Experiment No: 1(a)

STATIC HOSTING :

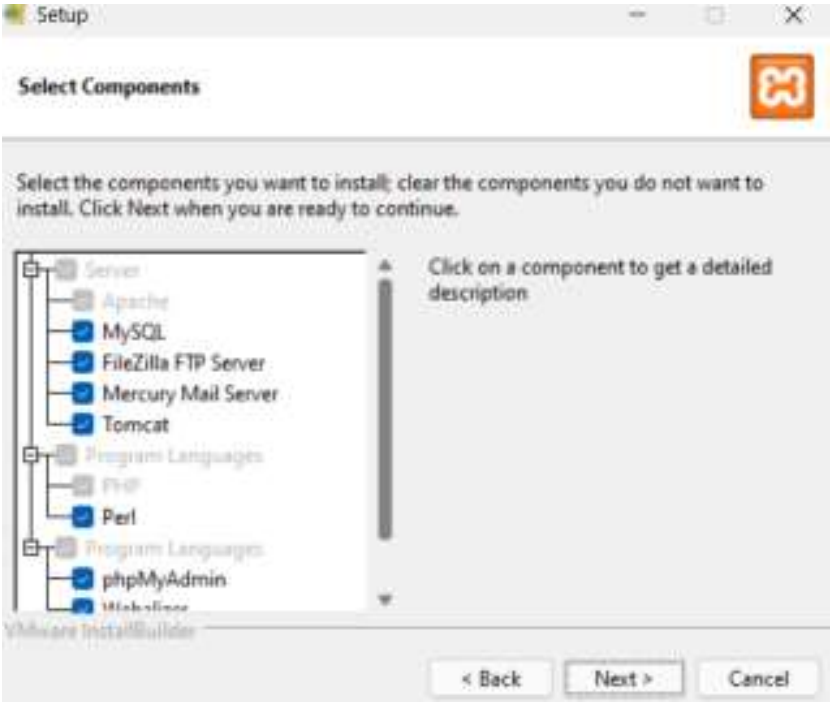
1) On local server (XAMPP)

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

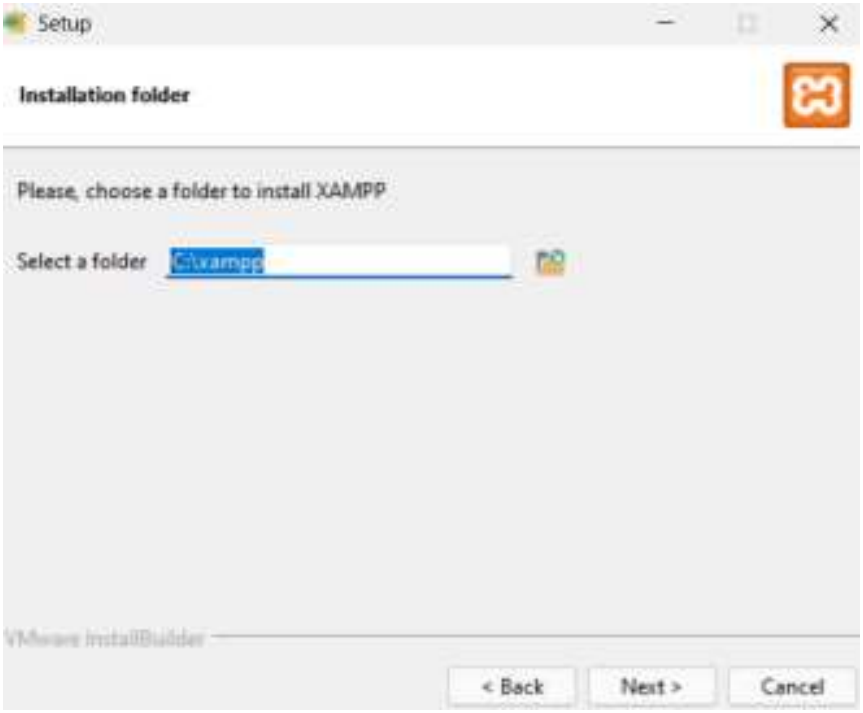
a) Select your OS. It will automatically start downloading

b) Open the setup file. Click on Next

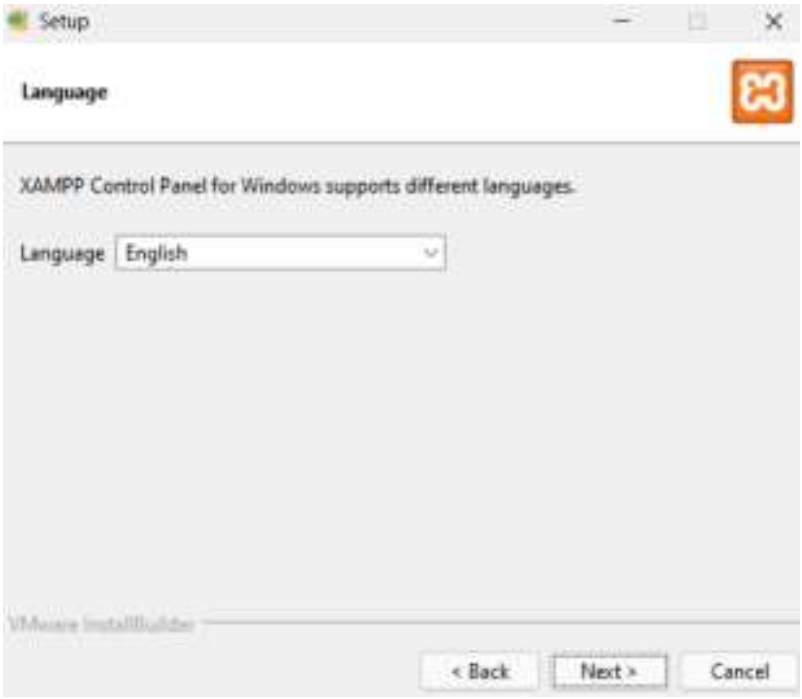
c) Select all the required components and click next



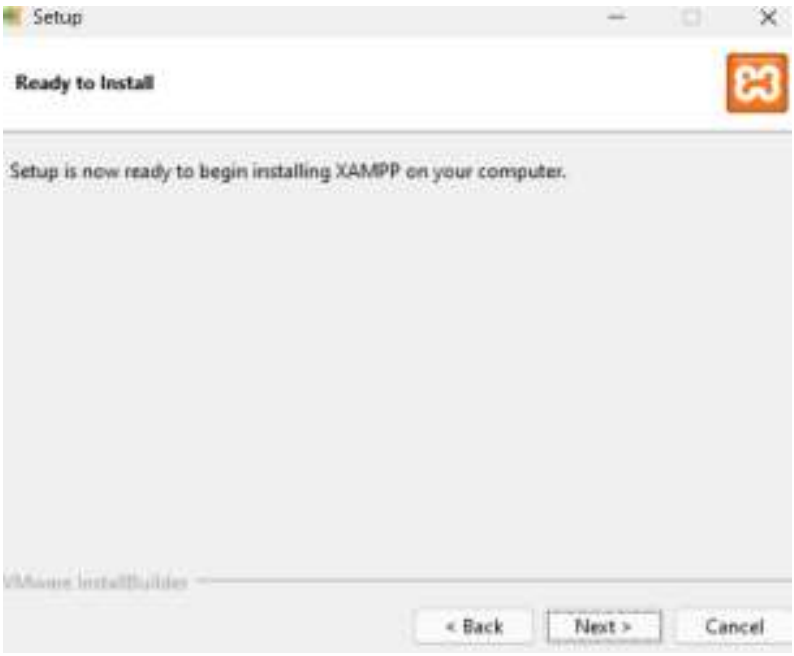
d) Choose the folder to install XAMPP in. Make sure the folder is empty. Click next



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

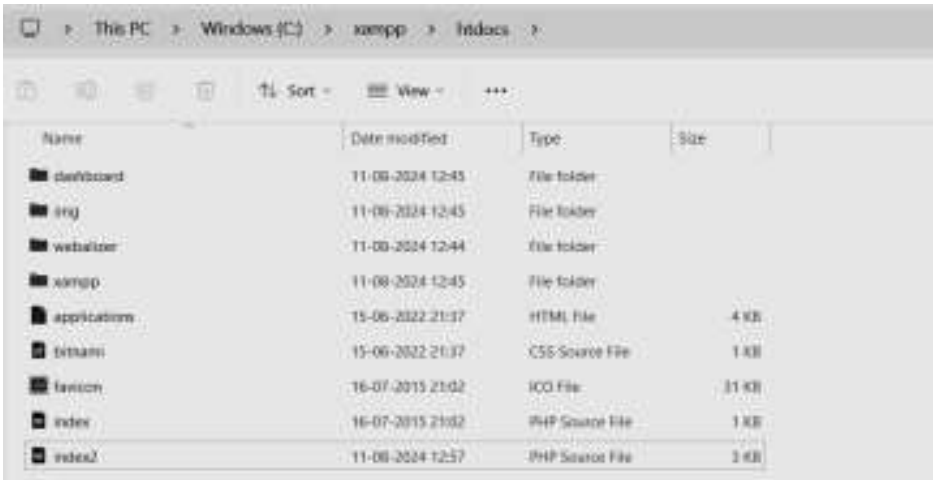


f)Click on Next

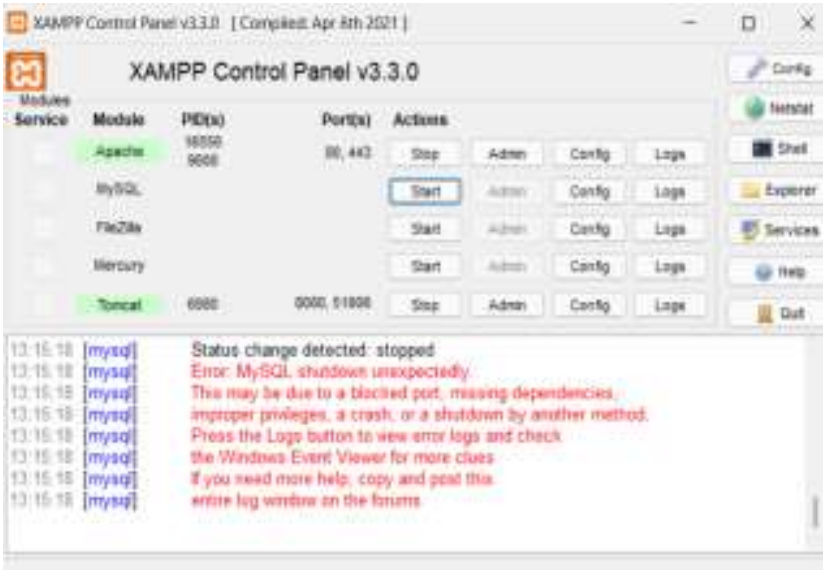


Step 2: Setup a file that is to be hosted on the server. Make sure the file has extension .php

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)



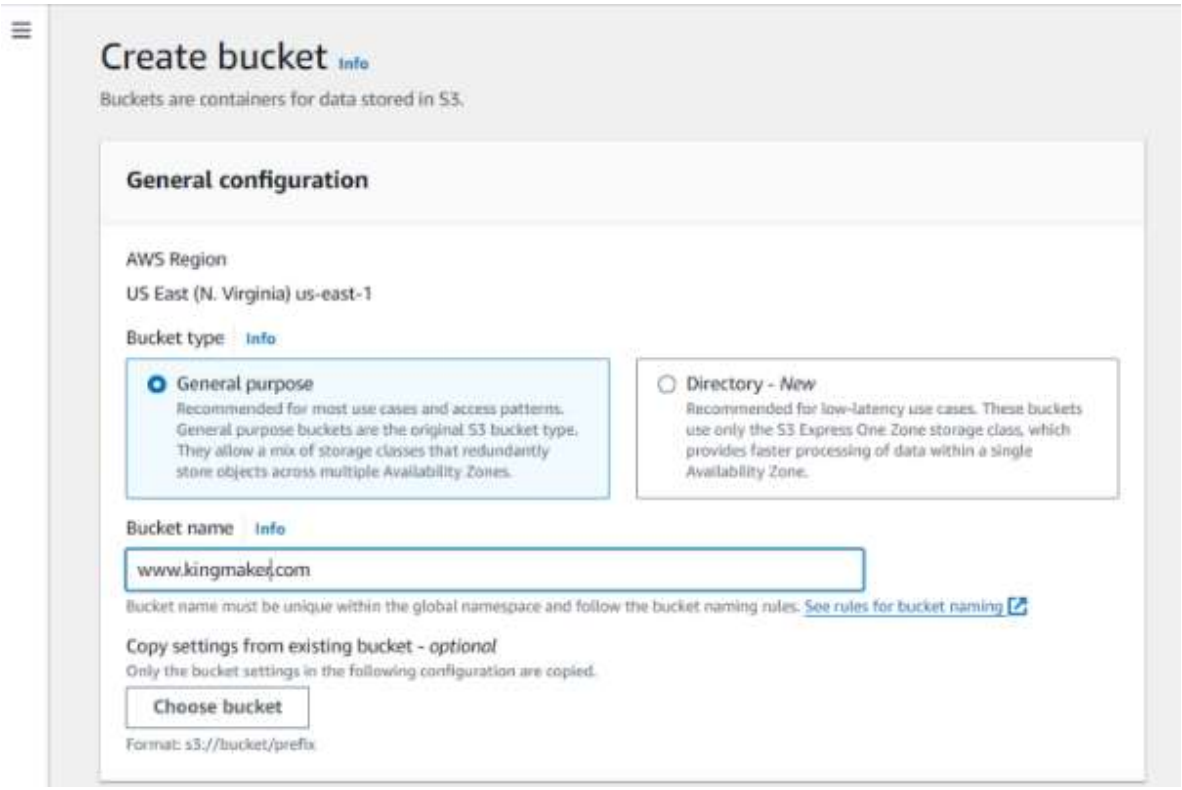
Step 5: Open your web browser. Type localhost/index2.php. This will open your website on your browser.

On AWS S3

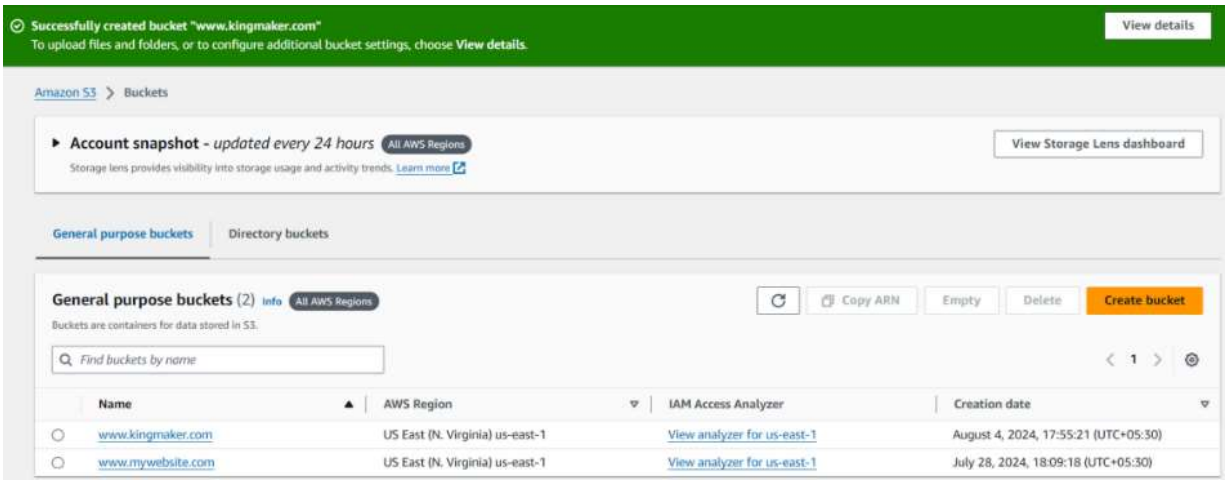
Step 1: Login to your AWS account. Go to services and open S3.

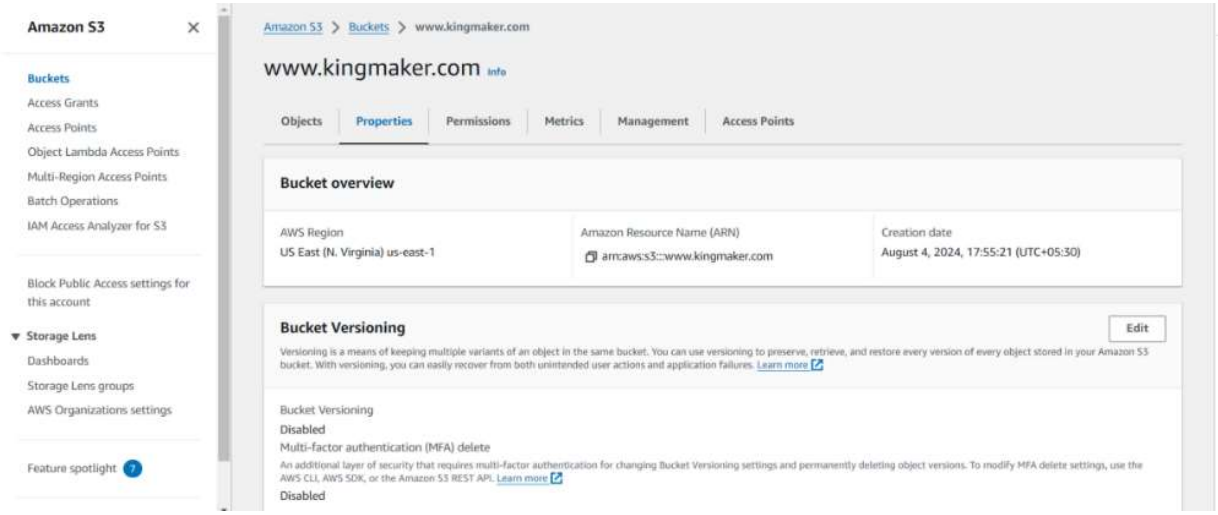
Step 2: Click on Create Bucket

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

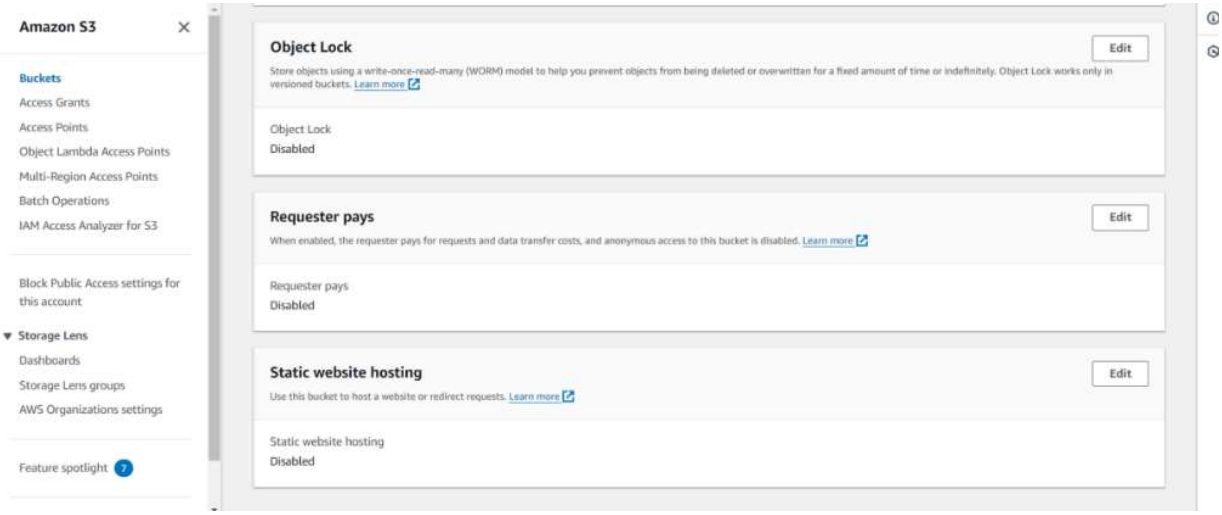


Step 4: Click on the name of your bucket and goto Properties

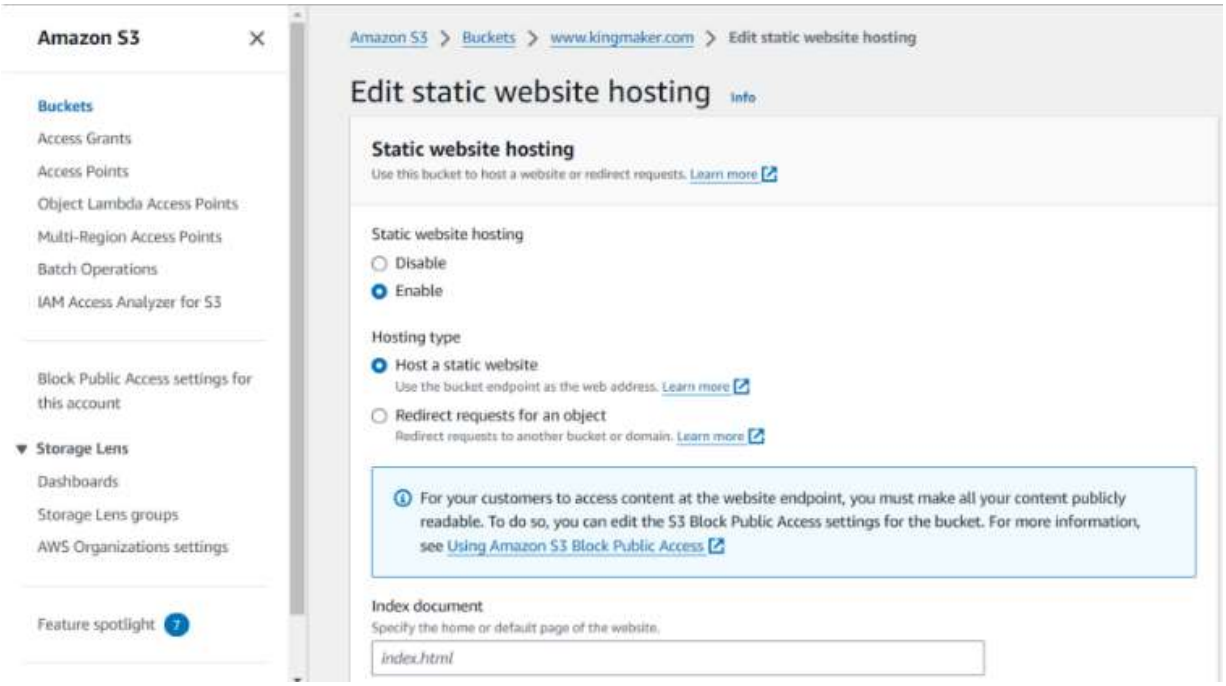




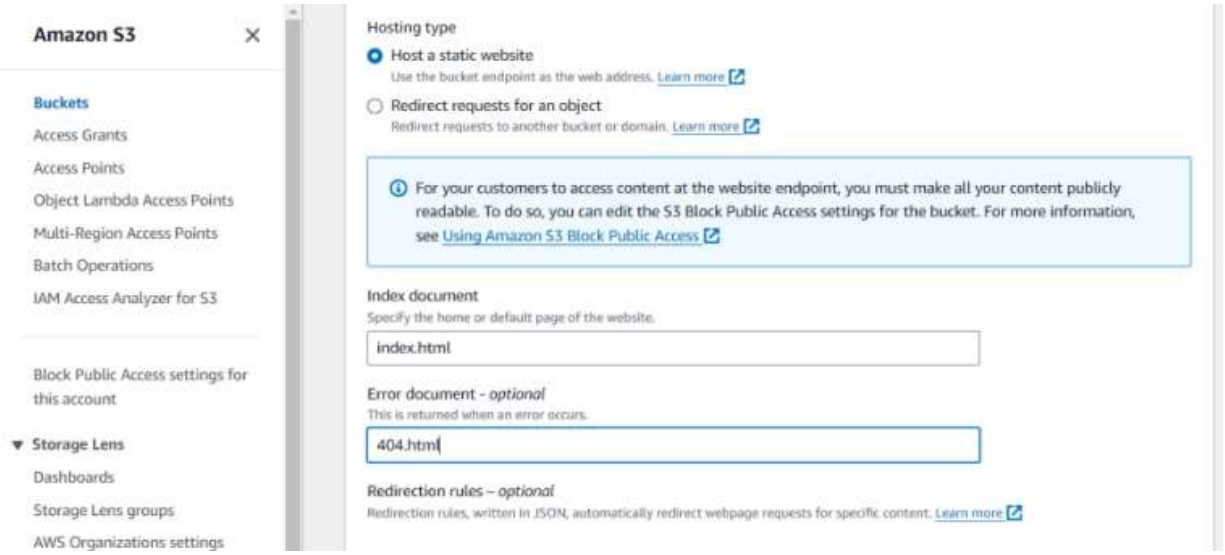
Step 5: Scroll down till you find Static website hosting, click on edit



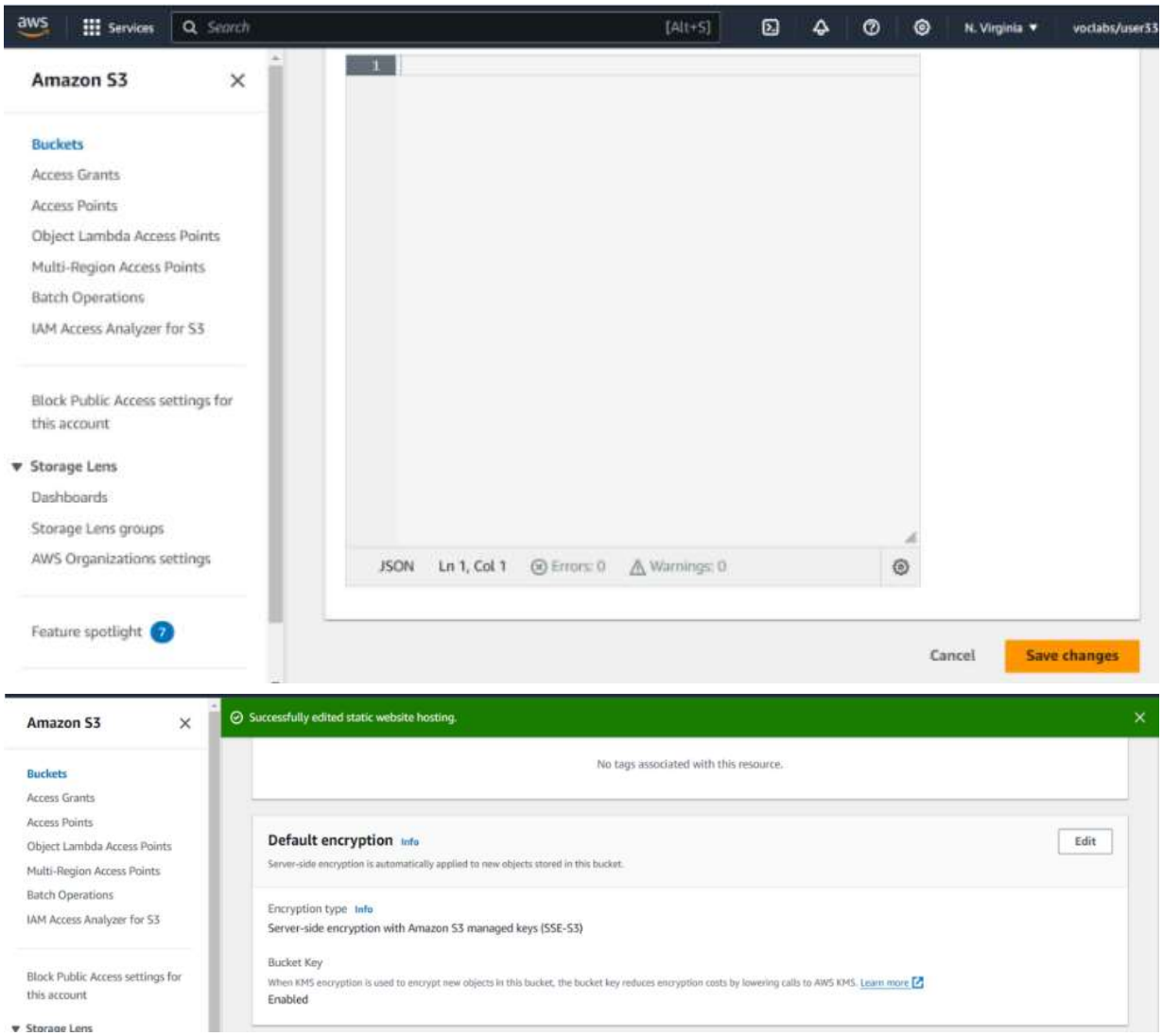
Step 6: Click on Enable static website hosting



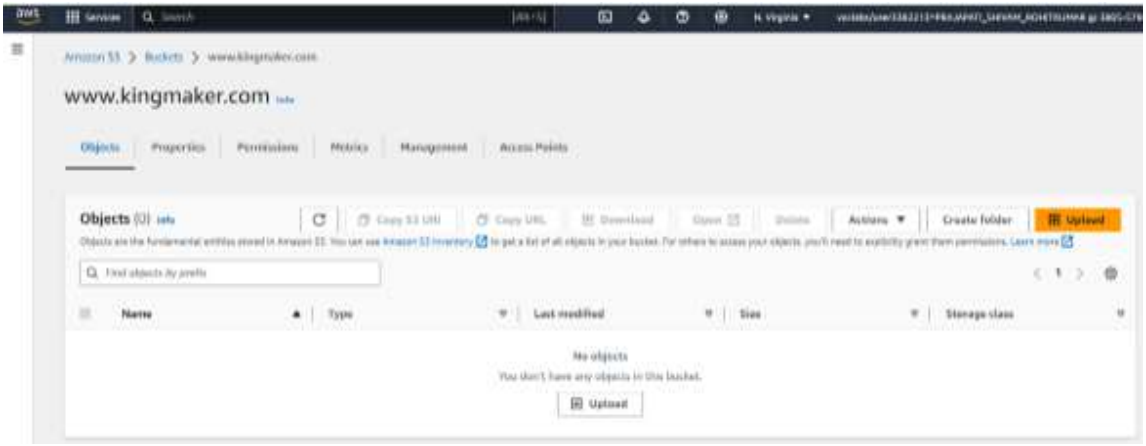
Step 7: Write the name of your document which you wanted to host on AWS from your local folder and in error document, give name as 404.html. Save your changes.



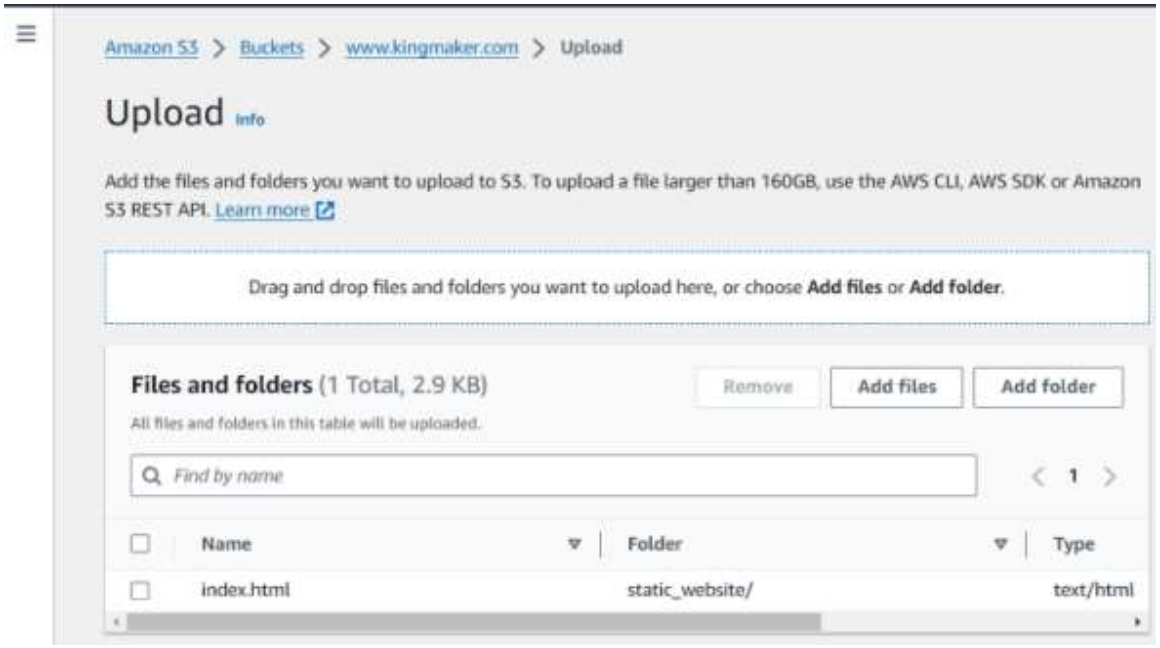
Step 8: Save the Changes

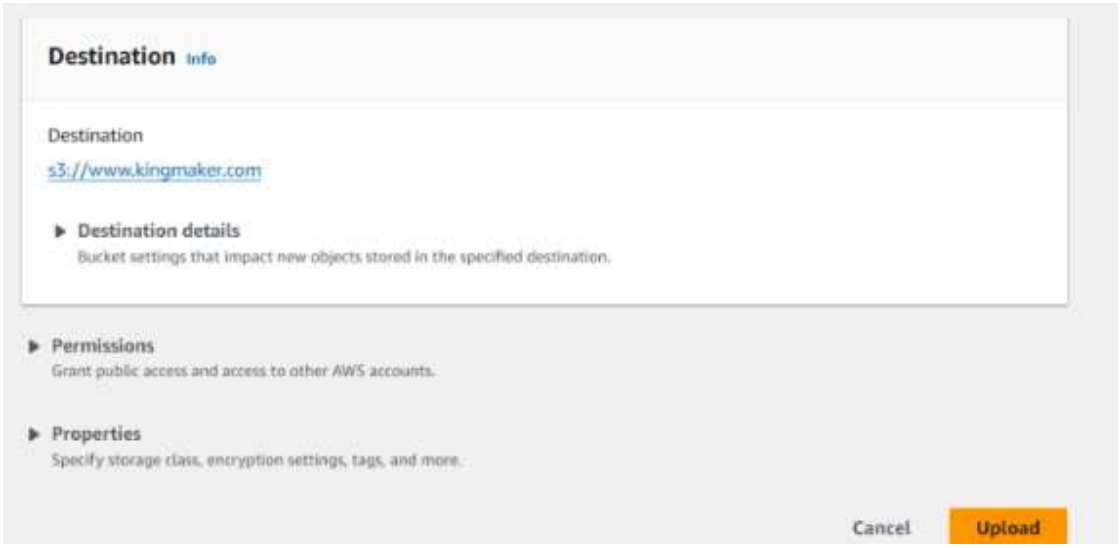


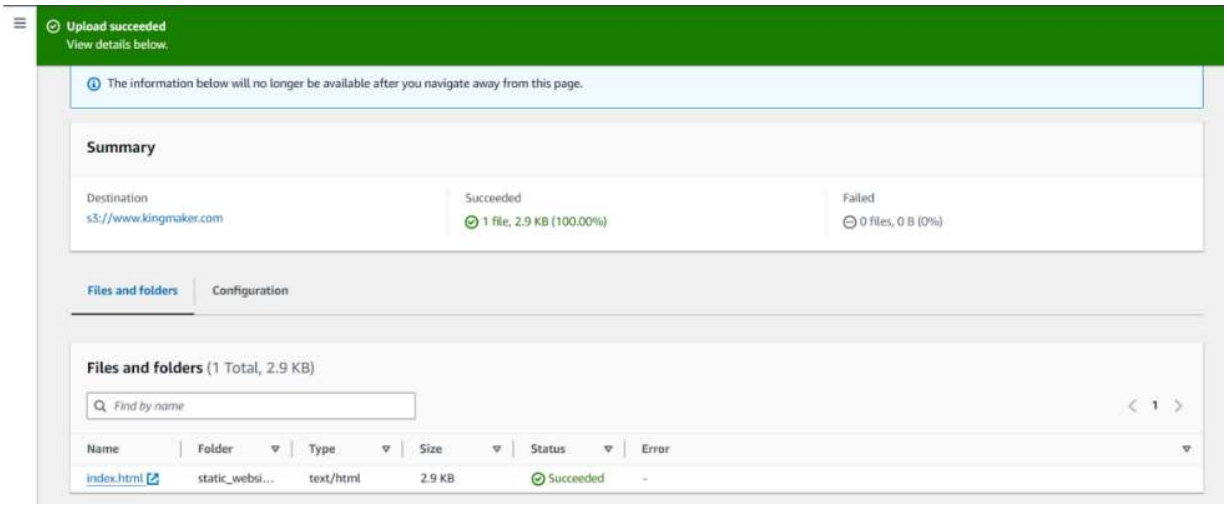
Step 9: Go to Objects tab and click on upload file



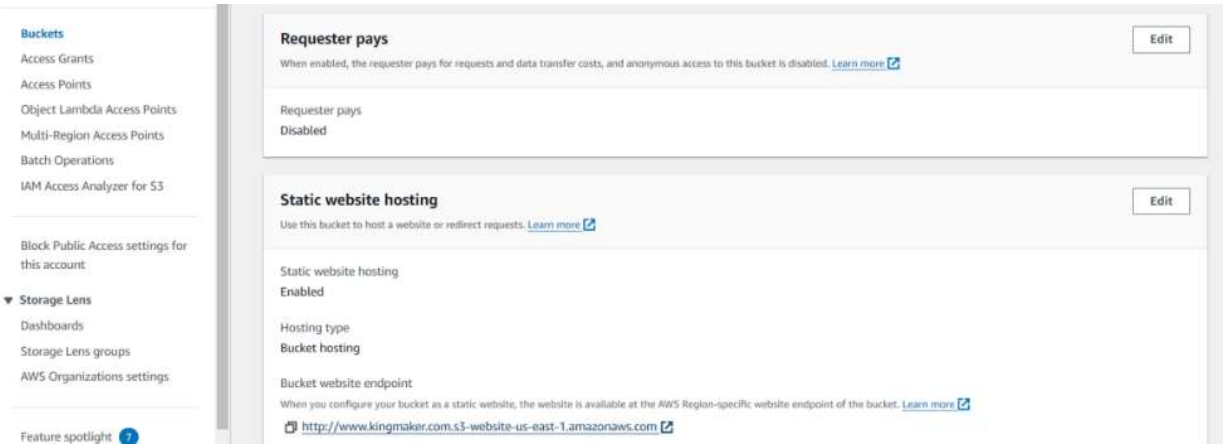
Step 10: Click on Add files. Add all the files you want to upload. Then scroll down and click on Upload







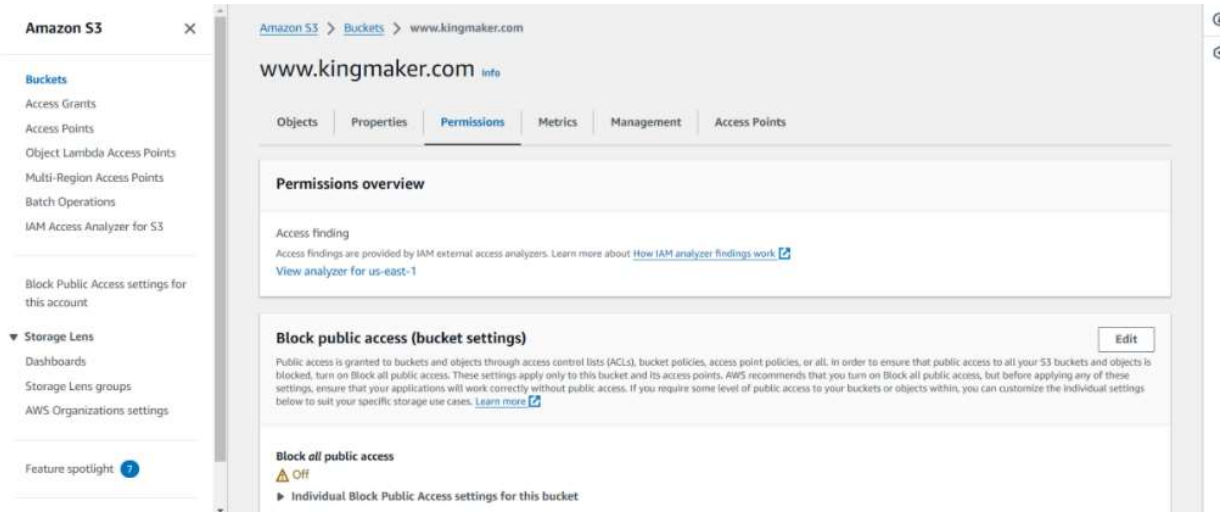
Step11: This will take you to the Objects screen. Switch to Properties, scroll down to Static web hosting. There you would find the link (Bucket website endpoint) to your website.



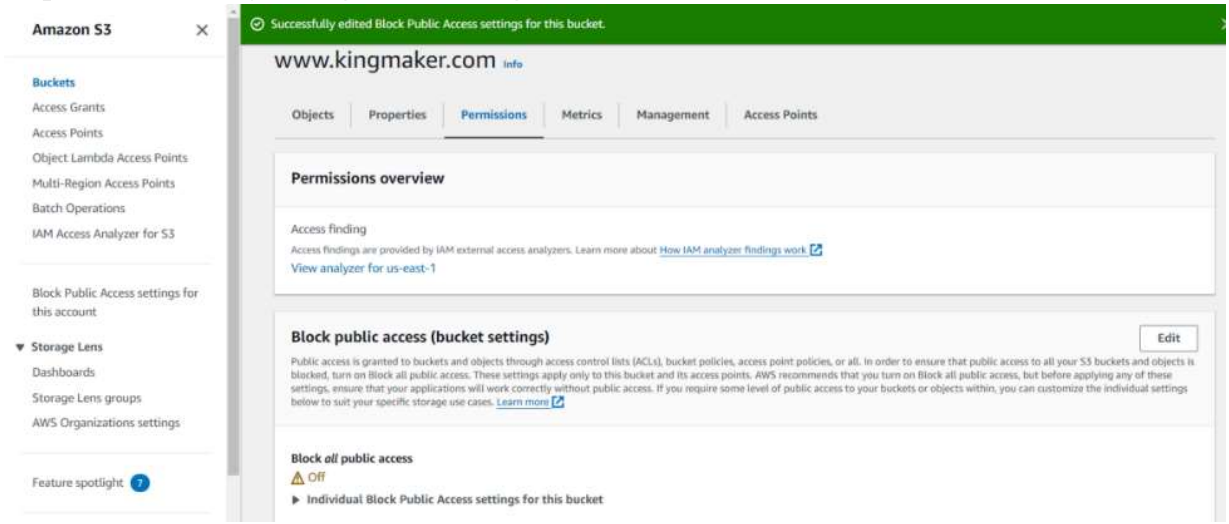
Step12: Open the link. It will show a 403 forbidden error screen as the contents of the bucket

are not available for the public users. To change this, go to Permissions tab, go to Block public access and click on edit

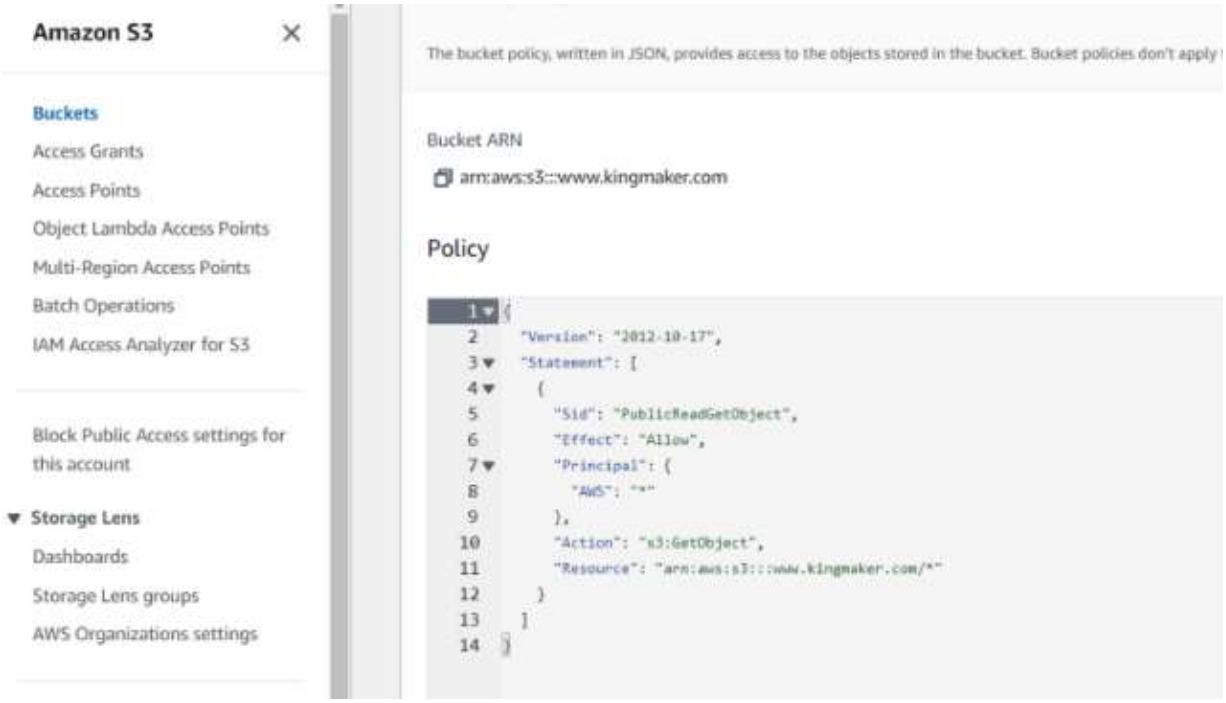
Step 13: Uncheck the Block all public access checkbox and click on save changes



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

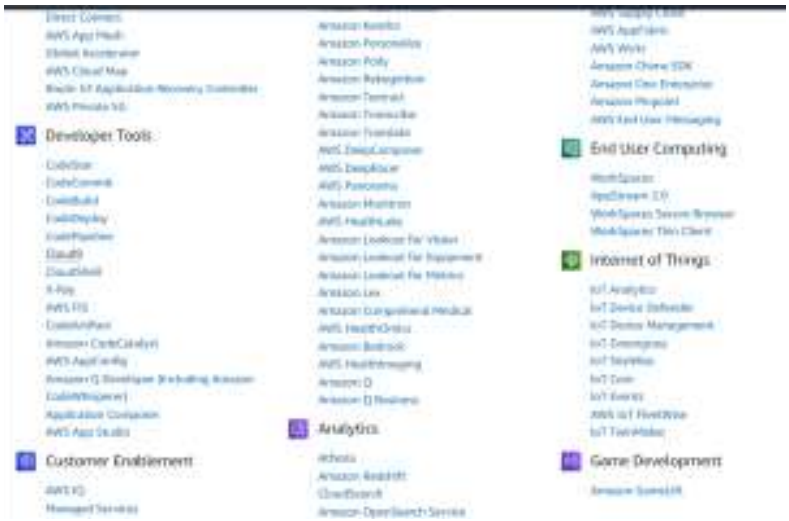


Step 16: Now reload the website. You can see your website

Experiment No: 1(B)

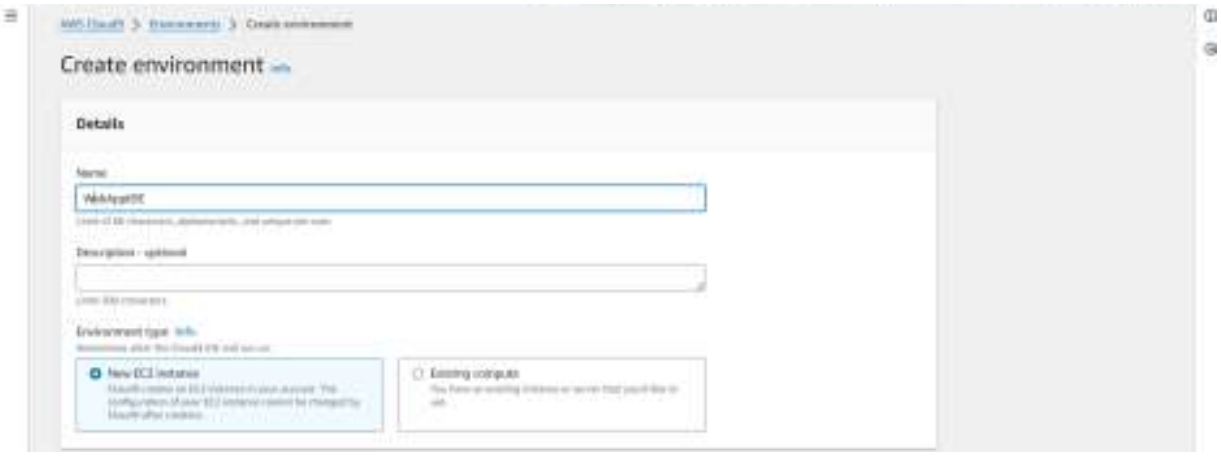
Step 1: Set up Cloud9 environment.

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



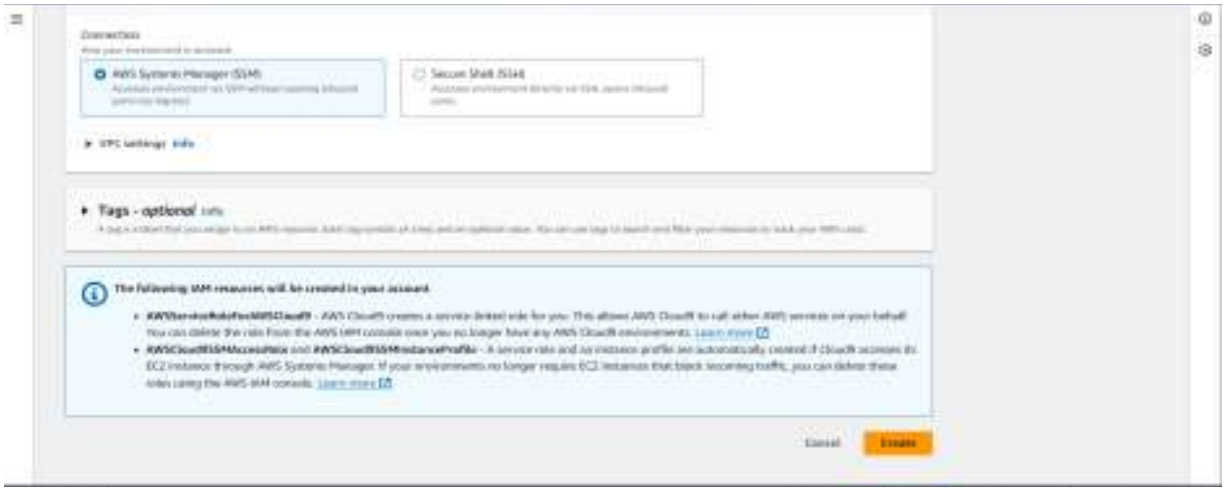
2) Click on create environment

3) Give the name to your Environment ,keeping the other settings as default



4) Select the correct platform type as shown below and keep the others details as default

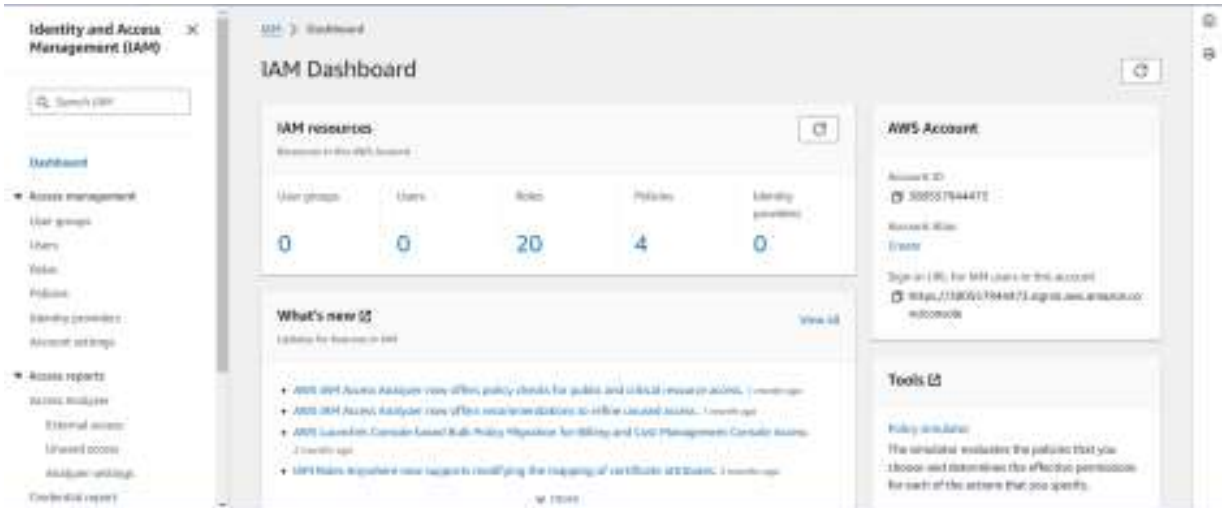
5) Click on SSH under connection type in network settings and click on Create



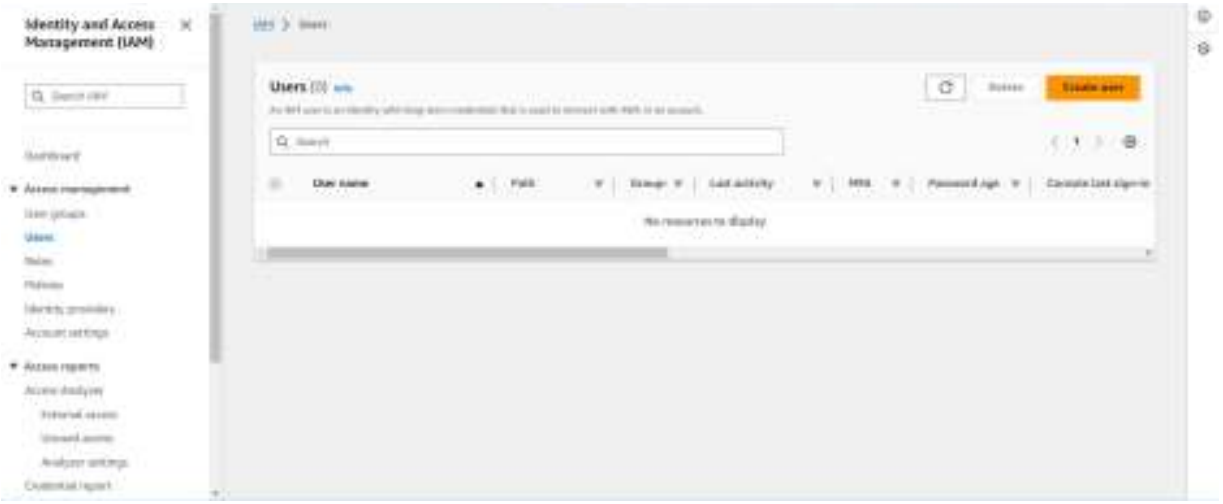
6) Successfully created the environment so now click on open

Step 2: Creating IAM user.

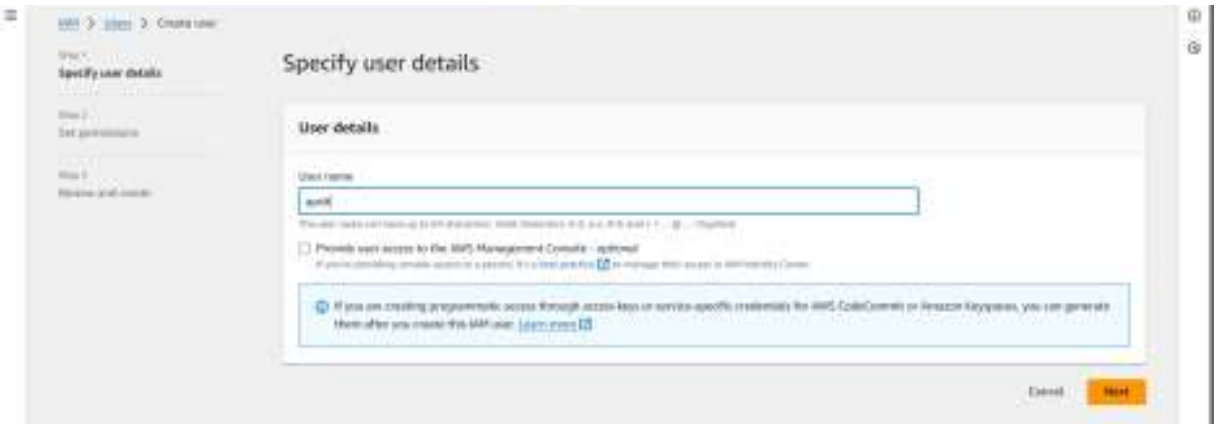
1) Search IAM on the services search bar and open it. Click on Create User



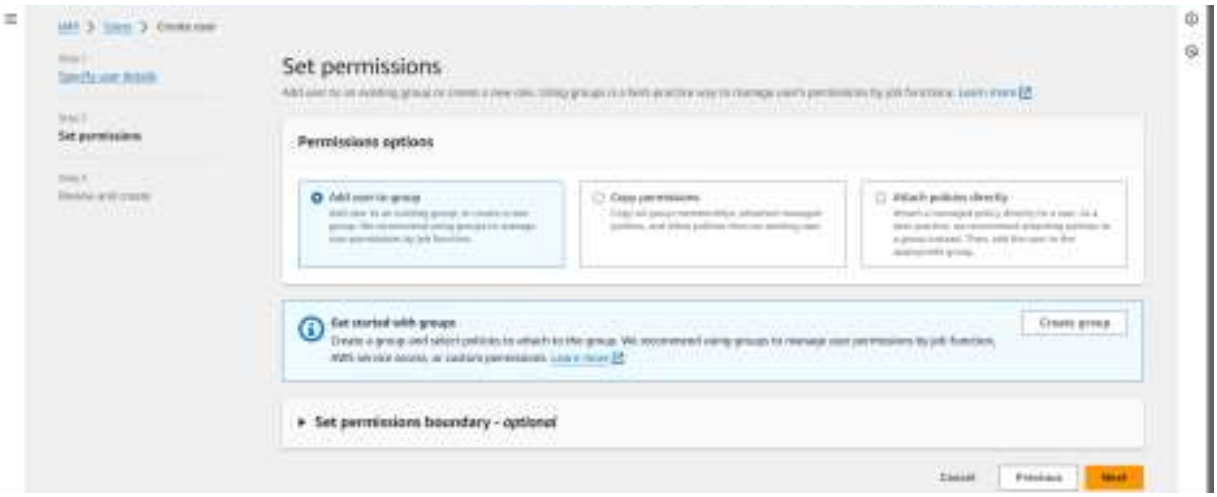
2) Click on the create user



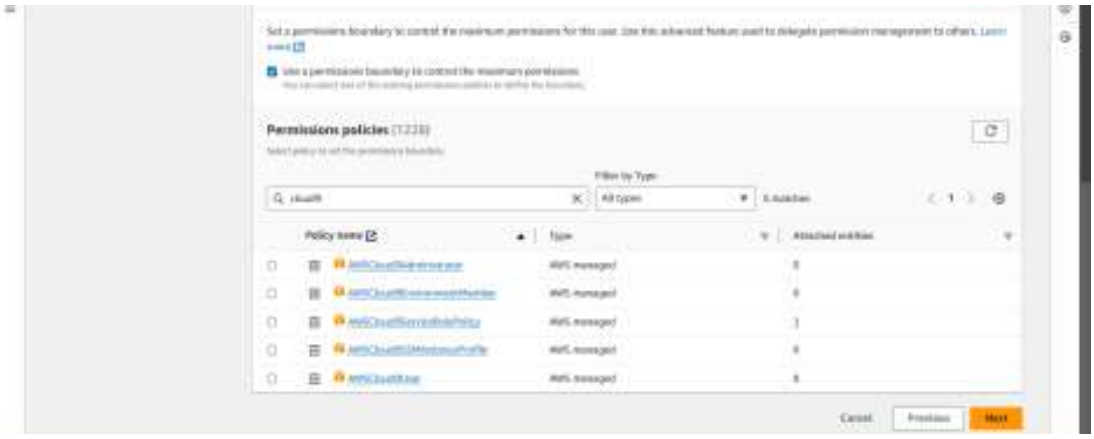
3) Write the name of the user you want to add and click on next



4) 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

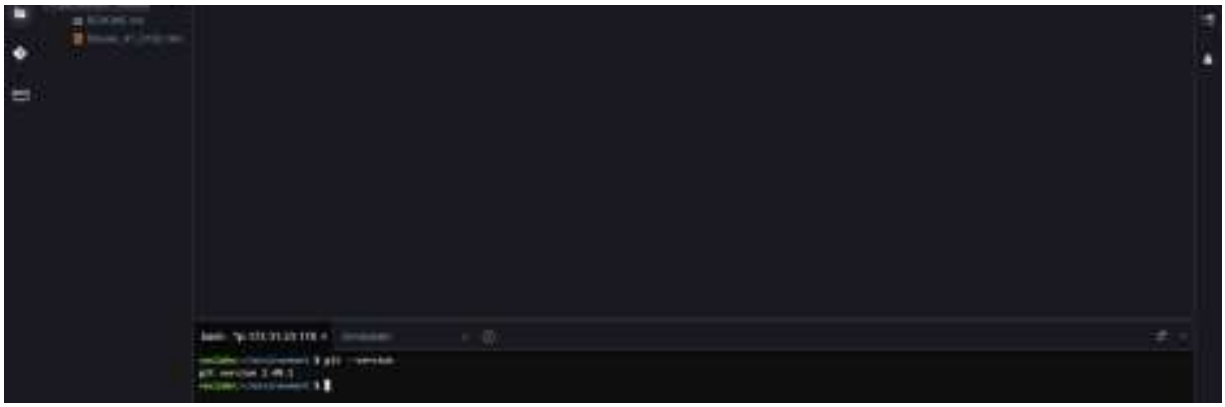


6) Scroll down and click on create user

Step 3: Working on Cloud9 IDE

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

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.

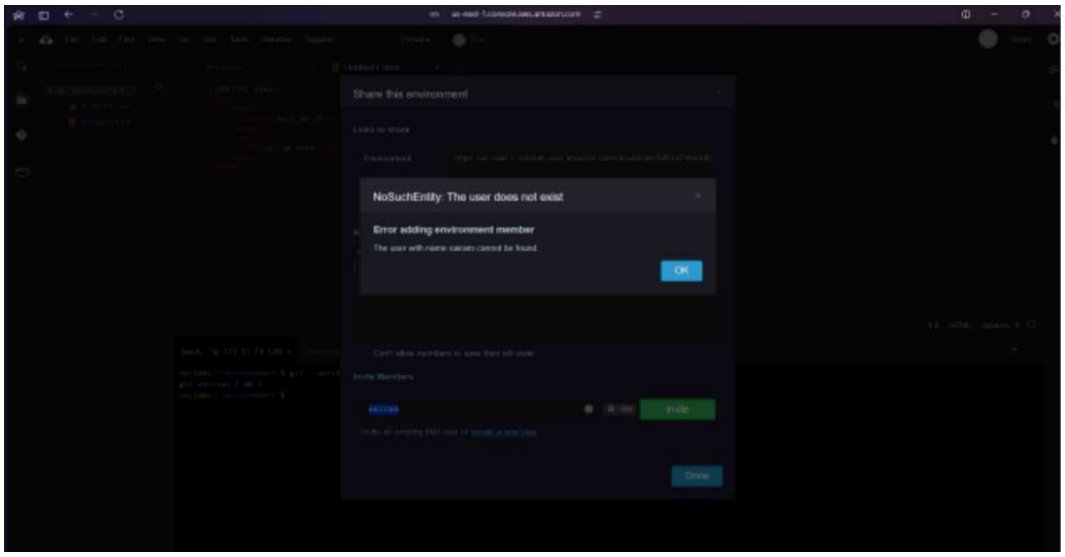


3) To add a file, click on file. For this experiment, we are to add an HTML file. So go to File → New From Template → HTML file. This gives a basic HTML template on the coding IDE

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

.

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

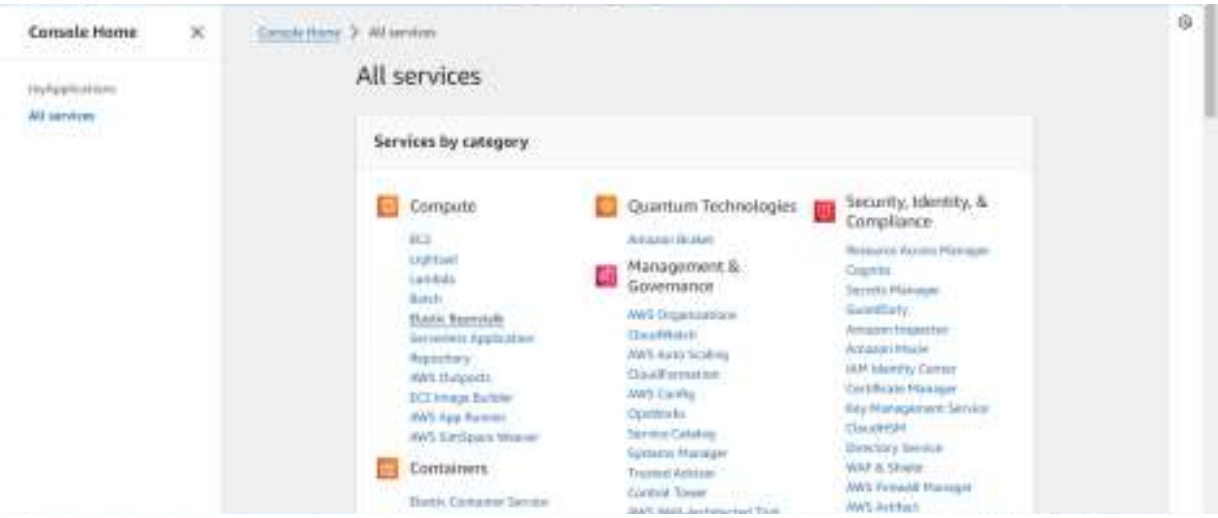


Here, it gives an error as Cloud9 was created on the academy account where creating an IAM group is not available, meanwhile on the personal account, the services of Cloud9 have been
deprecated. So currently, it is not possible to integrate the cloud9 and IAM parts of the experiment.

Name : Sandeshkumar.M.Yadav Div : D15C Roll : 61

Experiment No :2

Step 1: Login to your AWS console. Search for Elastic Beanstalk in the searchbar near services.

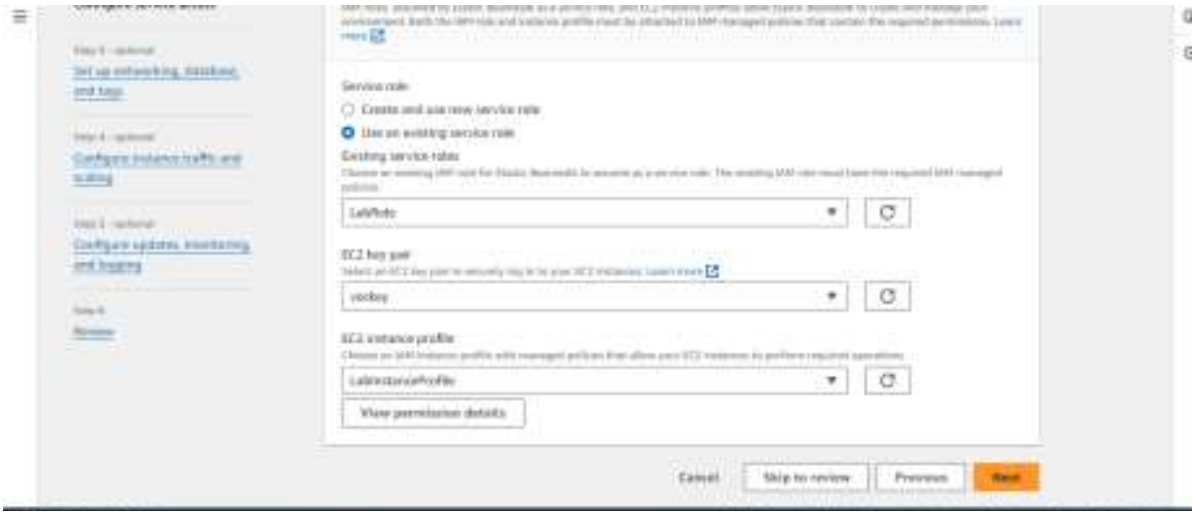


Step 2: Go to Elastic Beanstalk and click on Create Application

Step 3: Enter the name of your application. Scroll down and in the platform, select platform as PHP. Keep the application code as Sample Application. Set the instance to single instance. Click on NEXT.

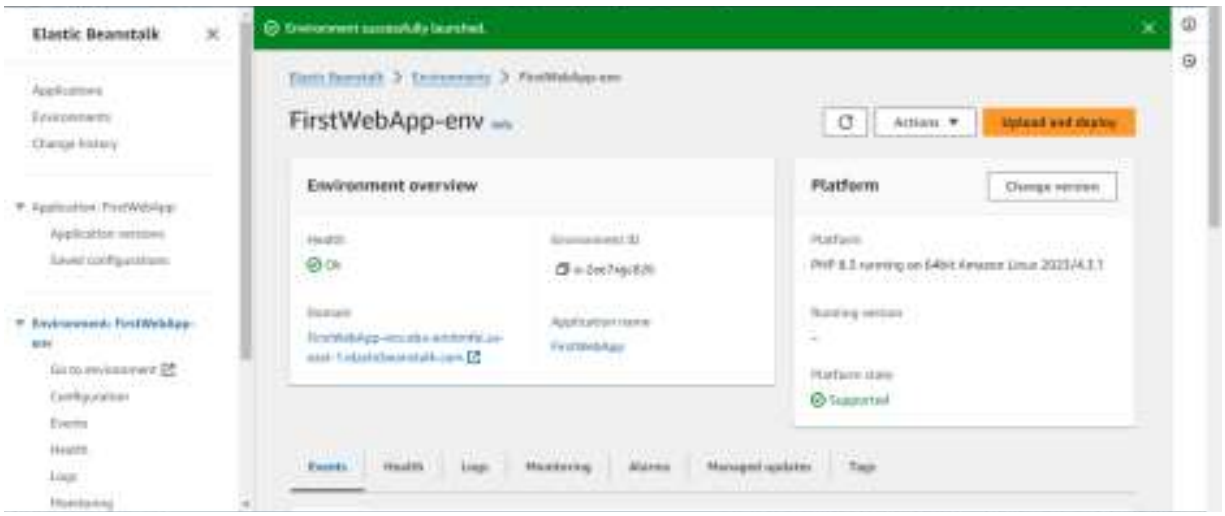
Use an existing service role and choose whatever service role is present on your account

Step 5: Click on Skip to Review



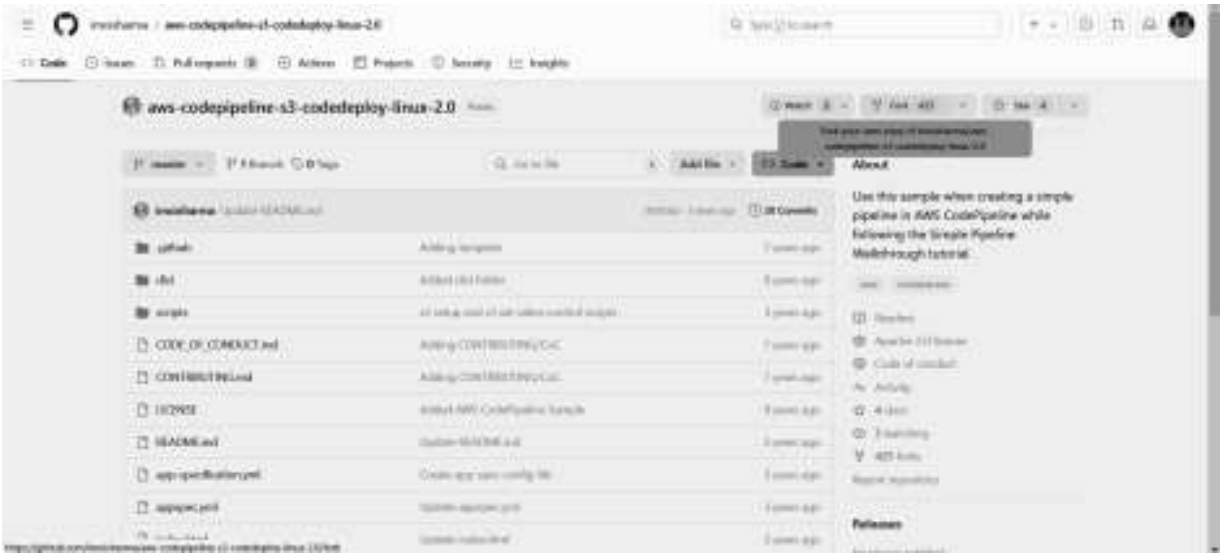
Name : Sandeshkumar.M.Yadav Div : D15C Roll : 61

Review the settings that you have set up for your application and submit your application

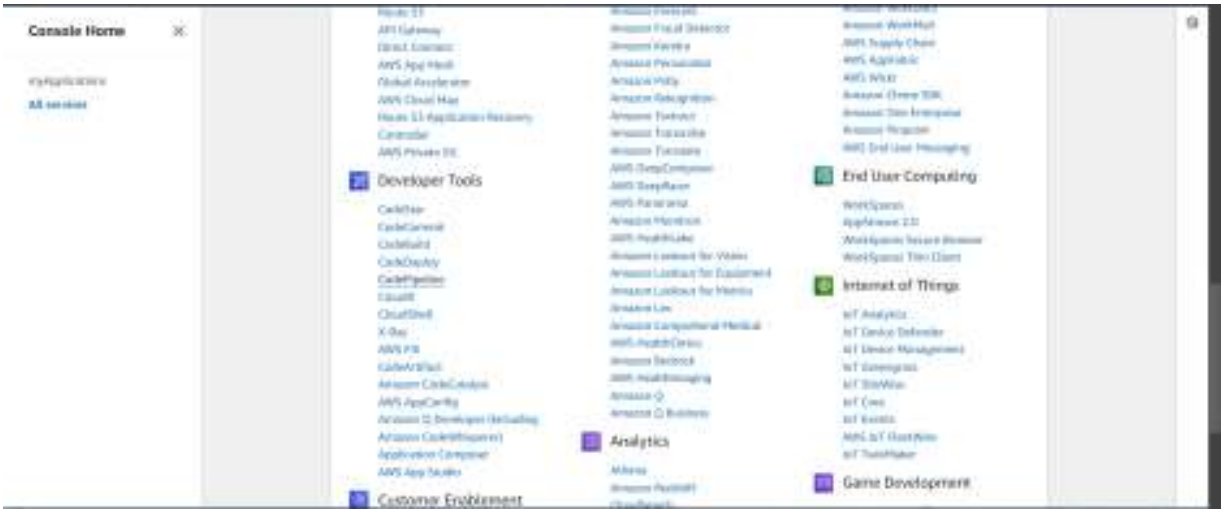


Step 7 : Go to the github link below. This is a github with a sample code for deploying a file on AWS CodePipeline. Fork this repository into your personal github.

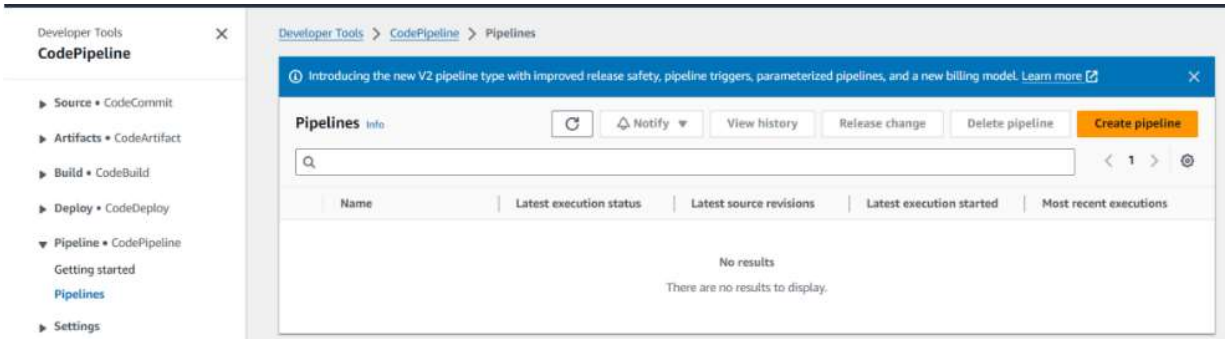
https://github.com/aws-samples/aws-codepipeline-s3-codedeploy-linux



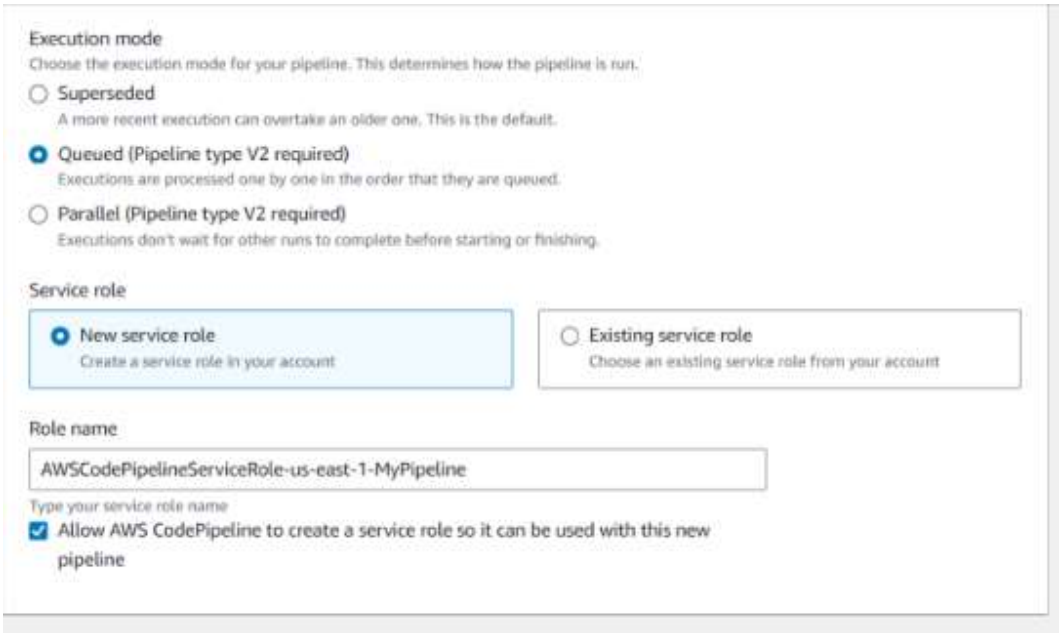
Step 8: Search CodePipeline in the services tab and click on it.



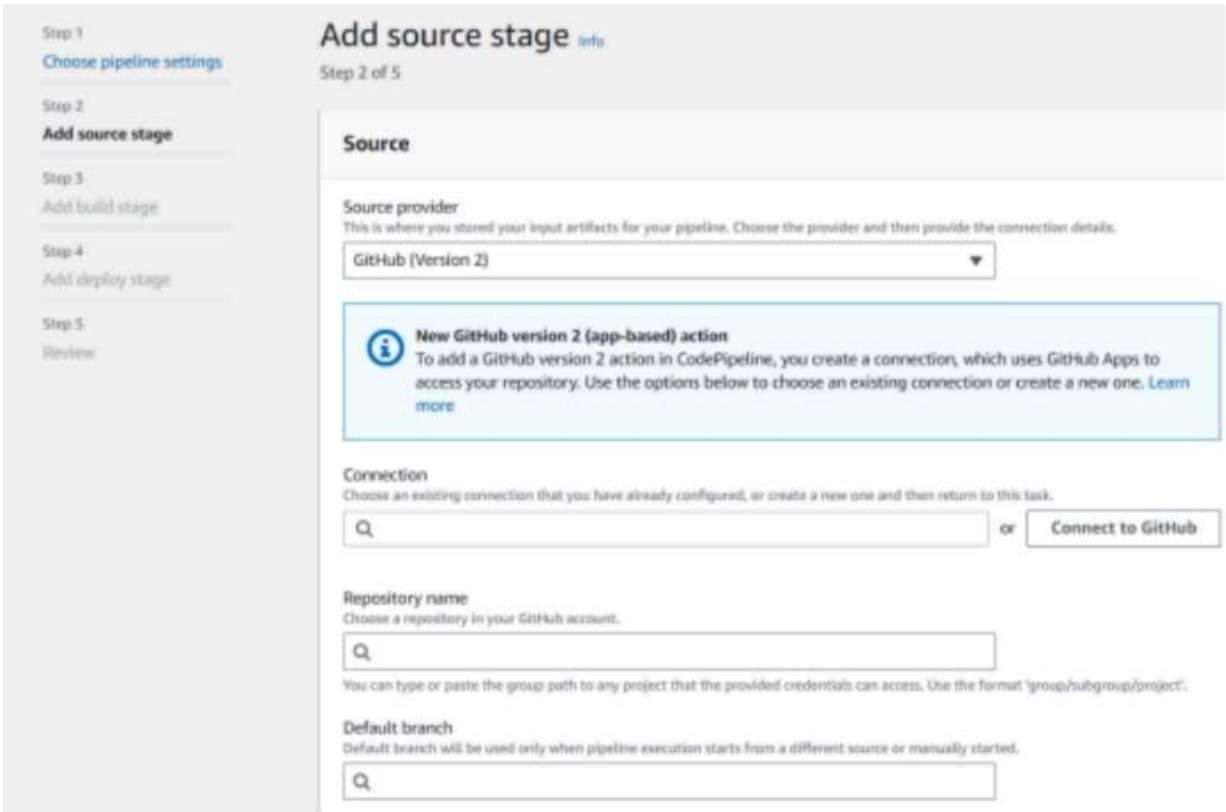
Step 9: Click on Create Pipeline.



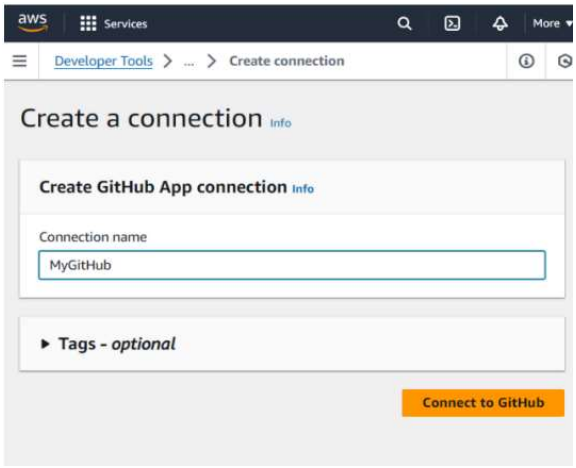
Step 10: Give a name to your Pipeline. A new service role would be created with the name of the pipeline.

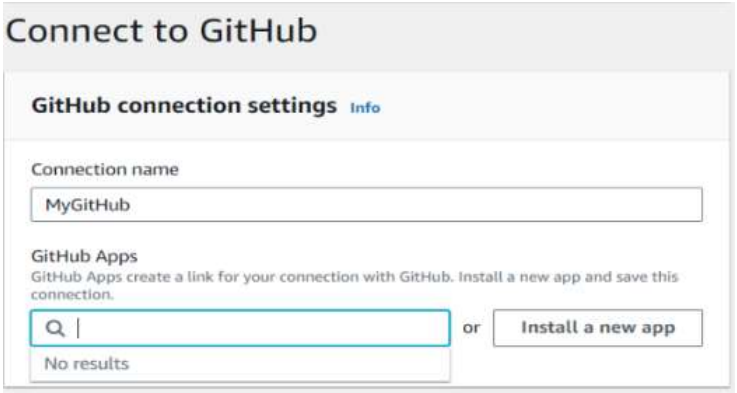


Step 11: Select a source provider (as GitHub Version (2)). Click on connect to Github

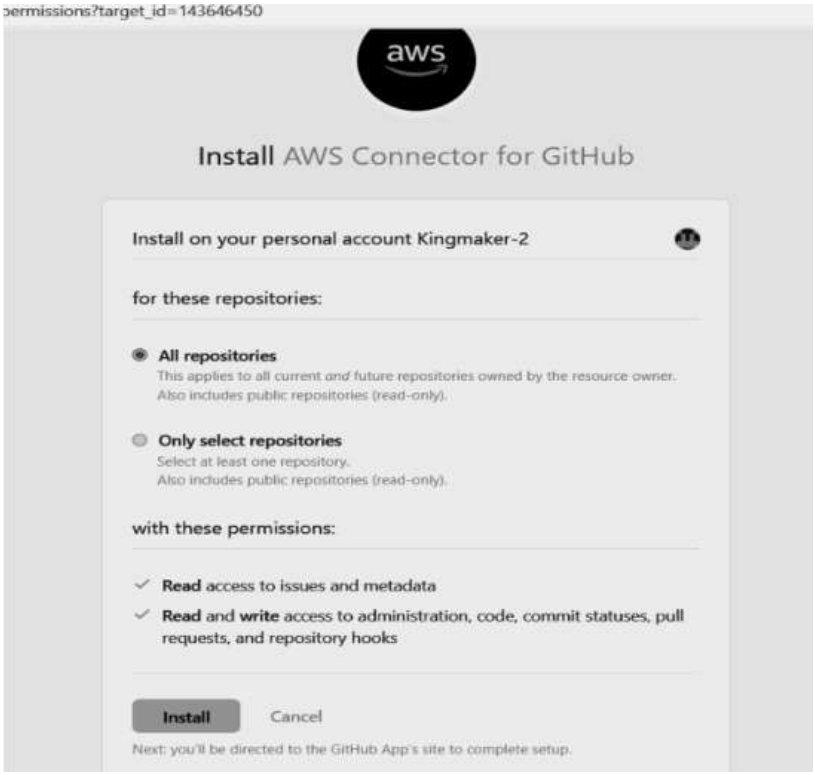


Step 12: Give a name to your GitHub app Connection and click on Connect. This will give you a prompt to either to select a GitHub app or to install a new app. If it is your first time, click on Install a new app.

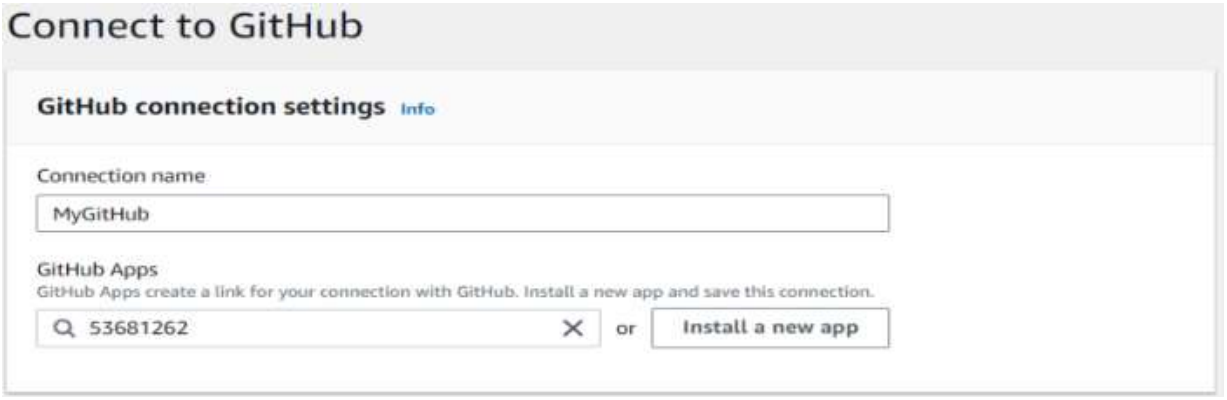


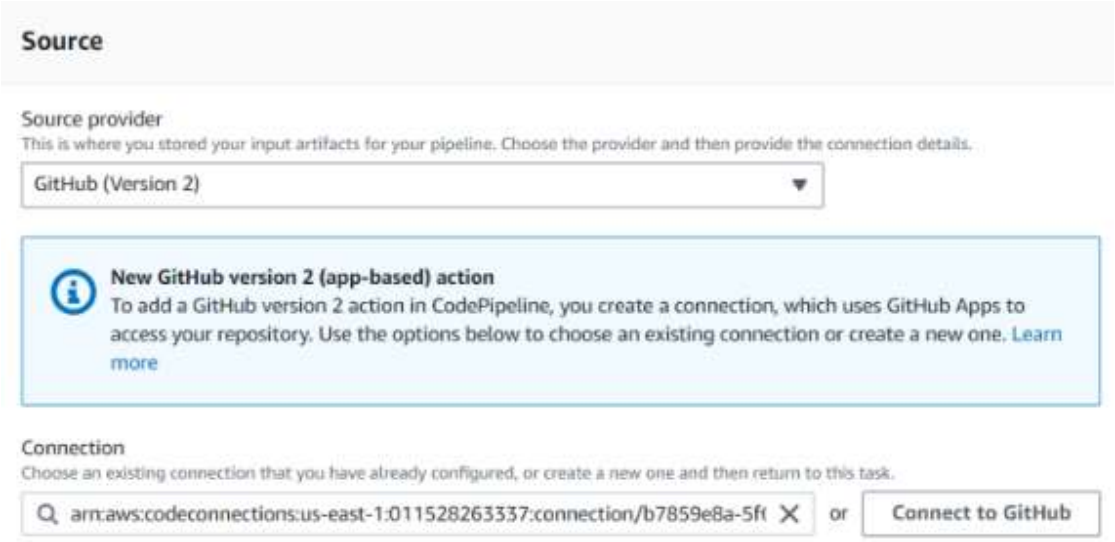


Step 13: This will direct you to install AWS connector on your GitHUb .Install it to your account and give it its permissions

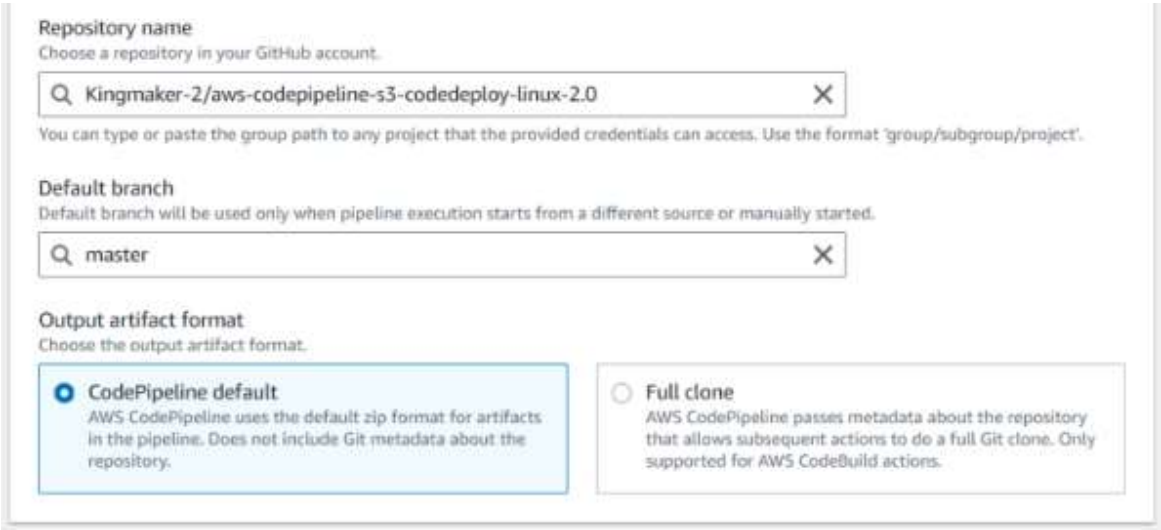


Step 14: After the app is set up, it gives the number in the text field. Click on Connect. After clicking on connect, the link is shown in the connection field and AWS shows that GitHub connection is ready to use.



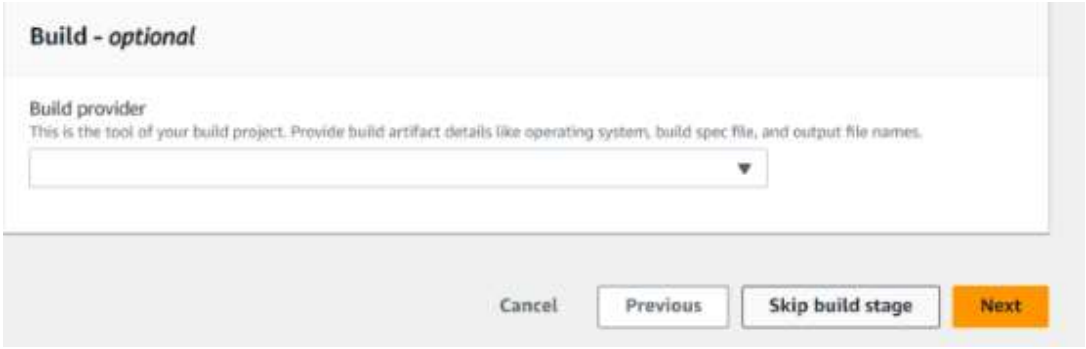


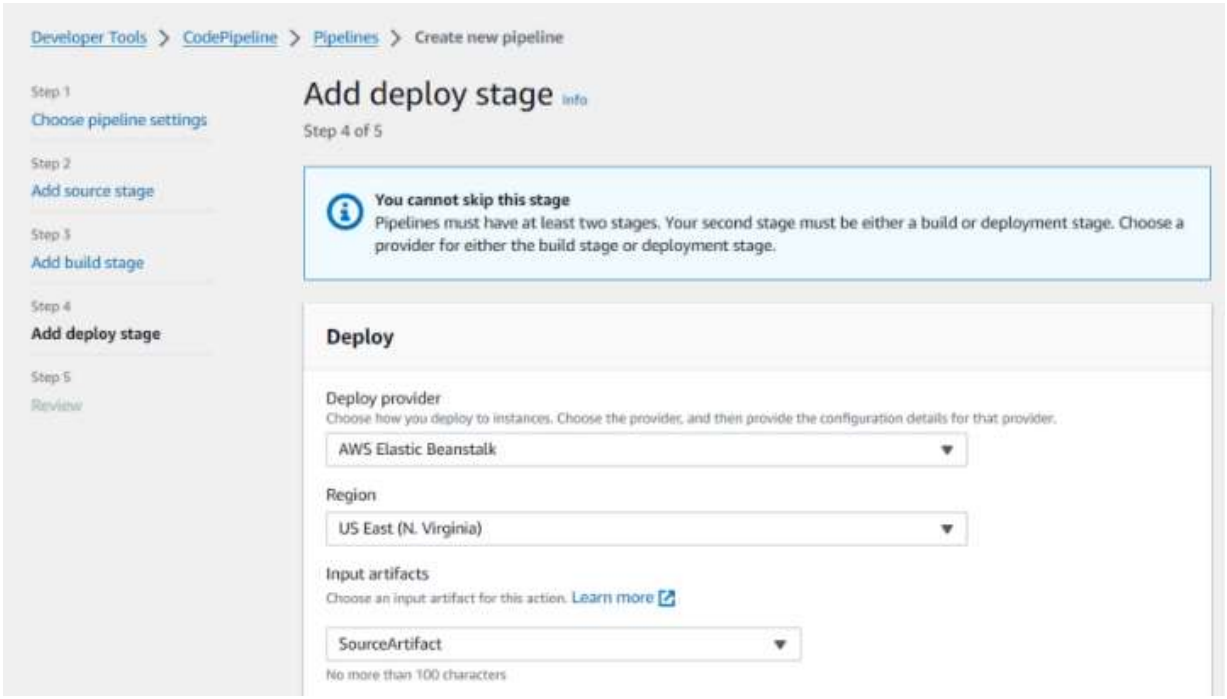
Step 15: Select the repository that you had forked to your GitHub. After that select the branch on which the files are present (default is Master).

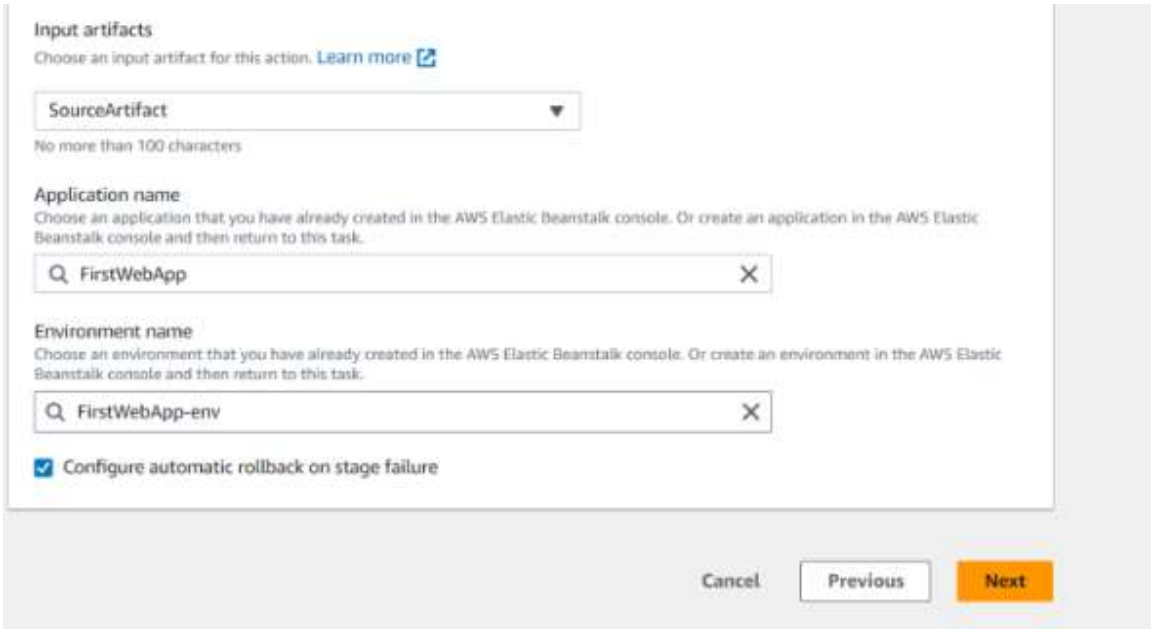


Step 16: Set the Trigger type as no filter. This would allow it to the website to update as soon as some change is made in the github.

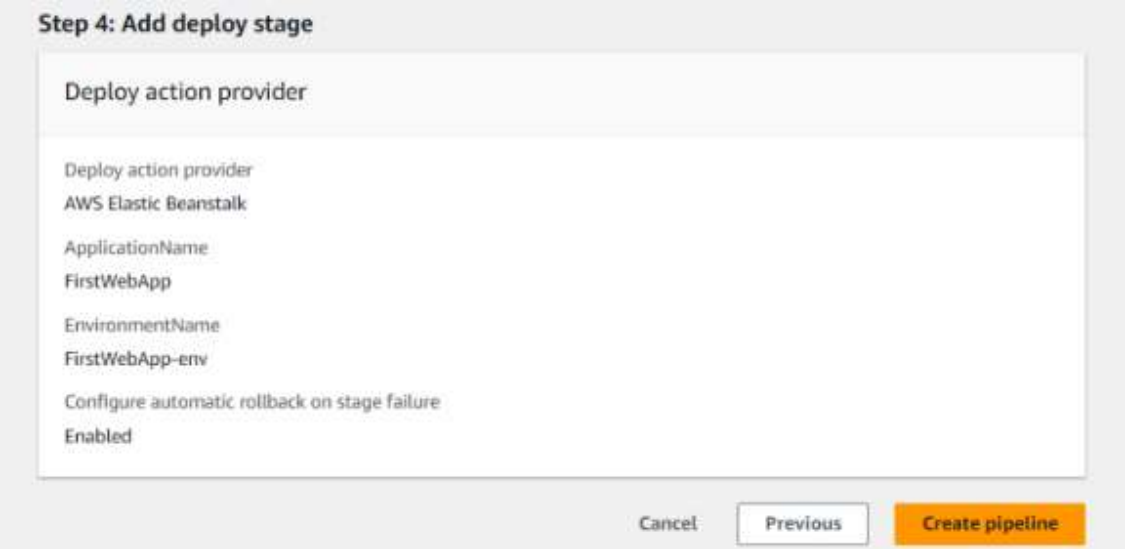
Step 17: Skip the build stage and go to Deploy. Select the deploy provider as AWS Elastic Beanstalk and Input Artifact as SourceArtifact. The application name would be the name of your Elastic Beanstalk. Then click on next.



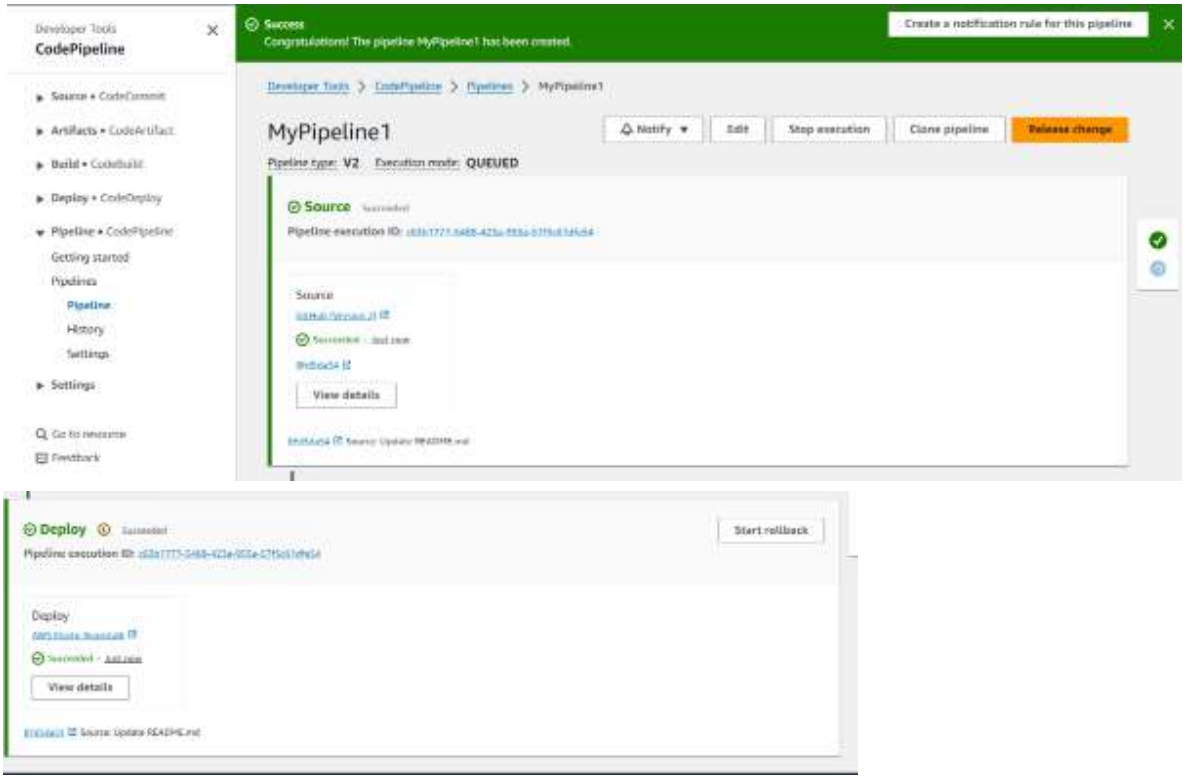

Developer Tools > CodePipeline > Pipelines > Create new pipeline
Step 1
Choose pipeline settings
Step 2
Add source stage
Step 3
Add build stage
Step 4
Add deploy stage
Step 5
Review
Add deploy stage info
Step 4 of 5
You cannot skip this stage
Pipelines must have at least two stages. Your second stage must be either a build or deployment stage. Choose a provider for either the build stage or deployment stage.
Deploy
Deploy provider
Choose how you deploy to instances. Choose the provider, and then provide the configuration details for that provider.
AWS Elastic Beanstalk
Region
US East (N. Virginia)
Input artifacts
Choose an input artifact for this action. Learn more
SourceArtifact
No more than 100 characters


Input artifacts
Choose an input artifact for this action. Learn more
SourceArtifact
No more than 100 characters
Application name
Choose an application that you have already created in the AWS Elastic Beanstalk console. Or create an application in the AWS Elastic Beanstalk console and then return to this task.
FirstWebApp
Environment name
Choose an environment that you have already created in the AWS Elastic Beanstalk console. Or create an environment in the AWS Elastic Beanstalk console and then return to this task.
FirstWebApp-env
Configure automatic rollback on stage failure
Cancel
Previous
Next

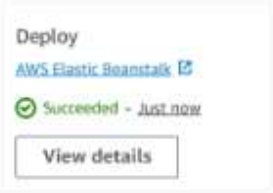
Step 18: Check all the information and click on create Pipeline



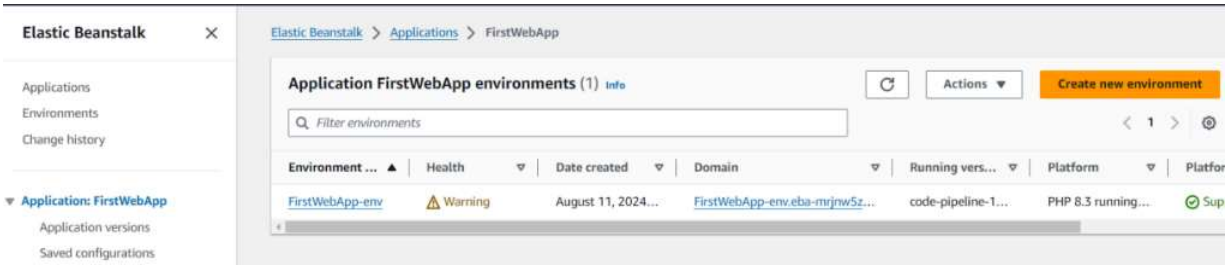
Step 19: If the pipeline is successfully deployed, this screen comes up where the source is set up and then it is transitioned to deploy



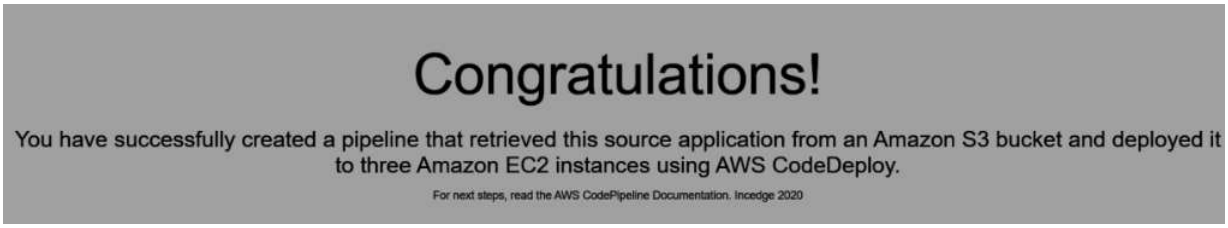
Step 20: Once the deployment is complete, click on the AWS Elastic Beanstalk under Deploy.



Step 21: This will redirect you to the application screen of Elastic Beanstalk. Click on the link shown under Domain



Step 22: This will successfully show the sample website hosted.



Name : Sandeshkumar.M.Yadav **Div : D15C** **Roll : 61**

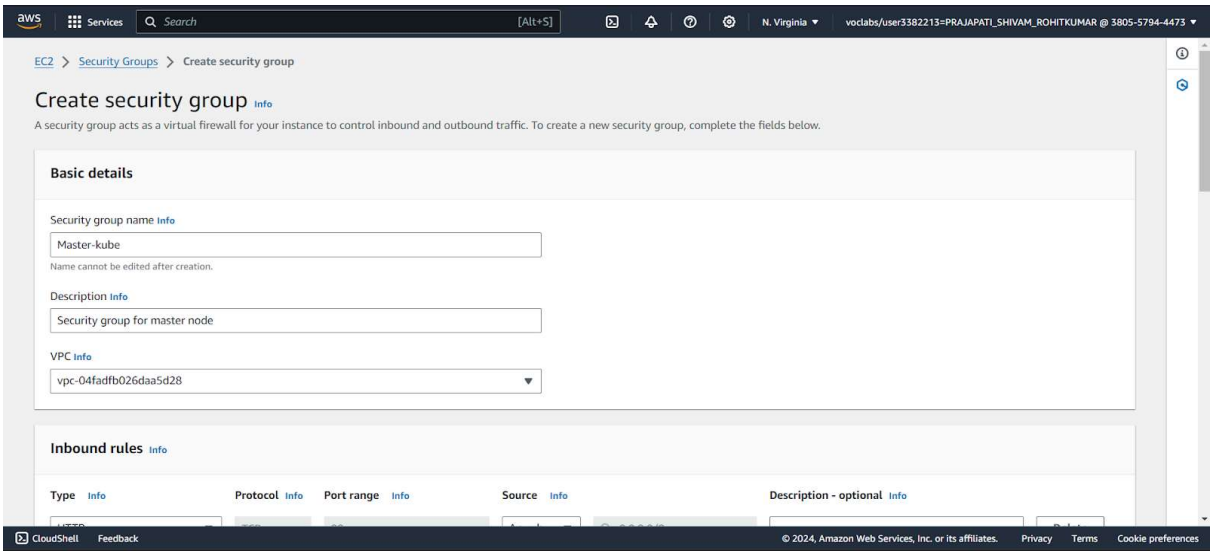
Experiment No : 3

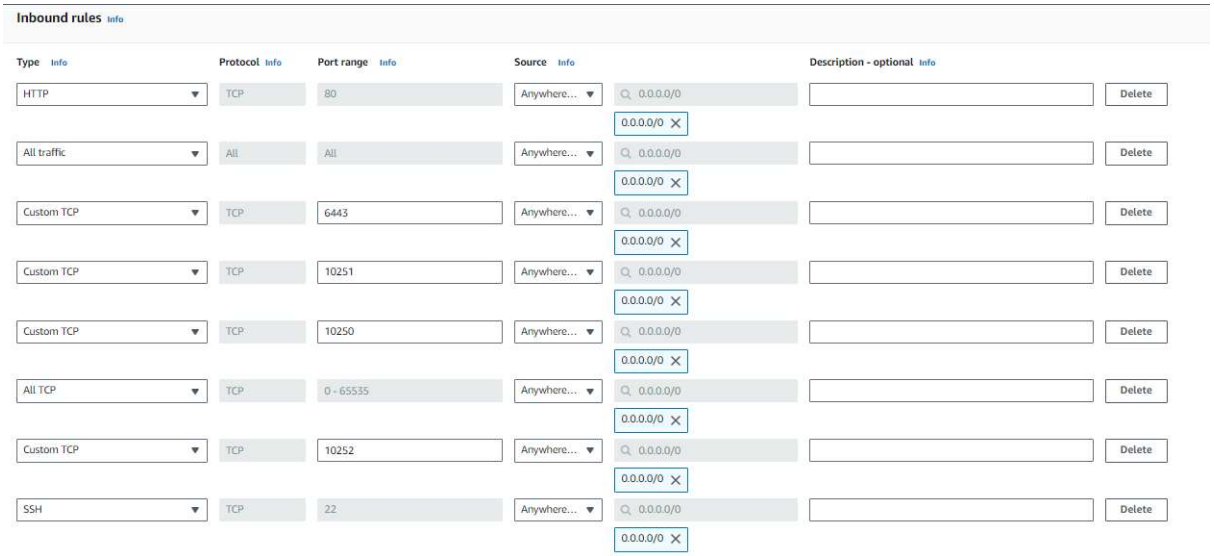
AIM: To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernete Cluster on Linux Machines/Cloud Platforms.

PREREQUISITES:

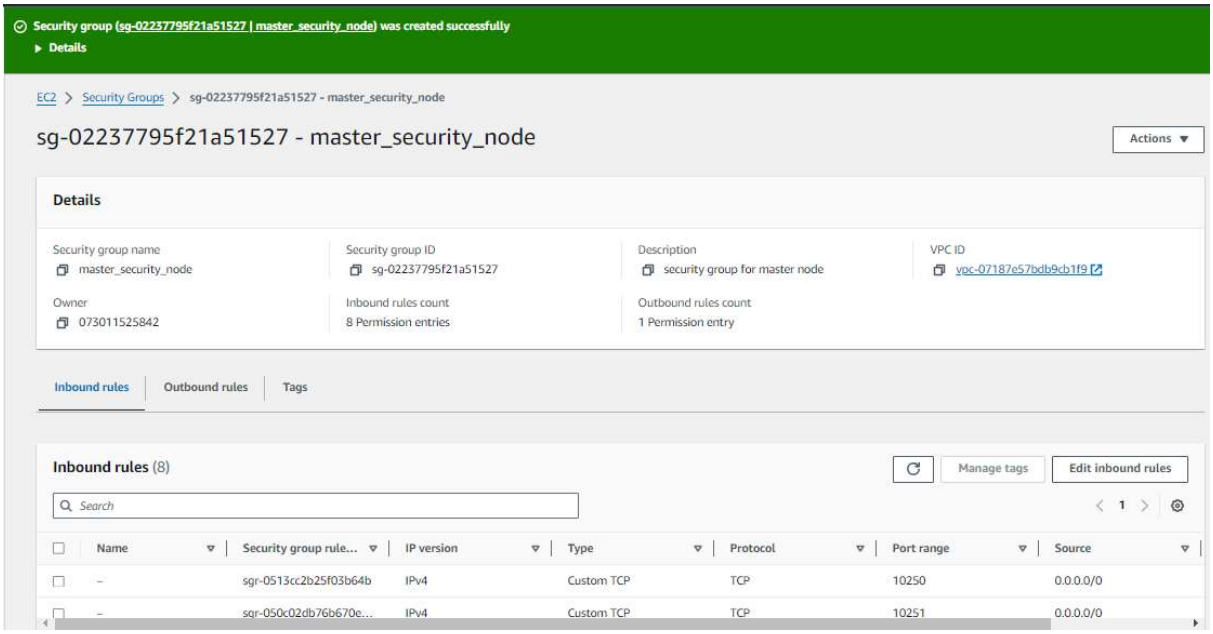
Create 2 Security Groups for Master Node and Worker Nodes and add the following inbound rules in those Groups.

Master Node Security Group:

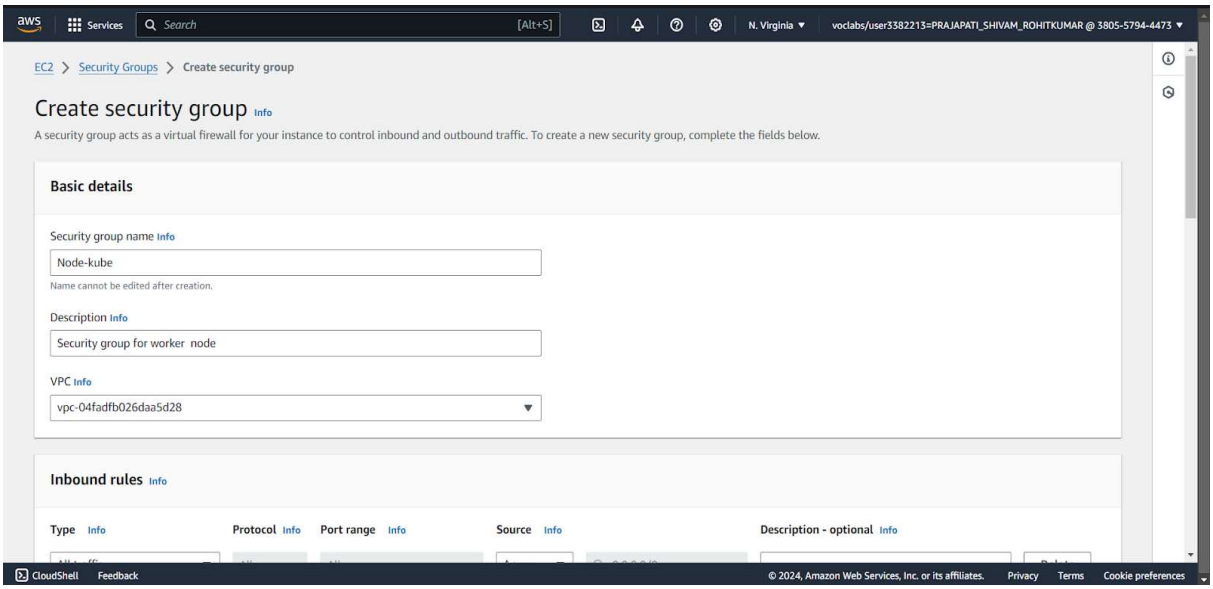


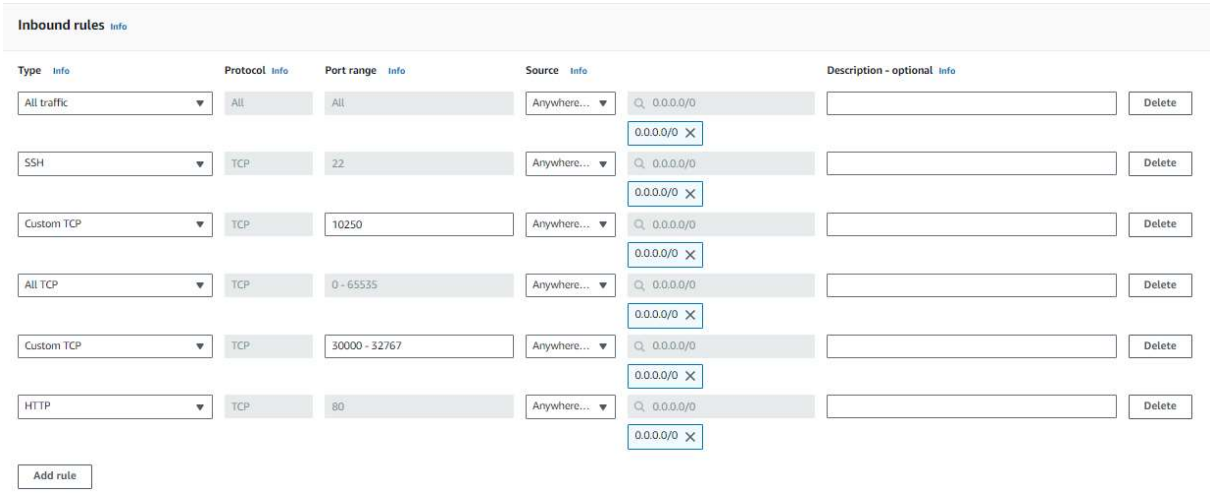


Name : Sandeshkumar.M.Yadav **Div : D15C** **Roll : 61**

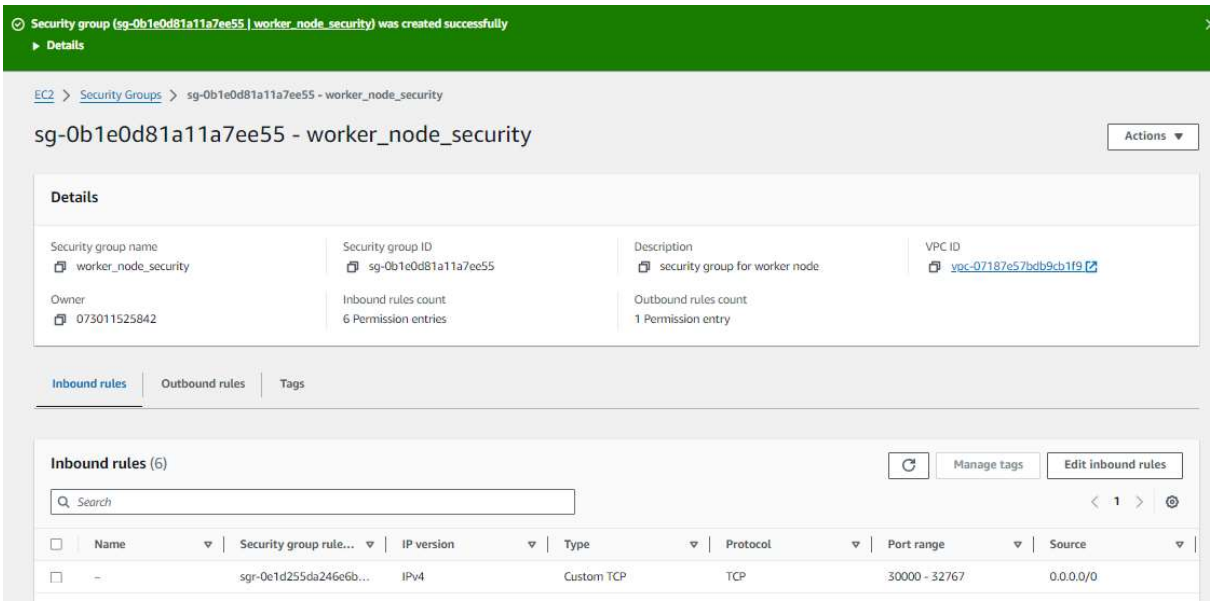


Worker Node Security Group :





Added Required Inbound rules for worker nodes

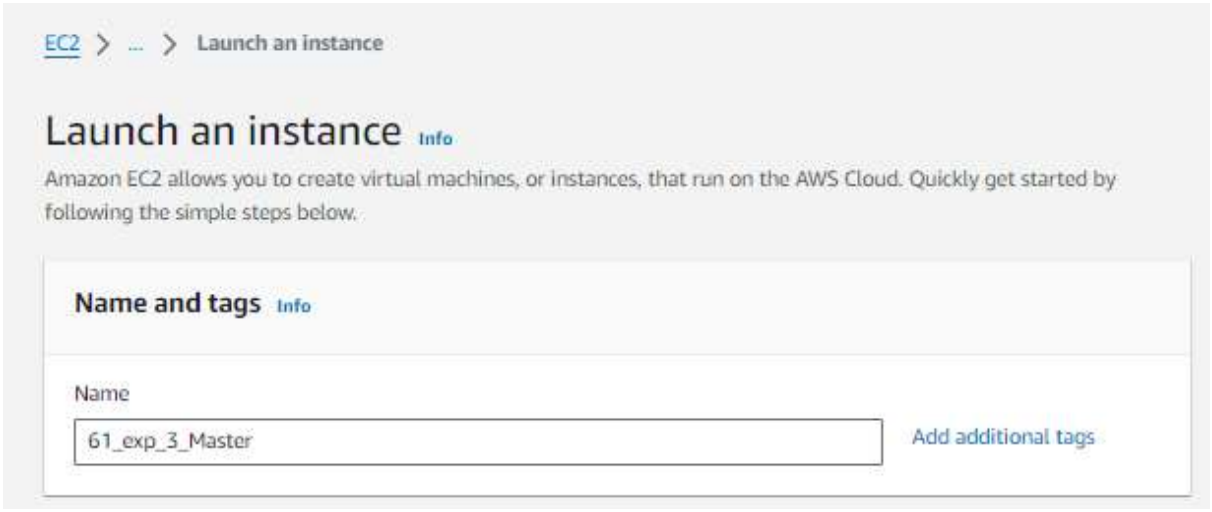


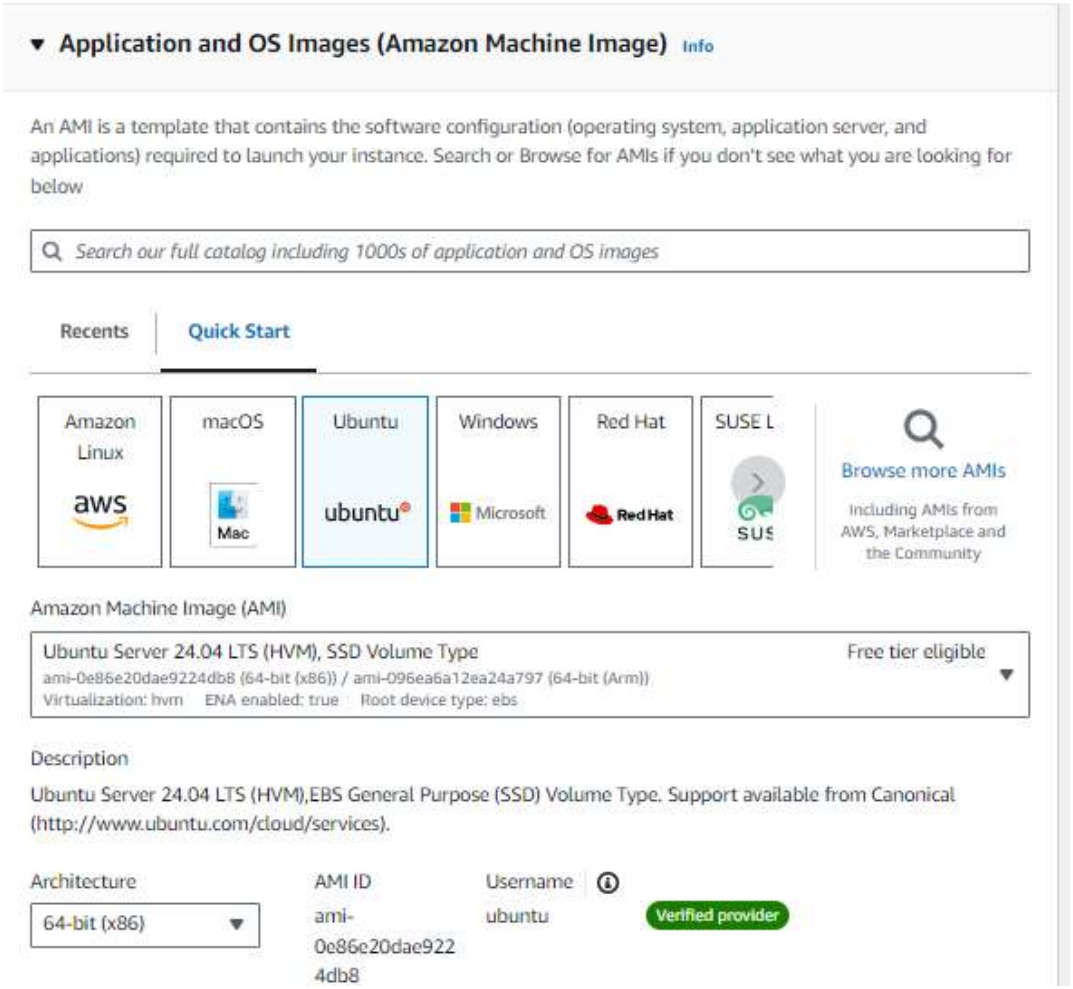
Step 1: Access your AWS Academy or personal account and create **three new EC2 instances**. For the AMI, choose **Ubuntu** and set the **instance type to t2.medium**. Generate an RSA key in .pem format and move the downloaded key to a new folder for **eg: Newfolder**; you can either use three separate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

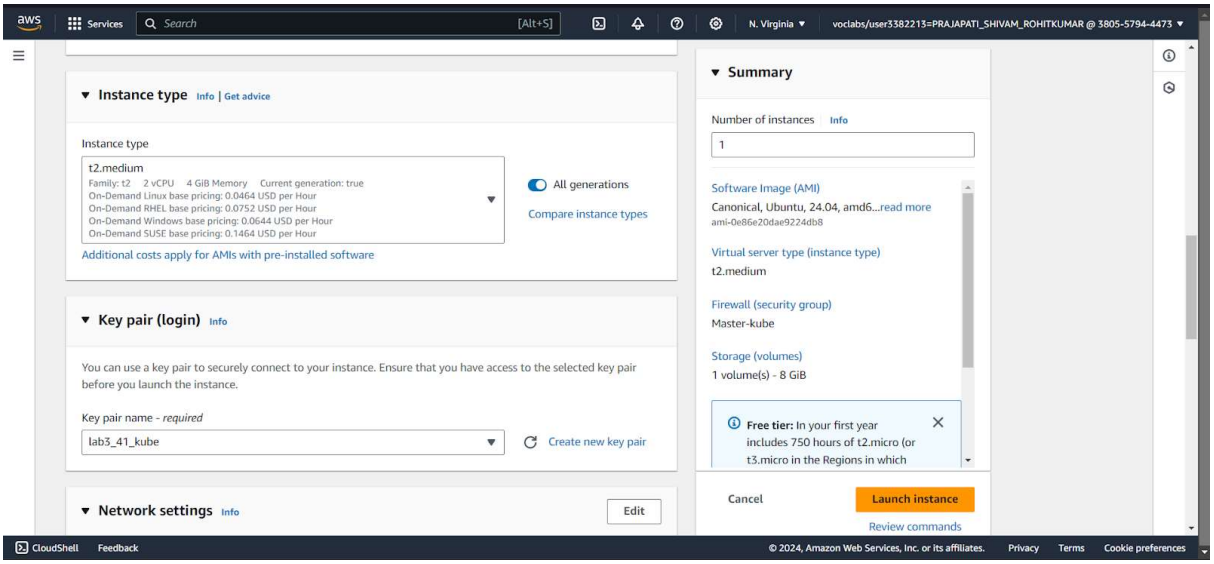
Master Node:



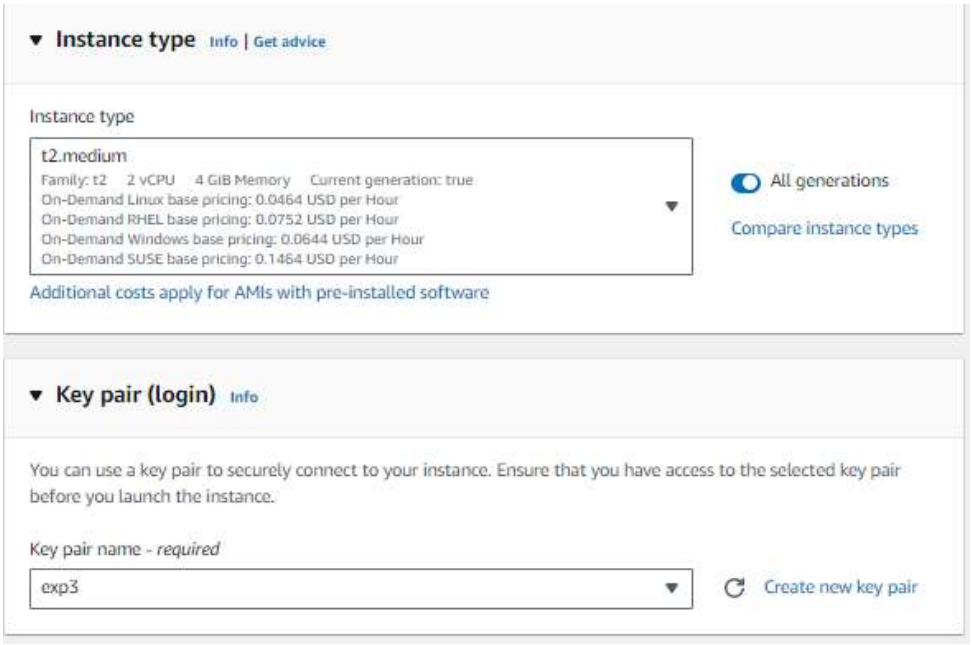


AMI as Ubuntu

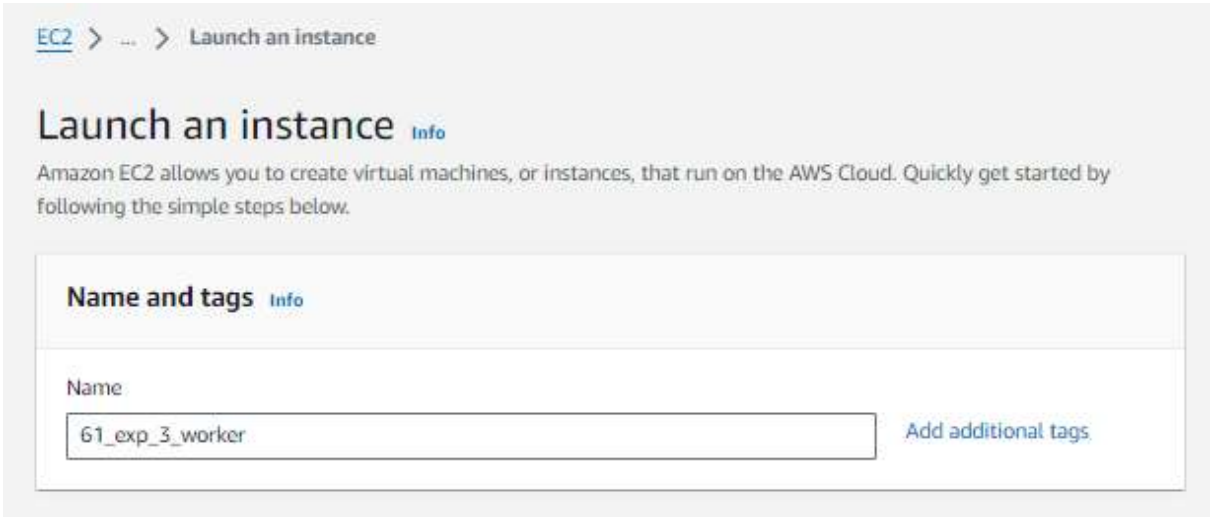
Name : Sandeshkumar.M.Yadav **Div : D15C** **Roll : 61**



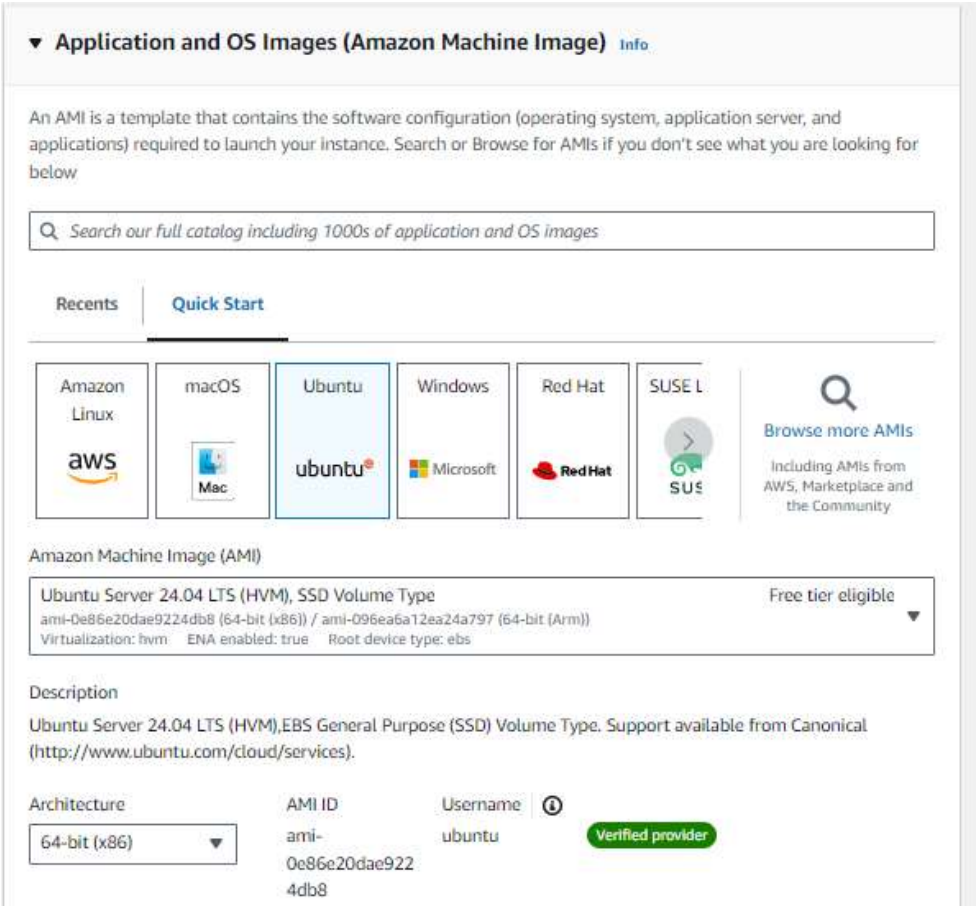
Generate Instance type and create key pair login



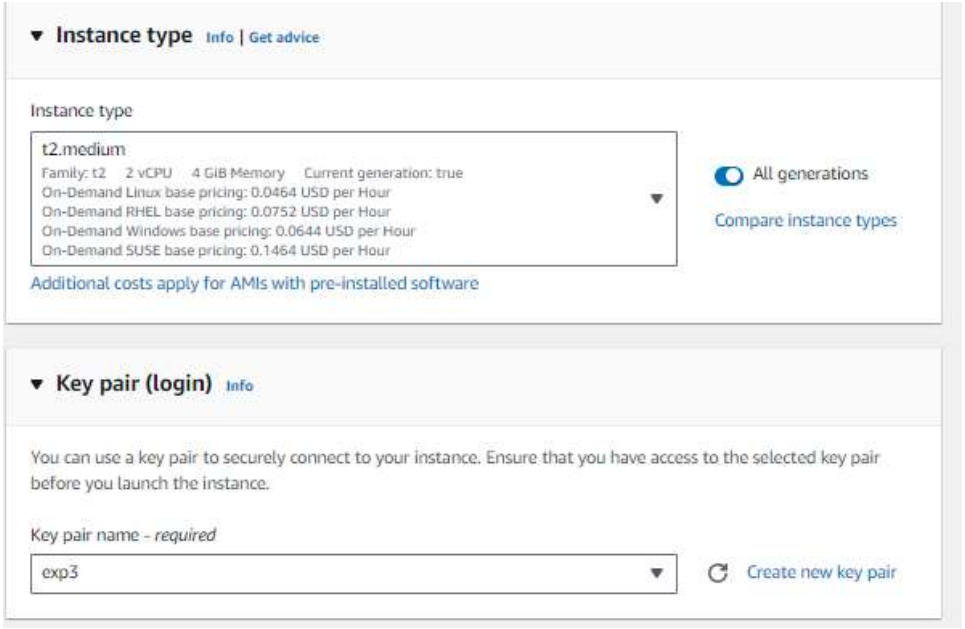
Worker Node 1:



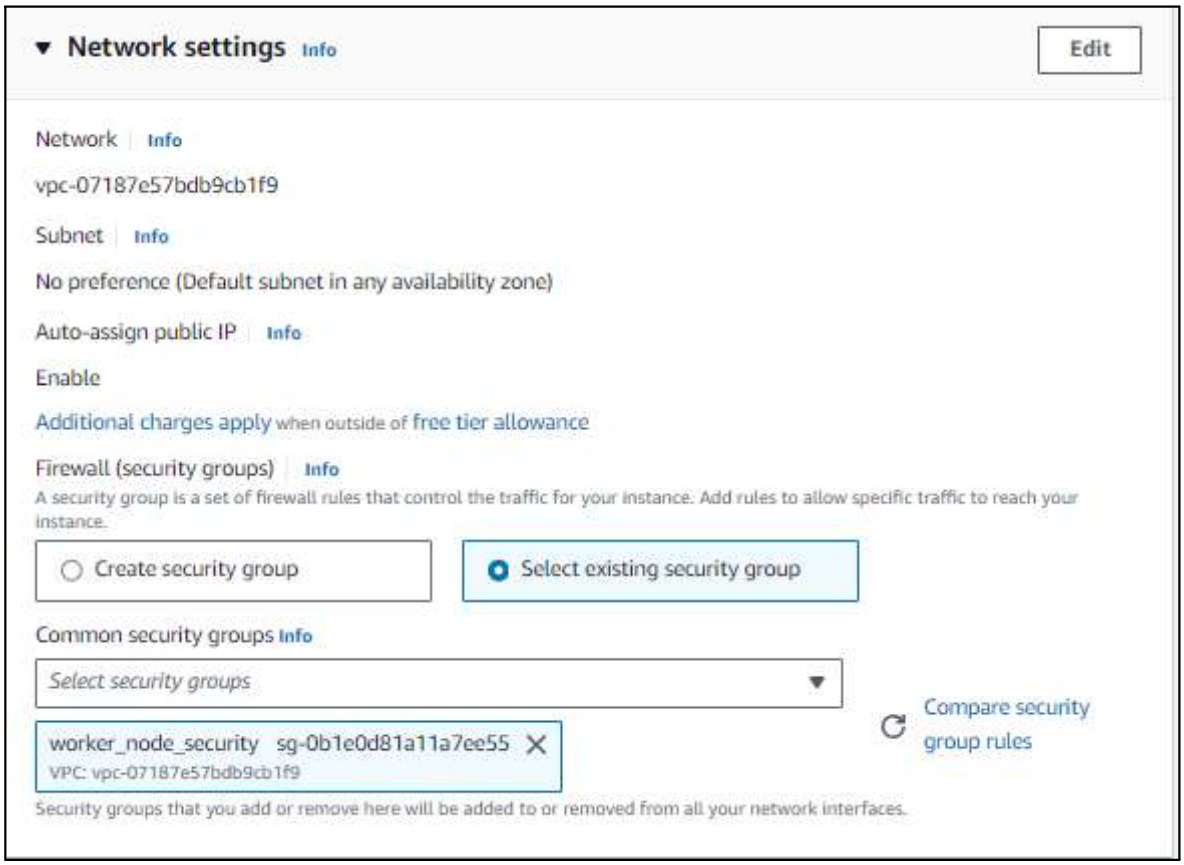
Give a name to your instance .



AMI as Ubuntu



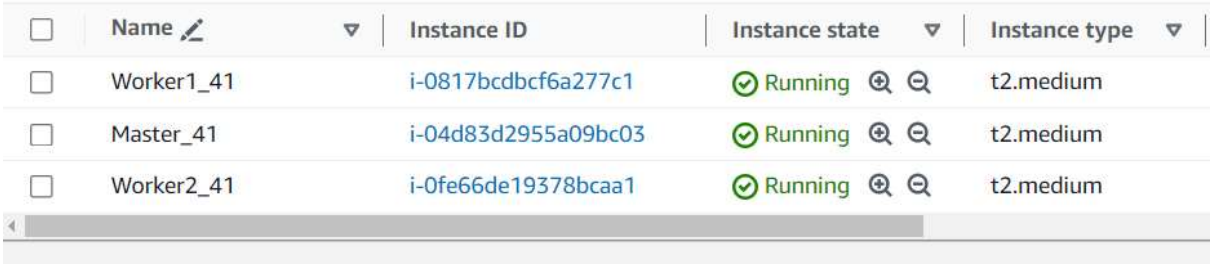
Created the Key pair login



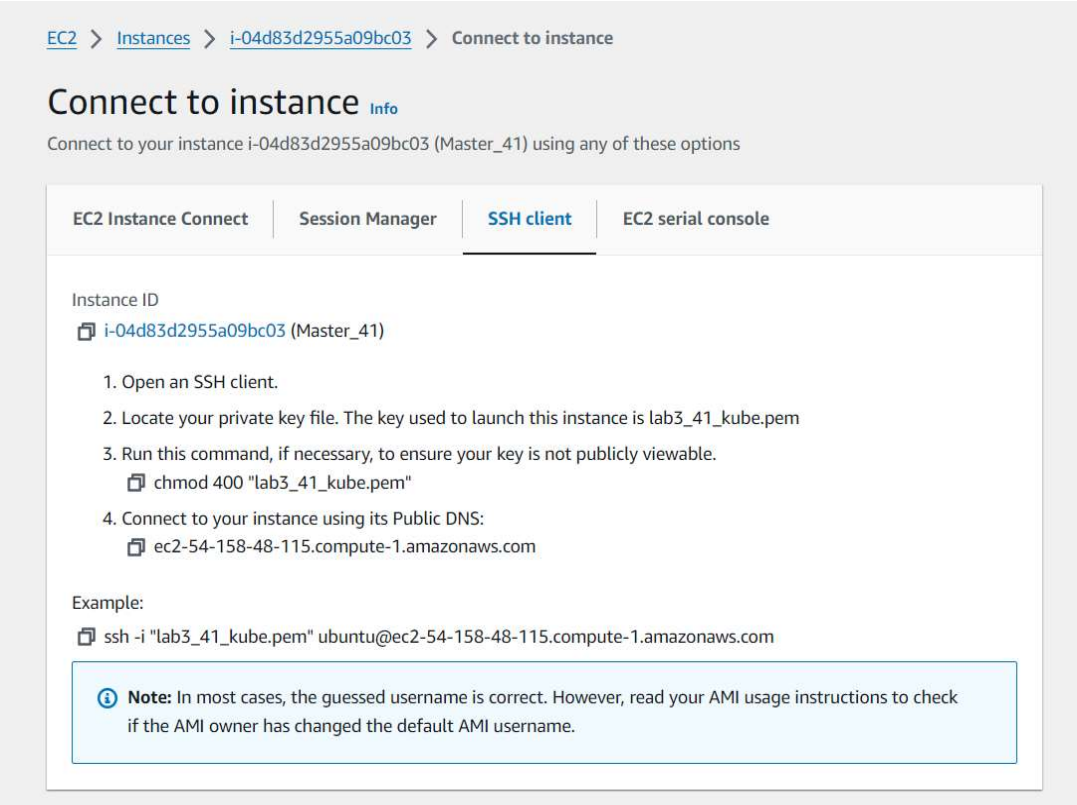
Select required Security Group

Instances are :

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

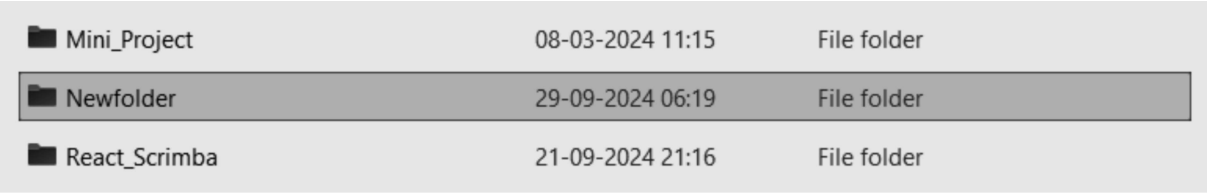


Master Node:



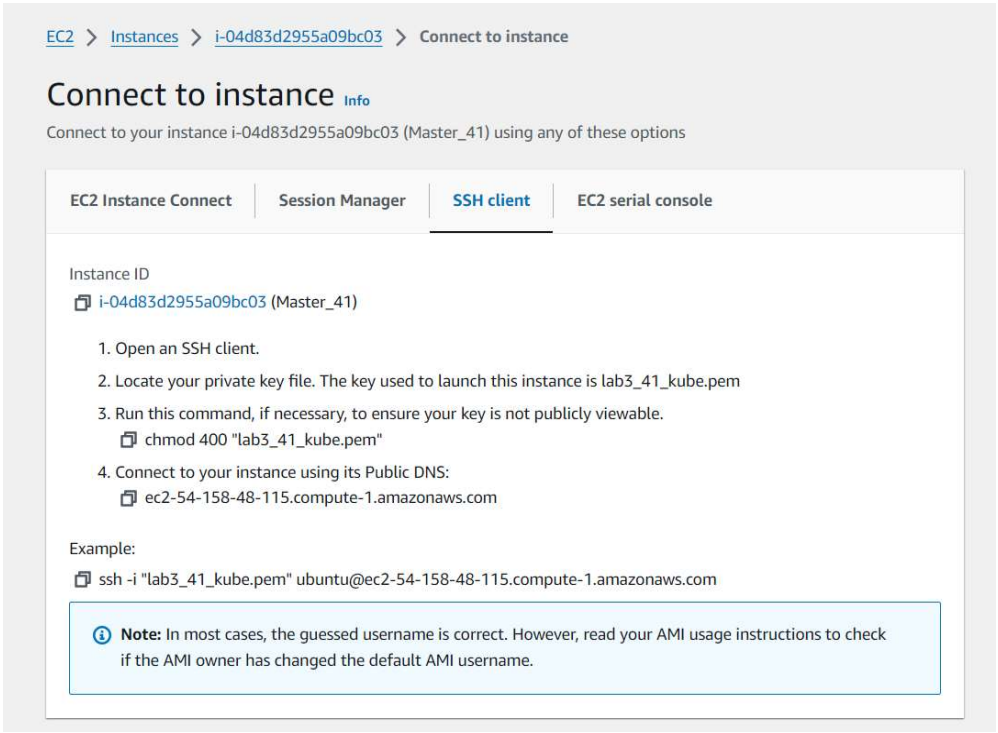
Step 3: Now open the folder in the terminal 3 times by right clicking on it for Master, Node1 & Node 2 where our .pem key is stored and paste the Example command (starting with ssh -i …..) from the ssh client section for instance in the terminal as shown above.

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

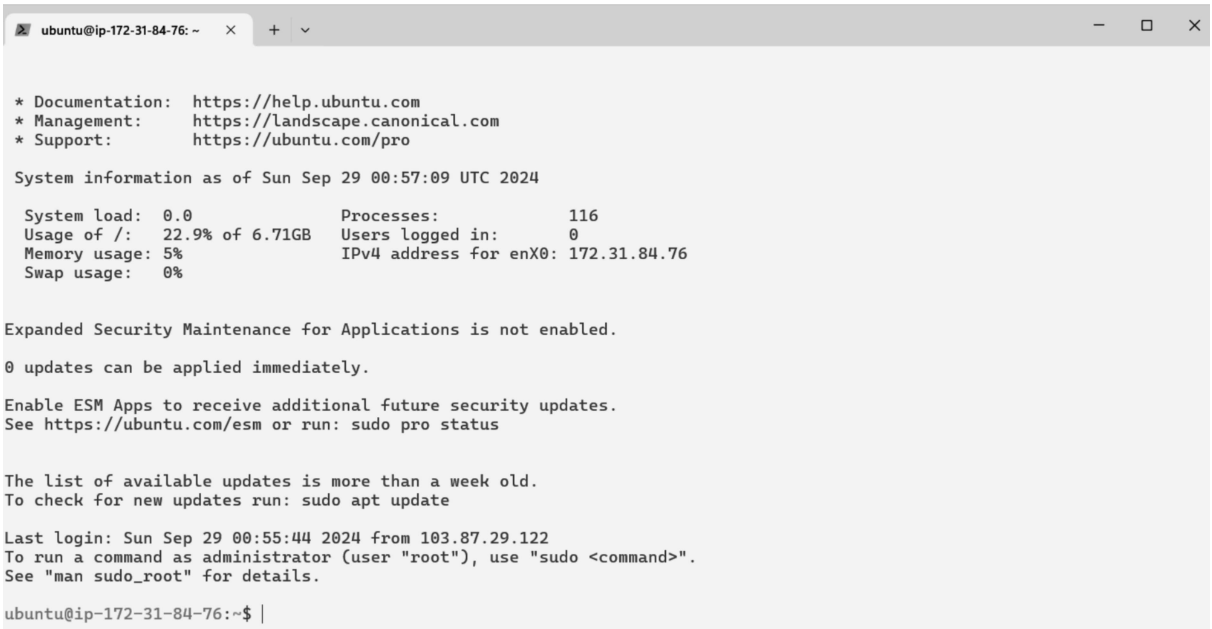


Name : Sandeshkumar.M.Yadav **Div : D15C** **Roll : 61**

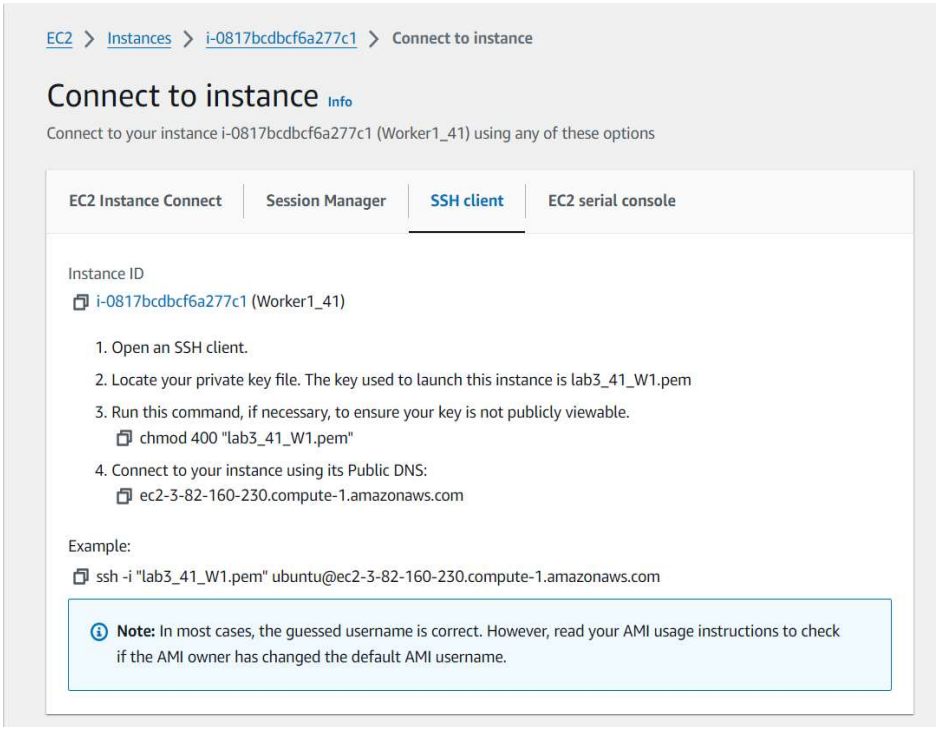
MasterNode:



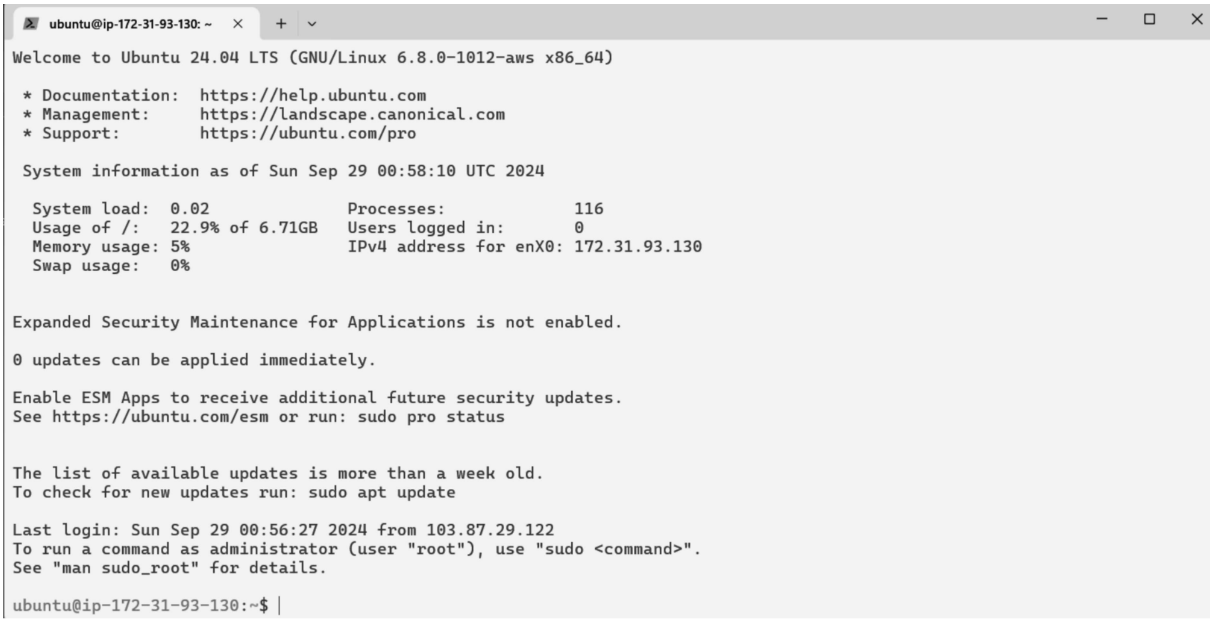
Successful Connection:



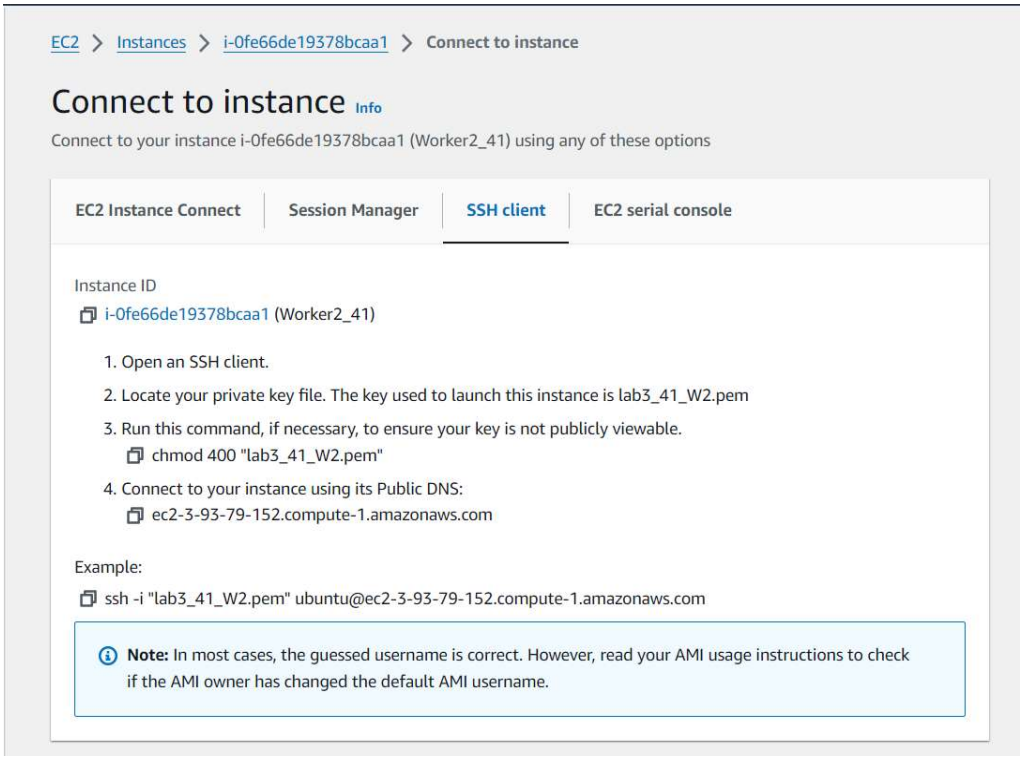
Worker Node 1:



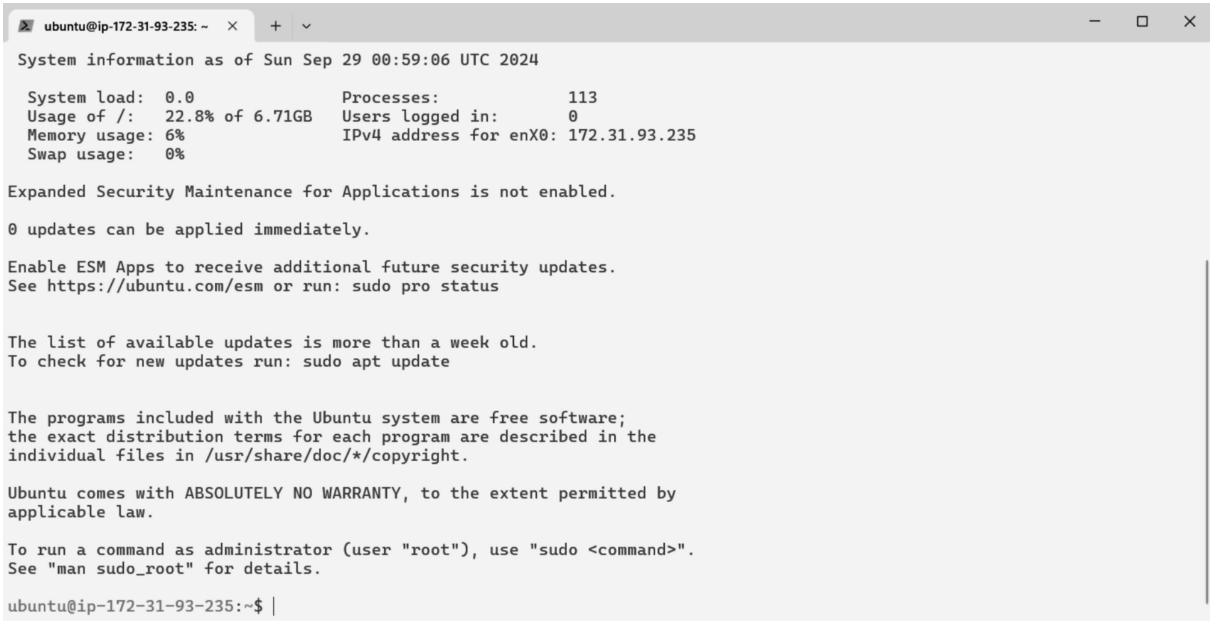
Successful Connection:



Worker Node 2:

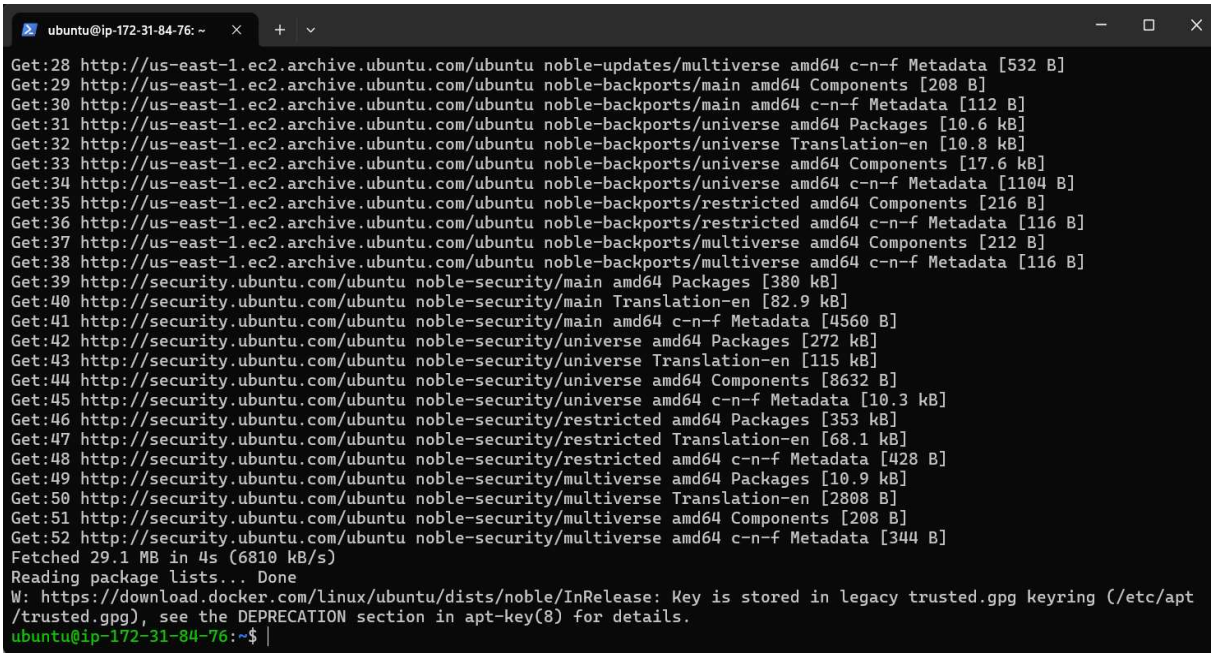


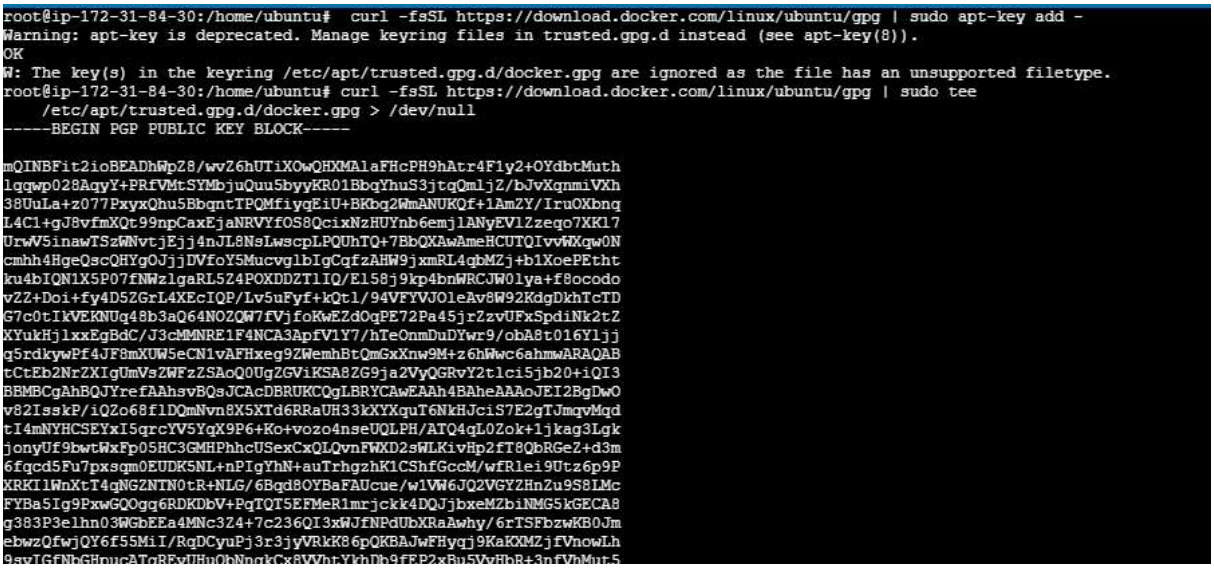
Successful Connection:



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

```
root@ip-172-31-88-203:/home/ubuntu#  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: The key(s) in the keyring /etc/apt/trusted.gpg.d/docker.gpg are ignc
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg),
details.
root@ip-172-31-88-203:/home/ubuntu#  sudo apt-get install -y docker-ce
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 sl
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]
```

```
root@ip-172-31-84-30:/home/ubuntu#  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 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: The key(s) in the keyring /etc/apt/trusted.gpg.d/docker.gpg are
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gp
details.
root@ip-172-31-84-30:/home/ubuntu#    sudo apt-get install -y docker-ce
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 slirp4ne
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite
The following NEW packages will be installed:
```

i-04343d785c0fca4d4 (61_exp_3_worker)

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

```
root@ip-172-31-88-203:/home/ubuntu# mkdir -p /etc/docker
root@ip-172-31-88-203:/home/ubuntu#  cat <<EOF | sudo tee /etc/docker/daemon.json
> {
  "exec-opts": ["native.cgroupdriver=systemd"]
 }
> EOF
{
  "exec-opts": ["native.cgroupdriver=systemd"]
 }
```

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

```
root@ip-172-31-88-203:/home/ubuntu# sudo systemctl enable docker
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/sy
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
root@ip-172-31-88-203:/home/ubuntu#  sudo systemctl daemon-reload
root@ip-172-31-88-203:/home/ubuntu#    sudo systemctl restart docker
root@ip-172-31-88-203:/home/ubuntu# docker -v
Docker version 27.3.1, build ce12230
```

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

```
root@ip-172-31-88-203:/home/ubuntu# rm -f /etc/apt/sources.list.d/kubernetes.list
```

b) sudo apt-get update

```
root@ip-172-31-88-203:/home/ubuntu# 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 InRelea
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: The key(s
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is st
details.
```

```
i-05d1cf82d5660367e (61_exp_3_Master)
```

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

```
root@ip-172-31-88-203:/home/ubuntu#  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:
  apt-transport-https
The following packages will be upgraded:
  curl libcurl3t64-gnutls libcurl4t64
3 upgraded, 1 newly installed, 0 to remove and 140 not upgraded.
Need to get 904 kB of archives.
After this operation, 38.9 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 apt-transport-https all 2.7.14bu
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 curl amd64 8.5.0-2ubuntu10.4
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 libcurl4t64 amd64 8.5.0-2ubu
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 libcurl3t64-gnutls amd64 8.5
Fetched 904 kB in 0s (16.1 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 ...
```

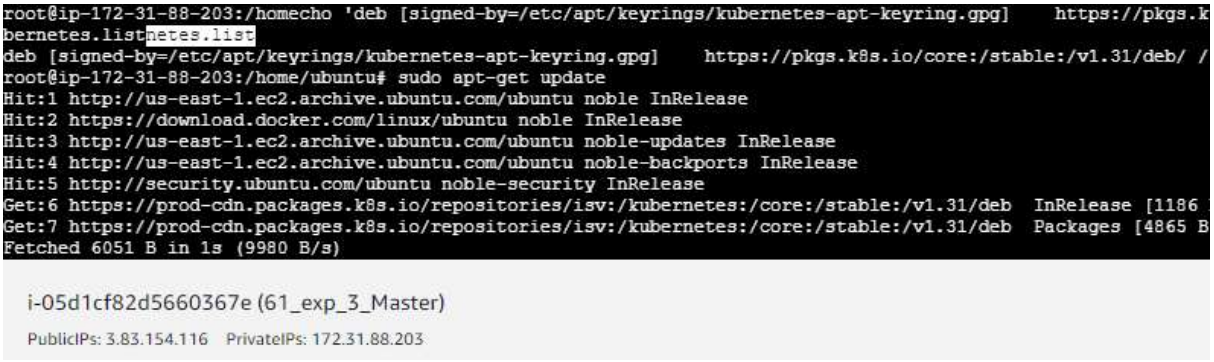
d) sudo mkdir -p -m 755 /etc/apt/keyrings

```
root@ip-172-31-88-203:/home/ubuntu#  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

```
root@ip-172-31-88-203:/home/ubuntu# curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg  --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpgeyring
pg
```

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



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

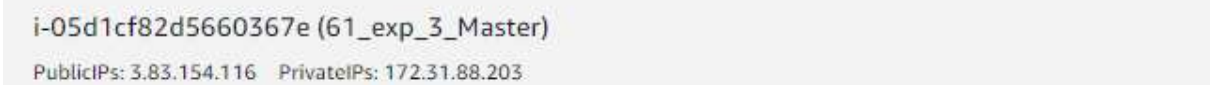
```
root@ip-172-31-88-203:/home/ubuntu# sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 https://download.docker.com/linux/ubuntu noble InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.
Get:7 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.
Fetched 6051 B in 1s (9980 B/s)
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: The key(s) in the k
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in le
details.
root@ip-172-31-88-203:/home/ubuntu#  sudo apt-get install -y kubelet kubeadm kubectl
Reading package lists... 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.
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
```

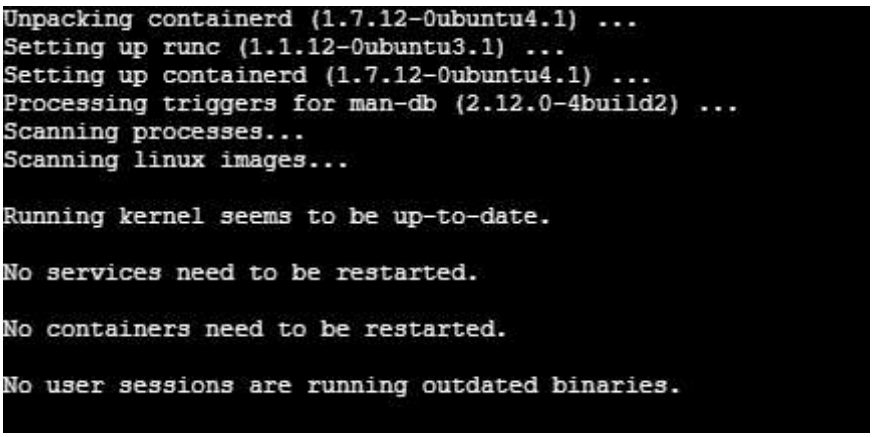
```
root@ip-172-31-88-203:/home/ubuntu#   sudo apt-mark hold kubelet kubeadm kubectl
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
```

g) sudo systemctl enable --now kubelet

```
root@ip-172-31-88-203:/home/ubuntu#  systemctl enable --now kubelet
root@ip-172-31-88-203:/home/ubuntu#  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 libl
Use 'sudo apt autoremove' to remove them.
```



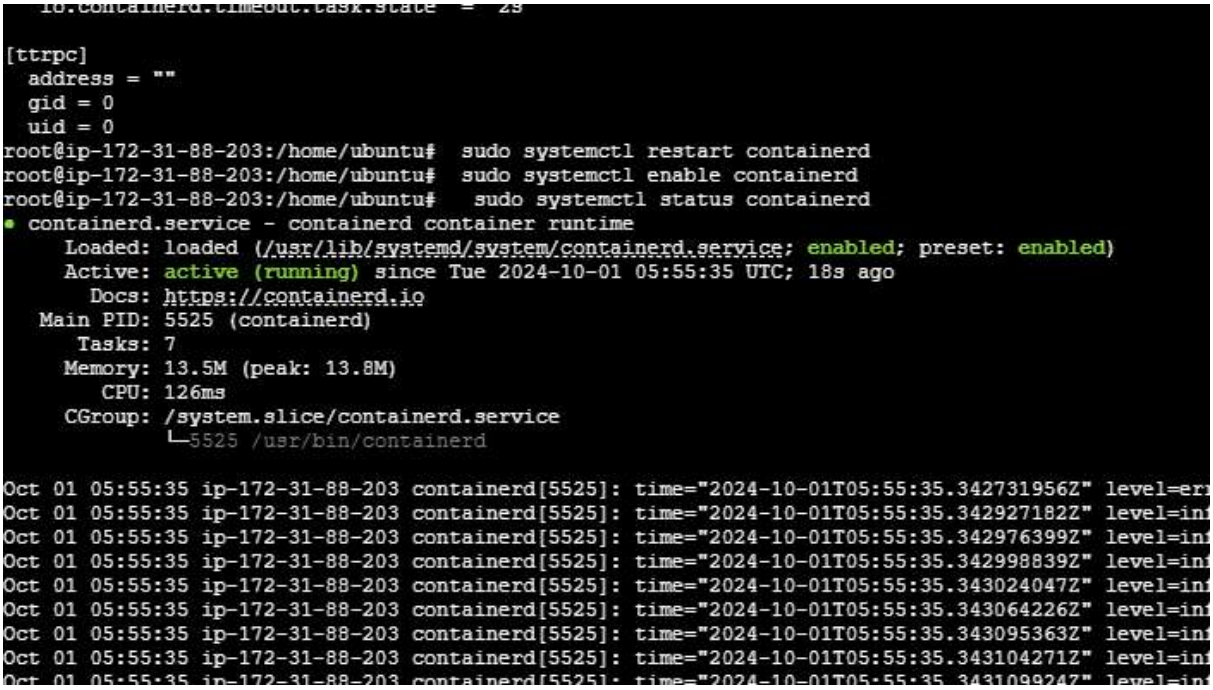
h) sudo apt-get install -y containerd



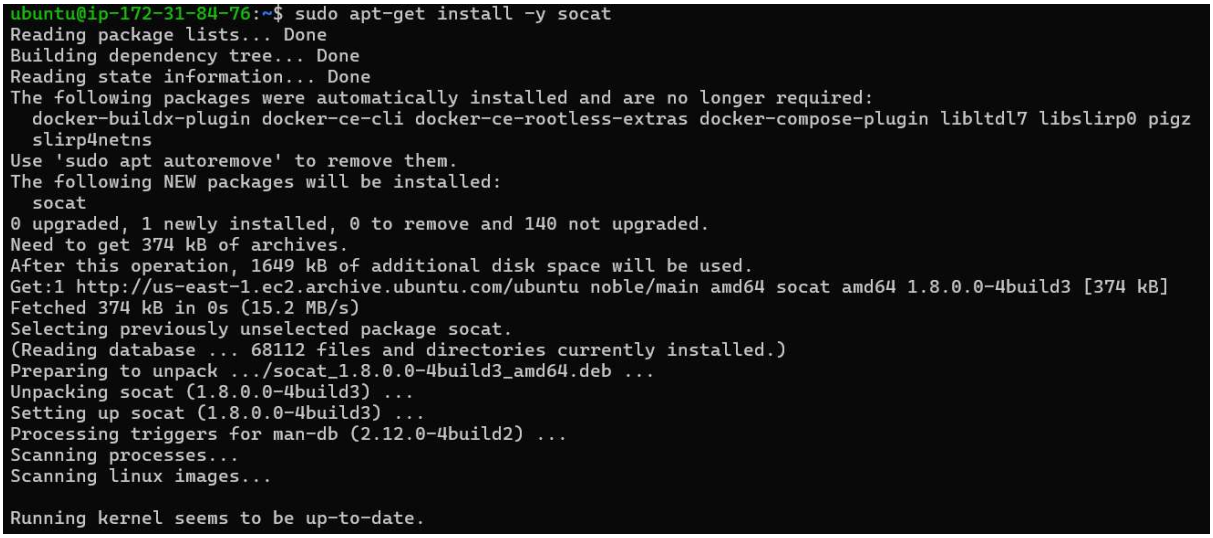
i) sudo mkdir -p /etc/containerd
sudo containerd config default | sudo tee /etc/containerd/config.toml

```
root@ip-172-31-88-203:/home/ubuntu#  sudo mkdir -p /etc/containerd
root@ip-172-31-88-203:/home/ubuntu# 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
```

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



k) sudo apt-get install -y socat



```
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.
```

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

```
root@ip-172-31-88-203:/home/ubuntu# 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 conn
[preflight] You can also perform this action beforehand using 'kubeadm config images pull
W1001 05:56:40.054582    5726 checks.go:846] detected that the sandbox image "registry.k8
ommended to use "registry.k8s.io/pause:3.10" as the CRI sandbox image.
[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-172-31-88-203 kubernetes kuber
1 172.31.88.203]
[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
```

i-05d1cf82d5660367e (61_exp_3_Master)

PublicIPs: 3.83.154.116 PrivateIPs: 172.31.88.203

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

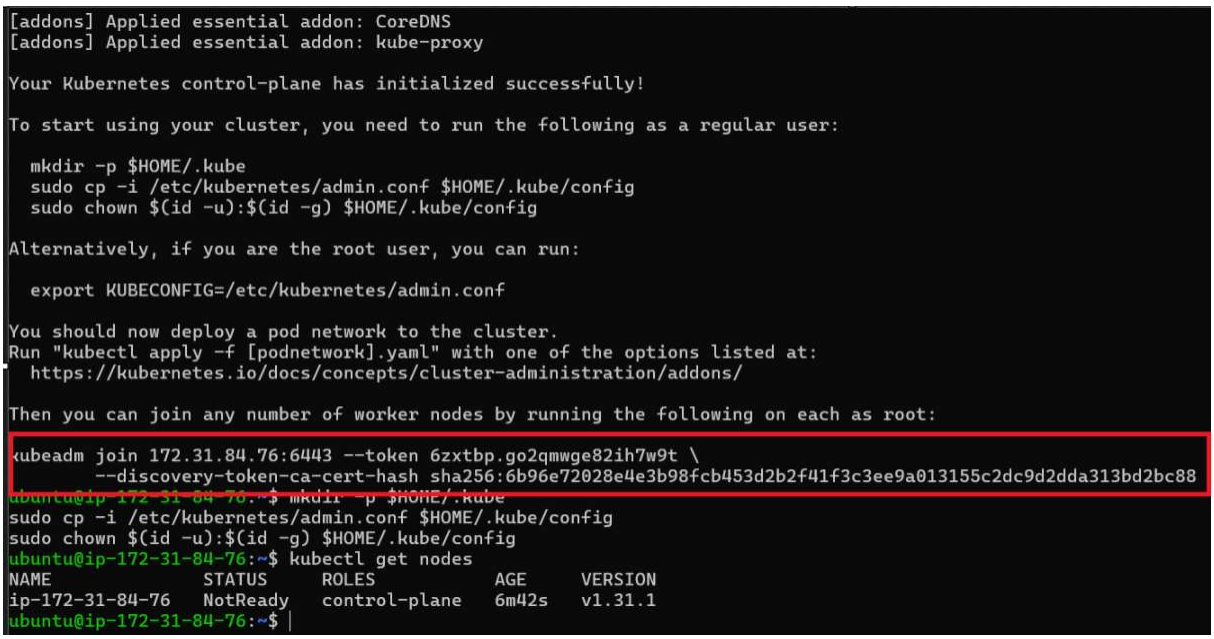
These commands create a .kube directory in your home folder, copy the Kubernetes admin configuration file into it, and change the ownership of that configuration file to the current user to ensure proper access.

```
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 172.31.88.203:6443 --token ef5xtx.okbe4aouwj2i4fdf \
        --discovery-token-ca-cert-hash sha256:71074273c6192db966a142598ef7d65305691636883a1cf66233df241d4c7c76
root@ip-172-31-88-203:/home/ubuntu#  mkdir -p $HOME/.kube
root@ip-172-31-88-203:/home/ubuntu# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
root@ip-172-31-88-203:/home/ubuntu# sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

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.

```
root@ip-172-31-88-203:/home/ubuntu# kubectl get nodes
NAME               STATUS     ROLES           AGE   VERSION
ip-172-31-88-203   NotReady   control-plane   57s   v1.31.1
```

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



Copy and paste: ***kubeadm join 172.31.88.203:6443 --token ef5xtx.okbe4aouwj2i4fdf ***

--discovery-token-ca-cert-hash sha256:71074273c6192db966a142598ef7d65305691636883a1cf66233df241d4c7c76

Worker Node 1:

```
root@ip-172-31-84-30:/home/ubuntu# kubeadm join 172.31.88.203:6443 --token ef5xtx.okbe4aouwj2i4fdf \
        --discovery-token-ca-cert-hash sha256:71074273c6192db966a142598ef7d65305691636883a1cf66233df241d4c7c76
[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 1.001611728s
[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.

root@ip-172-31-84-30:/home/ubuntu#
```

i-04343d785c0fca4d4 (61_exp_3_worker)

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

```
root@ip-172-31-88-203:/home/ubuntu# kubectl get nodes -o wide
NAME              STATUS   ROLES           AGE     VERSION   INTE
ip-172-31-84-30   Ready    Worker_Node1_61   2m43s   v1.31.1   172.3
ip-172-31-88-203  Ready    control-plane     13m     v1.31.1   172.3
```

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

```
root@ip-172-31-88-203:/home/ubuntu# 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
```

i-05d1cf82d5660367e (61_exp_3_Master)
PublicIPs: 3.83.154.116 PrivateIPs: 172.31.88.203

Now perform **sudo systemctl status kubelet**

```
root@ip-172-31-88-203:/home/ubuntu# 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 Tue 2024-10-01 05:56:58 UTC; 11min ago
       Docs: https://kubernetes.io/docs/
   Main PID: 6396 (kubelet)
      Tasks: 10 (limit: 4676)
     Memory: 32.5M (peak: 33.0M)
```

i-05d1cf82d5660367e (61_exp_3_Master)
PublicIPs: 3.83.154.116 PrivateIPs: 172.31.88.203

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_61: kubectl label node ip-172-31-84-30 kubernetes.io/role=Worker_Node_61

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

```
root@ip-172-31-88-203:/home/ubuntu# kubectl get nodes -o wide
NAME              STATUS   ROLES                          AGE     VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE
CONTAINER-RUNTIME
ip-172-31-84-30   Ready    Worker_Node1_61                3m47s   v1.31.1   172.31.84.30    <none>        Ubuntu 24.04 LTS
containerd://1.7.12
ip-172-31-88-203  Ready    Master_Node_61,control-plane   14m     v1.31.1   172.31.88.203   <none>        Ubuntu 24.04 LTS
containerd://1.7.12
root@ip-172-31-88-203:/home/ubuntu#
```

i-05d1cf82d5660367e (61_exp_3_Master)
PublicIPs: 3.83.154.116 PrivateIPs: 172.31.88.203

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.


```
root@ip-172-31-88-203:/home/ubuntu# kubectl get nodes -o wide
NAME              STATUS   ROLES                          AGE     VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE
CONTAINER-RUNTIME
ip-172-31-84-30   Ready    Worker_Node1_61                3m47s   v1.31.1   172.31.84.30    <none>        Ubuntu 24.04 LTS
containerd://1.7.12
ip-172-31-88-203  Ready    Master_Node_61,control-plane   14m     v1.31.1   172.31.88.203   <none>        Ubuntu 24.04 LTS
containerd://1.7.12
root@ip-172-31-88-203:/home/ubuntu#
```

i-05d1cf82d5660367e (61_exp_3_Master)
PublicIPs: 3.83.154.116 PrivateIPs: 172.31.88.203

CONCLUSION:

In this experiment, we established a Kubernetes cluster on AWS EC2 instances, consisting of one master node and two worker nodes. After installing Docker and the necessary Kubernetes tools (kubelet, kubeadm, kubectl, and containerd) on all nodes, we initialized the master and added the worker nodes to the cluster. Initially, the nodes showed a NotReady status, which we resolved by installing the Calico network plugin. We also tagged the nodes with their respective roles (control-plane and worker). Ultimately, all nodes transitioned to the Ready state, demonstrating our successful setup and management of the Kubernetes cluster.

Experiment No: 4

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

Theory :

Kubernetes, originally developed by Google, is an open-source container orchestration platform. It automates the deployment, scaling, and management of containerized applications, ensuring high availability and fault tolerance. Kubernetes is now the industry standard for container orchestration and is governed by the Cloud Native Computing Foundation (CNCF), with contributions from major cloud and software providers like Google, AWS, Microsoft, IBM, Intel, Cisco, and Red Hat. Kubernetes Deployment: Is a resource in Kubernetes that provides declarative updates for Pods and ReplicaSets. With a Deployment, you can define how many replicas of a pod should run, roll out new versions of an application, and roll back to previous versions if necessary. It ensures that the desired number of pod replicas are running at all times.

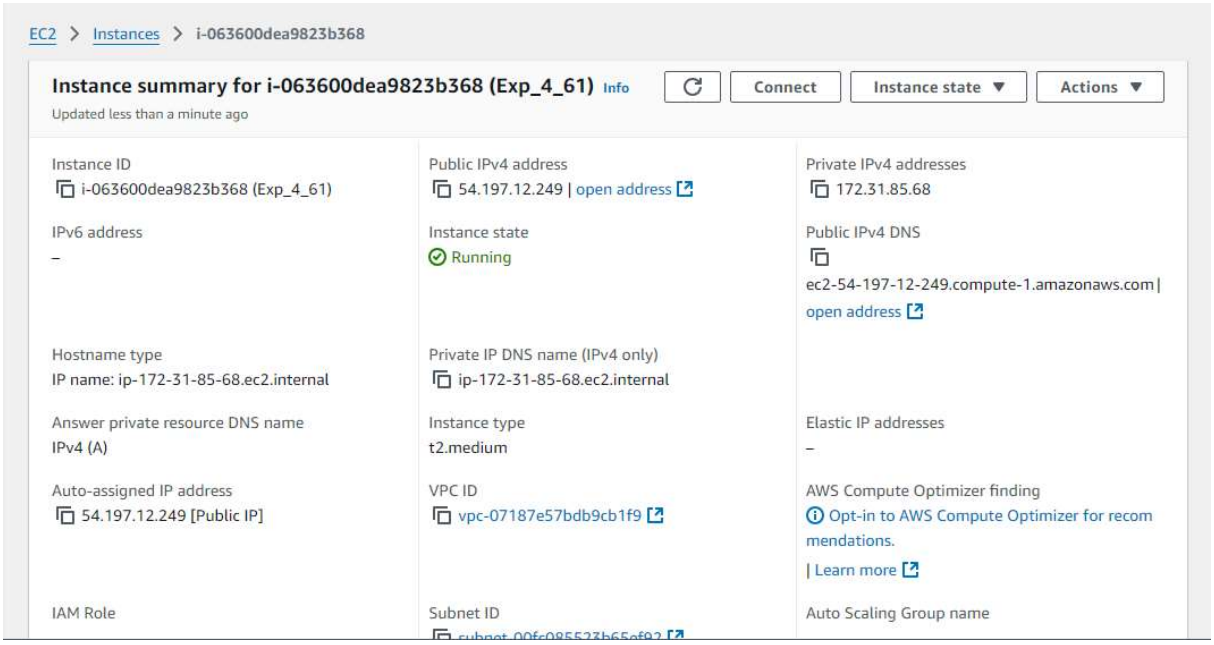
Necessary Requirements:

- EC2 Instance: The experiment required launching a t2.medium EC2 instance with 2 CPUs, as Kubernetes demands sufficient resources for effective functioning.
- Minimum Requirements:
  - Instance Type: t2.medium
  - CPUs: 2
  - Memory: Adequate for container orchestration.

This ensured that the Kubernetes cluster had the necessary resources to function smoothly

Name : Sandeshkumar.M.Yadav **Div : D15C** **Roll : 61**

Step 1: Log in to your AWS Academy/personal account and launch a new Ec2 Instance. Select Ubuntu as AMI and t2.medium as Instance Type, create a key of type RSA with .pem extension, and move the downloaded key to the new folder.



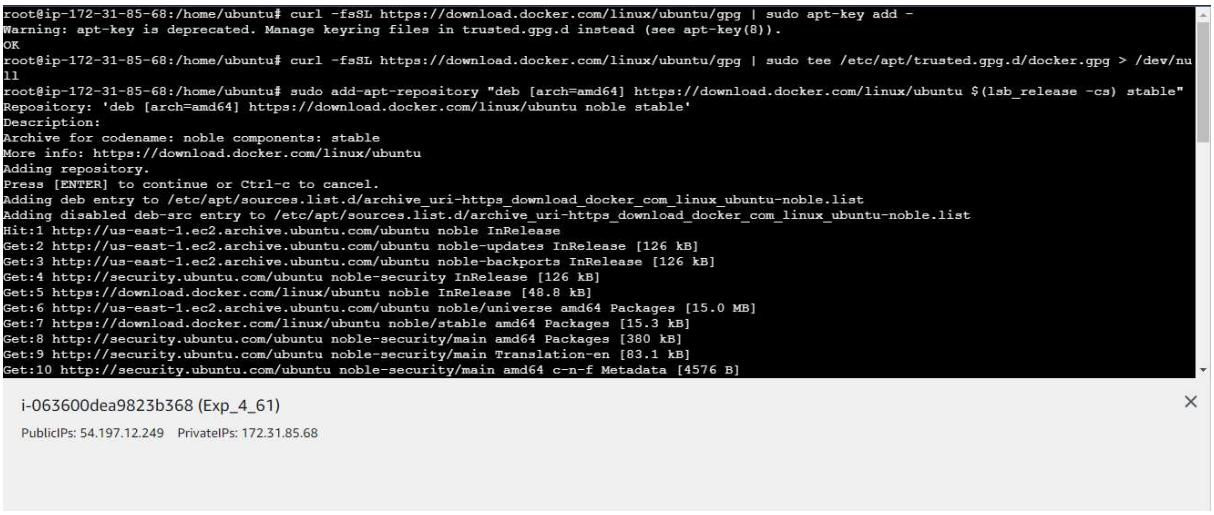
Step 2 :

Run the below commands to install and setup Docker.

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"



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

```
root@ip-172-31-85-68:/home/ubuntu# 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: The key(s) in the keyring /etc/apt/trusted.gpg.d/docker.gpg are ignored as the
 file has an unsupported filetype.
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DE
PRECATION section in apt-key(8) for details.
root@ip-172-31-85-68:/home/ubuntu# sudo apt-get install -y docker-ce
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.
```

i-063600dea9823b368 (Exp_4_61)
PublicIPs: 54.197.12.249 PrivateIPs: 172.31.85.68

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

```
root@ip-172-31-85-68:/home/ubuntu# sudo mkdir -p /etc/docker
root@ip-172-31-85-68:/home/ubuntu# cat <<EOF | sudo tee /etc/docker/daemon.json
> {
"exec-opts": ["native.cgroupdriver=systemd"]
}
> EOF
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
root@ip-172-31-85-68:/home/ubuntu#
```

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

```
root@ip-172-31-85-68:/home/ubuntu# sudo systemctl enable 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
root@ip-172-31-85-68:/home/ubuntu# sudo systemctl daemon-reload
root@ip-172-31-85-68:/home/ubuntu# sudo systemctl restart docker
root@ip-172-31-85-68:/home/ubuntu#
```

i-063600dea9823b368 (Exp_4_61)
PublicIPs: 54.197.12.249 PrivateIPs: 172.31.85.68

Step 3 :
Run the below command to install Kubernets.
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

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

```
root@ip-172-31-85-68:/home/ubuntu# curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/k
ubernetes-apt-keyring.gpg
root@ip-172-31-85-68:/home/ubuntu# 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
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
root@ip-172-31-85-68:/home/ubuntu#
```

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

```
root@ip-172-31-85-68:/home/ubuntu# sudo apt-get update
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
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  InRelease [1186 B]
Get:7 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  Packages [4865 B]
Fetched 6051 B in 1s (11.5 kB/s)
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: The key(s) in the keyring /etc/apt/trusted.gpg.d/docker.gpg are ignored
 file has an unsupported filetype.
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see
PRECATION section in apt-key(8) for details.
root@ip-172-31-85-68:/home/ubuntu# sudo apt-get install -y kubelet kubeadm kubectl
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 143 not upgraded.
Need to get 87.4 MB of archives.
```

i-063600dea9823b368 (Exp_4_61)
PublicIPs: 54.197.12.249 PrivateIPs: 172.31.85.68

```
root@ip-172-31-85-68:/home/ubuntu# sudo apt-mark hold kubelet kubeadm kubectl
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
root@ip-172-31-85-68:/home/ubuntu#
```

i-063600dea9823b368 (Exp_4_61)
PublicIPs: 54.197.12.249 PrivateIPs: 172.31.85.68

sudo systemctl enable --now kubelet
sudo kubeadm init --pod-network-cidr=10.244.0.0/16

```
root@ip-172-31-85-68:/home/ubuntu# sudo systemctl enable --now kubelet
root@ip-172-31-85-68:/home/ubuntu# sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
W1001 17:30:48.692533    5553 checks.go:1080] [preflight] WARNING: Couldn't create the int
d to create new CRI runtime service: validate service connection: validate CRI v1 runtime
.sock": rpc error: code = 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 internet conne
[preflight] You can also perform this action beforehand using 'kubeadm config images pull'
error execution phase preflight: [preflight] Some fatal errors occurred:
failed to create new CRI runtime service: validate service connection: validate CRI v1 run
inerd.sock": rpc error: code = Unimplemented desc = unknown service runtime.v1.RuntimeServ
make a check non-fatal with `--ignore-preflight-errors=...`
To see the stack trace of this error execute with --v=5 or higher
root@ip-172-31-85-68:/home/ubuntu#
```

i-063600dea9823b368 (Exp_4_61)

PublicIPs: 54.197.12.249 PrivateIPs: 172.31.85.68

sudo apt-get install -y containerd

```
root@ip-172-31-85-68:/home/ubuntu# 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 requi
  docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compo
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 143 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 amd
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd
Fetched 47.2 MB in 1s (53.2 MB/s)
(Reading database ... 68064 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 ... 68044 files and directories currently installed.)
```

i-063600dea9823b368 (Exp_4_61)

PublicIPs: 54.197.12.249 PrivateIPs: 172.31.85.68

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

```
root@ip-172-31-85-68:/home/ubuntu# sudo mkdir -p /etc/containerd
root@ip-172-31-85-68:/home/ubuntu# 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"
```

i-063600dea9823b368 (Exp_4_61)

PublicIPs: 54.197.12.249 PrivateIPs: 172.31.85.68

sudo systemctl restart containerd
sudo systemctl enable containerd
sudo systemctl status containerd

```
root@ip-172-31-85-68:/home/ubuntu# sudo systemctl enable containerd
root@ip-172-31-85-68:/home/ubuntu# sudo systemctl status containerd
● containerd.service - containerd container runtime
     Loaded: loaded (/usr/lib/systemd/system/containerd.service; enabled; preset: enabled)
     Active: active (running) since Tue 2024-10-01 17:32:39 UTC; 24s ago
       Docs: https://containerd.io
   Main PID: 5964 (containerd)
      Tasks: 8
     Memory: 13.6M (peak: 14.0M)
        CPU: 115ms
     CGroup: /system.slice/containerd.service
             └─5964 /usr/bin/containerd

Oct 01 17:32:39 ip-172-31-85-68 containerd[5964]: time="2024-10-01T17:32:39.371130654Z" level=
Oct 01 17:32:39 ip-172-31-85-68 containerd[5964]: time="2024-10-01T17:32:39.371164383Z" level=
Oct 01 17:32:39 ip-172-31-85-68 containerd[5964]: time="2024-10-01T17:32:39.371202243Z" level=
Oct 01 17:32:39 ip-172-31-85-68 containerd[5964]: time="2024-10-01T17:32:39.371227104Z" level=
Oct 01 17:32:39 ip-172-31-85-68 containerd[5964]: time="2024-10-01T17:32:39.372111713Z" level=
Oct 01 17:32:39 ip-172-31-85-68 containerd[5964]: time="2024-10-01T17:32:39.372136114Z" level=
Oct 01 17:32:39 ip-172-31-85-68 containerd[5964]: time="2024-10-01T17:32:39.372144030Z" level=
Oct 01 17:32:39 ip-172-31-85-68 containerd[5964]: time="2024-10-01T17:32:39.372151172Z" level=
Oct 01 17:32:39 ip-172-31-85-68 containerd[5964]: time="2024-10-01T17:32:39.372205611Z" level=
Oct 01 17:32:39 ip-172-31-85-68 systemd[1]: Started containerd.service - containerd container
lines 1-21/21 (END)
```

i-063600dea9823b368 (Exp_4_61)

PublicIPs: 54.197.12.249 PrivateIPs: 172.31.85.68

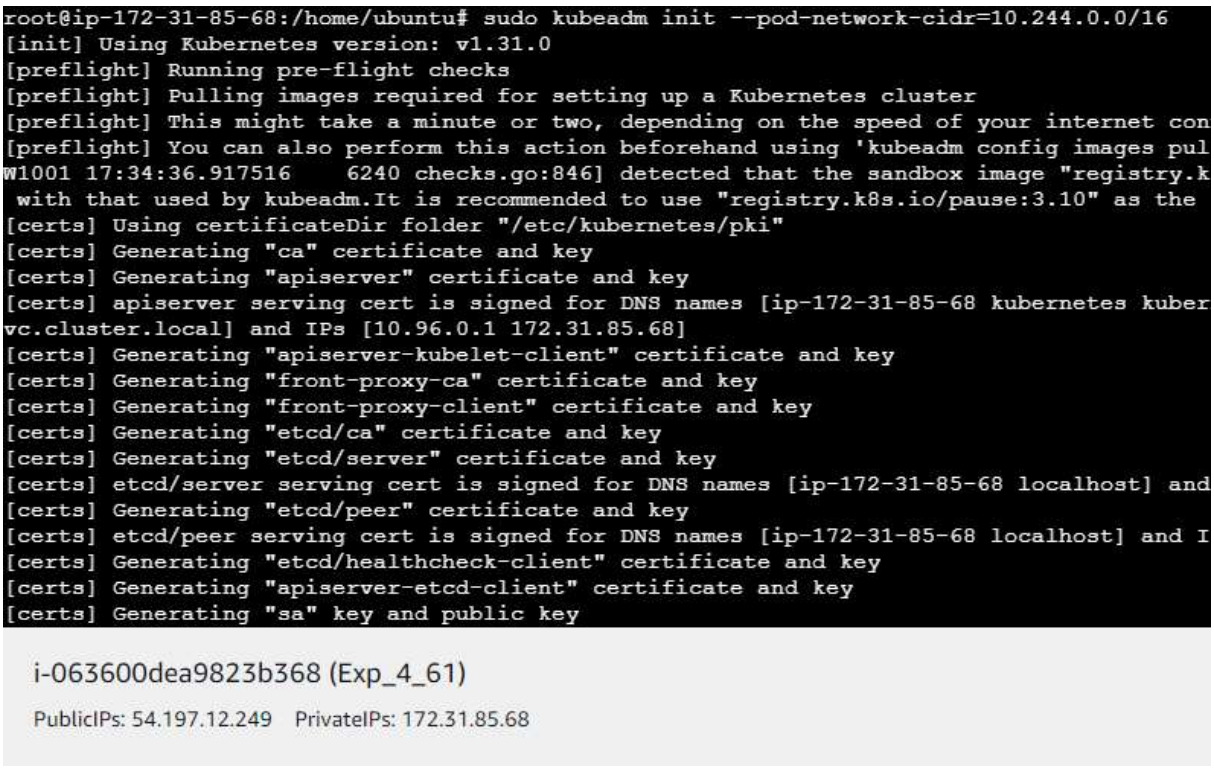
sudo apt-get install -y socat

```
root@ip-172-31-85-68:/home/ubuntu# sudo apt-get install -y socat
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longe
  docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docke
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  socat
0 upgraded, 1 newly installed, 0 to remove and 143 not upgraded.
Need to get 374 kB of archives.
After this operation, 1649 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd6
Fetched 374 kB in 0s (14.5 MB/s)
Selecting previously unselected package socat.
(Reading database ... 68108 files and directories currently installe
Preparing to unpack .../socat_1.8.0.0-4build3_amd64.deb ...
Unpacking socat (1.8.0.0-4build3) ...
Setting up socat (1.8.0.0-4build3) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

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

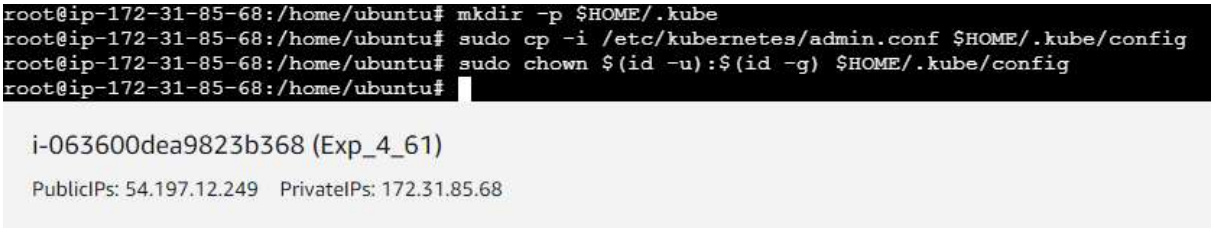
i-063600dea9823b368 (Exp_4_61)

PublicIPs: 54.197.12.249 PrivateIPs: 172.31.85.68

Step 4 : Initialize the Kubecluster
sudo kubeadm init --pod-network-cidr=10.244.0.0/16



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



Add a common networking plugin called flannel as mentioned in the code.
kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

Step 5 :
Now that the cluster is up and running, we can deploy our nginx server on this cluster.Apply this deployment file using this command to create a deployment
kubectl apply -f https://k8s.io/examples/application/deployment.yaml

```
root@ip-172-31-85-68:/home/ubuntu# kubectl apply -f https://k8s.io/examples/application/deployment.yaml
deployment.apps/nginx-deployment created
root@ip-172-31-85-68:/home/ubuntu#
```

kubectl get pods

```
root@ip-172-31-85-68:/home/ubuntu# kubectl get pods
NAME                               READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-4pf7j   0/1     Pending   0          25s
nginx-deployment-d556bf558-dqs9g   0/1     Pending   0          25s
root@ip-172-31-85-68:/home/ubuntu#
```

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

```
root@ip-172-31-85-68:/home/ubuntu# POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
root@ip-172-31-85-68:/home/ubuntu# kubectl port-forward $POD_NAME 8080:80
error: unable to forward port because pod is not running. Current status=Pending
root@ip-172-31-85-68:/home/ubuntu#
```

We have faced an error as pod status is pending so make it running run below commands then again run above 2 commands.
kubectl taint nodes --all node-role.kubernetes.io/control-plane-node/ip-172-31-20-171 untainted
kubectl get nodes

OR

kubectl taint nodes ip-172-31-85-68
node-role.kubernetes.io/control-plane:NoSchedule-

```
root@ip-172-31-85-68:/home/ubuntu# kubectl taint nodes --all node-role.kubernetes.io/control-plane/ip-172-31-85-68 untainted
error: at least one taint update is required
root@ip-172-31-85-68:/home/ubuntu#
root@ip-172-31-85-68:/home/ubuntu#
```

i-063600dea9823b368 (Exp_4_61)
PublicIPs: 54.197.12.249 PrivateIPs: 172.31.85.68

Kubectl get pods

```
root@ip-172-31-85-68:/home/ubuntu# kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-4pf7j    1/1     Running   0          10m
nginx-deployment-d556bf558-dqs9g    1/1     Running   0          10m
root@ip-172-31-85-68:/home/ubuntu#
```

i-063600dea9823b368 (Exp_4_61)

PublicIPs: 54.197.12.249 PrivateIPs: 172.31.85.68

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

kubectl port-forward $POD_NAME 8080:80

```
root@ip-172-31-85-68:/home/ubuntu# POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
root@ip-172-31-85-68:/home/ubuntu# kubectl port-forward $POD_NAME 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
^Croot@ip-172-31-85-68:/home/ubuntu# kubectl port-forward $POD_NAME 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
Handling connection for 8080
```

Step 6 : Verify your deployment

Open up a new terminal and ssh to your EC2 instance.

Then, use this curl command to check if the Nginx server is running.

curl --head http://127.0.0.1:8080

```
root@ip-172-31-85-68:/home/ubuntu# curl --head http://127.0.0.1:8080
HTTP/1.1 200 OK
Server: nginx/1.14.2
Date: Tue, 01 Oct 2024 17:53:16 GMT
Content-Type: text/html
Content-Length: 612
Last-Modified: Tue, 04 Dec 2018 14:44:49 GMT
Connection: keep-alive
ETag: "5c0692e1-264"
Accept-Ranges: bytes

root@ip-172-31-85-68:/home/ubuntu#
```

if the response is 200 OK and you can see the Nginx server name, your deployment was successful. We have successfully deployed our Nginx server on our EC2 instance.

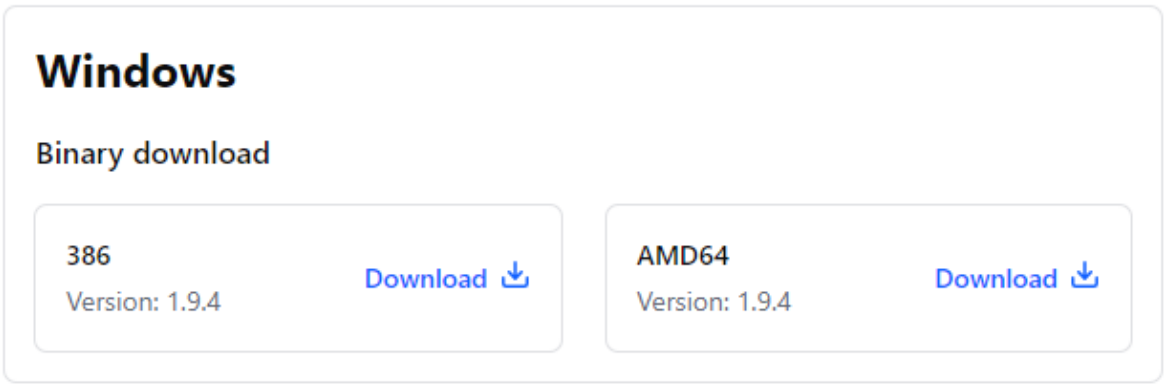
Conclusion:

In this experiment, we successfully installed Kubernetes on an EC2 instance and deployed an Nginx server using Kubectl commands. We encountered two key challenges: first, the Kubernetes pod was initially in a pending state, which we resolved by removing the control-plane taint with the command kubectl taint nodes --all. Second, we faced an issue with the missing containerd runtime, which we fixed by installing and starting containerd. Utilising a t2.medium EC2 instance with 2 CPUs ensured that we met the necessary resource requirements for the Kubernetes setup and deployment.

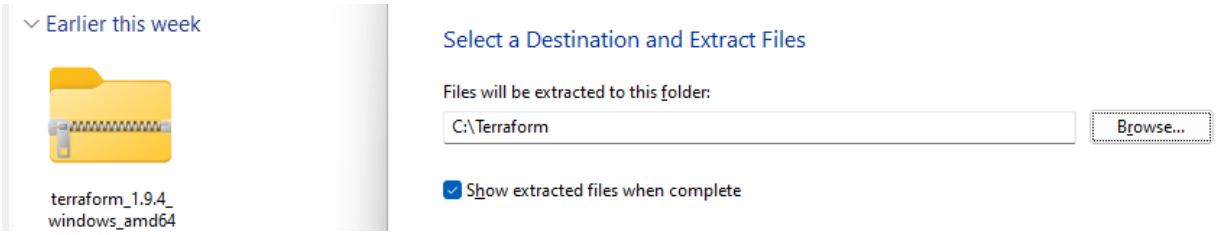
EXPERIMENT NO. 05

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

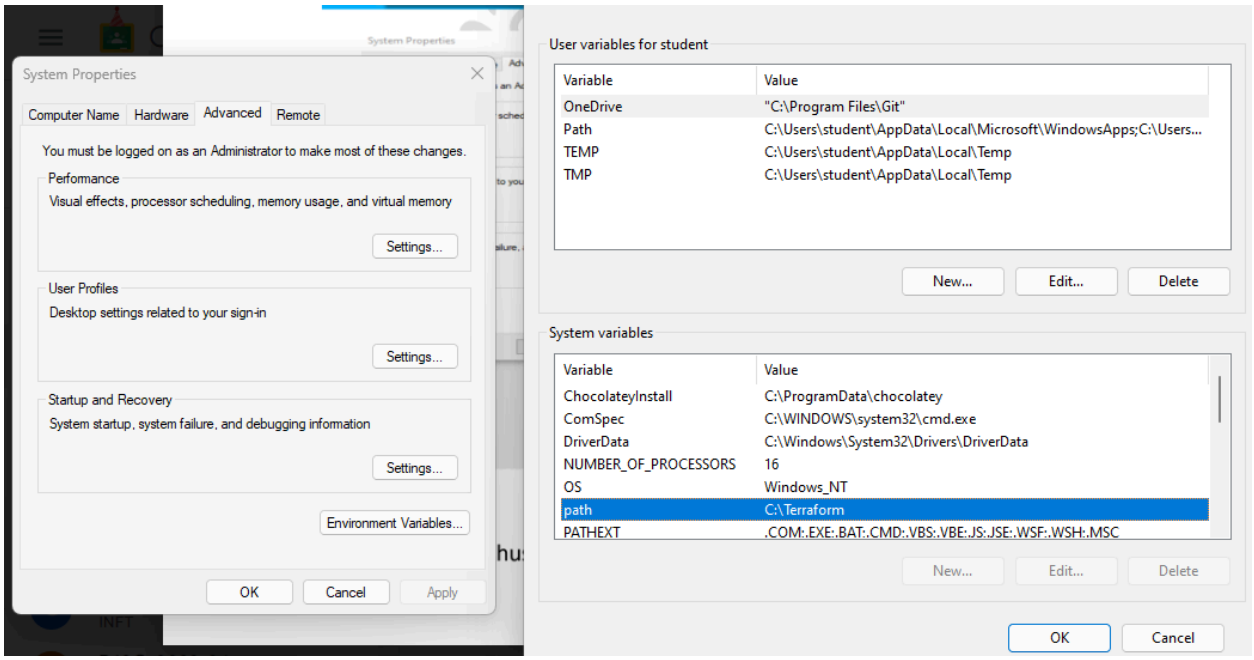
Step 1: Download terraform
A) Installation and Configuration of Terraform in Windows
To install Terraform, First Download the Terraform Cli Utility for windows from terraforms official website
website:https://www.terraform.io/downloads.html
Select the Operating System Windows followed by either 32bit or 64 bit based on your OS type.



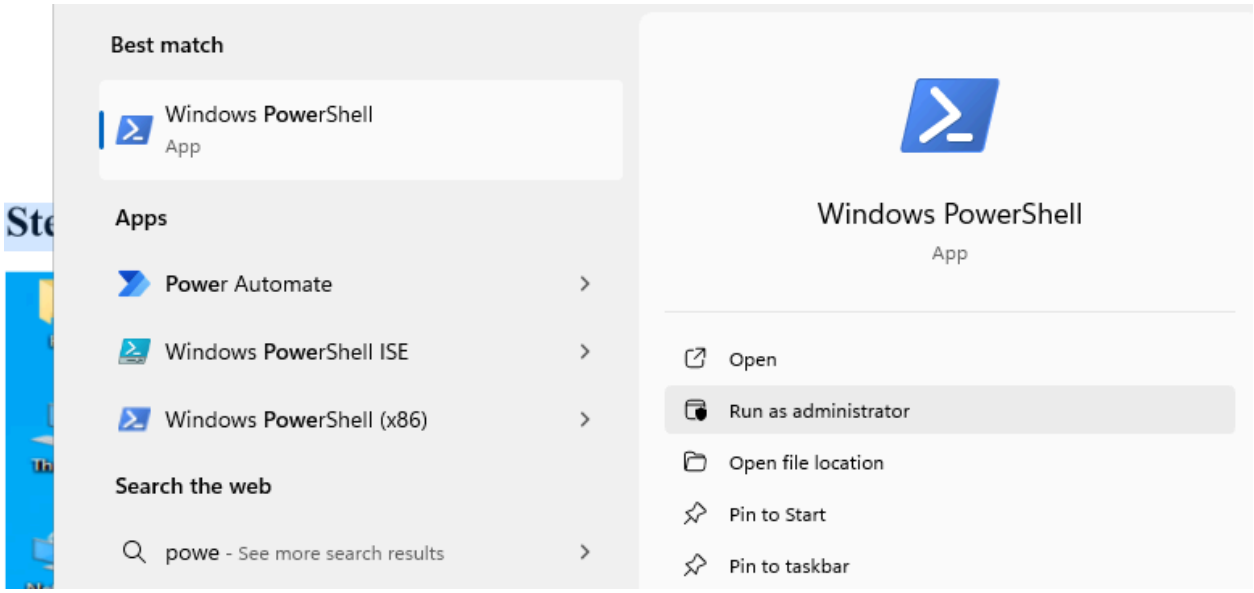
Step 2: Extract the downloaded setup file Terraform.exe in C:\Terraform directory



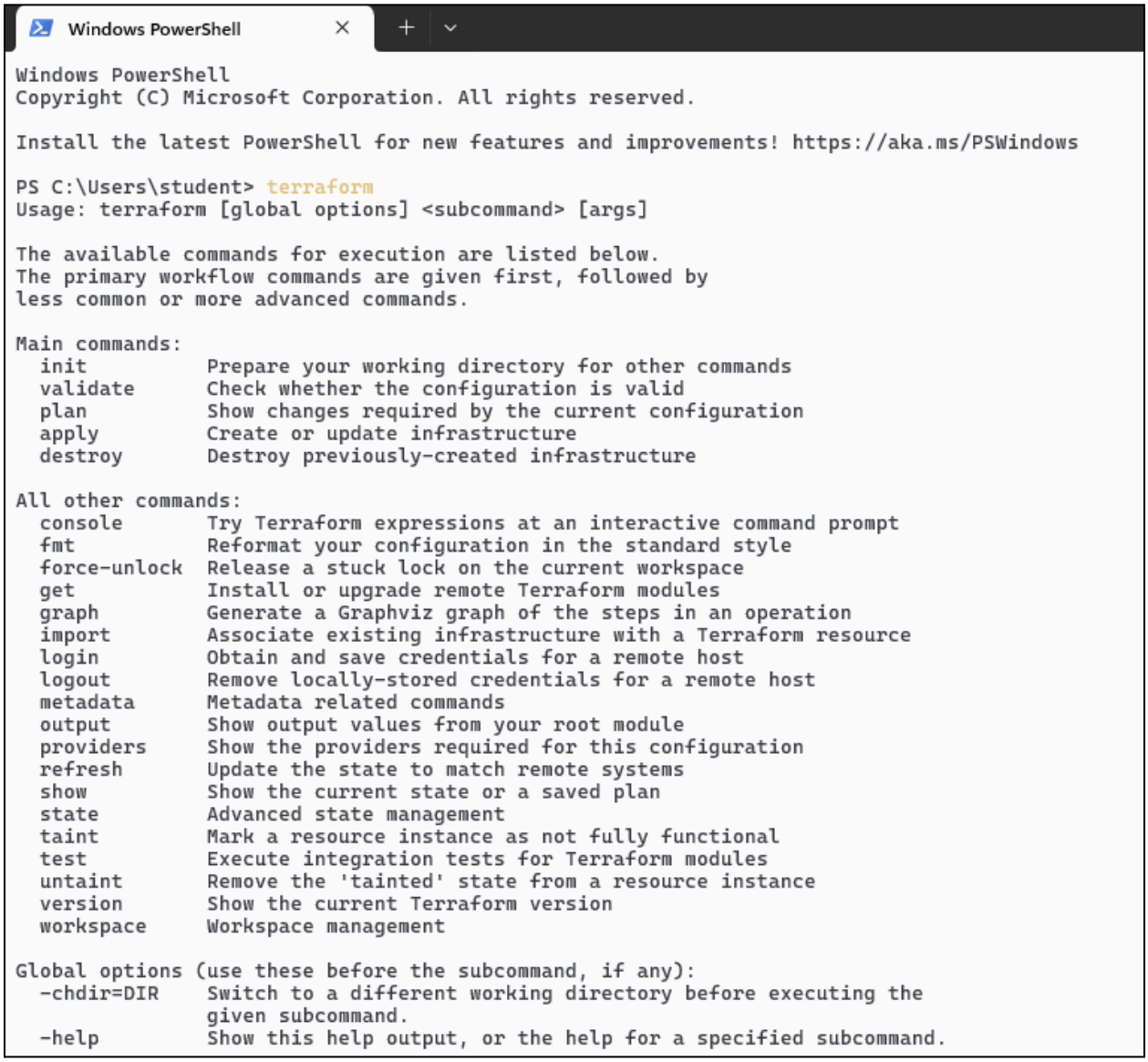
Step 3: Set the System path for Terraform in Environment Variables



Step 4: Open PowerShell with Admin Access



Step 5 : Open Terraform in PowerShell and check its functionality



Name : Sandeshkumar Div : D15C Roll : 61

Aim:Creating docker image using terraform

1) Download and Install Docker Desktop from https://www.docker.com/

Step 1: Check the docker functionality

```
(base) krushikeshsunilshelar@Krushikeshs-MacBook-Air Docker % docker -v
Docker version 27.0.3, build 7d4bcd8
```

Now, create a folder named 'Terraform Scripts' in which we save our different types of scripts which will be further used in this experiment.

Step 2: Firstly create a new folder named 'Docker' in the 'TerraformScripts' folder. Then create a new docker.tf file using nano editor 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 = "unix:///var/run/docker.sock"
}
# Pulls the image
resource "docker_image" "ubuntu" {
  name = "ubuntu:latest"
}
# Create a container
resource "docker_container" "foo" {
  image = docker_image.ubuntu.image_id
  name  = "foo"
  command = ["/bin/bash", "-c", "while true; do sleep 3600; done"]
}
```

```
  UW PICO 5.09

terraform {
  required_providers {
    docker = {
      source  = "kreuzwerker/docker"
      version = "2.21.0"
    }
  }
}

provider "docker" {
  host = "unix:///var/run/docker.sock"
}

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

# Create a container
resource "docker_container" "foo" {
  image = docker_image.ubuntu.image_id
  name  = "foo"
  command = ["/bin/bash", "-c", "while true; do sleep 3600; done"]
}
```

Step 3: Execute Terraform Init command to initialize the resources:

```
(base) krushikeshsunilshelar@Krushikeshs-MacBook-Air Docker % terraform init
Initializing the backend...
Initializing provider plugins...
- Finding kreuzwerker/docker versions matching "2.21.0"...
- Installing kreuzwerker/docker v2.21.0...
- Installed kreuzwerker/docker v2.21.0 (self-signed, key ID BD080C4571C6104C)
Partner and community providers are signed by their developers.
If you'd like to know more about provider signing, you can read about it here:
https://www.terraform.io/docs/cli/plugins/signing.html
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

Step 4: Execute Terraform plan to see the available resources

```
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
      + name             = "foo"
      + network_data     = (known after apply)
      + read_only        = false
      + remove_volumes   = true
      + restart          = "no"
      + rm               = false
      + runtime          = (known after apply)
      + security_opts    = (known after apply)
      + 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.
```

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"

```
Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

docker_container.foo: Creating...
docker_container.foo: Creation complete after 0s [id=f2b095b9576b22cc90eaae6860991144a11cc4a1255f9e1c4c99e5f4ca070857]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
(base) krushikeshsunilshelar@Krushikeshs-MacBook-Air Docker %
```

Docker images, After Executing Apply step:

```
Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
(base) krushikeshsunilshelar@Krushikeshs-MacBook-Air Docker % docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
ubuntu       latest    1a799365aa63   3 weeks ago   101MB
(base) krushikeshsunilshelar@Krushikeshs-MacBook-Air Docker % docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED              STATUS              PORTS     NAMES
d2fa174ea223   1a799365aa63   "/bin/bash -c 'while…"   About a minute ago   Up About a minute             foo
(base) krushikeshsunilshelar@Krushikeshs-MacBook-Air Docker %
```

Docker Validate and Docker providers:

```
(base) krushikeshsunilshelar@Krushikeshs-MacBook-Air Docker % terraform validate
Success! The configuration is valid.

(base) krushikeshsunilshelar@Krushikeshs-MacBook-Air Docker % terraform providers

Providers required by configuration:
.
└── provider[registry.terraform.io/kreuzwerker/docker] 2.21.0

Providers required by state:

    provider[registry.terraform.io/kreuzwerker/docker]

(base) krushikeshsunilshelar@Krushikeshs-MacBook-Air Docker %
```

Step 6: Execute Terraform destroy to delete the configuration, which will automatically delete the Ubuntu Container.
Docker images After Executing Destroy step

Name : Sandeshkumar Div : D15C Roll : 61

```
  # docker_image.ubuntu will be destroyed
  - resource "docker_image" "ubuntu" {
      - id          = "sha256:1a799365aa63eed3c0ebb1c01aa5fd9d90320c46fe52938b03fb007d530d8b02ubuntu:latest" -> null
      - image_id    = "sha256:1a799365aa63eed3c0ebb1c01aa5fd9d90320c46fe52938b03fb007d530d8b02" -> null
      - latest      = "sha256:1a799365aa63eed3c0ebb1c01aa5fd9d90320c46fe52938b03fb007d530d8b02" -> null
      - name        = "ubuntu:latest" -> null
      - repo_digest = "ubuntu@sha256:8a37d68f4f73ebf3d4efafbcf66379bf3728902a8038616808f04e34a9ab63ee" -> null
    }

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

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

docker_container.foo: Destroying... [id=d2fa174ea2233d74813507a43ac8bd8136f623420da4ea251f0e43503b63446c]
docker_container.foo: Destruction complete after 1s
docker_image.ubuntu: Destroying... [id=sha256:1a799365aa63eed3c0ebb1c01aa5fd9d90320c46fe52938b03fb007d530d8b02ubuntu:latest]
docker_image.ubuntu: Destruction complete after 0s

Destroy complete! Resources: 2 destroyed.
(base) krushikeshsunilshelar@Krushikeshs-MacBook-Air Docker % docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
(base) krushikeshsunilshelar@Krushikeshs-MacBook-Air Docker %
```

Name : Sandeshkumar.M.Yadav Div : D15C Roll : 61

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
```

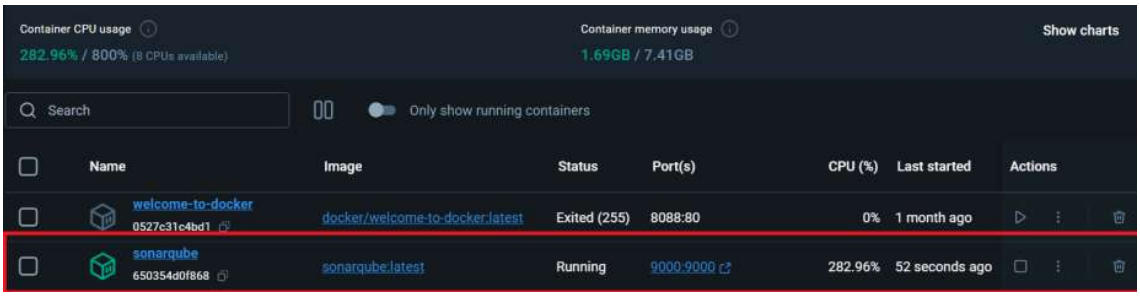
2) Install SonarQube Image:
The command docker pull sonarqube downloads a SonarQube image from Docker's online repository. This image lets you run SonarQube on your system using Docker without needing to install the full SonarQube software manually. It's like getting a ready-to-use version of SonarQube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

```
C:\Users\praja>docker pull sonarqube
Using default tag: latest
latest: Pulling from library/sonarqube
7478e0ac0f23: Pull complete
90a925ab929a: 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
```

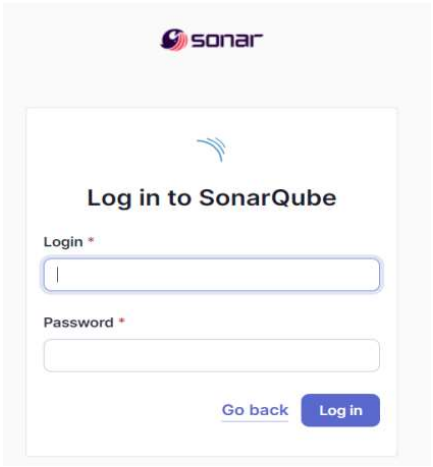
3) Make sure Jenkins is already installed on your system before starting the process. Jenkins will be used to automate tasks, like running SonarQube for code analysis. If Jenkins isn't installed yet, you can download and set it up from the official Jenkins website.

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

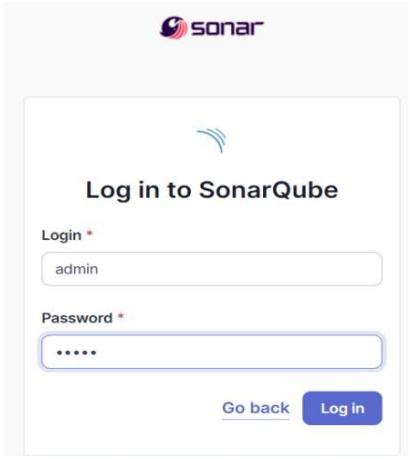
```
C:\Users\praja>docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
650354d0f868ae4ad2d800426080076c604eb09f29b10d4a251aee70f51ce907
```



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



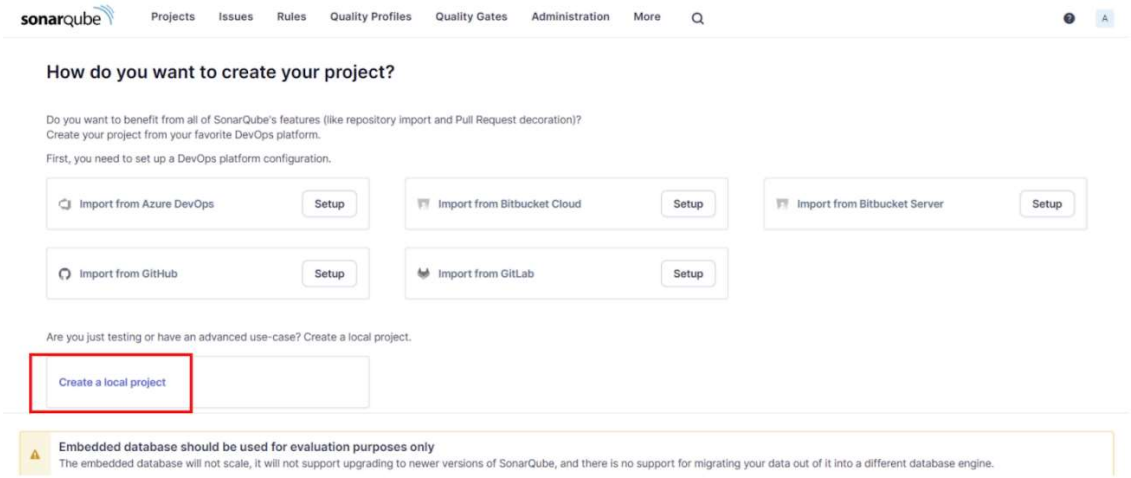
Step 3: On the SonarQube login page, use the default credentials: Username: admin , Password: admin. After logging in, you'll be prompted to change the password. Set a new password and make sure to remember it.



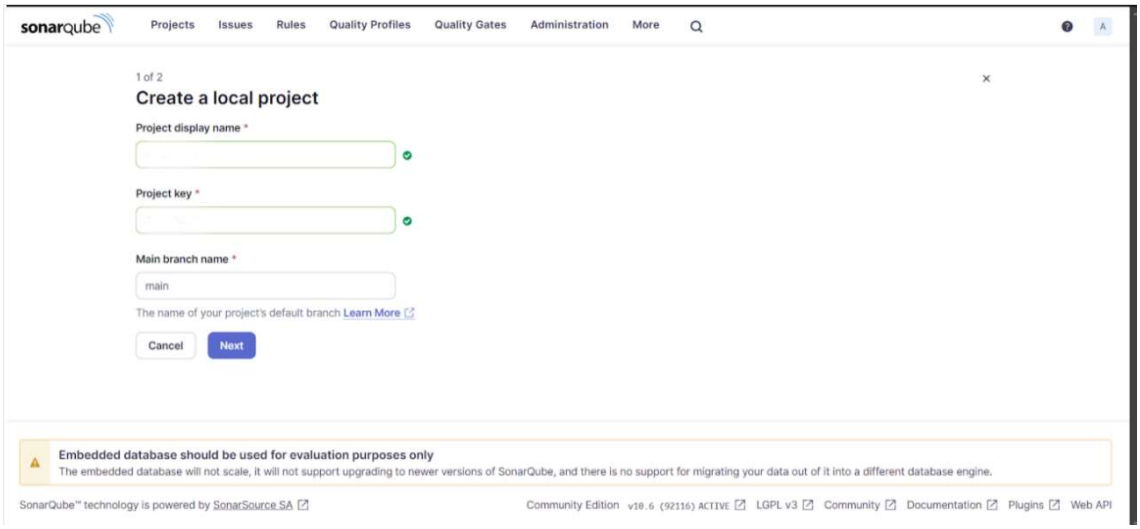
Click on Log in
Click on Update .

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

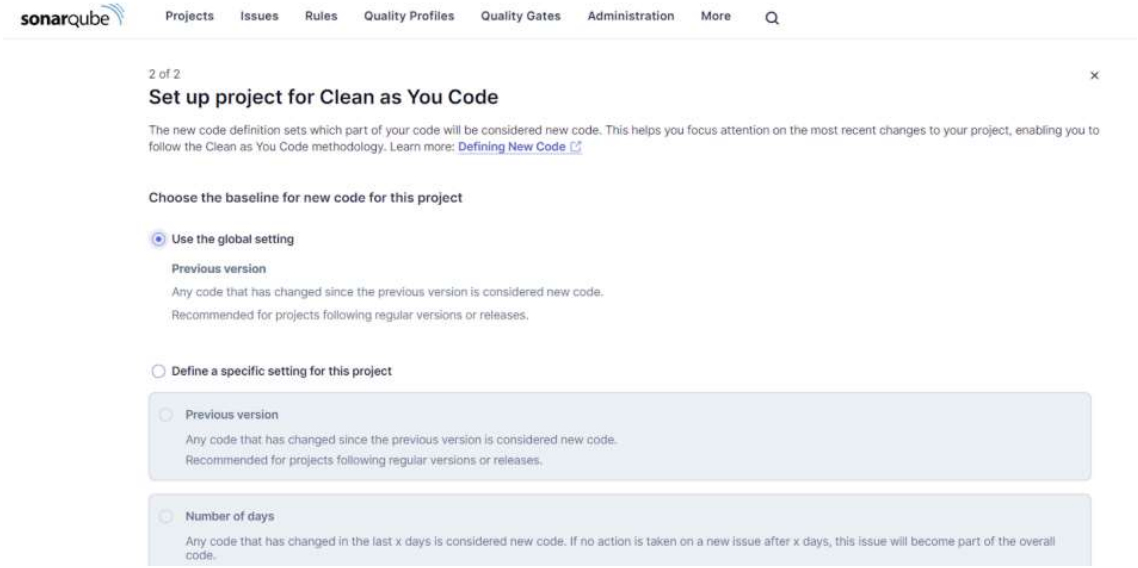
Name : Sandeshkumar.M.Yadav Div : D15C Roll : 61



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

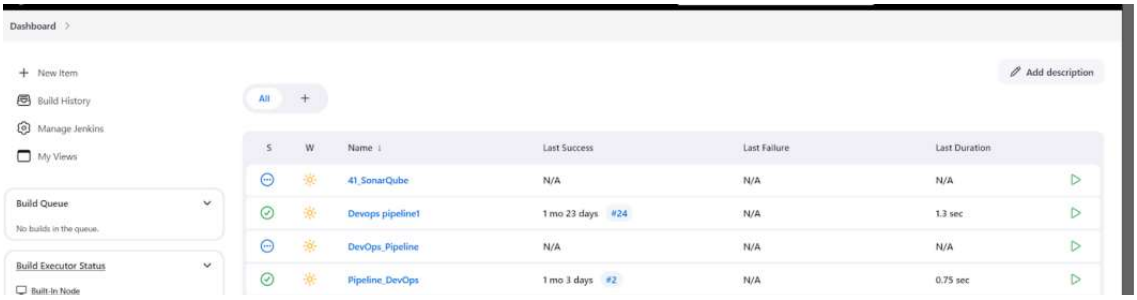


Step 6: Configure the project by providing the necessary settings like choosing the baseline for the new code for the project , then click Create to finalize the setup.

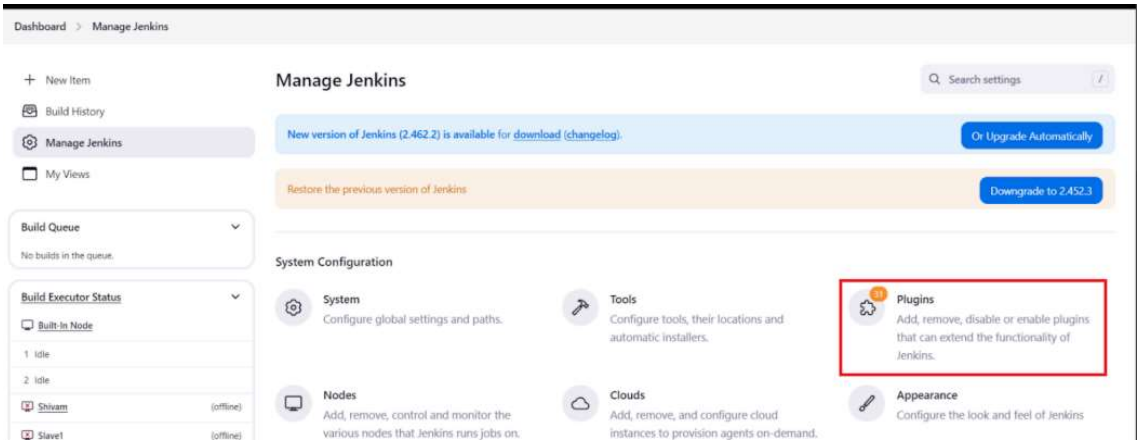


Scroll Down
Click on Create project

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

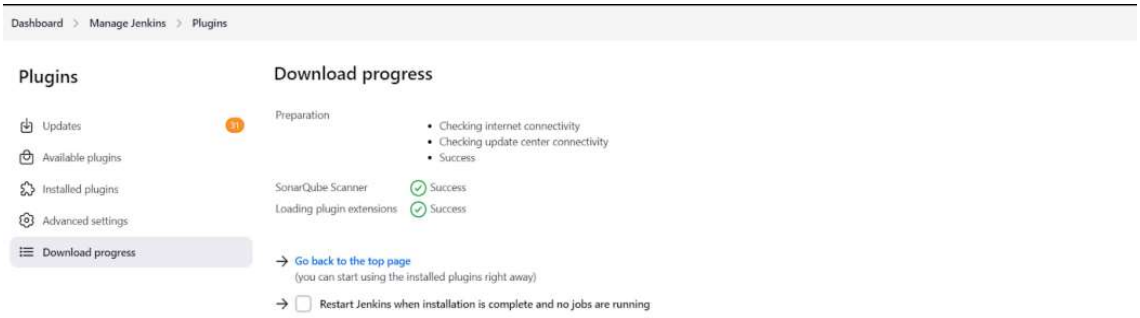


Step 8: Now go to Manage Jenkin then go for Plugins followed by Available plugins search for Sonarqube Scanner where we are going to install it as a plugin



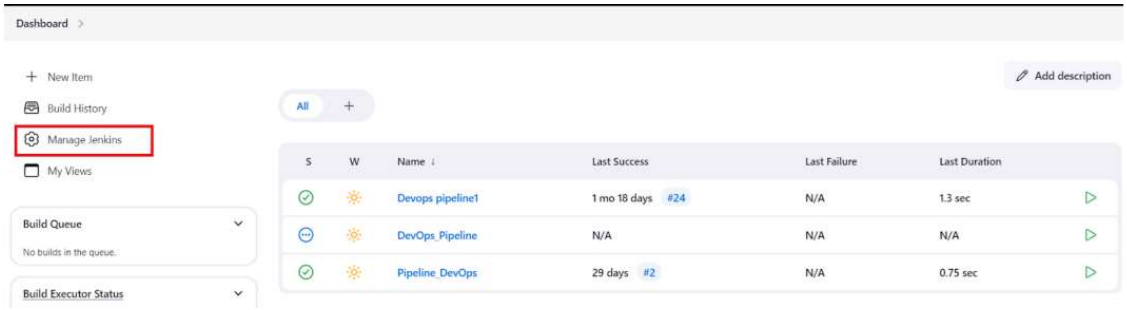
Click on Available Plugins.

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



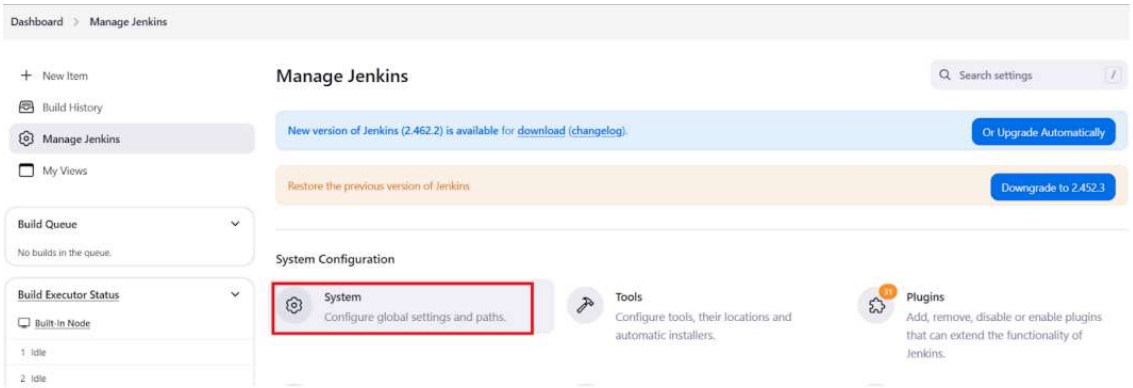
Plugin Installed Successfully

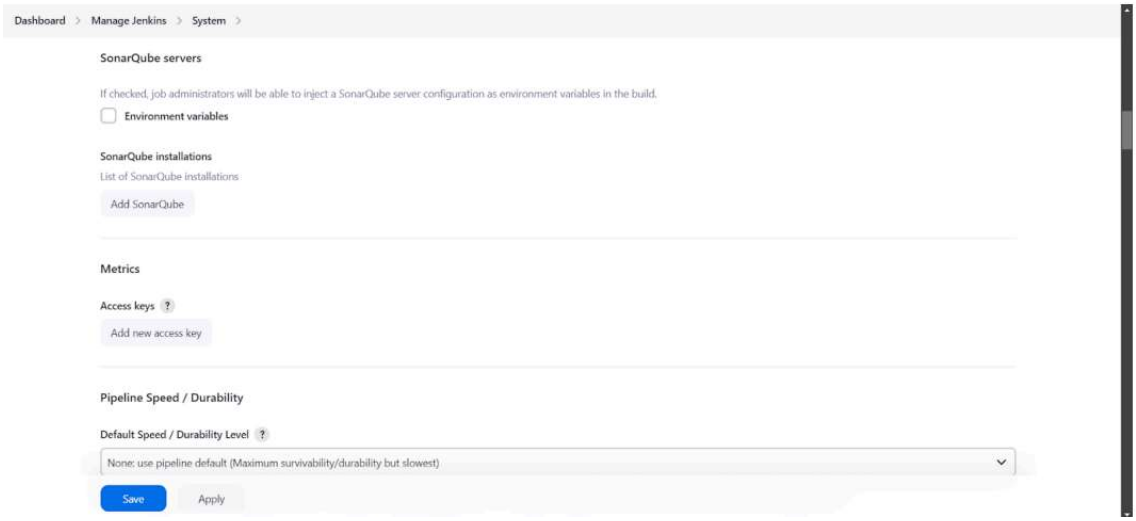
Step 9: In Jenkins, go to Manage Jenkins → System, then find SonarQube servers. Add a new server, and if required, include the authentication token for secure access



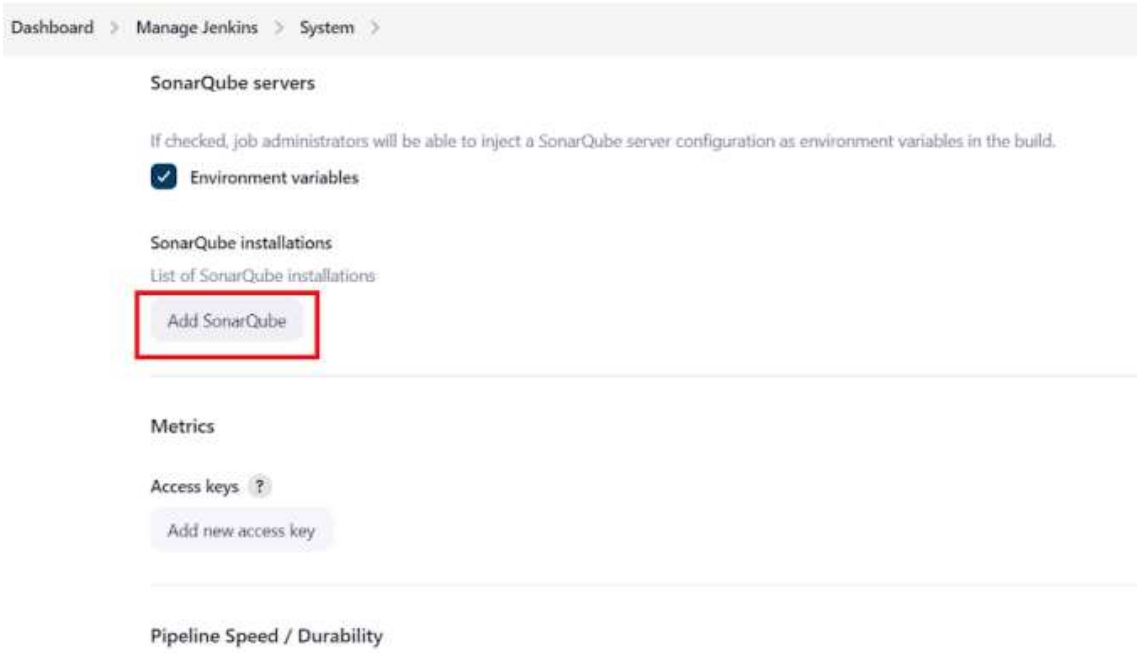
Name : Sandeshkumar.M.Yadav Div : D15C Roll : 61

Go to system

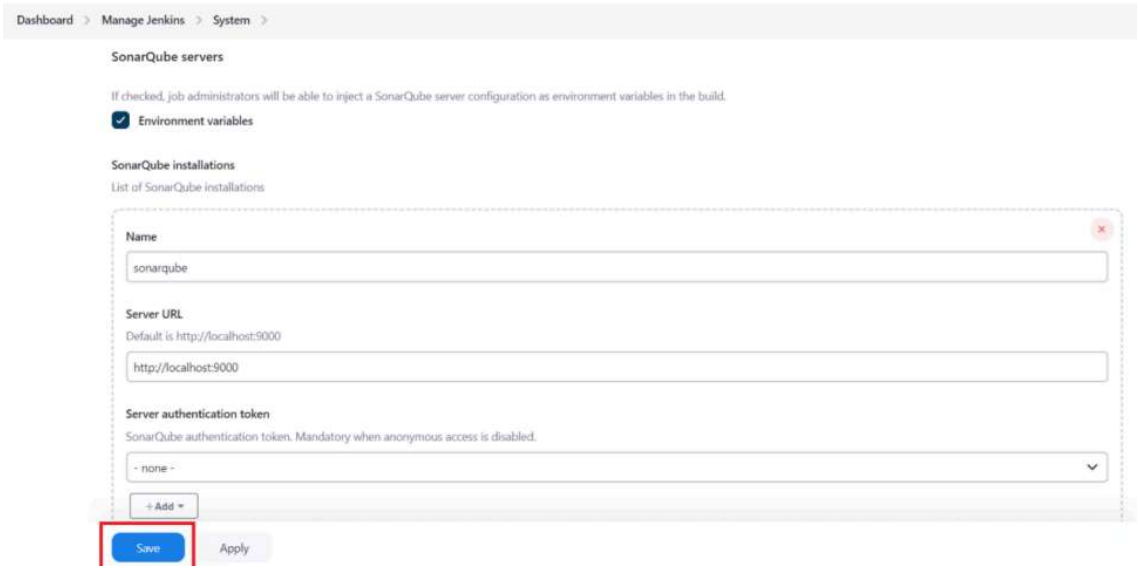




Select Environment Variable and Click on Add Sonar Qube button in order to Add SonarQube Server to Jenkin

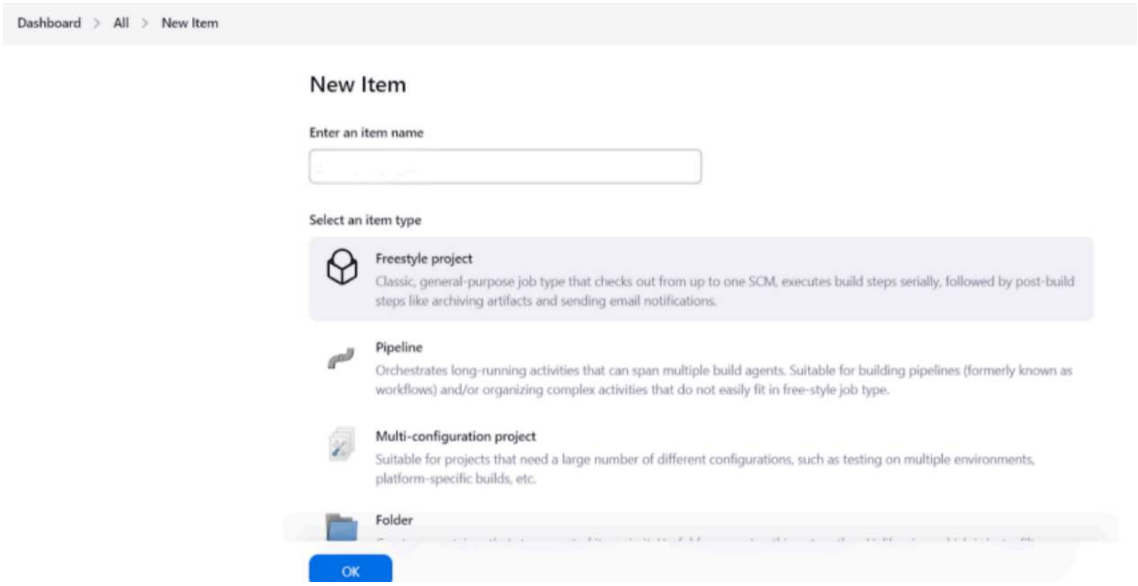


Do the required entries as shown below

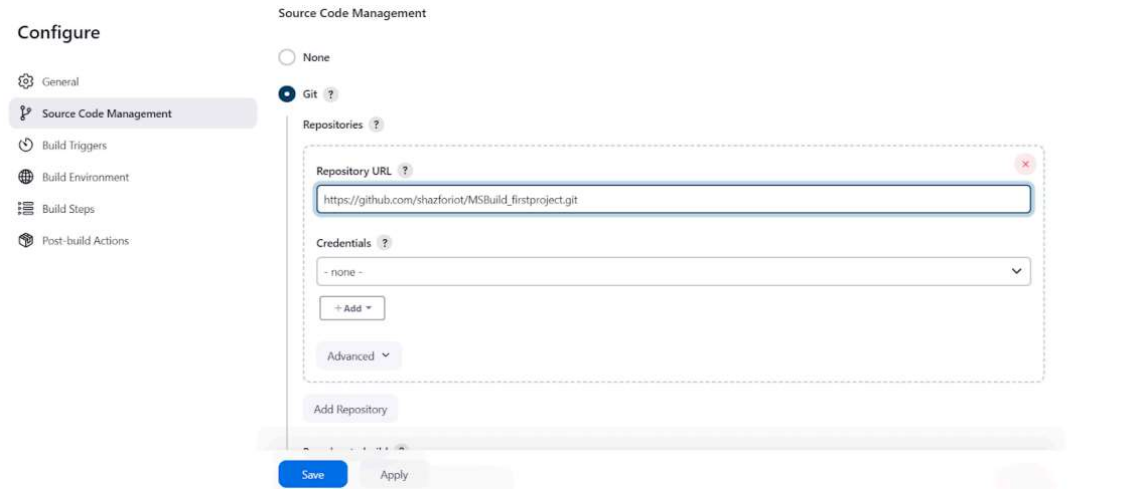


Click on save

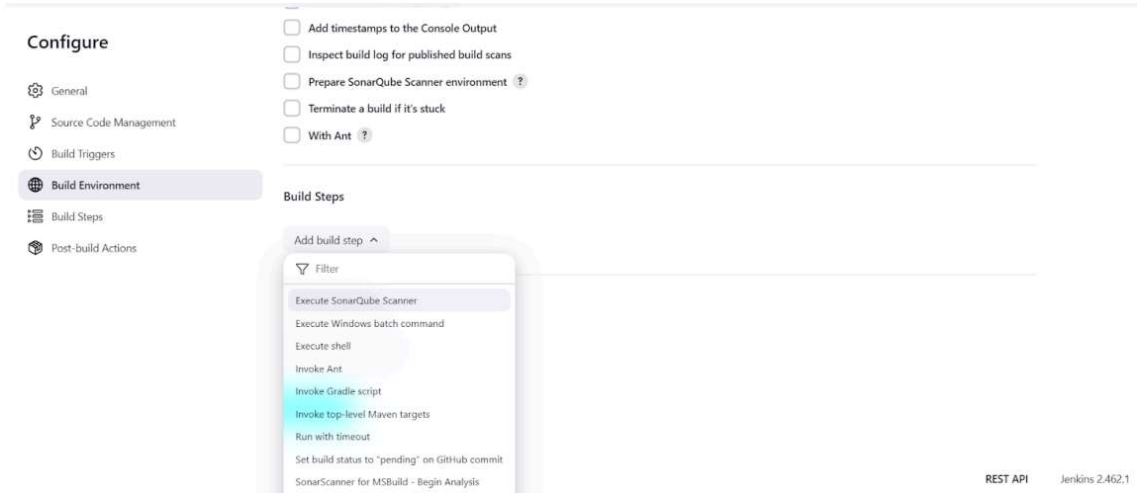
Step 10: After configuration, create a New Item → choose a freestyle project



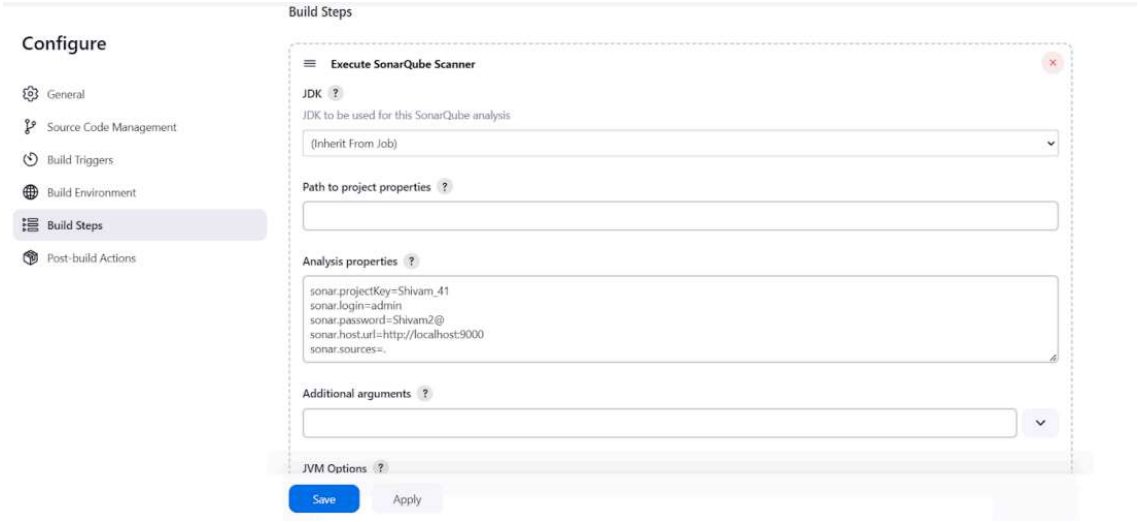
Step 11: Use this github repository in Source Code Management. https://github.com/shazforiot/MSBuild_firstproject. It is a sample hello-world project with no vulnerabilities.



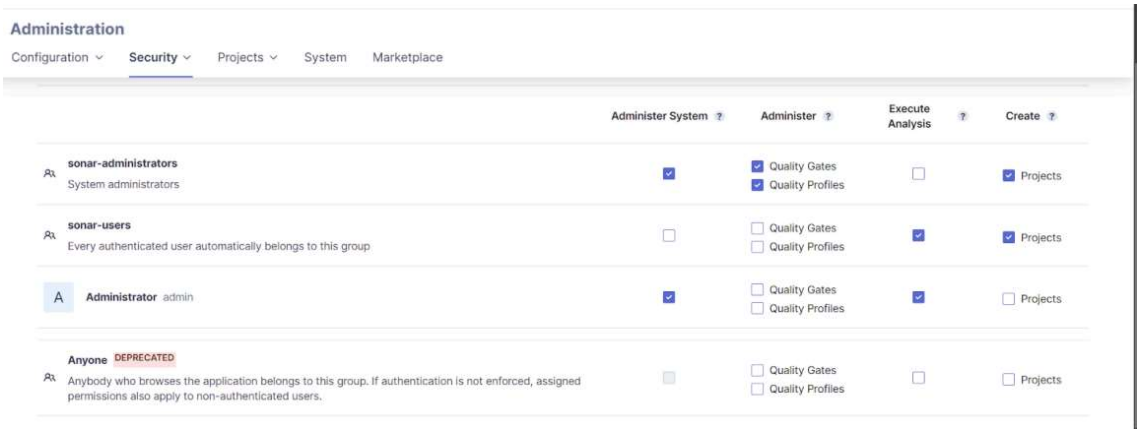
Step 12: Under Build Steps, enter Sonarqube Scanner, enter these Analysis properties. Mention the SonarQube Project Key, Login, Password, Source path and Host URL.



Click on execute sonar scanner

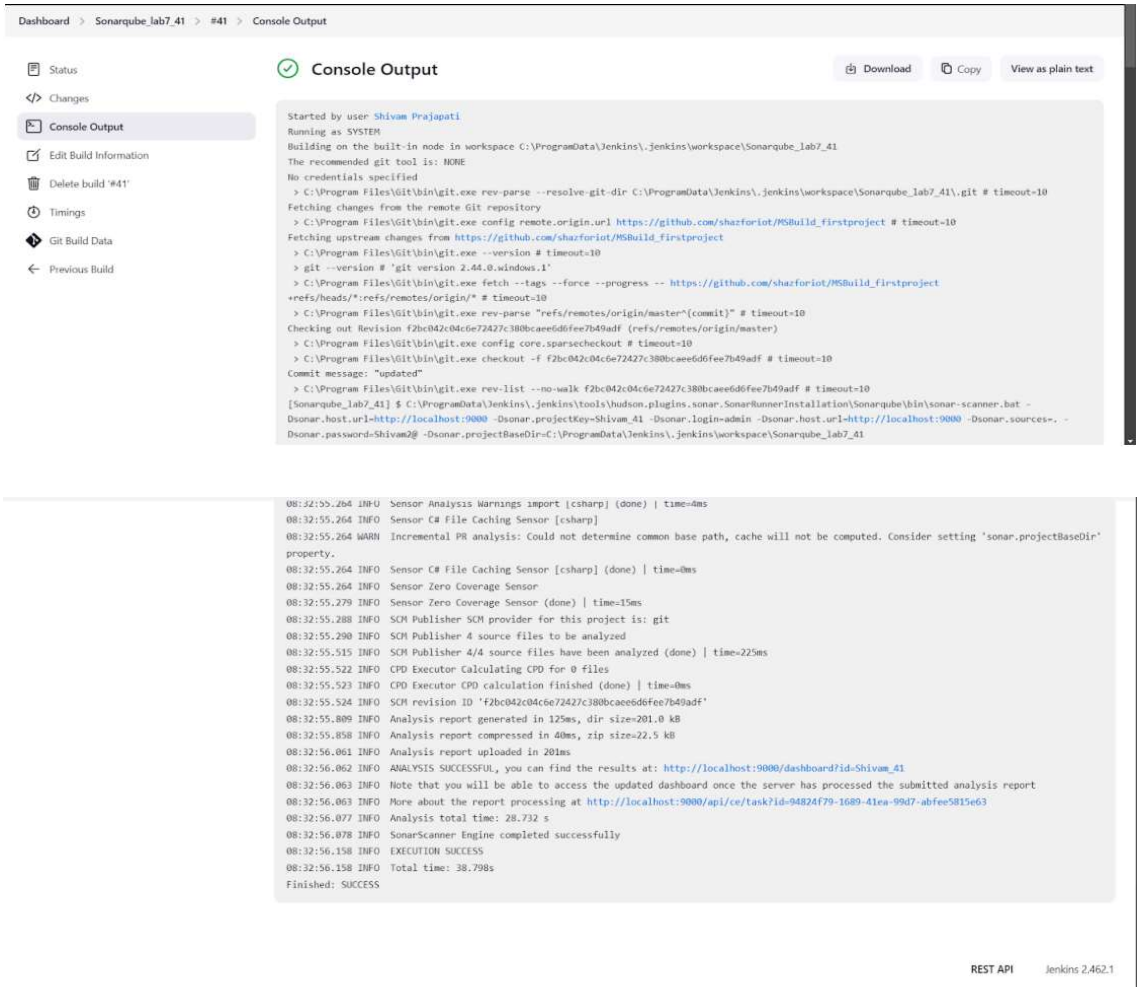


Step 13: Now, you need to grant the local user (here admin user) permissions to Execute the Analysis stage on SonarQube. For this, go to http://loaclhost:/admin/permissions and check the ‘Execute Analysis’ checkbox under Administrator.

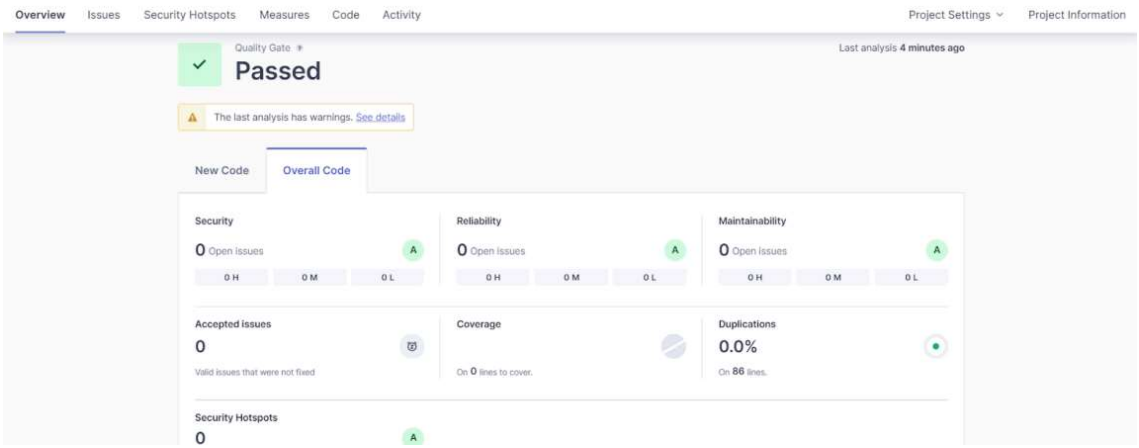


Name : Sandeshkumar.M.Yadav Div : D15C Roll : 61

Step 14: Go back to jenkins. Go to the job you had just built and click on Build Now and 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 successfully integrated Jenkins with SonarQube to perform static application security testing (SAST) on a project. We used Docker to run SonarQube without installing it directly on the system, simplifying the setup process. After configuring Jenkins with the SonarQube Scanner plugin and connecting it to a SonarQube server, we analysed a sample project from GitHub. The analysis demonstrated that the project had no vulnerabilities. This experiment helped us understand how to automate code analysis using Jenkins and SonarQube to ensure the security and quality of the code.

Name : Sandeshkumar.M.Yadav **Div : D15C** **Roll : 61**

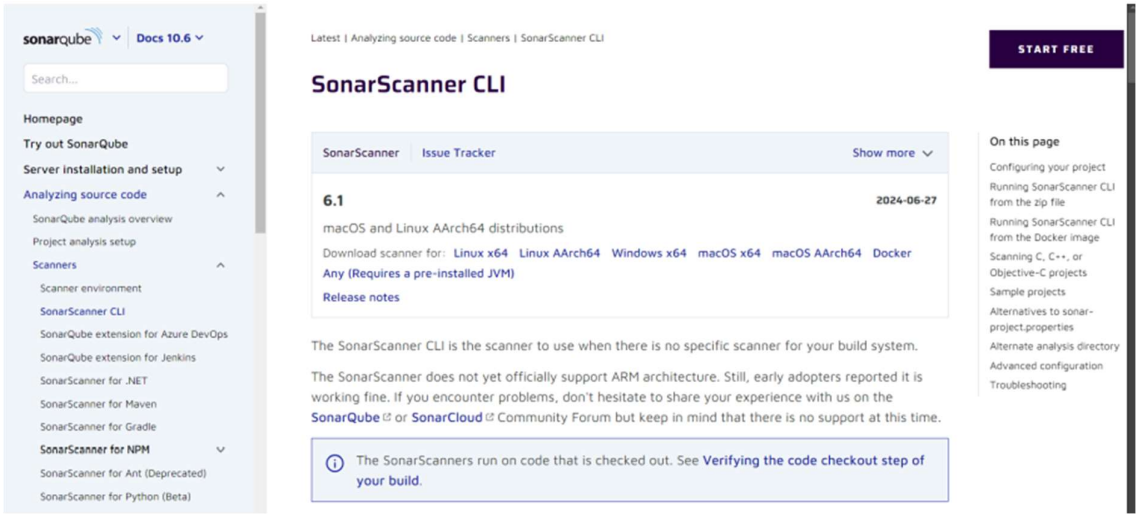
Experiment No: 8

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



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

```
C:\Users\Sandesh>docker -v
Docker version 27.2.0, build 3ab4256
```

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:bb444c58c1e04d8a147a3bb12af941c57e0100a5b21d10e599384d59
Status: Downloaded newer image for sonarqube:latest
docker.io/library/sonarqube:latest

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

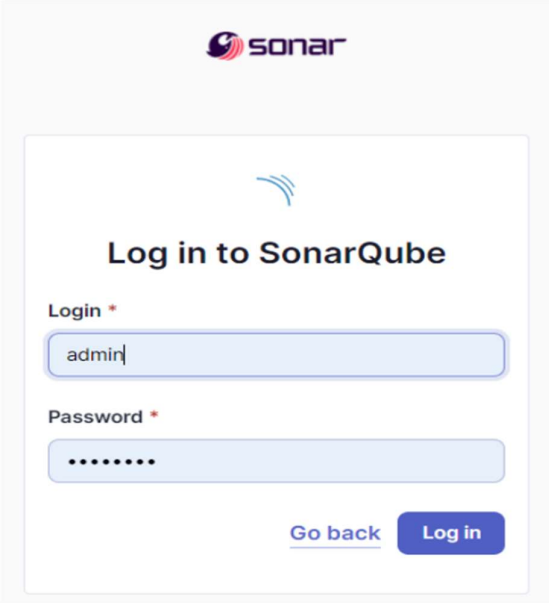
Step 4: Keep Jenkins installed on your system.

EXPERIMENT STEPS:

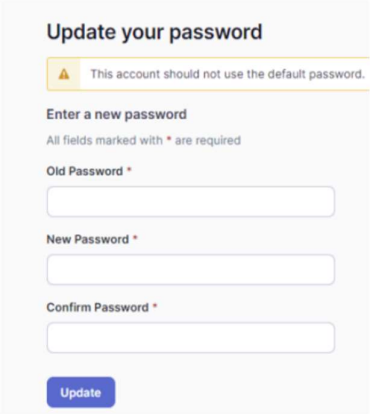
Step1: Run SonarQube image docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest .This command will run the SonarQube image that was just installed using docker.

```
:\Users\Student>docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
3330c33cd961d8d659f362c5f62c6cd1ff87f31ec99da134350b9b419370561
```

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

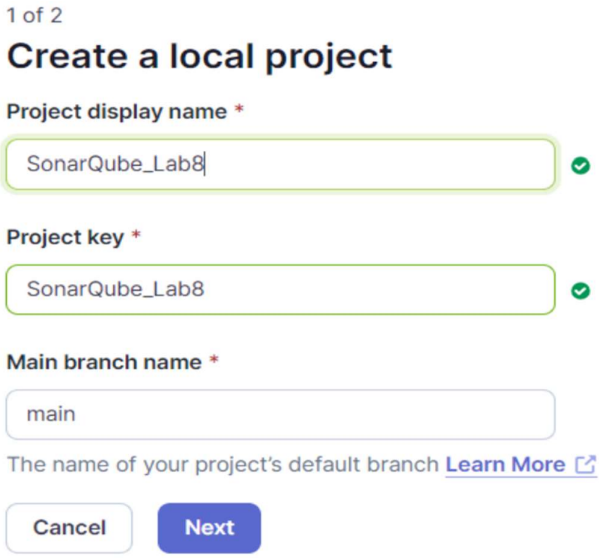


Step 3: On this interface, login with username = ‘admin’ and password = ‘admin’. Once logged in successfully, SonarQube will ask you to reset this password. Reset it and remember this password.



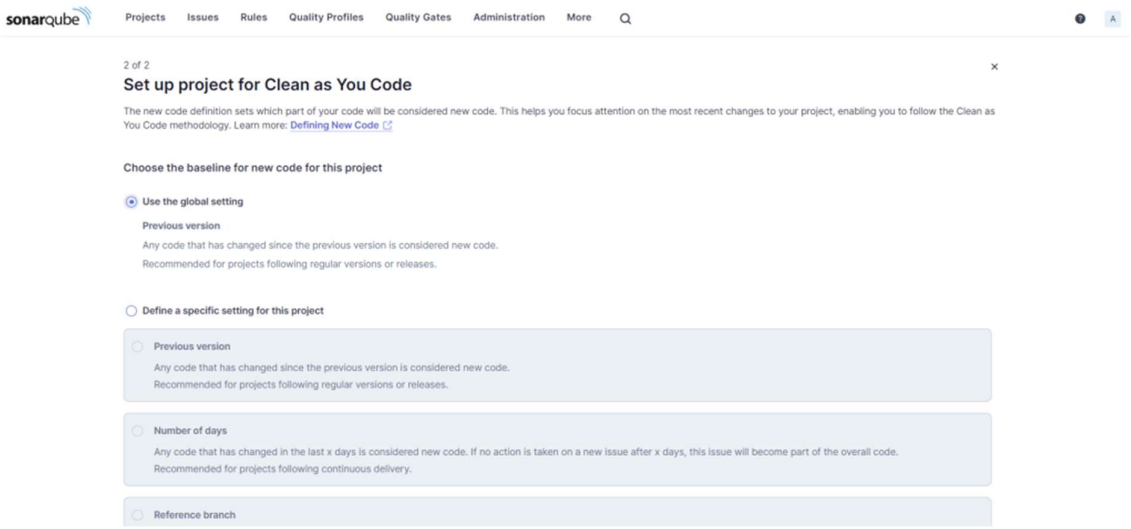
Step 4: After changing the password, you will be directed to this screen. Click on Create a Local Project. Give the project a display name and project key

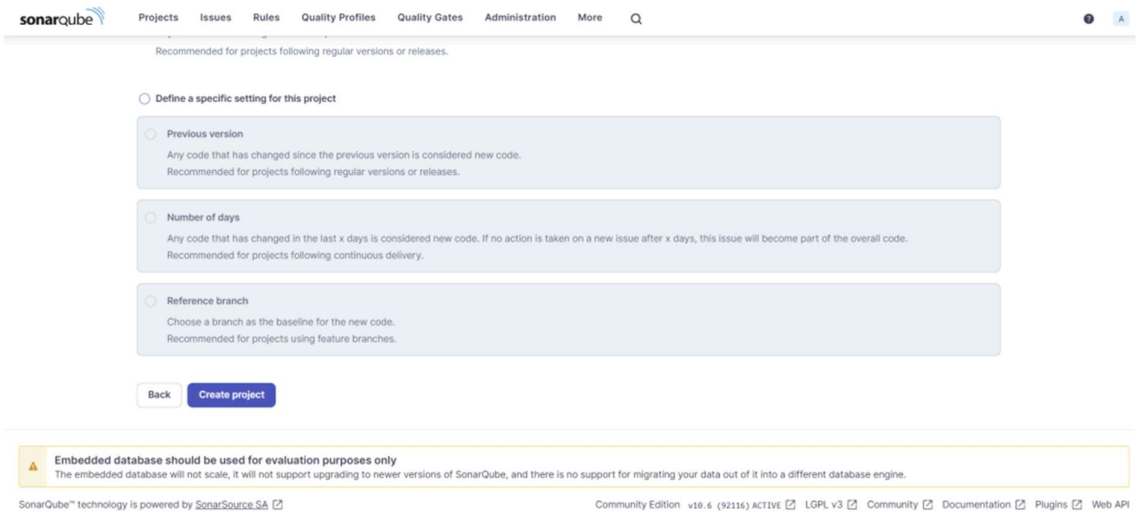
Click on Create Project



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 .

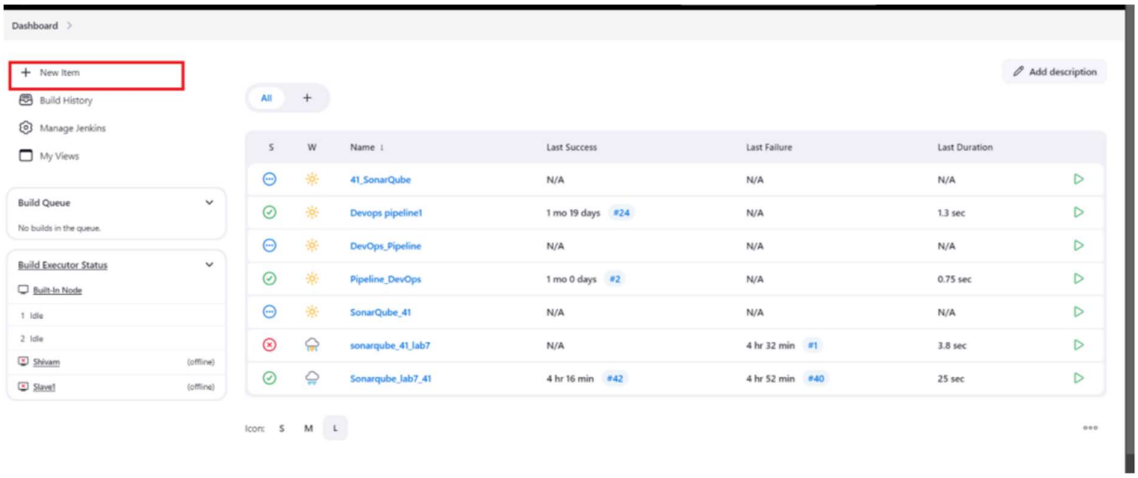




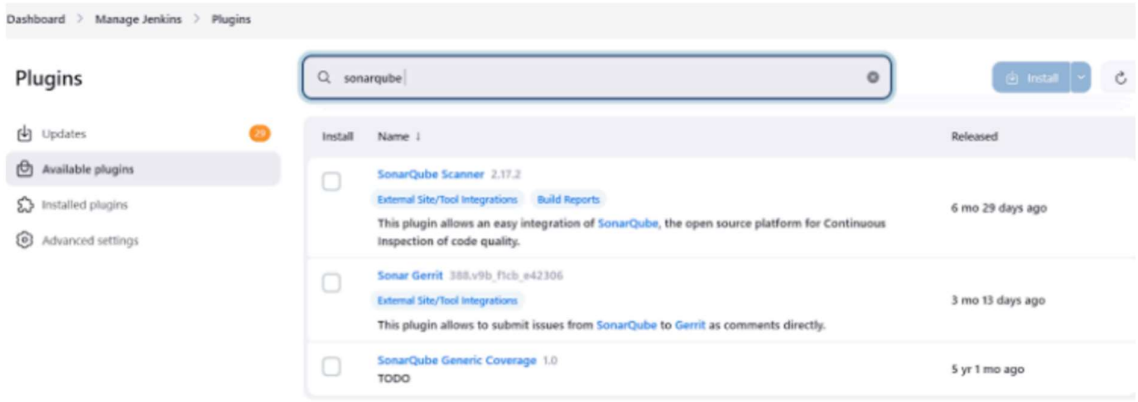
Click on Create

Project is created

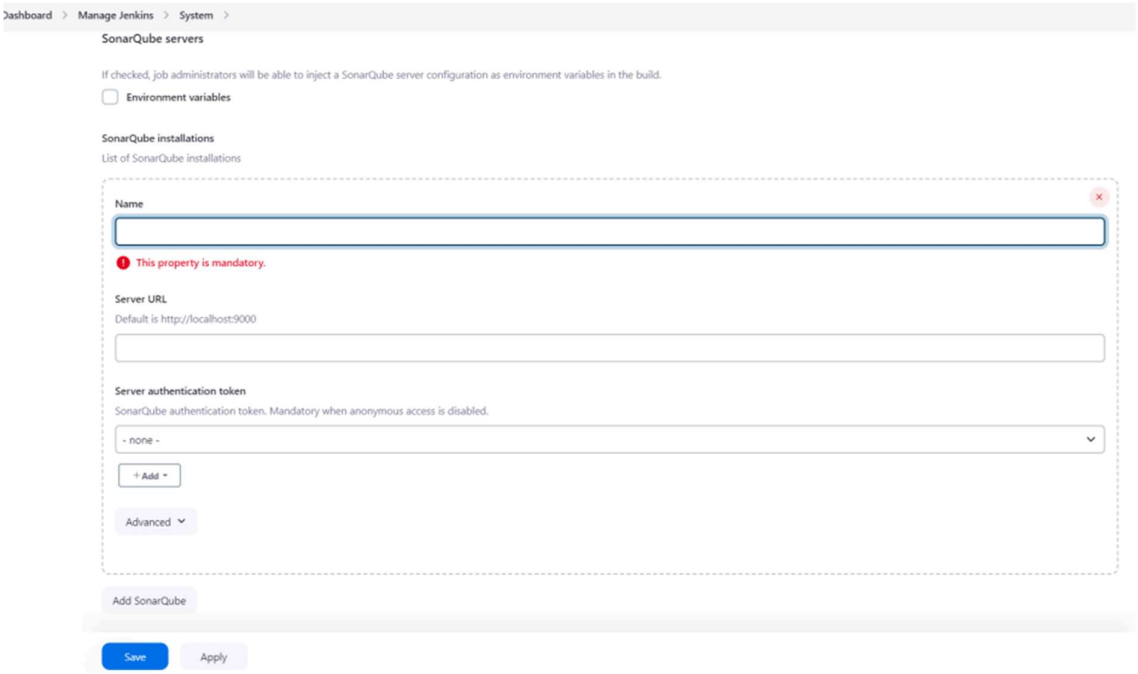
Step 5: Open Jenkins on whichever port it is installed. (http://loaclhost:). Go to the new item



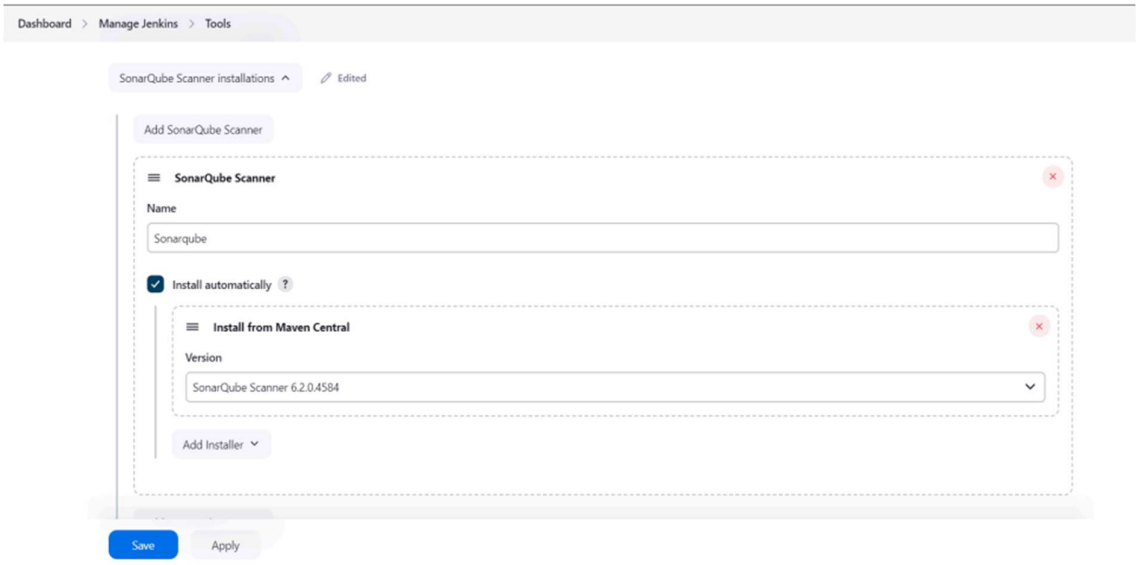
Step 6: Go to manage jenkins →available plugins then Search for Sonarqube Scanner for Jenkins and install it

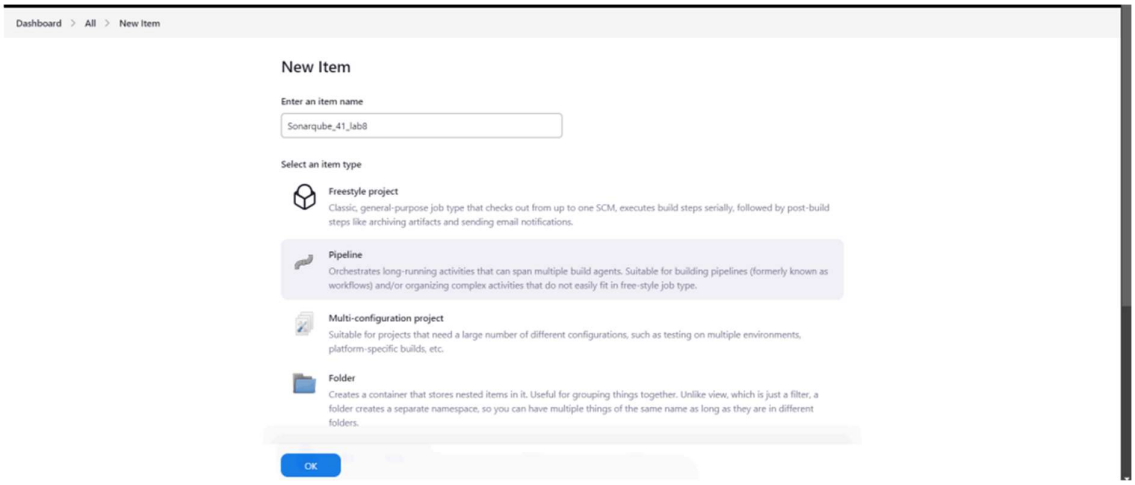


Step 7: Now, go to Manage Jenkins → System. Under Sonarqube servers, add a server. Add server authentication token if needed.



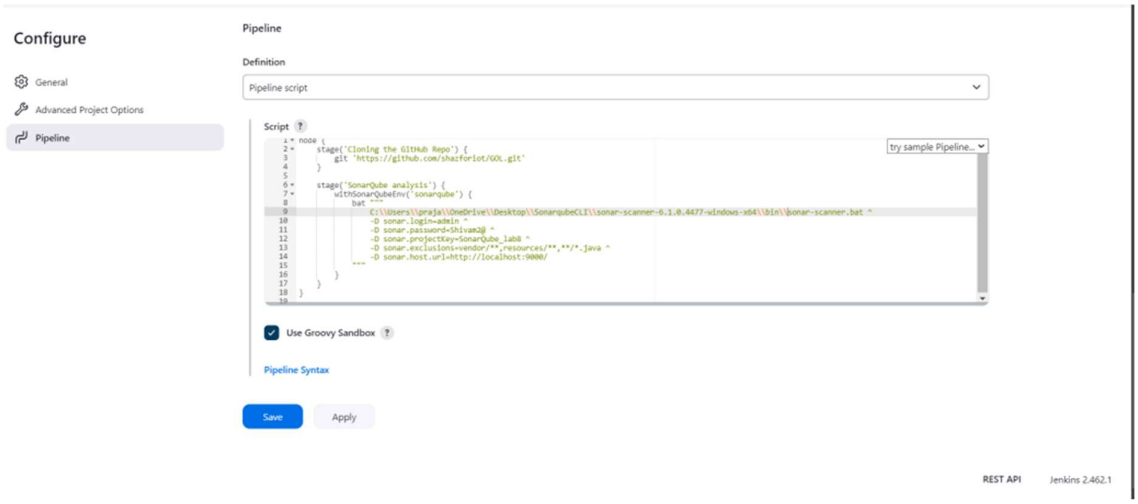
Step 8: Go to Manage Jenkins → Tools. Go to SonarQube scanner, choose the latest configuration and choose to install automatically.



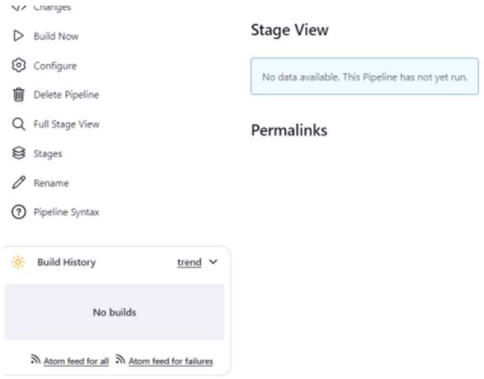


Step 10: Under Pipeline script, enter the following:

```
node {
stage('Cloning the GitHub Repo') {
git 'https://github.com/shazforiot/GOL.git
}
stage('SonarQube analysis') {
withSonarQubeEnv('sonarqube') {
bat """

C:\\Users\\Sandesh\\Downloads\\SonarqubeCLI\\sonar-scanner-6.1.0.4477-windows-x64\\bi
n\\sonar-scanner.bat ^
-D sonar.login=admin ^
-D sonar.password=Sandesh2@ ^
-D sonar.projectKey=SonarQube_Lab8 ^
-D sonar.exclusions=vendor/**,resources/**,**/*.java ^
-D sonar.host.url=http://localhost:9000/
"""
}
}
}
```

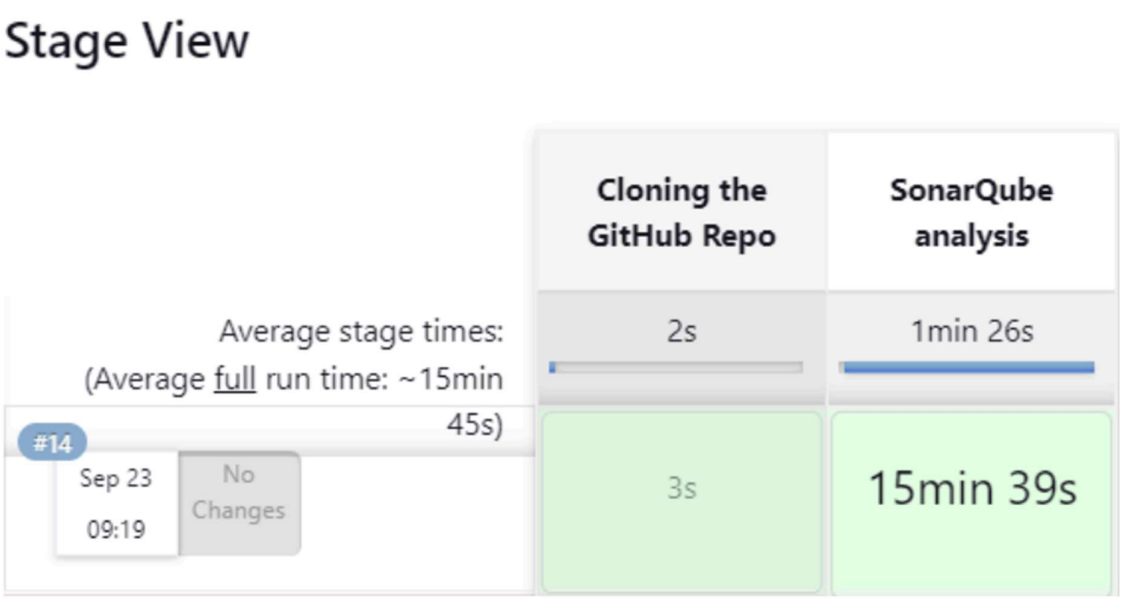


Click on save.



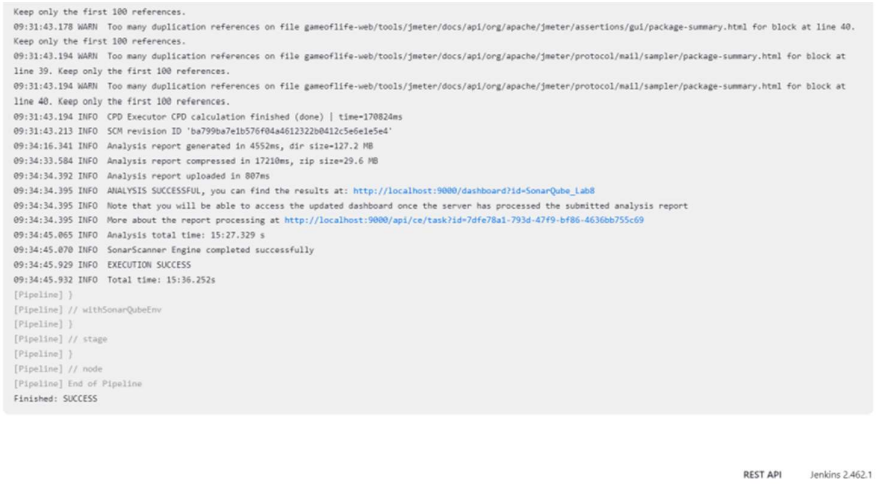
This is a Java sample project with many repetitive sections and coding issues that SonarQube will be able to detect during analysis.

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



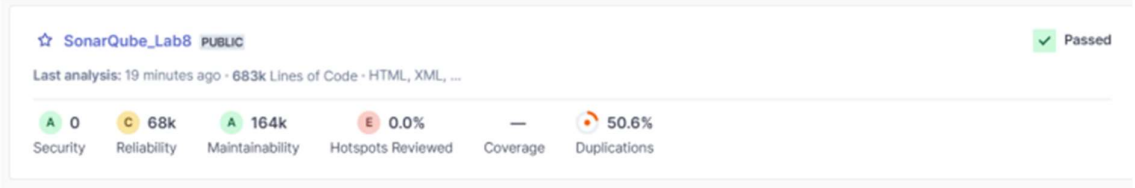
The problem was C:\windows\system32 was not there so we need to add in our environment variable .

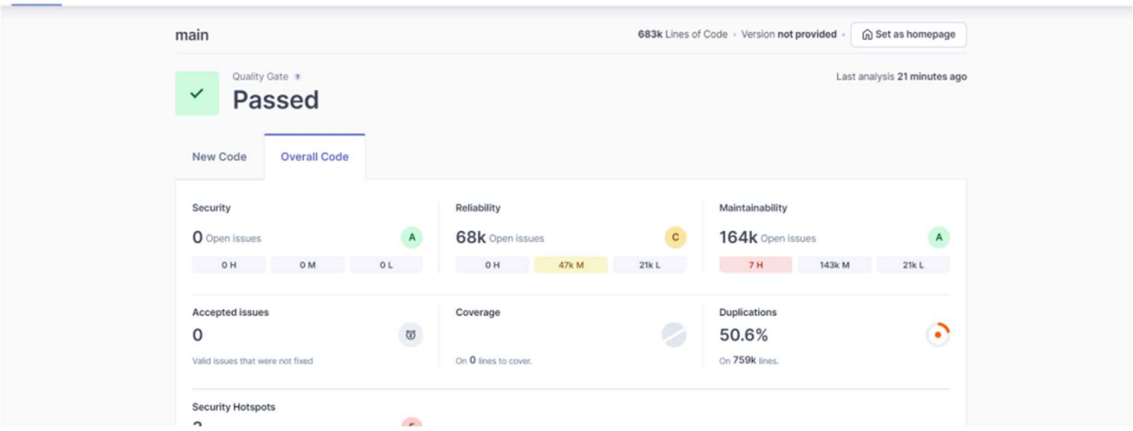
Now Check the console output once



Successfully BUILD

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

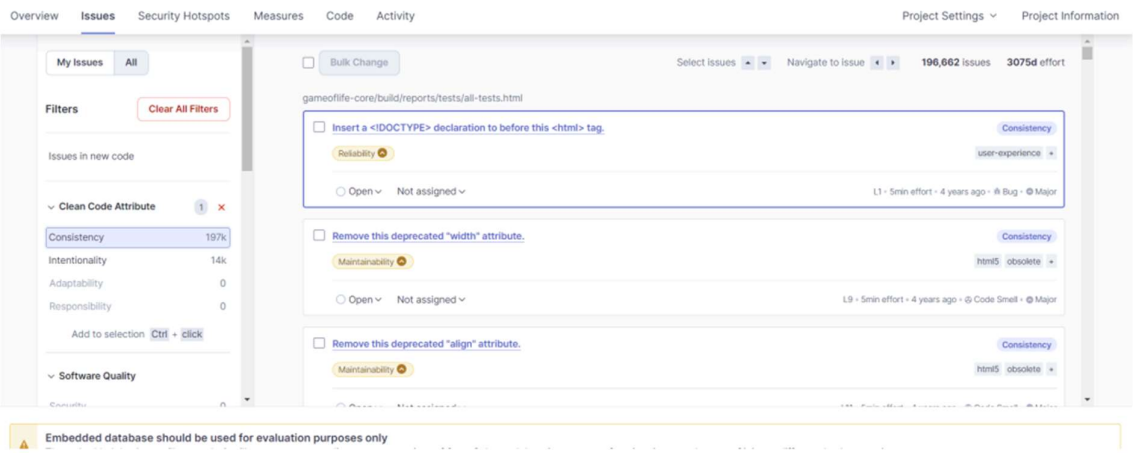




Under different options on the navbar , we can check all the issues with the code.

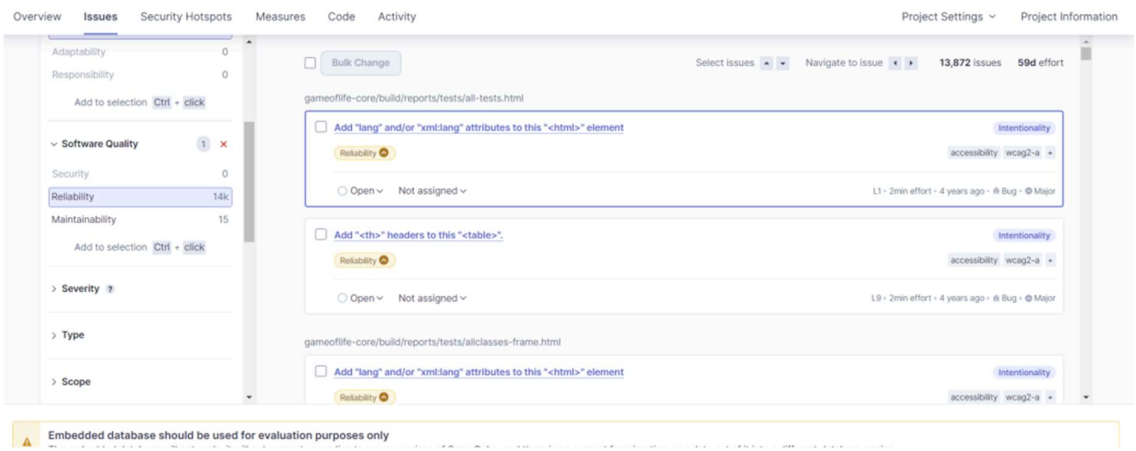
UNDER ISSUES:

1) Consistency

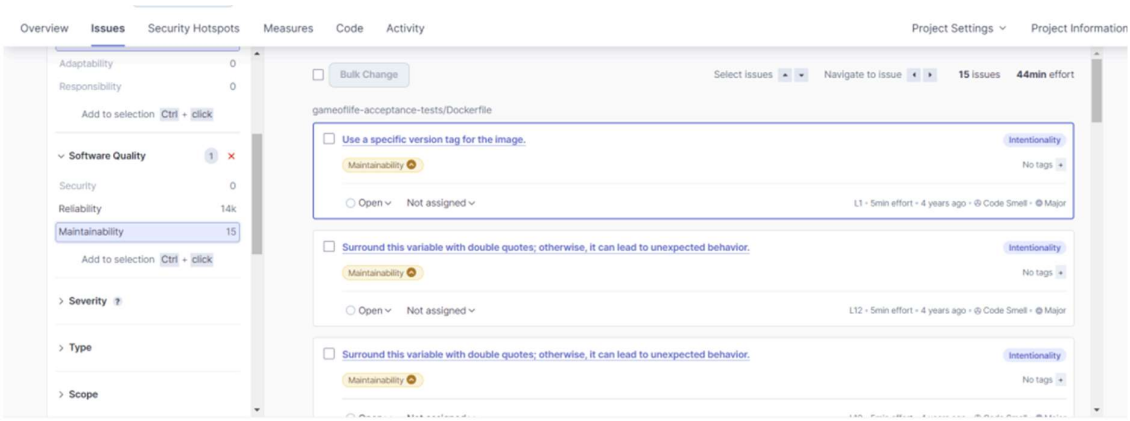


2) Reliability

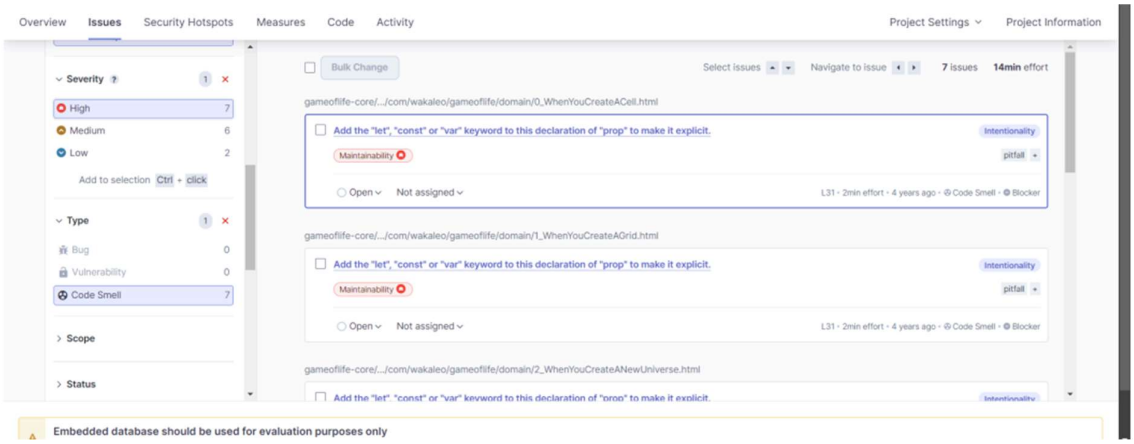
Name : Sandeshkumar.M.Yadav **Div : D15C** **Roll : 61**



3) Maintainability

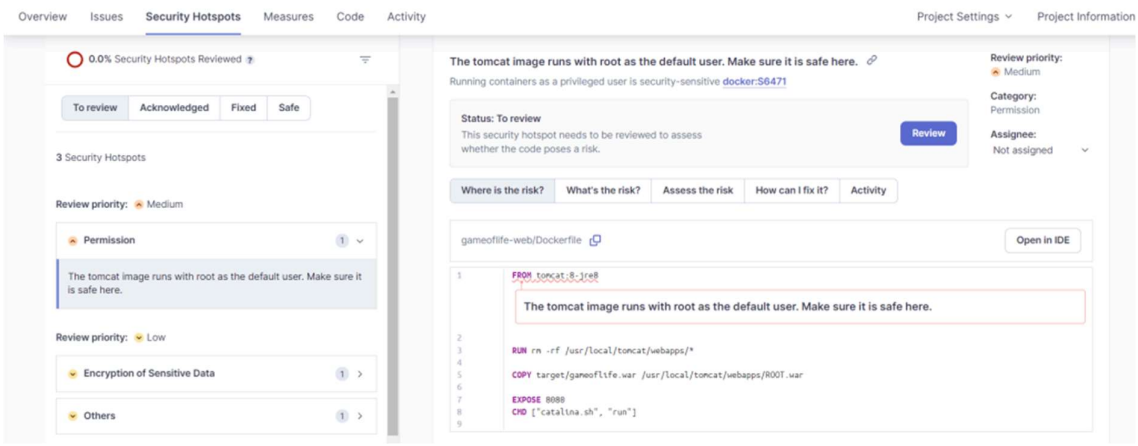


4) Severity

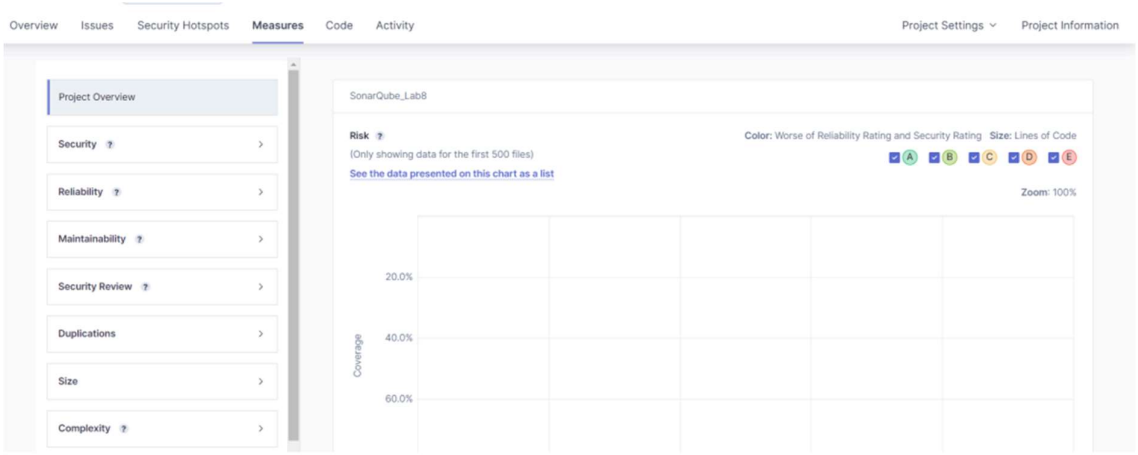


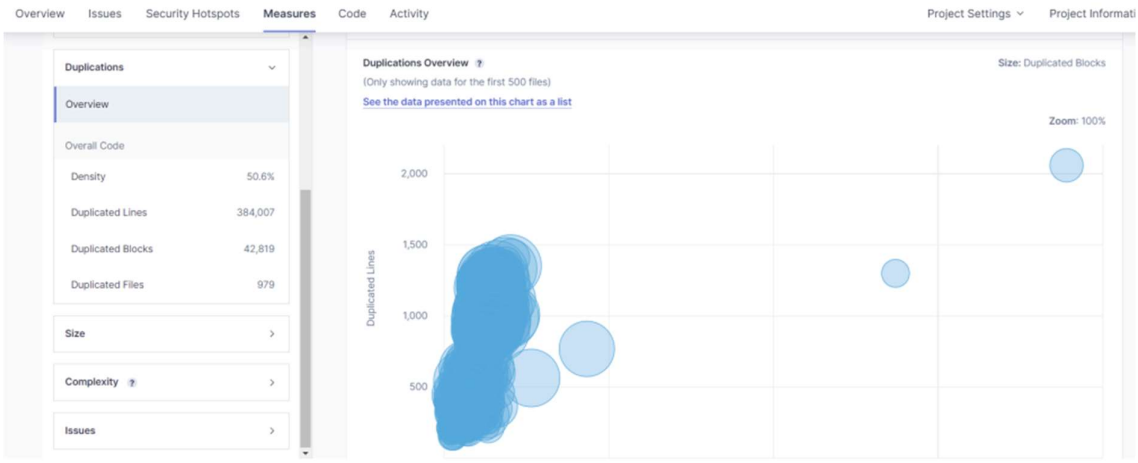
UNDER SECURITY HOTSPOT:

Name : Sandeshkumar.M.Yadav **Div : D15C** **Roll : 61**



UNDER MEASURES:





CONCLUSION:

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

Theory:

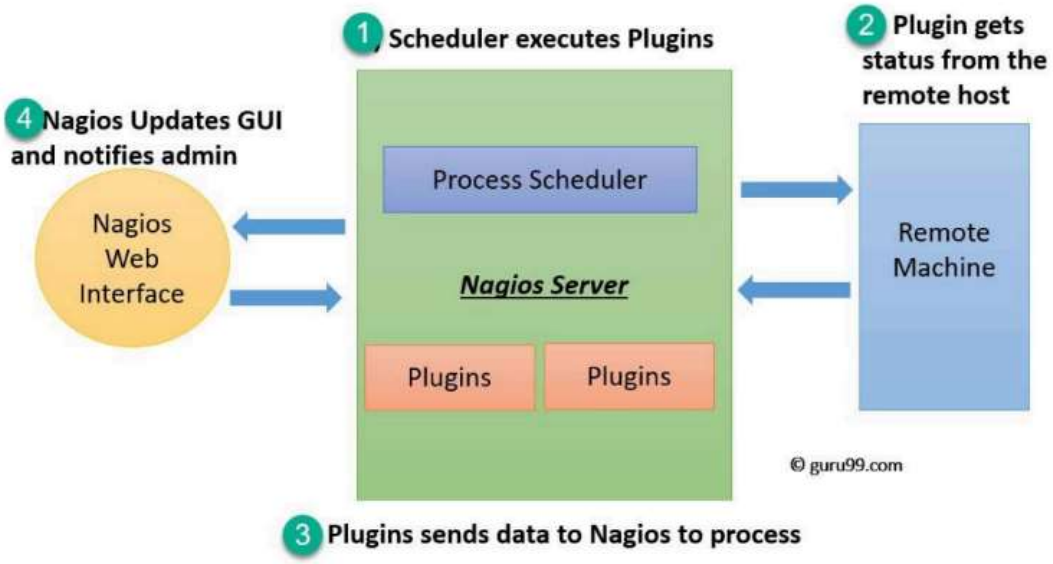
What is Nagios?
Nagios is an open-source software for continuous monitoring of systems, networks, and infrastructures. It runs plugins stored on a server that is connected with a host or another server on your network or the Internet. In case of any failure, Nagios alerts about the issues so that the technical team can perform the recovery process immediately.
Nagios is used for continuous monitoring of systems, applications, service and business processes in a DevOps culture.
Why We Need Nagios tool?
Here are the important reasons to use Nagios monitoring tool:

- Detects all types of network or server issues
- Helps you to find the root cause of the problem which allows you to get the permanent solution to the Problem
- Active monitoring of your entire infrastructure and business processes
- Allows you to monitor and troubleshoot server performance issues
- Helps you to plan for infrastructure upgrades before outdated systems create failures
- You can maintain the security and availability of the service
- Automatically fix problems in a panic situation Features of Nagios Following are the important features of Nagios monitoring tool:
- Relatively scalable, Manageable, and Secure
- Good log and database system
- Informative and attractive web interfaces
- Automatically send alerts if condition changes
- If the services are running fine, then there is no need to do check that host is an alive
- Helps you to detect network errors or server crashes
- You can troubleshoot the performance issues of the server.
- The issues, if any, can be fixed automatically as they are identified during the monitoring process
- You can monitor the entire business process and IT infrastructure with a single pass
- The product's architecture is easy to write new plugins in the language of your choice
- Nagios allows you to read its configuration from an entire directory which helps you to decide how to define individual files
- Utilises topology to determine dependencies
- Monitor network services like HTTP, SMTP, HTTP, SNMP, FTP, SSH, POP, etc.
- Helps you to define network host hierarchy using parent hosts
- Ability to define event handlers that runs during service or host events for proactive problem resolution
- Support for implementing redundant monitoring hosts

Name : Sandeshkumar.M.Yadav **Div : D15C** **Roll : 61**

Nagios Architecture :

Nagios is a client-server architecture. Usually, on a network, a Nagios server is running on a host, and plugins are running on all the remote hosts which should be monitored.

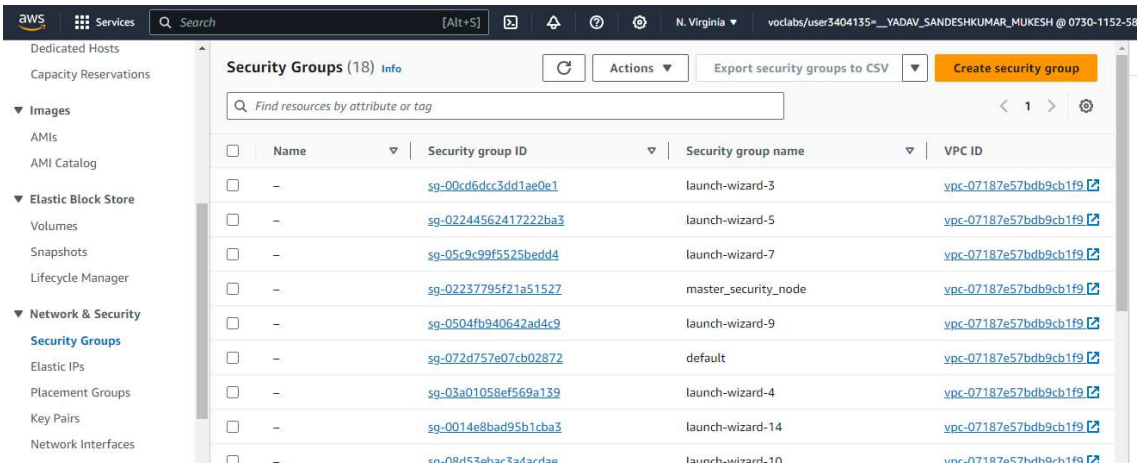


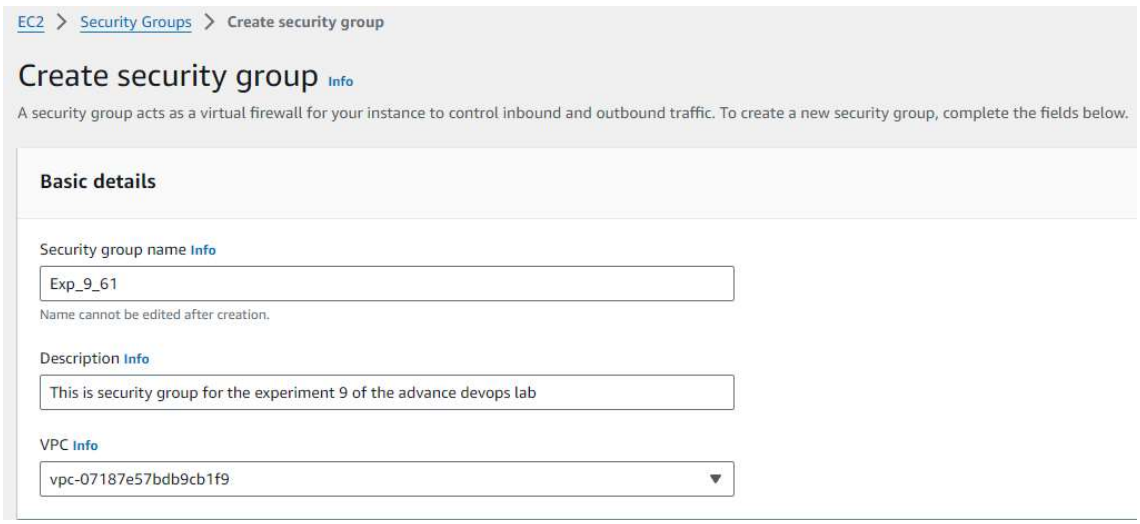
1. The scheduler is a component of the server part of Nagios. It sends a signal to execute the plugins at the remote host.
2. The plugin gets the status from the remote host
3. The plugin sends the data to the process scheduler
4. The process scheduler updates the GUI and notifications are sent to admins.

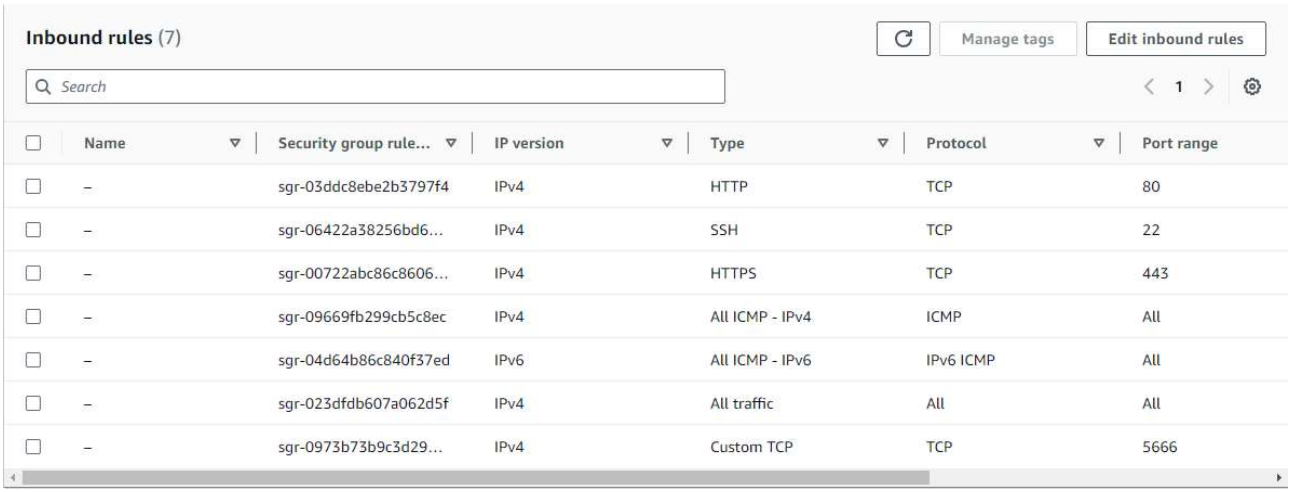
Name : Sandeshkumar.M.Yadav **Div : D15C** **Roll : 61**

Steps :

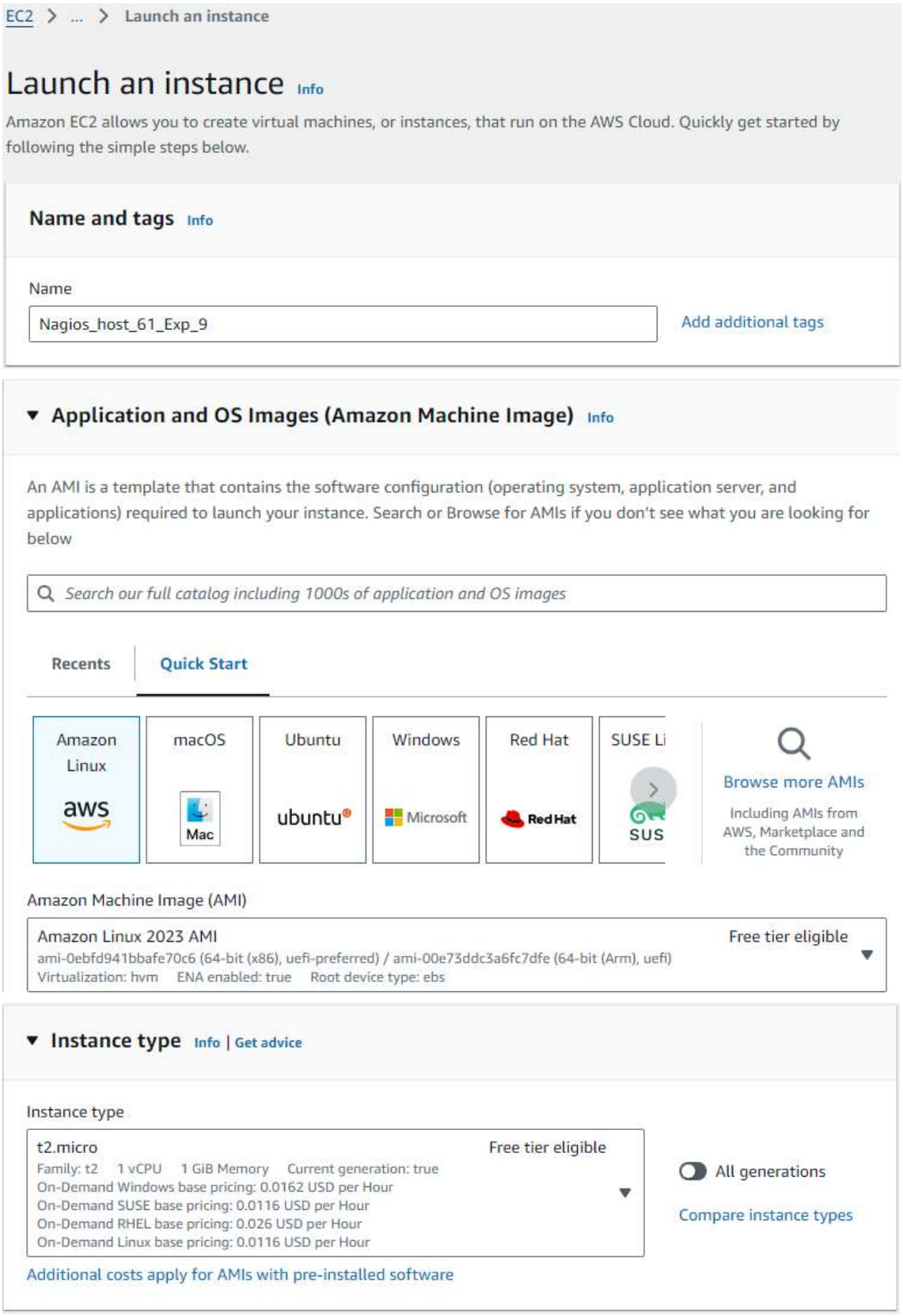
Step 1: : Login to your AWS account Personal / Academy. Click on EC2 instance then click on Create Security Group. Give the name as Nagios and any description and add the following inbounds rules.

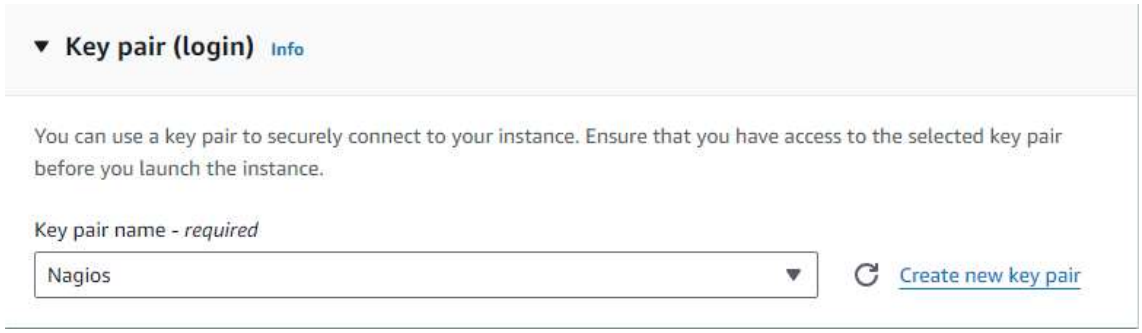




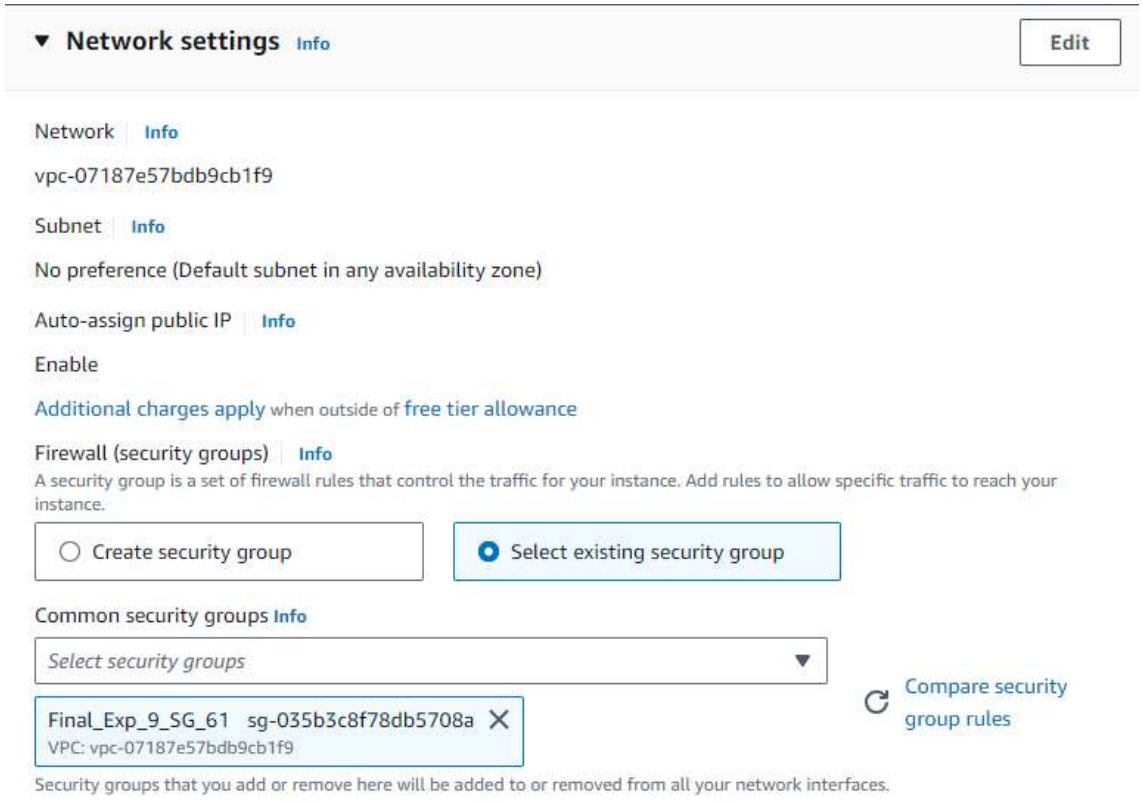


Step 2: Now Create a new EC2 instance. Name: Nagios-host ,AMI: Amazon Linux, Instance Type: t2.micro

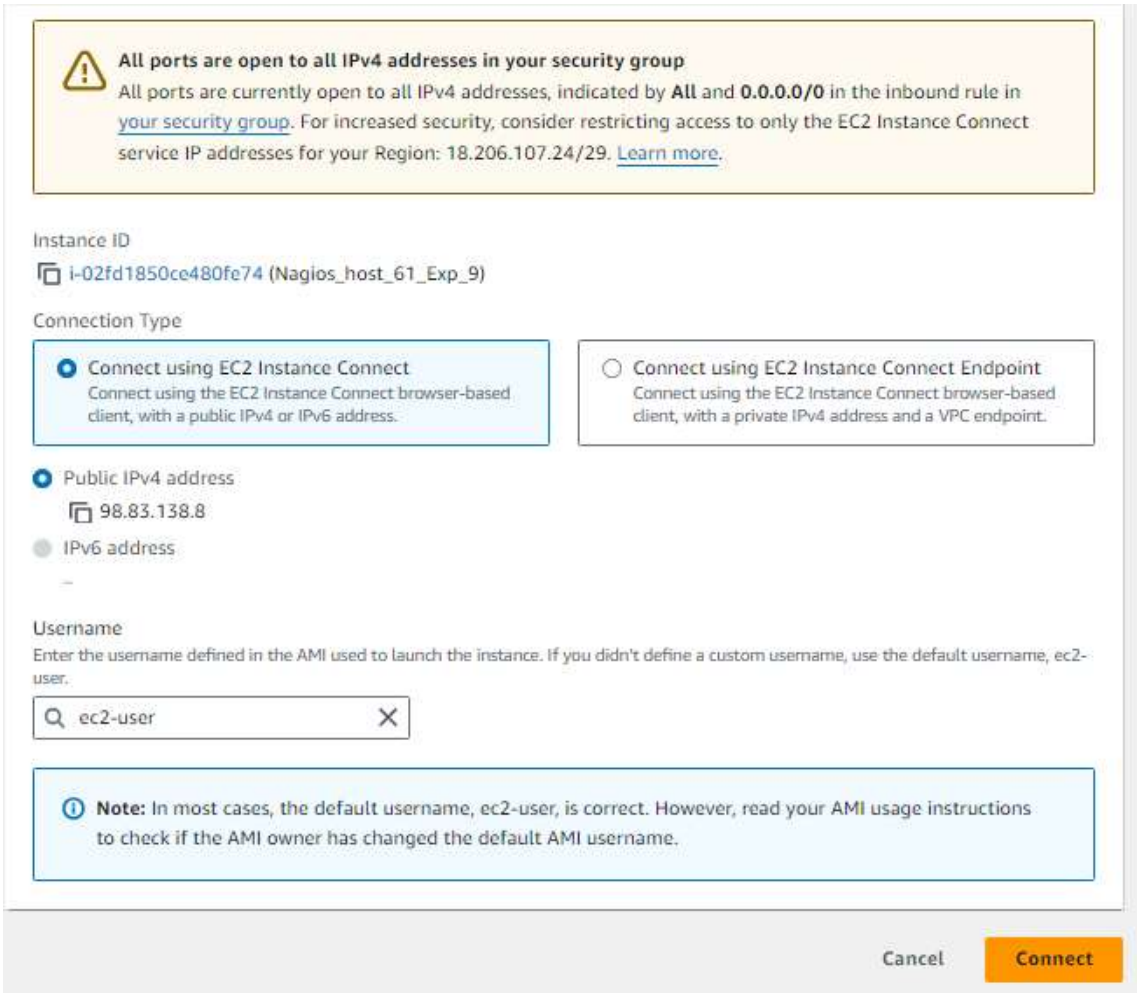




Select the Existing Security Group and select the Security Group we have created in Step 1.



Step 3: Now After creating the EC2 Instance click on connect and scroll down and connect on web server



Step 4: Now Run the following command to make a new user.
sudo adduser -m nagios
sudo passwd nagios

```
[root@ip-172-31-35-58 ec2-user]# sudo adduser -m nagios
[root@ip-172-31-35-58 ec2-user]# sudo passwd nagios
Changing password for user nagios.
New password:
BAD PASSWORD: The password fails the dictionary check - it is too simplistic/systematic
Retype new password:
passwd: all authentication tokens updated successfully.
[root@ip-172-31-35-58 ec2-user]#
```

Step 5: Now Run the following command to make a new user group.
sudo groupadd nagcmd
sudo usermod -a -G nagcmd nagios
sudo usermod -a -G nagcmd apache

```
[root@ip-172-31-41-153 ec2-user]# sudo groupadd nagcmd
[root@ip-172-31-41-153 ec2-user]# sudo usermod -a -G nagcmd nagios
[root@ip-172-31-41-153 ec2-user]# sudo usermod -a -G nagcmd apache
usermod: user 'apache' does not exist
[root@ip-172-31-41-153 ec2-user]#
```

It looks like the user apache doesn't exist on your Amazon Linux machine. On Amazon Linux, the web server user is typically called apache (for the Apache HTTP Server)

Install Apache (if not installed): If you confirm that Apache is not installed, you can install it with:
sudo yum install httpd

Then, start the Apache service:
sudo systemctl start httpd
sudo systemctl enable httpd

Add the nagios and apache users to the nagcmd group: If the Apache user is confirmed, you can add it to the group as you intended:
sudo usermod -a -G nagcmd nagios
sudo usermod -a -G nagcmd apache

Step 6: Now make a new directory and go to that directory.
mkdir ~/downloads
cd ~/downloads

```
[root@ip-172-31-41-153 ec2-user]# mkdir ~/downloads
[root@ip-172-31-41-153 ec2-user]# cd ~/downloads
[root@ip-172-31-41-153 downloads]#
```

Step 7: Now to download the Nagios 4.5.5 and Nagios-plugins 2.4.11 run the following commands respectively.
wget https://go.nagios.org/l/975333/2024-09-17/6kqcx

```
[root@ip-172-31-41-153 downloads]# wget https://go.nagios.org/l/975333/2024-09-17/6kqcx
--2024-10-02 09:49:39--  https://go.nagios.org/l/975333/2024-09-17/6kqcx
Resolving go.nagios.org (go.nagios.org)... 3.215.172.219, 18.208.125.13, 34.237.219.119, ...
Connecting to go.nagios.org (go.nagios.org)|3.215.172.219|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: http://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.5.tar.gz?utm_source=Nagios.org&utm_content=Download+Form&utm_camp
=Core+4.5.5+Download+&pi_content=1e9662c93afb2ed6bd2e3f3cc38771a7f01125e969f2a75b0e2254439d4a81d8 [following]
--2024-10-02 09:49:39--  http://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.5.tar.gz?utm_source=Nagios.org&utm_content=Downlo
rm&utm_campaign=Core+4.5.5+Download+&pi_content=1e9662c93afb2ed6bd2e3f3cc38771a7f01125e969f2a75b0e2254439d4a81d8
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|:80... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.5.tar.gz?utm_source=Nagios.org&utm_content=Download+Form&utm_ca
n=Core+4.5.5+Download+&pi_content=1e9662c93afb2ed6bd2e3f3cc38771a7f01125e969f2a75b0e2254439d4a81d8 [following]
--2024-10-02 09:49:39--  https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.5.tar.gz?utm_source=Nagios.org&utm_content=Downlo
orm&utm_campaign=Core+4.5.5+Download+&pi_content=1e9662c93afb2ed6bd2e3f3cc38771a7f01125e969f2a75b0e2254439d4a81d8
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: '6kqcx'

6kqcx                       100%[=========================================================>]   1.97M  7.53MB/s    in 0.3s
```

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

```
[root@ip-172-31-41-153 downloads]# wget https://nagios-plugins.org/download/nagios-plugins-2.4.11.tar.gz
--2024-10-02 09:50:25--  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  9.09MB/s    in 0.3s

2024-10-02 09:50:26 (9.09 MB/s) - 'nagios-plugins-2.4.11.tar.gz' saved [2753049/2753049]

[root@ip-172-31-41-153 downloads]#
```

Step 8: Now to extract the files from the downloaded Nagios 4.5.5 run the following command.
tar zxvf 6kqcx (Replace 6kqcx with your saved file name of Nagios 4.5.5 refer above screenshot(1st))

```
[root@ip-172-31-41-153 downloads]# tar zxvf 6kqcx
nagios-4.5.5/
nagios-4.5.5/.github/
nagios-4.5.5/.github/workflows/
nagios-4.5.5/.github/workflows/test.yml
nagios-4.5.5/.gitignore
nagios-4.5.5/CONTRIBUTING.md
nagios-4.5.5/Changelog
nagios-4.5.5/INSTALLING
nagios-4.5.5/LEGAL
nagios-4.5.5/LICENSE
nagios-4.5.5/Makefile.in
nagios-4.5.5/README.md
```

```
nagios-4.5.5/xdata/Makefile.in
nagios-4.5.5/xdata/xcddefault.c
nagios-4.5.5/xdata/xcddefault.h
nagios-4.5.5/xdata/xodtemplate.c
nagios-4.5.5/xdata/xodtemplate.h
nagios-4.5.5/xdata/xpddefault.c
nagios-4.5.5/xdata/xpddefault.h
nagios-4.5.5/xdata/xrddefault.c
nagios-4.5.5/xdata/xrddefault.h
nagios-4.5.5/xdata/xsddefault.c
nagios-4.5.5/xdata/xsddefault.h
[root@ip-172-31-41-153 downloads]#
```

Step 9: Now change the directory to nagios-4.5.5

```
[root@ip-172-31-41-153 downloads]# cd nagios-4.5.5
[root@ip-172-31-41-153 nagios-4.5.5]#
```

Step 10: Now run the following command to configure.
./configure --with-command-group=nagcmd

```
[root@ip-172-31-41-153 downloads]# cd nagios-4.5.5
[root@ip-172-31-41-153 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... no
checking for cc... no
checking for cl.exe... no
checking for clang... no
configure: error: in '/root/downloads/nagios-4.5.5':
configure: error: no acceptable C compiler found in $PATH
See 'config.log' for more details
```

The error you're encountering indicates that there is no C compiler installed on your system. The ./configure script is trying to find a C compiler (such as gcc or clang) but is failing because none is available.

To resolve this, you need to install a C compiler. You can install gcc, the GNU Compiler Collection, by running the following command on your Amazon Linux machine:
sudo yum groupinstall "Development Tools"

```
[root@ip-172-31-41-153 nagios-4.5.5]# sudo yum groupinstall "Development Tools"
Last metadata expiration check: 0:11:58 ago on Wed Oct  2 09:43:20 2024.
No match for group package "system-rpm-config"
No match for group package "rcs"
No match for group package "pkgconfig"
Dependencies resolved.
================================================================================
 Package                    Architecture        Version                    Repository              Size
================================================================================
Installing group/module packages:
 autoconf                   noarch              2.69-36.amzn2023.0.3       amazonlinux            666 k
 automake                   noarch              1.16.5-9.amzn2023.0.3      amazonlinux            677 k
 bison                      x86_64              3.7.4-2.amzn2023.0.2       amazonlinux            925 k
 byacc                      x86_64              2.0.20210109-2.amzn2023.0.3 amazonlinux            90 k
 cscope                     x86_64              15.9-15.amzn2023.0.3       amazonlinux            288 k
 ctags                      x86_64              5.9~1.20210725.0.amzn2023.0.2 amazonlinux         719 k
 diffstat                   x86_64              1.64-4.amzn2023.0.2        amazonlinux             43 k
 doxygen                    x86_64              2:1.9.4-1.amzn2023.0.3     amazonlinux            4.7 M
 elfutils                   x86_64              0.188-3.amzn2023.0.2       amazonlinux            525 k
```

```
  urw-base35-gothic-fonts-20200910-6.amzn2023.0.2.noarch
  urw-base35-nimbus-mono-ps-fonts-20200910-6.amzn2023.0.2.noarch
  urw-base35-nimbus-roman-fonts-20200910-6.amzn2023.0.2.noarch
  urw-base35-nimbus-sans-fonts-20200910-6.amzn2023.0.2.noarch
  urw-base35-p052-fonts-20200910-6.amzn2023.0.2.noarch
  urw-base35-standard-symbols-ps-fonts-20200910-6.amzn2023.0.2.noarch
  urw-base35-z003-fonts-20200910-6.amzn2023.0.2.noarch
  utf8proc-2.6.1-2.amzn2023.0.2.x86_64
  xml-common-0.6.3-56.amzn2023.0.2.noarch
  xorg-x11-fonts-ISO8859-1-100dpi-7.5-31.amzn2023.0.2.noarch
  xz-devel-5.2.5-9.amzn2023.0.2.x86_64
  zlib-devel-1.2.11-33.amzn2023.0.5.x86_64

Complete!
[root@ip-172-31-41-153 nagios-4.5.5]#
```

Now rerun the command ./configure --with-command-group=nagcmd
At the end we have found the error of cannot find ssl header

```
checking for Kerberos include files... configure: WARNING: could not find include files
checking for pkg-config... pkg-config
checking for SSL headers... configure: error: Cannot find ssl headers
[root@ip-172-31-41-153 nagios-4.5.5]#
```

So run following command to install ssl.
sudo yum install openssl-devel

```
[root@ip-172-31-41-153 nagios-4.5.5]# sudo yum install openssl-devel
Last metadata expiration check: 0:15:35 ago on Wed Oct  2 09:43:20 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                  28 MB/s | 3.0 MB     00:00
--------------------------------------------------------------------------------
Total                                                           18 MB/s | 3.0 MB     00:00
```

Now rerun the command ./configure --with-command-group=nagcmd

```
[root@ip-172-31-41-153 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
checking for unistd.h... yes
checking for arpa/inet.h... yes
checking for ctype.h... yes
checking for dirent.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.

[root@ip-172-31-41-153 nagios-4.5.5]#
```

Step 11: Now run the following commands to steup the Nagios.
sudo make install
Before that run make all to compile main program and CGIs

```
     - What version of Nagios you are using
     - What version of the plugins you are using
     - Relevant snippets from your config files
     - Relevant error messages from the Nagios log file

For more information on obtaining support for Nagios, visit:

       https://support.nagios.com

**************************************************************************

Enjoy.

[root@ip-172-31-41-153 nagios-4.5.5]#
```

```
[root@ip-172-31-41-153 nagios-4.5.5]# sudo make install
cd ./base && make install
make[1]: Entering directory '/root/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 '/root/downloads/nagios-4.5.5/base'
cd ./cgi && make install
make[1]: Entering directory '/root/downloads/nagios-4.5.5/cgi'
make install-basic
make[2]: Entering directory '/root/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 '/root/downloads/nagios-4.5.5/cgi'
```

sudo make install-init

```
[root@ip-172-31-41-153 nagios-4.5.5]# sudo make install-init
/usr/bin/install -c -m 755 -d -o root -g root /lib/systemd/system
/usr/bin/install -c -m 755 -o root -g root startup/default-service /lib/systemd/system/nagios.service
[root@ip-172-31-41-153 nagios-4.5.5]#
```

sudo make install-config

```
[root@ip-172-31-41-153 nagios-4.5.5]# sudo make install-config
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc/objects
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/nagios.cfg /usr/local/nagios/etc/nagios.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/cgi.cfg /usr/local/nagios/etc/cgi.cfg
/usr/bin/install -c -b -m 660 -o nagios -g nagios sample-config/resource.cfg /usr/local/nagios/etc/resource.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/templates.cfg /usr/local/nagios/etc/objects/templates.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/commands.cfg /usr/local/nagios/etc/objects/commands.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/contacts.cfg /usr/local/nagios/etc/objects/contacts.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/timeperiods.cfg /usr/local/nagios/etc/objects/timeperiods.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/localhost.cfg /usr/local/nagios/etc/objects/localhost.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/windows.cfg /usr/local/nagios/etc/objects/windows.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/printer.cfg /usr/local/nagios/etc/objects/printer.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/switch.cfg /usr/local/nagios/etc/objects/switch.cfg

*** 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.

[root@ip-172-31-41-153 nagios-4.5.5]#
```

sudo make install-webconf

The error you're seeing indicates that the directory /etc/httpd/conf.d/ does not exist, which means that Apache HTTP Server (httpd) might not be installed on your system.

```
[root@ip-172-31-41-153 nagios-4.5.5]# sudo make install-webconf
/usr/bin/install -c -m 644 sample-config/httpd.conf /etc/httpd/conf.d/nagios.conf
/usr/bin/install: cannot create regular file '/etc/httpd/conf.d/nagios.conf': No such file or directory
make: *** [Makefile:351: install-webconf] Error 1
[root@ip-172-31-41-153 nagios-4.5.5]#
```

To resolve this, follow these steps:
sudo yum install httpd
sudo mkdir -p /etc/httpd/conf.d
sudo make install-webconf
sudo systemctl start httpd
sudo systemctl enable httpd

```
[root@ip-172-31-41-153 nagios-4.5.5]# sudo yum install httpd
Last metadata expiration check: 0:25:04 ago on Wed Oct  2 09:43:20 2024.
Dependencies resolved.
================================================================================
 Package                         Architecture            Version
================================================================================
Installing:
 httpd                           x86_64                  2.4.62-1.amzn2023
Installing dependencies:
 generic-logos-httpd             noarch                  18.0.0-12.amzn2023.0.3
 httpd-core                      x86_64                  2.4.62-1.amzn2023
 httpd-filesystem                noarch                  2.4.62-1.amzn2023
 httpd-tools                     x86_64                  2.4.62-1.amzn2023
 mailcap                         noarch                  2.1.49-3.amzn2023.0.3
Installing weak dependencies:
 mod_http2                       x86_64                  2.0.27-1.amzn2023.0.3
 mod_lua                         x86_64                  2.4.62-1.amzn2023

Transaction Summary
================================================================================
Install  8 Packages
```

```
[root@ip-172-31-41-153 nagios-4.5.5]# sudo mkdir -p /etc/httpd/conf.d
[root@ip-172-31-41-153 nagios-4.5.5]# sudo make install-webconf
/usr/bin/install -c -m 644 sample-config/httpd.conf /etc/httpd/conf.d/nagios.conf
if [ 0 -eq 1 ]; then \
       ln -s /etc/httpd/conf.d/nagios.conf /etc/apache2/sites-enabled/nagios.conf; \
fi

*** Nagios/Apache conf file installed ***

[root@ip-172-31-41-153 nagios-4.5.5]# sudo systemctl start httpd
[root@ip-172-31-41-153 nagios-4.5.5]# sudo systemctl enable httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/lib/systemd/system/httpd.service.
[root@ip-172-31-41-153 nagios-4.5.5]#
```

Now sudo make install-webconf Has been successfully run

sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin

```
[root@ip-172-31-41-153 nagios-4.5.5]# sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
New password:
Re-type new password:
Adding password for user nagiosadmin
[root@ip-172-31-41-153 nagios-4.5.5]#
```

Now to restart the httpd service run the following command.
sudo service httpd restart

```
[root@ip-172-31-41-153 nagios-4.5.5]# sudo service httpd restart
Redirecting to /bin/systemctl restart httpd.service
[root@ip-172-31-41-153 nagios-4.5.5]#
```

Step 12: Now to extract the files from the downloaded Nagios plugin 2.4.11 run the following command
first change the directory.
cd ~/downloads
tar zxvf nagios-plugins-2.4.11.tar.gz

```
[root@ip-172-31-41-153 nagios-4.5.5]# cd ~/downloads
[root@ip-172-31-41-153 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
nagios-plugins-2.4.11/config_test/
nagios-plugins-2.4.11/config_test/Makefile
nagios-plugins-2.4.11/config_test/run_tests
nagios-plugins-2.4.11/config_test/child_test.c
nagios-plugins-2.4.11/gl/
nagios-plugins-2.4.11/gl/m4/
nagios-plugins-2.4.11/gl/m4/00gnulib.m4
nagios-plugins-2.4.11/gl/m4/absolute-header.m4
nagios-plugins-2.4.11/gl/m4/alloca.m4
nagios-plugins-2.4.11/gl/m4/arpa_inet_h.m4
nagios-plugins-2.4.11/gl/m4/base64.m4
nagios-plugins-2.4.11/gl/m4/btowc.m4
```

Step 13: Now change the directory to nagios-plugins-2.4.11 and run the config command to configure.
cd nagios-plugins-2.4.11
./configure --with-nagios-user=nagios --with-nagios-group=nagios

```
[root@ip-172-31-41-153 downloads]# cd nagios-plugins-2.4.11
[root@ip-172-31-41-153 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
```

```
config.status: creating po/Makefile.in
config.status: creating config.h
config.status: config.h is unchanged
config.status: executing depfiles commands
config.status: executing libtool commands
config.status: executing po-directories commands
config.status: creating po/POTFILES
config.status: creating po/Makefile
[root@ip-172-31-41-153 nagios-plugins-2.4.11]#
```

Step 14: Run the following commands to check nagios and start it.
sudo chkconfig --add nagios
sudo chkconfig nagios on

```
[root@ip-172-31-41-153 nagios-plugins-2.4.11]# sudo chkconfig --add nagios
error reading information on service nagios: No such file or directory
[root@ip-172-31-41-153 nagios-plugins-2.4.11]# sudo chkconfig nagios on
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.
[root@ip-172-31-41-153 nagios-plugins-2.4.11]# sudo chkconfig --add nagios
error reading information on service nagios: No such file or directory
[root@ip-172-31-41-153 nagios-plugins-2.4.11]#
```

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

```
[root@ip-172-31-41-153 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...
```

```
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...
        Checked 1 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-172-31-41-153 nagios-plugins-2.4.11]#
```

cd

sudo service nagios start

```
[root@ip-172-31-41-153 nagios-plugins-2.4.11]# cd
[root@ip-172-31-41-153 ~]# sudo service nagios start
Redirecting to /bin/systemctl start nagios.service
[root@ip-172-31-41-153 ~]#
```

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

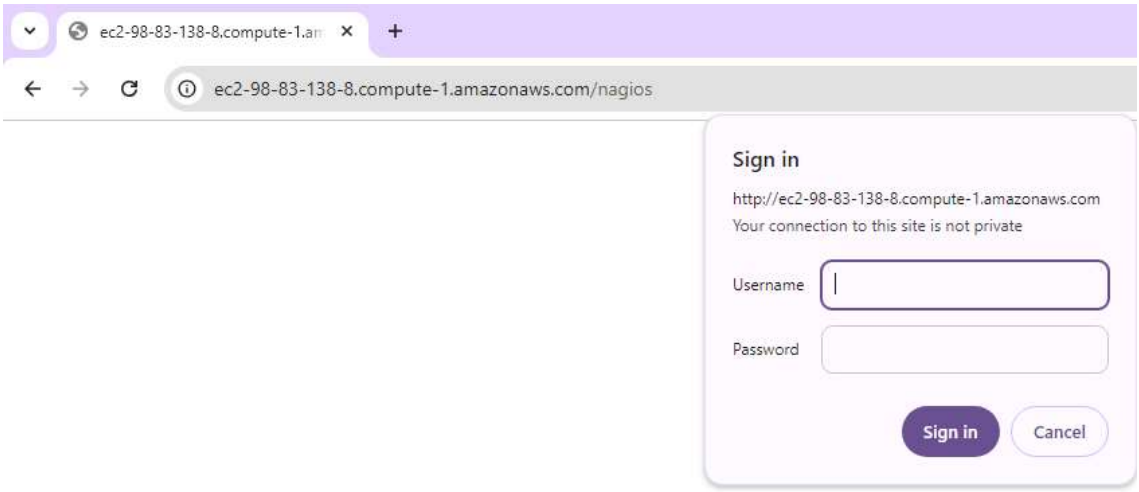
```
Running pre-flight check on configuration data...

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...
        Checked 1 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-172-31-41-153 ~]#
```

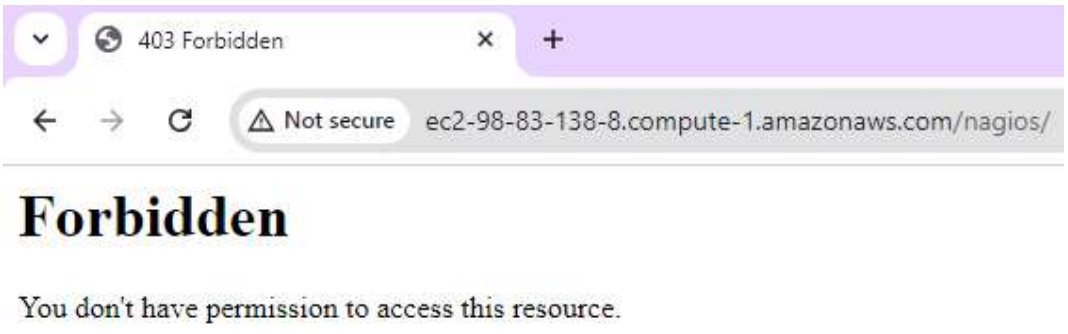
sudo systemctl status nagios

```
[root@ip-172-31-41-153 ~]# sudo systemctl status nagios
● nagios.service - Nagios Core 4.5.5
     Loaded: loaded (/usr/lib/systemd/system/nagios.service; enabled; preset: disabled)
     Active: active (running) since Wed 2024-10-02 10:23:28 UTC; 2min 14s ago
       Docs: https://www.nagios.org/documentation
   Main PID: 55480 (nagios)
      Tasks: 6 (limit: 1112)
     Memory: 5.5M
        CPU: 109ms
     CGroup: /system.slice/nagios.service
             ├─55480 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─55481 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─55482 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─55483 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─55484 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             └─55485 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg

Oct 02 10:23:28 ip-172-31-41-153.ec2.internal nagios[55480]: wproc: Registry request: name=Core Worker 55483;pid=55483
Oct 02 10:23:28 ip-172-31-41-153.ec2.internal nagios[55480]: wproc: Registry request: name=Core Worker 55482;pid=55482
Oct 02 10:23:28 ip-172-31-41-153.ec2.internal nagios[55480]: wproc: Registry request: name=Core Worker 55481;pid=55481
Oct 02 10:23:28 ip-172-31-41-153.ec2.internal nagios[55480]: Successfully launched command file worker with pid 55485
Oct 02 10:23:28 ip-172-31-41-153.ec2.internal nagios[55480]: HOST ALERT: localhost;DOWN;SOFT;1;(No output on stdout) stderr: execvp(/usr/local/n>
Oct 02 10:24:05 ip-172-31-41-153.ec2.internal nagios[55480]: SERVICE ALERT: localhost;Current Load;CRITICAL;HARD;1;(No output on stdout) stderr:>
Oct 02 10:24:28 ip-172-31-41-153.ec2.internal nagios[55480]: HOST ALERT: localhost;DOWN;SOFT;2;(No output on stdout) stderr: execvp(/usr/local/n>
lines 1-23
[root@ip-172-31-41-153 ~]#
```

Step 15: We can see we have successfully launched the Nagios now . Open http://<instance public ip >/nagios/ here it is http://52.23.153.85/nagios we can see the running web page of nagios

Or else
http://ec2-52-23-153-85.compute-1.amazonaws.com/nagios





Step 16 :

1. Check Apache Configuration
Make sure that Apache is configured to allow access to the Nagios web interface.

Open the Nagios configuration file for Apache, typically found at /etc/httpd/conf.d/nagios.conf or a similar location.
sudo nano /etc/httpd/conf.d/nagios.conf

Look for the section that defines access controls. It might look something like this:

```
<Directory "/usr/local/nagios/share">
   Options None
   AllowOverride None
   Require all granted
</Directory>
```

Ensure that the Require all granted directive is present. If it's set to Require all denied, change it to Require all granted.

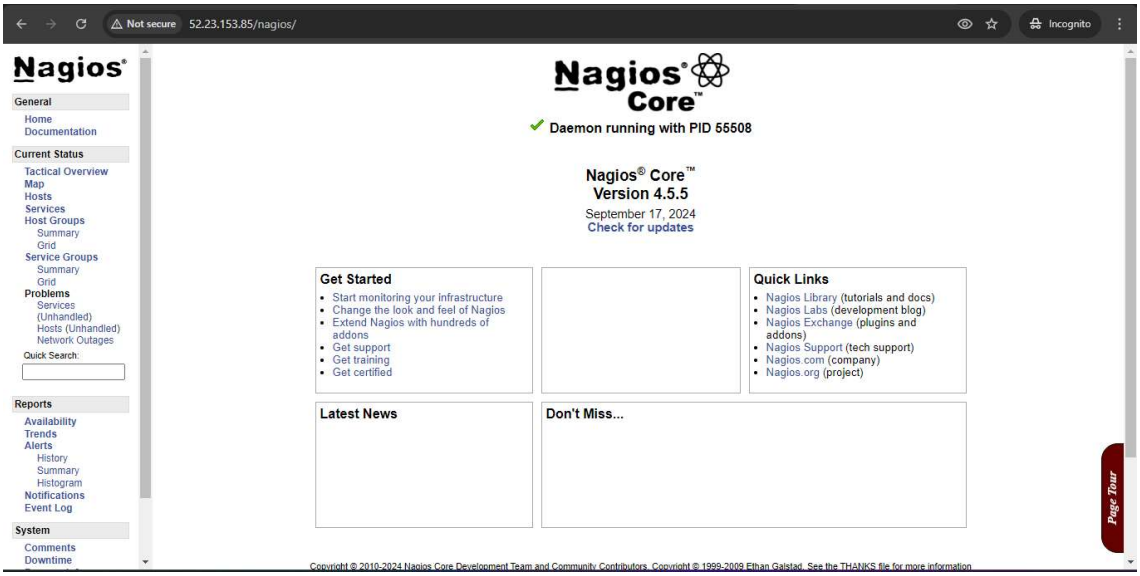
2.Check Directory Permissions
Ensure that the Apache user (apache or www-data depending on your distribution) has the correct permissions to access the Nagios web directory.

sudo chown -R apache:apache /usr/local/nagios/share
3.SELinux (if applicable)
If SELinux is enabled, it may block access even if Apache permissions are correct. You can check the status with:
sestatus

If it is enabled, you might need to adjust the security context:
sudo chcon -R -t httpd_sys_content_t /usr/local/nagios/share

4.Restart Apache
After making changes, restart the Apache service:

sudo systemctl restart httpd



Open
http://<instance public ip >/nagios/ here it is http://18.234.24.186/nagios we can see the running web page of nagios

Or else
http://ec2-52-23-153-85.compute-1.amazonaws.com/nagios

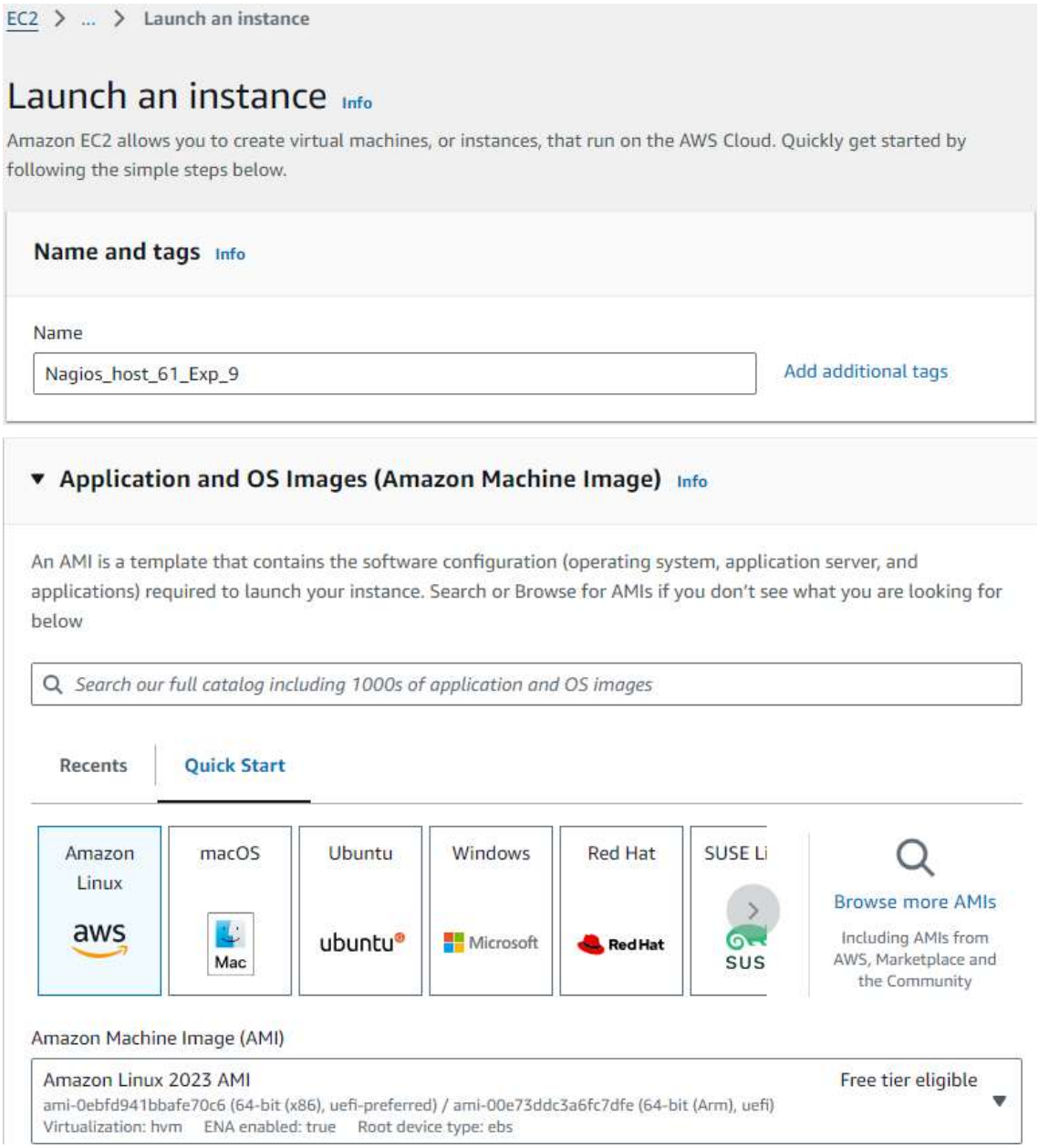
Conclusion :

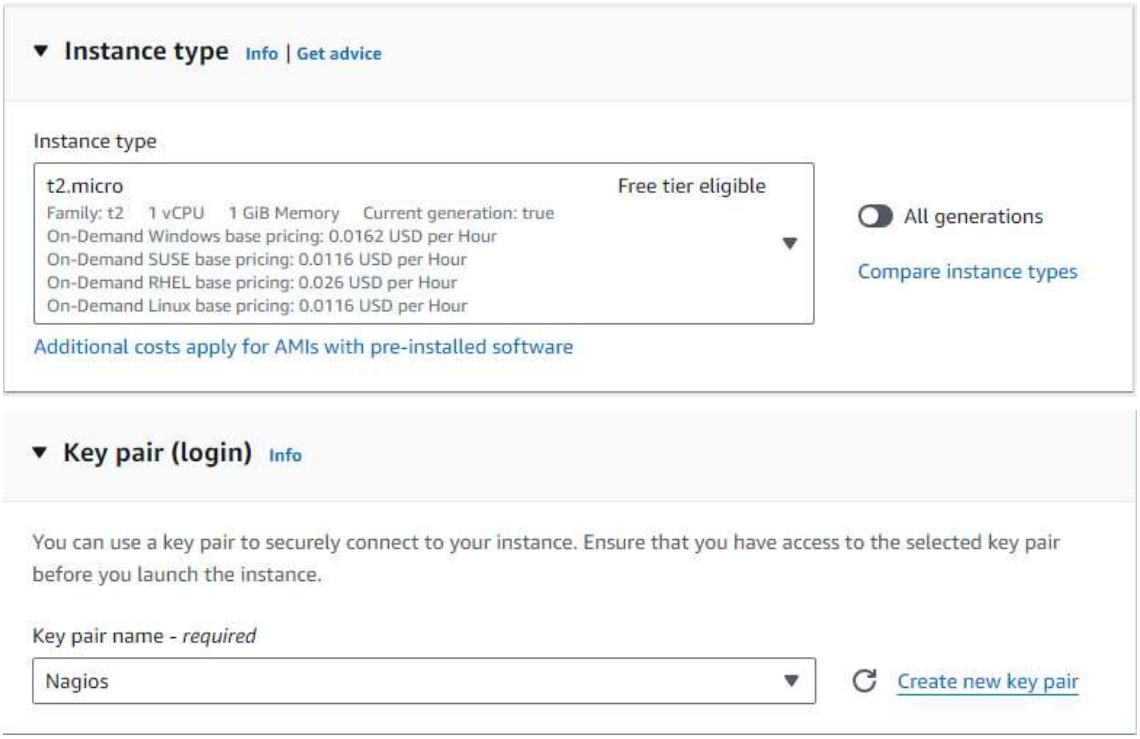
In this experiment, we have set up Nagios Core with plugins on Amazon Linux, which will help us understand continuous monitoring and installation. It is important to note that the set of rules added in Step 1 are crucial for the success of this experiment. By configuring Nagios, we enable effective monitoring of systems, networks, and applications, allowing for timely alerts and proactive management of infrastructure performance and security in a DevOps environment.

Experiment No:10

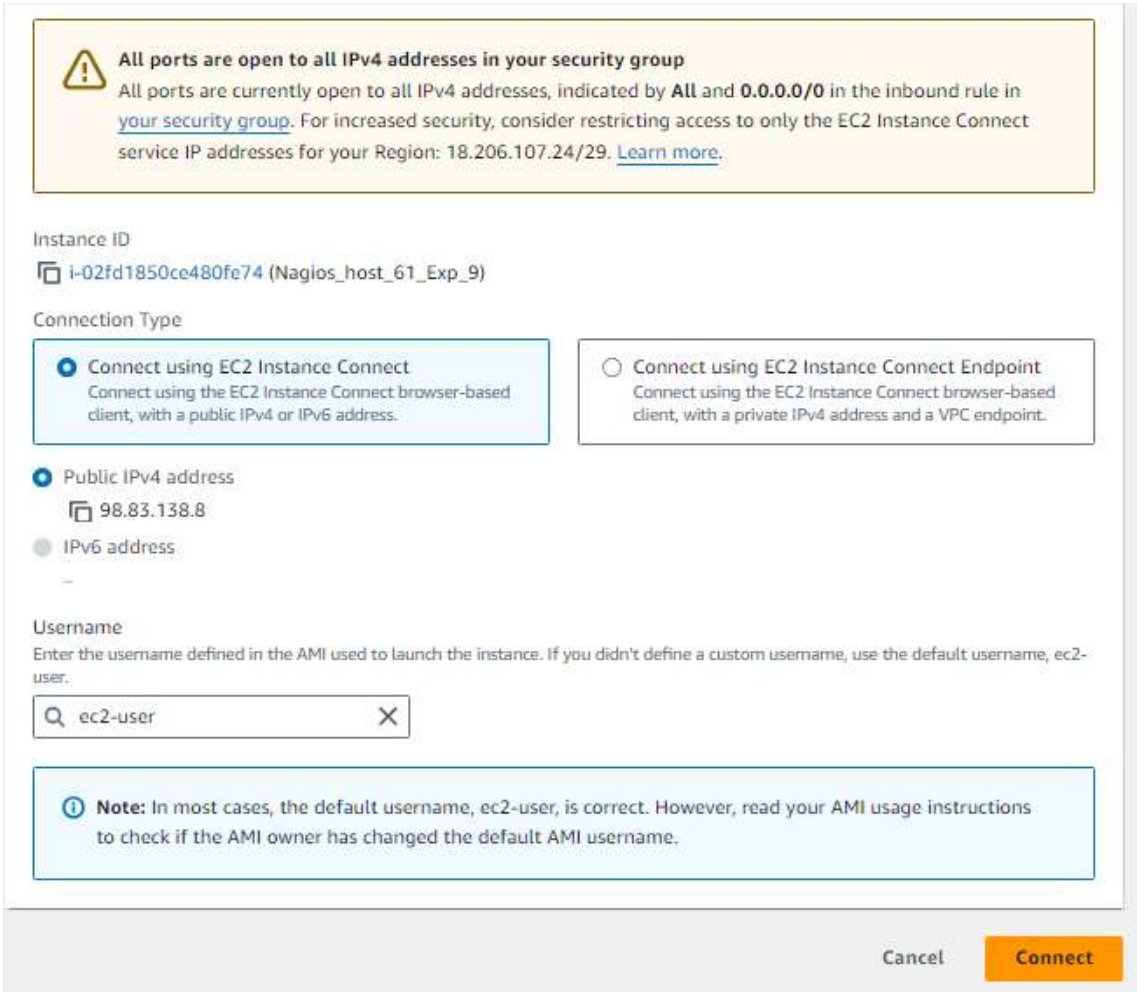
AIM: To perform Port, Service monitoring, Windows/Linux server monitoring using Nagios.

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.
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.





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.


All ports are open to all IPv4 addresses in your security group
All ports are currently open to all IPv4 addresses, indicated by All and 0.0.0.0/0 in the inbound rule in your security group. For increased security, consider restricting access to only the EC2 Instance Connect service IP addresses for your Region: 18.206.107.24/29. Learn more.
Instance ID
i-02fd1850ce480fe74 (Nagios_host_61_Exp_9)
Connection Type
Connect using EC2 Instance Connect
Connect using the EC2 Instance Connect browser-based client, with a public IPv4 or IPv6 address.
Connect using EC2 Instance Connect Endpoint
Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.
Public IPv4 address
98.83.138.8
IPv6 address
Username
Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, ec2-user.
ec2-user
Note: In most cases, the default username, ec2-user, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.
Cancel
Connect

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

```
[root@ip-172-31-35-58 ~]# ps -ef | grep nagios
nagios     55508       1  0 03:55 ?        00:00:00 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
nagios     55509   55508  0 03:55 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.q
nagios     55510   55508  0 03:55 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.q
nagios     55511   55508  0 03:55 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.q
nagios     55512   55508  0 03:55 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.q
nagios     55513   55508  0 03:55 ?        00:00:00 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
root       57705    2695  0 04:29 pts/1    00:00:00 grep --color=auto nagios
[root@ip-172-31-35-58 ~]#
```

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

```
[root@ip-172-31-35-58 ~]# mkdir /usr/local/nagios/etc/objects/monitorhosts
[root@ip-172-31-35-58 ~]# mkdir /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
[root@ip-172-31-35-58 ~]#
```

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

```
[root@ip-172-31-35-58 ~]# cp /usr/local/nagios/etc/objects/localhost.cfg /usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg
[root@ip-172-31-35-58 ~]#
```

So make the second directory again and run the cp command

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

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
}
```

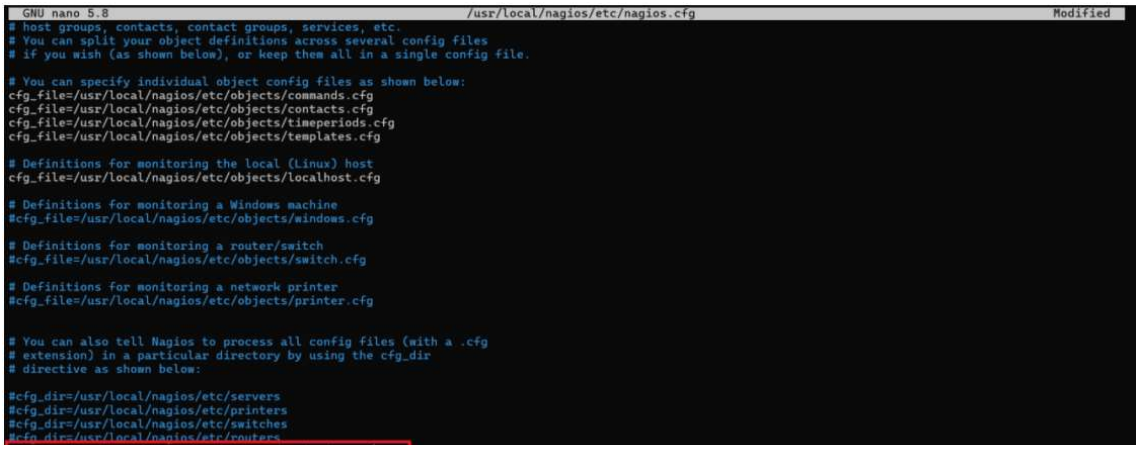
Change hostgroup_name to linux-servers1

```
# Define a host for the local machine

define host {

    use                     linux-server            ; Name of host template to use
                                                    ; This host definition will inherit all variables that are defined
                                                    ; in (or inherited by) the linux-server host template definition.
    host_name               linuxserver
    alias                   localhost
    address                 52.23.153.85
}
```

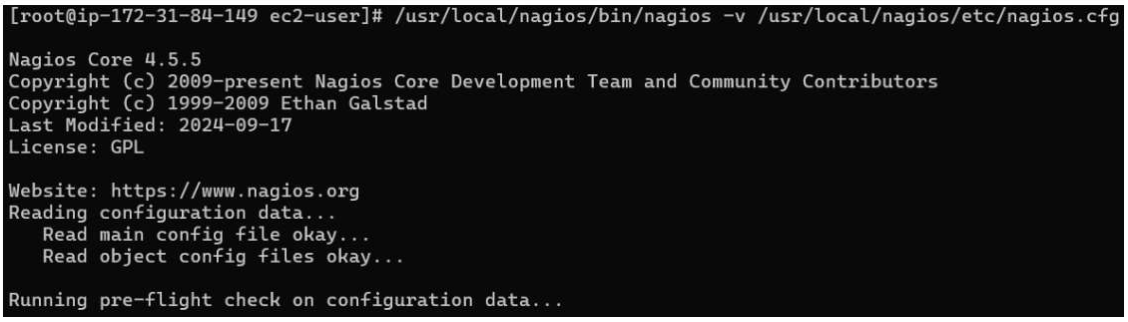
Change the occurrences of hostname further in the document from localhost to linuxserver
Now, we need to edit the nagios configuration file to add this directory. Run this command
nano /usr/local/nagios/etc/nagios.cfg

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

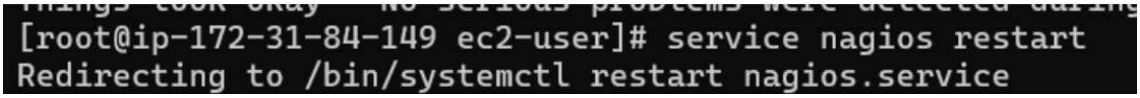
and add the following line cfg_dir=/usr/local/nagios/etc/objects/monitorhosts/

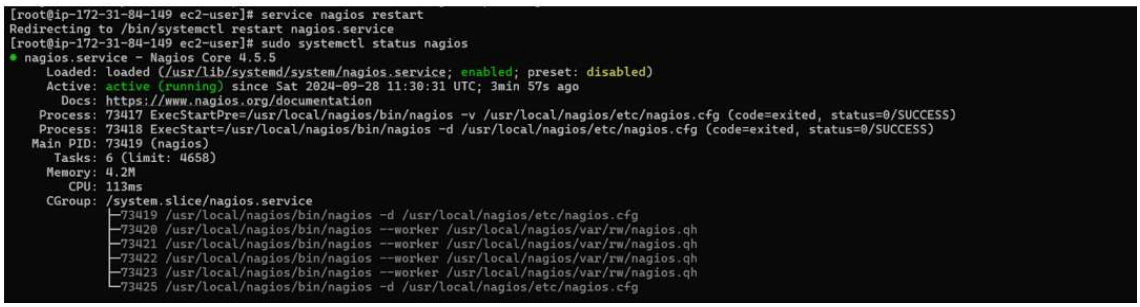


Now we verify the configuration files. /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg



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





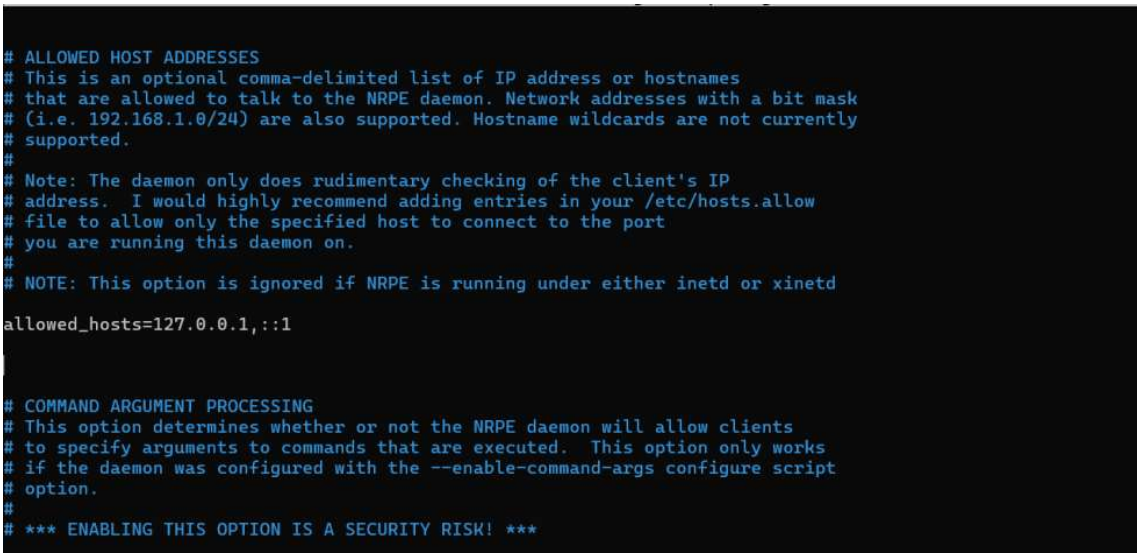
Step 3: Execute the following on Nagios Client machine (Ubuntu)
1) First, check for any available updates, and then proceed to install gcc, the Nagios NRPE server, and Nagios plugins.
sudo apt update -y
sudo apt install gcc -y
sudo apt install -y nagios-nrpe-server nagios-plugins

```
Restarting services...

Service restarts being deferred:
 /etc/needrestart/restart.d/dbus.service
 systemctl restart getty@tty1.service
 systemctl restart networkd-dispatcher.service
 systemctl restart serial-getty@ttyS0.service
 systemctl restart systemd-logind.service
 systemctl restart unattended-upgrades.service

No containers need to be restarted.

User sessions running outdated binaries:
 ubuntu @ session #6: sshd[1071,1184]
 ubuntu @ user manager service: systemd[1077]

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

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

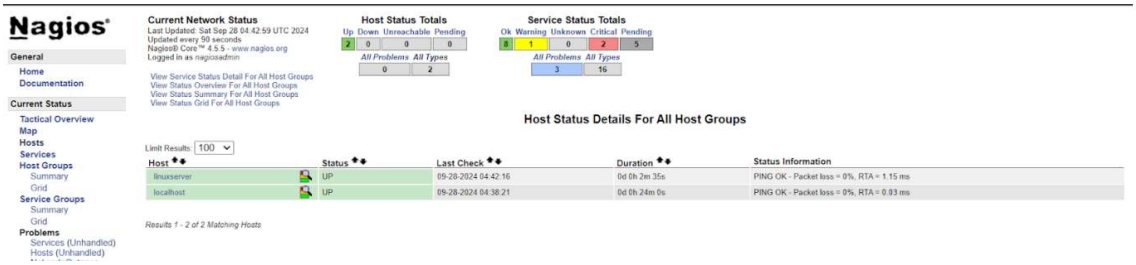
```
##############################################################################
#
#  Sample NRPE Config File
#
#  Notes:
#
#  This is a sample configuration file for the NRPE daemon.  It needs to be
#  located on the remote host that is running the NRPE daemon, not the host
#  from which the check_nrpe client is being executed.
#
##############################################################################


# LOG FACILITY
# The syslog facility that should be used for logging purposes.

log_facility=daemon



# LOG FILE
# If a log file is specified in this option, nrpe will write to
# that file instead of using syslog.

#log_file=/var/log/nrpe.log
```

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

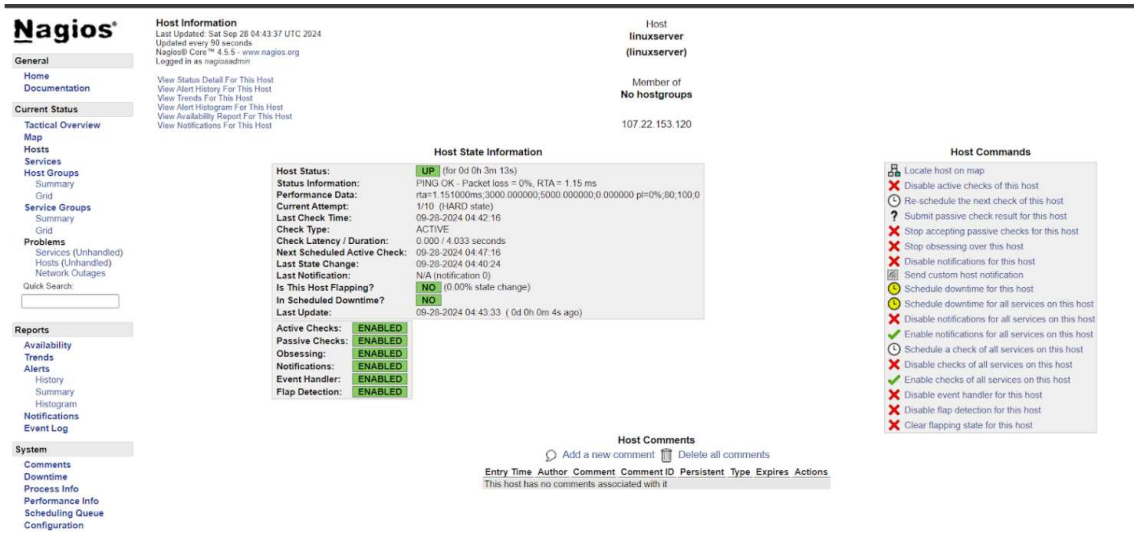


```
# ALLOWED HOST ADDRESSES
# This is an optional comma-delimited list of IP address or hostnames
# that are allowed to talk to the NRPE daemon. Network addresses with a bit mask
# (i.e. 192.168.1.0/24) are also supported. Hostname wildcards are not currently
# supported.
#
# Note: The daemon only does rudimentary checking of the client's IP
# address.  I would highly recommend adding entries in your /etc/hosts.allow
# file to allow only the specified host to connect to the port
# you are running this daemon on.
#
# NOTE: This option is ignored if NRPE is running under either inetd or xinetd

allowed_hosts=127.0.0.1,::1


# COMMAND ARGUMENT PROCESSING
# This option determines whether or not the NRPE daemon will allow clients
# to specify arguments to commands that are executed.  This option only works
# if the daemon was configured with the --enable-command-args configure script
# option.
#
# *** ENABLING THIS OPTION IS A SECURITY RISK! ***
```

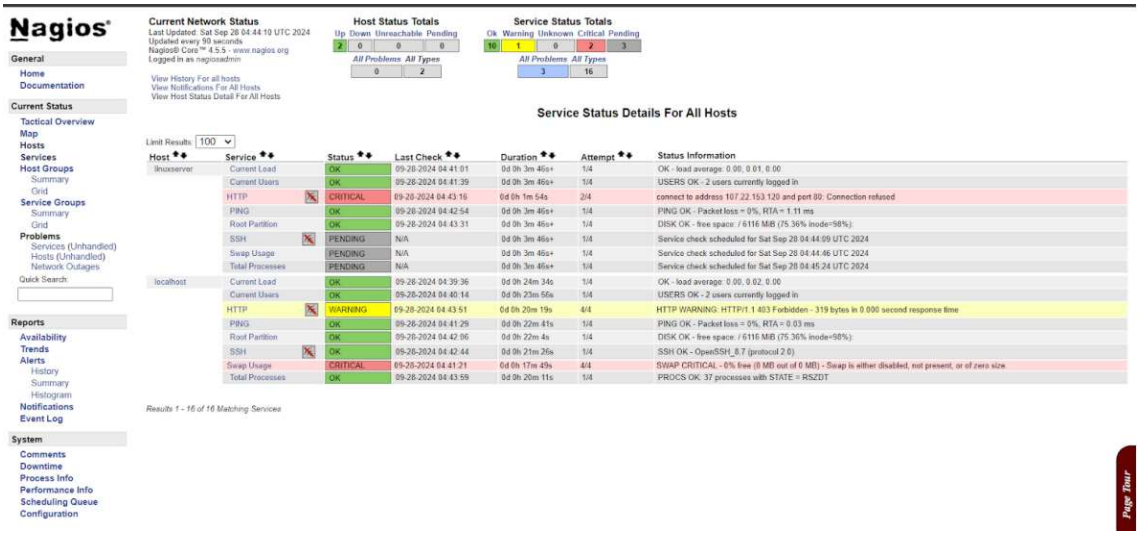
Step 4: Check the Nagios Dashboard.Go to Nagios dashboard, click on hosts.Here, we can see that the linuxserver is also added as a host.



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 successfully conducted port and service monitoring along with server monitoring using Nagios. We set up an Amazon Linux instance as the Nagios host and linked it with an Ubuntu instance as the monitored host. The configuration involved updating the Nagios host to recognize the Ubuntu server, editing the necessary configuration files, and installing Nagios NRPE on the client machine. Once the setup was complete, the `linuxserver` was successfully added as a monitored host in the Nagios dashboard, allowing us to view and track its services in real time.

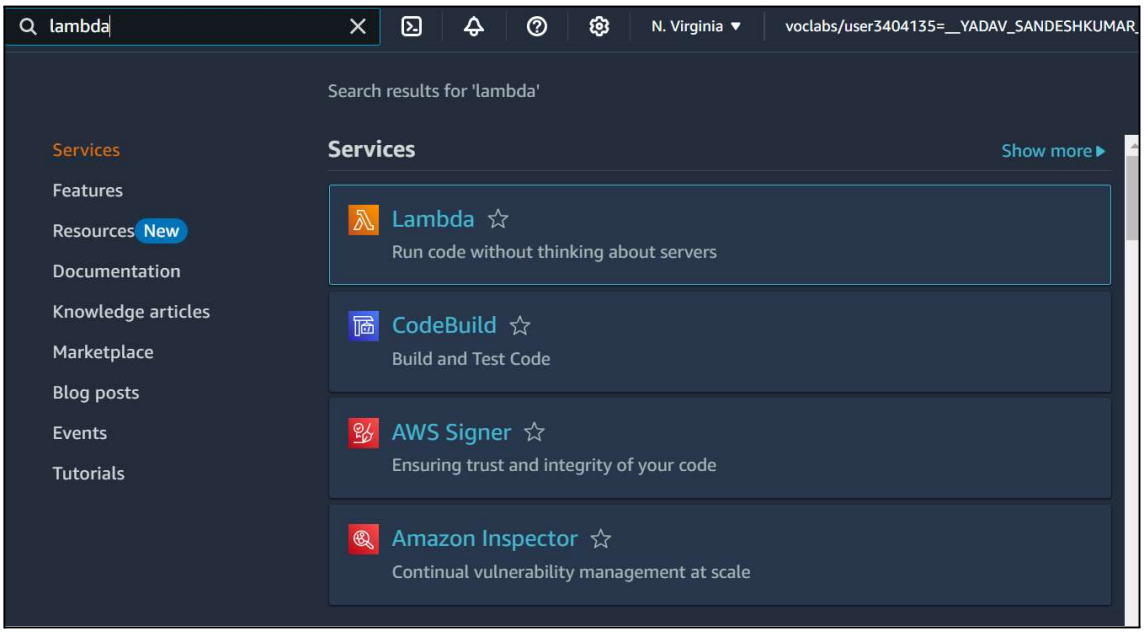
Name : Sandeshkumar.M.Yadav **Div : D15C** **Roll : 61**

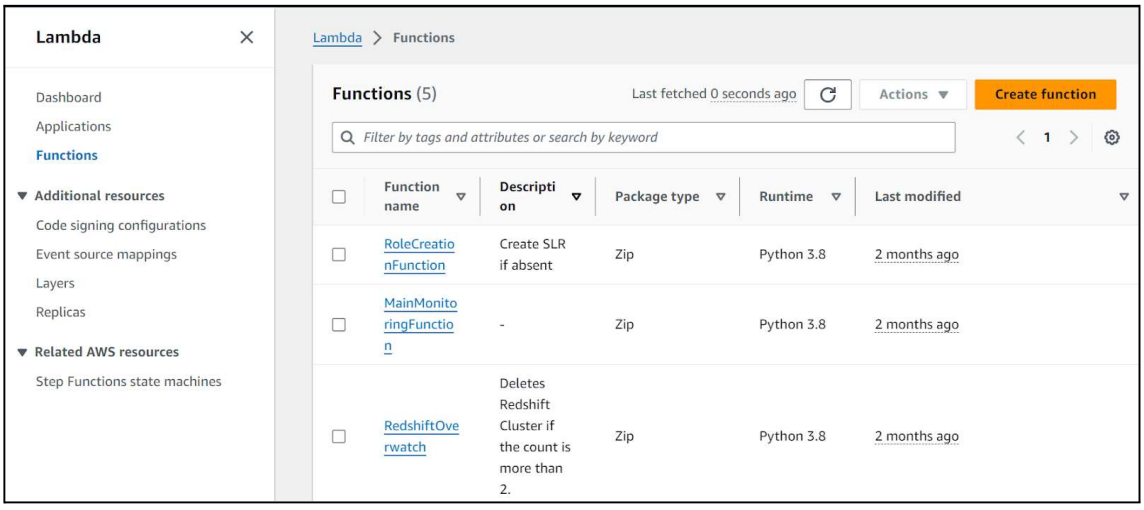
Experiment No: 11

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

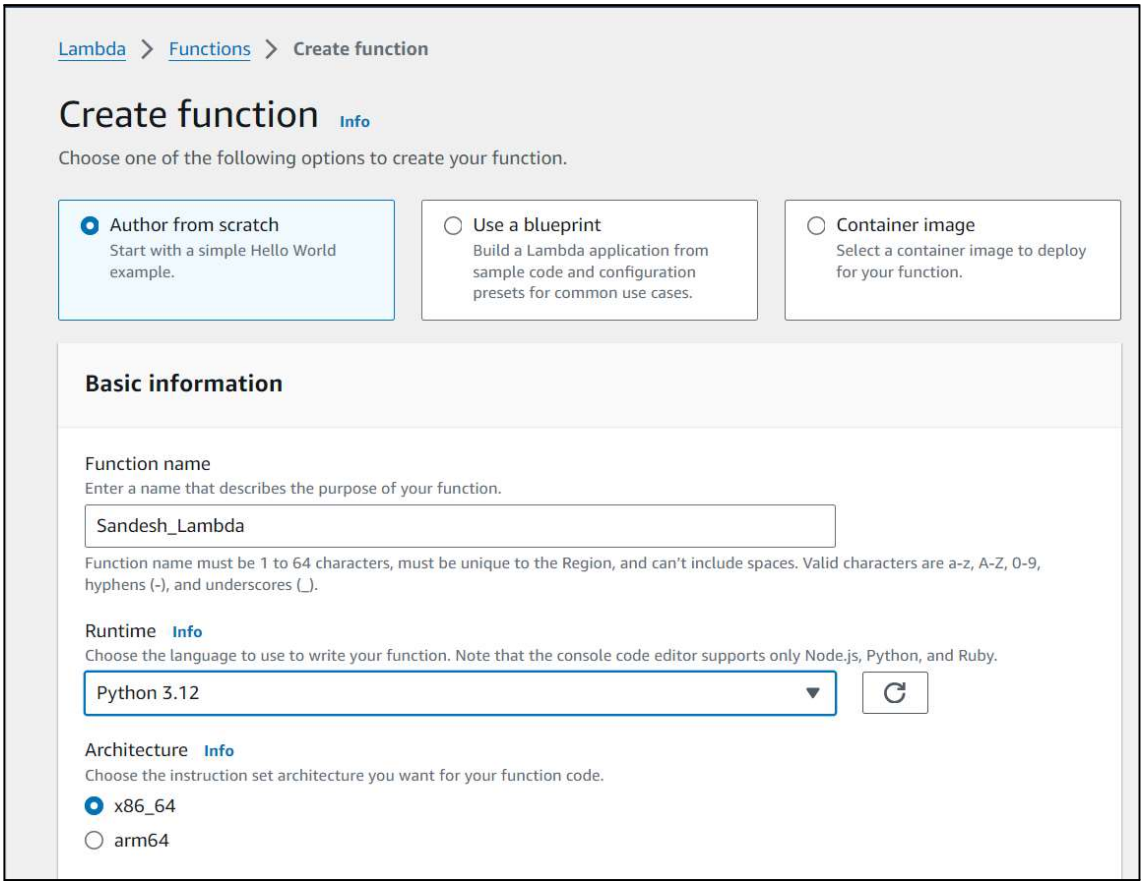
CREATION OF LAMBDA FUNCTION:

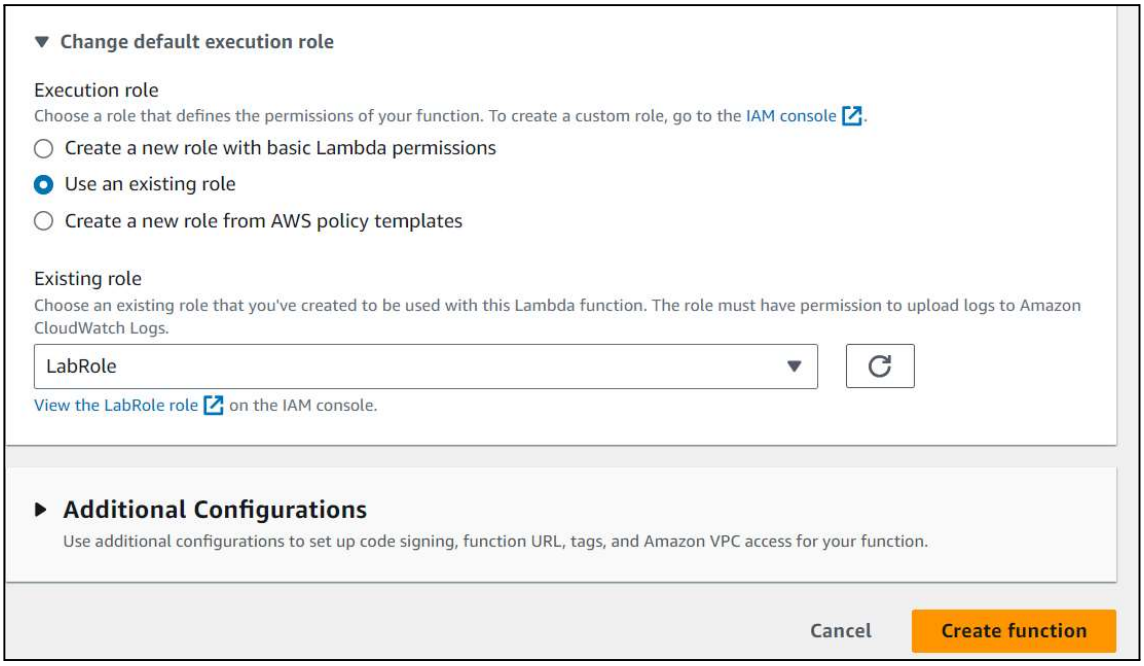
Step1 : Log in to your AWS Personal or Academy account. Navigate to Lambda, then select the 'Create Function' button

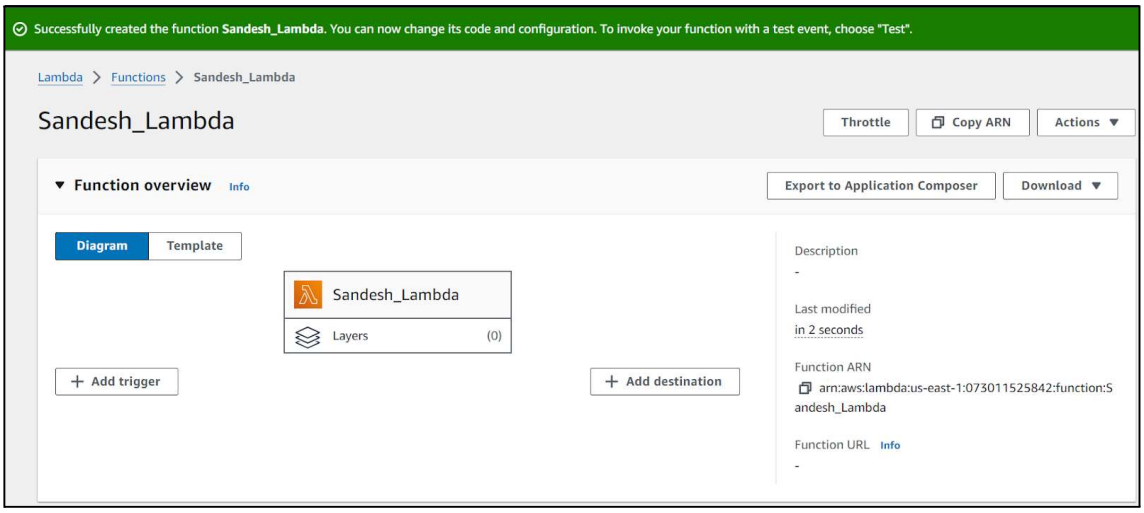




Step 2 : Give your Lambda function a name and choose a programming language. The code editor only supports Node.js, Python, and Ruby, so in my case I have chosen Python 3.12. Set the architecture to x86. For the execution role, select 'Use an existing role,' then pick 'Lab role' from the dropdown menu 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)



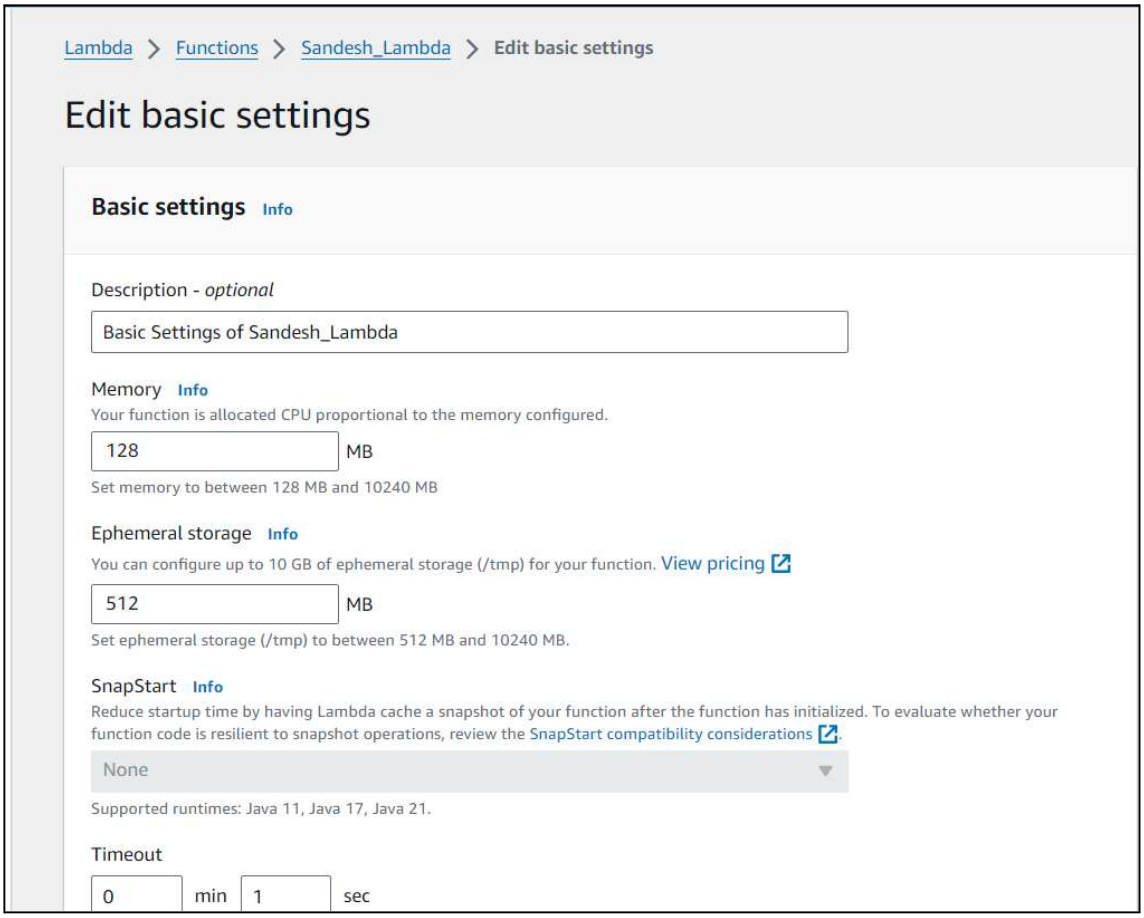


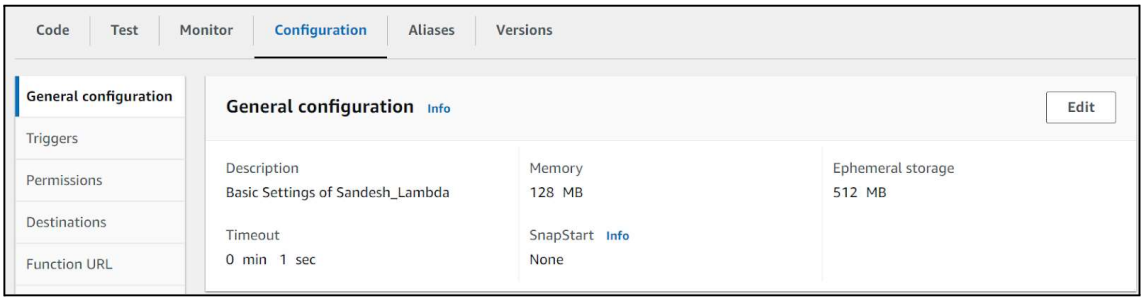


Successfully created Lambda function

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

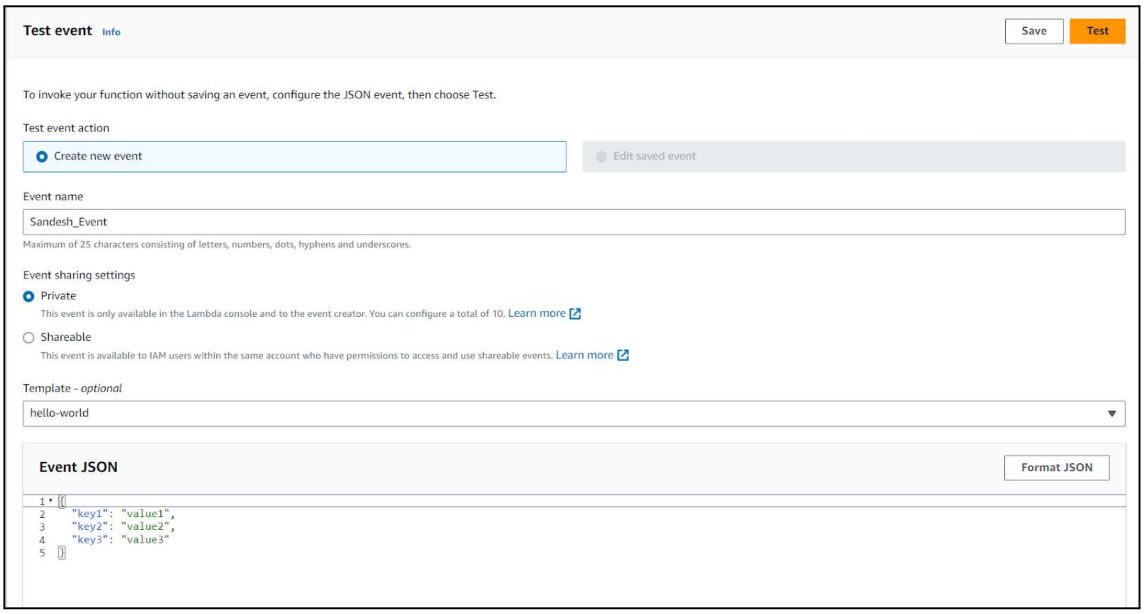
You can add a description and adjust the memory and timeout settings. I've changed the timeout to 1 second, as that's enough for now.



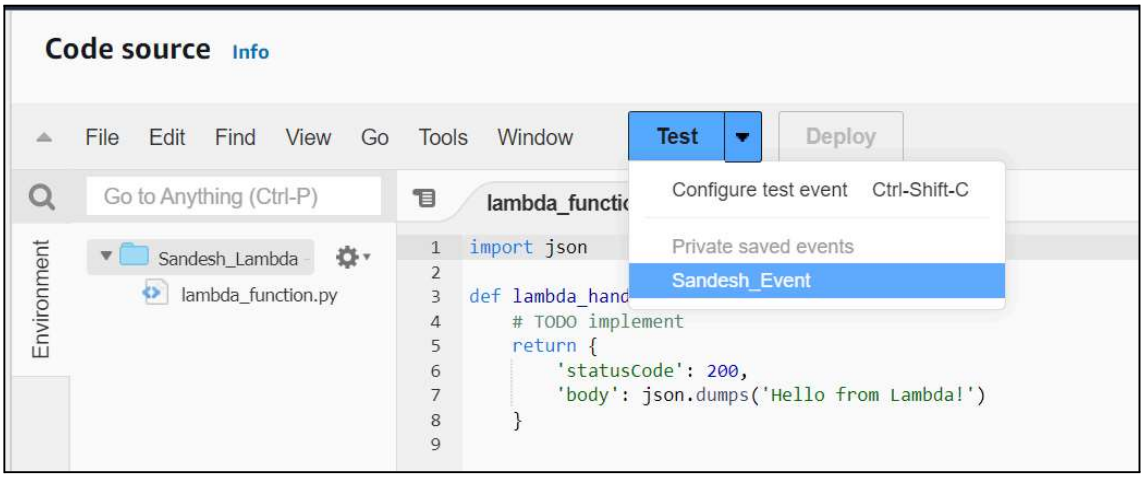


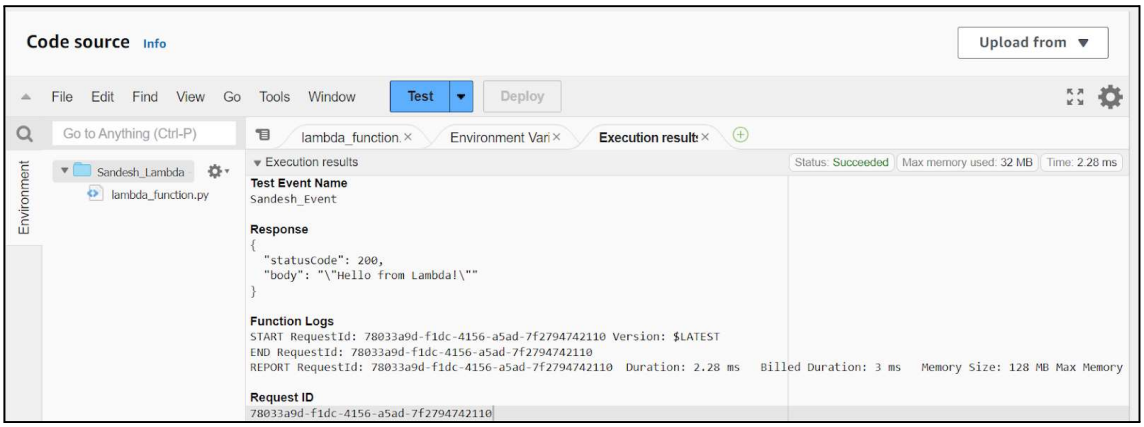
Step 4: Go to the 'Test' tab and click 'Create a new event.' Give the event a name, set 'Event Sharing' to private, and choose the 'hello-world' template.

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.



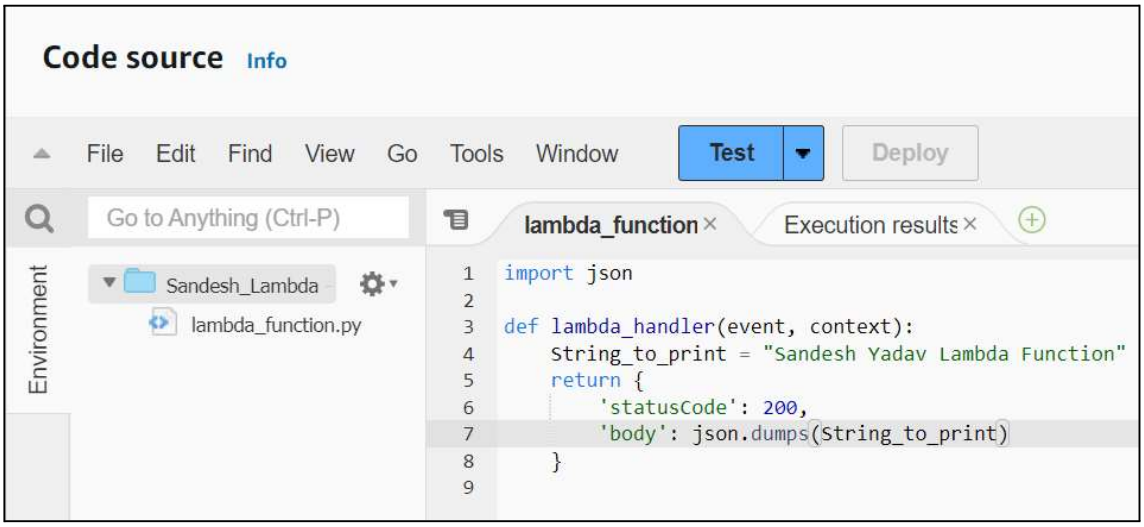
Step 5: In the Code section, select the event you created from the dropdown menu under 'Test,' then click 'Test.' You should see the output below."

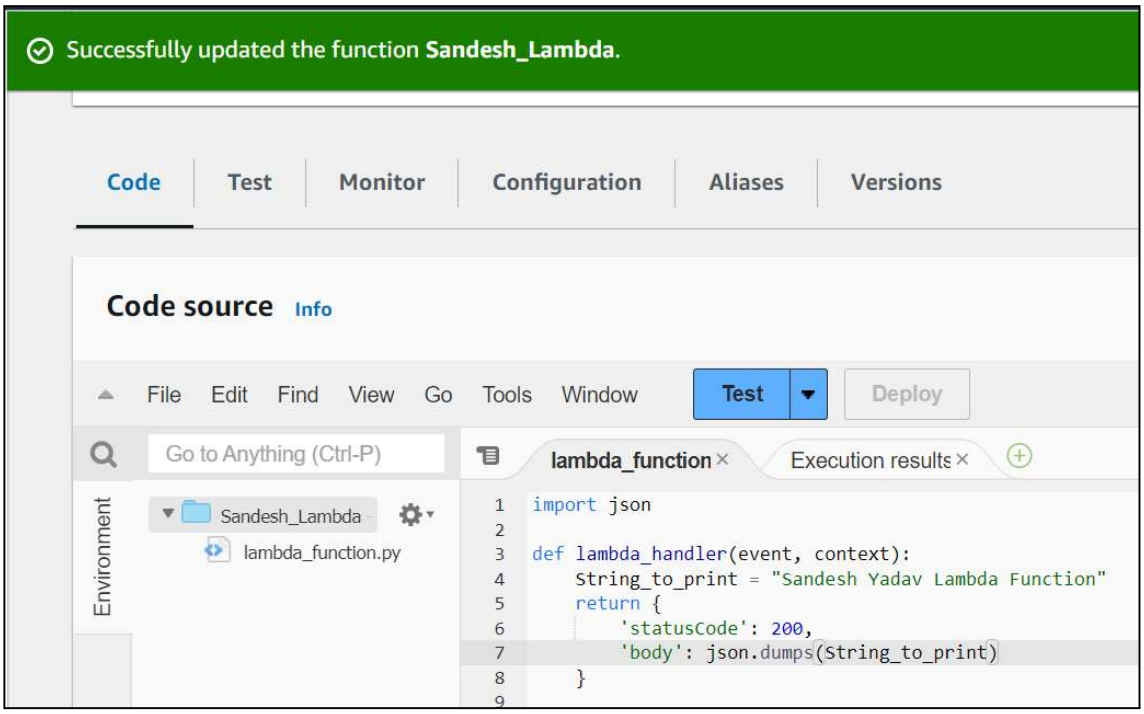




You select the created event to run the specific test you set up, and clicking 'Test' executes your Lambda function to check if it works as expected and produces the desired output

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





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 created and deployed an AWS Lambda function using Python. Key steps included setting up the function, adjusting the timeout, creating a test event, and modifying the code. AWS Lambda simplifies development by handling infrastructure, allowing developers to focus on code, while automatically scaling and managing servers.

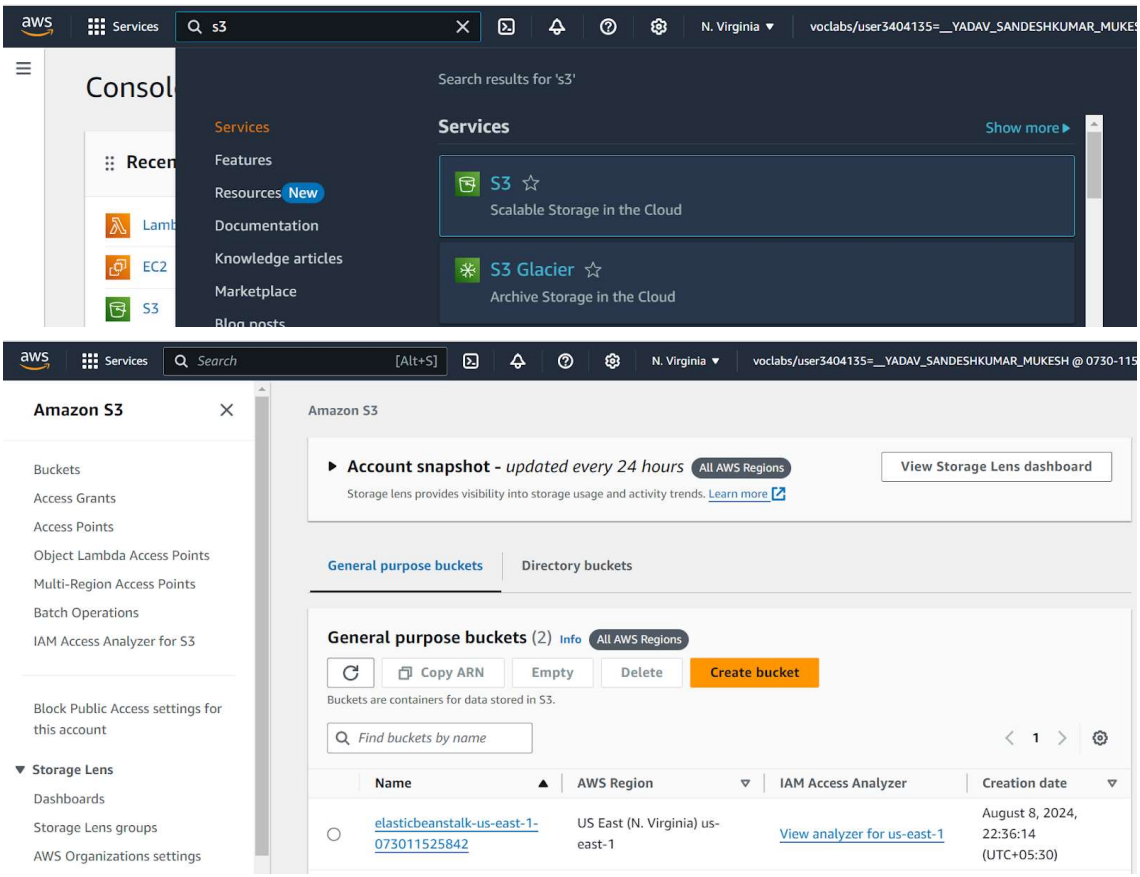
Name : Sandeshkumar.M.Yadav **Div : D15C** **Roll : 61**

Experiment No :12

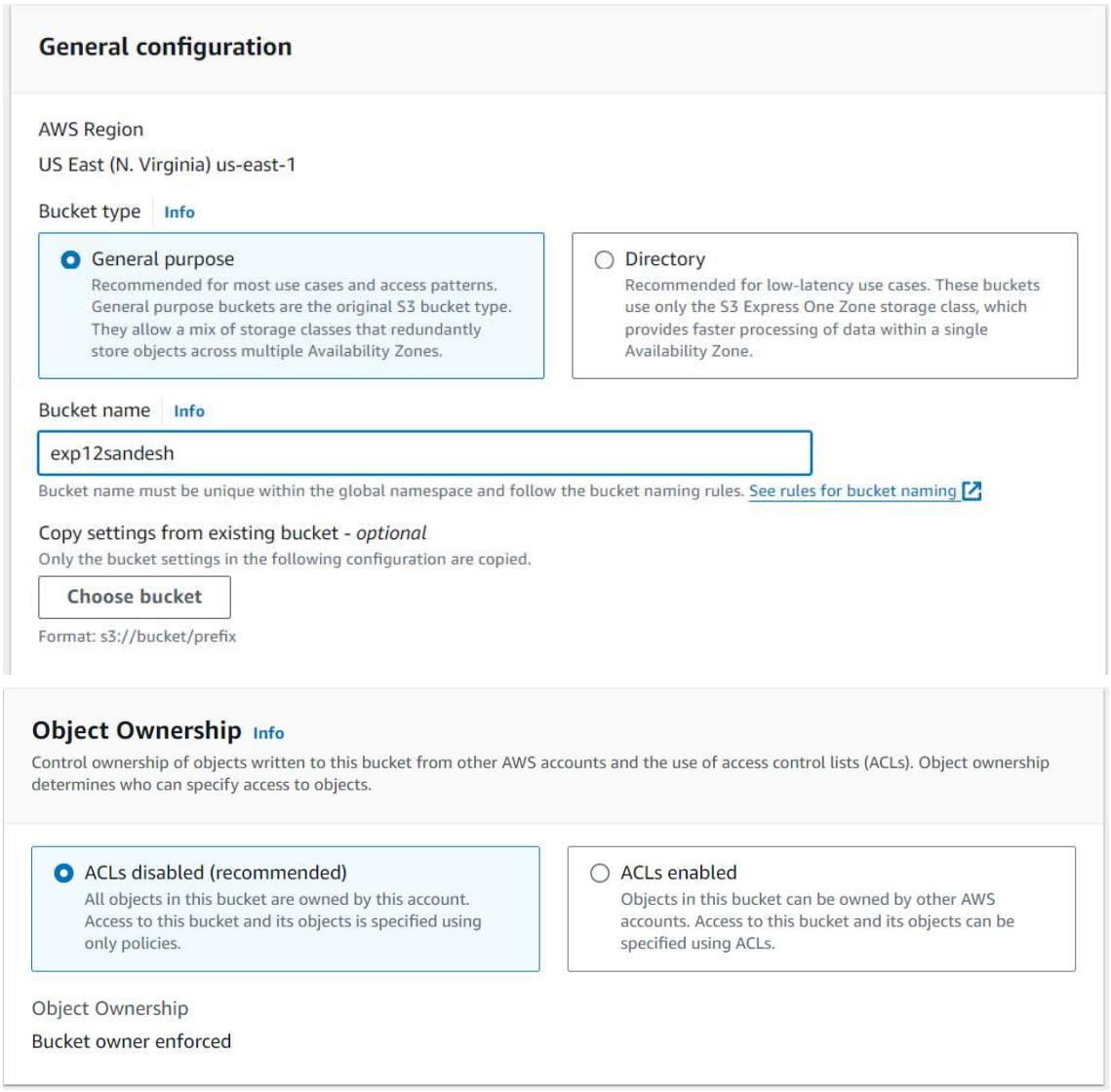
AIM : To create a Lambda function which will log “An Image has been added” once you add an object to a specific bucket in S3

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.



Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. Learn more

- ☐ **Block *all* public access**
  Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.
  - ☐ **Block public access to buckets and objects granted through *new* access control lists (ACLs)**
    S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
  - ☐ **Block public access to buckets and objects granted through *any* access control lists (ACLs)**
    S3 will ignore all ACLs that grant public access to buckets and objects.
  - ☐ **Block public access to buckets and objects granted through *new* public bucket or access point policies**
    S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
  - ☐ **Block public and cross-account access to buckets and objects through *any* public bucket or access point policies**
    S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

⚠ **Turning off block all public access might result in this bucket and the objects within becoming public**

AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.

☐ I acknowledge that the current settings might result in this bucket and the objects within becoming public.

✓ **Successfully created bucket "exp12sandesh"**
To upload files and folders, or to configure additional bucket settings, choose **View details**.

View details ×

Amazon S3 > Buckets

▸ **Account snapshot** - *updated every 24 hours* All AWS Regions
Storage lens provides visibility into storage usage and activity trends. Learn more

View Storage Lens dashboard

General purpose buckets | Directory buckets

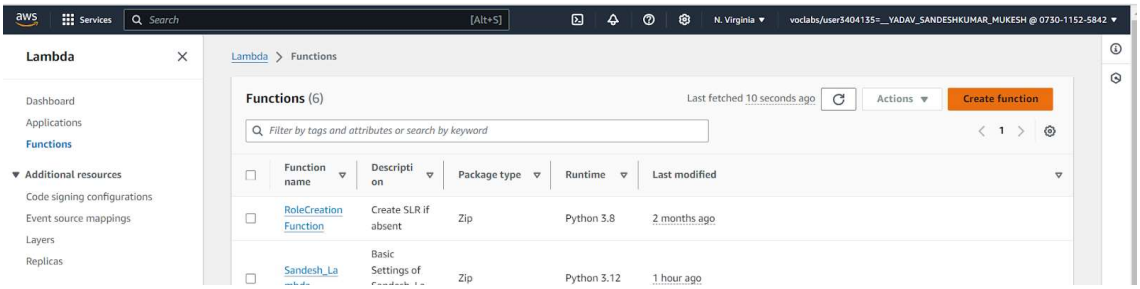
General purpose buckets (3) Info All AWS Regions
Buckets are containers for data stored in S3.

Copy ARN | Empty | Delete | Create bucket

Find buckets by name

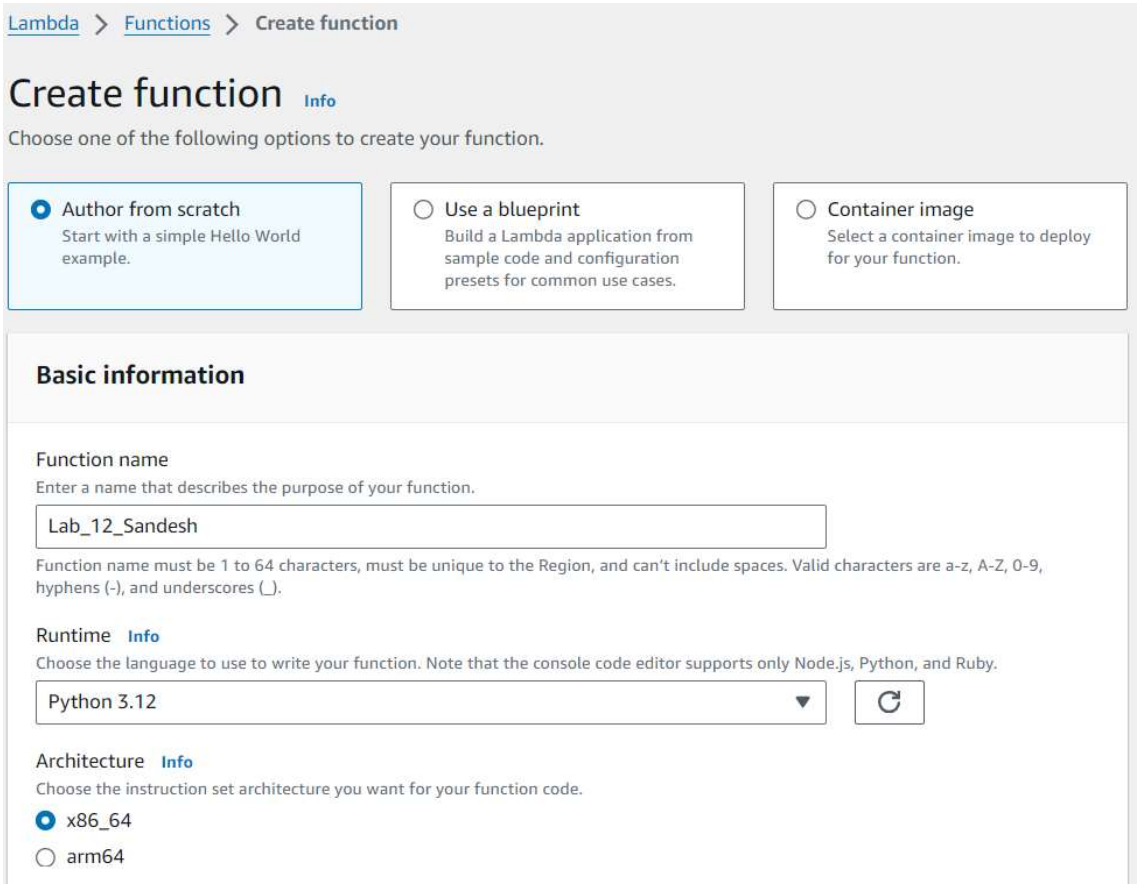
Name	AWS Region	IAM Access Analyzer	Creation date
elasticbeanstalk-us-east-1-073011525842	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 8, 2024, 22:36:14 (UTC+05:30)
exp12sandesh	US East (N. Virginia) us-east-1	View analyzer for us-east-1	October 10, 2024, 20:30:38 (UTC+05:30)
website-for-video	US East (N. Virginia) us-east-1	View analyzer for us-east-1	September 26, 2024, 20:03:00 (UTC+05:30)

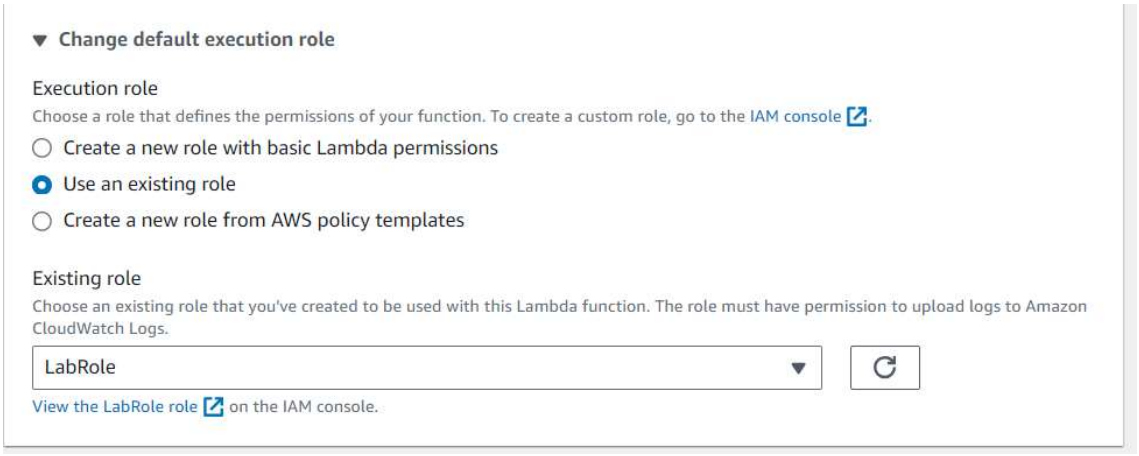
Step 3: Open lambda console and click on create function button.

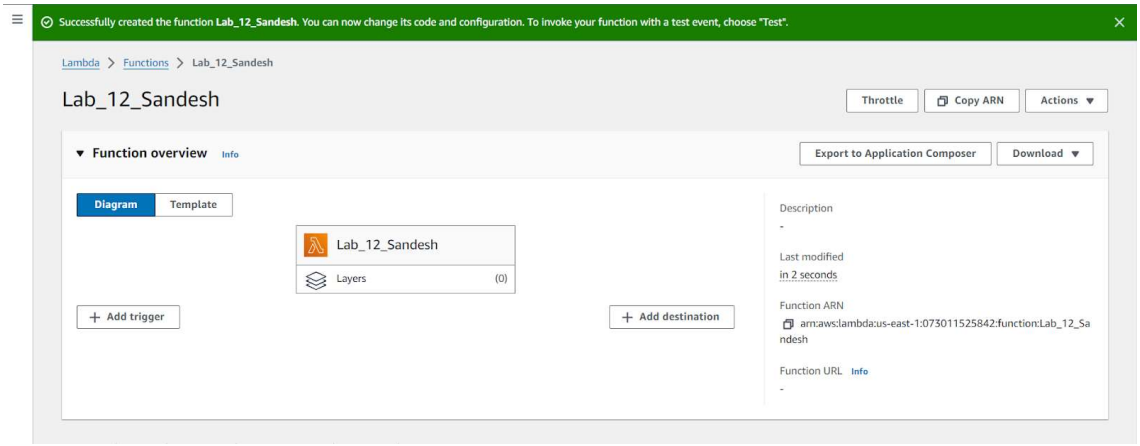


Step 4: Give your Lambda function a name and choose a programming language. The code editor only supports Node.js, Python, and Ruby, so in my case I have chosen Python 3.12. Set the architecture to x86. For the execution role, select 'Use an existing role,' then pick 'Lab role' from the dropdown menu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

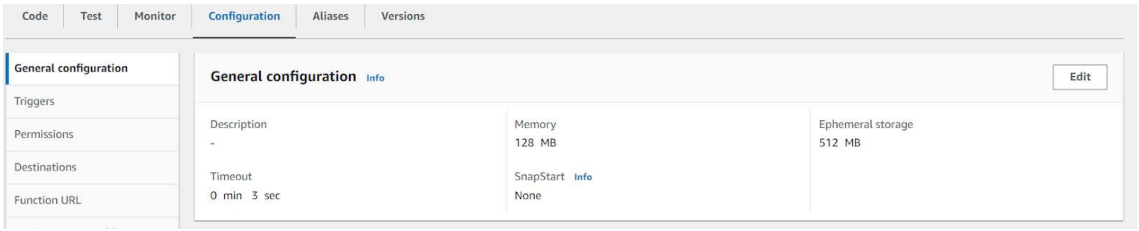
(This is because the Lab role already has the permissions needed for Lambda to run properly, so you don't need to create a new role from scratch. It's a quicker and more convenient option)

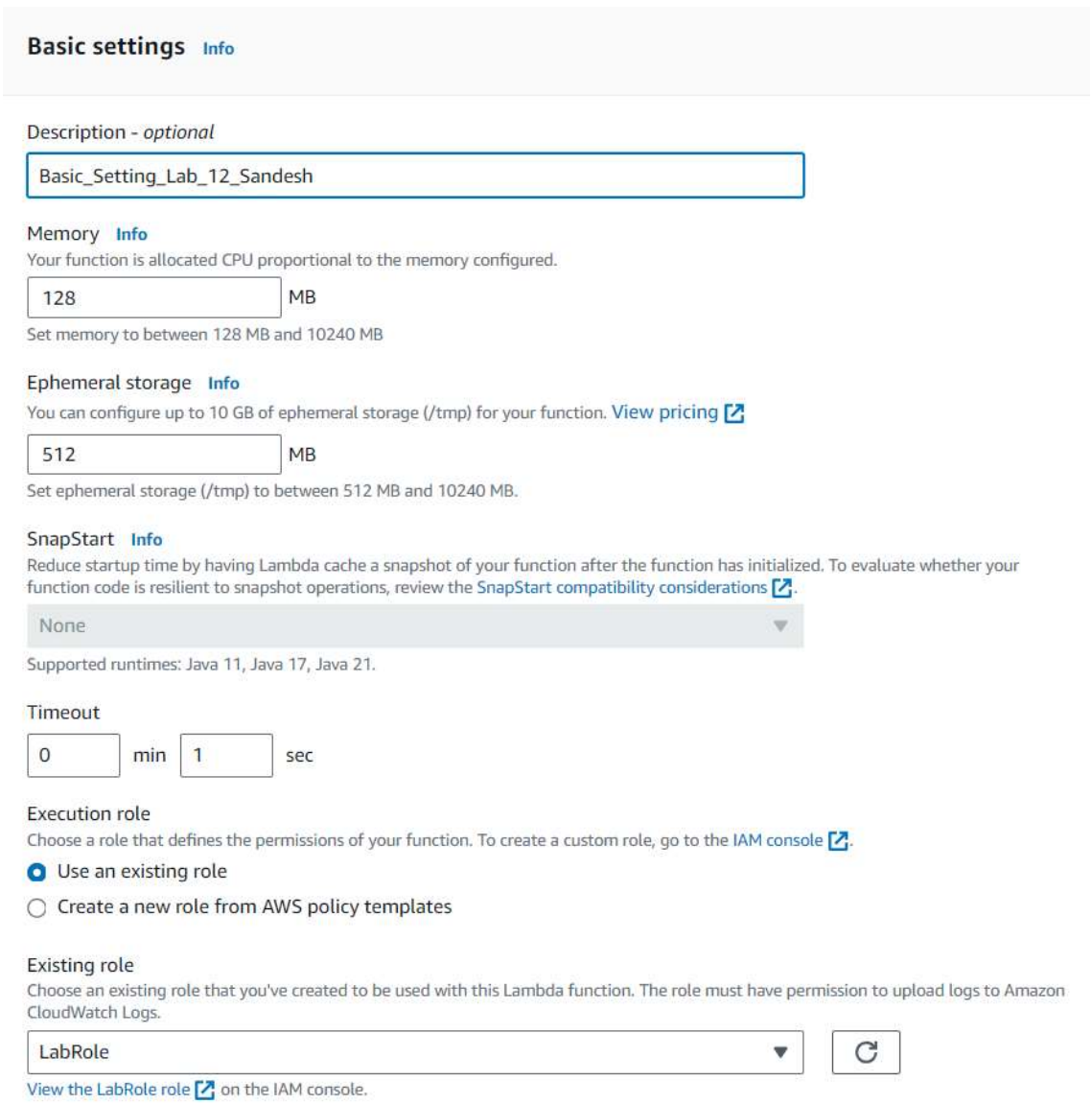



Change default execution role
Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.
Create a new role with basic Lambda permissions
Use an existing role
Create a new role from AWS policy templates
Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.
LabRole
View the LabRole role on the IAM console.

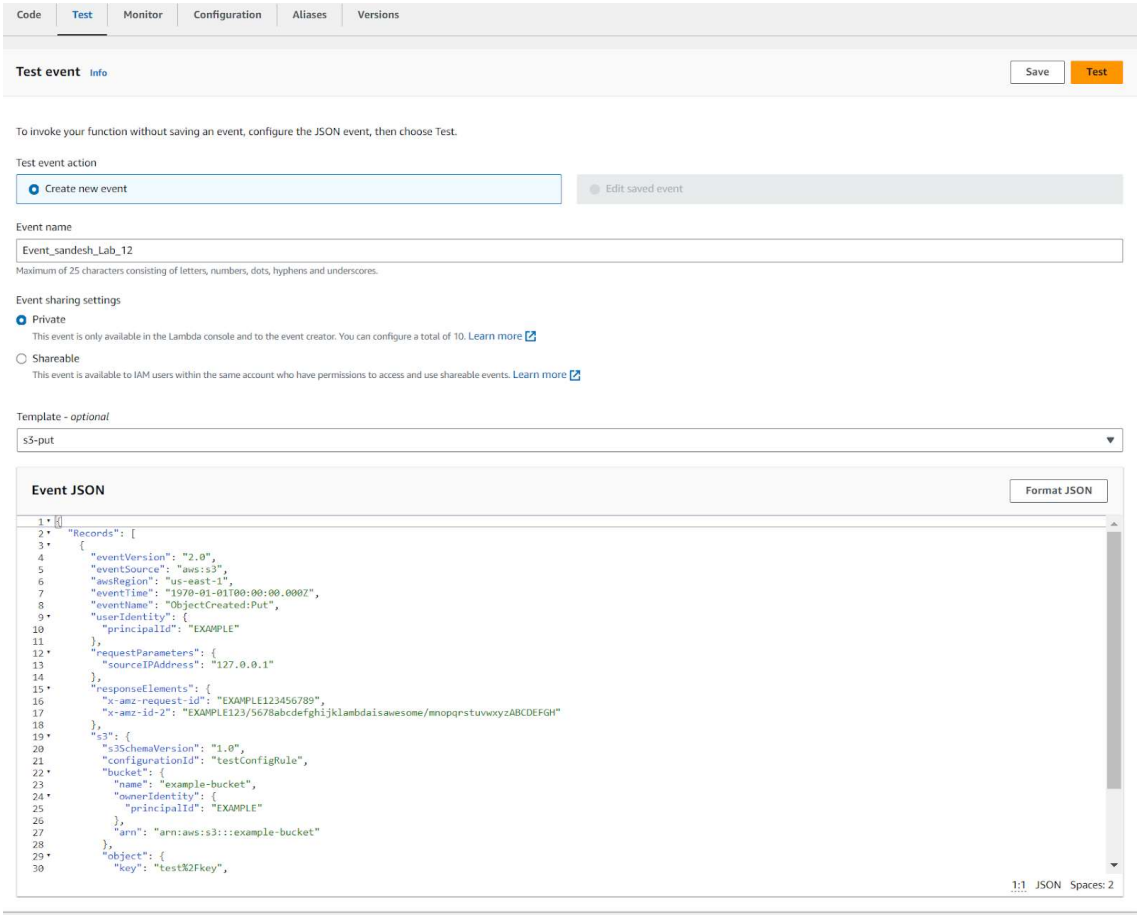

Successfully created the function Lab_12_Sandesh. You can now change its code and configuration. To invoke your function with a test event, choose "Test".
Lambda > Functions > Lab_12_Sandesh
Lab_12_Sandesh
Throttle
Copy ARN
Actions
Function overview Info
Export to Application Composer
Download
Diagram
Template
Lab_12_Sandesh
Layers (0)
Add trigger
Add destination
Description
-
Last modified
in 2 seconds
Function ARN
arn:aws:lambda:us-east-1:073011525842:function:Lab_12_Sandesh
Function URL Info
-

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

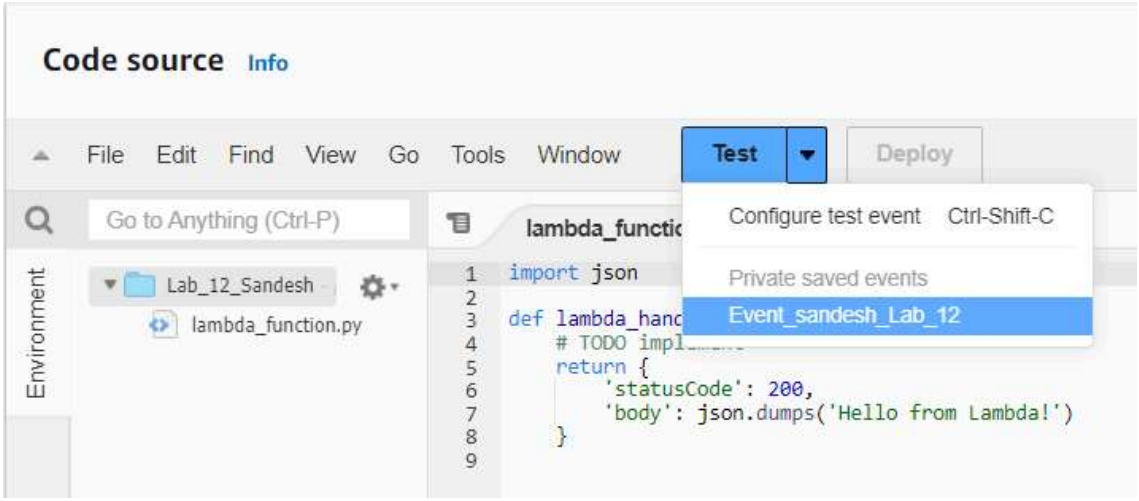




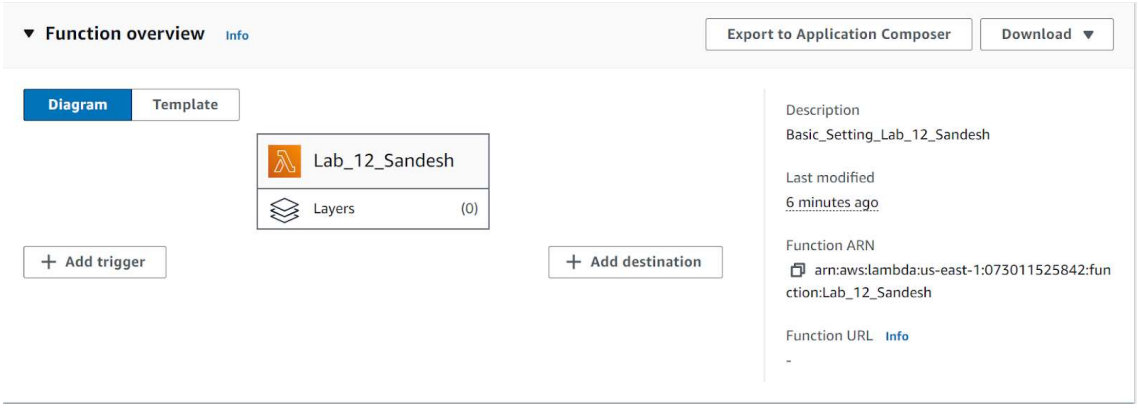
Step 6: Click on the "Test" tab, then select "Create a new event." Give the event a name, set "Event Sharing" to private, and choose the "S3 Put" template.S3 (Simple Storage Service) template allows you to test your Lambda function specifically for events related to uploading files to an S3 bucket.



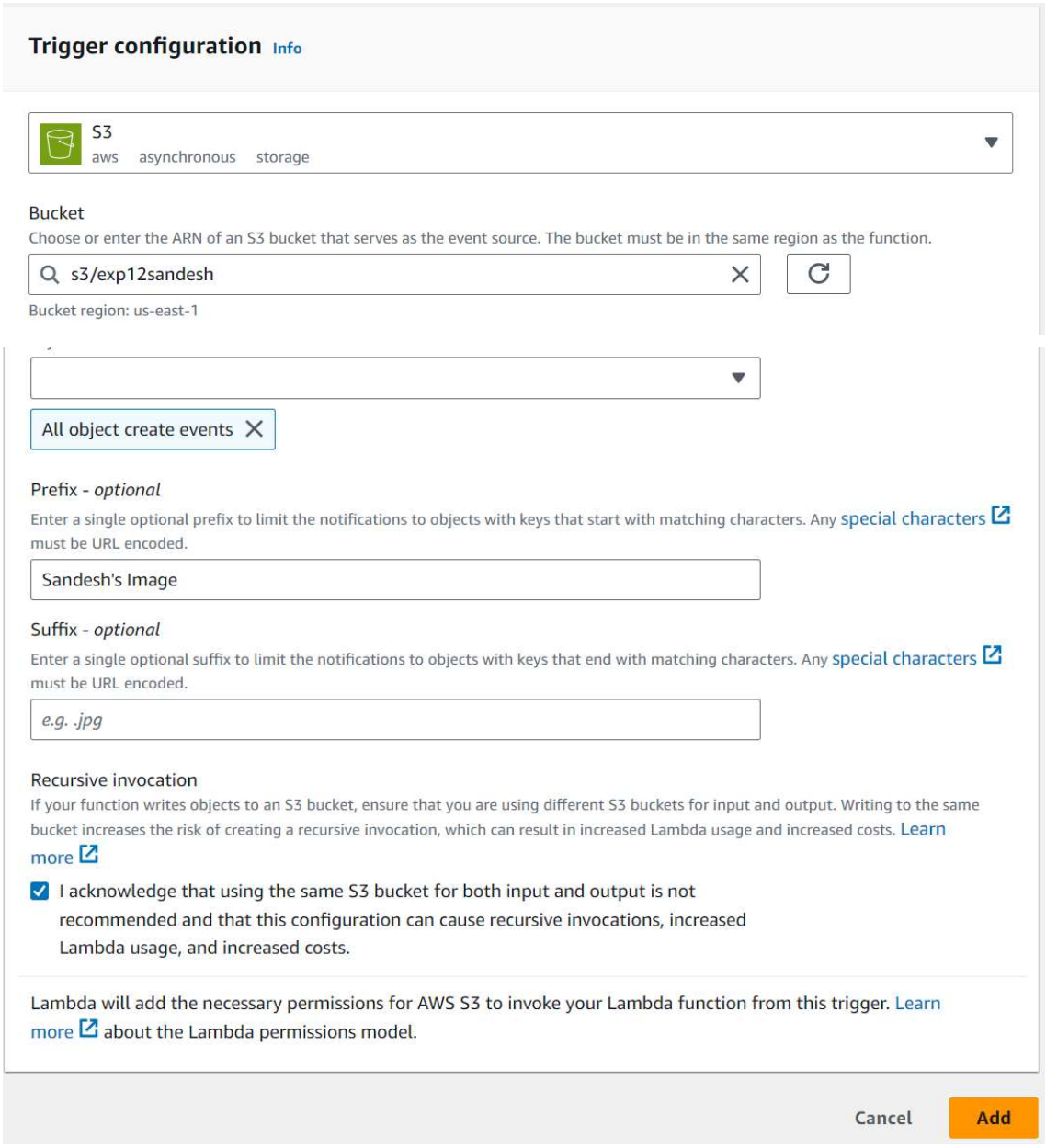
Step 7: Now In the Code section select the created event from the dropdown .

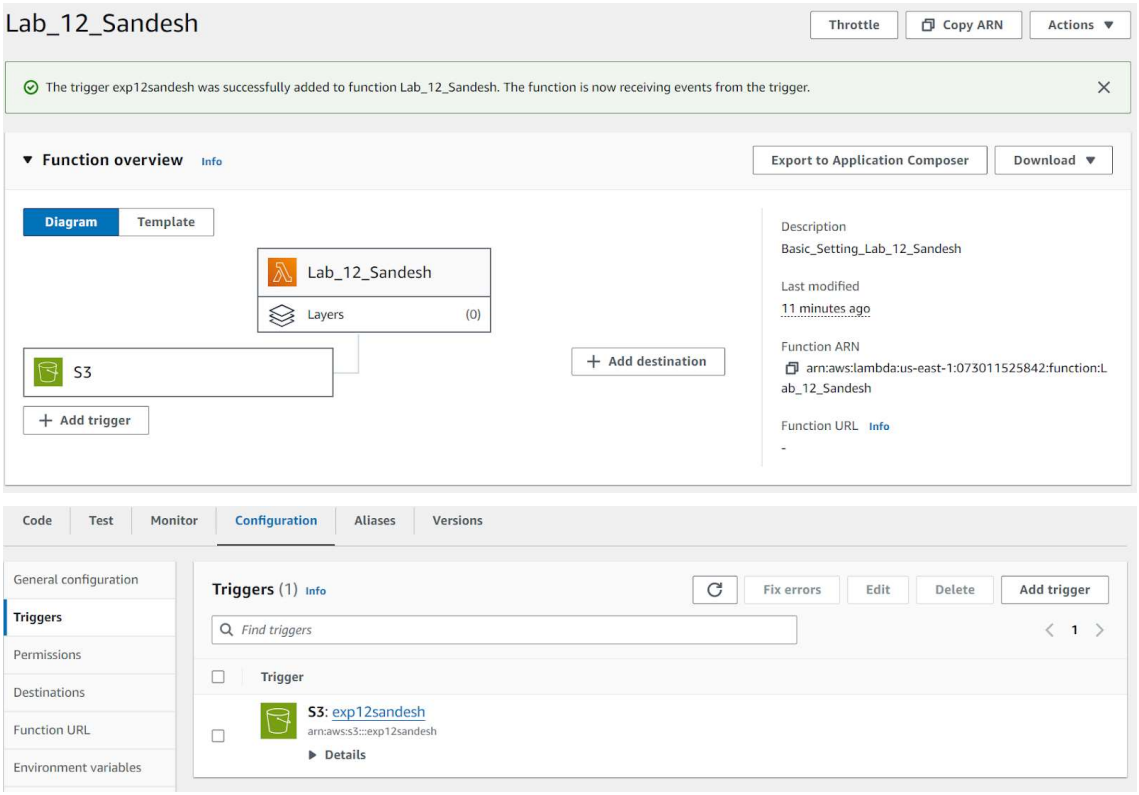


Step 8: In the Lambda function, click on "Add Trigger." Adding a trigger allows your Lambda function to automatically run in response to specific events such as uploads to an S3 bucket



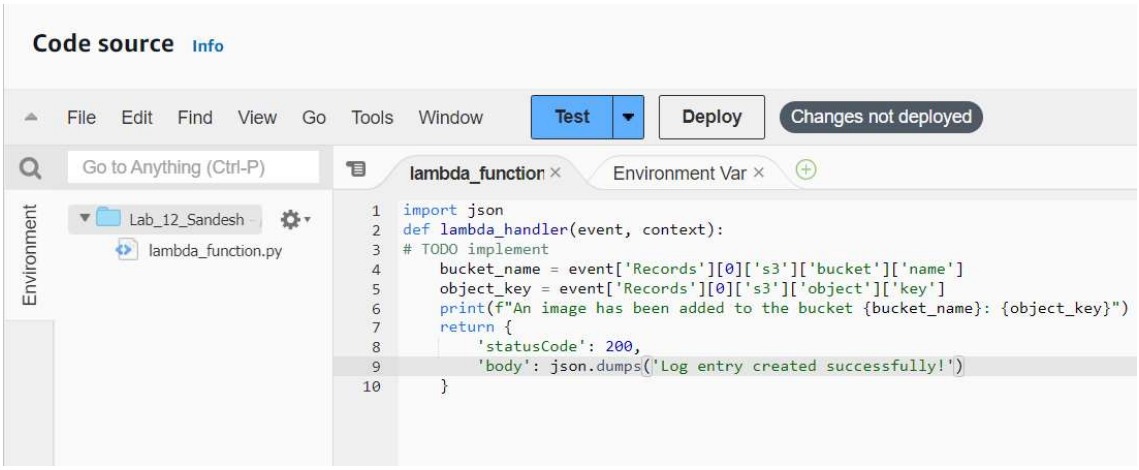
Now select the source as S3, then choose the bucket name from the dropdown menu. Keep the other settings as default, and you can also add a prefix for the image if you want.A prefix for an image (or any file) in S3 is a string that you can use to organize or filter files within a bucket. It acts like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

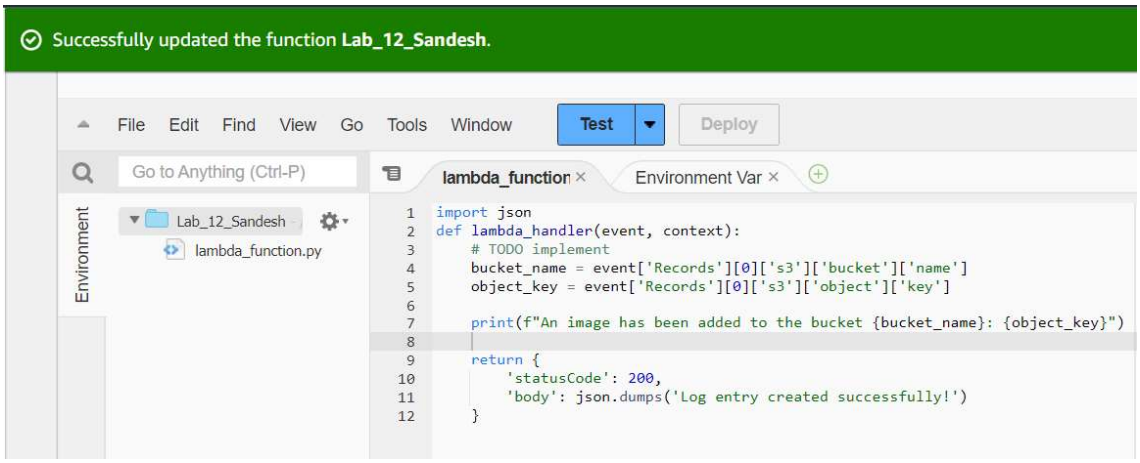

Trigger configuration Info
S3
aws asynchronous storage
Bucket
Choose or enter the ARN of an S3 bucket that serves as the event source. The bucket must be in the same region as the function.
s3/exp12sandesh
Bucket region: us-east-1
All object create events
Prefix - optional
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters. Any special characters must be URL encoded.
Sandesh's Image
Suffix - optional
Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters. Any special characters must be URL encoded.
e.g. .jpg
Recursive invocation
If your function writes objects to an S3 bucket, ensure that you are using different S3 buckets for input and output. Writing to the same bucket increases the risk of creating a recursive invocation, which can result in increased Lambda usage and increased costs. Learn more
I acknowledge that using the same S3 bucket for both input and output is not recommended and that this configuration can cause recursive invocations, increased Lambda usage, and increased costs.
Lambda will add the necessary permissions for AWS S3 to invoke your Lambda function from this trigger. Learn more about the Lambda permissions model.
Cancel
Add



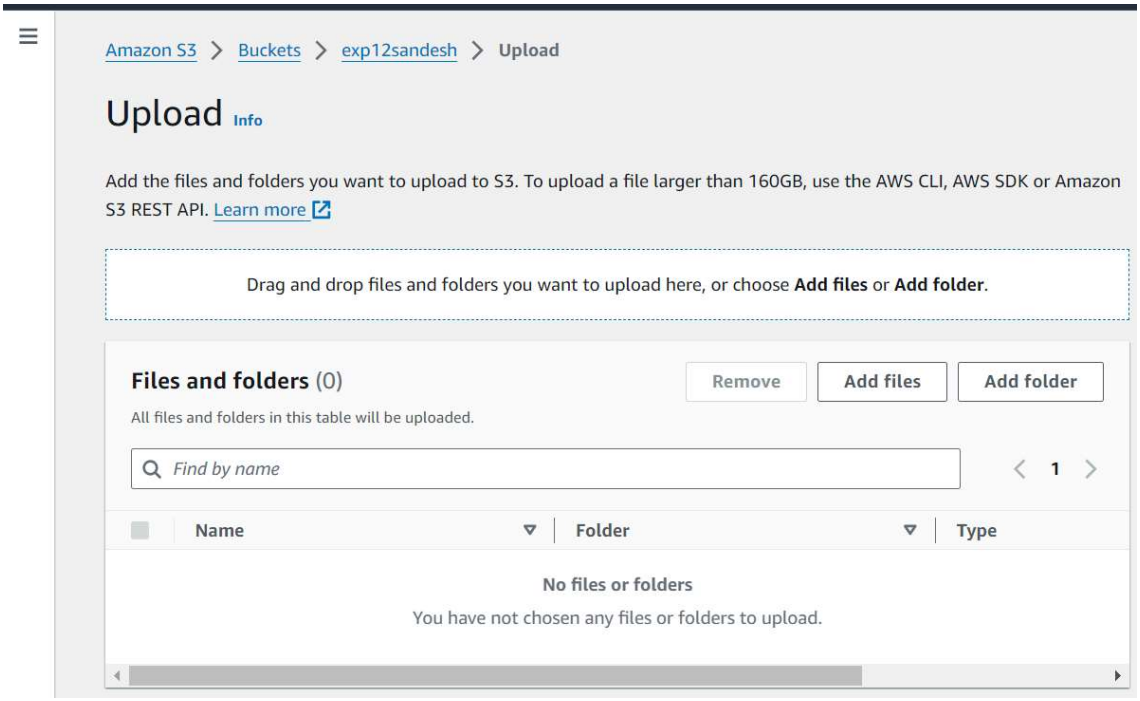
Step 9: Now Write code that logs a message like “An Image has been added” when triggered. Save the file and click on deploy.

```
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!')
}
```

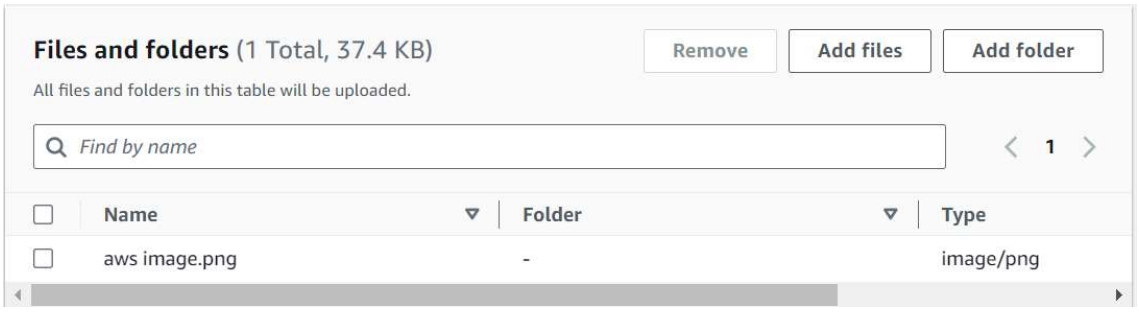




Step 10: Now we will upload any image to the bucket

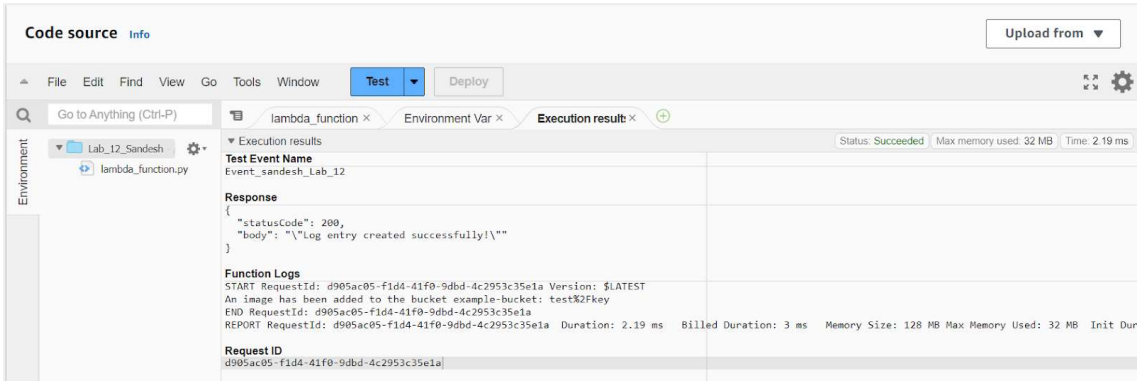


Click on add file where you can upload any image of your choice in your bucket

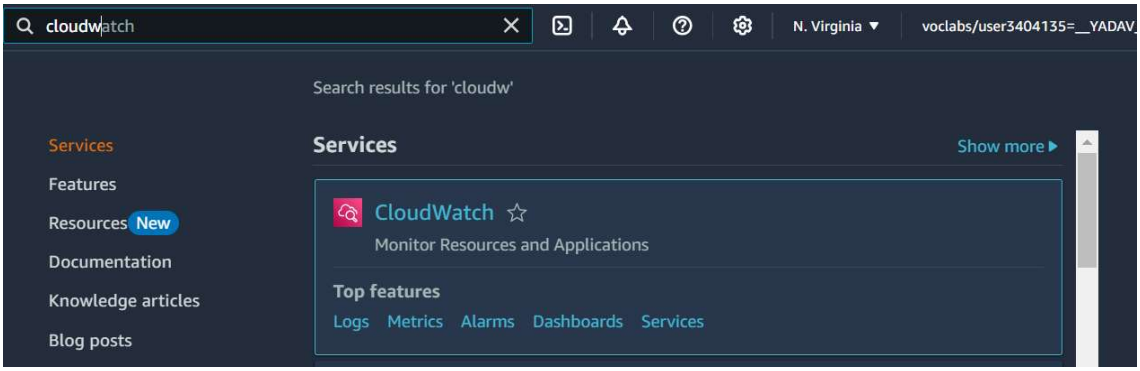


Click on Upload

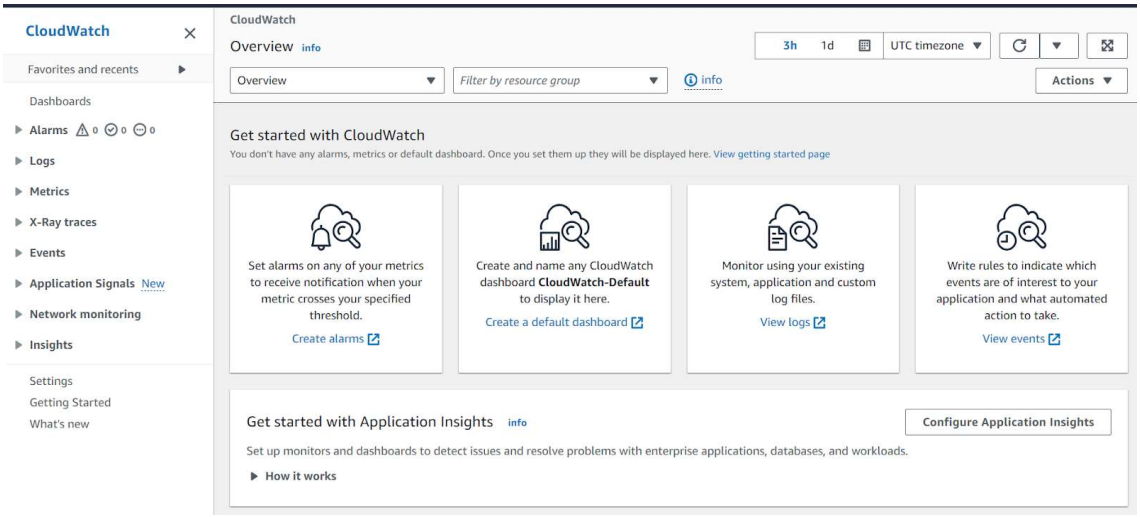
Step 11: Now click on "Test" in Lambda to see if it logs the activity when an image is added to S3.



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

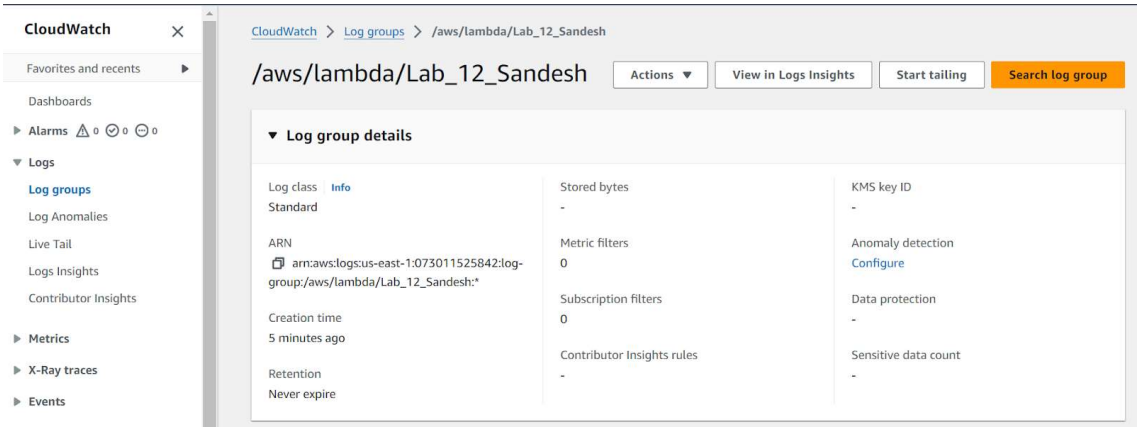


Click on the CloudWatch:

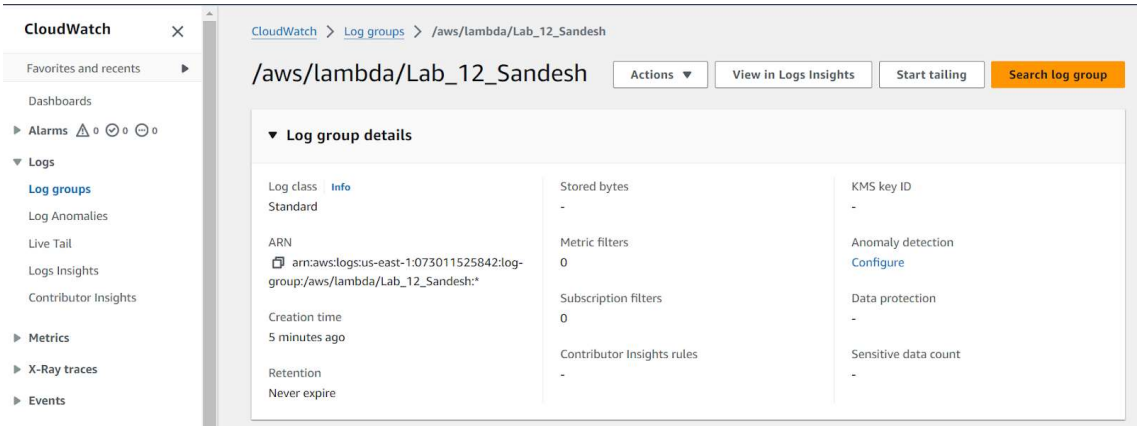


Name : Sandeshkumar.M.Yadav **Div : D15C** **Roll : 61**

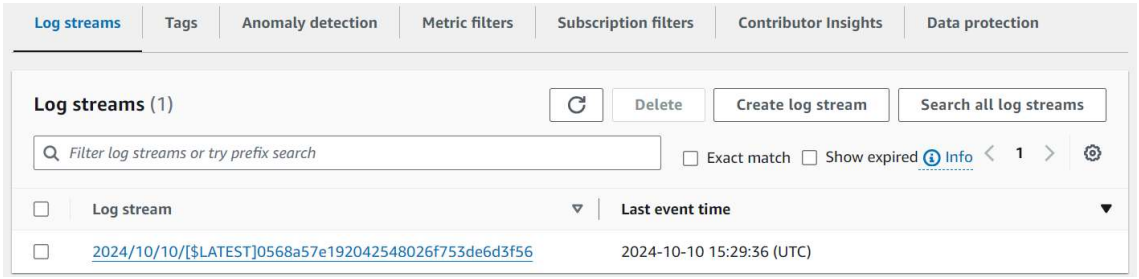
Click on the logs:

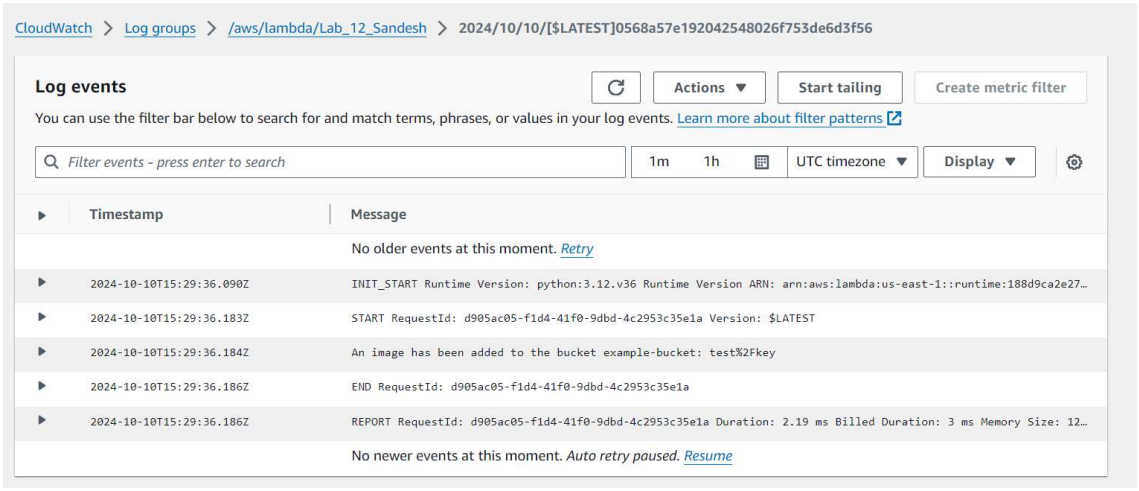


Click on the log group:



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. Using the S3 Put template ensured that the function was triggered correctly by S3 events. The experiment demonstrated Lambda's event-driven architecture, showing how it can automatically respond to file uploads in S3. Additionally, we learned how to troubleshoot common issues with event setup and verify activity logs using CloudWatch.