**Provisioning AWS CodePipeline to enable developers to deploy CloudFormation Stack in a safe and secure manner enforcing various security control and best practices**

This demonstrates how a Cloud Infrastructure Team can provision a AWS CodePipeline which will enable developers to deploy their own CloudFormation (CFN) Stack through the CodePipeline while maintaining various security control and best practices

This can also be integrated with Service Catalog to allow developers to create their own pipeline as a self-service offering.

CodePipeline will carry out the following tasks -

1. Enforce CFN Validation check
2. Enforce CFN Linting check
3. CFN scan using open-source tool cfn-nag. This will look for Insecure infrastructure
4. Create IAM permission boundary which will restrict access permissions while creating any IAM role inside CFN
5. Enforced creation of AWS resources using a meaningful prefix (name starts with prefix\*) through CFN
6. Create a place holder for unit test

This solution will create a four stage CodePipeline as follows

**Stage1:**

Pulls the source code from a specific Source Code Repository. In this demo AWS CodeCommit has been used

**Stage2:**

It’s a validation stage implemented with AWS CodeBuild project. In this stage, CloudFormation template is validated, and checked for linting and insecure infrastructure. It also optionally packages the source code and push it to S3 bucket for further deployment. This build project will be created with hardcoded checks and validation which developers cannot modify.

**Stage3:**

This is a another CodeBuild Project used as a placeholder where developers are free to write any custom code including unit test before deploying the CFN. This CodeBuild project will execute the build step specified in **buildspec.yml** file in source code. In this Demo, buildspec.yml file is not doing anything and acting as a place holder for any customized action (like unit test)

**Stage4:**

This stage is implemented with AWS CodeDeploy to deploy the CloudFormation template once all the previous checks are passed.

A sample pipeline with these 4 stages will look like as follows

****

**How to deploy the solution**

There are two CloudFormation template (CFN) involved here. The CFN template AWSCodePipeline.yml will provision the AWS CodePipeline with 4 stages as stated above.

This can be used to create a product in AWS service catalog as well as a self-service offerings.

Another CFN template template.yml which will be created by the developer to deploy AWS Resource through this pipeline. CFN template.yml and other files (including parameter files, lambda source, buildspec.yml file etc.) will be placed into Source repo in CodeCommit to trigger the pipeline.

**Deploying CFN AWSCodePipeline.yml to create AWS CodePipeline**

Navigate to AWS CloudFormation console to create stack. Select and Upload AWSCodePipeline.yml file to create the stack.

**Parameters:**

**pCloudformationPackageNeeded**

if you are deploying lambda function and wants to package it up before deploying select “True”, else select “False”

If True is selected, it would execute the following command inside the CodeBuild and produce a **template.packaged.yml** file which will be deployed by CodeDeploy.

aws cloudformation package --template-file template.yml --s3-bucket ${S3BUCKET\_NAME} --output-template-file template.packaged.yml

If “False” is selected, then template.yml file will be directly deployed by CodeBuild

Note – In either case, developers need to make sure the CloudFormation template is named as template.yml

**pCodeCommitRepo**

Provide the CodeCommit Repository name where source code (template.yml and other files) will be placed.

**pCodeCommitBranch**

Provide the branch name of the Source Repo. The commit to this branch will trigger the pipeline.

**pPrefix**

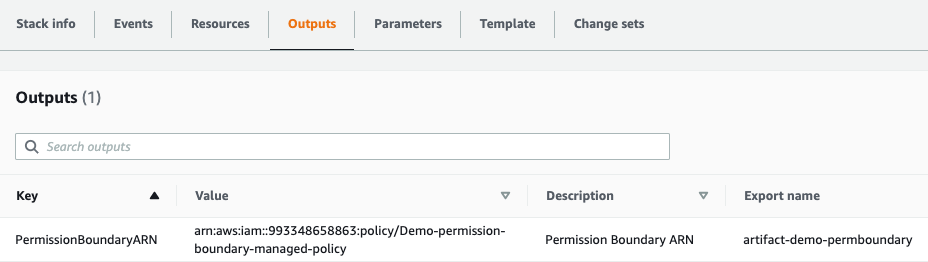
Provide a meaning prefix that will be used to name all the subsequent resources created by the stack. The CodePipeline will create all the necessary IAM roles for CloudFormation template to deploy resources only with ${pPrefix}\* name. template.yml file should be developed to create AWS resource only with ${pPrefix}\* naming convention. CodeDeploy Stage will also go ahead and create a parameter named pPrefix with Value provided by the user. This parameter and its value will be available inside the template.yml which will be deployed by CodeDeploy.

**Note** - This is to enforce the developers to create any AWS resource with a specific prefix so that IAM roles can be created and restricted based on the prefix if needed.



**Outputs:**

After successful deployment of this CloudFormation stack, navigate to outputs section. This stack will generate a permission boundary policy that needs to be set while creating any IAM role inside template.yml file. This is to provide a guardrail for user to prevent having privileged access. Copy the Value of the arn of the policy and



Copy the policy arn and use it while creating any IAM role inside template.yml as follows.

rMyLambdaRole:

Type: "AWS::IAM::Role"

Properties:

RoleName: !Sub ${pPrefix}-lambda-svc-role-${AWS::StackName}

**PermissionsBoundary:** **!Sub arn:aws:iam::${AWS::AccountId}:policy/${pPrefix}-permission-boundary-managed-policy**

AWSCodePipeline.yml file is attached.

**Deploying template.yml through the Codepipeline**

Once the AWS CodePipeline is created, it will monitor for any commit in the source repository/specific branch (these details will be provided during deployment of the pipeline). Developers needs to have following files in the source repo for CodePipeline to deploy the CFN.

**template.yml**

This is the main CloudFormation template file which defines all the AWS resource that needs to be created. All the resources should be named with ${pPrefix}\*.

**parameter.json**

This file contains all the parameters and their values that will be passed to CFN while deploying

**buildspec.yml**

This file contains any custom command (a good place to include unit test) that can be executed before the deployment. For more information, please refer <https://docs.aws.amazon.com/codebuild/latest/userguide/build-spec-ref.html>

In addition to that, I have included a lambda source folder that contains a sample lambda function which will be deployed via CFN.

All these sample files for the demo, can be found in **template.zip** file. Extract the content, place all the files in CodeCommit Repo/Branch to trigger the CodePipeline.

**Things to know for adjustment/enhancement**

Inside AWSCodePipeline.yml, the following resource can be adjusted as follows:

**rPermBoundaryPolicy**

This is the permission boundary policy (you get it from output section of the pipeline stack) that must be attached while creating or attaching any Role inside template.yml file.

This policy control maximum permission that will be granted to the newly created role inside your template.yml file. More information can be found at <https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_boundaries.html>

If you want to allow your user to create IAM role with any other permission, this can be modified accordingly

**rCloudFormationRole**

This Role defines what resources can be deployed via CFN template.yml. If you want to restrict the creation of certain resource, this needs to be modified. In this demo it only allows to create lambda function and DynamoDB table that starts with ${pPrefix}\*

**rCodeBuildProjectForValidation**

**source** property of this resource defines what all validation checks will be performed. This can be adjusted based on the requirement.