亚马逊云科技

Hooks User Guide

Amazon CloudFormation

Amazon CloudFormation: Hooks User Guide

Table of Contents

What are Amazon CloudFormation Hooks? 1
Hook implementation options 1
Amazon Control Tower proactive controls 1
Guard rules 1
Lambda functions 2
Custom Hooks 2
Creating and managing Hooks 3
Concepts 5
Hook 5
Failure mode 6
Hook targets 6
Target actions 7
Annotations 7
Hook handler 7
Timeout and retry limits 8
Proactive controls as Hooks 8
Amazon CLI commands for working with Hooks 9
Activate a proactive control-based Hook 9
Delete proactive control-based Hooks 13
Guard Hooks 14
Amazon CLI commands for working with Guard Hooks 15
Write Guard rules for Hooks 15
Prepare to create a Guard Hook 29
Activate a Guard Hook 31
View logs for Guard Hooks 36
Delete Guard Hooks 37
Lambda Hooks 38
Amazon CLI commands for working with Lambda Hooks 38
Create Lambda functions for Hooks 39
Prepare to create a Lambda Hook 63
Activate a Lambda Hook 65
View logs for Lambda Hooks 69
Delete Lambda Hooks 70
Custom Hooks 71

Prerequisites ... 72
Initiating a Hooks project ... 74
Modeling Hooks ... 77
Registering Hooks ... 145
Testing Hooks ... 149
Updating Hooks ... 159
Deregistering Hooks ... 159
Publishing Hooks ... 160
Schema syntax ... 168
Disable-enable Hooks ... 177
Disable and enable a Hook (console) ... 177
Disable and enable a Hook (Amazon CLI) ... 178
View Hook invocation results ... 179
View invocation results (console) ... 179
View results for all Hooks ... 179
View invocation history for individual Hooks ... 180
View results for stack-specific invocations ... 180
View invocation results (Amazon CLI) ... 181
Configuration schema ... 186
Hook configuration schema properties ... 186
Hook configuration examples ... 188
Stack level filters ... 188
`FilteringCriteria` ... 189
`StackNames` ... 190
`StackRoles` ... 191
`Include` and `Exclude` ... 192
Examples of stack level filters ... 192
Target filters ... 196
Examples of target filters ... 198
Using wildcards ... 200
Create Hooks using CloudFormation templates ... 209
Grant IAM permissions ... 211
Allow a user to manage Hooks ... 212
Allow a user to publish custom Hooks publicly ... 213
Allow a user to view Hook invocation results ... 213
List Hook invocation results ... 214

Allow a user to view detailed Hook invocation results 216
Amazon KMS key policy and permissions 217
Overview 218
Encryption context 219
Customer managed KMS key policy 219
KMS permissions for `SetTypeConfiguration` API 222
KMS permissions for `GetHookResult` API 223
Document history 225

What are Amazon CloudFormation Hooks?

Amazon CloudFormation Hooks is a feature that helps ensure that your CloudFormation resources, stacks, and change sets comply with your organization's security, operational, and cost optimization best practices. CloudFormation Hooks can also ensure this same level of compliance for your Amazon Cloud Control API resources. With CloudFormation Hooks, you can provide code that proactively inspects the configuration of your Amazon resources before provisioning. If non-compliant resources are found, Amazon CloudFormation either fails the operation and prevents the resources from being provisioned or emits a warning and allows the provisioning operation to continue.

You can use Hooks to enforce a variety of requirements and guidelines. For example, a security-related Hook can verify that security groups have appropriate inbound and outbound traffic rules for your Amazon VPC. A cost-related Hook can restrict development environments to use only smaller Amazon EC2 instance types. A Hook designed for data availability can enforce automatic backups for Amazon RDS.

Hook implementation options

CloudFormation provides multiple options for implementing Hooks, giving you flexibility to choose the approach that best suits your needs.

Amazon Control Tower proactive controls

The Amazon Control Tower Control Catalog offers standardized proactive controls that you can implement as Hooks. This approach saves setup time and helps you validate resource configurations against Amazon best practices across your organization without writing code.

Guard rules

Amazon CloudFormation Guard is a policy-as-code evaluation tool that provides a domain-specific language for writing custom evaluation logic for Hooks. This approach allows you to define compliance checks using Guard's declarative syntax, making it easy to create and maintain your evaluation logic without extensive programming knowledge.

Lambda functions

You can also implement Hooks using Lambda functions, allowing you to leverage the full power and flexibility of Lambda for your evaluation logic. You can use any Lambda-supported runtime language and integrate with other Amazon services as needed.

Custom Hooks

For advanced use cases, you can write your own evaluation logic using programming languages supported by the CloudFormation CLI. This approach provides maximum flexibility for implementing organization-specific governance requirements. As a supported extension type in the Amazon CloudFormation registry, your custom Hooks can be distributed and activated both publicly and privately.

Creating and managing Amazon CloudFormation Hooks

Amazon CloudFormation Hooks provide a mechanism to evaluate your CloudFormation resources before allowing stack creation, modification, or deletion. This feature helps you ensure that your CloudFormation resources comply with your organization's security, operational, and cost optimization best practices.

To create a Hook, you have four options.

- **Proactive controls as Hooks** – Evaluates resources using proactive controls from the Amazon Control Tower Control Catalog.
- **Guard Hook** – Evaluates resources using an Amazon CloudFormation Guard rule.
- **Lambda Hook** – Forwards requests for resource evaluation to an Amazon Lambda function.
- **Custom Hook** – Uses a custom Hook handler that you manually develop.

Proactive controls as Hooks

To create a Hook from proactive controls, follow these steps:

1. Navigate to the CloudFormation console and begin creating a Hook.
2. Choose specific controls from the Control Catalog that you want your Hook to evaluate resources against.

   These controls will automatically apply whenever specified resources are created or updated. Your selection determines which resource types the Hook will evaluate.
3. Set the Hook mode to either warn users about non-compliance or prevent non-compliant operations.
4. Configure optional filters to include or exclude stacks by stack name or stack role.
5. After completing the configuration, activate the Hook to begin enforcement.

Guard Hook

To create a Guard Hook, follow these steps:

1. Write your resource evaluation logic as a Guard policy rule using the Guard domain-specific language (DSL).

2. Store the Guard policy rule in an Amazon S3 bucket.
3. Navigate to the CloudFormation console and begin creating a Guard Hook.
4. Provide the Amazon S3 path to your Guard rule.
5. Choose the specific target types that the Hook will evaluate.
   - CloudFormation resources (`RESOURCE`)
   - Entire stack templates (`STACK`)
   - Change sets (`CHANGE_SET`)
   - Cloud Control API resources (`CLOUD_CONTROL`)
6. Choose the deployment actions (create, update, delete) that will invoke your Hook.
7. Choose how the Hook responds when it fails evaluation.
8. Configure optional filters to specify which resource types the Hook should evaluate
9. Configure optional filters to include or exclude stacks by stack name or stack role.
10. After completing the configuration, activate the Hook to begin enforcement.

Lambda Hook

To create a Lambda Hook, follow these steps:

1. Write your resource evaluation logic as a Lambda function.
2. Navigate to the CloudFormation console and begin creating a Lambda Hook.
3. Provide the Amazon Resource Name (ARN) for your Lambda function.
4. Choose the specific target types that the Hook will evaluate.
   - CloudFormation resources (`RESOURCE`)
   - Entire stack templates (`STACK`)
   - Change sets (`CHANGE_SET`)
   - Cloud Control API resources (`CLOUD_CONTROL`)
5. Choose the deployment actions (create, update, delete) that will invoke your Hook.
6. Choose how the Hook responds when it fails evaluation.
7. Configure optional filters to specify which resource types the Hook should evaluate
8. Configure optional filters to include or exclude stacks by stack name or stack role.

9. After completing the configuration, activate the Hook to begin enforcement.

Custom Hook

Custom Hooks are extensions that you register in the CloudFormation registry using the CloudFormation Command Line Interface (CFN-CLI).

To create a custom Hook, follow these main steps:

1. **Initiate the project** – Generate the files needed to develop a custom Hook.
2. **Model the Hook** – Write a schema that defines the Hook and the handlers that specify the operations that can invoke the Hook.
3. **Register and activate the Hook** – After you have created a Hook, you need to register it in the account and Region where you want to use it and this activates it.

The following topics provide more information for creating and managing Hooks.

Topics

- Amazon CloudFormation Hooks concepts
- Amazon Control Tower proactive controls as Hooks
- Guard Hooks
- Lambda Hooks
- Developing custom Hooks using the CloudFormation CLI

Amazon CloudFormation Hooks concepts

The following terminology and concepts are central to your understanding and use of Amazon CloudFormation Hooks.

Hook

A Hook contains code that is invoked immediately before CloudFormation creates, updates, or deletes stacks or specific resources. It can also be invoked during a create change set operation. Hooks can inspect the template, resources, or change set that CloudFormation is about to provision. Additionally, Hooks can be invoked immediately before the Cloud Control API creates, update, or deletes specific resources.

If a Hook identifies any configurations that don't comply with the organizational guidelines defined in your Hook logic, then you may choose to either `WARN` users or `FAIL`, preventing CloudFormation from provisioning the resource.

Hooks have the following characteristics:

- **Proactive validation** – Reduces risk, operational overhead, and cost by identifying non-compliant resources before they're created, updated, or deleted.
- **Automatic enforcement** – Provides enforcement in your Amazon Web Services account to prevent non-compliant resources from being provisioned by CloudFormation.

Failure mode

Your Hook logic can return success or failure. A success response will allow the operation to continue. A failure for non-compliant resources can result in the following:

- `FAIL` – Stops provisioning operation.
- `WARN` – Allows provisioning to continue with a warning message.

Creating Hooks in `WARN` mode is an effective way to monitor Hook behavior without affecting stack operations. First, activate Hooks in `WARN` mode to understand which operations will be impacted. After you have assessed the potential effects, you can switch the Hook to `FAIL` mode to start preventing non-compliant operations.

Hook targets

Hook targets specify the operations that a Hook will evaluate. These can be operations on:

- Resources supported by CloudFormation (`RESOURCE`)
- Stack templates (`STACK`)
- Change sets (`CHANGE_SET`)
- Resources supported by the Cloud Control API (`CLOUD_CONTROL`)

You define one or more targets that specify the broadest operations that the Hook will evaluate. For example, you can author a Hook targeting `RESOURCE` to target all Amazon resources and `STACK` to target all stack templates.

Target actions

Target actions define the specific actions (`CREATE`, `UPDATE`, or `DELETE`) that will invoke a Hook. For `RESOURCE`, `STACK`, and `CLOUD_CONTROL` targets, all target actions are applicable. For `CHANGE_SET` targets, only the `CREATE` action is applicable.

Annotations

GetHookResult responses can return *annotations* that provide detailed compliance check results and remediation guidance for each evaluated resource. For details on the API's annotation structure, see Annotation in the *Amazon CloudFormation API Reference*. For instructions on viewing these validation results, see View invocation results for Amazon CloudFormation Hooks.

You can encrypt annotations as needed for sensitive compliance information by specifying your own KMS key when configuring the Hook. For more information, see Hook configuration schema syntax reference. For information about setting up the key policy that you need when you specify your KMS key for Hooks, see Amazon KMS key policy and permissions for encrypting Amazon CloudFormation Hooks results at rest.

> **Important**
>
> Note that the `KmsKeyId` option to specify a customer managed key is currently only available when you use the Amazon CLI to configure your Hook.

Hook handler

For custom Hooks, this is the code that handles evaluation. It is associated with a target invocation point and a target action that mark an exact point where a Hook runs. You write handlers that host logic for these specific points. For example, a `PRE` target invocation point with `CREATE` target action makes a `preCreate` Hook handler. Code within the Hook handler runs when a matching target invocation point and service are performing an associated target action.

Valid values: (`preCreate` | `preUpdate` | `preDelete`)

Important

Stack operations that result in the status of `UpdateCleanup` do not invoke a Hook. For example, during the following two scenarios, the Hook's `preDelete` handler is not invoked:

- the stack is updated after removing one resource from the template.
- a resource with the update type of replacement is deleted.

Timeout and retry limits

Hooks have a 30-second timeout limit per invocation and are limited to 3 retry attempts. If an invocation exceeds the timeout, we return an error message stating that the Hook execution timed out. After the third retry, CloudFormation marks the Hook execution as failed.

Amazon Control Tower proactive controls as Hooks

The Amazon Control Tower Control Catalog provides pre-built compliance rules (proactive controls) that you can implement as Hooks. This approach saves setup time and helps you validate resource configurations against Amazon best practices across your organization without writing code.

Proactive controls evaluate Amazon resources before deployment, preventing non-compliant resources from being created rather than detecting issues later. They check configurations against established security, operational, and governance standards.

To get started, simply activate proactive control-based Hooks in your desired account and Region. These Hooks will then evaluate specific target types to ensure compliance with your selected controls.

For more information about available proactive controls, see the Amazon Control Tower Control Catalog.

Topics

- Amazon CLI commands for working with Hooks
- Activate a proactive control-based Hook in your account
- Delete proactive control-based Hooks in your account

Amazon CLI commands for working with Hooks

The Amazon CLI commands for working with proactive control-based Hooks include:

- activate-type to start the activation process for a proactive control-based Hook.
- set-type-configuration to specify the controls to apply to a proactive control-based Hook in your account.
- list-types to list the Hooks in your account.
- describe-type to return detailed information about a specific Hook or specific Hook version, including current configuration data.
- deactivate-type to remove a previously activated Hook from your account.

Activate a proactive control-based Hook in your account

The following topic shows you how to activate a proactive control-based Hook in your account, which makes it usable in the account and Region it was activated in.

> **Important**
>
> Before you continue, verify that you have the permissions required to work with Hooks and view proactive controls from the CloudFormation console. For more information, see Grant IAM permissions for CloudFormation Hooks.

Topics

- Activate a proactive control-based Hook (console)
- Activate a proactive control-based Hook (Amazon CLI)

Activate a proactive control-based Hook (console)

To activate a proactive control-based Hook for use in your account

1. Sign in to the Amazon Web Services Management Console and open the Amazon CloudFormation console at https://console.amazonaws.cn/cloudformation.
2. On the navigation bar at the top of the screen, choose the Amazon Web Services Region where you want to create the Hook in.

3. In the navigation pane on the left, choose **Hooks**.
4. On the **Hooks** page, choose **Create a Hook**, and then choose **With the Control Catalog**.
5. On the **Select controls** page, for **Proactive controls**, select one or more proactive controls to use.

   These controls will automatically apply whenever specified resources are created or updated. Your selection determines which resource types the Hook will evaluate.
6. Choose **Next**.
7. For **Hook name**, choose one of the following options:

   - Provide a short, descriptive name that will be added after `Private::Controls::`. For example, if you enter *`MyTestHook`*, the full Hook name becomes `Private::Controls::`*`MyTestHook`*.
   - Provide the full Hook name (also called an alias) using this format: *`Provider`*`::`*`ServiceName`*`::`*`HookName`*.
8. For **Hook mode**, choose how the Hook responds when controls fail their evaluation:

   - **Warn** — Issues warnings to users but allows actions to continue. This is useful for non-critical validations or informational checks.
   - **Fail** — Prevents the action from proceeding. This is helpful for enforcing strict compliance or security policies.
9. Choose **Next**.
10. (Optional) For **Hook filters**, do the following:

    a. For **Filtering criteria**, choose the logic for applying stack name and stack role filters:

       - **All stack names and stack roles** – The Hook will only be invoked when all specified filters match.
       - **Any stack names and stack roles** – The Hook will be invoked if at least one of the specified filters match.

    b. For **Stack names**, include or exclude specific stacks from Hook invocations.

       - For **Include**, specify the stack names to include. Use this when you have a small set of specific stacks you want to target. Only the stacks specified in this list will invoke the Hook.

- For **Exclude**, specify the stack names to exclude. Use this when you want to invoke the Hook on most stacks but exclude a few specific ones. All stacks except those listed here will invoke the Hook.

c. For **Stack roles**, include or exclude specific stacks from Hook invocations based on their associated IAM roles.

- For **Include**, specify one or more IAM role ARNs to target stacks associated with these roles. Only stack operations initiated by these roles will invoke the Hook.
- For **Exclude**, specify one or more IAM role ARNs for stacks you want to exclude. The Hook will be invoked on all stacks except those initiated by the specified roles.

11. Choose **Next**.
12. On the **Review and activate** page, review your choices. To make changes, choose **Edit** on the related section.
13. When you're ready to proceed, choose **Activate Hook**.

Activate a proactive control-based Hook (Amazon CLI)

Before you continue, confirm that you have identified the proactive controls that you'll use with this Hook. For more information, see the Amazon Control Tower Control Catalog.

To activate a proactive control-based Hook for use in your account (Amazon CLI)

1. To start activating a Hook, use the following activate-type command, replacing the placeholders with your specific values.

```
aws cloudformation activate-type --type HOOK \
  --type-name AWS::ControlTower::Hook  \
  --publisher-id aws-hooks \
  --type-name-alias MyOrg::Security::ComplianceHook \
  --region us-west-2
```

2. To finish activating the Hook, you must configure it using a JSON configuration file.

   Use the **cat** command to create a JSON file with the following structure. For more information, see Hook configuration schema syntax reference.

   This following example configures a Hook that invokes on specific IAM, Amazon EC2, and Amazon S3 resources during CREATE and UPDATE operations. It applies three proactive

controls (`CT.IAM.PR.5`, `CT.EC2.PR.17`, `CT.S3.PR.12`) to validate these resources against compliance standards. The hook operates in `WARN` mode, meaning it will flag non-compliant resources with warnings but won't block deployments.

```
$ cat > config.json
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": ["RESOURCE"],
      "FailureMode": "WARN",
      "Properties": {
        "ControlsToApply": "CT.IAM.PR.5,CT.EC2.PR.17,CT.S3.PR.12"
      },
      "TargetFilters": {
        "Actions": [
          "CREATE",
          "UPDATE"
        ]
      }
    }
  }
}
```

- `HookInvocationStatus`: Set to `ENABLED` to enable the Hook.
- `TargetOperations`: Set to `RESOURCE` as this is the only supported value for a proactive control-based Hook.
- `FailureMode`: Set to either `FAIL` or `WARN`.
- `ControlsToApply`: Specify the control IDs of the proactive controls to use. For more information, see the Amazon Control Tower Control Catalog.
- (Optional) `TargetFilters`: For `Actions`, you can specify `CREATE` or `UPDATE`, or both (default), to control when the Hook is invoked. Specifying `CREATE` alone limits the Hook to `CREATE` operations only. Other `TargetFilters` properties have no effect.

3. Use the following set-type-configuration command, along with the JSON file you created, to apply the configuration. Replace the placeholders with your specific values.

```
aws cloudformation set-type-configuration \
  --configuration file://config.json \
```

```
  --type-arn "arn:aws:cloudformation:us-west-2:123456789012:type/hook/MyOrg-
Security-ComplianceHook" \
  --region us-west-2
```

Delete proactive control-based Hooks in your account

When you no longer need an activated proactive control-based Hook, use the following procedures to delete it in your account.

To temporarily disable a Hook instead of deleting it, see Disable and enable Amazon CloudFormation Hooks.

Topics

- Delete a proactive control-based Hook in your account (console)
- Delete a proactive control-based Hook in your account (Amazon CLI)

Delete a proactive control-based Hook in your account (console)

To delete a proactive control-based Hook in your account

1. Sign in to the Amazon Web Services Management Console and open the Amazon CloudFormation console at https://console.amazonaws.cn/cloudformation.
2. On the navigation bar at the top of the screen, choose the Amazon Web Services Region where the Hook is located.
3. From the navigation pane, choose **Hooks**.
4. On the **Hooks** page, find the proactive control-based Hook you want to delete.
5. Select the check box next to your Hook and choose **Delete**.
6. When prompted for confirmation, type out the Hook name to confirm deleting the specified Hook and then choose **Delete**.

Delete a proactive control-based Hook in your account (Amazon CLI)

> **Note**
>
> Before you can delete the Hook, you must first disable it. For more information, see Disable and enable a Hook in your account (Amazon CLI).

Use the following deactivate-type command to deactivate a Hook, which removes it from your account. Replace placeholders with your specific values.

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

Guard Hooks

To use an Amazon CloudFormation Guard Hook in your account, you must *activate* the Hook for the account and Region where you want to use it. Activating a Hook makes it usable in stack operations in the account and Region where it's activated.

When you activate a Guard Hook, CloudFormation creates an entry in your account's registry for the activated Hook as a private Hook. This allows you to set any configuration properties the Hook includes. Configuration properties define how the Hook is configured for a given Amazon Web Services account and Region.

Topics

- Amazon CLI commands for working with Guard Hooks
- Write Guard rules to evaluate resources for Guard Hooks
- Prepare to create a Guard Hook
- Activate a Guard Hook in your account
- View logs for the Guard Hooks in your account
- Delete Guard Hooks in your account

Amazon CLI commands for working with Guard Hooks

The Amazon CLI commands for working with Guard Hooks include:

- activate-type to start the activation process for a Guard Hook.
- set-type-configuration to specify the configuration data for a Hook in your account.
- list-types to list the Hooks in your account.
- describe-type to return detailed information about a specific Hook or specific Hook version, including current configuration data.
- deactivate-type to remove a previously activated Hook from your account.

Write Guard rules to evaluate resources for Guard Hooks

Amazon CloudFormation Guard is an open-source and general purpose domain specific language (DSL) you can use to author policy-as-code. This topic explains how to use Guard to author example rules which can be run in the Guard Hook to automatically evaluate CloudFormation and Amazon Cloud Control API operations. It will also focus on the different types of inputs available to your Guard rules depending on when your Guard Hook runs. A Guard Hook can be configured to run during the following types of operations:

- Resource operations
- Stack operations
- Change set operations

For more information on writing Guard rules, see Writing Amazon CloudFormation Guard rules

Topics

- Resource operation Guard rules
- Stack operation Guard rules
- Change set operation Guard rules

Resource operation Guard rules

Any time you create, update, or delete a resource, that's considered a resource operation. As an example, if you run update a CloudFormation stack that creates a new resource, you have

completed a resource operation. When you create, update or delete a resource using Cloud Control API, that is also considered a resource operation. You can configure your Guard Hook to target `RESOURCE` and `CLOUD_CONTROL` operations in the `TargetOperations` configuration for your Hook. When your Guard Hook evaluates a resource operation, the Guard engine evaluates a resource input.

Topics

- Guard resource input syntax
- Example Guard resource operation input
- Guard rules for resource changes

Guard resource input syntax

The Guard resource input is the data that's made available to your Guard rules to evaluate.

The following is an example shape of a resource input:

```
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`

The ID of the Amazon Web Services account containing the resource being evaluated.

`StackId`

The stack ID of the CloudFormation stack that is part of the resource operation. This is empty if the caller is Cloud Control API.

`HookTypeName`

The name of the Hook that's running.

`HookTypeVersion`

The version of the Hook that is running.

`InvocationPoint`

The exact point in the provisioning logic where the Hook runs.

Valid values: (`CREATE_PRE_PROVISION` | `UPDATE_PRE_PROVISION` | `DELETE_PRE_PROVISION`)

`TargetName`

The target type being evaluated, for example, `AWS::S3::Bucket`.

`TargetType`

The target type being evaluated, for example `AWS::S3::Bucket`. For resources provisioned with Cloud Control API, this value will be `RESOURCE`.

`TargetLogicalId`

The `TargetLogicalId` of the resource being evaluated. If the origin of the Hook is CloudFormation, this will be the logical ID (also known as logical name) of the resource. If the origin of the Hook is Cloud Control API, this will be a constructed value.

`ChangeSetId`

The change set ID that was executed to cause the Hook invocation. This value is empty if the resource change was initiated by Cloud Control API, or the `create-stack`, `update-stack`, or `delete-stack` operations.

`Resources`

`ResourceLogicalID`

When the operation is initiated by CloudFormation, the `ResourceLogicalID` is the logical ID of the resource in the CloudFormation template.

When the operation's initiated by Cloud Control API, the `ResourceLogicalID` is a combination of the resource type, name, operation ID, and request ID.

`ResourceType`

The type name of the resource (example: `AWS::S3::Bucket`).

`ResourceProperties`

The proposed properties of the resource being modified. When the Guard Hook is running against the CloudFormation resource changes, any functions, parameters, and transforms will be fully resolved. If the resource is being deleted, this value will be empty.

`Previous`

`ResourceLogicalID`

When the operation is initiated by CloudFormation, the `ResourceLogicalID` is the logical ID of the resource in the CloudFormation template.

When the operation's initiated by Cloud Control API, the `ResourceLogicalID` is a combination of the resource type, name, operation ID, and request ID.

`ResourceType`

The type name of the resource (example: `AWS::S3::Bucket`).

`ResourceProperties`

The current properties associated with the resource being modified. If the resource is being deleted, this value will be empty.

Example Guard resource operation input

The following example input shows a Guard Hook that will receive the definition of the `AWS::S3::Bucket` resource to update. This is the data available to Guard for evaluation.

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

To see all of the properties available for the resource type, see AWS::S3::Bucket.

Guard rules for resource changes

When a Guard Hook evaluates resource changes, it starts by downloading all the rules configured with the Hook. These rules are then evaluated against the resource input. The Hook will fail if any rules fail their evaluation. If there are no failures, the Hook will pass.

The following example is a Guard rule that evaluates if the `ObjectLockEnabled` property is `true` for any `AWS::S3::Bucket` resource types.

```
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.
  >>
}
```

When this rule runs against the following input, it will fail since the `ObjectLockEnabled` property isn't set to `true`.

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

When this rule runs against the following input, it will pass since the `ObjectLockEnabled` is set to `true`.

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

When a Hook fails, the rules which failed will be propagated back to CloudFormation or Cloud Control API. If a logging bucket has been configured for the Guard Hook, additional rule feedback will be provided there. This additional feedback includes the `Violation` and `Fix` information.

Stack operation Guard rules

When a CloudFormation stack is created, updated, or deleted, you can configure your Guard Hook to start by evaluating the new template and potentially block the stack operation from proceeding. You can configure your Guard Hook to target `STACK` operations in the `TargetOperations` configuration for your Hook.

Topics

- Guard stack input syntax
- Example Guard stack operation input
- Guard rules for stack changes

Guard stack input syntax

The input for Guard stack operations provides the whole CloudFormation template for your Guard rules to evaluate.

The following is an example shape of a stack input:

```
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`

The ID of the Amazon Web Services account containing the resource.

`StackId`

The stack ID of the CloudFormation stack that is part of the stack operation.

`HookTypeName`

The name of the Hook that's running.

`HookTypeVersion`

The version of the Hook that is running.

`InvocationPoint`

The exact point in the provisioning logic where the Hook runs.

Valid values: (`CREATE_PRE_PROVISION` | `UPDATE_PRE_PROVISION` | `DELETE_PRE_PROVISION`)

`TargetName`

The name of the stack being evaluated.

`TargetType`

This value will be `STACK` when running as a stack-level Hook.

`ChangeSetId`

The change set ID that was executed to cause the Hook invocation. This value is empty if the stack operation was initiated by a `create-stack`, `update-stack`, or `delete-stack` operation.

`Proposed CloudFormation Template`

The full CloudFormation template value that was passed to CloudFormation `create-stack` or `update-stack` operations. This includes things like the `Resources`, `Outputs`, and `Properties`. It can be a JSON or YAML string depending on what was provided to CloudFormation.

In `delete-stack` operations, this value will be empty.

`Previous`

The last successfully deployed CloudFormation template. This value is empty if the stack is being created or deleted.

In `delete-stack` operations, this value will be empty.

> **Note**
>
> The templates provided are what is passed into `create` or `update` stack operations. When deleting a stack, no template values are provided.

Example Guard stack operation input

The following example input shows a Guard Hook that will receive a full template and the previously deployed template. The template in this example is using the JSON format.

```
HookContext:
```

```
  AwsAccountId: 123456789012
  StackId: "arn:aws:cloudformation:us-west-2:123456789012:stack/
MyStack/1a2345b6-0000-00a0-a123-00abc0abc000"
  HookTypeName: org::templatechecker::hook
  HookTypeVersion: "00001"
  InvocationPoint: UPDATE_PRE_PROVISION
  TargetName: MyStack
  TargetType: CHANGE_SET
  TargetLogicalId: arn:aws:cloudformation:us-west-2:123456789012:changeSet/
SampleChangeSet/1a2345b6-0000-00a0-a123-00abc0abc000
  ChangeSetId: arn:aws:cloudformation:us-west-2:123456789012:changeSet/
SampleChangeSet/1a2345b6-0000-00a0-a123-00abc0abc000
Resources: {
   "S3Bucket": {
        "Type": "AWS::S3::Bucket",
        "Properties": {
           "BucketEncryption": {
               "ServerSideEncryptionConfiguration": [
                {"ServerSideEncryptionByDefault":
                    {"SSEAlgorithm": "aws:kms",
                      "KMSMasterKeyID": "KMS-KEY-ARN" },
                      "BucketKeyEnabled": true }
                ]
           }
        }
}
Previous: {
    "AWSTemplateFormatVersion": "2010-09-09",
    "Resources": {
        "S3Bucket": {
            "Type": "AWS::S3::Bucket",
            "Properties": {}
        }
    }
}
```

Guard rules for stack changes

When a Guard Hook evaluates stack changes, it starts by downloading all the rules configured with the Hook. These rules are then evaluated against the resource input. The Hook will fail if any rules fail their evaluation. If there are no failures, the Hook will pass.

The following example is a Guard rule that evaluates if there are any `AWS::S3::Bucket` resource types containing a property called `BucketEncryption`, with the `SSEAlgorithm` set to either `aws:kms` or `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"
  >>
}
```

When the rule runs against the following template, it will `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}'
```

When the rule runs against the following template, it will `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
```

Change set operation Guard rules

When a CloudFormation change set is created, you can configure your Guard Hook to evaluate the template and changes proposed in the change set to block the change set execution.

Topics

- Guard change set input syntax
- Example Guard change set operation input
- Guard rule for change set operations

Guard change set input syntax

The Guard change set input is the data that's made available to your Guard rules to evaluate.

The following is an example shape of a change set input:

```
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}]
```

The `ResourceChange` model syntax is:

```
logicalResourceId: String
resourceType: String
action: CREATE, UPDATE, DELETE
```

```
lineNumber: Number
beforeContext: JSON String
afterContext: JSON String
```

HookContext

AWSAccountID

The ID of the Amazon Web Services account containing the resource.

StackId

The stack ID of the CloudFormation stack that is part of the stack operation.

HookTypeName

The name of the Hook that's running.

HookTypeVersion

The version of the Hook that is running.

InvocationPoint

The exact point in the provisioning logic where the Hook runs.

Valid values: (CREATE_PRE_PROVISION | UPDATE_PRE_PROVISION | DELETE_PRE_PROVISION)

TargetName

The name of the stack being evaluated.

TargetType

This value will be CHANGE_SET when running as a change set-level Hook.

TargetLogicalId

This value will be the ARN of the change set.

ChangeSetId

The change set ID that was executed to cause the Hook invocation. This value is empty if the stack operation was initiated by a create-stack, update-stack, or delete-stack operation.

`Proposed CloudFormation Template`

The full CloudFormation template that was provided to a `create-change-set` operation. It can be a JSON or YAML string depending on what was provided to CloudFormation.

`Previous`

The last successfully deployed CloudFormation template. This value is empty if the stack is being created or deleted.

`Changes`

The `Changes` model. This lists the resource changes.

Changes

logicalResourceId

The logical resource name of the changed resource.

resourceType

The resource type that will be changed.

action

The type of operation being performed on the resource.

Valid values: (`CREATE` | `UPDATE` | `DELETE`)

lineNumber

The line number in the template associated with the change.

beforeContext

A JSON string of properties of the resource before the change:

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

afterContext

A JSON string of properties of the resource after the change:

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

Example Guard change set operation input

The following example input shows a Guard Hook that will receive a full template, the previously deployed template, and a list of resource changes. The template in this example is using the JSON format.

```
HookContext:
  AwsAccountId: "00000000"
  StackId: MyStack
  HookTypeName: org::templatechecker::hook
  HookTypeVersion: "00001"
  InvocationPoint: UPDATE_PRE_PROVISION
  TargetName: my-example-stack
  TargetType:STACK
  TargetLogicalId: arn...:changeSet/change-set
  ChangeSetId: ""
Resources: {
    "S3Bucket": {
       "Type": "AWS::S3::Bucket",
       "Properties": {
           "BucketName": "amzn-s3-demo-bucket",
           "VersioningConfiguration":{
              "Status": "Enabled"
           }
        }
    }
Previous: {
    "AWSTemplateFormatVersion": "2010-09-09",
    "Resources": {
        "S3Bucket": {
            "Type": "AWS::S3::Bucket",
            "Properties": {
                "BucketName": "amzn-s3-demo-bucket",
                "VersioningConfiguration":{
                  "Status": "Suspended"
                }
            }
        }
    }
}
Changes: [
  {
    "logicalResourceId": "S3Bucket",
```

```
    "resourceType": "AWS::S3::Bucket",
    "action": "UPDATE",
    "lineNumber": 5,
    "beforeContext": "{\"Properties\":{\"VersioningConfiguration\":{\"Status\":
\"Suspended\"}}}",
    "afterContext": "{\"Properties\":{\"VersioningConfiguration\":{\"Status\":\"Enabled
\"}}}"
  }
]
```

Guard rule for change set operations

The following example is a Guard rule that evaluates changes to Amazon S3 buckets, and ensures that `VersionConfiguration` is not disabled.

```
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'
    }
}
```

Prepare to create a Guard Hook

Before you create a Guard Hook, you must complete the following prerequisites:

- You must have already created a Guard rule. For more information, see the Write Guard rules for Hooks.
- The user or role that creates the Hook must have sufficient permissions to activate Hooks. For more information, see Grant IAM permissions for CloudFormation Hooks.
- To use the Amazon CLI or an SDK to create a Guard Hook, you must manually create an execution role with IAM permissions and a trust policy to allow CloudFormation to invoke a Guard Hook.

Create an execution role for a Guard Hook

A Hook uses an execution role for the permissions that it requires to invoke that Hook in your Amazon Web Services account.

This role can be created automatically if you create a Guard Hook from the Amazon Web Services Management Console; otherwise, you must create this role yourself.

The following section shows you how to set up permissions to create your Guard Hook.

Required permissions

Follow the guidance at Create a role using custom trust policies in the *IAM User Guide* to create a role with a custom trust policy.

Then, complete the following steps to set up your permissions:

1. Attach the following minimum privilege policy to the IAM role you want to use to create the Guard Hook.

   JSON

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

```
}
```

2. Give your Hook permission to assume the role by adding a trust policy to the role. The following shows an example trust policy you can use.

   JSON

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

Activate a Guard Hook in your account

The following topic shows you how to activate a Guard Hook in your account, which makes it usable in the account and Region it was activated in.

Topics

- Activate a Guard Hook (console)
- Activate a Guard Hook (Amazon CLI)
- Related resources

Activate a Guard Hook (console)

To activate a Guard Hook for use in your account

1. Sign in to the Amazon Web Services Management Console and open the Amazon CloudFormation console at https://console.amazonaws.cn/cloudformation.

2. On the navigation bar at the top of the screen, choose the Amazon Web Services Region where you want to create the Hook in.
3. In the navigation pane on the left, choose **Hooks**.
4. On the **Hooks** page, choose **Create a Hook**, and then choose **With Guard**.
5. If you *haven't* created any Guard rules yet, create your Guard rule, store it in Amazon S3, and then return to this procedure. Refer to the example rules in Write Guard rules to evaluate resources for Guard Hooks to get started.

   If you have already created your Guard rule and stored it in S3, proceed to the next step.

   > **Note**
   >
   > The object stored in S3 must have one of the following file extensions: `.guard`, `.zip`, or `.tar.gz`.

6. For **Guard Hook source**, **Store your Guard rules in S3**, do the following:

   - For **S3 URI**, specify the S3 path to your rules file or use the **Browse S3** button to open a dialog box to browse for and select the S3 object.
   - (Optional) For **Object version**, if your S3 bucket has versioning enabled, you can select a specific version of the S3 object.

     The Guard Hook downloads your rules from S3 every time the Hook is invoked. To prevent accidental changes or deletions, we recommend using a version when configuring your Guard Hook.

7. (Optional) For **S3 bucket for Guard output report**, specify an S3 bucket to store the Guard output report. This report contains the results of your Guard rule validations.

   To configure the output report destination, choose one of the following options:

   - Select the **Use the same bucket my Guard rules are stored in** check box to use the same bucket where your Guard rules are located.
   - Choose a different S3 bucket name for storing the Guard output report.

8. (Optional) Expand **Guard rule input parameters**, and then provide the following information under **Store your Guard rule input parameters in S3**:

   - For **S3 URI**, specify the S3 path to a parameter file or use the **Browse S3** button to open a dialog box to browse for and select the S3 object.

- (Optional) For **Object version**, if your S3 bucket has versioning enabled, you can select a specific version of the S3 object.

9. Choose **Next**.
10. For **Hook name**, choose one of the following options:

    - Provide a short, descriptive name that will be added after `Private::Guard::`. For example, if you enter *`MyTestHook`*, the full Hook name becomes `Private::Guard::`*`MyTestHook`*.
    - Provide the full Hook name (also called an alias) using this format: *`Provider`*`::`*`ServiceName`*`::`*`HookName`*

11. For **Hook targets**, choose what to evaluate:

    - **Stacks** — Evaluates stack templates when users create, update, or delete stacks.
    - **Resources** — Evaluates individual resource changes when users update stacks.
    - **Change sets** — Evaluates planned updates when users create change sets.
    - **Cloud Control API** — Evaluates create, update or delete operations initiated by the Cloud Control API.

12. For **Actions**, choose which actions (create, update, delete) will invoke your Hook.
13. For **Hook mode**, choose how the Hook responds when rules fail their evaluation:

    - **Warn** — Issues warnings to users but allows actions to continue. This is useful for non-critical validations or informational checks.
    - **Fail** — Prevents the action from proceeding. This is helpful for enforcing strict compliance or security policies.

14. For **Execution role**, choose the IAM role that the Hook assumes to retrieve your Guard rules from S3 and optionally write a detailed Guard output report back. You can either allow CloudFormation to automatically create an execution role for you or you can specify a role that you've created.
15. Choose **Next**.
16. (Optional) For **Hook filters**, do the following:

    a. For **Resource filter**, specify which resource types can invoke the Hook. This ensures that the Hook is only invoked for relevant resources.

    b. For **Filtering criteria**, choose the logic for applying stack name and stack role filters:

- **All stack names and stack roles** – The Hook will only be invoked when all specified filters match.
- **Any stack names and stack roles** – The Hook will be invoked if at least one of the specified filters match.

> **Note**
>
> For Cloud Control API operations, all **Stack names** and **Stack roles** filters are ignored.

c. For **Stack names**, include or exclude specific stacks from Hook invocations.

- For **Include**, specify the stack names to include. Use this when you have a small set of specific stacks you want to target. Only the stacks specified in this list will invoke the Hook.
- For **Exclude**, specify the stack names to exclude. Use this when you want to invoke the Hook on most stacks but exclude a few specific ones. All stacks except those listed here will invoke the Hook.

d. For **Stack roles**, include or exclude specific stacks from Hook invocations based on their associated IAM roles.

- For **Include**, specify one or more IAM role ARNs to target stacks associated with these roles. Only stack operations initiated by these roles will invoke the Hook.
- For **Exclude**, specify one or more IAM role ARNs for stacks you want to exclude. The Hook will be invoked on all stacks except those initiated by the specified roles.

17. Choose **Next**.
18. On the **Review and activate** page, review your choices. To make changes, choose **Edit** on the related section.
19. When you're ready to proceed, choose **Activate Hook**.

Activate a Guard Hook (Amazon CLI)

Before you continue, confirm that you have created the Guard rule and the execution role that you'll use with this Hook. For more information, see Write Guard rules to evaluate resources for Guard Hooks and Create an execution role for a Guard Hook.

To activate a Guard Hook for use in your account (Amazon CLI)

1. To start activating a Hook, use the following activate-type command, replacing the placeholders with your specific values. This command authorizes the Hook to use a specified execution role from your Amazon Web Services account.

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

2. To finish activating the Hook, you must configure it using a JSON configuration file.

   Use the **cat** command to create a JSON file with the following structure. For more information, see Hook configuration schema syntax reference.

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

```
}
```

- `HookInvocationStatus`: Set to `ENABLED` to enable the Hook.
- `TargetOperations`: Specify the operations that the Hook will evaluate.
- `FailureMode`: Set to either `FAIL` or `WARN`.
- `ruleLocation`: Replace with the S3 URI where your rule is stored. The object stored in S3 must have one of the following file extensions: `.guard`, `.zip`, and `.tar.gz`.
- `logBucket`: (Optional) Specify the name of an S3 bucket for Guard JSON reports.
- `TargetFilters`: Specify the types of actions that will invoke the Hook.

3. Use the following set-type-configuration command, along with the JSON file you created, to apply the configuration. Replace the placeholders with your specific values.

```
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
```

Related resources

We provide template examples that you can use to understand how to declare a Guard Hook in a CloudFormation stack template. For more information, see AWS::CloudFormation::GuardHook in the *Amazon CloudFormation User Guide*.

View logs for the Guard Hooks in your account

When you activate a Guard Hook, you can specify an Amazon S3 bucket as the destination for the Hook output report. Once activated, the Hook automatically stores the results of your Guard rule validations in the specified bucket. You can then view these results in the Amazon S3 console.

View Guard Hook logs in the Amazon S3 console

To view the Guard Hook output log file

1. Sign-in to the https://console.amazonaws.cn/s3/.
2. On the navigation bar at the top of the screen, choose your Amazon Web Services Region.
3. Choose **Buckets**.

4. Choose the bucket you selected for your Guard output report.
5. Choose the desired validation output report log file.
6. Choose whether you want to **Download** the file or **Open** it to view.

Delete Guard Hooks in your account

When you no longer need an activated Guard Hook, use the following procedures to delete it in your account.

To temporarily disable a Hook instead of deleting it, see Disable and enable Amazon CloudFormation Hooks.

Topics

- Delete a Guard Hook in your account (console)
- Delete a Guard Hook in your account (Amazon CLI)

Delete a Guard Hook in your account (console)

To delete a Guard Hook in your account

1. Sign in to the Amazon Web Services Management Console and open the Amazon CloudFormation console at https://console.amazonaws.cn/cloudformation.
2. On the navigation bar at the top of the screen, choose the Amazon Web Services Region where the Hook is located.
3. From the navigation pane, choose **Hooks**.
4. On the **Hooks** page, find the Guard Hook you want to delete.
5. Select the check box next to your Hook and choose **Delete**.
6. When prompted for confirmation, type out the Hook name to confirm deleting the specified Hook and then choose **Delete**.

Delete a Guard Hook in your account (Amazon CLI)

> **Note**
>
> Before you can delete the Hook, you must first disable it. For more information, see Disable and enable a Hook in your account (Amazon CLI).

Use the following deactivate-type command to deactivate a Hook, which removes it from your account. Replace placeholders with your specific values.

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

Lambda Hooks

To use an Amazon Lambda Hook in your account, you must first *activate* the Hook for the account and Region where you want to use it. Activating a Hook makes it usable in stack operations in the account and Region where it's activated.

When you activate a Lambda Hook, CloudFormation creates an entry in your account's registry for the activated Hook as a private Hook. This allows you to set any configuration properties the Hook includes. Configuration properties define how the Hook is configured for a given Amazon Web Services account and Region.

Topics

- Amazon CLI commands for working with Lambda Hooks
- Create Lambda functions to evaluate resources for Lambda Hooks
- Prepare to create a Lambda Hook
- Activate a Lambda Hook in your account
- View logs for the Lambda Hooks in your account
- Delete Lambda Hooks in your account

Amazon CLI commands for working with Lambda Hooks

The Amazon CLI commands for working with Lambda Hooks include:

- activate-type to start the activation process for a Lambda Hook.
- set-type-configuration to specify the configuration data for a Hook in your account.
- list-types to list the Hooks in your account.
- describe-type to return detailed information about a specific Hook or specific Hook version, including current configuration data.
- deactivate-type to remove a previously activated Hook from your account.

Create Lambda functions to evaluate resources for Lambda Hooks

Amazon CloudFormation Lambda Hooks allows you to evaluate CloudFormation and Amazon Cloud Control API operations against your own custom code. Your Hook can block an operation from proceeding, or issue a warning to the caller and allow the operation to proceed. When you create a Lambda Hook, you can configure it to intercept and evaluate the following CloudFormation operations:

- Resource operations
- Stack operations
- Change set operations

Topics

- Developing a Lambda Hook
- Evaluating resource operations with Lambda Hooks
- Evaluating stack operations with Lambda Hooks
- Evaluating change set operations with Lambda Hooks

Developing a Lambda Hook

When Hooks invoke your Lambda it will wait up to 30 seconds for the Lambda to evaluate the input. The Lambda will return a JSON response that indicates whether the Hook succeeded or failed.

Topics

- Request input
- Response input

- Examples

Request input

The input passed to your Lambda function depends on the Hook target operation (examples: stack, resource, or change set).

Response input

In order to communicate to Hooks if your request succeeded or failed, your Lambda function needs to return a JSON response.

The following is an example shape of the response Hooks expects:

```
{
  "hookStatus": "SUCCESS" or "FAILED" or "IN_PROGRESS",
  "errorCode": "NonCompliant" or "InternalFailure"
  "message": String,
  "clientRequestToken": String,
  "callbackContext": None,
  "callbackDelaySeconds": Integer,
  "annotations": [
    {
      "annotationName": String,
      "status": "PASSED" or "FAILED" or "SKIPPED",
      "statusMessage": String,
      "remediationMessage": String,
      "remediationLink": String,
      "severityLevel": "INFORMATIONAL" or "LOW" or "MEDIUM" or "HIGH" or "CRITICAL"
    }
  ]
}
```

hookStatus

The status of the Hook. This is a required field.

Valid values: (SUCCESS | FAILED | IN_PROGRESS)

> **Note**
>
> A Hook can return `IN_PROGRESS` 3 times. If no result is returned, the Hook will fail. For a Lambda Hook, this means your Lambda function can be invoked up to 3 times.

errorCode

Shows whether the operation was evaluated and determined to be invalid, or if errors occurred within the Hook, preventing the evaluation. This field is required if the Hook fails.

Valid values: (`NonCompliant` | `InternalFailure`)

message

The message to the caller that states why the Hook succeeded or failed.

> **Note**
>
> When evaluating CloudFormation operations, this field is truncated to 4096 characters. When evaluating Cloud Control API operations, this field is truncated to 1024 characters.

clientRequestToken

The request token that was provided as an input to the Hook request. This is a required field.

callbackContext

If you indicate that the `hookStatus` is `IN_PROGRESS` you pass an additional context that's provided as input when the Lambda function is reinvoked.

callbackDelaySeconds

How long Hooks should wait to invoke this Hook again.

annotations

An array of annotation objects that provide further details and remediation guidance.

annotationName

An identifier for the annotation.

status

The Hook invocation status. This is helpful when annotations represent logic with pass/fail evaluation similar to a Guard rule.

Valid values: (`PASSED` | `FAILED` | `SKIPPED`)

statusMessage

Explanation for the specific status.

remediationMessage

Suggestion for fixing a `FAILED` status. For example, if a resource is missing encryption, you can state how to add encryption to the resource configuration.

remediationLink

An HTTP URL for additional remediation guidance.

severityLevel

Defines the relative risk associated with any violations of this type. When assigning severity levels to your Hook invocation results, you can reference the Amazon Security Hub CSPM severity framework as an example of how to structure meaningful severity categories.

Valid values: (`INFORMATIONAL` | `LOW` | `MEDIUM` | `HIGH` | `CRITICAL`)

Examples

The following is an example of a successful response:

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

The following is an example of a failed response:

```
{
  "hookStatus": "FAILED",
  "errorCode": "NonCompliant",
```

```
  "message": "S3 Bucket Versioning must be enabled.",
  "clientRequestToken": "123avjdjk31"
 }
```

Evaluating resource operations with Lambda Hooks

Any time you create, update, or delete a resource, that's considered a resource operation. As an example, if you run update a CloudFormation stack that creates a new resource, you have completed a resource operation. When you create, update or delete a resource using Cloud Control API, that is also considered a resource operation. You can configure your CloudFormation Lambda Hook to target `RESOURCE` and `CLOUD_CONTROL` operations in the Hook `TargetOperations` configuration.

> **Note**
>
> The `delete` Hook handler is only invoked when a resource is deleted using an operation trigger from Cloud Control API `delete-resource` or CloudFormation `delete-stack`.

Topics

- Lambda Hook resource input syntax
- Example Lambda Hook resource change input
- Example Lambda function for resource operations

Lambda Hook resource input syntax

When your Lambda is invoked for a resource operation, you'll receive a JSON input containing the resource properties, proposed properties, and the context around the Hook invocation.

The following is an example shape of the JSON input:

```
{
    "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`

The ID of the Amazon Web Services account containing the resource being evaluated.

`stackId`

The stack ID of the CloudFormation stack this operation is a part of. This field is empty if the caller is Cloud Control API.

`changeSetId`

The ID of the change set that initiated the Hook invocation. This value is empty if the resource change was initiated by Cloud Control API, or the `create-stack`, `update-stack`, or `delete-stack` operations.

`hookTypeName`

The name of the Hook that's running.

`hookTypeVersion`

The version of the Hook that's running.

`hookModel`

`LambdaFunction`

The current Lambda ARN invoked by the Hook.

`actionInvocationPoint`

The exact point in the provisioning logic where the Hook runs.

Valid values: (`CREATE_PRE_PROVISION` | `UPDATE_PRE_PROVISION` | `DELETE_PRE_PROVISION`)

`requestData`

`targetName`

The target type being evaluated, for example, `AWS::S3::Bucket`.

`targetType`

The target type being evaluated, for example `AWS::S3::Bucket`. For resources provisioned with Cloud Control API, this value will be `RESOURCE`.

`targetLogicalId`

The logical ID of the resource being evaluated. If the origin of the Hook invocation is CloudFormation, this will be the logical resource ID defined in your CloudFormation template. If the origin of this Hook invocation is Cloud Control API, this will be a constructed value.

`targetModel`

`resourceProperties`

The proposed properties of the resource being modified. If the resource is being deleted, this value will be empty.

`previousResourceProperties`

The properties that are currently associated with the resource being modified. If the resource is being created, this value will be empty.

`requestContext`

invocation

The current attempt at executing the Hook.

callbackContext

If the Hookwas set to `IN_PROGRESS`, and `callbackContext` was returned, it will be here after reinvocation.

Example Lambda Hook resource change input

The following example input shows a Lambda Hook that will receive the definition of the `AWS::DynamoDB::Table` resource to update, where the `ReadCapacityUnits` of `ProvisionedThroughput` is changed from 3 to 10. This is the data available to Lambda for evaluation.

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

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

To see all of the properties available for the resource type, see AWS::DynamoDB::Table.

Example Lambda function for resource operations

The following is a simple function that fails any resource update to DynamoDB, which tries to set the `ReadCapacity` of `ProvisionedThroughput` to something larger than 10. If the Hook succeeds, the message, "ReadCapacity is correctly configured," will display to the caller. If the request fails validation, the Hook will fail with the status, "ReadCapacity cannot be more than 10."

Node.js

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

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

Python

```
import json

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

    response = {
        "hookStatus": "SUCCESS",
        "message": "ReadCapacity is correctly configured.",
```

```
        "clientRequestToken": event.get('clientRequestToken')
    }

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

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

    return response
```

Evaluating stack operations with Lambda Hooks

Any time you create, update, or delete a stack with a new template, you can configure your CloudFormation Lambda Hook to start by evaluating the new template and potentially block the stack operation from proceeding. You can configure your CloudFormation Lambda Hook to target STACK operations in the Hook `TargetOperations` configuration.

Topics

- Lambda Hook stack input syntax
- Example Lambda Hook stack change input
- Example Lambda function for stack operations

Lambda Hook stack input syntax

When your Lambda is invoked for a stack operation, you'll receive a JSON request containing the Hook invocation context, `actionInvocationPoint`, and request context. Due to the size of CloudFormation templates, and the limited input size accepted by Lambda functions, the actual templates are stored in an Amazon S3 object. The input of the `requestData` includes an Amazon S3 resigned URL to another object, which contains the current and previous template version.

The following is an example shape of the JSON input:

```
{
    "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`

The request token that was provided as an input to the Hook request. This is a required field.

`awsAccountId`

The ID of the Amazon Web Services account containing the stack being evaluated.

`stackID`

The stack ID of the CloudFormation stack.

`changeSetId`

The ID of the change set that initiated the Hook invocation. This value is empty if the stack change was initiated by Cloud Control API, or the `create-stack`, `update-stack`, or `delete-stack` operations.

hookTypeName

The name of the Hook that's running.

`hookTypeVersion`

The version of the Hook that's running.

`hookModel`

`LambdaFunction`

The current Lambda ARN invoked by the Hook.

`actionInvocationPoint`

The exact point in the provisioning logic where the Hook runs.

Valid values: (`CREATE_PRE_PROVISION` | `UPDATE_PRE_PROVISION` | `DELETE_PRE_PROVISION`)

`requestData`

`targetName`

This value will be `STACK`.

`targetType`

This value will be `STACK`.

`targetLogicalId`

The stack name.

`payload`

The Amazon S3 presigned URL containing a JSON object with the current and previous template definitions.

`requestContext`

If the Hook is being reinvoked, this object will be set.

`invocation`

The current attempt at executing the Hook.

`callbackContext`

If the Hook was set to `IN_PROGRESS` and `callbackContext` was returned, it will be here upon reinvocation.

The `payload` property in the request data is a URL that your code needs to fetch. Once it has received the URL, you get an object with the following schema:

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

`template`

The full CloudFormation template that was provided to `create-stack` or `update-stack`. It can be a JSON or YAML string depending on what was provided to CloudFormation.

In `delete-stack` operations, this value will be empty.

`previousTemplate`

The previous CloudFormation template. It can be a JSON or YAML string depending on what was provided to CloudFormation.

In `delete-stack` operations, this value will be empty.

Example Lambda Hook stack change input

The following is an example stack change input. The Hook is evaluating a change which updates the `ObjectLockEnabled` to true, and adds an Amazon SQS queue:

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

```
        "targetLogicalId": "my-cloudformation-stack",
        "payload": "https://s3......"
    },
    "requestContext": {
        "invocation": 1,
        "callbackContext": null
    }
}
```

This is an example `payload` of the `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}}}}"
}
```

Example Lambda function for stack operations

The following example is a simple function that downloads the stack operation payload, parses the template JSON, and returns SUCCESS.

Node.js

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

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

```
            // Do something with the template templateHookPayload.previousTemplate
            // JSON or YAML
        }
    } catch (error) {
        console.log(error);
        response.hookStatus = "FAILED";
        response.message = "Failed to evaluate stack operation.";
        response.errorCode = "InternalFailure";
    }
    return response;
};
```

Python

To use Python, you'll need to import the `requests` library. To do this, you'll need to include the library in your deployment package when creating your Lambda function. For more information, see Creating a .zip deployment package with dependencies in the *Amazon Lambda Developer Guide*.

```
import json
import requests

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

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

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

        if 'template' in template_hook_payload:
            # Do something with the template template_hook_payload['template']
```

```
            # JSON or YAML
            pass

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

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

    return response
```

Evaluating change set operations with Lambda Hooks

Any time you create a change set, you can configure your CloudFormation Lambda Hook to first evaluate the new change set and potentially block its execution. You can configure your CloudFormation Lambda Hook to target `CHANGE_SET` operations in the Hook `TargetOperations` configuration.

Topics

- Lambda Hook change set input syntax
- Example Lambda Hook change set change input
- Example Lambda function for change set operations

Lambda Hook change set input syntax

The input for change set operations is similar to stack operations, but the payload of the `requestData` also includes a list of resource changes introduced by the change set.

The following is an example shape of the JSON input:

```
{
    "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`

The request token that was provided as an input to the Hook request. This is a required field.

`awsAccountId`

The ID of the Amazon Web Services account containing the stack being evaluated.

`stackID`

The stack ID of the CloudFormation stack.

`changeSetId`

The ID of the change set that initiated the Hook invocation.

`hookTypeName`

The name of the Hook that's running.

`hookTypeVersion`

The version of the Hook that's running.

`hookModel`

`LambdaFunction`

The current Lambda ARN invoked by the Hook.

`requestData`

`targetName`

This value will be `CHANGE_SET`.

`targetType`

This value will be `CHANGE_SET`.

`targetLogicalId`

The change set ARN..

`payload`

The Amazon S3 presigned URL containing a JSON object with the current template, as well as a list of changes introduced by this change set.

`requestContext`

If the Hook is being reinvoked, this object will be set.

`invocation`

The current attempt at executing the Hook.

`callbackContext`

If the Hook was set to `IN_PROGRESS` and `callbackContext` was returned, it will be here upon reinvocation.

The `payload` property in the request data is a URL that your code needs to fetch. Once it has received the URL, you get an object with the following schema:

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

```
    ]
}
```

`template`

The full CloudFormation template that was provided to `create-stack` or `update-stack`. It can be a JSON or YAML string depending on what was provided to CloudFormation.

`changedResources`

A list of changed resources.

`action`

The type of change applied to the resource.

Valid values: (`CREATE` | `UPDATE` | `DELETE`)

`beforeContext`

A JSON string of the resource properties before the change. This value is null when the resource is being created. All boolean and number values in this JSON string are STRINGS.

`afterContext`

A JSON string of the resources properties if this change set is executed. This value is null when the resource is being deleted. All boolean and number values in this JSON string are STRINGS.

`lineNumber`

The line number in the template that caused this change. If the action is `DELETE` this value will be null.

`logicalResourceId`

The logical resource ID of the resource being changed.

`resourceType`

The resource type that's being changed.

Example Lambda Hook change set change input

The following is an example change set change input. In the following example, you can see the changes introduced by the change set. The first change is deleting a queue called `CoolQueue`. The

second change is adding a new queue called `NewCoolQueue`. The last change is an update to the `DynamoDBTable`.

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

This is an example `payload` of the `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"}]}}'
    }
  ]
}
```

Example Lambda function for change set operations

The following example is a simple function that downloads the change set operation payload, loops through each change, and then prints out the before and after properties before it returns a SUCCESS.

Node.js

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

Python

To use Python, you'll need to import the `requests` library. To do this, you'll need to include the library in your deployment package when creating your Lambda function. For more information, see Creating a .zip deployment package with dependencies in the *Amazon Lambda Developer Guide*.

```
import json
import requests

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

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

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

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

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

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

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

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

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

    return response
```

Prepare to create a Lambda Hook

Before you create a Lambda Hook, you must complete the following prerequisites:

- You must have already created a Lambda function. For more information, see the Create Lambda functions for Hooks.
- The user or role that creates the Hook must have sufficient permissions to activate Hooks. For more information, see Grant IAM permissions for CloudFormation Hooks.
- To use the Amazon CLI or an SDK to create a Lambda Hook, you must manually create an execution role with IAM permissions and a trust policy to allow CloudFormation to invoke a Lambda Hook.

Create an execution role for a Lambda Hook

A Hook uses an execution role for the permissions that it requires to invoke that Hook in your Amazon Web Services account.

This role can be created automatically if you create a Lambda Hook from the Amazon Web Services Management Console; otherwise, you must create this role yourself.

The following section shows you how to set up permissions to create your Lambda Hook.

Required permissions

Follow the guidance at Create a role using custom trust policies in the *IAM User Guide* to create a role with a custom trust policy.

Then, complete the following steps to set up your permissions:

1. Attach the following minimum privilege policy to the IAM role you want to use to create the Lambda Hook.

   JSON

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

2. Give your Hook permission to assume the role by adding a trust policy to the role. The following shows an example trust policy you can use.

   JSON

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

Activate a Lambda Hook in your account

The following topic shows you how to activate a Lambda Hook in your account, which makes it usable in the account and Region it was activated in.

Topics

- Activate a Lambda Hook (console)
- Activate a Lambda Hook (Amazon CLI)
- Related resources

Activate a Lambda Hook (console)

To activate a Lambda Hook for use in your account

1. Sign in to the Amazon Web Services Management Console and open the Amazon CloudFormation console at https://console.amazonaws.cn/cloudformation.
2. On the navigation bar at the top of the screen, choose the Amazon Web Services Region where you want to create the Hook in.
3. If you *haven't* created a Lambda function for the Hook, do the following:

   - Open the Functions page on the Lambda console.
   - Create the Lambda function that you'll use with this Hook, and then return to this procedure. For more information, see Create Lambda functions to evaluate resources for Lambda Hooks.

   If you have already created your Lambda function, proceed to the next step.
4. In the navigation pane on the left, choose **Hooks**.
5. On the **Hooks** page, choose **Create a Hook**, and then choose **With Lambda**.
6. For **Hook name**, choose one of the following options:

   - Provide a short, descriptive name that will be added after `Private::Lambda::`. For example, if you enter *`MyTestHook`*, the full Hook name becomes `Private::Lambda::`*`MyTestHook`*.
   - Provide the full Hook name (also called an alias) using this format: *`Provider`*`::`*`ServiceName`*`::`*`HookName`*

7. For **Lambda function**, provide the Lambda function to be used with this Hook. You can use:
   - The full Amazon Resource Name (ARN) without a suffix.
   - A qualified ARN with a version or alias suffix.
8. For **Hook targets**, choose what to evaluate:
   - **Stacks** — Evaluates stack templates when users create, update, or delete stacks.
   - **Resources** — Evaluates individual resource changes when users update stacks.
   - **Change sets** — Evaluates planned updates when users create change sets.
   - **Cloud Control API** — Evaluates create, update or delete operations initiated by the Cloud Control API.
9. For **Actions**, choose which actions (create, update, delete) will invoke your Hook.
10. For **Hook mode**, choose how the Hook responds when the Lambda function invoked by the Hook returns a `FAILED` response:
    - **Warn** — Issues warnings to users but allows actions to continue. This is useful for non-critical validations or informational checks.
    - **Fail** — Prevents the action from proceeding. This is helpful for enforcing strict compliance or security policies.
11. For **Execution role**, choose the IAM role that the Hook assumes to invoke your Lambda function. You can either allow CloudFormation to automatically create an execution role for you or you can specify a role that you've created.
12. Choose **Next**.
13. (Optional) For **Hook filters**, do the following:
    a. For **Resource filter**, specify which resource types can invoke the Hook. This ensures that the Hook is only invoked for relevant resources.
    b. For **Filtering criteria**, choose the logic for applying stack name and stack role filters:
       - **All stack names and stack roles** – The Hook will only be invoked when all specified filters match.
       - **Any stack names and stack roles** – The Hook will be invoked if at least one of the specified filters match.

> **Note**
>
> For Cloud Control API operations, all **Stack names** and **Stack roles** filters are ignored.

c. For **Stack names**, include or exclude specific stacks from Hook invocations.

- For **Include**, specify the stack names to include. Use this when you have a small set of specific stacks you want to target. Only the stacks specified in this list will invoke the Hook.
- For **Exclude**, specify the stack names to exclude. Use this when you want to invoke the Hook on most stacks but exclude a few specific ones. All stacks except those listed here will invoke the Hook.

d. For **Stack roles**, include or exclude specific stacks from Hook invocations based on their associated IAM roles.

- For **Include**, specify one or more IAM role ARNs to target stacks associated with these roles. Only stack operations initiated by these roles will invoke the Hook.
- For **Exclude**, specify one or more IAM role ARNs for stacks you want to exclude. The Hook will be invoked on all stacks except those initiated by the specified roles.

14. Choose **Next**.
15. On the **Review and activate** page, review your choices. To make changes, choose **Edit** on the related section.
16. When you're ready to proceed, choose **Activate Hook**.

Activate a Lambda Hook (Amazon CLI)

Before you continue, confirm that you have created the Lambda function and the execution role that you'll use with this Hook. For more information, see Create Lambda functions to evaluate resources for Lambda Hooks and Create an execution role for a Lambda Hook.

To activate a Lambda Hook for use in your account (Amazon CLI)

1. To start activating a Hook, use the following activate-type command, replacing the placeholders with your specific values. This command authorizes the Hook to use a specified execution role from your Amazon Web Services account.

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

2. To finish activating the Hook, you must configure it using a JSON configuration file.

   Use the **cat** command to create a JSON file with the following structure. For more information, see Hook configuration schema syntax reference.

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

- `HookInvocationStatus`: Set to `ENABLED` to enable the Hook.
- `TargetOperations`: Specify the operations that the Hook will evaluate.
- `FailureMode`: Set to either `FAIL` or `WARN`.
- `LambdaFunction`: Specify the ARN of the Lambda function.
- `TargetFilters`: Specify the types of actions that will invoke the Hook.

3. Use the following set-type-configuration command, along with the JSON file you created, to apply the configuration. Replace the placeholders with your specific values.

```
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
```

Related resources

We provide template examples that you can use to understand how to declare a Lambda Hook in a CloudFormation stack template. For more information, see AWS::CloudFormation::LambdaHook in the *Amazon CloudFormation User Guide*.

View logs for the Lambda Hooks in your account

When using a Lambda Hook, your validation output report log file can be found in the Lambda console.

View Lambda Hook logs in the Lambda console

To view the Lambda Hook output log file

1. Sign-in to the Lambda console.
2. On the navigation bar at the top of the screen, choose your Amazon Web Services Region.
3. Choose **Functions**.
4. Choose desired Lambda function.
5. Choose the **Test** tab.
6. Choose **CloudWatch Logs Live Trail**
7. Choose the drop-down menu and select the log groups you want to view.

8. Choose **Start**. The log will display in the **CloudWatch Logs Live Trail** window. Choose **View in columns** or **View in plain text** depending on your preference.
    - You can add more filters to the results by adding them in the **Add filter pattern** field. This field allows you filter results to only include events that match the specified pattern.

For more information on viewing logs for Lambda functions, see Viewing CloudWatch Logs for Lambda functions.

Delete Lambda Hooks in your account

When you no longer need an activated Lambda Hook, use the following procedures to delete it in your account.

To temporarily disable a Hook instead of deleting it, see Disable and enable Amazon CloudFormation Hooks.

Topics

- Delete a Lambda Hook in your account (console)
- Delete a Lambda Hook in your account (Amazon CLI)

Delete a Lambda Hook in your account (console)

To delete a Lambda Hook in your account

1. Sign in to the Amazon Web Services Management Console and open the Amazon CloudFormation console at https://console.amazonaws.cn/cloudformation.
2. On the navigation bar at the top of the screen, choose the Amazon Web Services Region where the Hook is located.
3. From the navigation pane, choose **Hooks**.
4. On the **Hooks** page, find the Lambda Hook you want to delete.
5. Select the check box next to your Hook and choose **Delete**.
6. When prompted for confirmation, type out the Hook name to confirm deleting the specified Hook and then choose **Delete**.

Delete a Lambda Hook in your account (Amazon CLI)

> **Note**
>
> Before you can delete the Hook, you must first disable it. For more information, see Disable and enable a Hook in your account (Amazon CLI).

Use the following deactivate-type command to deactivate a Hook, which removes it from your account. Replace placeholders with your specific values.

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

Developing custom Hooks using the CloudFormation CLI

This section is for customers who want to develop custom Hooks and register them in the CloudFormation registry. It provides an overview of the structure of Amazon CloudFormation Hooks, and guides for developing, registering, testing, managing, and publishing your own Hooks with Python or Java.

There are three major steps in developing a custom Hook:

1. **Initiate**

   To develop custom Hooks, you must configure and use the CloudFormation CLI. To initiate a Hook's project and its required files, use the CloudFormation CLI init command and specify that you want to create a Hook. For more information, see Initiating a custom Amazon CloudFormation Hooks project.

2. **Model**

   To model, author, and validate your Hook schema, define the Hook, its properties, and their attributes.

   The CloudFormation CLI creates empty handler functions which correspond to a specific Hook invocation point. Add your own logic to these handlers to control what happens during your Hook invocation at each stage of its target lifecycle. For more information, see Modeling custom Amazon CloudFormation Hooks.

3. **Register**

   To register a Hook, submit your Hook to be registered either as a private or a public third-party extension. Register your Hook with the `submit` operation. For more information, see Registering a custom Hook with Amazon CloudFormation.

   The following tasks are associated with registering your Hook:

   a. *Publish* – Hooks are published to the registry.

   b. *Configure* – Hooks are configured when the type configuration invokes against stacks.

   > **Note**
   >
   > Hooks will time out after 30 seconds and retry up to 3 times. For more information, see Timeout and retry limits.

Topics

- Prerequisites for developing custom Amazon CloudFormation Hooks
- Initiating a custom Amazon CloudFormation Hooks project
- Modeling custom Amazon CloudFormation Hooks
- Registering a custom Hook with Amazon CloudFormation
- Testing a custom Hook in your Amazon Web Services account
- Updating a custom Hook
- Deregistering a custom Hook from the CloudFormation registry
- Publishing Hooks for public use
- Schema syntax reference for Amazon CloudFormation Hooks

Prerequisites for developing custom Amazon CloudFormation Hooks

You can develop a custom Hook with Java or Python. The following are the prerequisites for developing custom Hooks:

Java prerequisites

- Apache Maven

- JDK 17

> **Note**
>
> If you intend to use the CloudFormation Command Line Interface (CLI) to initiate a Hooks project for Java, you must install Python 3.8 or later as well. The Java plugin for the CloudFormation CLI can be installed through `pip` (Python's package manager), which is distrubted with Python.

To implement Hook handlers for your Java Hooks project, you can download the Java Hook handler example files.

Python prerequisites

- Python version 3.8 or later.

To implement Hook handlers for your Python Hooks project, you can download the Python Hook handler example files.

Permissions for developing Hooks

In addition to the CloudFormation `Create`, `Update`, and `Delete` stack permissions, you'll need access to the following Amazon CloudFormation operations. Access to these operations is managed through your IAM role's CloudFormation policy.

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

For more information, see Grant IAM permissions for CloudFormation Hooks.

Set up a development environment for Hooks

To develop Hooks, you should be familiar with CloudFormation templates, and either Python or Java.

To install the CloudFormation CLI, and the associated plugins:

1. Install the the CloudFormation CLI with `pip`, the Python package manager.

```
pip3 install cloudformation-cli
```

2. Install either the Python or Java plugin for the CloudFormation CLI.

   Python

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

   Java

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

To upgrade the CloudFormation CLI and the plugin, you can use the upgrade option.

Python

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

Java

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

Initiating a custom Amazon CloudFormation Hooks project

The first step in creating your custom Hooks project is to initiate the project. You can use the CloudFormation CLI `init` command to initiate your custom Hooks project.

The `init` command launches a wizard that walks you through setting up the project, including a Hooks schema file. Use this schema file as a starting point for defining the shape and semantics of your Hooks. For more information, see Schema syntax.

To inititate a Hook project:

1. Create a directory for the project.

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

2. Navigate to the new directory.

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

3. Use the CloudFormation CLI `init` command to initiate the project.

```
cfn init
```

The command returns the following output.

```
Initializing new project
```

4. The `init` command launches a wizard that walks you through setting up the project. When prompted, enter h to specify a Hooks project.

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

```
h
```

5. Enter a a name for your Hook type.

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

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

6. If only one language plugin is installed, it is selected by default. If more than one language plugin is installed, you can choose your desired language. Enter a number selection for the language of your choice.

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

7. Set up packaging based on chosen development lanaguage.

Python

(*Optional*) Choose Docker for platform-independent packaging. While Docker isn't required, it's highly recommended to make packaging easier.

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

Java

Set Java package name and choose a codegen model. You can use the default package name, or create a new one.

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

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

Results: You have successfully initiated the project and have generated the files needed to develop a Hook. The following is an example of the directories and files that make up a Hooks project for 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**
>
> The files in the `src` directory are created based on your language selection. There are some useful comments and examples in the generated files. Some files, such as `models.py`, are automatically updated in a later step when you run the `generate` command to add runtime code for your handlers.

Modeling custom Amazon CloudFormation Hooks

Modeling custom Amazon CloudFormation Hooks involves creating a schema that defines the Hook, its properties, and their attributes. When you create your custom Hook project using the `cfn init` command, an example Hook schema is created as a JSON-formatted text file, *hook-name*`.json`.

Target invocation points and target actions specify the exact point where the Hook is invoked. *Hook handlers* host executable custom logic for these points. For example, a target action of the `CREATE` operation uses a `preCreate` handler. Your code written in the handler will invoke when Hook targets and services perform a matching action. *Hook targets* are the destination where hooks are invoked. You can specify targets such as, Amazon CloudFormation public resources, private resources, or custom resources. Hooks support an unlimited number of Hook targets.

The schema contains permissions required for the Hook. Authoring the Hook requires you to specify permissions for each Hook handler. CloudFormation encourages authors to write policies that follow the standard security advice of granting *least privilege*, or granting only the permissions required to perform a task. Determine what users (and roles) need to do, and then craft policies that allow them to perform *only* those tasks for Hook operations. CloudFormation uses these permissions to scope-down Hook users provided permissions. These permissions are passed down to the Hook. Hook handlers use these permissions to access Amazon resources.

You can use the following schema file as a starting point to define your Hook. Use the Hook schema to specify which handlers you want to implement. If you choose not to implement a specific handler, remove it from the handlers' section of the Hook schema. For more details on the schema, see Schema syntax.

```
{
    "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
}
```

Topics

- Modeling custom Amazon CloudFormation Hooks using Java
- Modeling custom Amazon CloudFormation Hooks using Python

Modeling custom Amazon CloudFormation Hooks using Java

Modeling custom Amazon CloudFormation Hooks involves creating a schema that defines the Hook, its properties, and their attributes. This tutorial walks you through modeling custom Hooks using Java.

Step 1: Add project dependencies

Java based Hooks projects rely on Maven's `pom.xml` file as a dependency. Expand the following section and copy the source code into the `pom.xml` file in the root of the project.

Hook project dependencies (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>
```

Step 2: Generate the Hook project package

Generate your Hook project package. The CloudFormation CLI creates empty handler functions that correspond to specific Hook actions in the target lifecycle as defined in the Hook specification.

```
cfn generate
```

The command returns the following output.

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

> **Note**
>
> Make sure your Lambda runtimes are up-to-date to avoid using a deprecated version. For more information, see Updating Lambda runtimes for resource types and Hooks.

Step 3: Add Hook handlers

Add your own Hook handler runtime code to the handlers that you choose to implement. For example, you can add the following code for logging.

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

The CloudFormation CLI generates a Plain Old Java Objects (Java POJO). The following are output examples generated from `AWS::S3::Bucket`.

Example AwsS3BucketTargetModel.java

```
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;

}
```

With the POJOs generated, you can now write the handlers that actually implement the Hook's functionality. For this example, implement the `preCreate` and `preUpdate` invocation point for the handlers.

Step 4: Implement Hook handlers

Topics

- Coding the API client builder
- Coding the API request maker
- Implementing the helper code
- Implementing the base handler
- Implementing the preCreate handler
- Coding the preCreate handler
- Updating the preCreate test
- Implementing the preUpdate handler
- Coding the preUpdate handler
- Updating the preUpdate test
- Implementing the preDelete handler
- Coding the preDelete handler
- Updating the preDelete handler

Coding the API client builder

1. In your IDE, open the `ClientBuilder.java` file, located in the `src/main/java/com/mycompany/testing/mytesthook` folder.
2. Replace the entire contents of the `ClientBuilder.java` file with the following code.

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();
    }
}
```

Coding the API request maker

1. In your IDE, open the `Translator.java` file, located in the `src/main/java/com/mycompany/testing/mytesthook` folder.
2. Replace the entire contents of the `Translator.java` file with the following code.

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();
    }
}
```

Implementing the helper code

1. In your IDE, open the `AbstractTestBase.java` file, located in the `src/main/java/com/mycompany/testing/mytesthook` folder.

2. Replace the entire contents of the `AbstractTestBase.java` file with the following code.

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
                )
        );
```

```
    }
}
```

Implementing the base handler

1. In your IDE, open the `BaseHookHandlerStd.java` file, located in the `src/main/java/com/mycompany/testing/mytesthook` folder.
2. Replace the entire contents of the `BaseHookHandlerStd.java` file with the following code.

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);
        }
    }
}
```

Implementing the `preCreate` handler

The `preCreate` handler verifies the server-side encryption settings for either an `AWS::S3::Bucket` or `AWS::SQS::Queue` resource.

- For an `AWS::S3::Bucket` resource, the Hook will only pass if the following is true:
  - The Amazon S3 bucket encryption is set.
  - The Amazon S3 bucket key is enabled for the bucket.
  - The encryption algorithm set for the Amazon S3 bucket is the correct algorithm required.
  - The Amazon Key Management Service key ID is set.
- For an `AWS::SQS::Queue` resource, the Hook will only pass if the following is true:
  - The Amazon Key Management Service key ID is set.

Coding the `preCreate` handler

1. In your IDE, open the `PreCreateHookHandler.java` file, located in the `src/main/java/software/mycompany/testing/mytesthook` folder.

2. Replace the entire contents of the `PreCreateHookHandler.java` file with the following code.

```
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();
    }
}
```

Updating the `preCreate` test

1. In your IDE, open the `PreCreateHandlerTest.java` file, located in the `src/test/java/software/mycompany/testing/mytesthook` folder.
2. Replace the entire contents of `PreCreateHandlerTest.java` file with the following code.

```
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();
    }
}
```

Implementing the `preUpdate` handler

Implement a `preUpdate` handler, which initiates before the update operations for all specified targets in the handler. The `preUpdate` handler accomplishes the following:

- For an `AWS::S3::Bucket` resource, the Hook will only pass if the following is true:
  - The bucket encryption algorithm for an Amazon S3 bucket hasn't been modified.

Coding the `preUpdate` handler

1. In your IDE, open the `PreUpdateHookHandler.java` file, located in the `src/main/java/software/mycompany/testing/mytesthook` folder.
2. Replace the entire contents of the `PreUpdateHookHandler.java` file with the following code.

```
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();
    }
}
```

Updating the `preUpdate` test

1. In your IDE, open the `PreUpdateHandlerTest.java` file in the `src/main/java/com/mycompany/testing/mytesthook` folder.
2. Replace the entire contents of the `PreUpdateHandlerTest.java` file with the following code.

```
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();
    }
}
```

Implementing the `preDelete` handler

Implement a `preDelete` handler, which initiates before the delete operations for all specified targets in the handler. The `preDelete` handler accomplishes the following:

- For an `AWS::S3::Bucket` resource, the Hook will only pass if the following is true:
  - Verifies that the minimum required complaint resources will exist in the account after delete the resource.
  - The minimum required complaint resources amount is set in the Hook's type configuration.

Coding the `preDelete` handler

1. In your IDE, open the `PreDeleteHookHandler.java` file in the `src/main/java/com/mycompany/testing/mytesthook` folder.
2. Replace the entire contents of the `PreDeleteHookHandler.java` file with the following code.

```
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);
        }
    }
}
```

Updating the `preDelete` handler

1. In your IDE, open the `PreDeleteHookHandler.java` file in the `src/main/java/com/mycompany/testing/mytesthook` folder.
2. Replace the entire contents of the `PreDeleteHookHandler.java` file with the following code.

```
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();
    }
}
```

Modeling custom Amazon CloudFormation Hooks using Python

Modeling custom Amazon CloudFormation Hooks involves creating a schema that defines the Hook, its properties, and their attributes. This tutorial walks you through modeling custom Hooks using Python.

Step 1: Generate the Hook project package

Generate your Hook project package. The CloudFormation CLI creates empty handler functions that correspond to specific Hook actions in the target lifecycle as defined in the Hook specification.

```
cfn generate
```

The command returns the following output.

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

> **Note**
>
> Make sure your Lambda runtimes are up-to-date to avoid using a deprecated version. For more information, see Updating Lambda runtimes for resource types and Hooks.

Step 2: Add Hook handlers

Add your own Hook handler runtime code to the handlers that you choose to implement. For example, you can add the following code for logging.

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

The CloudFormation CLI generates the `src/models.py` file from the *Configuration schema*.

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
```

Step 3: Implement Hook handlers

With the Python data classes generated, you can write the handlers that actually implement the Hook's functionality. In this example, you'll implement the `preCreate`, `preUpdate`, and `preDelete` invocation points for the handlers.

Topics

- Implement the preCreate handler
- Implement the preUpdate handler
- Implement the preDelete handler
- Implement a Hook handler

Implement the preCreate handler

The `preCreate` handler verifies the server-side encryption settings for either an `AWS::S3::Bucket` or `AWS::SQS::Queue` resource.

- For an `AWS::S3::Bucket` resource, the Hook will only pass if the following is true.
  - The Amazon S3 bucket encryption is set.
  - The Amazon S3 bucket key is enabled for the bucket.
  - The encryption algorithm set for the Amazon S3 bucket is the correct algorithm required.
  - The Amazon Key Management Service key ID is set.
- For an `AWS::SQS::Queue` resource, the Hook will only pass if the following is true.
  - The Amazon Key Management Service key ID is set.

Implement the preUpdate handler

Implement a `preUpdate` handler, which initiates before the update operations for all specified targets in the handler. The `preUpdate` handler accomplishes the following:

- For an `AWS::S3::Bucket` resource, the Hook will only pass if the following is true:
  - The bucket encryption algorithm for an Amazon S3 bucket hasn't been modified.

Implement the preDelete handler

Implement a `preDelete` handler, which initiates before the delete operations for all specified targets in the handler. The `preDelete` handler accomplishes the following:

- For an `AWS::S3::Bucket` resource, the Hook will only pass if the following is true:
  - Verifies that the minimum required compliant resources will exist in the account after delete the resource.

- The minimum required compliant resources amount is set in the Hook's configuration.

Implement a Hook handler

1. In your IDE, open the `handlers.py` file, located in the `src` folder.
2. Replace the entire contents of the `handlers.py` file with the following code.

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}")
```

Continue to the next topic Registering a custom Hook with Amazon CloudFormation.

Registering a custom Hook with Amazon CloudFormation

Once you have created a custom Hook, you need to register it with Amazon CloudFormation so you can use it. In this section, you'll learn to package and register your Hook for use in your Amazon Web Services account.

Package a Hook (Java)

If you've developed your Hook with Java, use Maven to package it.

In the directory of your Hook project, run the following command to build your Hook, run unit tests, and package your project as a JAR file that you can use to submit your Hook to the CloudFormation registry.

```
mvn clean package
```

Register a custom Hook

To register a Hook

1. (Optional) Configure your default Amazon Web Services Region name to `us-west-2`, by submitting the configure operation.

   ```
   $ 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. (Optional) The following command builds and packages your Hook project without registering it.

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

3. Register your Hook by using the CloudFormation CLI submit operation.

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

   The command returns the following command.

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

Results: You've successfully registered your Hook.

Verifying Hooks are accessible in your account

Verify that your Hook is available in your Amazon Web Services account and in the Regions to which you have submitted it.

1. To verify your Hook, use the list-types command to list your newly registered Hook and return a summary description of it.

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

   The command returns the following output and will also show you publicly available Hooks you can activate in your Amazon Web Services account and Regions.

   ```
   {
       "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. Retrieve the `TypeArn` from the `list-type` output for your Hook and save it.

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

To learn how to publish Hooks for public use, see Publishing Hooks for public use.

Configure Hooks

After you've developed and registered your Hook, you can configure your Hook in your Amazon Web Services account by publishing it to the registry.

- To configure a Hook in your account, use the SetTypeConfiguration operation. This operation enables the Hook's properties that are defined in the Hook's schema `properties` section. In the following example, the `minBuckets` property is set to `1` in the configuration.

  > **Note**
  >
  > By enabling Hooks in your account, you are authorizing a Hook to use defined permissions from your Amazon Web Services account. CloudFormation removes non-required permissions before passing your permissions to the Hook. CloudFormation recommends customers or Hook users to review the Hook permissions and be aware of what permissions the Hooks are allowed to before enabling Hooks in your account.

  Specify the configuration data for your registered Hook extension in the same account and Amazon Web Services Region.

```
$ 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**
  >
  > To enable your Hook to proactively inspect the configuration of your stack, you must set the `HookInvocationStatus` to `ENABLED` in the `HookConfiguration` section, after the Hook has been registered and activated in your account.

Accessing Amazon APIs in handlers

If your Hooks uses an Amazon API in any of its handlers, the CFN-CLI automatically creates an IAM execution role template, `hook-role.yaml`. The `hook-role.yaml` template is based on the permissions specified for each handler in the handler's section of the Hook schema. If the `--role-`

`arn` flag is not used during the generate operation, the role in this stack will be provisioned and used as the execution role of the Hook.

For more information, see Accessing Amazon APIs from a resource type.

hook-role.yaml template

> **Note**
>
> If you choose to create your own execution role, we highly recommend practicing the principle of least privilege by allow listing only `hooks.cloudformation.amazonaws.com` and `resources.cloudformation.amazonaws.com`.

The following template uses the IAM, Amazon S3, and Amazon SQS permissions.

```
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
```

Testing a custom Hook in your Amazon Web Services account

Now that you've coded your handler functions that correspond to an invocation point, it's time to test your custom Hook on a CloudFormation stack.

The Hook failure mode is set to `FAIL` if the CloudFormation template didn't provision an S3 bucket with the following:

- The Amazon S3 bucket encryption is set.
- The Amazon S3 bucket key is enabled for the bucket.
- The encryption algorithm set for the Amazon S3 bucket is the correct algorithm required.
- The Amazon Key Management Service key ID is set.

In the following example, create a template called `my-failed-bucket-stack.yml` with a stack name of `my-hook-stack` that fails the stack configuration and stops before the resource provisions.

Testing Hooks by provisioning a stack

Example 1: To provision a stack

Provision a non-compliant stack

1. Author a template that specifies an S3 bucket. For example, `my-failed-bucket-stack.yml`.

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

2. Create a stack, and specify your template in the Amazon Command Line Interface (Amazon CLI). In the following example, specify the stack name as `my-hook-stack` and the template name as `my-failed-bucket-stack.yml`.

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

3. (Optional) View your stack progress by specifying your stack name. In the following example, specify the stack name `my-hook-stack`.

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

   Use the `describe-stack-events` operation to see the Hook failure while creating the bucket. The following is an example output of the command.

```
{
    "StackEvents": [
    ...
        {
            "StackId": "arn:aws-cn:cloudformation:us-west-2:123456789012: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"
        },
    ...
    ]
}
```

Results: The Hook invocation failed the stack configuration and stopped the resource from provisioning.

Use a CloudFormation template to pass Hook validation

1. To create a stack and pass the Hook validation, update the template so that your resource uses an encrypted S3 bucket. This example uses the template `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**
>
> Hooks won't be invoked for skipped resources.

2. Create a stack and specify your template. In this example, the stack name is `my-encrypted-bucket-stack`.

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

3. (Optional) View your stack progress by specifying the stack name.

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

Use the `describe-stack-events` command to view the response. The following is an example of the `describe-stack-events` command.

```
{
    "StackEvents": [
    ...
        {
            "StackId": "arn:aws-cn:cloudformation:us-west-2:123456789012: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-2-123456789012",
            "ResourceType": "AWS::S3::Bucket",
            "Timestamp": "2021-08-04T23:23:20.973000+00:00",
            "ResourceStatus": "CREATE_COMPLETE",
            "ResourceProperties": "{\"BucketName\":\"encryptedbucket-us-
west-2-123456789012\",\"BucketEncryption\":{\"ServerSideEncryptionConfiguration\":
[{\"BucketKeyEnabled\":\"true\",\"ServerSideEncryptionByDefault\":{\"SSEAlgorithm
\":\"aws:kms\",\"KMSMasterKeyID\":\"ENCRYPTION_KEY_ARN\"}}]}}",
            "ClientRequestToken": "Console-CreateStack-39df35ac-ca00-
b7f6-5661-4e917478d075"
        },
        {
            "StackId": "arn:aws-cn:cloudformation:us-west-2:123456789012: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-2-123456789012",
            "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-2-123456789012\",\"BucketEncryption\":{\"ServerSideEncryptionConfiguration\":
[{\"BucketKeyEnabled\":\"true\",\"ServerSideEncryptionByDefault\":{\"SSEAlgorithm
\":\"aws:kms\",\"KMSMasterKeyID\":\"ENCRYPTION_KEY_ARN\"}}]}}",
            "ClientRequestToken": "Console-CreateStack-39df35ac-ca00-
b7f6-5661-4e917478d075"
        },
        {
            "StackId": "arn:aws-cn:cloudformation:us-west-2:123456789012: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"
        },
    ...
    ]
}
```

Results: CloudFormation successfully created the stack. The Hook's logic verified that the `AWS::S3::Bucket` resource contained server-side encryption before provisioning the resource.

Example 2: To provision a stack

Provision a non-compliant stack

1. Author a template that specifies an S3 bucket. For example `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. Create a stack, and specify your template in the Amazon CLI. In the following example, specify the stack name as `my-hook-stack` and the template name as `aes256-bucket.yml`.

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

3. (Optional) View your stack progress by specifying your stack name. In the following example, specify the stack name `my-hook-stack`.

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

Use the `describe-stack-events` operation to see the Hook failure while creating the bucket. The following is an example output of the command.

```
{
    "StackEvents": [
    ...
        {
            "StackId": "arn:aws-cn:cloudformation:us-west-2:123456789012: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"
        },
    ...
    ]
}
```

Results: The Hook invocation failed the stack configuration and stopped the resource from provisioning. The stack failed due to the S3 bucket encryption configured incorrectly. The Hook type configuration requires `aws:kms` while this bucket uses AES256.

Use a CloudFormation template to pass Hook validation

1. To create a stack and pass the Hook validation, update the template so that your resource uses an encrypted S3 bucket. This example uses the template `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: !Sub 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**
>
> Hooks won't be invoked for skipped resources.

2. Create a stack and specify your template. In this example, the stack name is `my-encrypted-bucket-stack`.

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

3. (Optional) View your stack progress by specifying the stack name.

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

Use the `describe-stack-events` command to view the response. The following is an example of the `describe-stack-events` command.

```
{
    "StackEvents": [
    ...
        {
            "StackId": "arn:aws-cn:cloudformation:us-west-2:123456789012: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-2-123456789012",
            "ResourceType": "AWS::S3::Bucket",
            "Timestamp": "2021-08-04T23:23:20.973000+00:00",
            "ResourceStatus": "CREATE_COMPLETE",
            "ResourceProperties": "{\"BucketName\":\"encryptedbucket-us-
west-2-123456789012\",\"BucketEncryption\":{\"ServerSideEncryptionConfiguration\":
[{\"BucketKeyEnabled\":\"true\",\"ServerSideEncryptionByDefault\":{\"SSEAlgorithm
\":\"aws:kms\",\"KMSMasterKeyID\":\"ENCRYPTION_KEY_ARN\"}}]}}",
            "ClientRequestToken": "Console-CreateStack-39df35ac-ca00-
b7f6-5661-4e917478d075"
        },
        {
```

```
            "StackId": "arn:aws-cn:cloudformation:us-west-2:123456789012: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-2-123456789012",
            "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-2-123456789012\",\"BucketEncryption\":{\"ServerSideEncryptionConfiguration\":
[{\"BucketKeyEnabled\":\"true\",\"ServerSideEncryptionByDefault\":{\"SSEAlgorithm
\":\"aws:kms\",\"KMSMasterKeyID\":\"ENCRYPTION_KEY_ARN\"}}]}}",
            "ClientRequestToken": "Console-CreateStack-39df35ac-ca00-
b7f6-5661-4e917478d075"
        },
        {
            "StackId": "arn:aws-cn:cloudformation:us-west-2:123456789012: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"
        },
    ...
    ]
}
```

Results: CloudFormation successfully created the stack. The Hook's logic verified that the `AWS::S3::Bucket` resource contained server-side encryption before provisioning the resource.

Updating a custom Hook

Updating a custom Hook allows revisions in the Hook to be made available in the CloudFormation registry.

To update a custom Hook, submit your revisions to the CloudFormation registry through the CloudFormation CLI submit operation.

```
$ cfn submit
```

To specify the default version of your Hook in your account, use the set-type-default-version command and specify the type, type name, and version ID.

```
$ aws cloudformation set-type-default-version \
    --type HOOK \
    --type-name MyCompany::Testing::MyTestHook \
    --version-id 00000003
```

To retrieve information about the versions of a Hook, use list-type-versions.

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

Deregistering a custom Hook from the CloudFormation registry

Deregistering a custom Hook marks the extension or extension version as DEPRECATED in the CloudFormation registry, which removes it from active use. Once deprecated, the custom Hook can't be used in a CloudFormation operation.

> **Note**
>
> Before deregistering the Hook, you must individually deregister all previous active versions of that extension. For more information, see DeregisterType.

To deregister a Hook, use the deregister-type operation and specify your Hook ARN.

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

```
    --arn HOOK_TYPE_ARN
```

This command doesn't produce an output.

Publishing Hooks for public use

To develop a public third-party Hook, develop your Hook as a private extension. Then, in each Amazon Web Services Region in which you want to make the extension publicly available:

1. Register your Hook as a private extension in the CloudFormation registry.
2. Test your Hook to make sure it meets all necessary requirements for being published in the CloudFormation registry.
3. Publish your Hook to the CloudFormation registry.

> **Note**
>
> Before you publish any extension in a given Region, you must first register as an extension publisher in that Region. To do this in multiple Regions simultaneously, see Publishing extensions in multiple Regions using StackSets in the *Amazon CloudFormation CLI User Guide*.

After you've developed and registered your Hook, you can make it publicly available to general CloudFormation users by *publishing* it to the CloudFormation registry, as a third-party public extension.

Public third-party Hooks enable you to offer CloudFormation users to proactively inspect the configuration of Amazon resources before provisioning. As with private Hooks, public Hooks are treated the same as any Hook published by Amazon within CloudFormation.

Hooks published to the registry are visible by all CloudFormation users in the Amazon Web Services Regions in which they're published. Users can then *activate* your extension in their account, which makes it available for use in their templates. For more information, see Use third-party public extensions from the CloudFormation registry in the *Amazon CloudFormation User Guide*.

Testing a custom Hook for public use

In order to publish your registered custom Hook, it must pass all test requirements defined for it. The following is a list of requirements needed before publishing your custom Hook as a third-party extension.

Each handler and target is tested twice. Once for `SUCCESS` and once for `FAILED`.

- For `SUCCESS` response case:
  - Status must be `SUCCESS`.
  - Must not return an error code.
  - Callback delay should be set to `0` seconds, if specified.
- For `FAILED` response case:
  - Status must be `FAILED`.
  - Must return an error code.
  - Must have a message in response.
  - Callback delay should be set to `0` seconds, if specified.
- For `IN_PROGRESS` response case:
  - Must not return an error code.
  - `Result` field must not be set in response.

Specifying input data for use in contract tests

By default, the CloudFormation performs contract tests using input properties generated from the patterns you define in your Hook schema. However, most Hooks are complex enough that the input properties for precreating or preupdating provisioning stacks requires an understanding of the resource being provisioned. To address this, you can specify the input the CloudFormation uses when performing its contract tests.

CloudFormation offers two ways for you to specify the input data for it to use when performing contract tests:

- Overrides file

Using an `overrides` file provides a light-weight way of specifying input data for certain specific properties for the CloudFormation to use during `preCreate`, `preUpdate` and `preDelete` operations testing.

- Input files

  You can also use multiple `input` files to specify contract test input data if:

  - You want or need to specify different input data for create, update, and delete operations, or invalid data with which to test.
  - You want to specify multiple different input data sets.

Specifying input data using an override file

The following is an example of Amazon S3 Hook's input data using the `overrides` file.

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

Specifying input data using input files

Use `input` files to specify different kinds of input data for the CloudFormation to use: `preCreate` input, `preUpdate` input, and invalid input. Each kind of data is specified in a separate file. You can also specify multiple sets of input data for contract tests.

To specify `input` files for the CloudFormation to use in contract testing, add an `inputs` folder to the root directory of your Hooks project. Then add your input files.

Specify which kind of input data a file contains by using the following naming conventions, where *n* is an integer:

- `inputs_`*n*`_pre_create.json`: Use files with `preCreate` handlers for specifying inputs for creating the resource.
- `inputs_`*n*`_pre_update.json`: Use files with `preUpdate` handlers for specifying inputs for updating the resource.
- `inputs_`*n*`_pre_delete.json`: Use files with `preDelete` handlers for specifying inputs for deleting the resource.
- `inputs_`*n*`_invalid.json`: For specifying invalid inputs to test.

To specify multiple sets of input data for contract tests, increment the integer in the file names to order your input data sets. For example, your first set of input files should be named `inputs_1_pre_create.json`, `inputs_1_pre_update.json`, and `inputs_1_pre_invalid.json`. Your next set would be named `inputs_2_pre_create.json`, `inputs_2_pre_update.json`, and `inputs_2_pre_invalid.json`, and so on.

Each input file is a JSON file containing only the resource properties to be used in testing.

The following is an example directory for `inputs` for Amazon S3 specifying input data using input files.

`inputs_1_pre_create.json`

The following is an example of the `inputs_1_pre_create.json` contract test.

```
{
    "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`

The following is an example of the `inputs_1_pre_update.json` contract test.

```
{
    "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`

The following is an example of the `inputs_1_invalid.json` contract test.

```
{
    "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`

The following is an example of the `inputs_1_invalid_pre_update.json` contract test.

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

For more information, see Publishing extensions to make them available for public use in the *Amazon CloudFormation CLI User Guide*.

Schema syntax reference for Amazon CloudFormation Hooks

This section describes the syntax of the schema that you use to develop Amazon CloudFormation Hooks.

A Hook includes a Hook specification represented by a JSON schema and Hook handlers. The first step in creating a custom Hook is modeling a schema that defines the Hook, its properties, and their attributes. When you initialize a custom Hook project using the CloudFormation CLI init command, a Hook schema file is created for you. Use this schema file as a starting point for defining the shape and semantics of your custom Hook.

Schema syntax

The following schema is the structure for a 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`

The unique name for your Hook. Specifies a three-part namespace for your Hook, with a recommended pattern of `Organization::Service::Hook`.

> **Note**
>
> The following organization namespaces are reserved and can't be used in your Hook type names:
>
> - `Alexa`
> - `AMZN`
> - `Amazon`
> - `ASK`
> - `AWS`
> - `Custom`
> - `Dev`

Required: Yes

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

Minimum: `10`

Maximum: `196`

`description`

A short description of the Hook that's displayed in the CloudFormation console.

Required: Yes

`sourceUrl`

The URL of the source code for the Hook, if public.

Required: No

Maximum: `4096`

`documentationUrl`

The URL of a page providing detailed documentation for the Hook.

Required: Yes

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

Maximum: 4096

> **Note**
>
> Although the Hook schema should include complete and accurate property descriptions, you can use the `documentationURL` property to provide users with more details, including examples, use cases, and other detailed information.

`definitions`

Use the `definitions` block to provide shared Hook property schemas.

It's considered a best practice to use the `definitions` section to define schema elements that can be used at multiple points in your Hook type schema. You can then use a JSON pointer to reference that element at the appropriate places in your Hook type schema.

Required: No

`typeConfiguration`

The definition of a Hook's configuration data.

Required: Yes

`properties`

The properties of the Hook. All properties of a Hook must be expressed in the schema. Align the Hook schema properties with the Hook type configuration properties.

> **Note**
>
> Nested properties aren't allowed. Instead, define any nested properties in the `definitions` element, and use a `$ref` pointer to reference them in the desired property.

The following properties are currently supported:

- `default` – The default value of the property.
- `description` – A description of the property.
- `pattern` – A regex pattern used for validating the input.

- `type` – The accepted type of the property.

`additionalProperties`

`additionalProperties` must be set to `false`. All properties of a Hook must be expressed in the schema: arbitrary inputs aren't allowed.

Required: Yes

Valid values: `false`

`handlers`

Handlers specify the operations which can initiate the Hook defined in the schema, such as Hook invocation points. For example, a `preUpdate` handler is invoked before the update operations for all specified targets in the handler.

Valid values: `preCreate` | `preUpdate` | `preDelete`

> **Note**
>
> At least one value must be specified for the handler.

> **Important**
>
> Stack operations that result in the status of `UpdateCleanup` do not invoke a Hook. For example, during the following two scenarios, the Hook's `preDelete` handler is not invoked:
>
> - the stack is updated after removing one resource from the template.
> - a resource with the update type of replacement is deleted.

`targetNames`

A string array of type names that Hook targets. For example, if a `preCreate` handler has an `AWS::S3::Bucket` target, the Hook runs for Amazon S3 buckets during the preprovisioning phase.

- `TargetName`

  Specify at least one target name for each implemented handler.

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

Minimum: 1

Required: Yes

> **Warning**
>
> SSM SecureString and Secrets Manager dynamic references are not resolved before they are passed to Hooks.

`permissions`

A string array that specifies the Amazon permissions needed to invoke the handler.

Required: Yes

`additionalProperties`

`additionalProperties` must be set to `false`. All properties of a Hook must be expressed in the schema: arbitrary inputs aren't allowed.

Required: Yes

Valid values: `false`

Example Hooks schemas

Example 1

The Java and the Python walkthroughs use the following code example. The following is an example structure for a Hook called `mycompany-testing-mytesthook.json`.

```
{
    "typeName":"MyCompany::Testing::MyTestHook",
    "description":"Verifies S3 bucket and SQS queues properties before create and
 update",
    "sourceUrl":"https://mycorp.com/my-repo.git",
    "documentationUrl":"https://mycorp.com/documentation",
    "typeConfiguration":{
        "properties":{
            "minBuckets":{
```

```
                "description":"Minimum number of compliant buckets",
                "type":"string"
            },
            "minQueues":{
                "description":"Minimum number of compliant queues",
                "type":"string"
            },
            "encryptionAlgorithm":{
                "description":"Encryption algorithm for SSE",
                "default":"AES256",
                "type":"string",
                "pattern": "[a-zA-Z]*[1-9]"
            }
        },
        "required":[

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

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

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

```
                "s3:ListBucket",
                "s3:ListAllMyBuckets",
                "s3:GetEncryptionConfiguration",
                "sqs:ListQueues",
                "sqs:GetQueueAttributes",
                "sqs:GetQueueUrl"
            ]
        }
    },
    "additionalProperties":false
}
```

Example 2

The following example is a schema that uses the `STACK` and `CHANGE_SET` for `targetNames` to target a stack template and a change set operation.

```
{
    "typeName":"MyCompany::Testing::MyTestHook",
    "description":"Verifies Stack and Change Set properties before create and update",
    "sourceUrl":"https://mycorp.com/my-repo.git",
    "documentationUrl":"https://mycorp.com/documentation",
    "typeConfiguration":{
        "properties":{
            "minBuckets":{
                "description":"Minimum number of compliant buckets",
                "type":"string"
            },
            "minQueues":{
                "description":"Minimum number of compliant queues",
                "type":"string"
            },
            "encryptionAlgorithm":{
                "description":"Encryption algorithm for SSE",
                "default":"AES256",
                "type":"string",
                "pattern": "[a-zA-Z]*[1-9]"
            }
        },
        "required":[
        ],
        "additionalProperties":false
    },
```

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

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

Disable and enable Amazon CloudFormation Hooks

This topic describes how to disable and then re-enable a Hook to temporarily prevent it from being active in your account. Disabling Hooks can be useful when you need to investigate an issue without interference from Hooks.

Disable and enable a Hook in your account (console)

To disable a Hook in your account

1. Sign in to the Amazon Web Services Management Console and open the Amazon CloudFormation console at https://console.amazonaws.cn/cloudformation.
2. On the navigation bar at the top of the screen, choose the Amazon Web Services Region where the Hook is located.
3. From the navigation pane, choose **Hooks**.
4. Choose the name of the Hook you want to disable.
5. On the Hook details page, to the right of the Hook's name, choose the **Disable** button.
6. When prompted for confirmation, choose **Disable Hook**.

To re-enable a previously disabled Hook

1. Sign in to the Amazon Web Services Management Console and open the Amazon CloudFormation console at https://console.amazonaws.cn/cloudformation.
2. On the navigation bar at the top of the screen, choose the Amazon Web Services Region where the Hook is located.
3. From the navigation pane, choose **Hooks**.
4. Choose the name of the Hook you want to enable.
5. On the Hook details page, to the right of the Hook's name, choose the **Enable** button.
6. When prompted for confirmation, choose **Enable Hook**.

Disable and enable a Hook in your account (Amazon CLI)

> **Important**
>
> The Amazon CLI commands for disabling and enabling Hooks replace the entire Hook configuration with the values specified in the `--configuration` option. To avoid unintended changes, you must include all existing settings you wish to keep when running these commands. To view the current configuration data, use the describe-type command.

To disable a Hook

Use the following set-type-configuration command and specify `HookInvocationStatus` as `DISABLED` to disable the Hook. Replace the placeholders with your specific values.

```
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
```

To re-enable a previously disabled Hook

Use the following set-type-configuration command and specify `HookInvocationStatus` as `ENABLED` to re-enable the Hook. Replace the placeholders with your specific values.

```
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
```

For more information, see Hook configuration schema syntax reference.

View invocation results for Amazon CloudFormation Hooks

This topic describes how to view invocation results for Amazon CloudFormation Hooks. Viewing invocation results can help you understand how Hooks are evaluating your resources and resolve any issues that are detected when Hooks verify a resource.

Invocations are specific instances when your validation logic (whether it's a Amazon Control Tower proactive control, Guard rule, or Lambda function) runs during a resource's lifecycle.

View invocation results in the console

You can view invocation results in the console in three ways: through the **Invocation summary** page, through the invocation histories for individual Hooks, or through individual stack events for stack-specific invocations.

View results for all Hooks

The **Invocation summary** page provides a comprehensive view of all Hook invocations across your account and Region for the past 90 days.

To view results for all Hooks

1. Sign in to the Amazon Web Services Management Console and open the Amazon CloudFormation console at https://console.amazonaws.cn/cloudformation.
2. On the navigation bar at the top of the screen, choose the Amazon Web Services Region where you want to view Hook invocations.
3. From the navigation pane, choose **Invocation summary**.
4. The page displays a list of all Hook invocations from the past 90 days, including:

   - Invocation ID
   - Hook
   - Target
   - Mode (`Warn` or `Fail`)
   - Result (`Warning`, `Pass`, `Failed`, `In progress`)

- Invocation time
- Result message

5. You can filter the list using the search bar at the top of the table to find specific invocations.
6. Select a specific invocation to view more addition details about the invocation result, including remediation guidance for failed Hook invocations.

View invocation history for individual Hooks

You can also view invocation results through the invocation histories for individual Hooks.

To view Hook invocations for a specific Hook

1. Sign in to the Amazon Web Services Management Console and open the Amazon CloudFormation console at https://console.amazonaws.cn/cloudformation.
2. On the navigation bar at the top of the screen, choose the Amazon Web Services Region where you want to view Hook invocations.
3. From the navigation pane, choose **Hooks**.
4. Choose the Hook you want to view Hook invocations for.
5. Select a specific invocation to view more addition details about the invocation result, including remediation guidance for failed Hook invocations.

View results for stack-specific invocations

You can also view invocation results for a specific stack through the stack **Events** page.

To view Hook invocations for a specific stack

1. Sign in to the Amazon Web Services Management Console and open the Amazon CloudFormation console at https://console.amazonaws.cn/cloudformation.
2. On the navigation bar at the top of the screen, choose the Amazon Web Services Region where the stack operation occurred.
3. From the navigation pane, choose **Stacks**.
4. Select the stack you want to view Hook invocations for.
5. Choose the **Stack events** tab.

6. In the events list, look for events with **Hook invocations complete** in the **Status reason** column.
7. To view specific Hook invocation details, review the **Hook invocations** column, and choose the underlined text to open a pop-up window with more detailed information.

> **Note**
>
> To display hidden columns, choose the gear icon on the top right corner of the section to open the **Preferences** modal, update the settings as needed, and choose **Confirm**.

View invocation results using the Amazon CLI

Use the list-hook-results command to retrieve information about Hook invocations. This command supports the following filtering options:

- Get all Hook invocation results (no parameters required)
- Filter by Hook ARN (use `--type-arn`)
- Filter by Hook ARN and status (use `--type-arn` and `--status`)
- Search for specific targets (use `--target-type` and `--target-id`)

Filter results by Hook ARN

The following command lists all Hook invocation results for a specific Hook.

```
aws cloudformation list-hook-results \
  --type-arn arn:aws:cloudformation:us-west-2:123456789012:type/hook/MyOrg-Security-ComplianceHook \
  --region us-west-2
```

Example output:

```
{
  "HookResults": [
    {
      "TypeArn": "arn:aws:cloudformation:us-west-2:123456789012:type/hook/MyOrg-Security-ComplianceHook",
      "HookResultId": "59ef501c-0ac4-47c0-a193-e071cabf748d",
```

```
      "TypeName": "MyOrg::Security::ComplianceHook",
      "TypeVersionId": "00000001",
      "HookExecutionTarget": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-
stack/39f29d10-73ed-11f0-abc1-0affdfe4aebb",
      "InvokedAt": "2025-08-08T00:18:39.651Z",
      "FailureMode": "WARN",
      "HookStatusReason": "...",
      "InvocationPoint": "PRE_PROVISION",
      "Status": "HOOK_COMPLETE_FAILED"
    },
    ...
  ]
}
```

For a description of the fields in the response, see HookResultSummary in the *Amazon CloudFormation API Reference*.

Filter results by Hook ARN and status

To filter for a common status among the results, specify the `--status` option in the command. Valid values are:

- `HOOK_IN_PROGRESS`: The Hook is currently running.
- `HOOK_COMPLETE_SUCCEEDED`: The Hook completed successfully.
- `HOOK_COMPLETE_FAILED`: The Hook completed but failed validation.
- `HOOK_FAILED`: The Hook encountered an error during execution.

```
aws cloudformation list-hook-results \
  --type-arn arn:aws:cloudformation:us-west-2:123456789012:type/hook/MyOrg-Security-
ComplianceHook \
  --status HOOK_COMPLETE_FAILED \
  --region us-west-2
```

Example output:

```
{
  "HookResults": [
    {
      "TypeArn": "arn:aws:cloudformation:us-west-2:123456789012:type/hook/MyOrg-
Security-ComplianceHook",
```

```
      "HookResultId": "59ef501c-0ac4-47c0-a193-e071cabf748d",
      "TypeName": "MyOrg::Security::ComplianceHook",
      "TypeVersionId": "00000001",
      "HookExecutionTarget": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-
stack/39f29d10-73ed-11f0-abc1-0affdfe4aebb",
      "InvokedAt": "2025-08-08T00:18:39.651Z",
      "FailureMode": "WARN",
      "HookStatusReason": "...",
      "InvocationPoint": "PRE_PROVISION",
      "Status": "HOOK_COMPLETE_FAILED"
    },
    ...
  ]
}
```

For a description of the fields in the response, see HookResultSummary in the *Amazon CloudFormation API Reference*.

Filter results by target type and target ID

The following command lists all Hook invocation results for a specific Cloud Control API request.

```
aws cloudformation list-hook-results \
  --target-type CLOUD_CONTROL \
  --target-id d417b05b-9eff-46ef-b164-08c76aec1801 \
  --region us-west-2
```

Example output:

```
{
  "HookResults": [
    {
      "TargetType": "CLOUD_CONTROL",
      "TargetId": "d417b05b-9eff-46ef-b164-08c76aec1801",
      "HookResults": [
        {
          "TypeArn": "arn:aws:cloudformation:us-west-2:123456789012:type/hook/MyOrg-
Security-ComplianceHook",
          "HookResultId": "4e7f4766-d8fe-44e5-8587-5b327a148abe",
          "TypeName": "MyOrg::Security::ComplianceHook",
          "TypeVersionId": "00000001",
          "FailureMode": "WARN",
          "HookStatusReason": "...",
```

```
          "InvocationPoint": "PRE_PROVISION",
          "Status": "HOOK_COMPLETE_FAILED"
        },
        ...
      ]
    }
  ]
}
```

For a description of the fields in the response, see HookResultSummary in the *Amazon CloudFormation API Reference*.

Get detailed results for a specific invocation

Use the get-hook-result command to retrieve detailed information about a specific Hook invocation, including annotations with compliance check results and remediation guidance.

```
aws cloudformation get-hook-result \
  --hook-result-id 59ef501c-0ac4-47c0-a193-e071cabf748d \
  --region us-west-2
```

Example output:

```
{
  "HookResultId": "59ef501c-0ac4-47c0-a193-e071cabf748d",
  "InvocationPoint": "PRE_PROVISION",
  "FailureMode": "WARN",
  "TypeName": "MyOrg::Security::ComplianceHook",
  "TypeVersionId": "00000001",
  "TypeArn": "arn:aws:cloudformation:us-west-2:123456789012:type/hook/MyOrg-Security-
ComplianceHook",
  "Status": "HOOK_COMPLETE_FAILED",
  "HookStatusReason": "Hook completed with failed validations",
  "InvokedAt": "2025-08-08T00:18:39.651Z",
  "Target": {
    "TargetType": "RESOURCE",
    "TargetTypeName": "AWS::S3::Bucket",
    "TargetId": "my-s3-bucket",
    "Action": "CREATE"
  },
  "Annotations": [
    {
```

```
      "AnnotationName": "BlockPublicAccessCheck",
      "Status": "FAILED",
      "StatusMessage": "Bucket does not block public access",
      "RemediationMessage": "Enable block public access settings on the S3 bucket",
      "SeverityLevel": "HIGH"
    },
    {
      "AnnotationName": "BucketEncryptionCheck",
      "Status": "PASSED",
      "StatusMessage": "Bucket has encryption configured correctly"
    }
  ]
}
```

For a description of the fields in the response, see GetHookResult in the *Amazon CloudFormation API Reference*.

Hook configuration schema syntax reference

This section outlines the schema syntax used to configure Hooks. CloudFormation uses this configuration schema at runtime when invoking a Hook in an Amazon Web Services account.

To enable your Hook to proactively inspect the configuration of your stack, set the `HookInvocationStatus` to `ENABLED` after the Hook has been registered and activated in your account.

Topics

- Hook configuration schema properties
- Hook configuration examples
- Amazon CloudFormation Hooks stack level filters
- Amazon CloudFormation Hooks target filters
- Using wildcards with Hook target names

> **Note**
>
> The maximum amount of data that a Hook's configuration can store is 300 KB. This is in addition to all the constraints imposed on `Configuration` request parameter of SetTypeConfiguration operation.

Hook configuration schema properties

The following schema is the structure for a Hook configuration schema.

```
{
    "CloudFormationConfiguration": {
        "HookConfiguration": {
            "HookInvocationStatus": "ENABLED",
            "TargetOperations": ["STACK"],
            "FailureMode": "FAIL",
            "EncryptionConfiguration": {
                "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/abc-123"
            },
            "Properties": {
```

```
                ...
            }
        }
    }
}
```

`HookConfiguration`

Hook configuration supports activating or deactivating Hooks at stack level, failure modes, and Hook properties values.

The Hook configuration supports the following properties.

`HookInvocationStatus`

Specifies if the Hook is `ENABLED` or `DISABLED`.

Valid values: `ENABLED` | `DISABLED`

`TargetOperations`

Specifies the list of operations the Hook is run against. For more information, see Hook targets.

Valid values: `STACK` | `RESOURCE` | `CHANGE_SET` | `CLOUD_CONTROL`

`TargetStacks`

Available for backward compatibility. Use `HookInvocationStatus` instead.

If the mode is set to `ALL`, the Hook applies to all stacks in your account during a `CREATE`, `UPDATE`, or `DELETE` resource operation.

If the mode is set to `NONE`, the Hook won't apply to stacks in your account.

Valid values: `ALL` | `NONE`

`FailureMode`

This field tells the service how to treat Hook failures.

- If the mode is set to `FAIL`, and the Hook fails, then the fail configuration stops provisioning resources and rolls back the stack.
- If the mode is set to `WARN` and the Hook fails, then the warn configuration allows provisioning to continue with a warning message.

Valid values: `FAIL` | `WARN`

`EncryptionConfiguration`

Specifies encryption settings for Hook annotations data.

`KmsKeyId`

The alias, alias ARN, key ID, or key ARN of the symmetric encryption Amazon KMS key used to encrypt Hook annotations data. For more information, see KeyId in the Amazon KMS docs.

Before you can create Hooks with customer managed Amazon KMS keys, your user or role must have Amazon KMS permissions to `DescribeKey` and `GenerateDataKey`. For more information, see Amazon KMS key policy and permissions for encrypting Amazon CloudFormation Hooks results at rest.

`Properties`

Specifies Hook runtime properties. These should match the shape of the properties supported by Hooks schema.

Hook configuration examples

For examples of configuring Hooks from the Amazon CLI, see the following sections:

- Activate a proactive control-based Hook (Amazon CLI)
- Activate a Guard Hook (Amazon CLI)
- Activate a Lambda Hook (Amazon CLI)

Amazon CloudFormation Hooks stack level filters

You can add stack level filters to your CloudFormation Hooks to target specific stacks based on stack names and roles. This is useful in cases where you have multiple stacks with the same resource types, but the Hook is intended for specific stacks.

This section explains how these filters work and provides examples you can follow.

The basic structure of a Hook configuration without stack level filtering looks like this:

```
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "TargetFilters": {
        "Actions": [
          "CREATE",
          "UPDATE",
          "DELETE"
        ]
      }
    }
  }
}
```

For more information about the `HookConfiguration` syntax, see Hook configuration schema syntax reference.

To use stack level filters, add a `StackFilters` key under `HookConfiguration`.

The `StackFilters` key has one required member and has two optional members.

- `FilteringCriteria` (required)
- `StackNames` (optional)
- `StackRoles` (optional)

The `StackNames` or `StackRoles` properties are optional. However, you must specify at least one of these properties.

If you create a Hook that targets Cloud Control API operations, all stack level filters will be ignored.

`FilteringCriteria`

`FilteringCriteria` is a required parameter that specifies the filtering behavior. It can be set to either `ALL` or `ANY`.

- ALL invokes the Hook if all the filters are matched.
- ANY invokes the Hook if any one filter is matched.

StackNames

To specify one or more stack names as filters in your Hooks configuration, use the following JSON structure:

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

You must specify one of the following:

- Include: List of stack names to include. Only the stacks specified in this list will invoke the Hook.
  - Type: Array of strings
  - Max items: 50
  - Min items: 1
- Exclude: List of stack names to exclude. All stacks except those listed here will invoke the Hook.
  - Type: Array of strings
  - Max items: 50
  - Min items: 1

Each stack name in the Include and Exclude arrays must adhere to the following pattern and length requirements:

- Pattern: ^[a-zA-Z][-a-zA-Z0-9]*$
- Max length: 128

`StackNames` support concrete stack names and full wildcard matching. To see examples using wildcards, see Using wildcards with Hook target names.

`StackRoles`

To specify one or more IAM roles as filters in your Hook configuration, use the following JSON structure:

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

You must specify one of the following:

- `Include`: List of IAM role ARNs to target stacks associated with these roles. Only stack operations initiated by these roles will invoke the Hook.
  - Type: Array of strings
  - Max items: 50
  - Min items: 1
- `Exclude`: List of IAM role ARNs for stacks you want to exclude. The Hook will be invoked on all stacks except those initiated by the specified roles.
  - Type: Array of strings
  - Max items: 50
  - Min items: 1

Each stack role in the `Include` and `Exclude` arrays must adhere to the following pattern and length requirements:

- Pattern: `arn:.+:iam::[0-9]{12}:role/.+`
- Max length: 256

`StackRoles` allow wildcard characters in the following ARN syntax sections:

- `partition`
- `account-id`
- `resource-id`

To see examples using wildcards in the ARN syntax sections, see Using wildcards with Hook target names.

`Include` and `Exclude`

Each filter (`StackNames` and `StackRoles`) has an `Include` list and `Exclude` list. Using `StackNames` as an example, the Hook is only invoked on the stacks that are specified in `Include` list. If stack names are only specified in the `Exclude` list, the hook is only invoked on stacks that are *not* in the `Exclude` list. If both `Include` and `Exclude` are specified, the Hook targets what's in the `Include` list and not what's in the `Exclude` list.

For example, suppose you have four stacks: A, B, C, and D.

- `"Include": ["A","B"]` The Hook is invoked on A and B.
- `"Exclude": ["B"]` The Hook is invoked on A, C, and D.
- `"Include": ["A","B","C"], "Exclude": ["A","D"]` The Hook is invoked on B and C.
- `"Include": ["A","B","C"], "Exclude": ["A”,"B","C"]` The Hook is not invoked on any stack.

Examples of stack level filters

This section provides examples you can follow to create stack level filters for Amazon CloudFormation Hooks.

Example 1: Include specific stacks

The following example specifies an `Include` list. The Hook is only invoked on stacks named `stack-test-1`, `stack-test-2` and `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"
          ]
        }
      }
    }
  }
}
```

Example 2: Exclude specific stacks

If the stack names are instead added to the `Exclude` list, the Hook is invoked on any stack that is *not* named `stack-test-1`, `stack-test-2` or `stack-test-3`.

```
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "StackFilters": {
        "FilteringCriteria": "ALL",
        "StackNames": {
          "Exclude": [
            "stack-test-1",
            "stack-test-2",
            "stack-test-3"
          ]
        }
```

```
      }
    }
  }
}
```

Example 3: Combining include and exclude

If `Include` and `Exclude` lists aren't specified, the Hook is only invoked on the stacks in the `Include` that aren't in the `Exclude` list. In the following example, the Hook is only invoked on `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"
          ]
        }
      }
    }
  }
}
```

Example 4: Combining stack names and roles with ALL criteria

The following Hook includes three stack names, and one stack role. Because the `FilteringCriteria` is specified as `ALL`, the Hook is only invoked for stack that have *both* a matching stack name *and* the matching stack role.

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

Example 5: Combining stack names and roles with ANY criteria

The following Hook includes three stack names, and one stack role. Because the `FilteringCriteria` is specified as `ANY`, the Hook is invoked for stack that have *either* a matching stack name *or* the matching stack role.

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

Amazon CloudFormation Hooks target filters

This topic provides guidance on configuring target filters for Amazon CloudFormation Hooks. You can use target filters for more granular control over when and on which resources your Hook is invoked. You can configure filters ranging from simple resource type targeting to more complex combinations of resource types, actions, and invocation points.

To specify one or more stack names as filters in your Hooks configuration, add a `TargetFilters` key under `HookConfiguration`.

`TargetFilters` supports the following properties.

`Actions`

A string array that specifies the actions to target. For an example, see Example 1: Basic target filter.

Valid values: CREATE | UPDATE | DELETE

> **Note**
>
> For RESOURCE, STACK, and CLOUD_CONTROL targets, all target actions are applicable. For CHANGE_SET targets, only the CREATE action is applicable. For more information, see Hook targets.

`InvocationPoints`

A string array that specifies the invocation points to target.

Valid values: `PRE_PROVISION`

`TargetNames`

A string array that specifies the resource type names to target, for example, `AWS::S3::Bucket`.

Target names support concrete target names and full wildcard matching. For more information, see Using wildcards with Hook target names.

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

Maximum: `50`

`Targets`

An object array that specifies the list of targets to use for target filtering.

Each target in the targets array has the following properties.

`Actions`

The action for the specified target.

Valid values: `CREATE` | `UPDATE` | `DELETE`

`InvocationPoints`

The invocation point for the specified target.

Valid values: `PRE_PROVISION`

`TargetNames`

The resource type name to target.

> **Note**
>
> You can't include both the `Targets` object array and the `TargetNames`, `Actions`, or `InvocationPoints` arrays at the same time. If you want to use these three items and `Targets`, you must include them within the `Targets` object array. For an example, see Example 2: Using the `Targets` object array.

Examples of target filters

This section provides examples you can follow to create target filters for Amazon CloudFormation Hooks.

Example 1: Basic target filter

To create a basic target filter that focuses on specific resource types, use the `TargetFilters` object with the `Actions` array. The following target filter configuration will invoke the Hook on all `Create`, `Update`, and `Delete` actions for the specified target operations (in this case, both `RESOURCE` and `STACK` operations).

```
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "TargetFilters": {
        "Actions": [
           "Create",
           "Update",
           "Delete"
        ]
      }
    }
  }
}
```

Example 2: Using the `Targets` object array

For more advanced filters, you can use the `Targets` object array to list specific target, action, and invocation point combinations. This following target filter configuration will invoke the Hook before CREATE and UPDATE actions on S3 buckets and DynamoDB tables. It applies to both STACK and RESOURCE operations.

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

```
}
```

Using wildcards with Hook target names

You can use wildcards as part of the target name. You can use wildcard characters (* and ?) within your Hook target names. The asterisk (*) represents any combination of characters. The question mark (?) represents any single character. You can use multiple * and ? characters in a target name.

Example : Examples of target name wildcards in Hook schemas

The following example targets all resource types supported by Amazon S3.

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

The following example matches all resource types that have "Bucket" in the name.

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

The `AWS::*::Bucket*` might resolve to any of the following concrete resource types:

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

Example : Examples of target name wildcards in Hook configuration schemas

The following example configuration invokes the Hook for CREATE operations on all Amazon S3 resource types, and for UPDATE operations on all named table resource types, such as AWS::DynamobDB::Table or 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"
                    }
                ]
            }
        }
   }
}
```

The following example configuration invokes the Hook for CREATE and UPDATE operations on all Amazon S3 resource types, and also for CREATE and UPDATE operations on all named table resource types, such as AWS::DynamobDB::Table or AWS::Glue::Table.

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

Example : `Include` specific stacks

The following examples specifies an `Include` list. The Hook is only invoked if the stack names begins with `stack-test-`.

```
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "StackFilters": {
        "FilteringCriteria": "ALL",
        "StackNames": {
          "Include": [
            "stack-test-*"
```

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

Example : Exclude specific stacks

The following examples specifies an `Exclude` list. The Hook is invoked on any stack that does not begin with `stack-test-`.

```
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "StackFilters": {
        "FilteringCriteria": "ALL",
        "StackNames": {
          "Exclude": [
            "stack-test-*"
          ]
        }
      }
    }
  }
}
```

Example : Combining `Include` and `Exclude` for specific stacks

If `Include` and `Exclude` lists are specified, the Hook is only invoked on stacks matching in the `Include` that do not match in the `Exclude` list. In the following example, the Hook is invoked on all stacks that begin with `stack-test-` except for stacks named `stack-test-1`, `stack-test-2`, and `stack-test-3`.

```
{
```

```
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "StackFilters": {
        "FilteringCriteria": "ALL",
        "StackNames": {
          "Include": [
            "stack-test-*"
          ],
          "Exclude": [
            "stack-test-1",
            "stack-test-2",
            "stack-test-3"
          ]
        }
      }
    }
  }
}
```

Example : `Include` specific roles

The following example specifies an `Include` list with two wildcard patterns. The first entry will run the Hook for any role that begins with `hook-role` in any `partition` and `account-id`. The second entry will run any for any role in any `partition` that belongs to `account-id` `123456789012`.

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

```
      "StackFilters": {
        "FilteringCriteria": "ALL",
        "StackRoles": {
          "Include": [
            "arn:*:iam::*:role/hook-role*",
            "arn:*:iam::123456789012:role/*
          ]
        }
      }
    }
  }
}
```

Example : `Exclude` specific roles

The following examples specifies an `Exclude` list with two wildcard patterns. The first entry will skip Hook execution when a role has `exempt` in its name in any `partition` and any `account-id`. The second entry will skip Hook execution when a role belonging to `account-id` `123456789012` is used with the stack operation.

```
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "StackFilters": {
        "FilteringCriteria": "ALL",
        "StackRoles": {
          "Exclude": [
            "arn:*:iam::*:role/*exempt*",
            "arn:*:iam::123456789012:role/*
          ]
        }
      }
    }
  }
}
```

Example : Combining `Include` and `Exclude` for specific role ARN patterns

If `Include` and `Exclude` lists are specified, the Hook is only invoked on stacks used with roles that match those in `Include` that do not match in the `Exclude` list. In the following example, the Hook is invoked on stack operations with any `partition`, `account-id`, and `role` name, except if the role belongs to `account-id` `123456789012`.

```
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "StackFilters": {
        "FilteringCriteria": "ALL",
        "StackRoles": {
          "Include": [
            "arn:*:iam::*:role/*"
          ],
          "Exclude": [
            "arn:*:iam::123456789012:role/*"
          ]
        }
      }
    }
  }
}
```

Example : Combining stack names and roles with all criteria

The following Hook includes one stack name wildcard and one stack role wildcard. Because the `FilteringCriteria` is specified as `ALL`, the Hook is only invoked for stacks that have both, the matching `StackName` and matching `StackRoles`.

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

```
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "StackFilters": {
        "FilteringCriteria": "ALL",
        "StackNames": {
          "Include": [
            "stack-test-*"
          ]
        },
        "StackRoles": {
          "Include": ["arn:*:iam::*:role/hook-role*"]
        }
      }
    }
  }
}
```

Example : Combining `StackNames` and `StackRoles` with any criteria

The following Hook includes one stack name wildcard and one stack role wildcard. Because the `FilteringCriteria` is specified as ANY, the Hook is invoked for the stack that have either matching `StackNames` or matching `StackRoles`.

```
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "StackFilters": {
        "FilteringCriteria": "ANY",
        "StackNames": {
          "Include": [
            "stack-test-*"
          ]
```

```
        },
        "StackRoles": {
            "Include": ["arn:*:iam::*:role/hook-role*"]
        }
      }
    }
  }
}
```

Create Hooks using CloudFormation templates

This page provides links to sample CloudFormation templates and technical reference topics for Hooks.

By using CloudFormation templates to create Hooks, you can reuse your template to set up your Hooks consistently and repeatedly. This approach allows you to define your Hooks once, and then provision the same Hooks over and over in multiple Amazon Web Services accounts and Regions.

CloudFormation offers the following specialized resource types for Guard and Lambda Hook creation.

Task	Solution	Links
Create a Guard Hook	Use the `AWS::CloudFormation::GuardHook` resource type to create and activate a Guard Hook.	Sample template Technical reference
Create a Lambda Hook	Use the `AWS::CloudFormation::LambdaHook` resource type to create and activate a Lambda Hook.	Sample template Technical reference

CloudFormation also offers the following resource types that you can use in your stack templates for custom Hook creation.

Task	Solution	Links
Register a Hook	Use the `AWS::CloudFormation::HookVersion` resource type to publish a new or first version of a custom Hook to the CloudFormation registry.	Sample templates Technical reference
Set the Hook's configuration	Use the `AWS::CloudFormation::HookTypeConfig` resource type to specify the configuration of a custom Hook.	Sample templates Technical reference

Task	Solution	Links
Set the Hook's default version	Use the `AWS::CloudFormation::HookDefaultVersion` resource type to specify the default version of a custom Hook.	Sample templates Technical reference
Register your account as a publisher	Use the `AWS::CloudFormation::Publisher` resource type to register your account as a publisher of public extensions (Hooks, modules, and resource types) in the CloudFormation registry.	Technical reference
Publish a Hook publicly	Use the `AWS::CloudFormation::PublicTypeVersion` resource type to test and publish a registered custom Hook as a public, third-party Hook.	Technical reference
Activate public, third-party Hooks	The `AWS::CloudFormation::TypeActivation` resource type works together with the `AWS::CloudFormation::HookTypeConfig` resource type to activate a public, third-party custom Hook in your account.	Technical reference

Grant IAM permissions for CloudFormation Hooks

By default, a brand new user in your Amazon Web Services account doesn't have permission to manage Hooks using the Amazon Web Services Management Console, Amazon Command Line Interface (Amazon CLI), or Amazon API. To grant users permission, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

Use the policy examples in this topic to create your own custom IAM policies to give users permissions to work with Hooks.

To learn how to create an IAM identity-based policy using these example JSON policy documents, see Define custom IAM permissions with customer managed policies in the *IAM User Guide*.

This topic covers the permissions that are needed to do the following:

- **Manage Hooks** – Create, modify, and disable Hooks in your account.
- **Publish Hooks publicly** – Register, test, and publish your custom Hooks to make them available publicly in the CloudFormation registry.
- **View invocation results** – Access and query the results of Hook invocations in your account.
- **View details for an invocation result** – Access detailed information and remediation guidance for a specific Hook invocation result in your account.

As you create your IAM policies, you can find documentation for all of the actions, resources, and condition keys associated with the `cloudformation` service prefix in the Actions, resources, and condition keys for Amazon CloudFormation section of the *Service Authorization Reference*.

Topics

- Allow a user to manage Hooks
- Allow a user to publish custom Hooks publicly
- Allow a user to view Hook invocation results
- Allow a user to view detailed Hook invocation results
- Amazon KMS key policy and permissions for encrypting Amazon CloudFormation Hooks results at rest

Allow a user to manage Hooks

If you need to allow users to manage extensions, including Hooks, without the ability to make them public in the CloudFormation registry, you can use the following example IAM policy.

> **Important**
>
> The `ActivateType` and `SetTypeConfiguration` API calls work together to create Hooks in your account. When you grant a user permission to call the `SetTypeConfiguration` API, you automatically grant them the ability to modify and disable existing Hooks. You can't use resource-level permissions to restrict access to this API call. Therefore, ensure that you grant this permission only to authorized users in your account.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudformation:ActivateType",
                "cloudformation:DescribeType",
                "cloudformation:ListTypes",
                "cloudformation:SetTypeConfiguration"
            ],
            "Resource": "*"
        }
    ]
}
```

Users who manage Hooks might need some related permissions, for example:

- To view proactive controls from the Control Catalog in the CloudFormation console, the user must have the `controlcatalog:ListControls` permission in an IAM policy.
- To register custom Hooks as private extensions in the CloudFormation registry, the user must have the `cloudformation:RegisterType` permission in an IAM policy.

Allow a user to publish custom Hooks publicly

The following example IAM policy focuses specifically on publishing capabilities. Use this policy if you need to allow users to make extensions, including Hooks, available publicly in the CloudFormation registry.

> **Important**
>
> Publishing Hooks publicly makes them available to other Amazon Web Services accounts. Ensure that only authorized users have these permissions and that published extensions meet your organization's quality and security standards.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudformation:DescribePublisher",
                "cloudformation:DescribeTypeRegistration",
                "cloudformation:ListTypes",
                "cloudformation:ListTypeVersions",
                "cloudformation:PublishType",
                "cloudformation:RegisterPublisher",
                "cloudformation:RegisterType",
                "cloudformation:TestType"
            ],
            "Resource": "*"
        }
    ]
}
```

Allow a user to view Hook invocation results

The IAM permissions needed to view Hook invocation results change depending on the type of information being requested.

List Hook invocation results

To list Hook invocation results, users need different permissions depending on the API request being made.

- To grant permissions to request all Hook results, results for a specific Hook, or results for a specific Hook and invocation status, you must grant access to the `cloudformation:ListAllHookResults` action.
- To grant permissions to request results by specifying a Hook target, you must grant access to the `cloudformation:ListHookResults` action. This permission allows the API caller to specify the `TargetType` and `TargetId` parameters when calling `ListHookResults`.

The following shows an example of a basic permissions policy for listing Hook invocation results. IAM identities (users or roles) with this policy have permission to request all invocation results using all available parameter combinations.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudformation:ListAllHookResults",
                "cloudformation:ListHookResults"
            ],
            "Resource": "*"
        }
    ]
}
```

Control which change sets can be specified

The following example IAM policy grants permissions to the `cloudformation:ListHookResults` action to request results by specifying the target of the Hook. However, it also denies the action if the target is a change set named `example-changeset`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudformation:ListHookResults"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Deny",
            "Action": [
                "cloudformation:ListHookResults"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "cloudformation:ChangeSetName": "example-changeset"
                }
            }
        }
    ]
}
```

Control which Hooks can be specified

The following example IAM policy grants permissions to the `cloudformation:ListAllHookResults` action to request invocation results only when the Hook's ARN is provided in the request. It denies the action for a specified Hook ARN.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "cloudformation:ListAllHookResults"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Deny",
            "Action": [
                "cloudformation:ListAllHookResults"
            ],
            "Resource": "*",
            "Condition": {
                "Null": {
                    "cloudformation:TypeArn": "true"
                }
            }
        },
        {
            "Effect": "Deny",
            "Action": [
                "cloudformation:ListAllHookResults"
            ],
            "Resource": "*",
            "Condition": {
                "ArnEquals": {
                    "cloudformation:TypeArn": "arn:aws-cn:cloudformation:us-
east-1:123456789012:type/hook/MyCompany-MyHook"
                }
            }
        }
    ]
}
```

Allow a user to view detailed Hook invocation results

To grant permissions to view the detailed results of a specific Hook invocation, you must grant access to the `cloudformation:GetHookResult` action. This permission allows users to retrieve detailed information and remediation guidance for a specific Hook invocation result. For more information, see GetHookResult in the *Amazon CloudFormation API Reference*.

The following example IAM policy grants permissions to the `cloudformation:GetHookResult` action.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
       "Action": [
         "cloudformation:GetHookResult"
      ],
      "Resource": "*"
    }
  ]
}
```

Note

You can configure Hooks to encrypt detailed invocation results stored in the cloud with your own Amazon KMS keys. For information about how to set up the key policy and IAM permissions that you need when you use a customer managed key for encryption, see Amazon KMS key policy and permissions for encrypting Amazon CloudFormation Hooks results at rest.

Amazon KMS key policy and permissions for encrypting Amazon CloudFormation Hooks results at rest

This topic describes how to set up the Amazon KMS key policy and permissions that you need when you specify a customer managed key for encrypting Hooks annotations data that's available from the GetHookResult API.

> **Note**
>
> Amazon CloudFormation Hooks doesn't need additional authorization to use the default Amazon owned key to encrypt annotations data in your account.

Topics

- Overview
- Using encryption context to control access to your customer managed key
- Customer managed KMS key policy
- KMS permissions for SetTypeConfiguration API
- KMS permissions for GetHookResult API

Overview

The following Amazon KMS keys can be used for encrypting Hook annotations data:

- Amazon owned key – By default, CloudFormation uses an Amazon owned key to encrypt data. You can't view, manage, or use Amazon owned keys, or audit their use. However, you don't have to perform explicit configuration to protect the key that's used to encrypt your data. Amazon owned keys are provided free of charge (no monthly fees or usage fees). Unless you are required to audit or control the encryption key that protects your annotations data, an Amazon owned key is a good choice.
- Customer managed key – CloudFormation supports the use of a symmetric customer managed key that you create, own, and manage to add a second layer of encryption over the existing Amazon owned key. Amazon KMS charges apply. For more information, see Creating keys in the *Amazon Key Management Service Developer Guide*. To manage your key, use the Amazon Key Management Service (Amazon KMS) in the Amazon KMS console, the Amazon CLI, or the Amazon KMS API. For more information, see the Amazon Key Management Service Developer Guide.

You can configure customer managed keys when creating and updating Hooks. When you provide your customer managed key, CloudFormation uses this key to encrypt the annotations data before storing it. When the annotations data is later accessed during the `GetHookResult` API operation,

CloudFormation automatically decrypts it. For information about configuring your encryption key for Hooks, see Hook configuration schema syntax reference.

> **Important**
>
> Note that the `KmsKeyId` option to specify a customer managed key is currently only available when you use the Amazon CLI to configure your Hook.

Using encryption context to control access to your customer managed key

CloudFormation Hooks automatically includes encryption context with every annotation storage and retrieval operation. This allows you to set encryption context conditions in your key policy to ensure the key can only be used for specific Hooks:

- `kms:EncryptionContext:aws:cloudformation:hooks:service` – Ensures the key is only used by the CloudFormation Hooks service.
- `kms:EncryptionContext:aws:cloudformation:account-id` – Prevents cross-account key usage by matching your Amazon Web Services account ID.
- `kms:EncryptionContext:aws:cloudformation:arn` – Restrict usage to specific Hooks using ARN patterns.

These conditions provide additional protection against confused deputy attacks by cryptographically binding the encrypted data to the specific Hook context.

Customer managed KMS key policy

When creating your customer managed key, you must define its key policy to allow the CloudFormation Hooks service to perform Amazon KMS operations. To use the following key policy, replace the *`placeholder values`* with your own information.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
```

```
    {
      "Sid": "EnableIAMUserDescribeKey",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws-cn:iam::123456789012:role/ExampleRole"
      },
      "Action": "kms:DescribeKey",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "kms:ViaService": "cloudformation.us-east-1.amazonaws.com"
        }
      }
    },
    {
      "Sid": "EnableIAMUserGenerateDataKey",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws-cn:iam::123456789012:role/ExampleRole"
      },
      "Action": "kms:GenerateDataKey",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "kms:ViaService": "cloudformation.us-east-1.amazonaws.com",
          "kms:EncryptionContext:aws:cloudformation:hooks:service":
 "hooks.cloudformation.amazonaws.com",
          "kms:EncryptionContext:aws:cloudformation:account-id": "123456789012"
        },
        "ArnLike": {
          "kms:EncryptionContext:aws:cloudformation:arn": "arn:aws-
cn:cloudformation:*:123456789012:hook/*"
        }
      }
    },
    {
      "Sid": "EnableIAMUserDecrypt",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws-cn:iam::123456789012:role/ExampleRole"
      },
      "Action": "kms:Decrypt",
      "Resource": "*",
      "Condition": {
```

```
        "StringEquals": {
          "kms:ViaService": "cloudformation.us-east-1.amazonaws.com"
        }
      }
    },
    {
      "Sid": "AllowHooksServiceDescribeKey",
      "Effect": "Allow",
      "Principal": {
        "Service": "hooks.cloudformation.amazonaws.com"
      },
      "Action": "kms:DescribeKey",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws-cn:cloudformation:*:123456789012:hook/*"
        }
      }
    },
    {
      "Sid": "AllowHooksService",
      "Effect": "Allow",
      "Principal": {
        "Service": "hooks.cloudformation.amazonaws.com"
      },
      "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012",
          "kms:EncryptionContext:aws:cloudformation:hooks:service":
 "hooks.cloudformation.amazonaws.com",
          "kms:EncryptionContext:aws:cloudformation:account-id": "123456789012"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws-cn:cloudformation:*:123456789012:hook/*",
          "kms:EncryptionContext:aws:cloudformation:arn": "arn:aws-
cn:cloudformation:*:123456789012:hook/*"
```

```
        }
      }
    }
  ]
}
```

This policy grants permissions to both IAM roles (first three statements) and the CloudFormation Hooks service (last two statements). The `kms:ViaService` condition key ensures the KMS key can only be used through CloudFormation, preventing direct KMS API calls. The key operations are:

- `kms:DescribeKey` – Validates key properties and metadata. This operation is in separate statements because it cannot be used with encryption context conditions.
- `kms:GenerateDataKey` – Generates data encryption keys for encrypting annotations before storage. This operation includes encryption context conditions for scoped access control.
- `kms:Decrypt` – Decrypts previously encrypted annotations data. For IAM roles, this includes the `kms:ViaService` condition. For the service principal, this includes encryption context conditions.

The `aws:SourceAccount` and `aws:SourceArn` condition keys provide the primary protection against confused deputy attacks. The encryption context conditions provide additional validation layers. For more information, see Using aws:SourceArn or aws:SourceAccount condition keys in the *Amazon Key Management Service Developer Guide*.

> **Important**
>
> Hook execution roles do not need Amazon KMS permissions. The CloudFormation Hooks service principal performs all Amazon KMS operations.

KMS permissions for `SetTypeConfiguration` API

During the SetTypeConfiguration API call, CloudFormation validates user permissions to encrypt annotations data with the specified Amazon KMS key. Add the following IAM policy to the user or role that will configure encryption using the `SetTypeConfiguration` API. Replace the *`placeholder values`* with your own information.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "cloudformation:SetTypeConfiguration",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "kms:DescribeKey",
      "Resource": "arn:aws-cn:kms:us-east-1:123456789012:key/abc-123"
    },
    {
      "Effect": "Allow",
      "Action": "kms:GenerateDataKey",
      "Resource": "arn:aws-cn:kms:us-east-1:123456789012:key/abc-123",
      "Condition": {
        "StringEquals": {
          "kms:EncryptionContext:aws:cloudformation:hooks:service":
 "hooks.cloudformation.amazonaws.com",
          "kms:EncryptionContext:aws:cloudformation:account-id": "123456789012"
        },
        "ArnLike": {
          "kms:EncryptionContext:aws:cloudformation:arn": "arn:aws-
cn:cloudformation:*:123456789012:hook/*"
        }
      }
    }
  ]
}
```

KMS permissions for `GetHookResult` API

To call GetHookResult for Hooks that use your customer managed key, users must have `kms:Decrypt` permission for that key. Add the following IAM policy to the user or role that will call `GetHookResult`. Replace *arn:aws-cn:kms:us-east-1:123456789012:key/abc-123* with the ARN of your customer managed key.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "cloudformation:GetHookResult",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "kms:Decrypt",
      "Resource": "arn:aws-cn:kms:us-east-1:123456789012:key/abc-123"
    }
  ]
}
```

Document history for the Amazon CloudFormation Hooks user guide

The following table describes the important changes to the documentation since the last release of Amazon CloudFormation Hooks. For notification about updates to this documentation, you can subscribe to an RSS feed.

- **Latest documentation update:** September 4, 2025.

Change	Description	Date
Detailed compliance check results	Hooks now support *annotations* that provide detailed compliance check results and remediation guidance for each evaluated resource. View these detailed validation results through the CloudFormation console or `get-hook-result` CLI command.	November 13, 2025
Proactive controls as Hooks	You can now activate proactive control-based Hooks through the CloudFormation console or CLI using the `activate-type` and `set-type-configuration` commands. You can configure these Hooks to apply specific Amazon Control Tower Control Catalog proactive controls to evaluate resources	September 4, 2025

	during CREATE and UPDATE operations.	
Hooks invocation summary	You can now retrieve information about Hook invocations through the CloudFormation console, or use the `list-hook-results` CLI command to programmatically retrieve invocation details. You can also now filter `list-hook-results` results by Hook or invocation status to focus on relevant invocations.	September 4, 2025
Stack-level Hooks	Hooks are now supported at the stack-level, allowing customers to use CloudFormation Hooks to evaluate new templates and potentially block stack operations from proceeding.	November 13, 2024

Amazon Cloud Control API Hooks integration	Hooks are now integrated with Cloud Control API, allowing customers to use CloudFormation Hooks to proactively inspect the configuration of resources before provisioning. If non-compliant resources are found, the Hook either fails the operation and prevents the resources from being provisioned, or emits a warning and allows the provisioning operation to continue.	November 13, 2024
Amazon CloudFormation Guard Hooks	Amazon CloudFormation Guard is an open-source and general purpose domain specific language (DSL) you can use to author policy-as-code. Guard Hooks can evaluate Cloud Control API and CloudFormation operations to inspect the configuration of resources before provisioning. If non-compliant resources are found, the Hook either fails the operation and prevents the resources from being provisioned, or emits a warning and allows the provisioning operation to continue.	November 13, 2024

Amazon Lambda Hooks	Amazon CloudFormation Lambda Hooks allow you to evaluate CloudFormation and Cloud Control API operation s against custom your own custom code. Your Hook can block an operation from proceeding, or issue a warning to the caller and allow the operation to proceed.	November 13, 2024
Hooks User Guide	Initial version of the Amazon CloudFormation Hooks User Guide. Updates include a new introduction, getting started walkthrough, concepts and terminology, stack level filtering, and updated topics on prerequisites, setup, and CloudFormation Hooks development.	December 8, 2023