

AWS DevOps Project

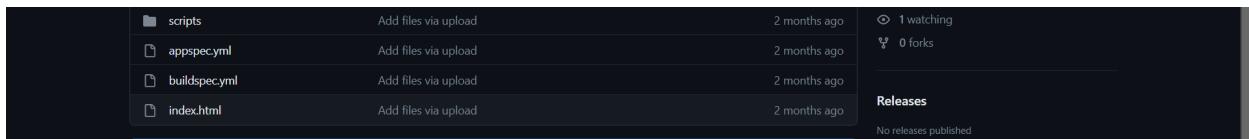
Problem Statement:

You are assigned to create a software development life cycle for an application your company has created. The company wants you to use AWS for the infrastructure part and AWS Developer tools for the pipeline part.

Tasks to be performed:

1. Create a website in any language of your choice and push the code into GitHub.
2. Migrate your GitHub repository into the AWS CodeCommit repository
3. Create two CodeDeploy deployments (for the QA stage and the Production stage) with an EC2 deployment group into which you can push the code from the CodeCommit repository
4. Using AWS CodePipeline, create a software development life cycle:
 - a. The source is the CodeCommit repository.
 - b. The code will be pushed into the deployment created in CodeDeploy.
 - c. There should be two stages in deployment, the QA stage and the Production stage
 - d. Only when the QA stage is successful, the Production stage should execute.
5. Create a third stage where the same website is pushed into an Elastic Beanstalk environment

According to the given project, the first thing we will need is an HTML website in the github repo.



GitHub Repo all files we will use, breakdown:

Index.html:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      body {
        font-family: Arial, sans-serif;
        background-color: #f2f2f2;
        color: #333;
        text-align: center;
      }

      h1 {
        font-size: 36px;
        margin-top: 50px;
        color: #6130e8;
      }

      p {
        font-size: 18px;
        margin: 20px 0;
      }
    </style>
  </head>
  <body>
    <h1>AWS Devops</h1>
    <p>Project</p>
  </body>
</html>
```

Index.html explained:

The given HTML code represents a simple webpage with a title "AWS DevOps" and a paragraph "Project". Here's a brief explanation of the code:

The `<!DOCTYPE html>` declaration specifies the HTML version used in the document.

The `<html>` element is the root element of an HTML page.

The `<head>` element contains meta-information about the HTML document, such as styles, scripts, and title.

Inside the `<head>` element, there is a `<style>` block that defines the CSS styles for the webpage.

The `<body>` element represents the content of the webpage and contains the visible elements.

The `<h1>` element represents a heading with the text "AWS DevOps". It has a larger font size, a margin-top of 50 pixels, and a specific color (#6130e8).

The `<p>` element represents a paragraph with the text "Project". It has a font size of 18 pixels and a margin of 20 pixels on the top and bottom.

The CSS styles specified in the `<style>` block define the appearance of the webpage, including the font family, background color, text color, and text alignment.

In summary, the HTML code creates a webpage with a heading and a paragraph, styled using CSS to have a specific appearance.

buildspec.yml:

The given buildspec.yml file is used in the AWS CodeBuild service to define the build specifications and actions for a project. Here's a brief explanation of the contents:

version: 0.2: Specifies the version of the buildspec file format.

phases: Represents different phases of the build process, such as installation, build, and post-build actions.

install: Contains commands to be executed during the installation phase.

`echo Installing NGINX`: Prints a message indicating that NGINX installation is starting.

`sudo apt-get update`: Updates the package lists on the system.

`sudo apt-get install nginx -y`: Installs NGINX by using the package manager with automatic confirmation (-y).

build: Contains commands to be executed during the build phase.

`echo Build started on date```: Prints a message with the current date and time.

`cp index.html /var/www/html/`: Copies the index.html file to the /var/www/html/ directory.

post_build: Contains commands to be executed after the build phase.

`echo Configuring NGINX`: Prints a message indicating that NGINX configuration is being performed.

artifacts: Specifies the artifacts to be generated and uploaded after the build phase.

files: Defines the files and directories to be included as artifacts.

'**/*': Includes all files and directories recursively, capturing everything in the build environment as artifacts.

In summary, this `buildspec.yml` file sets up an AWS CodeBuild project to install NGINX, copy an `index.html` file to the appropriate location, and generate artifacts that include all files and directories in the build environment.

appspec.yml:

version: 0.0

os: linux

files:

- **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

The given `appspec.yml` file provides configuration for the deployment process using AWS CodeDeploy. Here's a brief explanation of the contents:

version: 0.0: Specifies the version of the AppSpec file format.

os: linux: Specifies the operating system where the deployment will take place.

files: Defines the files to be deployed.

- **source: /:** Specifies the source directory or files to be deployed. In this case, it represents the root directory of the application.

destination: /var/www/html: Specifies the destination directory on the target instance where the files will be deployed. In this case, it points to the `/var/www/html` directory.

hooks: Defines the lifecycle event hooks, which are actions to be executed at specific stages of the deployment.

AfterInstall: Specifies the hook to be executed after the application files are installed.

- location: scripts/install_nginx.sh: Specifies the location of the script to be executed. In this case, it points to install_nginx.sh within the scripts directory.

timeout: 300: Sets the timeout duration for the script execution to 300 seconds (5 minutes).

runas: root: Specifies that the script should be executed as the root user.

ApplicationStart: Specifies the hook to be executed after the application is started.

- location: scripts/start_nginx.sh: Specifies the location of the script to be executed. In this case, it points to start_nginx.sh within the scripts directory.

timeout: 300: Sets the timeout duration for the script execution to 300 seconds (5 minutes).

runas: root: Specifies that the script should be executed as the root user.

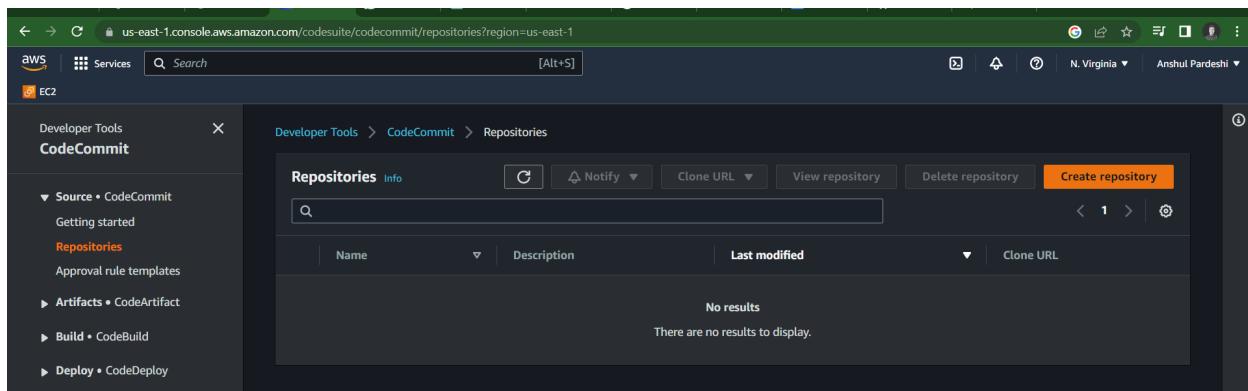
In summary, this appspec.yml file defines the deployment process for an application on a Linux operating system. It includes the files from the root directory of the application and deploys them to the /var/www/html directory. It also specifies the execution of two scripts, install_nginx.sh and start_nginx.sh, after the installation and application start stages, respectively. The scripts are executed as the root user with a timeout of 300 seconds for each.

Now we need HTML website in codecommit.

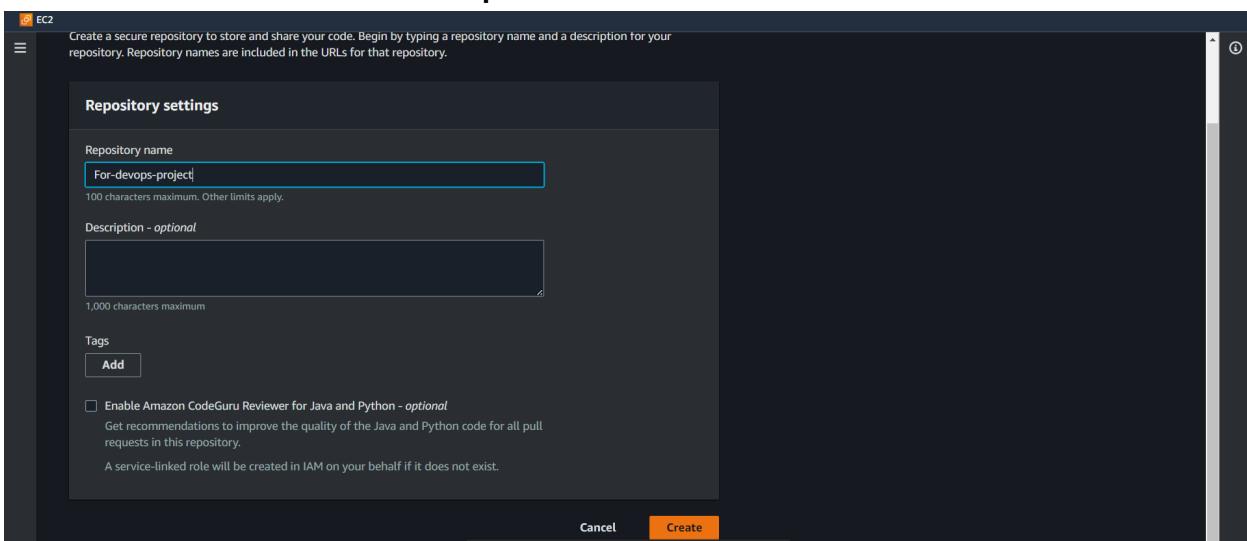
For this first create a codecommit repo in aws through console.

Search for codecommit service in aws console.

Click on create repository.

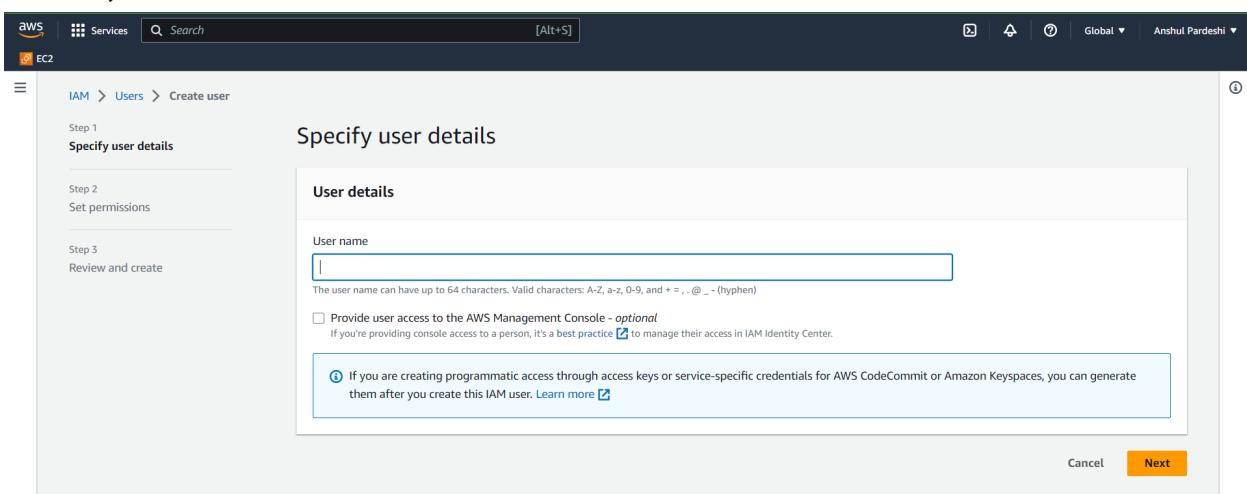


Just name it and click on create repo.

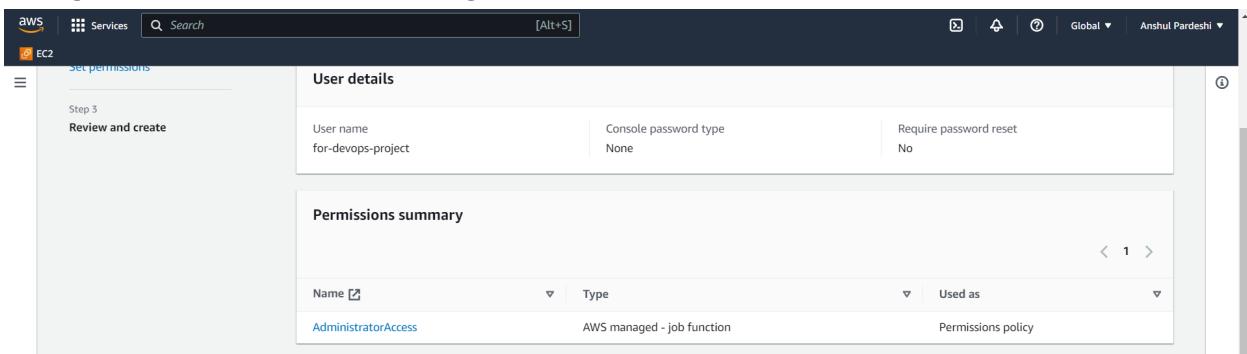


Now we want to clone github repo to codecommit.

For this, create a IAM user. Goto: IAM>User>Create User.



Just give a name to the user and give codecommit full access to it.



Now generate an IAM access key by going into: IAM>Users>User we created>security credentials>create access key

The screenshot shows the AWS IAM 'Create access key' interface. At the top, there's a navigation bar with 'AWS Services' and a search bar. Below it, the path 'IAM > Users > for-devops-project > Create access key' is visible. A sidebar on the left lists steps: Step 1 (Access key best practices & alternatives), Step 2 - optional (Set description tag), and Step 3 (Retrieve access keys). The main content area is titled 'Access key best practices & alternatives' with a note about avoiding long-term credentials. It lists five options with descriptions:

- Command Line Interface (CLI)
You plan to use this access key to enable the AWS CLI to access your AWS account.
- Local code
You plan to use this access key to enable application code in a local development environment to access your AWS account.
- Application running on an AWS compute service
You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.
- Third-party service
You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.
- Application running outside AWS
You plan to use this access key to enable an application running on an on-premises host, or to use a local AWS client or third-party AWS plugin.

GIT credentials required are created.

The screenshot shows the AWS IAM 'Credentials deleted' interface. On the left, a sidebar shows 'Identity and Access Management (IAM)'. Under 'Access management', 'Users' is selected. The main content area has a green header 'Credentials deleted.' with a 'Upload SSH public key' button. Below it, there are two sections:

- HTTPS Git credentials for AWS CodeCommit (1)**
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](#)
Actions ▾ Generate credentials
User name: for-devops-project-at-994272653862 | Created: 1 minute ago | Status: Active
- Credentials for Amazon Keyspaces (for Apache Cassandra) (0)**
Generate a user name and password you can use to authenticate to Amazon Keyspaces. You can have a maximum of two sets of credentials (active or inactive) at a time. [Learn more](#)
Actions ▾ Generate credentials
User name: | Created: | Status: No credentials | Generate credentials

Now in your local machine install git and clone the github repo to it.

For this use commands:

Sudo apt-get install git

Git clone <github repo clone link>

```
Get:24 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/main amd64 c-n-f Metadata [388 B]
Get:25 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/restricted amd64 c-n-f Metadata [116 B]
Get:26 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [23.4 kB]
Get:27 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe Translation-en [15.0 kB]
Get:28 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 c-n-f Metadata [548 B]
Get:29 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 c-n-f Metadata [116 B]
Get:30 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [403 kB]
Get:31 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [114 kB]
Get:32 http://security.ubuntu.com/ubuntu jammy-security/main amd64 c-n-f Metadata [9880 B]
Get:33 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [279 kB]
Get:34 http://security.ubuntu.com/ubuntu jammy-security/restricted Translation-en [40.8 kB]
Get:35 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [723 kB]
Get:36 http://security.ubuntu.com/ubuntu jammy-security/universe Translation-en [127 kB]
Get:37 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 c-n-f Metadata [14.7 kB]
Get:38 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [30.2 kB]
Get:39 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [5828 B]
Get:40 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 c-n-f Metadata [252 B]
Fetched 24.8 MiB in 4s (5561 kB/s)
Reading package lists... Done
ubuntu@ip-172-31-82-186:~$ git clone https://github.com/Anshuls-repo/for-devops-project-cloned.git
Cloning into 'for-devops-project-cloned'...
remote: Enumerating objects: 16, done.
remote: Counting objects: 100% (16/16), done.
remote: Compressing objects: 100% (13/13), done.
remote: Total 16 (delta 2), reused 16 (delta 2), pack-reused 0
Receiving objects: 100% (16/16), 84.93 KiB | 9.44 MiB/s, done.
Resolving deltas: 100% (2/2), done.
ubuntu@ip-172-31-82-186:~$ 
```

i-0c462ed654e5584e3 (forcodepush)
PublicIPs: 3.94.179.81 PrivateIPs: 172.31.82.186

Copy all the contents of this repo into another folder.

Use command:

cp -r /home/ubuntu/github cloned folder name/ /home/ubuntu/<new folder name>**

```
Receiving objects: 100% (16/16), 84.93 KiB | 9.44 MiB/s, done.
Resolving deltas: 100% (2/2), done.
ubuntu@ip-172-31-82-186:~$ git origin v
git: 'origin' is not a git command. See 'git --help'.
ubuntu@ip-172-31-82-186:~$ ls
for-devops-project-cloned
ubuntu@ip-172-31-82-186:~$ git clone https://git-codecommit.us-east-1.amazonaws.com/v1/repos/For-devops-project
Cloning into 'For-devops-project'...
Username for 'https://git-codecommit.us-east-1.amazonaws.com':
Password for 'https://git-codecommit.us-east-1.amazonaws.com':
fatal: Authentication failed for 'https://git-codecommit.us-east-1.amazonaws.com/v1/repos/For-devops-project/'
ubuntu@ip-172-31-82-186:~$ mkdir codecommit
ubuntu@ip-172-31-82-186:~$ pwd
/home/ubuntu
ubuntu@ip-172-31-82-186:~$ cd for-devops-project-cloned
ubuntu@ip-172-31-82-186:~/for-devops-project-cloned$ ls
images index.html
ubuntu@ip-172-31-82-186:~/for-devops-project-cloned$ pwd
/home/ubuntu/for-devops-project-cloned
ubuntu@ip-172-31-82-186:~/for-devops-project-cloned$ ~C
ubuntu@ip-172-31-82-186:~/for-devops-project-cloned$ cd ..
ubuntu@ip-172-31-82-186:~$ cp /home/ubuntu/for-devops-project-cloned/** /home/ubuntu/codecommit
cp: -r not specified; omitting directory '/home/ubuntu/for-devops-project-cloned/images'
ubuntu@ip-172-31-82-186:~$ cp -r /home/ubuntu/for-devops-project-cloned/** /home/ubuntu/codecommit
ubuntu@ip-172-31-82-186:~$ cd codecommit
ubuntu@ip-172-31-82-186:~/codecommit$ ls
images index.html
ubuntu@ip-172-31-82-186:~/codecommit$ 
```

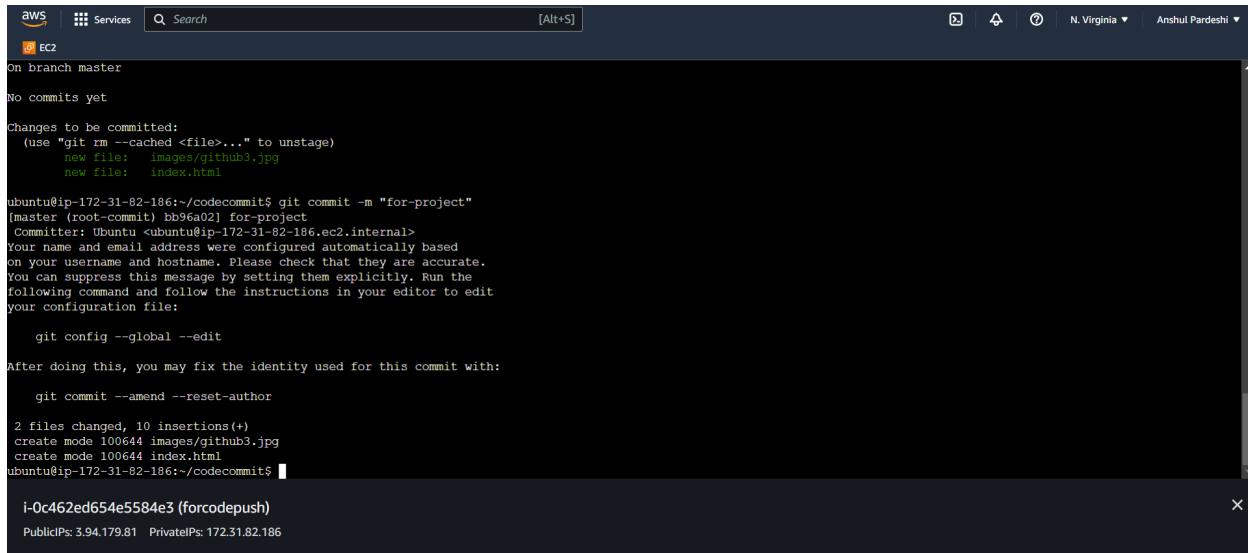
i-0c462ed654e5584e3 (forcodepush)
PublicIPs: 3.94.179.81 PrivateIPs: 172.31.82.186

Now push this new folder to codecommit repo. Use this command in that folder.

Git init

Git add .

Git commit -m "message."

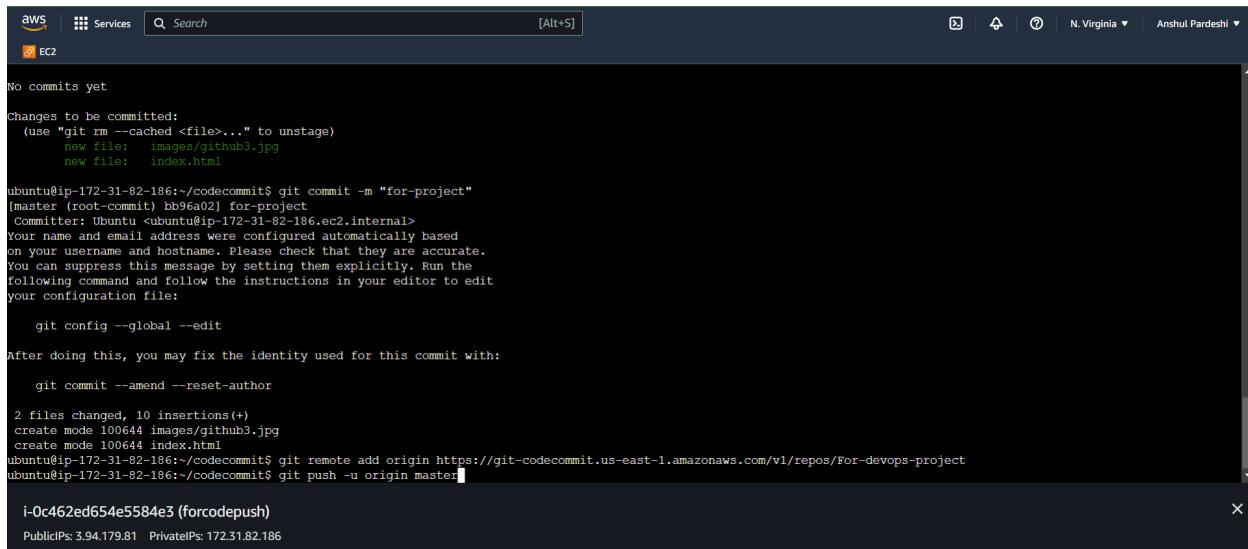


```
AWS Services Search [Alt+S] EC2 On branch master No commits yet Changes to be committed: (use "git rm --cached <file>..." to unstage) new file: images/github3.jpg new file: index.html ubuntu@ip-172-31-82-186:~/codecommit$ git commit -m "for-project" [master (root-commit) bb96a02] for-project Committer: Ubuntu <ubuntu@ip-172-31-82-186.ec2.internal> Your name and email address were configured automatically based on your username and hostname. Please check that they are accurate. You can suppress this message by setting them explicitly. Run the following command and follow the instructions in your editor to edit your configuration file: git config --global --edit After doing this, you may fix the identity used for this commit with: git commit --amend --reset-author 2 files changed, 10 insertions(+) create mode 100644 images/github3.jpg create mode 100644 index.html ubuntu@ip-172-31-82-186:~/codecommit$ i-0c462ed654e5584e3 (forcodepush) PublicIPs: 3.94.179.81 PrivateIPs: 172.31.82.186
```

Now let us push it to code commit.

Use command: git remote add origin <codecommit repo link>

Git push -u origin master



```
AWS Services Search [Alt+S] EC2 No commits yet Changes to be committed: (use "git rm --cached <file>..." to unstage) new file: images/github3.jpg new file: index.html ubuntu@ip-172-31-82-186:~/codecommit$ git commit -m "for-project" [master (root-commit) bb96a02] for-project Committer: Ubuntu <ubuntu@ip-172-31-82-186.ec2.internal> Your name and email address were configured automatically based on your username and hostname. Please check that they are accurate. You can suppress this message by setting them explicitly. Run the following command and follow the instructions in your editor to edit your configuration file: git config --global --edit After doing this, you may fix the identity used for this commit with: git commit --amend --reset-author 2 files changed, 10 insertions(+) create mode 100644 images/github3.jpg create mode 100644 index.html ubuntu@ip-172-31-82-186:~/codecommit$ git remote add origin https://git-codecommit.us-east-1.amazonaws.com/v1/repos/For-devops-project ubuntu@ip-172-31-82-186:~/codecommit$ git push -u origin master i-0c462ed654e5584e3 (forcodepush) PublicIPs: 3.94.179.81 PrivateIPs: 172.31.82.186
```

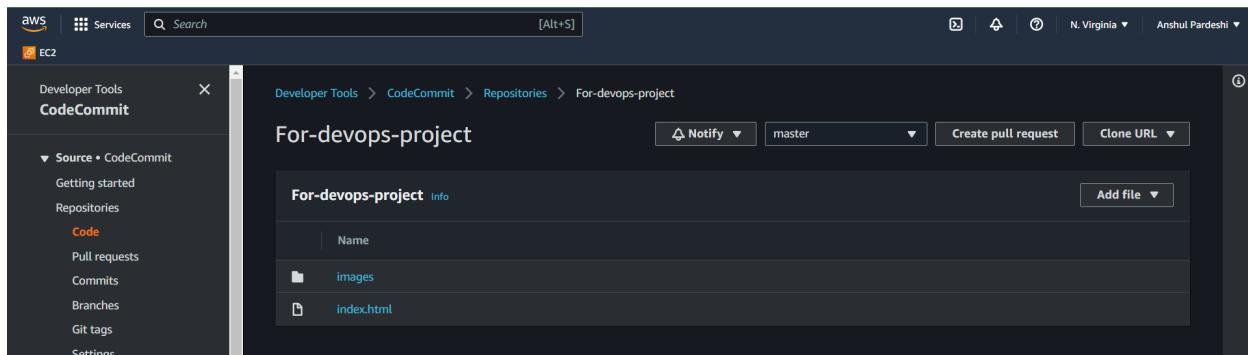
In credentials part, we will enter the credentials we created using IAM user before.

The screenshot shows a terminal window with the following text:

```
on your username and hostname. Please check that they are accurate.  
You can suppress this message by setting them explicitly. Run the  
following command and follow the instructions in your editor to edit  
your configuration file.  
git config --global --edit  
After doing this, you may fix the identity used for this commit with:  
git commit --amend --reset-author  
2 files changed, 10 insertions(+)  
create mode 100644 images/github3.jpg  
create mode 100644 index.html  
ubuntu@ip-172-31-82-186:~/codecommit$ git remote add origin https://git-codecommit.us-east-1.amazonaws.com/v1/repos/For-devops-project  
ubuntu@ip-172-31-82-186:~/codecommit$ git push -u origin master  
Username for 'https://git-codecommit.us-east-1.amazonaws.com': for-devops-project-at-994272653862  
Password for 'https://for-devops-project-at-994272653862@git-codecommit.us-east-1.amazonaws.com':  
Enumerating objects: 5, done.  
Counting objects: 100% (5/5), done.  
Compressing objects: 100% (4/4), done.  
Writing objects: 100% (5/5), 82.46 KiB | 13.74 MiB/s, done.  
total 5 (delta 0), reused 0 (delta 0), pack-reused 0  
remote: Validating objects: 100%  
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/For-devops-project  
 * [new branch] master -> master  
Branch 'master' set up to track remote branch 'master' from 'origin'.  
ubuntu@ip-172-31-82-186:~/codecommit$
```

i-0c462ed654e5584e3 (forcodepush)
Public IPs: 3.94.179.81 Private IPs: 172.31.82.186

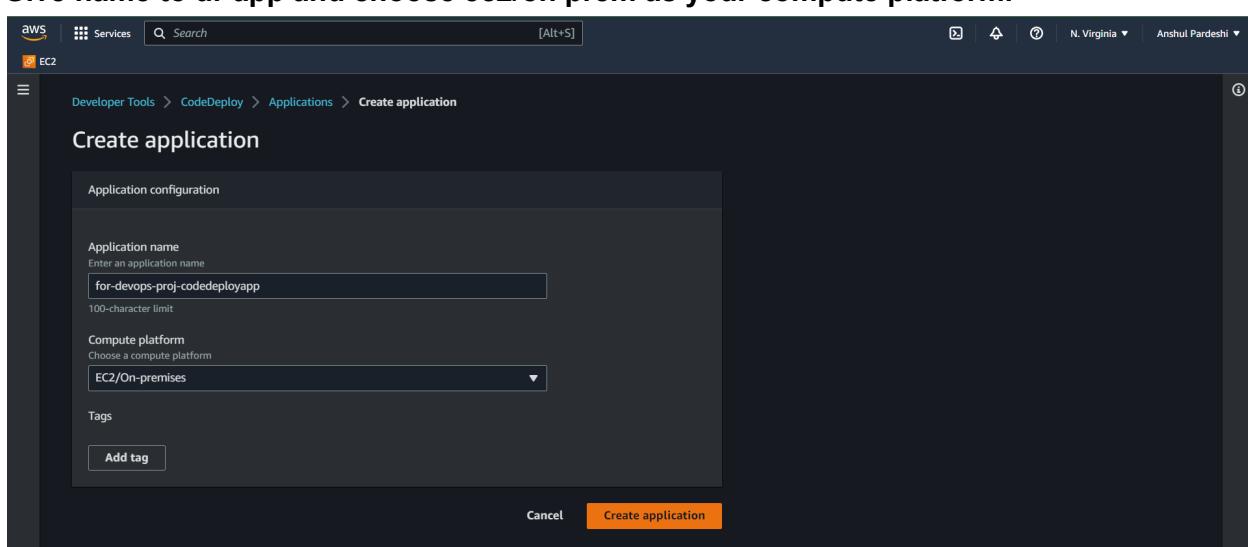
Changes has been reflected in our codecommit repo.



Now let us create 2 deployments, QA and Prod in EC2 deployment groups.

Goto AWS CodeDeploy>Create Application from AWS console.

Give name to ur app and choose ec2/on prem as your compute platform.



Now create 2 instances for QA and Prod. Also we will attach role to it so that it can access codecommit.

The screenshot shows the AWS EC2 Instances page. The left sidebar has 'New EC2 Experience' selected. The main table lists three instances:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
forcodepush	i-0c462ed654e5584e5	Running	t2.micro	2/2 checks passed	No alarms	us-east-1d	ec2-3-94-179-81
QA	i-095e35ee74b83f7c8	Running	t2.micro	Initializing	No alarms	us-east-1a	ec2-54-83-80-19
PROD	i-01efabe8e4b771a82	Running	t2.micro	Initializing	No alarms	us-east-1a	ec2-34-228-167-

Now we need to install CodeDeploy Agent into them.

Use these commands:

```
sudo apt-get install ruby -y
sudo apt-get install wget -y
cd /home/ubuntu # Change the directory to your desired location
wget https://aws-codedeploy-us-east-1.s3.us-east-1.amazonaws.com/latest/install
chmod +x ./install
sudo ./install auto
sudo service codedeploy-agent start
```

So code deploy agents are up and running in both qa and prod instances after using above commands.

```
Reading state information... done
Selecting previously unselected package codedeploy-agent.
(Reading database ... 67417 files and directories currently installed.)
Preparing to unpack .../codedeploy-agent_1.6.0-49_all.deb ...
Unpacking codedeploy-agent (1.6.0-49) ...
Setting up codedeploy-agent (1.6.0-49) ...
codedeploy-agent.service is not a native service, redirecting to systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable codedeploy-agent
I, [2023-05-26T08:20:37.928802 #2373] INFO -- : Update check complete.
I, [2023-05-26T08:20:37.931833 #2373] INFO -- : Stopping updater.
ubuntu@ip-172-31-23-95:~$ sudo service codedeploy-agent start
ubuntu@ip-172-31-23-95:~$ sudo service codedeploy-agent status
● codedeploy-agent.service - LSB: AWS CodeDeploy Host Agent
   Loaded: loaded (/etc/init.d/codedeploy-agent; generated)
   Active: active (running) since Fri 2023-05-26 08:20:37 UTC; 1min 49s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 2621 ExecStart=/etc/init.d/codedeploy-agent start (code=exited, status=0/SUCCESS)
    Tasks: 2 (limit: 1141)
   Memory: 58.7M
      CPU: 1.108s
     CGroup: /system.slice/codedeploy-agent.service
             └─2628 "codedeploy-agent: master 2628"
                  ├─2630 "codedeploy-agent: InstanceAgent:Plugins::CodeDeployPlugin::CommandPoller of master 2628"
May 26 08:20:37 ip-172-31-23-95 systemd[1]: Starting LSB: AWS CodeDeploy Host Agent...
May 26 08:20:37 ip-172-31-23-95 codedeploy-agent[2621]: Starting codedeploy-agent:
May 26 08:20:37 ip-172-31-23-95 systemd[1]: Started LSB: AWS CodeDeploy Host Agent.
ubuntu@ip-172-31-23-95:~$
```

Let us create a Deployment Group

The screenshot shows the AWS CodeDeploy application details page for the application 'for-devops-proj-codedeployapp'. The left sidebar has a 'CodeDeploy' section under 'Deploy'. The main area shows the application details with a name 'for-devops-proj-codedeployapp' and a compute platform 'EC2/On-premises'. Below this, the 'Deployment groups' tab is selected, showing a table with one entry: 'No deployment groups'. A note says 'Before you can deploy your application using CodeDeploy, you must create a deployment group.' A 'Create deployment group' button is at the bottom.

Give it a name and attach role to it with proper permissions.

The screenshot shows the 'Create deployment group' wizard. Step 1: Application selection. It lists the application 'for-devops-proj-codedeployapp' with a compute type 'EC2/On-premises'. Step 2: Deployment group name. It shows a text input field with 'For-devops-project' entered. Step 3: Service role. It shows a text input field with 'arn:aws:iam::994272653862:role/forcodepush' entered. Step 4: Deployment type. It shows a text input field with 'Choose how to deploy your application' entered.

Add tags of QA and Prod instance and create deployment group.

The screenshot shows the 'Environment configuration' section of the AWS EC2 Deployment Groups configuration. It includes fields for selecting Amazon EC2 instances, defining tag groups, and specifying On-premises instances. A note indicates that up to three tag groups can be defined per deployment group.

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
Q Name	X
Q Name	X

Add tag **+ Add tag group**

On-premises instances

Matching instances

We need to create CodeBuild as well. Also the build is stored in s3.
Let us create s3 bucket for this.

The screenshot shows the 'Create New Bucket' wizard. It prompts for a bucket name ('For-devops-project'), specifies the AWS Region ('US East (N. Virginia) us-east-1'), and provides options for copying settings from an existing bucket or choosing one. The 'Object Ownership' section shows 'ACLS disabled (recommended)' is selected. The 'Block Public Access settings for this bucket' section is also visible.

Bucket name: For-devops-project

AWS Region: US East (N. Virginia) us-east-1

Copy settings from existing bucket - optional

Choose bucket

Object Ownership Info

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

ACLs disabled (recommended)
All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

ACLs enabled
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

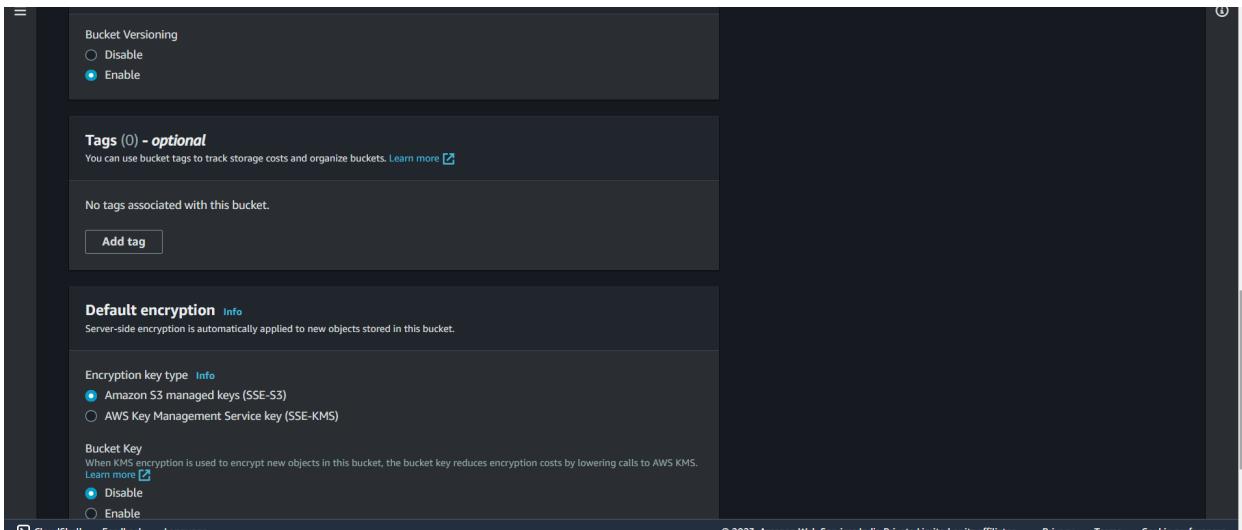
Object Ownership
Bucket owner enforced

Block Public Access settings for this bucket

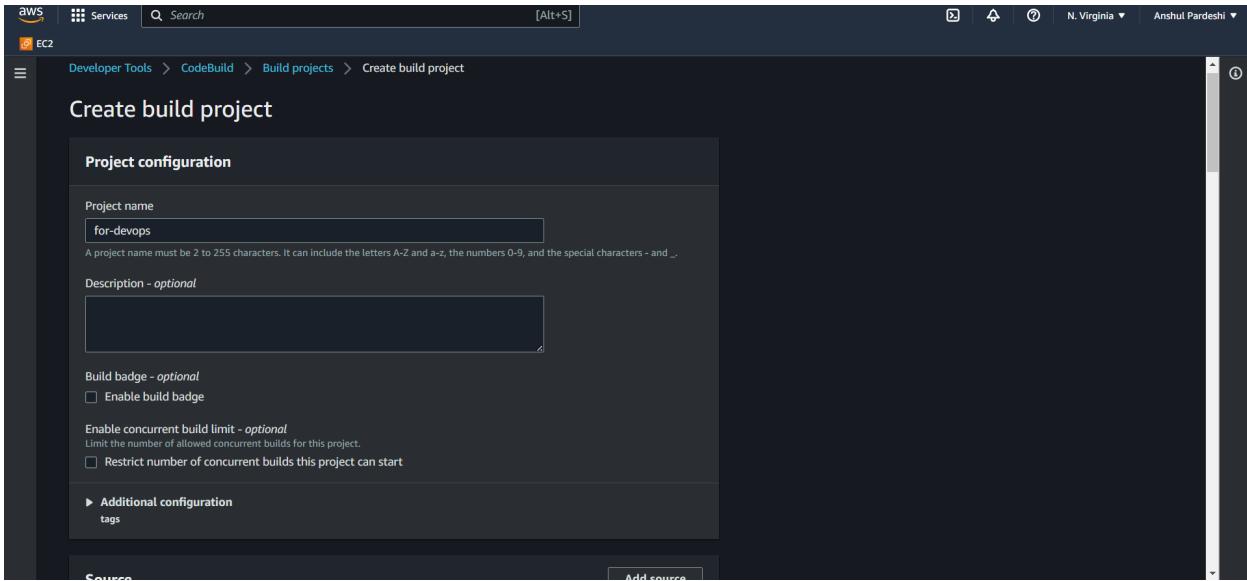
Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket

Block all public access.

Create a bucket.



Now let us create a build project. First just give name to build project.



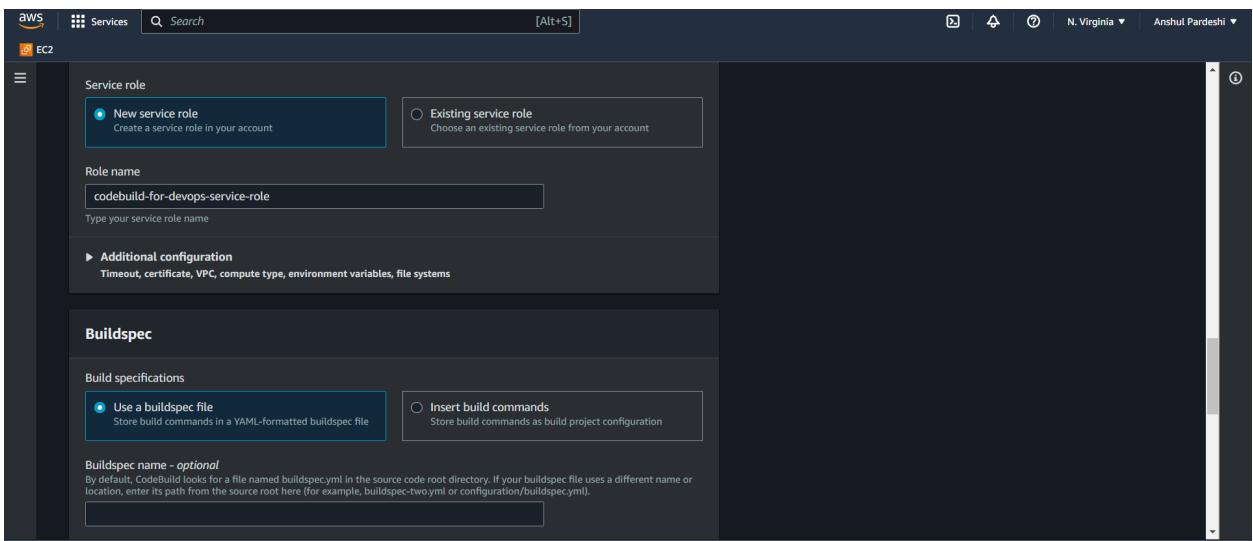
Select your source repo and branch.

The screenshot shows the 'Source' configuration page for a build project. The 'Source provider' is set to 'AWS CodeCommit'. The 'Repository' is 'for-devops-project'. The 'Reference type' is set to 'Branch', with 'master' selected as the branch. A 'Commit ID - optional' field is present but empty. Below the repository details, there is a 'Source version info' section showing 'refs/heads/master' and a commit hash '95a8e746 main project'. There is also a link to 'Additional configuration'.

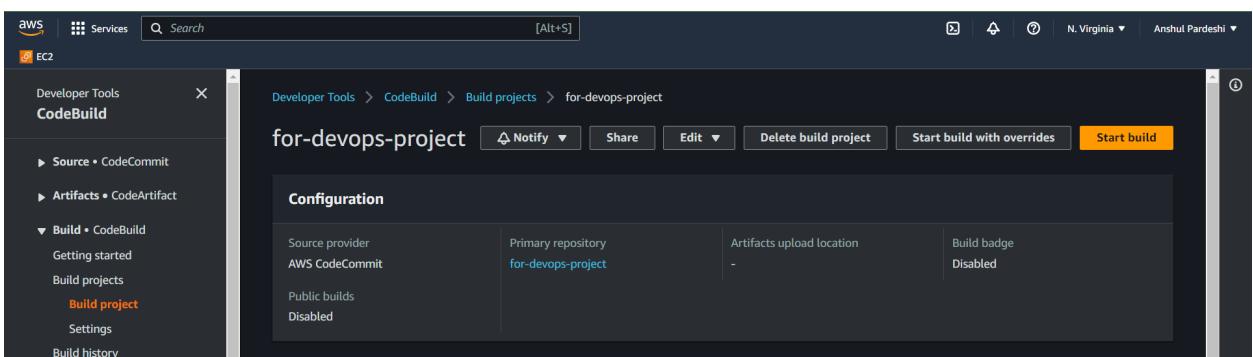
Select environment

The screenshot shows the 'Environment' configuration page. The 'Runtime(s)' is set to 'Standard'. The 'Image' is 'aws/codebuild/standard:5.0'. The 'Image version' dropdown is set to 'Always use the latest image for this runtime version'. The 'Environment type' is 'Linux'. Under 'Privileged', there is a checkbox for 'Enable this flag if you want to build Docker images or want your builds to get elevated privileges', which is unchecked. The 'Service role' section shows 'New service role' selected (with a note to 'Create a service role in your account') and 'Existing service role' (with a note to 'Choose an existing service role from your account'). The 'Role name' is 'codebuild-for-devops-service-role'. At the bottom of the page, there are links for CloudShell, Feedback, Language, and a footer with copyright information and links for Privacy, Terms, and Cookie preferences.

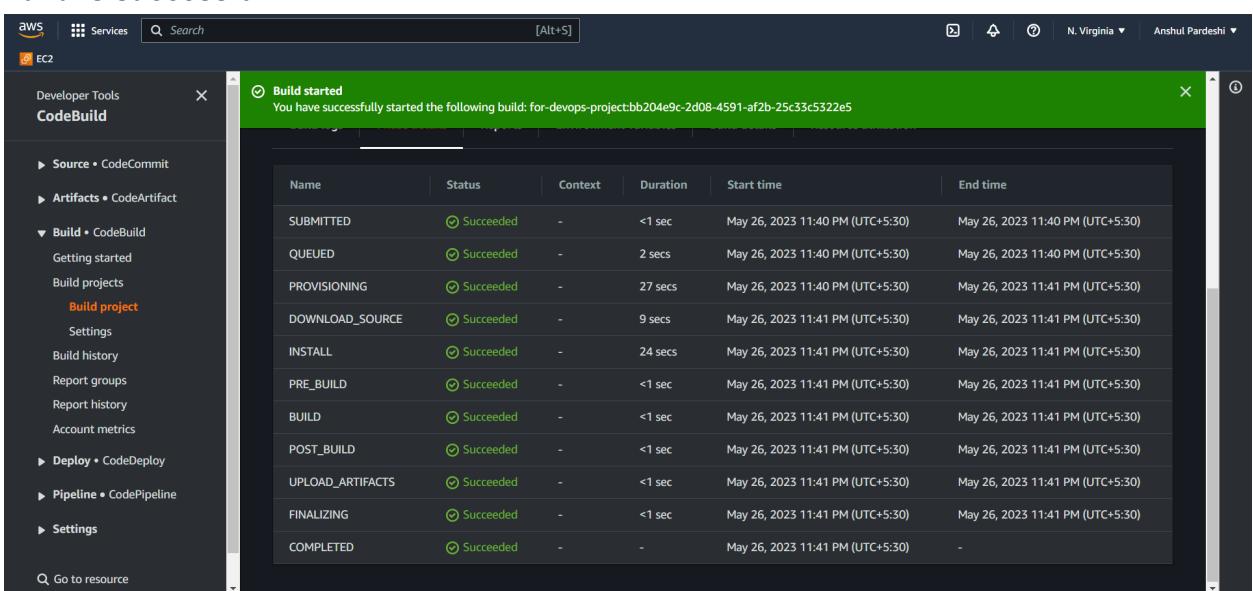
We will add artifacts later. Create build.



Click on start build



Build is successful



Let us add s3 as artifact.

Goto build and click on edit and add artifacts.

The screenshot shows the AWS CodeBuild console. On the left, a sidebar menu for 'CodeBuild' is open, showing options like 'Source', 'Artifacts', 'Build', and 'Build project'. The main area displays a 'Configuration' section for a build project named 'for-devops-project'. The configuration includes fields for 'Source provider' (AWS CodeCommit), 'Primary repository' (for-devops-project), 'Public builds' (Disabled), and 'Build badge' (Disabled). At the top right, there are buttons for 'Edit', 'Delete build project', 'Start build with overrides', and 'Start build'.

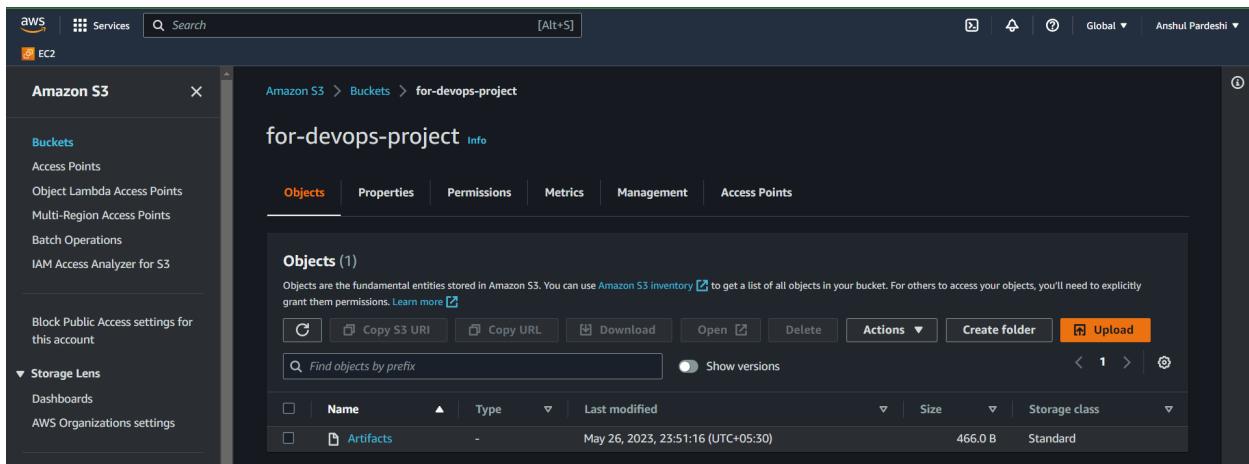
Choose the bucket in which you want artifact to be in.

The screenshot shows the 'Add artifact' screen. It has a 'Type' dropdown set to 'Amazon S3'. The 'Bucket name' field contains 'for-devops-project'. Below these, there's a 'Name' field with 'Artifacts' and an unchecked checkbox for 'Enable semantic versioning'. A 'Path - optional' field is present, though empty. At the bottom, there's a note about 'Namespace type - optional'.

Let us package it in .zip

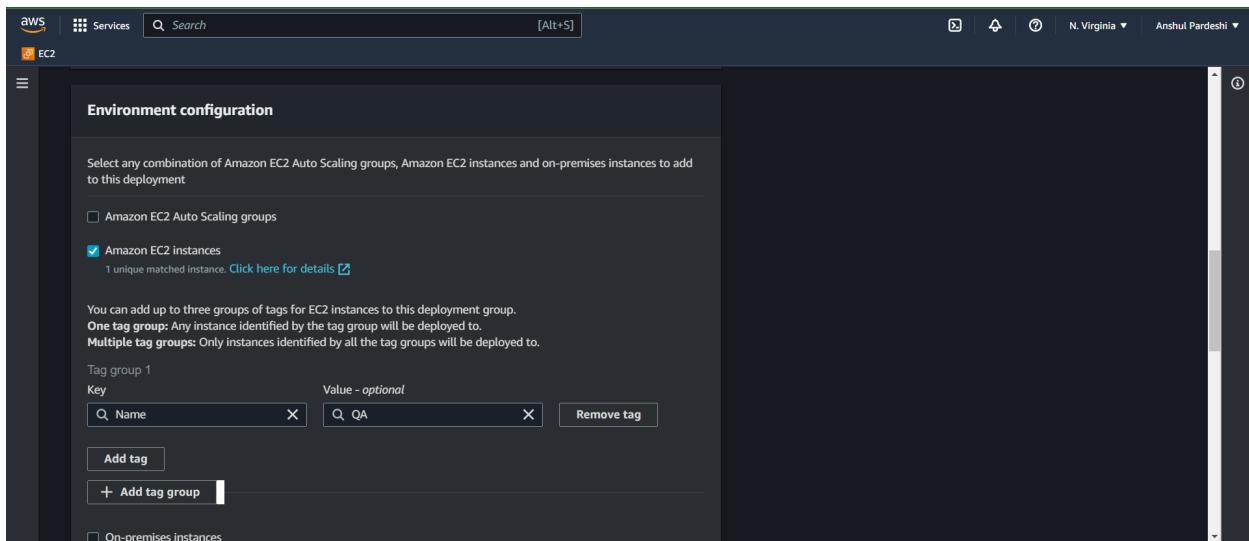
The screenshot shows the 'Additional configuration' screen. It includes sections for 'Namespace type - optional' (set to 'None'), 'Artifacts packaging' (with 'Zip' selected, indicating AWS CodeBuild will upload artifacts into a compressed file), 'Disable artifact encryption' (unchecked), and 'Service role permissions' (with a checked checkbox for 'Allow AWS CodeBuild to modify this service role so it can be used with this build project'). There's also a link for 'Additional configuration' at the bottom.

Now start build after setting up artifacts.
You can see that after successful build .zip file will reflect in s3.



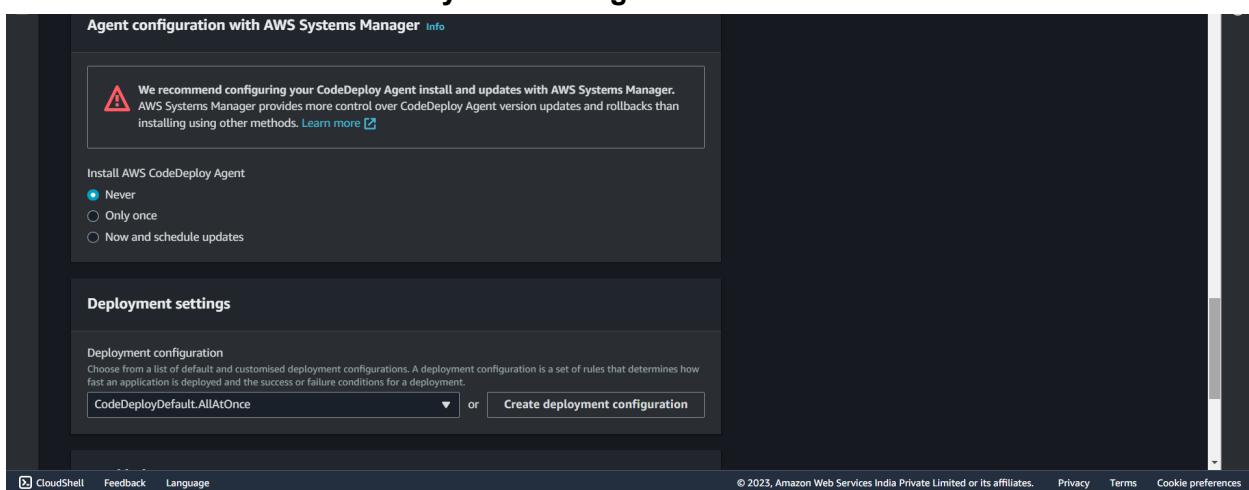
The screenshot shows the AWS Amazon S3 console. On the left, the sidebar includes 'Buckets', 'Access Points', 'Object Lambda Access Points', 'Multi-Region Access Points', 'Batch Operations', 'IAM Access Analyzer for S3', 'Block Public Access settings for this account', 'Storage Lens', 'Dashboards', and 'AWS Organizations settings'. The main area displays the 'for-devops-project' bucket. The 'Objects' tab is selected, showing one object named 'Artifacts'. Below the object list are buttons for 'Copy S3 URI', 'Copy URL', 'Download', 'Open', 'Delete', 'Actions', 'Create folder', and 'Upload'. A search bar for 'Find objects by prefix' and a 'Show versions' toggle are also present.

Now we have created CodeDEPLOY Application before.
Create 2 codedeploy groups in it. Here we choose QA tags



The screenshot shows the 'Environment configuration' section of the AWS CodeDeploy application configuration. It lists 'Amazon EC2 Auto Scaling groups' and 'Amazon EC2 instances'. The 'Amazon EC2 instances' checkbox is checked. Below it, a note says 'You can add up to three groups of tags for EC2 instances to this deployment group.' It defines 'One tag group': 'Any instance identified by the tag group will be deployed.' and 'Multiple tag groups': 'Only instances identified by all the tag groups will be deployed to.' A 'Tag group 1' section shows a key 'Name' with value 'QA' and a 'Remove tag' button. Buttons for 'Add tag' and '+ Add tag group' are also visible. A note at the bottom left says 'On-premises instances'.

Select never because we already installed agents before.



The screenshot shows the 'Agent configuration with AWS Systems Manager' section. It includes a note: 'We recommend configuring your CodeDeploy Agent install and updates with AWS Systems Manager. AWS Systems Manager provides more control over CodeDeploy Agent version updates and rollbacks than installing using other methods.' It features a radio button for 'Install AWS CodeDeploy Agent' with options: 'Never' (selected), 'Only once', and 'Now and schedule updates'. Below this is the 'Deployment settings' section, which includes a dropdown for 'Deployment configuration' set to 'CodeDeployDefault.AllAtOnce' and a 'Create deployment configuration' button.

Similarly, while creating deployment group for prod select prod instance tags.

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
1 unique matched instance. [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 PROD Remove tag

Add tag + Add tag group

On-premises instances

Matching instances
1 unique matched instance. [Click here for details](#)

Let us create deployment for QA group first.

Developer Tools > CodeDeploy > Applications > for-devops-project

for-devops-project

Application details

Name: for-devops-project Compute platform: EC2/On-premises

Deployments Deployment groups Revisions

Application deployment history

View details Actions Copy deployment Retry deployment Create deployment

Deployment ID	Status	Deployment type	Deployment group	Revision location	Initiating event	Start time	End time

In Deployment settings add artifacts s3 url and file type.

Application: for-devops-project

Deployment group: QA

Compute platform: EC2/On-premises

Deployment type: In-place

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

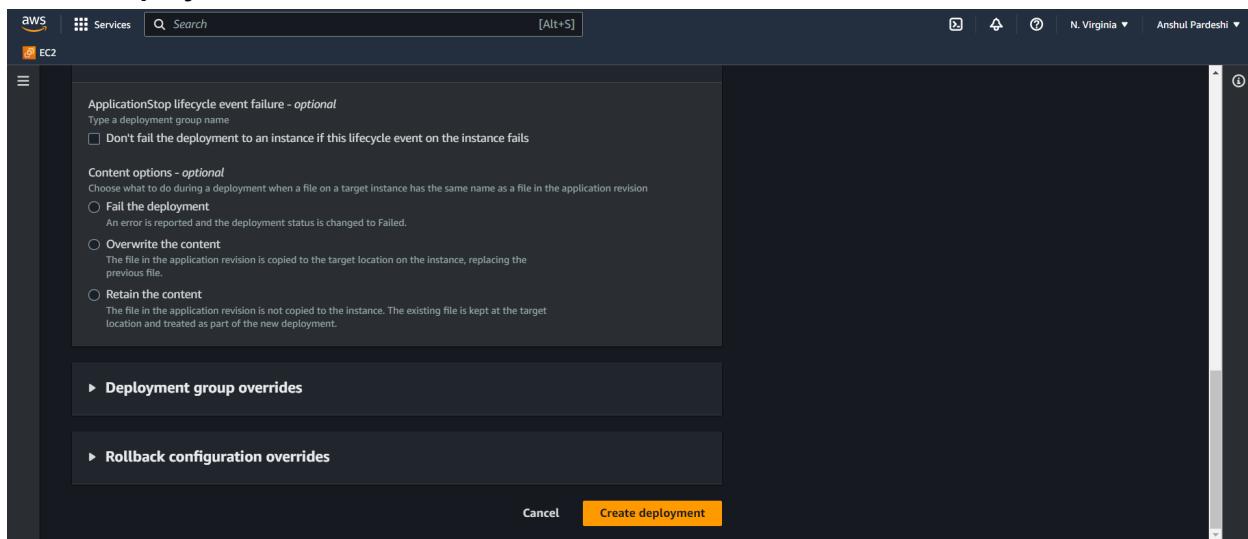
s3://for-devops-project/artifacts

s3://bucket-name/folder/object.[zip|tar|tgz]

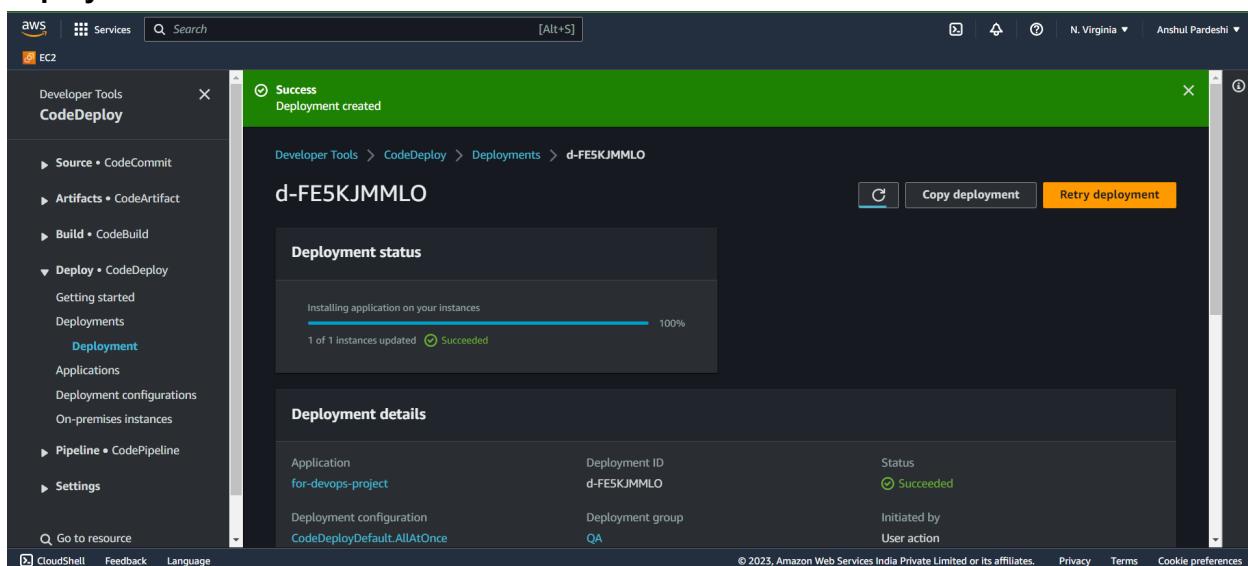
Revision file type:

.zip

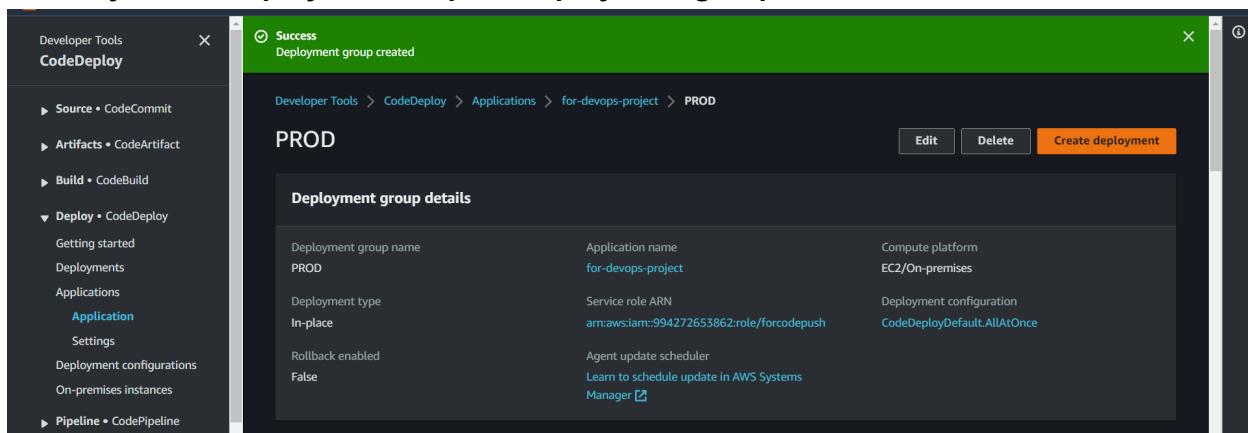
Create Deployment



Deployment succeeded.



Similarly create deployment for prod deployment group.



PROD deployment is also done.

The screenshot shows the AWS CodeDeploy console. On the left, a sidebar menu includes options like Source, Artifacts, Build, Deploy, Pipeline, Settings, and Go to resource. The main area displays a deployment named 'd-OSFM5RMLO' with a status of 'Success'. A message at the top says 'Deployment created'. Below it, the 'Deployment status' section shows a progress bar at 100% completion with the message '1 of 1 instances updated' and a 'Succeeded' status. The 'Deployment details' section provides information: Application ('for-devops-project'), Deployment ID ('d-OSFM5RMLO'), Status ('Succeeded'), Deployment configuration ('CodeDeployDefault.AllAtOnce'), Deployment group ('PROD'), and Initiated by ('User action').

We can copy public ip and check both the instances deployments.

Two browser windows are shown. Both have a header with a back arrow, forward arrow, refresh button, and a URL bar indicating 'Not secure'. The first window has a URL of '3.84.91.108' and the second has '54.92.245.160'. Both windows display the text 'AWS DevOps' and 'PROJECT' below it. The first window's URL is highlighted in blue.

Now, we need to create a pipeline.

Create pipeline:

The screenshot shows the AWS CodePipeline console. The sidebar menu includes Source, Artifacts, Build, Deploy, Pipeline, and Getting started. The main area shows a list of pipelines under the heading 'Pipelines'. A search bar is at the top of the list. Below it, there is a table with columns for Name, Most recent execution, Latest source revisions, and Last executed. A message at the bottom center says 'No results' and 'There are no results to display.' An orange 'Create pipeline' button is located at the top right of the list area.

Give it a name and let the role be default one and next.

The screenshot shows the 'Pipeline settings' step of the AWS CodePipeline setup. On the left, a sidebar lists steps: Step 2 (Add source stage), Step 3 (Add build stage), Step 4 (Add deploy stage), Step 5 (Review). The main area is titled 'Pipeline settings'. It contains fields for 'Pipeline name' (set to 'For-devops-project'), 'Service role' (radio button selected for 'New service role'), 'Role name' (set to 'AWSCodePipelineServiceRole-us-east-1-For-devops-project'), and a checked checkbox for 'Allow AWS CodePipeline to create a service role so it can be used with this new pipeline'.

Our source provider is codecommit so select it.

The screenshot shows the 'Source' configuration step. The sidebar shows steps 2 through 5. The main area is titled 'Source' and includes fields for 'Source provider' (set to 'AWS CodeCommit'), 'Repository name' (set to 'for-devops-project'), and 'Branch name' (set to 'master'). Under 'Change detection options', the 'AWS CodePipeline' option is selected. Under 'Output artifact format', the 'CodePipeline default' option is selected.

Our build provider is codebuild. Select the build that we created.

The screenshot shows the 'Add build stage' step. The sidebar shows steps 1 through 5. The main area is titled 'Build - optional' and includes fields for 'Build provider' (set to 'AWS CodeBuild'), 'Region' (set to 'US East (N. Virginia)'), 'Project name' (set to 'for-devops-project'), and an 'Environment variables - optional' section. Under 'Build type', the 'Single build' option is selected.

Our deploy provider is codedeploy. Select QA for now. We will add Prod in pipeline later.

The screenshot shows the 'Add deploy stage' step in a pipeline creation process. On the left, a sidebar lists steps: Step 1 (Choose pipeline settings), Step 2 (Add source stage), Step 3 (Add build stage), Step 4 (Add deploy stage), and Step 5 (Review). The main panel is titled 'Add deploy stage' with an 'Info' link. It contains a section titled 'Deploy - optional' with a dropdown for 'Deploy provider' set to 'AWS CodeDeploy'. Below it are fields for 'Region' (set to 'US East (N. Virginia)') and 'Application name' (set to 'for-devops-project'). A search bar for 'Deployment group' is also present, with 'QA' selected. At the bottom are 'Cancel', 'Previous', 'Skip deploy stage', and 'Next' buttons.

Create pipeline.

The screenshot shows the final step of creating a pipeline. The sidebar still shows Step 4 (Add deploy stage). The main panel is titled 'Step 4: Add deploy stage' and contains a 'Deploy action provider' section. It lists 'Deploy action provider' as 'AWS CodeDeploy', 'ApplicationName' as 'for-devops-project', and 'DeploymentGroupName' as 'QA'. At the bottom are 'Cancel', 'Previous', and 'Create pipeline' buttons. The 'Create pipeline' button is highlighted with a yellow background.

Current pipeline is working fine.

The screenshot shows the AWS CodePipeline console. On the left, a sidebar navigation includes 'Source' (CodeCommit), 'Artifacts' (CodeArtifact), 'Build' (CodeBuild), 'Deploy' (CodeDeploy), and 'Pipeline' (CodePipeline). Under Pipeline, there are 'Getting started', 'Pipelines', 'Pipeline' (selected), 'History', 'Settings', and 'Settings'. A 'Create a notification rule for this pipeline' button is visible at the top right. The main area displays a green 'Success' banner stating 'Congratulations! The pipeline For-devops-project has been created.' Below it, the pipeline's status is shown: 'Source' succeeded (AWS CodeCommit, bcf289b5, Succeeded - 1 minute ago). The pipeline ID is 52076da0-8b87-42e2-8bc7-aa964adb4941. The URL is https://us-east-1.console.aws.amazon.com/codesuite/codepipeline/pipelines/for-devops-project/edit?region=us-east-1.

What we want is after QA is successful PROD should get triggered.

Also if PROD is successful Beanstalk should update the code.

Let us create BeanStalk environment first.

The screenshot shows the AWS Elastic Beanstalk console home page. It features a dark header with the AWS logo, 'Services', 'Search', and 'N. Virginia' region. The main content area has a dark background with white text. It includes a 'Compute' section, a large 'Amazon Elastic Beanstalk' title, and a sub-section 'End-to-end web application management.'. Below this is a 'Get started' section with a 'Create application' button. Another 'Get started' section provides information about automatic deployment handling. To the right, a 'Pricing' section states there's no additional charge for Elastic Beanstalk. The URL is https://us-east-1.console.aws.amazon.com/elasticbeanstalk/home?region=us-east-1#/welcome.

Give name to your application. Select php as platform.

The screenshot shows the 'Create New Application' wizard in the AWS Elastic Beanstalk console. The 'Platform type' section has 'Managed platform' selected. In the 'Platform' section, 'PHP' is chosen from the dropdown. Below that, 'Platform branch' is set to 'PHP 8.1 running on 64bit Amazon Linux 2' and 'Platform version' is set to '3.5.7 (Recommended)'. The 'Application code' section shows 'Sample application' selected. A note indicates that existing versions can be uploaded or uploaded directly from S3.

Click on next.

The screenshot shows the 'Create New Application' wizard in the AWS Elastic Beanstalk console, step 2. The 'Application code' section shows 'Sample application' selected. The 'Presets' section is visible below it, showing configuration presets like 'Single instance (free tier eligible)' selected. At the bottom right, there are 'Cancel' and 'Next' buttons.

Give necessary permission through role and select instance details.

The screenshot shows the 'Create New Application' wizard in the AWS Elastic Beanstalk console, step 3. The 'Configure service access' section includes 'Step 3 - optional' (Set up networking, database, and tags), 'Step 4 - optional' (Configure instance traffic and scaling), 'Step 5 - optional' (Configure updates, monitoring, and logging), and 'Step 6' (Review). The main area shows 'Service role' selected as 'Use an existing service role' with 'aws-elasticbeanstalk-service-role' chosen. 'Existing service roles' and 'EC2 key pair' fields are also present. The 'EC2 instance profile' field is set to 'aws-elasticbeanstalk-ec2-role'. A 'View permission details' button is at the bottom.

Choose VPC and AZ in which you want the instance.

VPC
Launch your environment in a custom VPC instead of the default VPC. You can create a VPC and subnets in the VPC management console.
Learn more [\[Link\]](#)
vpc-0e42b04dcc524de2b | (172.31.0.0/16)

Create custom VPC [\[Link\]](#)

Instance settings
Choose a subnet in each AZ for the instances that run your application. To avoid exposing your instances to the Internet, run your instances in private subnets and load balancer in public subnets. To run your load balancer and instances in the same public subnets, assign public IP addresses to the instances. Learn more [\[Link\]](#)

Public IP address
Assign a public IP address to the Amazon EC2 instances in your environment.
 Activated

Instance subnets

	Availability Zone	Subnet	CIDR	Name
<input checked="" type="checkbox"/>	us-east-1a	subnet-05aac4393...	172.31.16.0/20	
<input checked="" type="checkbox"/>	us-east-1e	subnet-0682315c4...	172.31.48.0/20	

Select storage type and other specs needed according to you.

arm64 - new
This architecture uses AWS Graviton2 processors. You might have to recompile some third-party tools and libraries.

Instance types
Add instance types for your fleet. Change the order that the instances are in to set the preferred launch order. This only affects On-Demand instances. We recommend you include at least two instance types. Learn more [\[Link\]](#)
Choose x86 instance types [\[Link\]](#)

t3.micro [X](#) t3.small [X](#)

AMI ID
Elastic Beanstalk selects a default Amazon Machine Image (AMI) for your environment based on the Region, platform version, and processor architecture that you choose. Learn more [\[Link\]](#)
ami-0ca42562f43556e13

Availability Zones
Number of Availability Zones (AZs) to use.
Any

Placement
Specify Availability Zones (AZs) to use.
Choose Availability Zones (AZs) [\[Link\]](#)

Cancel Skip to review Previous Next

Recheck all settings and submit.

Log retention: 7 days Rotate logs: Deactivated Log file level: minor

X-Ray enabled: Deactivated

Environment properties

Key	Value
No environment properties	

There are no environment properties defined

Cancel Previous Submit

Environment is fully launched.

The screenshot shows the AWS Elastic Beanstalk console. A green banner at the top right says "Environment successfully launched." The main area displays the "For-devops-project-env" environment overview. It shows a "Health" status of "Green", an "Environment ID" of "e-vexetb9rbi", and a "Domain" of "For-devops-project-env.eba-udppbeh7.us-east-1.elasticbeanstalk.com". The "Platform" section indicates "PHP 8.1 running on 64bit Amazon Linux 2/3.5.7". Below the overview, there are tabs for Events, Health, Logs, Monitoring, Alarms, Managed updates, and Tags. The "Events" tab shows 12 events. On the left sidebar, under the "Environment: for-devops-project" section, there is a "Configuration" group containing "Events", "Health", "Logs", "Monitoring", and "Alarms".

Now Goto pipeline.

After codedeploy stage click on add stage.

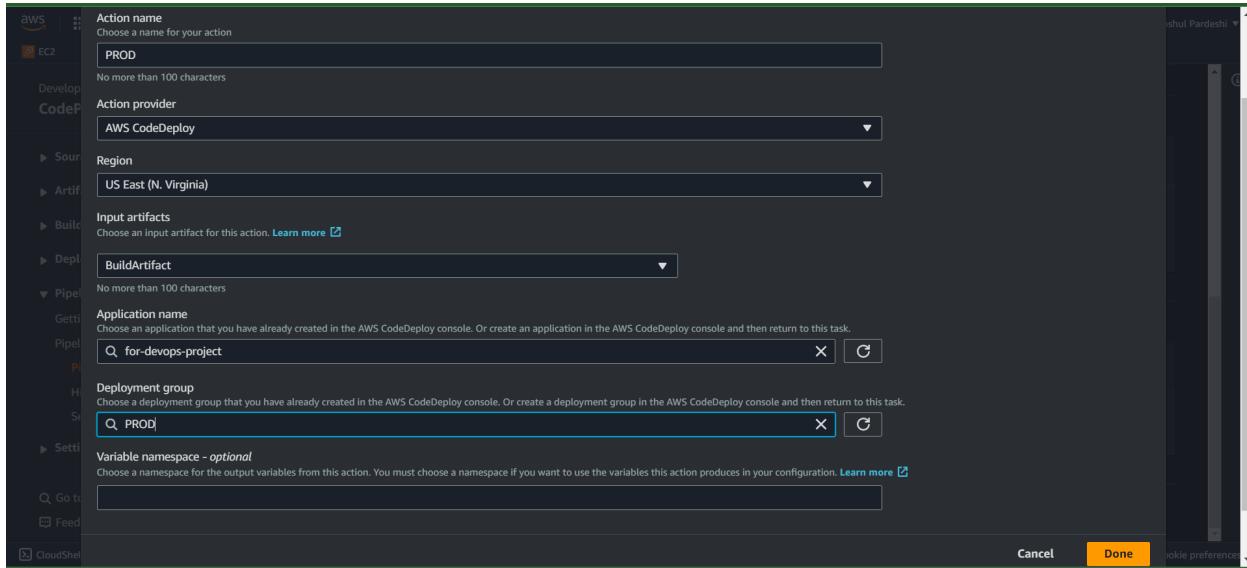
Give it a name.

The screenshot shows the AWS CodePipeline console. On the left sidebar, under the "Pipelines" section, there is a "Pipeline" group containing "Getting started", "Pipelines", "Pipeline", and "History". In the center, there is a modal window titled "Add stage" with the sub-header "Build AWS CodeBuild". The "Stage name" field contains "AWS CodeDeploy PROD". At the bottom right of the modal are "Cancel" and "Add stage" buttons. The background shows a pipeline named "Edit: Deploy" with several stages.

Add prod codedeploy.

The screenshot shows the AWS CodeDeploy configuration screen. On the left sidebar, under the "CodeDeploy" section, there is a "Deployment groups" group containing "Getting started", "Pipelines", "Deployment groups", and "CloudWatch Metrics". In the center, there is a form for creating a new deployment group. The "Action name" field is set to "PROD", the "Action provider" is "AWS CodeDeploy", the "Region" is "US East (N. Virginia)", and the "Input artifact" is "BuildArtifact". The "Application name" field contains "for-devops-project", and the "Deployment group" field also contains "PROD". The "Variable namespace - optional" field is empty. At the bottom right are "Cancel" and "Done" buttons.

Now add AWS Beanstalk ENV
Choose appropriate beanstalk env.



Action name
Choose a name for your action
PROD
No more than 100 characters

Action provider
AWS CodeDeploy

Region
US East (N. Virginia)

Input artifacts
Choose an input artifact for this action. [Learn more](#)
BuildArtifact
No more than 100 characters

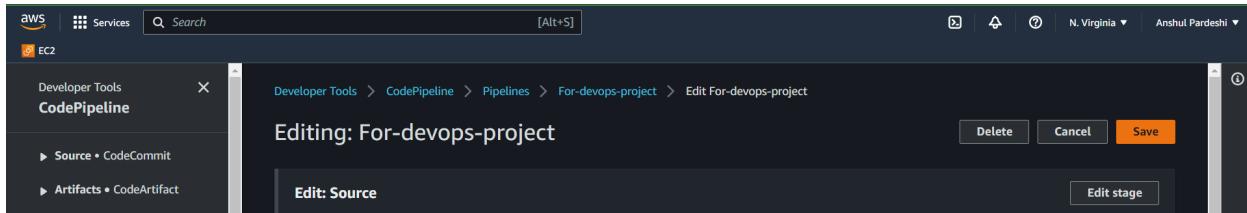
Application name
Choose an application that you have already created in the AWS CodeDeploy console. Or create an application in the AWS CodeDeploy console and then return to this task.
for-devops-project

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.
PROD

Variable namespace - optional
Choose a namespace for the output variables from this action. You must choose a namespace if you want to use the variables this action produces in your configuration. [Learn more](#)

Cancel Done

Click on save.



Developer Tools > CodePipeline > Pipelines > For-devops-project > Edit For-devops-project

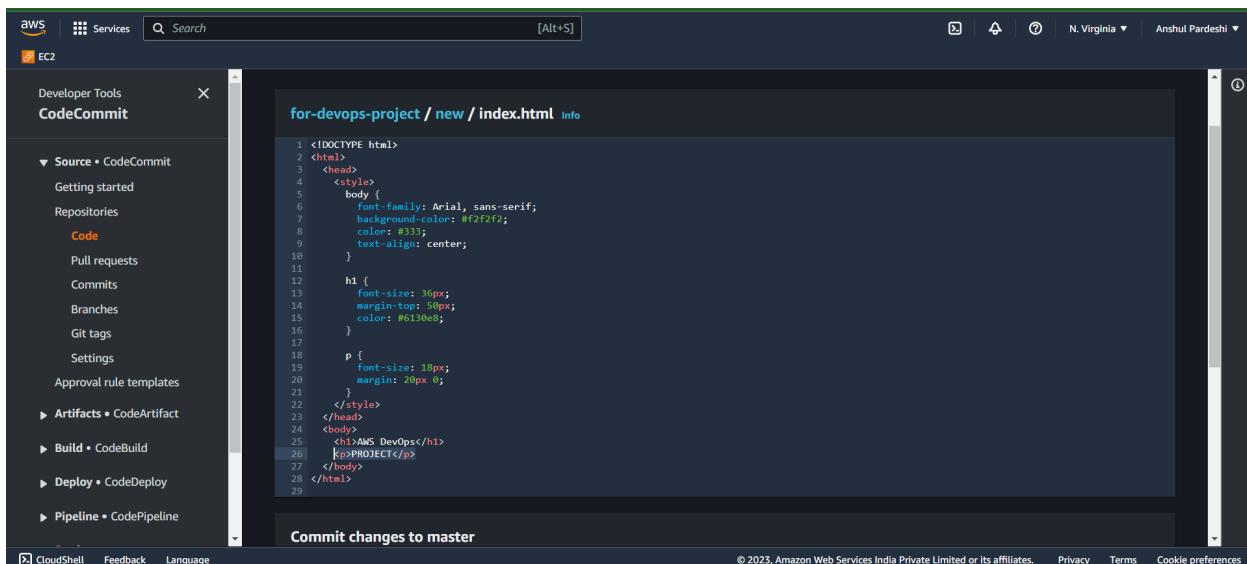
Editing: For-devops-project

Edit: Source

Delete Cancel Save

Let us make change in CodeCommit. Add extra line to it.

Before change:



```
<!DOCTYPE html>
<html>
<head>
<style>
body {
    font-family: Arial, sans-serif;
    background-color: #f2f2f2;
    color: #333;
    text-align: center;
}

h1 {
    font-size: 30px;
    margin-top: 50px;
    color: #6130e8;
}

p {
    font-size: 18px;
    margin: 20px 0;
}
</style>
</head>
<body>
<h1>AWS DevOps</h1>
<p>PROJECT</p>
</body>
</html>
```

Commit changes to master

After change:

The screenshot shows the AWS CodeCommit interface. On the left, the navigation sidebar is visible with options like Source, Artifacts, Build, Deploy, and Pipeline. The main area displays the content of the index.html file under the repository 'for-devops-project'. The code includes CSS styling for the body and h1 elements, and a message in the body stating 'All the triggers are working fine'.

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
    font-family: Arial, sans-serif;
    background-color: #f2f2f2;
    color: #333;
    text-align: center;
}
h1 {
    font-size: 36px;
    margin-top: 50px;
    color: #6130e8;
}
p {
    font-size: 18px;
    margin: 20px 0;
}
</style>
</head>
<body>
<h1>AWS DevOps</h1>
<p>PROJECT</p>
<p>All the triggers are working fine</p>
</body>
</html>
```

Commit it and lets see if it gets reflected in QA, PROD and Beanstalk ENV instance through pipeline.

Commit done.

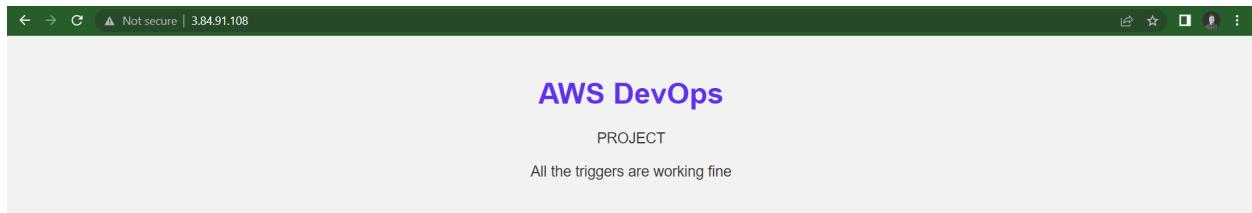
The screenshot shows the AWS CodeCommit interface again, but this time it includes a green banner at the top indicating 'new/index.html has been committed to master'. Below the banner, the commit details are shown, including the commit ID 'd0ac31d1'. The main area displays the same index.html file content as before.

Pipeline is successful.

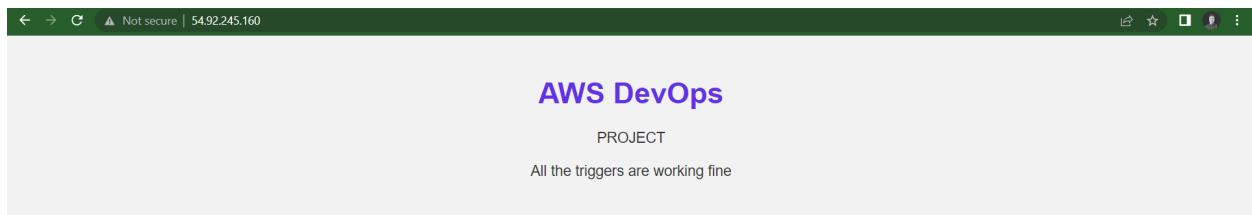
The screenshot shows the AWS CodePipeline interface. The left sidebar lists various stages: Source, Artifacts, Build, Deploy, and Pipeline. The Pipeline stage is currently selected. The main area displays the execution history of a pipeline named 'd0ac31d1'. It shows two stages: 'PROD' (AWS CodeDeploy) and 'Beanstalk' (AWS Elastic Beanstalk). Both stages are listed as 'Succeeded'. A vertical green bar on the right indicates the flow of the pipeline from Source to Deployment.

Let us check all ip's of all three instances.

QA



PROD



AWS Elastic Beanstalk

