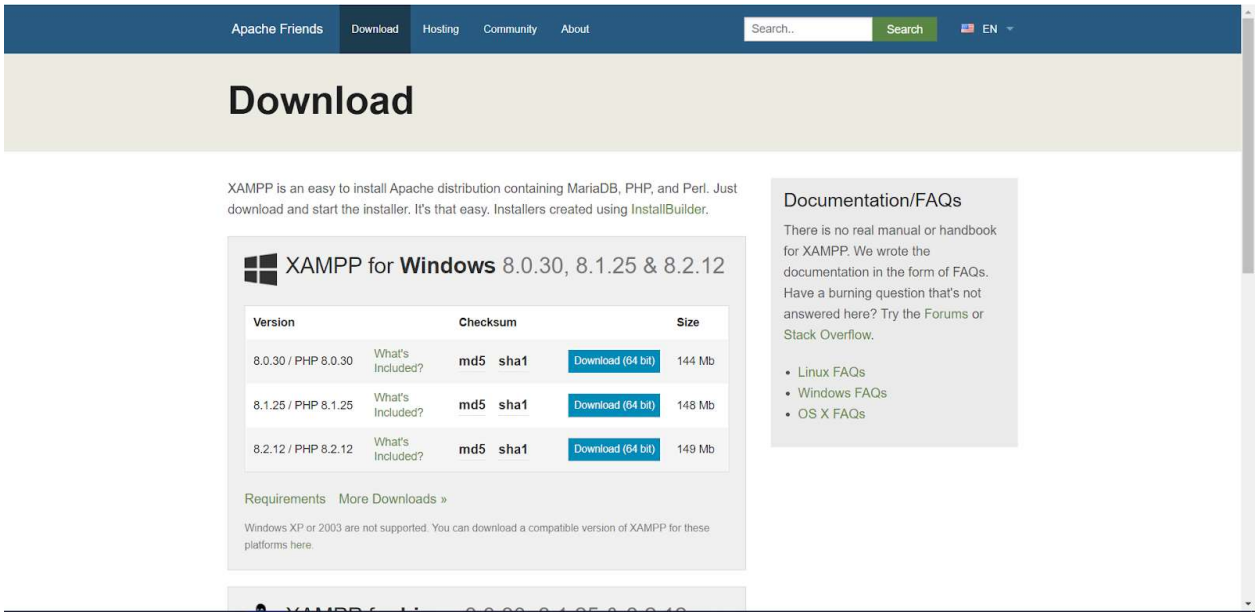
Experiment 1

Exp 1a: Prerequisites

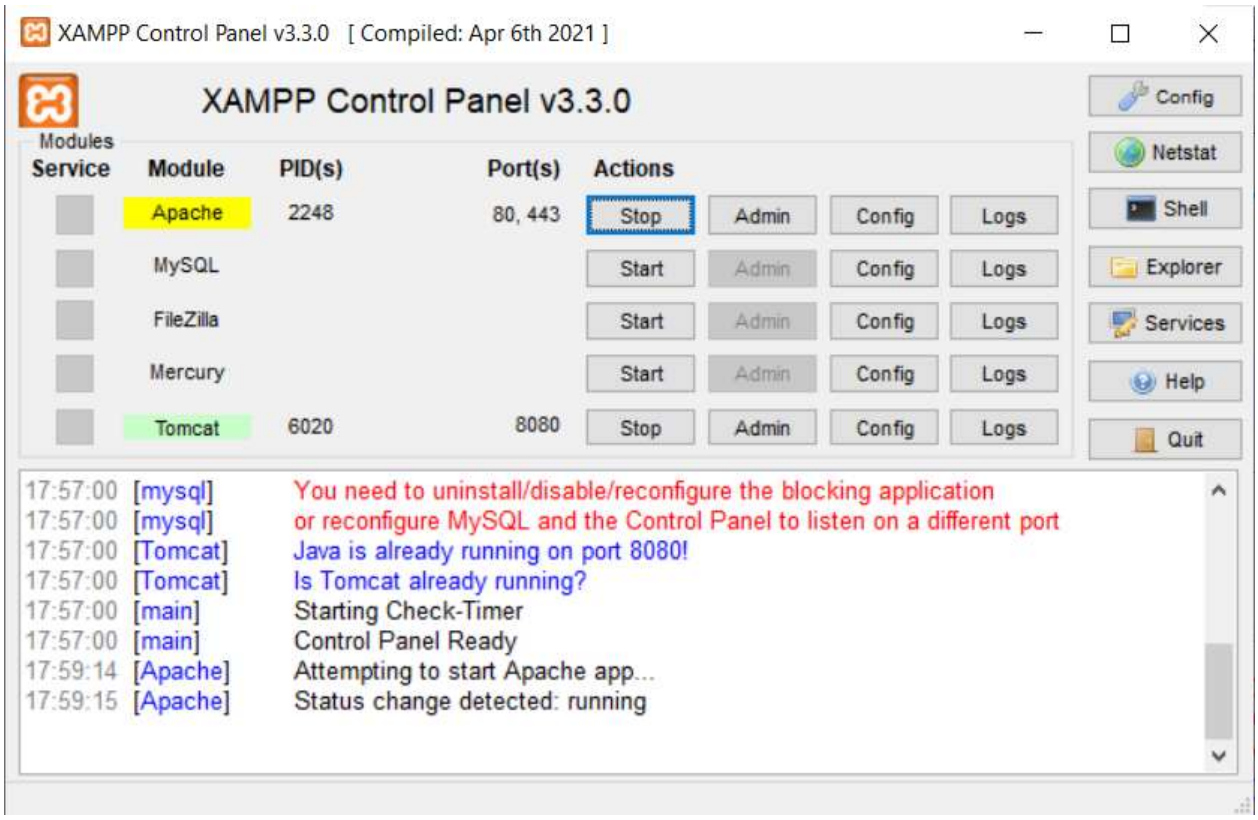
Xampp

1. Go to the official website of Xampp. https://www.apachefriends.org/download.html. Select the suitable version and complete the installation.

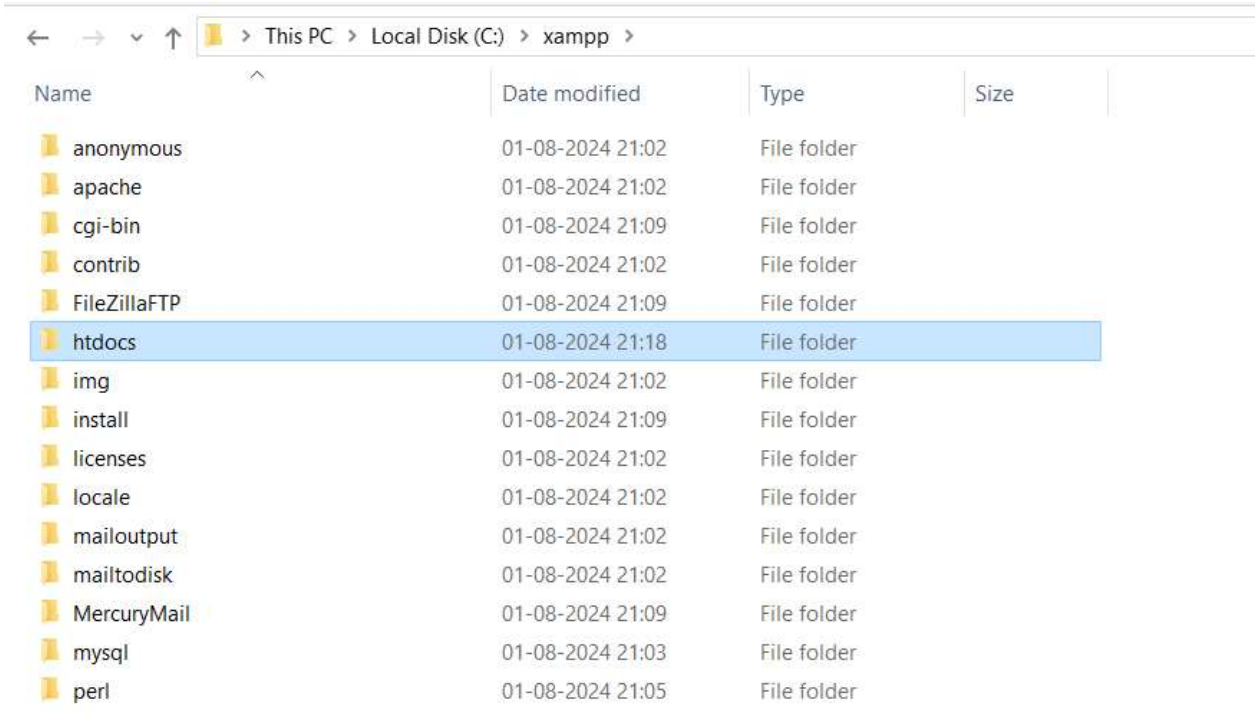


2. Once installation is complete, open the Xampp control panel. To host a php project locally we require to start the apache server.

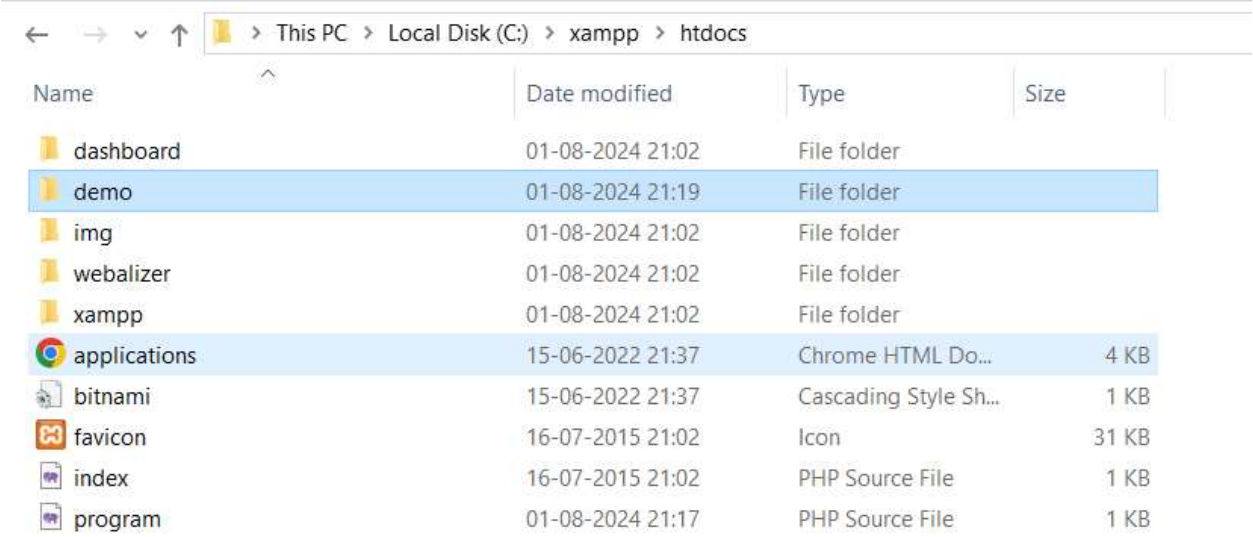
3. Click on start in front of apache and wait for the server to start.



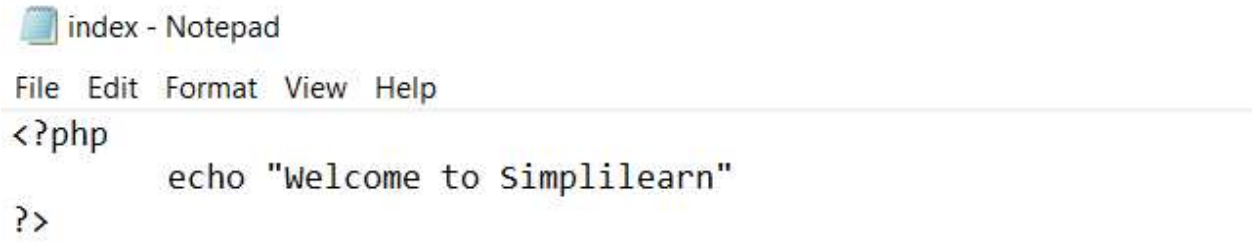
4. We can now set up the php project. In the xampp folder in C drive there will be a folder named htdocs. Every project that we want to host locally should be present in htdocs.



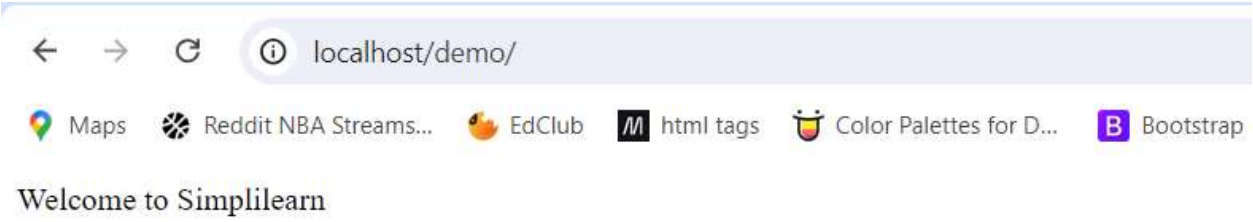
5. We will create a project named demo inside htdocs. This demo folder will contain our php source code.



6. Open the notepad and write a simple php script. Save this file inside the demo folder.

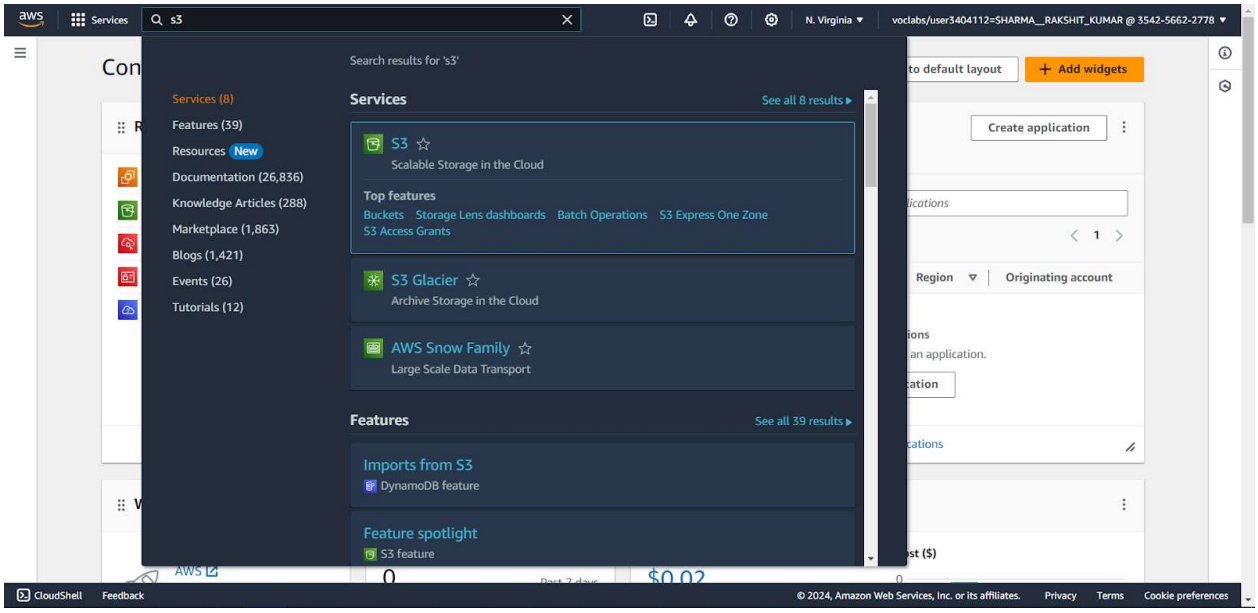


7. Go to your web browser and type “localhost/project_folder_name”, “localhost/demo” in this case. And we can see that our php script has been hosted locally using Xampp.

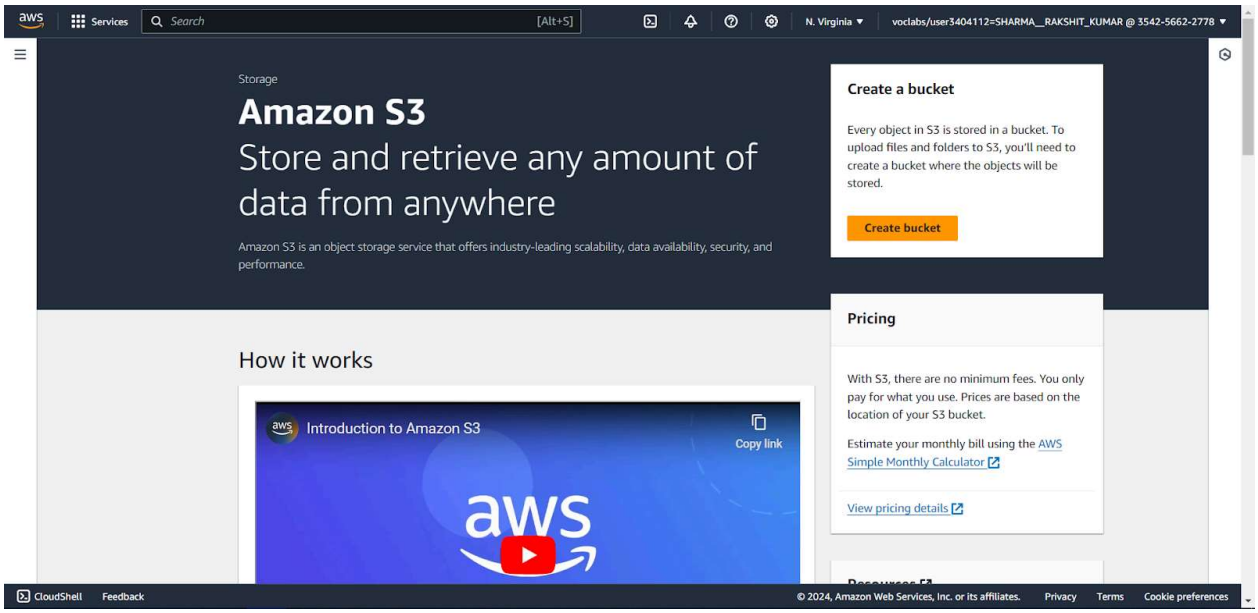


S3

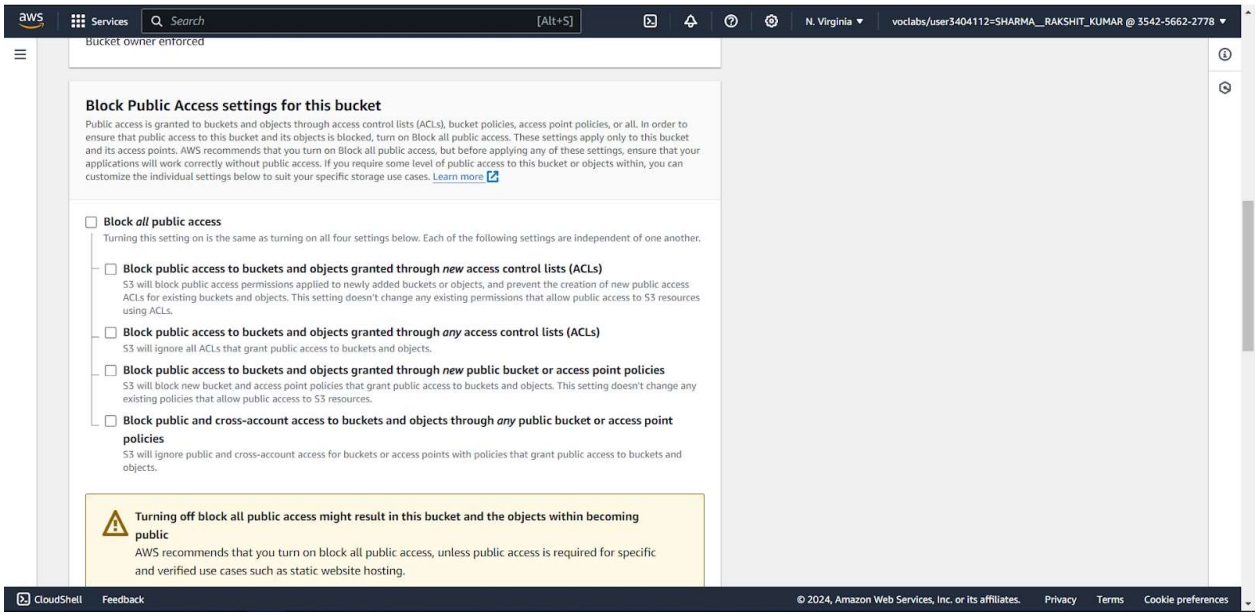
1. Go to the AWS academy lab and search for s3 service.



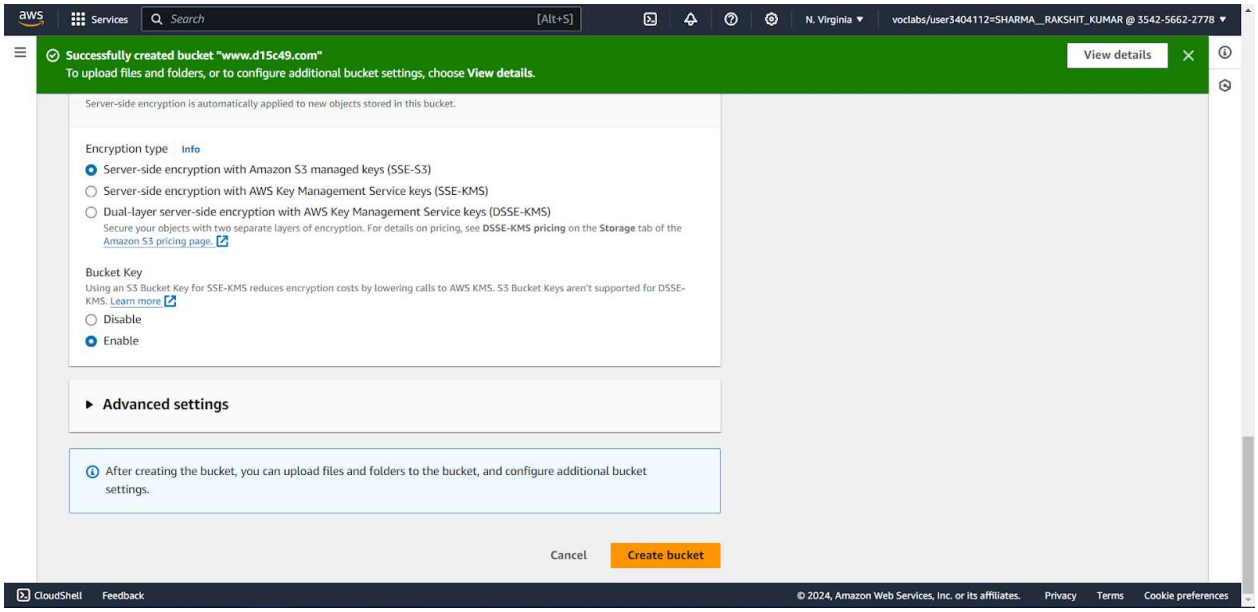
2. In the s3 service select the create bucket option.



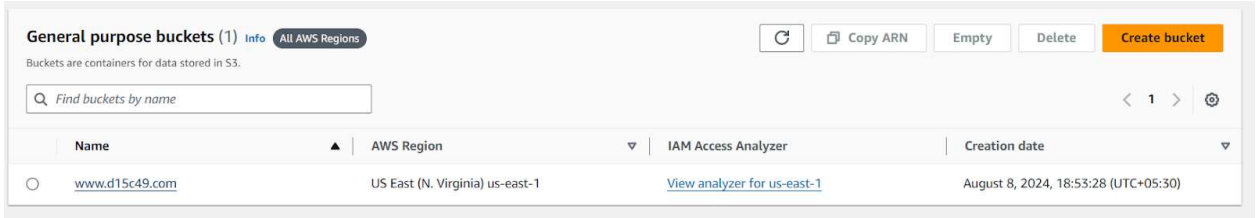
3. Select the name for bucket. Enter other details. Uncheck the block all public access option. Checking this box will give the 403 error when we try to access the website.



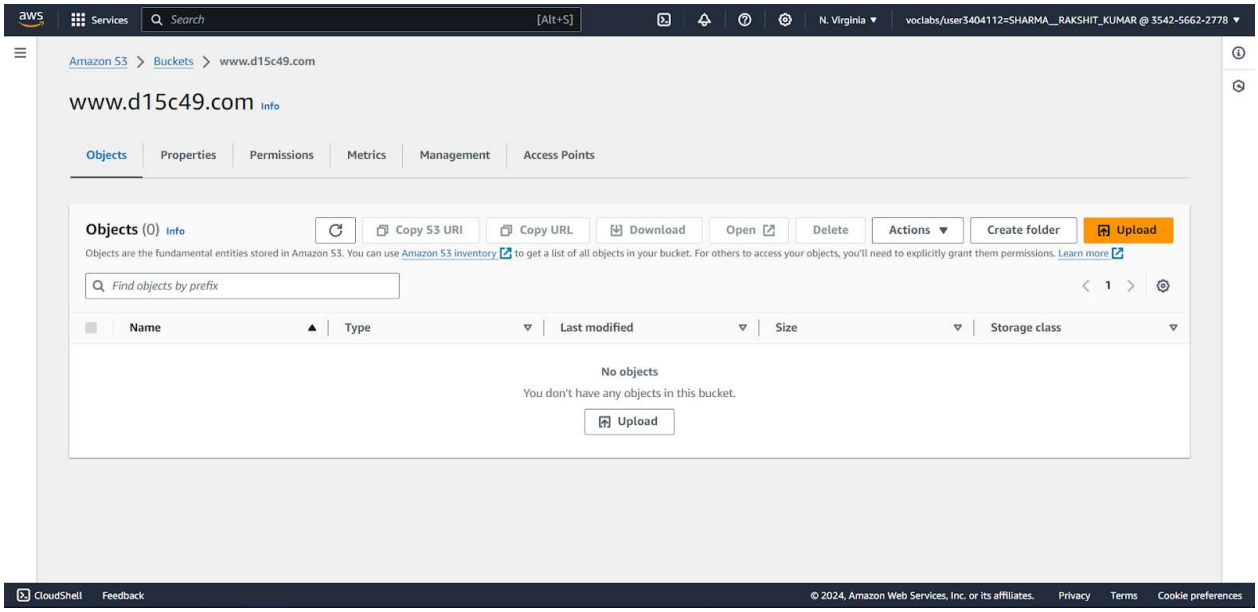
4. Use default setting for other options. Once configuration is completed, click on create bucket.



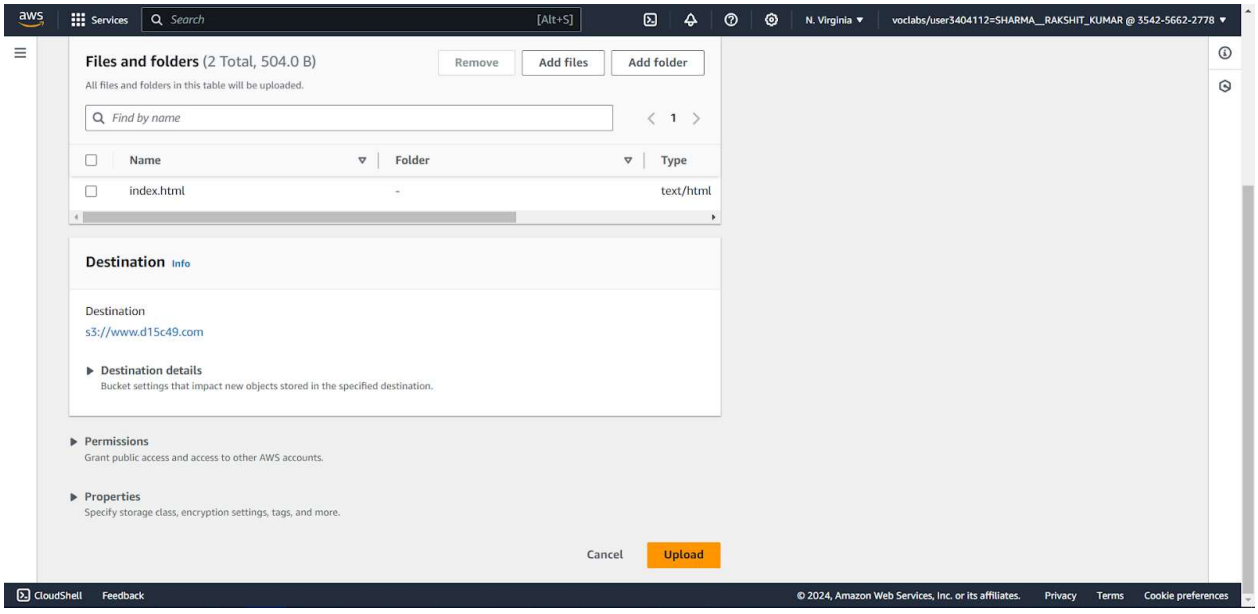
5. You can see the buckets created in General purpose buckets.



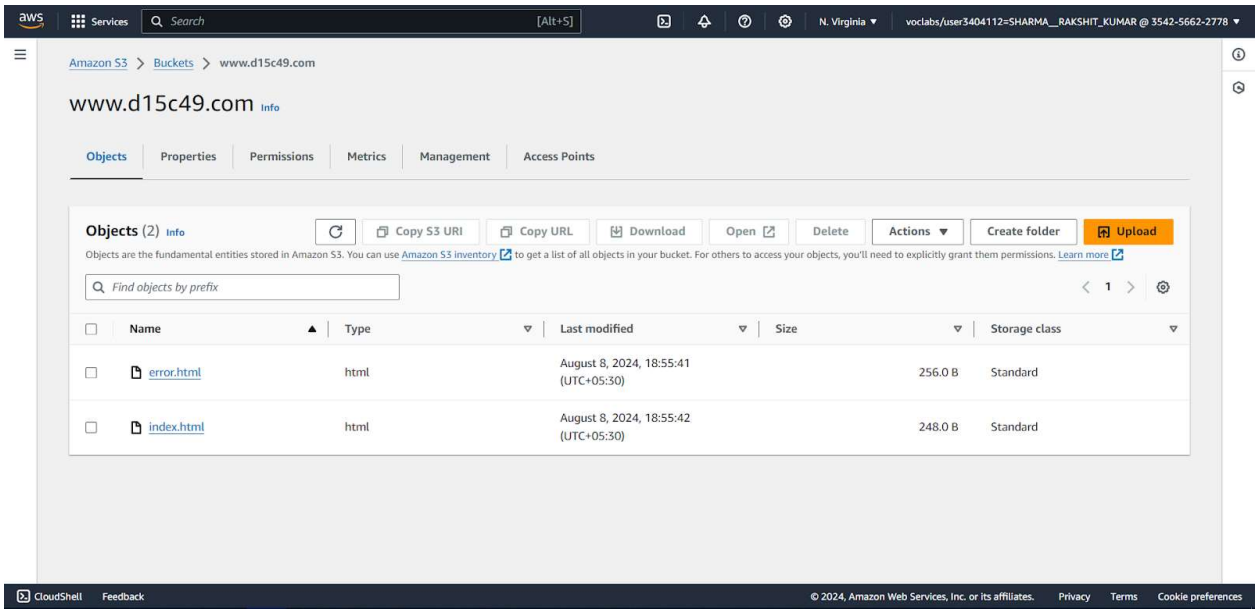
6. Open the bucket, now we will upload files in our bucket. Select the upload button



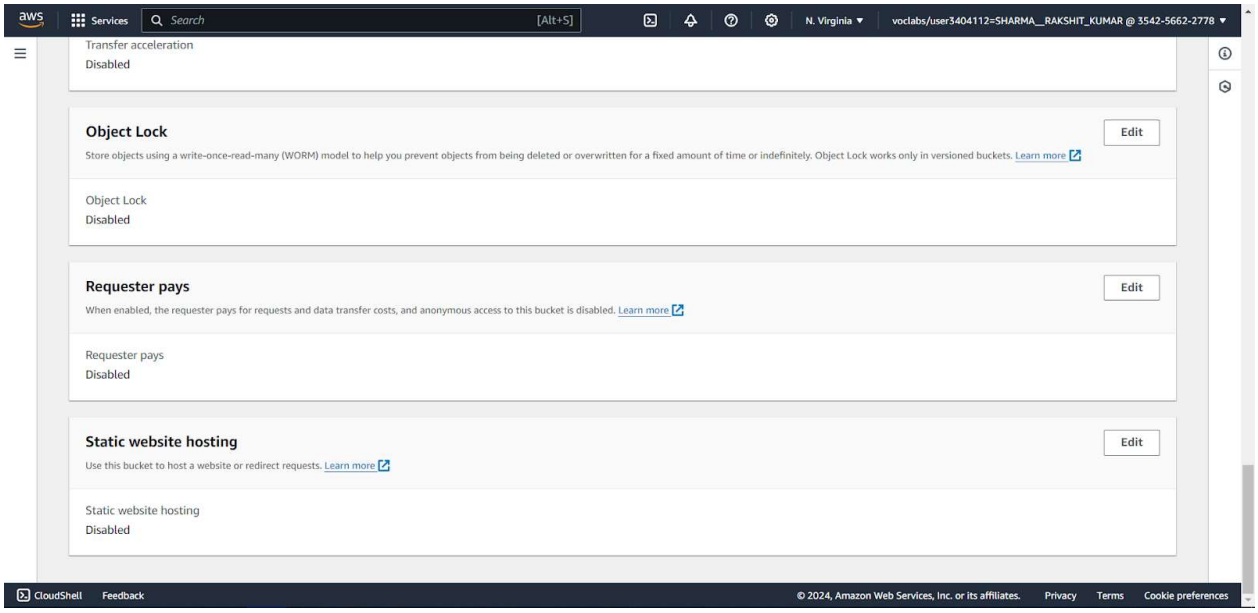
7. Select the files you want to upload in the bucket. Click on the upload button to upload files.



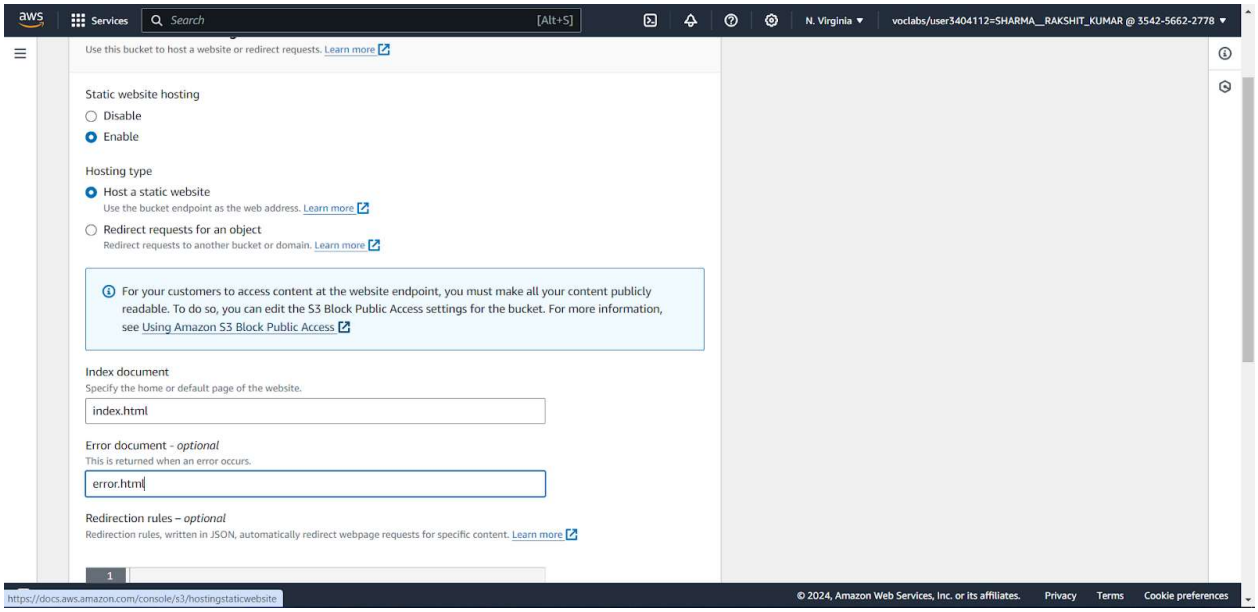
8. You can see all the files uploaded in the bucket. Now go to the properties section to enable the static hosting option for the bucket.



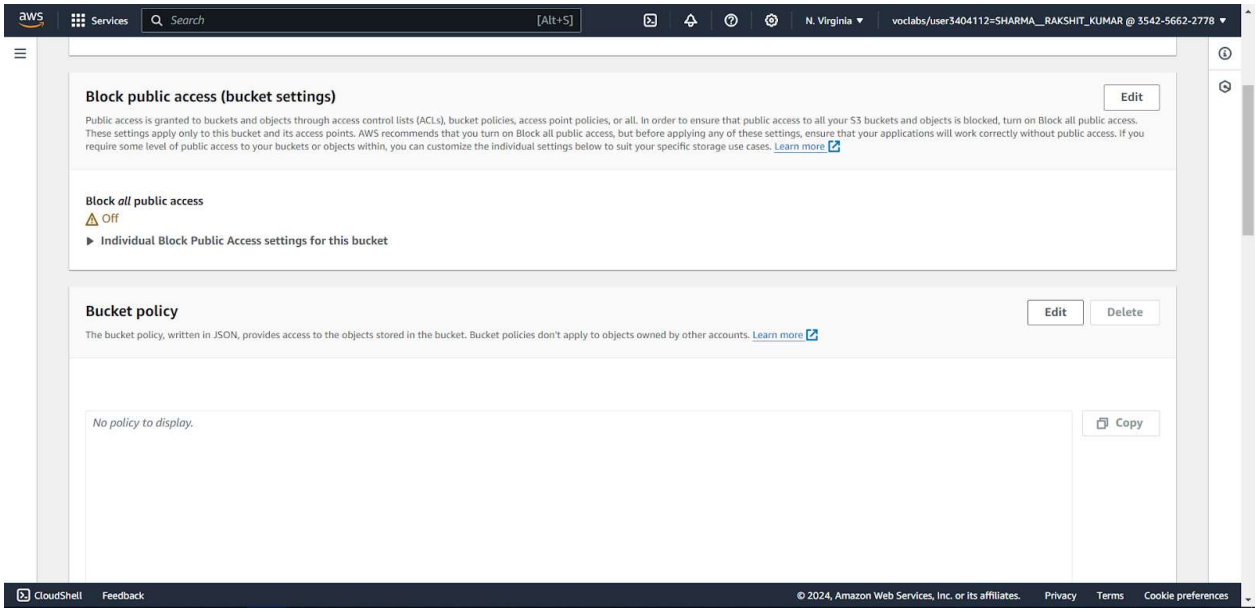
9. Edit the static website hosting setting.



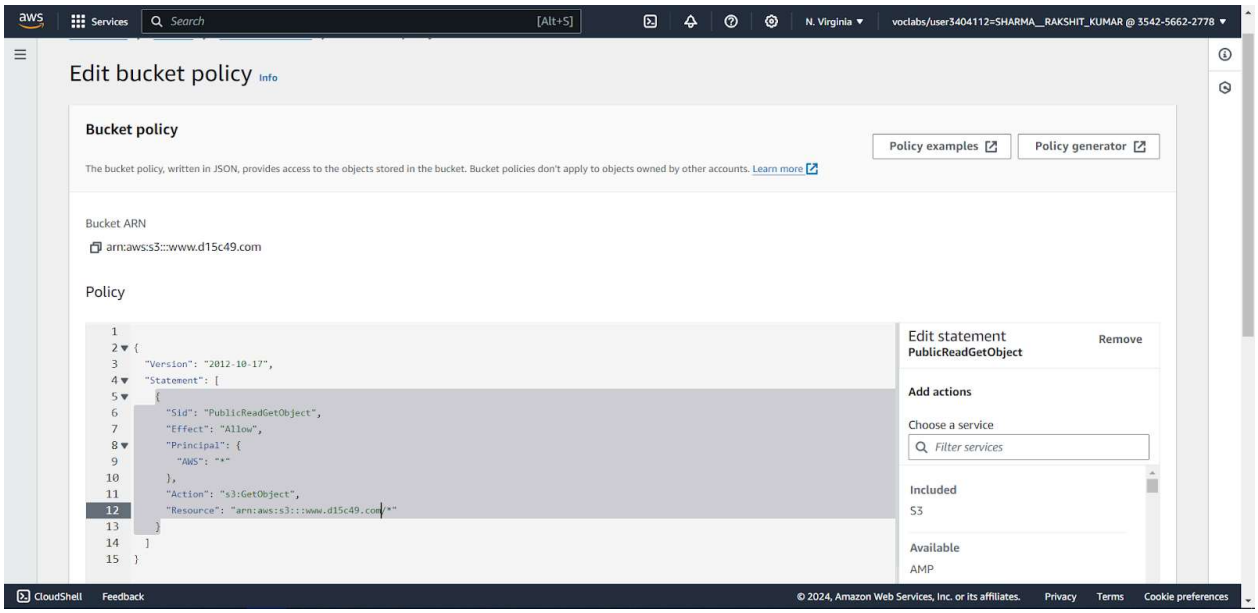
10. Enable the static website hosting, select the index files and error files(optional). These will be the files you have uploaded in the bucket



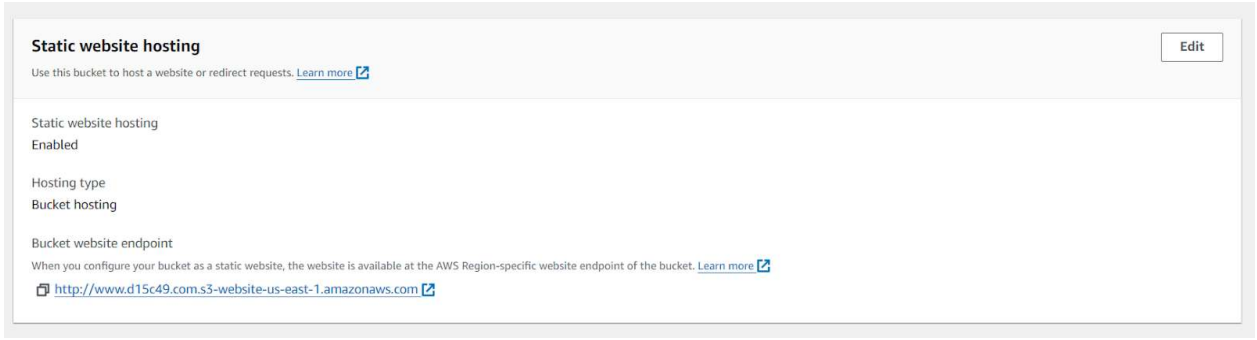
11. Now we need to change the bucket policy in permissions. Ensure that block public access is off. Edit the bucket policy.



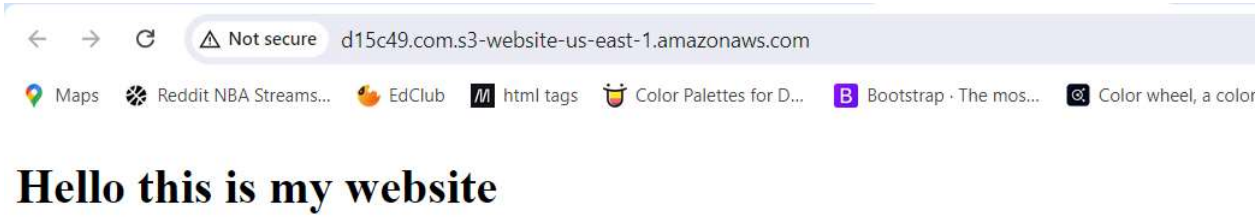
12. You can see examples of bucket policy. Use the appropriate bucket policy and enter your bucket name in it.



13. Once the bucket policy is added, you can see the link for website in static website hosting.

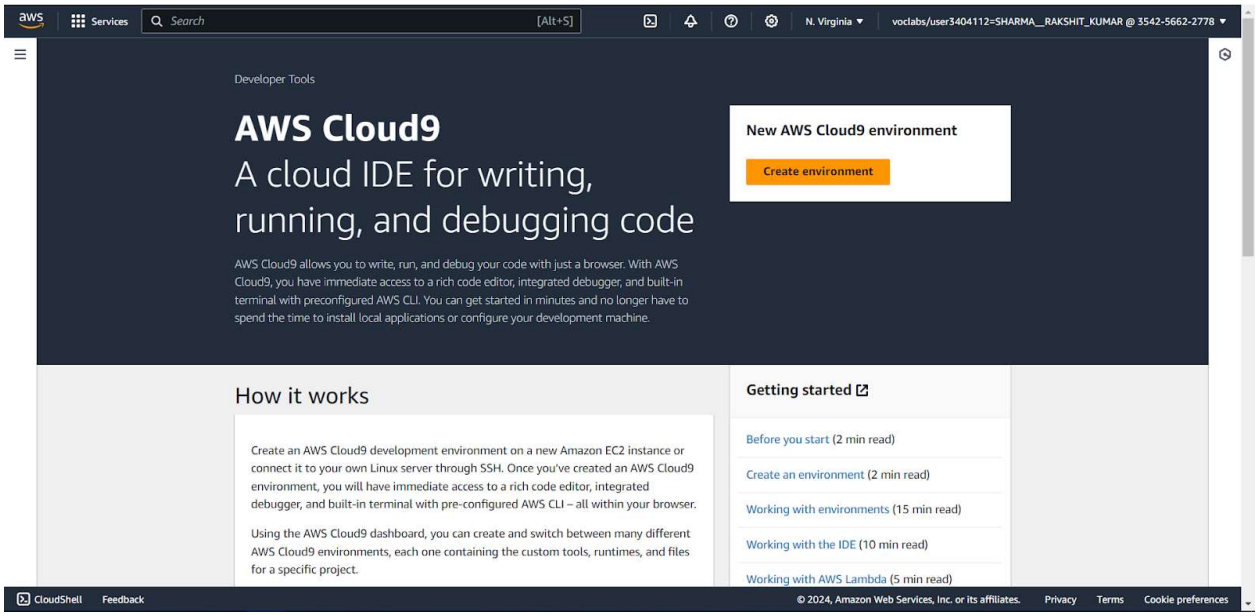


14. On clicking the link you will be redirected to the hosted website.

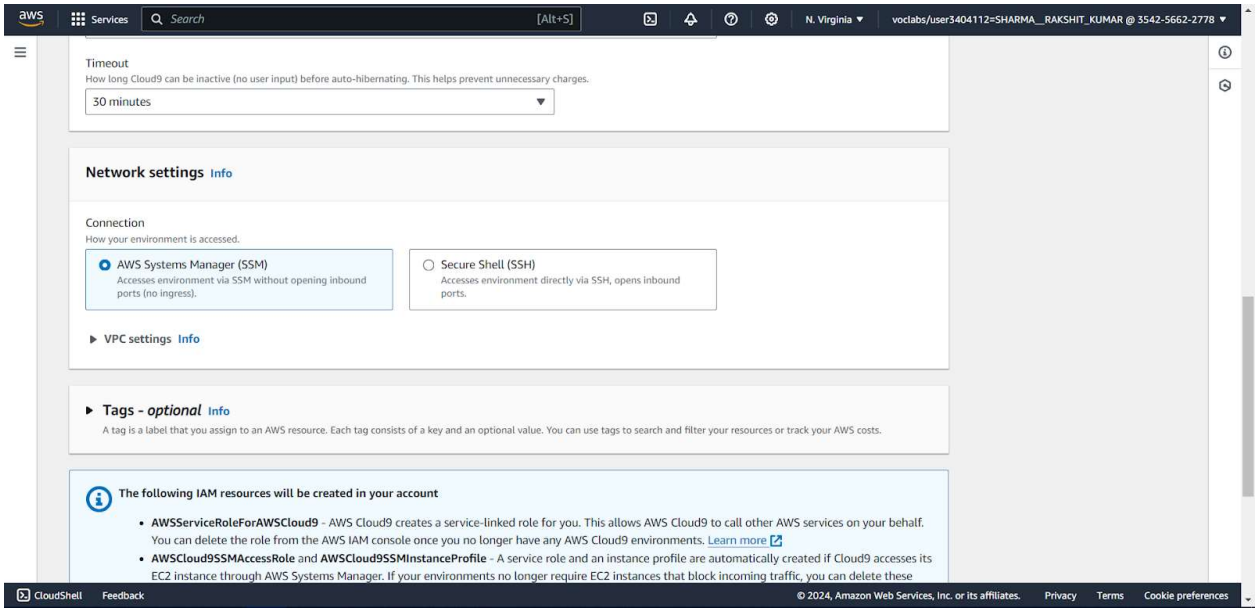


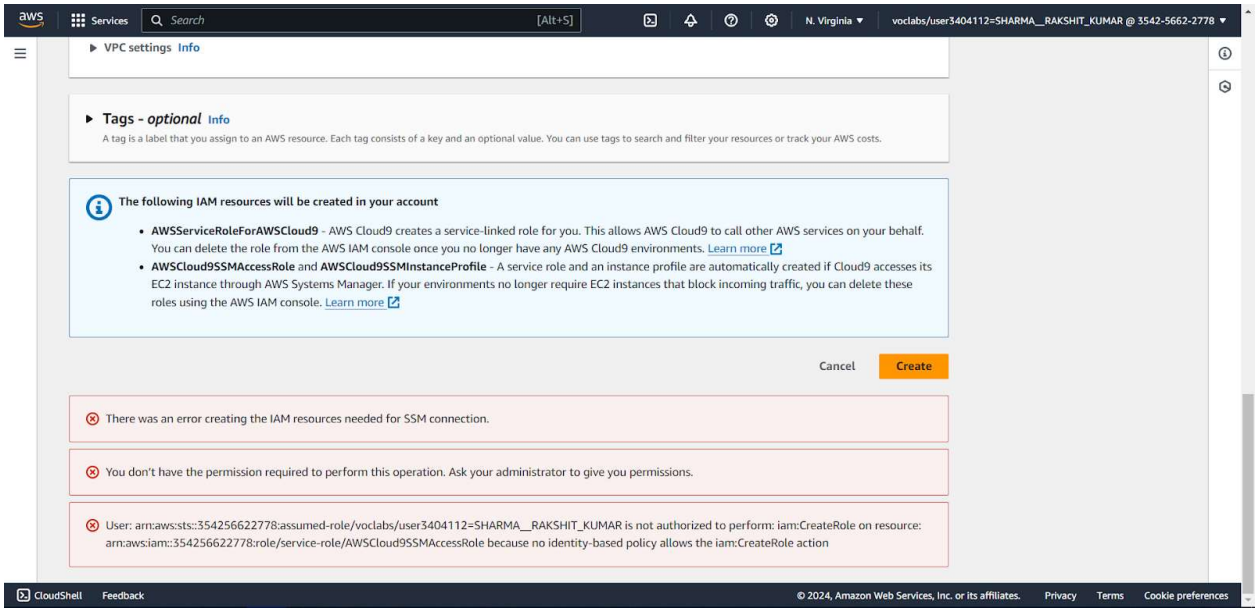
Aim: To understand the benefits of Cloud Infrastructure and Setup AWS Cloud9 IDE, Launch AWS Cloud9 IDE and Perform Collaboration Demonstration.

1. Open the AWS account and search for Cloud9. Click on create environment.

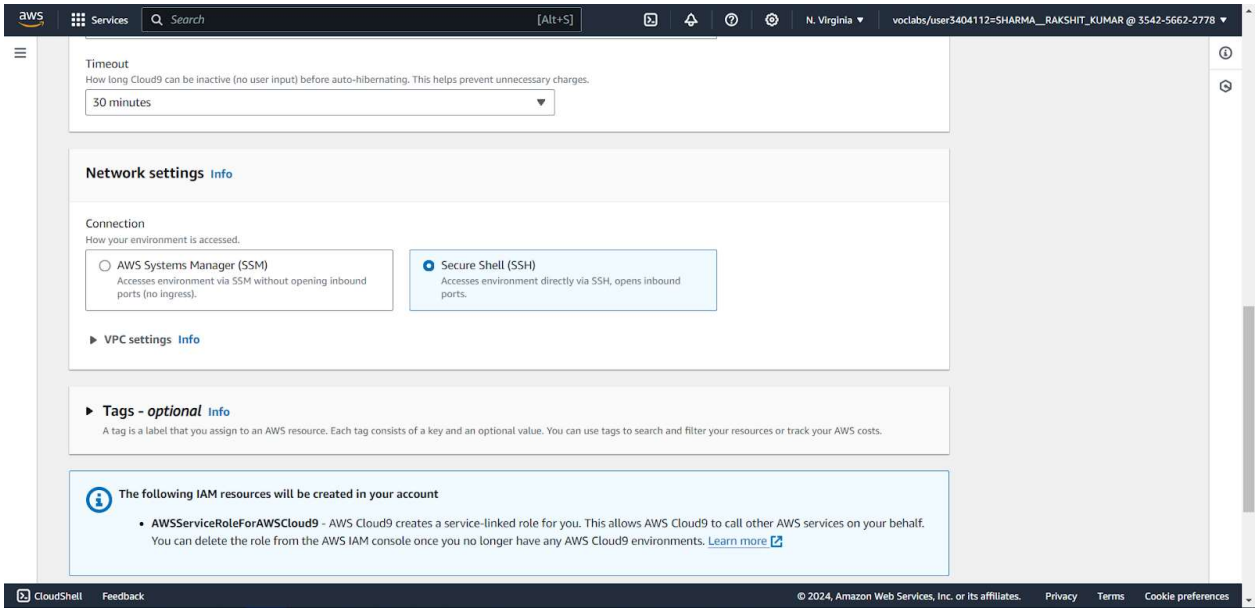


2. Enter the name and other required configuration for creating an environment. In network settings, using the AWS system manager gives an error while creating the environment. It states there was an error creating IAM resources needed for SSM.

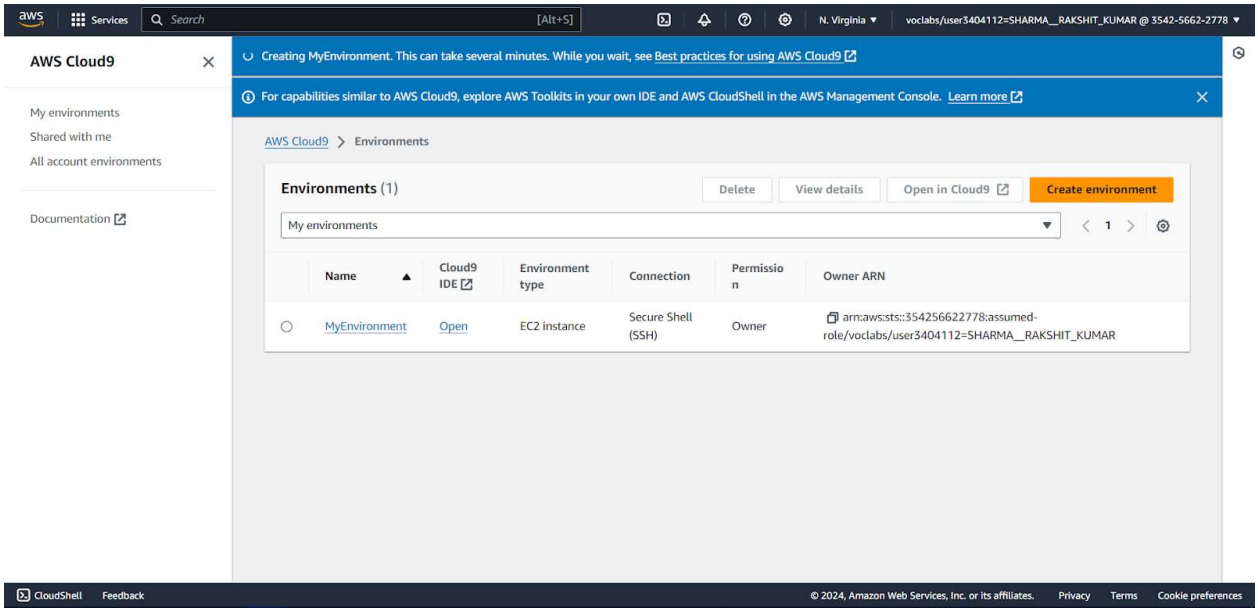




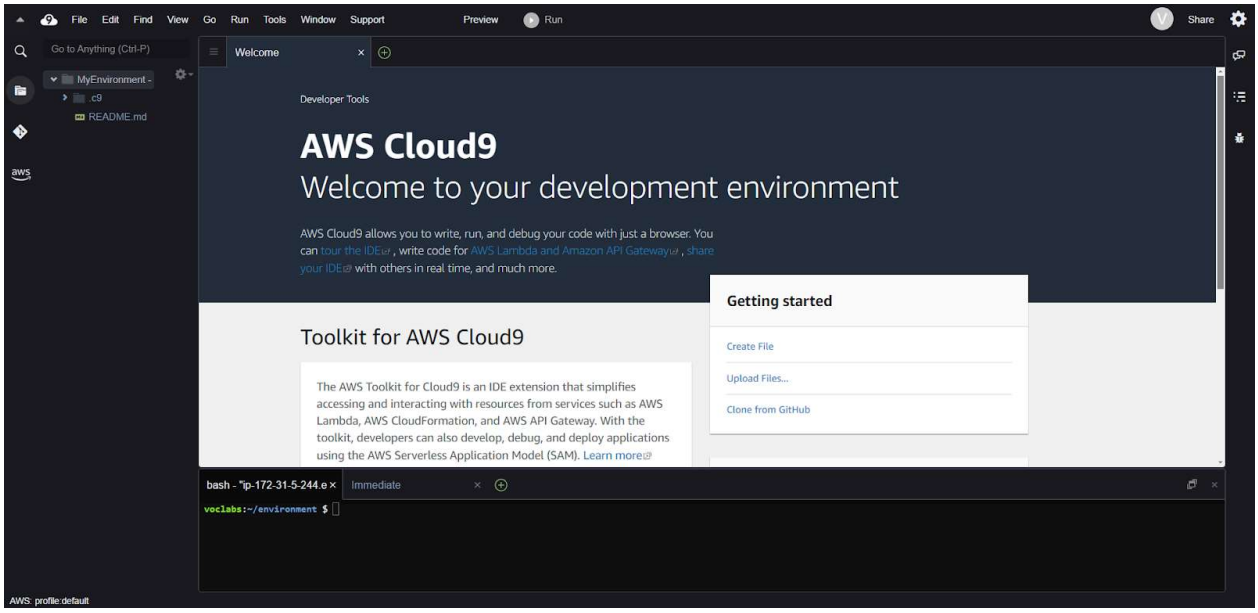
3. Use the Secure Shell option in Network settings.



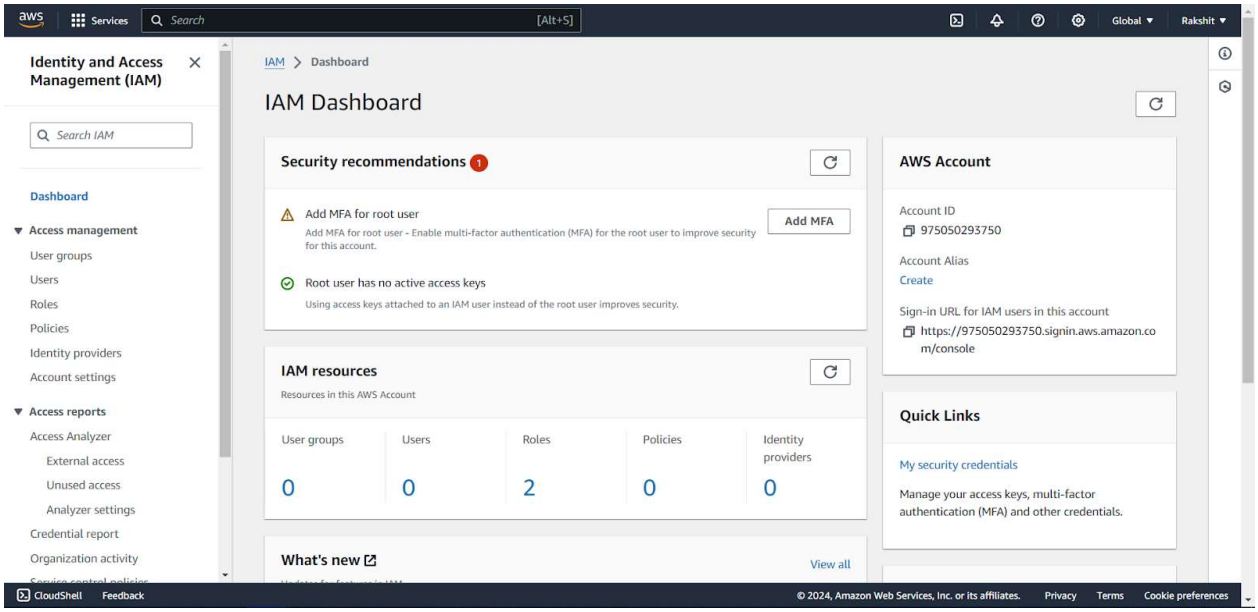
4. Once the configuration is complete, click on create environment to create a Cloud9 environment.



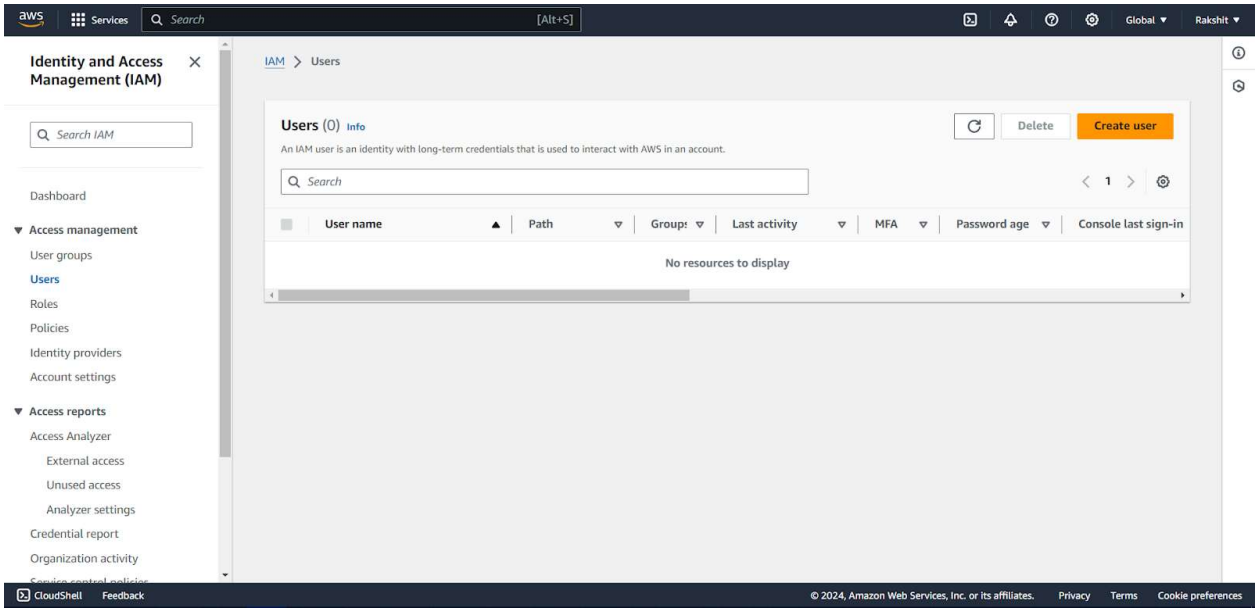
5. Click on the environment name to open the created Cloud9 Environment.



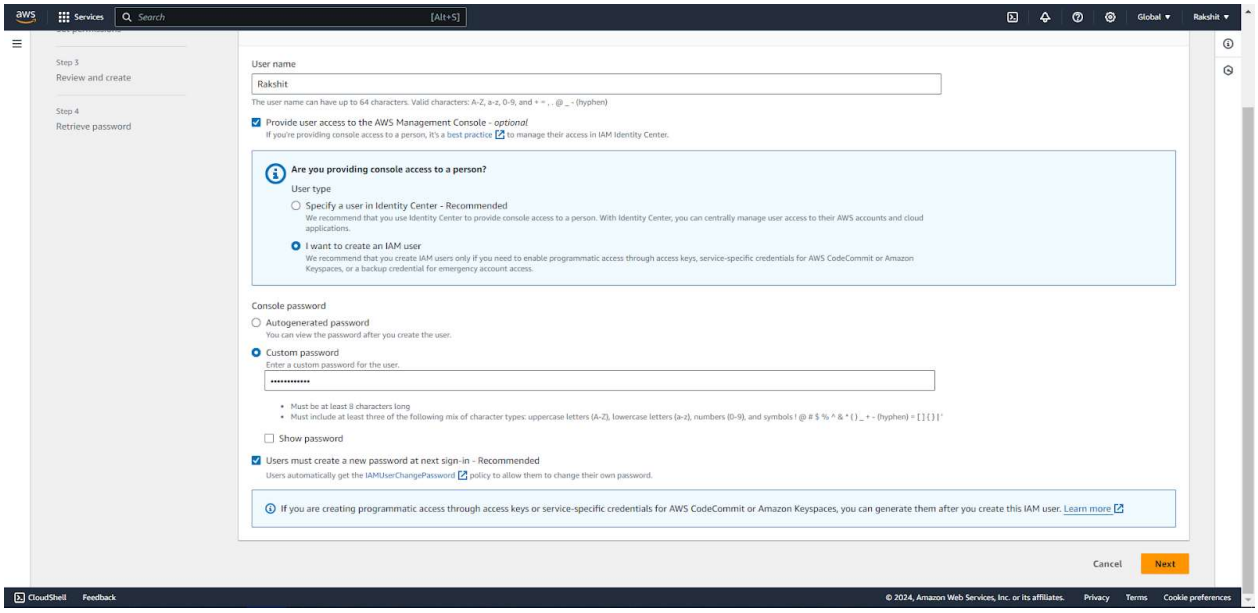
6. Open the aws account and search for IAM service.



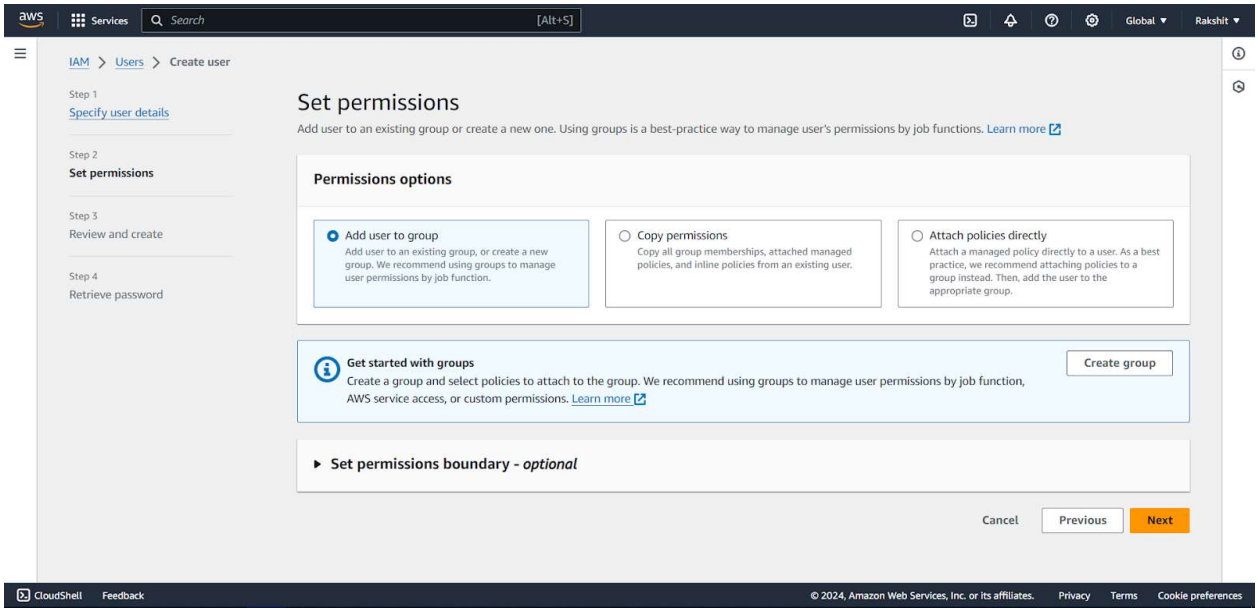
7. Go to the users tab. Click on create user to create a new user.



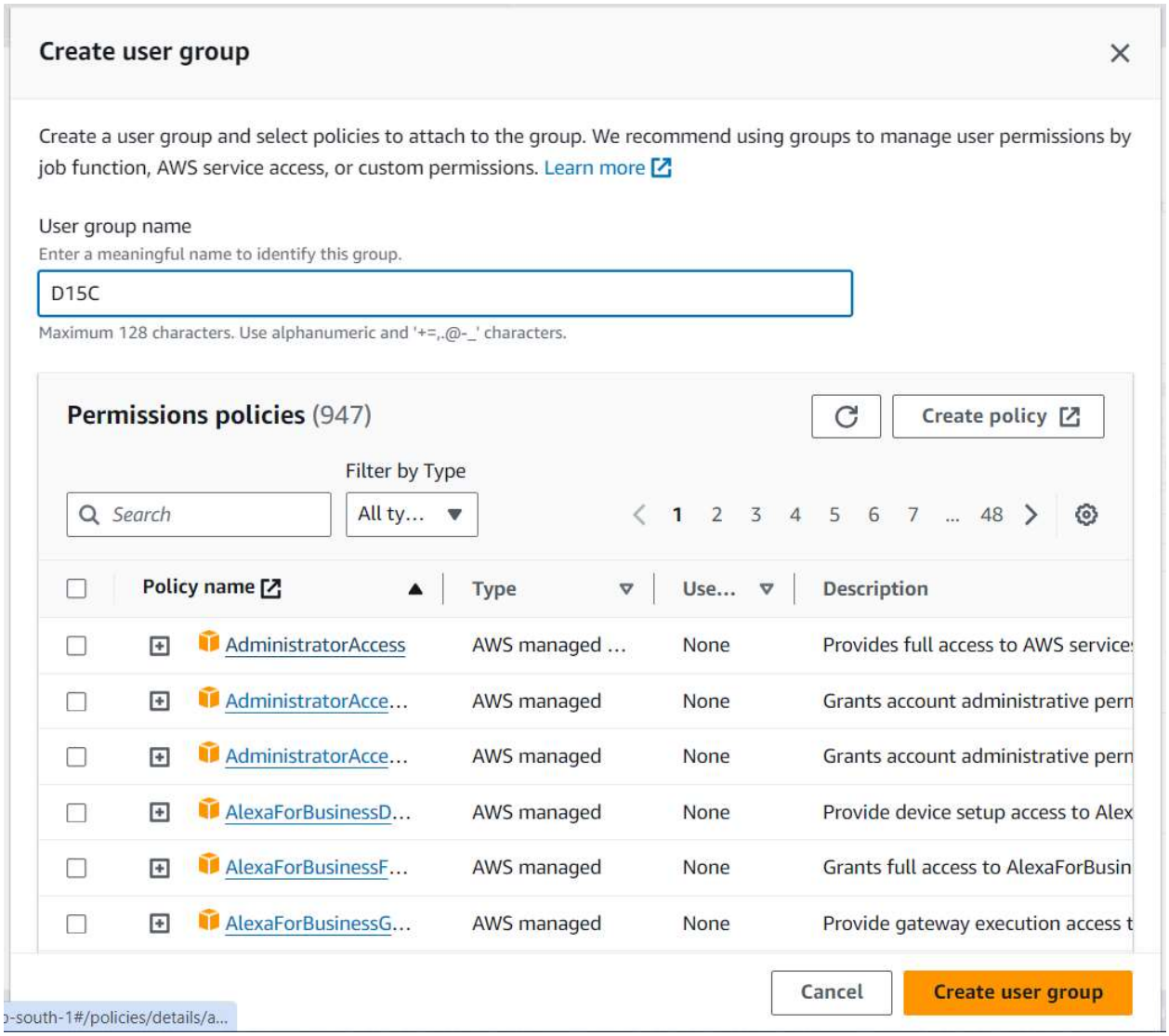
8. Give the user name, select the “provide user access” checkbox. Also select the option “I want to create an IAM user”. Otherwise we will have to enable the Identity center and specify a user there.



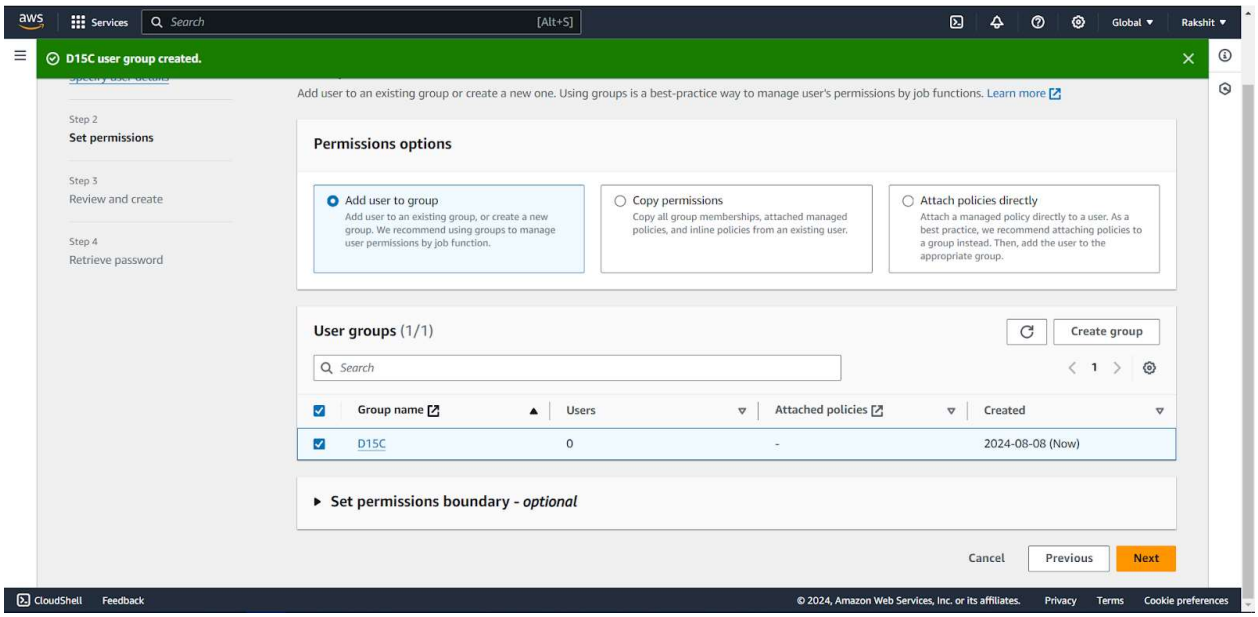
9. Next click on add user to group. If you do not have a existing group, select create group.



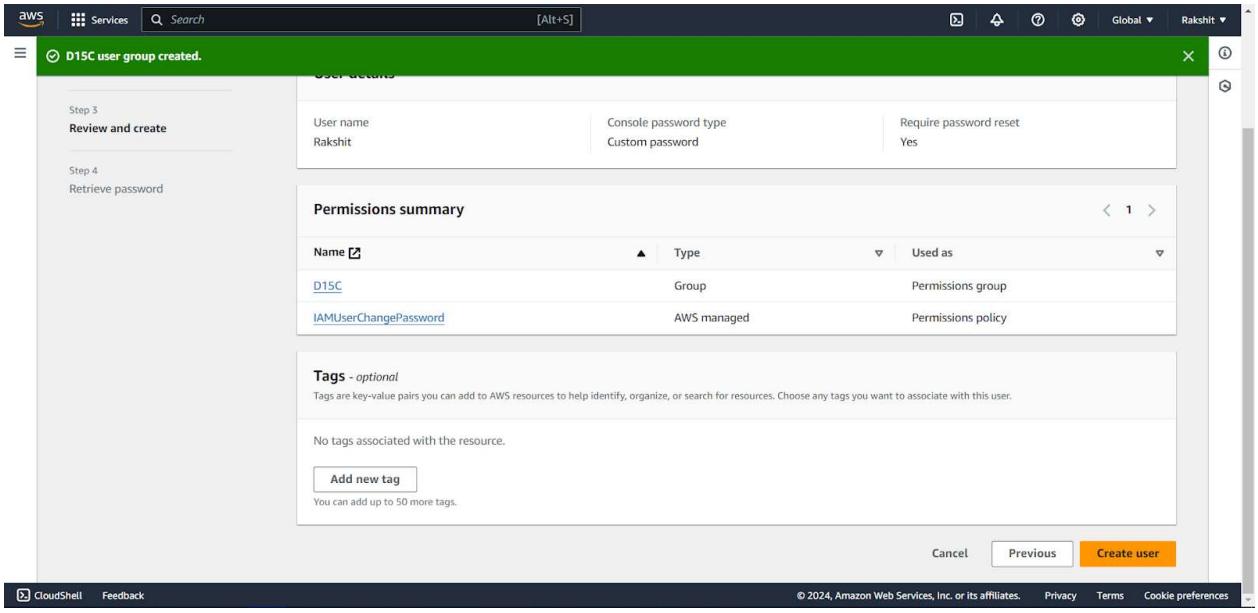
10. Give the group name and policies if required, and create a group.

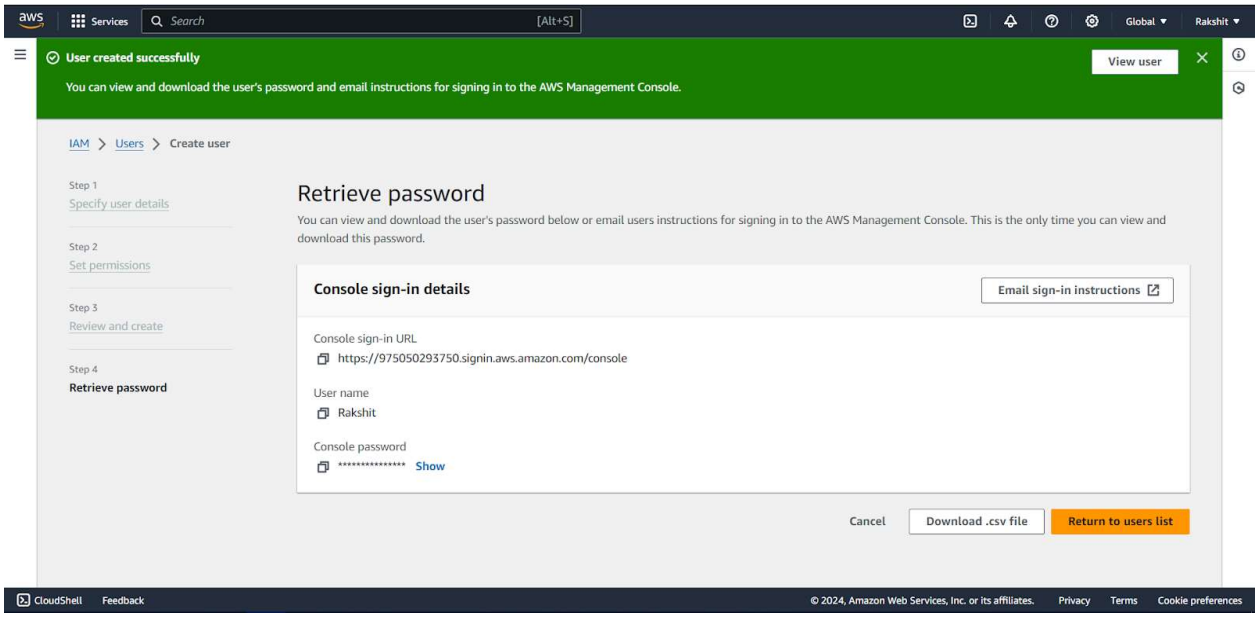


11. Once the group is created, select the group in which the user should be added.

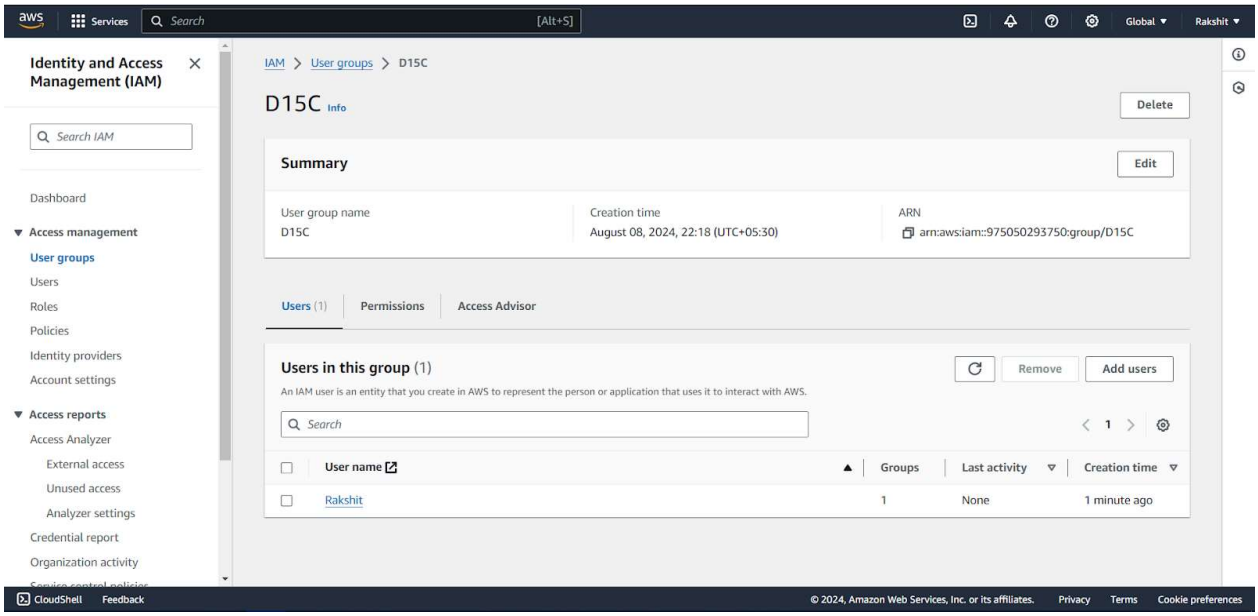


12. Recheck all the configuration and details of the user and click on create user.

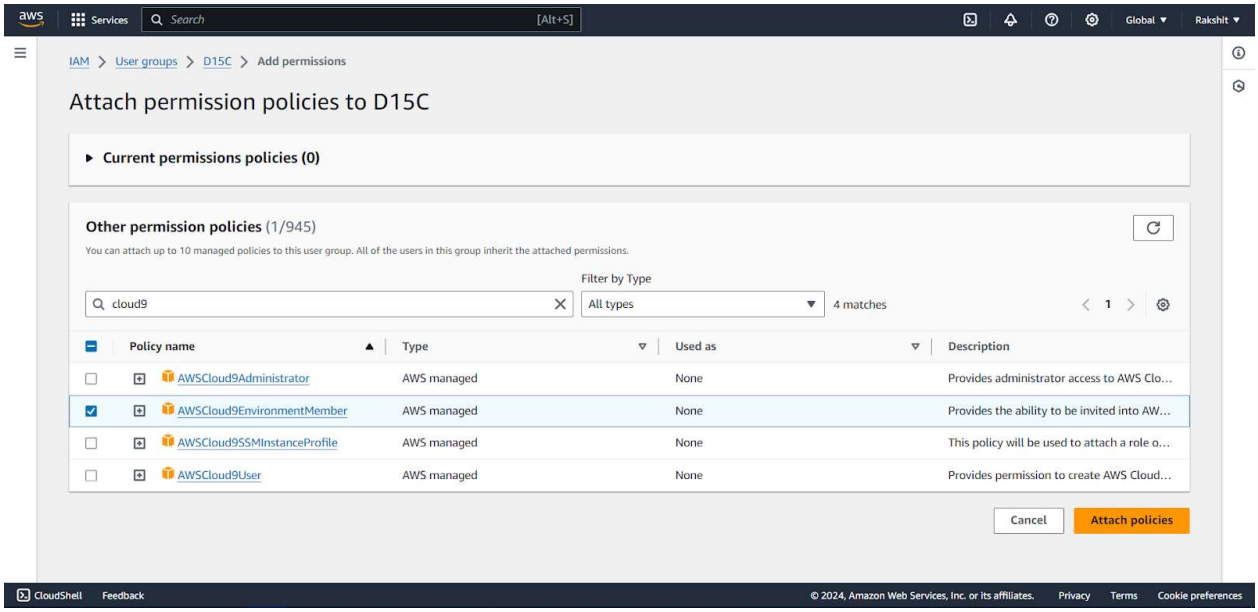




13. After creation of the user, go to the user groups tab and select the group in which the user has been added. Navigate to the permissions tab. Click on add permissions and attach policy.



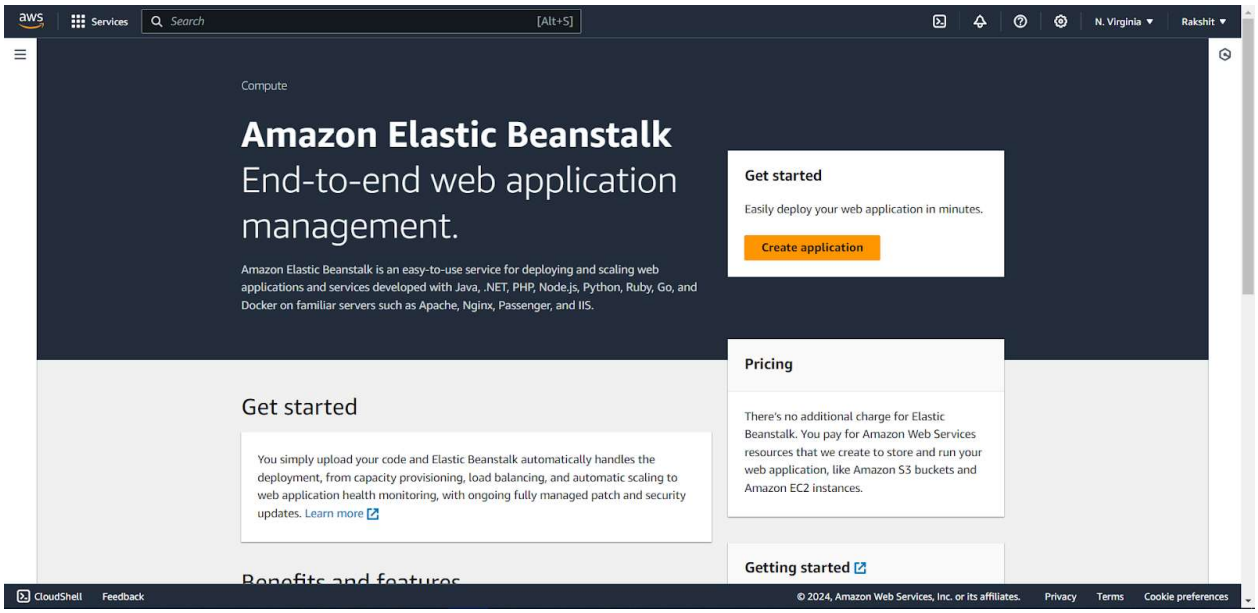
14. Search for the “AWSCloud9EnvironmentMember” policy and attach it.



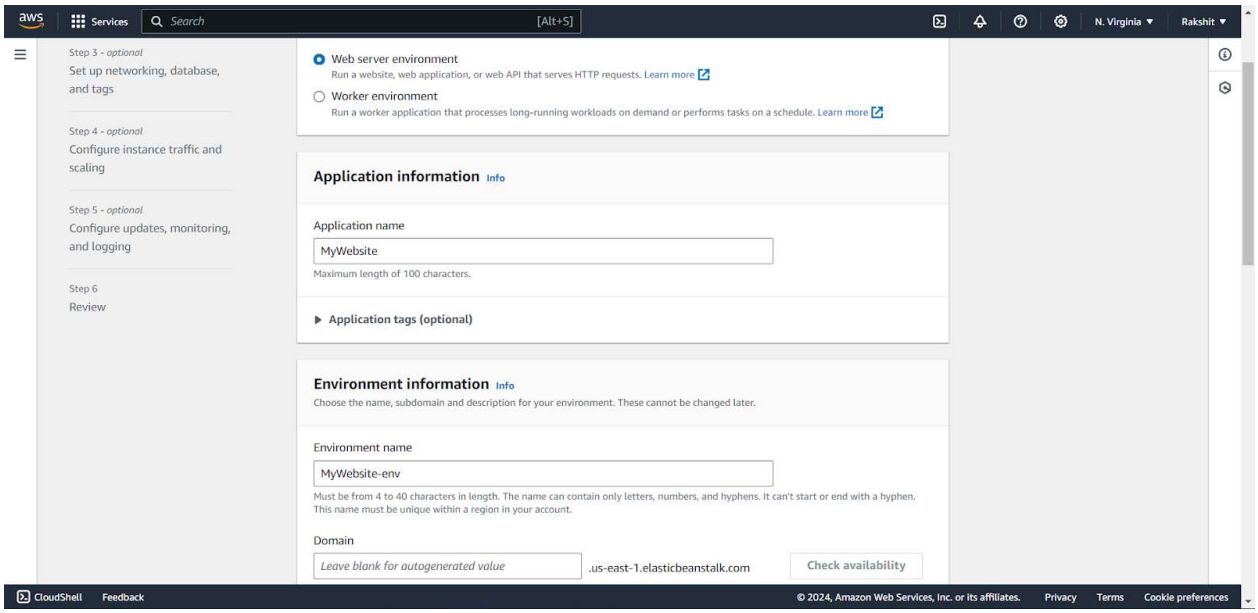
Experiment 2

Aim: To Build Your Application using AWS CodeBuild and Deploy on S3 / SEBS using AWS CodePipeline, deploy Sample Application on EC2 instance using AWS CodeDeploy.

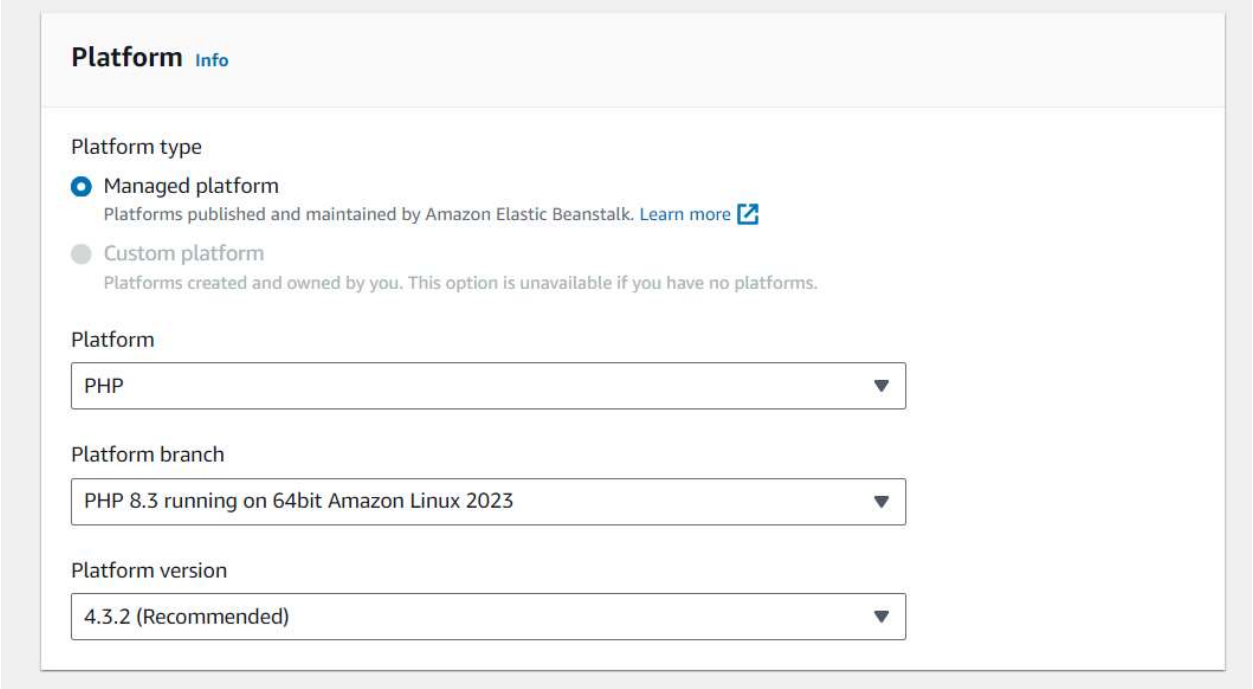
1. Login to your AWS account and search for Elastic Beanstalk.



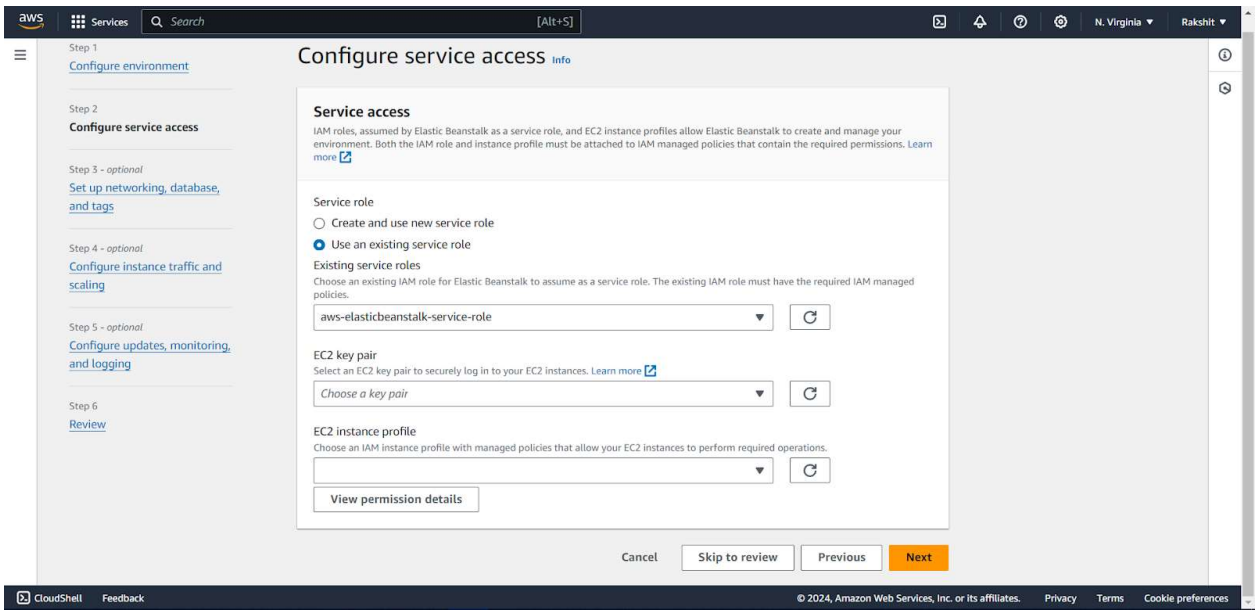
2. Click on create application. Enter your application names and other basic details.



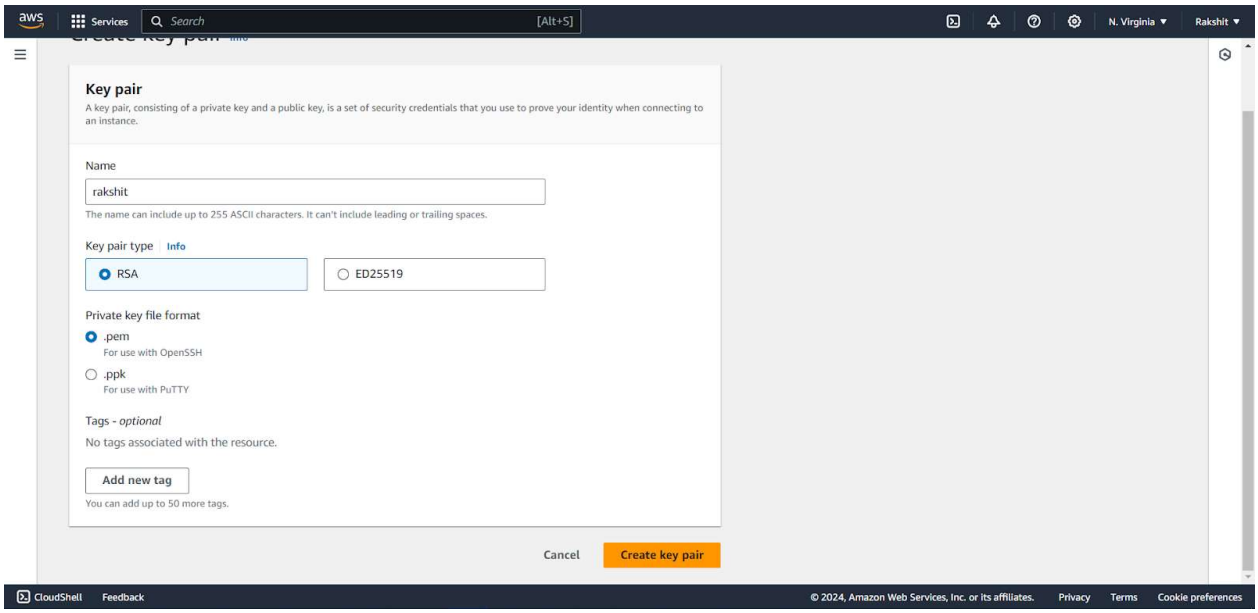
3. In the platform select PHP among other options. Platform branch and platform version will be entered automatically.



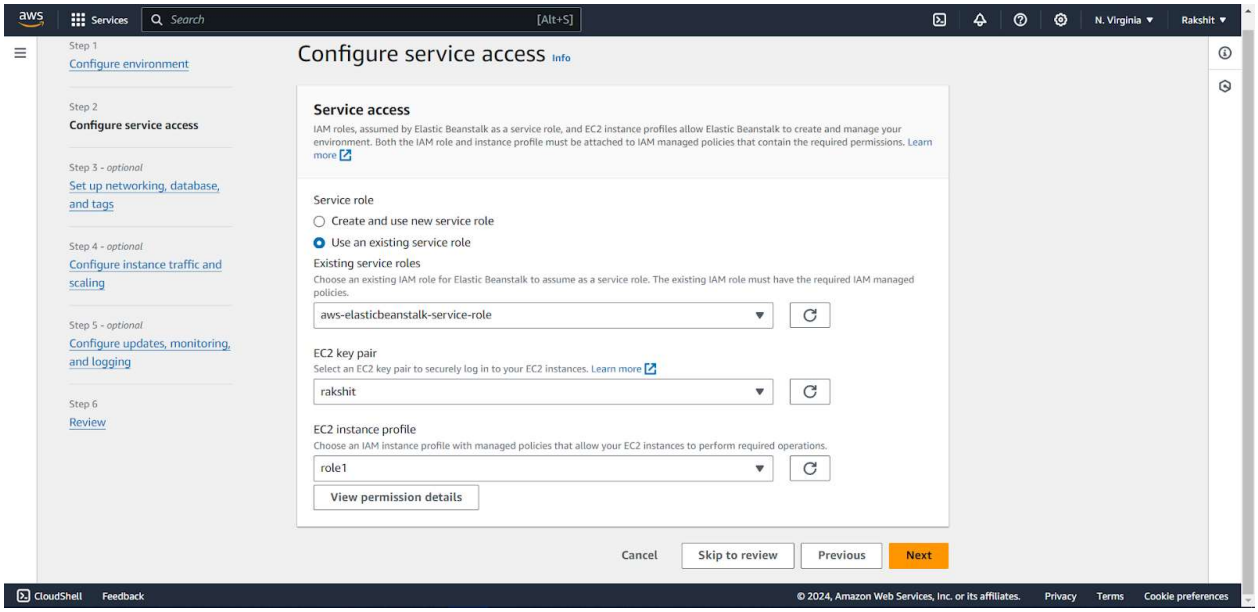
4. Keep the other setting to default and click on next. In service access click on “Use an existing service role”.



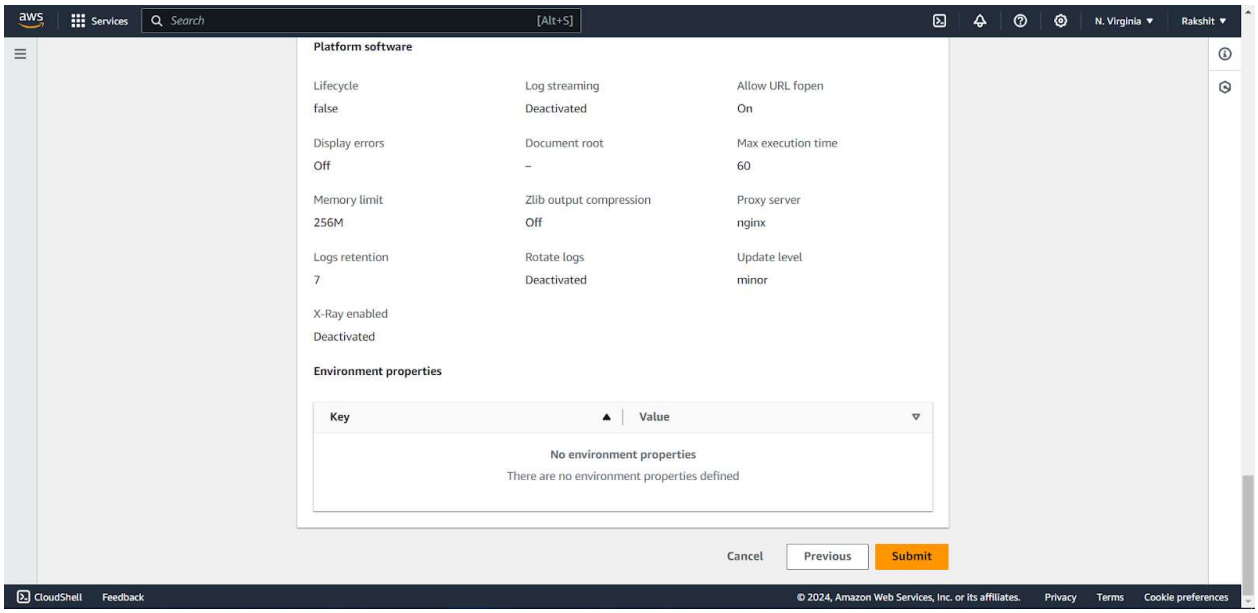
5. Go to EC2 service and click on Key pair to create a new key pair. Give the key pair a name and select the type as RSA. For private key file format select .pem.

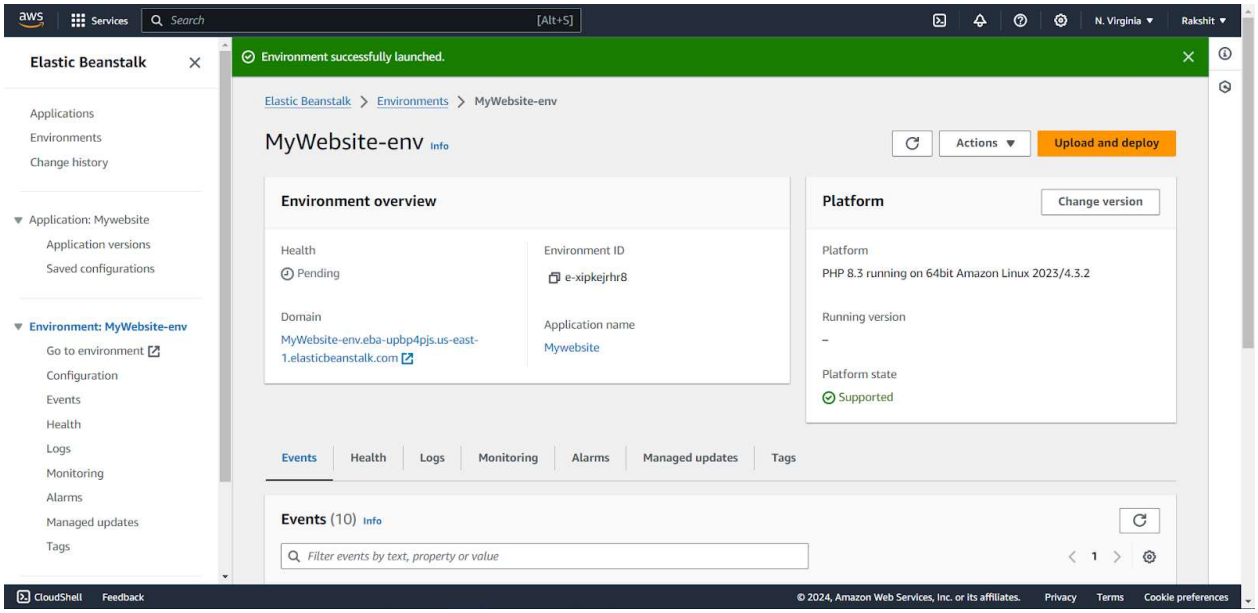


4. Come back to Elastic Beanstalk configuration. Select the newly created key pair from the dropdown menu. Also select the EC2 instance profile. Click on next.

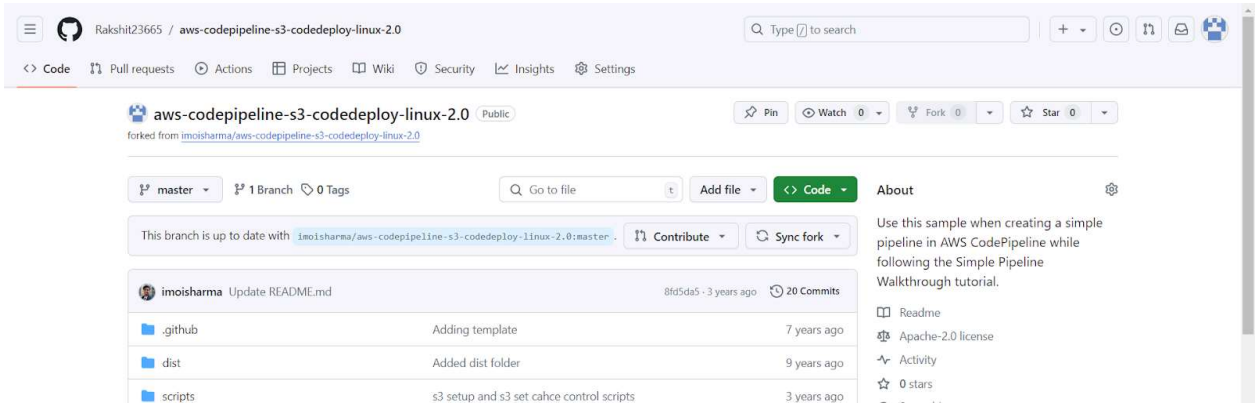


5. Skip to review. Review all the configurations and click on submit.Wait for the “Environment successfully launched” message.

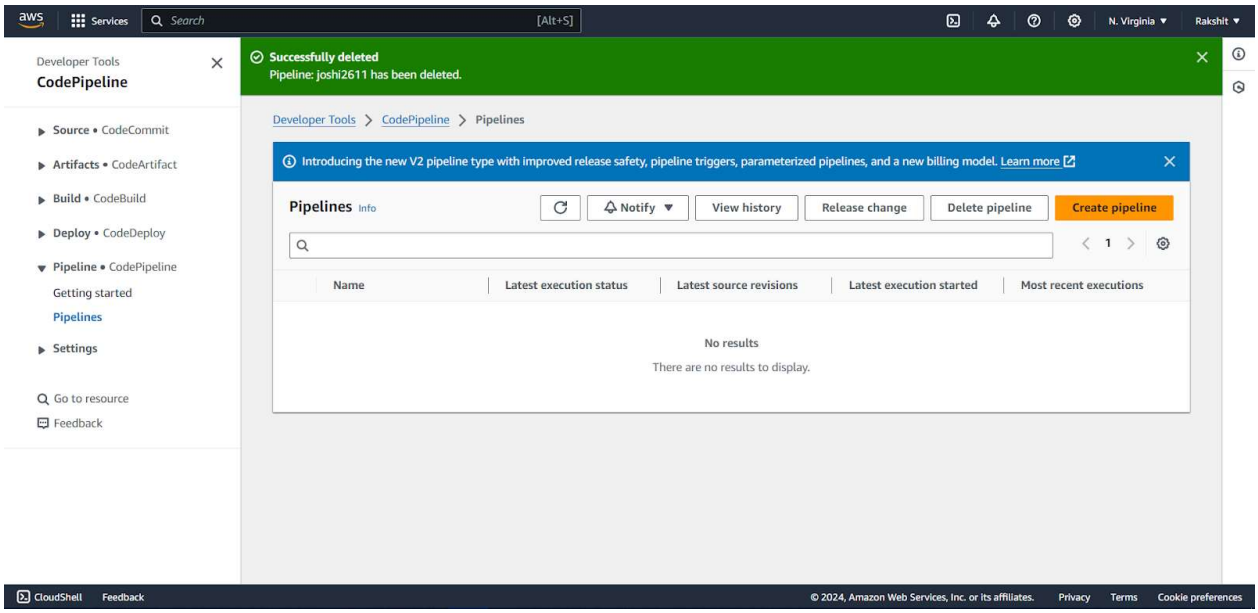




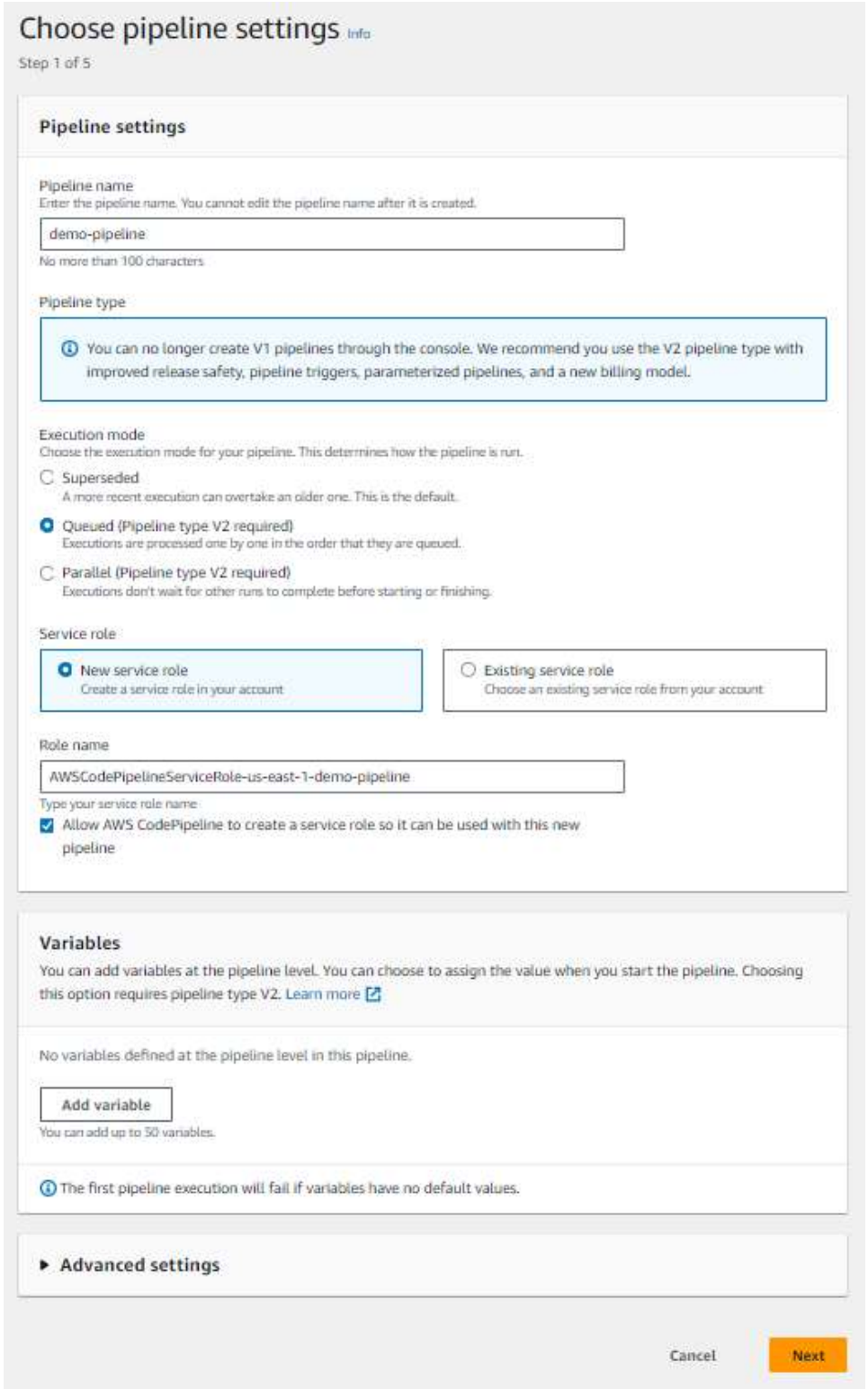
6. Create a github repository with the source code to be deployed. Here I have forked an existing repository.



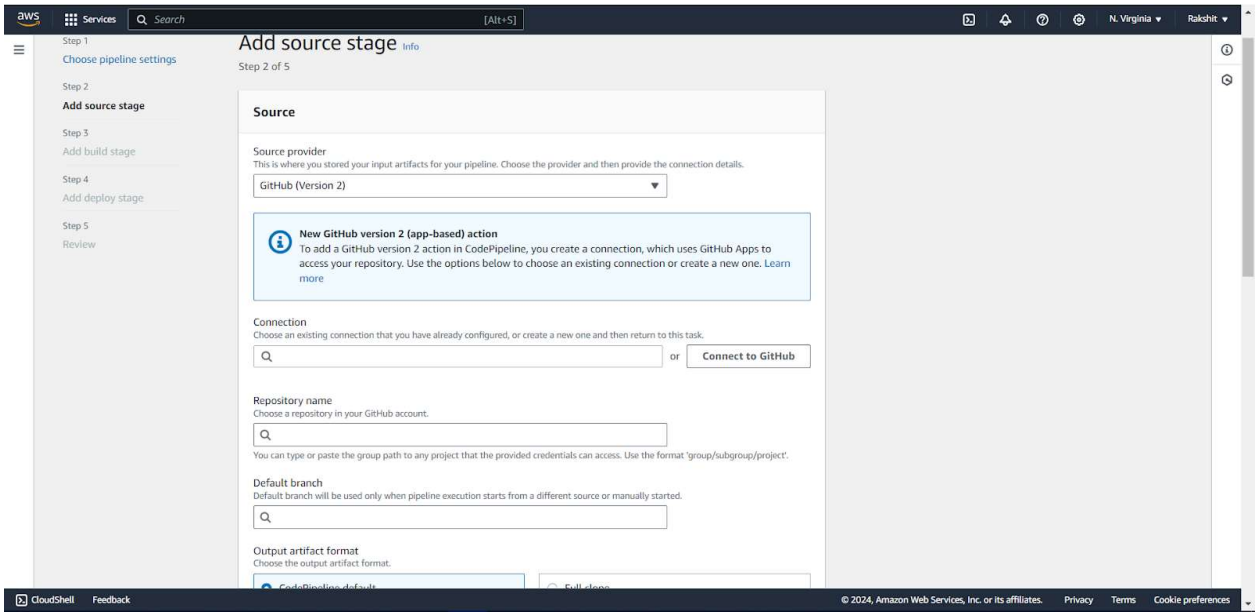
7. Go to CodePipeline service and click on create pipeline.



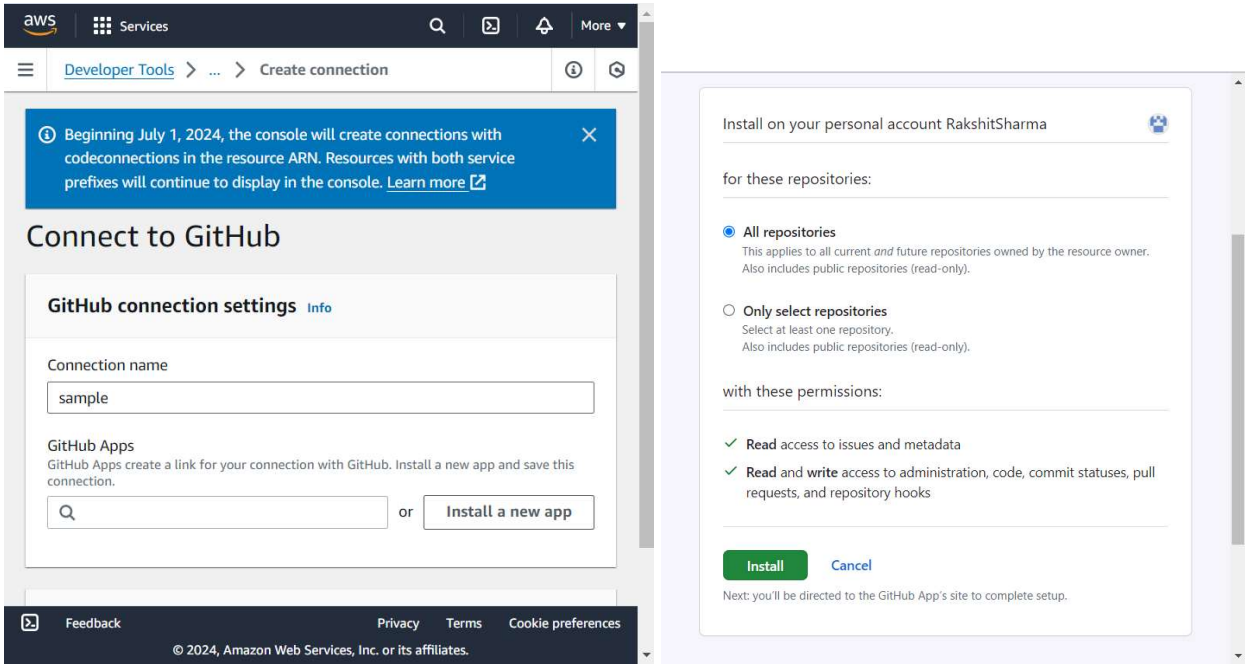
8. Give the pipeline a name. Role name will be generated automatically based on pipeline name

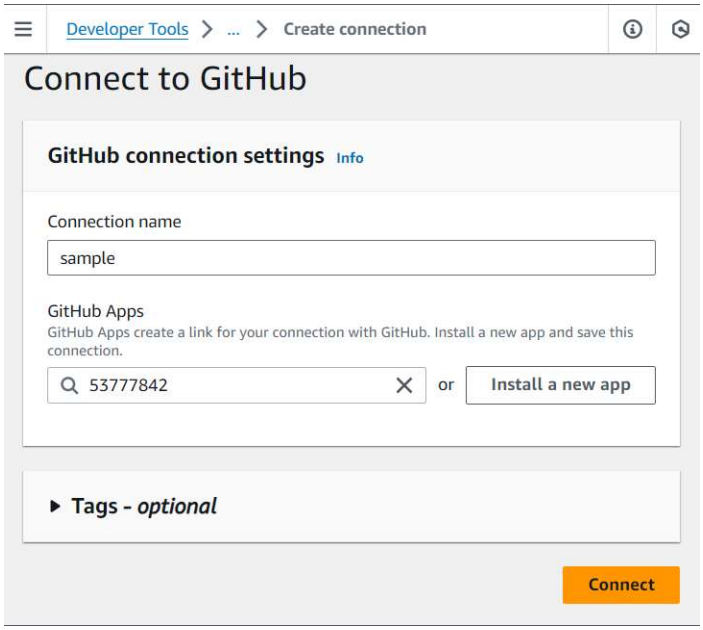


9. Select Github(Version 2) as a source provider. Click on connect to github to create a new connection if you don't have one.

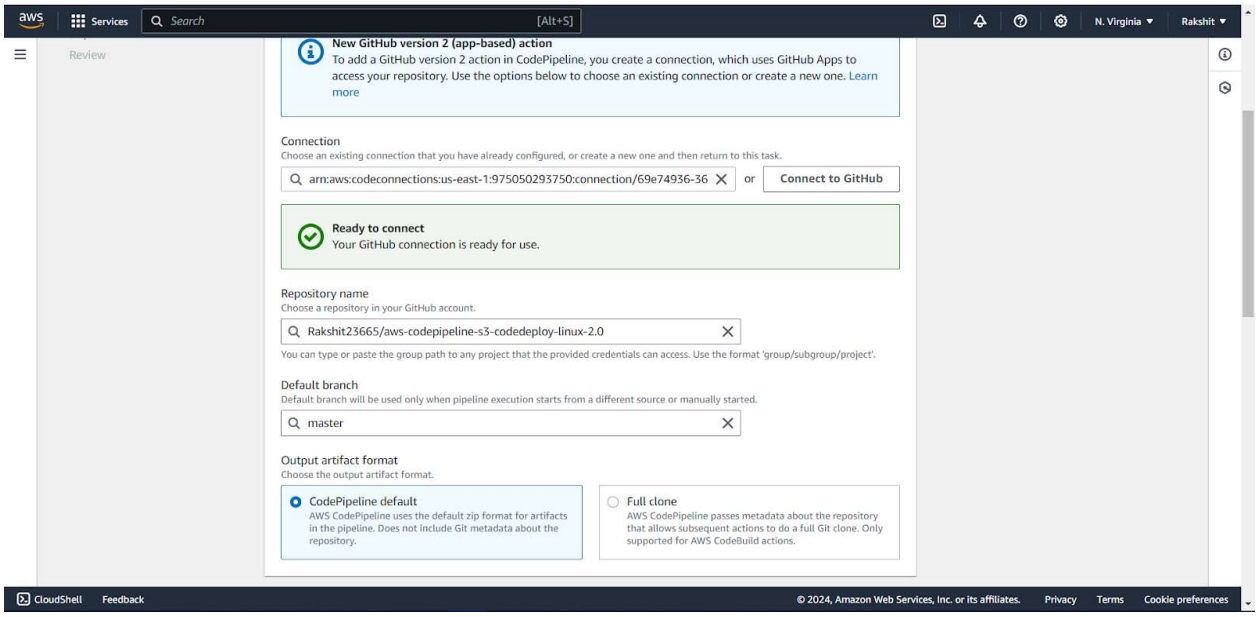


10. Give the connection a name and click on Install a new app. After this click on install. Once installation is complete click on connect to establish a connection.

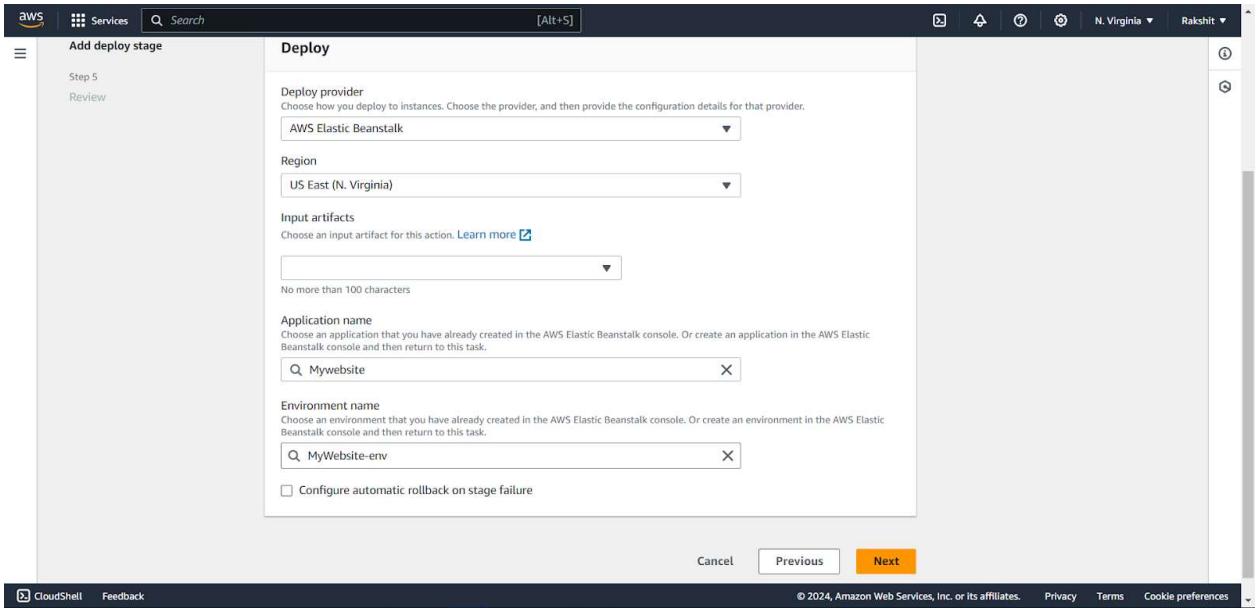




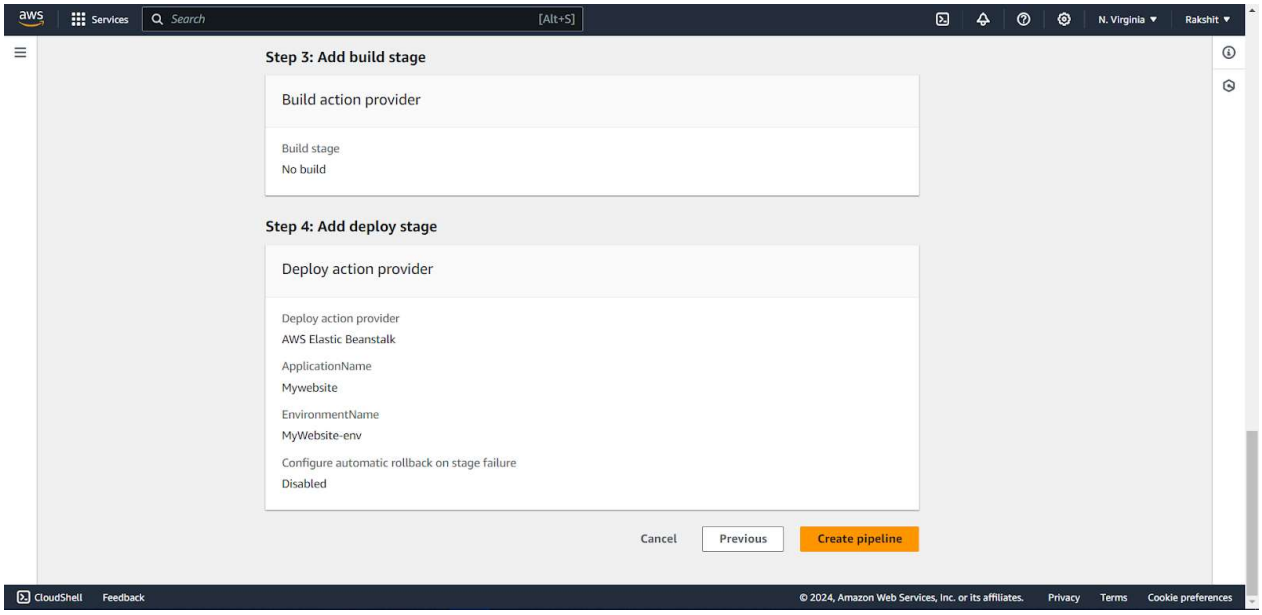
11. Once the connection is established, you will get a success message. Select the repository containing the source code. Also select the branch(usually master). Be sure to select the no filter option in Trigger section. Click on next.



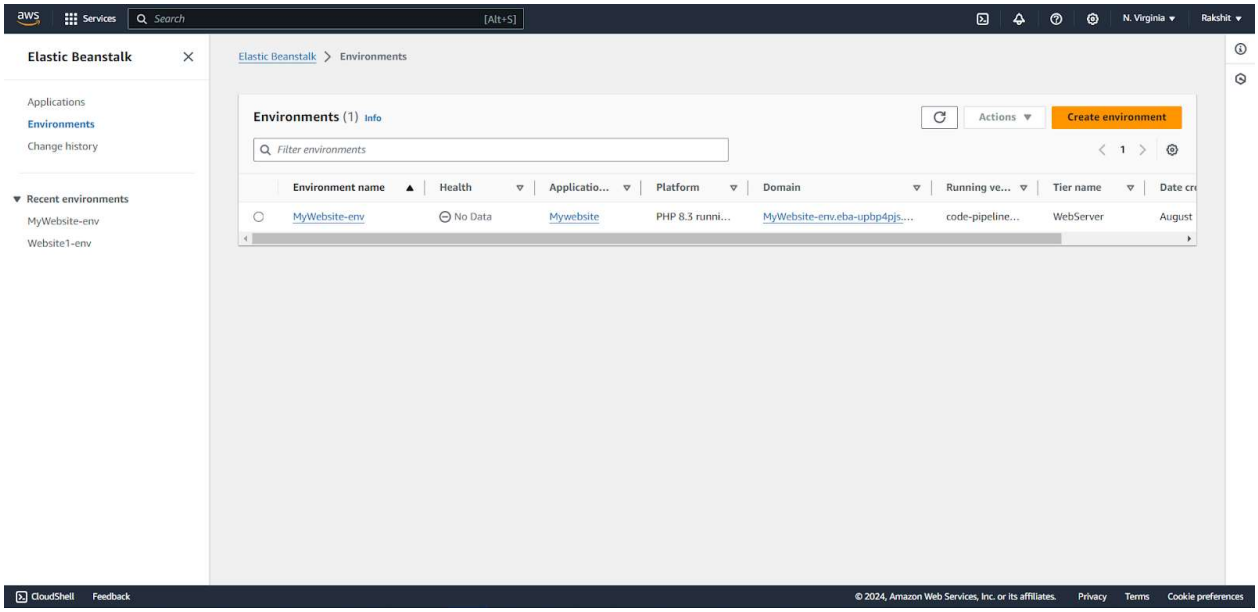
12. Skip the build stage and directly go to deploy stage. Select Elastic Beanstalk as Deploy provider. Select the Elastic Beanstalk application name that we created earlier. Click on next once done.

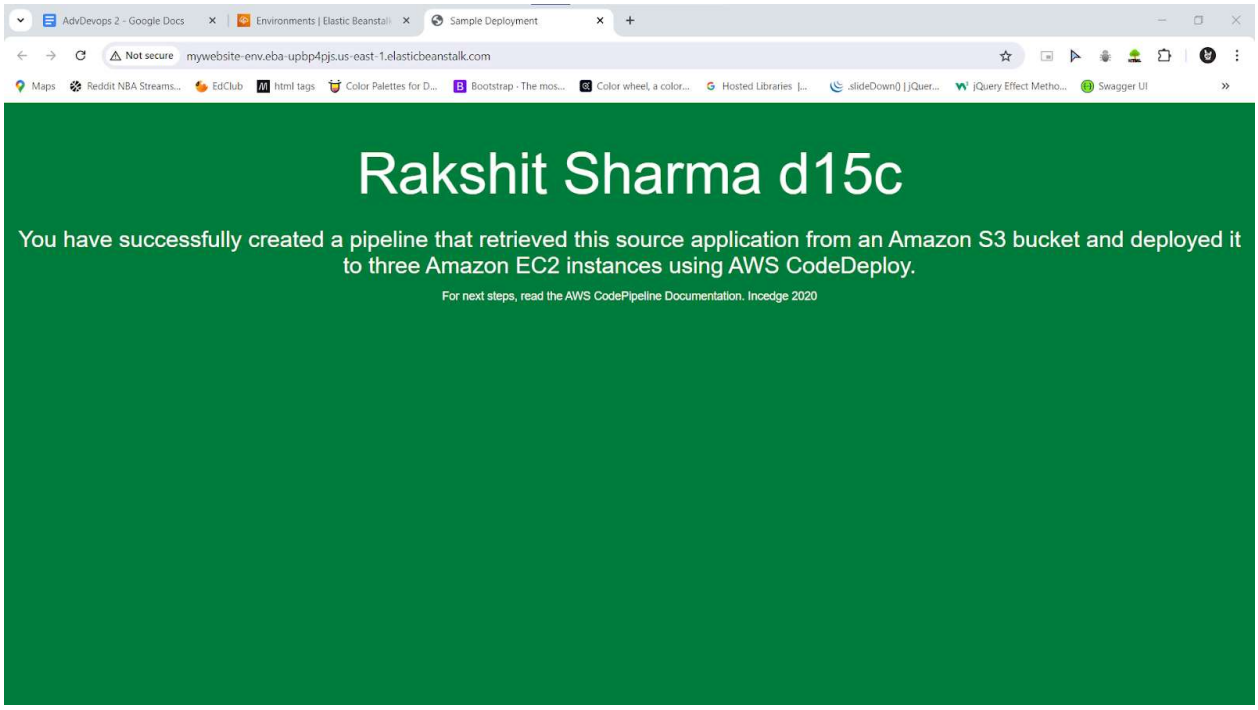


13. Review the configurations made and click on create pipeline.



14. Once the pipeline is created you can go to the environments page(Elastic Beanstalk). The website is hosted on the link under domain column. Click on the link to go to the hosted website.



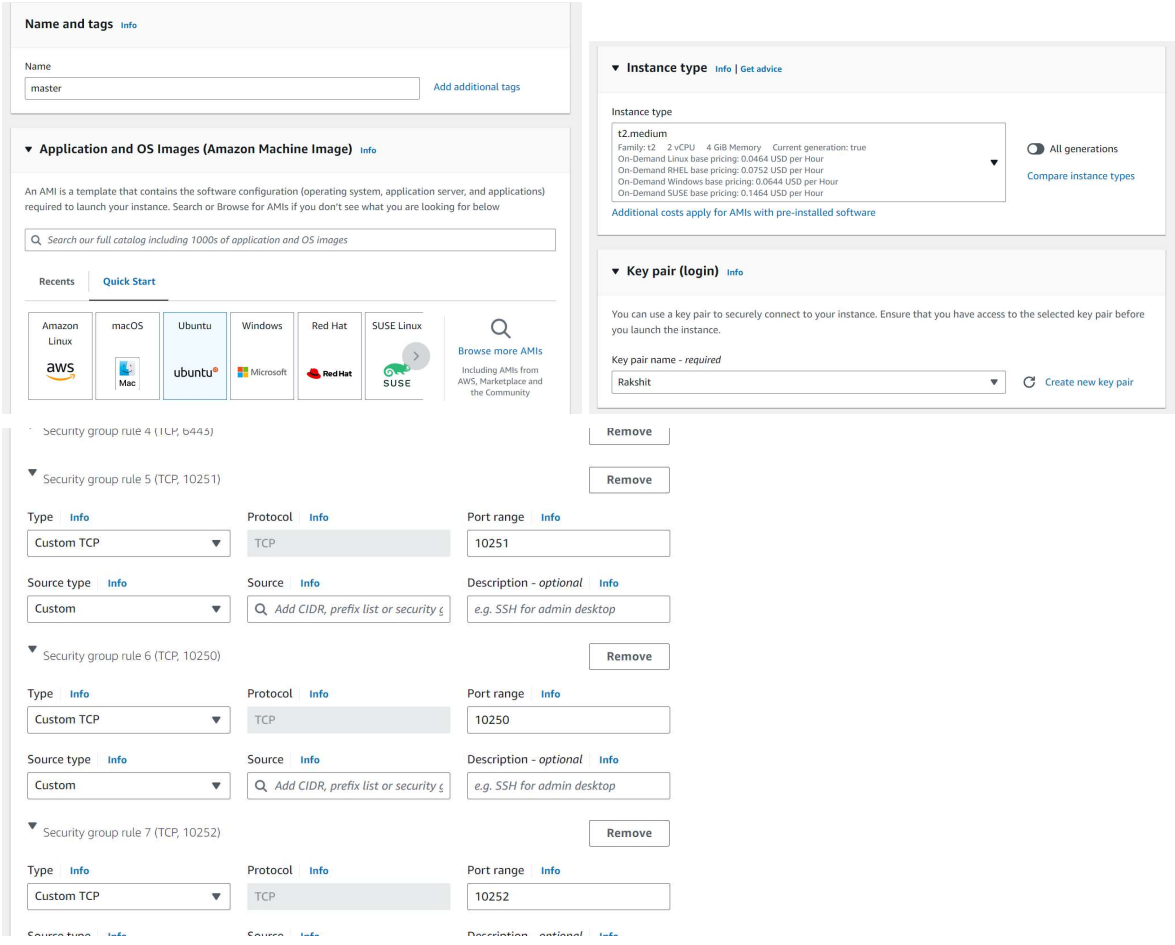


Experiment 3

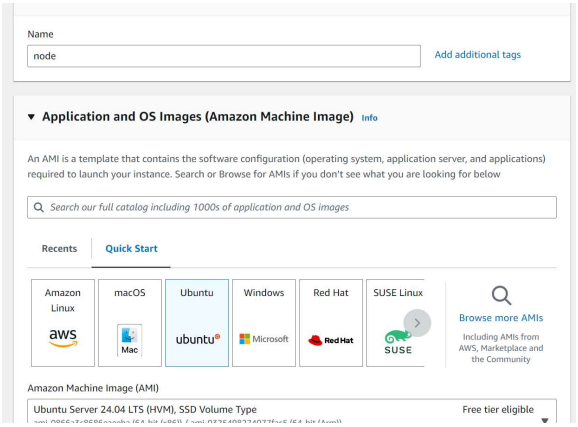
Aim: To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud Platforms.

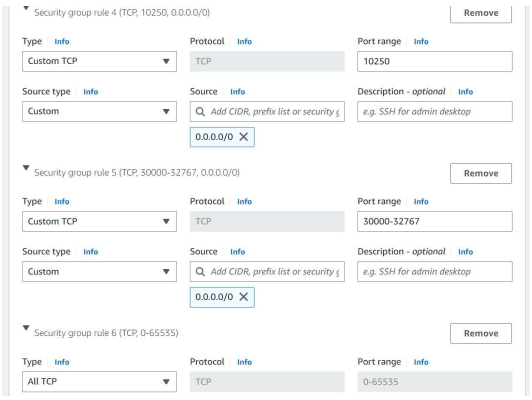
Step 1: Log in to your AWS Academy/personal account and launch 3 new Ec2 Instances. Select Ubuntu as AMI and t2.medium as Instance Type and create a key of type RSA with .pem extension and move the downloaded key to the new folder.We can use 3 Different keys or 1 common key also. Note: A minimum of 2 CPUs are required so Please select t2.medium and do not forget to stop the instance after the experiment because it is not available in the free tier.

Master:

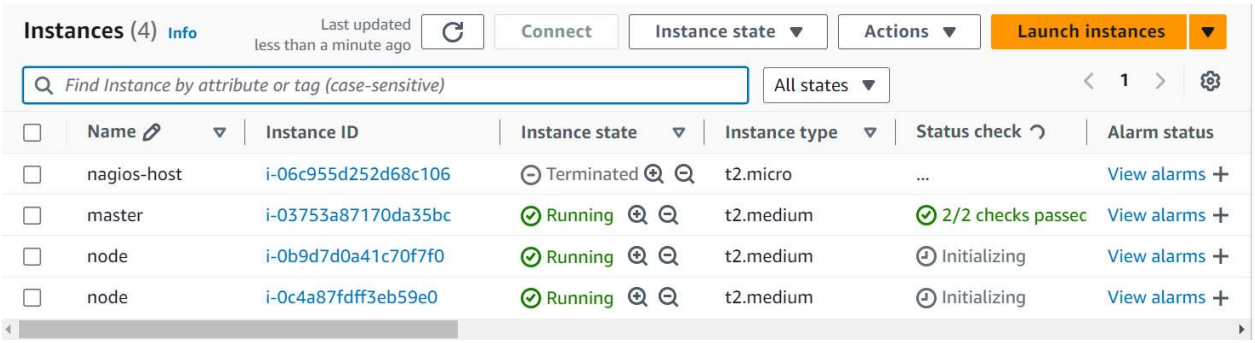


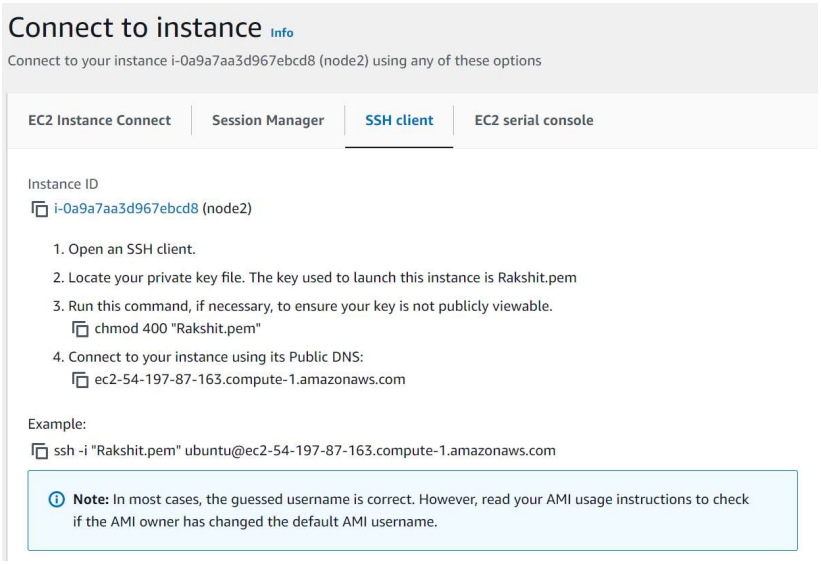
Worker:



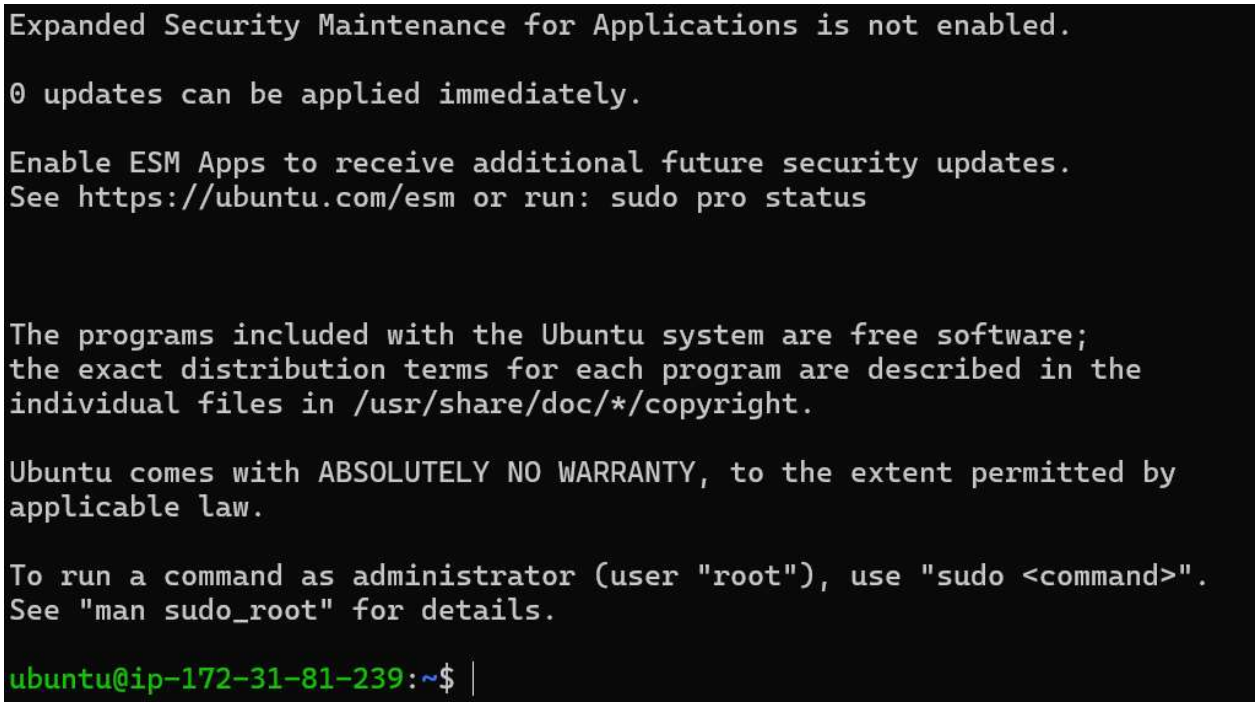


Step 2: After creating the instances click on Connect & connect all 3 instances and navigate to SSH Client.



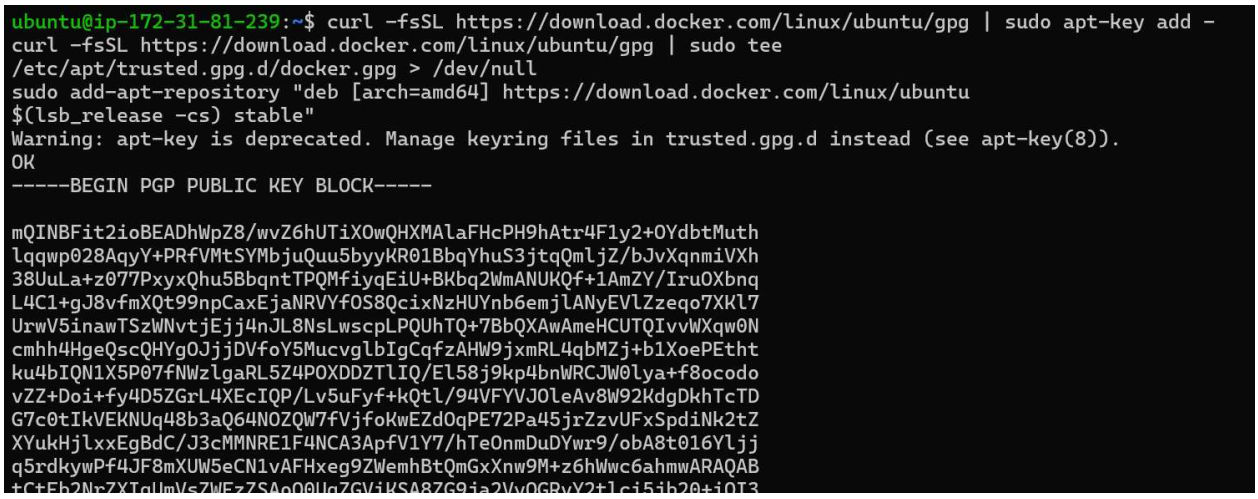


Step 3: Now open the folder in the terminal 3 times for Master, Node1& Node 2 where our .pem key is stored and paste the Example command (starting with ssh -i …..) in the terminal.(ssh -i "Master_Ec2_Key.pem" ubuntu@ec2-54-196-129-215.compute-1.amazonaws.com) Master:



Step 4: Run on Master,Node 1,and Node 2 the below commands to install and setup Docker in Master, Node1, and Node2.

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



sudo apt-get update

sudo apt-get install -y docker-ce

```
ubuntu@ip-172-31-81-239:~$ sudo apt-get update
sudo apt-get install -y docker-ce
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt
/trusted.gpg), see the DEPRECATION section in apt-key(8) for details.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  containerd.io docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0
  pigz slirp4netns
Suggested packages:
```

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

```
ubuntu@ip-172-31-81-239:~$ sudo mkdir -p /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
```

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

```
ubuntu@ip-172-31-81-239:~$ sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
```

Step 5: Run the below command to install Kubernets.

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

https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

```
ubuntu@ip-172-31-81-239:~$ 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
gpg: missing argument for option "-o"
-bash: /etc/apt/keyrings/kubernetes-apt-keyring.gpg: No such file or directory
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
```

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 apt-get install -y containerd

```
ubuntu@ip-172-31-81-239:~$ sudo apt-get install -y apt-transport-https ca-certificates curl
curl -fsSL https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
echo "deb https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list
sudo apt-get update
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20240203).
ca-certificates set to manually installed.
curl is already the newest version (8.5.0-2ubuntu10.4).
curl set to manually installed.
The following NEW packages will be installed:
  apt-transport-https
0 upgraded, 1 newly installed, 0 to remove and 6 not upgraded.
Need to get 3974 B of archives.
After this operation, 35.8 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 apt-transport-https all 2.7.14build2 [3974 B]
```

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

```
ubuntu@ip-172-31-81-239:~$ sudo mkdir -p /etc/containerd
sudo containerd config default | sudo tee /etc/containerd/config.toml
disabled_plugins = []
imports = []
oom_score = 0
plugin_dir = ""
required_plugins = []
root = "/var/lib/containerd"
state = "/run/containerd"
temp = ""
version = 2

[cgroup]
  path = ""

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

sudo systemctl restart containerd
sudo systemctl enable containerd

sudo systemctl status containerd

```
ubuntu@ip-172-31-81-239:~$ sudo systemctl restart containerd
sudo systemctl enable containerd
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 2024-10-03 12:03:56 UTC; 187ms ago
       Docs: https://containerd.io
   Main PID: 4388 (containerd)
      Tasks: 8
     Memory: 13.8M (peak: 14.1M)
        CPU: 47ms
     CGroup: /system.slice/containerd.service
             └─4388 /usr/bin/containerd

Oct 03 12:03:56 ip-172-31-81-239 containerd[4388]: time="2024-10-03T12:03:56.810259144Z" level=info msg="Start subscrib>
Oct 03 12:03:56 ip-172-31-81-239 containerd[4388]: time="2024-10-03T12:03:56.810436084Z" level=info msg="Start recoveri>
Oct 03 12:03:56 ip-172-31-81-239 containerd[4388]: time="2024-10-03T12:03:56.810483329Z" level=info msg=serving... addr>
Oct 03 12:03:56 ip-172-31-81-239 containerd[4388]: time="2024-10-03T12:03:56.810489545Z" level=info msg="Start event mo>
Oct 03 12:03:56 ip-172-31-81-239 containerd[4388]: time="2024-10-03T12:03:56.810583963Z" level=info msg="Start snapshot>
Oct 03 12:03:56 ip-172-31-81-239 containerd[4388]: time="2024-10-03T12:03:56.810599063Z" level=info msg="Start cni netw>
Oct 03 12:03:56 ip-172-31-81-239 containerd[4388]: time="2024-10-03T12:03:56.810605397Z" level=info msg="Start streamin>
Oct 03 12:03:56 ip-172-31-81-239 containerd[4388]: time="2024-10-03T12:03:56.810635592Z" level=info msg=serving... addr>
Oct 03 12:03:56 ip-172-31-81-239 systemd[1]: Started containerd.service - containerd container runtime.
Oct 03 12:03:56 ip-172-31-81-239 containerd[4388]: time="2024-10-03T12:03:56.813064087Z" level=info msg="containerd suc>
```

sudo apt-get install -y socat

```
ubuntu@ip-172-31-81-239:~$ 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 6 not upgraded.
Need to get 374 kB of archives.
After this operation, 1649 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 socat amd64 1.8.0.0-4build3 [374 kB]
Fetched 374 kB in 0s (12.2 MB/s)
Selecting previously unselected package socat.
(Reading database ... 68148 files and directories currently installed.)
Preparing to unpack .../socat_1.8.0.0-4build3_amd64.deb ...
Unpacking socat (1.8.0.0-4build3) ...
```

Step 6: Initialize the Kubecluster .Now Perform this Command only for Master.
sudo kubeadm init --pod-network-cidr=10.244.0.0/16

```
ubuntu@ip-172-31-82-119:~$ 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'
W1003 12:37:31.290835   10158 checks.go:846] detected that the sandbox image "registry.k8s.io/pause:3.8" of the containe
r runtime is inconsistent with that used by kubeadm.It is recommended to use "registry.k8s.io/pause:3.10" as the CRI san
dbox 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-172-31-82-119 kubernetes kubernetes.default kubernetes.defaul
t.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 172.31.82.119]
[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-172-31-82-119 localhost] and IPs [172.31.82.119 127.0.0.1 :
:1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [ip-172-31-82-119 localhost] and IPs [172.31.82.119 127.0.0.1 ::1
```

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

```
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 172.31.82.119:6443 --token h4iisg.g9tdfmc88m9toefp \
        --discovery-token-ca-cert-hash sha256:5b3c7cfdf8115a26f1e73b752820d2b27b84ed476b344aaa3bd1a90e4b1f2105
ubuntu@ip-172-31-82-119:~$  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Step 7: Now Run the command kubectl get nodes to see the nodes before executing Join command on nodes.

```
ubuntu@ip-172-31-82-119:~$ kubectl get nodes
NAME               STATUS     ROLES           AGE   VERSION
ip-172-31-82-119   NotReady   control-plane   82s   v1.31.1
```

Step 8: Now Run the following command on Node 1 and Node 2 to Join to master.
sudo kubeadm join 172.31.82.119:6443 --token h4iisg.g9tdfmc88m9toefp \
--discovery-token-ca-cert-hash
sha256:5b3c7cfdf8115a26f1e73b752820d2b27b84ed476b344aaa3bd1a90e4b1f2105

```
ubuntu@ip-172-31-84-169:~$ sudo kubeadm join 172.31.82.119:6443 --token h4iisg.g9tdfmc88m9toefp \
        --discovery-token-ca-cert-hash sha256:5b3c7cfdf8115a26f1e73b752820d2b27b84ed476b344aaa3bd1a90e4b1f2105
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-check] Waiting for a healthy kubelet at http://127.0.0.1:10248/healthz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 501.315929ms
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap

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

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

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

```
ubuntu@ip-172-31-82-119:~$ kubectl get nodes
NAME                STATUS     ROLES           AGE     VERSION
ip-172-31-82-119    NotReady   control-plane   2m56s   v1.31.1
ip-172-31-84-169    NotReady   <none>          43s     v1.31.1
ip-172-31-87-189    NotReady   <none>          39s     v1.31.1
```

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

kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml

```
ubuntu@ip-172-31-82-119:~$ 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
```

sudo systemctl status kubelet

```
ubuntu@ip-172-31-82-119:~$ 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 2024-10-03 12:37:52 UTC; 4min 4s ago
       Docs: https://kubernetes.io/docs/
   Main PID: 10849 (kubelet)
      Tasks: 10 (limit: 4676)
     Memory: 32.3M (peak: 33.0M)
        CPU: 5.030s
     CGroup: /system.slice/kubelet.service
             └─10849 /usr/bin/kubelet --bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf --kubeconfig=/etc/k>

Oct 03 12:41:52 ip-172-31-82-119 kubelet[10849]: E1003 12:41:52.678397   10849 kuberuntime_container.go:851] "Kill cont>
Oct 03 12:41:52 ip-172-31-82-119 kubelet[10849]:         rpc error: code = Unknown desc = failed to kill container "233>
Oct 03 12:41:52 ip-172-31-82-119 kubelet[10849]:         : unknown
Oct 03 12:41:52 ip-172-31-82-119 kubelet[10849]:  > pod="kube-system/etcd-ip-172-31-82-119" podUID="201366195a044a56173>
Oct 03 12:41:52 ip-172-31-82-119 kubelet[10849]: E1003 12:41:52.688038   10849 log.go:32] "StopPodSandbox from runtime >
Oct 03 12:41:52 ip-172-31-82-119 kubelet[10849]:         rpc error: code = Unknown desc = failed to stop container "233>
Oct 03 12:41:52 ip-172-31-82-119 kubelet[10849]:         : unknown
```

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

```
ubuntu@ip-172-31-82-119:~$ kubectl get nodes -o wide
NAME               STATUS   ROLES           AGE     VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE             KERNEL-
VERSION   CONTAINER-RUNTIME
ip-172-31-82-119   Ready    control-plane   4m32s   v1.31.1   172.31.82.119   <none>        Ubuntu 24.04.1 LTS   6.8.0-1
016-aws   containerd://1.7.12
ip-172-31-84-169   Ready    <none>          2m19s   v1.31.1   172.31.84.169   <none>        Ubuntu 24.04.1 LTS   6.8.0-1
016-aws   containerd://1.7.12
ip-172-31-87-189   Ready    <none>          2m15s   v1.31.1   172.31.87.189   <none>        Ubuntu 24.04.1 LTS   6.8.0-1
016-aws   containerd://1.7.12
```

Now to Rename run this command kubectl label node ip-172-31-18-135 kubernetes.io/role=worker Rename to Node 1:kubectl label node ip-172-31-28-117 kubernetes.io/role=Node1 Rename to Node 2:kubectl label node ip-172-31-18-135 kubernetes.io/role=Node2

Step 11: Run command kubectl get nodes -o wide . And Hence we can see we have Successfully connected Node 1 and Node 2 to the Master.

```
ubuntu@ip-172-31-82-119:~$ kubectl get nodes
NAME               STATUS   ROLES           AGE     VERSION
ip-172-31-82-119   Ready    control-plane   6m52s   v1.31.1
ip-172-31-84-169   Ready    Node1           4m39s   v1.31.1
ip-172-31-87-189   Ready    Node2           4m35s   v1.31.1
```

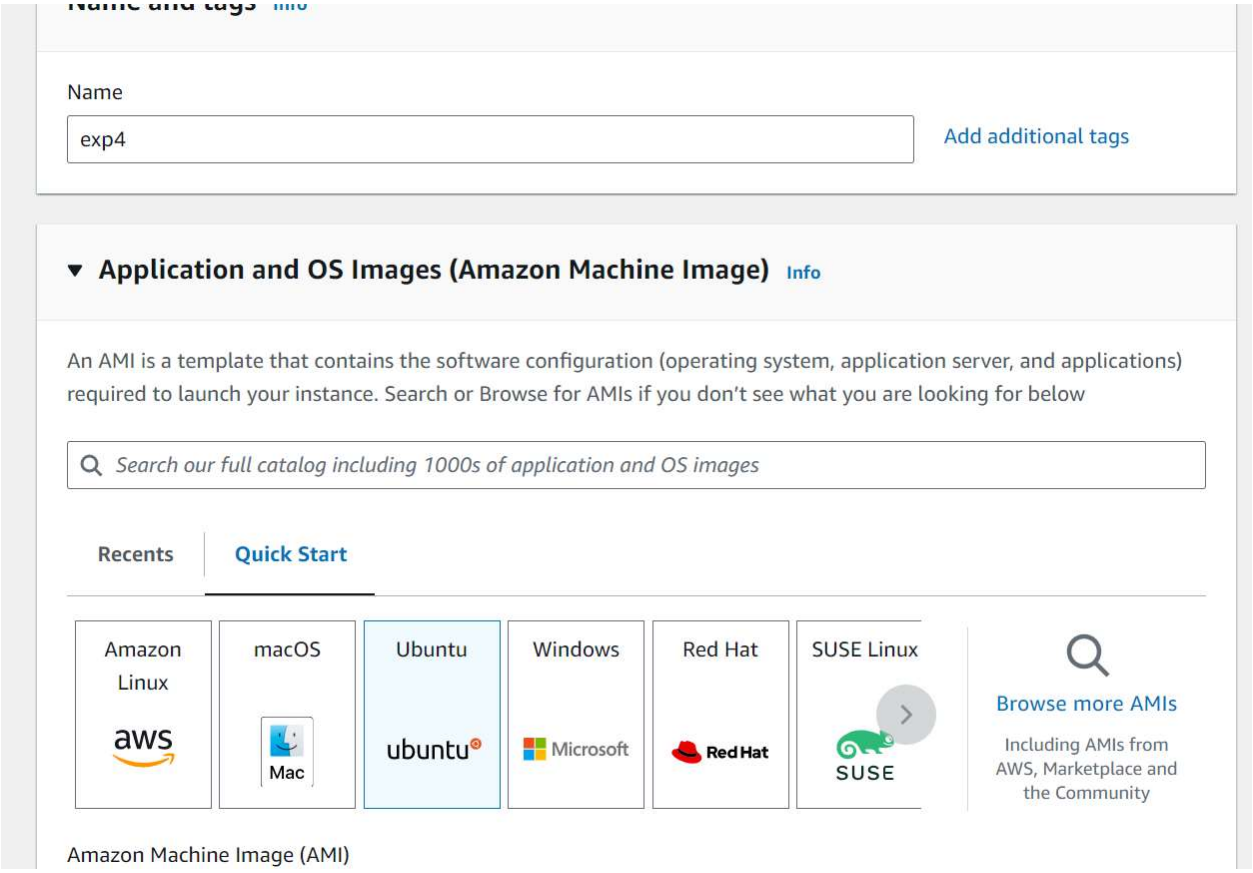
Conclusion:

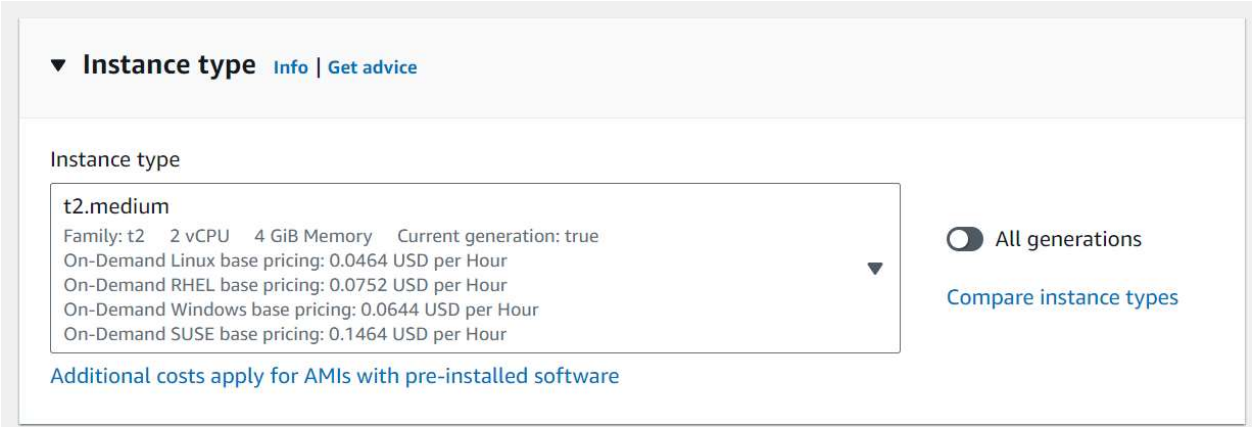
In this experiment, we successfully set up a Kubernetes cluster on AWS EC2 instances. We began by provisioning three EC2 instances—Master, Node1, and Node2—with appropriate security rules to facilitate communication within the cluster. After securely SSHing into each instance, we installed Docker as the container runtime on all machines. We proceeded by installing Kubernetes on each instance and initializing the cluster from the master node. Following cluster initialization, the worker nodes were connected to the master using the kubeadm join command, effectively forming a cohesive Kubernetes cluster. Additionally, we implemented a network plugin to ensure proper communication between nodes, allowing the nodes' status to transition to "Ready."

Experiment 4

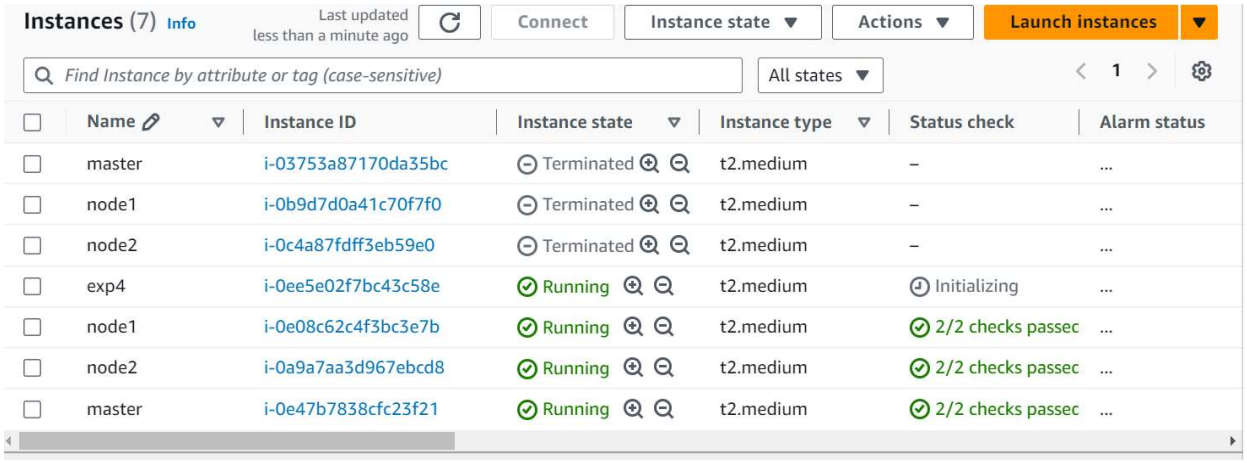
Aim: To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deploy Your First Kubernetes Application.

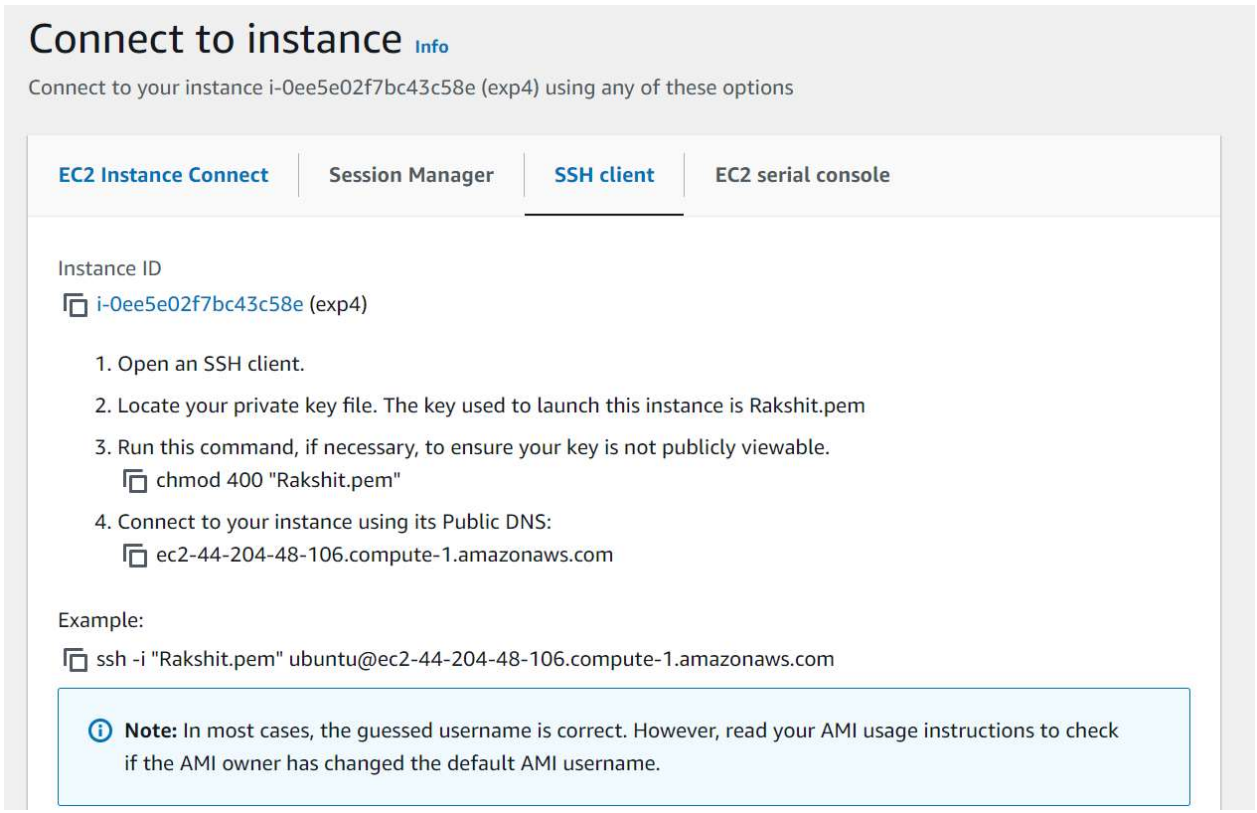
Step 1: Log in to your AWS Academy/personal account and launch a new Ec2 Instance. Select Ubuntu as AMI and t2.medium as Instance Type, create a key of type RSA with .pem extension, and move the downloaded key to the new folder. Note: A minimum of 2 CPUs are required so Please select t2.medium and do not forget to stop the instance after the experiment because it is not available in the free tier.





Step 2: After creating the instance click on Connect the instance and navigate to SSH Client.





Step 3: Now open the folder in the terminal where our .pem key is stored and paste the Example command (starting with ssh -i …..) in the terminal.(ssh -i "Master_Ec2_Key.pem"

ubuntu@ec2-54-196-129-215.compute-1.amazonaws.com)

```
 System information as of Thu Oct  3 13:05:18 UTC 2024

  System load:  0.34              Processes:             118
  Usage of /:   22.8% of 6.71GB   Users logged in:       0
  Memory usage: 5%                IPv4 address for enX0: 172.31.86.43
  Swap usage:   0%

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
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-86-43:~$
```

Step 4: Run the below commands to install and setup Docker.
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add - curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/trusted.gpg.d/docker.gpg > /dev/null sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"

```
ubuntu@ip-172-31-86-43:~$ 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"
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBFit2ioBEADhWpZ8/wvZ6hUTiXOwQHXMAlaFHcPH9hAtr4F1y2+OYdbtMuth
lqqwp028AqyY+PRfVMtSYMbjuQuu5byyKR01BbqYhuS3jtqQmljZ/bJvXqnmiVXh
38UuLa+z077PxyxQhu5BbqntTPQMfiyqEiU+BKbq2WmANUKQf+1AmZY/IruOXbnq
L4C1+gJ8vfmXQt99npCaxEjaNRVYfOS8QcixNzHUYnb6emjlANyEVlZzeqo7XKl7
UrwV5inawTSzWNvtjEjj4nJL8NsLwscpLPQUhTQ+7BbQXAwAmeHCUTQIvvWXqw0N
cmhh4HgeQscQHYgOJjjDVfoY5MucvglbIgCqfzAHW9jxmRL4qbMZj+b1XoePEtht
ku4bIQN1X5P07fNWzlgaRL5Z4POXDDZTlIQ/El58j9kp4bnWRCJW0lya+f8ocodo
vZZ+Doi+fy4D5ZGrL4XEcIQP/Lv5uFyf+kQtl/94VFYVJOleAv8W92KdgDkhTcTD
G7c0tIkVEKNUq48b3aQ64NOZQW7fVjfoKwEZdOqPE72Pa45jrZzvUFxSpdiNk2tZ
```

sudo apt-get update

sudo apt-get install -y docker-ce

```
ubuntu@ip-172-31-86-43:~$ sudo apt-get update
sudo apt-get install -y docker-ce
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/ap
/trusted.gpg), see the DEPRECATION section in apt-key(8) for details.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

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

```
ubuntu@ip-172-31-86-43:~$ sudo mkdir -p /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
    "exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
{
    "exec-opts": ["native.cgroupdriver=systemd"]
}
```

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

```
ubuntu@ip-172-31-86-43:~$ sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
```

Step 5: Run the below command to install Kubernets. curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]

https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

```
ubuntu@ip-172-31-86-43:~$ 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/ /' | sud
o tee /etc/apt/sources.list.d/kubernetes.list
gpg: missing argument for option "-o"
-bash: /etc/apt/keyrings/kubernetes-apt-keyring.gpg: No such file or directory
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
```

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

```
ubuntu@ip-172-31-86-43:~$ sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  InRelease [1186 B]
Err:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  InRelease
  The following signatures couldn't be verified because the public key is not available: NO_PUBKEY 234654DA9A296436
Reading package lists... Done
```

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

```
ubuntu@ip-172-31-86-43:~$ sudo systemctl enable --now kubelet
sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
W1003 13:11:39.162569    5084 checks.go:1080] [preflight] WARNING: Couldn't create the interface used for talking to the
 container runtime: failed to create new CRI runtime service: validate service connection: validate CRI v1 runtime API f
or endpoint "unix:///var/run/containerd/containerd.sock": rpc error: code = Unimplemented desc = unknown service runtime
.v1.RuntimeService
        [WARNING FileExisting-socat]: socat not found in system path
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action beforehand using 'kubeadm config images pull'
```

Now We have got an error. So we have to perform some additional commands as follow.
sudo apt-get install -y containerd

```
ubuntu@ip-172-31-86-43:~$ sudo apt-get install -y containerd
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0 pigz
  slirp4netns
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  runc
The following packages will be REMOVED:
  containerd.io docker-ce
The following NEW packages will be installed:
  containerd runc
0 upgraded, 2 newly installed, 2 to remove and 6 not upgraded.
Need to get 47.2 MB of archives.
After this operation, 53.1 MB disk space will be freed.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 runc amd64 1.1.12-0ubuntu3.1 [8599 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 containerd amd64 1.7.12-0ubuntu4.1 [38.6 M
```

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

```
ubuntu@ip-172-31-86-43:~$ sudo mkdir -p /etc/containerd
sudo containerd config default | sudo tee /etc/containerd/config.toml
disabled_plugins = []
imports = []
oom_score = 0
plugin_dir = ""
required_plugins = []
root = "/var/lib/containerd"
state = "/run/containerd"
temp = ""
version = 2

[cgroup]
  path = ""

[debug]
  address = ""
  format = ""
```

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

```
ubuntu@ip-172-31-86-43:~$ sudo systemctl restart containerd
sudo systemctl enable containerd
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 2024-10-03 13:12:51 UTC; 276ms ago
       Docs: https://containerd.io
   Main PID: 5462 (containerd)
      Tasks: 8
     Memory: 13.5M (peak: 14.1M)
        CPU: 54ms
     CGroup: /system.slice/containerd.service
             └─5462 /usr/bin/containerd

Oct 03 13:12:51 ip-172-31-86-43 containerd[5462]: time="2024-10-03T13:12:51.543254591Z" level=info msg=serving... addre>
Oct 03 13:12:51 ip-172-31-86-43 containerd[5462]: time="2024-10-03T13:12:51.543291344Z" level=info msg=serving... addre>
Oct 03 13:12:51 ip-172-31-86-43 containerd[5462]: time="2024-10-03T13:12:51.543326951Z" level=info msg="Start subscribi>
Oct 03 13:12:51 ip-172-31-86-43 containerd[5462]: time="2024-10-03T13:12:51.543358179Z" level=info msg="Start recoverin>
Oct 03 13:12:51 ip-172-31-86-43 containerd[5462]: time="2024-10-03T13:12:51.543399144Z" level=info msg="Start event mon>
Oct 03 13:12:51 ip-172-31-86-43 containerd[5462]: time="2024-10-03T13:12:51.543407458Z" level=info msg="Start snapshots>
Oct 03 13:12:51 ip-172-31-86-43 containerd[5462]: time="2024-10-03T13:12:51.543414974Z" level=info msg="Start cni netwo>
Oct 03 13:12:51 ip-172-31-86-43 containerd[5462]: time="2024-10-03T13:12:51.543422043Z" level=info msg="Start streaming>
Oct 03 13:12:51 ip-172-31-86-43 containerd[5462]: time="2024-10-03T13:12:51.543466839Z" level=info msg="containerd succ>
Oct 03 13:12:51 ip-172-31-86-43 systemd[1]: Started containerd.service - containerd container runtime.
```

sudo apt-get install -y socat

```
ubuntu@ip-172-31-86-43:~$ 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 6 not upgraded.
Need to get 374 kB of archives.
After this operation, 1649 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 socat amd64 1.8.0.0-4build3 [374 kB]
Fetched 374 kB in 0s (13.9 MB/s)
Selecting previously unselected package socat.
```

Step 6: Initialize the Kubecluster sudo kubeadm init --pod-network-cidr=10.244.0.0/16

```
ubuntu@ip-172-31-86-43:~$ 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'
W1003 13:13:28.336900    5640 checks.go:846] detected that the sandbox image "registry.k8s.io/pause:3.8" of the containe
r runtime is inconsistent with that used by kubeadm.It is recommended to use "registry.k8s.io/pause:3.10" as the CRI san
dbox image.
```

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

```
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 172.31.86.43:6443 --token oy5zuy.78xd696vk5gr8ip8 \
        --discovery-token-ca-cert-hash sha256:cb705f4200aa1d0a890b60cedb8802c8512842a85be9e9fd527799b09995c9ca
ubuntu@ip-172-31-86-43:~$ 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-172-31-86-43:~$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-fl
annel.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
```

Step 7: 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-172-31-86-43:~$ kubectl apply -f https://k8s.io/examples/application/deployment.yaml
deployment.apps/nginx-deployment created
```

kubectl get pods

```
ubuntu@ip-172-31-86-43:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-52k84    0/1     Pending   0          17s
nginx-deployment-d556bf558-d5fv5    0/1     Pending   0          17s
```

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

```
ubuntu@ip-172-31-86-43:~$ POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
kubectl port-forward $POD_NAME 8080:80
error: unable to forward port because pod is not running. Current status=Pending
```

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

```
ubuntu@ip-172-31-86-43:~$ kubectl taint nodes ip-172-31-86-43 node-role.kubernetes.io/control-plane:NoSchedule-
node/ip-172-31-86-43 untainted
ubuntu@ip-172-31-86-43:~$ kubectl get nodes
NAME              STATUS   ROLES           AGE     VERSION
ip-172-31-86-43   Ready    control-plane   4m30s   v1.31.1
```

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

```
ubuntu@ip-172-31-86-43:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-52k84   1/1     Running   0          3m51s
nginx-deployment-d556bf558-d5fv5   1/1     Running   0          3m51s
ubuntu@ip-172-31-86-43:~$ POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
kubectl port-forward $POD_NAME 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
```

Step 8: Verify your deployment Open up a new terminal and ssh to your EC2 instance. Then, use this curl command to check if the Nginx server is running.

```
Expanded Security Maintenance for Applications is not enabled.

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

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


Last login: Thu Oct  3 13:05:19 2024 from 110.226.182.217
ubuntu@ip-172-31-86-43:~$ curl --head http://127.0.0.1:8080
HTTP/1.1 200 OK
Server: nginx/1.14.2
Date: Thu, 03 Oct 2024 13:19:50 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

ubuntu@ip-172-31-86-43:~$
```

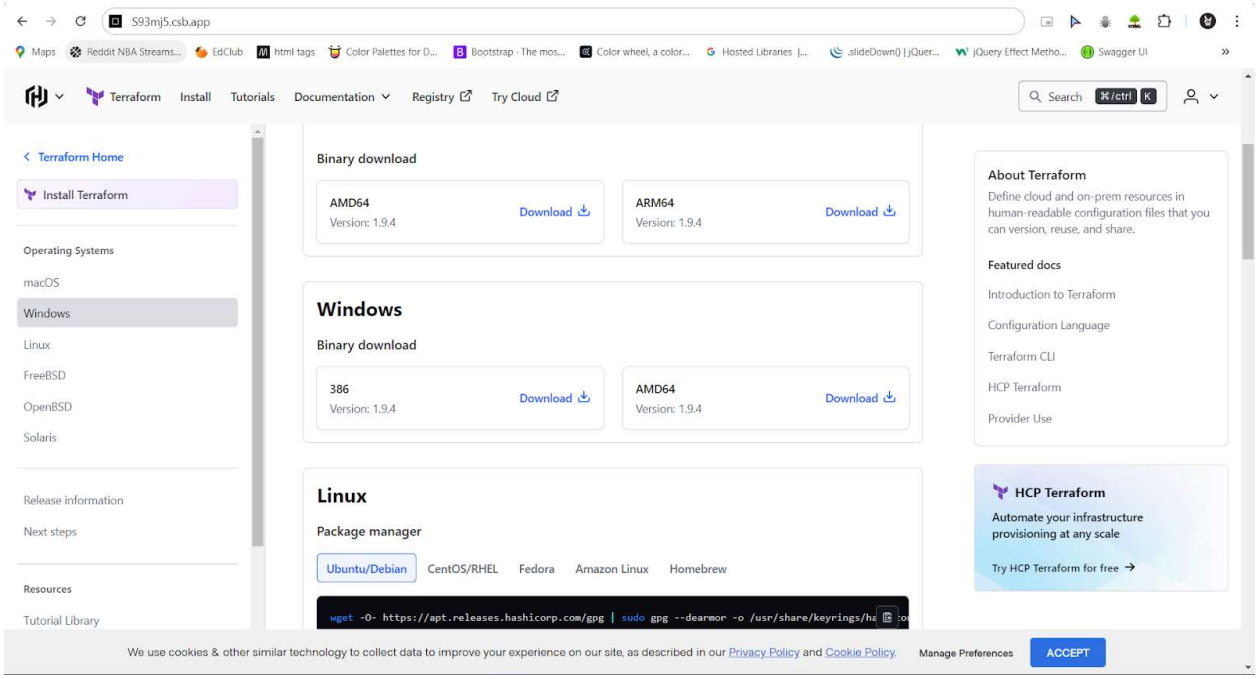
Conclusion:

In this experiment, we successfully set up a Kubernetes cluster on an AWS EC2 instance using kubectl and deployed a sample nginx application. By installing Docker and Kubernetes, configuring containerd, and initializing the cluster, we ensured smooth cluster operations. After deploying the nginx server, we verified its successful deployment by using kubectl port-forward and confirming access via curl. This process provided practical experience in managing Kubernetes clusters and deploying applications on cloud infrastructure.

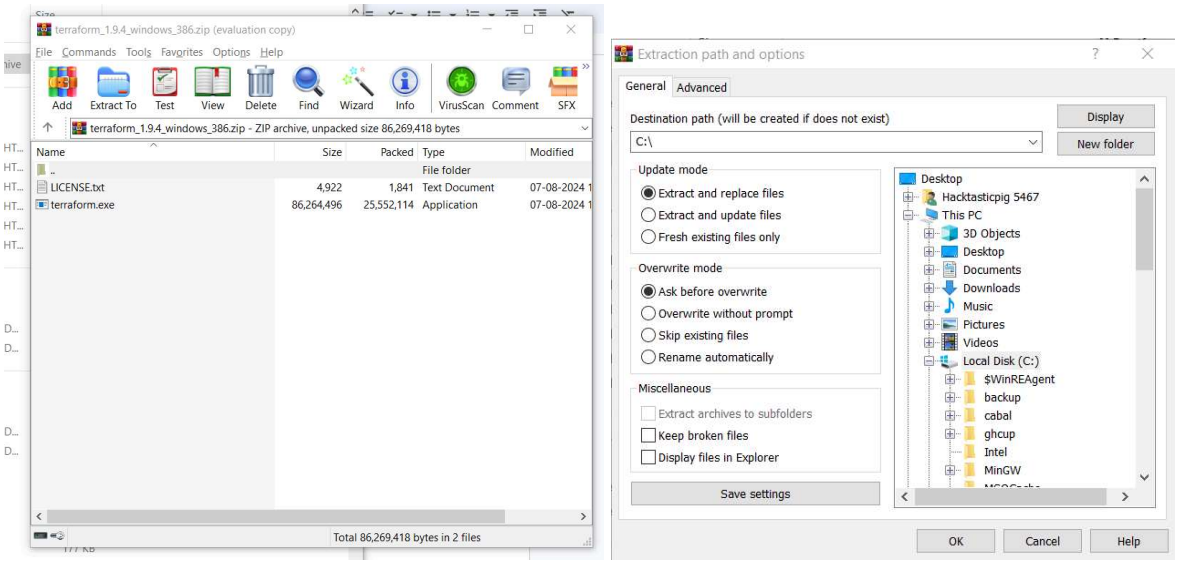
Experiment 5

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

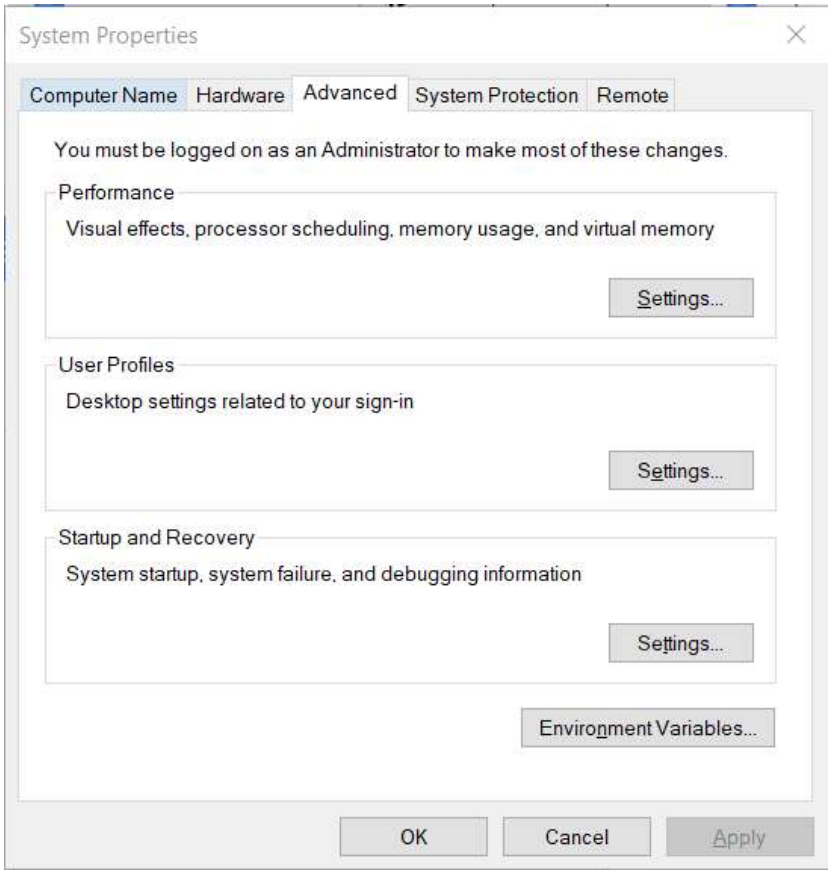
1. To install terraform, go to "https://developer.hashicorp.com/terraform/install". Visit the site and search for windows system download. You can either select 386 or AMD64 bit version.



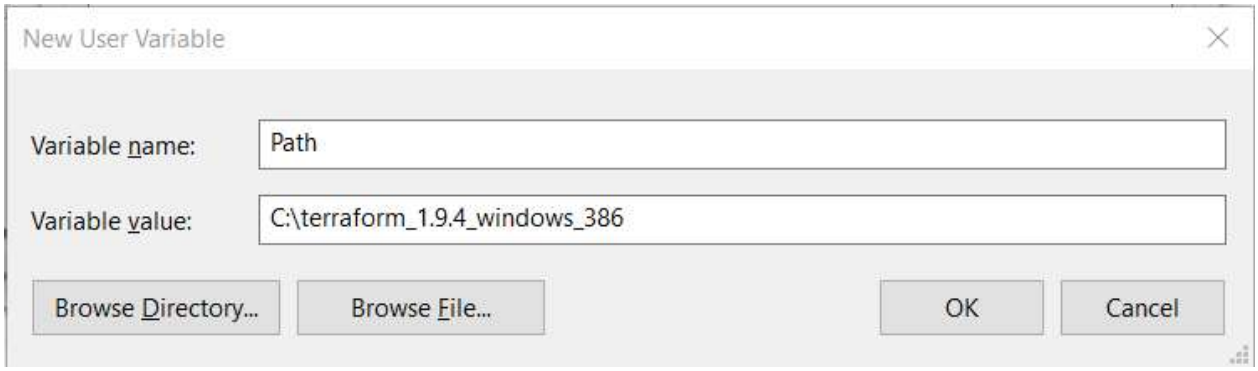
2. Extract the downloaded zip file in a suitable file in your pc.



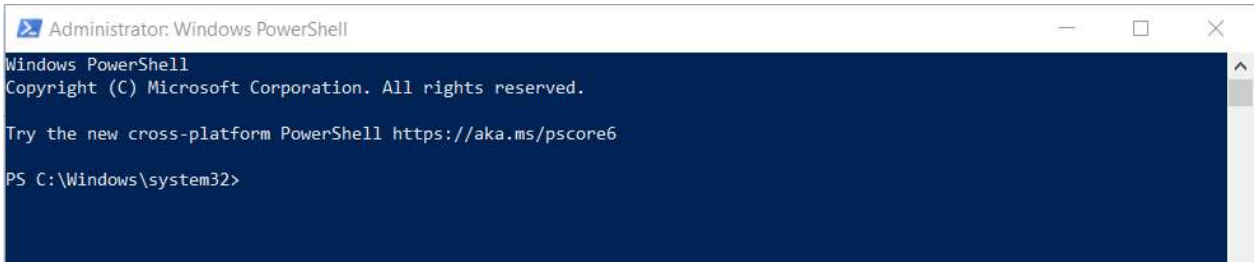
3. Copy the path where the file has been extracted. Now we need to setup the environment variable. Click on Environment variable in system properties.



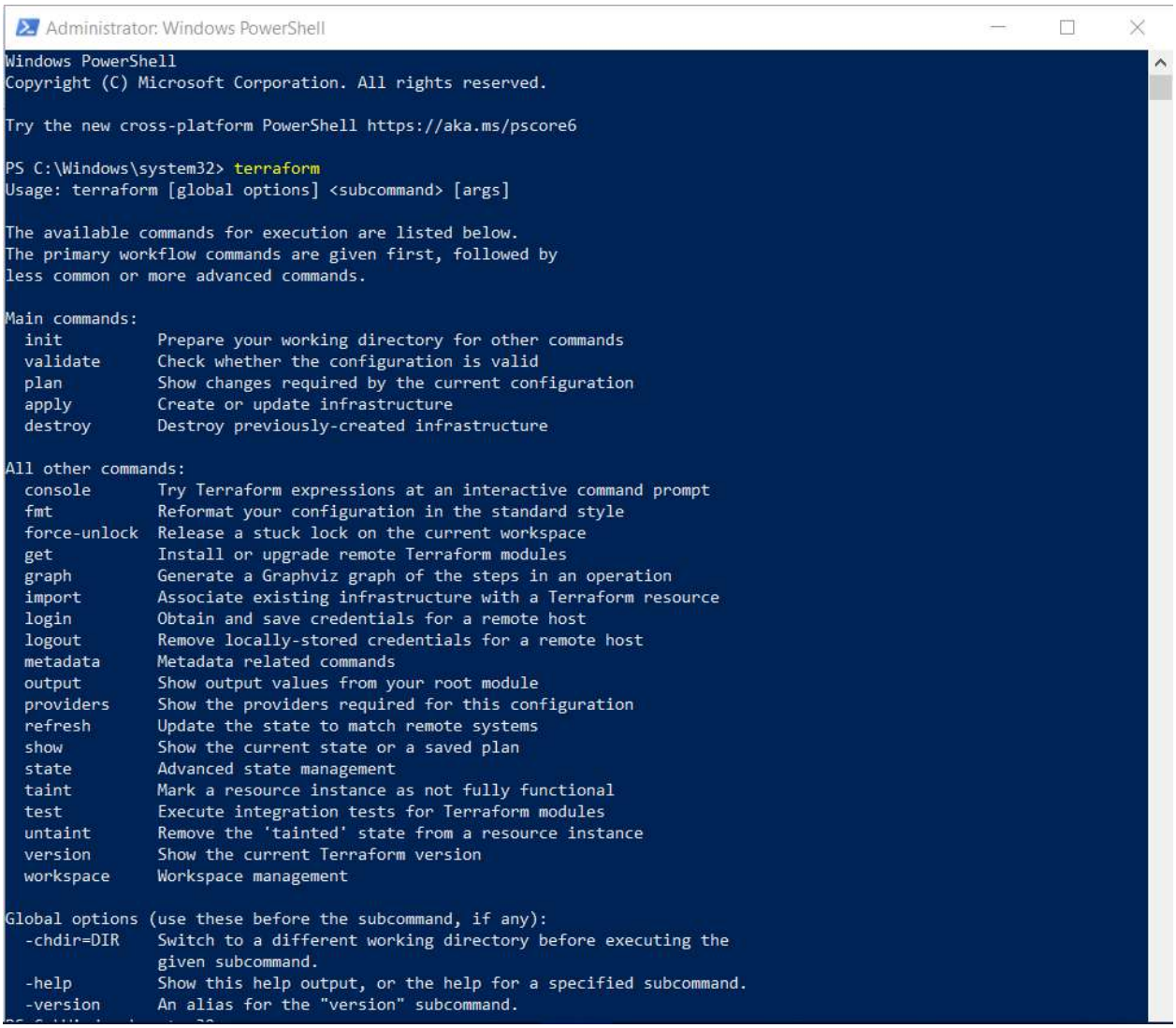
4. Create a new variable with name as Path and value as the path of the file where the terraform.exe file has been extracted, and save the changes.



5. Open the windows power shell as an administrator.



6. Type “terraform” in powershell. If the installation was successful and you have setup the path variable properly, you will be able to see list of available commands in terraform.



Experiment 6

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

1. Check docker installation by running the command “docker” and “docker --version”.

```
C:\Windows\system32>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
  compose*    Docker Compose
  container   Manage containers
  context     Manage contexts
  debug*      Get a shell into any image or container
  desktop*    Docker Desktop commands (Alpha)
  dev*        Docker Dev Environments
  extension*  Manages Docker extensions
  feedback*   Provide feedback, right in your terminal!
  image       Manage images
  init*       Creates Docker-related starter files for your project
  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
  scout*      Docker Scout
  system      Manage Docker
  trust       Manage trust on Docker images
  volume      Manage volumes

Swarm Commands:
  swarm       Manage Swarm
```

```
C:\Windows\system32>docker --version
Docker version 27.0.3, build 7d4bcd8

C:\Windows\system32>
```

2. Create a folder in your system named “terraformScripts”(Do not use terraform as name as it may recognize it as a keyword). Inside it create a subfolder docker. In this create a file docker.tf and type the following code in it.

docker.tf

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

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

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

# Create a container
resource "docker_container" "foo" {
  image = docker_image.ubuntu.image_id
  name  = "foo"
  command = ["sleep", "3600"]
}
```

3. Run the windows powershell as administrator. Navigate to the docker folder created in the above step. Run the terraform init command.

```
PS C:\Users\Lenovo\Desktop\TerraformScripts\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.
```

4. Terraform plan command is used to create a execution plan and see the resources.

```
PS C:\Users\Lenovo\Desktop\TerraformScripts\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          = [
          + "sleep",
          + "3600",
        ]
      + container_logs   = (known after apply)
      + entrypoint       = (known after apply)
      + env              = (known after apply)
      + exit_code        = (known after apply)
      + gateway          = (known after apply)
      + hostname         = (known after apply)
      + id               = (known after apply)
      + image            = (known after apply)
      + init             = (known after apply)
      + ip_address       = (known after apply)
      + ip_prefix_length = (known after apply)
      + ipc_mode         = (known after apply)
      + log_driver       = (known after apply)
      + logs             = false
      + must_run         = true
```

5. Docker image is command used to see all the images currently created in docker desktop

```
PS C:\Users\Lenovo\Desktop\TerraformScripts\Docker> docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
ubuntu       latest    edbfe74c41f8   3 weeks ago   78.1MB
PS C:\Users\Lenovo\Desktop\TerraformScripts\Docker>
```

6. Terraform apply command is used to execute the steps listed in terraform plan.

```
PS C:\Users\Lenovo\Desktop\TerraformScripts\Docker> terraform apply
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubun
tu:latest]
docker_container.foo: Refreshing state... [id=4de644086273a692820f646431610cae2f095755228f61eafc2bb712b2766040]

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          = [
          + "sleep",
          + "3600",
        ]
      + 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            = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a"
      + init             = (known after apply)
      + ip_address       = (known after apply)
      + ip_prefix_length = (known after apply)
```

7. Terraform destroy is used to destroy all resources that are currently being managed by terraform configuration.

```
PS C:\Users\Lenovo\Desktop\TerraformScripts\Docker> terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubun
tu:latest]
docker_container.foo: Refreshing state... [id=6d649285ef1d861ad451273401bc2771fe4e0be9b78f232aa64230ca0f58d36e]

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

Terraform will perform the following actions:

  # docker_container.foo will be destroyed
    resource "docker_container" "foo" {
        attach            = false -> null
        command           = [
            "sleep",
            "3600",
        ] -> null
        cpu_shares        = 0 -> null
        dns               = [] -> null
        dns_opts          = [] -> null
        dns_search        = [] -> null
        entrypoint        = [] -> null
        env               = [] -> null
        gateway           = "172.17.0.1" -> null
        group_add         = [] -> null
        hostname          = "6d649285ef1d" -> null
        id                = "6d649285ef1d861ad451273401bc2771fe4e0be9b78f232aa64230ca0f58d36e" -> null
        image             = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a" -> null
        init              = false -> null
        ip_address        = "172.17.0.2" -> null
        ip_prefix_length  = 16 -> null
        ipc_mode          = "private" -> null
        links             = [] -> null
        log_driver        = "json-file" -> null
        log_opts          = {} -> null
        logs              = false -> null
        max_retry_count   = 0 -> null
        memory            = 0 -> null
        memory_swap       = 0 -> null
        must_run          = true -> null
        name              = "foo" -> null
        network_data      = [
            {
```

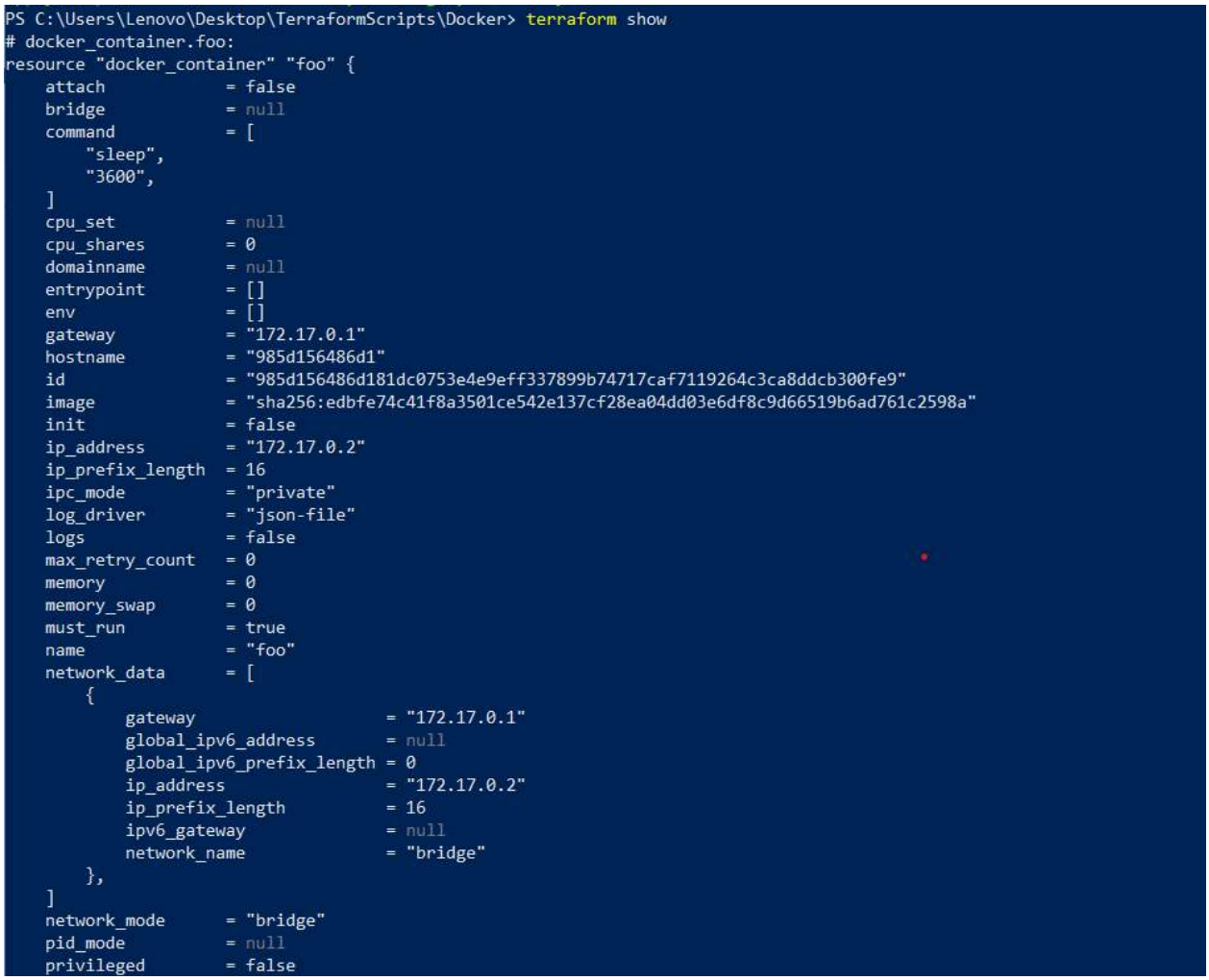
```
  # docker_image.ubuntu will be destroyed
    resource "docker_image" "ubuntu" {
        id          = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest" -> null
        image_id    = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a" ->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

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

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

  Enter a value: yes

docker_container.foo: Destroying... [id=6d649285ef1d861ad451273401bc2771fe4e0be9b78f232aa64230ca0f58d36e]
docker_container.foo: Destruction complete after 1s
docker_image.ubuntu: Destroying... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:lat
est]
docker_image.ubuntu: Destruction complete after 0s
```

```
Destroy complete! Resources: 2 destroyed.
PS C:\Users\Lenovo\Desktop\TerraformScripts\Docker> docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
```

8. Terraform validate is a command used to check for syntax and other errors in terraform configuration files.

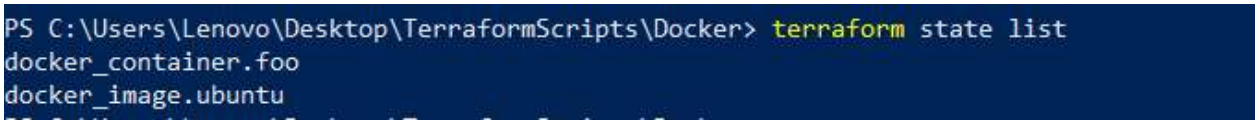
```
PS C:\Users\Lenovo\Desktop\TerraformScripts\Docker> terraform validate
Success! The configuration is valid.
```

9. The terraform show command in Terraform is used to display information about the state of resources managed by Terraform.



```
PS C:\Users\Lenovo\Desktop\TerraformScripts\Docker> terraform show
# docker_container.foo:
resource "docker_container" "foo" {
    attach            = false
    bridge            = null
    command           = [
        "sleep",
        "3600",
    ]
    cpu_set           = null
    cpu_shares        = 0
    domainname        = null
    entrypoint        = []
    env               = []
    gateway           = "172.17.0.1"
    hostname          = "985d156486d1"
    id                = "985d156486d181dc0753e4e9eff337899b74717caf7119264c3ca8ddcb300fe9"
    image             = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a"
    init              = false
    ip_address        = "172.17.0.2"
    ip_prefix_length  = 16
    ipc_mode          = "private"
    log_driver        = "json-file"
    logs              = false
    max_retry_count   = 0
    memory            = 0
    memory_swap       = 0
    must_run          = true
    name              = "foo"
    network_data      = [
        {
            gateway                   = "172.17.0.1"
            global_ipv6_address       = null
            global_ipv6_prefix_length = 0
            ip_address                = "172.17.0.2"
            ip_prefix_length          = 16
            ipv6_gateway              = null
            network_name              = "bridge"
        },
    ]
    network_mode      = "bridge"
    pid_mode          = null
    privileged        = false
```

10. The terraform state list command is used to list all the resources currently managed by your Terraform state



```
PS C:\Users\Lenovo\Desktop\TerraformScripts\Docker> terraform state list
docker_container.foo
docker_image.ubuntu
```

Experiment 7

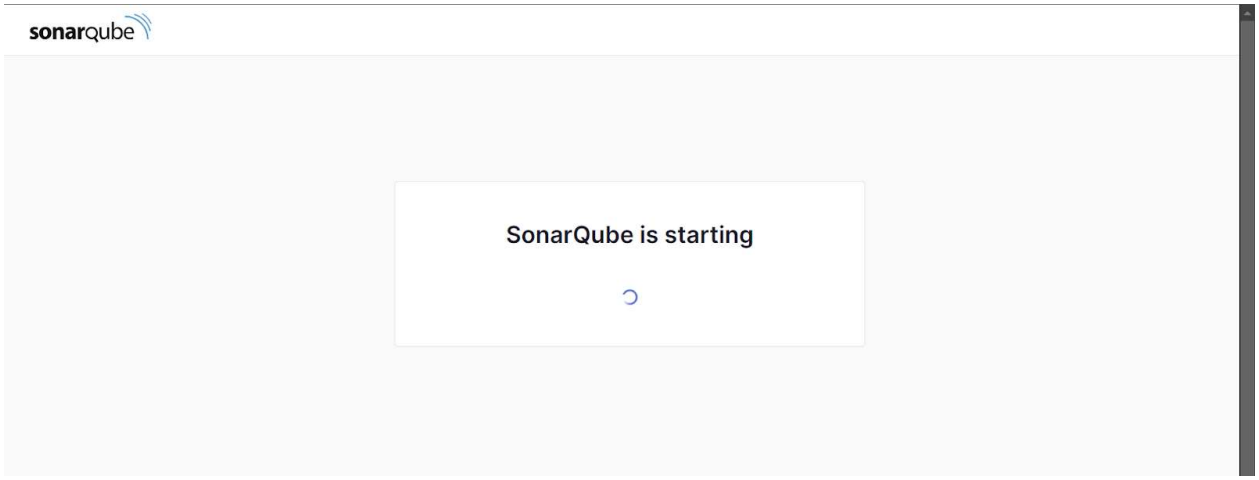
Aim: To understand Static Analysis SAST process and learn to integrate Jenkins SAST to SonarQube/GitLab.

1. Open docker desktop on your device. Then run the following command in powershell.

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

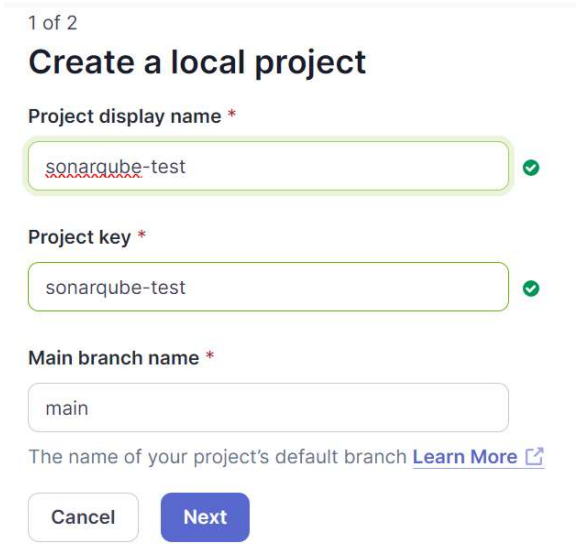
```
PS C:\Windows\system32> docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:
latest
Unable to find image 'sonarqube:latest' locally
latest: Pulling from library/sonarqube
7478e0ac0f23: Pull complete
90a925ab929a: Pull complete
7d9a34308537: Pull complete
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
5008a1b64daaff9ca9430d6426a4ceb303d964d039798555e9fb548ff7c89073
```

2. Once the command is run, you can verify running of sonarqube by checking the url http://localhost:9000



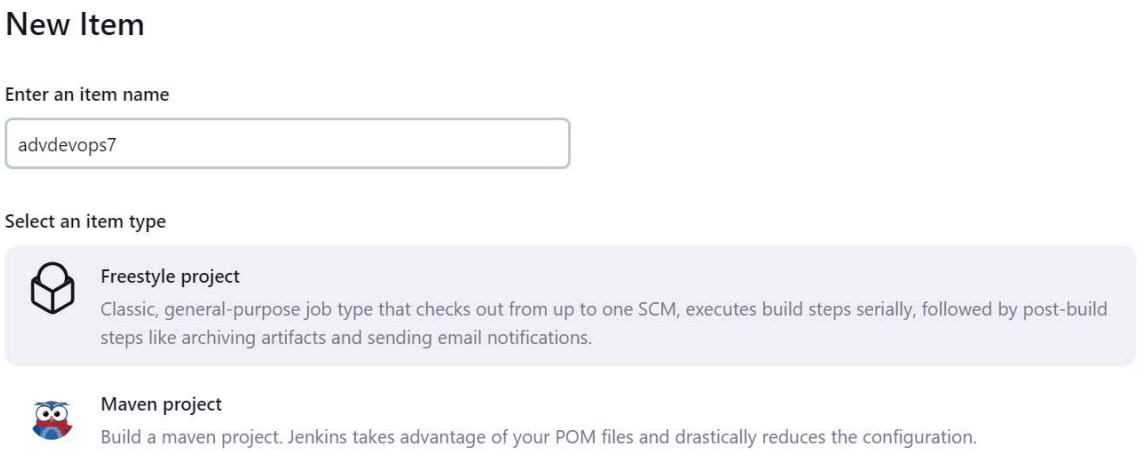
3. Login with username admin and password admin

4. Create a local project in sonarqube. Give the project a name, setup the project and click on create project.



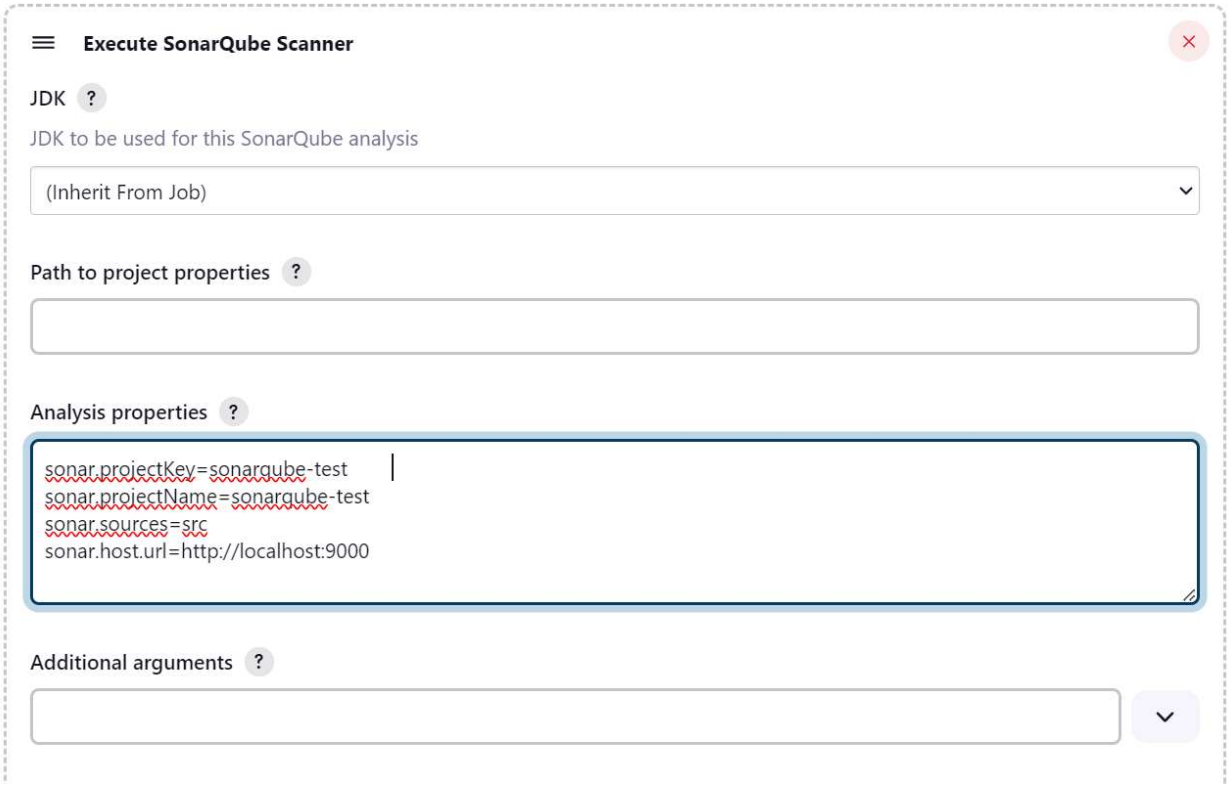
5. Go to jenkins, i.e http://localhost:8080. Go to Manage Jenkins -> Plugins -> Available Plugins. Search for SonarQube Scanner and install it.

6. Search for SonarQube Scanner under Global Tool Configuration. Choose the latest configuration and choose Install automatically.
7. Create a new project. Select the type as freestyle project.

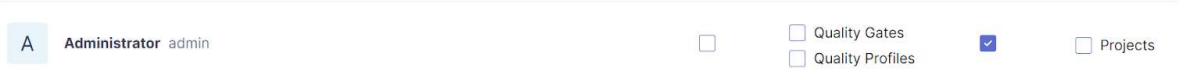


8. Select git in source code management. Enter the following as the url for repository https://github.com/shazforiot/MSBuild_firstproject.git
   It is a simple hello world project with no vulnerabilities and issues.

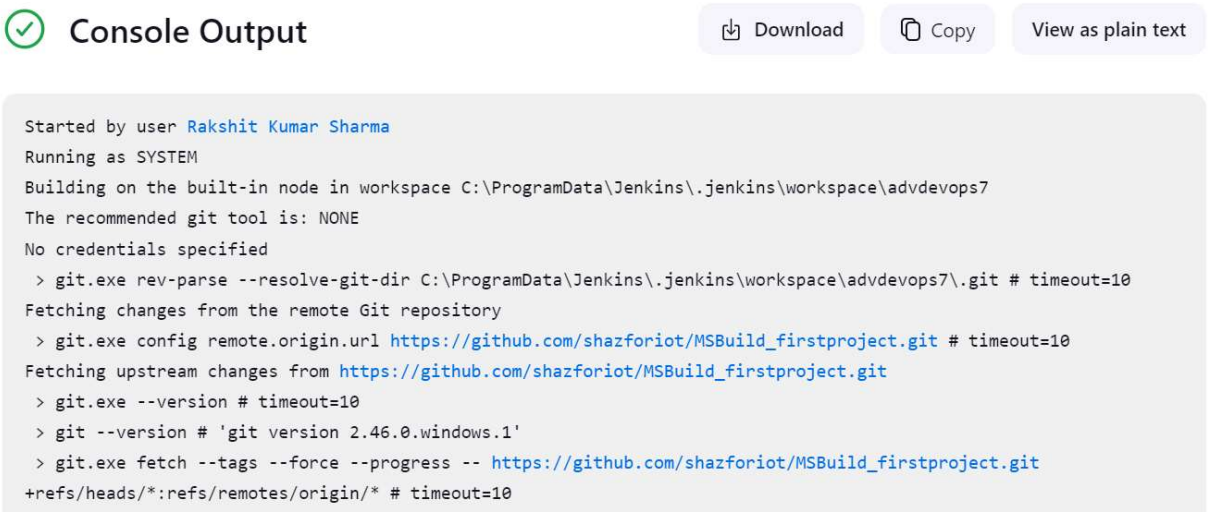
9. In build steps select Execute SonarQube Scanner. In analysis properties give the details about the sonarqube project.



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

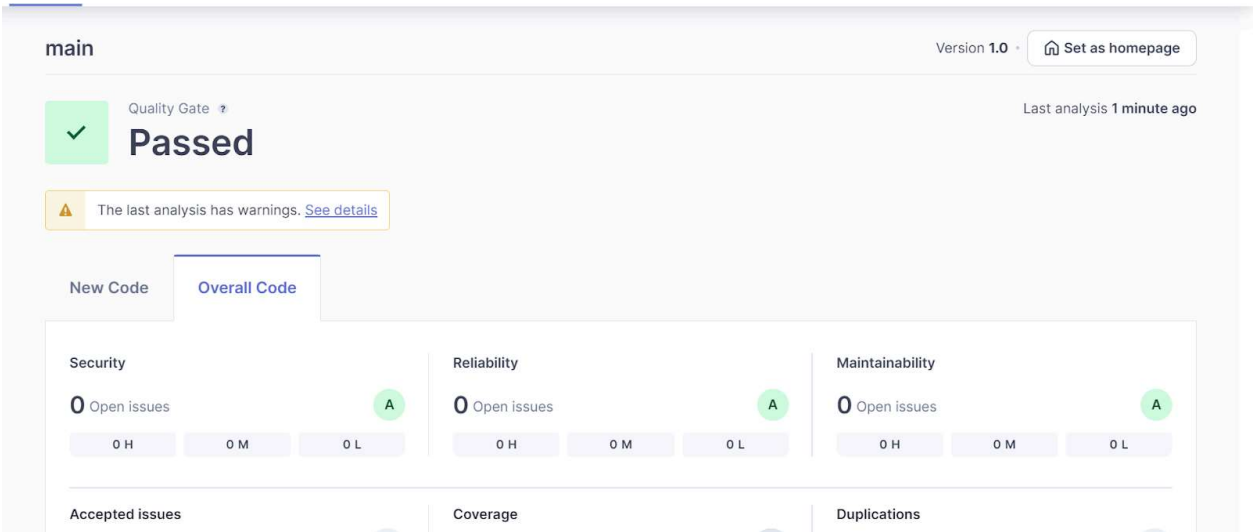


11. Once everything is set up, Click on Build now to build the freestyle project.



Check the console output to see the successful build of the project.

12. Once Build is complete, check the sonarqube project. You will see a passed message, this indicated the code did not have any issues while building.



Conclusion:

In this experiment, we successfully set up a static application security testing (SAST) pipeline by integrating SonarQube with Jenkins. We initiated a SonarQube server using Docker, allowing for local access and code analysis. A new project was created in SonarQube to facilitate static code evaluation. We configured Jenkins by installing the SonarQube Scanner plugin and created a Freestyle project linked to a sample GitHub repository. By executing the SonarQube analysis within the Jenkins build steps, we provide

Experiment 8

Aim: 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.

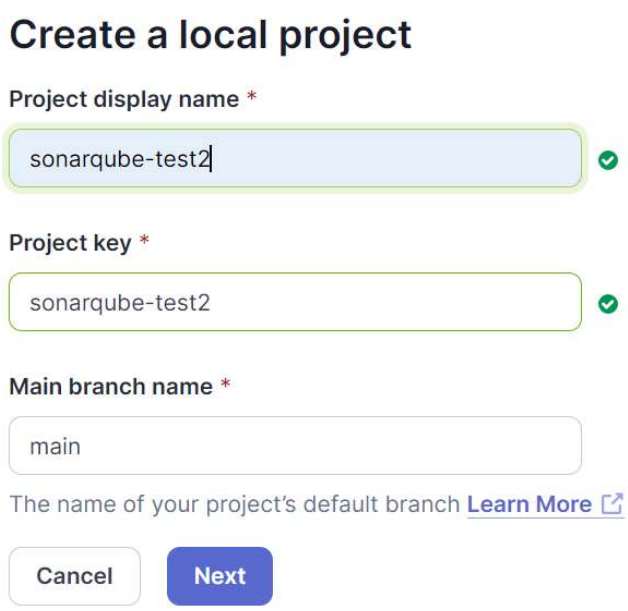
1. Open up the jenkins dashboard. Usually on localhost: 8080
2. Run SonarQube in a Docker container using this command -

```
PS C:\Windows\system32> 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
5008a1b64daaff9ca9430d6426a4ceb303d964d039798555e9fb548ff7c89073
```

3. Once the container is running, you can check it on localhost:9000. If you have already made a container named sonarqube, run the following command instead.

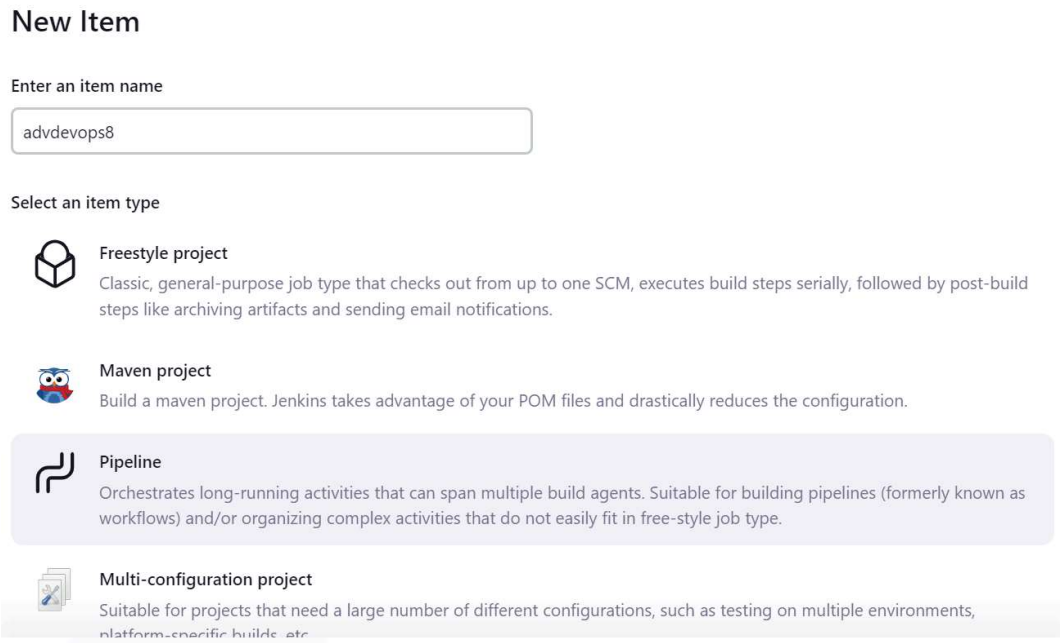
```
PS C:\Windows\system32> docker start sonarqube
>>
sonarqube
```

4. Login to sonarqube with username admin, password admin, or if you have setup any other credentials use that.
5. Create a local project. Give it a name(sonarqube-test2). Configure the project and click on create.



6. Now we need to install the SonarQube CLI.
   Go to following link:https://docs.sonarsource.com/sonarqube/latest/analyzing-source-code/scanners/sonarscanner/
   Download the suitable version compatible with your device.
   Extract the zip file downloaded. Inside the folder go to bin/sonar-scanner, copy the path of this file.

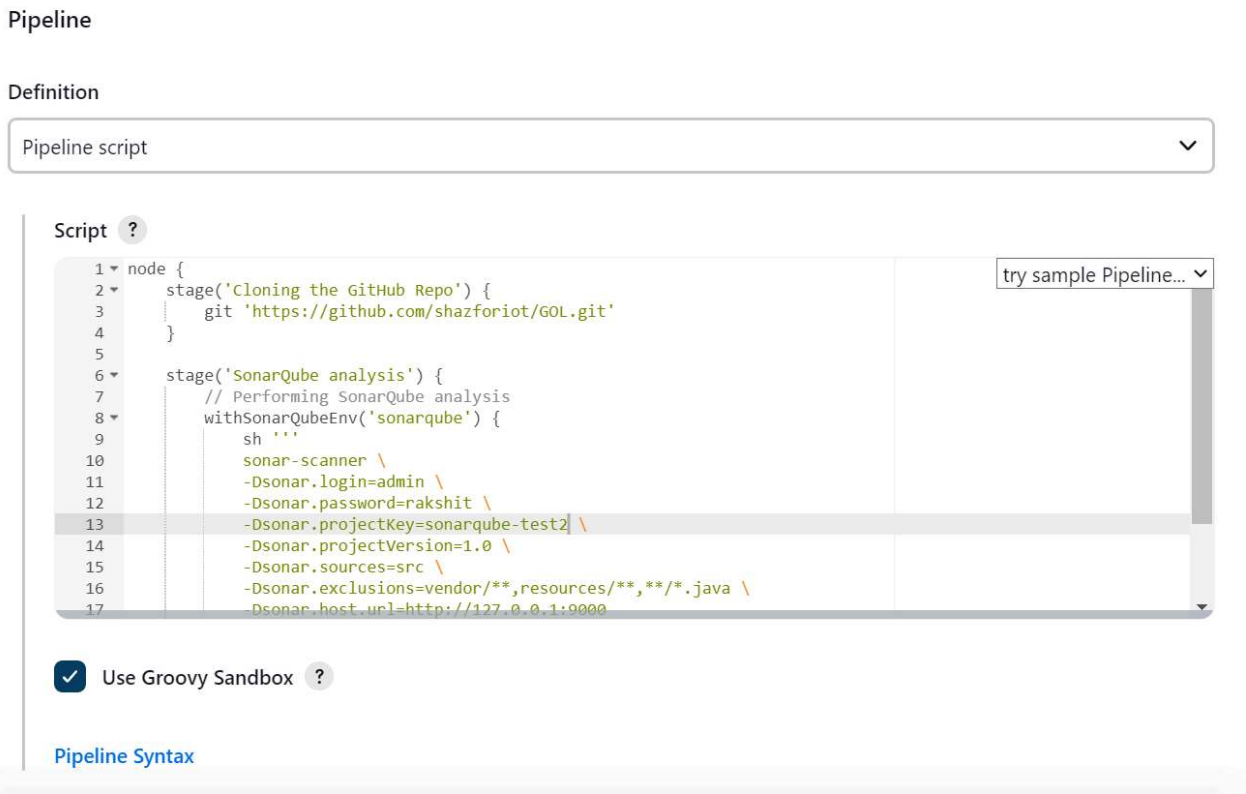
7. Create a new project in Jenkins. Give it a name and select the project type as pipeline in Jenkins.



8. Inside the pipeline script, give the following script. Use the path copied in earlier step instead of Path to sonarqube folder.

```
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/"
    }
  }
}
```

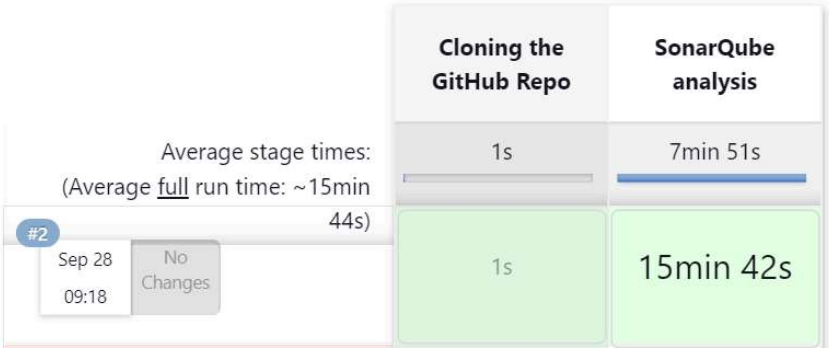
Click on save to create the pipeline.



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

9. Run the build.

Stage View


Cloning the GitHub Repo
SonarQube analysis
Average stage times:
(Average full run time: ~15min 44s)
1s
7min 51s
#2
Sep 28
09:18
No Changes
1s
15min 42s

10. Check the console output

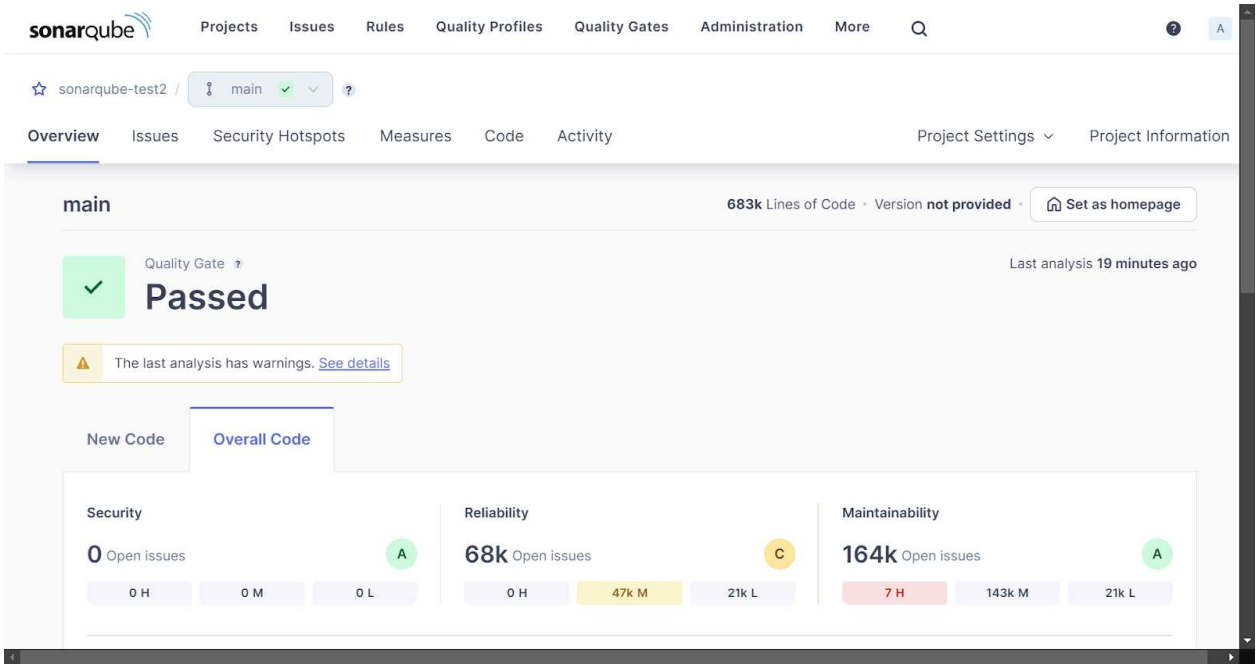
Console Output

Download Copy View as plain text

Skipping 4,249 KB.. Full Log

```
09:27:10.137 WARN  Too many duplication references on file gameoflife-
web/tools/jmeter/docs/api/org/apache/jmeter/gui/ReportMainFrame.WindowHappenings.html for block at line 212. Keep
only the first 100 references.
09:27:10.137 WARN  Too many duplication references on file gameoflife-
web/tools/jmeter/docs/api/org/apache/jmeter/gui/ReportMainFrame.WindowHappenings.html for block at line 215. Keep
only the first 100 references.
09:27:10.137 WARN  Too many duplication references on file gameoflife-
web/tools/jmeter/docs/api/org/apache/jmeter/gui/ReportMainFrame.WindowHappenings.html for block at line 212. Keep
only the first 100 references.
09:27:10.137 WARN  Too many duplication references on file gameoflife-
web/tools/jmeter/docs/api/org/apache/jmeter/gui/ReportMainFrame.WindowHappenings.html for block at line 296. Keep
only the first 100 references.
09:27:10.137 WARN  Too many duplication references on file gameoflife-
web/tools/jmeter/docs/api/org/apache/jmeter/gui/ReportMainFrame.WindowHappenings.html for block at line 17. Keep
only the first 100 references.
09:27:10.137 WARN  Too many duplication references on file gameoflife-
web/tools/jmeter/docs/api/org/apache/jmeter/gui/ReportMainFrame.WindowHappenings.html for block at line 212. Keep
only the first 100 references.
09:27:10.137 WARN  Too many duplication references on file gameoflife-
web/tools/jmeter/docs/api/org/apache/jmeter/gui/ReportMainFrame.WindowHappenings.html for block at line 215. Keep
09:27:15.430 INFO  CPD Executor CPD calculation finished (done) | time=28736ms
09:27:15.451 INFO  SCM revision ID 'ba799ba7e1b576f04a4612322b0412c5e6e1e5e4'
09:30:22.050 INFO  Analysis report generated in 138747ms, dir size=127.2 MB
09:33:36.168 INFO  Analysis report compressed in 194118ms, zip size=29.6 MB
09:33:36.981 INFO  Analysis report uploaded in 813ms
09:33:36.982 INFO  ANALYSIS SUCCESSFUL, you can find the results at: http://127.0.0.1:9000/dashboard?id=sonarqube-
test2
09:33:36.982 INFO  Note that you will be able to access the updated dashboard once the server has processed the
submitted analysis report
09:33:36.982 INFO  More about the report processing at http://127.0.0.1:9000/api/ce/task?id=0fd50499-d2c7-460f-
a5c0-06a3b9908b8b
09:33:43.711 INFO  Analysis total time: 15:38.015 s
09:33:43.729 INFO  SonarScanner Engine completed successfully
09:33:44.597 INFO  EXECUTION SUCCESS
09:33:44.597 INFO  Total time: 15:41.397s
[Pipeline] }
[Pipeline] // withSonarQubeEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

11. Since the build is successful, now check the project in sonarqube.

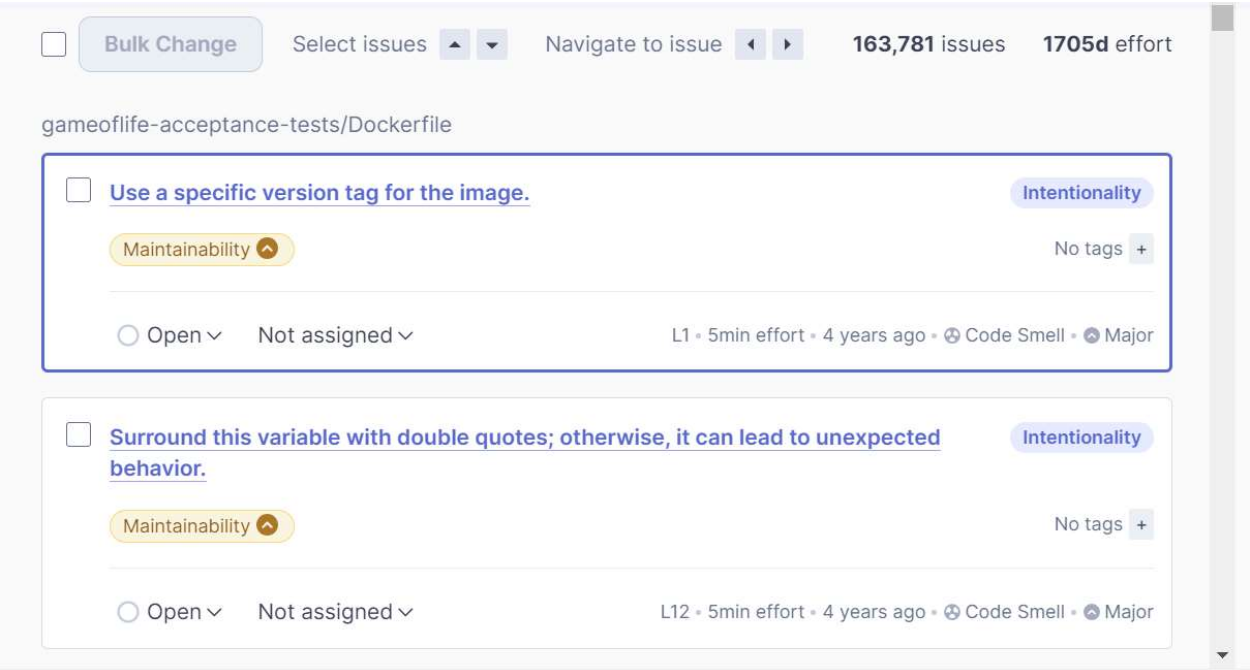


You will be able to see different issues with the code under different tabs listed.

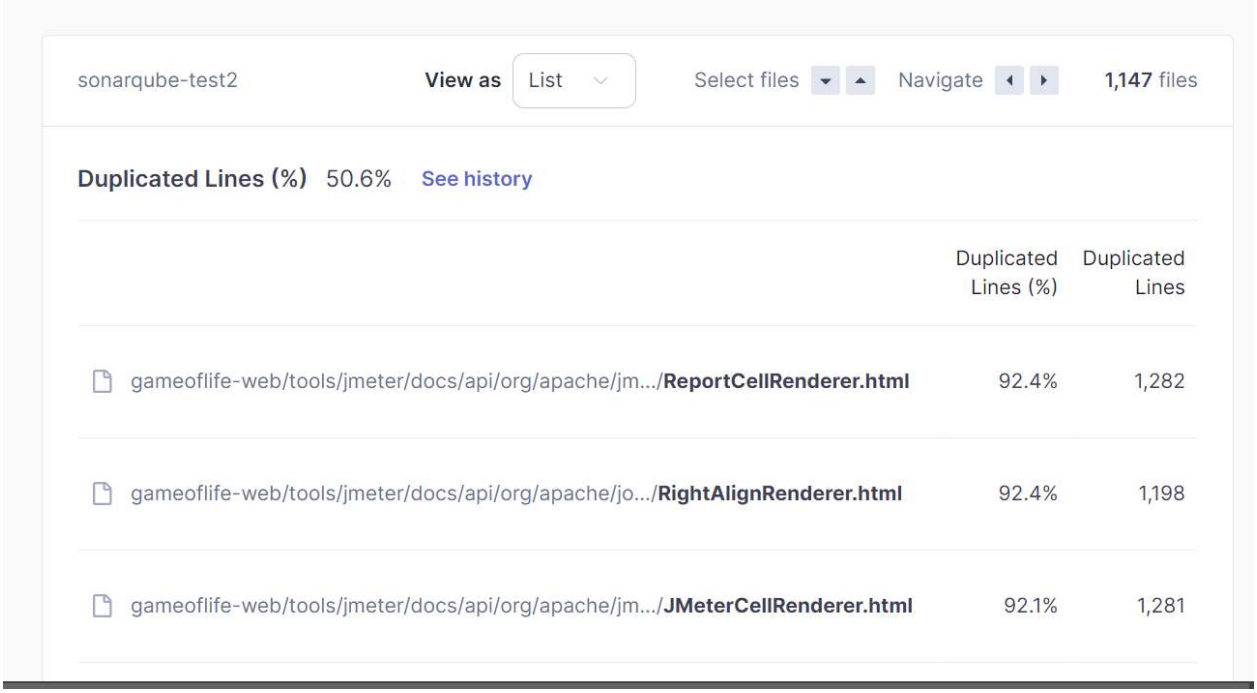
12. Code problems:

Reliability:

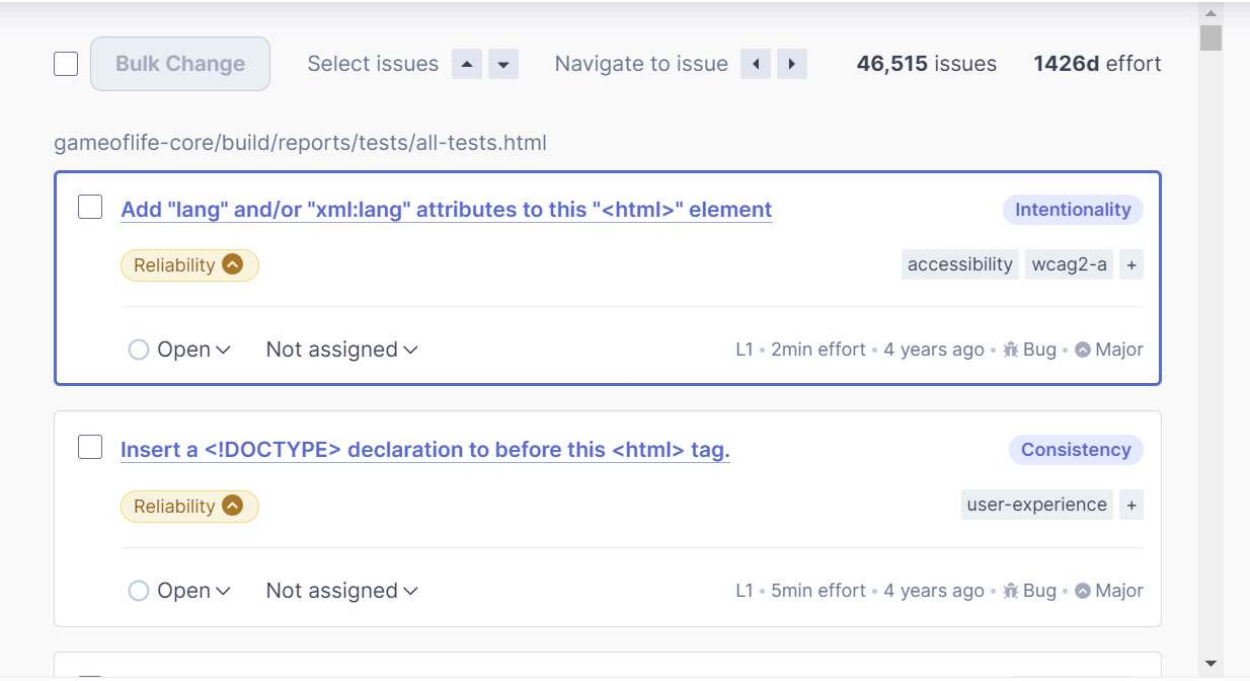
Maintainability:



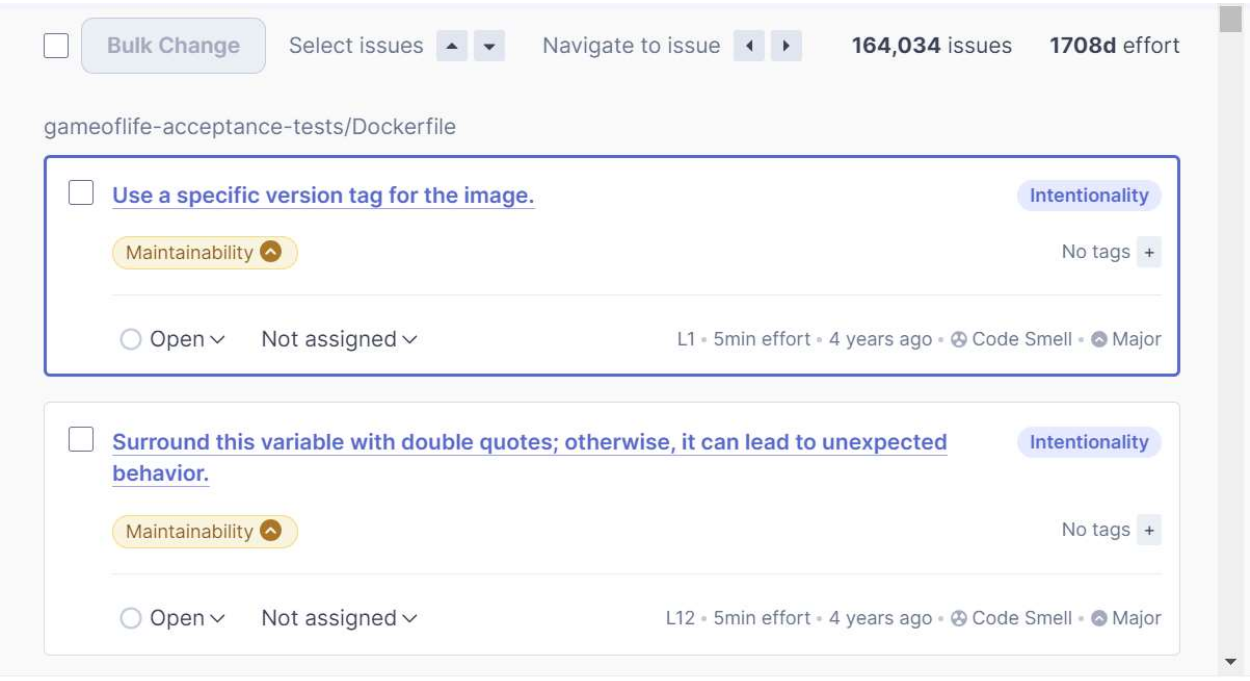
Duplications:



Bug:



Code smells:



And various other issues or problems can be seen in the sonarqube project dashboard.

Conclusion:

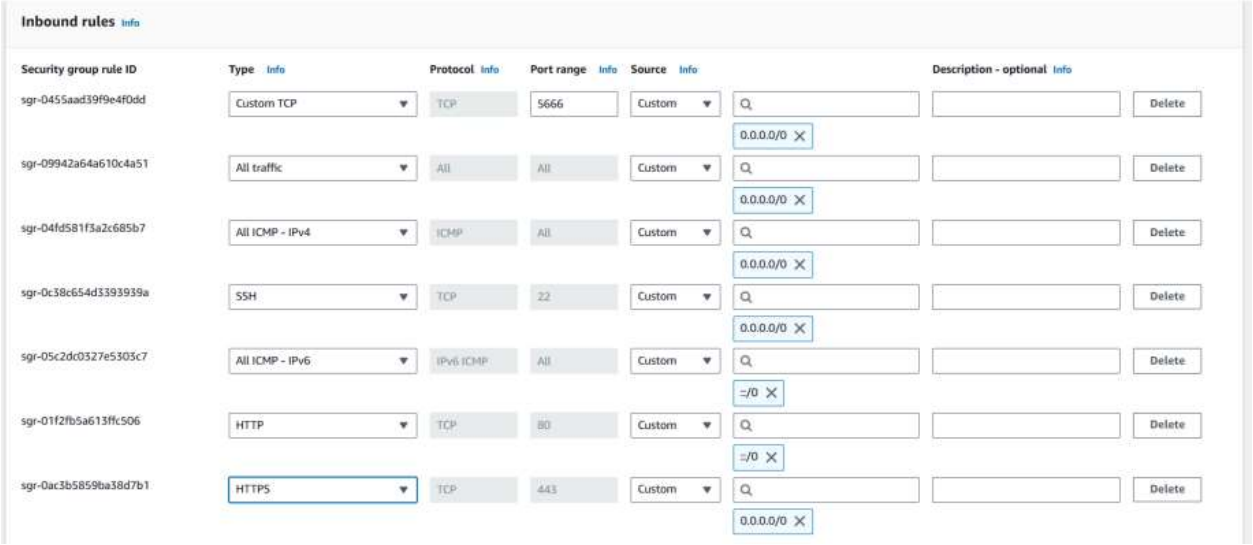
In this experiment, a Jenkins CI/CD pipeline was successfully integrated with SonarQube to perform static code analysis on a sample application. SonarQube, running in a Docker

container, analyzed the code for issues like bugs, code smells, and security vulnerabilities. The pipeline was automated by cloning the source code from a GitHub repository and using the SonarQube Scanner to detect potential code problems. After a successful Jenkins build, the SonarQube project dashboard provided information about code's reliability, maintainability, duplications, and detected bugs, offering feedback that help improve the code quality.

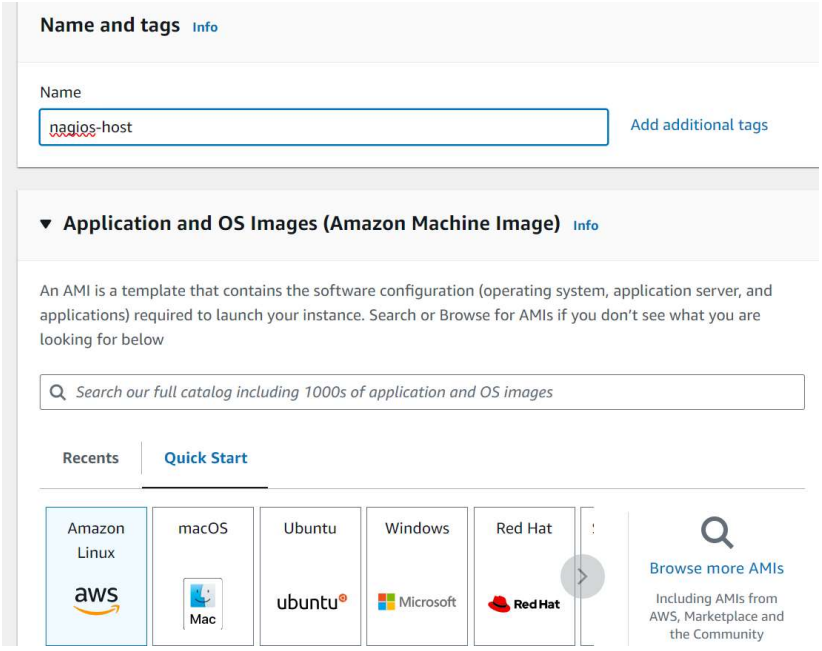
Experiment 9

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

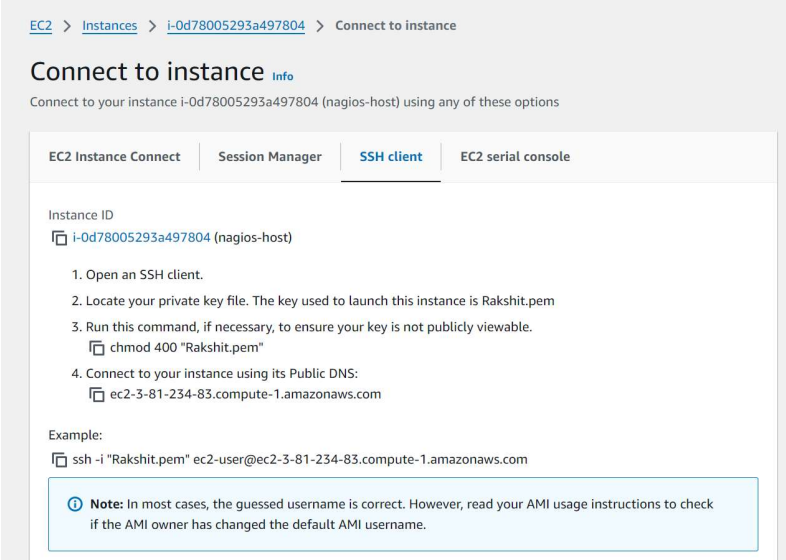
1. Firstly, go to the EC2 section in the Aws academy lab. Go to the security group and click on create a new security group. Add the following inbound rules in your security group and give it an appropriate name.



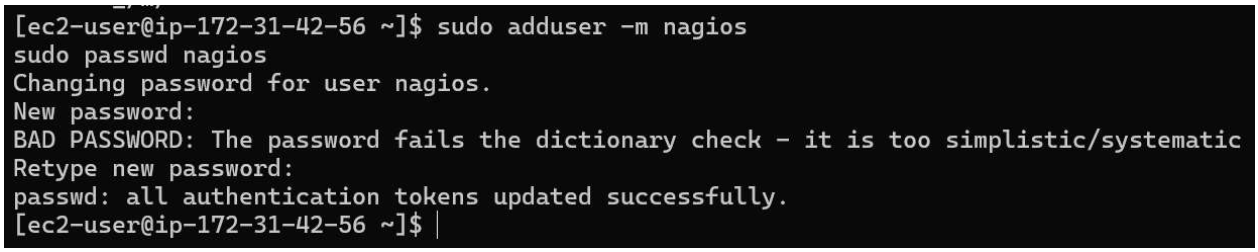
2. Now go to the EC2 instance dashboard and click on Launch Instance. Name your instance as nagios-host, select Amazon Linux as the instance type. Create a new key pair and download the corresponding .pem file in your pc. Also choose the existing security group which we created in the previous step.



3. Now click on the instance id in the dashboard and click on connect. Go to the ssh client tab and copy the example command. Locate the folder in which your .pem file was downloaded and open it in terminal. Run the copied command in the terminal to connect to your ec2 instance.



4. Run the commands:
   sudo adduser -m nagios
   sudo passwd nagios
   This creates a user named ‘nagios’, ensures it has a home directory and sets up a password for it.



5. Create a user group named ‘nagcmd’ to execute nagios commands.

sudo groupadd nagcmd
sudo usermod -a -G nagcmd nagios
sudo usermod -a -G nagcmd apache
Add users apache and nagios to this user group.

```
[ec2-user@ip-172-31-42-56 ~]$ sudo groupdel nagcmd
[ec2-user@ip-172-31-42-56 ~]$ sudo groupadd nagcmd
sudo usermod -a -G nagcmd nagios
sudo usermod -a -G nagcmd apache
[ec2-user@ip-172-31-42-56 ~]$
```

6. Cd into the downloads folder, make one if there is no such directory and run the following command,
   cd ~/downloads
   wget https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.5.tar.gz
   wget https://nagios-plugins.org/download/nagios-plugins-2.4.11.tar.gz
   Run the last command to install the latest version of nagios

```
[ec2-user@ip-172-31-42-56 downloads]$ wget https://go.nagios.org/l/975333/2024-09-17/6kqcx
--2024-10-08 01:20:02--  https://go.nagios.org/l/975333/2024-09-17/6kqcx
Resolving go.nagios.org (go.nagios.org)... 3.215.172.219, 3.92.120.28, 18.208.125.13, ...
Connecting to go.nagios.org (go.nagios.org)|3.215.172.219|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: http://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.5.tar.gz?utm_source=Nagios.org&utm_content=Download+Form&utm_campaign=Core+4.5.5+Download+&pi_content=1e9662c93afb2ed6bd2e3f3cc38771a7f01125e969f2a75b0e2254439d4a81d8 [following]
--2024-10-08 01:20:02--  http://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.5.tar.gz?utm_source=Nagios.org&utm_content=Download+Form&utm_campaign=Core+4.5.5+Download+&pi_content=1e9662c93afb2ed6bd2e3f3cc38771a7f01125e969f2a75
```

```
[ec2-user@ip-172-31-42-56 downloads]$ wget https://nagios-plugins.org/download/nagios-plugins-2.4.11.tar.gz
--2024-10-08 01:20:28--  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  6.29MB/s    in 0.4s

2024-10-08 01:20:28 (6.29 MB/s) - 'nagios-plugins-2.4.11.tar.gz' saved [2753049/2753049]

[ec2-user@ip-172-31-42-56 downloads]$
```

7. tar zxvf nagios-4.5.5.tar.gz This extracts the nagios-core files into the same directory using the tar command.

```
[ec2-user@ip-172-31-42-56 downloads]$ tar zxvf 6kqcx
nagios-4.5.5/
nagios-4.5.5/.github/
nagios-4.5.5/.github/workflows/
nagios-4.5.5/.github/workflows/test.yml
nagios-4.5.5/.gitignore
nagios-4.5.5/CONTRIBUTING.md
nagios-4.5.5/Changelog
nagios-4.5.5/INSTALLING
nagios-4.5.5/LEGAL
nagios-4.5.5/LICENSE
nagios-4.5.5/Makefile.in
nagios-4.5.5/README.md
nagios-4.5.5/THANKS
nagios-4.5.5/UPGRADING
nagios-4.5.5/aclocal.m4
nagios-4.5.5/autoconf-macros/
nagios-4.5.5/autoconf-macros/.gitignore
nagios-4.5.5/autoconf-macros/CHANGELOG.md
nagios-4.5.5/autoconf-macros/LICENSE
nagios-4.5.5/autoconf-macros/LICENSE.md
nagios-4.5.5/autoconf-macros/README.md
```

8. Cd into the nagios folder and run the following command.
   ./configure --with-command-group=nagcmd
   This command ensures that Nagios uses a specific group (in this case, nagcmd) for executing external commands.

```
[ec2-user@ip-172-31-42-56 downloads]$ cd nagios-4.5.5
[ec2-user@ip-172-31-42-56 nagios-4.5.5]$
```

```
[ec2-user@ip-172-31-42-56 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 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-172-31-42-56 nagios-4.5.5]$
```

Error occurs which says that ssl headers cannot be found. To fix the above error, run the ‘sudo yum install openssl-devel’ command.
Then, run the ‘./configure --with-command-group=nagcmd’ command again.

```
[ec2-user@ip-172-31-42-56 nagios-4.5.5]$ sudo yum install openssl-devel
Last metadata expiration check: 0:11:54 ago on Tue Oct  8 01:14:05 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
```

9. Now we need to compile all the components of this software as given in the instructions. Run the command
   make all,

   After that run the following commands:
   sudo make install
   sudo make install-init
   sudo make install-config
   sudo make install-commandmode

```
[ec2-user@ip-172-31-42-56 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
```

```
[ec2-user@ip-172-31-42-56 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
[ec2-user@ip-172-31-42-56 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
```

10. We need to update the email linked with this server to our email for it to send notifications.
    sudo nano /usr/local/nagios/etc/objects/contacts.cfg

```
  GNU nano 5.8                        /usr/local/nagios/etc/objects/contacts.cfg                        Modified
# Just one contact defined by default - the Nagios admin (that's you)
# This contact definition inherits a lot of default values from the
# 'generic-contact' template which is defined elsewhere.

define contact {

    contact_name            nagiosadmin             ; Short name of user
    use                     generic-contact         ; Inherit default values from generic-contact template (defined abo>
    alias                   Nagios Admin            ; Full name of user
    email                   2022.rakshit.sharma@ves.ac.in ; <<***** CHANGE THIS TO YOUR EMAIL ADDRESS ******
}



###############################################################################
#
# CONTACT GROUPS
#
###############################################################################

# We only have one contact in this simple configuration file, so there is
# no need to create more than one contact group.

define contactgroup {

    contactgroup_name       admins
```

11. sudo make install-webconf
    This installs the necessary configuration files for the Nagios web interface.

```
[ec2-user@ip-172-31-42-56 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-172-31-42-56 nagios-4.5.5]$
```

12. sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
    This would create a user named ‘nagiosadmin’ to access the nagios web interface. Save the password that you create as it will be needed later.

```
[ec2-user@ip-172-31-42-56 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-172-31-42-56 nagios-4.5.5]$
```

13. cd ~/downloads
    tar zxvf nagios-plugins-2.4.11.tar.gz
    This will extract the files for nagios plugins.

```
[ec2-user@ip-172-31-42-56 nagios-4.5.5]$ cd ~/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
```

14. Next, we must compile all components of this software according to the instructions in the Makefile. To do so, use the following commands:
    make
    sudo make install

15. sudo chkconfig --add nagios
    sudo chkconfig nagios on

This registers the Nagios service with the system ensuring that it can manage the server status.

```
[ec2-user@ip-172-31-42-133 nagios-plugins-2.4.11]$ sudo chkconfig --add nagios
sudo chkconfig nagios on
error reading information on service nagios: No such file or directory
Note: Forwarding request to 'systemctl enable nagios.service'.
Created symlink /etc/systemd/system/multi-user.target.wants/nagios.service → /usr/lib/systemd/system/nagios.service.
[ec2-user@ip-172-31-42-133 nagios-plugins-2.4.11]$
```

16. sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg
    Run the following command to check and verify if the configuration files are correct or not.
    If you see total errors and warnings as 0, it means the configurations are correct.

```
[ec2-user@ip-172-31-42-133 nagios-plugins-2.4.11]$ sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

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

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

Running pre-flight check on configuration data...
        Checked 0 service escalations.
Checking for circular paths...
        Checked 1 hosts
        Checked 0 service dependencies
        Checked 0 host dependencies
        Checked 5 timeperiods
Checking global event handlers...
Checking obsessive compulsive processor commands...
Checking misc settings...

Total Warnings: 0
Total Errors:   0
```

17. Run the command to start nagios service
    sudo service nagios start

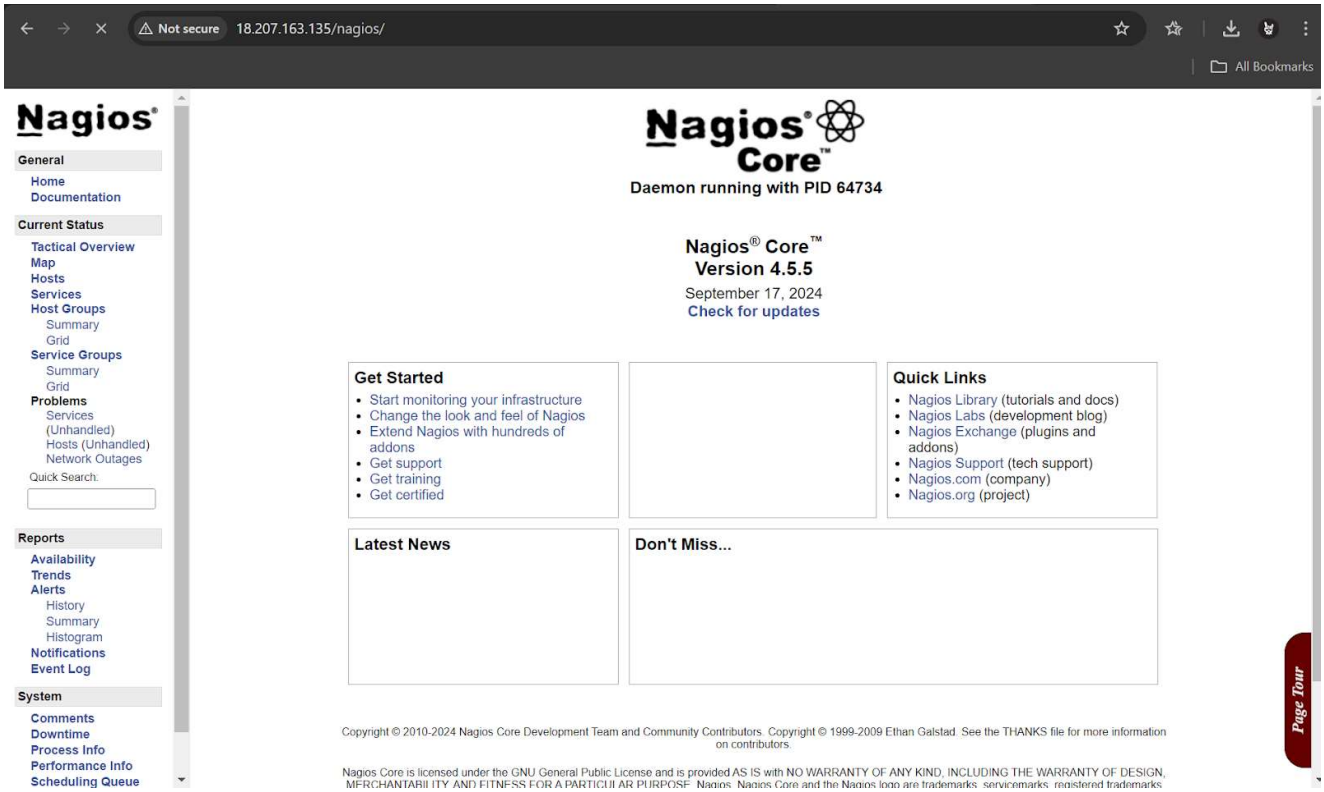
```
[ec2-user@ip-172-31-42-133 nagios-plugins-2.4.11]$ sudo service nagios start
Redirecting to /bin/systemctl start nagios.service
[ec2-user@ip-172-31-42-133 nagios-plugins-2.4.11]$
```

18. sudo systemctl status nagios
    This checks the status of Nagios. Ensure that it is ‘active(running)’.

```
[ec2-user@ip-172-31-42-133 nagios-plugins-2.4.11]$  sudo systemctl status nagios
● nagios.service - Nagios Core 4.5.5
     Loaded: loaded (/usr/lib/systemd/system/nagios.service; enabled; preset: disabled)
     Active: active (running) since Tue 2024-10-08 04:31:34 UTC; 31s ago
       Docs: https://www.nagios.org/documentation
    Process: 64732 ExecStartPre=/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg (code=exited, status=0>
    Process: 64733 ExecStart=/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SU>
   Main PID: 64734 (nagios)
      Tasks: 6 (limit: 1112)
     Memory: 5.6M
        CPU: 79ms
     CGroup: /system.slice/nagios.service
             ├─64734 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─64735 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─64736 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─64737 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─64738 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             └─64739 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg

Oct 08 04:31:34 ip-172-31-42-133.ec2.internal nagios[64734]: qh: Socket '/usr/local/nagios/var/rw/nagios.qh' successful>
Oct 08 04:31:34 ip-172-31-42-133.ec2.internal nagios[64734]: qh: core query handler registered
Oct 08 04:31:34 ip-172-31-42-133.ec2.internal nagios[64734]: qh: echo service query handler registered
Oct 08 04:31:34 ip-172-31-42-133.ec2.internal nagios[64734]: qh: help for the query handler registered
Oct 08 04:31:34 ip-172-31-42-133.ec2.internal nagios[64734]: wproc: Successfully registered manager as @wproc with quer>
Oct 08 04:31:34 ip-172-31-42-133.ec2.internal nagios[64734]: wproc: Registry request: name=Core Worker 64738;pid=64738
Oct 08 04:31:34 ip-172-31-42-133.ec2.internal nagios[64734]: wproc: Registry request: name=Core Worker 64737;pid=64737
Oct 08 04:31:34 ip-172-31-42-133.ec2.internal nagios[64734]: wproc: Registry request: name=Core Worker 64736;pid=64736
Oct 08 04:31:34 ip-172-31-42-133.ec2.internal nagios[64734]: wproc: Registry request: name=Core Worker 64735;pid=64735
Oct 08 04:31:34 ip-172-31-42-133.ec2.internal nagios[64734]: Successfully launched command file worker with pid 64739
lines 1-28/28 (END)
```

19. Again go to your ec2 instance, click on the id of your instance. On the opened page, copy the public IPV4 address of the instance.

20. In google or any other browser, enter http://<publicipaddress>/nagios. You will get a prompt box asking for username and passwords. Enter the username as nagiosadmin and password is the one we set earlier in step 12. If the password is correct you will be directed to the Nagios home page.



Conclusion:

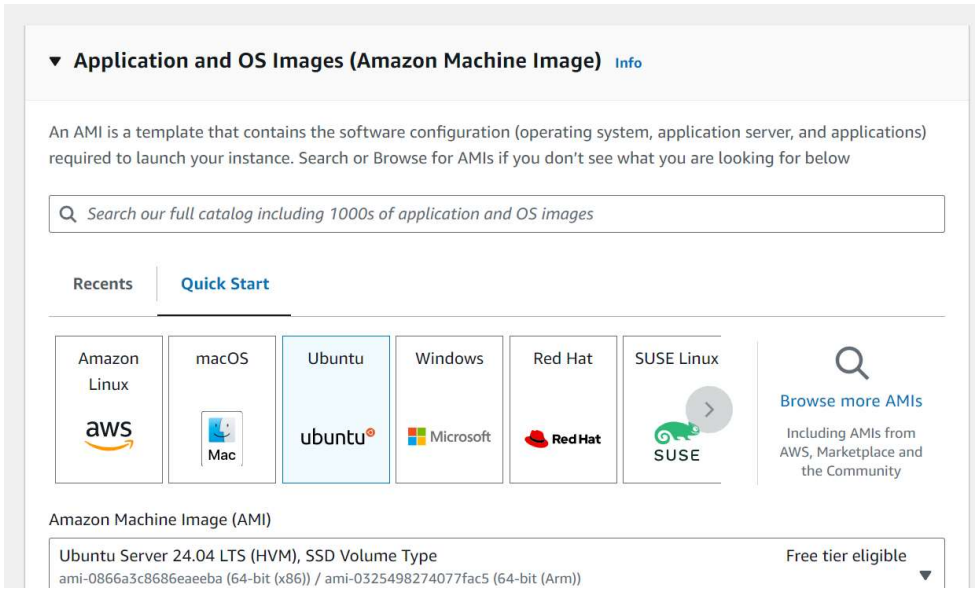
In this experiment, we successfully installed and configured Nagios Core and its essential components on an Amazon Linux EC2 instance. This setup involved creating a user for Nagios, configuring user groups, downloading and extracting Nagios and its plugins, and addressing SSL header errors by installing necessary dependencies. The Nagios service was successfully compiled and started, allowing us to access the Nagios web interface via a browser by using the instance's public IP address. With Nagios running, we now have a fully functional monitoring system in place for real-time monitoring and notifications.

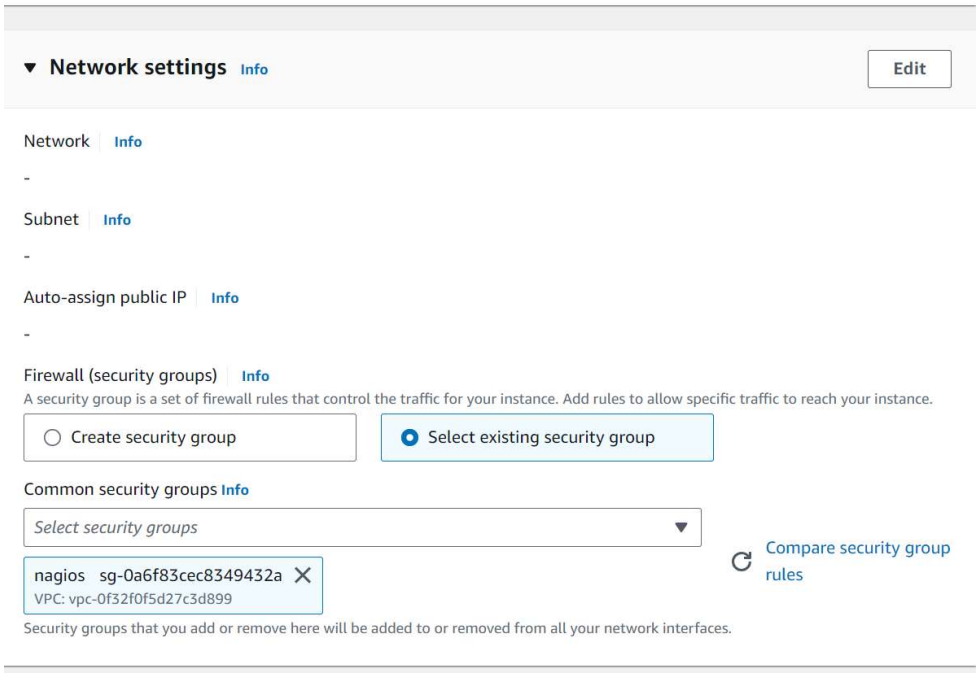
Experiment 10

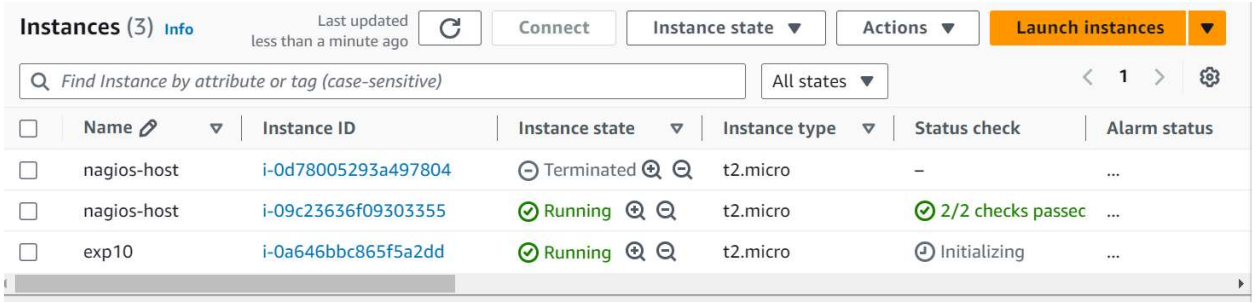
Aim: To perform Port, Service monitoring, Windows/Linux server monitoring using Nagios.

Prerequisites: An EC2 instance(nagios-host) with a nagios server already setup. (We can use the instance created in the previous experiment).

1. Go to EC2 on your AWS academy lab. Click on Launch instance and. Give an appropriate name and select Ubuntu as the instance type. Use the same key pair and the security group which was used in previous experiment. Confirm the configurations and click on create instance.







You should have both the host instance and the newly created instance.

2. Now click on the instance id for the newly created instance, click on connect. Go to the ssh tab and copy the example command. Open the folder where .pem file for key pair was installed in your terminal and run the copied command. This will connect your terminal to the ec2 instance. Do this for the host instance as well.

3. To verify whether the nagios service is running or not, run the following command
   ps -ef | grep nagios
   Perform the following commands in the host instance until specified to do otherwise.

```
[ec2-user@ip-172-31-42-133 ~]$  ps -ef | grep nagios
nagios     64734       1  0 04:31 ?        00:00:00 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
nagios     64735   64734  0 04:31 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagio
s.qh
nagios     64736   64734  0 04:31 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagio
s.qh
nagios     64737   64734  0 04:31 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagio
s.qh
nagios     64738   64734  0 04:31 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagio
s.qh
nagios     64739   64734  0 04:31 ?        00:00:00 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
root       64742    2398  0 04:32 pts/0    00:00:00 sudo systemctl status nagios
root       64744   64742  0 04:32 pts/1    00:00:00 sudo systemctl status nagios
root       64745   64744  0 04:32 pts/1    00:00:00 systemctl status nagios
ec2-user   65944   65905  0 04:51 pts/2    00:00:00 grep --color=auto nagios
```

4. sudo su
   mkdir -p /usr/local/nagios/etc/objects/monitorhosts
   mkdir -p /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
   This makes you the root user and creates two folders with the above paths.

```
[ec2-user@ip-172-31-42-133 ~]$ sudo su
mkdir -p /usr/local/nagios/etc/objects/monitorhosts
mkdir -p /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
[root@ip-172-31-42-133 ec2-user]#
```

5. Open the config file using the nano editor as we need to make some changes in the configuration.
   nano /usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg
   Change hostname and alias from ‘hostname’ to ‘linuxserver’.
   Change address to the public ip address of the ubuntu-client instance.
   Change hostgroup_name to ‘linux-servers1’.
   Change all the subsequent occurrences of hostname in the file from ‘localhost’ to linuxserver’.

```
  GNU nano 5.8            /usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg              Modified
##############################################################################



##############################################################################
#
# 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               linuxserver
    alias                   linuxserver
    address                 127.0.0.1
}



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

6. Open the Nagios config file using the following command:
   nano /usr/local/nagios/etc/nagios.cfg
   Then, add the following line to the config file:
   cfg_dir=/usr/local/nagios/etc/objects/monitorhosts/

```
  GNU nano 5.8                                 /usr/local/nagios/etc/nagios.cfg

# Definitions for monitoring the local (Linux) host
cfg_file=/usr/local/nagios/etc/objects/localhost.cfg

# Definitions for monitoring a Windows machine
#cfg_file=/usr/local/nagios/etc/objects/windows.cfg

# Definitions for monitoring a router/switch
#cfg_file=/usr/local/nagios/etc/objects/switch.cfg

# Definitions for monitoring a network printer
#cfg_file=/usr/local/nagios/etc/objects/printer.cfg


# You can also tell Nagios to process all config files (with a .cfg
# extension) in a particular directory by using the cfg_dir
# directive as shown below:

#cfg_dir=/usr/local/nagios/etc/servers
#cfg_dir=/usr/local/nagios/etc/printers
#cfg_dir=/usr/local/nagios/etc/switches
#cfg_dir=/usr/local/nagios/etc/routers
cfg_dir=/usr/local/nagios/etc/objects/monitorhosts/
```

7. To check and verify if the configurations are correct or not run the following command:
   /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

```
ocalhost.cfg', starting on line 58)
   Read object config files okay...

Running pre-flight check on configuration data...

Checking objects...
        Checked 8 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
```

In the end you will see “Total warning” and “total error” as 0, this confirms that the configurations is correct.

8. Now we will restart the nagios server to implement the above made changes.
   service nagios restart

```
[root@ip-172-31-42-133 ec2-user]# service nagios restart
Redirecting to /bin/systemctl restart nagios.service
```

9. systemctl status nagios
   Using the above command, we check the status of the nagios server and ensure that it is active (running).

```
[root@ip-172-31-42-133 ec2-user]#  systemctl status nagios
● nagios.service - Nagios Core 4.5.5
     Loaded: loaded (/usr/lib/systemd/system/nagios.service; enabled; preset: disabled)
     Active: active (running) since Tue 2024-10-08 05:04:49 UTC; 33s ago
       Docs: https://www.nagios.org/documentation
    Process: 66879 ExecStartPre=/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg (code=exited, status=0>
    Process: 66880 ExecStart=/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SU>
   Main PID: 66881 (nagios)
      Tasks: 6 (limit: 1112)
     Memory: 4.0M
        CPU: 23ms
     CGroup: /system.slice/nagios.service
             ├─66881 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─66882 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─66883 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─66884 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─66885 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             └─66886 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg

Oct 08 05:04:49 ip-172-31-42-133.ec2.internal nagios[66881]: wproc: Registry request: name=Core Worker 66882;pid=66882
Oct 08 05:04:49 ip-172-31-42-133.ec2.internal nagios[66881]: Warning: Duplicate definition found for service 'HTTP' on >
Oct 08 05:04:49 ip-172-31-42-133.ec2.internal nagios[66881]: Warning: Duplicate definition found for service 'SSH' on h>
Oct 08 05:04:49 ip-172-31-42-133.ec2.internal nagios[66881]: Warning: Duplicate definition found for service 'Swap Usag>
Oct 08 05:04:49 ip-172-31-42-133.ec2.internal nagios[66881]: Warning: Duplicate definition found for service 'Current L>
Oct 08 05:04:49 ip-172-31-42-133.ec2.internal nagios[66881]: Warning: Duplicate definition found for service 'Total Pro>
Oct 08 05:04:49 ip-172-31-42-133.ec2.internal nagios[66881]: Warning: Duplicate definition found for service 'Current U>
Oct 08 05:04:49 ip-172-31-42-133.ec2.internal nagios[66881]: Warning: Duplicate definition found for service 'Root Part>
Oct 08 05:04:49 ip-172-31-42-133.ec2.internal nagios[66881]: Warning: Duplicate definition found for service 'PING' on >
Oct 08 05:04:49 ip-172-31-42-133.ec2.internal nagios[66881]: Successfully launched command file worker with pid 66886
```

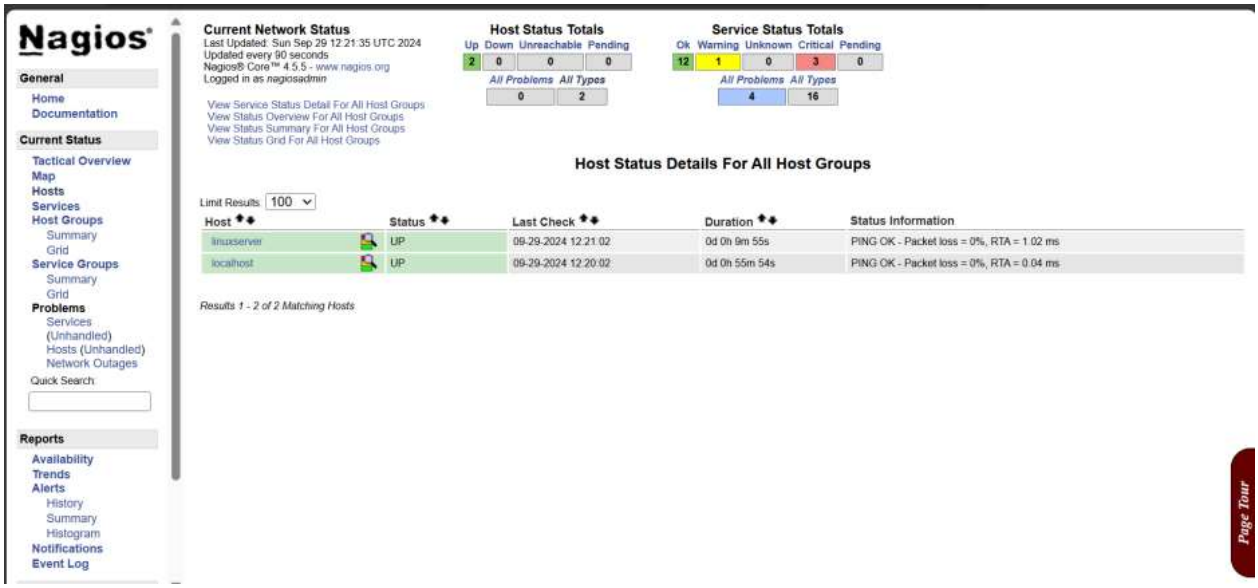
10. Now open the terminal which is connected to the ubuntu instance. If not connect, follow the 2nd step in similar fashion to connect to the instance, run the following command in ubuntu instance.
    sudo apt update -y
    sudo apt install gcc -y
    sudo apt install -y nagios-nrpe-server nagios-plugins

```
ubuntu@ip-172-31-42-172:~$ sudo apt update -y
sudo apt install gcc -y
sudo apt install -y nagios-nrpe-server nagios-plugins
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 [382 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Components [35.0 kB]
```

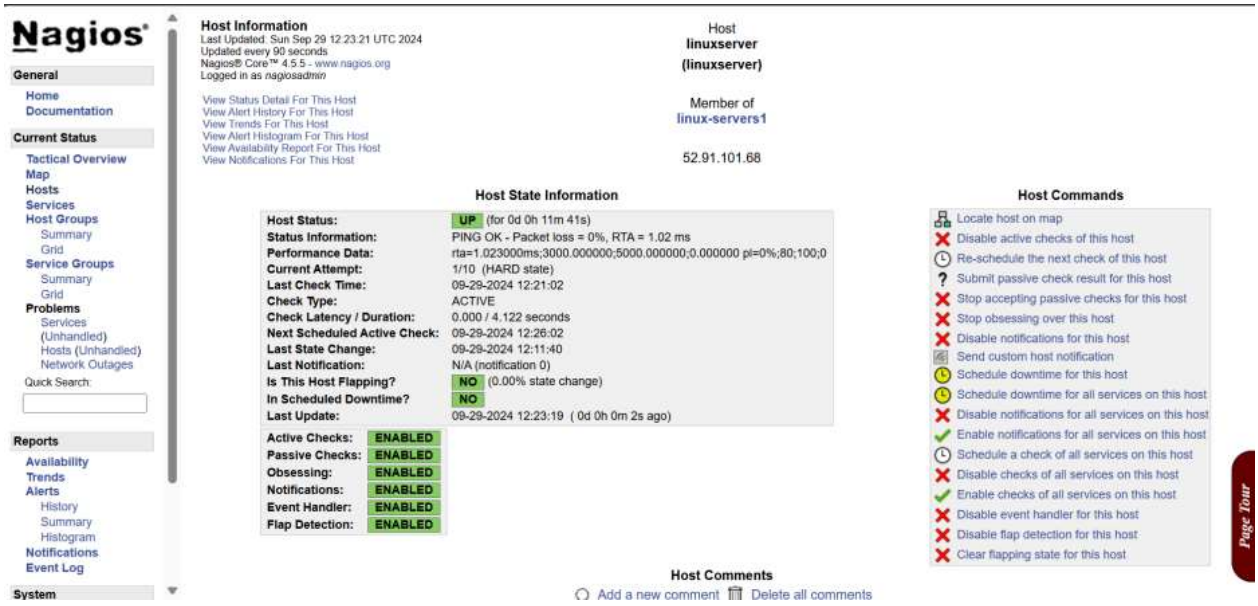
11. Run the following command:
    sudo nano /etc/nagios/nrpe.cfg
    The above command opens the NRPE config file. Here, we need to add the public IP address of our host nagios-host instance to the NRPE configuration file. Under allowed_hosts, add the nagios-host public IPv4 address. The public ip address can be seen by click on the instance id of the instance in EC2 dashboard.

```
  GNU nano 7.2                                    /etc/nagios/nrpe.cfg *
# You can either supply a group name or a GID.
#
# NOTE: This option is ignored if NRPE is running under either inetd or xinetd

nrpe_group=nagios



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

allowed_hosts=127.0.0.1,54.90.219.49



# COMMAND ARGUMENT PROCESSING
```

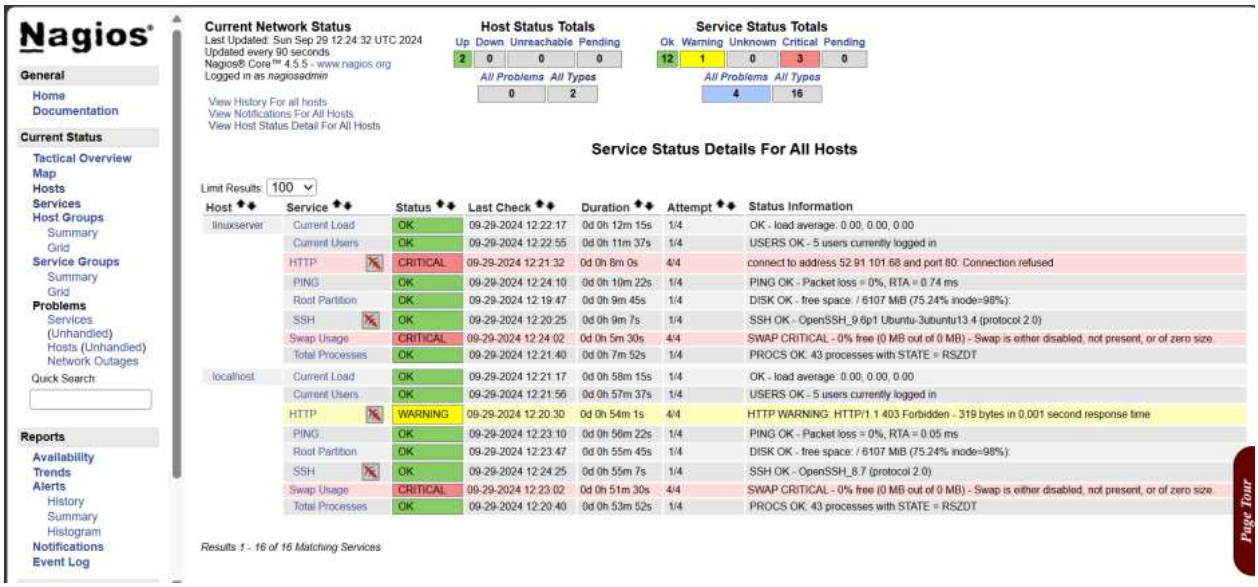
12. Once everything is completed, open the nagios dashboard in browser with url http://<publicipaddress>/nagiso. Click on the hosts and we will see that linuxserver has been added as a host



Click on ‘linuxserver’. Here, we can access all information about the ‘linuxserver’ host.



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



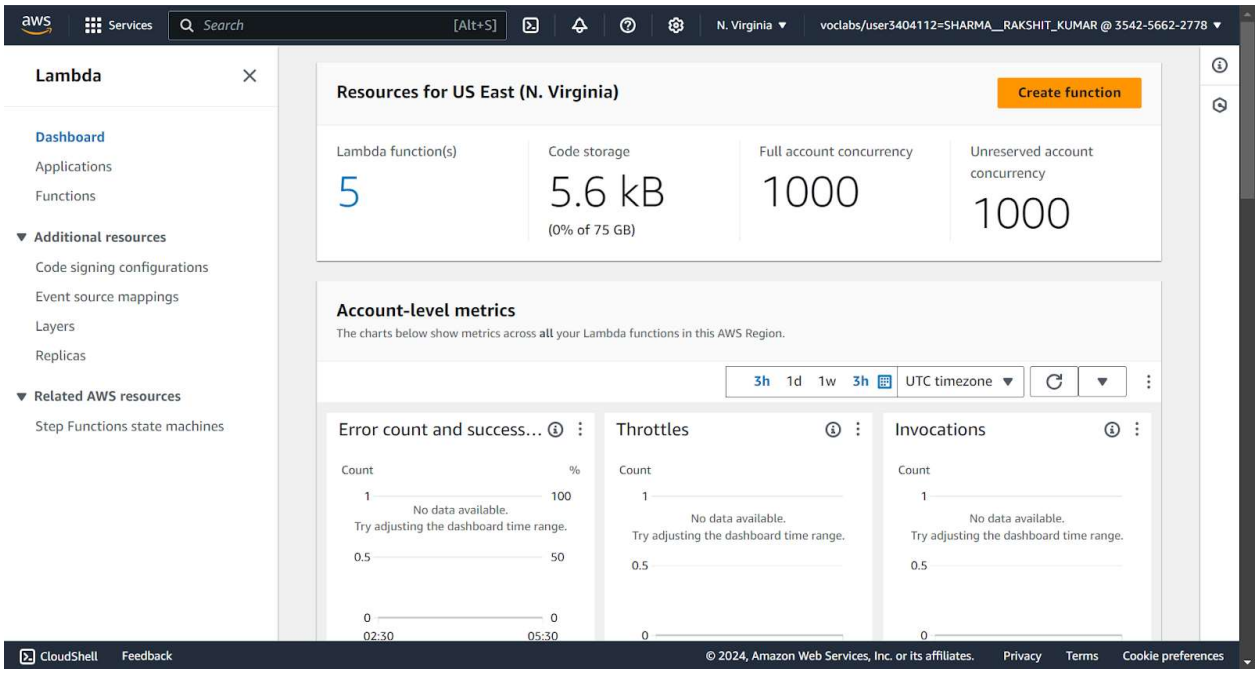
Conclusion:

In this experiment, we successfully performed port, service, and Linux server monitoring using Nagios. After setting up a new Ubuntu EC2 instance, we configured it as a monitored client by creating appropriate Nagios configuration files and modifying the host instance's settings. We installed the Nagios NRPE server and plugins on the Ubuntu instance, added the public IP of the Nagios host to the NRPE config file, and verified the changes in the Nagios dashboard.

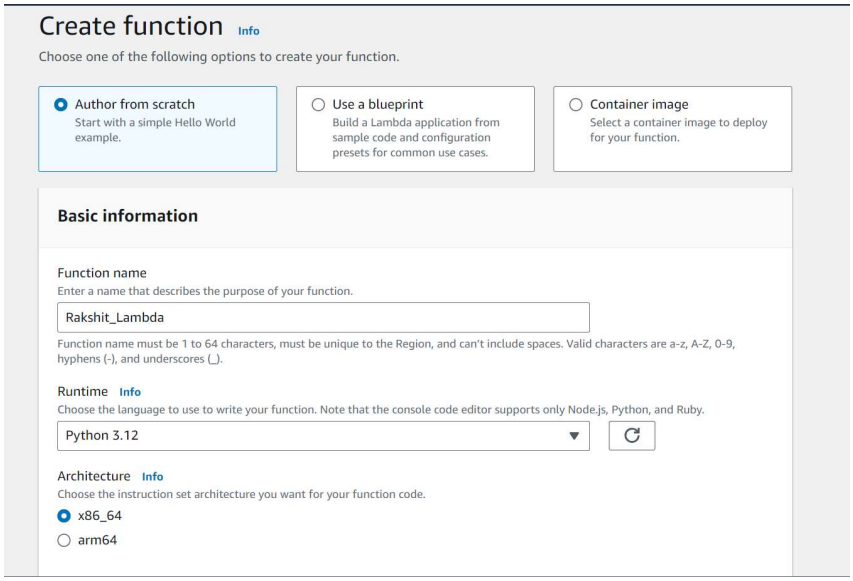
Experiment 11

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

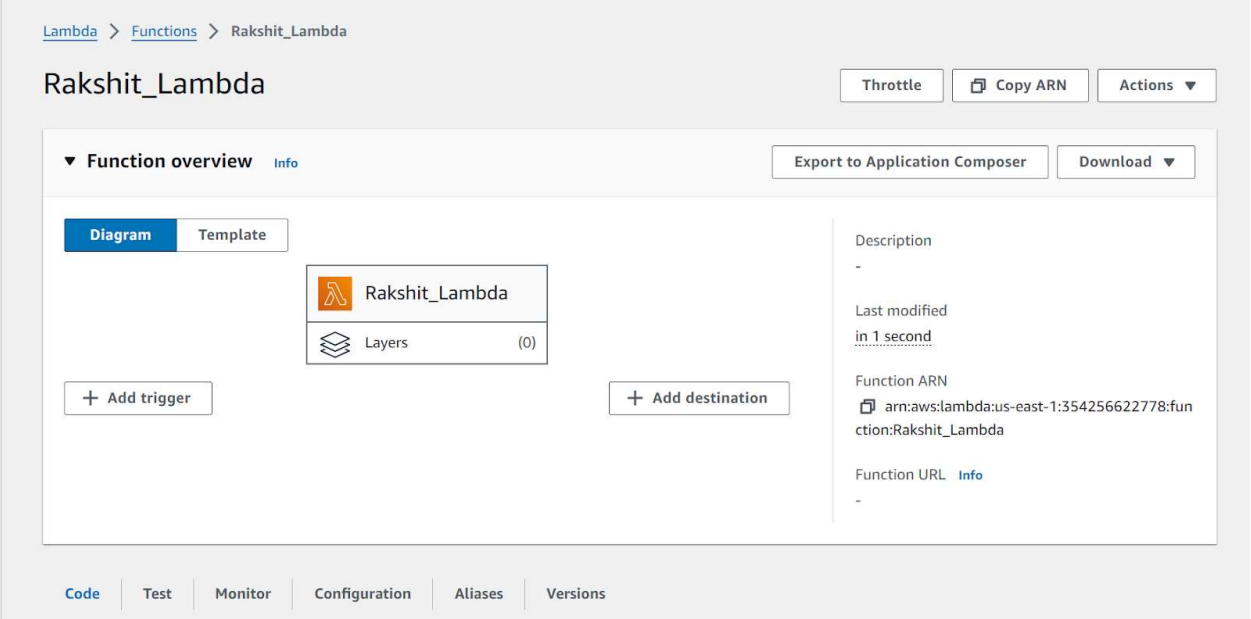
1. Log into your AWS academy account and search for Lambda service. Click on create function to create a new lambda function.

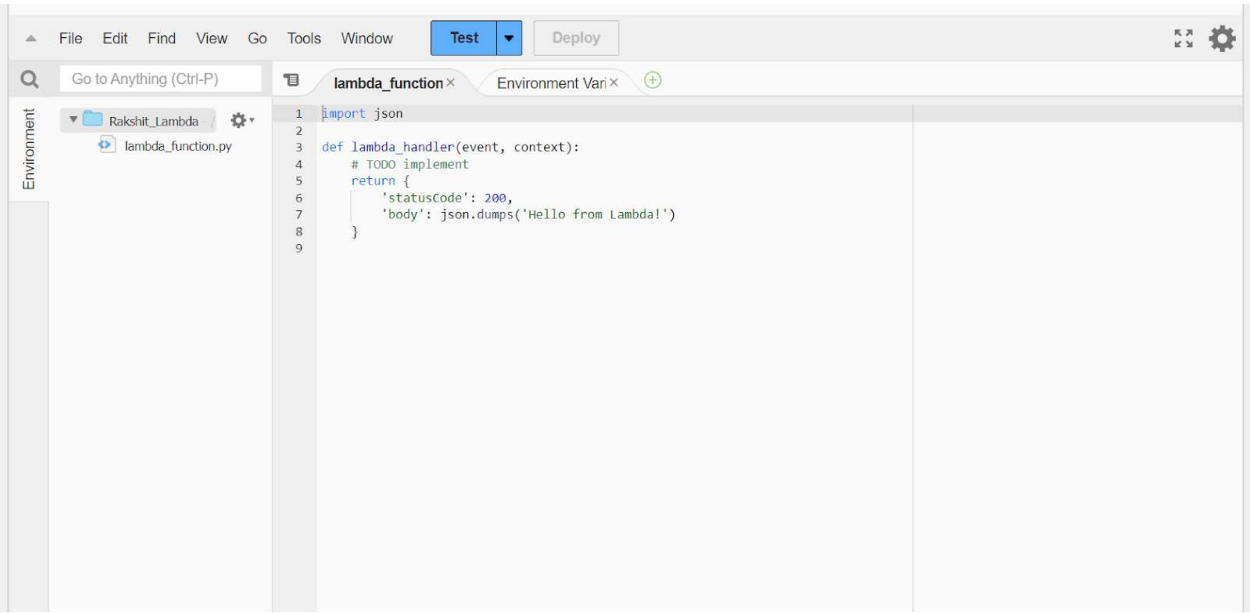


2. Give a name to your lambda function. Select the language you want to use to write the functions. We will use Python 3.12, Architecture x86. Select Exceution role to Create a new role with basic Lambda permissions.

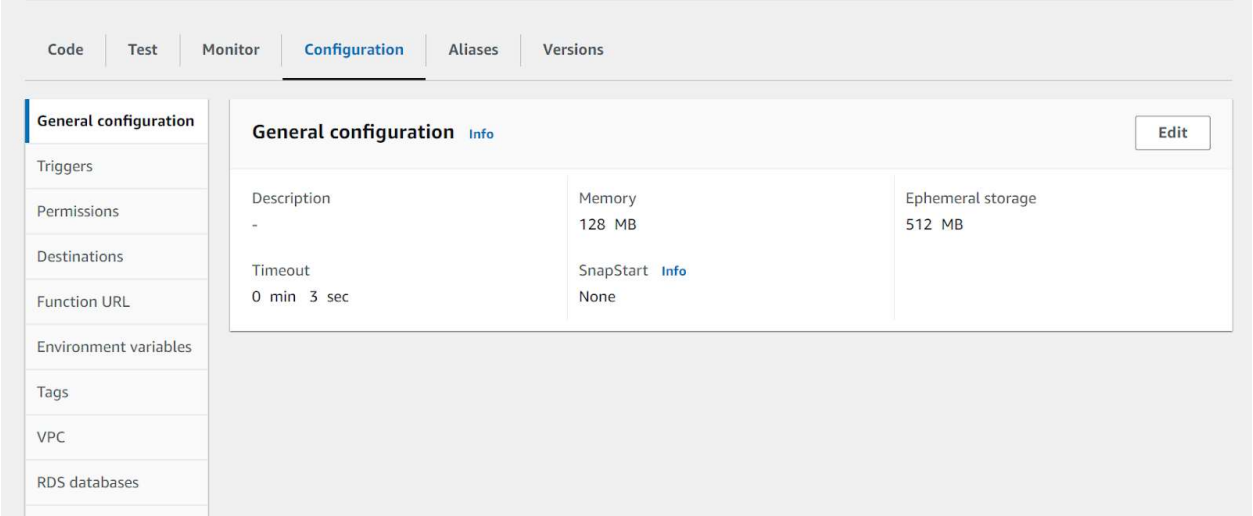


You will be able to see the created function in Functions tab



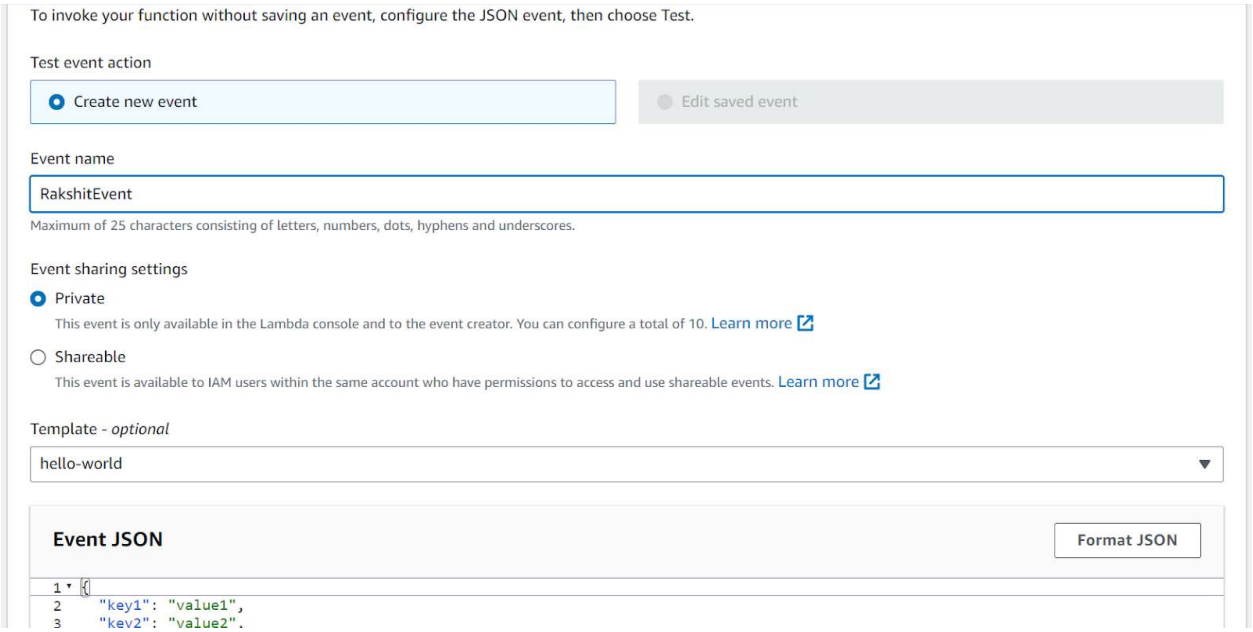


3. Scroll down and go to the configuration tab. In General configuration click on edit to change the configuration.

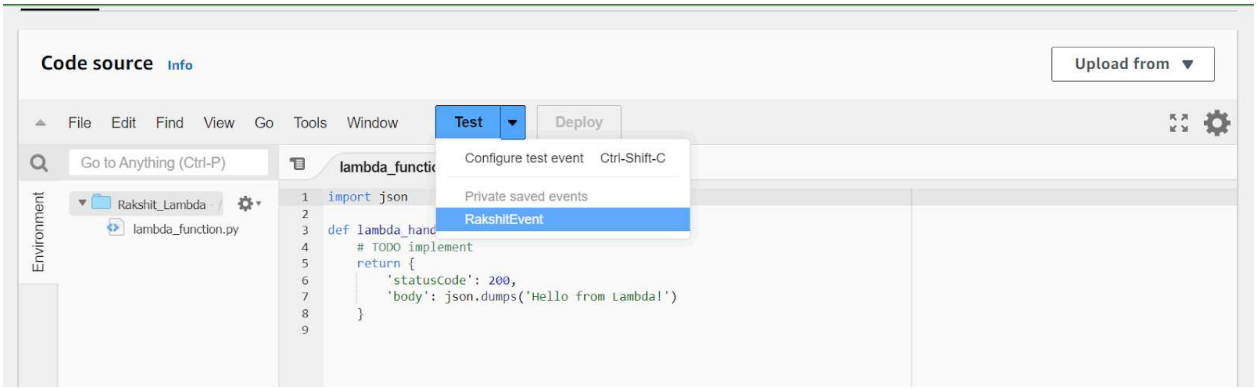


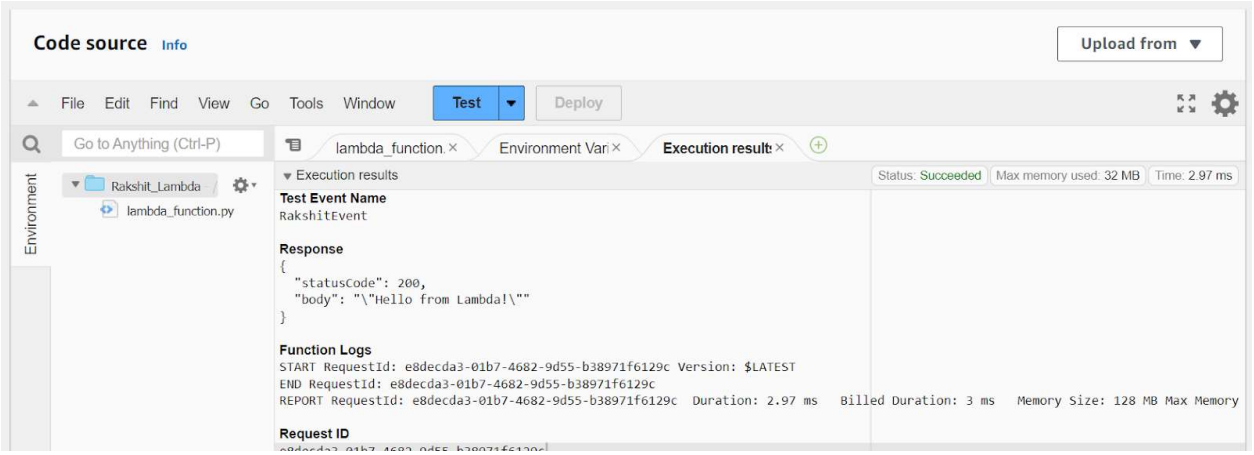
Here, you can enter a description and change Memory and Timeout. I've changed the Timeout period to 1 sec since that is sufficient for now

4. Now go to the test tab and select Create new event. Give the event an appropriate name, keep the sharing settings as private and template as hello world.

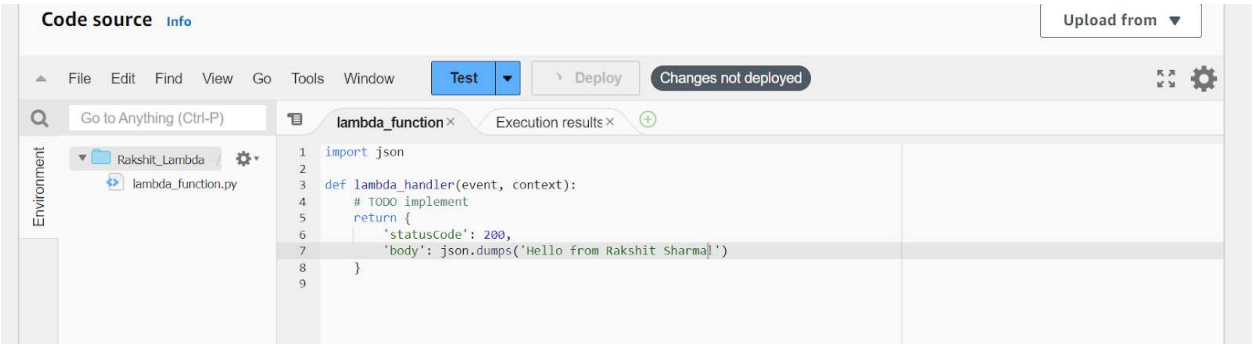


5. Now In Code section select the created event from the dropdown of test then click on test . Observe the output.

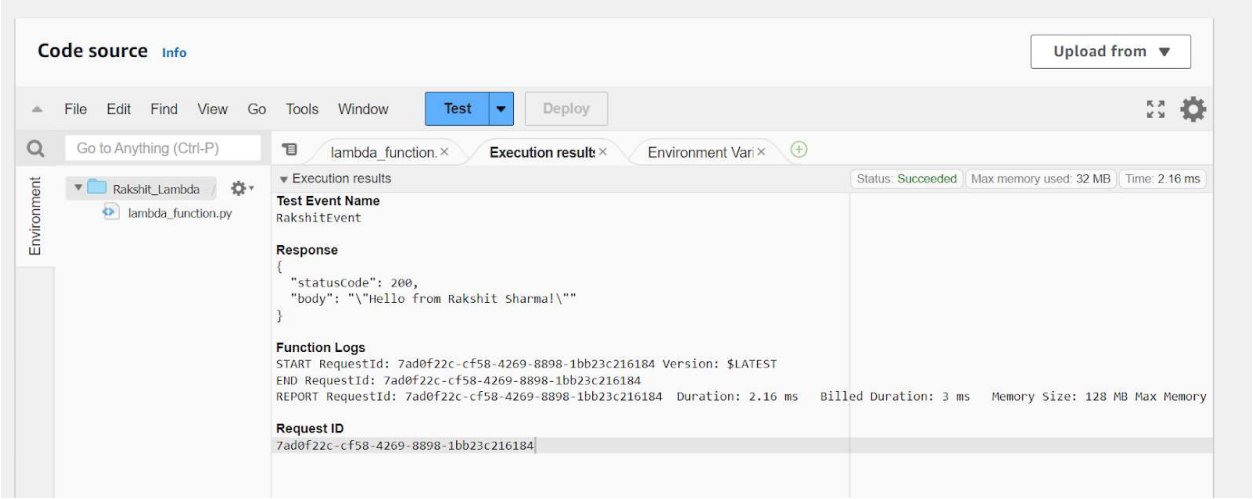




6. We can also edit the lambda function code. I have added my name in the print statement.



To save the changes click on deploy. And again select the event and click on Test. Check the output displayed.



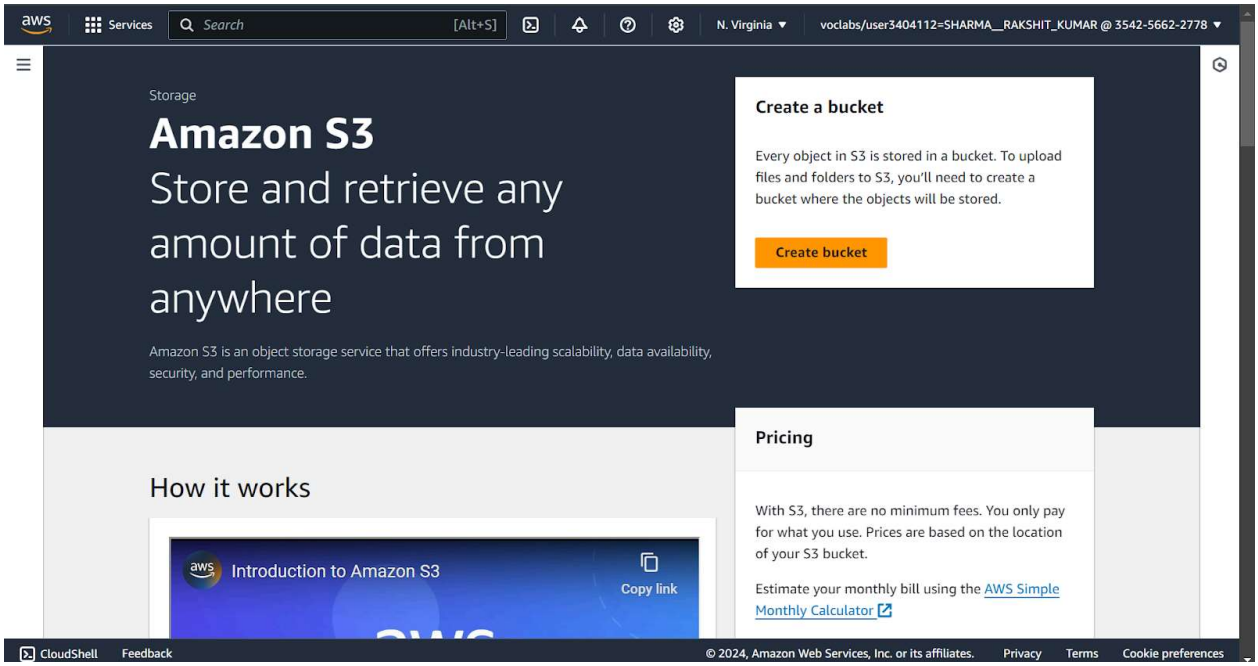
Conclusion:

In this experiment, we explored AWS Lambda, created a basic Lambda function, and tested it using Python 3.12. We walked through creating a Lambda function, configuring its settings, and testing it with a predefined event. We also demonstrated how to modify the function's code, deployed the changes, and observed the output. This experiment gave us insight into AWS Lambda’s workflow, showing how serverless functions can be easily created, tested, and deployed, making it a powerful tool for event-driven applications.

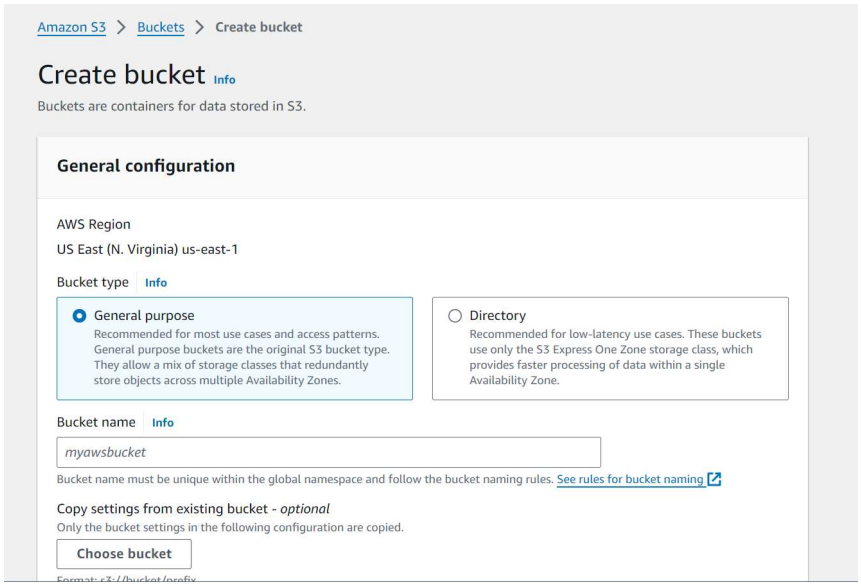
Experiment 12

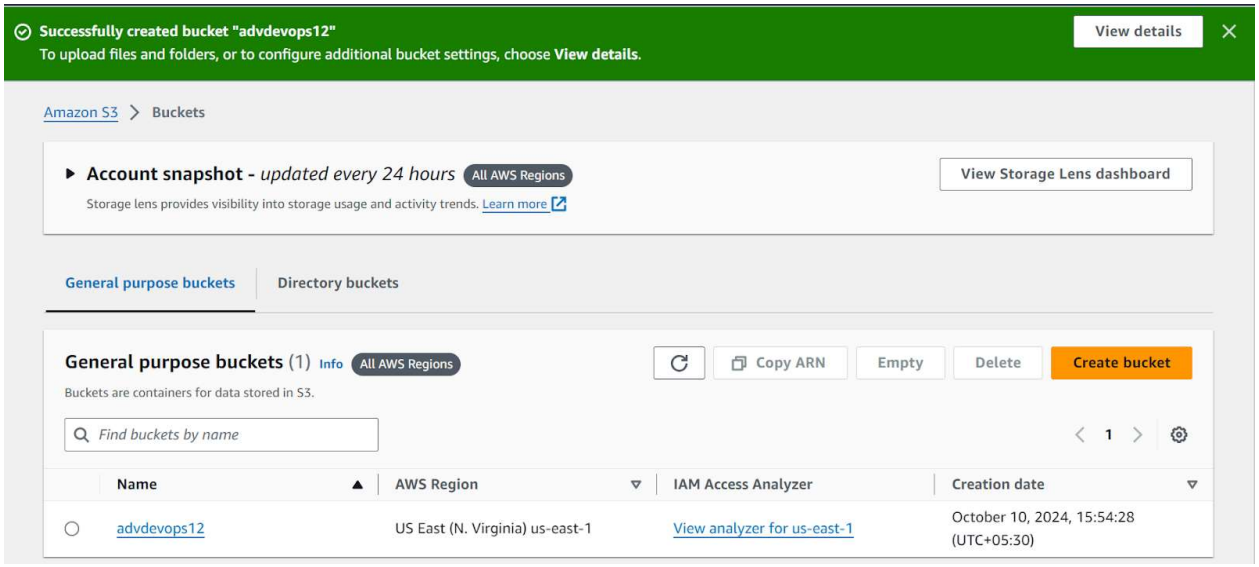
Aim: To create a Lambda function which will log “An Image has been added” once you add an object to a specific bucket in S3

1. Login to your AWS account. Search for S3 service, click on create bucket to make a new bucket.

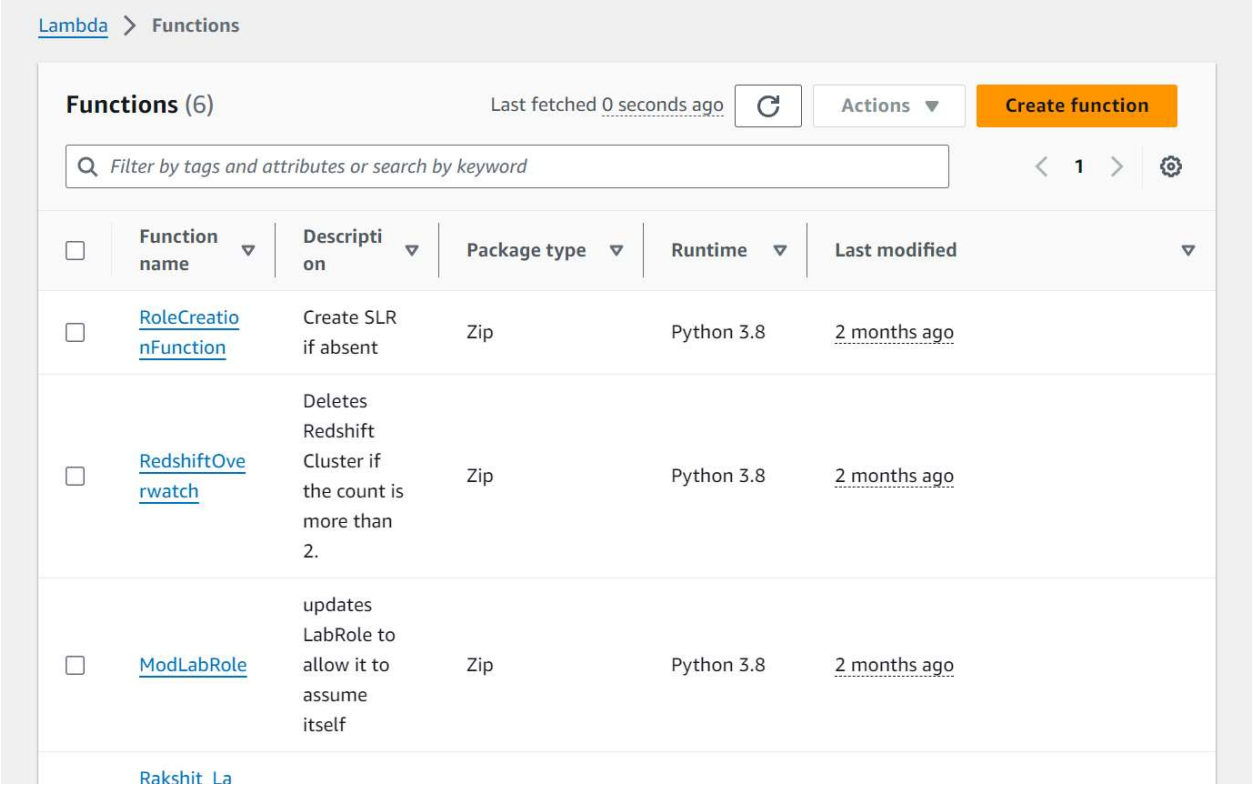


2. Select the bucket type as general purpose, give your bucket an appropriate name. Uncheck the block all public access. Keep rest of the settings as default and click on create bucket.

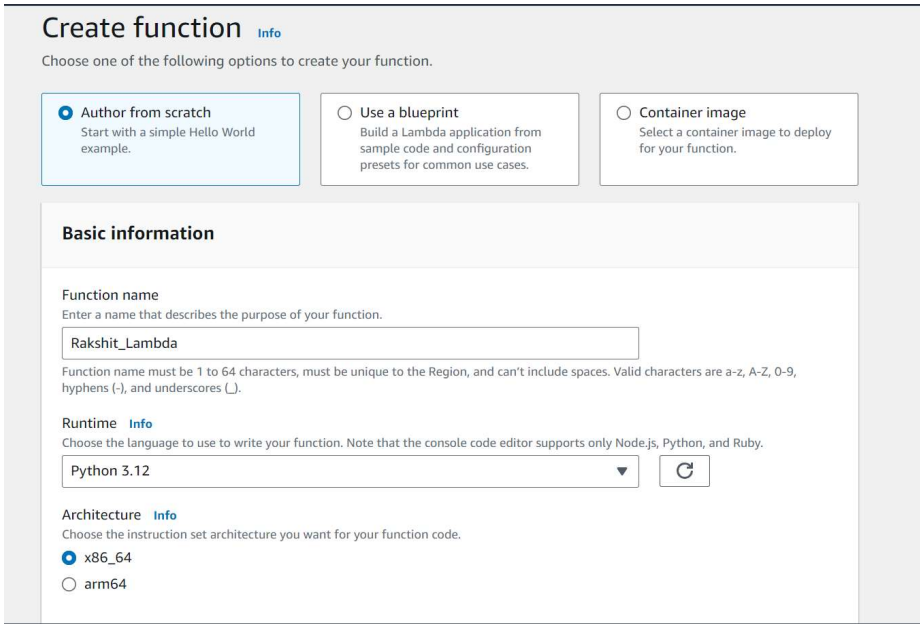


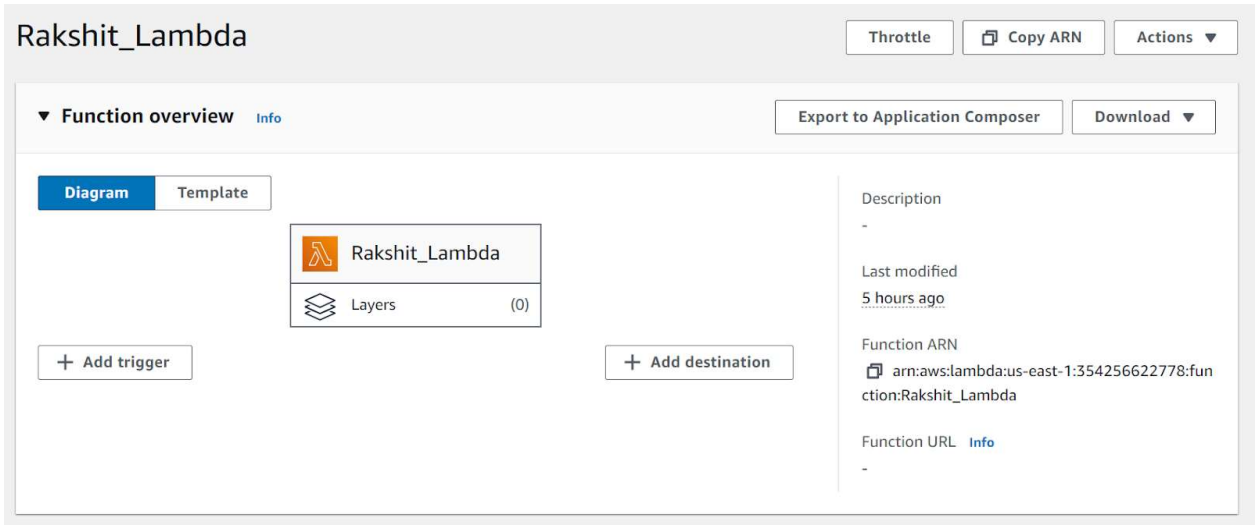


3. Search for lambda and open its console. Click on create function to make a new function.

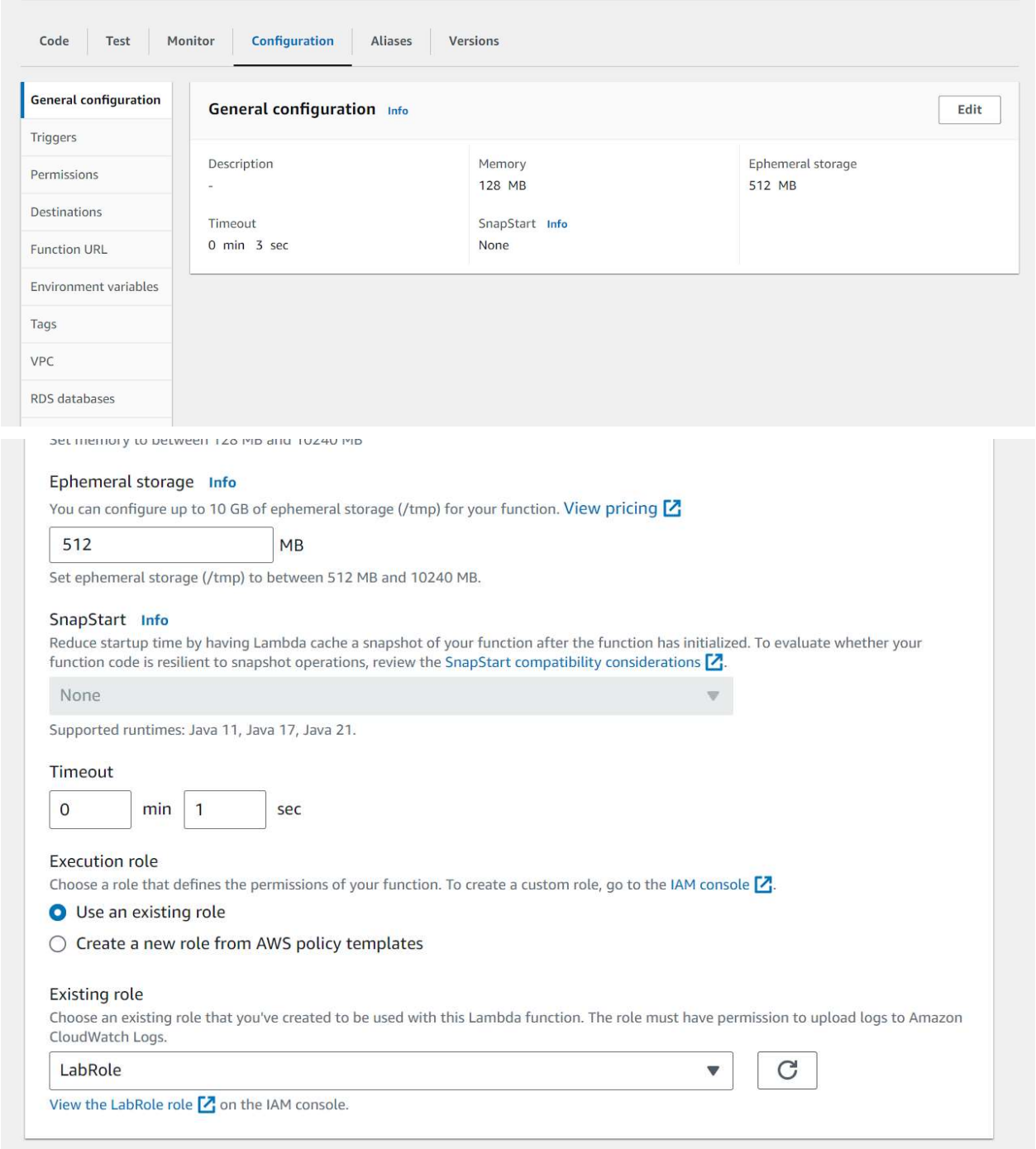


4. Give a name to your lambda function. Select the language you want to use to write the functions. We will use Python 3.12, Architecture x86. Select Execution role to Create a new role with basic Lambda permissions.



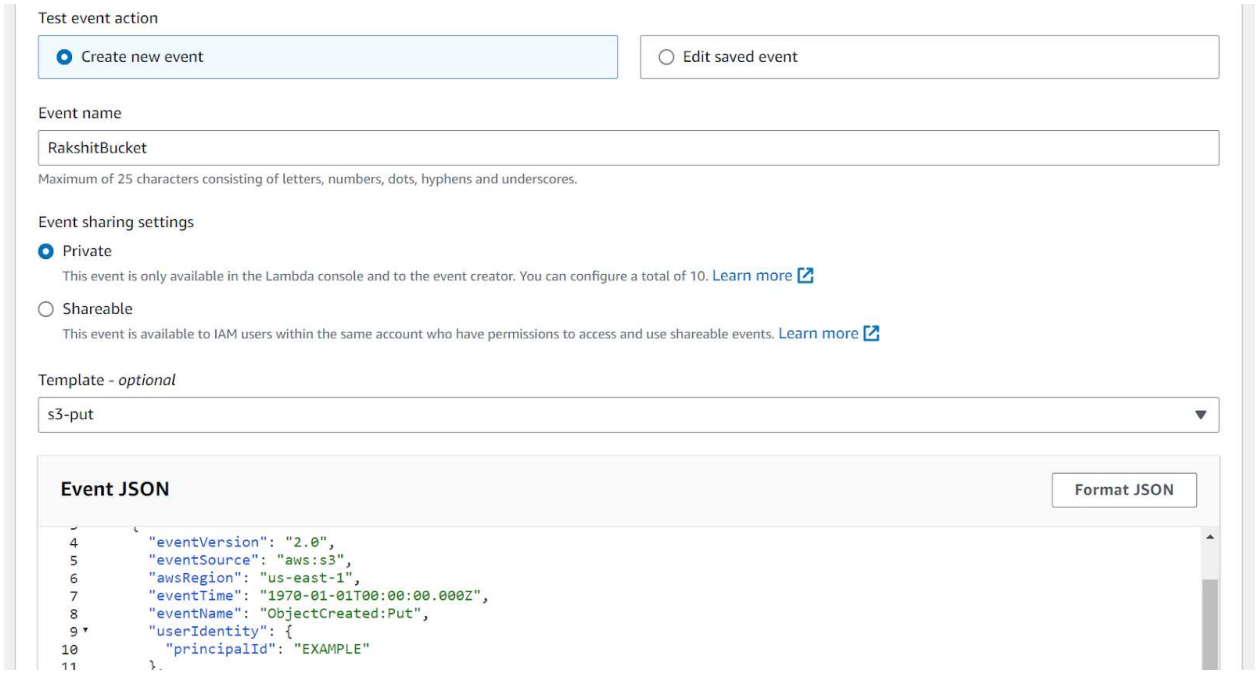


5. Scroll down and go to the configuration tab. In General configuration click on edit to change the configuration.

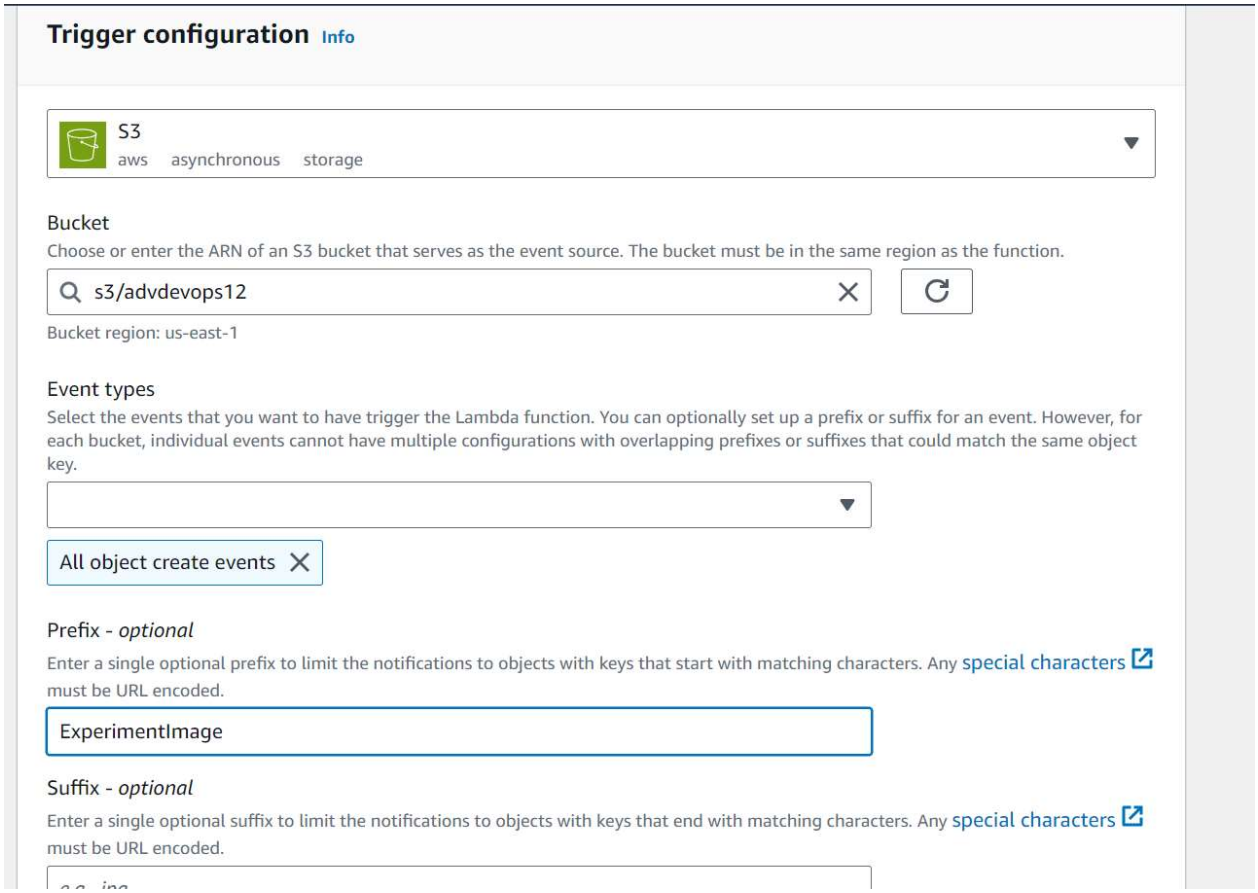


Here, you can enter a description and change Memory and Timeout. I've changed the Timeout period to 1 sec since that is sufficient for now

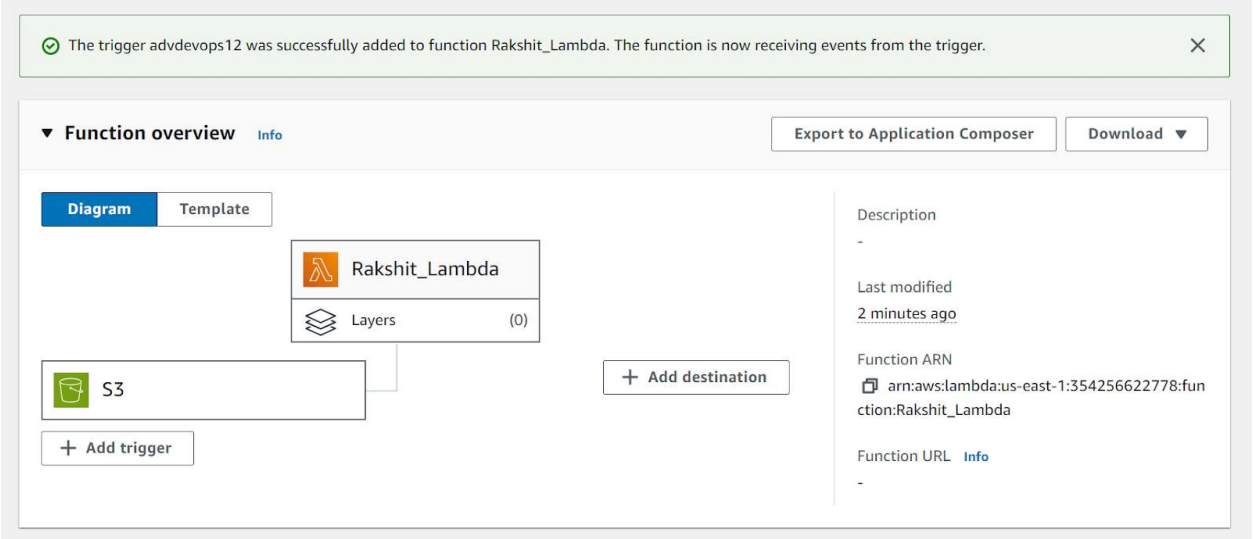
6. Now to create a new event, go to the test tab. Create a new event, give a name to the event and select Event Sharing to private, and select s3 put template.



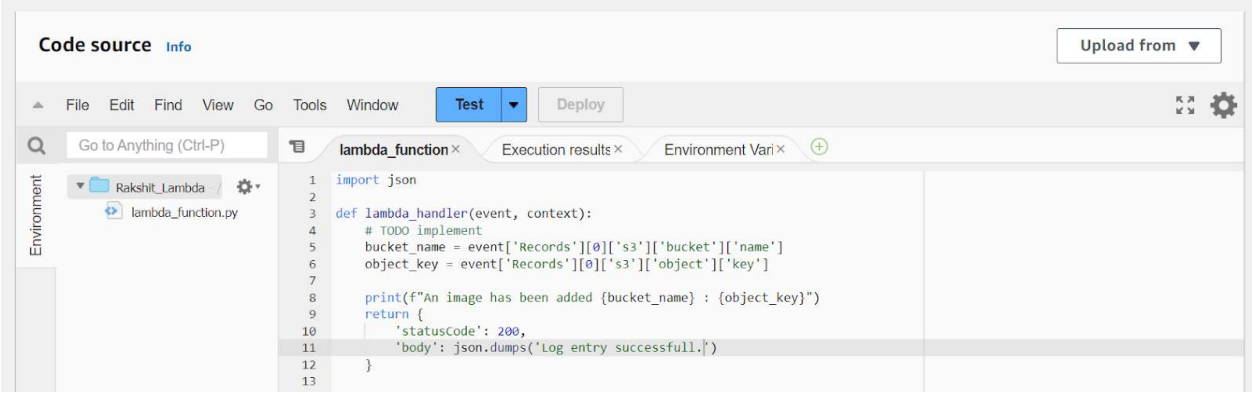
7. Now in the lambda function click on add trigger.



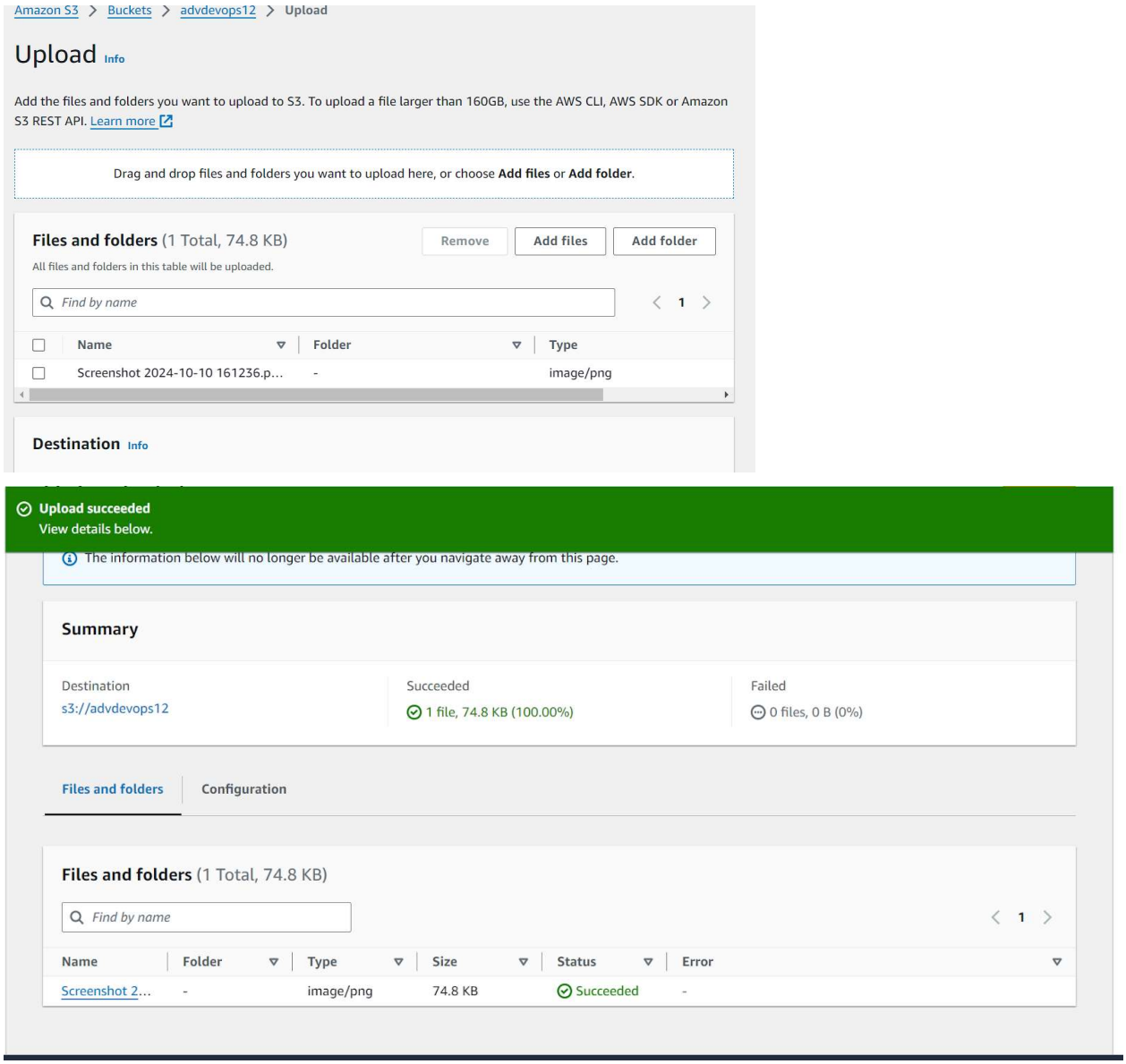
Select the source as s3, and search for the bucket that we created earlier. If you want you can add the prefix for the image. Keep the rest of the setting as default and add trigger.



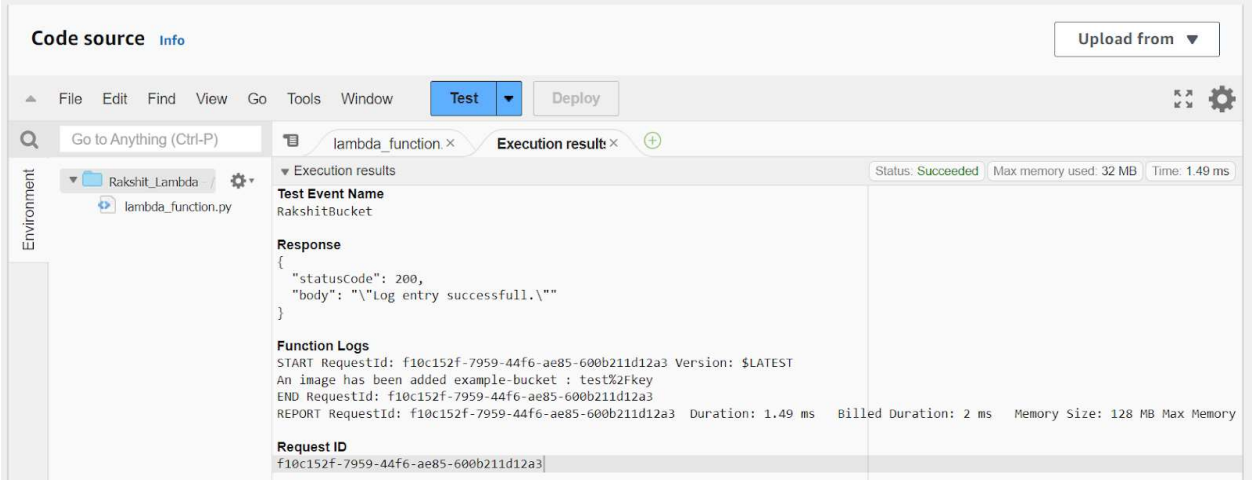
8. Now Write code that logs a message like “An Image has been added” when triggered. Save the file and click on deploy.



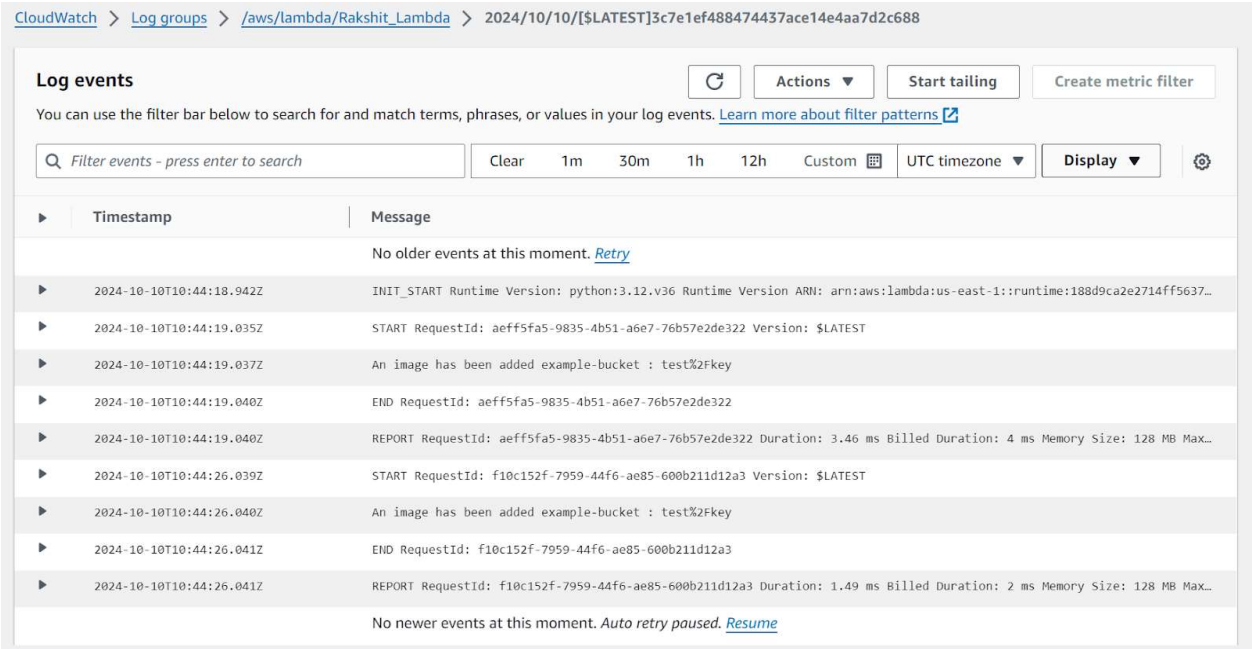
9. Now we will upload an image to our bucket.



10. Now in lambda function, select the event which we created now and click on test. Check if it is logging the string that we added in our code.



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 a Lambda function that logs "An Image has been added" when an object is uploaded to a specific S3 bucket. We started by creating an S3 bucket and setting up a Lambda function with Python 3.12. We configured a trigger for the Lambda function to monitor the S3 bucket and added code to log the message when an image is uploaded. After deploying the code and testing it by uploading an image, we verified the logs in CloudWatch. This experiment demonstrated the use of AWS Lambda for event-driven automation, specifically integrating Lambda with S3.