

The background features a dark navy blue field with abstract, overlapping organic shapes in shades of magenta and deep red. Two thin, light blue lines intersect diagonally across the upper right portion of the image.

# AWS re:Invent

DECEMBER 2 – 6, 2024 | LAS VEGAS, NV

COP407

# Coding for proactive controls with AWS CloudFormation

**David Killmon**

(he/him)

Principal Engineer  
AWS CloudFormation

**Ben Perak**

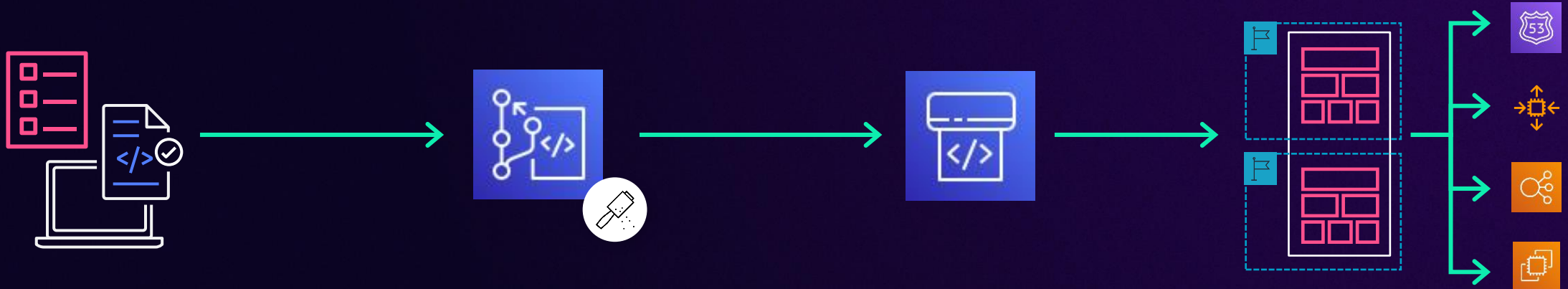
(he/him)

Principal Product Manager  
AWS CloudFormation



© 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.

# Provisioning and managing IaC



1

Code

2

Commit  
Git push/PR

3

Execute CI/CD  
test and  
deployment

4

Resources  
provisioned  
through  
CloudFormation



# The increasing role of deployment safety

AS IAC AUTHORIZING EVOLVES, ENSURING SAFE DEPLOYMENTS IS CRITICAL

- As customers empower their development team to use different methods to author templates, the need for guardrails increases
- Infrastructure-as-Code authoring is evolving with new layers of abstraction and the use of gen AI tools
- Increasing role of guardrails to force best practice configurations



# The increasing role of deployment safety

AS IAC AUTHORIZING EVOLVES, ENSURING SAFE DEPLOYMENTS IS CRITICAL

- As customers empower their development team to use different methods to author templates, the need for guardrails increases
- Infrastructure-as-Code authoring is evolving with new layers of abstraction and the use of gen AI tools
- Increasing role of guardrails to force best practice configurations



# Introduction to CloudFormation Hooks

SHIFT LEFT PROACTIVE CONTROLS

## Template

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

```
resultStatus = HookStatus.FAILED;
resultMessage = "Bucket Encryption not enabled for bucket with name: " + bucket.getBucketName();
```



Hooks (6)							
These are your activated Hooks within your current account and Region.							
<div>Find a Hook</div>							
	Name	Type	Mode	Targets	Status	Last updated	Description
<input type="radio"/>	<a href="#">Nrwl::Guard::Hook</a>	Guard	Fail	<a href="#">Stack</a>	Enabled	November 20, 2024, 09:49 AM (GMT-08:00)	AWS CloudFormation hook that invokes cfn-guard against any ruleset that exists in
<input type="radio"/>	<a href="#">bperak::Lambda::Test</a>	Lambda	Fail	<a href="#">Stack</a>	Disabled	November 11, 2024, 09:36 AM (GMT-08:00)	AWS CloudFormation hook that invokes AWS Lambda function you specify, where
<input type="radio"/>	<a href="#">bperak::Guard::demo44</a>	Guard	Fail	<a href="#">Stack</a>	Disabled	November 11, 2024, 09:11 AM (GMT-08:00)	AWS CloudFormation hook that invokes cfn-guard against any ruleset that exists in



# CloudFormation Hooks in the AWS Controls Narrative

AWS Prescriptive Guidance

Implementing security controls on AWS

Introduction

Targeted business outcomes

Security controls in the governance framework

▼ Types of security controls

Preventative controls

Proactive controls

Detective controls

Responsive controls

Next steps

FAQ

Resources

Document history

Glossary

Types of security controls

PDF | RSS

There are four main types of security controls:

- **Preventative controls** – These controls are designed to prevent an event from occurring.
- **Proactive controls** – These controls are designed to prevent the creation of noncompliant resources.
- **Detective controls** – These controls are designed to detect, log, and alert after an event has occurred.
- **Responsive controls** – These controls are designed to drive remediation of adverse events or deviations from your security baseline.

An effective security strategy includes all four types of security controls. While preventative controls are a first line of defense to help prevent unauthorized access or unwanted changes to your network, it is important to make sure that you establish detective and responsive controls so that you know when an event occurs and can take immediate and appropriate action to remediate it. Using proactive controls add another layer of security because it complements preventative controls, which are generally stricter in nature.

The following sections describe each type of control in more detail. They discuss the objectives, implementation process, use cases, technological considerations, and target outcomes of each control type.

Security controls in the governance framework

▼ Types of security controls

Preventative controls

**Proactive controls**

Detective controls

Responsive controls

Next steps

FAQ

Resources

Document history

Glossary

Technology

**CloudFormation hooks**

AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and Regions. **CloudFormation hooks** proactively evaluate the configuration of your CloudFormation resources before they are deployed. If noncompliant resources are found, it returns a failure status. Based on the hook failure mode, CloudFormation can fail the operation or present a warning that allows the user to continue with the deployment. You can use available hooks, or you can develop your own.

**AWS Control Tower**

AWS Control Tower helps you set up and govern an AWS multi-account environment, following prescriptive best practices. AWS Control Tower offers preconfigured **proactive controls** that you can enable in your landing zone. If your landing zone is setup using AWS Control Tower, you can use these optional proactive controls as a starting point for your organization. You can build additional, custom proactive controls in CloudFormation as needed.





# Use case #2: Prevent accidental operation impacts



## Original template

```
AWSTemplateFormatVersion: "2010-09-09"
Resources:
  myASG:
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      MinSize: 1
      MaxSize: 100
      DesiredCapacity: 50
      LaunchTemplate:
        LaunchTemplateId: !Ref myLaunchTemplate
        Version: !Ref myLaunchTemplateVersionNumber
        VPCZoneIdentifier: !Ref Subnets
```

Operational impacts often happen by making configuration mistakes. Prevent impacts by evaluating prior resource configurations against “to-be” configurations.



## Change set

```
AWSTemplateFormatVersion: "2010-09-09"
Resources:
  myASG:
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      MinSize: 1
      MaxSize: 10
      DesiredCapacity: 50
      LaunchTemplate:
        LaunchTemplateId: !Ref myLaunchTemplate
        Version: !Ref myLaunchTemplateVersionNumber
        VPCZoneIdentifier: !Ref Subnets
```

## Hook result

```
status = HookStatus.FAILED
message = f"The ASG change you are making is greater than 50% of the original value."
```



# Use case #3: Drive cost governance and best practices

CCOE and Cloud Cost Management professionals want to be notified when unnecessary spend occurs



## Template

```
AWSTemplateFormatVersion: "2010-09-09"
Resources:
  MyInstance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: "ami-79fd7eee"
      InstanceType: "p3dn.24xlarge"
```

## Hook result

```
status = HookStatus.FAILED
message = f"The EC2 instance you are provisioning is not covered by an Savings Plan or Reserved Instance."
```

```
status = HookStatus.FAILED
message = f"The EC2 instance you are provisioning is not allowed in a Sandbox environment."
```

# New features

## Support for new use cases

Template Hook invocations – inspect an entire template

Change set Hook invocations – surface violations early, and often

Cloud Control API Hook invocation – extend hooks beyond CloudFormation

## Simplifying the developer experience

New console and onboarding workflows

Managed Hook for Guard

Managed Hook for invoking an AWS Lambda function

New CloudFormation resources to provision Guard/Lambda Hooks



# Let's code!



© 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.

# Thank you!

**David Killmon**

killmond@amazon.com

**Ben Perak**

bperak@amazon.com



Please complete the session  
survey in the mobile app

