**Code related information will all json format policies.**

**role deny access:**

{

"Version": "2012-10-17",

"Statement": [

{

"**Effect": "Deny",**

"Action": [

"codecommit:GitPush",

"codecommit:DeleteBranch",

"codecommit:PutFile",

"codecommit:MergeBranchesByFastForward",

"codecommit:MergeBranchesBySquash",

"codecommit:MergeBranchesByThreeWay",

"codecommit:MergePullRequestByFastForward",

"codecommit:MergePullRequestBySquash",

"codecommit:MergePullRequestByThreeWay"

],

"Resource": "*arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo*",

"Condition": {

"StringEqualsIfExists": {

"codecommit:References": [

"refs/heads/*master*",

"refs/heads/*prod*"

]

},

"Null": {

"codecommit:References": false

}

}

}

]

}

**git comand lince create new branch:**

git checkout -b baranchname

**Cloudwatch rule for the codecommit:**

{

"source": [

"aws.codecommit"

],

"detail-type": [

"CodeCommit Repository State Change"

]

}

**Lambda codecommit:with python script polacies:it will monitor all codecommit activities and will act it.**

<https://docs.aws.amazon.com/codecommit/latest/userguide/how-to-notify-lambda.html>

import json

import boto3

codecommit = boto3.client('codecommit')

def lambda\_handler(event, context):

*#Log the updated references from the event*

references = { reference['ref'] for reference in event['Records'][0]['codecommit']['references'] }

print("References: " + str(references))

*#Get the repository from the event and show its git clone URL*

repository = event['Records'][0]['eventSourceARN'].split(':')[5]

try:

response = codecommit.get\_repository(repositoryName=repository)

print("Clone URL: " +response['repositoryMetadata']['cloneUrlHttp'])

return response['repositoryMetadata']['cloneUrlHttp']

except Exception as e:

print(e)

print('Error getting repository {}. Make sure it exists and that your repository is in the same region as this function.'.format(repository))

raise e

**Lambda code commit test json format:**

{

"Records": [

{

"awsRegion": "ap-south-1",

"codecommit": {

"references": [

{

"commit": "5c4ef1049f1d27deadbeeff313e0730018be182b",

"ref": "refs/heads/master"

}

]

},

"customData": "this is custom data",

"eventId": "5a824061-17ca-46a9-bbf9-114edeadbeef",

"eventName": "TriggerEventTest",

"eventPartNumber": 1,

"eventSource": "aws:codecommit",

"eventSourceARN": "arn:aws:codecommit:ap-south-1:123456789012:my-repo",

"eventTime": "2016-01-01T23:59:59.000+0000",

"eventTotalParts": 1,

"eventTriggerConfigId": "5a824061-17ca-46a9-bbf9-114edeadbeef",

"eventTriggerName": "my-trigger",

"eventVersion": "1.0",

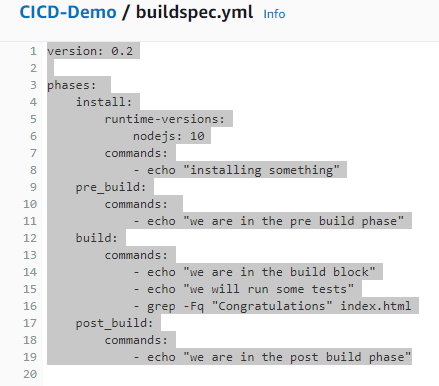
"userIdentityARN": "arn:aws:iam::123456789012:root"

}

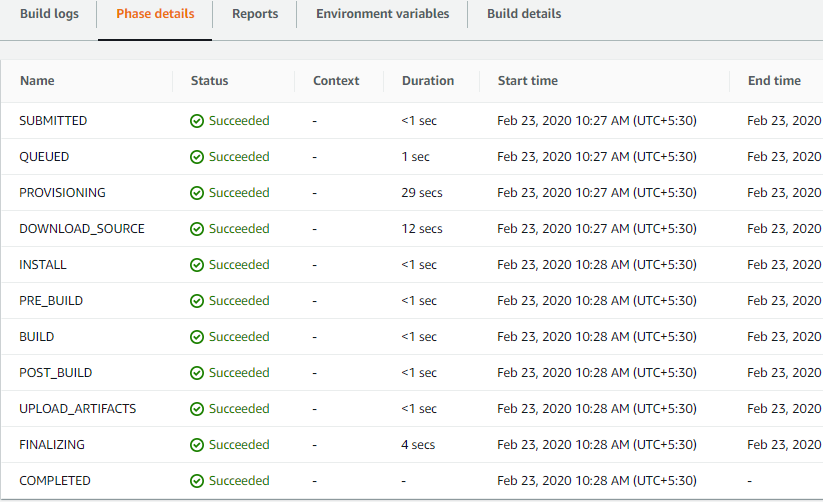
]

[**buildspec.yml**](https://ap-south-1.console.aws.amazon.com/codesuite/codecommit/repositories/CICD-Demo/browse/refs/heads/master/--/buildspec.yml?region=ap-south-1) **file:by using this file will build our build pipelins**

**ex:**

****

**Run the job will trigger with below phases:**

****

**BUILDSEP.YML sample file formate:** <https://docs.aws.amazon.com/codebuild/latest/userguide/build-spec-ref.html>

version: 0.2

run-as: *Linux-user-name*

env:

variables:

*key*: "*value*"

*key*: "*value*"

parameter-store:

*key*: "*value*"

*key*: "*value*"

exported-variables:

- *variable*

- *variable*

secrets-manager:

*key*: *secret-id*:*json-key*:*version-stage*:*version-id*

git-credential-helper: *yes*

proxy:

upload-artifacts: *yes*

logs: *yes*

phases:

install:

run-as: *Linux-user-name*

runtime-versions:

*runtime*: *version*

*runtime*: *version*

commands:

- *command*

- *command*

finally:

- *command*

- *command*

pre\_build:

run-as: *Linux-user-name*

commands:

- *command*

- *command*

finally:

- *command*

- *command*

build:

run-as: *Linux-user-name*

commands:

- *command*

- *command*

finally:

- *command*

- *command*

post\_build:

run-as: *Linux-user-name*

commands:

- *command*

- *command*

finally:

- *command*

- *command*

reports:

*report-name-or-arn*:

files:

- *location*

- *location*

base-directory: *location*

discard-paths: *yes*

file-format: JunitXml | CucumberJson | VisualStudioTrx | TestNGXml

artifacts:

files:

- *location*

- *location*

name: *artifact-name*

discard-paths: *yes*

base-directory: *location*

secondary-artifacts:

*artifactIdentifier*:

files:

- *location*

- *location*

name: *secondary-artifact-name*

discard-paths: *yes*

base-directory: *location*

*artifactIdentifier*:

files:

- *location*

- *location*

discard-paths: *yes*

base-directory: *location*

cache:

paths:

- *path*

- *path*

***Adding artefacts as well as environmental variables:***

*version: 0.2*

*phases:*

*install:*

*runtime-versions:*

*nodejs: 10*

*commands:*

*- printenv*

*- echo "installing something"*

*pre\_build:*

*commands:*

*- echo "we are in the pre build phase"*

*build:*

*commands:*

*- echo "we are in the build block"*

*- echo "we will run some tests"*

*- grep -Fq "Congratulations" index.html*

*post\_build:*

*commands:*

*- echo "we are in the post build phase"*

***artifacts:***

***files:***

***- '\*\*/\*'***

***name: Mywebapp-artifacts***

*Cloud watch rule to auto start our jobs-like web hooks:*

*{*

*"source": [*

*"aws.codebuild"*

*],*

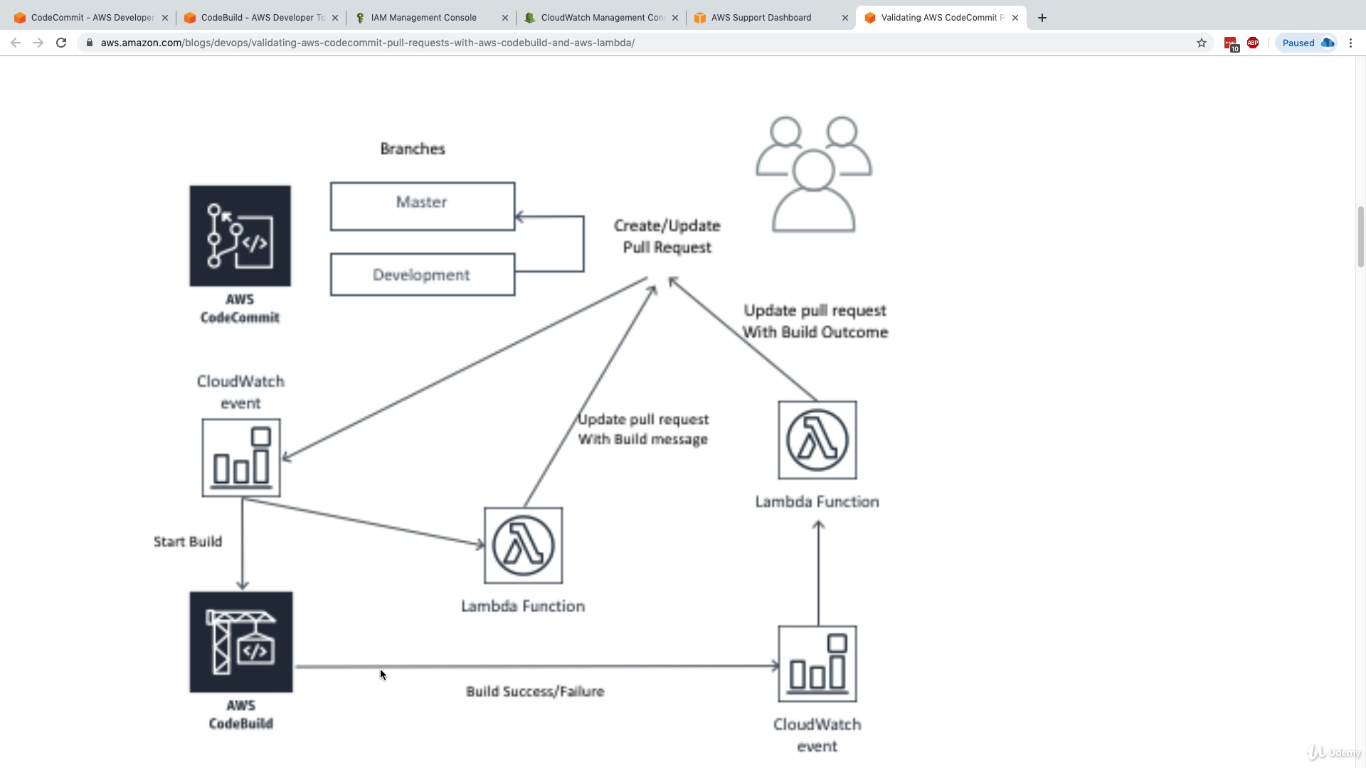
*"detail-type": [*

*"CodeBuild Build State Change"*

*]*

*}*

*Cloudwath with Lambda fuction:*

**

**AWS CodeCommit**

Create secure repositories for sharing your code in the cloud

AWS CodeCommit is a fully-managed source control service that makes it easy for companies to host secure and highly scalable private Git repositories. CodeCommit eliminates the need to operate your own source control system or worry about scaling its infrastructure.

**AWS CodeBuild**

Build and test code with elastic scaling. Pay only for the build time you use.

AWS CodeBuild is a fully managed continuous integration service that compiles source code, runs tests, and produces software packages that are ready to deploy. With CodeBuild, you don’t need to provision, manage, and scale your own build servers. CodeBuild scales continuously and processes multiple builds concurrently, so your builds are not left waiting in a queue.

**AWS CodeDeploy**

Automate code deployments to maintain application uptime

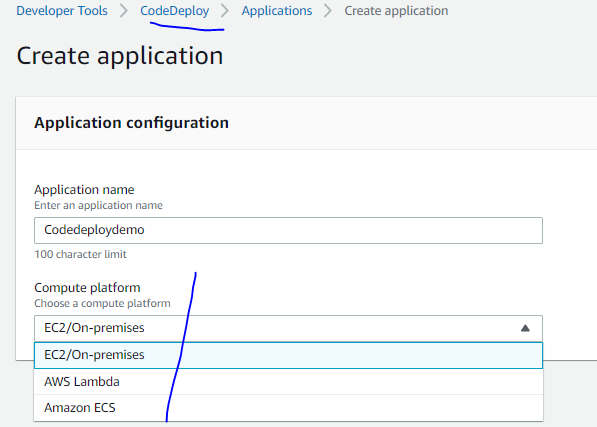
AWS CodeDeploy is a fully managed deployment service that automates software deployments to compute services such as Amazon EC2, AWS Lambda, and your on-premises servers. AWS CodeDeploy makes it easier for you to rapidly release new features, helps you avoid downtime during application deployment, and handles the complexity of updating your applications.

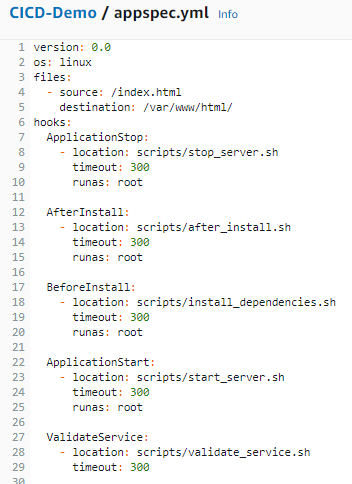
*After created Ec2 instacne connect and run below commands:*

*CreATING*

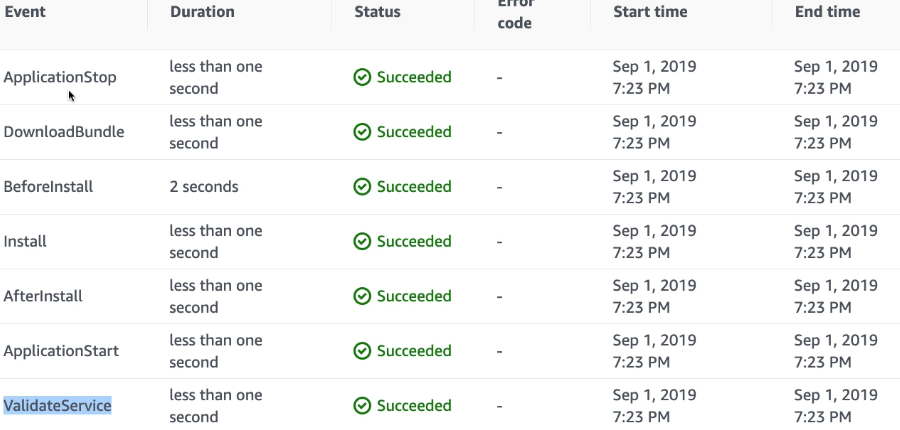
*CCc*

*CREATING THE CODEDEPLOYMENT*

**

**

*The above script will run and deploy the required deployment path as will web scripts and will deploying will as sequential manner.*

**

***index.htmlfilr:***

***<!DOCTYPE html>***

***<html>***

***<head>***

***<meta charset="utf-8">***

***<title>Sample Deployment</title>***

***<style>***

***body {***

***color: #ffffff;***

***background-color: #0188cc;***

***font-family: Arial, sans-serif;***

***font-size: 14px;***

***}***

***h1 {***

***font-size: 500%;***

***font-weight: normal;***

***margin-bottom: 0;***

***}***

***h2 {***

***font-size: 200%;***

***font-weight: normal;***

***margin-bottom: 0;***

***}***

***</style>***

***</head>***

***<body>***

***<div align="center">***

***<h1>Congratulations</h1>***

***<h2>This application was deployed using AWS CodeDeploy.</h2>***

***<p>For next steps, read the <a href="http://aws.amazon.com/documentation/codedeploy">AWS CodeDeploy Documentation</a>.</p>***

***</div>***

***</body>***

***</html>***

***Stop\_server.sh***

***#!/bin/bash***

***isExistApp = `pgrep httpd`***

***if [[ -n $isExistApp ]]; then***

***service httpd stop***

***fi***

***after\_install.sh:***

***#!/bin/bash***

***EC2\_INSTANCE\_ID=$(curl -s http://169.254.169.254/latest/meta-data/instance-id)***

***EC2\_AZ=$(curl -s http://169.254.169.254/latest/meta-data/placement/availability-zone)***

***sed -i "s/was deployed/was deployed on $EC2\_INSTANCE\_ID in $EC2\_AZ/g" /var/www/html/index.html***

***Install\_dependences.sh:***

***#!/bin/bash***

***yum install -y httpd***

***start\_server.sh:***

***#!/bin/bash***

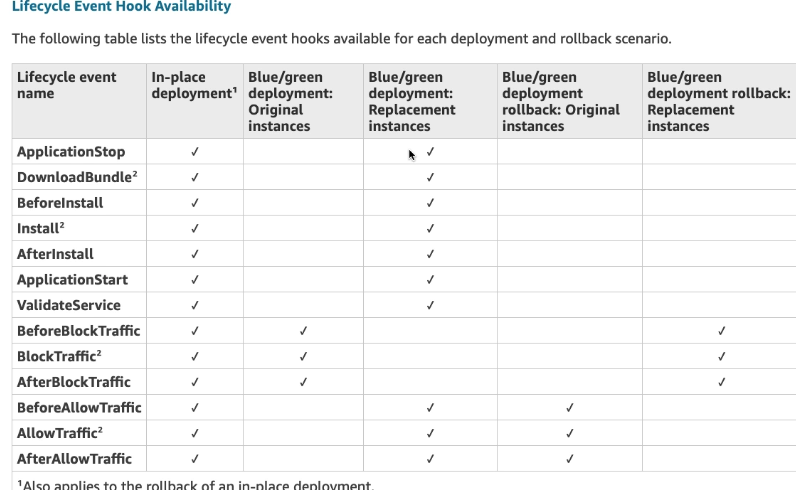
***service httpd start***

***Validate\_service.sh:***

***#!/bin/bash***

***# verify we can access our webpage successfully***

***curl -v --silent localhost:80 2>&1 | grep Congratulations***

******

Monitoring Deployments with Amazon CloudWatch Events

You can use Amazon CloudWatch Events to detect and react to changes in the state of an instance or a deployment (an "event") in your CodeDeploy operations. Then, based on rules you create, CloudWatch Events will invoke one or more target actions when a deployment or instance enters the state you specify in a rule. Depending on the type of state change, you might want to send notifications, capture state information, take corrective action, initiate events, or take other actions. You can select the following types of targets when using CloudWatch Events as part of your CodeDeploy operations:

* AWS Lambda functions
* Kinesis streams
* Amazon SQS queues
* Built-in targets (CloudWatch alarm actions)
* Amazon SNS topics

The following are some use cases:

* Use a Lambda function to pass a notification to a Slack channel whenever deployments fail.
* Push data about deployments or instances to a Kinesis stream to support comprehensive, real-time status monitoring.
* Use CloudWatch alarm actions to automatically stop, terminate, reboot, or recover Amazon EC2 instances when a deployment or instance event you specify occurs.

# Redeploy and Roll Back a Deployment with CodeDeploy

[**PDF**](https://docs.aws.amazon.com/codedeploy/latest/userguide/codedeploy-user.pdf#deployments-rollback-and-redeploy)

[**Kindle**](https://www.amazon.com/dp/B07641F5CS)

[**RSS**](https://docs.aws.amazon.com/codedeploy/latest/userguide/aws-codedeploy-user-guide-updates.rss)

CodeDeploy rolls back deployments by redeploying a previously deployed revision of an application as a new deployment. These rolled-back deployments are technically new deployments, with new deployment IDs, rather than restored versions of a previous deployment.

Deployments can be rolled back automatically or manually.

**Topics**

* [Automatic Rollbacks](https://docs.aws.amazon.com/codedeploy/latest/userguide/deployments-rollback-and-redeploy.html#deployments-rollback-and-redeploy-automatic-rollbacks)
* [Manual Rollbacks](https://docs.aws.amazon.com/codedeploy/latest/userguide/deployments-rollback-and-redeploy.html#deployments-rollback-and-redeploy-manual-rollbacks)
* [Rollback and Redeployment Workflow](https://docs.aws.amazon.com/codedeploy/latest/userguide/deployments-rollback-and-redeploy.html#deployments-rollback-and-redeploy-workflow)
* [Rollback Behavior with Existing Content](https://docs.aws.amazon.com/codedeploy/latest/userguide/deployments-rollback-and-redeploy.html#deployments-rollback-and-redeploy-content-options)

## Automatic Rollbacks

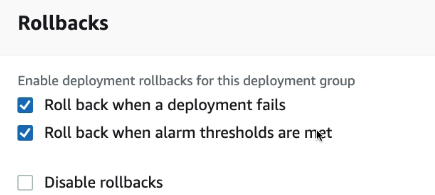
You can configure a deployment group or deployment to automatically roll back when a deployment fails or when a monitoring threshold you specify is met. In this case, the last known good version of an application revision is deployed. You configure automatic rollbacks when you create an application or create or update a deployment group.

## Manual Rollbacks

If you have not set up automatic rollbacks, you can manually roll back a deployment by creating a new deployment that uses any previously deployed application revision and following the steps to redeploy a revision. You might do this if an application has gotten into an unknown state. Rather than spending a lot of time troubleshooting, you can redeploy the application to a known working state. For more information, see [Create a Deployment with CodeDeploy](https://docs.aws.amazon.com/codedeploy/latest/userguide/deployments-create.html).

## Rollback and Redeployment Workflow

When automatic rollback is initiated, or when you manually initiate a redeployment or manual rollback, CodeDeploy first tries to remove from each participating instance all files that were last successfully installed. CodeDeploy does this by checking the cleanup file:



Working with Amazon EC2 Instances for CodeDeploy

An Amazon EC2 instance is a virtual computing environment that you create and configure using Amazon Elastic Compute Cloud. Amazon EC2 provides scalable computing capacity in the AWS Cloud. You can use Amazon EC2 to launch as many or as few virtual servers as you need for your CodeDeploy deployments.

For more information about Amazon EC2, see [*Amazon EC2 Getting Started Guide*](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/).

The instructions in this section show you how to create and configure Amazon EC2 instances for use in your CodeDeploy deployments.

**Topics**

* [Create an Amazon EC2 Instance for CodeDeploy](https://docs.aws.amazon.com/codedeploy/latest/userguide/instances-ec2-create.html)
* [Create an Amazon EC2 Instance (AWS CloudFormation Template)](https://docs.aws.amazon.com/codedeploy/latest/userguide/instances-ec2-create-cloudformation-template.html)
* [Configure an Amazon EC2 Instance](https://docs.aws.amazon.com/codedeploy/latest/userguide/instances-ec2-configure.html)

## Deployment Configurations on an Amazon ECS/LAMBDA Compute Platform

When you deploy to an Amazon ECS compute platform, the deployment configuration specifies how traffic is shifted to the updated Amazon ECS task set.

There are three ways traffic can be shifted during a deployment:

* **Canary**: Traffic is shifted in two increments. You can choose from predefined canary options that specify the percentage of traffic shifted to your updated Amazon ECS task set in the first increment and the interval, in minutes, before the remaining traffic is shifted in the second increment.
* **Linear**: Traffic is shifted in equal increments with an equal number of minutes between each increment. You can choose from predefined linear options that specify the percentage of traffic shifted in each increment and the number of minutes between each increment.
* **All-at-once**: All traffic is shifted from the original Amazon ECS task set to the updated Amazon ECS task set all at once.