# Cloud computing:

Cloud computing is the on-demand availability of [computer](https://en.wikipedia.org/wiki/Computer) [system resources](https://en.wikipedia.org/wiki/System_resource), especially [data storage](https://en.wikipedia.org/wiki/Data_storage) and [computing power](https://en.wikipedia.org/wiki/Computing_power), without direct active management by the user. The term is generally used to describe [data centers](https://en.wikipedia.org/wiki/Data_center) available to many users over the [Internet](https://en.wikipedia.org/wiki/Internet).

# What is a public cloud?

Public clouds are the most common way of deploying cloud computing. The cloud resources (like servers and storage) are owned and operated by a third-party [cloud service provider](https://azure.microsoft.com/en-in/overview/what-is-cloud-computing/) and delivered over the Internet. With a public cloud, all hardware, software and other supporting infrastructure is owned and managed by the cloud provider. In a public cloud, we can share the same hardware, storage and network devices with other organisations. We can access services and manage our account using a web browser. Public cloud deployments are frequently used to provide web-based email, online office applications, storage and testing and development environments.

# Advantages of public clouds:

* Lower costs—no need to purchase hardware or software and we can pay only for the service you use.
* No maintenance—our service provider provides the maintenance.
* Near-unlimited scalability—on-demand resources are available to meet your business needs.
* High reliability—a vast network of servers ensures against failure.

# What is a private cloud?

A private cloud consists of computing resources used exclusively by one business or organisation. The private cloud can be physically located at your organisation’s on-site datacenter or it can be hosted by a third-party service provider. But in a private cloud, the services and infrastructure are always maintained on a private network and the hardware and software are dedicated solely to your organisation. In this way, a private cloud can make it easier for an organisation to customise its resources to meet specific IT requirements. Private clouds are often used by government agencies, financial institutions, any other mid- to large-size organisations with business-critical operations seeking enhanced control over their environment.

# Advantages of a private clouds:

* More flexibility—your organisation can customise its cloud environment to meet specific business needs.
* Improved security—resources are not shared with others, so higher levels of control and security are possible.
* High scalability—private clouds still afford the scalability and efficiency of a public cloud.

# What is a hybrid cloud?

hybrid clouds combine on-premises infrastructure, or private clouds, with public clouds so organisations can reap the advantages of both. In a hybrid cloud, data and applications can move between private and public clouds for greater flexibility and more deployment options.

Advantages of hybrid clouds:

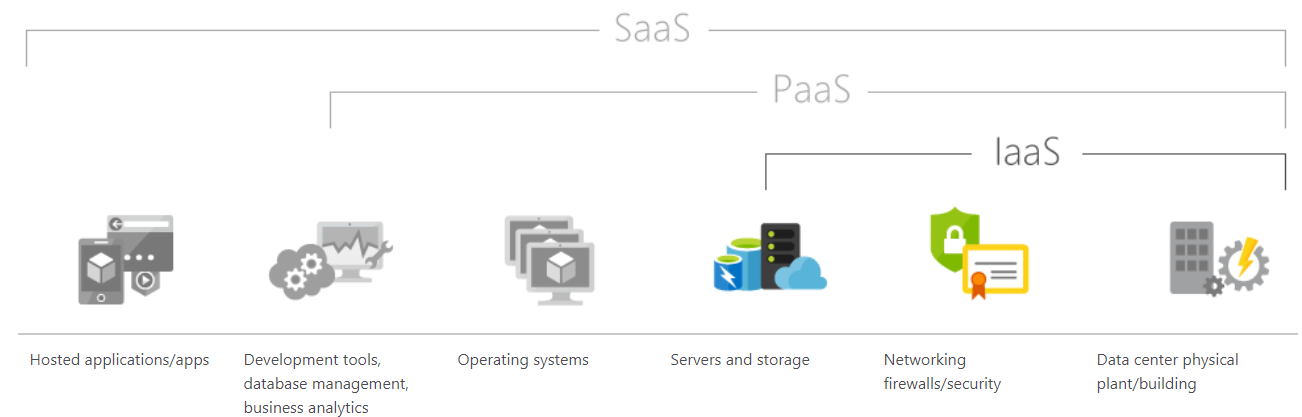
Control—your organisation can maintain a private infrastructure for sensitive assets.

* Flexibility—you can take advantage of additional resources in the public cloud when you need them.
* Cost-effectiveness—with the ability to scale to the public cloud, you pay for extra computing power only when needed.
* Ease—transitioning to the cloud does not have to be overwhelming because you can migrate gradually—phasing in workloads over time.

# What is IaaS?

Infrastructure as a service (IaaS) is an instant computing infrastructure, provisioned and managed over the internet. It’s one of the four types of cloud services, along with software as a service ([SaaS](https://azure.microsoft.com/en-in/overview/what-is-saas/)), platform as a service ([PaaS](https://azure.microsoft.com/en-in/overview/what-is-paas/)), and [serverless](https://azure.microsoft.com/en-in/overview/serverless-computing/).

IaaS quickly scales up and down with demand, letting you pay only for what you use. It helps you avoid the expense and complexity of buying and managing your own physical servers and other datacenter infrastructure. Each resource is offered as a separate service component, and you only need to rent a particular one for as long as you need it. A [cloud computing service provider](https://azure.microsoft.com/en-in/overview/choosing-a-cloud-service-provider/), such as [Azure](https://azure.microsoft.com/en-in/overview/what-is-azure/iaas/), manages the infrastructure, while you purchase, install, configure, and manage your own software—operating systems, middleware, and applications.



# What is PaaS?

Platform as a service (PaaS) is a complete development and deployment environment in the cloud, with resources that enable you to deliver everything from simple cloud-based apps to sophisticated, cloud-enabled enterprise applications. You purchase the resources you need from a [cloud service provider](https://azure.microsoft.com/en-in/overview/choosing-a-cloud-service-provider/) on a pay-as-you-go basis and access them over a secure Internet connection.

Like [IaaS](https://azure.microsoft.com/en-in/overview/what-is-iaas/), PaaS includes infrastructure—servers, storage and networking—but also middleware, development tools, business intelligence (BI) services, database management systems and more. PaaS is designed to support the complete web application lifecycle: building, testing, deploying, managing and updating.

PaaS allows you to avoid the expense and complexity of buying and managing software licenses, the underlying application infrastructure and middleware, container orchestrators such as [Kubernetes](https://azure.microsoft.com/en-in/topic/what-is-kubernetes/) or the development tools and other resources. You manage the applications and services you develop and the cloud service provider typically manages everything else.

# What is SaaS?

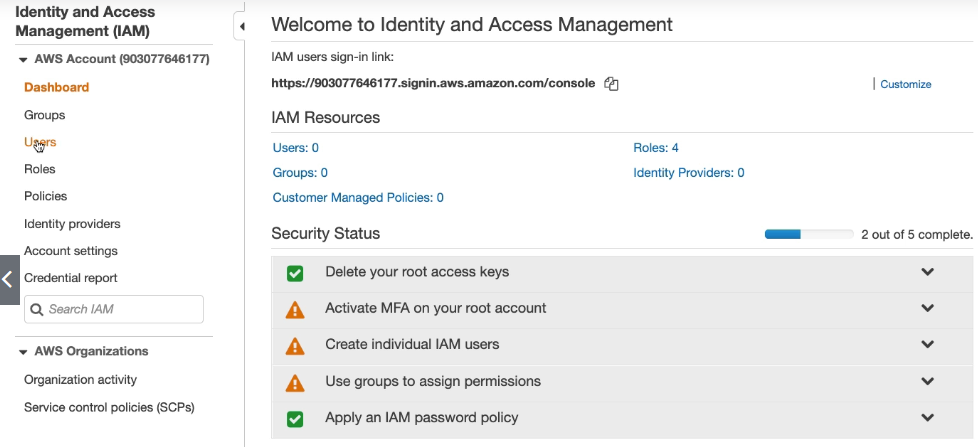
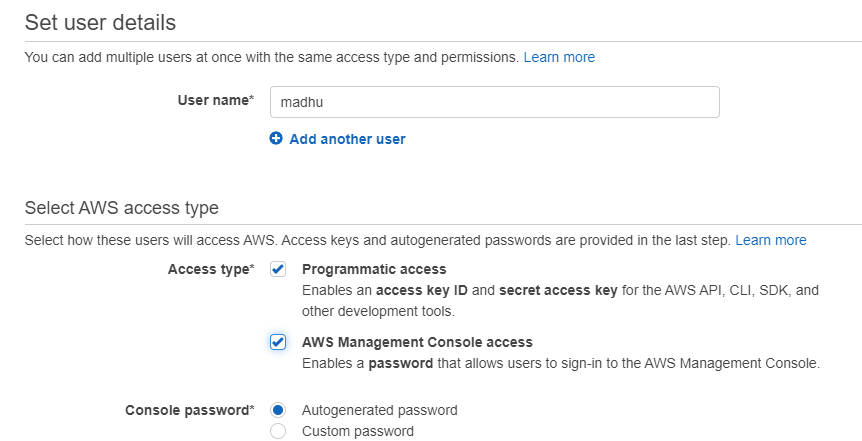
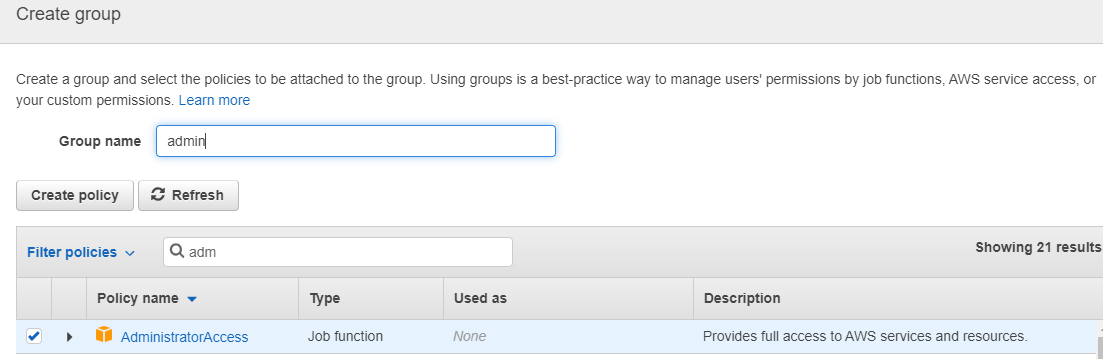
Software as a service (SaaS) allows users to connect to and use cloud-based apps over the Internet. SaaS provides a complete software solution which you purchase on a pay-as-you-go basis from a [cloud service provider](https://azure.microsoft.com/en-in/overview/choosing-a-cloud-service-provider/). You rent the use of an app for your organisation and your users connect to it over the Internet, usually with a web browser. All of the underlying infrastructure, middleware, app software and app data are located in the service provider’s data center. The service provider manages the hardware and software and with the appropriate service agreement, will ensure the availability and the security of the app and your data as well. SaaS allows your organisation to get quickly up and running with an app at minimal upfront cost.

# Identity and Access Management (IAM):

**AWS** Identity and Access Management (**IAM**) enables you to manage access to **AWS** services and resources securely. Using **IAM**, you can create and manage **AWS** users and groups, and use permissions to allow and deny their access to **AWS** resources. **IAM** is a feature of your **AWS** account offered at no additional charge.

# Using Multi-Factor Authentication (MFA) in AWS

MFA adds extra security because it requires users to provide unique authentication from an AWS supported MFA mechanism in addition to their regular sign-in credentials when they access AWS websites or services:

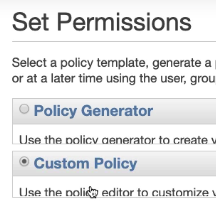
* **Virtual MFA devices**. A software app that runs on a phone or other device and emulates a physical device. The device generates a six-digit numeric code based upon a time-synchronized one-time password algorithm. The user must type a valid code from the device on a second webpage during sign-in. Each virtual MFA device assigned to a user must be unique.
* **U2F security key**. A device that you plug into a USB port on your computer. U2F is an open authentication standard hosted by the [FIDO Alliance](https://fidoalliance.org/). When you enable a U2F security key, you sign in by entering your credentials and then tapping the device instead of manually entering a code.
* **Hardware MFA device**. A hardware device that generates a six-digit numeric code based upon a time-synchronized one-time password algorithm. The user must type a valid code from the device on a second webpage during sign-in.
* **SMS text message-based MFA**. A type of MFA in which the IAM user settings include the phone number of the user's SMS-compatible mobile device. When the user signs in, AWS sends a six-digit numeric code by SMS text message to the user's mobile device. The user is required to type that code on a second webpage during sign-in.
* 
* 
* 
* AWS CLI connection:

Aws configure-it will ask user name and password for IAM user. and reason as well.

Ec2 describe instances |more –it will list of aws resources

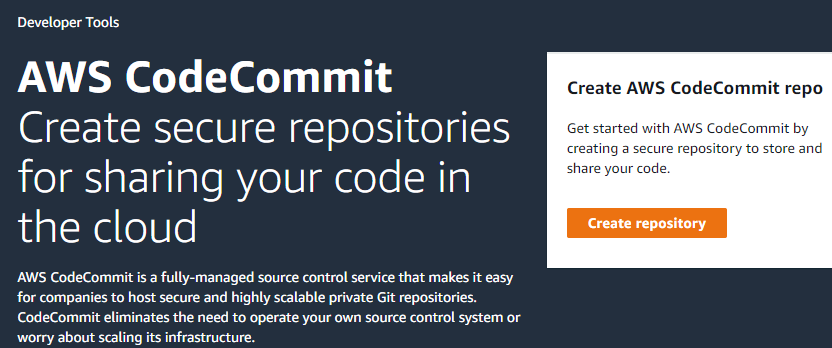
## Configure an IAM Policy to Limit Pushes and Merges to a Branch

You can create a policy in IAM that prevents users from updating a branch, including pushing commits to a branch and merging pull requests to a branch. To do this, your policy uses a conditional statement, so that the effect of the Deny statement applies only if the condition is met. The APIs you include in the Deny statement determine which actions are not allowed. You can configure this policy to apply to only one branch in a repository, a number of branches in a repository, or to all branches that match the criteria across all repositories in an AWS account.



Create new group and add custom policies and copy past the policies available in the aws codecommit documentation

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



{

"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:****\*:\*:\**",**

"Condition": {

"StringEqualsIfExists": {

"codecommit:References": [

"refs/heads/*master*"

]

},

"Null": {

"codecommit:References": false

}

}

}

]

}

And add user in the group and test working or not. Users-user-groups-add user to groups-select group and ok

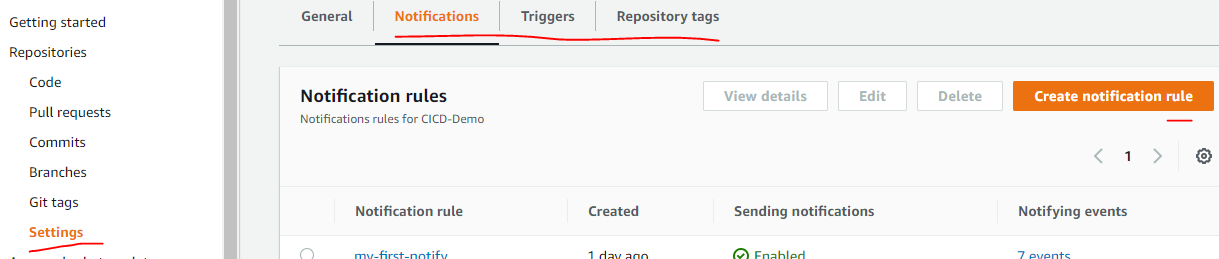
Then push code into master it will denied automatically

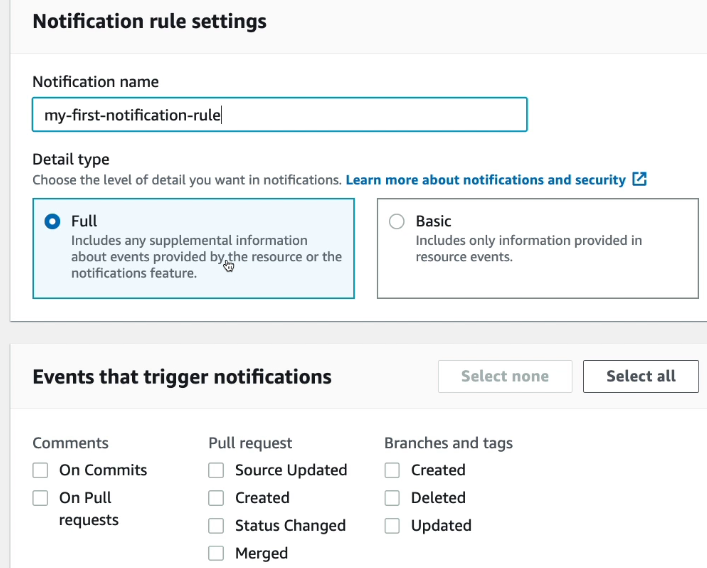
git branch -b feature –it will create new branch and push the code will work

git checkout feature

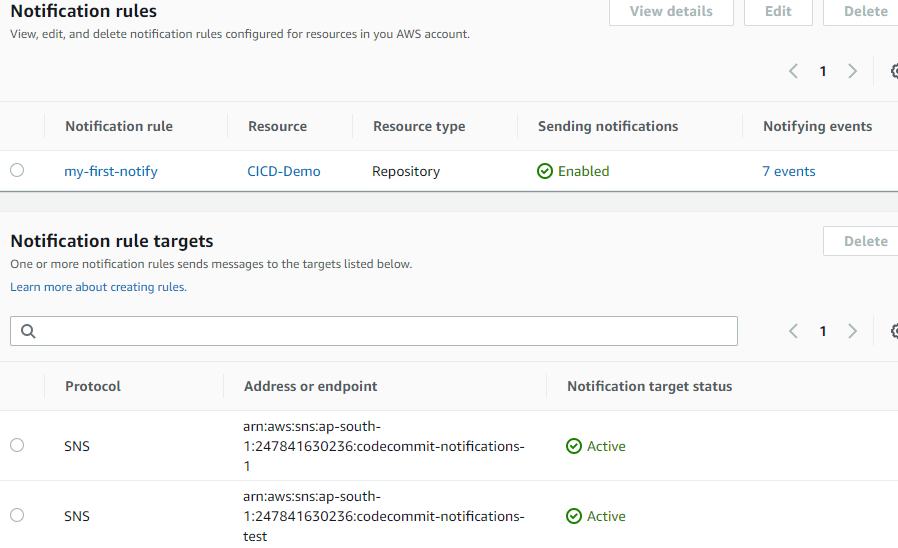
git push –set-upstream origin feature

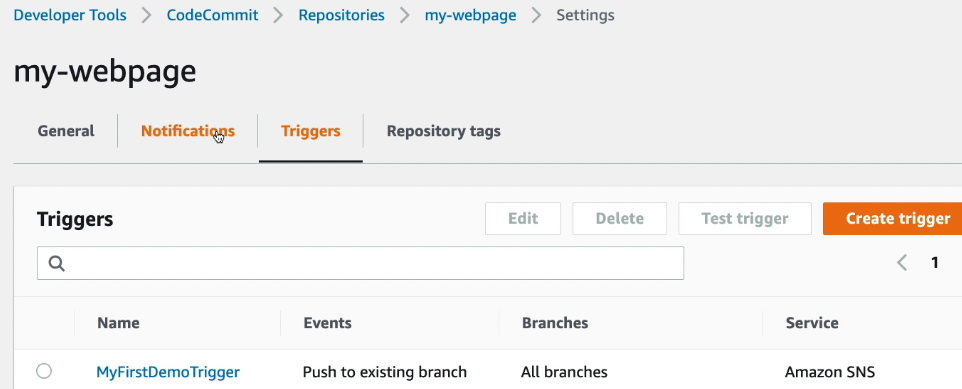
WEBHOOKS:



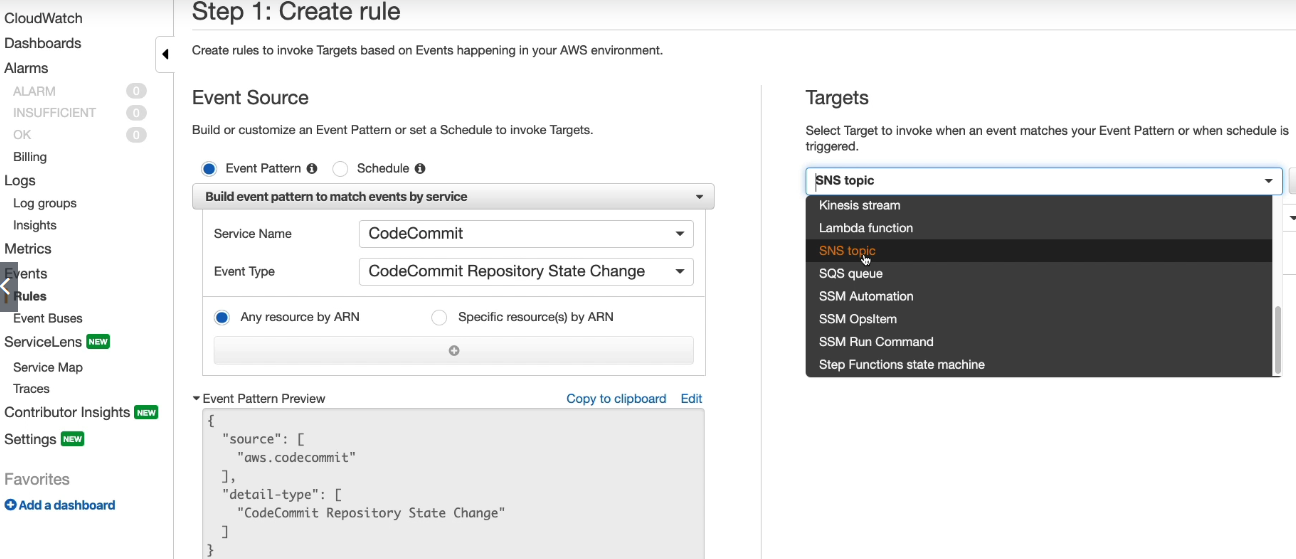


SNS topic will created and sent out the bucket notifications









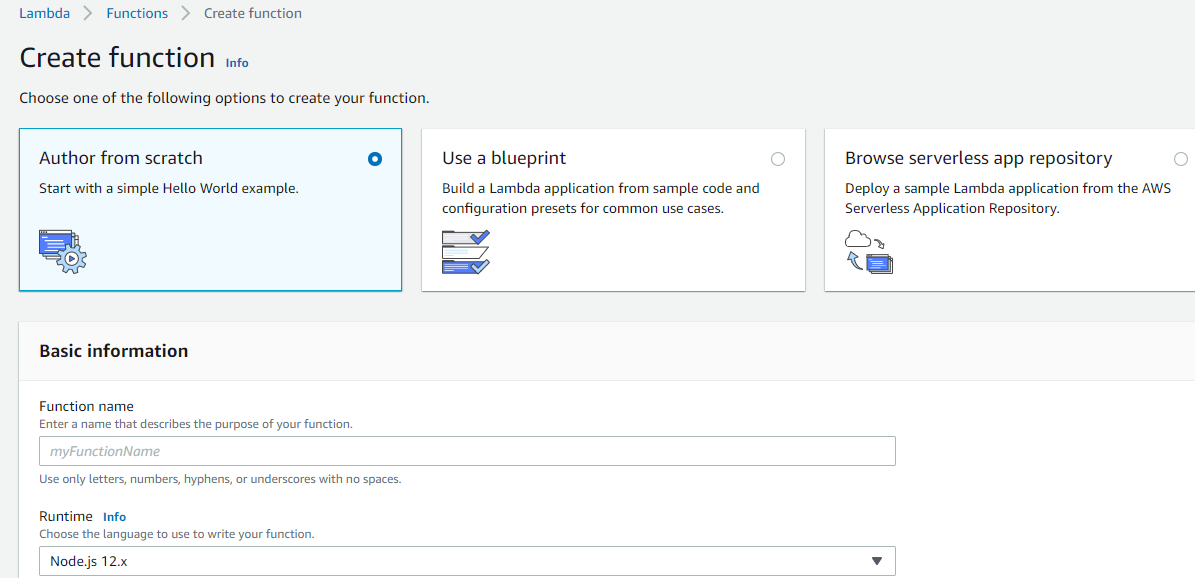
# **Create an AWS Code Commit Trigger for an AWS Lambda Function**

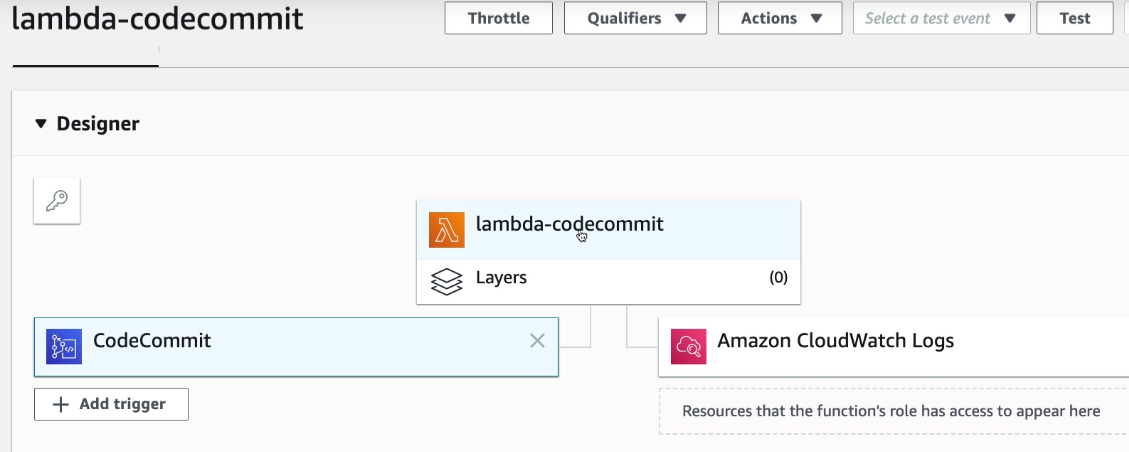
You can create a trigger for a Code Commit repository so that events in the repository invoke a Lambda function. In this example, you create a Lambda function that returns the URL used to clone the repository to an Amazon CloudWatch log.

## Create the Lambda Function

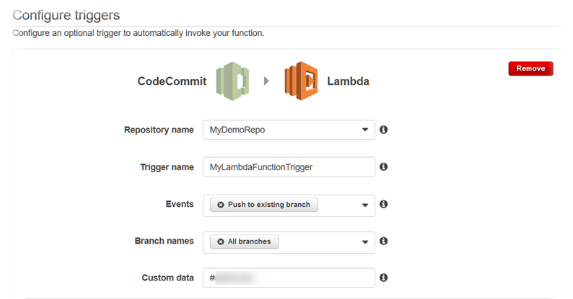
When you use the Lambda console to create the function, you can also create a Code Commit trigger for the Lambda function. The following steps include a sample Lambda function. The sample is available in two languages: JavaScript and Python. The function returns the URLs used for cloning a repository to a CloudWatch log.

Create a Lambda and add code commit for python code





Click Lambda-codecommit and add



Sample code: import json

def lambda\_handler(event, context):

# TODO implement

return {

'statusCode': 200,

'body': json.dumps('Hello from Lambda!')

}

Replace with below code: <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']

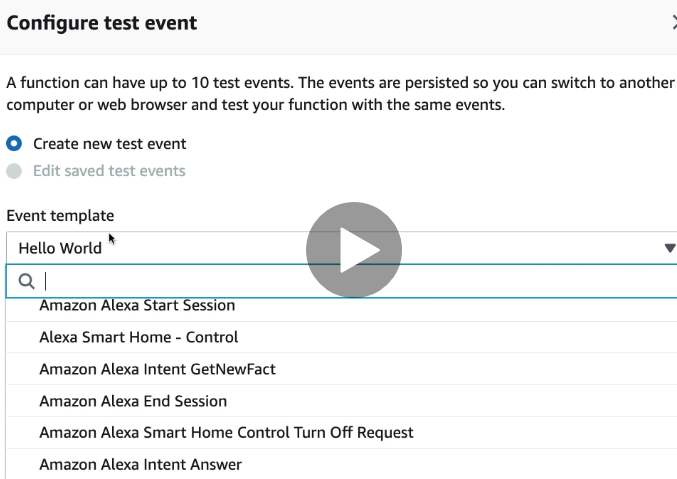
except Exception as e:

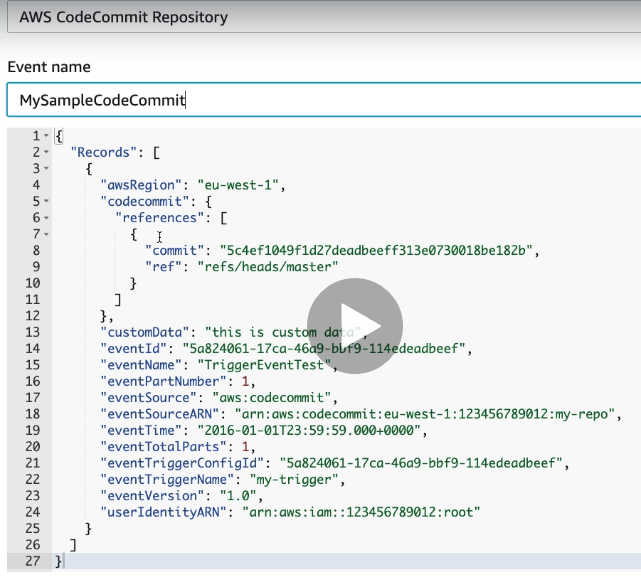
print(e)

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

raise e

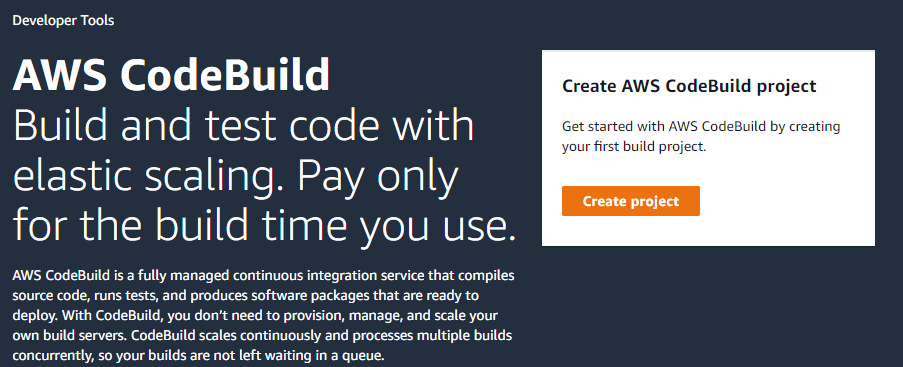
press TEST and run below as well.





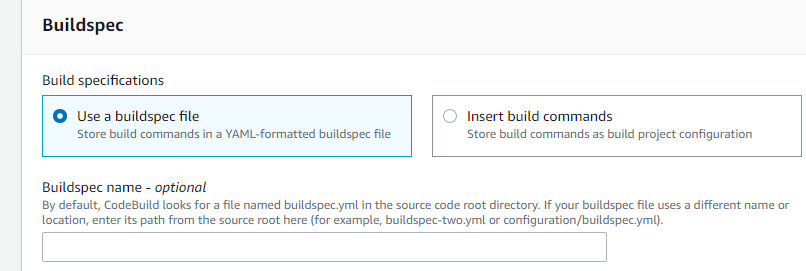
Then test it and it will be failed because of dummy branch and push and test the code commit and check in the cloud watch logs

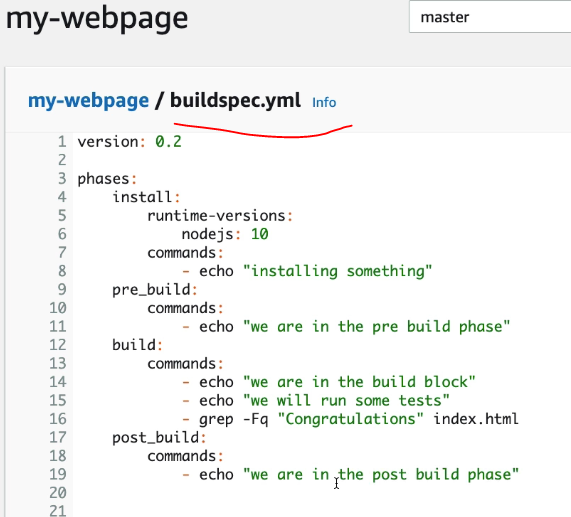




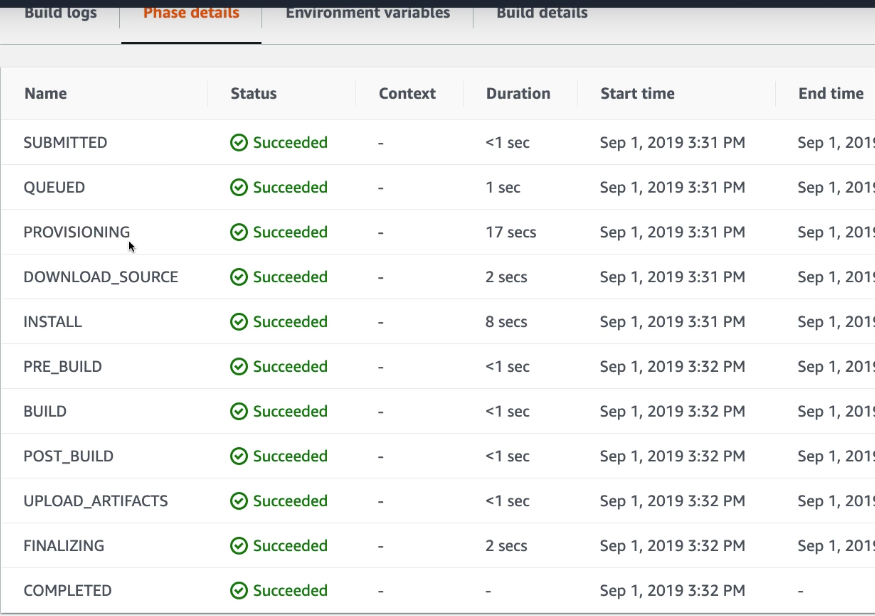
Create project and full fill the required details.

The job will run based on the buildsepc.yml based on the yml file syntaxes the job will run by order. Automatically will notify if it is in root directory and will build it.





Then start the build it will start using docker image and run build and CloudWatch will monitor logs will find the logs in the CloudWatch logs. based on the html file the job will successes.



BASIC OF SYNTAX OF builspec.yml:

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*

Example: DOCKER SAMPLE DEPLOY CODE

version: 0.2

env:

variables:

JAVA\_HOME: "/usr/lib/jvm/java-8-openjdk-amd64"

parameter-store:

LOGIN\_PASSWORD: /CodeBuild/dockerLoginPassword

phases:

install:

commands:

- echo Entered the install phase...

- apt-get update -y

- apt-get install -y maven

finally:

- echo This always runs even if the update or install command fails

pre\_build:

commands:

- echo Entered the pre\_build phase...

- docker login –u User –p $LOGIN\_PASSWORD

finally:

- echo This always runs even if the login command fails

build:

commands:

- echo Entered the build phase...

- echo Build started on `date`

- mvn install

finally:

- echo This always runs even if the install command fails

post\_build:

commands:

- echo Entered the post\_build phase...

- echo Build completed on `date`

reports:

arn:aws:codebuild:your-region:your-aws-account-id:report-group/report-group-name-1:

files:

- "\*\*/\*"

base-directory: 'target/tests/reports'

discard-paths: no

reportGroupCucumberJson:

files:

- 'cucumber/target/cucumber-tests.xml'

discard-paths: yes

file-format: CucumberJson *# default is JunitXml*

artifacts:

files:

- target/messageUtil-1.0.jar

discard-paths: yes

secondary-artifacts:

artifact1:

files:

- target/messageUtil-1.0.jar

discard-paths: yes

artifact2:

files:

- target/messageUtil-1.0.jar

discard-paths: yes

cache:

paths:

- '/root/.m2/\*\*/\*'

Example: DOCKER & ECR SAMPLE DEPLOY CODE

version: 0.2

phases:

install:

runtime-versions:

docker: 18

pre\_build:

commands:

- echo Logging in to Amazon ECR...

- $(aws ecr get-login --no-include-email --region $AWS\_DEFAULT\_REGION)

build:

commands:

- echo Build started on `date`

- echo Building the Docker image...

- docker build -t $IMAGE\_REPO\_NAME:$IMAGE\_TAG .

- docker tag $IMAGE\_REPO\_NAME:$IMAGE\_TAG $AWS\_ACCOUNT\_ID.dkr.ecr.$AWS\_DEFAULT\_REGION.amazonaws.com/$IMAGE\_REPO\_NAME:$IMAGE\_TAG

post\_build:

commands:

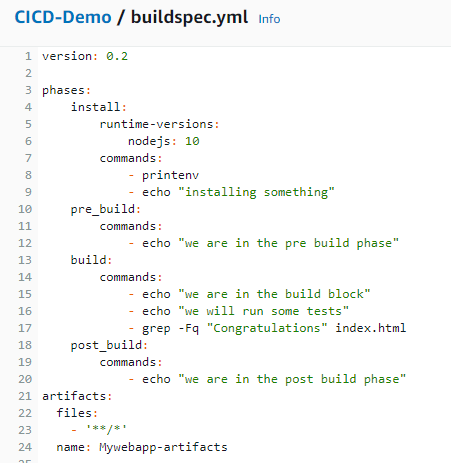
- echo Build completed on `date`

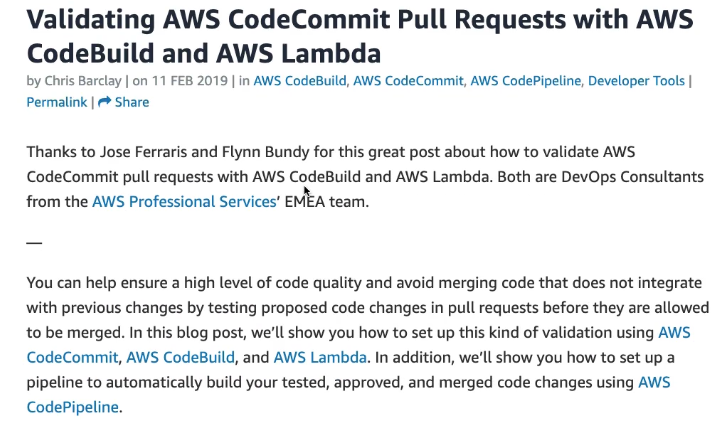
- echo Pushing the Docker image...

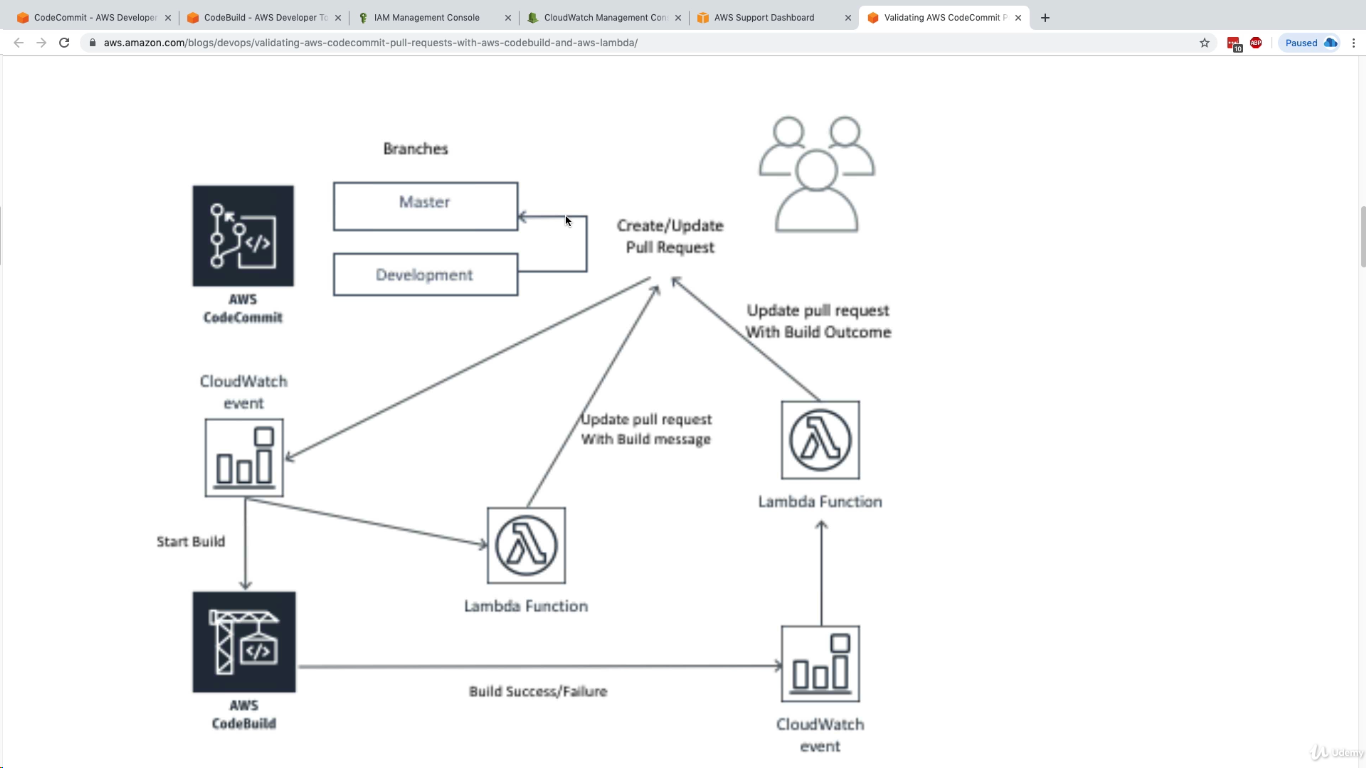
- docker push $AWS\_ACCOUNT\_ID.dkr.ecr.$AWS\_DEFAULT\_REGION.amazonaws.com/$IMAGE\_REPO\_NAME:$IMAGE\_TAG

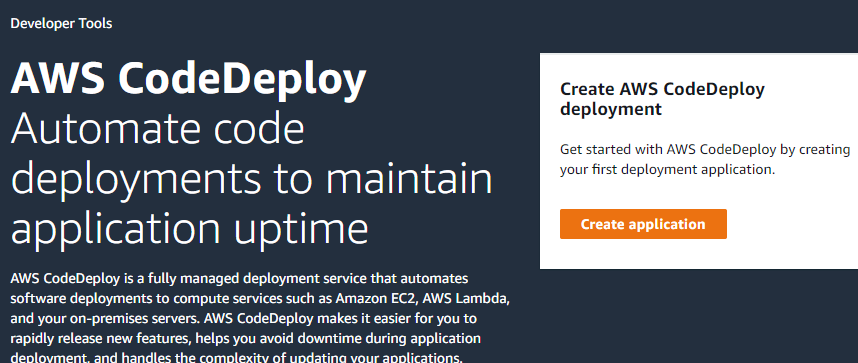
ENVIRONEMTAL VARIABLE TO ADD NEED TO ADD SYSTEM MANAGER NEED TO ADD THE USER NAME AND CREDENTIALS AND SAVE AND PASS IN THE CODEBUILD.BEFORE NEED TO RUN THE JOB NEED TO ADD THE USER POLACIES TO USER.

ADDING BUILD ARTIFACTS: I NTHE BUILDSPEC.YML FILE





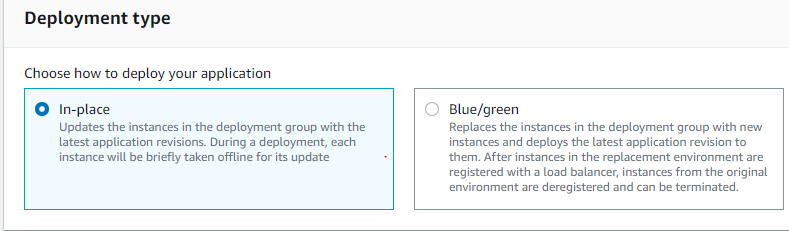


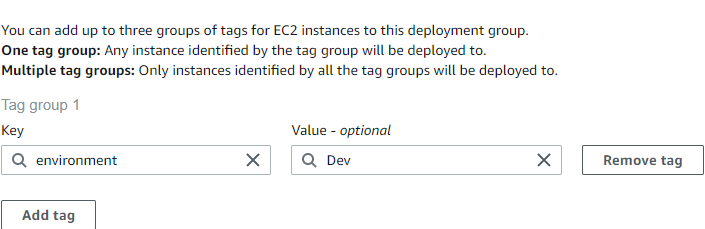


Before we are going deploy code, need to create ec2 instance and install below command:

|  |
| --- |
| [**CICD-Demo**](https://ap-south-1.console.aws.amazon.com/codesuite/codecommit/repositories/CICD-Demo/browse/refs/heads/master?region=ap-south-1)**/CODEDEPLOY.md file**  sudo yum update -y |
| sudo yum install -y ruby wget |
| wget https://aws-codedeploy-eu-west-1.s3.eu-west-1.amazonaws.com/latest/install |
| chmod +x ./install |
| sudo ./install auto |
| sudo service codedeploy-agent status |

Create IAM role for the codedeploy and create





Based on the ec2 instance tags will be deploy the code into ec2 instances.

**cd ~/.aws**

**vim credentials ==will find the profile name**

**Aws configure –profile madhureddy –it will login to aws command line**

**aws ec2 describe-instances --query Reservations[].Instances[].InstanceId --no-verify-ssl =it will display the ec2 instances hole information**

**aws ec2 describe-instances --query Reservations[].Instances[].InstanceId ==it will display** instance id’s only

# **create a bucket and enable versioning**

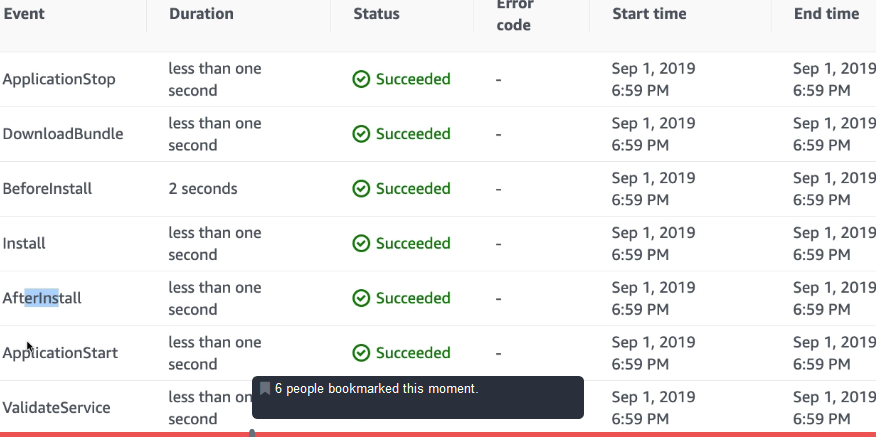
|  |
| --- |
| aws s3 mb s3://aws-devops-course-madhu1 --region eu-west-1 --profile  madhureddy --no-verify-ssl |
| aws s3api put-bucket-versioning --bucket aws-devops-course-madhu1 --versioning-  configuration Status=Enabled --region eu-west-1 --profile madhureddy --no-verify-ssl |

# **deploy the files into S3**

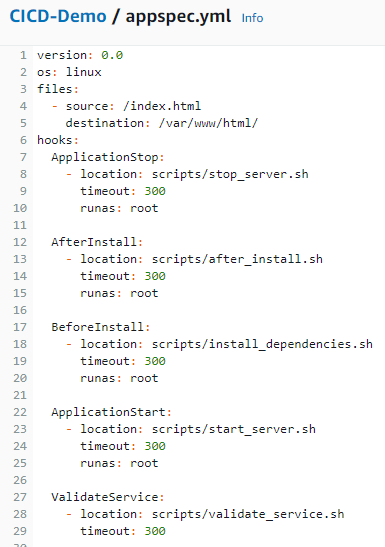
|  |
| --- |
| aws deploy push --application-name Codedeploydemo --s3-location  s3://aws-devops-course-madhu1/codedeploy-demo/app.zip --ignore-hidden-files  --region eu-west-1 --profile madhureddy --no-verify-ssl |

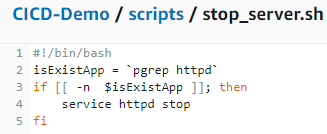
Verify the code is updated in the s3 or not

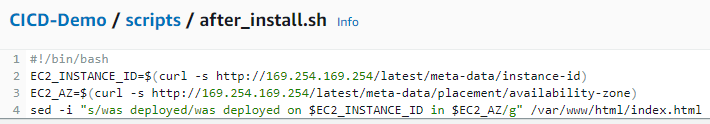
Then deploy the code into ec2 instance and verify the code using ec2 url.

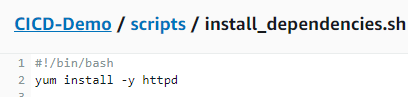


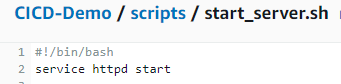
By using tags we can group the instance and deploy the code into multiple instance using CI/CD codedeploy. Based on the appspec.yml CI/CD will be running and start and stop the script for the deployment

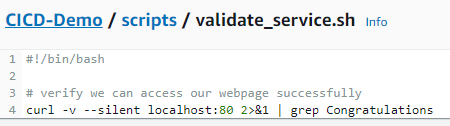












**INDEX.HTML file:Based on index.html file our code will be loaded into our webpage.**

<!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>1Congratulations v3</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>

# 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 Code Deploy 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 Code Deploy 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 Code Deploy

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 Behaviour 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.

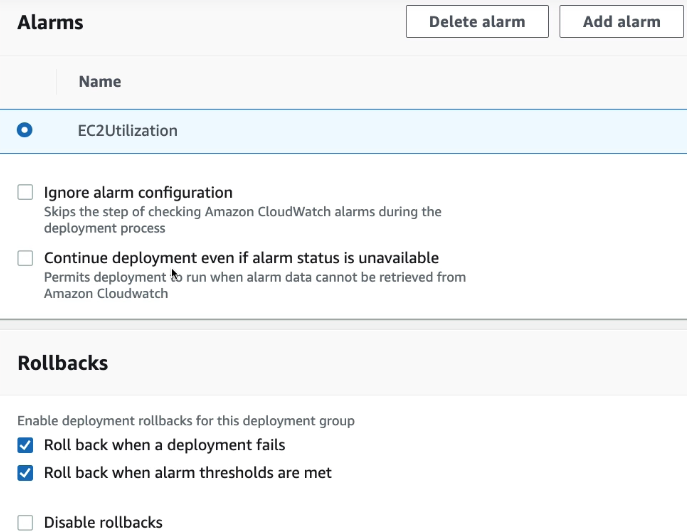
When you create a new deployment, you can also choose to override the automatic rollback configuration that were specified for the deployment group.

# 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 clean-up file:



# Working with On-Premises Instances for CodeDeploy

An on-premises instance is any physical device that is not an Amazon EC2 instance that can run the CodeDeploy agent and connect to public AWS service endpoints.

Deploying a CodeDeploy application revision to an on-premises instance involves two major steps:

Step 1 – Configure each on-premises instance, register it with CodeDeploy, and then tag it.

Step 2 – Deploy application revisions to the on-premises instance.

If you don't want an on-premises instance to be used in deployments anymore, you can simply remove the on-premises instance tags from the deployment groups. For a more robust approach, remove the on-premises instance tags from the instance. You can also explicitly deregister an on-premises instance so it can no longer be used in any deployments. For more information, see [Managing On-Premises Instances Operations in CodeDeploy](https://docs.aws.amazon.com/codedeploy/latest/userguide/on-premises-instances-operations.html).

The instructions in this section show you how to configure an on-premises instance and then register and tag it with CodeDeploy so it can be used in deployments. This section also describes how to use CodeDeploy to get information about on-premises instances and deregister an on-premises instance after you're no longer planning to deploy to it.

To register an on-premises instance, you must use an IAM identity to authenticate your requests. You can choose from the following options for the IAM identity and registration method you use:

Use an IAM user ARN to authenticate requests.

Use the [register](https://docs.aws.amazon.com/cli/latest/reference/deploy/register.html) command for the most automated registration process. Best for registering a single on-premises instance. For information, see [Use the register Command (IAM User ARN) to Register an On-Premises Instance](https://docs.aws.amazon.com/codedeploy/latest/userguide/instances-on-premises-register-instance.html).

Use the [register-on-premises-instance](https://docs.aws.amazon.com/cli/latest/reference/deploy/register-on-premises-instance.html) command to manually configure most registration options. Suitable for registering a small number of on-premises instances. For information, see [Use the register-on-premises-instance Command (IAM User ARN) to Register an On-Premises Instance](https://docs.aws.amazon.com/codedeploy/latest/userguide/register-on-premises-instance-iam-user-arn.html).

Use an IAM role ARN to authenticate requests.

Use the [register-on-premises-instance](https://docs.aws.amazon.com/cli/latest/reference/deploy/register-on-premises-instance.html) command and periodically refreshed temporary credentials generated with the AWS Security Token Service (AWS STS) to manually configure most registration options. Best for registering a large number of on-premises instances. For information, see [Use the register-on-premises-instance Command (IAM Session ARN) to Register an On-Premises Instance](https://docs.aws.amazon.com/codedeploy/latest/userguide/register-on-premises-instance-iam-session-arn.html).

# 1: Create an IAM User for the On-Premises Instance

Create an IAM user that the on-premises instance will use to authenticate and interact with CodeDeploy.

You must create a separate IAM user for each participating on-premises instance. If you try to reuse an individual IAM user for multiple on-premises instances, you might not be able to successfully register or tag those on-premises instances with CodeDeploy. Deployments to those on-premises instances might be stuck in a perpetual pending state or fail altogether.

We recommed that you assign the IAM user a name that identifies its purpose, such as CodeDeployUser-OnPrem.

You can use the AWS CLI or the IAM console to create an IAM user. For information, see [Creating an IAM User in Your AWS Account](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users_create.html).

# 2: Assign Permissions to the IAM User

If your on-premises instance will be deploying application revisions from Amazon S3 buckets, you must assign to the IAM user the permissions to interact with those buckets. You can use the AWS CLI or the IAM console to assign permissions.

{

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

"Statement": [

{

"Action": [

"s3:Get\*",

"s3:List\*"

],

"Effect": "Allow",

"Resource": "\*"

}

]

}

Use a text editor to add the following information to the newly created codedeploy.onpremises.yml or conf.onpremises.yml file:

---

aws\_access\_key\_id: *secret-key-id*

aws\_secret\_access\_key: *secret-access-key*

iam\_user\_arn: *iam-user-arn*

region: *supported-region*

# Step 9: Tag the On-Premises Instance

You can use either the AWS CLI or the CodeDeploy console to tag the on-premises instance. (CodeDeploy uses on-premises instance tags to identify the deployment targets during a deployment.)

To tag the on-premises instance (CLI)

Call the [add-tags-to-on-premises-instances](https://docs.aws.amazon.com/cli/latest/reference/deploy/add-tags-to-on-premises-instances.html) command, specifying:

The name that uniquely identifies the on-premises instance (with the --instance-names option).

The name of the on-premises instance tag key and tag value you want to use (with the --tags option). You must specify both a name and value. CodeDeploy does not allow on-premises instance tags that have values only.

For example:

aws deploy add-tags-to-on-premises-instances --instance-names AssetTag12010298EX --tags Key=Name,Value=CodeDeployDemo-OnPrem

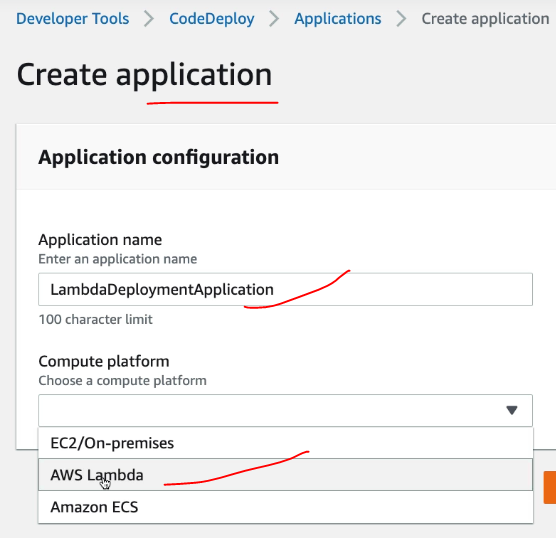
# Deployment Configurations on an Amazon ECS Compute Platform

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.

# LAMBDA

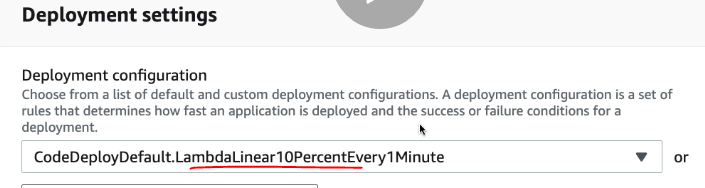


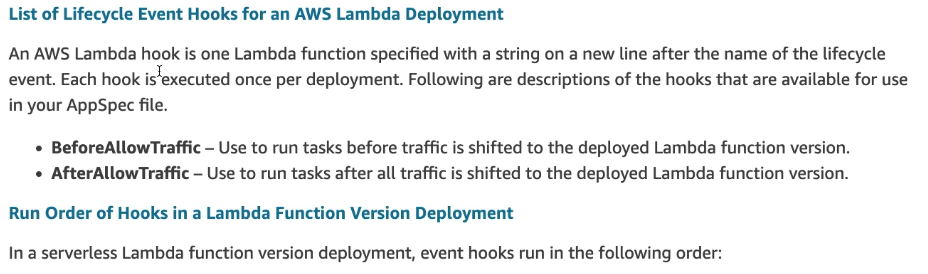
# Deployment Configurations on an AWS Lambda Compute Platform

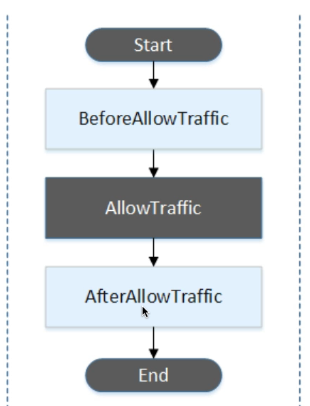
When you deploy to an AWS Lambda compute platform, the deployment configuration specifies the way traffic is shifted to the new Lambda function versions in your application.

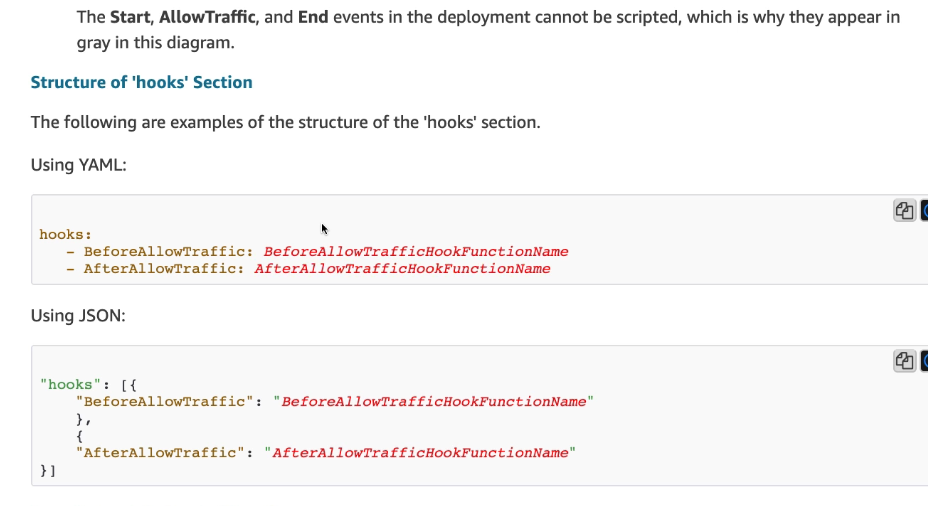
There are three ways traffic can shift during a deployment:

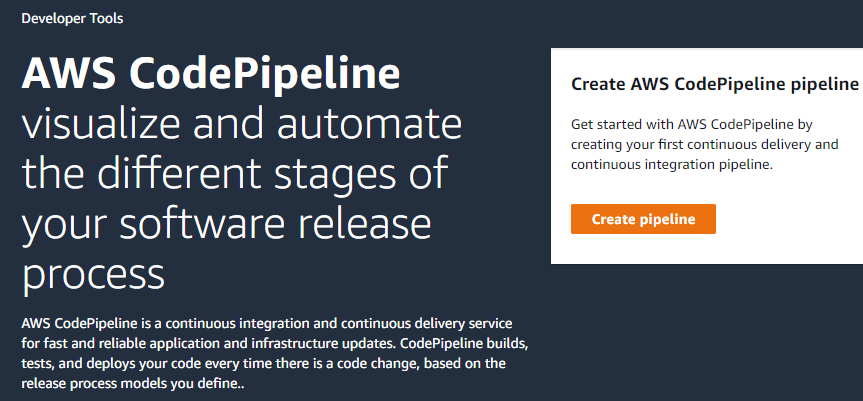
* **Canary**: Traffic is shifted in two increments. You can choose from predefined canary options that specify the percentage of traffic shifted to your updated Lambda function version in the first increment and the interval, in minutes, before the remaining traffic is shifted in the second increment.
* **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 Lambda function to the updated Lambda function version all at once.

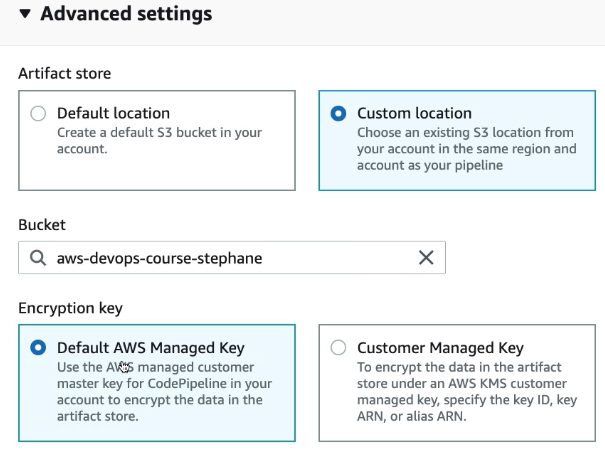


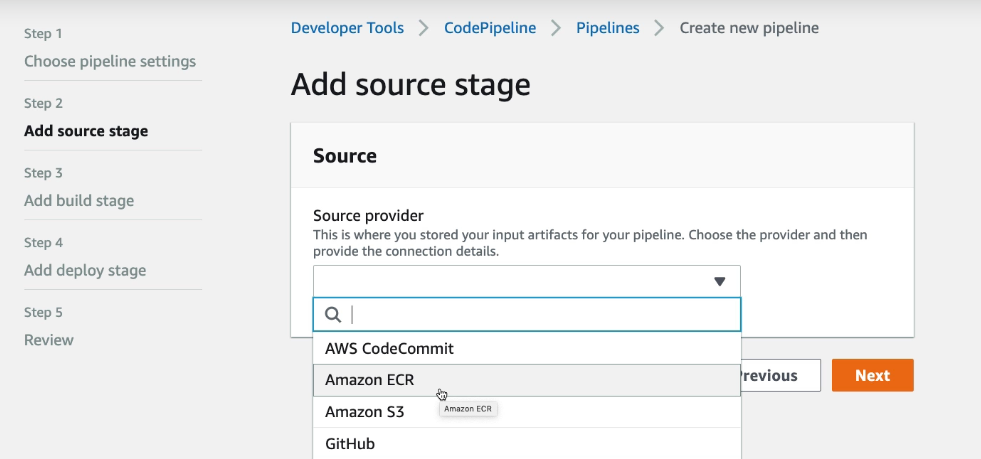


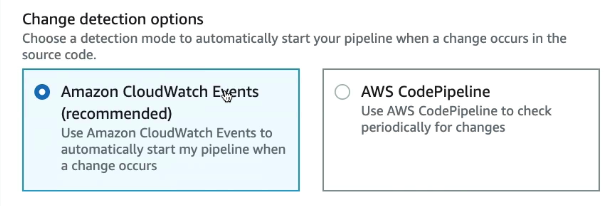


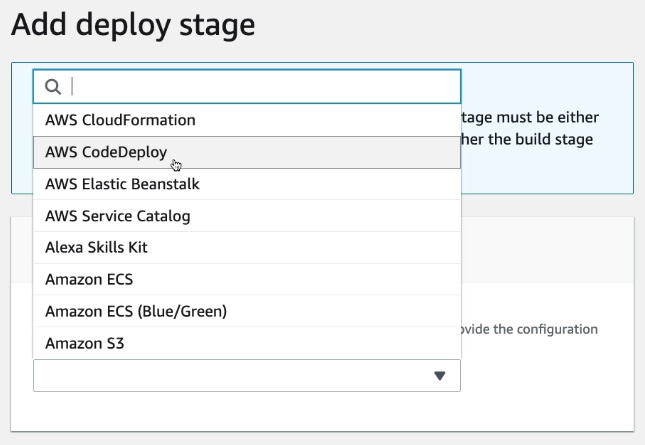


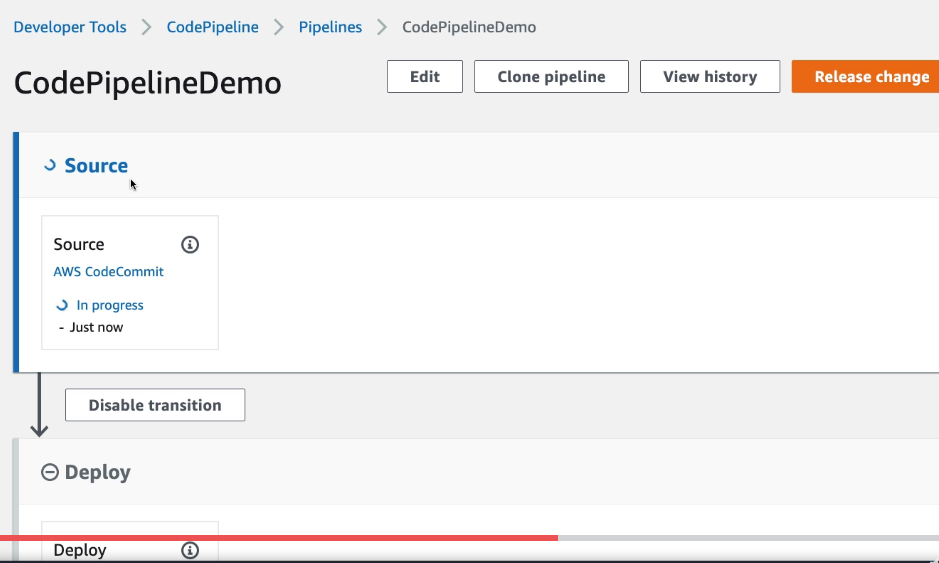


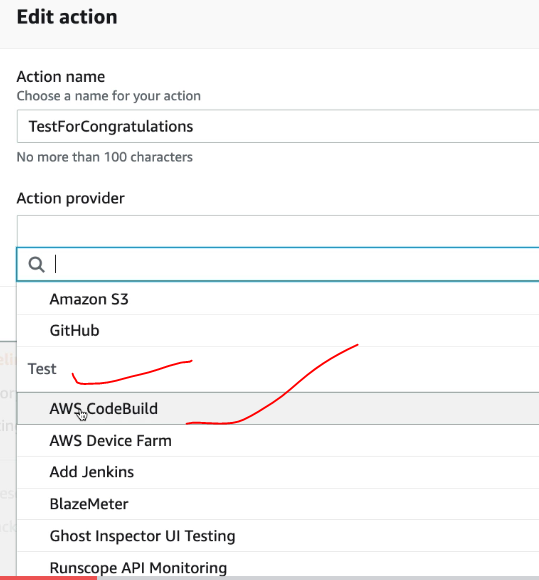


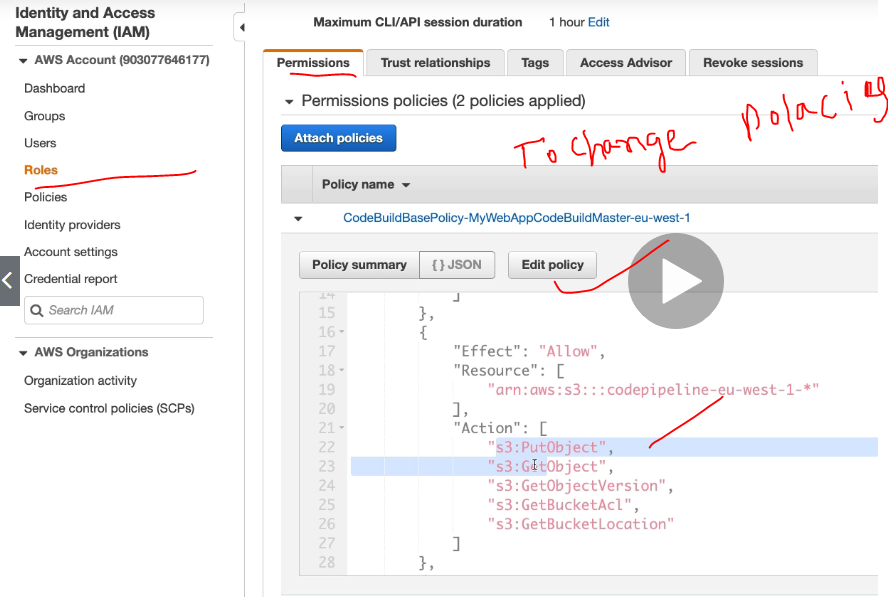


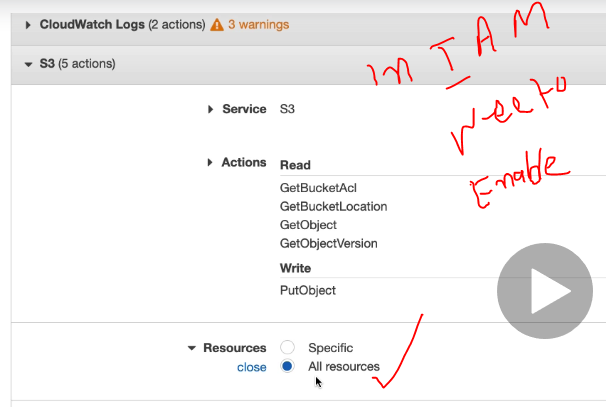


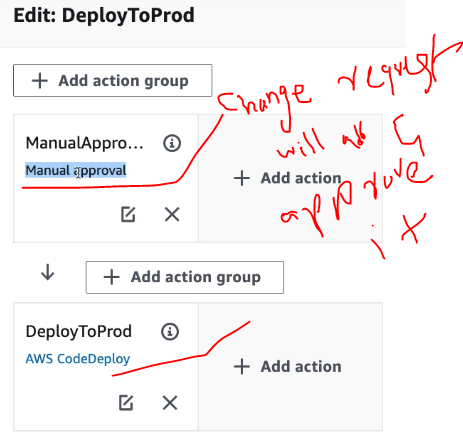


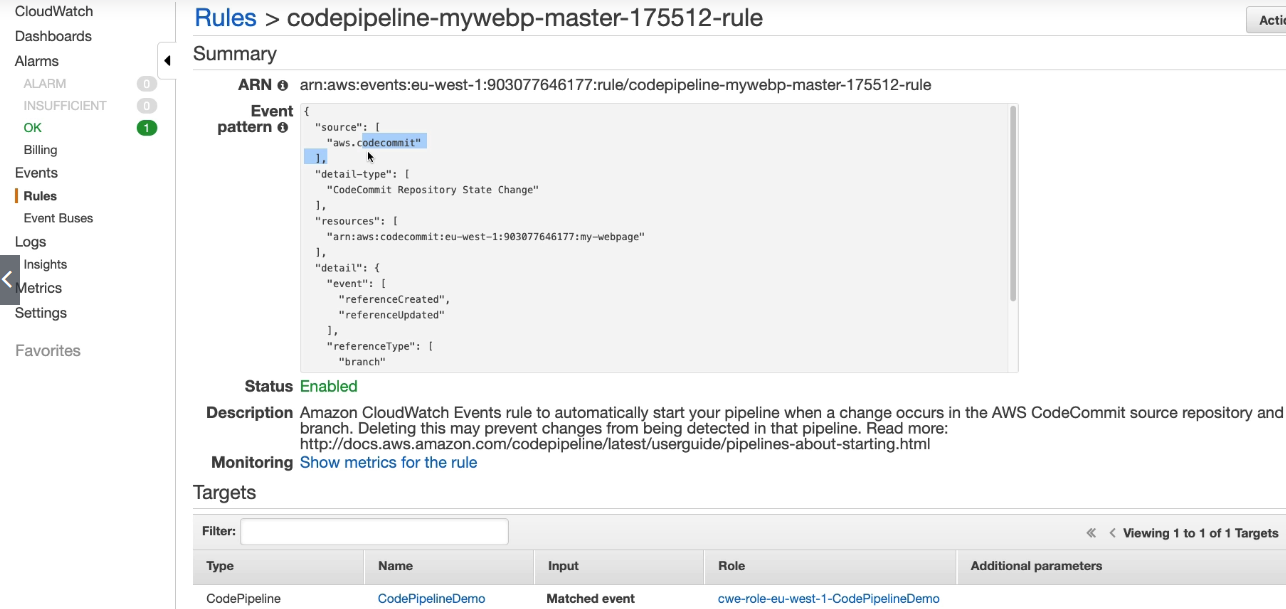


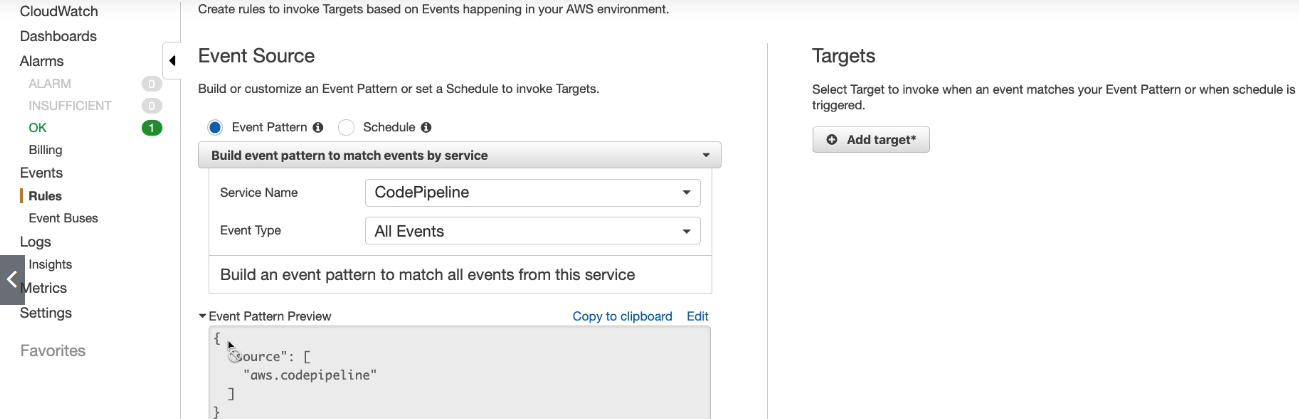












* [Use CodePipeline with Amazon S3, AWS CodeCommit, and AWS CodeDeploy](https://docs.aws.amazon.com/codepipeline/latest/userguide/best-practices.html#use-cases-S3-codedeploy)
* [Use CodePipeline with Third-party Action Providers (GitHub and Jenkins)](https://docs.aws.amazon.com/codepipeline/latest/userguide/best-practices.html#use-cases-thirdparty)
* [Use CodePipeline with AWS CodeStar to Build a Pipeline in a Code Project](https://docs.aws.amazon.com/codepipeline/latest/userguide/best-practices.html#use-cases-codestar)
* [Use CodePipeline to Compile, Build, and Test Code with CodeBuild](https://docs.aws.amazon.com/codepipeline/latest/userguide/best-practices.html#use-cases-codebuild)
* [Use CodePipeline with Amazon ECS for Continuous Delivery of Container-Based Applications to the Cloud](https://docs.aws.amazon.com/codepipeline/latest/userguide/best-practices.html#use-cases-ecs)
* [Use CodePipeline with Elastic Beanstalk for Continuous Delivery of Web Applications to the Cloud](https://docs.aws.amazon.com/codepipeline/latest/userguide/best-practices.html#use-cases-elasticbeanstalk)
* [Use CodePipeline with AWS Lambda for Continuous Delivery of Lambda-Based and Serverless Applications](https://docs.aws.amazon.com/codepipeline/latest/userguide/best-practices.html#use-cases-lambda)
* [Use CodePipeline with AWS CloudFormation Templates for Continuous Delivery to the Cloud](https://docs.aws.amazon.com/codepipeline/latest/userguide/best-practices.html#use-cases-cloudformation)

# **Invoke an AWS Lambda Function in a Pipeline in CodePipeline**

<https://docs.aws.amazon.com/codepipeline/latest/userguide/actions-invoke-lambda-function.html>

[AWS Lambda](https://docs.aws.amazon.com/lambda/latest/dg/) is a compute service that lets you run code without provisioning or managing servers. You can create Lambda functions and add them as actions in your pipelines. Because Lambda allows you to write functions to perform almost any task, you can customize the way your pipeline works.

Here are some ways Lambda functions can be used in pipelines:

* To roll out changes to your environment by applying or updating an AWS CloudFormation template.
* To create resources on demand in one stage of a pipeline using AWS CloudFormation and delete them in another stage.
* To deploy application versions with zero downtime in AWS Elastic Beanstalk with a Lambda function that swaps CNAME values.
* To deploy to Amazon ECS Docker instances.
* To back up resources before building or deploying by creating an AMI snapshot.
* To add integration with third-party products to your pipeline, such as posting messages to an IRC client.

This topic assumes you are familiar with AWS CodePipeline and AWS Lambda and know how to create pipelines, functions, and the IAM policies and roles on which they depend. This topic shows you how to:

* Create a Lambda function that tests whether a webpage was deployed successfully.
* Configure the CodePipeline and Lambda execution roles and the permissions required to run the function as part of the pipeline.
* Edit a pipeline to add the Lambda function as an action.
* Test the action by manually releasing a change.

This topic includes sample functions to demonstrate the flexibility of working with Lambda functions in CodePipeline:

* [Basic Lambda function](https://docs.aws.amazon.com/codepipeline/latest/userguide/actions-invoke-lambda-function.html#LambdaSample1)
  + Creating a basic Lambda function to use with CodePipeline.
  + Returning success or failure results to CodePipeline in the **Details** link for the action.
* [Sample Python Function That Uses an AWS CloudFormation Template](https://docs.aws.amazon.com/codepipeline/latest/userguide/actions-invoke-lambda-function.html#actions-invoke-lambda-function-samples-python-cloudformation)
  + Using JSON-encoded user parameters to pass multiple configuration values to the function (get\_user\_params).
  + Interacting with .zip artifacts in an artifact bucket (get\_template).
  + Using a continuation token to monitor a long-running asynchronous process (continue\_job\_later). This allows the action to continue and the function to succeed even if it exceeds a fifteen-minute runtime (a limit in Lambda).

Each sample function includes information about the permissions you must add to the role. For information about limits in AWS Lambda, see [Limits](https://docs.aws.amazon.com/lambda/latest/dg/limits.html) in the AWS Lambda Developer Guide.

## Step 2: Create the Lambda Function

In this step, you create a Lambda function that makes an HTTP request and checks for a line of text on a webpage. As part of this step, you must also create an IAM policy and Lambda execution role. For more information, see [Permissions Model](https://docs.aws.amazon.com/lambda/latest/dg/intro-permission-model.html#lambda-intro-execution-role) in the AWS Lambda Developer Guide.

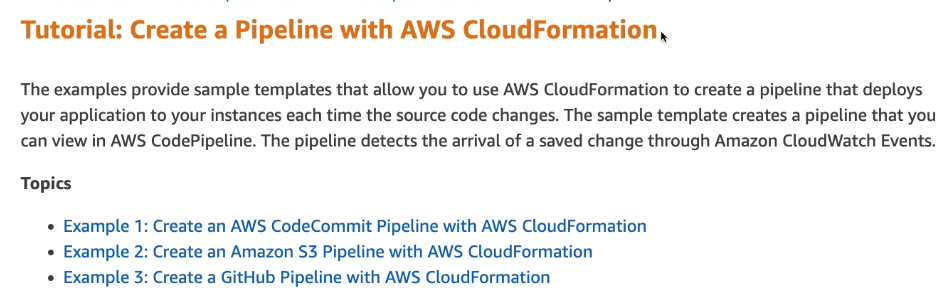
**To create the execution role**

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. Choose **Policies**, and then choose **Create Policy**. Choose the **JSON** tab, and then paste the following policy into the field.
3. {
4. "Version": "2012-10-17",
5. "Statement": [
6. {
7. "Action": [
8. "logs:\*"
9. ],
10. "Effect": "Allow",
11. "Resource": "arn:aws:logs:\*:\*:\*"
12. },
13. {
14. "Action": [
15. "codepipeline:PutJobSuccessResult",
16. "codepipeline:PutJobFailureResult"
17. ],
18. "Effect": "Allow",
19. "Resource": "\*"
20. }
21. ]
22. }
23. Choose **Review policy**.

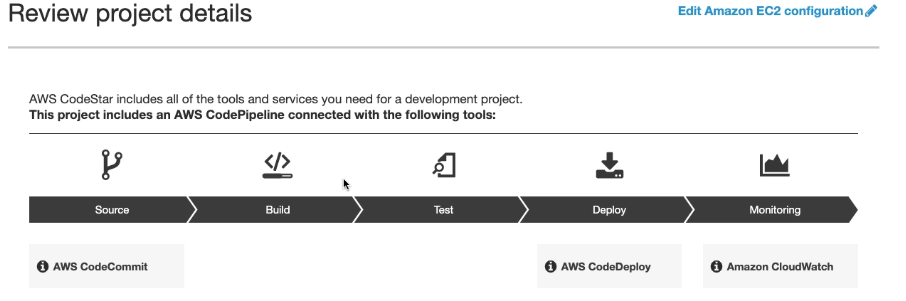
The above policy need to attached to lambda IAM role and will work with any issues.

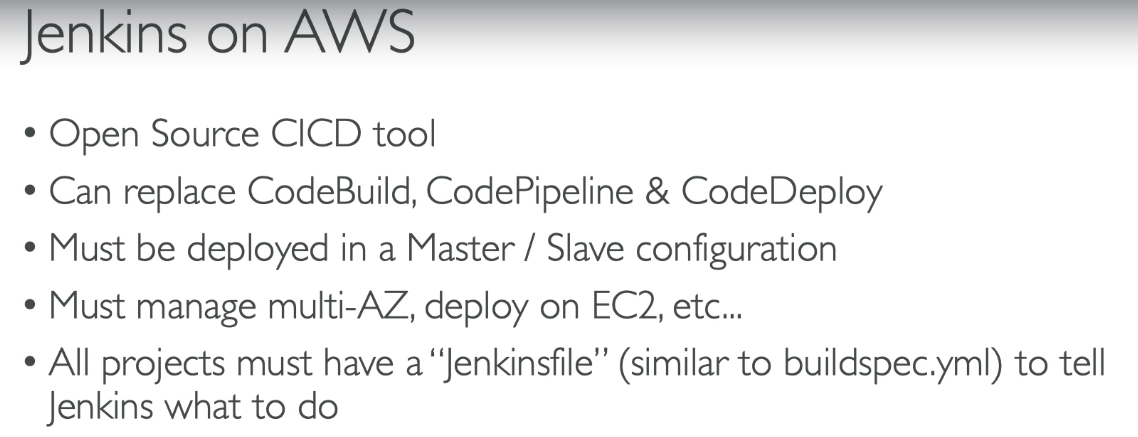
Below policy will mange API call and all the Lambda functionality.

1. var assert = require('assert');
2. var AWS = require('aws-sdk');
3. var http = require('http');
4. exports.handler = function(event, context) {
5. var codepipeline = new AWS.CodePipeline();
7. *// Retrieve the Job ID from the Lambda action*
8. var jobId = event["CodePipeline.job"].id;
10. *// Retrieve the value of UserParameters from the Lambda action configuration in AWS CodePipeline, in this case a URL which will be*
11. *// health checked by this function.*
12. var url = event["CodePipeline.job"].data.actionConfiguration.configuration.UserParameters;
14. *// Notify AWS CodePipeline of a successful job*
15. var putJobSuccess = function(message) {
16. var params = {
17. jobId: jobId
18. };
19. codepipeline.putJobSuccessResult(params, function(err, data) {
20. if(err) {
21. context.fail(err);
22. } else {
23. context.succeed(message);
24. }
25. });
26. };
28. *// Notify AWS CodePipeline of a failed job*
29. var putJobFailure = function(message) {
30. var params = {
31. jobId: jobId,
32. failureDetails: {
33. message: JSON.stringify(message),
34. type: 'JobFailed',
35. externalExecutionId: context.invokeid
36. }
37. };
38. codepipeline.putJobFailureResult(params, function(err, data) {
39. context.fail(message);
40. });
41. };
43. *// Validate the URL passed in UserParameters*
44. if(!url || url.indexOf('http://') === -1) {
45. putJobFailure('The UserParameters field must contain a valid URL address to test, including http:// or https://');
46. return;
47. }
49. *// Helper function to make a HTTP GET request to the page.*
50. *// The helper will test the response and succeed or fail the job accordingly*
51. var getPage = function(url, callback) {
52. var pageObject = {
53. body: '',
54. statusCode: 0,
55. contains: function(search) {
56. return this.body.indexOf(search) > -1;
57. }
58. };
59. http.get(url, function(response) {
60. pageObject.body = '';
61. pageObject.statusCode = response.statusCode;
63. response.on('data', function (chunk) {
64. pageObject.body += chunk;
65. });
67. response.on('end', function () {
68. callback(pageObject);
69. });
71. response.resume();
72. }).on('error', function(error) {
73. *// Fail the job if our request failed*
74. putJobFailure(error);
75. });
76. };
78. getPage(url, function(returnedPage) {
79. try {
80. *// Check if the HTTP response has a 200 status*
81. assert(returnedPage.statusCode === 200);
82. *// Check if the page contains the text "Congratulations"*
83. *// You can change this to check for different text, or add other tests as required*
84. assert(returnedPage.contains('Congratulations'));
86. *// Succeed the job*
87. putJobSuccess("Tests passed.");
88. } catch (ex) {
89. *// If any of the assertions failed then fail the job*
90. putJobFailure(ex);
91. }
92. });
93. };

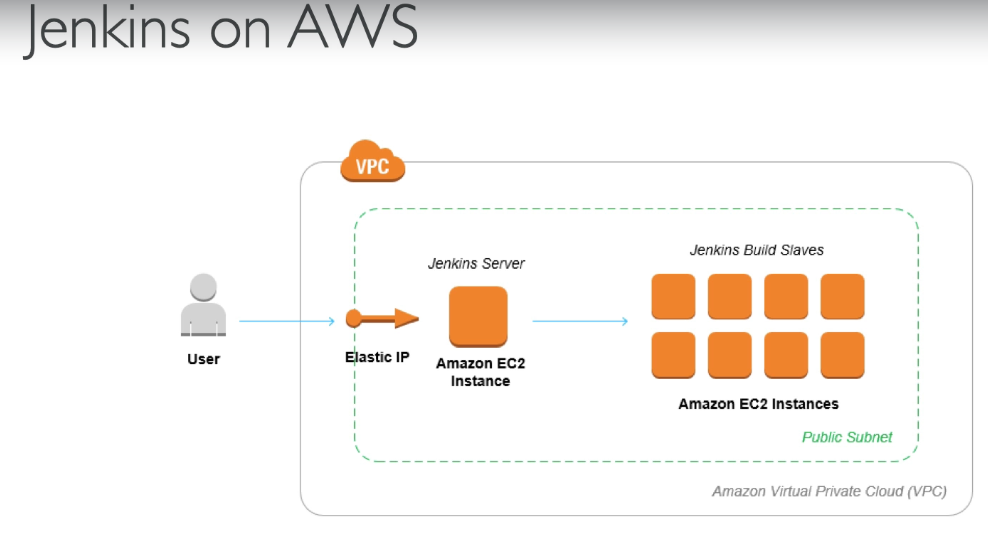




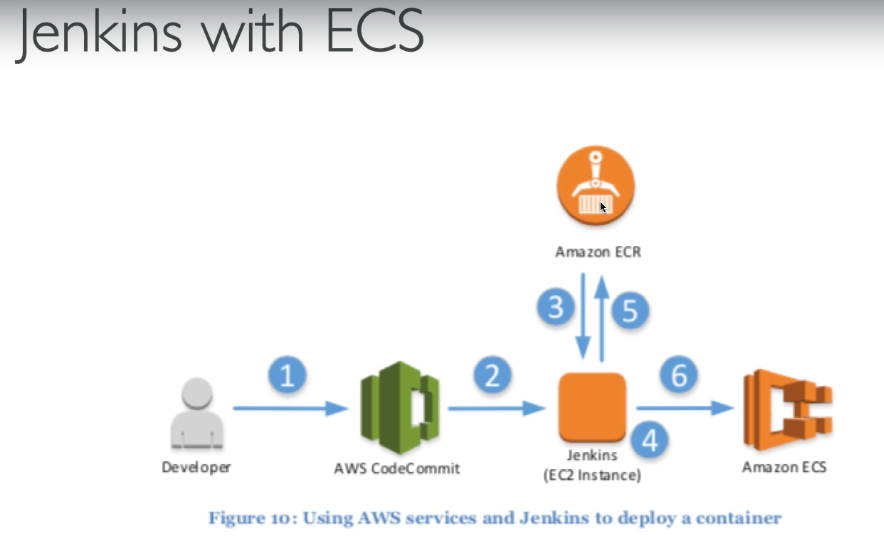


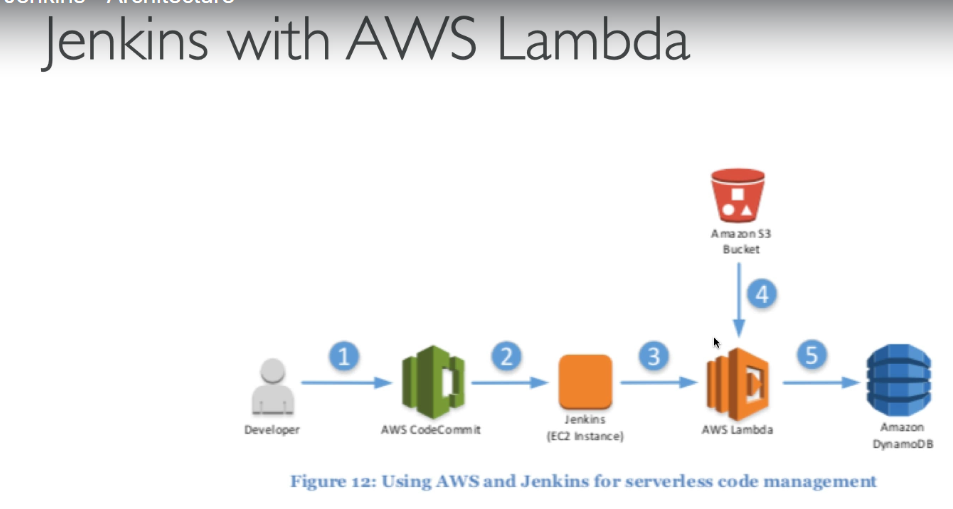


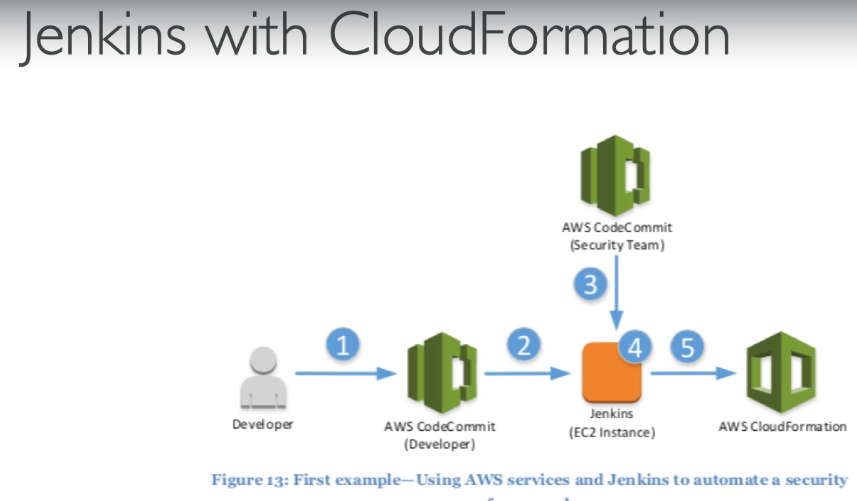




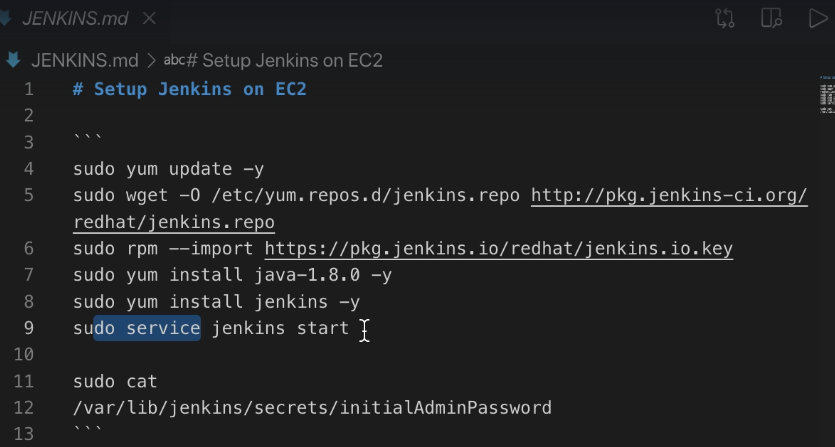




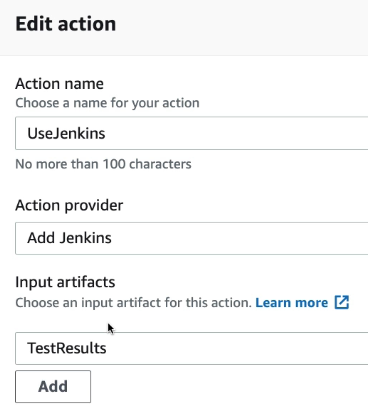




Create Jenkins instance and run the below commands:



Add Jenkins in the AWS pipelines in the edit actions:



[Amazon EC2](https://plugins.jenkins.io/ec2)

This plugin integrates Jenkins with [Amazon EC2](http://aws.amazon.com/ec2/) or anything implementing the EC2 API's such as an Ubuntu.

With this plugin, if Jenkins notices that your build cluster is overloaded, it'll start instances using the EC2 API and automatically connect them as Jenkins agents. When the load goes down, excess EC2 instances will be terminated. This set up allows you to maintain a small in-house cluster, then spill the spiky build/test loads into EC2 or another EC2 compatible cloud.

[AWS CodeBuild](https://plugins.jenkins.io/aws-codebuild)

Build your project on AWS CodeBuild. [AWS CodeBuild](https://aws.amazon.com/codebuild/) is a fully managed build 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. You can get started quickly by using prepackaged build environments, or you can create custom build environments that use your own build tools. With CodeBuild, you are charged by the minute for the compute resources you use.

This is the official plugin for AWS CodeBuild that's managed by CodeBuild engineering team. You can reach out to us for submitting feedback or feature requests at codebuild-feedback@amazon.com.

[**Amazon Elastic Container Service (ECS) / Fargate**](https://plugins.jenkins.io/amazon-ecs)

Use Amazon EC2 Container Service to provide elastic agents.This Jenkins plugin uses [Amazon Elastic Container Service](http://docs.aws.amazon.com/AmazonECS/latest/developerguide/Welcome.html) to host jobs execution inside docker containers.Jenkins delegates to Amazon ECS the execution of the builds on Docker based agents. Each Jenkins build is executed on a dedicated Docker container that is wiped-out at the end of the build.

[Amazon EC2 Container Service plugin with autoscaling capabilities](https://plugins.jenkins.io/scalable-amazon-ecs)

Use Amazon EC2 Container Service to provide elastic slaves.

[Amazon EC2 Container Service (ECS)](https://aws.amazon.com/ecs/) is AWS' service for Docker container orchestration letting you deploy Docker based applications on a cluster. This plugin lets you use Amazon ECS Container Service to manage Jenkins cloud agents. Jenkins delegates to Amazon ECS the execution of the builds on Docker based agents.  
Each Jenkins build is executed on a dedicated Docker container that is wiped-out at the end of the build.The ECS cluster is composed of Amazon EC2 virtual machines instantiated within the boundaries the user's account (typically in an Amazon VPC). These virtual machines can be declared statically or can be managed dynamically by AWS ECS thanks to AWS Auto Scaling and AWS CloudFormation.

Jenkins agents are connected to the Jenkins master using the JNLP protocol.

[AWS CodePipeline](https://plugins.jenkins.io/aws-codepipeline)

AWS CodePipeline Integration

[AWS CodePipeline](https://aws.amazon.com/codepipeline/) is a continuous delivery service for fast and reliable application updates. The AWS CodePipeline plugin for Jenkins provides a pre-build SCM and a post-build (publisher) step for your Jenkins project. It will poll for AWS CodePipeline jobs, and download input artifacts. When a build succeeds, it will compress the build artifacts and upload them to AWS CodePipeline.

[Artifact Manager on S3](https://plugins.jenkins.io/artifact-manager-s3)

A Jenkins plugin to keep artefacts and Pipeline stashes in Amazon S3.

Artifact Manager on S3 plugin:Artifact Manager on S3 plugin is an Artifact Manager that allow you to store you artifacts into a S3 Bucket on Amazon. The use of this S3 Bucket as a artefact storage is transparent to Jenkins and your jobs, it works like the default Artifact Manager.