



Guida per l'utente di Hooks

CloudFormation



CloudFormation: Guida per l'utente di Hooks

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

I marchi e l'immagine commerciale di Amazon non possono essere utilizzati in relazione a prodotti o servizi che non siano di Amazon, in una qualsiasi modalità che possa causare confusione tra i clienti o in una qualsiasi modalità che denigri o discreditì Amazon. Tutti gli altri marchi non di proprietà di Amazon sono di proprietà dei rispettivi proprietari, che possono o meno essere affiliati, collegati o sponsorizzati da Amazon.

Table of Contents

Cosa sono i CloudFormation ganci?	1
Opzioni di implementazione di Hook	1
AWS Control Tower controlli proattivi	1
Regole di guardia	1
Funzioni Lambda	2
Ganci personalizzati	2
Creazione e gestione di Hooks	3
Concetti	5
Gancio	6
Modalità di errore	6
Obiettivi Hook	6
Azioni mirate	7
Annotazioni	7
Gestore Hook	7
Limiti di timeout e nuovi tentativi	8
Controlli proattivi come Hook	8
AWS CLI comandi per lavorare con gli Hooks	9
Attiva un Hook basato su un controllo proattivo	9
Elimina gli Hooks basati sul controllo proattivo	13
Guard Hooks	14
AWS CLI comandi per lavorare con Guard Hooks	15
Scrivi le regole di Guard per Hooks	15
Preparati a creare un Guard Hook	29
Attiva un Guard Hook	31
Visualizza i log di Guard Hooks	37
Elimina Guard Hooks	37
Lambda Hooks	38
AWS CLI comandi per lavorare con Lambda Hooks	39
Crea funzioni Lambda per Hooks	39
Preparati a creare un Lambda Hook	63
Attiva un Lambda Hook	65
Visualizza i log per Lambda Hooks	69
Elimina Lambda Hooks	70
Ganci personalizzati	71

Prerequisiti	72
Avvio di un progetto Hooks	74
Ganci per modellazione	77
Registrazione di Hooks	144
Ganci di prova	149
Aggiornamento degli hook	158
Annullamento della registrazione degli Hooks	159
Ganci editoriali	160
Sintassi dello schema	168
Disabilita/abilita gli Hooks	177
Disabilita e abilita un Hook (console)	177
Disabilita e abilita un Hook (AWS CLI)	178
Visualizza i risultati della chiamata di Hook	179
Visualizza i risultati della chiamata (console)	179
Visualizza i risultati per tutti gli Hooks	179
Visualizza la cronologia delle chiamate per i singoli Hook	180
Visualizza i risultati per le chiamate specifiche dello stack	180
Visualizza i risultati della chiamata ()AWS CLI	181
Schema di configurazione	186
Proprietà dello schema di configurazione Hook	186
Esempi di configurazione Hook	188
Filtri a livello di pila	188
FilteringCriteria	190
StackNames	190
StackRoles	191
Include e Exclude	192
Esempi di filtri a livello di stack	192
Filtri target	196
Esempi di filtri target	198
Utilizzo di caratteri jolly	200
Crea Hooks utilizzando modelli CloudFormation	209
Concedi le autorizzazioni IAM	211
Consenti a un utente di gestire gli Hooks	212
Consenti a un utente di pubblicare pubblicamente Hooks personalizzati	213
Consenti a un utente di visualizzare i risultati della chiamata di Hook	214
Elenca i risultati della chiamata di Hook	214

Consenti a un utente di visualizzare i risultati dettagliati dell'invocazione di Hook	217
AWS KMS policy e autorizzazioni chiave	218
Panoramica	218
Contesto di crittografia	219
Policy delle chiavi KMS gestite dal cliente	220
Autorizzazioni KMS per l'API SetTypeConfiguration	223
Autorizzazioni KMS per l'API GetHookResult	224
Cronologia dei documenti	225
	ccxxix

Cosa sono i CloudFormation ganci?

CloudFormation Hooks è una funzionalità che aiuta a garantire che le CloudFormation risorse, gli stack e i set di modifiche siano conformi alle best practice aziendali in materia di sicurezza, operatività e ottimizzazione dei costi. CloudFormation Hooks può anche garantire lo stesso livello di conformità per le tue risorse. AWS Cloud Control API Con CloudFormation Hooks, puoi fornire codice che ispeziona in modo proattivo la configurazione delle tue AWS risorse prima del provisioning. Se vengono rilevate risorse non conformi, l'operazione fallisce e impedisce il provisioning delle risorse CloudFormation oppure emette un avviso e consente il proseguimento dell'operazione di provisioning.

È possibile utilizzare Hooks per applicare una serie di requisiti e linee guida. [Ad esempio, un Hook relativo alla sicurezza può verificare che i gruppi di sicurezza abbiano regole di traffico in entrata e in uscita appropriate per il tuo Amazon VPC](#). Un Hook legato ai costi può limitare gli ambienti di sviluppo all'utilizzo solo di tipi di EC2 istanze [Amazon](#) più piccoli. Un Hook progettato per la disponibilità dei dati può imporre backup automatici per [Amazon](#) RDS.

Opzioni di implementazione di Hook

CloudFormation offre diverse opzioni per l'implementazione di Hooks, offrendoti la flessibilità di scegliere l'approccio più adatto alle tue esigenze.

AWS Control Tower controlli proattivi

Il AWS Control Tower Control Catalog offre controlli proattivi standardizzati che puoi implementare come Hooks. Questo approccio consente di risparmiare tempo di configurazione e consente di convalidare le configurazioni delle risorse in base alle AWS migliori pratiche dell'organizzazione senza scrivere codice.

Regole di guardia

AWS CloudFormation Guard è uno strumento di policy-as-code valutazione che fornisce un linguaggio specifico del dominio per scrivere logiche di valutazione personalizzate per Hooks. Questo approccio consente di definire i controlli di conformità utilizzando la sintassi dichiarativa di Guard, semplificando la creazione e la manutenzione della logica di valutazione senza una conoscenza approfondita della programmazione.

Funzioni Lambda

Puoi anche implementare Hooks utilizzando le funzioni Lambda, consentendoti di sfruttare tutta la potenza e la flessibilità di Lambda per la tua logica di valutazione. È possibile utilizzare qualsiasi linguaggio di runtime supportato da Lambda e integrarlo con altri AWS servizi in base alle esigenze.

Ganci personalizzati

Per casi d'uso avanzati, è possibile scrivere la propria logica di valutazione utilizzando linguaggi di programmazione supportati dalla [CloudFormation CLI](#). Questo approccio offre la massima flessibilità per l'implementazione dei requisiti di governance specifici dell'organizzazione. Essendo un tipo di estensione supportato nel [CloudFormation registro](#), gli Hooks personalizzati possono essere distribuiti e attivati sia pubblicamente che privatamente.

Creazione e gestione di CloudFormation Hooks

CloudFormation Gli hook forniscono un meccanismo per valutare le CloudFormation risorse prima di consentire la creazione, la modifica o l'eliminazione degli stack. Questa funzionalità consente di garantire che le CloudFormation risorse siano conformi alle best practice aziendali in materia di sicurezza, operatività e ottimizzazione dei costi.

Per creare un Hook, hai quattro opzioni.

- Controlli proattivi come Hooks: valuta le risorse utilizzando i controlli proattivi del Control Catalog. AWS Control Tower
- Guard Hook: valuta le risorse utilizzando una regola. AWS CloudFormation Guard
- Lambda Hook: inoltra le richieste di valutazione delle risorse a una funzione. AWS Lambda
- Hook personalizzato: utilizza un gestore Hook personalizzato sviluppato manualmente.

Proactive controls as Hooks

Per creare un Hook da controlli proattivi, segui questi passaggi:

1. Vai alla CloudFormation console e inizia a creare un Hook.
2. Scegli controlli specifici dal Control Catalog in base ai quali desideri che Hook valuti le risorse.

Questi controlli verranno applicati automaticamente ogni volta che vengono create o aggiornate risorse specificate. La selezione determina i tipi di risorse che Hook valuterà.

3. Imposta la modalità Hook per avvisare gli utenti della non conformità o impedire operazioni non conformi.
4. Configura filtri opzionali per includere o escludere gli stack in base al nome dello stack o al ruolo dello stack.
5. Dopo aver completato la configurazione, attiva l'Hook per iniziare l'applicazione.

Guard Hook

Per creare un Guard Hook, segui questi passaggi:

1. Scrivi la tua logica di valutazione delle risorse come regola della politica di Guard utilizzando il linguaggio DSL (Domain-Specific Language) di Guard.
2. Archivia la regola della policy Guard in un bucket Amazon S3.
3. Vai alla CloudFormation console e inizia a creare un Guard Hook.
4. Fornisci il percorso Amazon S3 alla tua regola Guard.
5. Scegli i tipi di target specifici che Hook valuterà.
 - CloudFormation risorse (RESOURCE)
 - Modelli di interi stack () STACK
 - Set di modifiche () CHANGE_SET
 - Risorse API Cloud Control (CLOUD_CONTROL)
6. Scegli le azioni di implementazione (creazione, aggiornamento, eliminazione) che richiameranno il tuo Hook.
7. Scegli come risponde l'Hook quando la valutazione fallisce.
8. Configura filtri opzionali per specificare quali tipi di risorse l'Hook deve valutare
9. Configura filtri opzionali per includere o escludere gli stack in base al nome dello stack o al ruolo dello stack.
10. Dopo aver completato la configurazione, attiva l'Hook per iniziare l'applicazione.

Lambda Hook

Per creare un Lambda Hook, segui questi passaggi:

1. Scrivi la tua logica di valutazione delle risorse come funzione Lambda.
2. Vai alla CloudFormation console e inizia a creare un Lambda Hook.
3. Fornisci l'Amazon Resource Name (ARN) per la tua funzione Lambda.
4. Scegli i tipi di target specifici che Hook valuterà.
 - CloudFormation risorse (RESOURCE)
 - Modelli di interi stack () STACK
 - Set di modifiche () CHANGE_SET
 - Risorse API Cloud Control (CLOUD_CONTROL)
5. Scegli le azioni di implementazione (creazione, aggiornamento, eliminazione) che richiameranno il tuo Hook.

6. Scegli come risponde l'Hook quando la valutazione fallisce.
7. Configura filtri opzionali per specificare quali tipi di risorse l'Hook deve valutare
8. Configura filtri opzionali per includere o escludere gli stack in base al nome dello stack o al ruolo dello stack.
9. Dopo aver completato la configurazione, attiva l'Hook per iniziare l'applicazione.

Custom Hook

Gli hook personalizzati sono estensioni che si registrano nel CloudFormation registro utilizzando l'interfaccia a riga di CloudFormation comando (CFN-CLI).

Per creare un Hook personalizzato, segui questi passaggi principali:

1. Avvia il progetto: genera i file necessari per sviluppare un Hook personalizzato.
2. Modella l'Hook: scrivi uno schema che definisca l'Hook e i gestori che specifichino le operazioni che possono richiamare l'Hook.
3. Registra e attiva l'Hook: dopo aver creato un Hook, devi registrarlo nell'account e nella regione in cui desideri utilizzarlo e questo lo attiva.

I seguenti argomenti forniscono ulteriori informazioni sulla creazione e la gestione degli Hook.

Argomenti

- [CloudFormation Concetti di ganci](#)
- [AWS Control Tower controlli proattivi come Hooks](#)
- [Guard Hooks](#)
- [Lambda Hooks](#)
- [Sviluppo di hook personalizzati utilizzando la CLI CloudFormation](#)

CloudFormation Concetti di ganci

La terminologia e i concetti seguenti sono fondamentali per la comprensione e l'uso di Hooks.

CloudFormation

Gancio

Un Hook contiene codice che viene richiamato immediatamente prima di CloudFormation creare, aggiornare o eliminare pile o risorse specifiche. Può anche essere richiamato durante un'operazione di creazione di un set di modifiche. Gli hook possono ispezionare il modello, le risorse o il set di modifiche che CloudFormation si sta per fornire. Inoltre, gli Hooks possono essere richiamati immediatamente prima che l'[API Cloud Control](#) crei, aggiorni o elimini risorse specifiche.

Se un Hook identifica configurazioni che non sono conformi alle linee guida organizzative definite nella logica di Hook, puoi scegliere se utilizzare WARN gli utenti o impedire CloudFormation il provisioning della FAIL risorsa.

Gli hook hanno le seguenti caratteristiche:

- Convalida proattiva: riduce i rischi, il sovraccarico operativo e i costi identificando le risorse non conformi prima che vengano create, aggiornate o eliminate.
- Applicazione automatica: garantisce l'applicazione delle norme Account AWS per impedire che vengano fornite risorse non conformi da parte di CloudFormation

Modalità di errore

La tua logica Hook può restituire un successo o un fallimento. Una risposta di successo consentirà il proseguimento dell'operazione. Un errore relativo a risorse non conformi può causare quanto segue:

- FAIL— Interrompe l'operazione di provisioning.
- WARN— Consente di continuare il provisioning con un messaggio di avviso.

La creazione di Hook in WARN modalità è un modo efficace per monitorare il comportamento degli Hook senza influire sulle operazioni di stack. Innanzitutto, attiva Hooks in WARN modalità per capire quali operazioni saranno influenzate. Dopo aver valutato i potenziali effetti, puoi passare alla FAIL modalità Hook per iniziare a prevenire le operazioni non conformi.

Obiettivi Hook

Gli Hook target specificano le operazioni che un Hook valuterà. Queste possono essere operazioni su:

- Risorse supportate da CloudFormation (RESOURCE)

- Modelli Stack () STACK
- Set di modifiche () CHANGE_SET
- Risorse supportate dall'[API Cloud Control](#) (CLOUD_CONTROL)

Definisci uno o più obiettivi che specificano le operazioni più ampie che Hook valuterà. Ad esempio, potete creare un Hook mirato a tutte le AWS risorse e RESOURCE STACK a tutti i modelli di stack.

Azioni mirate

Le azioni mirate definiscono le azioni specifiche (CREATE,UPDATE, oDELETE) che richiameranno un Hook. Per RESOURCE, e CLOUD_CONTROL targetSTACK, sono applicabili tutte le azioni mirate. Per CHANGE_SET gli obiettivi, è applicabile solo l'CREATEazione.

Annotazioni

[GetHookResult](#) le risposte possono restituire annotazioni che forniscono risultati dettagliati del controllo di conformità e indicazioni sulla correzione per ciascuna risorsa valutata. Per i dettagli sulla struttura delle annotazioni dell'API, consulta [Annotation nel riferimento API](#). AWS CloudFormation Per istruzioni sulla visualizzazione di questi risultati di convalida, consulta. [Visualizza i risultati di chiamata per Hooks CloudFormation](#)

È possibile crittografare le annotazioni necessarie per le informazioni sensibili sulla conformità specificando la propria chiave KMS durante la configurazione dell'Hook. Per ulteriori informazioni, consulta [Riferimento alla sintassi dello schema di configurazione Hook](#). Per informazioni sulla configurazione della politica delle chiavi di cui hai bisogno quando specifichi la tua chiave KMS per Hooks, vedi. [AWS KMS policy chiave e autorizzazioni per crittografare i risultati di CloudFormation Hooks a riposo](#)

Important

Tieni presente che l'KmsKeyId opzione per specificare una chiave gestita dal cliente è attualmente disponibile solo quando utilizzi AWS CLI per configurare il tuo Hook.

Gestore Hook

Per gli Hooks personalizzati, questo è il codice che gestisce la valutazione. È associato a un punto di invocazione di destinazione e a un'azione di destinazione che contrassegnano un punto esatto

in cui viene eseguito un Hook. Si scrivono gestori che ospitano la logica per questi punti specifici. Ad esempio, un punto di invocazione di PRE destinazione con azione CREATE target crea un gestore Hook. preCreate Il codice all'interno del gestore Hook viene eseguito quando un punto di invocazione e un servizio di destinazione corrispondenti eseguono un'azione di destinazione associata.

Valori validi: (| |) preCreate preUpdate preDelete

Important

Le operazioni di stack che determinano lo stato di UpdateCleanup non richiamano un Hook. Ad esempio, durante i due scenari seguenti, il preDelete gestore di Hook non viene richiamato:

- lo stack viene aggiornato dopo aver rimosso una risorsa dal modello.
- viene eliminata una risorsa con il tipo di aggiornamento [sostitutivo](#).

Limiti di timeout e nuovi tentativi

Gli hook hanno un limite di timeout di 30 secondi per chiamata e sono limitati a 3 tentativi di nuovo tentativo. Se una chiamata supera il timeout, restituiamo un messaggio di errore che indica che l'esecuzione dell'Hook è scaduta. Dopo il terzo tentativo, contrassegna l'esecuzione dell'Hook come fallita CloudFormation .

AWS Control Tower controlli proattivi come Hooks

Il AWS Control Tower Control Catalog fornisce regole di conformità predefinite (controlli proattivi) che puoi implementare come Hooks. Questo approccio consente di risparmiare tempo di configurazione e consente di convalidare le configurazioni delle risorse in base alle AWS migliori pratiche dell'organizzazione senza scrivere codice.

I controlli proattivi valutano AWS le risorse prima dell'implementazione, impedendo la creazione di risorse non conformi anziché rilevare i problemi in un secondo momento. Controllano le configurazioni rispetto agli standard di sicurezza, operativi e di governance stabiliti.

Per iniziare, è sufficiente attivare gli Hooks basati sul controllo proattivo nell'account e nella regione desiderati. Questi Hooks valuteranno quindi tipi di bersaglio specifici per garantire la conformità con i controlli selezionati.

Per ulteriori informazioni sui controlli proattivi disponibili, consulta il [AWS Control Tower Control Catalog](#).

Argomenti

- [AWS CLI comandi per lavorare con gli Hooks](#)
- [Attiva un Hook basato sul controllo proattivo nel tuo account](#)
- [Elimina gli Hooks basati sul controllo proattivo dal tuo account](#)

AWS CLI comandi per lavorare con gli Hooks

I AWS CLI comandi per lavorare con gli Hooks basati sul controllo proattivo includono:

- [activate-type](#)per avviare il processo di attivazione di un Hook basato su un controllo proattivo.
- [set-type-configuration](#)per specificare i controlli da applicare a un Hook basato su controlli proattivi nel tuo account.
- [list-types](#)per elencare gli Hooks presenti nel tuo account.
- [describe-type](#)per restituire informazioni dettagliate su uno specifico Hook o una versione specifica di Hook, inclusi i dati di configurazione correnti.
- [deactivate-type](#)per rimuovere un Hook precedentemente attivato dal tuo account.

Attiva un Hook basato sul controllo proattivo nel tuo account

L'argomento seguente mostra come attivare un Hook basato sul controllo proattivo nel tuo account, in modo da renderlo utilizzabile nell'account e nella regione in cui è stato attivato.

Important

Prima di continuare, verifica di disporre delle autorizzazioni necessarie per lavorare con Hooks e visualizzare i controlli proattivi dalla console. CloudFormation Per ulteriori informazioni, consulta [Concedi le autorizzazioni IAM per CloudFormation Hooks](#).

Argomenti

- [Attiva un Hook \(console\) basato sul controllo proattivo](#)
- [Attiva un Hook basato su un controllo proattivo \(\)AWS CLI](#)

Attiva un Hook (console) basato sul controllo proattivo

Per attivare un Hook basato su un controllo proattivo da utilizzare nel tuo account

1. [Accedi Console di gestione AWS e apri la CloudFormation console all'indirizzo /cloudformation.
https://console.aws.amazon.com](#)
2. Nella barra di navigazione nella parte superiore dello schermo, scegli Regione AWS dove vuoi creare l'Hook in.
3. Nel riquadro di navigazione a sinistra, scegli Hooks.
4. Nella pagina Hooks, scegli Crea un hook, quindi scegli With the Control Catalog.
5. Nella pagina Seleziona controlli, per i controlli proattivi, seleziona uno o più controlli proattivi da utilizzare.

Questi controlli verranno applicati automaticamente ogni volta che vengono create o aggiornate risorse specifiche. La selezione determina i tipi di risorse che Hook valuterà.

6. Scegli Next (Successivo).
7. Per Hook name, scegliete una delle seguenti opzioni:
 - Fornisci un nome breve e descrittivo che verrà aggiunto dopo `Private::Controls::`. Ad esempio, se inserisci **MyTestHook**, il nome completo di Hook diventa `Private::Controls::MyTestHook`.
 - Fornisci il nome completo dell'Hook (chiamato anche alias) utilizzando questo formato: **Provider::ServiceName::HookName**.
8. Per la modalità Hook, scegli come risponde l'Hook quando i controlli non superano la valutazione:
 - Avvisa: invia avvisi agli utenti ma consente il proseguimento delle azioni. Ciò è utile per convalide non critiche o controlli informativi.
 - Fallito: impedisce il proseguimento dell'azione. Ciò è utile per applicare rigorose politiche di conformità o sicurezza.
9. Scegli Next (Successivo).
10. (Facoltativo) Per i filtri Hook, procedi come segue:
 - a. Per i criteri di filtraggio, scegliete la logica per applicare i filtri del nome dello stack e del ruolo dello stack:

- Tutti i nomi degli stack e i ruoli dello stack: l'Hook verrà richiamato solo quando tutti i filtri specificati corrispondono.
- Qualsiasi nome dello stack e ruolo dello stack: l'Hook verrà richiamato se almeno uno dei filtri specificati corrisponde.

- b. Per i nomi degli stack, includi o escludi pile specifiche dalle invocazioni di Hook.
- Per Include, specificate i nomi degli stack da includere. Usalo quando hai un piccolo set di pile specifiche a cui vuoi rivolgerti. Solo gli stack specificati in questo elenco richiameranno l'Hook.
 - Per Exclude, specifica i nomi degli stack da escludere. Usalo quando vuoi invocare l'Hook sulla maggior parte degli stack ma escluderne alcuni specifici. Tutti gli stack tranne quelli elencati qui invocheranno l'Hook.
- c. Per i ruoli Stack, includi o escludi stack specifici dalle invocazioni di Hook in base ai ruoli IAM associati.
- Per Include, specifica uno o più ruoli IAM ARNs per indirizzare gli stack associati a questi ruoli. Solo le operazioni di stack avviate da questi ruoli richiameranno l'Hook.
 - Per Exclude, specifica uno o più ruoli IAM ARNs per gli stack che desideri escludere. L'Hook verrà richiamato su tutti gli stack ad eccezione di quelli avviati dai ruoli specificati.

11. Scegli Next (Successivo).
12. Nella pagina Rivedi e attiva, rivedi le tue scelte. Per apportare modifiche, scegli Modifica nella sezione correlata.
13. Quando sei pronto per procedere, scegli Activate Hook.

Attiva un Hook basato su un controllo proattivo ()AWS CLI

Prima di continuare, conferma di aver identificato i controlli proattivi che utilizzerai con questo Hook. Per ulteriori informazioni, consulta il [AWS Control Tower Control Catalog](#).

Per attivare un Hook basato sul controllo proattivo da utilizzare nel tuo account ()AWS CLI

1. Per iniziare ad attivare un Hook, usa il seguente [activate-type](#) comando, sostituendo i segnaposto con i tuoi valori specifici.

```
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. Per completare l'attivazione dell'Hook, devi configurarlo utilizzando un file di configurazione JSON.

Usa il cat comando per creare un file JSON con la seguente struttura. Per ulteriori informazioni, consulta [Riferimento alla sintassi dello schema di configurazione Hook](#).

L'esempio seguente configura un Hook che richiama risorse IAM, Amazon e Amazon EC2 S3 specifiche durante le operazioni. CREATE UPDATE Applica tre controlli proattivi (CT.IAM.PR.5,CT.EC2.PR.17,CT.S3.PR.12) per convalidare queste risorse rispetto agli standard di conformità. L'hook funziona in WARN modalità, il che significa che contrassegnerà le risorse non conformi con avvisi ma non bloccherà le distribuzioni.

```
$ 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**: Impostato per abilitare l'Hook. ENABLED
- **TargetOperations**: Impostato su RESOURCE poiché questo è l'unico valore supportato per un Hook basato sul controllo proattivo.
- **FailureMode**: impostare su FAIL o su WARN.

- **ControlsToApply**: Specificate il controllo IDs dei controlli proattivi da utilizzare. Per ulteriori informazioni, consulta il [AWS Control Tower Control Catalog](#).
 - (Facoltativo) **TargetFilters**: Per Actions, puoi specificare CREATE o UPDATE, o entrambi (impostazione predefinita), per controllare quando viene richiamato l'Hook. La CREATE sola specificazione limita l'Hook alle sole CREATE operazioni. **TargetFilters**Le altre proprietà non hanno effetto.
3. Utilizzate il [set-type-configuration](#) comando seguente, insieme al file JSON creato, per applicare la configurazione. Sostituisci i segnaposto con i tuoi valori specifici.

```
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
```

Elimina gli Hooks basati sul controllo proattivo dal tuo account

Quando non hai più bisogno di un Hook attivato e basato sul controllo proattivo, utilizza le seguenti procedure per eliminarlo dal tuo account.

Per disabilitare temporaneamente un Hook anziché eliminarlo, consulta [Disabilita e abilita gli CloudFormation Hooks](#)

Argomenti

- [Elimina un Hook basato sul controllo proattivo dal tuo account \(console\)](#)
- [Elimina un Hook basato sul controllo proattivo nel tuo account \(\)AWS CLI](#)

Elimina un Hook basato sul controllo proattivo dal tuo account (console)

Per eliminare un Hook basato su un controllo proattivo dal tuo account

1. [Accedi Console di gestione AWS e apri la CloudFormation console all'indirizzo /cloudformazione. <https://console.aws.amazon.com>](#)
2. Nella barra di navigazione nella parte superiore dello schermo, scegli Regione AWS dove si trova l'Hook.
3. Dal pannello di navigazione, scegli Hooks.

4. Nella pagina Hooks, trova l'Hook basato sul controllo proattivo che desideri eliminare.
5. Seleziona la casella di controllo accanto al tuo Hook e scegli Elimina.
6. Quando viene richiesta la conferma, digita il nome dell'Hook per confermare l'eliminazione dell'Hook specificato, quindi scegli Elimina.

Elimina un Hook basato sul controllo proattivo nel tuo account ()AWS CLI

Note

Prima di poter eliminare l'Hook, devi prima disabilitarlo. Per ulteriori informazioni, consulta [Disattiva e abilita un Hook nel tuo account \(\)AWS CLI](#).

Usa il seguente `deactivate-type` comando per disattivare un Hook, che lo rimuove dal tuo account. Sostituisci i segnaposto con i tuoi valori specifici.

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

Guard Hooks

Per utilizzare un AWS CloudFormation Guard Hook nel tuo account, devi attivare l'Hook per l'account e la regione in cui desideri utilizzarlo. L'attivazione di un Hook lo rende utilizzabile nelle operazioni di pila nell'account e nella regione in cui è attivato.

Quando attivi un Guard Hook, CloudFormation crea una voce nel registro del tuo account per l'Hook attivato come Hook privato. Ciò consente di impostare tutte le proprietà di configurazione incluse nell'Hook. Le proprietà di configurazione definiscono come l'Hook è configurato per una determinata Account AWS regione.

Argomenti

- [AWS CLI comandi per lavorare con Guard Hooks](#)
- [Scrivi le regole di Guard per valutare le risorse per Guard Hooks](#)
- [Preparati a creare un Guard Hook](#)

- [Attiva un Guard Hook nel tuo account](#)
- [Visualizza i log dei Guard Hooks nel tuo account](#)
- [Elimina Guard Hooks dal tuo account](#)

AWS CLI comandi per lavorare con Guard Hooks

I AWS CLI comandi per lavorare con Guard Hooks includono:

- [activate-type](#)per avviare il processo di attivazione di un Guard Hook.
- [set-type-configuration](#)per specificare i dati di configurazione per un Hook nel tuo account.
- [list-types](#)per elencare gli Hooks presenti nel tuo account.
- [describe-type](#)per restituire informazioni dettagliate su uno specifico Hook o una versione specifica di Hook, inclusi i dati di configurazione correnti.
- [deactivate-type](#)per rimuovere un Hook precedentemente attivato dal tuo account.

Scrivi le regole di Guard per valutare le risorse per Guard Hooks

AWS CloudFormation Guard è un linguaggio DSL (Domain Specific Language) open source e generico che puoi utilizzare per la creazione. policy-as-code Questo argomento spiega come utilizzare Guard per creare regole di esempio che possono essere eseguite in Guard Hook per valutazioni CloudFormation e AWS Cloud Control API operazioni automatiche. Si concentrerà anche sui diversi tipi di input disponibili per le regole di Guard a seconda del momento in cui Guard Hook viene eseguito. Un Guard Hook può essere configurato per essere eseguito durante i seguenti tipi di operazioni:

- Operazioni sulle risorse
- Operazioni di stack
- Operazioni di modifica del set

Per ulteriori informazioni sulla scrittura delle regole di Guard, vedi [Writing AWS CloudFormation Guard rules](#)

Argomenti

- [Gestione delle risorse: regole di Guard](#)
- [Operazione Stack \(regole Guard\)](#)

- [Modifica le regole operative di Guard](#)

Gestione delle risorse: regole di Guard

Ogni volta che crei, aggiorni o elimini una risorsa, questa viene considerata un'operazione relativa alle risorse. Ad esempio, se esegui l'aggiornamento di uno CloudFormation stack che crea una nuova risorsa, hai completato un'operazione sulla risorsa. Quando crei, aggiorni o elimini una risorsa utilizzando l'API Cloud Control, anche questa viene considerata un'operazione relativa alle risorse. Puoi configurare Guard Hook in base alla destinazione RESOURCE e CLOUD_CONTROL alle operazioni nella TargetOperations configurazione del tuo Hook. Quando il Guard Hook valuta il funzionamento di una risorsa, il motore Guard valuta l'input di una risorsa.

Argomenti

- [Sintassi di input delle risorse Guard](#)
- [Esempio: input relativo al funzionamento delle risorse Guard](#)
- [Regole di protezione per le modifiche alle risorse](#)

Sintassi di input delle risorse Guard

L'input delle risorse Guard è costituito dai dati messi a disposizione delle regole di Guard per la valutazione.

Di seguito è riportato un esempio di forma di input di risorse:

```
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**

L'ID del Account AWS contenitore della risorsa da valutare.

StackId

L'ID dello CloudFormation stack che fa parte dell'operazione relativa alla risorsa. È vuoto se il chiamante è Cloud Control API.

HookTypeName

Il nome dell'Hook in esecuzione.

HookTypeVersion

La versione dell'Hook in esecuzione.

InvocationPoint

Il punto esatto della logica di provisioning in cui viene eseguito l'Hook.

Valori validi: (CREATE_PRE_PROVISION|UPDATE_PRE_PROVISION
|DELETE_PRE_PROVISION)

TargetName

Il tipo di oggetto da valutare, ad esempio, AWS::S3::Bucket.

TargetType

Il tipo di oggetto da valutare, ad esempio. AWS::S3::Bucket Per le risorse fornite con l'API Cloud Control, questo valore sarà. RESOURCE

TargetLogicalId

La risorsa oggetto TargetLogicalId di valutazione. Se l'origine dell'Hook è CloudFormation, questo sarà l'ID logico (noto anche come nome logico) della risorsa. Se l'origine dell'Hook è l'API Cloud Control, questo sarà un valore costruito.

ChangeSetId

L'ID del set di modifiche che è stato eseguito per causare la chiamata di Hook. Questo valore è vuoto se la modifica delle risorse è stata avviata dall'API Cloud Control o dalle operazioni `create-stack`, `update-stack`, o `delete-stack`.

Resources

ResourceLogicalID

Quando l'operazione viene avviata da CloudFormation, `ResourceLogicalID` è l'ID logico della risorsa nel CloudFormation modello.

Quando l'operazione viene avviata dall'API Cloud Control, `ResourceLogicalID` è una combinazione del tipo di risorsa, del nome, dell'ID dell'operazione e dell'ID della richiesta.

ResourceType

Il nome del tipo della risorsa (esempio:`AWS::S3::Bucket`).

ResourceProperties

Le proprietà proposte della risorsa da modificare. Quando Guard Hook viene eseguito in caso di modifiche alla CloudFormation risorsa, tutte le funzioni, i parametri e le trasformazioni verranno completamente risolti. Se la risorsa viene eliminata, questo valore sarà vuoto.

Previous

ResourceLogicalID

Quando l'operazione viene avviata da CloudFormation, `ResourceLogicalID` è l'ID logico della risorsa nel CloudFormation modello.

Quando l'operazione viene avviata dall'API Cloud Control, `ResourceLogicalID` è una combinazione del tipo di risorsa, del nome, dell'ID dell'operazione e dell'ID della richiesta.

ResourceType

Il nome del tipo della risorsa (esempio:`AWS::S3::Bucket`).

ResourceProperties

Le proprietà correnti associate alla risorsa da modificare. Se la risorsa viene eliminata, questo valore sarà vuoto.

Esempio: input relativo al funzionamento delle risorse Guard

L'input di esempio seguente mostra un Guard Hook che riceverà la definizione della AWS::S3::Bucket risorsa da aggiornare. Questi sono i dati a disposizione di Guard per la valutazione.

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
```

Per visualizzare tutte le proprietà disponibili per il tipo di risorsa, vedere [AWS::S3::Bucket](#).

Regole di protezione per le modifiche alle risorse

Quando un Guard Hook valuta le modifiche alle risorse, inizia scaricando tutte le regole configurate con l'Hook. Queste regole vengono quindi valutate in base all'input delle risorse. L'Hook fallirà se una regola fallisce nella sua valutazione. Se non ci sono errori, l'Hook passerà.

L'esempio seguente è una regola Guard che valuta se la ObjectLockEnabled proprietà è true per qualsiasi tipo di AWS::S3::Bucket risorsa.

```
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.  
>>  
}
```

Quando questa regola viene eseguita sul seguente input, avrà esito negativo poiché la ObjectLockEnabled proprietà non è impostata su. true

Resources:

MyS3Bucket:

Type: AWS::S3::Bucket

Properties:

BucketName: *amzn-s3-demo-bucket*

ObjectLockEnabled: false

Quando questa regola viene eseguita sul seguente input, passerà poiché ObjectLockEnabled è impostata a true.

Resources:

MyS3Bucket:

Type: AWS::S3::Bucket

Properties:

BucketName: *amzn-s3-demo-bucket*

ObjectLockEnabled: true

Quando un Hook fallisce, le regole che hanno fallito verranno propagate nuovamente alla nostra CloudFormation API Cloud Control. Se è stato configurato un bucket di registrazione per Guard Hook, qui verrà fornito un feedback aggiuntivo sulle regole. Questo feedback aggiuntivo include le informazioni Violation e Fix.

Operazione Stack (regole Guard)

Quando uno CloudFormation stack viene creato, aggiornato o eliminato, puoi configurare Guard Hook in modo che inizi a valutare il nuovo modello e potenzialmente bloccare il proseguimento dell'operazione sullo stack. Puoi configurare il tuo Guard Hook per indirizzare STACK le operazioni nella TargetOperations configurazione del tuo Hook.

Argomenti

- [Sintassi di input Guard Stack](#)
- [Esempio di input dell'operazione Guard stack](#)
- [Regole Guard per le modifiche allo stack](#)

Sintassi di input Guard Stack

L'input per le operazioni dello stack Guard fornisce l'intero CloudFormation modello per la valutazione delle regole di Guard.

Di seguito è riportato un esempio di forma di input dello stack:

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

HookContext

AWSAccountID

L'ID del Account AWS contenitore della risorsa.

StackId

L'ID dello CloudFormation stack che fa parte dell'operazione stack.

HookTypeName

Il nome dell'Hook in esecuzione.

HookTypeVersion

La versione dell'Hook in esecuzione.

InvocationPoint

Il punto esatto della logica di provisioning in cui viene eseguito l'Hook.

Valori validi: (CREATE_PRE_PROVISION| UPDATE_PRE_PROVISION
|DELETE_PRE_PROVISION)

TargetName

Il nome dello stack da valutare.

TargetType

Questo valore sarà presente STACK quando viene eseguito come Hook a livello di stack.

ChangeSetId

L'ID del set di modifiche che è stato eseguito per causare la chiamata di Hook. Questo valore è vuoto se l'operazione stack è stata avviata da un'operazione create-stack, update-stack o delete-stack.

Proposed CloudFormation Template

Il valore completo del CloudFormation modello che è stato passato alle nostre operazioni CloudFormation create-stack, update-stack. Ciò include elementi come Resources, Outputs, e Properties. Può essere una stringa JSON o YAML a seconda di ciò che è stato fornito. CloudFormation

Nelle delete-stack operazioni, questo valore sarà vuoto.

Previous

L'ultimo CloudFormation modello distribuito correttamente. Questo valore è vuoto se lo stack viene creato o eliminato.

Nelle delete-stack operazioni, questo valore sarà vuoto.

Note

I modelli forniti sono quelli che vengono passati create o le operazioni di update impilamento. Quando si elimina uno stack, non viene fornito alcun valore di modello.

Esempio di input dell'operazione Guard stack

L'input di esempio seguente mostra un Guard Hook che riceverà un modello completo e il modello precedentemente distribuito. Il modello in questo esempio utilizza il formato JSON.

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

Regole Guard per le modifiche allo stack

Quando un Guard Hook valuta le modifiche allo stack, inizia scaricando tutte le regole configurate con l'Hook. Queste regole vengono quindi valutate in base all'input delle risorse. L'Hook fallirà se una regola fallisce nella sua valutazione. Se non ci sono errori, l'Hook passerà.

L'esempio seguente è una regola Guard che valuta se esistono tipi di AWS::S3::Bucket risorse contenenti una proprietà chiamata `BucketEncryption`, con l'`SSEAlgorithm` impostazione su `aws:kms`. `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.ServerSideEncryptionConfiguration
  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"
  >>
}
```

Quando la regola viene eseguita sul modello seguente, lo farà `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}'
```

Quando la regola viene eseguita sul modello seguente, lo farà `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

```

Modifica le regole operative di Guard

Quando viene creato un set di CloudFormation modifiche, è possibile configurare Guard Hook per valutare il modello e le modifiche proposte nel set di modifiche per bloccare l'esecuzione del set di modifiche.

Argomenti

- [Guard change set \(sintassi di input\)](#)
- [Esempio: input dell'operazione Guard change set](#)
- [Regola Guard per le operazioni relative ai set di modifiche](#)

Guard change set (sintassi di input)

L'input del set di modifiche di Guard è costituito dai dati messi a disposizione delle regole di Guard per la valutazione.

Di seguito è riportato un esempio di forma di input di un set di modifiche:

```

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

```

```
{CloudFormation Template}  
Changes: [{ResourceChange}]
```

La sintassi ResourceChange del modello è:

```
logicalResourceId: String  
resourceType: String  
operazione: CREATE, UPDATE, DELETE  
Numero di riga: Number  
BeforeContext: JSON String  
AfterContext: JSON String
```

HookContext

AWSAccountID

L'ID del Account AWS contenitore della risorsa.

StackId

L'ID dello CloudFormation stack che fa parte dell'operazione stack.

HookTypeName

Il nome dell'Hook in esecuzione.

HookTypeVersion

La versione dell'Hook in esecuzione.

InvocationPoint

Il punto esatto della logica di provisioning in cui viene eseguito l'Hook.

Valori validi: (CREATE_PRE_PROVISION| UPDATE_PRE_PROVISION
|DELETE_PRE_PROVISION)

TargetName

Il nome dello stack da valutare.

TargetType

Questo valore sarà presente CHANGE_SET quando viene eseguito come Hook a livello di set di modifiche.

TargetLogicalId

Questo valore sarà l'ARN del set di modifiche.

ChangeSetId

L'ID del set di modifiche che è stato eseguito per causare la chiamata di Hook. Questo valore è vuoto se l'operazione stack è stata avviata da un'operazione `create-stack`, `update-stack` o `delete-stack`

Proposed CloudFormation Template

Il CloudFormation modello completo fornito per un'operazione `create-change-set`. Può essere una stringa JSON o YAML a seconda del tipo di stringa fornita. CloudFormation

Previous

L'ultimo modello distribuito con successo. CloudFormation Questo valore è vuoto se lo stack viene creato o eliminato.

Changes

Il Changes modello. Questo elenca le modifiche alle risorse.

Modifiche

logicalResourceId

Il nome della risorsa logica della risorsa modificata.

resourceType

Il tipo di risorsa che verrà modificata.

operazione

Il tipo di operazione eseguita sulla risorsa.

Valori validi: (CREATE| UPDATE |DELETE)

Numero di riga

Il numero di riga nel modello associato alla modifica.

BeforeContext

Una stringa JSON di proprietà della risorsa prima della modifica:

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

AfterContext

Una stringa JSON di proprietà della risorsa dopo la modifica:

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

Esempio: input dell'operazione Guard change set

L'input di esempio seguente mostra un Guard Hook che riceverà un modello completo, il modello precedentemente distribuito e un elenco di modifiche alle risorse. Il modello in questo esempio utilizza il formato JSON.

HookContext:

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

```
    "S3Bucket": {
        "Type": "AWS::S3::Bucket",
        "Properties": {
            "BucketName": "amzn-s3-demo-bucket",
            "VersioningConfiguration": {
                "Status": "Enabled"
            }
        }
    }
}
```

```
Previous: {
    "AWSTemplateFormatVersion": "2010-09-09",
    "Resources": {
        "S3Bucket": {
            "Type": "AWS::S3::Bucket",
            "Properties": {
                "BucketName": "amzn-s3-demo-bucket",
            }
        }
    }
}
```

```

        "VersioningConfiguration": {
            "Status": "Suspended"
        }
    }
}
}

Changes: [
{
    "logicalResourceId": "S3Bucket",
    "resourceType": "AWS::S3::Bucket",
    "action": "UPDATE",
    "lineNumber": 5,
    "beforeContext": "{\"Properties\":{\"VersioningConfiguration\":{\"Status\":\"Suspended\"}}}",
    "afterContext": "{\"Properties\":{\"VersioningConfiguration\":{\"Status\":\"Enabled\"}}}"
}
]

```

Regola Guard per le operazioni relative ai set di modifiche

L'esempio seguente è una regola Guard che valuta le modifiche ai bucket Amazon S3 e garantisce `VersionConfiguration` che non sia disabilitata.

```

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'
    }
}

```

Preparati a creare un Guard Hook

Prima di creare un Guard Hook, è necessario completare i seguenti prerequisiti:

- Devi aver già creato una regola Guard. Per ulteriori informazioni, consulta [Scrivi le regole di Guard per Hooks](#).

- L'utente o il ruolo che crea l'Hook deve disporre di autorizzazioni sufficienti per attivare gli Hook. Per ulteriori informazioni, consulta [Concedi le autorizzazioni IAM per CloudFormation Hooks](#).
- Per utilizzare AWS CLI o un SDK per creare un Guard Hook, devi creare manualmente un ruolo di esecuzione con autorizzazioni IAM e una policy di fiducia che CloudFormation consenta di richiamare un Guard Hook.

Crea un ruolo di esecuzione per un Guard Hook

Un Hook utilizza un ruolo di esecuzione per le autorizzazioni necessarie per richiamare quell'Hook nel tuo Account AWS

Questo ruolo può essere creato automaticamente se crei un Guard Hook da Console di gestione AWS; in caso contrario, devi creare questo ruolo tu stesso.

La sezione seguente mostra come impostare le autorizzazioni per creare il tuo Guard Hook.

Autorizzazioni richieste

Segui le indicazioni riportate in [Create a role using custom trust policies](#) nella Guida per l'utente IAM per creare un ruolo con una policy di fiducia personalizzata.

Quindi, completa i seguenti passaggi per configurare le tue autorizzazioni:

1. Allega la seguente politica di privilegi minimi al ruolo IAM che desideri utilizzare per creare il Guard Hook.

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "s3>ListBucket",  
        "s3GetObject",  
        "s3GetObjectVersion"  
      ],  
      "Resource": [  
        "arn:aws:s3:::my-guard-output-bucket/*",  
        "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. Concedi al tuo Hook il permesso di assumere il ruolo aggiungendo una politica di fiducia al ruolo. Di seguito viene mostrato un esempio di politica di fiducia che è possibile utilizzare.

JSON

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

Attiva un Guard Hook nel tuo account

L'argomento seguente mostra come attivare un Guard Hook nel tuo account, il che lo rende utilizzabile nell'account e nella regione in cui è stato attivato.

Argomenti

- [Attiva un Guard Hook \(console\)](#)
- [Attiva un Guard Hook \(AWS CLI\)](#)
- [Risorse correlate](#)

Attiva un Guard Hook (console)

Per attivare un Guard Hook da utilizzare nel tuo account

1. Accedi Console di gestione AWS e apri la CloudFormation console all'indirizzo <https://console.aws.amazon.com/cloudformation>.
2. Nella barra di navigazione nella parte superiore dello schermo, scegli Regione AWS dove vuoi creare l'Hook in.
3. Nel riquadro di navigazione a sinistra, scegli Hooks.
4. Nella pagina Hooks, scegli Crea un hook, quindi scegli With Guard.
5. Se non hai ancora creato alcuna regola Guard, crea la tua regola Guard, archiviala in Amazon S3 e poi torna a questa procedura. Per iniziare, consulta le regole [Scrivi le regole di Guard per valutare le risorse per Guard Hooks](#) di esempio riportate in.

Se hai già creato la regola Guard e l'hai archiviata in S3, procedi al passaggio successivo.

Note

L'oggetto archiviato in S3 deve avere una delle seguenti estensioni di file: .guard, .zip, o .tar.gz

6. Per il codice sorgente Guard Hook, Store your Guard rules in S3, procedi come segue:
 - Per l'URI S3, specifica il percorso S3 del file delle regole o usa il pulsante Browse S3 per aprire una finestra di dialogo per cercare e selezionare l'oggetto S3.
 - (Facoltativo) Per la versione Object, se il tuo bucket S3 ha il controllo delle versioni abilitato, puoi selezionare una versione specifica dell'oggetto S3.

Guard Hook scarica le regole da S3 ogni volta che viene richiamato l'Hook. Per evitare modifiche o eliminazioni accidentali, consigliamo di utilizzare una versione durante la configurazione di Guard Hook.

7. (Facoltativo) Per il rapporto di output del bucket S3 per Guard, specifica un bucket S3 per archiviare il rapporto di output di Guard. Questo rapporto contiene i risultati delle convalide delle regole Guard.

Per configurare la destinazione del rapporto di output, scegli una delle seguenti opzioni:

- Seleziona la casella di controllo Usa lo stesso bucket in cui sono memorizzate le regole di Guard per utilizzare lo stesso bucket in cui si trovano le regole di Guard.
- Scegli un nome diverso per il bucket S3 per archiviare il rapporto di output di Guard.

 8. (Facoltativo) Espandi i parametri di input delle regole Guard, quindi fornisci le seguenti informazioni in Memorizza i parametri di input delle regole Guard in S3:

- Per l'URI S3, specifica il percorso S3 a un file di parametri o usa il pulsante Sfoglia S3 per aprire una finestra di dialogo per cercare e selezionare l'oggetto S3.
- (Facoltativo) Per la versione Object, se il tuo bucket S3 ha il controllo delle versioni abilitato, puoi selezionare una versione specifica dell'oggetto S3.

9. Scegli Next (Successivo).

10. Per Hook name, scegli una delle seguenti opzioni:

- Fornisci un nome breve e descrittivo che verrà aggiunto dopo `Private::Guard::`. Ad esempio, se inserisci **MyTestHook**, il nome completo di Hook diventa `Private::Guard::MyTestHook`.
- Fornite il nome completo dell'Hook (chiamato anche alias) utilizzando questo formato: **Provider::ServiceName::HookName**

11. Per gli obiettivi Hook, scegli cosa valutare:

- Pile: valuta i modelli di stack quando gli utenti creano, aggiornano o eliminano gli stack.
- Risorse: valuta le modifiche delle singole risorse quando gli utenti aggiornano gli stack.
- Set di modifiche: valuta gli aggiornamenti pianificati quando gli utenti creano set di modifiche.
- API Cloud Control: valuta le operazioni di creazione, aggiornamento o eliminazione avviate dall'API Cloud Control.

12. Per Actions, scegli quali azioni (creazione, aggiornamento, eliminazione) richiameranno il tuo Hook.

13. Per la modalità Hook, scegli come risponde l'Hook quando le regole non superano la valutazione:

- **Avvisa:** invia avvisi agli utenti ma consente il proseguimento delle azioni. Ciò è utile per convalide non critiche o controlli informativi.
 - **Fallito:** impedisce il proseguimento dell'azione. Ciò è utile per applicare rigorose politiche di conformità o sicurezza.
14. Per il ruolo Execution, scegli il ruolo IAM che Hook assume per recuperare le regole di Guard da S3 e, facoltativamente, scrivi un rapporto dettagliato sull'output di Guard. Puoi consentire la creazione automatica CloudFormation di un ruolo di esecuzione per te oppure puoi specificare un ruolo che hai creato.
15. Scegli Next (Successivo).
16. (Facoltativo) Per i filtri Hook, procedi come segue:
- a. Per il filtro Resource, specifica quali tipi di risorse possono richiamare l'Hook. Ciò garantisce che l'Hook venga richiamato solo per le risorse pertinenti.
 - b. Per i criteri di filtraggio, scegli la logica per applicare i filtri del nome dello stack e del ruolo dello stack:
 - Tutti i nomi degli stack e i ruoli dello stack: l'Hook verrà richiamato solo quando tutti i filtri specificati corrispondono.
 - Qualsiasi nome dello stack e ruolo dello stack: l'Hook verrà richiamato se almeno uno dei filtri specificati corrisponde.

 **Note**

Per le operazioni dell'API Cloud Control, tutti i filtri dei nomi degli stack e dei ruoli dello stack vengono ignorati.

- c. Per i nomi Stack, includi o escludi stack specifici dalle invocazioni di Hook.
 - Per **Include**, specificate i nomi degli stack da includere. Usalo quando hai un piccolo set di pile specifiche a cui vuoi rivolgerti. Solo gli stack specificati in questo elenco richiameranno l'Hook.
 - Per **Exclude**, specifica i nomi degli stack da escludere. Usalo quando vuoi invocare l'Hook sulla maggior parte degli stack ma escluderne alcuni specifici. Tutti gli stack tranne quelli elencati qui invocheranno l'Hook.

- d. Per i ruoli Stack, includi o escludi stack specifici dalle invocazioni di Hook in base ai ruoli IAM associati.
 - Per Include, specifica uno o più ruoli IAM ARNs per indirizzare gli stack associati a questi ruoli. Solo le operazioni di stack avviate da questi ruoli richiameranno l'Hook.
 - Per Exclude, specifica uno o più ruoli IAM ARNs per gli stack che desideri escludere. L'Hook verrà richiamato su tutti gli stack ad eccezione di quelli avviati dai ruoli specificati.
17. Scegli Next (Successivo).
18. Nella pagina Rivedi e attiva, rivedi le tue scelte. Per apportare modifiche, scegli Modifica nella sezione da modificare.
19. Quando sei pronto per procedere, scegli Activate Hook.

Attiva un Guard Hook (AWS CLI)

Prima di continuare, conferma di aver creato la regola Guard e il ruolo di esecuzione che utilizzerai con questo Hook. Per ulteriori informazioni, consultare [Scrivi le regole di Guard per valutare le risorse per Guard Hooks](#) e [Crea un ruolo di esecuzione per un Guard Hook](#).

Per attivare un Guard Hook da utilizzare nel tuo account (AWS CLI)

1. Per iniziare ad attivare un Hook, usate il seguente [activate-type](#) comando, sostituendo i segnaposto con i vostri valori specifici. Questo comando autorizza l'Hook a utilizzare un ruolo di esecuzione specificato dal tuo Account AWS

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

2. Per completare l'attivazione dell'Hook, è necessario configurarlo utilizzando un file di configurazione JSON.

Usa il cat comando per creare un file JSON con la seguente struttura. Per ulteriori informazioni, consulta [Riferimento alla sintassi dello schema di configurazione Hook](#).

```
$ cat > config.json
```

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

- **HookInvocationStatus**: Impostato per ENABLED abilitare l'Hook.
 - **TargetOperations**: Specificate le operazioni che l'Hook valuterà.
 - **FailureMode**: impostare su FAIL o su WARN.
 - **ruleLocation**: Sostituisci con l'URI S3 in cui è archiviata la regola. L'oggetto memorizzato in S3 deve avere una delle seguenti estensioni di file:.guard, .zip e. .tar.gz
 - **logBucket**: (Facoltativo) Specificate il nome di un bucket S3 per i report JSON di Guard.
 - **TargetFilters**: Specificate i tipi di azioni che richiameranno l'Hook.
3. Usa il [set-type-configuration](#) comando seguente, insieme al file JSON che hai creato, per applicare la configurazione. Sostituisci i segnaposto con i tuoi valori specifici.

```
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
```

Risorse correlate

Forniamo esempi di modelli che puoi utilizzare per capire come dichiarare un Guard Hook in un CloudFormation modello di pila. Per ulteriori informazioni, consulta [AWS::CloudFormation::GuardHook](#) nella Guida per l'utente di AWS CloudFormation .

Visualizza i log dei Guard Hooks nel tuo account

Quando attivi un Guard Hook, puoi specificare un bucket Amazon S3 come destinazione per il report di output di Hook. Una volta attivato, Hook memorizza automaticamente i risultati delle convalide delle regole Guard nel bucket specificato. Potrai quindi visualizzare questi risultati nella console Amazon S3.

Visualizza i log di Guard Hook nella console Amazon S3

Per visualizzare il file di log di output di Guard Hook

1. Accedi a. <https://console.aws.amazon.com/s3/>
2. Nella barra di navigazione nella parte superiore dello schermo, scegli la Regione AWS.
3. Scegli Buckets.
4. Scegli il bucket che hai selezionato per il rapporto di output di Guard.
5. Scegli il file di registro del rapporto di output di convalida desiderato.
6. Scegli se scaricare il file o aprirlo per visualizzarlo.

Elimina Guard Hooks dal tuo account

Quando non hai più bisogno di un Guard Hook attivato, utilizza le seguenti procedure per eliminarlo dal tuo account.

Per disabilitare temporaneamente un Hook invece di eliminarlo, vedi [Disabilita e abilita gli CloudFormation Hooks](#).

Argomenti

- [Elimina un Guard Hook dal tuo account \(console\)](#)
- [Elimina un Guard Hook dal tuo account \(\)AWS CLI](#)

Elimina un Guard Hook dal tuo account (console)

Per eliminare un Guard Hook dal tuo account

1. Accedi Console di gestione AWS e apri la CloudFormation console all'indirizzo <https://console.aws.amazon.com/cloudformation>.
2. Nella barra di navigazione nella parte superiore dello schermo, scegli Regione AWS dove si trova l'Hook.
3. Dal pannello di navigazione, scegli Hooks.
4. Nella pagina Hooks, trova il Guard Hook che desideri eliminare.
5. Seleziona la casella di controllo accanto al tuo Hook e scegli Elimina.
6. Quando viene richiesta la conferma, digita il nome dell'Hook per confermare l'eliminazione dell'Hook specificato, quindi scegli Elimina.

Elimina un Guard Hook dal tuo account ()AWS CLI

Note

Prima di poter eliminare l'Hook, devi prima disabilitarlo. Per ulteriori informazioni, consulta [Disattiva e abilita un Hook nel tuo account \(\)AWS CLI](#).

Usa il seguente [deactivate-type](#) comando per disattivare un Hook, che lo rimuove dal tuo account. Sostituisci i segnaposto con i tuoi valori specifici.

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

Lambda Hooks

Per utilizzare un AWS Lambda Hook nel tuo account, devi prima attivare l'Hook per l'account e la regione in cui desideri utilizzarlo. L'attivazione di un Hook lo rende utilizzabile nelle operazioni di pila nell'account e nella regione in cui è attivato.

Quando attivi un Lambda Hook, CloudFormation crea una voce nel registro del tuo account per l'Hook attivato come Hook privato. Ciò consente di impostare tutte le proprietà di configurazione

incluse nell'Hook. Le proprietà di configurazione definiscono come l'Hook è configurato per una determinata Account AWS regione.

Argomenti

- [AWS CLI comandi per lavorare con Lambda Hooks](#)
- [Crea funzioni Lambda per valutare le risorse per Lambda Hooks](#)
- [Preparati a creare un Lambda Hook](#)
- [Attiva un Lambda Hook nel tuo account](#)
- [Visualizza i log dei Lambda Hooks nel tuo account](#)
- [Elimina Lambda Hooks dal tuo account](#)

AWS CLI comandi per lavorare con Lambda Hooks

I AWS CLI comandi per lavorare con Lambda Hooks includono:

- [activate-type](#) per avviare il processo di attivazione di un Lambda Hook.
- [set-type-configuration](#) per specificare i dati di configurazione per un Hook nel tuo account.
- [list-types](#) per elencare gli Hooks presenti nel tuo account.
- [describe-type](#) per restituire informazioni dettagliate su uno specifico Hook o una versione specifica di Hook, inclusi i dati di configurazione correnti.
- [deactivate-type](#) per rimuovere un Hook precedentemente attivato dal tuo account.

Crea funzioni Lambda per valutare le risorse per Lambda Hooks

CloudFormation Lambda Hooks ti consente di effettuare valutazioni CloudFormation e AWS Cloud Control API operazioni in base al tuo codice personalizzato. Hook può bloccare l'esecuzione di un'operazione o inviare un avviso al chiamante e consentire all'operazione di continuare. Quando crei un Lambda Hook, puoi configurarlo per intercettare e valutare le seguenti operazioni: CloudFormation

- Operazioni sulle risorse
- Operazioni di stack
- Operazioni di modifica del set

Argomenti

- [Sviluppo di un Lambda Hook](#)
- [Valutazione delle operazioni relative alle risorse con Lambda Hooks](#)
- [Valutazione delle operazioni degli stack con Lambda Hooks](#)
- [Valutazione delle operazioni relative ai set di modifiche con Lambda Hooks](#)

Sviluppo di un Lambda Hook

Quando gli Hooks richiamano la tua Lambda, aspetteranno fino a 30 secondi affinché Lambda valuti l'input. Lambda restituirà una risposta JSON che indica se l'Hook è riuscito o meno.

Argomenti

- [Richiedi un input](#)
- [Input di risposta](#)
- [Esempi](#)

Richiedi un input

L'input passato alla funzione Lambda dipende dall'operazione di destinazione Hook (esempi: stack, resource o change set).

Input di risposta

Per comunicare a Hooks se la richiesta è riuscita o meno, la funzione Lambda deve restituire una risposta JSON.

Di seguito è riportato un esempio di forma della risposta che Hooks si aspetta:

```
{  
  "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

Lo stato dell'Hook. Questo è un campo obbligatorio.

Valori validi: (SUCCESS| FAILED |IN_PROGRESS)

Note

Un Hook può tornare IN_PROGRESS 3 volte. Se non viene restituito alcun risultato, l'Hook fallirà. Per un Lambda Hook, ciò significa che la funzione Lambda può essere richiamata fino a 3 volte.

errorCode

Indica se l'operazione è stata valutata e ritenuta non valida o se si sono verificati errori all'interno dell'Hook che hanno impedito la valutazione. Questo campo è obbligatorio se l'Hook fallisce.

Valori validi: (NonCompliant|InternalFailure)

message

Il messaggio al chiamante che indica il motivo per cui l'Hook ha avuto esito positivo o negativo.

Note

Durante la valutazione CloudFormation delle operazioni, questo campo viene troncato a 4096 caratteri.

Quando si valutano le operazioni dell'API Cloud Control, questo campo viene troncato a 1024 caratteri.

clientRequestToken

Il token di richiesta fornito come input alla richiesta Hook. Questo è un campo obbligatorio.

CallbackContext

Se indichi che hookStatus è IN_PROGRESS, passi un contesto aggiuntivo che viene fornito come input quando la funzione Lambda viene reinvocata.

callbackDelaySeconds

Quanto tempo devono aspettare gli Hook per richiamare nuovamente questo Hook.

annotations

Una serie di oggetti di annotazione che forniscono ulteriori dettagli e indicazioni per la correzione.

Nome dell'annotazione

Un identificatore per l'annotazione.

status

Lo stato di invocazione di Hook. Ciò è utile quando le annotazioni rappresentano una logica con una valutazione pass/fail simile a una regola Guard.

Valori validi: (| |) PASSED FAILED SKIPPED

statusMessage

Spiegazione dello stato specifico.

Messaggio di riparazione

Suggerimento per correggere uno stato. FAILED Ad esempio, se in una risorsa manca la crittografia, puoi indicare come aggiungere la crittografia alla configurazione della risorsa.

Link di riparazione

Un URL HTTP per ulteriori indicazioni sulla riparazione.

severityLevel

Definisce il rischio relativo associato a qualsiasi violazione di questo tipo. Quando si assegnano i livelli di gravità ai risultati della chiamata Hook, è possibile fare riferimento al [framework di AWS Security Hub CSPM gravità](#) come esempio di come strutturare categorie di gravità significative.

Valori validi: (INFORMATIONAL| | |LOW) MEDIUM HIGH CRITICAL

Esempi

Di seguito è riportato un esempio di risposta riuscita:

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

Di seguito è riportato un esempio di risposta non riuscita:

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

Valutazione delle operazioni relative alle risorse con Lambda Hooks

Ogni volta che crei, aggiorni o elimini una risorsa, questa viene considerata un'operazione relativa alle risorse. Ad esempio, se esegui l'aggiornamento di uno CloudFormation stack che crea una nuova risorsa, hai completato un'operazione sulla risorsa. Quando crei, aggiorni o elimini una risorsa utilizzando l'API Cloud Control, anche questa viene considerata un'operazione relativa alle risorse. Puoi configurare il tuo CloudFormation Lambda Hook per la destinazione RESOURCE e CLOUD_CONTROL le operazioni nella configurazione HookTargetOperations.

Note

Il gestore delete Hook viene richiamato solo quando una risorsa viene eliminata utilizzando un trigger operativo dall'API Cloud Control o. `delete-resource` CloudFormation `delete-stack`

Argomenti

- [Sintassi di input delle risorse Lambda Hook](#)
- [Esempio di input per la modifica delle risorse di Lambda Hook](#)

- [Esempio di funzione Lambda per le operazioni sulle risorse](#)

Sintassi di input delle risorse Lambda Hook

Quando la tua Lambda viene richiamata per un'operazione su una risorsa, riceverai un input JSON contenente le proprietà della risorsa, le proprietà proposte e il contesto attorno alla chiamata di Hook.

Di seguito è riportato un esempio di forma dell'input JSON:

```
{  
  "awsAccountId": String,  
  "stackId": String,  
  "changeSetId": String,  
  "hookTypeName": String,  
  "hookTypeVersion": String,  
  "hookModel": {  
    "LambdaFunction": String  
  },  
  "actionInvocationPoint": "CREATE_PRE_PROVISION" or "UPDATE_PRE_PROVISION" or  
  "DELETE_PRE_PROVISION"  
  "requestData": {  
    "targetName": String,  
    "targetType": String,  
    "targetLogicalId": String,  
    "targetModel": {  
      "resourceProperties": {...},  
      "previousResourceProperties": {...}  
    }  
  },  
  "requestContext": {  
    "invocazione": 1,  
    "CallbackContext": null  
  }  
}
```

awsAccountId

L'ID del Account AWS contenitore della risorsa da valutare.

stackId

L'ID dello stack di cui CloudFormation fa parte questa operazione. Questo campo è vuoto se il chiamante è Cloud Control API.

changeSetId

L'ID del set di modifiche che ha avviato l'invocazione di Hook. Questo valore è vuoto se la modifica delle risorse è stata avviata dall'API Cloud Control o dalle operazioni, o. `create-stack` `update-stack` `delete-stack`

hookTypeName

Il nome dell'Hook in esecuzione.

hookTypeVersion

La versione dell'Hook in esecuzione.

hookModel

LambdaFunction

L'attuale Lambda ARN richiamato dall'Hook.

actionInvocationPoint

Il punto esatto della logica di provisioning in cui viene eseguito l'Hook.

Valori validi: (CREATE_PRE_PROVISION| UPDATE_PRE_PROVISION |DELETE_PRE_PROVISION)

requestData

targetName

Il tipo di oggetto da valutare, ad esempio, `AWS::S3::Bucket`.

targetType

Il tipo di oggetto da valutare, ad esempio. `AWS::S3::Bucket` Per le risorse fornite con l'API Cloud Control, questo valore sarà. `RESOURCE`

targetLogicalId

L'ID logico della risorsa da valutare. Se l'origine dell'invocazione di Hook è CloudFormation, questo sarà l'ID della risorsa logica definito nel modello. CloudFormation Se l'origine di questa chiamata di Hook è l'API Cloud Control, questo sarà un valore costruito.

targetModel

resourceProperties

Le proprietà proposte della risorsa da modificare. Se la risorsa viene eliminata, questo valore sarà vuoto.

previousResourceProperties

Le proprietà attualmente associate alla risorsa da modificare. Se la risorsa viene creata, questo valore sarà vuoto.

requestContext

invocazione

L'attuale tentativo di eseguire l'Hook.

CallbackContext

Se l'Hook è stato impostato su ed è `callbackContext` stato restituito `IN_PROGRESS`, sarà qui dopo la reinvocazione.

Esempio di input per la modifica delle risorse di Lambda Hook

L'input di esempio seguente mostra un Lambda Hook che riceverà la definizione della `AWS::DynamoDB::Table` risorsa da aggiornare, dove l'`ReadCapacityUnits` di `ProvisionedThroughput` viene modificato da 3 a 10. Questi sono i dati a disposizione di Lambda per la valutazione.

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

```
{  
    "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
}
}
```

Per visualizzare tutte le proprietà disponibili per il tipo di risorsa, consulta [AWS::DynamoDB::Table](#).

Esempio di funzione Lambda per le operazioni sulle risorse

Di seguito è riportata una semplice funzione che fallisce qualsiasi aggiornamento delle risorse a DynamoDB, che tenta di impostare of su un ReadCapacity valore ProvisionedThroughput maggiore di 10. Se l'Hook ha successo, al chiamante verrà visualizzato il messaggio «ReadCapacity è configurato correttamente». Se la richiesta non viene convalidata, l'Hook avrà esito negativo e lo stato «ReadCapacity non può essere superiore a 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
```

Valutazione delle operazioni degli stack con Lambda Hooks

Ogni volta che crei, aggiorni o elimini uno stack con un nuovo modello, puoi configurare il tuo CloudFormation Lambda Hook per iniziare a valutare il nuovo modello e potenzialmente bloccare il proseguimento dell'operazione dello stack. Puoi configurare il tuo CloudFormation Lambda Hook per eseguire STACK operazioni mirate nella configurazione `HookTargetOperations`.

Argomenti

- [Sintassi di input dello stack Lambda Hook](#)
- [Esempio di input per la modifica dello stack Lambda Hook](#)

- [Esempio di funzione Lambda per operazioni in pila](#)

Sintassi di input dello stack Lambda Hook

Quando la tua Lambda viene richiamata per un'operazione stack, riceverai una richiesta JSON contenente il contesto di invocazione Hook e il contesto di richiesta. `actionInvocationPoint` A causa delle dimensioni dei CloudFormation modelli e della dimensione di input limitata accettata dalle funzioni Lambda, i modelli effettivi vengono archiviati in un oggetto Amazon S3. L'input di `requestData` include un URL rassegnato di Amazon S3 a un altro oggetto, che contiene la versione attuale e precedente del modello.

Di seguito è riportato un esempio di forma dell'input JSON:

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

clientRequesttoken

Il token di richiesta che è stato fornito come input alla richiesta Hook. Questo è un campo obbligatorio.

awsAccountId

L'ID dello stack Account AWS contenente lo stack da valutare.

stackID

L'ID dello stack. CloudFormation

changeSetId

L'ID del set di modifiche che ha avviato l'invocazione di Hook. Questo valore è vuoto se la modifica dello stack è stata avviata dall'API Cloud Control o dalle operazioni,, or. `create-stack` `update-stack` `delete-stack`

hookTypeName

Il nome dell'Hook in esecuzione.

hookTypeVersion

La versione dell'Hook in esecuzione.

hookModel**LambdaFunction**

L'attuale Lambda ARN richiamato dall'Hook.

actionInvocationPoint

Il punto esatto della logica di provisioning in cui viene eseguito l'Hook.

Valori validi: (CREATE_PRE_PROVISION| UPDATE_PRE_PROVISION |DELETE_PRE_PROVISION)

requestData**targetName**

Questo valore sarà STACK.

targetType

Questo valore sarà STACK.

targetLogicalId

Il nome dello stack.

payload

L'URL predefinito di Amazon S3 contenente un oggetto JSON con le definizioni del modello correnti e precedenti.

requestContext

Se l'Hook viene reinvocato, questo oggetto verrà impostato.

invocation

L'attuale tentativo di eseguire l'Hook.

callbackContext

Se l'Hook è stato impostato su IN_PROGRESS ed callbackContext è stato restituito, sarà qui dopo la reinvocazione.

La payload proprietà nei dati della richiesta è un URL che il codice deve recuperare. Una volta ricevuto l'URL, si ottiene un oggetto con lo schema seguente:

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

template

Il CloudFormation modello completo fornito a create-stack or update-stack. Può essere una stringa JSON o YAML a seconda di ciò che è stato fornito. CloudFormation

Nelle delete-stack operazioni, questo valore sarà vuoto.

previousTemplate

Il CloudFormation modello precedente. Può essere una stringa JSON o YAML a seconda di cosa è stata fornita. CloudFormation

Nelle delete-stack operazioni, questo valore sarà vuoto.

Esempio di input per la modifica dello stack Lambda Hook

Di seguito è riportato un esempio di input per la modifica dello stack. Hook sta valutando una modifica che aggiorna ObjectLockEnabled a true e aggiunge una coda Amazon SQS:

```
{  
  "clientRequestToken": "f8da6d11-b23f-48f4-814c-0fb6a667f50e",
```

```

"awsAccountId": "123456789012",
"stackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/
MyStack/1a2345b6-0000-00a0-a123-00abc0abc000",
"changeSetId": null,
"hookTypeName": "my::lambda::stackhook",
"hookTypeVersion": "00000008",
"hookModel": {
    "LambdaFunction": "arn:aws:lambda:us-west-2:123456789012:function:MyFunction"
},
"actionInvocationPoint": "UPDATE_PRE_PROVISION",
"requestData": {
    "targetName": "STACK",
    "targetType": "STACK",
    "targetLogicalId": "my-cloudformation-stack",
    "payload": "https://s3....."
},
"requestContext": {
    "invocation": 1,
    "callbackContext": null
}
}

```

Questo è un esempio payload di: requestData

```

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

```

Esempio di funzione Lambda per operazioni in pila

L'esempio seguente è una semplice funzione che scarica il payload dell'operazione stack, analizza il modello JSON e restituisce. 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

Per usare Python, devi importare la `requests` libreria. A tale scopo, è necessario includere la libreria nel pacchetto di distribuzione durante la creazione della funzione Lambda. Per ulteriori informazioni, consulta [Creazione di un pacchetto di distribuzione.zip con dipendenze](#) nella Guida per gli AWS Lambda sviluppatori.

```
import json
import requests

def lambda_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
```

Valutazione delle operazioni relative ai set di modifiche con Lambda Hooks

Ogni volta che crei un set di modifiche, puoi configurare il tuo CloudFormation Lambda Hook per valutare prima il nuovo set di modifiche e potenzialmente bloccarne l'esecuzione. Puoi configurare il tuo CloudFormation Lambda Hook per eseguire CHANGE_SET operazioni mirate nella configurazione HookTargetOperations.

Argomenti

- [Sintassi di input del set di modifiche Lambda Hook](#)
- [Esempio di input di modifica del set di modifiche Lambda Hook](#)
- [Esempio di funzione Lambda per le operazioni sui set di modifiche](#)

Sintassi di input del set di modifiche Lambda Hook

L'input per le operazioni sui set di modifiche è simile alle operazioni sullo stack, ma il payload del set include `requestData` anche un elenco di modifiche alle risorse introdotte dal set di modifiche.

Di seguito è riportato un esempio di forma dell'input JSON:

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

clientRequesttoken

Il token di richiesta che è stato fornito come input alla richiesta Hook. Questo è un campo obbligatorio.

awsAccountId

L'ID dello stack Account AWS contenente lo stack da valutare.

stackID

L'ID dello stack. CloudFormation

changeSetId

L'ID del set di modifiche che ha avviato l'invocazione di Hook.

hookTypeName

Il nome dell'Hook in esecuzione.

hookTypeVersion

La versione dell'Hook in esecuzione.

hookModel**LambdaFunction**

L'attuale Lambda ARN richiamato dall'Hook.

requestData**targetName**

Questo valore sarà. CHANGE_SET

targetType

Questo valore saràCHANGE_SET.

targetLogicalId

Il set di modifiche ARN..

payload

L'URL predefinito di Amazon S3 contenente un oggetto JSON con il modello corrente e un elenco di modifiche introdotte da questo set di modifiche.

requestContext

Se l'Hook viene reinvocato, questo oggetto verrà impostato.

invocation

L'attuale tentativo di eseguire l'Hook.

callbackContext

Se l'Hook è stato impostato su IN_PROGRESS ed callbackContext è stato restituito, sarà qui dopo la reinvocazione.

La payload proprietà nei dati della richiesta è un URL che il codice deve recuperare. Una volta ricevuto l'URL, si ottiene un oggetto con lo schema seguente:

```
{  
  "template  "changedResources    {  
      "action      "beforeContext      "afterContext      "lineNumber      "logicalResourceId      "resourceType    }  
  ]  
}
```

template

Il CloudFormation modello completo fornito a `create-stack` or `update-stack`. Può essere una stringa JSON o YAML a seconda di ciò che è stato fornito. CloudFormation

changedResources

Un elenco di risorse modificate.

action

Il tipo di modifica applicata alla risorsa.

Valori validi: (CREATE| UPDATE |DELETE)

beforeContext

Una stringa JSON delle proprietà della risorsa prima della modifica. Questo valore è nullo quando la risorsa viene creata. Tutti i valori booleani e numerici in questa stringa JSON sono STRINGS.

afterContext

Una stringa JSON delle proprietà delle risorse se questo set di modifiche viene eseguito. Questo valore è nullo quando la risorsa viene eliminata. Tutti i valori booleani e numerici in questa stringa JSON sono STRINGS.

lineNumber

Il numero di riga del modello che ha causato questa modifica. Se l'azione è, DELETE questo valore sarà nullo.

logicalResourceId

L'ID della risorsa logica della risorsa da modificare.

resourceType

Il tipo di risorsa che viene modificata.

Esempio di input di modifica del set di modifiche Lambda Hook

Di seguito è riportato un esempio di input per la modifica del set di modifiche. Nell'esempio seguente, è possibile visualizzare le modifiche introdotte dal set di modifiche. La prima modifica consiste nell'eliminare una coda chiamata CoolQueue. La seconda modifica consiste nell'aggiungere una nuova coda chiamata NewCoolQueue. L'ultima modifica è un aggiornamento di 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  
  }  
}
```

Questo è un esempio payload di `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"}]}]  
}  
]  
}
```

Esempio di funzione Lambda per le operazioni sui set di modifiche

L'esempio seguente è una semplice funzione che scarica il payload dell'operazione Change Set, esegue un ciclo su ogni modifica e quindi stampa le proprietà before e after prima di restituire 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

Per usare Python, devi importare la `requests` libreria. A tale scopo, è necessario includere la libreria nel pacchetto di distribuzione durante la creazione della funzione Lambda. Per ulteriori informazioni, consulta [Creazione di un pacchetto di distribuzione.zip con dipendenze](#) nella Guida per gli AWS Lambda sviluppatori.

```
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
```

Preparati a creare un Lambda Hook

Prima di creare un Lambda Hook, devi completare i seguenti prerequisiti:

- È necessario aver già creato una funzione Lambda. Per ulteriori informazioni, consulta [Crea funzioni Lambda per Hooks](#).
- L'utente o il ruolo che crea l'Hook deve disporre di autorizzazioni sufficienti per attivare gli Hook. Per ulteriori informazioni, consulta [Concedi le autorizzazioni IAM per CloudFormation Hooks](#).
- Per utilizzare AWS CLI o un SDK per creare un Lambda Hook, devi creare manualmente un ruolo di esecuzione con autorizzazioni IAM e una policy di fiducia per CloudFormation consentire di richiamare un Lambda Hook.

Creare un ruolo di esecuzione per un Lambda Hook

Un Hook utilizza un ruolo di esecuzione per le autorizzazioni necessarie per richiamare quell'Hook nel tuo Account AWS

Questo ruolo può essere creato automaticamente se crei un Lambda Hook da Console di gestione AWS; in caso contrario, devi crearlo tu stesso.

La sezione seguente mostra come configurare le autorizzazioni per creare il tuo Lambda Hook.

Autorizzazioni richieste

Segui le indicazioni riportate in [Create a role using custom trust policies](#) nella Guida per l'utente IAM per creare un ruolo con una policy di fiducia personalizzata.

Quindi, completa i seguenti passaggi per configurare le tue autorizzazioni:

1. Allega la seguente politica di privilegi minimi al ruolo IAM che desideri utilizzare per creare il Lambda Hook.

JSON

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

2. Concedi al tuo Hook il permesso di assumere il ruolo aggiungendo una policy di fiducia al ruolo. Di seguito viene mostrato un esempio di politica di fiducia che è possibile utilizzare.

JSON

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

}

Attiva un Lambda Hook nel tuo account

L'argomento seguente mostra come attivare un Lambda Hook nel tuo account, il che lo rende utilizzabile nell'account e nella regione in cui è stato attivato.

Argomenti

- [Attiva un Lambda Hook \(console\)](#)
- [Attiva un Lambda Hook \(\)AWS CLI](#)
- [Risorse correlate](#)

Attiva un Lambda Hook (console)

Per attivare un Lambda Hook da utilizzare nel tuo account

1. Accedi Console di gestione AWS e apri la CloudFormation console all'indirizzo <https://console.aws.amazon.com/cloudformation>.
2. Nella barra di navigazione nella parte superiore dello schermo, scegli Regione AWS dove vuoi creare l'Hook in.
3. Se non hai creato una funzione Lambda per l'Hook, procedi come segue:
 - Aprire la [pagina Functions \(Funzioni\)](#) nella console Lambda.
 - Crea la funzione Lambda che utilizzerai con questo Hook, quindi torna a questa procedura. Per ulteriori informazioni, consulta [Crea funzioni Lambda per valutare le risorse per Lambda Hooks](#).

Se hai già creato la tua funzione Lambda, procedi al passaggio successivo.

4. Nel riquadro di navigazione a sinistra, scegli Hooks.
5. Nella pagina Hooks, scegli Crea un hook, quindi scegli Con Lambda.
6. Per Hook name, scegli una delle seguenti opzioni:
 - Fornisci un nome breve e descrittivo che verrà aggiunto dopo `Private::Lambda::`. Ad esempio, se inserisci **MyTestHook**, il nome completo di Hook diventa `Private::Lambda::MyTestHook`.

- Fornite il nome completo dell'Hook (chiamato anche alias) utilizzando questo formato:
Provider::ServiceName::HookName
7. Per la funzione Lambda, fornisci la funzione Lambda da utilizzare con questo Hook. È possibile utilizzare:
- L'Amazon Resource Name (ARN) completo senza suffisso.
 - Un ARN qualificato con un suffisso di versione o alias.
8. Per gli obiettivi Hook, scegli cosa valutare:
- Pile: valuta i modelli di stack quando gli utenti creano, aggiornano o eliminano gli stack.
 - Risorse: valuta le modifiche delle singole risorse quando gli utenti aggiornano gli stack.
 - Set di modifiche: valuta gli aggiornamenti pianificati quando gli utenti creano set di modifiche.
 - API Cloud Control: valuta le operazioni di creazione, aggiornamento o eliminazione avviate dall'API [Cloud](#) Control.
9. Per Actions, scegli quali azioni (creazione, aggiornamento, eliminazione) richiameranno il tuo Hook.
10. Per la modalità Hook, scegli come risponde l'Hook quando la funzione Lambda richiamata dall'Hook restituisce una risposta: FAILED
- Avvisa: invia avvisi agli utenti ma consente il proseguimento delle azioni. Ciò è utile per convalide non critiche o controlli informativi.
 - Fallito: impedisce il proseguimento dell'azione. Ciò è utile per applicare rigorose politiche di conformità o sicurezza.
11. Per il ruolo Execution, scegli il ruolo IAM che Hook assume per richiamare la tua funzione Lambda. Puoi consentire la creazione automatica CloudFormation di un ruolo di esecuzione per te oppure puoi specificare un ruolo che hai creato.
12. Scegli Next (Successivo).
13. (Facoltativo) Per i filtri Hook, procedi come segue:
- Per il filtro Resource, specifica quali tipi di risorse possono richiamare l'Hook. Ciò garantisce che l'Hook venga richiamato solo per le risorse pertinenti.
 - Per i criteri di filtraggio, scegli la logica per applicare i filtri del nome dello stack e del ruolo dello stack:

- Tutti i nomi degli stack e i ruoli dello stack: l'Hook verrà richiamato solo quando tutti i filtri specificati corrispondono.
- Qualsiasi nome dello stack e ruolo dello stack: l'Hook verrà richiamato se almeno uno dei filtri specificati corrisponde.

 Note

Per le operazioni dell'API Cloud Control, tutti i filtri dei nomi degli stack e dei ruoli dello stack vengono ignorati.

- Per i nomi Stack, includi o escludi stack specifici dalle invocazioni di Hook.
 - Per Include, specificate i nomi degli stack da includere. Usalo quando hai un piccolo set di pile specifiche a cui vuoi rivolgerti. Solo gli stack specificati in questo elenco richiameranno l'Hook.
 - Per Exclude, specifica i nomi degli stack da escludere. Usalo quando vuoi invocare l'Hook sulla maggior parte degli stack ma escluderne alcuni specifici. Tutti gli stack tranne quelli elencati qui invocheranno l'Hook.
 - Per i ruoli Stack, includi o escludi stack specifici dalle invocazioni di Hook in base ai ruoli IAM associati.
 - Per Include, specifica uno o più ruoli IAM ARNs per indirizzare gli stack associati a questi ruoli. Solo le operazioni di stack avviate da questi ruoli richiameranno l'Hook.
 - Per Exclude, specifica uno o più ruoli IAM ARNs per gli stack che desideri escludere. L'Hook verrà richiamato su tutti gli stack ad eccezione di quelli avviati dai ruoli specificati.
14. Scegli Next (Successivo).
 15. Nella pagina Rivedi e attiva, rivedi le tue scelte. Per apportare modifiche, scegli Modifica nella sezione da modificare.
 16. Quando sei pronto per procedere, scegli Activate Hook.

Attiva un Lambda Hook ()AWS CLI

Prima di continuare, conferma di aver creato la funzione Lambda e il ruolo di esecuzione che utilizzerai con questo Hook. Per ulteriori informazioni, consultare [Crea funzioni Lambda per valutare le risorse per Lambda Hooks](#) e [Creare un ruolo di esecuzione per un Lambda Hook](#).

Per attivare un Lambda Hook da utilizzare nel tuo account ()AWS CLI

1. Per iniziare ad attivare un Hook, usa il seguente [activate-type](#) comando, sostituendo i segnaposto con i tuoi valori specifici. Questo comando autorizza l'Hook a utilizzare un ruolo di esecuzione specificato dal tuo Account AWS

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

2. Per completare l'attivazione dell'Hook, è necessario configurarlo utilizzando un file di configurazione JSON.

Usa il cat comando per creare un file JSON con la seguente struttura. Per ulteriori informazioni, consulta [Riferimento alla sintassi dello schema di configurazione Hook](#).

```
$ 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: Impostato per ENABLED abilitare l'Hook.
 - TargetOperations: Specificate le operazioni che l'Hook valuterà.
 - FailureMode: impostare su FAIL o su WARN.
 - LambdaFunction: Specificare l'ARN della funzione Lambda.
 - TargetFilters: Specificate i tipi di azioni che richiameranno l'Hook.
3. Usa il [set-type-configuration](#) comando seguente, insieme al file JSON che hai creato, per applicare la configurazione. Sostituisci i segnaposto con i tuoi valori specifici.

```
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
```

Risorse correlate

Forniamo esempi di modelli che puoi utilizzare per capire come dichiarare un Lambda Hook in CloudFormation un modello di pila. Per ulteriori informazioni, consulta [AWS::CloudFormation::LambdaHook](#) nella Guida per l'utente di AWS CloudFormation .

Visualizza i log dei Lambda Hooks nel tuo account

Quando si utilizza un Lambda Hook, il file di registro del rapporto di output di convalida si trova nella console Lambda.

Visualizza i log di Lambda Hook nella console Lambda

Per visualizzare il file di registro di output di Lambda Hook

1. Accedi alla console Lambda.
2. Nella barra di navigazione nella parte superiore dello schermo, scegli la Regione AWS.
3. Scegli Funzioni.
4. Scegli la funzione Lambda desiderata.
5. Seleziona la scheda Test.
6. Scegli CloudWatch Logs Live Trail
7. Scegli il menu a discesa e seleziona i gruppi di log che desideri visualizzare.

8. Scegli Avvia. Il registro verrà visualizzato nella finestra CloudWatch Logs Live Trail. Scegli Visualizza in colonne o Visualizza in testo semplice a seconda delle tue preferenze.
 - Puoi aggiungere altri filtri ai risultati aggiungendoli nel campo Aggiungi modello di filtro. Questo campo consente di filtrare i risultati per includere solo gli eventi che corrispondono al modello specificato.

Per ulteriori informazioni sulla visualizzazione dei log per le funzioni Lambda, [vedere CloudWatch Visualizzazione dei log per le funzioni Lambda](#).

Elimina Lambda Hooks dal tuo account

Quando non hai più bisogno di un Lambda Hook attivato, utilizza le seguenti procedure per eliminarlo dal tuo account.

Per disabilitare temporaneamente un Hook anziché eliminarlo, consulta. [Disabilita e abilita gli CloudFormation Hooks](#)

Argomenti

- [Elimina un Lambda Hook nel tuo account \(console\)](#)
- [Elimina un Lambda Hook nel tuo account \(\)AWS CLI](#)

Elimina un Lambda Hook nel tuo account (console)

Per eliminare un Lambda Hook dal tuo account

1. Accedi Console di gestione AWS e apri la CloudFormation console all'indirizzo <https://console.aws.amazon.com/cloudformation>.
2. Nella barra di navigazione nella parte superiore dello schermo, scegli Regione AWS dove si trova l'Hook.
3. Dal pannello di navigazione, scegli Hooks.
4. Nella pagina Hooks, trova il Lambda Hook che desideri eliminare.
5. Seleziona la casella di controllo accanto al tuo Hook e scegli Elimina.
6. Quando viene richiesta la conferma, digita il nome dell'Hook per confermare l'eliminazione dell'Hook specificato, quindi scegli Elimina.

Elimina un Lambda Hook nel tuo account ()AWS CLI

Note

Prima di poter eliminare l'Hook, devi prima disabilitarlo. Per ulteriori informazioni, consulta [Disattiva e abilita un Hook nel tuo account \(\)AWS CLI](#).

Usa il seguente [deactivate-type](#) comando per disattivare un Hook, che lo rimuove dal tuo account. Sostituisci i segnaposto con i tuoi valori specifici.

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

Sviluppo di hook personalizzati utilizzando la CLI CloudFormation

Questa sezione è dedicata ai clienti che desiderano sviluppare Hooks personalizzati e registrarli nel CloudFormation registro. Fornisce una panoramica della struttura degli CloudFormation Hooks e guide per lo sviluppo, la registrazione, il test, la gestione e la pubblicazione dei propri Hooks con Python o Java.

Esistono tre passaggi principali nello sviluppo di un Hook personalizzato:

1. Avviare

Per sviluppare Hooks personalizzati, è necessario configurare e utilizzare la CloudFormation CLI. Per avviare un progetto Hook e i relativi file richiesti, usa il comando CloudFormation [init](#) CLI e specifica che desideri creare un Hook. Per ulteriori informazioni, consulta [Avvio di un progetto CloudFormation Hooks personalizzato](#).

2. Modello

Per modellare, creare e convalidare lo schema Hook, definisci l'Hook, le sue proprietà e i relativi attributi.

La CloudFormation CLI crea funzioni di gestione vuote che corrispondono a un punto di invocazione Hook specifico. Aggiungi la tua logica a questi gestori per controllare cosa succede durante la chiamata di Hook in ogni fase del ciclo di vita di destinazione. Per ulteriori informazioni, consulta [Modellazione di ganci personalizzati CloudFormation](#).

3. Registrati

Per registrare un Hook, invia il tuo Hook affinché venga registrato come estensione privata o pubblica di terze parti. Registra il tuo Hook con l'[submit](#) operazione. Per ulteriori informazioni, consulta [Registrazione di un Hook personalizzato con CloudFormation](#).

Le seguenti attività sono associate alla registrazione del tuo Hook:

- Pubblica: gli hook vengono pubblicati nel registro.
- Configura: gli hook vengono configurati quando la configurazione del tipo viene richiamata sugli stack.

Note

Gli hook scadranno dopo 30 secondi e riproveranno fino a 3 volte. Per ulteriori informazioni, consulta [Limiti di timeout e nuovi tentativi](#).

Argomenti

- [Prerequisiti per lo sviluppo di ganci personalizzati CloudFormation](#)
- [Avvio di un progetto CloudFormation Hooks personalizzato](#)
- [Modellazione di ganci personalizzati CloudFormation](#)
- [Registrazione di un Hook personalizzato con CloudFormation](#)
- [Testare un Hook personalizzato nel tuo Account AWS](#)
- [Aggiornamento di un Hook personalizzato](#)
- [Annullamento della registrazione di un Hook personalizzato dal registro CloudFormation](#)
- [Ganci di pubblicazione per uso pubblico](#)
- [Riferimento alla sintassi dello schema per CloudFormation Hooks](#)

Prerequisiti per lo sviluppo di ganci personalizzati CloudFormation

Puoi sviluppare un Hook personalizzato con Java o Python. Di seguito sono riportati i prerequisiti per lo sviluppo di Hooks personalizzati:

Prerequisiti Java

- [Apache Maven](#)

- [JDK 17](#)

 Note

Se intendi utilizzare la [CloudFormation Command Line Interface \(CLI\)](#) per avviare un progetto Hooks per Java, devi installare anche Python 3.8 o versione successiva. Il plugin Java per la CloudFormation CLI può essere installato tramite (gestore di pacchetti di pip Python), che è distribuito con Python.

[Per implementare i gestori Hook per il progetto Java Hooks, puoi scaricare i file di esempio del gestore Java Hook.](#)

Prerequisiti Python

- [Python versione 3.8](#) o successiva.

[Per implementare i gestori Hook per il tuo progetto Python Hooks, puoi scaricare i file di esempio del gestore Python Hook.](#)

Autorizzazioni per lo sviluppo di Hooks

Oltre ai permessi CloudFormation `CreateUpdate`, e `Delete` stack, avrai bisogno di accedere alle seguenti operazioni. AWS CloudFormation L'accesso a queste operazioni è gestito tramite la policy del CloudFormation tuo ruolo IAM.

- [register-type](#)
- [list-types](#)
- [deregister-type](#)
- [set-type-configuration](#)

Per ulteriori informazioni, consulta [Concedi le autorizzazioni IAM per CloudFormation Hooks.](#)

Configura un ambiente di sviluppo per Hooks

Per sviluppare Hooks, è necessario avere familiarità con i [CloudFormation template](#) e con Python o Java.

Per installare la CloudFormation CLI e i plugin associati:

1. Installa la CloudFormation CLI con pip, il gestore di pacchetti Python.

```
pip3 install cloudformation-cli
```

2. Installa il plugin Python o Java per la CLI CloudFormation .

Python

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

Java

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

Per aggiornare la CloudFormation CLI e il plug-in, puoi utilizzare l'opzione di aggiornamento.

Python

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

Java

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

Avvio di un progetto CloudFormation Hooks personalizzato

Il primo passo per creare un progetto Hooks personalizzato è avviare il progetto. Puoi usare il CloudFormation CLI `init` comando per avviare il tuo progetto Hooks personalizzato.

Il `init` comando avvia una procedura guidata che guida l'utente nella configurazione del progetto, incluso un file di schema Hooks. Utilizzate questo file di schema come punto di partenza per definire la forma e la semantica dei vostri Hooks. Per ulteriori informazioni, consulta [Sintassi dello schema](#).

Per avviare un progetto Hook:

1. Crea una directory per il progetto.

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

2. Passa alla nuova directory.

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

3. Usa il CloudFormation CLI init comando per avviare il progetto.

```
cfn init
```

Questo comando restituisce il seguente output.

```
Initializing new project
```

4. Il init comando avvia una procedura guidata che guida l'utente nella configurazione del progetto. Quando richiesto, immettete h per specificare un progetto Hooks.

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

```
h
```

5. Immettete un nome per il tipo di Hook.

```
What's the name of your hook type?
```

```
(Organization::Service::Hook)
```

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

6. Se è installato solo un plug-in in una lingua, viene selezionato per impostazione predefinita. Se è installato più di un plug-in linguistico, puoi scegliere la lingua desiderata. Inserisci una selezione numerica per la lingua che preferisci.

```
Select a language for code generation:
```

```
[1] java
[2] python38
[3] python39
(enter an integer):
```

7. Configura il packaging in base al linguaggio di sviluppo scelto.

Python

(Facoltativo) Scegli Docker per un packaging indipendente dalla piattaforma. Sebbene Docker non sia necessario, è altamente consigliato per semplificare l'imballaggio.

Use docker for platform-independent packaging (Y/n)?

This is highly recommended unless you are experienced with cross-platform Python packaging.

Java

Imposta il nome del pacchetto Java e scegli un modello codegen. È possibile utilizzare il nome del pacchetto predefinito o crearne uno nuovo.

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

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

Risultati: Il progetto è stato avviato con successo e sono stati generati i file necessari per sviluppare un Hook. Quello che segue è un esempio delle directory e dei file che compongono un progetto Hooks per Python 3.8.

```
mycompany-testing-mytesthook.json
rdk.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

I file nella `src` directory vengono creati in base alla lingua selezionata. Ci sono alcuni commenti ed esempi utili nei file generati. Alcuni file, ad esempio `models.py`, vengono aggiornati automaticamente in un passaggio successivo quando si esegue il `generate` comando per aggiungere codice di runtime per i gestori.

Modellazione di ganci personalizzati CloudFormation

La modellazione di CloudFormation Hooks personalizzati implica la creazione di uno schema che definisce l'Hook, le sue proprietà e i relativi attributi. Quando crei il tuo progetto Hook personalizzato utilizzando il `cfn init` comando, uno schema Hook di esempio viene creato come file di testo in formato JSON, `hook-name.json`

I punti di invocazione di Target e le azioni target specificano il punto esatto in cui viene richiamato l'Hook. I gestori Hook ospitano una logica personalizzata eseguibile per questi punti. Ad esempio, un'azione mirata dell'CREATE operazione utilizza un `preCreate` gestore. Il codice scritto nel gestore verrà invocato quando le destinazioni e i servizi Hook eseguono un'azione corrispondente. I target Hook sono la destinazione in cui vengono richiamati gli hook. È possibile specificare obiettivi come risorse CloudFormation pubbliche, risorse private o risorse personalizzate. Gli hook supportano un numero illimitato di obiettivi Hook.

Lo schema contiene le autorizzazioni necessarie per l'Hook. La creazione dell'Hook richiede di specificare le autorizzazioni per ogni gestore Hook. CloudFormation incoraggia gli autori a scrivere politiche che seguano i consigli di sicurezza standard che prevedono la concessione dei privilegi minimi o la concessione solo delle autorizzazioni necessarie per eseguire un'attività. Determina cosa devono fare gli utenti (e i ruoli), quindi crea politiche che consentano loro di eseguire solo le attività relative alle operazioni Hook. CloudFormation utilizza queste autorizzazioni per definire le autorizzazioni fornite dagli utenti di Hook. Queste autorizzazioni vengono trasmesse all'Hook. I gestori di Hook utilizzano queste autorizzazioni per accedere alle risorse. AWS

Puoi usare il seguente file di schema come punto di partenza per definire il tuo Hook. Utilizzate lo schema Hook per specificare quali gestori desiderate implementare. Se scegli di non implementare un gestore specifico, rimuovilo dalla sezione dedicata ai gestori dello schema Hook. Per ulteriori dettagli sullo schema, vedere [Sintassi dello schema](#)

```
{  
  "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
}

```

Argomenti

- [Modellazione di CloudFormation hook personalizzati utilizzando Java](#)
- [Modellazione di CloudFormation Hooks personalizzati usando Python](#)

Modellazione di CloudFormation hook personalizzati utilizzando Java

La modellazione di CloudFormation Hooks personalizzati implica la creazione di uno schema che definisce l'Hook, le sue proprietà e i relativi attributi. Questo tutorial illustra la modellazione di Hooks personalizzati utilizzando Java.

Fase 1: Aggiungere le dipendenze del progetto

I progetti Hooks basati su Java si basano sul pom.xml file di Maven come dipendenza. Espandi la sezione seguente e copia il codice sorgente nel pom.xml file nella radice del progetto.

Dipendenze del progetto Hook (pom.xml)

```

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

```

```
<modelVersion>4.0.0</modelVersion>

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

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

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

<dependencies>
    <!-- https://mvnrepository.com/artifact/software.amazon.cloudformation/aws-cloudformation-rpdk-java-plugin -->
    <dependency>
        <groupId>software.amazon.cloudformation</groupId>
        <artifactId>aws-cloudformation-rpdk-java-plugin</artifactId>
```

```
<version>[2.0.0,3.0.0)</version>
</dependency>

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

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

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

<!-- https://mvnrepository.com/artifact/commons-io/commons-io -->
```

```
<dependency>
    <groupId>commons-io</groupId>
    <artifactId>commons-io</artifactId>
    <version>${commons-io.version}</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.apache.commons/commons-lang3 -->
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-lang3</artifactId>
    <version>3.9</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.apache.commons/commons-collections4
-->
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-collections4</artifactId>
    <version>4.4</version>
</dependency>
<!-- https://mvnrepository.com/artifact/com.google.guava/guava -->
<dependency>
    <groupId>com.google.guava</groupId>
    <artifactId>guava</artifactId>
    <version>29.0-jre</version>
</dependency>
<!-- https://mvnrepository.com/artifact/com.amazonaws/aws-java-sdk-
cloudformation -->
<dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-java-sdk-cloudformation</artifactId>
    <version>1.11.555</version>
    <scope>test</scope>
</dependency>

<!-- https://mvnrepository.com/artifact/commons-codec/commons-codec -->
<dependency>
    <groupId>commons-codec</groupId>
    <artifactId>commons-codec</artifactId>
    <version>1.14</version>
</dependency>
<!-- https://mvnrepository.com/artifact/software.amazon.cloudformation/aws-
cloudformation-resource-schema -->
<dependency>
    <groupId>software.amazon.cloudformation</groupId>
    <artifactId>aws-cloudformation-resource-schema</artifactId>
```

```
        <version>[2.0.5, 3.0.0)</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.dataformat/
jackson-databind -->
    <dependency>
        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-databind</artifactId>
        <version>${jackson.version}</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.dataformat/
jackson-dataformat-cbor -->
    <dependency>
        <groupId>com.fasterxml.jackson.dataformat</groupId>
        <artifactId>jackson-dataformat-cbor</artifactId>
        <version>${jackson.version}</version>
    </dependency>

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

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

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

```
</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>
  </execution>
  <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>
```

Passaggio 2: generare il pacchetto del progetto Hook

Genera il tuo pacchetto di progetto Hook. CloudFormation CLI crea funzioni di gestione vuote che corrispondono a specifiche azioni Hook nel ciclo di vita di destinazione, come definito nella specifica Hook.

```
cfn generate
```

Questo comando restituisce il seguente output.

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

Note

Assicurati che i runtime Lambda evitino l'uso up-to-date di una versione obsoleta. Per ulteriori informazioni, consulta [Aggiornamento dei runtime Lambda per i tipi di risorse](#) e gli hook.

Fase 3: Aggiungere i gestori Hook

Aggiungi il tuo codice di runtime del gestore Hook ai gestori che scegli di implementare. Ad esempio, puoi aggiungere il seguente codice per la registrazione.

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

CloudFormation CLIGenera un Plain Old Java Objects (JavaPOJO). Di seguito sono riportati esempi di output generati daAWS::S3::Bucket.

Example WSS3 .java BucketTargetModel

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

import ...


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

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

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

  @JsonProperty("WebsiteConfiguration")
  private WebsiteConfiguration websiteConfiguration;

  @JsonProperty("DualStackDomainName")
  private String dualStackDomainName;

  @JsonProperty("AccessControl")
  private String accessControl;

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

```
@JsonProperty("AccelerateConfiguration")
private AccelerateConfiguration accelerateConfiguration;

@JsonProperty("PublicAccessBlockConfiguration")
private PublicAccessBlockConfiguration publicAccessBlockConfiguration;

@JsonProperty("BucketName")
private String bucketName;

@JsonProperty("RegionalDomainName")
private String regionalDomainName;

@JsonProperty("OwnershipControls")
private OwnershipControls ownershipControls;

@JsonProperty("ObjectLockConfiguration")
private ObjectLockConfiguration objectLockConfiguration;

@JsonProperty("ObjectLockEnabled")
private Boolean objectLockEnabled;

@JsonProperty("LoggingConfiguration")
private LoggingConfiguration loggingConfiguration;

@JsonProperty("ReplicationConfiguration")
private ReplicationConfiguration replicationConfiguration;

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

@JsonProperty("DomainName")
private String domainName;

@JsonProperty("BucketEncryption")
private BucketEncryption bucketEncryption;

@JsonProperty("WebsiteURL")
private String websiteURL;

@JsonProperty("NotificationConfiguration")
private NotificationConfiguration notificationConfiguration;

@JsonProperty("LifecycleConfiguration")
private LifecycleConfiguration lifecycleConfiguration;
```

```
@JsonProperty("VersioningConfiguration")
private VersioningConfiguration versioningConfiguration;

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

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

@JsonProperty("CorsConfiguration")
private CorsConfiguration corsConfiguration;

@JsonProperty("Id")
private String id;

@JsonProperty("Arn")
private String arn;

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

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

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

Example BucketEncryption.java

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

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

Example ServerSideEncryptionRule.java

```
package com.mycompany.testing.mytesthook.model.aws.s3.bucket;  
  
import ...  
  
@Data  
@Builder  
@AllArgsConstructor  
@NoArgsConstructor  
@JsonAutoDetect(fieldVisibility = Visibility.ANY, getterVisibility = Visibility.NONE,  
    setterVisibility = Visibility.NONE)  
public class ServerSideEncryptionRule {  
    @JsonProperty("BucketKeyEnabled")  
    private Boolean bucketKeyEnabled;  
  
    @JsonProperty("ServerSideEncryptionByDefault")  
    private ServerSideEncryptionByDefault serverSideEncryptionByDefault;  
  
}
```

Example ServerSideEncryptionByDefault.java

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

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

    @JsonProperty("KMSMasterKeyID")
    private String kMSMasterKeyID;

}
```

Con il file POJOs generato, ora puoi scrivere i gestori che implementano effettivamente la funzionalità di Hook. Per questo esempio, implementa il punto di preUpdate invocazione preCreate and per i gestori.

Fase 4: Implementazione dei gestori Hook

Argomenti

- [Codifica del client builder API](#)
- [Codifica del creatore della richiesta API](#)
- [Implementazione del codice di supporto](#)
- [Implementazione del gestore di base](#)
- [Implementazione del preCreate gestore](#)
- [Codifica del gestore preCreate](#)
- [Aggiornamento del preCreate test](#)
- [Implementazione del preUpdate gestore](#)
- [Codifica del gestore preUpdate](#)
- [Aggiornamento del preUpdate test](#)
- [Implementazione del preDelete gestore](#)
- [Codifica del gestore preDelete](#)
- [Aggiornamento del preDelete gestore](#)

Codifica del client builder API

1. Nel tuo IDE, apri il `ClientBuilder.java` file, che si trova nella `src/main/java/com/mycompany/testing/mytesthook` cartella.
2. Sostituisci l'intero contenuto del `ClientBuilder.java` file con il codice seguente.

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();
    }
}
```

Codifica del creatore della richiesta API

1. Nel tuo IDE, apri il `Translator.java` file, che si trova nella `src/main/java/com/mycompany/testing/mytesthook` cartella.
2. Sostituisci l'intero contenuto del `Translator.java` file con il codice seguente.

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();
    }
}
```

}

Implementazione del codice di supporto

1. Nel tuo IDE, apri il `AbstractTestBase.java` file, che si trova nella `src/main/java/com/mycompany/testing/mytesthook` cartella.
2. Sostituisci l'intero contenuto del `AbstractTestBase.java` file con il codice seguente.

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
        )
    );
}
```

Implementazione del gestore di base

1. Nel tuo IDE, apri il `BaseHookHandlerStd.java` file, che si trova nella `src/main/java/com/mycompany/testing/mytesthook` cartella.
2. Sostituisci l'intero contenuto del `BaseHookHandlerStd.java` file con il codice seguente.

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);
        }
    }
}
```

Implementazione del **preCreate** gestore

Il **preCreate** gestore verifica le impostazioni di crittografia sul lato server per una risorsa or. AWS::S3::Bucket AWS::SQS::Queue

- Per una AWS::S3::Bucket risorsa, l'Hook passerà solo se è vero quanto segue:
 - La crittografia del bucket Amazon S3 è impostata.
 - La chiave del bucket Amazon S3 è abilitata per il bucket.
 - L'algoritmo di crittografia impostato per il bucket Amazon S3 è l'algoritmo corretto richiesto.
 - L'ID della AWS Key Management Service chiave è impostato.

- Per una AWS::SQS::Queue risorsa, l'Hook passerà solo se è vero quanto segue:
 - L'ID della AWS Key Management Service chiave è impostato.

Codifica del gestore **preCreate**

1. Nel tuo IDE, apri il `PreCreateHookHandler.java` file, che si trova nella `src/main/java/software/mycompany/testing/mytesthook` cartella.
2. Sostituisci l'intero contenuto del `PreCreateHookHandler.java` file con il codice seguente.

```
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();
}
```

```
    }  
}
```

Aggiornamento del **preCreate** test

1. Nel tuo IDE, apri il `PreCreateHandlerTest.java` file, che si trova nella `src/test/java/software/mycompany/testing/mytesthook` cartella.
2. Sostituisci l'intero contenuto del `PreCreateHandlerTest.java` file con il seguente codice.

```
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);
        assertEquals(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();
    }
}
```

Implementazione del **preUpdate** gestore

preUpdate implementa un gestore, che si avvia prima delle operazioni di aggiornamento per tutte le destinazioni specificate nel gestore. Il **preUpdate** gestore esegue le seguenti operazioni:

- Per una AWS::S3::Bucket risorsa, l'Hook passerà solo se è vero quanto segue:
 - L'algoritmo di crittografia dei bucket per un bucket Amazon S3 non è stato modificato.

Codifica del gestore **preUpdate**

1. Nel tuo IDE, apri il `PreUpdateHookHandler.java` file, che si trova nella `src/main/java/software/mycompany/testing/mytesthook` cartella.
2. Sostituisci l'intero contenuto del `PreUpdateHookHandler.java` file con il codice seguente.

```
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();
    }
}

```

Aggiornamento del **preUpdate** test

1. Nel tuo IDE, apri il `PreUpdateHandlerTest.java` file nella `src/main/java/com/mycompany/testing/mytesthook` cartella.
2. Sostituisci l'intero contenuto del `PreUpdateHandlerTest.java` file con il codice seguente.

```

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();
    }
}
```

Implementazione del **preDelete** gestore

preDelete implementa un gestore, che si avvia prima delle operazioni di eliminazione per tutte le destinazioni specificate nel gestore. Il **preDelete** gestore esegue le seguenti operazioni:

- Per una AWS::S3::Bucket risorsa, l'Hook passerà solo se è vero quanto segue:
 - Verifica che le risorse minime richieste per i reclami esistano nell'account dopo l'eliminazione della risorsa.
 - La quantità minima di risorse richieste per i reclami è impostata nella configurazione del tipo di Hook.

Codifica del gestore **preDelete**

1. Nel tuo IDE, apri il `PreDeleteHookHandler.java` file nella cartella `src/main/java/com/mycompany/testing/mytesthook`.
2. Sostituisci l'intero contenuto del `PreDeleteHookHandler.java` file con il codice seguente.

```
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);
    }
}
}
}

```

Aggiornamento del **preDelete** gestore

1. Nel tuo IDE, apri il `PreDeleteHookHandler.java` file nella `src/main/java/com/mycompany/testing/mytesthook` cartella.
2. Sostituisci l'intero contenuto del `PreDeleteHookHandler.java` file con il codice seguente.

```

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 AbstractTestCase {

    @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());
}

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();
    .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();
}
}
```

Modellazione di CloudFormation Hooks personalizzati usando Python

La modellazione di CloudFormation Hooks personalizzati implica la creazione di uno schema che definisce l'Hook, le sue proprietà e i relativi attributi. Questo tutorial ti guida attraverso la modellazione di Hooks personalizzati usando Python.

Fase 1: Generare il pacchetto del progetto Hook

Genera il tuo pacchetto di progetto Hook. CloudFormation CLI crea funzioni di gestione vuote che corrispondono a specifiche azioni Hook nel ciclo di vita di destinazione, come definito nella specifica Hook.

```
cfn generate
```

Questo comando restituisce il seguente output.

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

Note

Assicurati che i runtime Lambda evitino l'uso up-to-date di una versione obsoleta. Per ulteriori informazioni, consulta [Aggiornamento dei runtime Lambda per i tipi di risorse](#) e gli hook.

Fase 2: Aggiungere i gestori Hook

Aggiungi il tuo codice di runtime del gestore Hook ai gestori che scegli di implementare. Ad esempio, puoi aggiungere il seguente codice per la registrazione.

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

CloudFormation CLI genera il `src/models.py` file da [Schema di configurazione](#)

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
```

Fase 3: Implementazione dei gestori Hook

Con le classi di dati Python generate, puoi scrivere i gestori che implementano effettivamente la funzionalità dell'Hook. In questo esempio, implemetterete i punti `preCreate`, `preUpdate`, e di `preDelete` invocazione per i gestori.

Argomenti

- [Implementa il gestore preCreate](#)

- [Implementa il preUpdate gestore](#)
- [Implementa il gestore preDelete](#)
- [Implementa un gestore Hook](#)

Implementa il gestore preCreate

Il `preCreate` gestore verifica le impostazioni di crittografia sul lato server per una risorsa or.

`AWS::S3::Bucket` `AWS::SQS::Queue`

- Per una `AWS::S3::Bucket` risorsa, l'Hook passerà solo se è vero quanto segue.
 - La crittografia del bucket Amazon S3 è impostata.
 - La chiave del bucket Amazon S3 è abilitata per il bucket.
 - L'algoritmo di crittografia impostato per il bucket Amazon S3 è l'algoritmo corretto richiesto.
 - L'ID della AWS Key Management Service chiave è impostato.
- Per una `AWS::SQS::Queue` risorsa, l'Hook passerà solo se è vero quanto segue.
 - L'ID della AWS Key Management Service chiave è impostato.

Implementa il preUpdate gestore

`preUpdate` implementa un gestore, che si avvia prima delle operazioni di aggiornamento per tutte le destinazioni specificate nel gestore. Il `preUpdate` gestore esegue le seguenti operazioni:

- Per una `AWS::S3::Bucket` risorsa, l'Hook passerà solo se è vero quanto segue:
 - L'algoritmo di crittografia del bucket per un bucket Amazon S3 non è stato modificato.

Implementa il gestore preDelete

`preDelete` implementa un gestore, che si avvia prima delle operazioni di eliminazione per tutte le destinazioni specificate nel gestore. Il `preDelete` gestore esegue le seguenti operazioni:

- Per una `AWS::S3::Bucket` risorsa, l'Hook passerà solo se è vero quanto segue:
 - Verifica che le risorse conformi minime richieste esistano nell'account dopo l'eliminazione della risorsa.
 - La quantità minima di risorse conformi richiesta è impostata nella configurazione di Hook.

Implementa un gestore Hook

1. Nel tuo IDE, apri il `handlers.py` file, che si trova nella `src` cartella.
2. Sostituisci l'intero contenuto del `handlers.py` file con il codice seguente.

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: {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}")
```

Continua fino all'argomento successivo [Registrazione di un Hook personalizzato con CloudFormation](#).

Registrazione di un Hook personalizzato con CloudFormation

Dopo aver creato un Hook personalizzato, devi registrarlo per CloudFormation poterlo utilizzare. In questa sezione, imparerai a impacchettare e registrare il tuo Hook per utilizzarlo nel tuo Account AWS.

Package a Hook (Java)

Se hai sviluppato il tuo Hook con Java, usa Maven per impacchettarlo.

Nella directory del tuo progetto Hook, esegui il seguente comando per creare il tuo Hook, esegui test unitari e impacchetta il progetto come JAR file da utilizzare per inviare il tuo Hook al CloudFormation registro.

```
mvn clean package
```

Registra un Hook personalizzato

Per registrare un Hook

1. (Facoltativo) Configura il tuo Regione AWS nome predefinito inus-west-2, inviando il [configure](#)operazione.

```
$ 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. (Facoltativo) Il comando seguente crea e impacchetta il progetto Hook senza registrarlo.

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

3. Registra il tuo Hook utilizzando il CloudFormation CLI [submit](#)operazione.

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

Questo comando restituisce il comando seguente.

```
{'ProgressStatus': 'COMPLETE'}
```

Risultati: Hai registrato con successo il tuo Hook.

Gli Hooks di verifica sono accessibili nel tuo account

Verifica che il tuo Hook sia disponibile nella tua Account AWS e nelle regioni a cui lo hai inviato.

1. Per verificare il tuo Hook, usa il [list-types](#) comando per elencare l'Hook appena registrato e restituirne una descrizione riassuntiva.

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

Il comando restituisce il seguente risultato e ti mostrerà anche gli Hook disponibili pubblicamente che puoi attivare nelle tue Account AWS regioni.

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

2. Recupera l'list-typeoutput TypeArn del tuo Hook e salvalo.

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

Per informazioni su come pubblicare Hooks per uso pubblico, consulta. [Ganci di pubblicazione per uso pubblico](#)

Configura gli Hooks

Dopo aver sviluppato e registrato il tuo Hook, puoi configurare il tuo Hook nel tuo Account AWS pubblicandolo nel registro.

- Per configurare un Hook nel tuo account, usa il [SetTypeConfiguration](#) operazione. Questa operazione abilita le proprietà dell'Hook definite nella properties sezione dello schema di Hook. Nell'esempio seguente, la minBuckets proprietà è impostata su 1 nella configurazione.

Note

Abilitando Hooks nel tuo account, autorizzi un Hook a utilizzare le autorizzazioni definite dal tuo. Account AWS CloudFormation rimuove le autorizzazioni non richieste prima di passare le autorizzazioni a Hook. CloudFormation consiglia ai clienti o agli utenti di Hook di rivedere le autorizzazioni di Hook e di essere consapevoli delle autorizzazioni a cui sono consentiti gli Hook prima di abilitare gli Hook nel proprio account.

Specificate i dati di configurazione per l'estensione Hook registrata nello stesso account e.

Regione AWS

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

Important

Per consentire a Hook di controllare in modo proattivo la configurazione dello stack, è necessario impostare l'HookInvocationStatus opzione ENABLED nella HookConfiguration sezione, dopo che l'Hook è stato registrato e attivato nel proprio account.

Accesso nei gestori AWS APIs

Se il tuo Hooks utilizza un AWS API in uno dei suoi gestori, CFN - crea CLI automaticamente un IAM modello di ruolo di esecuzione,. hook-role.yaml Il hook-role.yaml modello si basa sui permessi specificati per ogni gestore nella sezione dedicata al gestore dello schema Hook. Se il --role-arn flag non viene utilizzato durante il [generate](#) operazione, il ruolo in questo stack verrà assegnato e utilizzato come ruolo di esecuzione dell'Hook.

Per ulteriori informazioni, vedere [Accesso AWS APIs da un tipo di risorsa](#).

modello hook-role.yaml

Note

Se scegli di creare il tuo ruolo di esecuzione, ti consigliamo vivamente di mettere in pratica il principio del privilegio minimo, autorizzando solo l'inserimento nell'elenco e. `hooks.cloudformation.amazonaws.com`
`resources.cloudformation.amazonaws.com`

Il modello seguente utilizza IAM le autorizzazioni Amazon S3 e AmazonSQS.

```
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'
    - 'sns:GetQueueAttributes'
    - 'sns:GetQueueUrl'
    - 'sns>ListQueues'
  Resource: '*'

Outputs:
  ExecutionRoleArn:
    Value: !GetAtt
      - ExecutionRole
      - Arn
```

Testare un Hook personalizzato nel tuo Account AWS

Ora che hai codificato le funzioni del gestore che corrispondono a un punto di chiamata, è il momento di testare il tuo Hook personalizzato su una pila CloudFormation

La modalità di errore Hook è impostata su FAIL se il CloudFormation modello non ha fornito a un bucket S3 quanto segue:

- La crittografia del bucket Amazon S3 è impostata.
- La chiave del bucket Amazon S3 è abilitata per il bucket.
- L'algoritmo di crittografia impostato per il bucket Amazon S3 è l'algoritmo corretto richiesto.
- L'ID della AWS Key Management Service chiave è impostato.

Nell'esempio seguente, create un modello chiamato `my-failed-bucket-stack.yml` con un nome di stack `my-hook-stack` che non riporta la configurazione dello stack e si interrompe prima del rifornimento delle risorse.

Testare gli Hooks mediante il provisioning di uno stack

Esempio 1: effettuare il provisioning di uno stack

Esegui il provisioning di uno stack non conforme

1. Crea un modello che specifichi un bucket S3. Ad esempio, `my-failed-bucket-stack.yml`.

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

2. Crea uno stack e specifica il tuo modello in (). AWS Command Line Interface AWS CLI

Nell'esempio seguente, specificate il nome dello stack as `my-hook-stack` e il nome del modello come `my-failed-bucket-stack.yml`

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

3. (Facoltativo) Visualizza l'avanzamento dello stack specificando il nome dello stack. Nell'esempio seguente, specificate il nome dello stack. `my-hook-stack`

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

Utilizzate l'`describe-stack-events` operazione per vedere l'errore dell'Hook durante la creazione del bucket. Di seguito è riportato un esempio di output del comando.

```
{
  "StackEvents": [
    ...
    {
      "StackId": "arn:aws: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"
```

```
 },  
 ...  
 ]  
 }
```

Risultati: l'invocazione di Hook non ha consentito la configurazione dello stack e ha interrotto il provisioning della risorsa.

Usa un CloudFormation modello per passare la convalida di Hook

1. Per creare uno stack e superare la convalida Hook, aggiorna il modello in modo che la risorsa utilizzi un bucket S3 crittografato. Questo esempio utilizza il modello `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

Gli hook non verranno richiamati per le risorse ignorate.

2. Crea una pila e specifica il tuo modello. In questo esempio, il nome dello stack è `my-encrypted-bucket-stack`

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

3. (Facoltativo) Visualizza l'avanzamento dello stack specificando il nome dello stack.

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

Utilizzate il `describe-stack-events` comando per visualizzare la risposta. Di seguito è riportato un comando `describe-stack-events` di esempio.

```
{  
  "StackEvents": [  
    ...  
    {  
      "StackId": "arn:aws: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": "ENCRIPTION_KEY_ARN"}}], "ClientRequestToken": "Console-CreateStack-39df35ac-ca00-b7f6-5661-4e917478d075"}, {"StackId": "arn:aws: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\": \"ENCRIPTION_KEY_ARN\"}}]}}, "ClientRequestToken": "Console-CreateStack-39df35ac-ca00-b7f6-5661-4e917478d075"}, {"StackId": "arn:aws: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"}, {"...}], }
```

Risultati: lo stack è stato creato CloudFormation con successo. La logica di Hook ha verificato che la AWS::S3::Bucket risorsa contenesse la crittografia lato server prima di effettuare il provisioning della risorsa.

Esempio 2: effettuare il provisioning di uno stack

Esegui il provisioning di uno stack non conforme

1. Crea un modello che specifichi un bucket S3. Ad esempio, `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. Crea uno stack e specifica il tuo modello in AWS CLI Nell'esempio seguente, specificate il nome dello stack as `my-hook-stack` e il nome del modello come `aes256-bucket.yml`

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

3. (Facoltativo) Visualizza l'avanzamento dello stack specificando il nome dello stack. Nell'esempio seguente, specificate il nome dello stack. `my-hook-stack`

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

Utilizzate l'operazione `describe-stack-events` per vedere l'errore dell'Hook durante la creazione del bucket. Di seguito è riportato un esempio di output del comando.

```
{  
  "StackEvents": [  
    ...  
    {  
      "StackId": "arn:aws: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"  
    },  
    ...  
  ]  
}
```

Risultati: l'invocazione di Hook non ha consentito la configurazione dello stack e ha interrotto il provisioning della risorsa. Lo stack non è riuscito a causa della crittografia del bucket S3 configurata in modo errato. La configurazione del tipo Hook richiede l'utilizzo di questo `aws:kms` bucket. AES256

Usa un CloudFormation modello per passare la convalida di Hook

1. Per creare uno stack e superare la convalida Hook, aggiorna il modello in modo che la risorsa utilizzi un bucket S3 crittografato. Questo esempio utilizza il modello `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

Gli hook non verranno richiamati per le risorse ignorate.

2. Crea una pila e specifica il tuo modello. In questo esempio, il nome dello stack è. `my-encrypted-bucket-stack`

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

3. (Facoltativo) Visualizza l'avanzamento dello stack specificando il nome dello stack.

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

Utilizzate il `describe-stack-events` comando per visualizzare la risposta. Di seguito è riportato un comando `describe-stack-events` di esempio.

```
{
  "StackEvents": [
    ...
    {
      "StackId": "arn:aws: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: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: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"
    },
    ...
}
```

Risultati: lo stack è stato creato CloudFormation con successo. La logica di Hook ha verificato che la AWS::S3::Bucket risorsa contenesse la crittografia lato server prima di effettuare il provisioning della risorsa.

Aggiornamento di un Hook personalizzato

L'aggiornamento di un Hook personalizzato consente di rendere disponibili le revisioni dell'Hook nel CloudFormation registro.

Per aggiornare un Hook personalizzato, invia le tue revisioni al CloudFormation registro tramite CloudFormation CLI [submit](#) operazione.

```
$ cfn submit
```

Per specificare la versione predefinita di Hook nel tuo account, usa il [set-type-default-version](#) comando e specifica il tipo, il nome del tipo e l'ID della versione.

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

Per recuperare informazioni sulle versioni di un Hook, usa [list-type-versions](#).

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

Annnullamento della registrazione di un Hook personalizzato dal registro CloudFormation

L'annnullamento della registrazione di un Hook personalizzato contrassegna l'estensione o la versione dell'estensione come DEPRECATED nel CloudFormation registro, il che la rimuove dall'uso attivo. Una volta obsoleto, l'Hook personalizzato non può essere utilizzato in un'operazione. CloudFormation

Note

Prima di annullare la registrazione dell'Hook, è necessario annullare singolarmente tutte le versioni attive precedenti di quell'estensione. Per ulteriori informazioni, consulta [DeregisterType](#).

Per annullare la registrazione di un Hook, usa il [deregister-type](#) operazione e specifica il tuo Hook ARN

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

```
--arn HOOK_TYPE_ARN
```

Questo comando non produce un output.

Ganci di pubblicazione per uso pubblico

Per sviluppare un Hook pubblico di terze parti, sviluppa il tuo Hook come estensione privata. Quindi, in ognuna delle aree Regione AWS in cui desideri rendere l'estensione disponibile al pubblico:

1. Registra il tuo Hook come estensione privata nel CloudFormation registro.
2. Testa il tuo Hook per assicurarti che soddisfi tutti i requisiti necessari per essere pubblicato nel CloudFormation registro.
3. Pubblica il tuo Hook nel CloudFormation registro.

Note

Prima di pubblicare qualsiasi estensione in una determinata regione, devi prima registrarti come editore di estensioni in quella regione. Per eseguire questa operazione in più aree contemporaneamente, consulta [Pubblicazione di estensioni in più aree geografiche StackSets](#) nella Guida per l' CloudFormation CLIutente.

Dopo aver sviluppato e registrato il tuo Hook, puoi renderlo disponibile pubblicamente CloudFormation agli utenti generici pubblicandolo nel CloudFormation registro, come estensione pubblica di terze parti.

Gli Hook pubblici di terze parti consentono di offrire agli CloudFormation utenti la possibilità di ispezionare in modo proattivo la configurazione delle AWS risorse prima del provisioning. Come per gli Hook privati, gli Hook pubblici vengono trattati allo stesso modo di qualsiasi Hook pubblicato da within. AWS CloudFormation

Gli hook pubblicati nel registro sono visibili da tutti CloudFormation gli utenti nella cartella Regioni AWS in cui sono pubblicati. Gli utenti possono quindi attivare l'estensione nel proprio account, il che la rende disponibile per l'uso nei propri modelli. Per ulteriori informazioni, consulta [Utilizzare estensioni pubbliche di terze parti dal CloudFormation registro](#) nella Guida per l'CloudFormation utente.

Test di un Hook personalizzato per uso pubblico

Per pubblicare il tuo Hook personalizzato registrato, deve superare tutti i requisiti di test definiti per esso. Di seguito è riportato un elenco di requisiti necessari prima di pubblicare l'Hook personalizzato come estensione di terze parti.

Ogni gestore e target viene testato due volte. Una volta per `SUCCESS` e una volta per `FAILED`.

- Per caso di `SUCCESS` risposta:
 - Lo stato deve essere `SUCCESS`.
 - Non deve restituire un codice di errore.
 - Il ritardo di callback deve essere impostato su 0 secondi, se specificato.
- Per il caso `FAILED` di risposta:
 - Lo stato deve essere `FAILED`.
 - Deve restituire un codice di errore.
 - Deve avere un messaggio in risposta.
 - Il ritardo di callback deve essere impostato su 0 secondi, se specificato.
- Per il caso `IN_PROGRESS` di risposta:
 - Non deve restituire un codice di errore.
 - Result il campo non deve essere impostato in risposta.

Specificazione dei dati di input da utilizzare nei test contrattuali

Per impostazione predefinita, CloudFormation esegue i test contrattuali utilizzando le proprietà di input generate dai modelli definiti nello schema Hook. Tuttavia, la maggior parte degli Hook è sufficientemente complessa che le proprietà di input per la precreazione o il preaggiornamento degli stack di provisioning richiedono una comprensione della risorsa da fornire. Per risolvere questo problema, è possibile specificare l'input che CloudFormation utilizza durante l'esecuzione dei test contrattuali.

CloudFormation offre due modi per specificare i dati di input da utilizzare durante l'esecuzione dei test contrattuali:

- Sostituisce il file

L'utilizzo di un `overrides` file fornisce un modo semplice per specificare i dati di input per determinate proprietà specifiche CloudFormation da utilizzare durante `preCreate` i test operativi. `preUpdate` `preDelete`

- File di input

Puoi anche utilizzare più `input` file per specificare i dati di input del test contrattuale se:

- Volete o dovete specificare dati di input diversi per le operazioni di creazione, aggiornamento ed eliminazione oppure dati non validi con cui eseguire il test.
- Desiderate specificare più set di dati di input diversi.

Specificare i dati di input utilizzando un file di override

Di seguito è riportato un esempio di dati di input di Amazon S3 Hook che utilizzano il `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": "anotherbucket",  
          "/BucketEncryption/ServerSideEncryptionConfiguration": [  
            {  
              "BucketKeyEnabled": true,  
              "ServerSideEncryptionByDefault": {  
                "KMSMasterKeyId": "KMS-KEY-ARN",  
                "SSEAlgorithm": "aws:kms"  
              }  
            }  
          ]  
        }  
      }  
    }  
  }  
}
```

```
        "/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"
        }
    }
}
}
```

Specificazione dei dati di input utilizzando i file di input

Utilizzate `input` i file per specificare diversi tipi di dati di input CloudFormation da utilizzare: `preCreate` input, `preUpdate` input e input non valido. Ogni tipo di dati è specificato in un file separato. È inoltre possibile specificare più set di dati di input per i test contrattuali.

Per specificare `input` i file CloudFormation da utilizzare nei test contrattuali, aggiungi una `inputs` cartella alla directory principale del tuo progetto Hooks. Quindi aggiungi i tuoi file di input.

Specificate il tipo di dati di input contenuti in un file utilizzando le seguenti convenzioni di denominazione, dove `n` è un numero intero:

- `inputs_n_pre_create.json`: utilizza file con `preCreate` gestori per specificare gli input per la creazione della risorsa.
- `inputs_n_pre_update.json`: Utilizza file con `preUpdate` gestori per specificare gli input per l'aggiornamento della risorsa.
- `inputs_n_pre_delete.json`: Usa file con `preDelete` gestori per specificare gli input per l'eliminazione della risorsa.
- `inputs_n_invalid.json`: Per specificare input non validi da testare.

Per specificare più set di dati di input per i test contrattuali, incrementa il numero intero nei nomi dei file per ordinare i set di dati di input. Ad esempio, il primo set di file di input dovrebbe essere denominato `inputs_1_pre_create.json`, `inputs_1_pre_update.json`, `e.inputs_1_pre_invalid.json`. Il set successivo si chiamerebbe `inputs_2_pre_create.json`, `inputs_2_pre_update.json`, `inputs_2_pre_invalid.json`, e così via.

Ogni file di input è un JSON file contenente solo le proprietà delle risorse da utilizzare nei test.

Di seguito è riportata una directory di esempio `inputs` per Amazon S3 specificare i dati di input utilizzando i file di input.

`inputs_1_pre_create.json`

Di seguito è riportato un esempio di test del `inputs_1_pre_create.json` contratto.

```
{  
  "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

Quello che segue è un esempio di test del `inputs_1_pre_update.json` contratto.

```
{  
  "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

Quello che segue è un esempio di test del `inputs_1_invalid.json` contratto.

```
{  
  "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

Quello che segue è un esempio di test del `inputs_1_invalid_pre_update.json` contratto.

```

{
    "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"
}
}
```

Per ulteriori informazioni, consulta [Pubblicazione delle estensioni per renderle disponibili per l'uso pubblico](#) nella Guida per l' CloudFormation CLIutente.

Riferimento alla sintassi dello schema per CloudFormation Hooks

Questa sezione descrive la sintassi dello schema utilizzato per sviluppare CloudFormation Hooks.

Un Hook include una specifica Hook rappresentata da uno schema JSON e da gestori Hook. Il primo passaggio per creare un Hook personalizzato consiste nella modellazione di uno schema che definisca l'Hook, le sue proprietà e i relativi attributi. Quando inizializzi un progetto Hook personalizzato utilizzando il comando CloudFormation [init](#)CLI, viene creato automaticamente un file di schema Hook. Utilizzate questo file di schema come punto di partenza per definire la forma e la semantica del vostro Hook personalizzato.

Sintassi dello schema

Lo schema seguente è la struttura di un Hook.

```
{
  "typeName": "string",
  "description": "string",
  "sourceUrl": "string",
  "documentationUrl": "string",
  "definitions": {
    "definitionName": {
      ...
    }
  }
}
```

```
        },
    },
    "typeConfiguration": {
        "properties": {
            "propertyName": {
                "description": "string",
                "type": "string",
                ...
            },
        },
    },
    "required": [
        "propertyName"
        ...
    ],
    "additionalProperties": false
},
"handlers": {
    "preCreate": {
        "targetNames": [
        ],
        "permissions": [
        ]
    },
    "preUpdate": {
        "targetNames": [
        ],
        "permissions": [
        ]
    },
    "preDelete": {
        "targetNames": [
        ],
        "permissions": [
        ]
    }
},
"additionalProperties": false
}
```

typeName

Il nome univoco del tuo Hook. Specifica uno spazio dei nomi in tre parti per il tuo Hook, con uno schema consigliato di Organization::Service::Hook

Note

I seguenti namespace dell'organizzazione sono riservati e non possono essere utilizzati nei nomi dei tipi di Hook:

- Alexa
- AMZN
- Amazon
- ASK
- AWS
- Custom
- Dev

Campo obbligatorio: sì

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

Minimum: 10

Maximum: 196

`description`

Una breve descrizione dell'Hook visualizzato nella console CloudFormation

Campo obbligatorio: sì

`sourceUrl`

L'URL del codice sorgente dell'Hook, se pubblico.

Required: No

Maximum: 4096

`documentationUrl`

L'URL di una pagina che fornisce la documentazione dettagliata per l'Hook.

Campo obbligatorio: sì

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

Maximum: 4096

Note

Sebbene lo schema Hook debba includere descrizioni delle proprietà complete e accurate, è possibile utilizzare la `documentationURL` proprietà per fornire agli utenti maggiori dettagli, inclusi esempi, casi d'uso e altre informazioni dettagliate.

definitions

Utilizzate il `definitions` blocco per fornire schemi di proprietà Hook condivisi.

È consigliabile utilizzare la `definitions` sezione per definire gli elementi dello schema che possono essere utilizzati in più punti dello schema di tipo Hook. È quindi possibile utilizzare un puntatore JSON per fare riferimento a quell'elemento nei punti appropriati dello schema di tipo Hook.

Required: No

typeConfiguration

La definizione dei dati di configurazione di un Hook.

Campo obbligatorio: sì

properties

Le proprietà dell'Hook. Tutte le proprietà di un Hook devono essere espresse nello schema. Allinea le proprietà dello schema Hook con le proprietà di configurazione del tipo Hook.

Note

Le proprietà annidate non sono consentite. Definite invece tutte le proprietà annidate nell'`definitionselemento` e utilizzate un `$ref` puntatore per farvi riferimento nella proprietà desiderata.

Attualmente sono supportate le seguenti proprietà:

- `default`— Il valore predefinito della proprietà.
- `description`— Una descrizione della proprietà.

- **pattern**— Un modello regex utilizzato per convalidare l'input.
- **type**— Il tipo di proprietà accettato.

additionalProperties

`additionalProperties` deve essere impostato su `false`. Tutte le proprietà di un Hook devono essere espresse nello schema: non sono consentiti input arbitrari.

Campo obbligatorio: sì

Valori validi: `false`

handlers

I gestori specificano le operazioni che possono avviare l'Hook definito nello schema, come i punti di invocazione Hook. Ad esempio, un `preUpdate` gestore viene richiamato prima delle operazioni di aggiornamento per tutte le destinazioni specificate nel gestore.

Valori validi: `preCreate` | `preUpdate` | `preDelete`

 Note

È necessario specificare almeno un valore per il gestore.

 Important

Le operazioni di stack che determinano lo stato di `UpdateCleanup` non richiamano un Hook. Ad esempio, durante i due scenari seguenti, il `preDelete` gestore di Hook non viene richiamato:

- lo stack viene aggiornato dopo aver rimosso una risorsa dal modello.
- viene eliminata una risorsa con il tipo di aggiornamento [sostitutivo](#).

targetNames

Una matrice di stringhe di nomi di tipo a cui Hook si rivolge. Ad esempio, se un `preCreate` gestore ha una `AWS::S3::Bucket` destinazione, l'Hook viene eseguito per i bucket Amazon S3 durante la fase di preprovisioning.

- `TargetName`

Specificare almeno un nome di destinazione per ogni gestore implementato.

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

Minimum: 1

Campo obbligatorio: sì

 Warning

I riferimenti dinamici a SSM SecureString e Secrets Manager non vengono risolti prima di essere passati a Hooks.

permissions

Un array di stringhe che specifica le AWS autorizzazioni necessarie per richiamare il gestore.

Campo obbligatorio: sì

additionalProperties

additionalProperties deve essere impostato su `false`. Tutte le proprietà di un Hook devono essere espresse nello schema: non sono consentiti input arbitrari.

Campo obbligatorio: sì

Valori validi: `false`

Esempi di schemi Hooks

Esempio 1

Le procedure dettagliate per Java e Python utilizzano il seguente esempio di codice. Di seguito è riportato un esempio di struttura per un Hook chiamato. `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",
    "sns>ListQueues",
    "sns>GetQueueAttributes",
    "sns>GetQueueUrl"
  ]
},
"additionalProperties":false
}
```

Esempio 2

L'esempio seguente è uno schema che utilizza STACK and CHANGE_SET for per targetNames indirizzare un modello di stack e un'operazione di set di modifiche.

```
{
  "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
  }
}
```

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

Disabilita e abilita gli CloudFormation Hooks

Questo argomento descrive come disattivare e riattivare un Hook per impedirne temporaneamente l'attivazione nel tuo account. La disattivazione degli Hooks può essere utile quando è necessario esaminare un problema senza interferenze da parte degli Hooks.

Disattiva e abilita un Hook nel tuo account (console)

Per disabilitare un Hook nel tuo account

1. Accedi Console di gestione AWS e apri la CloudFormation console all'indirizzo <https://console.aws.amazon.com/cloudformation>.
2. Nella barra di navigazione nella parte superiore dello schermo, scegli Regione AWS dove si trova l'Hook.
3. Dal pannello di navigazione, scegli Hooks.
4. Scegli il nome dell'Hook che desideri disabilitare.
5. Nella pagina dei dettagli dell'Hook, a destra del nome dell'Hook, scegli il pulsante Disabilita.
6. Quando viene richiesta la conferma, scegli Disabilita Hook.

Per riattivare un Hook precedentemente disabilitato

1. Accedi Console di gestione AWS e apri la CloudFormation console all'indirizzo <https://console.aws.amazon.com/cloudformation>.
2. Nella barra di navigazione nella parte superiore dello schermo, scegli Regione AWS dove si trova l'Hook.
3. Dal pannello di navigazione, scegli Hooks.
4. Scegli il nome dell'Hook che desideri abilitare.
5. Nella pagina dei dettagli dell'Hook, a destra del nome dell'Hook, scegli il pulsante Abilita.
6. Quando viene richiesta la conferma, scegli Enable Hook.

Disattiva e abilita un Hook nel tuo account ()AWS CLI

⚠ Important

I AWS CLI comandi per disabilitare e abilitare gli Hooks sostituiscono l'intera configurazione di Hook con i valori specificati nell'--configurationopzione. Per evitare modifiche involontarie, è necessario includere tutte le impostazioni esistenti che si desidera mantenere durante l'esecuzione di questi comandi. Per visualizzare i dati di configurazione correnti, utilizzare [describe-type](#) comando.

Per disabilitare un Hook

Usa quanto segue [set-type-configuration](#) comando e specifica HookInvocationStatus come DISABLED disabilitare l'Hook. Sostituisci i segnaposto con i tuoi valori specifici.

```
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
```

Per riattivare un Hook precedentemente disabilitato

Usa quanto segue [set-type-configuration](#) comando e specifica HookInvocationStatus come ENABLED riattivare l'Hook. Sostituisci i segnaposto con i tuoi valori specifici.

```
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
```

Per ulteriori informazioni, consulta [Riferimento alla sintassi dello schema di configurazione Hook](#).

Visualizza i risultati di chiamata per Hooks CloudFormation

Questo argomento descrive come visualizzare i risultati delle chiamate per Hooks. CloudFormation La visualizzazione dei risultati delle chiamate può aiutarti a capire come gli Hooks valutano le tue risorse e a risolvere eventuali problemi rilevati quando gli Hooks verificano una risorsa.

Le chiamate sono istanze specifiche in cui la logica di convalida (che si tratti di un controllo AWS Control Tower proattivo, di una regola Guard o di una funzione Lambda) viene eseguita durante il ciclo di vita di una risorsa.

Visualizza i risultati della chiamata nella console

È possibile visualizzare i risultati delle chiamate nella console in tre modi: tramite la pagina di riepilogo delle chiamate, tramite le cronologie delle invocazioni per singoli Hook o tramite singoli eventi dello stack per chiamate specifiche dello stack.

Visualizza i risultati per tutti gli Hooks

La pagina di riepilogo delle chiamate fornisce una visualizzazione completa di tutte le chiamate Hook relative al tuo account e alla tua regione negli ultimi 90 giorni.

Per visualizzare i risultati di tutti gli Hooks

1. Accedi Console di gestione AWS e apri la CloudFormation console all'indirizzo <https://console.aws.amazon.com/cloudformation>.
2. Nella barra di navigazione nella parte superiore dello schermo, scegli Regione AWS dove vuoi visualizzare le invocazioni di Hook.
3. Dal pannello di navigazione, scegli Riepilogo delle chiamate.
4. La pagina mostra un elenco di tutte le invocazioni di Hook degli ultimi 90 giorni, tra cui:
 - ID di chiamata
 - Gancio
 - Target
 - Modalità (WarnFail)
 - Risultato (Warning,Pass,Failed,In progress)
 - Tempo di invocazione

- Messaggio del risultato
5. Puoi filtrare l'elenco utilizzando la barra di ricerca nella parte superiore della tabella per trovare chiamate specifiche.
 6. Seleziona una chiamata specifica per visualizzare ulteriori dettagli aggiuntivi sul risultato della chiamata, incluse le linee guida per la correzione delle chiamate Hook non riuscite.

Visualizza la cronologia delle chiamate per i singoli Hook

Puoi anche visualizzare i risultati delle chiamate tramite le cronologie delle chiamate per i singoli Hook.

Per visualizzare le invocazioni Hook per un Hook specifico

1. [Accedi Console di gestione AWS e apri la CloudFormation console all'indirizzo https://console.aws.amazon.com /cloudformazione.](https://console.aws.amazon.com/cloudformation/)
2. Nella barra di navigazione nella parte superiore dello schermo, scegli Regione AWS dove vuoi visualizzare le invocazioni di Hook.
3. Dal pannello di navigazione, scegli Hooks.
4. Scegli l'Hook per il quale desideri visualizzare le invocazioni di Hook.
5. Seleziona una chiamata specifica per visualizzare ulteriori dettagli aggiuntivi sul risultato della chiamata, incluse le linee guida per la correzione delle chiamate Hook non riuscite.

Visualizza i risultati per le chiamate specifiche dello stack

Puoi anche visualizzare i risultati delle chiamate per uno stack specifico tramite la pagina Eventi dello stack.

Per visualizzare le invocazioni Hook per uno stack specifico

1. [Accedi Console di gestione AWS e apri la CloudFormation console all'indirizzo /cloudformazione.](https://console.aws.amazon.com/cloudformation/)
2. Nella barra di navigazione nella parte superiore dello schermo, scegli il Regione AWS luogo in cui è avvenuta l'operazione stack.
3. Scegli Stack nel riquadro di navigazione.
4. Seleziona lo stack per il quale desideri visualizzare le invocazioni di Hook.

5. Scegli la scheda Stack events.
 6. Nell'elenco degli eventi, cerca gli eventi con invocazioni Hook complete nella colonna Status reason.
 7. Per visualizzare dettagli specifici sulle invocazioni di Hook, consulta la colonna delle invocazioni di Hook e scegli il testo sottolineato per aprire una finestra pop-up con informazioni più dettagliate.

 Note

Per visualizzare le colonne nascoste, scegli l'icona a forma di ingranaggio nell'angolo in alto a destra della sezione per aprire la finestra modale Preferenze, aggiorna le impostazioni secondo necessità e scegli Conferma.

Visualizza i risultati della chiamata utilizzando il AWS CLI

Usa il [list-hook-results](#) comando per recuperare informazioni sulle invocazioni di Hook. Questo comando supporta le seguenti opzioni di filtro:

- Ottieni tutti i risultati della chiamata di Hook (nessun parametro richiesto)
 - Filtra per Hook ARN (uso `--type-arn`)
 - Filtra per Hook ARN e stato (uso `--type-arn` e) `--status`
 - Cerca obiettivi specifici (uso `--target-type` e `--target-id`)

Filtra i risultati per Hook ARN

Il comando seguente elenca tutti i risultati di invocazione di Hook per un Hook specifico.

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

Output di esempio:

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

Per una descrizione dei campi nella risposta, consulta l'AWS CloudFormation API [HookResultSummary](#) Reference.

Filtra i risultati per Hook, ARN e stato

Per filtrare in base a uno stato comune tra i risultati, specifica l'--statusopzione nel comando. I valori validi sono:

- HOOK_IN_PROGRESS: L'Hook è attualmente in esecuzione.
- HOOK_COMPLETE_SUCCEEDED: L'Hook è stato completato con successo.
- HOOK_COMPLETE_FAILED: La convalida dell'Hook è stata completata ma non è riuscita.
- HOOK_FAILED: L'Hook ha riscontrato un errore durante l'esecuzione.

```
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
```

Output di esempio:

```
{
```

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

Per una descrizione dei campi della risposta, consulta [HookResultSummary](#) nell'AWS CloudFormation API Reference.

Filtra i risultati per tipo e ID di destinazione

Il comando seguente elenca tutti i risultati di invocazione di Hook per una richiesta API Cloud Control specifica.

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

Output di esempio:

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

```
        "HookResultId": "4e7f4766-d8fe-44e5-8587-5b327a148abe",
        "TypeName": "MyOrg::Security::ComplianceHook",
        "TypeVersionId": "00000001",
        "FailureMode": "WARN",
        "HookStatusReason": "...",
        "InvocationPoint": "PRE_PROVISION",
        "Status": "HOOK_COMPLETE_FAILED"
    },
    ...
]
}
]
```

Per una descrizione dei campi nella risposta, consulta l'AWS CloudFormation API [HookResultSummary](#) Reference.

Ottieni risultati dettagliati per una chiamata specifica

Utilizza il [get-hook-result](#) comando per recuperare informazioni dettagliate su una specifica invocazione di Hook, incluse annotazioni con i risultati del controllo di conformità e linee guida per la correzione.

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

Output di esempio:

```
{
    "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",
```

```
"TargetType": "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"
}
]
```

Per una descrizione dei campi della risposta, [GetHookResult](#) consulta l'API Reference AWS CloudFormation

Riferimento alla sintassi dello schema di configurazione Hook

Questa sezione descrive la sintassi dello schema utilizzata per configurare gli Hooks. CloudFormation utilizza questo schema di configurazione in fase di esecuzione quando richiama un Hook in un.

Account AWS

Per consentire a Hook di ispezionare in modo proattivo la configurazione dello stack, imposta su `ENABLED` dopo che l'`HookInvocationStatus` è stato registrato e attivato nel tuo account.

Argomenti

- [Proprietà dello schema di configurazione Hook](#)
- [Esempi di configurazione Hook](#)
- [CloudFormation Filtri Hooks Stack Level](#)
- [CloudFormation Aggancia i filtri target](#)
- [Usare i caratteri jolly con i nomi degli obiettivi di Hook](#)

 Note

La quantità massima di dati che la configurazione di un Hook può memorizzare è di 300 KB. Questo si aggiunge a tutti i vincoli imposti al parametro di funzionamento della Configuration [SetTypeConfiguration](#) richiesta.

Proprietà dello schema di configurazione Hook

Lo schema seguente è la struttura di uno schema di configurazione Hook.

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

```
  },
  "Properties": {
    ...
  }
}
```

HookConfiguration

La configurazione Hook supporta l'attivazione o la disattivazione degli Hook a livello di stack, le modalità di errore e i valori delle proprietà Hook.

La configurazione Hook supporta le seguenti proprietà.

HookInvocationStatus

Specifica se l'Hook è ENABLED o DISABLED.

Valori validi: ENABLED | DISABLED

TargetOperations

Specifica l'elenco delle operazioni su cui viene eseguito l'Hook. Per ulteriori informazioni, consulta [Obiettivi Hook](#).

Valori validi: STACK | RESOURCE | CHANGE_SET | CLOUD_CONTROL

TargetStacks

Disponibile per la compatibilità con le versioni precedenti. Usa *HookInvocationStatus* invece.

Se la modalità è impostata su ALL, l'Hook si applica a tutti gli stack dell'account durante un'operazione su CREATEUPDATE, o su una DELETE risorsa.

Se la modalità è impostata su NONE, l'Hook non si applicherà agli stack del tuo account.

Valori validi: ALL | NONE

FailureMode

Questo campo indica al servizio come trattare gli errori di Hook.

- Se la modalità è impostata su e l'Hook fallisce, la configurazione di errore interrompe il provisioning delle risorse e ripristina lo stack. FAIL

- Se la modalità è impostata su WARN e l'Hook fallisce, la configurazione warn consente di continuare il provisioning con un messaggio di avviso.

Valori validi: FAIL | WARN

EncryptionConfiguration

Specifica le impostazioni di crittografia per i dati delle annotazioni Hook.

KmsKeyId

L'alias, l'alias ARN, l'ID della chiave o l'ARN della chiave di crittografia AWS KMS simmetrica utilizzata per crittografare i dati delle annotazioni Hook. Per ulteriori informazioni, consulta i documenti. [KeyId AWS KMS](#)

Prima di poter creare Hooks con AWS KMS chiavi gestite dal cliente, l'utente o il ruolo deve disporre AWS KMS delle autorizzazioni per e. [DescribeKey](#) [GenerateDataKey](#) Per ulteriori informazioni, consulta [AWS KMS policy chiave e autorizzazioni per crittografare i risultati di CloudFormation Hooks a riposo.](#)

Properties

Specifica le proprietà di runtime di Hook. Queste devono corrispondere alla forma delle proprietà supportate dallo schema Hooks.

Esempi di configurazione Hook

Per esempi di configurazione di Hooks da AWS CLI, consulta le seguenti sezioni:

- [Attiva un Hook basato su un controllo proattivo \(\)AWS CLI](#)
- [Attiva un Guard Hook \(AWS CLI\)](#)
- [Attiva un Lambda Hook \(\)AWS CLI](#)

CloudFormation Filtri Hooks Stack Level

Puoi aggiungere filtri a livello di stack ai tuoi CloudFormation Hooks per indirizzare stack specifici in base ai nomi e ai ruoli degli stack. Ciò è utile nei casi in cui si dispone di più stack con gli stessi tipi di risorse, ma l'Hook è destinato a stack specifici.

Questa sezione spiega come funzionano questi filtri e fornisce esempi da seguire.

La struttura di base di una configurazione Hook senza filtraggio a livello di stack è la seguente:

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

Per ulteriori informazioni sulla HookConfiguration sintassi, vedere. [Riferimento alla sintassi dello schema di configurazione Hook](#)

Per utilizzare i filtri a livello di stack, aggiungi una StackFilters chiave sotto. HookConfiguration

La StackFilters chiave ha un membro obbligatorio e due membri opzionali.

- FilteringCriteria(obbligatorio)
- StackNames (facoltativo)
- StackRoles (facoltativo)

Le StackRoles proprietà StackNames or sono opzionali. Tuttavia, è necessario specificare almeno una di tali proprietà.

Se crei un Hook destinato alle operazioni dell'[API Cloud Control](#), tutti i filtri a livello di stack verranno ignorati.

FilteringCriteria

FilteringCriteria è un parametro obbligatorio che specifica il comportamento di filtraggio. Può essere impostato su uno dei due valori. ANY

- ALL richiama l'Hook se tutti i filtri corrispondono.
- ANY richiama l'Hook se c'è un filtro corrispondente.

StackNames

Per specificare uno o più nomi di stack come filtri nella configurazione di Hooks, utilizza la seguente struttura JSON:

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

È necessario specificare una delle seguenti opzioni:

- Include: elenco dei nomi degli stack da includere. Solo gli stack specificati in questo elenco richiameranno l'Hook.
 - Tipo: matrice di stringhe
 - Numero massimo di elementi: 50
 - Numero minimo di articoli: 1
- Exclude: Elenco dei nomi degli stack da escludere. Tutti gli stack tranne quelli elencati qui invocheranno l'Hook.
 - Tipo: matrice di stringhe
 - Numero massimo di oggetti: 50
 - Numero minimo di articoli: 1

Ogni nome di stack negli `Exclude` array `Include` and deve rispettare i seguenti requisiti di modello e lunghezza:

- Modello: `^[a-zA-Z][-a-zA-Z0-9]*$`
- Lunghezza massima: 128

StackName supporta nomi di stack concreti e la corrispondenza completa dei caratteri jolly. Per vedere esempi di utilizzo dei caratteri jolly, consulta. [Usare i caratteri jolly con i nomi degli obiettivi di Hook](#)

StackRoles

Per specificare uno o più [ruoli IAM](#) come filtri nella configurazione di Hook, utilizza la seguente struttura JSON:

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

È necessario specificare una delle seguenti opzioni:

- `Include`: Elenco dei ruoli IAM ARNs a cui indirizzare gli stack associati a questi ruoli. Solo le operazioni di stack avviate da questi ruoli richiameranno l'Hook.
 - Tipo: matrice di stringhe
 - Numero massimo di articoli: 50
 - Numero minimo di articoli: 1
- `Exclude`: Elenco dei ruoli IAM ARNs per gli stack che desideri escludere. L'Hook verrà richiamato su tutti gli stack ad eccezione di quelli avviati dai ruoli specificati.
 - Tipo: matrice di stringhe
 - Numero massimo di articoli: 50
 - Numero minimo di articoli: 1

Ogni ruolo dello stack negli `Exclude` array `Include` and deve rispettare i seguenti requisiti di modello e lunghezza:

- Modello: `arn:.+:iam::[0-9]{12}:role/.+`
- Lunghezza massima: 256

`StackRoles` consente caratteri jolly nelle seguenti sezioni della [sintassi ARN](#):

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

Per vedere esempi di utilizzo dei caratteri jolly nelle sezioni sulla sintassi ARN, vedere. [Usare i caratteri jolly con i nomi degli obiettivi di Hook](#)

Include e Exclude

Ogni filtro (`StackNames` and `StackRoles`) ha una `Include` lista e una `Exclude` lista. Ad esempio, l'Hook viene richiamato solo sugli stack specificati nell'`Include` elenco. Se i nomi degli stack sono specificati solo nell'`Exclude` elenco, l'hook viene richiamato solo sugli stack che non sono presenti nell'elenco. Se `Exclude` vengono specificati entrambi `Include` e, l'Hook prende di mira ciò che è nell'`Include` elenco e non ciò che è nell'elenco. `Exclude`

Ad esempio, supponiamo di avere quattro pile: A, B, C e D.

- "Include": `["A", "B"]` L'Hook viene invocato su A e B.
- "Exclude": `["B"]` L'Hook viene invocato su A, C e D.
- "Include": `["A", "B", "C"]`, "Exclude": `["A", "D"]` L'Hook viene invocato su B e C.
- "Include": `["A", "B", "C"]`, "Exclude": `["A", "B", "C"]` L'Hook non viene invocato su nessuno stack.

Esempi di filtri a livello di stack

Questa sezione fornisce esempi che puoi seguire per creare filtri a livello di stack per Hooks.

CloudFormation

Esempio 1: Includi pile specifiche

L'esempio seguente specifica un `Include` elenco. L'Hook viene richiamato solo sugli stack `stack-test-1` `stack-test-2` `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"  
          ]  
        }  
      }  
    }  
  }  
}
```

Esempio 2: Escludi pile specifiche

Se i nomi degli stack vengono invece aggiunti all'`Exclude` elenco, l'Hook viene richiamato su qualsiasi pila senza nome, oppure `stack-test-1` `stack-test-2` `stack-test-3`

```
{  
  "CloudFormationConfiguration": {  
    "HookConfiguration": {  
      "HookInvocationStatus": "ENABLED",  
      "TargetOperations": [  
        "STACK",  
        "RESOURCE"  
      ],  
      "Exclude": [  
        "stack-test-1",  
        "stack-test-2",  
        "stack-test-3"  
      ]  
    }  
  }  
}
```

```
"FailureMode": "WARN",
"Properties": {},
"StackFilters": {
  "FilteringCriteria": "ALL",
  "StackNames": {
    "Exclude": [
      "stack-test-1",
      "stack-test-2",
      "stack-test-3"
    ]
  }
}
}
```

Esempio 3: Combinazione di inclusione ed esclusione

Se Include gli Exclude elenchi non sono specificati, l'Hook viene invocato solo sugli stack Include che non sono presenti nell'Exclude elenco. Nell'esempio seguente, l'Hook viene invocato solo su 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"
          ]
        }
      }
    }
  }
}
```

```
        ]
    }
}
}
}
```

Esempio 4: Combinazione di nomi e ruoli degli stack con criteri **ALL**

Il seguente Hook include tre nomi di stack e un ruolo di stack. Poiché *FilteringCriteria* è specificato come **ALL**, l'Hook viene richiamato solo per gli stack che hanno sia un nome di stack corrispondente che il ruolo dello stack corrispondente.

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

Esempio 5: combinazione di nomi e ruoli dello stack con criteri ANY

Il seguente Hook include tre nomi di stack e un ruolo di stack. Poiché *FilteringCriteria* è specificato come ANY, l'Hook viene richiamato per gli stack che hanno un nome di stack corrispondente o il ruolo dello stack corrispondente.

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

CloudFormation Aggancia i filtri target

Questo argomento fornisce indicazioni sulla configurazione dei filtri di destinazione per CloudFormation gli Hooks. Puoi utilizzare i filtri di destinazione per un controllo più granulare su quando e su quali risorse viene richiamato il tuo Hook. Puoi configurare filtri che vanno dal semplice targeting per tipo di risorsa a combinazioni più complesse di tipi di risorse, azioni e punti di richiamo.

Per specificare uno o più nomi di stack come filtri nella configurazione di Hooks, aggiungi una chiave sotto. **TargetFilters** **HookConfiguration**

TargetFilters supporta le seguenti proprietà.

Actions

Un array di stringhe che specifica le azioni da intraprendere. Per vedere un esempio, consulta [Esempio 1: filtro target di base](#).

Valori validi: CREATE | UPDATE | DELETE

Note

Per RESOURCE, e CLOUD_CONTROL targetSTACK, sono applicabili tutte le azioni di destinazione. Per CHANGE_SET gli obiettivi, è applicabile solo l'CREATEazione. Per ulteriori informazioni, consulta [Obiettivi Hook](#).

InvocationPoints

Un array di stringhe che specifica i punti di invocazione da utilizzare come target.

Valori validi: PRE_PROVISION

TargetNames

Un array di stringhe che specifica i nomi dei tipi di risorse da utilizzare come target, ad esempio.

AWS::S3::Bucket

I nomi delle destinazioni supportano nomi di destinazione concreti e la corrispondenza completa con caratteri jolly. Per ulteriori informazioni, consulta [Usare i caratteri jolly con i nomi degli obiettivi di Hook](#).

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

Maximum: 50

Targets

Un array di oggetti che specifica l'elenco di obiettivi da utilizzare per il filtraggio degli obiettivi.

Ogni oggetto nell'array targets ha le seguenti proprietà.

Actions

L'azione per l'obiettivo specificato.

Valori validi: CREATE | UPDATE | DELETE

InvocationPoints

Il punto di invocazione per il target specificato.

Valori validi: PRE_PROVISION

TargetNames

Il nome del tipo di risorsa da scegliere come target.

Note

Non è possibile includere contemporaneamente sia l'array di Targets oggetti che gli InvocationPoints array TargetNamesActions, o. Se si desidera utilizzare questi tre elementi eTargets, è necessario includerli nell'array di Targets oggetti. Per vedere un esempio, consulta [Esempio 2: utilizzo dell'array di Targets oggetti](#).

Esempi di filtri target

Questa sezione fornisce esempi che puoi seguire per creare filtri di destinazione per CloudFormation Hooks.

Esempio 1: filtro target di base

Per creare un filtro di destinazione di base incentrato su tipi di risorse specifici, utilizzate l'TargetFilters oggetto con l'Actionsarray. La seguente configurazione del filtro di destinazione richiamerà l'Hook su tutte le Create operazioni di destinazione e sulle Delete azioni per le operazioni di destinazione specificate (in questo caso, entrambe RESOURCE le STACK operazioni). Update

```
{
```

```

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

```

Esempio 2: utilizzo dell'array di Targets oggetti

Per filtri più avanzati, è possibile utilizzare l'array di Targets oggetti per elencare combinazioni specifiche di obiettivi, azioni e punti di invocazione. La seguente configurazione del filtro di destinazione richiamerà l'Hook prima CREATE e UPDATE le azioni sui bucket S3 e sulle tabelle DynamoDB. Si applica a entrambe le operazioni. STACK RESOURCE

```

{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "TargetFilters": {
        "Targets": [
          {
            "TargetName": "AWS::S3::Bucket",
            "Action": "CREATE",
            "InvocationPoint": "PRE_PROVISION"
          }
        ]
      }
    }
  }
}

```

```
},
{
  "TargetName": "AWS::S3::Bucket",
  "Action": "UPDATE",
  "InvocationPoint": "PRE_PROVISION"
},
{
  "TargetName": "AWS::DynamoDB::Table",
  "Action": "CREATE",
  "InvocationPoint": "PRE_PROVISION"
},
{
  "TargetName": "AWS::DynamoDB::Table",
  "Action": "UPDATE",
  "InvocationPoint": "PRE_PROVISION"
}
]
}
}
}
```

Usare i caratteri jolly con i nomi degli obiettivi di Hook

Puoi usare i caratteri jolly come parte del nome del bersaglio. Puoi usare caratteri jolly (* e ?) nei nomi dei tuoi obiettivi Hook. L'asterisco (*) rappresenta qualsiasi combinazione di caratteri. Il punto interrogativo (?) rappresenta ogni singolo carattere. È possibile utilizzare più ? caratteri * e in un nome di destinazione.

Example : Esempi di caratteri jolly dei nomi di destinazione negli schemi Hook

L'esempio seguente riguarda tutti i tipi di risorse supportati da Amazon S3.

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

```
 }  
 ...  
 }
```

L'esempio seguente corrisponde a tutti i tipi di risorse che hanno»Bucket«nel nome.

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

AWS::*:Bucket* Potrebbero risolversi in uno dei seguenti tipi di risorse concrete:

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

Example : Esempi di caratteri jolly dei nomi di destinazione negli schemi di configurazione di Hook

La seguente configurazione di esempio richiama l'Hook per CREATE le operazioni su tutti i tipi di risorse Amazon S3 e UPDATE per le operazioni su tutti i tipi di risorse di tabella denominati, come o. AWS::DynamoDB::Table 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"  
      }  
    ]  
  }  
}  
}
```

La seguente configurazione di esempio richiama l'Hook CREATE e UPDATE le operazioni su tutti i tipi di risorse Amazon S3 e anche CREATE per UPDATE e le operazioni su tutti i tipi di risorse di tabella denominati, come o. AWS::DynamoDB::Table AWS::Glue::Table

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

{

Example : stack specifici **Include**

Gli esempi seguenti specificano un **Include** elenco. L'Hook viene richiamato solo se i nomi degli stack iniziano con `stack-test-`

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

Example : pile specifiche **Exclude**

Gli esempi seguenti specificano un **Exclude** elenco. L'Hook viene richiamato su qualsiasi pila che non inizi con `stack-test-`

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

```
"FailureMode": "WARN",
"Properties": {},
"StackFilters": {
  "FilteringCriteria": "ALL",
  "StackNames": {
    "Exclude": [
      "stack-test-*"
    ]
  }
}
}
```

Example : Combinazione **Include** e **Exclude** per pile specifiche

Se vengono **Include** specificati **Exclude** elenchi, l'Hook viene richiamato solo sugli stack che corrispondono a quelli **Include** che non corrispondono nell'elenco. **Exclude** Nell'esempio seguente, l'Hook viene richiamato su tutte le pile che iniziano con, `stack-test-` ad eccezione delle pile denominate, e. `stack-test-1` `stack-test-2` `stack-test-3`

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

```
    }
}
}
}
```

Example : ruoli specifici **Include**

L'esempio seguente specifica un **Include** elenco con due pattern di caratteri jolly. La prima voce eseguirà l'Hook per qualsiasi ruolo che inizia con `hook-role` in qualsiasi `partition` and `account-id`. La seconda voce eseguirà qualsiasi ruolo in qualsiasi ruolo a `partition` cui appartiene `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 : ruoli **Exclude** specifici

Gli esempi seguenti specificano un **Exclude** elenco con due pattern di caratteri jolly. La prima voce salterà l'esecuzione di Hook quando un ruolo ha `exempt` nel nome qualsiasi `partition` `account-id`. La seconda voce salterà l'esecuzione di Hook quando un ruolo appartenente a `account-id` 123456789012 viene utilizzato con l'operazione `stack`.

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

Example : Combinazione **Include** e **Exclude** per ruoli specifici dei modelli ARN

Se vengono **Include** specificati **Exclude** elenchi, l'Hook viene richiamato solo sugli stack utilizzati con ruoli che corrispondono a quelli **Include** che non corrispondono nell'elenco. **Exclude** Nell'esempio seguente, l'Hook viene richiamato nelle operazioni di stack con **anypartition**, e **role nameaccount-id**, tranne se il ruolo appartiene a **account-id 123456789012**

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

```
        "StackRoles": {
            "Include": [
                "arn:*:iam::*:role/*"
            ],
            "Exclude": [
                "arn:*:iam::123456789012:role/*"
            ]
        }
    }
}
```

Example : Combinazione di nomi e ruoli dello stack con tutti i criteri

Il seguente Hook include una jolly per il nome dello stack e una jolly per il ruolo dello stack. Poiché `FilteringCriteria` è specificato come `ALL`, l'Hook viene richiamato solo per gli stack che hanno sia la corrispondenza che la corrispondenza. `StackName StackRoles`

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

Example : Combinando **StackNames** e **StackRoles** con qualsiasi criterio

Il seguente Hook include una jolly per il nome dello stack e una jolly per il ruolo dello stack. Poiché **FilteringCriteria** è specificato comeANY, l'Hook viene richiamato per lo stack che presenta corrispondenze o corrispondenze. **StackNames** **StackRoles**

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

Crea Hooks utilizzando modelli CloudFormation

Questa pagina fornisce collegamenti a CloudFormation modelli di esempio e argomenti di riferimento tecnici per Hooks.

Utilizzando i CloudFormation modelli per creare Hooks, puoi riutilizzare il template per configurare gli Hooks in modo coerente e ripetuto. Questo approccio ti consente di definire gli Hooks una sola volta e poi di fornire gli stessi Hook più e più volte in più regioni. Account AWS

CloudFormation offre i seguenti tipi di risorse specializzate per la creazione di Guard e Lambda Hook.

Attività	Soluzione	Link
Crea un Guard Hook	Usa il tipo di AWS::CloudFormation::GuardHook risorsa per creare e attivare un Guard Hook.	Modello di esempio Riferimento tecnico
Creare un Lambda Hook	Usa il tipo di AWS::CloudFormation::LambdaHook risorsa per creare e attivare un Lambda Hook.	Modello di esempio Riferimento tecnico

CloudFormation offre anche i seguenti tipi di risorse che è possibile utilizzare nei modelli di stack per la creazione di Hook personalizzati.

Attività	Soluzione	Link
Registra un Hook	Usa il tipo di AWS::CloudFormation::HookVersion risorsa per pubblicare una nuova o prima versione di un Hook personalizzato nel CloudFormation registro.	Modelli di esempio Riferimento tecnico
Imposta la configurazione di Hook	Usa il tipo di AWS::CloudFormation::HookType risorsa per impostare la configurazione di un Hook.	Modelli di esempio Riferimento tecnico

Attività	Soluzione	Link
	Imposta la <code>peConfig</code> risorsa per specificare la configurazione di un Hook personalizzato.	
Imposta la versione predefinita di Hook	Usa il tipo di AWS::CloudFormation::HookDefaultVersion risorsa per specificare la versione predefinita di un Hook personalizzato.	Modelli di esempio Riferimento tecnico
Registra il tuo account come editore	Usa il tipo di AWS::CloudFormation::Publisher risorsa per registrare il tuo account come editore di estensioni pubbliche (hook, moduli e tipi di risorse) nel CloudFormation registro.	Riferimento tecnico
Pubblica un Hook pubblicamente	Usa il tipo di AWS::CloudFormation::PublicTypeVersion risorsa per testare e pubblicare un Hook personalizzato registrato come Hook pubblico di terze parti.	Riferimento tecnico
Attiva Hooks pubblici di terze parti	Il tipo di AWS::CloudFormation::TypeActivation risorsa collabora con il tipo di AWS::CloudFormation::HookTypeConfig risorsa per attivare un Hook personalizzato pubblico di terze parti nel tuo account.	Riferimento tecnico

Concedi le autorizzazioni IAM per CloudFormation Hooks

Per impostazione predefinita, un nuovo utente Account AWS non è autorizzato a gestire gli Hooks utilizzando l'API Console di gestione AWS, AWS Command Line Interface (AWS CLI) o AWS . Per concedere l'autorizzazione agli utenti, un amministratore IAM può creare policy IAM. L'amministratore può quindi aggiungere le policy IAM ai ruoli e gli utenti possono assumere i ruoli.

Usa gli esempi di policy riportati in questo argomento per creare policy IAM personalizzate che consentano agli utenti di lavorare con gli Hooks.

Per scoprire come creare una policy basata sull'identità IAM utilizzando questi esempi di documenti di policy JSON, consulta [Definire le autorizzazioni IAM personalizzate con le policy gestite dai clienti nella Guida per l'utente IAM](#).

Questo argomento tratta le autorizzazioni necessarie per eseguire le seguenti operazioni:

- Gestisci gli Hooks: crea, modifica e disabilita gli Hooks nel tuo account.
- Pubblica gli Hooks pubblicamente: registra, testa e pubblica i tuoi Hooks personalizzati per renderli disponibili pubblicamente nel registro CloudFormation
- Visualizza i risultati delle chiamate: accedi e richiedi i risultati delle chiamate Hook nel tuo account.
- Visualizza i dettagli del risultato di una chiamata: accedi a informazioni dettagliate e alle linee guida per la correzione di uno specifico risultato di chiamata di Hook nel tuo account.

Durante la creazione delle policy IAM, è possibile trovare la documentazione relativa a tutte le azioni, le risorse e le chiavi di condizione associate al prefisso del *CloudFormation* servizio nella CloudFormation sezione [Actions, resources and condition keys for](#) del Service Authorization Reference.

Argomenti

- [Consenti a un utente di gestire gli Hooks](#)
- [Consenti a un utente di pubblicare pubblicamente Hooks personalizzati](#)
- [Consenti a un utente di visualizzare i risultati della chiamata di Hook](#)
- [Consenti a un utente di visualizzare i risultati dettagliati dell'invocazione di Hook](#)
- [AWS KMS policy chiave e autorizzazioni per crittografare i risultati di CloudFormation Hooks a riposo](#)

Consenti a un utente di gestire gli Hooks

Se devi consentire agli utenti di gestire le estensioni, inclusi gli Hooks, senza la possibilità di renderle pubbliche nel CloudFormation registro, puoi utilizzare il seguente esempio di politica IAM.

Important

Le chiamate `ActivateType` e le chiamate `SetTypeConfiguration` API collaborano per creare Hooks nel tuo account. Quando concedi a un utente l'autorizzazione a chiamare `SetTypeConfigurationAPI`, gli concedi automaticamente la possibilità di modificare e disabilitare gli Hooks esistenti. Non puoi utilizzare le autorizzazioni a livello di risorsa per limitare l'accesso a questa chiamata API. Pertanto, assicurati di concedere questa autorizzazione solo agli utenti autorizzati del tuo account.

JSON

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

Gli utenti che gestiscono gli Hooks potrebbero aver bisogno di alcune autorizzazioni correlate, ad esempio:

- Per visualizzare i controlli proattivi dal Control Catalog nella CloudFormation console, l'utente deve disporre dell'`controlcatalog>ListControls` autorizzazione in una policy IAM.

- Per registrare gli Hooks personalizzati come estensioni private nel CloudFormation registro, l'utente deve disporre dell'`cloudformation:RegisterType` autorizzazione in una policy IAM.

Consenti a un utente di pubblicare pubblicamente Hooks personalizzati

L'esempio seguente di politica IAM si concentra specificamente sulle funzionalità di pubblicazione. Utilizza questa policy se devi consentire agli utenti di rendere le estensioni, inclusi gli Hooks, disponibili pubblicamente nel CloudFormation registro.

Important

Publishing Hooks le rende disponibili pubblicamente ad altri. Account AWS Assicurati che solo gli utenti autorizzati dispongano di queste autorizzazioni e che le estensioni pubblicate soddisfino gli standard di qualità e sicurezza della tua organizzazione.

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": "*"  
    }  
  ]  
}
```

Consenti a un utente di visualizzare i risultati della chiamata di Hook

Le autorizzazioni IAM necessarie per visualizzare i risultati delle chiamate Hook cambiano a seconda del tipo di informazioni richieste.

Elenca i risultati della chiamata di Hook

Per elencare i risultati della chiamata di Hook, gli utenti necessitano di autorizzazioni diverse a seconda della richiesta API effettuata.

- Per concedere le autorizzazioni per richiedere tutti i risultati di Hook, i risultati per un Hook specifico o i risultati per uno specifico Hook e lo stato di invocazione, devi concedere l'accesso all'azione. `cloudformation>ListAllHookResults`
- Per concedere le autorizzazioni per richiedere risultati specificando una destinazione Hook, devi concedere l'accesso all'azione. `cloudformation>ListHookResults` Questa autorizzazione consente al chiamante dell'API di specificare i `TargetId` parametri `TargetType` and durante la chiamata. `ListHookResults`

Di seguito viene mostrato un esempio di una politica di autorizzazioni di base per elencare i risultati della chiamata di Hook. Le identità IAM (utenti o ruoli) con questa policy sono autorizzate a richiedere tutti i risultati di chiamata utilizzando tutte le combinazioni di parametri disponibili.

JSON

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

Controlla quali set di modifiche possono essere specificati

L'esempio seguente di policy IAM concede le autorizzazioni all'cloudformation:ListHookResultsazione per richiedere risultati specificando la destinazione dell'Hook. Tuttavia, nega l'azione anche se l'obiettivo è un set di modifiche denominato. 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"  
                }  
            }  
        }  
    ]  
}
```

Controlla quali ganci possono essere specificati

Il seguente esempio di policy IAM concede le autorizzazioni all'cloudformation:ListAllHookResultsazione per richiedere i risultati di chiamata solo quando l'ARN dell'Hook è fornito nella richiesta. Nega l'azione per un Hook ARN specificato.

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:cloudformation:us-  
east-1:123456789012:type/hook/MyCompany-MyHook"  
        }  
      }  
    }  
  ]  
}
```

Consenti a un utente di visualizzare i risultati dettagliati dell'invocazione di Hook

Per concedere le autorizzazioni per visualizzare i risultati dettagliati di una specifica chiamata Hook, è necessario concedere l'accesso all'azione. `cloudformation:GetHookResult` Questa autorizzazione consente agli utenti di recuperare informazioni dettagliate e linee guida per la riparazione di uno specifico risultato di chiamata di Hook. Per ulteriori informazioni, consulta [GetHookResult](#) nella documentazione di riferimento dell'API AWS CloudFormation .

Il seguente esempio di policy IAM concede le autorizzazioni per l'azione.

`cloudformation:GetHookResult`

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

Note

Puoi configurare Hooks per crittografare i risultati di invocazione dettagliati archiviati nel cloud con le tue chiavi. AWS KMS Per informazioni su come configurare la policy delle chiavi e le autorizzazioni IAM necessarie quando si utilizza una chiave gestita dal cliente per la crittografia, consulta. [AWS KMS policy chiave e autorizzazioni per crittografare i risultati di CloudFormation Hooks a riposo](#)

AWS KMS policy chiave e autorizzazioni per crittografare i risultati di CloudFormation Hooks a riposo

Questo argomento descrive come impostare la politica AWS KMS chiave e le autorizzazioni necessarie quando si specifica una chiave gestita dal cliente per crittografare i dati delle annotazioni Hooks disponibile dall'API. [GetHookResult](#)

Note

CloudFormation Hooks non necessita di un'autorizzazione aggiuntiva per utilizzare l'impostazione predefinita per Chiave di proprietà di AWS crittografare i dati delle annotazioni nel tuo account.

Argomenti

- [Panoramica](#)
- [Utilizzo del contesto di crittografia per controllare l'accesso alla chiave gestita dal cliente](#)
- [Policy delle chiavi KMS gestite dal cliente](#)
- [Autorizzazioni KMS per l'API SetTypeConfiguration](#)
- [Autorizzazioni KMS per l'API GetHookResult](#)

Panoramica

Quanto segue AWS KMS keys può essere utilizzato per crittografare i dati delle annotazioni Hook:

- [Chiave di proprietà di AWS](#)— Per impostazione predefinita, CloudFormation utilizza an Chiave di proprietà di AWS per crittografare i dati. Non è possibile visualizzare, gestire Chiavi di proprietà di AWS, utilizzare o controllare il loro utilizzo. Tuttavia, non è necessario eseguire una configurazione esplicita per proteggere la chiave utilizzata per crittografare i dati. Chiavi di proprietà di AWS sono forniti gratuitamente (senza canoni mensili o costi di utilizzo). A meno che non sia necessario verificare o controllare la chiave di crittografia che protegge i dati delle annotazioni, an Chiave di proprietà di AWS è una buona scelta.
- [Chiave gestita dal cliente](#): CloudFormation supporta l'uso di una chiave simmetrica gestita dal cliente che puoi creare, possedere e gestire per aggiungere un secondo livello di crittografia rispetto a quella esistente. Chiave di proprietà di AWS AWS KMS si applicano costi. Per ulteriori

informazioni, consulta [Creazione di chiavi](#) nella Guida per gli sviluppatori di AWS Key Management Service . Per gestire la tua chiave, usa il AWS Key Management Service (AWS KMS) nella [AWS KMS console](#) AWS CLI, o nell' AWS KMS API. Per ulteriori informazioni, consulta la [Guida per gli sviluppatori di AWS Key Management Service](#).

Puoi configurare le chiavi gestite dal cliente durante la creazione e l'aggiornamento degli Hooks. Quando fornisci la chiave gestita dal cliente, CloudFormation utilizza questa chiave per crittografare i dati delle annotazioni prima di archiviarli. Quando successivamente si accede ai dati delle annotazioni durante l'operazione dell'GetHookResultAPI, li decrittografa CloudFormation automaticamente. Per informazioni sulla configurazione della chiave di crittografia per Hooks, consulta. [Riferimento alla sintassi dello schema di configurazione Hook](#)

 **Important**

Tieni presente che l'KmsKeyId opzione per specificare una chiave gestita dal cliente è attualmente disponibile solo quando utilizzi AWS CLI per configurare il tuo Hook.

Utilizzo del contesto di crittografia per controllare l'accesso alla chiave gestita dal cliente

CloudFormation Hooks include automaticamente il contesto di crittografia in ogni operazione di memorizzazione e recupero delle annotazioni. Ciò consente di impostare le condizioni del contesto di crittografia nella politica delle chiavi per garantire che la chiave possa essere utilizzata solo per Hook specifici:

- `kms:EncryptionContext:aws:cloudformation:hooks:service`— Assicura che la chiave venga utilizzata solo dal servizio CloudFormation Hooks.
- `kms:EncryptionContext:aws:cloudformation:account-id`— Impedisce l'utilizzo delle chiavi tra account abbinando il tuo Account AWS ID.
- `kms:EncryptionContext:aws:cloudformation:arn`— Limita l'utilizzo a specifici Hooks utilizzando modelli ARN.

Queste condizioni forniscono una protezione aggiuntiva contro i confusi attacchi secondari collegando critograficamente i dati crittografati allo specifico contesto Hook.

Policy delle chiavi KMS gestite dal cliente

Quando si crea una chiave gestita dal cliente, è necessario definirne la politica chiave per consentire al servizio CloudFormation Hooks di eseguire le operazioni. AWS KMS Per utilizzare la seguente politica chiave, sostituiscila *placeholder values* con le tue informazioni.

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "EnableIAMUserDescribeKey",  
      "Effect": "Allow",  
      "Principal": {  
        "AWS": "arn:aws:iam::111122223333:role/ExampleRole"  
      },  
      "Action": "kms:DescribeKey",  
      "Resource": "*",  
      "Condition": {  
        "StringEquals": {  
          "kms:ViaService": "cloudformation.us-east-1.amazonaws.com"  
        }  
      }  
    },  
    {  
      "Sid": "EnableIAMUserGenerateDataKey",  
      "Effect": "Allow",  
      "Principal": {  
        "AWS": "arn:aws:iam::111122223333: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": "111122223333"  
        },  
        "StringLike": {  
          "kms:EncryptionContext:aws:cloudformation:arn":  
            "arn:aws:cloudformation:*:111122223333:hook/*"  
        }  
      }  
    }  
  ]  
}
```

```
},
{
  "Sid": "EnableIAMUserDecrypt",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333: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": "111122223333"
    },
    "StringLike": {
      "aws:SourceArn": "arn:aws:cloudformation:*:111122223333:hook/*"
    }
  }
},
{
  "Sid": "AllowHooksService",
  "Effect": "Allow",
  "Principal": {
    "Service": "hooks.cloudformation.amazonaws.com"
  },
  "Action": [
    "kms:Decrypt",
    "kms:GenerateDataKey"
  ],
  "Resource": "*",
  "Condition": {
```

```
  "StringEquals": {
    "aws:SourceAccount": "111122223333",
    "kms:EncryptionContext:aws:cloudformation:hooks:service": "hooks.cloudformation.amazonaws.com",
    "kms:EncryptionContext:aws:cloudformation:account-id": "111122223333"
  },
  "StringLike": {
    "aws:SourceArn": "arn:aws:cloudformation:*:111122223333:hook/*",
    "kms:EncryptionContext:aws:cloudformation:arn": "arn:aws:cloudformation:*:111122223333:hook/*"
  }
}
]
```

Questa policy concede le autorizzazioni sia ai ruoli IAM (prime tre istruzioni) che al servizio CloudFormation Hooks (ultime due istruzioni). La chiave di kms : ViaService condizione garantisce che la chiave KMS possa essere utilizzata solo tramite CloudFormation, impedendo chiamate dirette all'API KMS. Le operazioni chiave sono:

- kms : DescribeKey— Convalida le proprietà chiave e i metadati. Questa operazione è in istruzioni separate perché non può essere utilizzata con le condizioni del contesto di crittografia.
- kms : GenerateDataKey— Genera chiavi di crittografia dei dati per crittografare le annotazioni prima dell'archiviazione. Questa operazione include le condizioni del contesto di crittografia per il controllo degli accessi con ambito.
- kms : Decrypt— Decrypta i dati delle annotazioni precedentemente crittografati. Per i ruoli IAM, ciò include la condizione kms : ViaService Per il responsabile del servizio, ciò include le condizioni del contesto di crittografia.

Le chiavi aws : SourceAccount e aws : SourceArn condition forniscono la protezione principale contro gli attacchi confusi dei vice. Le condizioni del contesto di crittografia forniscono livelli di convalida aggiuntivi. Per ulteriori informazioni, consulta [Using aws:SourceArn or aws:SourceAccount condition keys](#) nella AWS Key Management Service Developer Guide.

⚠️ Important

I ruoli di esecuzione degli hook non richiedono AWS KMS autorizzazioni. Il responsabile del servizio CloudFormation Hooks esegue tutte le AWS KMS operazioni.

Autorizzazioni KMS per l'API **SetTypeConfiguration**

Durante la chiamata [SetTypeConfiguration](#) API, CloudFormation convalida le autorizzazioni degli utenti per crittografare i dati delle annotazioni con la chiave specificata. AWS KMS Aggiungi la seguente policy IAM all'utente o al ruolo che configurerà la crittografia utilizzando l'API.

SetTypeConfiguration Sostituisci `arn:aws:kms:us-east-1:123456789012:key/abc-123` con l'ARN della tua chiave gestita dal cliente.

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": "cloudformation:SetTypeConfiguration",  
      "Resource": "*"  
    },  
    {  
      "Effect": "Allow",  
      "Action": "kms:DescribeKey",  
      "Resource": "arn:aws:kms:us-east-1:123456789012:key/abc-123"  
    },  
    {  
      "Effect": "Allow",  
      "Action": "kms:GenerateDataKey",  
      "Resource": "arn:aws: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"  
        },  
        "StringLike": {  
          "kms:EncryptionContext:aws:cloudformation:arn":  
            "arn:aws:cloudformation:*:123456789012:hook/*"  
        }  
      }  
    }  
  ]  
}
```

```
    }
  }
]
}
```

Autorizzazioni KMS per l'API **GetHookResult**

[GetHookResult](#) Per richiedere Hooks che utilizzano la chiave gestita dal cliente, gli utenti devono disporre kms:Decrypt dell'autorizzazione per tale chiave. Aggiungi la seguente policy IAM all'utente o al ruolo che GetHookResult chiamerà. Sostituisci *arn:aws:kms:us-east-1:123456789012:key/abc-123* con l'ARN della tua chiave gestita dal cliente.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "cloudformation:GetHookResult",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "kms:Decrypt",
      "Resource": "arn:aws:kms:us-east-1:123456789012:key/abc-123"
    }
  ]
}
```

Cronologia dei documenti per la guida utente di CloudFormation Hooks

La tabella seguente descrive le importanti modifiche alla documentazione dall'ultima versione di Hooks. CloudFormation Per ricevere notifiche sugli aggiornamenti di questa documentazione, è possibile sottoscrivere un feed RSS.

- Ultimo aggiornamento della documentazione: 4 settembre 2025.

Modifica	Descrizione	Data
<u>Risultati dettagliati del controllo di conformità</u>	Gli hook ora supportano le annotazioni che forniscono risultati dettagliati dei controlli di conformità e linee guida per la correzione per ciascuna risorsa valutata. Visualizza questi risultati di convalida dettagliati tramite la CloudFormation console o il comando <code>get-hook-result</code> CLI.	13 novembre 2025
<u>Controlli proattivi come Hooks</u>	Ora puoi attivare Hooks basati sul controllo proattivo tramite la console CloudFormation o la CLI utilizzando i comandi <code>and. activate-type set-type-configuration</code> . È possibile configurare questi Hooks per applicare controlli proattivi specifici di AWS Control Tower Control Catalog per valutare le risorse durante le operazioni. <code>CREATE UPDATE</code>	4 settembre 2025

Riepilogo delle chiamate di Hooks

È ora possibile recuperare informazioni sulle invocazioni di Hook tramite la CloudFormation console o utilizzare il comando `list-hook-results` CLI per recuperare a livello di codice i dettagli delle chiamate. Ora puoi anche filtrare i `list-hook-results` risultati in base a Hook o allo stato di invocazione per concentrarti sulle invocazioni pertinenti.

4 settembre 2025

Hooks a livello di stack

Gli hook sono ora supportati a livello di stack, consentendo ai clienti di utilizzare gli CloudFormation Hooks per valutare nuovi modelli e potenzialmente impedire il proseguimento delle operazioni sullo stack.

13 novembre 2024

[AWS Cloud Control API](#) [Integrazione con gli Hooks](#)

Gli Hooks sono ora integrati con l'API Cloud Control, che consente ai clienti di utilizzare CloudFormation Hooks per ispezionare in modo proattivo la configurazione delle risorse prima del provisioning. Se vengono rilevate risorse non conformi, l'Hook fallisce l'operazione e impedisce il provisioning delle risorse oppure emette un avviso e consente il proseguimento dell'operazione di provisioning.

[AWS CloudFormation Guard Ganci](#)

AWS CloudFormation Guard è un linguaggio DSL (Domain Specific Language) open source e generico che puoi utilizzare per la creazione . policy-as-code Guard Hooks può valutare l'API e le CloudFormation operazioni di Cloud Control per ispezionare la configurazione delle risorse prima del provisioning. Se vengono rilevate risorse non conformi, l'Hook fallisce l'operazione e impedisce il provisioning delle risorse oppure emette un avviso e consente il proseguimento dell'operazione di provisioning.

13 novembre 2024

13 novembre 2024

[AWS Lambda Ganci](#)

AWS CloudFormation Lambda 13 novembre 2024

Hooks ti consente di valutare CloudFormation e gestire le operazioni dell'API Cloud Control in base al tuo codice personalizzato personalizzato. Hook può bloccare l'esecuzione di un'operazione o inviare un avviso al chiamante e consentire all'operazione di continuare.

[Guida per l'utente di Hooks](#)

Versione iniziale della Guida per l'utente di CloudFormation Hooks. Gli aggiornamenti

includono una nuova introduzione, una guida introduttiva, concetti e terminologia, filtri a livello di stack e argomenti aggiornati sui prerequisiti, la configurazione e lo sviluppo di Hooks. CloudFormation

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.