AWS CloudFormation Validation Pipeline

AWS Implementation Guide

Ruald Andreae

Jay McConnell

September 2017

Contents

About This Guide

This implementation guide discusses architectural considerations and configuration steps for deploying the AWS CloudFormation Validation Pipeline solution on the Amazon Web Services (AWS) Cloud. It includes links to AWS CloudFormation templates that launch, configure, and run the AWS services required to deploy this solution using AWS best practices for security and availability.

The guide is intended for IT infrastructure architects, administrators, and DevOps professionals who have practical experience architecting on the AWS Cloud.

Overview

Amazon Web Services (AWS) offers AWS Developer Tools, a set of services designed to enable customers to rapidly and safely deliver software. Together, these services help you follow continuous integration and continuous delivery practices, including secure storage and version control, and enable you to automatically build, validate, and deploy your code.

Many customers use AWS CloudFormation to manage their infrastructure as code and to help deploy AWS resources in a controlled and predictable way. As with other source code, DevOps teams are commonly tasked with validating AWS CloudFormation templates before launch to ensure they follow industry best practices and satisfy company-specific business and governance requirements. Often, they will use AWS Developer Tools to create their own development and deployment pipelines to validate templates against a set of predefined business, architectural, and security rules.

To help customers more easily and more reliably build, test, deploy, and manage their AWS CloudFormation templates, AWS offers the AWS CloudFormation Validation Pipeline solution. This reference implementation automatically provisions and configures the necessary services, including AWS CodePipeline and AWS Lambda, to run a set of predefined and customizable tests against AWS CloudFormation templates, and then stage those templates for manual deployment into a production environment. The validation pipeline automatically assesses logical and functional integrity using preconfigured AWS Lambda test functions, a default set of tests from cfn-nag (an open source linting tool for AWS CloudFormation), and any user-developed tests.

The AWS CloudFormation Validation Pipeline is designed to integrate with an existing AWS CodeCommit repository, and validates all AWS CloudFormation templates committed to that repository, helping to accelerate template development and deployment. The AWS CloudFormation Validation Pipeline leverages the AWS Quick Start testing methodology, which enables users to define specific template parameters and AWS Regions for testing. This solution includes a supplementary AWS CloudFormation template that configures a fully

functioning demo environment using a popular Quick Start architecture. This demo enables customers to modify and experiment with pipeline functionality while familiarizing themselves with Quick Start best practices for building AWS CloudFormation-based reference implementations.

Cost

You are responsible for the cost of the AWS services used while running this solution. Example monthly pricing is shown in the following table. This pricing represents the **base cost** for running the AWS CloudFormation Validation Pipeline with default settings in the US East (N. Virginia) Region and includes base charges for AWS CodePipeline, AWS CodeCommit, and Amazon DynamoDB.

AWS CodePipeline Pipelines	AWS CodeCommit Users	Monthly Base Cost
1	3	$4.81
10	10	$25.85
20	30	$61.45

These cost estimates do not reflect variable charges for AWS CodeCommit (storage and Git requests), AWS CodeBuild, Amazon Simple Storage Service (Amazon S3), or AWS Lambda beyond free tier usage. This pricing also does not include the cost of resources deployed in each test stack. Prices are subject to change. For full details, see the pricing webpage for each AWS service you will be using in this solution.

Architecture Overview

Deploying this solution with the **default parameters** builds the following environment in the AWS Cloud.

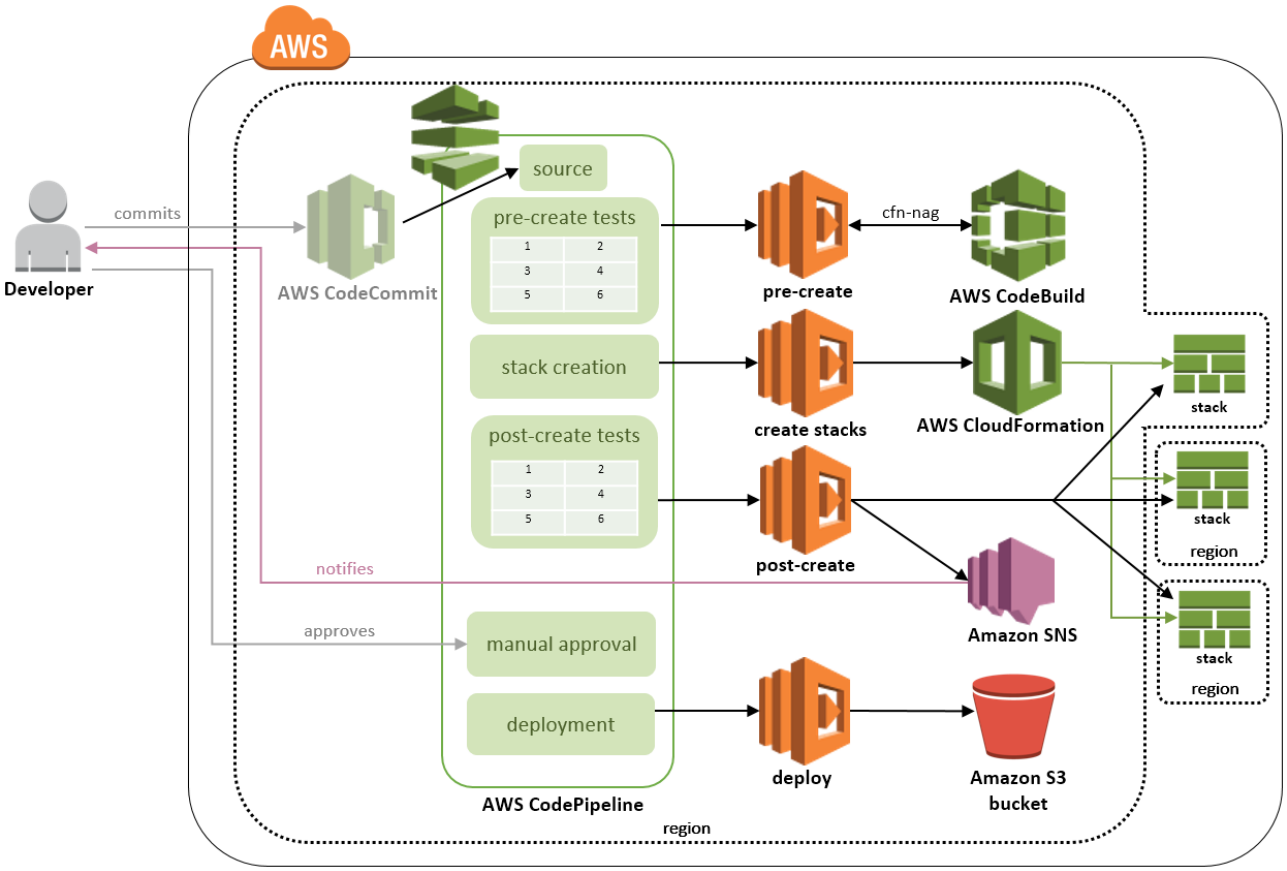

Figure 1: AWS CloudFormation Validation Pipeline default architecture

The solution includes two AWS CloudFormation templates that automate the deployment of a validation pipeline for AWS CloudFormation templates hosted in a customer's existing AWS CodeCommit repository. Together, the solution templates deploy and configure AWS CodePipeline, AWS Lambda functions that manage overall testing processes, necessary AWS Identity and Access Management roles, an Amazon Simple Notification Service (Amazon SNS) topic, an Amazon DynamoDB table, an Amazon S3 bucket, AWS CloudFormation test stacks, and AWS CodeBuild (as an optional resource).

When a user commits an AWS CloudFormation template to the AWS CodeCommit repository, the pipeline source action is triggered (see Working with Actions in AWS CodePipeline). This invokes a Lambda function that runs logical pre-create tests on the template code, including a default test on template syntax, an optional test that uses AWS CodeBuild to run cfn-nag rules, and any user-defined tests. The pipeline then invokes

Lambda functions that launch test stacks and configure resources, as defined in the customer-provided configuration file. Once the testing environment is configured, the pipeline invokes another Lambda function that runs functional post-create tests on the stacks. The solution includes preconfigured test functions, but the pipeline is designed to accommodate additional functions for custom testing scenarios. You have the option to keep or automatically delete stacks after testing.

If all tests are successful, the solution sends an Amazon SNS email notification to let you know that the template is ready for manual approval in AWS CodePipeline. If you approve the action, the pipeline invokes a Lambda function that copies the template to a solution-created S3 bucket. The function uploads the template with two different S3 prefixes: one that uses the commit ID, and one that uses a generic `latest` key. This allows you to overwrite an actively referenced template while preserving all previous versions.

The solution uses Amazon CloudWatch data on the solution's Lambda functions to create a custom report on pipeline failures and manual approvals. It uploads this report to the same S3 bucket as the approved templates.

Implementation Considerations

Test Functions

The AWS CloudFormation Validation Pipeline includes a set of preconfigured AWS Lambda functions for validating your template code (pre-create tests) and test stacks (post-create tests). These functions are part of the Central Microservices stack, which you launch using the solution-provided AWS CloudFormation template.

The following table describes each function and the test it runs.

Lambda Function	Test Type	Description
Validate_Template	Pre-create, Default	Runs a native AWS CloudFormation template validation command that checks the syntax of your template. **Note:** This Lambda function is hardcoded into the pipeline.
Lint_Template	Pre-create, Optional	Builds the template in AWS CodeBuild and runs predefined cfn-nag rules to validate the code. Cfn-nag is a third-party linting tool for AWS CloudFormation templates. For more information and troubleshooting guidance, see Appendix A. **Note:** If you choose to run this test, the solution will deploy AWS CodeBuild.

Lambda Function	Test Type	Description
Subnet_Name	Post-create, Optional	Runs a test to search for specific string values in VPC subnet name tags. This function is provided as a starting point for integrating tag-based string checks into your pipeline.
Test_Connectivity	Post-create, Optional	For templates that deploy networking resources, tests for outbound network connectivity. The test searches for private subnets by name, and, in each subnet, launches a Lambda function that attempts to connect to an endpoint. The test searches for subnet names that begin with `PrivateSubnet`, but you can modify the test (for example, name string and endpoint) as necessary.
AMI_Check	Post-create, Optional	Verifies that any Amazon Machine Images (AMIs) references in the template are up to date.

The solution includes these common tests as a starting point. Customers can use these Lambda functions as a reference for creating additional unit tests tailored to their requirements. For example, you can test a template against the AWS Well-Architected framework or rank a template according to the Center for Information Security (CIS) AWS Foundations Benchmark. See Appendix B for detailed instructions.

The solution's AWS CloudFormation template includes parameters for four pre-create tests and four post-create tests. You modify the AWS CodePipeline resource in the solution template to increase the number of tests. During initial configuration, enter the exact names of the Lambda functions (solution-provided or custom) to include those tests in your validation pipeline.

Configuration Files

The AWS CloudFormation Validation Pipeline uses customer-managed parameter files and a configuration file (`config.yml`) to launch test stacks. The parameter files supply default values for test stack configuration. The configuration file specifies the templates to validate, the parameter files to use for each test stack, and the AWS Regions to validate those test stacks in. This methodology is based on the AWS Quick Start framework for automated testing. For more information, see Create a parameters file for automated testing. For instructions on preparing these files, see Step 1.

These files are stored in the `ci` folder in your repository (see Repository Requirements). As explained in the previous sections, during initial configuration, you specify which tests (Lambda functions) to include in your pipeline. The pipeline will run the pre-create tests on your template(s). If your pre-create tests include calls to the AWS SDK, these calls will default to the same AWS Region(s) you have designated the test stack(s) to run in. The pipeline will then launch test stacks as specified in the configuration file, and run post-create tests on those stacks.

Test Stack Timeouts

The stack creation stage is a custom action, and is subject to an AWS CodePipeline timeout limit of one hour. If your test stacks collectively take longer than one hour to launch, the stage will fail and return a timeout error. To work around this issue, break your template into smaller substacks and test them in separate pipelines.

Repository Requirements

The AWS CloudFormation Validation Pipeline is designed to integrate with an existing AWS CodeCommit repository. To launch the solution successfully, your repository must contain two folders: `template` and `ci`. The `ci` folder contains the `config.yml` file described in the previous section.

If you want to use an Amazon Simple Storage Service (Amazon S3) bucket or GitHub as your repository location, you must modify the source stage of the pipeline and configure access appropriately. See Working with Pipelines in AWS CodePipeline for detailed instructions. For guidance on incorporating Git repositories hosted elsewhere, see Integrating Git with AWS CodePipeline on the AWS DevOps Blog.

Regional Restrictions

You must deploy this solution in an AWS Region that supports AWS CodePipeline, AWS CodeBuild, and AWS CodeCommit.[1] Once deployed, the solution is designed to validate templates in any AWS Region. The pipeline will create stacks and run post-create tests in the region(s) you specify in your test configuration (`config.yml`) file.

AWS CloudFormation Templates

This solution uses two AWS CloudFormation templates to automate the deployment of the AWS CloudFormation Validation Pipeline. You can download the templates before deployment.

View template

central-microservices.template: Use this template to deploy AWS Lambda functions (including the preconfigured test functions) and AWS Identity and Access Management (IAM) roles that manage pipeline tasks within an AWS Region. Launch this template only once in each AWS Region where you want to run the solution.

[1] For the most current service availability by region, see https://aws.amazon.com/about-aws/global-infrastructure/regional-product-services/

View template **main-pipeline.template:** Use this template to deploy AWS CodePipeline, AWS CodeBuild, AWS Lambda functions, an Amazon Simple Storage Service (Amazon S3) bucket, an Amazon Simple Notification Service (Amazon SNS) topic, an Amazon DynamoDB table, and associated IAM roles. Launch this template separately for each repository that contains templates to validate.

Automated Deployment

Before you launch the automated deployment, please review the implementation considerations and prerequisites discussed in this guide. Follow the step-by-step instructions in this section to configure and deploy the AWS CloudFormation Validation Pipeline into your account.

Time to deploy: Approximately 10 minutes

Prerequisites

Before you deploy the AWS CloudFormation Validation Pipeline, you must complete these tasks:

- Review the solution-provided tests, and prepare any custom tests.
- Add the required folders (`template` and `ci`) to your repository.

What We'll Cover

The procedure for deploying this architecture on AWS consists of the following steps. For detailed instructions, follow the links for each step.

Step 1. Prepare the Parameter and Configuration Files

- Download and modify the configuration file.
- Commit the configuration file to your repository.

Step 2. Launch the Central Microservices Stack

- Launch the AWS CloudFormation template into your AWS account.
- Enter values for required parameters: **Stack name**, **Create Stack Role ARN**.
- Review the other template parameters, and adjust if necessary.

Step 3. Launch the Main Pipeline Stack

- Launch the AWS CloudFormation template into your AWS account.
- Enter values for required parameters: **Stack name**, **Repository Name**, **Primary Email**, **Create Stack Role ARN**.
- Review the other template parameters, and adjust if necessary.

Step 1. Prepare the Parameter and Configuration Files

As explained in Configuration Files, the AWS CloudFormation Validation Pipeline leverages the AWS Quick Start framework for automated testing, and uses customer-managed files to define how and where to launch test stacks. These include parameter files that define stack values, and a configuration file that defines which parameters and regions to use for testing.

1. Prepare a parameters file(s) for your test stacks. For detailed guidance and a list of supported variables, see the Quick Start Automated Validation user guide.
2. The solution includes a default `config.yml` file that you can modify and commit to your repository. Go to this URL to download the configuration file locally.
3. In the configuration file, add references to your parameters file(s) as applicable.
4. Review and modify the test regions as necessary.
5. Commit both the parameter file(s) and configuration file to the `ci` folder in your repository.

Step 2. Launch the Central Microservices Stack

Use this procedure to launch the Central Microservices template. Deploy this template once per AWS Region, even if you plan to validate multiple repositories within the same region.

> **Note**: You are responsible for the cost of the AWS services used while running this solution. See the Cost section for more details. For full details, see the pricing webpage for each AWS service you will be using in this solution.

1. Sign in to the AWS Management Console and click the button to the right to launch the `central-microservices` AWS CloudFormation template.
   You can also download the template as a starting point for your own implementation.

2. The template is launched in the US East (N. Virginia) Region by default. To launch the Central Microservices stack in a different AWS Region, use the region selector in the console navigation bar.

> **Important**: You must launch this template in the same AWS Region as your AWS CodeCommit repository. This must be a region that supports AWS CodePipeline, AWS CodeBuild, and AWS CodeCommit.[2]

[2] For the most current service availability by region, see https://aws.amazon.com/about-aws/global-infrastructure/regional-product-services/

3. On the **Select Template** page, verify that you selected the correct template and choose **Next**.
4. On the **Specify Details** page, assign a name to your Central Microservices stack.
5. Under **Parameters**, enter an AWS CloudFormation service role in the **Create Stack Role ARN** parameter. This role must have the appropriate permissions to deploy the resources in your test stacks.
6. Choose **Next.**
7. On the **Options** page, choose **Next**.
8. On the **Review** page, review and confirm the settings. Be sure to check the box acknowledging that the template will create AWS Identity and Access Management (IAM) resources.
9. Choose **Create** to deploy the stack.

   You can view the status of the stack in the AWS CloudFormation Console in the **Status** column. You should see a status of CREATE_COMPLETE in approximately one minute.

Step 3. Launch the Main Pipeline Stack

The Main Pipeline template automatically deploys services that work together with the central microservices to create a validation pipeline for AWS CloudFormation templates in a specific repository. Complete this procedure for each repository with templates you want to validate, even if they are in the same AWS Region.

> **Note**: You are responsible for the cost of the AWS services used while running this solution. See the Cost section for more details. For full details, see the pricing webpage for each AWS service you will be using in this solution.

1. Sign in to the AWS Management Console and click the button to the right to launch the `main-pipeline` AWS CloudFormation template.

   Launch Main Pipeline

   You can also download the template as a starting point for your own implementation.

2. The template is launched in the US East (N. Virginia) Region by default. To launch the Main Pipeline stack in a different AWS Region, use the region selector in the console navigation bar.

> **Important**: You must launch this template in the same AWS Region as your AWS CodeCommit repository. This must be a region that supports AWS CodePipeline, AWS CodeBuild, and AWS CodeCommit.[3]

3. On the **Select Template** page, verify that you selected the correct template and choose **Next**.
4. On the **Specify Details** page, assign a name to your Main Pipeline stack.
5. Under **Parameters**, review the parameters for the template and modify them as necessary. This solution uses the following default values.

Parameter	Default	Description
Repository Name	*<Requires input>*	The AWS CodeCommit repository to use as the pipeline source
Branch	`master`	The AWS CodeCommit repository branch to monitor
Primary Email	*<Requires input>*	The primary email to notify for pending manual approvals and pipeline execution notifications
Secondary Email	<Optional input>	The secondary email to notify for pending manual approvals and pipeline execution notifications
Manual Approval	`Yes`	Require manual approval before uploading a template to Amazon S3
Create Stack Role ARN	*<Requires input>*	The AWS CloudFormation service role with the appropriate permissions to launch test stacks. Choose the same role you used in Step 2. Launch the Central Microservices Stack.
Delete Successful Stacks	`Yes`	When any test stack fails, delete all other test stacks that are in a complete or in-progress state
Retention Period: Successful Stacks	2	If you chose `Yes` for the previous parameter (**Delete Successful Stacks**), enter the number of days to retain successful test stacks before automatic deletion.
Delete Previous Stacks	`Yes`	Delete test stacks from previous pipeline executions before creating new test stacks
Delete Failed Stacks	`No`	Delete test stacks that are in a failed state
Retention Period: Failed Stacks	2	If you chose `Yes` for the previous parameter (**Delete Failed Stacks**), enter the number of days to retain failed test stacks before automatic deletion.
Logical (Pre-Stack-Create) Tests		
1st Test	<Optional input>	Enter the name of a pre-create Lambda function or other action (test).
2nd Test	<Optional input>	Enter the name of a pre-create Lambda function or other action (test).
3rd Test	<Optional input>	Enter the name of a pre-create Lambda function or other action (test).

[3] For the most current service availability by region, see https://aws.amazon.com/about-aws/global-infrastructure/regional-product-services/

Parameter	Default	Description
4th Test	<Optional input>	Enter the name of a pre-create Lambda function or other action (test).
Functional (Post-Stack-Create) Tests		
1st Test	<Optional input>	Enter the name of a post-create Lambda function or other action (test).
2nd Test	<Optional input>	Enter the name of a post-create Lambda function or other action (test).
3rd Test	<Optional input>	Enter the name of a post-create Lambda function or other action (test).
4th Test	<Optional input>	Enter the name of a post-create Lambda function or other action (test).

6. Choose **Next.**
7. On the **Options** page, choose **Next**.
8. On the **Review** page, review and confirm the settings. Be sure to check the box acknowledging that the template will create IAM resources.
9. Choose **Create** to deploy the stack.

   You can view the status of the stack in the AWS CloudFormation Console in the **Status** column. You should see a status of CREATE_COMPLETE in approximately five minutes. In the **Outputs** tab, you will see the Amazon S3 bucket (**DeploymentBucket**) the template created.

Security

When you build systems on AWS infrastructure, security responsibilities are shared between you and AWS. This shared model can reduce your operational burden as AWS operates, manages, and controls the components from the host operating system and virtualization layer down to the physical security of the facilities in which the services operate. For more information about security on AWS, visit the AWS Security Center.

Demo Environment

The solution includes a supplementary AWS CloudFormation template that configures a fully functioning, independent demo pipeline. This demo enables customers to explore pipeline functionality and experiment with custom tests in a contained environment.

The demo template automatically deploys the two main solution stacks, including all preconfigured AWS Lambda test functions, and creates an AWS CodeCommit repository. The

demo then clones an existing GitHub repository to AWS CodeCommit. This repository contains a modified version of the AWS Quick Start template for an Amazon Virtual Private Cloud (Amazon VPC) architecture, and includes sample configuration and parameter files. We chose to use this Quick Start template because it is a good example of a modular and customizable architecture, and also demonstrates how to incorporate networking-related tests in the pipeline.

Once the demo environment is configured, the pipeline will automatically begin to validate the template and launch test VPC stacks. See the Quick Start Reference Deployment Guide for an architecture diagram of the default virtual networking environment that each test stack creates.

Launch the Demo Environment

> **Note**: You are responsible for the cost of the AWS services used while running the demo. See the Cost section for more details. For full details, see the pricing webpage for each AWS service you will be using in this solution.

1. Sign in to the AWS Management Console and click the button to the right to launch the `aws-cloudformation-validation-pipeline` AWS CloudFormation template.

   Launch Demo Environment

   You can also download the template as a starting point for your own implementation.
2. The template is launched in the US East (N. Virginia) Region by default. To launch the demo stacks in a different AWS Region, use the region selector in the console navigation bar.

   > **Important**: You must launch this template in an AWS Region that supports AWS CodePipeline, AWS CodeBuild, and AWS CodeCommit.[4]
3. On the **Select Template** page, verify that you selected the correct template and choose **Next**.
4. On the **Specify Details** page, assign a name to your demo stack.
5. Under **Parameters**, review the parameters for the template and modify them as necessary. This solution uses the following default values.

[4] For the most current service availability by region, see https://aws.amazon.com/about-aws/global-infrastructure/regional-product-services/

Parameter	Default	Description
Repository URL	`https://s3.amazonaws.com/solutions-reference/aws-cloudformation-validation-pipeline/latest/demo_source.zip`	The GitHub repository the demo will clone to AWS CodeCommit and use as the pipeline source
Branch	`master`	The AWS CodeCommit repository branch to monitor
Primary Email	*<Requires input>*	The primary email to notify for pending manual approvals and pipeline execution notifications
Create Stack Role ARN	<Optional input>	The AWS CloudFormation service role used to launch test stacks. You can enter an existing role. If you leave this field blank, the solution will create a demo-specific Administrator role with minimum permissions necessary to deploy the Amazon VPC Quick Start.

6. Choose **Next.**
7. On the **Options** page, choose **Next**.
8. On the **Review** page, review and confirm the settings. Be sure to check the box acknowledging that the template will create AWS Identity and Access Management (IAM) resources.
9. Choose **Create** to deploy the demo environment.

You can view the status of the demo stacks in the AWS CloudFormation Console in the **Status** column. You should see a status of CREATE_COMPLETE in approximately five minutes.

These stacks include the overall demo stack, a Central Microservices Stack, a Main Pipeline stack, and a stack that clones a public Git repository into a new AWS CodeCommit repository. The pipeline will begin running post-create tests automatically, so you will see VPC test stacks in different AWS Regions.

You can review the demo pipeline stages and statuses in the AWS CodePipeline console, and experiment with the demo repository contents and custom test functions.

Additional Resources

AWS services

- AWS CloudFormation
- AWS CodeCommit
- AWS CodePipeline
- AWS CodeBuild
- AWS Lambda
- Amazon SNS
- Amazon DynamoDB
- Amazon S3
- Amazon VPC

Associated AWS DevOps Blog Post

- Integrating Git with AWS CodePipeline

Appendix A: Template Linting with cfn-nag

The `Lint_Template` AWS Lambda Function runs cfn-nag, a third-party linting tool that is designed to identify patterns in AWS CloudFormation templates that might result in insecure infrastructure. If you use this test, the solution will deploy AWS CodeBuild and run a default set of cfn-nag rules that check for the following issues:

- AWS Identity and Access Management and resource policies that are too permissive (wildcards)
- Security group rules that are too permissive (wildcards)
- Access logs that are not enabled for applicable resources
- Encryption that is not enabled for applicable resources

AWS CodeBuild records log data for the cfn-nag test. To troubleshoot linting failures, search your Amazon CloudWatch logs for AWS CodeBuild log files from this solution (the project name is listed in the resources of the Central Microservices AWS CloudFormation stack):

```
/aws/codebuild/<name of AWS CodeBuild project>
```

For more information on cfn-nag, see the Stelligent article Finding Security Problems Early in the Development Process of a CloudFormation Template with "cfn-nag" and the cfn-nag GitHub repository.

Appendix B: Custom Testing

To add a custom test to the validation pipeline, first add it to the Central Microservices template (`central-microservices.template`) and then reference it in the Main Pipeline template (`main-pipeline.template`) as follows.

> **Note:** The AWS CloudFormation Validation Pipeline provides a framework for tests run using AWS Lambda functions, however you can create custom tests using any technology that integrates with AWS CodePipeline actions. See Integrations with AWS CodePipeline Action Types for detailed information.

1. In the Main Pipeline template, find the section for the AWS CodePipeline resource.
2. Under `Properties` > `Stages` > `Pre/PostCreateTests`, add the new test `Action`, as applicable. The action should point to an existing AWS Lambda function or other test script.
3. For a new pre-create test (`PreCreateTests`), set the `InputArtifact` to `TemplateArtifact`.
   For a new post-create test (`PostCreateTests`), set the `InputArtifact` to `StackArtifact`.
4. The solution includes a class library (cfnpipeline) that you can reference in custom test actions (see the Validate_Template and Subnet_Name functions as examples). If you call this library, the action's `UserParameters` should be valid JSON and include the following JSON key pairs:

```
PreCreate:
  "UserParameters": {
    "Fn::Sub": [
      "{ \"CleanupPrevious\": \"${CleanupPrevious}\",
         \"CleanupNonFailed\": \"${CleanupNonFailed}\",
         \"CleanupFailed\": \"${CleanupFailed}\",
         \"CITestPath\": \"${CITestPath}\",
         \"ScratchBucket\": \"${ScratchBucket}\",
         \"StackCreationRoleArn\": \"${StackCreationRoleArn}\",
         \"KeyBucket\": \"${KeyBucket}\" }",
           {
             "CITestPath": {
               "Fn::FindInMap": [
                 "General",
                 "CIConfig",
                 "CITestPath"
                           ]
                     }
                }
```

```
                ]
            }
PostCreate:
  "UserParameters": {
    "Fn::Sub": [
      "{ \"CITestPath\": \"${CITestPath}\",
         \"ScratchBucket\": \"${ScratchBucket}\" }",
        {
          "CITestPath": {
          "Fn::FindInMap": [
            "General",
            "CIConfig",
            "CITestPath"
                          ]
                       }
                    }
                 ]
              }
```

Appendix C: Collection of Anonymous Data

This solution includes an option to send anonymous usage data to AWS. We use this data to better understand how customers use this solution and related services and products. When enabled, the following information is collected and sent to AWS each time the solution is deployed or an AWS Lambda function in the Central Microservices stack is invoked:

- **Solution ID:** The AWS solution identifier
- **Unique ID (UUID):** Randomly generated, unique identifier for each pipeline deployment and each function in the pipeline
- **Timestamp:** Data-collection timestamp
- **Pipeline Data:** Anonymous status data (success/failure) and error messages for each pipeline invocation

Note that AWS will own the data gathered via this survey. Data collection will be subject to the AWS Privacy Policy. To opt out of this feature, modify the `Mappings` section in both solution AWS CloudFormation templates as follows:

```
"SendAnonymousData": {
    "Enabled": "Yes"
 }
```

to

```
"SendAnonymousData": {
     "Enabled": "No"
 }
```

To opt out of this feature after launching the solution, in the AWS Lambda console, select each of the following Lambda functions and set the **SEND_ANONYMOUS_DATA** environment variable to `No`:

- `Test_Connectivity`
- `Lint_Template`
- `Subnet_Name`
- `Create_Stacks-<stackname>`
- `Generate_Report-<stackname>`
- `AMI_Check`
- `Validate_Template`
- `Deploy_To_S3-<stackname>`

Send Us Feedback

We welcome your questions and comments. Please post your feedback on the AWS Solutions Forum.

You can visit our GitHub repository to download the templates and scripts for this solution, and to share your customizations with others.

Document Revisions

Date	Change	In sections
September 2017	Initial release	--
September 2017	Removed guidance on sharing the Central Microservices stack; updated link to class library; updated AWS Lambda functions in anonymous data reference.	Test Functions, AWS CloudFormation Templates, Security, Appendix B, Appendix C

Notices

This document is provided for informational purposes only. It represents AWS's current product offerings and practices as of the date of issue of this document, which are subject to change without notice. Customers are responsible for making their own independent assessment of the information in this document and any use of AWS's products or services, each of which is provided "as is" without warranty of any kind, whether express or implied. This document does not create any warranties, representations, contractual commitments, conditions or assurances from AWS, its affiliates, suppliers or licensors. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

The AWS CloudFormation Validation Pipeline solution is licensed under the terms of the Amazon Software License available at https://aws.amazon.com/asl/.