Name: Shivam.R.Prajapati Div: D15C Roll No:41 Academic Year: 2024-25

Experiment No: 1 (A)

AIM: To develop a website and host it on local machine or virtual machine and Amazon S3 Bucket

STATIC HOSTING :

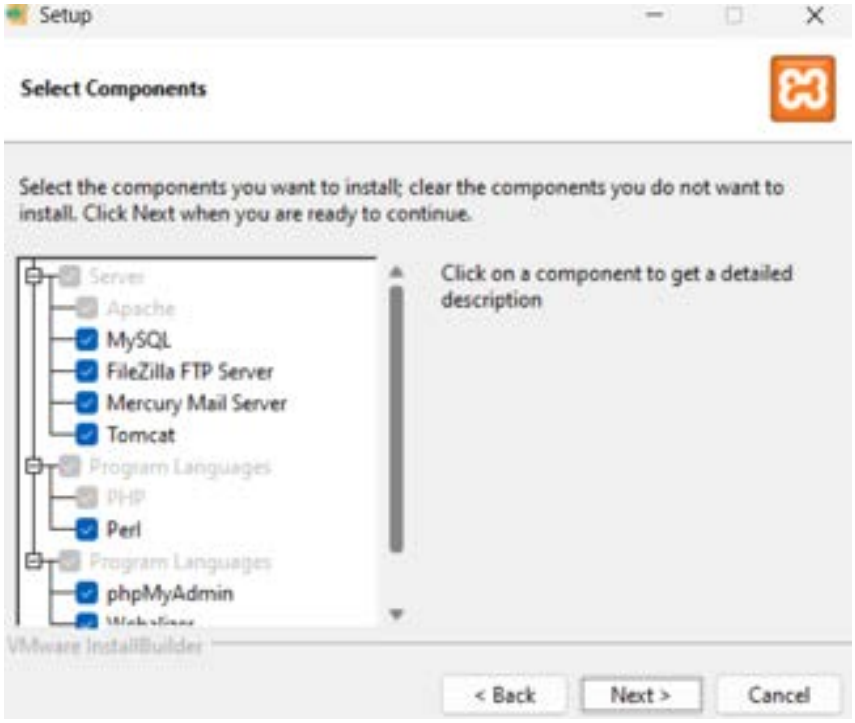
1) On local server (XAMPP)

Step 1: Install XAMPP from https://www.apachefriends.org/

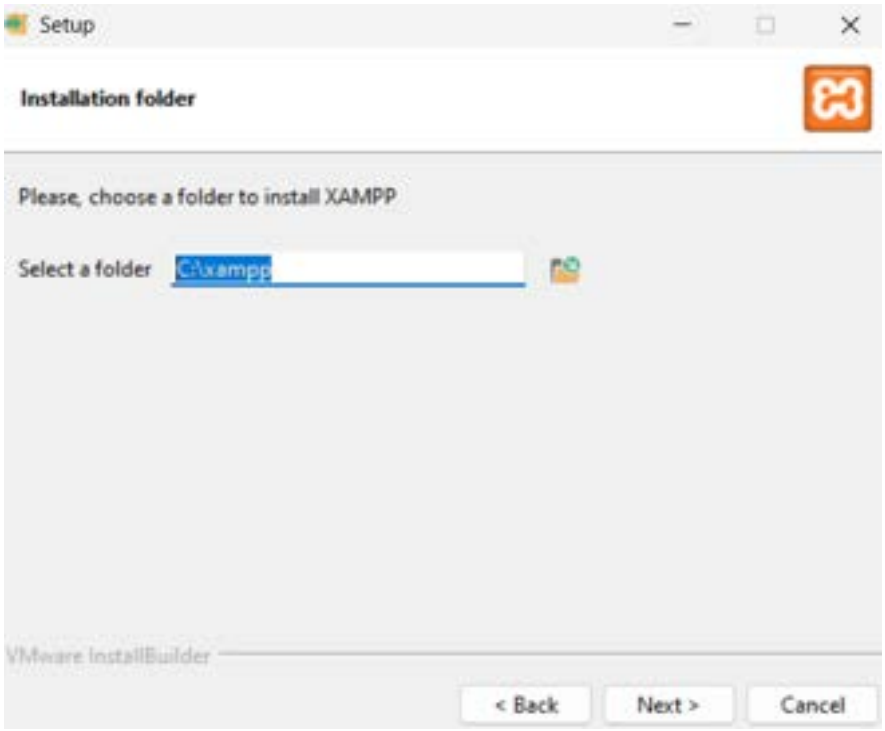
a) Select your OS. It will automatically start downloading

b) Open the setup file. Click on Next

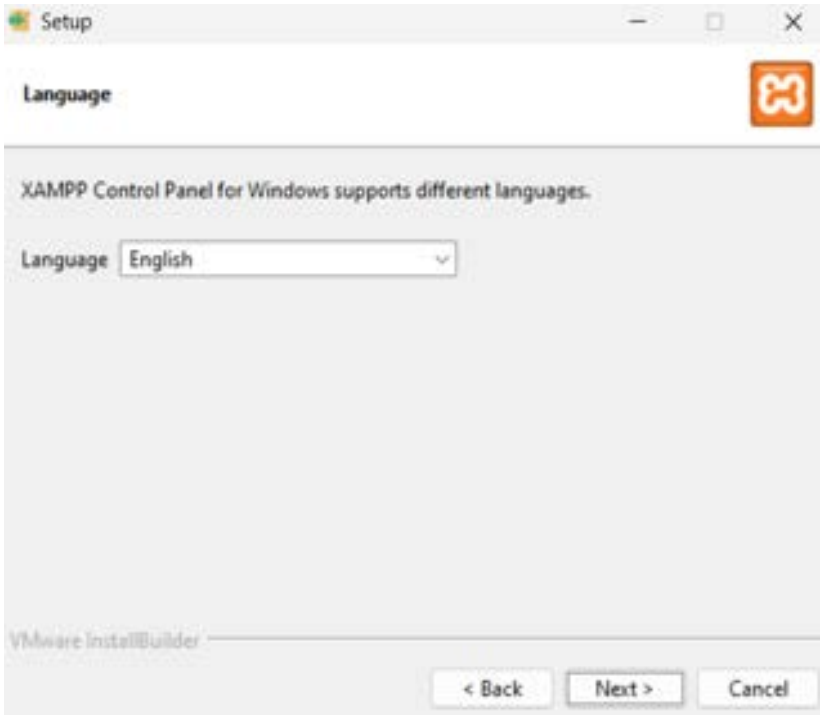
c) Select all the required components and then click on next



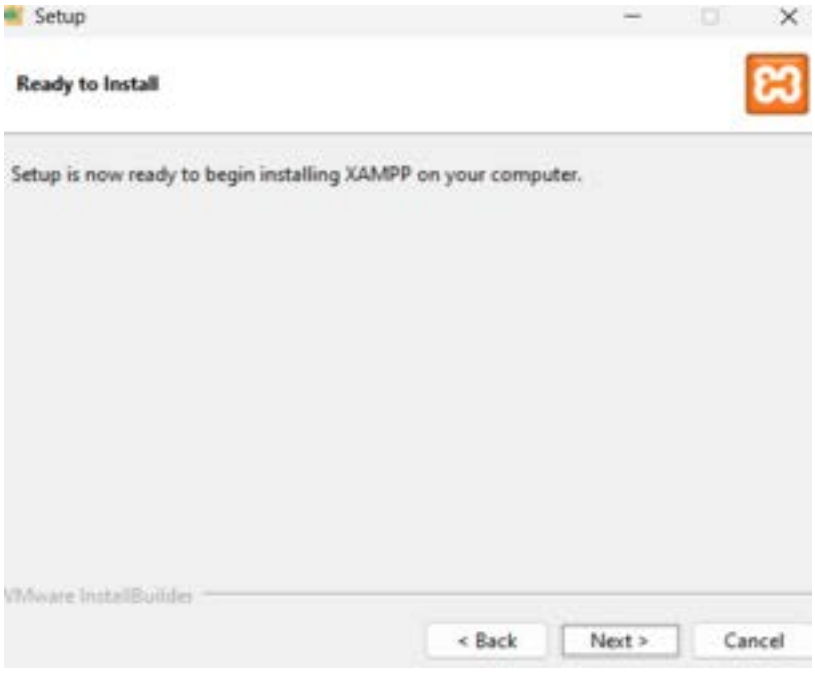
d) Choose the folder to install XAMPP in. Make sure the folder is empty. Click on next



e) Select the language, click next. XAMPP starts to install

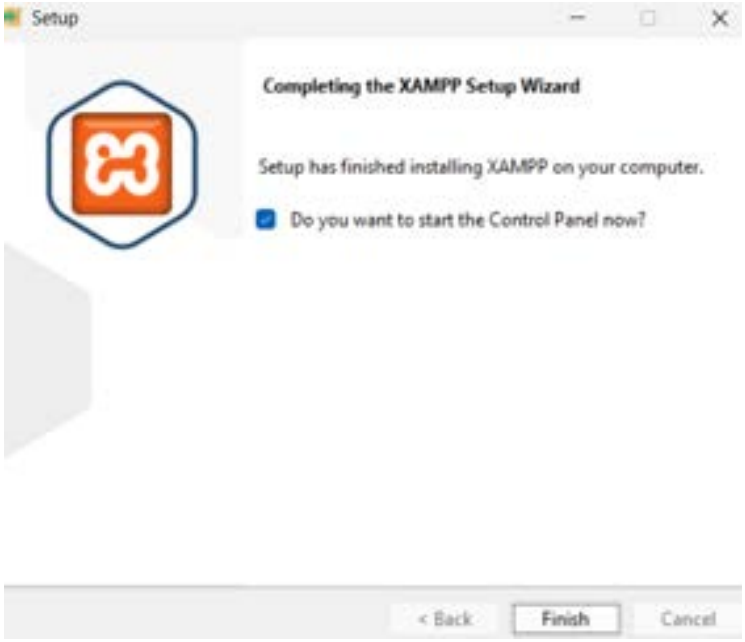


f) Now it will ask user for final confirmation for installing thus Click on Next



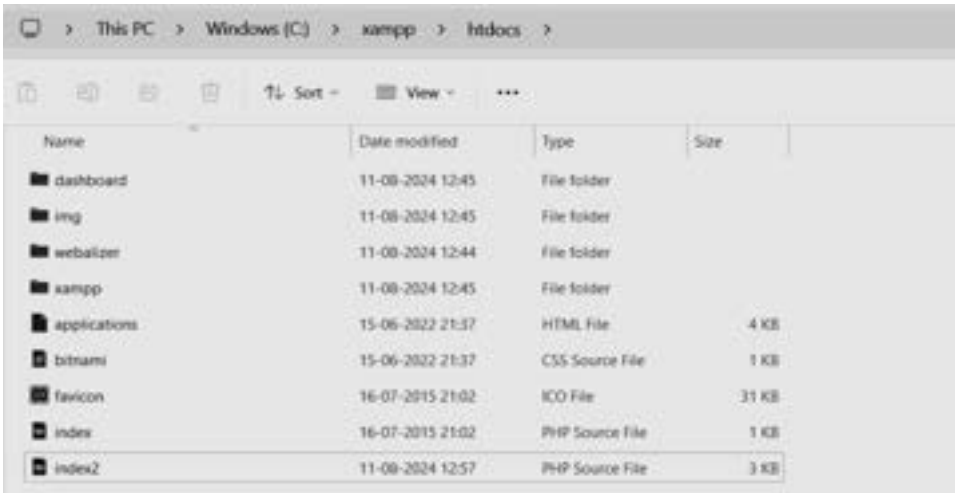
g) Wait until unpacking of all files is done

h) The installation is complete. Click on Finish



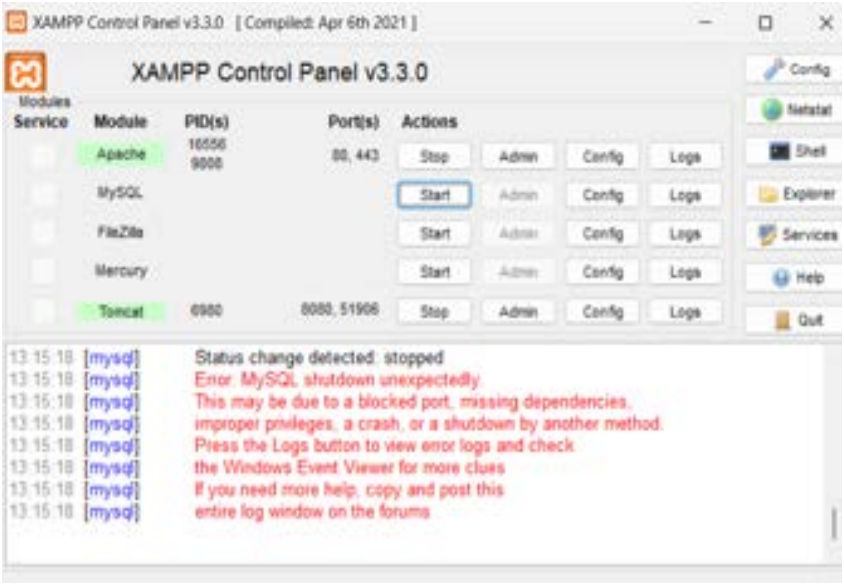
Step 2: Setup a file that is to be hosted on the server. Make sure the file has extension .php

Step 3: Go to the directory where XAMPP was installed. Go to htdocs folder. Place your folder in this directory i.e Paste the index2.php here



Step 4: Open XAMPP Control Panel, start the Apache service (Required) and mySQL service (if needed)

Click on Start button corresponding to Apache

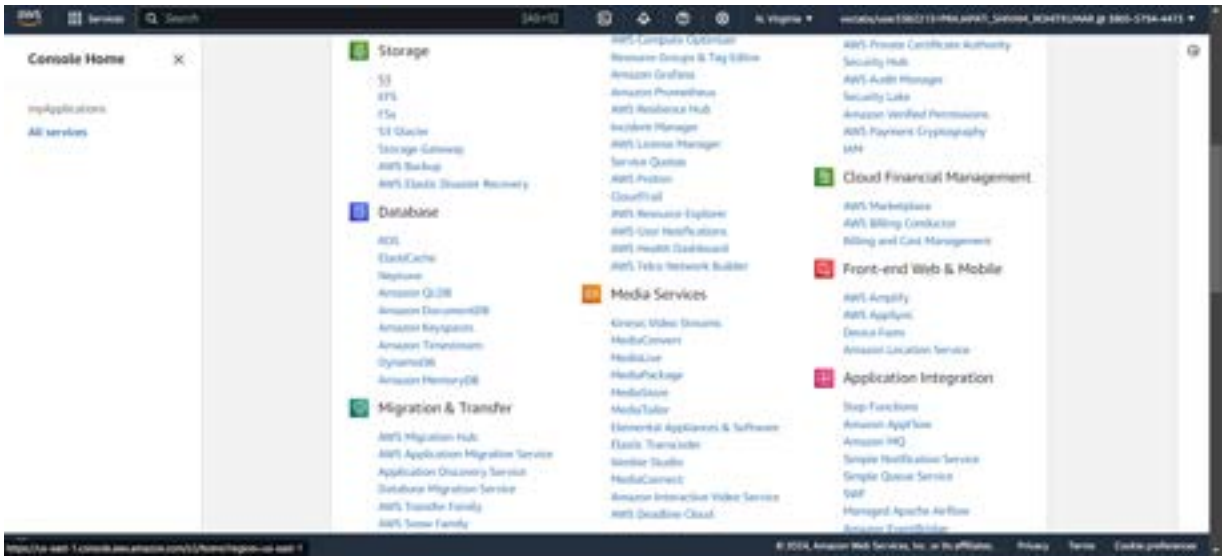


Step 5: Open your web browser. Type localhost/index2.php in the browser . This will open your website on your browser.

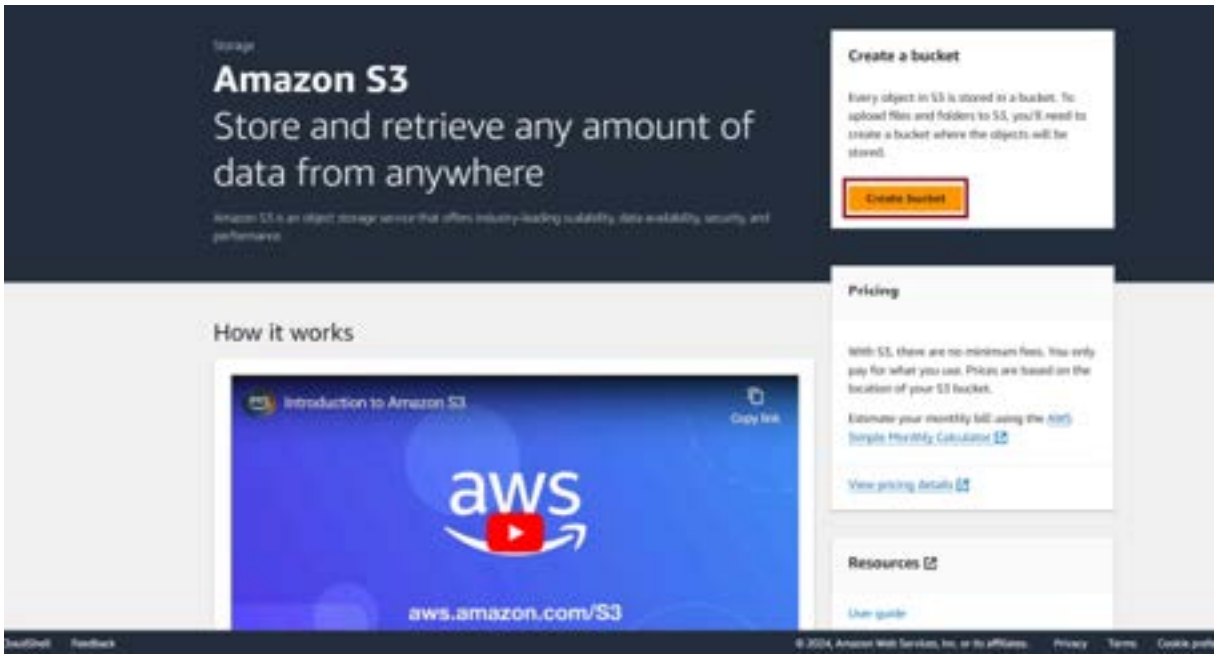
Name: Shivam.R.Prajapati Div: D15C Roll No:41 Academic Year: 2024-25

2) On AWS S3

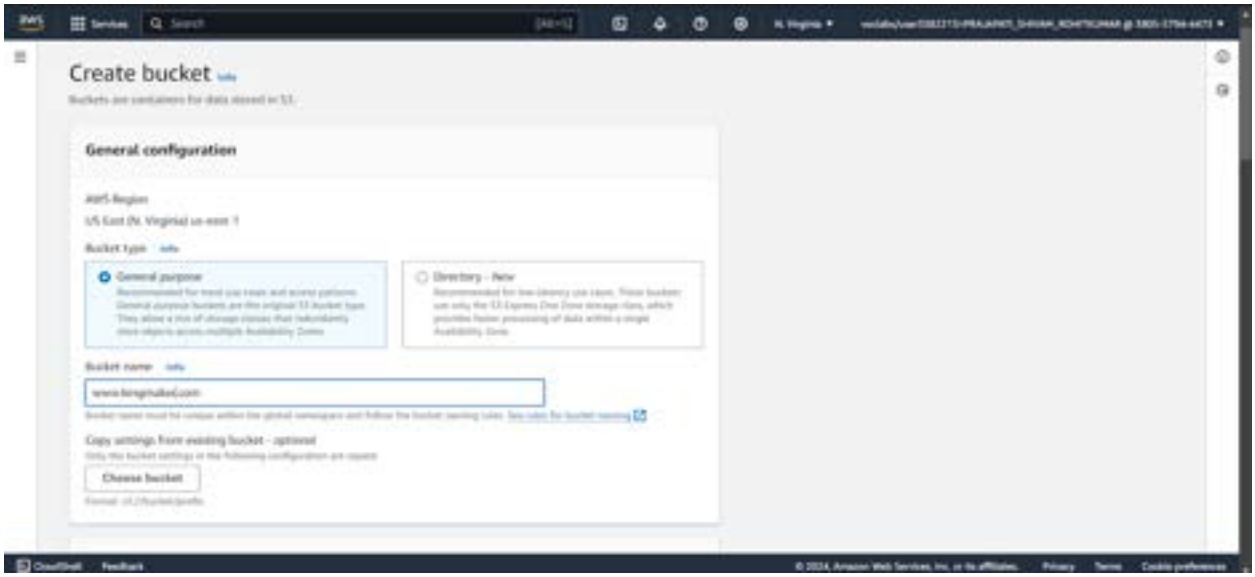
Step 1: Login to your AWS account. Go to services and open S3.

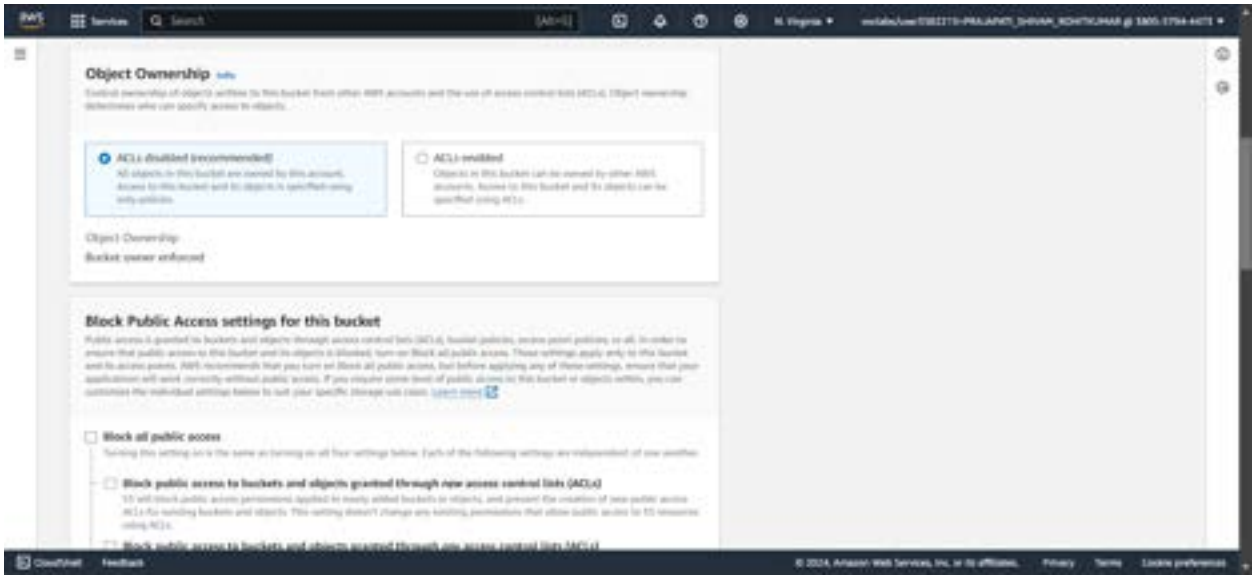


Step 2: Click on Create Bucket

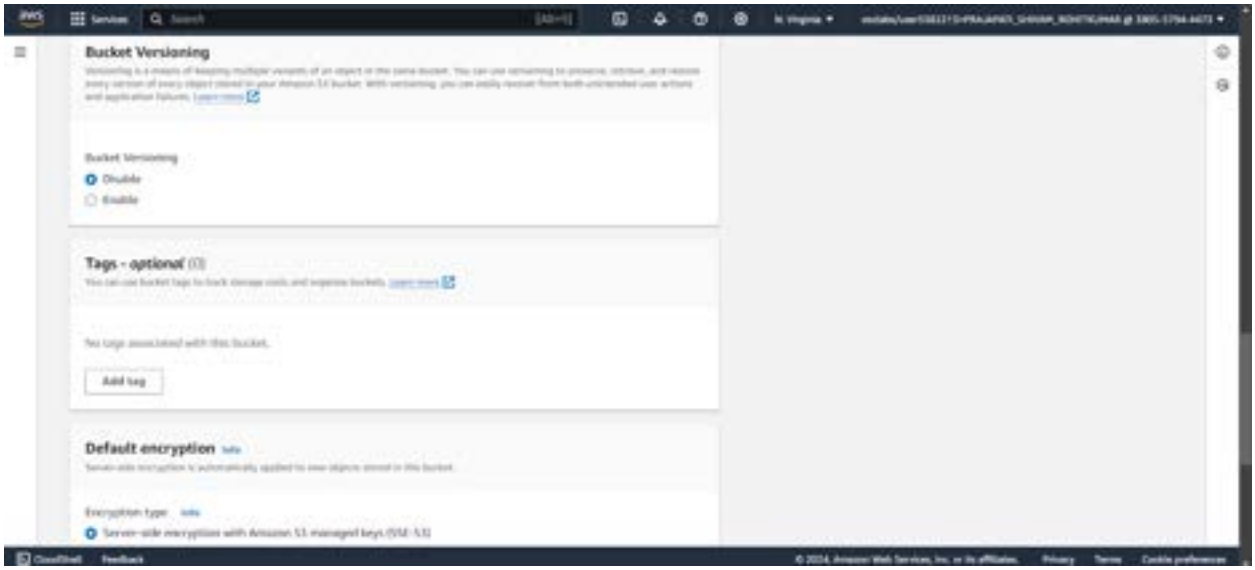


Step 3: Give a name to your bucket, keeping other options default like bucket type as general purpose , scroll down and click on Create Bucket

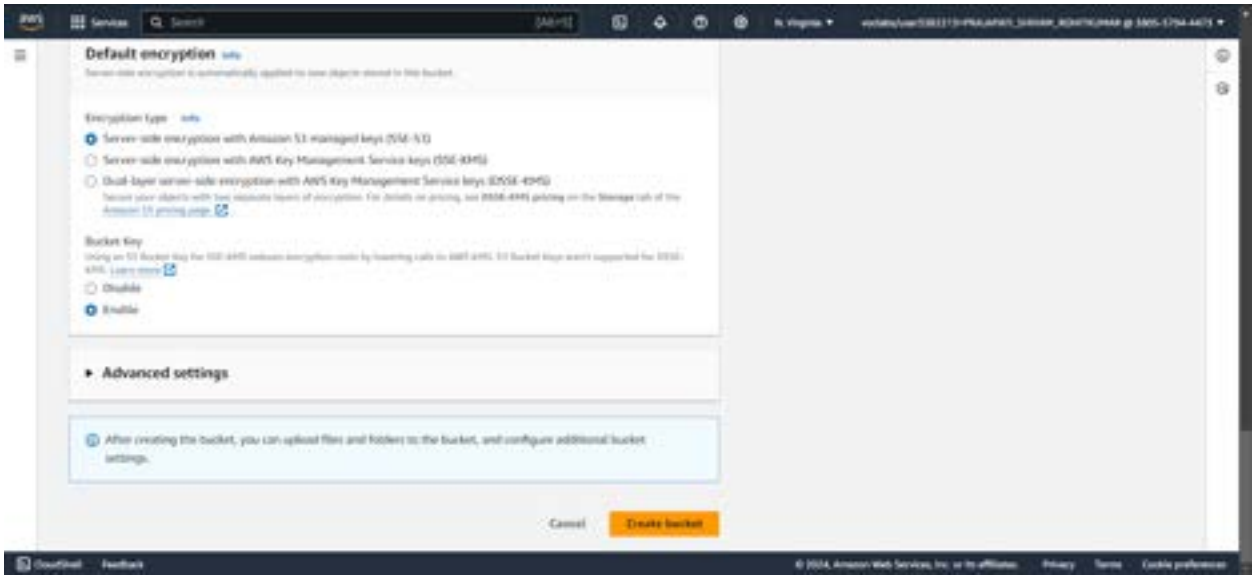




Uncheck the box under Block Public Access settings if it is checked because we want bucket to be publicly accessed

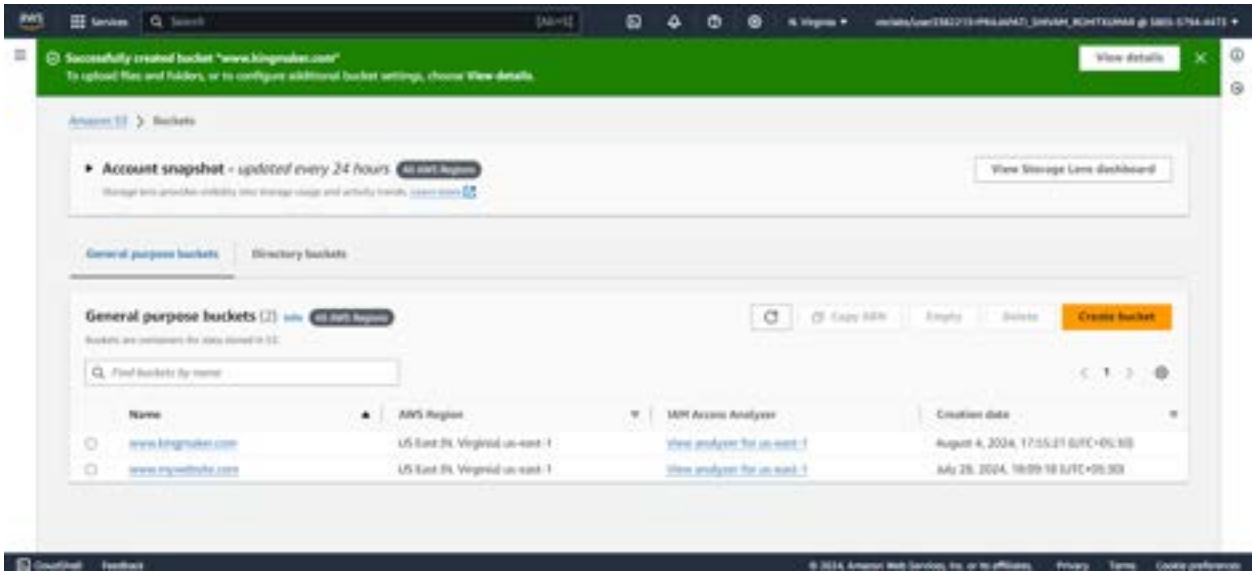


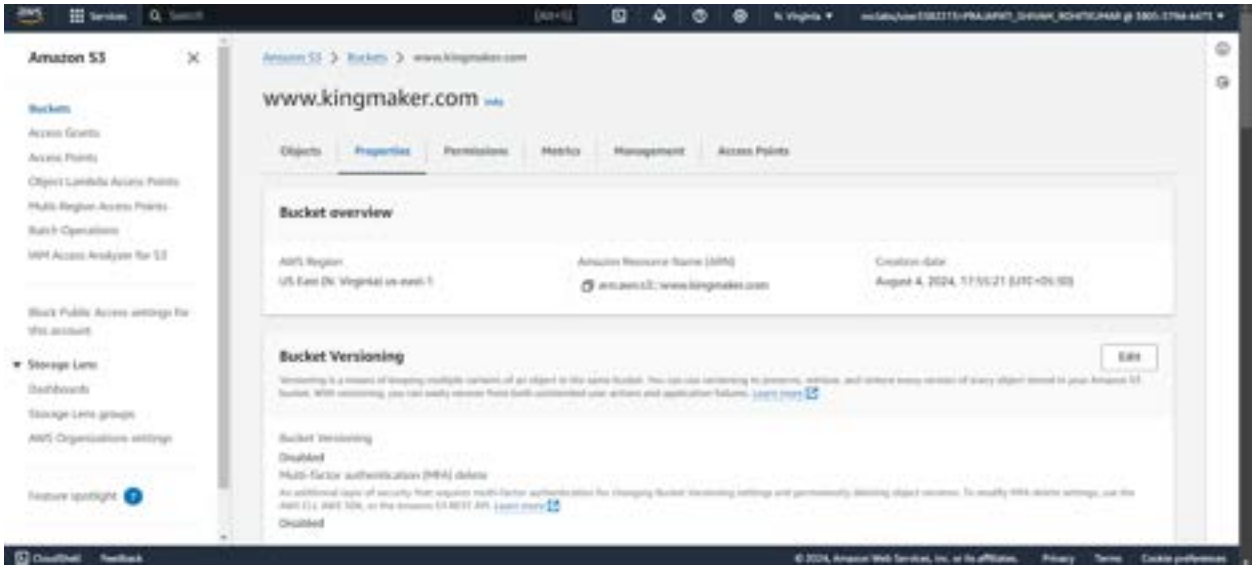
Keep the Bucket Versioning disabled as we dont want variants of an object to the same bucket.



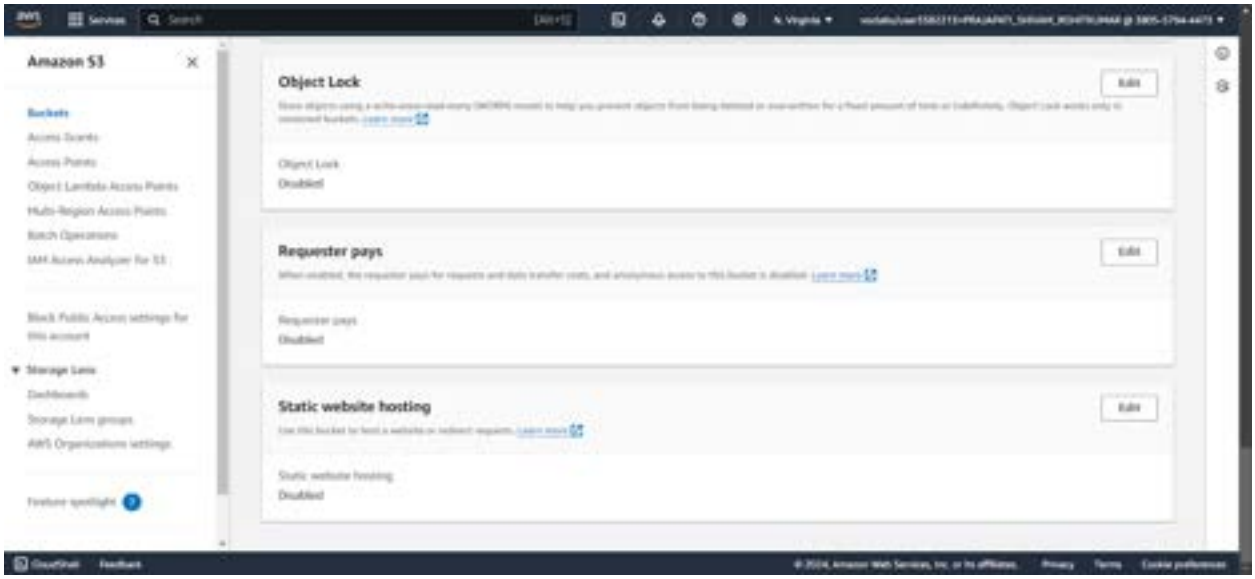
Click on Create Bucket

Step 4: Click on the name of your bucket and goto Properties.

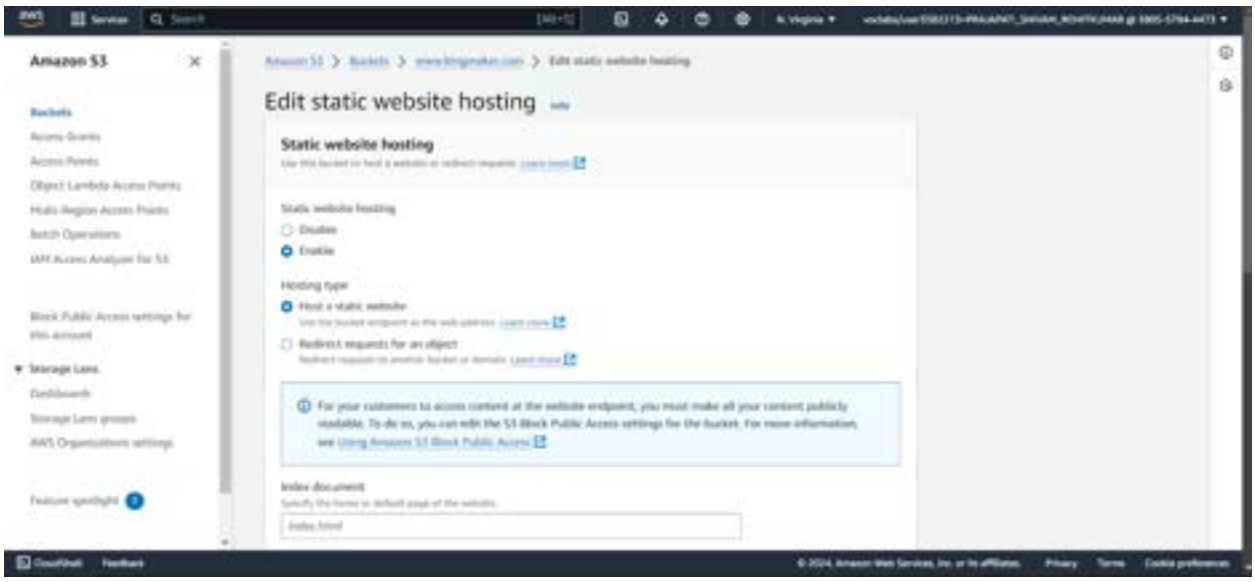




Step 5: Scroll down till you find Static website hosting, click on edit

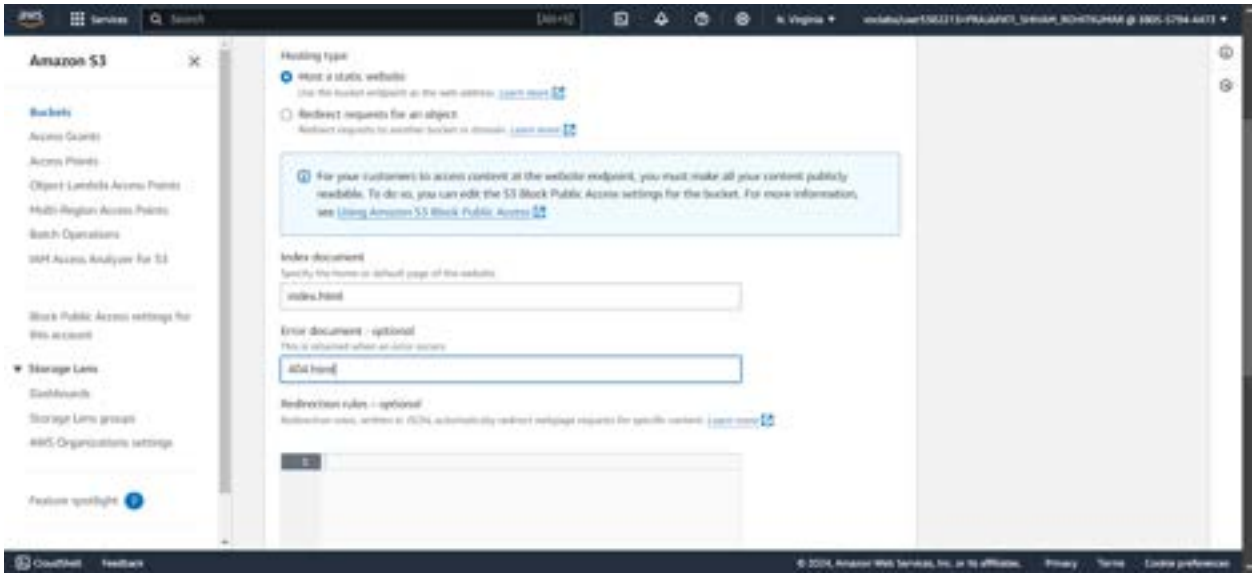


Step 6: Click on Enable static website hosting because we want to host our static website to the bucket

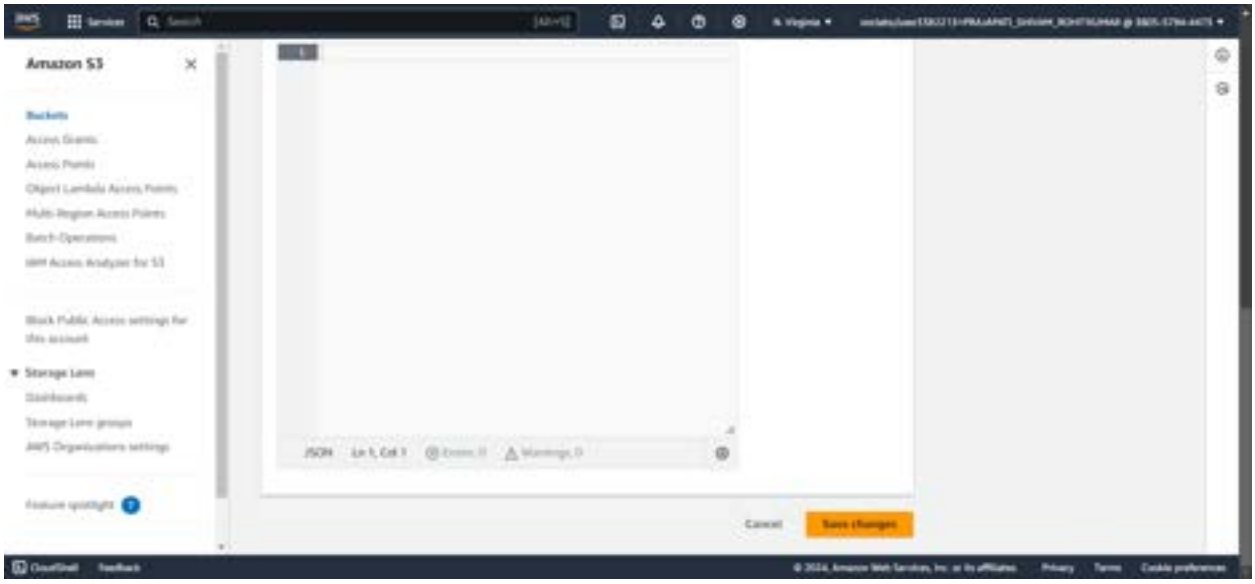


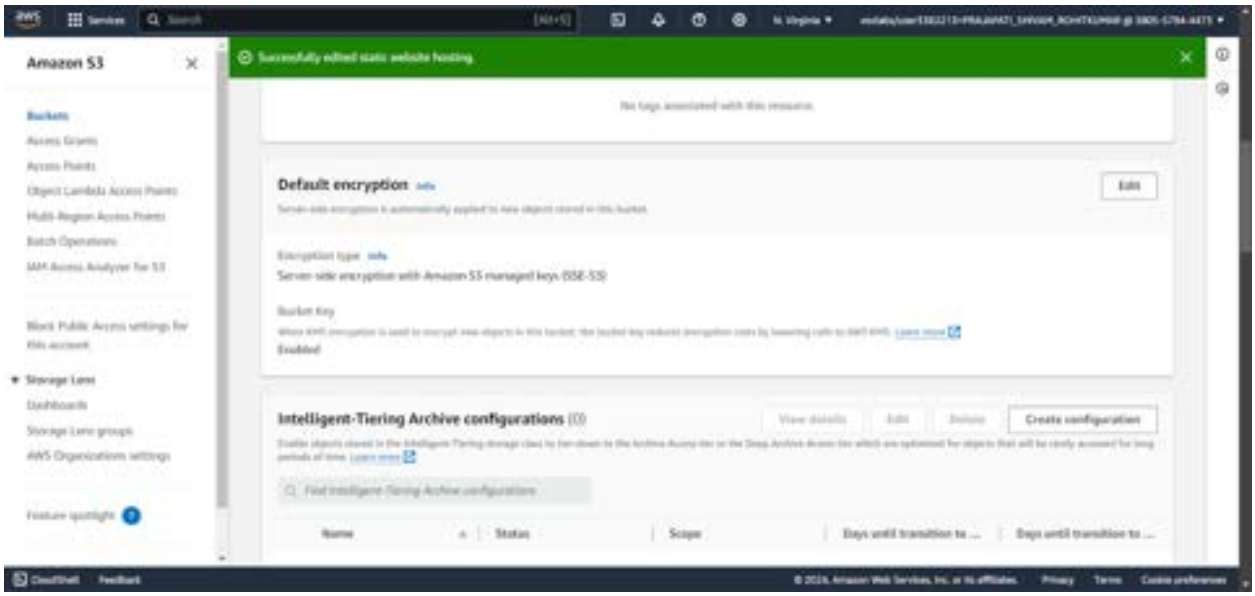
Name: Shivam.R.Prajapati Div: D15C Roll No:41 Academic Year: 2024-25

Step 7: Write the name of your document which you wanted to host on AWS from your local folder and in error document, give name as 404.html. Save your changes.

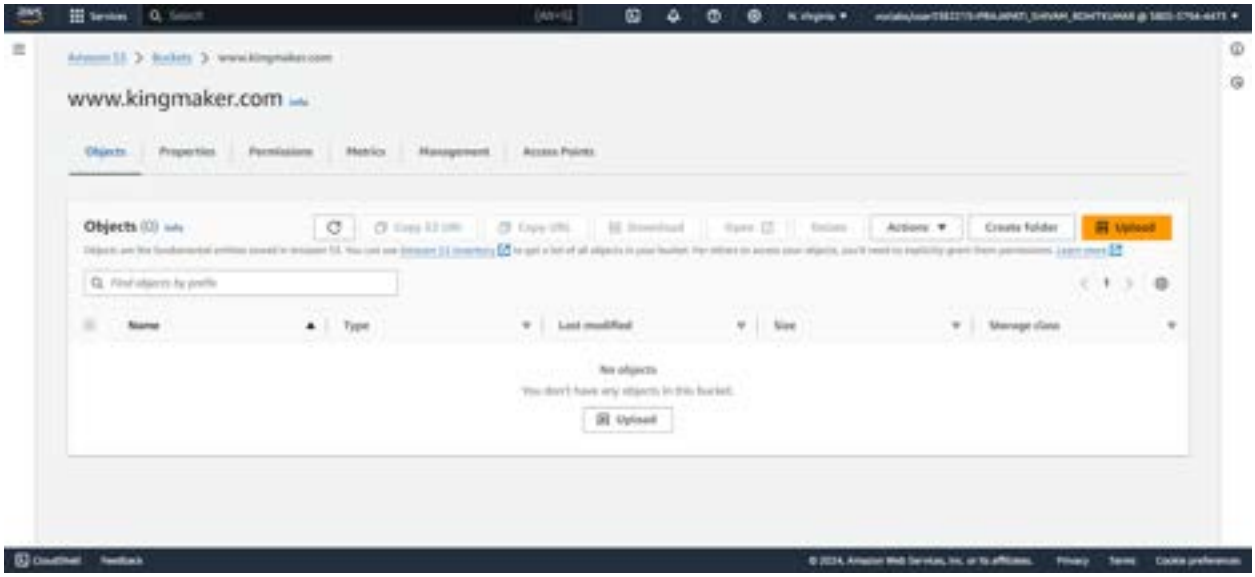


Step 8: Click on Save Changes

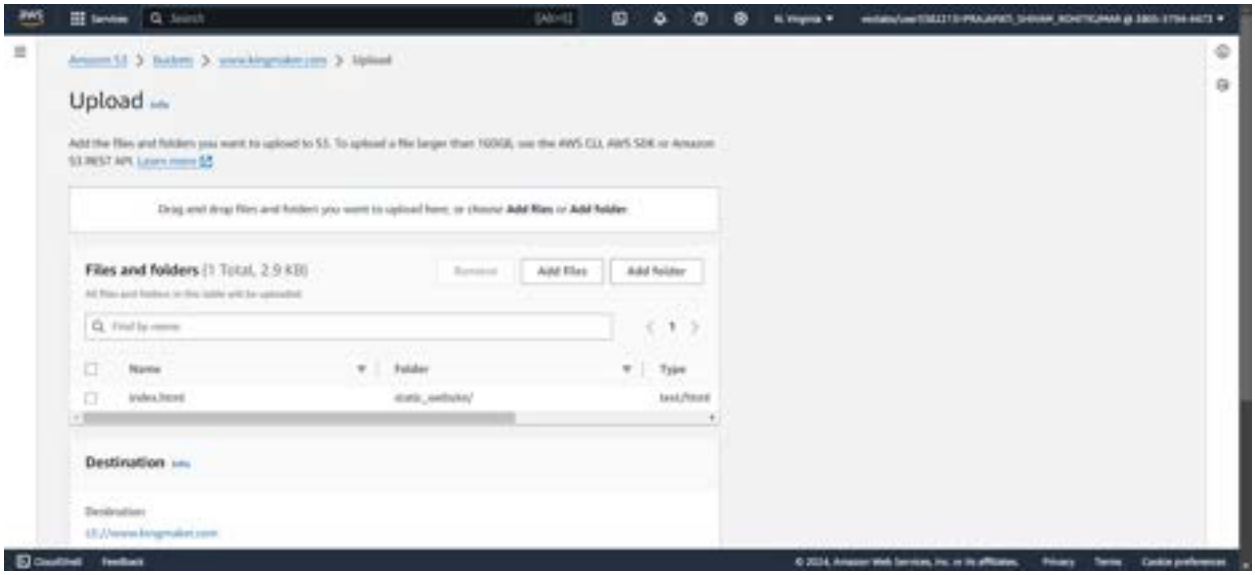


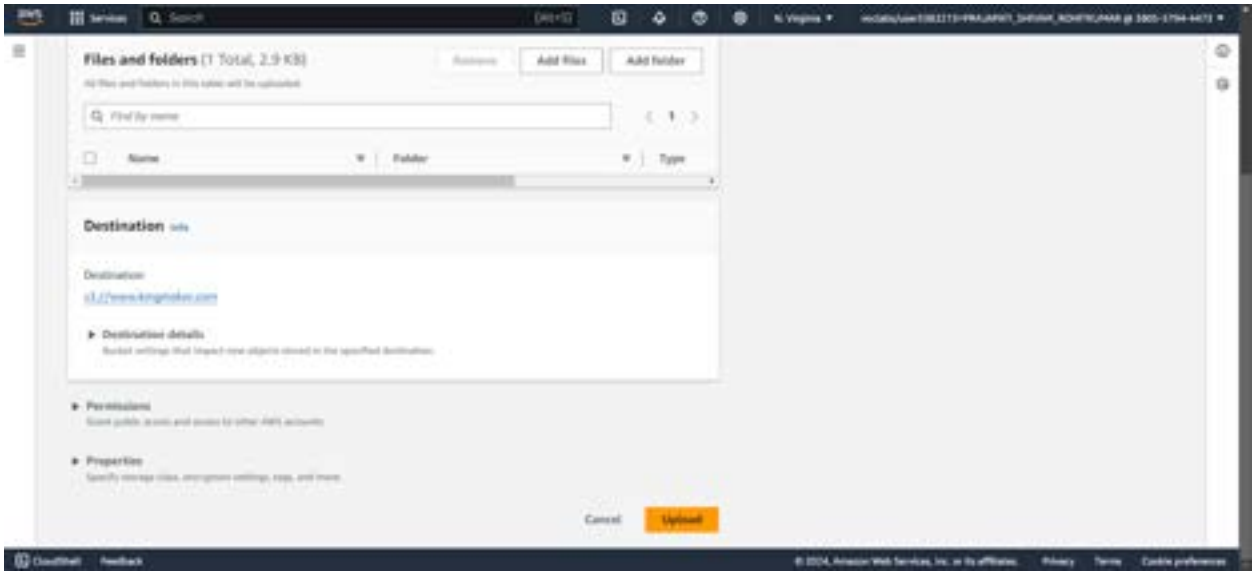


Step 9: Go to Objects tab and click on upload file to upload file to your bucket

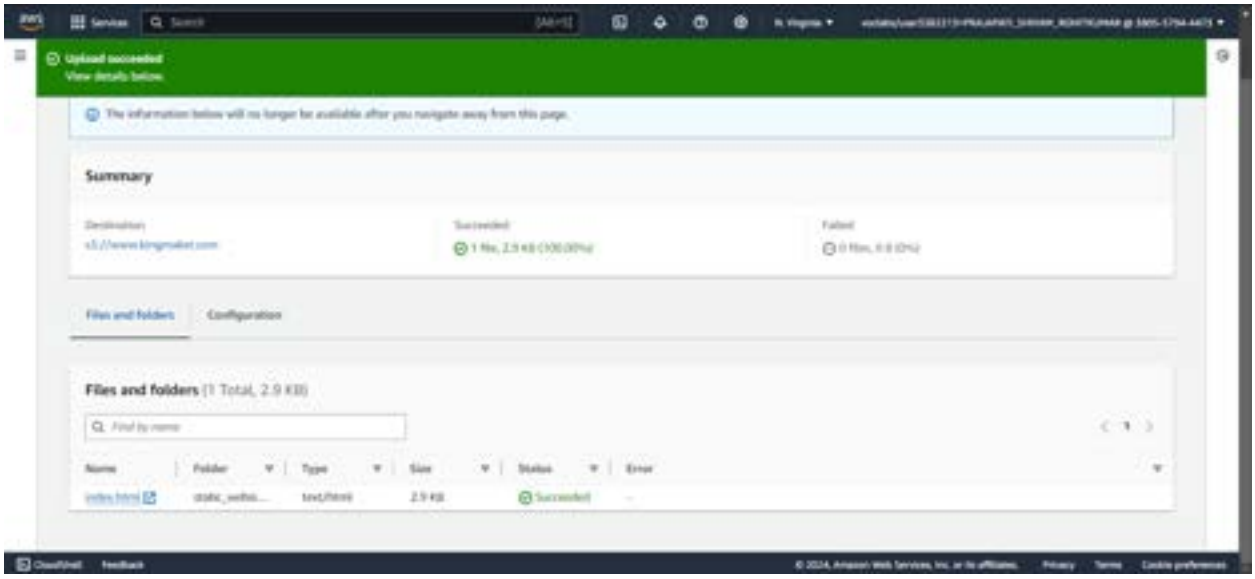


Step 10: Click on Add files. Add all the files you want to upload. Then scroll down and click on Upload





Click on Upload



Step11: This will take you to the Objects screen. Switch to Properties, scroll down to Static web hosting. There you would find the link (Bucket website endpoint) to your website.

Step12: Open the link. It will show a 403 forbidden error screen as the contents of the bucket are not available for the public users. To change this, go to Permissions tab, go to Block public access and click on edit

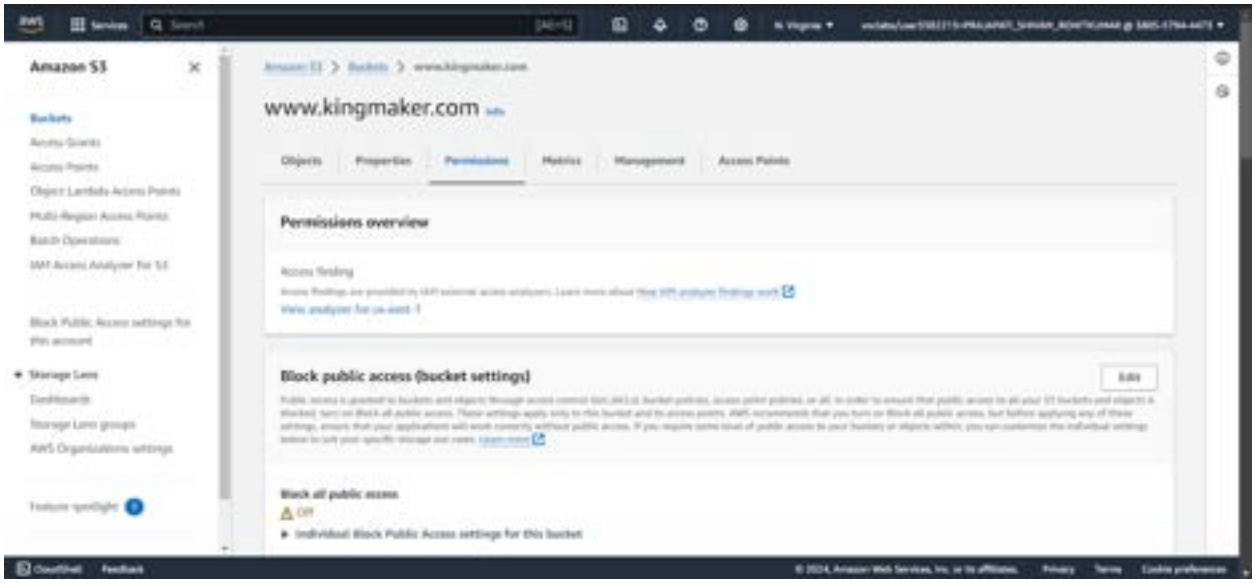
403 Forbidden

- Code: AccessDenied
- Message: Access Denied
- RequestId: 8TQ4EGP4TK06MVPB
- HostId: hF+ToadQUoCuDM8H+iFRsXdA28TGp+xikYbjb4CICS/t+3it4ihA/tvgA1Xr1xo+JL5AhkT6hJs=

An Error Occurred While Attempting to Retrieve a Custom Error Document

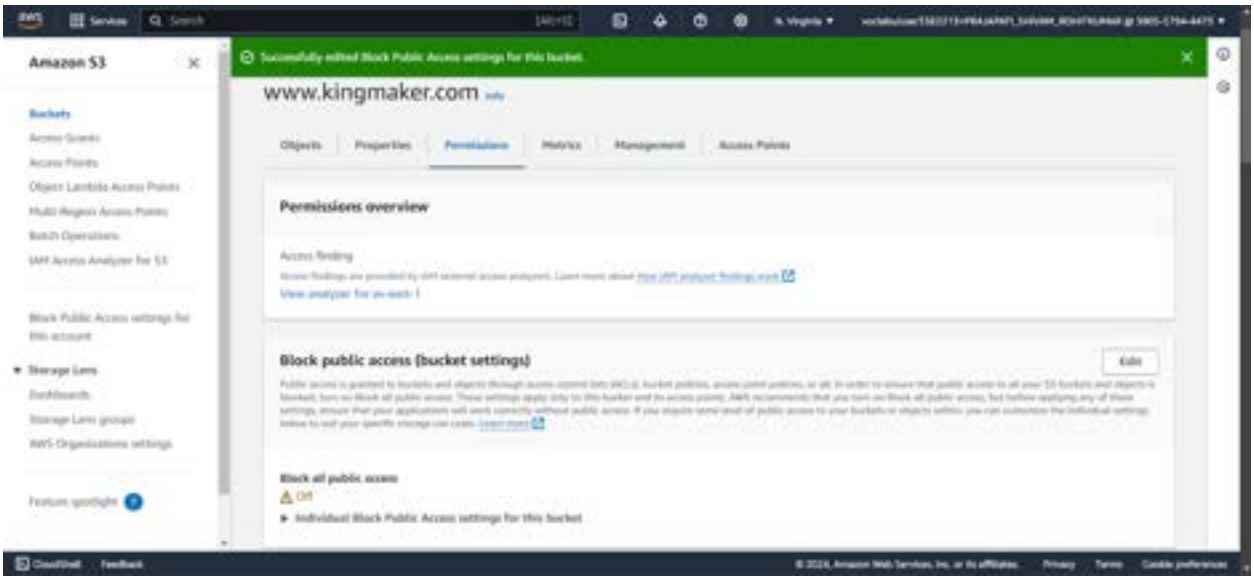
- Code: AccessDenied
- Message: Access Denied

Step 13: Uncheck the Block all public access checkbox and click on save changes



Type confirm in the given field and click on confirm button as shown above

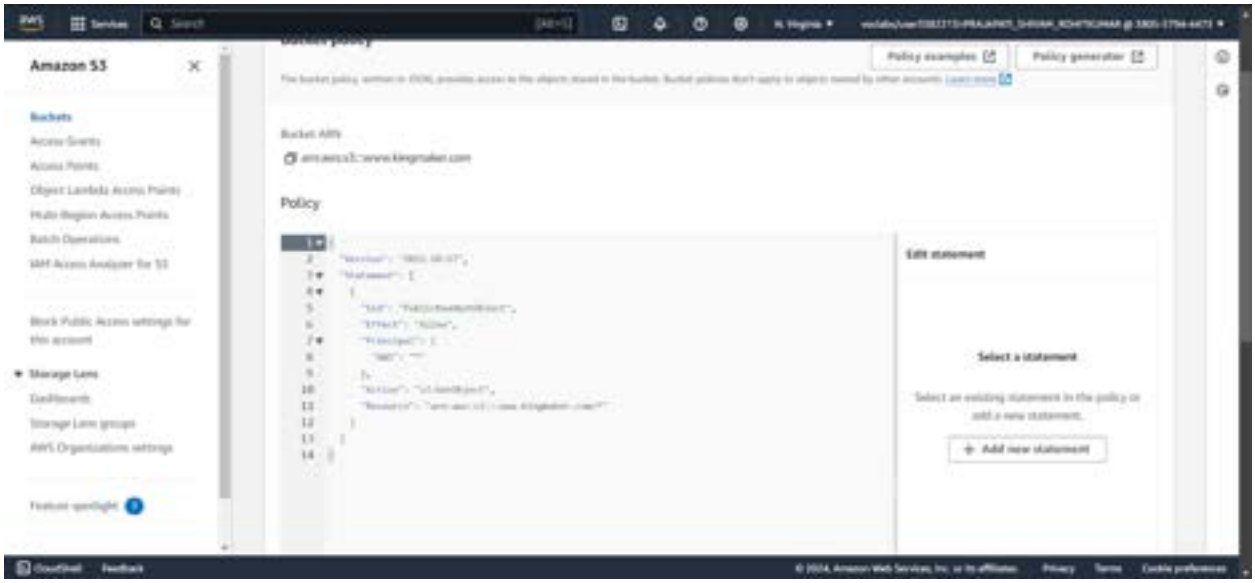
Step 14:Successfully Changed the Settings



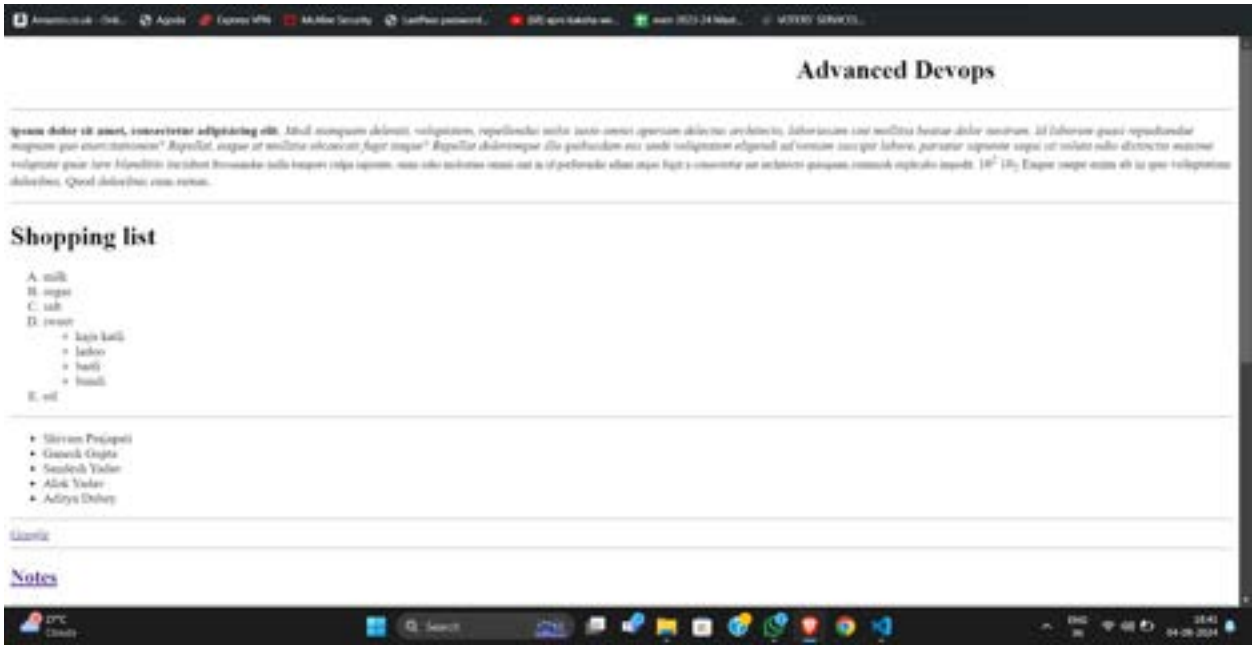
Step 15: Scroll down to bucket policy and click edit and paste the code from given Github Link https://gist.github.com/Savjee/b4b3a21d143a30e7dc07

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "PublicReadGetObject",
   "Effect": "Allow",
   "Principal": {
    "AWS": "*"
   },
   "Action": "s3:GetObject",
   "Resource": "arn:aws:s3:::YOUR-BUCKET-NAME-HERE/*"
  }
 ]
}
```

Paste this code snippet in the policy textarea. Replace YOUR-BUCKET-NAME-HERE with the name you have given to your bucket. Save the changes



Step 16: Now reload the website. You can see your website



CONCLUSION:

In this experiment we have hosted the local website on the XAMPP server.The user can access to his website by using the browser then XAMPP server gives the responses to the users in the form of a website which was locally hosted to XAMPP server.

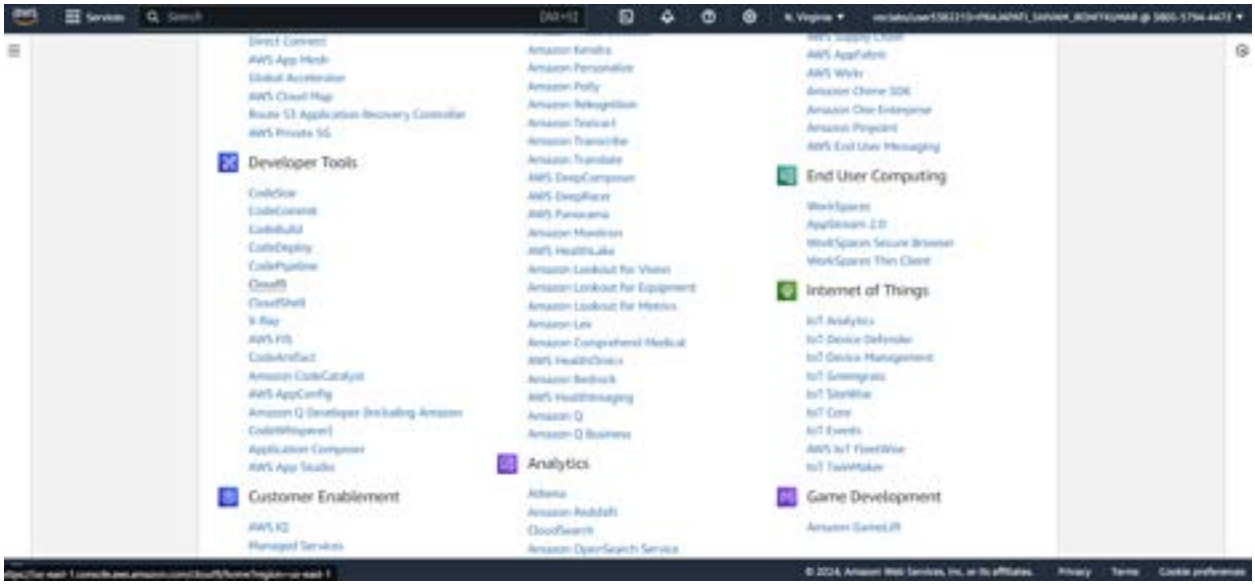
Name: Shivam.R.Prajapati Div: D15C Roll No:41 Academic Year: 2024-25

Experiment No: 1(B)

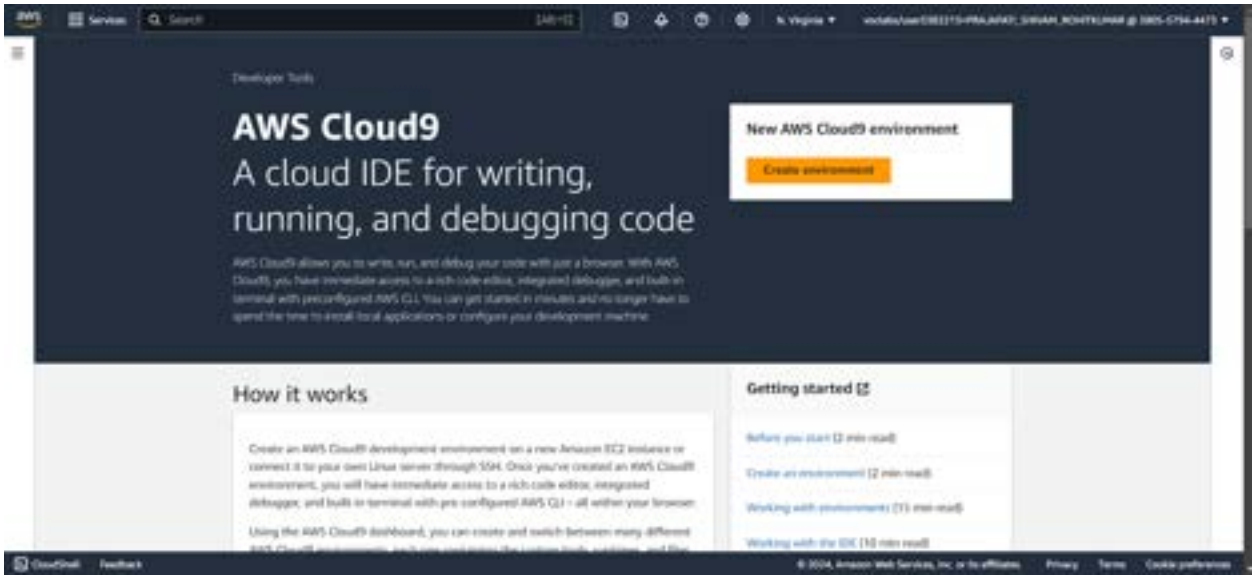
AIM: To understand the benefits of Cloud Infrastructure and Setup AWS Cloud9 IDE, Launch AWS Cloud9 IDE and Perform Collaboration Demonstration.

Step 1: Set up a Cloud9 environment.

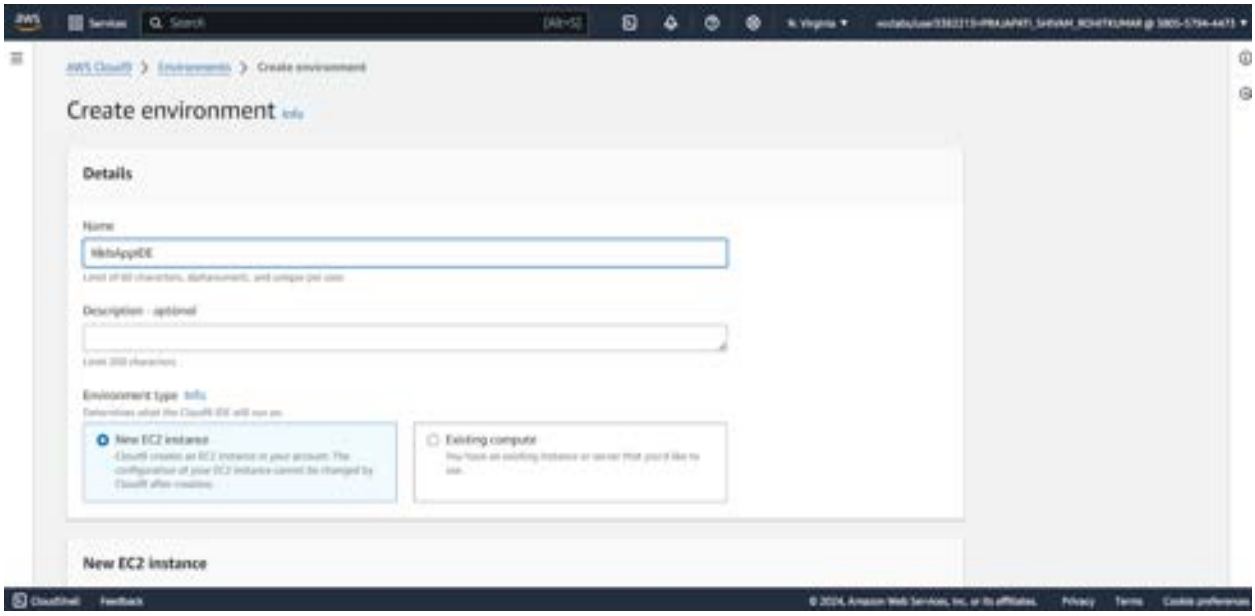
1) Go to Cloud9 services under developers tool in All services



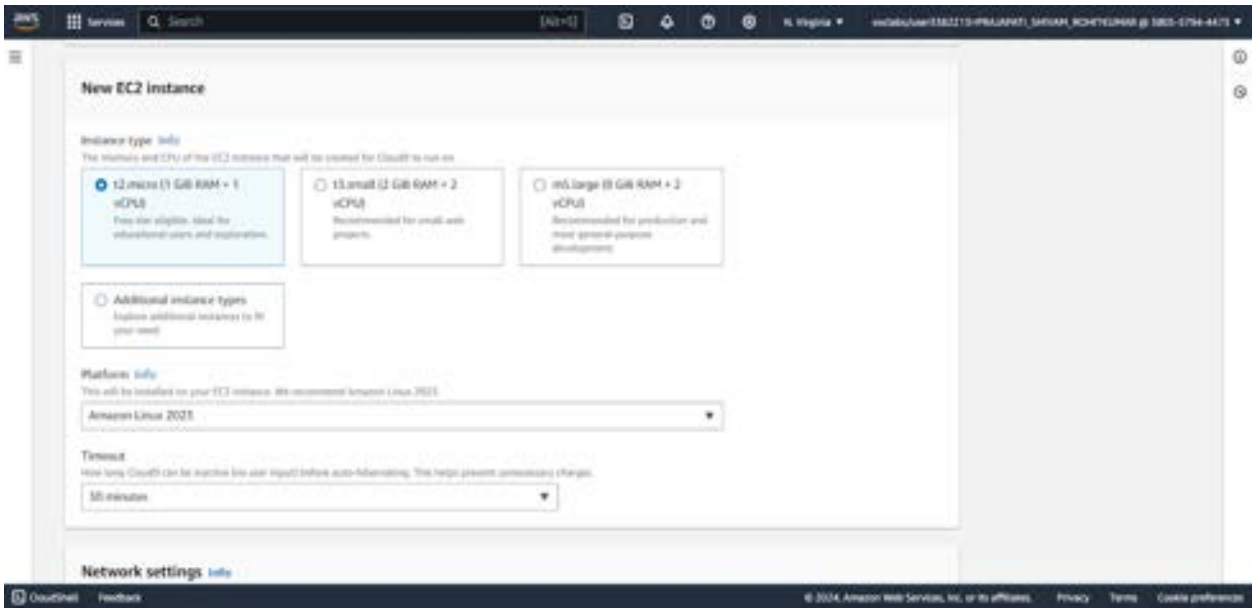
2) Click on create environment



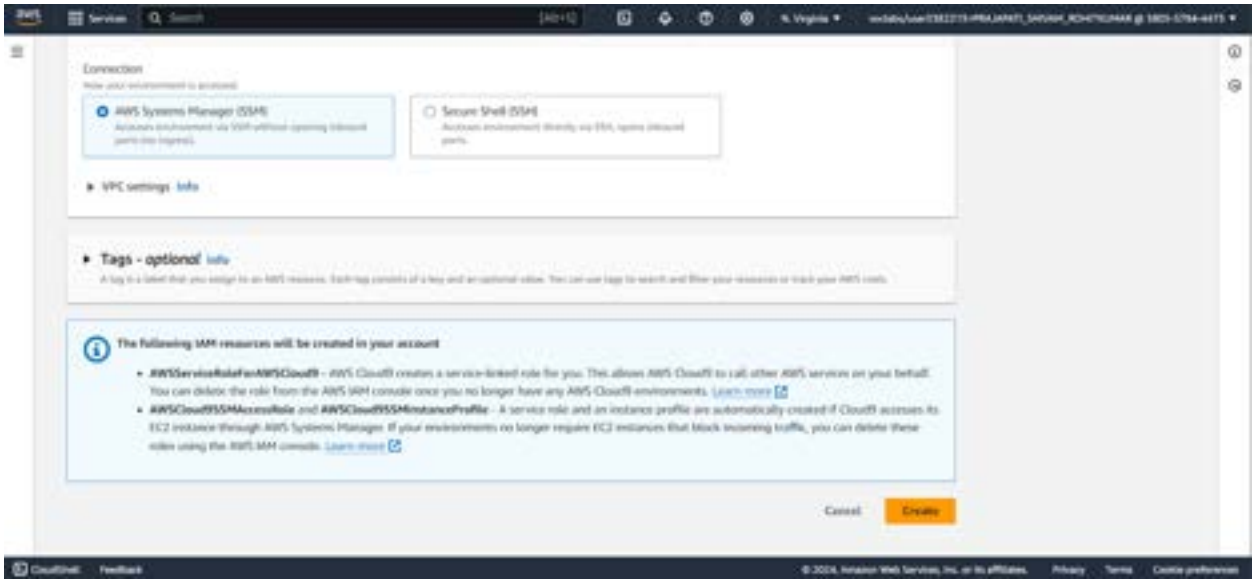
3) Give the name to your Environment ,keeping the other settings as default like environment type should be New EC2 instance



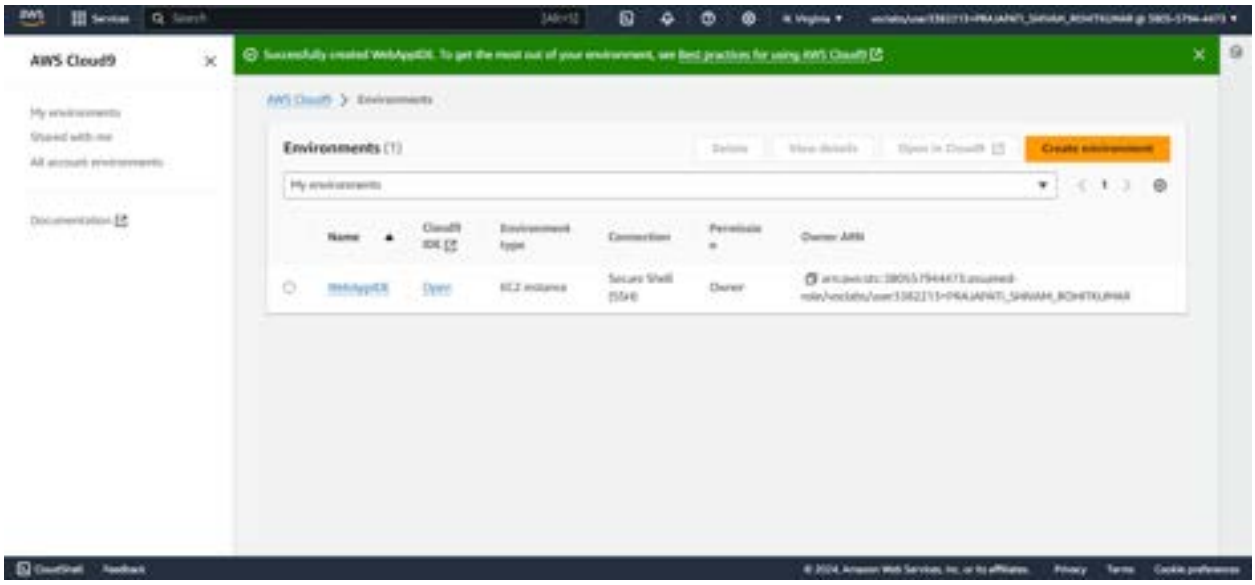
4) Select the correct platform type as shown below and keep the others details as default like instance type as t2.micro which gives the user 1GB RAM + 1 Virtual CPU



5) Click on SSH under connection type in network settings if we go for AWS Manager(SSM) then it won't allow to create an environment then click on Create



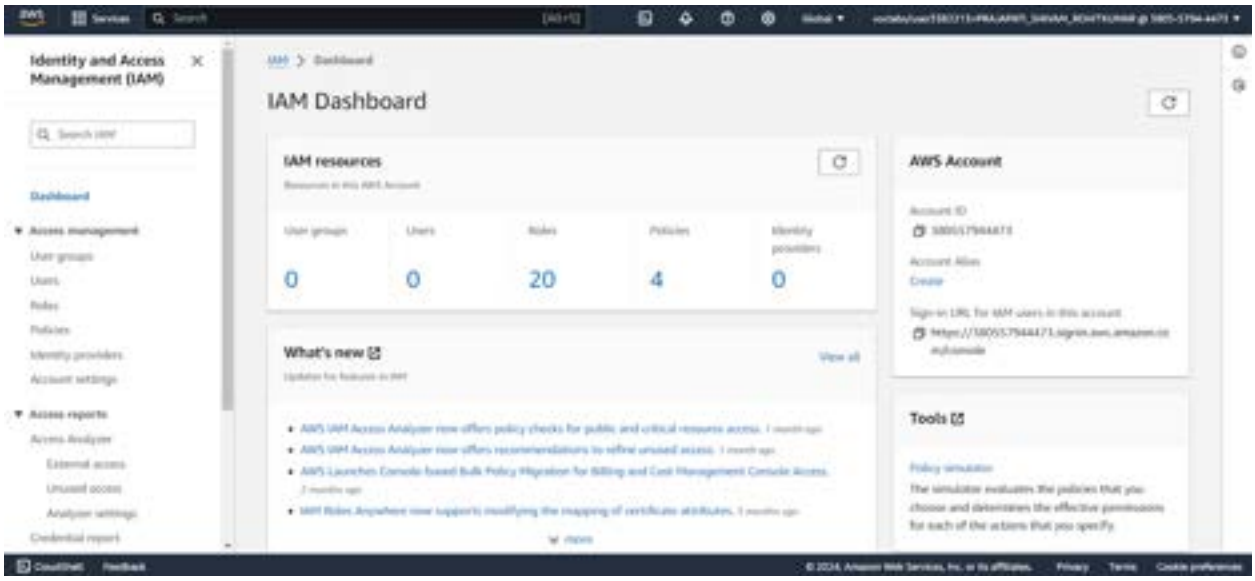
6) Successfully created the environment so now click on open



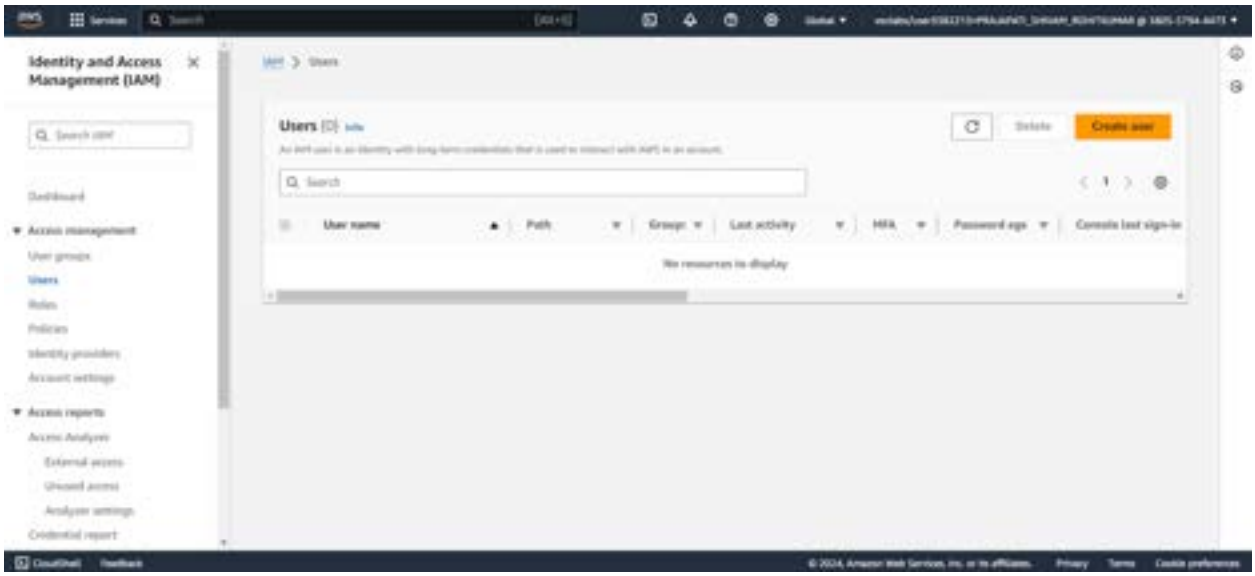
Name: Shivam.R.Prajapati Div: D15C Roll No:41 Academic Year: 2024-25

Step 2: Creating IAM user.

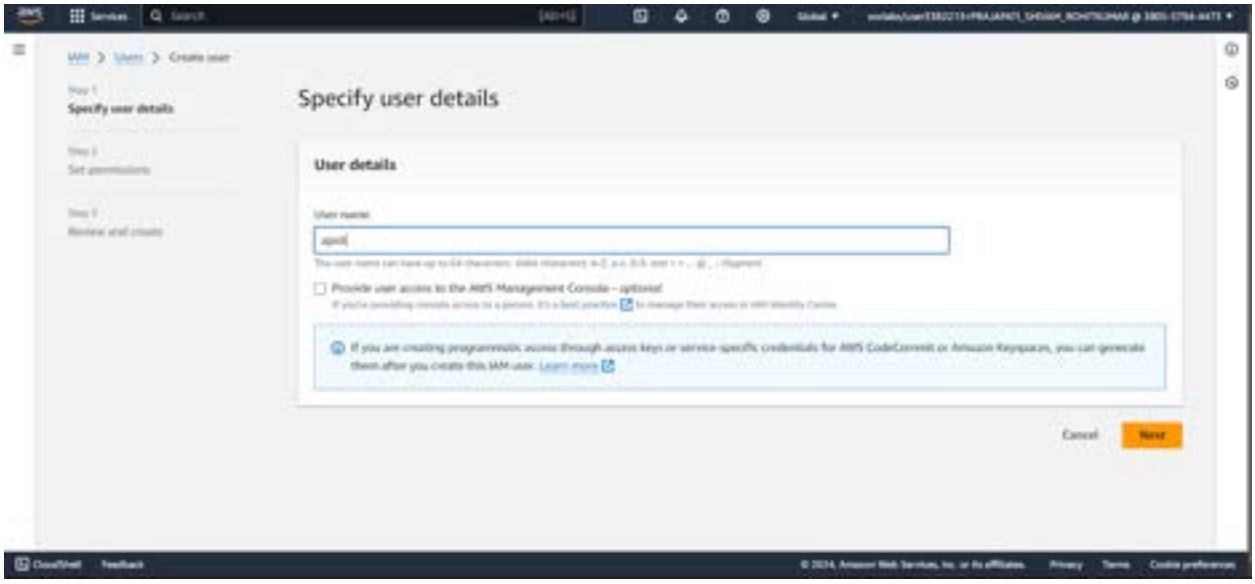
1) Search IAM on the services search bar and open it. Click on Create User



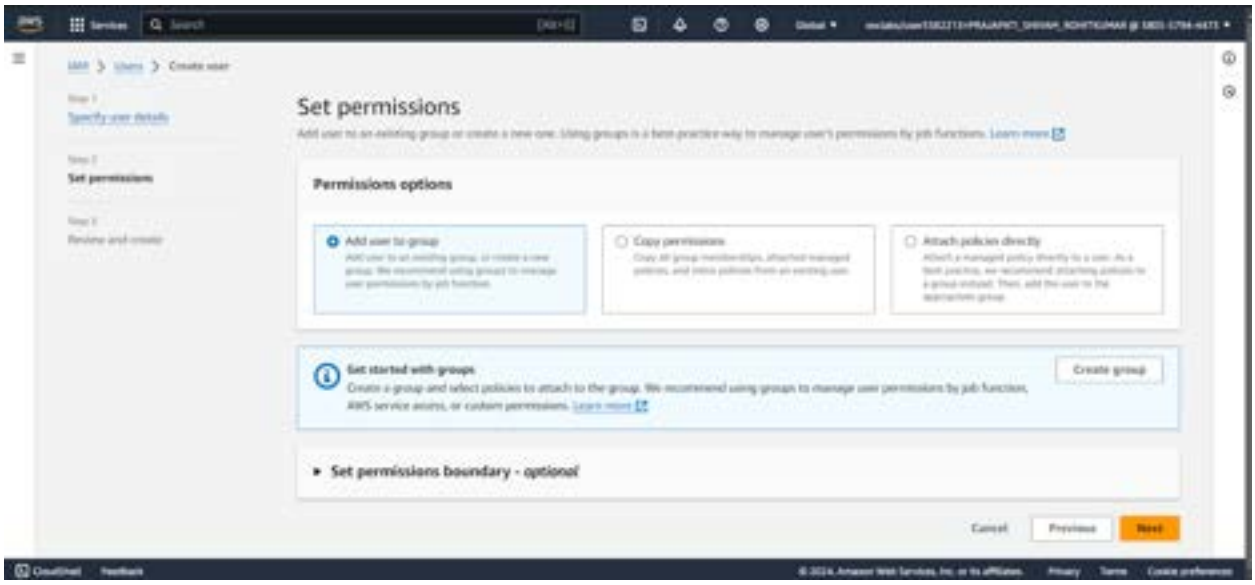
2) Click on the create user



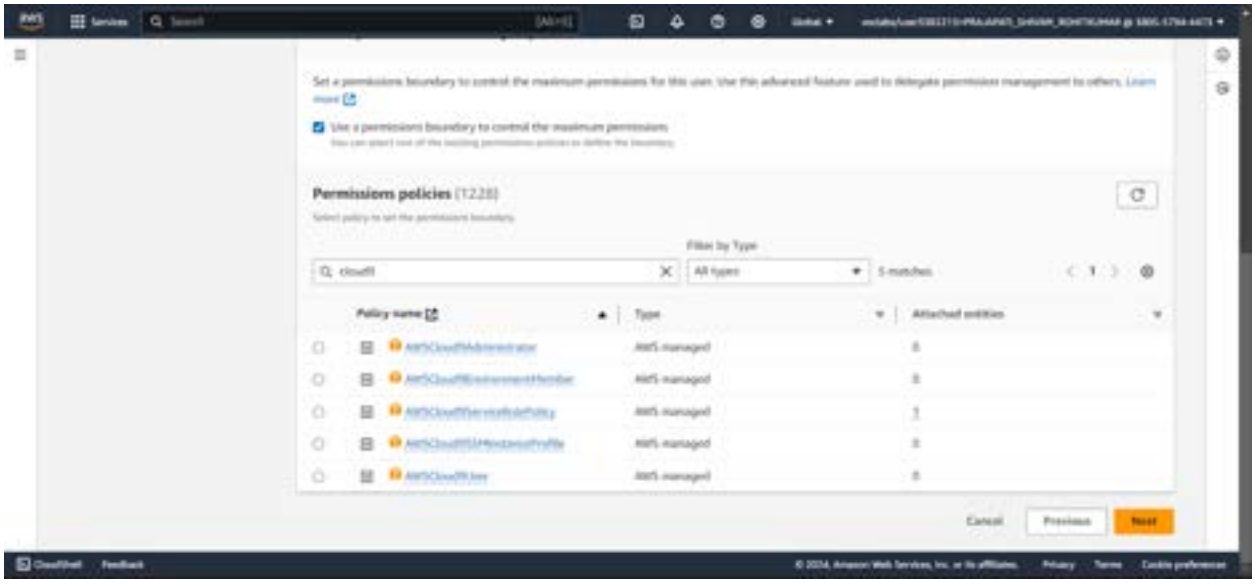
3) Write the name of the user you want to add and click on next



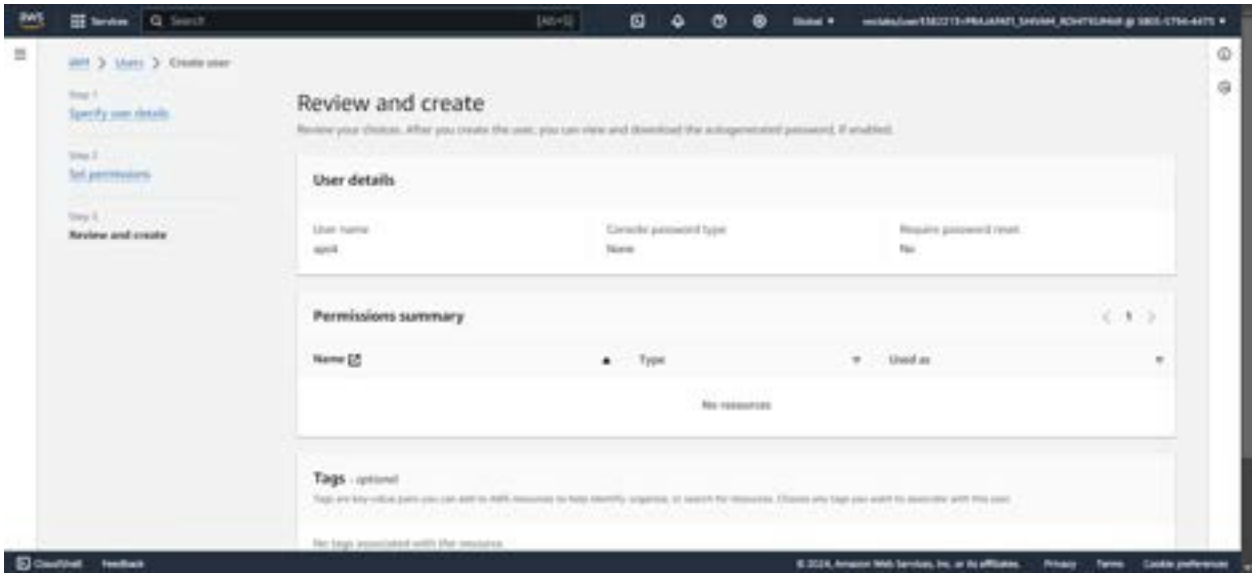
4) Select add User to Group. If there are no user groups on your accounts, you will have to create one. Click on Create Group. Click on the drop down menu of the set permissions boundary

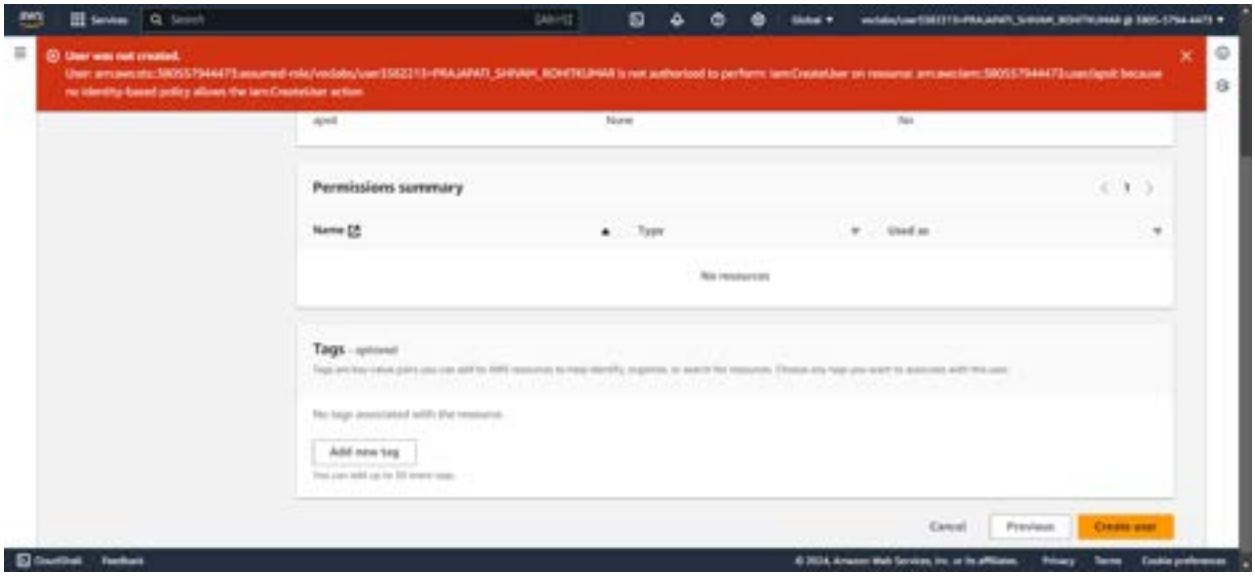


5)Click on the checkbox and search for cloud9 under permissions policies ,click on next



6) Scroll down and click on create user

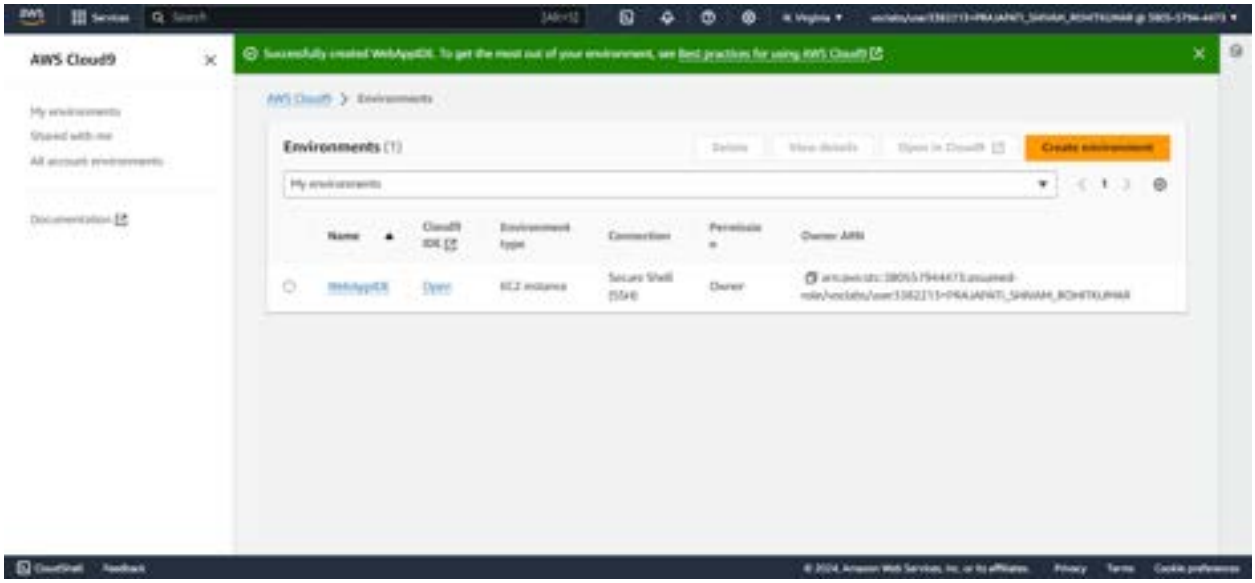




When we go to add user to a group, the AWS Academy account throws an error as we do not have the permissions to create a group. So we have to use our personal AWS account for this part.

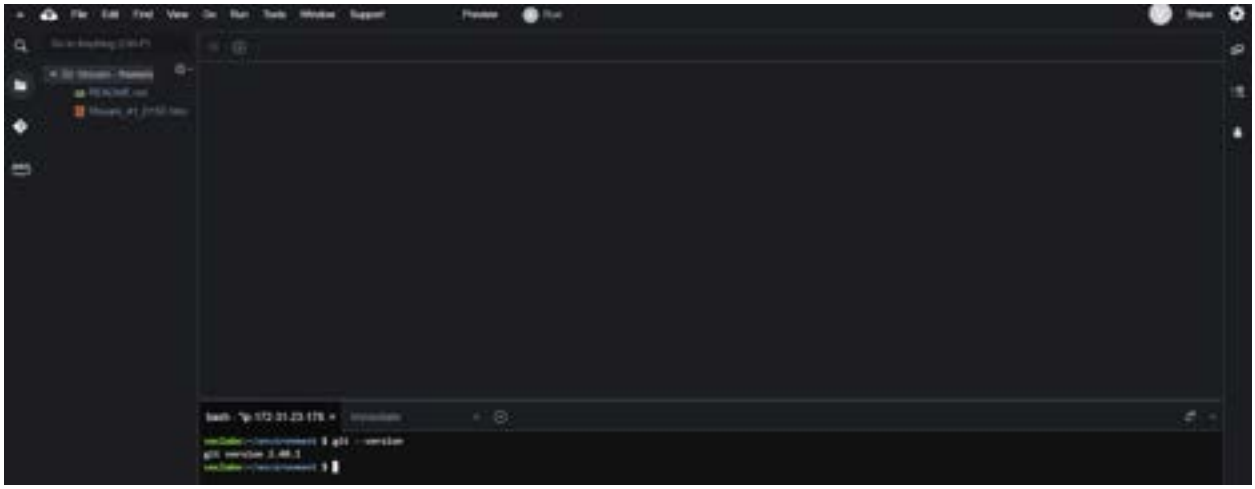
Step 3: Working on Cloud9 IDE

1) Go to Cloud9 services. Click on Open under Cloud9 IDE

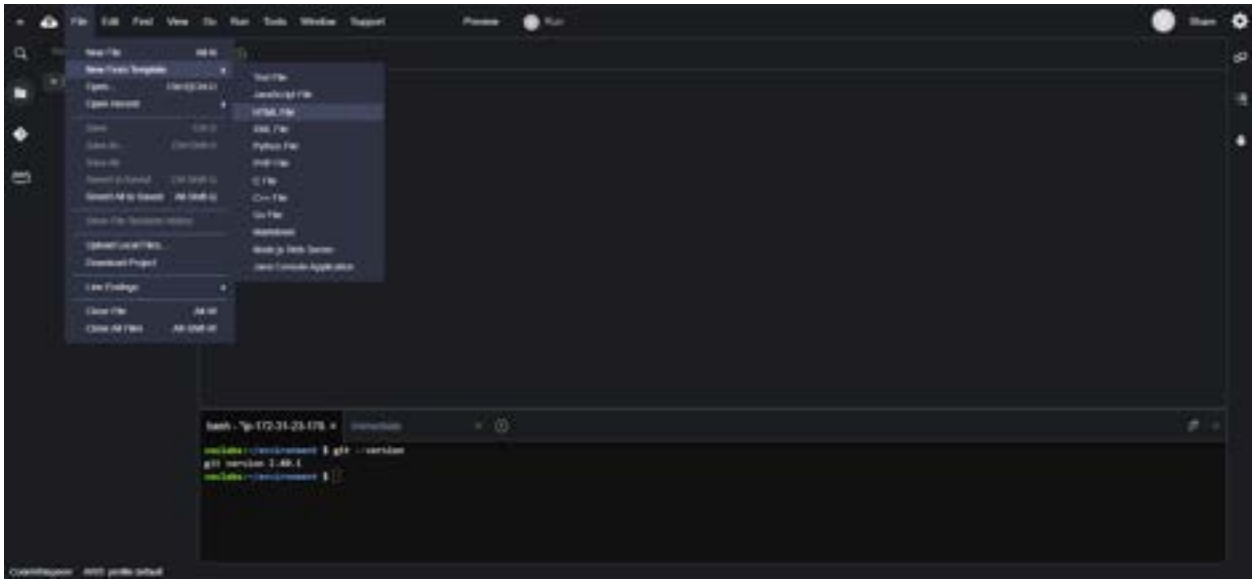


Name: Shivam.R.Prajapati Div: D15C Roll No:41 Academic Year: 2024-25

2) This is the Cloud9 IDE interface. The major part of the screen is the coding IDE. There is a command console just below it. For example, the command git --version is run.

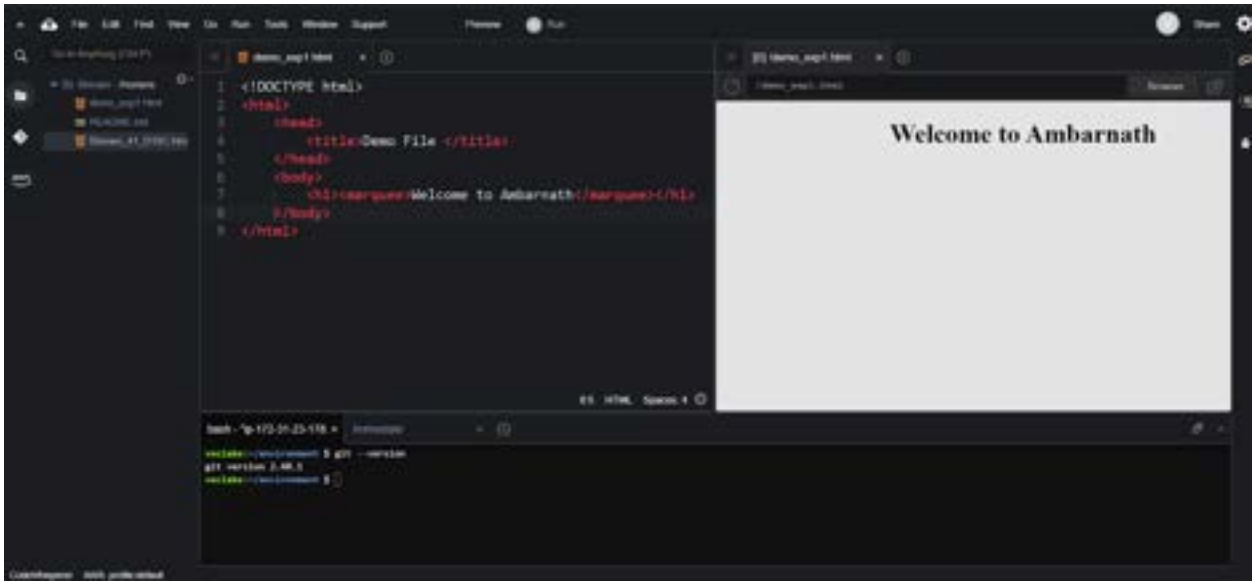


3) To add a file, click on file. For this experiment, we are to add an HTML file. So go to File → New From Template → HTML file. This gives a basic HTML template on the coding IDE



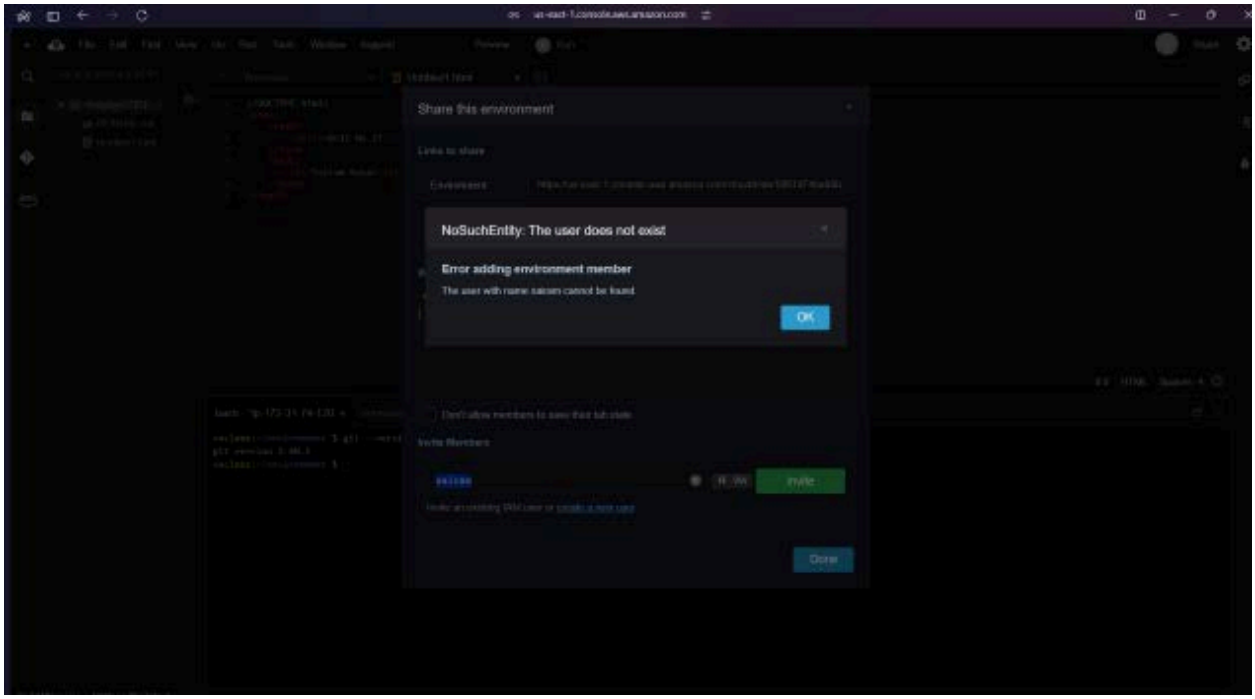
Name: Shivam.R.Prajapati Div: D15C Roll No:41 Academic Year: 2024-25

4) Make a basic website on the HTML template and save it.



.

After saving, on the toolbar towards far right, click on Share. Then put the username that you had put during creating IAM user.



Here, it gives an error as Cloud9 was created on the academy account where creating an IAM group is not available, meanwhile on the personal account, the services of Cloud9 have been deprecated. So currently, it is not possible to integrate the cloud9 and IAM parts of the experiment.

CONCLUSION:

In this part of the experiment we have hosted the website on the AWS cloud using the S3 services .For hosting of the website on the local machine or on the cloud does not require to use the personal account compulsorily ,In S3 service we need to create a bucket and then store our local file/folder to it. After doing required configuration we can globally access our website

Name: Shivam.R.Prajapati Div: D15C Roll No:41 Academic Year: 2024-25

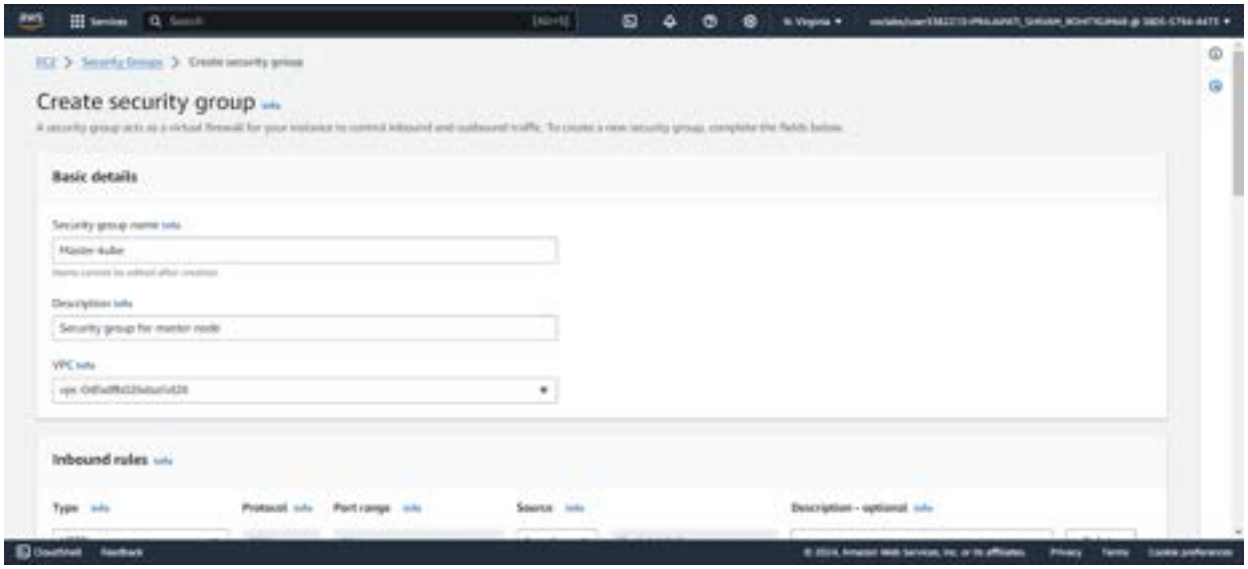
Experiment No : 3

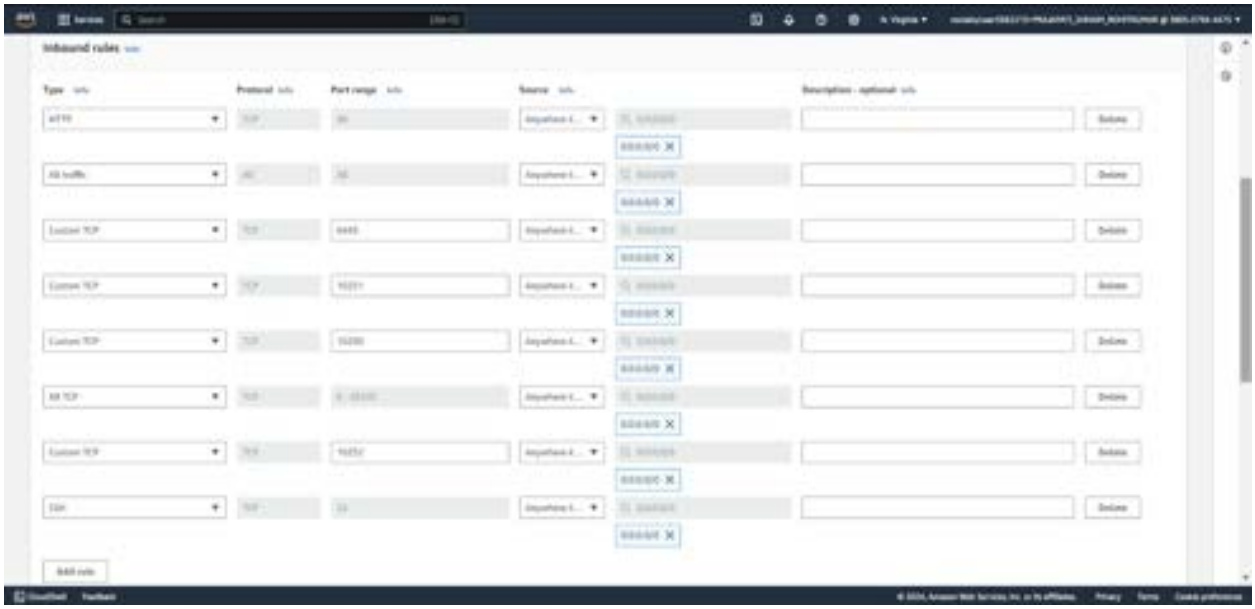
AIM: To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernete Cluster on Linux Machines/Cloud Platforms.

PREREQUISITES:

Create 2 Security Groups for Master Node and Worker Nodes and add the following inbound rules in those Groups.

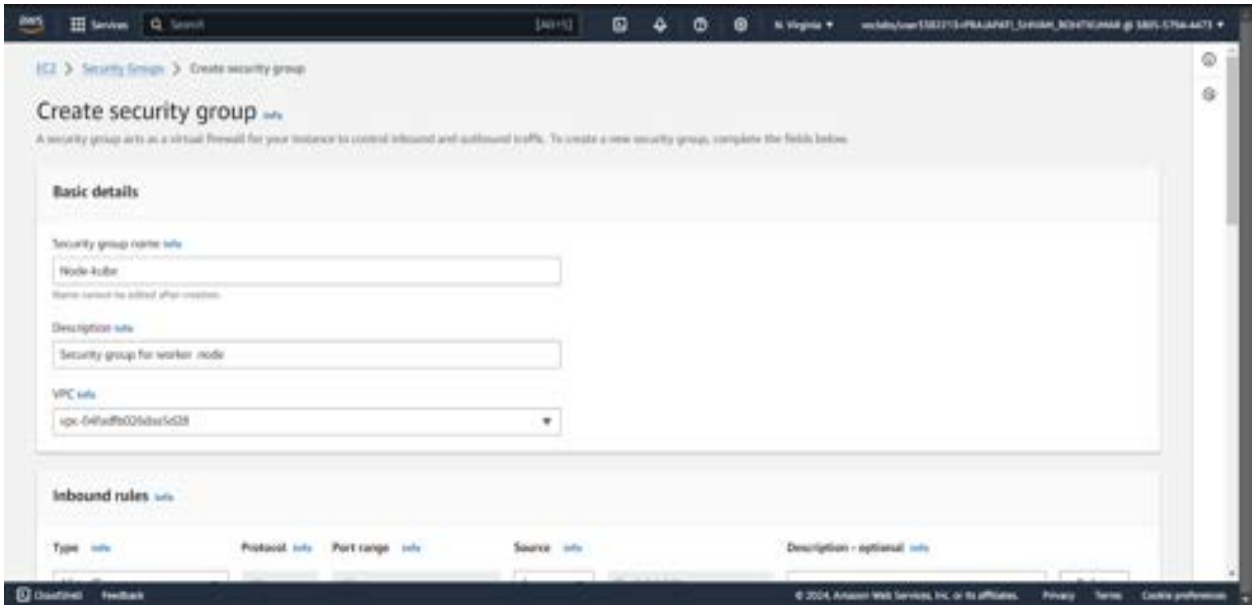
Master Node Security Group:

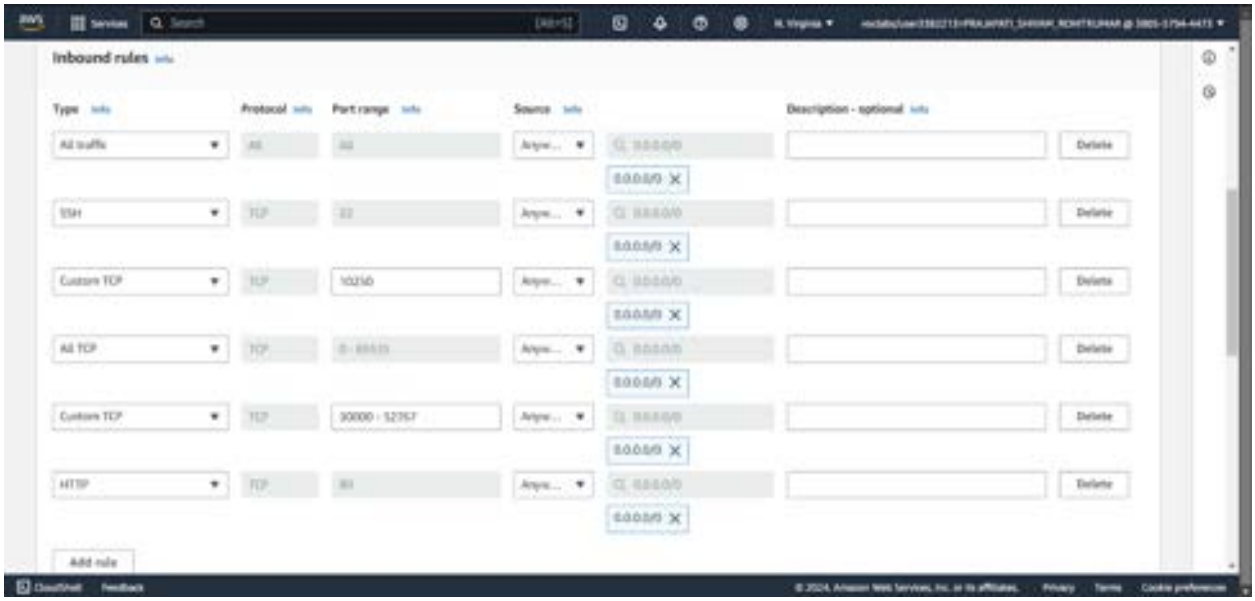




Name: Shivam.R.Prajapati Div: D15C Roll No:41 Academic Year: 2024-25

Worker Node Security Group :



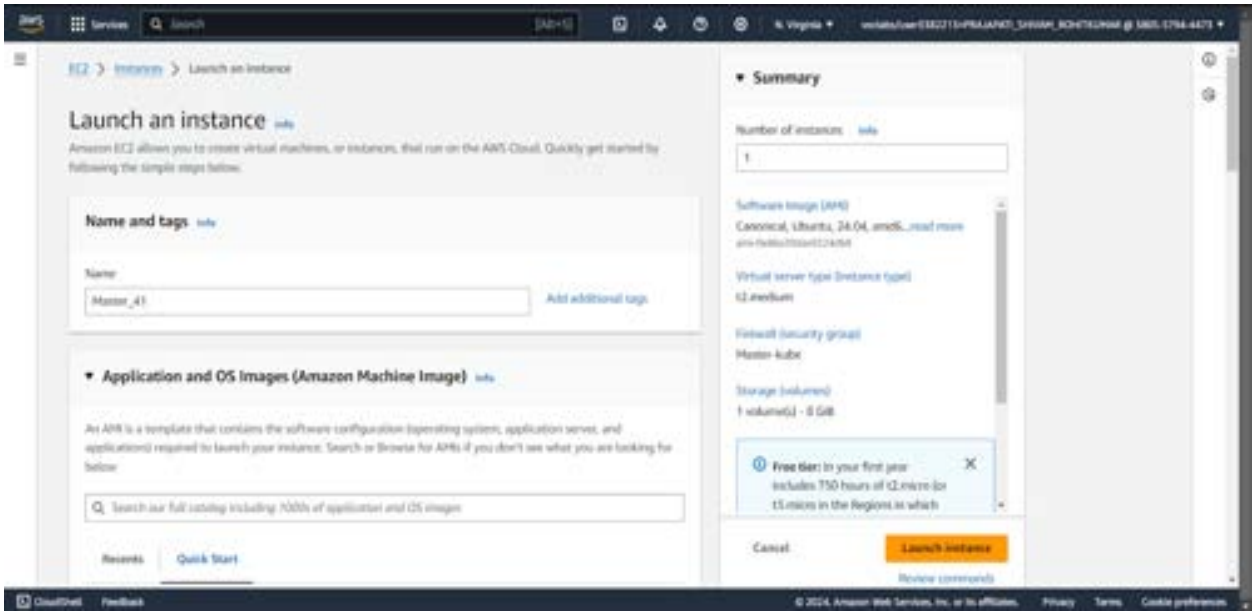


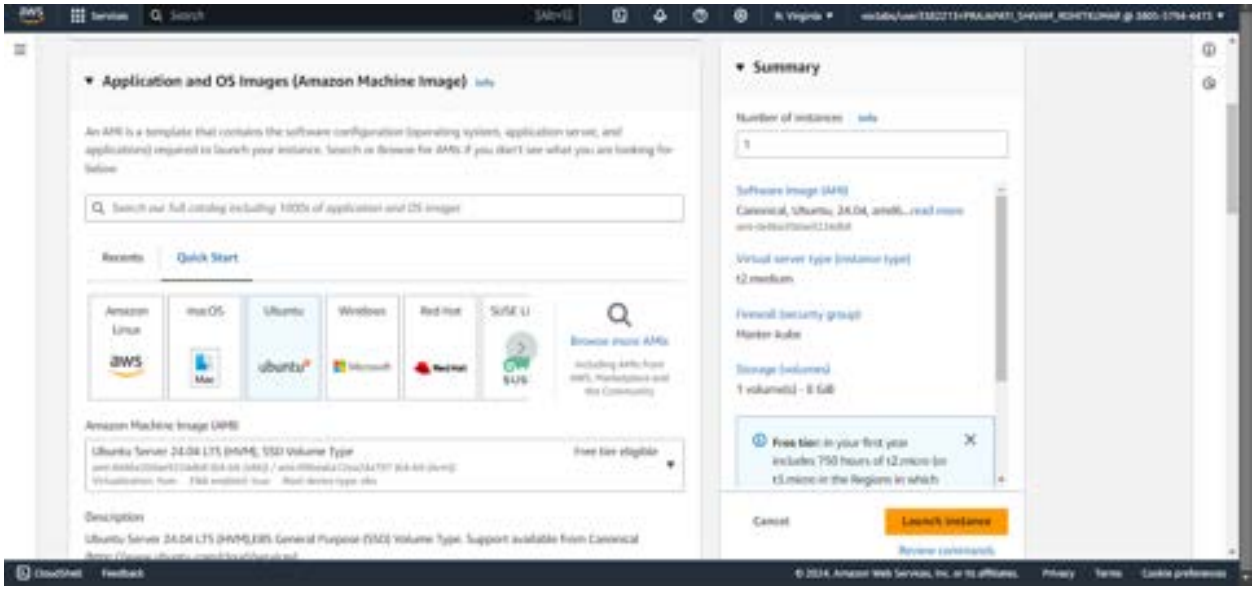
Added Required Inbound rules for worker nodes

Step 1: Access your AWS Academy or personal account and create **three new EC2 instances**. For the AMI, choose **Ubuntu** and set the **instance type to t2.medium**. Generate an RSA key in .pem format and move the downloaded key to a new folder for **eg: Newfolder**; you can either use three separate keys or one shared key for all instances ,in my case I have made three different keys .

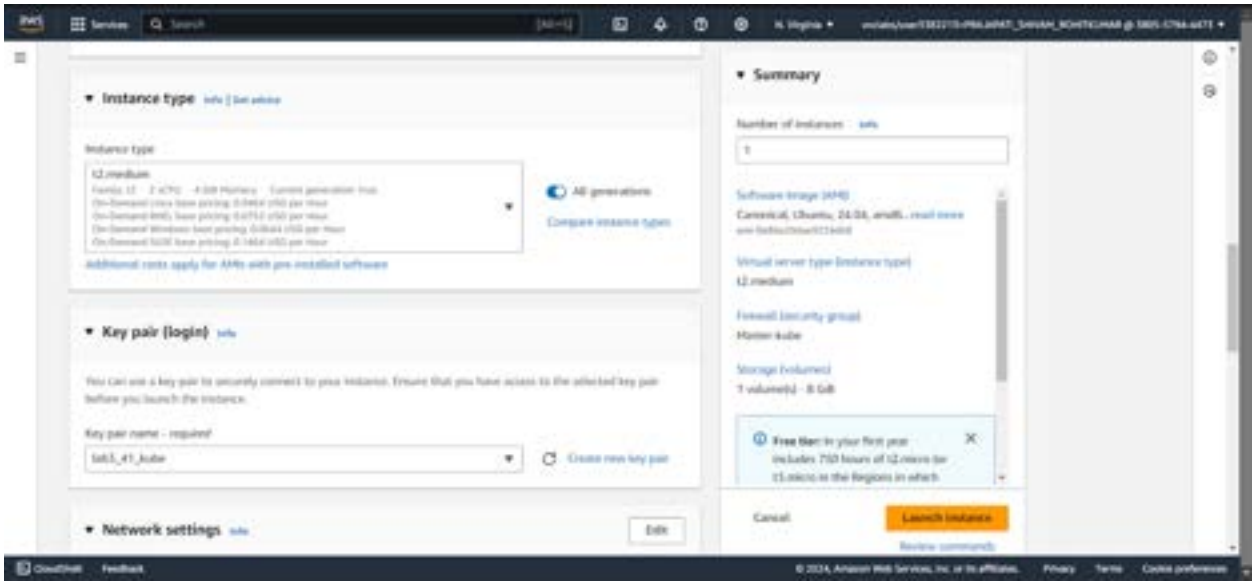
It's essential to select t2.medium, as it includes at least 2 CPUs, which are required for this setup.. Additionally, make sure to select security groups from the ones already available i.e master node and worker node will select their own security groups respectively

Master Node:

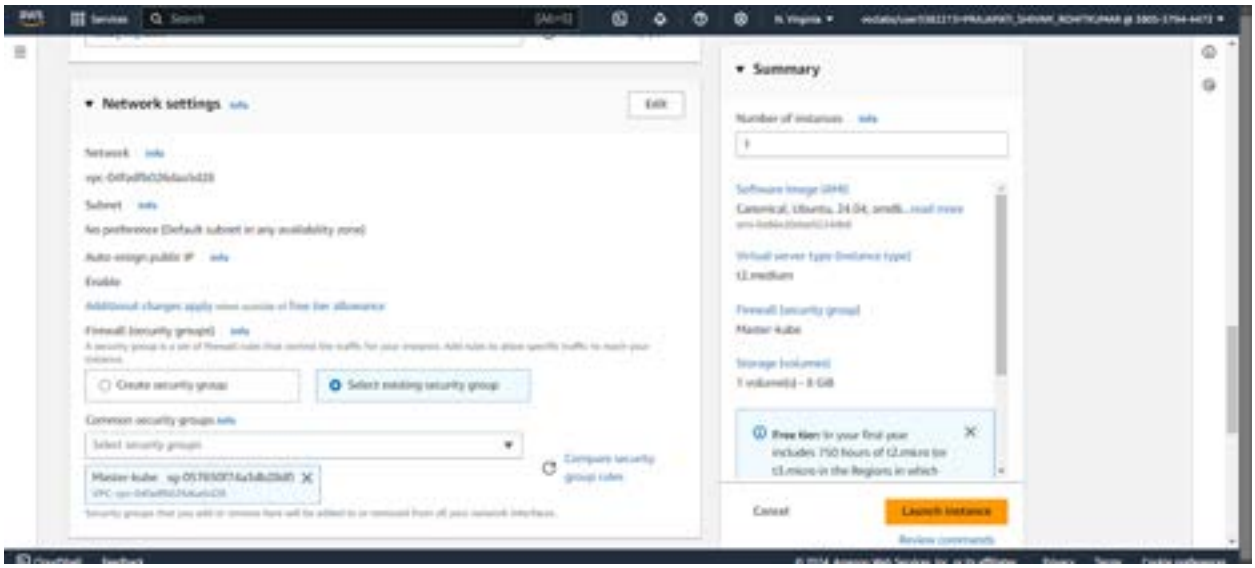




AMI as Ubuntu

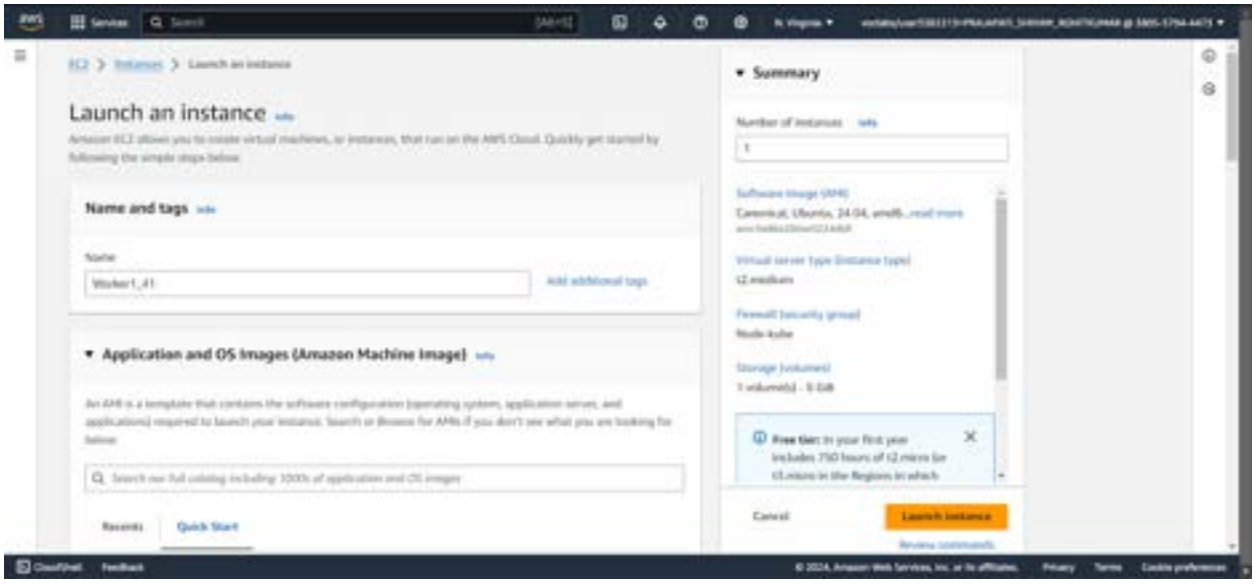


Selected Appropriate Instance type and created key pair login

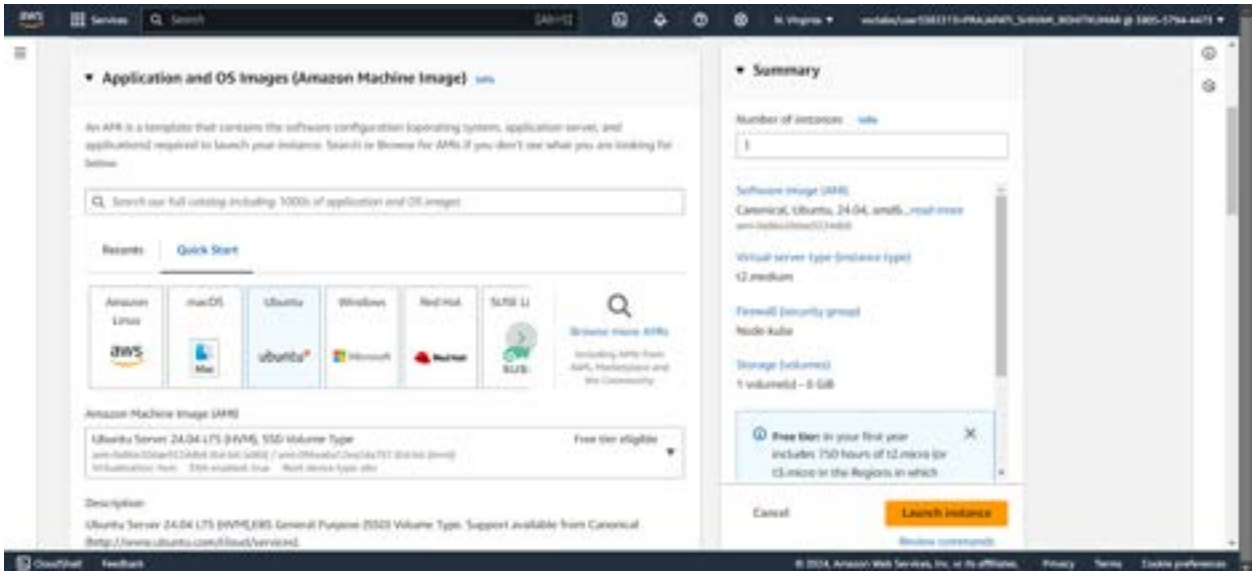


Selected Required Security Group

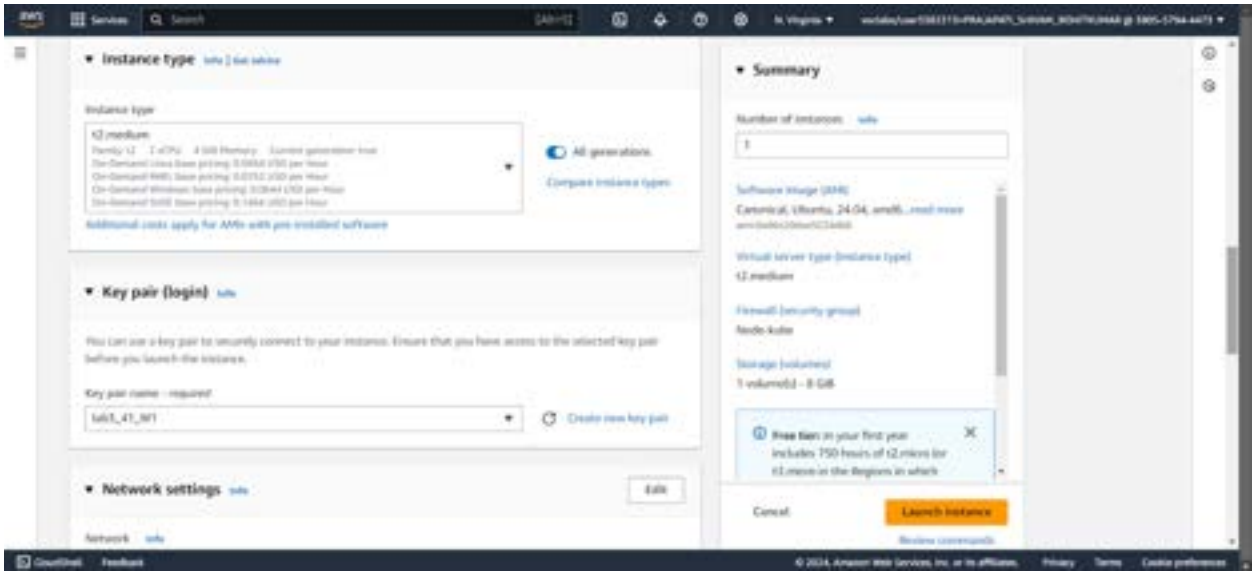
Worker Node 1:



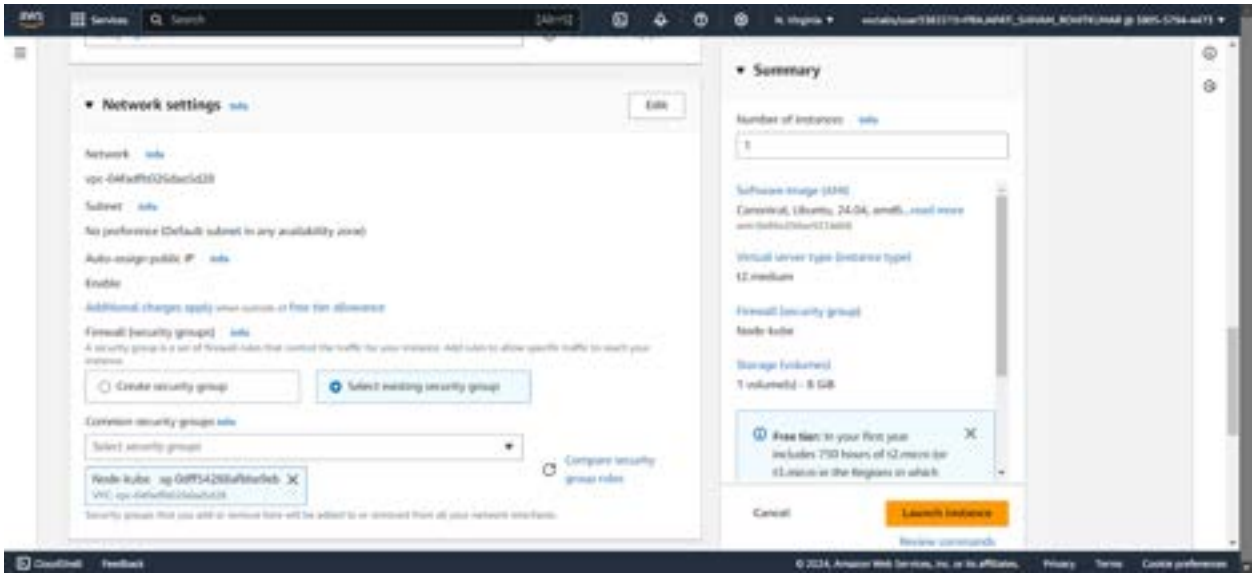
Give a name to your instance .



AMI as Ubuntu



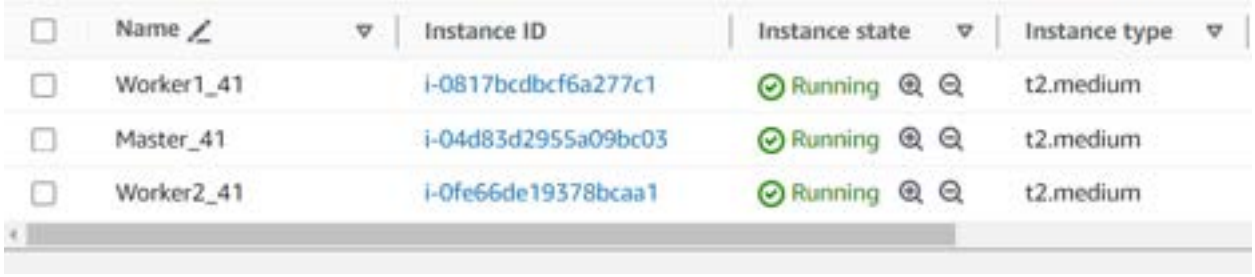
Created the Key pair login



Selected required Security Group

Worker Node 2: Create Worker2_41 instance similarly to previous worker node created .Give Name to your Instance, Select ubuntu as Ami, t2.medium as instance type, create key-pair login as Lab3_41_W2 (you can create of your own) , Select proper security group

Instances are :

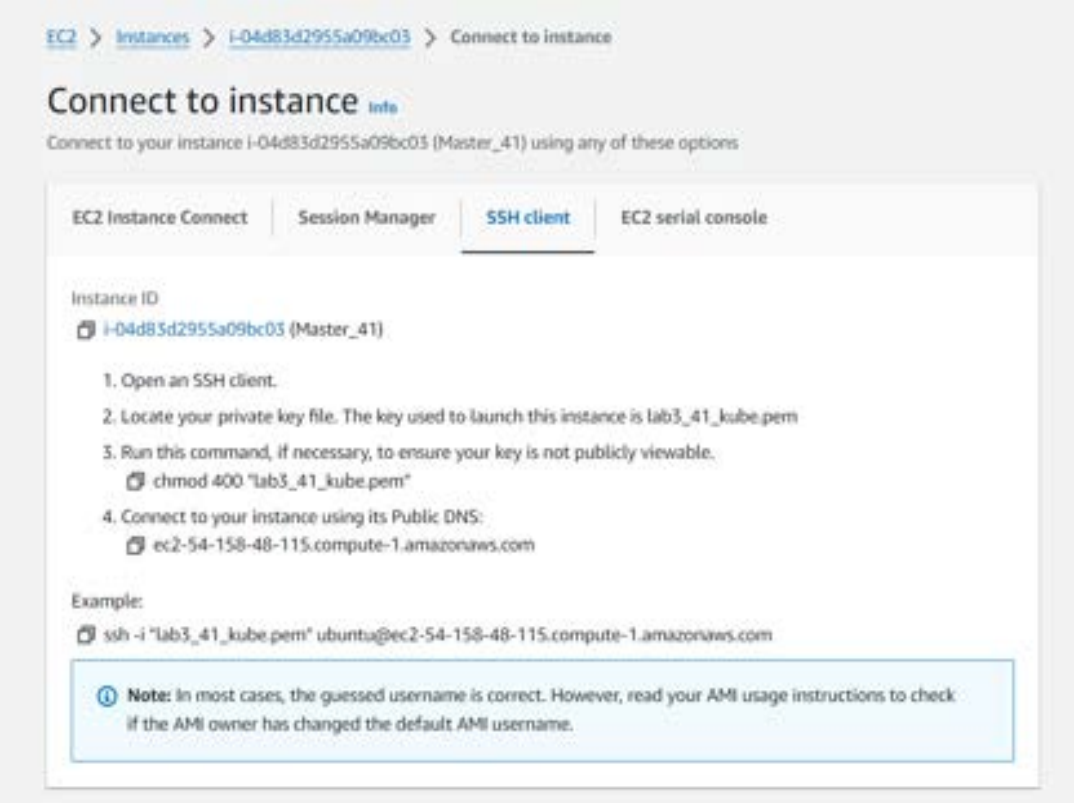


Name	Instance ID	Instance state	Instance type
Worker1_41	i-0817bcdbcf6a277c1	Running	t2.medium
Master_41	i-04d83d2955a09bc03	Running	t2.medium
Worker2_41	i-0fe66de19378bcaa1	Running	t2.medium

Step 2: After creating the instances click on Connect for all the instances one by one and navigate to the SSH Client section .

Name	Instance ID	Instance state	Instance type
Worker1_41	i-0817bcdbcf6a277c1	Running	t2.medium
Master_41	i-04d83d2955a09bc03	Running	t2.medium
Worker2_41	i-0fe66de19378bcaa1	Running	t2.medium

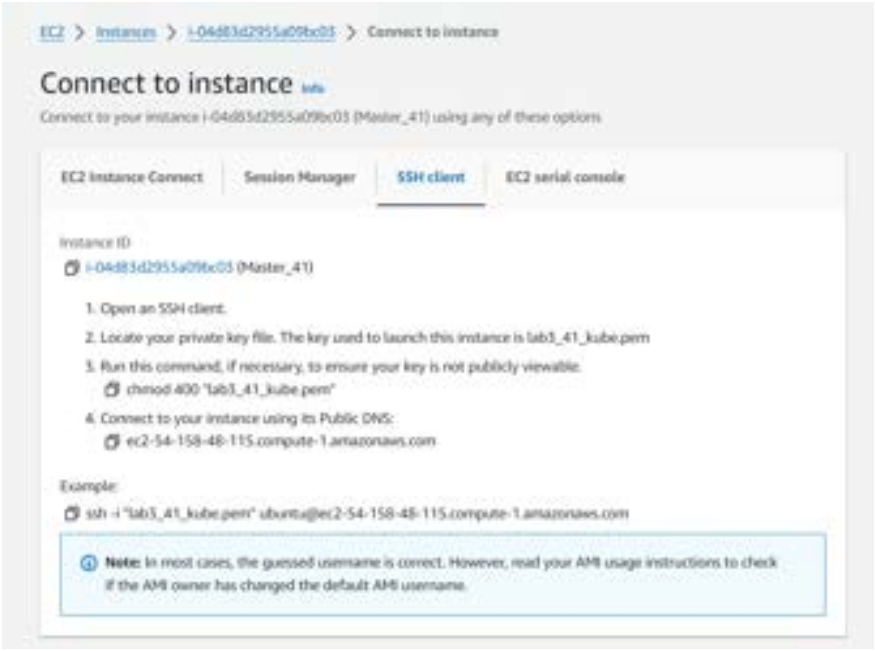
Master Node:



Step 3: Now open the folder in the terminal 3 times by right clicking on it for Master, Node1 & Node 2 where our .pem key is stored and paste the Example command (starting with ssh -i …..) from the ssh client section for instance in the terminal as shown above.

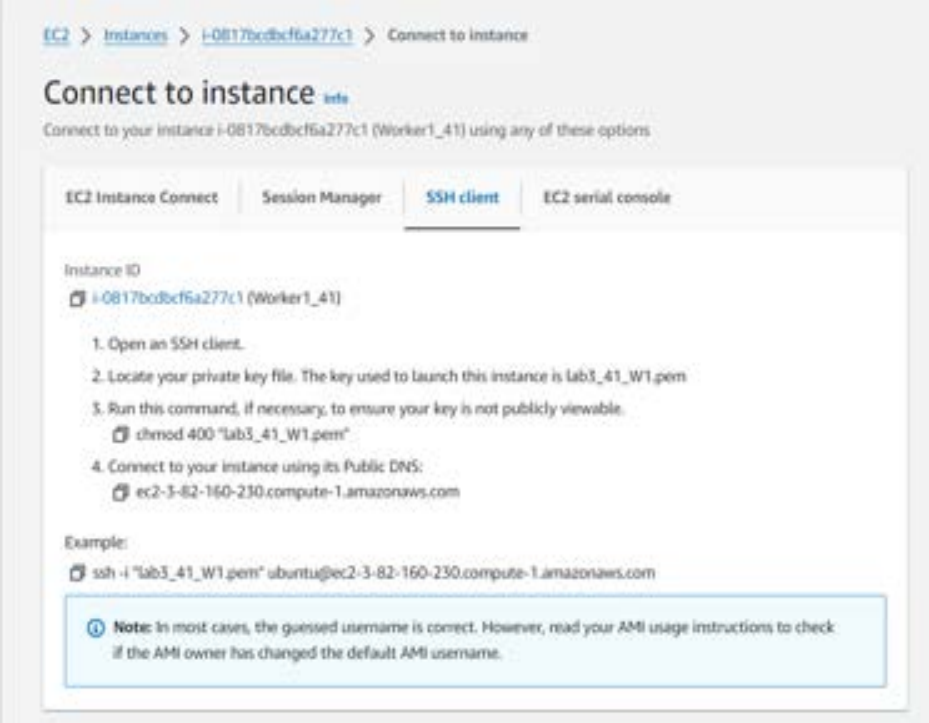
This will basically make our terminal to do remote login on our ec2 instance via SSH

MasterNode:

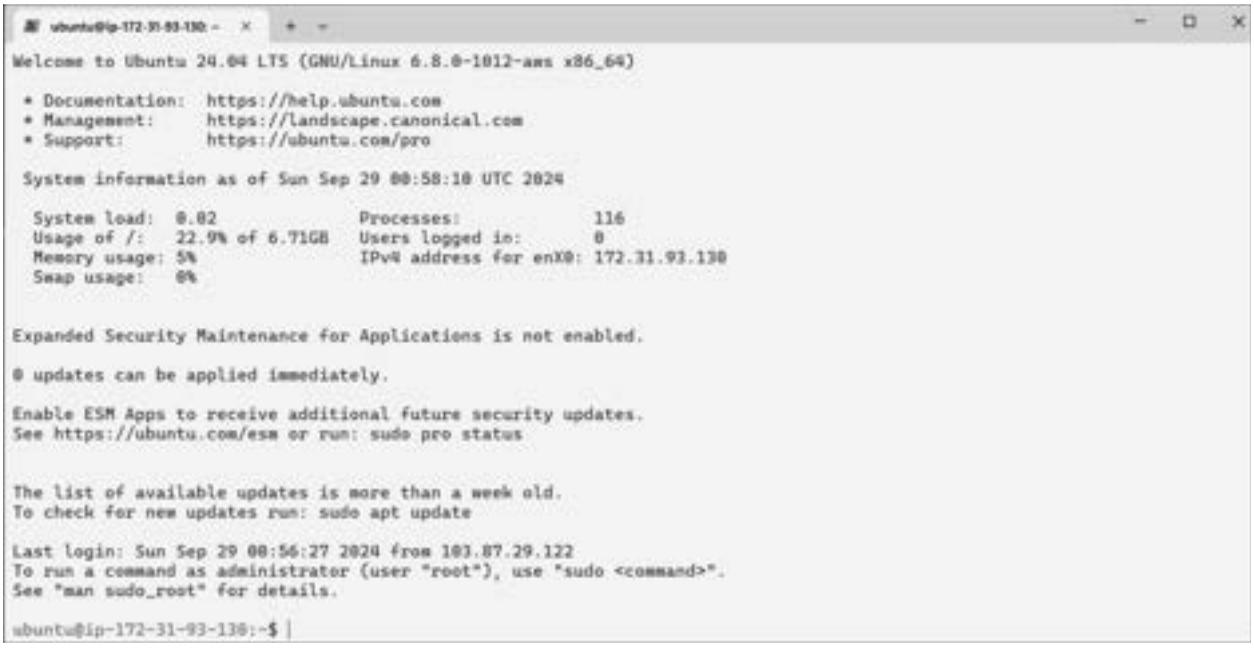


Successful Connection:

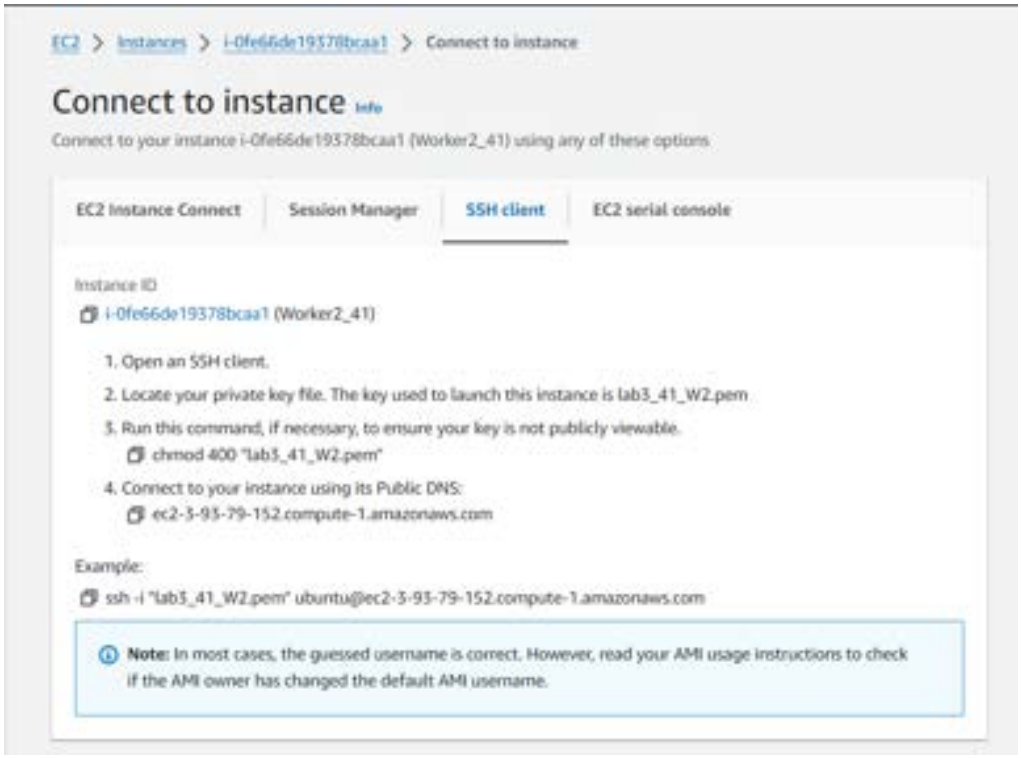
Worker Node 1:



Successful Connection:



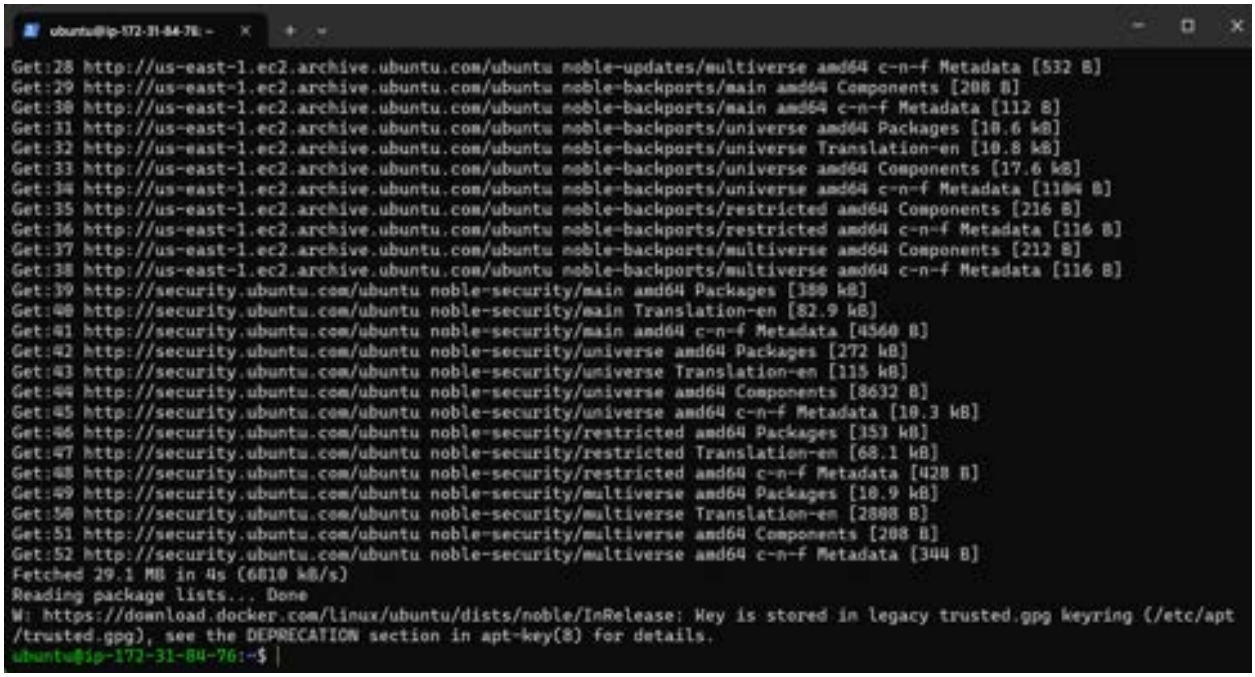
Worker Node 2:



Successful Connection:

Step 4: Run on Master,Worker Node 1,and Worker Node 2 the below commands to install and setup Docker in Master,Worker Node 1,and Worker Node 2.

a) curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/trusted.gpg.d/docker.gpg > /dev/null
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"



b) sudo apt-get update
sudo apt-get install -y docker-ce

```
ubuntu@ip-172-31-80-76:~$ sudo apt-get update
sudo apt-get install -y docker-ce
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt
/trusted.gpg), see the DEPRECATION section in apt-key(8) for details.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  containerd.io docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0
  pigz slirp4netns
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite
The following NEW packages will be installed:
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7
  libslirp0 pigz slirp4netns
0 upgraded, 10 newly installed, 0 to remove and 143 not upgraded.
Need to get 123 MB of archives.
After this operation, 442 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 pigz amd64 2.8-1 [65.6 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libltdl7 amd64 2.4.7-7build1 [40.3 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libslirp0 amd64 4.7.0-1ubuntu3 [63.8 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 slirp4netns amd64 1.2.1-1build2 [34.9 kB]
Get:5 https://download.docker.com/linux/ubuntu noble/stable amd64 containerd.io amd64 1.7.22-1 [29.5 MB]
```

```
ubuntu@ip-172-31-84-76: ~
Unpacking slirp4netns (1.2.1-1build2) ...
Setting up docker-buildx-plugin (0.17.1-1~ubuntu.24.04~noble) ...
Setting up containerd.io (1.7.22-1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /usr/lib/systemd/system/containerd.serv
ice.
Setting up docker-compose-plugin (2.29.7-1~ubuntu.24.04~noble) ...
Setting up libltdl7:amd64 (2.4.7-7build1) ...
Setting up docker-ce-cli (5:27.3.1-1~ubuntu.24.04~noble) ...
Setting up libslirp0:amd64 (4.7.0-1ubuntu3) ...
Setting up pigz (2.8-1) ...
Setting up docker-ce-rootless-extras (5:27.3.1-1~ubuntu.24.04~noble) ...
Setting up slirp4netns (1.2.1-1build2) ...
Setting up docker-ce (5:27.3.1-1~ubuntu.24.04~noble) ...
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /usr/lib/systemd/system/docker.socket.
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for libc-bin (2.39-0ubuntu8.2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-84-76:~$
```

c)sudo mkdir -p /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"]

}
EOF

```
ubuntu@ip-172-31-84-76:~$ sudo mkdir -p /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
ubuntu@ip-172-31-84-76:~$
```

d) sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker

```
ubuntu@ip-172-31-84-76:~$ sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
ubuntu@ip-172-31-84-76:~$
```

Step 5: Run the below command to **install Kubernetes** on all the three terminals one by one

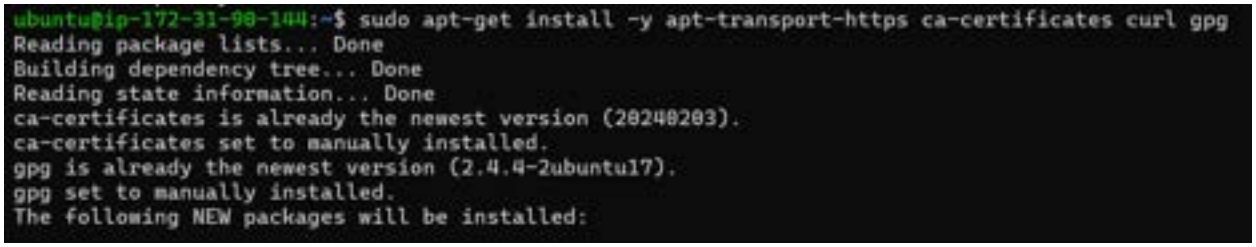
a) sudo rm -f /etc/apt/sources.list.d/kubernetes.list

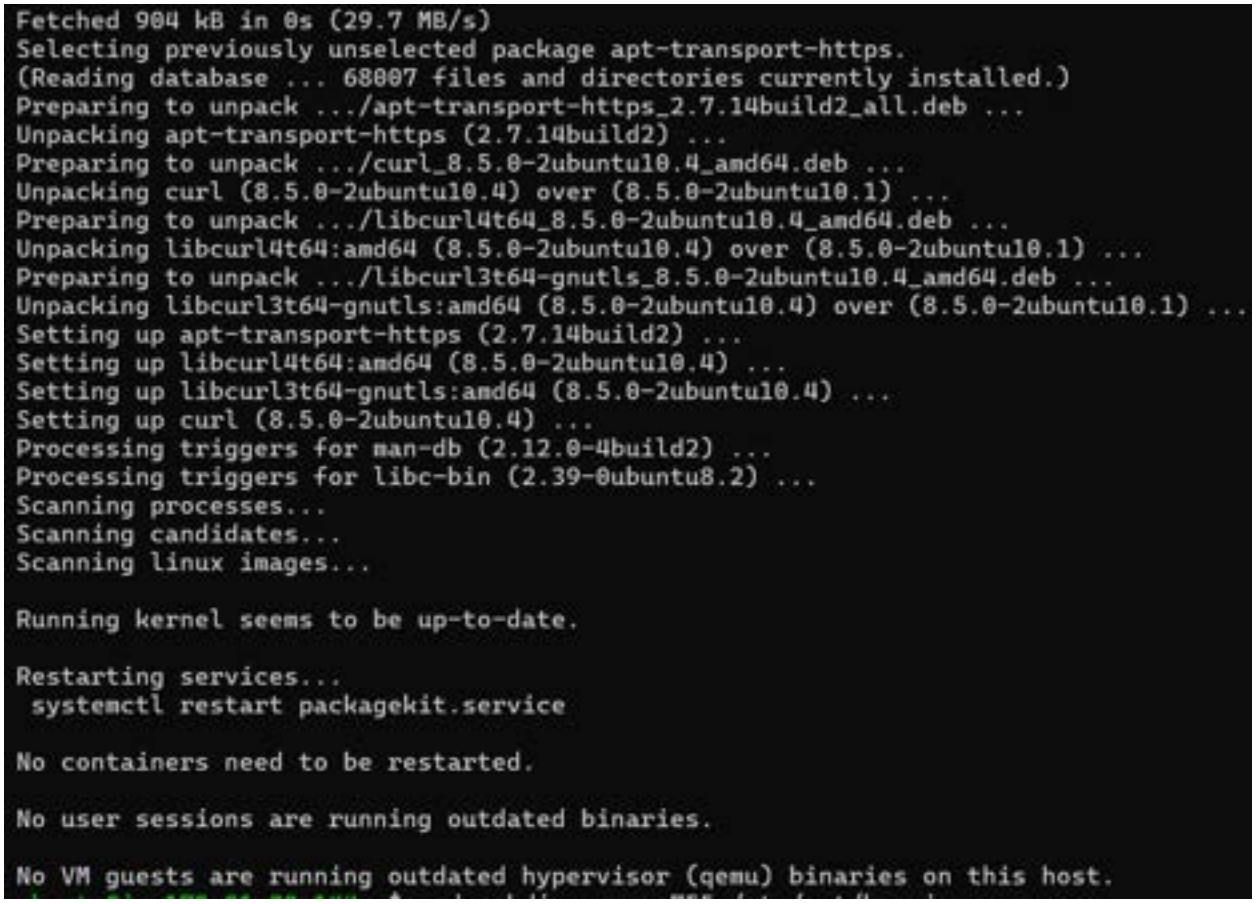
```
ubuntu@ip-172-31-90-144:~$ sudo rm -f /etc/apt/sources.list.d/kubernetes.list
```

b) sudo apt-get update

```
ubuntu@ip-172-31-90-144:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:5 https://download.docker.com/linux/ubuntu noble InRelease
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored
ection in apt-key(8) for details.
```

c) sudo apt-get install -y apt-transport-https ca-certificates curl gpg





d) sudo mkdir -p -m 755 /etc/apt/keyrings

f) curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

g) echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

```
ubuntu@ip-172-31-90-144:~$ echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.
c/apt/sources.list.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
```

f) sudo apt-get update

sudo apt-get install -y kubelet kubeadm kubectl

sudo apt-mark hold kubelet kubeadm kubectl

```
ubuntu@ip-172-31-90-144:~$ sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 https://download.docker.com/linux/ubuntu noble InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  InRelease [1186 B]
Hit:6 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:7 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  Packages [4865 B]
Fetched 6051 B in 1s (9987 B/s)
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.
ection in apt-key(8) for details.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  conntrack cri-tools kubernetes-cni
The following NEW packages will be installed:
  conntrack cri-tools kubeadm kubectl kubelet kubernetes-cni
0 upgraded, 6 newly installed, 0 to remove and 140 not upgraded.
Need to get 87.4 MB of archives.
After this operation, 314 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 conntrack amd64 1:1.4.8-1ubuntu1 [37.9 kB]
Get:2 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  cri-tools 1.31.1-1.1 [15.7 MB]
Get:3 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  kubeadm 1.31.1-1.1 [11.4 MB]
Get:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  kubectl 1.31.1-1.1 [11.2 MB]
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  kubernetes-cni 1.5.1-1.1 [33.9 MB]
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  kubelet 1.31.1-1.1 [15.2 MB]
Fetched 87.4 MB in 1s (70.6 MB/s)
Selecting previously unselected package conntrack.
```

```
Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
```

g) sudo systemctl enable --now kubelet

```
ubuntu@ip-172-31-84-76:~$ sudo systemctl enable --now kubelet
```

h) sudo apt-get install -y containerd

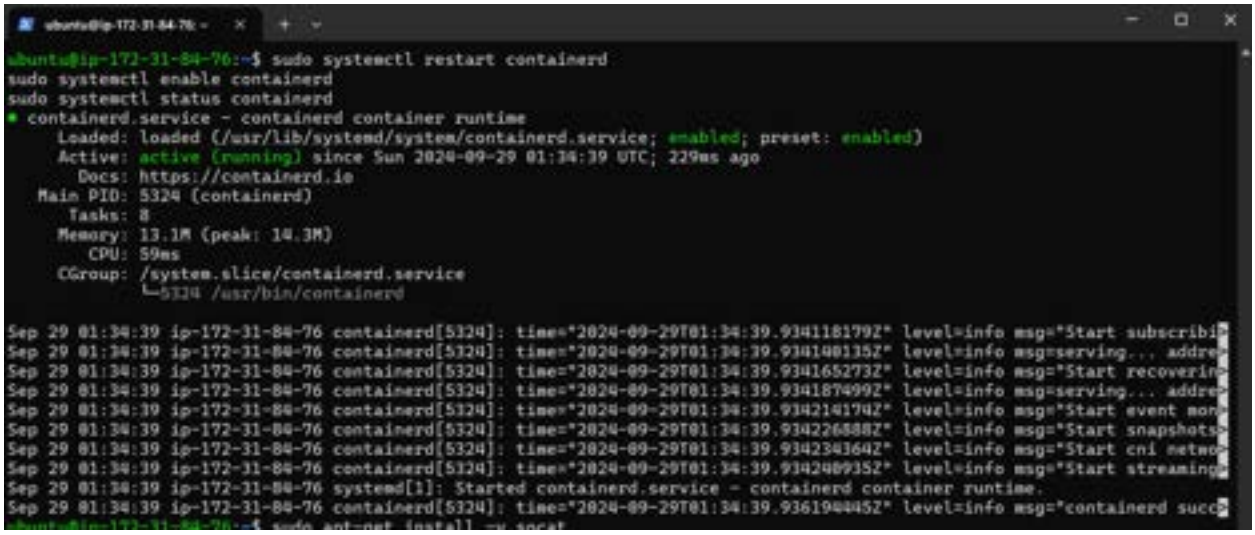
```
ubuntu@ip-172-31-84-76:~$ sudo apt-get install -y containerd
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0 pigz
  slirp4netns
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  runc
The following packages will be REMOVED:
  containerd.io docker-ce
The following NEW packages will be installed:
  containerd runc
0 upgraded, 2 newly installed, 2 to remove and 140 not upgraded.
Need to get 47.2 MB of archives.
After this operation, 53.1 MB disk space will be freed.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 runc amd64 1.1.12-0ubuntu3.1 [8599 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 containerd amd64 1.7.12-0ubuntu4.1 [38.6 M
B]
Fetched 47.2 MB in 1s (83.3 MB/s)
(Reading database ... 68068 files and directories currently installed.)
Removing docker-ce (5:27.3.1-1~ubuntu.24.04~noble) ...
Removing containerd.io (1.7.22-1) ...
Selecting previously unselected package runc.
(Reading database ... 68048 files and directories currently installed.)
Preparing to unpack .../runc_1.1.12-0ubuntu3.1_amd64.deb ...
Unpacking runc (1.1.12-0ubuntu3.1) ...
Selecting previously unselected package containerd.
```

```
Removing docker-ce (5:27.3.1-1~ubuntu.24.04~noble) ...
Removing containerd.io (1.7.22-1) ...
Selecting previously unselected package runc.
(Reading database ... 68048 files and directories currently installed.)
Preparing to unpack .../runc_1.1.12-0ubuntu3.1_amd64.deb ...
Unpacking runc (1.1.12-0ubuntu3.1) ...
Selecting previously unselected package containerd.
Preparing to unpack .../containerd_1.7.12-0ubuntu4.1_amd64.deb ...
Unpacking containerd (1.7.12-0ubuntu4.1) ...
Setting up runc (1.1.12-0ubuntu3.1) ...
Setting up containerd (1.7.12-0ubuntu4.1) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
```

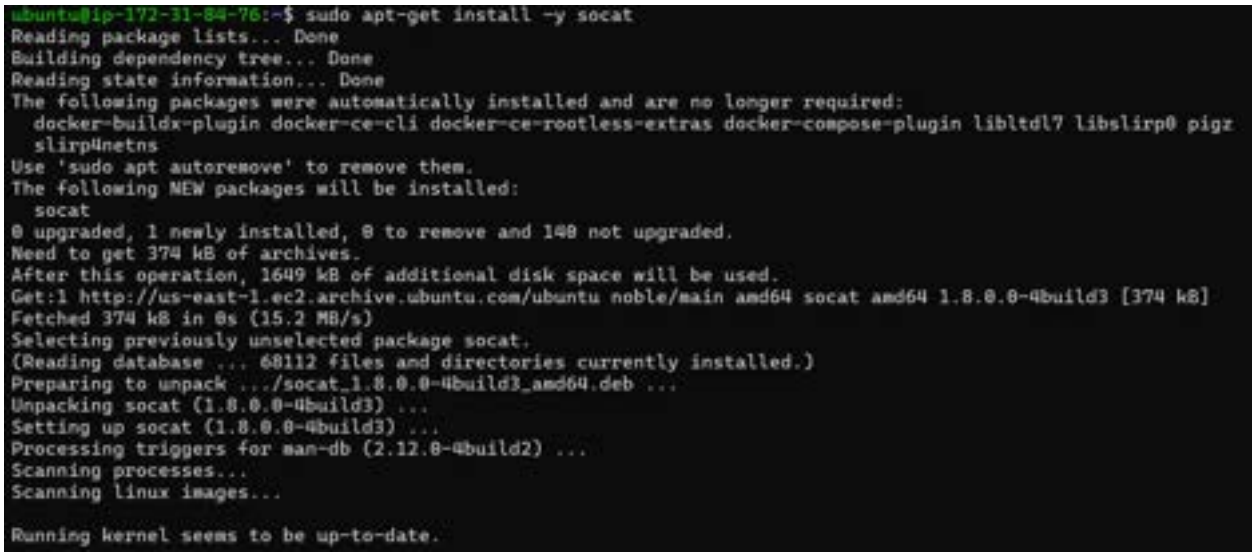
i) sudo mkdir -p /etc/containerd
sudo containerd config default | sudo tee /etc/containerd/config.toml

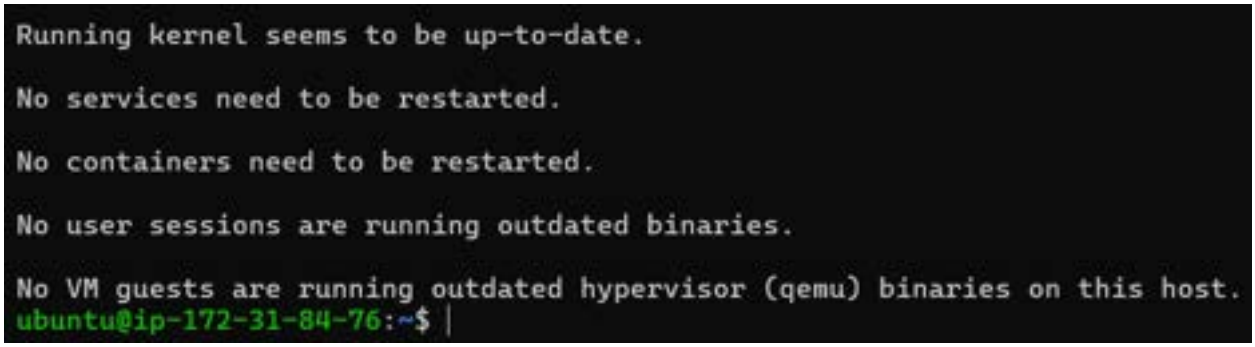
```
ubuntu@ip-172-31-90-144:~$ sudo mkdir -p /etc/containerd
sudo containerd config default | sudo tee /etc/containerd/config.toml
disabled_plugins = []
imports = []
oom_score = 0
plugin_dir = ""
required_plugins = []
root = "/var/lib/containerd"
state = "/run/containerd"
temp = ""
version = 2

[cgroup]
  path = ""

[debug]
  address = ""
  format = ""
  gid = 0
  level = ""
  uid = 0

[grpc]
  address = "/run/containerd/containerd.sock"
  gid = 0
  max_recv_message_size = 16777216
  max_send_message_size = 16777216
  tcp_address = ""
  tcp_tls_ca = ""
  tcp_tls_cert = ""
  tcp_tls_key = ""
  uid = 0

[metrics]
  address = ""
  grpc_histogram = false

[plugins]
```

j) sudo systemctl restart containerd
sudo systemctl enable containerd
sudo systemctl status containerd



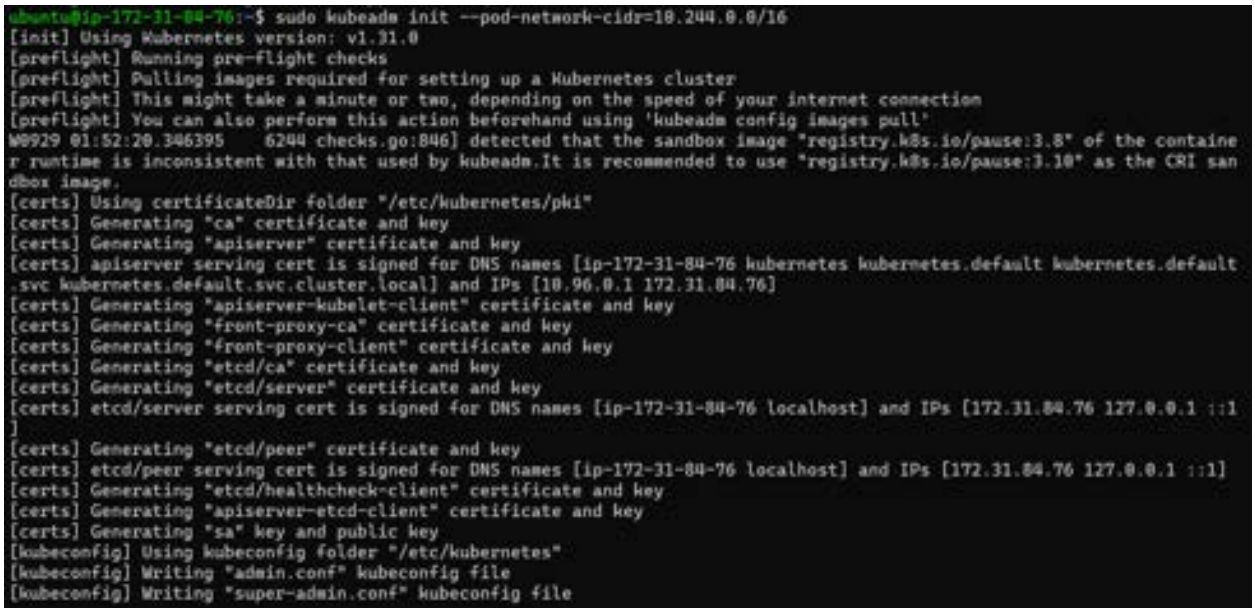
k) sudo apt-get install -y socat

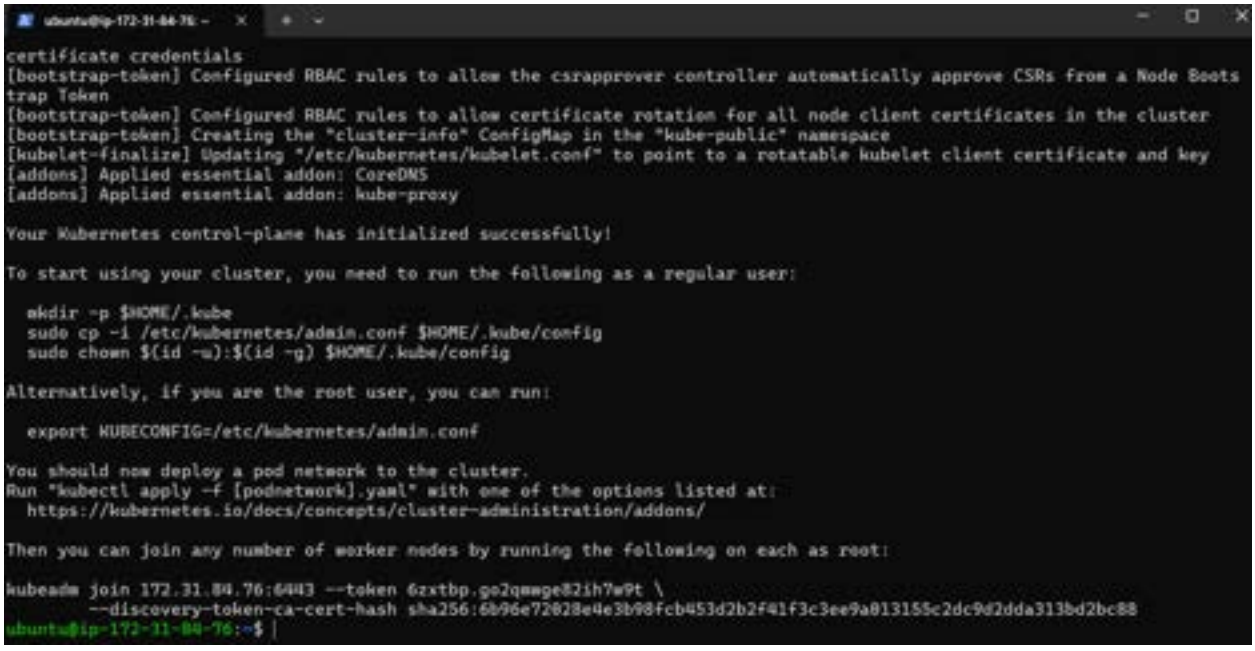




Step 6: Set up the Kubernetes cluster by running the following command exclusively on the master node:

sudo kubeadm init --pod-network-cidr=10.244.0.0/16. This command initializes the Kubernetes control plane and specifies the pod network range to be used within the cluster i.e kubeadm will initialize the kubernetes cluster and cidr is Classless Inter Domain Range which decides the range of network range of the pods





Run this command on master:

mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

These commands create a .kube directory in your home folder, copy the Kubernetes admin configuration file into it, and change the ownership of that configuration file to the current user to ensure proper access.

```
ubuntu@ip-172-31-84-76:~$ mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
ubuntu@ip-172-31-84-76:~$
```

Step 7: Now Run the command **kubectl get nodes** to see the nodes before executing Join command on nodes. kubectl is a command-line tool used to interact with and manage Kubernetes clusters. It allows users to deploy applications, inspect and manage cluster resources, and view logs, among other tasks, by sending commands to the Kubernetes API server.

```
ubuntu@ip-172-31-84-76:~$ kubectl get nodes
NAME               STATUS     ROLES           AGE     VERSION
ip-172-31-84-76    NotReady   control-plane   6m42s   v1.31.1
ubuntu@ip-172-31-84-76:~$
```

Step 8: Now Run the following command on Node 1 and Node 2 to Join to master.
For this from the kubeadm init command output for the master node copy and paste this on both the node

```
You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 172.31.84.76:6443 --token 6zxtbp.go2qmmge82ih7w9t \
        --discovery-token-ca-cert-hash sha256:6b96e72828e4e3b98fcb453d2b2f41f3c3ee9a813155c2dc9d2dda313bd2bc88
ubuntu@ip-172-31-84-76:~$ mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
ubuntu@ip-172-31-84-76:~$ kubectl get nodes
NAME               STATUS     ROLES           AGE     VERSION
ip-172-31-84-76    NotReady   control-plane   6m42s   v1.31.1
ubuntu@ip-172-31-84-76:~$
```

**<u>Copy and paste:</u> sudo kubeadm join 172.31.84.76:6443 --token 6zxtbp.go2qmwge82ih7w9t **
--discovery-token-ca-cert-hash sha256:6b96e72028e4e3b98fcb453d2b2f41f3c3ee9a013155c2dc9d2dda313bd2bc88

Worker Node 1:

```
ubuntu@ip-172-31-93-130:~$ sudo kubeadm join 172.31.84.76:6443 --token 6zxtbp.go2qmwge82ih7w9t \
        --discovery-token-ca-cert-hash sha256:6b96e72028e4e3b98fcb453d2b2f41f3c3ee9a013155c2dc9d2dda313bd2bc88
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-check] Waiting for a healthy kubelet at http://127.0.0.1:10248/healthz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 1.001865005s
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

Worker Node 2:

```
ubuntu@ip-172-31-93-235:~$ sudo kubeadm join 172.31.84.76:6443 --token 6zxtbp.go2qmwge82ih7w9t \
        --discovery-token-ca-cert-hash sha256:6b96e72028e4e3b98fcb453d2b2f41f3c3ee9a013155c2dc9d2dda313bd2bc88
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-check] Waiting for a healthy kubelet at http://127.0.0.1:10248/healthz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 1.000833351s
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

Step 9: Run the command **kubectl get nodes** to see the nodes after executing Join command on nodes.

```
ubuntu@ip-172-31-84-76:~$ kubectl get nodes
NAME               STATUS     ROLES           AGE     VERSION
ip-172-31-84-76    NotReady   control-plane   30m     v1.31.1
ip-172-31-93-130   NotReady   <none>          4m23s   v1.31.1
ip-172-31-93-235   NotReady   <none>          37s     v1.31.1
ubuntu@ip-172-31-84-76:~$
```

Step 10: Since Status is NotReady we have to add a network plugin. And also we have to give the name to the nodes.

kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml .This command applies the Calico network plugin configuration, which enables networking capabilities in the Kubernetes cluster .It enhances the effective communication between pods in network

```
ubuntu@ip-172-31-84-76:~$ kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml
poddisruptionbudget.policy/calico-kube-controllers created
serviceaccount/calico-kube-controllers created
serviceaccount/calico-node created
configmap/calico-config created
customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgppeers.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/blockaffinities.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/caliconodestatuses.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/clusterinformations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/felixconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworksets.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/hostendpoints.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamblocks.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamconfigs.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamhandles.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ippools.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipreservations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/kubecontrollersconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networksets.crd.projectcalico.org created
clusterrole.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrole.rbac.authorization.k8s.io/calico-node created
clusterrolebinding.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrolebinding.rbac.authorization.k8s.io/calico-node created
daemonset.apps/calico-node created
deployment.apps/calico-kube-controllers created
ubuntu@ip-172-31-84-76:~$
```

Now perform **sudo systemctl status kubelet**

```
ubuntu@ip-172-31-84-76:~$ sudo systemctl status kubelet
● kubelet.service - kubelet: The Kubernetes Node Agent
     Loaded: loaded (/usr/lib/systemd/system/kubelet.service; enabled; preset: enabled)
    Drop-In: /usr/lib/systemd/system/kubelet.service.d
             └─10-kubeadm.conf
     Active: active (running) since Sun 2024-09-29 01:52:38 UTC; 35min ago
       Docs: https://kubernetes.io/docs/
   Main PID: 6918 (kubelet)
      Tasks: 10 (limit: 4676)
     Memory: 33.6M (peak: 34.0M)
        CPU: 26.546s
     CGroup: /system.slice/kubelet.service
             └─6918 /usr/bin/kubelet --bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf --kubeconfig=/etc/ku

Sep 29 02:27:45 ip-172-31-84-76 kubelet[6918]:         : unknown
Sep 29 02:27:45 ip-172-31-84-76 kubelet[6918]:  > pod="kube-system/coredns-7c65d6cfc9-7nrwx" podUID="1b227dfc-524d-4b11
Sep 29 02:27:45 ip-172-31-84-76 kubelet[6918]: E0929 02:27:45.874236    6918 log.go:32] "StopPodSandbox from runtime se
Sep 29 02:27:45 ip-172-31-84-76 kubelet[6918]:         rpc error: code = Unknown desc = failed to stop container "1daf6
Sep 29 02:27:45 ip-172-31-84-76 kubelet[6918]:         : unknown
Sep 29 02:27:45 ip-172-31-84-76 kubelet[6918]:  > podSandboxID="8ce61ac60e9c40c62ac2ac734ef78143faf196ccf04c67427bc23a9
Sep 29 02:27:45 ip-172-31-84-76 kubelet[6918]: E0929 02:27:45.874271    6918 kuberuntime_manager.go:1479] "Failed to st
Sep 29 02:27:45 ip-172-31-84-76 kubelet[6918]: E0929 02:27:45.960413    6918 kubelet.go:1865] "KillPod failed" err="[fa
Sep 29 02:27:49 ip-172-31-84-76 kubelet[6918]: I0929 02:27:49.223548    6918 scope.go:117] "RemoveContainer" containerI
Sep 29 02:27:49 ip-172-31-84-76 kubelet[6918]: E0929 02:27:49.223664    6918 pod_workers.go:1301] "Error syncing pod, s
ubuntu@ip-172-31-84-76:~$
```

Now Run command **kubectl get nodes -o wide we can see Status is ready**.

```
ubuntu@ip-172-31-84-76:~$ kubectl get nodes -o wide
NAME              STATUS   ROLES           AGE     VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION   CONTAINER-RUNTIME
ip-172-31-84-76   Ready    control-plane   39m     v1.31.1   172.31.84.76    <none>        Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
ip-172-31-93-130  Ready    <none>          12m     v1.31.1   172.31.93.130   <none>        Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
ip-172-31-93-235  Ready    <none>          8m45s   v1.31.1   172.31.93.235   <none>        Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
ubuntu@ip-172-31-84-76:~$
```

Now to Rename run this command
Rename to Worker_Node1_41: kubectl label node ip-172-31-93-138 kubernetes.io/role=Worker_Node1_41
Rename to Worker_Node2_41 : kubectl label node ip-172-31-93-235 kubernetes.io/role=Worker_Node2_41

Make Sure to give Correct corresponding IP while renaming which can be seen from previous output also.

```
ubuntu@ip-172-31-84-76:~$ kubectl label node ip-172-31-93-130 kubernetes.io/role=Worker_Node1_41
node/ip-172-31-93-130 labeled
ubuntu@ip-172-31-84-76:~$ kubectl label node ip-172-31-93-235 kubernetes.io/role=Worker_Node2_41
node/ip-172-31-93-235 labeled
ubuntu@ip-172-31-84-76:~$ kubectl get nodes -o wide
NAME              STATUS   ROLES           AGE   VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION   CONTAINER-RUNTIME
ip-172-31-84-76   Ready    control-plane   43m   v1.31.1   172.31.84.76    <none>        Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
ip-172-31-93-130  Ready    Worker_Node1_41 16m   v1.31.1   172.31.93.130   <none>        Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
ip-172-31-93-235  Ready    Worker_Node2_41 12m   v1.31.1   172.31.93.235   <none>        Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
ubuntu@ip-172-31-84-76:~$
```

Step 11: Run command **kubectl get nodes -o wide** . And Hence we can see we have Successfully connected both the Worker Nodes to the Master Node.

```
ubuntu@ip-172-31-84-76:~$ kubectl get nodes -o wide
NAME              STATUS   ROLES           AGE   VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION   CONTAINER-RUNTIME
ip-172-31-84-76   Ready    control-plane   43m   v1.31.1   172.31.84.76    <none>        Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
ip-172-31-93-130  Ready    Worker_Node1_41 16m   v1.31.1   172.31.93.130   <none>        Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
ip-172-31-93-235  Ready    Worker_Node2_41 12m   v1.31.1   172.31.93.235   <none>        Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
ubuntu@ip-172-31-84-76:~$
```

CONCLUSION:

In this experiment, we established a Kubernetes cluster on AWS EC2 instances, comprising one master node and two worker nodes. After setting up Docker and the required Kubernetes tools (kubelet, kubeadm, kubectl, and containerd) on all nodes, we initialized the master and incorporated the worker nodes into the cluster. Initially, the nodes displayed a NotReady status, which we corrected by installing the Calico network plugin. We also tagged the nodes with their respective roles (control-plane and worker). In the end, all nodes changed to the Ready state, showcasing that we successfully set up and managed the Kubernetes cluster.

Name: Shivam.R.Prajapati Div: D15C Roll No:41 Academic Year: 2024-25

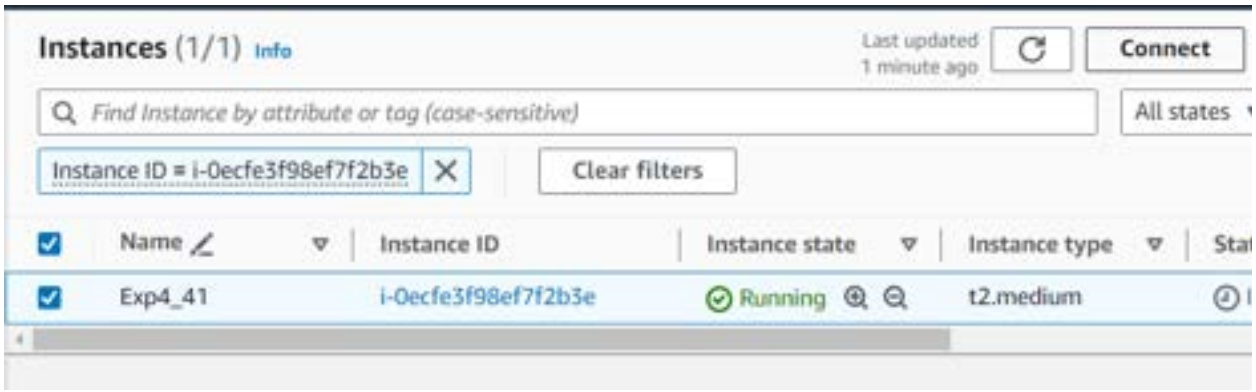
Experiment No: 4

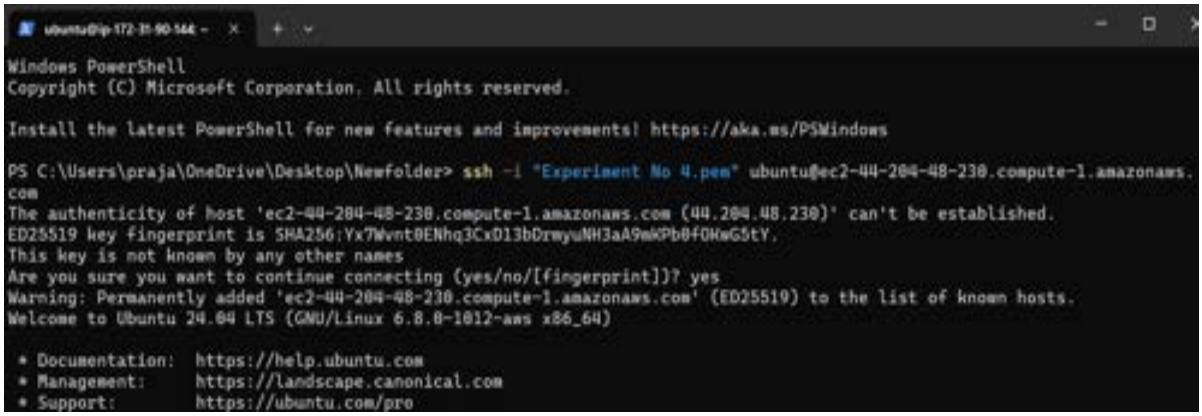
AIM: To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deploy Your First Kubernetes Application.

STEPS:

Step 1: Access your AWS Academy or personal account and initiate the launch of a new EC2 instance. Select **Ubuntu as the Amazon Machine Image (AMI)** and choose **t2.medium** for the instance type. Generate an RSA key with a .pem file extension, and relocate the downloaded key to the designated folder. **Note:** Since at least 2 CPUs are needed, ensure you select t2.medium

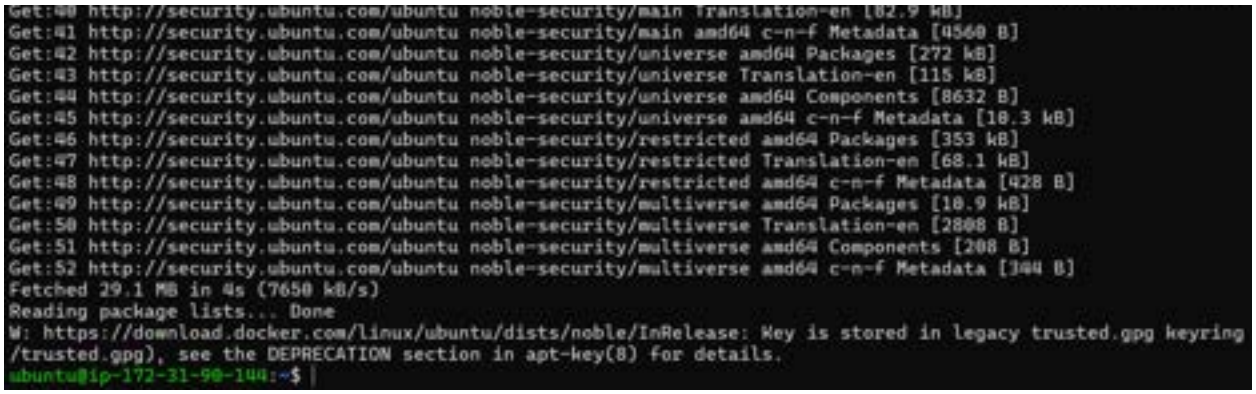
Step 2: Once the instance is created, click on the option to connect to the instance and go to the SSH client section.Now open the folder in the terminal where our .pem key is stored and paste the Example command (starting with ssh -i …..) in the terminal.





Step 3: Execute the following commands to **install and set up Docker.**

a) **curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -**
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/trusted.gpg.d/docker.gpg > /dev/null
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"



b) **sudo apt-get update**
sudo apt-get install -y docker-ce

```
ubuntu@ip-172-31-90-144:~$ sudo apt-get update
sudo apt-get install -y docker-ce
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored
/trusted.gpg), see the DEPRECATION section in apt-key(8) for details.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

```
Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-90-144:~$
```

c) **sudo mkdir -p /etc/docker**
cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOF

```
ubuntu@ip-172-31-90-144:~$ sudo mkdir -p /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
ubuntu@ip-172-31-90-144:~$
```

d) **sudo systemctl enable docker**
sudo systemctl daemon-reload
sudo systemctl restart docker

```
ubuntu@ip-172-31-90-144:~$ sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
ubuntu@ip-172-31-90-144:~$
```

Step 4: Execute the following command to **install Kubernetes**.

a) **sudo rm -f /etc/apt/sources.list.d/kubernetes.list**

```
ubuntu@ip-172-31-90-144:~$ sudo rm -f /etc/apt/sources.list.d/kubernetes.list
```

b) **sudo apt-get update**

```
ubuntu@ip-172-31-90-144:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:5 https://download.docker.com/linux/ubuntu noble InRelease
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored
ection in apt-key(8) for details.
```

c) **sudo apt-get install -y apt-transport-https ca-certificates curl gpg**

```
ubuntu@ip-172-31-90-144:~$ sudo apt-get install -y apt-transport-https ca-certificates curl gpg
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20240203).
ca-certificates set to manually installed.
gpg is already the newest version (2.4.4-2ubuntu17).
gpg set to manually installed.
The following NEW packages will be installed:
```

```
Fetched 904 kB in 0s (29.7 MB/s)
Selecting previously unselected package apt-transport-https.
(Reading database ... 68007 files and directories currently installed.)
Preparing to unpack .../apt-transport-https_2.7.14build2_all.deb ...
Unpacking apt-transport-https (2.7.14build2) ...
Preparing to unpack .../curl_8.5.0-2ubuntu10.4_amd64.deb ...
Unpacking curl (8.5.0-2ubuntu10.4) over (8.5.0-2ubuntu10.1) ...
Preparing to unpack .../libcurl4t64_8.5.0-2ubuntu10.4_amd64.deb ...
Unpacking libcurl4t64:amd64 (8.5.0-2ubuntu10.4) over (8.5.0-2ubuntu10.1) ...
Preparing to unpack .../libcurl3t64-gnutls_8.5.0-2ubuntu10.4_amd64.deb ...
Unpacking libcurl3t64-gnutls:amd64 (8.5.0-2ubuntu10.4) over (8.5.0-2ubuntu10.1) ...
Setting up apt-transport-https (2.7.14build2) ...
Setting up libcurl4t64:amd64 (8.5.0-2ubuntu10.4) ...
Setting up libcurl3t64-gnutls:amd64 (8.5.0-2ubuntu10.4) ...
Setting up curl (8.5.0-2ubuntu10.4) ...
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for libc-bin (2.39-0ubuntu8.2) ...
Scanning processes...
Scanning candidates...
Scanning linux images...

Running kernel seems to be up-to-date.

Restarting services...
 systemctl restart packagekit.service

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
```

d) sudo mkdir -p -m 755 /etc/apt/keyrings

```
ubuntu@ip-172-31-90-144:~$ sudo mkdir -p -m 755 /etc/apt/keyrings
```

e) curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

```
ubuntu@ip-172-31-90-144:~$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key
ng.gpg
File '/etc/apt/keyrings/kubernetes-apt-keyring.gpg' exists. Overwrite? (y/N) y
```

f) echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

```
ubuntu@ip-172-31-90-144:~$ echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.
c/apt/sources.list.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
```

g) sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl

```
ubuntu@ip-172-31-90-144:~$ sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 https://download.docker.com/linux/ubuntu noble InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  InRelease [1186 B]
Hit:6 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:7 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  Packages [4865 B]
Fetched 6051 B in 1s (9987 B/s)
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.
ection in apt-key(8) for details.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  conntrack cri-tools kubernetes-cni
The following NEW packages will be installed:
  conntrack cri-tools kubeadm kubectl kubelet kubernetes-cni
0 upgraded, 6 newly installed, 0 to remove and 140 not upgraded.
Need to get 87.4 MB of archives.
After this operation, 314 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 conntrack amd64 1:1.4.8-1ubuntu1 [37.9 kB]
Get:2 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  cri-tools 1.31.1-1.1 [15.7 MB]
Get:3 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  kubeadm 1.31.1-1.1 [11.4 MB]
Get:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  kubectl 1.31.1-1.1 [11.2 MB]
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  kubernetes-cni 1.5.1-1.1 [33.9 MB]
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  kubelet 1.31.1-1.1 [15.2 MB]
Fetched 87.4 MB in 1s (70.6 MB/s)
Selecting previously unselected package conntrack.
```

```
Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
```

h) sudo systemctl enable --now kubelet

```
ubuntu@ip-172-31-90-144:~$ sudo systemctl enable --now kubelet
ubuntu@ip-172-31-90-144:~$
```

i) sudo kubeadm init –pod-network-cidr=10.244.0.0/16

```
ubuntu@ip-172-31-90-144:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
W0928 04:27:54.973362    7321 checks.go:1080] [preflight] WARNING: Couldn't create
 new CRI runtime service: validate service connection: validate CRI v1 runtime API
e = Unimplemented desc = unknown service runtime.v1.RuntimeService
        [WARNING FileExisting-socat]: socat not found in system path
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your intern
[preflight] You can also perform this action beforehand using 'kubeadm config imag
error execution phase preflight: [preflight] Some fatal errors occurred:
failed to create new CRI runtime service: validate service connection: validate CR
: rpc error: code = Unimplemented desc = unknown service runtime.v1.RuntimeService
 with '--ignore-preflight-errors=...'
To see the stack trace of this error execute with --v=5 or higher
```

Here we got an error so to resolve this we use NEXT FOLLOWING COMMAND

j) sudo apt-get install -y containerd

```
ubuntu@ip-172-31-90-144:~$ sudo apt-get install -y containerd
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer re
  docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-co
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  runc
The following packages will be REMOVED:
  containerd.io docker-ce
The following NEW packages will be installed:
  containerd runc
0 upgraded, 2 newly installed, 2 to remove and 140 not upgraded.
Need to get 47.2 MB of archives.
After this operation, 53.1 MB disk space will be freed.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main
Fetched 47.2 MB in 1s (72.8 MB/s)
(Reading database ... 68068 files and directories currently installed.)
Removing docker-ce (5:27.3.1-1~ubuntu.24.04~noble) ...
Removing containerd.io (1.7.22-1) ...
Selecting previously unselected package runc.
(Reading database ... 68048 files and directories currently installed.)
Preparing to unpack .../runc_1.1.12-0ubuntu3.1_amd64.deb ...
Unpacking runc (1.1.12-0ubuntu3.1) ...
Selecting previously unselected package containerd.
Preparing to unpack .../containerd_1.7.12-0ubuntu4.1_amd64.deb ...
Unpacking containerd (1.7.12-0ubuntu4.1) ...
Setting up runc (1.1.12-0ubuntu3.1) ...
Setting up containerd (1.7.12-0ubuntu4.1) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.
```

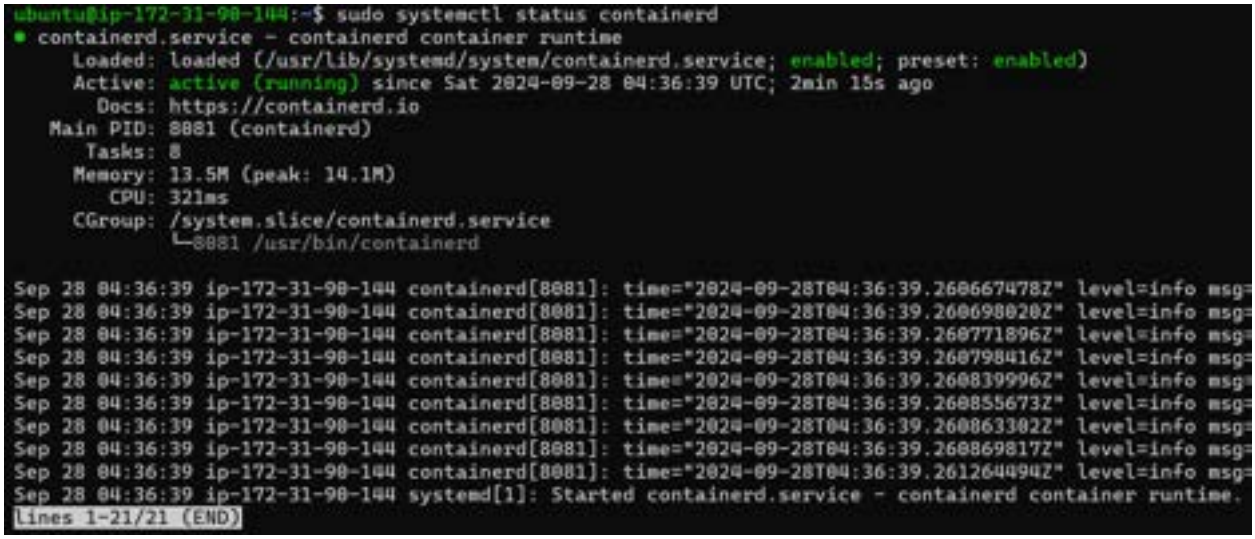
k) sudo mkdir -p /etc/containerd
sudo containerd config default | sudo tee /etc/containerd/config.toml

```
ubuntu@ip-172-31-90-144:~$ sudo mkdir -p /etc/containerd
sudo containerd config default | sudo tee /etc/containerd/config.toml
disabled_plugins = []
imports = []
oom_score = 0
plugin_dir = ""
required_plugins = []
root = "/var/lib/containerd"
state = "/run/containerd"
temp = ""
version = 2

[cgroup]
  path = ""

[debug]
  address = ""
  format = ""
  gid = 0
  level = ""
  uid = 0

[grpc]
  address = "/run/containerd/containerd.sock"
  gid = 0
  max_recv_message_size = 16777216
  max_send_message_size = 16777216
  tcp_address = ""
  tcp_tls_ca = ""
  tcp_tls_cert = ""
  tcp_tls_key = ""
  uid = 0

[metrics]
  address = ""
  grpc_histogram = false

[plugins]
```

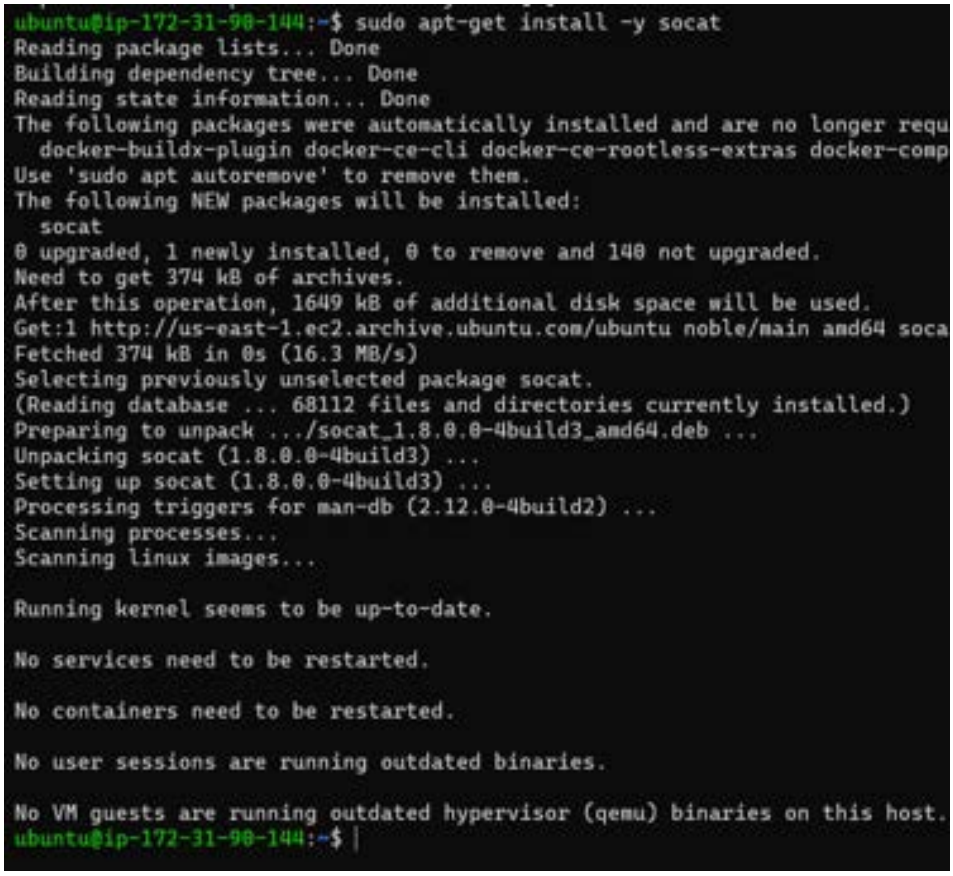
l) sudo systemctl restart containerd
sudo systemctl enable containerd

```
ubuntu@ip-172-31-90-144:~$ sudo systemctl restart containerd
sudo systemctl enable containerd
ubuntu@ip-172-31-90-144:~$
```

m) sudo systemctl status containerd



n) sudo apt-get install -y socat



Step 5: Initialize the **Kubecluster**.

sudo kubeadm init --pod-network-cidr=10.244.0.0/16

```
ubuntu@ip-172-31-90-144:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet
[preflight] You can also perform this action beforehand using 'kubeadm config images
W0928 04:46:10.270292    8652 checks.go:846] detected that the sandbox image "regist
used by kubeadm.It is recommended to use "registry.k8s.io/pause:3.10" as the CRI san
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [ip-172-31-90-144 kubernetes
.local] and IPs [10.96.0.1 172.31.90.144]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [ip-172-31-90-144 localhost
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [ip-172-31-90-144 localhost]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "super-admin.conf" kubeconfig file
[kubeconfig] Writing "kubelet.conf" kubeconfig file
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
[kubeconfig] Writing "scheduler.conf" kubeconfig file
[etcd] Creating static Pod manifest for local etcd in "/etc/kubernetes/manifests"
[control-plane] Using manifest folder "/etc/kubernetes/manifests"
[control-plane] Creating static Pod manifest for "kube-apiserver"
[control-plane] Creating static Pod manifest for "kube-controller-manager"
[control-plane] Creating static Pod manifest for "kube-scheduler"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubele
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Starting the kubelet
```

```
Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 172.31.90.144:6443 --token 75sp2s.4moocvobfpkimn9q \
        --discovery-token-ca-cert-hash sha256:42855a37f6c6446da3abb79740e05317a4aadc79d228cf7df4339bd90fd6f19d
ubuntu@ip-172-31-90-144:~$
```

Step 6: Copy the mkdir and chown commands from the top and execute them.
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

```
ubuntu@ip-172-31-90-144:~$ mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
ubuntu@ip-172-31-90-144:~$
```

Step 7: Add a common networking plugin called **flannel** as mentioned in the code. Flannel is a simple and widely used networking plugin for Kubernetes that facilitates pod-to-pod communication across a cluster.
kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

```
ubuntu@ip-172-31-90-144:~$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
namespace/kube-flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
ubuntu@ip-172-31-90-144:~$
```

Step 8: With the cluster now running, we can deploy our **Nginx server** onto it. Run the following command to apply the deployment configuration and set up the Nginx deployment.
kubectl apply -f https://k8s.io/examples/application/deployment.yaml

```
ubuntu@ip-172-31-90-144:~$ kubectl apply -f https://k8s.io/examples/application/deployment.yaml
deployment.apps/nginx-deployment created
ubuntu@ip-172-31-90-144:~$
```

kubectl get pods

```
ubuntu@ip-172-31-90-144:~$ kubectl get pods
NAME                               READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-6r7bm   0/1     Pending   0          76s
nginx-deployment-d556bf558-bv4vz   0/1     Pending   0          76s
ubuntu@ip-172-31-90-144:~$
```

```
ubuntu@ip-172-31-90-144:~$ POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
ubuntu@ip-172-31-90-144:~$
```

POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
kubectl port-forward $POD_NAME 8080:80

```
ubuntu@ip-172-31-90-144:~$ POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
ubuntu@ip-172-31-90-144:~$
```

```
ubuntu@ip-172-31-90-144:~$ kubectl port-forward $POD_NAME 8080:80
error: unable to forward port because pod is not running. Current status=Pending
ubuntu@ip-172-31-90-144:~$
```

Note: If you encounter an error where the pod status shows as "**Pending**," follow these steps. Run the commands below to resolve the issue, and then re-run the previous two commands.

kubectl taint nodes --all
node-role.kubernetes.io/control-plane-node/ip-172-31-20-171 untainted

```
ubuntu@ip-172-31-90-144:~$ kubectl taint nodes --all node-role.kubernetes.io/control-plane-
node/ip-172-31-90-144 untainted
ubuntu@ip-172-31-90-144:~$
```

kubectl get nodes

```
ubuntu@ip-172-31-90-144:~$ kubectl get nodes
NAME               STATUS   ROLES           AGE   VERSION
ip-172-31-90-144   Ready    control-plane   40m   v1.31.1
ubuntu@ip-172-31-90-144:~$
```

kubectl get pods

```
ubuntu@ip-172-31-90-144:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-6r7bm    1/1     Running   0          27m
nginx-deployment-d556bf558-bv4vz    1/1     Running   0          27m
ubuntu@ip-172-31-90-144:~$
```

POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
kubectl port-forward $POD_NAME 8080:80

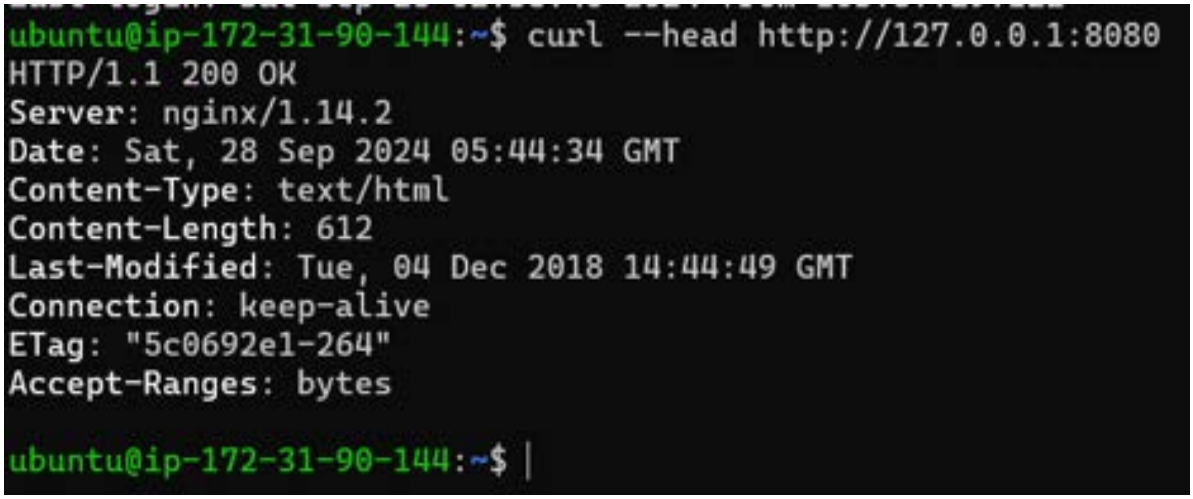
```
ubuntu@ip-172-31-90-144:~$ POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
ubuntu@ip-172-31-90-144:~$
```

```
ubuntu@ip-172-31-90-144:~$ POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
kubectl port-forward $POD_NAME 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
```

Step 9 : Check if your deployment is working. Open a new terminal and connect to your EC2 instance using SSH. Then, run the curl command to see if the Nginx server is up and running.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\praja\OneDrive\Desktop\Newfolder> ssh -i "Experiment No 4.pem" ubuntu@ec2-44-204-48-230.compute-1.amazonaws
com
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Sat Sep 28 05:42:45 UTC 2024

  System load:  0.08              Processes:             153
  Usage of /:   55.6% of 6.71GB   Users logged in:       1
  Memory usage: 19%               IPv4 address for enX0: 172.31.90.144
  Swap usage:   0%

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

   https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

142 updates can be applied immediately.
38 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable
```

curl --head http://127.0.0.1:8080



After handling connecton

If you get a **200 OK** response and see the Nginx server name, your deployment was successful.

CONCLUSION:

In this experiment, we successfully installed Kubernetes on an EC2 instance and deployed an Nginx server using various Kubectl commands. During the process, we faced two main problems. The first issue was that the Kubernetes pod was stuck in a pending state. We fixed this by removing the control-plane taint with the command **kubectl taint nodes --all**, which allowed the pod to schedule correctly. The second problem was a missing containerd runtime, which we resolved by installing and starting the containerd service. To make sure our Kubernetes setup had enough resources, we used a t2.medium EC2 instance with 2 virtual CPUs. Overall, these steps led to a successful deployment of the Nginx server, showing how effective Kubernetes can be in a cloud environment.

Name: Shivam.R.Prajapati Div: D15C Roll No:41 Academic Year: 2024-25

Experiment No: 5

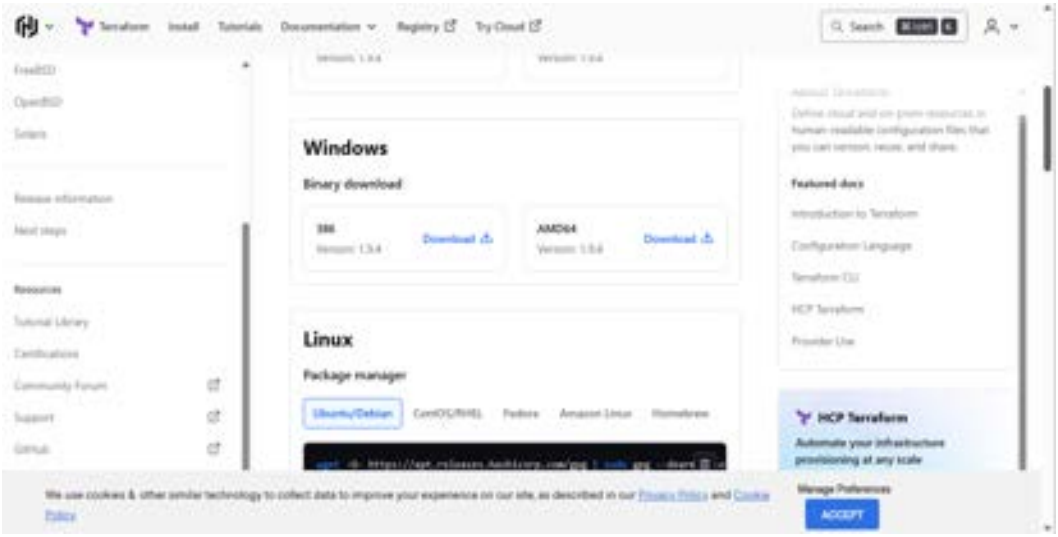
AIM: To understand terraform lifecycle, core concepts/terminologies and install it on a Window/Linux Machine.

Installation and Configuration Steps of Terraform in Windows:

Step 1: Download terraform

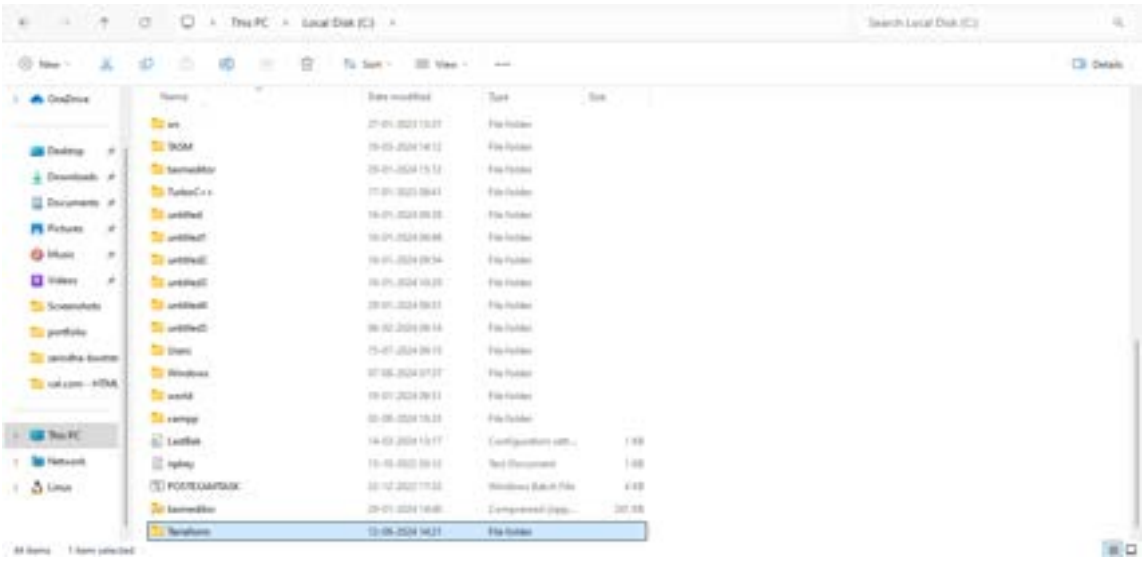
1) To install Terraform, First Download the Terraform Cli Utility for windows from terraforms official website https://www.terraform.io/downloads.html

2)Select the Operating System Windows followed by either 32 bit or 64 bit based on your OS type.

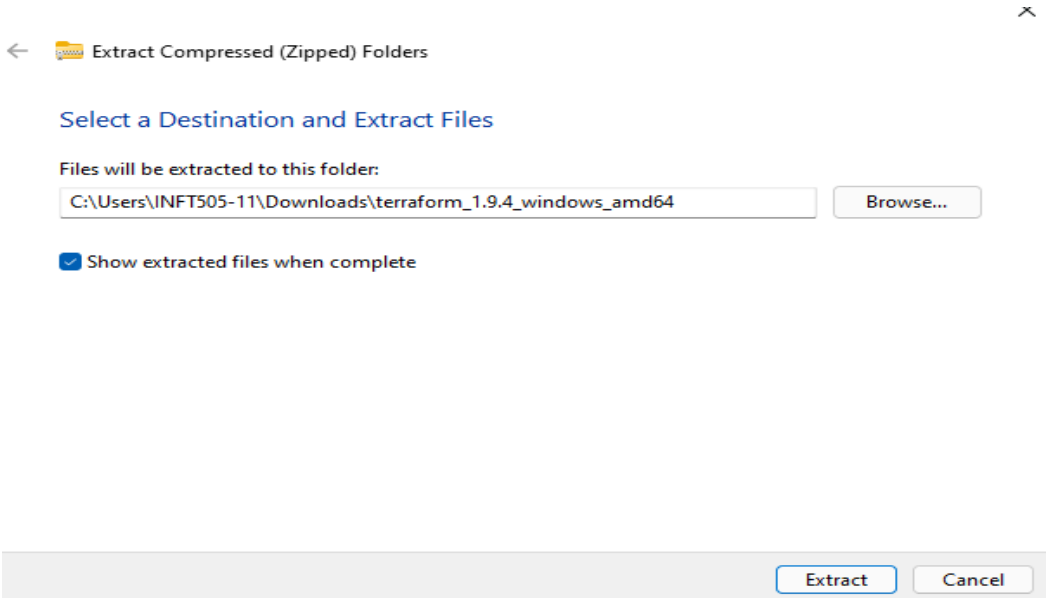


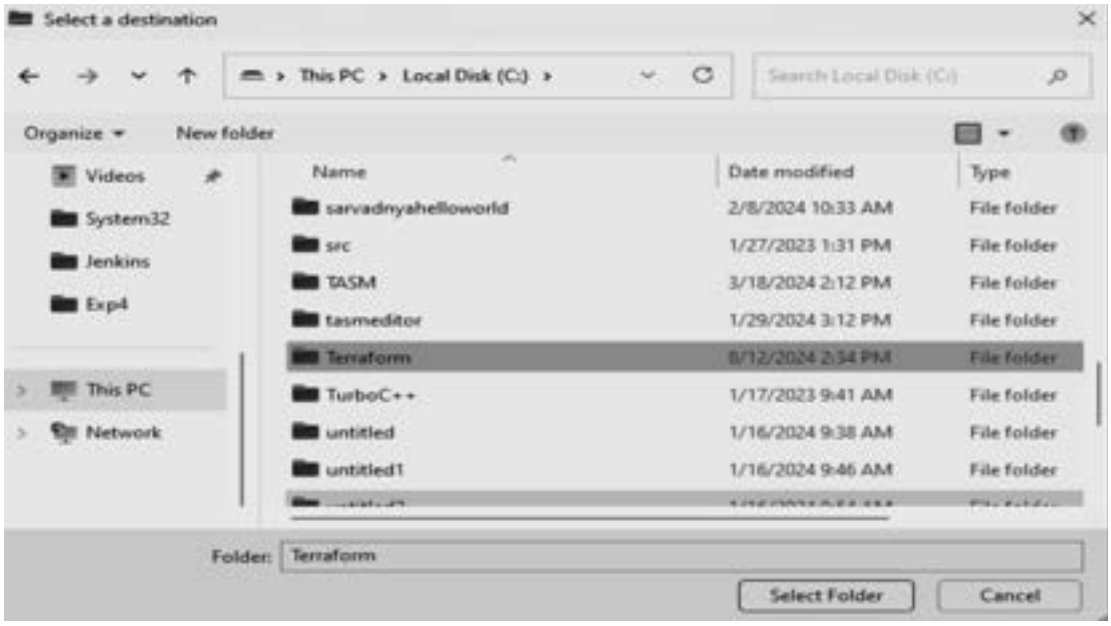
Step 2: Extract the downloaded setup file Terraform.exe in C:\Terraform Directory

1) Go to file manager -> Click on This PC -> move to C drive and create a new folder for example named terraform

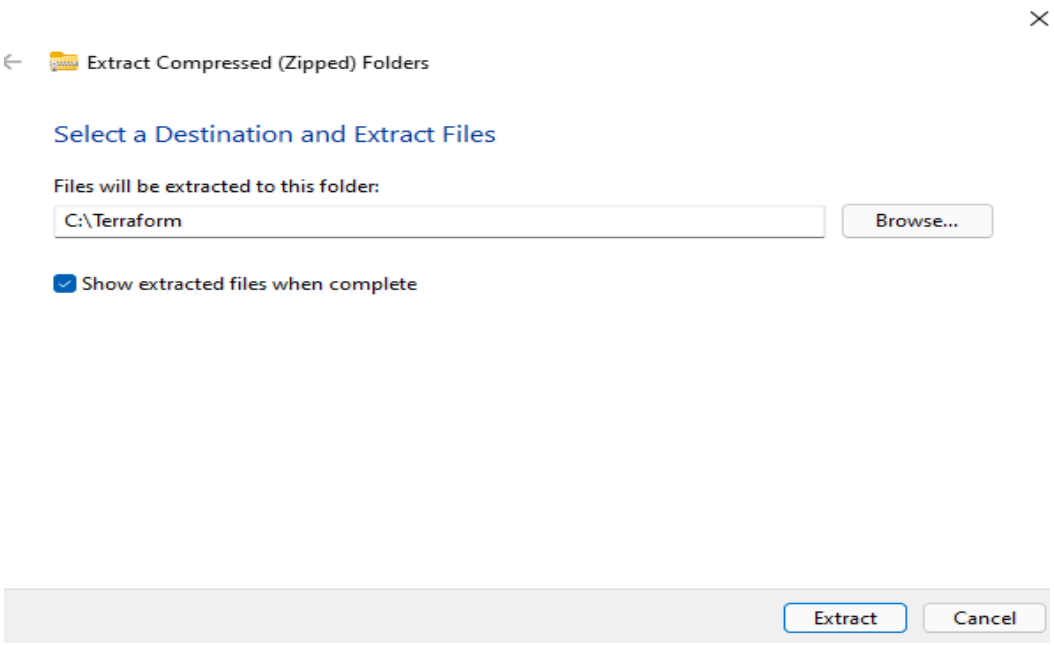


2) Got to location where main file is downloaded , right click on the file name and click on extract all while clicking on extract all it displays the location of the file to be saved , after extraction initially the destination of file to be extracted is not the folder which we have created earlier in previous steps , so to get the exact folder click on Browse and select it.

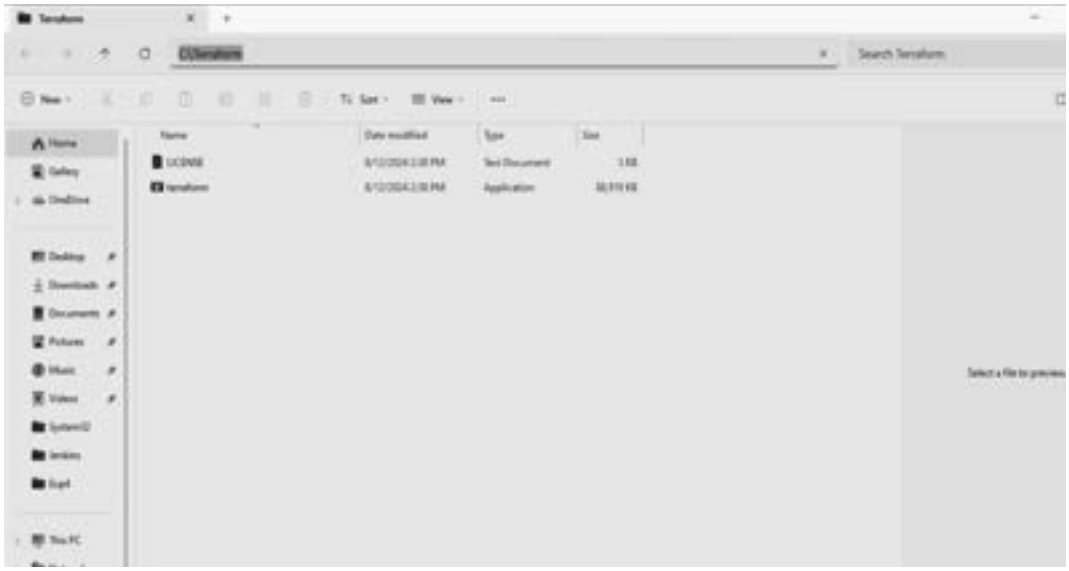




3) Click on extract Button.

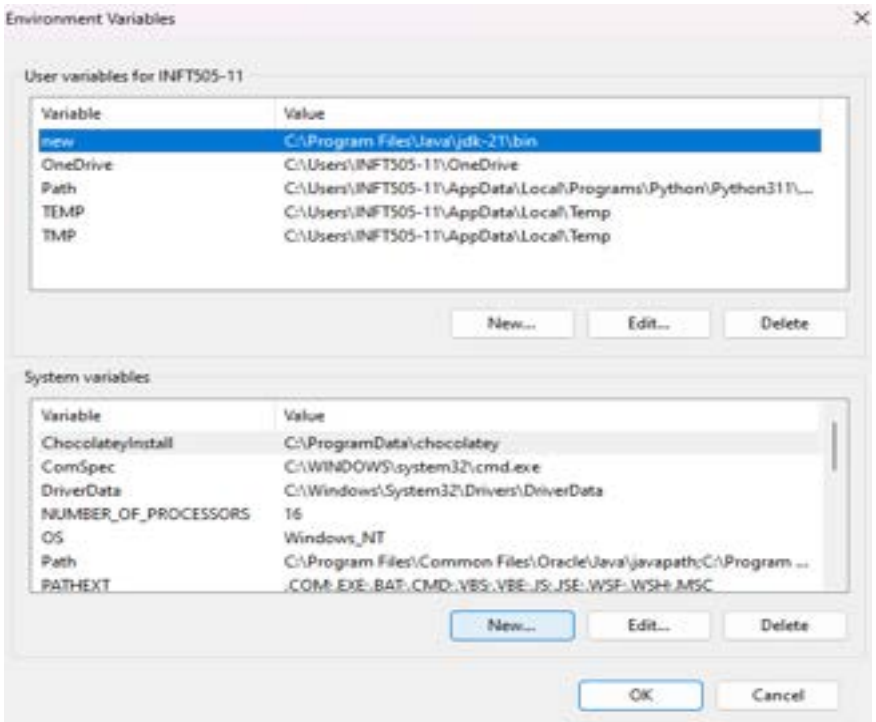


4) After extracting it will redirect to file manager copy the path which will be used for the further steps

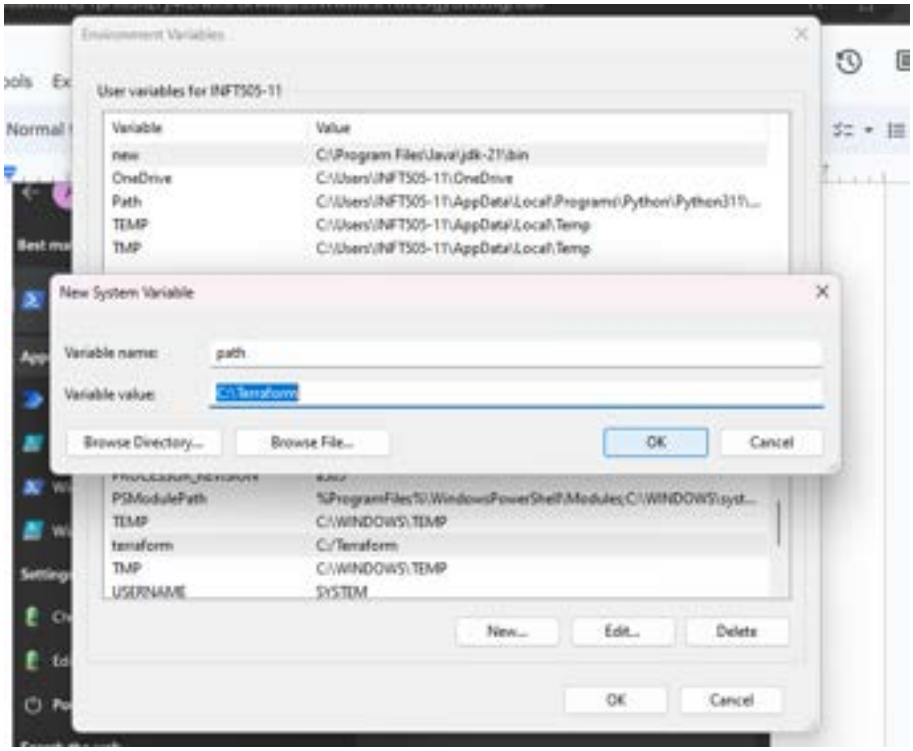


Step 3: Set the System path for Terraform in Environment Variables

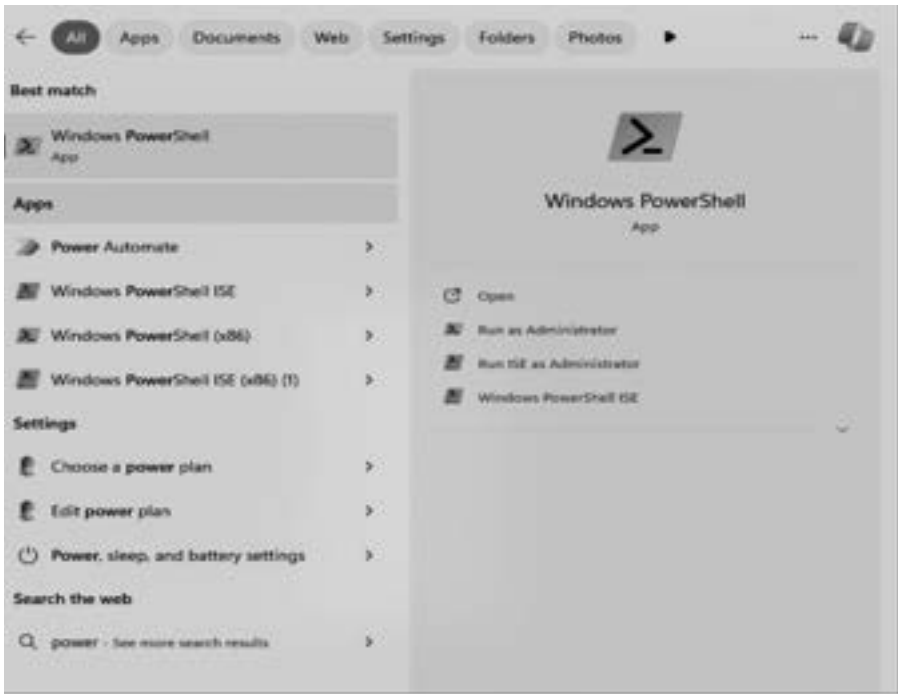
1) Search Environment Variable from search area in the Taskbar then Click on New under System Variables



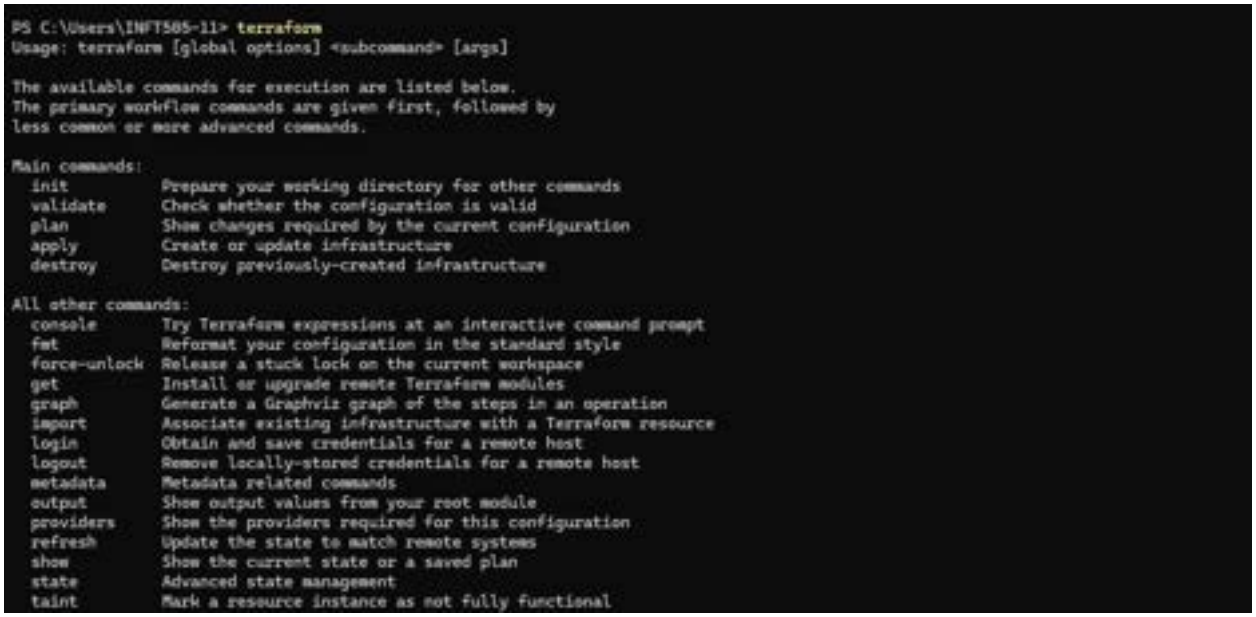
2) Set the Variable name as path and paste the location of the folder which we copied in previous step after extraction process as the variable value .Click on Ok



Step 4: Open PowerShell with Admin Access



Step 5 : Open Terraform in PowerShell and check its functionality



If the case of any Errors, then please recheck or set the path of Terraform in Environment variable again.

CONCLUSION:

So This experiment confirms that the terraform can be installed correctly on a local machine by following the standard installation steps, By downloading the appropriate package and verifying The installation ensures that the Terraform is set up and operating correctly.This experiment underscores the importance of each installation step to avoid the potential issues in using Terraform.

Name: Shivam.R.Prajapati Div: D15C Roll No:41 Academic Year: 2024-25

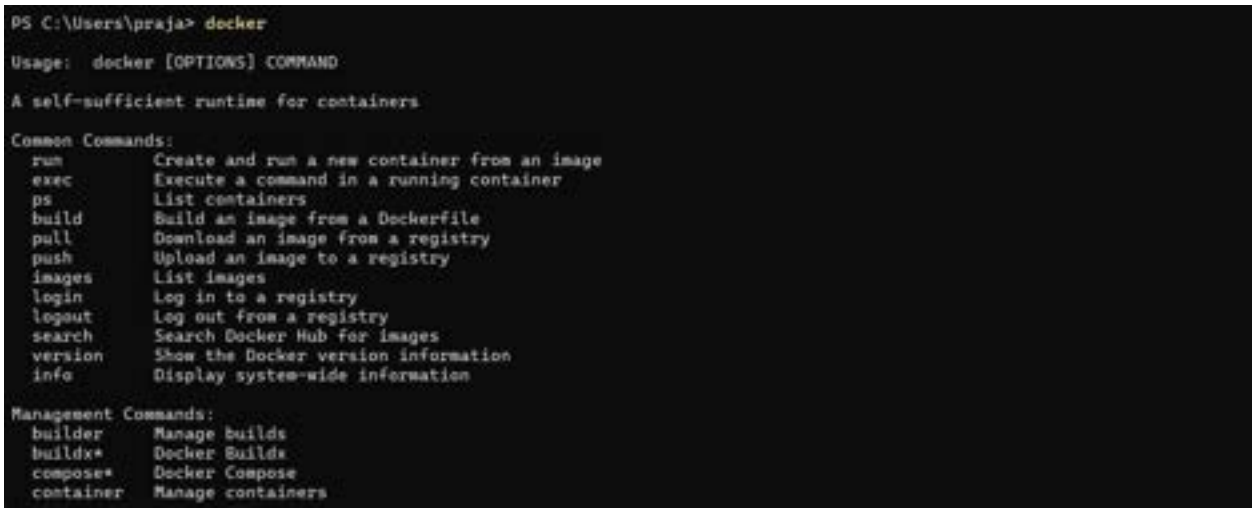
Experiment No: 6

AIM: To Build, change, and destroy AWS / GCP /Microsoft Azure/ DigitalOcean infrastructure Using Terraform. (S3 bucket or Docker) fdp.

A. Creating docker image using terraform

Step 1: In this experiment ,we need to install docker on our local Machine. Go to https://www.docker.com/ and download the file according to the OS you have.

Open the file and start the installation. After Installation, open your terminal and run 'docker' command. If this is your output, then docker is installed successfully .

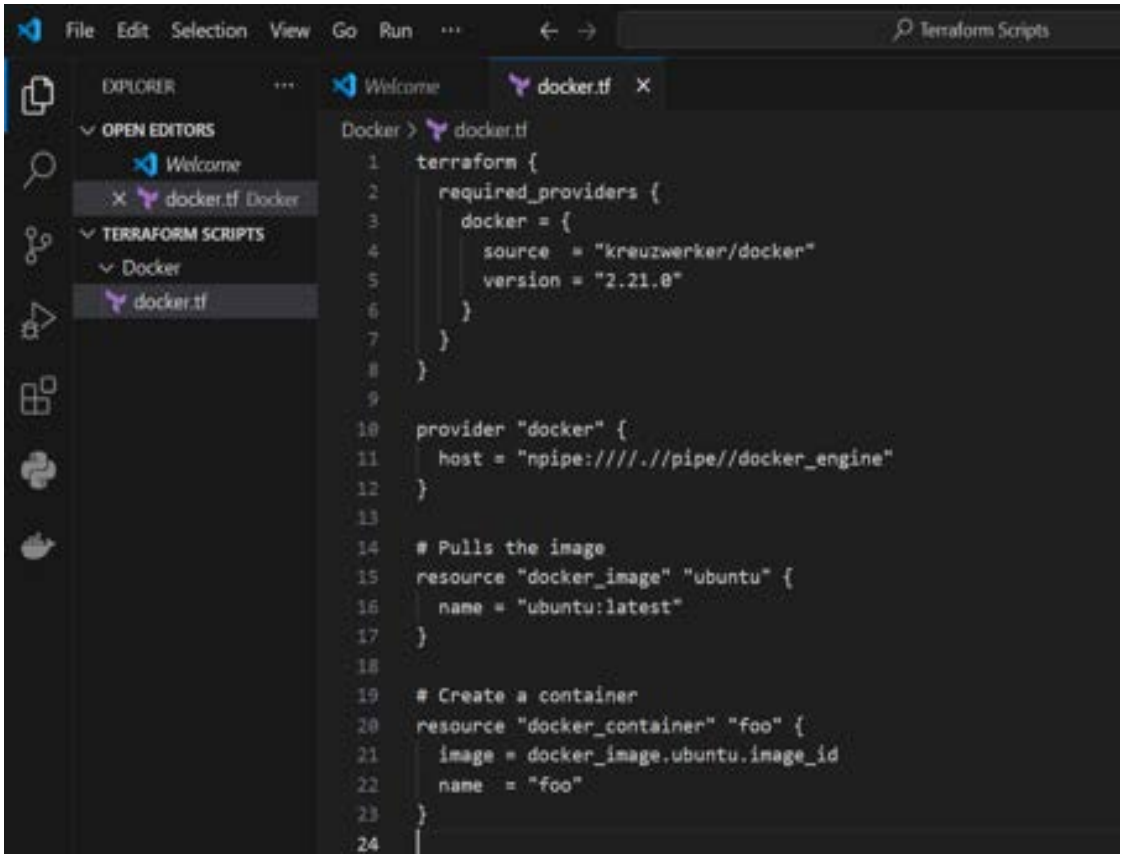


Step 2: Create a Folder named 'Docker' in the 'TerraformScripts' folder. Then create a new docker.tf file using Atom editor if you are using linux or else use VS code in windows and write the following contents into it to create a Ubuntu Linux container.

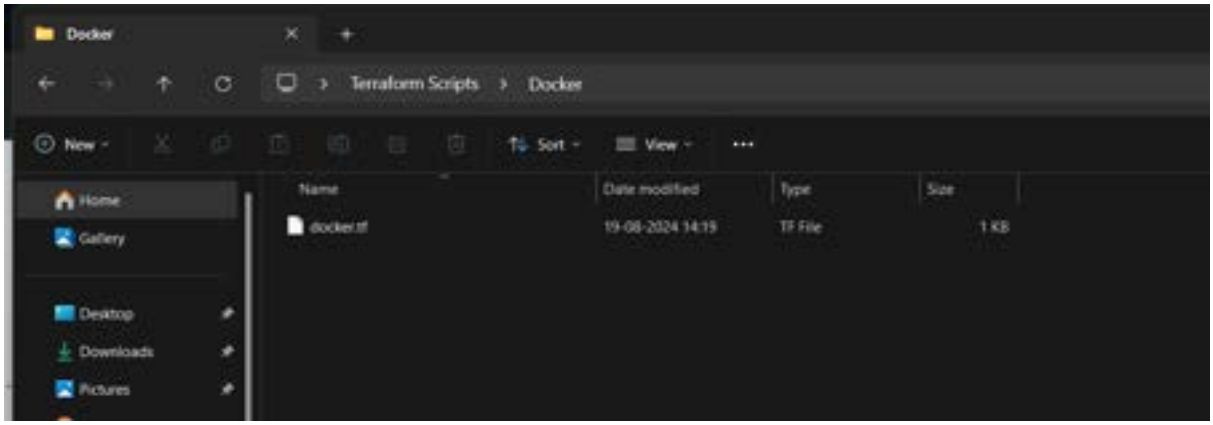
Script:

```
terraform
  { required_providers
  {docker = {
```

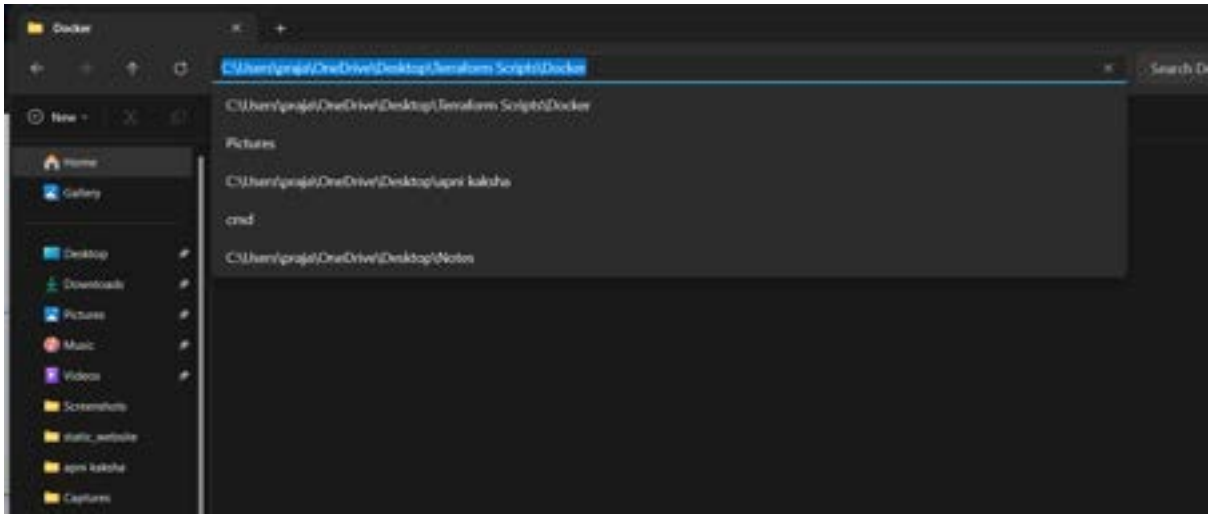
```
      source =
     "kreuzwerker/docker"
     version = "2.21.0"
    }
  }
}

        provider "docker" {
          host =
         "npipe:////.//pipe//docker_eng
         ine" }

        # Pulls the image
        resource
           "docker_image"
           "ubuntu" {name =
           "ubuntu:latest"
         }

        # Create a container
        resource
           "docker_container"
           "foo" { image =
          docker_image.ubuntu.imag
          e_idname = "foo"
         }
```

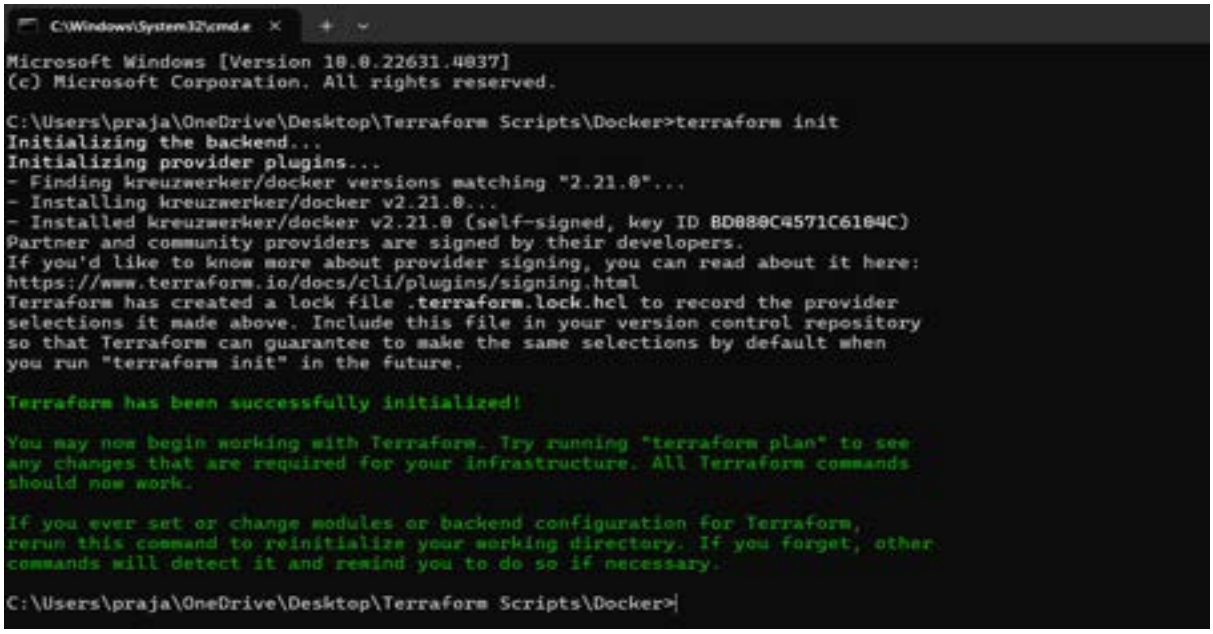


Step 3: Execute Terraform Init command to initialize the resources .Now for this go to file manager ->Open Terraform script folder then open Docker folder ->Click on the path of these folder and type cmd this will open Command Prompt window to initialize it in our directory.



Name: Shivam.R.Prajapati Div: D15C Roll No:41 Academic Year: 2024-25





Step 4: Execute Terraform plan to see the available resources.

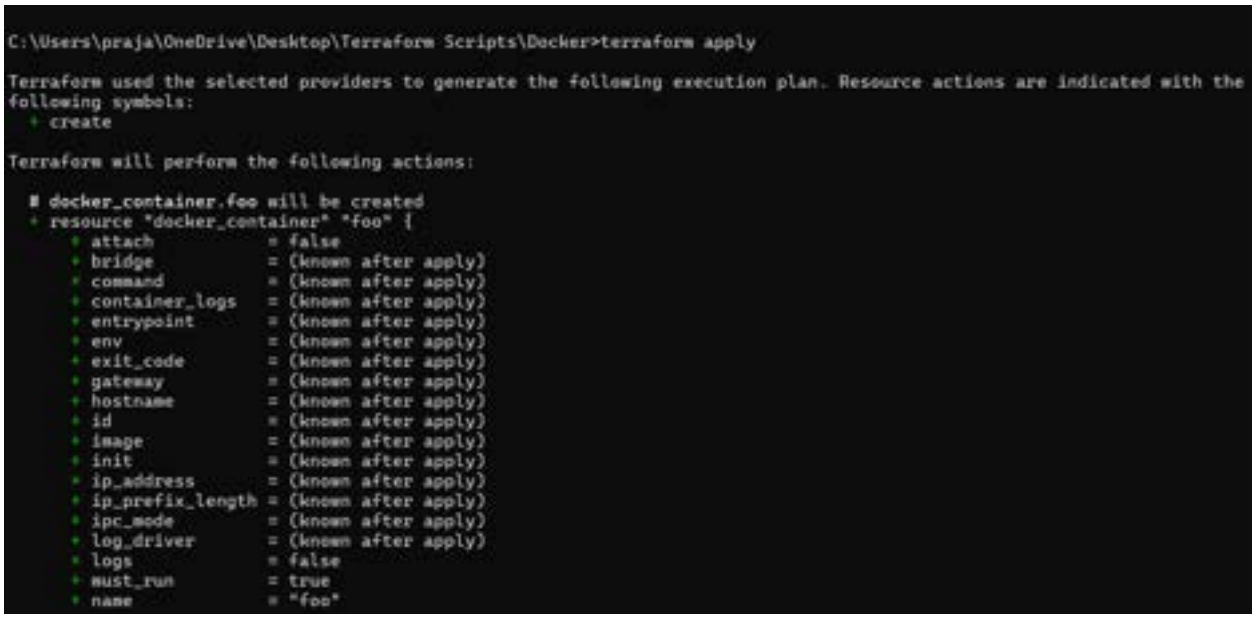
This command helps to get an execution plan and lets us overview changes that are going to happen in your infrastructure.

Name: Shivam.R.Prajapati Div: D15C Roll No:41 Academic Year: 2024-25

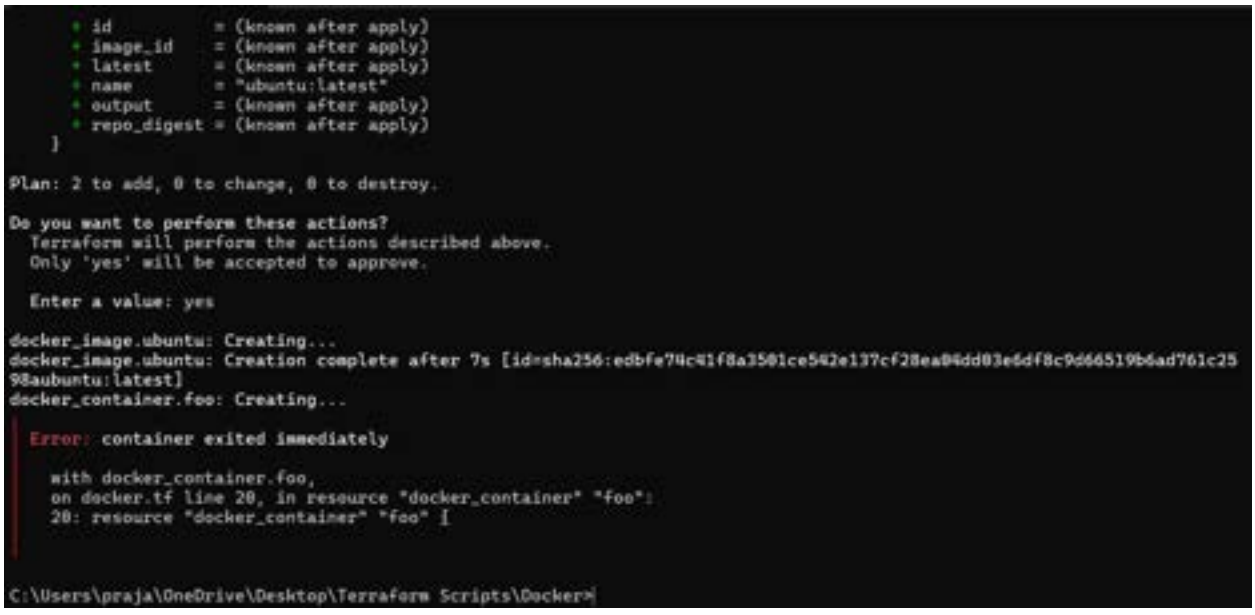
```
C:\Users\praja\OneDrive\Desktop\Terraform Scripts\Docker>terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
  + create

Terraform will perform the following actions:

  # docker_container.foo will be created
  + resource "docker_container" "foo" {
      + attach           = false
      + bridge           = (known after apply)
      + command          = (known after apply)
      + container_logs   = (known after apply)
      + entrypoint       = (known after apply)
      + env              = (known after apply)
      + exit_code        = (known after apply)
      + gateway          = (known after apply)
      + hostname         = (known after apply)
      + id               = (known after apply)
      + image            = (known after apply)
      + init             = (known after apply)
      + ip_address       = (known after apply)
      + ip_prefix_length = (known after apply)
      + ipc_mode         = (known after apply)
      + log_driver       = (known after apply)
      + logs             = false
      + must_run         = true
```

```
      + shm_size         = (known after apply)
      + start            = true
      + stdin_open       = false
      + stop_signal      = (known after apply)
      + stop_timeout     = (known after apply)
      + tty              = false

      + healthcheck (known after apply)

      + labels (known after apply)
    }

  # docker_image.ubuntu will be created
  + resource "docker_image" "ubuntu" {
      + id          = (known after apply)
      + image_id    = (known after apply)
      + latest      = (known after apply)
      + name        = "ubuntu:latest"
      + output      = (known after apply)
      + repo_digest = (known after apply)
    }

Plan: 2 to add, 0 to change, 0 to destroy.

─────────────────────────────────────────────────────────────────────────────

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if
you run "terraform apply" now.

C:\Users\praja\OneDrive\Desktop\Terraform Scripts\Docker>
```

Step 5: Execute Terraform apply to apply the configuration, which will automatically create and run the Ubuntu Linux container based on our configuration. Using command : “terraform apply”



This will ask You to enter a value so Type “Yes”.



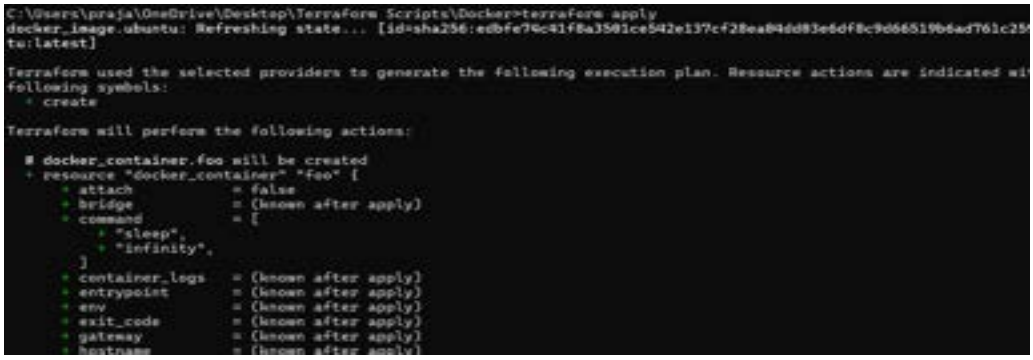
The script that we are using is going to throw an error. Error: container exited immediately .This is because the script used is way too small or took a lot less time to execute. To fix this, we add a

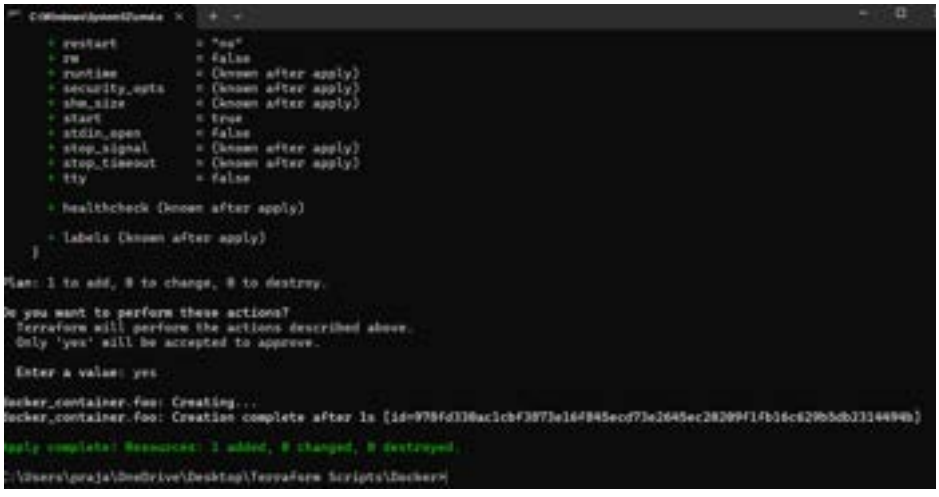
line to the code. ‘Command = [“sleep”, “infinity”]’. This line of code lets docker know to keep the program in sleep mode for an infinite amount of time so that the output can be observed rather than stopping after running immediately.

Do the following changes in the last line of the code as follows to solve the error

```
# Create a container
resource "docker_container" "foo" {
  image = docker_image.ubuntu.image_id
  name  = "foo"
  command = ["sleep","infinity"]
}
```

Now run the command again





Docker images, Before Executing Apply step:

```
C:\Users\praja\OneDrive\Desktop\Terraform Scripts\Docker>docker images
REPOSITORY                 TAG       IMAGE ID       CREATED        SIZE
docker/welcome-to-docker   latest    c1f619b6477e   9 months ago   18.6MB
```

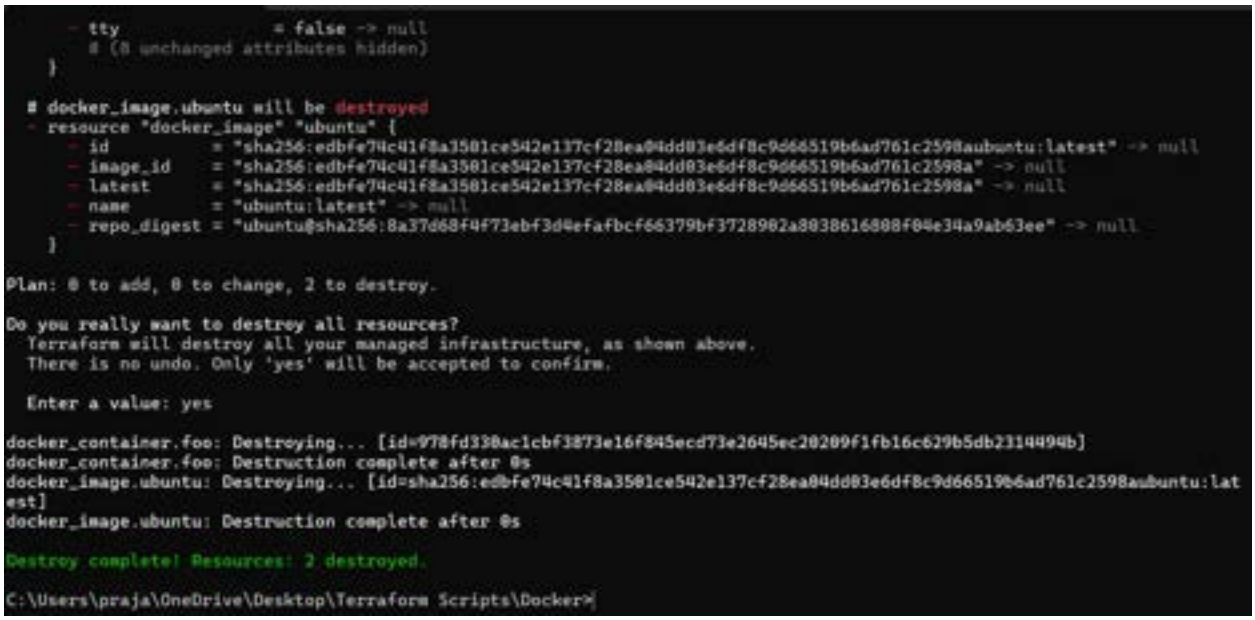
Docker images, After Executing Apply step:

```
C:\Users\praja\OneDrive\Desktop\Terraform Scripts\Docker>docker images
REPOSITORY                 TAG       IMAGE ID       CREATED        SIZE
ubuntu                     latest    edbfe74c41f8   2 weeks ago    78.1MB
docker/welcome-to-docker   latest    c1f619b6477e   9 months ago   18.6MB
```

Step 6: Now the image is created, if we have to destroy it. For this, we use the '*terraform destroy*' command. Again, this command will ask for a prompt to enter yes, as a confirmation to destroy the image we created.

Type Yes.

```
C:\Users\praja\OneDrive\Desktop\Terraform Scripts\Docker>terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubun
tu:latest]
docker_container.foo: Refreshing state... [id=978fd338ac1cbf3873e16f845ecd73e2645ec20209f1fb16c629b5db2314494b]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
  - destroy

Terraform will perform the following actions:

  # docker_container.foo will be destroyed
  - resource "docker_container" "foo" {
      - attach           = false -> null
      - command          = [
          - "sleep",
          - "infinity",
        ] -> null
      - cpu_shares       = 0 -> null
      - dns              = [] -> null
      - dns_opts         = [] -> null
      - dns_search       = [] -> null
      - entrypoint       = [] -> null
      - env              = [] -> null
      - gateway          = "172.17.0.1" -> null
      - group_add        = [] -> null
      - hostname         = "978fd338ac1c" -> null
      - id               = "978fd338ac1cbf3873e16f845ecd73e2645ec20209f1fb16c629b5db2314494b" -> null
```



Docker images After Executing Destroy step

Thus we have Successfully created the Docker image using terraform in this experiment and have also destroyed it

CONCLUSION:

The experiment successfully demonstrate the use of Terraform to automate the deployment and management of docker container .By creating the Terraform Script, initializing the environment and executing necessary commands we were able to deploy on Ubuntu Linux Container which was deleted at the last part of the experiment

Name: Shivam.R.Prajapati Div: D15C Roll No:41 Academic Year: 2024-25

Experiment No :7

AIM: To understand Static Analysis SAST process and learn to integrate Jenkins SAST to SonarQube/GitLab.

PREREQUISITES:

1) Docker :
   Run **docker -v command**. We use this command to check if docker is installed and running on your system.

```
C:\Users\praja>docker -v
Docker version 27.0.3, build 7d4bcd8
```

2) Install SonarQube Image:
   The command **docker pull sonarqube** downloads a SonarQube image from Docker's online repository. This image lets you run SonarQube on your system using Docker without needing to install the full SonarQube software manually. It's like getting a ready-to-use version of SonarQube that can be started with Docker.

```
C:\Users\praja>docker pull sonarqube
Using default tag: latest
latest: Pulling from library/sonarqube
7478e0ac0f23: Pull complete
98a925ab929a: Pull complete
7d9a34308537: Pull complete
80338217a4ab: Pull complete
1a5fd5c7e184: Pull complete
7b87d6fa783d: Pull complete
bd819c9b5ead: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:72e9feec71242af83faf65f95a40d5e3bb2822a6c3b2cda8568790f3d31aecde
Status: Downloaded newer image for sonarqube:latest
docker.io/library/sonarqube:latest

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview sonarqube
```

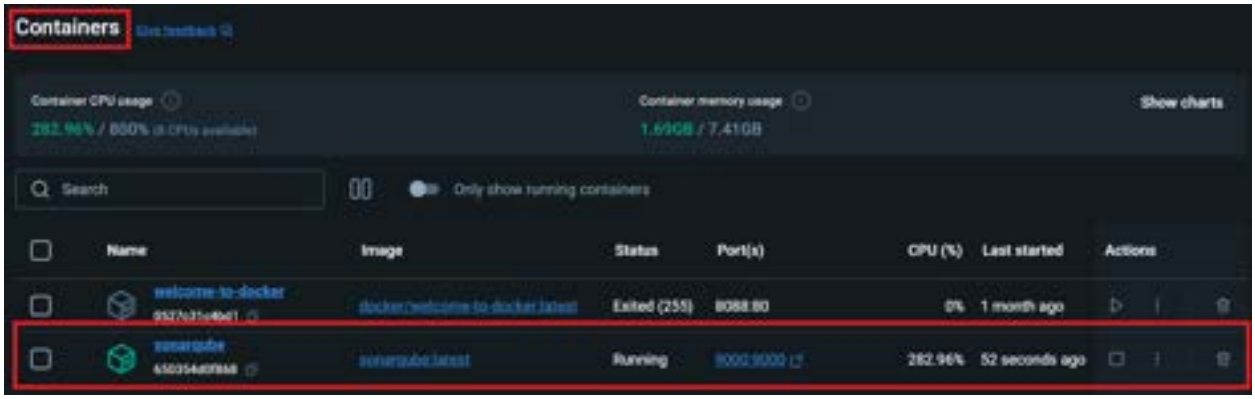
3) Make sure **Jenkins** is already installed on your system before starting the process. Jenkins will be used to automate tasks, like running SonarQube for code analysis. If Jenkins isn't installed yet, you can download and set it up from the official Jenkins website.

Name: Shivam.R.Prajapati Div: D15C Roll No:41 Academic Year: 2024-25

STEPS:

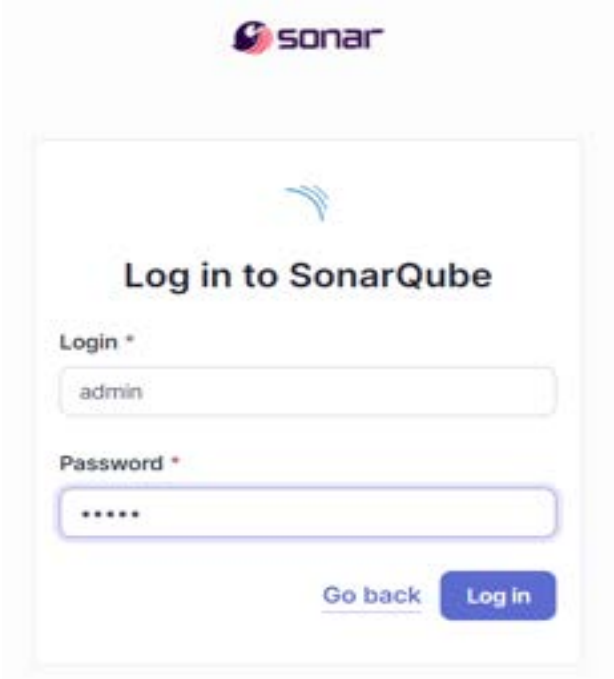
Step1: The command **docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest** starts SonarQube in the background on port 9000 using Docker, allowing you to access it at http://localhost:9000

```
C:\Users\praja>docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
650354d0f868ae4ad2d800426080076c604eb09f29b10d4a251aee70f51ce907

C:\Users\praja>
```

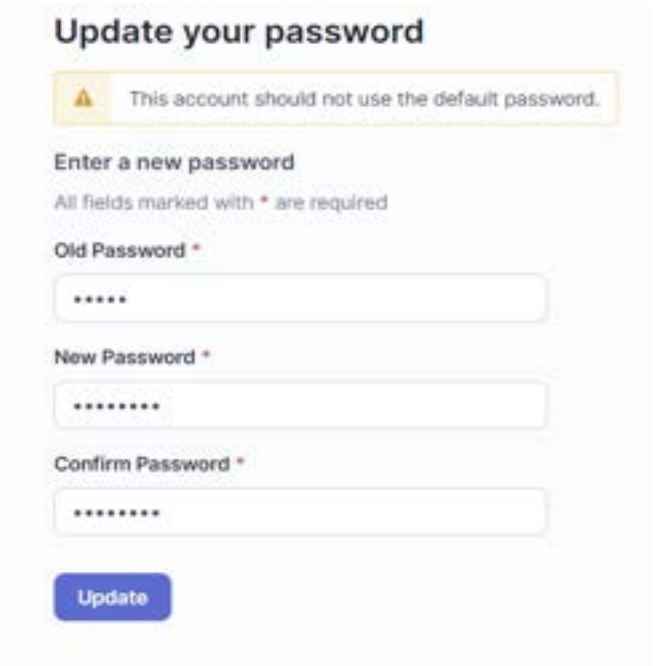


Step2: After starting the SonarQube image, open your browser and go to **http://localhost:9000** to access SonarQube.

Step 3: On the SonarQube login page, use the default credentials: **Username: admin** , **Password: admin**. After logging in, you'll be prompted to change the password. Set a new password and make sure to remember it.

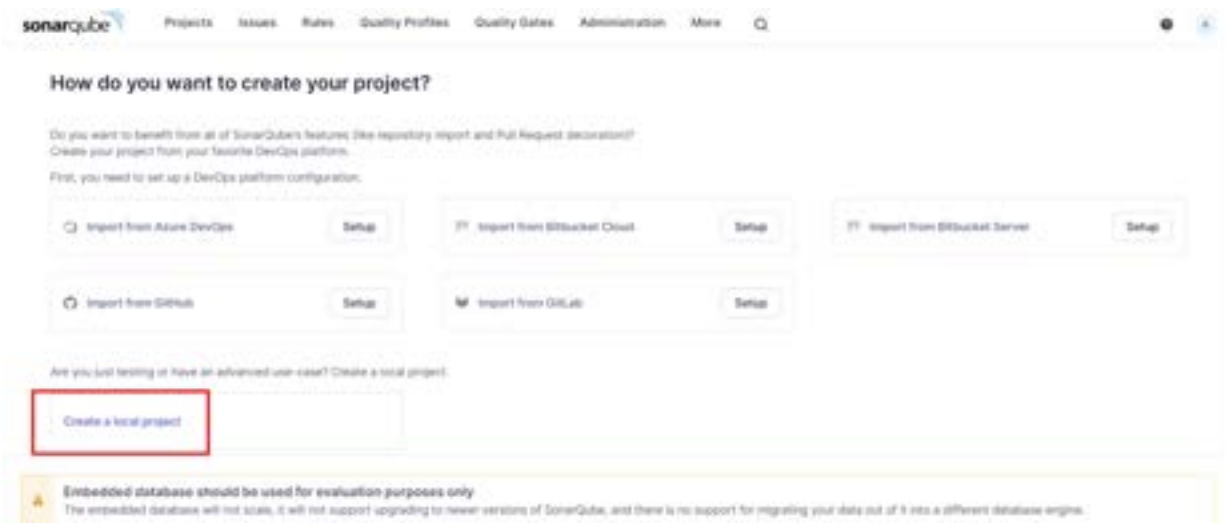


Click on Log in



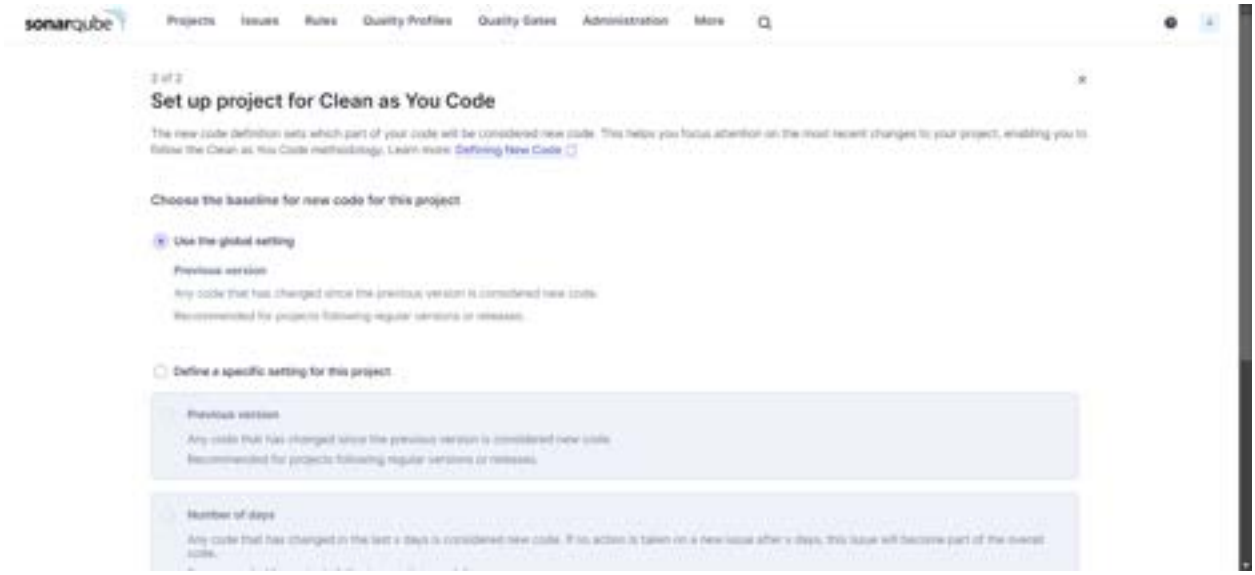
Click on Update .

Step 4: After changing the password, you will be directed to this screen. Click on **Create a Local Project**.

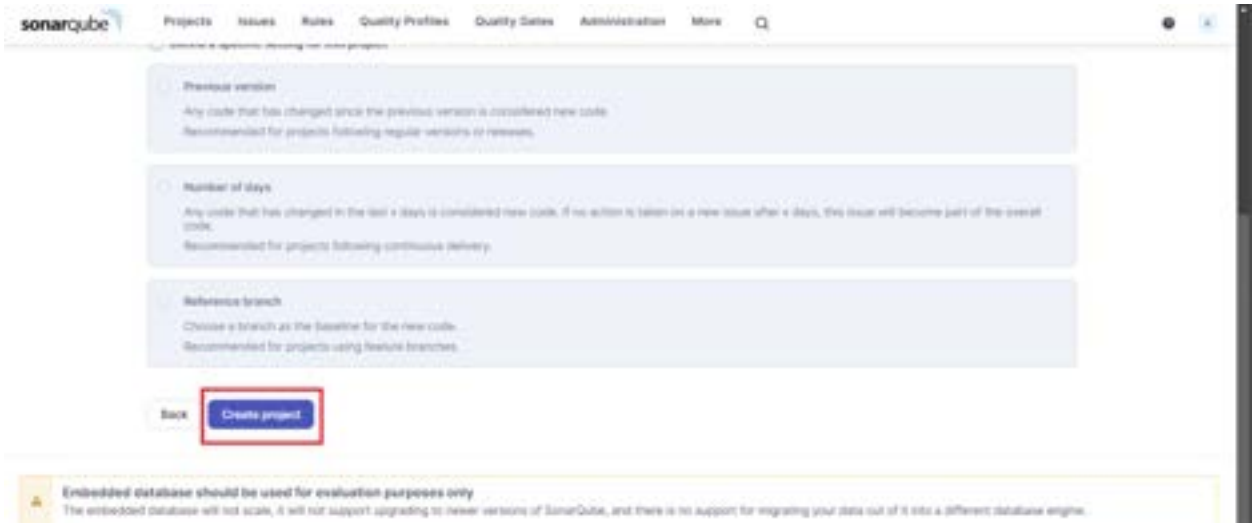


Step 5: Give your project , a display name and project key

Step 6: Configure the project by providing the necessary settings like choosing the baseline for the new code for the project , then click **Create** to finalize the setup.

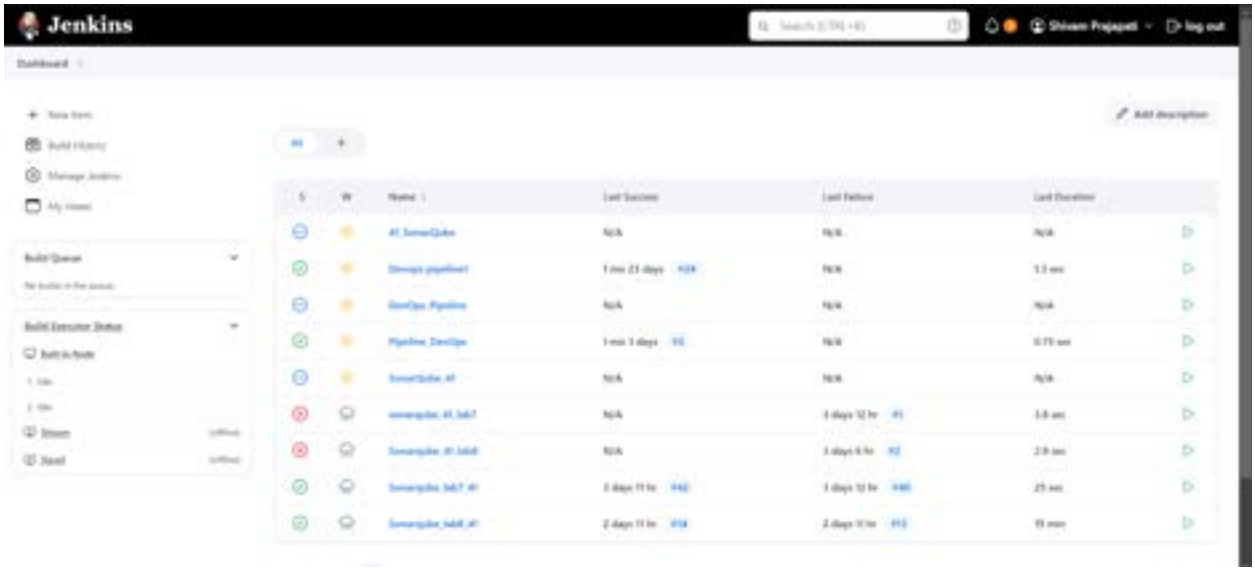


Scroll Down

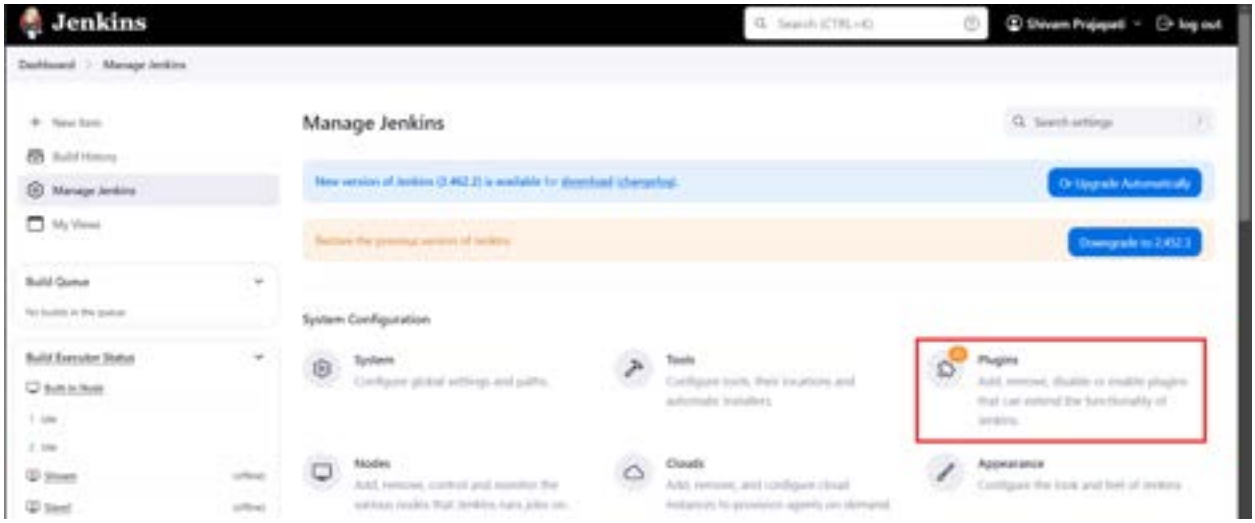


Click on Create project

Step 7: Open Jenkins by going to **http://localhost:<port_number>** in your browser, replacing <port_number> with the specific port Jenkins is running on.



Step 8: Now go to Manage Jenkin then go for Plugins followed by Available plugins search for **Sonarqube Scanner** where we are going to install it as a plugin.

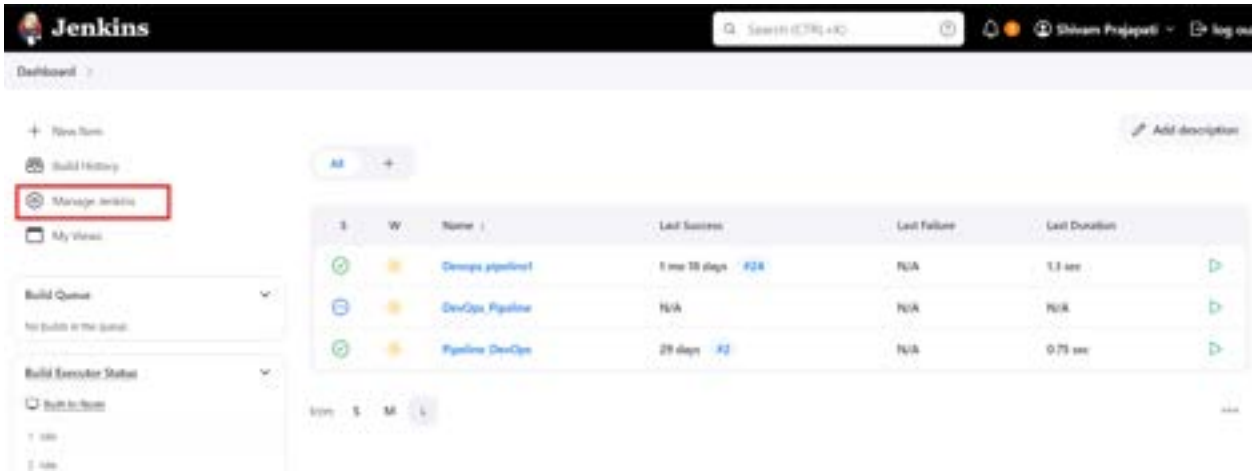


Click on Available Plugins.

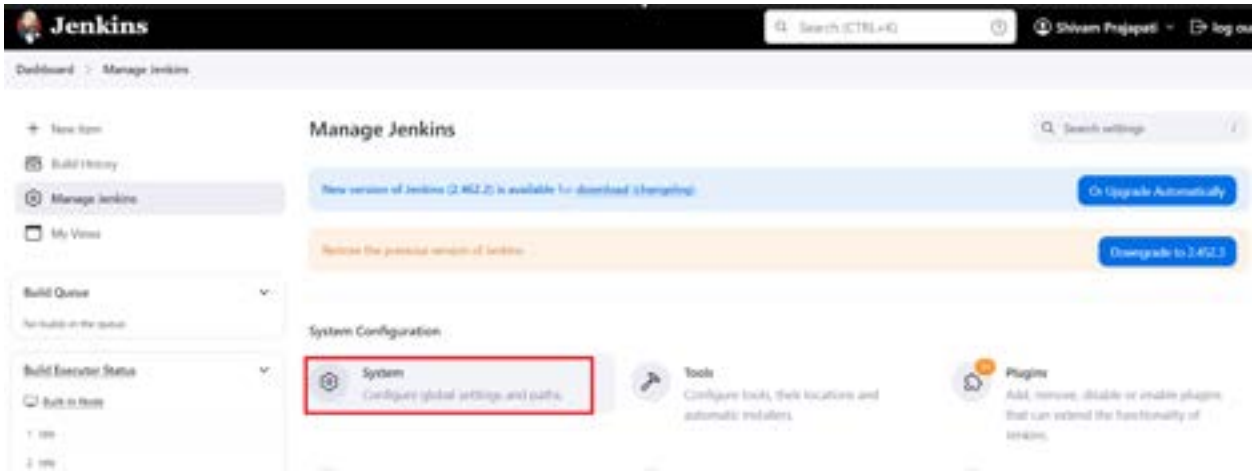
Search in the Search bar the required Plugin Name and click on Install.

Plugin Installed Successfully.

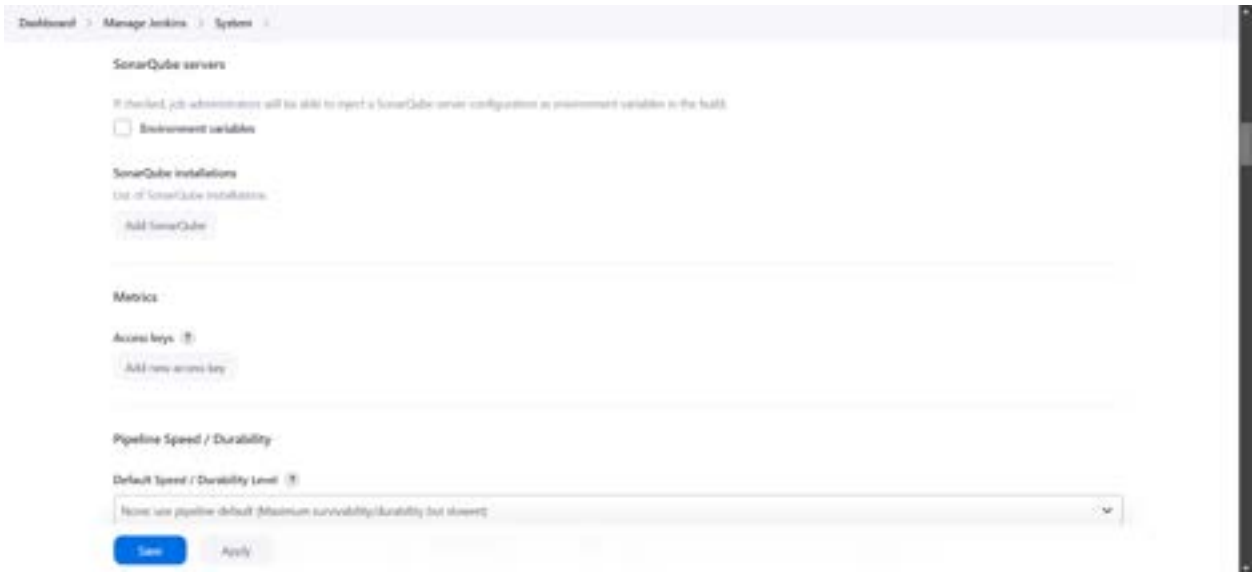
Step 9: In Jenkins, go to **Manage Jenkins → System**, then find **SonarQube servers**. Add a new server, and if required, include the authentication token for secure access



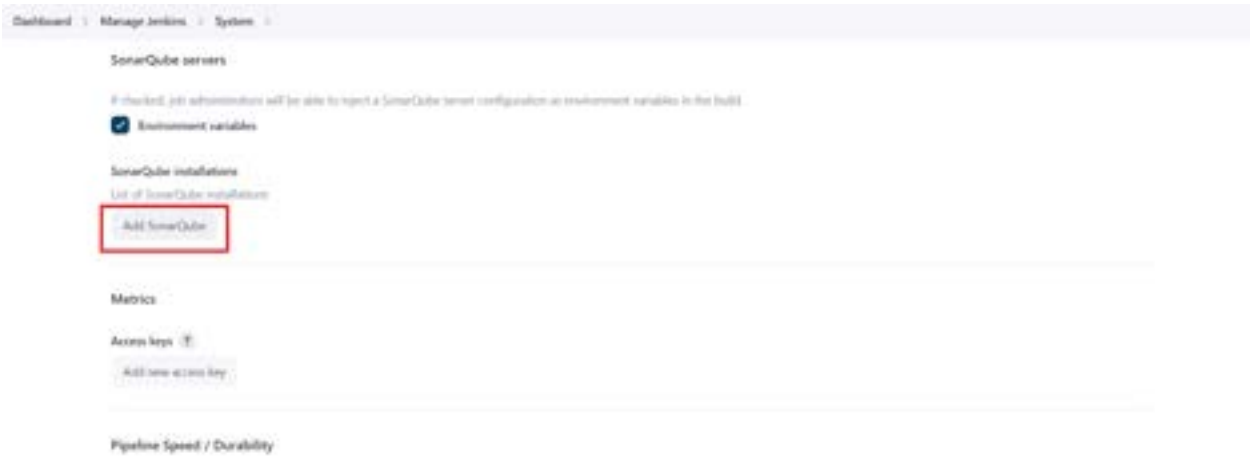
Go to system



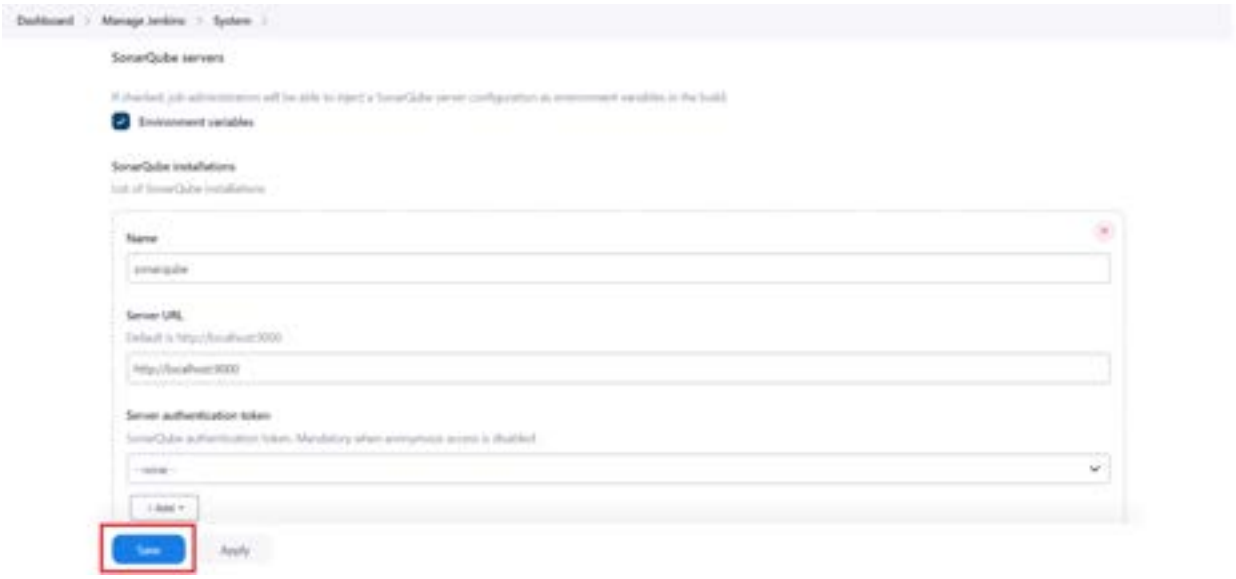
Scroll Down



Select Environment Variable and Click on Add Sonar Qube button in order to Add SonarQube Server to Jenkin

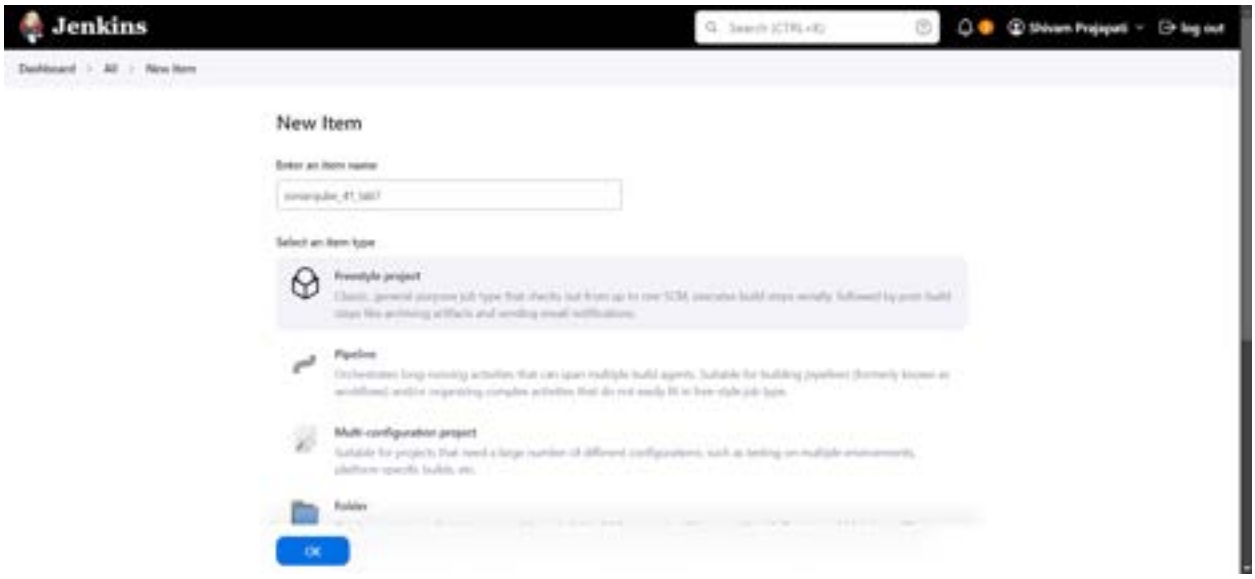


Do the required entries as shown below

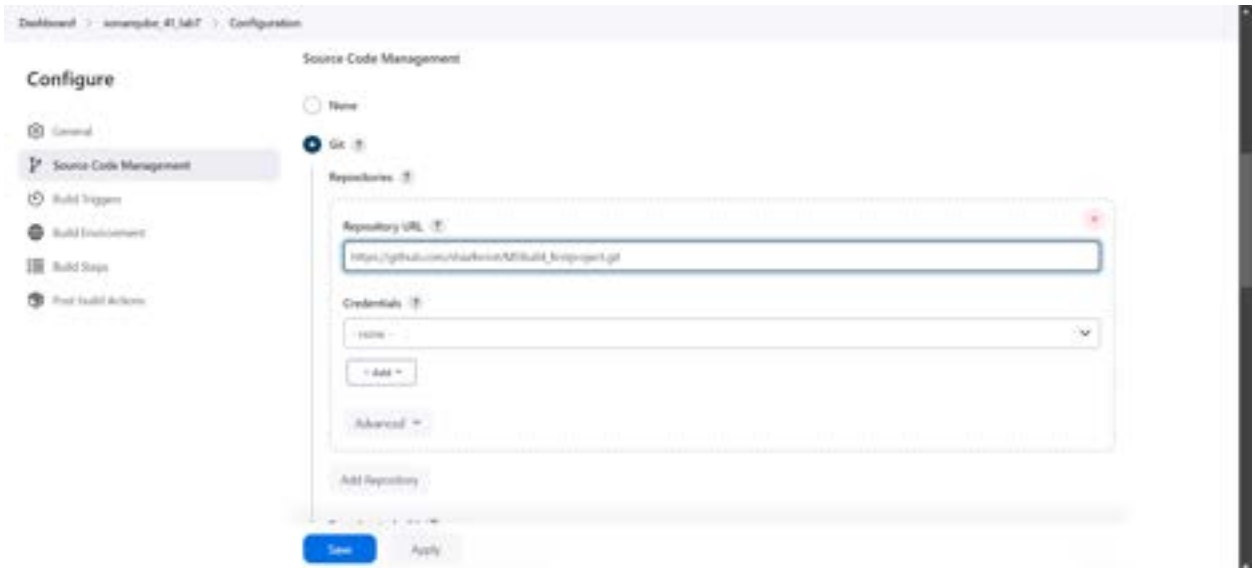


Click on save

Step 10: After configuration, create a New Item → choose a freestyle project.

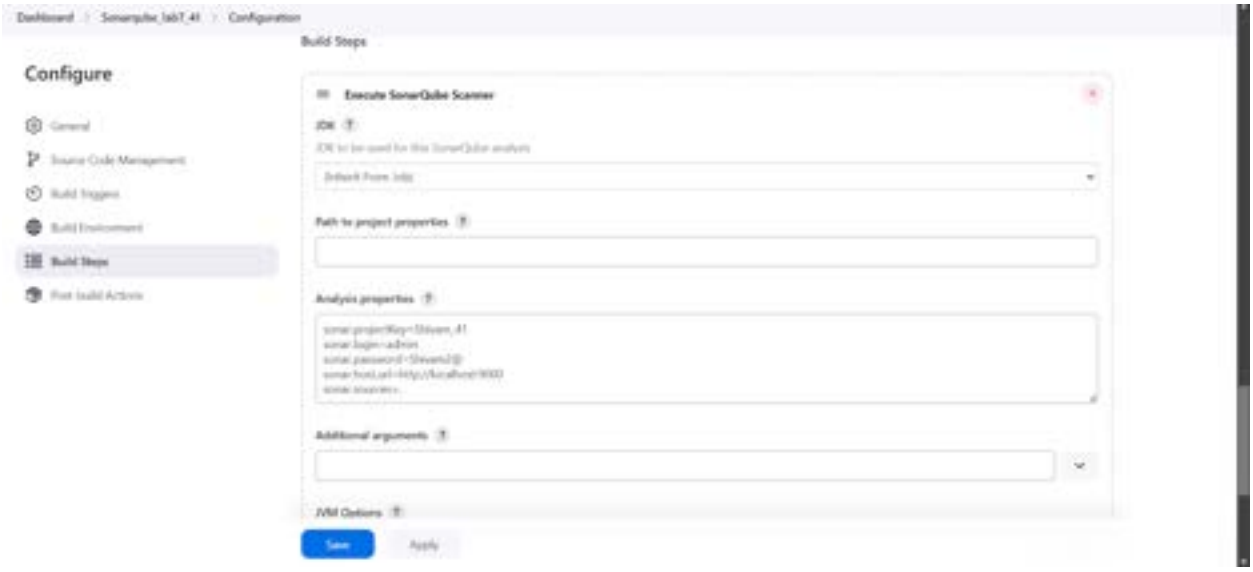


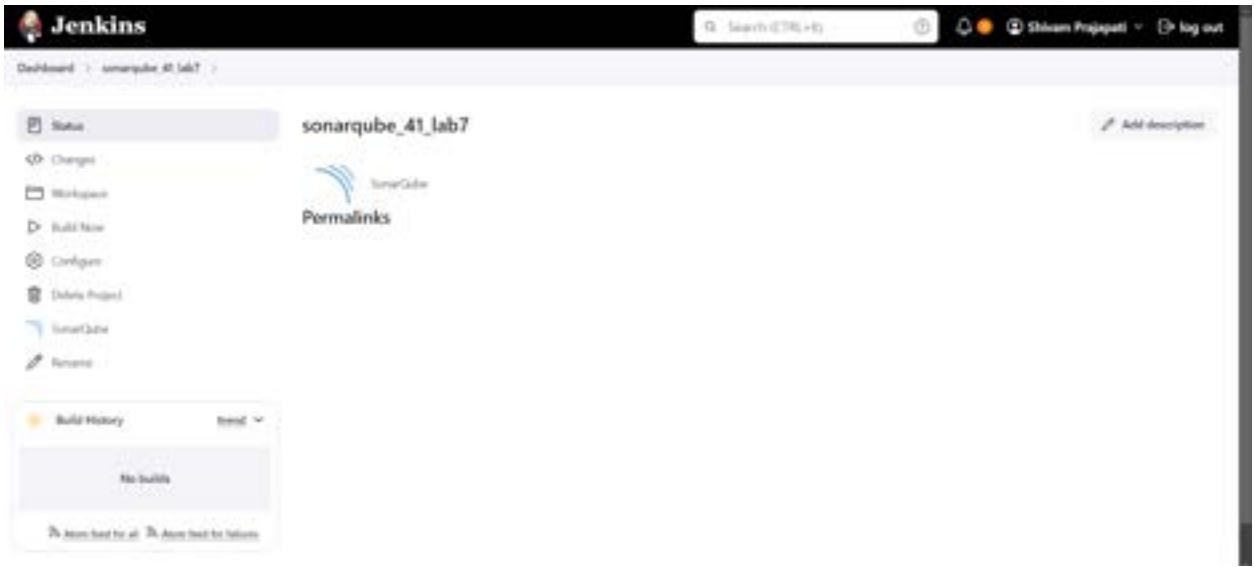
Step 11: Use this github repository in Source Code Management. **https://github.com/shazforiot/MSBuild_firstproject.** It is a sample hello-world project with no vulnerabilities.



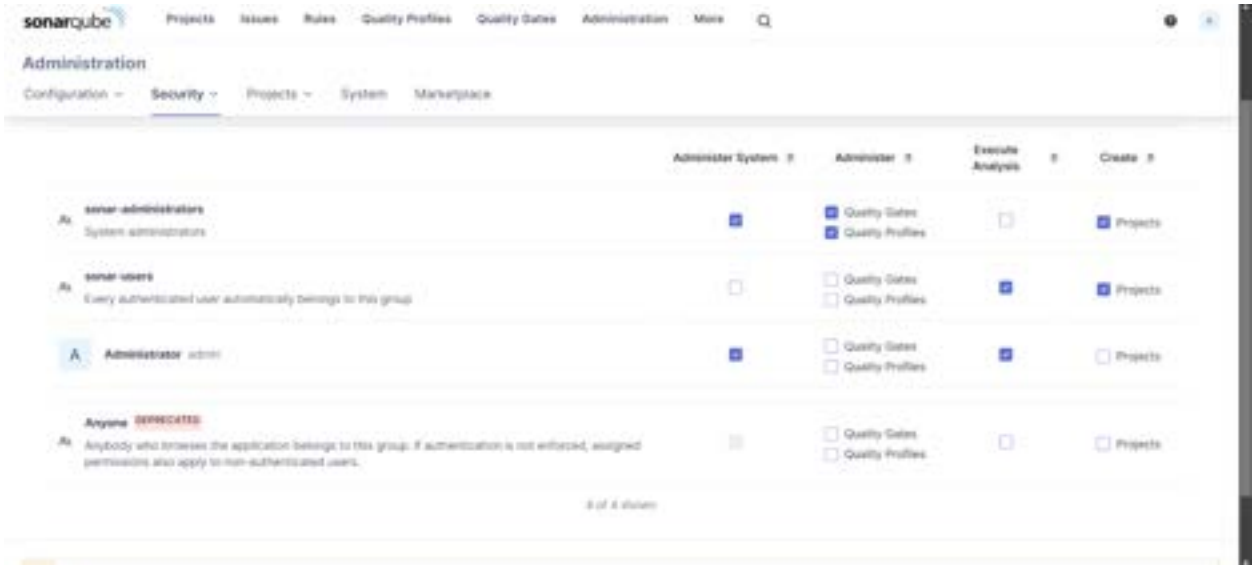
Step 12: Under Build Steps, enter Sonarqube Scanner, enter these Analysis properties. Mention the SonarQube Project Key, Login, Password, Source path and Host URL.

Click on execute sonar scanner

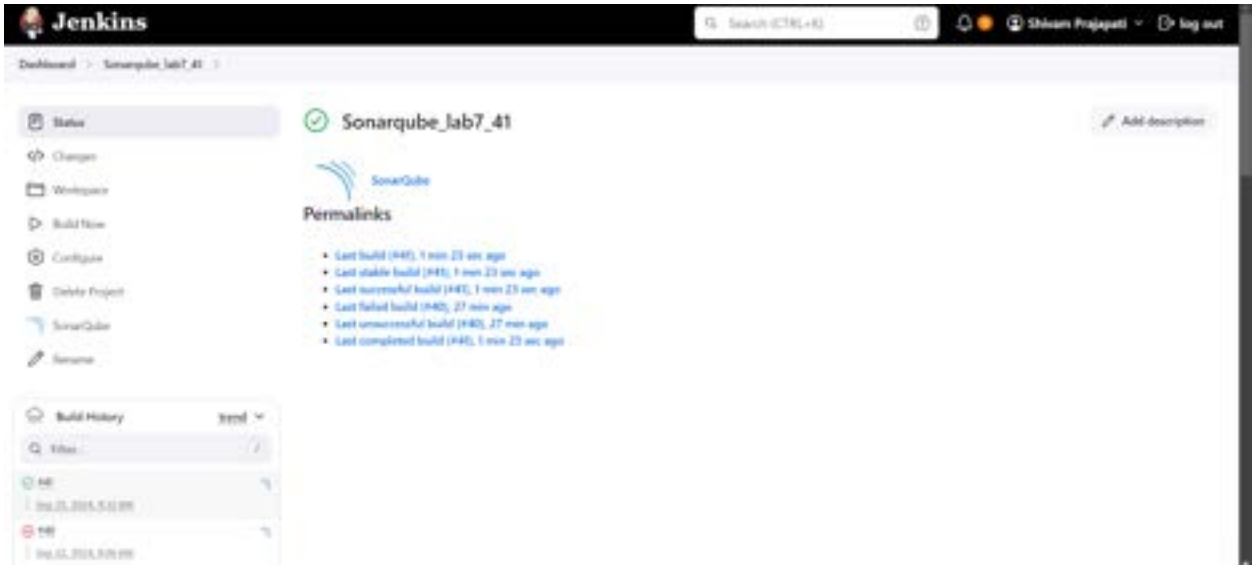




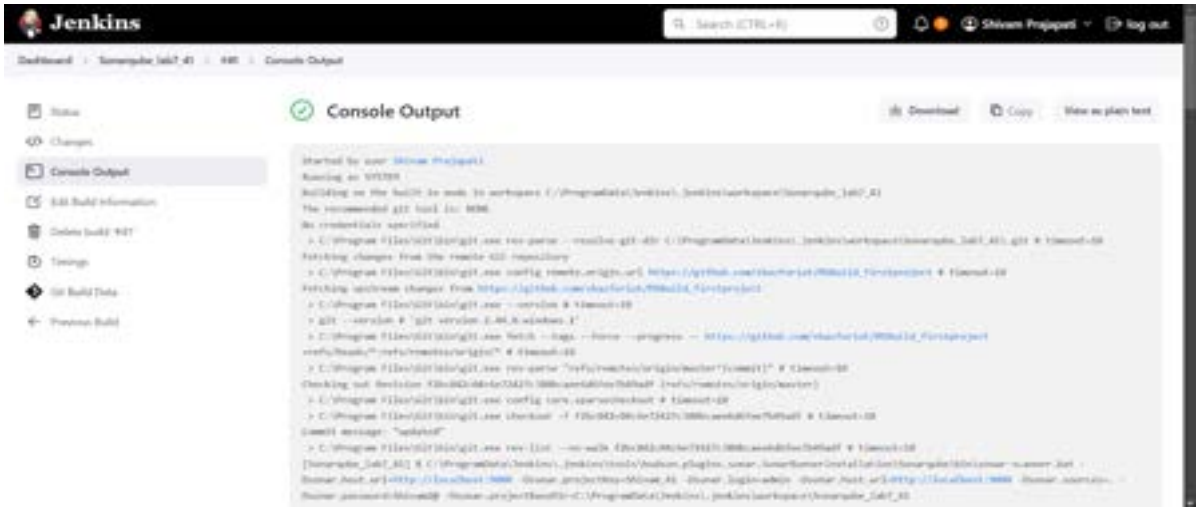
Step 13: Now, you need to grant the local user (here admin user) permissions to Execute the Analysis stage on SonarQube. For this, go to **http://loaclhost:<port_number>/admin/permissions** and check the 'Execute Analysis' checkbox under Administrator.



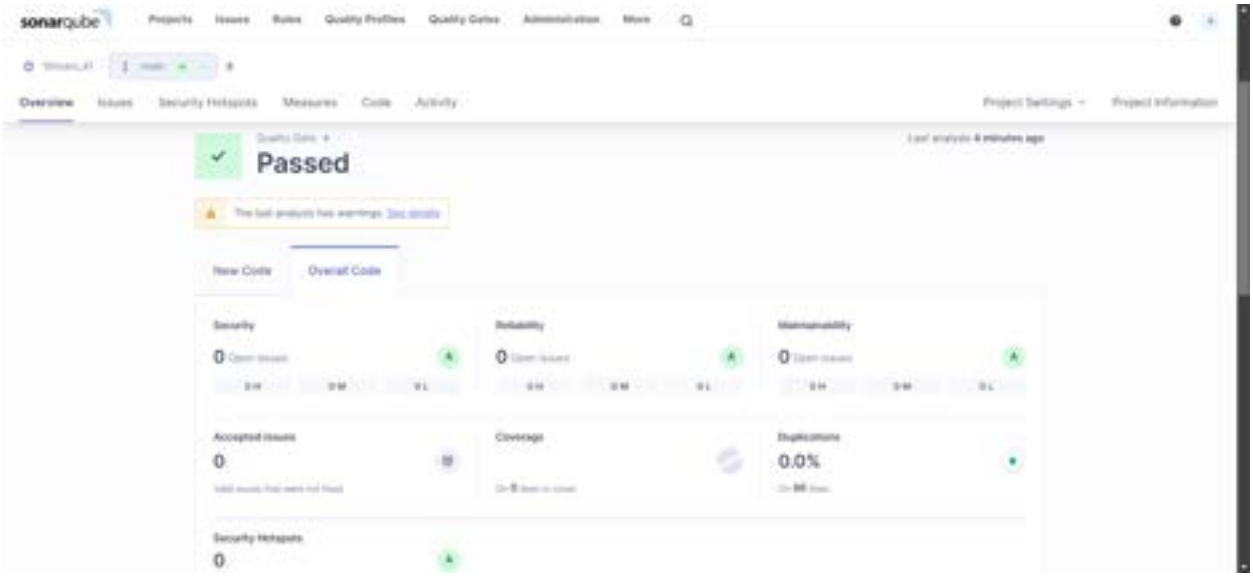
Step 14: Go back to jenkins. Go to the job you had just built and click on Build Now.



Check the Console Output



Step 15: Once the build is complete, go back to SonarQube and check the project linked



.

CONCLUSION:

In this experiment, we have learned how to perform Jenkins SAST using SonarQube. For this, we used a docker image of SonarQube so as to not install it locally on our system. After installing the required configurations on Jenkins, using a coe from a gihub repository, we analyze its code using SonarQube. Once we build the project, we can see that the SonarQube project displays that the code has no errors.

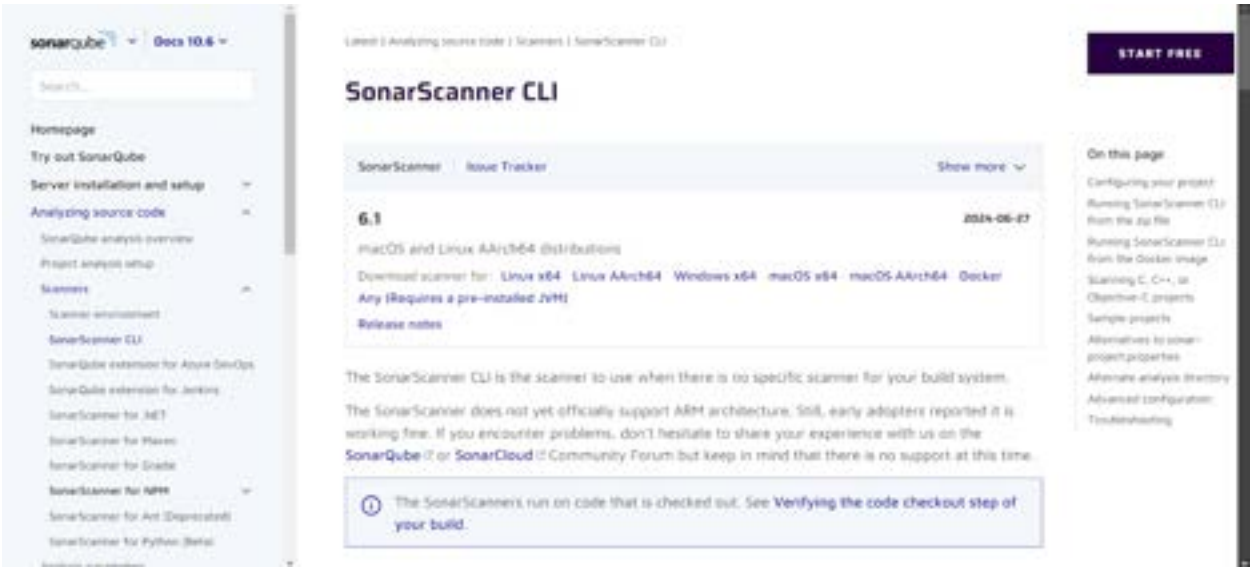
Name: Shivam.R.Prajapati Div: D15C Roll No:41 Academic Year: 2024-25

Experiment No: 8

AIM: Create a Jenkins CI/CD Pipeline with SonarQube / GitLab Integration to perform a static analysis of the code to detect bugs, code smells, and security vulnerabilities on a sample Web / Java / Python application.

PREREQUISITES:

Step 1: Download sonar scanner https://docs.sonarsource.com/sonarqube/latest/analyzing-source-code/scanners/sonarscanner/ . Visit this link and download the sonarqube scanner CLI



Extract the downloaded zip file in a folder.

Step 2: Docker Run **docker -v command .**If docker is not installed so install it

```
C:\Users\praja>docker --version
Docker version 27.0.3, build 7d4bcd8
```

Step 3: Install sonarqube image Command: **docker pull sonarqube**

```
C:\Users\Student>docker pull sonarqube
Using default tag: latest
latest: Pulling from library/sonarqube
762bedf4b1b7: Pull complete
95f9bd9906fa: Pull complete
a32d681e6b99: Pull complete
aabdd0a18314: Pull complete
5161e45ecd8d: Pull complete
aeb0020dfa06: Pull complete
01548d361aea: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:bb444c58c1e04d8a147a3bb12af941c57e0100a5b21d10e599384d59b
Status: Downloaded newer image for sonarqube:latest
docker.io/library/sonarqube:latest

What's next:
    View a summary of image vulnerabilities and recommendations → docker

C:\Users\Student>
C:\Users\Student>
```

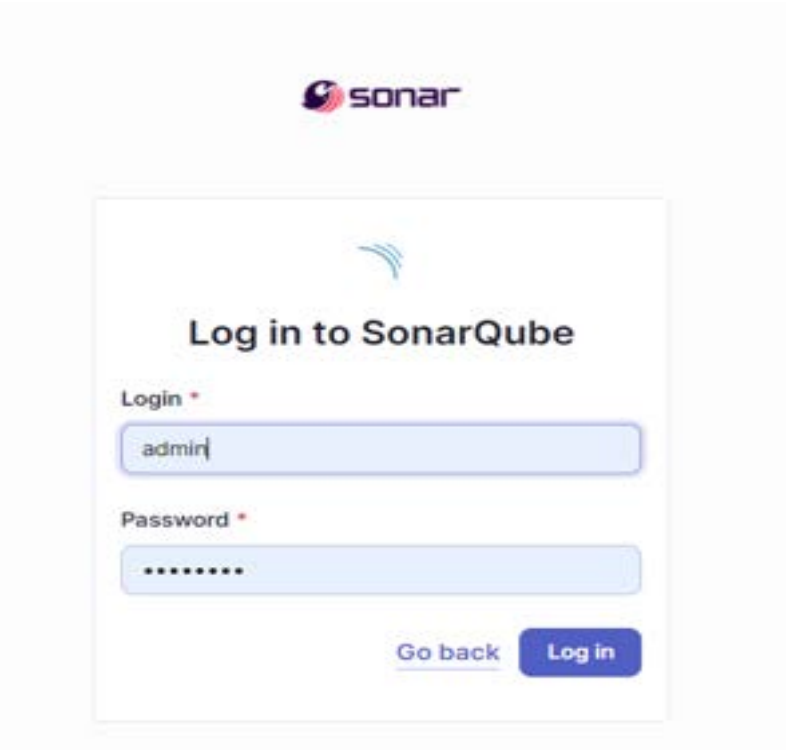
Step 4: Keep **Jenkins** installed on your system.

EXPERIMENT STEPS:

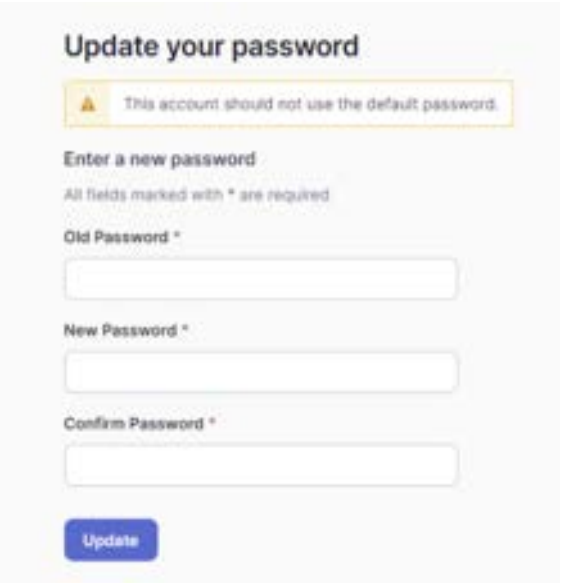
Step1: Run SonarQube image **docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest .**This command will run the SonarQube image that was just installed using docker.

```
C:\Users\Student>docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
83330c33cd961d8d659f362c5f62c6cd1ff87f31ec99da134350b9b419370561

C:\Users\Student>
```

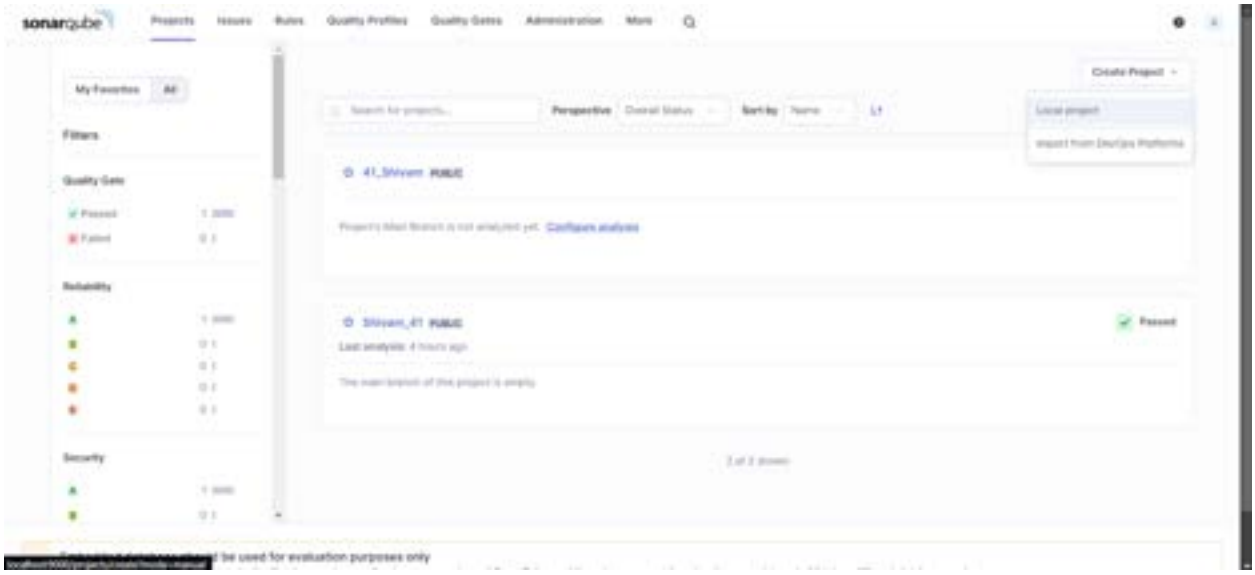
Step 2: Once the SonarQube image is started, you can go to **http://localhost:9000** to find the SonarQube that has started



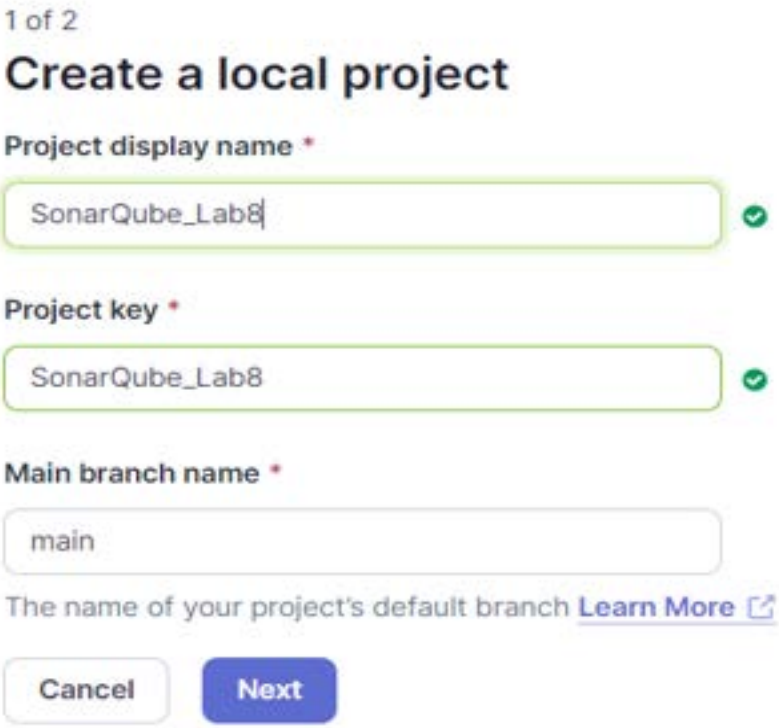
Step 3: On this interface, login with **username = 'admin'** and **password = 'admin**'. Once logged in successfully, SonarQube will ask you to reset this password. Reset it and remember this password.



Step 4: After changing the password, you will be directed to this screen. Click on **Create a Local Project**. Give the project a display name and project key

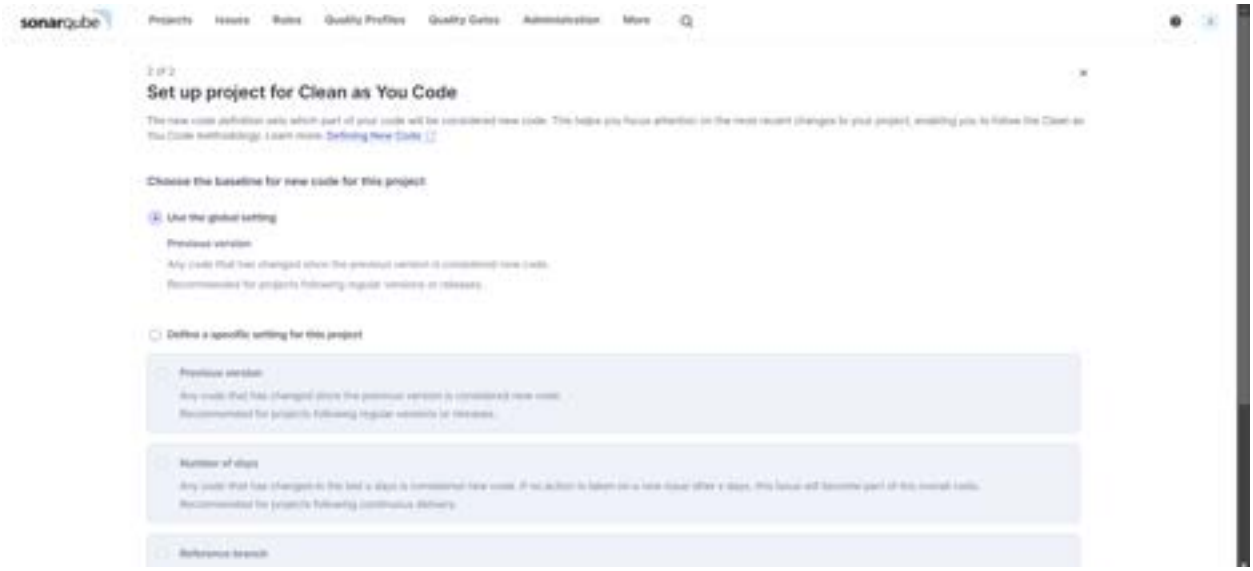


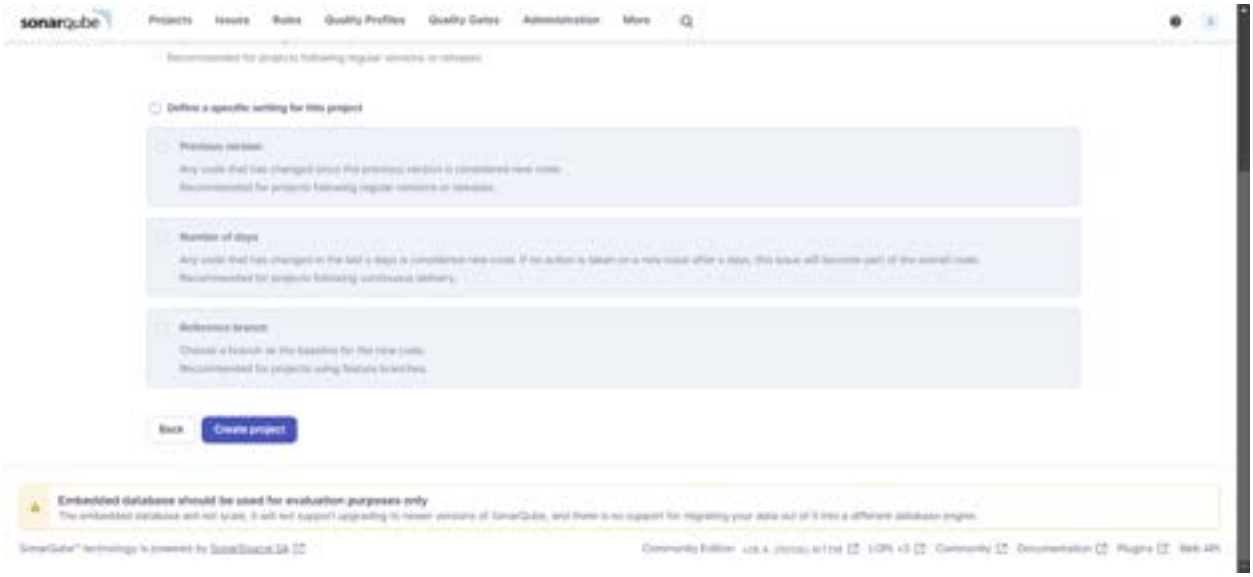
Click on Create Project



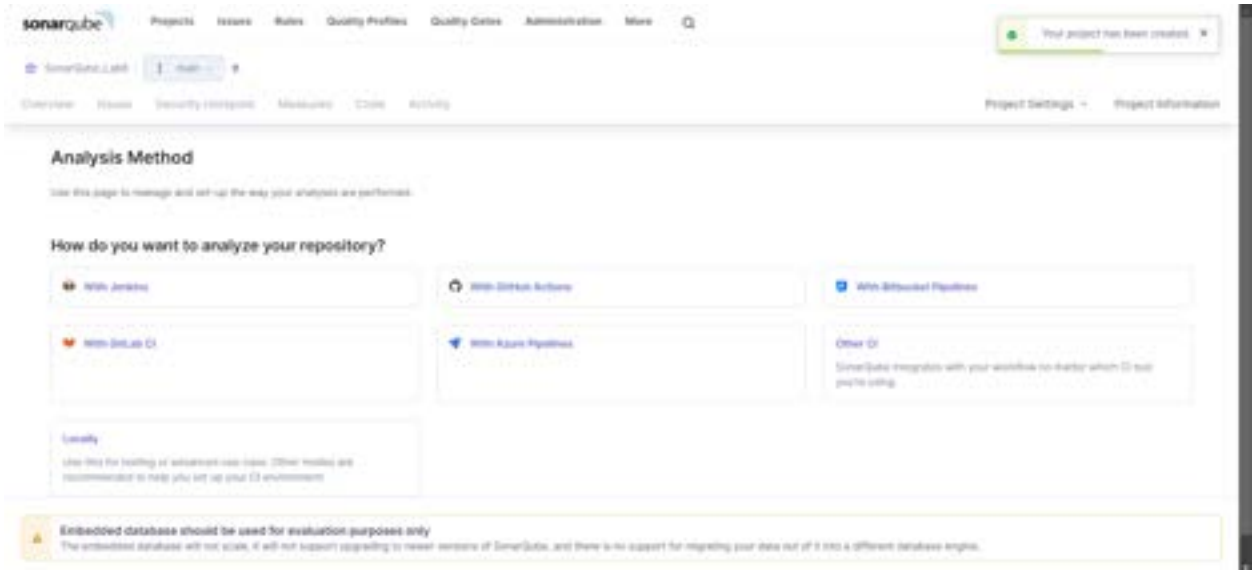
Set up the project as required and click on create.

In the Step 2 while creating the project,Sonarqube ask you regarding which code should be considered as the new code for examining it .



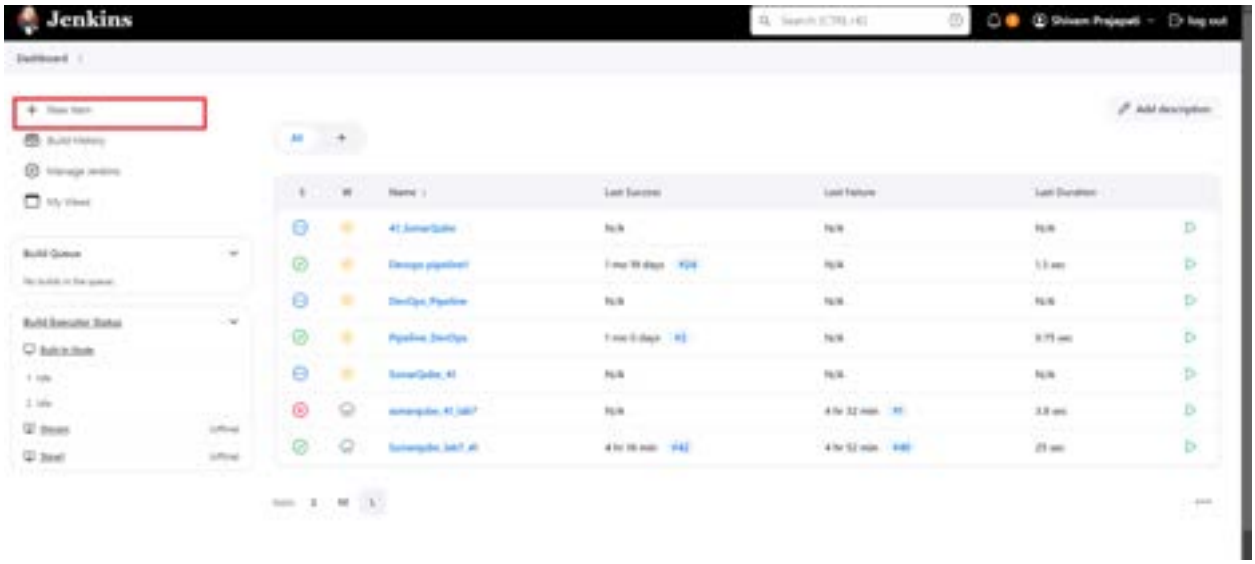


Click on Create

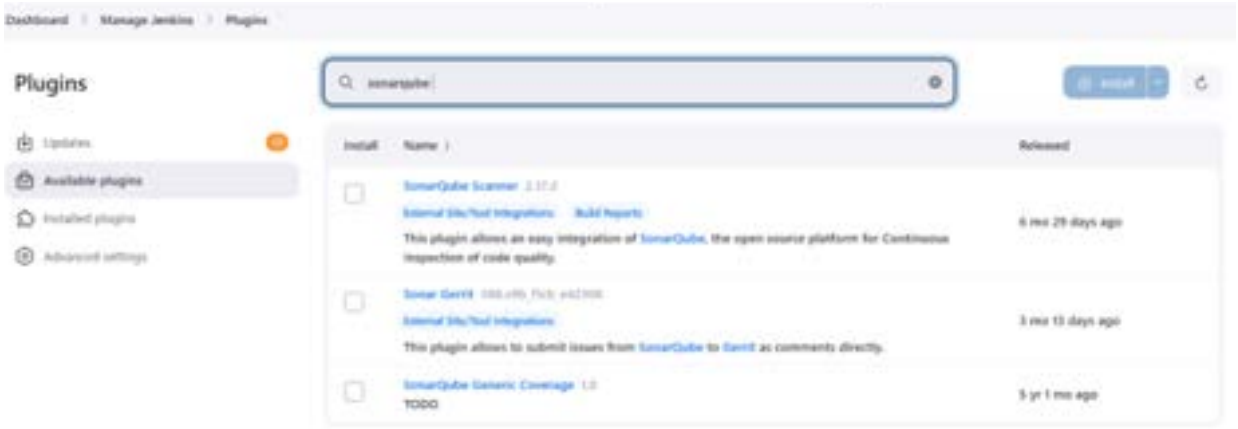


Project is created

Step 5: Open **Jenkins** on whichever port it is installed. (http://loaclhost:). Go to the new item

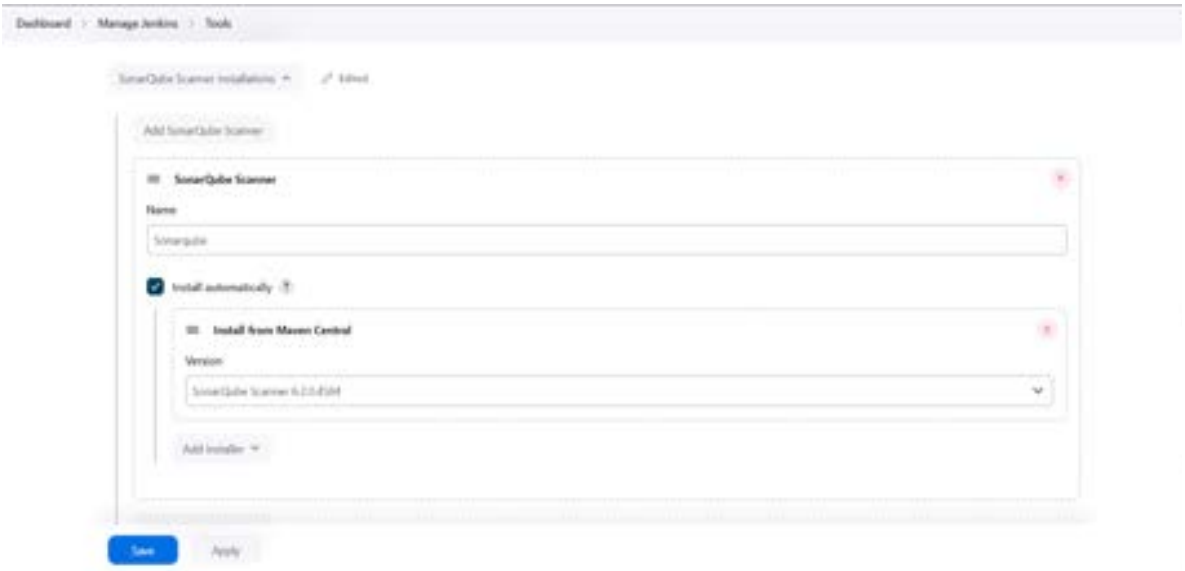


Step 6: Go to manage jenkins →available plugins then Search for **Sonarqube Scanner** for Jenkins and install it



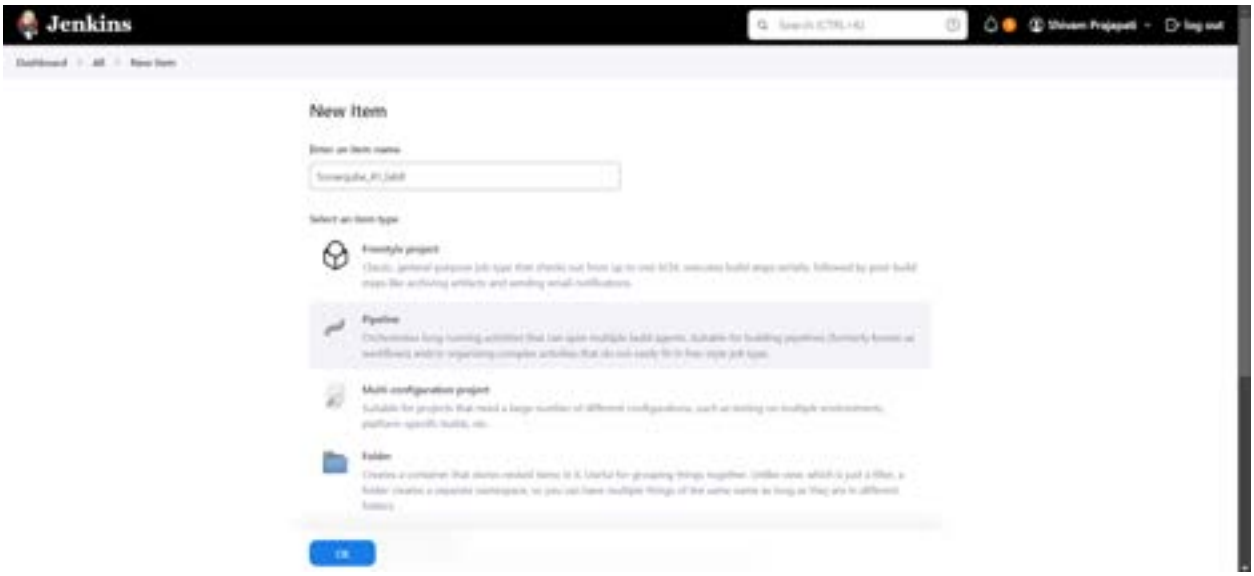
Step 7: Now, go to Manage Jenkins → System. Under Sonarqube servers, add a server. Add server authentication token if needed.

Step 8: Go to Manage Jenkins → Tools. Go to SonarQube scanner, choose the latest configuration and choose to install automatically.



Step 9: After configuring, click on **New Item** and select **Pipeline Project**

.



Step 10: Under Pipeline script, enter the following:

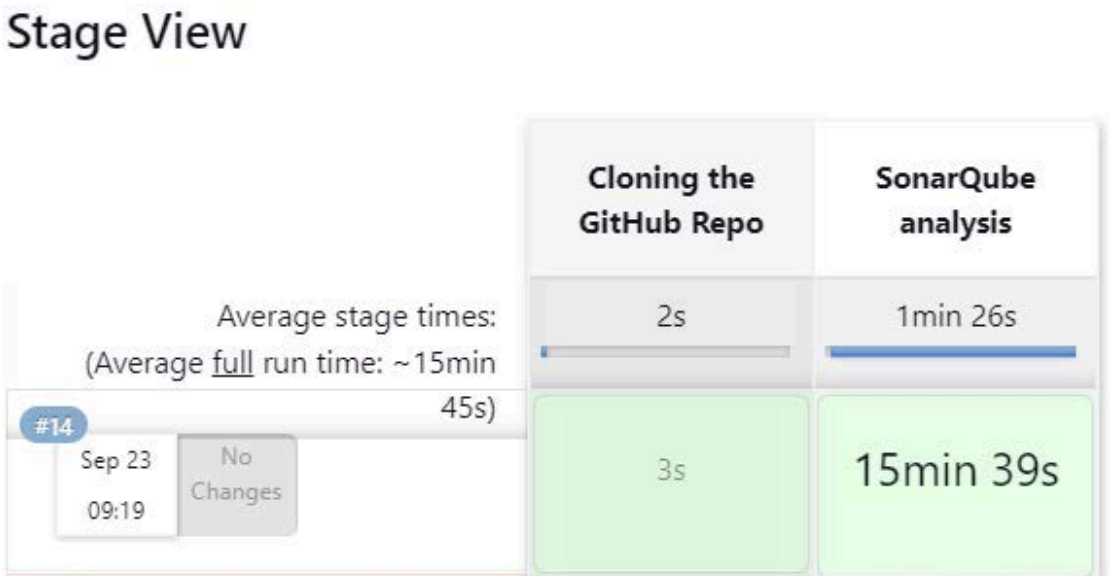
```
node {
   stage('Cloning the GitHub Repo') {
      git 'https://github.com/shazforiot/GOL.git'
```

```
    }
    stage('SonarQube analysis') {
        withSonarQubeEnv('sonarqube') {
            bat """

C:\\Users\\praja\\Downloads\\SonarqubeCLI\\sonar-scanner-6.1.0.4477-windows-x64\\bin\\sonar-scanner.bat ^
            -D sonar.login=admin ^
            -D sonar.password=Shivam2@ ^
            -D sonar.projectKey=SonarQube_Lab8 ^
            -D sonar.exclusions=vendor/**,resources/**,**/*.java ^
            -D sonar.host.url=http://localhost:9000/
            """
        }
    }
}
```

Click on save.

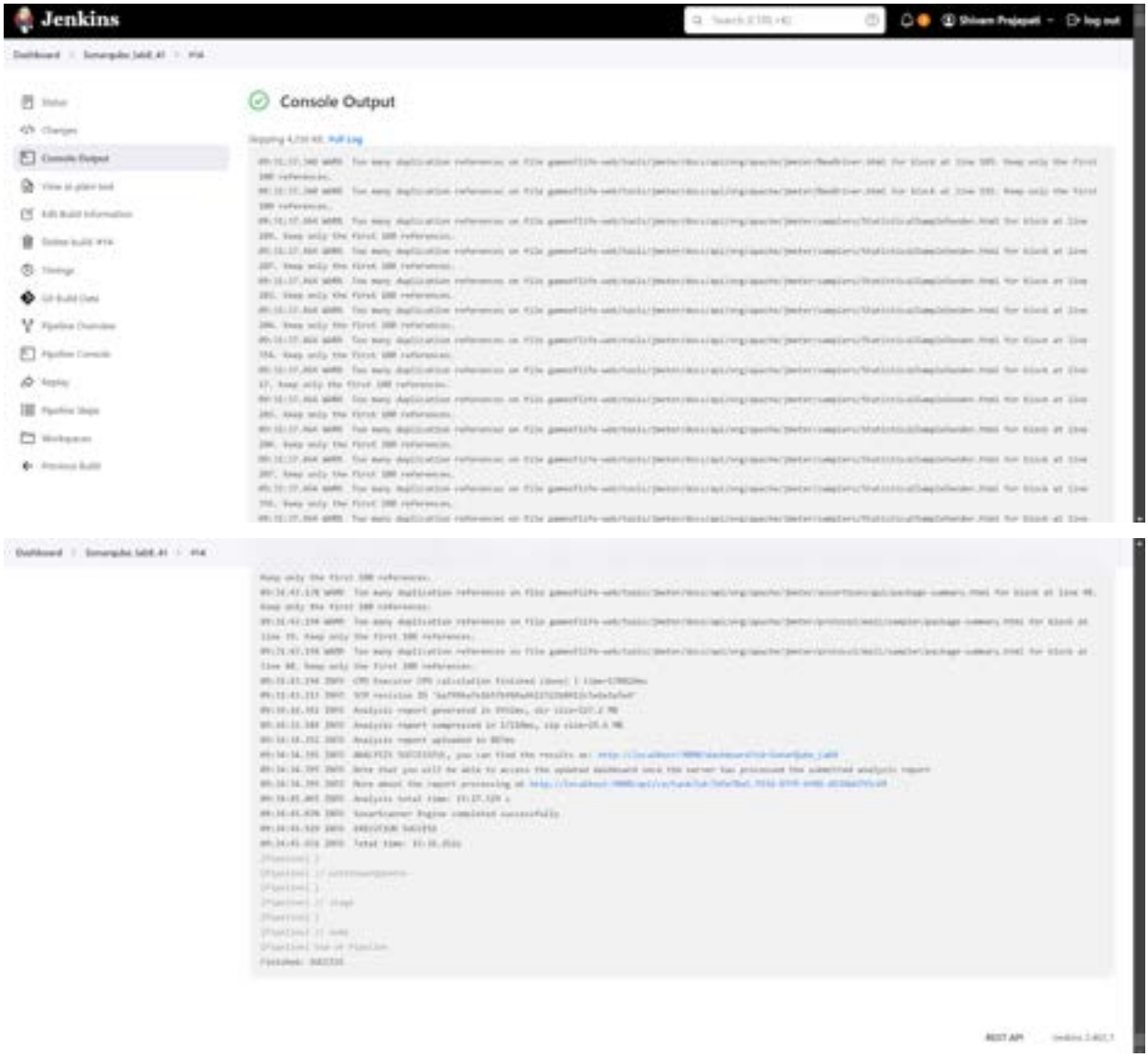
This is a Java sample project with many repetitive sections and coding issues that SonarQube will be able to detect during analysis.

Step 11: Go back to jenkins. Go to the job you had just built and click on Build Now.



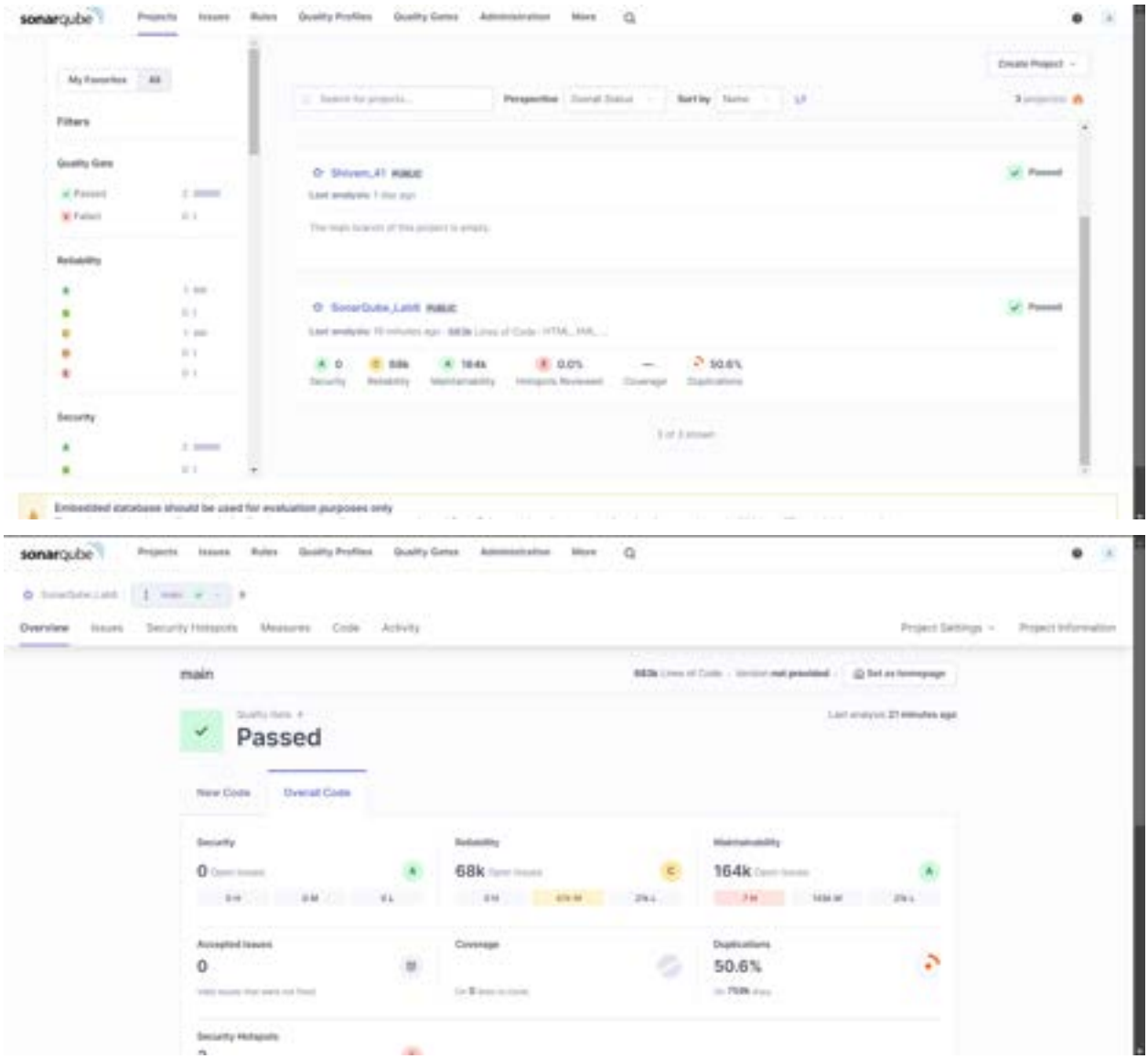
The problem was C:\windows\system32 was not there so we need to add in our environment variable .

Now Check the console output once



Successfully BUILD

Step 12: After the build is finished, return to SonarQube and review the linked project in detail.

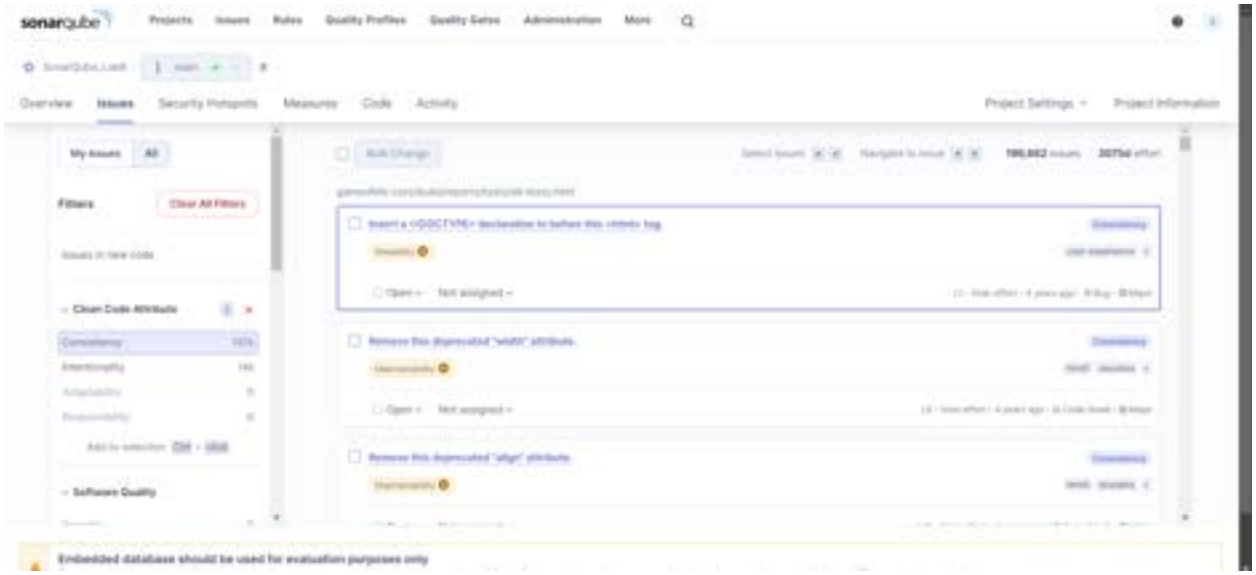


Under different options on the navbar , we can check all the issues with the code.

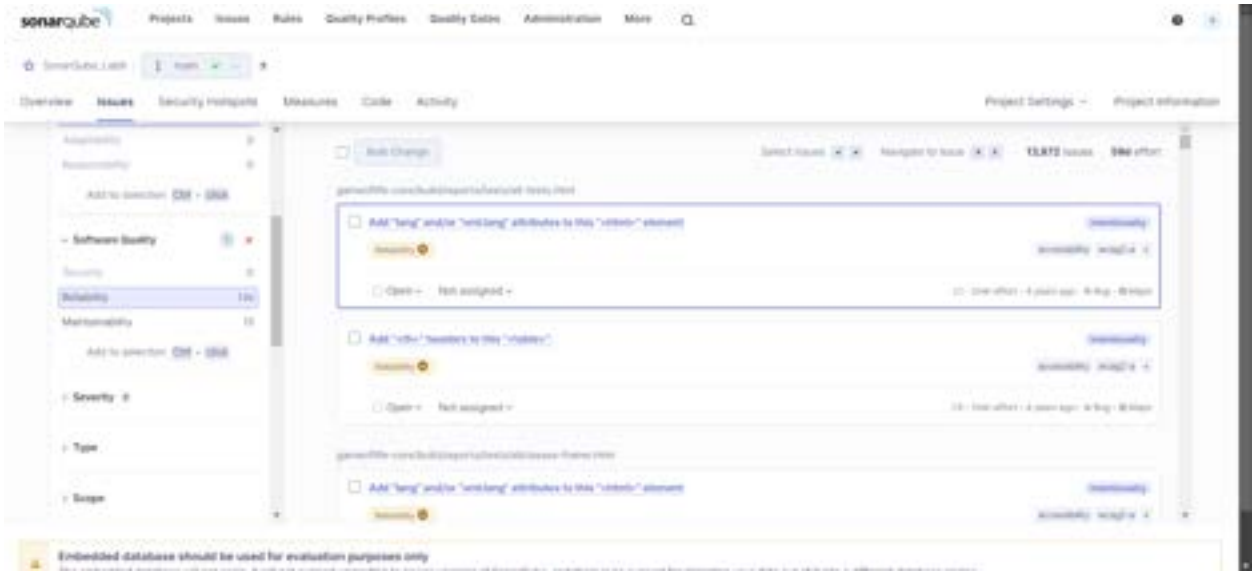
Name: Shivam.R.Prajapati Div: D15C Roll No:41 Academic Year: 2024-25

UNDER ISSUES:

1) Consistency

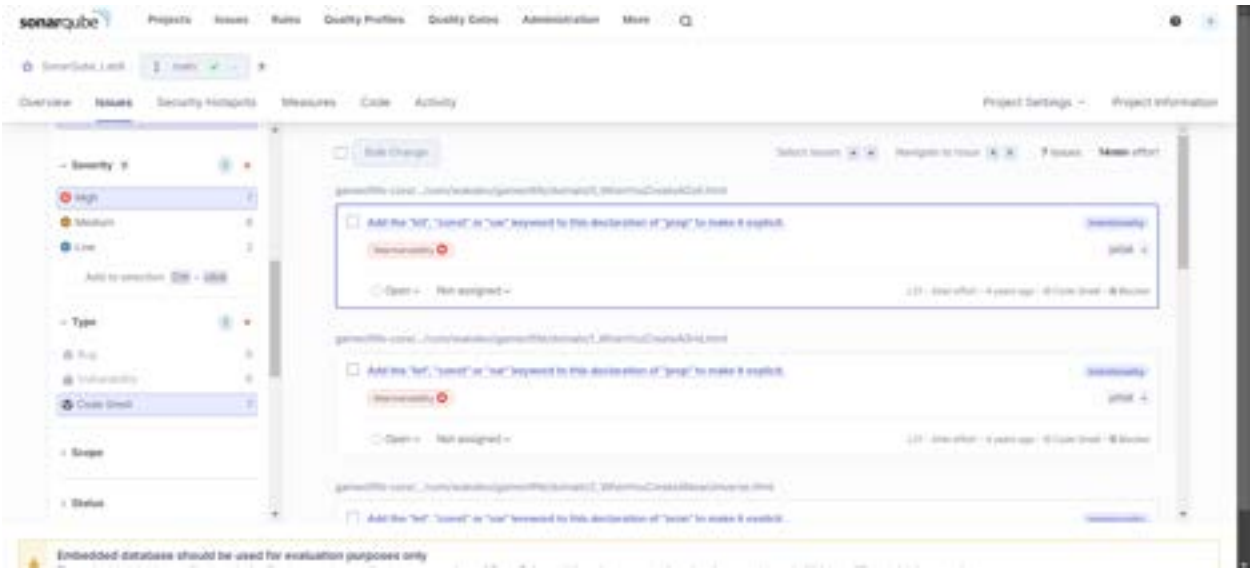


2) Reliability



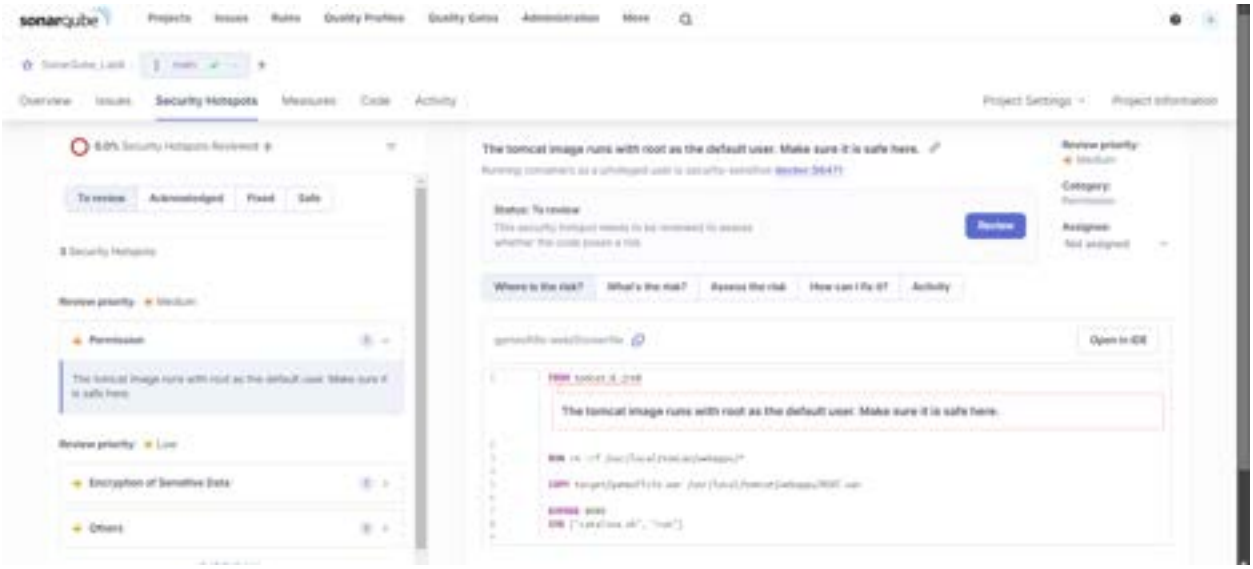
3) Maintainability

4) Severity

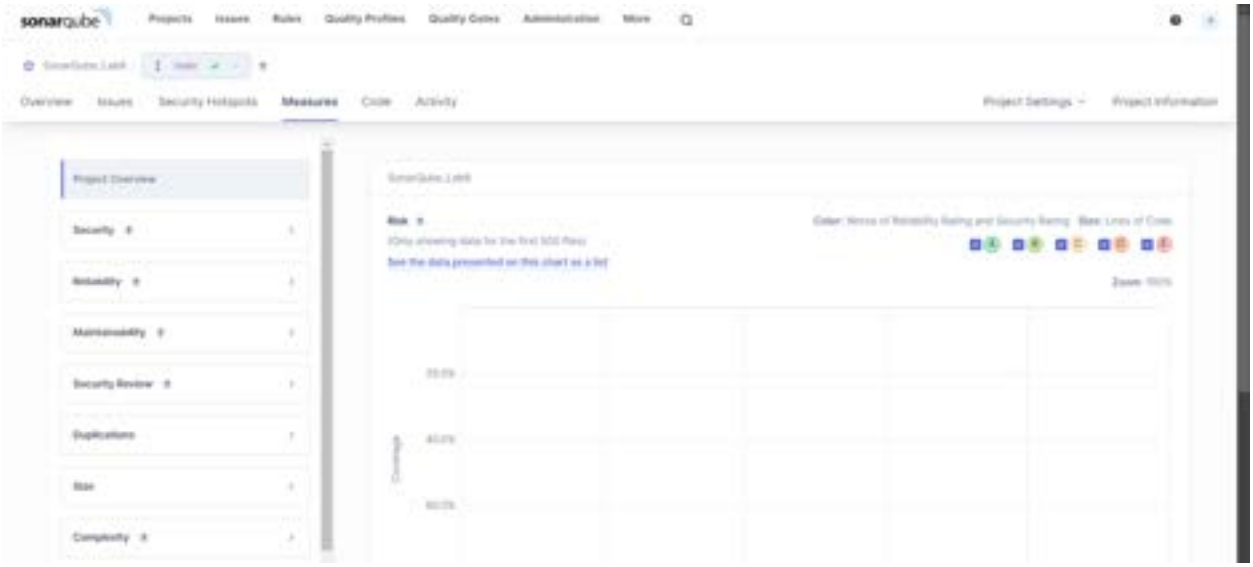


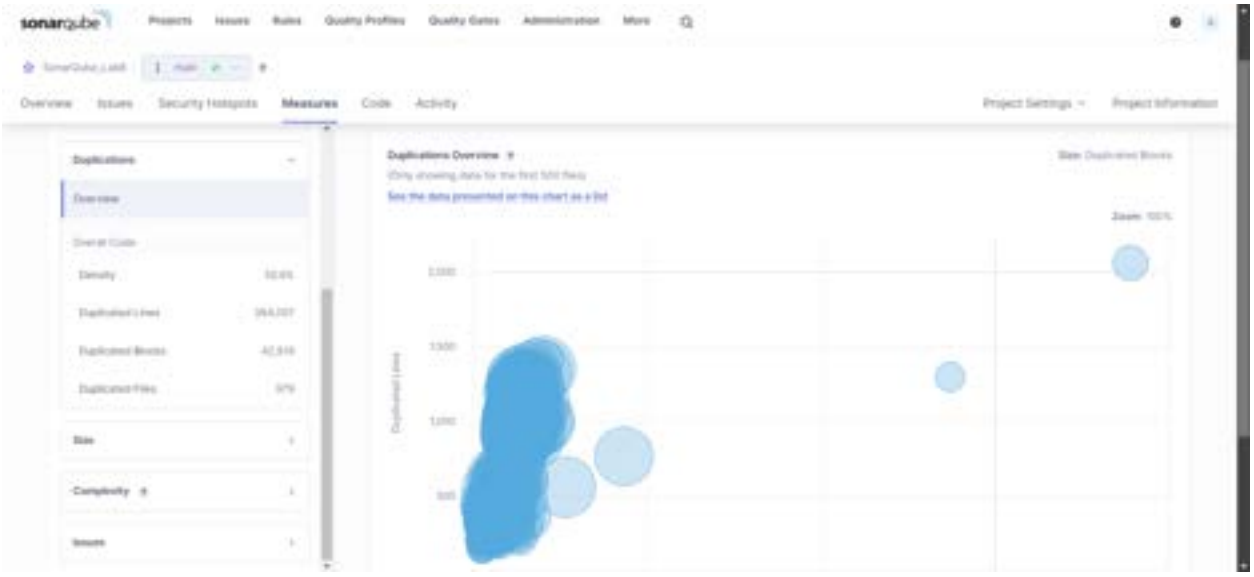
Name: Shivam.R.Prajapati Div: D15C Roll No:41 Academic Year: 2024-25

UNDER SECURITY HOTSPOT:



UNDER MEASURES:





CONCLUSION:

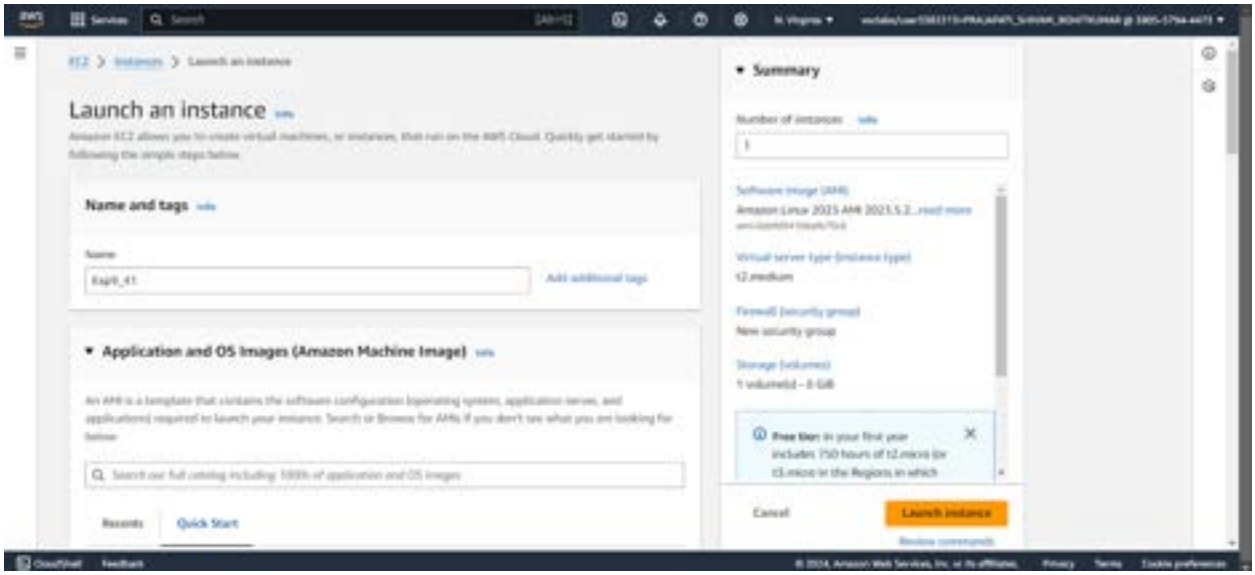
In this experiment, we demonstrated how to perform static code analysis using Jenkins CI/CD Pipeline with SonarQube integration. We created a pipeline project with a specific script that contains all the instructions necessary to run SonarQube analysis. After configuring Jenkins appropriately, we built the project. The analyzed code in this experiment had several issues, such as errors, bugs, and duplications, all of which were detected and displayed in the linked SonarQube project.

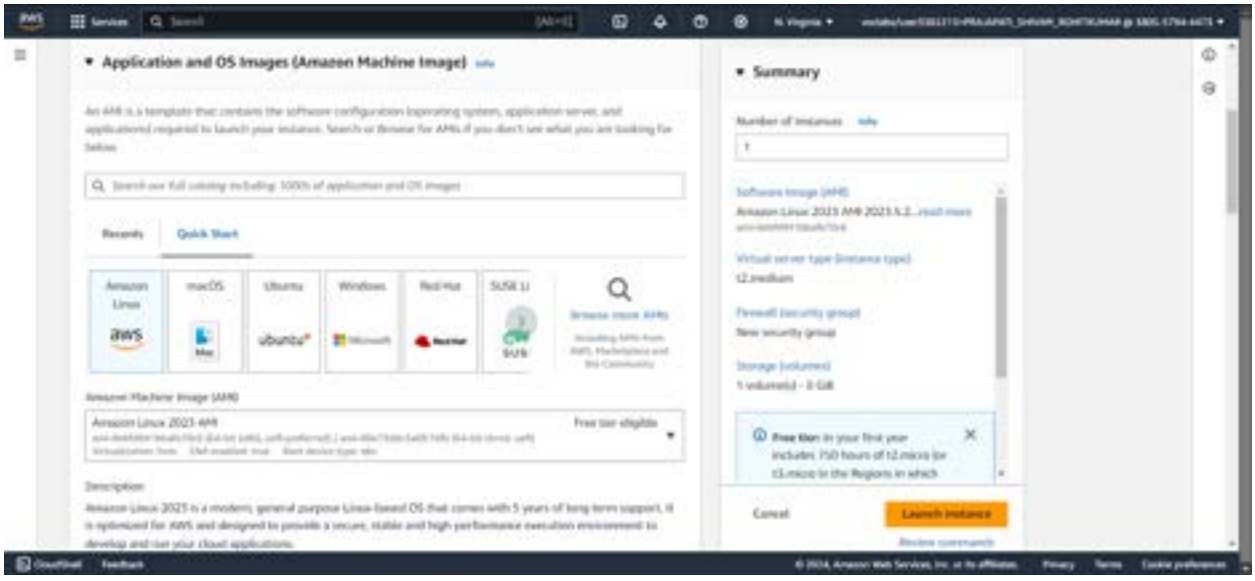
Name: Shivam.R.Prajapati Div: D15C Roll No:41 Academic Year: 2024-25

Experiment No: 9

AIM: To Understand Continuous monitoring and Installation and configuration of Nagios Core, Nagios Plugins and NRPE (Nagios Remote Plugin Executor) on Linux Machine.

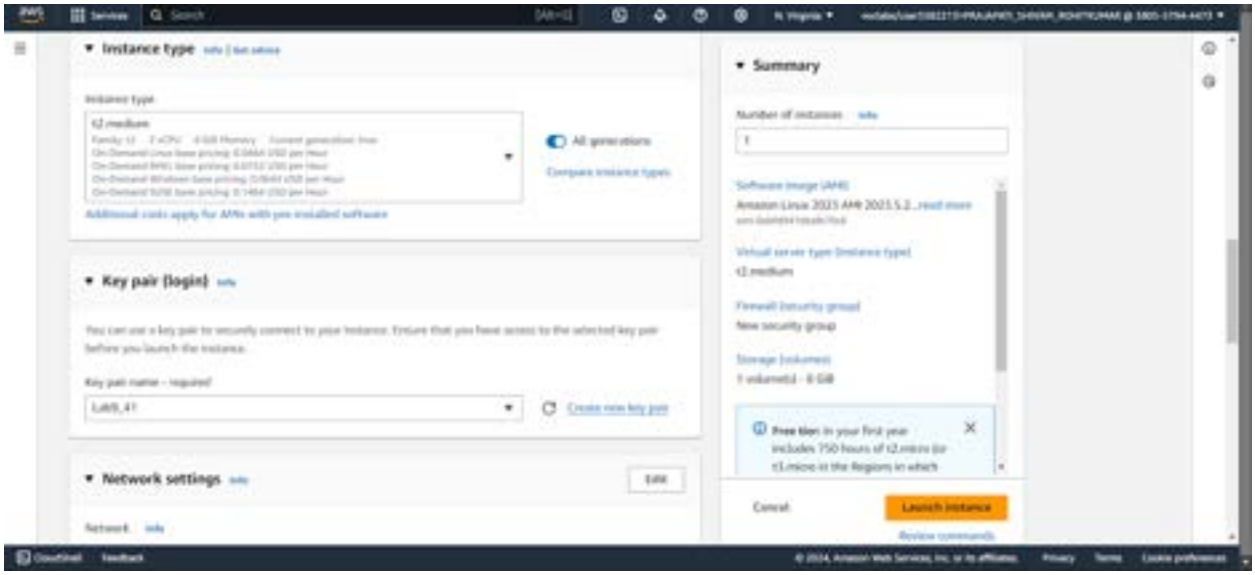
Step 1: Sign in to your AWS account. Look for EC2 in the services list. Open it and click on "Create Instance."

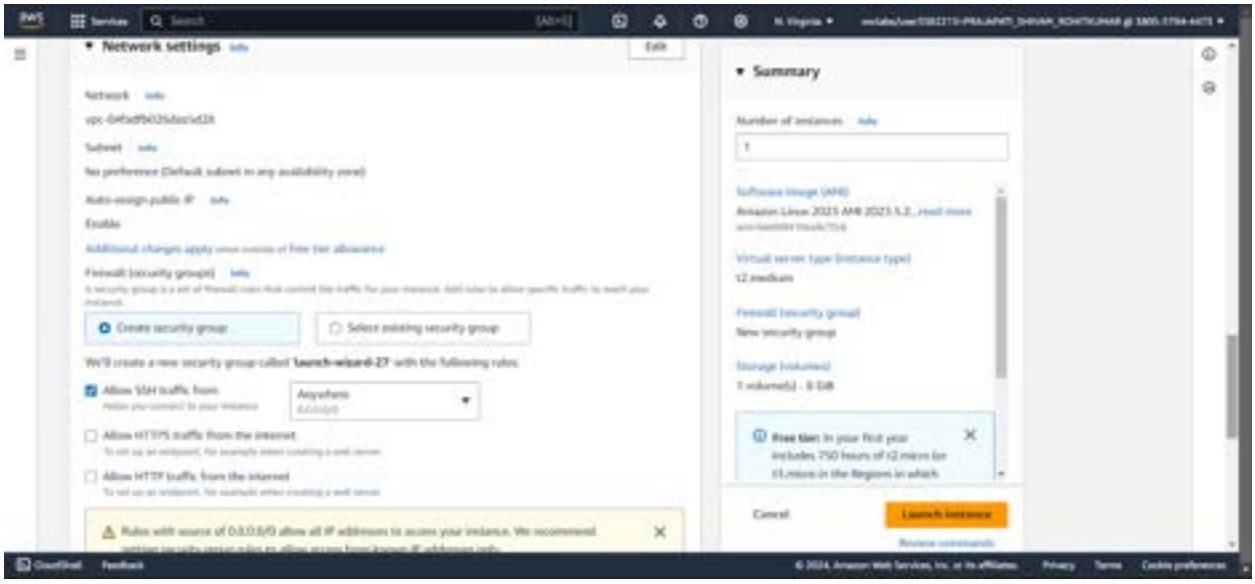




Select The OS Image as Amazon Linux.

Step 2: If you haven't created a private key or a .pem file yet, click on "**Create a key pair**." Otherwise, choose the key pair you created earlier. (Be sure to remember where the .pem file for that key is located on your system.) .in my case i have created a new one

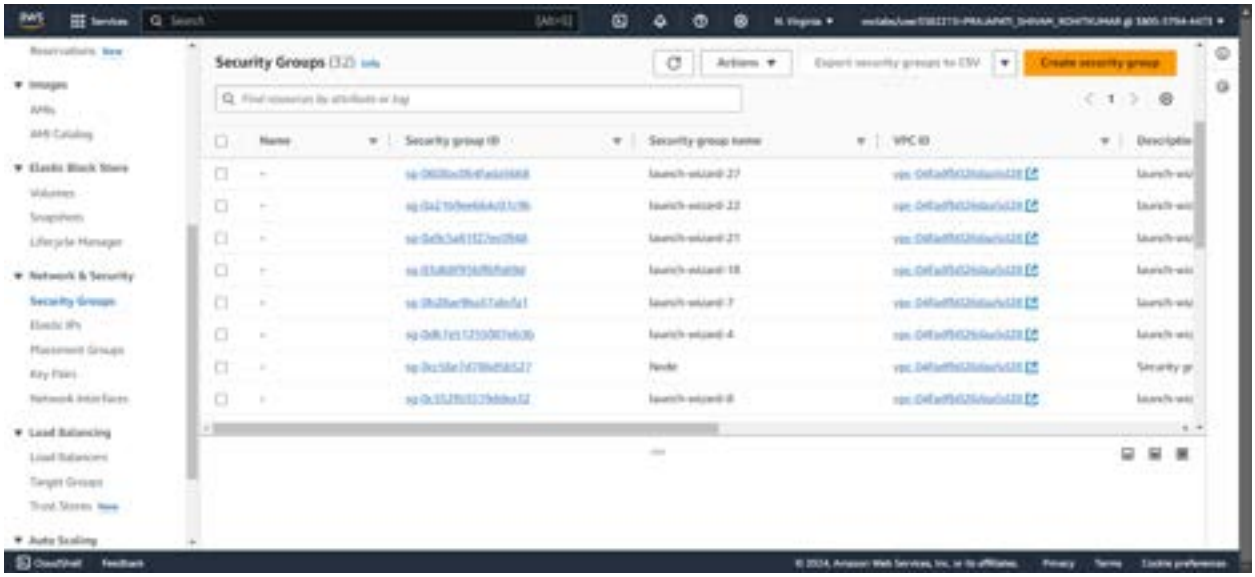




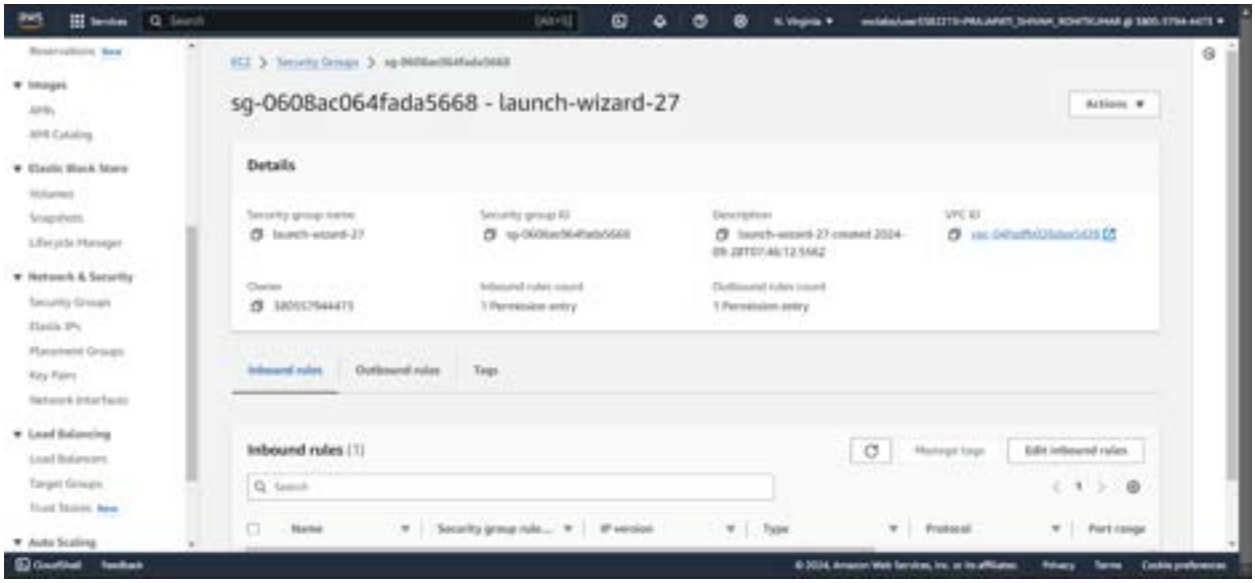
AWS will create a security group for this instance. Keep the name of that instance saved.

Instance:

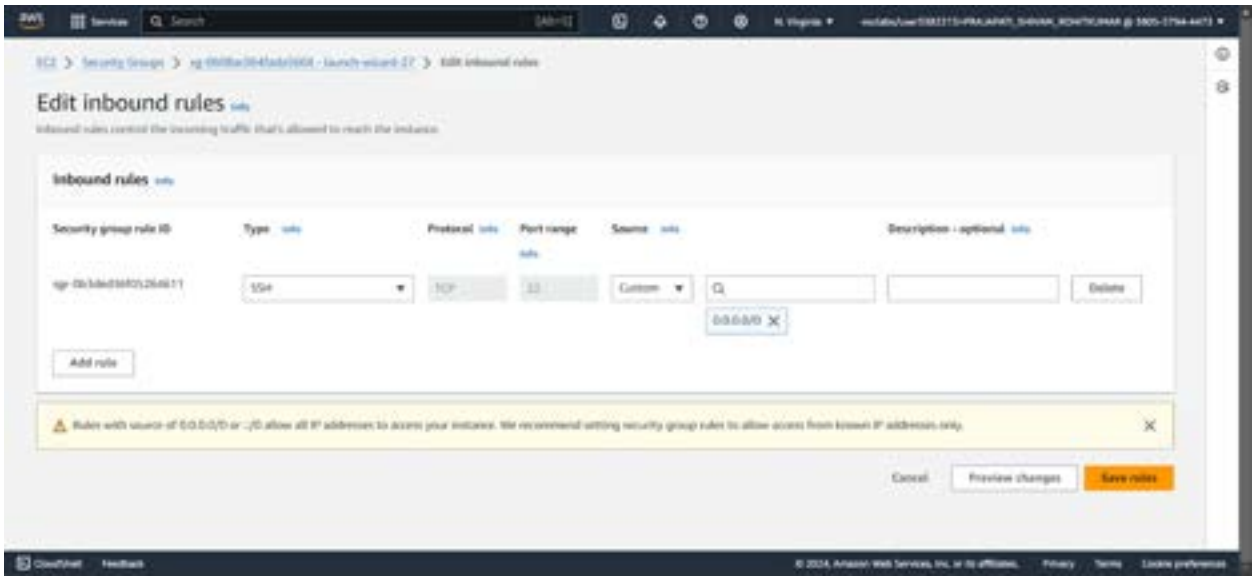
Step3: After you create the instance, click on "**Security Groups**" in the left sidebar. Look for the security group that corresponds to your instance, and then click on the security group ID for that group. (in my case launch-wizard-27 is the latest one.)



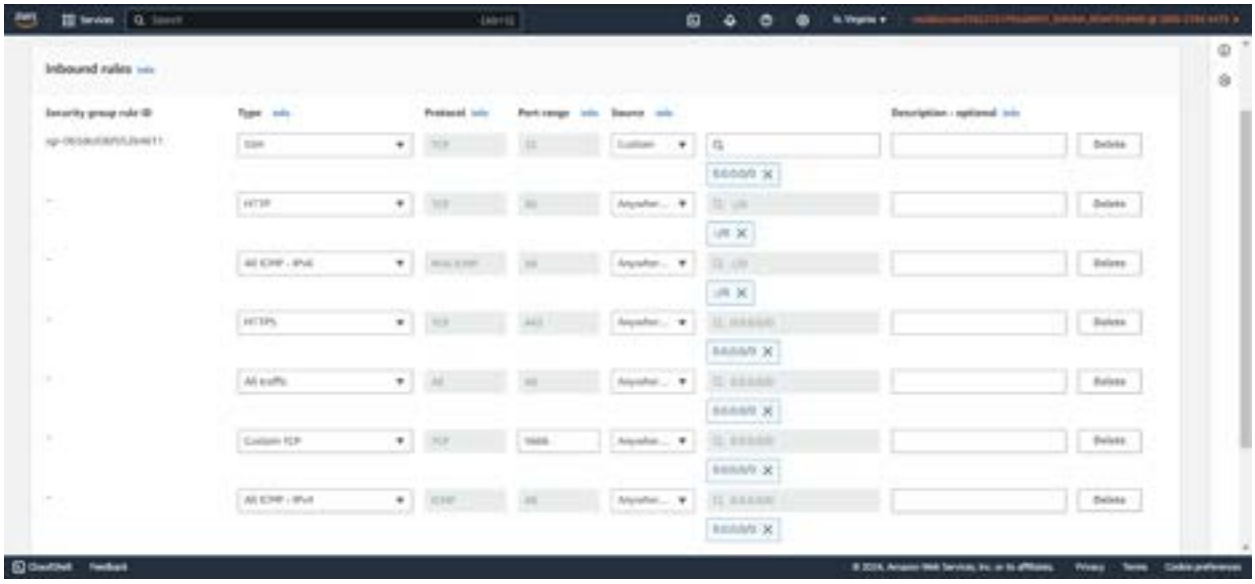
Click on Id



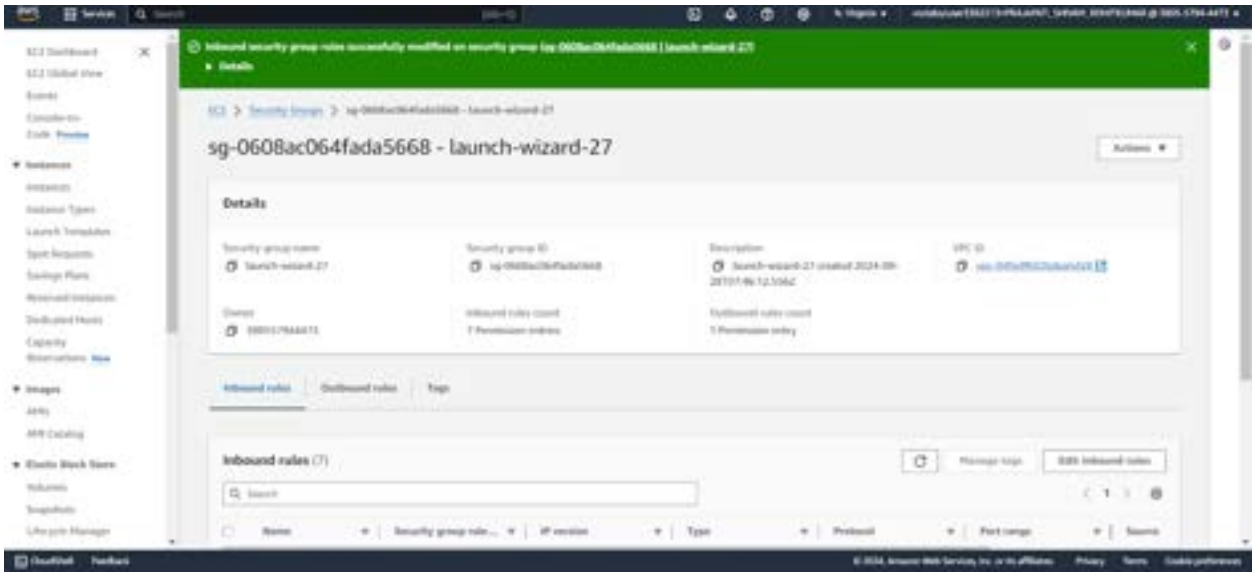
Click on the edit inbound rules

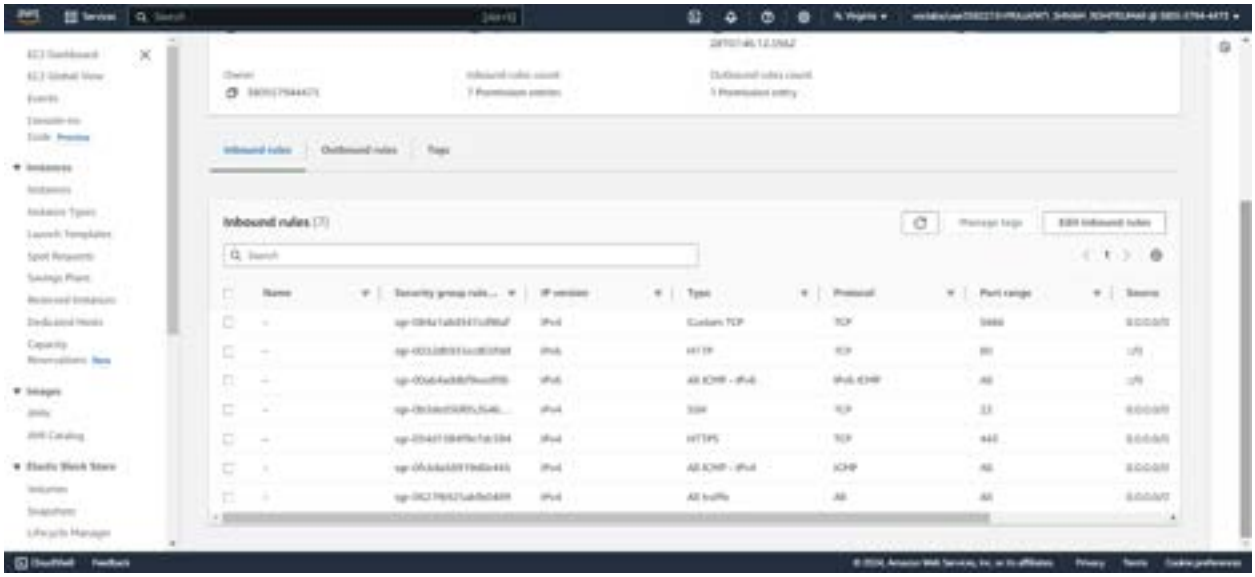


Next, click on "Add rules" and set up rules for the following protocols: HTTP, All ICMP (IPv6), HTTPS, All traffic, Custom TCP (Port 5666), and All ICMP (IPv4).

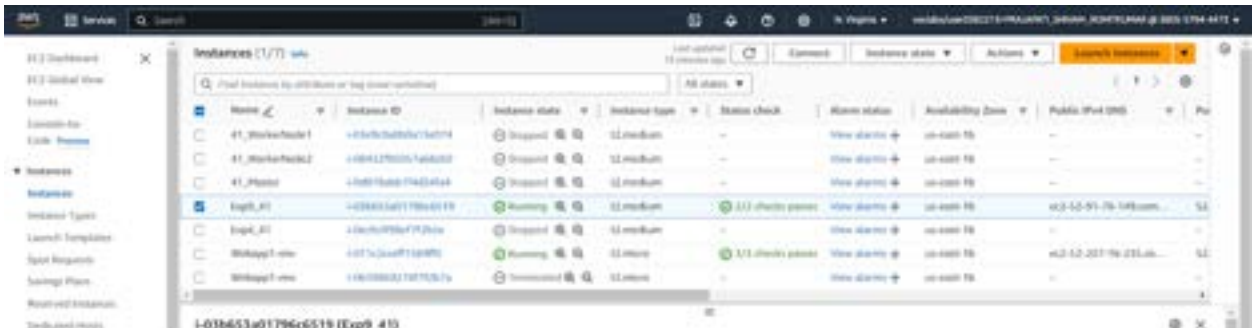


Click on save. This will add all the inbound rules to the security group.

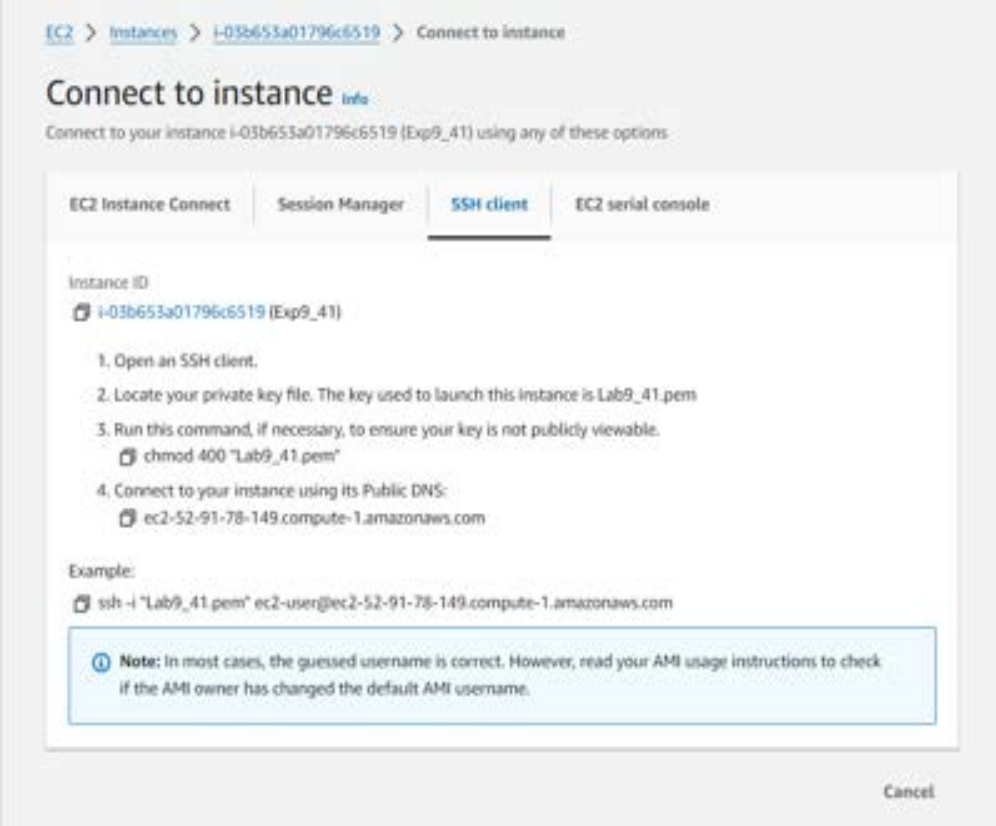




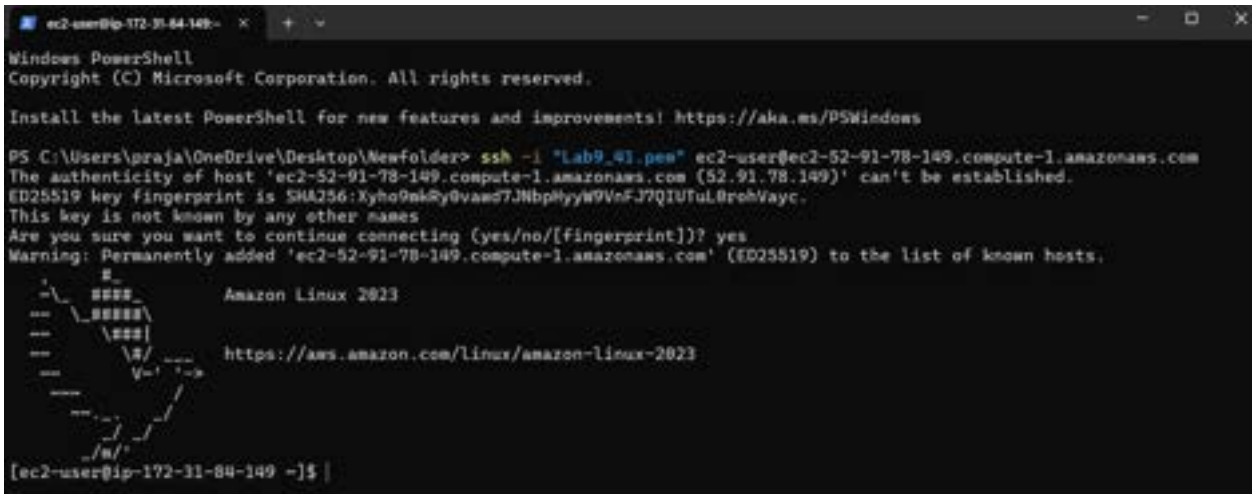
Step 4: Return to the instances screen and click on the instance ID of your instance. Then, click on "**Connect.**"



Click on "**SSH client**" and copy the example command provided.



Step 5: Now, we need to connect our local terminal to the instance using SSH. Open the terminal where your private key file (.pem) is located by actually going to the folder which has the .pem file, paste the copied SSH command, and run it.



Step 6: Now we start working on this terminal. First run the command **sudo yum update** .This command will check for any updates for the YUM library to ensure that all libraries are with up to date with the latest features and security fixes

```
[ec2-user@ip-172-31-84-149 ~]$ sudo yum update
Last metadata expiration check: 0:30:14 ago on Sat Sep 28 07:51:31 2024.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-84-149 ~]$
```

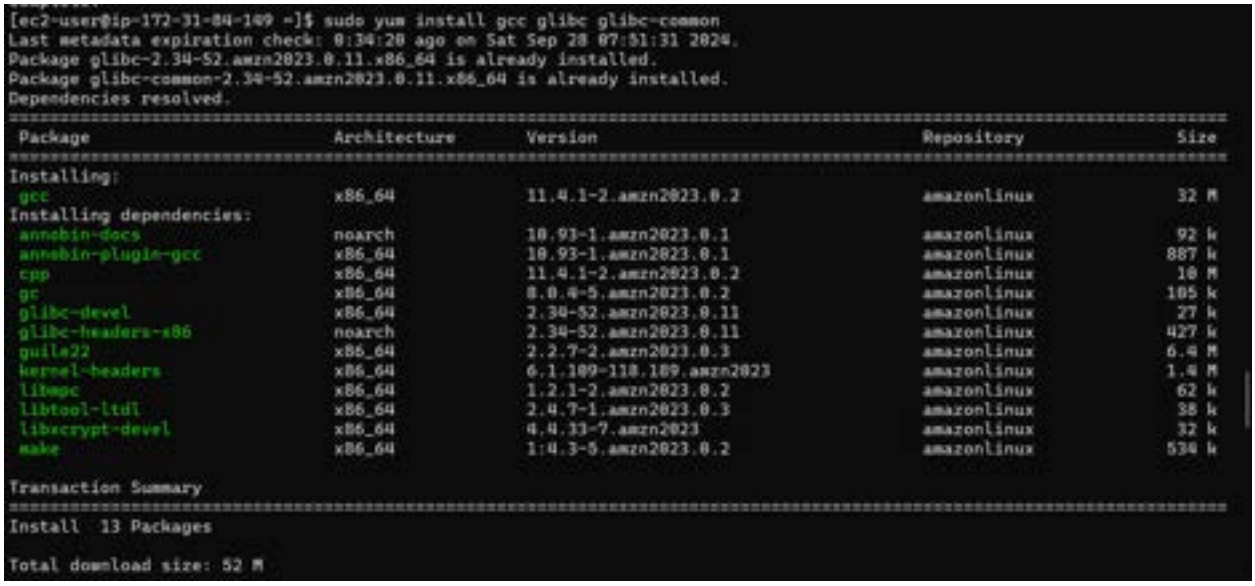
Step 7: We are going to set up a web server software application called **Apache** and a programming language called **PHP** in this instance . To do this, run this command in your terminal

sudo yum install httpd php

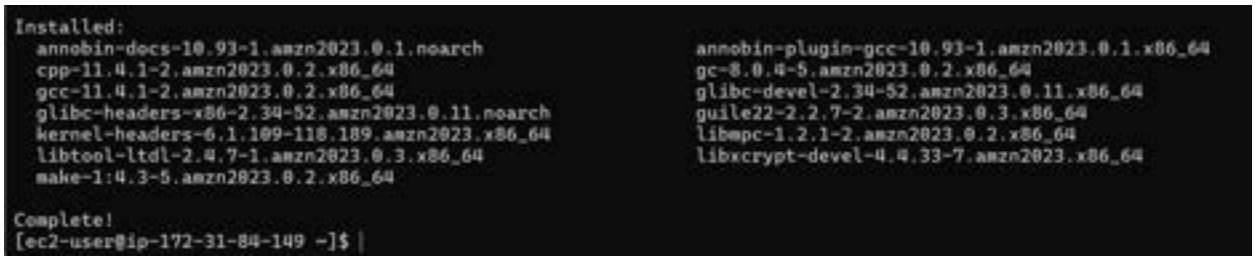
```
[ec2-user@ip-172-31-84-149 ~]$ sudo yum install httpd php
Last metadata expiration check: 0:31:34 ago on Sat Sep 28 07:51:31 2024.
Dependencies resolved.
================================================================================
 Package                 Architecture  Version                        Repository      Size
================================================================================
Installing:
 httpd                   x86_64        2.4.62-1.amzn2023              amazonlinux     48 k
 php8.3                  x86_64        8.3.10-1.amzn2023.0.1          amazonlinux     10 k
Installing dependencies:
 apr                     x86_64        1.7.2-2.amzn2023.0.2           amazonlinux    129 k
 apr-util                x86_64        1.6.3-1.amzn2023.0.1           amazonlinux     98 k
 generic-logos-httpd     noarch        18.0.0-12.amzn2023.0.3         amazonlinux     19 k
 httpd-core              x86_64        2.4.62-1.amzn2023              amazonlinux    1.4 M
 httpd-filesystem        noarch        2.4.62-1.amzn2023              amazonlinux     14 k
 httpd-tools             x86_64        2.4.62-1.amzn2023              amazonlinux     81 k
 libbrotli               x86_64        1.0.9-4.amzn2023.0.2           amazonlinux    315 k
 libsodium               x86_64        1.0.19-4.amzn2023              amazonlinux    176 k
 libxslt                 x86_64        1.1.34-5.amzn2023.0.2          amazonlinux    241 k
 mailcap                 noarch        2.1.49-3.amzn2023.0.3          amazonlinux     33 k
 nginx-filesystem        noarch        1:1.24.0-1.amzn2023.0.4        amazonlinux    9.8 k
 php8.3-cli              x86_64        8.3.10-1.amzn2023.0.1          amazonlinux    3.7 M
 php8.3-common           x86_64        8.3.10-1.amzn2023.0.1          amazonlinux    737 k
 php8.3-process          x86_64        8.3.10-1.amzn2023.0.1          amazonlinux     45 k
 php8.3-xml              x86_64        8.3.10-1.amzn2023.0.1          amazonlinux    154 k
Installing weak dependencies:
 apr-util-openssl        x86_64        1.6.3-1.amzn2023.0.1           amazonlinux     17 k
 mod_http2               x86_64        2.0.27-1.amzn2023.0.3          amazonlinux    166 k
 mod_lua                 x86_64        2.4.62-1.amzn2023              amazonlinux     61 k
```

```
Installed:
  apr-1.7.2-2.amzn2023.0.2.x86_64
  apr-util-1.6.3-1.amzn2023.0.1.x86_64
  apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64
  generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch
  httpd-2.4.62-1.amzn2023.x86_64
  httpd-core-2.4.62-1.amzn2023.x86_64
  httpd-filesystem-2.4.62-1.amzn2023.noarch
  httpd-tools-2.4.62-1.amzn2023.x86_64
  libbrotli-1.0.9-4.amzn2023.0.2.x86_64
  libsodium-1.0.19-4.amzn2023.x86_64
  libxslt-1.1.34-5.amzn2023.0.2.x86_64
  mailcap-2.1.49-3.amzn2023.0.3.noarch
  mod_http2-2.0.27-1.amzn2023.0.3.x86_64
  mod_lua-2.4.62-1.amzn2023.x86_64
  nginx-filesystem-1:1.24.0-1.amzn2023.0.4.noarch
  php8.3-8.3.10-1.amzn2023.0.1.x86_64
  php8.3-cli-8.3.10-1.amzn2023.0.1.x86_64
  php8.3-common-8.3.10-1.amzn2023.0.1.x86_64
  php8.3-fpm-8.3.10-1.amzn2023.0.1.x86_64
  php8.3-mbstring-8.3.10-1.amzn2023.0.1.x86_64
  php8.3-opcache-8.3.10-1.amzn2023.0.1.x86_64
  php8.3-pdo-8.3.10-1.amzn2023.0.1.x86_64
  php8.3-process-8.3.10-1.amzn2023.0.1.x86_64
  php8.3-sodium-8.3.10-1.amzn2023.0.1.x86_64
  php8.3-xml-8.3.10-1.amzn2023.0.1.x86_64

Complete!
[ec2-user@ip-172-31-84-149 ~]$
```

Step 8: Now, we will **install the GCC compiler**, which is used for compiling and running C and C++ programs, along with the essential C libraries. To do this, enter the following command: **sudo yum install gcc glibc glibc-common**

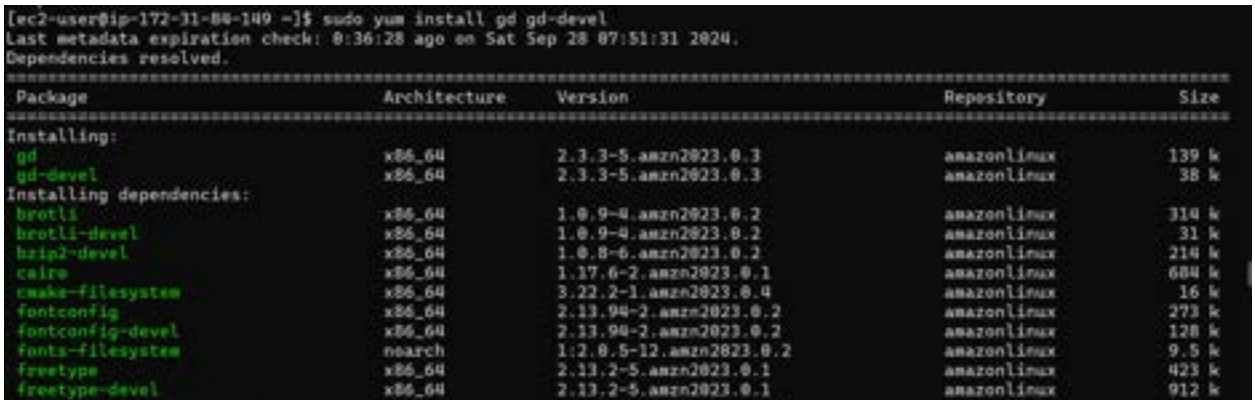


```
[ec2-user@ip-172-31-84-149 ~]$ sudo yum install gcc glibc glibc-common
Last metadata expiration check: 0:34:20 ago on Sat Sep 28 07:51:31 2024.
Package glibc-2.34-52.amzn2023.0.11.x86_64 is already installed.
Package glibc-common-2.34-52.amzn2023.0.11.x86_64 is already installed.
Dependencies resolved.
================================================================================
 Package                Architecture    Version                      Repository      Size
================================================================================
Installing:
 gcc                    x86_64          11.4.1-2.amzn2023.0.2        amazonlinux     32 M
Installing dependencies:
 annobin-docs           noarch          10.93-1.amzn2023.0.1         amazonlinux     92 k
 annobin-plugin-gcc     x86_64          10.93-1.amzn2023.0.1         amazonlinux    887 k
 cpp                    x86_64          11.4.1-2.amzn2023.0.2        amazonlinux     10 M
 gc                     x86_64          8.0.4-5.amzn2023.0.2         amazonlinux    105 k
 glibc-devel            x86_64          2.34-52.amzn2023.0.11        amazonlinux     27 k
 glibc-headers-x86      noarch          2.34-52.amzn2023.0.11        amazonlinux    427 k
 guile22                x86_64          2.2.7-2.amzn2023.0.3         amazonlinux    6.4 M
 kernel-headers         x86_64          6.1.109-118.189.amzn2023     amazonlinux    1.4 M
 libmpc                 x86_64          1.2.1-2.amzn2023.0.2         amazonlinux     62 k
 libtool-ltdl           x86_64          2.4.7-1.amzn2023.0.3         amazonlinux     38 k
 libxcrypt-devel        x86_64          4.4.33-7.amzn2023            amazonlinux     32 k
 make                   x86_64          1:4.3-5.amzn2023.0.2         amazonlinux    534 k

Transaction Summary
================================================================================
Install  13 Packages

Total download size: 52 M
```

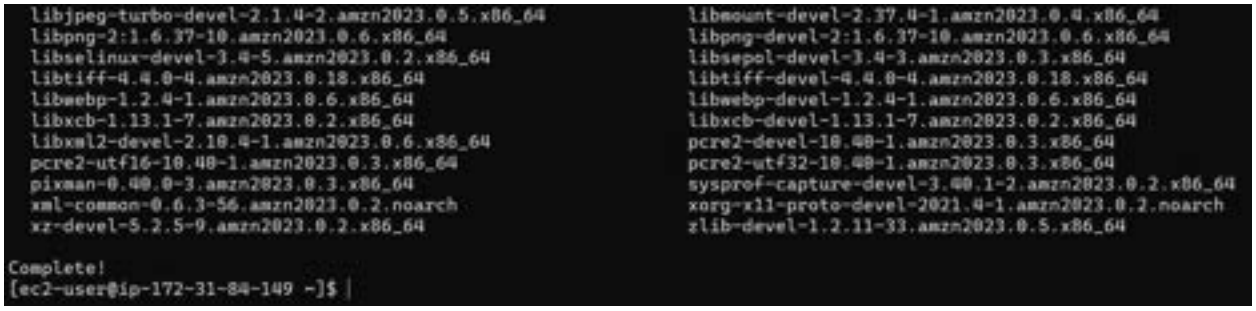


```
Installed:
  annobin-docs-10.93-1.amzn2023.0.1.noarch
  annobin-plugin-gcc-10.93-1.amzn2023.0.1.x86_64
  cpp-11.4.1-2.amzn2023.0.2.x86_64
  gc-8.0.4-5.amzn2023.0.2.x86_64
  gcc-11.4.1-2.amzn2023.0.2.x86_64
  glibc-devel-2.34-52.amzn2023.0.11.x86_64
  glibc-headers-x86-2.34-52.amzn2023.0.11.noarch
  guile22-2.2.7-2.amzn2023.0.3.x86_64
  kernel-headers-6.1.109-118.189.amzn2023.x86_64
  libmpc-1.2.1-2.amzn2023.0.2.x86_64
  libtool-ltdl-2.4.7-1.amzn2023.0.3.x86_64
  libxcrypt-devel-4.4.33-7.amzn2023.x86_64
  make-1:4.3-5.amzn2023.0.2.x86_64

Complete!
[ec2-user@ip-172-31-84-149 ~]$
```

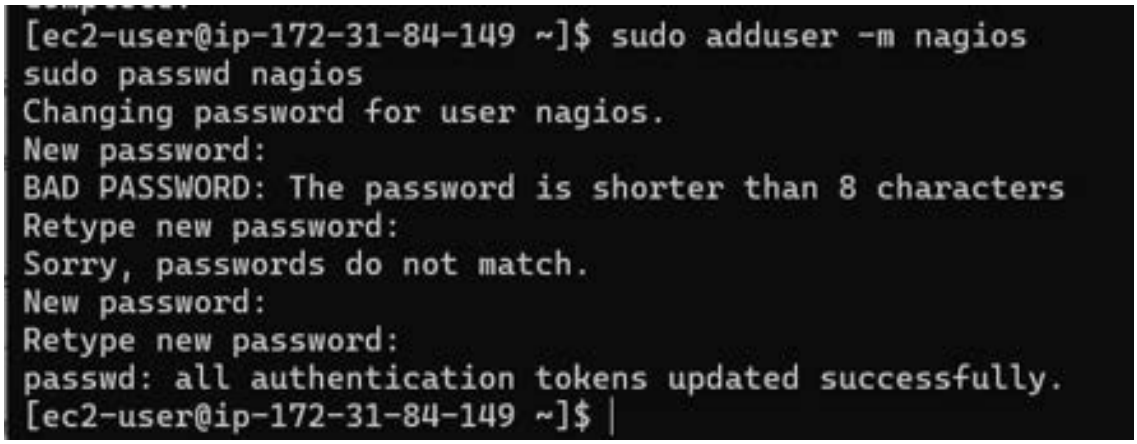
Step 9: Next, we need to **install the GD library**, along with its development tools. This library helps with creating and manipulating images. For that, run this command **sudo yum install gd gd-devel**



```
[ec2-user@ip-172-31-84-149 ~]$ sudo yum install gd gd-devel
Last metadata expiration check: 0:36:28 ago on Sat Sep 28 07:51:31 2024.
Dependencies resolved.
================================================================================
 Package                Architecture    Version                      Repository      Size
================================================================================
Installing:
 gd                     x86_64          2.3.3-5.amzn2023.0.3         amazonlinux    139 k
 gd-devel               x86_64          2.3.3-5.amzn2023.0.3         amazonlinux     38 k
Installing dependencies:
 brotli                 x86_64          1.0.9-4.amzn2023.0.2         amazonlinux    314 k
 brotli-devel           x86_64          1.0.9-4.amzn2023.0.2         amazonlinux     31 k
 bzip2-devel            x86_64          1.0.8-6.amzn2023.0.2         amazonlinux    214 k
 cairo                  x86_64          1.17.6-2.amzn2023.0.1        amazonlinux    684 k
 cmake-filesystem       x86_64          3.22.2-1.amzn2023.0.4        amazonlinux     16 k
 fontconfig             x86_64          2.13.94-2.amzn2023.0.2       amazonlinux    273 k
 fontconfig-devel       x86_64          2.13.94-2.amzn2023.0.2       amazonlinux    128 k
 fonts-filesystem       noarch          1:2.0.5-12.amzn2023.0.2      amazonlinux    9.5 k
 freetype               x86_64          2.13.2-5.amzn2023.0.1        amazonlinux    423 k
 freetype-devel         x86_64          2.13.2-5.amzn2023.0.1        amazonlinux    912 k
```

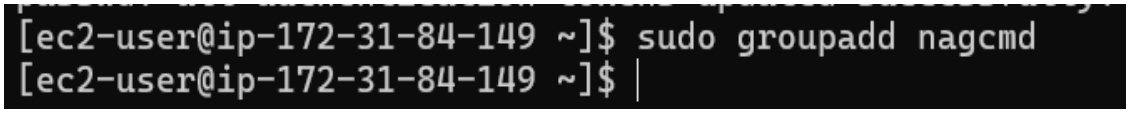


Step 10: Now, we create a user called '**nagios**' and make sure that it has a home directory, and set up a password for it.
sudo adduser -m nagios
sudo passwd nagios



Password :Shivam2@

Step 11: Now, we need to create a user group named **nagcmd**, which will be used to execute Nagios commands. To do this, run the following command: **sudo groupadd nagcmd**



Step 12: Next, we'll **add the users apache and nagios to the nagcmd group**. This allows them to execute Nagios commands.
sudo usermod -a -G nagcmd nagios
sudo usermod -a -G nagcmd apache

```
[ec2-user@ip-172-31-84-149 ~]$ sudo usermod -a -G nagcmd nagios
sudo usermod -a -G nagcmd apache
[ec2-user@ip-172-31-84-149 ~]$
```

Step 13: We'll create a directory called **downloads** to store the files related to the Nagios server that we download.

mkdir ~/downloads

cd ~/downloads

```
[ec2-user@ip-172-31-84-149 ~]$ mkdir ~/downloads
cd ~/downloads
[ec2-user@ip-172-31-84-149 downloads]$
```

Step 14: Now we need to install the latest versions of nagios-core and nagios-plugins. Go to the respective websites and check whether a better version is available.If newer versions are available, then right click on the download button → Copy link address. Paste this link address in place of the current link in command. If not run these commands.

wget https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.5.tar.gz

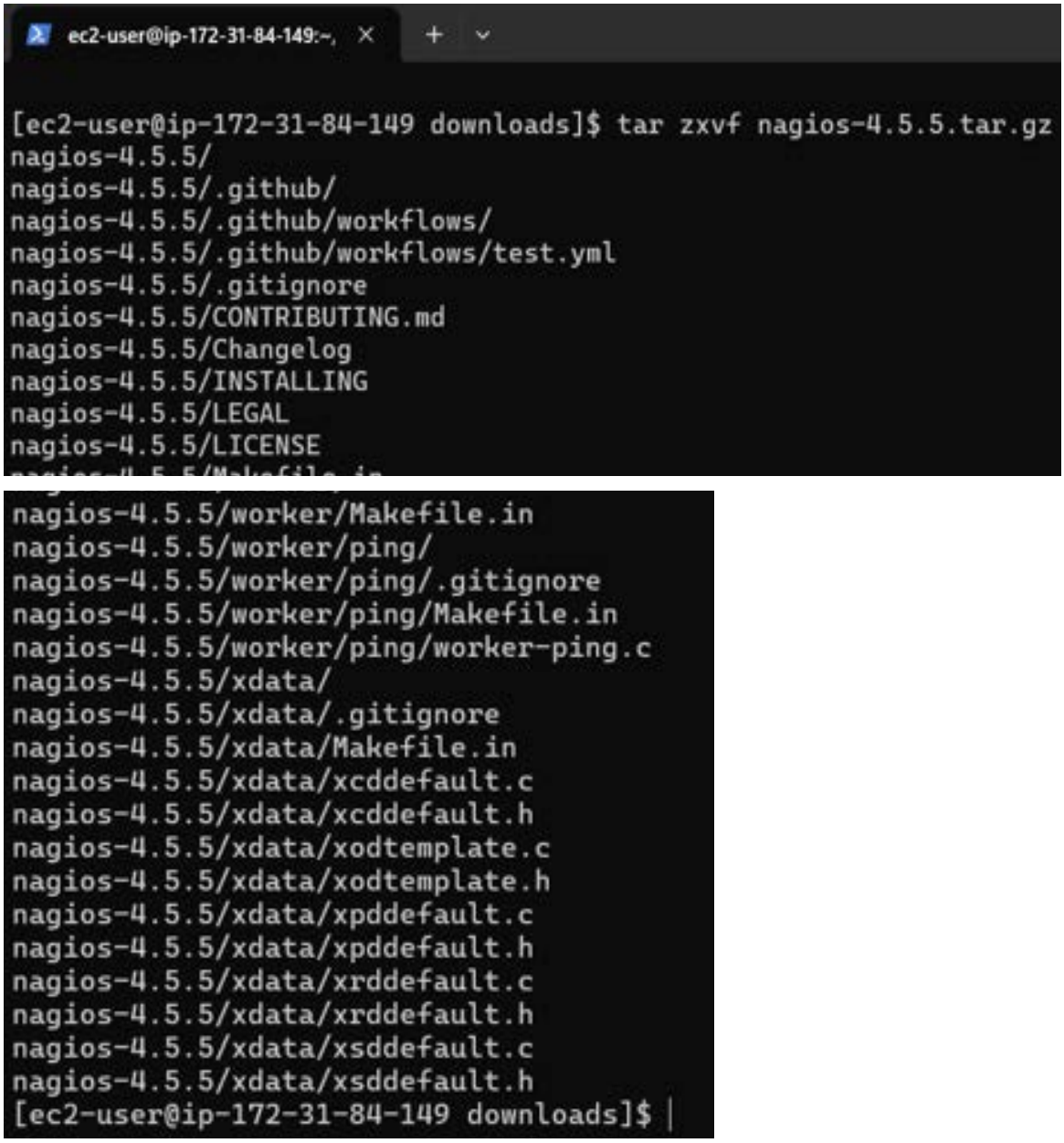
```
[ec2-user@ip-172-31-84-149 downloads]$ wget https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.5.tar.gz
--2024-09-28 08:37:22--  https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.5.tar.gz
Resolving assets.nagios.com (assets.nagios.com)... 45.79.49.120, 2600:3c00::f03c:92ff:fef7:45ce
Connecting to assets.nagios.com (assets.nagios.com)|45.79.49.120|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2065473 (2.0M) [application/x-gzip]
Saving to: 'nagios-4.5.5.tar.gz'

nagios-4.5.5.tar.gz          100%[=============================================>]   1.97M  5.07MB/s    in 0.4s

2024-09-28 08:37:23 (5.07 MB/s) - 'nagios-4.5.5.tar.gz' saved [2065473/2065473]

[ec2-user@ip-172-31-84-149 downloads]$
```

wget https://nagios-plugins.org/download/nagios-plugins-2.4.11.tar.gz

```
[ec2-user@ip-172-31-84-149 downloads]$ wget https://nagios-plugins.org/download/nagios-plugins-2.4.11.tar.gz
--2024-09-28 08:38:31--  https://nagios-plugins.org/download/nagios-plugins-2.4.11.tar.gz
Resolving nagios-plugins.org (nagios-plugins.org)... 45.56.123.251
Connecting to nagios-plugins.org (nagios-plugins.org)|45.56.123.251|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2753049 (2.6M) [application/x-gzip]
Saving to: 'nagios-plugins-2.4.11.tar.gz'

nagios-plugins-2.4.11.tar.gz  100%[=============================================>]   2.62M  7.16MB/s    in 0.4s

2024-09-28 08:38:32 (7.16 MB/s) - 'nagios-plugins-2.4.11.tar.gz' saved [2753049/2753049]

[ec2-user@ip-172-31-84-149 downloads]$
```

Step 15: Now, we need to extract the Nagios Core file into the same directory. We can do this using the tar command.

tar zxvf nagios-4.5.5.tar.gz



Step16: Now, we need to ensure that Nagios uses the nagcmd group for executing external commands.

./configure --with-command-group=nagcmd

```
[ec2-user@ip-172-31-84-149 downloads]$ ./configure --with-command-group=nagcmd
-bash: ./configure: No such file or directory
[ec2-user@ip-172-31-84-149 downloads]$
```

An error was encountered here: ./configure: no such path or directory . So Navigate to the nagios-4.5.5 folder in downloads. (version could vary)

ls :

```
[ec2-user@ip-172-31-84-149 downloads]$ ls
nagios-4.5.5  nagios-4.5.5.tar.gz  nagios-plugins-2.4.11.tar.gz
[ec2-user@ip-172-31-84-149 downloads]$
```

- cd nagios-4.5.5 (use the version shown by your ls command)
- ./configure --with-command-group=nagcmd

Another error could be Cannot find SSL headers. To solve this, we need to install OpenSSL Dev Library : **sudo yum install openssl-devel**

```
Last metadata expiration check: 0:58:10 ago on Sat Sep 28 07:51:31 2024.
Dependencies resolved.
================================================================================
 Package                 Architecture        Version                  Repository                 Size
================================================================================
Installing:
 openssl-devel           x86_64              1:3.0.8-1.amzn2023.0.14  amazonlinux               3.0 M

Transaction Summary
================================================================================
Install  1 Package

Total download size: 3.0 M
Installed size: 4.7 M
Is this ok [y/N]: y
Downloading Packages:
openssl-devel-3.0.8-1.amzn2023.0.14.x86_64.rpm                  18 MB/s | 3.0 MB     00:00
--------------------------------------------------------------------------------
Total                                                           14 MB/s | 3.0 MB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
```

./configure --with-command-group=nagcmd

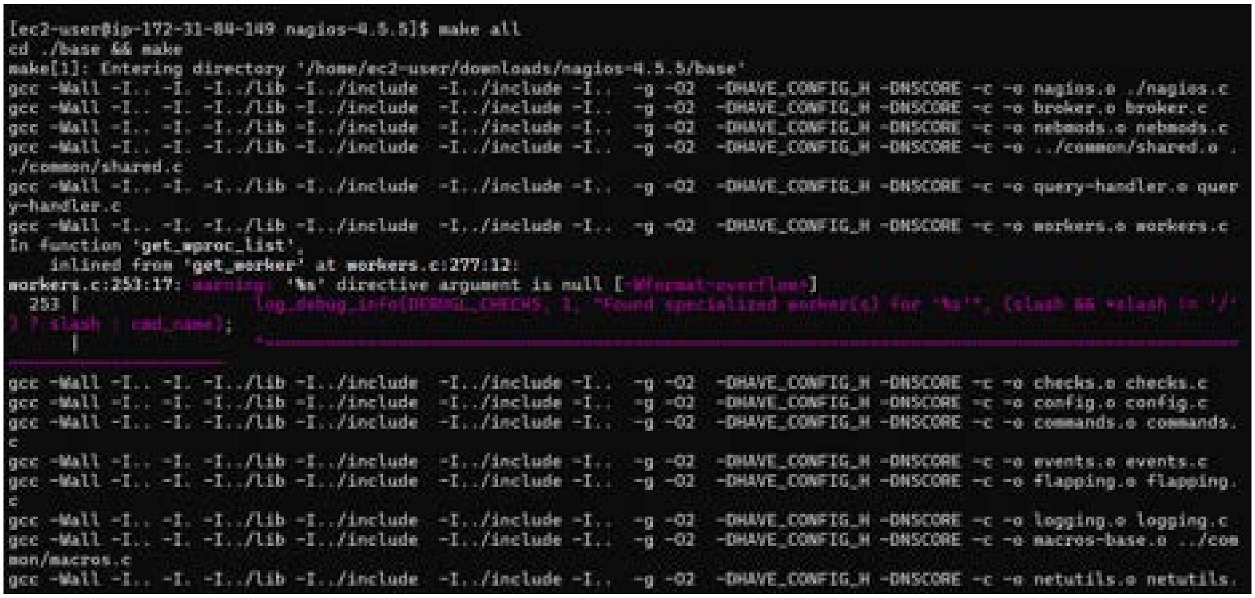
```
[ec2-user@ip-172-31-84-149 nagios-4.5.5]$ ./configure --with-command-group=nagcmd
checking for a BSD-compatible install... /usr/bin/install -c
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether the compiler supports GNU C... yes
checking whether gcc accepts -g... yes
checking for gcc option to enable C11 features... none needed
checking whether make sets $(MAKE)... yes
checking whether ln -s works... yes
checking for strip... /usr/bin/strip
checking for sys/wait.h that is POSIX.1 compatible... yes
checking for stdio.h... yes
checking for stdlib.h... yes
checking for string.h... yes
checking for inttypes.h... yes
checking for stdint.h... yes
checking for strings.h... yes
checking for sys/stat.h... yes
checking for sys/types.h... yes
```

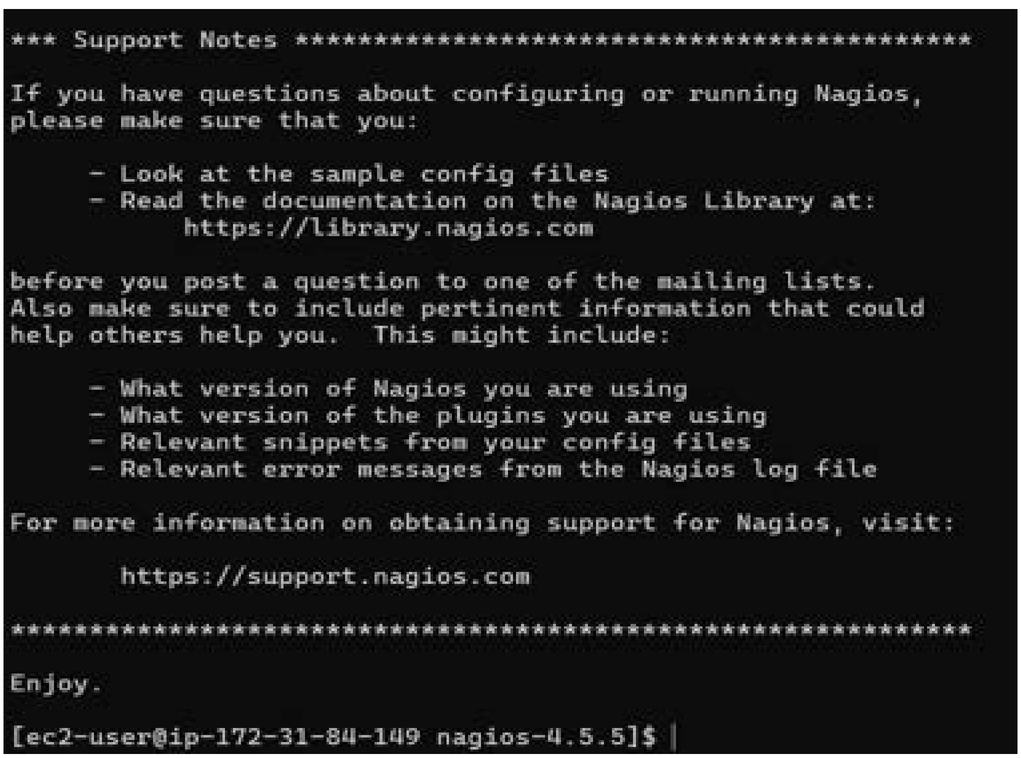
```
*** Configuration summary for nagios 4.5.5 2024-09-17 ***:

 General Options:
 -------------------------
        Nagios executable:  nagios
        Nagios user/group:  nagios,nagios
       Command user/group:  nagios,nagcmd
             Event Broker:  yes
        Install ${prefix}:  /usr/local/nagios
    Install ${includedir}:  /usr/local/nagios/include/nagios
                Lock file:  /run/nagios.lock
   Check result directory:  /usr/local/nagios/var/spool/checkresults
           Init directory:  /lib/systemd/system
  Apache conf.d directory:  /etc/httpd/conf.d
             Mail program:  /bin/mail
                  Host OS:  linux-gnu
          IOBroker Method:  epoll

 Web Interface Options:
 ------------------------
                 HTML URL:  http://localhost/nagios/
                  CGI URL:  http://localhost/nagios/cgi-bin/
 Traceroute (used by WAP):  /usr/bin/traceroute


Review the options above for accuracy.  If they look okay,
type 'make all' to compile the main program and CGIs.

[ec2-user@ip-172-31-84-149 nagios-4.5.5]$
```

Step 17: Next, we need to compile all the components of the software based on the instructions in the Makefile. For that, use this command: **make all** Then,
sudo make install

Name: Shivam.R.Prajapati Div: D15C Roll No:41 Academic Year: 2024-25

sudo make install-init
sudo make install-config
sudo make install-commandmode



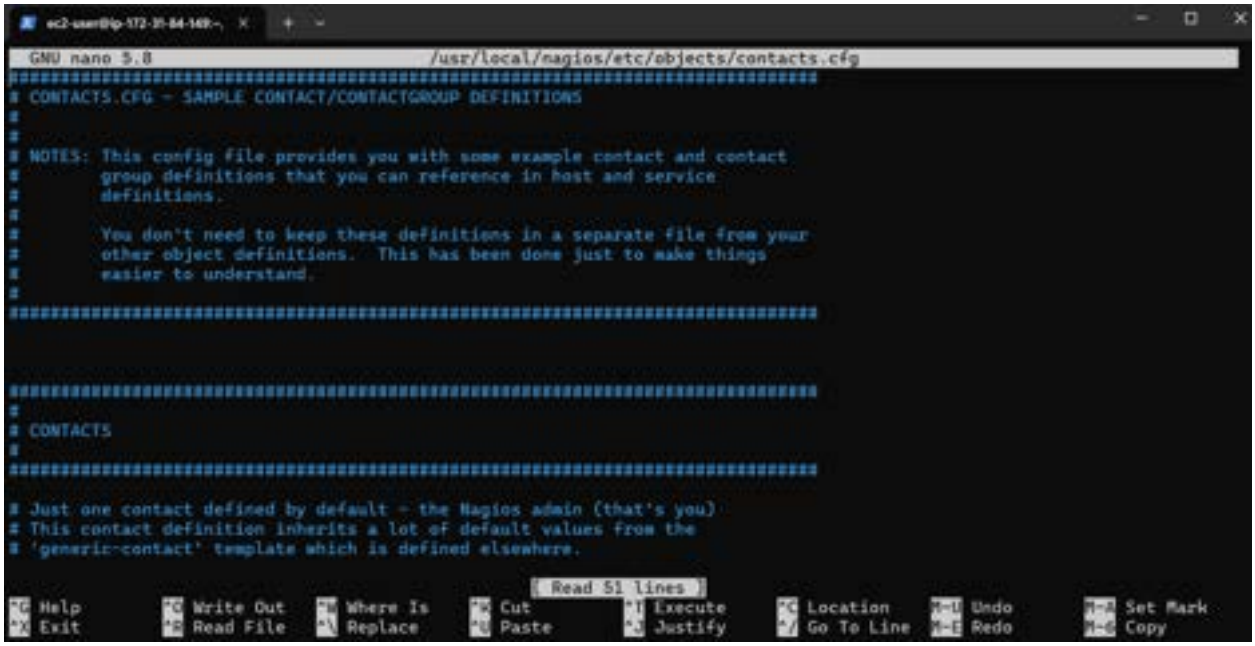


```
[ec2-user@ip-172-31-84-149 nagios-4.5.5]$ sudo make install
sudo make install-init
sudo make install-config
sudo make install-commandmode
cd ./base && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/base'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagios /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagiostats /usr/local/nagios/bin
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5/base'
cd ./cgi && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
make install-basic
make[2]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/sbin
for file in *.cgi; do \
        /usr/bin/install -c -s -m 775 -o nagios -g nagios $file /usr/local/nagios/sbin; \
done
make[2]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
cd ./html && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/html'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/media
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/stylesheets
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/contexthelp
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/docs
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/docs/images
```

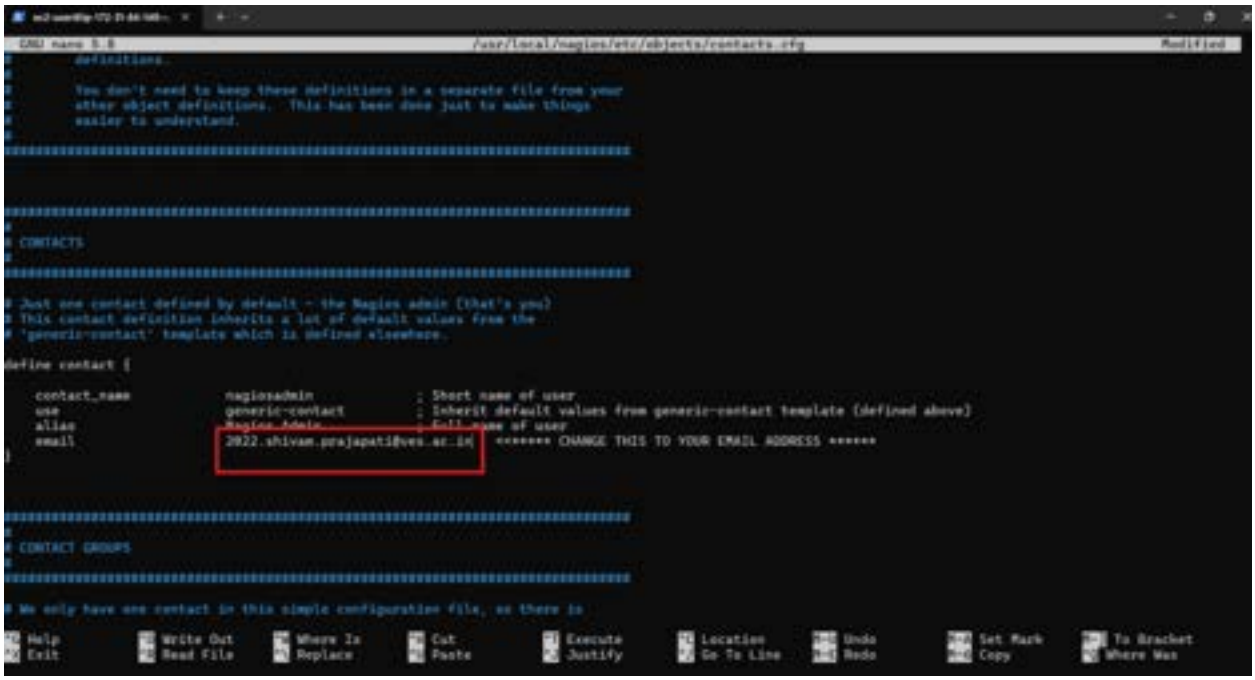
```
*** Config files installed ***

Remember, these are *SAMPLE* config files.  You'll need to read
the documentation for more information on how to actually define
services, hosts, etc. to fit your particular needs.

/usr/bin/install -c -m 775 -o nagios -g nagcmd -d /usr/local/nagios/var/rw
chmod g+s /usr/local/nagios/var/rw

*** External command directory configured ***

[ec2-user@ip-172-31-84-149 nagios-4.5.5]$
```

Step 18: We need to update the email linked with this server to our email for it to send notifications (if any needed). **sudo nano /usr/local/nagios/etc/objects/contacts.cfg**

```
[ec2-user@ip-172-31-84-149 nagios-4.5.5]$ sudo nano /usr/local/nagios/etc/objects/contacts.cfg
```



Here, change the email under ‘define contact{}’ to your email address



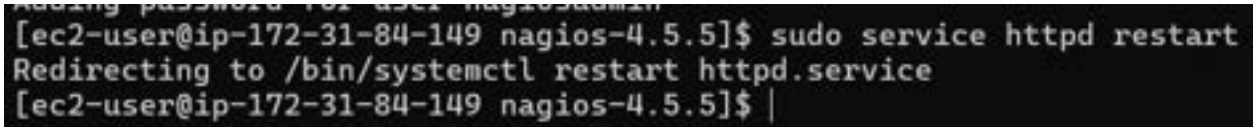
To save this use the following shortcut sequence CTRL+O→Enter→CTRL+X.
CTRL+O: Overwrite the existing file with edited file
CTRL+X: Exit nano editor

Step 19: We need to install the necessary configuration files for the Nagios web interface.
sudo make install-webconf

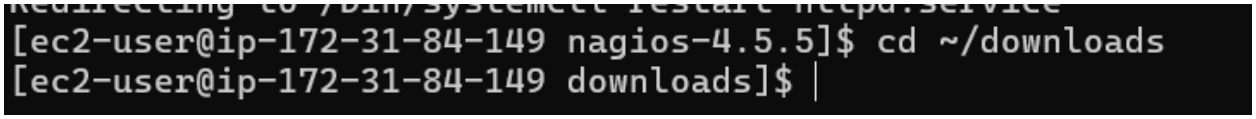
Step 20: Now we need to create a user to access the Nagios web interface. For that, run this command to create a user named '**nagiosadmin**'.Keep this username and password saved as it is needed to login to the web interface. **sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin**

Kingmaker is the password

Step 21: Now, we need to restart the Apache server to apply all the recent configurations. Use this command: **sudo service httpd restart**



Step 22: Now we go back to the downloads folder and extract the files of nagios plugin.
cd ~/downloads
tar zxvf nagios-plugins-2.4.11.tar.gz (Version may vary)



```
[ec2-user@ip-172-31-84-149 downloads]$ tar zxvf nagios-plugins-2.4.11.tar.gz
nagios-plugins-2.4.11/
nagios-plugins-2.4.11/build-aux/
nagios-plugins-2.4.11/build-aux/compile
nagios-plugins-2.4.11/build-aux/config.guess
nagios-plugins-2.4.11/build-aux/config.rpath
nagios-plugins-2.4.11/build-aux/config.sub
nagios-plugins-2.4.11/build-aux/install-sh
nagios-plugins-2.4.11/build-aux/ltmain.sh
nagios-plugins-2.4.11/build-aux/missing
nagios-plugins-2.4.11/build-aux/mkinstalldirs
nagios-plugins-2.4.11/build-aux/depcomp
nagios-plugins-2.4.11/build-aux/snippet/
nagios-plugins-2.4.11/build-aux/snippet/_Noreturn.h
nagios-plugins-2.4.11/build-aux/snippet/arg-nonnull.h
nagios-plugins-2.4.11/build-aux/snippet/c++defs.h
nagios-plugins-2.4.11/build-aux/snippet/warn-on-use.h
nagios-plugins-2.4.11/build-aux/test-driver
```

Step 23: Again, we need to install the configurations for these files.
cd nagios-plugins-2.4.11 (version may vary)

```
[ec2-user@ip-172-31-84-149 downloads]$ cd nagios-plugins-2.4.11
[ec2-user@ip-172-31-84-149 nagios-plugins-2.4.11]$
```

./configure --with-nagios-user=nagios --with-nagios-group=nagios

```
[ec2-user@ip-172-31-84-149 nagios-plugins-2.4.11]$ ./configure --with-nagios-user=nagios --with-nagios-group=nagios
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /usr/bin/mkdir -p
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking whether make supports nested variables... yes
checking whether to enable maintainer-specific portions of Makefiles... yes
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking whether gcc understands -c and -o together... yes
checking whether make supports the include directive... yes (GNU style)
checking dependency style of gcc... gcc3
checking how to run the C preprocessor... gcc -E
checking for grep that handles long lines and -e... /usr/bin/grep
checking for egrep... /usr/bin/grep -E
```

Step 24: We need to compile all the components of this software based on the instructions in the Makefile.

make
sudo make install

```
[ec2-user@ip-172-31-84-149 nagios-plugins-2.4.11]$ make
make  all-recursive
make[1]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11'
Making all in gl
make[2]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11/gl'
make  all-recursive
make[3]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11/gl'
make[4]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11/gl'
make[4]: Nothing to be done for 'all-am'.
```

```
[ec2-user@ip-172-31-84-149 nagios-plugins-2.4.11]$ sudo make install
Making install in gl
make[1]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11/gl'
make  install-recursive
make[2]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11/gl'
make[3]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11/gl'
make[4]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11/gl'
if test yes = no; then \
  case 'linux-gnu' in \
    darwin[56]*) \
      need_charset_alias=true ;; \
    darwin* | cygwin* | mingw* | pw32* | cegcc*) \
      need_charset_alias=false ;; \
    *) \
      need_charset_alias=true ;; \
  esac ; \
```

```
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-plugins-2.4.11/po'
make[1]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11'
make[2]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11'
make[2]: Nothing to be done for 'install-exec-am'.
make[2]: Nothing to be done for 'install-data-am'.
make[2]: Leaving directory '/home/ec2-user/downloads/nagios-plugins-2.4.11'
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-plugins-2.4.11'
[ec2-user@ip-172-31-84-149 nagios-plugins-2.4.11]$
```

Step 25: We need to register the Nagios service with the system to enable it to manage the server status
sudo chkconfig --add nagios
sudo chkconfig nagios on

```
[ec2-user@ip-172-31-84-149 nagios-plugins-2.4.11]$ sudo chkconfig --add nagios
sudo chkconfig nagios on
error reading information on service nagios: No such file or directory
Note: Forwarding request to 'systemctl enable nagios.service'.
Created symlink /etc/systemd/system/multi-user.target.wants/nagios.service → /usr/lib/systemd/system/nagios.service.
```

Step 26: We need to verify the Nagios configuration for any syntax errors or issues before starting or restarting the Nagios service.

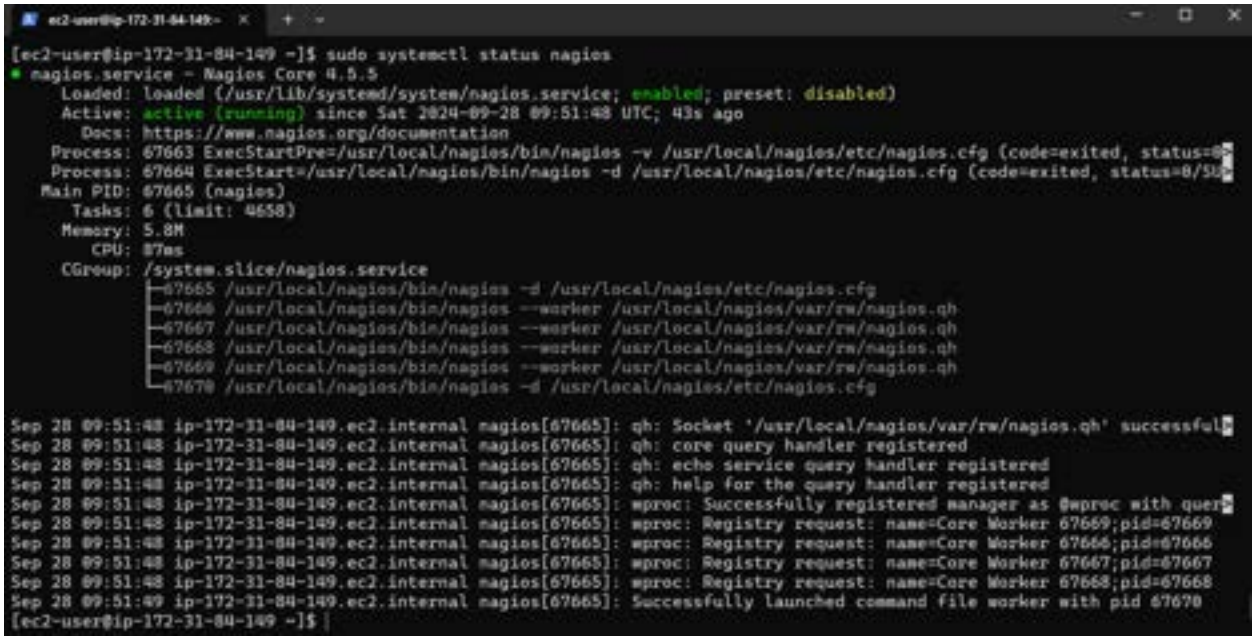
sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

```
[ec2-user@ip-172-31-84-149 nagios-plugins-2.4.11]$ sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

Nagios Core 4.5.5
Copyright (c) 2009-present Nagios Core Development Team and Community Contributors
Copyright (c) 1999-2009 Ethan Galstad
Last Modified: 2024-09-17
License: GPL

Website: https://www.nagios.org
Reading configuration data...
   Read main config file okay...
   Read object config files okay...

Running pre-flight check on configuration data...

Checking objects...
       Checked 8 services.
       Checked 1 hosts.
       Checked 1 host groups.
       Checked 0 service groups.
       Checked 1 contacts.
       Checked 1 contact groups.
       Checked 24 commands.
       Checked 5 time periods.
       Checked 0 host escalations.
       Checked 0 service escalations.
Checking for circular paths...
```

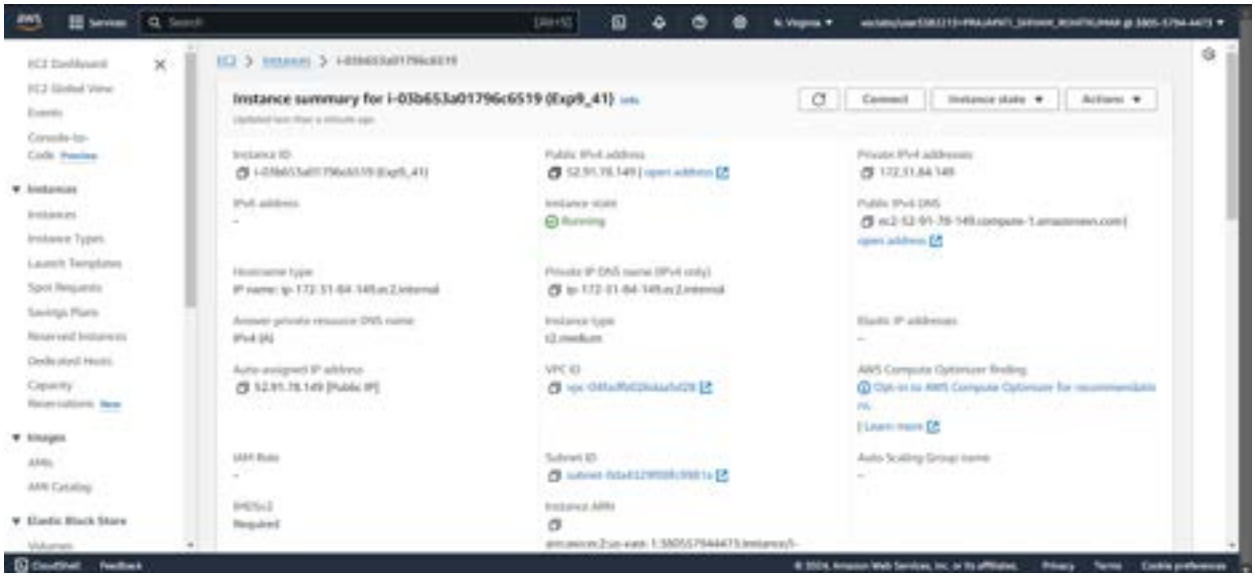
sudo service nagios start

```
[ec2-user@ip-172-31-84-149 nagios-plugins-2.4.11]$ cd
[ec2-user@ip-172-31-84-149 ~]$ sudo service nagios start
Redirecting to /bin/systemctl start nagios.service
[ec2-user@ip-172-31-84-149 ~]$
```

Step 27: Check the status of the nagios.



Step 28: Go back to EC2 Console and copy the Public IP address of this instance. Open up your browser and look for **http://<your_public_ip_address>/nagios**



http://52.91.78.149/nagios.

Enter **username as nagiosadmin** and **password as Kingmaker.**

Step 29: After entering the correct credentials, you will see this page

CONCLUSION:

In this experiment, we have learned how to install and configure Nagios Core, Nagios Plugins, and NRPE on a Linux machine. We used an Amazon Linux OS instance with the necessary security rules in place. It's important to ensure that the links for Nagios Core and Nagios Plugins are up to date (when using wget). After extracting and configuring these files, we should check for any issues before starting the server. Once everything is set up, we can start the Nagios server. By using the public IP address of the EC2 instance, we can access the Nagios dashboard by navigating to that IP followed by /nagios.

Name: Shivam.R.Prajapati Div: D15C Roll No:41 Academic Year: 2024-25

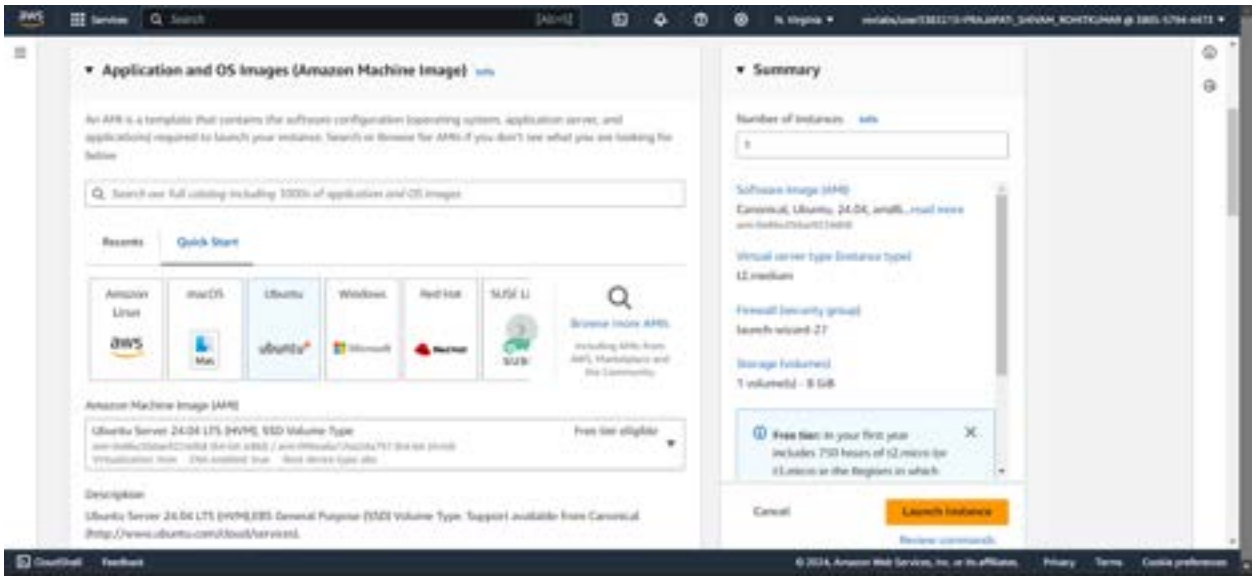
Experiment No:10

AIM: To perform Port, Service monitoring, Windows/Linux server monitoring using Nagios.

PREREQUISITES : We should have an Amazon Linux instance with nagios already set up.

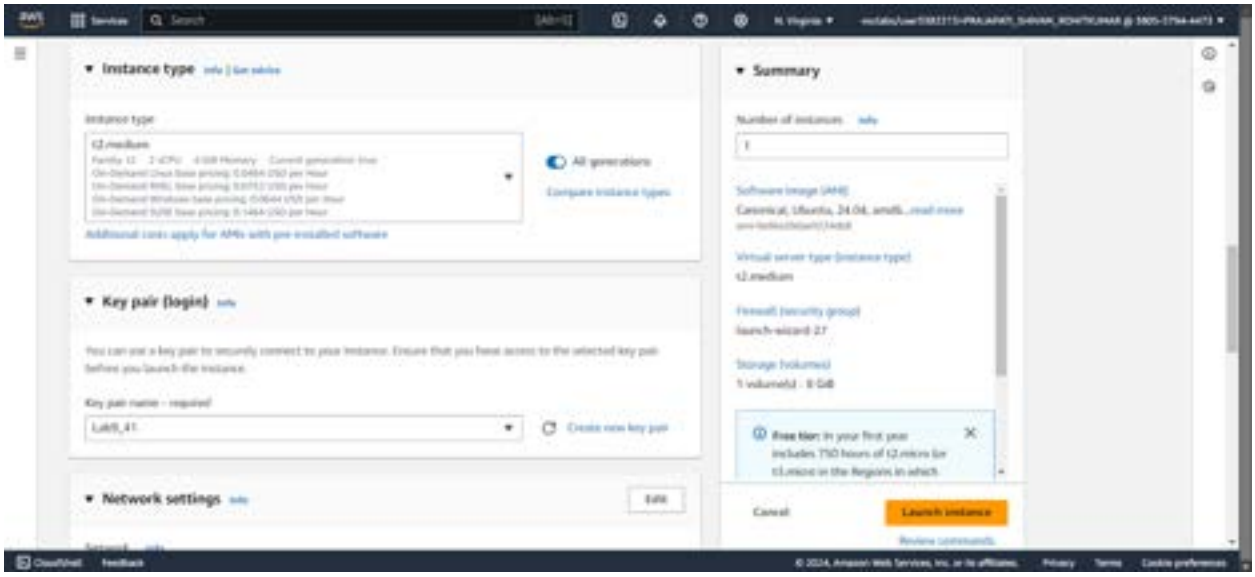
Step 1: Set up ubuntu instance

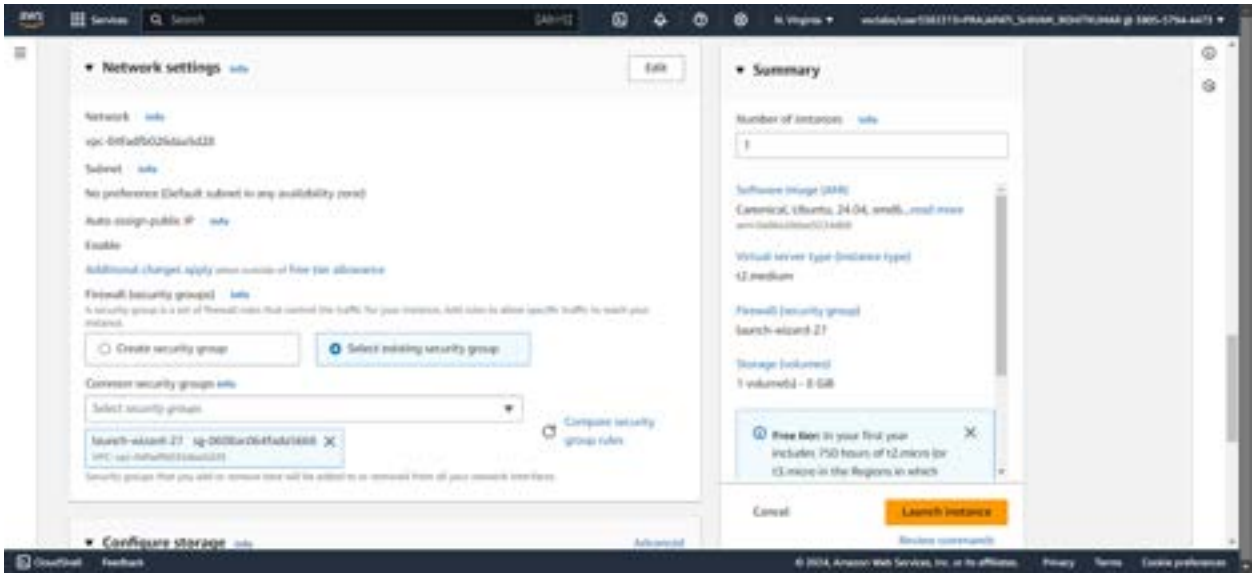
1) Log in to your AWS account. Look for EC2 in the services menu. Open the interface and select Create Instance.



Select The OS Image as Ubuntu.

2) Ensure that you choose the same private key you created for the Amazon Linux instance. Additionally, select the same security group that you configured for the Linux instance.

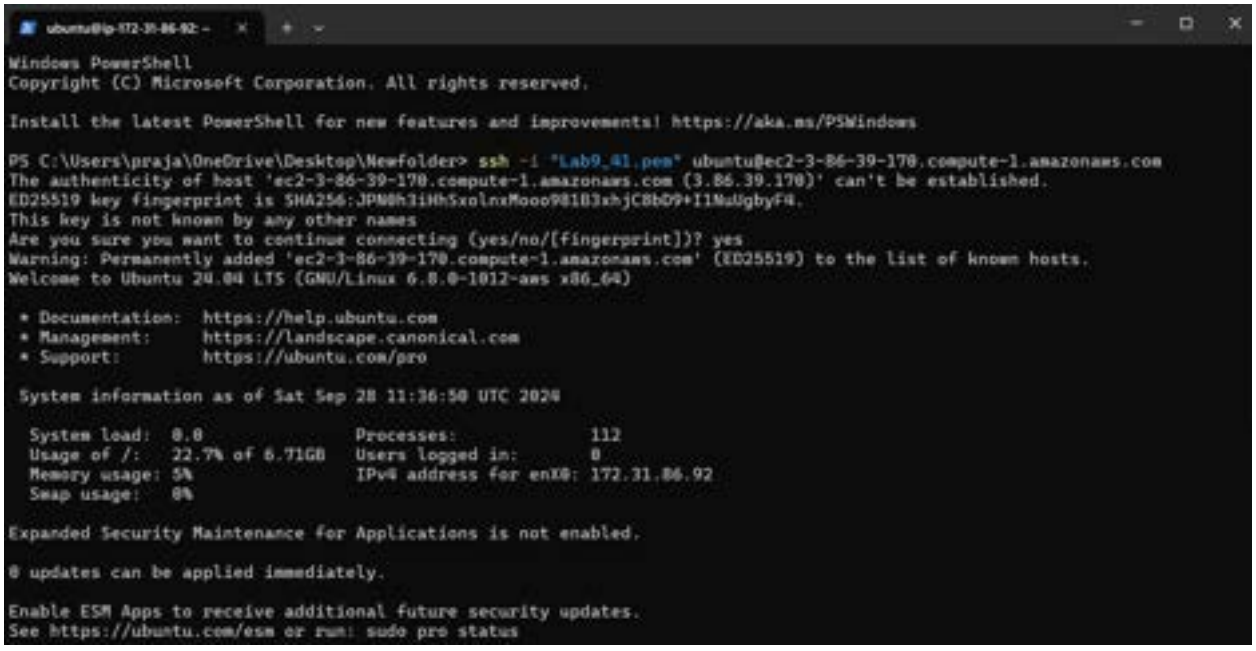




Instance is:

3) Now return to the instances screen. Click on the instance ID of your instance, then select Connect. Click on SSH client and copy the example command. Next, we need to connect our local OS terminal to the instance using SSH. To do this, open the terminal where the private key file (.pem) is stored. Paste the copied SSH command and execute it.

Copy the example command



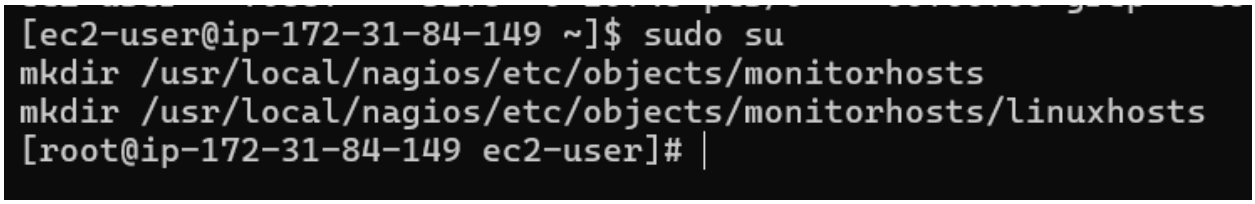
*Successfully connected the instance via **SSH***

Step 2: On Nagios Host machine (Linux) execute the following which we have already created as a prerequisites:

1) We need to verify whether the nagios service is running or not. Fo that, run this command : **ps -ef | grep nagios**

2) Next, switch to the root user and create a directory at the path '/usr/local/nagios/etc/objects/monitorhosts/linuxhosts'.

sudo su
mkdir -p /usr/local/nagios/etc/objects/monitorhosts/linuxhosts



3) We need to create a configuration file in this directory. To do this, copy the contents of the existing localhost configuration into the new file named **'linuxserver.cfg'.**

cp /usr/local/nagios/etc/objects/localhost.cfg
/usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg

So make the second directory again and run the cp command

We need to make some changes in this config file. Open it using a nano editor.

nano /usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg

```
[root@ip-172-31-84-149 ec2-user]# nano /usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg
```

Change **hostname** and **alias** to **linuxserver .**Change address to **public ip address of client instance** (Ubuntu instance)

```
# Define a host for the local machine

define host {

    use                     linux-server


    host_name               linuxserver
    alias                   linuxserver
    address                 3.86.39.170
}
```

Change hostgroup_name to **linux-servers1**

```
# Define an optional hostgroup for Linux machines

define hostgroup {

    hostgroup_name          linux-servers1          ; The name of the hostgroup
    alias                   Linux Servers           ; Long name of the group
    members                 localhost               ; Comma separated list of hosts that belong to this group
}


##############################################################################
#
# SERVICE DEFINITIONS
#
##############################################################################
```

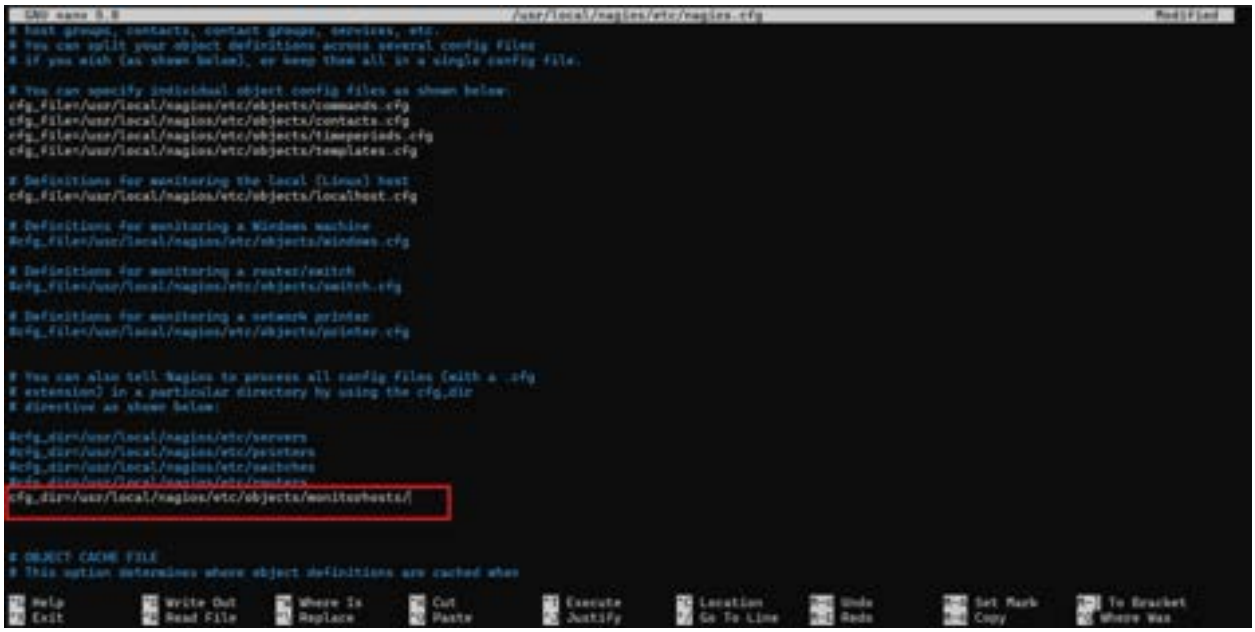
Change the **occurrences of hostname** further in the document from **localhost** to **linuxserver**

Now, we need to edit the nagios configuration file to add this directory. Run this command

nano /usr/local/nagios/etc/nagios.cfg

```
[root@ip-172-31-84-149 ec2-user]# nano /usr/local/nagios/etc/nagios.cfg
```

and add the following line **cfg_dir=/usr/local/nagios/etc/objects/monitorhosts/**



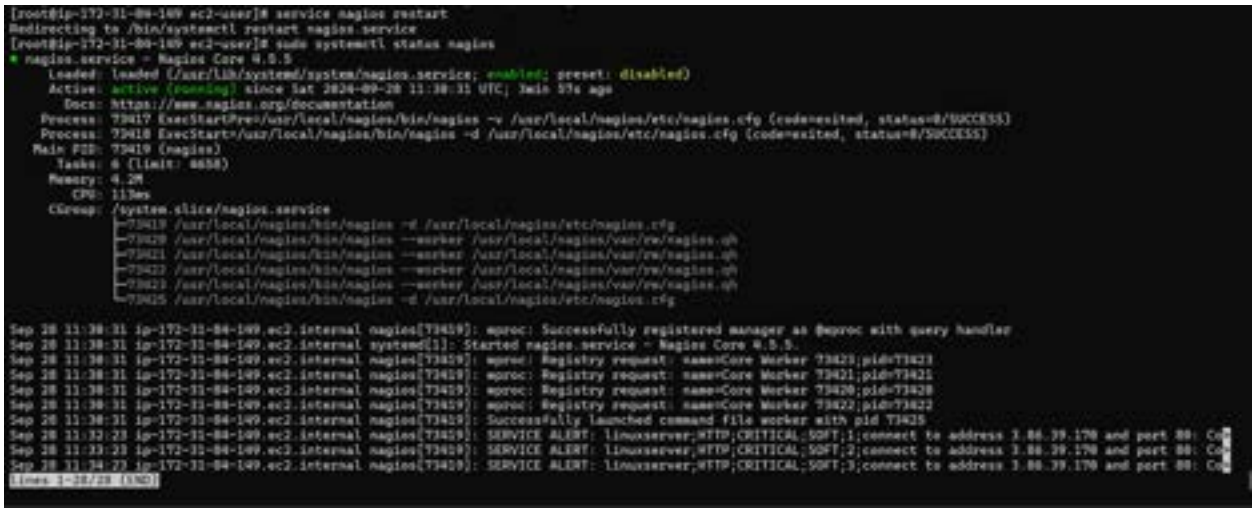
Now we verify the configuration files. **/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg**

```
[root@ip-172-31-84-149 ec2-user]# /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

Nagios Core 4.5.5
Copyright (c) 2009-present Nagios Core Development Team and Community Contributors
Copyright (c) 1999-2009 Ethan Galstad
Last Modified: 2024-09-17
License: GPL

Website: https://www.nagios.org
Reading configuration data...
   Read main config file okay...
   Read object config files okay...

Running pre-flight check on configuration data...

Checking objects...
        Checked 16 services.
        Checked 2 hosts.
        Checked 2 host groups.
        Checked 0 service groups.
        Checked 1 contacts.
        Checked 1 contact groups.
```

```
Checking objects...
        Checked 16 services.
        Checked 2 hosts.
        Checked 2 host groups.
        Checked 0 service groups.
        Checked 1 contacts.
        Checked 1 contact groups.
        Checked 24 commands.
        Checked 5 time periods.
        Checked 0 host escalations.
        Checked 0 service escalations.
Checking for circular paths...
        Checked 2 hosts
        Checked 0 service dependencies
        Checked 0 host dependencies
        Checked 5 timeperiods
Checking global event handlers...
Checking obsessive compulsive processor commands...
Checking misc settings...

Total Warnings: 0
Total Errors:   0

Things look okay - No serious problems were detected during the pre-flight check
[root@ip-172-31-84-149 ec2-user]#
```

Once the files are verified, we need to restart the server: **service nagios restart**

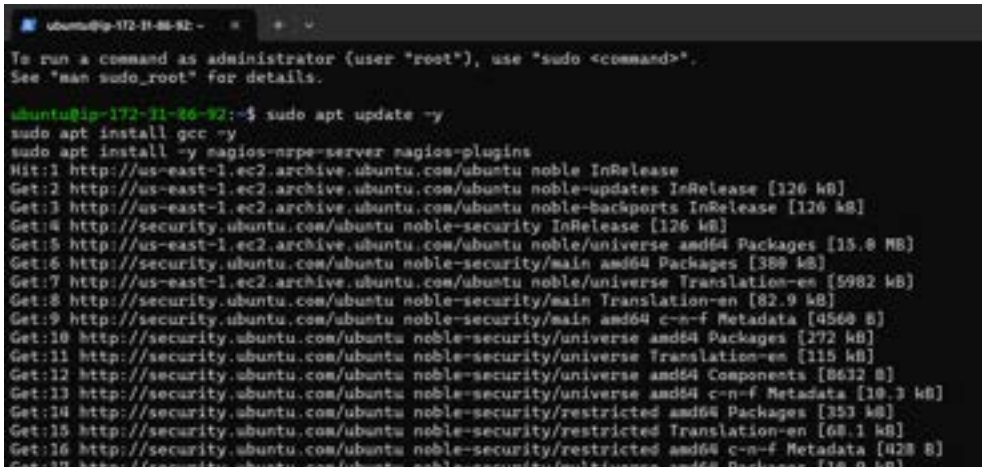
```
[root@ip-172-31-84-149 ec2-user]# service nagios restart
Redirecting to /bin/systemctl restart nagios.service
[root@ip-172-31-84-149 ec2-user]#
```

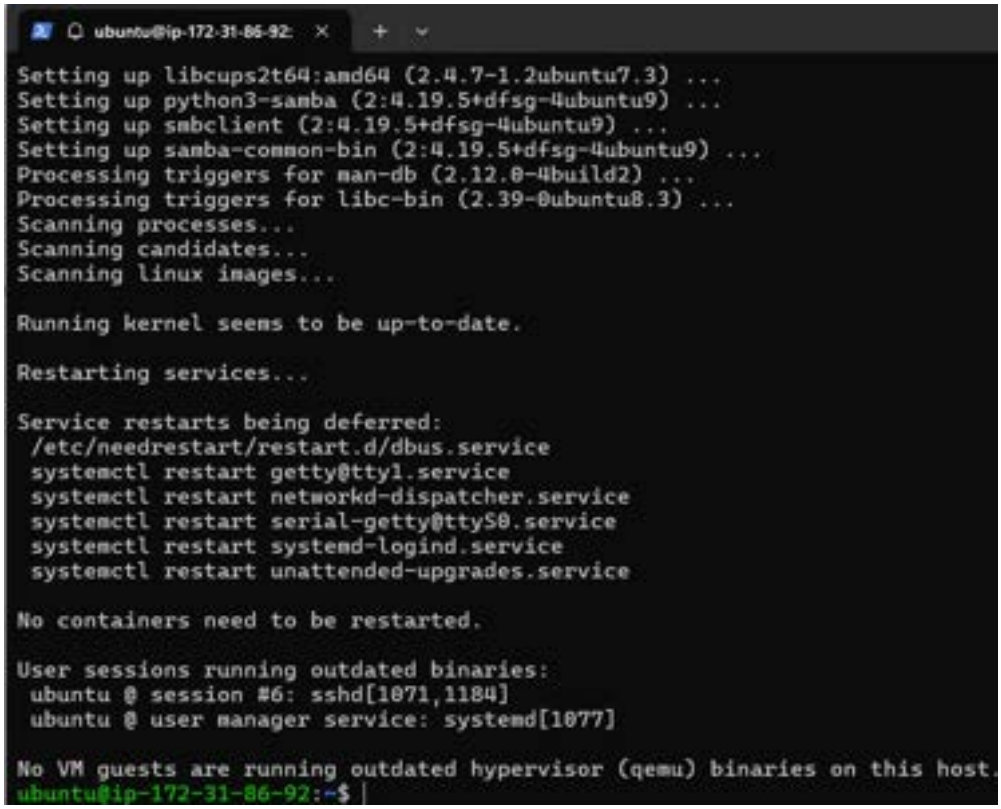


Step 3: Execute the following on Nagios Client machine (Ubuntu)

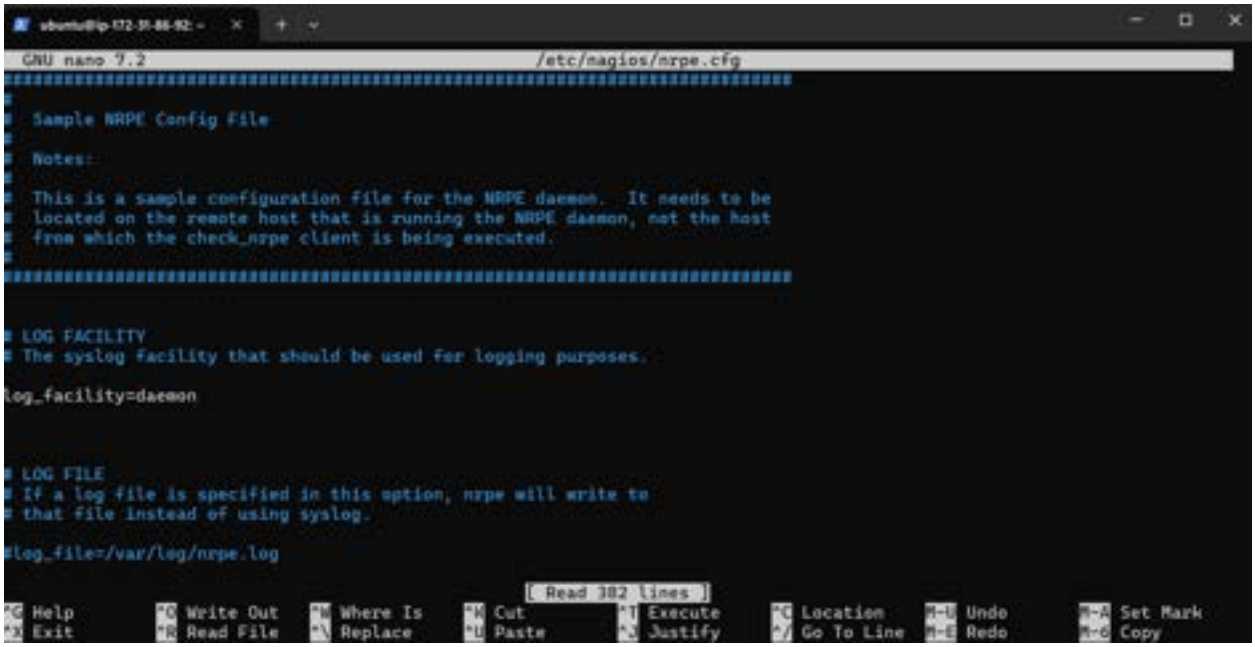
1) First, check for any available updates, and then proceed to install gcc, the Nagios NRPE server, and Nagios plugins.

   sudo apt update -y
   sudo apt install gcc -y
   sudo apt install -y nagios-nrpe-server nagios-plugins

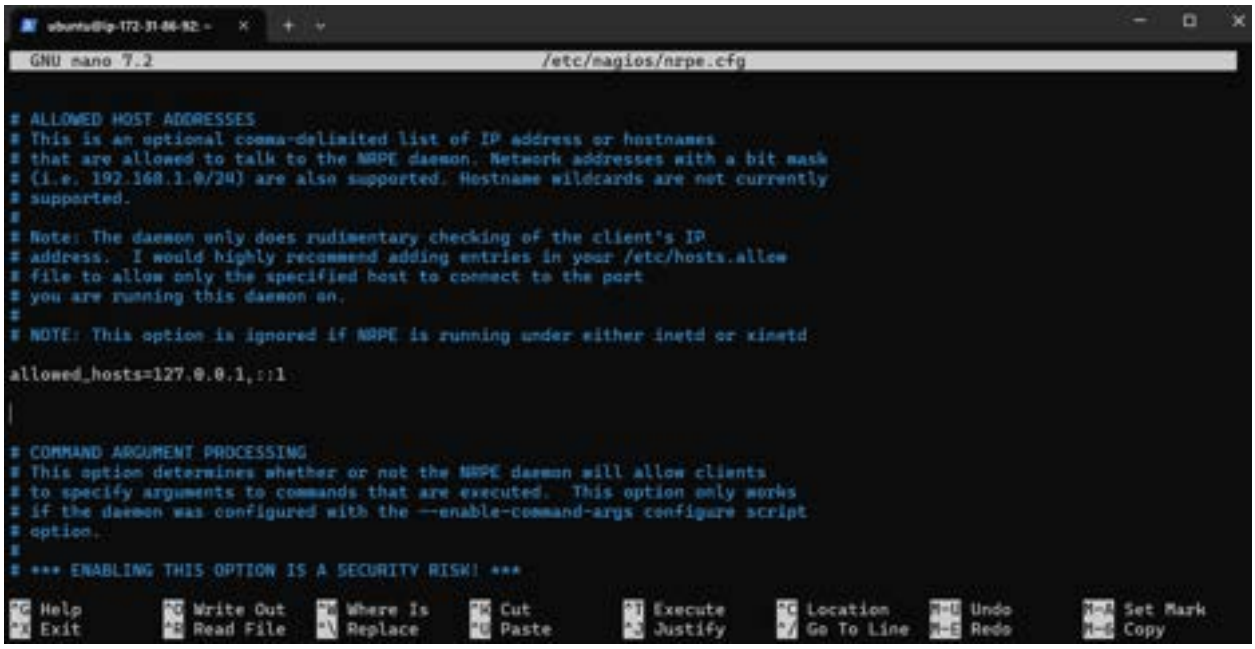




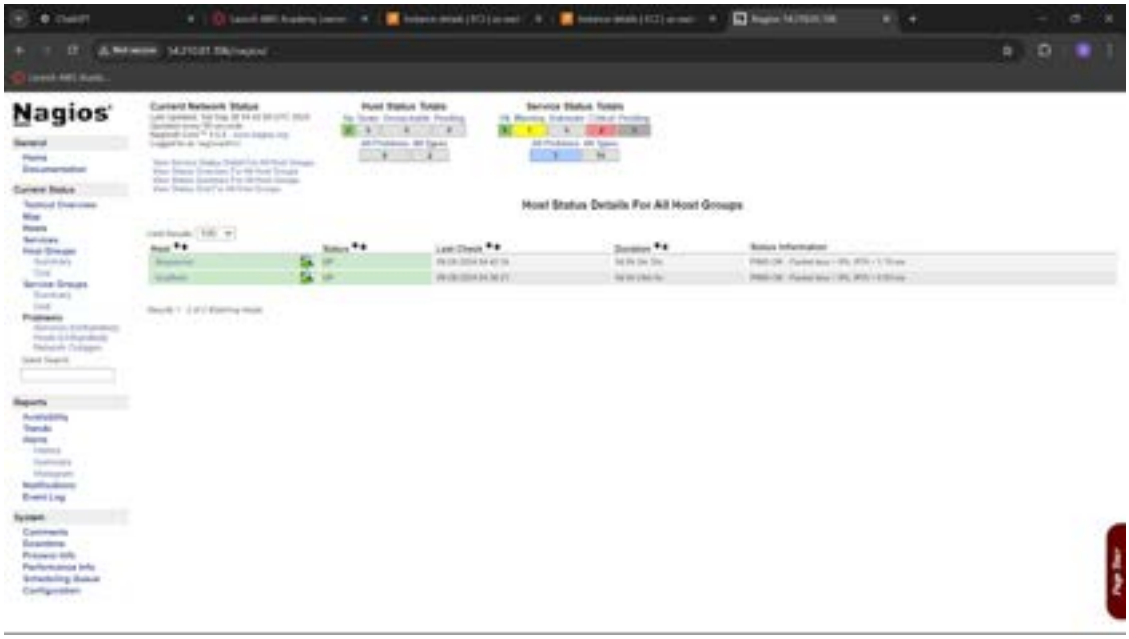
2) We need to include the public IP address of our Nagios host machine (Linux) in the NRPE configuration file. **sudo nano /etc/nagios/nrpe.cfg**



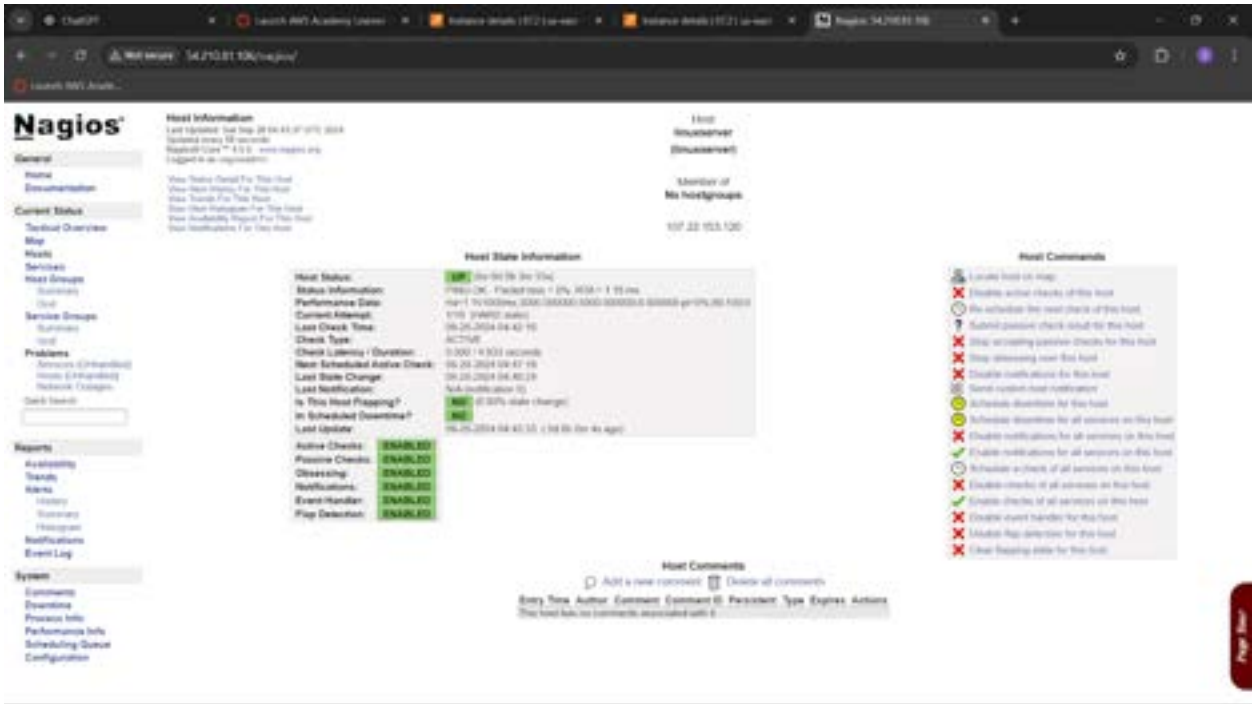
Under allowed_hosts, add the nagios host ip address (public)



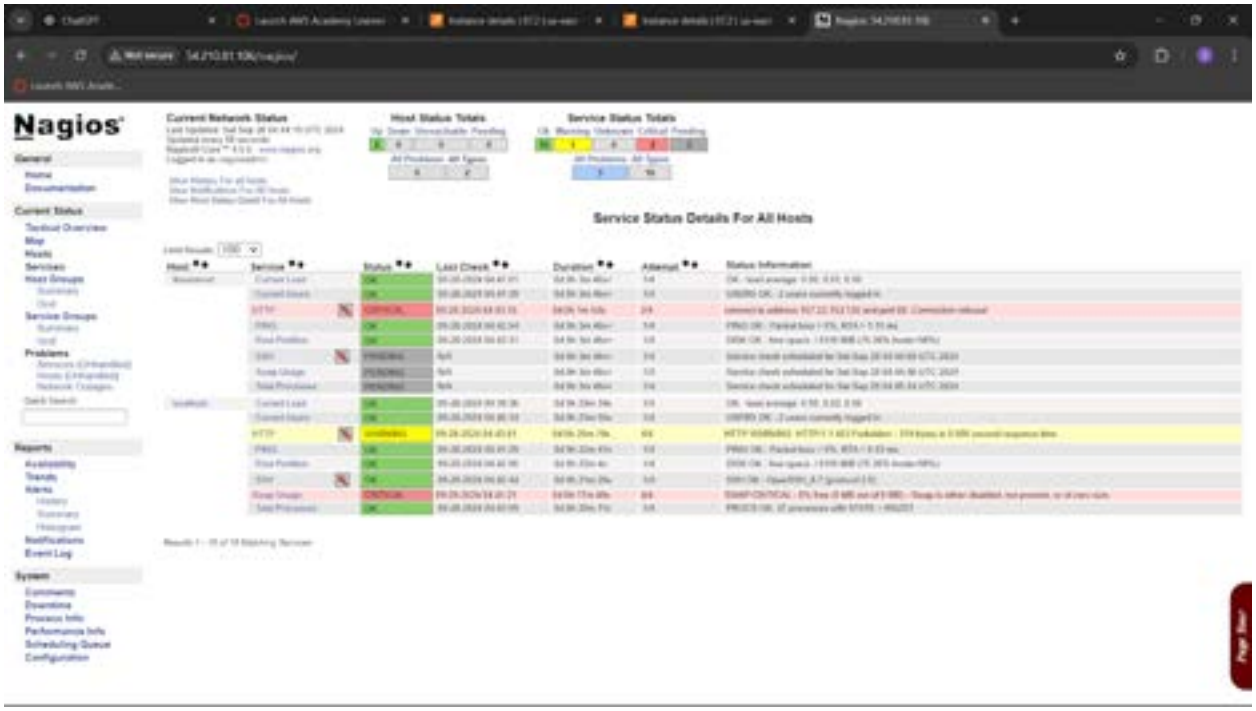
Step 4: Check the Nagios Dashboard.Go to Nagios dashboard, click on hosts.Here, we can see that the linuxserver is also added as a host.



Click on linuxserver. we can check all the information about linuxserver host.



Click on services. Here we can see all the services that are being monitored by linuxserver.



CONCLUSION:

In this experiment, we learned to conduct port service monitoring and server monitoring using Nagios. To do this, we require a Linux instance to host the Nagios dashboard and a separate Ubuntu instance linked as a second host. We need to configure the Linux instance and include the IP address of the Ubuntu instance. Subsequently, we must replicate the initial setup from the Linux instance on the Ubuntu instance by adding the IP address of the Linux instance to the list of allowed hosts. After restarting the NRPE server, we should see the ‘linux server’ host listed.

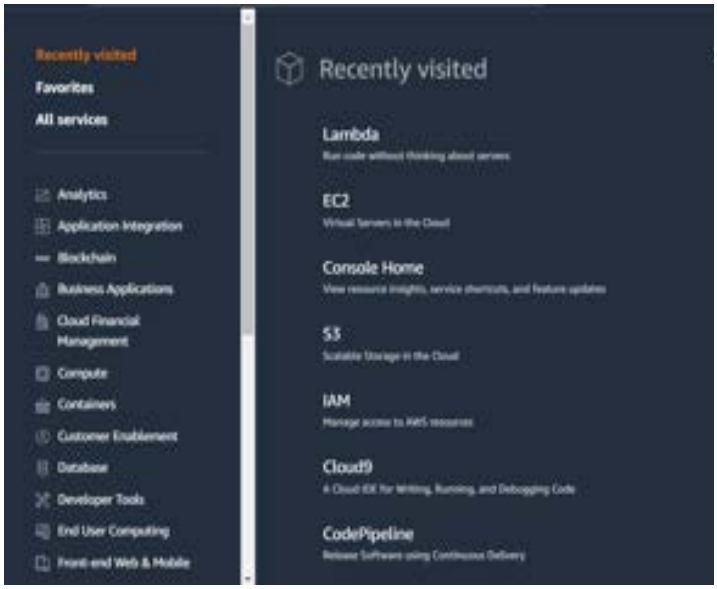
Name: Shivam.R.Prajapati Div: D15C Roll No:41 Academic Year: 2024-25

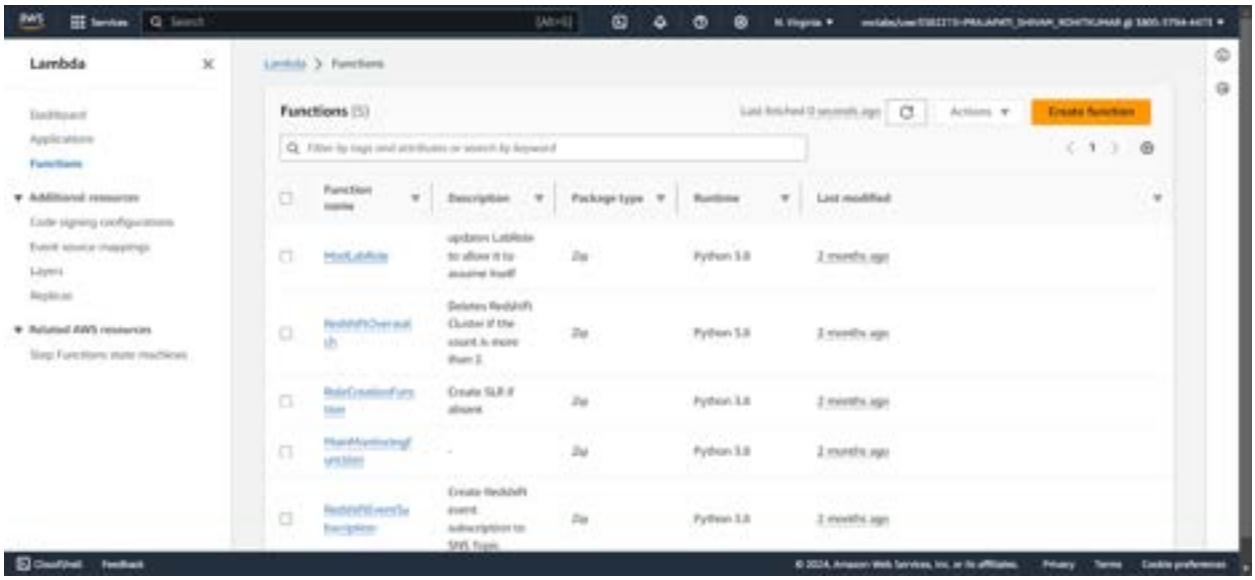
Experiment No: 11

AIM: To understand AWS Lambda, its workflow, various functions and create your first Lambda functions using Python / Java / Nodejs.

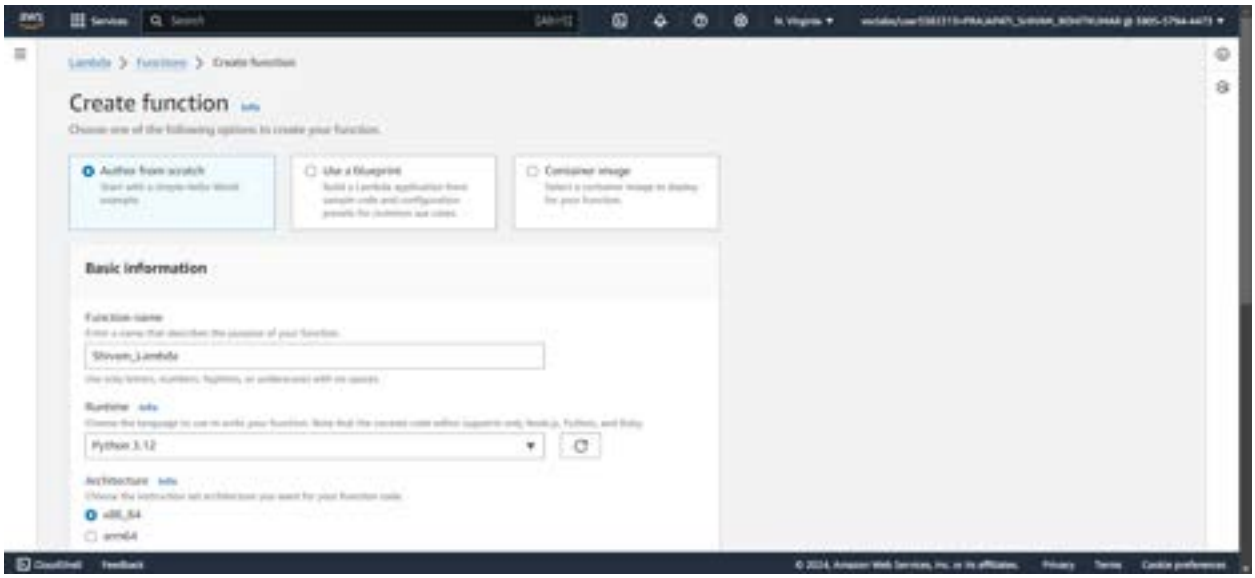
CREATION OF LAMBDA FUNCTION:

Step1: Log in to your AWS Personal or Academy account. Navigate to Lambda, then select the 'Create Function' button



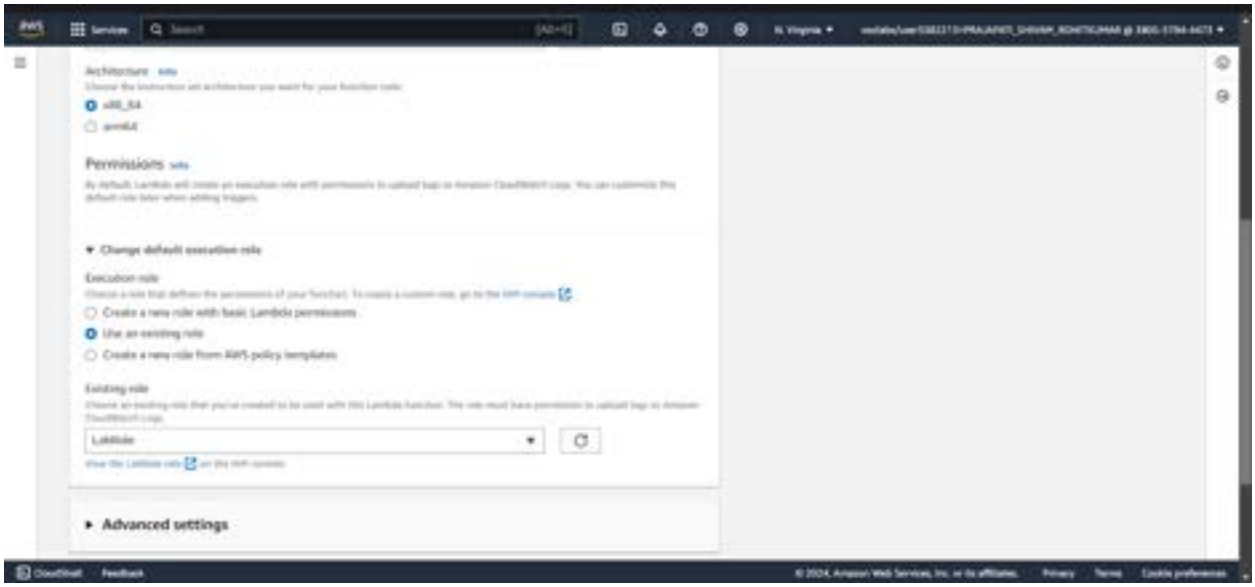


Step 2: Give your Lambda function a name and choose a programming language. The code editor only supports Node.js, Python, and Ruby, so in my case I have chosen **Python 3.12**. Set the **architecture to x86**. For the execution role, select '**Use an existing role**,' then pick '**Lab role**' from the dropdown menu under existing roles .
(This is because the Lab role already has the permissions needed for Lambda to run properly, so you don't need to create a new role from scratch. It's a quicker and more convenient option)

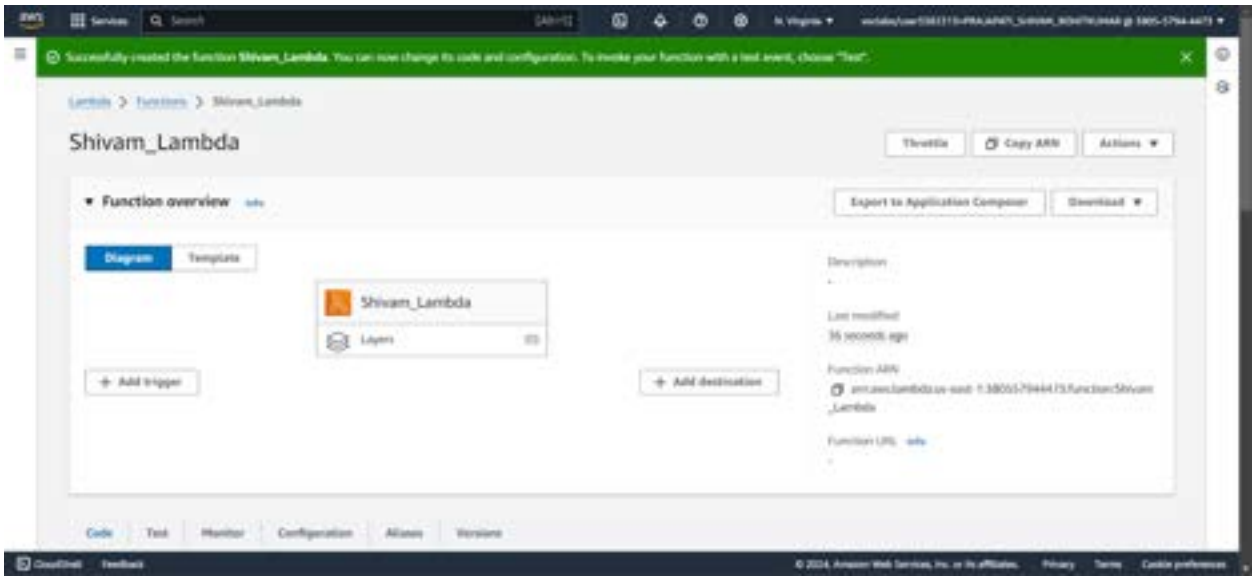


Give the function name and select required language for lambda function

Architecture will be x86_64

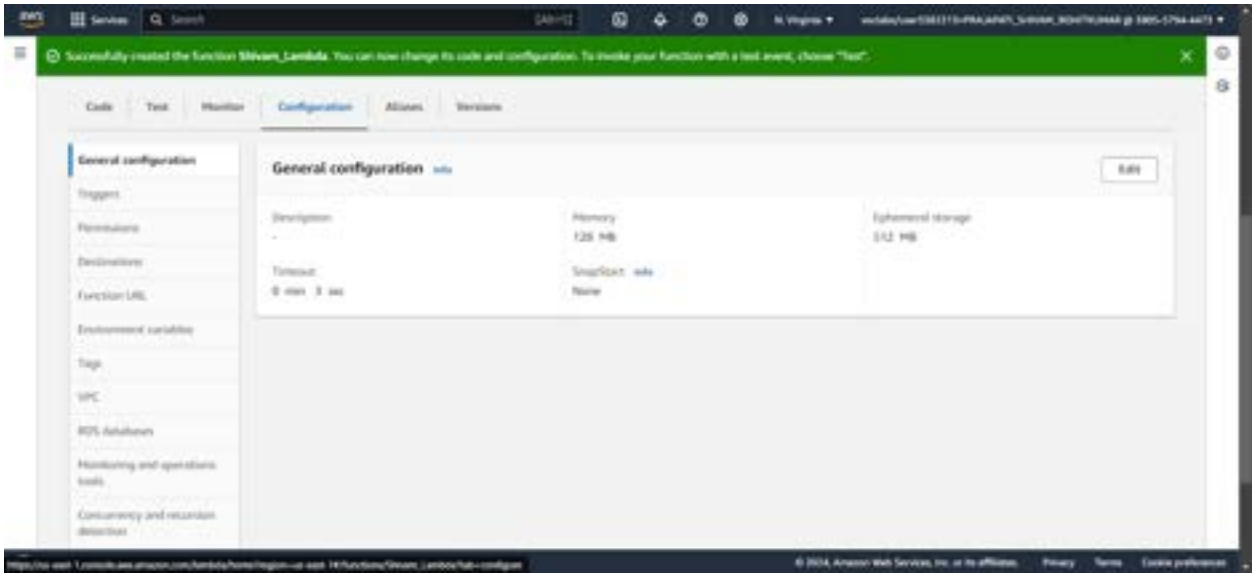


Select proper Execution role

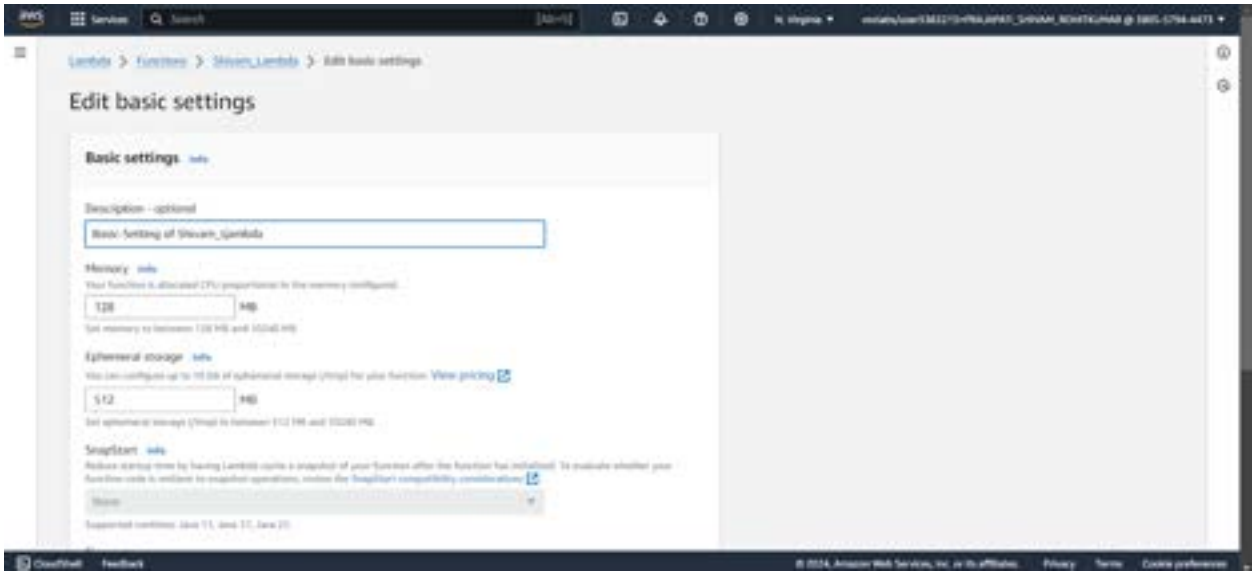


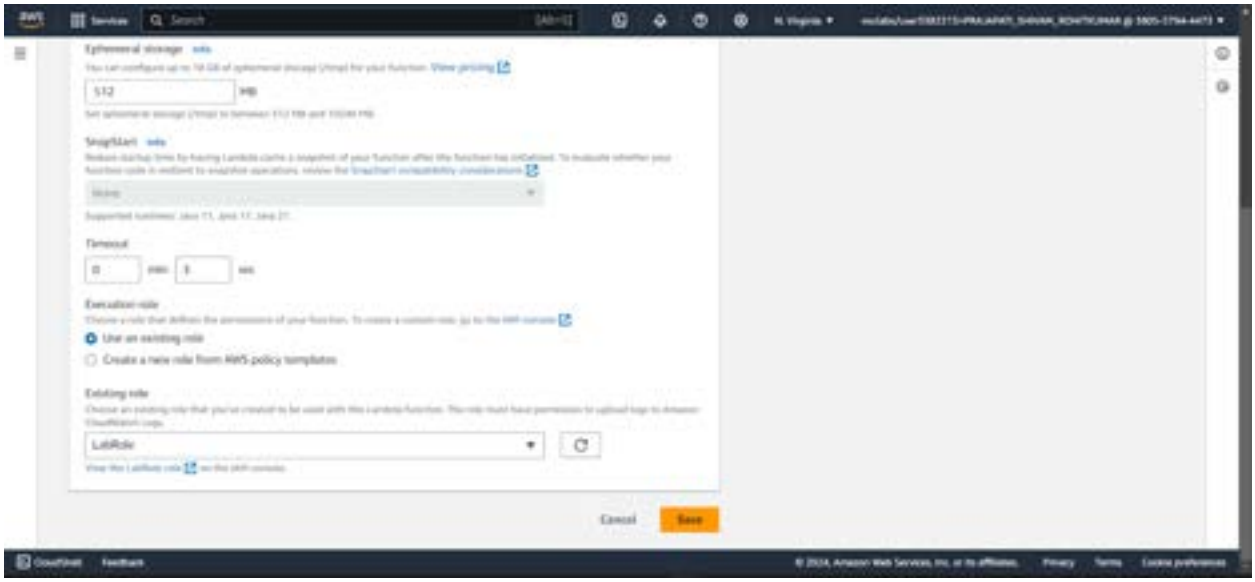
Successfully created Lambda function

Step 3: To view or change the basic settings, go to the 'Configuration' tab and click 'Edit' under 'General settings.' (THIS STEP IS OPTIONAL)



You can add a description and adjust the memory and timeout settings. I've changed the timeout to 1 second, as that's enough for now.

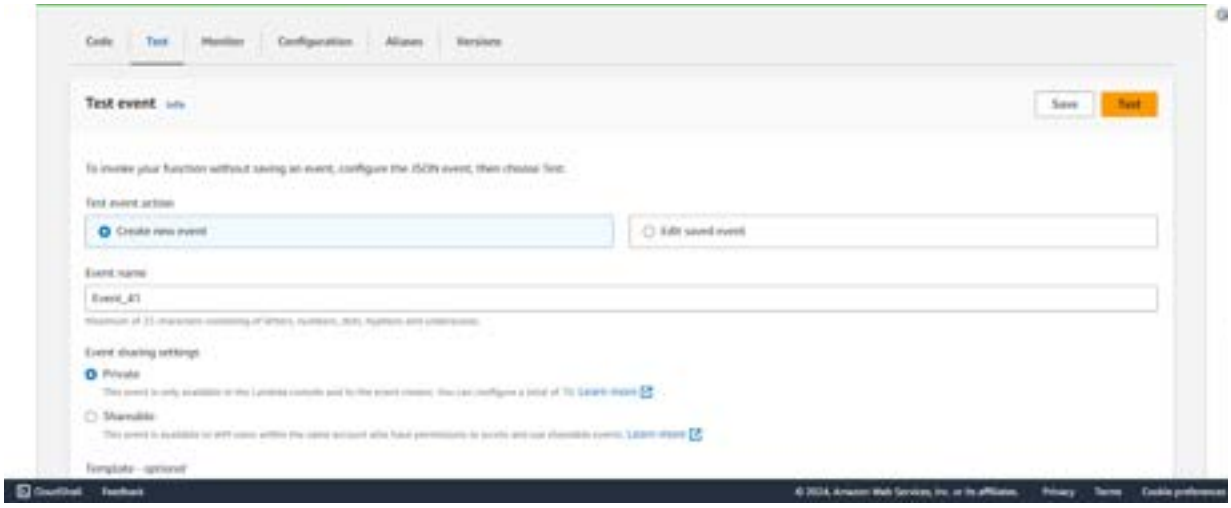


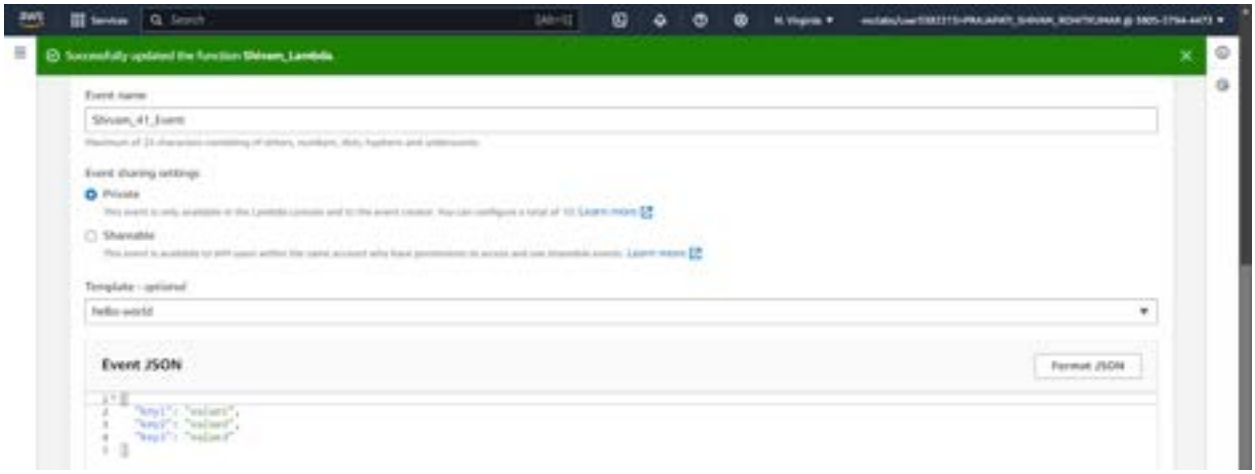


Click on Save

Step 4: Go to the 'Test' tab and click 'Create a new event.' Give the event a name, set 'Event Sharing' to private, and choose the 'hello-world' template.

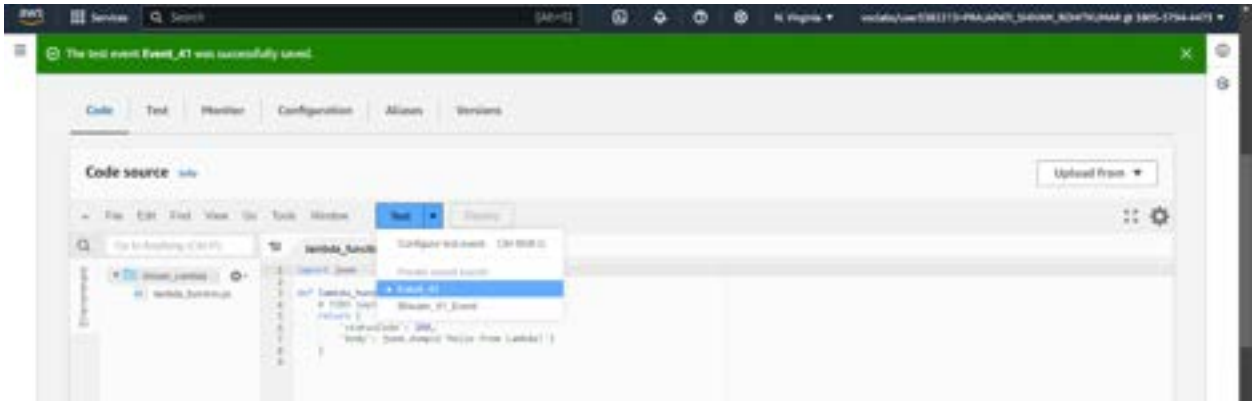
We basically create a new event to test and verify your Lambda function; setting Event Sharing to private keeps it secure and choosing the "hello-world" template provides a simple structure for testing without complex inputs.



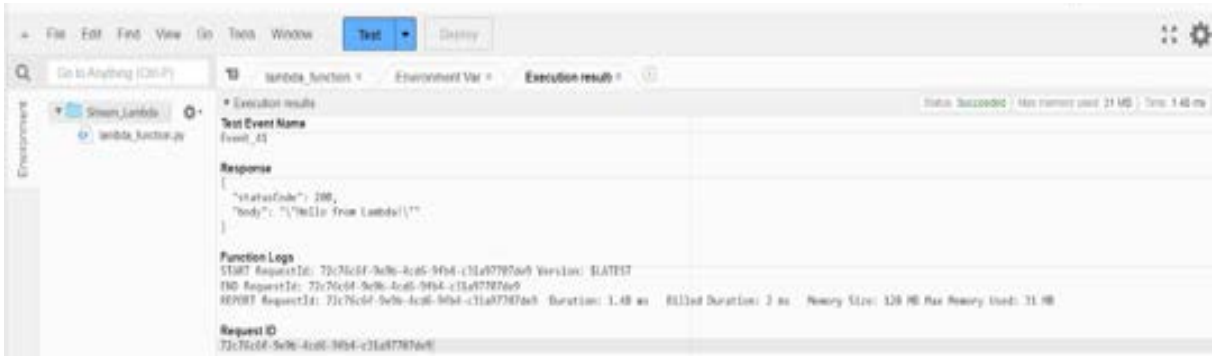


Click on save button above

Step 5: In the Code section, select the event you created from the dropdown menu under 'Test,' then click 'Test.' You should see the output below."

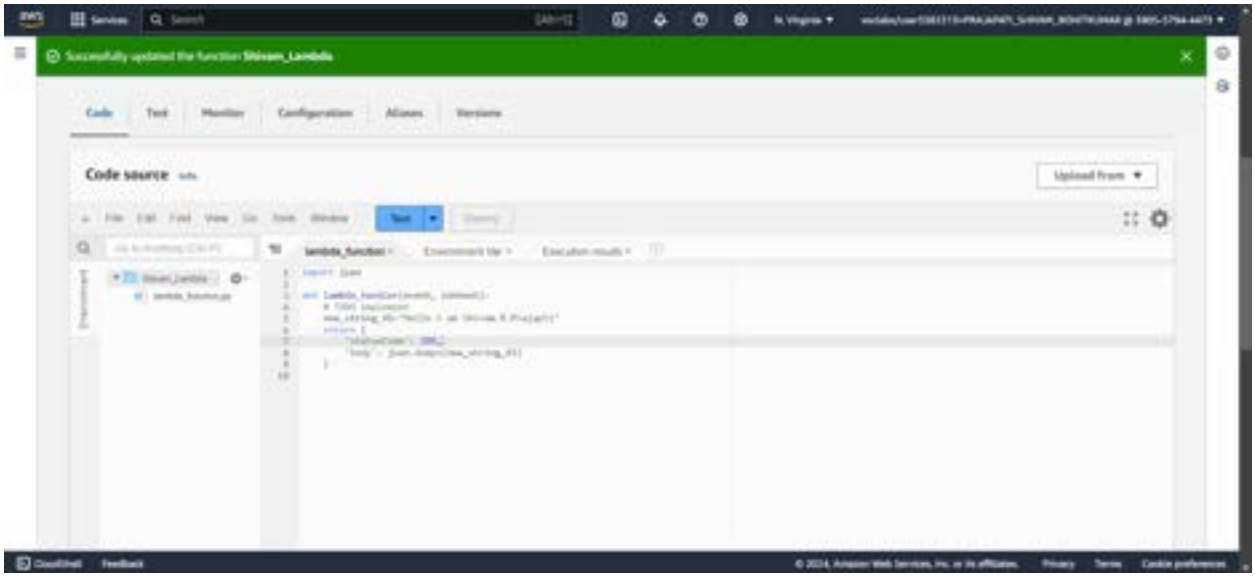


Output :



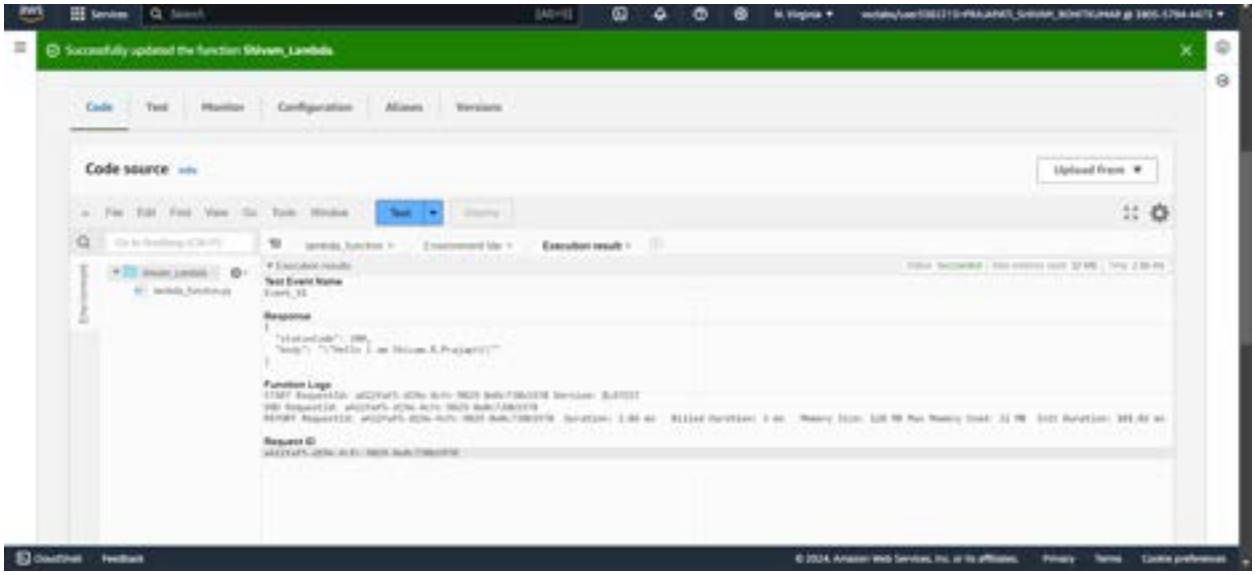
You select the created event to run the specific test you set up, and clicking 'Test' executes your Lambda function to check if it works as expected and produces the desired output.

Step 6: You can edit your lambda function code. I have changed the code to display the new String. After Changing save it by Control + S and click on Deploy . Make sure you have internet connectivity while deploying or else it will show failed deployment



Successfully changed the function.

Step 7: Click on '**Test**' to see the output. You'll get a status code of **200** which means "OK" and indicates that the request was successful, your string output, and the function logs, showing that it was deployed successfully.



CONCLUSION:

In this experiment, we successfully created an AWS Lambda function and followed the important steps involved. First, we set up the function using Python and adjusted the timeout setting to 1 second. Then, we created a test event to see how the function works and checked the output to ensure it was correct. We also modified the function's code and redeployed it to see the changes in real-time. So Lambda Function allows you to concentrate on writing code while AWS manages the infrastructure and automatically scales the service as needed. This makes it easier to develop and run applications without worrying about server management.

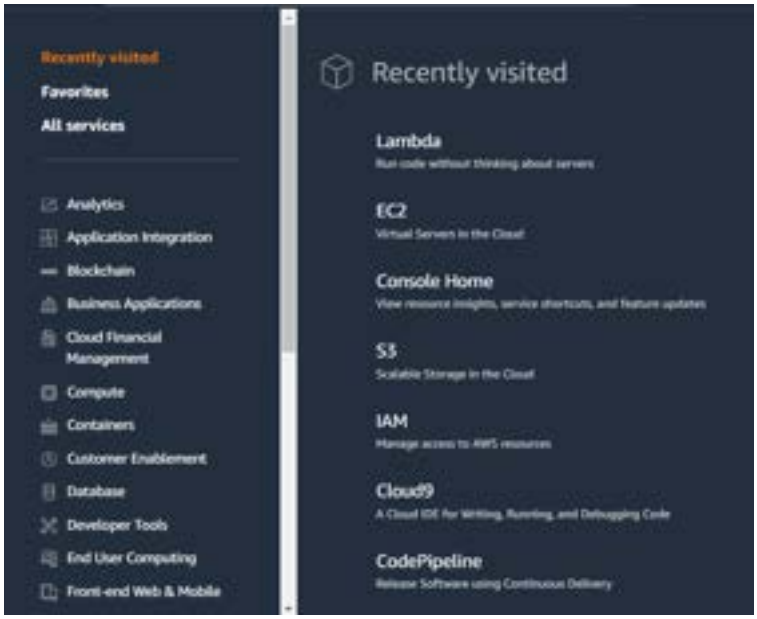
Name: Shivam.R.Prajapati Div: D15C Roll No:41 Academic Year: 2024-25

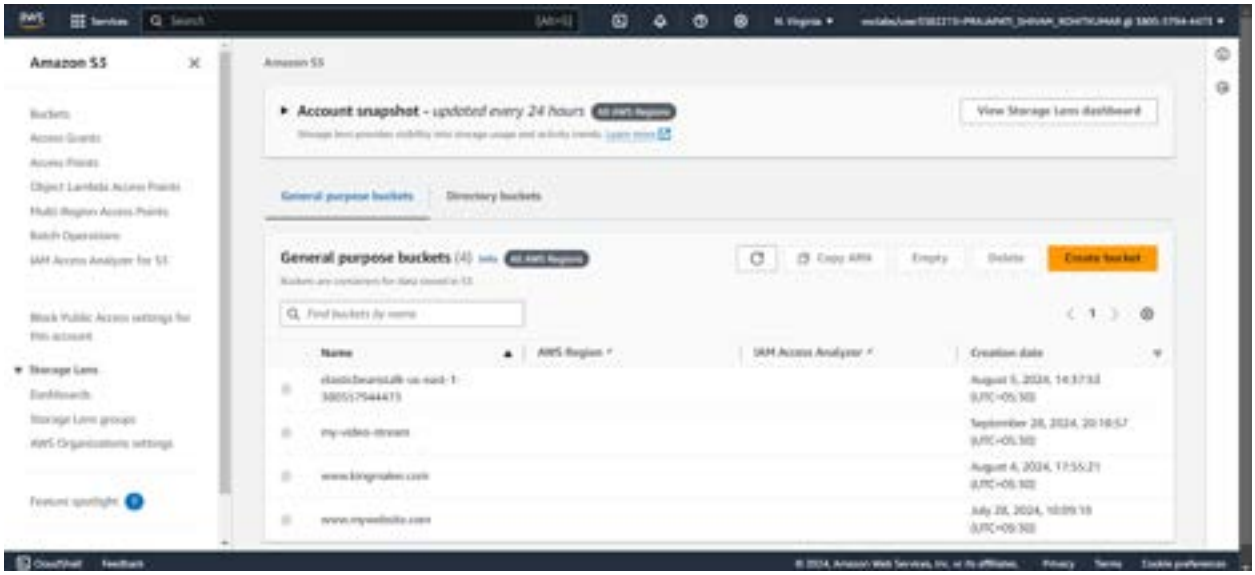
Experiment No :12

AIM : To create a Lambda function which will log “An Image has been added” once you add an object to a specific bucket in S3

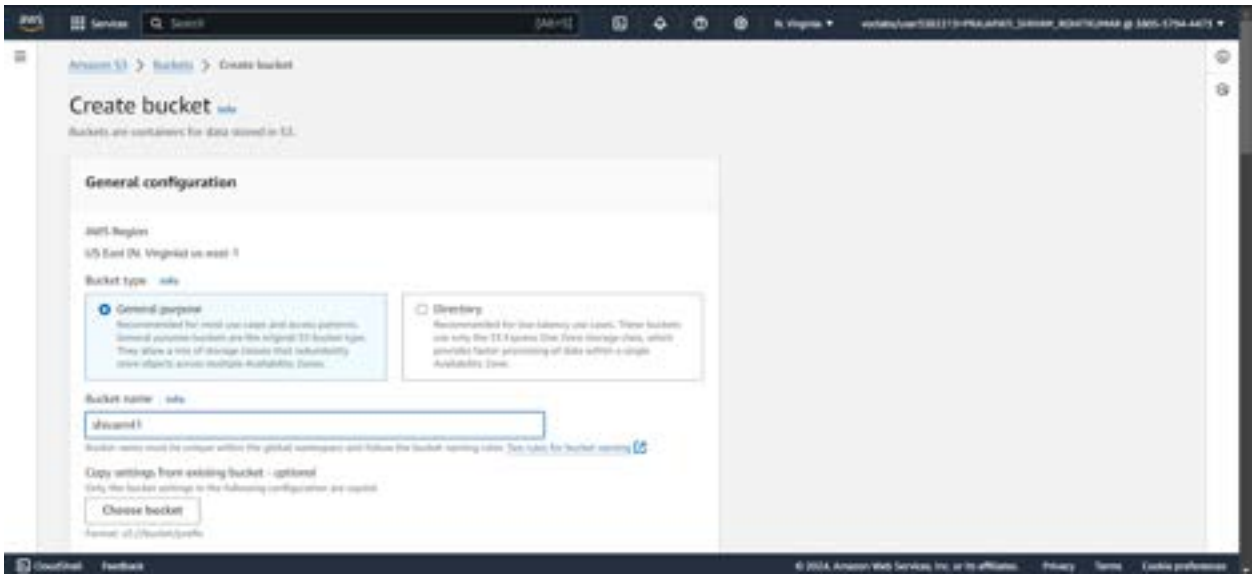
CREATING LAMBDA FUNCTION :

Step 1: Log in to your AWS Personal account. Then go to S3 in the services menu and click on "Create S3 Bucket."



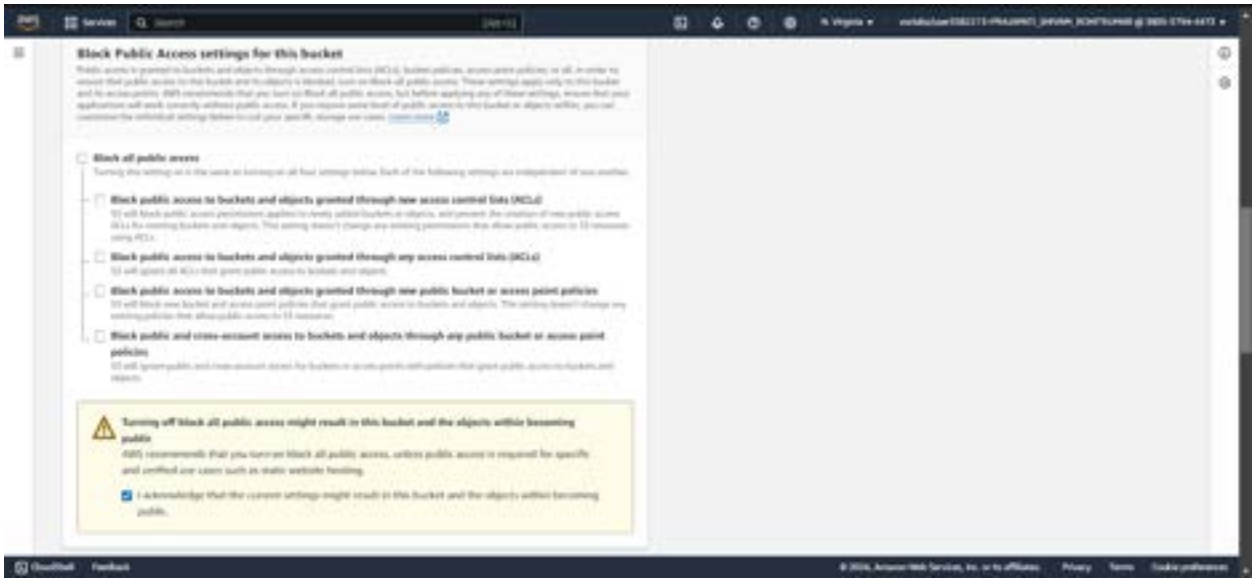


Step 2: Give your bucket a name, select "General purpose project," then uncheck "Block public access." Keep the other settings as they are.

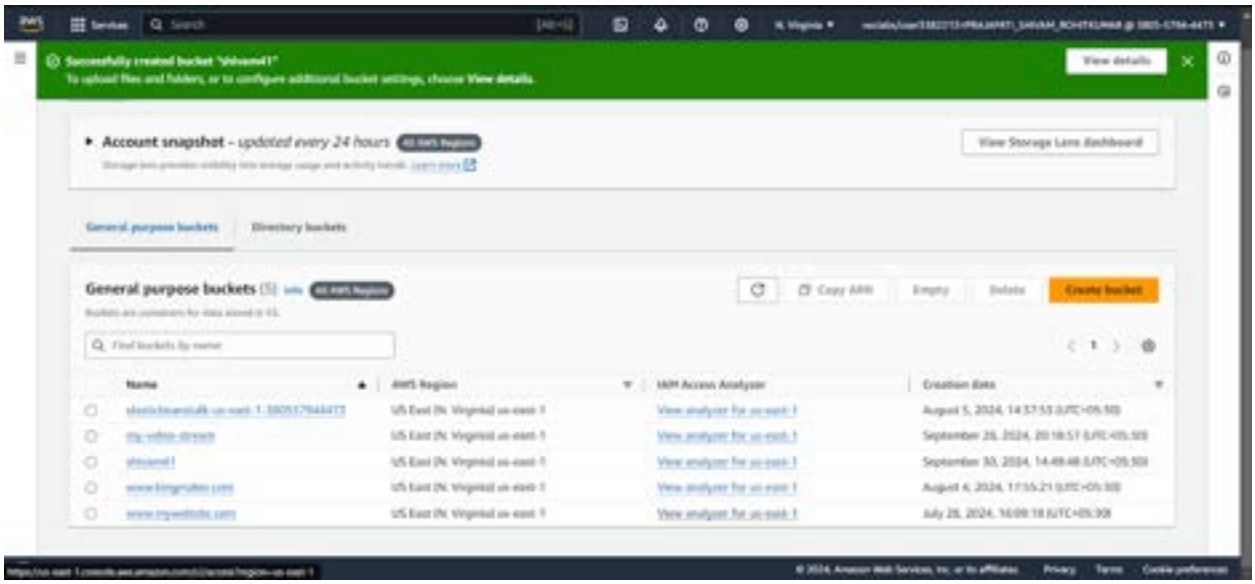


Given proper bucket Name

Selected Appropriate object Ownership

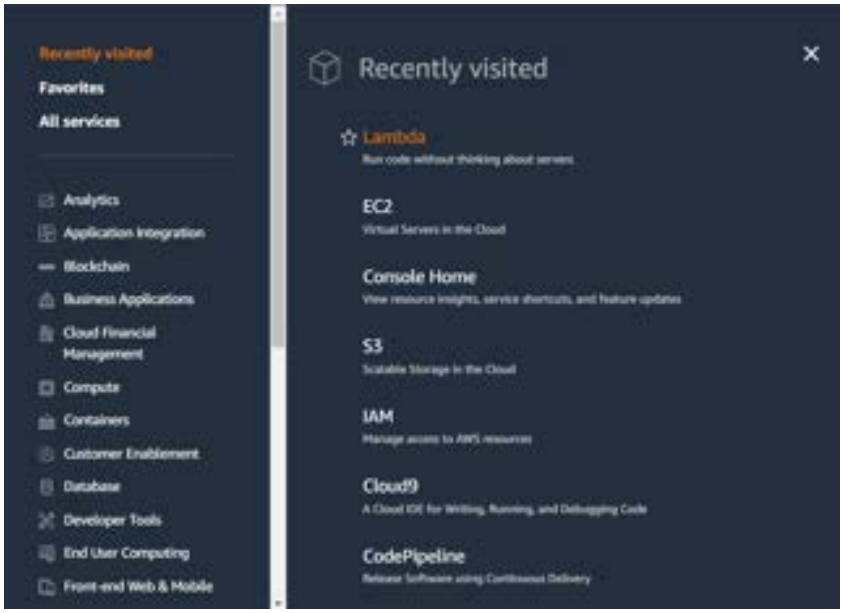


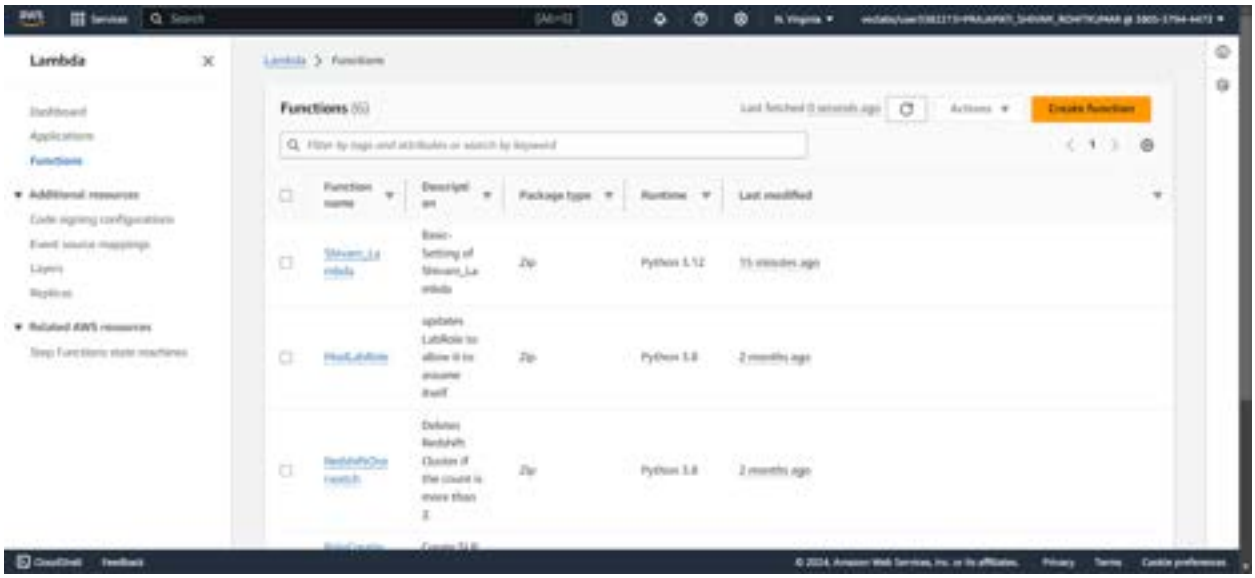
Unchecked the block all public access checkbox and checked the lower checkbox.



Successfully created the bucket

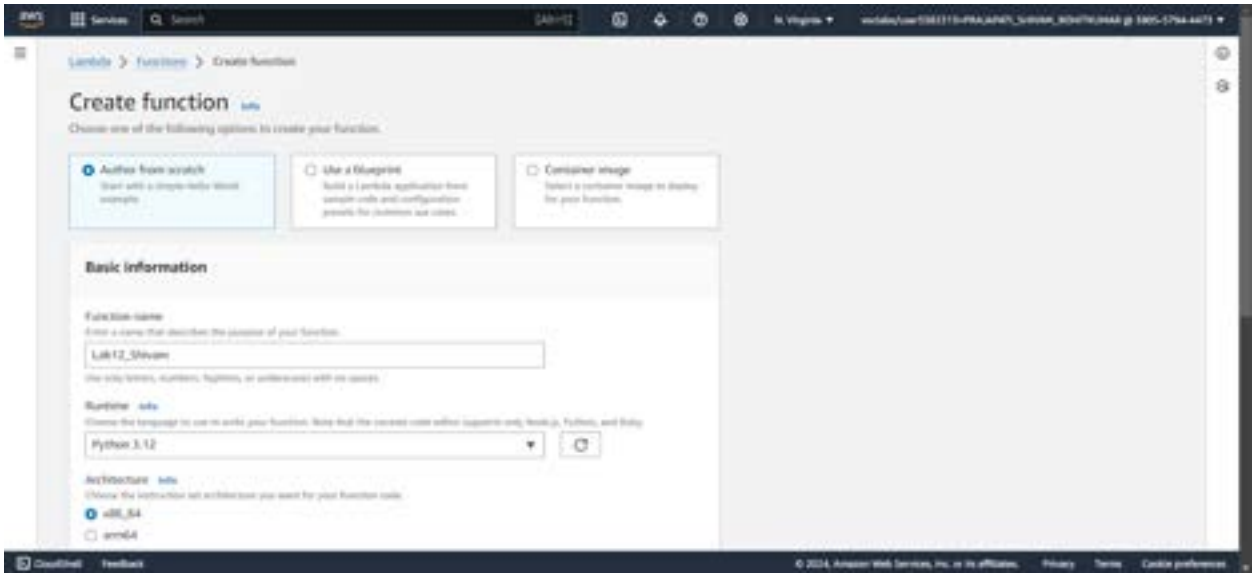
Step 3: Open lambda console and click on create function button.



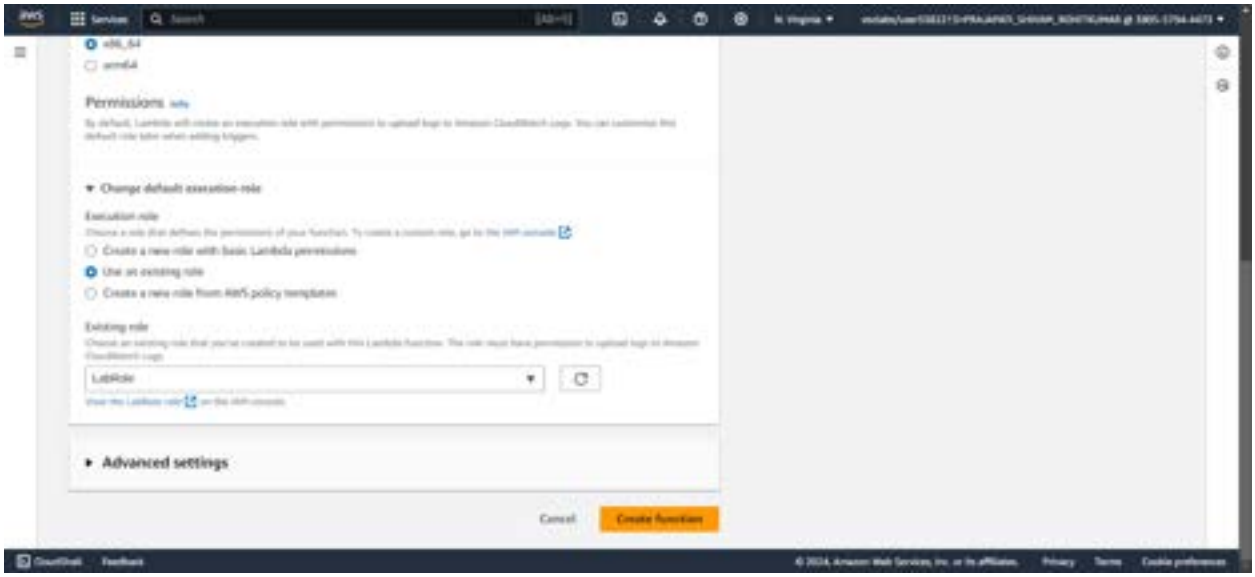


Step 4: Give your Lambda function a name and choose a programming language. The code editor only supports Node.js, Python, and Ruby, so in my case I have chosen **Python 3.12**. Set the **architecture to x86**. For the execution role, select '**Use an existing role**,' then pick '**Lab role**' from the dropdown menu under existing roles .

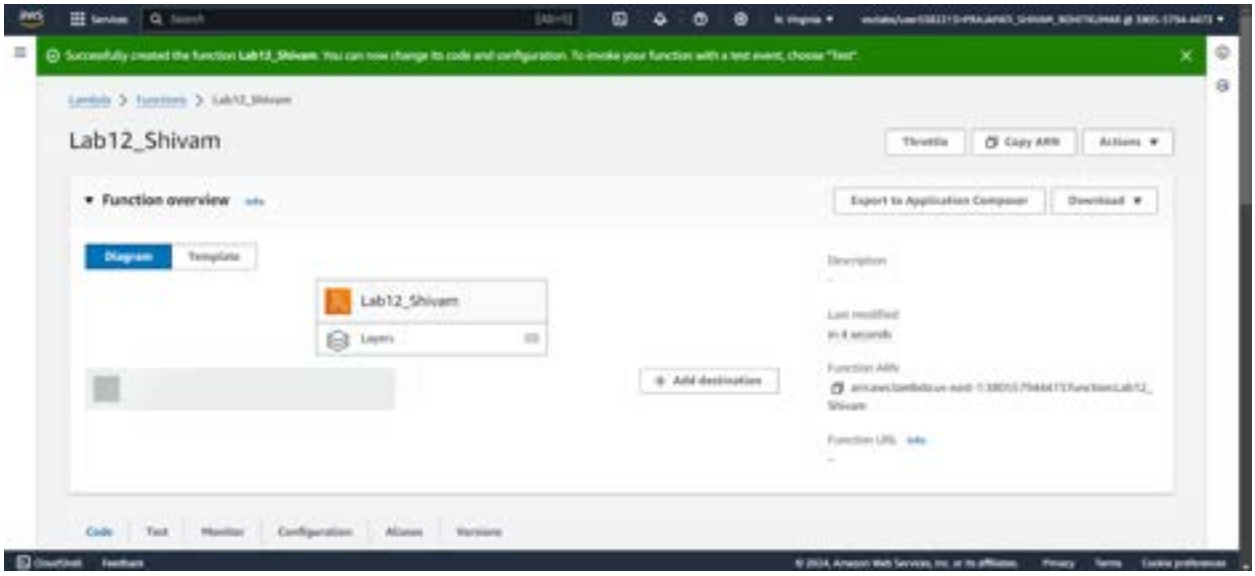
(This is because the Lab role already has the permissions needed for Lambda to run properly, so you don't need to create a new role from scratch. It's a quicker and more convenient option)



Given proper function name and selected language for Lambda function

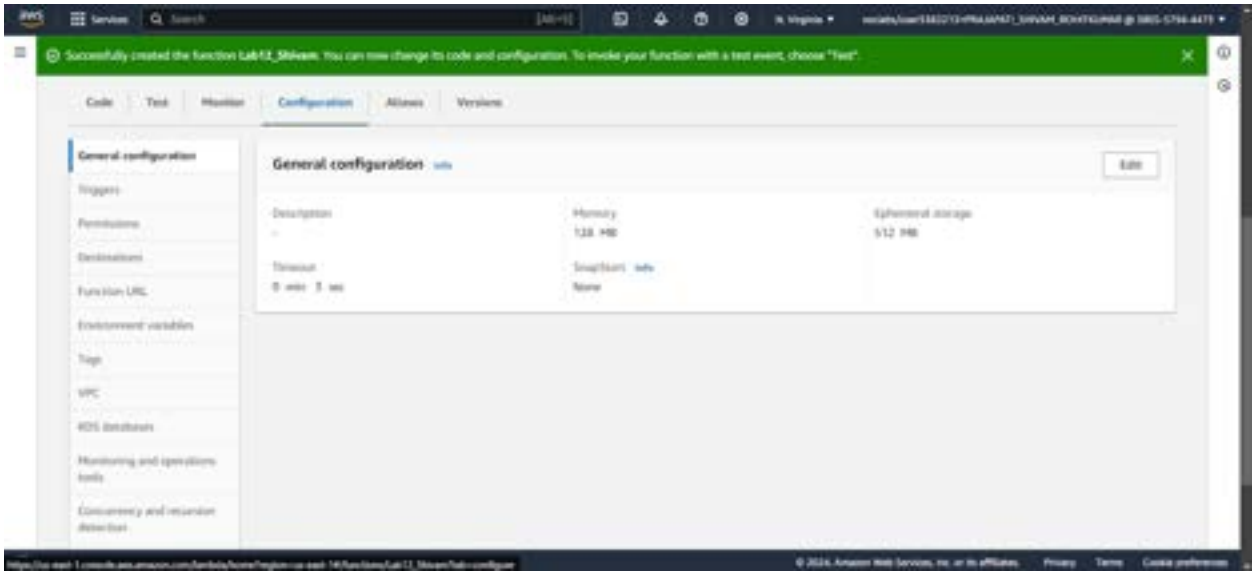


Selected Appropriate Execution role

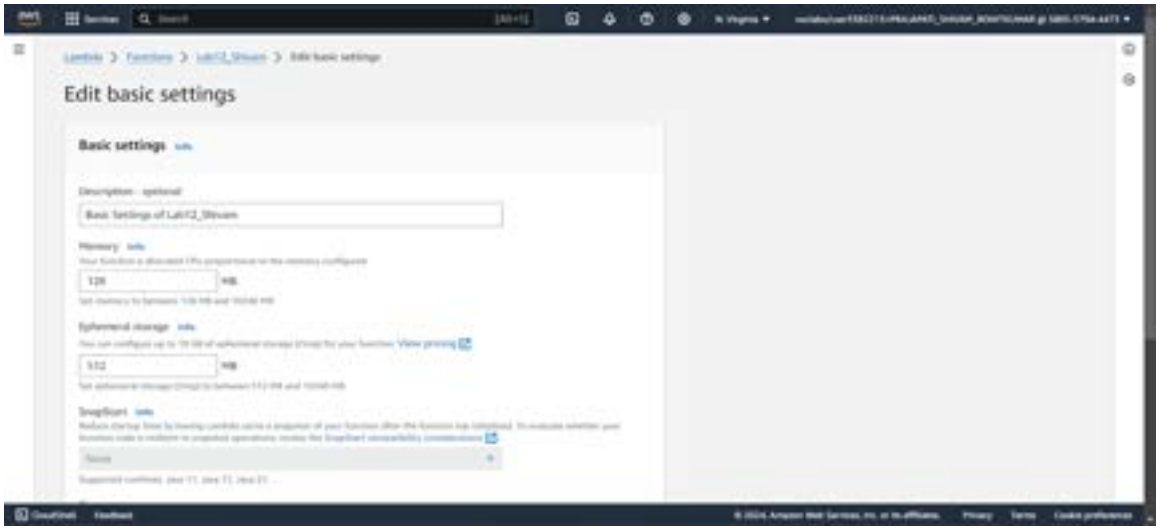


Successfully created the Lambda function.

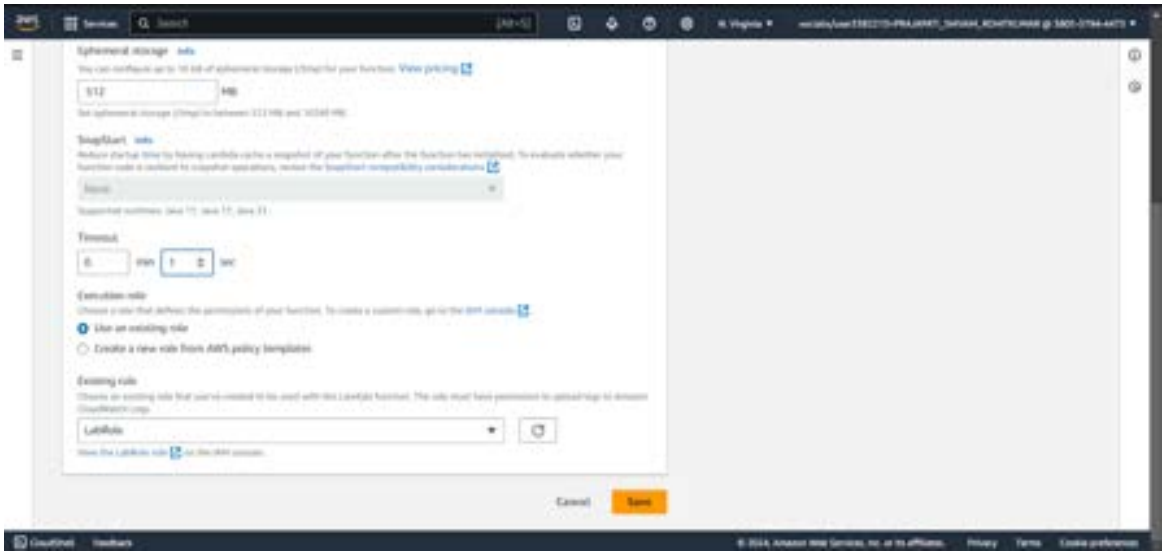
Step 5: To view or change the basic settings, go to the 'Configuration' tab and click 'Edit' under 'General settings.' (THIS STEP IS OPTIONAL)



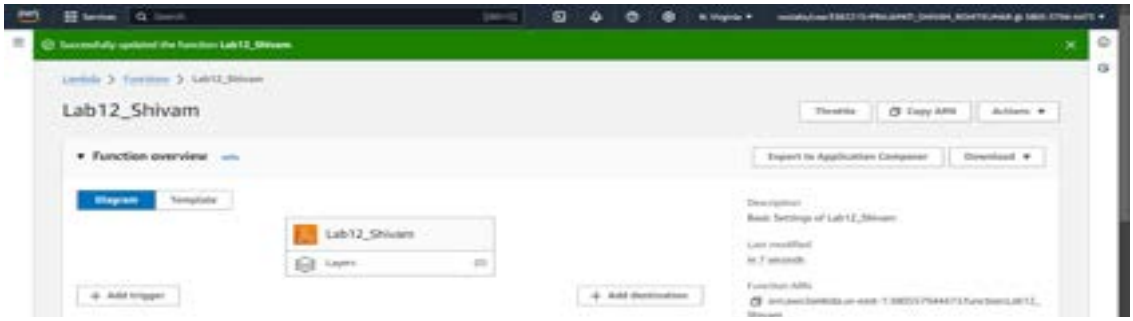
You can add a description and adjust the memory and timeout settings. I've changed the timeout to 1 second, as that's enough for now.



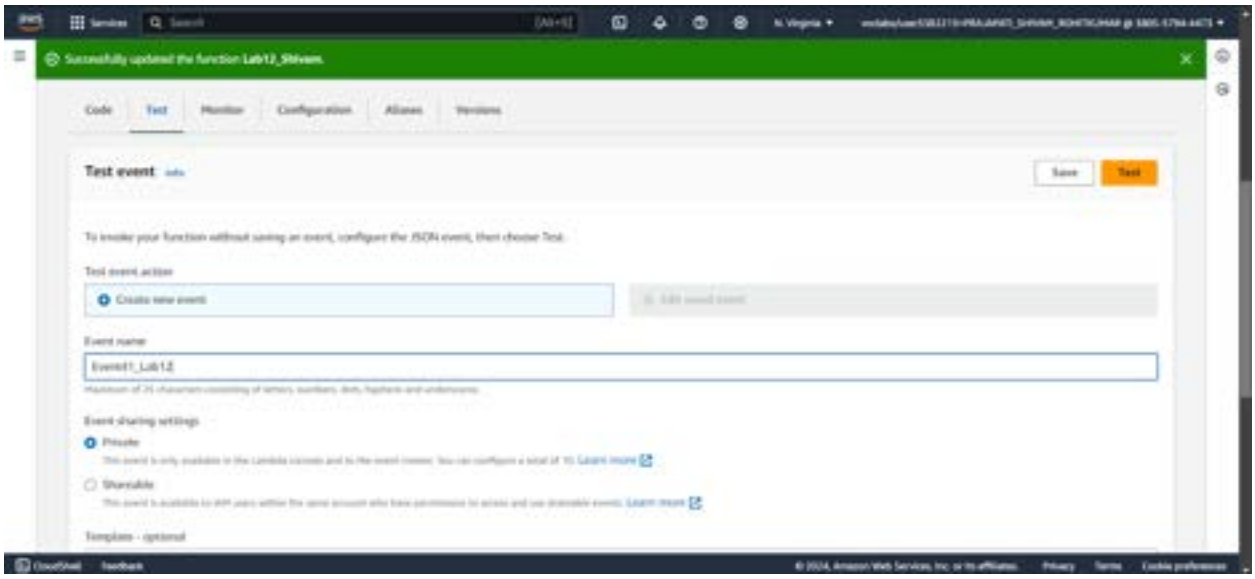
Given some description for your settings

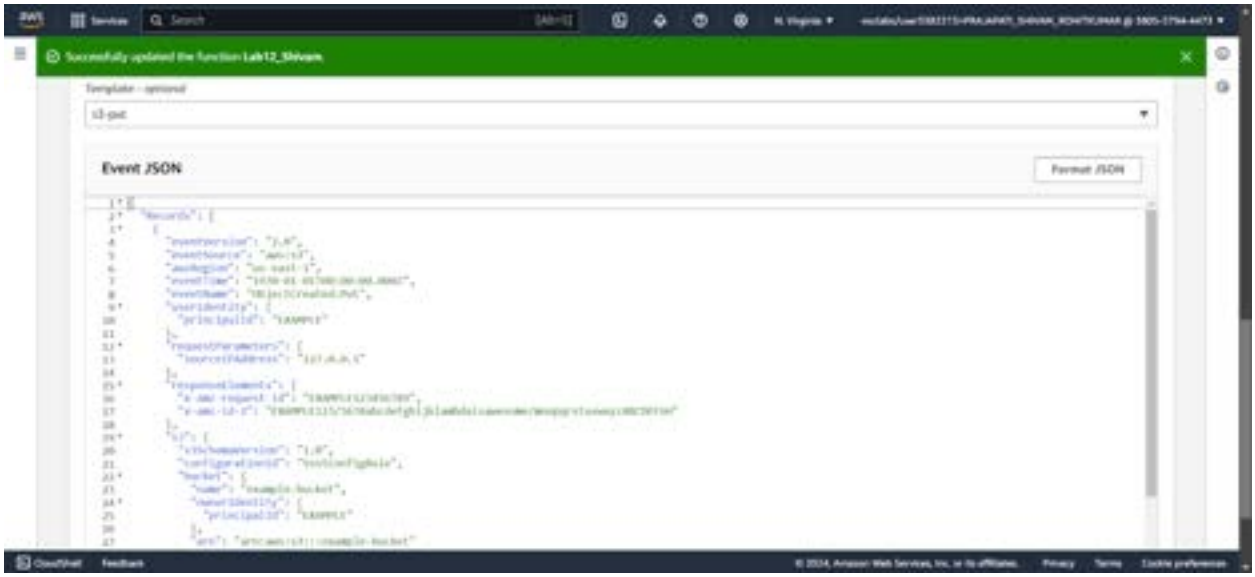


Click on Save.



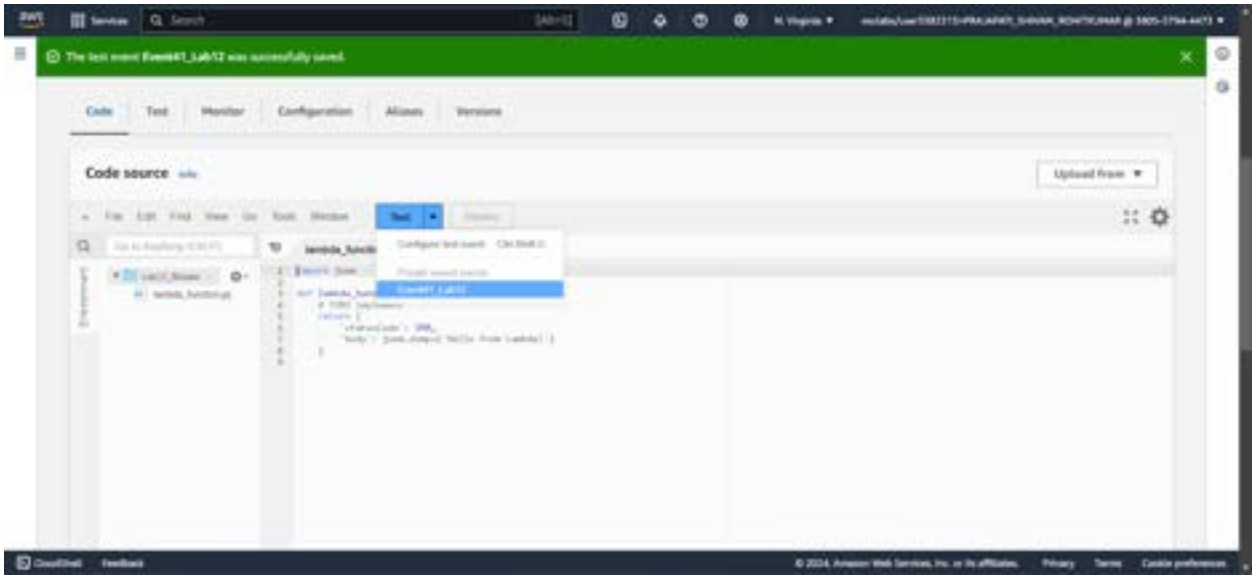
Step 6: Click on the "Test" tab, then select "Create a new event." Give the event a name, set "Event Sharing" to private, and choose the "S3 Put" template.S3 (Simple Storage Service) template allows you to test your Lambda function specifically for events related to uploading files to an S3 bucket.



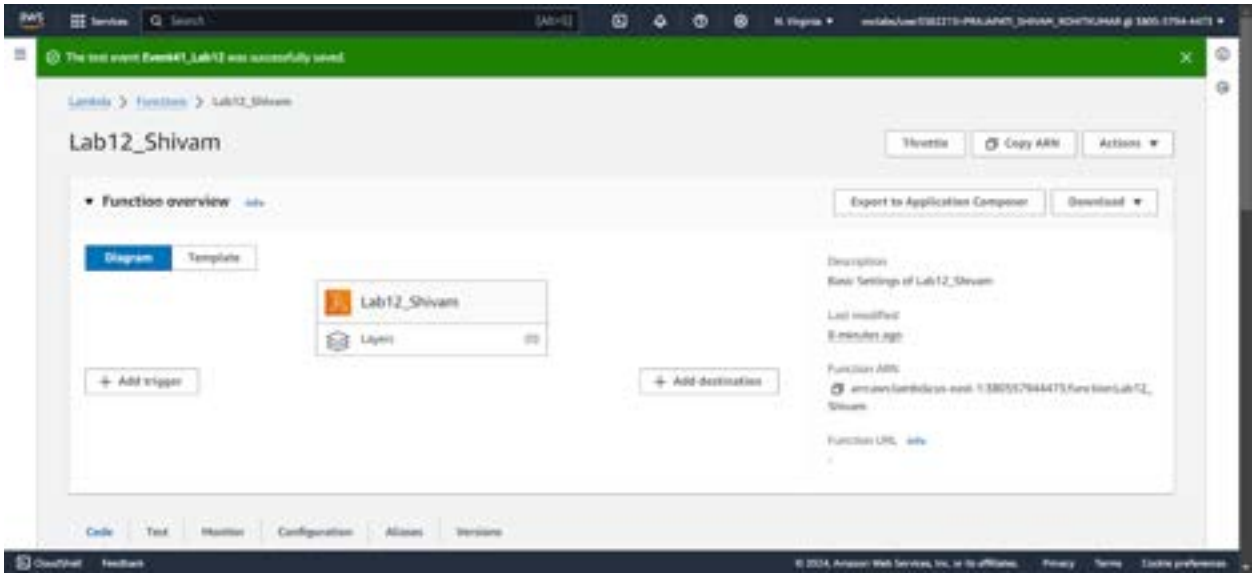


Event Jason code will be automatically generated once S3 -put is selected.

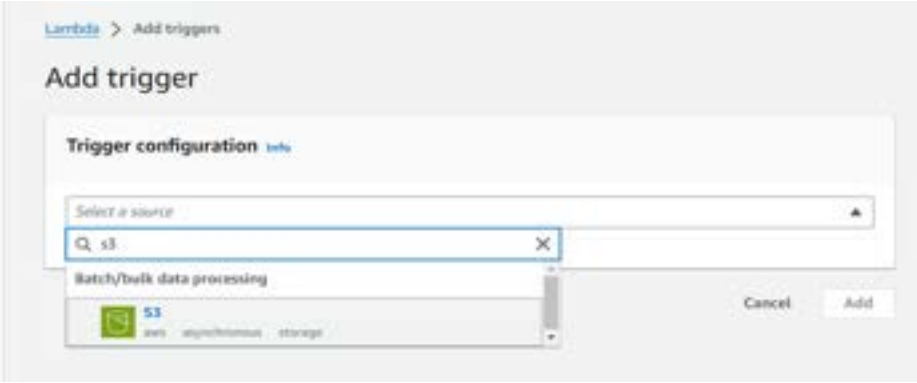
Step 7: Now In the Code section select the created event from the dropdown .

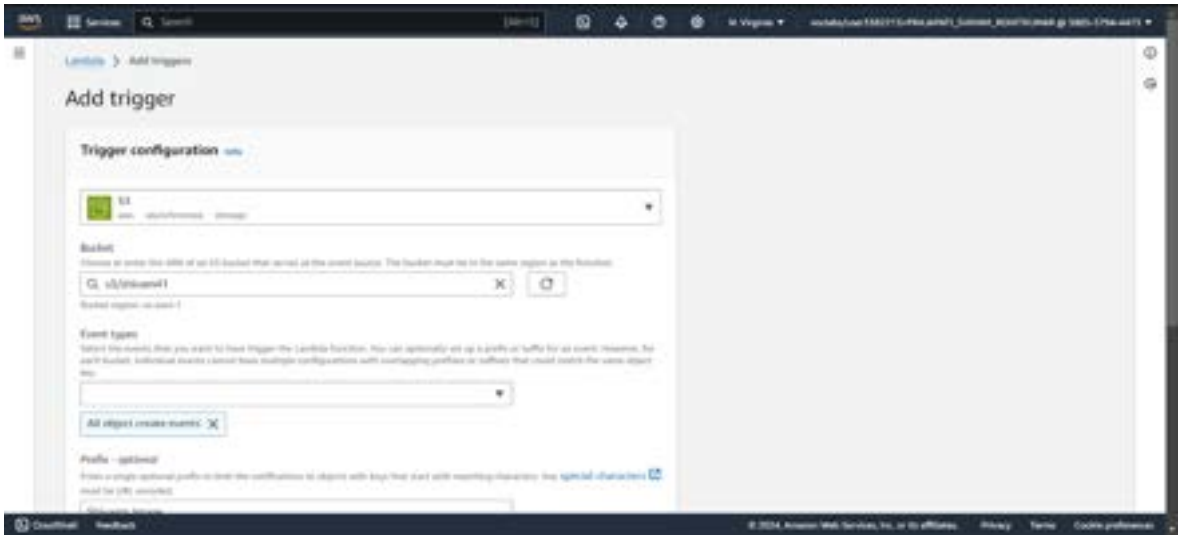


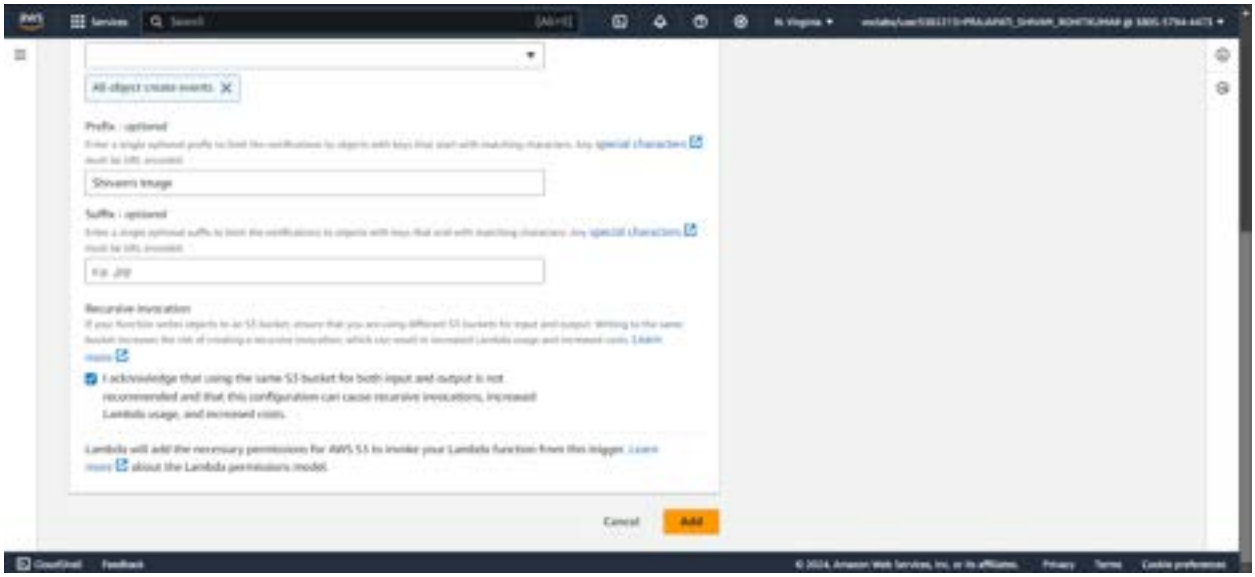
Step 8: In the Lambda function, click on "**Add Trigger.**" Adding a trigger allows your Lambda function to automatically run in response to specific events such as uploads to an S3 bucket



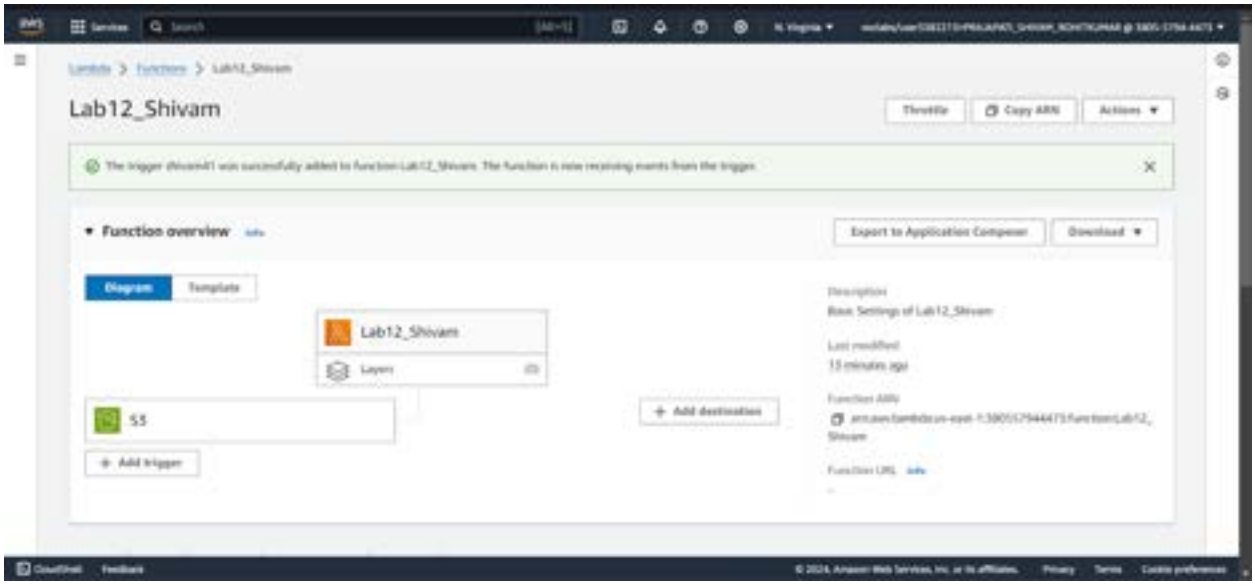
Now select the source as S3, then choose the bucket name from the dropdown menu. Keep the other settings as default, and you can also add a prefix for the image if you want.A prefix for an image (or any file) in S3 is a string that you can use to organize or filter files within a bucket. It acts like a folder name, helping to categorize your files.



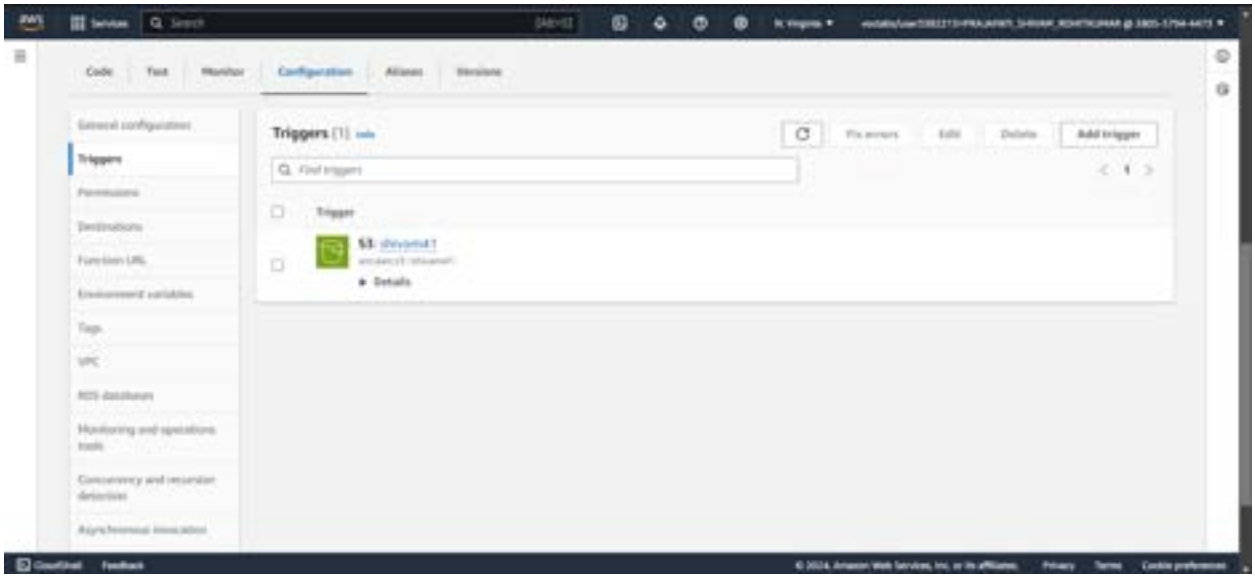




Click on save

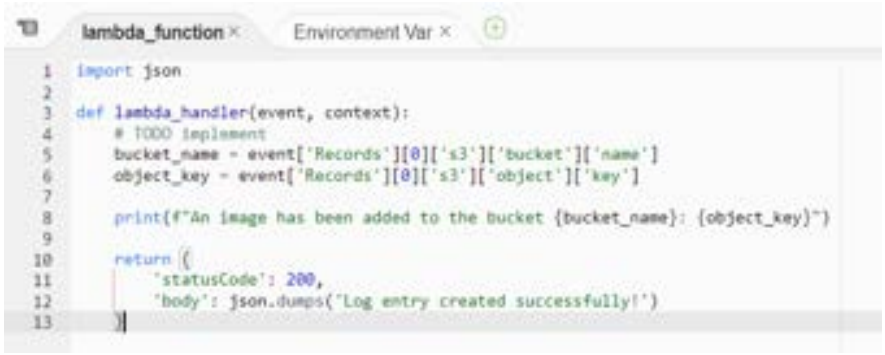


Successfully triggered the event

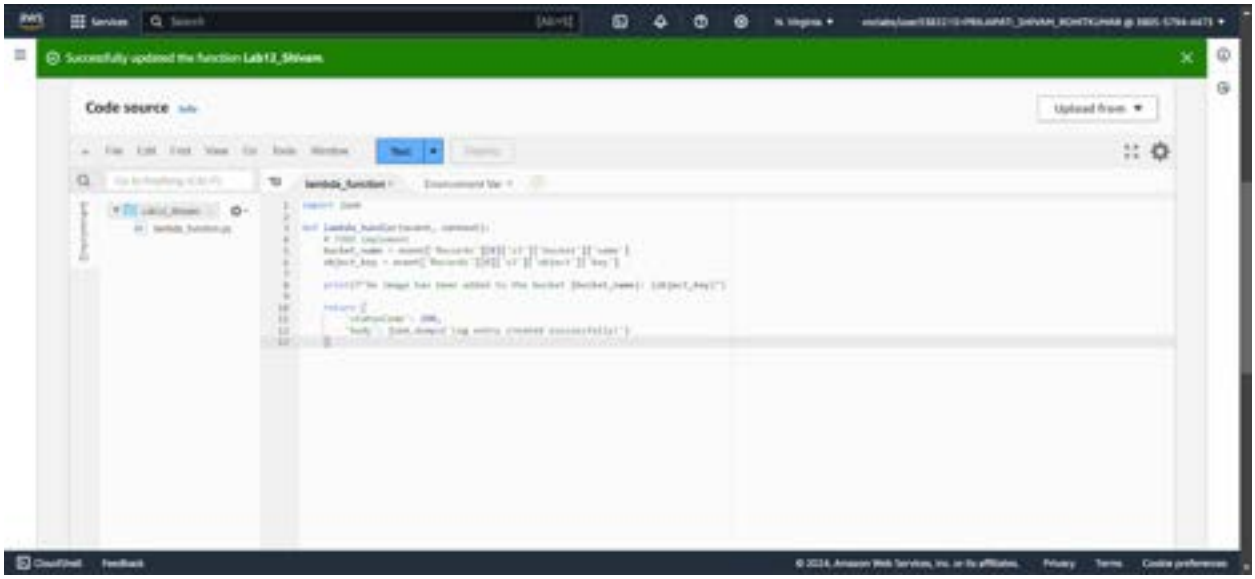


Step 9: Now Write code that logs a message like “An Image has been added” when triggered. Save the file and click on deploy.

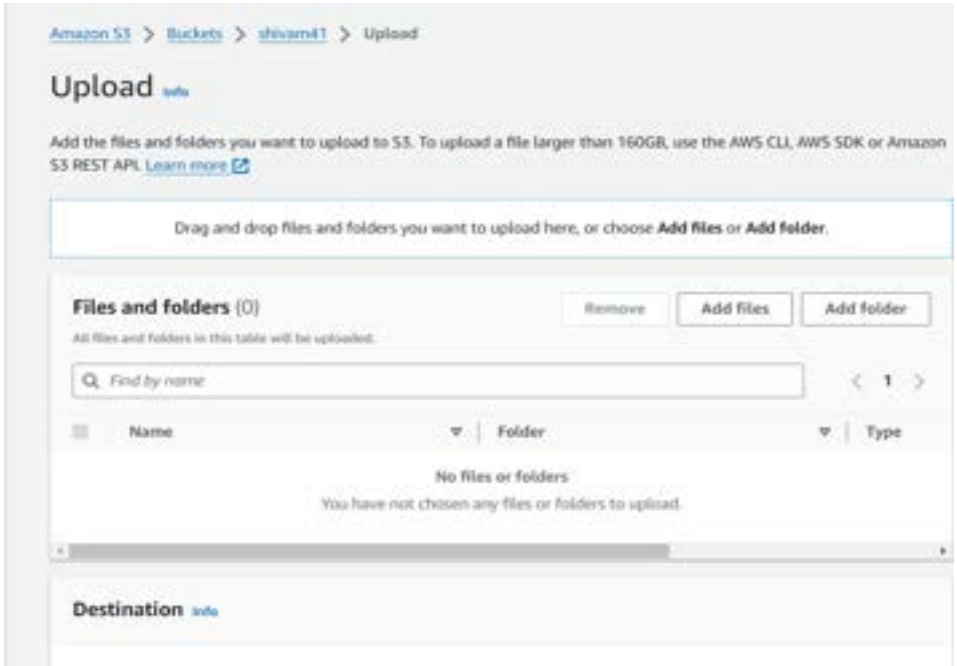
```
import json
def lambda_handler(event, context):
    # TODO implement
    bucket_name = event['Records'][0]['s3']['bucket']['name']
    object_key = event['Records'][0]['s3']['object']['key']

    print(f"An image has been added to the bucket {bucket_name}: {object_key}")

    return {
        'statusCode': 200,
        'body': json.dumps('Log entry created successfully!')
    }
```



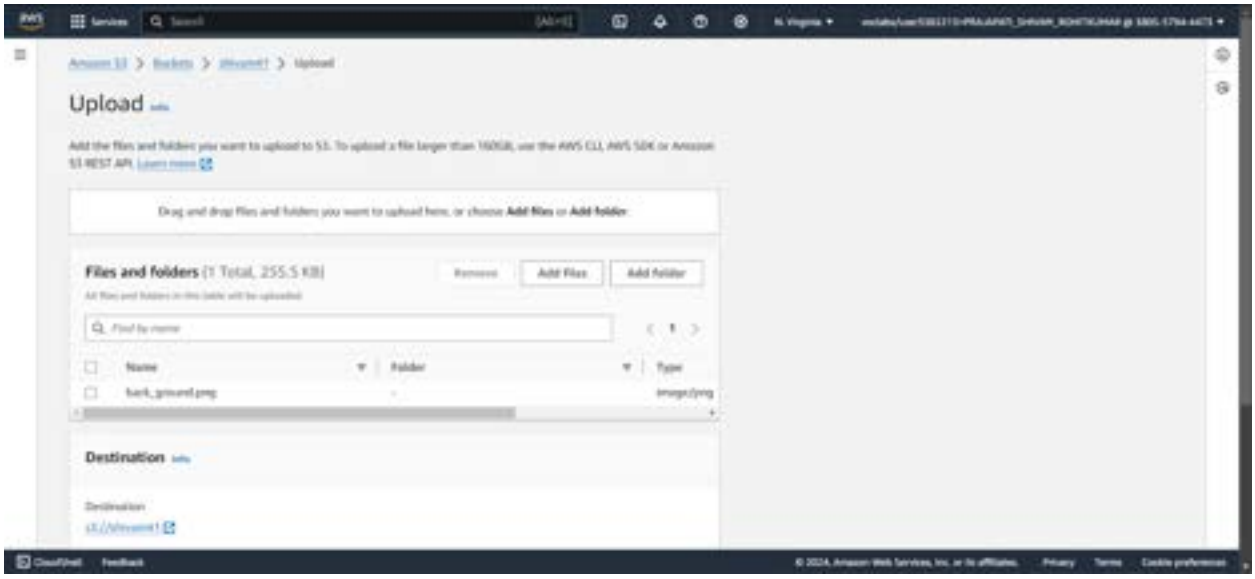
Save it by Control + S and deploy

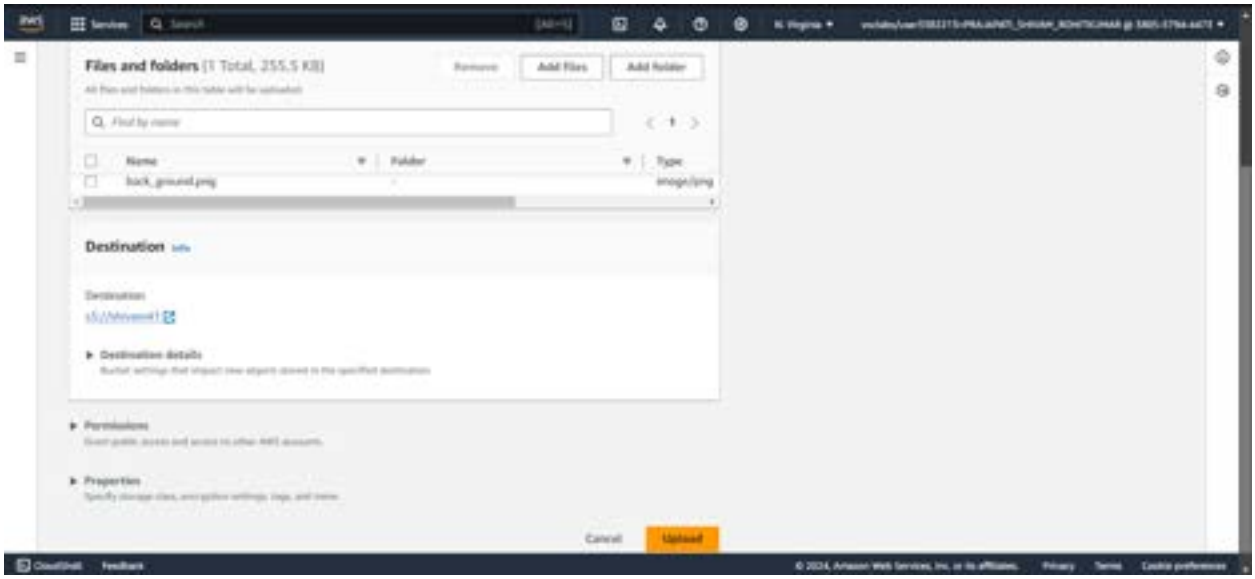


Step 10: Now we will upload any image to the bucket



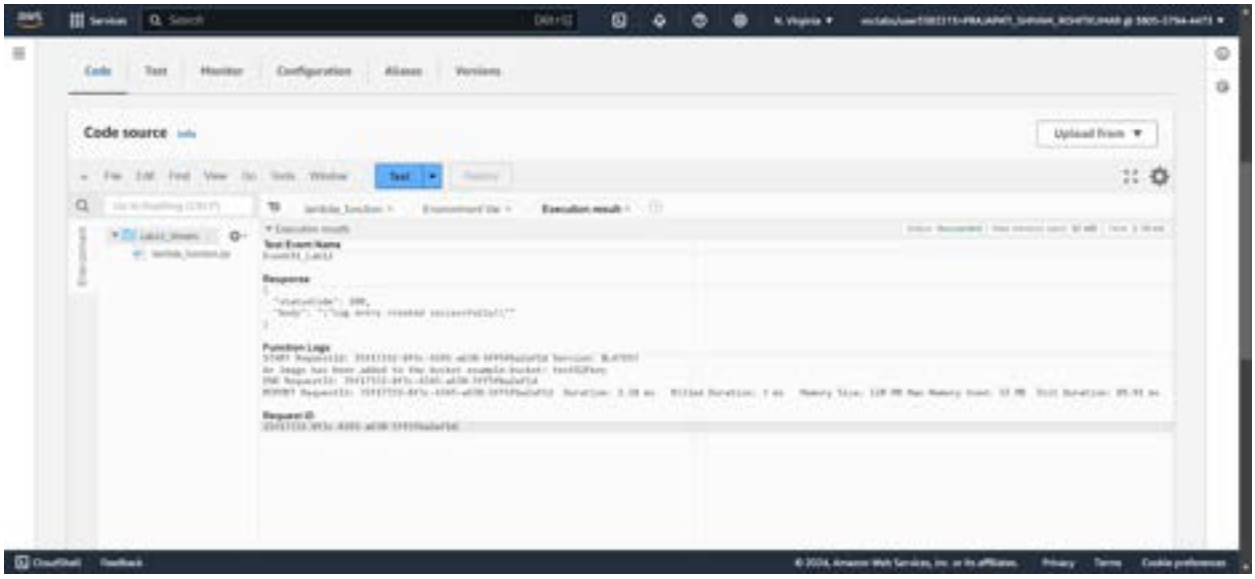
Click on add file where you can upload any image of your choice in your bucket



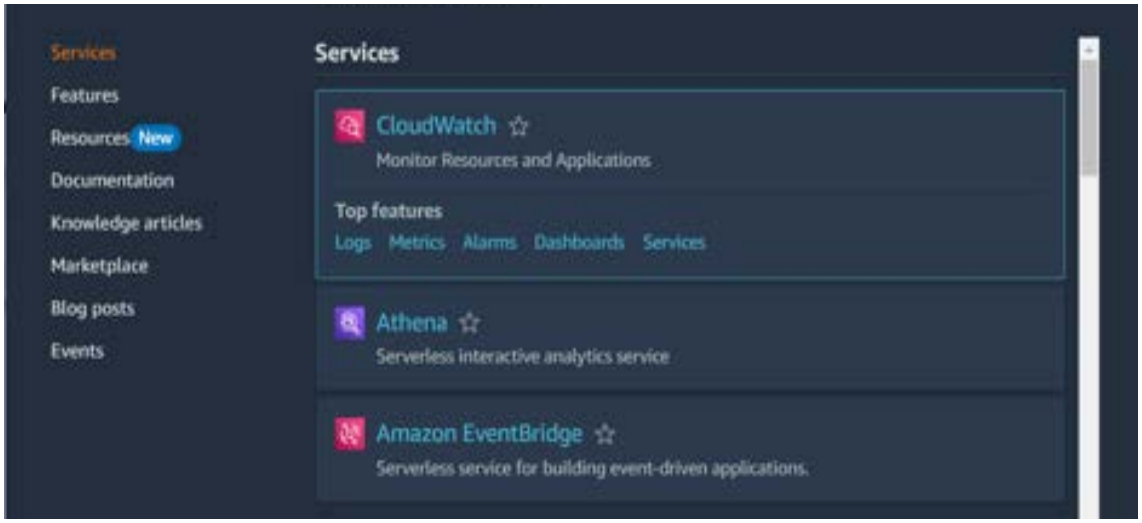


Click on Upload

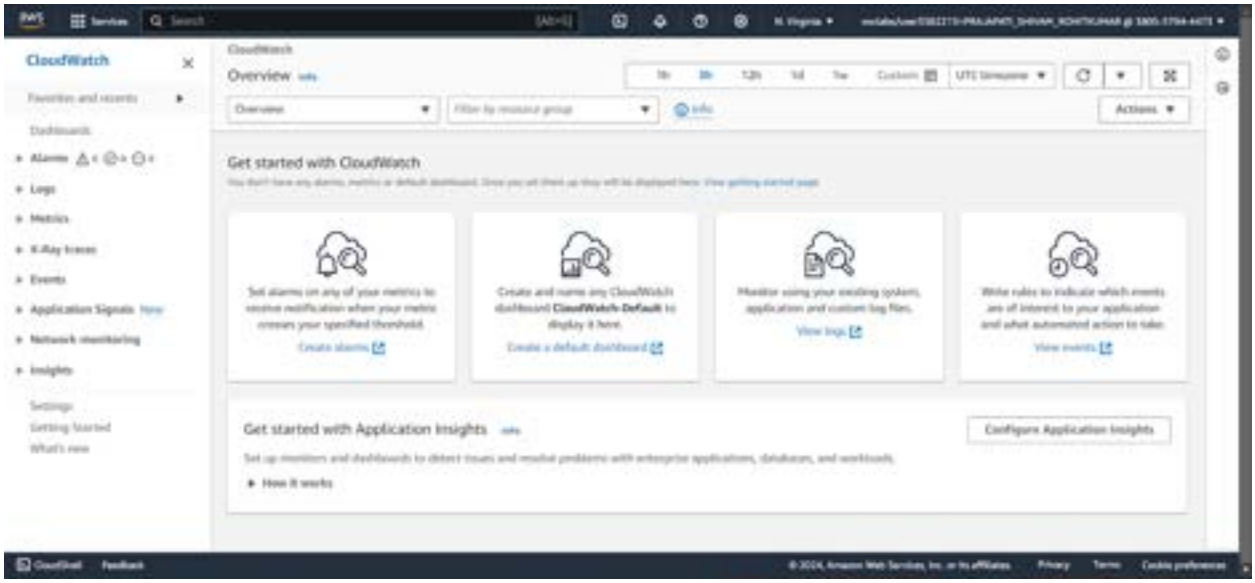
Step 11: Now click on "Test" in Lambda to see if it logs the activity when an image is added to S3.



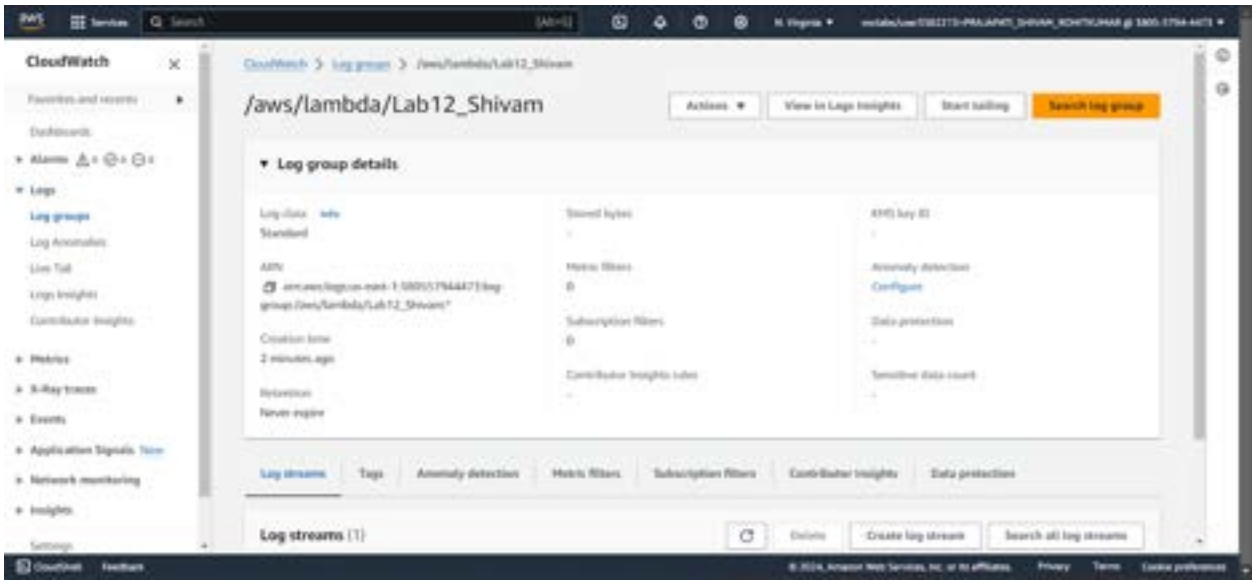
Step 12: Now let's check the logs on CloudWatch. Go to the "Monitor" section and click on "View CloudWatch Logs.



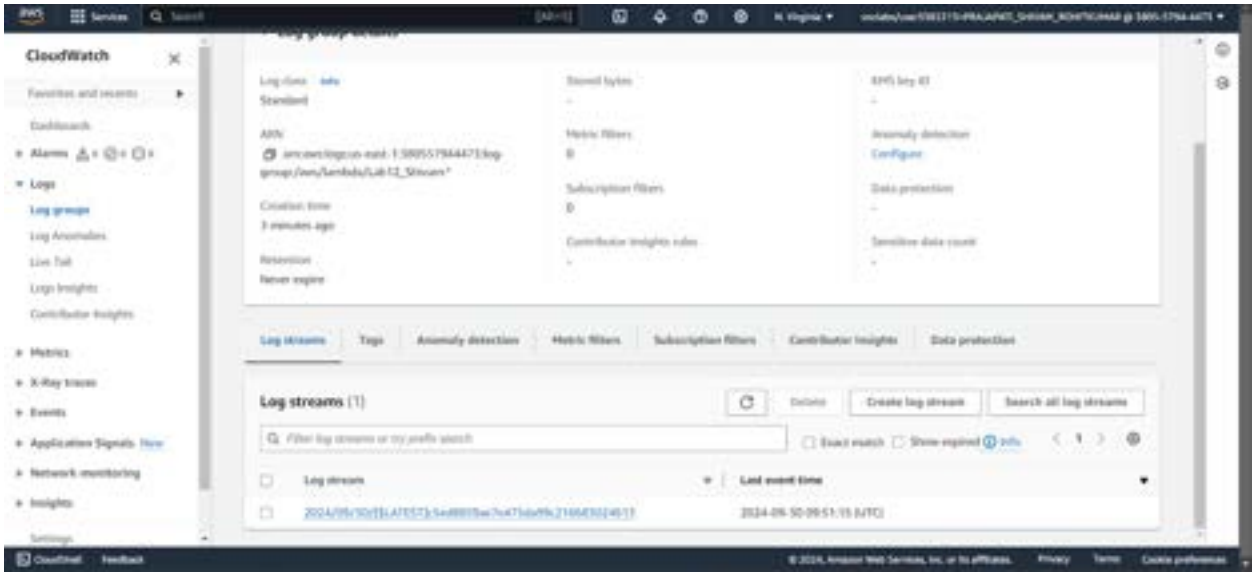
Click on the CloudWatch:



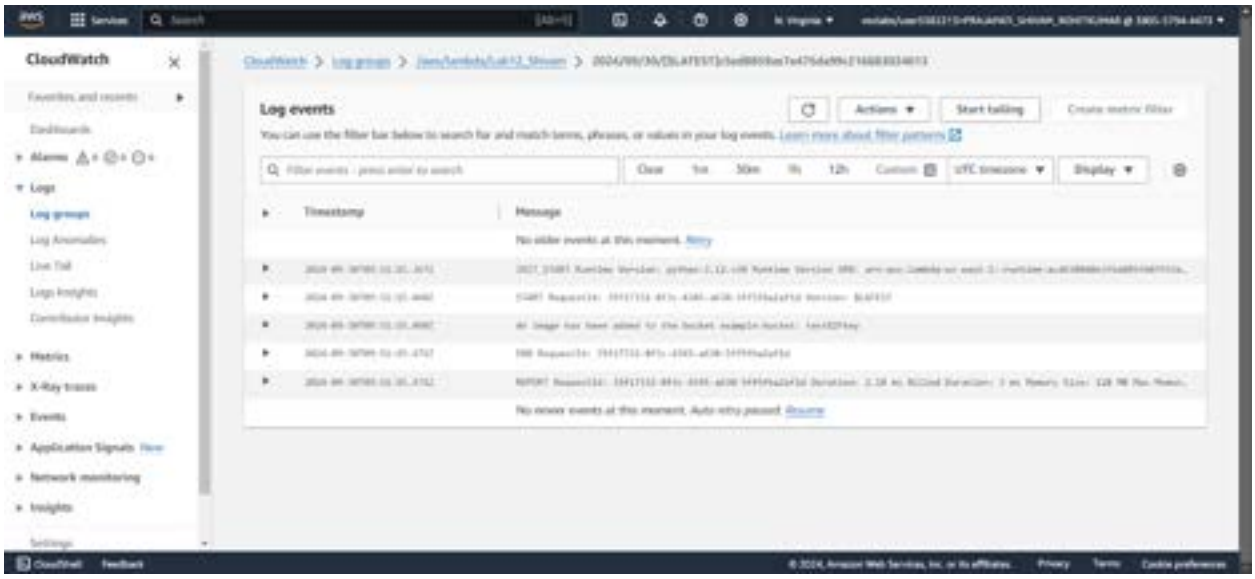
Click on the logs:



Click on the log group:



Scrolled down and click on the log stream:



CONCLUSION:

In this experiment, we successfully created an AWS Lambda function that logs a message when an image is uploaded to an S3 bucket. It's important to choose the S3 Put template for the event; otherwise, the code will give an error. The function was triggered correctly when files were uploaded to S3, showing that Lambda's event-driven design works well. This experiment showed how Lambda can respond to S3 events and how to fix common problems with the event setup.