide de la `cfn init` commande, un exemple de schéma Hook est créé sous la forme d'un fichier texte JSON au format,. *`hook-name`*`.json`

Les points d'invocation cibles et les actions cibles spécifient le point exact où le Hook est invoqué. Les gestionnaires de crochets hébergent une logique personnalisée exécutable pour ces points. Par exemple, une action cible de l'`CREATE`opération utilise un `preCreate` gestionnaire. Votre code écrit dans le gestionnaire sera invoqué lorsque les cibles et les services Hook exécuteront une action correspondante. Les cibles des crochets sont la destination où les crochets sont invoqués. Vous pouvez spécifier des cibles telles que des ressources AWS CloudFormation publiques, des ressources privées ou des ressources personnalisées. Les Hooks prennent en charge un nombre illimité de cibles Hook.

Le schéma contient les autorisations requises pour le Hook. Pour créer le Hook, vous devez spécifier des autorisations pour chaque gestionnaire Hook. CloudFormation encourage les auteurs à rédiger des politiques qui suivent les conseils de sécurité standard consistant à accorder le moindre privilège ou à n'accorder que les autorisations requises pour effectuer une tâche. Déterminez ce que les utilisateurs (et les rôles) doivent faire, puis élaborez des politiques qui leur permettent d'effectuer uniquement ces tâches pour les opérations Hook. CloudFormation utilise ces autorisations pour réduire les autorisations fournies par les utilisateurs de Hook. Ces autorisations sont transmises au Hook. Les gestionnaires de Hook utilisent ces autorisations pour accéder aux AWS ressources.

Vous pouvez utiliser le fichier de schéma suivant comme point de départ pour définir votre Hook. Utilisez le schéma Hook pour spécifier les gestionnaires que vous souhaitez implémenter. Si vous

choisissez de ne pas implémenter un gestionnaire spécifique, supprimez-le de la section des gestionnaires du schéma Hook. Pour plus de détails sur le schéma, consultezSyntaxe du schéma.

```
{
    "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
}
```

Rubriques

- Modélisation de AWS CloudFormation Hooks personnalisés en utilisant Java
- Modélisation de AWS CloudFormation Hooks personnalisés à l'aide de Python

Modélisation de AWS CloudFormation Hooks personnalisés en utilisant Java

La modélisation de AWS CloudFormation Hooks personnalisés implique la création d'un schéma qui définit le Hook, ses propriétés et ses attributs. Ce didacticiel vous explique comment modéliser des Hooks personnalisés à l'aide de Java.

Étape 1 : Ajouter les dépendances du projet

Les projets Hooks basés sur Java s'appuient sur le `pom.xml` fichier de Maven comme dépendance. Développez la section suivante et copiez le code source dans le `pom.xml` fichier situé à la racine du projet.

Dépendances du projet Hook (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>
```

Étape 2 : Générer le package du projet Hook

Générez votre package de projet Hook. CloudFormation CLIcrée des fonctions de gestion vides qui correspondent à des actions Hook spécifiques dans le cycle de vie cible, telles que définies dans la spécification Hook.

```
cfn generate
```

La commande renvoie le résultat suivant.

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

> **Note**
>
> Assurez-vous que vos environnements d'exécution Lambda doivent éviter up-to-date d'utiliser une version obsolète. Pour plus d'informations, consultez la section Mise à jour des environnements d'exécution Lambda pour les types de ressources et les Hooks.

Étape 3 : Ajouter des gestionnaires Hook

Ajoutez votre propre code d'exécution du gestionnaire Hook aux gestionnaires que vous choisissez d'implémenter. Par exemple, vous pouvez ajouter le code suivant pour la journalisation.

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

CloudFormation CLIGénère un vieux objet Java ordinaire (JavaPOJO). Voici des exemples de sortie générés à 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;

}
```

Avec le POJOs fichier généré, vous pouvez désormais écrire les gestionnaires qui implémentent réellement les fonctionnalités du Hook. Pour cet exemple, implémentez le point `preUpdate` d'invocation `preCreate` and pour les gestionnaires.

Étape 4 : Implémenter les gestionnaires Hook

Rubriques

- Codage du générateur de API clients
- Codage de l'auteur API de la demande
- Implémentation du code d'assistance
- Implémentation du gestionnaire de base
- Implémentation du preCreate gestionnaire
- Codage du preCreate gestionnaire
- Mettre à jour le preCreate test

- Implémentation du preUpdate gestionnaire
- Codage du preUpdate gestionnaire
- Mettre à jour le preUpdate test
- Implémentation du preDelete gestionnaire
- Codage du preDelete gestionnaire
- Mettre à jour le preDelete gestionnaire

Codage du générateur de API clients

1. Dans votreIDE, ouvrez le `ClientBuilder.java` fichier situé dans le `src/main/java/com/mycompany/testing/mytesthook` dossier.
2. Remplacez l'intégralité du contenu du `ClientBuilder.java` fichier par le code suivant.

   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();
    }
}
```

Codage de l'auteur API de la demande

1. Dans votreIDE, ouvrez le `Translator.java` fichier situé dans le `src/main/java/com/mycompany/testing/mytesthook` dossier.
2. Remplacez l'intégralité du contenu du `Translator.java` fichier par le code suivant.

   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();
    }
}
```

Implémentation du code d'assistance

1. Dans votreIDE, ouvrez le `AbstractTestBase.java` fichier situé dans le `src/main/java/com/mycompany/testing/mytesthook` dossier.
2. Remplacez l'intégralité du contenu du `AbstractTestBase.java` fichier par le code suivant.

   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
                )
        );
    }
}
```

Implémentation du gestionnaire de base

1. Dans votreIDE, ouvrez le `BaseHookHandlerStd.java` fichier situé dans le `src/main/java/com/mycompany/testing/mytesthook` dossier.
2. Remplacez l'intégralité du contenu du `BaseHookHandlerStd.java` fichier par le code suivant.

   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);
        }
    }
}
```

Implémentation du **`preCreate`** gestionnaire

Le `preCreate` gestionnaire vérifie les paramètres de chiffrement côté serveur pour une ressource ou. `AWS::S3::Bucket` `AWS::SQS::Queue`

- Pour une `AWS::S3::Bucket` ressource, le Hook ne sera accepté que si les conditions suivantes sont vraies :
  - Le chiffrement du compartiment Amazon S3 est défini.
  - La clé du compartiment Amazon S3 est activée pour le compartiment.
  - L'algorithme de chiffrement défini pour le compartiment Amazon S3 est le bon algorithme requis.
  - L'identifiant de la AWS Key Management Service clé est défini.
- Pour une `AWS::SQS::Queue` ressource, le Hook ne sera accepté que si les conditions suivantes sont vraies :
  - L'identifiant de la AWS Key Management Service clé est défini.

Codage du **`preCreate`** gestionnaire

1. Dans votreIDE, ouvrez le `PreCreateHookHandler.java` fichier situé dans le `src/main/java/software/mycompany/testing/mytesthook` dossier.
2. Remplacez l'intégralité du contenu du `PreCreateHookHandler.java` fichier par le code suivant.

```
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();
    }
}
```

Mettre à jour le **`preCreate`** test

1. Dans votreIDE, ouvrez le `PreCreateHandlerTest.java` fichier situé dans le `src/test/java/software/mycompany/testing/mytesthook` dossier.
2. Remplacez l'intégralité du contenu du `PreCreateHandlerTest.java` fichier par le code suivant.

```
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();
    }
}
```

Implémentation du **`preUpdate`** gestionnaire

Implémentez un `preUpdate` gestionnaire, qui démarre avant les opérations de mise à jour pour toutes les cibles spécifiées dans le gestionnaire. Le `preUpdate` gestionnaire effectue les opérations suivantes :

- Pour une `AWS::S3::Bucket` ressource, le Hook ne sera accepté que si les conditions suivantes sont vraies :
  - L'algorithme de chiffrement des compartiments pour un compartiment Amazon S3 n'a pas été modifié.

Codage du **`preUpdate`** gestionnaire

1. Dans votreIDE, ouvrez le `PreUpdateHookHandler.java` fichier situé dans le `src/main/java/software/mycompany/testing/mytesthook` dossier.
2. Remplacez l'intégralité du contenu du `PreUpdateHookHandler.java` fichier par le code suivant.

```
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();
```

```
    }
}
```

Mettre à jour le **`preUpdate`** test

1. Dans votreIDE, ouvrez le `PreUpdateHandlerTest.java` fichier du `src/main/java/com/mycompany/testing/mytesthook` dossier.
2. Remplacez l'intégralité du contenu du `PreUpdateHandlerTest.java` fichier par le code suivant.

```
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();
    }
}
```

Implémentation du **`preDelete`** gestionnaire

Implémentez un `preDelete` gestionnaire, qui démarre avant les opérations de suppression pour toutes les cibles spécifiées dans le gestionnaire. Le `preDelete` gestionnaire effectue les opérations suivantes :

- Pour une `AWS::S3::Bucket` ressource, le Hook ne sera accepté que si les conditions suivantes sont vraies :
  - Vérifie que les ressources minimales requises pour les plaintes existeront dans le compte après la suppression de la ressource.
  - Le montant minimum de ressources requises pour les réclamations est défini dans la configuration de type du Hook.

Codage du **`preDelete`** gestionnaire

1. Dans votreIDE, ouvrez le `PreDeleteHookHandler.java` fichier du `src/main/java/com/mycompany/testing/mytesthook` dossier.
2. Remplacez l'intégralité du contenu du `PreDeleteHookHandler.java` fichier par le code suivant.

```
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);
        }
    }
}
```

Mettre à jour le **`preDelete`** gestionnaire

1. Dans votreIDE, ouvrez le `PreDeleteHookHandler.java` fichier du `src/main/java/com/mycompany/testing/mytesthook` dossier.
2. Remplacez l'intégralité du contenu du `PreDeleteHookHandler.java` fichier par le code suivant.

```
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();
    }
}
```

Modélisation de AWS CloudFormation Hooks personnalisés à l'aide de Python

La modélisation de AWS CloudFormation Hooks personnalisés implique la création d'un schéma qui définit le Hook, ses propriétés et ses attributs. Ce didacticiel vous explique comment modéliser des Hooks personnalisés à l'aide de Python.

Étape 1 : Générer le package du projet Hook

Générez votre package de projet Hook. CloudFormation CLIcrée des fonctions de gestion vides qui correspondent à des actions Hook spécifiques dans le cycle de vie cible, telles que définies dans la spécification Hook.

```
cfn generate
```

La commande renvoie le résultat suivant.

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

> **Note**
>
> Assurez-vous que vos environnements d'exécution Lambda doivent éviter up-to-date d'utiliser une version obsolète. Pour plus d'informations, consultez la section Mise à jour des environnements d'exécution Lambda pour les types de ressources et les Hooks.

Étape 2 : Ajouter des gestionnaires Hook

Ajoutez votre propre code d'exécution du gestionnaire Hook aux gestionnaires que vous choisissez d'implémenter. Par exemple, vous pouvez ajouter le code suivant pour la journalisation.

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

CloudFormation CLIGénère le `src/models.py` fichier à partir duSchéma de configuration.

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
```

Étape 3 : Implémenter les gestionnaires Hook

Avec les classes de données Python générées, vous pouvez écrire les gestionnaires qui implémentent réellement les fonctionnalités du Hook. Dans cet exemple, vous allez implémenter les points `preCreatepreUpdate`, et `preDelete` d'invocation pour les gestionnaires.

Rubriques

- Implémenter le preCreate gestionnaire
- Implémenter le preUpdate gestionnaire
- Implémenter le preDelete gestionnaire
- Implémenter un gestionnaire Hook

Implémenter le preCreate gestionnaire

Le `preCreate` gestionnaire vérifie les paramètres de chiffrement côté serveur pour une ressource ou. `AWS::S3::Bucket` `AWS::SQS::Queue`

- Pour une `AWS::S3::Bucket` ressource, le Hook ne sera accepté que si les conditions suivantes sont vraies.
  - Le chiffrement du compartiment Amazon S3 est défini.
  - La clé du compartiment Amazon S3 est activée pour le compartiment.
  - L'algorithme de chiffrement défini pour le compartiment Amazon S3 est le bon algorithme requis.
  - L'identifiant de la AWS Key Management Service clé est défini.

- Pour une `AWS::SQS::Queue` ressource, le Hook ne sera accepté que si les conditions suivantes sont vraies.
  - L'identifiant de la AWS Key Management Service clé est défini.

Implémenter le preUpdate gestionnaire

Implémentez un `preUpdate` gestionnaire qui démarre avant les opérations de mise à jour pour toutes les cibles spécifiées dans le gestionnaire. Le `preUpdate` gestionnaire effectue les opérations suivantes :

- Pour une `AWS::S3::Bucket` ressource, le Hook ne sera accepté que si les conditions suivantes sont vraies :
  - L'algorithme de chiffrement des compartiments pour un compartiment Amazon S3 n'a pas été modifié.

Implémenter le preDelete gestionnaire

Implémentez un `preDelete` gestionnaire, qui démarre avant les opérations de suppression pour toutes les cibles spécifiées dans le gestionnaire. Le `preDelete` gestionnaire effectue les opérations suivantes :

- Pour une `AWS::S3::Bucket` ressource, le Hook ne sera accepté que si les conditions suivantes sont vraies :
  - Vérifie que les ressources conformes minimales requises existeront dans le compte après la suppression de la ressource.
  - Le montant minimum de ressources conformes requis est défini dans la configuration du Hook.

Implémenter un gestionnaire Hook

1. Dans votreIDE, ouvrez le `handlers.py` fichier situé dans le `src` dossier.
2. Remplacez l'intégralité du contenu du `handlers.py` fichier par le code suivant.

   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}")
```

Passez à la rubrique suivante Enregistrer un Hook personnalisé avec AWS CloudFormation.

Enregistrer un Hook personnalisé avec AWS CloudFormation

Une fois que vous avez créé un Hook personnalisé, vous devez l' AWS CloudFormation enregistrer pour pouvoir l'utiliser. Dans cette section, vous allez apprendre à empaqueter et à enregistrer votre Hook pour l'utiliser dans votre Compte AWS.

Package d'un Hook (Java)

Si vous avez développé votre Hook avec Java, utilisez Maven pour le packager.

Dans le répertoire de votre projet Hook, exécutez la commande suivante pour créer votre Hook, exécuter des tests unitaires et empaqueter votre projet sous forme de `JAR` fichier que vous pouvez utiliser pour soumettre votre Hook au CloudFormation registre.

```
mvn clean package
```

Enregistrer un Hook personnalisé

Pour enregistrer un Hook

1. (Facultatif) Configurez votre Région AWS nom par défaut en soumettant le `us-west-2` configureopération.

   ```
   $ 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. (Facultatif) La commande suivante crée et empaquette votre projet Hook sans l'enregistrer.

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

3. Enregistrez votre Hook en utilisant le CloudFormation CLI submitopération.

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

   Cette commande renvoie la commande suivante.

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

   Résultats : Vous avez enregistré votre Hook avec succès.

Vérifier que les Hooks sont accessibles dans votre compte

Vérifiez que votre Hook est disponible dans votre région Compte AWS et dans les régions auxquelles vous l'avez envoyé.

1. Pour vérifier votre Hook, utilisez le list-typescommande pour répertorier le Hook que vous venez d'enregistrer et en renvoyer une description sommaire.

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

   La commande renvoie le résultat suivant et vous montrera également les Hooks accessibles au public que vous pouvez activer dans votre région Compte AWS et dans votre région.

```
{
    "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. Récupérez le `TypeArn` depuis la `list-type` sortie de votre Hook et enregistrez-le.

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

Pour savoir comment publier des Hooks destinés à un usage public, consultezHooks de publication destinés à un usage public.

Configurer les Hooks

Après avoir développé et enregistré votre Hook, vous pouvez le configurer dans votre Hook en le Compte AWS publiant dans le registre.

- Pour configurer un Hook dans votre compte, utilisez le SetTypeConfigurationopération. Cette opération active les propriétés du Hook définies dans la `properties` section du schéma du Hook. Dans l'exemple suivant, la `minBuckets` propriété est définie sur 1 dans la configuration.

> **Note**
>
> En activant les Hooks dans votre compte, vous autorisez un Hook à utiliser les autorisations définies par votre Compte AWS. CloudFormation supprime les autorisations non requises avant de les transmettre au Hook. CloudFormation recommande aux clients ou aux utilisateurs de Hook de consulter les autorisations Hook et de connaître

les autorisations auxquelles les Hooks sont autorisés avant d'activer Hooks dans votre compte.

Spécifiez les données de configuration de votre extension Hook enregistrée dans le même compte et Région 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

Pour permettre à votre Hook d'inspecter de manière proactive la configuration de votre stack, vous devez définir le `HookInvocationStatus` to `ENABLED` dans la `HookConfiguration` section, une fois le Hook enregistré et activé dans votre compte.

Accès AWS APIs dans les gestionnaires

Si votre Hooks utilise un AWS API dans l'un de ses gestionnaires, le CFN - crée CLI automatiquement un modèle de rôle IAM d'exécution,`hook-role.yaml`. Le `hook-role.yaml` modèle est basé sur les autorisations spécifiées pour chaque gestionnaire dans la section du gestionnaire du schéma Hook. Si le `--role-arn` drapeau n'est pas utilisé pendant le generateopération, le rôle dans cette pile sera provisionné et utilisé comme rôle d'exécution du Hook.

Pour plus d'informations, consultez la section Accès à AWS APIs partir d'un type de ressource.

modèle hook-role.yaml

Note

Si vous choisissez de créer votre propre rôle d'exécution, nous vous recommandons vivement de suivre le principe du moindre privilège en autorisant uniquement le listage `hooks.cloudformation.amazonaws.com` et`resources.cloudformation.amazonaws.com`.

Le modèle suivant utilise IAM les SQS autorisations Amazon S3 et 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
```

Tester un Hook personnalisé dans votre Compte AWS

Maintenant que vous avez codé les fonctions de votre gestionnaire correspondant à un point d'invocation, il est temps de tester votre Hook personnalisé sur une CloudFormation pile.

Le mode d'échec du Hook est défini sur `FAIL` si le CloudFormation modèle n'a pas approvisionné un compartiment S3 avec les éléments suivants :

- Le chiffrement du compartiment Amazon S3 est défini.
- La clé du compartiment Amazon S3 est activée pour le compartiment.
- L'algorithme de chiffrement défini pour le compartiment Amazon S3 est le bon algorithme requis.
- L'identifiant de la AWS Key Management Service clé est défini.

Dans l'exemple suivant, créez un modèle appelé `my-failed-bucket-stack.yml` avec le nom de pile `my-hook-stack` qui échoue à la configuration de la pile et s'arrête avant la mise à disposition des ressources.

Tester les Hooks en provisionnant une pile

Exemple 1 : pour provisionner une pile

Provisionner une pile non conforme

1. Créez un modèle qui spécifie un compartiment S3. Par exemple, `my-failed-bucket-stack.yml`.

   ```
   AWSTemplateFormatVersion: 2010-09-09
   Resources:
     S3Bucket:
       Type: 'AWS::S3::Bucket'
       Properties: {}
   ```

2. Créez une pile et spécifiez votre modèle dans le AWS Command Line Interface (AWS CLI). Dans l'exemple suivant, spécifiez le nom de la pile comme `my-hook-stack` et le nom du modèle comme`my-failed-bucket-stack.yml`.

```
$ aws cloudformation create-stack \
  --stack-name my-hook-stack \
  --template-body file://my-failed-bucket-stack.yml
```

3. (Facultatif) Consultez la progression de votre pile en spécifiant le nom de votre pile. Dans l'exemple suivant, spécifiez le nom de la pile`my-hook-stack`.

```
$ aws cloudformation describe-stack-events \
  --stack-name my-hook-stack
```

Utilisez cette `describe-stack-events` opération pour voir l'échec du Hook lors de la création du bucket. Voici un exemple de sortie de la commande.

```
{
    "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"
        },
    ...
    ]
}
```

Résultats : L'invocation de Hook a échoué dans la configuration de la pile et a empêché le provisionnement de la ressource.

Utiliser un CloudFormation modèle pour réussir la validation Hook

1. Pour créer une pile et passer la validation Hook, mettez à jour le modèle afin que votre ressource utilise un compartiment S3 chiffré. Cet exemple utilise le modèle`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**
>
> Les hooks ne seront pas invoqués pour les ressources ignorées.

2. Créez une pile et spécifiez votre modèle. Dans cet exemple, le nom de la pile est`my-encrypted-bucket-stack`.

```
$ aws cloudformation create-stack \
  --stack-name my-encrypted-bucket-stack \
  --template-body file://my-encrypted-bucket-stack.yml \
```

3. (Facultatif) Consultez la progression de votre pile en spécifiant le nom de la pile.

```
$ aws cloudformation describe-stack-events \
  --stack-name my-encrypted-bucket-stack
```

Utilisez la `describe-stack-events` commande pour afficher la réponse. Voici un exemple de la commande `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"
        },
    ...
    ]
}
```


Résultats : la pile a été créée CloudFormation avec succès. La logique du Hook a vérifié que la `AWS::S3::Bucket` ressource contenait un chiffrement côté serveur avant de la provisionner.

Exemple 2 : pour provisionner une pile

Provisionner une pile non conforme

1. Créez un modèle qui spécifie un compartiment S3. Par exemple, `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. Créez une pile et spécifiez votre modèle dans le AWS CLI. Dans l'exemple suivant, spécifiez le nom de la pile comme `my-hook-stack` et le nom du modèle comme`aes256-bucket.yml`.

```
$ aws cloudformation create-stack \
  --stack-name my-hook-stack \
  --template-body file://aes256-bucket.yml
```

3. (Facultatif) Consultez la progression de votre pile en spécifiant le nom de votre pile. Dans l'exemple suivant, spécifiez le nom de la pile`my-hook-stack`.

```
$ aws cloudformation describe-stack-events \
  --stack-name my-hook-stack
```

   Utilisez cette `describe-stack-events` opération pour voir l'échec du Hook lors de la création du bucket. Voici un exemple de sortie de la commande.

```
{
    "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"
        },
    ...
    ]
}
```

Résultats : L'invocation de Hook a échoué dans la configuration de la pile et a empêché le provisionnement de la ressource. La pile a échoué en raison d'une configuration incorrecte du chiffrement du compartiment S3. La configuration de type Hook est requise `aws:kms` lors de l'utilisation de ce bucket`AES256`.

Utiliser un CloudFormation modèle pour réussir la validation Hook

1. Pour créer une pile et passer la validation Hook, mettez à jour le modèle afin que votre ressource utilise un compartiment S3 chiffré. Cet exemple utilise le modèle`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**
>
> Les hooks ne seront pas invoqués pour les ressources ignorées.

2. Créez une pile et spécifiez votre modèle. Dans cet exemple, le nom de la pile est`my-encrypted-bucket-stack`.

```
$ aws cloudformation create-stack \
  --stack-name my-encrypted-bucket-stack \
  --template-body file://kms-bucket-and-queue.yml
```

3. (Facultatif) Consultez la progression de votre pile en spécifiant le nom de la pile.

```
$ aws cloudformation describe-stack-events \
  --stack-name my-encrypted-bucket-stack
```

Utilisez la `describe-stack-events` commande pour afficher la réponse. Voici un exemple de la commande `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"
        },
    ...
    ]
}
```

Résultats : la pile a été créée CloudFormation avec succès. La logique du Hook a vérifié que la `AWS::S3::Bucket` ressource contenait un chiffrement côté serveur avant de la provisionner.

Mettre à jour un Hook personnalisé

La mise à jour d'un Hook personnalisé permet de rendre les révisions du Hook disponibles dans le CloudFormation registre.

Pour mettre à jour un Hook personnalisé, soumettez vos révisions au CloudFormation registre via le CloudFormation CLI submitopération.

```
$ cfn submit
```

Pour spécifier la version par défaut de votre Hook dans votre compte, utilisez set-type-default-versioncommande et spécifiez le type, le nom du type et l'ID de version.

```
$ aws cloudformation set-type-default-version \
```

```
    --type HOOK \
    --type-name MyCompany::Testing::MyTestHook \
    --version-id 00000003
```

Pour récupérer des informations sur les versions d'un Hook, utilisez list-type-versions.

```
$ aws cloudformation list-type-versions \
  --type HOOK \
  --type-name "MyCompany::Testing::MyTestHook"
```

Désenregistrer un Hook personnalisé du registre CloudFormation

Le désenregistrement d'un Hook personnalisé marque l'extension ou la version de l'extension comme étant DEPRECATED dans le CloudFormation registre, ce qui la met hors service. Une fois obsolète, le Hook personnalisé ne peut pas être utilisé dans une CloudFormation opération.

> **Note**
>
> Avant de désenregistrer le Hook, vous devez désenregistrer individuellement toutes les versions actives précédentes de cette extension. Pour plus d'informations, consultez .DeregisterType.

Pour annuler l'enregistrement d'un Hook, utilisez deregister-typeopération et spécifiez votre HookARN.

```
$ aws cloudformation deregister-type \
    --arn HOOK_TYPE_ARN
```

Cette commande ne produit pas de sortie.

Hooks de publication destinés à un usage public

Pour développer un Hook tiers public, développez votre Hook en tant qu'extension privée. Ensuite, Région AWS dans chaque cas où vous souhaitez rendre l'extension accessible au public :

1. Enregistrez votre Hook en tant qu'extension privée dans le CloudFormation registre.
2. Testez votre Hook pour vous assurer qu'il répond à toutes les exigences nécessaires pour être publié dans le CloudFormation registre.

3. Publiez votre Hook dans le CloudFormation registre.

> **Note**
>
> Avant de publier une extension dans une région donnée, vous devez d'abord vous enregistrer en tant qu'éditeur d'extensions dans cette région. Pour ce faire simultanément dans plusieurs régions, reportez-vous à la section Publication d'extensions dans plusieurs régions StackSets à l'aide du guide de AWS CloudFormation CLI l'utilisateur.

Une fois que vous avez développé et enregistré votre Hook, vous pouvez le rendre accessible au public en CloudFormation le publiant CloudFormation dans le registre, sous la forme d'une extension publique tierce.

Les Hooks tiers publics vous permettent de proposer aux CloudFormation utilisateurs d'inspecter de manière proactive la configuration des AWS ressources avant le provisionnement. Comme pour les Hooks privés, les Hooks publics sont traités de la même manière que tout Hook publié par AWS within CloudFormation.

Les hooks publiés dans le registre sont visibles par tous les CloudFormation utilisateurs dans le registre Régions AWS dans lequel ils sont publiés. Les utilisateurs peuvent ensuite activer votre extension dans leur compte, ce qui la rend disponible pour utilisation dans leurs modèles. Pour plus d'informations, consultez la section Utiliser des extensions publiques tierces depuis le CloudFormation registre dans le Guide de AWS CloudFormation l'utilisateur.

Tester un Hook personnalisé pour un usage public

Afin de publier votre Hook personnalisé enregistré, il doit satisfaire à toutes les exigences de test définies pour celui-ci. Voici une liste des exigences requises avant de publier votre Hook personnalisé en tant qu'extension tierce.

Chaque gestionnaire et chaque cible sont testés deux fois. Une fois pour `SUCCESS` et une fois pour`FAILED`.

- Pour le cas de `SUCCESS` réponse :
  - Le statut doit êtreSUCCESS.
  - Ne doit pas renvoyer de code d'erreur.
  - Le délai de rappel doit être fixé à `0` quelques secondes, s'il est spécifié.

- Pour le cas de `FAILED` réponse :
  - Le statut doit êtreFAILED.
  - Doit renvoyer un code d'erreur.
  - Il doit y avoir un message en réponse.
  - Le délai de rappel doit être fixé à `0` quelques secondes, s'il est spécifié.
- Pour le cas de `IN_PROGRESS` réponse :
  - Ne doit pas renvoyer de code d'erreur.
  - `Result`le champ ne doit pas être défini en réponse.

Spécification des données d'entrée à utiliser dans les tests de contrats

Par défaut, il CloudFormation effectue des tests de contrat en utilisant les propriétés d'entrée générées à partir des modèles que vous définissez dans votre schéma Hook. Cependant, la plupart des Hooks sont suffisamment complexes pour que les propriétés d'entrée permettant de précréer ou de prémettre à jour les piles de provisionnement nécessitent une compréhension de la ressource provisionnée. Pour résoudre ce problème, vous pouvez spécifier l'entrée qu'il CloudFormation utilise lors de ses tests de contrat.

CloudFormation vous propose deux méthodes pour spécifier les données d'entrée à utiliser lors de tests contractuels :

- Remplace le fichier

  L'utilisation d'un `overrides` fichier permet de spécifier des données d'entrée pour certaines propriétés spécifiques CloudFormation à utiliser pendant les tests `preUpdate` et `preCreate` les tests `preDelete` opérationnels.
- Fichiers d'entrée

  Vous pouvez également utiliser plusieurs `input` fichiers pour spécifier les données d'entrée des tests de contrat si :
  - Vous souhaitez ou devez spécifier des données d'entrée différentes pour les opérations de création, de mise à jour et de suppression, ou des données non valides pour les tests.
  - Vous souhaitez spécifier plusieurs ensembles de données d'entrée différents.

Spécification des données d'entrée à l'aide d'un fichier de remplacement

Voici un exemple de données d'entrée de Amazon S3 Hook utilisant le `overrides` fichier.

```
{
    "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"
```

```
            }
        }
    }
}
```

Spécification des données d'entrée à l'aide de fichiers d'entrée

Utilisez `input` des fichiers pour spécifier les différents types de données d'entrée CloudFormation à utiliser : `preCreate` entrée, `preUpdate` entrée et entrée non valide. Chaque type de données est spécifié dans un fichier distinct. Vous pouvez également spécifier plusieurs ensembles de données d'entrée pour les tests de contrat.

Pour spécifier `input` les fichiers CloudFormation à utiliser dans les tests de contrats, ajoutez un `inputs` dossier dans le répertoire racine de votre projet Hooks. Ajoutez ensuite vos fichiers d'entrée.

Spécifiez le type de données d'entrée qu'un fichier contient en utilisant les conventions de dénomination suivantes, où *n*est un entier :

- `inputs_`*n*`_pre_create.json`: utilisez des fichiers dotés de `preCreate` gestionnaires pour spécifier les entrées nécessaires à la création de la ressource.
- `inputs_`*n*`_pre_update.json`: utilisez des fichiers dotés de `preUpdate` gestionnaires pour spécifier les entrées nécessaires à la mise à jour de la ressource.
- `inputs_`*n*`_pre_delete.json`: utilisez des fichiers dotés de `preDelete` gestionnaires pour spécifier les entrées permettant de supprimer la ressource.
- `inputs_`*n*`_invalid.json`: pour spécifier des entrées non valides à tester.

Pour spécifier plusieurs ensembles de données d'entrée pour les tests de contrat, incrémentez le nombre entier dans les noms de fichiers afin de classer vos ensembles de données d'entrée. Par exemple, votre premier ensemble de fichiers d'entrée doit être nommé `inputs_1_pre_create.jsoninputs_1_pre_update.json`, et`inputs_1_pre_invalid.json`. Votre prochain ensemble serait nommé`inputs_2_pre_create.json`, et `inputs_2_pre_update.jsoninputs_2_pre_invalid.json`, et ainsi de suite.

Chaque fichier d'entrée est un JSON fichier contenant uniquement les propriétés des ressources à utiliser lors des tests.

Voici un exemple de répertoire permettant de Amazon S3 spécifier des `inputs` données d'entrée à l'aide de fichiers d'entrée.

inputs_1_pre_create.json

Voici un exemple de test `inputs_1_pre_create.json` contractuel.

```
{
    "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

Voici un exemple de test `inputs_1_pre_update.json` contractuel.

```
{
    "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`

Voici un exemple de test `inputs_1_invalid.json` contractuel.

```
{
    "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`

Voici un exemple de test `inputs_1_invalid_pre_update.json` contractuel.

```
{
    "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"
        }
    }
```

```
}
```

Pour plus d'informations, consultez la section Publication d'extensions pour les rendre accessibles au public dans le Guide de AWS CloudFormation CLI l'utilisateur.

Référence syntaxique du schéma pour les AWS CloudFormation Hooks

Cette section décrit la syntaxe du schéma que vous utilisez pour développer des AWS CloudFormation Hooks.

Un Hook inclut une spécification Hook représentée par un JSON schéma et des gestionnaires Hook. La première étape de la création d'un Hook personnalisé consiste à modéliser un schéma qui définit le Hook, ses propriétés et ses attributs. Lorsque vous initialisez un projet Hook personnalisé à l'aide du CloudFormation CLI initcommande, un fichier de schéma Hook est créé pour vous. Utilisez ce fichier de schéma comme point de départ pour définir la forme et la sémantique de votre Hook personnalisé.

Syntaxe du schéma

Le schéma suivant représente la structure d'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`

Le nom unique de votre Hook. Spécifie un espace de noms en trois parties pour votre Hook, avec un modèle recommandé de. `Organization::Service::Hook`

> **Note**
>
> Les espaces de noms d'organisation suivants sont réservés et ne peuvent pas être utilisés dans les noms de vos types de Hook :
>
> - `Alexa`
> - `AMZN`
> - `Amazon`
> - `ASK`
> - `AWS`

- `Custom`
- `Dev`

Obligatoire : oui

Modèle : `^[a-zA-Z0-9]{2,64}::[a-zA-Z0-9]{2,64}::[a-zA-Z0-9]{2,64}$`

Minimum : `10`

Maximum : `196`

`description`

Brève description du Hook affiché dans la CloudFormation console.

Obligatoire : oui

`sourceUrl`

Le code source URL du Hook, s'il est public.

Obligatoire : non

Maximum : `4096`

`documentationUrl`

Le URL d'une page fournissant une documentation détaillée pour le Hook.

Obligatoire : oui

Modèle : `^https\:\/\/[0-9a-zA-Z]([-.\w]*[0-9a-zA-Z])(\:[0-9]*)*([\?/#].*)?$`

Maximum : `4096`

> **Note**
>
> Bien que le schéma Hook doive inclure des descriptions de propriétés complètes et précises, vous pouvez utiliser la `documentationURL` propriété pour fournir aux utilisateurs plus de détails, notamment des exemples, des cas d'utilisation et d'autres informations détaillées.

`definitions`

Utilisez le `definitions` bloc pour fournir des schémas de propriétés Hook partagés.

Il est considéré comme une bonne pratique d'utiliser cette `definitions` section pour définir des éléments de schéma qui peuvent être utilisés à plusieurs points de votre schéma de type Hook. Vous pouvez ensuite utiliser un JSON pointeur pour référencer cet élément aux endroits appropriés de votre schéma de type Hook.

Obligatoire : non

`typeConfiguration`

Définition des données de configuration d'un Hook.

Obligatoire : oui

`properties`

Les propriétés du Hook. Toutes les propriétés d'un Hook doivent être exprimées dans le schéma. Alignez les propriétés du schéma Hook avec les propriétés de configuration du type Hook.

> **Note**
>
> Les propriétés imbriquées ne sont pas autorisées. Définissez plutôt les propriétés imbriquées de l'`definitions`élément et utilisez un `$ref` pointeur pour les référencer dans la propriété souhaitée.

`additionalProperties`

`additionalProperties` doit être défini sur `false`. Toutes les propriétés d'un Hook doivent être exprimées dans le schéma : les entrées arbitraires ne sont pas autorisées.

Obligatoire : oui

Valeurs valides : `false`

`handlers`

Les gestionnaires spécifient les opérations qui peuvent initier le Hook défini dans le schéma, telles que les points d'invocation du Hook. Par exemple, un `preUpdate` gestionnaire est invoqué avant les opérations de mise à jour pour toutes les cibles spécifiées dans le gestionnaire.

Valeurs valides : `preCreate` | `preUpdate` | `preDelete`

> **Note**
>
> Au moins une valeur doit être spécifiée pour le gestionnaire.

> **Important**
>
> Les opérations de pile qui se traduisent par le statut de `UpdateCleanup` n'invoquent pas de Hook. Par exemple, dans les deux scénarios suivants, le `preDelete` gestionnaire du Hook n'est pas invoqué :
>
> - la pile est mise à jour après la suppression d'une ressource du modèle.
> - une ressource dont le type de mise à jour est remplacé est supprimée.

`targetNames`

Un tableau de chaînes de noms de types que Hook cible. Par exemple, si un `preCreate` gestionnaire a une `AWS::S3::Bucket` cible, le Hook s'exécute pour les compartiments Amazon S3 pendant la phase de préapprovisionnement.

- `TargetName`

  Spécifiez au moins un nom de cible pour chaque gestionnaire implémenté.

  Modèle : `^[a-zA-Z0-9]{2,64}::[a-zA-Z0-9]{2,64}::[a-zA-Z0-9]{2,64}$`

  Minimum : 1

  Obligatoire : oui

  > **Warning**
  >
  > SSM SecureString et les références dynamiques de Secrets Manager ne sont pas résolues avant d'être transmises à Hooks.

`permissions`

Un tableau de chaînes qui spécifie les AWS autorisations nécessaires pour appeler le gestionnaire.

Obligatoire : oui

`additionalProperties`

`additionalProperties` doit être défini sur `false`. Toutes les propriétés d'un Hook doivent être exprimées dans le schéma : les entrées arbitraires ne sont pas autorisées.

Obligatoire : oui

Valeurs valides : `false`

Exemples de schémas Hooks

Exemple 1

Les procédures pas à pas pour Java et Python utilisent l'exemple de code suivant. Voici un exemple de structure pour un Hook appelé`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"
            }
        },
        "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
}
```

Exemple 2

L'exemple suivant est un schéma qui utilise le `STACK` et `CAHNGE_SET` pour `targetNames` pour cibler un modèle de pile et une opération d'ensemble de modifications.

```
{
    "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"
            }
        },
        "required":[
        ],
        "additionalProperties":false
    },
    "handlers":{
        "preCreate":{
            "targetNames":[
                "STACK",
                "CHANGE_SET"
            ],
            "permissions":[
            ]
        },
        "preUpdate":{
            "targetNames":[
                "STACK"
            ],
            "permissions":[
            ]
        },
        "preDelete":{
            "targetNames":[
```

```
                "STACK"
            ],
            "permissions":[

            ]
        }
    },
    "additionalProperties":false
}
```

Désactiver et activer les AWS CloudFormation Hooks

Cette rubrique explique comment désactiver puis réactiver un Hook pour l'empêcher temporairement d'être actif sur votre compte. La désactivation des Hooks peut être utile lorsque vous devez étudier un problème sans ingérence de la part des Hooks.

Désactiver et activer un Hook dans votre compte (console)

Pour désactiver un Hook dans votre compte

1. Connectez-vous à la AWS CloudFormation console AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/cloudformation.
2. Dans la barre de navigation en haut de l'écran, choisissez l' Région AWS emplacement du crochet.
3. Dans le volet de navigation, choisissez Hooks.
4. Choisissez le nom du Hook que vous souhaitez désactiver.
5. Sur la page de détails du Hook, à droite du nom du Hook, cliquez sur le bouton Désactiver.
6. Lorsque vous êtes invité à confirmer, choisissez Disable Hook.

Pour réactiver un Hook précédemment désactivé

1. Connectez-vous à la AWS CloudFormation console AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/cloudformation.
2. Dans la barre de navigation en haut de l'écran, choisissez l' Région AWS emplacement du crochet.
3. Dans le volet de navigation, choisissez Hooks.
4. Choisissez le nom du Hook que vous souhaitez activer.
5. Sur la page de détails du Hook, à droite du nom du Hook, cliquez sur le bouton Activer.
6. Lorsque vous êtes invité à confirmer, choisissez Enable Hook.

Désactiver et activer un Hook dans votre compte (AWS CLI)

> **Important**
>
> Les AWS CLI commandes de désactivation et d'activation des Hooks remplacent l'ensemble de la configuration des Hooks par les valeurs spécifiées dans l'`--configuration`option. Pour éviter toute modification involontaire, vous devez inclure tous les paramètres existants que vous souhaitez conserver lors de l'exécution de ces commandes. Pour afficher les données de configuration actuelles, utilisez describe-typecommande.

Pour désactiver un Hook

Utilisez ce qui suit set-type-configurationcommande et spécifiez `HookInvocationStatus` `DISABLED` comment désactiver le Hook. Remplacez les espaces réservés par vos valeurs spécifiques.

```
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
```

Pour réactiver un Hook précédemment désactivé

Utilisez ce qui suit set-type-configurationcommande et spécifiez `HookInvocationStatus` comment `ENABLED` réactiver le Hook. Remplacez les espaces réservés par vos valeurs spécifiques.

```
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
```

Pour de plus amples informations, veuillez consulter Référence syntaxique du schéma de configuration Hook.

Référence syntaxique du schéma de configuration Hook

Cette section décrit la syntaxe du schéma utilisée pour configurer les Hooks. CloudFormation utilise ce schéma de configuration au moment de l'exécution lors de l'appel d'un Hook dans un Compte AWS.

Pour permettre à votre Hook d'inspecter de manière proactive la configuration de votre stack, définissez le paramètre sur `HookInvocationStatus` une `ENABLED` fois que le Hook a été enregistré et activé dans votre compte.

Rubriques

> **Note**
>
> La quantité maximale de données que la configuration d'un Hook peut stocker est de 300 Ko. Cela s'ajoute à toutes les contraintes imposées au paramètre de `Configuration` demande de SetTypeConfigurationopération.

Propriétés du schéma de configuration des crochets

Le schéma suivant est la structure d'un schéma de configuration Hook.

```
{
    "CloudFormationConfiguration": {
        "HookConfiguration": {
            "HookInvocationStatus": "ENABLED",
            "TargetOperations": "STACK",
            "FailureMode": "FAIL",
            "Properties": {
                ...
```

```
            }
        }
    }
}
```

`HookConfiguration`

La configuration des crochets prend en charge l'activation ou la désactivation des crochets au niveau de la pile, des modes de défaillance et des valeurs des propriétés des crochets.

La configuration Hook prend en charge les propriétés suivantes.

`HookInvocationStatus`

Spécifie si le Hook est `ENABLED` ou`DISABLED`.

Valeurs valides : `ENABLED` | `DISABLED`

`TargetOperations`

Spécifie le type d'opération contre lequel le Hook est exécuté.

Valeurs valides : `STACK` | `RESOURCE` | `CHANGE_SET` | `CLOUD_CONTROL`

`TargetStacks`

Disponible à des fins de rétrocompatibilité. Utilisez *`HookInvocationStatus`* plutôt.

Si le mode est défini sur`ALL`, le Hook s'applique à toutes les piles de votre compte pendant une opération `CREATEUPDATE`, ou une opération sur une `DELETE` ressource.

Si le mode est défini sur`NONE`, le Hook ne s'appliquera pas aux piles de votre compte.

Valeurs valides : `ALL` | `NONE`

`FailureMode`

Ce champ indique au service comment traiter les défaillances de Hook.

- Si le mode est défini sur et que le Hook échoue, la configuration d'échec arrête le provisionnement des ressources et annule la pile. `FAIL`
- Si le mode est défini sur `WARN` et que le Hook échoue, la configuration d'avertissement permet de poursuivre le provisionnement avec un message d'avertissement.

Valeurs valides : `FAIL` | `WARN`

`Properties`

Spécifie les propriétés d'exécution de Hook. Elles doivent correspondre à la forme des propriétés prises en charge par le schéma Hooks.

Exemples de configuration de crochets

Pour des exemples de configuration de Hooks depuis le AWS CLI, consultez les sections suivantes :

- Activer un crochet de protection (AWS CLI)
- Activer un crochet Lambda ()AWS CLI

AWS CloudFormation Filtres de niveau Hooks Stack

Vous pouvez ajouter des filtres de niveau de pile à vos CloudFormation Hooks pour cibler des piles spécifiques en fonction des noms et des rôles des piles. Cela est utile dans les cas où vous avez plusieurs piles avec les mêmes types de ressources, mais que le Hook est destiné à des piles spécifiques.

Cette section explique le fonctionnement de ces filtres et fournit des exemples que vous pouvez suivre.

La structure de base d'une configuration Hook sans filtrage au niveau de la pile ressemble à ceci :

```
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {}
    }
  }
}
```

Pour plus d'informations sur la `HookConfiguration` syntaxe, consultezRéférence syntaxique du schéma de configuration Hook.

Pour utiliser les filtres au niveau de la pile, ajoutez une `StackFilters` clé sous`HookConfiguration`.

La `StackFilters` clé comporte un membre obligatoire et deux membres facultatifs.

- `FilteringCriteria` (obligatoire)
- `StackNames` (facultatif)
- `StackRoles` (facultatif)

Les `StackRoles` propriétés `StackNames` or sont facultatives. Cependant, vous devez spécifier au moins une de ces propriétés.

Si vous créez un Hook qui cible les API opérations de Cloud Control, tous les filtres au niveau de la pile seront ignorés.

`FilteringCriteria`

`FilteringCriteria`est un paramètre obligatoire qui spécifie le comportement de filtrage. Il peut être réglé sur `ALL` ou`ANY`.

- `ALL`invoque le Hook si tous les filtres correspondent.
- `ANY`invoque le Hook si l'un des filtres correspond.

`StackNames`

Pour spécifier un ou plusieurs noms de pile sous forme de filtres dans votre configuration Hooks, utilisez la JSON structure suivante :

```
"StackNames": {
  "Include": [
    "string"
  ],
  "Exclude": [
    "string"
  ]
}
```

Vous devez spécifier l'une des options suivantes :

- `Include`: liste des noms de pile à inclure. Seules les piles spécifiées dans cette liste invoqueront le Hook.
  - Type : tableau de chaînes
  - Nombre maximum d'articles : 50
  - Nombre minimum d'articles : 1
- `Exclude`: liste des noms de pile à exclure. Toutes les piles, à l'exception de celles répertoriées ici, invoqueront le Hook.
  - Type : tableau de chaînes
  - Nombre maximum d'articles : 50
  - Nombre minimum d'articles : 1

Chaque nom de pile dans les `Exclude` tableaux `Include` et doit respecter les exigences de modèle et de longueur suivantes :

- Modèle : `^[a-zA-Z][-a-zA-Z0-9]*$`
- Longueur maximale : 128

`StackRoles`

Pour spécifier un ou plusieurs IAMrôles en tant que filtres dans votre configuration Hook, utilisez la JSON structure suivante :

```
"StackRoles": {
  "Include": [
    "string"
  ],
  "Exclude": [
    "string"
  ]
}
```

Vous devez spécifier l'une des options suivantes :

- `Include`: liste des IAM rôles ARNs à cibler pour les piles associées à ces rôles. Seules les opérations de stack initiées par ces rôles invoqueront le Hook.

  - Type : tableau de chaînes
  - Nombre maximum d'articles : 50
  - Nombre minimum d'articles : 1
- `Exclude`: liste des IAM rôles ARNs pour les piles que vous souhaitez exclure. Le Hook sera invoqué sur toutes les piles sauf celles initiées par les rôles spécifiés.
  - Type : tableau de chaînes
  - Nombre maximum d'articles : 50
  - Nombre minimum d'articles : 1

Chaque rôle de pile dans les `Exclude` tableaux `Include` et doit respecter les exigences de modèle et de longueur suivantes :

- Modèle : `arn:.+:iam::[0-9]{12}:role/.+`
- Longueur maximale : 256

`Include` et **`Exclude`**

Chaque filtre (`StackNamesetStackRoles`) possède une `Include` liste et une `Exclude` liste. À `StackNames` titre d'exemple, le Hook n'est invoqué que sur les piles spécifiées dans la `Include` liste. Si les noms des piles ne sont spécifiés que dans la `Exclude` liste, le hook n'est invoqué que sur les piles qui ne figurent pas dans la `Exclude` liste. Si `Include` les deux `Exclude` sont spécifiés, le Hook cible le contenu de la `Include` liste et non le contenu de la `Exclude` liste.

Supposons, par exemple, que vous ayez quatre piles : A, B, C et D.

- `"Include": ["A","B"]`Le Hook est invoqué sur A et B.
- `"Exclude": ["B"]`Le Hook est invoqué sur A, C et D.
- `"Include": ["A","B","C"], "Exclude": ["A","D"]`Le Hook est invoqué sur B et C.
- `"Include": ["A","B","C"], "Exclude": ["A”,"B","C"]`Le Hook n'est invoqué sur aucune pile.

Exemples de filtres au niveau de la pile

Cette section fournit des exemples que vous pouvez suivre pour créer des filtres au niveau de la pile pour les AWS CloudFormation Hooks.

Exemple 1 : inclure des piles spécifiques

L'exemple suivant indique une `Include` liste. Le Hook n'est invoqué que sur les piles nomméesstack-test-1, stack-test-2 etstack-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"
          ]
        }
      }
    }
  }
}
```

Exemple 2 : Exclure des piles spécifiques

Si les noms des piles sont plutôt ajoutés à la `Exclude` liste, le Hook est invoqué sur toute pile non nomméestack-test-1, stack-test-2 oustack-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"
          ]
        }
      }
    }
  }
}
```

Exemple 3 : combinaison d'inclusion et d'exclusion

Si `Include` aucune `Exclude` liste n'est spécifiée, le Hook n'est invoqué que sur les piles `Include` qui ne figurent pas dans la `Exclude` liste. Dans l'exemple suivant, le Hook n'est invoqué que sur`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"
```

```
          ]
        }
      }
    }
  }
}
```

Exemple 4 : combinaison de noms de pile et de rôles avec des **ALL** critères

Le Hook suivant inclut trois noms de pile et un rôle de pile. Comme le Hook `FilteringCriteria` est spécifié comme`ALL`, le Hook n'est invoqué que pour les piles qui ont à la fois un nom de pile correspondant et le rôle de pile correspondant.

```
{
  "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"]
        }
      }
    }
  }
}
```

Exemple 5 : combinaison de noms de pile et de rôles avec des **ANY** critères

Le Hook suivant inclut trois noms de pile et un rôle de pile. Comme le Hook `FilteringCriteria` est spécifié commeANY, le Hook est invoqué pour les piles qui ont soit un nom de pile correspondant, soit le rôle de pile correspondant.

```
{
  "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 Filtres cibles Hooks

Cette rubrique fournit des conseils sur la configuration des filtres cibles pour les AWS CloudFormation Hooks. Vous pouvez utiliser des filtres cibles pour contrôler plus précisément quand et sur quelles ressources votre Hook est invoqué. Vous pouvez configurer des filtres allant du simple ciblage des types de ressources à des combinaisons plus complexes de types de ressources, d'actions et de points d'invocation.

Pour spécifier un ou plusieurs noms de pile sous forme de filtres dans votre configuration Hooks, ajoutez une `TargetFilters` clé sous`HookConfiguration`.

`TargetFilters`prend en charge les propriétés suivantes.

`TargetNames`

Tableau de chaînes qui indique les noms des types de ressources à cibler. Pour obtenir un exemple, consultez Exemple 1 : filtre cible de base.

Les noms de cibles prennent en charge les noms de cibles concrets et la correspondance complète des caractères génériques. Pour de plus amples informations, veuillez consulter Utilisation de caractères génériques avec les noms de cibles Hook.

Modèle : `^[a-zA-Z0-9]{2,64}::[a-zA-Z0-9]{2,64}::[a-zA-Z0-9]{2,64}$`

Maximum : `50`

`Actions`

Tableau de chaînes qui spécifie les actions cibles pour les cibles que vous listez dans la liste`TargetNames`.

Valeurs valides : `CREATE` | `UPDATE` | `DELETE`

> **Note**
>
> Lorsque vous utilisez `RESOURCE` les cibles`STACK`, et `CLOUD_CONTROL` Hook, toutes les actions cibles sont applicables. Lorsque vous utilisez des cibles `CHANGE_SET` Hook, seule l'`CREATE`action est applicable.

`InvocationPoints`

Un tableau de chaînes qui indique les points d'appel pour les cibles que vous listez. `TargetNames`

Valeurs valides : `PRE_PROVISION`

`Targets`

Tableau d'objets qui indique la liste des cibles à utiliser pour le filtrage des cibles.

Chaque cible du tableau de cibles possède les propriétés suivantes.

`TargetNames`

Nom du type de ressource à cibler.

`Actions`

Action pour la cible spécifiée.

Valeurs valides : `CREATE` | `UPDATE` | `DELETE`

`InvocationPoints`

Point d'invocation pour la cible spécifiée.

Valeurs valides : `PRE_PROVISION`

> **Note**
>
> Vous ne pouvez pas inclure à la fois le tableau `TargetNames` d'`Targets`objets et `InvocationPoints` les tableaux`Actions`, ou. Si vous souhaitez utiliser ces trois éléments`Targets`, vous devez les inclure dans le tableau `Targets` d'objets. Pour obtenir un exemple, consultez Exemple 2 : Utilisation du tableau `Targets` d'objets.

Exemples de filtres cibles

Cette section fournit des exemples que vous pouvez suivre pour créer des filtres cibles pour les AWS CloudFormation Hooks.

Exemple 1 : filtre cible de base

Pour créer un filtre cible de base axé sur des types de ressources spécifiques, utilisez l'`TargetFilters`objet avec le `TargetNames` tableau. La configuration de filtre cible suivante invoquera le Hook pour toutes les opérations sur les buckets S3 et les tables DynamoDB.

```
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
```

```
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "TargetFilters": {
        "TargetNames": [
           "AWS::S3::Bucket",
           "AWS::DynamoDB::Table"
        ]
      }
    }
  }
}
```

Exemple 2 : Utilisation du tableau **`Targets`** d'objets

Pour des filtres plus avancés, vous pouvez utiliser le tableau d'`Targets`objets pour répertorier des combinaisons spécifiques de cibles, d'actions et de points d'invocation. La configuration de filtre cible suivante invoquera le Hook avant `CREATE` les `UPDATE` opérations sur les buckets S3 et les tables DynamoDB.

```
{
  "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"
         }
       ]
     }
   }
  }
}
```

Utilisation de caractères génériques avec les noms de cibles Hook

Vous pouvez utiliser des caractères génériques dans le nom de la cible. Vous pouvez utiliser des caractères génériques (*et?) dans les noms de vos cibles Hook. L'astérisque (*) représente n'importe quelle combinaison de caractères. Le point d'interrogation (?) représente n'importe quel caractère. Vous pouvez utiliser plusieurs ? caractères * et dans le nom d'une cible.

Example : exemples de caractères génériques pour le nom de la cible dans les schémas Hook

L'exemple suivant cible tous les types de ressources pris en charge par Amazon S3.

```
{
...
    "handlers": {
       "preCreate": {
            "targetNames": [
                "AWS::S3::*"
            ],
            "permissions": []
       }
    }
...
}
```

L'exemple suivant correspond à tous les types de ressources qui ont »Bucket« dans le nom.

```
{
...
    "handlers": {
       "preCreate": {
            "targetNames": [
                "AWS::*::Bucket*"
            ],
            "permissions": []
       }
    }
...
}
```

`AWS::*::Bucket*`Cela peut concerner l'un des types de ressources concrets suivants :

- `AWS::Lightsail::Bucket`
- `AWS::S3::Bucket`
- `AWS::S3::BucketPolicy`
- `AWS::S3Outpost::Bucket`
- `AWS::S3Outpost::BucketPolicy`

Example : exemples de caractères génériques pour le nom de la cible dans les schémas de configuration Hook

L'exemple de configuration suivant invoque le Hook pour les `CREATE` opérations sur tous les types de ressources Amazon S3 et pour les `UPDATE` opérations sur tous les types de ressources de table nommés, tels que `AWS::DynamobDB::Table` ou`AWS::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"
                    }
                ]
            }
        }
   }
}
```

L'exemple de configuration suivant invoque le Hook `CREATE` et les `UPDATE` opérations sur tous les types de ressources Amazon S3, ainsi que les `UPDATE` opérations `CREATE` et les opérations sur tous les types de ressources de table nommés, tels que `AWS::DynamobDB::Table` ou`AWS::Glue::Table`.

```
{
   "CloudFormationConfiguration": {
        "HookConfiguration": {
            "TargetStacks": "ALL",
            "FailureMode": "FAIL",
            "Properties": {},
            "TargetFilters":{
                "TargetNames": [
                    "AWS::S3::*",
                    "AWS::*::Table"
                ],
                "Actions": [
                    "CREATE",
                    "UPDATE"
                ],
                "InvocationPoints": [
                    "PRE_PROVISION"
                ]
            }
        }
   }
}
```

Créez des Hooks à l'aide CloudFormation de modèles

Cette page fournit des liens vers des exemples de CloudFormation modèles et des rubriques de référence techniques pour les Hooks.

En utilisant CloudFormation des modèles pour créer des Hooks, vous pouvez réutiliser votre modèle pour configurer vos Hooks de manière cohérente et répétée. Cette approche vous permet de définir vos Hooks une seule fois, puis de configurer les mêmes Hooks encore et encore dans plusieurs Comptes AWS régions.

CloudFormation propose les types de ressources spécialisées suivants pour la création de crochets Guard et Lambda.

Tâche	Solution	Liens
Créez un crochet de protection	Utilisez le type de `AWS::CloudFormation::GuardHook` ressource pour créer et activer un Guard Hook.	Exemple de modèle Référence technique
Création d'un crochet Lambda	Utilisez le type de `AWS::CloudFormation::LambdaHook` ressource pour créer et activer un Lambda Hook.	Exemple de modèle Référence technique

CloudFormation propose également les types de ressources suivants que vous pouvez utiliser dans vos modèles de pile pour créer des Hooks personnalisés.

Tâche	Solution	Liens
Définissez la version par défaut du Hook	Utilisez le type de `AWS::CloudFormation::HookDefaultVersion` ressource pour spécifier la version par défaut d'un Hook personnalisé.	Exemples de modèle Référence technique

Tâche	Solution	Liens
Définissez la configuration du Hook	Utilisez le type de `AWS::CloudFormation::HookTypeConfig` ressource pour spécifier la configuration d'un Hook personnalisé.	Exemples de modèle Référence technique
Enregistrer un Hook	Utilisez le type de `AWS::CloudFormation::HookVersion` ressource pour publier une nouvelle version ou une première version d'un Hook personnalisé CloudFormation dans le registre.	Exemples de modèle Référence technique
Publier un Hook publiquement	Utilisez le type de `AWS::CloudFormation::PublicTypeVersion` ressource pour tester et publier un Hook personnalisé enregistré en tant que Hook tiers public.	Référence technique
Enregistrez votre compte en tant qu'éditeur	Utilisez le type de `AWS::CloudFormation::Publisher` ressource pour enregistrer votre compte en tant qu'éditeur d'extensions publiques (Hooks, modules et types de ressources) dans le CloudFormation registre.	Référence technique
Activer des Hooks publics et tiers	Le type de `AWS::CloudFormation::TypeActivation` ressource fonctionne avec le type de `AWS::CloudFormation::HookTypeConfig` ressource pour activer un Hook personnalisé public tiers dans votre compte.	Référence technique

Historique du document pour le guide de l'utilisateur de AWS CloudFormation Hooks

Le tableau suivant décrit les modifications importantes apportées à la documentation depuis la dernière version de AWS CloudFormation Hooks. Pour être informé des mises à jour de cette documentation, vous pouvez vous abonner à un RSS flux.

- Dernière mise à jour de la documentation : 8 décembre 2023.

Modification	Description	Date
Crochets empilables	Les Hooks sont désormais pris en charge au niveau de la pile, ce qui permet aux clients d'utiliser les CloudFormation Hooks pour évaluer de nouveaux modèles et éventuellement empêcher les opérations de pile de se poursuivre.	13 novembre 2024
AWS Cloud Control API Intégration de Hooks	Les Hooks sont désormais intégrés à Cloud ControlAPI, ce qui permet aux clients d'utiliser CloudFormation des Hooks pour inspecter de manière proactive la configuration des ressources avant le provisionnement. Si des ressources non conformes sont détectées, le Hook échoue à l'opération et empêche le provisionnement des ressources, ou émet un avertissement et autorise la	13 novembre 2024

	poursuite de l'opération de provisionnement.	
AWS CloudFormation Guard Crochets	AWS CloudFormation Guard est un langage open source et à usage général spécifiqu e à un domaine (DSL) que vous pouvez utiliser pour créer. policy-as-code Guard Hooks peut évaluer le contrôle API et les CloudFormation opérations du cloud afin d'inspecter la configuration des ressources avant le provision nement. Si des ressources non conformes sont détectées, le Hook échoue à l'opération et empêche le provisionnement des ressources, ou émet un avertissement et autorise la poursuite de l'opération de provisionnement.	13 novembre 2024
AWS Lambda Crochets	AWS CloudFormation Les Lambda Hooks vous permettent d'évaluer CloudFormation et de contrôler les API opérations dans le cloud par rapport à votre propre code personnal isé. Votre Hook peut bloquer le déroulement d'une opération ou envoyer un avertissement à l'appelant et autoriser le déroulement de l'opération.	13 novembre 2024

Guide de l'utilisateur de Hooks	Version initiale du guide de l'utilisateur de AWS CloudFormation Hooks. Les mises à jour incluent une nouvelle introduction, une procédure pas à pas, des concepts et une terminologie, un filtrage au niveau de la pile et des rubriques mises à jour sur les prérequis, la configuration et le développement des CloudFormation Hooks.	8 décembre 2023

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.