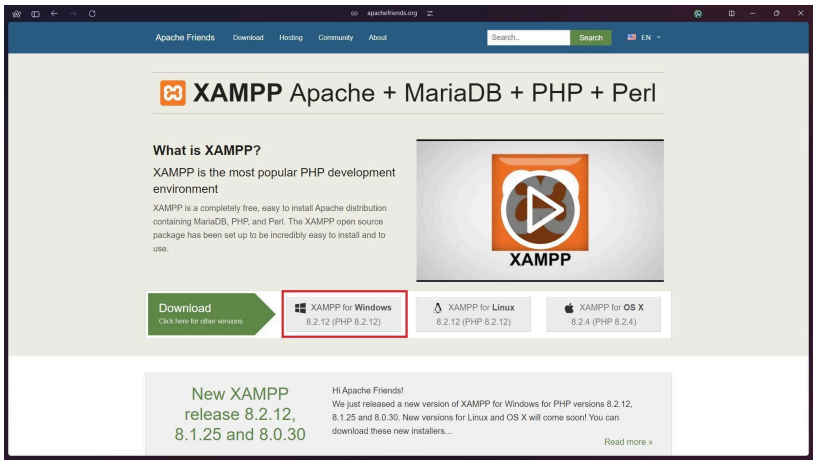
Name: Shreyash Kamat Div: D15C Roll no: 22

Static Hosting:

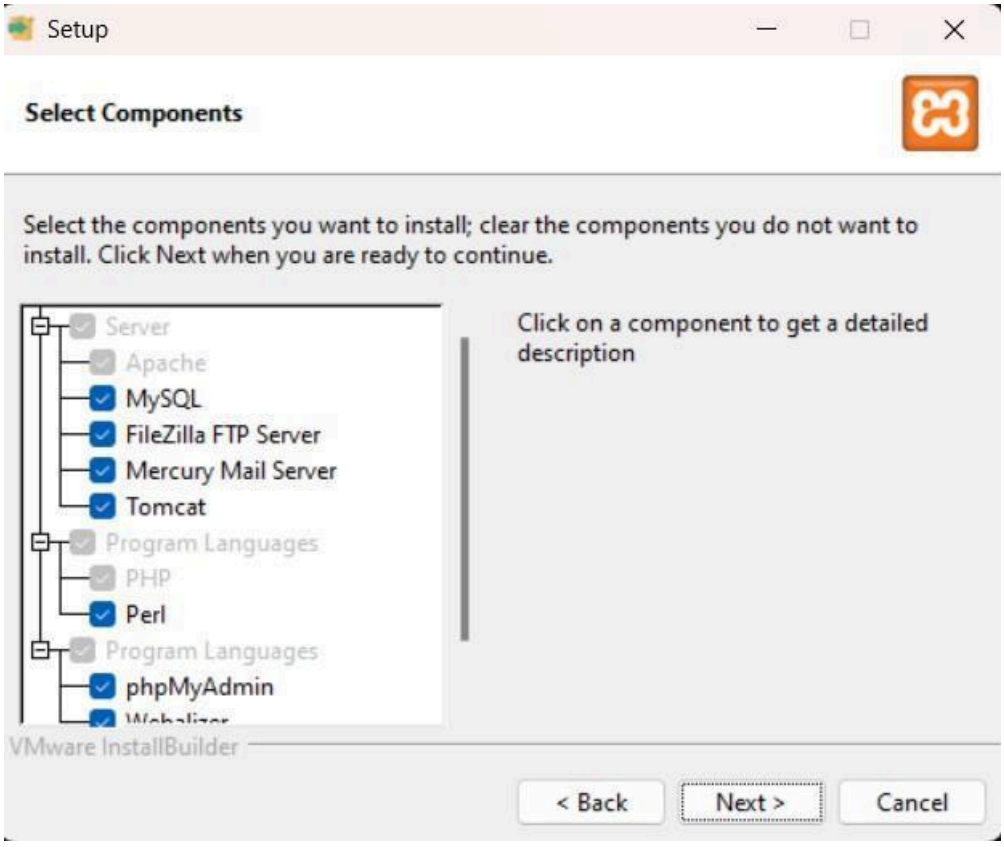
1) On local server (XAMPP)

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

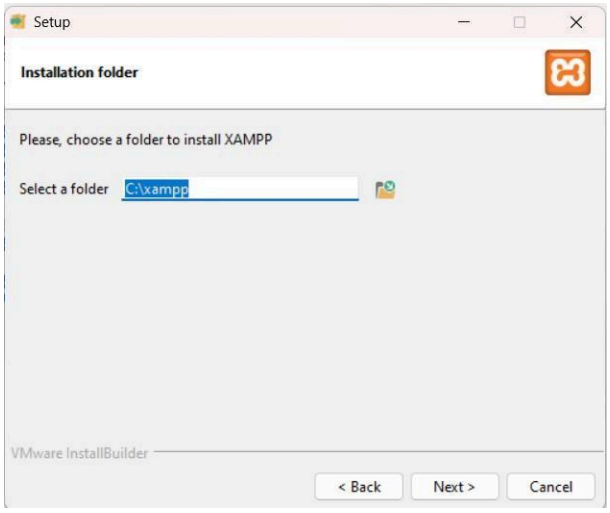
1) Select your OS. It will automatically start downloading.



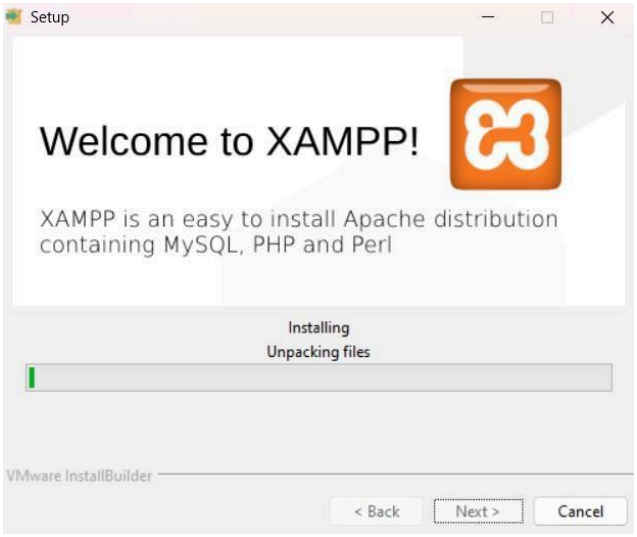
2) Open the setup file. Select all the required components and click next



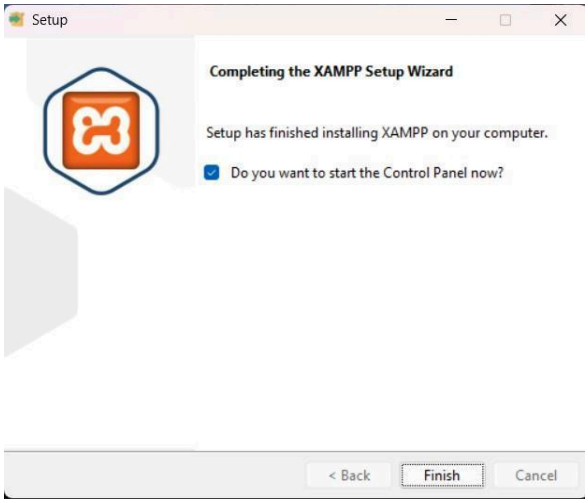
3) Choose the folder to install XAMPP in. Make sure the folder is empty. Click next



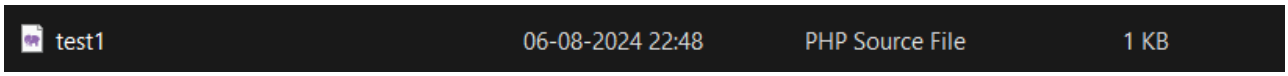
4) Select the language, click next. XAMPP starts to install



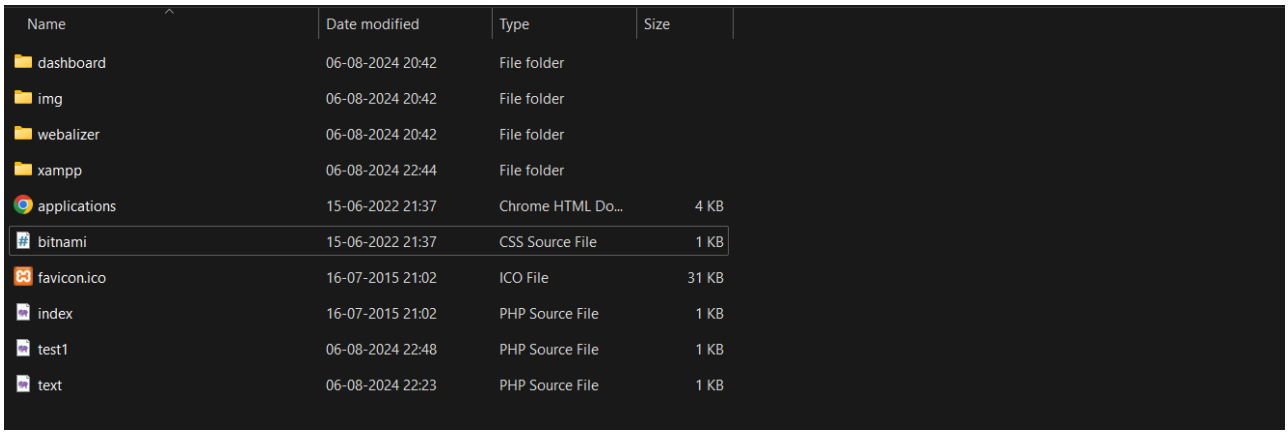
5) The installation is complete. Click Finish



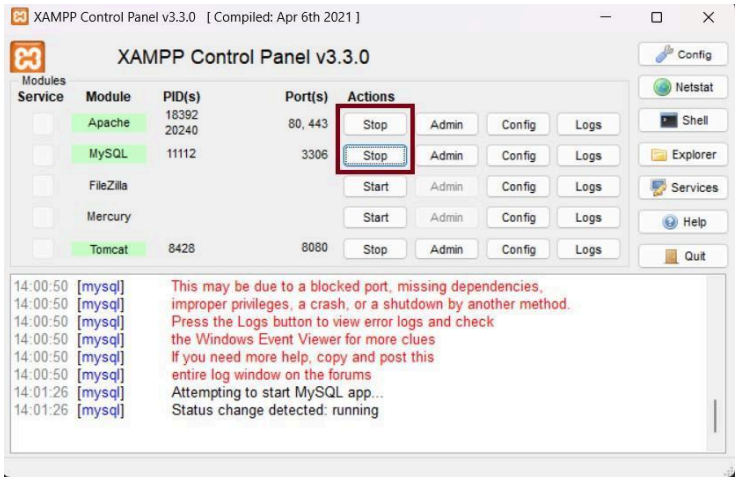
Step 2: Setup a file that is to be hosted on the server. Make sure the file has extension **.php**



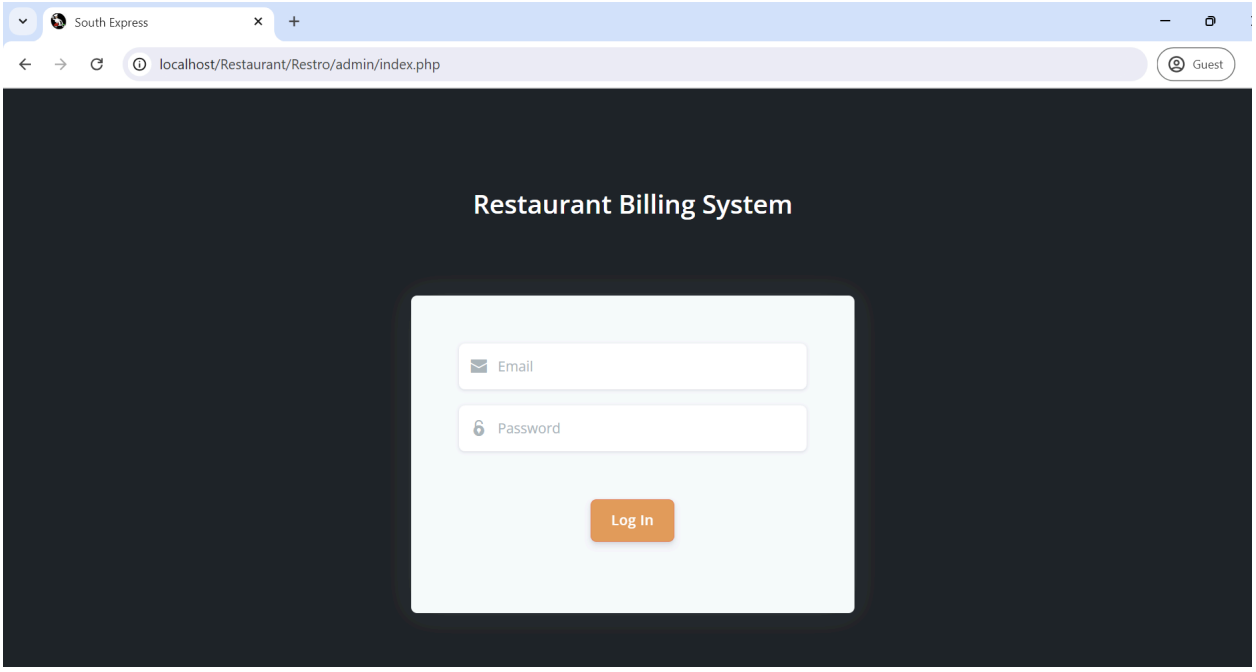
Step 3: Go to the directory where XAMPP was installed. Go to **htdocs** folder. Place your folder in this directory.



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

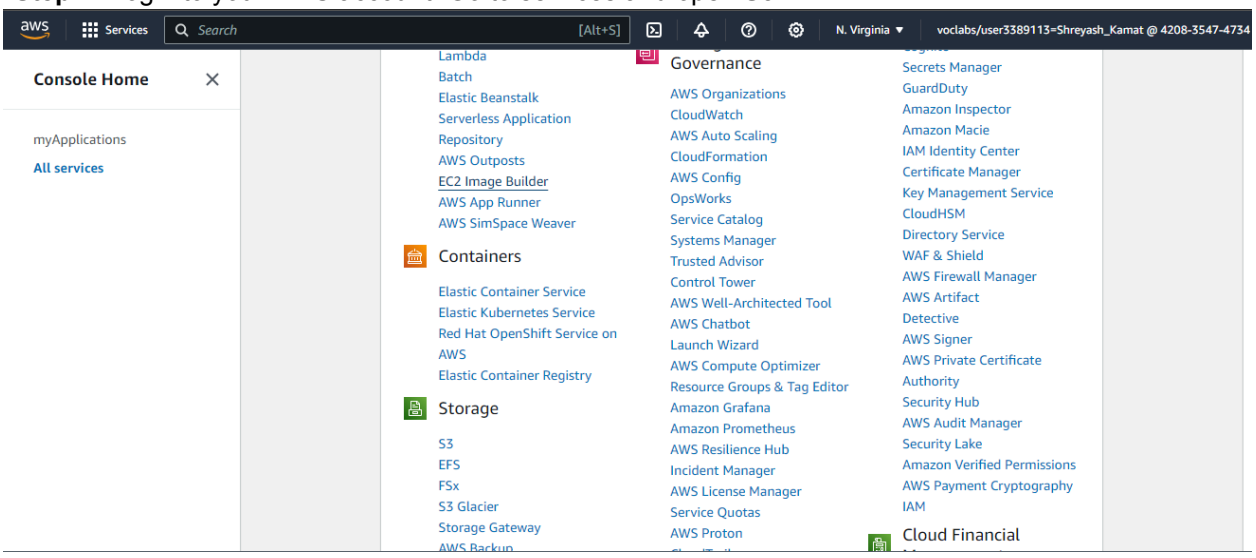


Step 5: Open your web browser. Type localhost/YOUR_FILENAME.php. This will open your website on your browser.

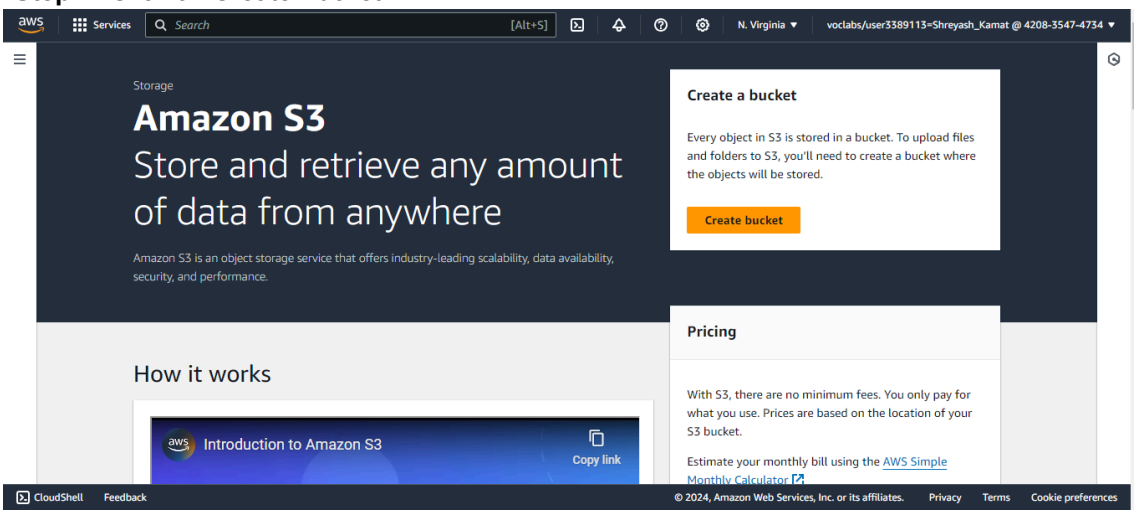


2) AWS S3

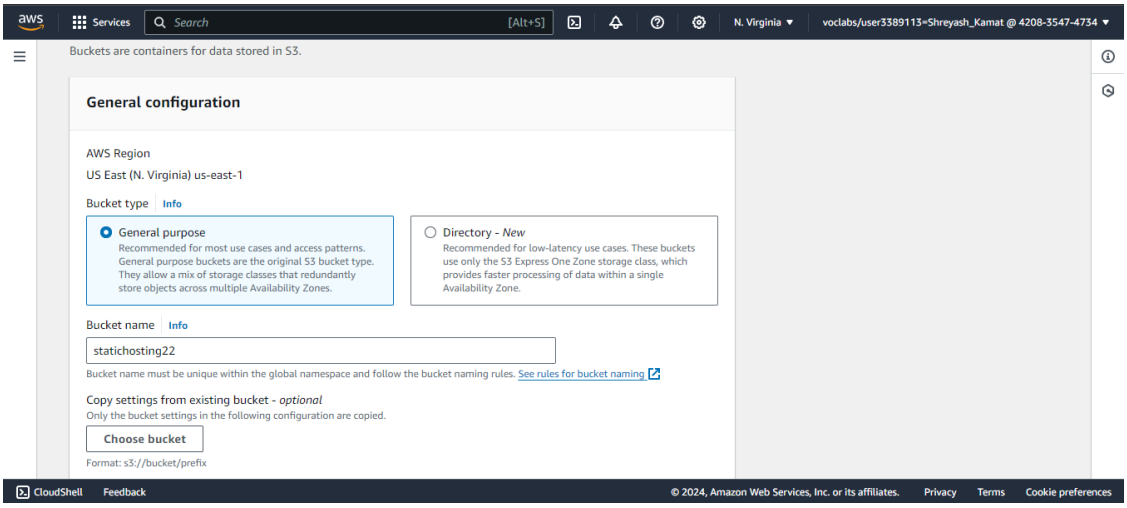
Step 1: Login to your AWS account. Go to services and open **S3.**



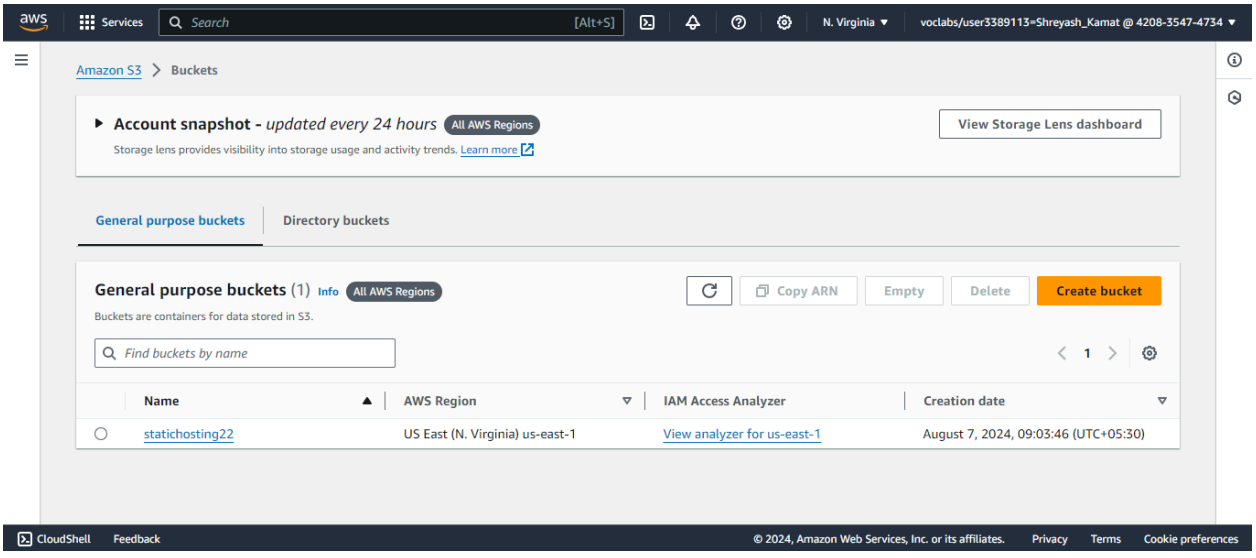
Step 2: Click on Create Bucket

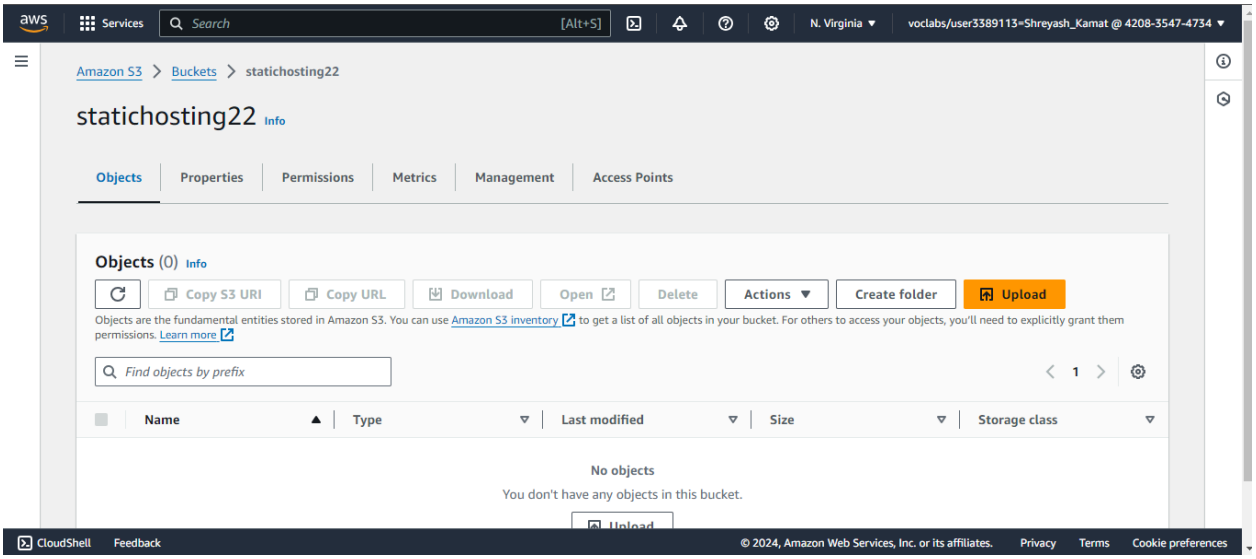


Step 3: Give a name to your bucket, keeping other options default, scroll down and click on Create Bucket

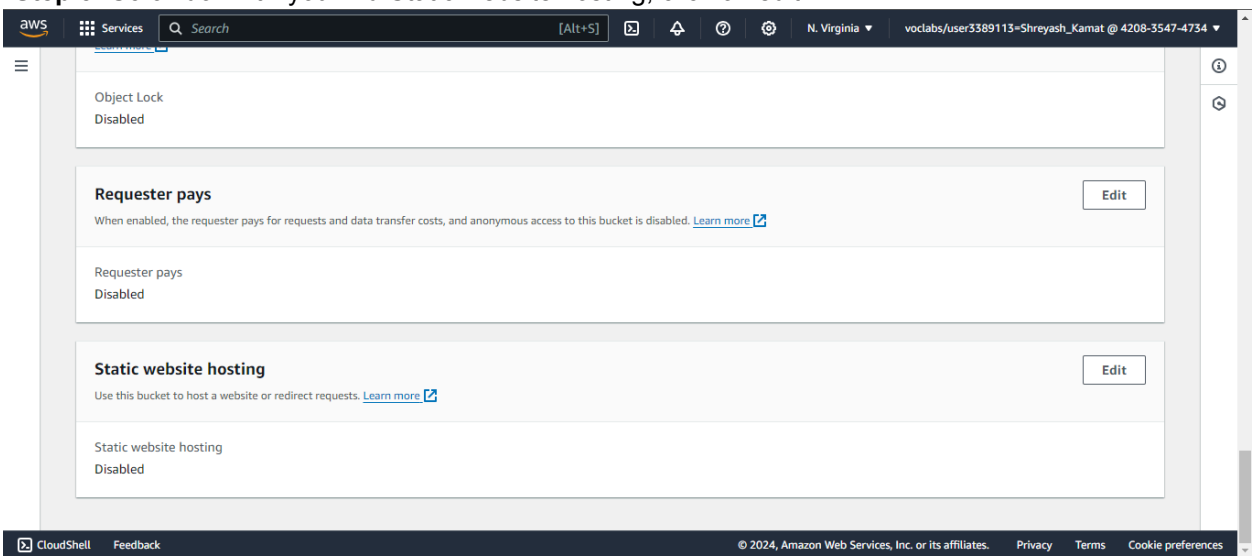


Step 4: Click on the name of your bucket and goto Properties

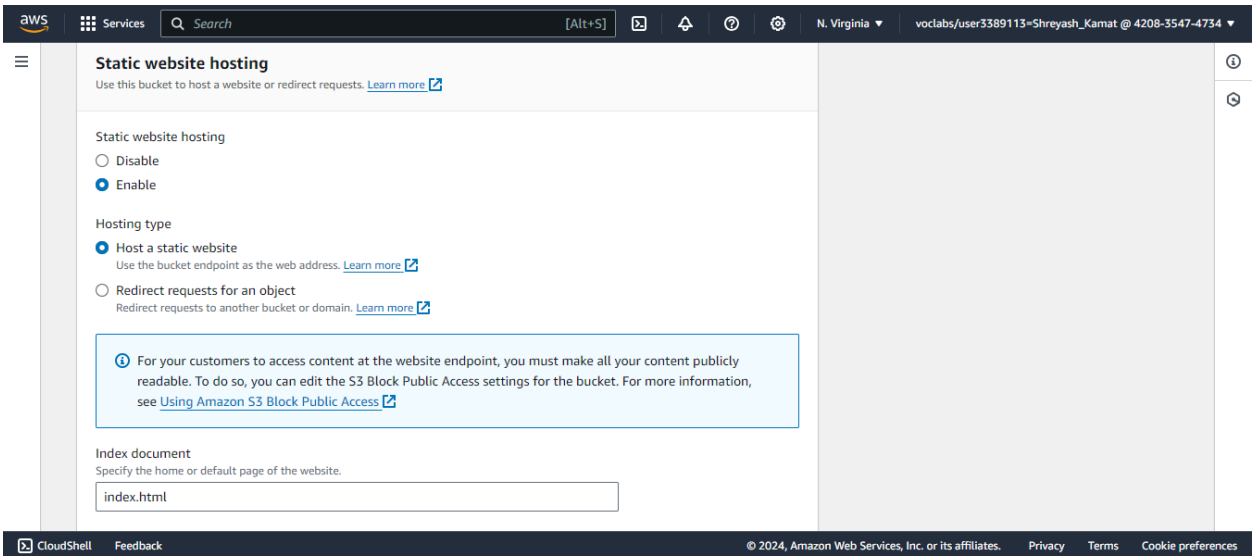




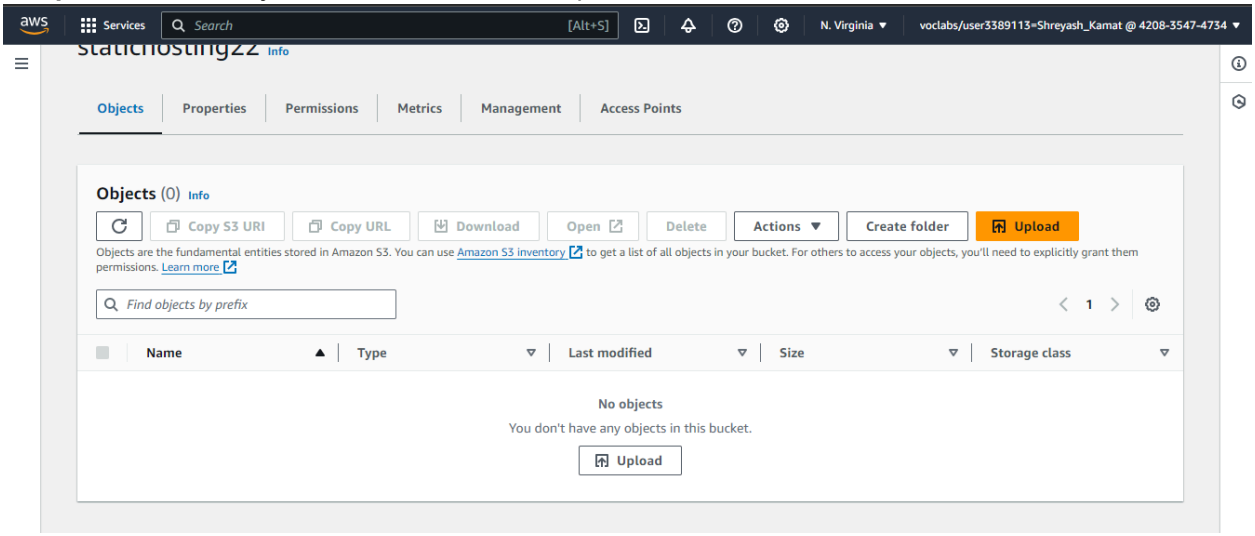
Step 5: Scroll down till you find Static website hosting, click on edit



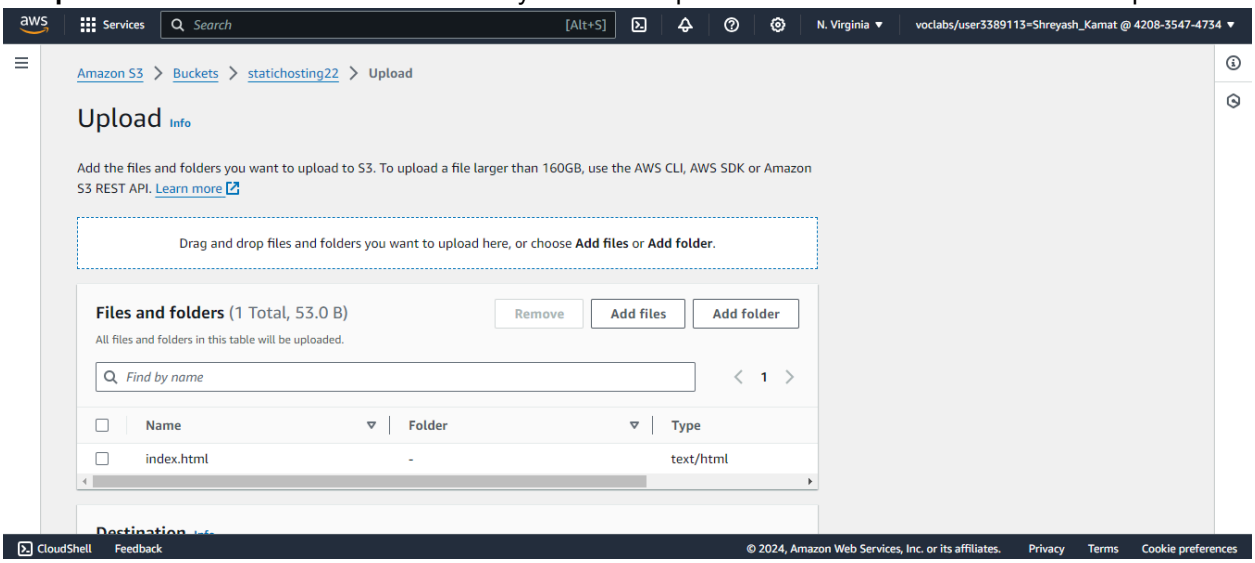
Step 6: Enable static website hosting, in Index document, write the name of your document and in error document, give name as 404.html. Save your changes.



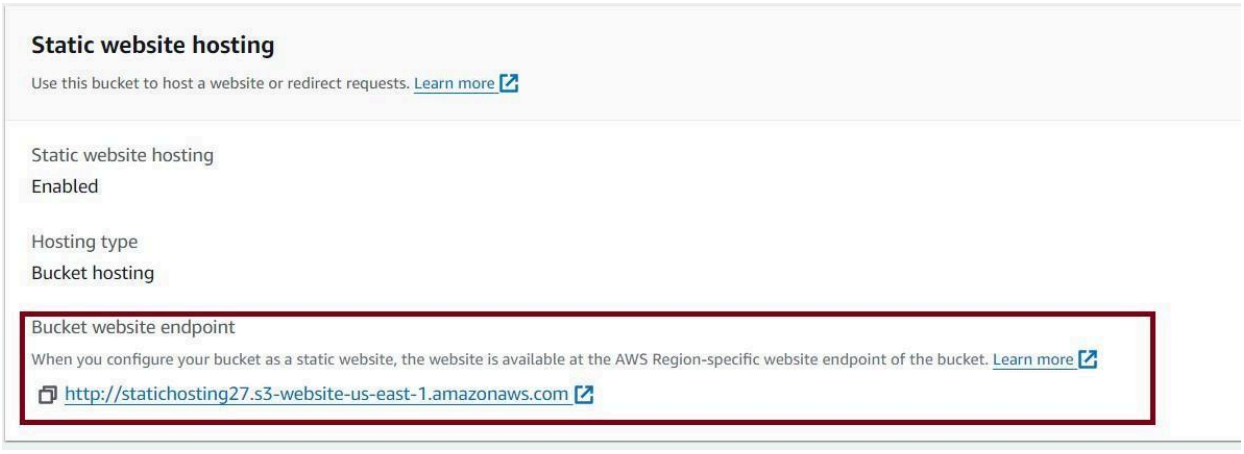
Step 7: Go to the Objects tab and click on the upload file.



Step 8: Click on Add files. Add all the files you want to upload. Then scroll down and click on Upload

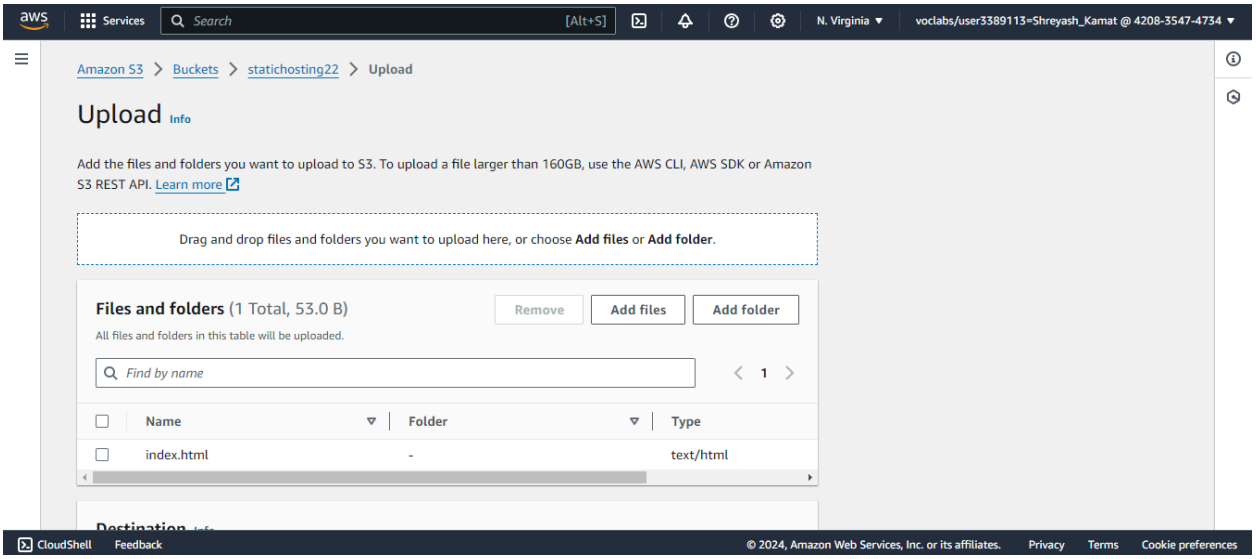


Step 9: This will take you to the Objects screen. Switch to Properties, scroll down to Static web hosting. There you would find the link (Bucket website endpoint) to your website.

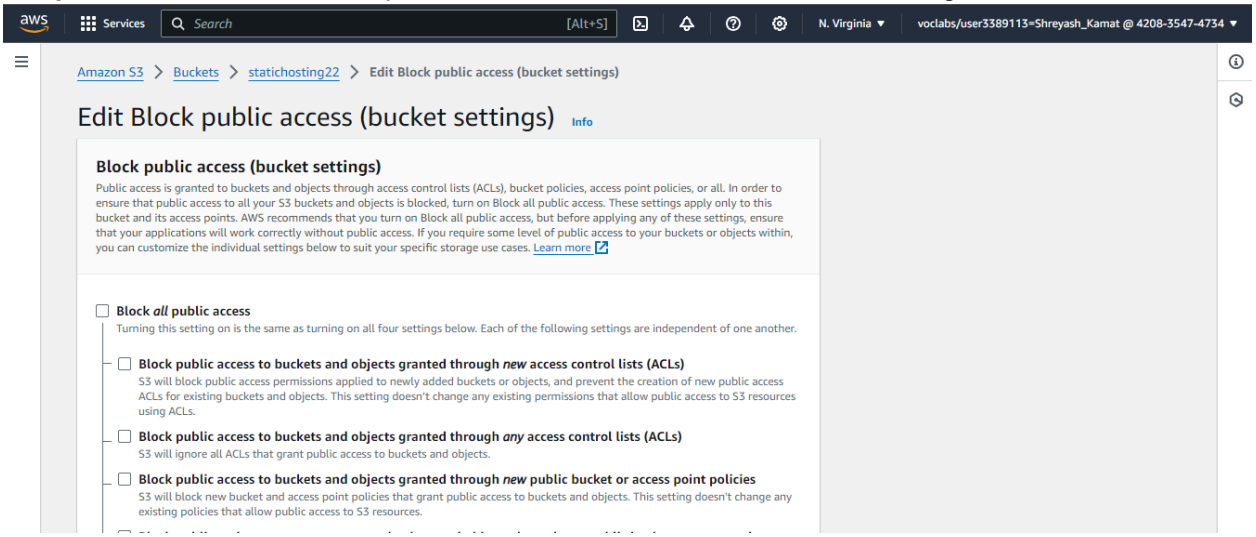


Step 10: Open the link. It will show a 403 forbidden error screen as the contents of the bucket are not

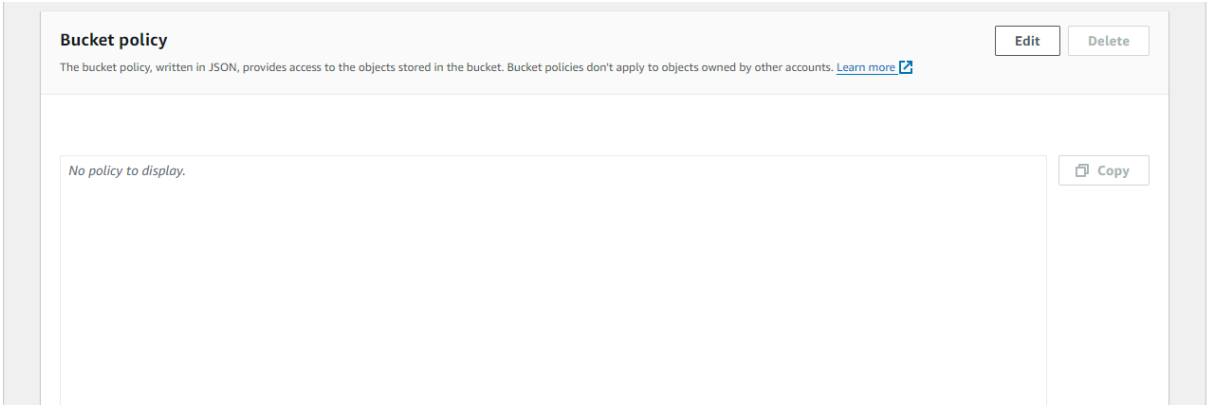
available for the public users. To change this, go to Permissions tab, go to Block public access and click on edit



Step 11: Uncheck the Block all public access checkbox and click on save changes



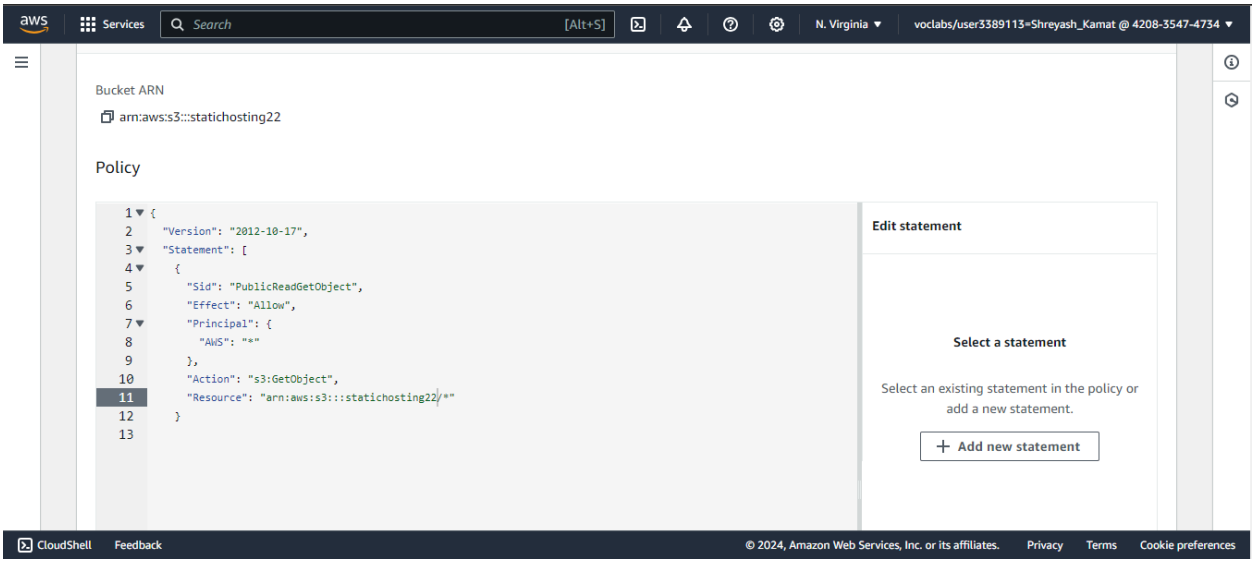
Step 12: Scroll down to bucket policy and click edit



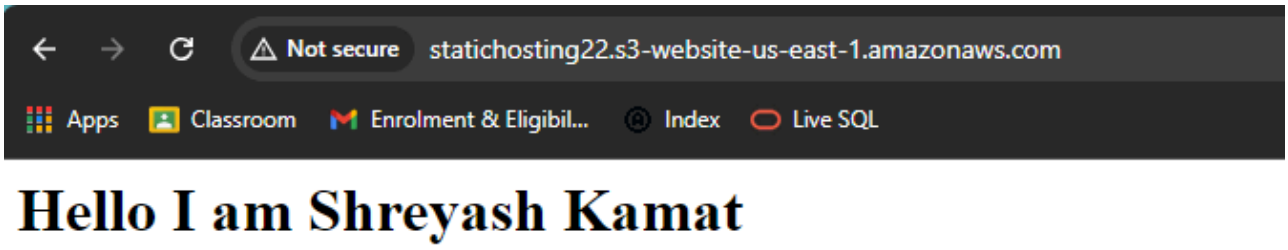
Step 13:

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

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



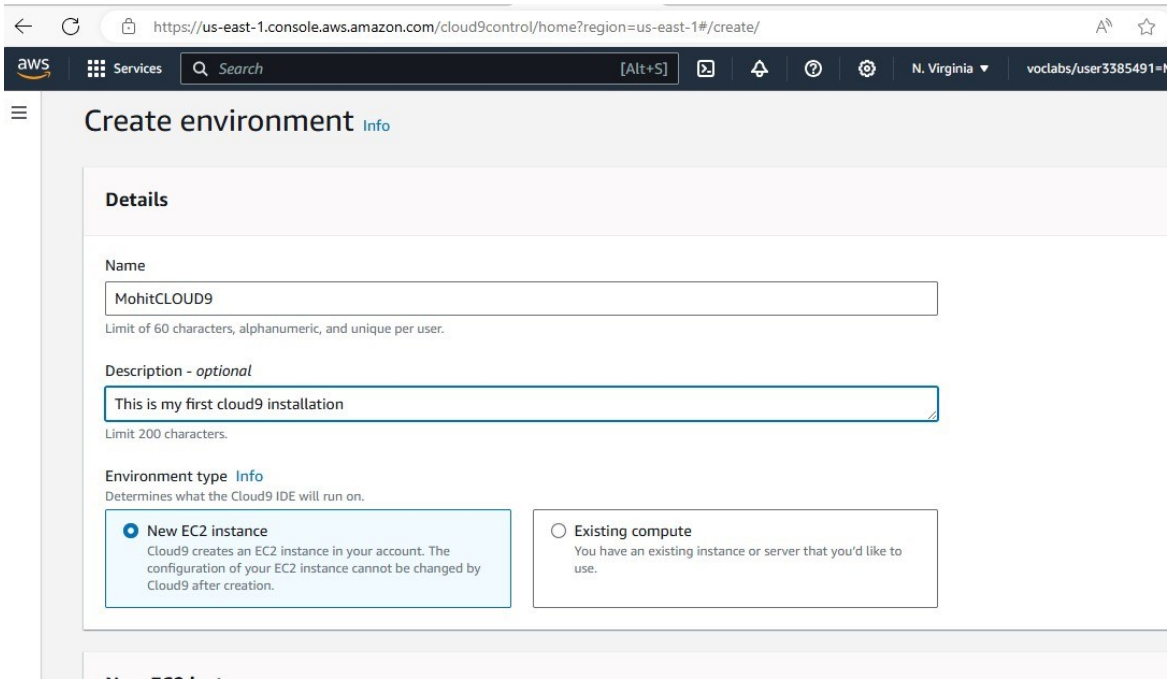
Step 14: Now reload the website. You can see your website

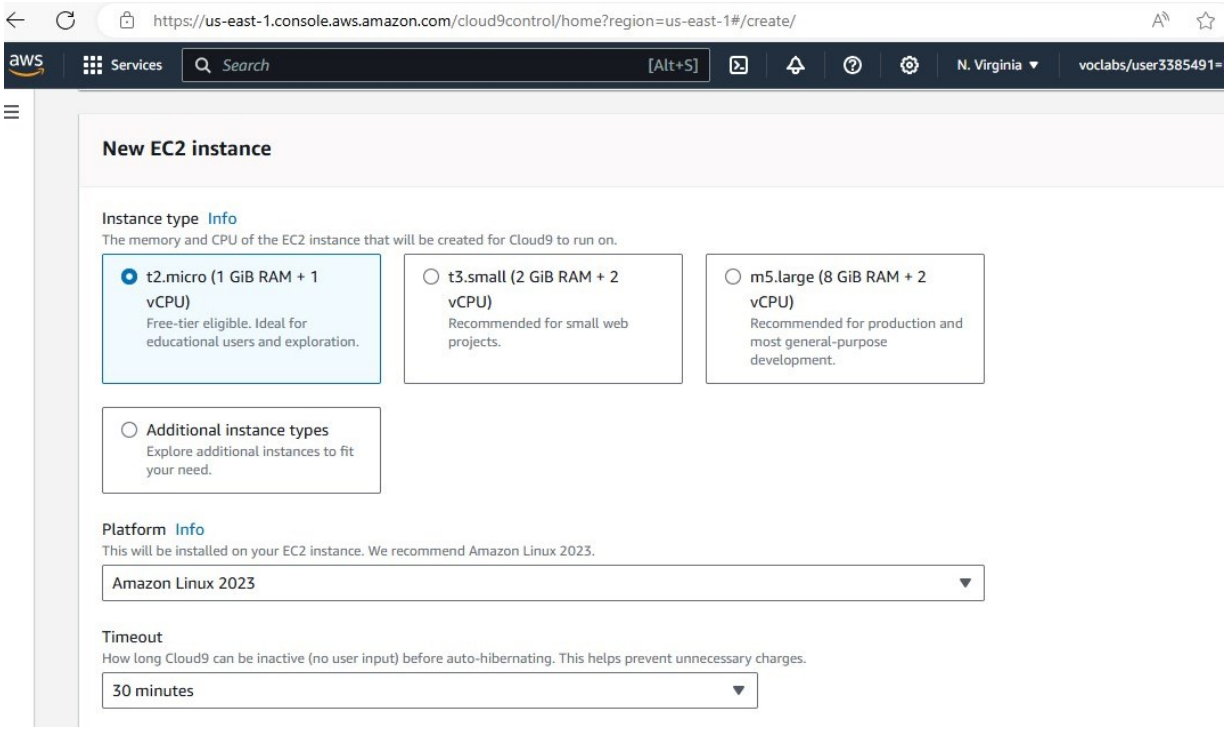


Hello I am Shreyash Kamat

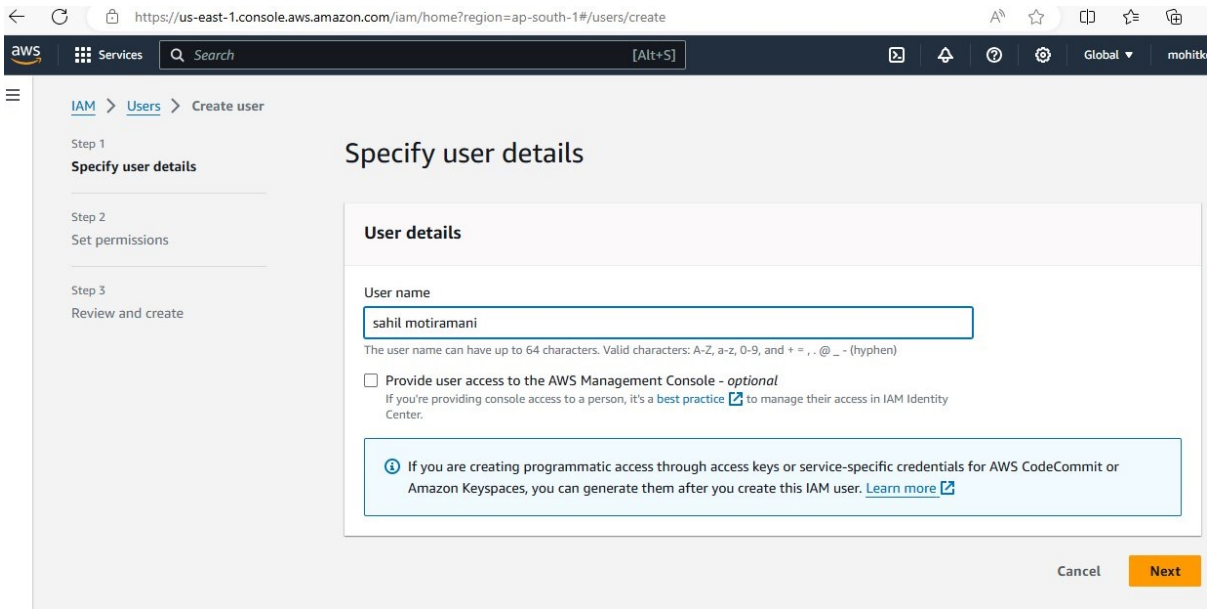
Exp 1b: Cloud9 Setup and Launch, Collaboration demonstration by creation of IAM groups and users.

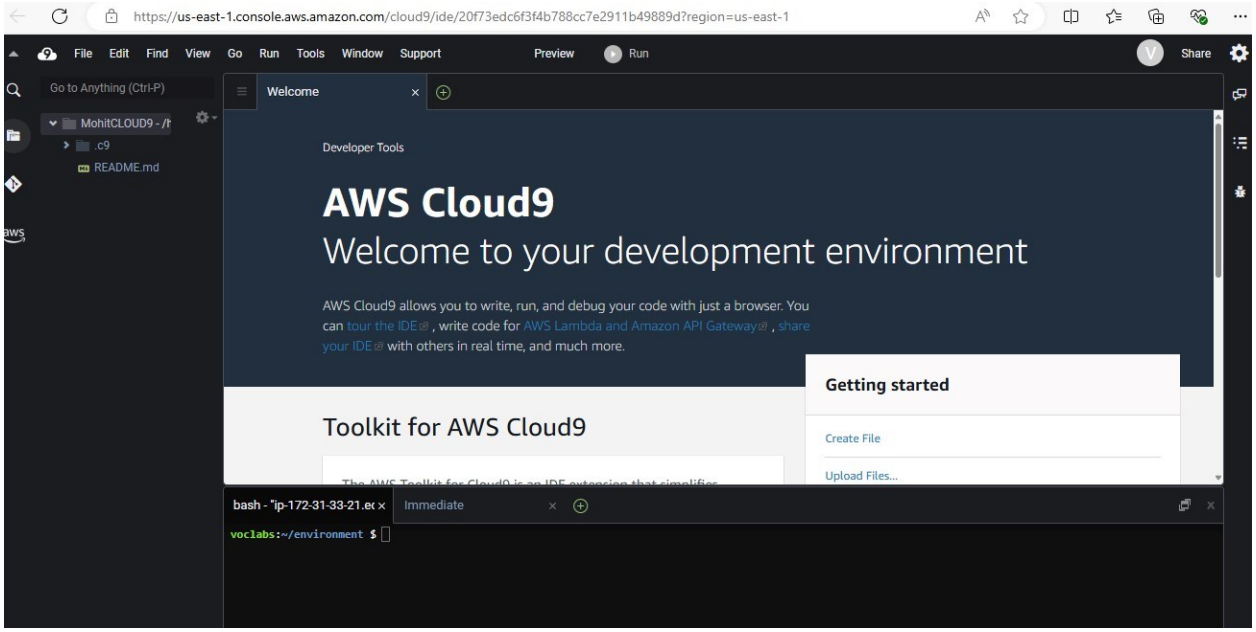
1. Open your AWS account and search for Cloud9 service inside Developer tools. Create a new Cloud9 environment by filling in the required details. Make sure you use an EC2 instance to create your environment.



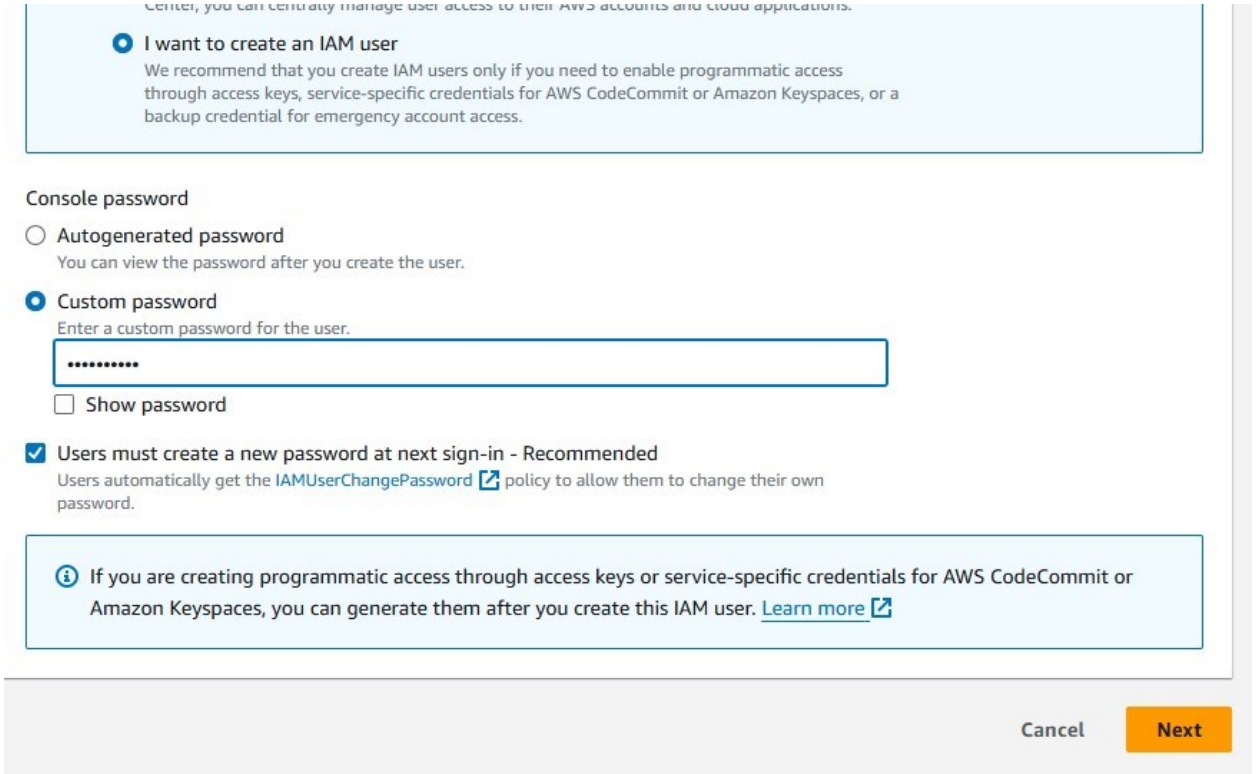


2. We have successfully setup and launched our Cloud9 environment. Over here, we can build and develop programs as per our desire. We are also allowed to collaborate with multiple other users and access shared resources.

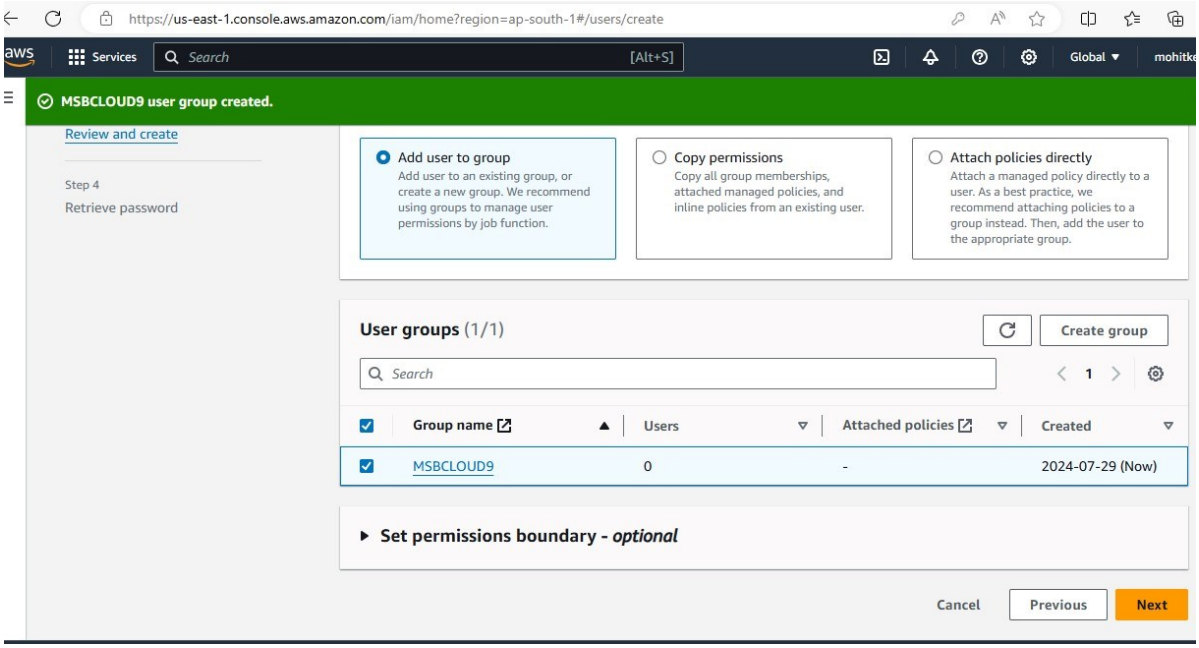




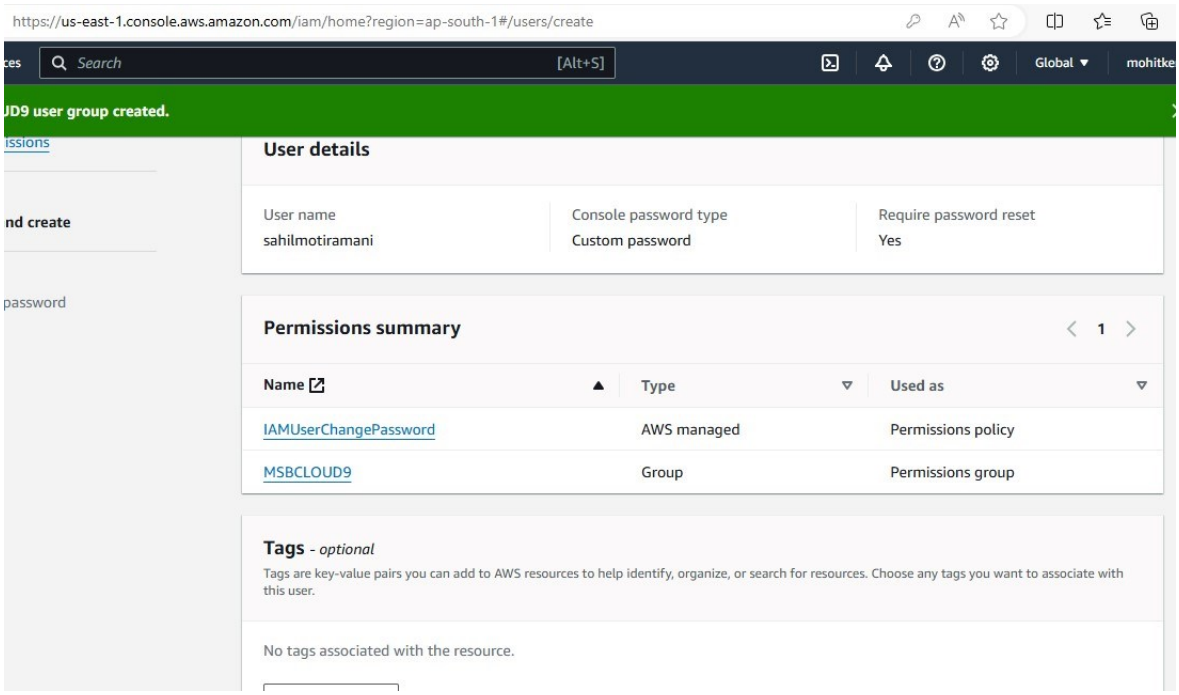
3. Moving on, we are supposed to create a new user. Give a suitable name to the user and decide the password for the same.



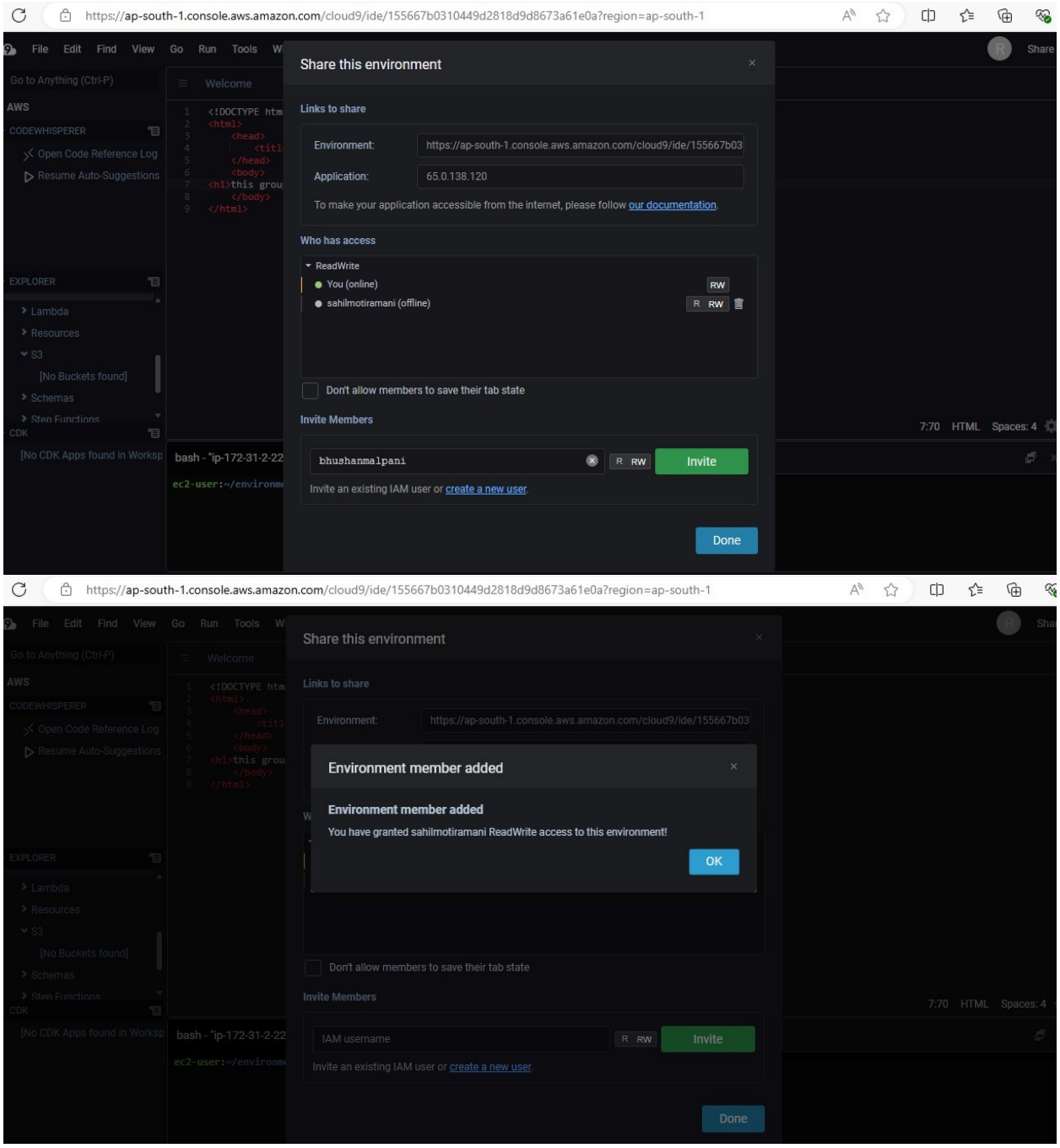
4. Similarly, create a new group and provide a suitable name for the same. Include the IAM users in this group together for our convenience i.e to provide similar kinds of permissions to the entire group rather than an individual user.



5. The user has successfully been created i.e There is a custom made username and a password for the IAM user.



6. Go back to the cloud9 environment. Click on share this environment option so as to allow other collaborators to access you environment. Include your newly made IAM user in this environment and enable Read/Write permissions for it



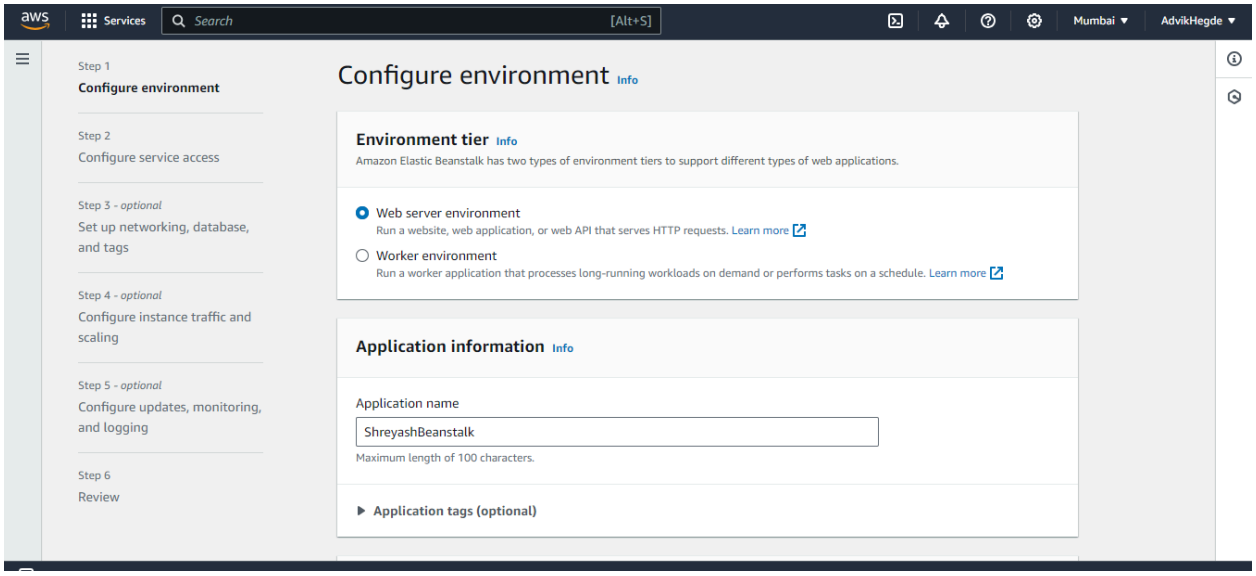
7. Further, we are supposed to login from another browser using the credentials of the IAM user, so as to access the shared cloud9 environment with us.
   These steps could not be completed because Cloud9 services have been disrupted and there is no access to the IAM user from the remote login

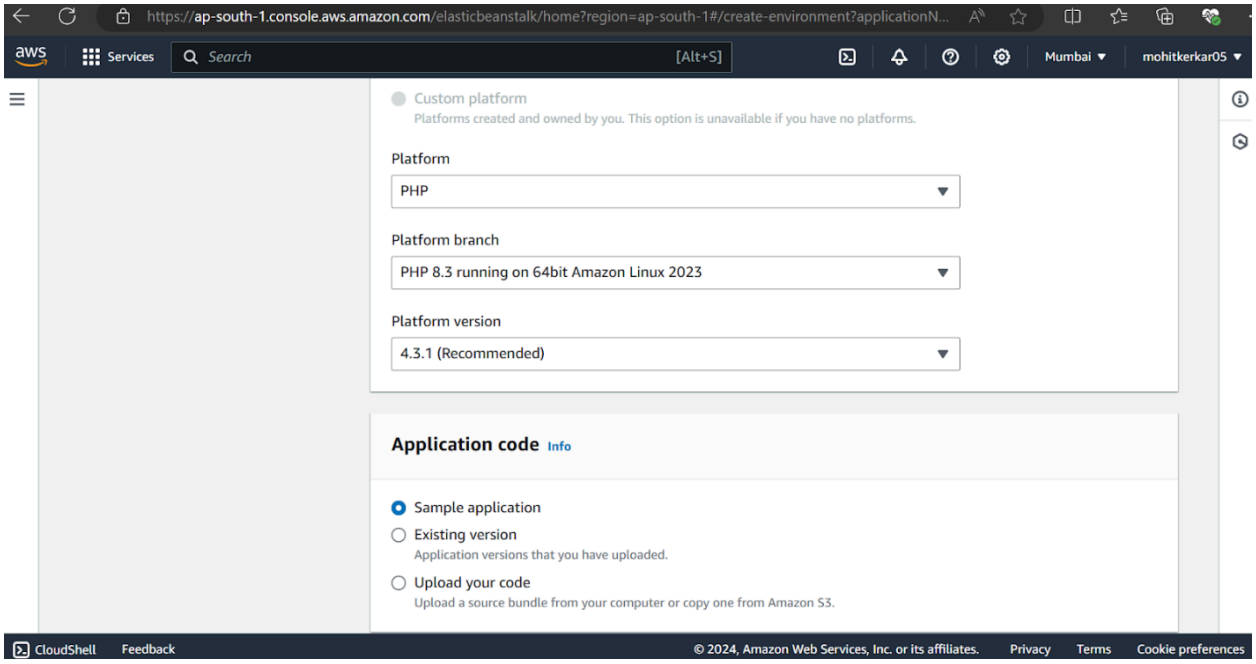
Name:Shreyash Kamat Div/Roll No. : D15C/22

Exp 02:To Build Your Application using AWS Code Build and Deploy on S3 / SEBS using AWS CodePipeline, deploy Sample Application on EC2 instance using AWS Code Deploy.

Step 1: Create our Elastic Beanstalk Environment

Login into your AWS account and navigate to services. Search for Elastic Beanstalk service and click on create application. Give your application a suitable name. For the platform, select PHP. The rest of the configuration settings are to be kept as default.

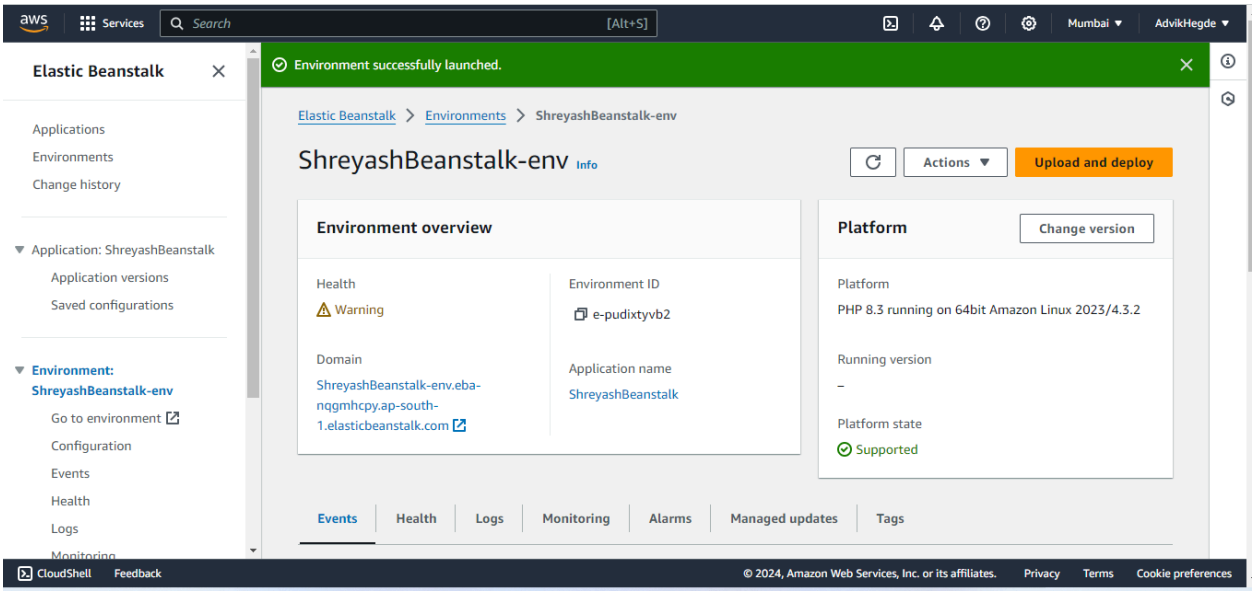




Now, while creating the environment, we are asked to provide an IAM role with the necessary EC2 permissions. We are supposed to make sure that we have made an existing IAM role with the following set of permissions:

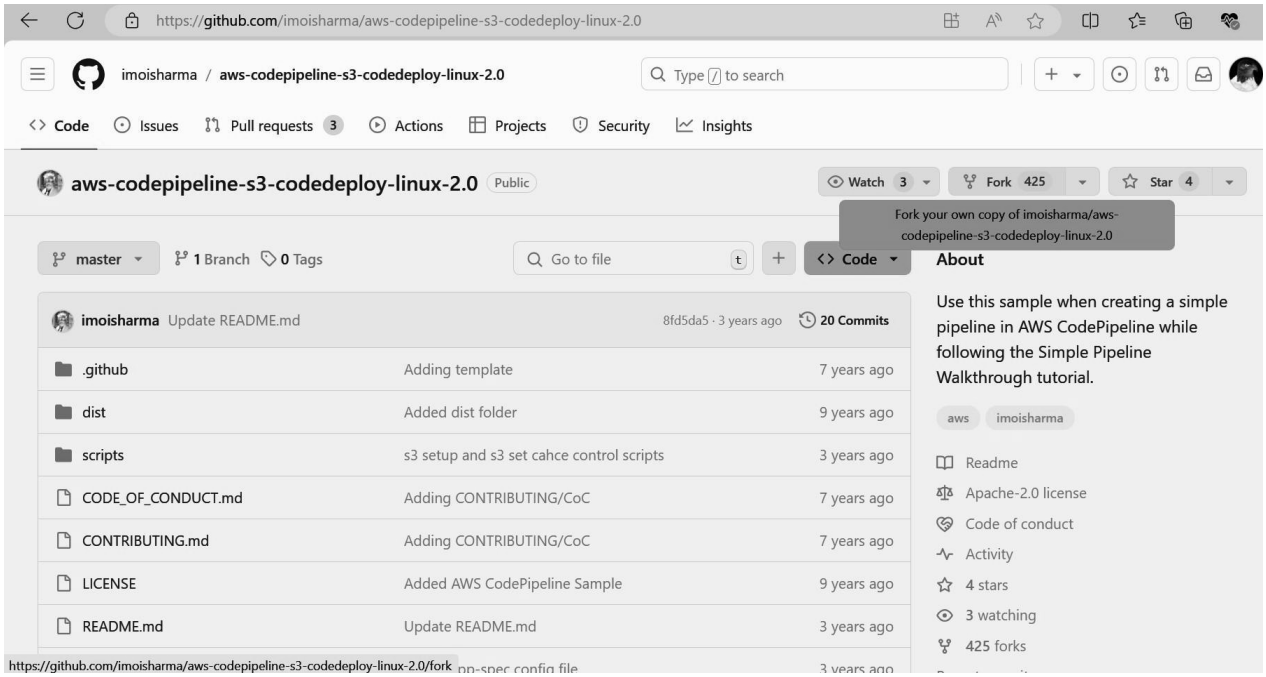
1. AWSElasticBeanStalkWebTier
2. AWSElasticBeanStalkWorkerTier
3. AWSElasticBeanStalkMulticontainerDocker

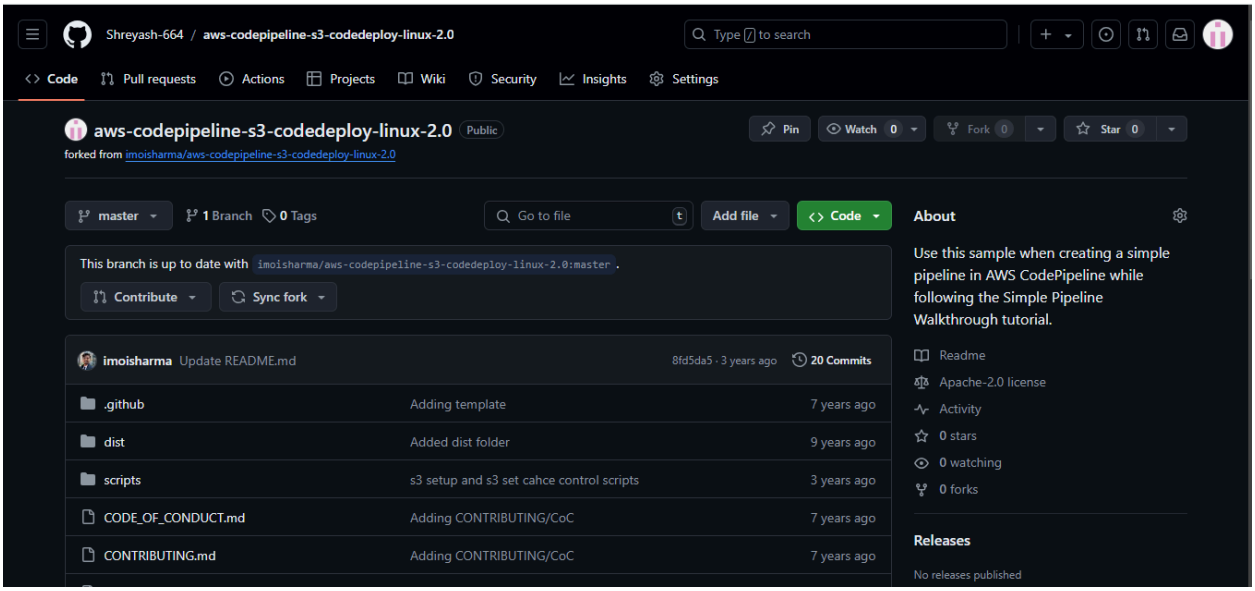
We can skip the steps to follow after the initial few steps mentioned above and move straight to review the settings of our environment. After reviewing everything properly, our environment can successfully be created.



Step 2: Fork the required repository onto our github account

The repository to be forked is - imoisharma/aws-codepipeline-s3-codedeploy-linux-2.0

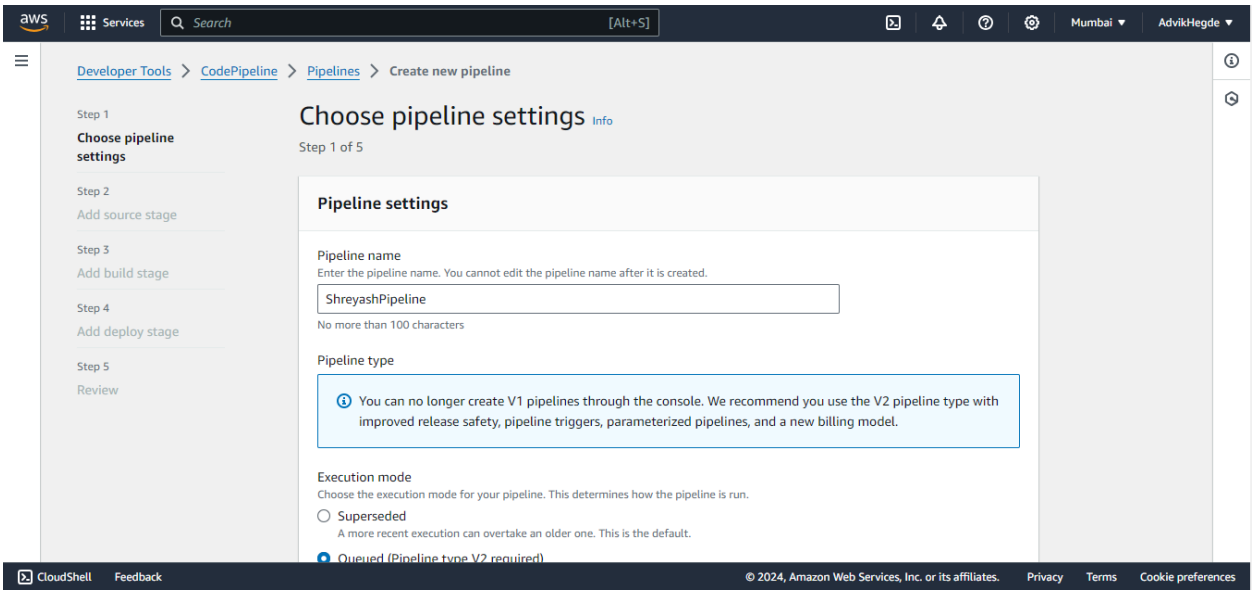




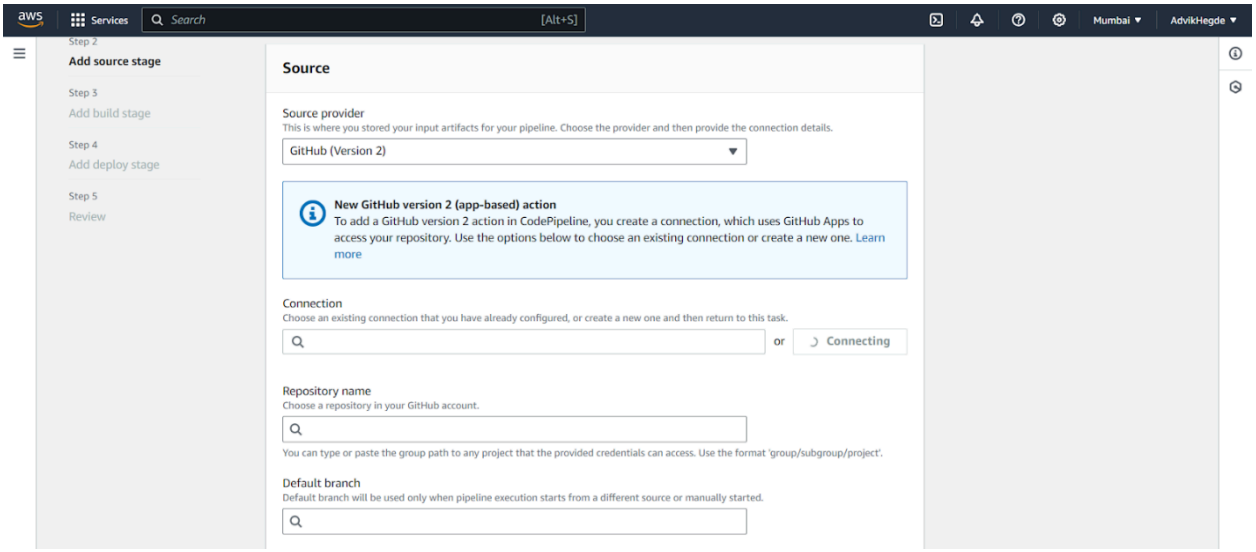
This step is necessary for the execution of the steps to follow. It will be helpful in the creation of a pipeline.

Step 3: Creation of the Pipeline

Navigate to Codepipeline inside Developer Tools. Give a suitable name to the pipeline you want to create.

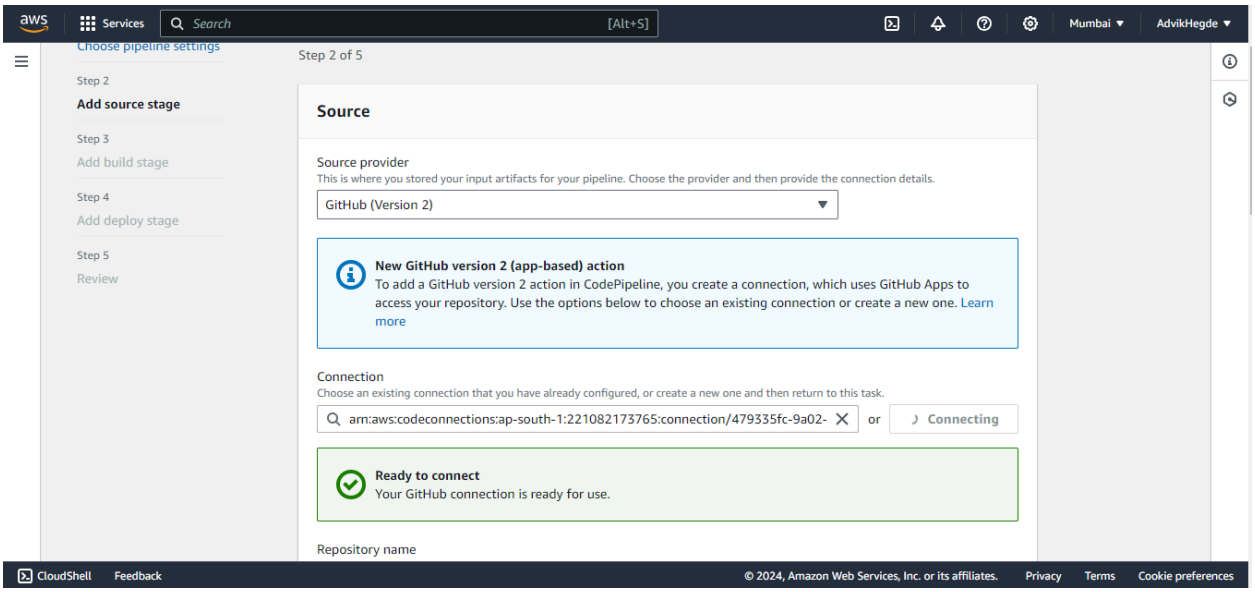


And click on next …

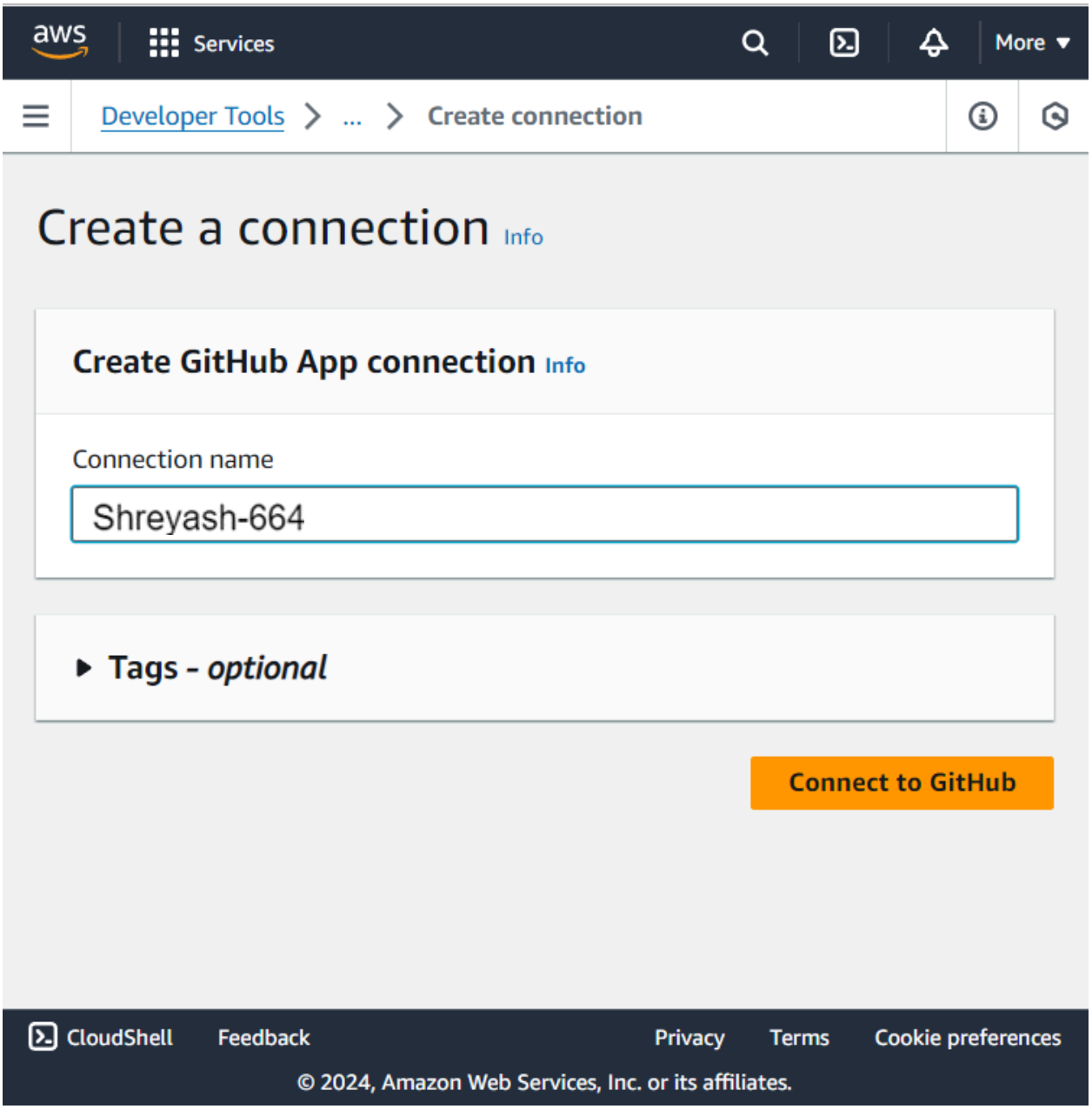


Step 4: GitHub connection

In this step, we are supposed to create a GitHub connection and add our existing repository over here i.e. the one we forked earlier.

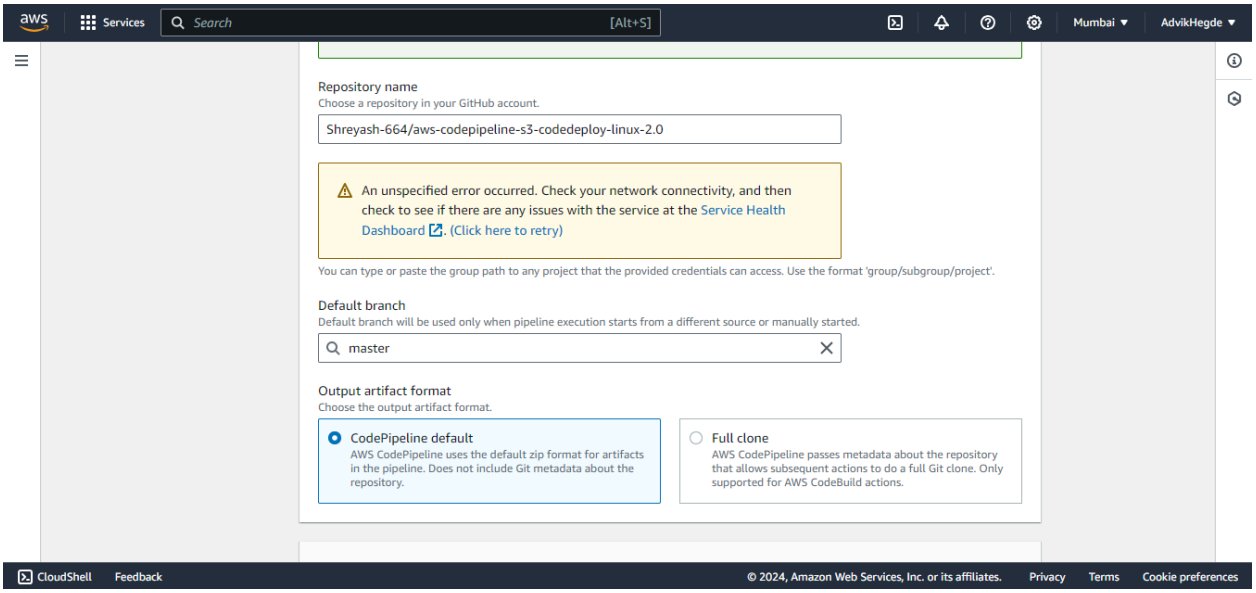


We are supposed to enter our GitHub username so as to proceed towards making the connection.


aws
Services
More
Developer Tools
...
Create connection
Create a connection Info
Create GitHub App connection Info
Connection name
Shreyash-664
Tags - optional
Connect to GitHub
CloudShell
Feedback
Privacy
Terms
Cookie preferences
© 2024, Amazon Web Services, Inc. or its affiliates.

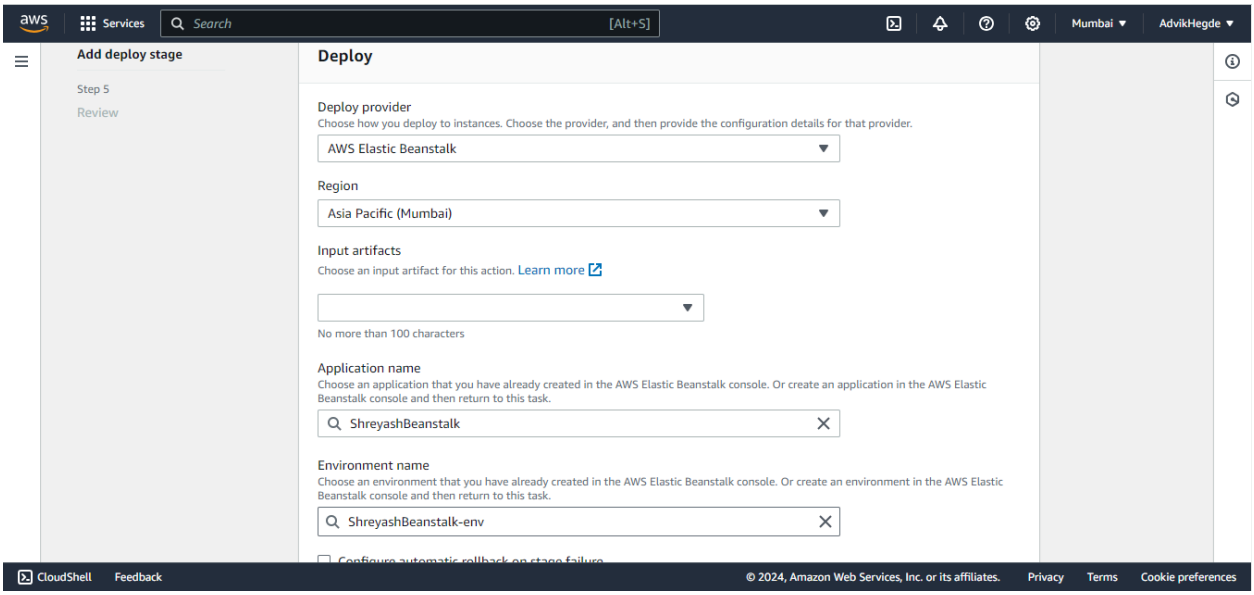
Now to finalize our connection, we are to install an application which connects AWS to our GitHub account and repository.

Post the establishment of the connection, this is the message that is displayed. We can further select the branch of our repository that we want to connect.

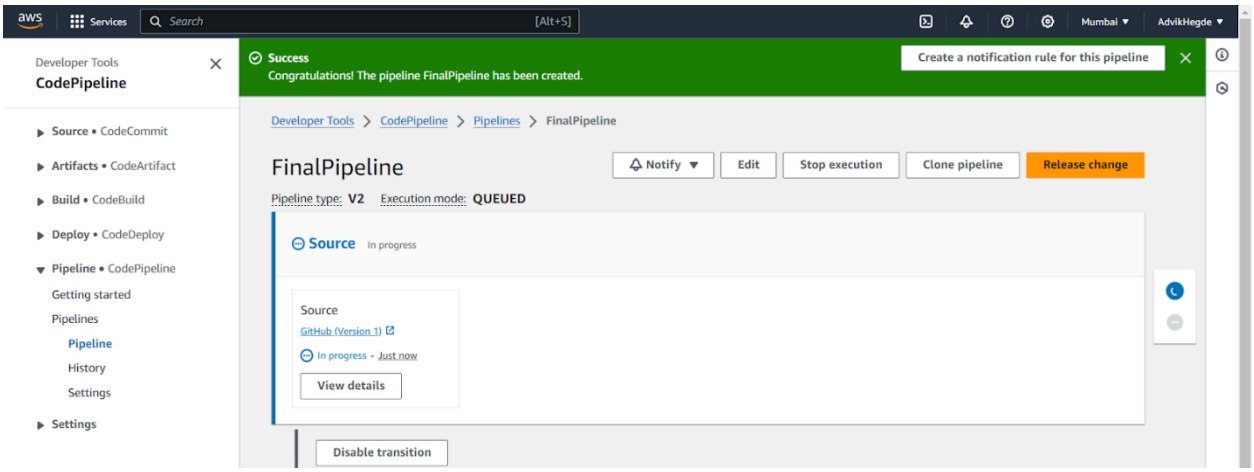


Step 5: Deployment stage:

We are expected to skip the build stage and move towards the deployment step. In the deployment step we are supposed to choose the Elastic Beanstalk application and the environment that we created earlier and proceed with our pipeline creation.

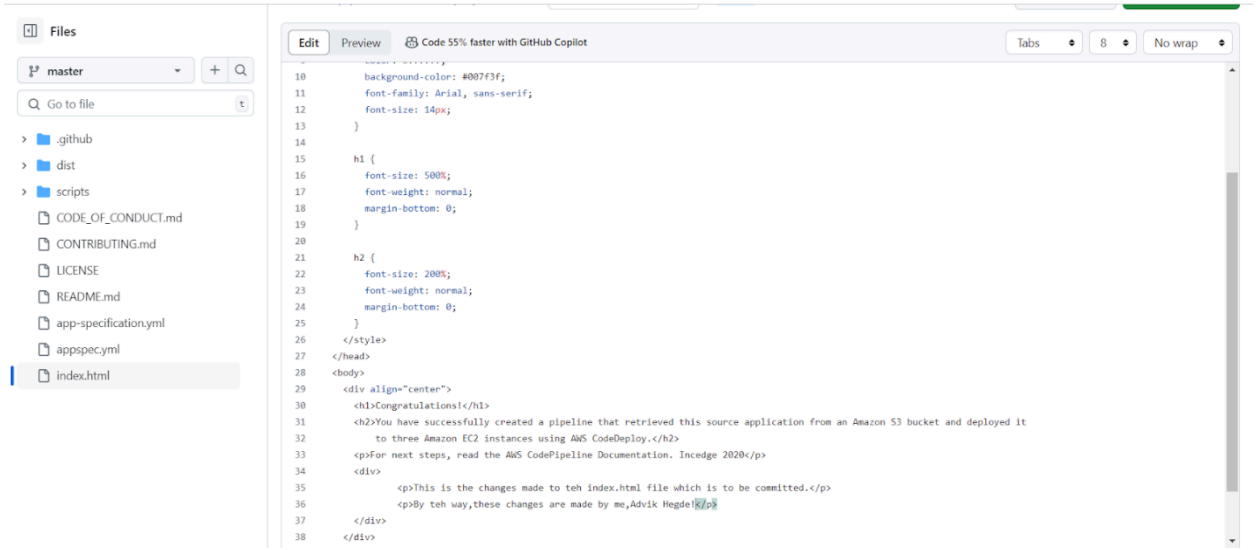


Step 6: Post deployment stage: When all the stages run successfully, this is what is displayed onto the screen. It shows us that our application and our environment have successfully been deployed using a dedicated pipeline created.



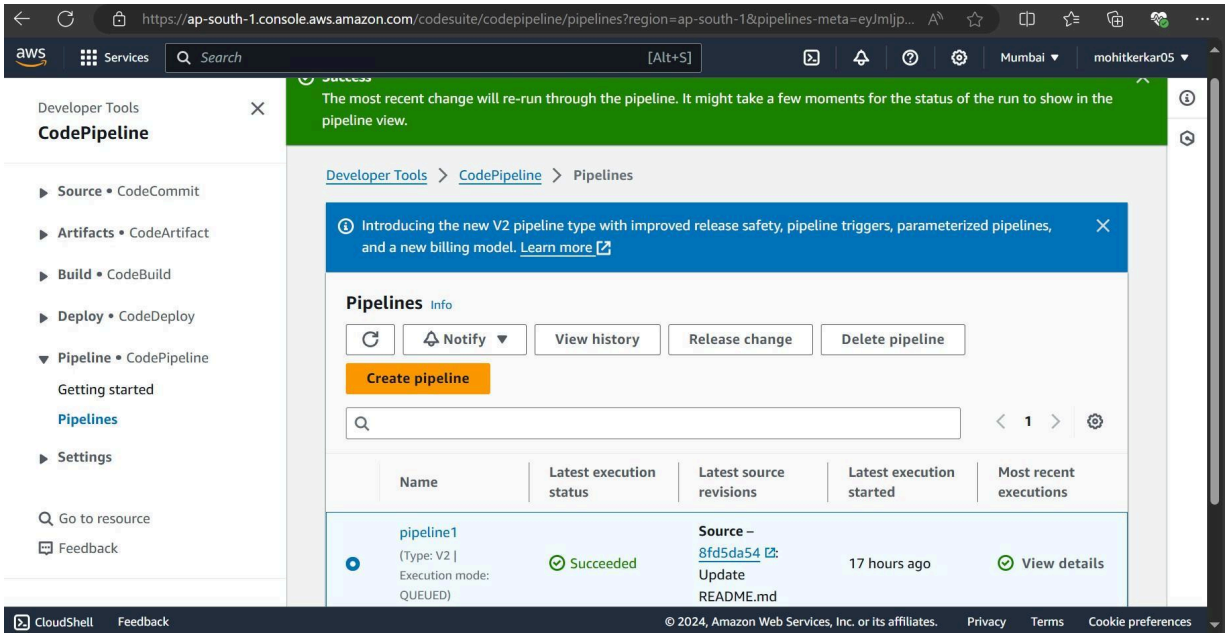
Step 7: Committing changes to your GitHub code

Now, we will go to our forked repository and make some changes to the index.html file. On making the desired changes, we are supposed to commit those changes on our forked repository. Write a good commit message so as to recognize it when it appears on the pipeline.

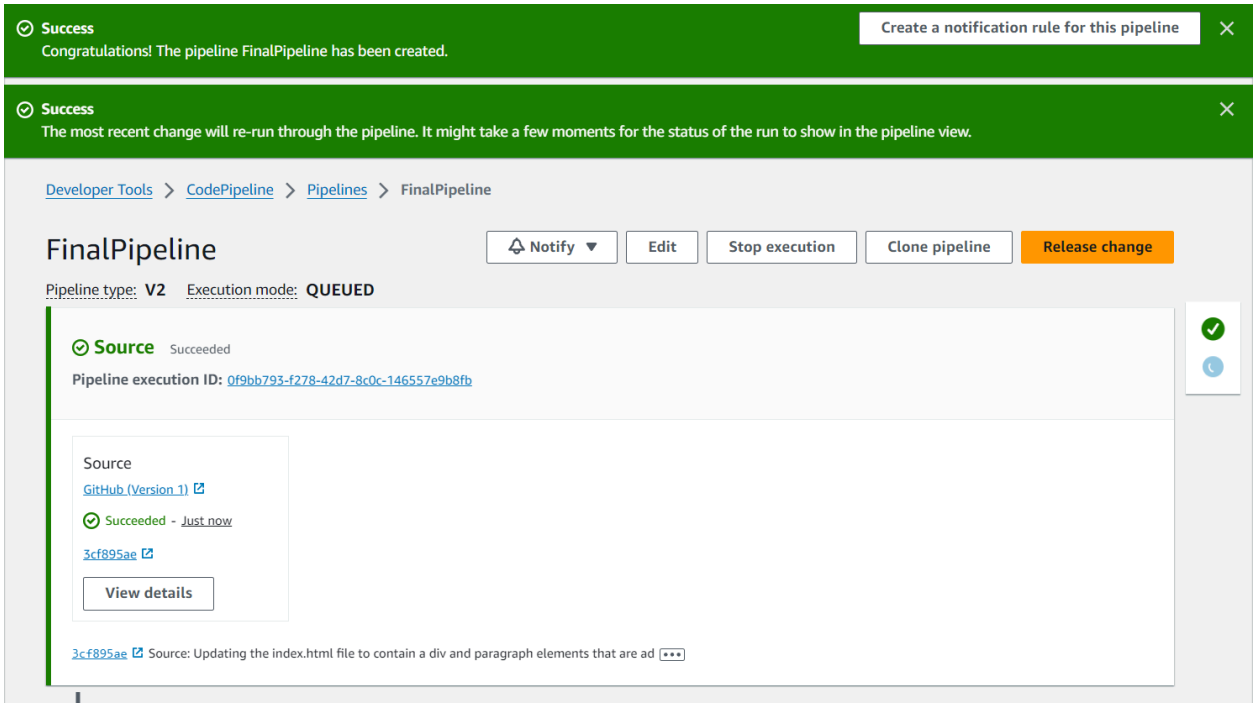


Step 8: Apply the newly made changes in index.html onto our pipeline

Come back to the Codepipeline section and select the pipeline through which we successfully created and deployed our application. Click on the release change option to apply the latest changes/commits from our GitHub repository to our pipeline.



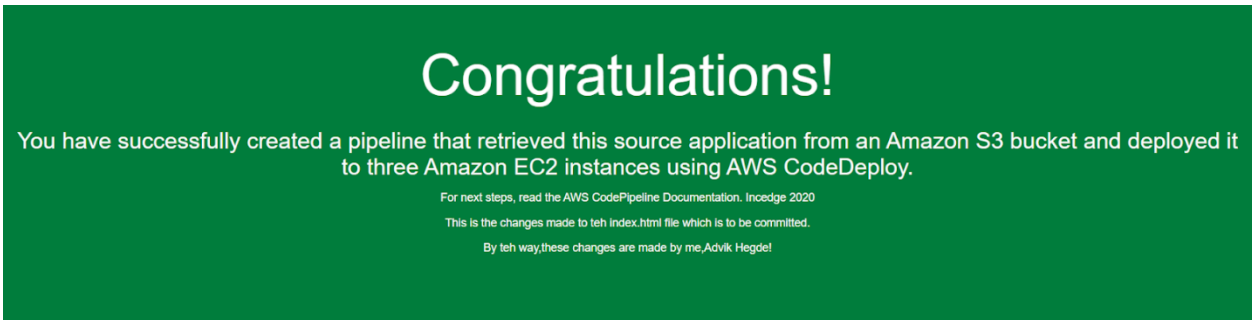
Once the changes have been applied, we see the commit message that we wrote for the latest commit on our repository being reflected on our pipeline. Over here, it would be seen somewhere near the bottom of the image that is attached. “Update index.html” was the latest commit message in the GitHub repository.



Step 9: Open the Domain of our Elastic Beanstalk environment

Now, we navigate back to our Elastic Beanstalk environment and open the environment domain of our deployed application.

The text in this image is clearly distinguishable from the earlier website’s text meaning that the changes that we made to our code in index.html has successfully been applied to the website that we deployed.



Name: Shreyash Kamat Div: D15C Roll No: 22

Adv DevOps Lab Exp 03

Aim: To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud

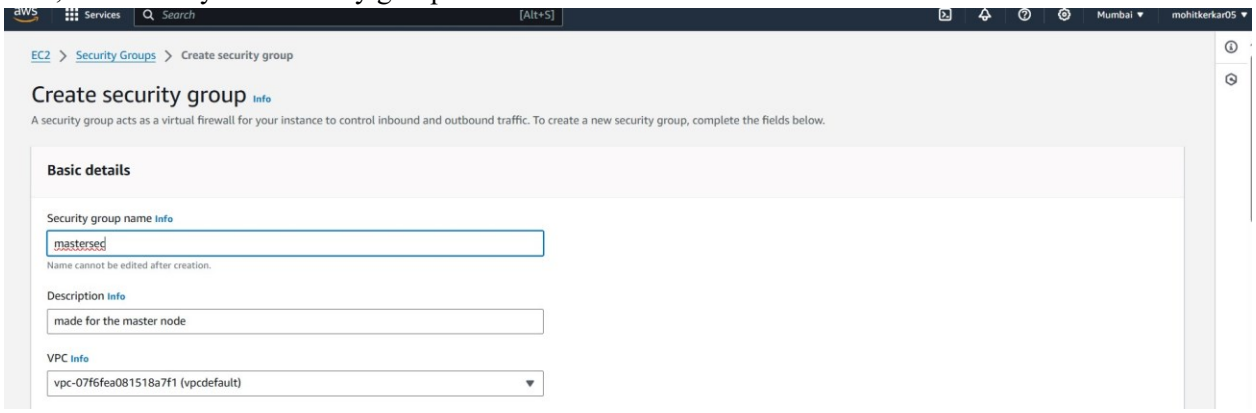
(I have performed this experiment on my personal AWS account)

Step 1: Create Key-pair, Security groups and required default VPCs

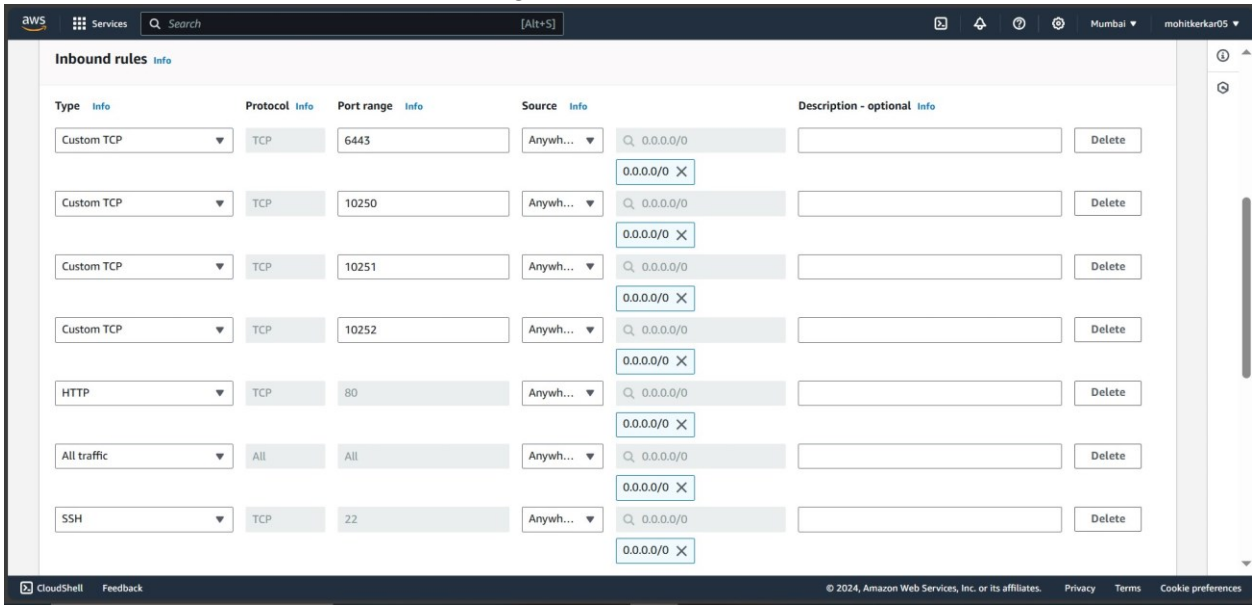
I started off by creating two separate security groups i.e one for the master instance (kubeadm to be initialized within it) and the other for the 2 worker instances. For creating any EC2 security groups, we require a VPC (Virtual private cloud) and a subnet along with it .. so that we can work with or allow inbound and outbound traffic and communication. Make sure you have a default VPC and a subnet already created which can be used for this experiment.

Also, make sure that you have a key pair installed of the type RSA with .pem extension on your local machine. Save it at a place which is accessible and where you can work from your terminal.

Here, i created my first security group

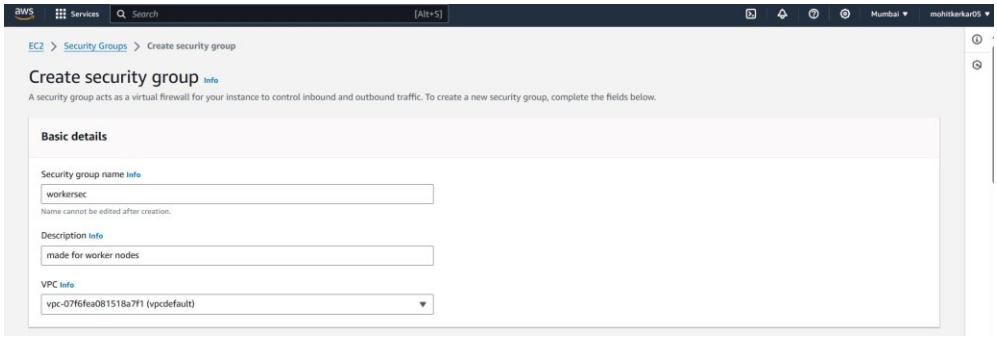


I modified the inbound rules in the following fashion for the master node

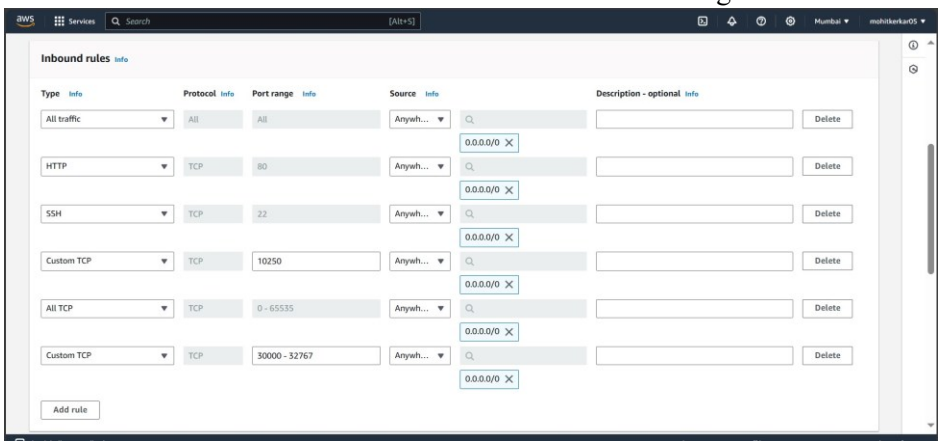


Then, i created a security group for the worker nodes

Name: Shreyash Kamat Div: D15C Roll No: 22



Edited the inbound rules for the worker nodes in the following fashion

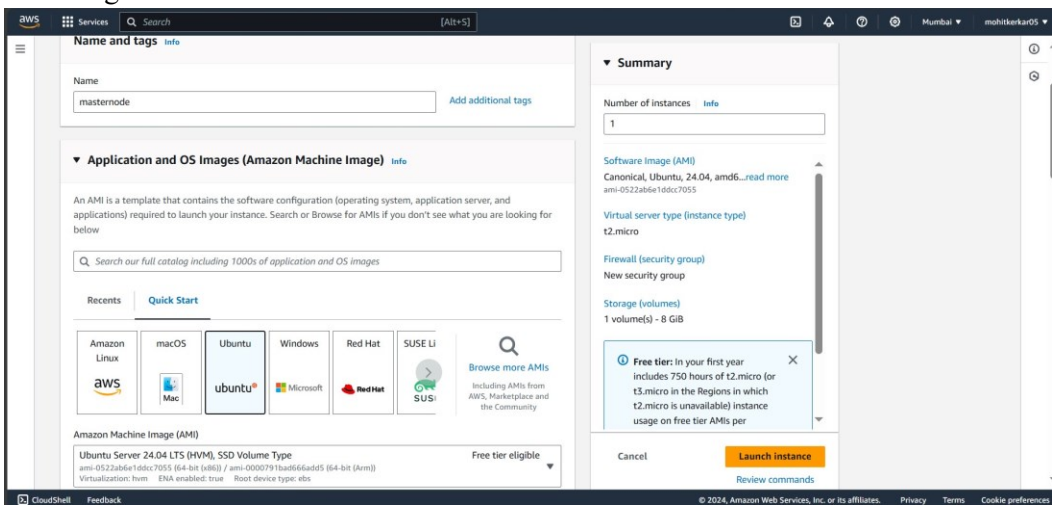


Step 2: Create 3 EC2 instances i.e one master and other 2 workers

Now, in order to work with the instances, navigate to EC2 instances section and launch a few instances. Launch one instance and name it as masternode and the other two as worker.

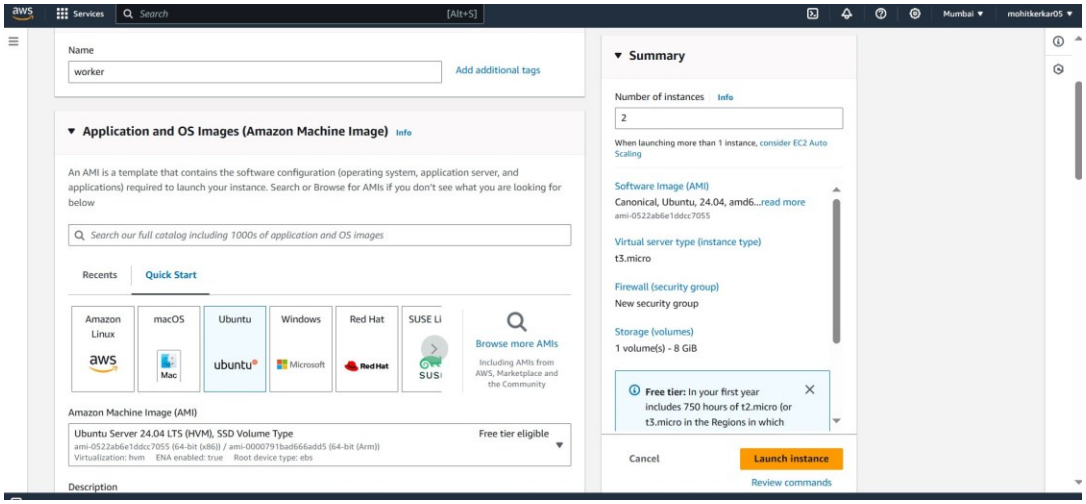
I selected **AMI as ubuntu** among the list of OS that we shown available in the free tier eligibility list

Following is the master instance node:

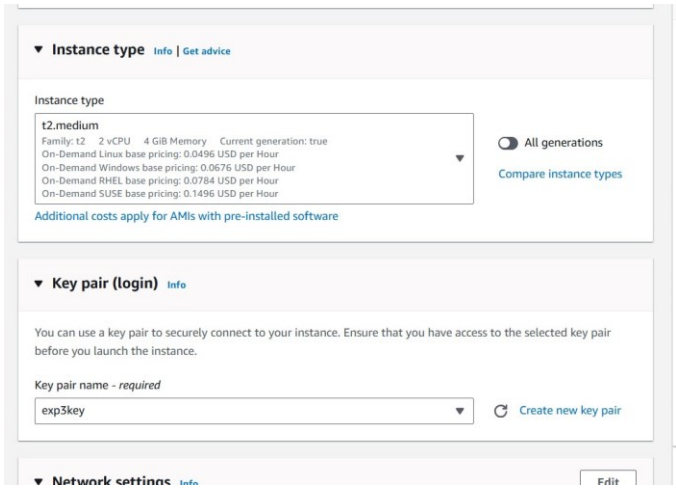


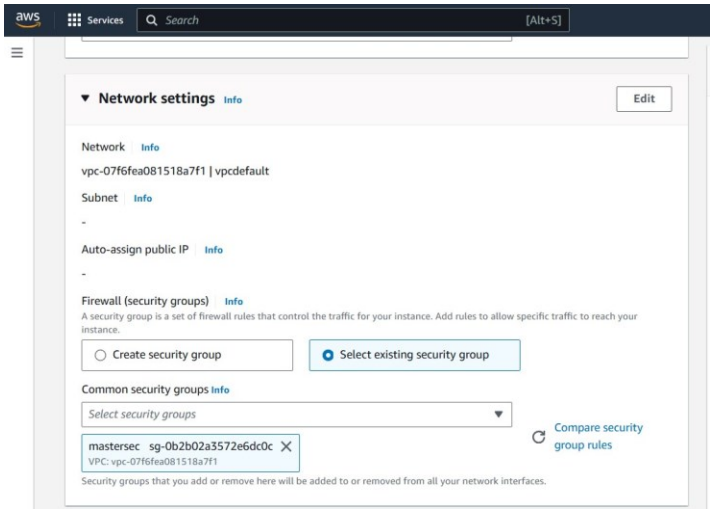
Following is the worker instance node:

Name: Shreyash Kamat Div: D15C Roll No: 22



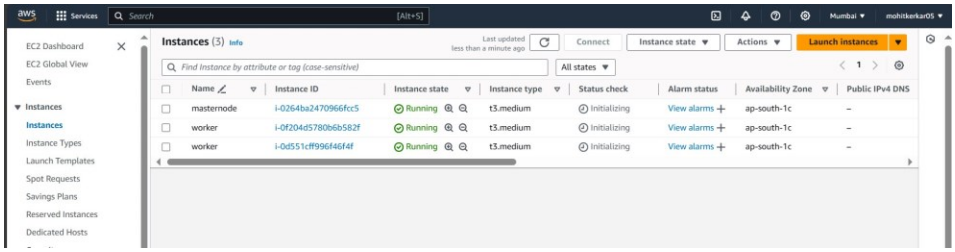
IMPORTANT: Select t2.medium as the instance type for all 3 instances, as kubernetes requires at least 2 CPUs to work with and sufficient amount of other resources as well



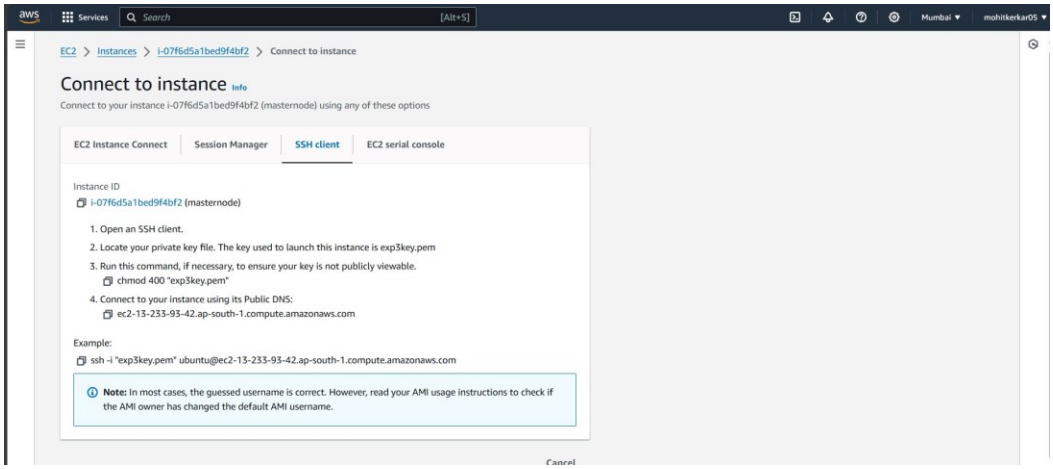


After launching all the 3 instances, select one instance and press ‘Connect’

Name: Shreyash Kamat Div: D15C Roll No: 22



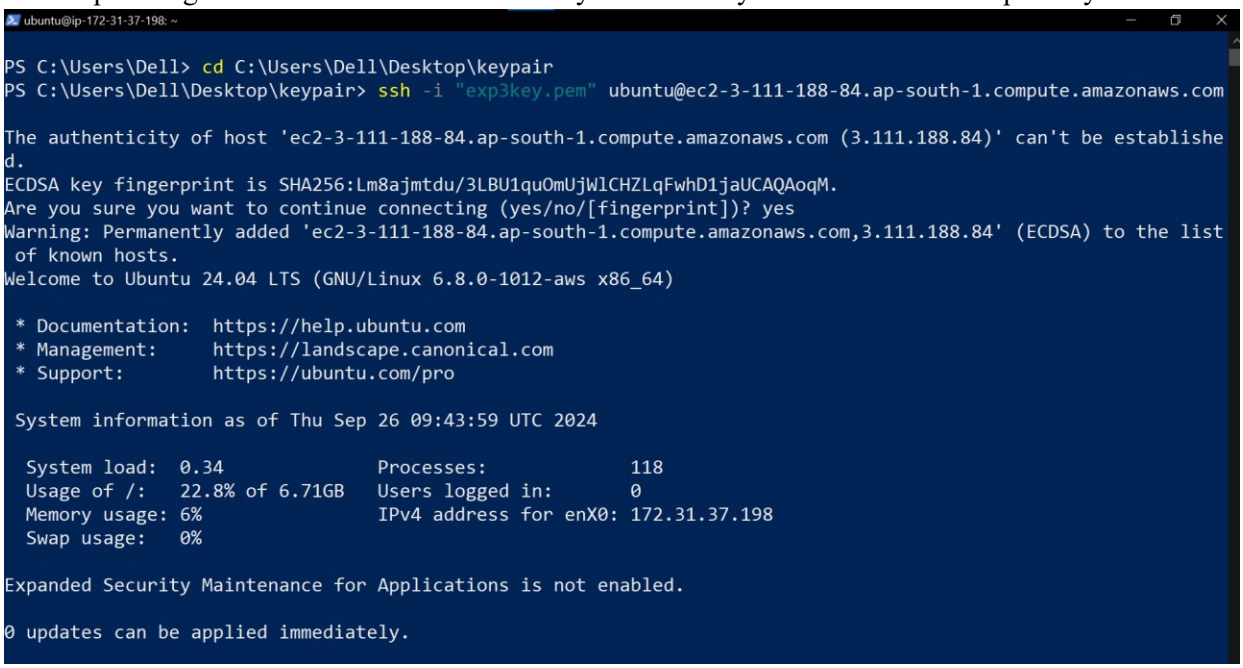
Navigate to SSH client section and copy the command thats listed for connecting to the node remotely
ssh -i "exp3key.pem" ubuntu@ec2-13-233-93-42.ap-south-1.compute.amazonaws.com…for the master node
Similarly, do this for all nodes

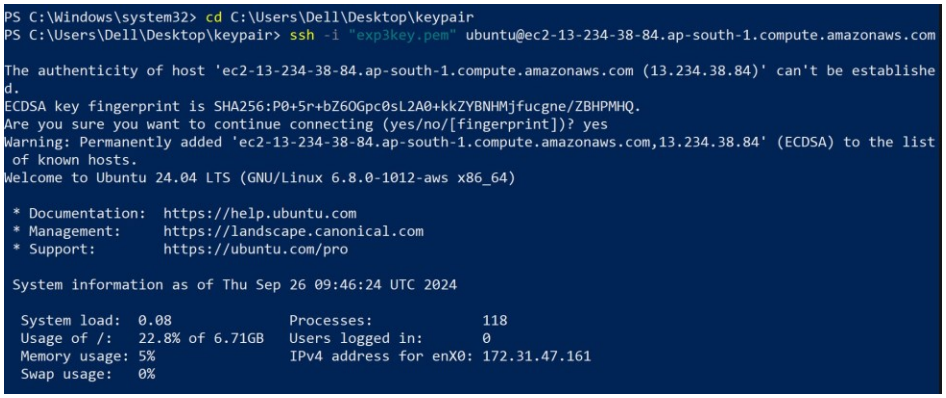


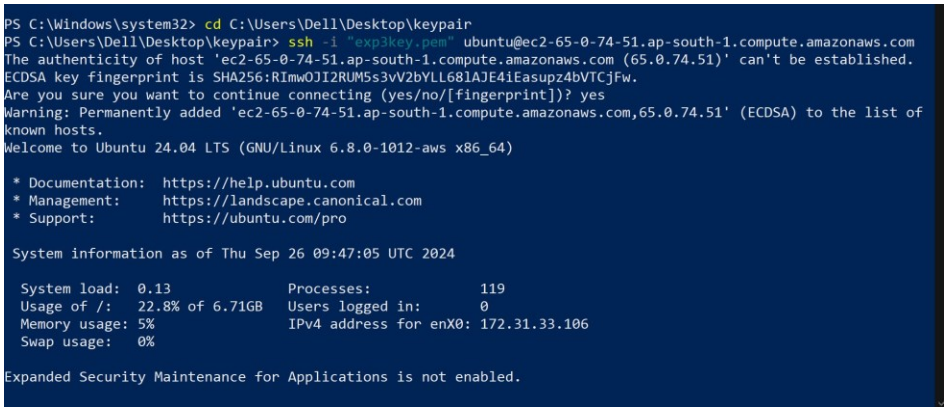
Now, open 3 separate terminals and run change directories to the folder which contains the key we created earlier

Run their respective SSH commands (one in each terminal)

This helps us log onto those instances individually and remotely and work on them separately





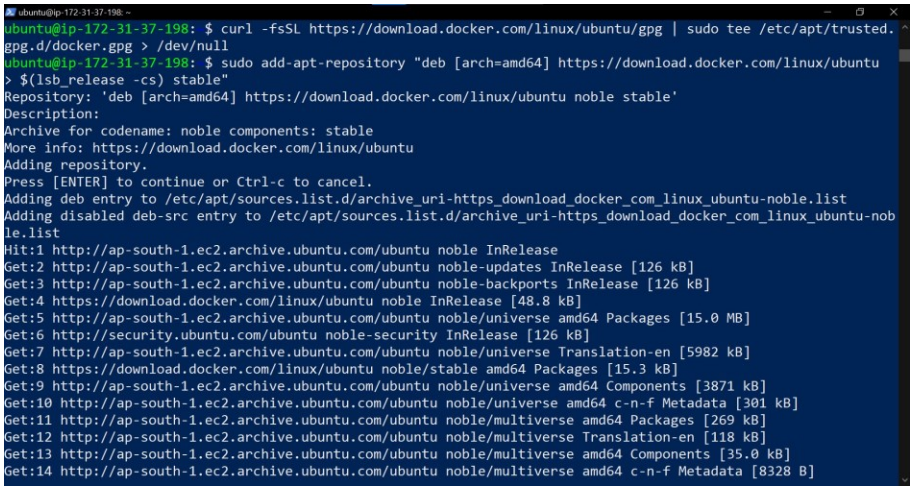


Step 3: Docker installation:

Run the following command: These commands are used to install Docker on an Ubuntu system by adding Docker's official GPG key and configuring the Docker repository.

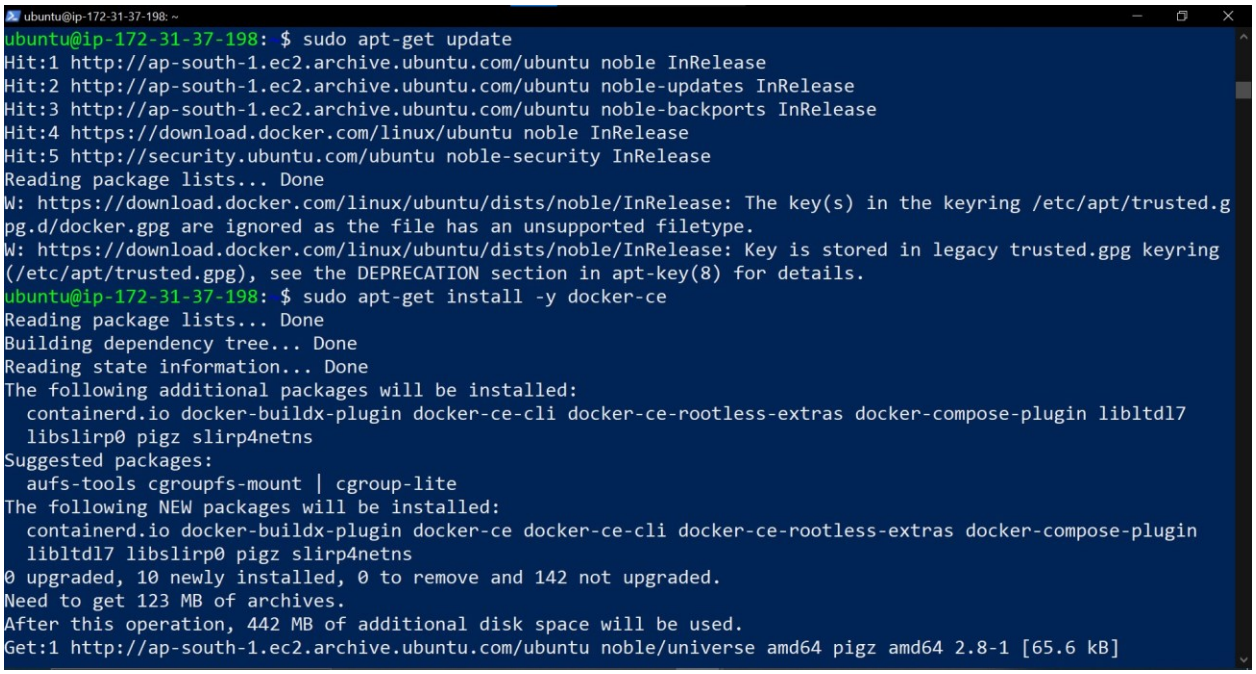
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/trusted.gpg.d/docker.gpg > /dev/null sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"

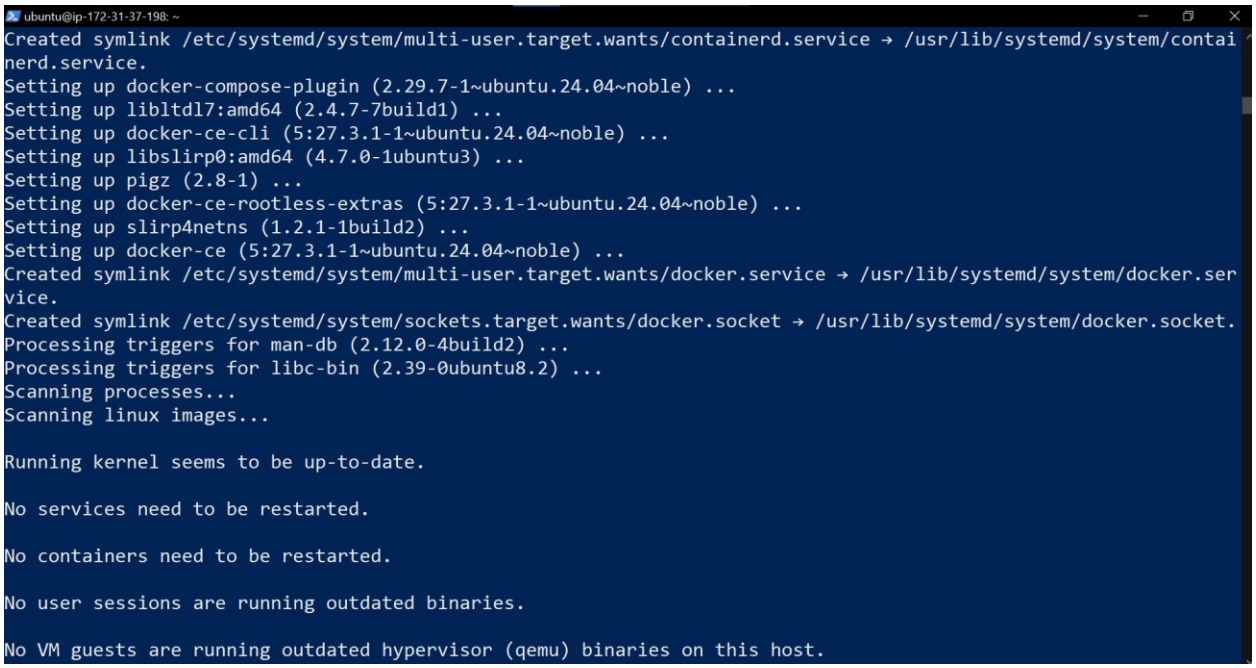
Name: Shreyash Kamat Div: D15C Roll No: 22



Run the following commands to refresh your local package list to ensure the latest packages are available and install Docker Community Edition on your system without prompting for confirmation.

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





Run the following commands to create the Docker configuration directory and write a configuration file for Docker to use the systemd cgroup driver.

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

```
ubuntu@ip-172-31-37-198: ~
ubuntu@ip-172-31-37-198:~$ sudo mkdir -p /etc/docker
ubuntu@ip-172-31-37-198:~$ cat <<EOF | sudo tee /etc/docker/daemon.json
> {
> "exec-opts": ["native.cgroupdriver=systemd"]
> }
> EOF
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
```

Run the following commands to ensure Docker starts automatically on system boot, reload systemd to apply new configurations, restart Docker to apply the new configuration.

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

```
ubuntu@ip-172-31-37-198:~$ 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
ubuntu@ip-172-31-37-198:~$ sudo systemctl daemon-reload
ubuntu@ip-172-31-37-198:~$ sudo systemctl restart docker
```

Step 4: Kubernetes Installation:

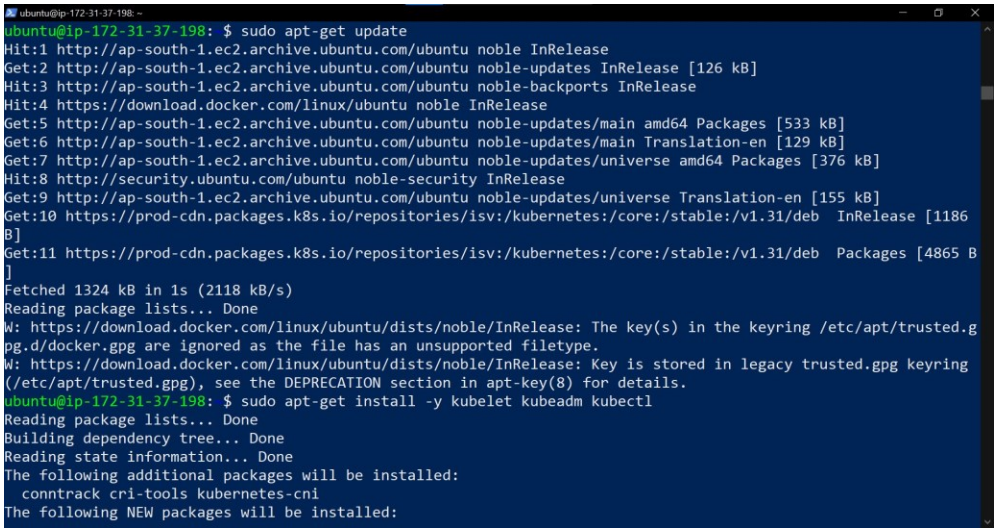
Run the following commands to add the Kubernetes package repository to your Ubuntu system in order to install Kubernetes components like kubectl, kubelet, and kubeadm.

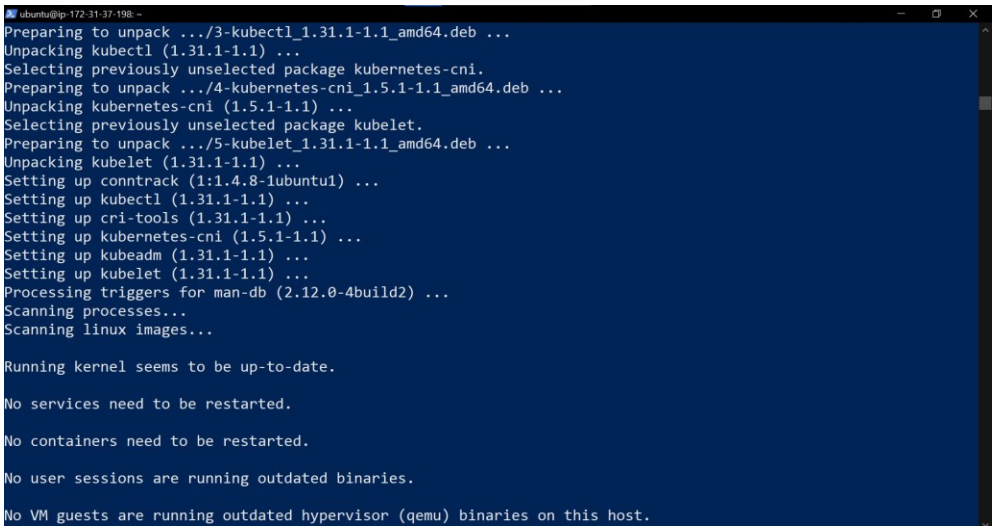
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

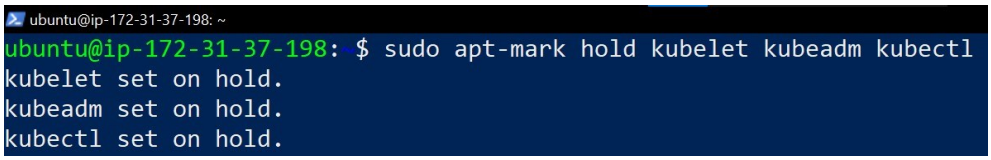
```
ubuntu@ip-172-31-37-198: ~
ubuntu@ip-172-31-37-198:~$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dear
mor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
ubuntu@ip-172-31-37-198:~$ echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.
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/ /
```

Next, run these commands to refresh the local package index to include the newly added Kubernetes repository, install the main Kubernetes components (kubelet, kubeadm, kubectl), prevent the installed Kubernetes components from being automatically updated, ensuring cluster stability until manual updates are performed.

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

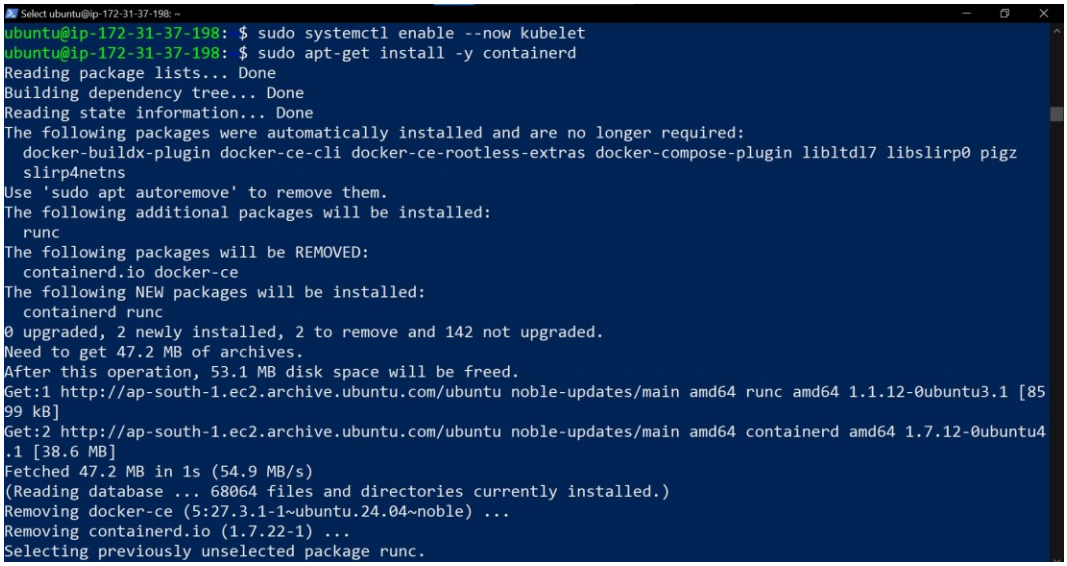




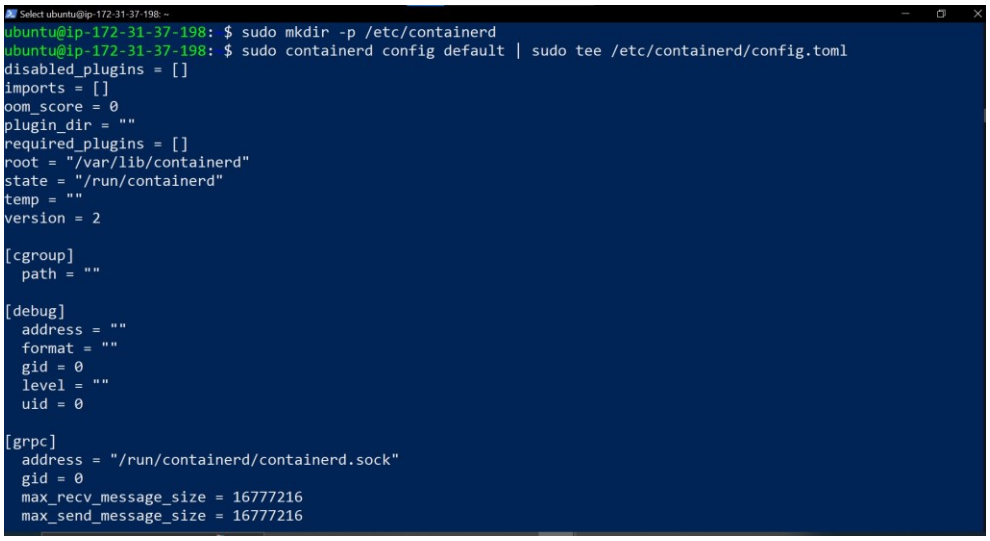


The command **sudo systemctl enable --now kubelet** enables the **kubelet** service, ensuring it starts automatically at boot and immediately starts running without needing a system reboot. The **sudo apt-get install -y containerd** command installs **containerd**, a container runtime that Kubernetes uses to manage and run containers.

sudo systemctl enable --now kubelet
sudo apt-get install -y containerd



The command `sudo mkdir -p /etc/containerd` creates the directory for containerd configuration files if it doesn't already exist. The next command, `sudo containerd config default | sudo tee /etc/containerd/config.toml`, generates a default configuration for containerd and saves it as `config.toml` in that directory. Together, these commands set up the necessary directory and create a default configuration file for containerd. sudo mkdir -p /etc/containerd sudo containerd config default | sudo tee /etc/containerd/config.toml



The command `sudo systemctl restart containerd` restarts the `containerd` service to apply any changes. `sudo systemctl enable containerd` sets it to start automatically at boot, while `sudo systemctl status containerd` checks its current status, showing whether it's active and any errors. Together, these commands manage the operation and health of the container runtime.

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

```
ubuntu@ip-172-31-37-198: $ sudo systemctl restart containerd
ubuntu@ip-172-31-37-198: $ sudo systemctl enable containerd
ubuntu@ip-172-31-37-198: $ sudo systemctl status containerd
● containerd.service - containerd container runtime
     Loaded: loaded (/usr/lib/systemd/system/containerd.service; enabled; preset: enabled)
     Active: active (running) since Thu 2024-09-26 10:09:37 UTC; 14s ago
       Docs: https://containerd.io
   Main PID: 4771 (containerd)
      Tasks: 7
     Memory: 13.0M (peak: 13.4M)
        CPU: 123ms
     CGroup: /system.slice/containerd.service
             └─4771 /usr/bin/containerd

Sep 26 10:09:37 ip-172-31-37-198 containerd[4771]: time="2024-09-26T10:09:37.127045263Z" level=info msg="Start
Sep 26 10:09:37 ip-172-31-37-198 containerd[4771]: time="2024-09-26T10:09:37.127119958Z" level=info msg="Start
Sep 26 10:09:37 ip-172-31-37-198 containerd[4771]: time="2024-09-26T10:09:37.127202179Z" level=info msg="Start
Sep 26 10:09:37 ip-172-31-37-198 containerd[4771]: time="2024-09-26T10:09:37.127214694Z" level=info msg="Start
Sep 26 10:09:37 ip-172-31-37-198 containerd[4771]: time="2024-09-26T10:09:37.127239102Z" level=info msg="Start
Sep 26 10:09:37 ip-172-31-37-198 containerd[4771]: time="2024-09-26T10:09:37.127260747Z" level=info msg="Start
Sep 26 10:09:37 ip-172-31-37-198 containerd[4771]: time="2024-09-26T10:09:37.127807399Z" level=info msg=servin
Sep 26 10:09:37 ip-172-31-37-198 containerd[4771]: time="2024-09-26T10:09:37.127910878Z" level=info msg=servin
Sep 26 10:09:37 ip-172-31-37-198 containerd[4771]: time="2024-09-26T10:09:37.128176100Z" level=info msg="conta
Sep 26 10:09:37 ip-172-31-37-198 systemd[1]: Started containerd.service - containerd container runtime.
```

Socat installation:

sudo apt-get install -y socat

```
Select ubuntu@ip-172-31-37-198: ~
ubuntu@ip-172-31-37-198: $ 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 longer required:
  docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0 pigz
  slirp4netns
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  socat
0 upgraded, 1 newly installed, 0 to remove and 142 not upgraded.
Need to get 374 kB of archives.
After this operation, 1649 kB of additional disk space will be used.
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 socat amd64 1.8.0.0-4build3 [374 kB]
Fetched 374 kB in 0s (16.7 MB/s)
Selecting previously unselected package socat.
(Reading database ... 68108 files and directories currently installed.)
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.

No services need to be restarted.
```

Step 5: Kubernetes Cluster

Run this command only on the master instance sudo

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

It initializes a Kubernetes cluster with kubeadm and sets up the control plane (master node).

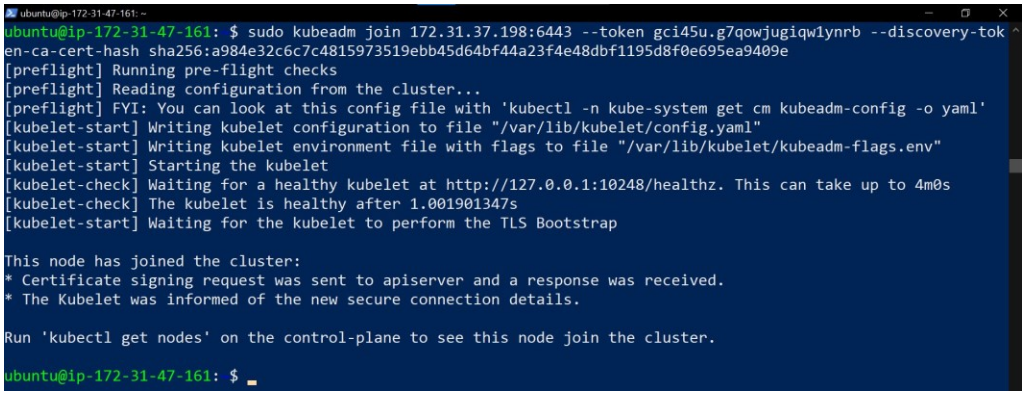
```
Select ubuntu@ip-172-31-37-198: ~
ubuntu@ip-172-31-37-198:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action beforehand using 'kubeadm config images pull'
W0926 10:13:04.272663    5083 checks.go:846] detected that the sandbox image "registry.k8s.io/pause:3.8" of the
 container runtime is inconsistent with that used by kubeadm.It is recommended 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-172-31-37-198 kubernetes kubernetes.default kubernet
es.default.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 172.31.37.198]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [ip-172-31-37-198 localhost] and IPs [172.31.37.198 12
7.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [ip-172-31-37-198 localhost] and IPs [172.31.37.198 127.
0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
```

```
Select ubuntu@ip-172-31-37-198: ~
luster
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotatable kubelet client certificate a
nd key
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!

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

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

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

  export KUBECONFIG=/etc/kubernetes/admin.conf

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

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

kubeadm join 172.31.37.198:6443 --token gci45u.g7qowjugiqw1ynrb \
        --discovery-token-ca-cert-hash sha256:a984e32c6c7c4815973519ebb45d64bf44a23f4e48dbf1195d8f0e695ea9409e
```

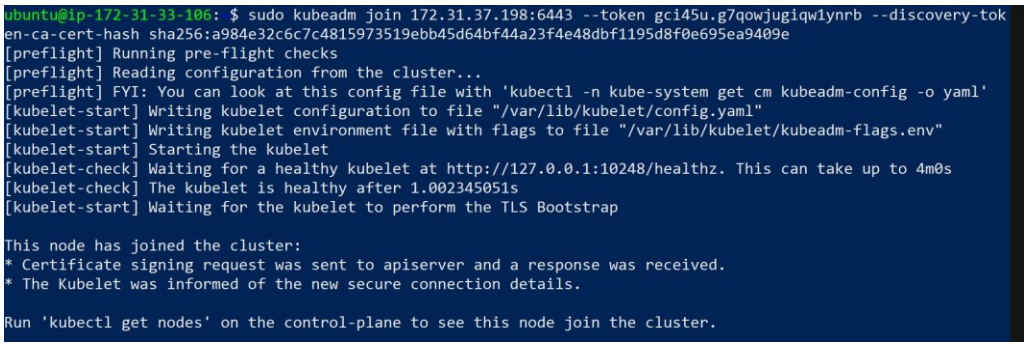
Run this command on master and also copy and save the Join command from above.
mkdir -p $HOME/.kube sudo cp -i /etc/kubernetes/admin.conf
$HOME/.kube/config sudo chown $(id -u):$(id -g) $HOME/.kube/config

```
ubuntu@ip-172-31-37-198: $ mkdir -p $HOME/.kube
ubuntu@ip-172-31-37-198: $ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
ubuntu@ip-172-31-37-198: $ sudo chown $(id -u):$(id -g) $HOME/.kube/config
ubuntu@ip-172-31-37-198: $ kubectl get nodes
NAME               STATUS     ROLES           AGE   VERSION
ip-172-31-37-198   NotReady   control-plane   12m   v1.31.1
```

Connect master and worker nodes by running this command on the worker logged in terminals: I ran the following command to do this, kubeadm join 172.31.37.198:6443 --token gci45u.g7qowjugiqw1ynrb \ --discovery-token-ca-cert-hash sha256:a984e32c6c7c4815973519ebb45d64bf44a23f4e48dbf1195d8f0e695ea9409e On node 1:

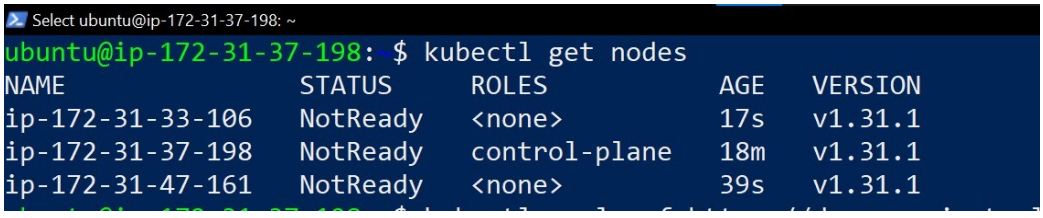


On node 2:



Run kubectl get nodes on the master instance to confirm the joining of the worker nodes
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

The command kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml deploys Calico, a networking and network security solution for Kubernetes, by applying the configuration specified in the provided YAML file from the Calico documentation. This sets up the necessary resources to enable networking capabilities within the Kubernetes cluster.



Name: Shreyash Kamat Div: D15C Roll No: 22

```
ubuntu@ip-172-31-37-198: $ kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml
poddisruptionbudget.policy/calico-kube-controllers created
serviceaccount/calico-kube-controllers created
serviceaccount/calico-node created
configmap/calico-config created
customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgppeers.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/blockaffinities.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/caliconodestatuses.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/clusterinformations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/felixconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworksets.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/hostendpoints.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamblocks.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamconfigs.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamhandles.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ippools.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipreservations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/kubecontrollersconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networksets.crd.projectcalico.org created
clusterrole.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrole.rbac.authorization.k8s.io/calico-node created
clusterrolebinding.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrolebinding.rbac.authorization.k8s.io/calico-node created
daemonset.apps/calico-node created
```

sudo systemctl status kubelet

```
ubuntu@ip-172-31-37-198: $ 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 Thu 2024-09-26 10:13:42 UTC; 18min ago
       Docs: https://kubernetes.io/docs/
   Main PID: 5773 (kubelet)
      Tasks: 10 (limit: 4676)
     Memory: 32.6M (peak: 33.3M)
        CPU: 18.442s
     CGroup: /system.slice/kubelet.service
             └─5773 /usr/bin/kubelet --bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf --kubeconfig

Sep 26 10:32:24 ip-172-31-37-198 kubelet[5773]: I0926 10:32:24.596915    5773 reconciler_common.go:245] "opera
Sep 26 10:32:24 ip-172-31-37-198 kubelet[5773]: I0926 10:32:24.596938    5773 reconciler_common.go:245] "opera
Sep 26 10:32:24 ip-172-31-37-198 kubelet[5773]: I0926 10:32:24.596958    5773 reconciler_common.go:245] "opera
Sep 26 10:32:24 ip-172-31-37-198 kubelet[5773]: I0926 10:32:24.596978    5773 reconciler_common.go:245] "opera
Sep 26 10:32:24 ip-172-31-37-198 kubelet[5773]: I0926 10:32:24.596997    5773 reconciler_common.go:245] "opera
Sep 26 10:32:24 ip-172-31-37-198 kubelet[5773]: I0926 10:32:24.597017    5773 reconciler_common.go:245] "opera
Sep 26 10:32:25 ip-172-31-37-198 kubelet[5773]: I0926 10:32:25.193047    5773 scope.go:117] "RemoveContainer"
Sep 26 10:32:25 ip-172-31-37-198 kubelet[5773]: E0926 10:32:25.193206    5773 pod_workers.go:1301] "Error sync
Sep 26 10:32:27 ip-172-31-37-198 kubelet[5773]: E0926 10:32:27.544395    5773 kubelet.go:2902] "Container runt
Sep 26 10:32:32 ip-172-31-37-198 kubelet[5773]: E0926 10:32:32.545352    5773 kubelet.go:2902] "Container runt
lines 1-23/23 (END)
ubuntu@ip-172-31-37-198: $
```

kubectl get nodes -o wide helps us get to know that the Status is ready.

```
ubuntu@ip-172-31-37-198: $ kubectl get nodes -o wide
NAME               STATUS   ROLES           AGE   VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE           KE
RNEL-VERSION   CONTAINER-RUNTIME
ip-172-31-33-106   Ready    <none>          66s   v1.31.1   172.31.33.106   <none>        Ubuntu 24.04 LTS   6.
8.0-1012-aws   containerd://1.7.12
ip-172-31-37-198   Ready    control-plane   19m   v1.31.1   172.31.37.198   <none>        Ubuntu 24.04 LTS   6.
8.0-1012-aws   containerd://1.7.12
ip-172-31-47-161   Ready    <none>          88s   v1.31.1   172.31.47.161   <none>        Ubuntu 24.04 LTS   6.
8.0-1012-aws   containerd://1.7.12
ubuntu@ip-172-31-37-198: $ kubectl label node ip-172-31-28-117 kubernetes.io/role=Node1
Error from server (NotFound): nodes "ip-172-31-28-117" not found
ubuntu@ip-172-31-37-198: $ kubectl label node ip-172-31-47-161 kubernetes.io/role=Node1
node/ip-172-31-47-161 labeled
ubuntu@ip-172-31-37-198: $ kubectl label node ip-172-31-33-106 kubernetes.io/role=Node2
node/ip-172-31-33-106 labeled
ubuntu@ip-172-31-37-198: $ kubectl get nodes -o wide
NAME               STATUS   ROLES           AGE     VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE
KERNEL-VERSION   CONTAINER-RUNTIME
ip-172-31-33-106   Ready    Node2           3m26s   v1.31.1   172.31.33.106   <none>        Ubuntu 24.04 LTS
6.8.0-1012-aws   containerd://1.7.12
ip-172-31-37-198   Ready    control-plane   21m     v1.31.1   172.31.37.198   <none>        Ubuntu 24.04 LTS
6.8.0-1012-aws   containerd://1.7.12
ip-172-31-47-161   Ready    Node1           3m48s   v1.31.1   172.31.47.161   <none>        Ubuntu 24.04 LTS
6.8.0-1012-aws   containerd://1.7.12
```

Or else run kubectl get nodes command

```
ubuntu@ip-172-31-37-198:~$ kubectl get nodes
NAME                STATUS   ROLES           AGE     VERSION
ip-172-31-33-106    Ready    Node2           3m42s   v1.31.1
ip-172-31-37-198    Ready    control-plane   21m     v1.31.1
ip-172-31-47-161    Ready    Node1           4m4s    v1.31.1
ubuntu@ip-172-31-37-198:~$
```

Conclusion: In this experiment, we successfully set up a Kubernetes cluster on AWS by creating the necessary infrastructure and configuring security groups. We installed Docker and Kubernetes components, initialized the master node, and connected the worker nodes. By deploying Calico for networking, we enabled effective communication within the cluster. This hands-on experience enhanced our understanding of Kubernetes architecture and equipped us with practical skills for managing containerized applications in a cloud environment.

Name: Shreyash Kamat Div: D15C Roll No: 22

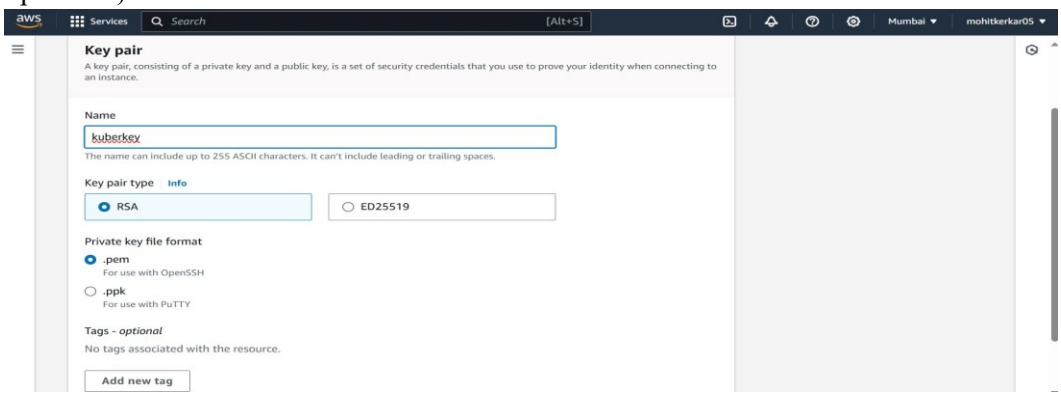
Adv DevOps Lab Exp 04

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

(I performed this Experiment on my personal AWS account)

Step 1: Creation of EC2 Instance, its Key-pair, remote login using SSH

Firstly, create a key pair with a desired name and type 'RSA' with .pem extension. (.pem is useful with OpenSSH)

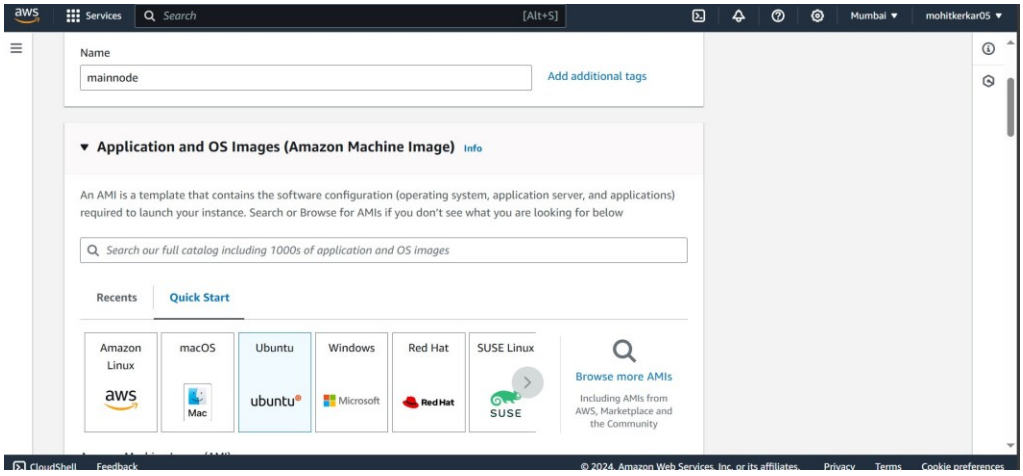


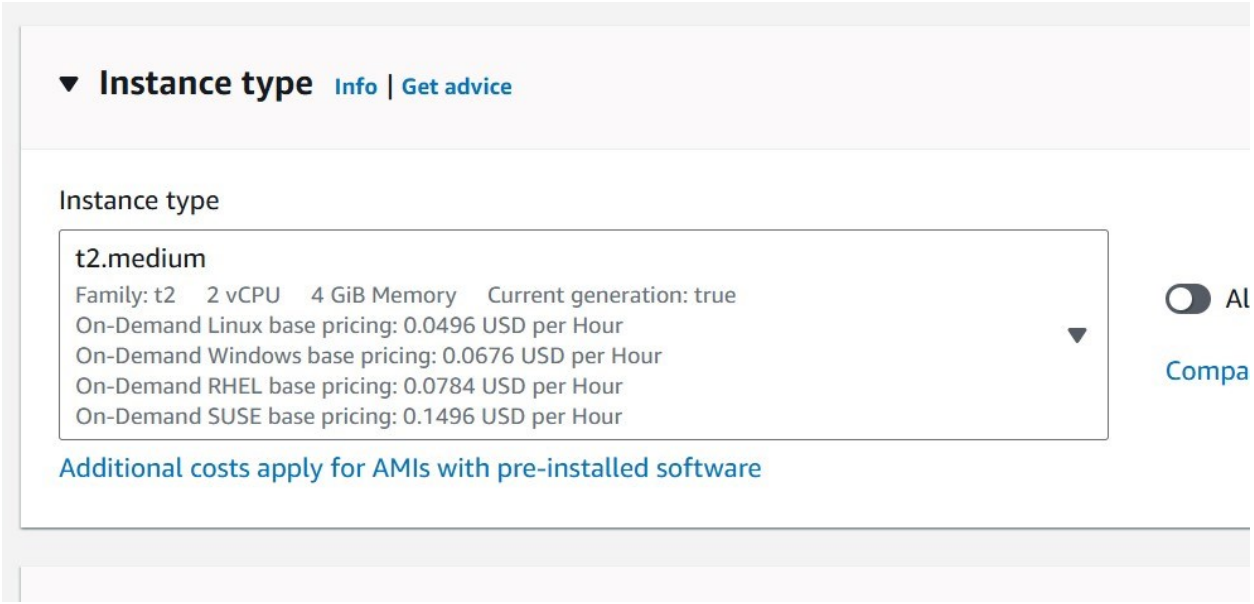
After creating the key, it automatically gets downloaded onto your machine. We are to create a new folder and place our within there. After this, we are to access our key and establish our connection with our instance by changing the directory to our key's directory

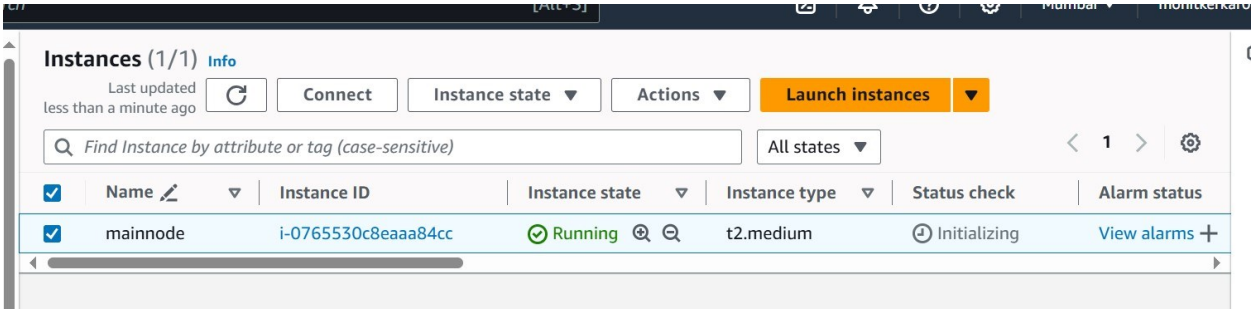
Now, proceed to the creation of the EC2 instance. I selected Amazon machine image as **Ubuntu** and instance type as **t2.medium**, just for the purpose of convenience and also for successful execution of the experiment.

The execution of the experiment is determined by the fact that how many resources is the instance allowed to access or use. The Free Tier eligible instance type **t2.micro avails the instance with only one** CPU, whereas with t2.medium we get 2 CPU and much more execution space.

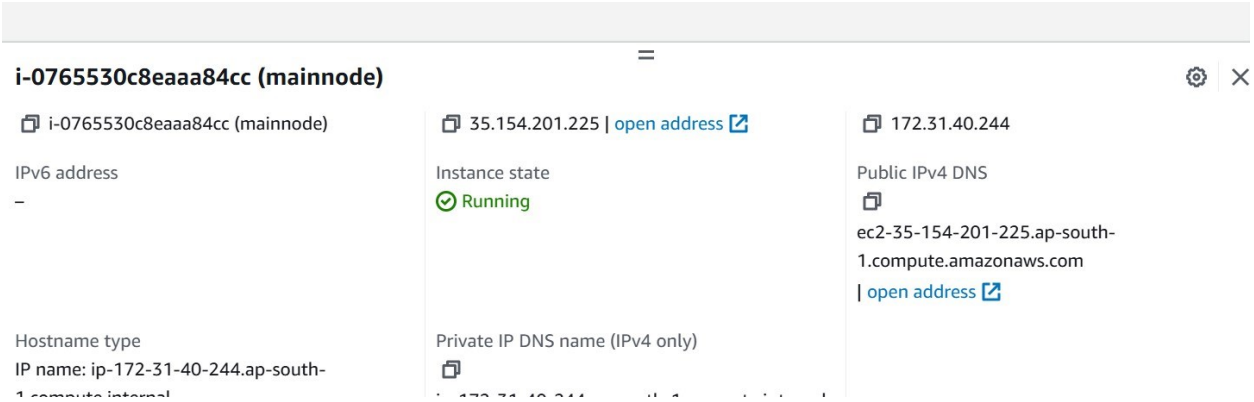
Be mindful that along with all this, **t2.medium does not have free tier eligibility**. So, use the instances and the resources that you allow them to access resourcefully and economically. Shut/terminate them as and when work related to them is over



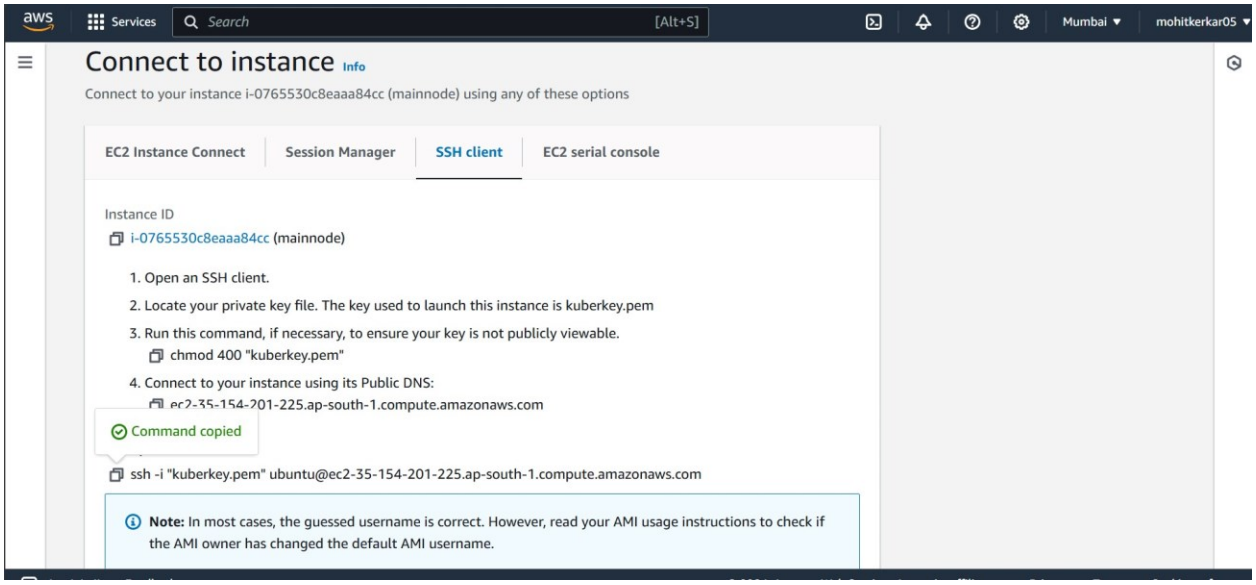




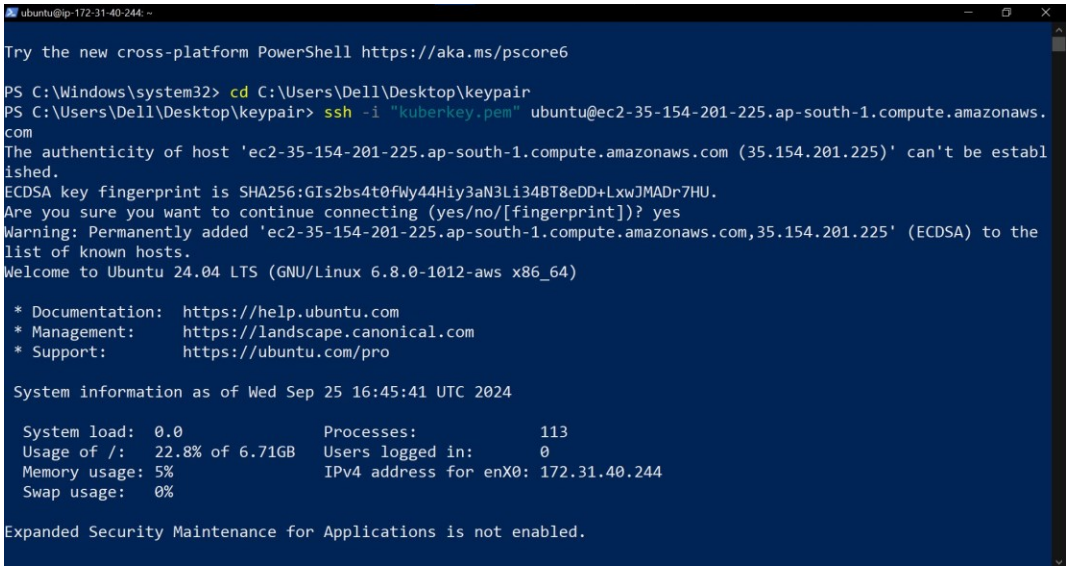
Here are my instance details. From here on, after remotely logging in from my local terminal, you would see my IPv4 address in my terminal (Powershell)



Select the instance and press connect. Navigate to the SSH client section and copy the SSH command provided to us. We will require this for remotely logging into our instance from our local terminal.



As one can see, I changed my directory to the folder in which i had stored my key (.pem) and ran the ssh command so as to remotely login onto my instance and perform the necessary tasks from my terminal itself



Step 2: Docker installation

Run the below mentioned commands in order to install docker on your instance

Adding key: curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add **Adding key without apt-key:** curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/trusted.gpg.d/docker.gpg > /dev/null

Add docker repository: sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"

```
ubuntu@ip-172-31-40-244: $ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
ubuntu@ip-172-31-40-244: $ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/trusted.
gpg.d/docker.gpg > /dev/null
ubuntu@ip-172-31-40-244: $ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $
(lsb_release -cs) stable"
Repository: 'deb [arch=amd64] https://download.docker.com/linux/ubuntu noble stable'
Description:
Archive for codename: noble components: stable
More info: https://download.docker.com/linux/ubuntu
Adding repository.
Press [ENTER] to continue or Ctrl-c to cancel.
Adding deb entry to /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-noble.list
Adding disabled deb-src entry to /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-nob
le.list
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 https://download.docker.com/linux/ubuntu noble InRelease [48.8 kB]
Get:5 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:6 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:7 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:8 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:9 https://download.docker.com/linux/ubuntu noble/stable amd64 Packages [15.3 kB]
Get:10 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
```

Run these commands to update the package list and install Docker:

sudo apt update

sudo apt install docker-ce docker-ce-cli containerd.io

```
ubuntu@ip-172-31-40-244: ~
nerd.service.
Setting up docker-compose-plugin (2.29.7-1~ubuntu.24.04~noble) ...
Setting up libltdl7:amd64 (2.4.7-7build1) ...
Setting up docker-ce-cli (5:27.3.1-1~ubuntu.24.04~noble) ...
Setting up libslirp0:amd64 (4.7.0-1ubuntu3) ...
Setting up pigz (2.8-1) ...
Setting up docker-ce-rootless-extras (5:27.3.1-1~ubuntu.24.04~noble) ...
Setting up slirp4netns (1.2.1-1build2) ...
Setting up docker-ce (5:27.3.1-1~ubuntu.24.04~noble) ...
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.ser
vice.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /usr/lib/systemd/system/docker.socket.
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for libc-bin (2.39-0ubuntu8.2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-40-244: $
```

Next, run this command…. sudo mkdir -p

/etc/docker cat <<EOF | sudo tee

/etc/docker/daemon.json

{

"exec-opts": ["native.cgroupdriver=systemd"]

}

EOF

sudo systemctl enable docker

sudo systemctl daemon-reload

sudo systemctl restart docker

```
ubuntu@ip-172-31-40-244:~$ sudo mkdir -p /etc/docker
ubuntu@ip-172-31-40-244:~$ cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
ubuntu@ip-172-31-40-244:~$ 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
ubuntu@ip-172-31-40-244:~$ sudo systemctl daemon-reload
ubuntu@ip-172-31-40-244:~$ sudo systemctl restart docker
```

Step 3: Kubernetes Installation Run these commands to install kubernetes: 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

```
ubuntu@ip-172-31-40-244:~$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dear
mor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
ubuntu@ip-172-31-40-244:~$ echo "deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.
io/core:/stable:/v1.31/deb/ /" | Out-File -FilePath "C:\path\to\your\directory\kubernetes.list" -Encoding utf8
Out-File: command not found
ubuntu@ip-172-31-40-244:~$ 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/ /
ubuntu@ip-172-31-40-244:~$
```

After this step, run the following commands,
sudo apt-get update sudo apt-get install -y
kubelet kubeadm kubectl sudo apt-mark hold
kubelet kubeadm kubectl

Over here, while running sudo apt-get update if you encounter an error like this

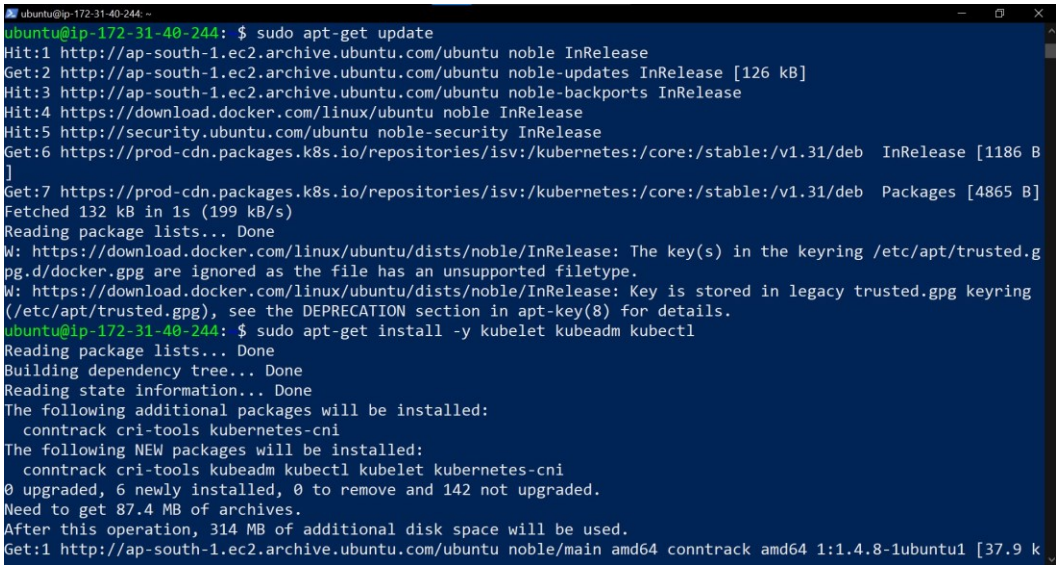
```
ubuntu@ip-172-31-40-244:~$ sudo apt-get update
E: Malformed entry 1 in list file /etc/apt/sources.list.d/kubernetes.list (URI)
E: The list of sources could not be read.
```

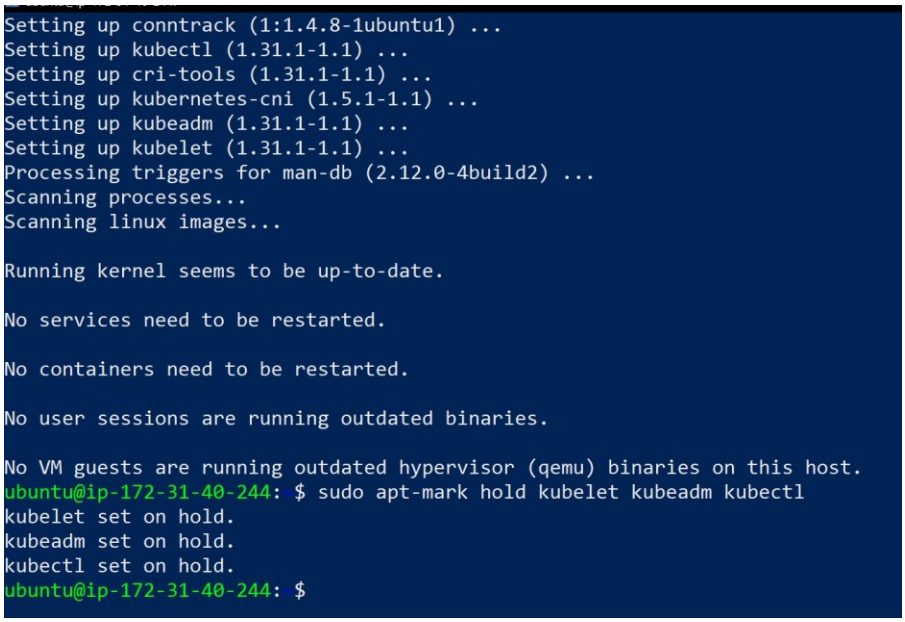
Then run the following command and ensure that the file contains the following texts:
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /

```
ubuntu@ip-172-31-40-244: ~
  GNU nano 7.2                    /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/ /
```

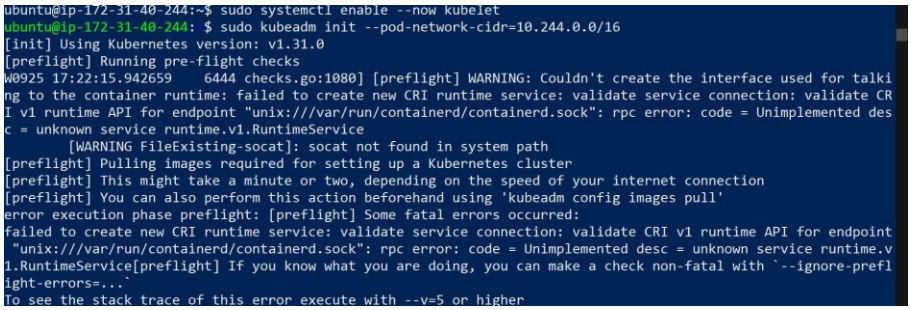
Be sure to save this file and then proceed with these commands

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





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



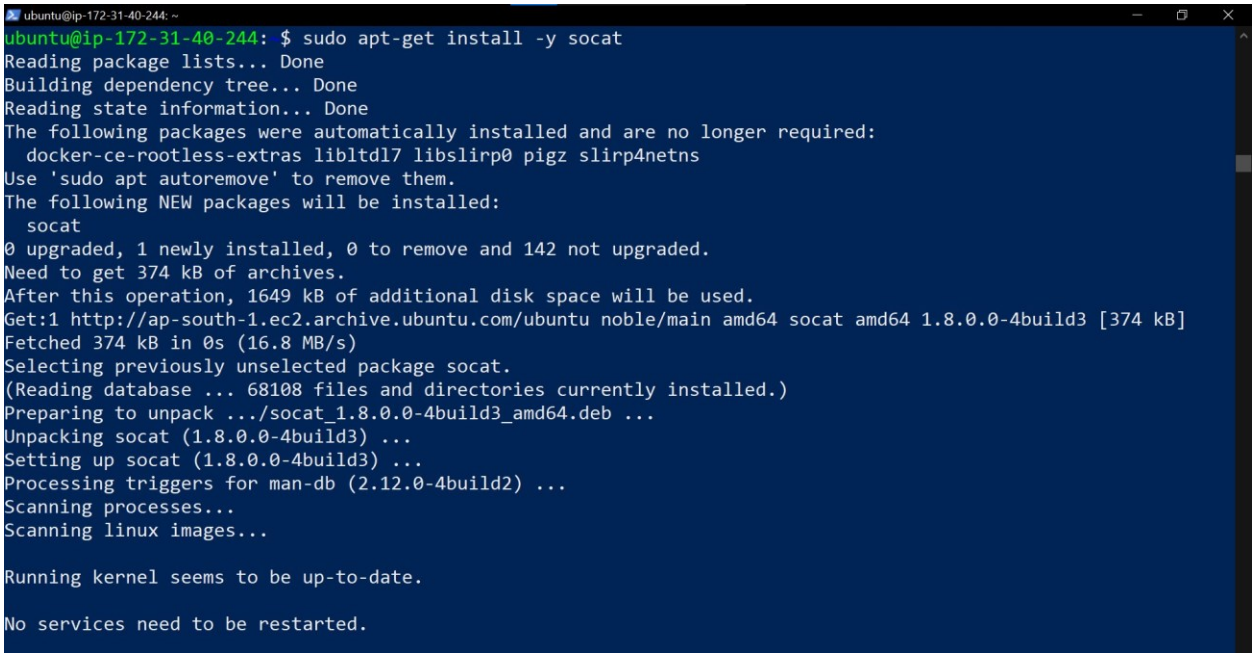
Since running that command ran us into an error, we are expected to install containerd first
Run this command: sudo apt-get install -y containerd
Then run, sudo mkdir -p /etc/containerd
sudo containerd config default | sudo tee /etc/containerd/config.toml

```
ubuntu@ip-172-31-40-244:~$ sudo apt-get update
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://ap-south-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
Hit:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  InRelease
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: The key(s) in the keyring /etc/apt/trusted.g
pg.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 DEPRECATION section in apt-key(8) for details.
ubuntu@ip-172-31-40-244:~$ 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-ce-rootless-extras libltdl7 libslirp0 pigz slirp4netns
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  runc
The following packages will be REMOVED:
  containerd.io docker-ce
The following NEW packages will be installed:
  containerd runc
0 upgraded, 2 newly installed, 2 to remove and 142 not upgraded.
Need to get 47.2 MB of archives.
```

Post this step, run these commands one by one
sudo systemctl restart containerd sudo
systemctl enable containerd sudo systemctl
status containerd

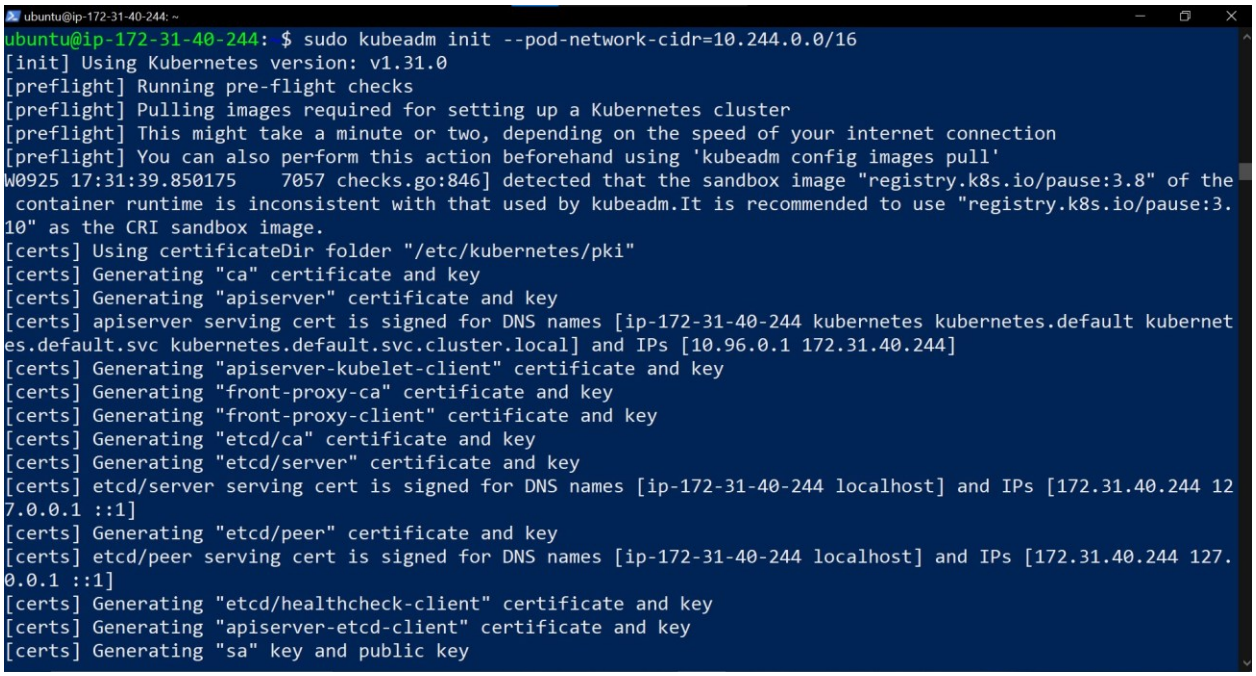
```
ubuntu@ip-172-31-40-244:~$ sudo systemctl restart containerd
ubuntu@ip-172-31-40-244:~$ sudo systemctl enable containerd
ubuntu@ip-172-31-40-244:~$ sudo systemctl status containerd
● containerd.service - containerd container runtime
     Loaded: loaded (/usr/lib/systemd/system/containerd.service; enabled; preset: enabled)
     Active: active (running) since Wed 2024-09-25 17:23:54 UTC; 17s ago
       Docs: https://containerd.io
   Main PID: 6555 (containerd)
      Tasks: 7
     Memory: 13.5M (peak: 14.1M)
        CPU: 121ms
     CGroup: /system.slice/containerd.service
             └─6555 /usr/bin/containerd

Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231856399Z" level=info msg="Start>
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231887455Z" level=info msg=servin>
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231896683Z" level=info msg="Start>
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231919602Z" level=info msg=servin>
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231952908Z" level=info msg="Start>
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231972572Z" level=info msg="Start>
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231980403Z" level=info msg="Start>
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231986601Z" level=info msg="Start>
Sep 25 17:23:54 ip-172-31-40-244 systemd[1]: Started containerd.service - containerd container runtime.
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.234185443Z" level=info msg="conta>
```

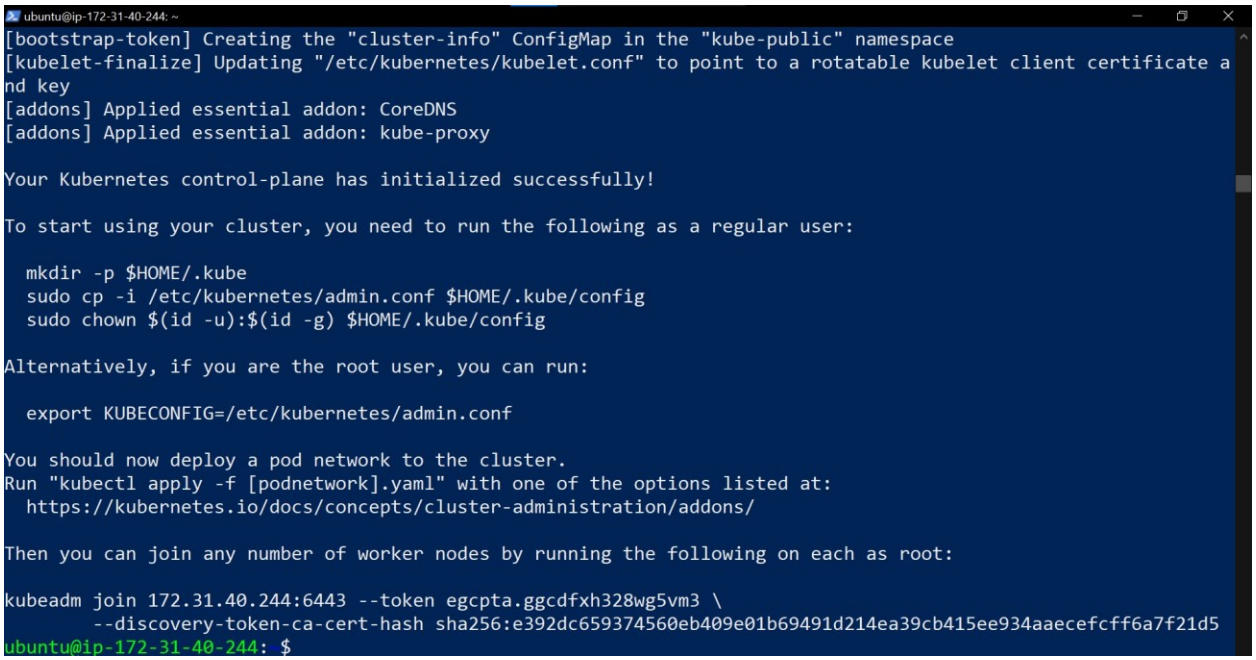
After this, run sudo apt-get
install -y socat



Step 4: Initialize 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

```
ubuntu@ip-172-31-40-244:~$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
namespace/kube-flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
```

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

```
ubuntu@ip-172-31-40-244:~$ kubectl apply -f https://k8s.io/examples/application/deployment.yaml
deployment.apps/nginx-deployment created
ubuntu@ip-172-31-40-244:~$
```

kubectl get pods

```
ubuntu@ip-172-31-40-244:~$ kubectl get pods
NAME                               READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-brgb7   0/1     Pending   0          48s
nginx-deployment-d556bf558-s5rtj   0/1     Pending   0          48s
ubuntu@ip-172-31-40-244:~$
```

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

```
ubuntu@ip-172-31-40-244: $ POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
ubuntu@ip-172-31-40-244: $ kubectl port-forward $POD_NAME 8080:80
error: unable to forward port because pod is not running. Current status=Pending
ubuntu@ip-172-31-40-244: $
```

To untaint all nodes, run the following command
kubectl taint nodes --all node-role.kubernetes.io/control-plane:NoSchedule-

Then run, kubectl get nodes

```
ubuntu@ip-172-31-40-244:~$ kubectl taint nodes --all node-role.kubernetes.io/control-plane:NoSchedule-
node/ip-172-31-40-244 untainted
ubuntu@ip-172-31-40-244:~$ kubectl get nodes
NAME              STATUS   ROLES           AGE   VERSION
ip-172-31-40-244   Ready    control-plane   16m   v1.31.1
ubuntu@ip-172-31-40-244:~$
```

Then run, kubectl get pods command again

```
ubuntu@ip-172-31-40-244:~$ kubectl get pods
NAME                               READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-brgb7   1/1     Running   0          7m15s
nginx-deployment-d556bf558-s5rtj   1/1     Running   0          7m15s
ubuntu@ip-172-31-40-244:~$
```

Since, Jenkins server/service is already occupying port 8080, i tried starting the nginx service at 8082 port number. Following is the command that i executed was, kubectl port-forward $POD_NAME 8082:80

```
ubuntu@ip-172-31-40-244:~$ kubectl port-forward svc/nginx 8082:80
Forwarding from 127.0.0.1:8082 -> 80
Forwarding from [::1]:8082 -> 80
Handling connection for 8082
```

The last step consisted of opening a new terminal and logging in using the SSH method that we used earlier and running the following command so as to finally confirm that we have successfully deployed our kubernetes application over the nginx server, using nginx service (opional) curl --head http://127.0.0.1:8082

```
Last login: Wed Sep 25 19:13:52 2024 from 45.112.58.76
ubuntu@ip-172-31-40-244:~$ curl --head http://127.0.0.1:8082
HTTP/1.1 200 OK
Server: nginx/1.14.2
Date: Wed, 25 Sep 2024 19:36:26 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
```

Conclusion:

In this experiment, we successfully deployed an Nginx web server on a Kubernetes cluster and exposed it using NodePort services. By creating the deployment and configuring the service, we demonstrated Kubernetes' ability to manage containerized applications efficiently.

The use of `kubectl` commands allowed us to check pod status and access the Nginx server locally via port forwarding on the EC2 instance. This experience provided valuable insights into service management and networking concepts within Kubernetes.

Overall, this experiment reinforced the fundamental skills needed for deploying cloud-native applications and laid the groundwork for exploring more advanced Kubernetes features in future projects.

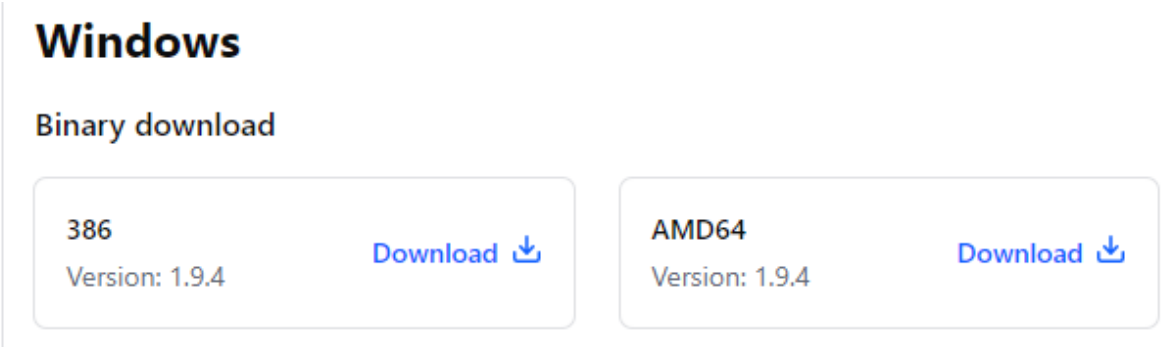
Name:Shreyash Kamat Div:D15C Roll no:22

Experiment 5

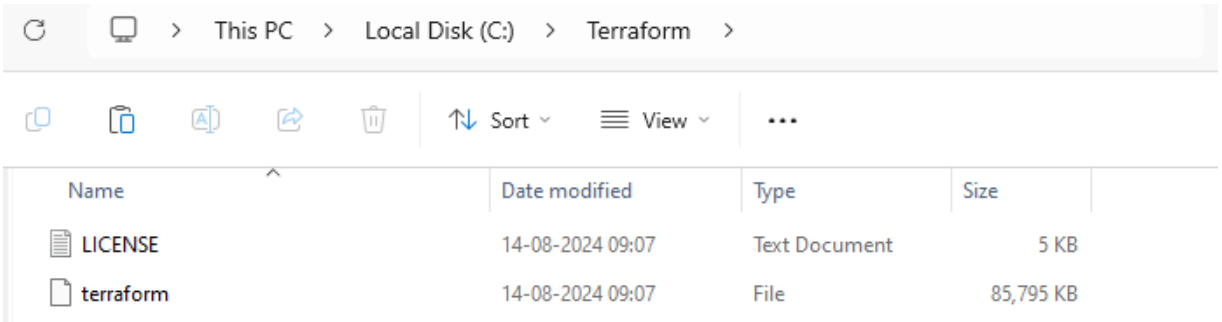
Aim:To Understand Terraform lifecycle,core concepts,technologies and install it on a Linux or Windows machine.

Installation and Configuration of Terraform in Windows

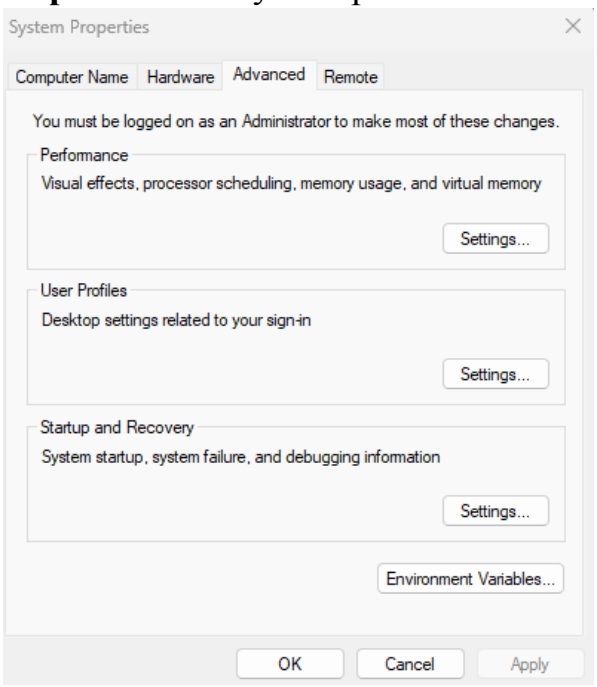
1: To install Terraform, First Download the Terraform Cli Utility for windows from terraforms official website: :https://www.terraform.io/downloads.html Select the Operating System Windows followed by either 32 bit or 64 bit based on your OS type.

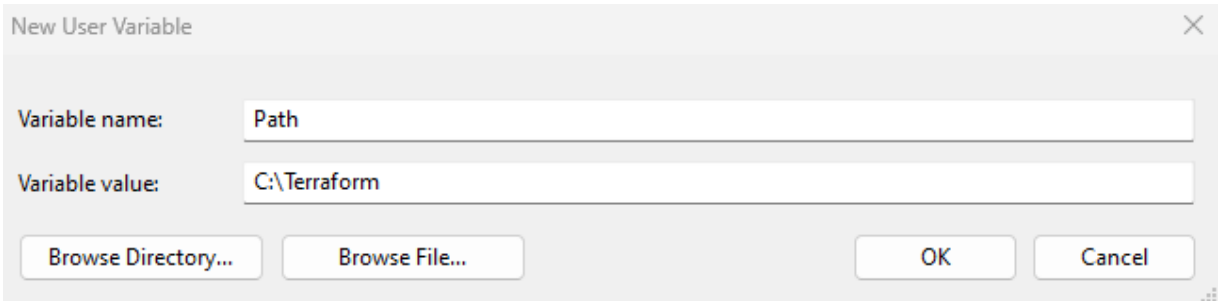


Step 2: Extract the downloaded setup file Terraform.exe in C:\Terraform directory

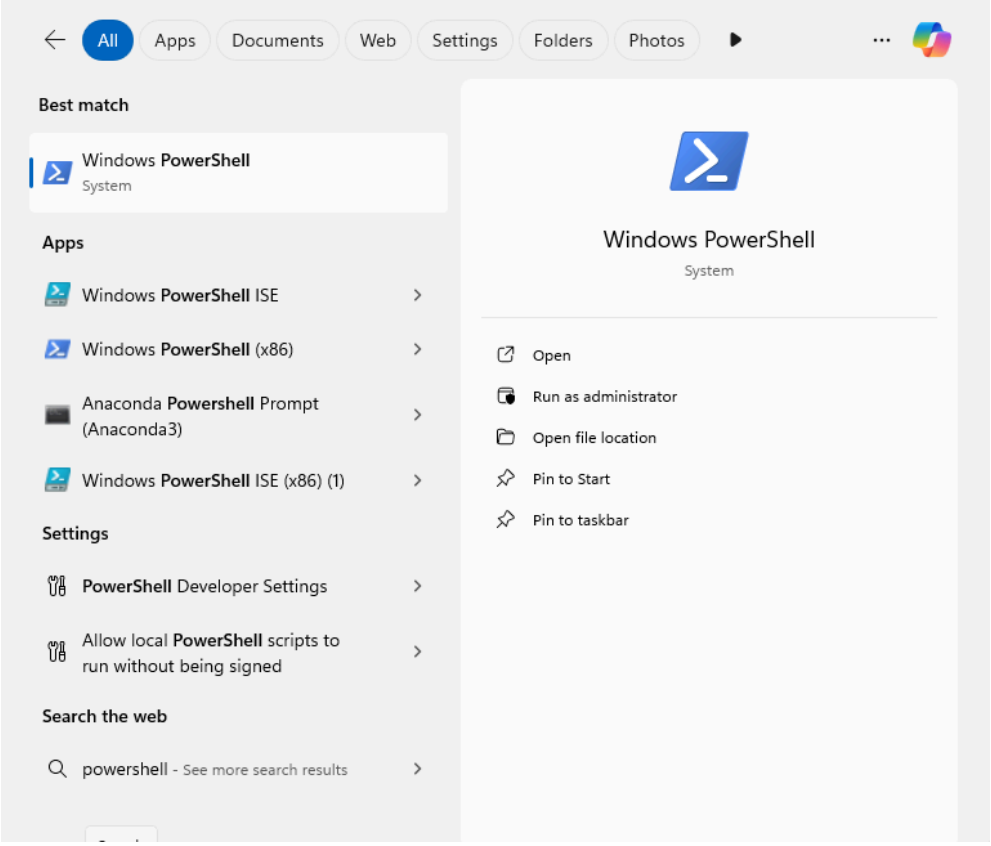


Step 3: Set the System path for Terraform in Environment Variables





Step 4: Open PowerShell with Admin Access



Step 5 : Open Terraform in PowerShell and check its functionality

```
PS C:\Users\student.VESIT505-17.000> terraform
Usage: terraform [global options] <subcommand> [args]

The available commands for execution are listed below.
The primary workflow commands are given first, followed by
less common or more advanced commands.

Main commands:
  init          Prepare your working directory for other commands
  validate      Check whether the configuration is valid
  plan          Show changes required by the current configuration
  apply         Create or update infrastructure
  destroy       Destroy previously-created infrastructure

All other commands:
  console       Try Terraform expressions at an interactive command prompt
  fmt           Reformat your configuration in the standard style
  force-unlock  Release a stuck lock on the current workspace
  get           Install or upgrade remote Terraform modules
  graph         Generate a Graphviz graph of the steps in an operation
  import        Associate existing infrastructure with a Terraform resource
  login         Obtain and save credentials for a remote host
  logout        Remove locally-stored credentials for a remote host
  metadata      Metadata related commands
  output        Show output values from your root module
  providers     Show the providers required for this configuration
  refresh       Update the state to match remote systems
  show          Show the current state or a saved plan
  state         Advanced state management
  taint         Mark a resource instance as not fully functional
```

Name: Shreyash Kamat **Div/Roll no: D15C/22**

Experiment No: 6

Implementation:

A. Creating docker image using terraform

Prerequisite:

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

Step 1: Check the docker functionality

```
PS C:\Users\INFT505-07> docker

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Common Commands:
  run         Create and run a new container from an image
  exec        Execute a command in a running container
  ps          List containers
  build       Build an image from a Dockerfile
  pull        Download an image from a registry
  push        Upload an image to a registry
  images      List images
  login       Log in to a registry
  logout      Log out from a registry
  search      Search Docker Hub for images
  version     Show the Docker version information
  info        Display system-wide information

Management Commands:
  builder     Manage builds
  buildx*     Docker Buildx (Docker Inc., v0.11.2-desktop.5)
  compose*    Docker Compose (Docker Inc., v2.22.0-desktop.2)
  container   Manage containers
  context     Manage contexts
  dev*        Docker Dev Environments (Docker Inc., v0.1.0)
  extension*  Manages Docker extensions (Docker Inc., v0.2.20)
  image       Manage images
  init*       Creates Docker-related starter files for your project (Docker Inc., v0.1.0-beta.8)
  manifest    Manage Docker image manifests and manifest lists
  network     Manage networks
  plugin      Manage plugins
  sbom*       View the packaged-based Software Bill Of Materials (SBOM) for an image (Anchore Inc., 0.6.0)
  scan*       Docker Scan (Docker Inc., v0.26.0)
```

```
PS C:\Users\INFT505-07> docker --version
Docker version 24.0.6, build ed223bc
PS C:\Users\INFT505-07>
```

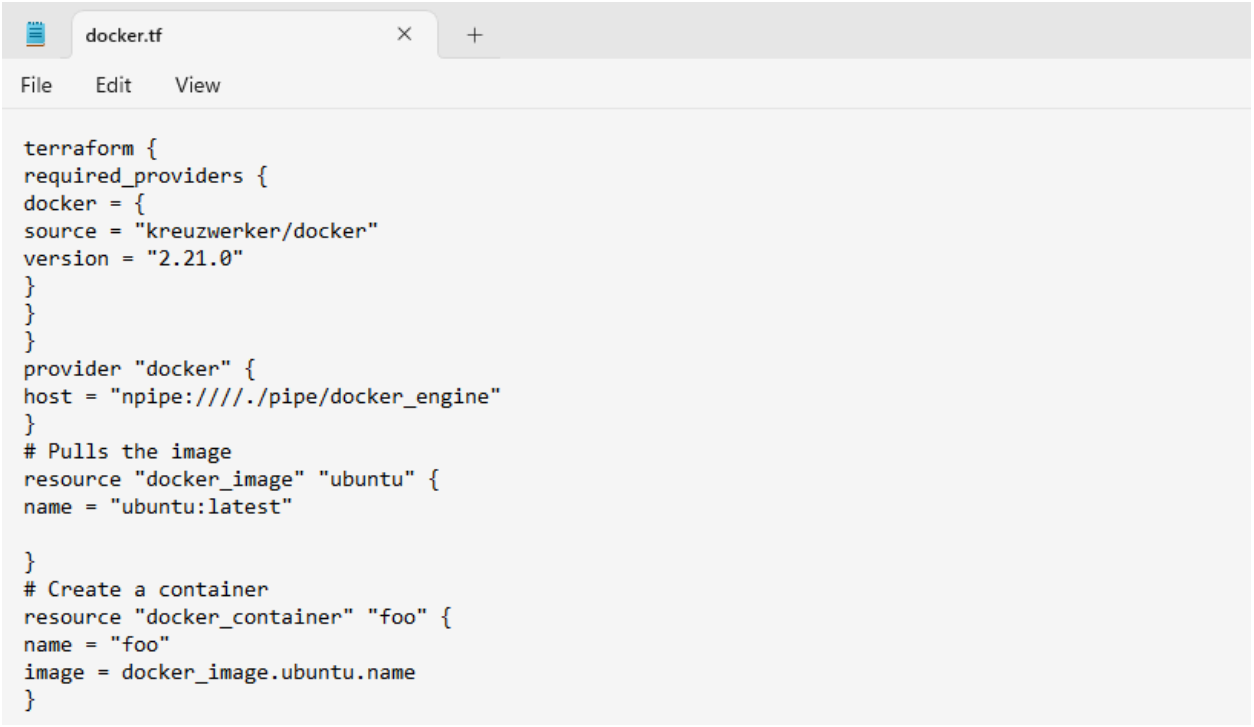
Now, create a folder named 'Terraform Scripts' in which we save our different types of scripts which will be further used in this experiment.

Step 2: Firstly create a new folder named 'Docker' in the 'TerraformScripts' folder. Then create a new docker.tf file using Atom editor and write the followingcontents into it to create a Ubuntu Linux container.

Script:

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

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

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

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

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

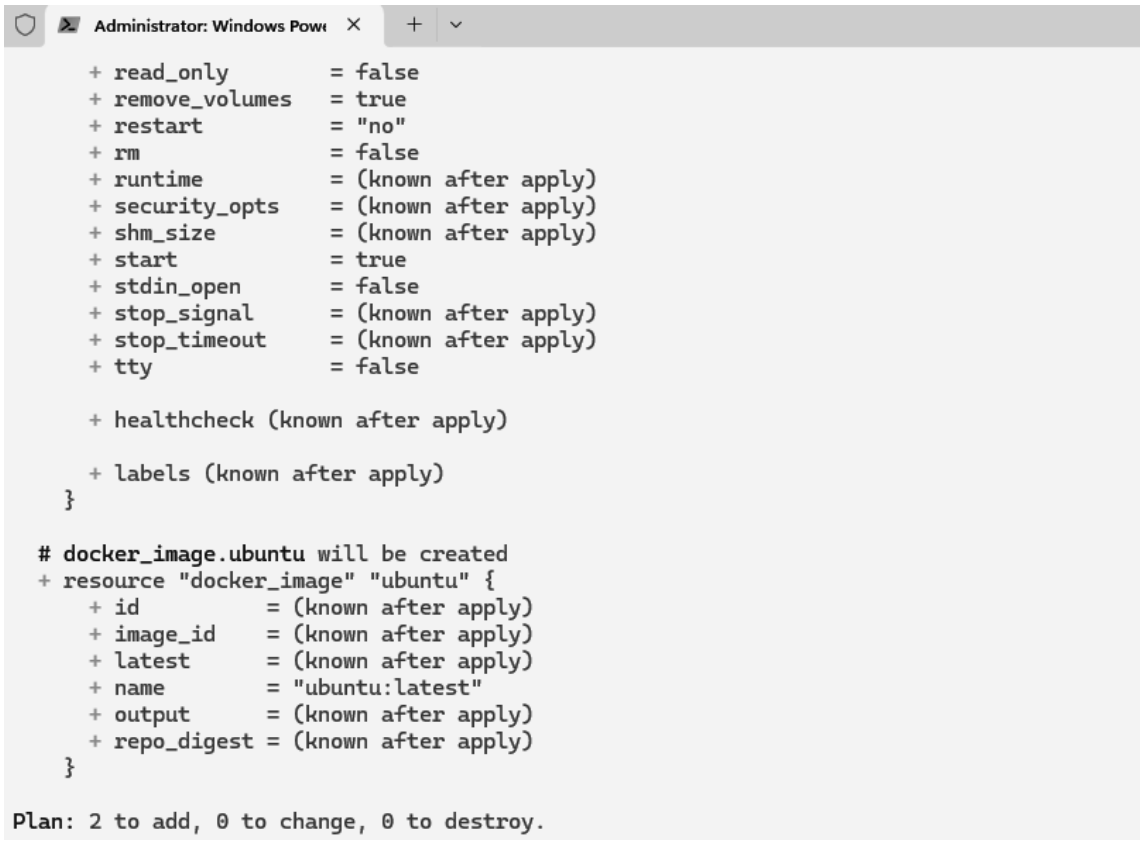


Step 3: Execute Terraform Init command to initialize the resources

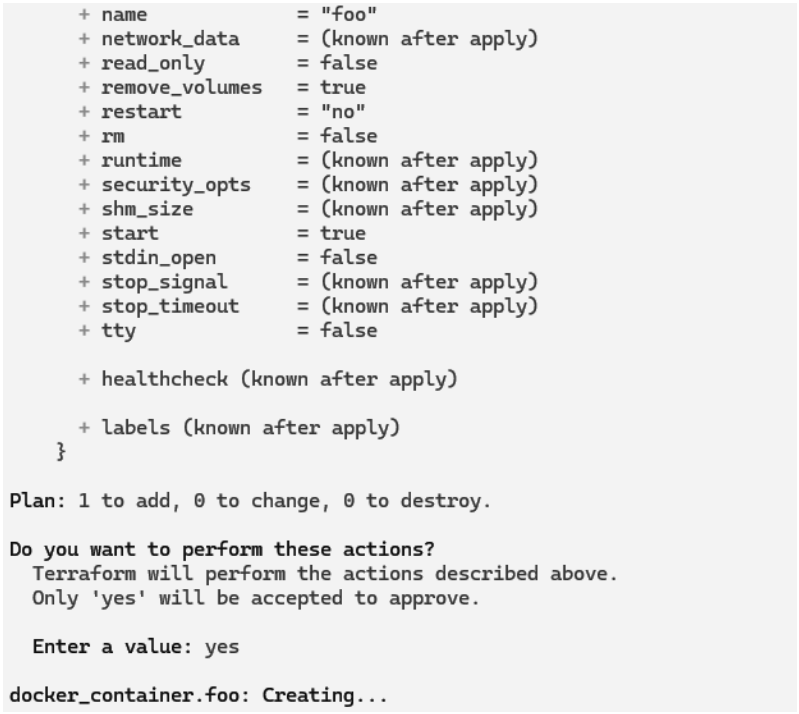
```
PS C:\terraform_scripts\Docker> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding kreuzwerker/docker versions matching "2.21.0"...
- Installing kreuzwerker/docker v2.21.0...
- Installed kreuzwerker/docker v2.21.0 (self-signed, key ID BD080C4571C6104C)
Partner and 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

```
PS C:\terraform_scripts\Docker> terraform plan

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

Terraform will perform the following actions:

  # docker_container.foo will be created
  + resource "docker_container" "foo" {
      + attach           = false
      + bridge           = (known after apply)
      + command          = (known after apply)
      + container_logs   = (known after apply)
      + entrypoint       = (known after apply)
      + env              = (known after apply)
      + exit_code        = (known after apply)
      + gateway          = (known after apply)
      + hostname         = (known after apply)
      + id               = (known after apply)
      + image            = "ubuntu:latest"
      + 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)
```



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"**



Docker images, Before Executing Apply step:

```
PS C:\Users\91773\Desktop\College Resources\TerraformScripts\Docker> docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
```

Docker images, After Executing Apply step:

```
PS C:\terraform_scripts\Docker> docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
ubuntu       latest    edbfe74c41f8   2 weeks ago   78.1MB
```

Step 6: Execute Terraform destroy to delete the configuration, which will automatically delete the Ubuntu Container.

```
PS C:\terraform_scripts\Docker> terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]

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

Terraform will perform the following actions:

  # docker_image.ubuntu will be destroyed
  - resource "docker_image" "ubuntu" {
      - id          = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest" -> null
      - image_id    = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a" -> null
      - latest      = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a" -> null
      - name        = "ubuntu:latest" -> null
      - repo_digest = "ubuntu@sha256:8a37d68f4f73ebf3d4efafbcf66379bf3728902a8038616808f04e34a9ab63ee" -> null
    }

Plan: 0 to add, 0 to change, 1 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_image.ubuntu: Destroying... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_image.ubuntu: Destruction complete after 0s

Destroy complete! Resources: 1 destroyed.
```

Docker images After Executing Destroy step

```
PS C:\terraform_scripts\Docker> docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
```

Name:Shreyash Kamat **Div/Roll No:D15C/22**

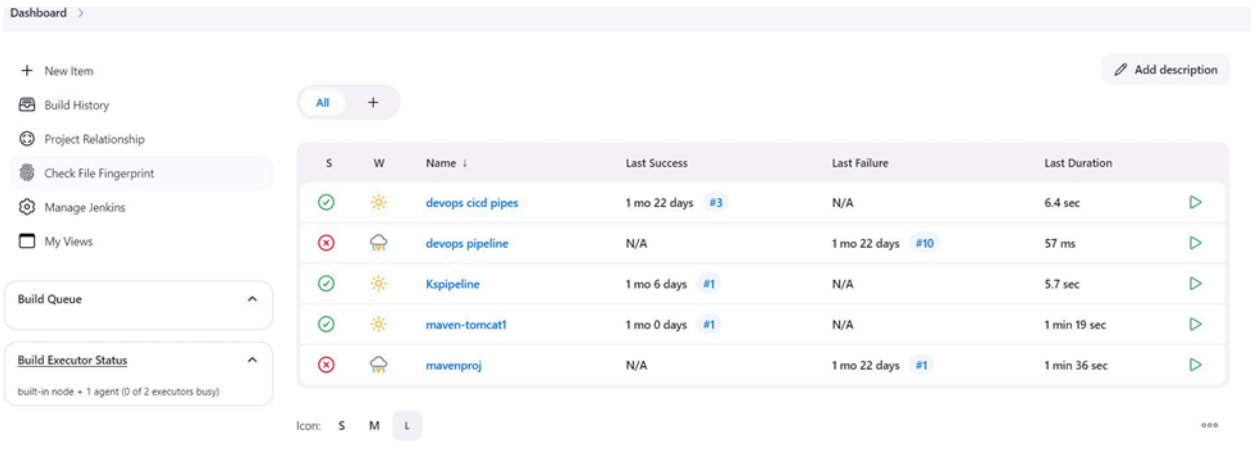
Exp 7:Understand Static Analysis SAST process and learn to integrate Jenkins SAST to SonarQube/GitLab.

Integrating Jenkins with SonarQube:

- Jenkins installed
- Docker Installed (for SonarQube)
- SonarQube Docker Image

Steps to integrate Jenkins with SonarQube

1. Open up Jenkins Dashboard on localhost, port 8090 or whichever port it is at for you.



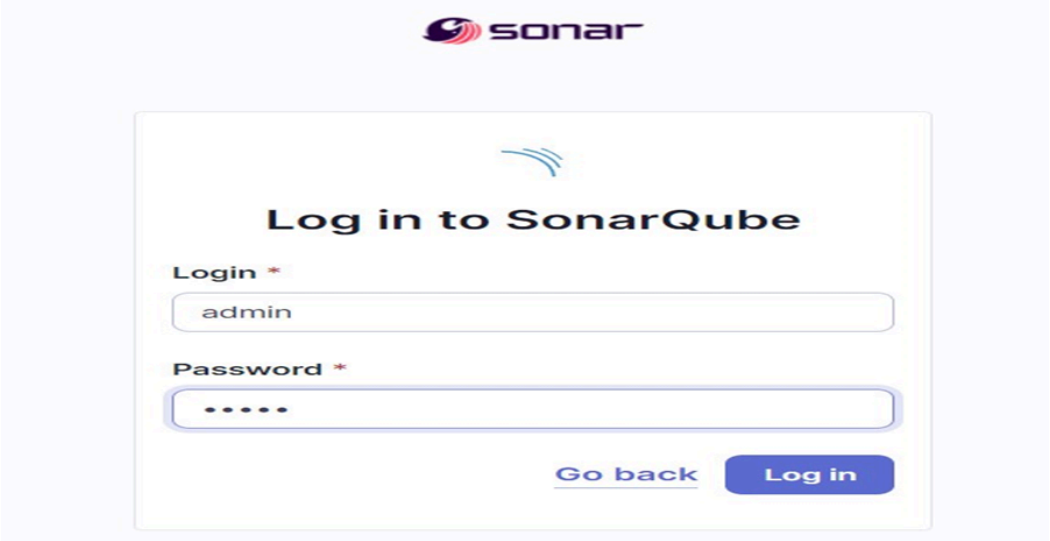
2. Run SonarQube in a Docker container using this command -

docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest

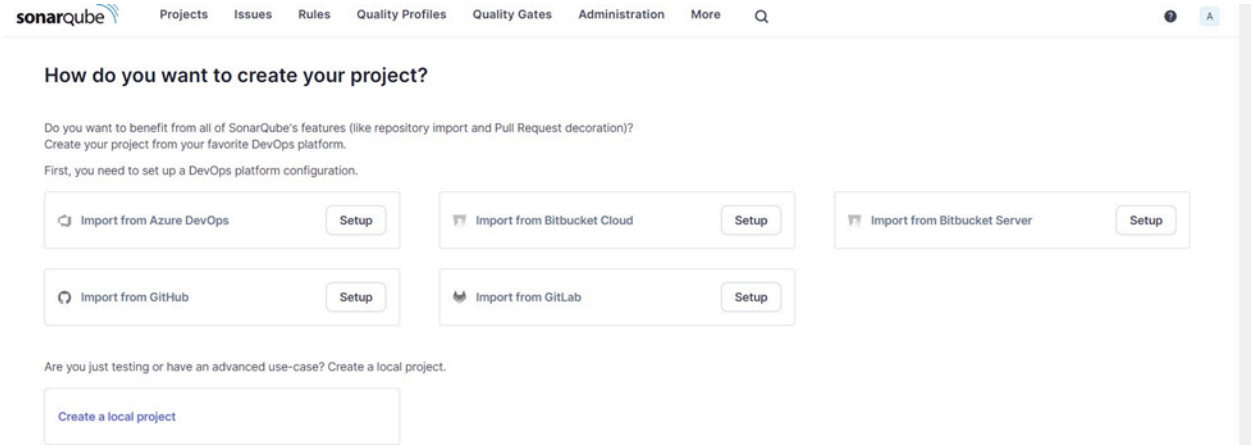
----------Warning: run below command only once

```
PS C:\Users\91773\Desktop\College Resources\Exp7 adv devops> docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS
_DISABLE=true -p 9000:9000 sonarqube:latest
Unable to find image 'sonarqube:latest' locally
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
77e678cded2ef5f989912d3d9e6991dd548eac03faa1eed68dd906614be53acc
PS C:\Users\91773\Desktop\College Resources\Exp7 adv devops>
```

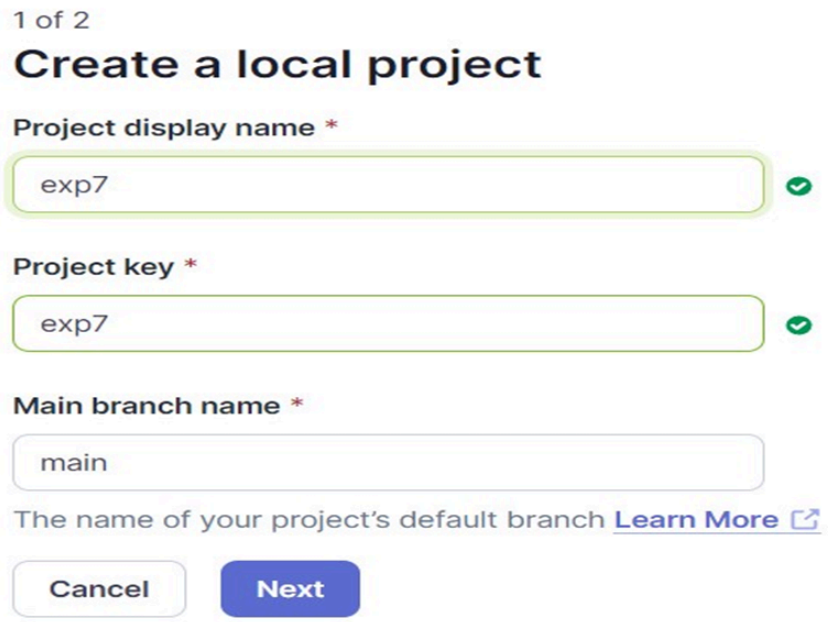
3. Once the container is up and running, you can check the status of SonarQube at localhost port 9000.



4. Login to SonarQube using username admin and password admin.

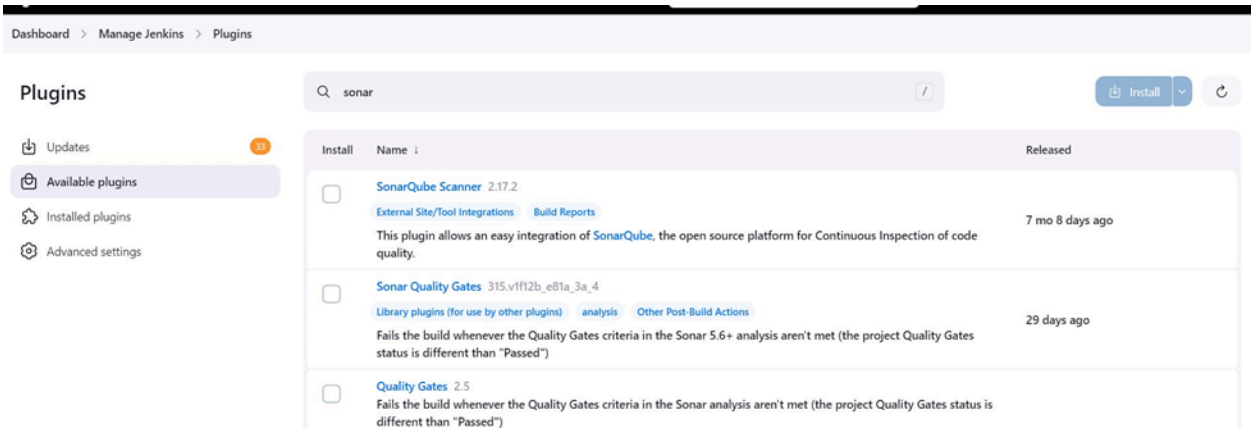


5. Create a manual project in SonarQube with the name sonarqube



Setup the project and come back to Jenkins Dashboard.

Go to Manage Jenkins and search for SonarQube Scanner for Jenkins and install it.



6. Under Jenkins 'Manage Jenkins' then go to 'system', scroll and look for **SonarQube Servers** and enter the details.

Enter the Server Authentication token if needed.

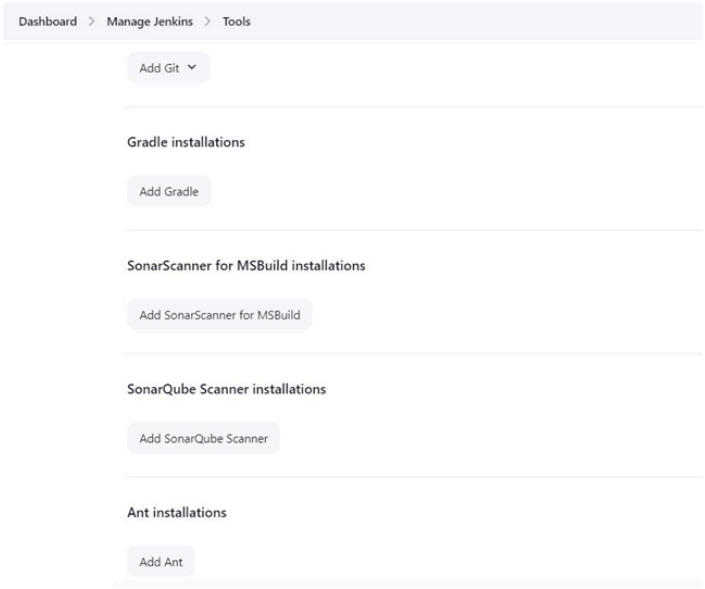
In SonarQube installations: Under **Name** add <project name of sonarqube> for me

exp7

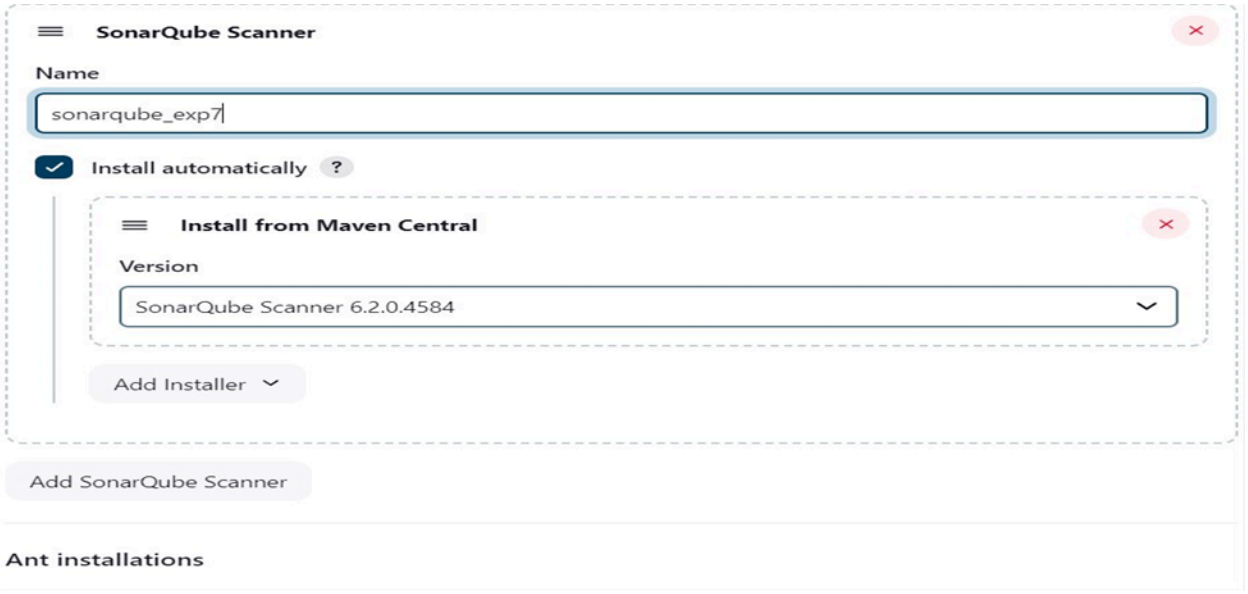
In **Server URL** Default is **http://localhost:9000**

7. Search for SonarQube Scanner under Global Tool Configuration. Choose the latest configuration and choose Install automatically.

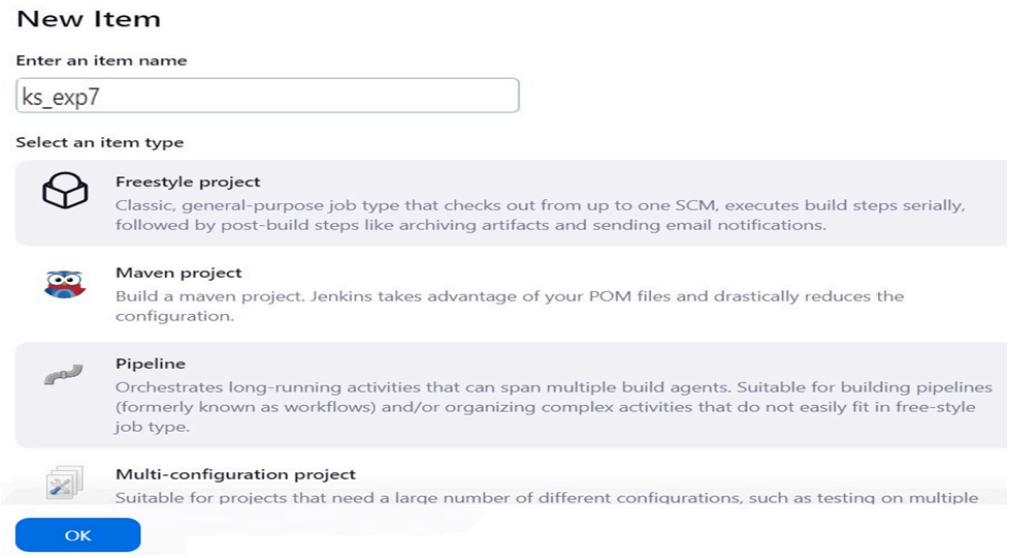
Dashboard > Manage Jenkins > Tools



Check the “Install automatically” option. → Under name any name as identifier
→ Check the “Install automatically” option.



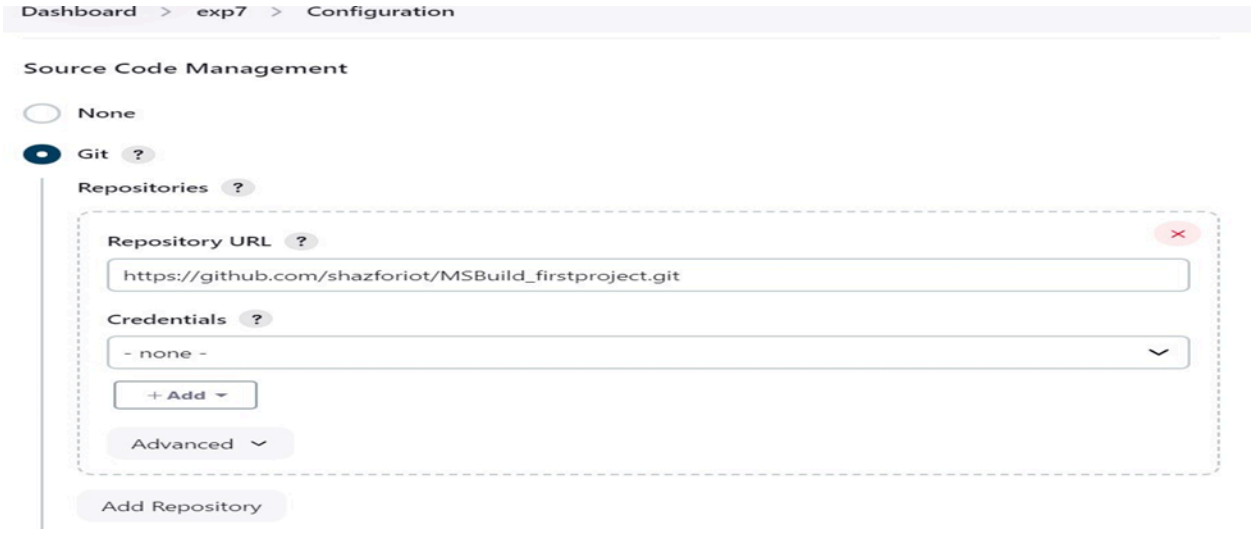
8. After the configuration, create a New Item in Jenkins, choose a freestyle project with name ks_exp7



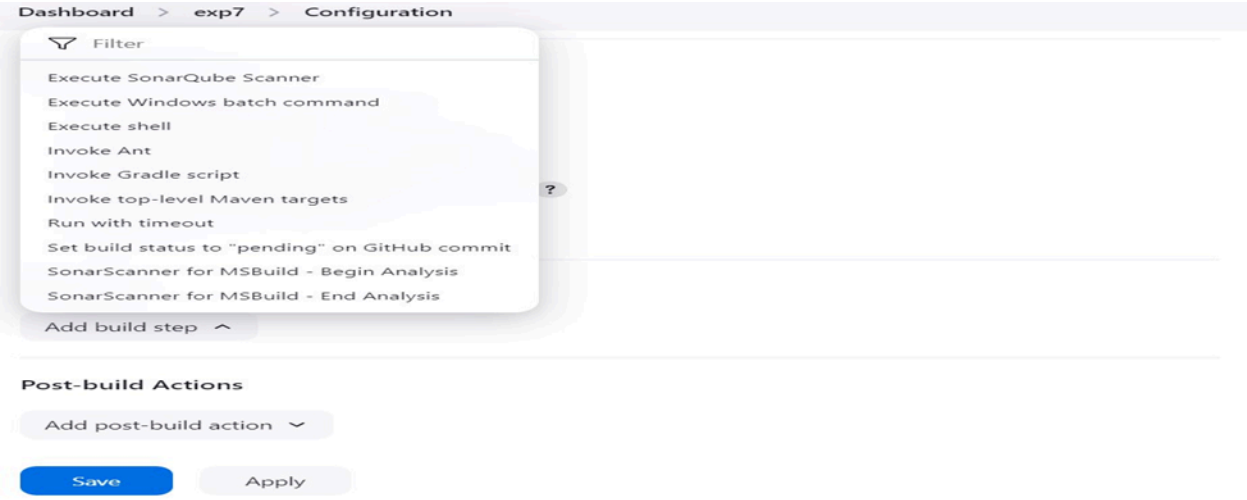
9. Choose this GitHub repository in Source Code Management.

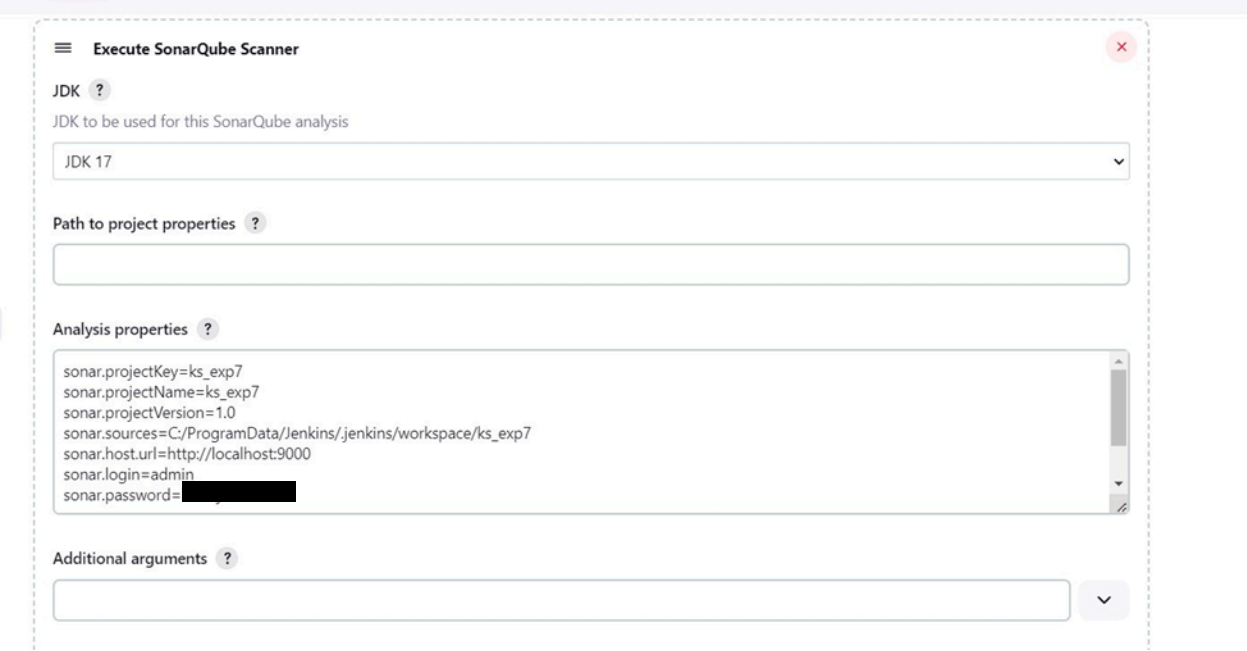
https://github.com/shazforiot/MSBuild_firstproject.git

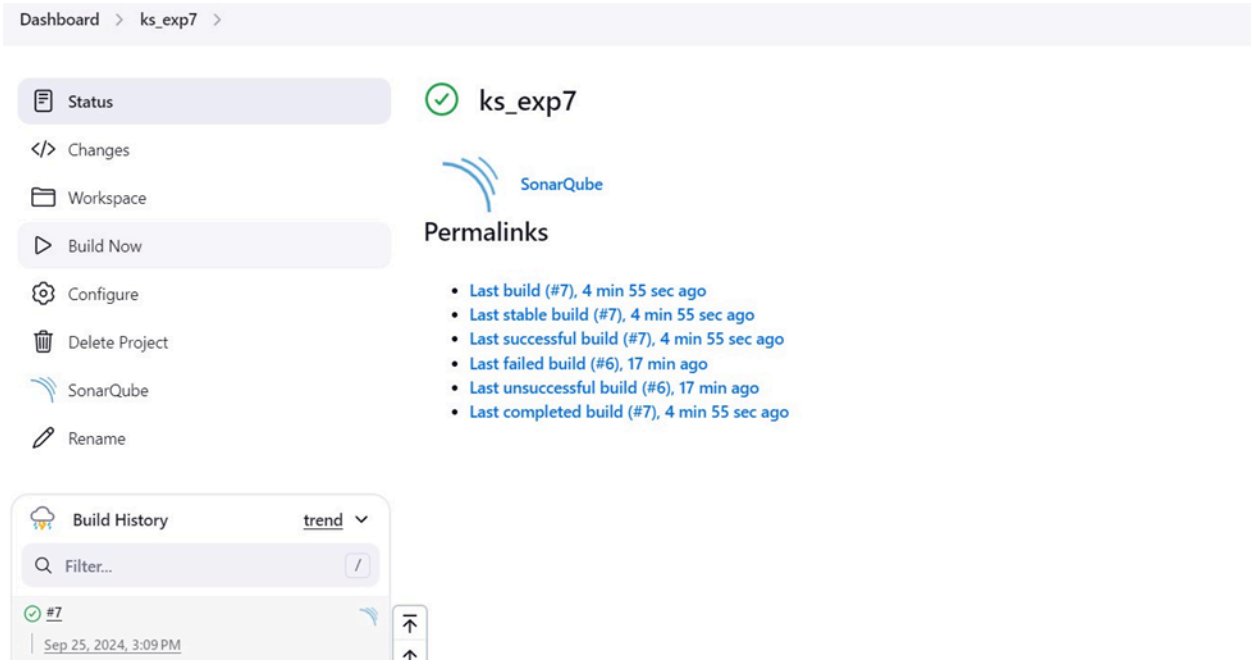
It is a sample hello-world project with no vulnerabilities and issues, just to test the integration.



10. Under **Select project → Configuration → Build steps → Execute SonarQube Scanner**, enter these Analysis properties. Mention the SonarQube Project Key, Login, Password, Source path and Host URL.





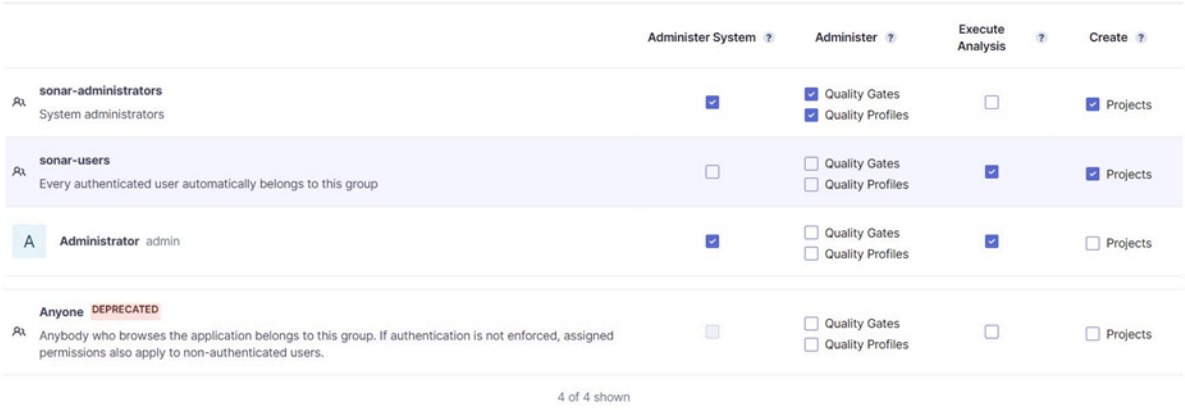


Console Output

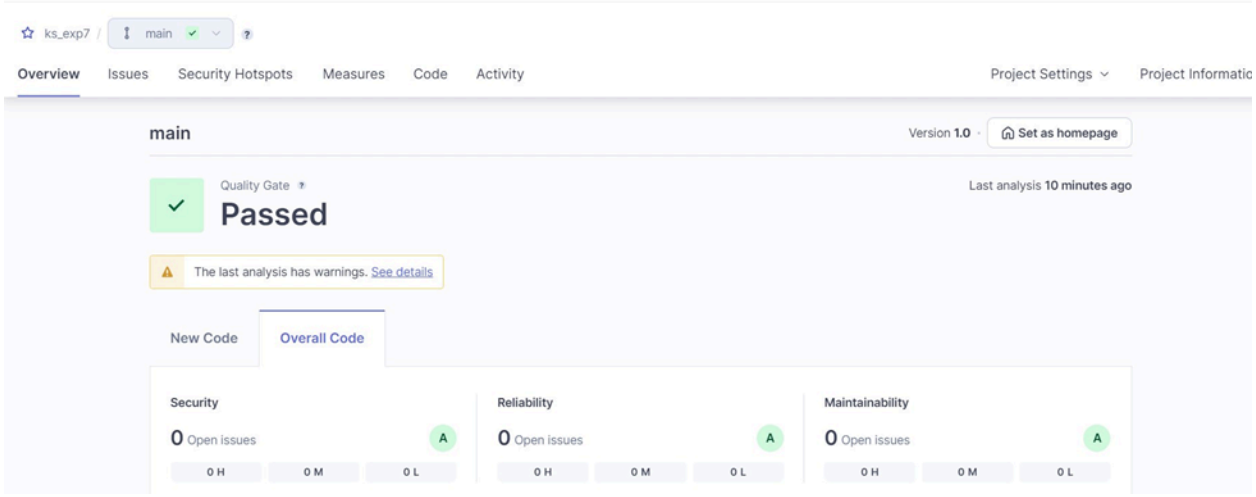
Download Copy View as plai

```
Started by user Shreyash Kamat
Running as SYSTEM
Building on the built-in node in workspace C:\ProgramData\Jenkins\.jenkins\workspace\ks_exp7
The recommended git tool is: NONE
No credentials specified
 > git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\ks_exp7\.git # timeout=10
Fetching changes from the remote Git repository
 > git.exe config remote.origin.url https://github.com/shazforiot/MSBuild_firstproject.git # timeout=10
Fetching upstream changes from https://github.com/shazforiot/MSBuild_firstproject.git
 > git.exe --version # timeout=10
 > git --version # 'git version 2.46.0.windows.1'
 > git.exe fetch --tags --force --progress -- https://github.com/shazforiot/MSBuild_firstproject.git +refs/heads/*:refs/remotes/origin/* # timeou
 > git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision f2bc042c04c6e72427c380bcaee6d6fee7b49adf (refs/remotes/origin/master)
 > git.exe config core.sparsecheckout # timeout=10
 > git.exe checkout -f f2bc042c04c6e72427c380bcaee6d6fee7b49adf # timeout=10
Commit message: "updated"
 > git.exe rev-list --no-walk f2bc042c04c6e72427c380bcaee6d6fee7b49adf # timeout=10
[ks_exp7] $ C:\ProgramData\Jenkins\.jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\sonarqube1_exp7\bin\sonar-scanner.bat -
Dsonar.host.url=http://localhost:9000 -Dsonar.projectKey=ks_exp7 -Dsonar.projectName=ks_exp7 -Dsonar.host.url=http://localhost:9000 -
Dsonar.login=admin -Dsonar.projectVersion=1.0 -Dsonar.sources=C:/ProgramData/Jenkins/.jenkins/workspace/ks_exp7 -Dsonar.password=kshitij24 -
Dsonar.projectBaseDir=C:\ProgramData\Jenkins\.jenkins\workspace\ks_exp7
15:09:08.473 WARN  Property 'sonar.host.url' with value 'http://localhost:9000' is overridden with value 'http://localhost:9000'
```

11. Go to http://localhost:9000/<user_name>/permissions and allow Execute Permissions to the Admin user.



12. Once the build is complete, check project on SonarQube



In this way, we have integrated Jenkins with SonarQube for SAST.

Conclusion: In this project, we successfully integrated Jenkins with SonarQube to implement automated static application security testing (SAST). We initiated the process by deploying SonarQube via Docker, followed by configuring Jenkins with the required plugins and authentication methods. Next, we connected Jenkins to a GitHub repository and incorporated the SonarQube scanner as a build step. This setup allows for continuous code analysis, identifying vulnerabilities, code smells, and quality issues, thus ensuring automated reporting and ongoing enhancements in code quality.

Name:Shreyash Kamat **Div:D15C Roll No:22**

Exp 8: Create a Jenkins CICD 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.

Theory:

What is SAST?

Static application security testing (SAST), or static analysis, is a testing methodology that analyzes source code to find security vulnerabilities that make your organization's applications susceptible to attack. SAST scans an application before the code is compiled. It's also known as white box testing.

What problems does SAST solve?

SAST takes place very early in the software development life cycle (SDLC) as it does not require a working application and can take place without code being executed. It helps developers identify vulnerabilities in the initial stages of development and quickly resolve issues without breaking builds or passing on vulnerabilities to the final release of the application.

SAST tools give developers real-time feedback as they code, helping them fix issues before they pass the code to the next phase of the SDLC. This prevents security-related issues from being considered an afterthought. SAST tools also provide graphical representations of the issues found, from source to sink. These help you navigate the code easier. Some tools point out the exact location of vulnerabilities and highlight the risky code. Tools can also provide in-depth guidance on how to fix issues and the best place in the code to fix them, without requiring deep security domain expertise.

It's important to note that SAST tools must be run on the application on a regular basis, such as during daily/monthly builds, every time code is checked in, or during a code release.

Why is SAST important?

Developers dramatically outnumber security staff. It can be challenging for an organization to find the resources to perform code reviews on even a fraction of its applications. A key strength of SAST tools is the ability to analyze 100% of the codebase. Additionally, they are much faster than manual secure code reviews performed by humans. These tools can scan millions of lines

of code in a matter of minutes. SAST tools automatically identify critical vulnerabilities—such as buffer overflows, SQL injection, cross-site scripting, and others—with high confidence.

What is a CI/CD Pipeline?

CI/CD pipeline refers to the Continuous Integration/Continuous Delivery pipeline. Before we dive deep into this segment, let's first understand what is meant by the term ‘pipeline’?

A pipeline is a concept that introduces a series of events or tasks that are connected in a sequence to make quick software releases. For example, there is a task, that task has got five different stages, and each stage has got some steps. All the steps in phase one have to be completed, to mark the latter stage to be complete.

Now, consider the CI/CD pipeline as the backbone of the DevOps approach. This Pipeline is responsible for building codes, running tests, and deploying new software versions. The Pipeline executes the job in a defined manner by first coding it and then structuring it inside several blocks that may include several steps or tasks.

What is SonarQube?

SonarQube is an open-source platform developed by SonarSource for continuous inspection of code quality. Sonar does static code analysis, which provides a detailed report of bugs, code smells, vulnerabilities, code duplications.
It supports 25+ major programming languages through built-in rulesets and can also be extended with various plugins.

Benefits of SonarQube

- **Sustainability -** Reduces complexity, possible vulnerabilities, and code duplications, optimising the life of applications.
- **Increase productivity -** Reduces the scale, cost of maintenance, and risk of the application; as such, it removes the need to spend more time changing the code
- **Quality code -** Code quality control is an inseparable part of the process of software development.
- **Detect Errors -** Detects errors in the code and alerts developers to fix them automatically before submitting them for output.
- **Increase consistency -** Determines where the code criteria are breached and enhances the quality
- **Business scaling -** No restriction on the number of projects to be evaluated
- **Enhance developer skills -** Regular feedback on quality problems helps developers to improve their coding skills

Integrating Jenkins with SonarQube:

Prerequisites:

- Jenkins installed
- Docker Installed (for SonarQube)
- SonarQube Docker Image

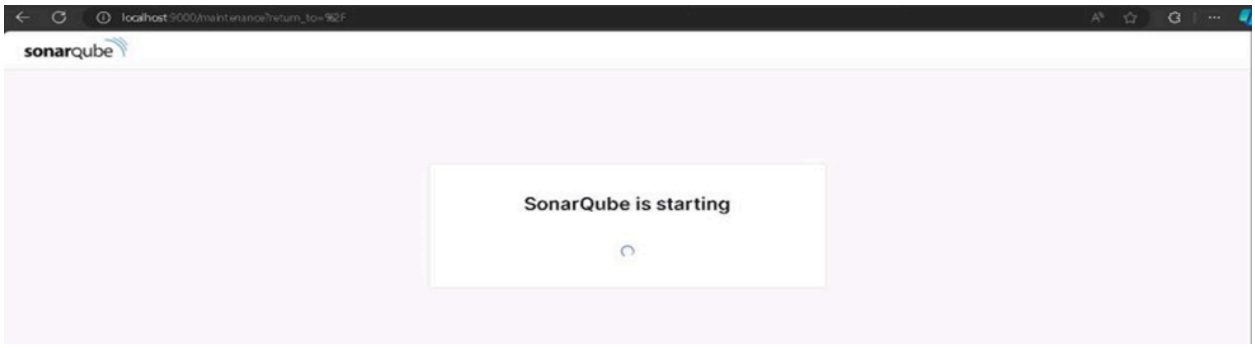
Steps to create a Jenkins CI/CD Pipeline and use SonarQube to perform SAST

1. Open up Jenkins Dashboard on localhost, port 8080 or whichever port it is at for you.

2. Run SonarQube in a Docker container using this command -

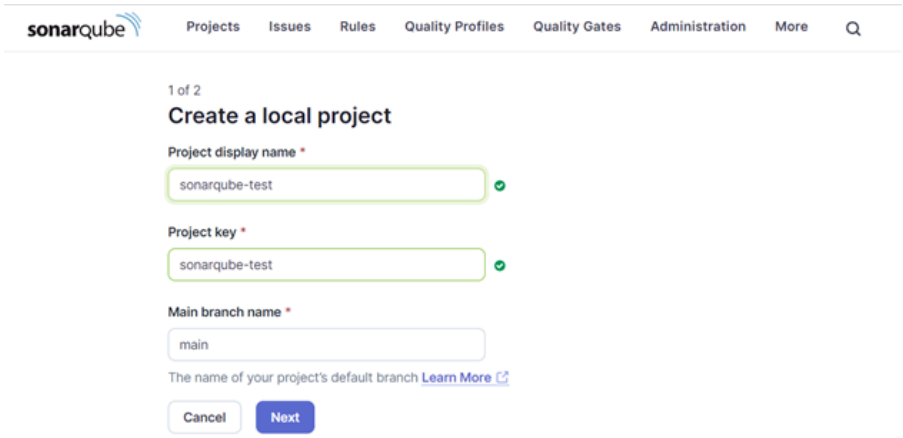
```
PS C:\Users\91773\Desktop\College Resources\Advdevops Exp8> docker run -d --name sonarqube2 -e SONAR_ES_BOOTSTRAP_CHECKS
_DISABLE=true -p 9000:9000 sonarqube:latest
71fc67f0b15baa5be5bdcdd66966938e18682683d020beadcbc909dd027cfe7a
PS C:\Users\91773\Desktop\College Resources\Advdevops Exp8>
```

3. Once the container is up and running, you can check the status of SonarQube at localhost port 9000.

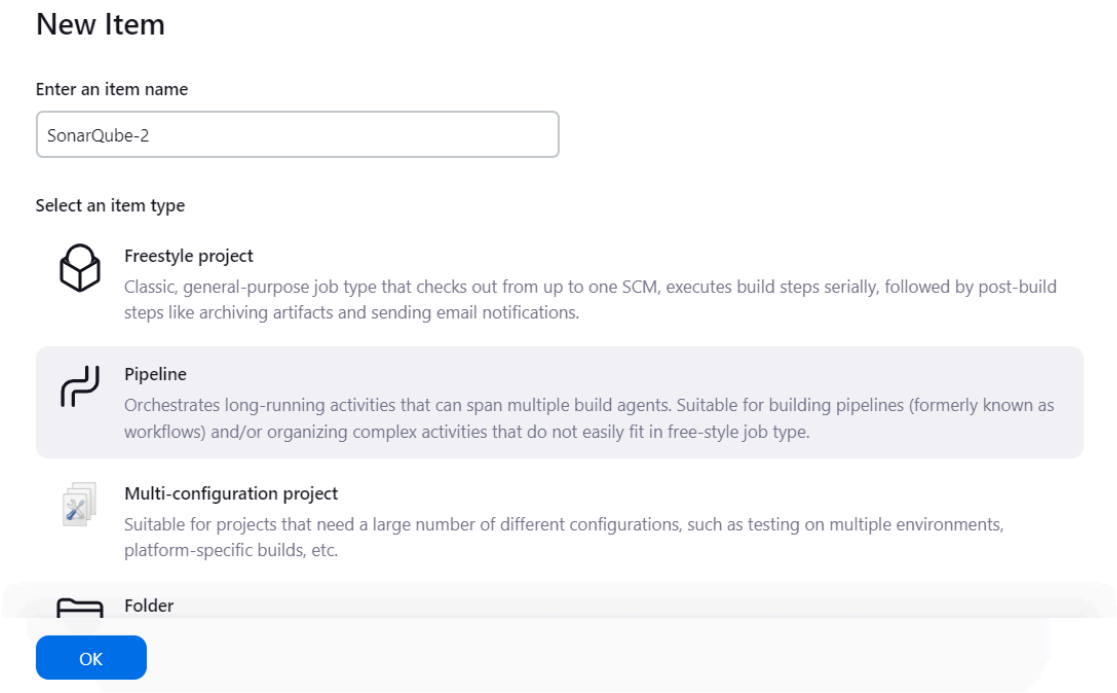


4. Login to SonarQube using username *admin* and password *admin*.

5. Create a manual project in SonarQube with the name **sonarqube-test** Setup the project and come back to Jenkins Dashboard.

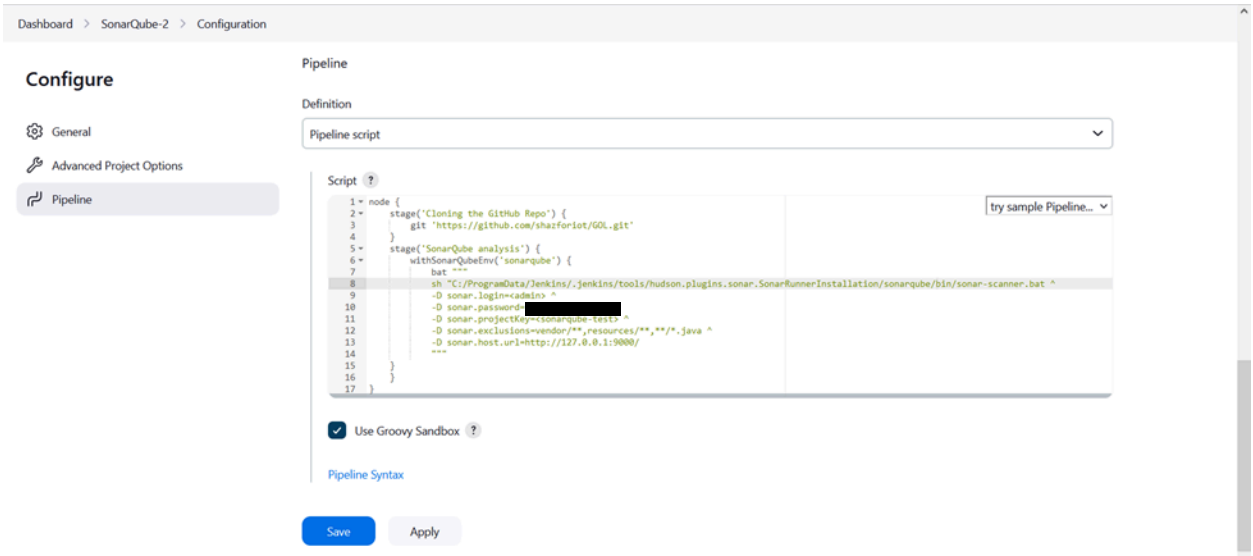


6. Create a New Item in Jenkins, choose **Pipeline**.



7. Under Pipeline Script, enter the following -

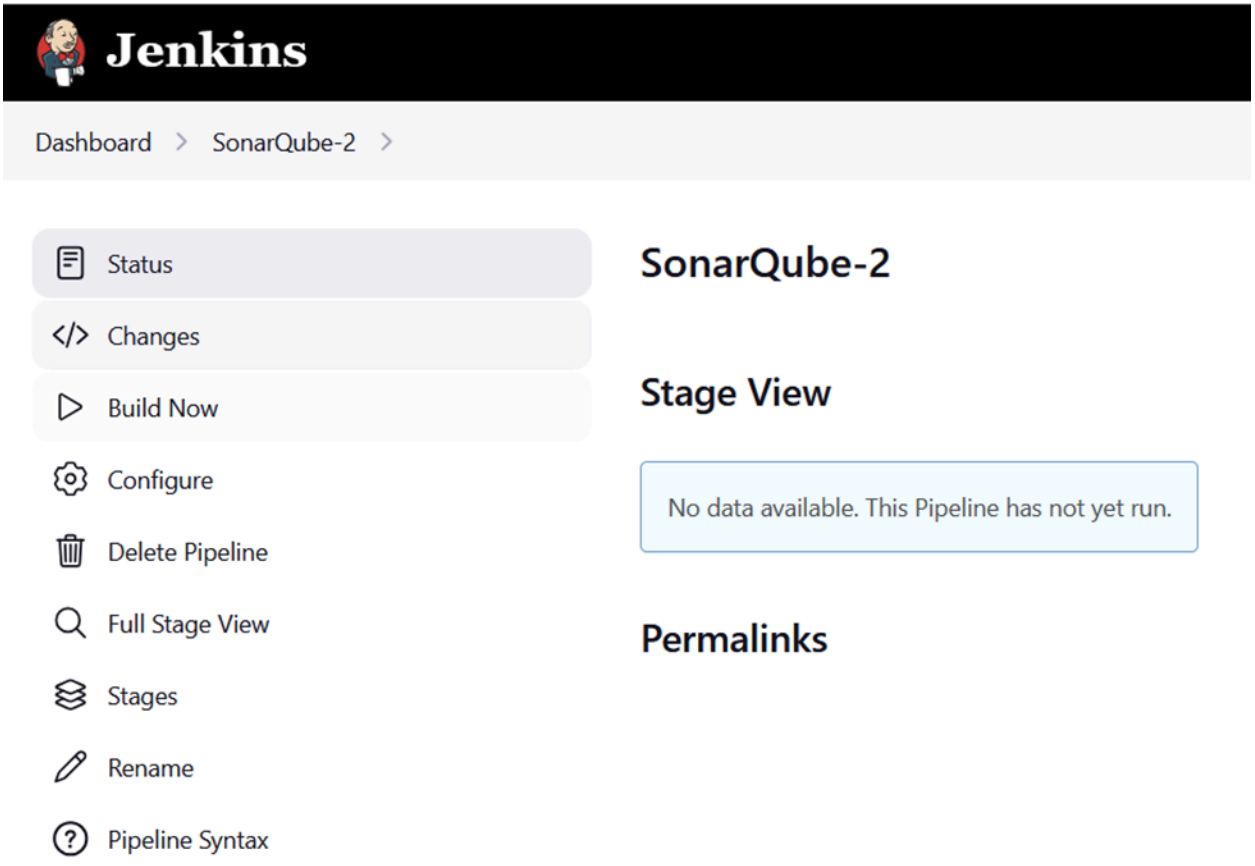
```
node {
    stage('Cloning the GitHub Repo') {
        git 'https://github.com/shazforiot/GOL.git'
    } stage('SonarQube analysis') {
        withSonarQubeEnv('sonarqube') {
            sh "<PATH_TO_SONARQUBE_FOLDER>//bin//sonar-scanner \
            -D sonar.login=<SonarQube_USERNAME> \
            -D sonar.password=<SonarQube_PASSWORD> \
            -D sonar.projectKey=<Project_KEY> \
-D sonar.exclusions=vendor/**,resources/**,**/*.java \
            -D sonar.host.url=http://127.0.0.1:9000/" }
    }
}
```



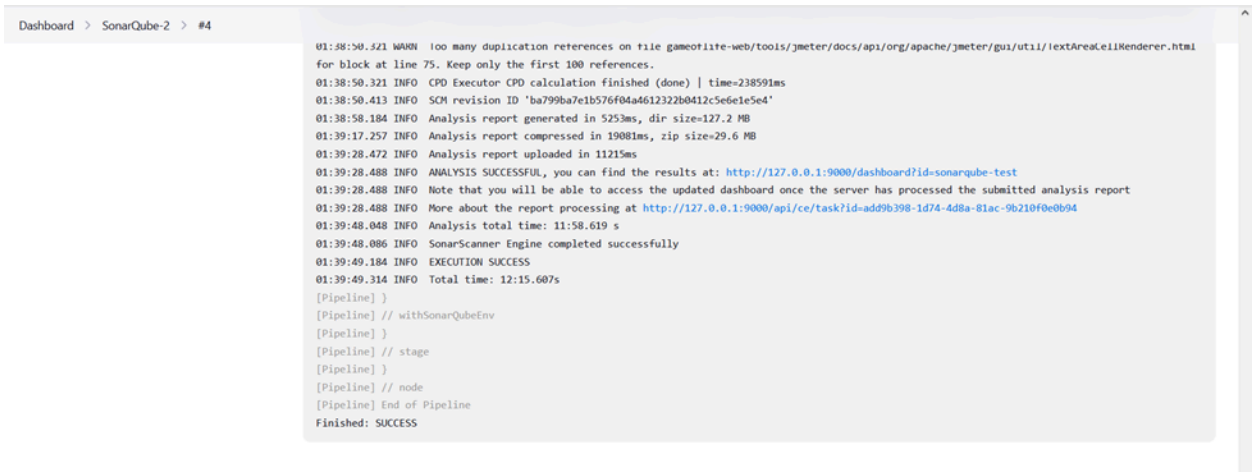
It is a java sample project which has a lot of repetitions and issues that will be detected by SonarQube.

8. Run The Build.

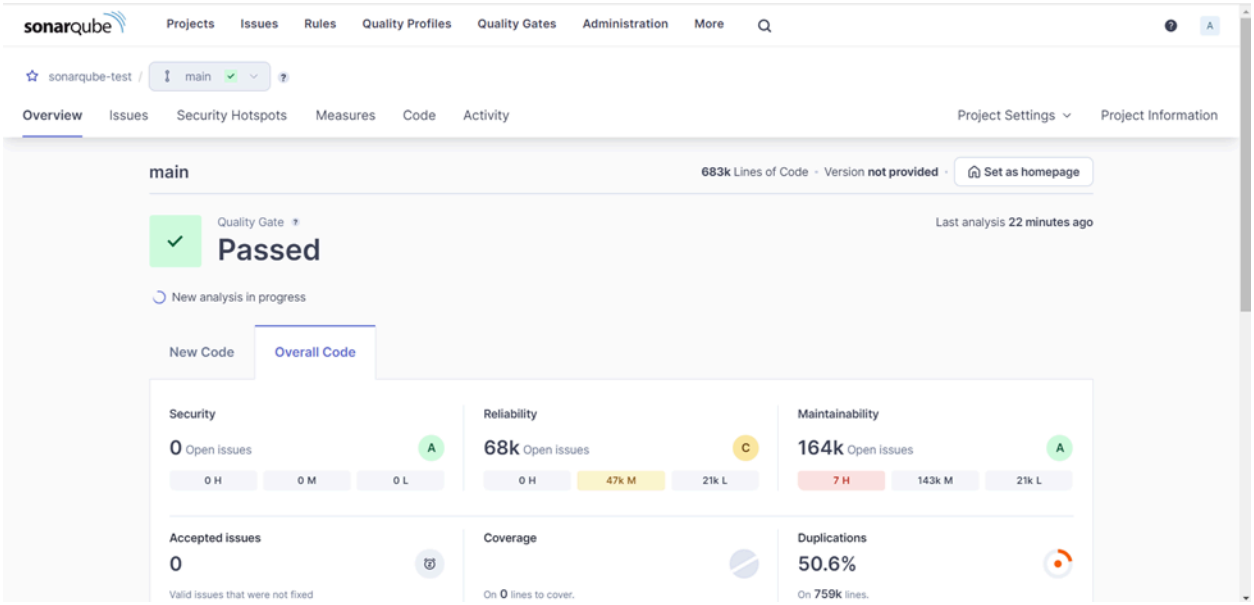
9. Check the console output once the build is complete



.

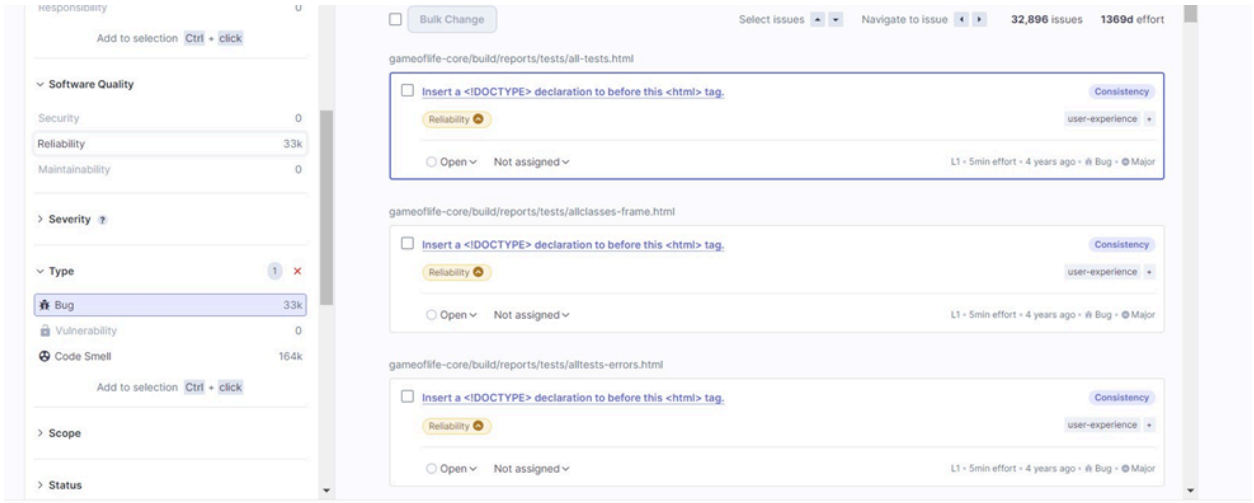


10. After that, check the project in SonarQube.

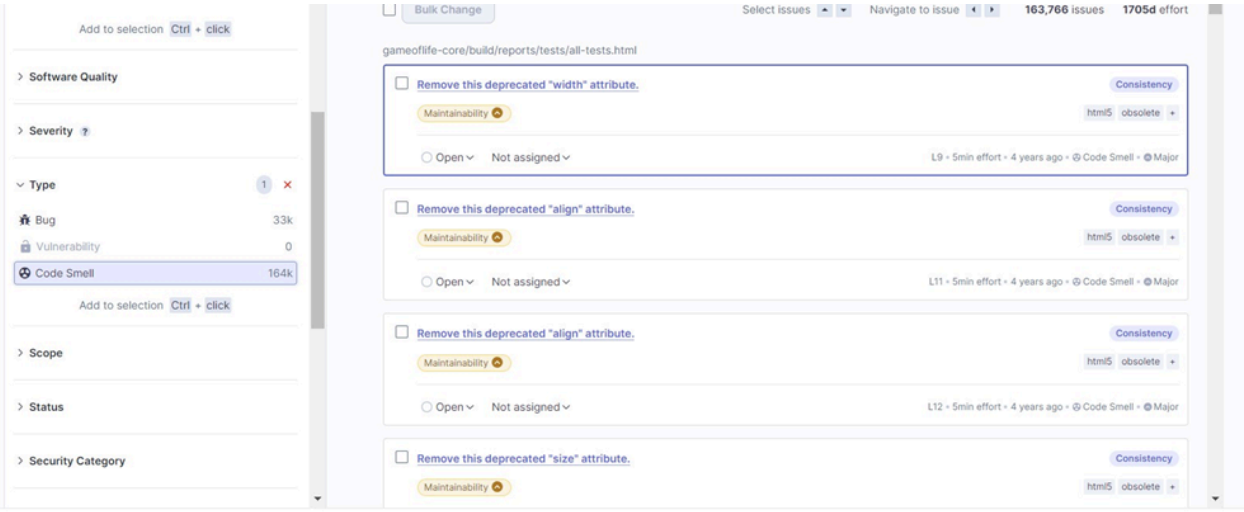


Under different tabs, check all different issues with the code.

11. **Bugs**

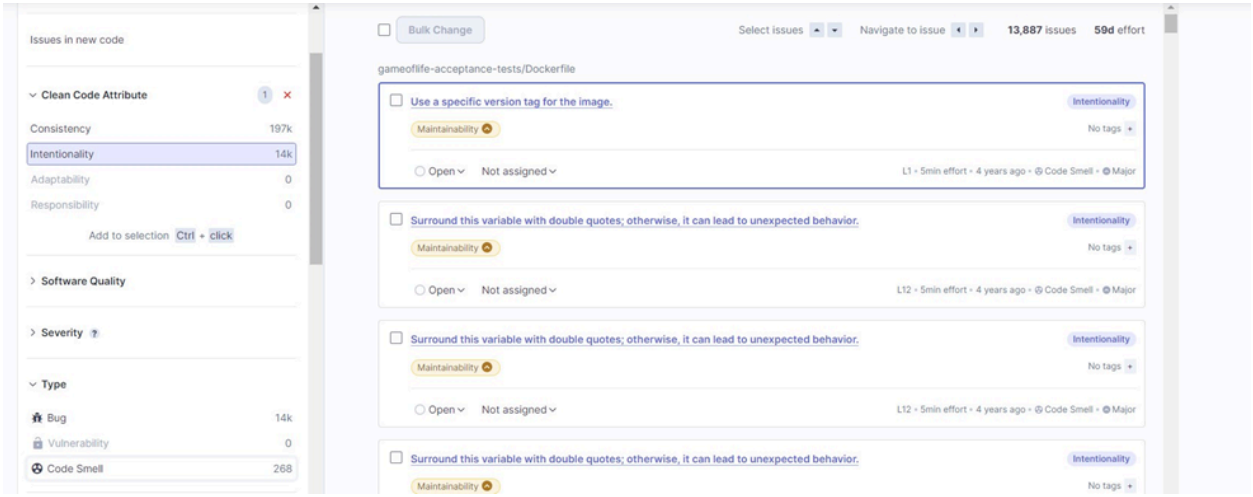


Code Smells

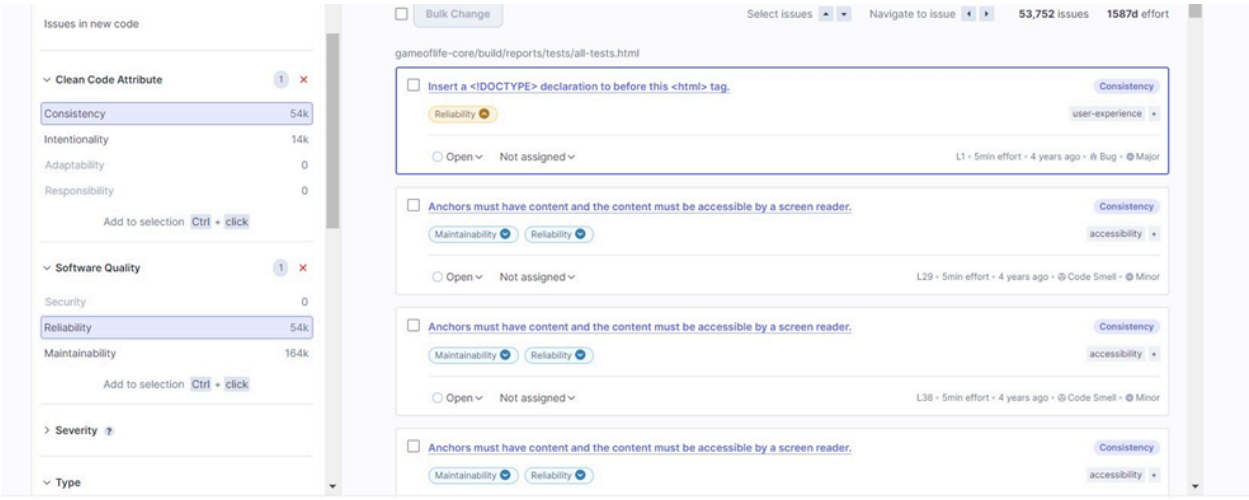


database should be used for evaluation purposes only

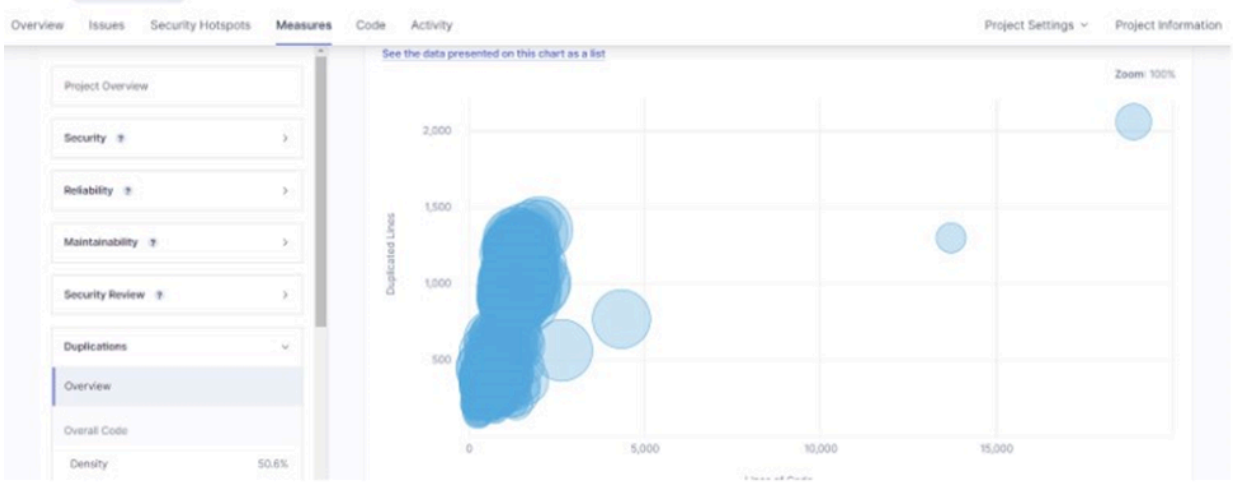
Intentional issues



Reliabilities issue



Duplicates



In this way, we have created a CI/CD Pipeline with Jenkins and integrated it with SonarQube to find issues in the code like bugs, code smells, duplicates, cyclomatic complexities, etc.

Conclusion:

In this experiment, we successfully integrated Jenkins with SonarQube to automate code quality checks within our CI/CD pipeline. We began by deploying SonarQube using Docker, configuring a project, and setting it up for comprehensive code analysis. Following this, we configured Jenkins by installing the SonarQube Scanner plugin, providing the necessary SonarQube server details, and establishing the scanner tool. We then created a Jenkins pipeline to streamline the process of cloning a GitHub repository and executing SonarQube analysis on the code. This integration facilitates continuous monitoring of code quality, enabling us to identify issues such as bugs, code smells, and security vulnerabilities throughout the development lifecycle.

Name: Shreyash Kamat Div: D15C Roll No: 22

Adv DevOps Exp 09

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

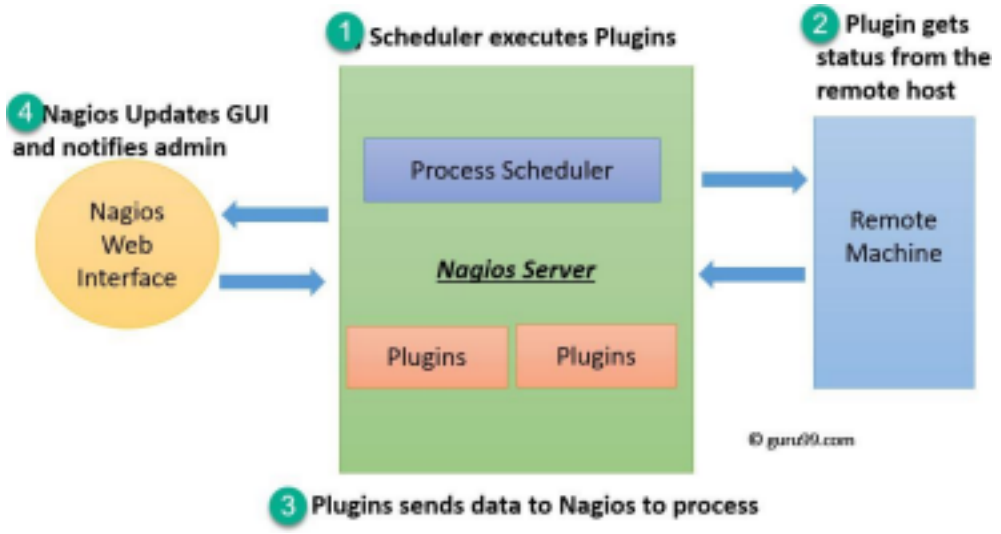
Following are the important features of Nagios monitoring tool:

- 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
- Utilizes 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

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.

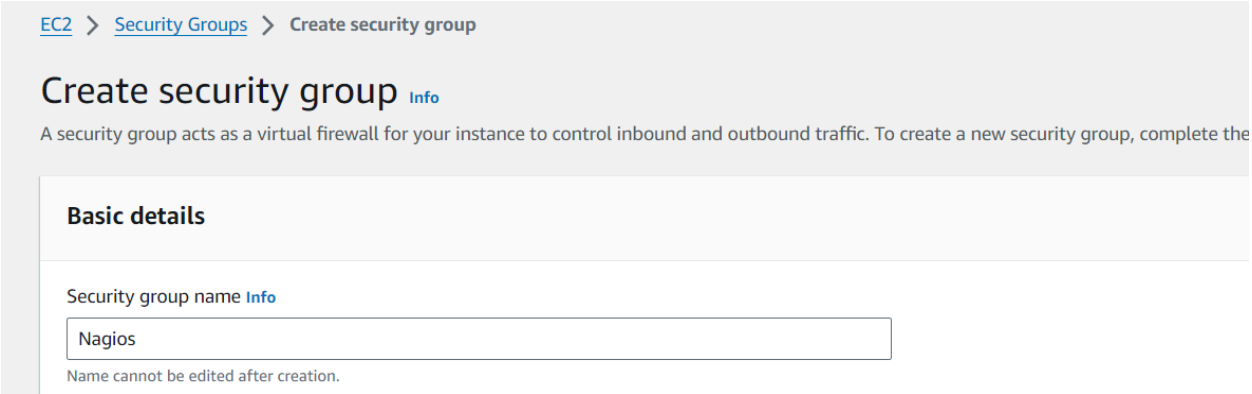


1. The scheduler is a component of the server part of Nagios. It sends a signal to execute the plugins at the remote host.
2. The plugin gets the status from the remote host
3. The plugin sends the data to the process scheduler
4. The process scheduler updates the GUI and notifications are sent to admins.

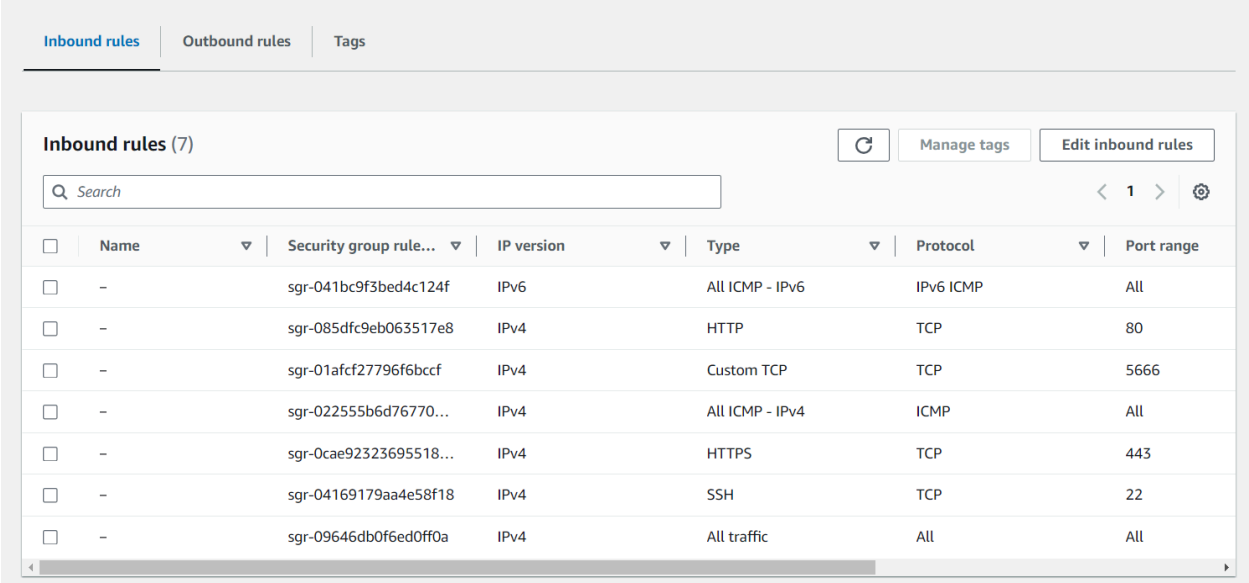
Name: Shreyash Kamat Div: D15C Roll No: 22

Step 1: Create a security group with the required configurations

I have created a new security group with a name ‘newsecurity’



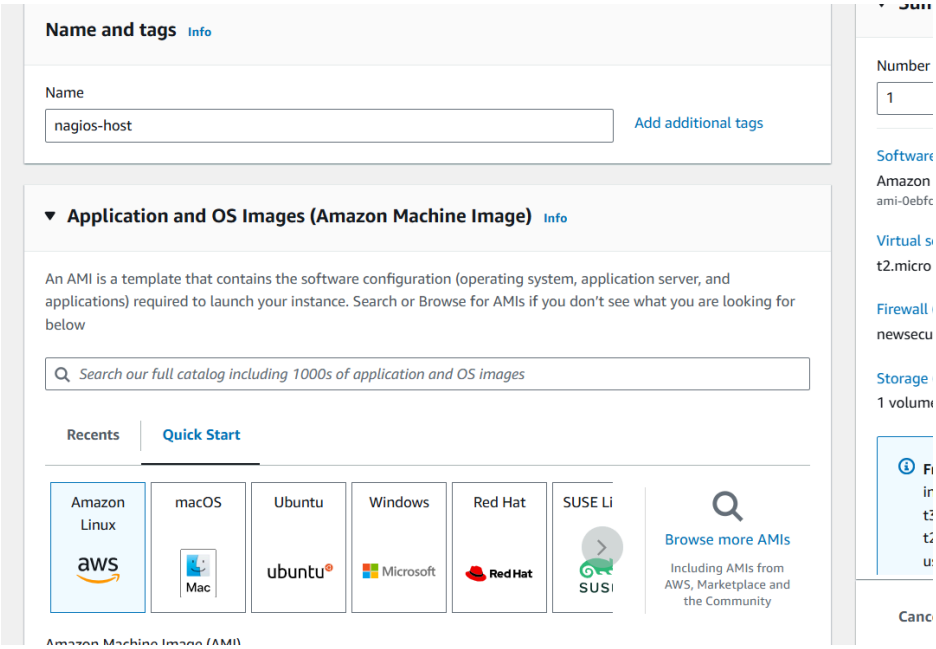
I have modified the INBOUND RULES as follows

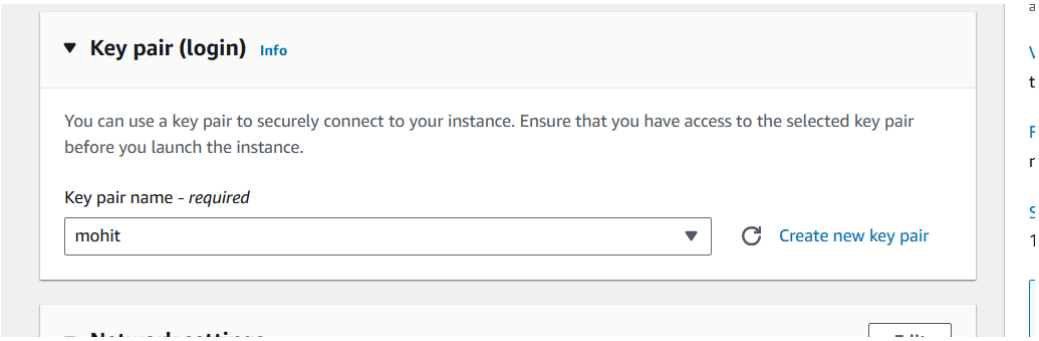


Step 2: Create ec2 instance

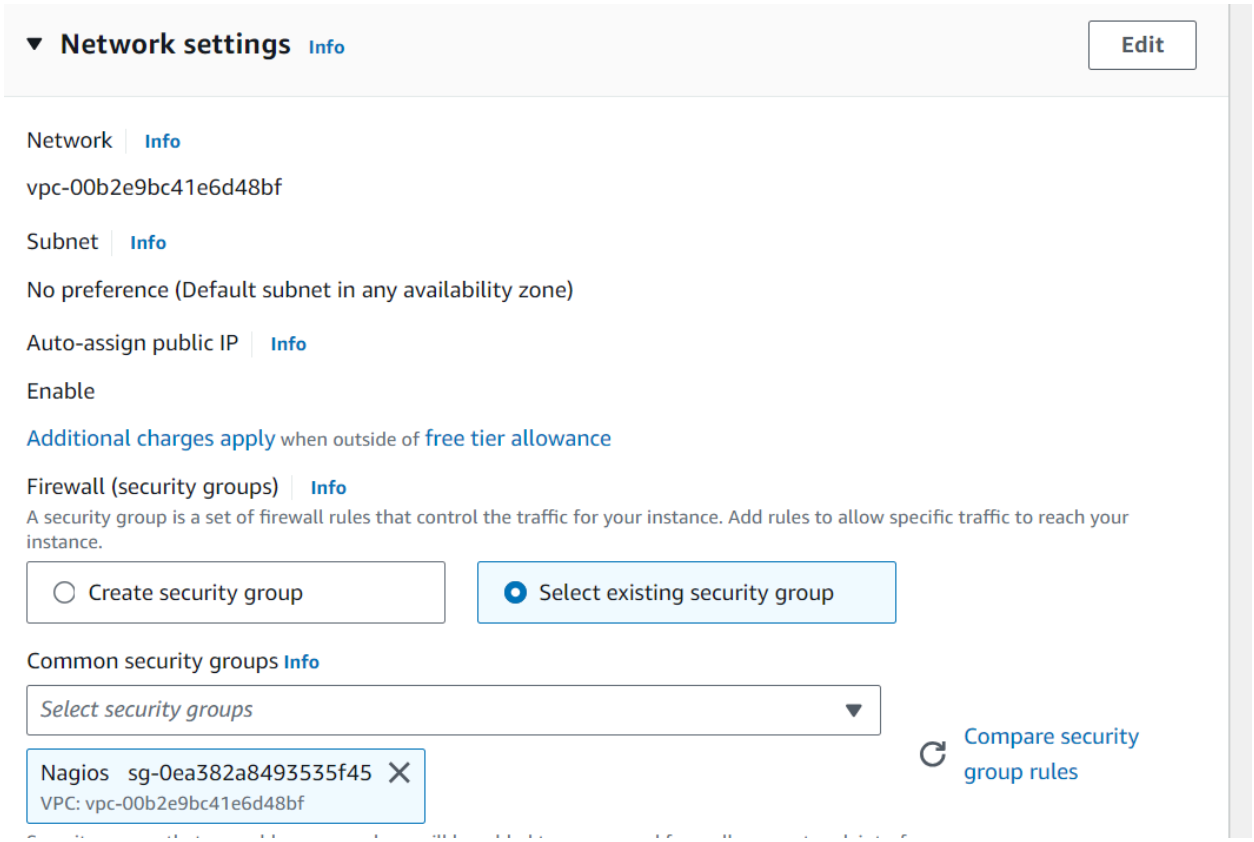
Name it as nagios-host. Select instance type as amazon-linux and choose the already created key pair and security group

Name: Shreyash Kamat Div: D15C Roll No: 22

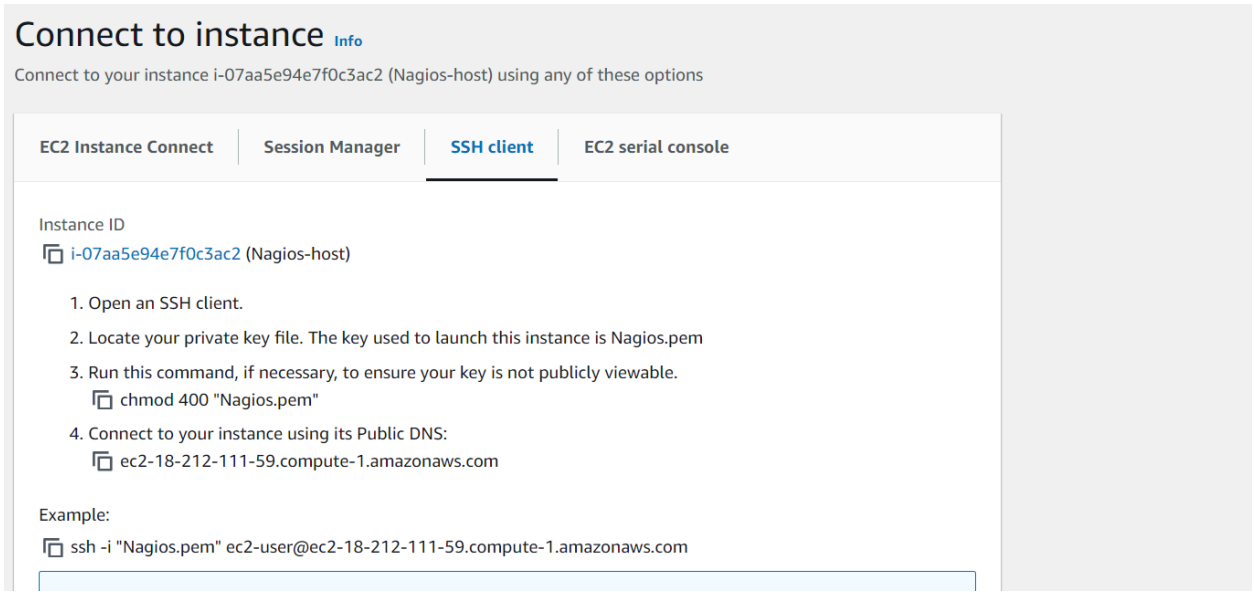




Name: Shreyash Kamat Div: D15C Roll No: 22

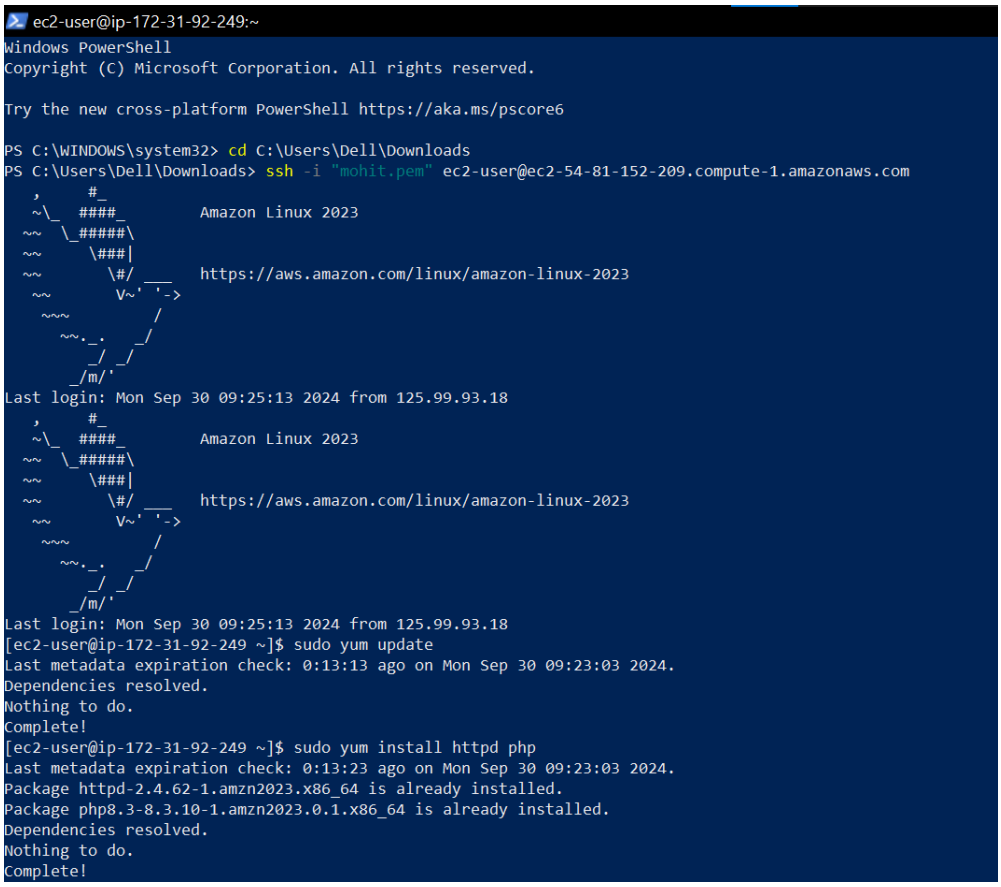


Copy the given ssh command, as we will require it for logging into our nagios-host instance from our windows powershell



Name: Shreyash Kamat Div: D15C Roll No: 22

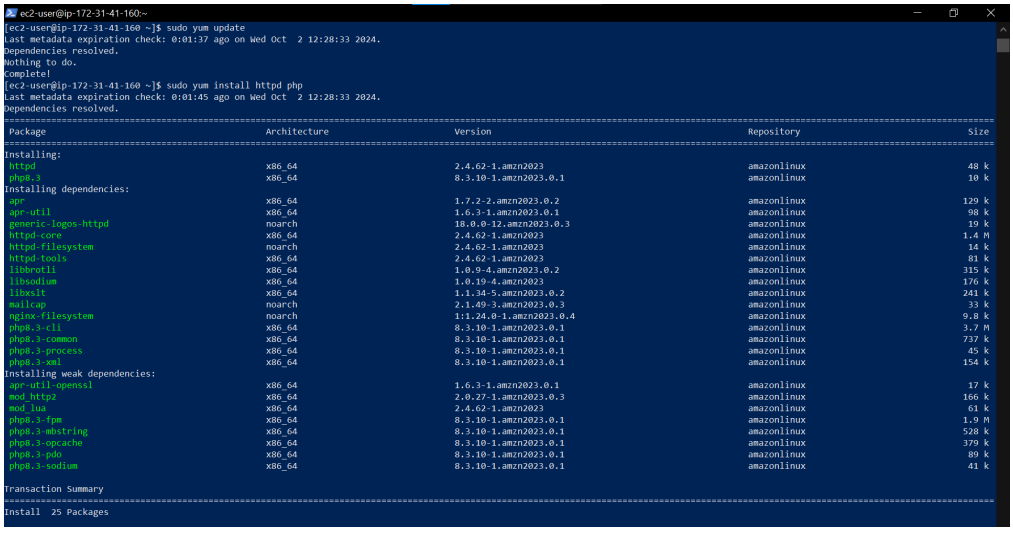
Step 3: Open an administrative powershell and remotely login using the above mentioned ssh command



And then run these commands

sudo yum update

sudo yum install httpd php



sudo yum install gcc glibc glibc-common

Name: Shreyash Kamat Div: D15C Roll No: 22

```
ec2-user@ip-172-31-41-160:~
[ec2-user@ip-172-31-41-160 ~]$ sudo yum install gcc glibc glibc-common
Last metadata expiration check: 0:02:02 ago on Wed Oct  2 12:28:33 2024.
Package glibc-2.34-52.amzn2023.0.11.x86_64 is already installed.
Package glibc-common-2.34-52.amzn2023.0.11.x86_64 is already installed.
Dependencies resolved.
================================================================================
 Package                 Architecture            Version                 Repository              Size
================================================================================
Installing:
 gcc                     x86_64                  11.4.1-2.amzn2023.0.2   amazonlinux             32 M
Installing dependencies:
 annobin-docs            noarch                  10.93-1.amzn2023.0.1    amazonlinux             92 k
 annobin-plugin-gcc      x86_64                  10.93-1.amzn2023.0.1    amazonlinux            887 k
 cpp                     x86_64                  11.4.1-2.amzn2023.0.2   amazonlinux             10 M
 gc                      x86_64                  8.0.4-5.amzn2023.0.2    amazonlinux            105 k
 glibc-devel             x86_64                  2.34-52.amzn2023.0.11   amazonlinux             27 k
 glibc-headers-x86       noarch                  2.34-52.amzn2023.0.11   amazonlinux            427 k
 guile22                 x86_64                  2.2.7-2.amzn2023.0.3    amazonlinux            6.4 M
 kernel-headers          x86_64                  6.1.109-118.189.amzn2023 amazonlinux           1.4 M
 libmpc                  x86_64                  1.2.1-2.amzn2023.0.2    amazonlinux             62 k
 libtool-ltdl            x86_64                  2.4.7-1.amzn2023.0.3    amazonlinux             38 k
 libxcrypt-devel         x86_64                  4.4.33-7.amzn2023       amazonlinux             32 k
 make                    x86_64                  1:4.3-5.amzn2023.0.2    amazonlinux            534 k

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

Total download size: 52 M
Installed size: 168 M
Is this ok [y/N]: y
Downloading Packages:
(1/13): annobin-docs-10.93-1.amzn2023.0.1.noarch.rpm          852 kB/s |  92 kB     00:00
(2/13): annobin-plugin-gcc-10.93-1.amzn2023.0.1.x86_64.rpm    6.5 MB/s | 887 kB     00:00
(3/13): gc-8.0.4-5.amzn2023.0.2.x86_64.rpm                    2.3 MB/s | 105 kB     00:00
(4/13): glibc-devel-2.34-52.amzn2023.0.11.x86_64.rpm          1.1 MB/s |  27 kB     00:00
(5/13): cpp-11.4.1-2.amzn2023.0.2.x86_64.rpm                   32 MB/s |  10 MB     00:00
(6/13): glibc-headers-x86-2.34-52.amzn2023.0.11.noarch.rpm    2.9 MB/s | 427 kB     00:00
(7/13): kernel-headers-6.1.109-118.189.amzn2023.x86_64.rpm     16 MB/s | 1.4 MB     00:00
(8/13): libmpc-1.2.1-2.amzn2023.0.2.x86_64.rpm                2.1 MB/s |  62 kB     00:00
(9/13): guile22-2.2.7-2.amzn2023.0.3.x86_64.rpm                27 MB/s | 6.4 MB     00:00
(10/13): libtool-ltdl-2.4.7-1.amzn2023.0.3.x86_64.rpm         322 kB/s |  38 kB     00:00
(11/13): libxcrypt-devel-4.4.33-7.amzn2023.x86_64.rpm         1.4 MB/s |  32 kB     00:00
```

sudo yum install gd gd-devel

```
ec2-user@ip-172-31-41-160:~
[ec2-user@ip-172-31-41-160 ~]$ sudo yum install gd gd-devel
Last metadata expiration check: 0:02:25 ago on Wed Oct  2 12:28:33 2024.
Dependencies resolved.
================================================================================
 Package                 Architecture            Version                 Repository              Size
================================================================================
Installing:
 gd                      x86_64                  2.3.3-5.amzn2023.0.3    amazonlinux            139 k
 gd-devel                x86_64                  2.3.3-5.amzn2023.0.3    amazonlinux             38 k
Installing dependencies:
 brotli                  x86_64                  1.0.9-4.amzn2023.0.2    amazonlinux            314 k
 brotli-devel            x86_64                  1.0.9-4.amzn2023.0.2    amazonlinux             31 k
 bzip2-devel             x86_64                  1.0.8-6.amzn2023.0.2    amazonlinux            214 k
 cairo                   x86_64                  1.17.6-2.amzn2023.0.1   amazonlinux            684 k
 cmake-filesystem        x86_64                  3.22.2-1.amzn2023.0.4   amazonlinux             16 k
 fontconfig              x86_64                  2.13.94-2.amzn2023.0.2  amazonlinux            273 k
 fontconfig-devel        x86_64                  2.13.94-2.amzn2023.0.2  amazonlinux            128 k
 fonts-filesystem        noarch                  1:2.0.5-12.amzn2023.0.2 amazonlinux            9.5 k
 freetype                x86_64                  2.13.2-5.amzn2023.0.1   amazonlinux            423 k
 freetype-devel          x86_64                  2.13.2-5.amzn2023.0.1   amazonlinux            912 k
 glib2-devel             x86_64                  2.74.7-689.amzn2023.0.2 amazonlinux            486 k
 google-noto-fonts-common noarch                 20201206-2.amzn2023.0.2 amazonlinux             15 k
 google-noto-sans-vf-fonts noarch                20201206-2.amzn2023.0.2 amazonlinux            492 k
 graphite2               x86_64                  1.3.14-7.amzn2023.0.2   amazonlinux             97 k
 graphite2-devel         x86_64                  1.3.14-7.amzn2023.0.2   amazonlinux             21 k
 harfbuzz                x86_64                  7.0.0-2.amzn2023.0.1    amazonlinux            868 k
 harfbuzz-devel          x86_64                  7.0.0-2.amzn2023.0.1    amazonlinux            404 k
 harfbuzz-icu            x86_64                  7.0.0-2.amzn2023.0.1    amazonlinux             18 k
 jbigkit-libs            x86_64                  2.1-21.amzn2023.0.2     amazonlinux             54 k
 langpacks-core-font-en  noarch                  3.0-21.amzn2023.0.4     amazonlinux             10 k
 libICE                  x86_64                  1.0.10-6.amzn2023.0.2   amazonlinux             71 k
 libSM                   x86_64                  1.2.3-8.amzn2023.0.2    amazonlinux             42 k
 libX11                  x86_64                  1.7.2-3.amzn2023.0.4    amazonlinux            657 k
 libX11-common           noarch                  1.7.2-3.amzn2023.0.4    amazonlinux            152 k
 libX11-devel            x86_64                  1.7.2-3.amzn2023.0.4    amazonlinux            939 k
 libX11-xcb              x86_64                  1.7.2-3.amzn2023.0.4    amazonlinux             12 k
 libXau                  x86_64                  1.0.9-6.amzn2023.0.2    amazonlinux             31 k
 libXau-devel            x86_64                  1.0.9-6.amzn2023.0.2    amazonlinux             14 k
 libXext                 x86_64                  1.3.4-6.amzn2023.0.2    amazonlinux             41 k
 libXpm                  x86_64                  3.5.15-2.amzn2023.0.3   amazonlinux             65 k
 libXpm-devel            x86_64                  3.5.15-2.amzn2023.0.3   amazonlinux             59 k
 libXrender              x86_64                  0.9.10-14.amzn2023.0.2  amazonlinux             28 k
 libXt                   x86_64                  1.2.0-4.amzn2023.0.2    amazonlinux            181 k
 libblkid-devel          x86_64                  2.37.4-1.amzn2023.0.4   amazonlinux             15 k
```

Name: Shreyash Kamat Div: D15C Roll No: 22

Create a new Nagios User with its password. You'll have to enter the password twice for confirmation.

sudo adduser -m nagios
sudo passwd nagios

```
[ec2-user@ip-172-31-41-160 ~]$ sudo adduser -m nagios
[ec2-user@ip-172-31-41-160 ~]$ sudo passwd nagios
Changing password for user nagios.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[ec2-user@ip-172-31-41-160 ~]$
```

Create a new user group & create a new directory for Nagios downloads using the following commands

sudo groupadd nagcmd
sudo usermod -a -G nagcmd nagios
sudo usermod -a -G nagcmd apache
mkdir ~/downloads
cd ~/downloads

Use **wget** to download the source zip files.
In this step, we are downloading, the latest version of nagios and the necessary plugins required to carry out the tasks of setting up a nagios server

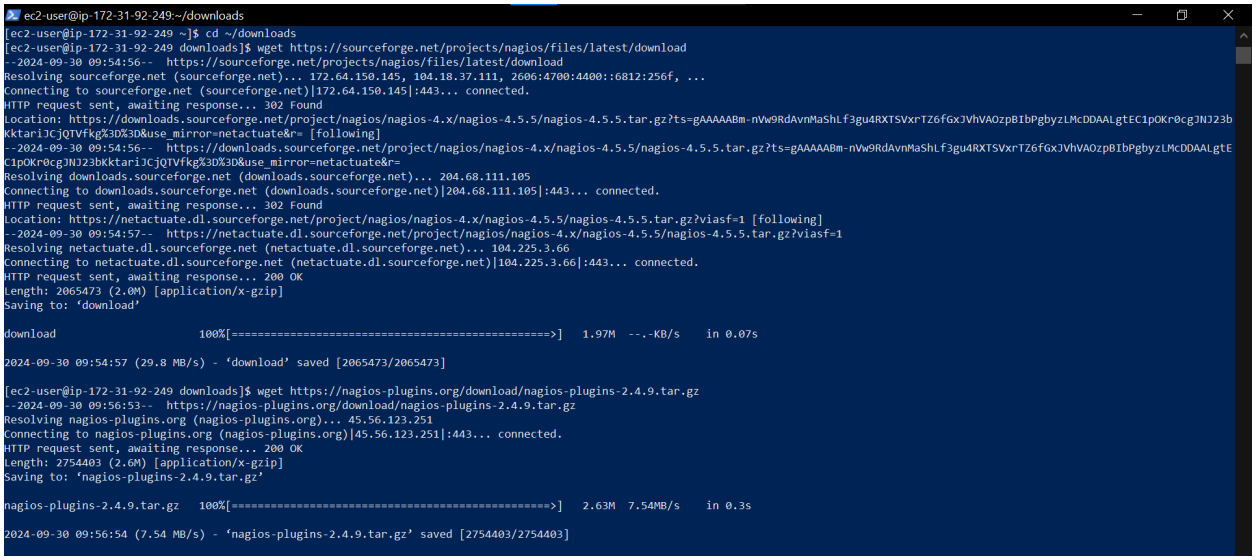
wget https://sourceforge.net/projects/nagios/files/latest/download

```
ec2-user@ip-172-31-41-160:~/downloads/nagios-4.5.5
[ec2-user@ip-172-31-41-160 downloads]$ wget https://sourceforge.net/projects/nagios/files/latest/download
--2024-10-02 12:34:21--  https://sourceforge.net/projects/nagios/files/latest/download
Resolving sourceforge.net (sourceforge.net)... 172.64.150.145, 104.18.37.111, 2606:4700:4400::6812:256f, ...
Connecting to sourceforge.net (sourceforge.net)|172.64.150.145|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://downloads.sourceforge.net/project/nagios/nagios-4.x/nagios-4.5.5/nagios-4.5.5.tar.gz?ts=gAAAAABm_T3NWNSZPzP-6la2TltvoOGCG7VvV7QGVHO8n3tC24QehfMw7vHcoKbGHg2iIRxbmfugII0LccN
fxEaOixg3j2kG3w%3D%3D&use_mirror=phoenixnap&r= [following]
--2024-10-02 12:34:21--  https://downloads.sourceforge.net/project/nagios/nagios-4.x/nagios-4.5.5/nagios-4.5.5.tar.gz?ts=gAAAAABm_T3NWNSZPzP-6la2TltvoOGCG7VvV7QGVHO8n3tC24QehfMw7vHcoKbGHg2i
IRxbmfugII0LccNfxEaOixg3j2kG3w%3D%3D&use_mirror=phoenixnap&r=
Resolving downloads.sourceforge.net (downloads.sourceforge.net)... 204.68.111.105
Connecting to downloads.sourceforge.net (downloads.sourceforge.net)|204.68.111.105|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://phoenixnap.dl.sourceforge.net/project/nagios/nagios-4.x/nagios-4.5.5/nagios-4.5.5.tar.gz?viasf=1 [following]
--2024-10-02 12:34:21--  https://phoenixnap.dl.sourceforge.net/project/nagios/nagios-4.x/nagios-4.5.5/nagios-4.5.5.tar.gz?viasf=1
Resolving phoenixnap.dl.sourceforge.net (phoenixnap.dl.sourceforge.net)... 184.164.141.26
Connecting to phoenixnap.dl.sourceforge.net (phoenixnap.dl.sourceforge.net)|184.164.141.26|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2065473 (2.0M) [application/x-gzip]
Saving to: 'download'

download                                    100%[=========================================================================================>]   1.97M  4.23MB/s    in 0.5s

2024-10-02 12:34:22 (4.23 MB/s) - 'download' saved [2065473/2065473]
```

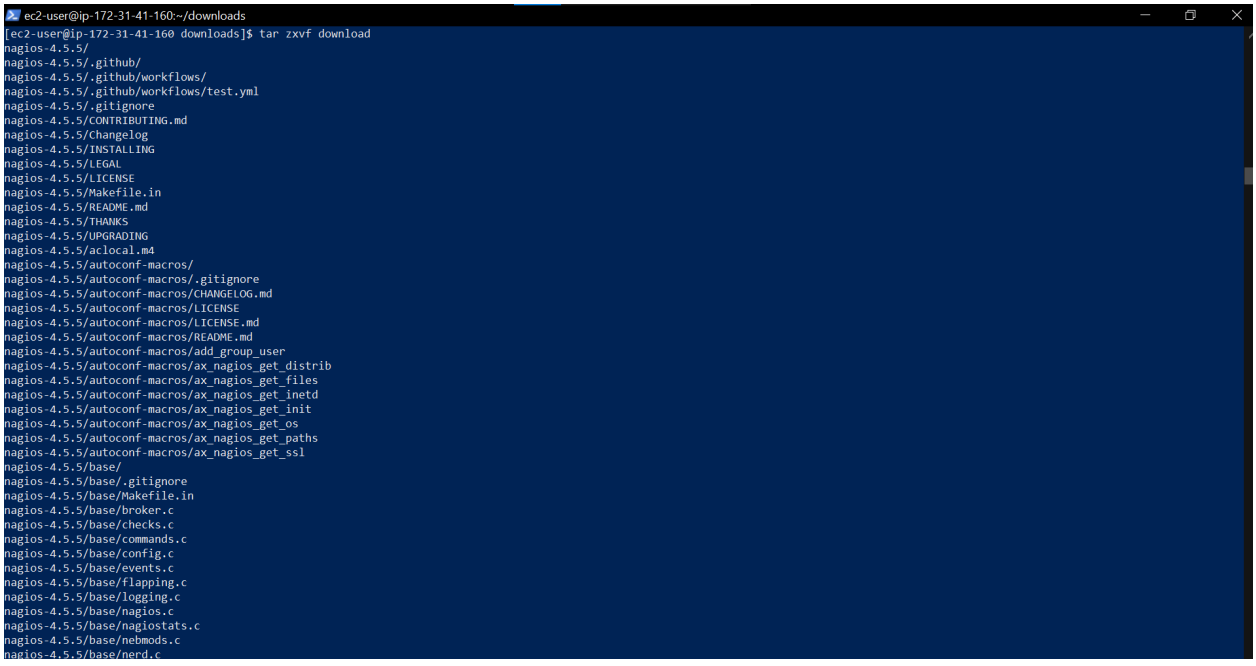
wget https://nagios-plugins.org/download/nagios-plugins-2.4.11.tar.gz

```
ec2-user@ip-172-31-41-160:~/downloads/nagios-4.5.5
[ec2-user@ip-172-31-41-160 downloads]$ wget https://nagios-plugins.org/download/nagios-plugins-2.4.11.tar.gz
--2024-10-02 12:34:46--  https://nagios-plugins.org/download/nagios-plugins-2.4.11.tar.gz
Resolving nagios-plugins.org (nagios-plugins.org)... 45.56.123.251
Connecting to nagios-plugins.org (nagios-plugins.org)|45.56.123.251|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2753049 (2.6M) [application/x-gzip]
Saving to: 'nagios-plugins-2.4.11.tar.gz'

nagios-plugins-2.4.11.tar.gz                100%[=========================================================================================>]   2.62M  7.48MB/s    in 0.4s

2024-10-02 12:34:46 (7.48 MB/s) - 'nagios-plugins-2.4.11.tar.gz' saved [2753049/2753049]
```

Name: Shreyash Kamat Div: D15C Roll No: 22



Now, we run the next command in the following manner
tar zxvf <nagios-4.5.5 version> (for me it has gotten saved as 'download')
So i wrote **tar zxvf download**



After which we are supposed to **change our directory** over there
For eg. **cd nagios-4.5.5**… depending on the version that we have downloaded

Next, Run this command (make sure that you are working inside nagios-4.x.x directory)
./configure --with-command-group=nagcmd

Name: Shreyash Kamat Div: D15C Roll No: 22

```
ec2-user@ip-172-31-41-160:~/downloads/nagios-4.5.5
[ec2-user@ip-172-31-41-160 downloads]$  cd nagios-4.5.5
[ec2-user@ip-172-31-41-160 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
checking for errno.h... yes
checking for fcntl.h... yes
checking for getopt.h... yes
checking for grp.h... yes
checking for libgen.h... yes
checking for limits.h... yes
checking for math.h... yes
checking for netdb.h... yes
checking for netinet/in.h... yes
checking for pwd.h... yes
checking for regex.h... yes
checking for signal.h... yes
checking for socket.h... no
checking for stdarg.h... yes
```

```
checking for strtoul... yes
checking for unsetenv... yes
checking for type of socket size... size_t
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
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$
```

After running this command, we get an **error related to ssl header being absent**

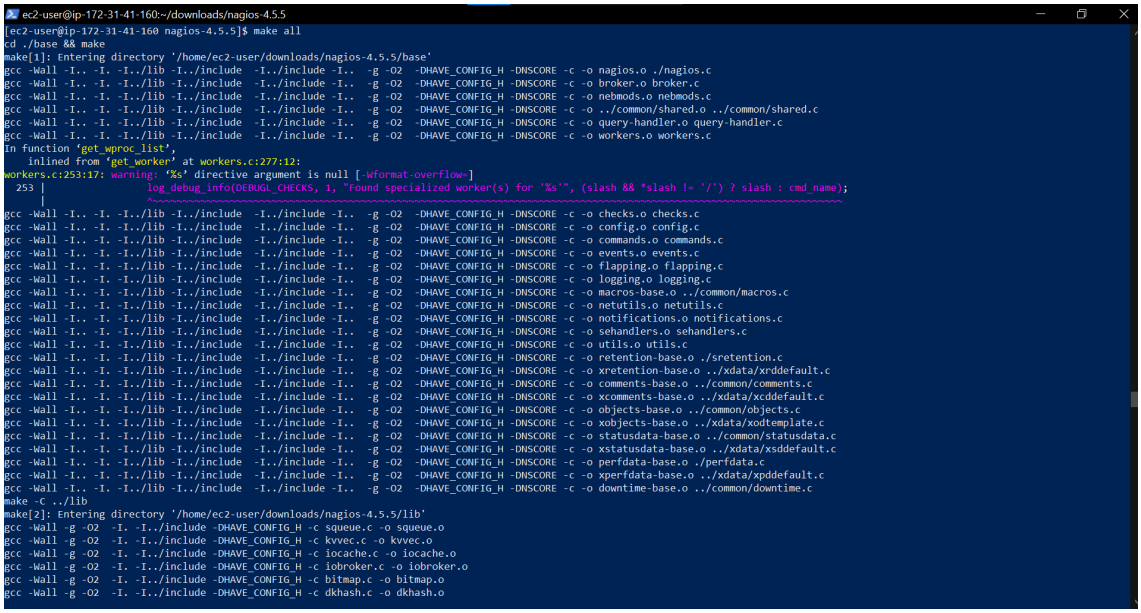
For that purpose, we are to run the following command.
sudo yum install openssl-devel (for ssl header)

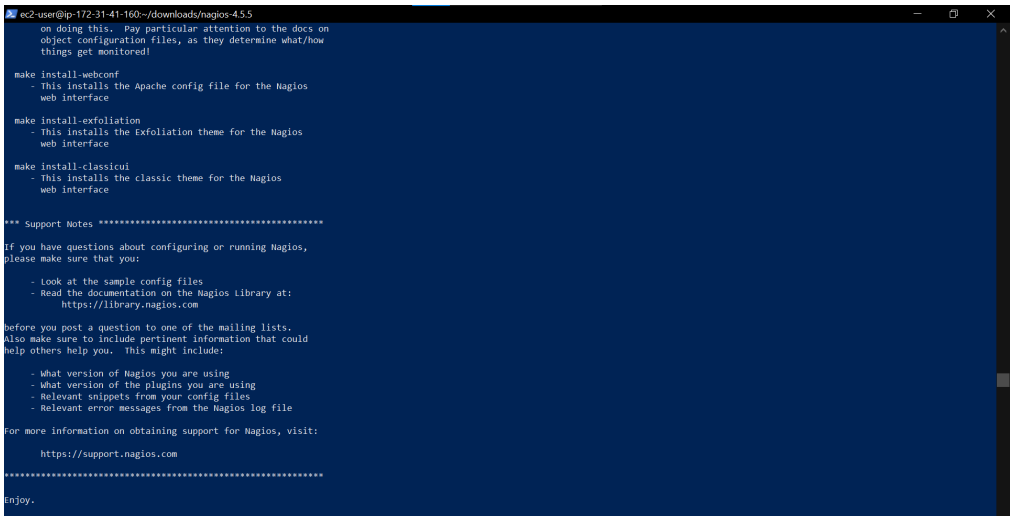
```
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ sudo yum install openssl-devel
Last metadata expiration check: 0:12:11 ago on Wed Oct  2 12:28:33 2024.
Dependencies resolved.
================================================================================
 Package                 Architecture        Version                  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   26 MB/s | 3.0 MB     00:00
--------------------------------------------------------------------------------
Total                                            17 MB/s | 3.0 MB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                        1/1
  Installing       : openssl-devel-1:3.0.8-1.amzn2023.0.14.x86_64           1/1
  Running scriptlet: openssl-devel-1:3.0.8-1.amzn2023.0.14.x86_64           1/1
  Verifying        : openssl-devel-1:3.0.8-1.amzn2023.0.14.x86_64           1/1

Installed:
  openssl-devel-1:3.0.8-1.amzn2023.0.14.x86_64

Complete!
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$
```

Now, Re-run **./configure --with-command-group=nagcmd**

Name: Shreyash Kamat Div: D15C Roll No: 22

After this, run **make all** command





Run the following set of commands to ensure that
sudo make install

Name: Shreyash Kamat Div: D15C Roll No: 22

```
ec2-user@ip-172-31-41-160:~/downloads/nagios-4.5.5
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ sudo make install
cd ./base && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/base'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagios /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagiostats /usr/local/nagios/bin
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5/base'
cd ./cgi && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
make install-basic
make[2]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/sbin
for file in *.cgi; do \
        /usr/bin/install -c -s -m 775 -o nagios -g nagios $file /usr/local/nagios/sbin; \
done
make[2]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
cd ./html && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/html'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/media
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/stylesheets
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/contexthelp
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/docs
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/docs/images
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/js
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/images
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/images/logos
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/includes
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/ssi
/usr/bin/install -c -m 664 -o nagios -g nagios ./robots.txt /usr/local/nagios/share
/usr/bin/install -c -m 664 -o nagios -g nagios ./jsonquery.html /usr/local/nagios/share
rm -f /usr/local/nagios/share/index.html
rm -f /usr/local/nagios/share/main.html
rm -f /usr/local/nagios/share/side.html
rm -f /usr/local/nagios/share/map.html
rm -f /usr/local/nagios/share/rss-*
rm -f /usr/local/nagios/share/graph-header.html
rm -f /usr/local/nagios/share/histogram.html
rm -f /usr/local/nagios/share/histogram-form.html
rm -f /usr/local/nagios/share/histogram-graph.html
rm -f /usr/local/nagios/share/histogram-links.html
rm -f /usr/local/nagios/share/infobox.html
rm -f /usr/local/nagios/share/map.php
```

sudo make install-init

```
ec2-user@ip-172-31-41-160:~/downloads/nagios-4.5.5
[ec2-user@ip-172-31-41-160 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
```

sudo make install-config

```
[ec2-user@ip-172-31-41-160 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.
```

sudo make install-webconf

```
[ec2-user@ip-172-31-41-160 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 ***

[ec2-user@ip-172-31-41-160 nagios-4.5.5]$
```

Next, we are supposed to create a nagiosadmin account for nagios login along with password. Specify the password twice.

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

```
[ec2-user@ip-172-31-41-160 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
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$
```

Restart Apache

sudo service httpd restart

Go back to the downloads folder and unzip the plugins zip file.

cd ~/downloads

tar zxvf nagios-plugins-2.4.11.tar.gz

```
ec2-user@ip-172-31-41-160:~/downloads
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ sudo service httpd restart
Redirecting to /bin/systemctl restart httpd.service
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ cd ~/downloads
[ec2-user@ip-172-31-41-160 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
```

Name: Shreyash Kamat Div: D15C Roll No: 22

Compile and install plugins
cd nagios-plugins-2.4.11
./configure --with-nagios-user=nagios --with-nagios-group=nagios

Run the following command:
sudo chkconfig --add nagios
On running the above command

```
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$ sudo chkconfig --add nagios
error reading information on service nagios: No such file or directory
```

If this is the output that one is getting, then it means that the init script is missing…
We can check this by running ls /etc/init.d/

```
[ec2-user@ip-172-31-92-249 nagios-plugins-2.4.9]$ ls /etc/init.d/
README  functions
[ec2-user@ip-172-31-92-249 nagios-plugins-2.4.9]$
```

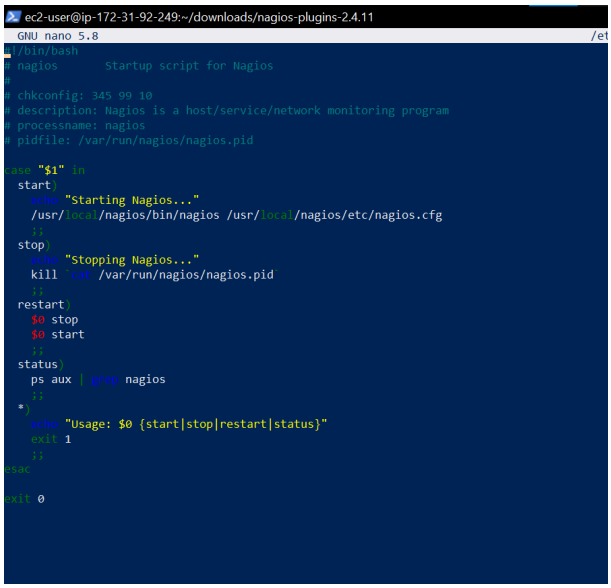
With ls command, we must see a file named nagios, which i was not able to see
If the Init Script is Missing i.e If you don't see the nagios script in /etc/init.d/, you can create it manually. Here's how:

Run the following command:
sudo nano /etc/init.d/nagios

Within this file, paste the following script

```
#!/bin/bash
# nagios       Startup script for Nagios
#
# chkconfig: 345 99 10
# description: Nagios is a host/service/network monitoring program
# processname: nagios
# pidfile: /var/run/nagios/nagios.pid
case "$1" in
  start)
    echo "Starting Nagios..."
    /usr/local/nagios/bin/nagios /usr/local/nagios/etc/nagios.cfg
    ;;
  stop)
    echo "Stopping Nagios..."
    kill `cat /var/run/nagios/nagios.pid`
    ;;
  restart)
    $0 stop
    $0 start
```

```
    ;;
  status)
    ps aux | grep nagios
    ;;
  *)
    echo "Usage: $0 {start|stop|restart|status}"
    exit 1
    ;;
esac
exit 0
```



Make the Script Executable: After saving the file, run the following command to make it executable:

sudo chmod +x /etc/init.d/nagios

Run **sudo chkconfig --add nagios** again

And then run **sudo chkconfig nagios on**

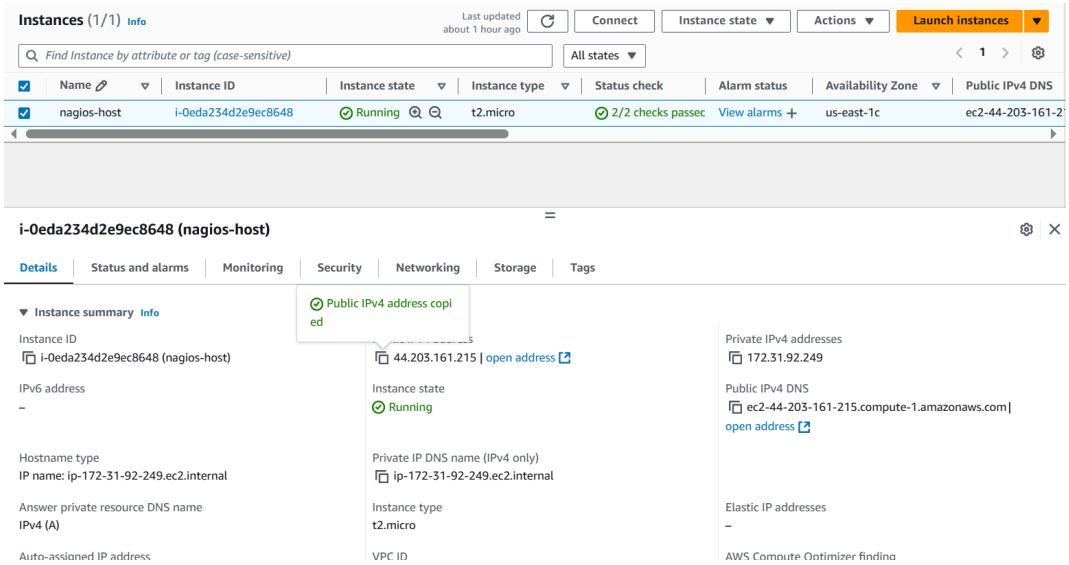
```
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$ sudo nano /etc/init.d/nagios
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$ sudo chmod +x /etc/init.d/nagios
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$ sudo chkconfig --add nagios
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$ sudo chkconfig nagios on
Note: Forwarding request to 'systemctl enable nagios.service'.
Synchronizing state of nagios.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable nagios
Created symlink /etc/systemd/system/multi-user.target.wants/nagios.service → /usr/lib/systemd/system/nagios.service.
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$
```

sudo service nagios start

Name: Shreyash Kamat Div: D15C Roll No: 22

```
[ec2-user@ip-172-31-92-249 nagios-plugins-2.4.11]$ sudo service nagios start
Starting Nagios...

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
Nagios 4.5.5 starting... (PID=72261)
Local time is Tue Oct 01 20:59:58 UTC 2024
wproc: Successfully registered manager as @wproc with query handler
wproc: Registry request: name=Core Worker 72265;pid=72265
wproc: Registry request: name=Core Worker 72264;pid=72264
wproc: Registry request: name=Core Worker 72263;pid=72263
wproc: Registry request: name=Core Worker 72262;pid=72262
Successfully launched command file worker with pid 72266
wproc: NOTIFY job 4 from worker Core Worker 72262 is a non-check helper but exited with return code 127
wproc:   host=localhost; service=Swap Usage; contact=nagiosadmin
wproc:   early_timeout=0; exited_ok=1; wait_status=32512; error_code=0;
wproc:   stderr line 01: /bin/sh: line 1: /bin/mail: No such file or directory
wproc:   stderr line 02: /usr/bin/printf: write error: Broken pipe
```

Get your public IPv4 address from your instance. We will require it for connecting to our nginx server



Browse for this url: http://<your_public_ip_address>/nagios
The browser may ask you for your nagios credentials which set in the earlier steps
The username is nagiosadmin and enter the password that you set earlier

Name: Shreyash Kamat Div: D15C Roll No: 22

Conclusion:

In this experiment, we successfully installed and configured Nagios Core on an Amazon Linux EC2 instance, showcasing its role in continuous monitoring within a DevOps environment. We learned about user management and service configuration, emphasizing Nagios's ability to monitor systems and networks effectively. This experience laid the groundwork for enhancing infrastructure reliability and integrating advanced monitoring strategies in future projects.

Name: Shreyash Kamat Div: D15C Roll No: 22

Adv DevOps Exp 10

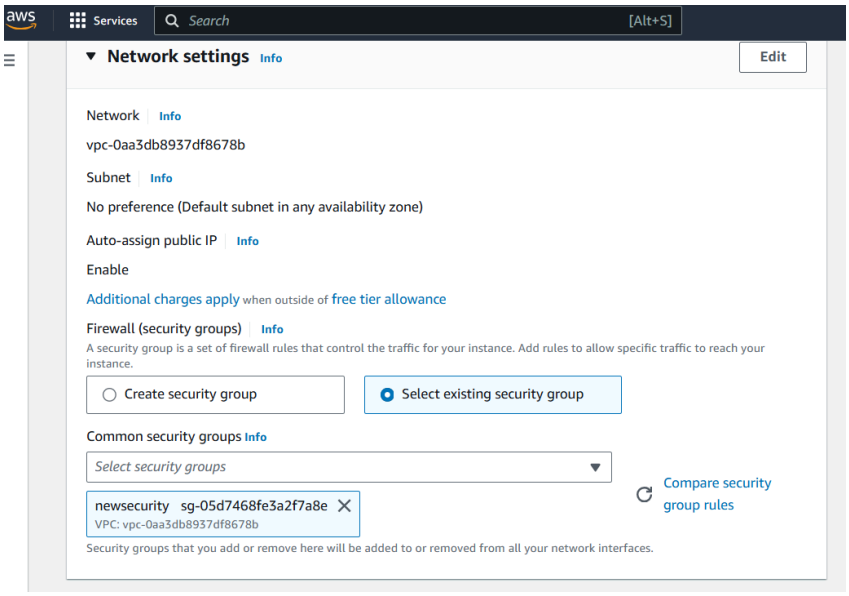
Aim: To perform Port, Service monitoring, Windows/Linux server monitoring using Nagios.

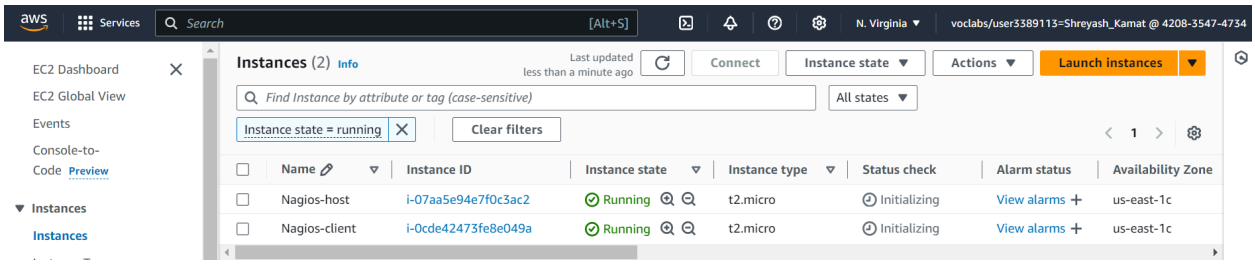
Monitoring Using Nagios:

Step 1: To Confirm Nagios is running on the server side Perform the following command on your Amazon Linux Machine (Nagios-host).
Run this command **sudo systemctl status**

```
ec2-user@ip-172-31-41-160:~/downloads/nagios-plugins-2.4.11
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$ sudo systemctl status
● ip-172-31-41-160.ec2.internal
    State: running
    Units: 296 loaded (incl. loaded aliases)
     Jobs: 0 queued
   Failed: 0 units
    Since: Wed 2024-10-02 12:28:05 UTC; 33min ago
  systemd: 252.23-2.amzn2023
  CGroup: /
          ├─init.scope
          │ └─1 /usr/lib/systemd/systemd --switched-root --system --deserialize=32
          ├─system.slice
          │ ├─acpid.service
          │ │ ├─1938 /usr/bin/systemd-inhibit --what=handle-suspend-key:handle-hibernate-key --who=noah "--why=acpid instead" --mode=block /usr/sbin/acpid -f
          │ │ └─2059 /usr/sbin/acpid -f
          │ ├─amazon-ssm-agent.service
          │ │ └─2141 /usr/bin/amazon-ssm-agent
          │ ├─atd.service
          │ │ └─2152 /usr/sbin/atd -f
          │ ├─auditd.service
          │ │ └─1768 /sbin/auditd
          │ ├─chronyd.service
          │ │ └─2175 /usr/sbin/chronyd -F 2
          │ ├─dbus-broker.service
          │ │ ├─1946 /usr/bin/dbus-broker-launch --scope system --audit
          │ │ └─1954 dbus-broker --log 4 --controller 9 --machine-id ec2e4d759a3e2f6fe850b14e4cdacabe --max-bytes 536870912 --max-fds 4096 --max-matches 16384 --audit
          │ ├─gssproxy.service
          │ │ └─1959 /usr/sbin/gssproxy -D
          │ ├─httpd.service
          │ │ ├─49553 /usr/sbin/httpd -DFOREGROUND
          │ │ ├─49555 /usr/sbin/httpd -DFOREGROUND
          │ │ ├─49556 /usr/sbin/httpd -DFOREGROUND
          │ │ ├─49557 /usr/sbin/httpd -DFOREGROUND
          │ │ ├─49558 /usr/sbin/httpd -DFOREGROUND
          │ │ └─62800 /usr/sbin/httpd -DFOREGROUND
          │ ├─libstoragemgmt.service
          │ │ └─1940 /usr/bin/lsmd -d
```

Step 2: Before we begin,
To monitor a Linux machine, create an **Ubuntu 20.04 server** EC2 Instance in AWS. Provide it with the **same security group** as the Nagios Host and name it ‘nagios-client’ alongside the host.





Step 3: TO BE DONE IN THE Nagios-host TERMINAL
In the nagios-host terminal, run this command
ps -ef | grep nagios

```
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$ ps -ef | grep nagios
ec2-user   63115    2315  0 13:03 pts/0    00:00:00 grep --color=auto nagios
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$
```

To become a root user, run **'sudo su'** and make two directories using the following commands. If one is running these commands in windows powershell, make sure that he/she copies it line by line as powershell might make an error while interpreting multiple lines
mkdir /usr/local/nagios/etc/objects/monitorhosts
mkdir /usr/local/nagios/etc/objects/monitorhosts/linuxhosts

```
[ec2-user@ip-172-31-92-249 ~]$ sudo su
[root@ip-172-31-92-249 ec2-user]# mkdir /usr/local/nagios/etc/objects/monitorhosts
[root@ip-172-31-92-249 ec2-user]# mkdir /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
[root@ip-172-31-92-249 ec2-user]#
```

Copy the sample localhost.cfg file to linuxhost folder. Use the following mentioned command to achieve it
cp /usr/local/nagios/etc/objects/localhost.cfg
/usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg

Open linuxserver.cfg using nano and make the following changes. This is a conf type file in which we will have to modify the configurations in way which will help us specify the hosts and clients to be monitored
nano /usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg

Changes to be made:

1. Change the hostname to linux-server (EVERYWHERE ON THE FILE)
2. Change address to the public IP address of your LINUX CLIENT.

3. Change hostgroup_name under hostgroup to linux-servers1

```
# HOST DEFINITION
#
###############################################################################

# 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               linux-server
    alias                   localhost
    address                 54.172.92.226
}



###############################################################################
```

```
# Define an optional hostgroup for Linux machines

define hostgroup {

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

IMP: Everywhere else on the file, change the hostname to linux-server instead of localhost.

Open the Nagios Config file and add the following line
nano /usr/local/nagios/etc/nagios.cfg

Add the following line in the file and save
cfg_dir=/usr/local/nagios/etc/objects/monitorhosts/

```
# OBJECT CONFIGURATION FILE(S)
# These are the object configuration files in which you define hosts,
# host groups, contacts, contact groups, services, etc.
# You can split your object definitions across several config files
# if you wish (as shown below), or keep them all in a single config file.

# You can specify individual object config files as shown below:
cfg_file=/usr/local/nagios/etc/objects/commands.cfg
cfg_file=/usr/local/nagios/etc/objects/contacts.cfg
cfg_file=/usr/local/nagios/etc/objects/timeperiods.cfg
cfg_file=/usr/local/nagios/etc/objects/templates.cfg

# Definitions for monitoring the local (Linux) host
cfg_file=/usr/local/nagios/etc/objects/localhost.cfg
cfg_dir=/usr/local/nagios/etc/objects/monitorhosts/
# Definitions for monitoring a Windows machine
#cfg_file=/usr/local/nagios/etc/objects/windows.cfg
```

Verify the configuration files by running the following command
/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

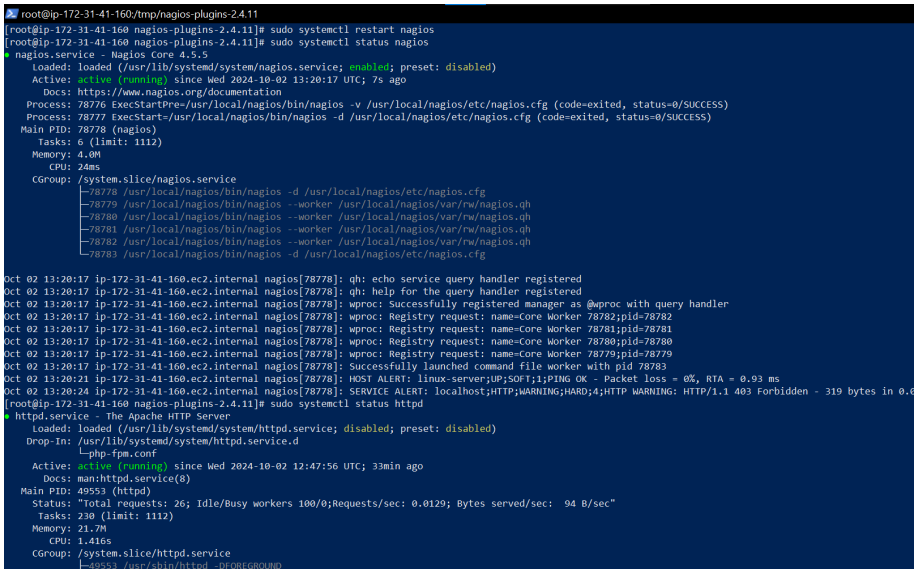
```
[root@ip-172-31-41-160 nagios-plugins-2.4.11]# /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

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

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

Running pre-flight check on configuration data...

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

Total Warnings: 0
Total Errors:   0

Things look okay - No serious problems were detected during the pre-flight check
[root@ip-172-31-41-160 nagios-plugins-2.4.11]#
```

You are good to go if there are no errors.

Restart the nagios service

service nagios restart

And by running sudo systemctl status nagios, we can again check whether our server is running or not



Step 4: TO BE DONE IN THE Nagios-client TERMINAL

Now it is time to switch to the client machine.

Name: Shreyash Kamat Div: D15C Roll No: 22

SSH into the machine or simply use the EC2 Instance Connect feature.

```
PS C:\WINDOWS\system32> cd C:\Users\Dell\Downloads
PS C:\Users\Dell\Downloads> ssh -i "mohit.pem" ubuntu@ec2-54-172-92-226.compute-1.amazonaws.com
The authenticity of host 'ec2-54-172-92-226.compute-1.amazonaws.com (54.172.92.226)' can't be established.
ECDSA key fingerprint is SHA256:e/WkFQRuHSqPjqQ5hDMaA0dku8msNhETN9SAgzEy53E.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-54-172-92-226.compute-1.amazonaws.com,54.172.92.226' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-1016-aws x86_64)

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

 System information as of Wed Oct  2 13:26:11 UTC 2024

  System load:  0.0               Processes:             104
  Usage of /:   22.8% of 6.71GB   Users logged in:       0
  Memory usage: 20%               IPv4 address for enX0: 172.31.36.100
  Swap usage:   0%

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

   https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status



The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
```

Make a package index update and install gcc, nagios-nrpe-server and the plugins. Run the following commands to achieve the same.

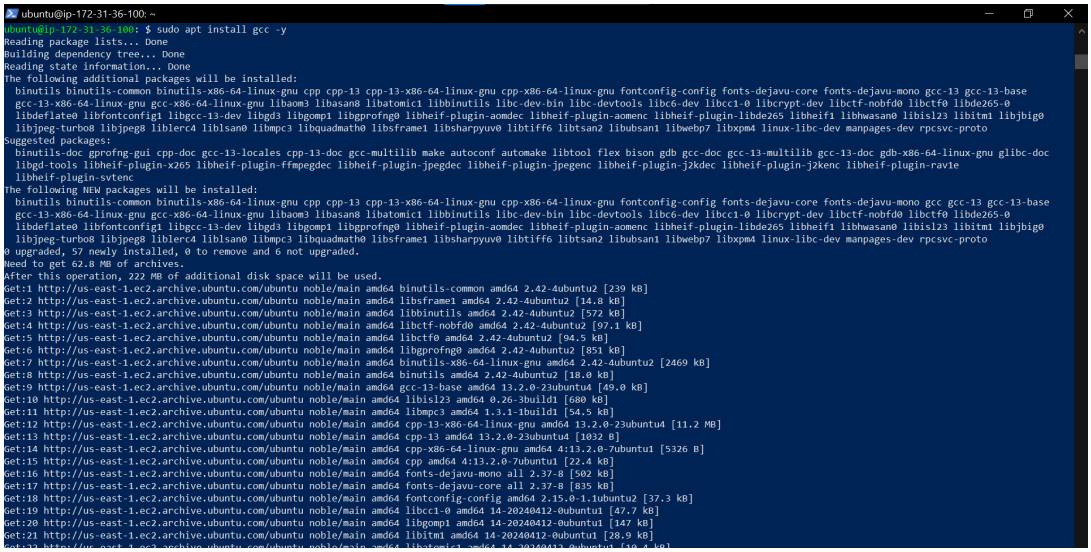
sudo apt update -y

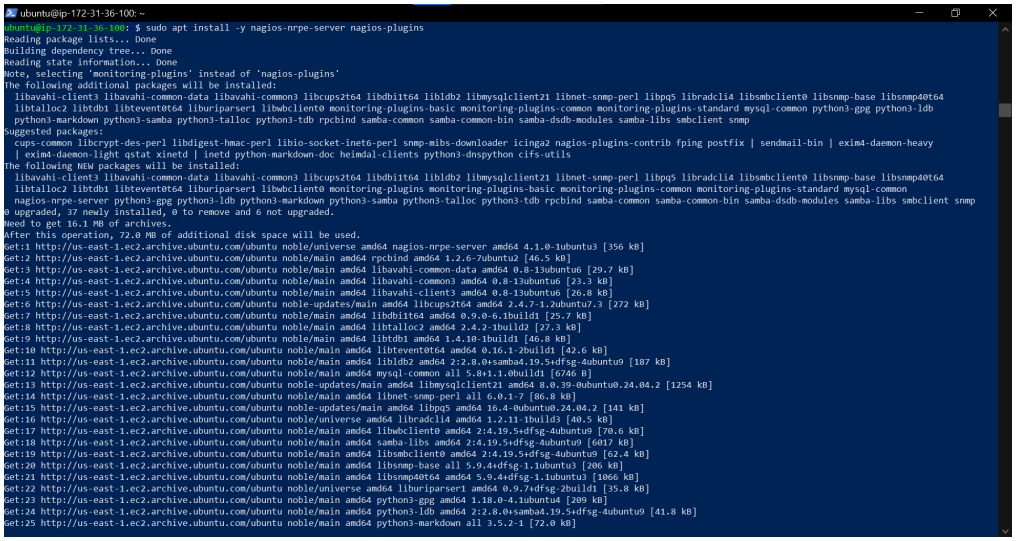
sudo apt install gcc -y

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

```
ubuntu@ip-172-31-36-100: ~
ubuntu@ip-172-31-36-100: $ sudo apt update -y
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:6 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [380 kB]
Get:7 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [83.1 kB]
Get:8 http://security.ubuntu.com/ubuntu noble-security/main amd64 c-n-f Metadata [4576 B]
Get:9 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [275 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:11 http://security.ubuntu.com/ubuntu noble-security/universe Translation-en [116 kB]
Get:12 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [8632 B]
Get:13 http://security.ubuntu.com/ubuntu noble-security/universe amd64 c-n-f Metadata [10.4 kB]
Get:14 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Packages [10.9 kB]
Get:15 http://security.ubuntu.com/ubuntu noble-security/multiverse Translation-en [2808 B]
Get:16 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [208 B]
Get:17 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 c-n-f Metadata [344 B]
Get:18 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:19 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
Get:20 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
Get:21 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
Get:22 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Components [35.0 kB]
Get:23 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 c-n-f Metadata [8328 B]
Get:24 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [535 kB]
Get:25 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [130 kB]
Get:26 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 c-n-f Metadata [8676 B]
Get:27 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [380 kB]
Get:28 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [157 kB]
Get:29 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [45.0 kB]
Get:30 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 c-n-f Metadata [14.9 kB]
Get:31 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Packages [14.4 kB]
Get:32 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse Translation-en [3608 B]
Get:33 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [212 B]
Get:34 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 c-n-f Metadata [532 B]
Get:35 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 Components [208 B]
Get:36 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 c-n-f Metadata [112 B]
Get:37 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Packages [10.6 kB]
Get:38 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe Translation-en [10.8 kB]
Get:39 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Components [17.6 kB]
Get:40 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 c-n-f Metadata [1104 B]
Get:41 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/restricted amd64 Components [216 B]
Get:42 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/restricted amd64 c-n-f Metadata [116 B]
Get:43 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 Components [212 B]
```

Name: Shreyash Kamat Div: D15C Roll No: 22

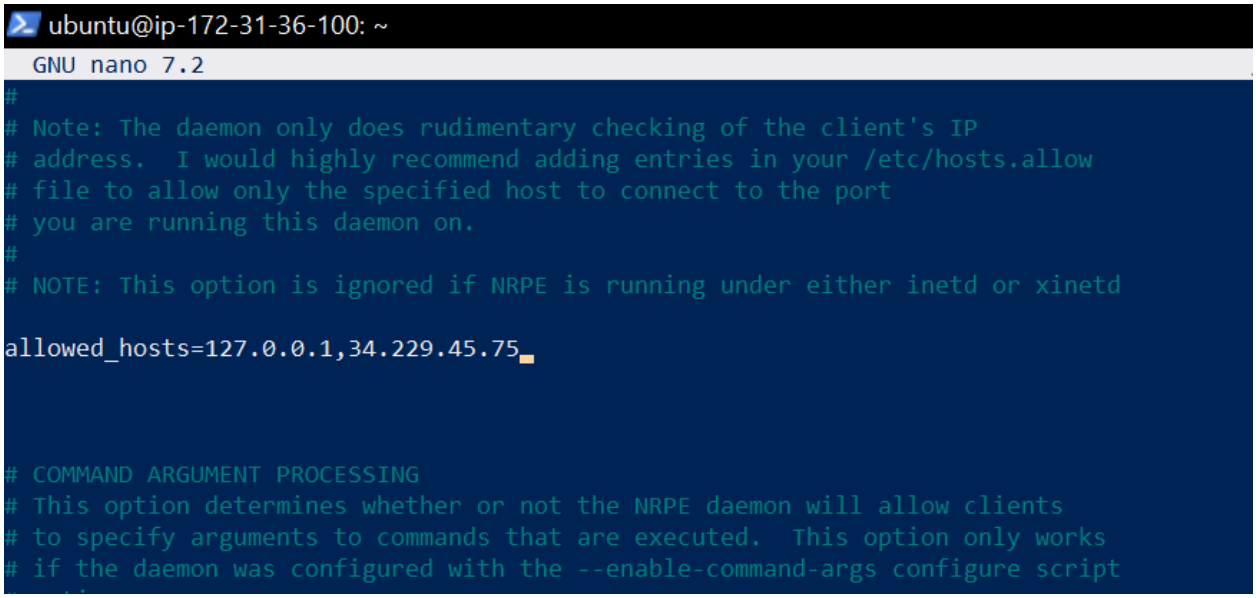




Open nrpe.cfg file to make changes.

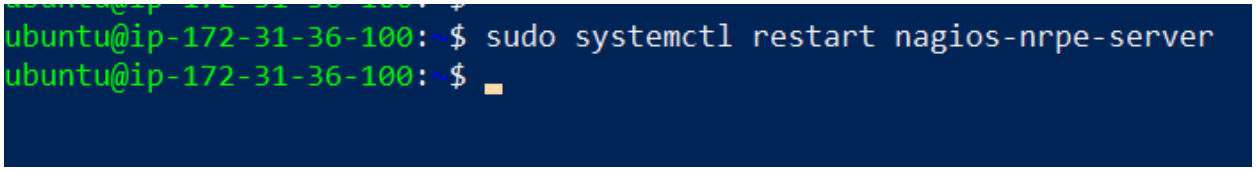
sudo nano /etc/nagios/nrpe.cfg

Under allowed_hosts, add your nagios host IP address like so



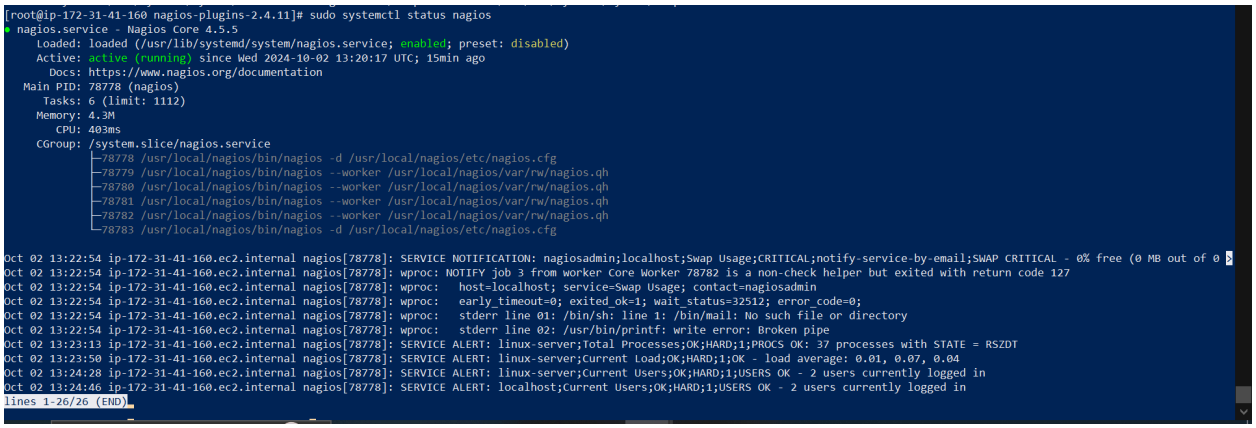
Now restart the NRPE server by this command.

sudo systemctl restart nagios-nrpe-server



Run the following command in the Nagios-host terminal

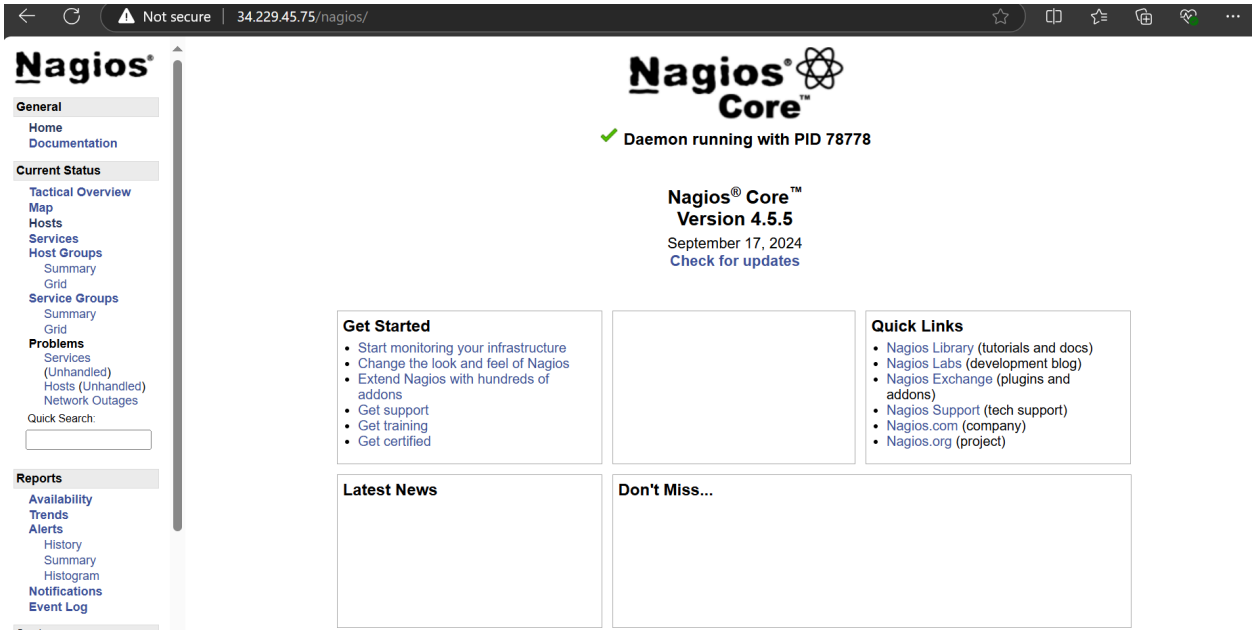
sudo systemctl status nagios



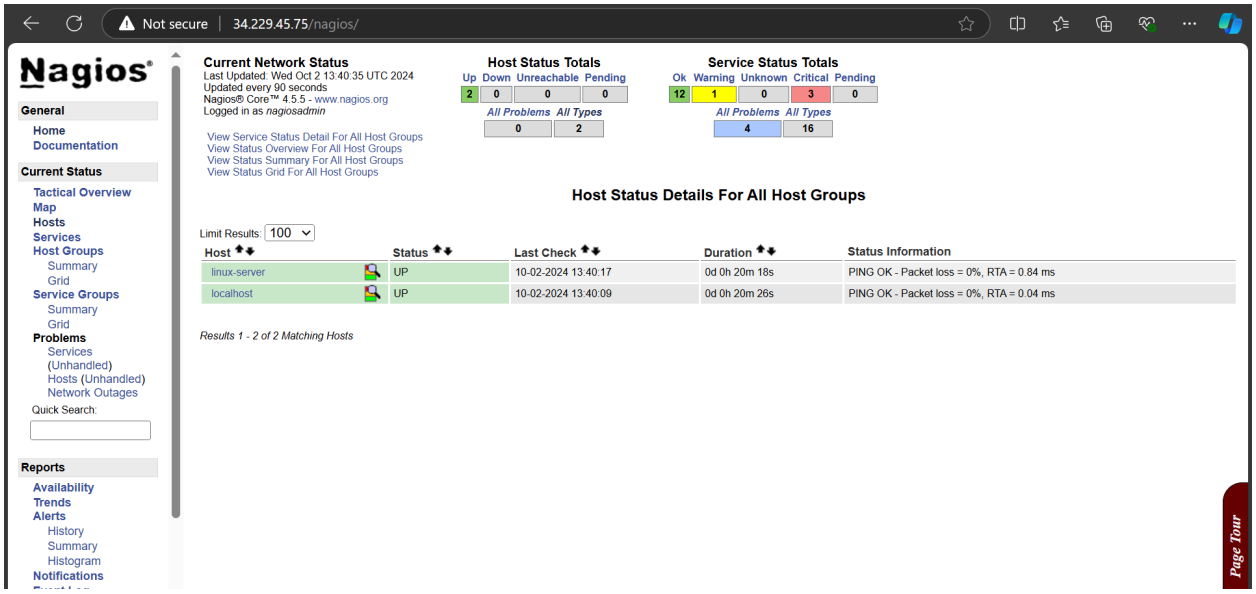
Step 5: Visiting your nagios server using your nagios-host ip address

Open up your browser and look for http://<public_ip_address_of_nagios-host>/nagios

Name: Shreyash Kamat Div: D15C Roll No: 22

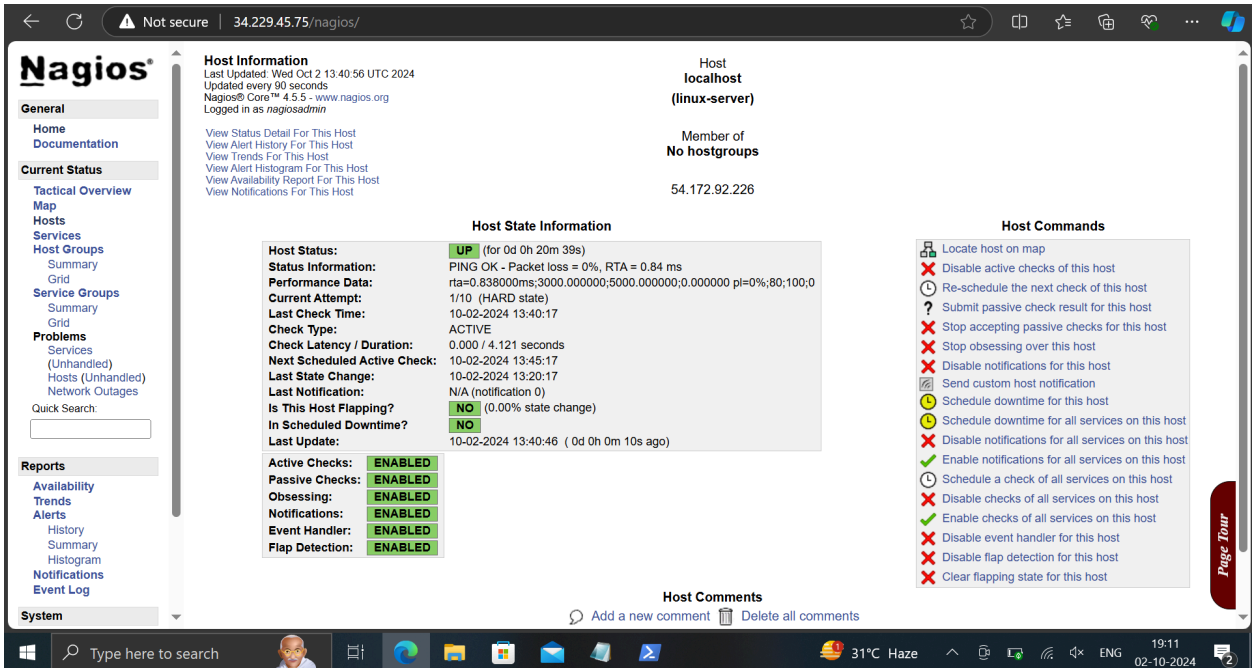


Click on Hosts.

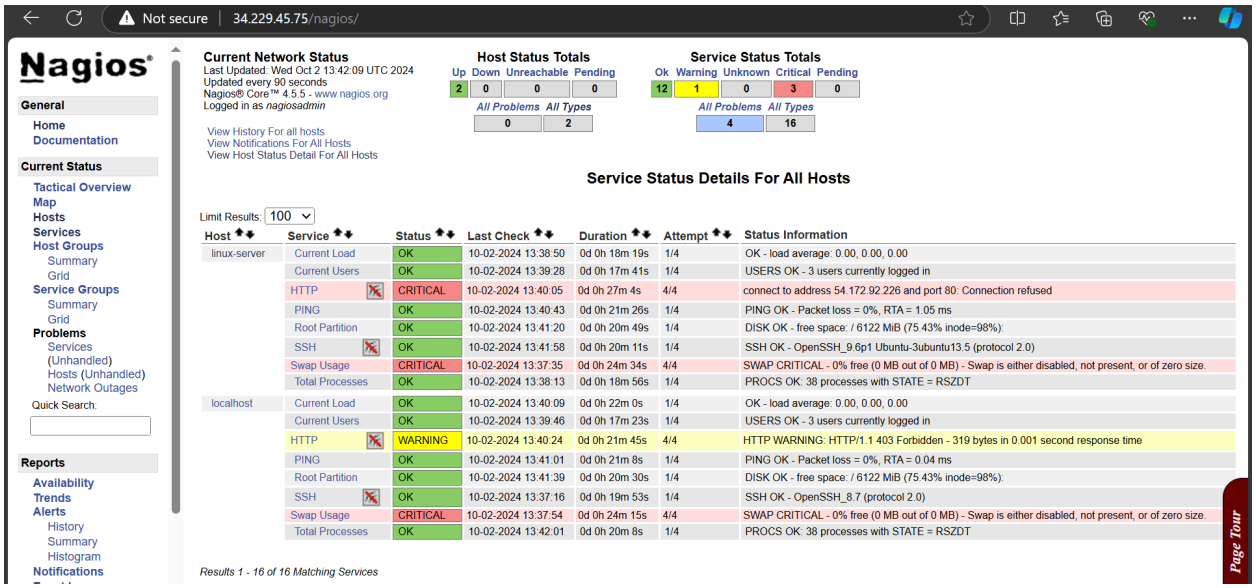


Click on linux-server to view host information

Name: Shreyash Kamat Div: D15C Roll No: 22



We can even navigate to the services section, which explicitly mentions the status, duration, checks, information about the numerous services present on our hosts



Conclusion: In conclusion, the experiment focused on monitoring ports, services, and a Linux server using Nagios. Through the step-by-step process, we successfully configured Nagios to monitor essential network services on the Linux server. By setting up both the Nagios host and client, we were able to track system performance, ensure service availability, and monitor key metrics like CPU and memory usage

Name: **Shreyash Kamat** Div/ Roll no: **D15C/ 22**

Experiment 11

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

Theory:

AWS Lambda

A fully managed, serverless computing service where you run code without provisioning or managing servers. Lambda automatically scales your application based on the number of incoming requests or events, ensuring efficient resource utilization. You are only charged for the time your code is running, with no upfront cost, making it cost-effective for on-demand workloads.

Lambda Workflow

- **Create a Function**: Write the function code and define its handler (entry point). You can use the AWS Console, CLI, or upload a deployment package.
- **Set Event Sources**: Define how the function is triggered (e.g., when an object is uploaded to S3 or a DynamoDB table is updated).
- **Execution**: When triggered, Lambda runs your function, executes the logic, and automatically scales to handle the incoming event volume.
- **Scaling and Concurrency**: Lambda scales automatically by launching more instances of the function to handle simultaneous invocations. There are also options for configuring **reserved concurrency** to manage traffic.
- **Monitoring and Logging**: Lambda integrates with Amazon CloudWatch for logging and monitoring. Logs for each invocation are sent to CloudWatch, allowing you to track performance and troubleshoot errors.

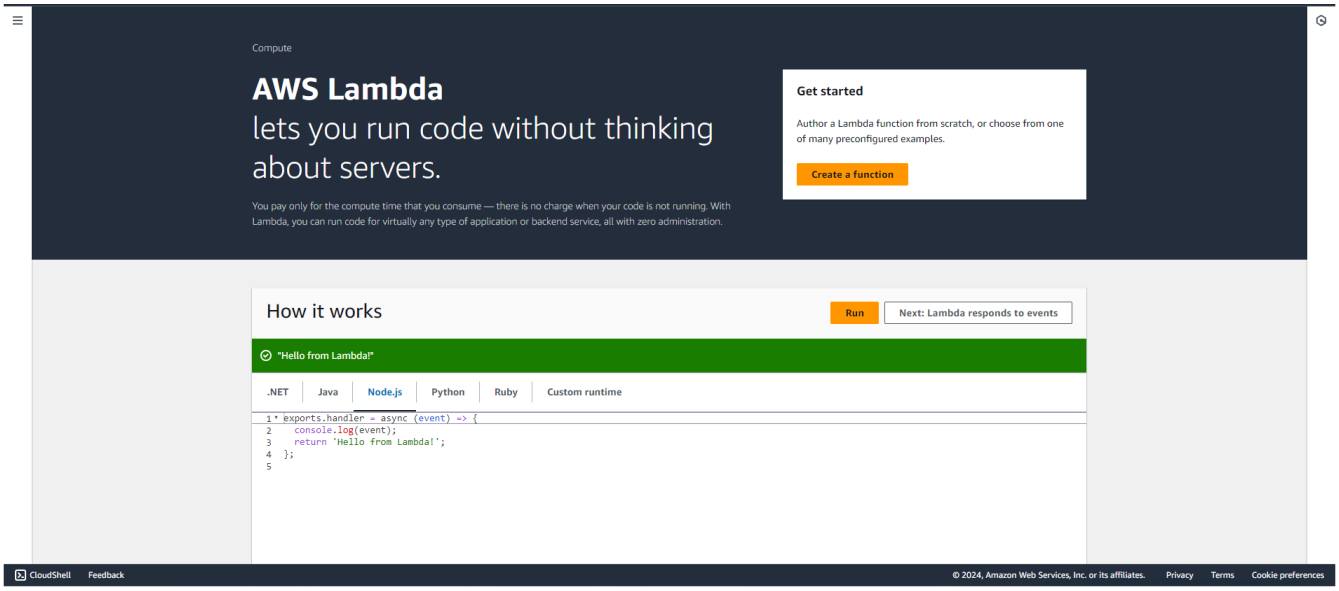
AWS Lambda Functions

- **Python**: Great for quick development with its rich standard library and support for lightweight tasks.
- **Java**: Typically used for more complex, compute-intensive tasks. While it's robust, cold start times can be higher.
- **Node.js**: Excellent for I/O-bound tasks like handling APIs or streaming data, with fast startup times and efficient memory usage.

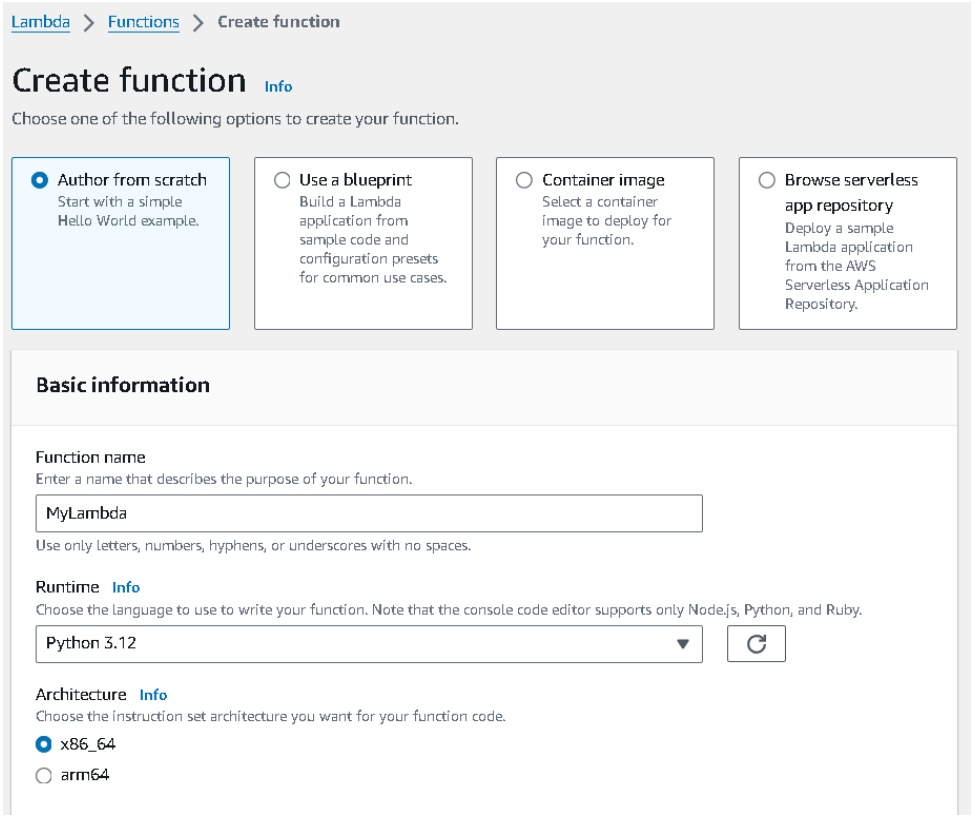
Prerequisites: AWS Personal/Academy Account

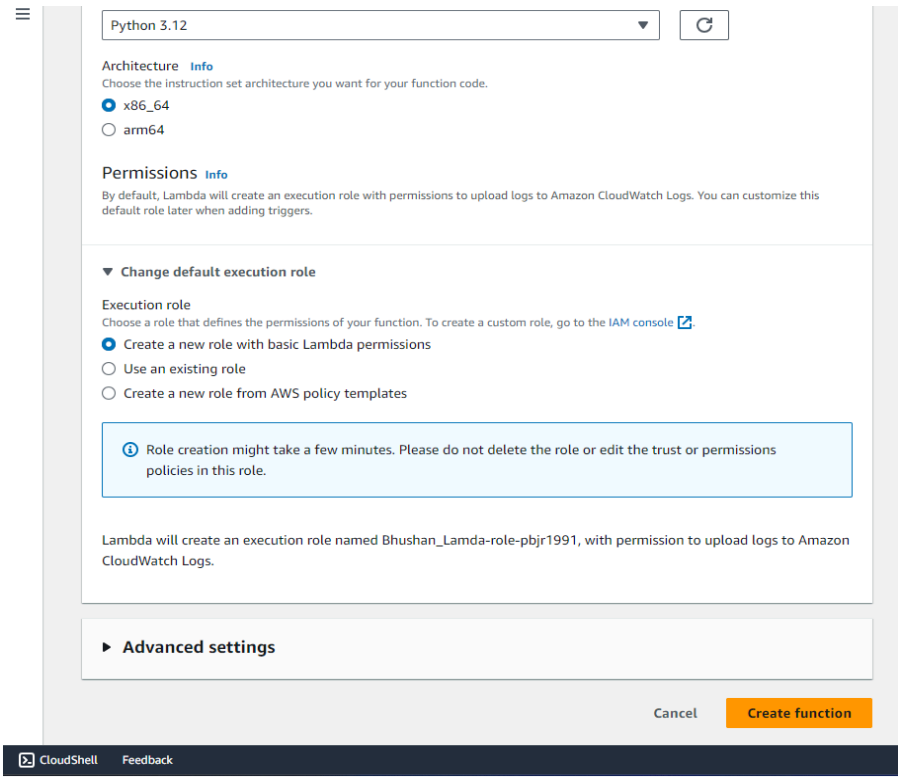
Steps To create the lambda function:

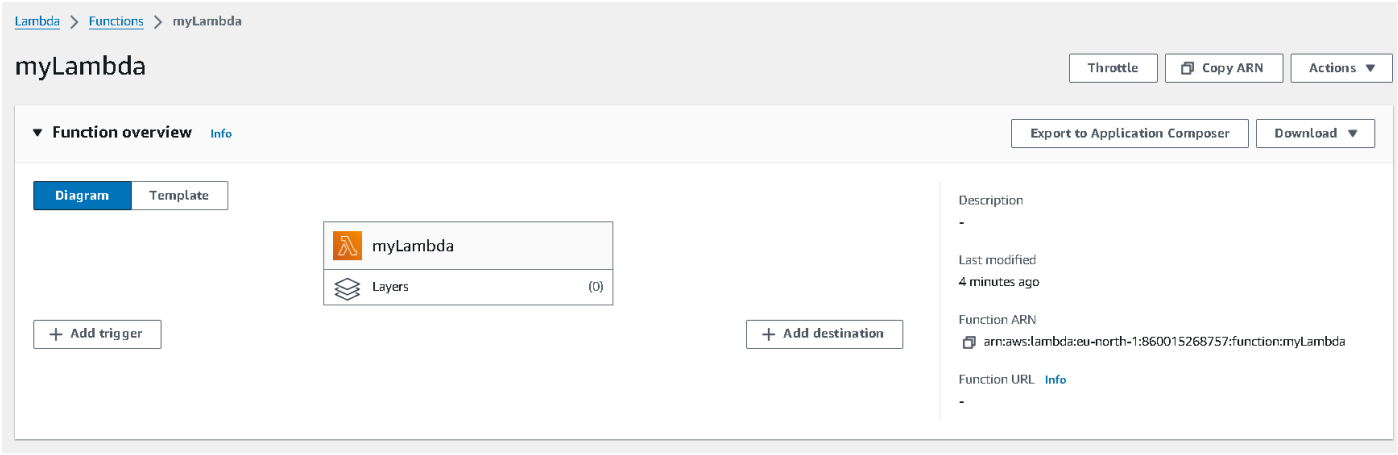
Step 1: Login to your AWS Personal/Academy Accout.Open lambda and click on create function button.

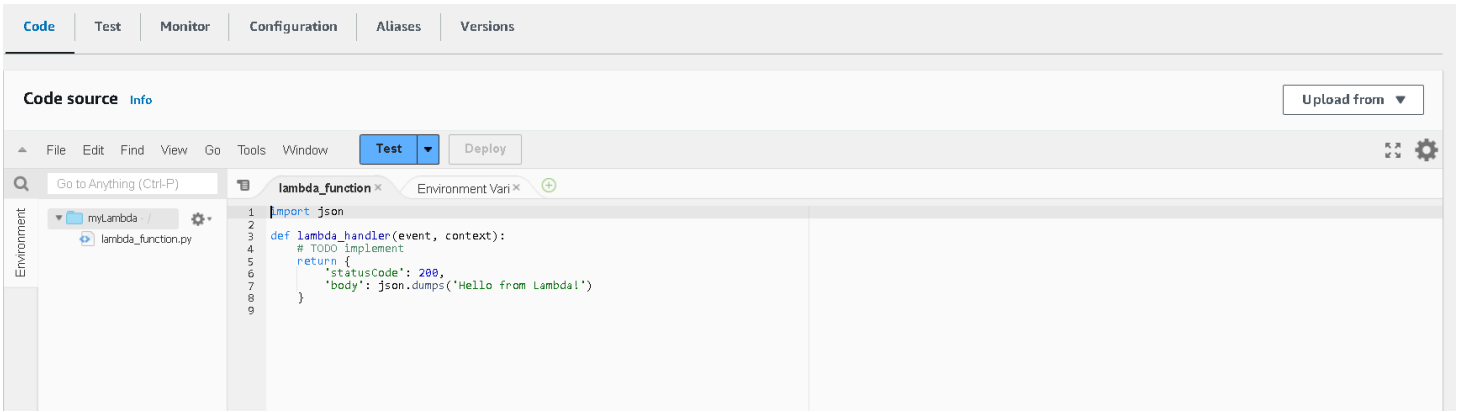


Step 2: Now Give a name to your Lambda function, Select the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby. So will select Python 3.12, Architecture as x86, and Execution role to Create a new role with basic Lambda permissions.

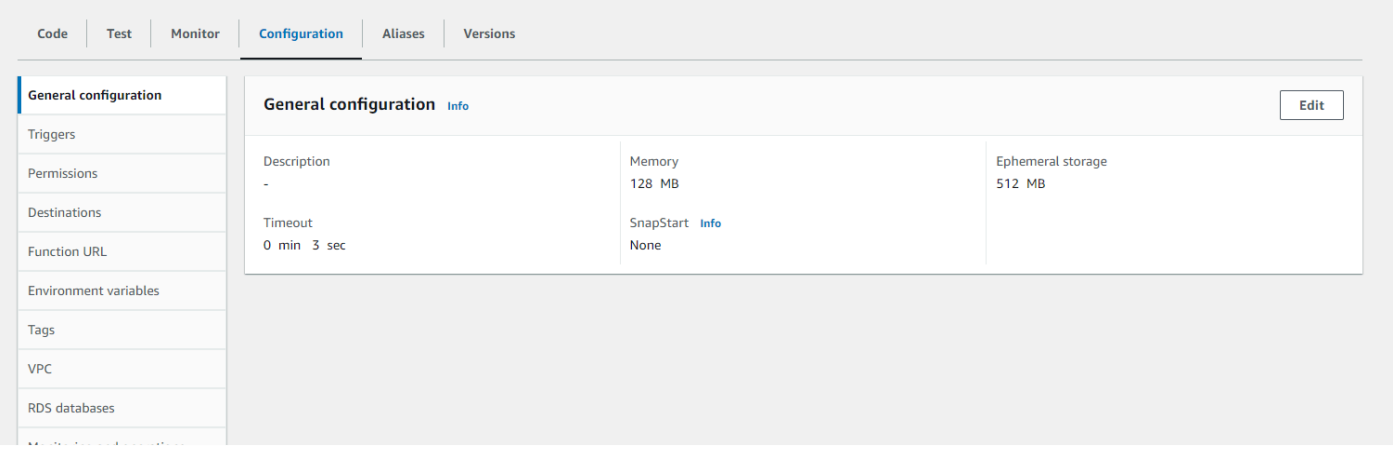



Python 3.12
Architecture Info
Choose the instruction set architecture you want for your function code.
x86_64
arm64
Permissions Info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.
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
Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.
Lambda will create an execution role named Bhushan_Lamda-role-pbjr1991, with permission to upload logs to Amazon CloudWatch Logs.
Advanced settings
Cancel
Create function
CloudShell
Feedback

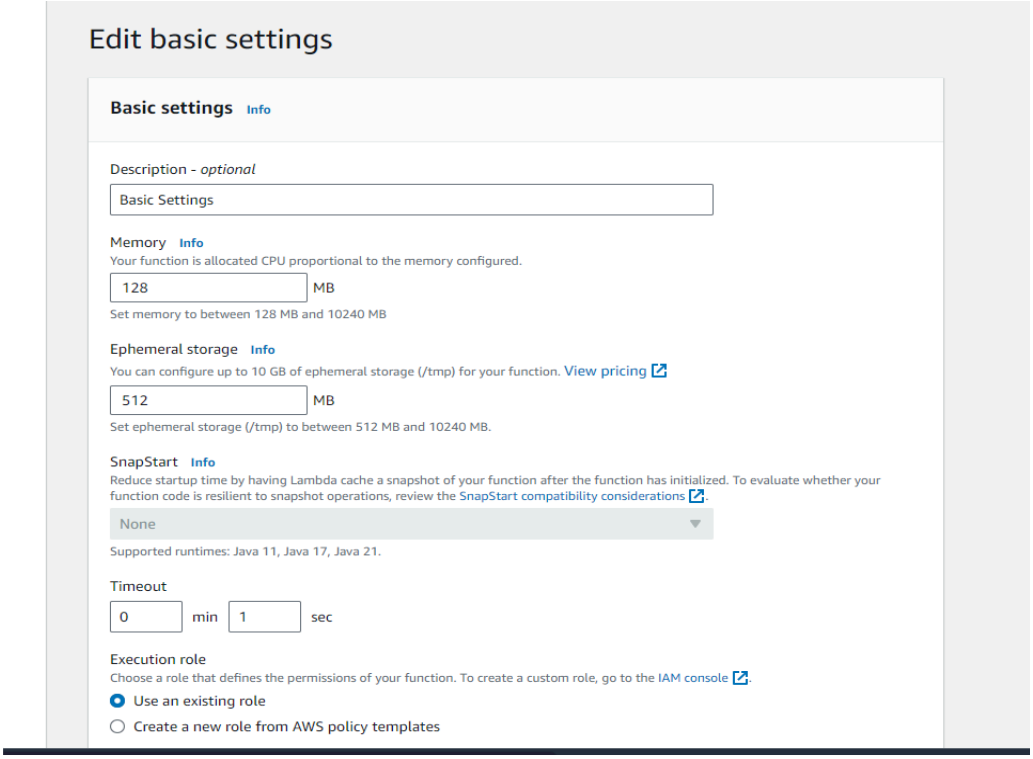

Lambda > Functions > myLambda
myLambda
Throttle
Copy ARN
Actions
Function overview Info
Export to Application Composer
Download
Diagram
Template
myLambda
Layers (0)
Add trigger
Add destination
Description
-
Last modified
4 minutes ago
Function ARN
arn:aws:lambda:eu-north-1:860015268757:function:myLambda
Function URL Info
-


Code
Test
Monitor
Configuration
Aliases
Versions
Code source Info
Upload from
File Edit Find View Go Tools Window
Test
Deploy
Go to Anything (Ctrl-P)
lambda_function
Environment Vari
Environment
myLambda
lambda_function.py
import json
def lambda_handler(event, context):
# TODO implement
return {
'statusCode': 200,
'body': json.dumps('Hello from Lambda!')
}

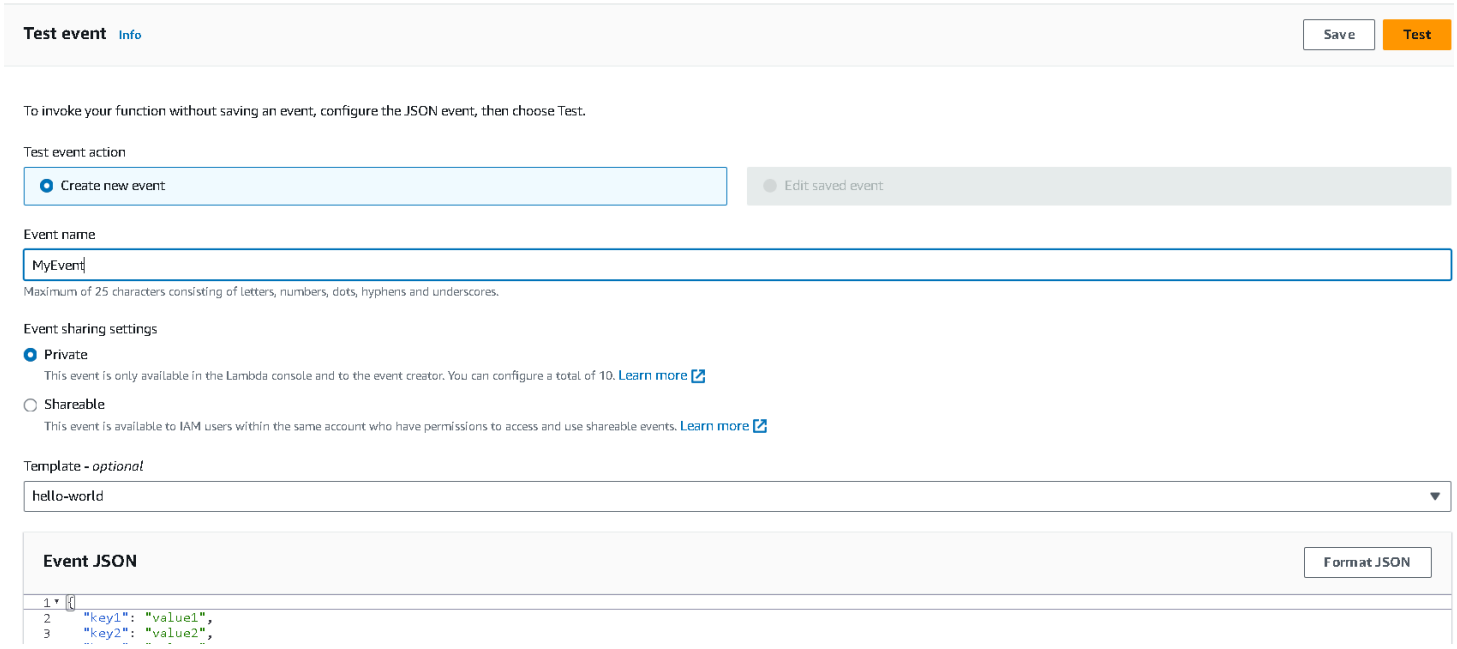
So See or Edit the basic settings go to configuration then click on edit general setting.



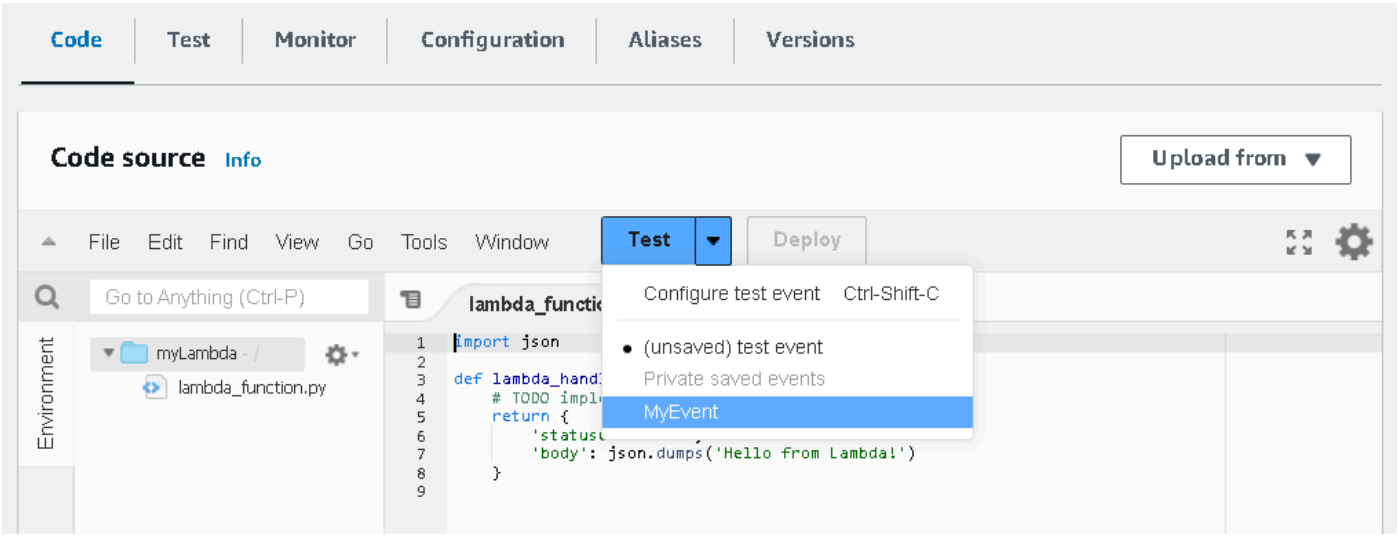
Here, you can enter a description and change Memory and Timeout. I've changed the Timeout period to 1 sec since that is sufficient for now.

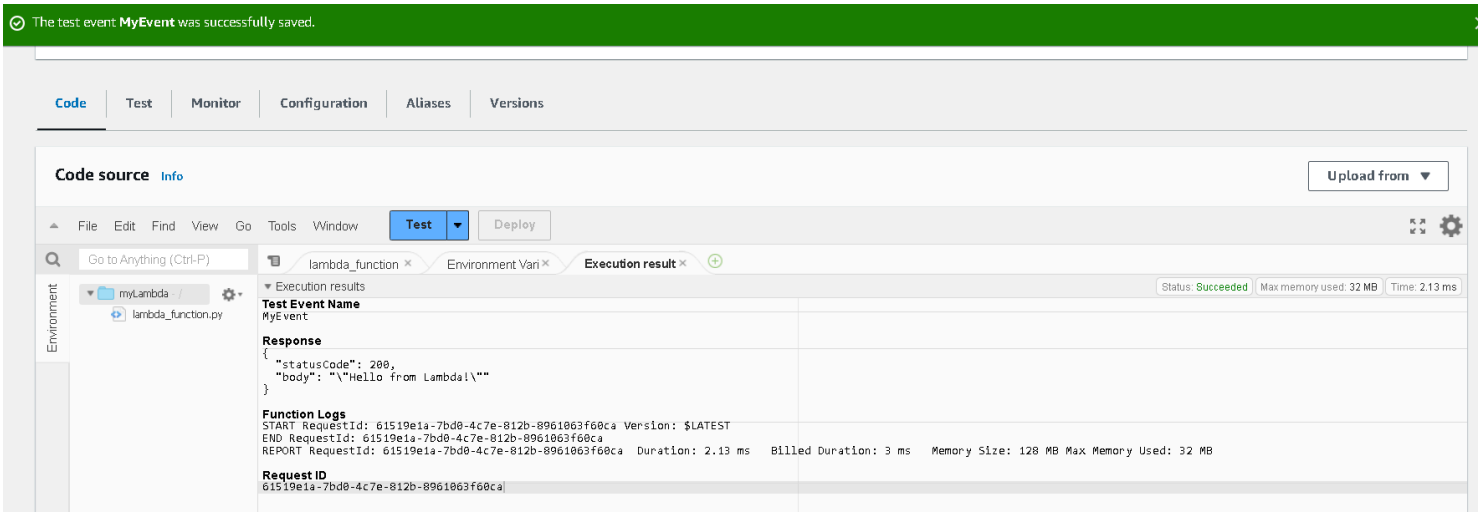


Step 3: Now Click on the Test tab then select Create a new event, give a name to the event and select Event Sharing to private, and select hello-world template.



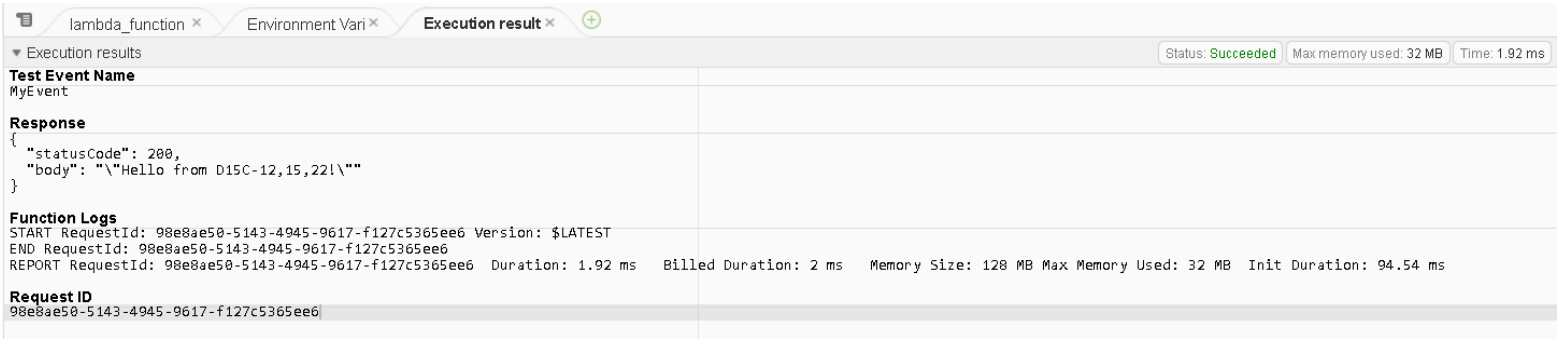
Step 4: Now In the Code section select the created event from the dropdown of test then click on test . You will see the below output.





Step 5: You can edit your lambda function code. I have changed the code to display the new String.

Step 6: Now click on the test and observe the output. We can see the status code 200 and your string output and function logs. On successful deployment.



Conclusion:

In this experiment, we successfully implemented an AWS Lambda function, covering all the key steps involved. Starting with the function's setup in Python, we configured essential settings such as adjusting the timeout to 1 second. A test event was then created, followed by deploying the function and verifying its output. We also made code modifications to the Lambda function, redeployed it, and observed the real-time effects of these changes. This hands-on experience highlighted the ease and adaptability of AWS Lambda for building serverless applications, enabling developers to concentrate on writing code while AWS handles infrastructure and scalability.

Name: **Shreyash Kamat** Div/ Roll no: **D15C/ 22**

Experiment 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

Theory:

AWS Lambda and S3 Integration:

AWS Lambda allows you to execute code in response to various events, including those triggered by Amazon S3. When an object is added to an S3 bucket, it can trigger a Lambda function to execute, allowing for event-driven processing without managing servers.

Workflow:

1. **Create an S3 Bucket:**
   - First, create an S3 bucket that will store the objects. This bucket will act as the trigger source for the Lambda function.

2. **Create the Lambda Function:**
   - Set up a new Lambda function using AWS Lambda’s console. You can choose a runtime environment like Python, Node.js, or Java.
   - Write code that logs a message like “An Image has been added” when triggered.

3. **Set Up Permissions:**
   - Ensure that the Lambda function has the necessary permissions to access S3. You can do this by attaching an IAM role with policies that allow reading from the bucket and writing logs to CloudWatch.

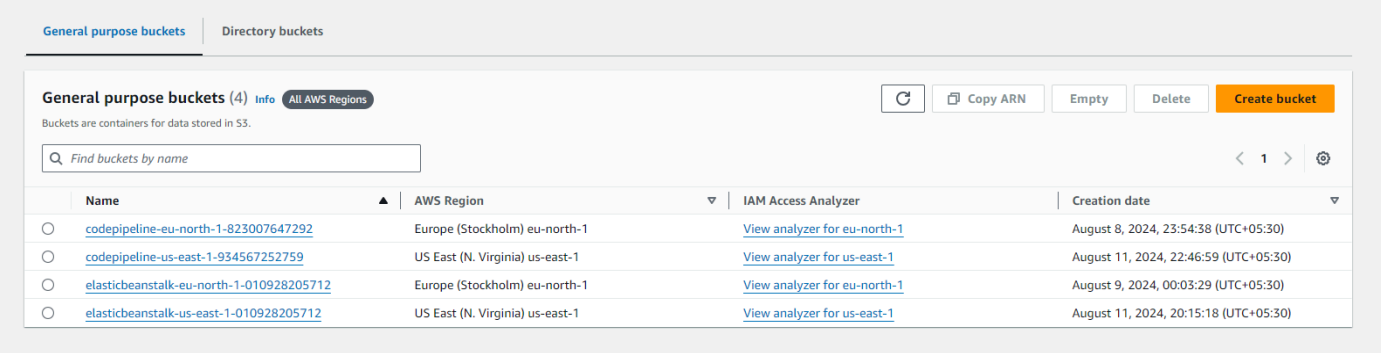
4. **Configure S3 Trigger:**
   - Link the S3 bucket to the Lambda function by setting up a trigger. Specify that the function should be triggered when an object is created in the bucket (e.g., when an image is uploaded).

5. **Test the Setup:**
   - Upload an object (e.g., an image) to the S3 bucket to test the trigger. The Lambda function should execute and log the message “An Image has been added” in AWS CloudWatch Logs.

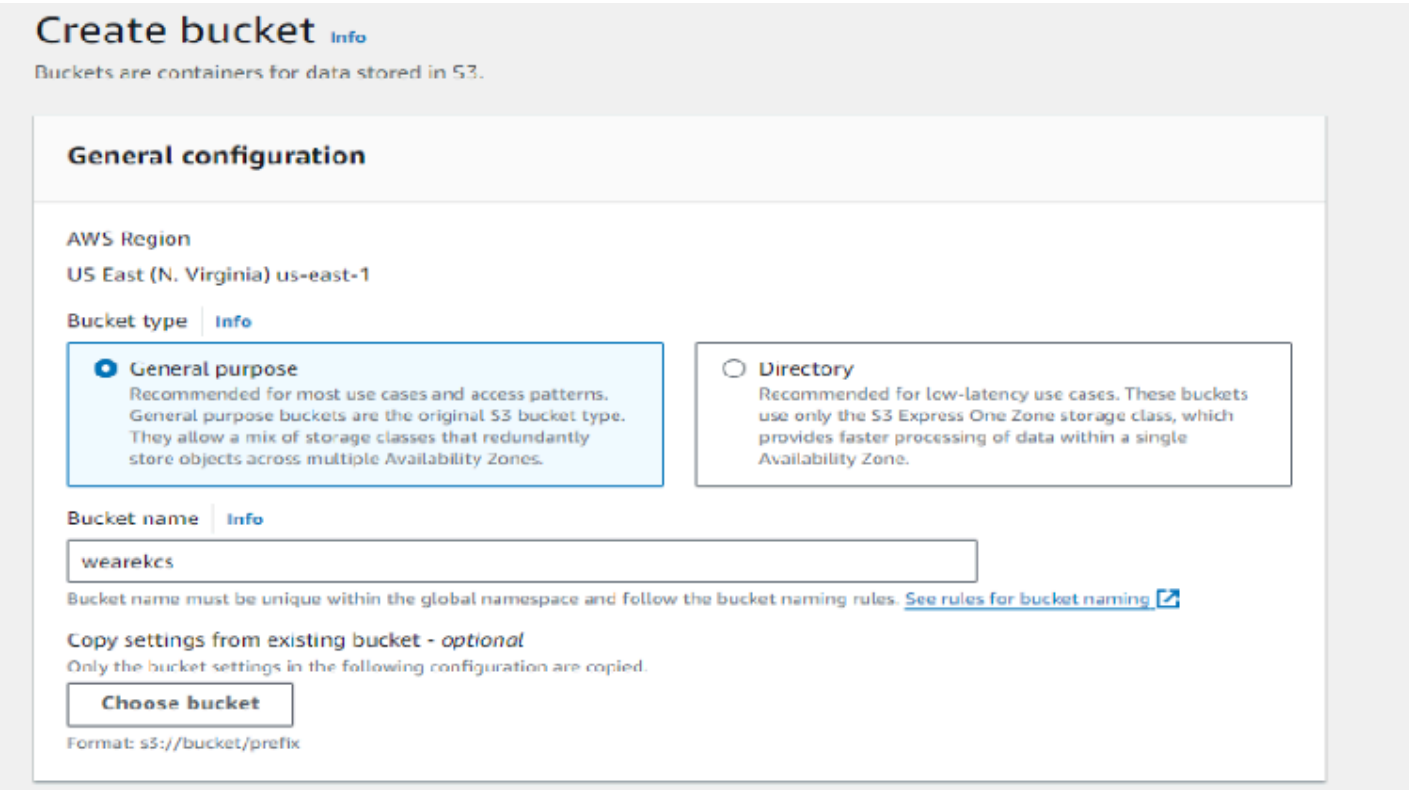
Prerequisites: AWS Personal Account

Steps To create the lambda function:

Step 1: Login to your AWS Personal account. Now open S3 from services and click on create S3 bucket.



Step 2: Now Give a name to the Bucket, select general purpose project and deselect the Block public access and keep other this to default.



Object Ownership Info

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

- (•) ACLs disabled (recommended)
  All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.
- ( ) ACLs enabled
  Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

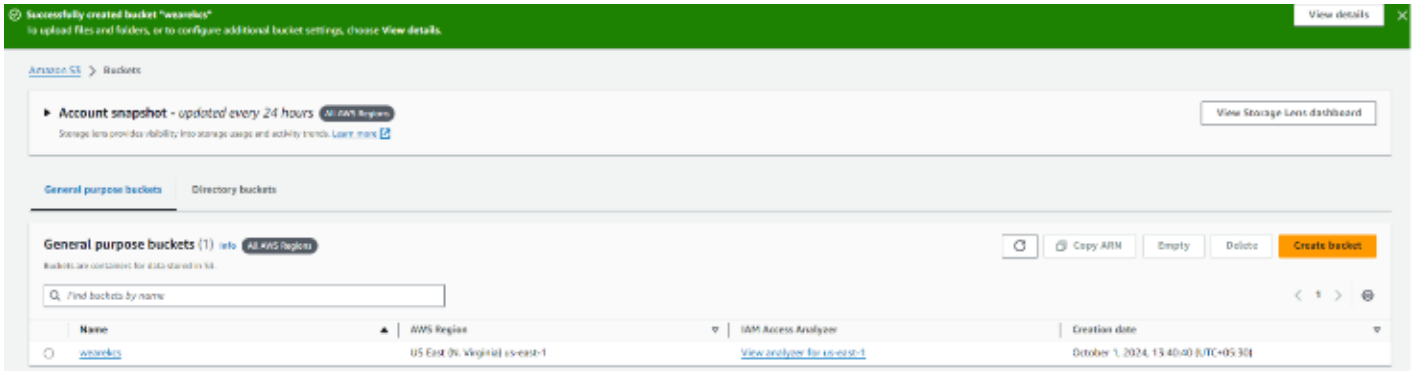
Object Ownership

Bucket owner enforced

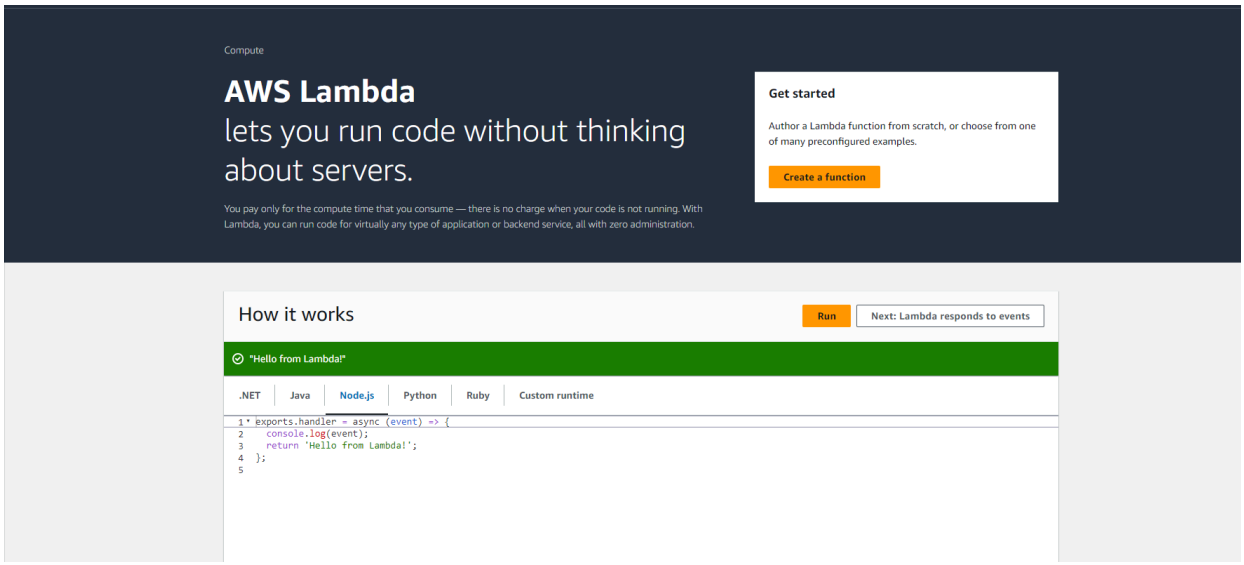
Block Public Access settings for this bucket

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

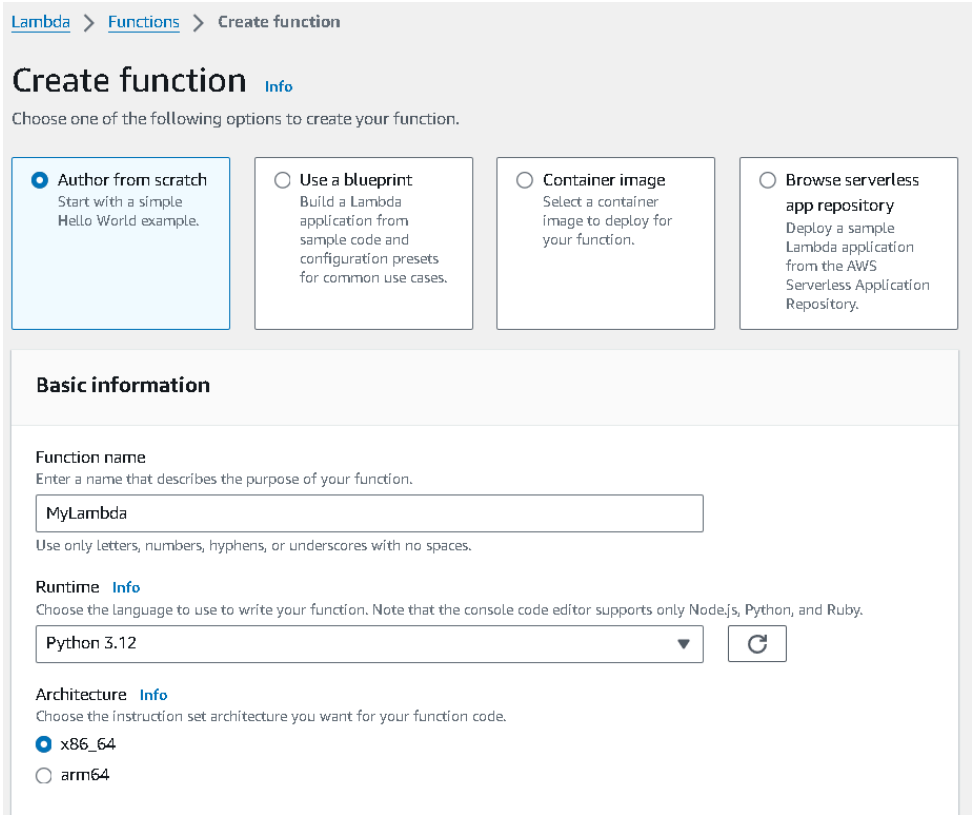
- [ ] **Block *all* public access**
  Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.
  - [ ] **Block public access to buckets and objects granted through *new* access control lists (ACLs)**
    S3 will block public 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.

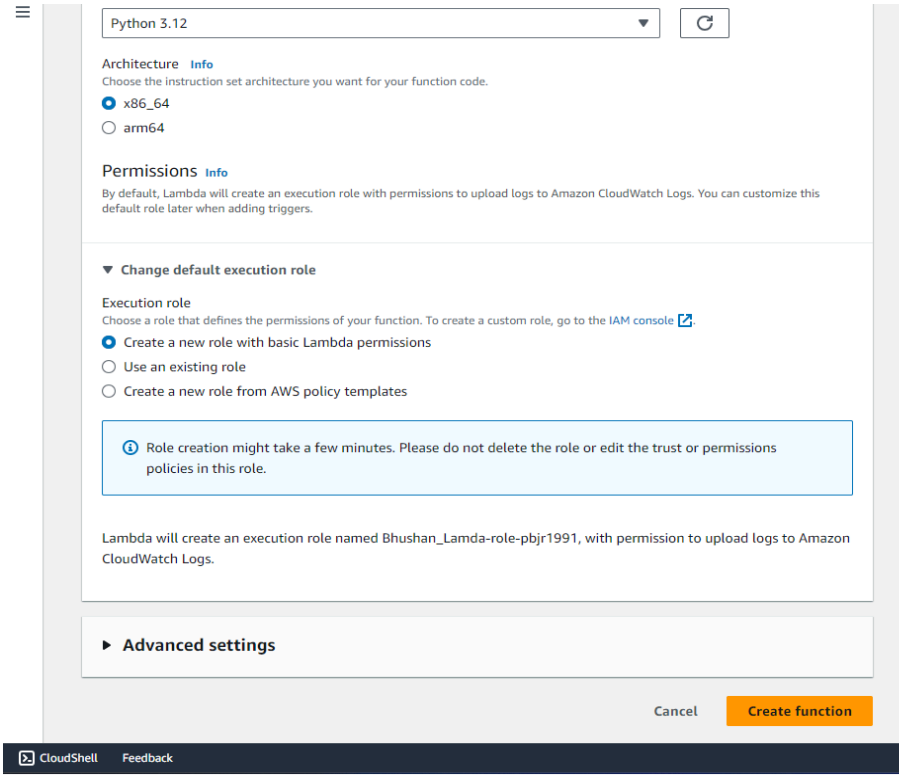


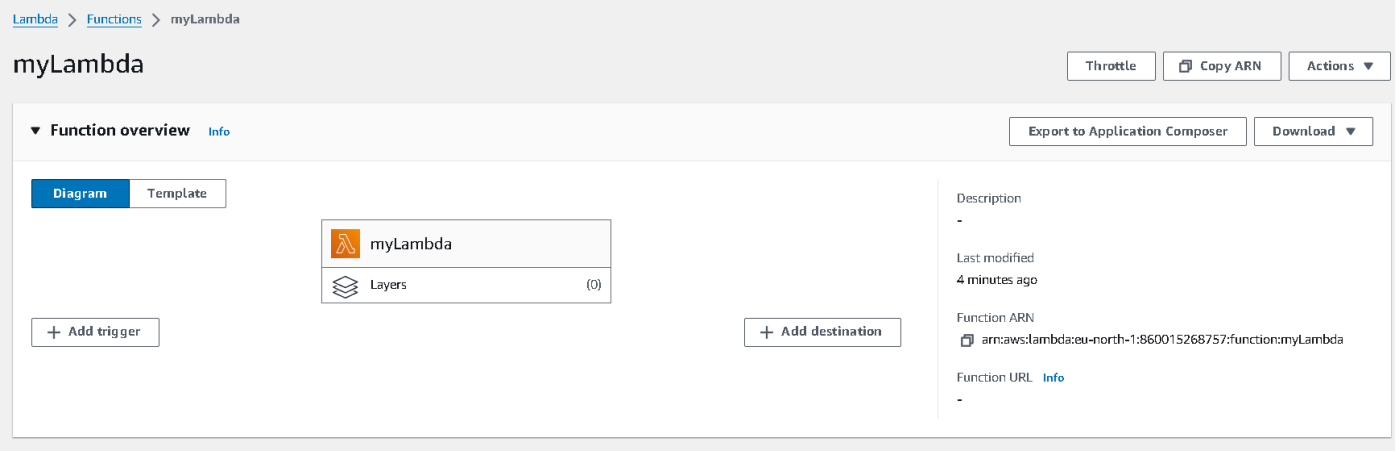
Step 3: Open lambda console and click on create function button.

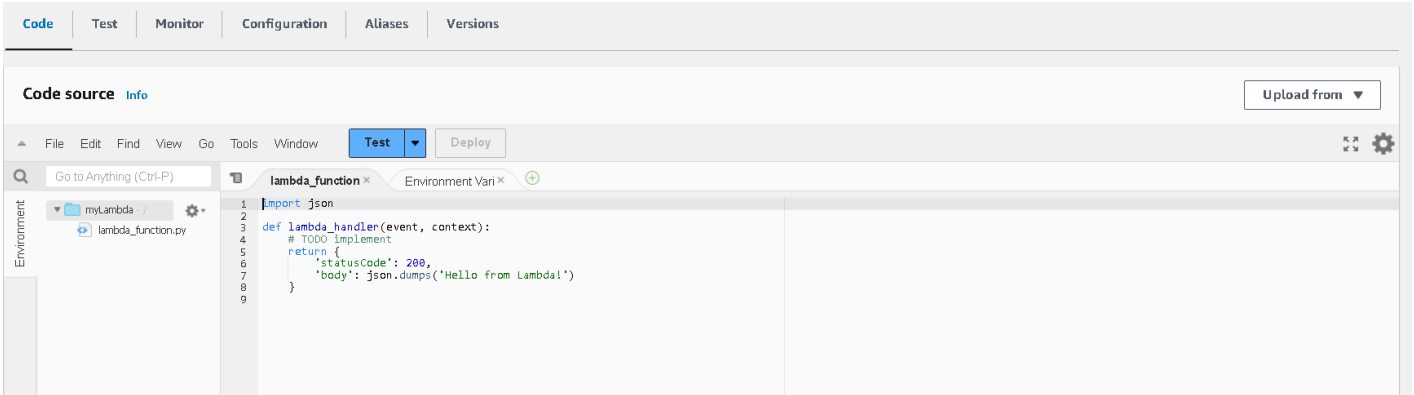


Step 4: Now Give a name to your Lambda function, Select the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby. So will select Python 3.12
, Architecture as x86, and Exceution role to Create a new role with basic Lambda permissions.

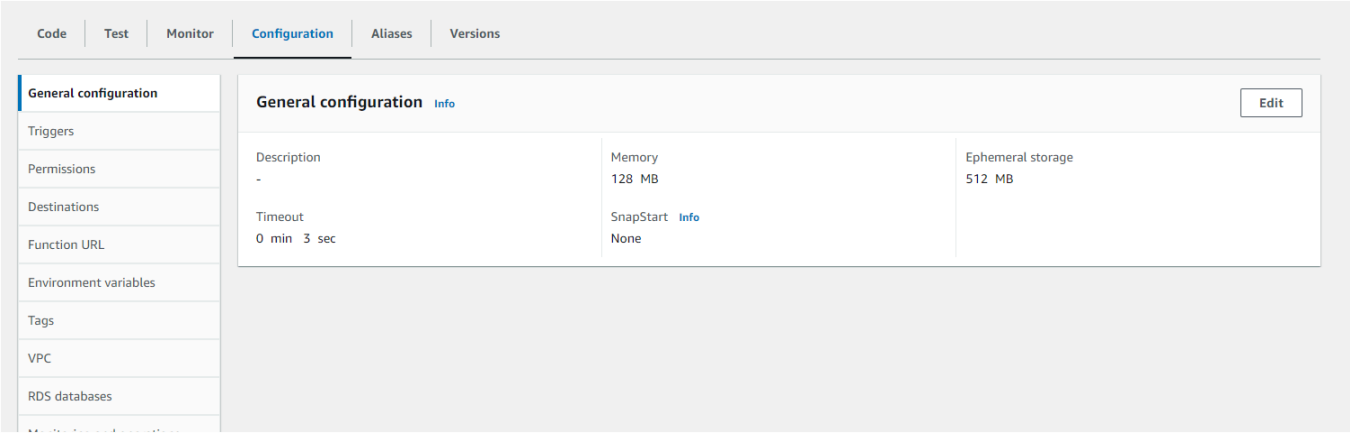



Python 3.12
Architecture Info
Choose the instruction set architecture you want for your function code.
x86_64
arm64
Permissions Info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.
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
Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.
Lambda will create an execution role named Bhushan_Lamda-role-pbjr1991, with permission to upload logs to Amazon CloudWatch Logs.
Advanced settings
Cancel
Create function
CloudShell
Feedback

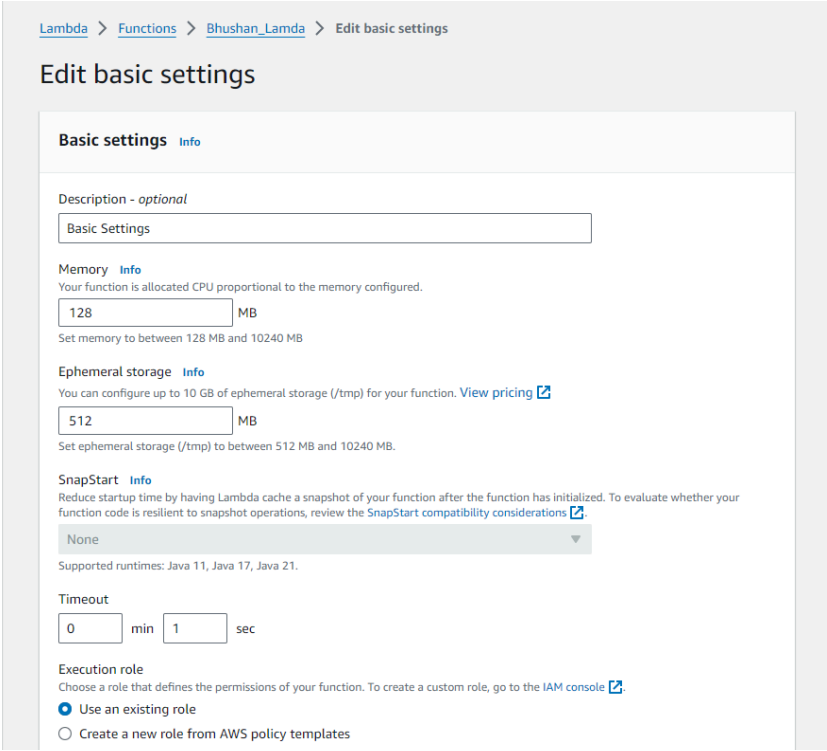

Lambda > Functions > myLambda
myLambda
Throttle
Copy ARN
Actions
Function overview Info
Export to Application Composer
Download
Diagram
Template
myLambda
Layers (0)
Add trigger
Add destination
Description
-
Last modified
4 minutes ago
Function ARN
arn:aws:lambda:eu-north-1:860015268757:function:myLambda
Function URL Info
-


Code
Test
Monitor
Configuration
Aliases
Versions
Code source Info
Upload from
File Edit Find View Go Tools Window
Test
Deploy
Go to Anything (Ctrl-P)
lambda_function
Environment Vari
Environment
myLambda
lambda_function.py
import json
def lambda_handler(event, context):
# TODO implement
return {
'statusCode': 200,
'body': json.dumps('Hello from Lambda!')
}

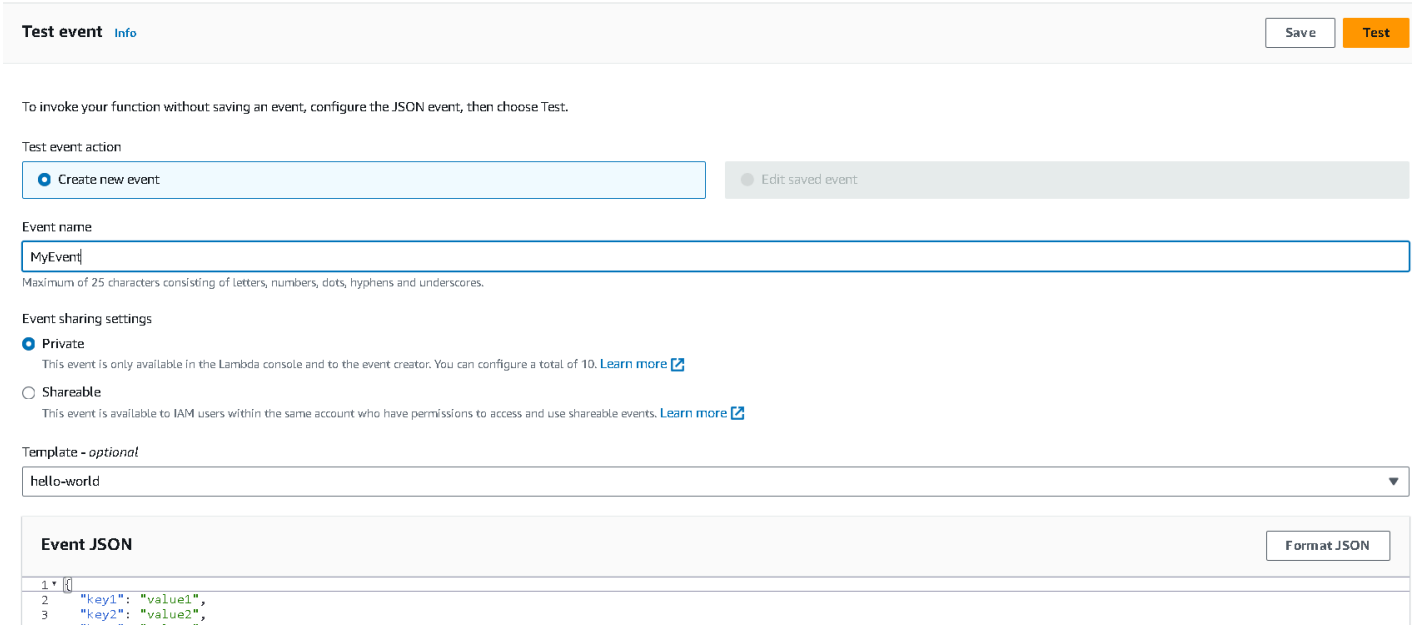
So See or Edit the basic settings go to configuration then click on edit general setting.

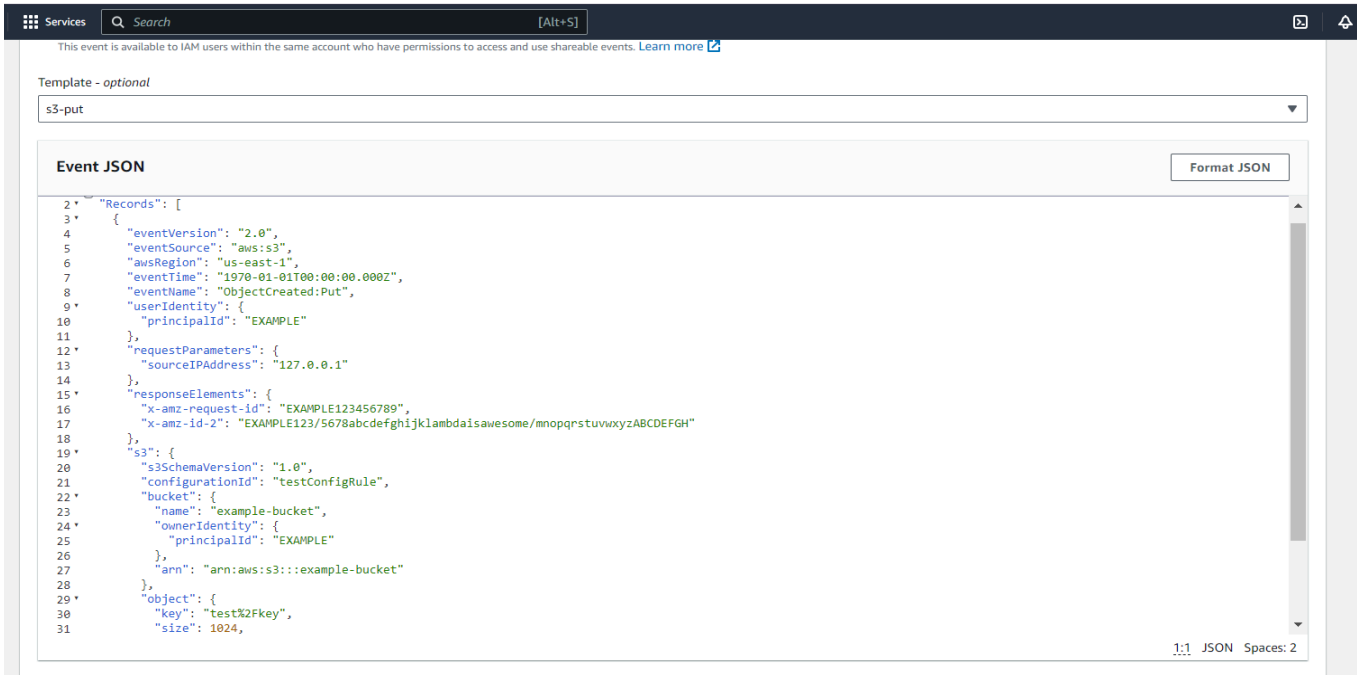


Here, you can enter a description and change Memory and Timeout. I've changed the Timeout period to 1 sec since that is sufficient for now.

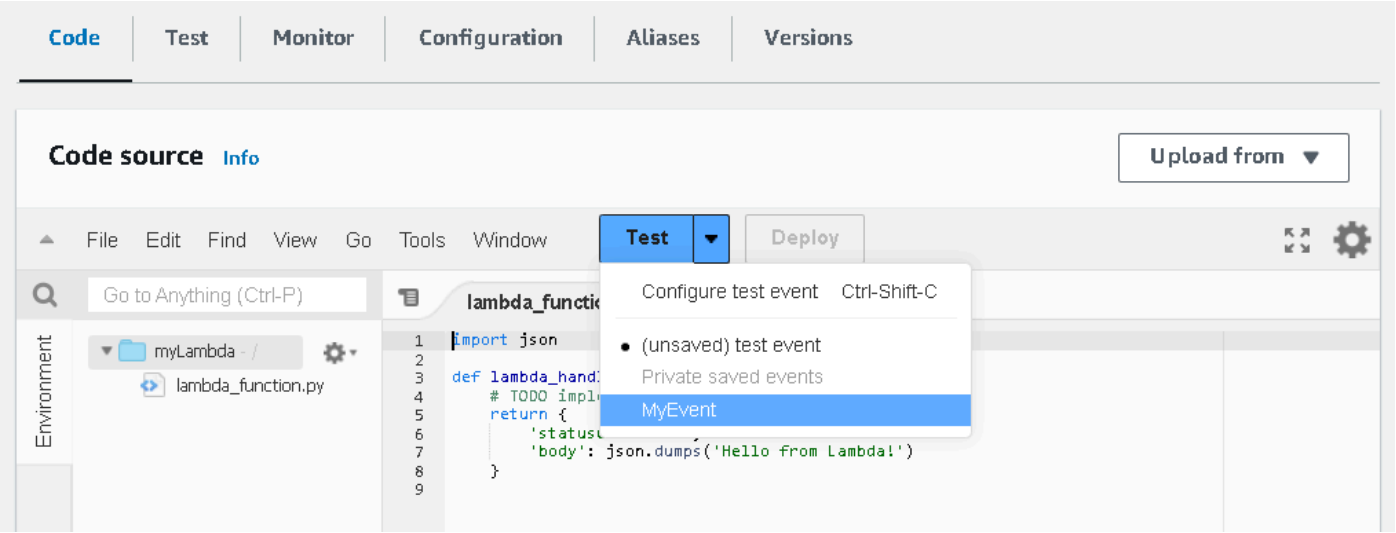


Step 5: Now Click on the Test tab then select Create a new event, give a name to the event and select Event Sharing to private, and select s3 put template.

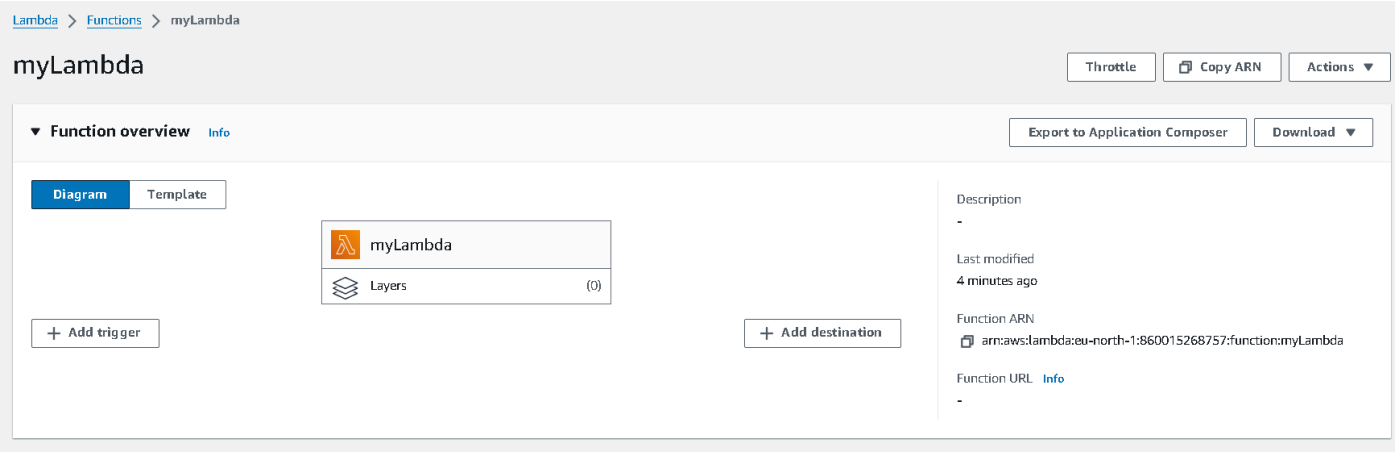




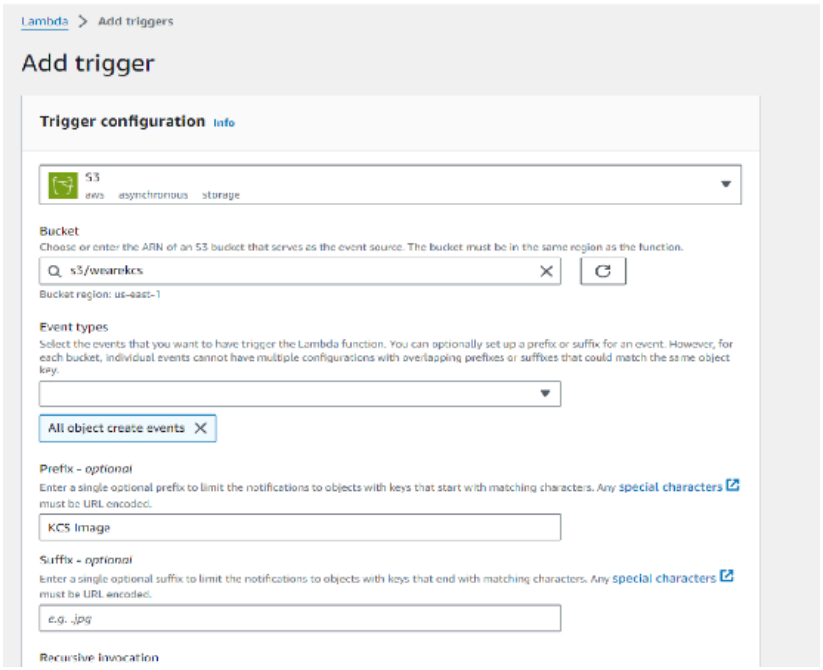
Step 6: Now In Code section select the created event from the dropdown .

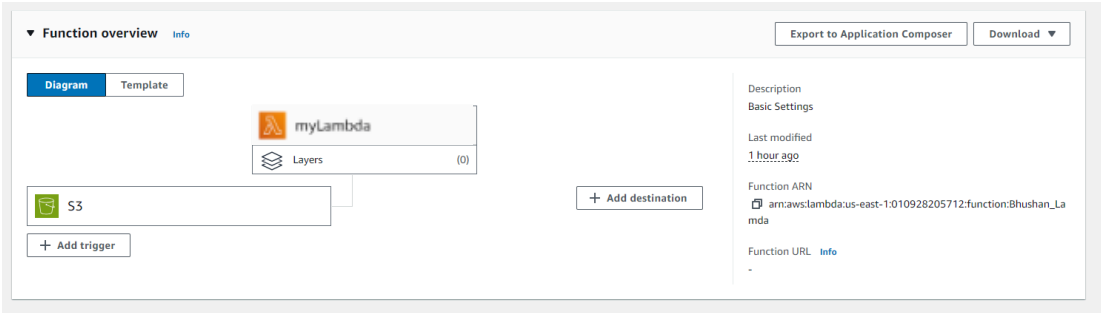


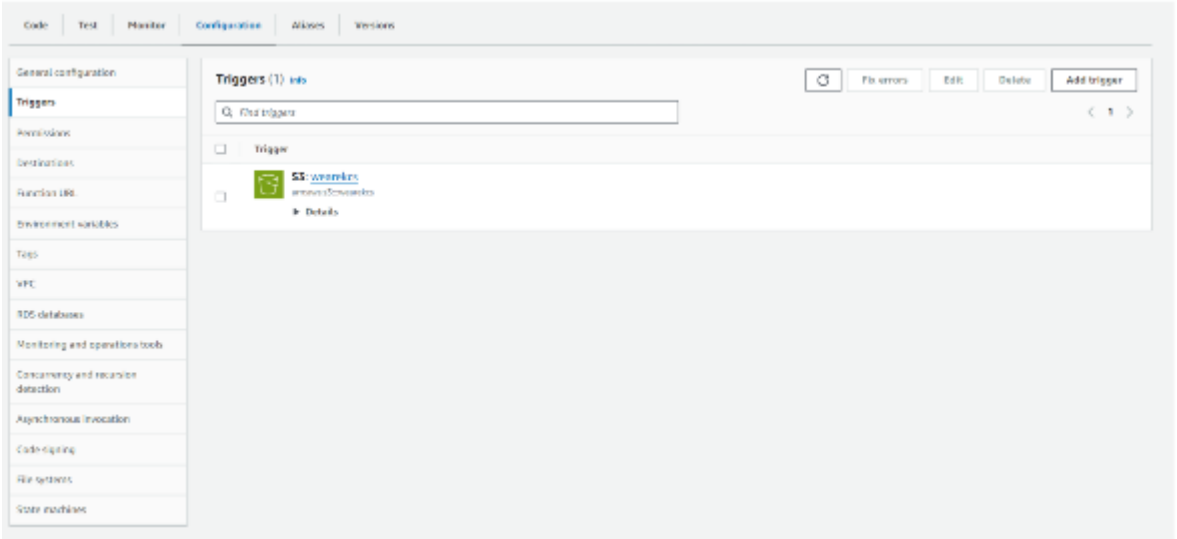
Step 7: Now In the Lambda function click on add tigger.



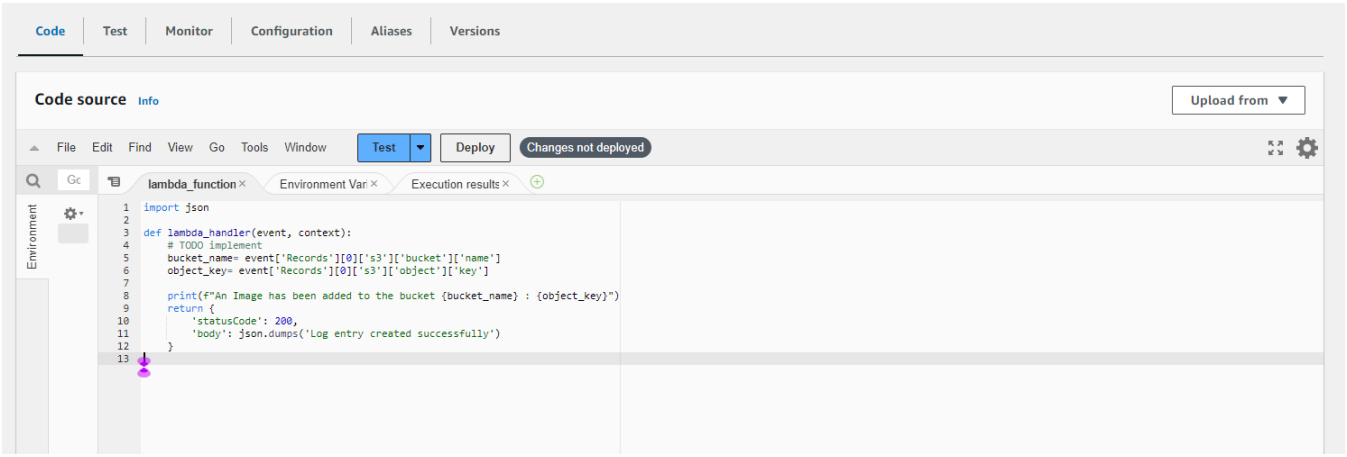
Now select the source as S3 then select the bucket name from the dropdown, keep other things to default and also you can add prefix to image.


Lambda > Add triggers
Add trigger
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/wearnkcs
Bucket region: us-east-1
Event types
Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.
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.
KCS 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

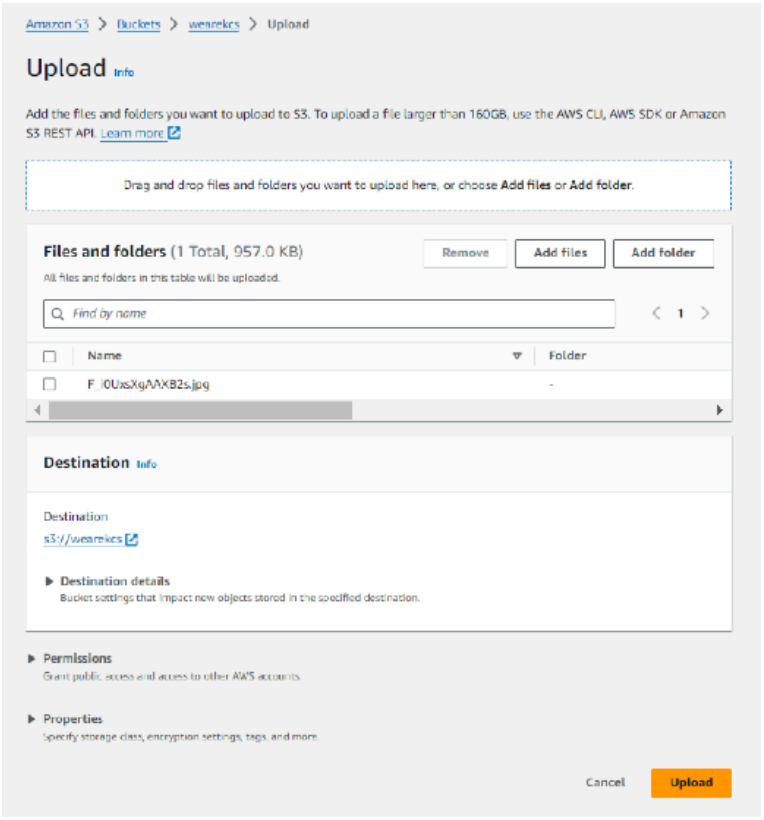

Function overview Info
Export to Application Composer
Download
Diagram
Template
myLambda
Layers (0)
S3
Add trigger
Add destination
Description
Basic Settings
Last modified
1 hour ago
Function ARN
arn:aws:lambda:us-east-1:010928205712:function:Bhushan_Lamda
Function URL Info
-


Code
Test
Monitor
Configuration
Aliases
Versions
General configuration
Triggers
Permissions
Destinations
Function URL
Environment variables
Tags
VPC
RDS databases
Monitoring and operations tools
Concurrency and recursion detection
Asynchronous invocation
Code signing
File systems
State machines
Triggers (1) Info
Filter triggers
Trigger
S3: wearnkcs
Details
Edit
Delete
Add trigger

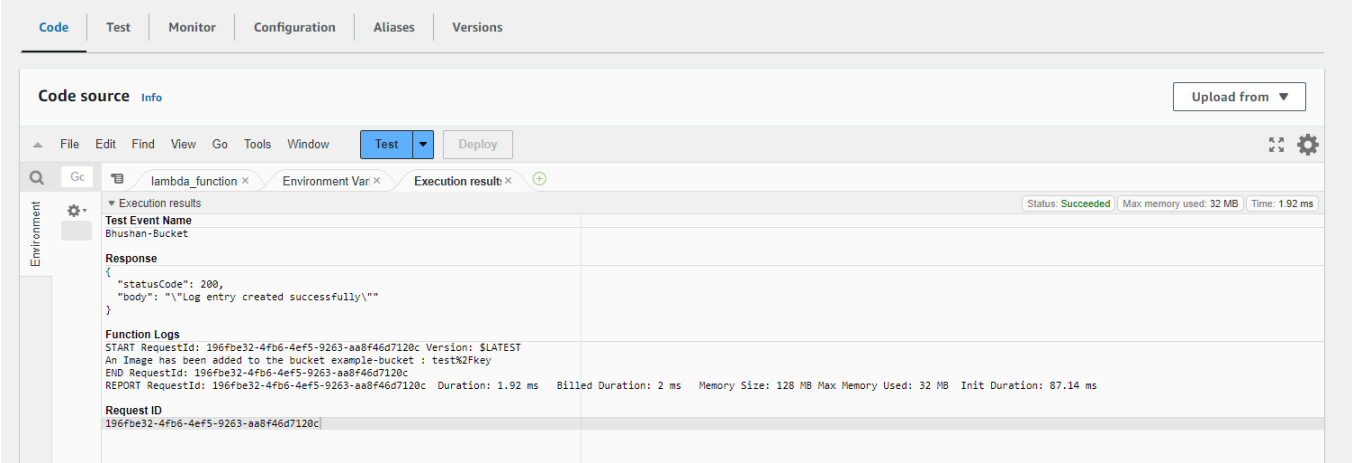
Step 8: Now Write code that logs a message like “An Image has been added” when triggered. Save the file and click on deploy.



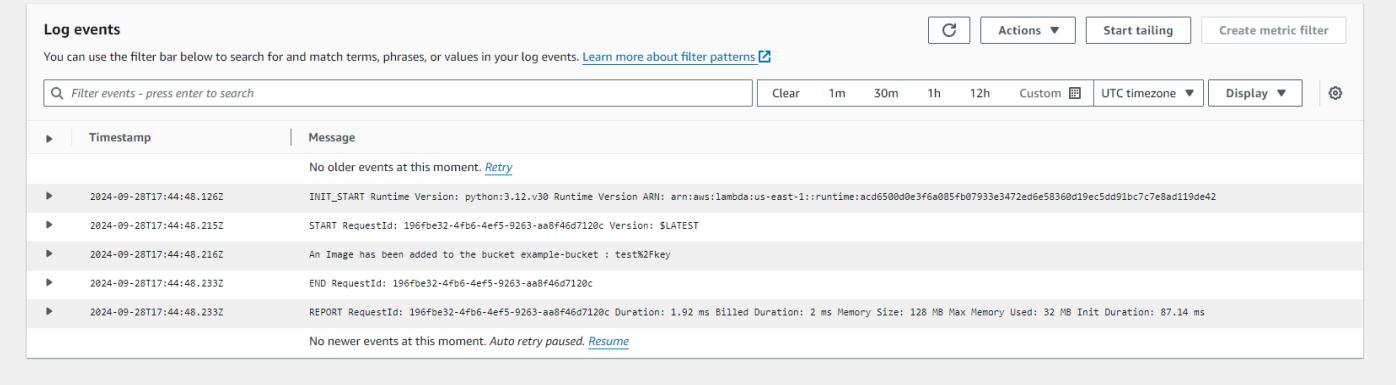
Step 9: Now upload any image to the bucket.



Step 10: Now to click on test in lambda to check whether it is giving log when image is added to S3.



Step 11: Now Lets see the log on Cloud watch.To see it go to monitor section and then click on view cloudwatch logs.



Conclusion:

In this experiment, we successfully created an AWS Lambda function that logs a message when an image is uploaded to an S3 bucket. It is important to note that we have to select S3-put template in the event otherwise code will give an error.The function was successfully triggered by S3 object uploads, validating the functionality of Lambda's event-driven architecture. This experiment demonstrated how Lambda can efficiently respond to S3 events and how to troubleshoot common issues with event structure.