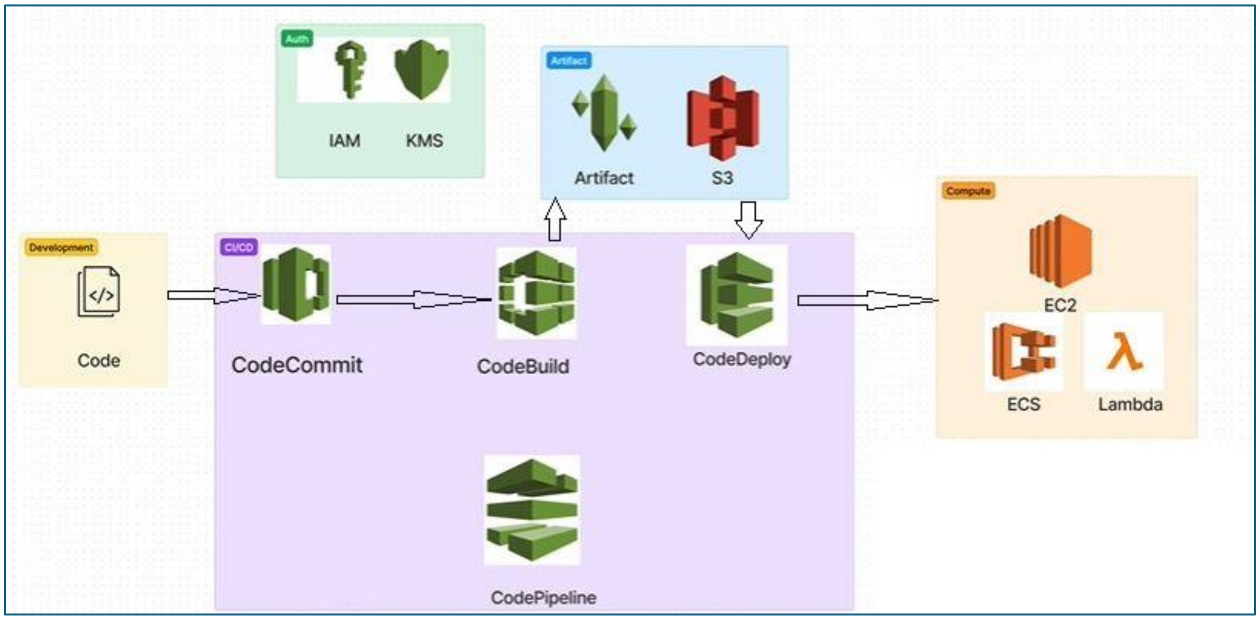
CI/CD Automation on AWS

Project Created By: AMAN DUGGAL
LinkedIn: https://www.linkedin.com/in/aman-duggal-4591bb146



Overview:

The CI/CD Architecture on AWS streamlines the development lifecycle by automating key processes from code development to deployment. It encompasses various AWS services and tools to ensure efficient and reliable delivery of software applications.

AWS Services Used:

- AWS IAM (Identity and Access Management)
- AWS KMS (Key Management Service)
- AWS S3 (Simple Storage Service)
- AWS CodeCommit
- AWS CodeBuild
- AWS CodeDeploy
- AWS CodePipeline

Overview:

The CI/CD Architecture on AWS is designed to enhance development efficiency and reliability by automating key stages of the software delivery process. It begins with authentication services provided by IAM, ensuring secure access to resources. KMS facilitates encryption and decryption of sensitive data. Artifacts and S3 serve as storage solutions for code and related files.

The pivotal components of the architecture include CodeCommit for version control, CodeBuild for building applications, CodeDeploy for deploying applications, and CodePipeline for orchestrating the entire workflow.

Implementation Steps:

1. Setup IAM:
- Create IAM users with appropriate permissions for accessing AWS services.
- Assign IAM policies such as 'AWSCodeCommitPowerUser' to grant necessary access.

2. Utilize KMS:
- Use KMS to manage encryption keys for securing sensitive data and code.

3. Storage Configuration:
- Store code artifacts and related files in Amazon S3 buckets for easy access and version control.

4. CodeCommit Setup:
- Create repositories in AWS CodeCommit to store and manage source code.
- Configure access permissions for users or teams to interact with repositories.

5. Code Build Process:
- Set up AWS CodeBuild projects to automate the build process.
- Define build specifications in a `buildspec.yml` file to specify the build tasks.
- Configure build environments and specify build triggers.

6. Code Deployment:
- Configure AWS CodeDeploy to automate the deployment of code to EC2 instances.
- Create deployment groups and application configurations in CodeDeploy.
- Define deployment specifications in an `appspec.yml` file to outline the deployment process.
- Install and configure the CodeDeploy agent on EC2 instances.

7. Code Pipeline Configuration:
- Create a pipeline in AWS CodePipeline to automate the end-to-end software delivery process.
- Specify source, build, and deploy stages in the pipeline.
- Configure triggers to initiate pipeline execution upon code changes.

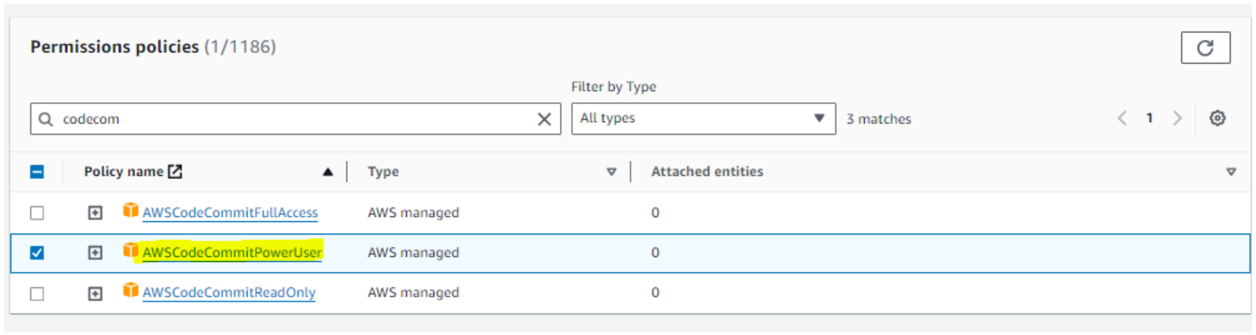
Result

The successful implementation of the CI/CD Architecture on AWS enables seamless code management, building, testing, and deployment processes. Developers can efficiently commit code changes, trigger automatic builds, and deploy applications with confidence.

This streamlined workflow fosters collaboration, accelerates time-to-market, and ensures the delivery of high-quality software products.

Step 1: Code Commit

Step 1: Create the IAM user.
Add Permission > **'AWSCodeCommitPowerUser'.**

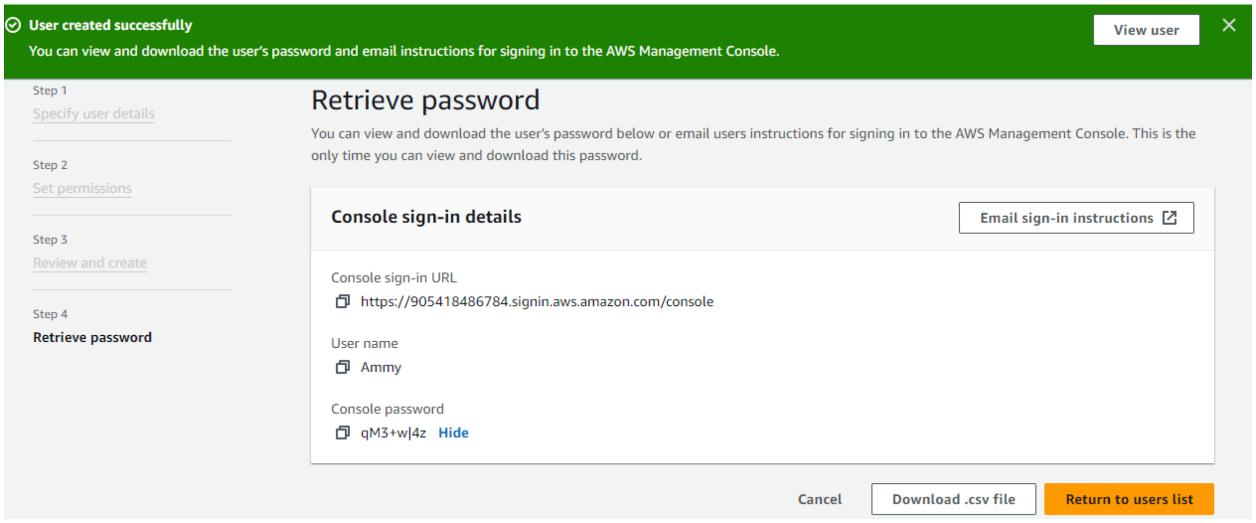


Permissions policies (1/1186)

codecom

Filter by Type: All types — 3 matches

	Policy name	Type	Attached entities
☐	AWSCodeCommitFullAccess	AWS managed	0
☑	AWSCodeCommitPowerUser	AWS managed	0
☐	AWSCodeCommitReadOnly	AWS managed	0



User created successfully
You can view and download the user's password and email instructions for signing in to the AWS Management Console.

Retrieve password

You can view and download the user's password below or email users instructions for signing in to the AWS Management Console. This is the only time you can view and download this password.

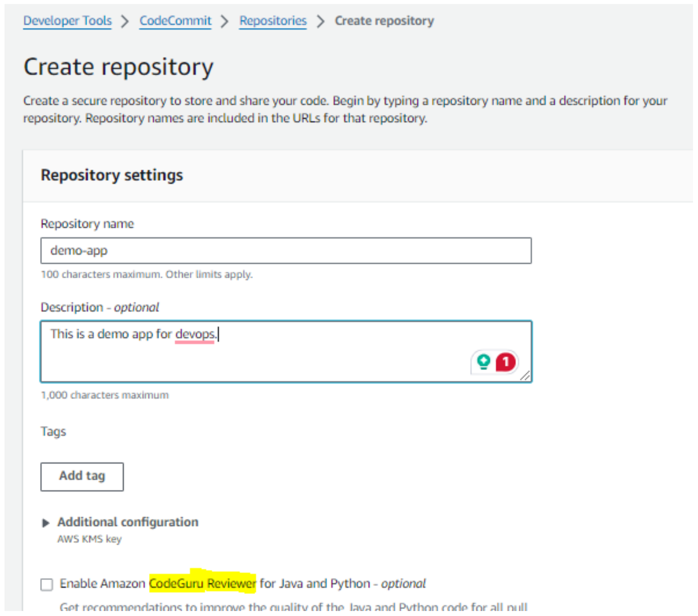
Console sign-in details

Console sign-in URL
https://905418486784.signin.aws.amazon.com/console

User name
Ammy

Console password
qM3+w|4z Hide

Step 2: Now create one repository in the Code Commit.



Developer Tools > CodeCommit > Repositories > Create repository

Create repository

Create a secure repository to store and share your code. Begin by typing a repository name and a description for your repository. Repository names are included in the URLs for that repository.

Repository settings

Repository name
demo-app
100 characters maximum. Other limits apply.

Description - *optional*
This is a demo app for devops.
1,000 characters maximum

Tags

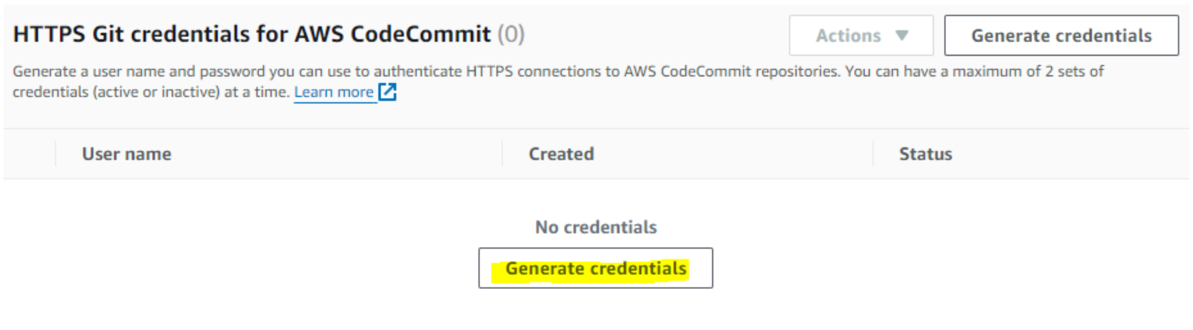
Additional configuration
AWS KMS key

☐ Enable Amazon CodeGuru Reviewer for Java and Python - *optional*
Get recommendations to improve the quality of the Java and Python code for all pull

NOTE: CodeGuru Reviewer is like a SonarQube, which scans the code and tell if any vulnerability or any defects in the code is present.

Step 3: Now give the access of Code Commit Repo to our IAM user.

- Go to IAM > Open your user > Security Credentials.
- Go to **'HTTPS Git credentials for AWS Code Commit'** > Generate Credentials.

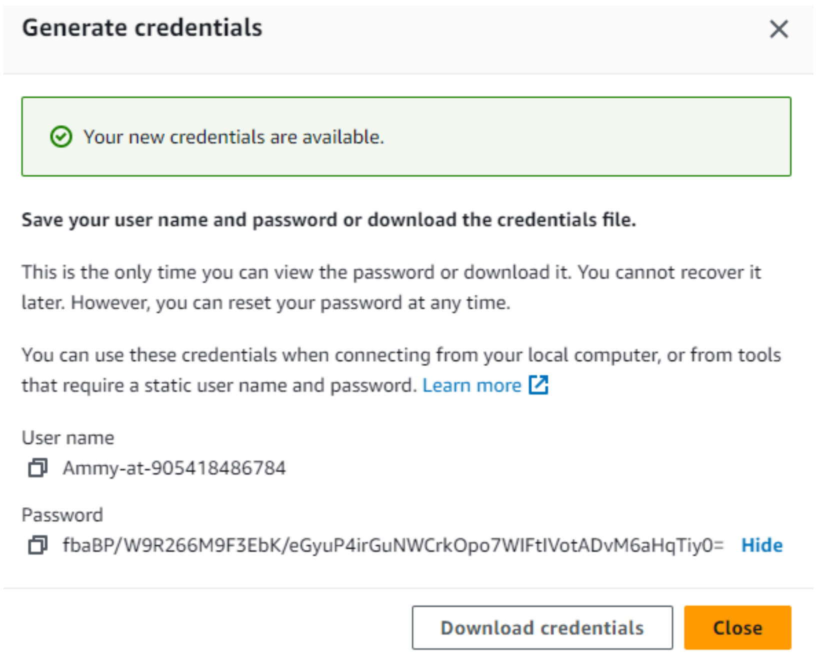


HTTPS Git credentials for AWS CodeCommit (0)

Generate a user name and password you can use to authenticate HTTPS connections to AWS CodeCommit repositories. You can have a maximum of 2 sets of credentials (active or inactive) at a time. Learn more

User name	Created	Status

No credentials



Generate credentials

Your new credentials are available.

Save your user name and password or download the credentials file.

This is the only time you can view the password or download it. You cannot recover it later. However, you can reset your password at any time.

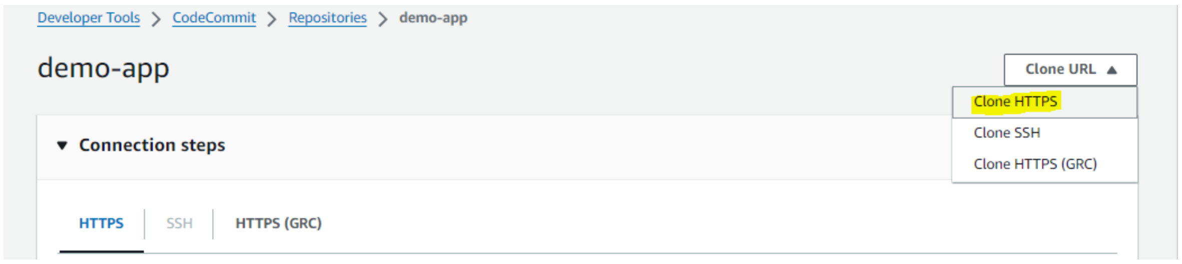
You can use these credentials when connecting from your local computer, or from tools that require a static user name and password. Learn more

User name
Ammy-at-905418486784

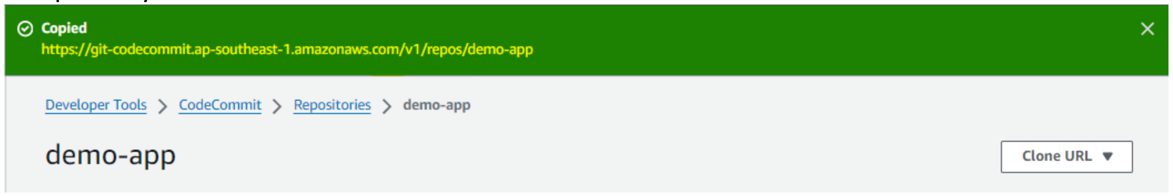
Password
fbaBP/W9R266M9F3EbK/eGyuP4irGuNWCrkOpo7WIFtIVotADvM6aHqTiy0=

- Now, by using these credentials we can access the code commit repo.

Step 4: To Clone the Code Commit Repo. Select the Clone URL > Clone HTTPS.



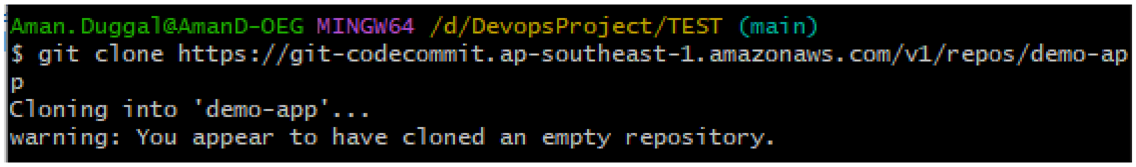
- Once you click on Clone HTTPS, you will get the URL and by using this URL we can clone the repository.


Copied
https://git-codecommit.ap-southeast-1.amazonaws.com/v1/repos/demo-app

- Now open any suitable terminal like Git bash, Visual studio and run the below command.

git clone https://git-codecommit.ap-southeast-1.amazonaws.com/v1/repos/demo-app

- Enter the Repo Credentials which we created earlier.

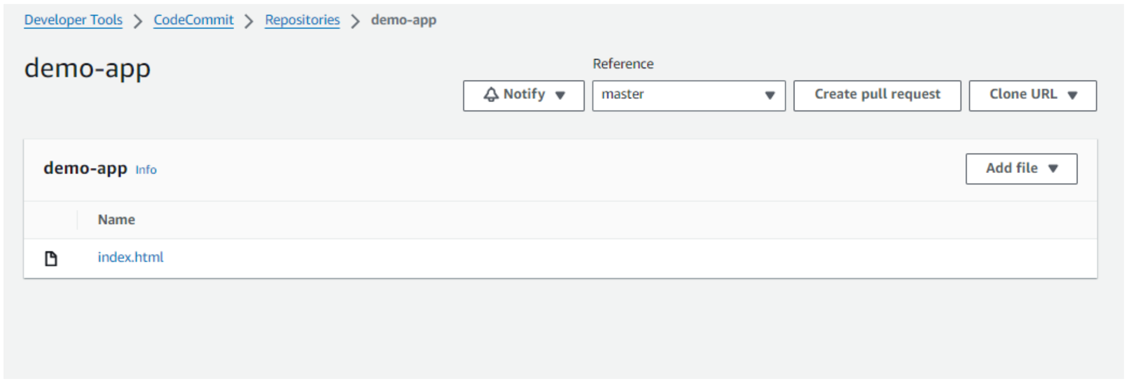


```
Aman.Duggal@AmanD-OEG MINGW64 /d/DevopsProject/TEST (main)
$ git clone https://git-codecommit.ap-southeast-1.amazonaws.com/v1/repos/demo-ap
p
Cloning into 'demo-app'...
warning: You appear to have cloned an empty repository.
```

- Create a file in the repository.

vi index.html.
<!DOCTYPE html>
<h1>My Demo APP</h1>

Now this file is present in our local repository, and now we want to push our file to remote repo.
git status
git add .
git commit -m "Adding new file"
git push origin master



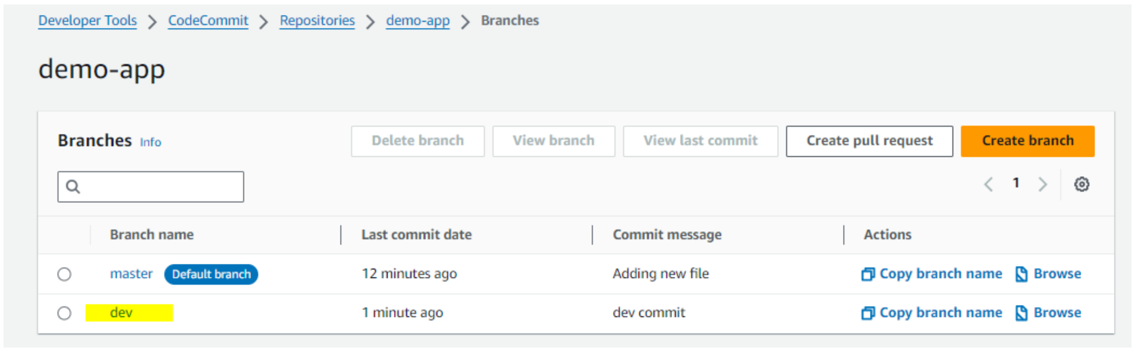
By this we can add our files from local repo to remote repo.

Step: 5 Now, to create a new branch.
git checkout -b dev **--#New branch is created and switched to dev branch.**

vi index.html --#Now add new lines in the file.

<!DOCTYPE html>
<h1>My Demo APP</h1>
<p1>New line</p>

Now, commit this file in the dev branch.
git add .
git commit -m "Added new line"
git push origin dev



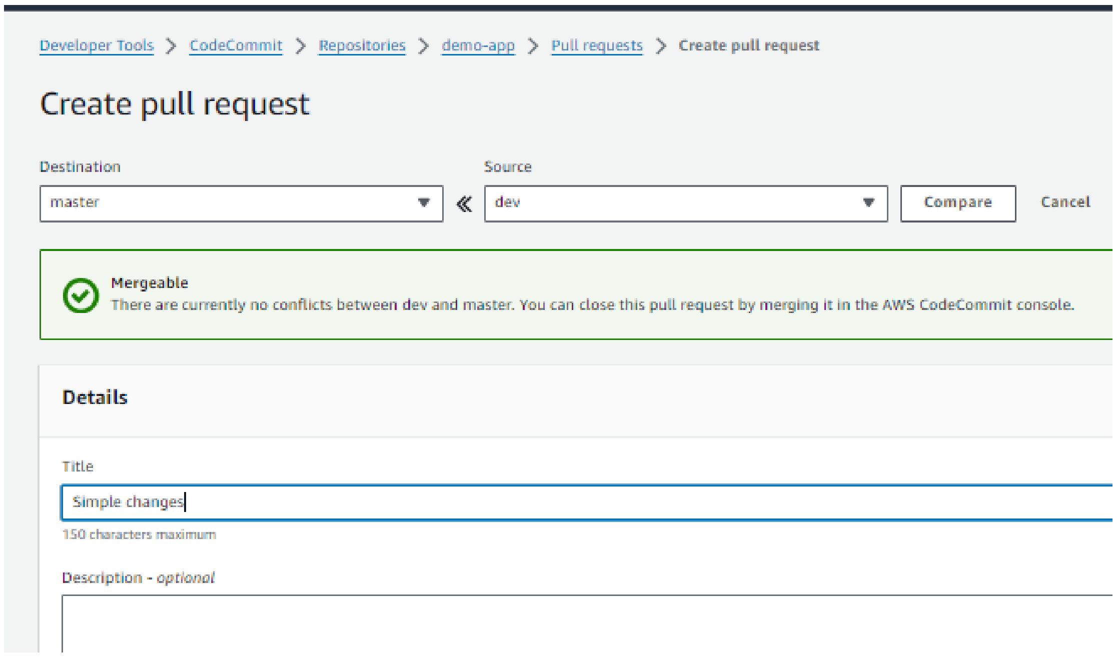
Developer Tools > CodeCommit > Repositories > demo-app > Branches

demo-app

Branches Info

	Branch name	Last commit date	Commit message	Actions
○	master Default branch	12 minutes ago	Adding new file	Copy branch name Browse
○	dev	1 minute ago	dev commit	Copy branch name Browse

To **merge** the dev branch to master branch.
Go to 'Create pull request'



Developer Tools > CodeCommit > Repositories > demo-app > Pull requests > Create pull request

Create pull request

Destination: master
Source: dev

Mergeable
There are currently no conflicts between dev and master. You can close this pull request by merging it in the AWS CodeCommit console.

Details

Title: Simple changes
150 characters maximum

Description - *optional*

Till now, the code commit is done.

Step 2: Code Build

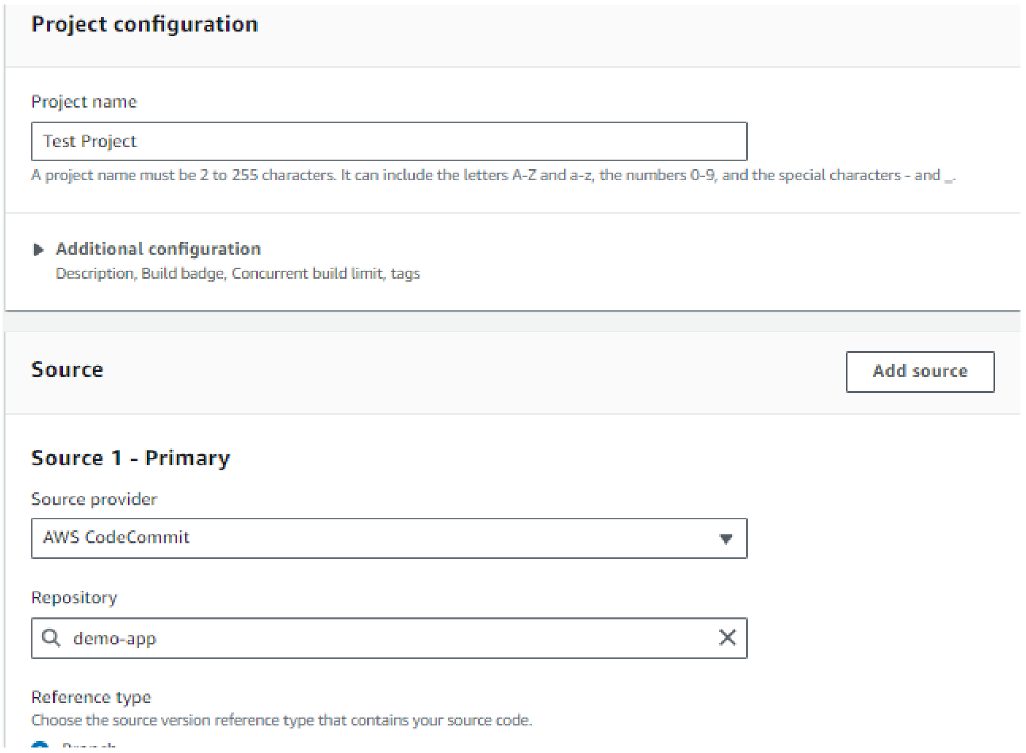
Code Build works like a Jenkins.

Step 1: Now create a project.

- Select Code Build service > Build Projects > Create Project.

  Project name: Test Project
  Source provider: AWS Code Commit --#From where the code will be picked.
  Repository: demo-app
  Branch: Master
  Environment Image: Manage Image --#On which environment you want to build, is it a Linux or Ubuntu.
  Role name: codebuild-Test Project-service-role

NOTE: IAM created the Code Commit repo and other services but to access these services or resources, we need the access. So IAM created a service role. Basically, Service Role is like a policy which tells that the particular services are running on which accesses.
IAM will give the role to user and service also.



Project configuration

Project name

Test Project

A project name must be 2 to 255 characters. It can include the letters A-Z and a-z, the numbers 0-9, and the special characters - and _.

Additional configuration
Description, Build badge, Concurrent build limit, tags

Source

Source 1 - Primary

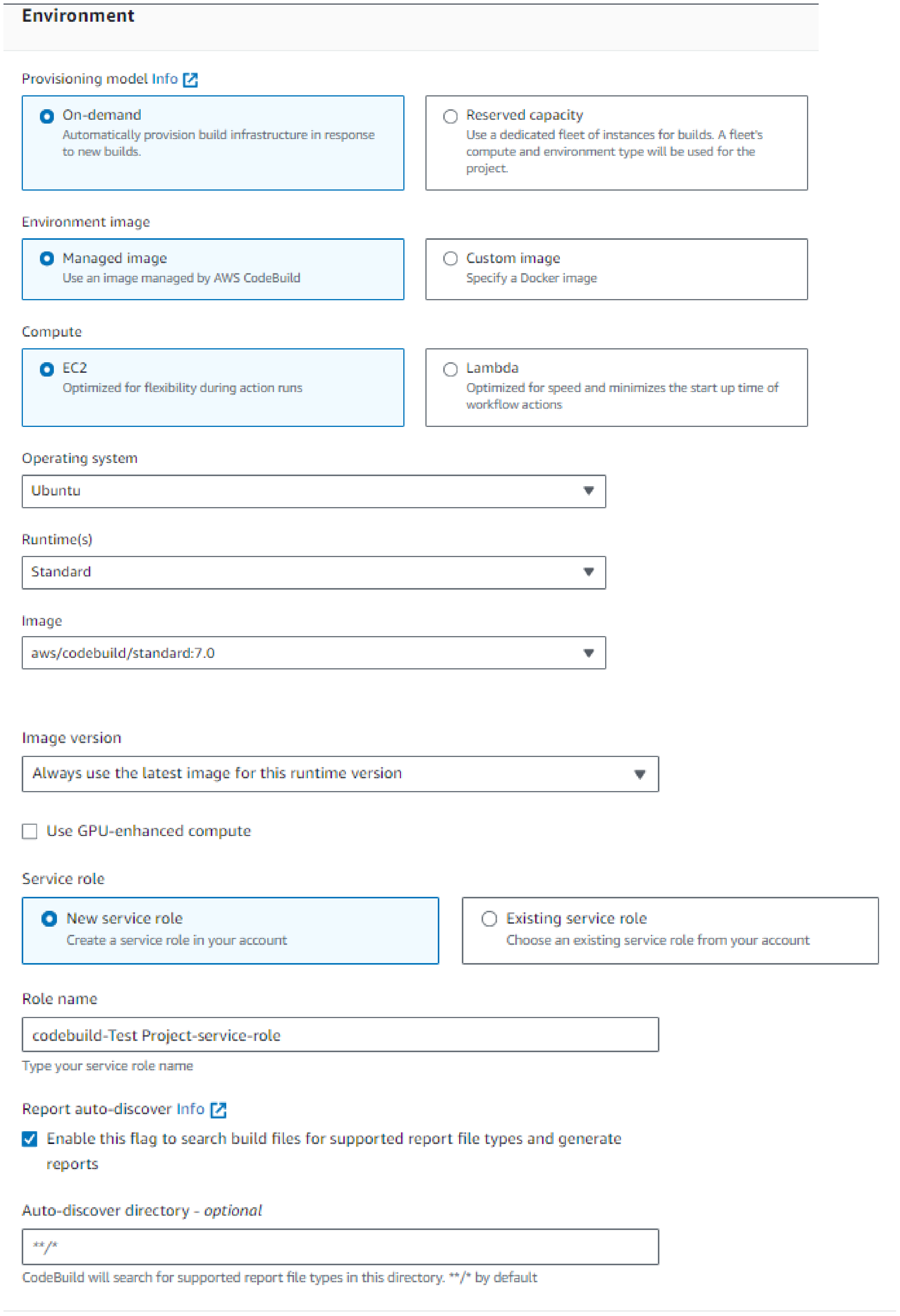
Source provider

AWS CodeCommit

Repository

demo-app

Reference type
Choose the source version reference type that contains your source code.


Environment

Provisioning model Info

[✓ On-demand] — Automatically provision build infrastructure in response to new builds.

Reserved capacity — Use a dedicated fleet of instances for builds. A fleet's compute and environment type will be used for the project.

Environment image

[✓ Managed image] — Use an image managed by AWS CodeBuild

Custom image — Specify a Docker image

Compute

[✓ EC2] — Optimized for flexibility during action runs

Lambda — Optimized for speed and minimizes the start up time of workflow actions

Operating system: Ubuntu

Runtime(s): Standard

Image: aws/codebuild/standard:7.0

Image version: Always use the latest image for this runtime version

☐ Use GPU-enhanced compute

Service role

[✓ New service role] — Create a service role in your account

Existing service role — Choose an existing service role from your account

Role name: codebuild-Test Project-service-role

Type your service role name

Report auto-discover Info

✓ Enable this flag to search build files for supported report file types and generate reports

Auto-discover directory - *optional*: **/*

CodeBuild will search for supported report file types in this directory. **/* by default

- Under Build Spec, we need to write the configuration file. Build Spec is like a specification file in which the tasks are mentioned which are going to be performed while building.

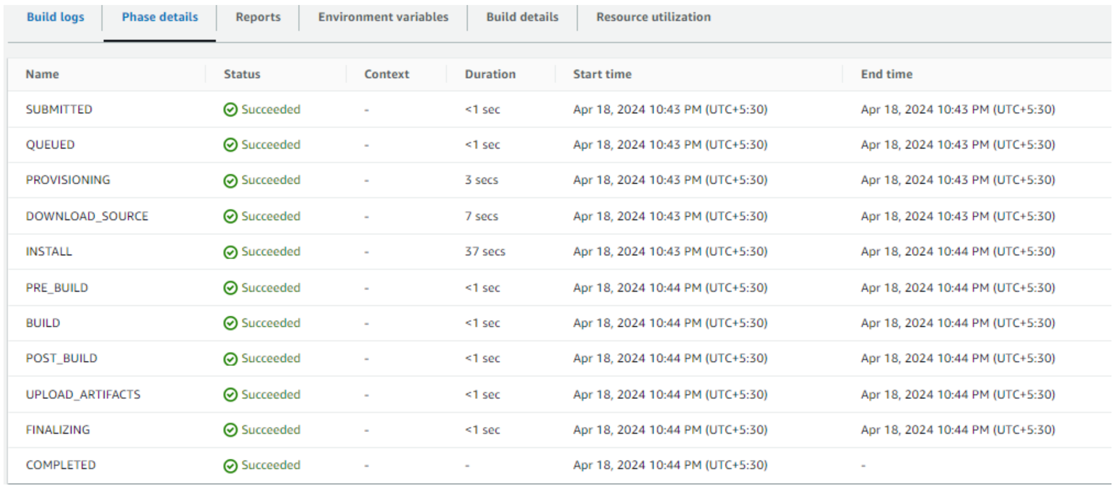
```
vi buildspec.yml  --#Create file with the same name.

version: 0.2

phases:
  install:
    commands:
      - echo Installing NGINX
      - sudo apt-get update
      - sudo apt-get install nginx -y
  build:
    commands:
      - echo Build started on 'date'
      - cp index.html /var/www/html/
  post_build:
    commands:
      - echo Configuring NGINX

artifacts:
    files:
      - '**/*'
```

- Now commit and push this file to our remote repo.

Note: Change the branch to master.

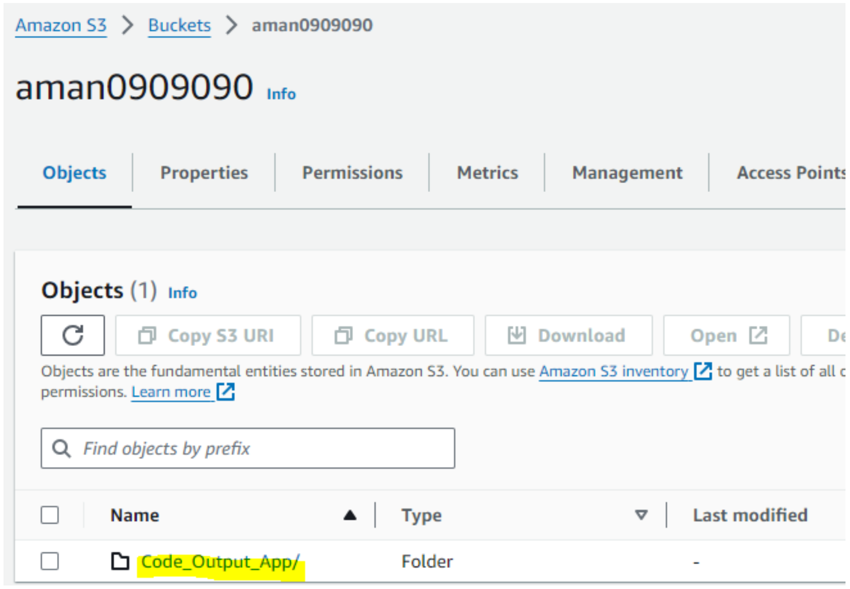
```
git add .
git commit -m "Adding buildspec file"
git push origin master
```

- Now click on Create Project > Start Build.



Name	Status	Context	Duration	Start time	End time
SUBMITTED	Succeeded	-	<1 sec	Apr 18, 2024 10:43 PM (UTC+5:30)	Apr 18, 2024 10:43 PM (UTC+5:30)
QUEUED	Succeeded	-	<1 sec	Apr 18, 2024 10:43 PM (UTC+5:30)	Apr 18, 2024 10:43 PM (UTC+5:30)
PROVISIONING	Succeeded	-	3 secs	Apr 18, 2024 10:43 PM (UTC+5:30)	Apr 18, 2024 10:43 PM (UTC+5:30)
DOWNLOAD_SOURCE	Succeeded	-	7 secs	Apr 18, 2024 10:43 PM (UTC+5:30)	Apr 18, 2024 10:43 PM (UTC+5:30)
INSTALL	Succeeded	-	37 secs	Apr 18, 2024 10:43 PM (UTC+5:30)	Apr 18, 2024 10:44 PM (UTC+5:30)
PRE_BUILD	Succeeded	-	<1 sec	Apr 18, 2024 10:44 PM (UTC+5:30)	Apr 18, 2024 10:44 PM (UTC+5:30)
BUILD	Succeeded	-	<1 sec	Apr 18, 2024 10:44 PM (UTC+5:30)	Apr 18, 2024 10:44 PM (UTC+5:30)
POST_BUILD	Succeeded	-	<1 sec	Apr 18, 2024 10:44 PM (UTC+5:30)	Apr 18, 2024 10:44 PM (UTC+5:30)
UPLOAD_ARTIFACTS	Succeeded	-	<1 sec	Apr 18, 2024 10:44 PM (UTC+5:30)	Apr 18, 2024 10:44 PM (UTC+5:30)
FINALIZING	Succeeded	-	<1 sec	Apr 18, 2024 10:44 PM (UTC+5:30)	Apr 18, 2024 10:44 PM (UTC+5:30)
COMPLETED	Succeeded	-	-	Apr 18, 2024 10:44 PM (UTC+5:30)	-

- Now, if we want that our code will build at any specific location we can provide the artifact location.
    - Edit the project > Artifact
    - Select Amazon S3 > Select your bucket. --#Create the S3 bucket and create the folder inside the bucket in which your outputs will be stored.



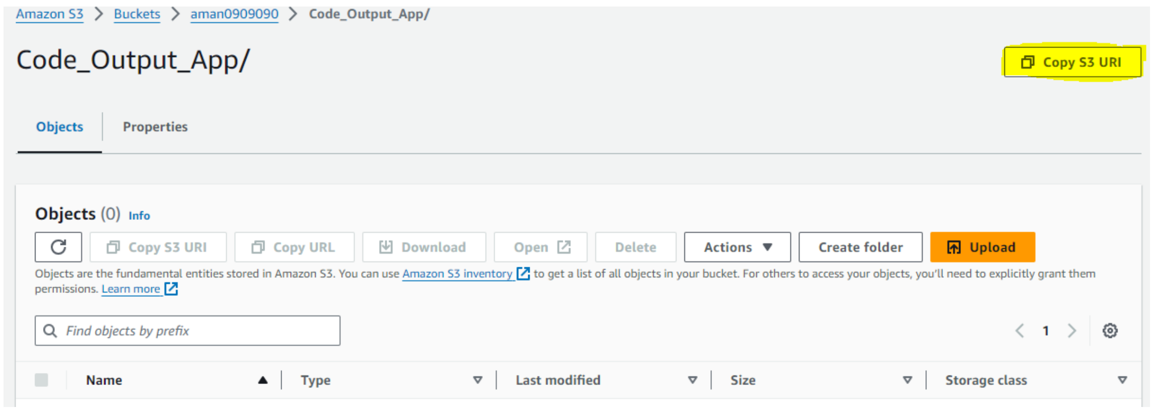
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all c
permissions. Learn more

Find objects by prefix

Name	Type	Last modified
Code_Output_App/	Folder	-

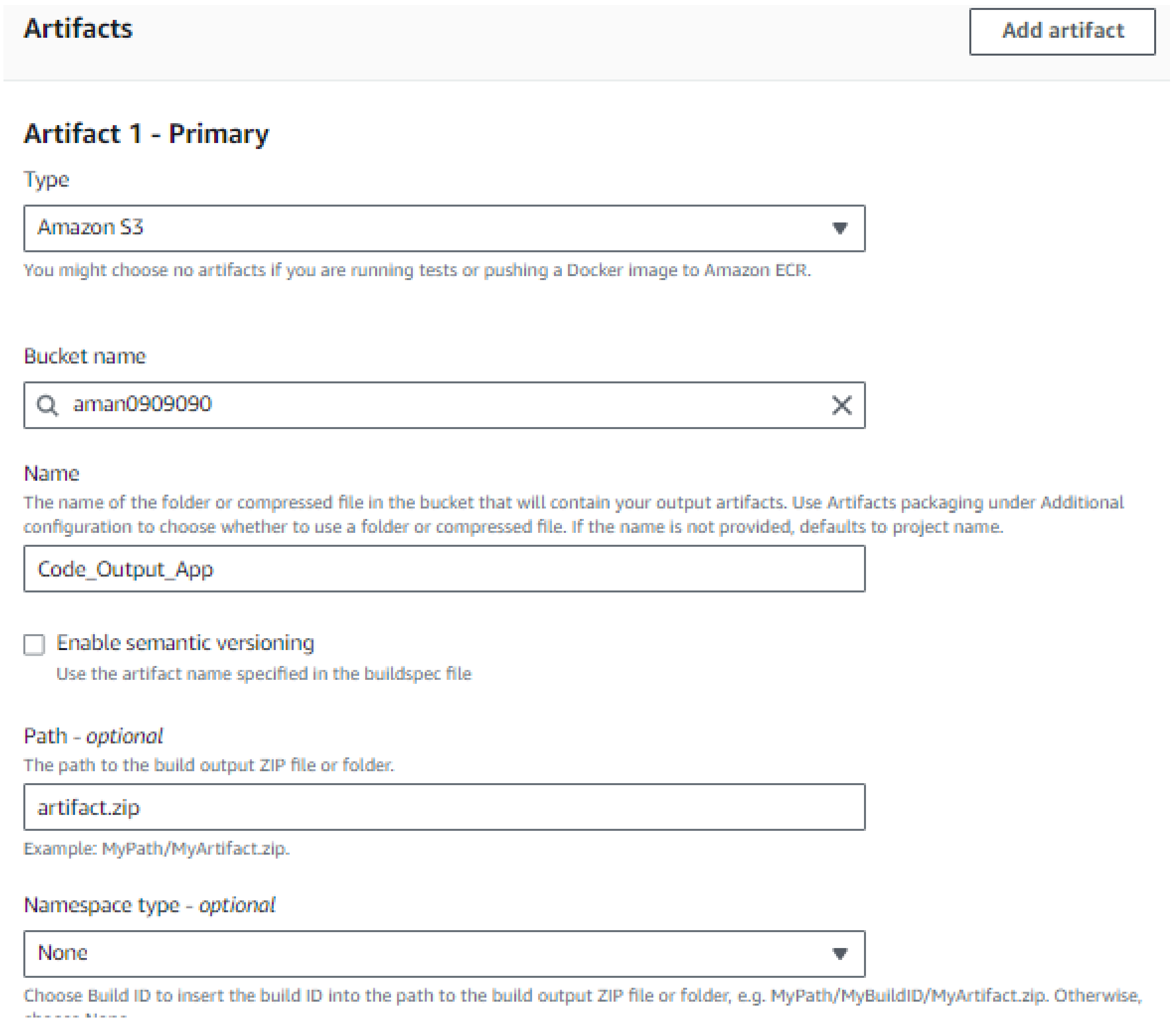
- Copy the S3 folder URL and paste it in the path option.

Ex: s3://aman0909090/Code_Output_App/artifact.zip --#Adding the artifact.zip at the end so it will become the final output.



Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

- Enter the details in the Artifacts and Update Project. Now every time the build happen the files will be stored in S3 bucket.



Type: Amazon S3

You might choose no artifacts if you are running tests or pushing a Docker image to Amazon ECR.

Bucket name: aman0909090

Name

The name of the folder or compressed file in the bucket that will contain your output artifacts. Use Artifacts packaging under Additional configuration to choose whether to use a folder or compressed file. If the name is not provided, defaults to project name.

Code_Output_App

☐ Enable semantic versioning

Use the artifact name specified in the buildspec file

Path - *optional*

The path to the build output ZIP file or folder.

artifact.zip

Example: MyPath/MyArtifact.zip.

Namespace type - *optional*

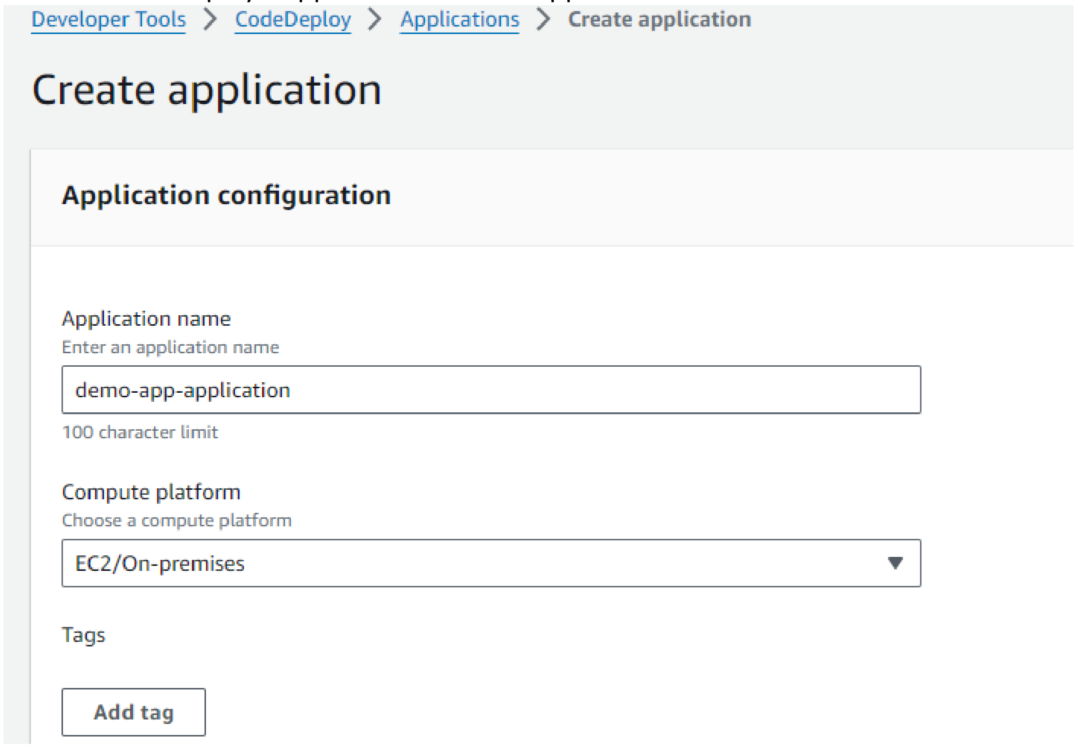
None

Choose Build ID to insert the build ID into the path to the build output ZIP file or folder, e.g. MyPath/MyBuildID/MyArtifact.zip. Otherwise,

Step 3: Code Deploy

Once we commit the code & Build the code, now we have to deploy the code. In code deploy, we deploy our application.

- Go to Code Deploy> Application > Create Application.



Developer Tools > CodeDeploy > Applications > Create application

Create application

Application configuration

Application name
Enter an application name
demo-app-application
100 character limit

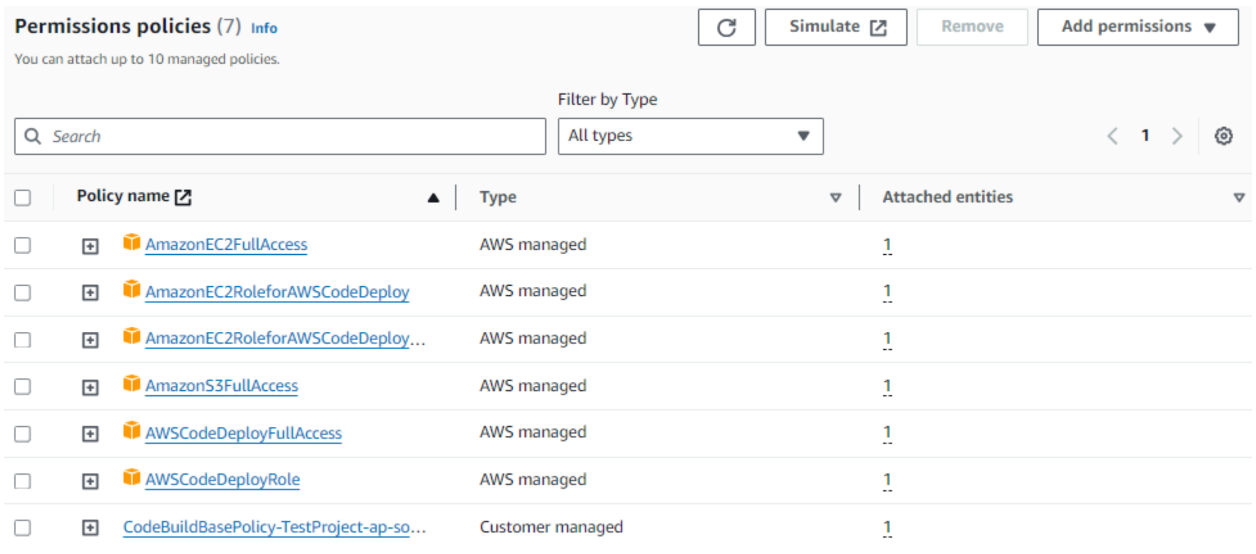
Compute platform
Choose a compute platform
EC2/On-premises

Tags

- Once your application is created, the environment or server on which it will run or be deployed is referred to as the deployment group. Now, create the deployment group.
  Deployment group is like, our application should deploy on single server or multiple server for that we need to create the deployment group.

- Click on Create Deployment Group.
  - **Deployment group name:** demo-app-depl-grp
  - **Service role:** <Enter the service role id> **Ex-** arn:aws:iam::905418486784:role/service-role/codebuild-TestProject-service-role

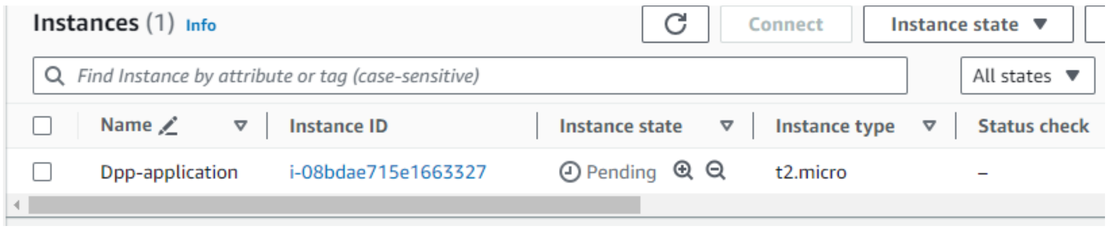
  Note: Create the service role for code deploy. The service role must have these access. This means, where our code will run that will have the necessary permissions.
  Create a service role in IAM with these permissions.


Permissions policies (7) Info
You can attach up to 10 managed policies.
Simulate
Remove
Add permissions
Filter by Type
Search
All types

Policy name	Type	Attached entities
AmazonEC2FullAccess	AWS managed	1
AmazonEC2RoleforAWSCodeDeploy	AWS managed	1
AmazonEC2RoleforAWSCodeDeploy...	AWS managed	1
AmazonS3FullAccess	AWS managed	1
AWSCodeDeployFullAccess	AWS managed	1
AWSCodeDeployRole	AWS managed	1
CodeBuildBasePolicy-TestProject-ap-so...	Customer managed	1

- **Deployment type:** In-place
- **Environment configuration:** Amazon EC2 instances --#In this EC2 instance, the code deploy will run our application.

  Create the EC2 instance for Ubuntu.



Name	Instance ID	Instance state	Instance type	Status check
Dpp-application	i-08bdae715e1663327	Pending	t2.micro	–

Now, enter:

- **Key:** Name
- **Name:** Dpp-application **--#Your EC2 instance name.**
- **Install AWS CodeDeploy Agent:** Never

The reason behind this is that when your application is deployed on EC2, Docker, or any other platform, you must ensure that the necessary tools are pre-installed and configured. For example, if you're deploying an NGINX application, or if another developer is deploying a Dockerized or any other type of application, Docker and related tools must be available beforehand.

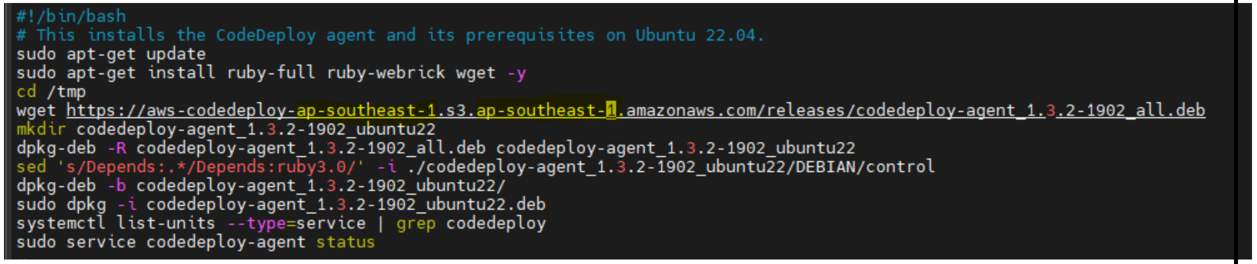
Otherwise, who will handle the installation? That's right, the agent. This agent is primarily designed for CodeDeploy operations. It's a fundamental aspect. Occasionally, mismatches between the CodeDeploy agent version and the CodeDeploy core agent version can lead to issues.

Basically, you are creating the setup between Code Deploy and EC2. By creating the Agent, it will be the link for communication.

- Connect your EC2 instance and execute the below script.

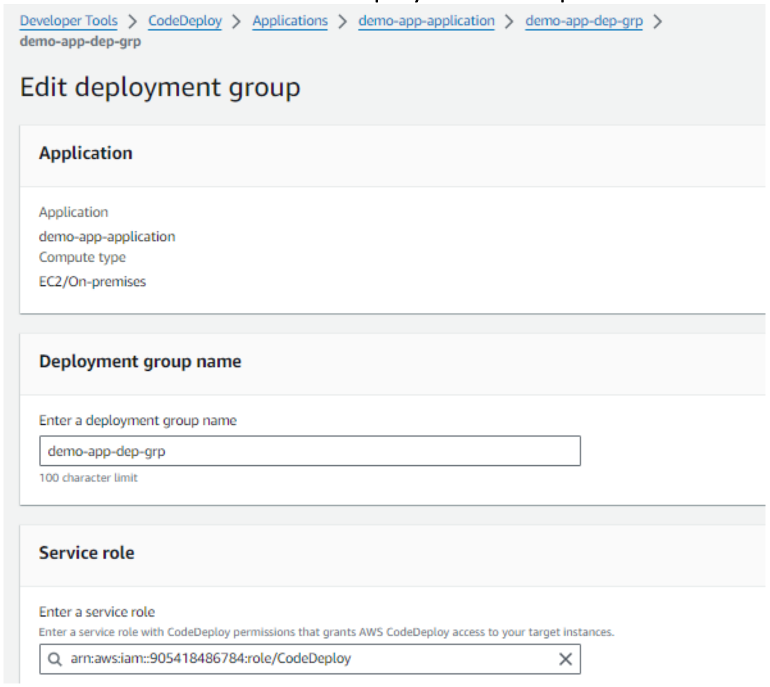
```
vi install.sh

sudo apt update
sudo apt install ruby-full
sudo apt install wget
cd /home/ubuntu
wget https://aws-codedeploy-eu-west-1.s3.eu-west-1.amazonaws.com/latest/install
chmod +x ./install
sudo ./install auto
systemctl status codedeploy-agent

service codedeploy-agent restart
bash install. sh
```

Note: There is a catch, since we are downloading the code deploy agent file, and that file will be present in the s3 bucket for every region. So on whatever region you created the instance or working on change the region name in the script. Same as highlighted below.



```
#!/bin/bash
# This installs the CodeDeploy agent and its prerequisites on Ubuntu 22.04.
sudo apt-get update
sudo apt-get install ruby-full ruby-webrick wget -y
cd /tmp
wget https://aws-codedeploy-ap-southeast-1.s3.ap-southeast-1.amazonaws.com/releases/codedeploy-agent_1.3.2-1902_all.deb
mkdir codedeploy-agent_1.3.2-1902_ubuntu22
dpkg-deb -R codedeploy-agent_1.3.2-1902_all.deb codedeploy-agent_1.3.2-1902_ubuntu22
sed 's/Depends:.*/Depends:ruby3.0/' -i ./codedeploy-agent_1.3.2-1902_ubuntu22/DEBIAN/control
dpkg-deb -b codedeploy-agent_1.3.2-1902_ubuntu22/
sudo dpkg -i codedeploy-agent_1.3.2-1902_ubuntu22.deb
systemctl list-units --type=service | grep codedeploy
sudo service codedeploy-agent status
```

- Once it's done create the Deployment Group.



Developer Tools > CodeDeploy > Applications > demo-app-application > demo-app-dep-grp > demo-app-dep-grp

Edit deployment group

Application

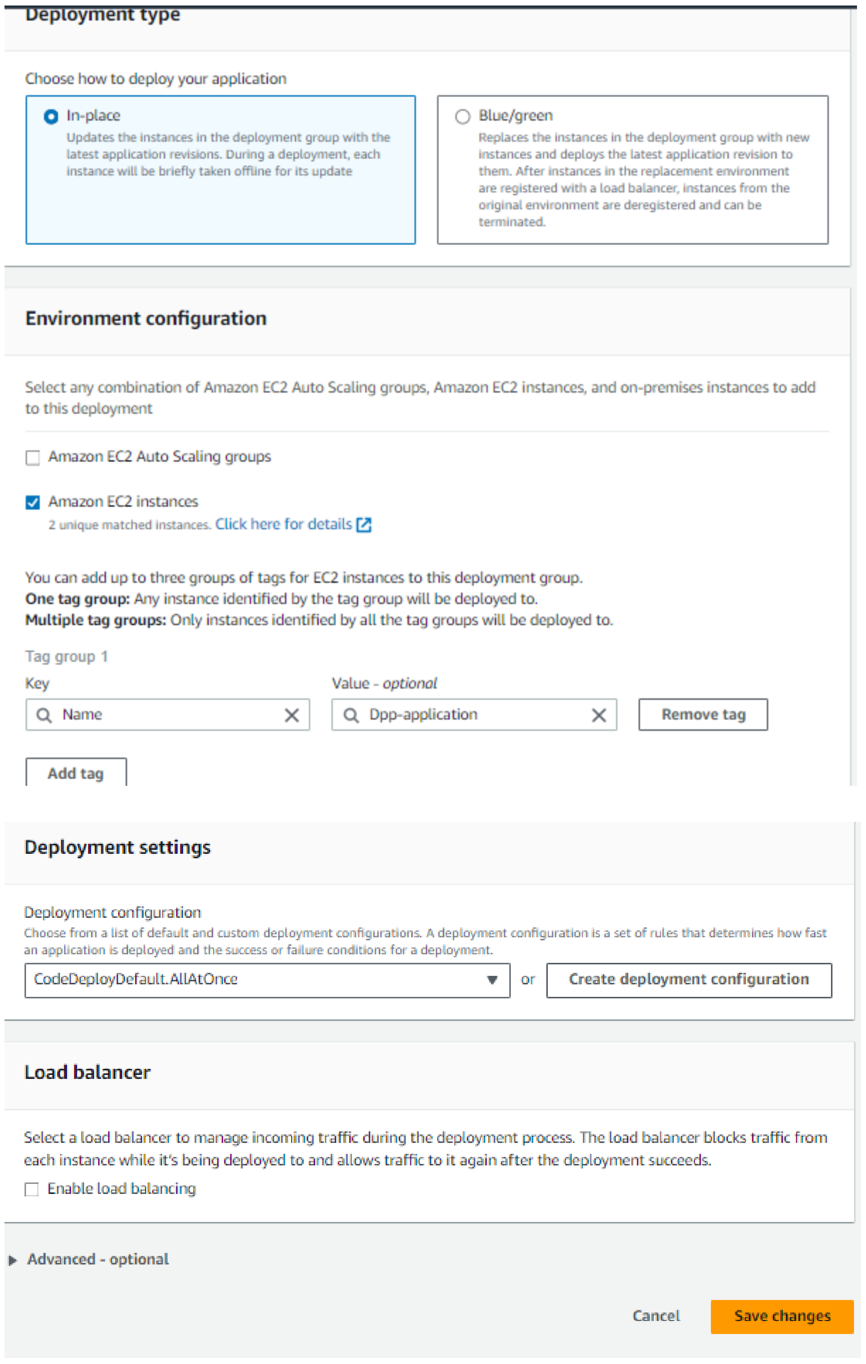
Application
demo-app-application
Compute type
EC2/On-premises

Deployment group name

Enter a deployment group name
demo-app-dep-grp
100 character limit

Service role

Enter a service role
Enter a service role with CodeDeploy permissions that grants AWS CodeDeploy access to your target instances.
arn:aws:iam::905418486784:role/CodeDeploy


Deployment type

Choose how to deploy your application

[✓ In-place]
Updates the instances in the deployment group with the latest application revisions. During a deployment, each instance will be briefly taken offline for its update

Blue/green
Replaces the instances in the deployment group with new instances and deploys the latest application revision to them. After instances in the replacement environment are registered with a load balancer, instances from the original environment are deregistered and can be terminated.

Environment configuration

Select any combination of Amazon EC2 Auto Scaling groups, Amazon EC2 instances, and on-premises instances to add to this deployment

☐ Amazon EC2 Auto Scaling groups

☑ Amazon EC2 instances
2 unique matched instances. Click here for details

You can add up to three groups of tags for EC2 instances to this deployment group.
One tag group: Any instance identified by the tag group will be deployed to.
Multiple tag groups: Only instances identified by all the tag groups will be deployed to.

Tag group 1

Key	Value - *optional*
Name	Dpp-application

Deployment settings

Deployment configuration
Choose from a list of default and custom deployment configurations. A deployment configuration is a set of rules that determines how fast an application is deployed and the success or failure conditions for a deployment.

CodeDeployDefault.AllAtOnce

Load balancer

Select a load balancer to manage incoming traffic during the deployment process. The load balancer blocks traffic from each instance while it's being deployed to and allows traffic to it again after the deployment succeeds.

☐ Enable load balancing

- Now our Deployment is created, and now to deploy the application we need to create one file i.e. **'appspec.yml'**.

It is similar file, that we created for the Code Build (buildspec.yml) which is the configuration file for code build.

The **appspec.yml** file is used by AWS CodeDeploy to specify how to deploy an application to an EC2 instance running on Linux. Overall, this appspec.yml file outlines the deployment process for an application, including where to deploy files and what actions to take before and after deployment.

vi appspec.yml

```
version: 0.0
os: linux
file:
  - source: /
    destination: /var/www/html
hooks:
  AfterInstall:
   - location: scripts/install_nginx.sh
     timeout: 300
     runas: root
  ApplicationStart:
   - location: scripts/start_nginx.sh
     timeout: 300
     runas: root
```

- **version:** Specifies the version of the AppSpec file format being used. In this case, it's version 0.0.

- **os:** Specifies the operating system of the target instance. In this case, it's Linux.

- **file:** Defines the files to be deployed and their destinations on the target instance. Here, it specifies that all files in the root directory (/) of the source (which could be the index.sh file present in the root (/) directory) should be deployed to the destination directory /var/www/html on the target instance.

- **hooks:** Contains lifecycle event hooks that specify actions to be performed at different stages of the deployment process.

- **AfterInstall:** Specifies actions to be performed after the application files are copied to the target instance but before the application is started. In this case, it indicates the execution of a shell script **install_nginx.sh** located in the scripts directory. It also specifies a timeout for the script execution and the user (root) under which the script should run.

Basically, the **install_nginx.sh** file having the command to install the nginx server.

- **ApplicationStart:** Specifies actions to be performed after the application is installed and ready to start. Here, it indicates the execution of a shell **script start_nginx.sh** located in the scripts directory. Similar to After Install, it also specifies a timeout for the script execution and the user (root) under which the script should run.

Basically, the **start_nginx.sh** file having the command to start the nginx server.

The agent that we installed will run this file in the EC2 instance.

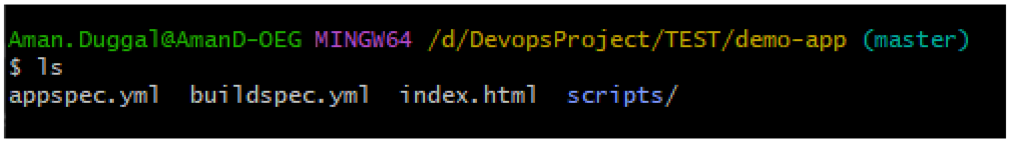
mkdir scripts
cd scripts

vi install_nginx.sh

sudo apt-get update
sudo apt-get install -y nginx

vi start_nginx.sh

sudo service nginx start

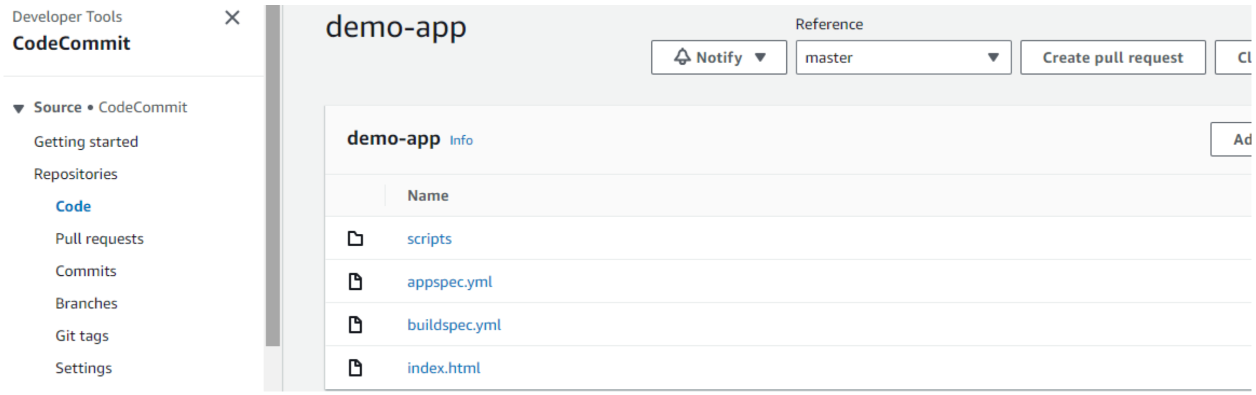


```
Aman.Duggal@AmanD-OEG MINGW64 /d/DevopsProject/TEST/demo-app (master)
$ ls
appspec.yml  buildspec.yml  index.html  scripts/
```

- Now, commit and push these changes to our remote repo.

git add .
git commit -m "Adding the appspec.yml file"
git push -m origin master

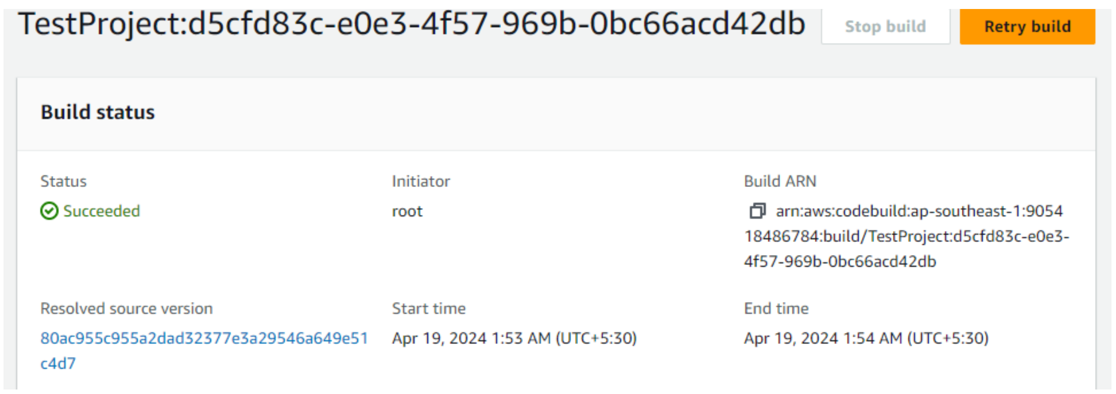
- Once you push, all these changes copied to our CodeCommit.



Name
scripts
appspec.yml
buildspec.yml
index.html

- Now, once all changes are moved to Code commit, now we build the code. By building, our latest code will be moved to s3 bucket.
    - Go to Code Build > Start Build.

Till now, to run the application on the server, we needed a configuration file named "appspec.yml" and we added it. Now, we should have stored it on s3 , but instead of uploading it we build the code so that the latest code reaches to s3 bucket, and from there CodeDeploy can easily pick them up.

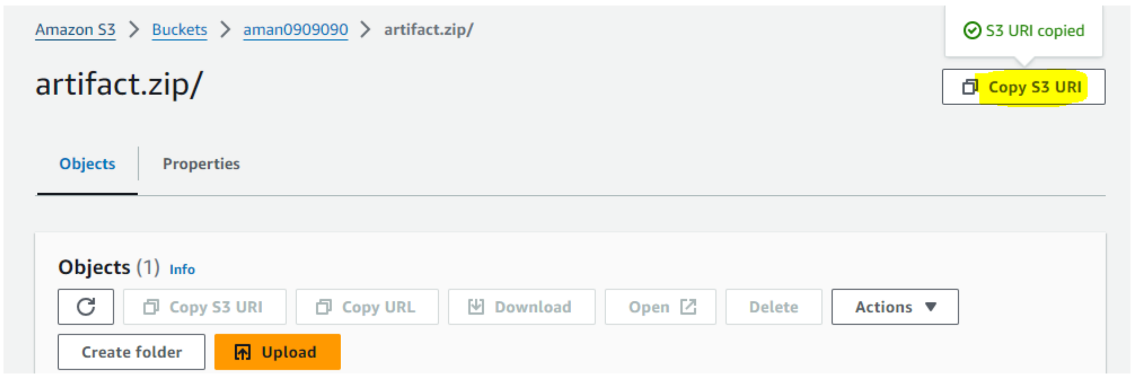

TestProject:d5cfd83c-e0e3-4f57-969b-0bc66acd42db

Build status

Status	Initiator	Build ARN
Succeeded	root	arn:aws:codebuild:ap-southeast-1:905418486784:build/TestProject:d5cfd83c-e0e3-4f57-969b-0bc66acd42db

Resolved source version	Start time	End time
80ac955c955a2dad32377e3a29546a649e51c4d7	Apr 19, 2024 1:53 AM (UTC+5:30)	Apr 19, 2024 1:54 AM (UTC+5:30)

- Now to start the deployment, open the application in deployment > deployment group.
- **Revision location:** **<Enter the S3 location where the code is build> Ex: s3://aman0909090/artifact.zip/**


Amazon S3 > Buckets > aman0909090 > artifact.zip/

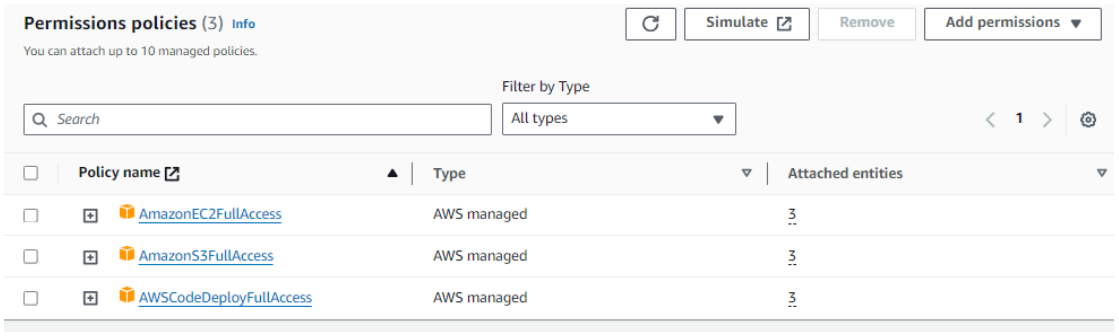
artifact.zip/

But when you, start the deployment, all the stages went on pending status and the deployment is failed. The reason, is since we get the permission to run the application, code deploy got the permission to connect EC2 and S3.

But the EC2 don't get the permission to fetch the data from s3 and communicate with code deploy. That why the deployment will get failed.

To resolve this, we need to create the new IAM role i.e **'ec2-code-deploy'.**

- **Go to IAM > Create Role > 'ec2-code-deploy' > EC2 > Add permission**

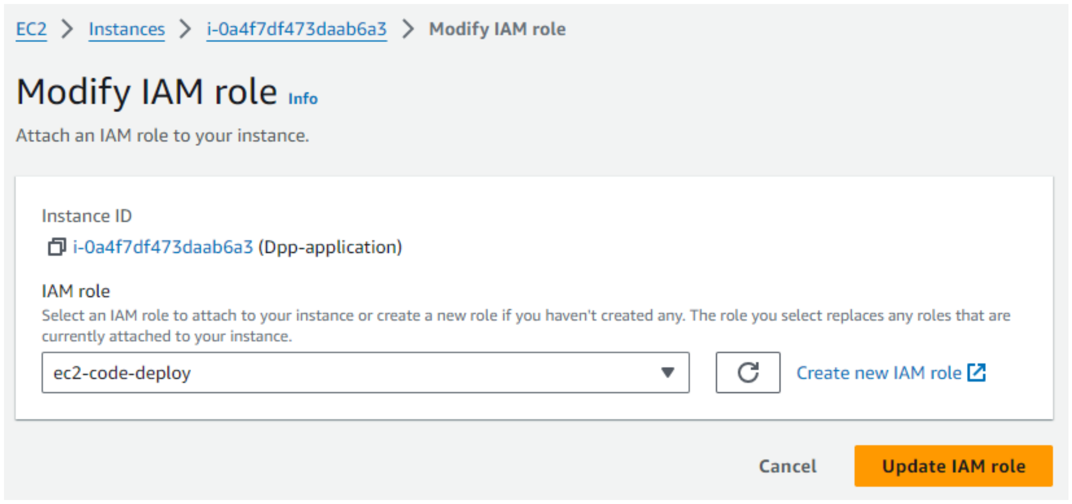


Permissions policies (3) Info

You can attach up to 10 managed policies.

Policy name	Type	Attached entities
AmazonEC2FullAccess	AWS managed	3
AmazonS3FullAccess	AWS managed	3
AWSCodeDeployFullAccess	AWS managed	3

- Now we need to assign this role to our EC2 instance.
  - **Select EC2 instance > Actions > Security > Modify IAM role > Update the IAM role which we created.**



EC2 > Instances > i-0a4f7df473daab6a3 > Modify IAM role

Modify IAM role Info

Attach an IAM role to your instance.

Instance ID

i-0a4f7df473daab6a3 (Dpp-application)

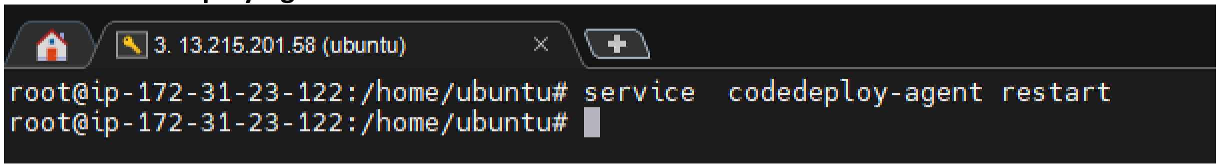
IAM role

Select an IAM role to attach to your instance or create a new role if you haven't created any. The role you select replaces any roles that are currently attached to your instance.

ec2-code-deploy

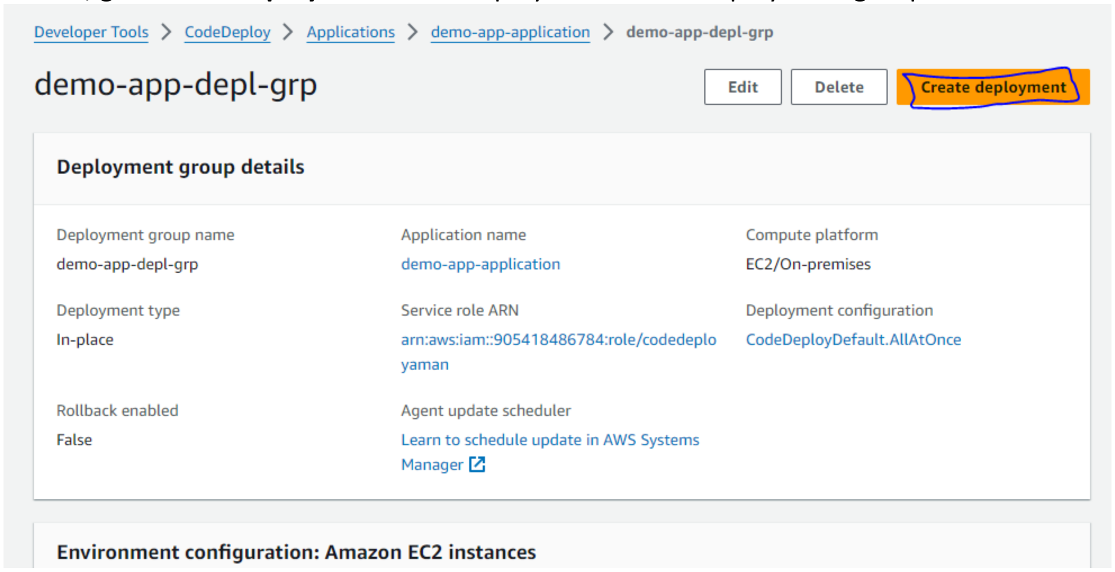
- Open EC2 terminal and restart the codedeploy-agent service.

service codedeploy-agent restart



```
root@ip-172-31-23-122:/home/ubuntu# service  codedeploy-agent restart
root@ip-172-31-23-122:/home/ubuntu#
```

- Now, go to **CodeDeploy** and create deployment under deployment group.

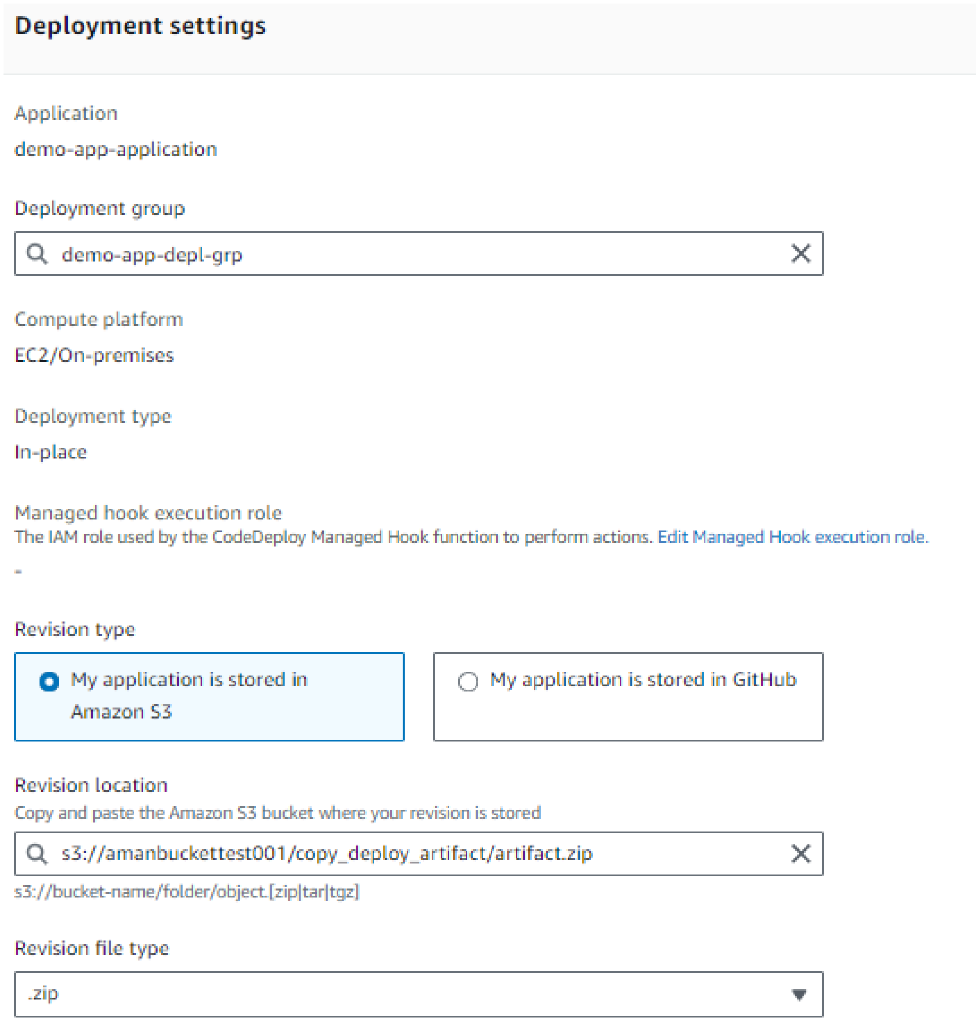

Developer Tools > CodeDeploy > Applications > demo-app-application > demo-app-depl-grp

demo-app-depl-grp

Deployment group details

Deployment group name	Application name	Compute platform
demo-app-depl-grp	demo-app-application	EC2/On-premises
Deployment type	Service role ARN	Deployment configuration
In-place	arn:aws:iam::905418486784:role/codedeplo yaman	CodeDeployDefault.AllAtOnce
Rollback enabled	Agent update scheduler	
False	Learn to schedule update in AWS Systems Manager	

Environment configuration: Amazon EC2 instances


Deployment settings

Application
demo-app-application

Deployment group
demo-app-depl-grp

Compute platform
EC2/On-premises

Deployment type
In-place

Managed hook execution role
The IAM role used by the CodeDeploy Managed Hook function to perform actions. Edit Managed Hook execution role.
-

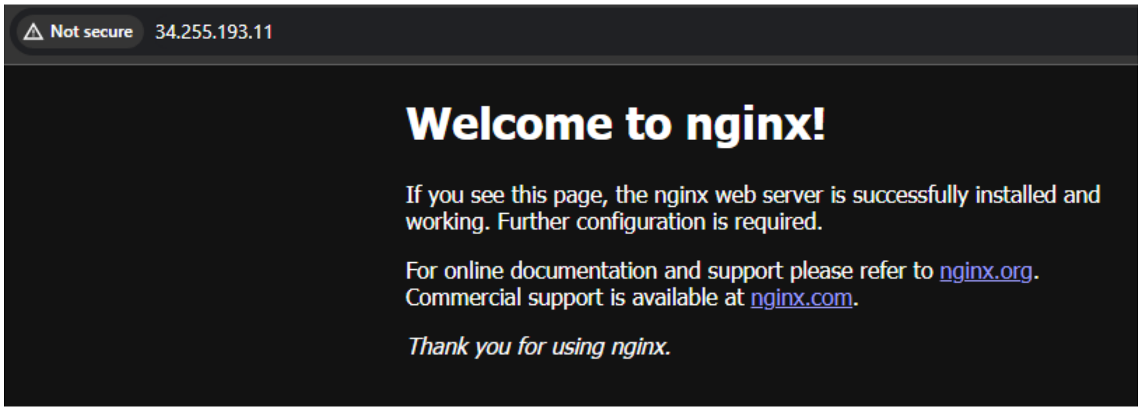
Revision type
[✓ My application is stored in Amazon S3] My application is stored in GitHub

Revision location
Copy and paste the Amazon S3 bucket where your revision is stored
s3://amanbuckettest001/copy_deploy_artifact/artifact.zip
s3://bucket-name/folder/object.[zip|tar|tgz]

Revision file type
.zip

- The deployment is successful and now check the application. Copy the public IP of the EC2 instance and run it on the browser.

Our application is deployed successfully.



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

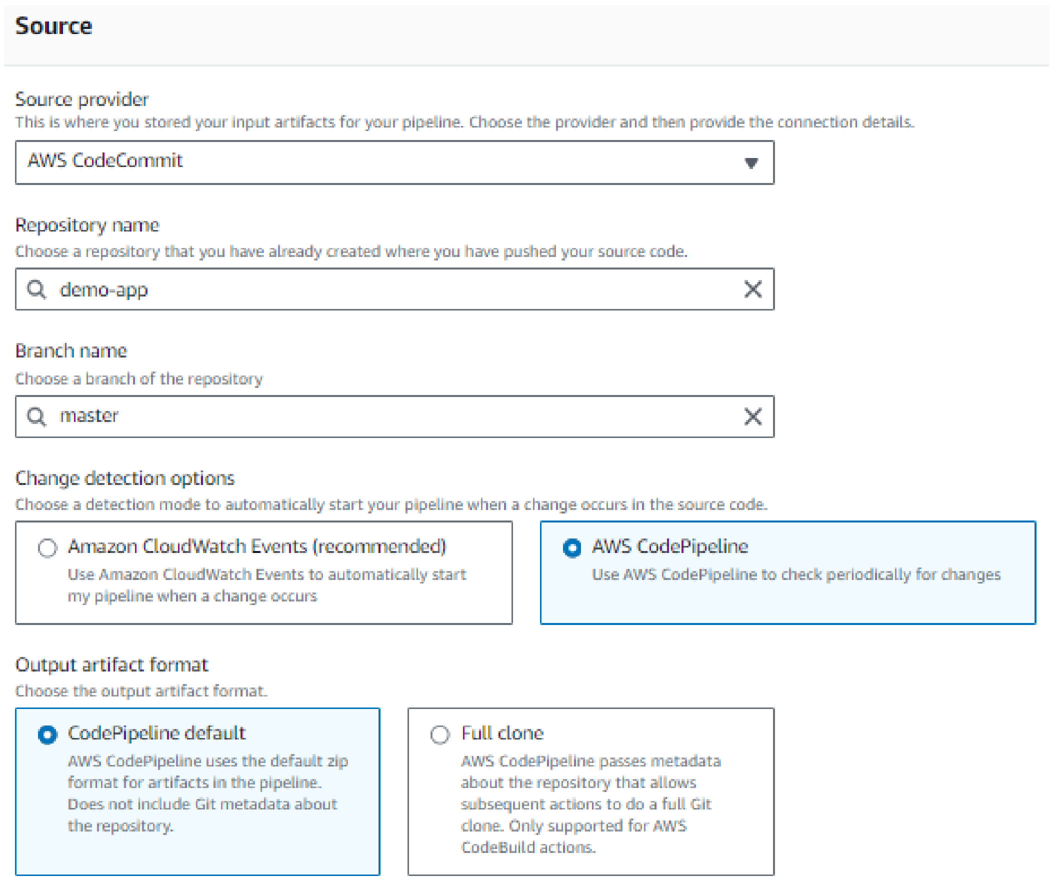
Thank you for using nginx.

Step 4: Code Pipeline

Now let's create a Pipeline where whenever our code is committed, it should build and deploy automatically.

Is committing code, building, and deploying the symbol of DevOps?
Yes

- Go to **CodePipeline in AWS console > Create New Pipeline.**
    - **Pipeline name:** demo-app-pipeline
    - **Pipeline type:** V2
    - **Execution mode:** Queued (Pipeline type V2 required)
    - **Service role:** New
    - **Role name:** **--#Auto create the role**
    - **Source provider:** AWS CodeCommit
    - **Repository name:** demo-app
    - **Branch:** master
    - **Change detection options:** AWS CodePipeline **--#It means if our code gets changed, so the pipeline will get executed on every change.**
    - **Output artifact format:** CodePipeline default


Source

Source provider
This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.
AWS CodeCommit

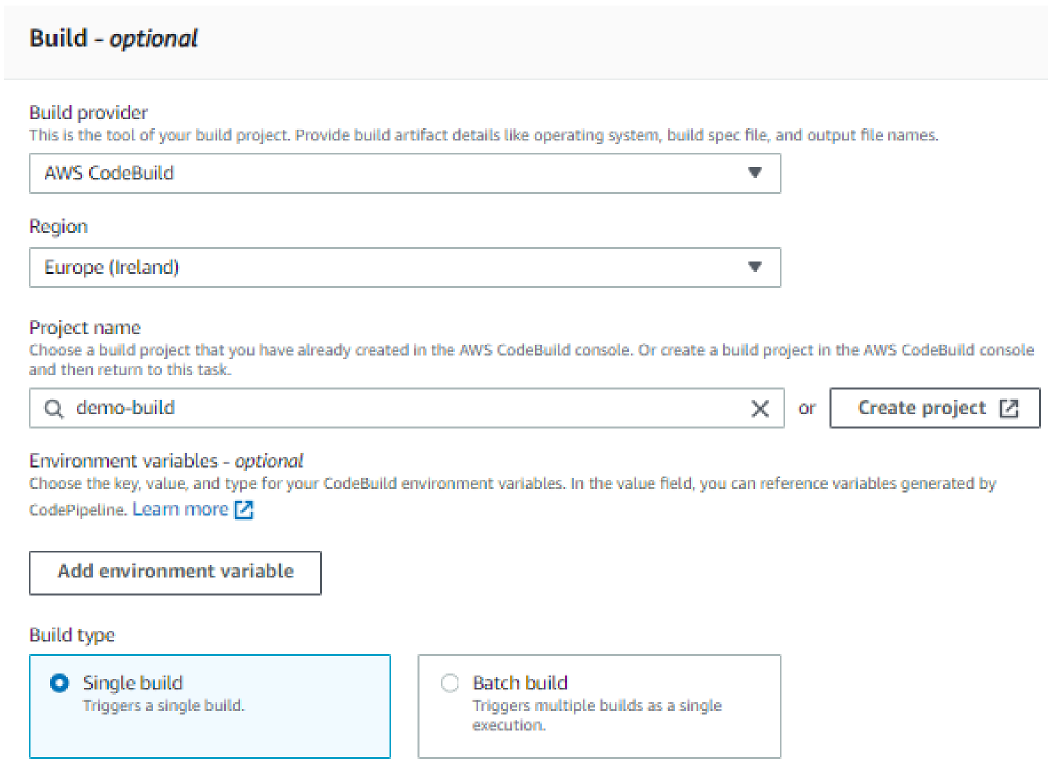
Repository name
Choose a repository that you have already created where you have pushed your source code.
demo-app

Branch name
Choose a branch of the repository
master

Change detection options
Choose a detection mode to automatically start your pipeline when a change occurs in the source code.
- [ ] Amazon CloudWatch Events (recommended) — Use Amazon CloudWatch Events to automatically start my pipeline when a change occurs
- [✓ AWS CodePipeline] — Use AWS CodePipeline to check periodically for changes

Output artifact format
Choose the output artifact format.
- [✓ CodePipeline default] — AWS CodePipeline uses the default zip format for artifacts in the pipeline. Does not include Git metadata about the repository.
- [ ] Full clone — AWS CodePipeline passes metadata about the repository that allows subsequent actions to do a full Git clone. Only supported for AWS CodeBuild actions.

- **Build provider:** AWS CodeBuild
- **Project Name:** demo-build


Build - *optional*

Build provider
This is the tool of your build project. Provide build artifact details like operating system, build spec file, and output file names.
AWS CodeBuild

Region
Europe (Ireland)

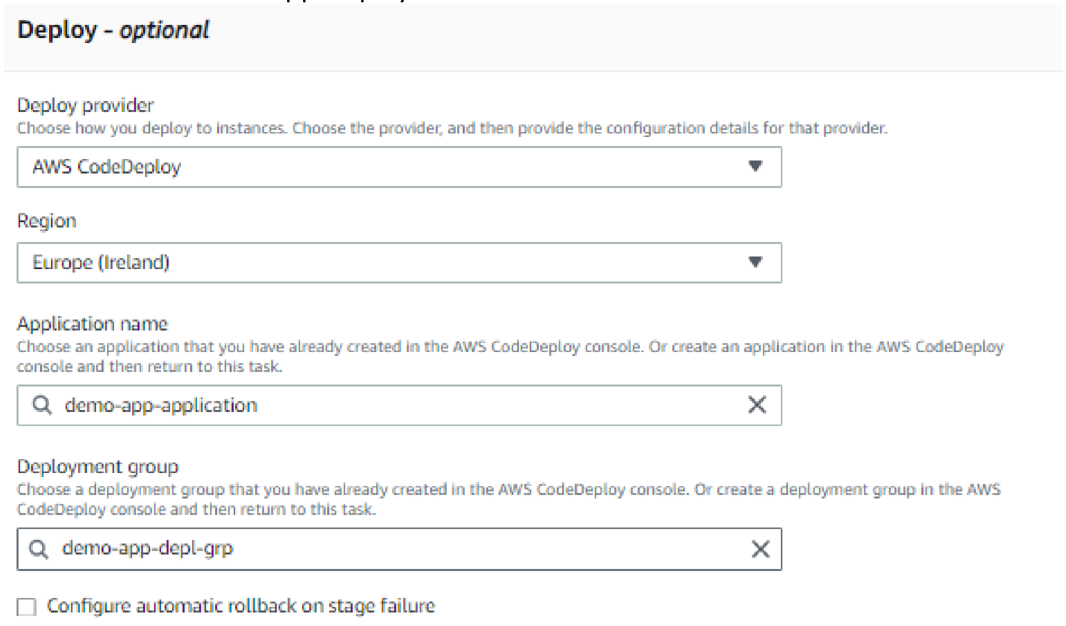
Project name
Choose a build project that you have already created in the AWS CodeBuild console. Or create a build project in the AWS CodeBuild console and then return to this task.
demo-build
or Create project

Environment variables - *optional*
Choose the key, value, and type for your CodeBuild environment variables. In the value field, you can reference variables generated by CodePipeline. Learn more

Add environment variable

Build type
[✓ Single build] Triggers a single build.
Batch build Triggers multiple builds as a single execution.

- **Deploy provider:** AWS CodeDeploy
- **Application name:** demo-app-deploy


Deploy - *optional*

Deploy provider
Choose how you deploy to instances. Choose the provider, and then provide the configuration details for that provider.
AWS CodeDeploy

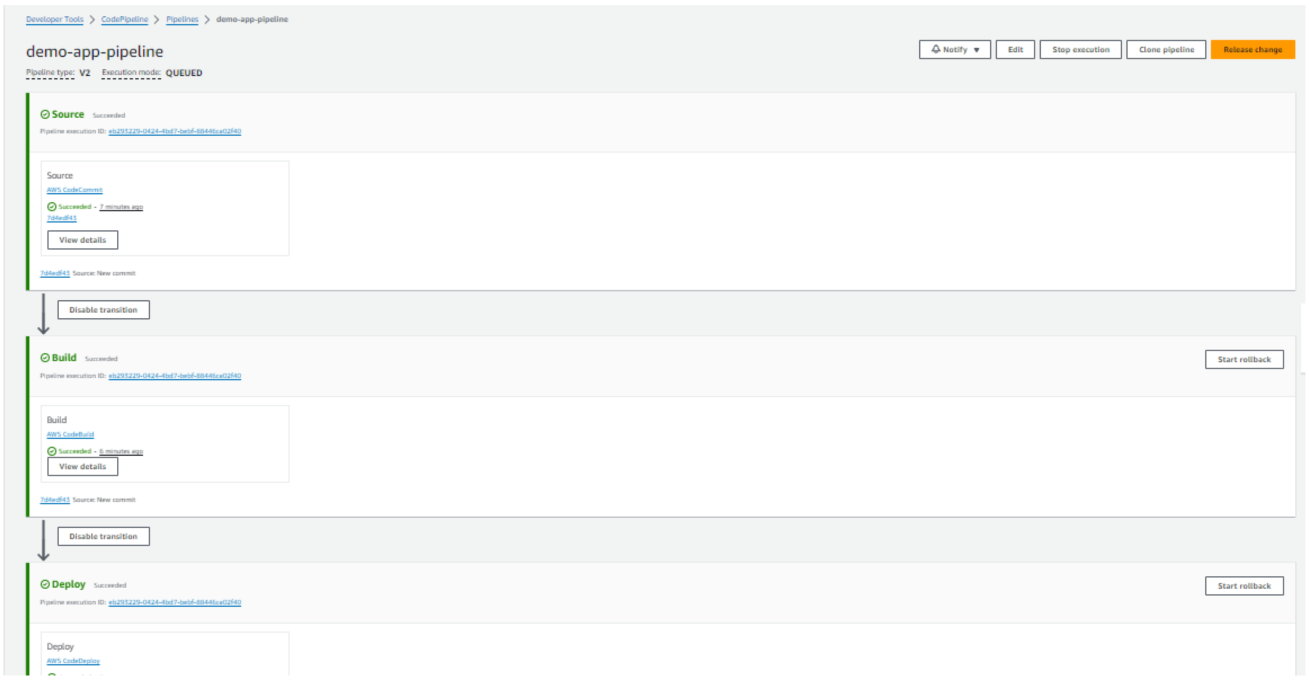
Region
Europe (Ireland)

Application name
Choose an application that you have already created in the AWS CodeDeploy console. Or create an application in the AWS CodeDeploy console and then return to this task.
demo-app-application

Deployment group
Choose a deployment group that you have already created in the AWS CodeDeploy console. Or create a deployment group in the AWS CodeDeploy console and then return to this task.
demo-app-depl-grp

☐ Configure automatic rollback on stage failure

- Once we start deploy, the pipeline gets executed and successfully completed.



Developer Tools > CodePipeline > Pipelines > demo-app-pipeline
Notify
Edit
Stop execution
Clone pipeline
Release change

demo-app-pipeline

Pipeline type: V2 Execution mode: QUEUED

Source Succeeded

Pipeline execution ID: eb293229-0424-4bd7-bebf-88446ca02f40

Source
AWS CodeCommit
Succeeded - 7 minutes ago
7d4edf43

7d4edf43 Source: New commit

Build Succeeded

Pipeline execution ID: eb293229-0424-4bd7-bebf-88446ca02f40

Build
AWS CodeBuild
Succeeded - 6 minutes ago

7d4edf43 Source: New commit

Deploy Succeeded

Pipeline execution ID: eb293229-0424-4bd7-bebf-88446ca02f40

Deploy
AWS CodeDeploy