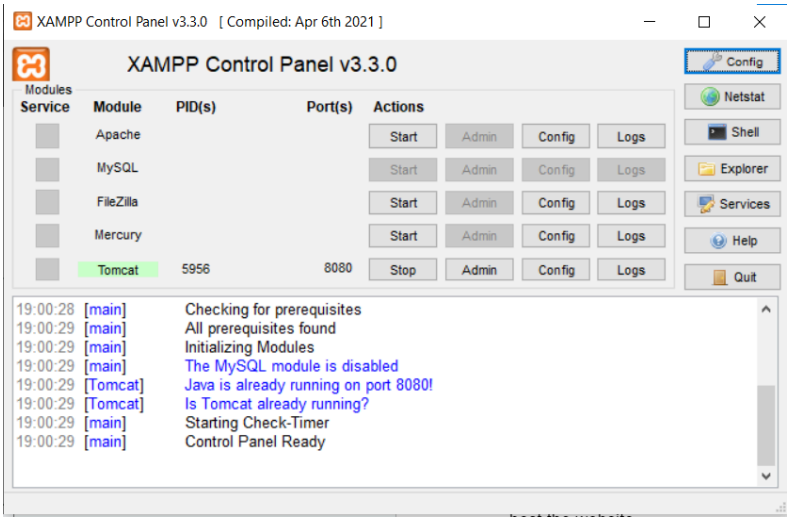
Name: Veydant Sharma Class: D15C Roll no.: 50

Experiment 1A

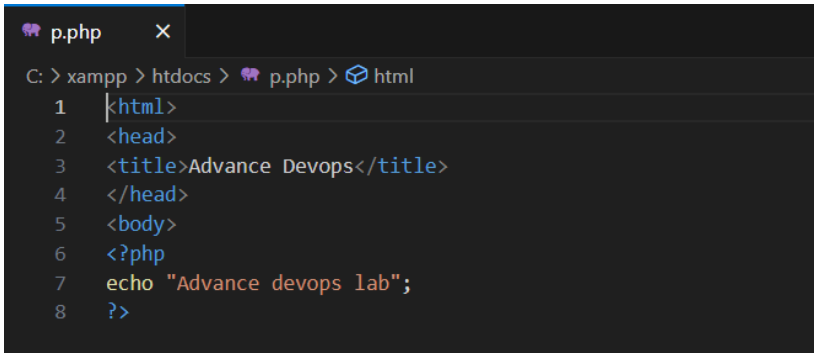
Part 1: To develop a website and host it on Xampp:

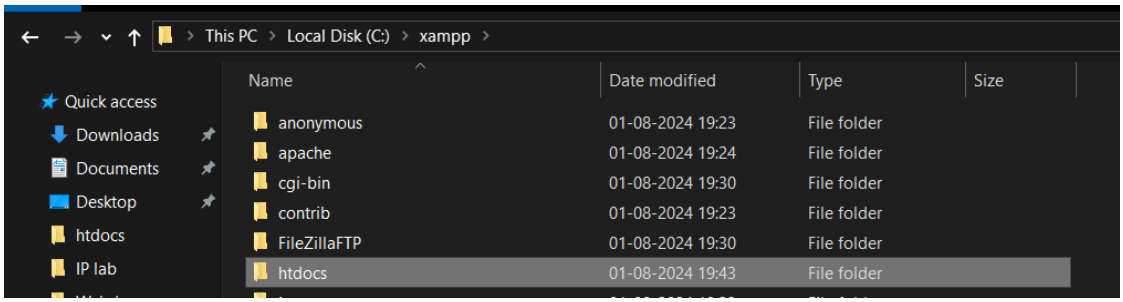
Step 1: Go to the official website of Xampp. https://www.apachefriends.org/download.html. Select the suitable version and complete the installation.

Step 2: After installation open the Xampp control panel, we need to start the Apache server to host the website.



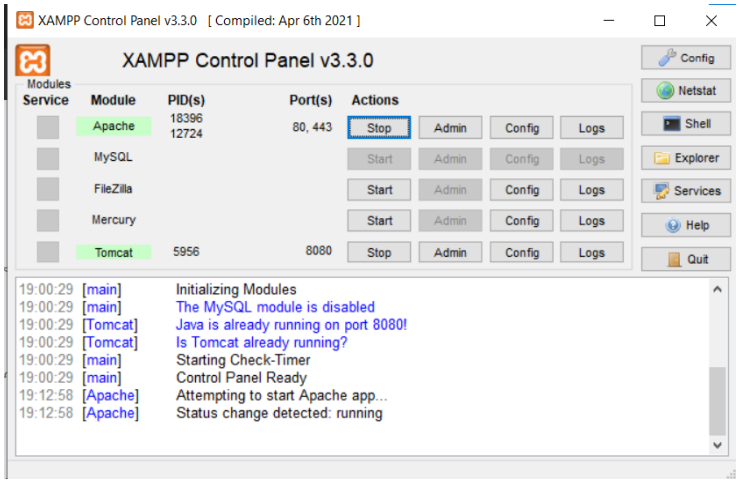
Step 3: Setup a php project, create a php project and save it as .php file. The file should be saved in ‘htdocs’ folder present in the ‘Xampp’ folder created after installation.

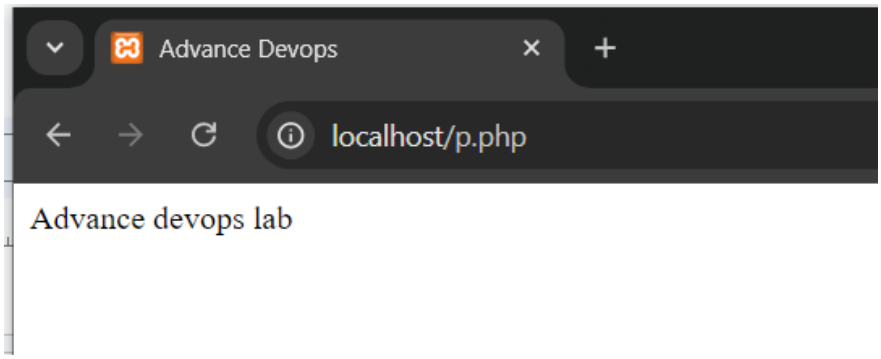




Name: Veydant Sharma Class: D15C Roll no.: 50

Step 4: Start the Xampp server and go to localhost in browser. Edit the URL such that it is localhost/filename.php or localhost/folder_name .





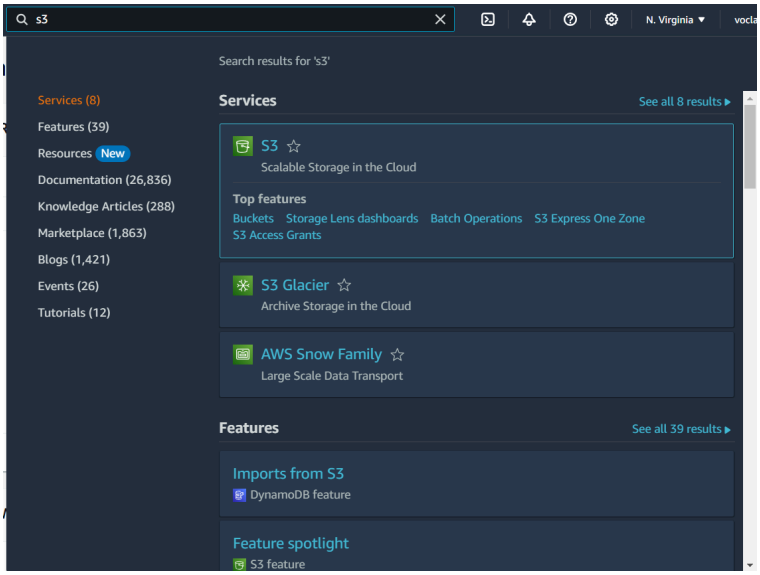
Our website has now been hosted using xampp

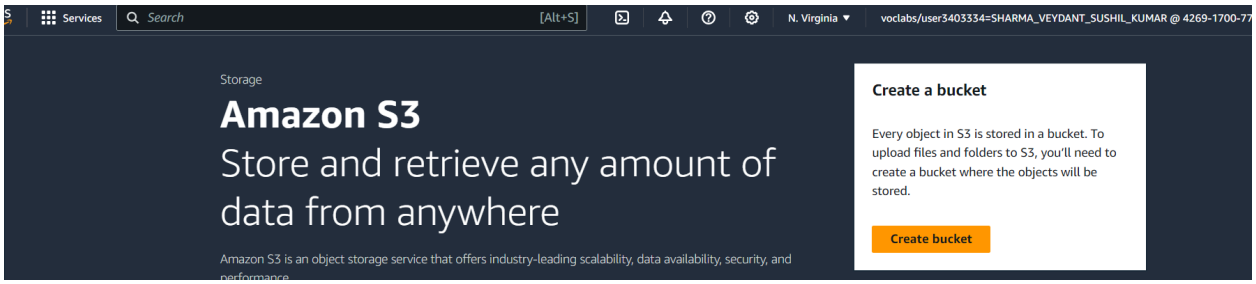
Name: Veydant Sharma Class: D15C Roll no.: 50

Part 2: Hosting a static website on AWS S3.

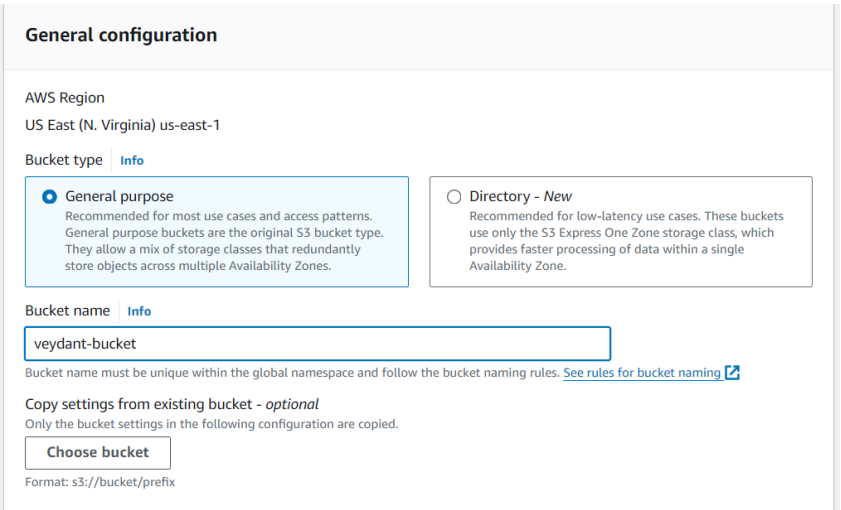
Step1: Login to AWS Academy and launch learner's lab.

Step 2: Search for the S3 service and Create a bucket.

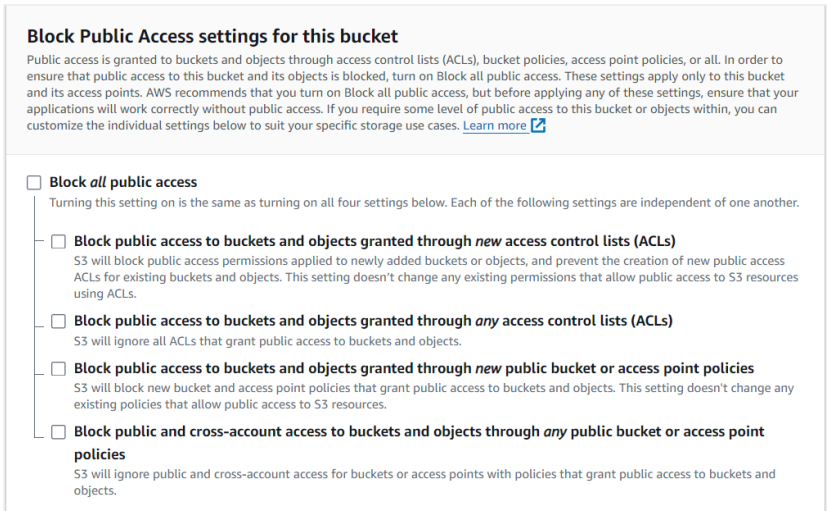




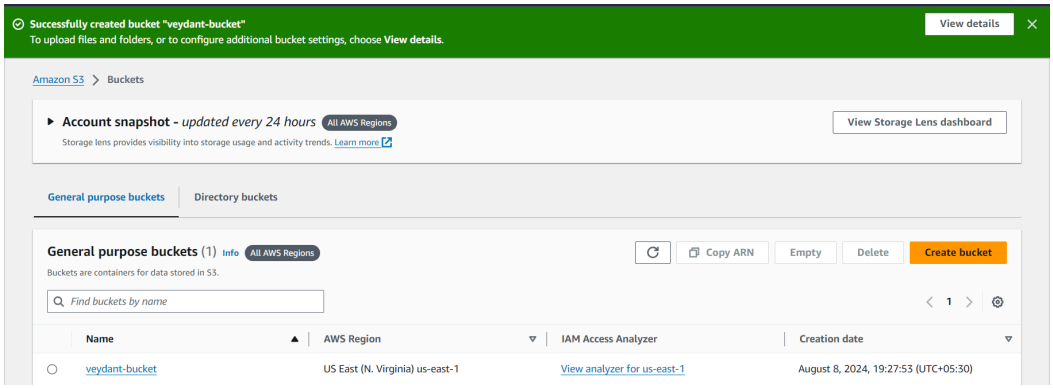
Step 3: Fill in the details and name your bucket. Use default settings. Uncheck the "Block all public access" box to prevent error.



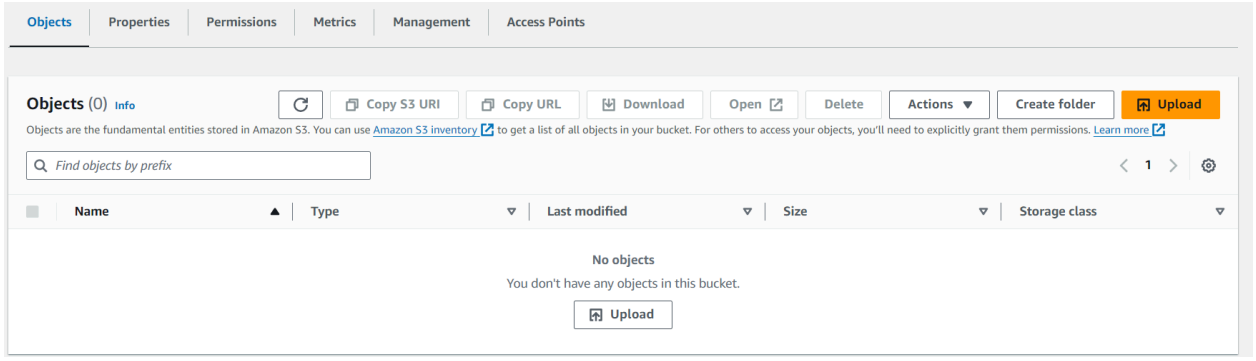
Name: Veydant Sharma Class: D15C Roll no.: 50

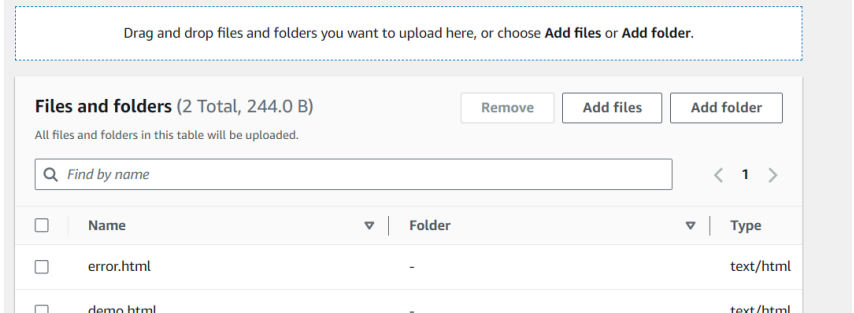


Once completed, Click ‘Create Bucket’.



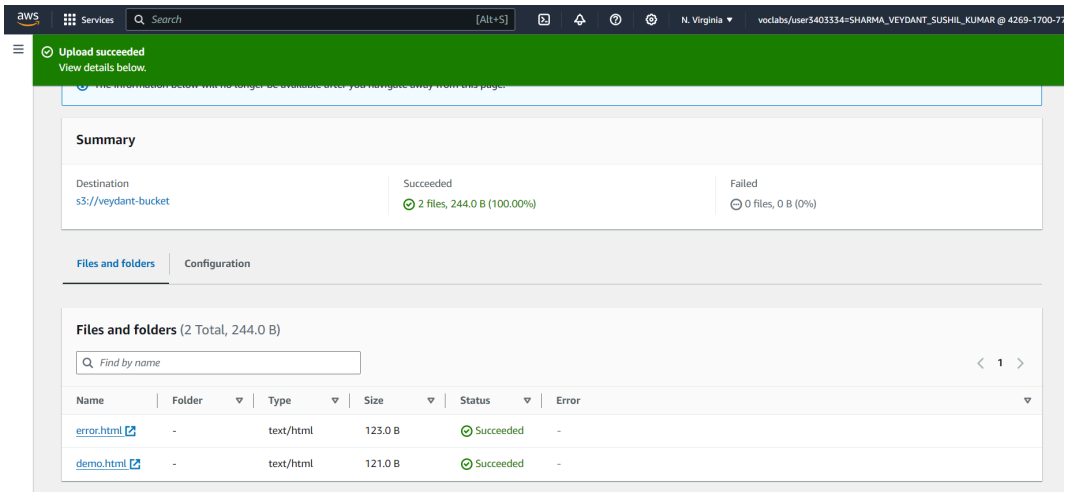
Step 4: Open the bucket and click on upload in the objects tab.



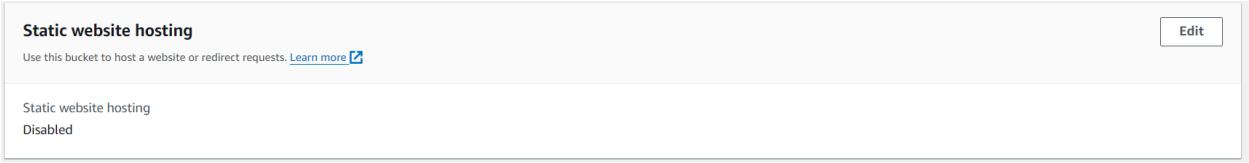


Name: Veydant Sharma Class: D15C Roll no.: 50

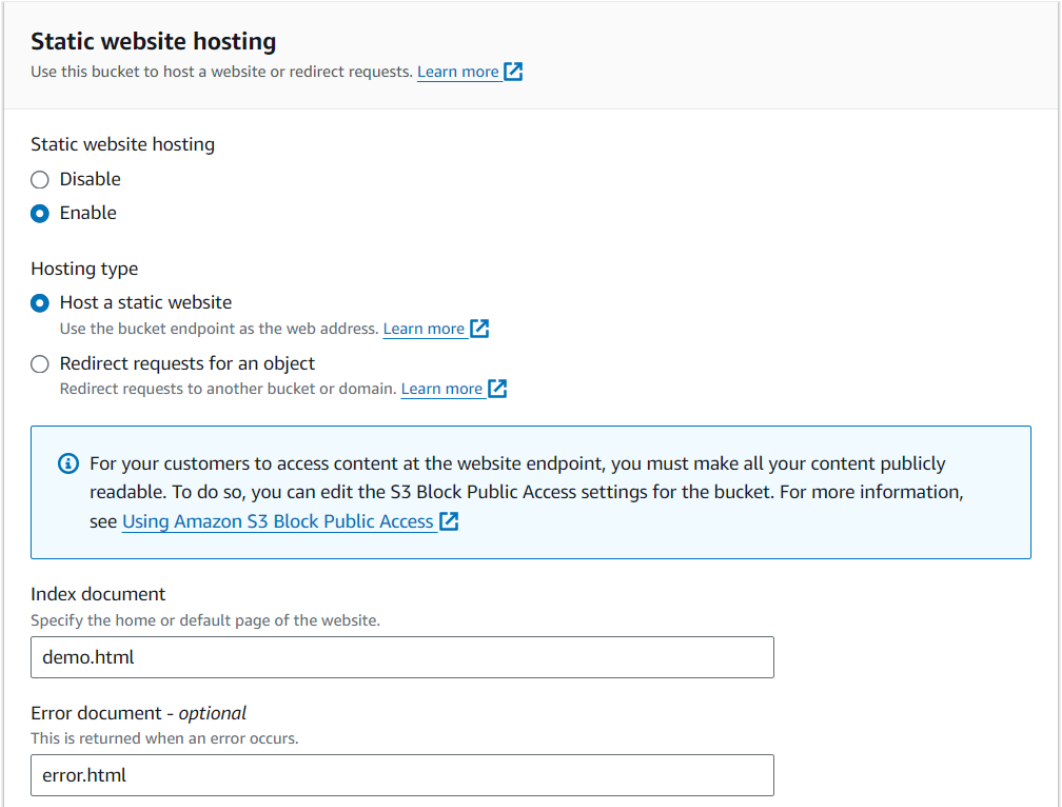
After adding the files click on upload files.



Step 5: Go to the properties tab and navigate to the “Static website hosting” section and click on edit , then click on enable.

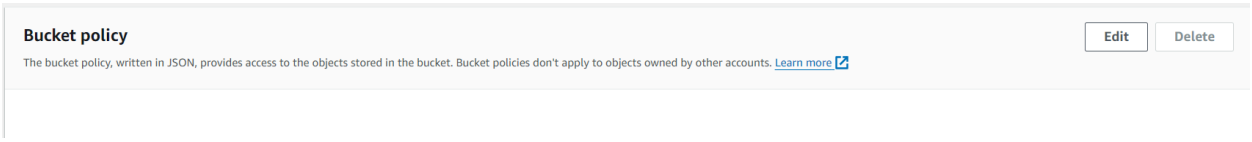


Specify the document/filenames and click on save changes.

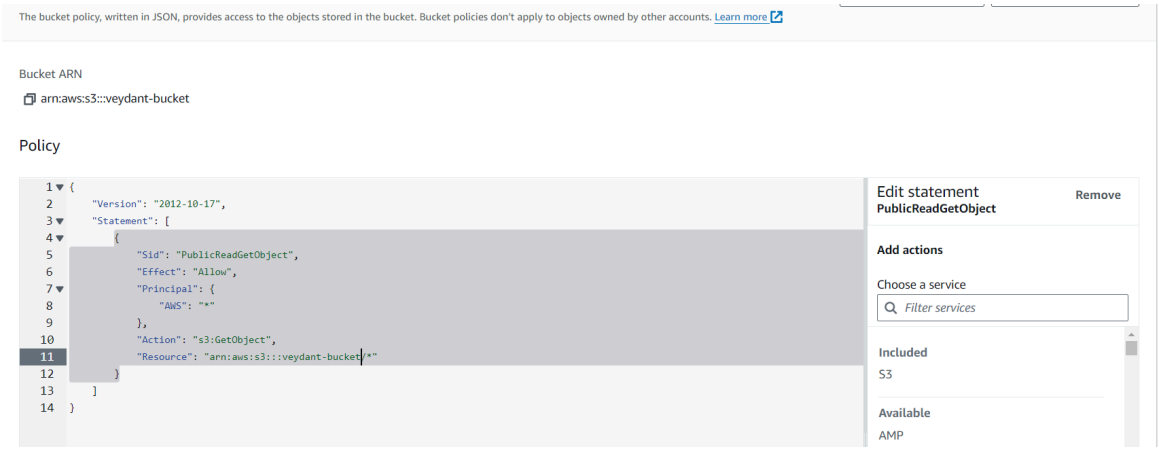


Name: Veydant Sharma Class: D15C Roll no.: 50

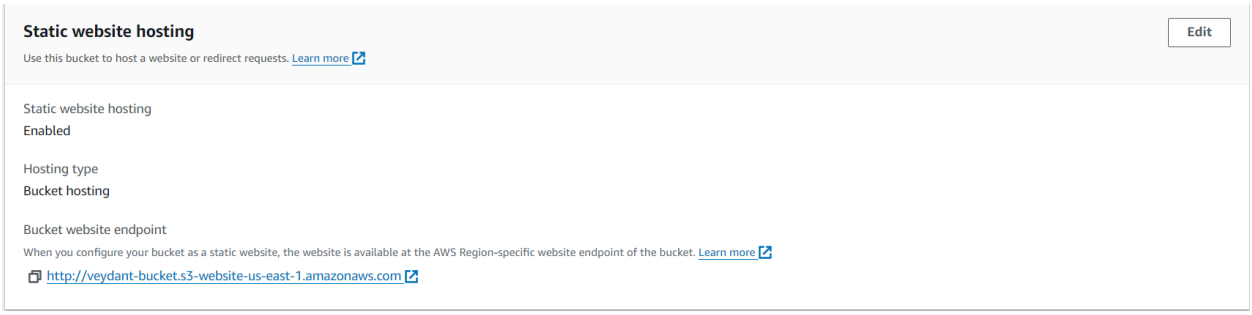
Step 6: Head to the Permissions tab and navigate to the “Bucket Policy” section and click on edit.



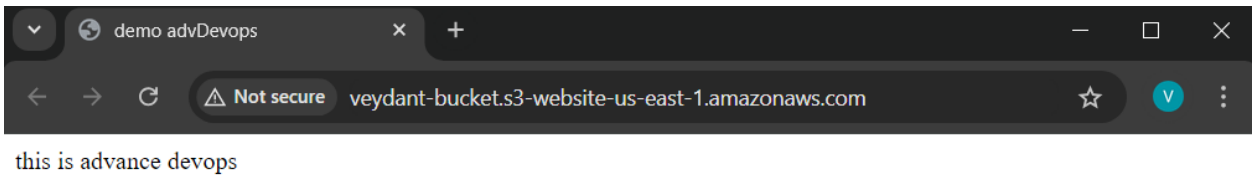
Add the bucket policy , save changes and exit.



Step 7: Once the Bucket policy has been updated , navigate back to “Static website hosting”, a link will be available, click on the link to view the hosted/deployed website.



Output:

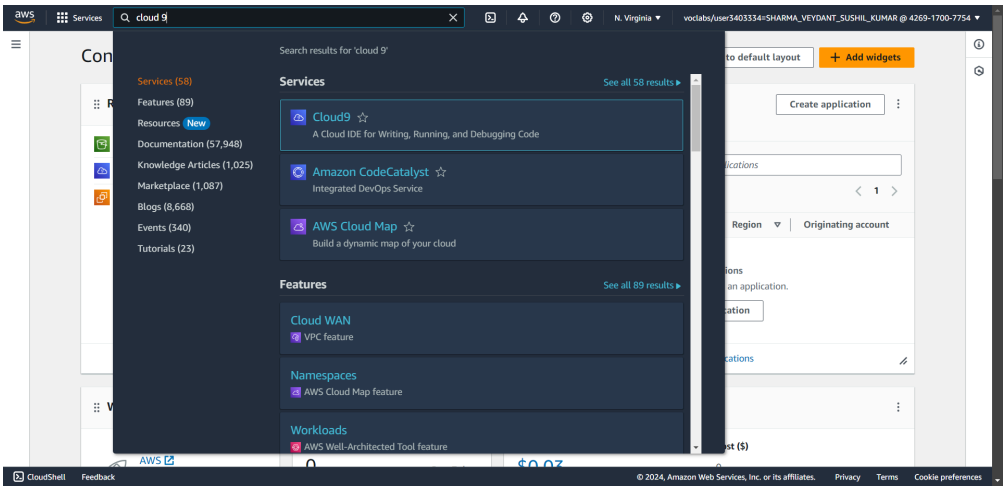


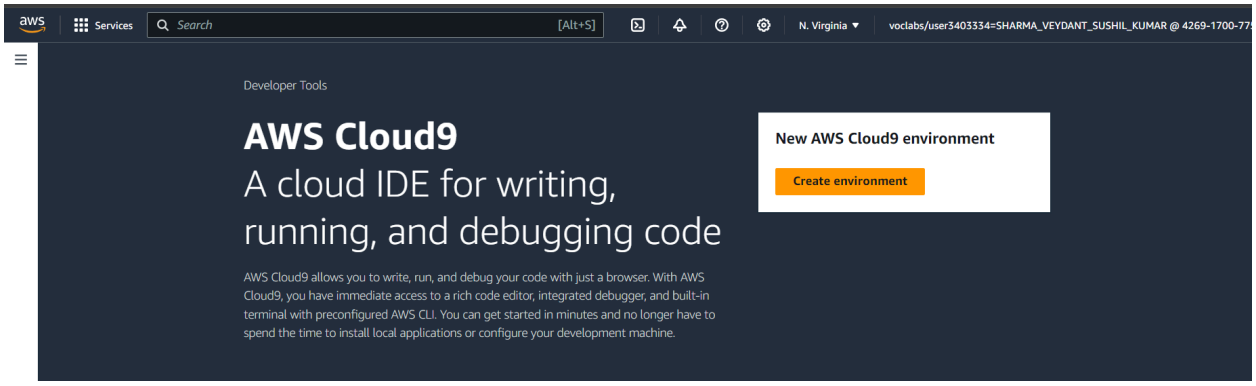
Veydant Sharma D15C 50 Adv DevOps

Experiment 1B

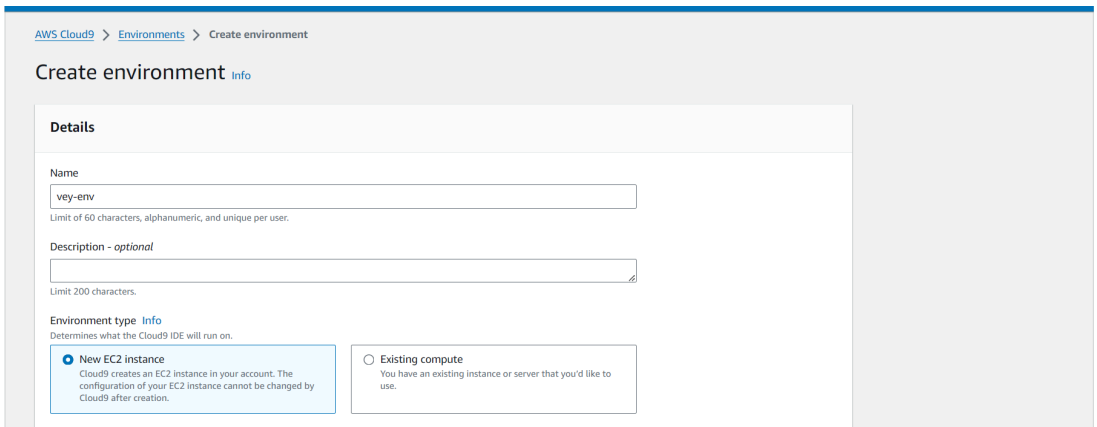
Aim: To understand the benefits of Cloud Infrastructure and Setup AWS Cloud9 IDE, Launch
AWS Cloud9 IDE and Perform Collaboration Demonstration.

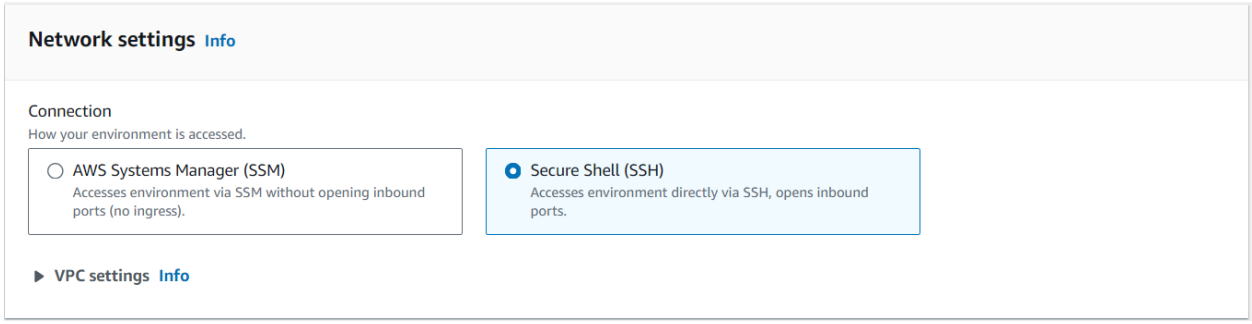
Step 1: Navigate to Cloud 9 service





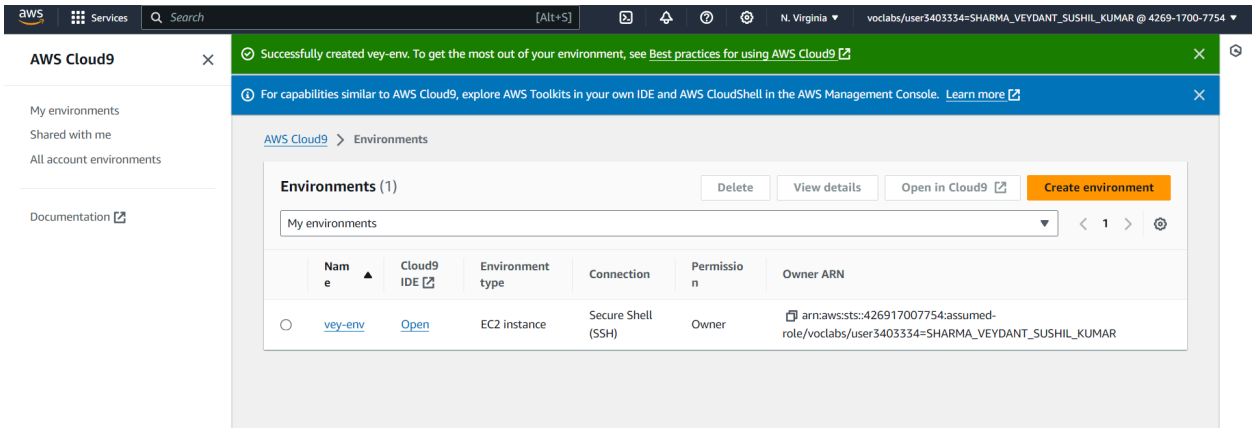
Step 2: Create Environment



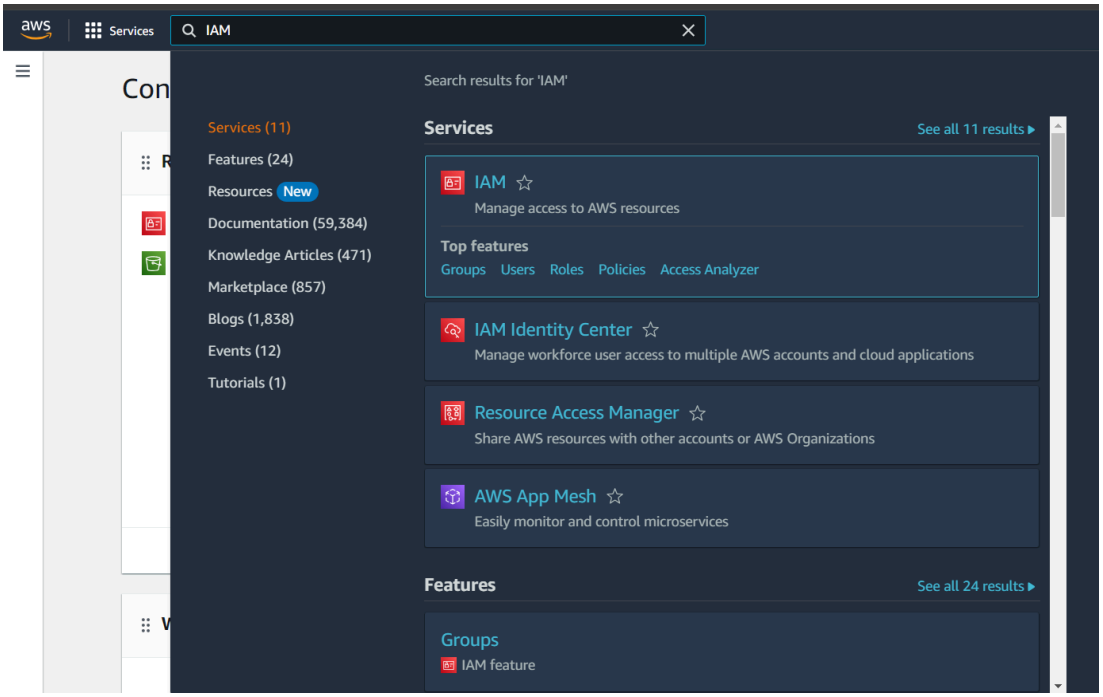


Click on “Create”

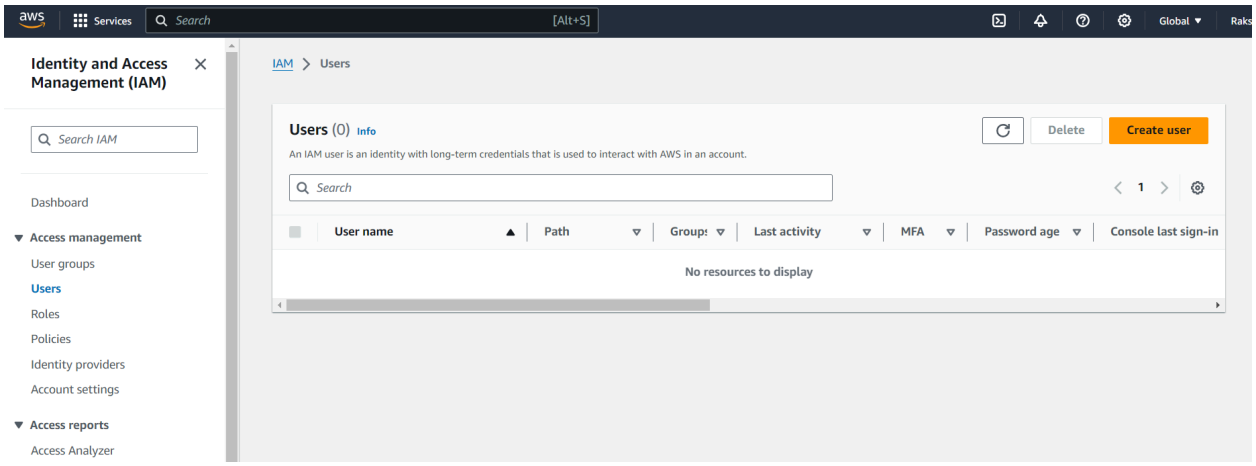
Step 3: Environment Will be created.



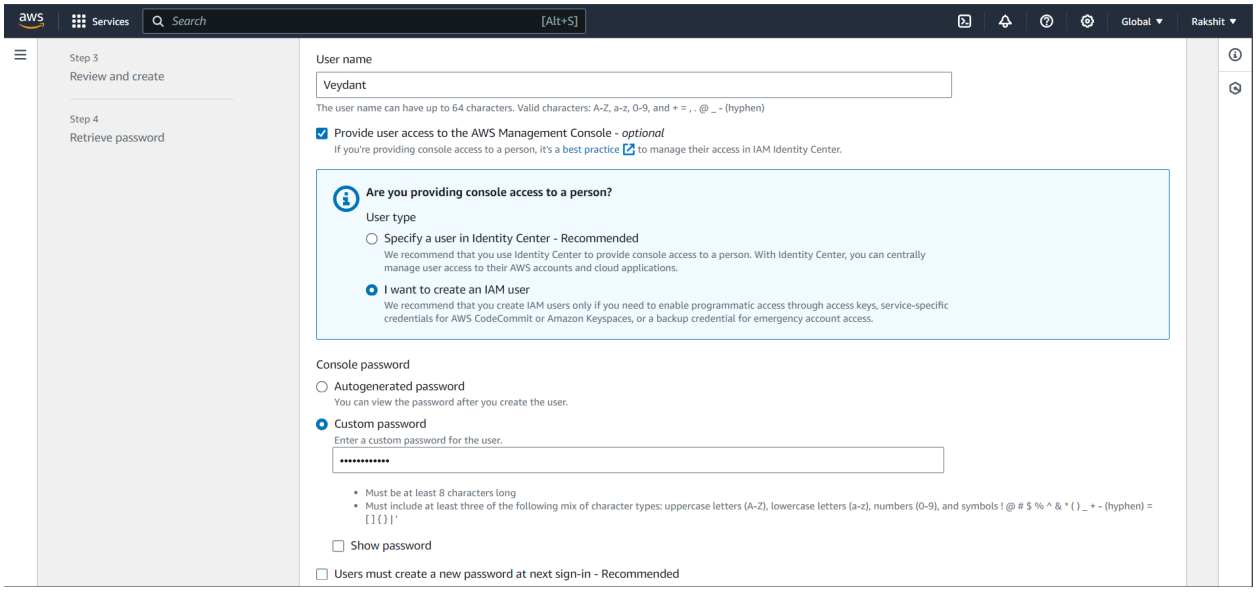
Step 4: Navigate to IAM service in AWS.

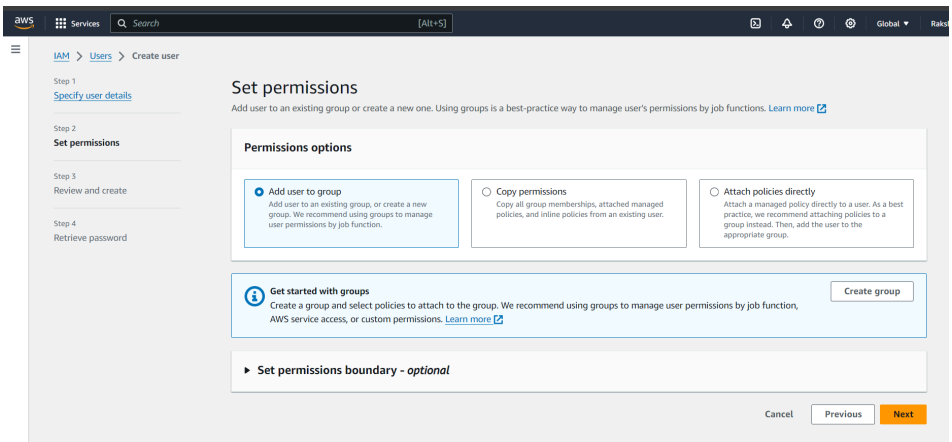


Step 5: Navigate to users tab and Create new User.



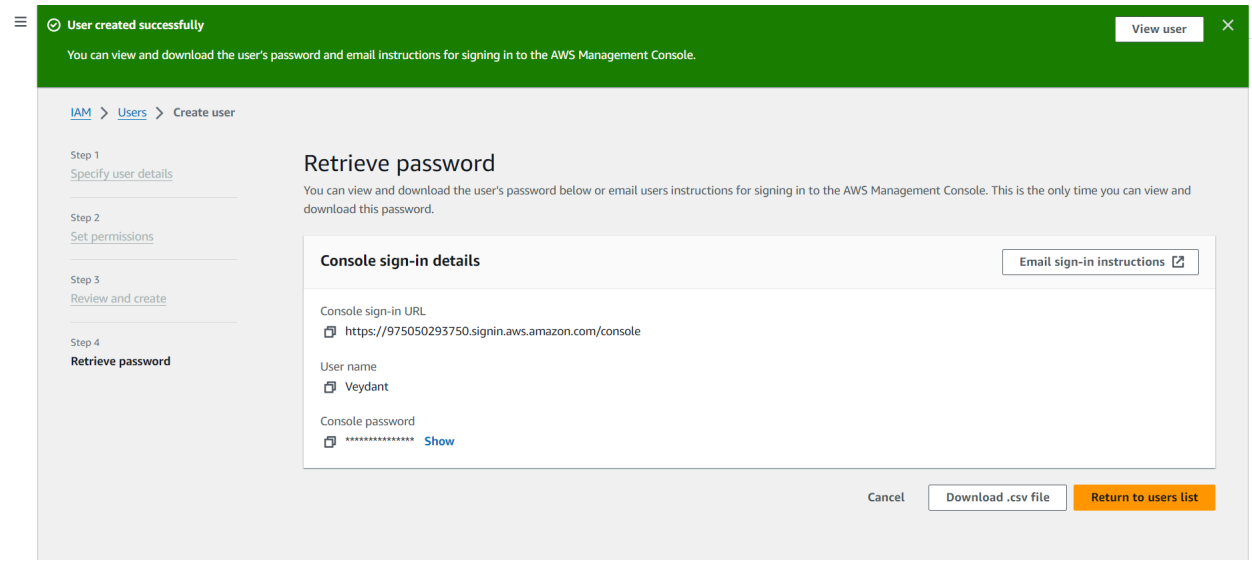
Add name, custom password. Keep the “Useers create new password at next sign in” button checked



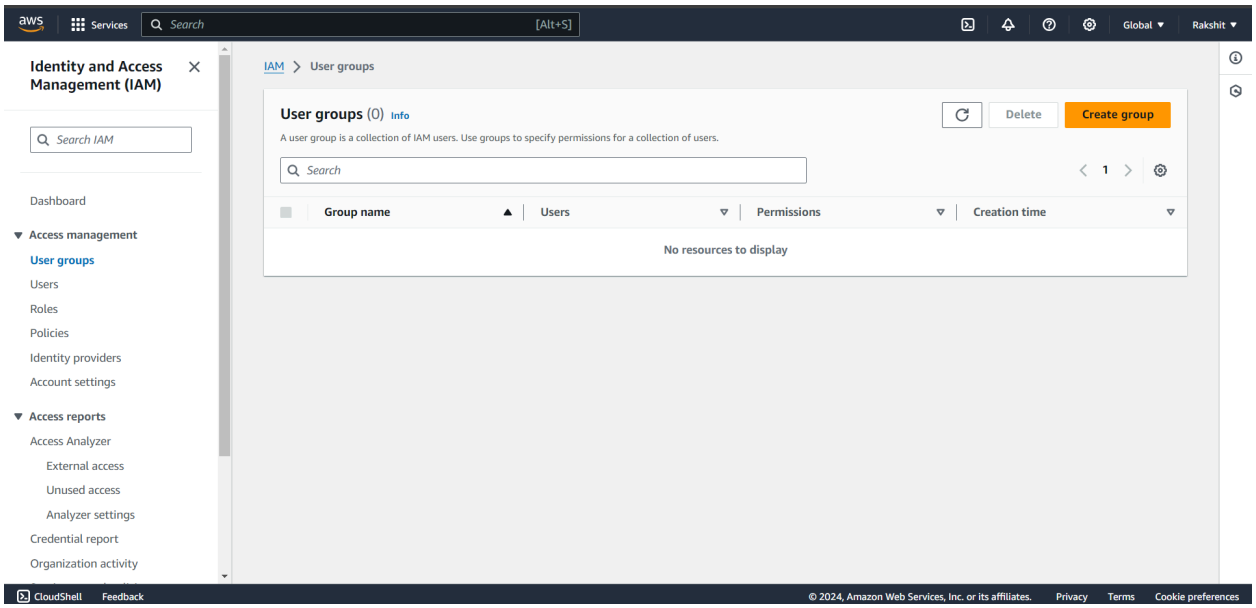


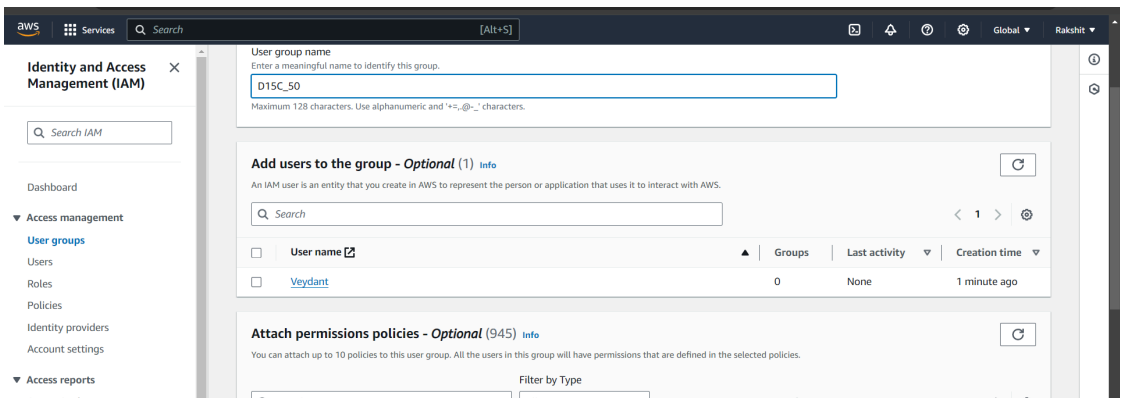
Veydant Sharma D15C 50 Adv DevOps

Select default settings for the rest.

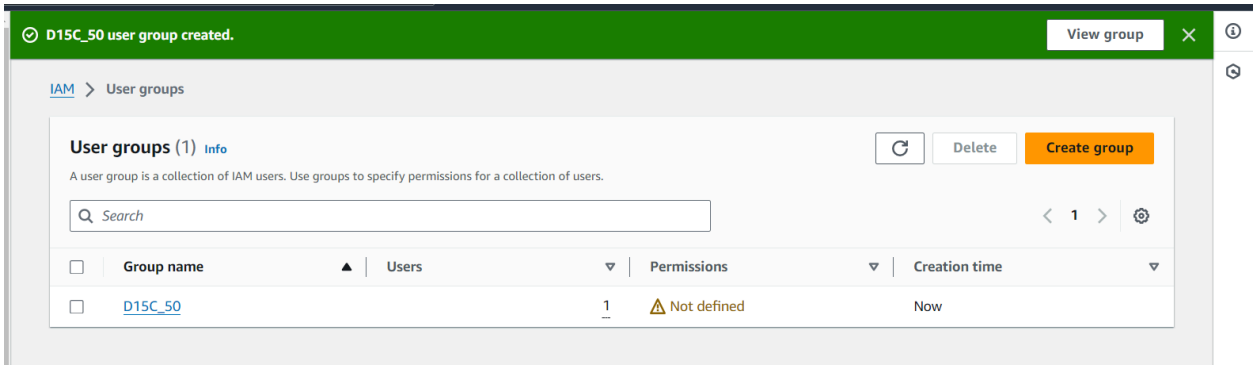


Step 6: Navigate to User groups tab.

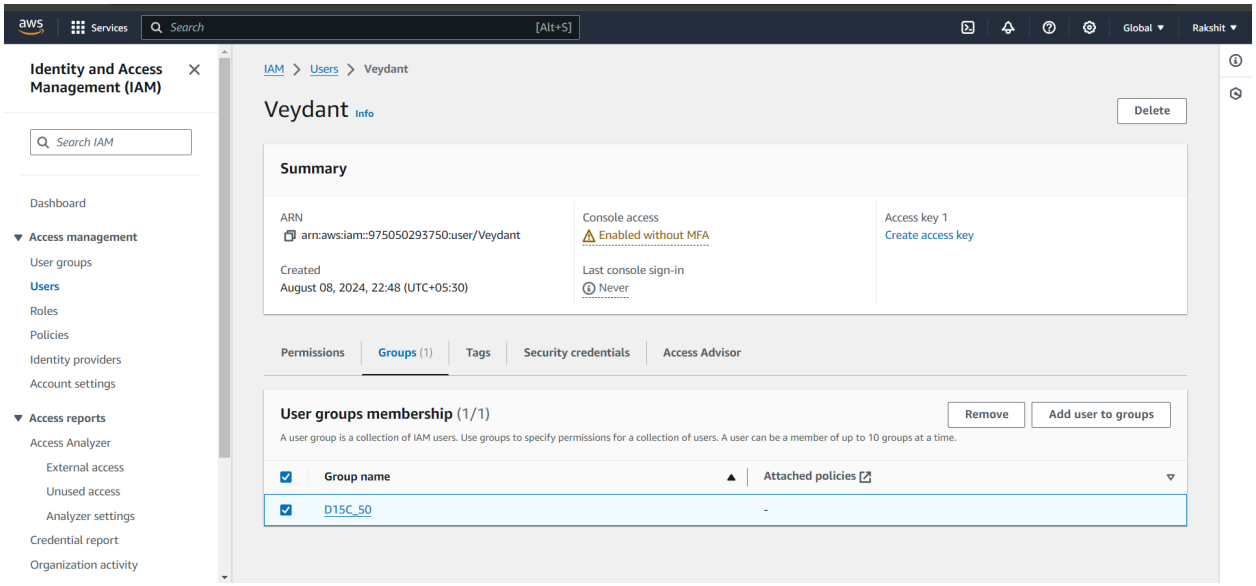


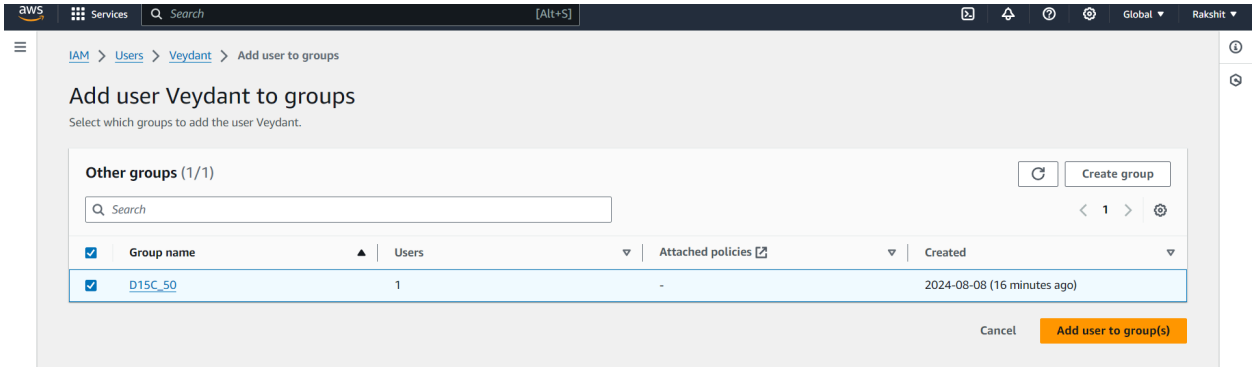


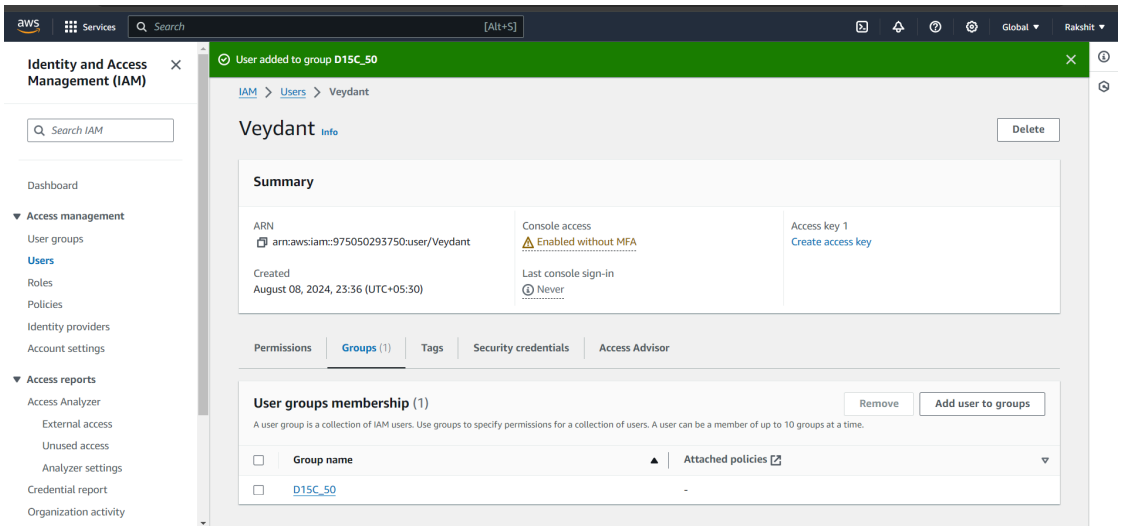
Create a new user group.



Step 7: Navigate to Users tab , go to Groups tab and add User to the group.

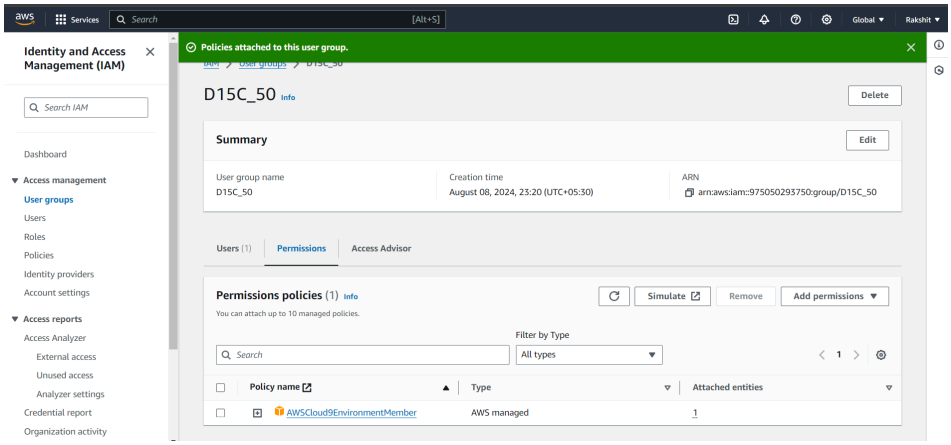






The user has been added to the the user group.

Step 8 : Navigate to the permissions tab in User group, select “AWSCloud9EnvironmentMember” and add the permissions.

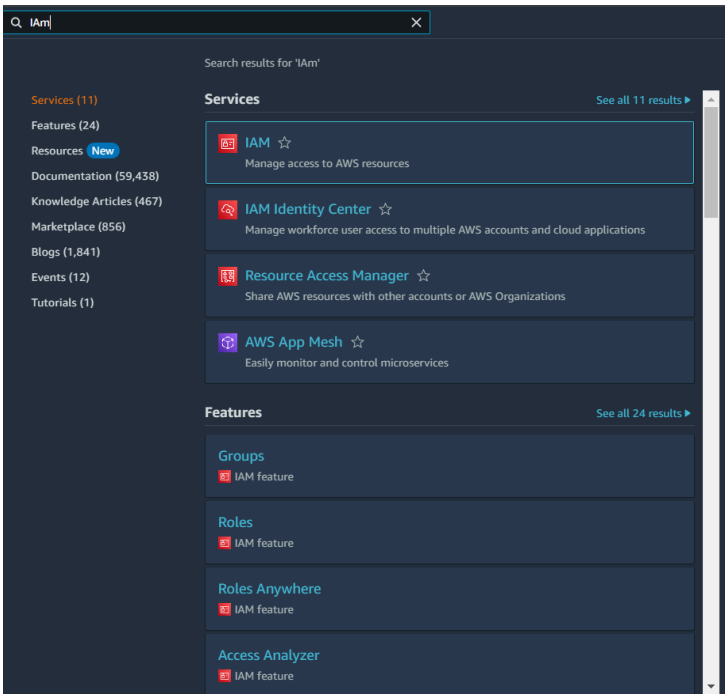


Name: Veydant Sharma Class: D15C Roll No.:50

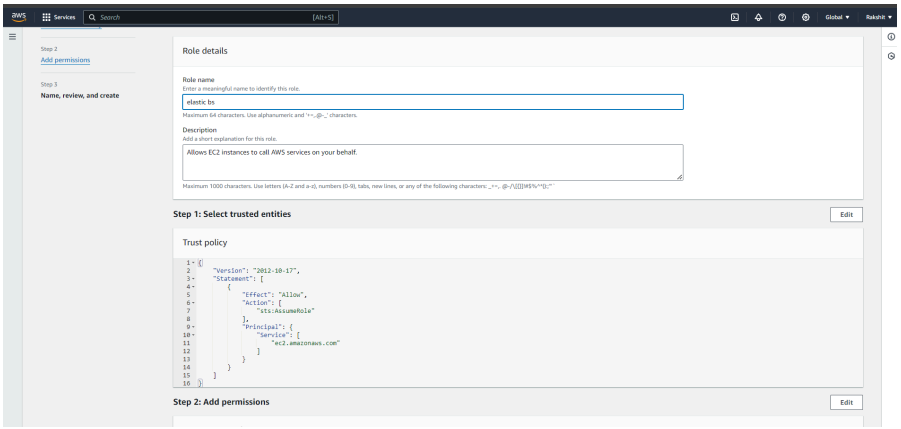
Experiment No: 2

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.

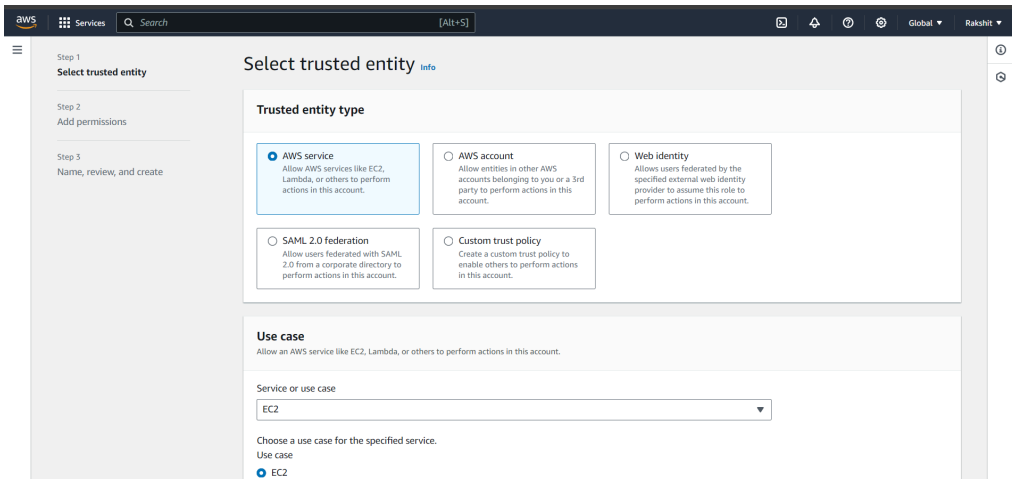
Step 1: Search for IAM , Go to Roles section and click on create a role.



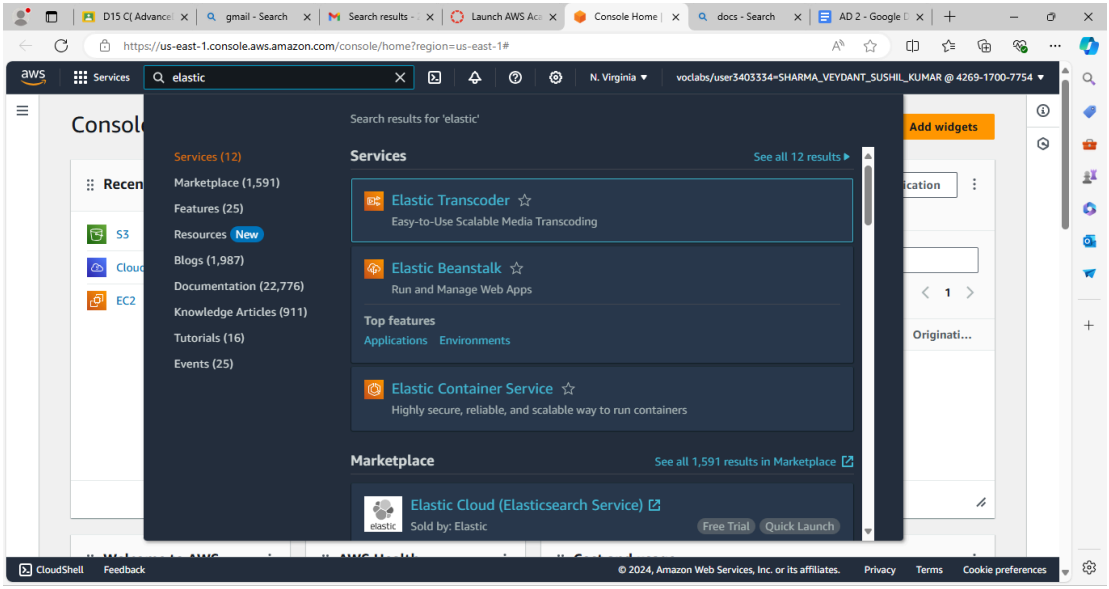
Step 2: Fill the role details. Select AWS service and enable web tier and work tier permissions.

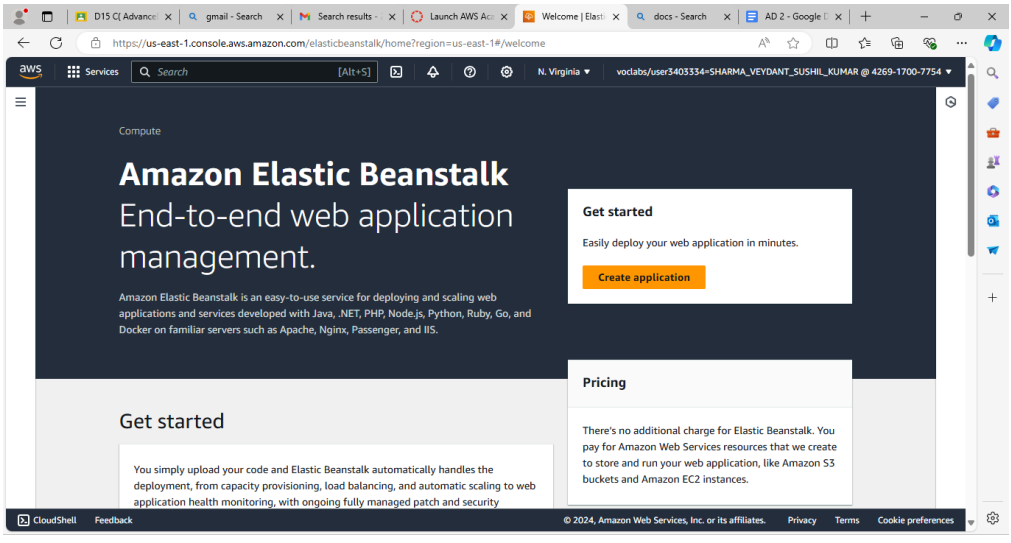


Name: Veydant Sharma Class: D15C Roll No.:50



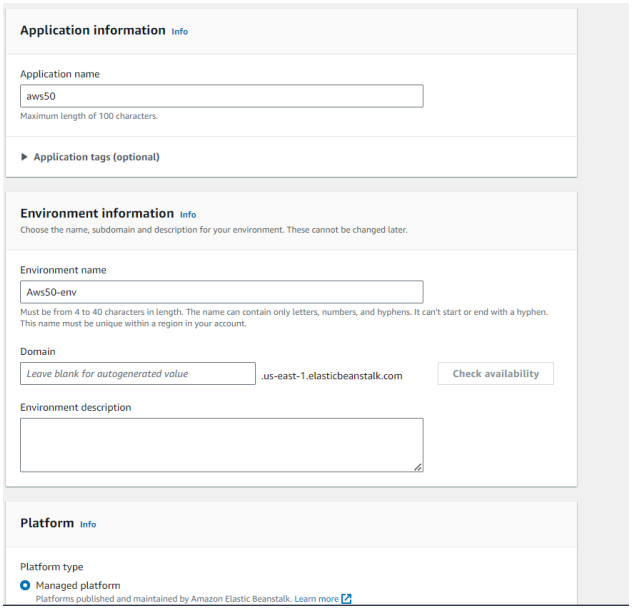
Step 3: Search elastic beanstalk . After loading the page click on Create application.



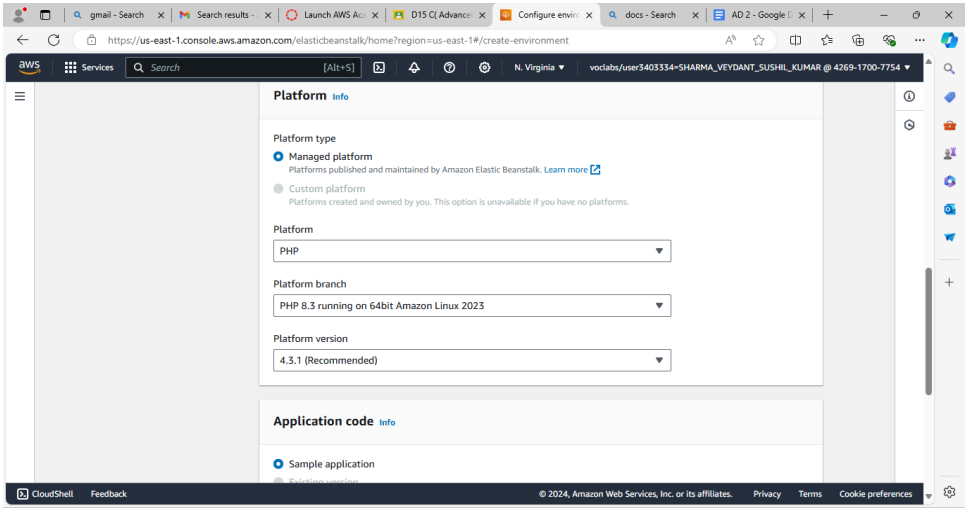


Name: Veydant Sharma Class: D15C Roll No.:50

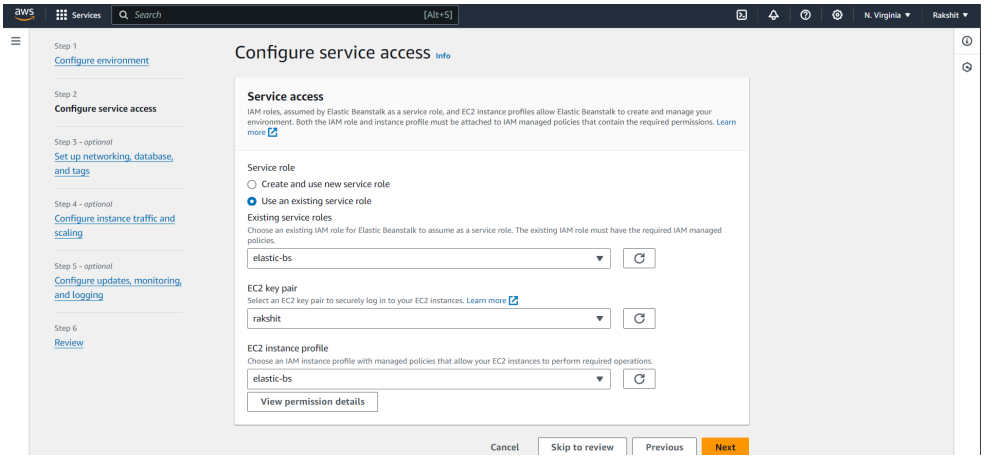
Step 4: Create application , provide application name and environment



Select the platform.

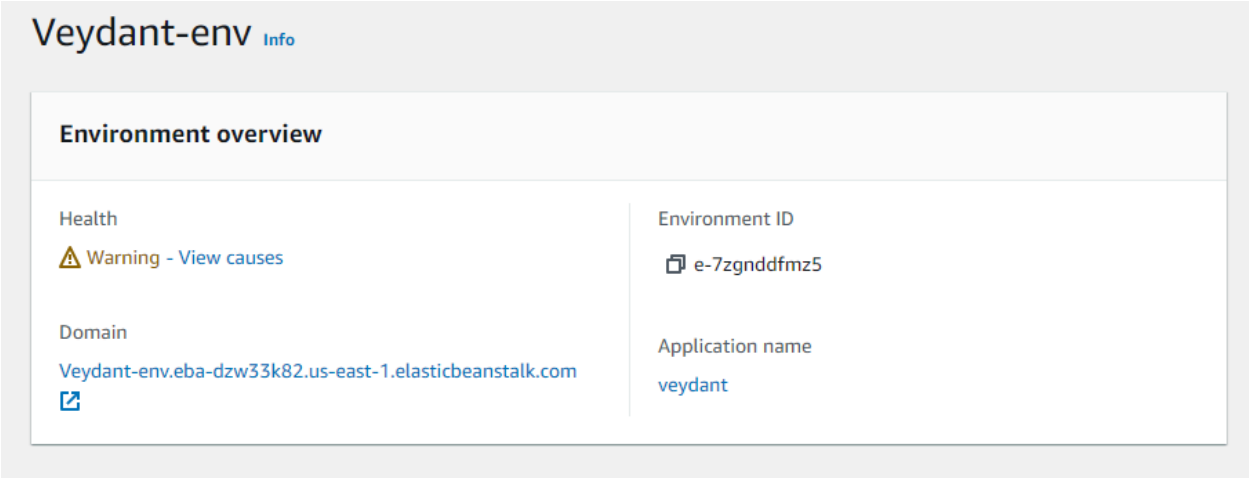


Click on ‘Use existing service role’ in the service menu . Assign the service role as the one create in IAM, also assign the key pair.

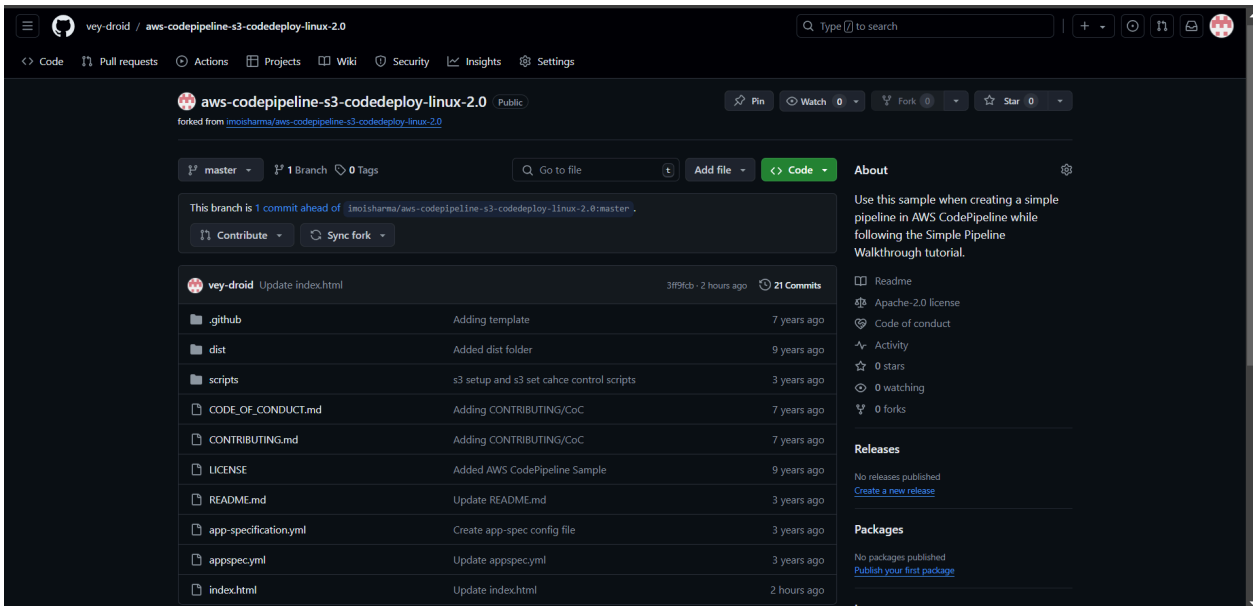


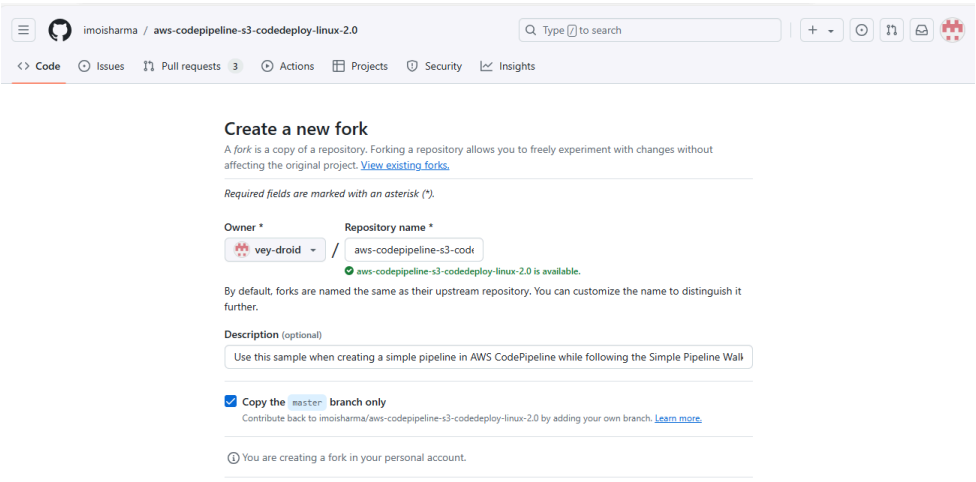
Name: Veydant Sharma Class: D15C Roll No.:50

Keep the rest of the settings as default. Wait for the environment to be created/launched successfully.



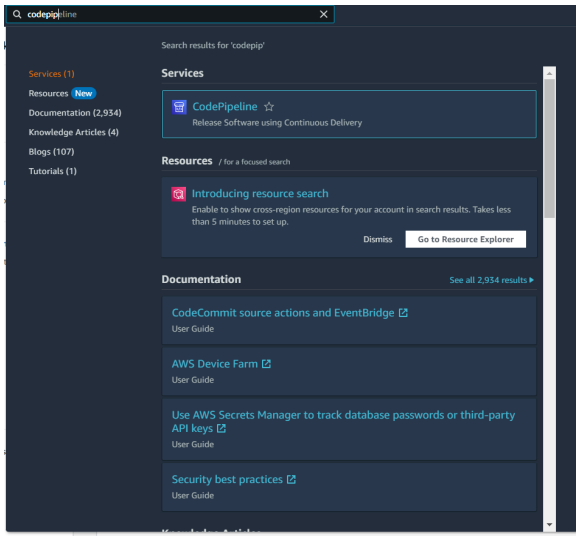
Step 5: Get a sample of the code from github , fork the repository.



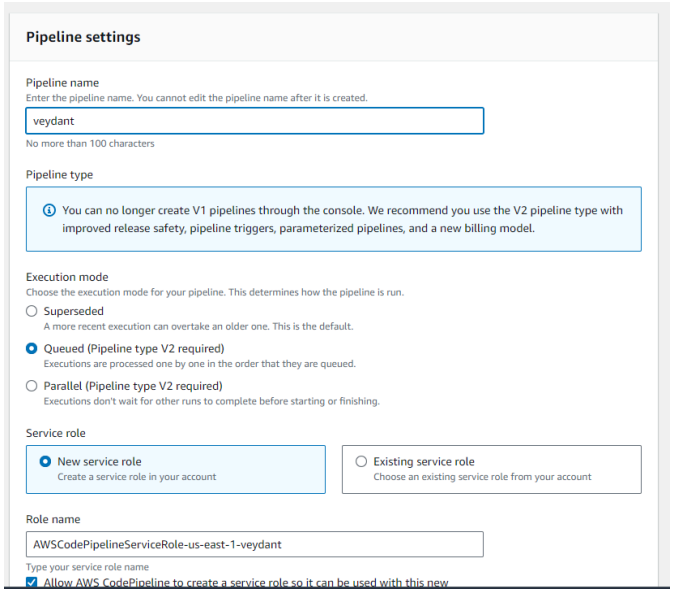


Name: Veydant Sharma Class: D15C Roll No.:50

Step 6: Go to codepipeline, Create new pipeline.

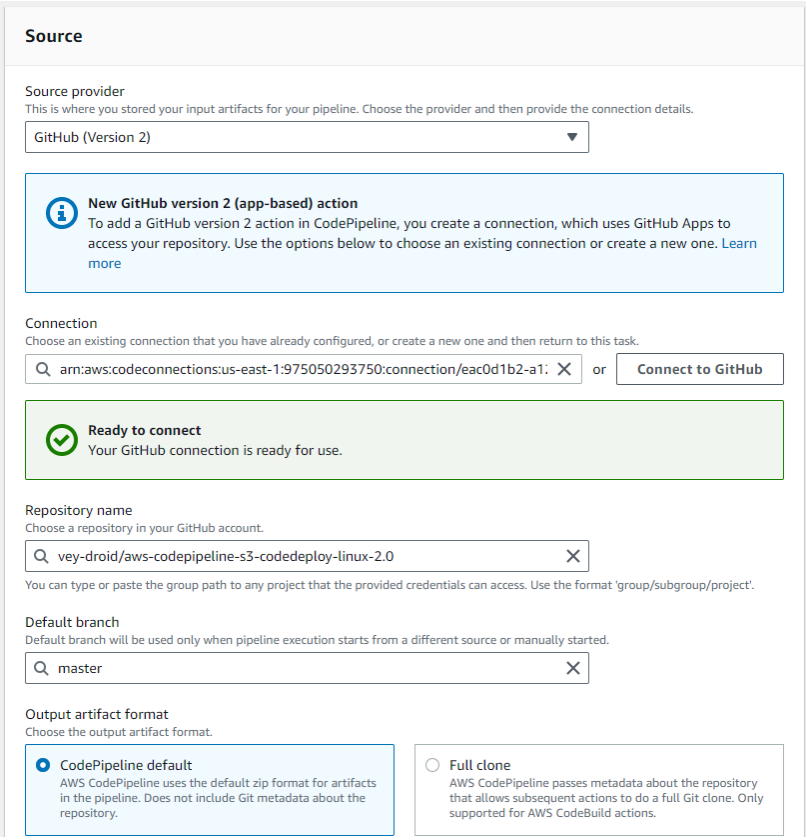


Step 7: Name the pipeline. Head on to the ‘Source’ Page.



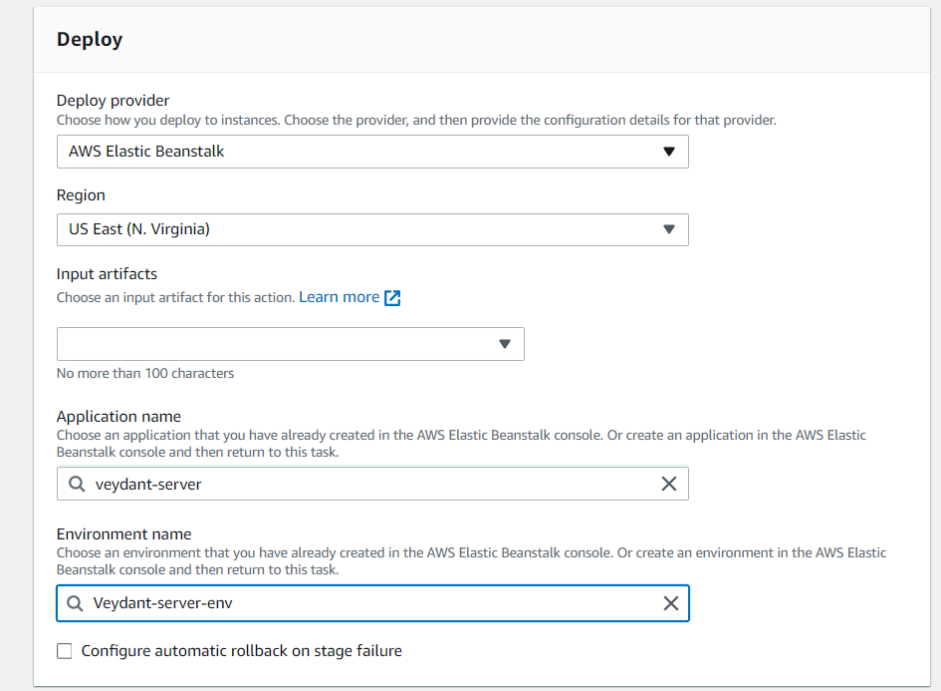
On the source page, select source provider as Github V2 , connect github account, and select the repository and the branch.

Name: Veydant Sharma Class: D15C Roll No.:50



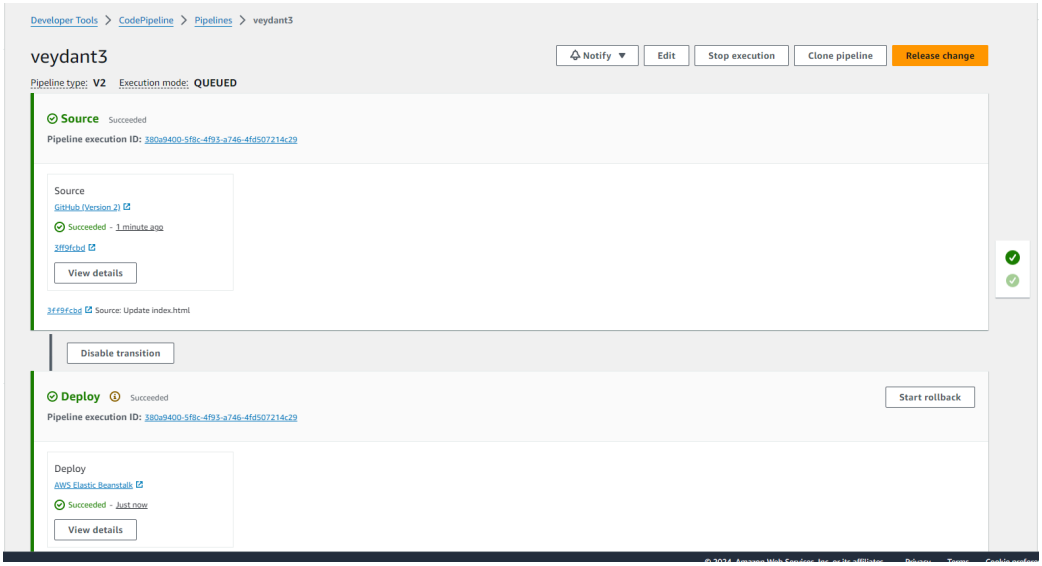
Select defaults and skip the build page.

On the deploy page , select provider as AWS Elastic Beanstalk. Select the Application and environment name, then proceed further.

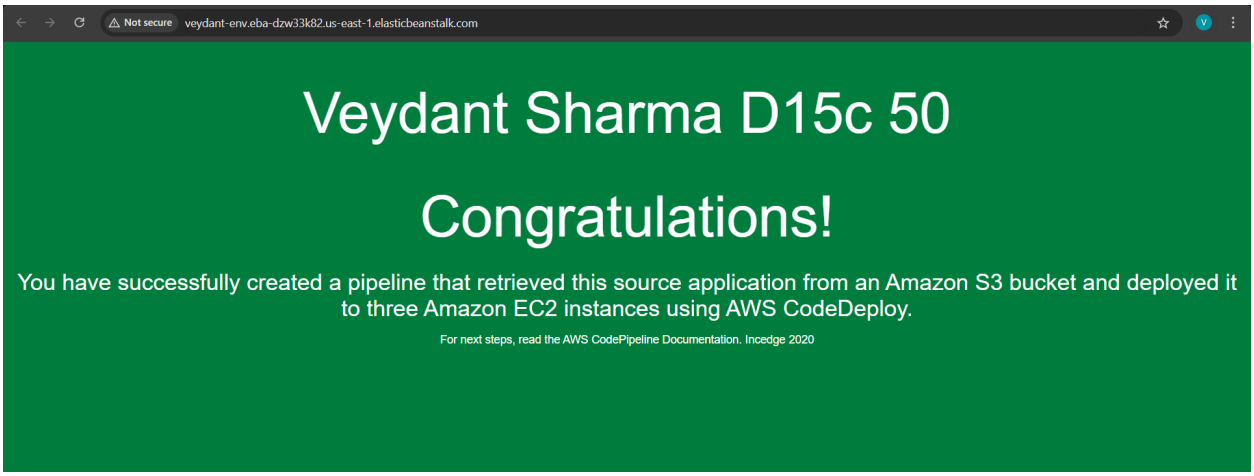


Name: Veydant Sharma Class: D15C Roll No.:50

Step 8: The pipeline will be launched in a few moments, once it is successfully created head over to elastic beanstalk environment.



Step 9: A URL will be created on the environment page, that is the hosted link, click on it.

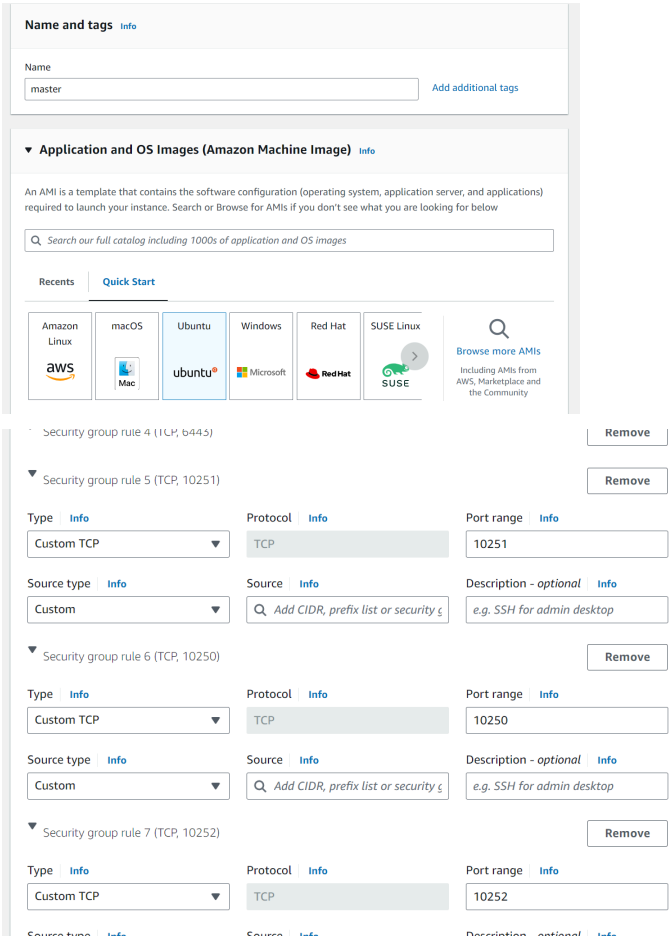


Veydant Sharma D15C 50 AdvDevop Experiment 3

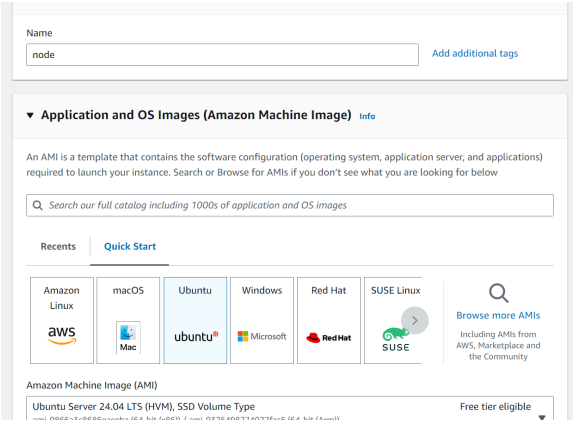
Aim: To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud Platforms.

Step 1: Make 3 EC2 Instances, 1 Master 2 Node instances, use Ubuntu , t2.medium adn add the security groups

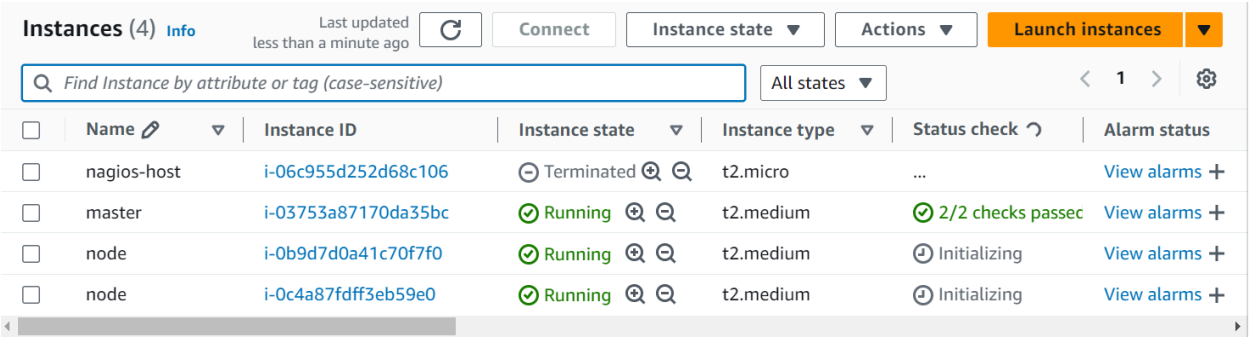
Master:

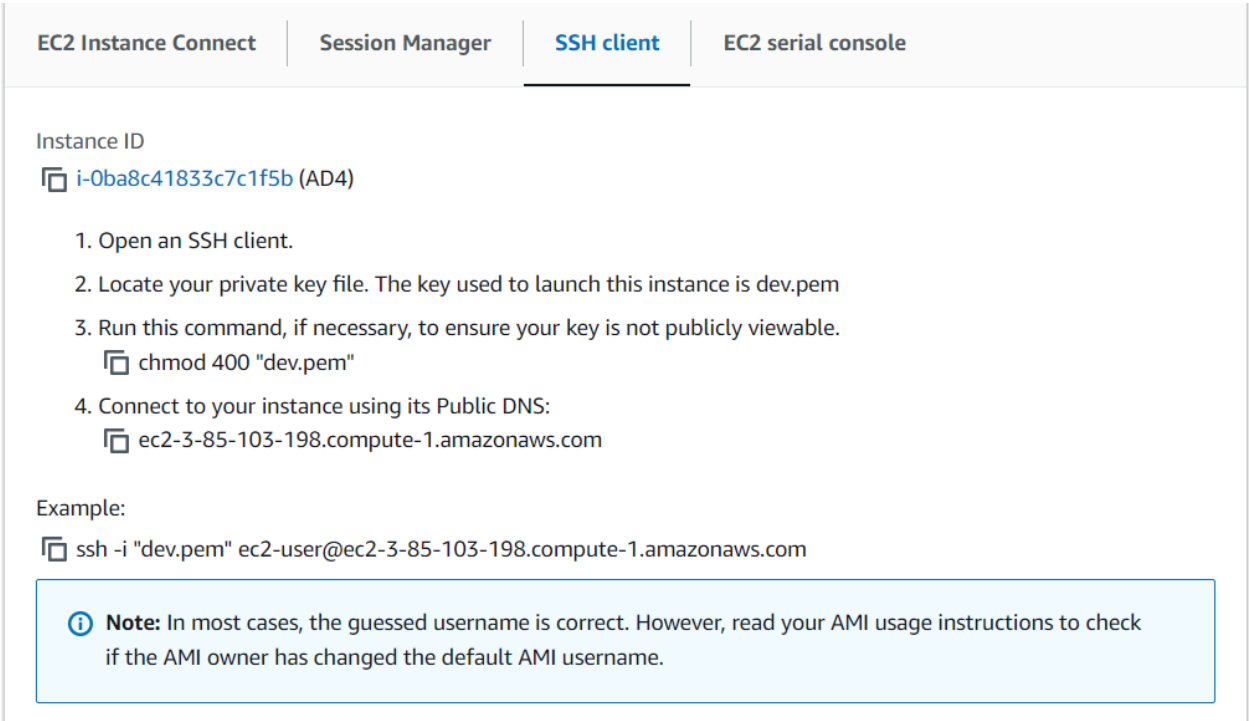


Worker:



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"

```
ubuntu@ip-172-31-81-239:~$ 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
XYukHjlxxEgBdC/J3cMMNRE1F4NCA3ApfV1Y7/hTeOnmDuDYwr9/obA8t016Yljj
q5rdkywPf4JF8mXUW5eCN1vAFHxeg9ZWemhBtQmGxXnw9M+z6hWwc6ahmwARAQAB
tCtFb2NrZXIgUmVsZWFzZSAoQ0UgZGViKSA8ZG9ja2VyQGRvY2tlci5jb20+iQI3
```

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>
```

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

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.

```
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.

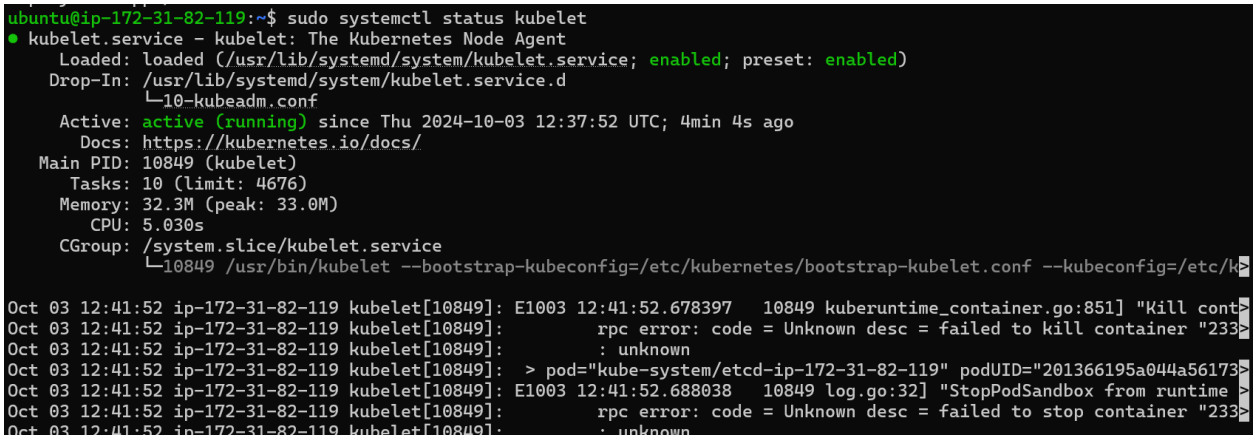
```
ubuntu@ip-172-31-82-119:~$ kubectl get nodes
NAME                STATUS     ROLES           AGE     VERSION
ip-172-31-82-119    NotReady   control-plane   2m56s   v1.31.1
ip-172-31-84-169    NotReady   <none>          43s     v1.31.1
ip-172-31-87-189    NotReady   <none>          39s     v1.31.1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

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

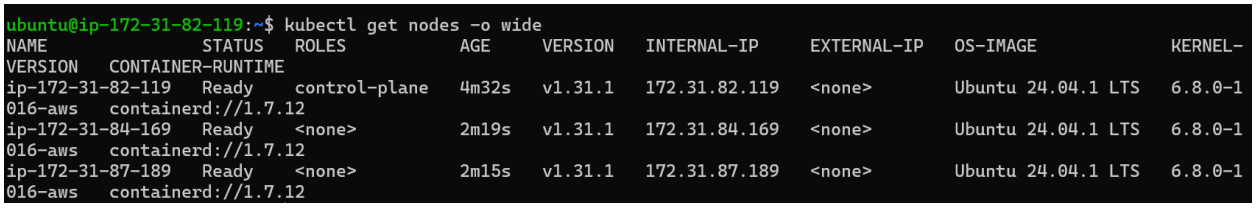
kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml

```
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

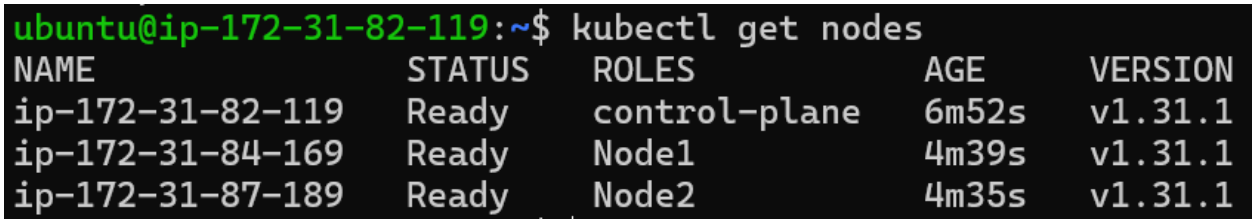


Now Run command kubectl get nodes -o wide we can see Status is ready.



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.



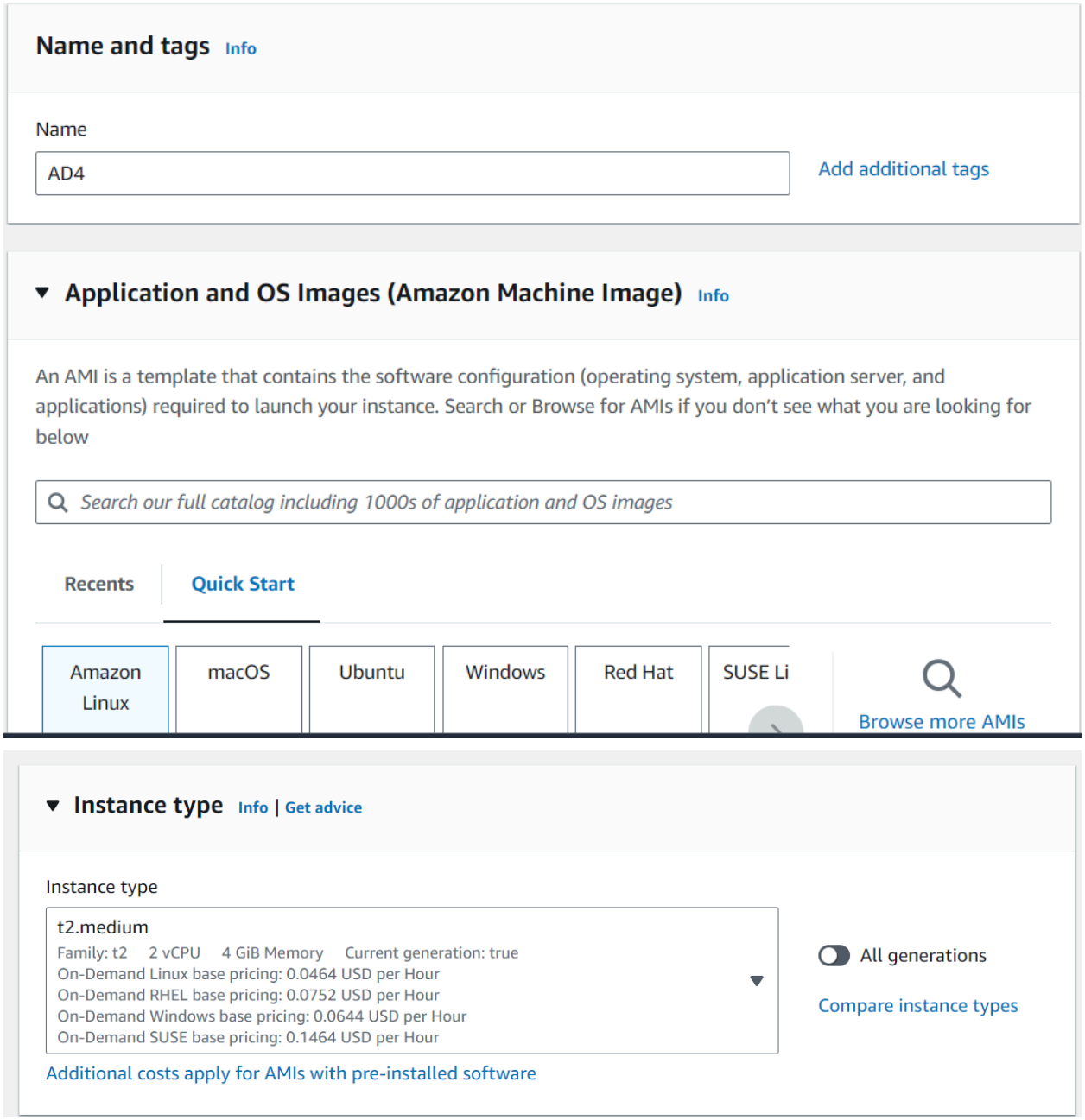
Conclusion:

- Successfully set up a Kubernetes cluster on AWS EC2 instances.
- Provisioned three EC2 instances: Master, Node1, and Node2, with appropriate security rules for internal communication.
- Securely SSHed into each instance to manage installations.
- Installed Docker as the container runtime on all instances.
- Installed Kubernetes on each instance and initialized the cluster from the master node.

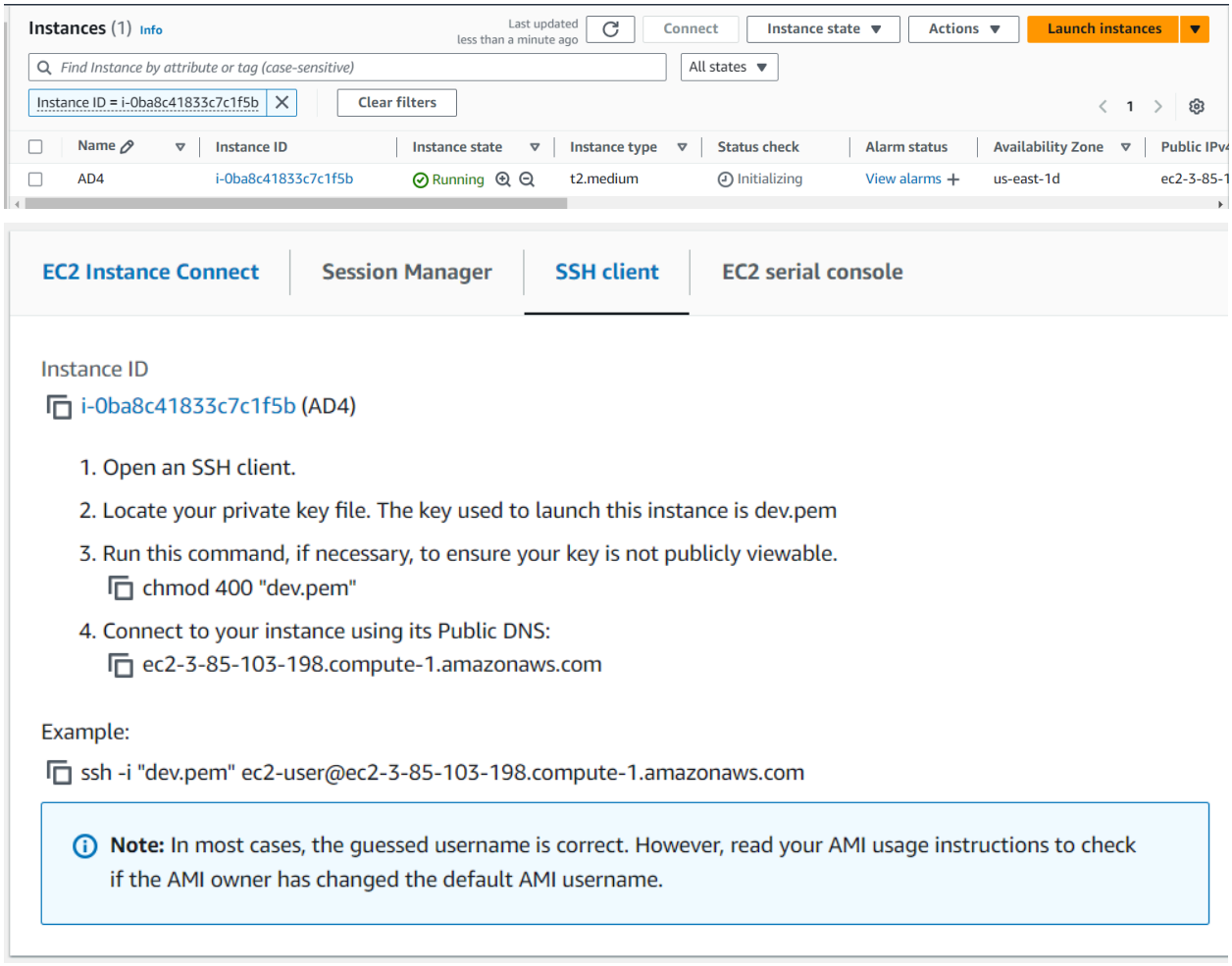
- Connected the worker nodes to the master using the kubeadm join command, forming a cohesive Kubernetes cluster.
- Implemented a network plugin to ensure proper communication between nodes.
- Verified that the nodes' status transitioned to "Ready."

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

Step 1: Login to AWS Account, Select EC2 instance, and choose instance type as t2.medium



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 command

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

```
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'
```

TO fix the error:
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
```

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: we can deploy our nginx server on this cluster.

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
```

```
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 : To fix the eror:

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,

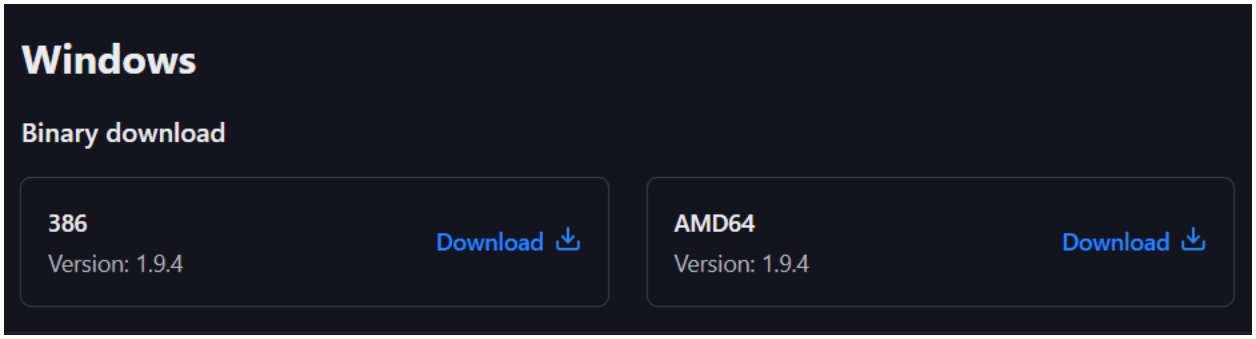
- Successfully set up a Kubernetes cluster on an AWS EC2 instance using kubectl.
- Deployed a sample nginx application to the cluster.

- Installed Docker and Kubernetes, and configured containerd to ensure proper operations.
- Initialized the Kubernetes cluster to facilitate application deployment.
- Verified the successful deployment of the nginx server using kubectl port-forward.
- Confirmed access to the application via curl.
- Gained practical experience in managing Kubernetes clusters and deploying applications on cloud infrastructure.

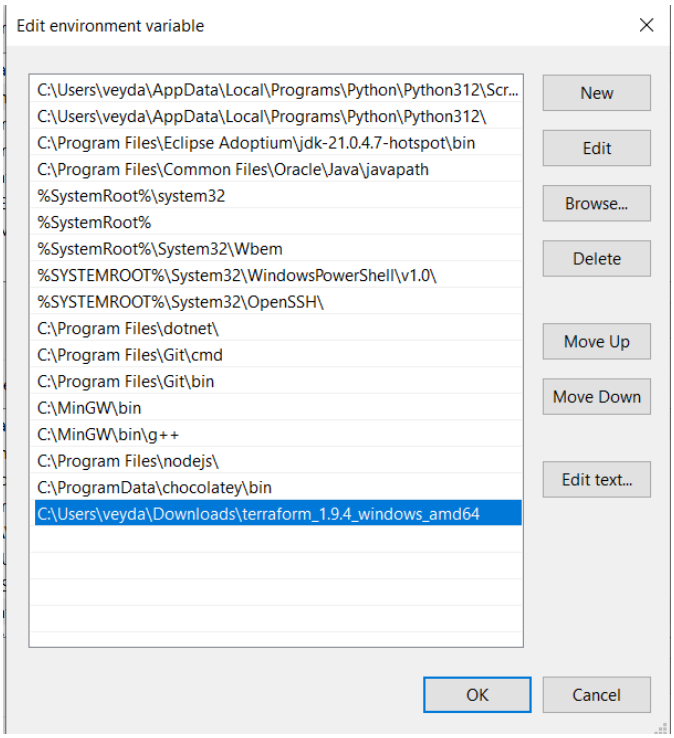
Name: Veydant Sharma Class: D15C Roll no.: 50 Sub: Adv DevOps Lab

Experiment 5

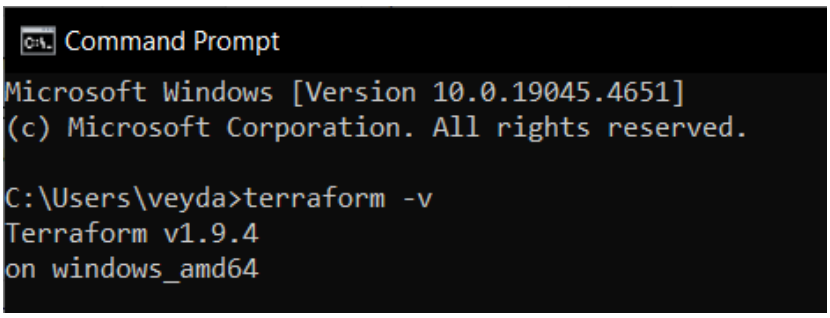
Step 1: Download terraform
To install Terraform, First Download the Terraform Cli Utility for windows from terraforms official website



Step 2: Extract the files from the download.
Nam
Step 3: Add terraform path to the environment variable



Step 4: Check if terraform was installed by going to the command prompt and type ‘terraform -v’.



Name: Veydant Sharma Class: D15C Roll no.: 50 Sub: Adv DevOps Lab

Step 5: Open Powershell and check terraform functionality.

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

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\veyda> terraform
Usage: terraform [global options] <subcommand> [args]

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

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

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

Global options (use these before the subcommand, if any):
  -chdir=DIR    Switch to a different working directory before executing the
                given subcommand.
  -help         Show this help output, or the help for a specified subcommand.
  -version      An alias for the "version" subcommand.
PS C:\Users\veyda>
```

Veydant Sharma D15C 50 AdvDevops

Experiment 6

A. Creating docker image using terraform

Prerequisite:

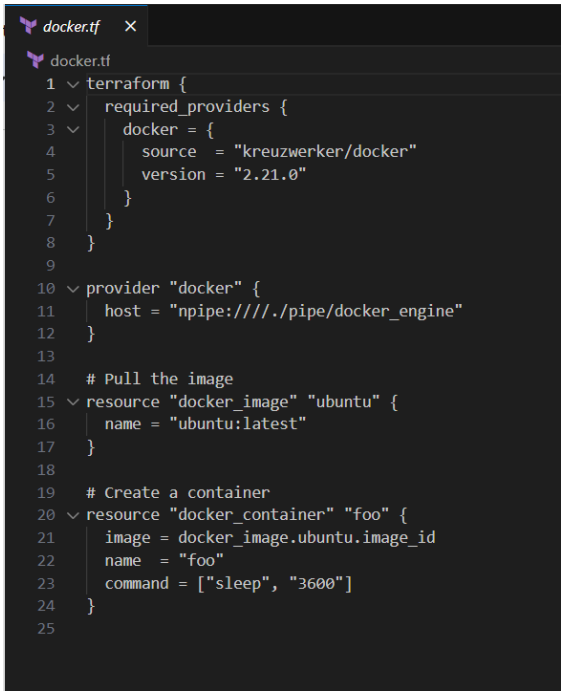
Step 1: Download and Install Docker Desktop from https://www.docker.com/
Check the docker functionality

```
PS C:\Users\veyda> 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
  checkpoint  Manage checkpoints
  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
```

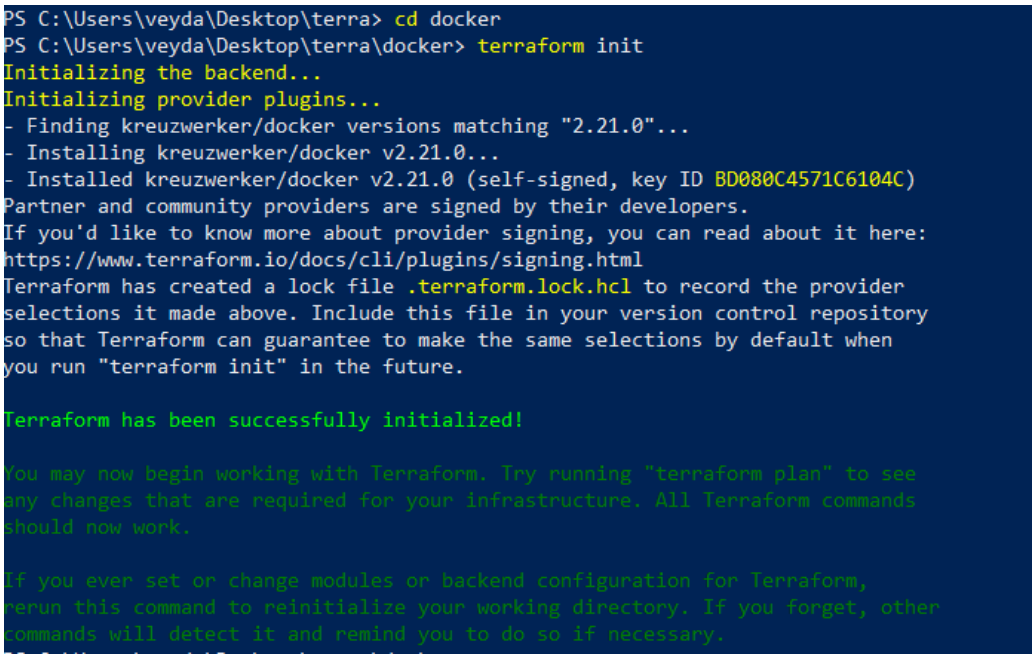
```
Command Prompt
Microsoft Windows [Version 10.0.19045.4780]
(c) Microsoft Corporation. All rights reserved.

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

```
Microsoft Windows [Version 10.0.19045.4780]
(c) Microsoft Corporation. All rights reserved.

C:\Users\veyda>docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
31e907dcc94a: Pull complete
Digest: sha256:8a37d68f4f73ebf3d4efafbcf66379bf3728902a8038616808f04e34a9ab63ee
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest

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

C:\Users\veyda>docker run -it ubuntu
root@0ea95fa5b779:/#
root@0ea95fa5b779:/#
```

Step 2: Create a folder called terra (do not use the word terraform or scripts as that may b e a keyword.)and then create a folder called docker inside it.
Then create a file called docker.tf



Step 3: Go to powershell, switch directory to where the docker.tf script was saved, then run the terraform init command.



Step 4: Execute terraform plan to view resources.

Veydant Sharma D15C 50 AdvDevops

```
PS C:\Users\veyda\Desktop\terra\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
      + name             = "foo"
      + network_data     = (known after apply)
      + read_only        = false
      + remove_volumes   = true
      + restart          = "no"
      + rm               = false
      + runtime          = (known after apply)
      + security_opts    = (known after apply)
      + shm_size         = (known after apply)
      + start            = true
      + stdin_open       = false
      + stop_signal      = (known after apply)
      + stop_timeout     = (known after apply)
      + tty              = false

      + healthcheck (known after apply)
```

Step 5: terraform show command for viewing all the state files

```
PS C:\Users\veyda\Desktop\terra\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          = "dfaafae11b3b"
    id                = "dfaafae11b3bf9d4a108b967b3c24943543a9b381fc75576cc06cd4fce85ee72"
    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
    publish_all_ports = false
    read_only         = false
    remove_volumes    = true
    restart           = "no"
    rm                = false
    runtime           = "runc"
    security_opts     = []
    shm_size          = 64
```

Step 6: Execute Terraform apply to apply the configuration, which will automatically create and run the Ubuntu Linux container based on our configuration. Using command : “terraform apply”

Docker images before apply step:

```
PS C:\Users\veyda\Desktop\terra\docker> docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
PS C:\Users\veyda\Desktop\terra\docker>
```

Docker images after apply step:

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

Step 7: terraform refresh command is helpful to sync your state file with the real-world infrastructure without applying any changes.

```
PS C:\Users\veyda\Desktop\terra\docker> terraform refresh
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Refreshing state... [id=dfaafae11b3bf9d4a108b967b3c24943543a9b381fc75576cc06cd4fce85ee72]
```

Step 8: Execute Terraform destroy to delete the configuration, which will automatically delete the Ubuntu Container.

```
PS C:\Users\veyda\Desktop\terra\docker> terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Refreshing state... [id=dfaafae11b3bf9d4a108b967b3c24943543a9b381fc75576cc06cd4fce85ee72]

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",
```

Docker images After Executing Destroy step

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

Veydant Sharma D15C 50 AdvDevops

Veydant Sharma D15C 50 AdvDevops

Experiment 7

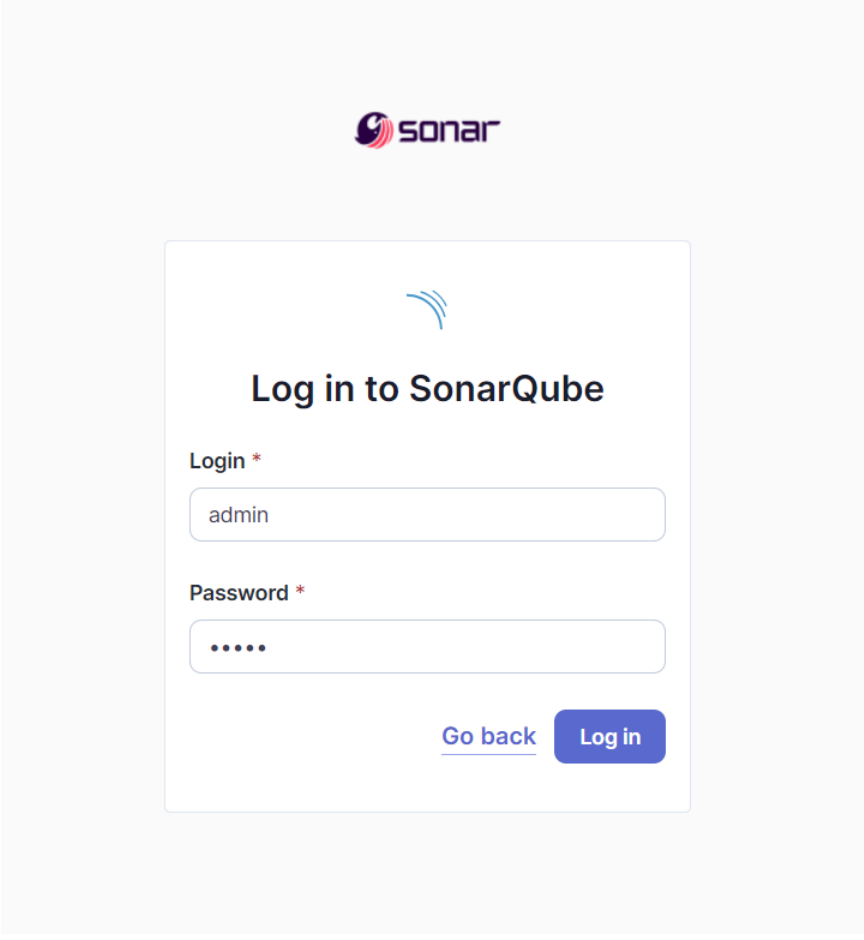
Aim: To understand Static Analysis SAST process and learn to integrate Jenkins SAST to SonarQube/GitLab.

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

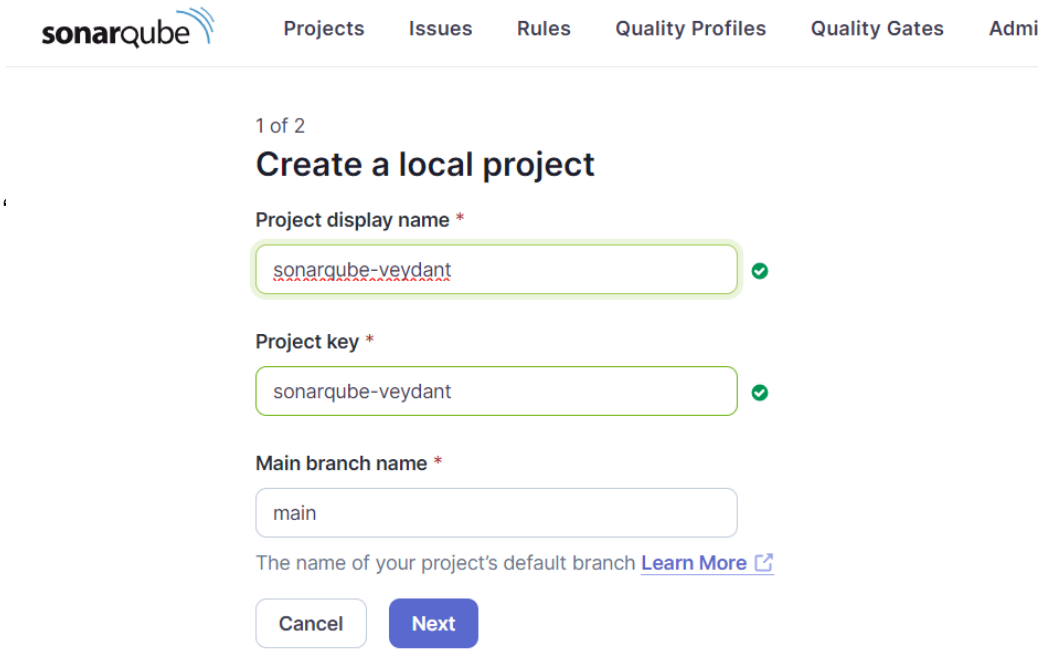
2. Run SonarQube in a Docker container using this command -

```
C:\Windows\system32>docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:late
st
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
b6dd73afc810a20ec3d643e9a148ab9643a3b5beff2766406df21f5f54a090c1
```

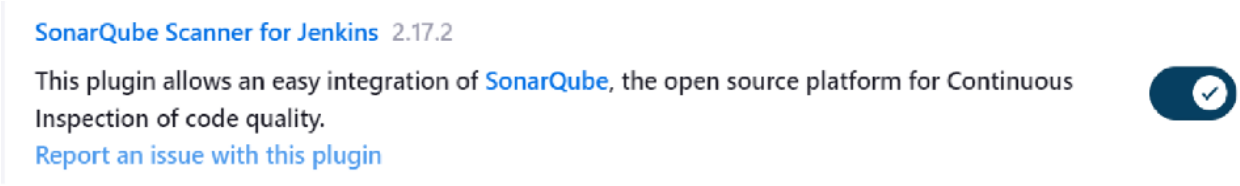
3. Once the container is up and running, you can check the status of SonarQube at localhost port 9000.



4. Login to SonarQube using username admin and password admin.
5. Create a manual project in SonarQube with the name sonarqube



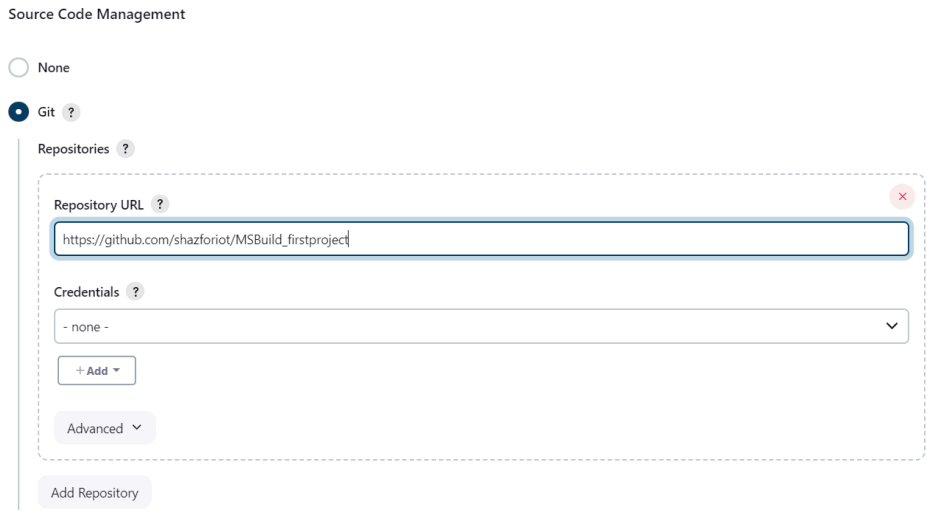
Setup the project and come back to Jenkins Dashboard.
Go to Manage Jenkins and search for SonarQube Scanner for Jenkins and install it.



6. Under Jenkins 'Configure System', look for SonarQube Servers and enter the details. Enter the Server Authentication token if needed.

7. Search for SonarQube Scanner under Global Tool Configuration. Choose the

latest configuration and choose Install automatically.
8. After the configuration, create a New Item in Jenkins, choose a freestyle project.
9. Choose this GitHub repository in Source Code
Management.
https://github.com/shazforiot/MSBuild_firstproject.git



10. Under Build-> Execute SonarQube Scanner, enter these Analysis properties. Mention the SonarQube Project Key, Login, Password, Source path and Host URL.
11. Go to http://localhost:9000/<user_name>/permissions and allow Execute Permissions to the Admin user.

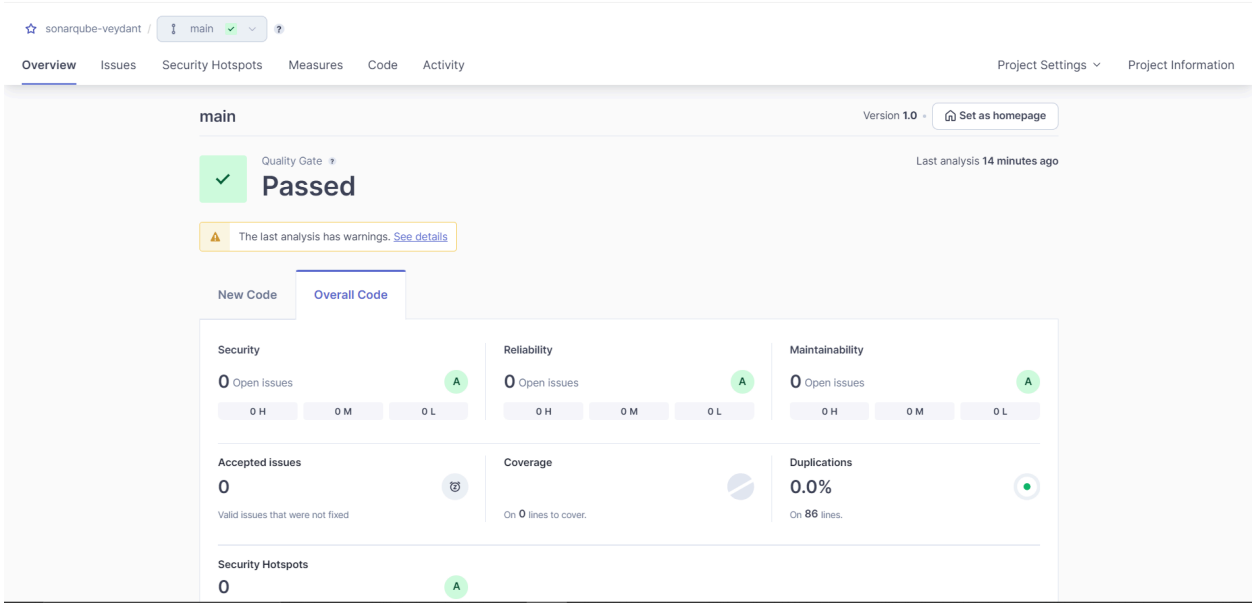
12. Run the build and check the output

Console Output

Download Copy View as plain text

```
Started by user unknown or anonymous
Running as SYSTEM
Building on the built-in node in workspace C:\ProgramData\Jenkins\.jenkins\workspace\sonarqube-test
The recommended git tool is: NONE
No credentials specified
 > git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\sonarqube-test\.git # timeout=10
Fetching changes from the remote Git repository
 > git.exe config remote.origin.url https://github.com/shazforiot/MSBuild_firstproject # timeout=10
Fetching upstream changes from https://github.com/shazforiot/MSBuild_firstproject
 > git.exe --version # timeout=10
 > git --version # 'git version 2.45.2.windows.1'
 > git.exe fetch --tags --force --progress -- https://github.com/shazforiot/MSBuild_firstproject +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision f2bc042c04c6e72427c380bcaee6d6fee7b49adf (refs/remotes/origin/master)
 > git.exe config core.sparsecheckout # timeout=10
 > git.exe checkout -f f2bc042c04c6e72427c380bcaee6d6fee7b49adf # timeout=10
Commit message: "updated"
 > git.exe rev-list --no-walk f2bc042c04c6e72427c380bcaee6d6fee7b49adf # timeout=10
[sonarqube-test] $ C:\ProgramData\Jenkins\.jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\sonarqube\bin\sonar-scanner.bat -Dsonar.host.url=http://localhost:9000
-Dsonar.projectKey=sonarqube-veydant -Dsonar.projectName=sonarqube-veydant -Dsonar.host.url=http://localhost:9000 -Dsonar.login=squ_3c67bbd5196ad7f467c5a12b65dfb090e8769e1c
-Dsonar.projectVersion=1.0 -Dsonar.sources=. -Dsonar.projectBaseDir=C:\ProgramData\Jenkins\.jenkins\workspace\sonarqube-test
21:42:11.084 WARN  Property 'sonar.host.url' with value 'http://localhost:9000' is overridden with value 'http://localhost:9000'
21:42:11.111 INFO  Scanner configuration file: C:\ProgramData\Jenkins\.jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\sonarqube\bin\..\conf\sonar-
scanner.properties
21:42:11.114 INFO  Project root configuration file: NONE
21:42:11.182 INFO  SonarScanner CLI 6.2.0.4584
21:42:11.186 INFO  Java 21.0.4 Eclipse Adoptium (64-bit)
21:42:11.195 INFO  Windows 10 10.0 amd64
21:42:11.257 INFO  User cache: C:\Windows\system32\config\systemprofile\.sonar\cache
21:42:12.960 INFO  JRE provisioning: os[windows], arch[amd64]
21:42:25.421 INFO  Communicating with SonarQube Server 10.6.0.92116
21:42:27.194 INFO  Starting SonarScanner Engine...
21:42:27.196 INFO  Java 17.0.11 Eclipse Adoptium (64-bit)
```

13. Once the build is complete check on sonarqube



Conclusion

- In this experiment we worked on the sonarqube project along with jenkins.
- Project Build step issues: Issues faced were due to permissions from sonarqube project.
- The steps involved logging into SonarQube, creating a project, and configuring necessary settings within Jenkins to facilitate automated analysis of our sample GitHub repository. This integration not only enhances our ability to identify vulnerabilities early in the development lifecycle but also promotes a culture of security within our development practices.
- By integrating Jenkins with SonarQube, we established an automated framework for continuous static analysis, enhancing our CI/CD pipeline's security posture.

Veydant Sharma D15C 50 AdvDevops

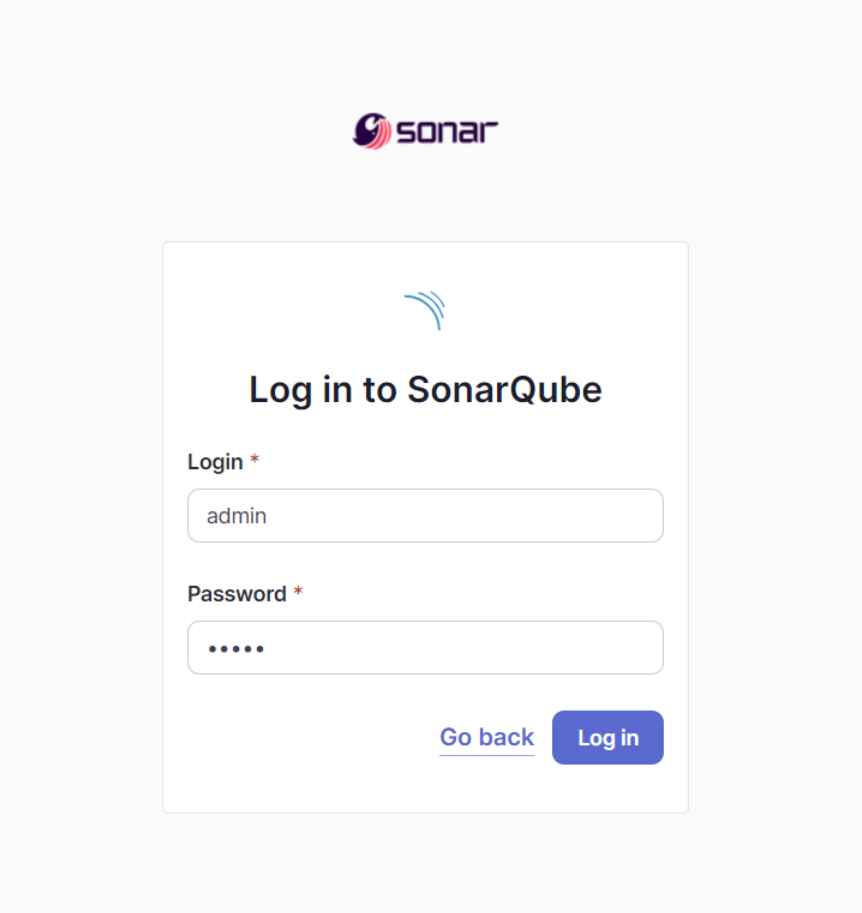
Experiment 8

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

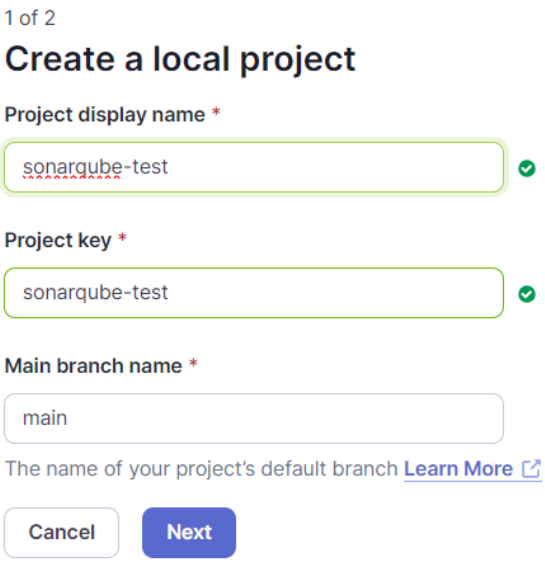
2. Run SonarQube in a Docker container using this command -

```
C:\Windows\system32>docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:late
st
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
b6dd73afc810a20ec3d643e9a148ab9643a3b5beff2766406df21f5f54a090c1
```

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

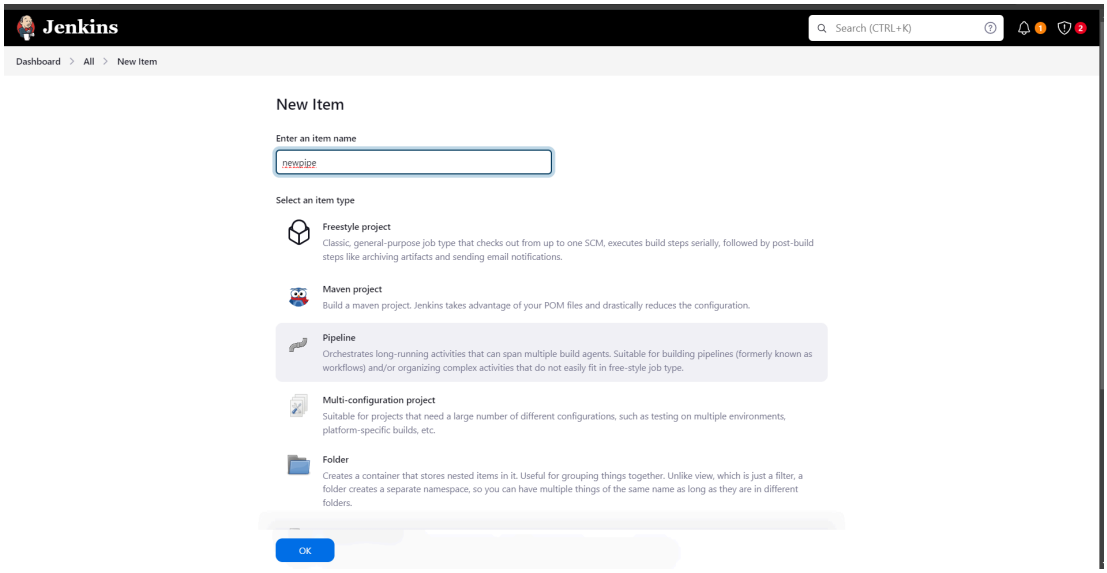


4. Create a new project



5. Click on Security and create a new name and token, select type as user token

6. Create a new pipeline



7. Go to sonarqube website and download the SonarScanner CLI

SonarScanner CLI

SonarScanner | Issue Tracker Show more

6.2 2024-09-17

Support PKCS12 truststore generated with OpenSSL

Download scanner for: Linux x64 Linux AArch64 Windows x64 macOS x64 macOS AArch64 Docker

Any (Requires a pre-installed JVM)

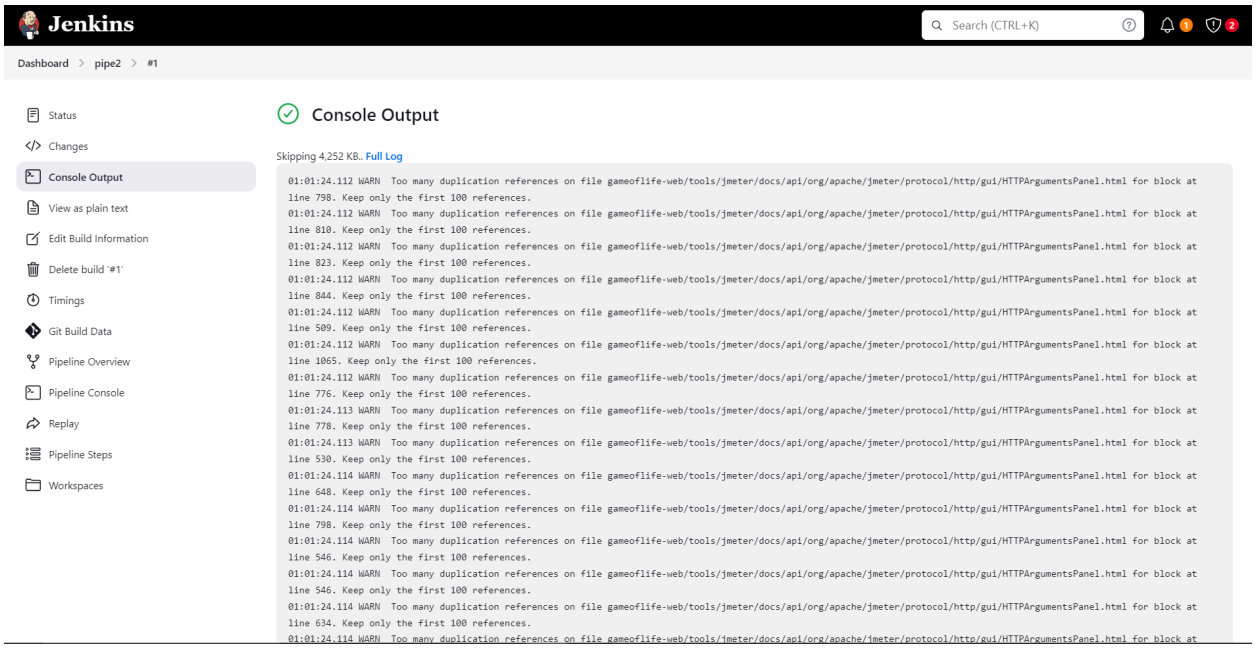
Release notes

Once it is downloaded, copy its path and add to the script

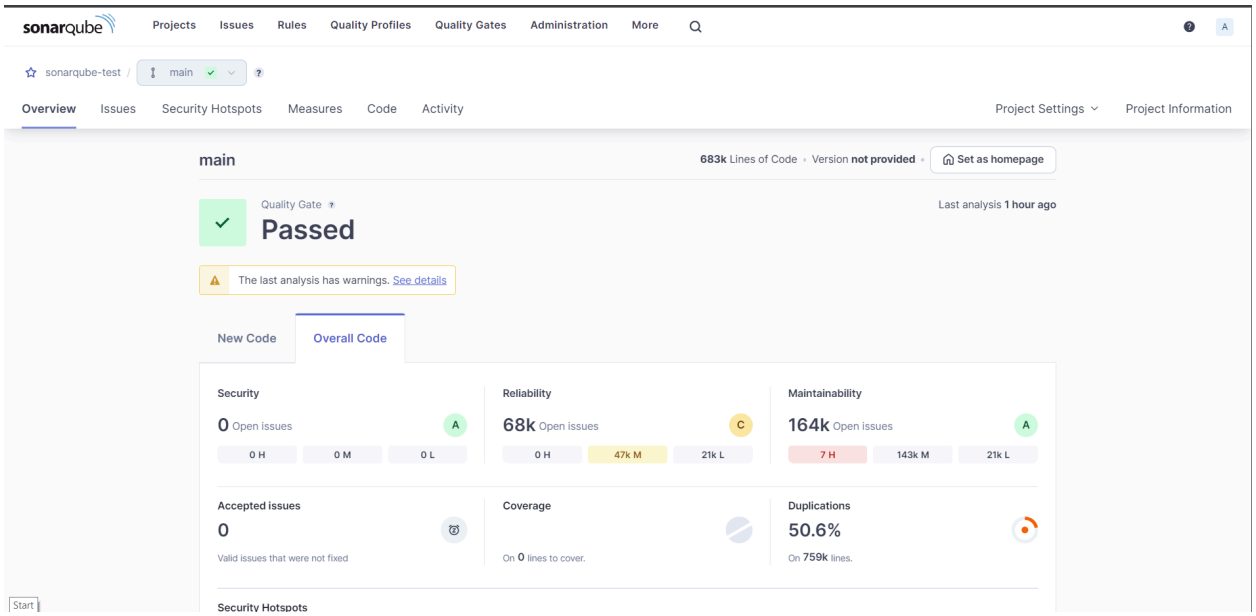
8. Inside the pipeline section add the following script

```
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/"
}
}
}
```

9. Save and build the pipeline, this step may take a while. After the build is successful, check the console output



10. Check the sonarqube project, whether it has passed or not.



Veydant Sharma D15C 50 AdvDevops

Conclusion:

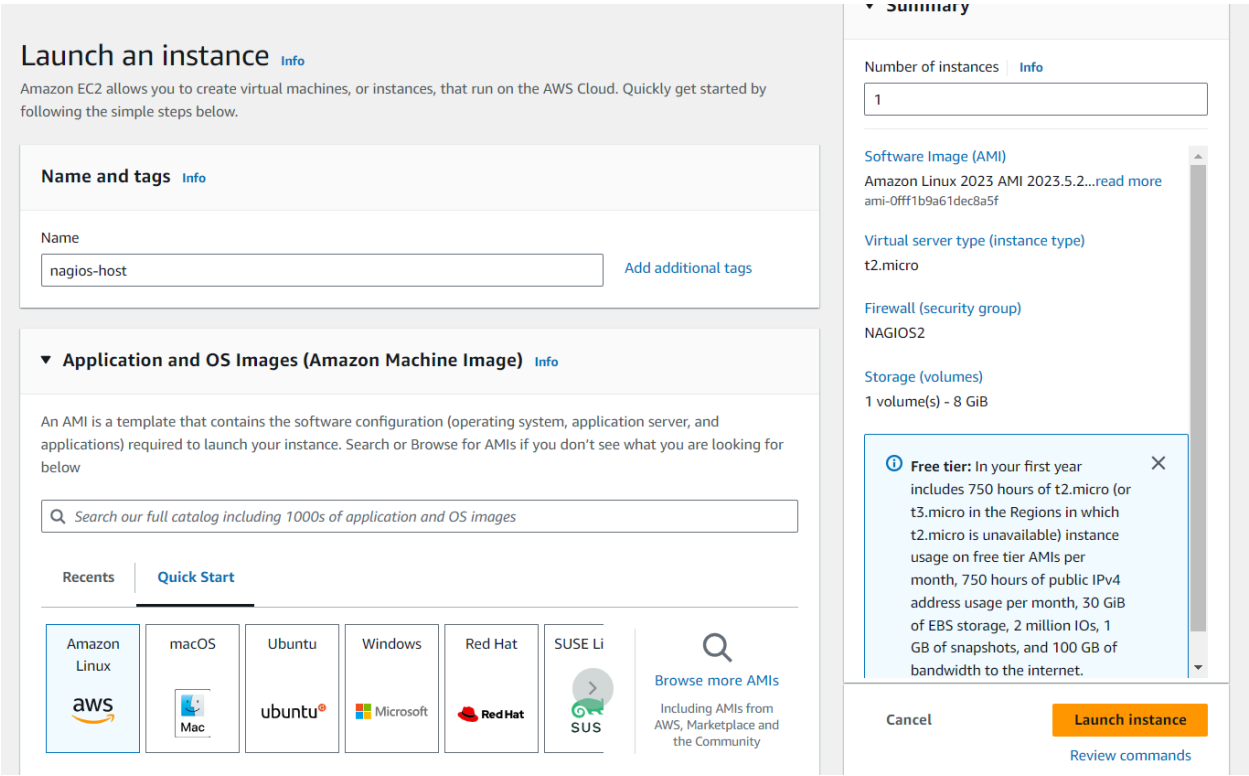
- We created a sonarqube project locally
- We created a jenkins pipeline and added a script with github link
- We also added sonarqube project details like key, token etc.
- Many issues were faced with the script as the build was initially failing, later succeeded with some changes.
- The build was successful on jenkins and passed on sonarqube

Veydant Sharma D15C 50 AdvDevop

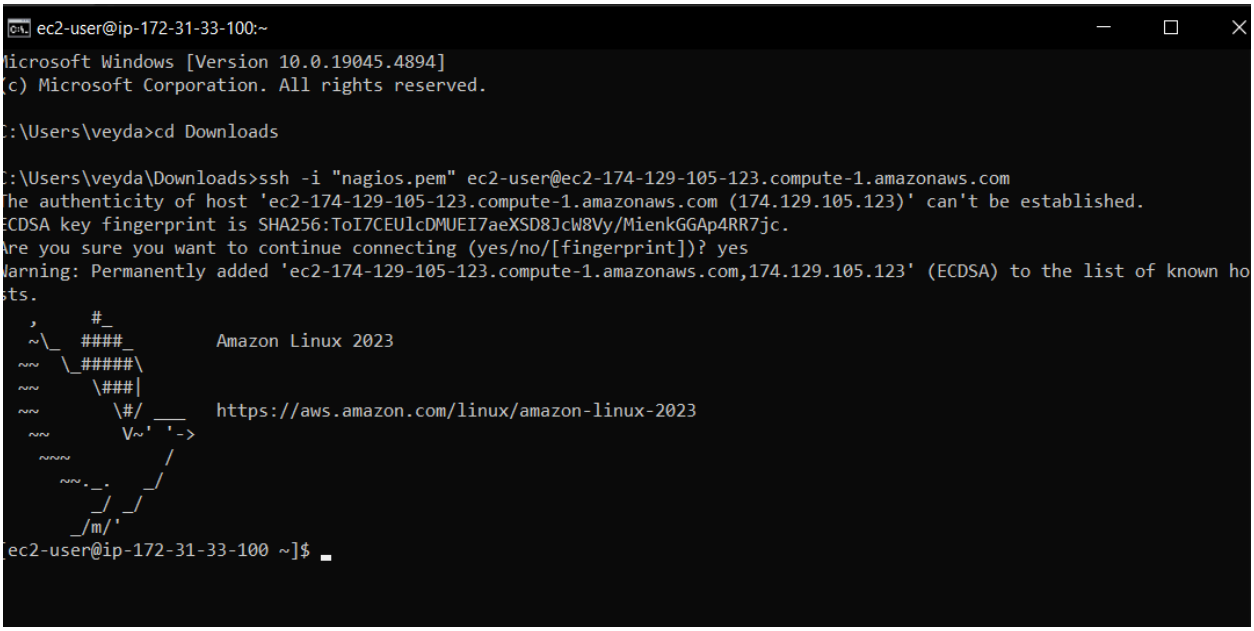
Experiment 9

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

Step 1: Create an ec2 instance, select amazon linux, t2.micro , select a key name and launch instance.



Step 2 : Connect to the instance using the SSH client, and copy and paste the command on the terminal



Step 3: First, run the following command:- sudo yum update This command will check for any updates for the YUM library.

```
[ec2-user@ip-172-31-35-21 ~]$ sudo yum update
Last metadata expiration check: 0:02:03 ago on Tue Oct  8 05:47:23 2024.
Dependencies resolved.
Nothing to do.
Complete!
```

Step 4: Run the command: sudo yum install httpd php This installs an Apache server and a PHP on your instance.

```
[ec2-user@ip-172-31-35-21 ~]$ sudo yum install httpd php
Last metadata expiration check: 0:02:58 ago on Tue Oct  8 05:47:23 2024.
Dependencies resolved.
================================================================================
 Package                 Arch      Version                         Repository      Size
================================================================================
Installing:
 httpd                   x86_64    2.4.62-1.amzn2023               amazonlinux     48 k
 php8.3                  x86_64    8.3.10-1.amzn2023.0.1           amazonlinux     10 k
Installing dependencies:
 apr                     x86_64    1.7.2-2.amzn2023.0.2            amazonlinux    129 k
 apr-util                x86_64    1.6.3-1.amzn2023.0.1            amazonlinux     98 k
 generic-logos-httpd     noarch    18.0.0-12.amzn2023.0.3          amazonlinux     19 k
 httpd-core              x86_64    2.4.62-1.amzn2023               amazonlinux    1.4 M
 httpd-filesystem        noarch    2.4.62-1.amzn2023               amazonlinux     14 k
```

Step 5: Run the command: sudo yum install gcc glibc glibc-common This installs the C/C++ compiler (GCC) along with the necessary C libraries required for compiling and running C programs.

```
[ec2-user@ip-172-31-35-21 ~]$ sudo yum install gcc glibc glibc-common
Last metadata expiration check: 0:03:21 ago on Tue Oct  8 05:47:23 2024.
Package glibc-2.34-52.amzn2023.0.11.x86_64 is already installed.
Package glibc-common-2.34-52.amzn2023.0.11.x86_64 is already installed.
Dependencies resolved.
================================================================================
 Package               Arch      Version                       Repository      Size
================================================================================
Installing:
 gcc                   x86_64    11.4.1-2.amzn2023.0.2         amazonlinux     32 M
Installing dependencies:
 annobin-docs          noarch    10.93-1.amzn2023.0.1          amazonlinux     92 k
 annobin-plugin-gcc    x86_64    10.93-1.amzn2023.0.1          amazonlinux    887 k
 cpp                   x86_64    11.4.1-2.amzn2023.0.2         amazonlinux     10 M
 gc                    x86_64    8.0.4-5.amzn2023.0.2          amazonlinux    105 k
```

Step 6: Run the command: sudo yum install gd gd-devel

```
[ec2-user@ip-172-31-35-21 ~]$ sudo yum install gd gd-devel
Last metadata expiration check: 0:03:44 ago on Tue Oct  8 05:47:23 2024.
Dependencies resolved.
================================================================================
 Package                   Arch     Version                     Repository     Size
================================================================================
Installing:
 gd                        x86_64   2.3.3-5.amzn2023.0.3        amazonlinux   139 k
 gd-devel                  x86_64   2.3.3-5.amzn2023.0.3        amazonlinux    38 k
Installing dependencies:
 brotli                    x86_64   1.0.9-4.amzn2023.0.2        amazonlinux   314 k
 brotli-devel              x86_64   1.0.9-4.amzn2023.0.2        amazonlinux    31 k
 bzip2-devel               x86_64   1.0.8-6.amzn2023.0.2        amazonlinux   214 k
 cairo                     x86_64   1.17.6-2.amzn2023.0.1       amazonlinux   684 k
 cmake-filesystem          x86_64   3.22.2-1.amzn2023.0.4       amazonlinux    16 k
```

Step 7: Run the commands: sudo adduser -m nagios sudo passwd nagios

```
[ec2-user@ip-172-31-35-21 ~]$ sudo adduser -m nagios
[ec2-user@ip-172-31-35-21 ~]$ sudo passwd -m nagios
passwd: bad argument -m: unknown option
[ec2-user@ip-172-31-35-21 ~]$ sudo passwd nagios
Changing password for user nagios.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
```

```
[ec2-user@ip-172-31-35-21 ~]$ sudo usermod -a -G nagcmd nagios
[ec2-user@ip-172-31-35-21 ~]$ sudo usermod -a -G nagcmd apache
```

Step 10: mkdir ~/downloads cd ~/downloads This creates a directory named ‘downloads’, to store the files of the nagios server that are downloaded

```
[ec2-user@ip-172-31-35-21 ~]$ : mkdir ~/downloads
[ec2-user@ip-172-31-35-21 ~]$ cd ~/downloads
```

Step 11: wget https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.5.tar.gz The above command installs the latest version of nagios-core
Step 12: wget https://nagios-plugins.org/download/nagios-plugins-2.4.11.tar.gz The above command installs the latest version of nagios-plugins.

```
[ec2-user@ip-172-31-35-21 downloads]$ wget https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.5.tar.g
--2024-10-09 05:42:49--  https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.5.tar.gz
Resolving assets.nagios.com (assets.nagios.com)... 45.79.49.120, 2600:3c00::f03c:92ff:fef7:45ce
Connecting to assets.nagios.com (assets.nagios.com)|45.79.49.120|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2065473 (2.0M) [application/x-gzip]
Saving to: ‘nagios-4.5.5.tar.gz.1’

nagios-4.5.5.tar.gz.1        100%[=============================================>]   1.97M  5.56MB/s    in 0.4s

2024-10-09 05:42:50 (5.56 MB/s) - ‘nagios-4.5.5.tar.gz.1’ saved [2065473/2065473]

[ec2-user@ip-172-31-35-21 downloads]$ wget https://nagios-plugins.org/download/nagios-plugins-2.4.11.tar.gz
--2024-10-09 05:43:04--  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.1’

nagios-plugins-2.4.11.tar.gz. 100%[=============================================>]   2.62M  5.49MB/s    in 0.5s

2024-10-09 05:43:05 (5.49 MB/s) - ‘nagios-plugins-2.4.11.tar.gz.1’ saved [2753049/2753049]
```

Step 13:
tar zxvf nagios-4.5.5.tar.gz This extracts the nagios-core files into the same directory using the tar command.
Use ‘ls’ command to find the correct directory
run the ‘sudo yum install openssl-devel’ command.

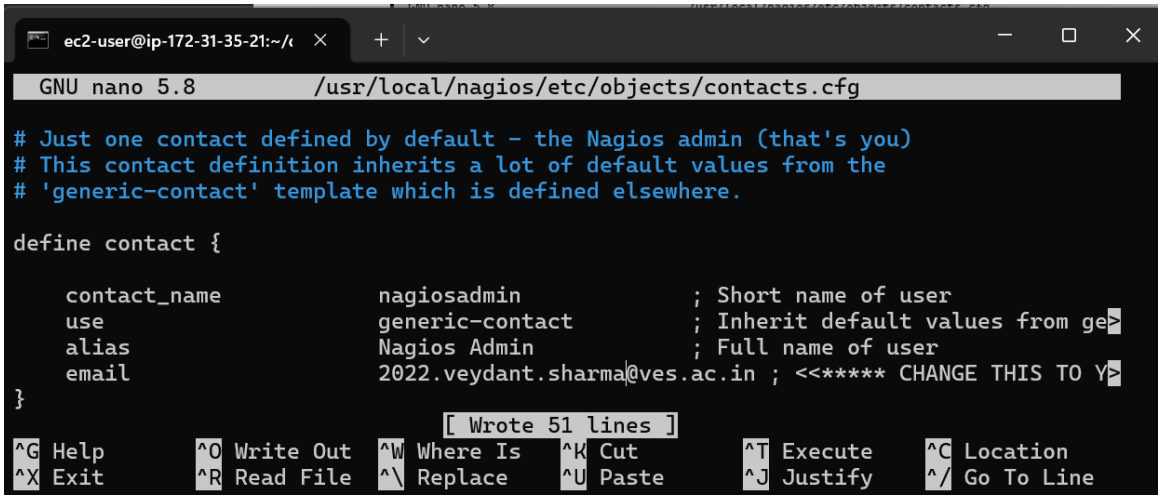
```
[ec2-user@ip-172-31-35-21 downloads]$ ls
nagios-4.5.5  nagios-4.5.5.tar.gz  nagios-4.5.5.tar.gz.1  nagios-plugins-2.4.11.tar.gz
[ec2-user@ip-172-31-35-21 downloads]$ cd nagios-4.5.5
[ec2-user@ip-172-31-35-21 nagios-4.5.5]$ sudo yum install openssl-devel
Last metadata expiration check: 23:58:21 ago on Tue Oct  8 05:47:23 2024.
Package openssl-devel-1:3.0.8-1.amzn2023.0.14.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-35-21 nagios-4.5.5]$ ./configure --with-command-group=nagcmd
```

Step 14: Next, we must compile all components of this software according to the instructions in the Makefile. To do so, use the following command: make all Then, sudo make install sudo make install-init sudo make install-config sudo make install-commandmode

```
[ec2-user@ip-172-31-35-21 nagios-4.5.5]$ make all
cd ./base && make
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/base'
make -C ../lib
```

```
[ec2-user@ip-172-31-35-21 nagios-4.5.5]$ sudo make install
sudo make install-init
sudo make install-config
sudo make install-commandmode
cd ./base && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/base'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagios /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagiostats /usr/local/nagios/bin
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5/base'
```

Step 15: : We need to update the email linked with this server to our email for it to send notifications (if any needed). sudo nano /usr/local/nagios/etc/objects/contacts.cfg



Step 17: sudo make install-webconf This installs the necessary configuration files for the Nagios web interface.

Step 18: sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin This creates a user named ‘nagiosadmin’ to access the nagios web interface. Create a password and keep it in mind as it will be required in the future steps.

Step 19: Restart the apache server to apply all the recent configurations. sudo service httpd restart
Step 20: cd ~/downloads tar zxvf nagios-plugins-2.4.11.tar.gz This changes the directory to the ‘downloads’ directory and extracts the files for nagios-plugins

```
[ec2-user@ip-172-31-35-21 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-172-31-35-21 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-172-31-35-21 nagios-4.5.5]$ sudo service httpd restart
Redirecting to /bin/systemctl restart httpd.service
[ec2-user@ip-172-31-35-21 nagios-4.5.5]$ cd ~/downloads
[ec2-user@ip-172-31-35-21 downloads]$ tar zxvf nagios-plugins-2.4.11.tar.gz
nagios-plugins-2.4.11/
nagios-plugins-2.4.11/build-aux/
```

Step 21: cd nagios-plugins-2.4.11 ./configure --with-nagios-user=nagios --with-nagios-group=nagios This installs the configurations for the nagios-plugins files.

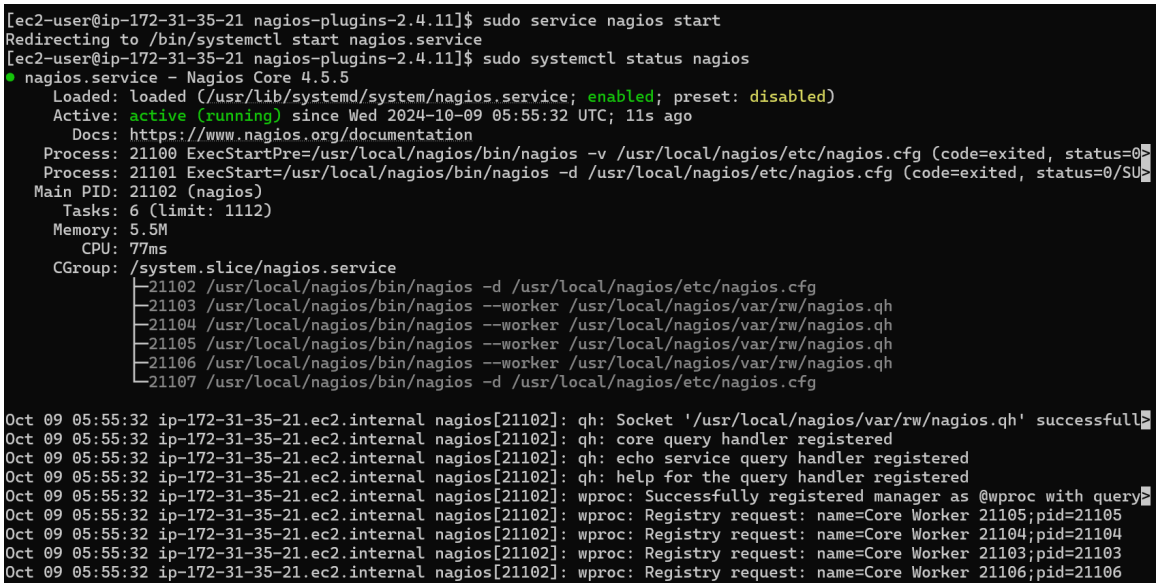
```
[ec2-user@ip-172-31-35-21 downloads]$ cd nagios-plugins-2.4.11
[ec2-user@ip-172-31-35-21 nagios-plugins-2.4.11]$ ./configure --with-nagios-user=nagios --with-nagios-group=nagios
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /usr/bin/mkdir -p
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking whether make supports nested variables... yes
checking whether to enable maintainer-specific portions of Makefiles... yes
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
```

Step22:: sudo chkconfig --add nagios
sudo chkconfig nagios on

```
[ec2-user@ip-172-31-35-21 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-172-31-35-21 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...
```

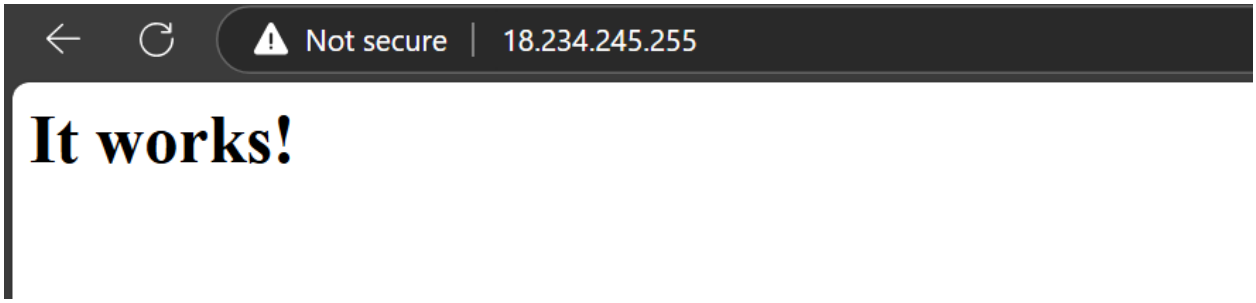
```
[ec2-user@ip-172-31-35-21 downloads]$ tar zxvf nagios-4.5.5.tar.gz
nagios-4.5.5/
nagios-4.5.5/.github/
nagios-4.5.5/.github/workflows/
nagios-4.5.5/.github/workflows/test.yml
nagios-4.5.5/.gitignore
nagios-4.5.5/CONTRIBUTING.md
```

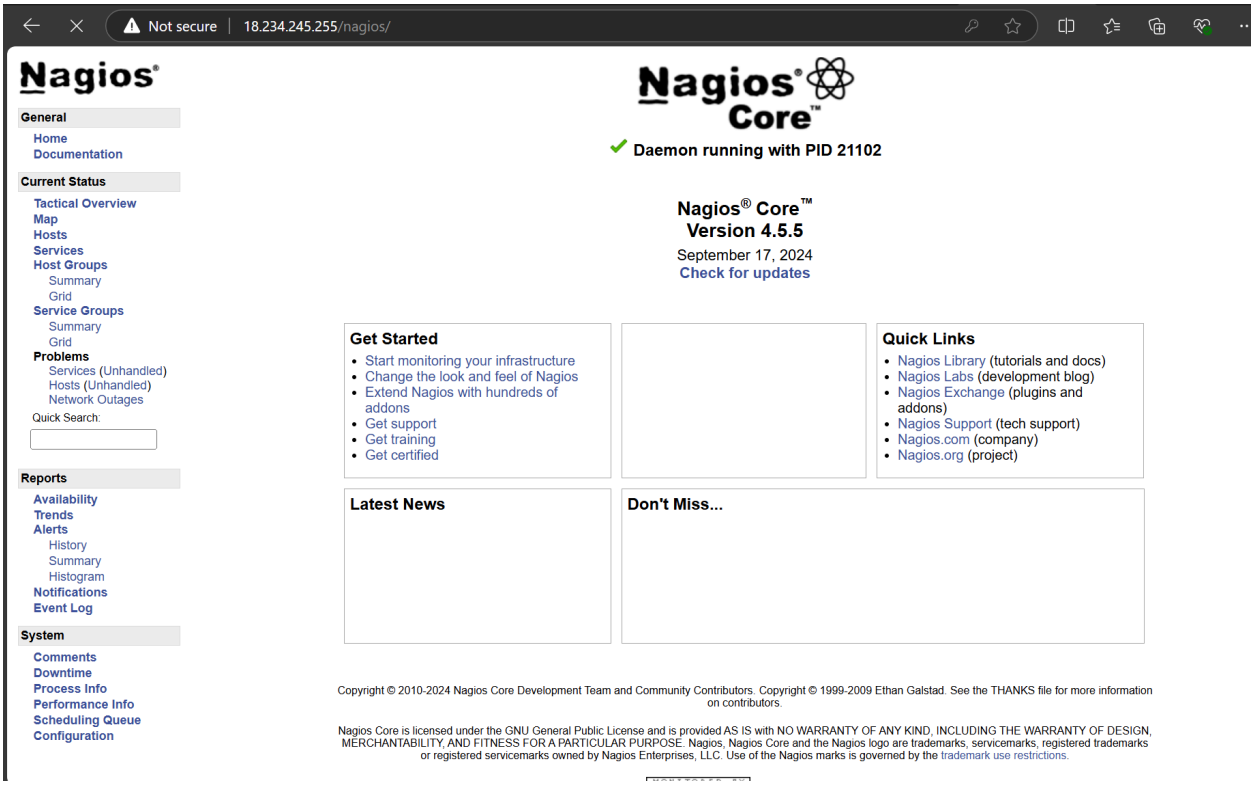
Step 23: sudo service nagios start This starts the Nagios service.
sudo systemctl status nagios This checks the status of Nagios. Ensure that it is ‘active(running)’.



Veydant Sharma D15C 50 AdvDevop

Step 24: In the address bar, enter ‘http://<ipv4 address>/nagios’.





Veydant Sharma D15C 50 AdvDevop

Conclusion:

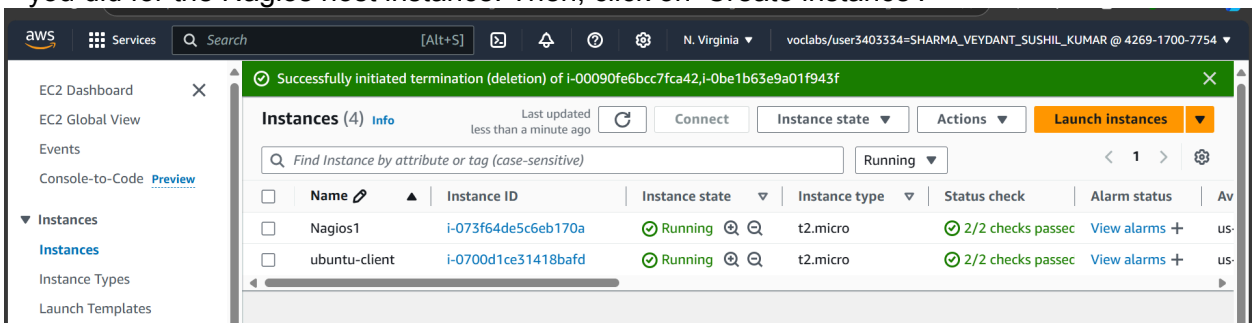
- In the above experiment, we learned how to install and configure Nagios Core, Nagios Plugins and NRPE (Nagios Remote Plugin Executor) on Linux Machine.
- We faced some issues with the configuration command, which were fixed with the right directory and the sudo yum install openssl-devel' command.
- Once the setup was complete, we hosted the Nagios server and accessed the Nagios dashboard by pasting the public IPv4 address of our instance in the browser

Veydant Sharma D15C 50 AdvDevop Experiment 10

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

Step 1: Navigate to EC2 on the AWS console using the ‘Services’ section and click on ‘Create instance’. Give your instance a name and choose ‘Ubuntu’ as the instance type.

Ensure that you choose the same key pair and security group for the Ubuntu client instance as you did for the Nagios host instance. Then, click on ‘Create instance’.



Step 2: Click on the instance ID of your nagios-server instance and click on ‘Connect’. Then, click on ‘SSH client’ and copy the command under ‘Example’. Then, open the terminal in the folder where the .pem file for your instance’s key pair is located and paste the SSH command that you just copied. This connects your instance to your local terminal using SSH.

Step 3: ps -ef | grep nagios Run the above command on the nagios-host instance. This verifies whether the nagios service is running or not.

```
[ec2-user@ip-172-31-35-21 ~]$ ps -ef | grep nagios
nagios     21102       1  0 05:55 ?        00:00:00 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
nagios     21103   21102  0 05:55 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagi
s.qh
nagios     21104   21102  0 05:55 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagi
s.qh
nagios     21105   21102  0 05:55 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagi
s.qh
nagios     21106   21102  0 05:55 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagi
s.qh
nagios     21107   21102  0 05:55 ?        00:00:00 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
root       21110    2900  0 05:55 pts/0    00:00:00 sudo systemctl status nagios
root       21112   21110  0 05:55 pts/1    00:00:00 sudo systemctl status nagios
root       21113   21112  0 05:55 pts/1    00:00:00 systemctl status nagios
ec2-user   22304   22272  0 06:15 pts/2    00:00:00 grep --color=auto nagios
```

Step 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-172-31-35-21 ~]$ sudo su
mkdir -p /usr/local/nagios/etc/objects/monitorhosts
mkdir -p /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
[root@ip-172-31-35-21 ec2-user]#
```

Step 5: We need to create a config file in this folder. So, copy the contents of the existing localhost config to the new file ‘linuxserver.cfg’. cp /usr/local/nagios/etc/objects/localhost.cfg /usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg

```
root@ip-172-31-35-21 ec2-user]# sudo mkdir -p /usr/local/nagios/etc/objects/monitorhosts/linuxhosts/
root@ip-172-31-35-21 ec2-user]# sudo cp /usr/local/nagios/etc/objects/localhost.cfg /usr/local/nagios/etc/objects/monit
rhosts/linuxhosts/linuxserver.cf
```

Step 6: We need to make some changes in this config file. Open it using nano editor:- nano /usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg 1. Change hostname and alias from ‘hostname’ to ‘linuxserver’. 2. Change address to the public ip address of the ubuntu-client instance.

Step 7: Once the files are verified and it is confirmed that there are no errors, we must restart the server. service nagios restart

```
[root@ip-172-31-35-21 ec2-user]# /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

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

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

Step 8: systemctl status nagios Using the above command, we check the status of the nagios server and ensure that it is active (running).

```
[root@ip-172-31-35-21 ec2-user]# service nagios restart
Redirecting to /bin/systemctl restart nagios.service
[root@ip-172-31-35-21 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 Wed 2024-10-09 06:32:00 UTC; 12s ago
       Docs: https://www.nagios.org/documentation
    Process: 23269 ExecStartPre=/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg (code=exited, status=0>
    Process: 23270 ExecStart=/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SU>
   Main PID: 23271 (nagios)
      Tasks: 6 (limit: 1112)
     Memory: 5.4M
        CPU: 69ms
     CGroup: /system.slice/nagios.service
             ├─23271 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─23272 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─23273 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─23274 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─23275 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             └─23276 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
```

Step 9: Connect your ubuntu-client instance to your local terminal using SSH in the same way as you connected the nagios-host instance to your local terminal using SSH

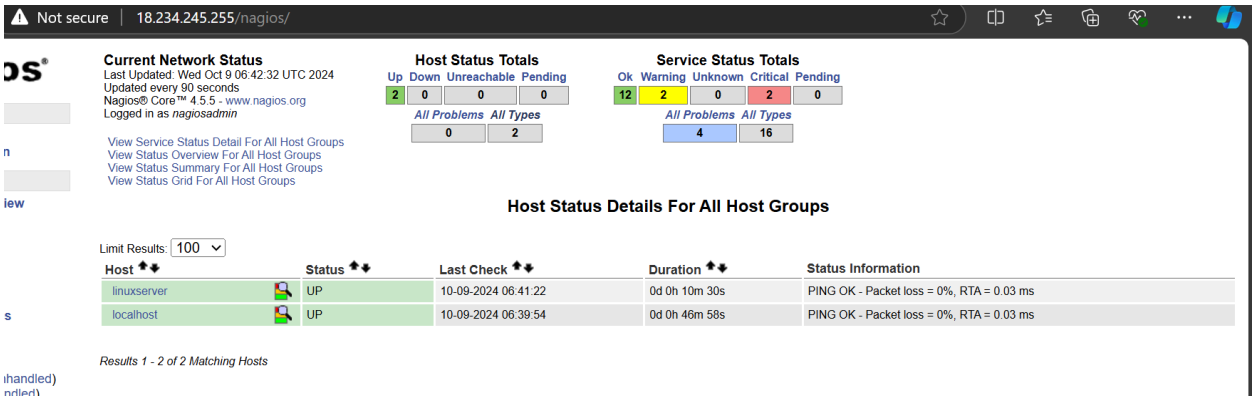
Step 10: On your ubuntu-client instance, run the following commands:- sudo apt update -y sudo apt install gcc -y sudo apt install -y nagios-nrpe-server nagios-plugins The above commands check for any new updates and then install gcc, Nagios NRPE server and Nagios plugins.

```
ubuntu@ip-172-31-94-199:~$ 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://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:7 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [380 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]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 c-n-f Metadata [8328 B]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [535 kB]
Get:15 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [82.9 kB]
```

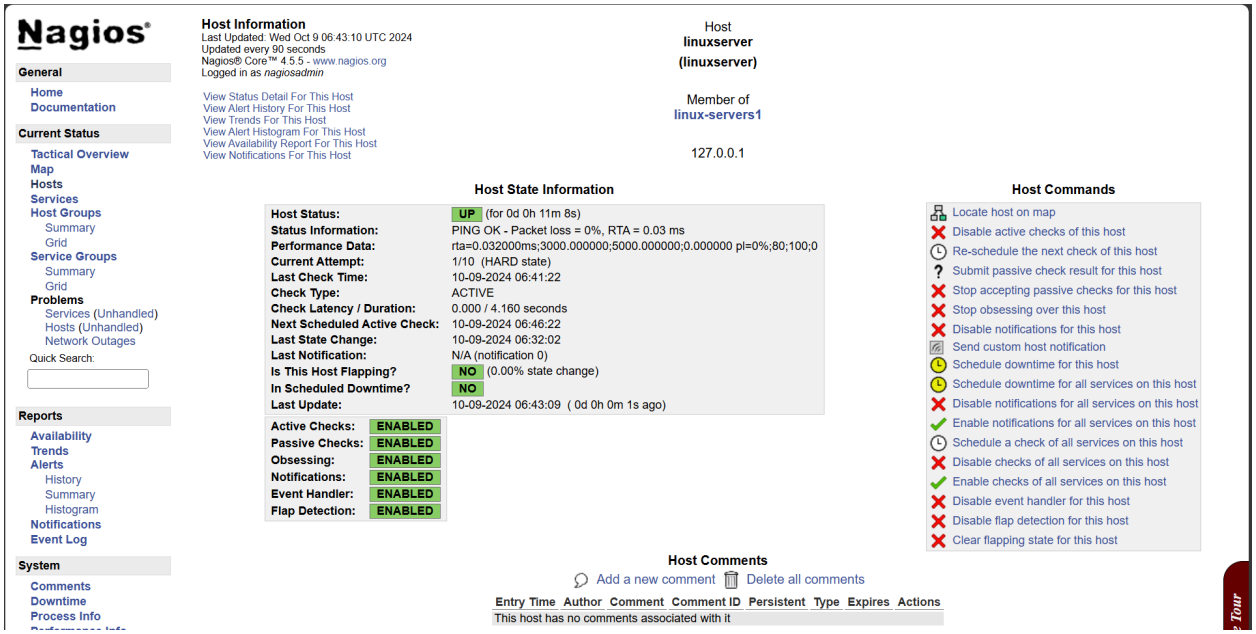
Step 11: Run the following command: sudo nano /etc/nagios/nrpe.cfg The above command opens the NRPE config file. Here, we need to add the public IP address of our host nagios-host instance to the NRPE configuration file. Under allowed_hosts, add the nagios-host public IPv4 address.

```
# and you do not want nrpe to bind on all interfaces.
# NOTE: This option is ignored if NRPE is running under either inetd or xinetd
18.234.245.255
#server_address=127.0.0.1



# LISTEN QUEUE SIZE
# Listen queue size (backlog) for serving incoming connections.
# You may want to increase this value under high load.
```

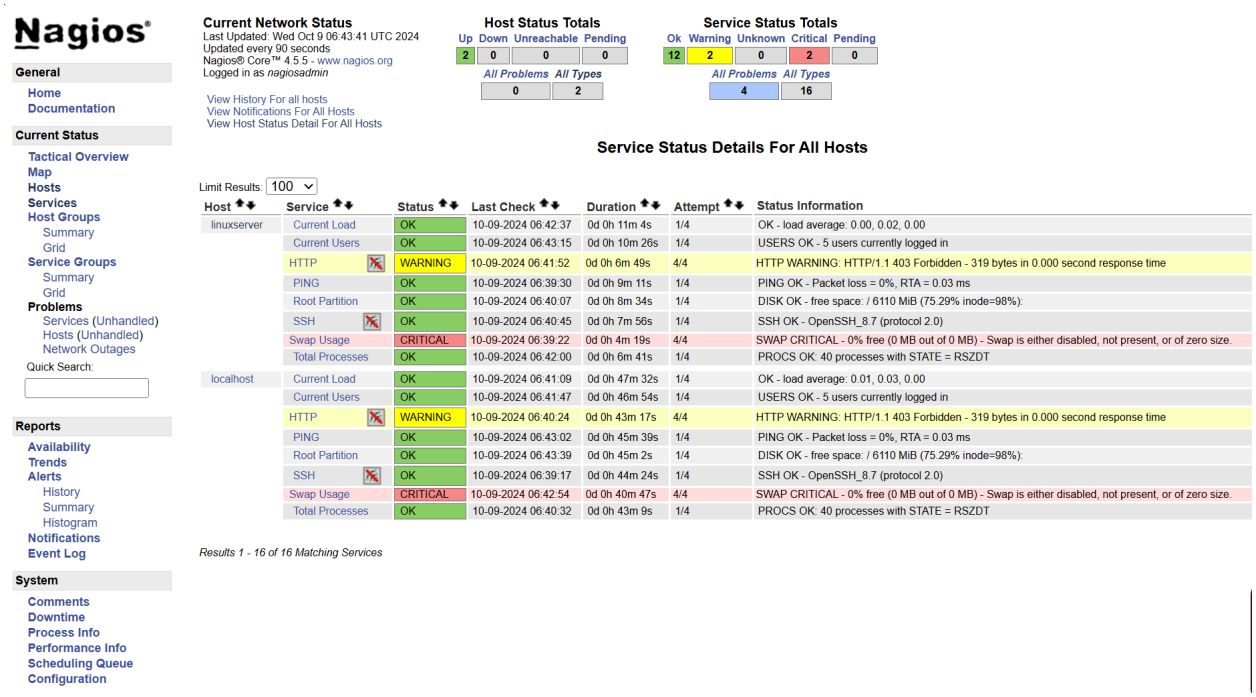
Step 14: Navigate to the Nagios dashboard. Click on ‘hosts’. We see that linuxserver has been added as a host.



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



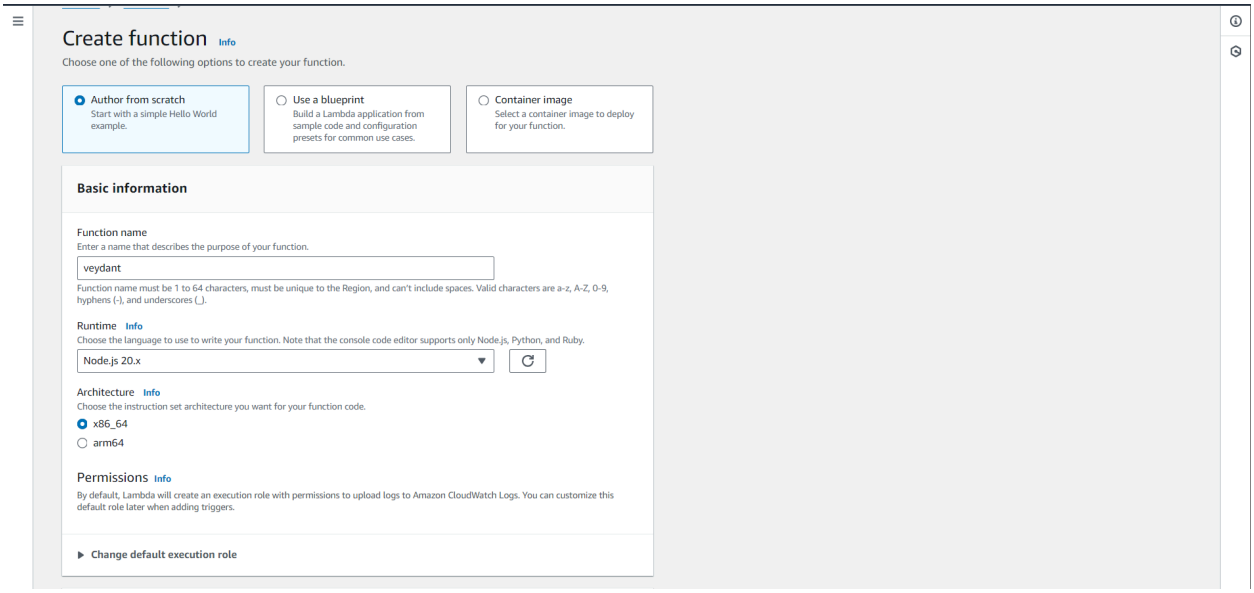
Conclusion:

- We have learnt how to perform port, service monitoring using nagios.
- We added the linuxserver as host for ubuntu client.
- After restarting the NRPE server, we can see the ‘linuxserver’ host added.

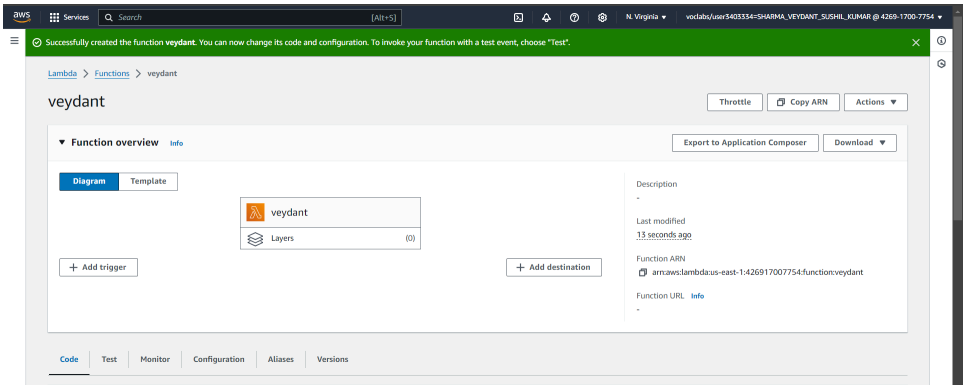
Veydant Sharma D15C 50 AdvDevop Experiment 11

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

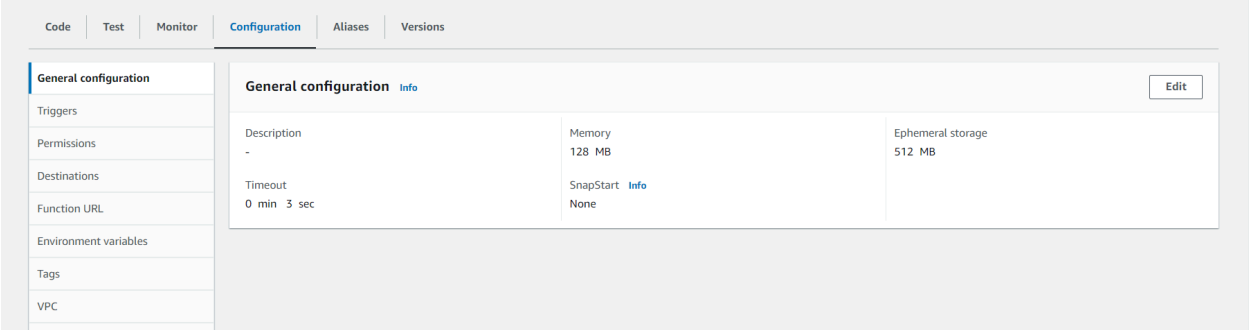
Step 1: On your AWS console, click on ‘Lambda’ in the services section and click on ‘Create function’.
Step 2: Give your Lambda function a name. Select the language to use to write your function (Node.js is the default and what we will use in this experiment). Keep other options as default.

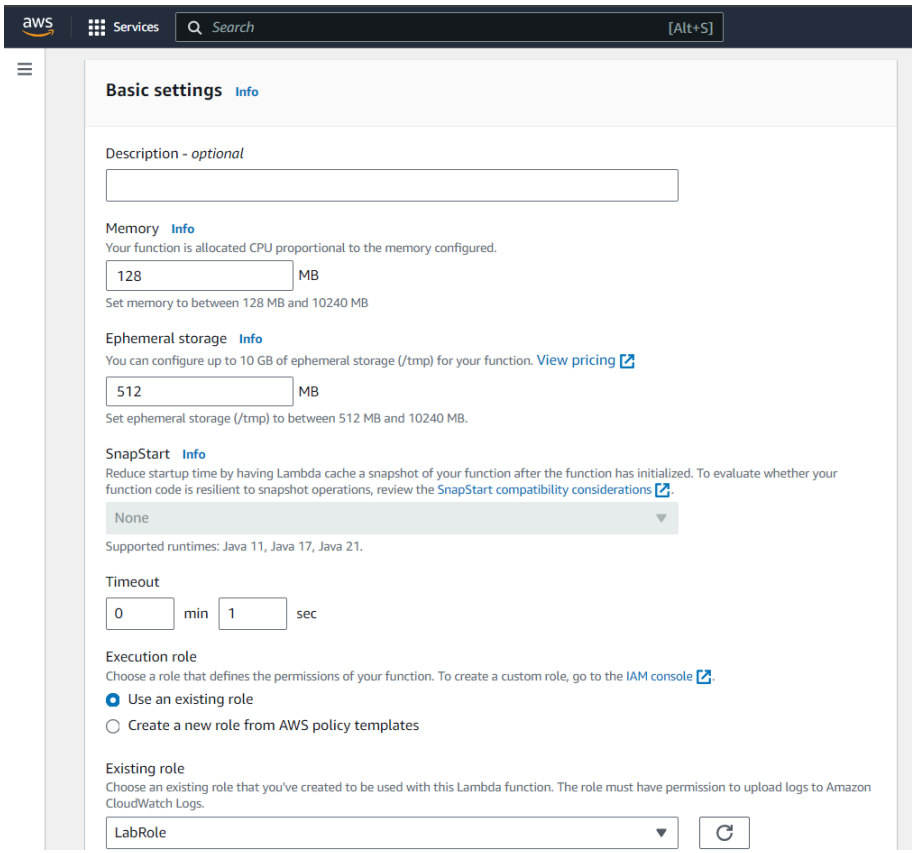


Your Lambda function gets created.

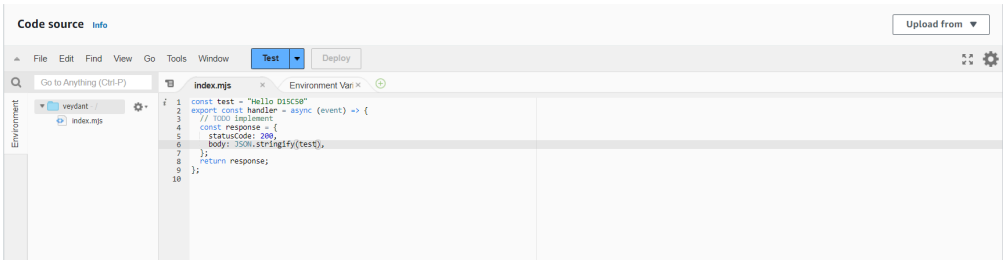


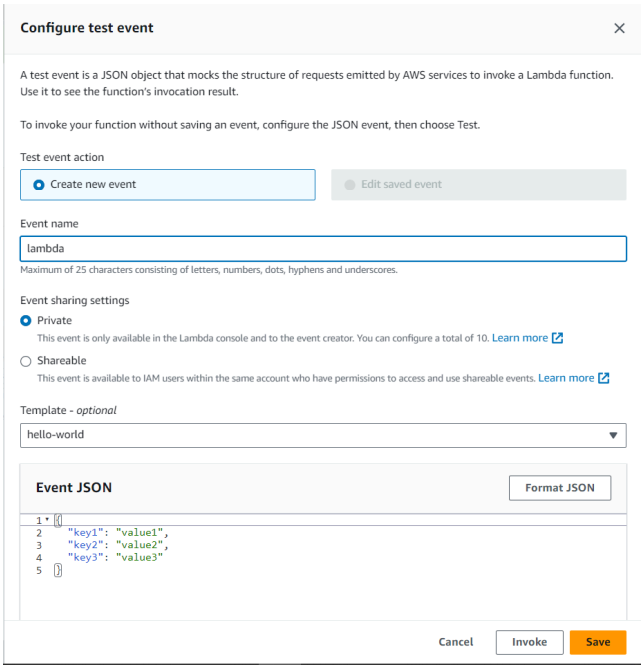
Step 3: The general configuration of the function is visible in the ‘Configuration’ tab. To change the configuration, click on ‘Edit’ Change the timeout to ‘1’. Then click on ‘Save’.



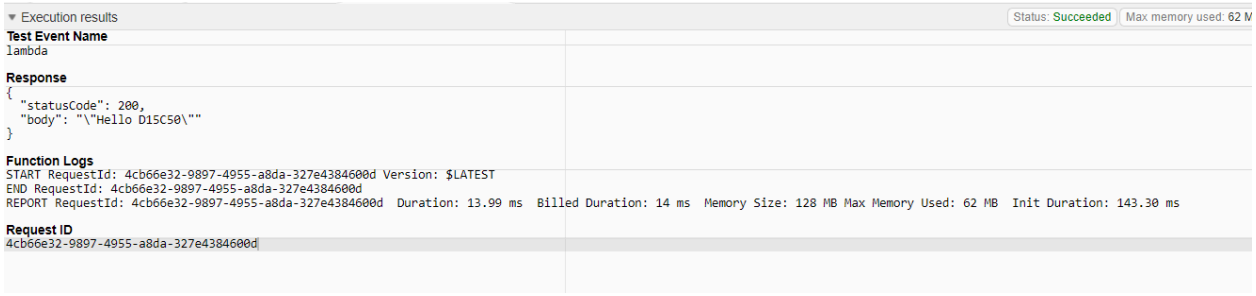


Step4: Navigate to code source and click on the arrow near test to create test event.
Step 5: Give your test event a name, keep all other options as default and click on ‘Save’





Step 6: Click on the 'Test' button. The following output appears.



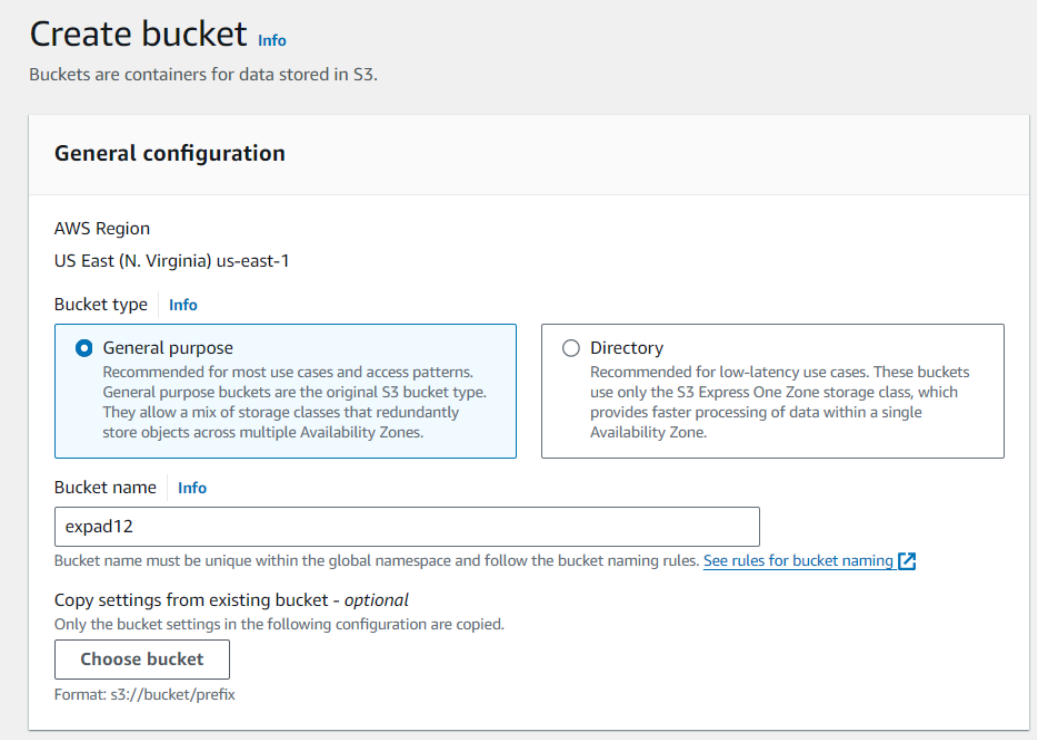
Conclusion:

- We created a lambda function and configured using Node.js
- We setup a test event for the function.
- From the output of the tests, we learn about the working of the Lambda function and how changes in its configuration affects its functionality and outputs.

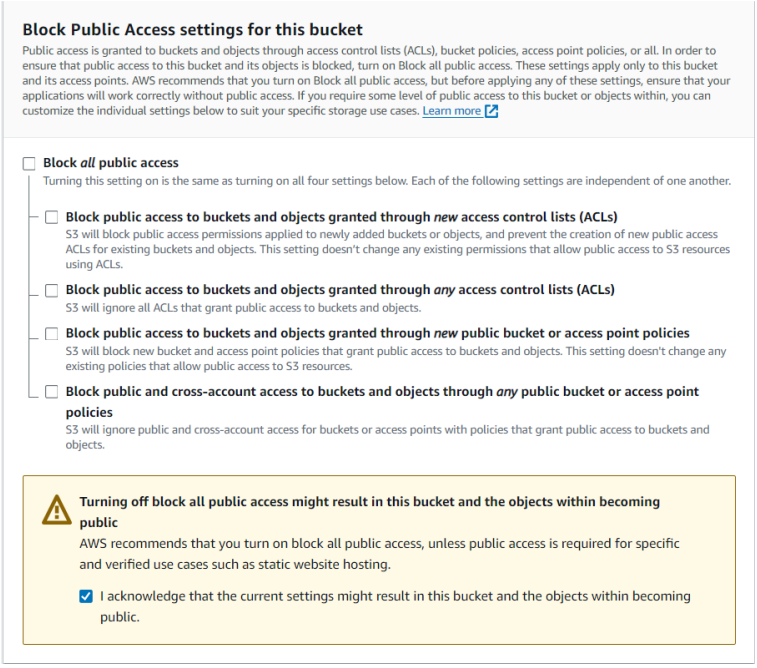
Veydant Sharma D15 C 50 AdvDevop Experiment 12

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.

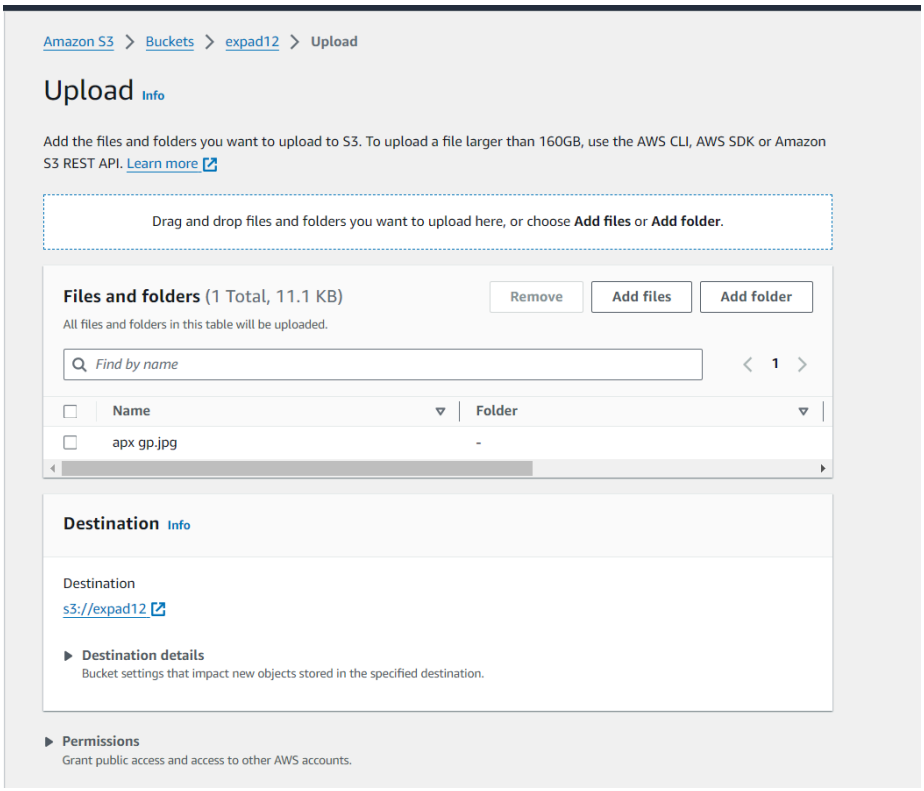
Step 1: On your AWS console, click on ‘S3’ in the services section and click on ‘Create bucket’. Give your bucket a name.



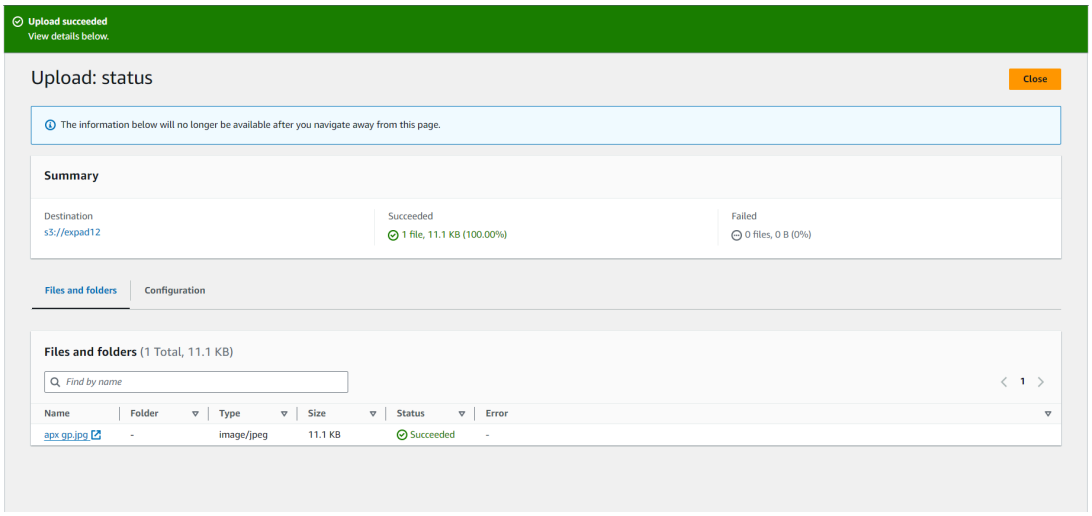
Uncheck the ‘Block all public access’ box.



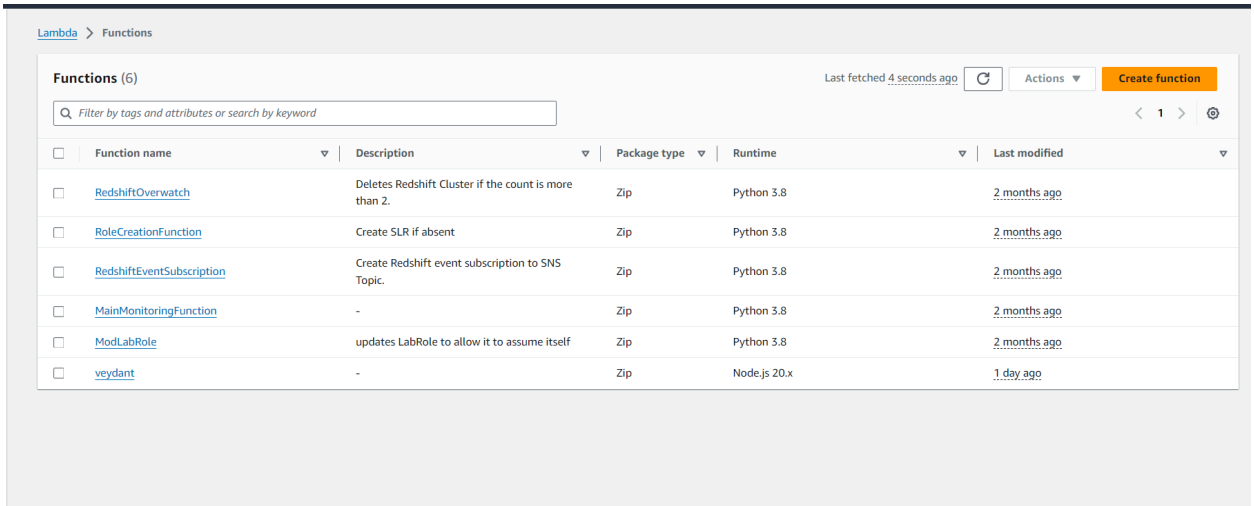
Step 2: Upload an image onto your S3 bucket by clicking on your S3 bucket, clicking on ‘Upload’, clicking on ‘Add files’, navigating to your image and selecting it.



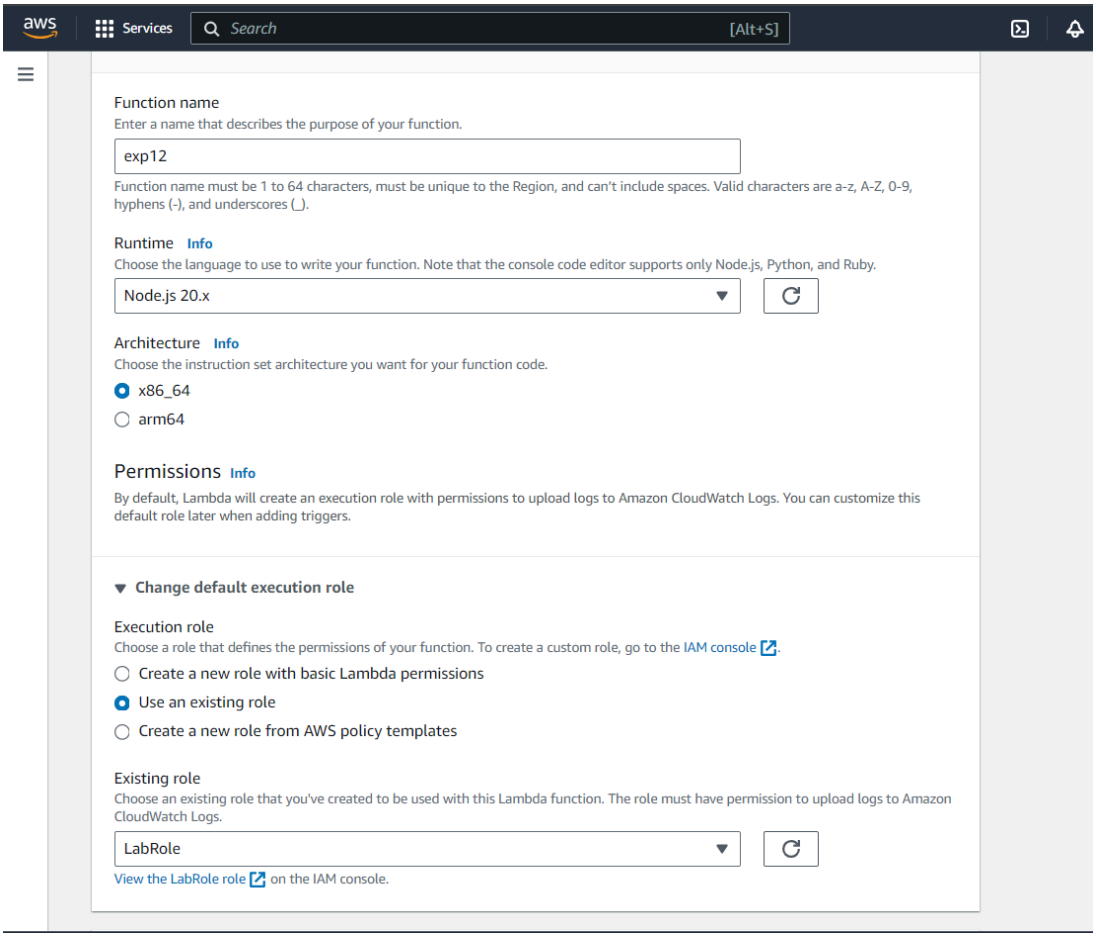
Your image gets uploaded onto the S3 bucket



Step 3: Navigate to the AWS Lambda console using the ‘Services’ section. Click on ‘Create function’.

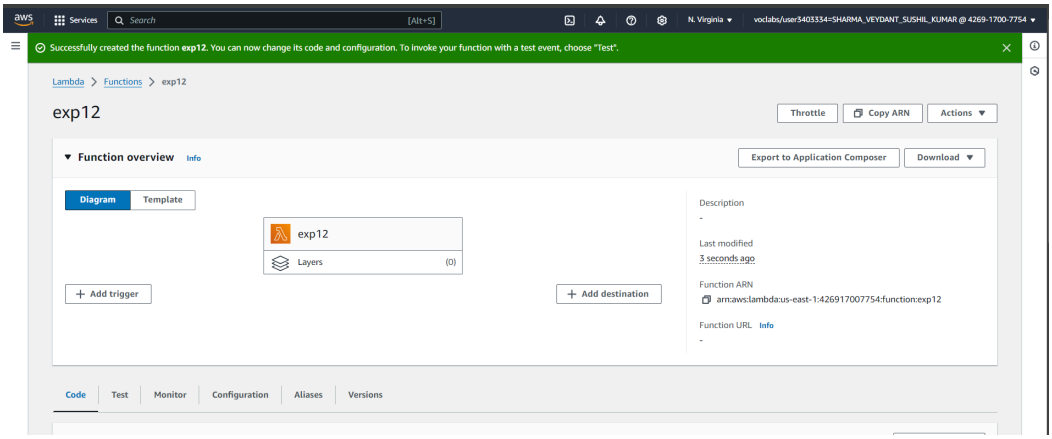


Step 4: Give your function a name and keep other settings as default

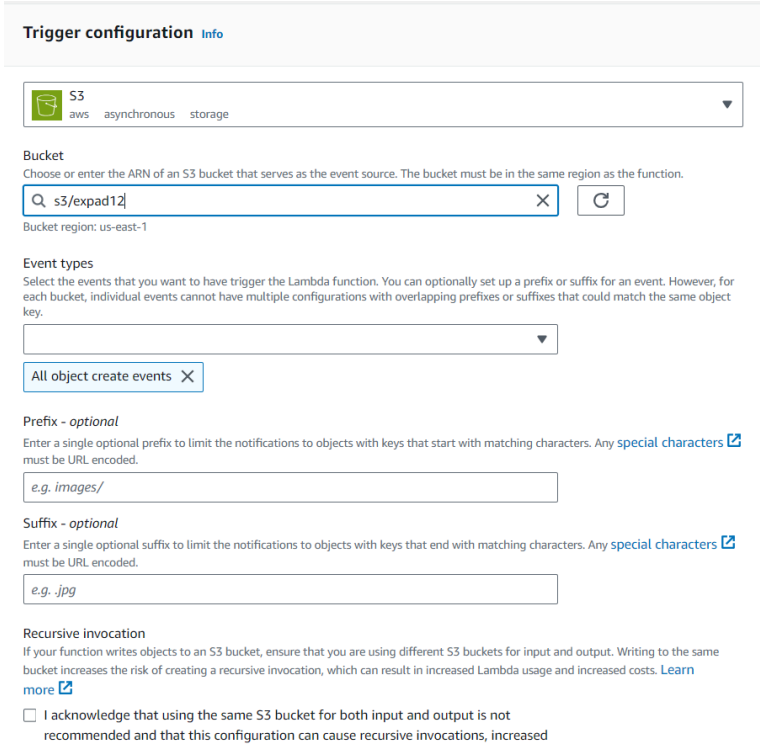


Step 5: On the page of the function you created, click on ‘Add trigger’

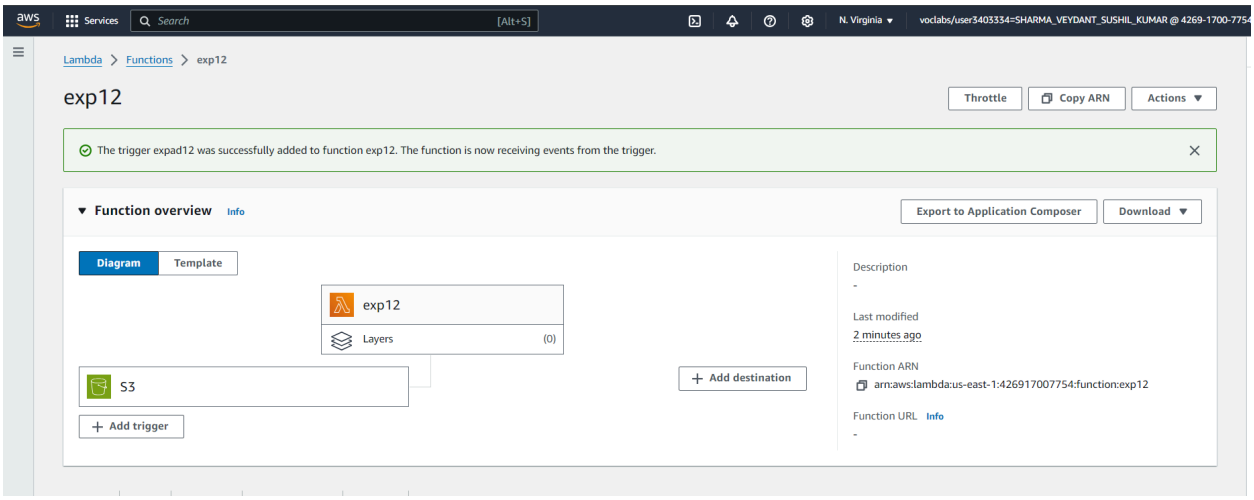
Step 6: Choose ‘Trigger configuration’ as S3 and select the name of your bucket in the dropdown box below it. Keep other options as default and click on ‘Add’.



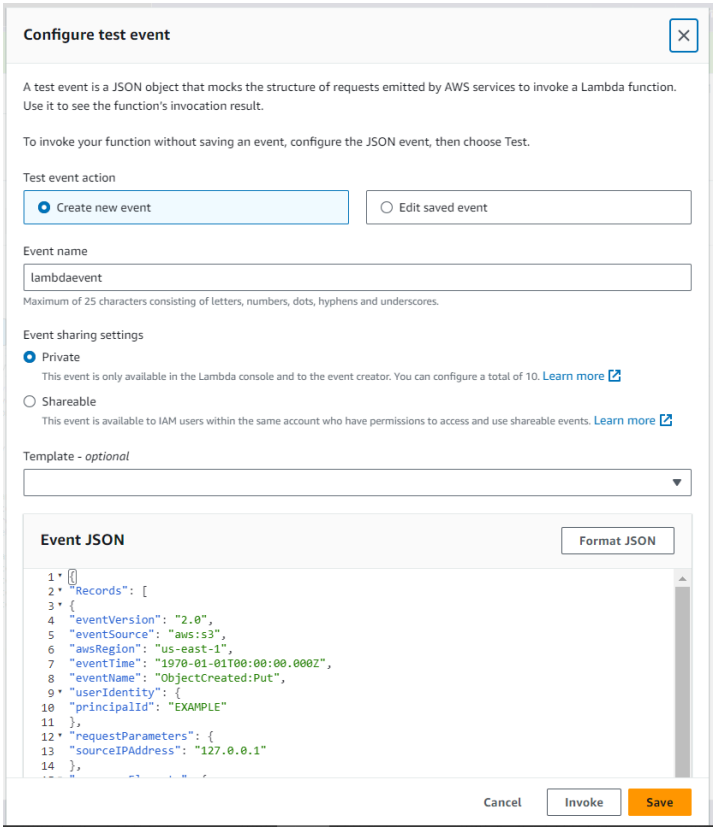
The trigger gets successfully added to your function



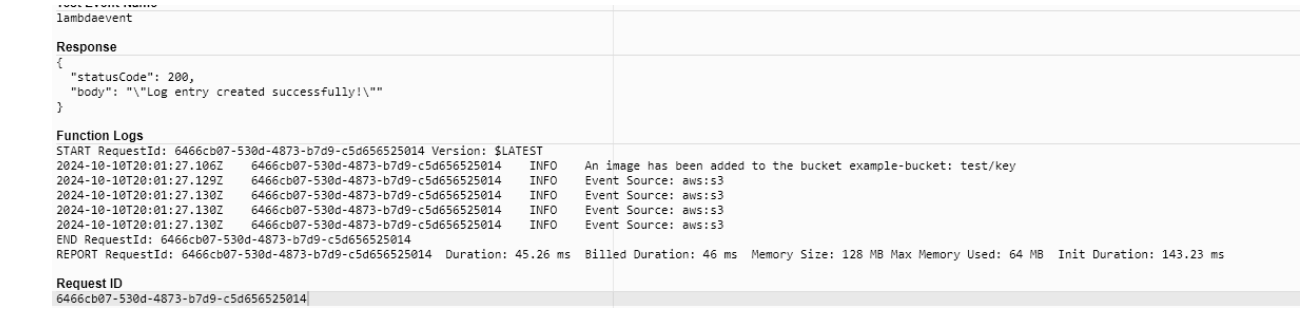
Step 7: In the ‘Code source’ section of your function, paste the following javascript code instead of the existing code:-
export const handler = async (event) => { if (!event.Records || event.Records.length === 0) { console.error("No records found in the event."); return { statusCode: 400, body: JSON.stringify('No records found in the event') }; } // Extract bucket name and object key from the event const record = event.Records[0]; const bucketName = record.s3.bucket.name; const objectKey = decodeURIComponent(record.s3.object.key.replace(/\+/g, ' ')); // Handle encoded keys console.log(`An image has been added to the bucket ${bucketName}: ${objectKey}`); console.log(`Event Source: ${record.eventSource}`); console.log(`Event Source: ${record.eventSource}`); console.log(`Event Source: ${record.eventSource}`); console.log(`Event Source: ${record.eventSource}`); return { statusCode: 200, body: JSON.stringify('Log entry created successfully!') }; };

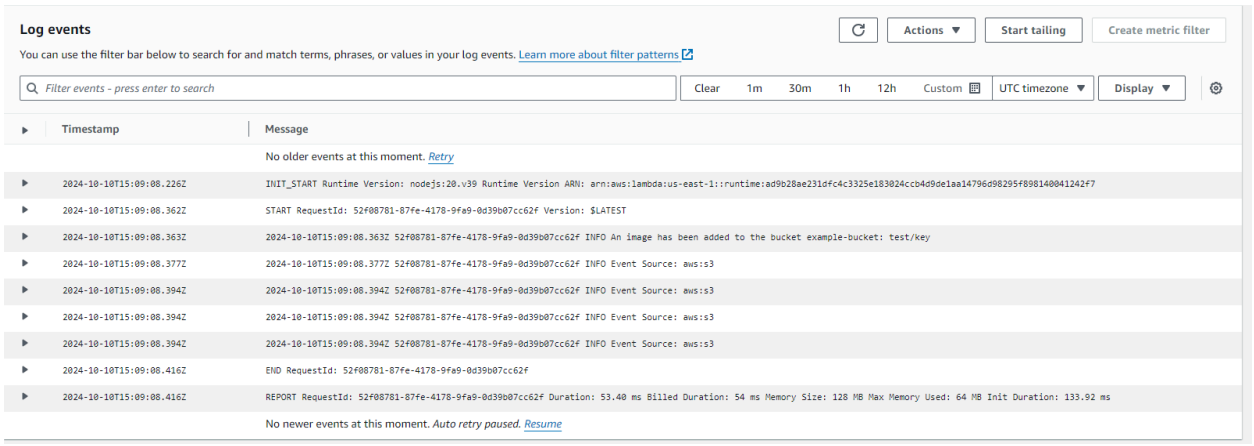


Step 8: Click on the arrow next to the ‘Test’ button and click on ‘Configure test event’. In the popup box that appears, if you have an existing event, enter the name of your event or create a new event and add the code.



Step 9: Response generated after testing the code logs that the image was uploaded successfully.





Conclusion

- In this experiment we created Lambda function with S3 bucket that uploads our image.
- We configured the ‘Code section’ of our Lambda function and a test event for our Lambda function.
- A log of our function was also created showing successful operation.

Vaidant Sharma
D15C 50

05/03

Assignment 1

Q1] Use S3 bucket and host video streaming

→ Step 1: Create an S3 bucket.

i) Navigate to S3 bucket

ii) create a new bucket
- → Click on "Create bucket"
- → Enter bucket name.
- → Choose a region
- → Create bucket

ii) Configure bucket permissions
- → Click on your bucket
- → Go to permissions tab
- → Edit block public acces setting
- → Add bucket policy to allow

iii) Upload video files
- → Use console, AWS CLI or SDKs to upload files on supported formats.

→ Step 2: Set up CloudFront

i) → Open cloudfront
- → Create origin access identities and give name.

iii) → Go to create CloudFront distribution
→ Select legacy access identities, select identity that you created.

iv) In default cache behaviour, select redirect HTTP to HTTPs.

v) To access, copy domain name of distributer

vi) Go to S3 bucket and click on the name of the video you uploaded.

vii) Combine domain name and key of distribution and video respectively to make the final link.

Q2) Discuss BMW & hotstar case studies using aws.

→ i) BMW case study
Objective: BMW aims to enhance connected vehicle services & improve data management.

AWS Utilization:
→ ~~BMW lever~~
→ a) Data storage & Analytics:
BMW leverages amazon S3 for storing vast amounts of data collected from vehicles and is analyzed by AWS using Redshift or athena.

b) IoT integration:
The company employs AWS IOT core to securely connect its vehicles to the cloud facilitating real time data transfers.

c) Machine Learning:
BMW utilizes Amazon sage maker for developing machine learning models that enhance customer experiences.

ii) Hotstar case study

Objective: Hotstar, a leading Streaming service provider, aims to deliver high quality video content to millions of users.

AWS utilization:

a) Scalable infrastructure:
Hotstar uses EC2 instances to scale its computing resources dynamically based on demand.

b) Content delivery:
Platform uses Amazon CloudFront to distribute video globally, ensuring low latency and high availability.

c) Data analytics.
Hotstar employs AWS analytic service to gain insights into user preferences and viewing habits.

Q3) Why Kubernetes & advantages, disadvantages of kubernetes. How adidas uses kubernetes

→ Kubernetes is an open source container orchestration platform that automates deployment, scaling and management of containerized applications.

- Advantages:

1 Scalability:
→ Kubernetes can automatically scale applications based on demand.

2 High availability:-
→ Provides mechanisms for automatic restarts, replication & load balancing.

3 Resource Optimization
→ Schedules & manages containers effectively.

4 Portability:
→ Applications can run consistently across various environments.

- Disadvantages
i) Complexity:
→ Has a steep learning curve.

ii) Overhead

iii) Security challenges
→ Misconfigurations can lead to vulnerabilities.

iv) Monitoring needs.
→ Effective monitoring is essential.

* How Adidas uses Kubernetes.
→ Adidas uses it to manage its microservices architecture enabling rapid deployment and development of application

→ Its benifits include:

i) Support for Ecommerce Operations

ii) Enhanced Collaboration

iii) Improved resource management.

Q4 What are Nagios & explain how Nagios is are used in E services?

→ Nagios is an open source monitoring system that helps organizations monitor the health and performance of IT infrastructure

→ Nagios in E-services:

1 Monitoring uptime & downtime:
Nagios monitors availability of e services, ~~monitoring key~~ such as websites, email servers or cloud platforms ensuring that these services stay online.

2 Performance monitoring:
It tracks the performance of E services, monitoring key metrix.

3 Alerts and notifications:
Nagios can be configured to send notifications via email or SMS.

4. Log Monitoring

5. Custom plugins:
→ Allows installation of custom plugins for monitoring specific e-services.

34 What are Nagios & explain how Nagios is are used in E services?

→ Nagios is an open source monitoring system that helps organizations monitor the health and performance of IT infrastructure

→ Nagios in E-services:

1 Monitoring uptime & downtime:
Nagios monitors availaibility of e services, ~~monitoring key~~ such as websites, email servers or cloud platforms ensuring that these services stay online.

2 Performance monitoring:
It tracks the performance of E services, monitoring key metrix.

3 Alerts and notifications:
Nagios can be configured to send notifications via email or SMS.

4. Log Monitoring

5. Custom plugins:
→ Allows installation of custom plugins for monitoring specific e-services.

Vedant Sharma
D15C 50

Assignment 2

Create a rest api with a serverless framework.

Install Node.js and NPM from node.js website steps:

Install serverless framework
The serverless framework is a CLI tool that helps deploy serverless applications, managing resources like Lambda, API Gateway & more.
command: npm install -g serverless

Create a new serverless Project.
The serverless framework provides templates to scaffold a new serverless service.
command: serverless create --template aws-nodejs --path my-rest-api
cd my-rest-api

Configure credentials
command: Aws configure

Define API routes
command: Modify serverless.yml as follows

```
service: my-rest-api
provider:
name: aws
runtime: nodejs14.x
region: us-east-1
stage: dev
functions:
createItem:
handler: handler.createItem
events:
-http:
path: item
method: post
getItem:
Handler: handler.getItem
events:
-http:
path: item/{id}
method: get
updateItem:
handler: handler.updateItem
events:
-http:
path: item/{id}
method: put
deleteItem:
handler: handler.deleteitem
events:
-http:
path: item/{id}
method: delete
```

5. Write lambda function in handler.js.

6. Deploy REST API
command: serverless deploy

7. Test the API
Test by using tools like Postman or curl to REST API

8. Monitor & Logs:
For performance analysis.

9. Case study for Sonarqube

10. Create your own profile for Sonarqube for testing project quality.

- Use Sonarcloud to analyze your github code.

- Install sonarlint in your java intellij id or eclipse.

Analyze python project with sonarqube.

Analyze nodejs project with sonarqube.

→ Sonarqube helps developers identify issues like bugs, security etc.

1. Create personal profile
   - Sign in to sonarqube
   - Connect your project.

2. Sonarcloud integrates with github, offering continuous analysis.

3. Sonarlint is a plugin that offers realtime code quality checks while coding in IDE.

4. Sonarqube supports python projects To analy
   - setup the python project with a sonar proj properties file.
   - Run Sonarscanner
   - View the detailed

5. Analysis for node.js
   - create a sonar-project-properties fi in project root
   - Use sonarscanner for analysis
   - Review the report.

At a large organization your centralized operations team may get many repetetive infrastructure requests. You can use terraform to build a self serve infrastructure model that lets product teams manage infrastructure independently. You can create and use terraform modules that codify the standards for deploying and managing services in compliance with your organizations.

→ Self serve infrastructure with terraform
By using terraform you can build reusable modules that encapsulate the best practices and compliance standard of your own organizat
These modules can be made available to various product teams, empowering them to provision infrastructure themselves.

→ Terraform modules for standardization.
Terraform modules are reusable templates that simplify the process of deploy infrastructure by codifying the standards for managing resources.

Integration with ticketing systems like service now.
Terraform cloud on Enterprise can integrate with tools like service now, automating the process of generating new infrastructure request

• For instance, when a product team submits a ticket for infrastructure resources, the cloud can automatically handle provisioning based on preapproved templates.