AWS CODE DEPLOY AND CODE PIPELINE

INTRODUCTION:

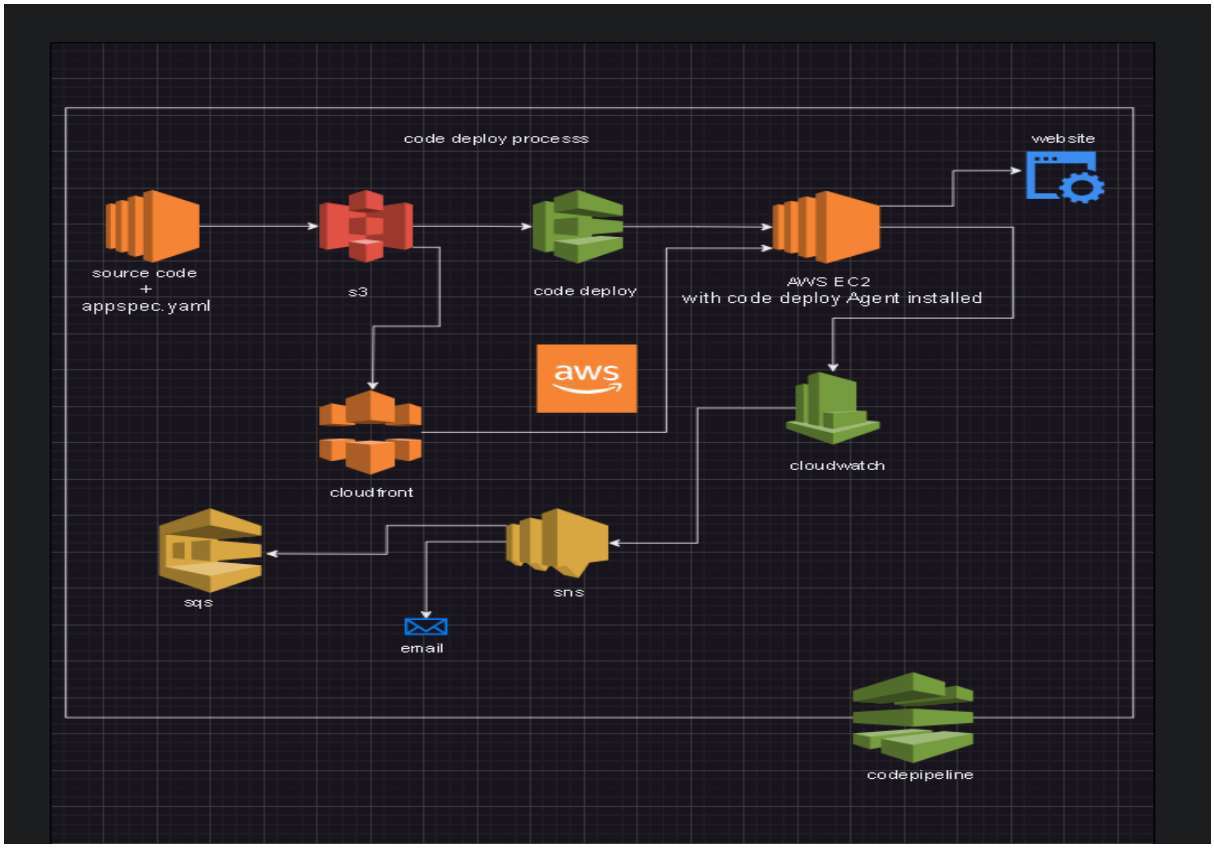
CODEDEPLOY:

AWS CodeDeploy is a service that automates application deployments to various compute services like EC2, Lambda, and on-premises servers, ensuring reliable updates and minimizing downtime.

CODEPIPLINE:

A CodePipeline automates the release process by integrating code from repositories, testing it, and deploying it to production environments, ensuring a streamlined CI/CD workflow. It enables continuous delivery of applications using AWS services.

MY PROJECT CONCEPT OUTPUT:

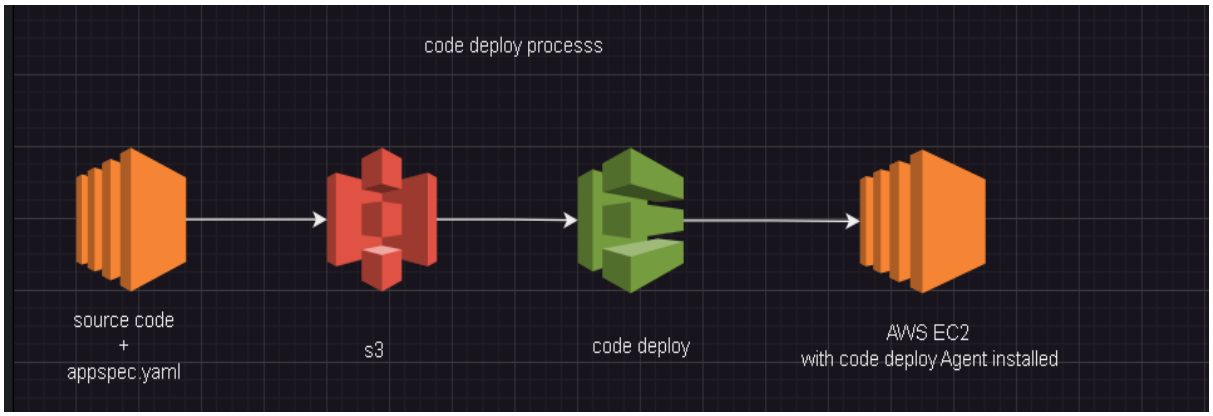


OBJECTIVE:

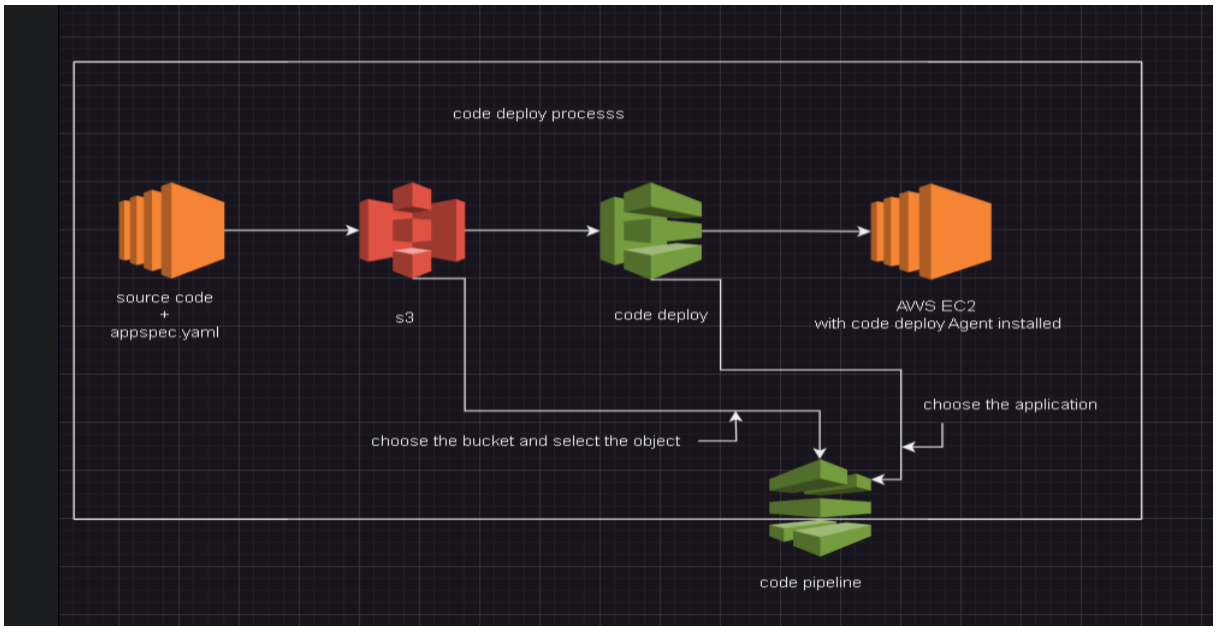
1: Streamline Continuous Deployment
"To automate and streamline the continuous deployment process using AWS CodePipeline and CodeDeploy, ensuring rapid and reliable delivery of application updates with minimal manual intervention."

2: Enhance Deployment Efficiency
"To implement an efficient CI/CD pipeline using AWS CodePipeline and CodeDeploy, aiming to reduce deployment time, minimize errors, and ensure seamless integration and delivery of new features across multiple environments."

Code deploy:



Code pipeline:



PROJECT IMPLEMENTATION DETAILS:

CODE DEPLOY AND CODE PIPELINE PROJECT STEPS:

Setup in Brief:

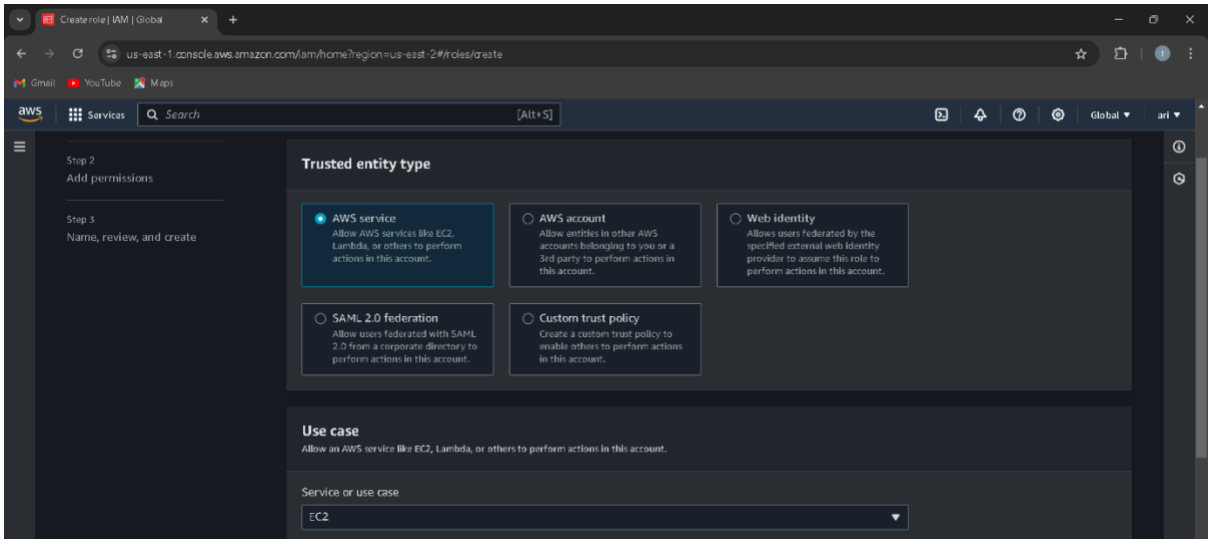
I have set up two EC2 instances of Amazon Linux 2. The first instance will serve as the web server running the CodeDeploy agent, which we will be configuring. The second EC2 instance is intended for developers to write and test code. Feel free to name the resources as you see fit for your own experiment

Project Step-By-Step Concept

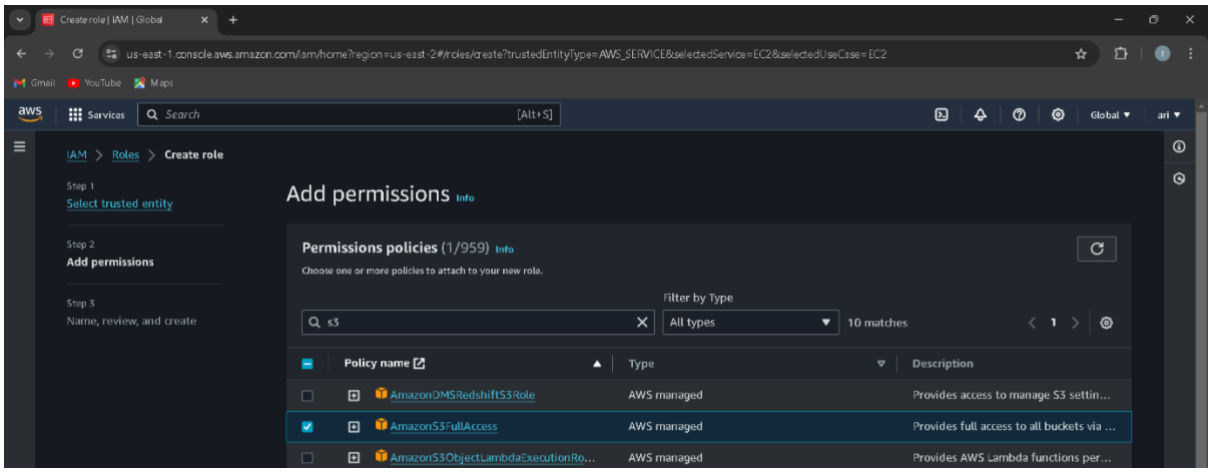
1) **Create IAM Roles for EC2-S3-CodeDeploy access.**
2) **Create an IAM user account for the developer.**
3) **Create 2 EC2 instances – Developer Machine & Production Machine.**
4) **Connect IAM user in Developer Machine CLI.**
5) **Install the ruby in the production machine and next install the code-deploy agent.**
6) **Create a directory, yaml scripts, vi intdex.html, and script files in the developer machine.**
7) **Create an S3 Bucket to store website files.**
8) **Copy the code deploy the application and paste it onto the developer's machine.**
9) **The next one is to copy the s3 command and paste it onto the developer's machine.**
10) **The next step is to check the s3 bucket and code deploy.**
11) **Create a deployment group and the next one create a code deployment.**
12) **Successfully create the code deploy and copy the production IP address paste the IP address into your browser and visible your output**
13) **Next step is to create a code pipeline and run your output**
14) **Add additional services**
15) **Create (SQS & SNS) for Communication services.**
16) **Create a (cloud watch) for an alarm.**
17) **Create a (Cloud Front).**
18) **Create my own (portfolio website) html css.**
19) **Browse the website.**

Step 1: Create IAM Roles for EC2-S3-access:

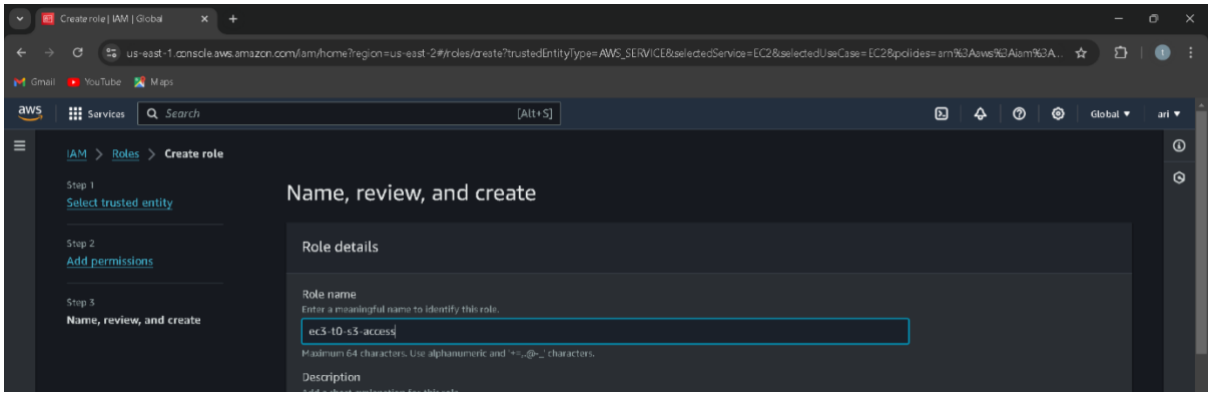
1) **Select the ec2 in the use case**



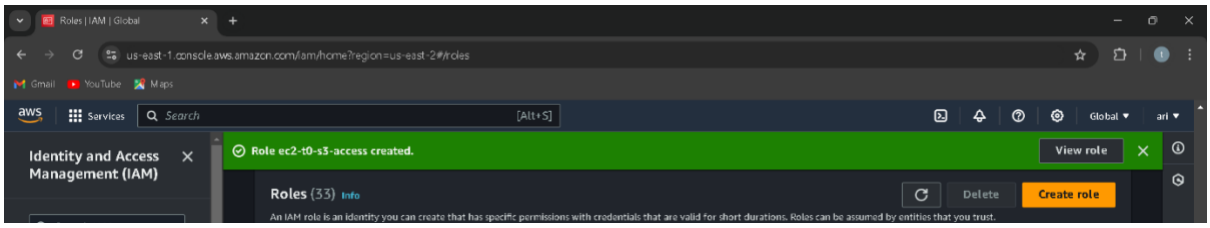
2) **Select the S3 permissions.**



3) **Create a role name.**

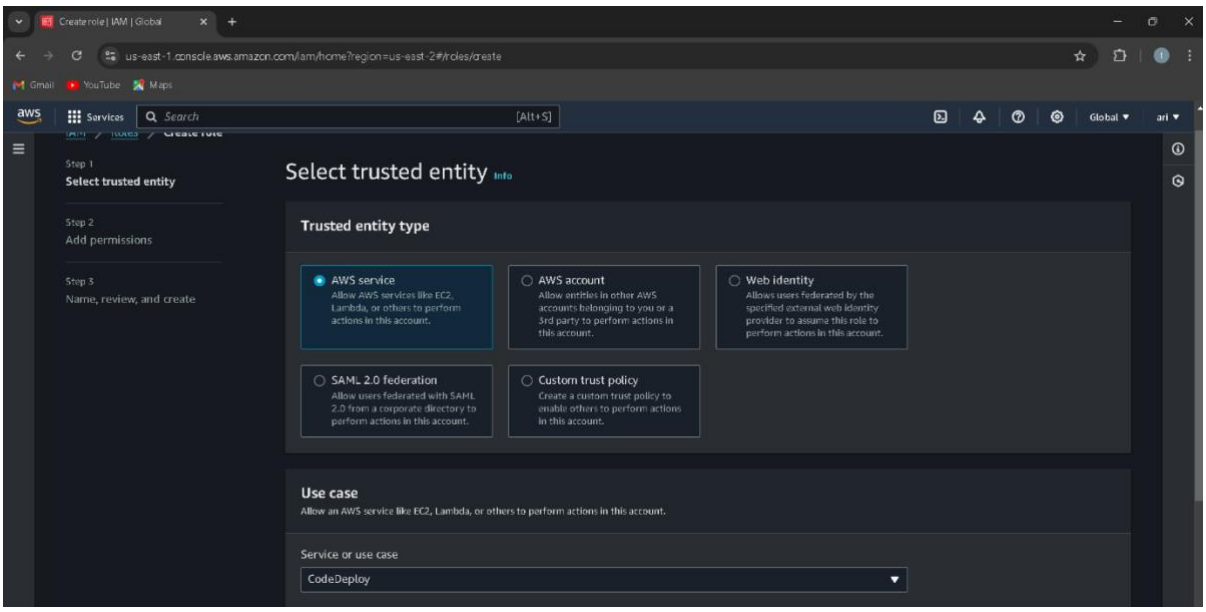


4) **Successfully create the ec2-s3-access**

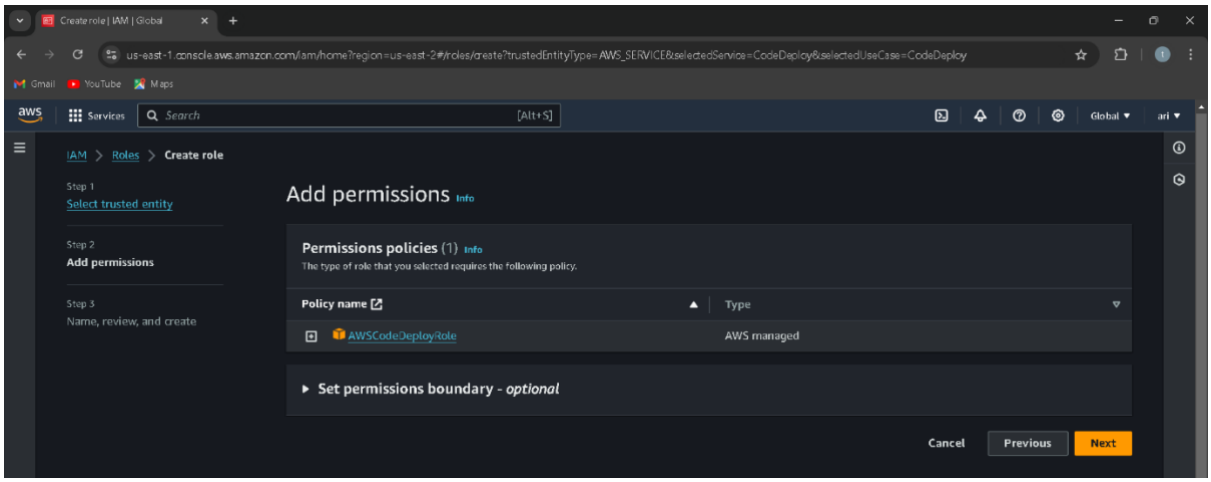


Step 2: Create a code-deploy access:

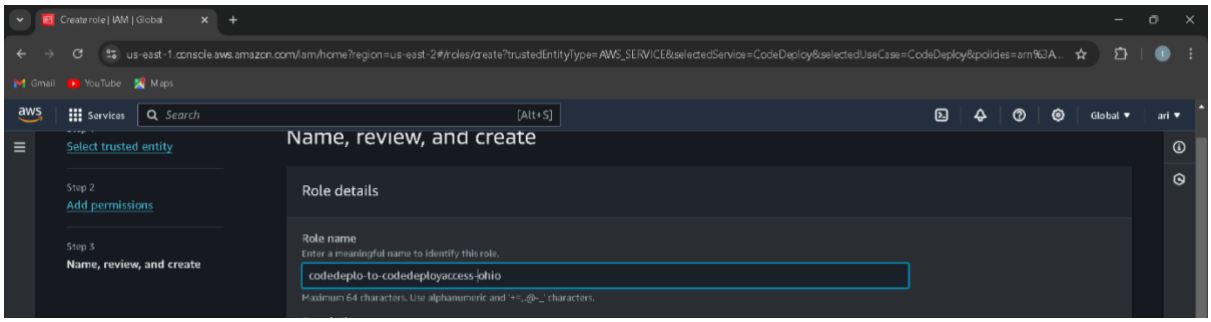
1) **Select the use case in code-deploy.**



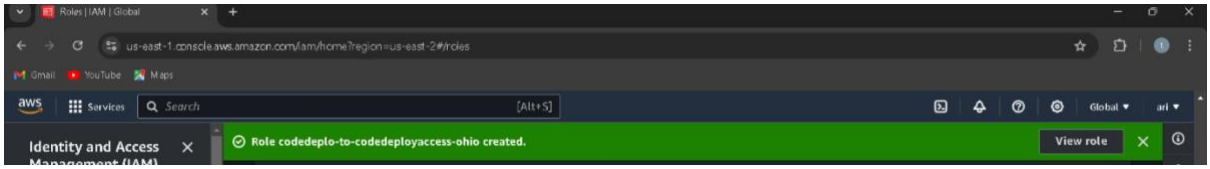
2) **Add the code-deploy permission**



3) **Create a code-deploy access name.**

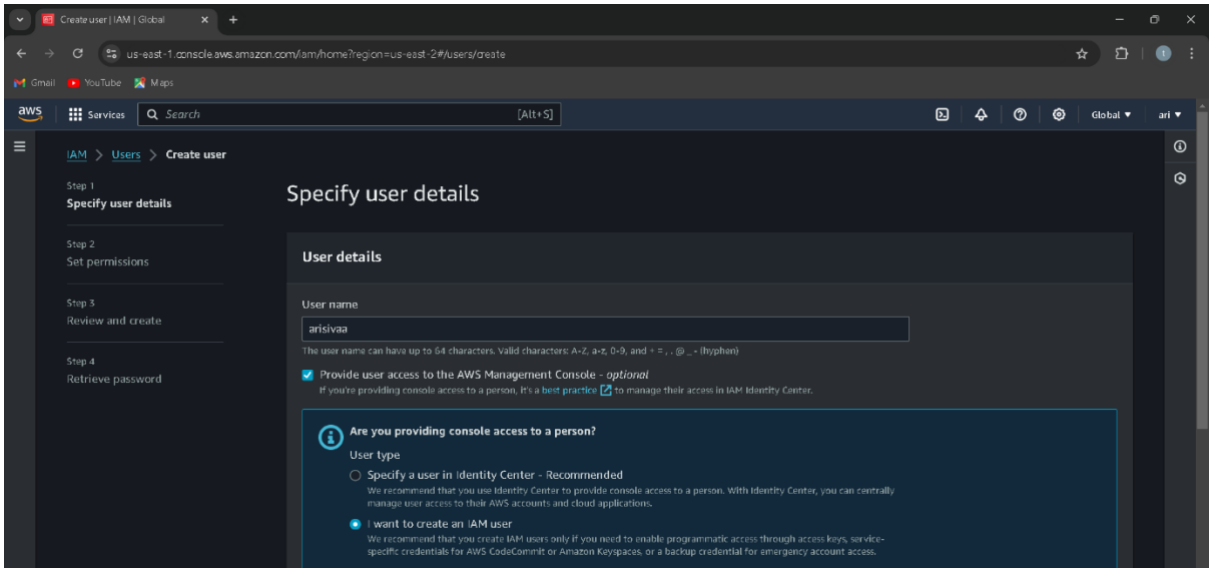


4) **Successfully create a code-deploy access.**

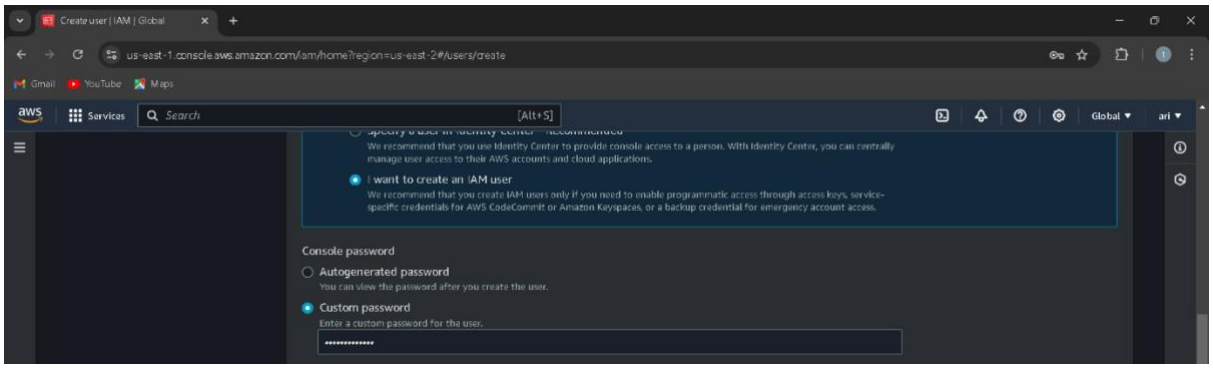


Step 3: Create an IAM user:

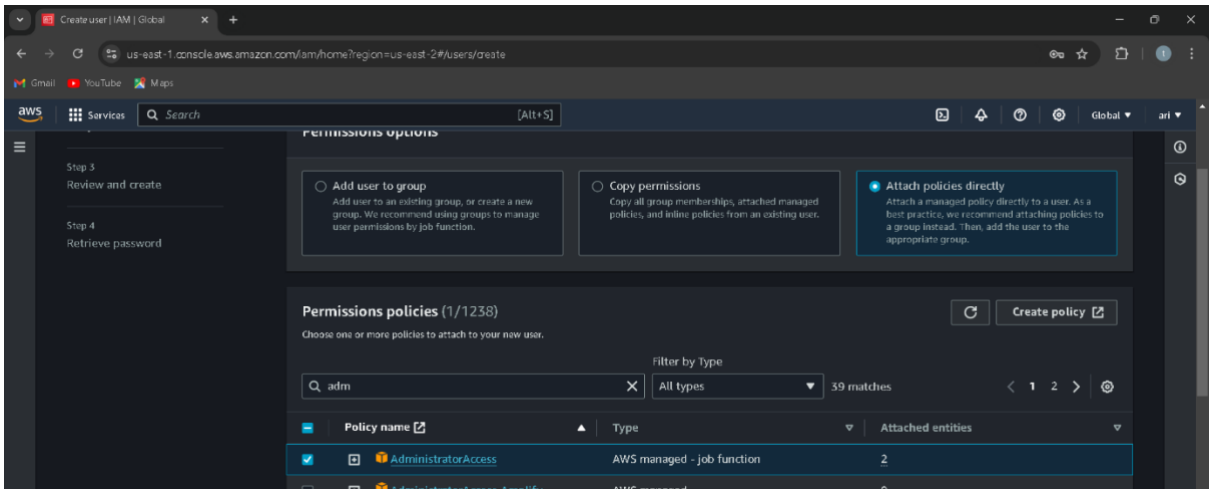
1) **Create a user name.**



2) Next create a password.

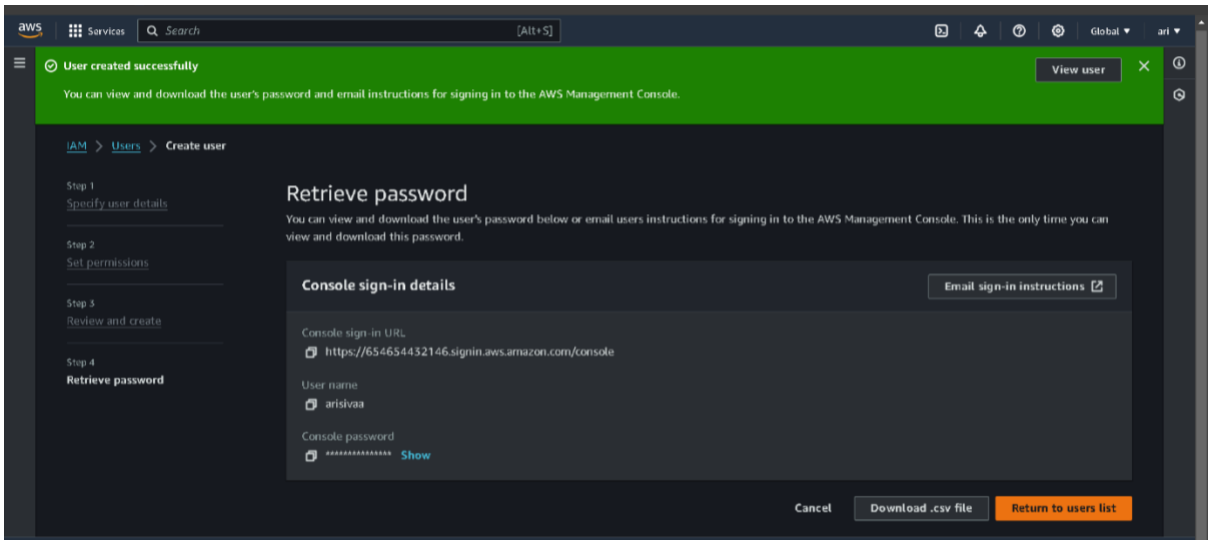


3) Attached are the policies directly.



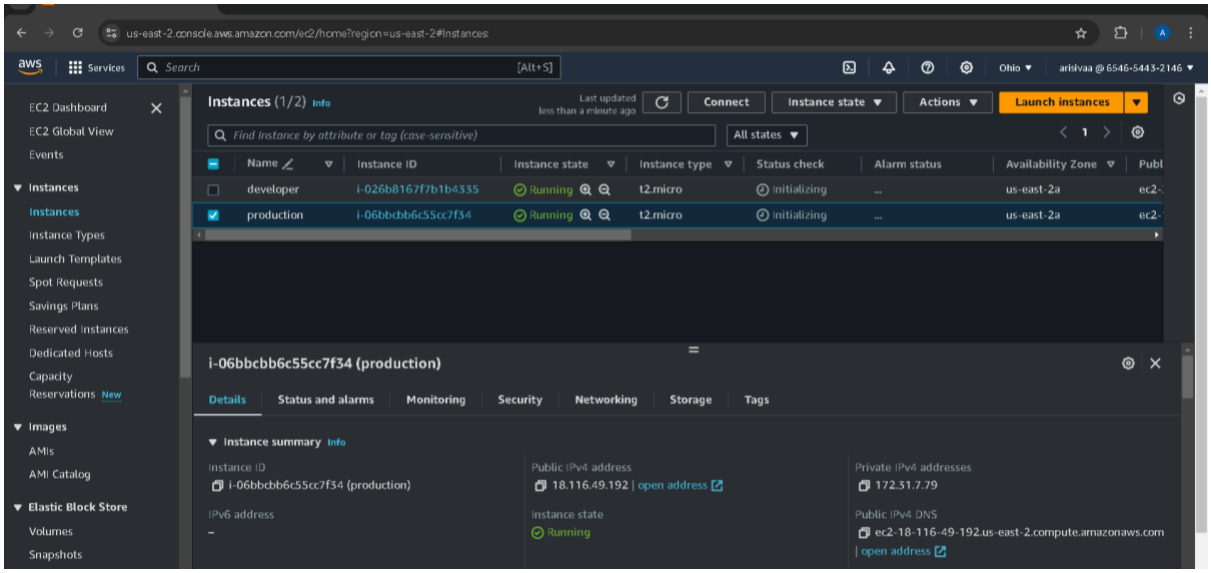
4) Successfully create an IAM user name

IAM user name : arisivaa



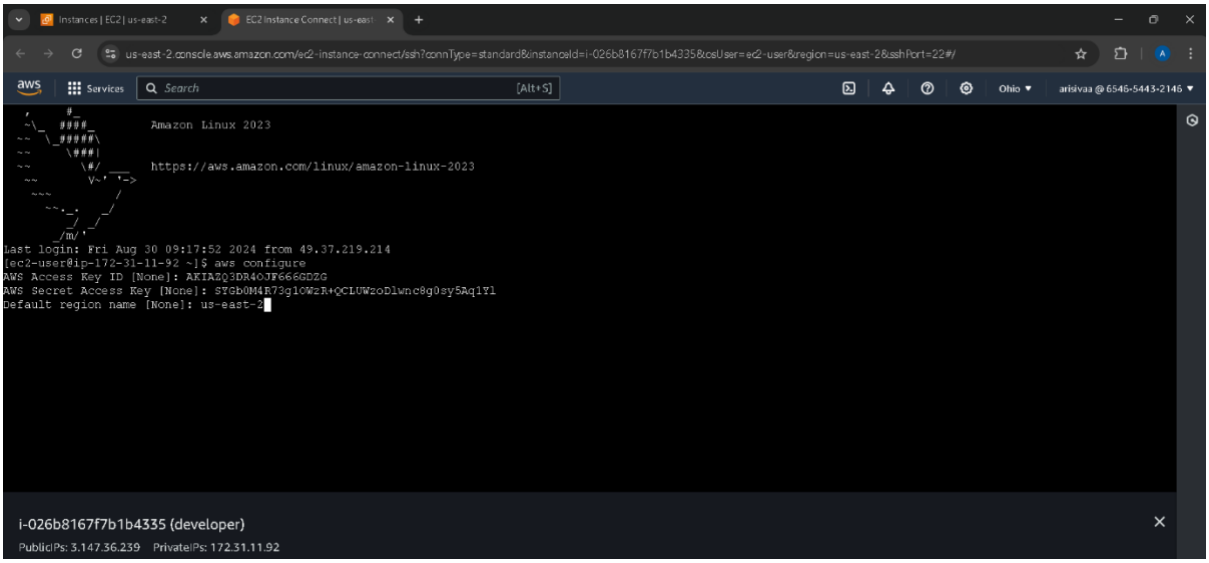
Step4:Create a two ec2 instances (developer&production) machine:

1) **Create an Ec2 instance and the same configuration**



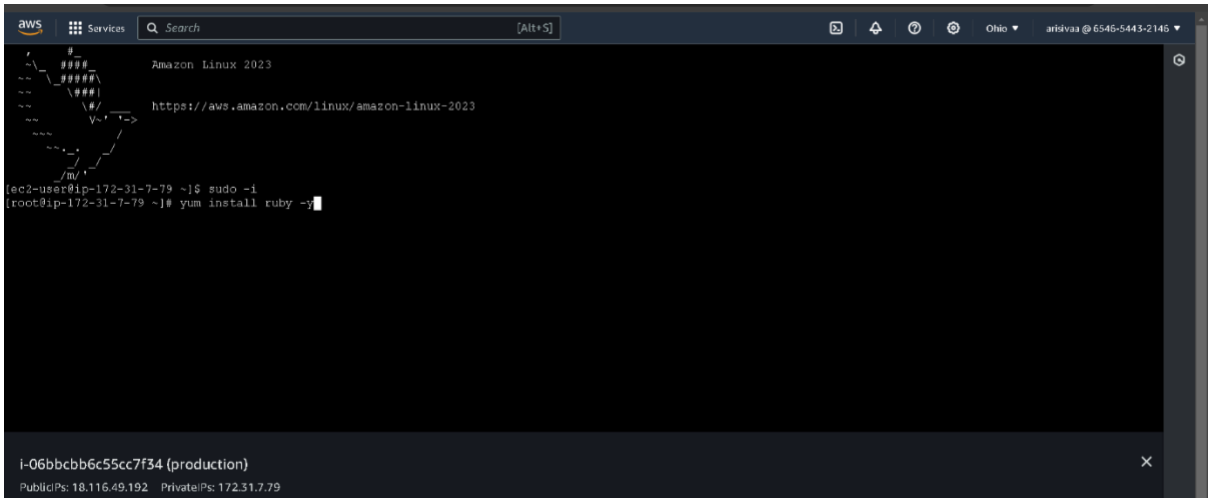
2) **The next one is to connect IAM users in the Developer machine <u>CLI</u>.**

Aws configure
Acess key:
Secret key:
Region:
Default ouput:

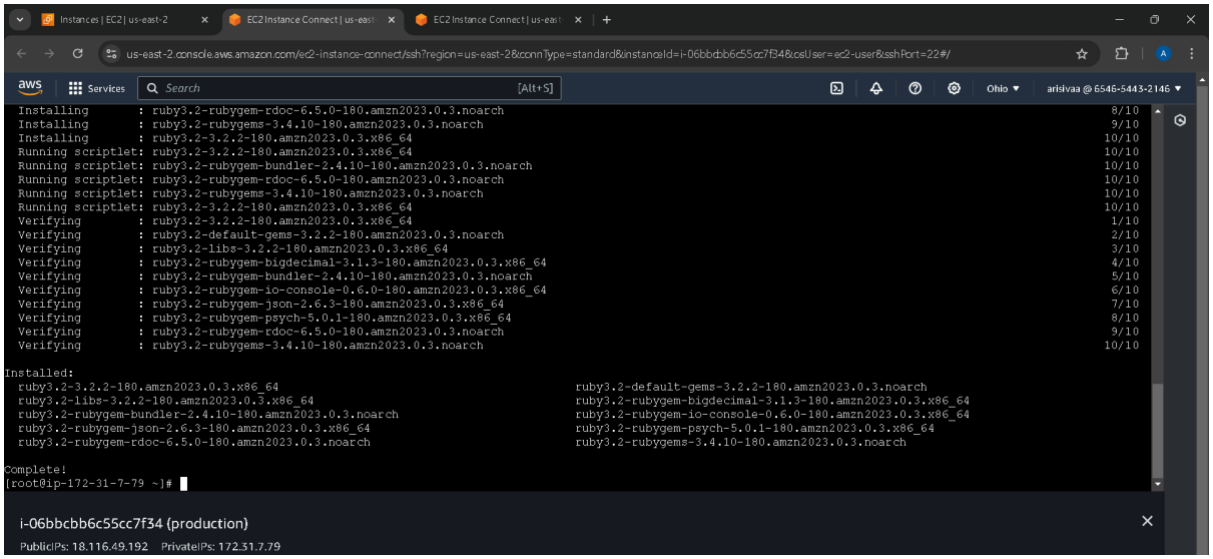


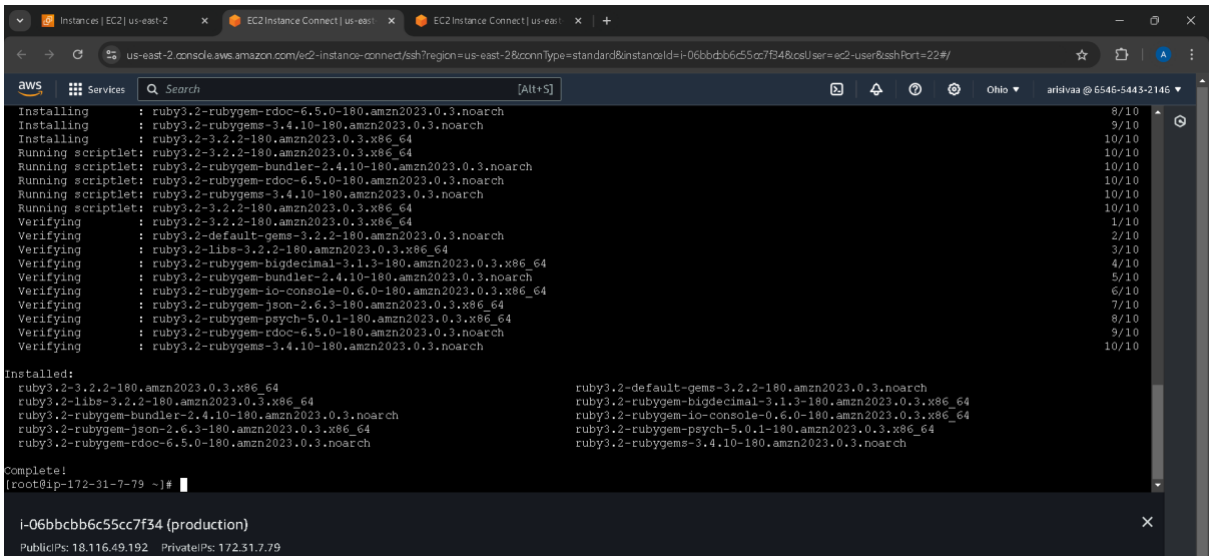
3) Install the ruby in the production machine and next install the code-deploy age

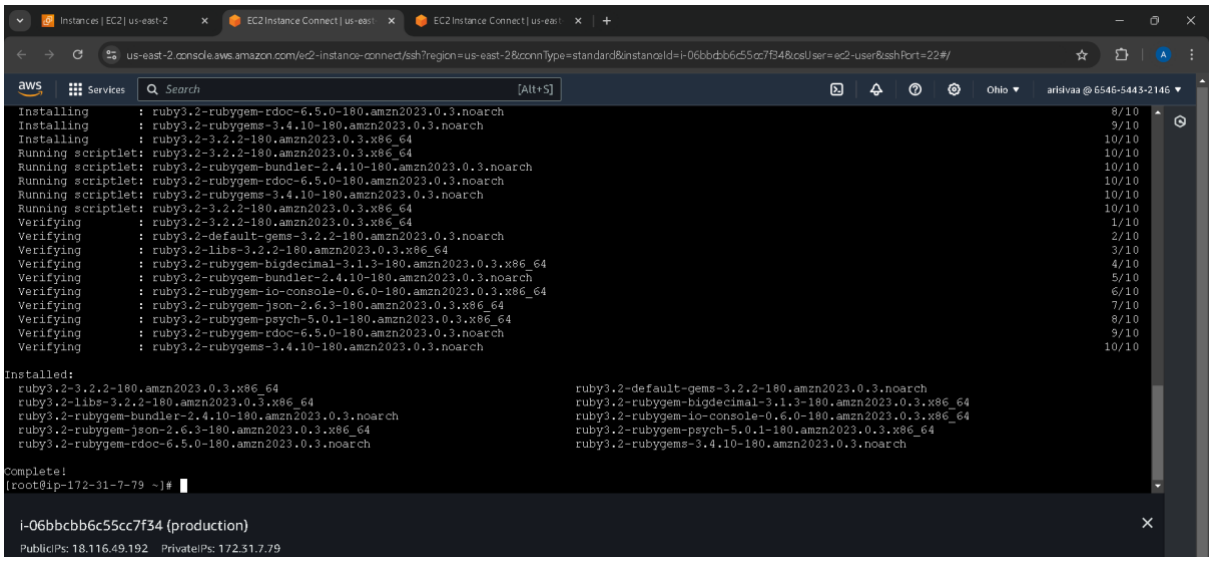
Ruby install command: **Yum install ruby -y**

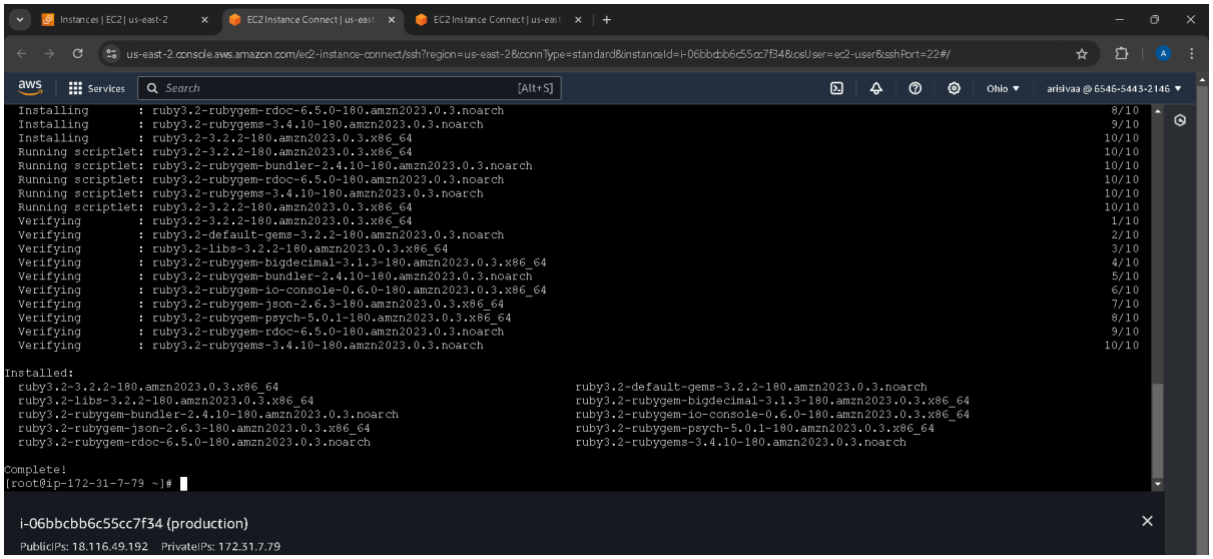


Ruby installation complete.


Installing : ruby3.2-rubygem-rdoc-6.5.0-180.amzn2023.0.3.noarch 8/10
Installing : ruby3.2-rubygems-3.4.10-180.amzn2023.0.3.noarch 9/10
Installing : ruby3.2-3.2.2-180.amzn2023.0.3.x86_64 10/10
Running scriptlet: ruby3.2-3.2.2-180.amzn2023.0.3.x86_64 10/10
Running scriptlet: ruby3.2-rubygem-bundler-2.4.10-180.amzn2023.0.3.noarch 10/10
Running scriptlet: ruby3.2-rubygem-rdoc-6.5.0-180.amzn2023.0.3.noarch 10/10
Running scriptlet: ruby3.2-rubygems-3.4.10-180.amzn2023.0.3.noarch 10/10
Running scriptlet: ruby3.2-3.2.2-180.amzn2023.0.3.x86_64 10/10
Verifying : ruby3.2-3.2.2-180.amzn2023.0.3.x86_64 1/10
Verifying : ruby3.2-default-gems-3.2.2-180.amzn2023.0.3.noarch 2/10
Verifying : ruby3.2-libs-3.2.2-180.amzn2023.0.3.x86_64 3/10
Verifying : ruby3.2-rubygem-bigdecimal-3.1.3-180.amzn2023.0.3.x86_64 4/10
Verifying : ruby3.2-rubygem-bundler-2.4.10-180.amzn2023.0.3.noarch 5/10
Verifying : ruby3.2-rubygem-io-console-0.6.0-180.amzn2023.0.3.x86_64 6/10
Verifying : ruby3.2-rubygem-json-2.6.3-180.amzn2023.0.3.x86_64 7/10
Verifying : ruby3.2-rubygem-psych-5.0.1-180.amzn2023.0.3.x86_64 8/10
Verifying : ruby3.2-rubygem-rdoc-6.5.0-180.amzn2023.0.3.noarch 9/10
Verifying : ruby3.2-rubygems-3.4.10-180.amzn2023.0.3.noarch 10/10
Installed:
ruby3.2-3.2.2-180.amzn2023.0.3.x86_64
ruby3.2-default-gems-3.2.2-180.amzn2023.0.3.noarch
ruby3.2-libs-3.2.2-180.amzn2023.0.3.x86_64
ruby3.2-rubygem-bigdecimal-3.1.3-180.amzn2023.0.3.x86_64
ruby3.2-rubygem-bundler-2.4.10-180.amzn2023.0.3.noarch
ruby3.2-rubygem-io-console-0.6.0-180.amzn2023.0.3.x86_64
ruby3.2-rubygem-json-2.6.3-180.amzn2023.0.3.x86_64
ruby3.2-rubygem-psych-5.0.1-180.amzn2023.0.3.x86_64
ruby3.2-rubygem-rdoc-6.5.0-180.amzn2023.0.3.noarch
ruby3.2-rubygems-3.4.10-180.amzn2023.0.3.noarch
Complete!
[root@ip-172-31-7-79 ~]#
i-06bbcbb6c55cc7f34 (production)
PublicIPs: 18.116.49.192 PrivateIPs: 172.31.7.79


Installing : ruby3.2-rubygem-rdoc-6.5.0-180.amzn2023.0.3.noarch 8/10
Installing : ruby3.2-rubygems-3.4.10-180.amzn2023.0.3.noarch 9/10
Installing : ruby3.2-3.2.2-180.amzn2023.0.3.x86_64 10/10
Running scriptlet: ruby3.2-3.2.2-180.amzn2023.0.3.x86_64 10/10
Running scriptlet: ruby3.2-rubygem-bundler-2.4.10-180.amzn2023.0.3.noarch 10/10
Running scriptlet: ruby3.2-rubygem-rdoc-6.5.0-180.amzn2023.0.3.noarch 10/10
Running scriptlet: ruby3.2-rubygems-3.4.10-180.amzn2023.0.3.noarch 10/10
Running scriptlet: ruby3.2-3.2.2-180.amzn2023.0.3.x86_64 10/10
Verifying : ruby3.2-3.2.2-180.amzn2023.0.3.x86_64 1/10
Verifying : ruby3.2-default-gems-3.2.2-180.amzn2023.0.3.noarch 2/10
Verifying : ruby3.2-libs-3.2.2-180.amzn2023.0.3.x86_64 3/10
Verifying : ruby3.2-rubygem-bigdecimal-3.1.3-180.amzn2023.0.3.x86_64 4/10
Verifying : ruby3.2-rubygem-bundler-2.4.10-180.amzn2023.0.3.noarch 5/10
Verifying : ruby3.2-rubygem-io-console-0.6.0-180.amzn2023.0.3.x86_64 6/10
Verifying : ruby3.2-rubygem-json-2.6.3-180.amzn2023.0.3.x86_64 7/10
Verifying : ruby3.2-rubygem-psych-5.0.1-180.amzn2023.0.3.x86_64 8/10
Verifying : ruby3.2-rubygem-rdoc-6.5.0-180.amzn2023.0.3.noarch 9/10
Verifying : ruby3.2-rubygems-3.4.10-180.amzn2023.0.3.noarch 10/10
Installed:
ruby3.2-3.2.2-180.amzn2023.0.3.x86_64
ruby3.2-default-gems-3.2.2-180.amzn2023.0.3.noarch
ruby3.2-libs-3.2.2-180.amzn2023.0.3.x86_64
ruby3.2-rubygem-bigdecimal-3.1.3-180.amzn2023.0.3.x86_64
ruby3.2-rubygem-bundler-2.4.10-180.amzn2023.0.3.noarch
ruby3.2-rubygem-io-console-0.6.0-180.amzn2023.0.3.x86_64
ruby3.2-rubygem-json-2.6.3-180.amzn2023.0.3.x86_64
ruby3.2-rubygem-psych-5.0.1-180.amzn2023.0.3.x86_64
ruby3.2-rubygem-rdoc-6.5.0-180.amzn2023.0.3.noarch
ruby3.2-rubygems-3.4.10-180.amzn2023.0.3.noarch
Complete!
[root@ip-172-31-7-79 ~]#
i-06bbcbb6c55cc7f34 (production)
PublicIPs: 18.116.49.192 PrivateIPs: 172.31.7.79

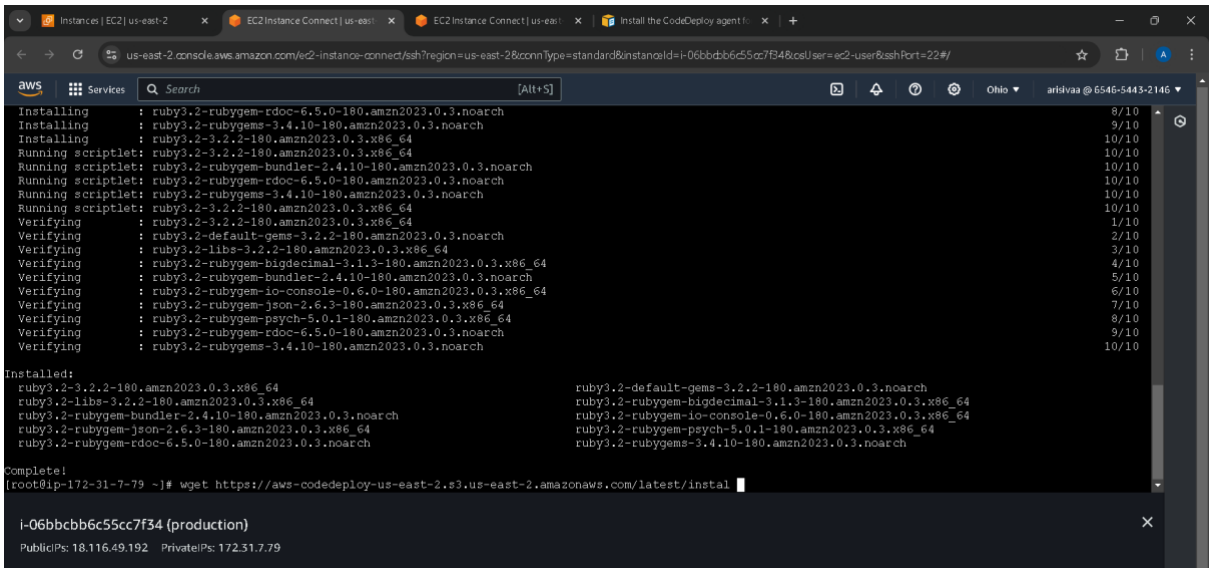

Installing : ruby3.2-rubygem-rdoc-6.5.0-180.amzn2023.0.3.noarch 8/10
Installing : ruby3.2-rubygems-3.4.10-180.amzn2023.0.3.noarch 9/10
Installing : ruby3.2-3.2.2-180.amzn2023.0.3.x86_64 10/10
Running scriptlet: ruby3.2-3.2.2-180.amzn2023.0.3.x86_64 10/10
Running scriptlet: ruby3.2-rubygem-bundler-2.4.10-180.amzn2023.0.3.noarch 10/10
Running scriptlet: ruby3.2-rubygem-rdoc-6.5.0-180.amzn2023.0.3.noarch 10/10
Running scriptlet: ruby3.2-rubygems-3.4.10-180.amzn2023.0.3.noarch 10/10
Running scriptlet: ruby3.2-3.2.2-180.amzn2023.0.3.x86_64 10/10
Verifying : ruby3.2-3.2.2-180.amzn2023.0.3.x86_64 1/10
Verifying : ruby3.2-default-gems-3.2.2-180.amzn2023.0.3.noarch 2/10
Verifying : ruby3.2-libs-3.2.2-180.amzn2023.0.3.x86_64 3/10
Verifying : ruby3.2-rubygem-bigdecimal-3.1.3-180.amzn2023.0.3.x86_64 4/10
Verifying : ruby3.2-rubygem-bundler-2.4.10-180.amzn2023.0.3.noarch 5/10
Verifying : ruby3.2-rubygem-io-console-0.6.0-180.amzn2023.0.3.x86_64 6/10
Verifying : ruby3.2-rubygem-json-2.6.3-180.amzn2023.0.3.x86_64 7/10
Verifying : ruby3.2-rubygem-psych-5.0.1-180.amzn2023.0.3.x86_64 8/10
Verifying : ruby3.2-rubygem-rdoc-6.5.0-180.amzn2023.0.3.noarch 9/10
Verifying : ruby3.2-rubygems-3.4.10-180.amzn2023.0.3.noarch 10/10
Installed:
ruby3.2-3.2.2-180.amzn2023.0.3.x86_64
ruby3.2-default-gems-3.2.2-180.amzn2023.0.3.noarch
ruby3.2-libs-3.2.2-180.amzn2023.0.3.x86_64
ruby3.2-rubygem-bigdecimal-3.1.3-180.amzn2023.0.3.x86_64
ruby3.2-rubygem-bundler-2.4.10-180.amzn2023.0.3.noarch
ruby3.2-rubygem-io-console-0.6.0-180.amzn2023.0.3.x86_64
ruby3.2-rubygem-json-2.6.3-180.amzn2023.0.3.x86_64
ruby3.2-rubygem-psych-5.0.1-180.amzn2023.0.3.x86_64
ruby3.2-rubygem-rdoc-6.5.0-180.amzn2023.0.3.noarch
ruby3.2-rubygems-3.4.10-180.amzn2023.0.3.noarch
Complete!
[root@ip-172-31-7-79 ~]#
i-06bbcbb6c55cc7f34 (production)
PublicIPs: 18.116.49.192 PrivateIPs: 172.31.7.79



4) Already wget a command package installed on my server

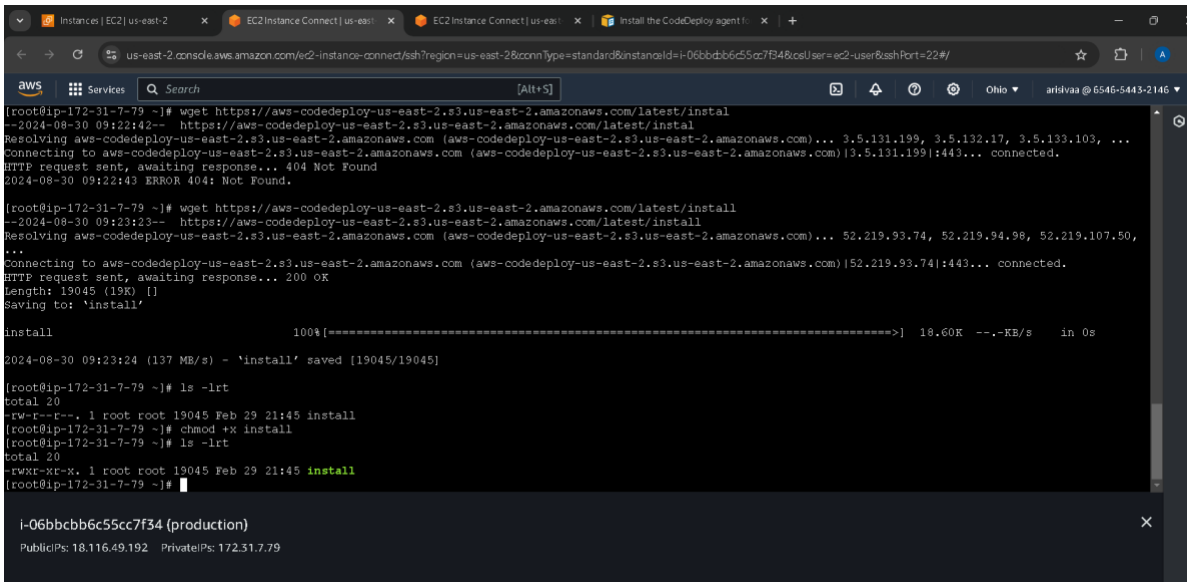
(COPY THE CODE AGENT COMMENT AND PASTE)

1) Copy the code deploy agent and paste it to your server.

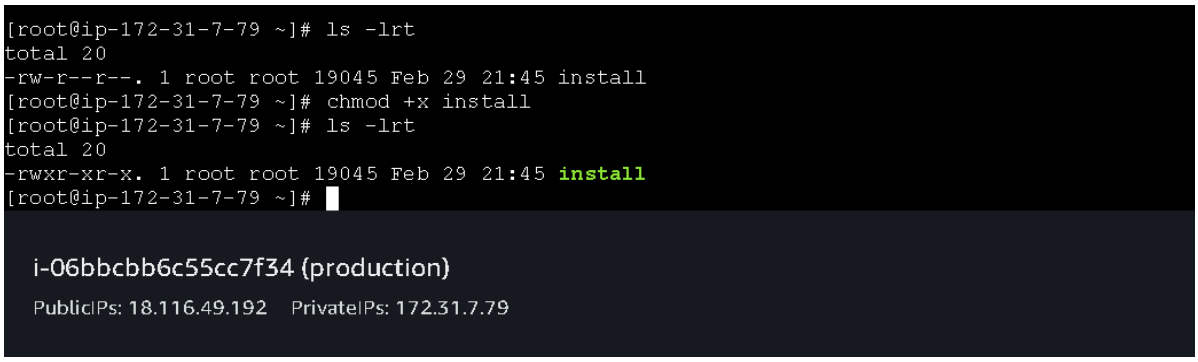


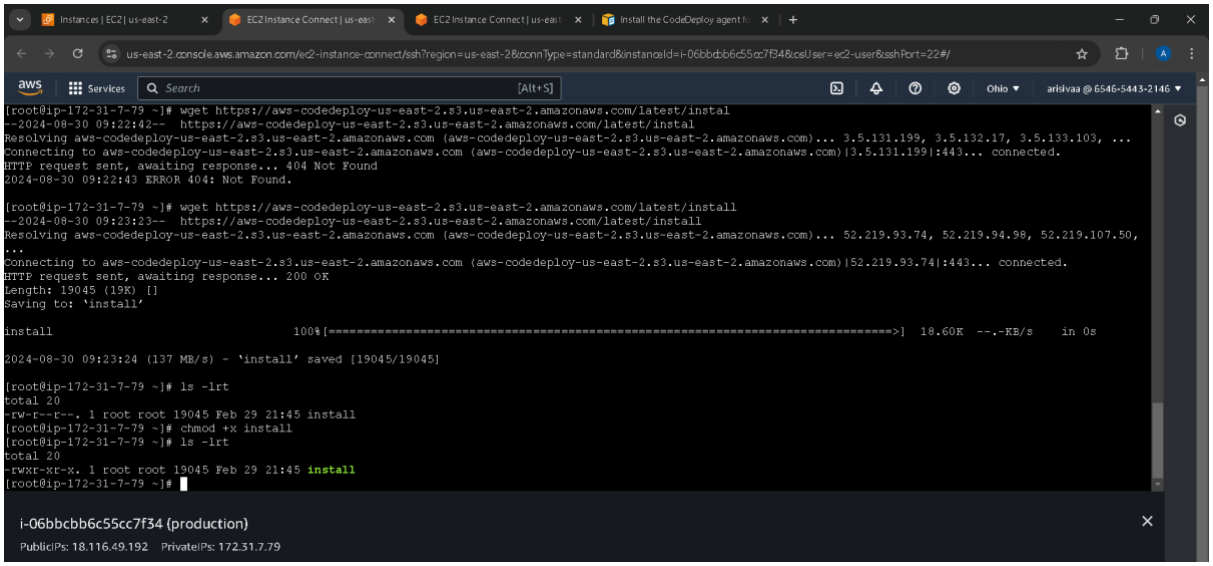
Wget aws deploy create-application --application-name sampleapp

2) Successfully created a code deploy-agent

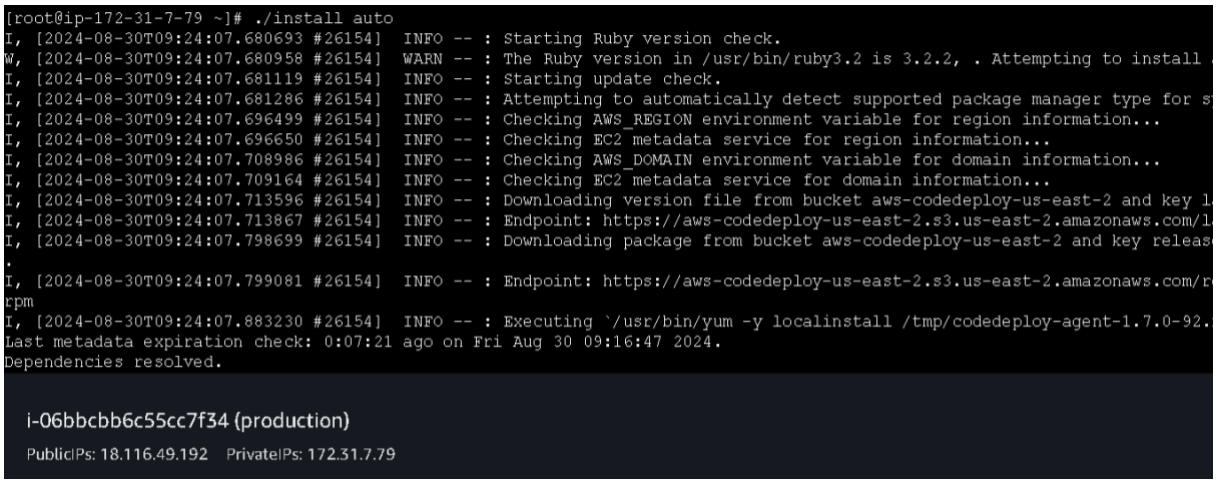


3) Change the file mode using this command (**chmod +x install**)

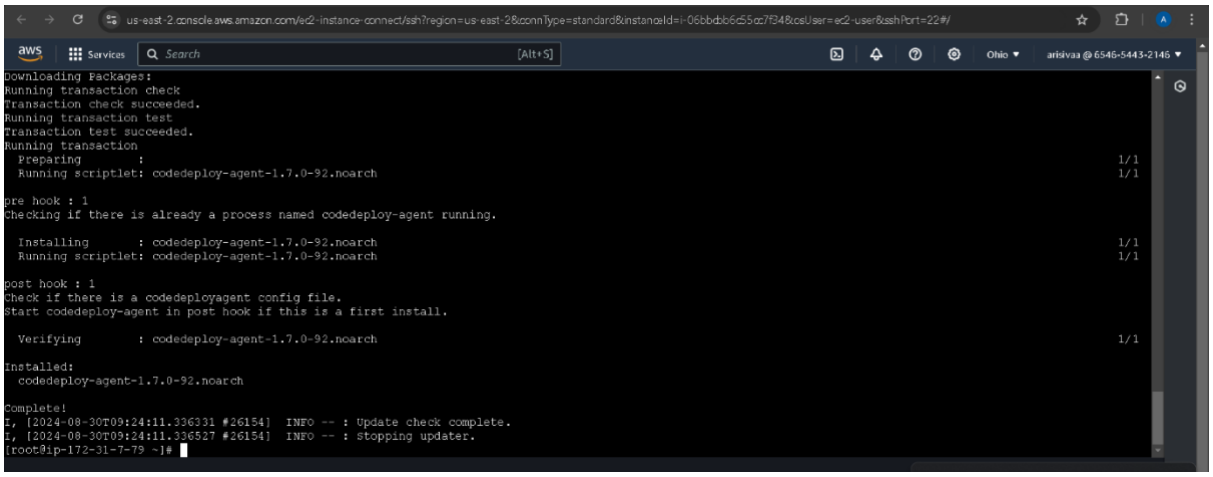




4) **The Next one is to type this command (./your file name)**

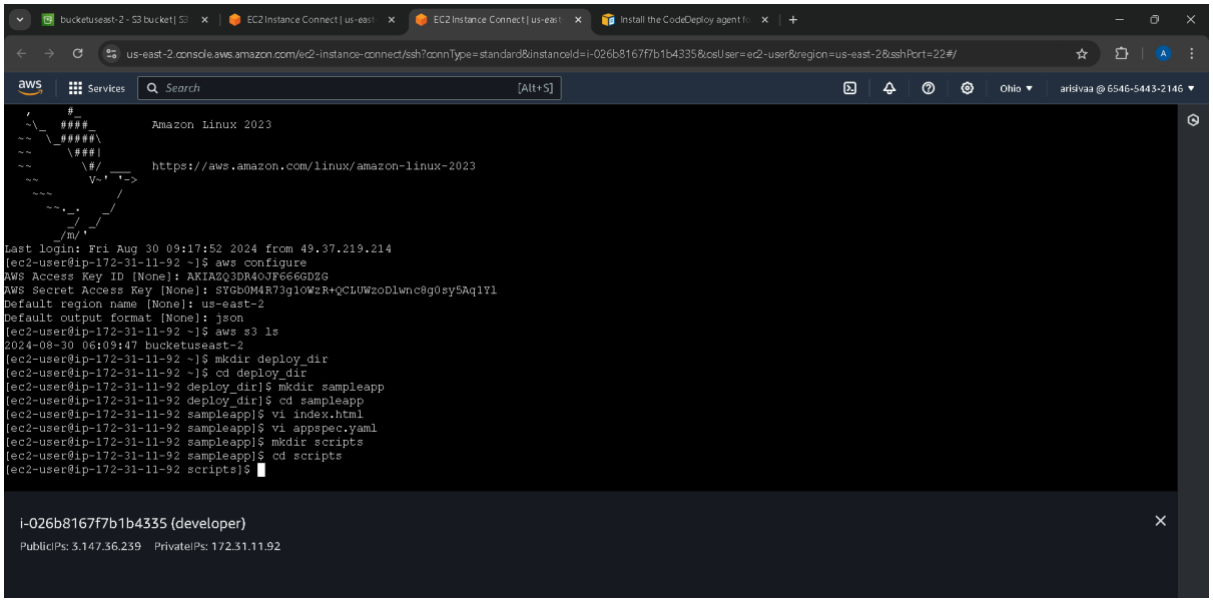


5) **Successfully create code-deploy-agent**



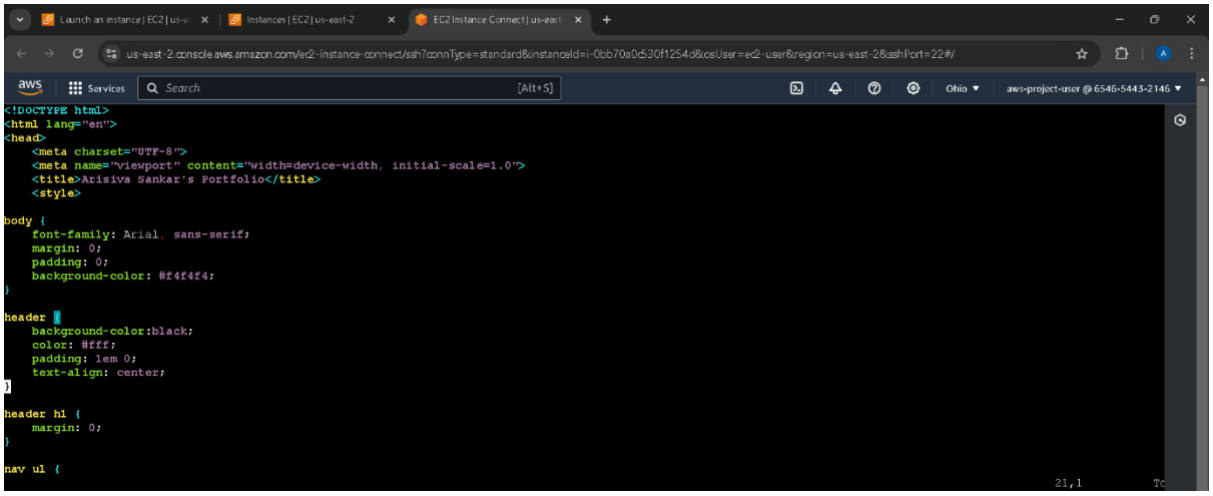
Step 5: Create the code from the developer machine - white machine. Work in root

1) **mkdir deploy_dir**
   cd deploy_dir
   mkdir sampleapp
   cd sampleapp
   ls -lrt



2) vi index.html

Enter the content below and save by typing :wq! and pressing Enter

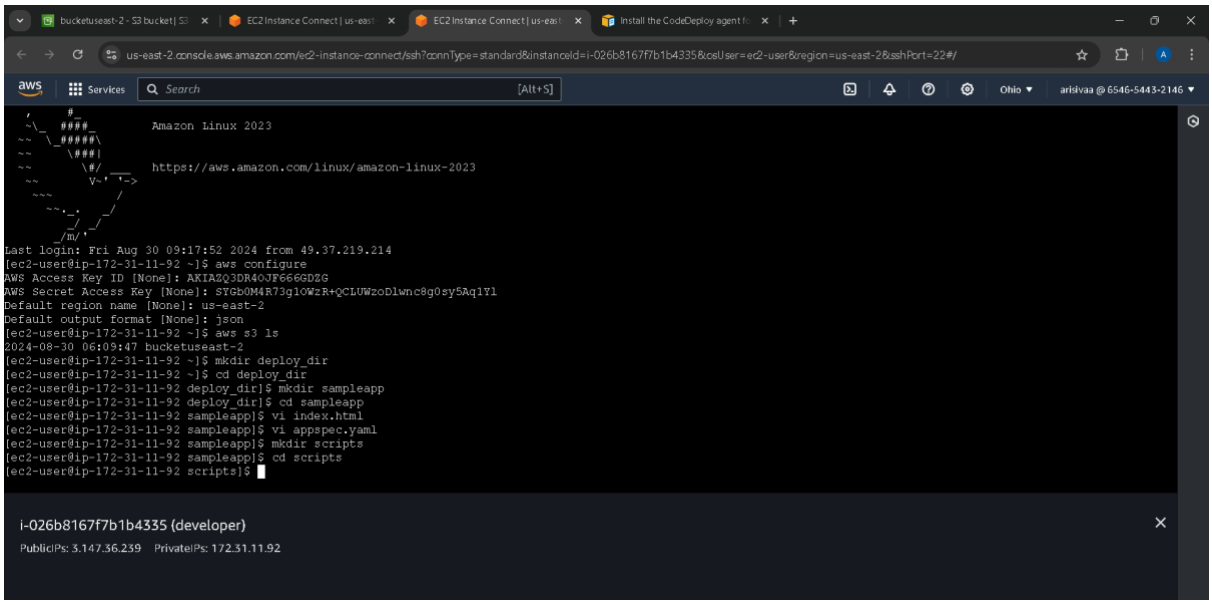


:wq! Then Enter

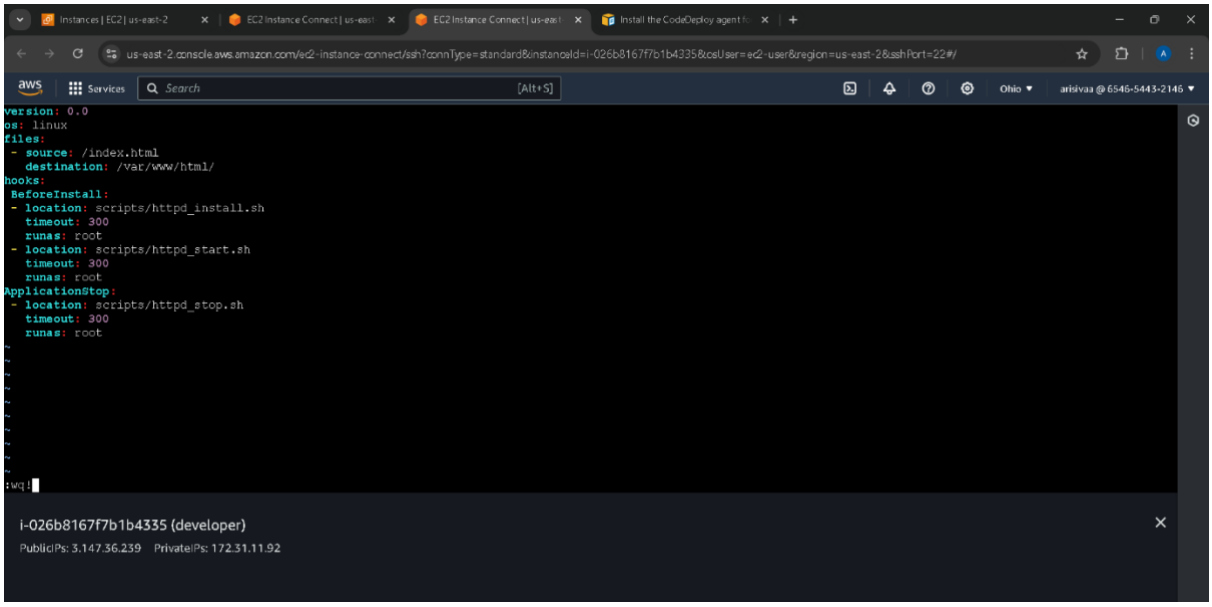
Step 6: Create appspec.yaml file: This file helps in automatically deploying the source code to The webserver.

1) **Vi appspec.yaml**

```
c2-user@ip-172-31-11-92 sampleapp]$ vi appspec.yaml
```



2) **The next one uploads the code in vi appspec.yaml**

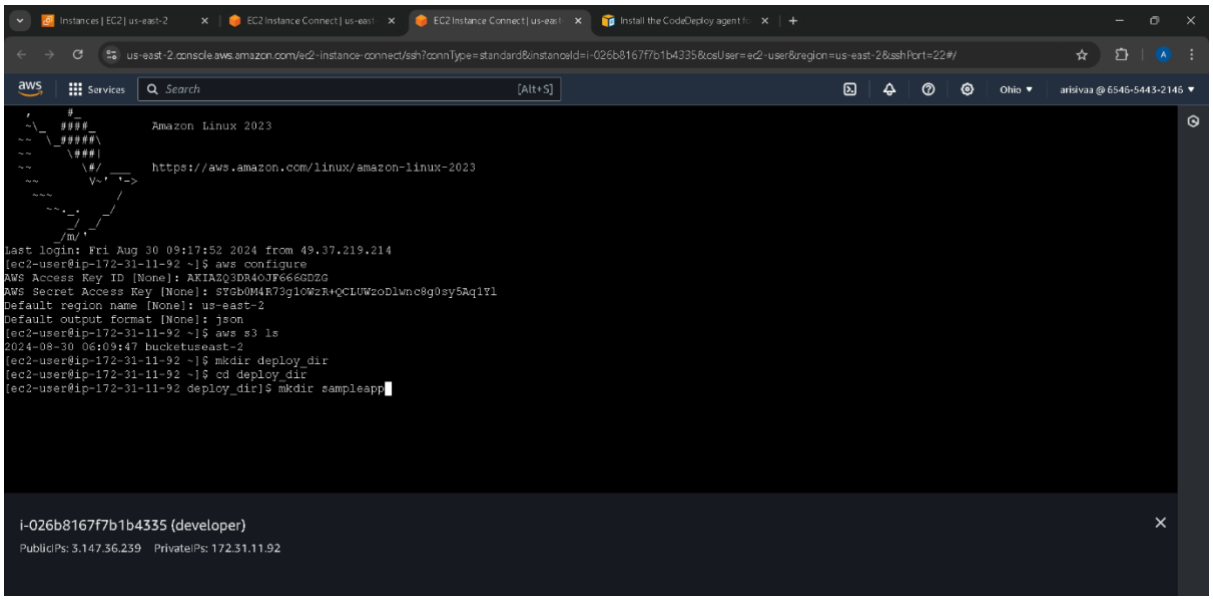


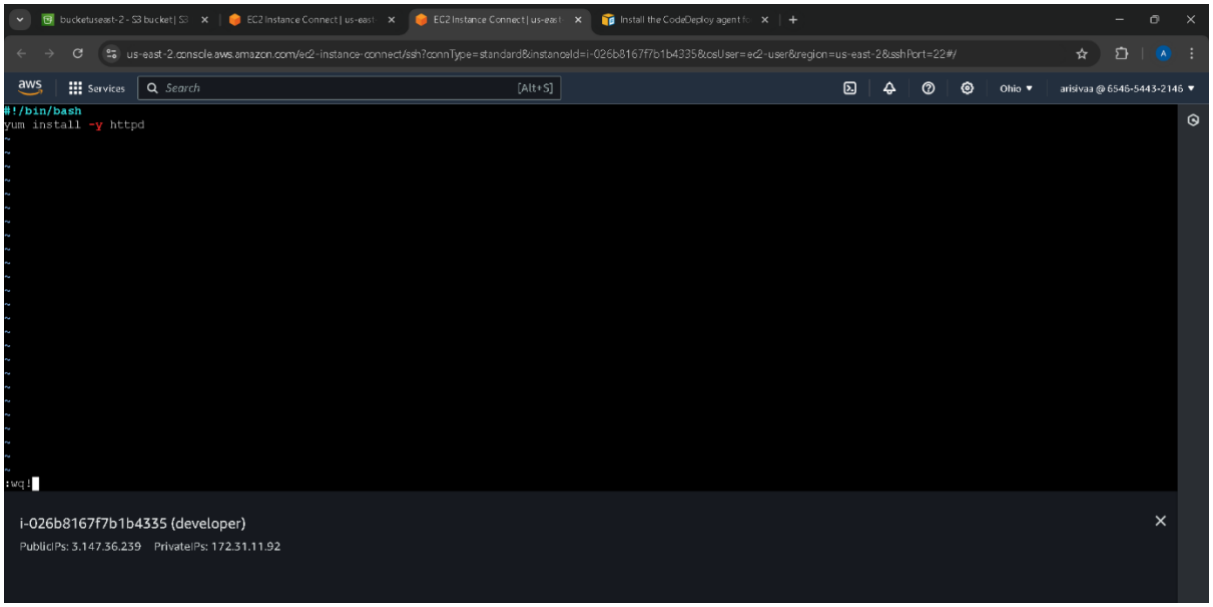
:wq!

Step 7: Create a scripts directory

```
[ec2-user@ip-172-31-11-92 sampleapp]$ mkdir scripts
[ec2-user@ip-172-31-11-92 sampleapp]$ cd scripts
[ec2-user@ip-172-31-11-92 scripts]$
```

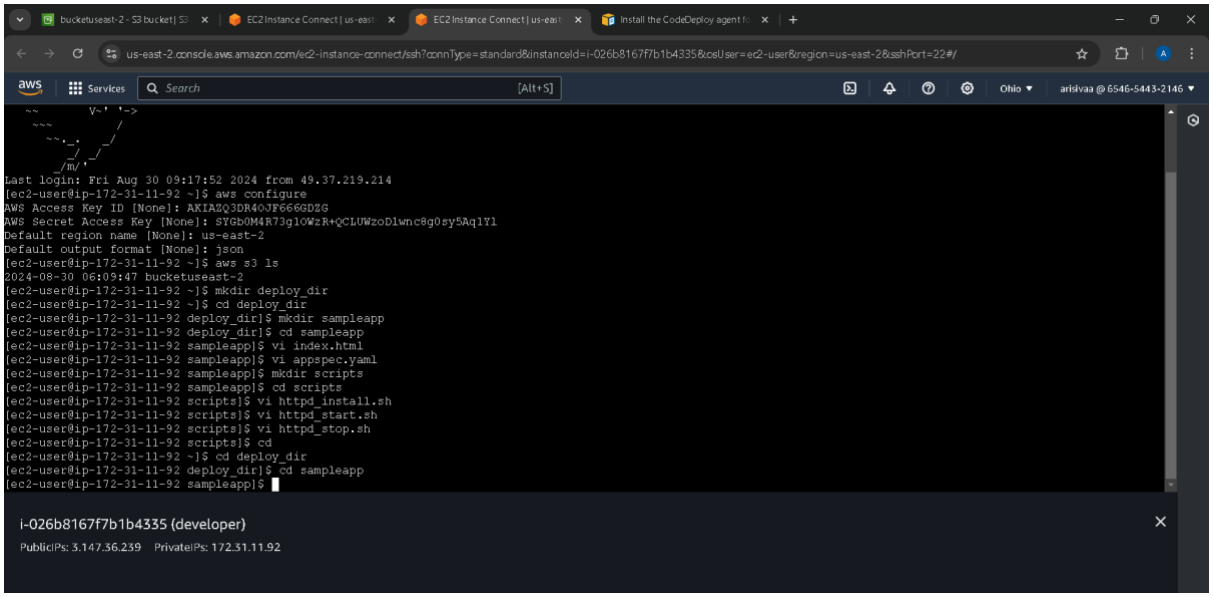
The first one is httpd_install.sh

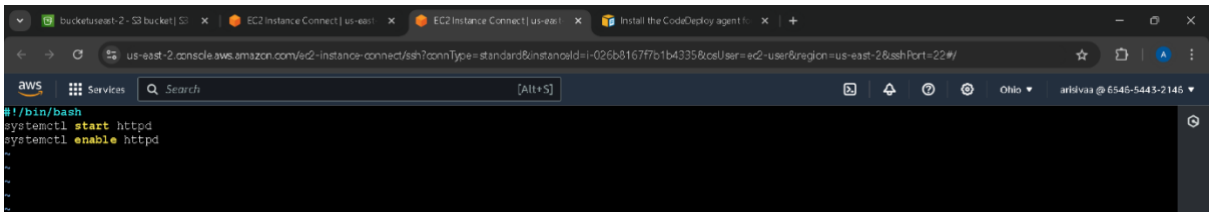




:wq!

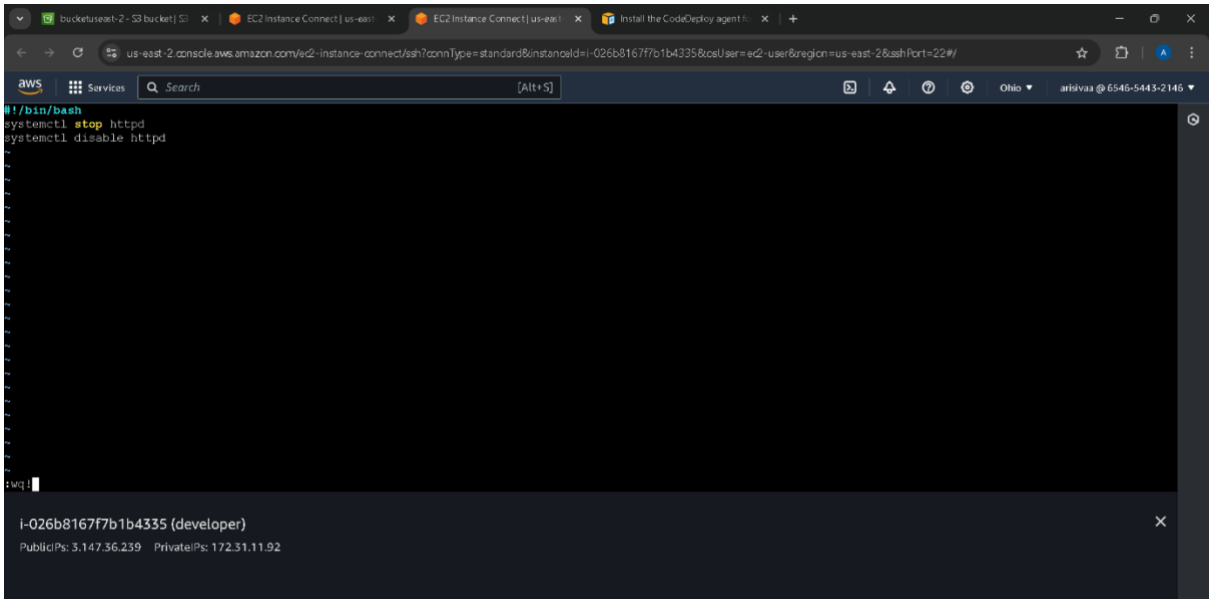
The second one is **httpd_starrt.sh**





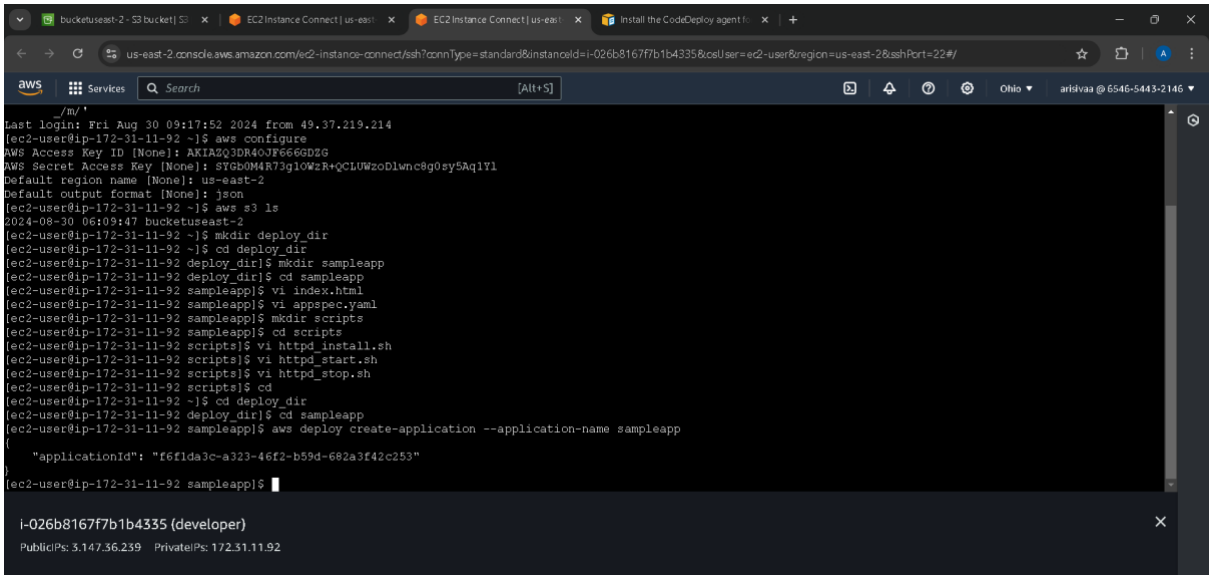
:wq!

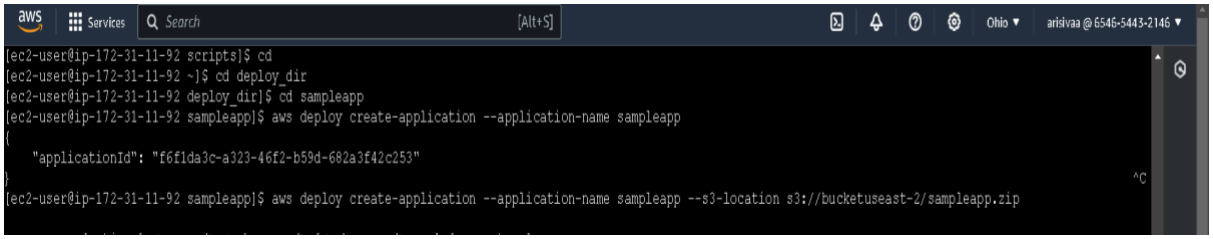
The third one is **htppd_stop.sh**



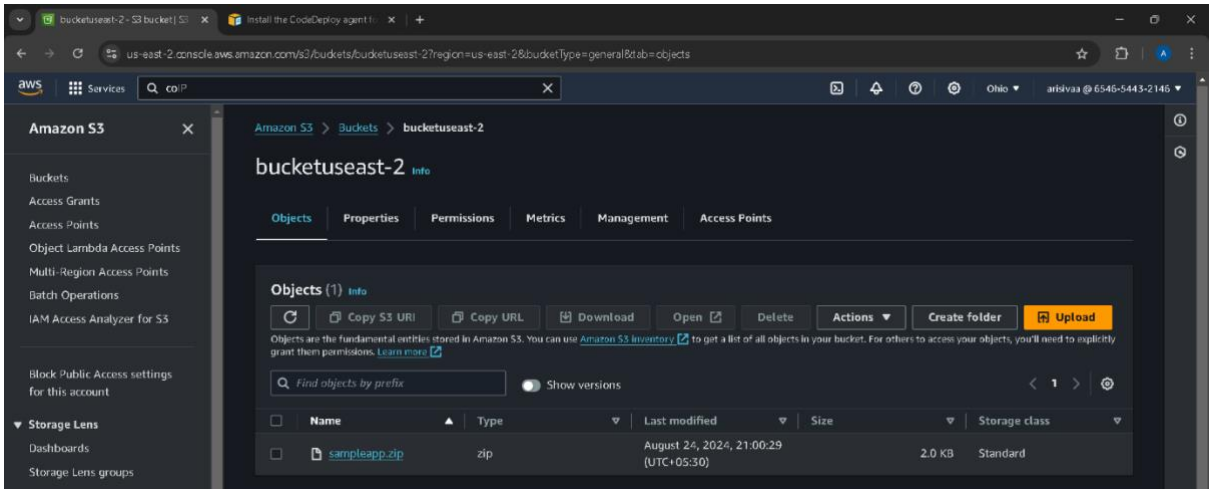
:wq!

Step 7: Create the code-deploy application and push the code to the s3 bucket developer machine

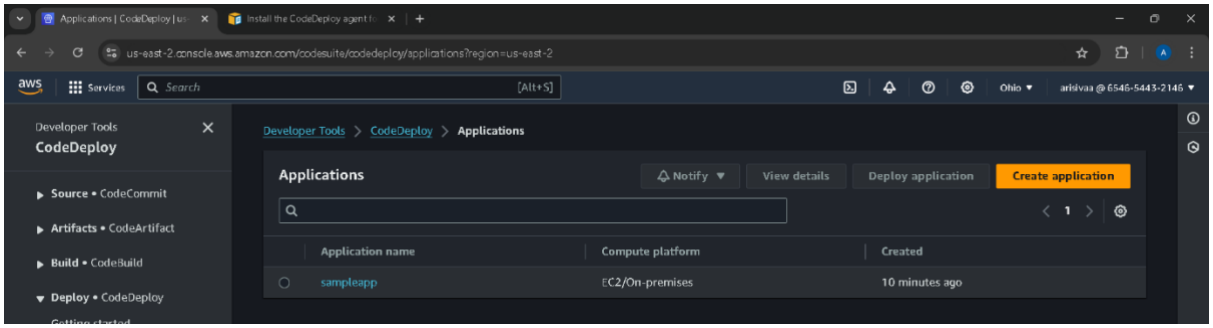




a) **create a bucket name called bucket bucketuseast-2 – make it public policy**

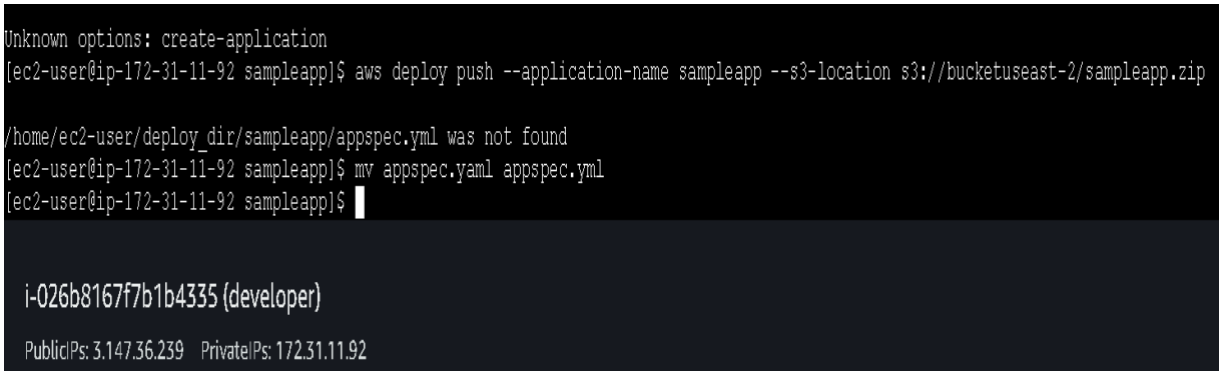


Check the Codedeploy in the Console Level in the GUI

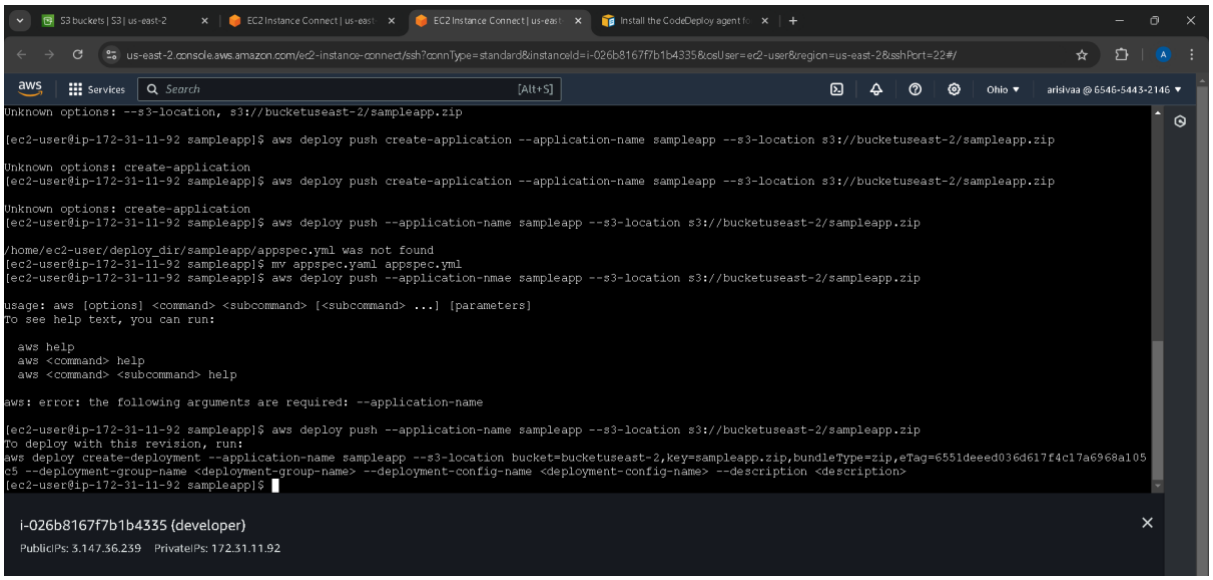


Next, create a s3 command

 # aws deploy push --application-name sampleapp --s3-location s3://bucketuseast-2/sampleapp.zip



Successfully create a s3 command.



Step 8: Create a deployment group.

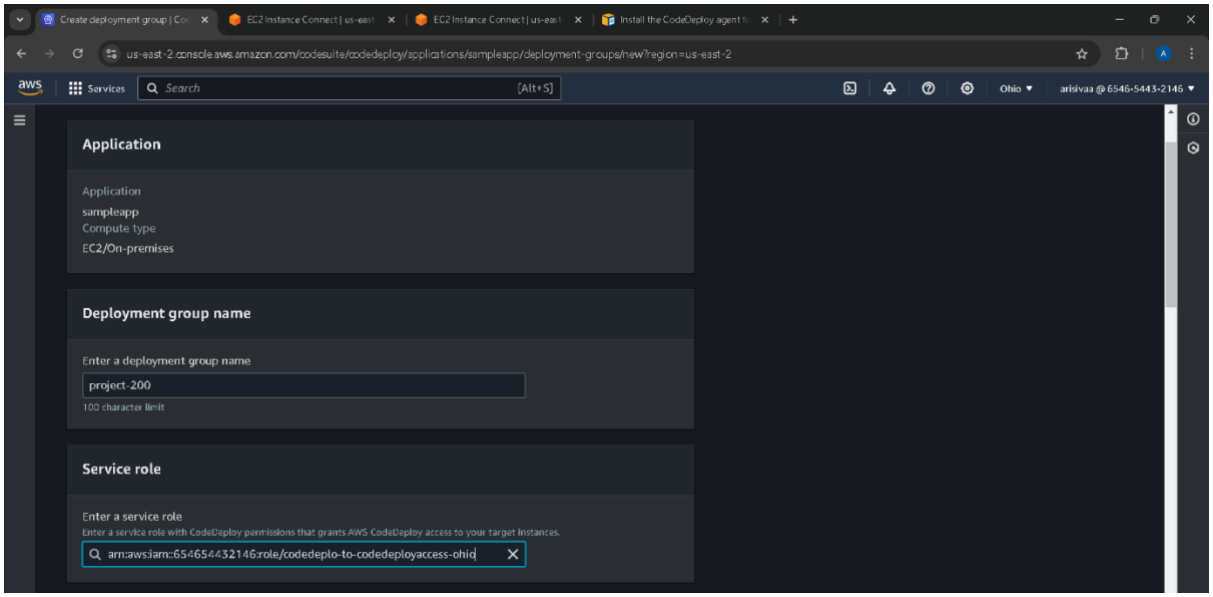
Application:

Sampleapp

Compute type:

EC2/on premises

Create a deployment group name and service role



The next one choose how to deploy your application (in place or blue/green)

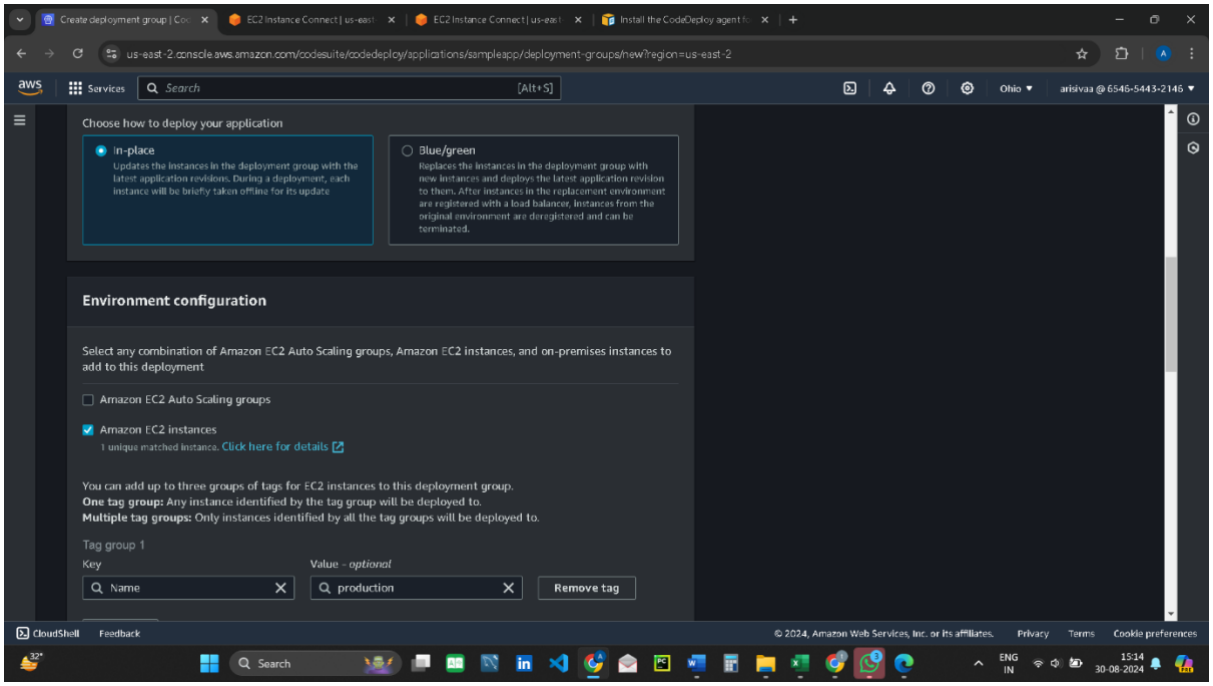
Choose the environment configuration

Amazon EC2 instances

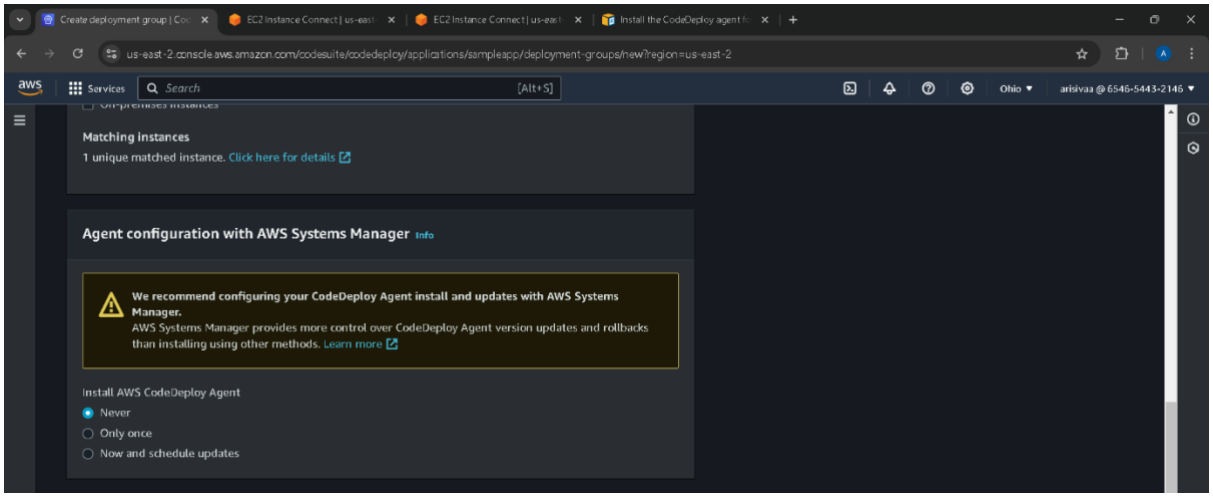
Tag group:

Key: name

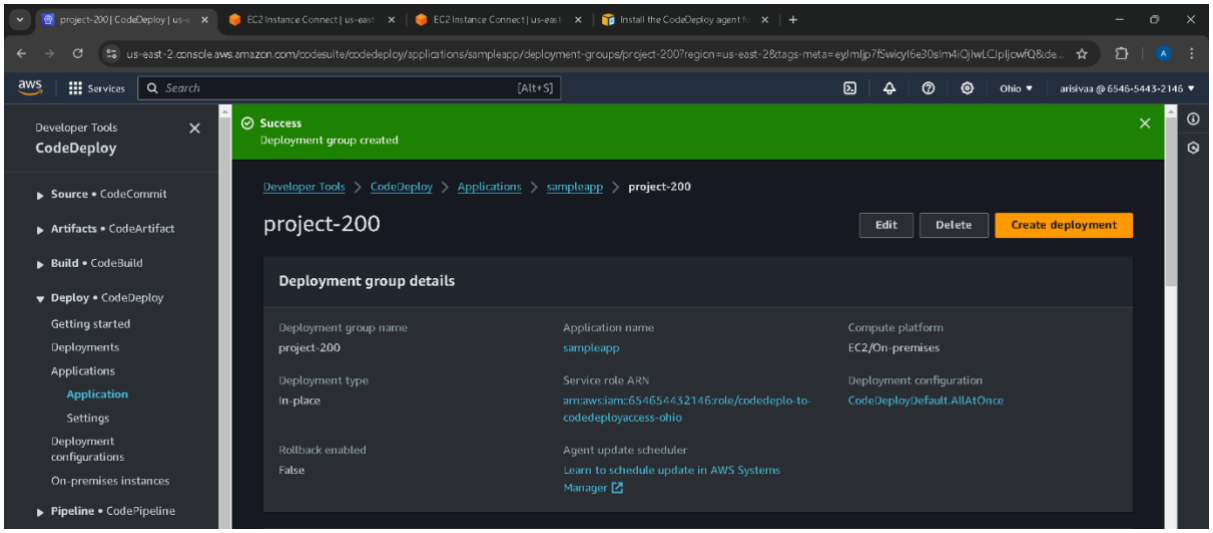
Value: production



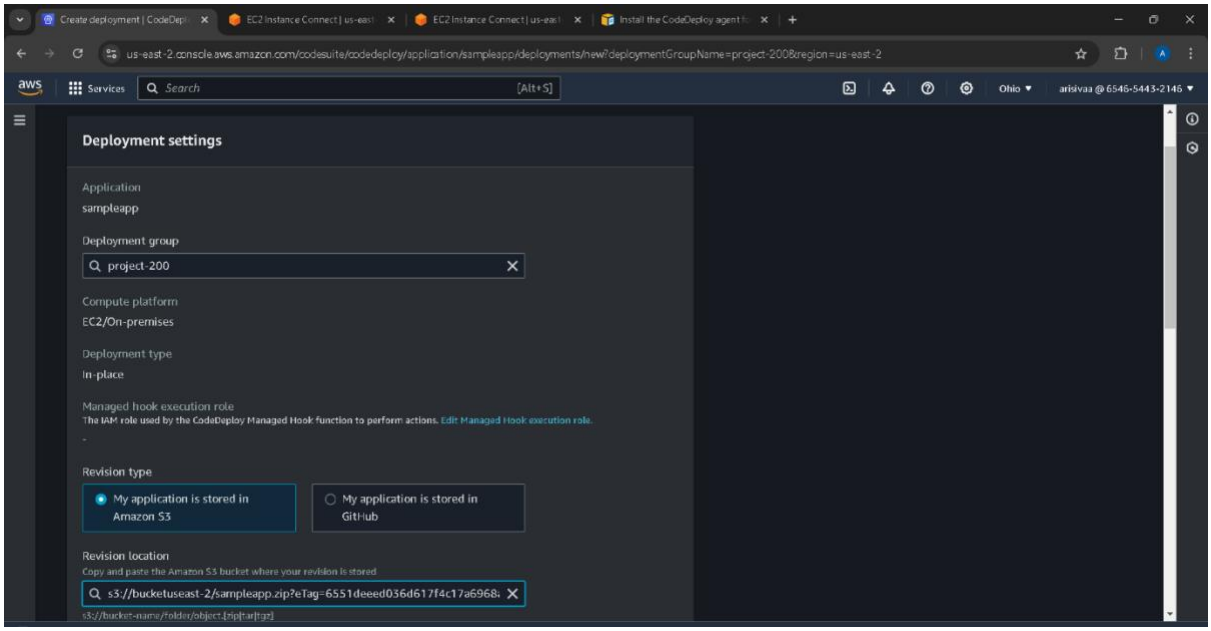
Install AWS code-deploy agent **Never**



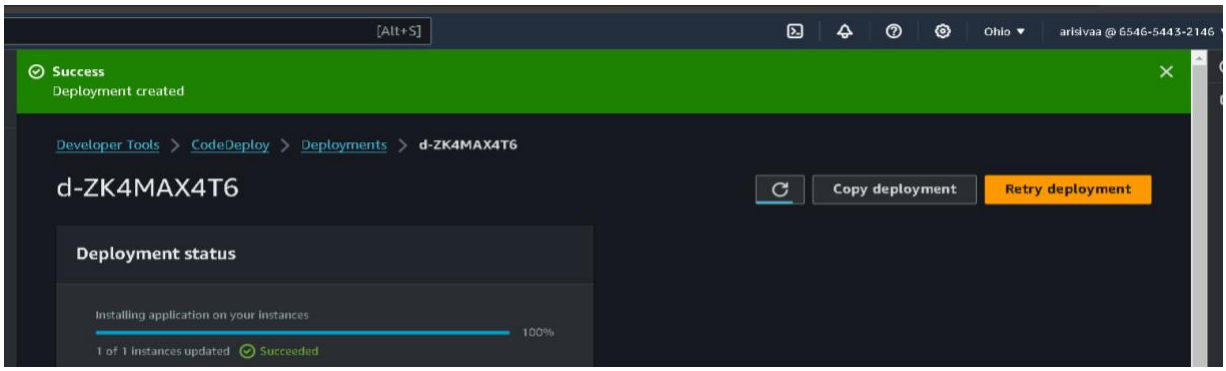
Successfully create a deployment group



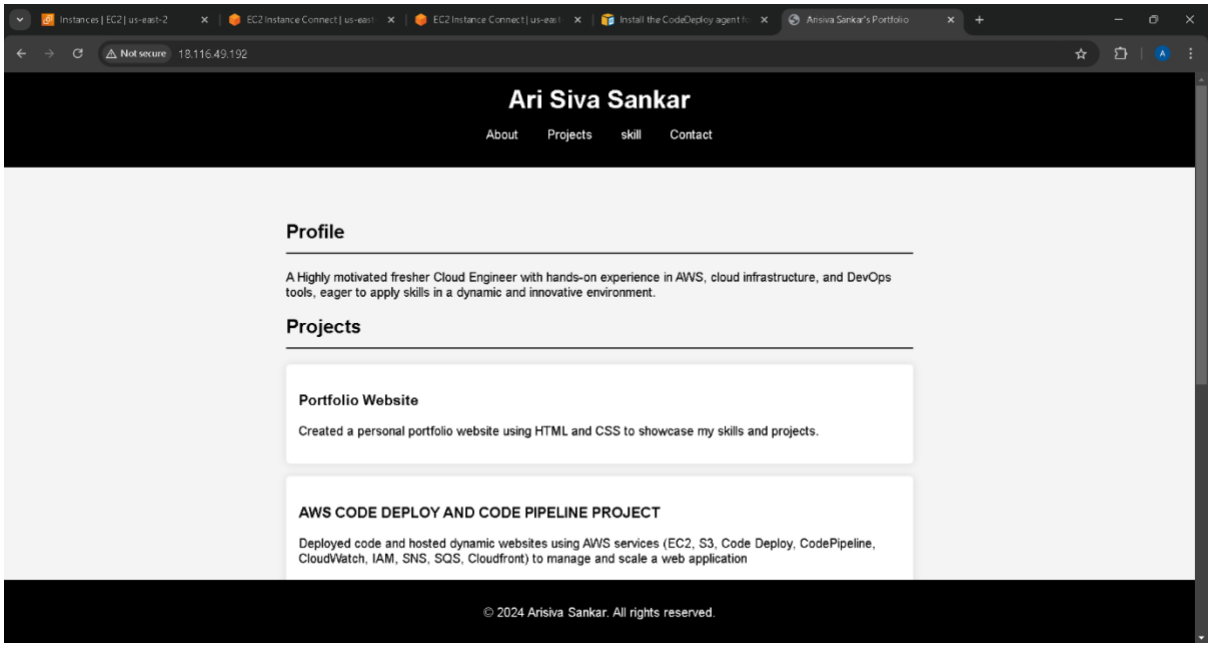
I have created a code deployment.



Successfully create code deployment.

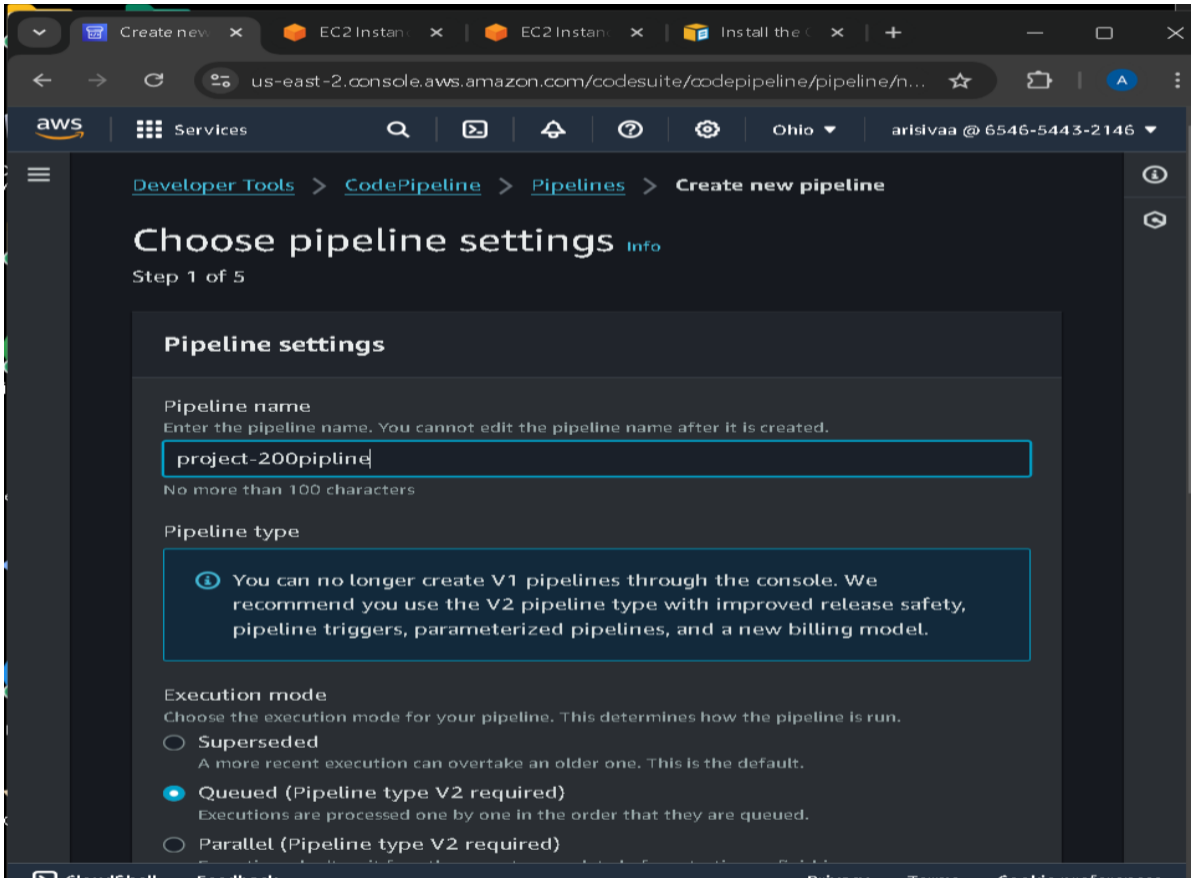


The next step copy the production machine's IP address and paste it Chrome browser next run the browser to check the output.

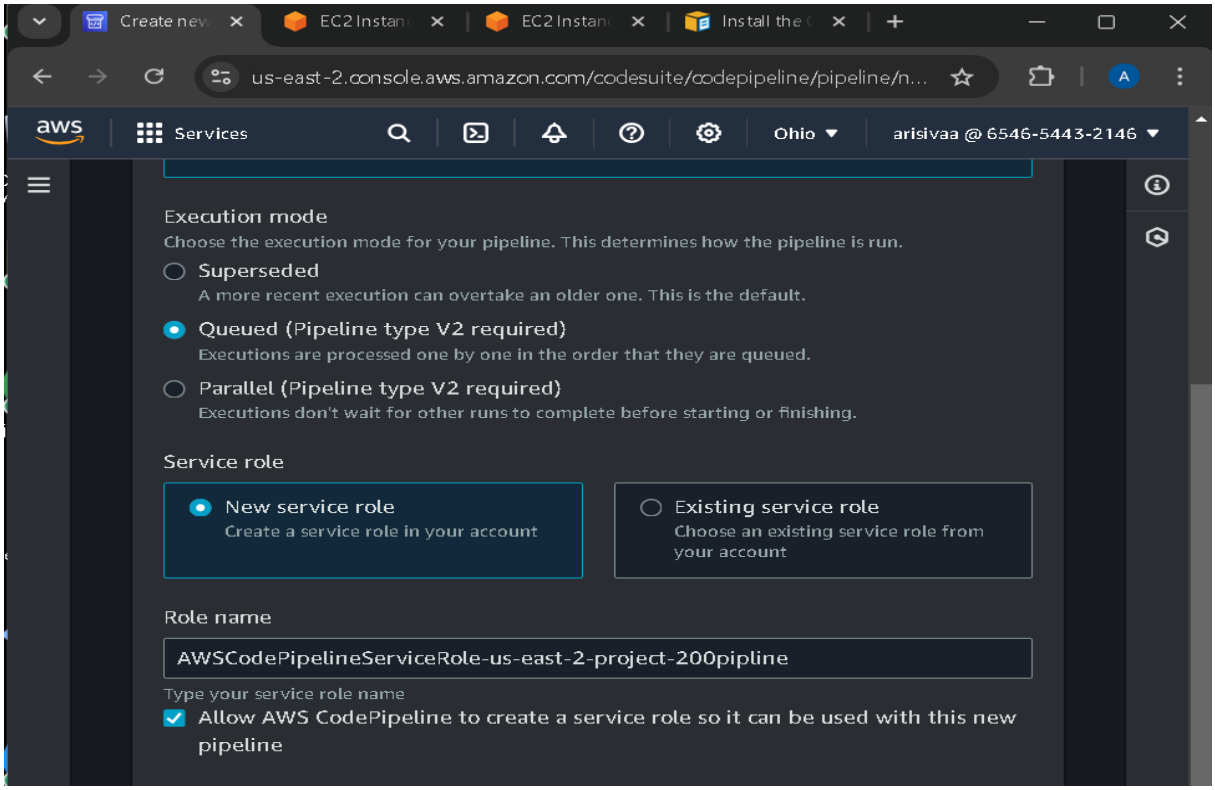


Step 8: Create a code pipeline

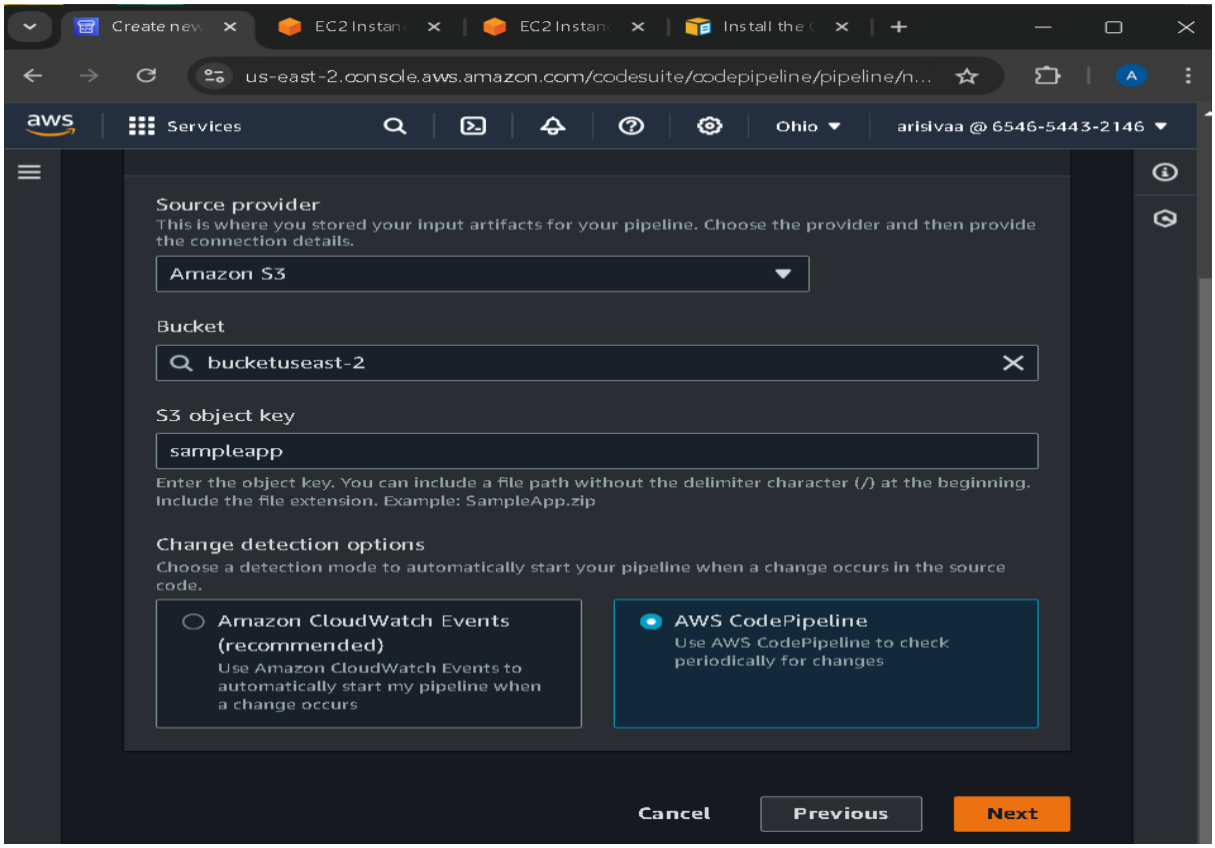
1) create a code pipeline name and Choose the pipeline type.



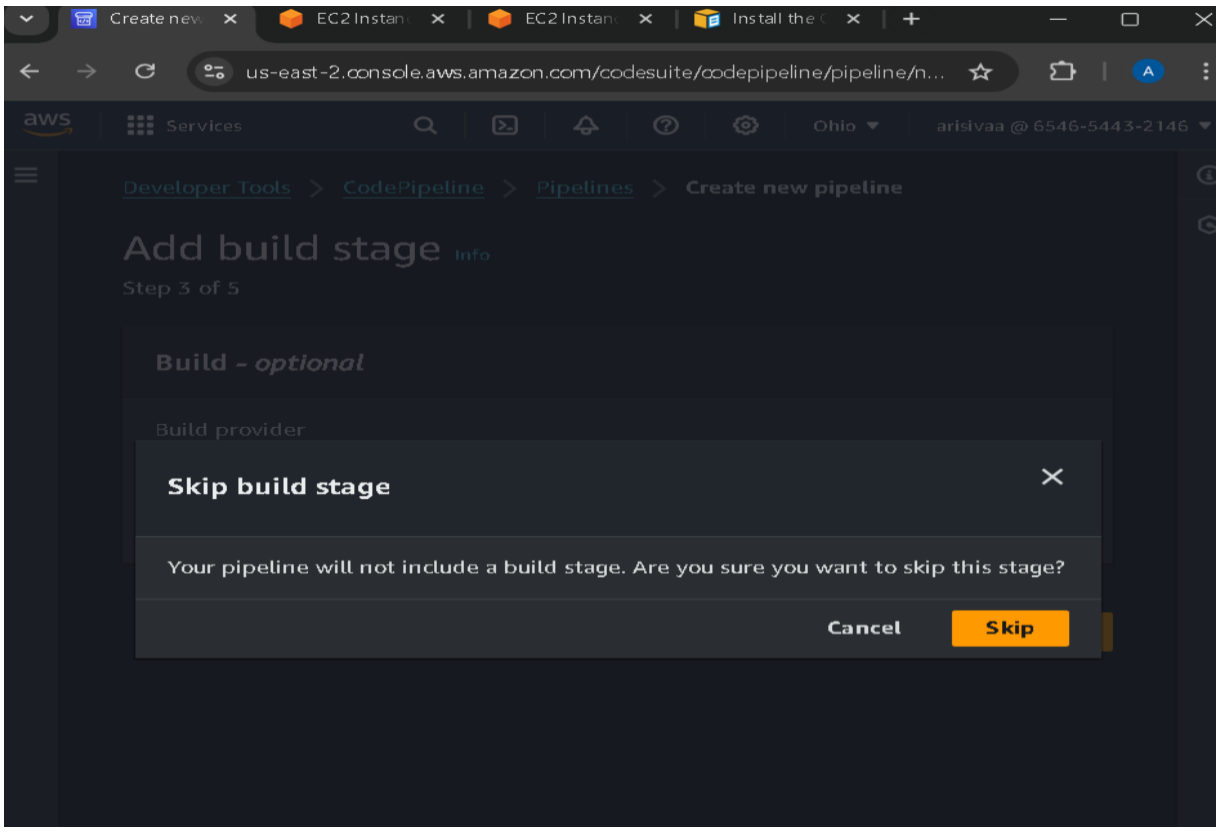
2) choose the exicted service role and role arn.



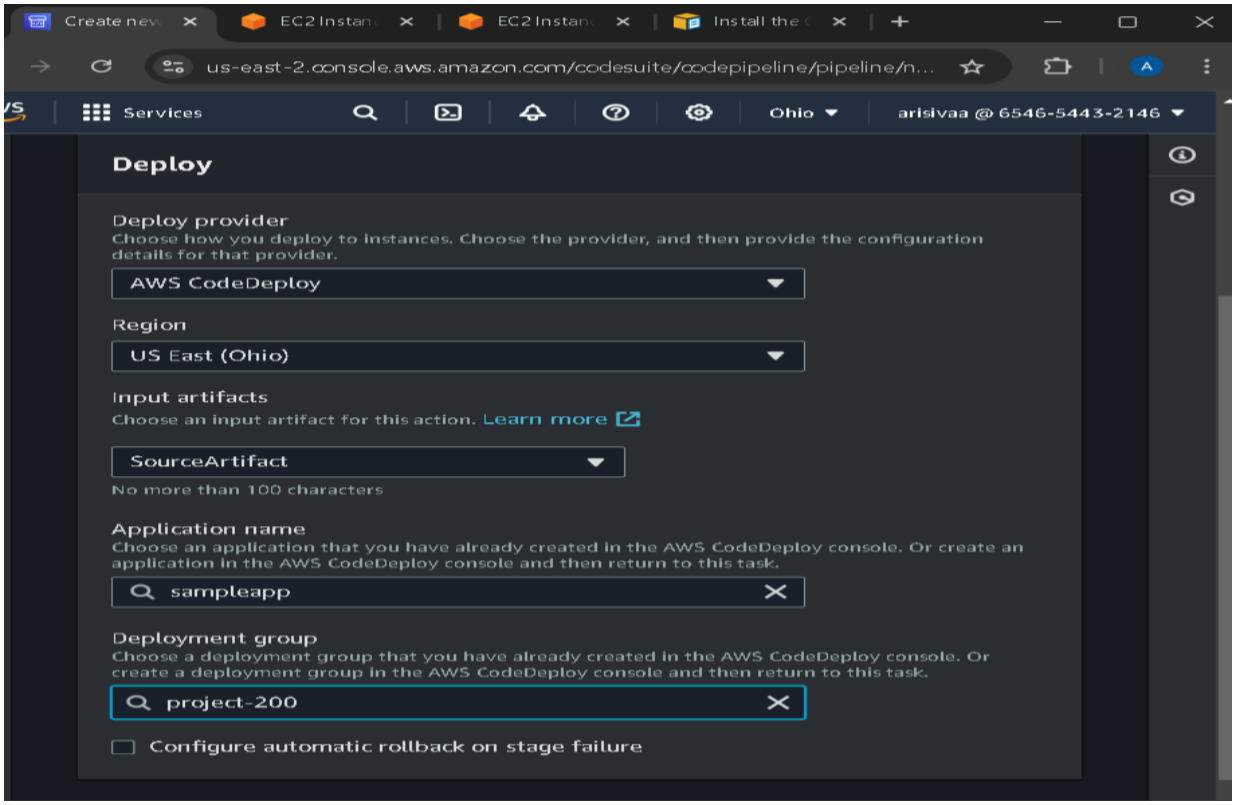
3) add source stage choose the bucket and select the object next one create a key value and choose detection option.



4) skip build stage.

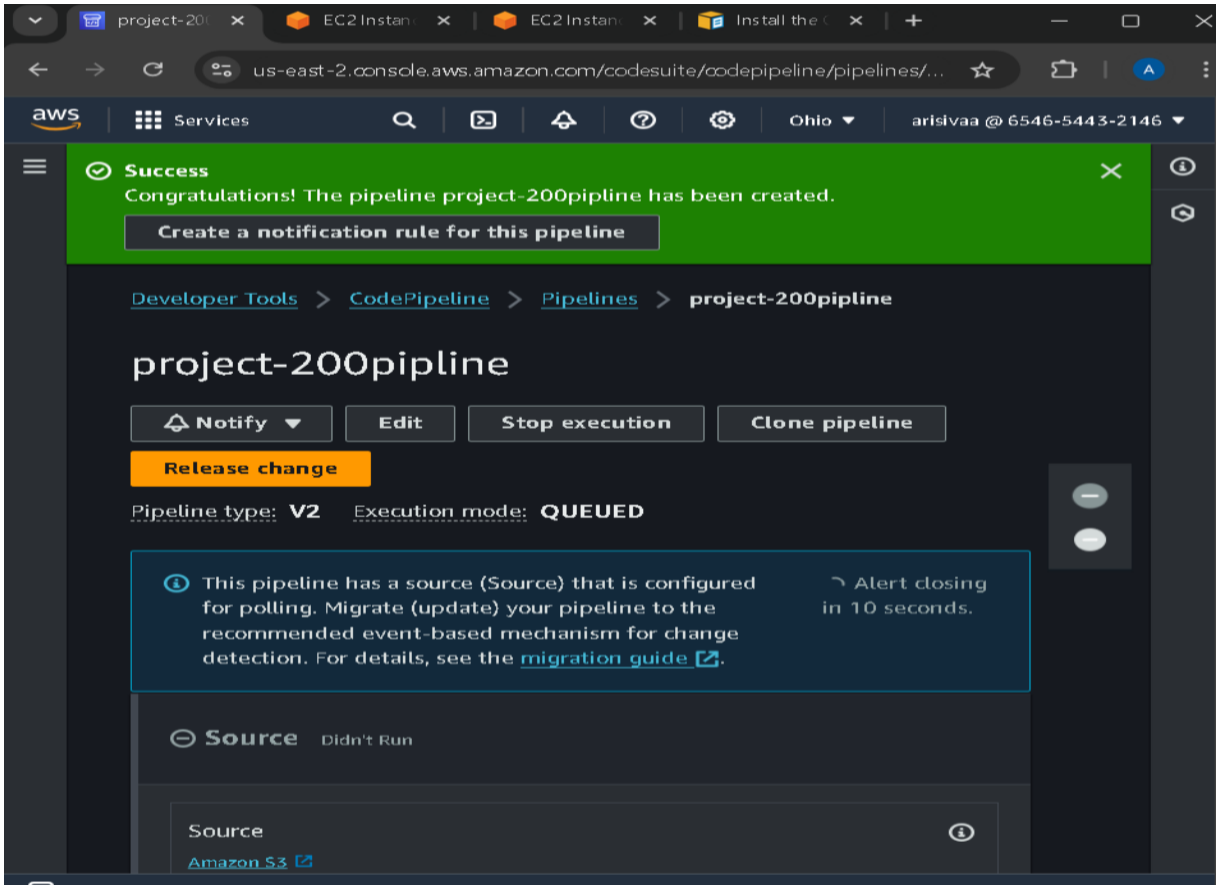


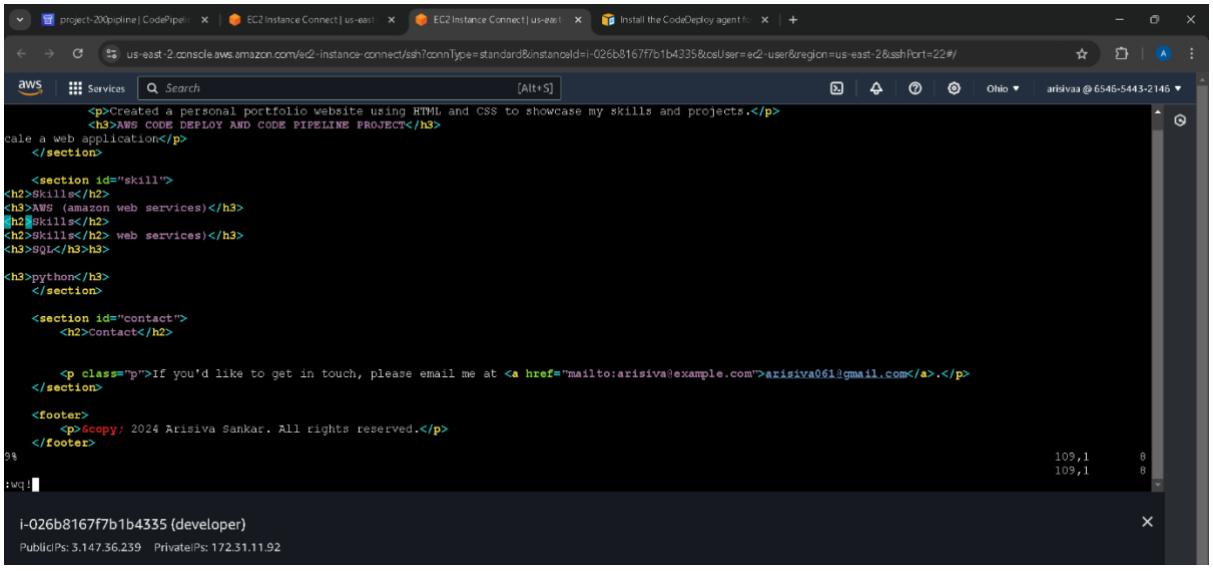
5) choose the deploy porivder and region. the second one select the input artifacts and then select the applicationnam and deployment group



6) Successfully create a codepipeline.

The last one enables the s3 versioning and then checks the code pipeline.





The next step edit the vi file **vi index.html**

Zip -r ../sampleapp.zip

Ls

Cd ..

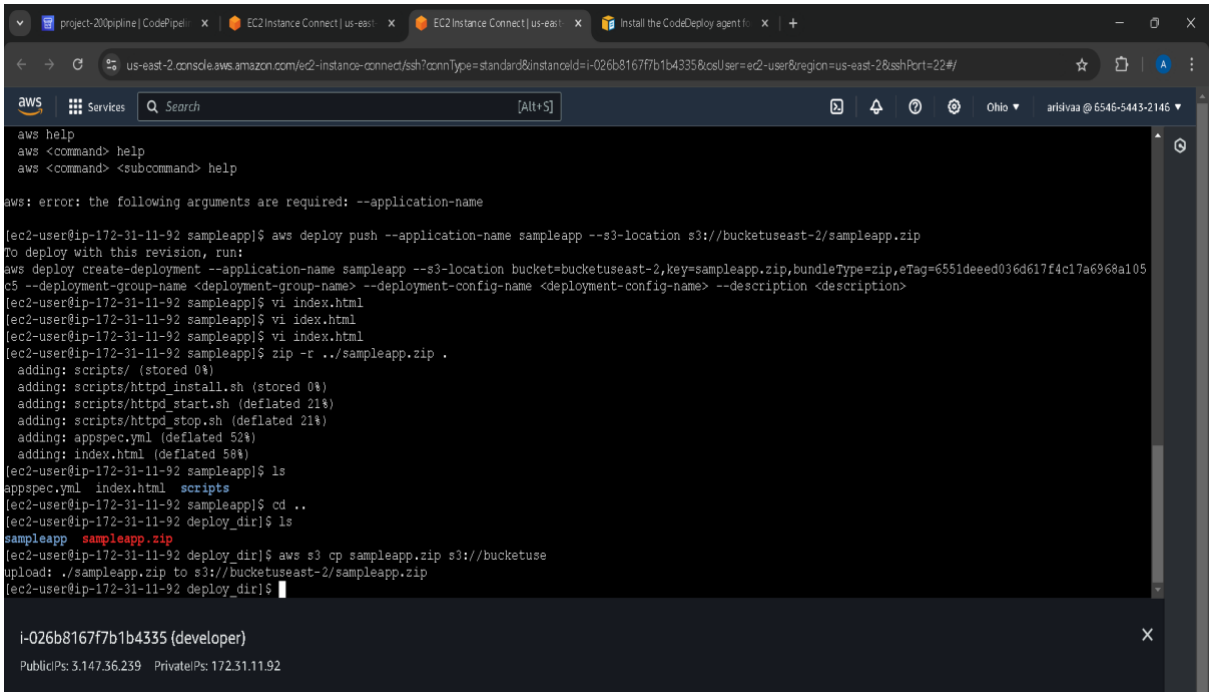
Ls

Cd ..

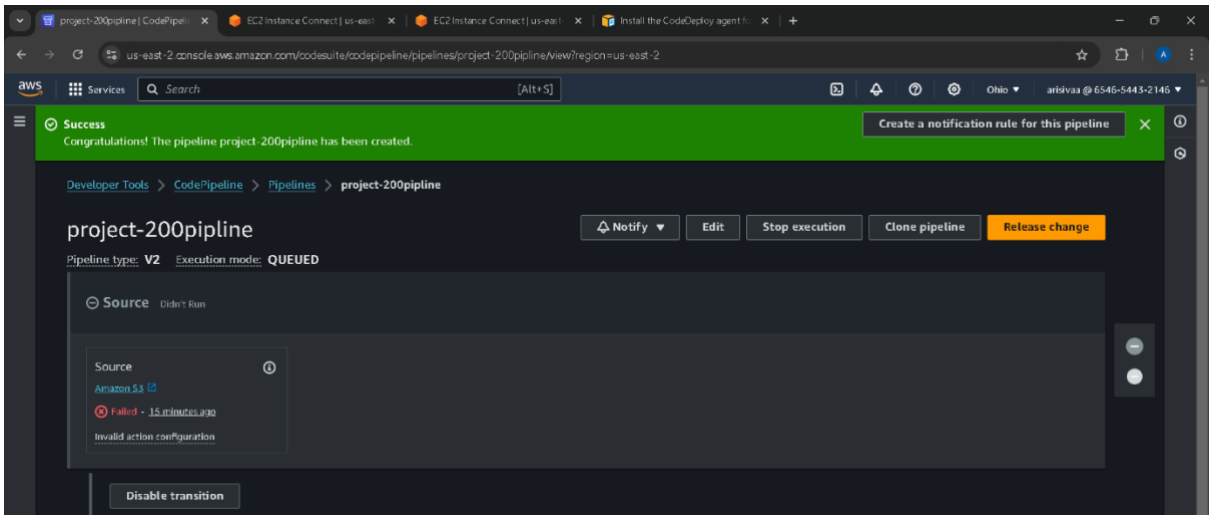
Ls

You see the sampleapp.zip

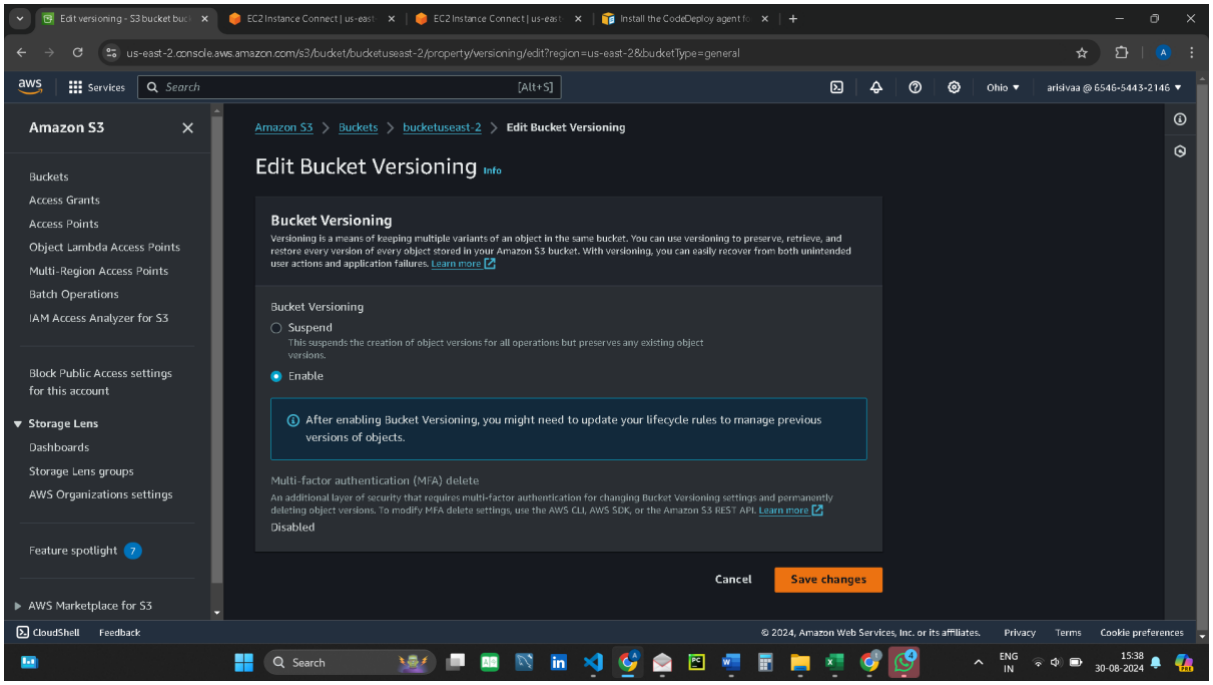
The next time, type this command. aws s3 cp sampleapp.zip s3://bucketuseast-2



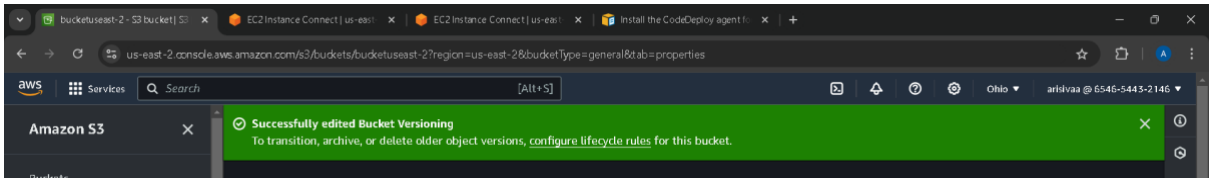
S3 invalid action configuration.



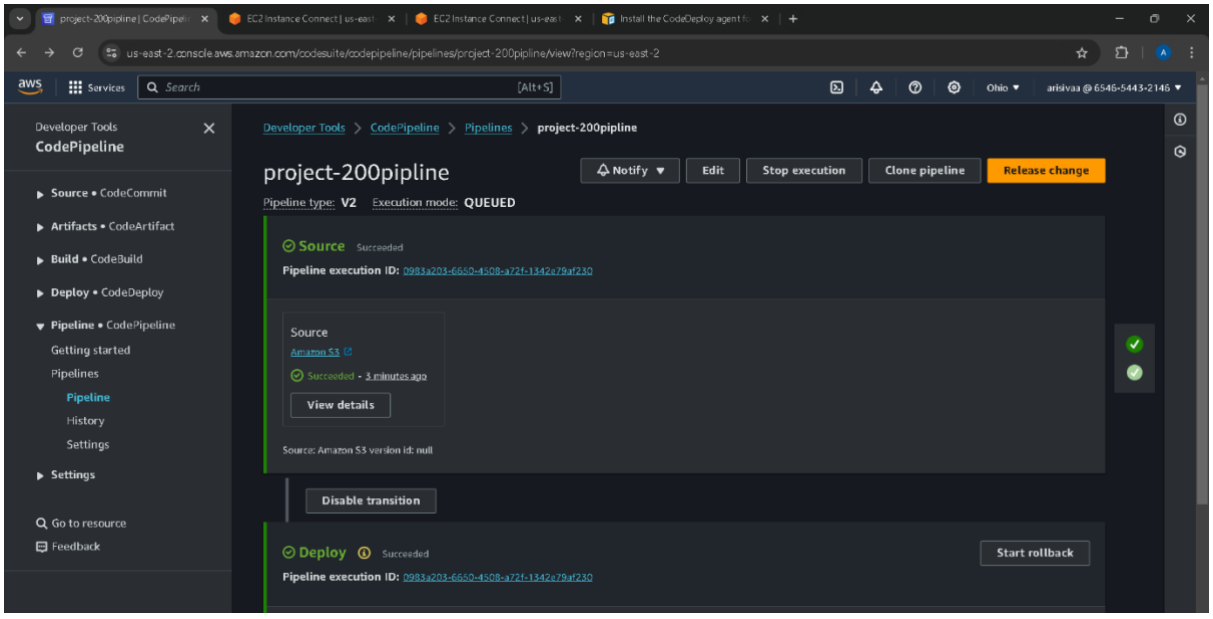
Edit bucket versioning. **enable**



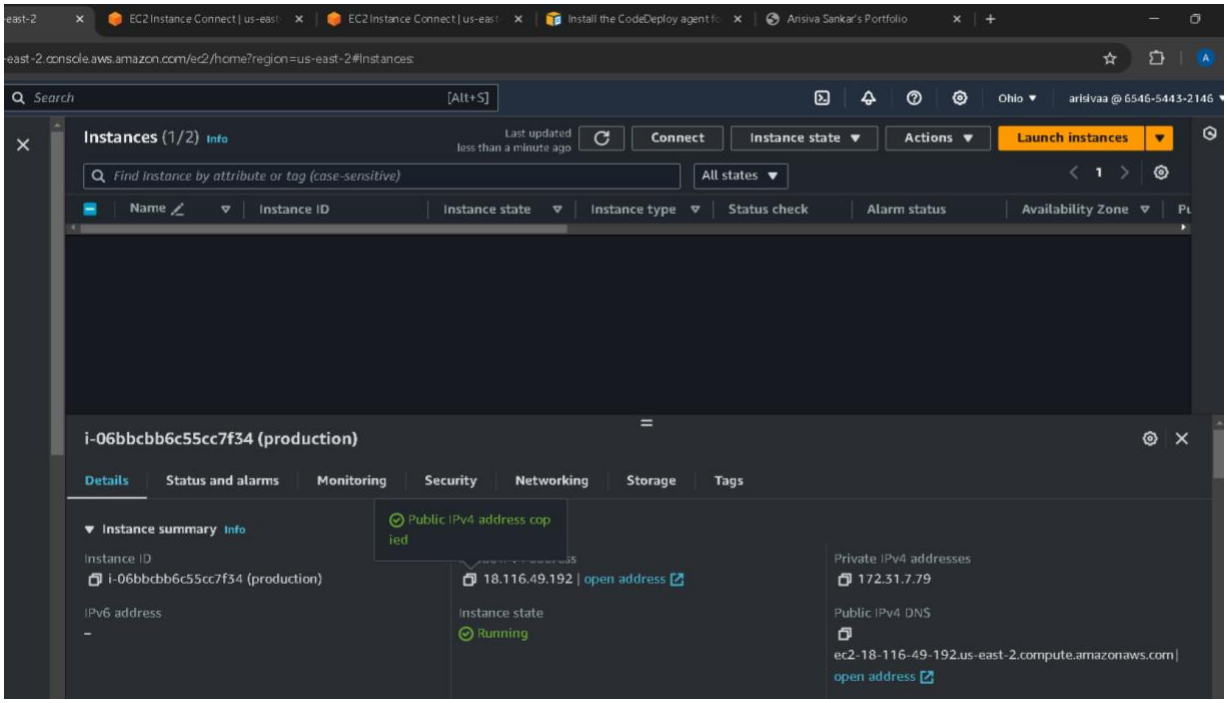
Successfully edited bucket versioning.



Successfully finished

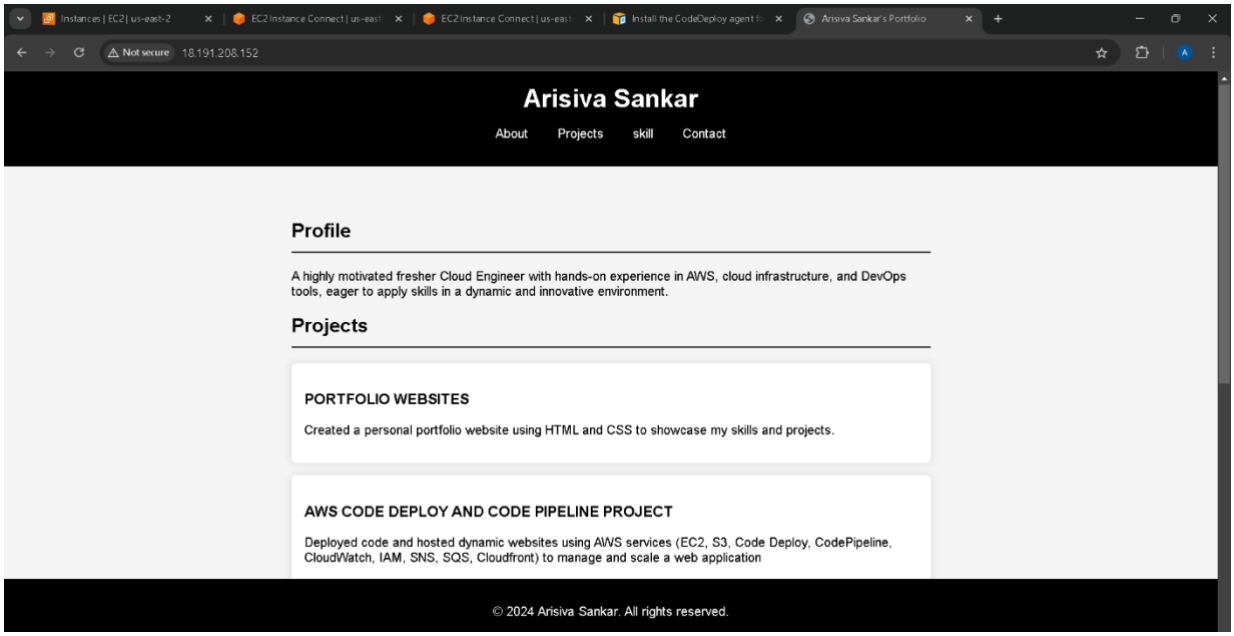


The next one copies the production machine IP address and pastes it into your browser.



Output

Successfully run my portfolio website

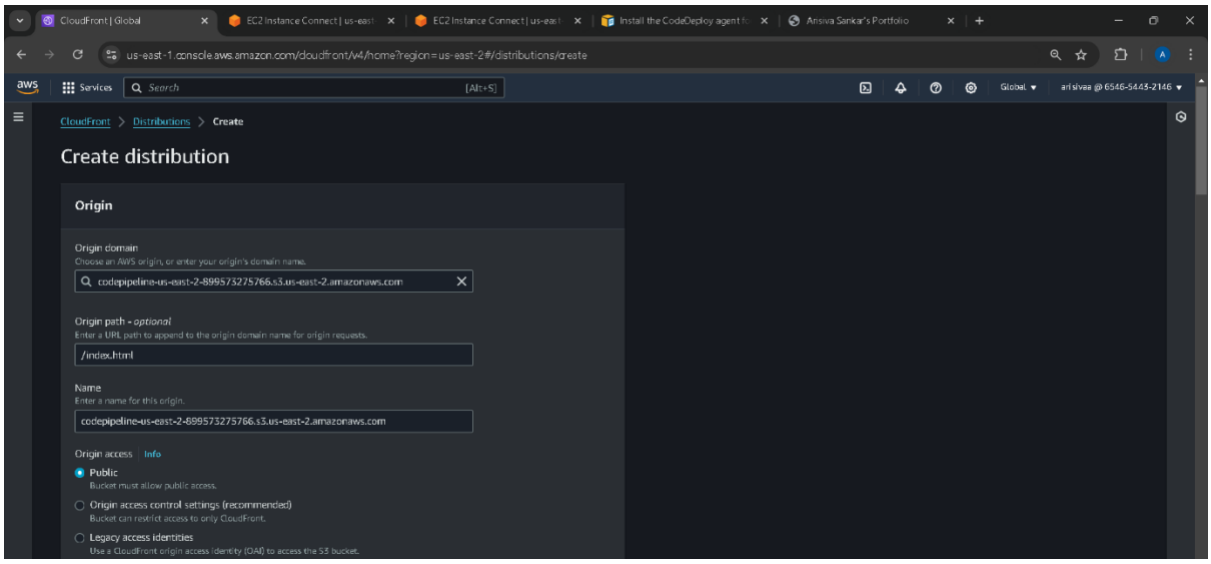


AWS SERVICES USED:

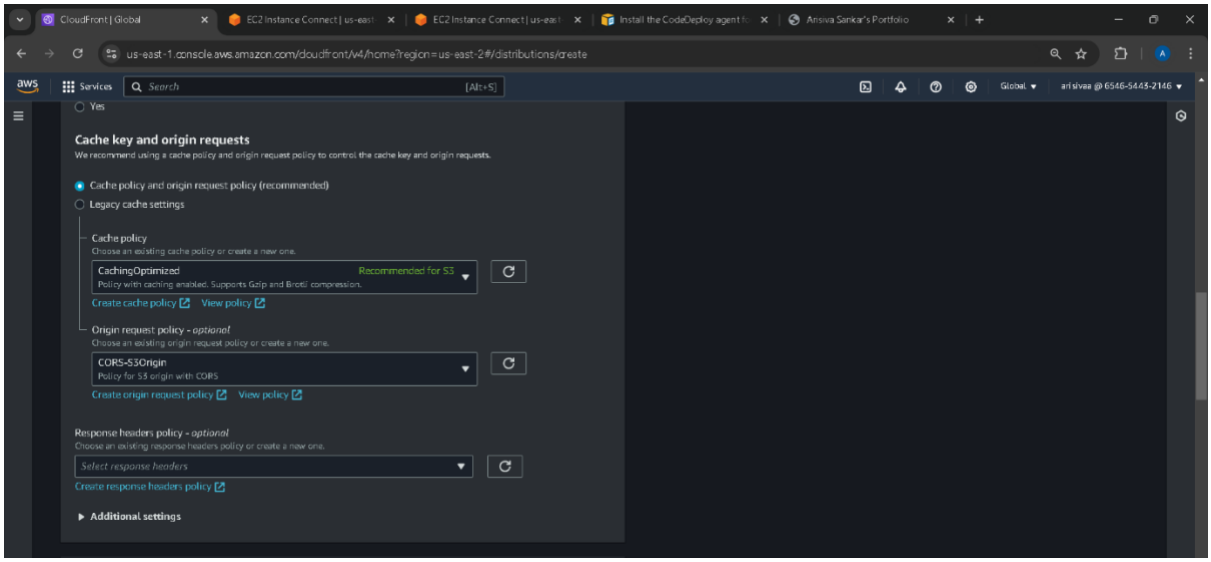
CloudFront:

AWS CloudFront is a content delivery network (CDN) that speeds up the distribution of your web content by caching it at edge locations globally, reducing latency for users. It also provides DDoS protection and supports seamless integration with other AWS services.

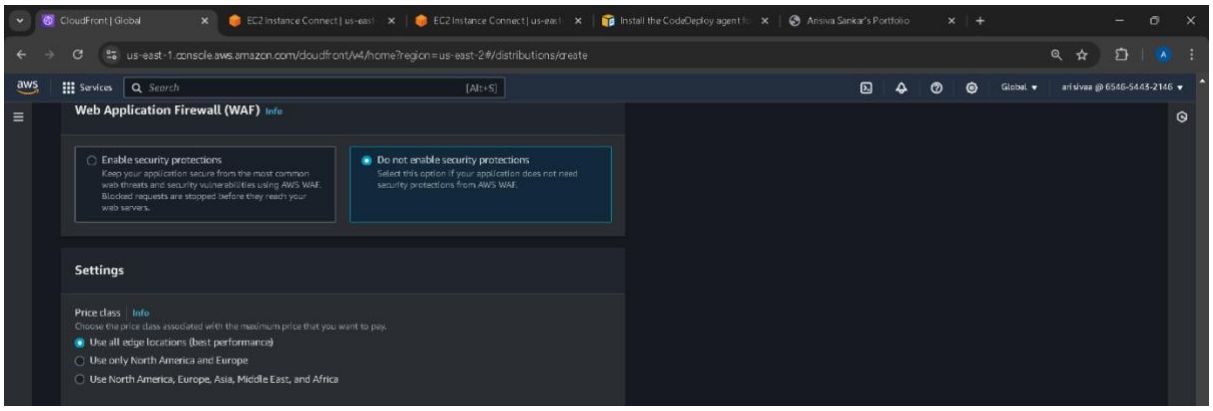
1) **Select the Origin and origin path then select origin access public**



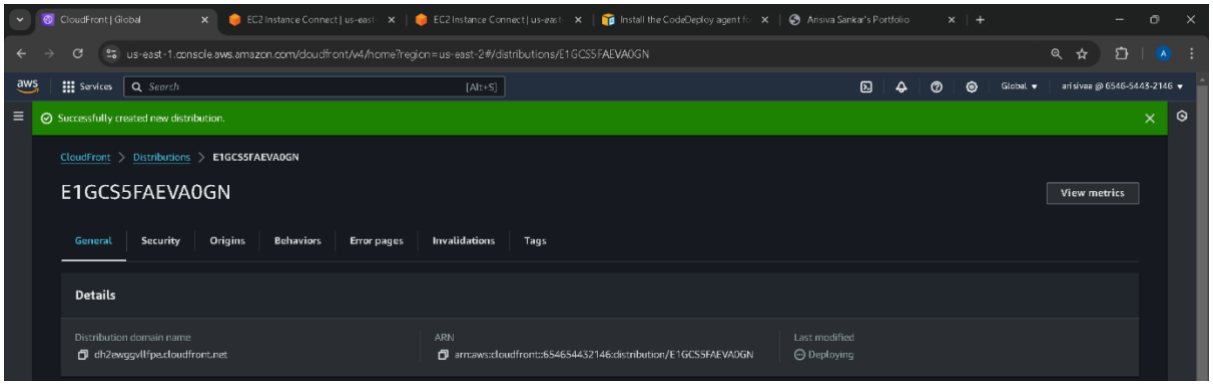
2) **Cache key and origin request.**



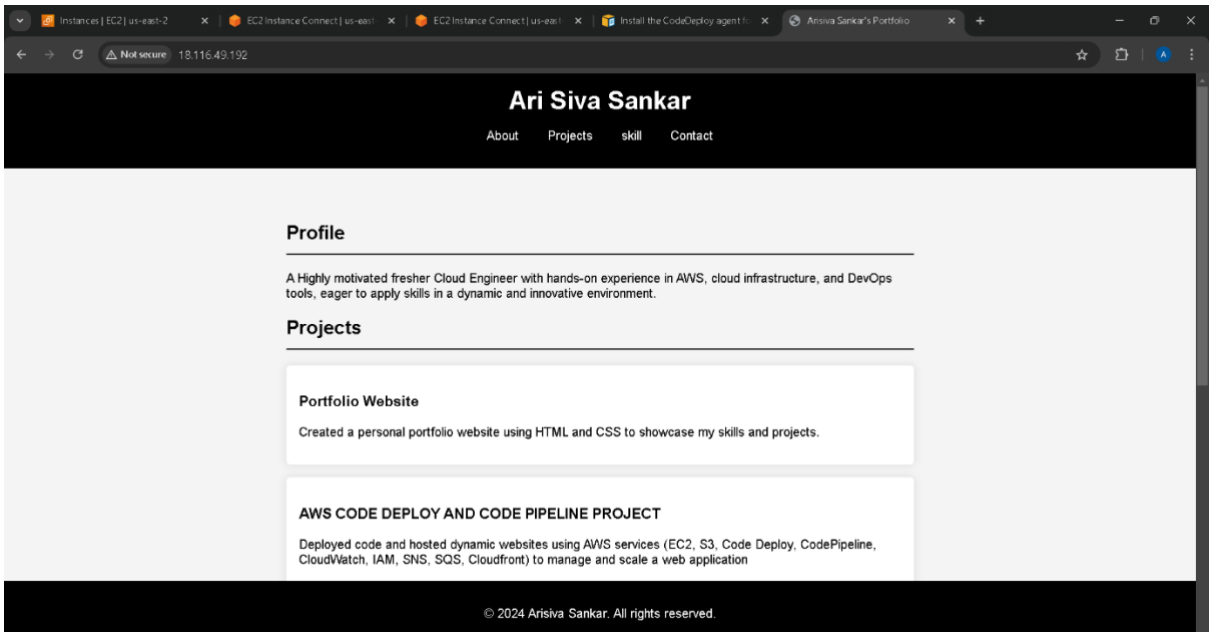
3) **Do not enable security protections and select the use all edge location best performances**



4) **Successfully created new distribution**

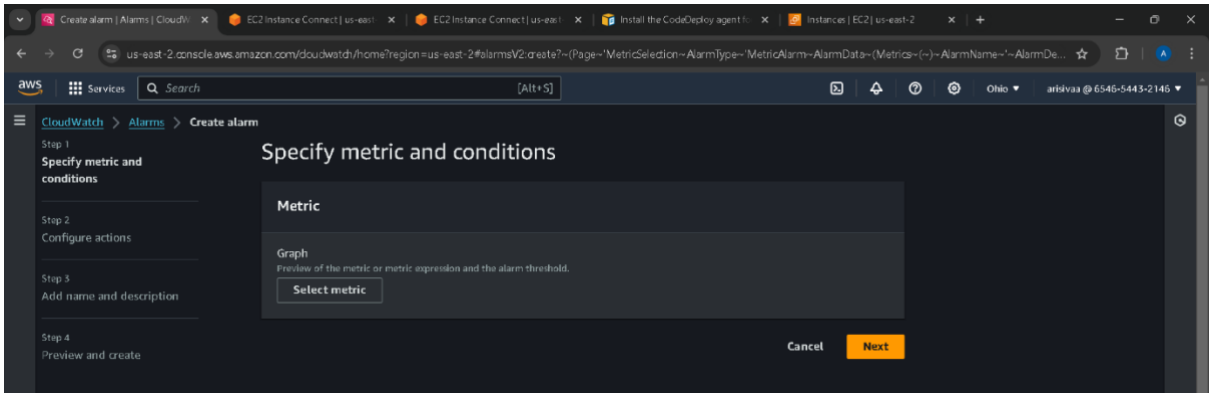


Output:

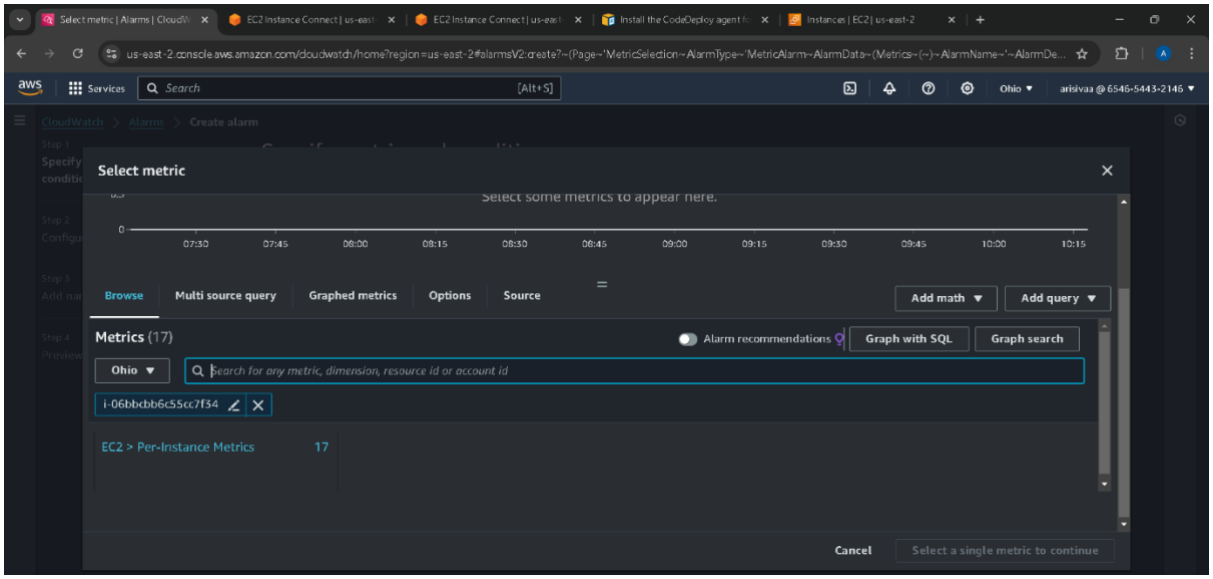


Cloudwatch:

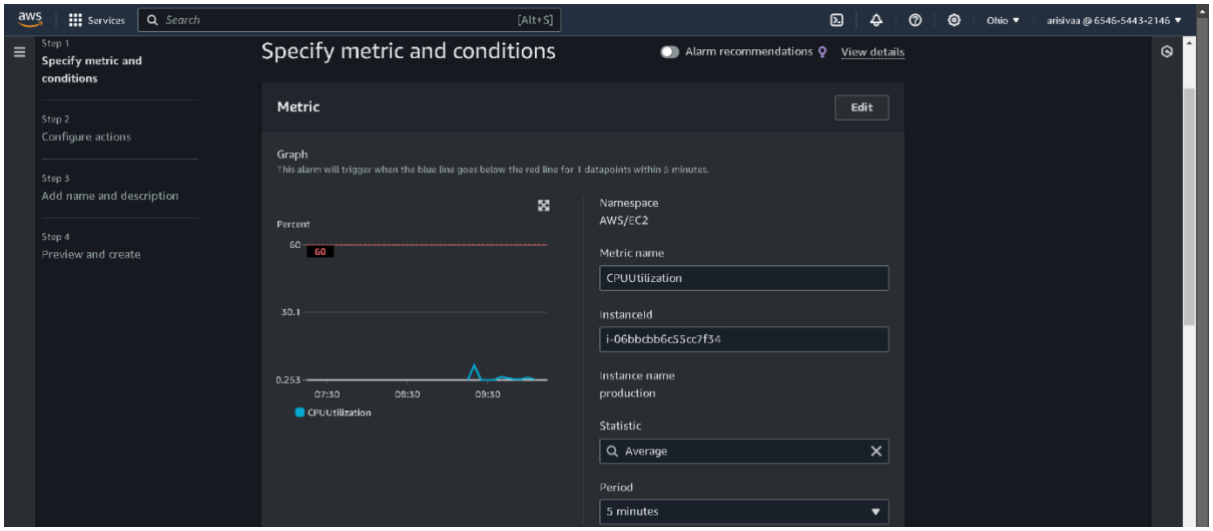
1) **Select metric.**



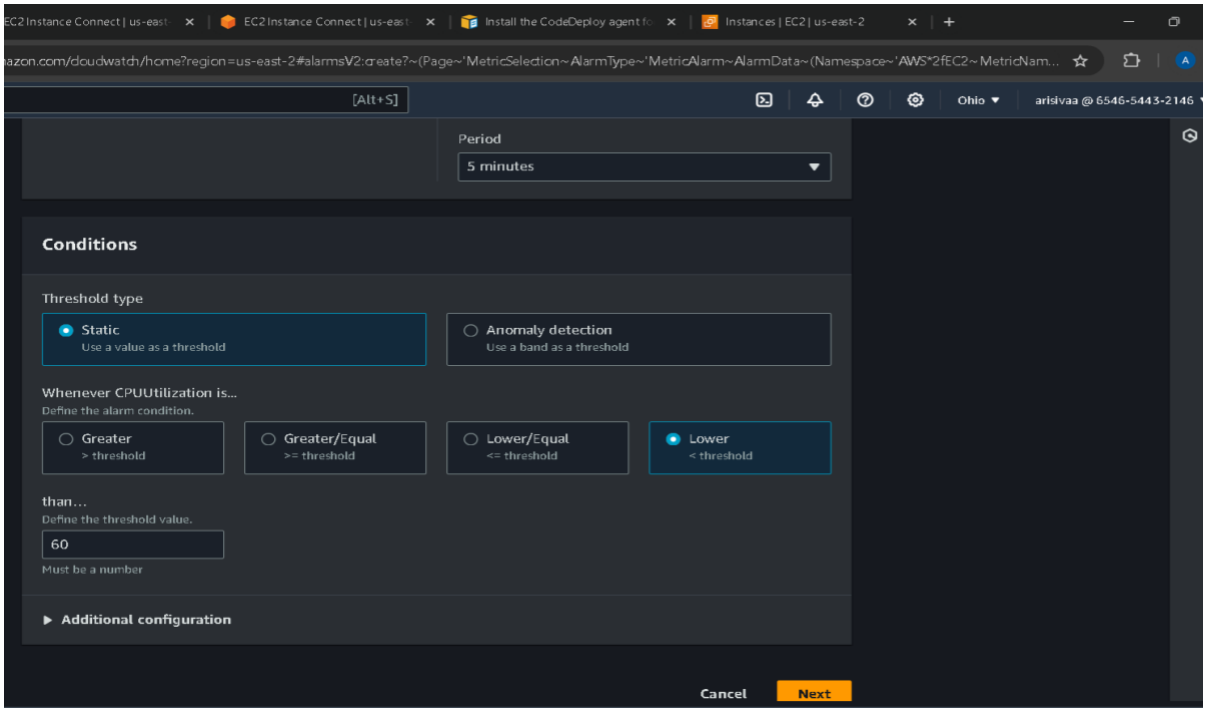
2) **Pre-instance metric.**



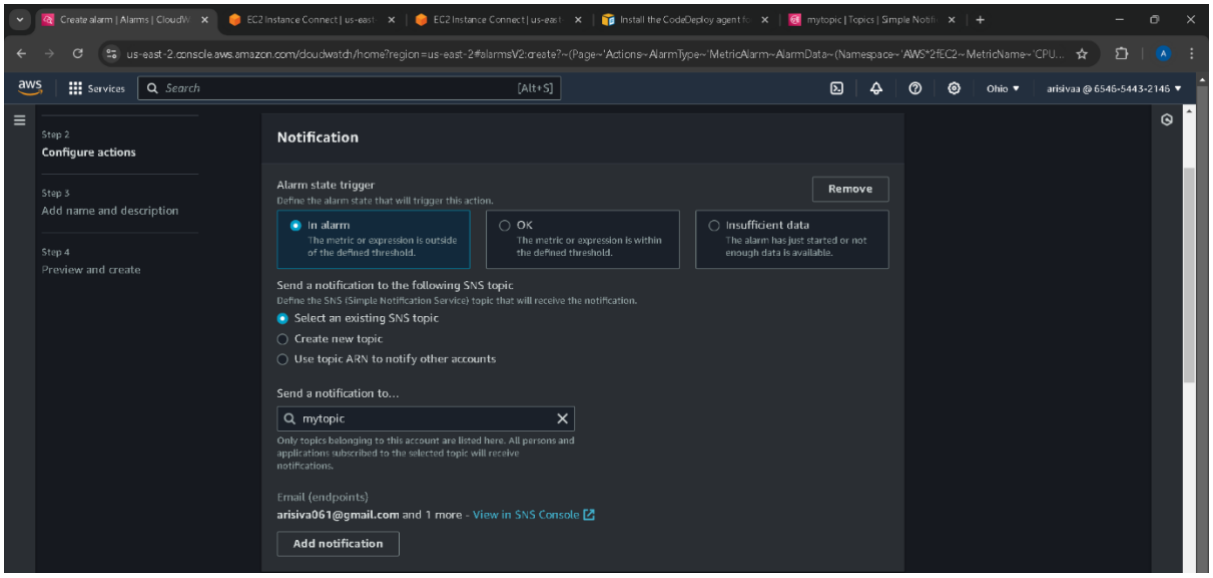
3) **Select the metric <u>CPU utilization</u>.**



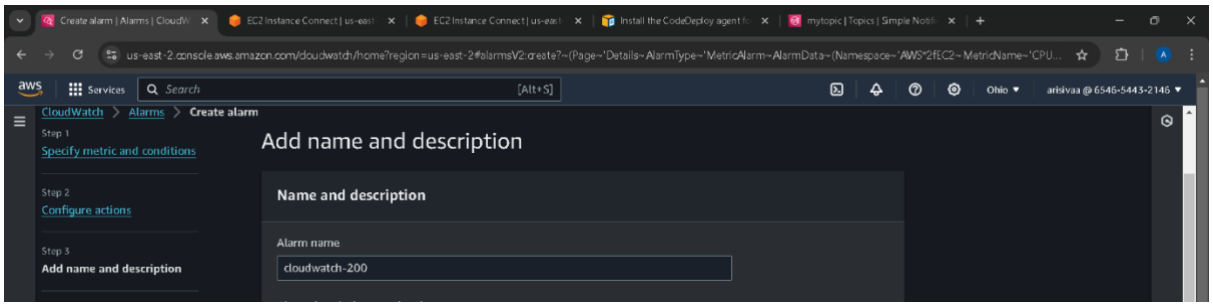
4) Select the conditions.



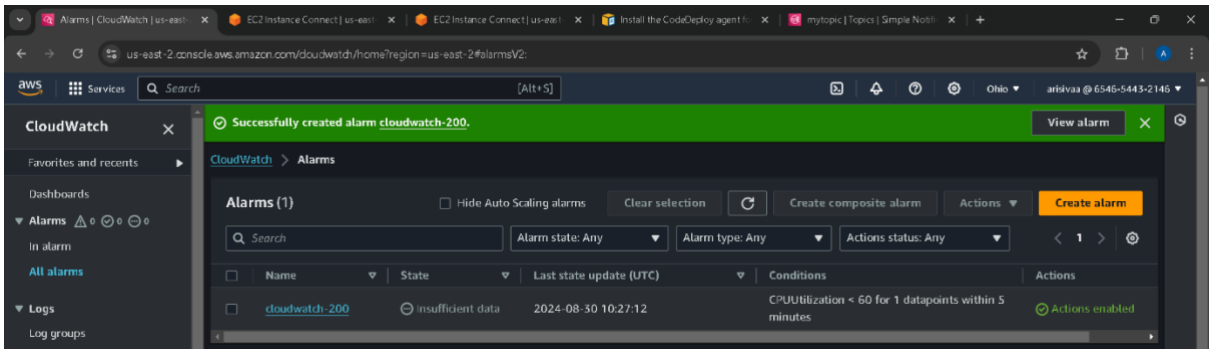
5) Notification.



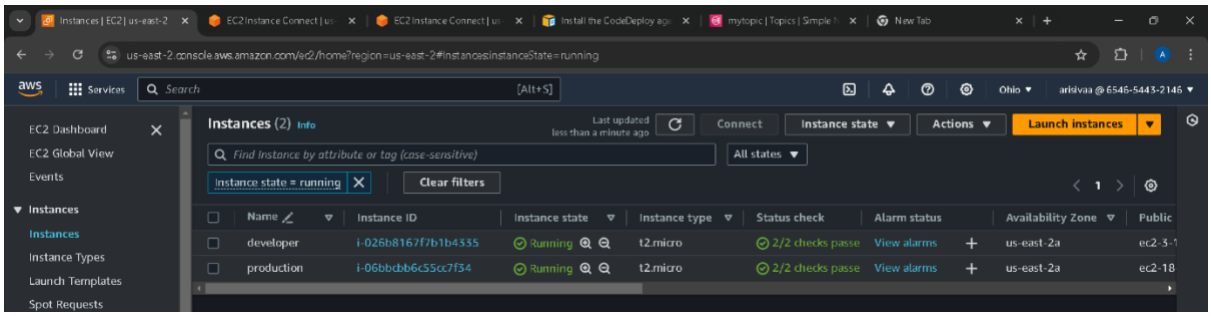
6) **Create a alarm name.**



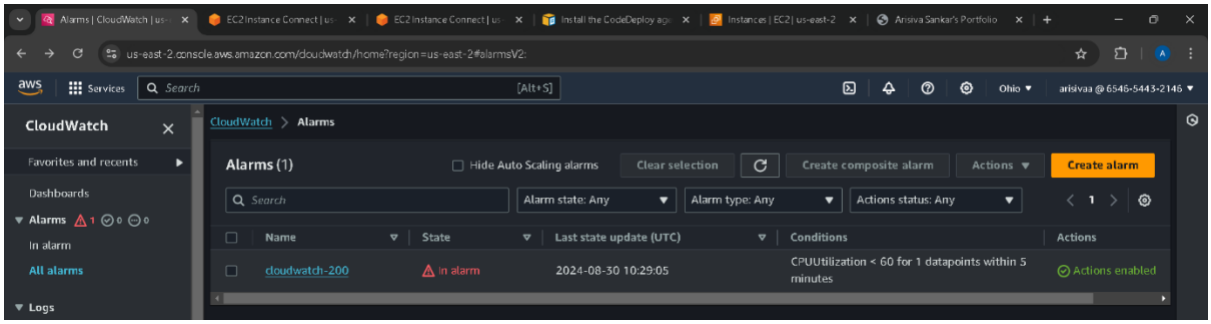
7) **Successfully create an alarm.**



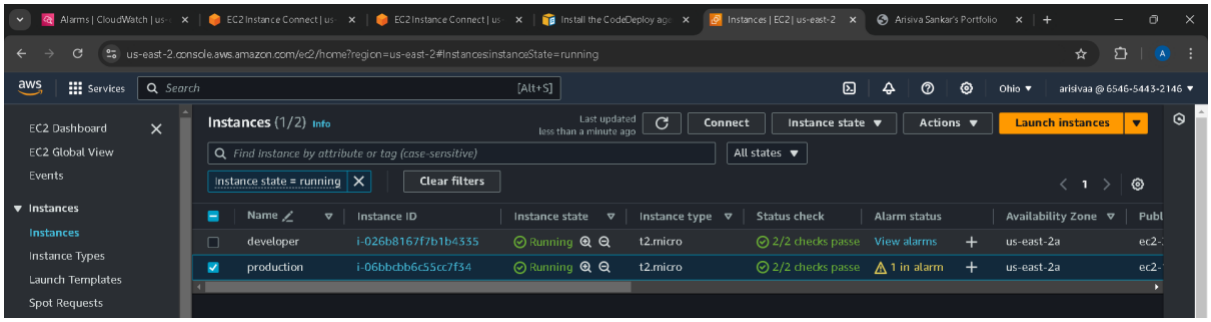
8) **Check instances 4 or 5 times reload the page and check the cloud watch.**



9) The check cloud watch in alarm status.

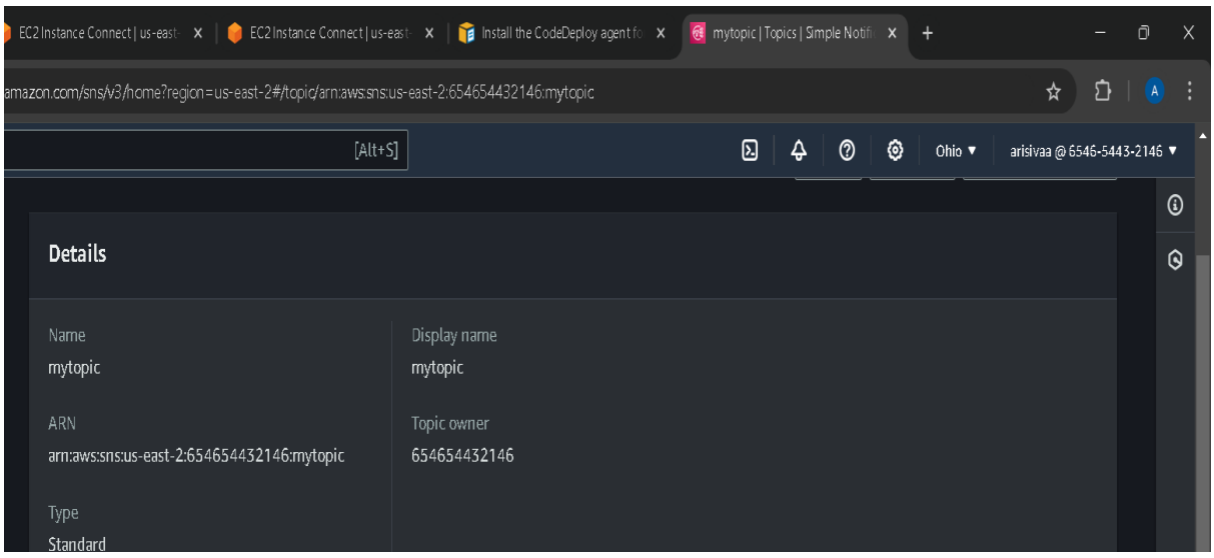


10) The next check is the EC2 instances.



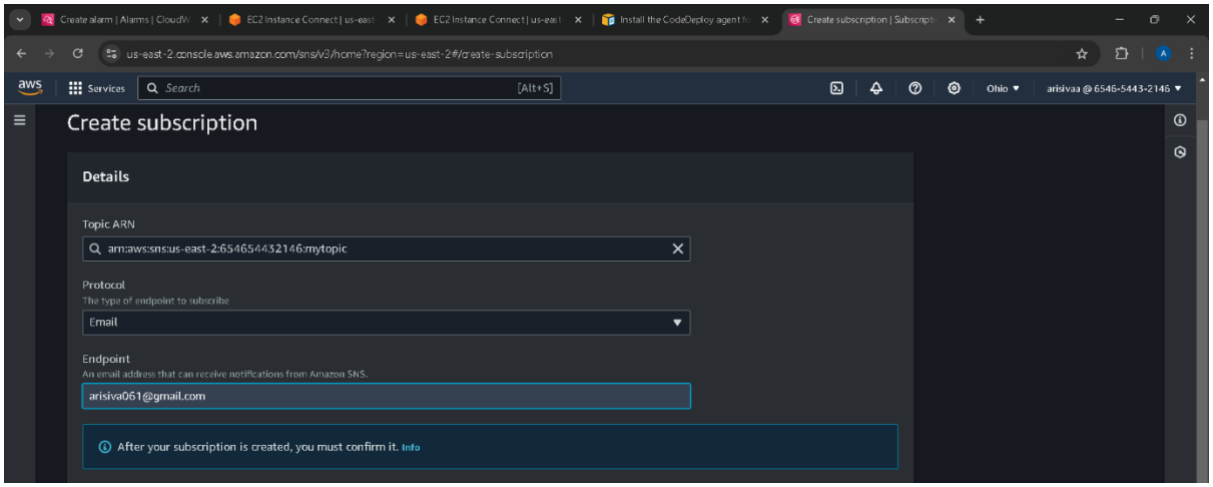
SNS (SIMPLE NOTIFICATION SERVICE)

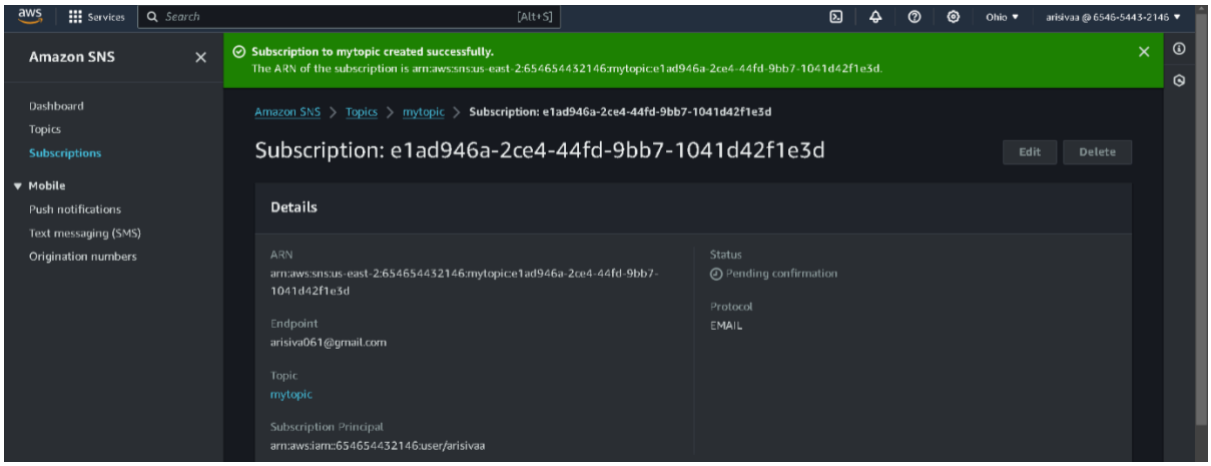
1) I have already created an SNS topic so I don't have any screen sorry.



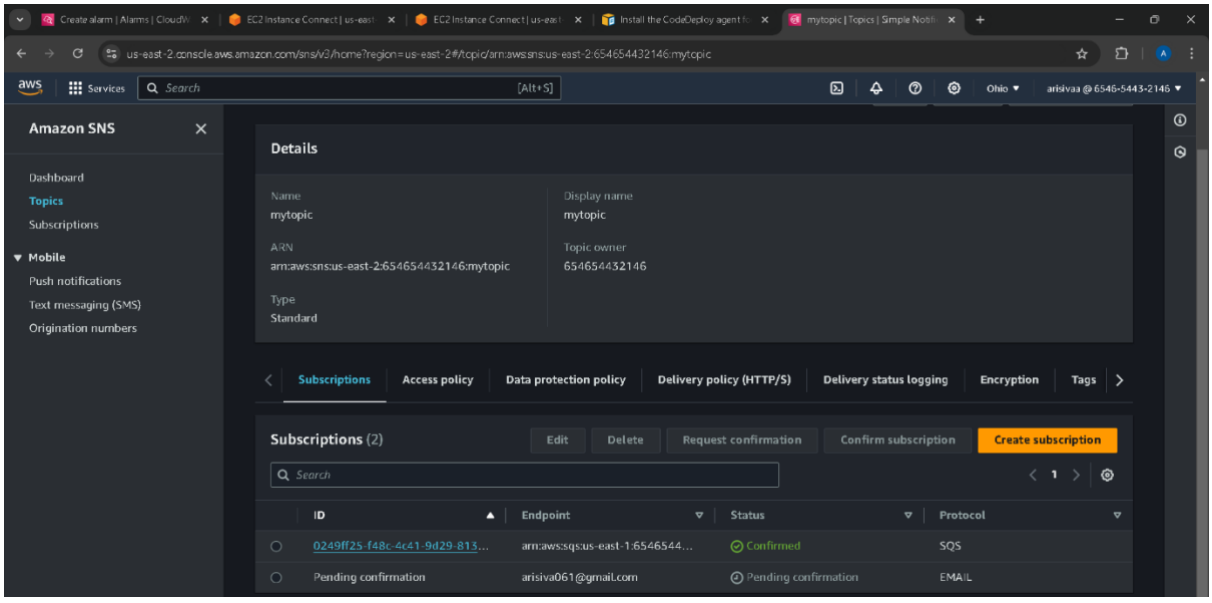
2) Next create a subscription first Email.

Select the topic ARN and protocol Email an d endpoint arisiva061@gmail.com

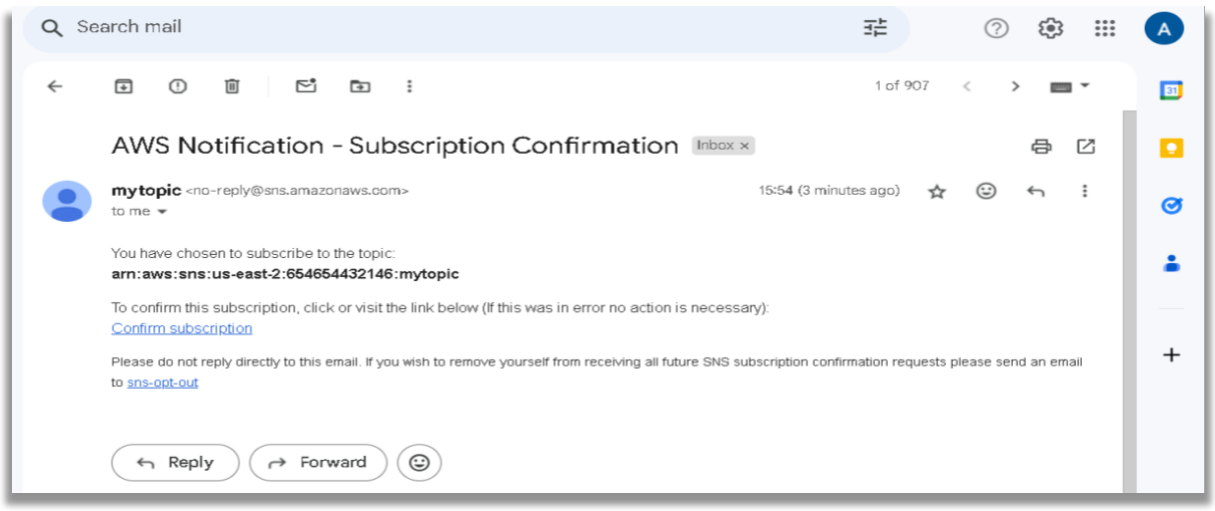




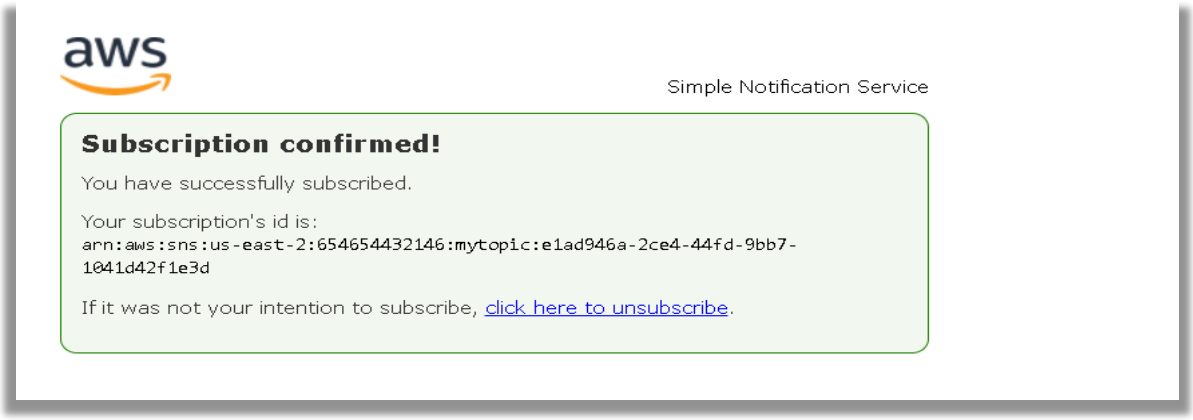
3) **Pending confirmation.**



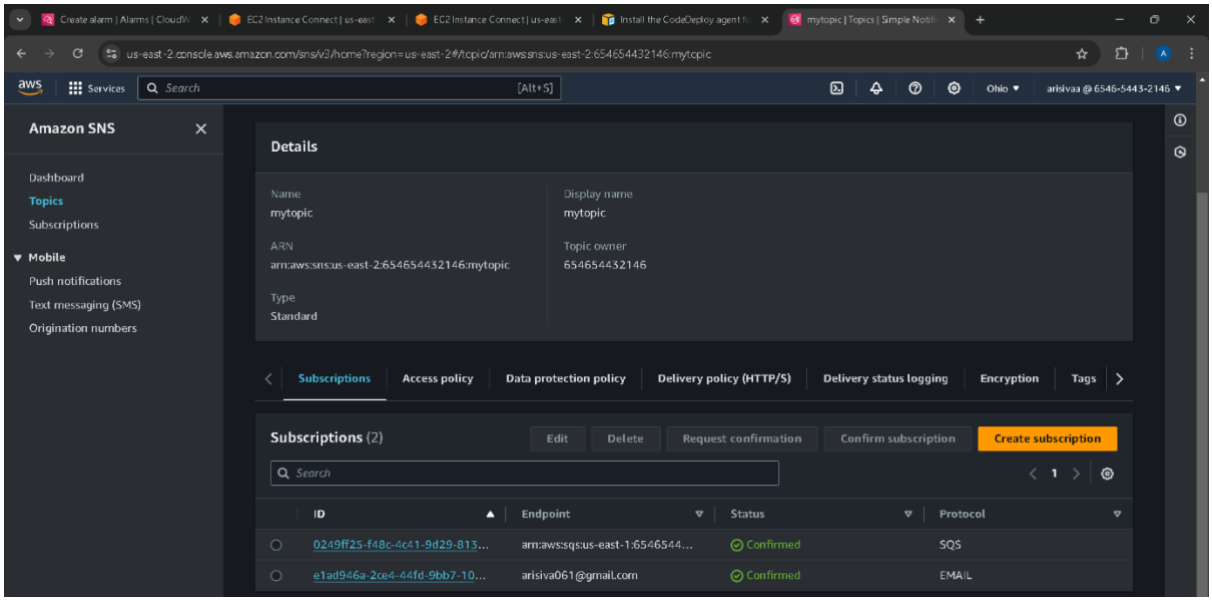
4) **Check the <u>Email AWS Notification</u> -Subscription Confirmation.**



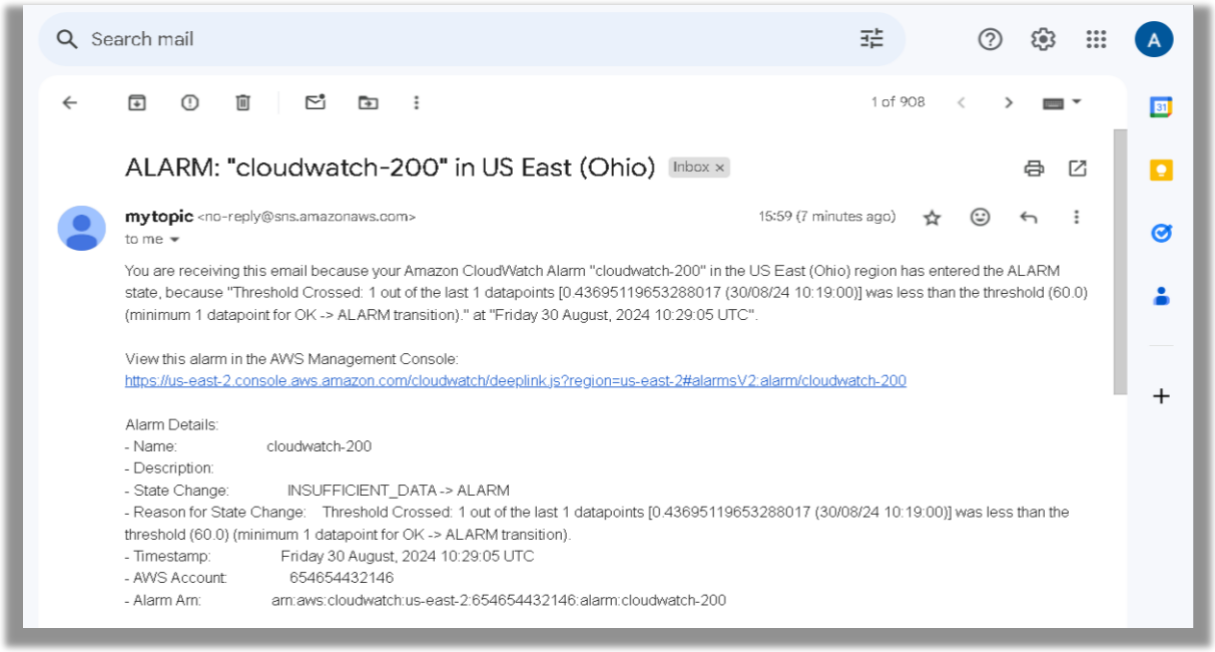
5) Subscription confirmed.



6) EMAIL confirmed.

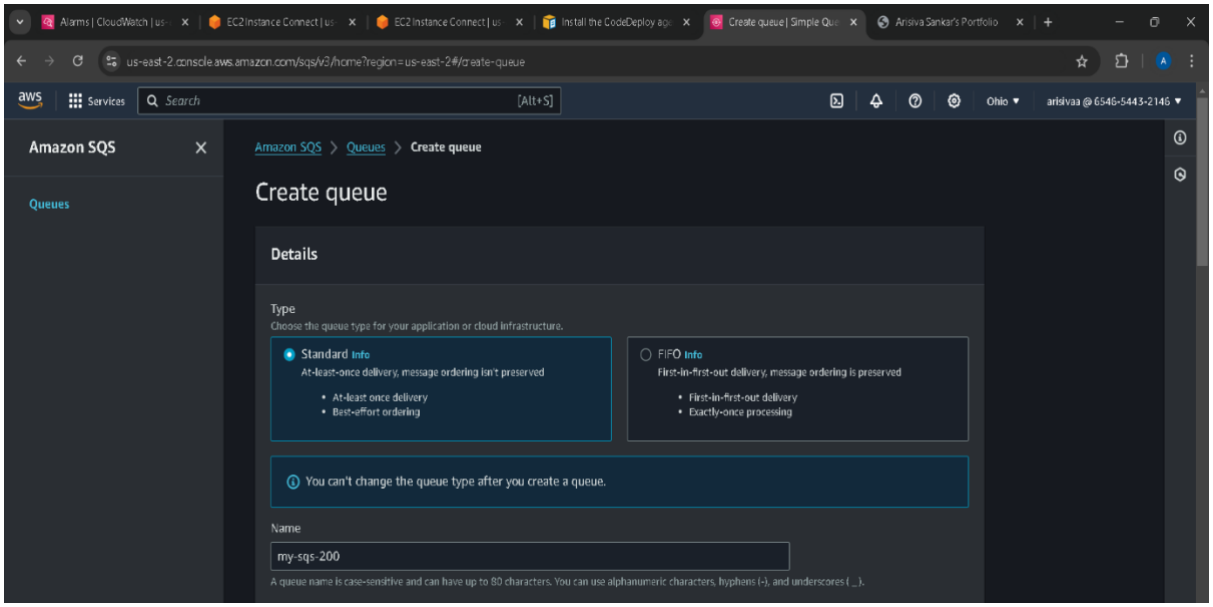


SNS EMAIL NOTIFICATION (CLOUDWATCH)

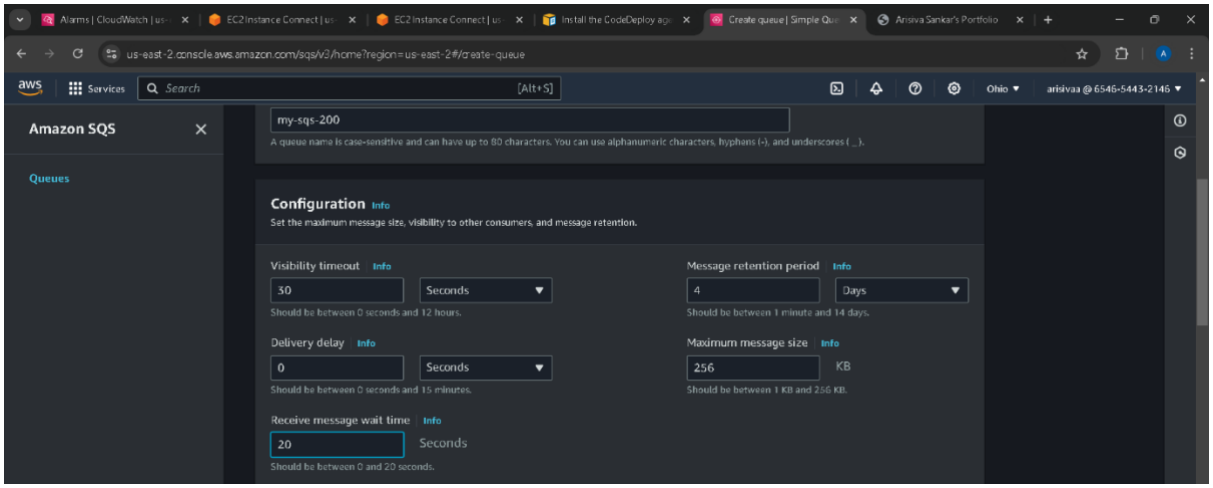


SQS (Simple Queue Service)

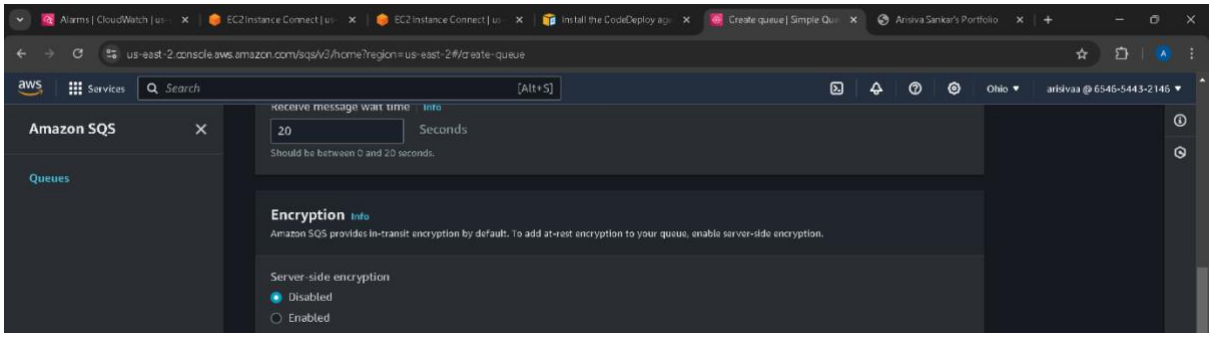
1) **Create a queue name**



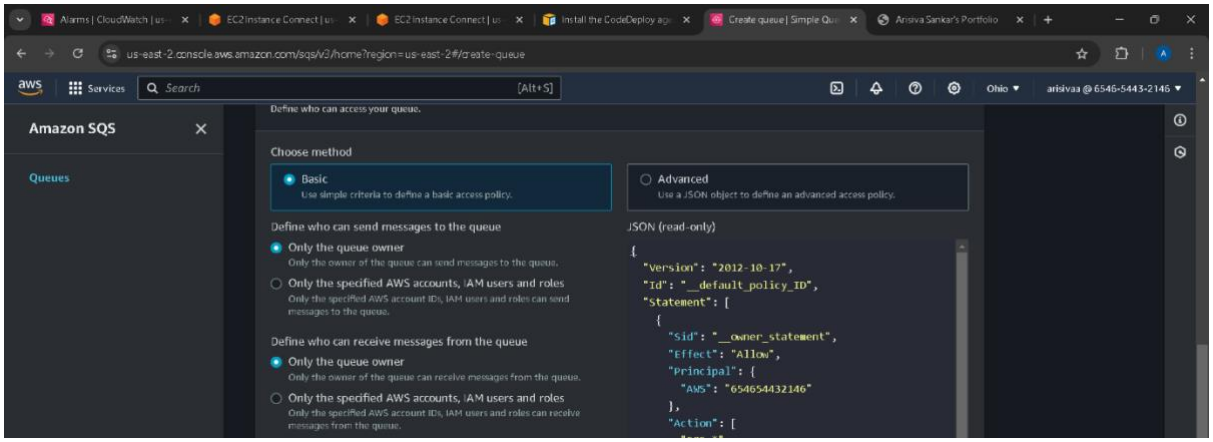
2) **Configuration time system**



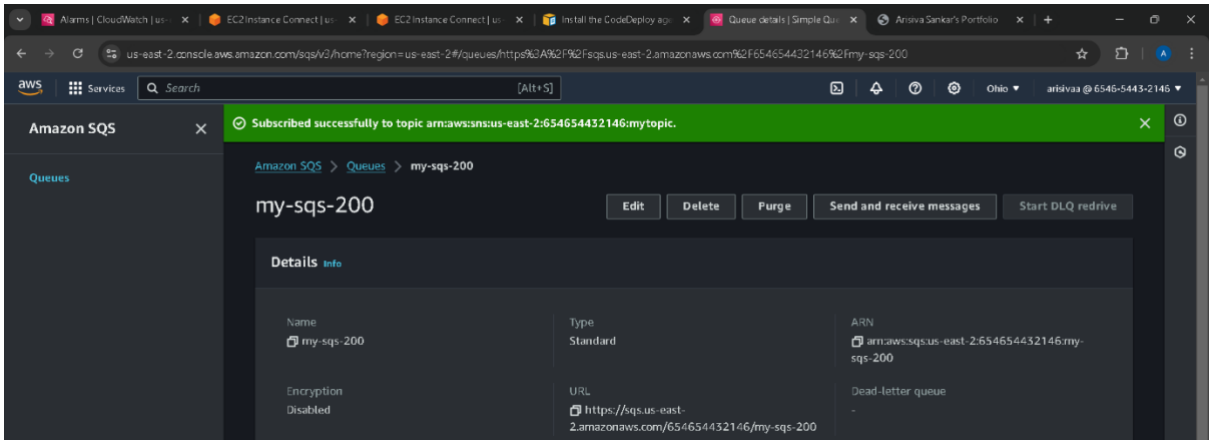
3) **Encryption.**



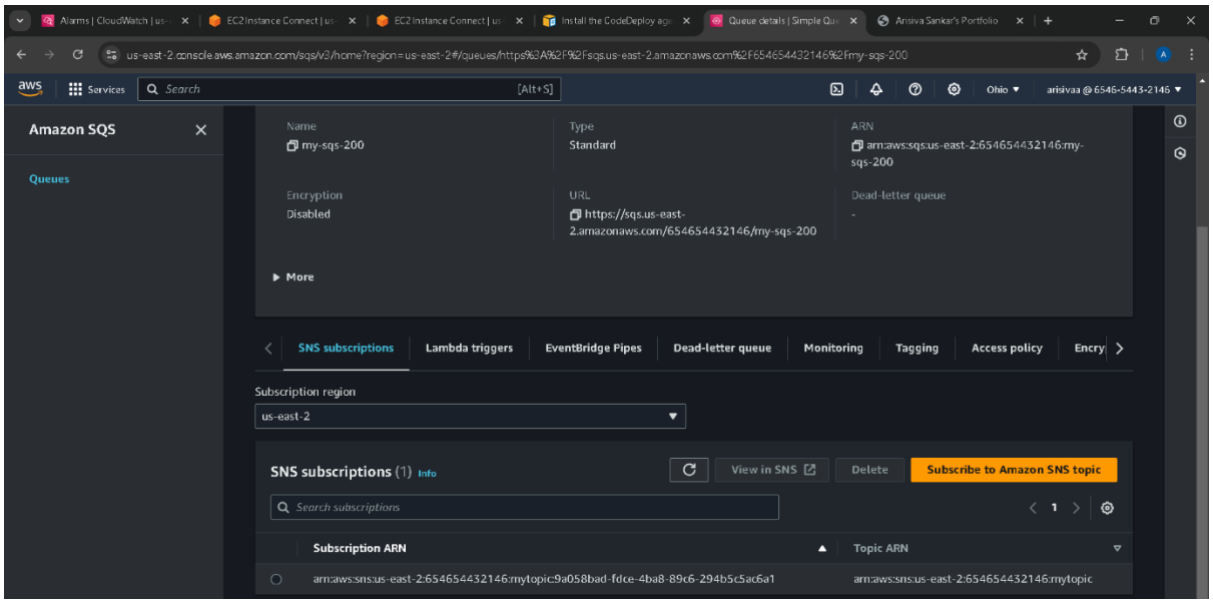
4) **Choose a method.**



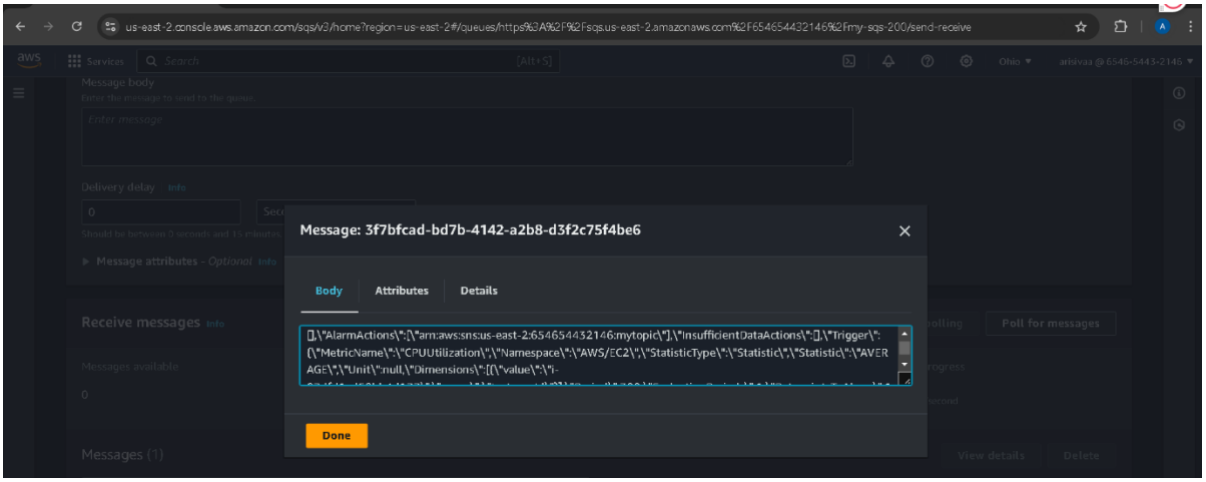
5) **Successfully SQS created.**



6) **SNS Subscription.**



7) **SQS Notification.**


Message: 3f7bfcad-bd7b-4142-a2b8-d3f2c75f4be6
Body
Attributes
Details
Done