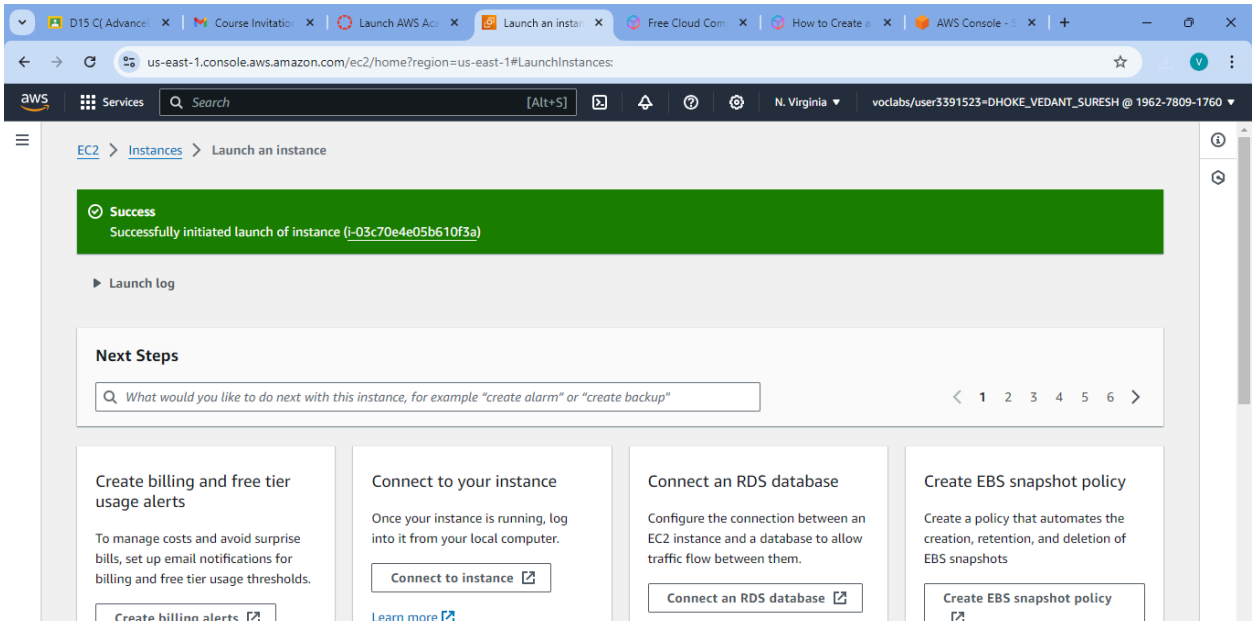
NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/9

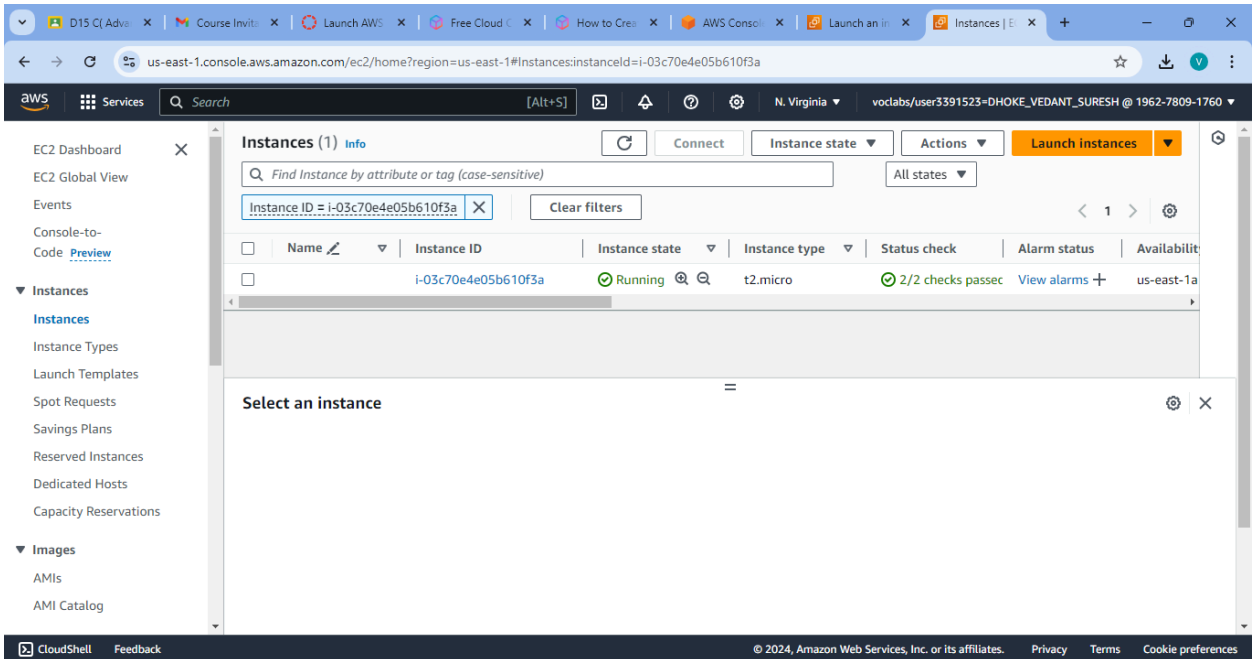
# AdvanceDevOps Exp 1A

**AIM: To understand the Creation of an EC2 Instance . To develop a website and host it on your local machine on a VM.Hosting a static website on Amazon S3.**

Firstly we have to create an EC2 Instance.



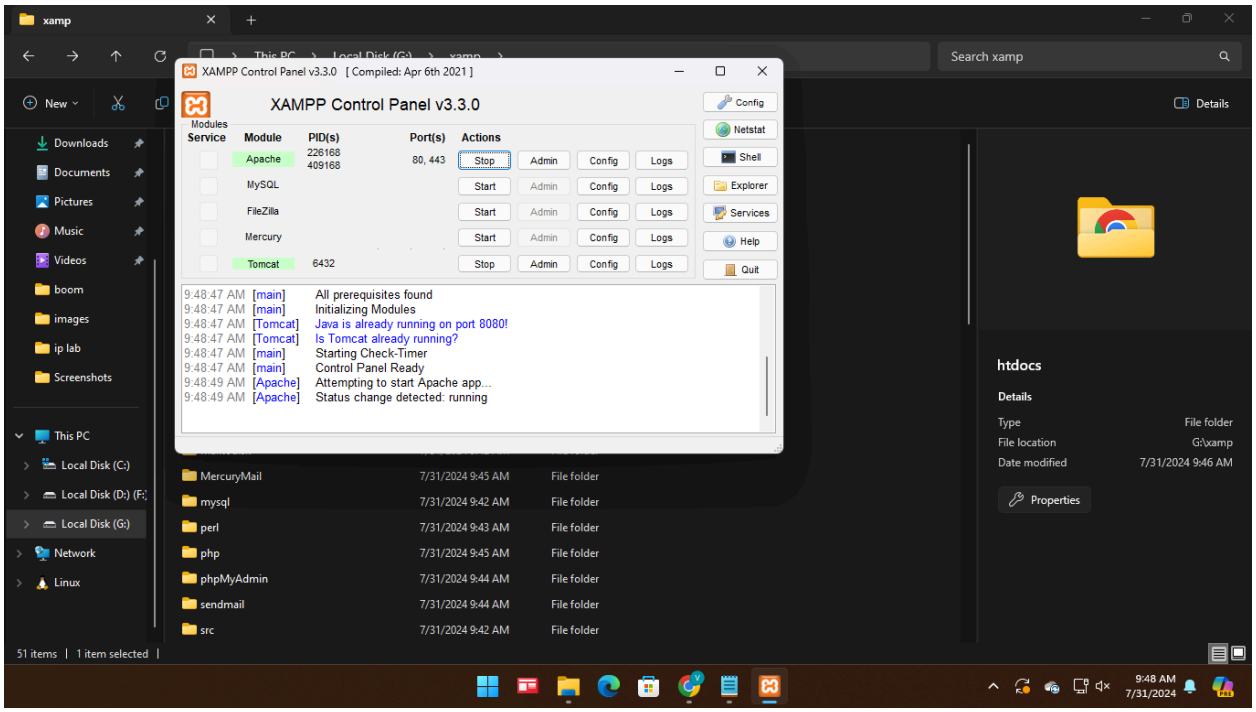

Launching our EC2 Instance



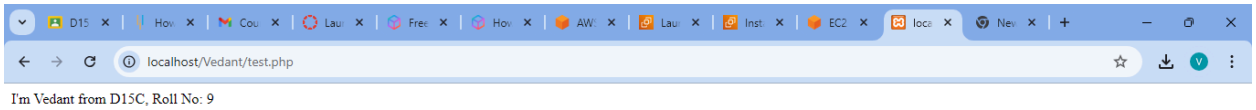

nce

NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/9

Download the XAMPP Server and Start it which will help us to host our website on local machine.

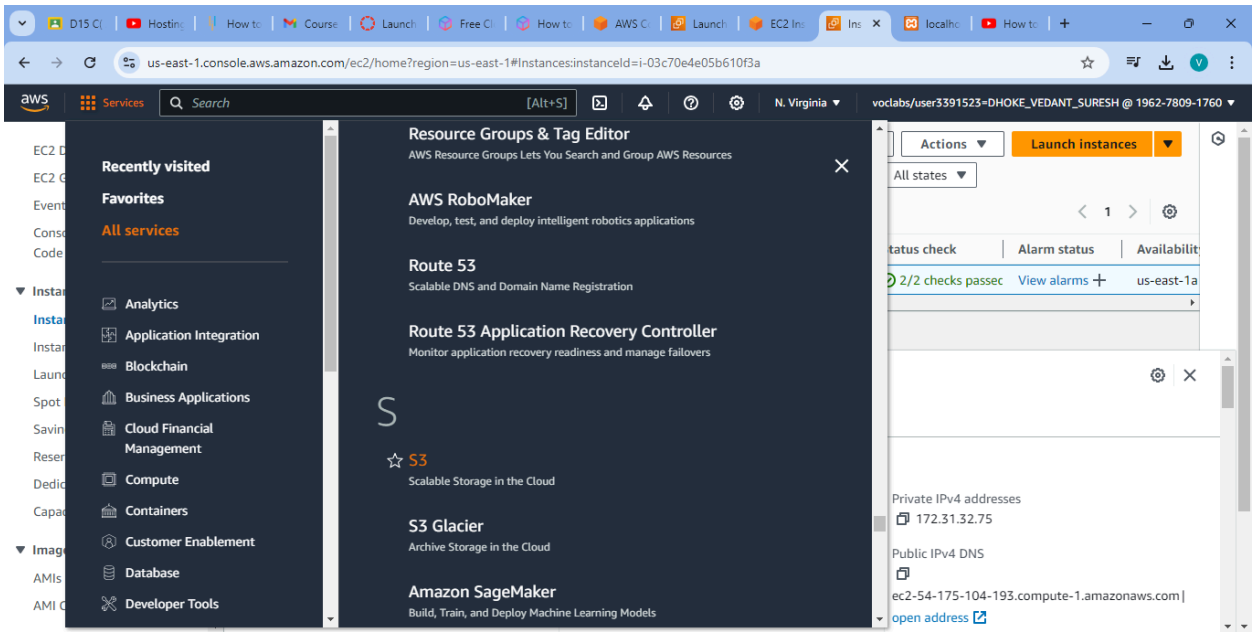



We can see that our website is running on the local machine.

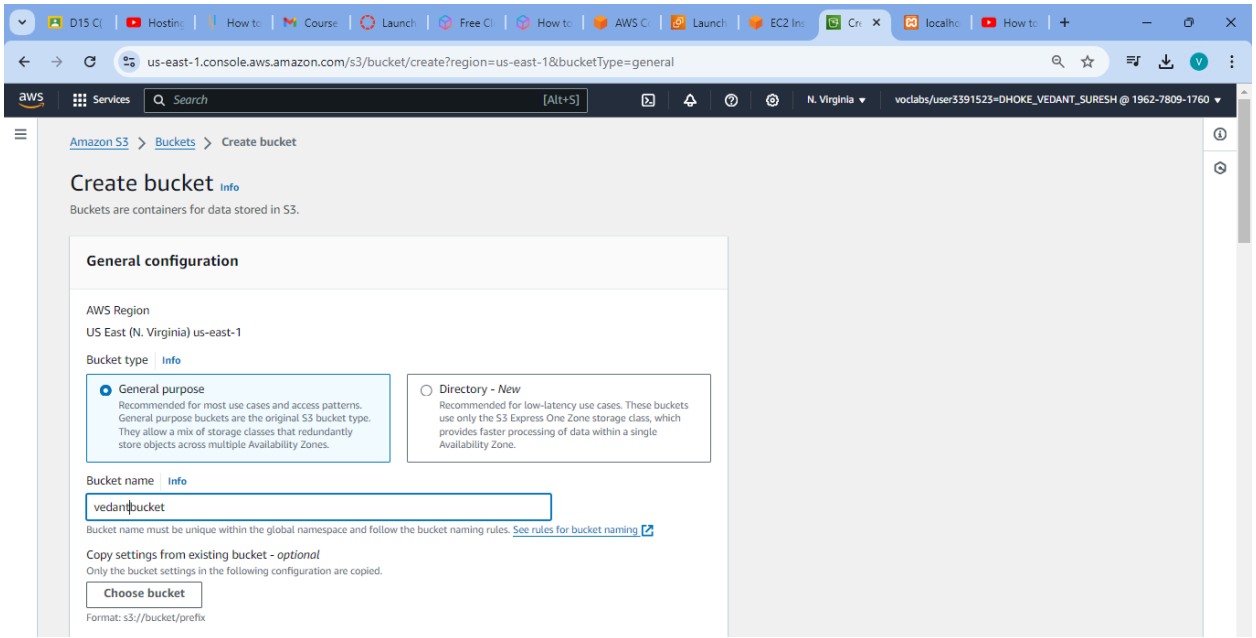


NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/9

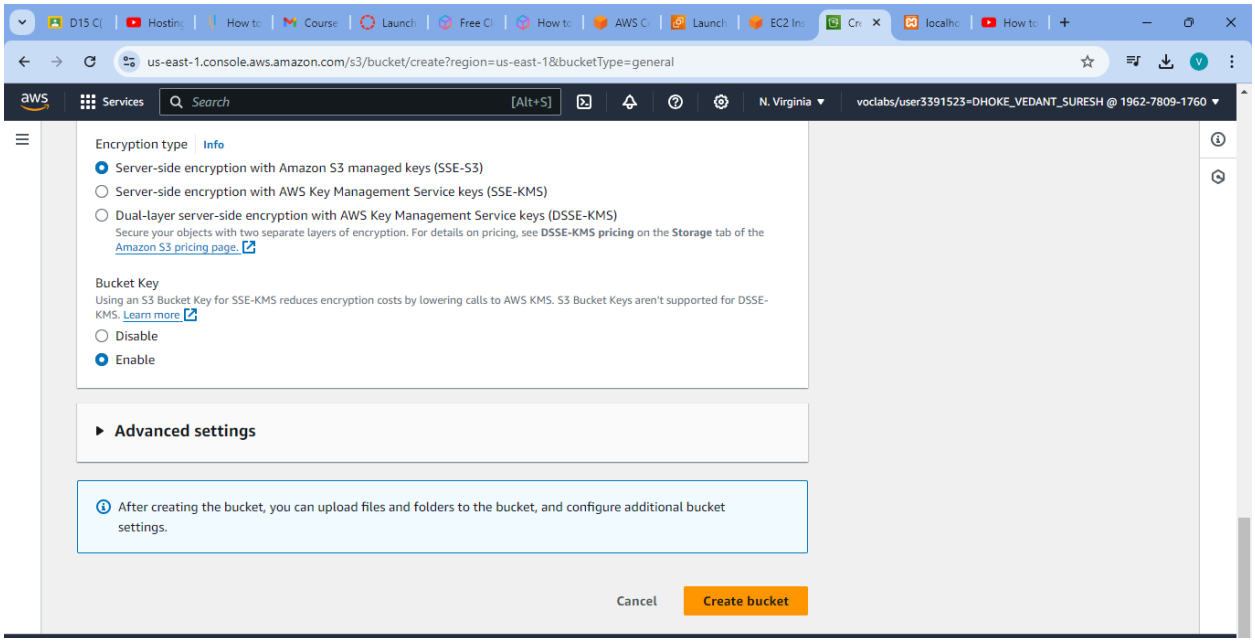
Now we want to create a S3 Bucket. Go to the services and select S3



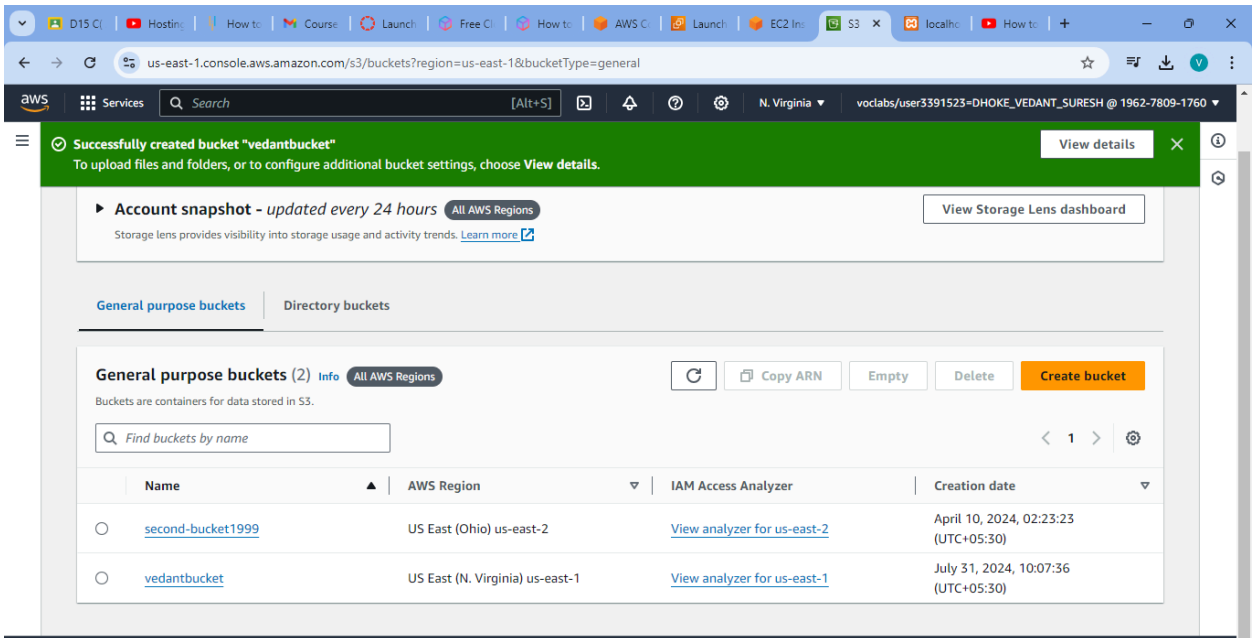

Give name to your bucket and configure the settings properly and click on create bucket.



NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/9

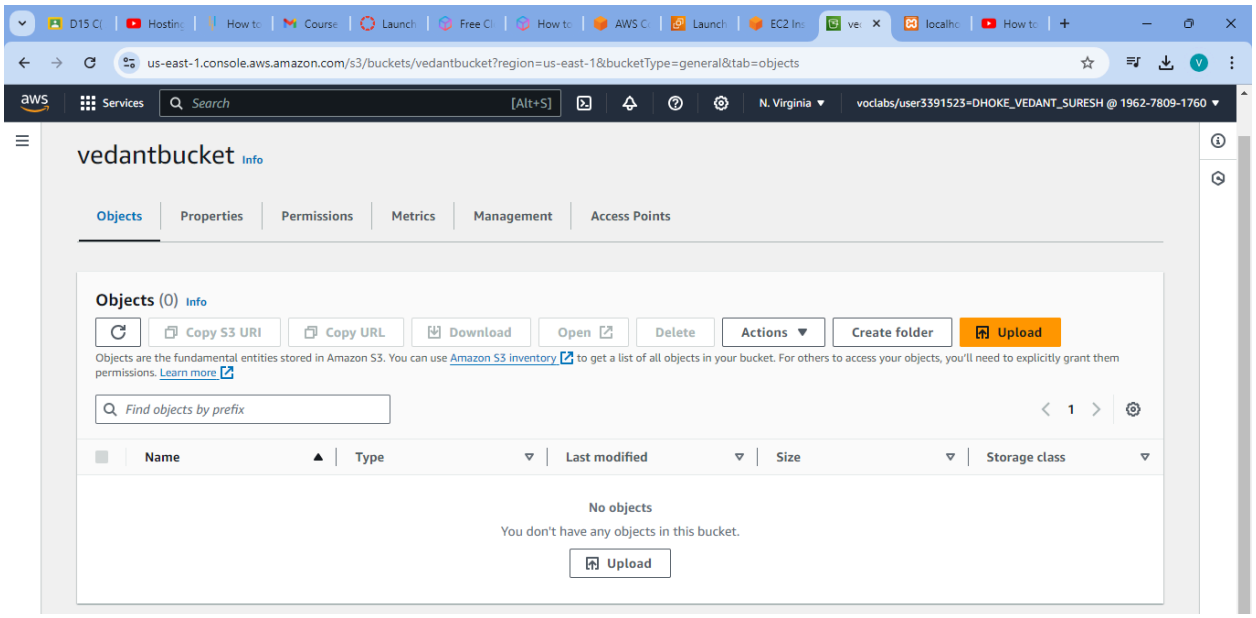



Now we can see that our bucket named “vedantbucket” is created Successfully.

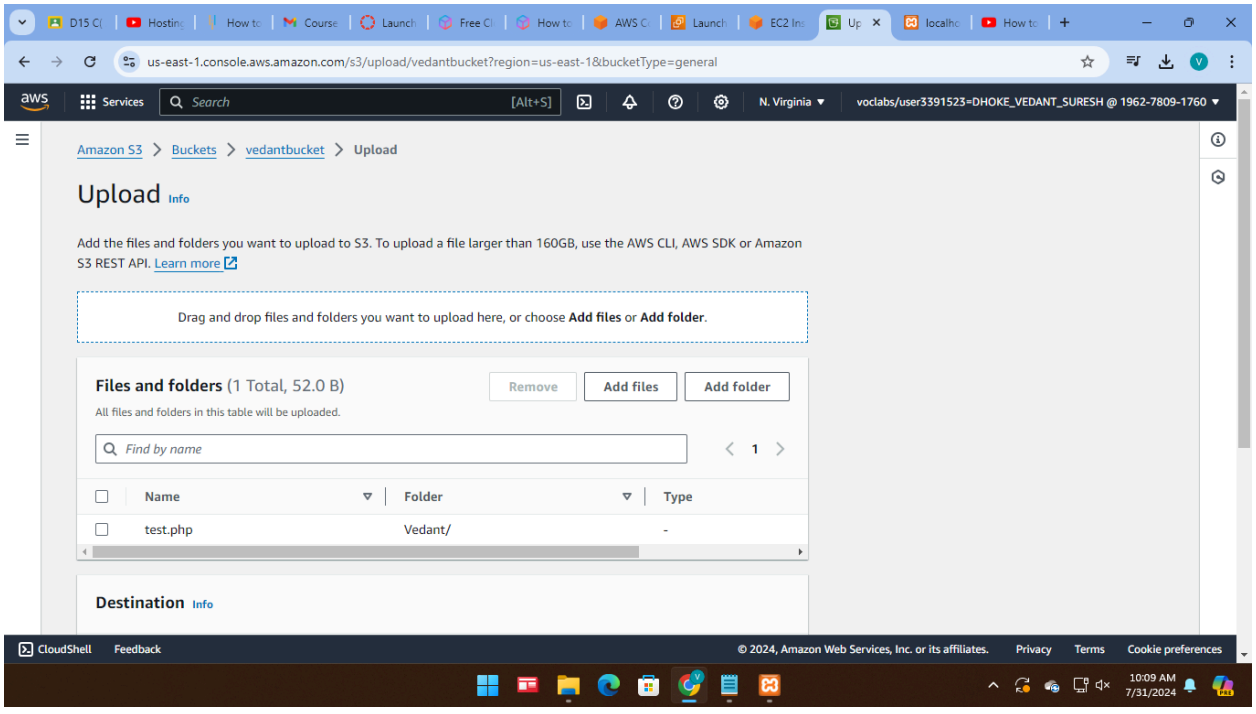


NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/9

This is how our bucket's structure looks like , We have to upload our files into the bucket so click on Upload button

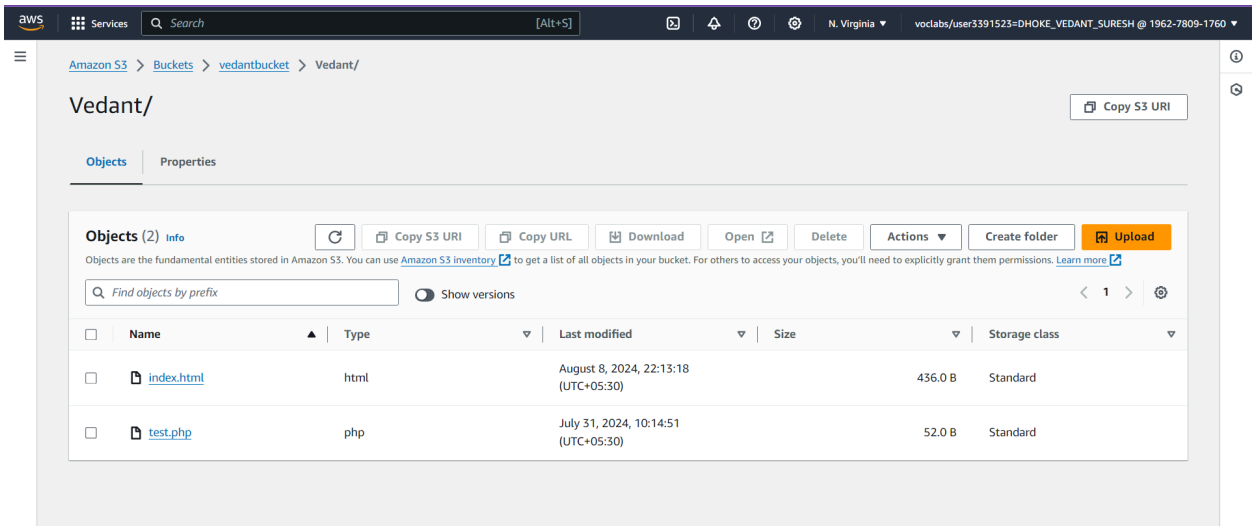



We now uploaded our test.php file in our bucket.

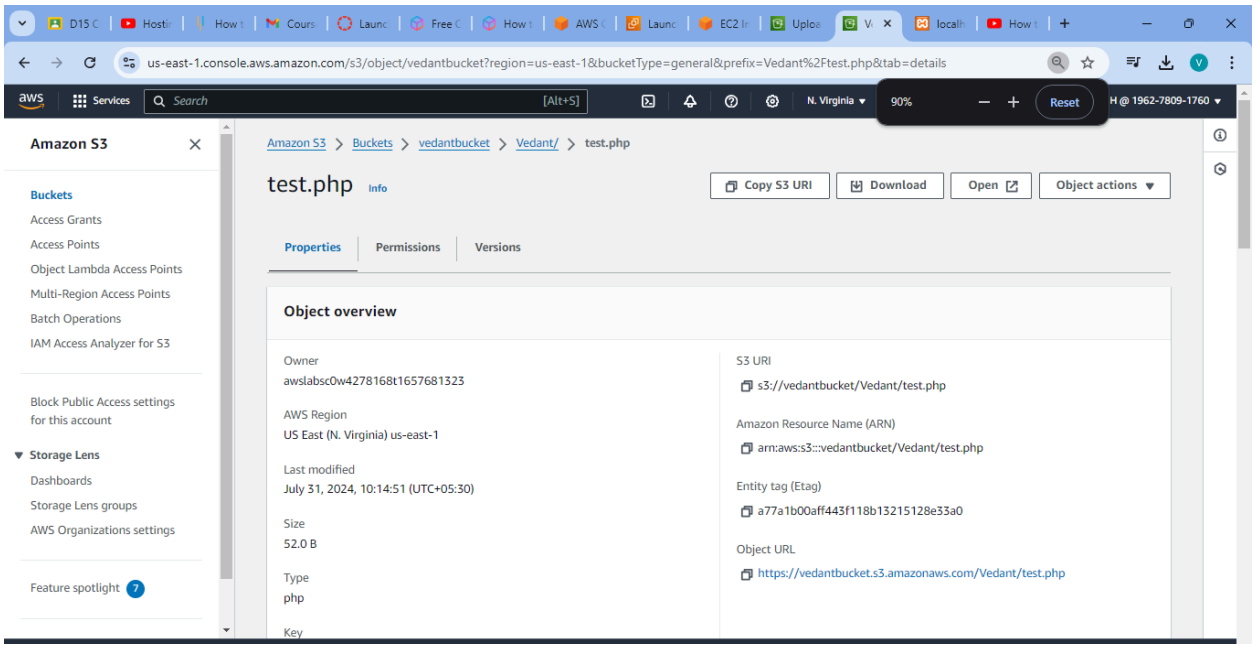


NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/9

Finally we have uploaded test.php and index.html to our bucket.



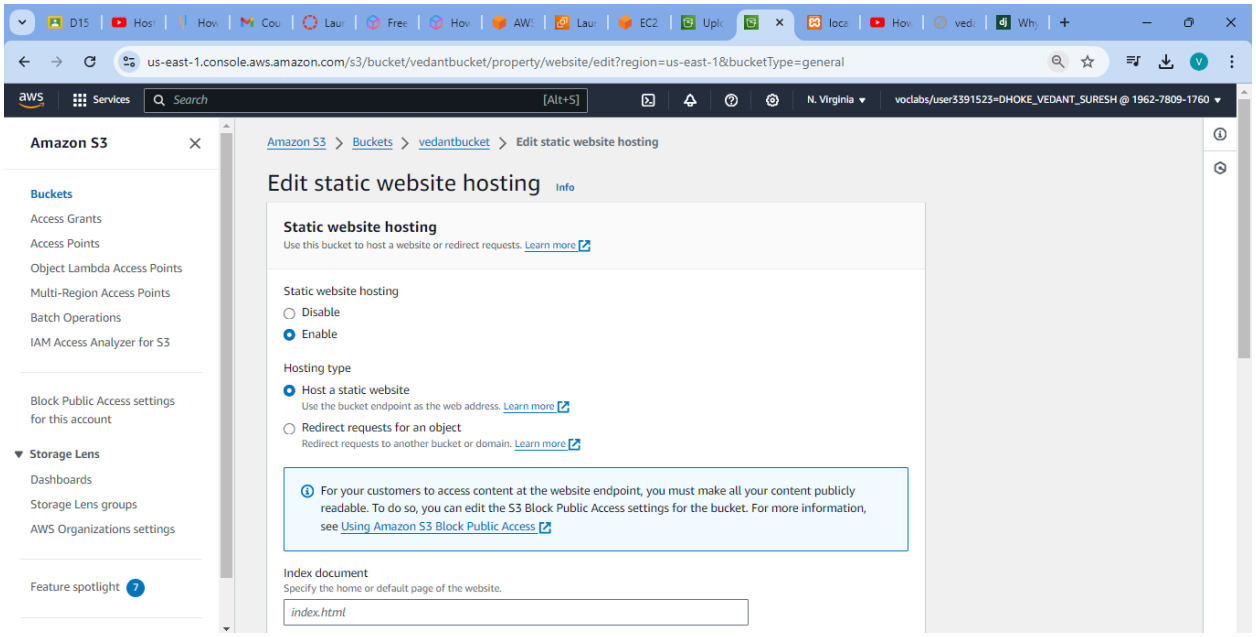

By Clicking on the files we can see their properties and configurations.



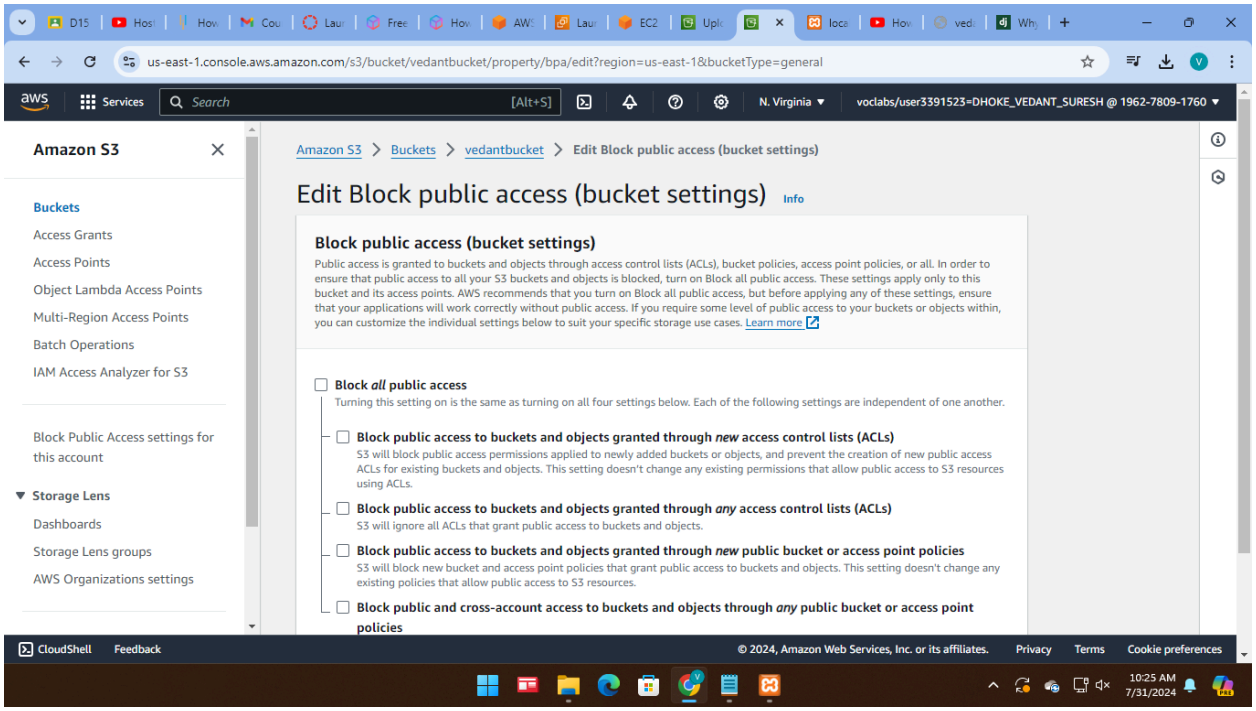
NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/9

Now to avoid access denied issue we have to edit some of the properties.

1. Editing the hosting type as Host a static website.

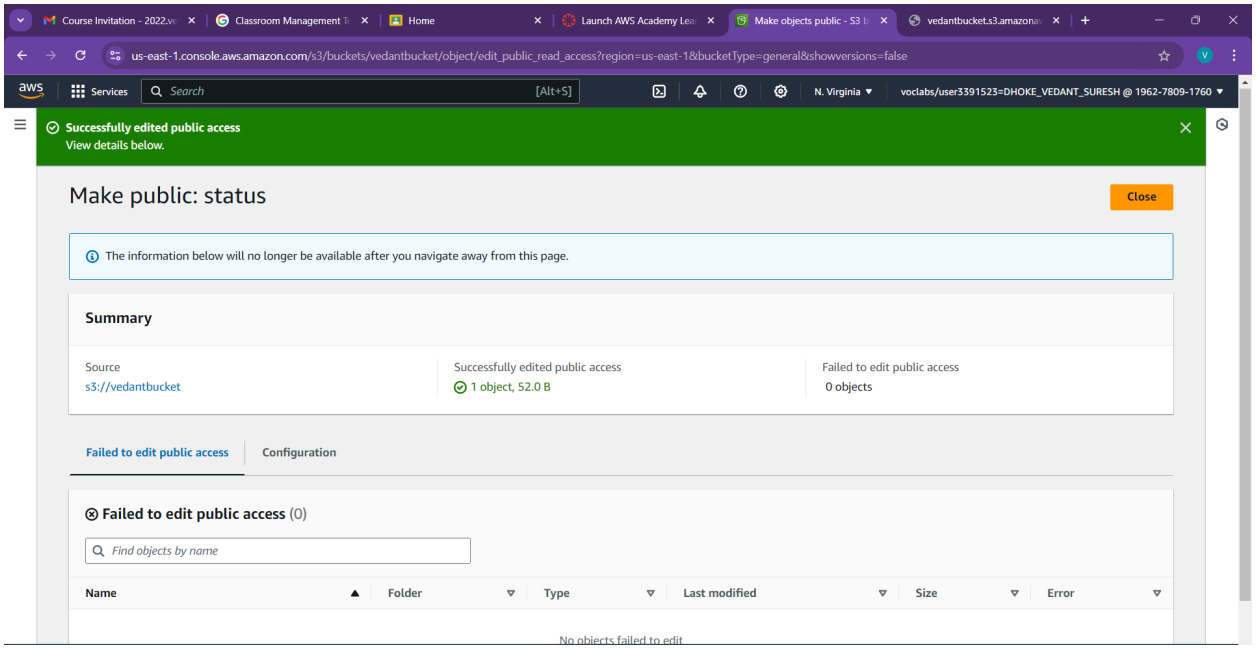



2. Editing the Block public access.

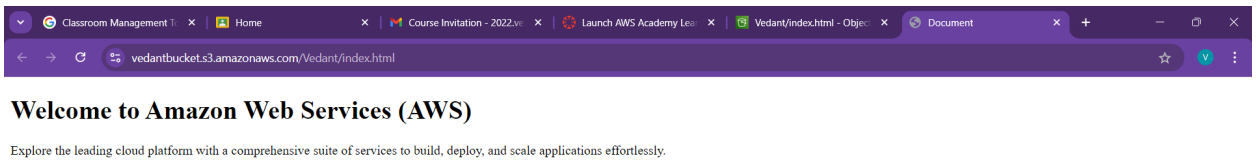


NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/9

3. Editing the Public access status.




Finally our static website is hosted Amazon S3.

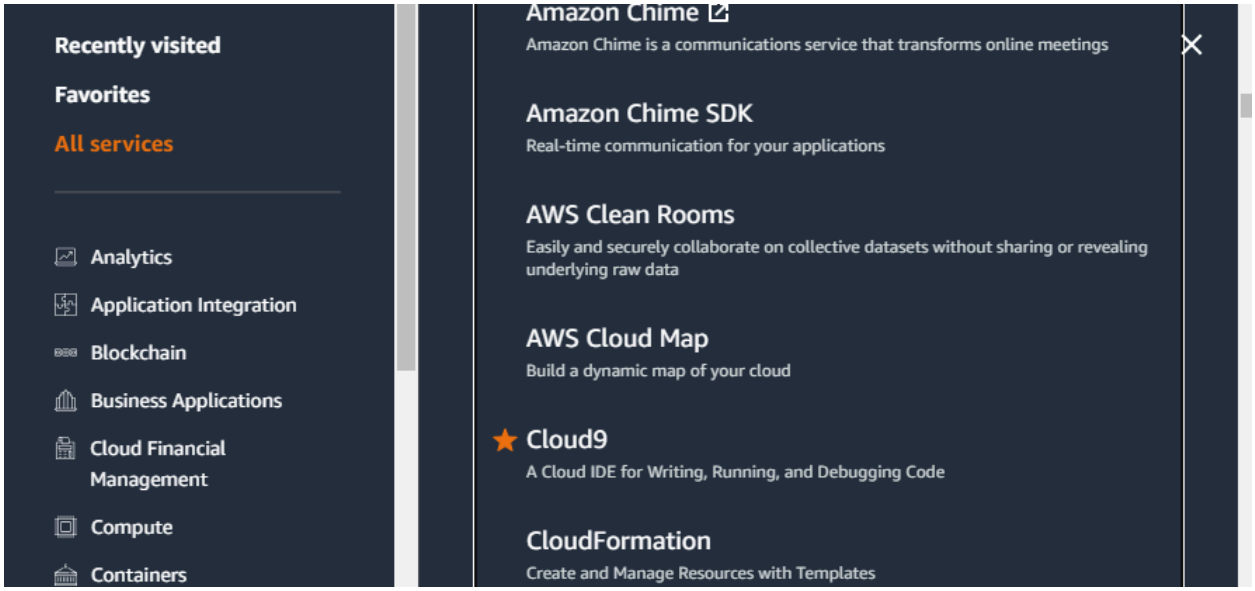


NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/9

# AdvDevops Exp 1B

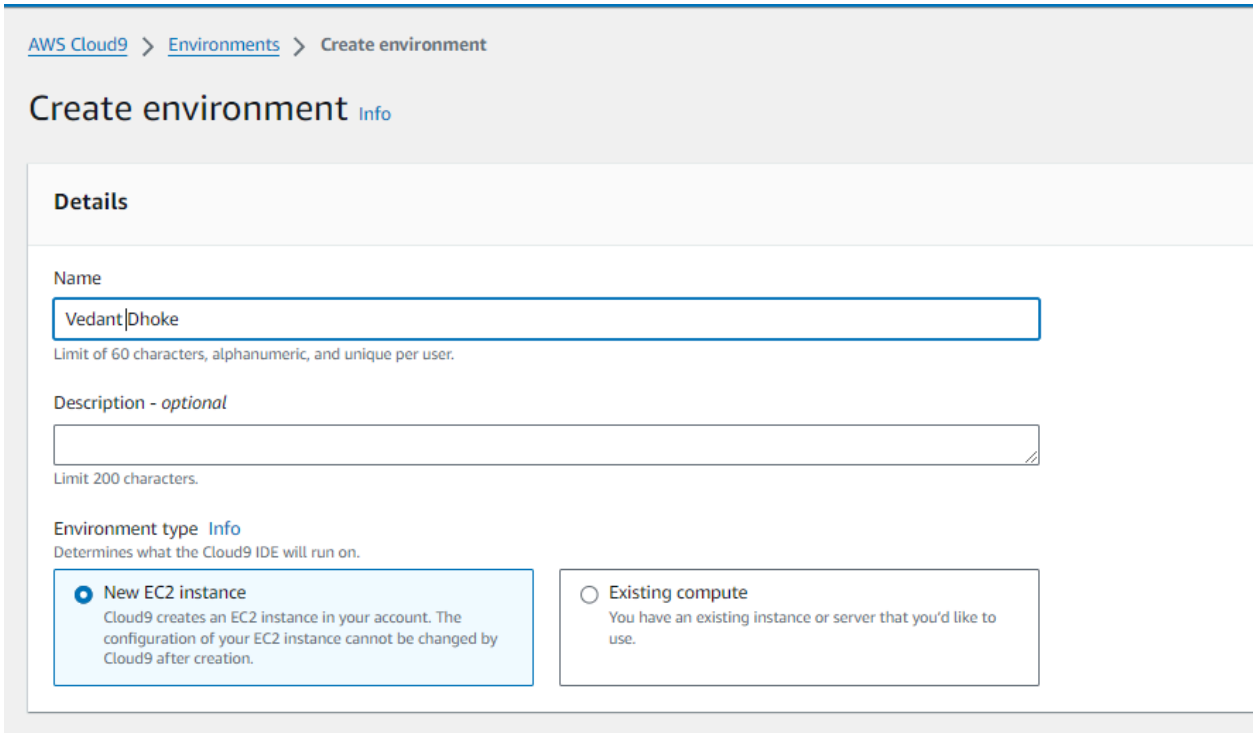
**Aim: To understand the benefits of Cloud Infrastructure and Setup AWS Cloud9 IDE, Launch AWS Cloud9 IDE and Perform Collaboration Demonstration.**

1. Login to your AWS account and navigate to the Cloud9.

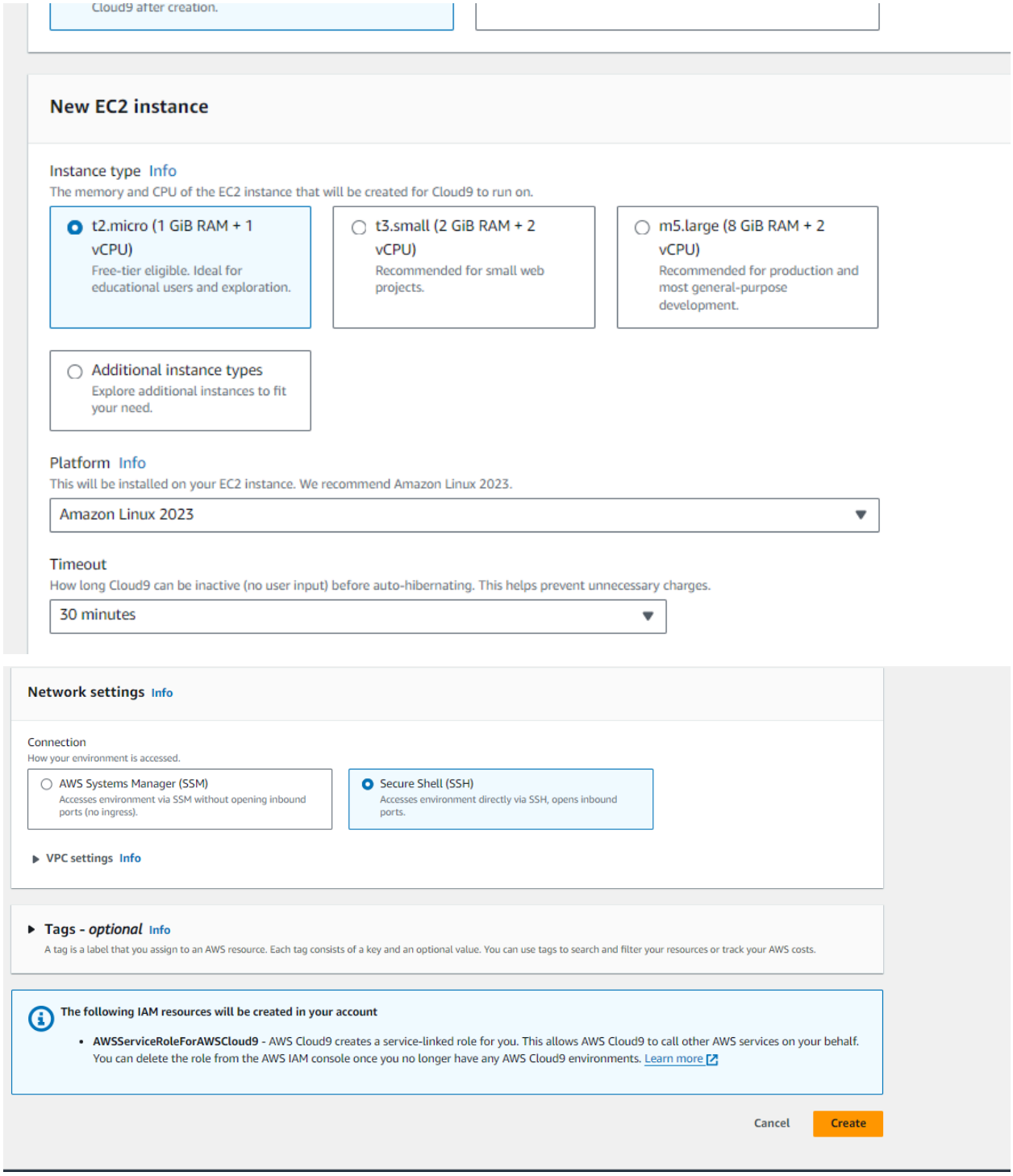


NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/9

2. We want to create an environment in Cloud9.

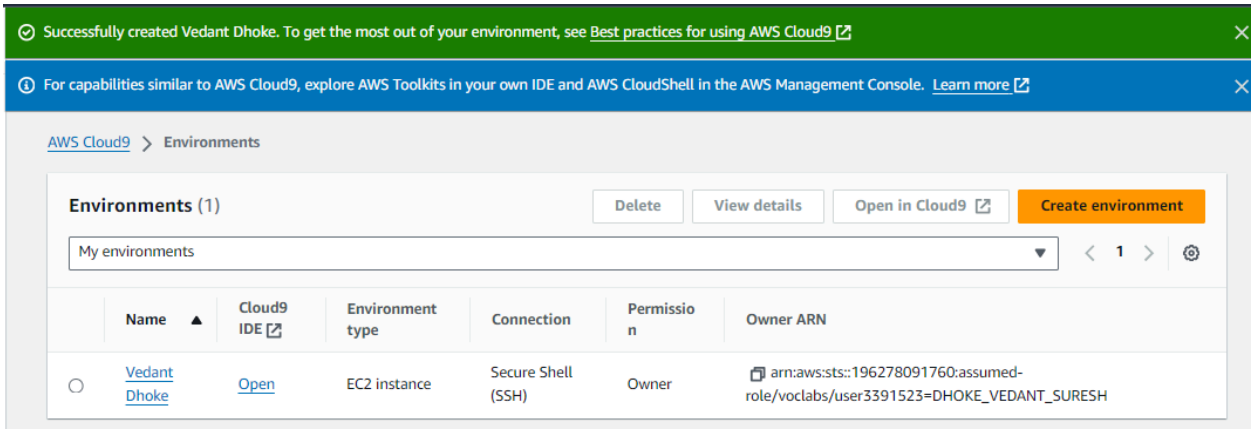


3. Configure the environment settings and click on **create** button.

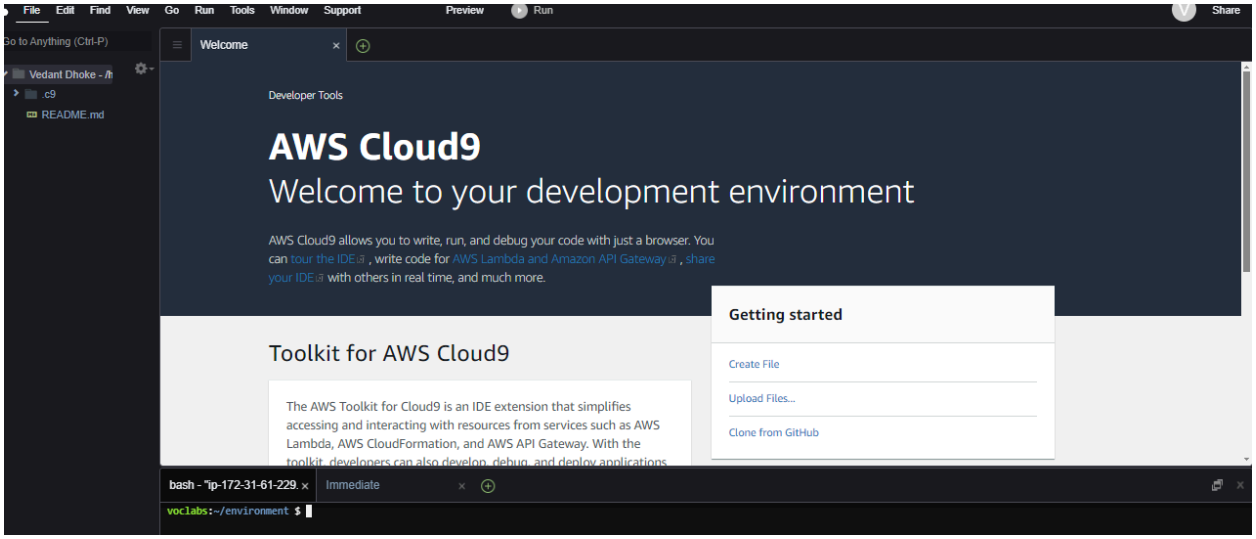


NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/9

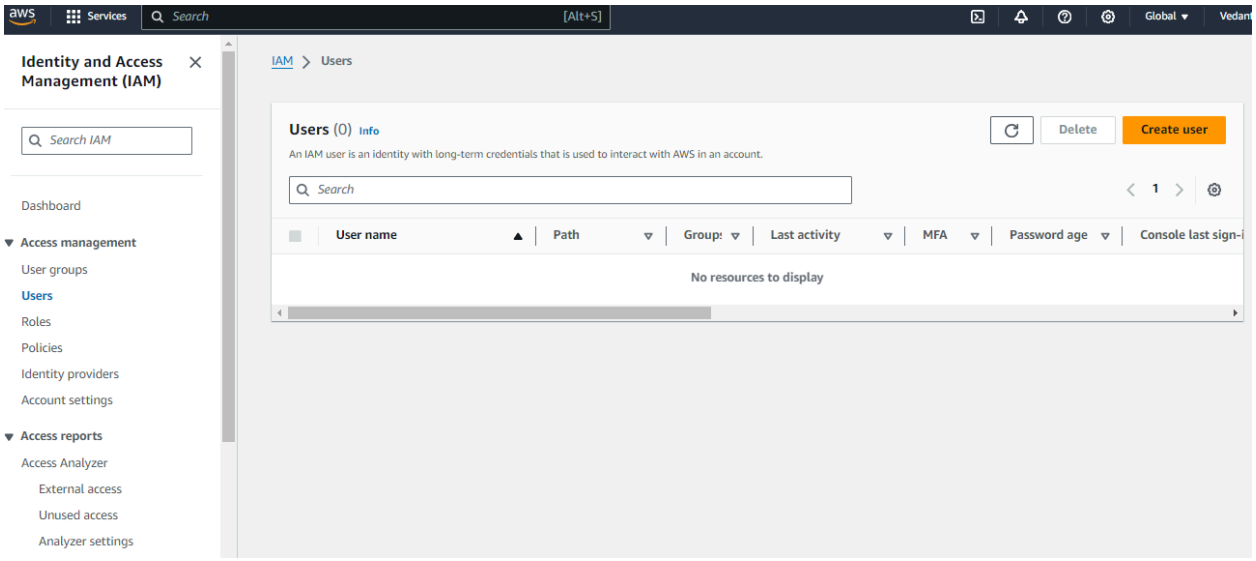
4. Our Enviroment is created Successfully.



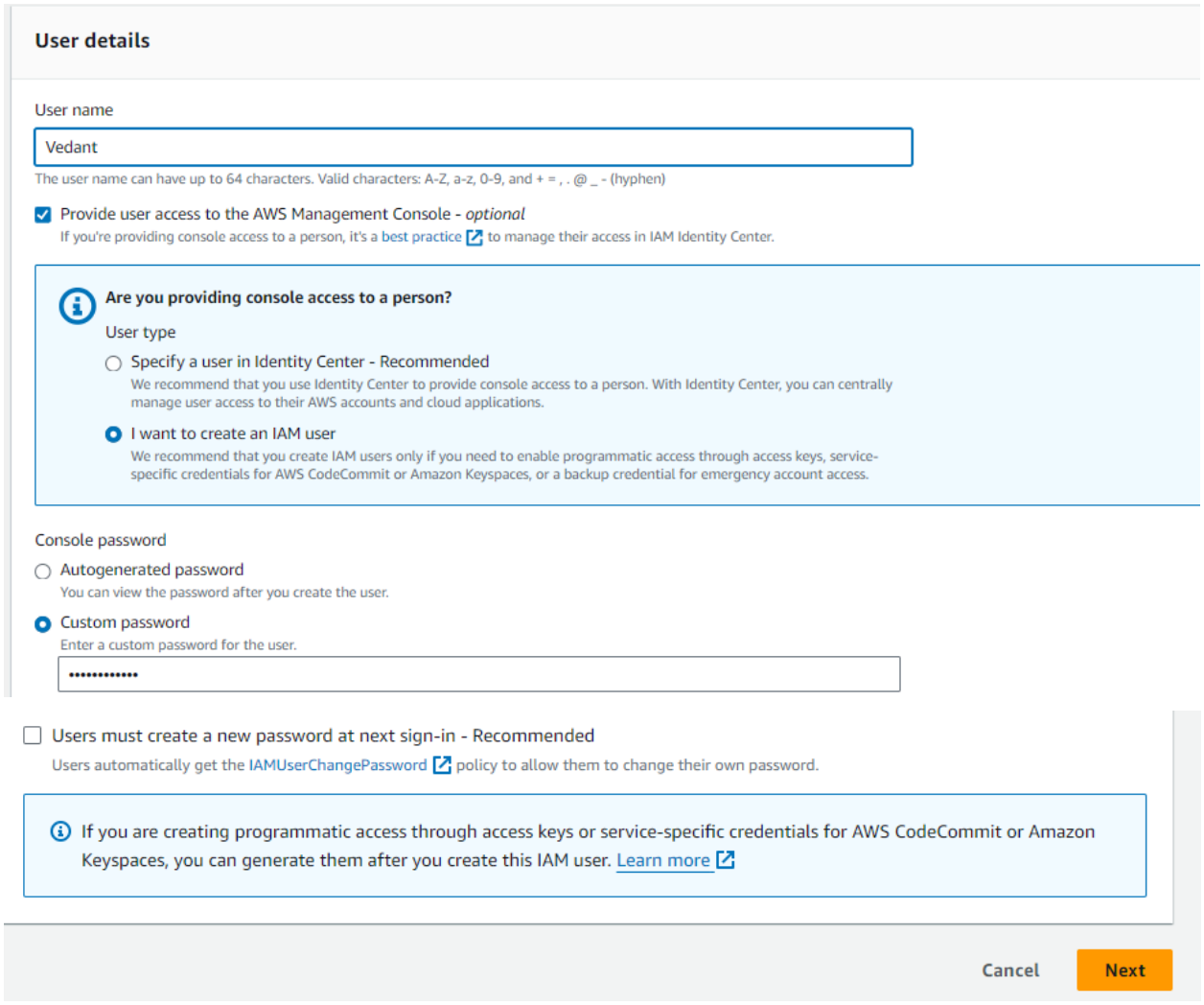

5. It will take few minutes to create aws instance for your Cloud 9 Enviornment.




6.Now we want to Add users

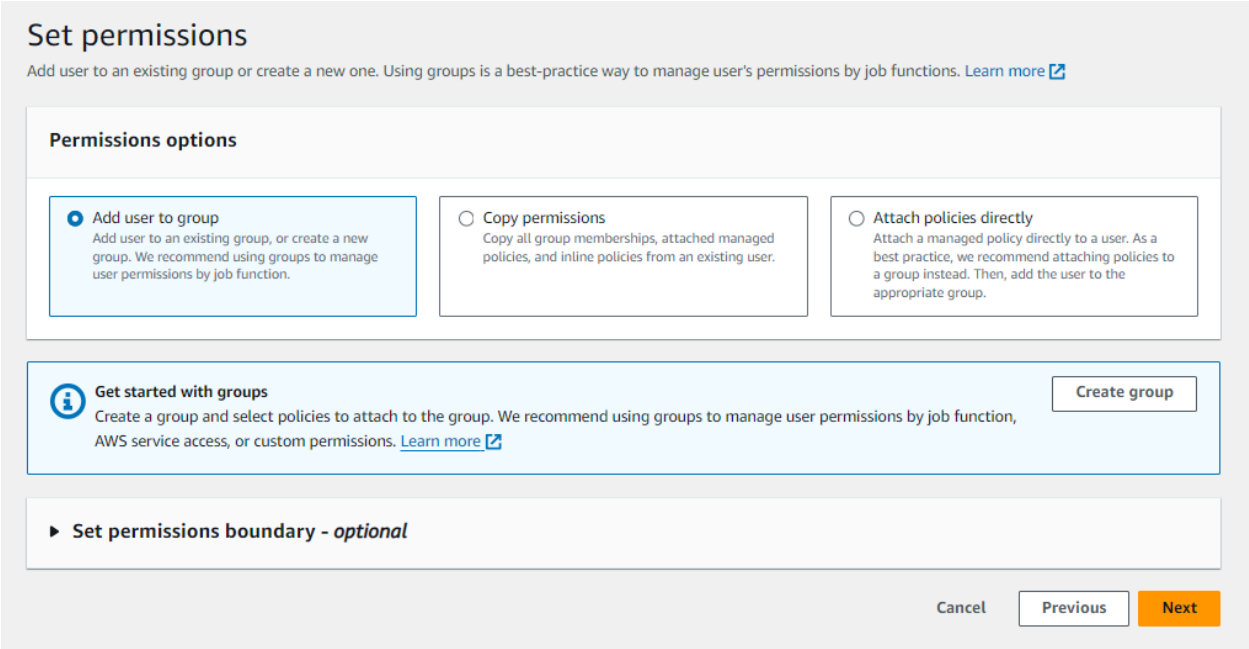


7. Add user provide manual password if you want and click on Next.

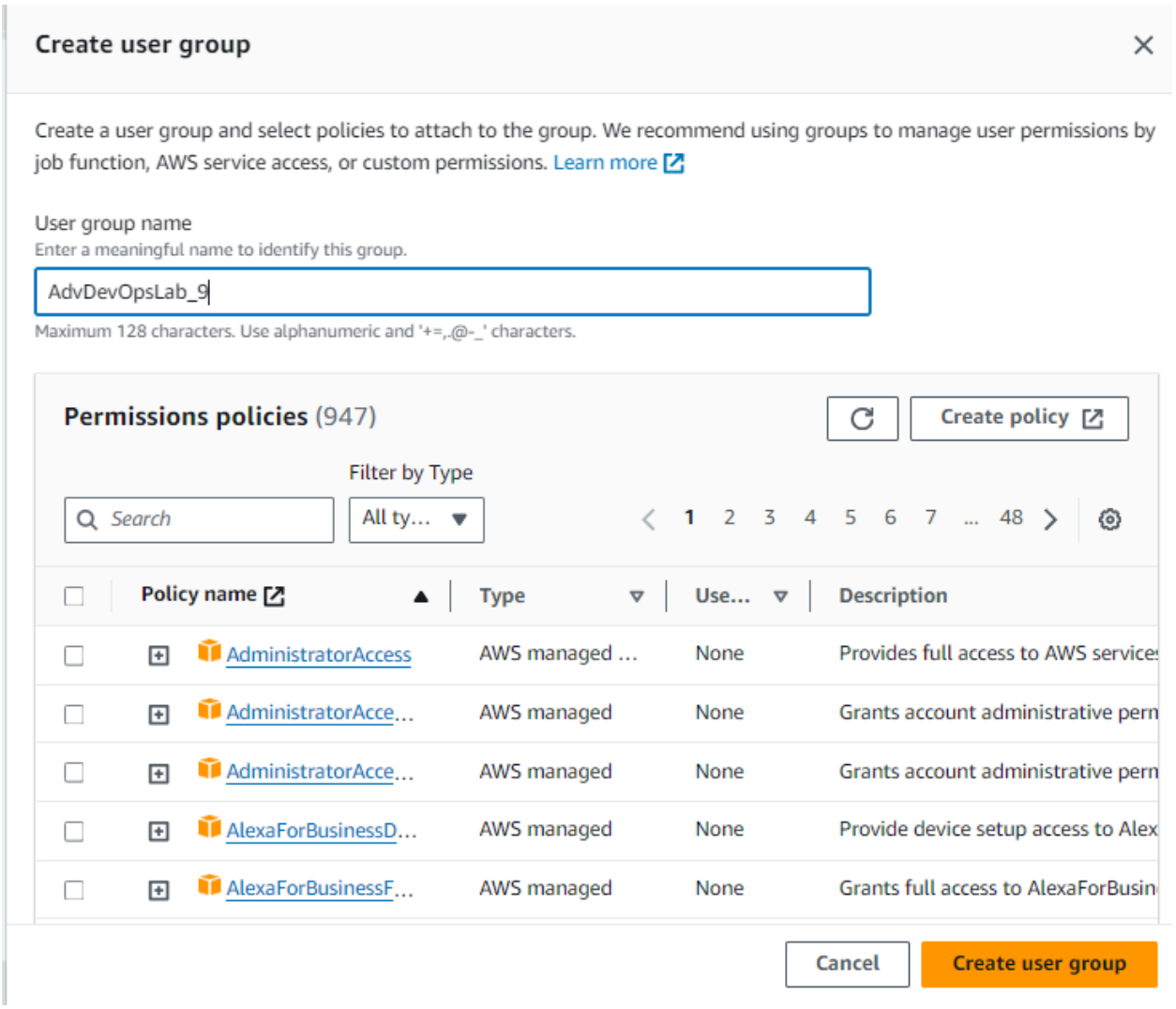


NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/9

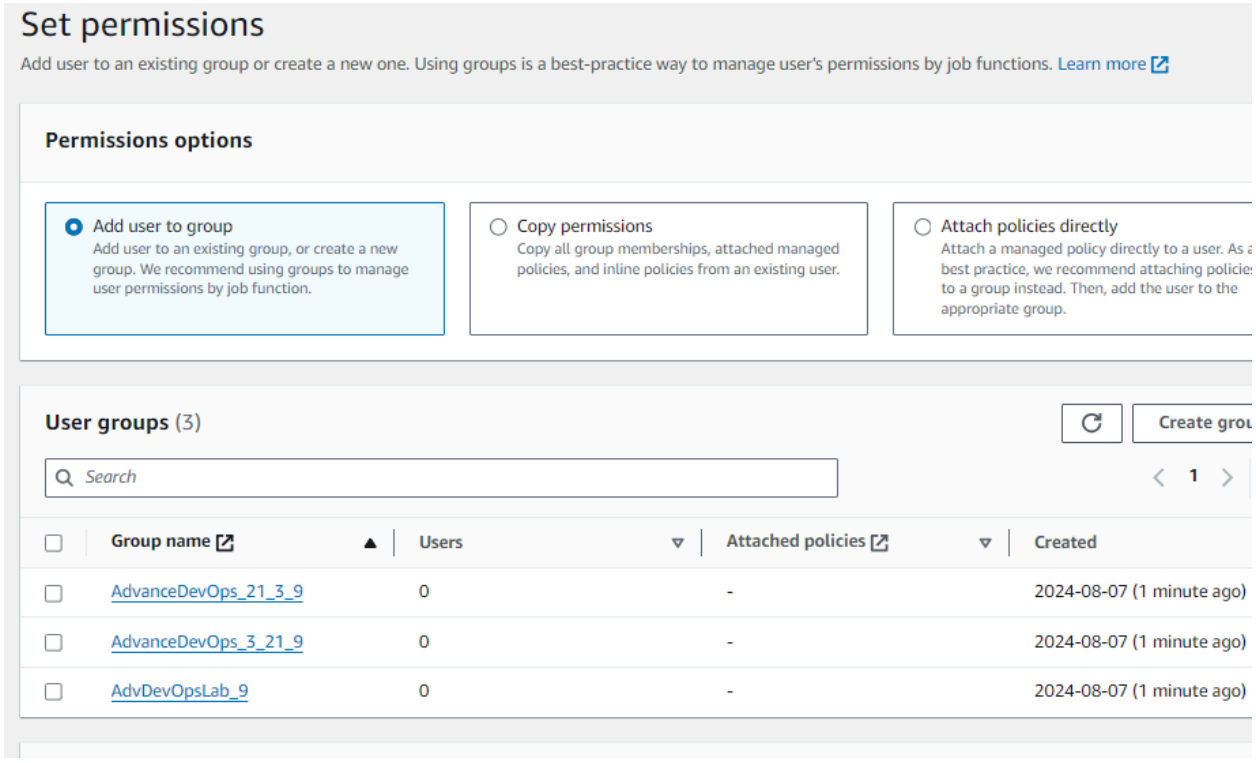
8. Set the Permissons ,click on create group

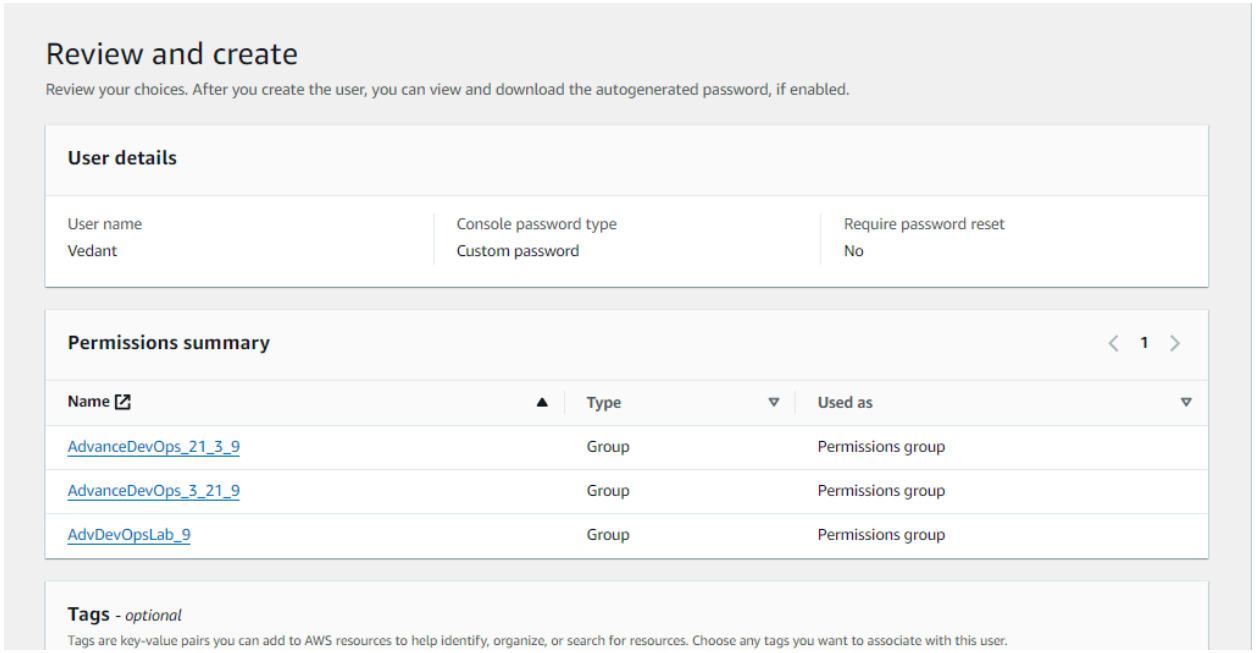


9. Provide group name and click on create user group.



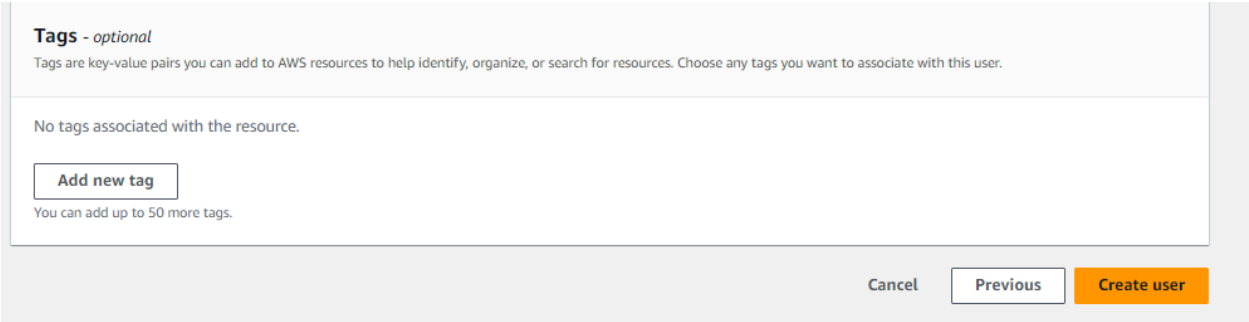
10. Navigate back to user groups



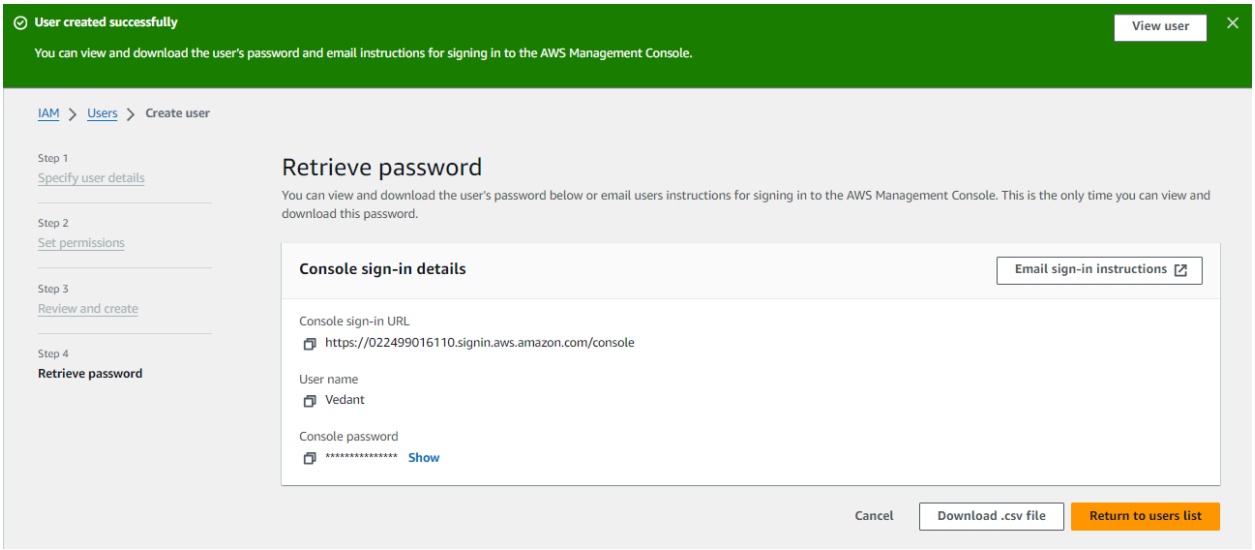



NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/9

11. Click on create user



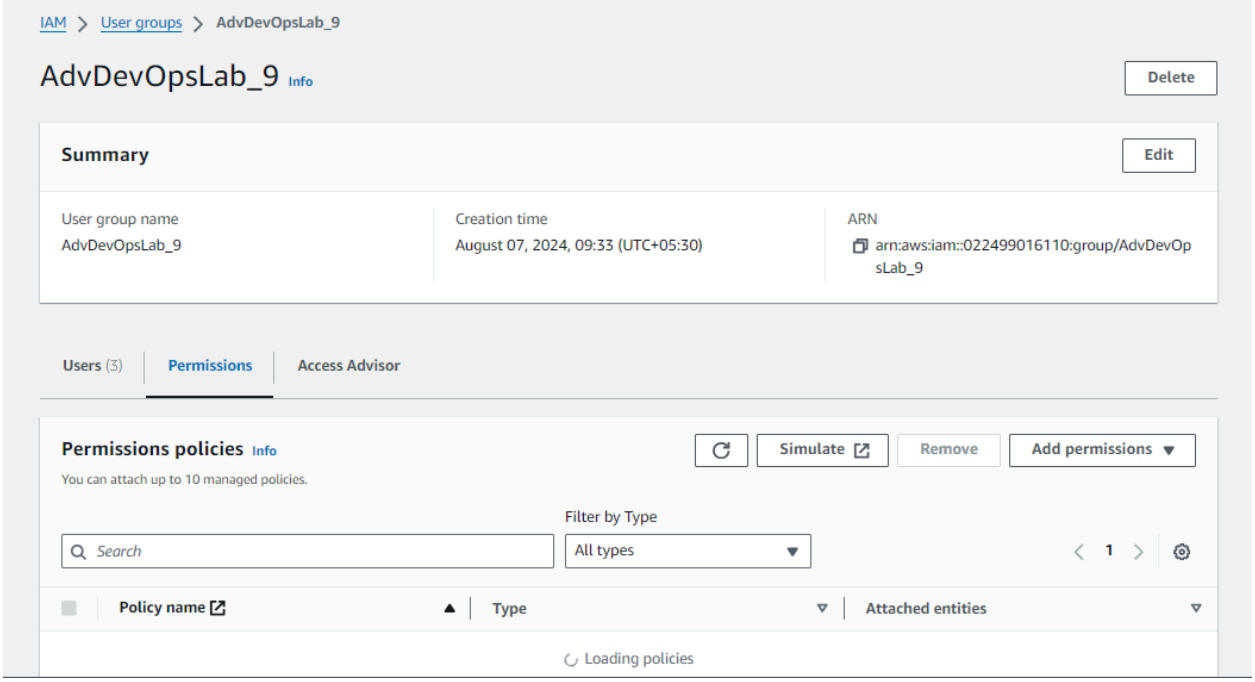

12. User is created Successfully



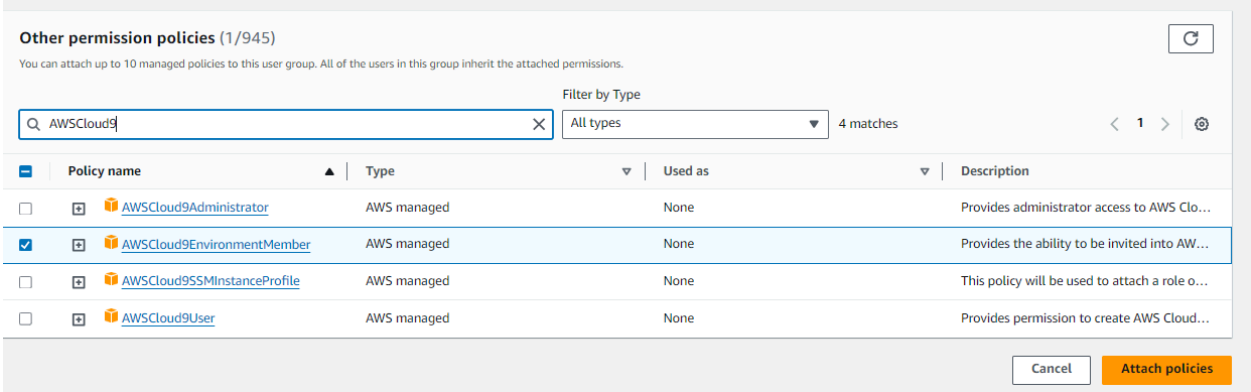

‘

NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/9

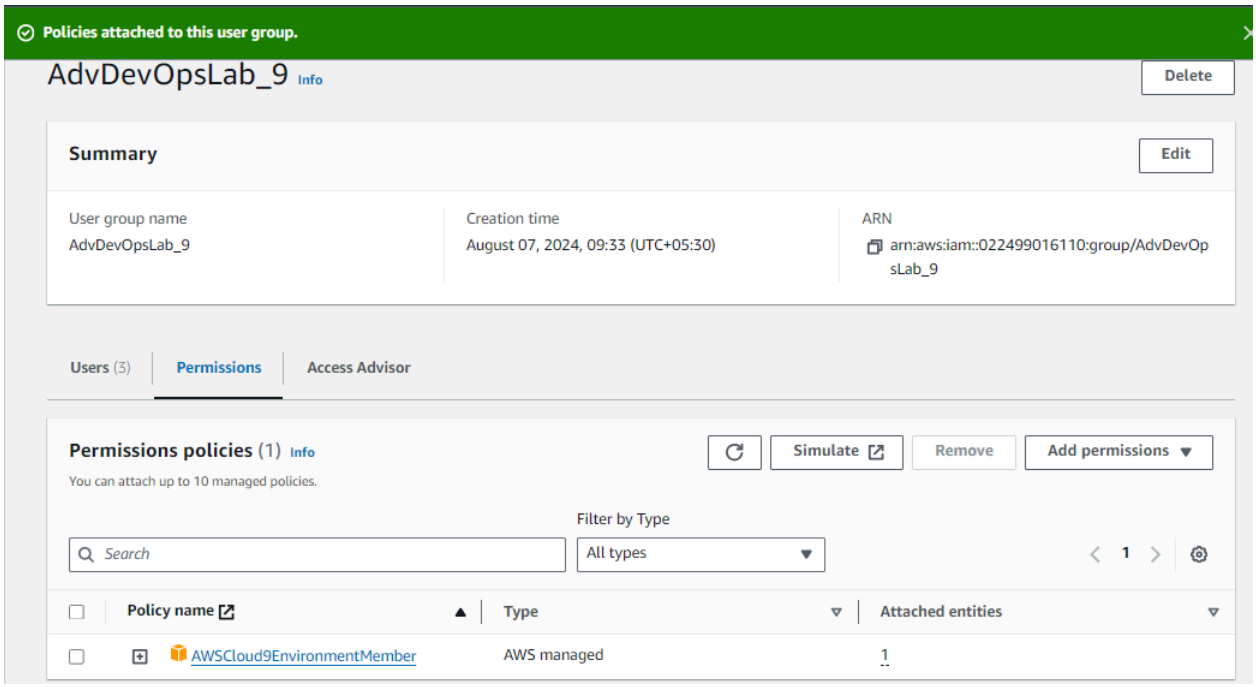
13.. Click on your group name which you have created and navigate to permission tab as shown:




14. Search AWSCloud9 policy and click on Attach policies.



15. We can see that Policy is attached Successfully.

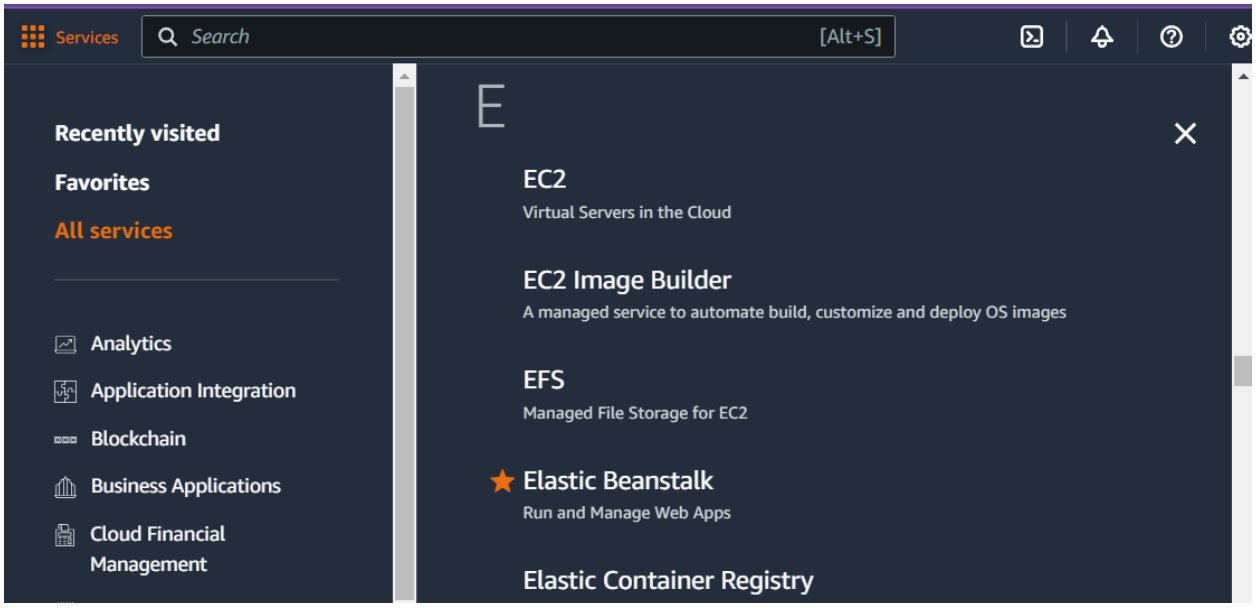


NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/9

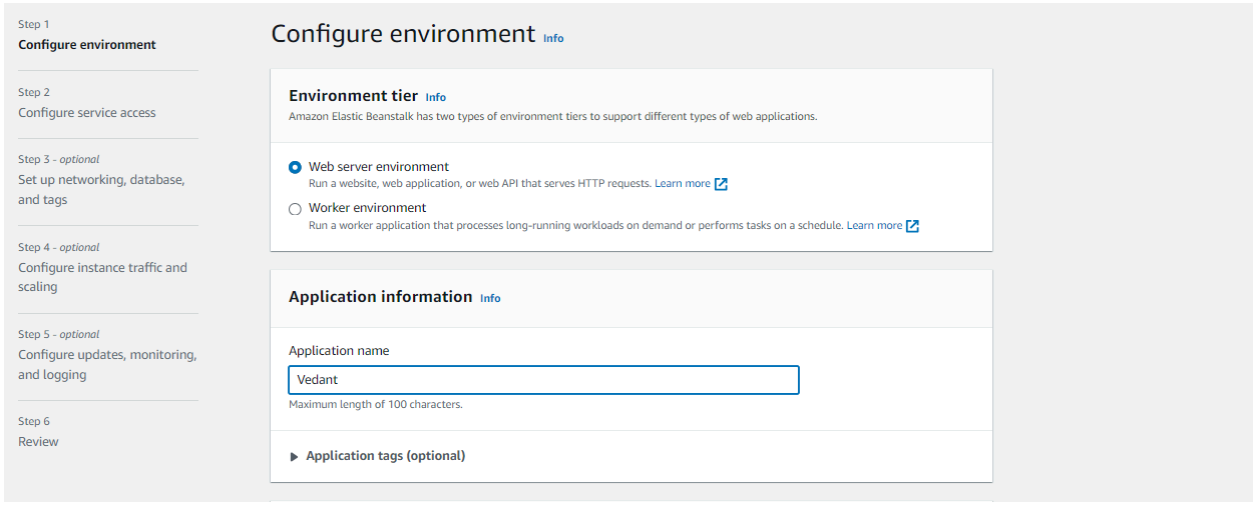
# AdvDevOps Exp 2

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

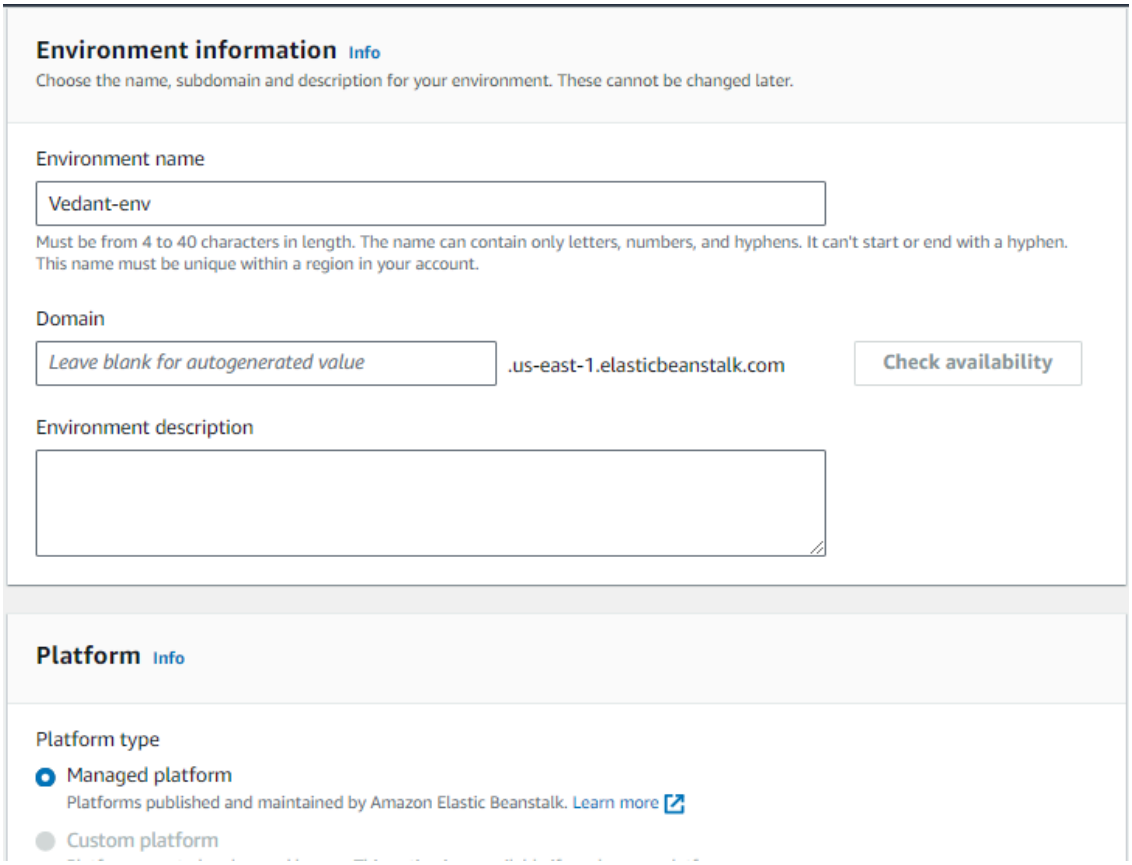
1. Login to your AWS account and search for Elastic Beanstalk in the search box.



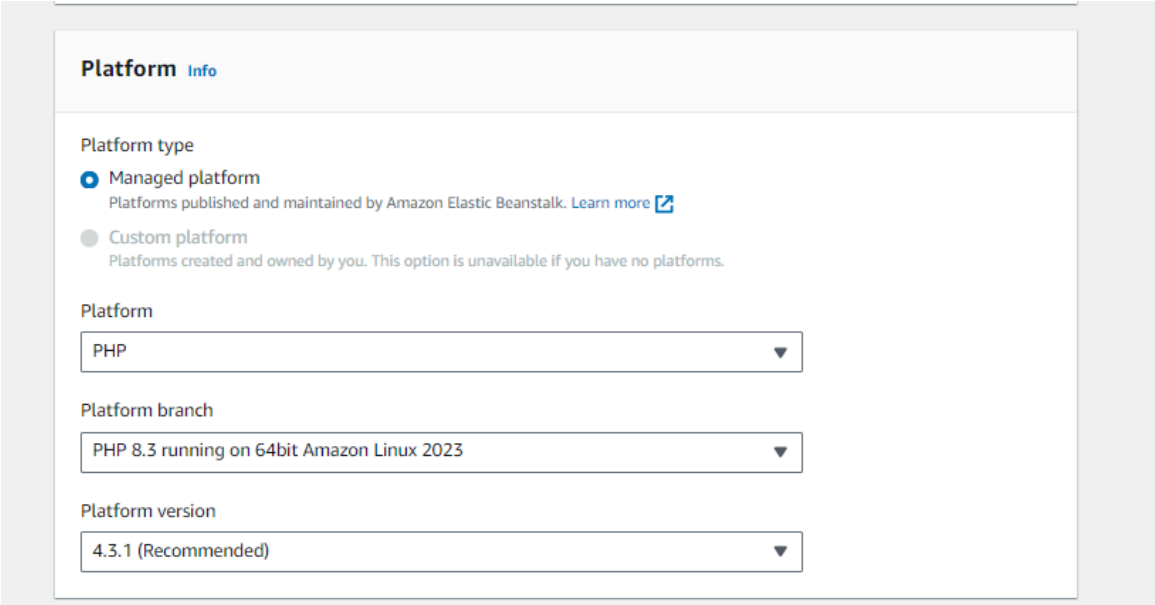

2. Open the Elastic Beanstalk and name your web app as shown below.



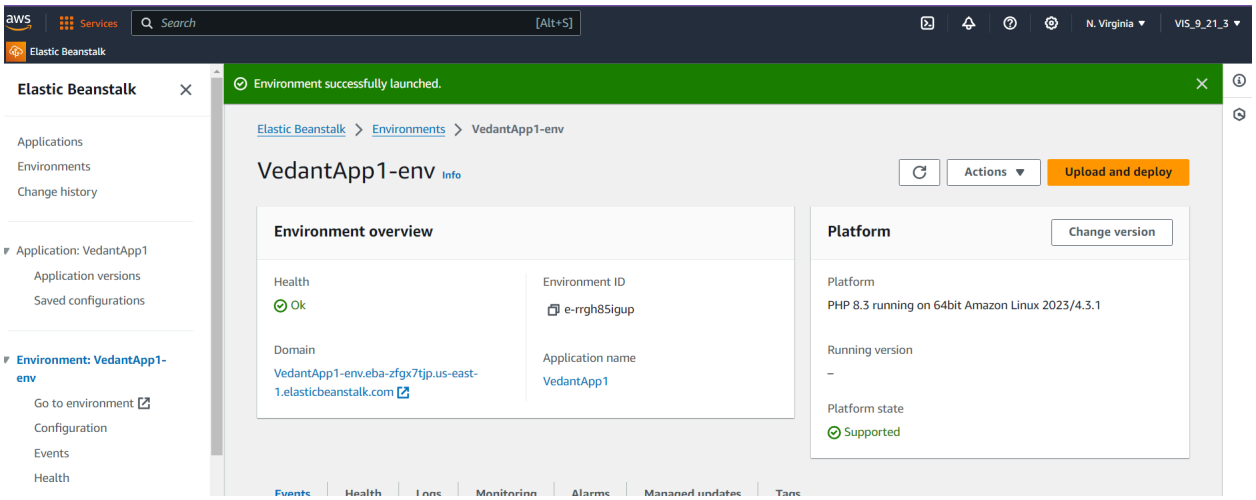
3. Automatically your environment name is set.




4. Choose PHP from the drop-down menu and then click Create Application.




5. We can see that our Environment is created and launched successfully

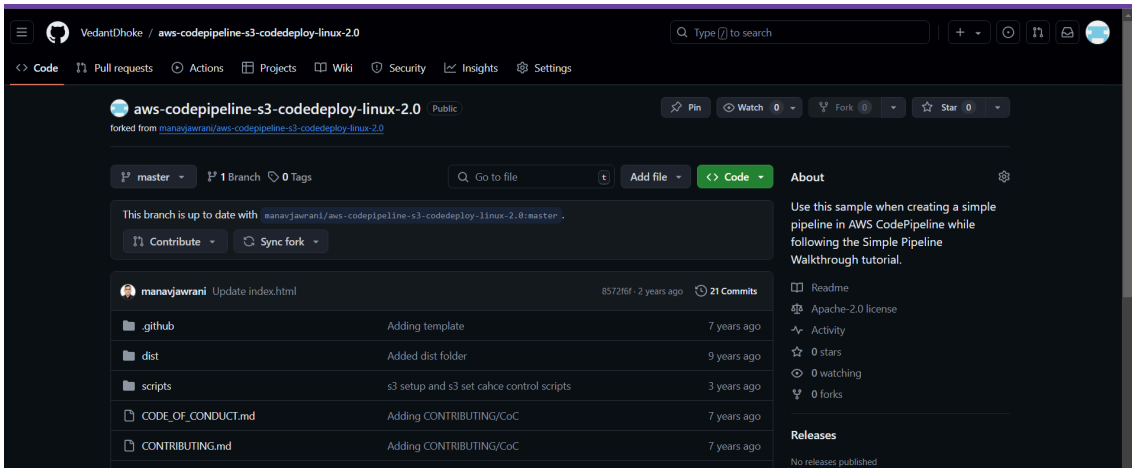



6. In this step, we'll obtain the sample code from a GitHub repository to host it later. The pipeline pulls the code from the source and executes specific actions.

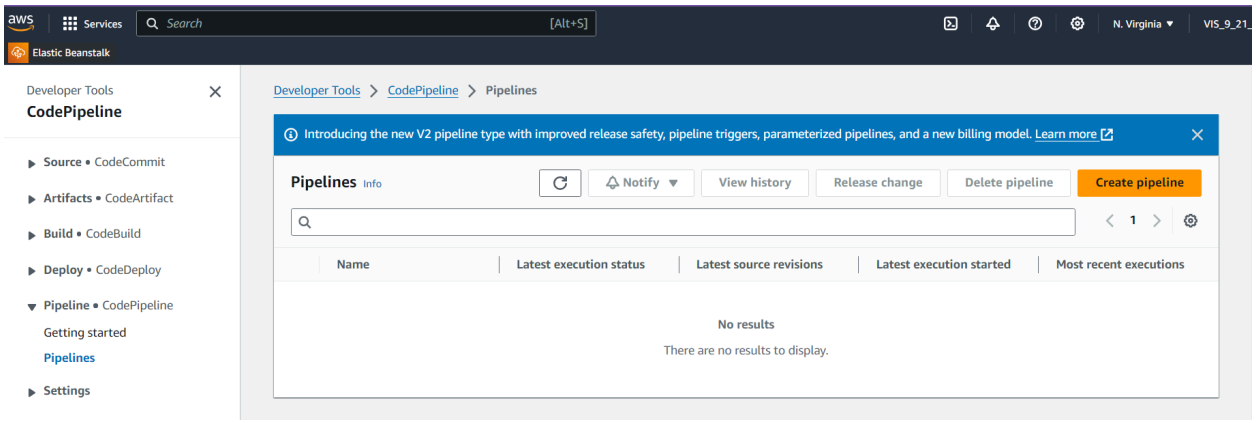
To get started, navigate to the provided GitHub repository and fork it to your account.r this experiment, we will use a forked GitHub repository as our source.

We can see that in VedantDhoke repository the provided GitHub repository is forked.

NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/9

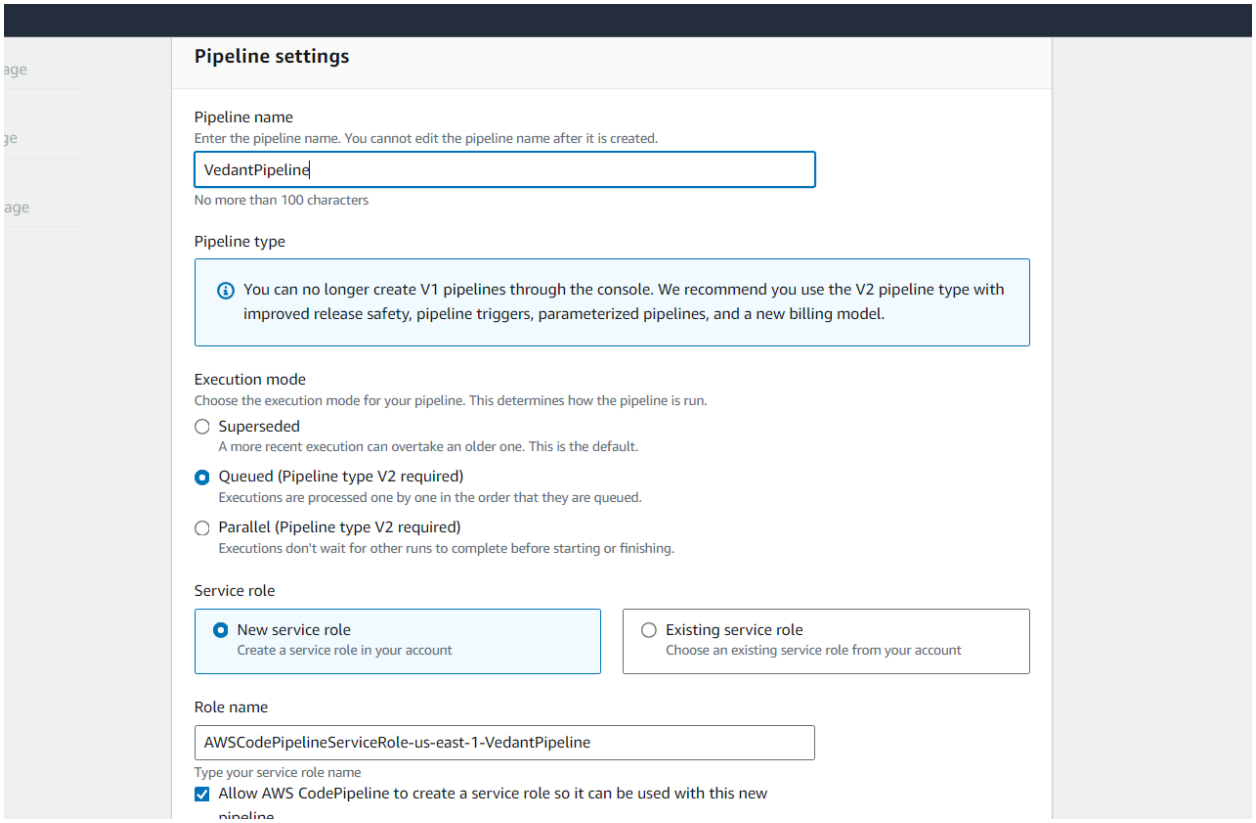



7. Now we want to create a Pipeline. Navigate to Developer Tools -> CodePipeline and
Click on create pipeline.

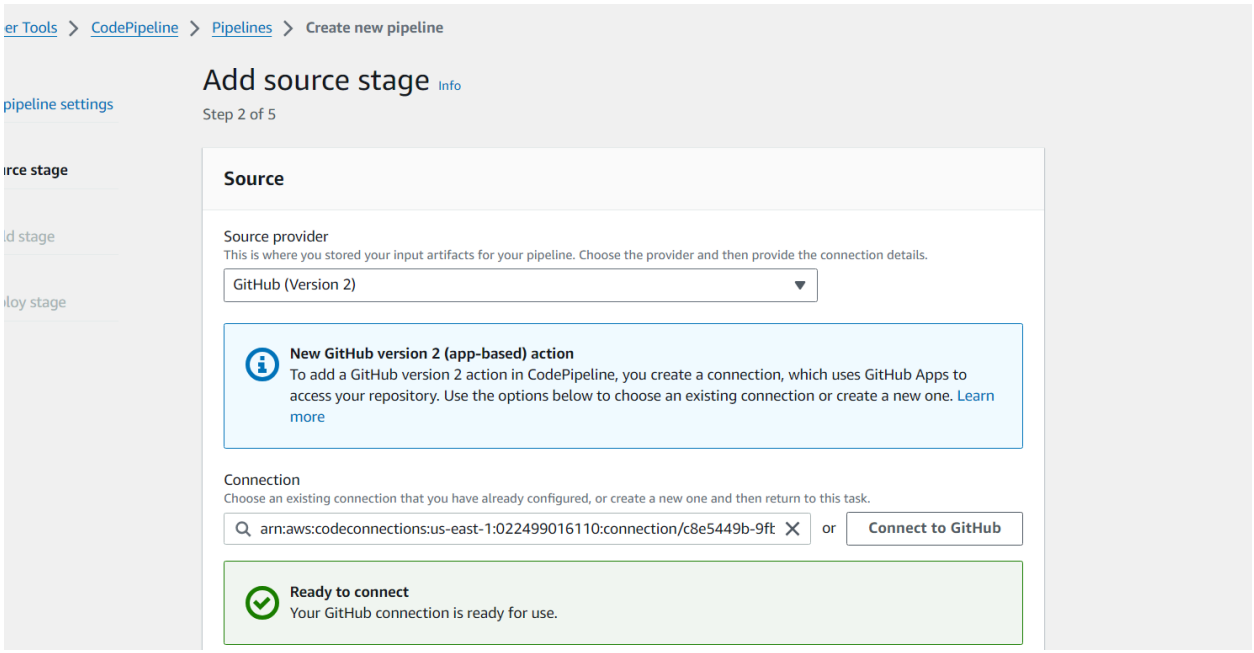


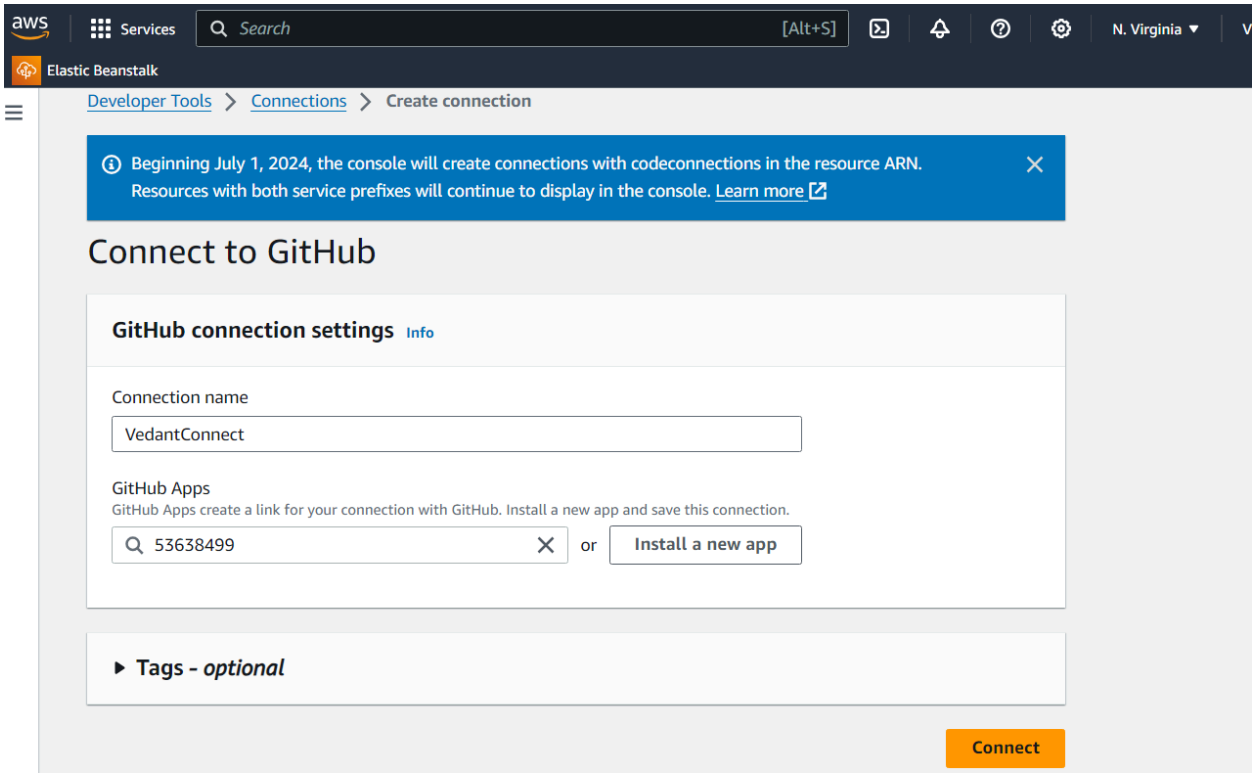
NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/9

8. Give name to your pipeline and do the settings in step 1.

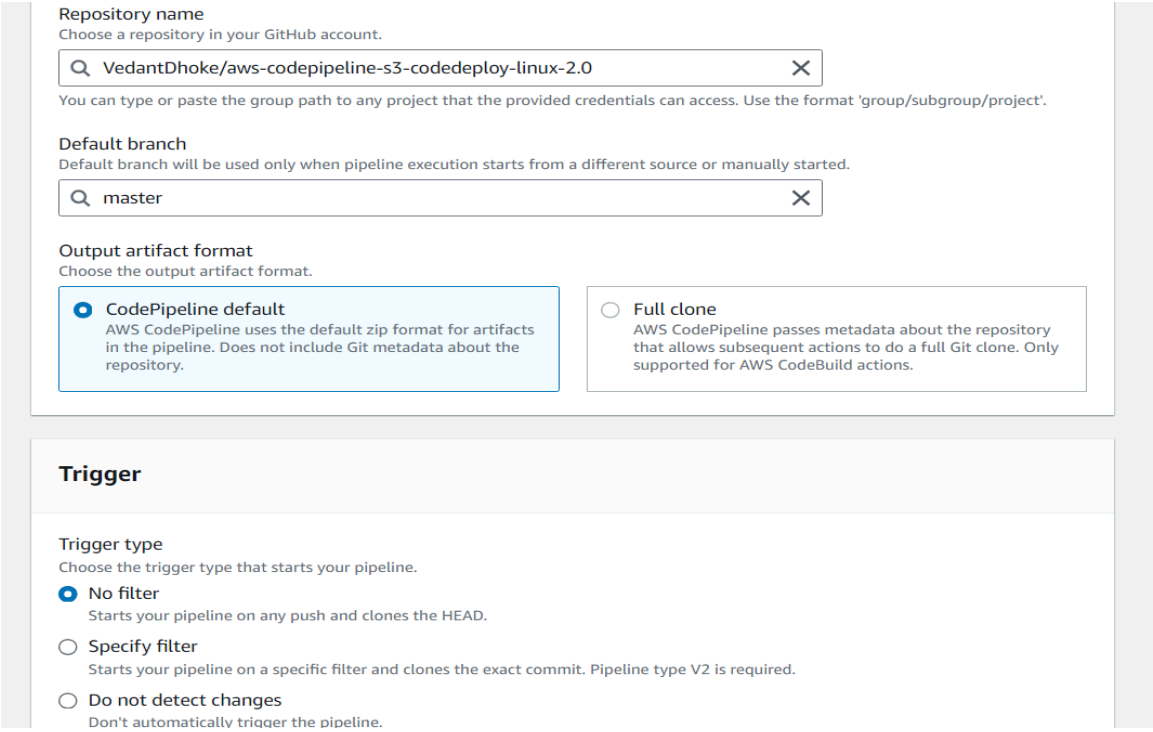



9. In the Source Stage, Choose GitHub(Version 2) and click on Connect to GitHub. You'll need your GitHub credentials to authorize and integrate AWS with your forked GitHub repository.

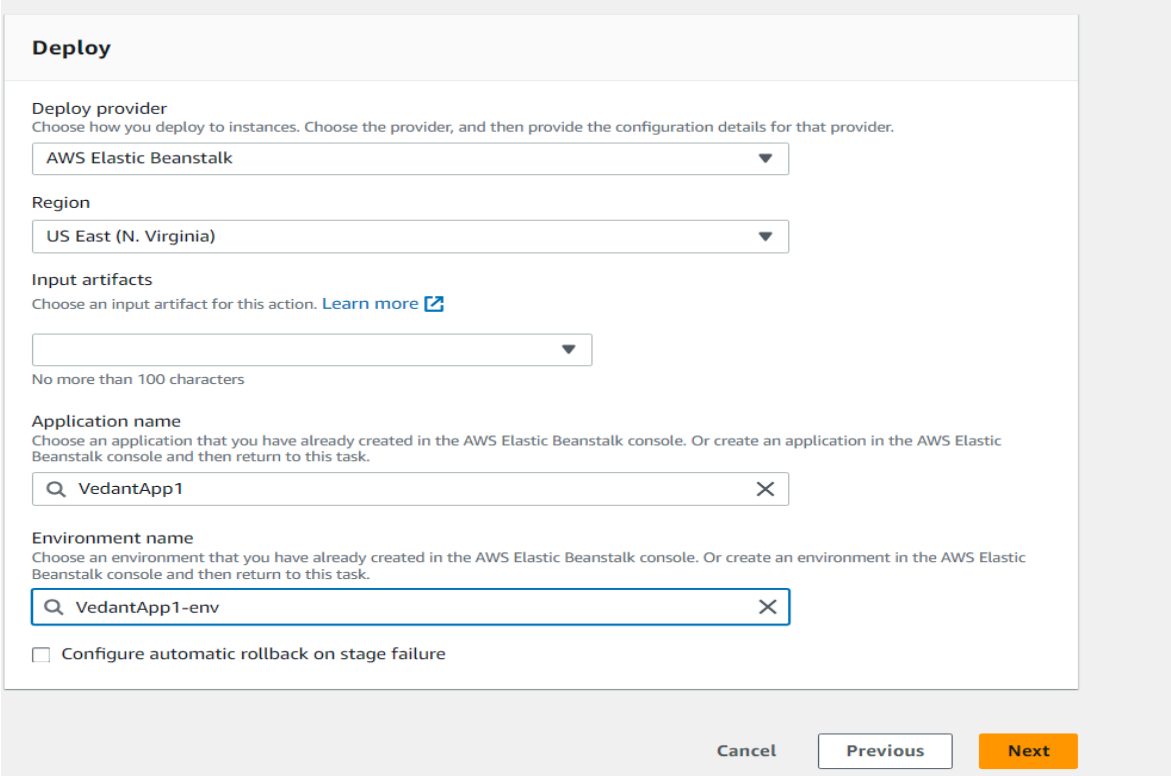



10.Next, select the forked repository and the appropriate branch from the search box. Click Continue, then skip the build stage and proceed directly to the Deployment stage.

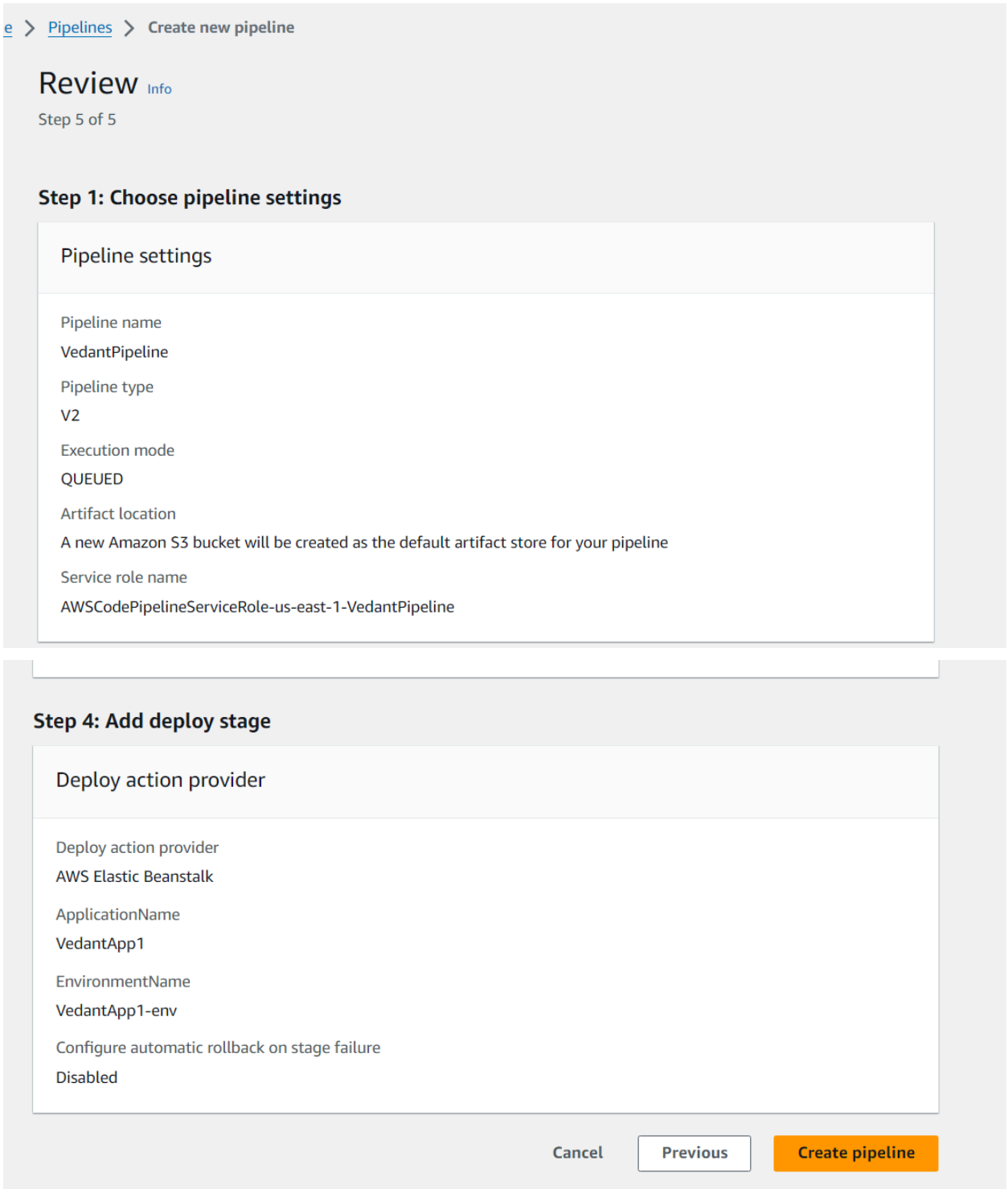



11. Select Deploy and specify the application name and environment name, then click "Next." Review the details and create the pipeline.



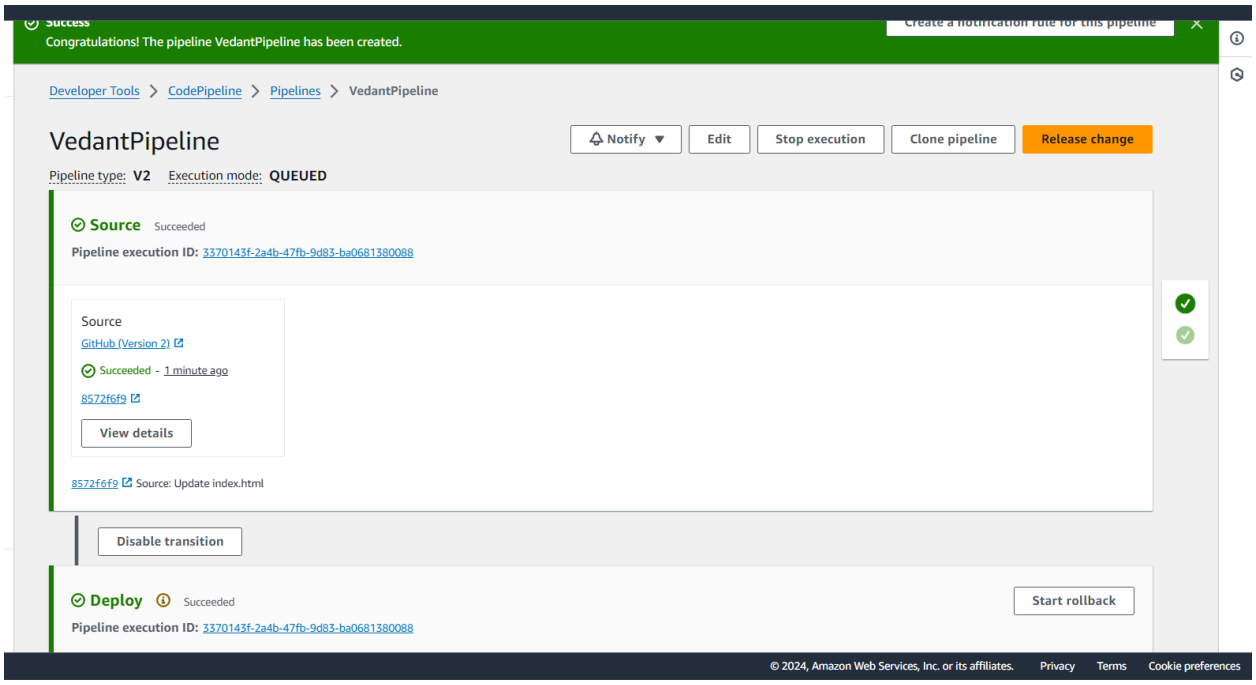

12. Review all the settings and click on create pipeline.

NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/9

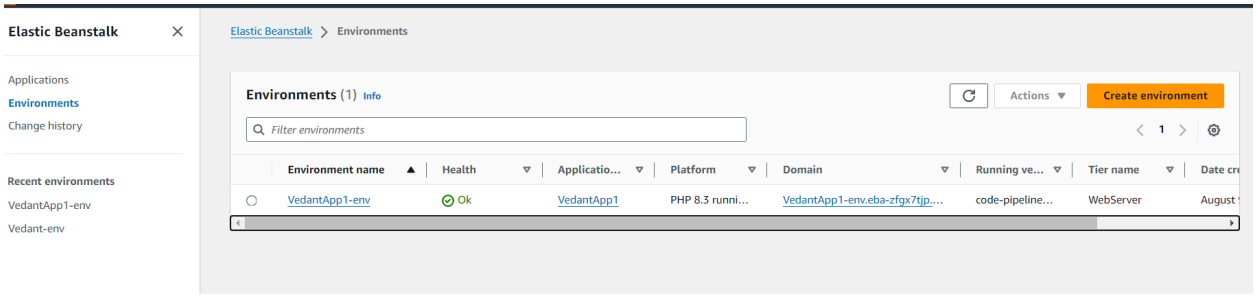


NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/9

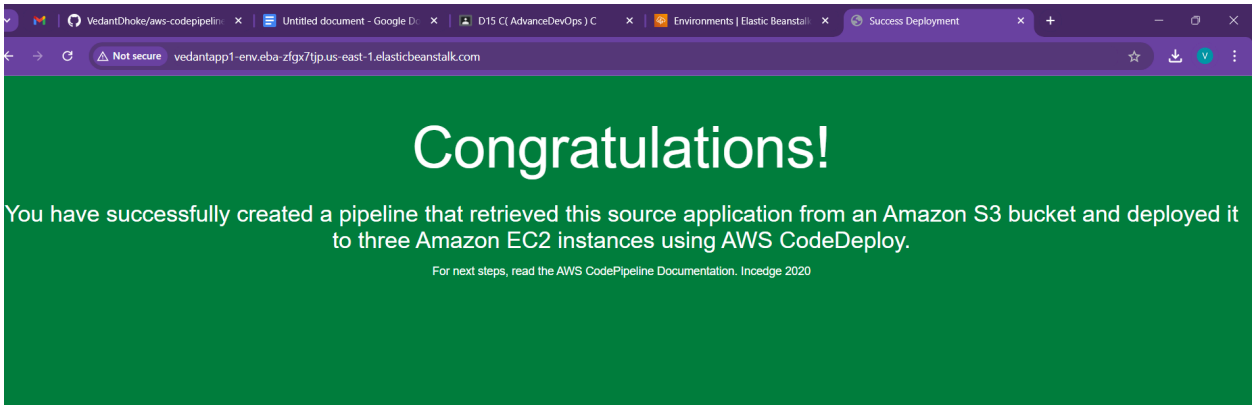
13. We can see our pipeline is successfully created. We need to ensure that both stages i.e Source and Deploy shows Succeeded as shown below.




14. Later Click on the URL which is provided in the Domain. It will navigate us to our website.




15. This is our sample website.



NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/9

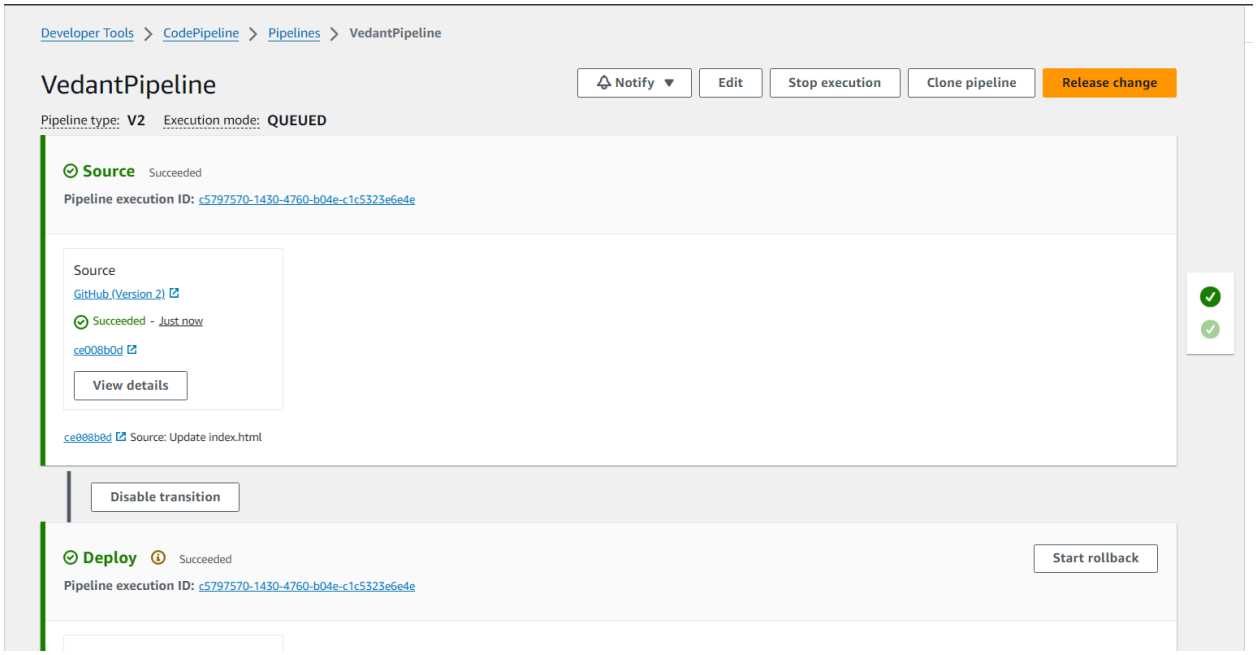
15. Committing the changes to update app .

1. Update the HTML file in your forked repository.

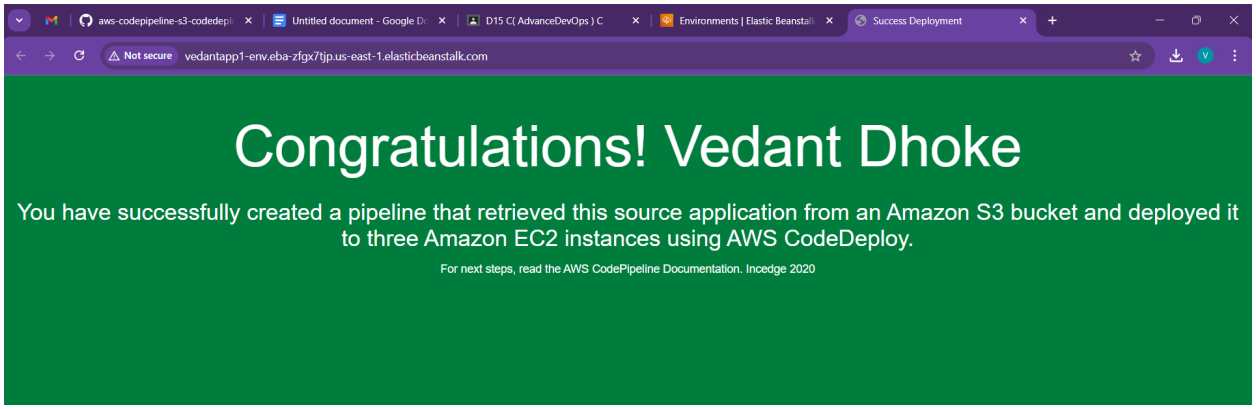
2. In GitHub, open index.html and modify the heading or paragraph tag. Commit the changes directly on GitHub.

3. Observe the real-time updates in the Source panel after committing.

4. Once the deployment is successful, view the changes on the website using the same URL.




16. Again click on the URL and we can see our Updated App.



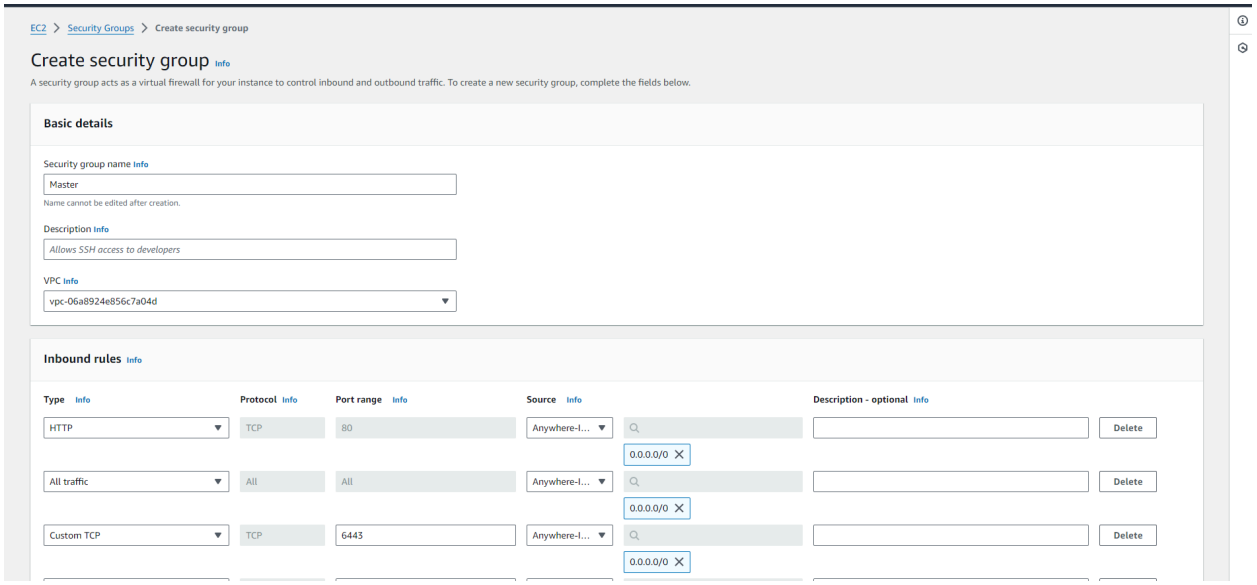
NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/ 9

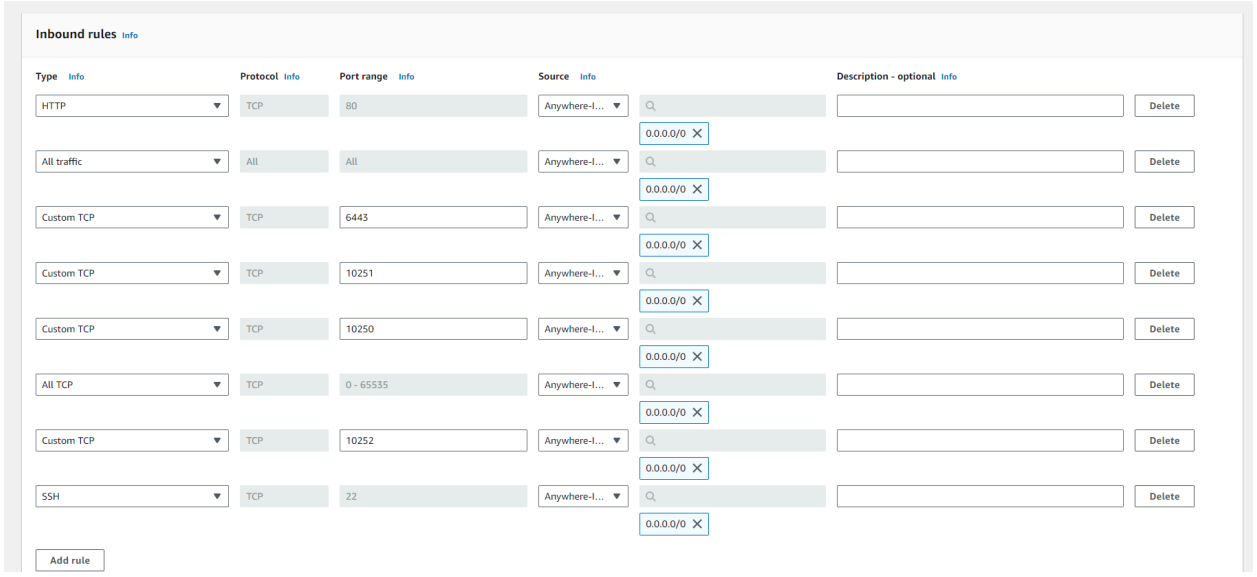
**EXPERIMENT NO: 3**

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

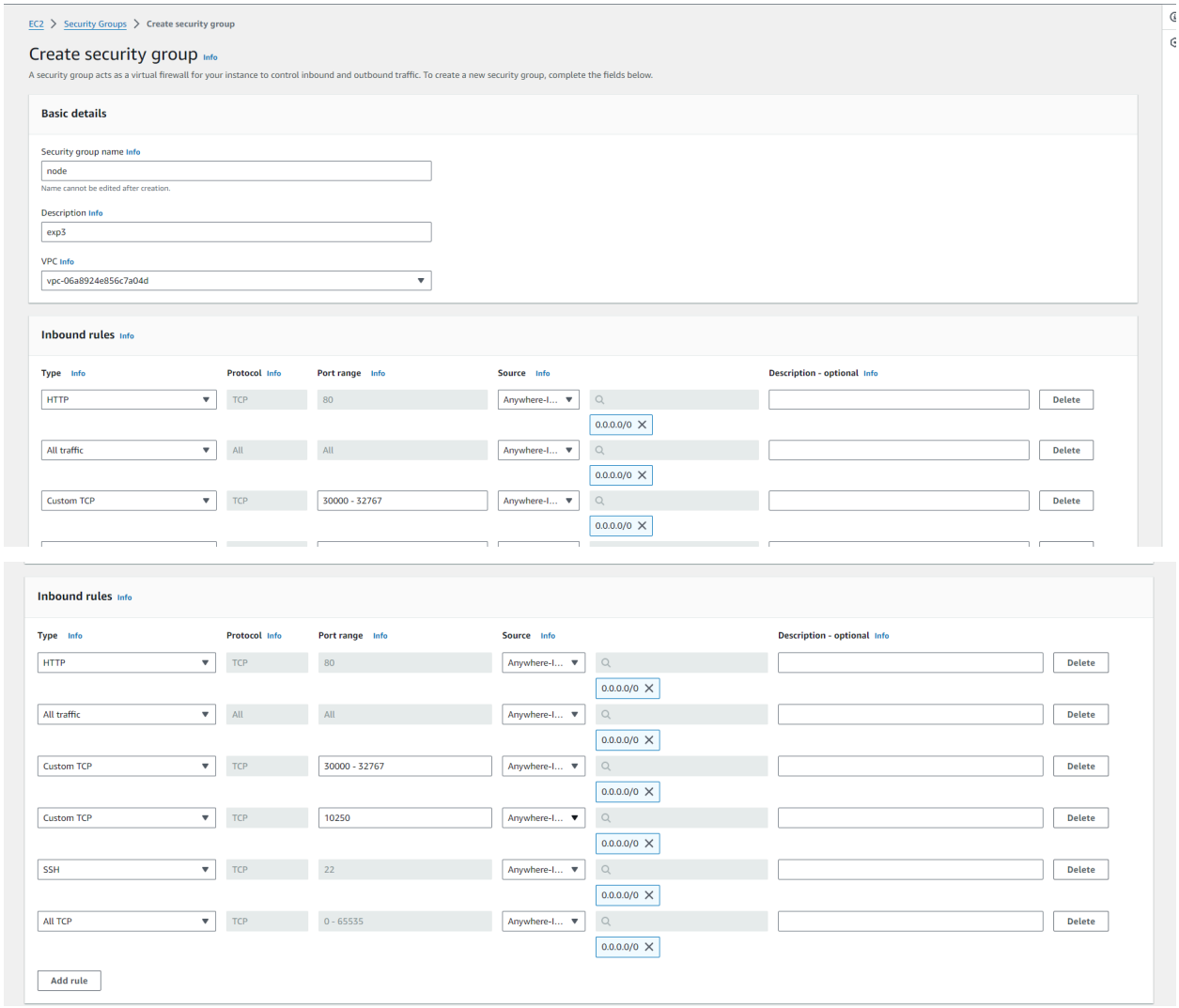
**STEPS :**

**1.Create Security Groups with the following inbound rules**



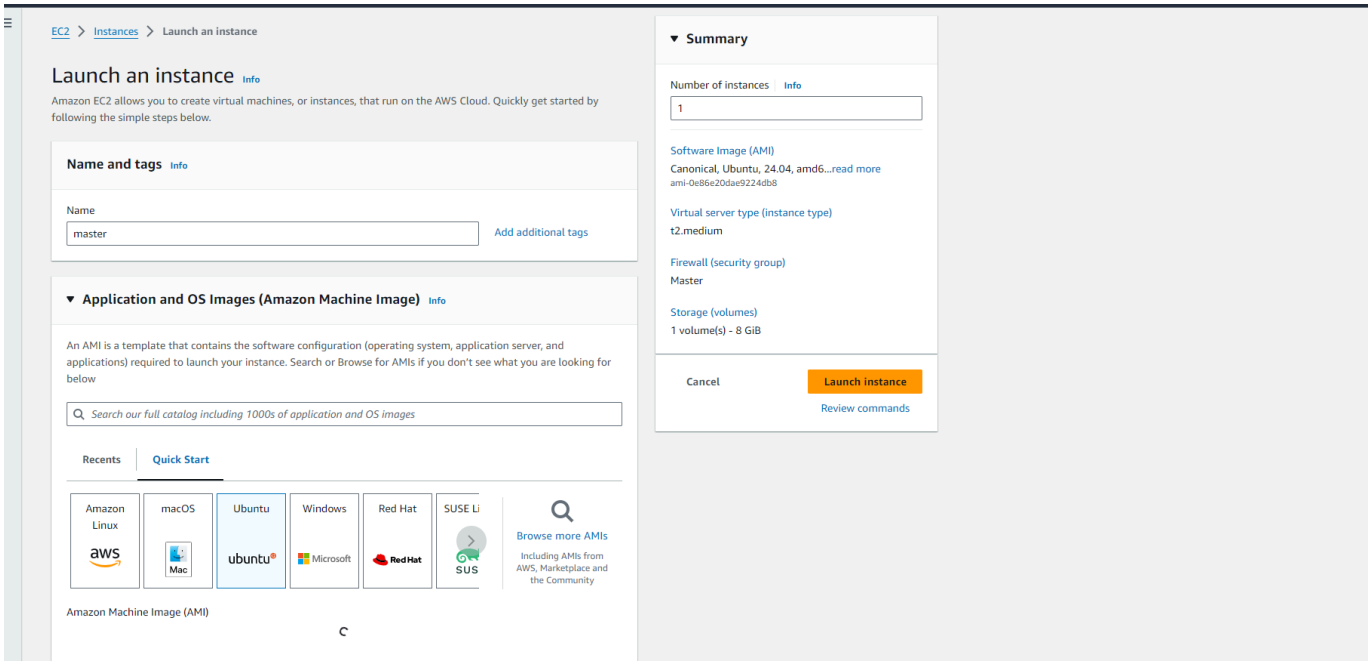



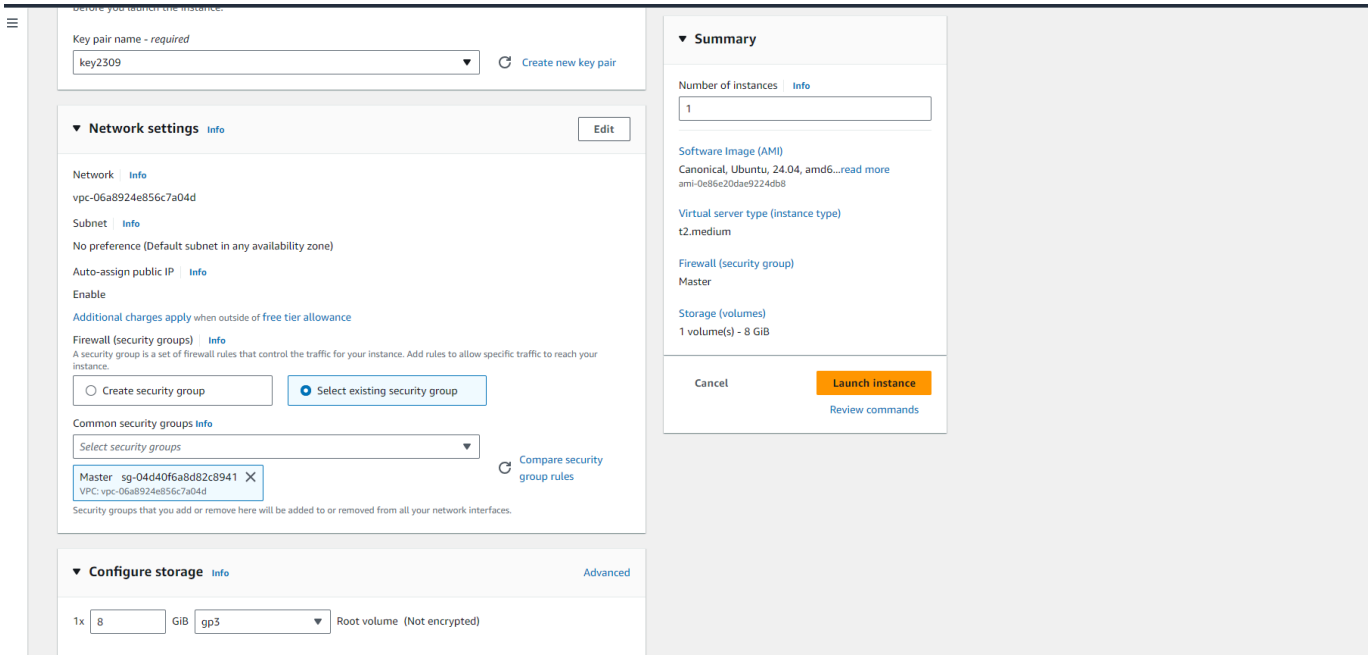
NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/ 9



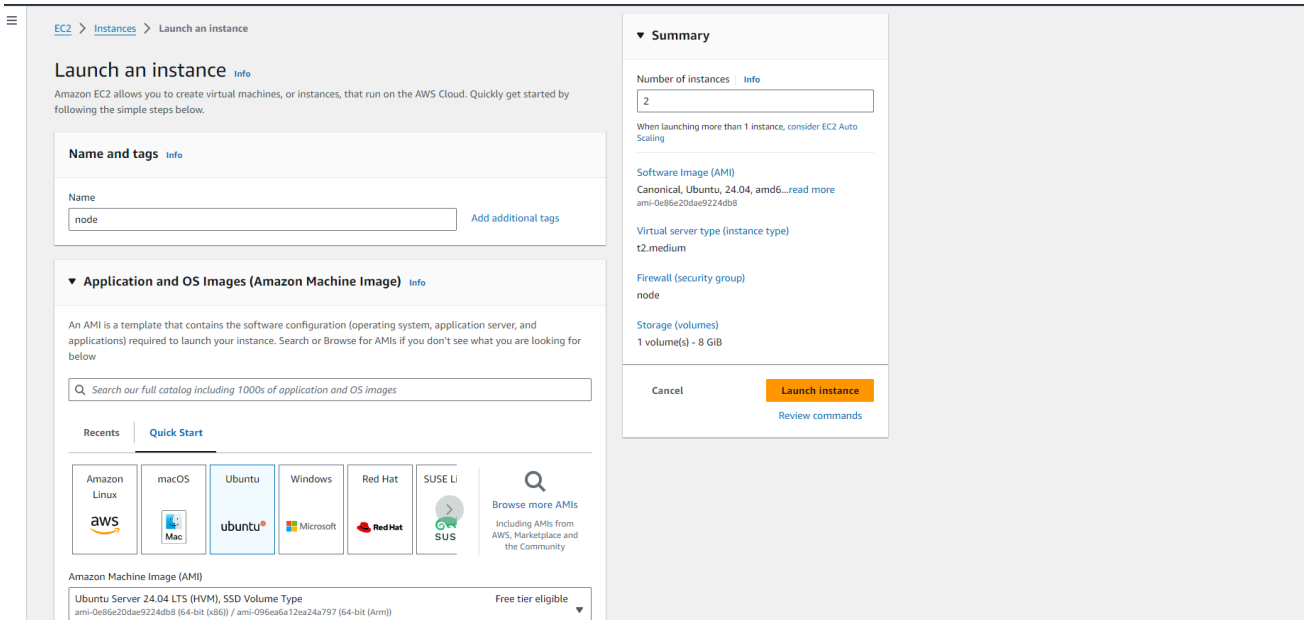
## 2.**Create Instances:**

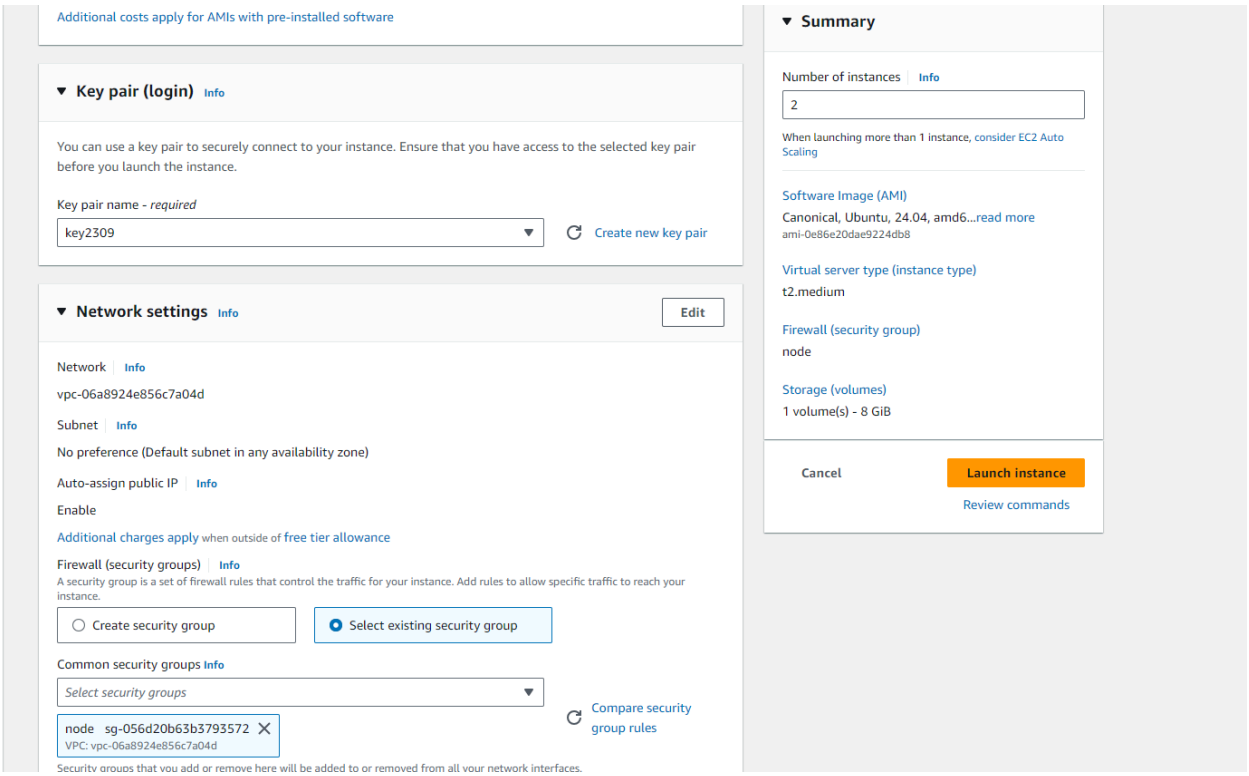
We initiated the creation of three virtual machines or instances, naming them Master, node-1, and node-2. These instances will act as the nodes in our Kubernetes cluster.



NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/ 9

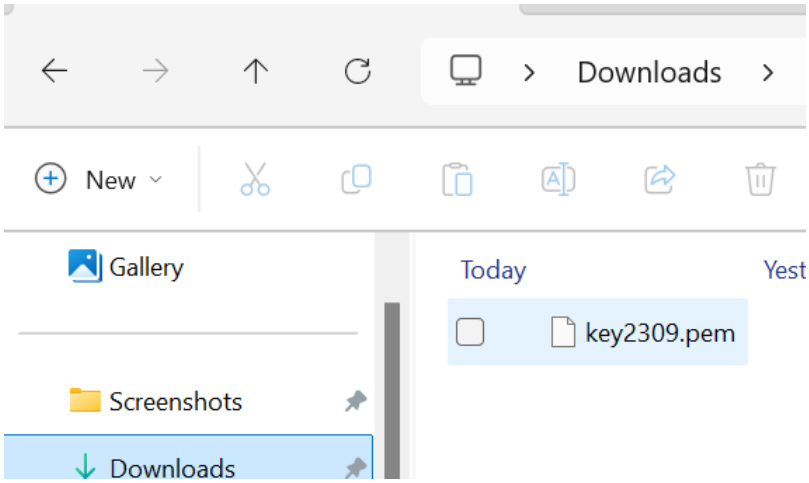


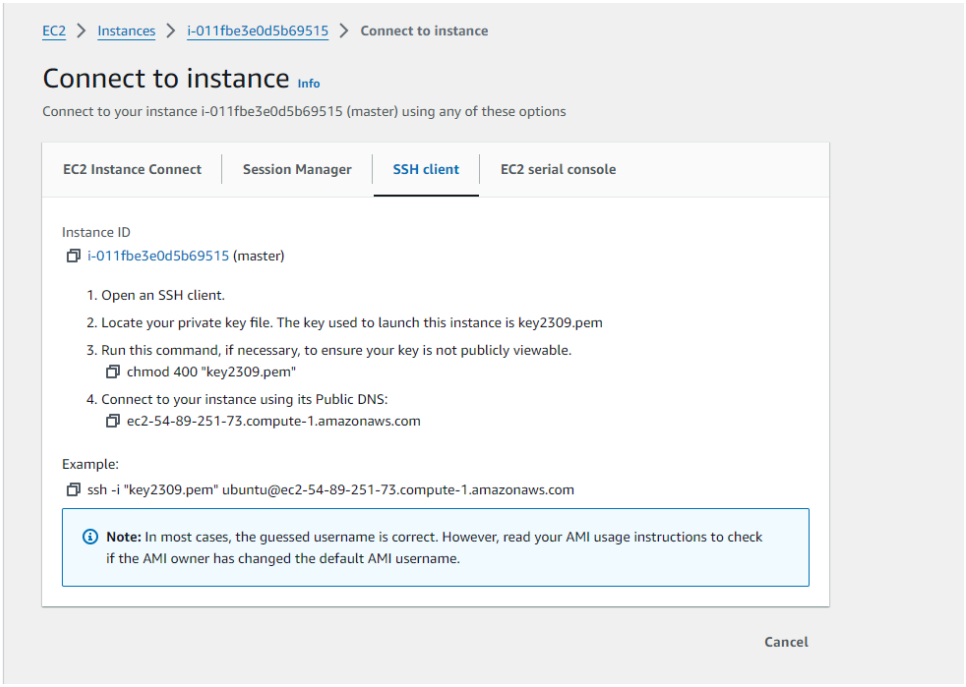

These the instances we have created successfully.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| ☐ | node-1 | i-019d881a62651c90d | Running | t2.medium | Initializing | View alarms + | us-east-1b | ec2-3-86-95-118.comp... | 3.86.95.118 |
| ☐ | node-2 | i-05333ee8d27d26a64 | Running | t2.medium | Initializing | View alarms + | us-east-1b | ec2-35-174-170-105.co... | 35.174.170.105 |
| ☐ | master | i-011fbe3e0d5b69515 | Running | t2.medium | Initializing | View alarms + | us-east-1b | ec2-54-89-251-73.com... | 54.89.251.73 |

It is important to have the download file of your key.




Copy the command below in the example part and paste it in the cmd.



2.**Install Docker**:

```
PS C:\Users\91900> ssh -i "key2309.pem" ubuntu@ec2-35-174-170-105.compute-1.amazonaws.com
The authenticity of host 'ec2-35-174-170-105.compute-1.amazonaws.com (35.174.170.105)' can't be established.
ED25519 key fingerprint is SHA256:777d6VJLUANXWdvC2Rqcqg6Jscu79S7iwHCjSgnMwM0.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-35-174-170-105.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

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

 System information as of Mon Sep 23 15:19:47 UTC 2024

  System load:  0.0               Processes:             112
  Usage of /:   22.8% of 6.71GB   Users logged in:       0
  Memory usage: 5%                IPv4 address for enX0: 172.31.86.235
  Swap usage:   0%
```

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-172-31-95-11:~$ 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
```

```
Get:51 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [208 B]
Get:52 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 c-n-f Metadata [344 B]
Fetched 29.1 MB in 5s (6401 kB/s)
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg)
, see the DEPRECATION section in apt-key(8) for details.
```

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

```
ubuntu@ip-172-31-95-11:~$ 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 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:5 https://download.docker.com/linux/ubuntu noble InRelease
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in l
, see the DEPRECATION section in apt-key(8) for details.
Reading package lists... Done
```

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

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

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

```
No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qe
```

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

```
ubuntu@ip-172-31-95-11:~$ 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-172-31-95-11:~$ 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
```

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-172-31-95-11:~$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyring
s/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/ap
t/sources.list.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
```

**Install Kubernetes components:**

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

```
ubuntu@ip-172-31-95-11:~$ sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 https://download.docker.com/linux/ubuntu noble InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  InRelease [1186 B]
Hit:6 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:7 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  Packages [4865 B]
Fetched 6051 B in 1s (11.5 kB/s)
```

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

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

```
ubuntu@ip-172-31-95-11:~$ sudo systemctl enable --now kubelet
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:
```

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

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

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

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

```
ubuntu@ip-172-31-95-11:~$ 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
```

```
[timeouts]
  "io.containerd.timeout.bolt.open" = "0s"
  "io.containerd.timeout.metrics.shimstats" = "2s"
  "io.containerd.timeout.shim.cleanup" = "5s"
  "io.containerd.timeout.shim.load" = "5s"
  "io.containerd.timeout.shim.shutdown" = "3s"
  "io.containerd.timeout.task.state" = "2s"

[ttrpc]
  address = ""
  gid = 0
  uid = 0
```

sudo systemctl restart containerd

```
ubuntu@ip-172-31-95-11:~$ 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 Mon 2024-09-23 15:35:29 UTC; 229ms ago
       Docs: https://containerd.io
   Main PID: 4614 (containerd)
      Tasks: 7
     Memory: 13.1M (peak: 14.4M)
        CPU: 64ms
     CGroup: /system.slice/containerd.service
             └─4614 /usr/bin/containerd

Sep 23 15:35:29 ip-172-31-95-11 containerd[4614]: time="2024-09-23T15:35:29.093975568Z" level=info msg=serving... address=/run/conta>
Sep 23 15:35:29 ip-172-31-95-11 containerd[4614]: time="2024-09-23T15:35:29.094023950Z" level=info msg=serving... address=/run/conta>
Sep 23 15:35:29 ip-172-31-95-11 containerd[4614]: time="2024-09-23T15:35:29.094053439Z" level=info msg="Start subscribing containerd>
Sep 23 15:35:29 ip-172-31-95-11 containerd[4614]: time="2024-09-23T15:35:29.094080493Z" level=info msg="Start recovering state"
Sep 23 15:35:29 ip-172-31-95-11 containerd[4614]: time="2024-09-23T15:35:29.094120880Z" level=info msg="Start event monitor"
Sep 23 15:35:29 ip-172-31-95-11 containerd[4614]: time="2024-09-23T15:35:29.094128980Z" level=info msg="Start snapshots syncer"
Sep 23 15:35:29 ip-172-31-95-11 containerd[4614]: time="2024-09-23T15:35:29.094137306Z" level=info msg="Start cni network conf synce>
Sep 23 15:35:29 ip-172-31-95-11 containerd[4614]: time="2024-09-23T15:35:29.094142936Z" level=info msg="Start streaming server"
Sep 23 15:35:29 ip-172-31-95-11 containerd[4614]: time="2024-09-23T15:35:29.094178857Z" level=info msg="containerd successfully boot>
Sep 23 15:35:29 ip-172-31-95-11 systemd[1]: Started containerd.service - containerd container runtime.
lines 1-21/21 (END)
```

```
ubuntu@ip-172-31-95-11:~$ 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:
```

```
Preparing to unpack .../socat_1.8.0.0-4build3_amd64.deb ...
Unpacking socat (1.8.0.0-4build3) ...
Setting up socat (1.8.0.0-4build3) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

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

**6.Initialize kubeadm on Master Node:**

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

On the Master node, initialize the Kubernetes cluster using kubeadm. This process sets up the Kubernetes control plane and generates commands for joining worker nodes:
Copy the commands displayed in the output of the initialization process to configure permissions and obtain the join token. This includes a join command link needed for worker nodes to connect to the master.

```
ubuntu@ip-172-31-95-11:~$ 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'
W0923 15:48:50.266640    5352 checks.go:846] detected that the sandbox image "registry.k8s.io/pause:3.8" of the container runtime is
inconsistent with that used by kubeadm.It is recommended to use "registry.k8s.io/pause:3.10" as the CRI sandbox image.
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
```

```
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 172.31.95.11:6443 --token 16q9ib.pv09d9wixd10d50s \
        --discovery-token-ca-cert-hash sha256:7ec275faa3af2297d3e809dea0a29175b2d2af6db63d2673456039bdba306fc6
ubuntu@ip-172-31-95-11:~$
```

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

```
ubuntu@ip-172-31-95-11:~$ mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

```
ubuntu@ip-172-31-95-11:~$ kubectl get nodes
NAME              STATUS     ROLES           AGE     VERSION
ip-172-31-95-11   NotReady   control-plane   2m50s   v1.31.1
```

**7.Join Worker Nodes:**

Now Run the following command on Node 1 and Node 2 to Join to master.

sudo kubeadm join 172.31.27.176:6443 --token ttay2x.n0sqeukjai8sgfg3 \

--discovery-token-ca-cert-hash

sha256:d6fc5fb7e984c83e2807780047fec6c4f2acfe9da9184ecc028d77157608fbb6

```
ubuntu@ip-172-31-86-235:~$ sudo kubeadm join 172.31.95.11:6443 --token 16q9ib.pv09d9wixd10d50s \
        --discovery-token-ca-cert-hash sha256:7ec275faa3af2297d3e809dea0a29175b2d2af6db63d2673456039bdba306fc6
[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 501.238074ms
[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.
```

```
ubuntu@ip-172-31-95-11:~$ kubectl get nodes
NAME               STATUS     ROLES           AGE     VERSION
ip-172-31-86-235   NotReady   <none>          70s     v1.31.1
ip-172-31-91-211   NotReady   <none>          61s     v1.31.1
ip-172-31-95-11    NotReady   control-plane   5m25s   v1.31.1
ubuntu@ip-172-31-95-11:~$ 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
```

```
ubuntu@ip-172-31-95-11:~$ kubectl get nodes -o wide
NAME               STATUS   ROLES           AGE     VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE           KERNEL-VERSION   CONTA
INER-RUNTIME
ip-172-31-86-235   Ready    <none>          3m44s   v1.31.1   172.31.86.235   <none>        Ubuntu 24.04 LTS   6.8.0-1012-aws   conta
inerd://1.7.12
ip-172-31-91-211   Ready    <none>          3m35s   v1.31.1   172.31.91.211   <none>        Ubuntu 24.04 LTS   6.8.0-1012-aws   conta
inerd://1.7.12
ip-172-31-95-11    Ready    control-plane   7m59s   v1.31.1   172.31.95.11    <none>        Ubuntu 24.04 LTS   6.8.0-1012-aws   conta
inerd://1.7.12
```

```
ubuntu@ip-172-31-95-11:~$ kubectl label node ip-172-31-91-211 kubernetes.io/role=Node1
node/ip-172-31-91-211 labeled
```

```
ubuntu@ip-172-31-95-11:~$ kubectl label node ip-172-31-95-11 kubernetes.io/role=worker
node/ip-172-31-95-11 labeled
```

```
ubuntu@ip-172-31-95-11:~$ kubectl label node ip-172-31-86-235 kubernetes.io/role=Node2
error: 'kubernetes.io/role' already has a value (Node1), and --overwrite is false
```

```
ubuntu@ip-172-31-95-11:~$ kubectl label node ip-172-31-86-235 kubernetes.io/role=Node2 --overwrite
node/ip-172-31-86-235 labeled
```

```
ubuntu@ip-172-31-95-11:~$ kubectl get nodes -o wide
NAME               STATUS   ROLES                  AGE     VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE           KERNEL-VERSION
  CONTAINER-RUNTIME
ip-172-31-86-235   Ready    Node2                  9m29s   v1.31.1   172.31.86.235   <none>        Ubuntu 24.04 LTS   6.8.0-1012-aws
  containerd://1.7.12
ip-172-31-91-211   Ready    Node1                  9m20s   v1.31.1   172.31.91.211   <none>        Ubuntu 24.04 LTS   6.8.0-1012-aws
  containerd://1.7.12
ip-172-31-95-11    Ready    control-plane,worker   13m     v1.31.1   172.31.95.11    <none>        Ubuntu 24.04 LTS   6.8.0-1012-aws
  containerd://1.7.12
ubuntu@ip-172-31-95-11:~$
```

**CONCLUSION :**

**Docker installation:** After installing docker on all instances, sometimes docker services may fail to restart.

**Network Configuration Issue:** Connectivity issue between master and worker modes might be caused by the firewall blocking that required ports.

**CrashLoopBackOff**: There are errors indicating that the containers for Kubernetes components are restarting repeatedly but failing to start properly.

.

# AdvanceDevops Experiment 4

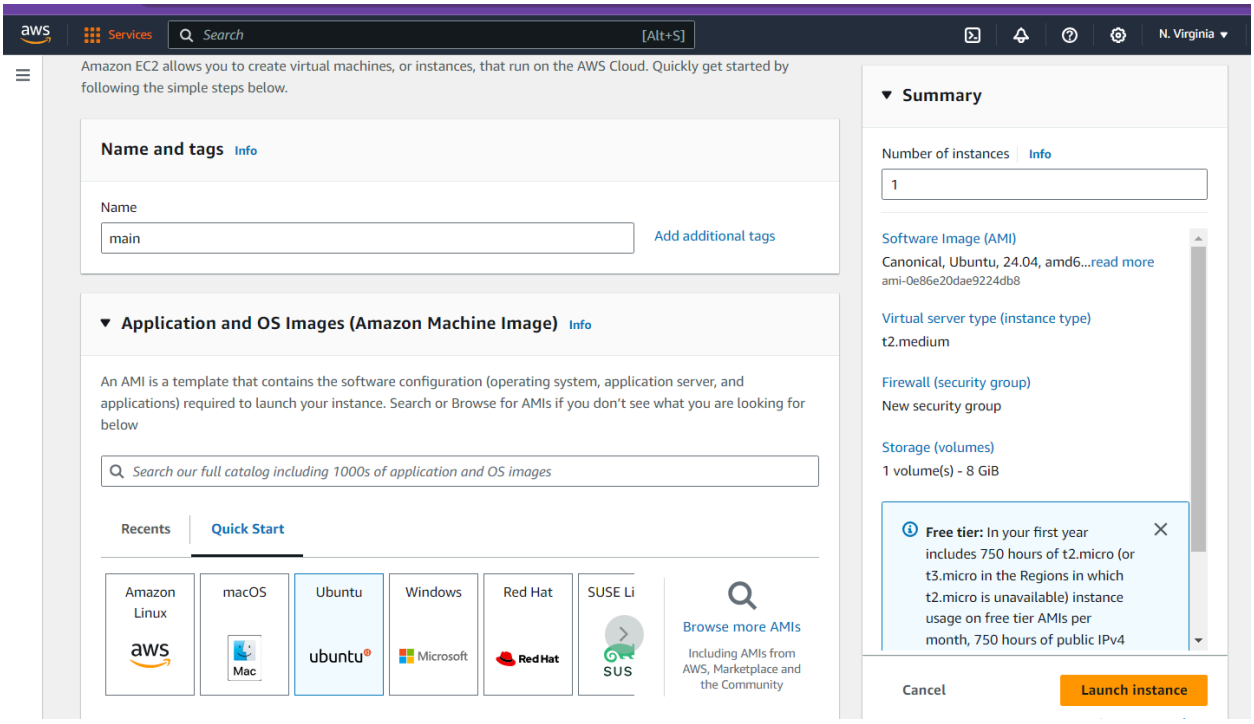
**Aim:** To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deploy Your First Kubernetes Application.

## Theory:

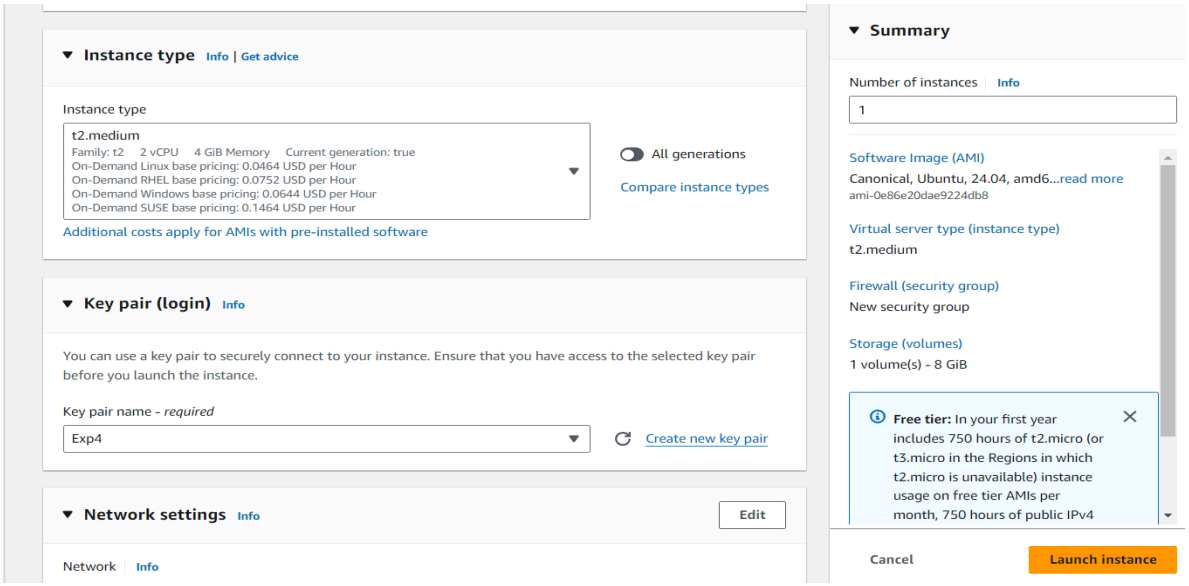
**Kubernetes**, originally developed by Google, is an open-source container orchestration platform. It automates the deployment, scaling, and management of containerized applications, ensuring high availability and fault tolerance. Kubernetes is now the industry standard for container orchestration and is governed by the **Cloud Native Computing Foundation (CNCF)**, with contributions from major cloud and software providers like Google, AWS, Microsoft, IBM, Intel, Cisco, and Red Hat.

**Kubernetes Deployment:** Is a resource in Kubernetes that provides declarative updates for Pods and ReplicaSets. With a Deployment, you can define how many replicas of a pod should run, roll out new versions of an application, and roll back to previous versions if necessary. It ensures that the desired number of pod replicas are running at all times.

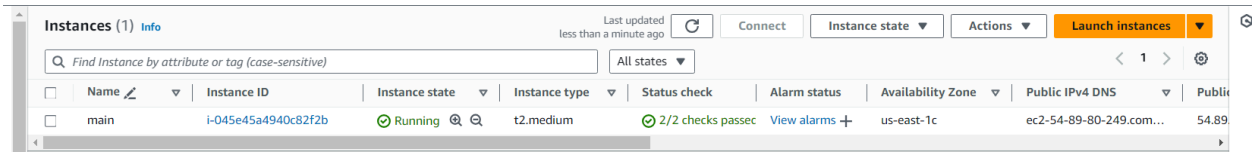
**Step 1:** Log in to your AWS Academy/personal account and launch a new Ec2 Instance. Select Ubuntu as AMI and t2.medium as Instance Type, create a key of type RSA with .pem extension, and move the downloaded key to the new folder.

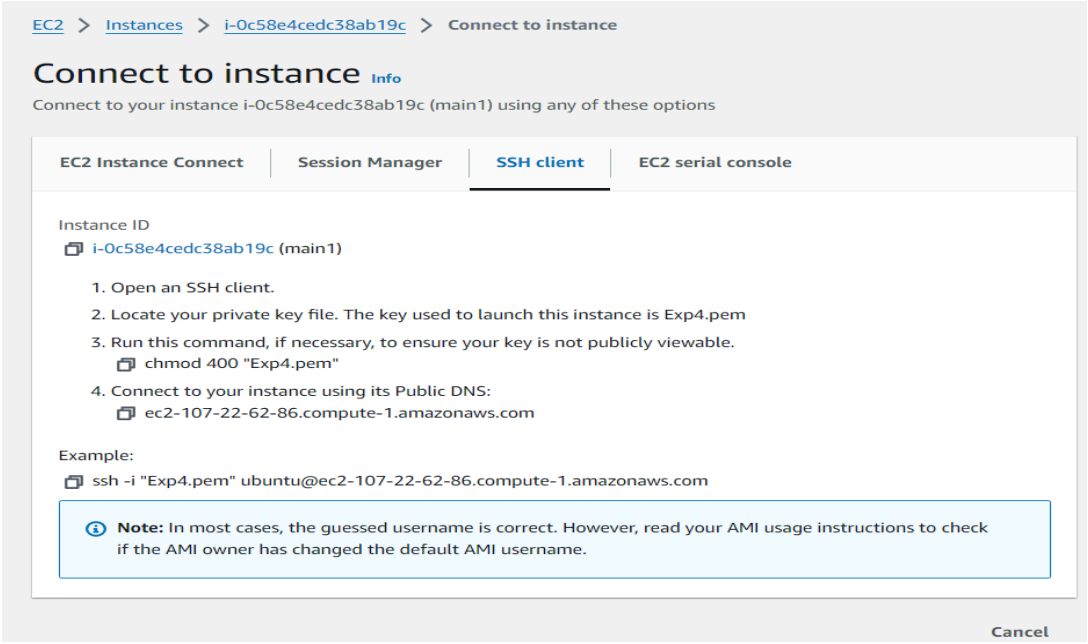


NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/ 9




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



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

Run the below commands to install and setup Docker.

**curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -**
**curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/trusted.gpg.d/docker.gpg > /dev/null**

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

```
ubuntu@ip-172-31-92-253:~$ 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
Repository: 'deb [arch=amd64] https://download.docker.com/linux/ubuntu noble stable'
Description:
Archive for codename: noble components: stable
More info: https://download.docker.com/linux/ubuntu
Adding repository.
Press [ENTER] to continue or Ctrl-c to cancel.
Adding deb entry to /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-noble.
list
Adding disabled deb-src entry to /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_
ubuntu-noble.list
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 https://download.docker.com/linux/ubuntu noble InRelease [48.8 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
```

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

NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/ 9

```
ubuntu@ip-172-31-92-253:~$ 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: The key(s) in the keyring /etc/apt
/trusted.gpg.d/docker.gpg are ignored as the file has an unsupported filetype.
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gp
g keyring (/etc/apt/trusted.gpg), see the DEPRECATION section in apt-key(8) for details.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  containerd.io docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plugin
  libltdl7 libslirp0 pigz slirp4netns
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite
The following NEW packages will be installed:
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli docker-ce-rootless-extras
  docker-compose-plugin libltdl7 libslirp0 pigz slirp4netns
0 upgraded, 10 newly installed, 0 to remove and 141 not upgraded.
Need to get 123 MB of archives.
After this operation, 442 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 pigz amd64 2.8-1 [65.6 kB]
```

```
Setting up docker-ce (5:27.3.1-1~ubuntu.24.04~noble) ...
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/
docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /usr/lib/systemd/system/dock
er.socket.
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for libc-bin (2.39-0ubuntu8.2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-92-253:~$
```

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

```
ubuntu@ip-172-31-92-253:~$ sudo mkdir -p /etc/docker
ubuntu@ip-172-31-92-253:~$ cat <<EOF | sudo tee /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
{
  "exec-opts": ["native.cgroupdriver=systemd"]
}
ubuntu@ip-172-31-92-253:~$
```

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

```
ubuntu@ip-172-31-92-253:~$ 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-sys
v-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
ubuntu@ip-172-31-92-253:~$
```

**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-172-31-92-253:~$ 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
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-172-31-92-253:~$ sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 https://download.docker.com/linux/ubuntu noble InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
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  InRelea
se [1186 B]
Get:7 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  Package
s [4865 B]
Fetched 6051 B in 1s (11.3 kB/s)
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: The key(s) in the keyring /etc/apt
/trusted.gpg.d/docker.gpg are ignored as the file has an unsupported filetype.
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gp
g keyring (/etc/apt/trusted.gpg), see the DEPRECATION section in apt-key(8) for details.
Reading package lists... Done
Building dependency tree... Done
```

NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/ 9

```
Setting up kubectl (1.31.1-1.1) ...
Setting up cri-tools (1.31.1-1.1) ...
Setting up kubernetes-cni (1.5.1-1.1) ...
Setting up kubeadm (1.31.1-1.1) ...
Setting up kubelet (1.31.1-1.1) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
ubuntu@ip-172-31-92-253:~$
```

**sudo systemctl enable --now kubelet**

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

```
ubuntu@ip-172-31-92-253:~$ 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
W0925 16:13:24.636784    4473 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 con
nection: validate CRI v1 runtime API for endpoint "unix:///var/run/containerd/containerd.sock": rpc e
rror: 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'
error execution phase preflight: [preflight] Some fatal errors occurred:
failed to create new CRI runtime service: validate service connection: validate CRI v1 runtime API fo
r endpoint "unix:///var/run/containerd/containerd.sock": rpc error: code = Unimplemented desc = unkno
wn service runtime.v1.RuntimeService[preflight] If you know what you are doing, you can make a check
non-fatal with `--ignore-preflight-errors=...`
To see the stack trace of this error execute with --v=5 or higher
ubuntu@ip-172-31-92-253:~$
```

**Now We have got an error.**

**So we have to perform some additional commands as follow.**

**sudo apt-get install -y containerd**

NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/ 9

```
ubuntu@ip-172-31-92-253:~$ 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 141 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 1.1.12-0ubun
tu3.1 [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 MB]
Fetched 47.2 MB in 1s (79.0 MB/s)
(Reading database ... 68064 files and directories currently installed.)
```

```
Setting up containerd (1.7.12-0ubuntu4.1) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-92-253:~$
```

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

NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/ 9

```
ubuntu@ip-172-31-92-253:~$ 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 Wed 2024-09-25 16:19:04 UTC; 255ms ago
       Docs: https://containerd.io
   Main PID: 5059 (containerd)
      Tasks: 8
     Memory: 13.4M (peak: 13.8M)
        CPU: 59ms
     CGroup: /system.slice/containerd.service
             └─5059 /usr/bin/containerd

Sep 25 16:19:04 ip-172-31-92-253 containerd[5059]: time="2024-09-25T16:19:04.090155391Z" level=info >
Sep 25 16:19:04 ip-172-31-92-253 containerd[5059]: time="2024-09-25T16:19:04.091499805Z" level=info >
Sep 25 16:19:04 ip-172-31-92-253 containerd[5059]: time="2024-09-25T16:19:04.091559889Z" level=info >
Sep 25 16:19:04 ip-172-31-92-253 containerd[5059]: time="2024-09-25T16:19:04.091567736Z" level=info >
Sep 25 16:19:04 ip-172-31-92-253 containerd[5059]: time="2024-09-25T16:19:04.091598787Z" level=info >
Sep 25 16:19:04 ip-172-31-92-253 containerd[5059]: time="2024-09-25T16:19:04.091606039Z" level=info >
Sep 25 16:19:04 ip-172-31-92-253 containerd[5059]: time="2024-09-25T16:19:04.091607687Z" level=info >
Sep 25 16:19:04 ip-172-31-92-253 containerd[5059]: time="2024-09-25T16:19:04.091646696Z" level=info >
Sep 25 16:19:04 ip-172-31-92-253 containerd[5059]: time="2024-09-25T16:19:04.091701973Z" level=info >
Sep 25 16:19:04 ip-172-31-92-253 systemd[1]: Started containerd.service - containerd container runti>
lines 1-21/21 (END)
```

**sudo apt-get install -y socat**

```
ubuntu@ip-172-31-92-253:~$ 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 141 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 [37
4 kB]
Fetched 374 kB in 0s (17.2 MB/s)
Selecting previously unselected package socat.
(Reading database ... 68108 files and directories currently installed.)
Preparing to unpack .../socat_1.8.0.0-4build3_amd64.deb ...
Unpacking socat (1.8.0.0-4build3) ...
Setting up socat (1.8.0.0-4build3) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

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

**Step 6:** Initialize the Kubecluster

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

```
ubuntu@ip-172-31-92-253:~$ 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'
W0925 16:26:17.463609    5539 checks.go:846] detected that the sandbox image "registry.k8s.io/pause:3
.8" of the container runtime is inconsistent with that used by kubeadm.It is recommended to use "regi
stry.k8s.io/pause:3.10" as the CRI sandbox image.
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [ip-172-31-92-253 kubernetes kubernetes.defaul
t kubernetes.default.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 172.31.92.253]
[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-172-31-92-253 localhost] and IPs [172.31
.92.253 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [ip-172-31-92-253 localhost] and IPs [172.31.9
2.253 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
```

NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/ 9

```
tificate and key
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!

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

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

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

  export KUBECONFIG=/etc/kubernetes/admin.conf

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

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

kubeadm join 172.31.92.253:6443 --token 90n342.w0y0ehbtppbqeocz \
        --discovery-token-ca-cert-hash sha256:072239c6fcbbd2b8842bd4badd167478379f455ddc7525f46bb5902
87f594319
```

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

```
ubuntu@ip-172-31-92-253:~$ mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

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

```
ubuntu@ip-172-31-92-253:~$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/D
ocumentation/kube-flannel.yml
namespace/kube-flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
```

**Step 7: Now that the cluster is up and running, we can deploy our nginx server on this cluster.Apply this deployment file using this command to create a deployment**

**kubectl apply -f https://k8s.io/examples/application/deployment.yaml**

```
ubuntu@ip-172-31-92-253:~$ kubectl apply -f https://k8s.io/examples/application/deployment.yaml
deployment.apps/nginx-deployment created
```

**kubectl get pods**

```
ubuntu@ip-172-31-92-253:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-llw22    0/1     Pending   0          25s
nginx-deployment-d556bf558-xbpvn    0/1     Pending   0          25s
```

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

**kubectl port-forward $POD_NAME 8080:80**

```
ubuntu@ip-172-31-92-253:~$ POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.
name}")
ubuntu@ip-172-31-92-253:~$ kubectl port-forward $POD_NAME 8080:80
error: unable to forward port because pod is not running. Current status=Pending
```

**kubectl get nodes**

```
ubuntu@ip-172-31-92-253:~$ kubectl get nodes
NAME                STATUS   ROLES           AGE     VERSION
ip-172-31-92-253    Ready    control-plane   7m43s   v1.31.1
```

**get pods**

```
ubuntu@ip-172-31-92-253:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-llw22    1/1     Running   0          11m
nginx-deployment-d556bf558-xbpvn    1/1     Running   0          11m
```

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

**kubectl port-forward $POD_NAME 8080:80**

```
ubuntu@ip-172-31-92-253:~$ POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items
[0].metadata.name}")
ubuntu@ip-172-31-92-253:~$ kubectl port-forward $POD_NAME 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
Handling connection for 8080
```

**Step 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.

NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/ 9

**curl --head http://127.0.0.1:8080**

```
PS C:\Users\LENOVO> ssh -i "Exp4.pem" ubuntu@ec2-54-164-58-232.compute-1.amazonaws.com
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

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

 System information as of Wed Sep 25 16:45:32 UTC 2024

  System load:  0.07              Processes:             155
  Usage of /:   55.5% of 6.71GB   Users logged in:       1
  Memory usage: 19%               IPv4 address for enX0: 172.31.92.253
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

143 updates can be applied immediately.
41 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: Wed Sep 25 16:12:02 2024 from 203.194.102.247
```

```
ubuntu@ip-172-31-92-253:~$ curl --head http://127.0.0.1:8080
HTTP/1.1 200 OK
Server: nginx/1.14.2
Date: Wed, 25 Sep 2024 16:45:49 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
```

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

NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/ 9

**Conclusion:**

**1. EC2 Instance Launch Issues:** Incorrect AMI Selection: Selecting the wrong Amazon Machine Image (AMI) could cause issues, especially if it doesn't support the required software for Kubernetes.

**2. Kubernetes Installation Issues:** Installation Fails Due to Network Issues: Sometimes, the installation of Kubernetes can fail due to network errors or misconfigured package repositories.

**3. Nginx Deployment Issues:** Nginx Server Not Running: After deploying Nginx, the server might not start due to insufficient resources on the EC2 instance or missing configurations.

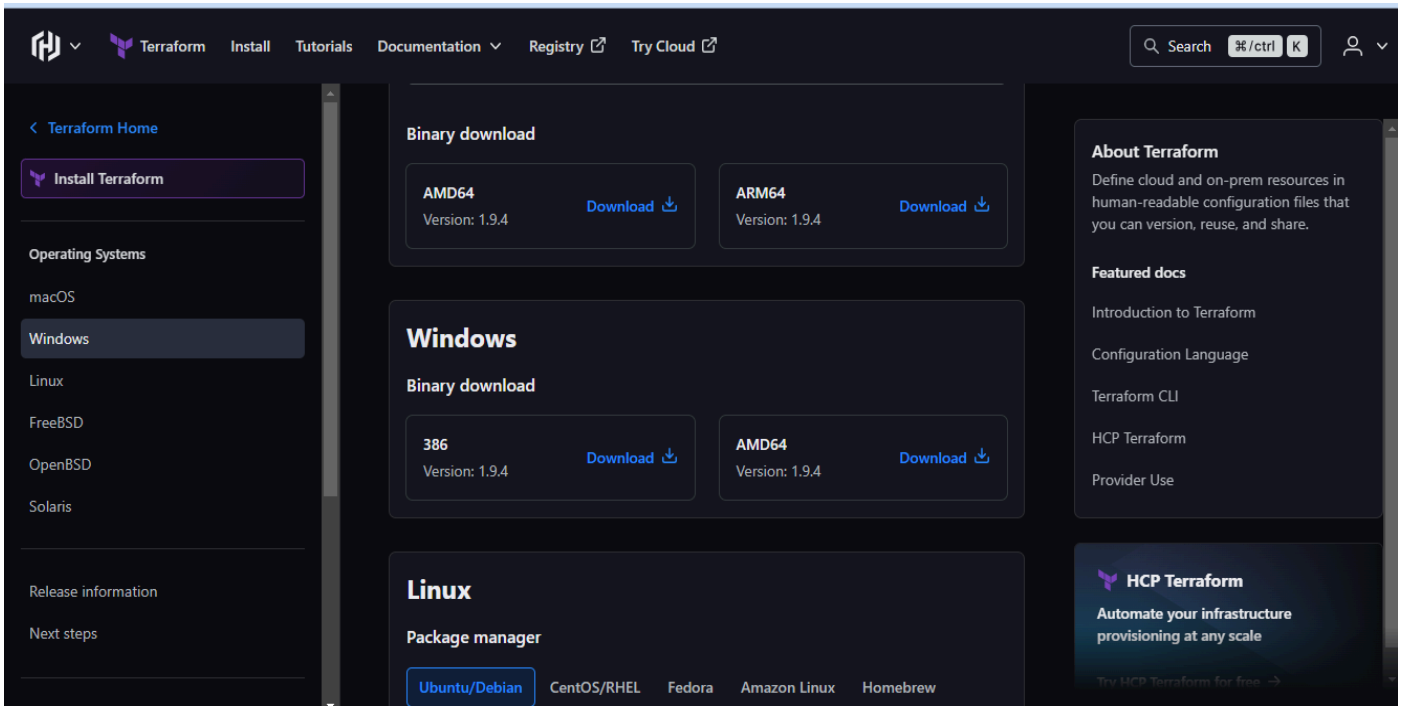
NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/9

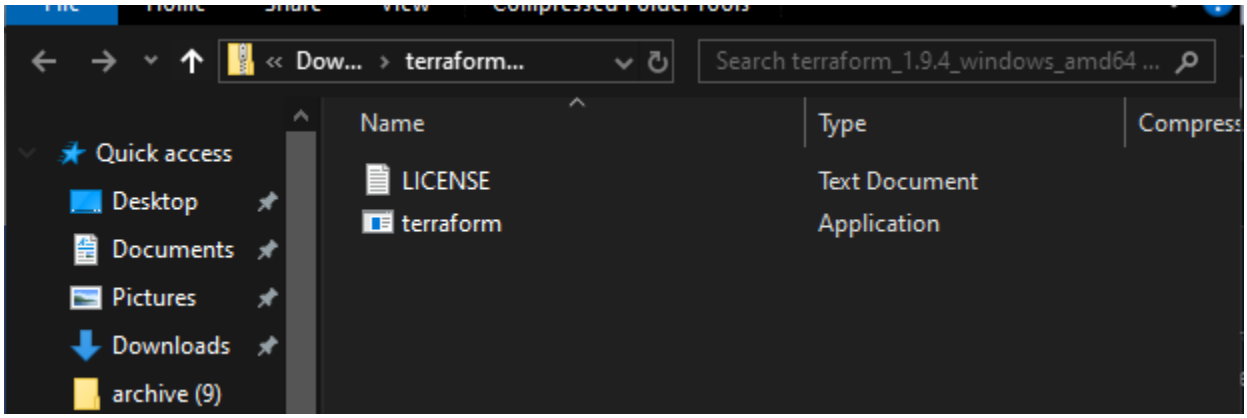
**AdvanceDevops Experiment 5**

## Aim: To understand terraform lifecycle, core concepts/terminologies and install it on Windows.

### 1. Download the Terraform:

click on the link https://www.terraform.io/downloads.html
Select the Operating System and download Terraform version according to configurations.



**3. Find the Terraform Executable Path:**

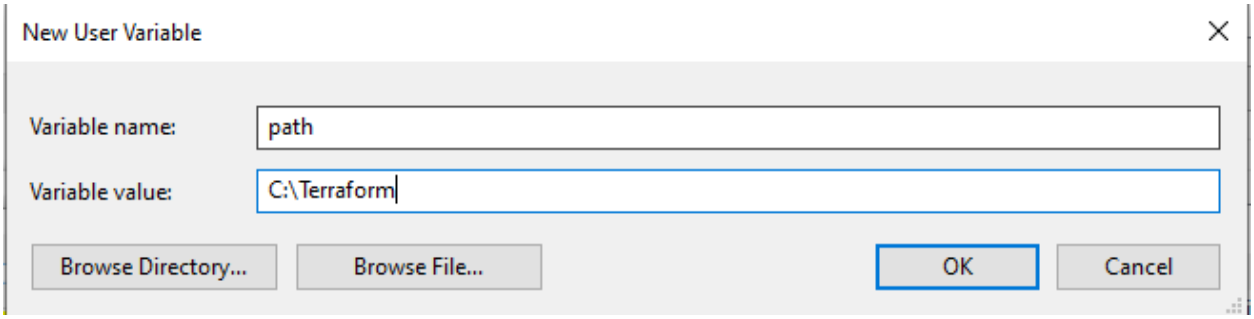
After installation, locate the directory where Terraform executable (terraform.exe) is present. For example, it could be in C:\terraform or any other path you chose during Installation.

**4. Edit System Environment Variables:**

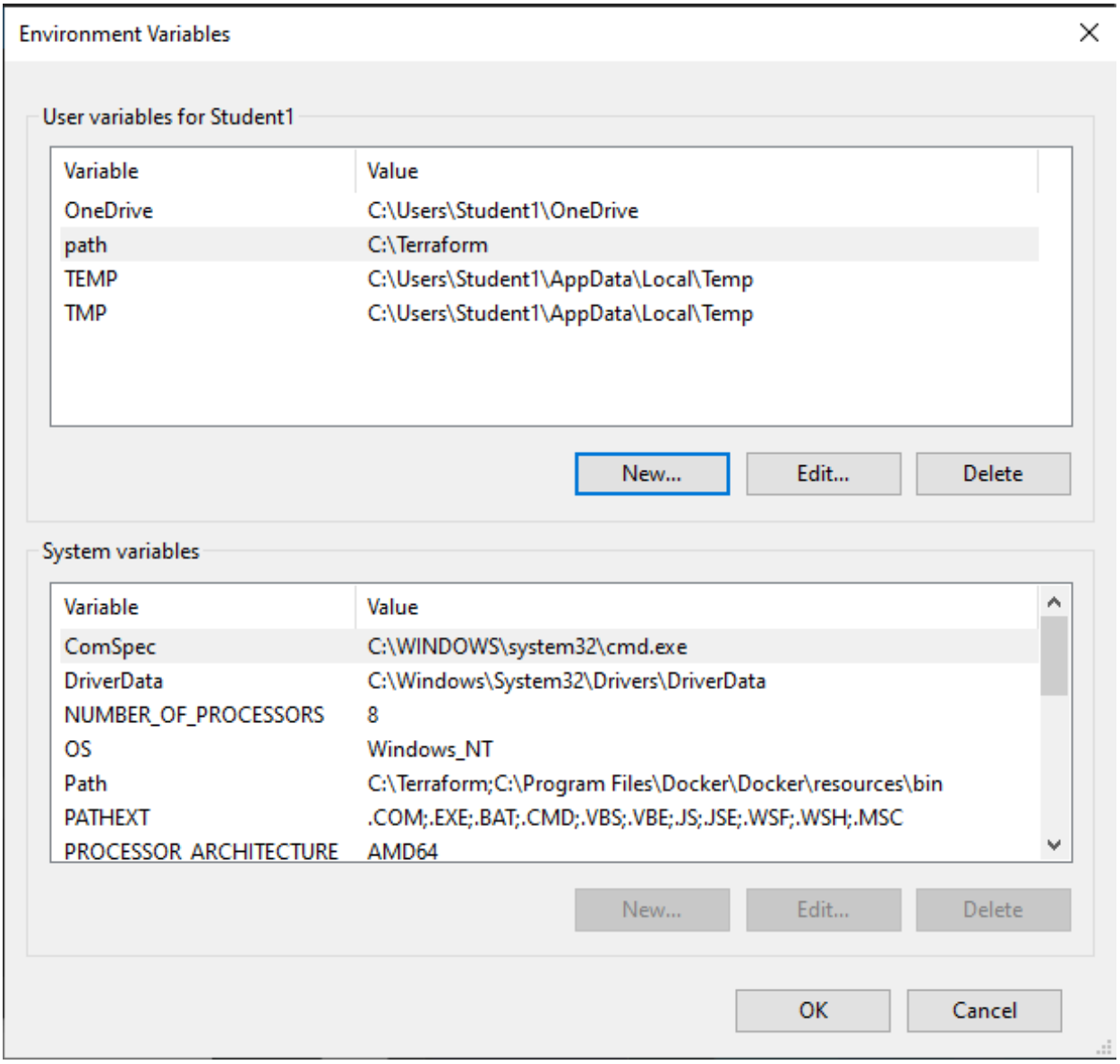
In the Environment Variables window (System Properties), under the section System Variables, find and select the variable named Path, then click Edit.

**5. Add Terraform to Path:**

In the Edit Environment Variable window, click New to add a new entry.
Enter the path to the directory where Terraform executable (terraform.exe) is located. For example, if Terraform is installed in C:\terraform, add C:\terraform to the list.

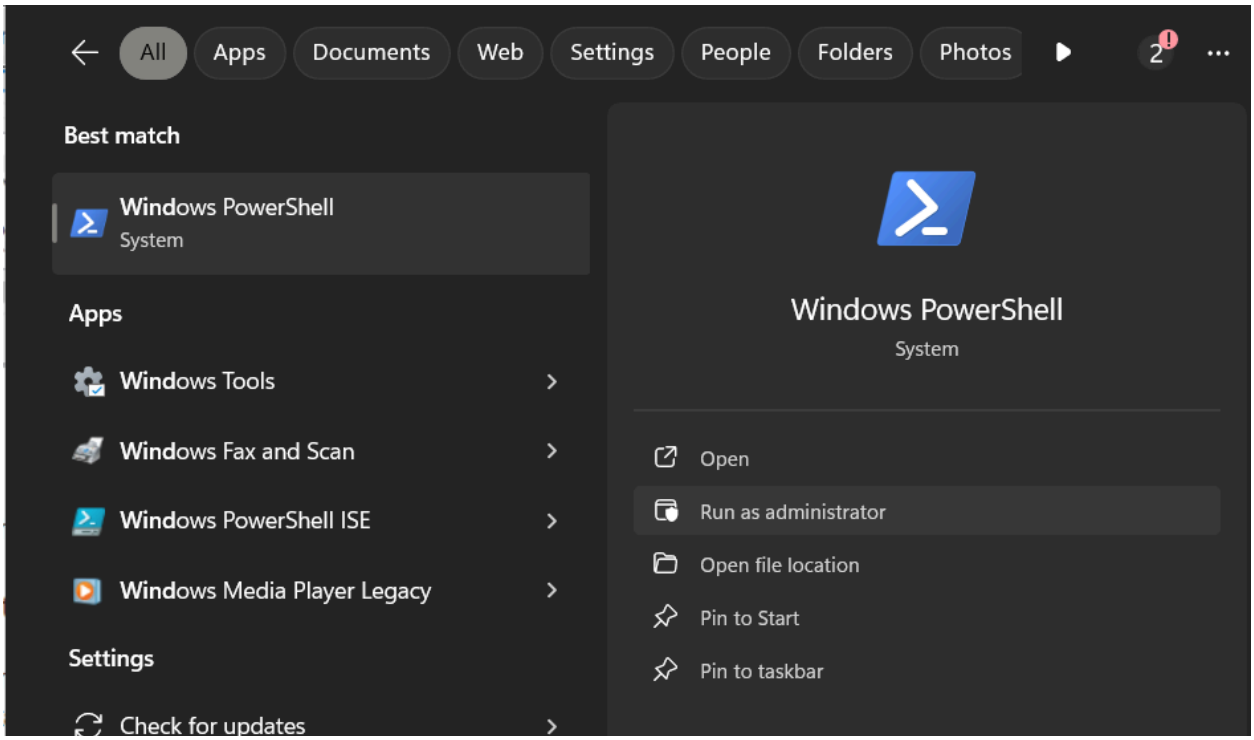


NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/9



NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/9

**6. Open PowerShell with Administrator Access.**

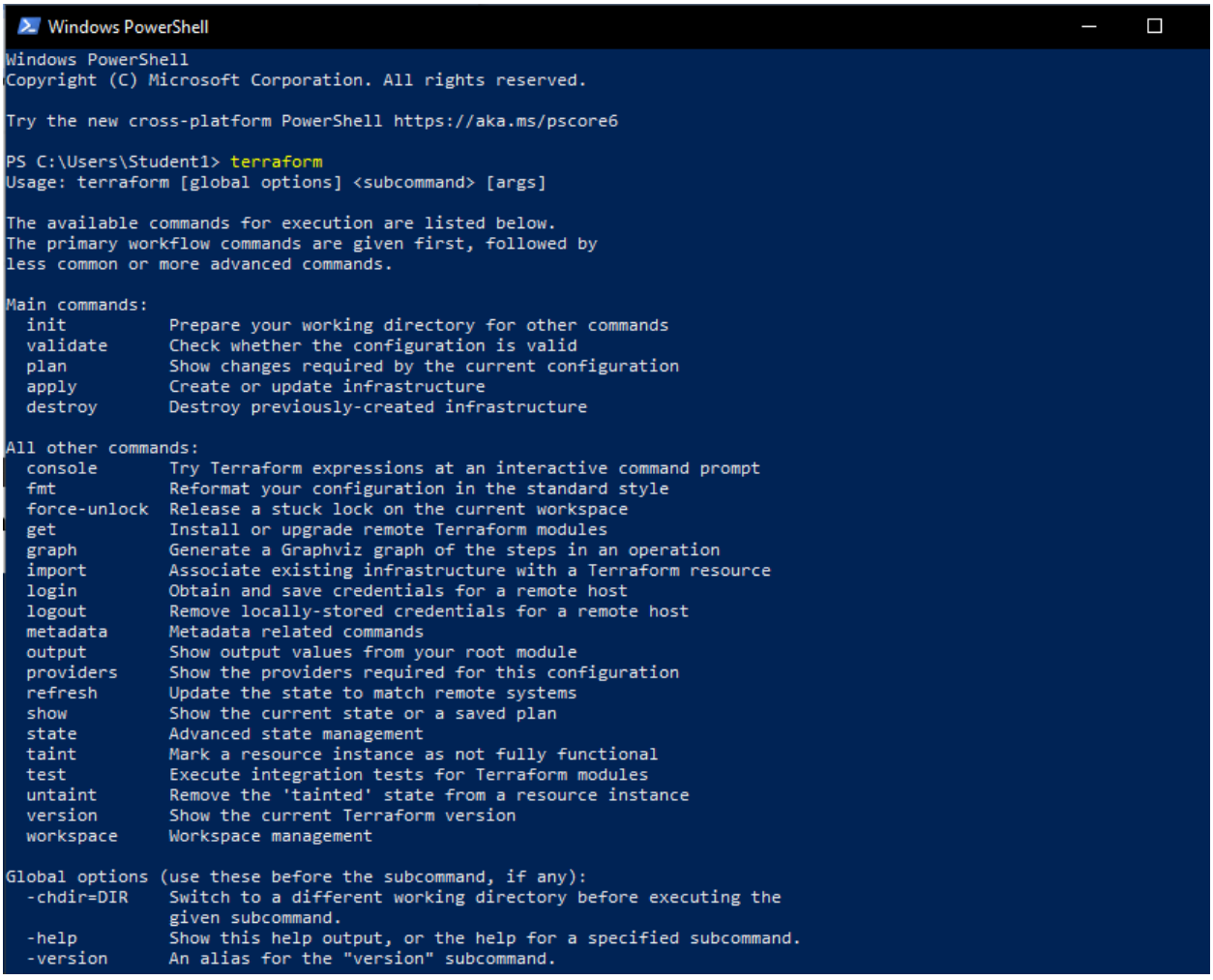


NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/9

**7.Verify Installation:**

Open a new command prompt (to ensure it picks up the updated environment variables).

Type **terraform** and press Enter. You should see the Terraform version information printed on the screen if the path configuration was successful.



NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/ 9

**AdvanceDevops Experiment: 6**

**AIM: To Build, change, and destroy AWS /GCP/ Microsoft Azure/ Digital Ocean infrastructure using Terraform. (S3 bucket or Docker)**

**A. Creating docker image using terraform**

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

**Step 1:** Check the docker functionality

```
PS C:\Users\INFT505-16> 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
```

```
PS C:\Users\INFT505-16> docker --version
Docker version 24.0.6, build ed223bc
PS C:\Users\INFT505-16>
```

NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/ 9

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

**Step 2:** Firstly create a new folder named 'Docker' in the 'TerraformScripts' folder. Then create a new docker.tf file using Atom editor. Copy the Script into it.

**Script:**

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

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

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

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

NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/ 9

docker.tf

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

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

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

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

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

```
C:\Users\INFT505-16>cd desktop\TerraformScripts\Docker

C:\Users\INFT505-16\Desktop\TerraformScripts\Docker>terraform init
Initializing the backend...
Initializing provider plugins...
- Finding kreuzwerker/docker versions matching "2.21.0"...
- Installing kreuzwerker/docker v2.21.0...
- Installed kreuzwerker/docker v2.21.0 (self-signed, key ID BD080C4571C6104C)
Partner and community providers are signed by their developers.
If you'd like to know more about provider signing, you can read about it here:
https://www.terraform.io/docs/cli/plugins/signing.html
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

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

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

NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/ 9

**Step 4:** 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”

```
C:\Users\INFT505-16\Desktop\TerraformScripts\Docker>terraform apply

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)
```

```
      + output      = (known after apply)
      + repo_digest = (known after apply)
    }

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

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

  Enter a value: yes

docker_image.ubuntu: Creating...
docker_image.ubuntu: Still creating... [10s elapsed]
docker_image.ubuntu: Still creating... [20s elapsed]
docker_image.ubuntu: Still creating... [30s elapsed]
docker_image.ubuntu: Still creating... [40s elapsed]
docker_image.ubuntu: Still creating... [50s elapsed]
docker_image.ubuntu: Creation complete after 53s [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Creating...
docker_container.foo: Creation complete after 2s [id=353bd0cae537e335931797ed86d2b603c682520b71c2af7d5b72f3c09eed2b11]
```

Docker images, After Executing Apply step:

```
C:\Users\INFT505-16\Desktop\TerraformScripts\Docker>docker images
REPOSITORY   TAG       IMAGE ID       CREATED         SIZE
ubuntu       latest    edbfe74c41f8   3 weeks ago     78.1MB
sonarqube    latest    3183d6818c6e   10 months ago   716MB
```

**Step 5**: Execute Terraform destroy to delete the configuration, which will automatically delete the Ubuntu Container.

```
C:\Users\INFT505-16\Desktop\TerraformScripts\Docker>terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Refreshing state... [id=353bd0cae537e335931797ed86d2b603c682520b71c2af7d5b72f3c09eed2b11]

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

Terraform will perform the following actions:

  # docker_container.foo will be destroyed
  - resource "docker_container" "foo" {
      - attach           = false -> null
      - command          = [
          - "sleep",
          - "3600",
        ] -> null
      - cpu_shares       = 0 -> null
      - dns              = [] -> null
      - dns_opts         = [] -> null
      - dns_search       = [] -> null
      - entrypoint       = [] -> null
      - env              = [] -> null
      - gateway          = "172.17.0.1" -> null
      - group_add        = [] -> null
```

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

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

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

  Enter a value: yes

docker_container.foo: Destroying... [id=353bd0cae537e335931797ed86d2b603c682520b71c2af7d5b72f3c09eed2b11]
docker_container.foo: Destruction complete after 0s
docker_image.ubuntu: Destroying... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_image.ubuntu: Destruction complete after 1s

Destroy complete! Resources: 2 destroyed.
```

**Step 6:** This command outputs the state or plan in a human-readable format, helping you review the details of your configuration..

```
C:\Users\INFT505-16\Desktop\TerraformScripts\Docker>terraform show
# docker_container.foo:
resource "docker_container" "foo" {
    attach            = false
    bridge            = null
    command           = [
        "sleep",
        "3600",
    ]
    cpu_set           = null
    cpu_shares        = 0
    domainname        = null
    entrypoint        = []
    env               = []
    gateway           = "172.17.0.1"
    hostname          = "353bd0cae537"
    id                = "353bd0cae537e335931797ed86d2b603c682520b71c2af7d5b72f3c09eed2b11"
    image             = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a"
    init              = false
    ip_address        = "172.17.0.2"
    ip_prefix_length  = 16
```

**Step 7:** This command generates a visual graph of your Terraform resources, which can help you understand the dependencies and relationships between them.

```
C:\Users\INFT505-16\Desktop\TerraformScripts\Docker>terraform graph
digraph G {
  rankdir = "RL";
  node [shape = rect, fontname = "sans-serif"];
  "docker_container.foo" [label="docker_container.foo"];
  "docker_image.ubuntu" [label="docker_image.ubuntu"];
  "docker_container.foo" -> "docker_image.ubuntu";
}
```

**Step 8:** This command lists all the resources tracked by the Terraform state, allowing you to see which resources have been created and are being managed by Terraform..

```
C:\Users\INFT505-16\Desktop\TerraformScripts\Docker>terraform state list
docker_container.foo
docker_image.ubuntu

C:\Users\INFT505-16\Desktop\TerraformScripts\Docker>
```

# AdvanceDevops Experiment 7

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

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

## What problems does SAST solve?

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

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

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

### Why is SAST important?

Developers dramatically outnumber security staff. It can be challenging for an organization to find the resources to perform code reviews on even a fraction of its applications. A key strength of SAST tools is the ability to analyze 100% of the codebase. Additionally, they are much faster than manual secure code reviews performed by humans. These tools can scan millions of lines of code in a matter of minutes. SAST tools automatically identify critical vulnerabilities—such as buffer overflows, SQL injection, cross-site scripting, and others—with high confidence. Thus, integrating static analysis into the SDLC can yield

dramatic results in the overall quality of the code developed.

**What are the key steps to run SAST effectively?**

There are six simple steps needed to perform SAST efficiently in organizations that have a very large number of applications built with different languages, frameworks, and platforms.

1. **Finalize the tool.** Select a static analysis tool that can perform code reviews of applications written in the programming languages you use. The tool should also be able to comprehend the underlying framework used by your software.
2. **Create the scanning infrastructure, and deploy the tool.** This step involves handling the licensing requirements, setting up access control and authorization, and procuring the resources required (e.g., servers and databases) to deploy the tool.
3. **Customize the tool.** Fine-tune the tool to suit the needs of the organization. For example, you might configure it to reduce false positives or find additional security vulnerabilities by writing new rules or updating existing ones. Integrate the tool into the build environment, create dashboards for tracking scan results, and build custom reports.
4. **Prioritize and onboard applications.** Once the tool is ready, onboard your applications. If you have a large number of applications, prioritize the high-risk applications to scan first. Eventually, all your applications should be onboarded and scanned regularly, with application scans synced with release cycles, daily or monthly builds, or code check-ins.
5. **Analyze scan results.** This step involves triaging the results of the scan to remove false positives. Once the set of issues is finalized, they should be tracked and provided to the deployment teams for proper and timely remediation.
6. **Provide governance and training.** Proper governance ensures that your development teams are employing the scanning tools properly. The software security touchpoints should be present within the SDLC. SAST should be incorporated as part of your application development and deployment process.

## Integrating Jenkins with SonarQube:

Windows installation
Step 1 Install JDK 1.8
Step 2 download and install jenkins
https://www.blazemeter.com/blog/how-to-install-jenkins-on-windows

**Ubuntu installation**

**https://www.digitalocean.com/community/tutorials/how-to-install-java-with-apt on-ubuntu-20-04#installing-the-default-jre-jdk**

Step 1 Install JDK 1.8
sudo apt-get install openjdk-8-jre

sudo apt install default-jre

https://www.digitalocean.com/community/tutorials/how-to-install-jenkins-on-ubuntu 20-04

Open SSH

**Prerequisites:**

- Jenkins installed
- Docker Installed (for SonarQube)

(sudo apt-get install docker-ce=5:20.10.15~3-0~ubuntu-jammy
docker-ce-cli=5:20.10.15~3-0~ubuntu-jammy containerd.io docker-compose-plugin)

- SonarQube Docker Image

## Steps to integrate Jenkins with SonarQube

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

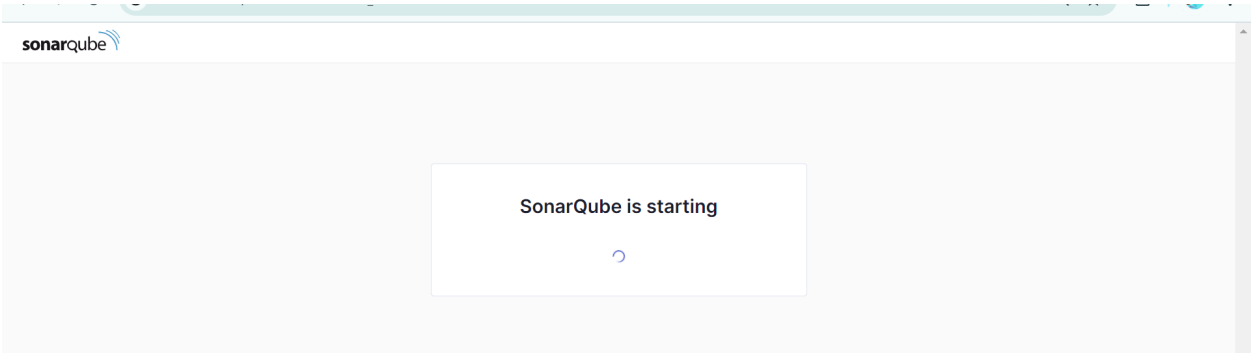
2. Run SonarQube in a Docker container using this command -

```
PS C:\Users\91900> docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
8b2a833004ac39a9b009118bacac47e5808c9ec8df3f59f8657bd23fa23f48f2
```

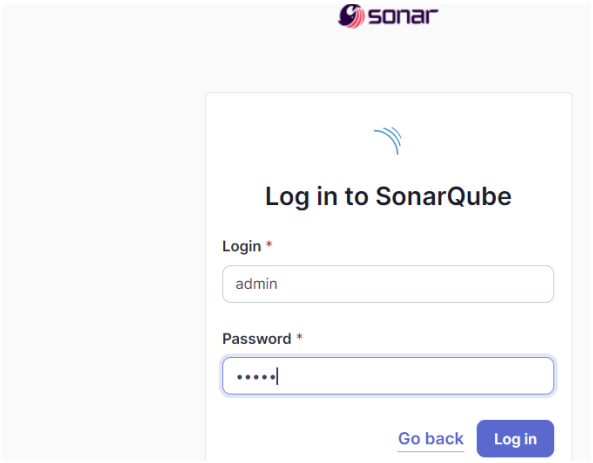
Warning: run below command only once

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

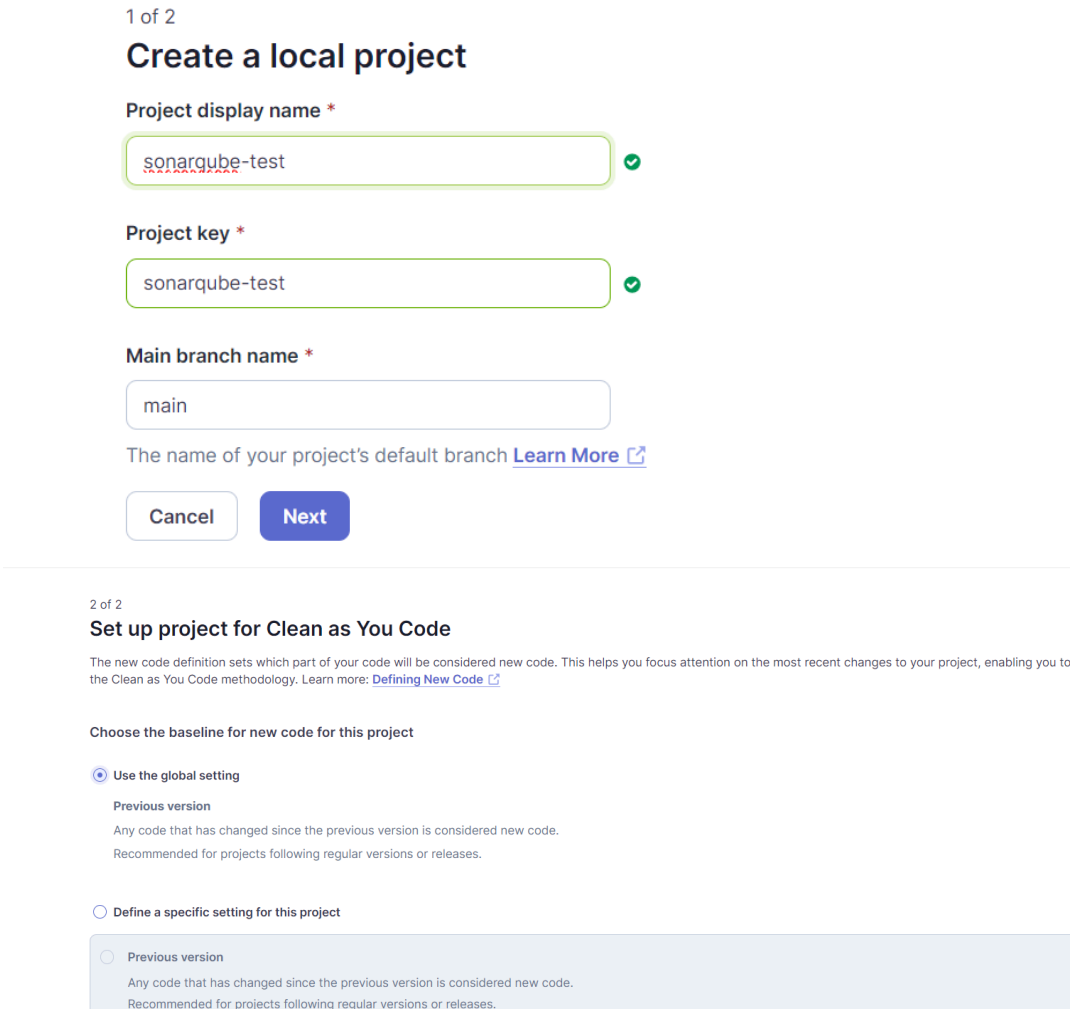
3. Once the container is up and running, you can check the status of SonarQube at localhost port 9000.




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




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

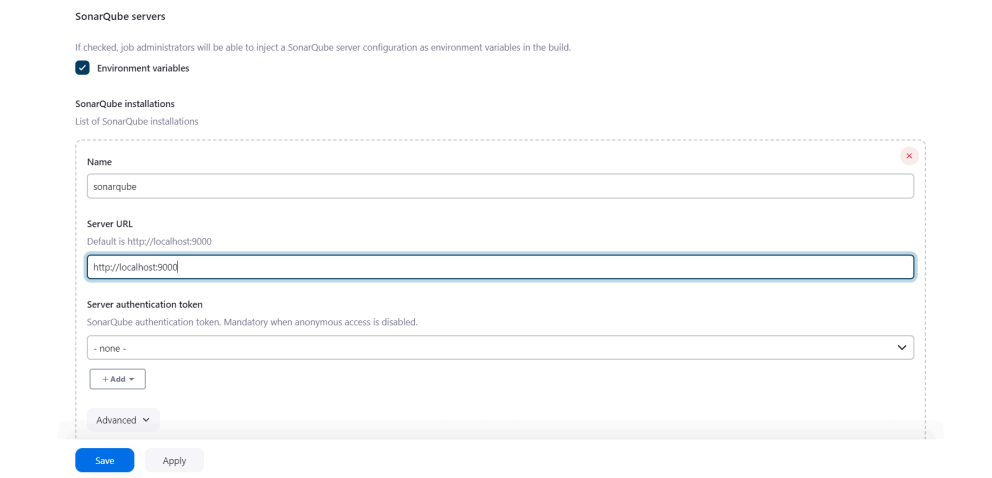



Setup the project and come back to Jenkins Dashboard.

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

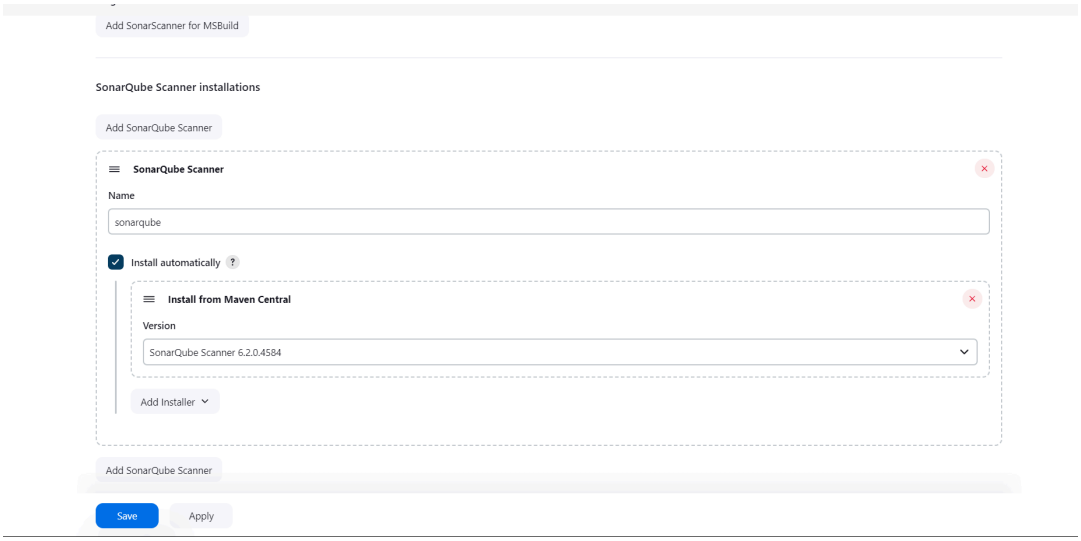

6. Under Jenkins 'Configure System', look for SonarQube Servers and enter the details.

Enter the Server Authentication token if needed.

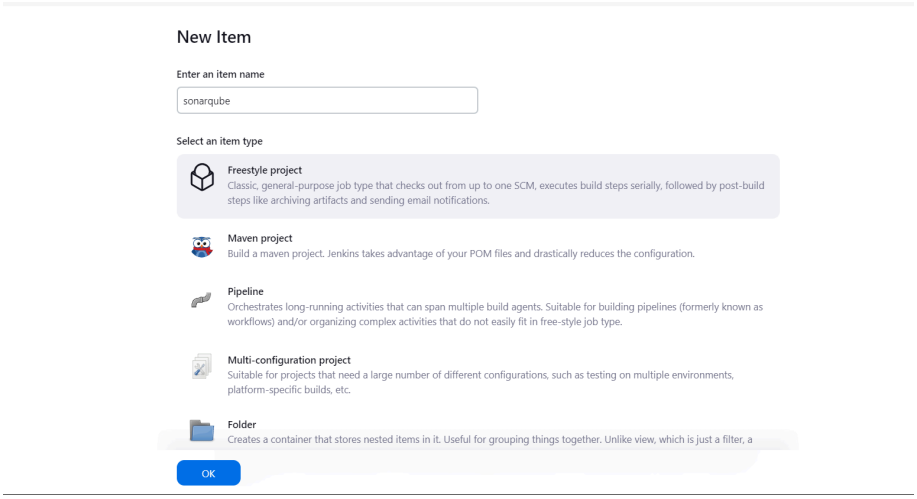



7. Search for SonarQube Scanner under Global Tool Configuration. Choose the latest configuration and choose Install automatically

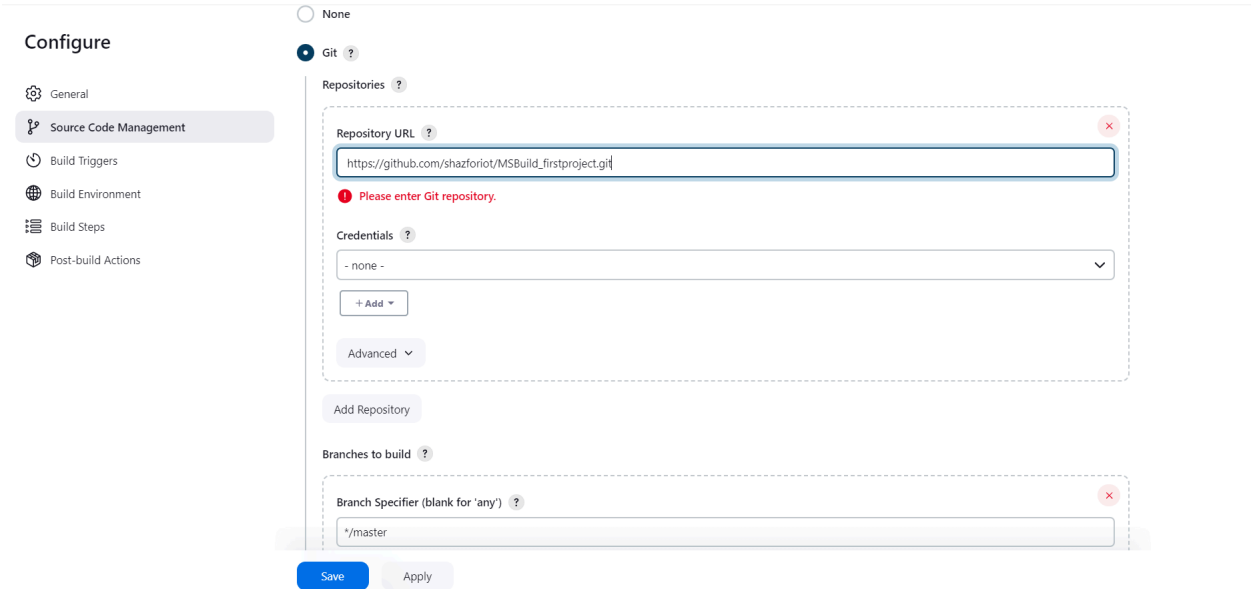
.




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

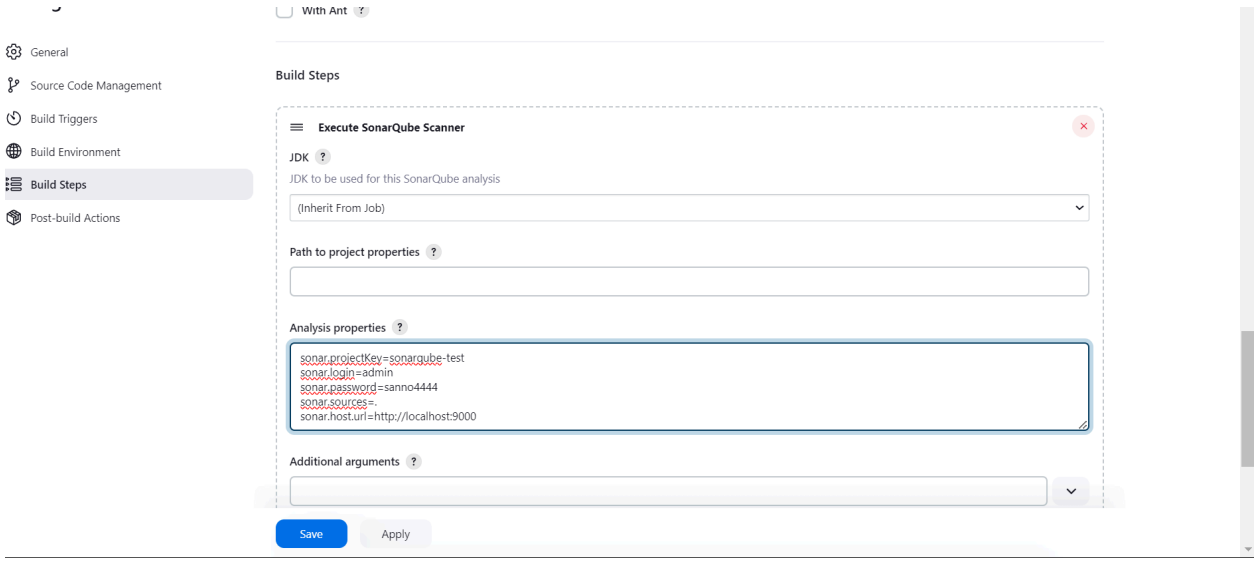



9. Choose this GitHub repository in Source Code Management.
https://github.com/shazforiot/MSBuild_firstproject.git
It is a sample hello-world project with no vulnerabilities and issues, just to test

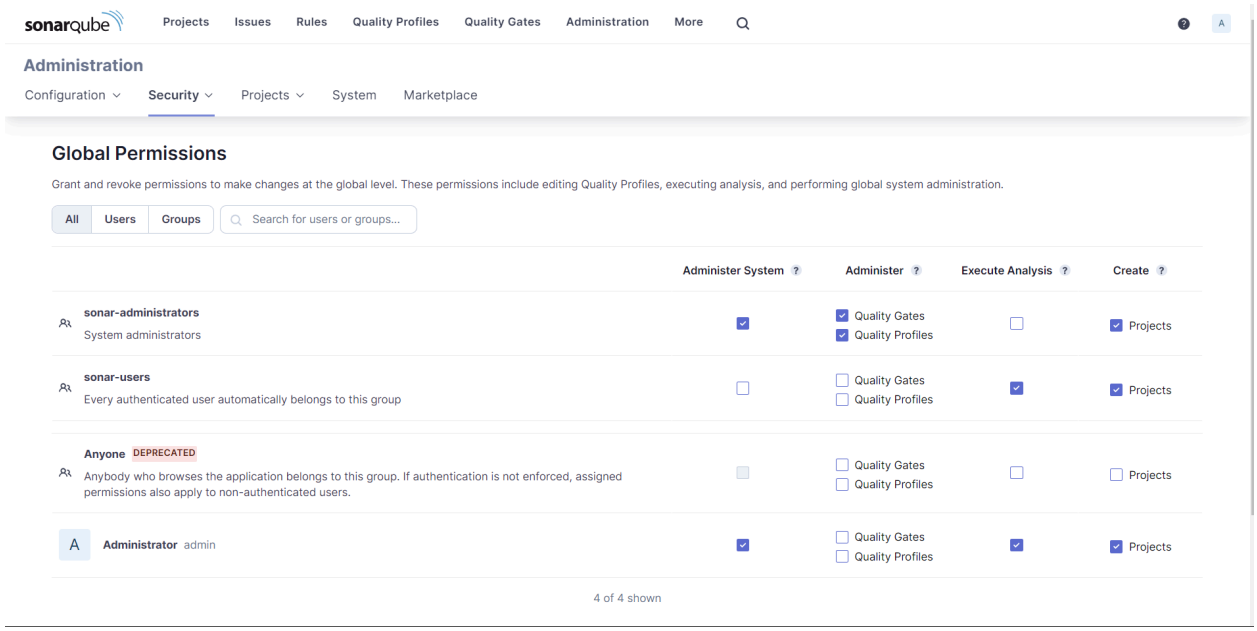



the integration.

10. Under Build-> Execute SonarQube Scanner, enter these Analysis properties. Mention the SonarQube Project Key, Login, Password, Source path and Host URL.

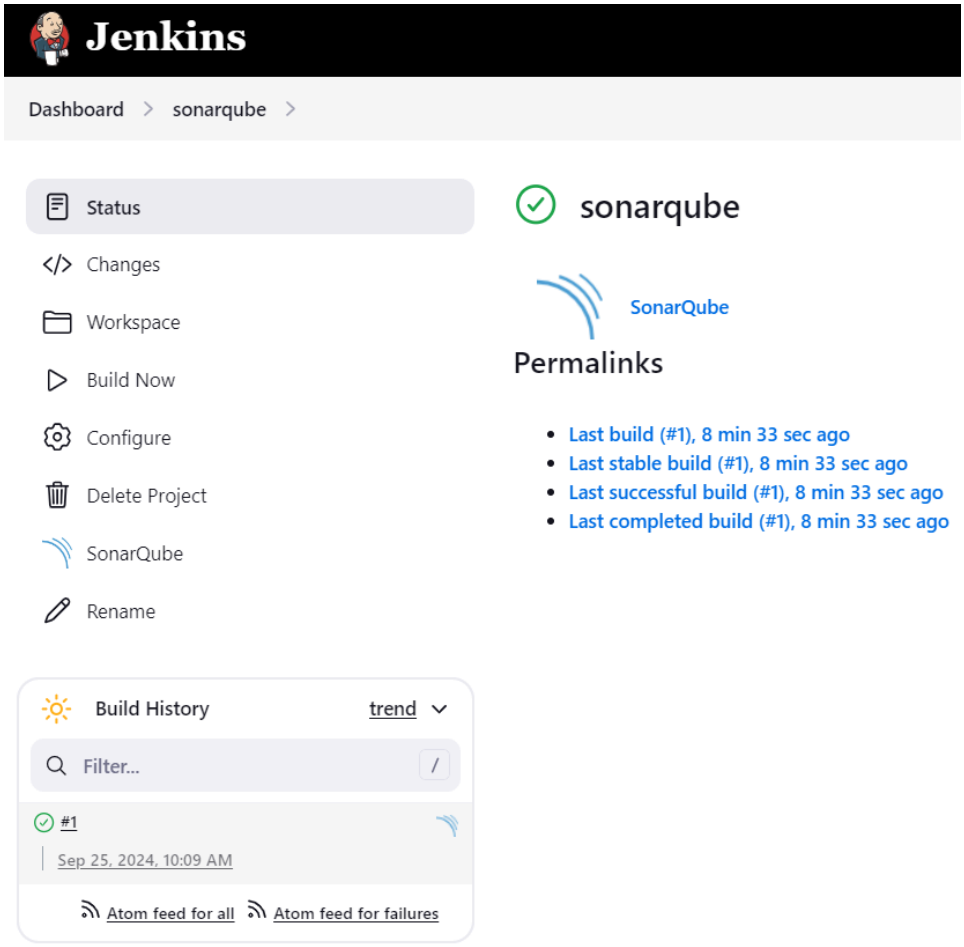



11. Go to http://localhost:9000/<user_name>/permissions and allow Execute Permissions to the Admin user.

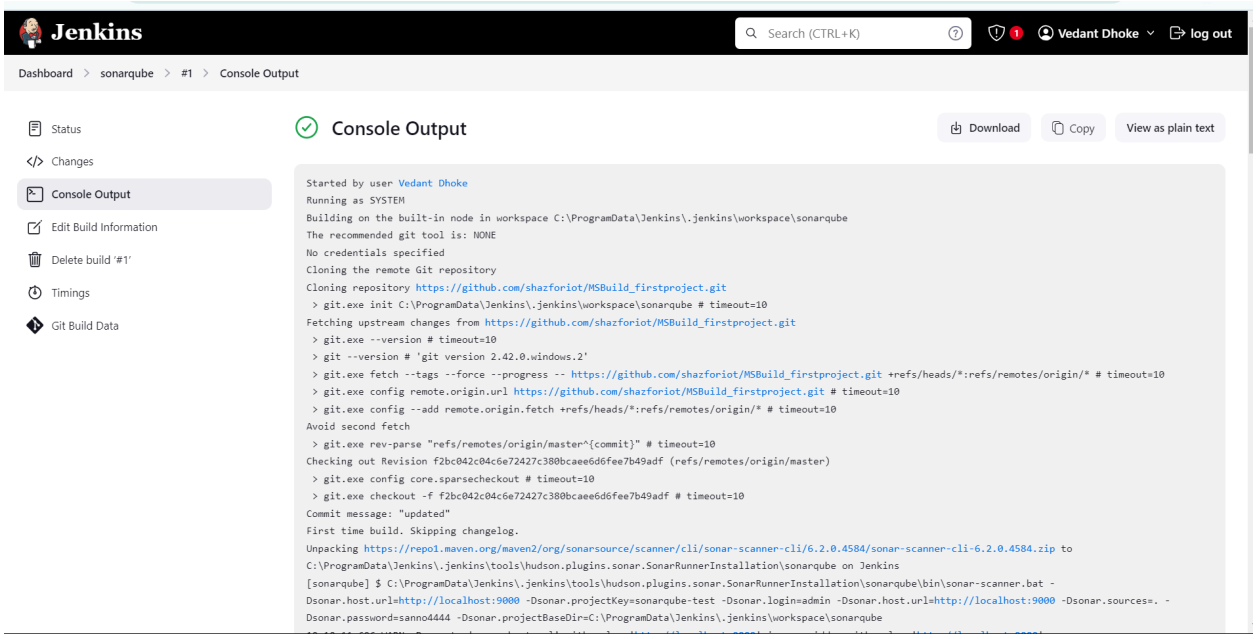



12. Run The Build.

NAME: VEDANT DHOKE CLASS/ ROLL NO: D15C/ 9

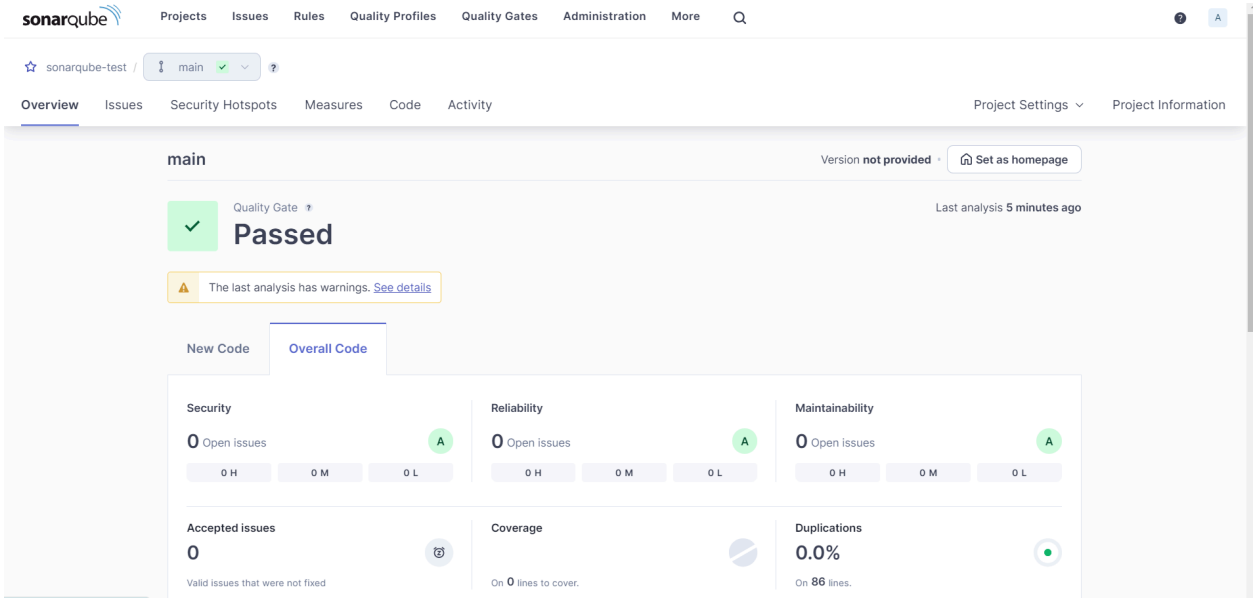



Check the console output.



```
SonarScanner for .NET 5.x or higher, see https://redirect.sonarsource.com/doc/install-configure-scanner-msbuild.html
10:10:57.397 INFO  Sensor C# [csharp] (done) | time=2ms
10:10:57.397 INFO  Sensor Analysis Warnings import [csharp]
10:10:57.399 INFO  Sensor Analysis Warnings import [csharp] (done) | time=4ms
10:10:57.401 INFO  Sensor C# File Caching Sensor [csharp]
10:10:57.405 WARN  Incremental PR analysis: Could not determine common base path, cache will not be computed. Consider setting 'sonar.projectBaseDir'
property.
10:10:57.405 INFO  Sensor C# File Caching Sensor [csharp] (done) | time=5ms
10:10:57.405 INFO  Sensor Zero Coverage Sensor
10:10:57.424 INFO  Sensor Zero Coverage Sensor (done) | time=19ms
10:10:57.428 INFO  SCM Publisher SCM provider for this project is: git
10:10:57.430 INFO  SCM Publisher 4 source files to be analyzed
10:10:58.315 INFO  SCM Publisher 4/4 source files have been analyzed (done) | time=883ms
10:10:58.324 INFO  CPD Executor Calculating CPD for 0 files
10:10:58.363 INFO  CPD Executor CPD calculation finished (done) | time=0ms
10:10:58.372 INFO  SCM revision ID 'f2bc042c04c6e72427c380bcaee6d6fee7b49adf'
10:10:58.843 INFO  Analysis report generated in 226ms, dir size=201.0 kB
10:10:58.903 INFO  Analysis report compressed in 45ms, zip size=22.2 kB
10:10:59.397 INFO  Analysis report uploaded in 491ms
10:10:59.401 INFO  ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=sonarqube-test
10:10:59.402 INFO  Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
10:10:59.403 INFO  More about the report processing at http://localhost:9000/api/ce/task?id=2620d8fe-827f-4a3c-999f-1ed8a7b15249
10:10:59.429 INFO  Analysis total time: 30.223 s
10:10:59.431 INFO  SonarScanner Engine completed successfully
10:10:59.519 INFO  EXECUTION SUCCESS
10:10:59.521 INFO  Total time: 47.815s
Finished: SUCCESS
```

13. Once the build is complete, check the project in SonarQube.




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

## Conclusion

**1. Docker Container Issues:** The SonarQube container might not start because your system doesn't have enough memory or processing power. SonarQube needs around 2GB of RAM to work properly, so if your system is low on resources, the container won't run.

**2. Login Problems in SonarQube:** You might have trouble logging in with the default username (admin) and password (admin). This could happen if there was a configuration issue with SonarQube or if the default password was changed during previous setups.

**3. Jenkins Plugin Installation Errors:** While installing the SonarQube Scanner plugin in Jenkins, you might encounter failures due to network issues or proxy settings, preventing the plugin from downloading correctly.

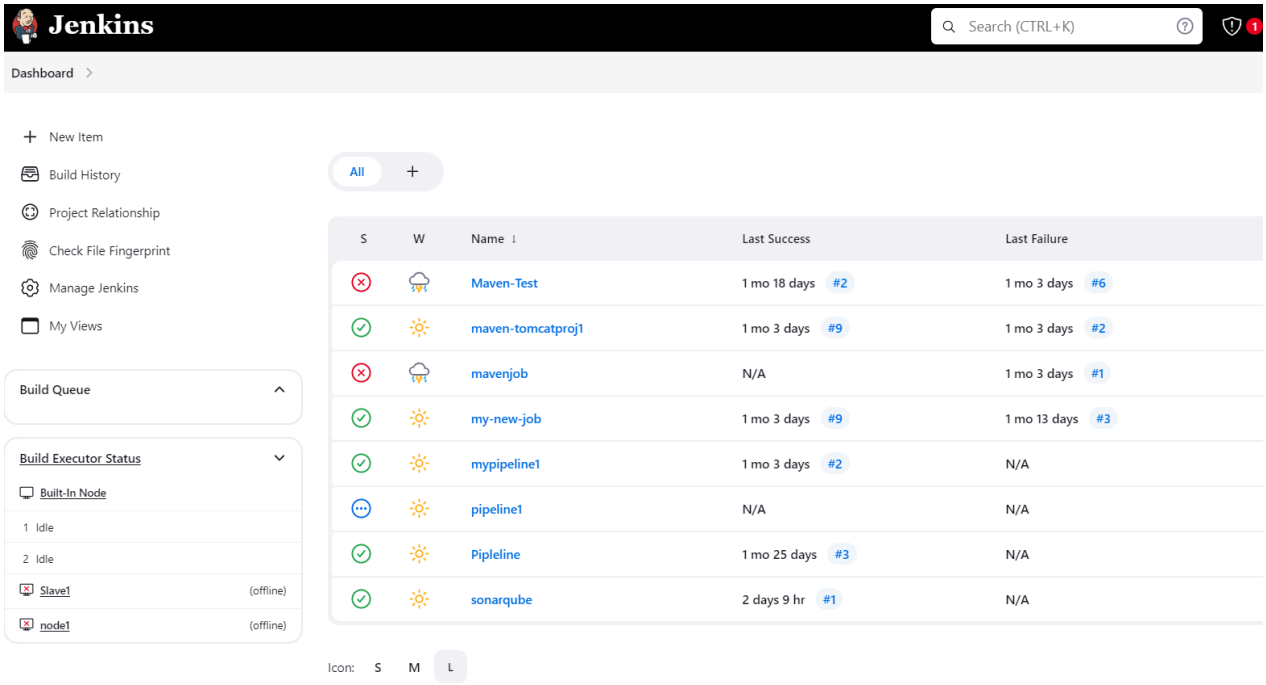
**4. Incorrect SonarQube Configuration in Jenkins:** While configuring SonarQube in Jenkins, entering the wrong project key, username, or password can cause the scan to fail. Ensuring accurate information is critical for a successful scan.

NAME: VEDANT DHOKE CLASS/ ROLL NO: D15C/ 9

# AdvanceDevops Experiment 8

**Aim:**Create a Jenkins CICD Pipeline with SonarQube / GitLab Integration to perform a static analysis of the code to detect bugs, code smells, and security vulnerabilities on a sample Web Java / Python application. dive deep into this segment, let's first understand what is meant by the term ‘pipeline’?

1. Open Jenkin dashboard.

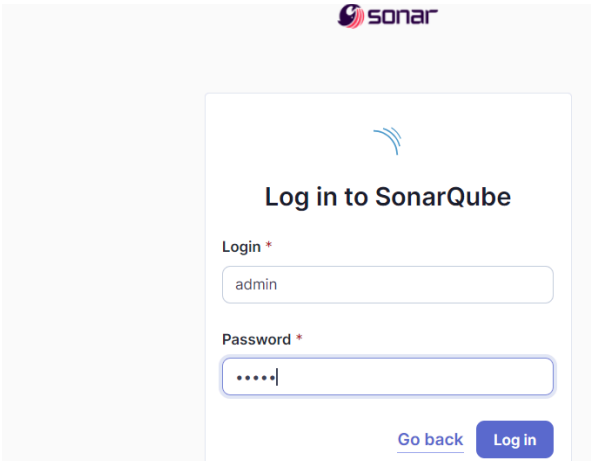



2. Run SonarQube in a Docker container using this command -

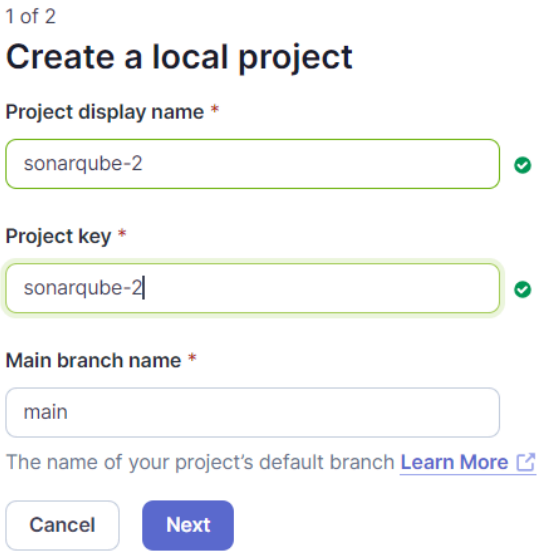
```
C:\Users\91900>docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
d23acccacd96c274f5f87912674ecf2d9adfff185a940c24740f44b29534485
```

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

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

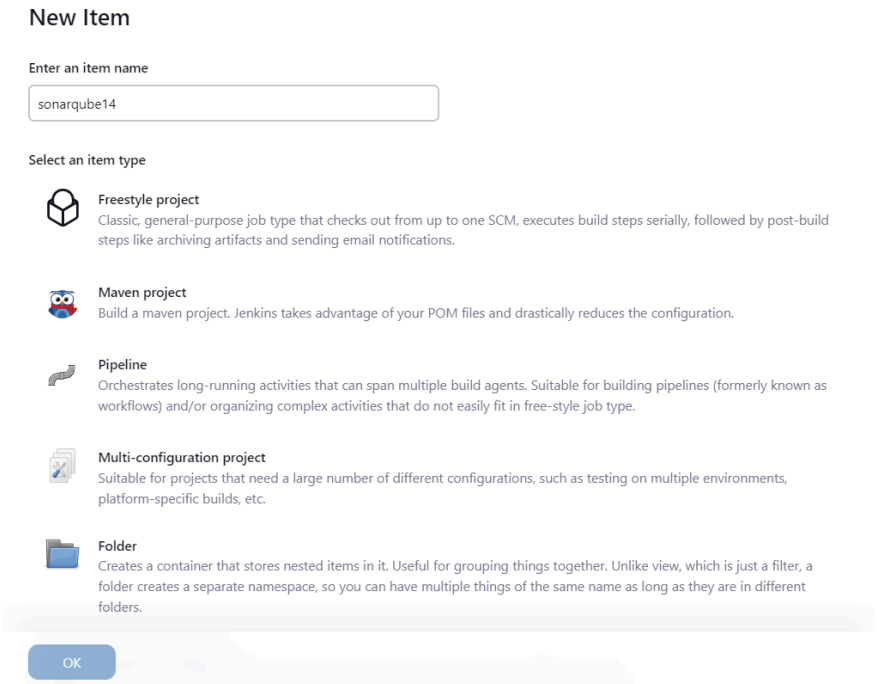



5. Create a manual project in SonarQube with the name **sonarqube-test**



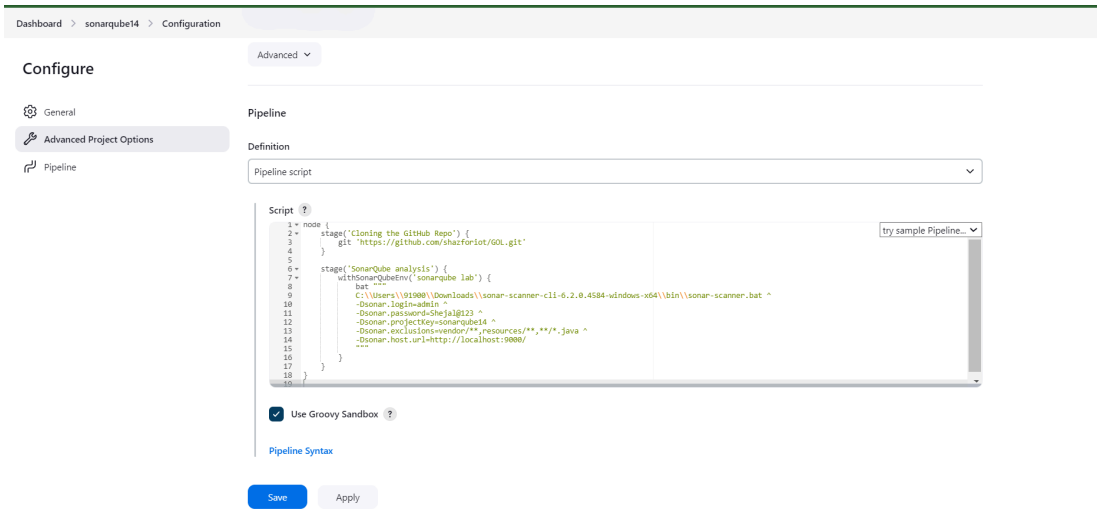

Setup the project and come back to Jenkins Dashboard.

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



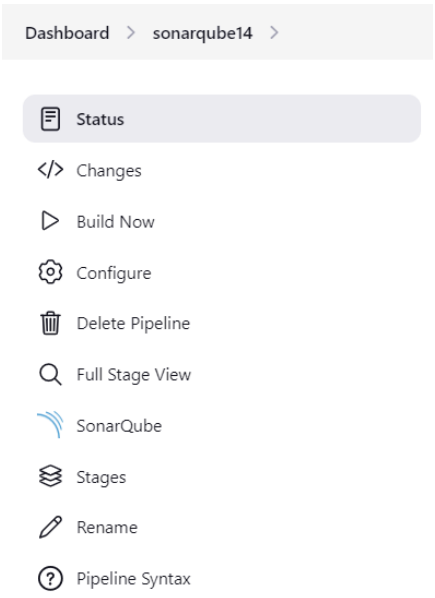

7. Under Pipeline Script, enter the following -

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

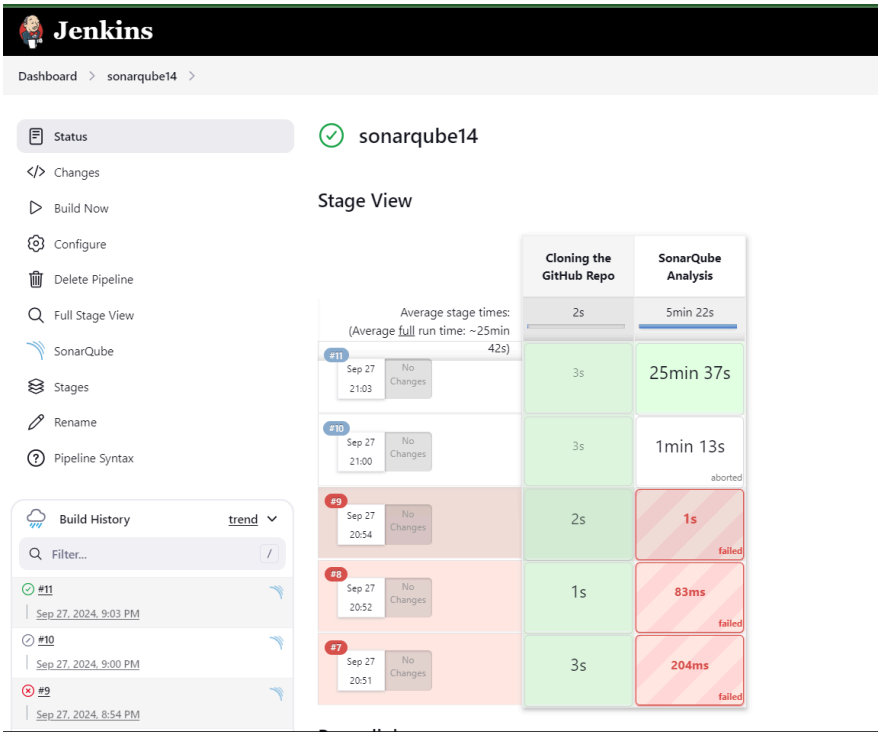



It is a java sample project which has a lot of repetitions and issues that will be detected by SonarQube.

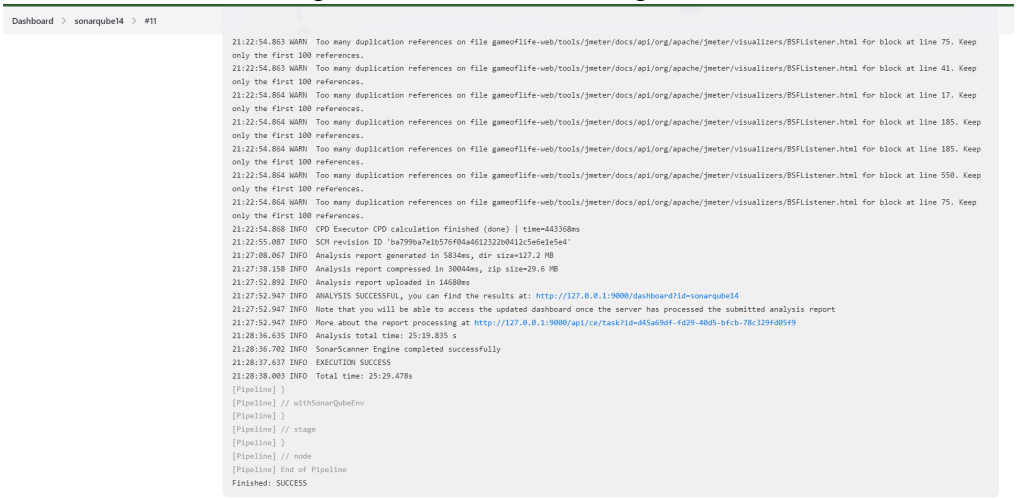
8. Run The Build.



NAME: VEDANT DHOKE CLASS/ ROLL NO: D15C/ 9

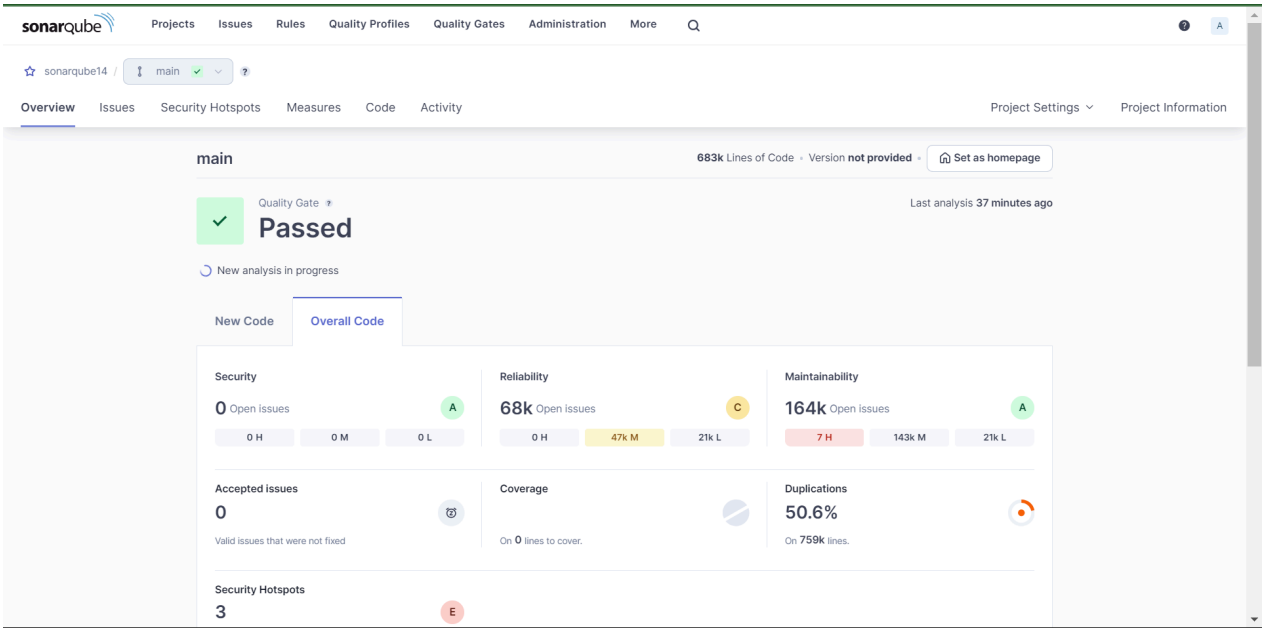



9. Check the console output once the build is complete.



NAME: VEDANT DHOKE CLASS/ ROLL NO: D15C/ 9

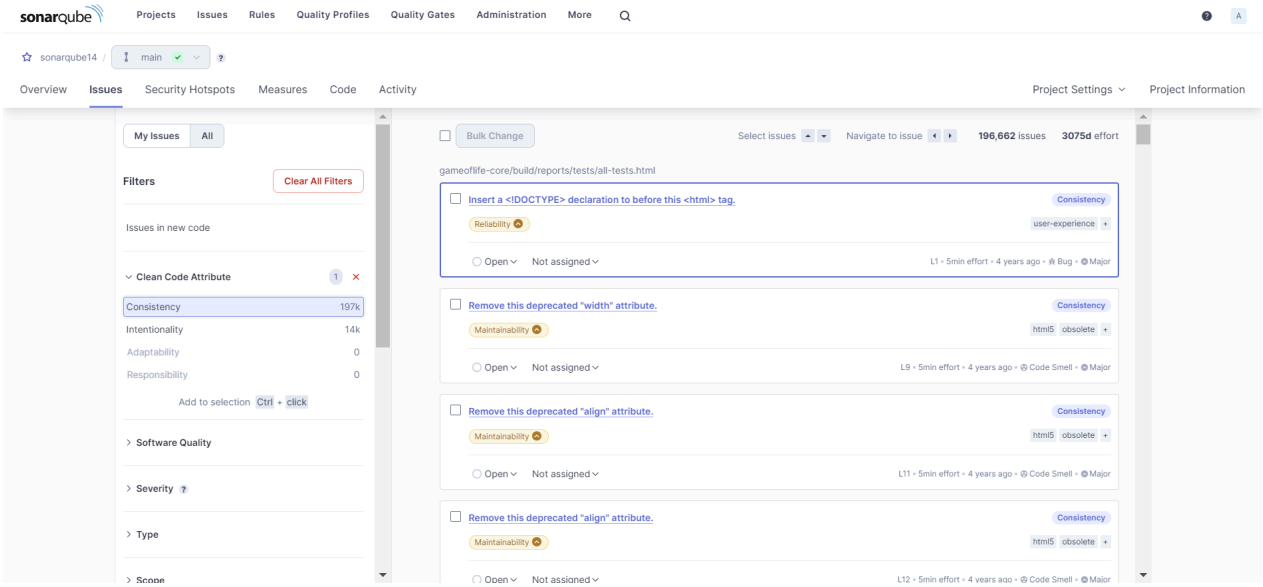
10. After that, check the project in SonarQube.




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

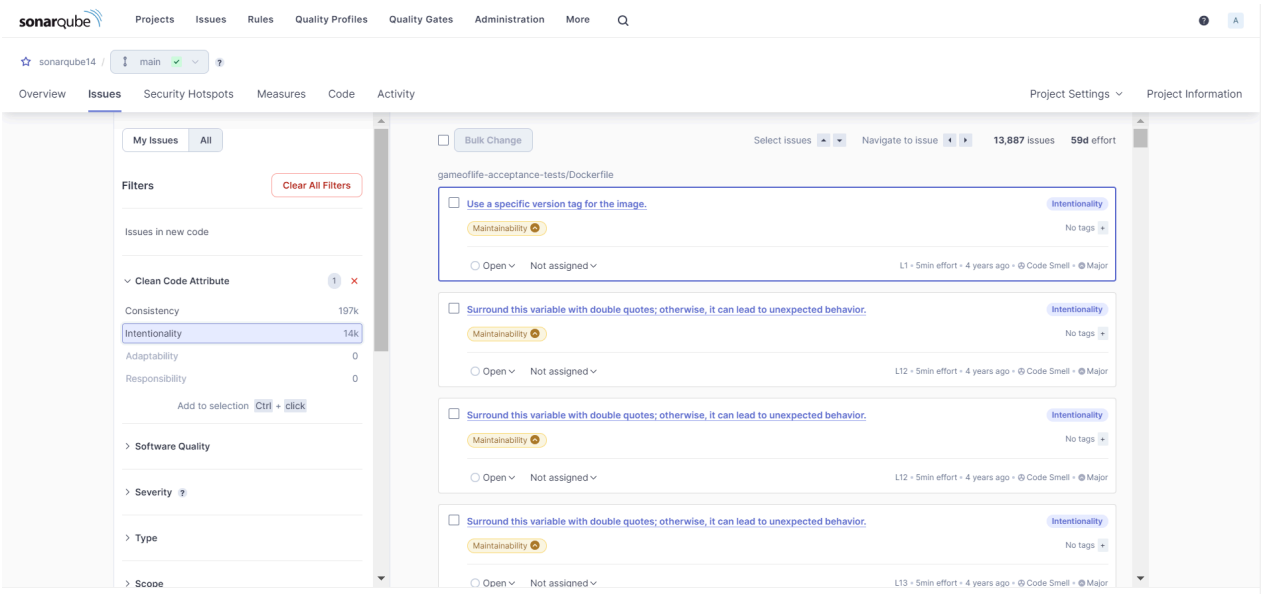
11. **Code Problems -**

**Consistency**

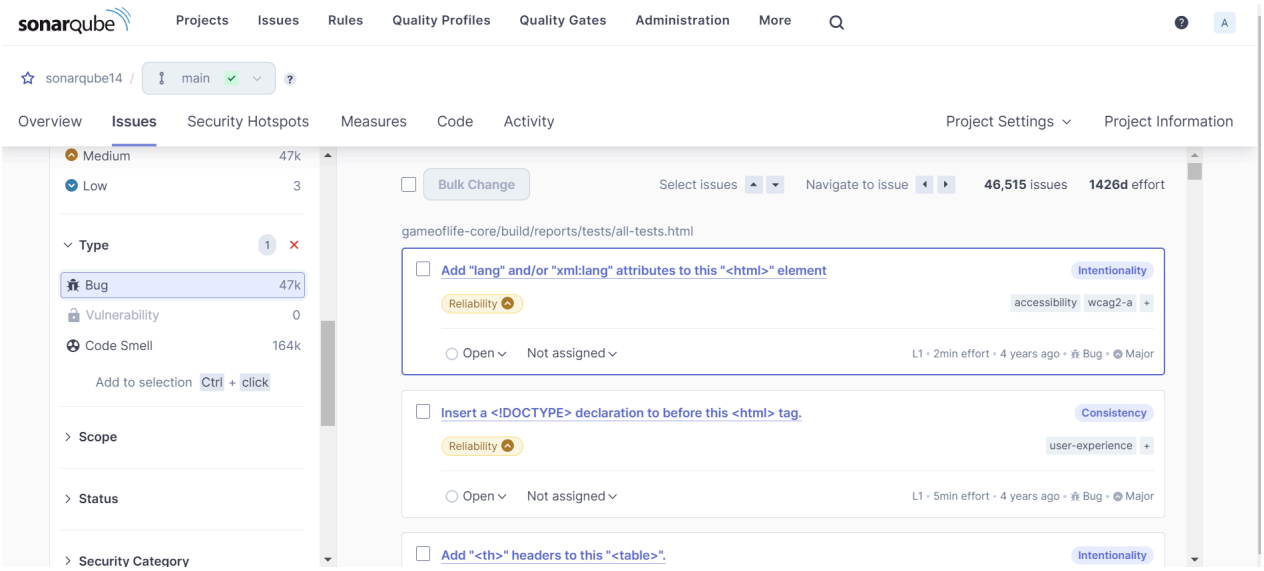


NAME: VEDANT DHOKE CLASS/ ROLL NO: D15C/ 9

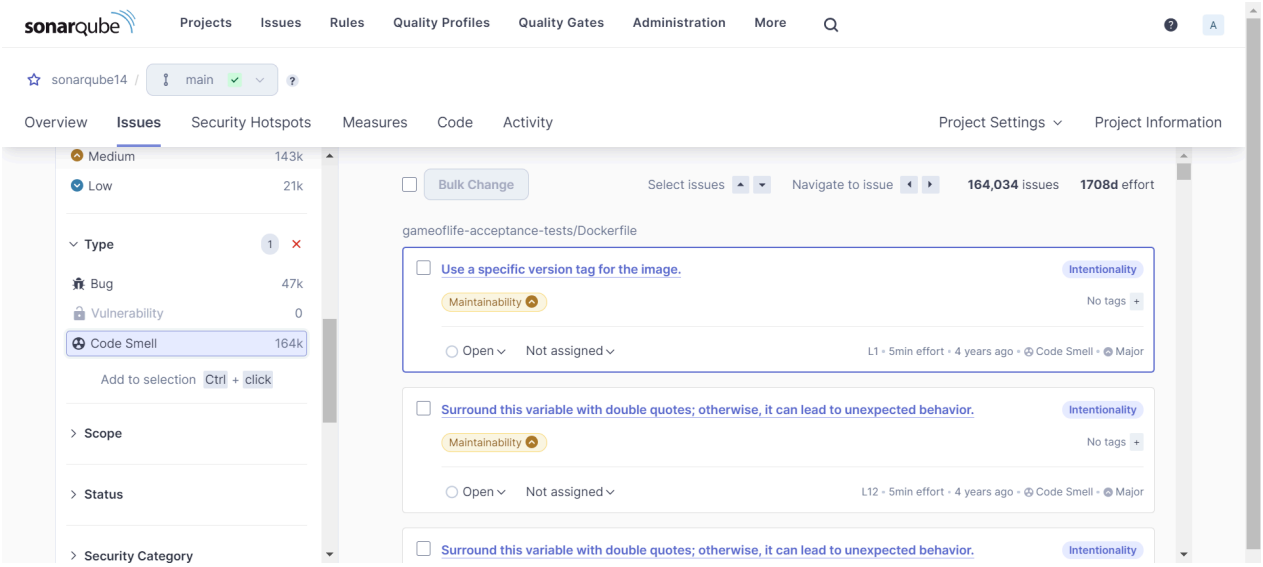
## Intentionality



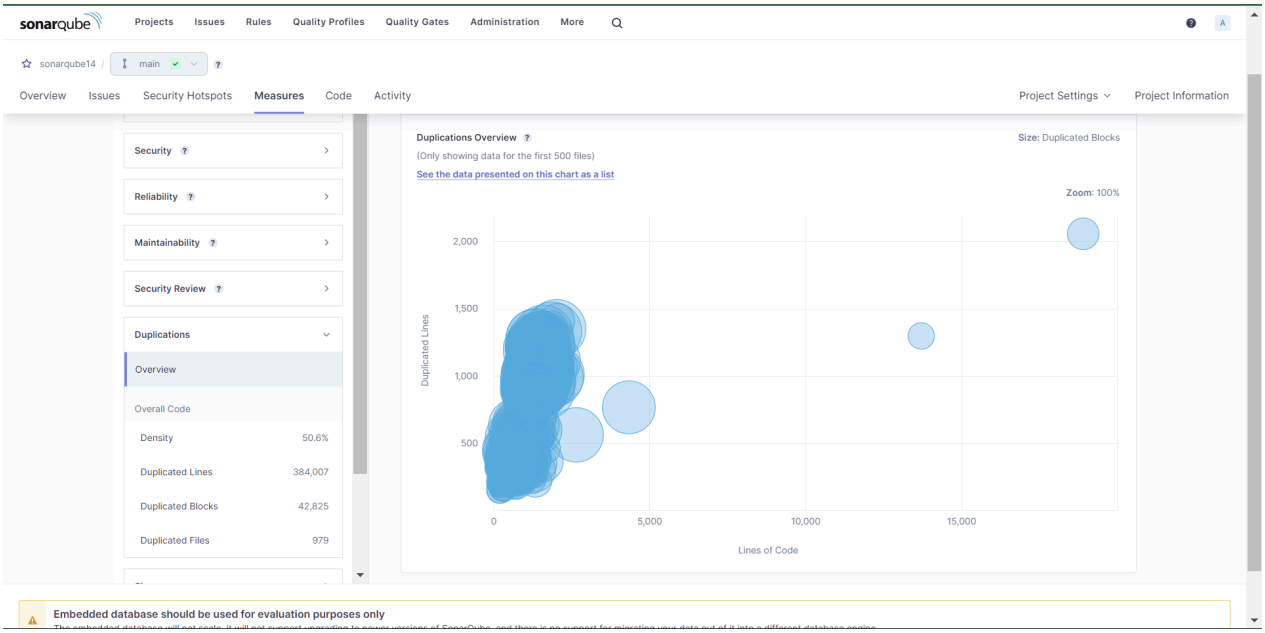

## Bugs and Code Smells



NAME: VEDANT DHOKE CLASS/ ROLL NO: D15C/ 9

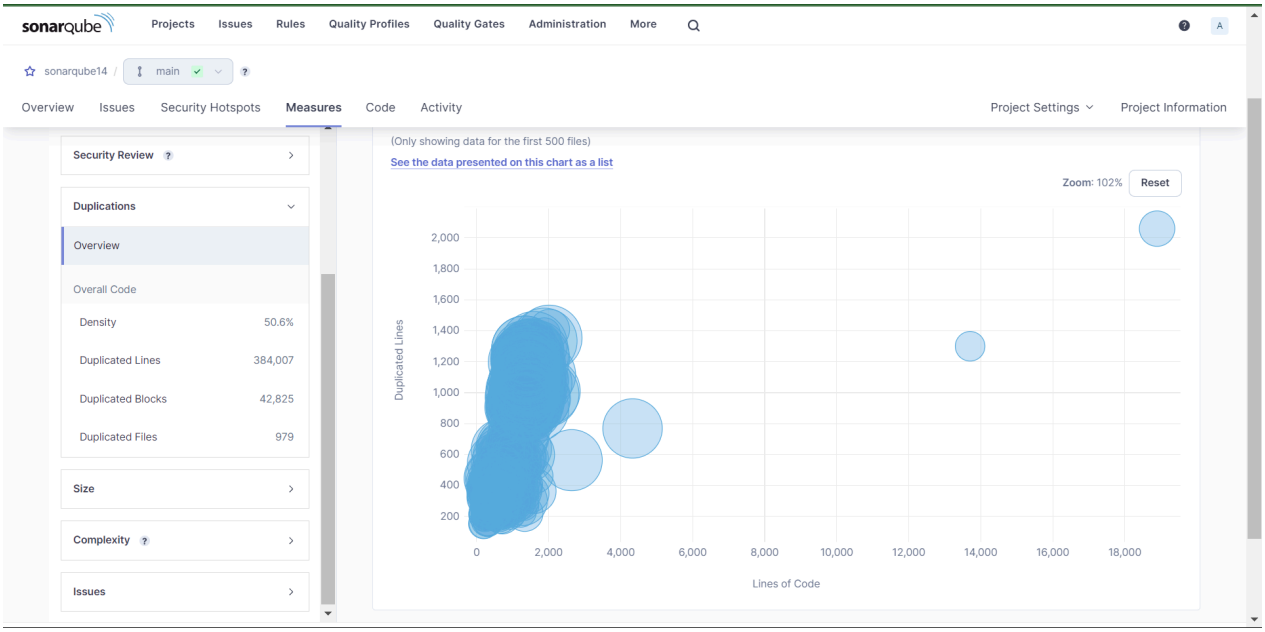



## Duplicates



NAME: VEDANT DHOKE CLASS/ ROLL NO: D15C/ 9

**Cyclomatic Complexities**




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

**Conclusion:**

In this experiment, we successfully cloned a GitHub repository and integrated it with SonarQube for comprehensive code analysis. SonarQube provided valuable insights into various types of program issues, such as:

Consistency: Detected non-adherence to coding standards and formatting rules.

Intentionality: Flagged potential logical or structural errors within the code.

Severity: Classified issues by their level of criticality (e.g., critical, major, minor).

Duplicates: Identified redundant code segments, suggesting potential optimization.

Cyclomatic Complexity: Measured the complexity of the code based on the number of control flow paths, highlighting sections that might be difficult to maintain or prone to errors.

**Issues Faced:**

**SonarQube Scanner Path Error:** Initially, Jenkins failed to detect the correct path for the SonarQube Scanner. This required manual intervention to adjust the pipeline script and specify the correct path for the scanner bash file, allowing the pipeline to execute successfully.

**SonarQube Login Issues:** Default credentials for logging into SonarQube (username: admin, password: admin) did not work due to a configuration reset or password change in a previous session. This necessitated troubleshooting and reconfiguration of access credentials.

**Slow Analysis Process:** During the scan of larger repositories, the SonarQube analysis process was notably slow.

NAME: VEDANT DHOKE CLASS/ ROLL NO: D15C/ 9

# AdvanceDevops Experiment No: 9

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

**Theory:**

**What is Nagios?**

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

**Why We Need Nagios tool?**

Here are the important reasons to use Nagios monitoring tool:
● Detects all types of network or server issues
● Helps you to find the root cause of the problem which allows you to get the permanent solution to the problem
● Active monitoring of your entire infrastructure and business processes
● Allows you to monitor and troubleshoot server performance issues
● Helps you to plan for infrastructure upgrades before outdated systems create failures
● You can maintain the security and availability of the service
● Automatically fix problems in a panic situation

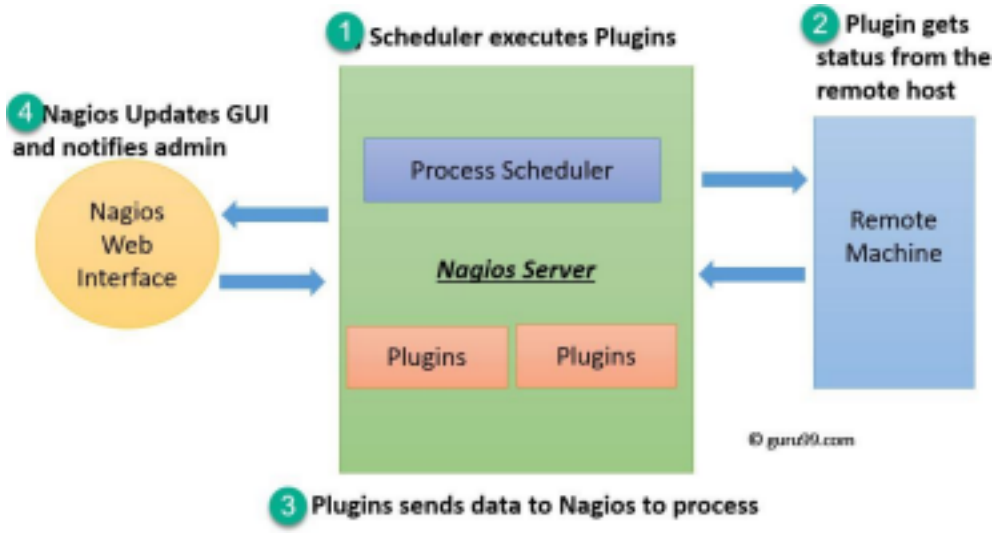
**Features of Nagios**
Following are the important features of Nagios monitoring tool:
● Relatively scalable, Manageable, and Secure
● Good log and database system
● Informative and attractive web interfaces
● Automatically send alerts if condition changes
● If the services are running fine, then there is no need to do check that host is an alive
● Helps you to detect network errors or server crashes
● You can troubleshoot the performance issues of the server.
● The issues, if any, can be fixed automatically as they are identified during the monitoring process
● You can monitor the entire business process and IT infrastructure with a single pass
● The product's architecture is easy to write new plugins in the language of your choice
● Nagios allows you to read its configuration from an entire directory which helps you to decide how to define individual files
● Utilizes topology to determine dependencies
● Monitor network services like HTTP, SMTP, HTTP, SNMP, FTP, SSH, POP, etc.

NAME: VEDANT DHOKE CLASS/ ROLL NO: D15C/ 9

● Helps you to define network host hierarchy using parent hosts
● Ability to define event handlers that runs during service or host events for proactive problem resolution
● Support for implementing redundant monitoring hosts

Nagios Architecture

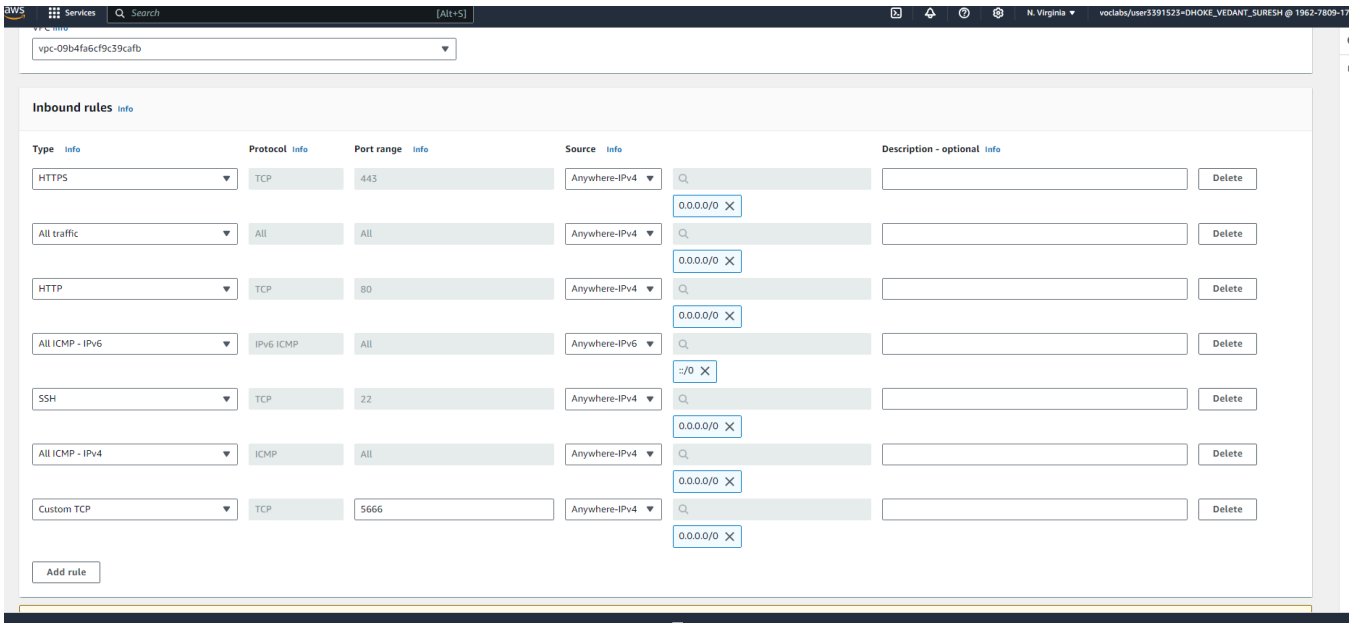
Nagios is a client-server architecture. Usually, on a network, a Nagios server is running on a host, and plugins are running on all the remote hosts which should be monitored.



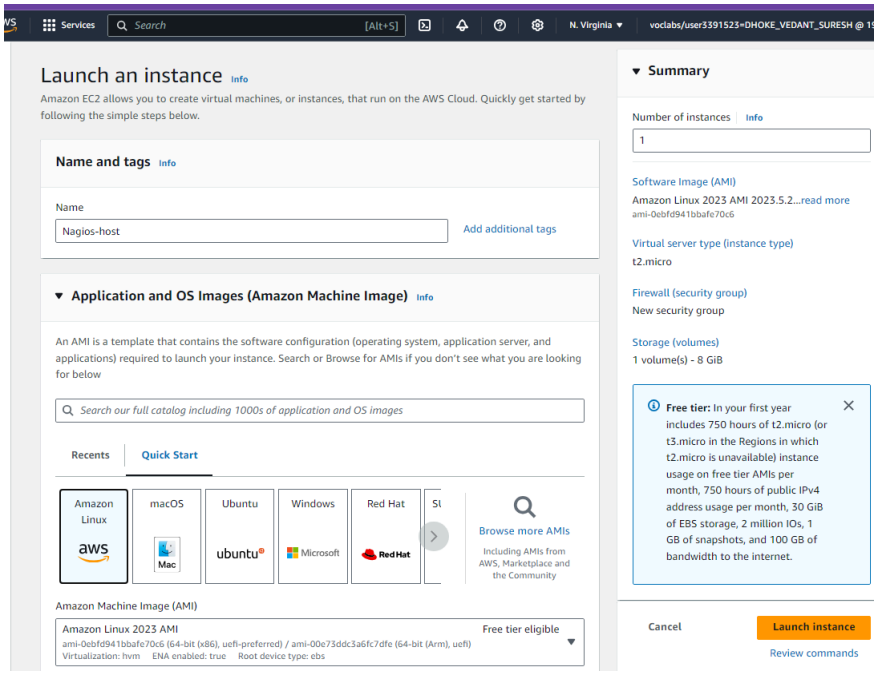

1. The scheduler is a component of the server part of Nagios. It sends a signal to execute the plugins at the remote host.
2. The plugin gets the status from the remote host
3. The plugin sends the data to the process scheduler
4. The process scheduler updates the GUI and notifications are sent to admins.

NAME: VEDANT DHOKE CLASS/ ROLL NO: D15C/ 9

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



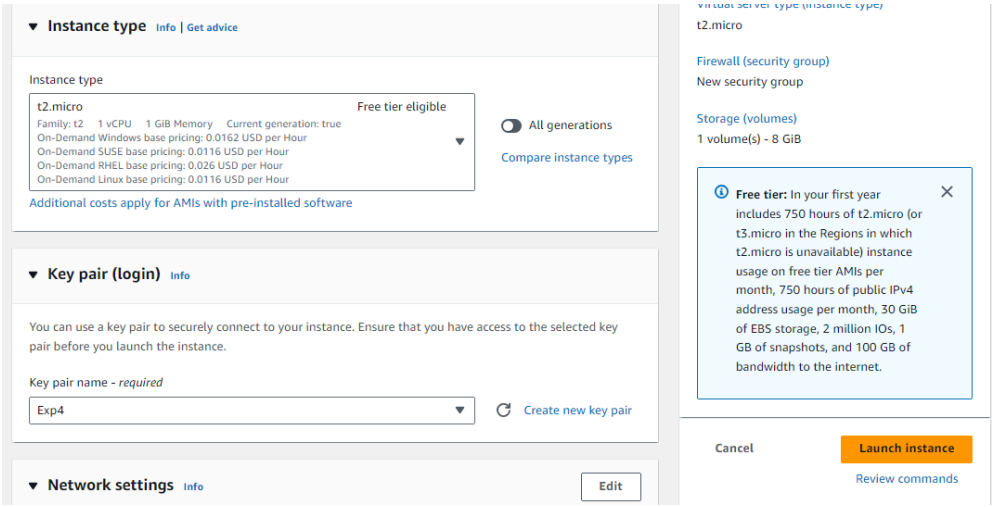

**Step 2:** Now Create a new EC2 instance. Name: Nagios-host ,AMI: Amazon Linux, Instance Type: t2.micro.



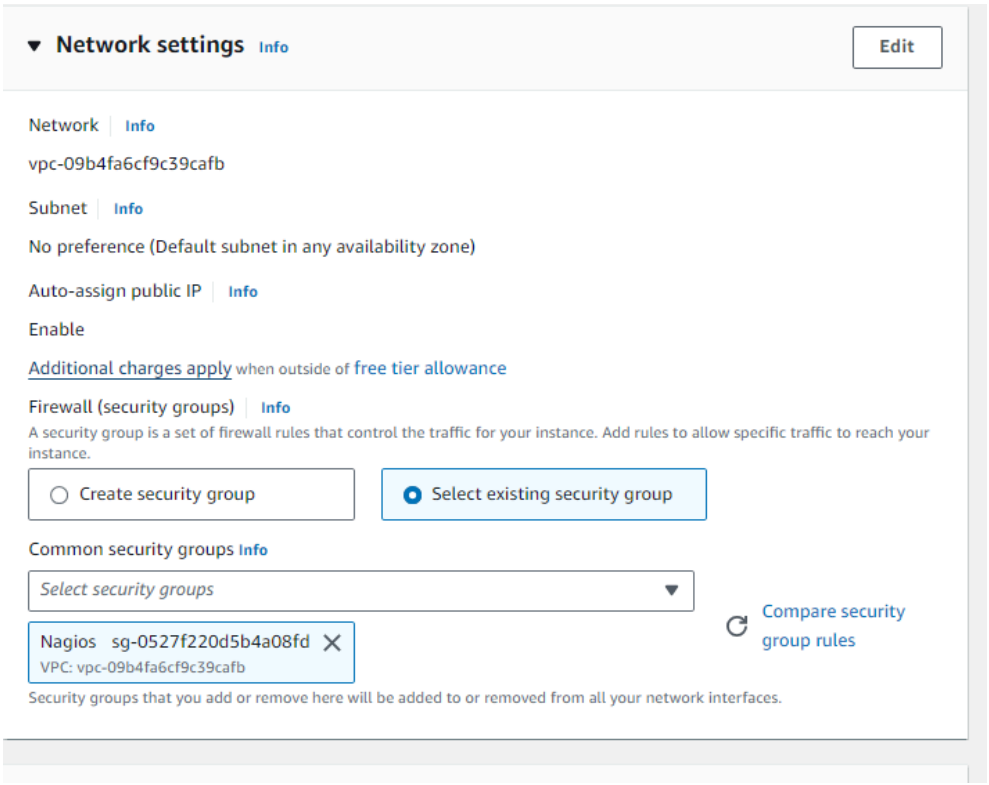
NAME: VEDANT DHOKE CLASS/ ROLL NO: D15C/ 9

**For Key pair :** Click on create key and make key of type RSA with extension .pem . Key will be downloaded to your local machine.

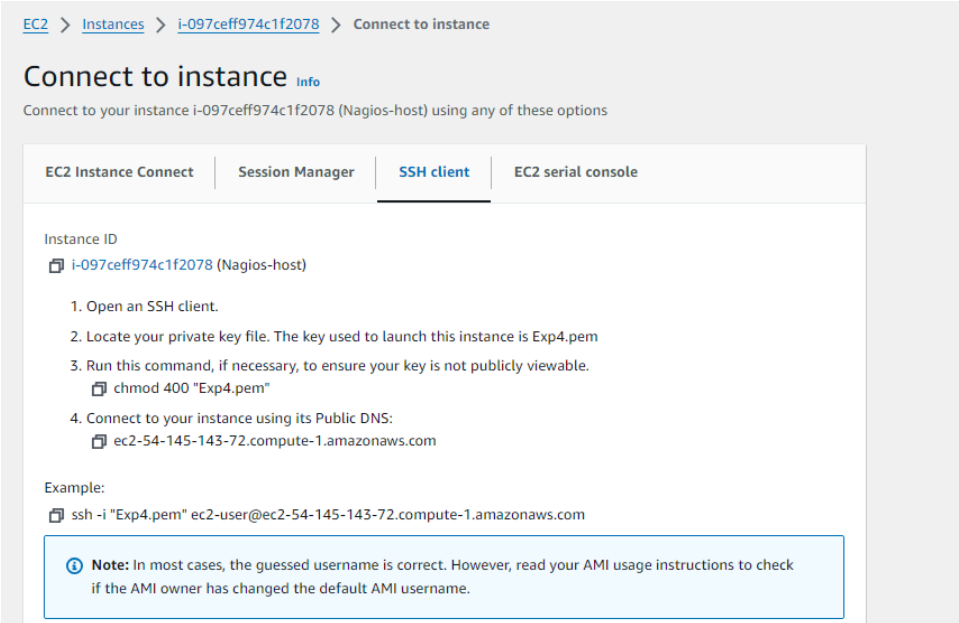
Now select that key in key pair if you already have key with type RSA and extension .pem no need to create new key but you must have that key downloaded.




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



**Step 3:** Now After creating the EC2 Instance click on connect and then copy the command which is given as example in the SSH Client section .




Now open the terminal in the folder where your key(RSA key with .pem) is located.and paste that copied command.

Successfully connected to the instance.

```
C:\Users\LENOVO>ssh -i "Exp4.pem" ec2-user@ec2-54-210-4-52.compute-1.amazonaws.com
The authenticity of host 'ec2-54-210-4-52.compute-1.amazonaws.com (54.210.4.52)' can't be establ
ished.
ED25519 key fingerprint is SHA256:YqoGBku3mptyrGnwBwnKC060wMjlNnInQm5MPBeB1RM.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-54-210-4-52.compute-1.amazonaws.com' (ED25519) to the list of kn
own hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-31-38-62 ~]$
```

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

```
[ec2-user@ip-172-31-38-62 ~]$ sudo adduser -m nagios
[ec2-user@ip-172-31-38-62 ~]$ sudo passwd nagios
Changing password for user nagios.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[ec2-user@ip-172-31-38-62 ~]$
```

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

```
[ec2-user@ip-172-31-38-62 ~]$ sudo groupadd nagcmd
[ec2-user@ip-172-31-38-62 ~]$ sudo usermod -a -G nagcmd nagios
sudo usermod -a -G nagcmd apache
[ec2-user@ip-172-31-38-62 ~]$
```

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

```
[ec2-user@ip-172-31-38-62 ~]$ mkdir ~/downloads
cd ~/downloads
```

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

```
[ec2-user@ip-172-31-38-62 downloads]$ wget https://go.nagios.org/l/975333/2024-09-17/6kqcx
--2024-10-02 07:05:23--  https://go.nagios.org/l/975333/2024-09-17/6kqcx
Resolving go.nagios.org (go.nagios.org)... 34.237.219.119, 3.92.120.28, 18.208.125.13, ...
Connecting to go.nagios.org (go.nagios.org)|34.237.219.119|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: http://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.5.tar.gz?utm_source=Nagios.org&utm_content=Download+Form&utm_camp
aign=Core+4.5.5+Download+&pi_content=1e9662c93afb2ed6bd2e3f3cc38771a7f01125e969f2a75b0e2254439d4a81d8 [following]
--2024-10-02 07:05:23--  http://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.5.tar.gz?utm_source=Nagios.org&utm_content=Downloa
d+Form&utm_campaign=Core+4.5.5+Download+&pi_content=1e9662c93afb2ed6bd2e3f3cc38771a7f01125e969f2a75b0e2254439d4a81d8
Resolving assets.nagios.com (assets.nagios.com)... 45.79.49.120, 2600:3c00::f03c:92ff:fef7:45ce
Connecting to assets.nagios.com (assets.nagios.com)|45.79.49.120|:80... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.5.tar.gz?utm_source=Nagios.org&utm_content=Download+Form&utm_cam
paign=Core+4.5.5+Download+&pi_content=1e9662c93afb2ed6bd2e3f3cc38771a7f01125e969f2a75b0e2254439d4a81d8 [following]
--2024-10-02 07:05:23--  https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.5.tar.gz?utm_source=Nagios.org&utm_content=Downlo
ad+Form&utm_campaign=Core+4.5.5+Download+&pi_content=1e9662c93afb2ed6bd2e3f3cc38771a7f01125e969f2a75b0e2254439d4a81d8
Connecting to assets.nagios.com (assets.nagios.com)|45.79.49.120|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2065473 (2.0M) [application/x-gzip]
Saving to: ‘6kqcx’

6kqcx                 100%[==============================>]   1.97M  6.65MB/s    in 0.3s

2024-10-02 07:05:24 (6.65 MB/s) - ‘6kqcx’ saved [2065473/2065473]

[ec2-user@ip-172-31-38-62 downloads]$
```

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

```
[ec2-user@ip-172-31-38-62 downloads]$ wget https://nagios-plugins.org/download/nagios-plugins-2.
4.11.tar.gz
--2024-10-02 07:05:58--  https://nagios-plugins.org/download/nagios-plugins-2.4.11.tar.gz
Resolving nagios-plugins.org (nagios-plugins.org)... 45.56.123.251
Connecting to nagios-plugins.org (nagios-plugins.org)|45.56.123.251|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2753049 (2.6M) [application/x-gzip]
Saving to: ‘nagios-plugins-2.4.11.tar.gz’

nagios-plugins-2.4.11.t 100%[==============================>]   2.62M  7.32MB/s    in 0.4s

2024-10-02 07:05:59 (7.32 MB/s) - ‘nagios-plugins-2.4.11.tar.gz’ saved [2753049/2753049]

[ec2-user@ip-172-31-38-62 downloads]$
```

**Step 8:** Now to extract the files from the downloaded Nagios 4.5.5 run the following command.
**tar zxvf 6kqcx**

```
[ec2-user@ip-172-31-38-62 downloads]$ tar zxvf 6kqcx
nagios-4.5.5/
nagios-4.5.5/.github/
nagios-4.5.5/.github/workflows/
nagios-4.5.5/.github/workflows/test.yml
nagios-4.5.5/.gitignore
nagios-4.5.5/CONTRIBUTING.md
nagios-4.5.5/Changelog
nagios-4.5.5/INSTALLING
nagios-4.5.5/LEGAL
nagios-4.5.5/LICENSE
nagios-4.5.5/Makefile.in
nagios-4.5.5/README.md
```

**Step 9:** Now change the directory to nagios-4.5.5 (Or which version you have downloaded)

```
[ec2-user@ip-172-31-38-62 downloads]$ cd nagios-4.5.5
```

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

```
[ec2-user@ip-172-31-38-62 nagios-4.5.5]$ ./configure --with-command-group=nagcmd
checking for a BSD-compatible install... /usr/bin/install -c
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether the compiler supports GNU C... yes
checking whether gcc accepts -g... yes
checking for gcc option to enable C11 features... none needed
checking whether make sets $(MAKE)... yes
checking whether ln -s works... yes
checking for strip... /usr/bin/strip
checking for sys/wait.h that is POSIX.1 compatible... yes
checking for stdio.h... yes
checking for stdlib.h... yes
checking for string.h... yes
checking for inttypes.h... yes
```

At the end we have found the error of cannot find ssl header .

```
checking for type of socket size... size_t
checking for Kerberos include files... configure: WARNING: could not find include files
checking for pkg-config... pkg-config
checking for SSL headers... configure: error: Cannot find ssl headers
[ec2-user@ip-172-31-38-62 nagios-4.5.5]$
```

So run following command to install ssl.

**sudo yum install openssl-devel**

```
[ec2-user@ip-172-31-38-62 nagios-4.5.5]$ sudo yum install openssl-devel
Last metadata expiration check: 0:10:57 ago on Wed Oct  2 06:57:40 2024.
Dependencies resolved.
================================================================================
 Package              Architecture   Version                       Repository           Size
================================================================================
Installing:
 openssl-devel        x86_64         1:3.0.8-1.amzn2023.0.14       amazonlinux         3.0 M

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

Total download size: 3.0 M
Installed size: 4.7 M
Is this ok [y/N]: y
Downloading Packages:
openssl-devel-3.0.8-1.amzn2023.0.14.x86_64.rpm                  21 MB/s | 3.0 MB     00:00
--------------------------------------------------------------------------------
Total                                                           16 MB/s | 3.0 MB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                        1/1
  Installing       : openssl-devel-1:3.0.8-1.amzn2023.0.14.x86_64           1/1
  Running scriptlet: openssl-devel-1:3.0.8-1.amzn2023.0.14.x86_64           1/1
  Verifying        : openssl-devel-1:3.0.8-1.amzn2023.0.14.x86_64           1/1

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

Complete!
[ec2-user@ip-172-31-38-62 nagios-4.5.5]$
```

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

```
[ec2-user@ip-172-31-38-62 nagios-4.5.5]$ ./configure --with-command-group=nagcmd
checking for a BSD-compatible install... /usr/bin/install -c
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether the compiler supports GNU C... yes
checking whether gcc accepts -g... yes
checking for gcc option to enable C11 features... none needed
checking whether make sets $(MAKE)... yes
checking whether ln -s works... yes
checking for strip... /usr/bin/strip
checking for sys/wait.h that is POSIX.1 compatible... yes
checking for stdio.h... yes
```

```
       web interface

  make install-classicui
     - This installs the classic theme for the Nagios
       web interface


*** Support Notes *******************************************

If you have questions about configuring or running Nagios,
please make sure that you:

     - Look at the sample config files
     - Read the documentation on the Nagios Library at:
          https://library.nagios.com

before you post a question to one of the mailing lists.
Also make sure to include pertinent information that could
help others help you.  This might include:

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

For more information on obtaining support for Nagios, visit:

       https://support.nagios.com

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

Enjoy.

[ec2-user@ip-172-31-38-62 nagios-4.5.5]$
```

**Step 11:** Now run the following commands to setup the Nagios.
**sudo make install**

NAME: VEDANT DHOKE CLASS/ ROLL NO: D15C/ 9

```
[ec2-user@ip-172-31-38-62 nagios-4.5.5]$ sudo make install
cd ./base && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/base'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagios /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagiostats /usr/local/nagios/bin
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5/base'
cd ./cgi && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
make install-basic
make[2]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/sbin
for file in *.cgi; do \
        /usr/bin/install -c -s -m 775 -o nagios -g nagios $file /usr/local/nagios/sbin; \
done
make[2]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
cd ./html && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/html'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/media
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/stylesheets
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/contexthelp
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/docs
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/docs/images
/usr/bin/install -c -m 775 -o nagios -g nagcmd -d /usr/local/nagios/var/spool/checkres
chmod g+s /usr/local/nagios/var/spool/checkresults

*** Main program, CGIs and HTML files installed ***

You can continue with installing Nagios as follows (type 'make'
without any arguments for a list of all possible options):

  make install-init
     - This installs the init script in /lib/systemd/system

  make install-commandmode
     - This installs and configures permissions on the
       directory for holding the external command file

  make install-config
     - This installs sample config files in /usr/local/nagios/etc

make[1]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5'
[ec2-user@ip-172-31-38-62 nagios-4.5.5]$
```

**sudo make install-init**

```
[ec2-user@ip-172-31-38-62 nagios-4.5.5]$ sudo make install-init
/usr/bin/install -c -m 755 -d -o root -g root /lib/systemd/system
/usr/bin/install -c -m 755 -o root -g root startup/default-service /lib/systemd/system/nagios.service
[ec2-user@ip-172-31-38-62 nagios-4.5.5]$ sudo make install-config
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc/objects
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/nagios.cfg /usr/local/nagios/etc/nagios.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/cgi.cfg /usr/local/nagios/etc/cgi.cfg
/usr/bin/install -c -b -m 660 -o nagios -g nagios sample-config/resource.cfg /usr/local/nagios/etc/resource.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/templates.cfg /usr/local/nagios/etc/objects/templates.
cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/commands.cfg /usr/local/nagios/etc/objects/commands.cf
g
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/contacts.cfg /usr/local/nagios/etc/objects/contacts.cf
g
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/timeperiods.cfg /usr/local/nagios/etc/objects/timeperi
ods.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/localhost.cfg /usr/local/nagios/etc/objects/localhost.
cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/windows.cfg /usr/local/nagios/etc/objects/windows.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/printer.cfg /usr/local/nagios/etc/objects/printer.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/switch.cfg /usr/local/nagios/etc/objects/switch.cfg

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

**sudo make install-webconf**

```
[ec2-user@ip-172-31-38-62 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-172-31-38-62 nagios-4.5.5]$
```

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

```
[ec2-user@ip-172-31-38-62 nagios-4.5.5]$ sudo htpasswd -c /usr/local/nagios/etc/htpasswd.use
rs nagiosadmin
New password:
Re-type new password:
Adding password for user nagiosadmin
[ec2-user@ip-172-31-38-62 nagios-4.5.5]$
```

Now to restart the httpd service run the following command.

**sudo service httpd restart**

```
[ec2-user@ip-172-31-38-62 nagios-4.5.5]$ sudo service httpd restart
Redirecting to /bin/systemctl restart httpd.service
```

**Step 12:** Now to extract the files from the downloaded Nagios plugin 2.4.11 run the following command first change the directory.

**cd ~/downloads**

**tar zxvf nagios-plugins-2.4.11.tar.gz**

```
[ec2-user@ip-172-31-38-62 nagios-4.5.5]$ cd ~/downloads
[ec2-user@ip-172-31-38-62 downloads]$ tar zxvf nagios-plugins-2.4.11.tar.gz
nagios-plugins-2.4.11/
nagios-plugins-2.4.11/build-aux/
nagios-plugins-2.4.11/build-aux/compile
nagios-plugins-2.4.11/build-aux/config.guess
nagios-plugins-2.4.11/build-aux/config.rpath
nagios-plugins-2.4.11/build-aux/config.sub
nagios-plugins-2.4.11/build-aux/install-sh
nagios-plugins-2.4.11/build-aux/ltmain.sh
nagios-plugins-2.4.11/build-aux/missing
nagios-plugins-2.4.11/build-aux/mkinstalldirs
nagios-plugins-2.4.11/build-aux/depcomp
nagios-plugins-2.4.11/build-aux/snippet/
```

**Step 13:** Now change the directory to nagios-plugins-2.4.11 and run the config command to configure.
**cd nagios-plugins-2.4.11**

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

```
[ec2-user@ip-172-31-38-62 nagios-plugins-2.4.11]$ ./configure --with-nagios-user=nagios --with-nagios-group=nagios
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /usr/bin/mkdir -p
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking whether make supports nested variables... yes
checking whether to enable maintainer-specific portions of Makefiles... yes
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
```

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

```
[ec2-user@ip-172-31-38-62 nagios-plugins-2.4.11]$ sudo chkconfig --add nagios
error reading information on service nagios: No such file or directory
[ec2-user@ip-172-31-38-62 nagios-plugins-2.4.11]$
```

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

```
[ec2-user@ip-172-31-38-62 nagios-plugins-2.4.11]$ sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

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

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

Running pre-flight check on configuration data...

Checking objects...
        Checked 8 services.
        Checked 1 hosts.
        Checked 1 host groups.
        Checked 0 service groups.
        Checked 1 contacts.
        Checked 1 contact groups.
        Checked 24 commands.
        Checked 5 time periods.
        Checked 0 host escalations.
        Checked 0 service escalations.
Checking for circular paths...
        Checked 1 hosts
        Checked 0 service dependencies
        Checked 0 host dependencies
        Checked 5 timeperiods
Checking global event handlers...
Checking obsessive compulsive processor commands...
Checking misc settings...

Total Warnings: 0
Total Errors:   0

Things look okay - No serious problems were detected during the pre-flight check
[ec2-user@ip-172-31-38-62 nagios-plugins-2.4.11]$
```

**cd**

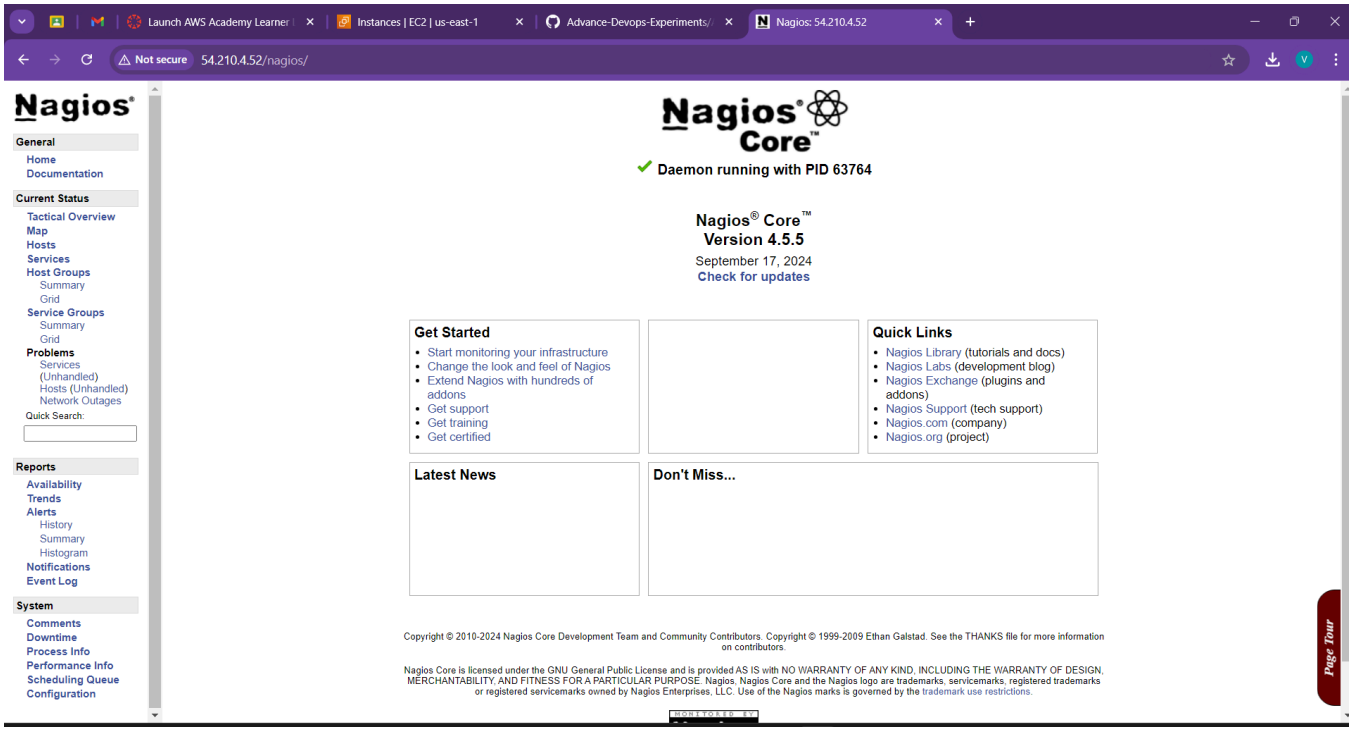
**sudo service nagios start**

```
[ec2-user@ip-172-31-38-62 nagios-plugins-2.4.11]$ cd
[ec2-user@ip-172-31-38-62 ~]$ sudo service nagios start
Redirecting to /bin/systemctl start nagios.service
[ec2-user@ip-172-31-38-62 ~]$
```

**sudo systemctl status nagios**

```
[ec2-user@ip-172-31-38-62 ~]$ sudo systemctl status nagios
● nagios.service - Nagios Core 4.5.5
     Loaded: loaded (/usr/lib/systemd/system/nagios.service; disabled; preset: disabled)
     Active: active (running) since Wed 2024-10-02 07:41:50 UTC; 5s ago
       Docs: https://www.nagios.org/documentation
    Process: 63762 ExecStartPre=/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg (c>
    Process: 63763 ExecStart=/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg (code>
   Main PID: 63764 (nagios)
      Tasks: 6 (limit: 1112)
     Memory: 5.4M
        CPU: 74ms
     CGroup: /system.slice/nagios.service
             ├─63764 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─63765 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─63766 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─63767 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─63768 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             └─63769 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg

Oct 02 07:41:50 ip-172-31-38-62.ec2.internal nagios[63764]: qh: core query handler registered
Oct 02 07:41:50 ip-172-31-38-62.ec2.internal nagios[63764]: qh: echo service query handler register>
Oct 02 07:41:50 ip-172-31-38-62.ec2.internal nagios[63764]: qh: help for the query handler register>
Oct 02 07:41:50 ip-172-31-38-62.ec2.internal nagios[63764]: wproc: Successfully registered manager >
Oct 02 07:41:50 ip-172-31-38-62.ec2.internal nagios[63764]: wproc: Registry request: name=Core Work>
Oct 02 07:41:50 ip-172-31-38-62.ec2.internal nagios[63764]: wproc: Registry request: name=Core Work>
Oct 02 07:41:50 ip-172-31-38-62.ec2.internal nagios[63764]: wproc: Registry request: name=Core Work>
Oct 02 07:41:50 ip-172-31-38-62.ec2.internal nagios[63764]: wproc: Registry request: name=Core Work>
Oct 02 07:41:51 ip-172-31-38-62.ec2.internal nagios[63764]: Successfully launched command file work>
Oct 02 07:41:51 ip-172-31-38-62.ec2.internal nagios[63764]: HOST ALERT: localhost;DOWN;SOFT;1;(No o>
lines 1-28/28 (END)
```

**Step 15:** We can see we have successfully launched the Nagios now . Open **http://<instance public ip >/nagios/** here it is **http://54.210.4.52/nagios** we can see the running web page of nagios.




**Conclusion:**

In this experiment, we successfully installed and configured Nagios Core, Nagios Plugins, and NRPE on a Linux machine within an AWS EC2 instance. The aim of continuously monitoring a remote system was achieved by integrating Nagios with the EC2 environment and allowing web access via the Nagios dashboard. We faced several challenges that required troubleshooting:

- **Security Group Configuration:** Setting up the correct inbound rules in the AWS security group was essential but prone to mistakes. Incorrectly configured ports could block HTTP and NRPE communication, preventing access to the Nagios dashboard or monitoring checks.

- **User and Group Permissions:** There were some issues when configuring user and group permissions, especially while adding users to the nagcmd group. If the commands weren't run correctly, Nagios failed to run properly due to incorrect access rights.

- **Dependencies and Package Installation:** While installing Nagios and its plugins, we encountered dependency issues, particularly with OpenSSL. Missing packages or libraries often halted the configuration process. Resolving these involved installing required dependencies and restarting the configuration steps.

# AdvanceDevOps Experiment No: 10

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

**Theory:**

**Port and Service Monitoring**
Port and service monitoring in Nagios involves checking the availability and responsiveness of network services running on specific ports. This ensures that critical services (like HTTP, FTP, or SSH) are operational. Nagios uses plugins to ping the ports and verify whether services are up and responding as expected, allowing administrators to be alerted in case of outages.

**Windows/Linux Server Monitoring**
Windows/Linux server monitoring with Nagios entails tracking the performance and health of servers running these operating systems. It includes monitoring metrics such as CPU usage, memory consumption, disk space, and system logs. Nagios employs various plugins to gather data, enabling administrators to ensure optimal performance, identify potential issues, and maintain uptime across their server infrastructure.

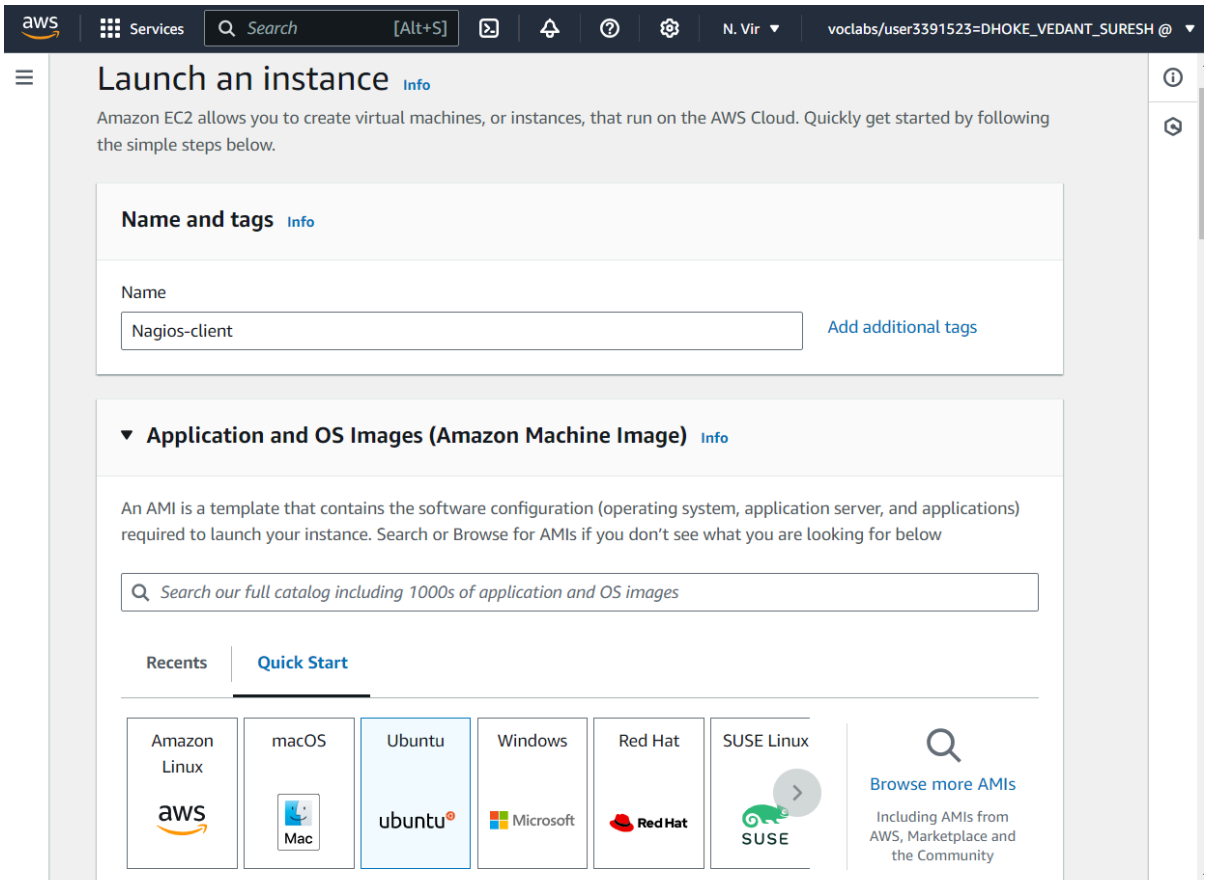
**Prerequisites:**
AWS Academy or Personal account.
Nagios Server running on Amazon Linux Machine. (Refer Experiment No 9)

**Monitoring Using Nagios:**

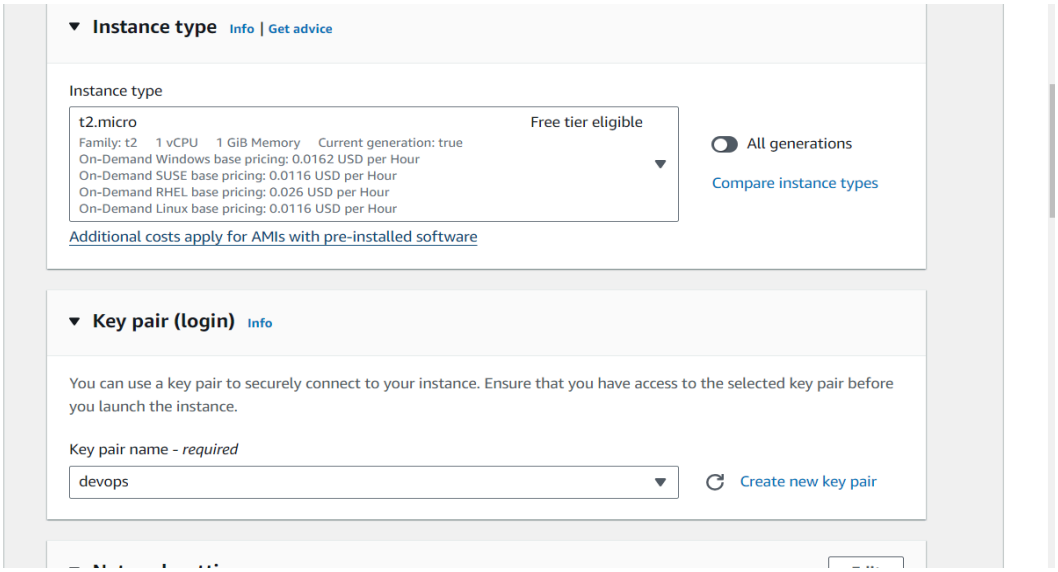
**Step 1:** To Confirm Nagios is running on the server side Perform the following command on your Amazon Linux Machine (Nagios-host).
**sudo systemctl status nagios**

You can now proceed if you get the above message/output.

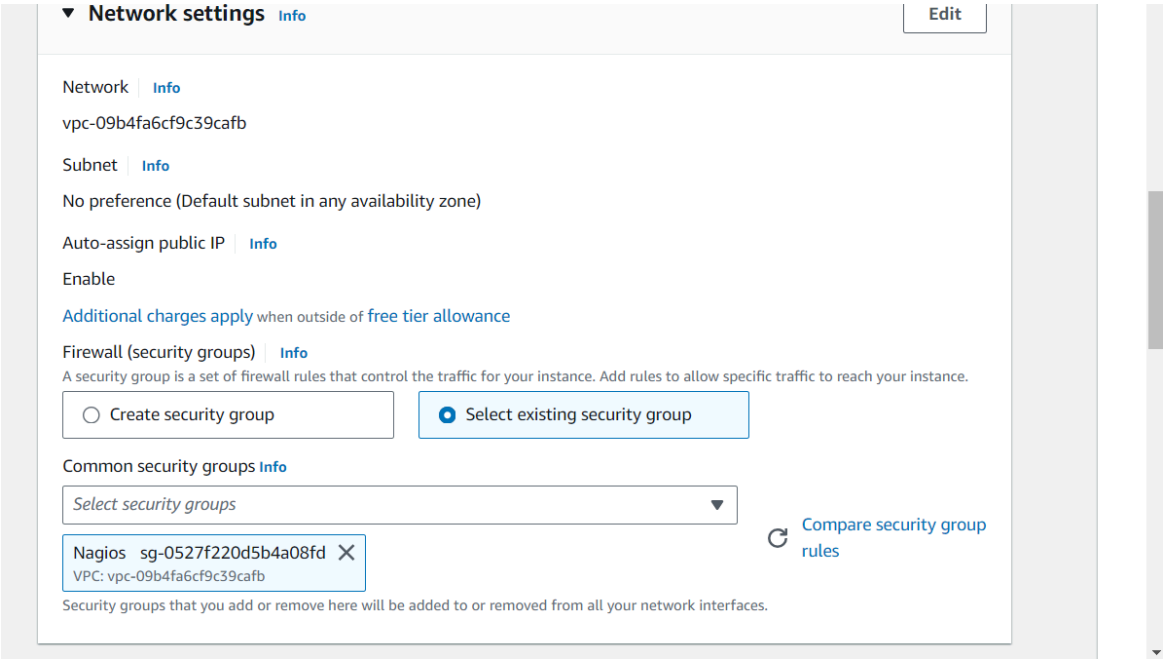
**Step 2:** Now Create a new EC2 instance. Name: Nagios-client, AMI: Ubuntu Instance Type: t2.micro.




**For Key pair :** Click on create key and make key of type RSA with extension .pem . Key will be downloaded to your local machine.



Select the Existing Security Group and select the Security Group that we have created in Experiment no 9 or the same one you have used for the Nagios server (Nagios-host).




**Step 3:** Now After creating the EC2 Instance click on connect and then copy the command which is given as example in the SSH Client section .
Now open the terminal in the folder where your key(RSA key with .pem) is located. and paste that copied command.Successfully connected to the instance.

```
PS C:\Users\Vedant> ssh -i "devops.pem" ubuntu@ec2-3-81-218-241.compute-1.amazonaws.com
The authenticity of host 'ec2-3-81-218-241.compute-1.amazonaws.com (3.81.218.241)' can't be establish
ed.
ED25519 key fingerprint is SHA256:7YtdUbwcFY6vK575h5DIfKqnlOf22OVC34blKsmOQcw.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-81-218-241.compute-1.amazonaws.com' (ED25519) to the list of known
hosts.
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-1016-aws x86_64)

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

 System information as of Fri Sep 27 08:38:26 UTC 2024

  System load:    1.36      Processes:             26
  Usage of /home: unknown   Users logged in:       0
  Memory usage:   4%        IPv4 address for eth0: 10.10.10.2
  Swap usage:     0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

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


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


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

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

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

**Now perform all the commands on the Nagios-host till step 10**

**Step 4:** Now on the server Nagios-host run the following command.

**ps -ef | grep nagios**

```
[ec2-user@ip-172-31-46-196 ~]$ ps -ef | grep nagios
nagios      2428       1  0 11:05 ?        00:00:00 /usr/local/nagios/bin/nagios -d /usr/local/nagio
/etc/nagios.cfg
nagios      2430    2428  0 11:05 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local
nagios/var/rw/nagios.qh
nagios      2431    2428  0 11:05 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local
nagios/var/rw/nagios.qh
nagios      2432    2428  0 11:05 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local
nagios/var/rw/nagios.qh
nagios      2433    2428  0 11:05 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local
nagios/var/rw/nagios.qh
nagios      2437    2428  0 11:05 ?        00:00:00 /usr/local/nagios/bin/nagios -d /usr/local/nagio
/etc/nagios.cfg
ec2-user    3367    3273  0 11:16 pts/0    00:00:00 grep --color=auto nagios
[ec2-user@ip-172-31-46-196 ~]$
```

**Step 5:** Now Become root user and create root directories.

**sudo su**

**mkdir /usr/local/nagios/etc/objects/monitorhosts**

**mkdir /usr/local/nagios/etc/objects/monitorhosts/linuxhosts**

```
[ec2-user@ip-172-31-46-196 ~]$ sudo su
[root@ip-172-31-46-196 ec2-user]# mkdir /usr/local/nagios/etc/objects/monitorhosts
[root@ip-172-31-46-196 ec2-user]# mkdir /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
[root@ip-172-31-46-196 ec2-user]#
```

**6:** Copy the sample localhost.cfg to linuxhost.cfg by running the following command.(Below command should come in one line see screenshot below)

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

```
[root@ip-172-31-46-196 ec2-user]# cp /usr/local/nagios/etc/objects/localhost.cfg /usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg
[root@ip-172-31-46-196 ec2-user]#
```

**Step 7:**Open linuxserver.cfg using nano and make the following changes in all positions?everywhere in file.

Change `hostname` to **linuxserver**.

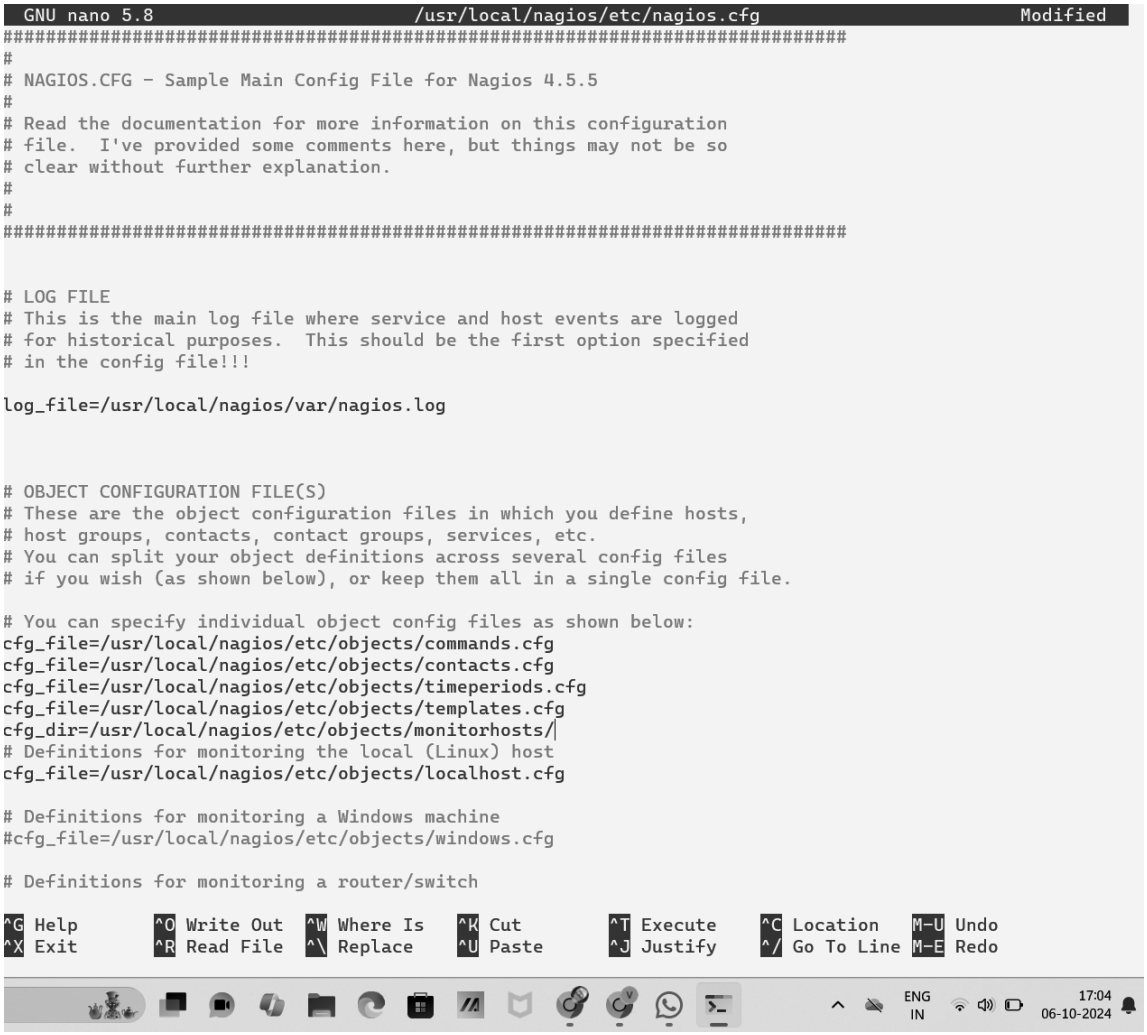
Change `address` to the public IP of your Linux client.

Set `hostgroup_name` to **linux-servers1**.

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


**Step 8:** Now update the Nagios config file .Add the following line in the file.

**Line to add : cfg_dir=/usr/local/nagios/etc/objects/monitorhosts/**
Run the command : **nano /usr/local/nagios/etc/nagios.cfg**




**Step 9:** Now Verify the configuration files by running the following commands.
**/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg**

```
[root@ip-172-31-46-196 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...
Warning: Duplicate definition found for service 'HTTP' on host 'localhost' (config file '/usr/local/nagios
fg', starting on line 152)
Warning: Duplicate definition found for service 'SSH' on host 'localhost' (config file '/usr/local/nagios/
g', starting on line 138)
Warning: Duplicate definition found for service 'Swap Usage' on host 'localhost' (config file '/usr/local/
rver.cfg', starting on line 125)
Warning: Duplicate definition found for service 'Current Load' on host 'localhost' (config file '/usr/loca
server.cfg', starting on line 112)
Warning: Duplicate definition found for service 'Total Processes' on host 'localhost' (config file '/usr/l
nuxserver.cfg', starting on line 100)
Warning: Duplicate definition found for service 'Current Users' on host 'localhost' (config file '/usr/loc
xserver.cfg', starting on line 86)
Warning: Duplicate definition found for service 'Root Partition' on host 'localhost' (config file '/usr/lo
uxserver.cfg', starting on line 72)
Warning: Duplicate definition found for service 'PING' on host 'localhost' (config file '/usr/local/nagios
fg', starting on line 58)
   Read object config files okay...

Running pre-flight check on configuration data...

Checking objects...
        Checked 8 services.
        Checked 2 hosts.
        Checked 2 host groups.
        Checked 0 service groups.
        Checked 1 contacts.
        Checked 1 contact groups.
        Checked 24 commands.
        Checked 5 time periods.
        Checked 0 host escalations.
        Checked 0 service escalations.
Checking for circular paths...
        Checked 2 hosts
        Checked 0 service dependencies
        Checked 0 host dependencies
        Checked 5 timeperiods
Checking global event handlers...
Checking obsessive compulsive processor commands...
Checking misc settings...

Total Warnings: 0
Total Errors:   0

[root@ip-172-31-46-196 ec2-user]#
```

**Step 10:** Now restart the services of nagios by running the following command.
**service nagios restart**

```
[root@ip-172-31-46-196 ec2-user]# service nagios restart
Redirecting to /bin/systemctl restart nagios.servicetart
[root@ip-172-31-46-196 ec2-user]#
```

**Step 11:** Now Go to the Nagios-client ssh terminal and update and install the packages by running the following command.

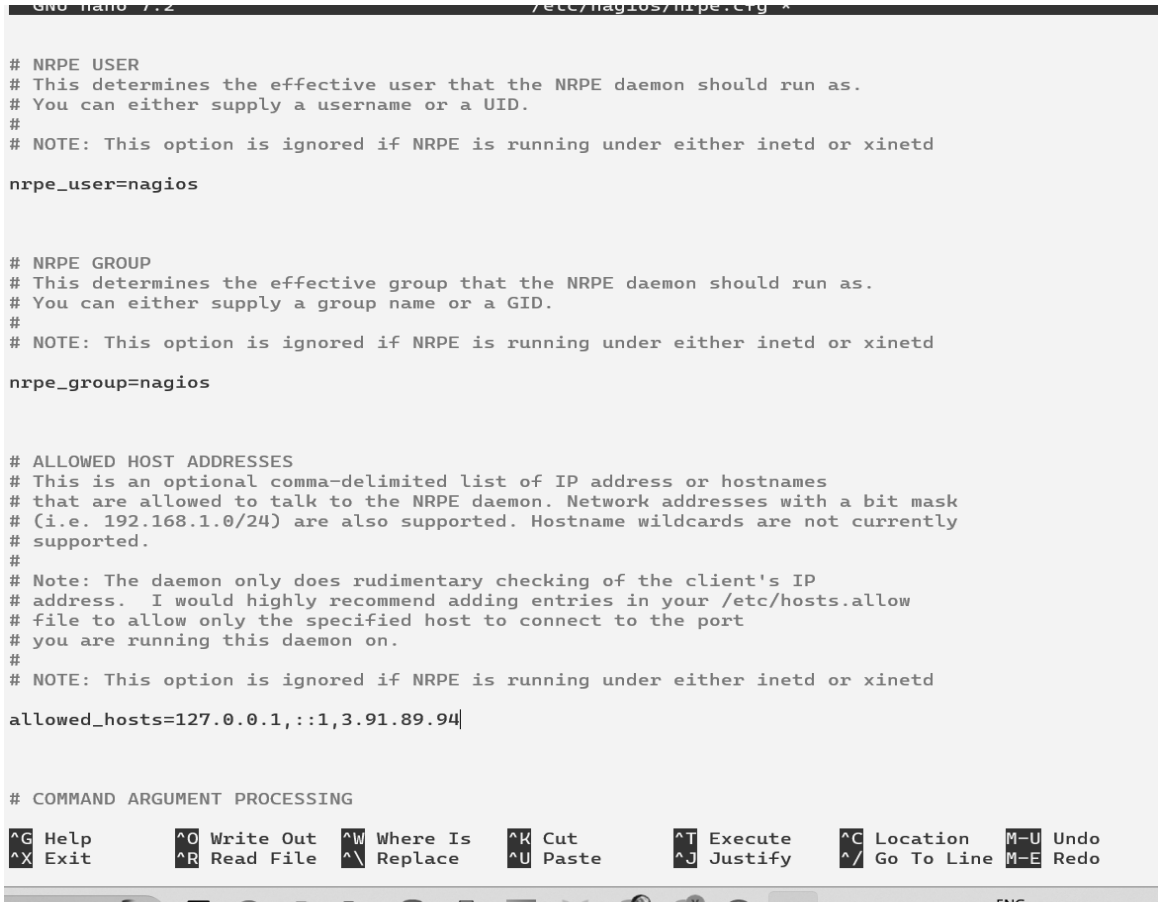
NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/ 9

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

```
ubuntu@ip-172-31-37-150:~$ sudo apt update -y
sudo apt install gcc -y
sudo apt install -y nagios-nrpe-server nagios-plugins
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:6 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [382 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:8 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [83.9 kB]
Get:9 http://security.ubuntu.com/ubuntu noble-security/main amd64 c-n-f Metadata [4704 B]
Get:10 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [277 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:12 http://security.ubuntu.com/ubuntu noble-security/universe Translation-en [117 kB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
Get:16 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [8632 B]
Get:17 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Components [35.0 kB]
Get:18 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 c-n-f Metadata [8328 B]
Get:19 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [537 kB]
Get:20 http://security.ubuntu.com/ubuntu noble-security/universe amd64 c-n-f Metadata [10.4 kB]
Get:21 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Packages [10.9 kB]
```

```
Setting up python3-ldb (2:2.8.0+samba4.19.5+dfsg-4ubuntu9) ...
Setting up samba-dsdb-modules:amd64 (2:4.19.5+dfsg-4ubuntu9) ...
Setting up libsmbclient0:amd64 (2:4.19.5+dfsg-4ubuntu9) ...
Setting up libcups2t64:amd64 (2.4.7-1.2ubuntu7.3) ...
Setting up python3-samba (2:4.19.5+dfsg-4ubuntu9) ...
Setting up smbclient (2:4.19.5+dfsg-4ubuntu9) ...
Setting up samba-common-bin (2:4.19.5+dfsg-4ubuntu9) ...
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for libc-bin (2.39-0ubuntu8.3) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-37-150:~$
```

**Step 12:** Open nrpe.cfg file to make changes.Under allowed_hosts, add your nagios host IP address.
**sudo nano /etc/nagios/nrpe.cfg**




**Step 13:** Now restart the NRPE server by this command.
**sudo systemctl restart nagios-nrpe-server**

```
ubuntu@ip-172-31-37-150:~$ sudo systemctl restart nagios-nrpe-server
ubuntu@ip-172-31-37-150:~$
```

**Step 14:** Now again check the status of Nagios by running this command on Nagios-host and also check httpd is active and run the command to active it.
**sudo systemctl status nagios**

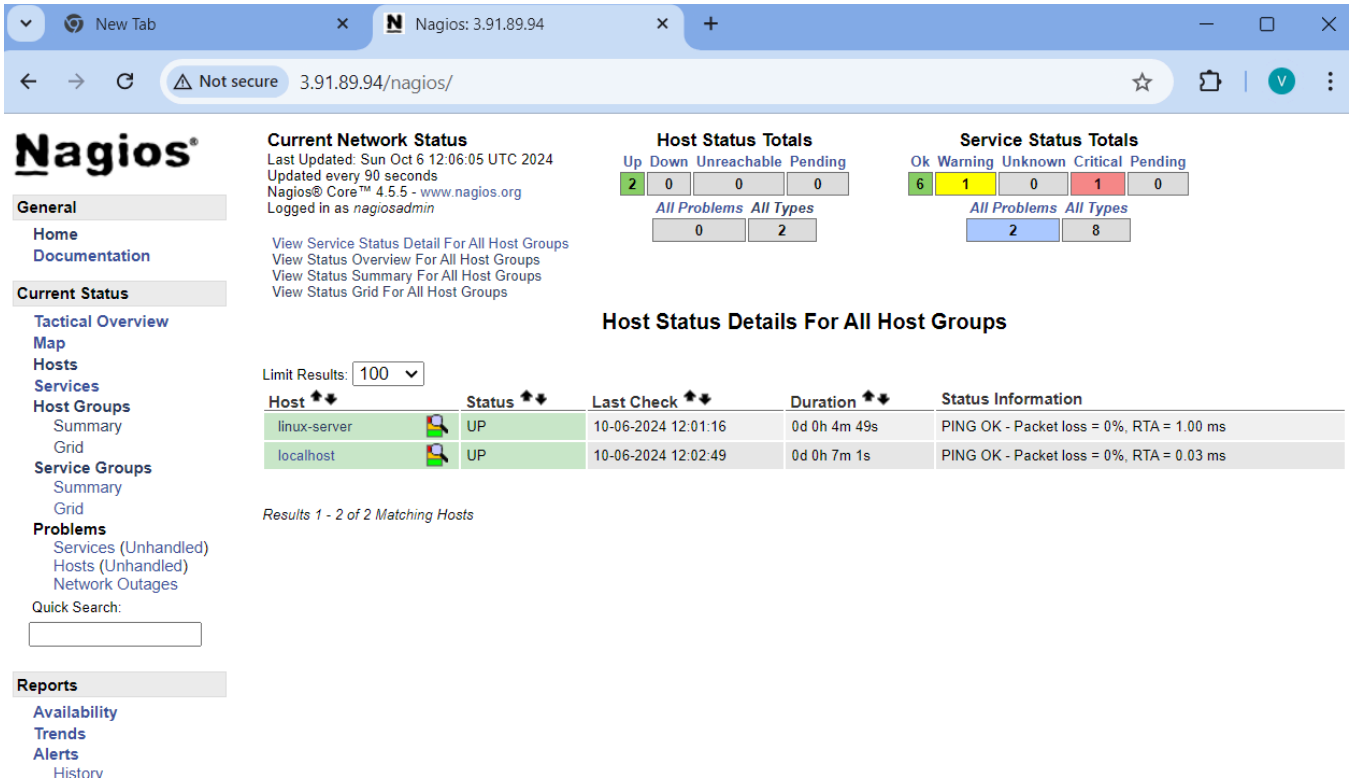
```
[root@ip-172-31-46-196 ec2-user]# sudo systemctl status nagios
● nagios.service - Nagios Core 4.5.5
     Loaded: loaded (/usr/lib/systemd/system/nagios.service; disabled; preset: disabled)
     Active: active (running) since Sun 2024-10-06 11:37:16 UTC; 9min ago
       Docs: https://www.nagios.org/documentation
    Process: 4481 ExecStartPre=/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg (code=exited, st>
    Process: 4482 ExecStart=/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg (code=exited, statu>
   Main PID: 4488 (nagios)
      Tasks: 6 (limit: 1112)
     Memory: 4.1M
        CPU: 108ms
     CGroup: /system.slice/nagios.service
             ├─4488 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─4489 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─4490 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─4491 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─4492 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             └─4497 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg

Oct 06 11:37:16 ip-172-31-46-196.ec2.internal nagios[4488]: HOST ALERT: linux-server;DOWN;SOFT;1;(No output on s>
Oct 06 11:38:16 ip-172-31-46-196.ec2.internal nagios[4488]: HOST ALERT: linux-server;DOWN;SOFT;2;(No output on s>
Oct 06 11:39:16 ip-172-31-46-196.ec2.internal nagios[4488]: HOST ALERT: linux-server;DOWN;SOFT;3;(No output on s>
Oct 06 11:40:16 ip-172-31-46-196.ec2.internal nagios[4488]: HOST ALERT: linux-server;DOWN;SOFT;4;(No output on s>
Oct 06 11:41:16 ip-172-31-46-196.ec2.internal nagios[4488]: HOST ALERT: linux-server;DOWN;SOFT;5;(No output on s>
Oct 06 11:42:16 ip-172-31-46-196.ec2.internal nagios[4488]: HOST ALERT: linux-server;DOWN;SOFT;6;(No output on s>
Oct 06 11:43:16 ip-172-31-46-196.ec2.internal nagios[4488]: HOST ALERT: linux-server;DOWN;SOFT;7;(No output on s>
Oct 06 11:44:16 ip-172-31-46-196.ec2.internal nagios[4488]: HOST ALERT: linux-server;DOWN;SOFT;8;(No output on s>
Oct 06 11:45:16 ip-172-31-46-196.ec2.internal nagios[4488]: HOST ALERT: linux-server;DOWN;SOFT;9;(No output on s>
Oct 06 11:46:16 ip-172-31-46-196.ec2.internal nagios[4488]: HOST ALERT: linux-server;DOWN;HARD;10;(No output on >
lines 1-28/28 (END)
```

**sudo systemctl status httpd**
**sudo systemctl start httpd**
**sudo systemctl enable httpd**

```
[root@ip-172-31-46-196 ec2-user]# sudo systemctl status httpd
● httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset: disabled)
    Drop-In: /usr/lib/systemd/system/httpd.service.d
             └─php-fpm.conf
     Active: active (running) since Sun 2024-10-06 11:08:08 UTC; 42min ago
       Docs: man:httpd.service(8)
   Main PID: 2546 (httpd)
     Status: "Total requests: 48; Idle/Busy workers 100/0;Requests/sec: 0.0188; Bytes served/sec: 121 B/sec"
      Tasks: 230 (limit: 1112)
     Memory: 25.1M
        CPU: 1.834s
     CGroup: /system.slice/httpd.service
             ├─2546 /usr/sbin/httpd -DFOREGROUND
             ├─2548 /usr/sbin/httpd -DFOREGROUND
             ├─2554 /usr/sbin/httpd -DFOREGROUND
             ├─2555 /usr/sbin/httpd -DFOREGROUND
             ├─2556 /usr/sbin/httpd -DFOREGROUND
             └─2889 /usr/sbin/httpd -DFOREGROUND

Oct 06 11:08:07 ip-172-31-46-196.ec2.internal systemd[1]: Starting httpd.service - The Apache HTTP Server...
Oct 06 11:08:08 ip-172-31-46-196.ec2.internal systemd[1]: Started httpd.service - The Apache HTTP Server.
Oct 06 11:08:08 ip-172-31-46-196.ec2.internal httpd[2546]: Server configured, listening on: port 80
[root@ip-172-31-46-196 ec2-user]# sudo systemctl start httpd
[root@ip-172-31-46-196 ec2-user]# sudo systemctl enable httpd
[root@ip-172-31-46-196 ec2-user]#
```

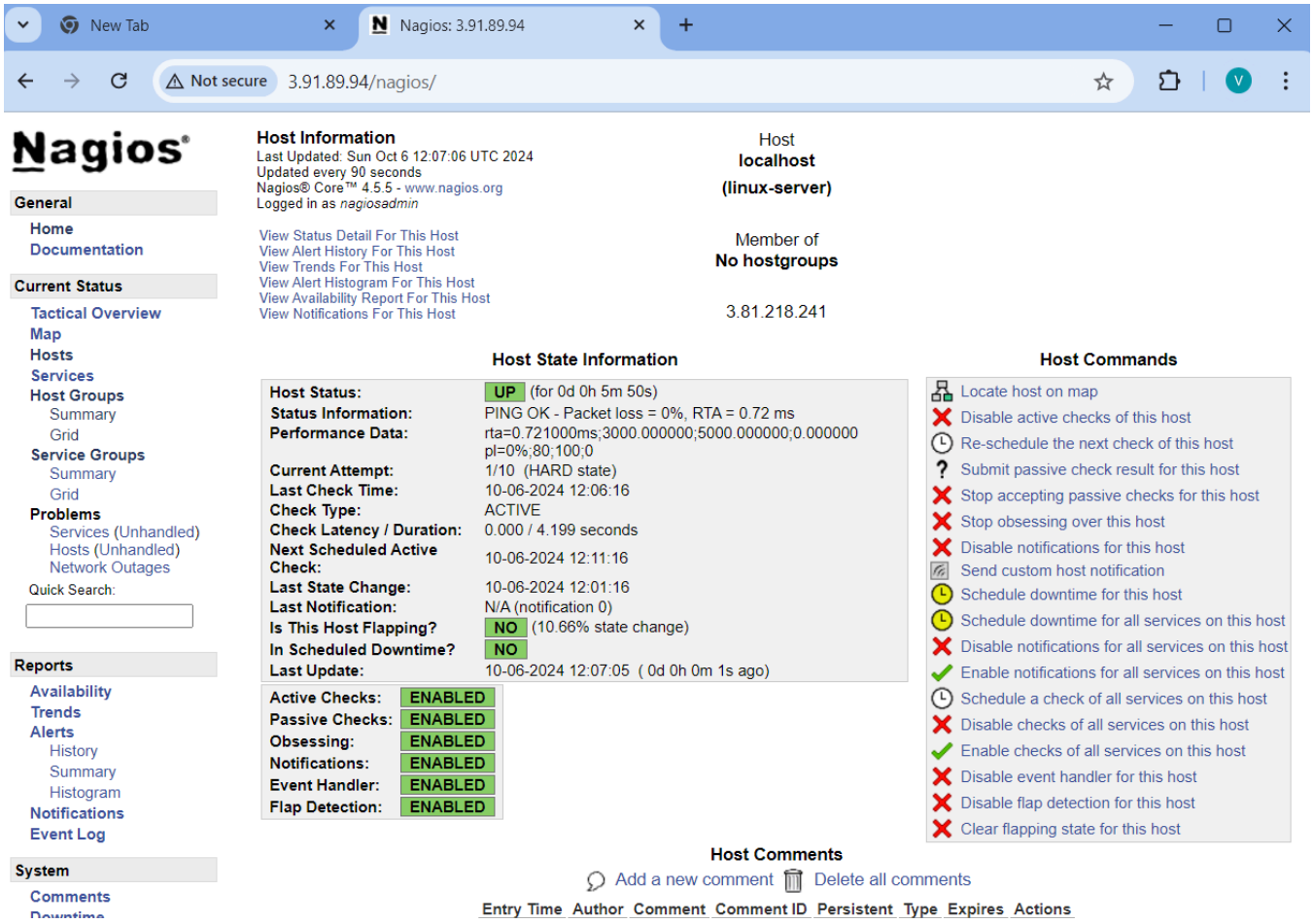
**Step 15:** Now to check Nagios dashboard go to http://<Nagios-host ip>/nagios .
**Now Click on Hosts from left side panel**

NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/ 9

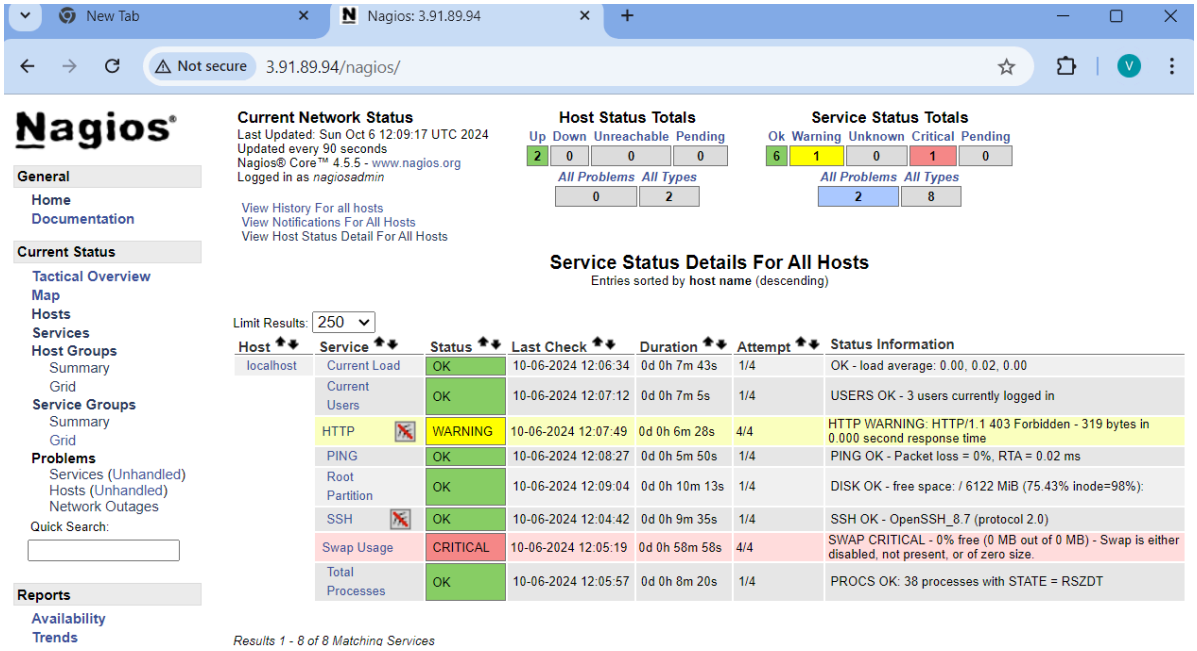


NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/ 9

**We can see our linuxserver now click on it we can see the host information.**




**Current Network Status**




**Conclusion:**

In this experiment, we successfully implemented port, service, and Windows/Linux server monitoring using Nagios, but encountered a few challenges.

- **Configuration Issues:** Setting up monitoring hosts and editing files like linuxserver.cfg led to some errors in file paths and syntax, which required careful review.
- **NRPE Setup:** Configuring NRPE for remote monitoring was tricky due to firewall and permission issues, often causing connectivity problems between the Nagios host and clients.
- **Service Restarts:** Restarting Nagios and NRPE to apply changes didn't always work smoothly, with misconfigurations requiring troubleshooting.
- **Dashboard Access:** Accessing the Nagios dashboard was hindered by incorrect AWS security group rules, needing adjustments to allow proper HTTP and TCP traffic.

NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/ 9

**AdvanceDevOps Experiment No: 11**

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

**Theory:**

**AWS Lambda**

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

**Lambda Workflow**

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

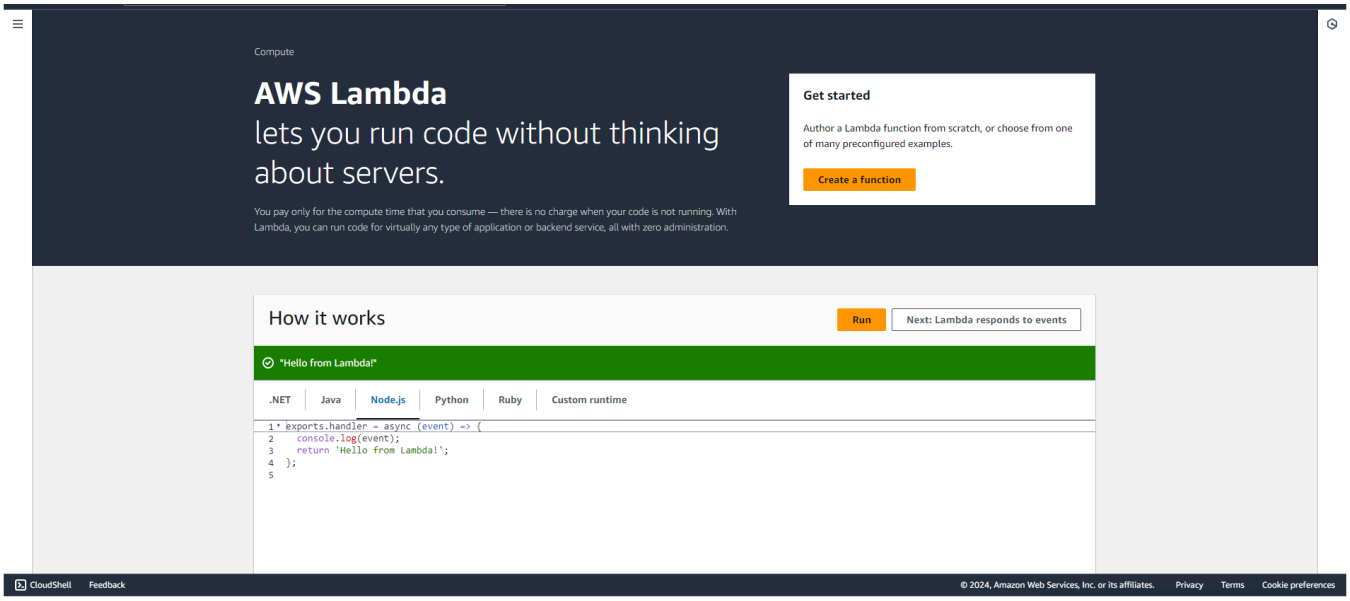
**AWS Lambda Functions**

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

NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/ 9

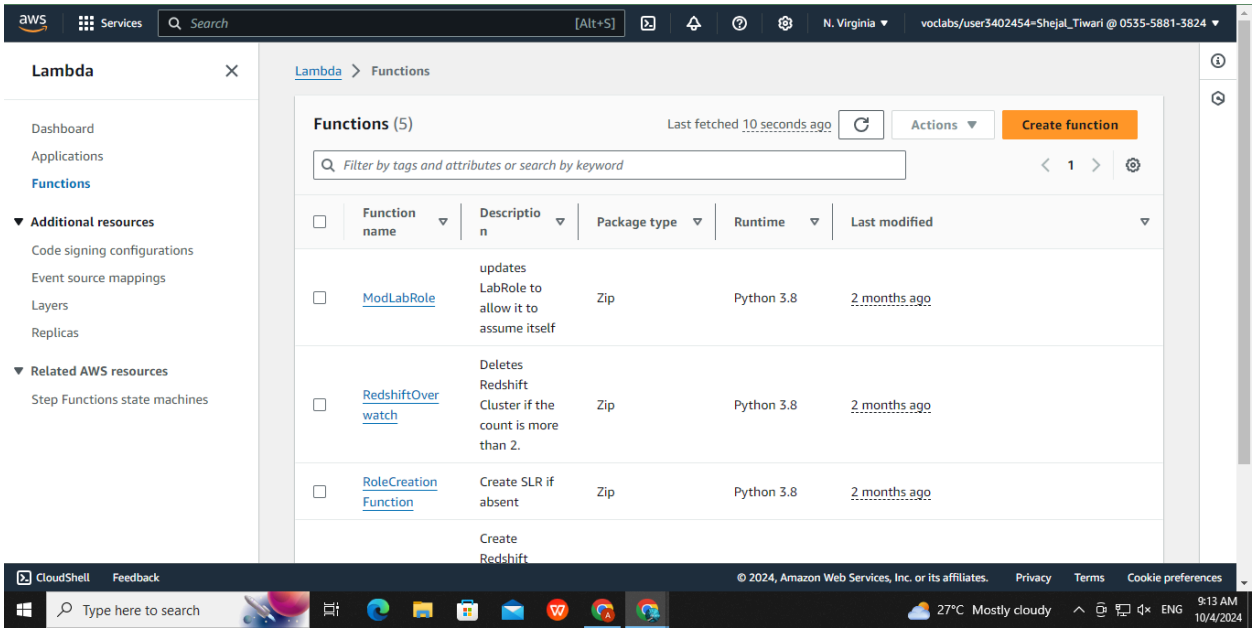
**Steps To create the lambda function:**

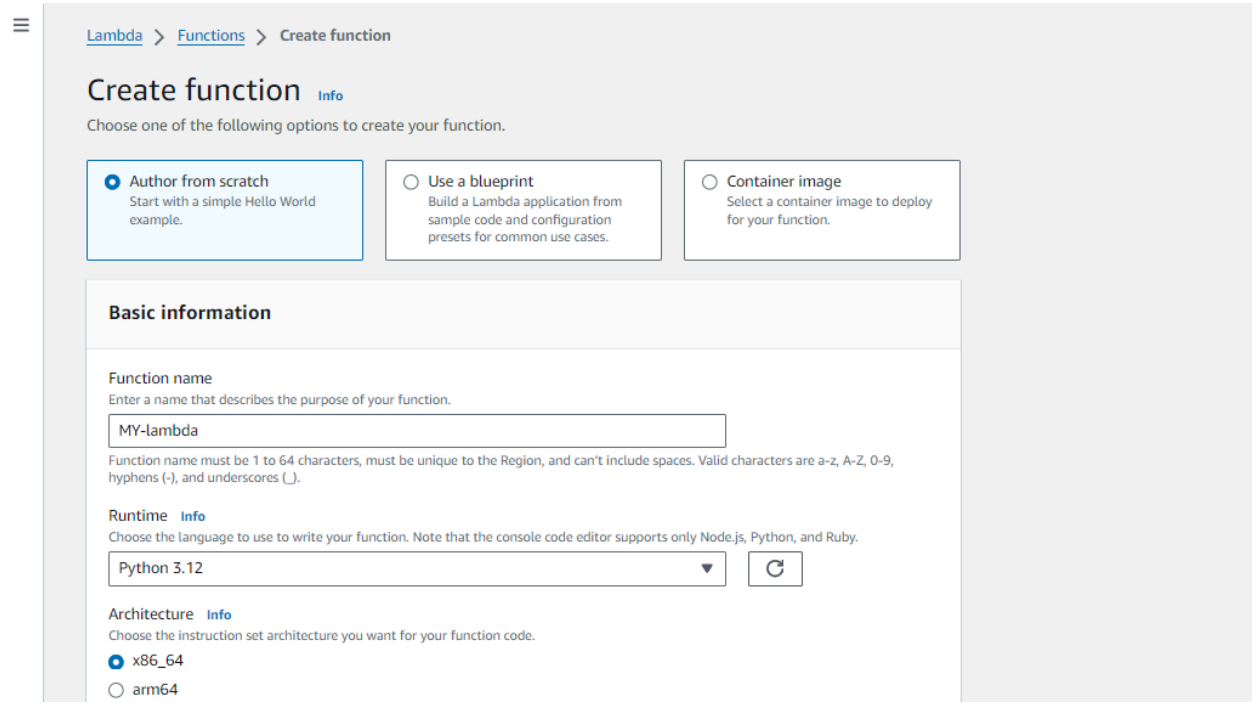
**Step 1:** Login to your AWS Personal/Academy Accout.Open lambda and click on create function button.



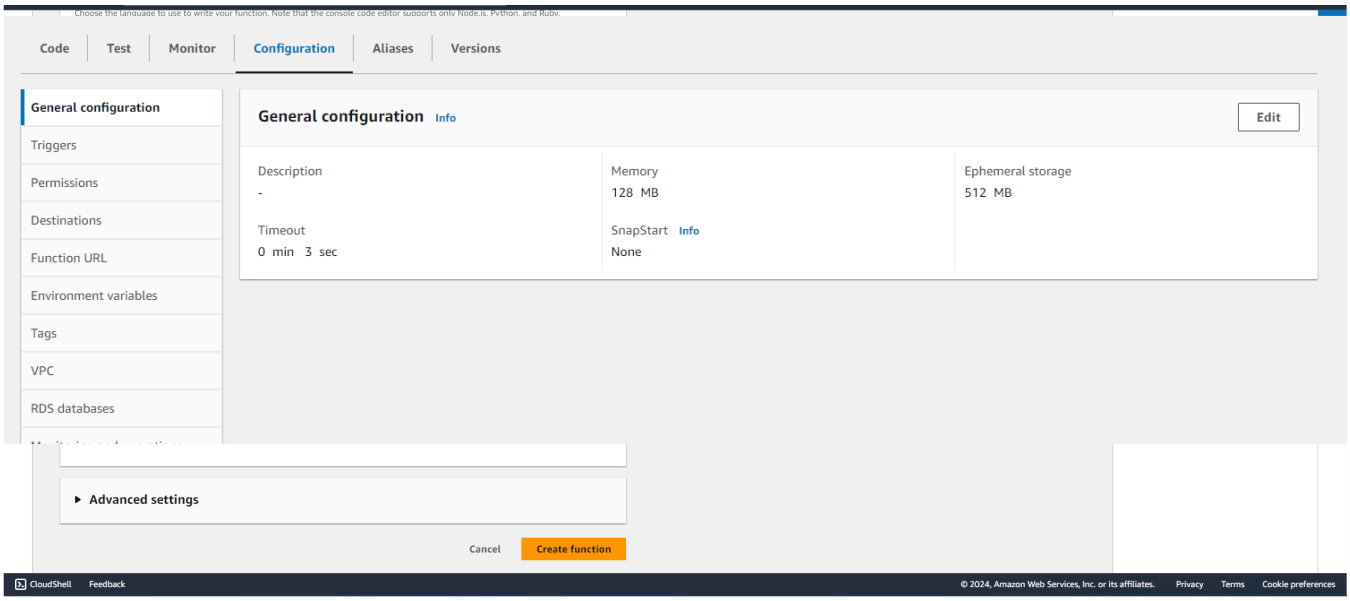

**Step 2:** Now Give a name to your Lambda function, Select the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby. So will select Python 3.12 , Architecture as x86, and Execution role to Create a new role with basic Lambda permissions.

NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/ 9

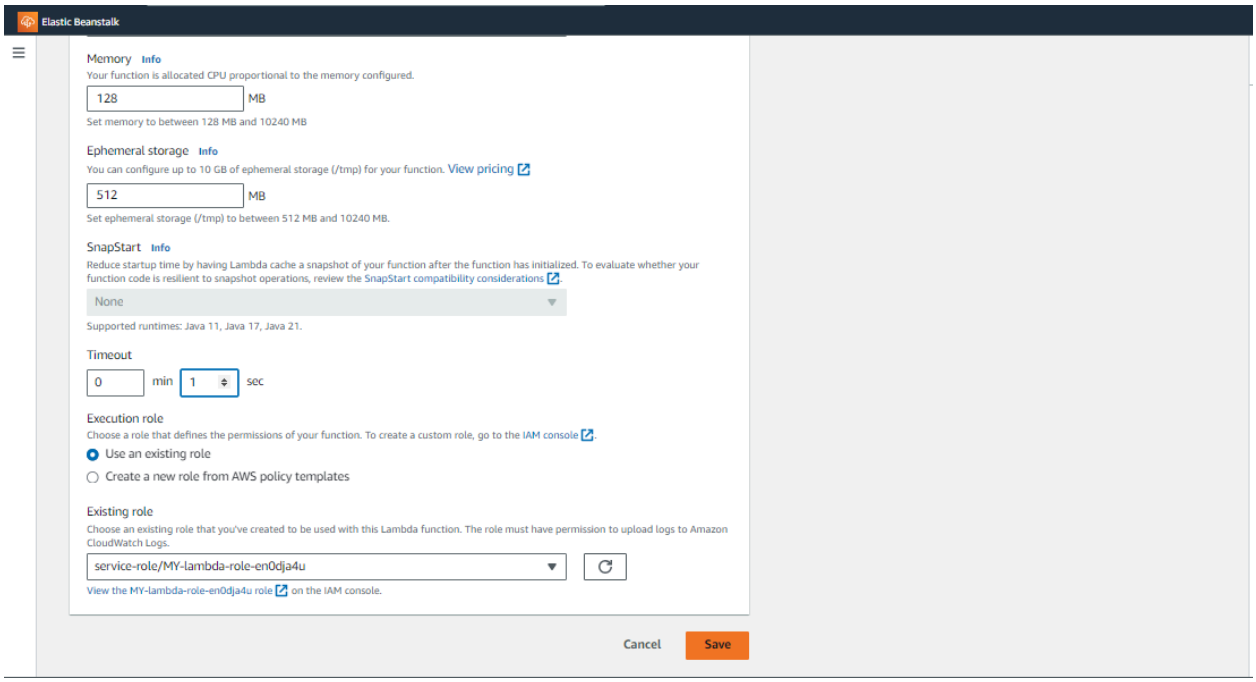



So See or Edit the basic settings go to configuration then click on edit general setting.



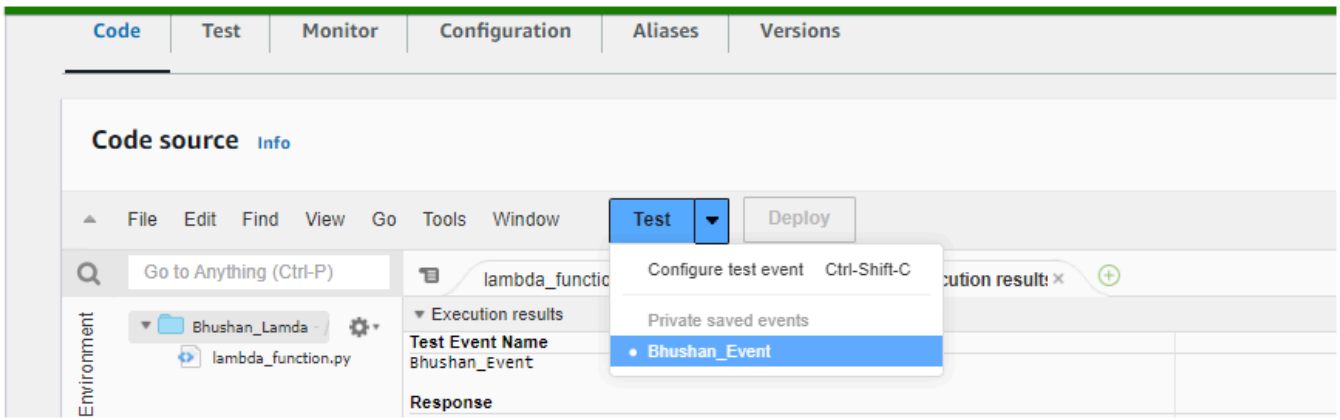

Here, you can enter a description and change Memory and Timeout. I've changed the Timeout period to 1 sec since that is sufficient for now.

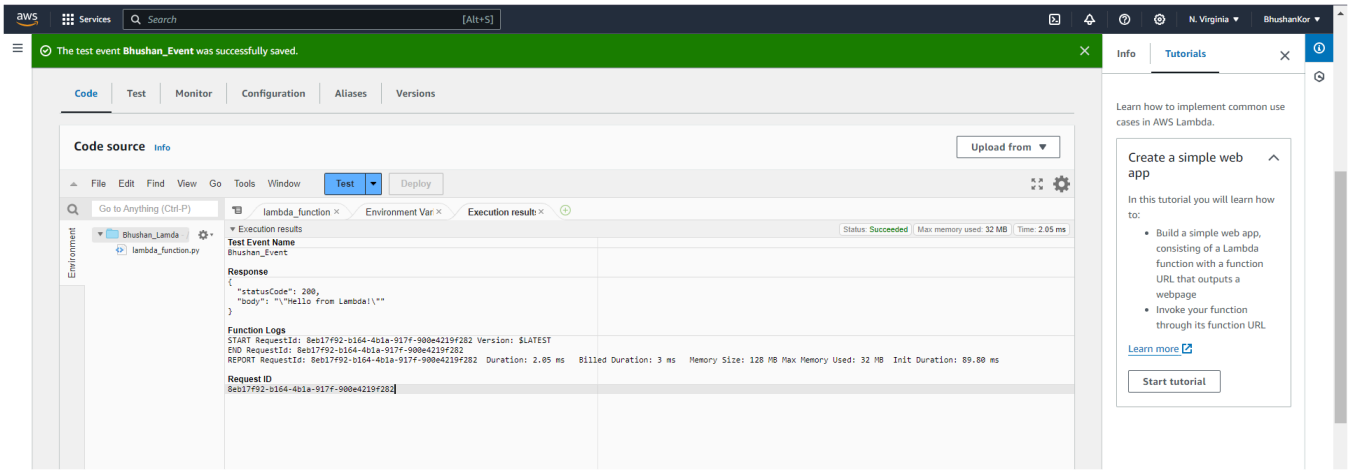


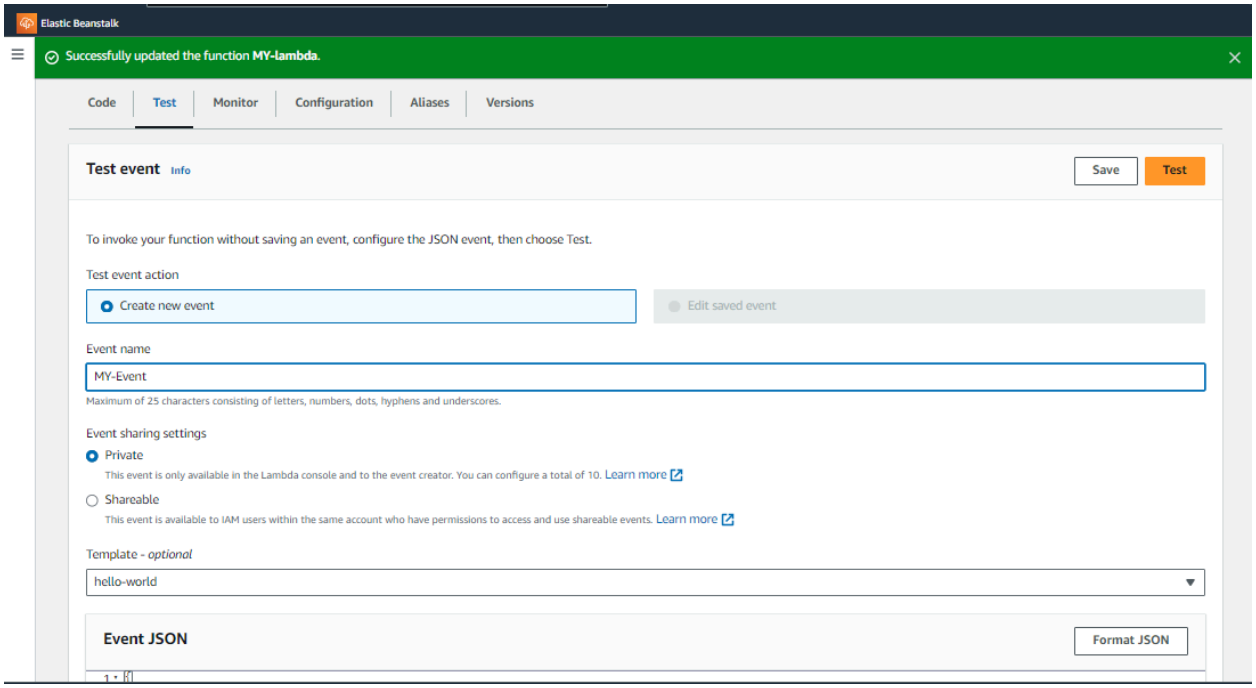

**Step 3:** Now Click on the Test tab then select Create a new event, give a name to the event and select Event Sharing to private, and select hello-world template.

NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/ 9

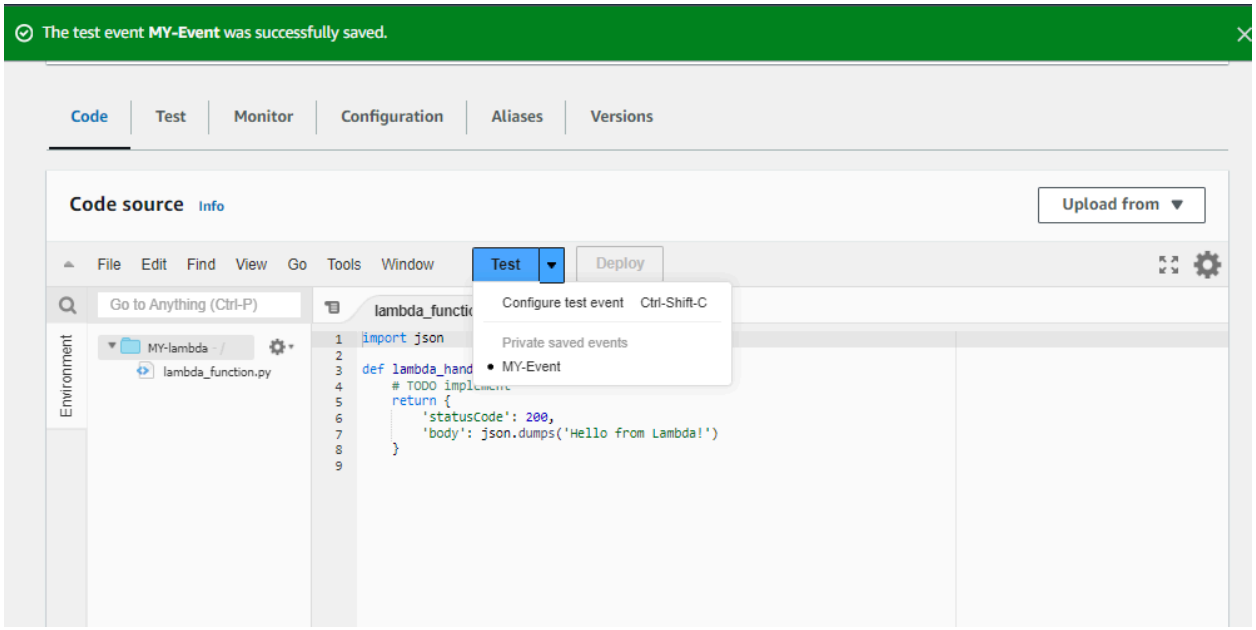
**Step 4:** Now In Code section select the created event from the dropdown of test then click on test . You will see the below output.



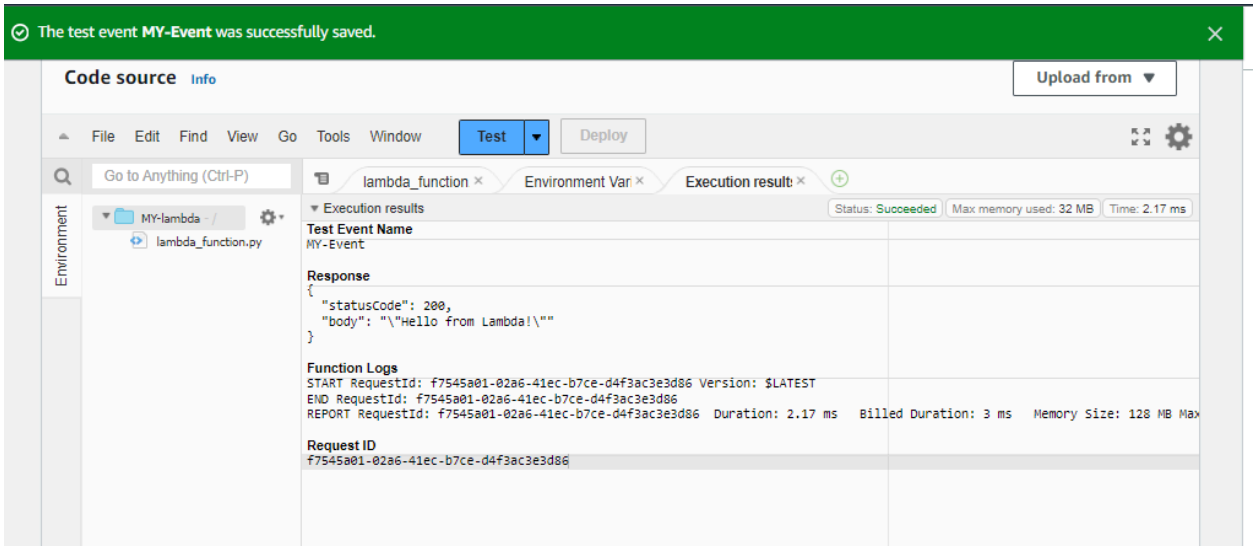



**Step 5:** You can edit your lambda function code. I have changed the code to display the new String. Now ctrl+s to save and click on deploy to deploy the changes.




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

NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/ 9




**Conclusion:**

In this experiment, we successfully created and deployed our first AWS Lambda function using Python, gaining an understanding of its workflow and capabilities. The function was executed and tested, allowing us to observe the output and logs. While the overall process was smooth, there were several challenges that we encountered:

- **Role and Permissions Configuration:** Setting up the correct execution role with the necessary permissions was critical, as misconfigurations could prevent the function from running or accessing other AWS services. Debugging permission issues was time-consuming, especially when using new roles.

- **Timeout Issues:** Initially, the default timeout setting was insufficient for some operations. Adjusting the timeout to a value that suits the function's workload was necessary, especially for more complex operations beyond basic tasks.

- **Event Testing:** Configuring and testing events within Lambda required careful attention. Choosing the wrong template or incorrect event parameters often led to failures during the test phase, requiring adjustments to the event settings.

NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/ 9

**AdvanceDevOps Experiment No: 12**

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

**Theory:**

**AWS Lambda and S3 Integration:**

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

**Workflow:**

**1. Create an S3 Bucket:**

○ First, create an S3 bucket that will store the objects. This bucket will act as the trigger source for the Lambda function.

**2. Create the Lambda Function:**

○ Set up a new Lambda function using AWS Lambda’s console. You can choose a runtime environment like Python, Node.js, or Java.

○ Write code that logs a message like “An Image has been added” when triggered.

**3. Set Up Permissions:**

○ Ensure that the Lambda function has the necessary permissions to access S3. You can do this by attaching an IAM role with policies that allow reading from the bucket and writing logs to CloudWatch.

**4. Configure S3 Trigger:**

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

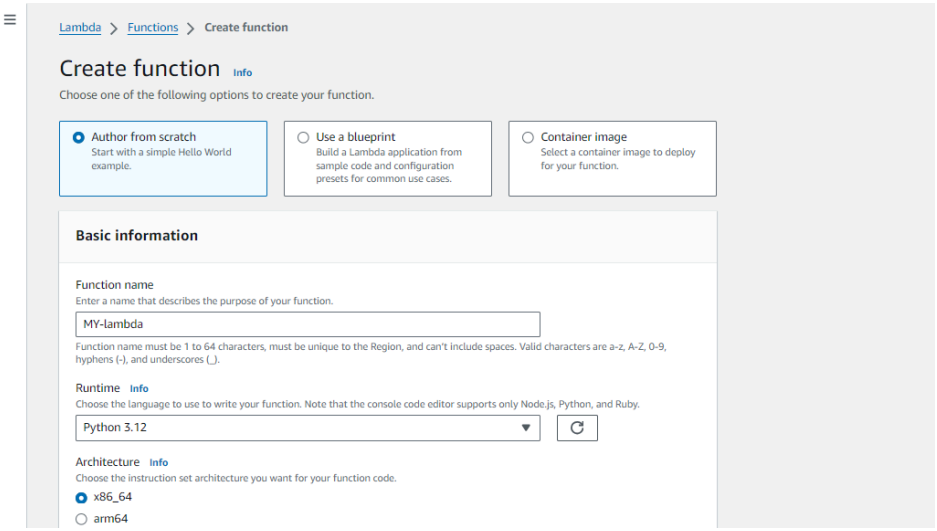
**5. Test the Setup:**

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

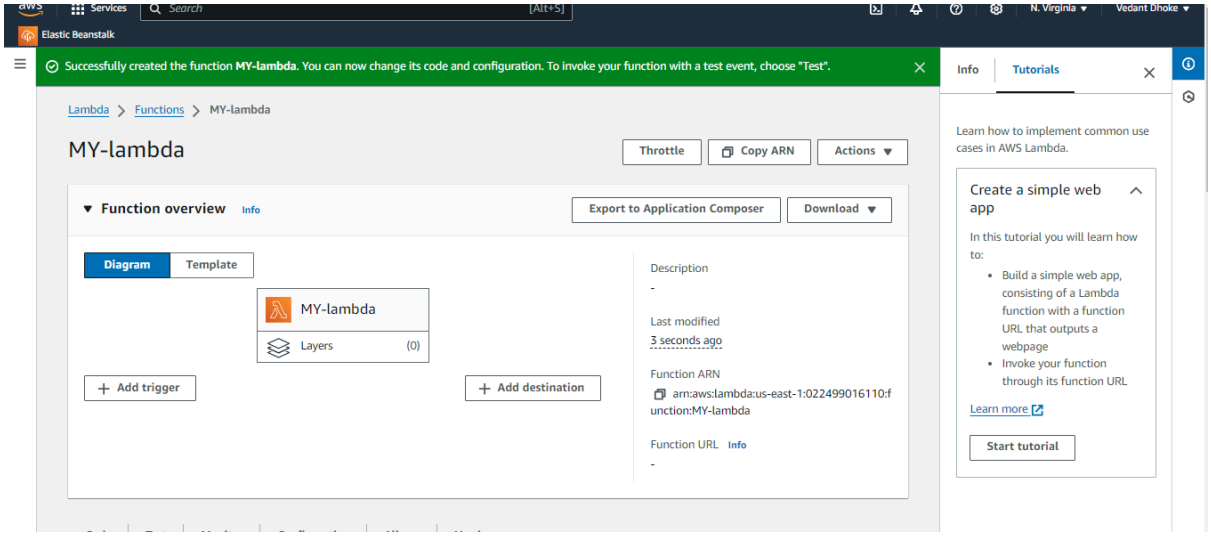
NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/ 9

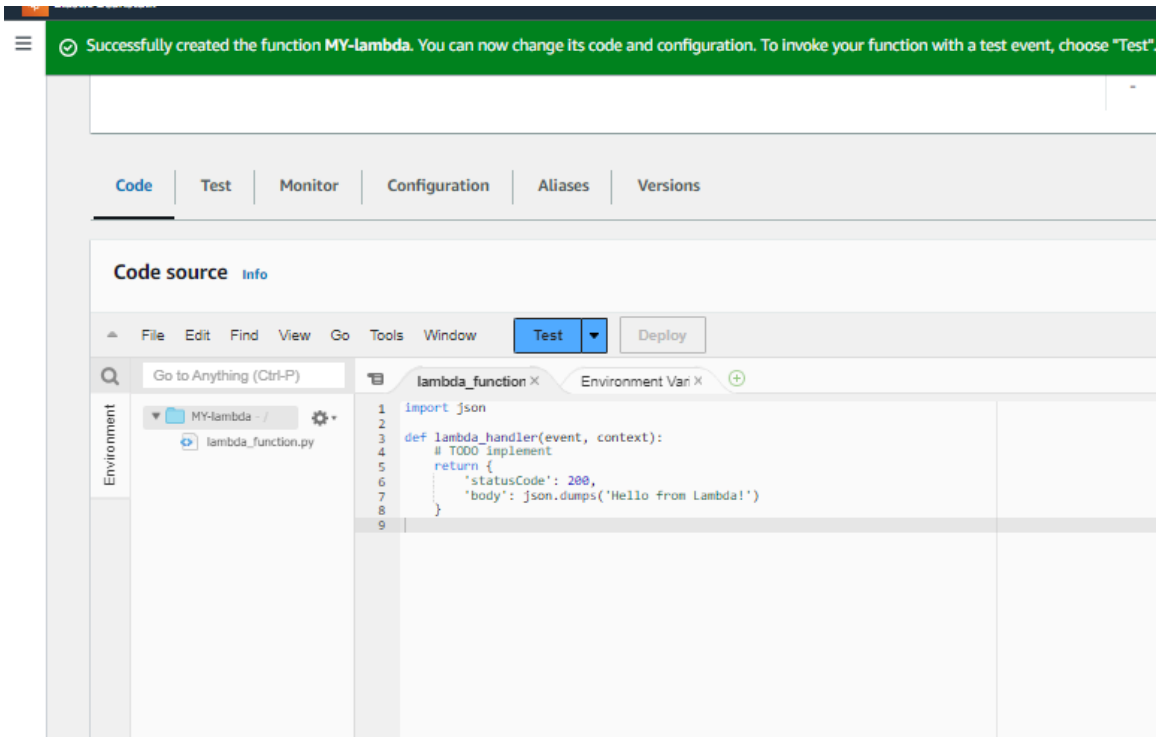
**Steps To create the lambda function:**

**Step 1:** Login to your AWS Personal account. Now open S3 from services and click on create S3 bucket.
**Step 2:** I have used already created bucket v2bucket if you dont have then you can create a basic bucket for this experiment .
**Step 3:** Open lambda console and click on create function button.
**Step 4:** Now Give a name to your Lambda function, Select the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby. So will select Python 3.12 , Architecture as x86, and Exceution role to Create a new role with basic Lambda permissions.



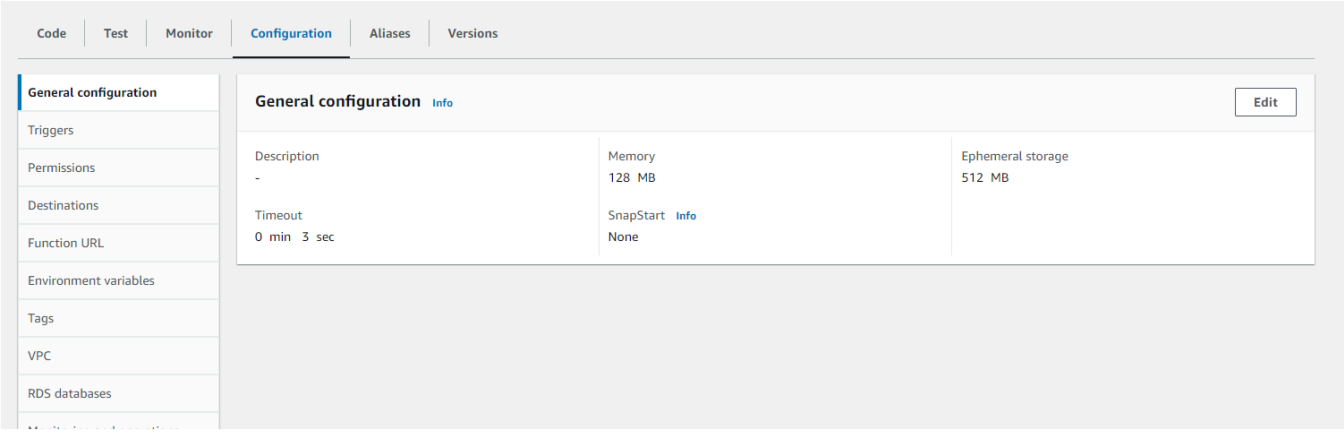
NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/ 9



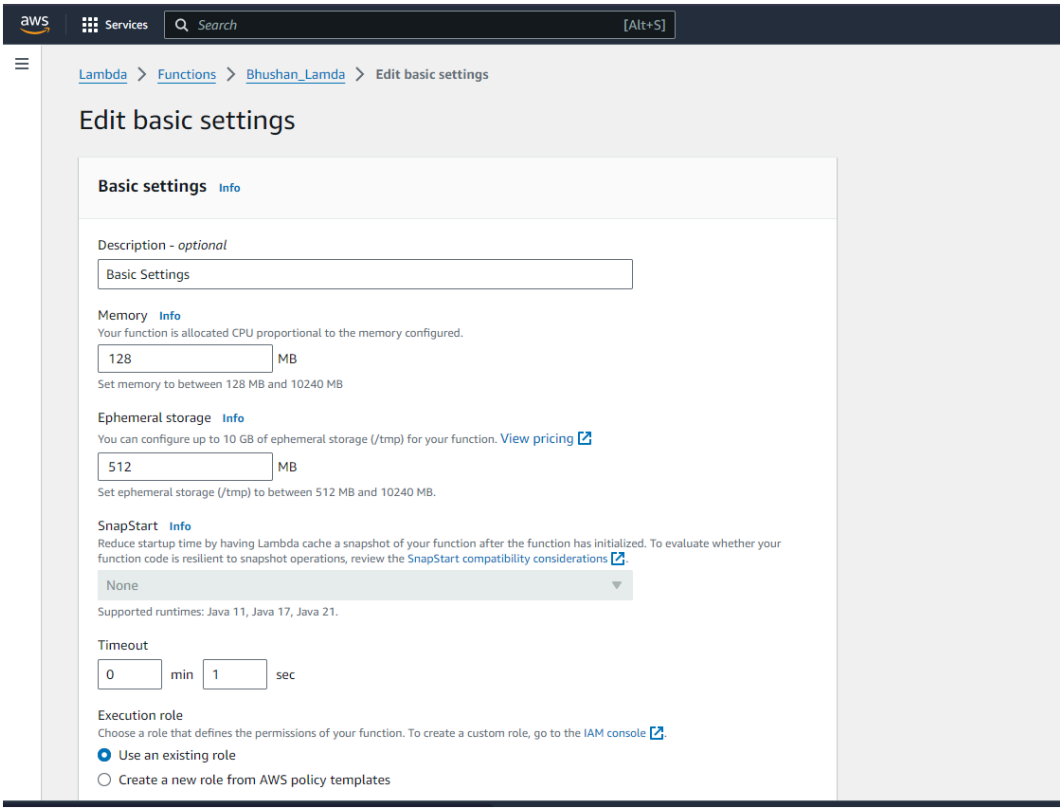




So See or Edit the basic settings go to configuration then click on edit general setting.

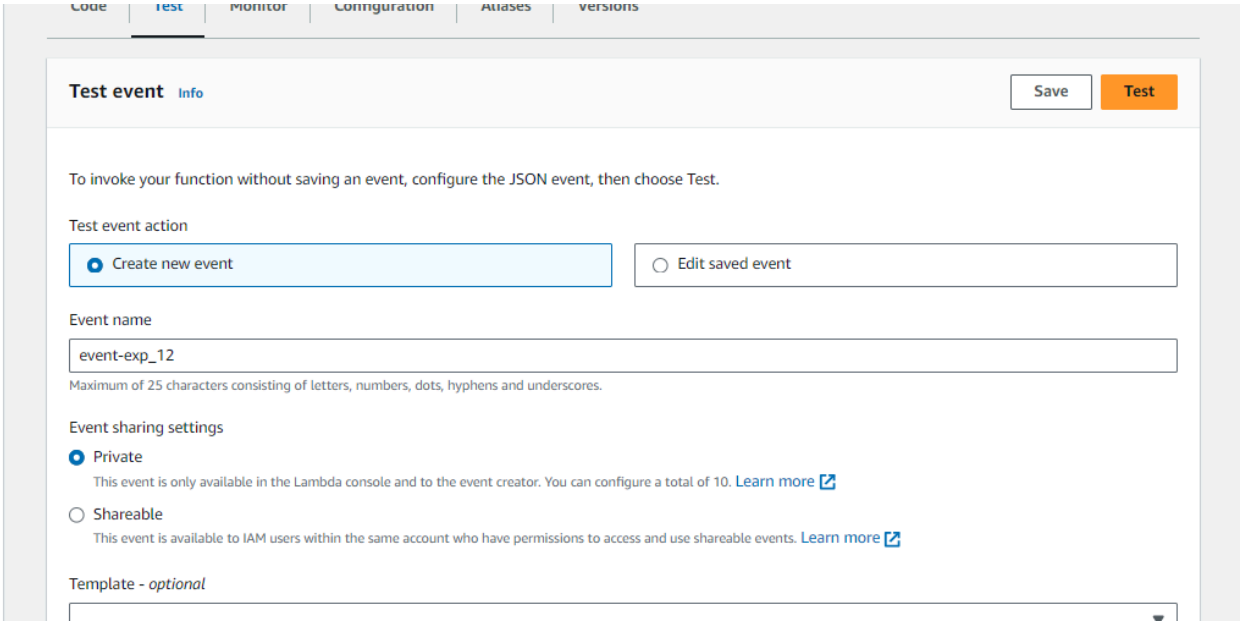
NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/ 9

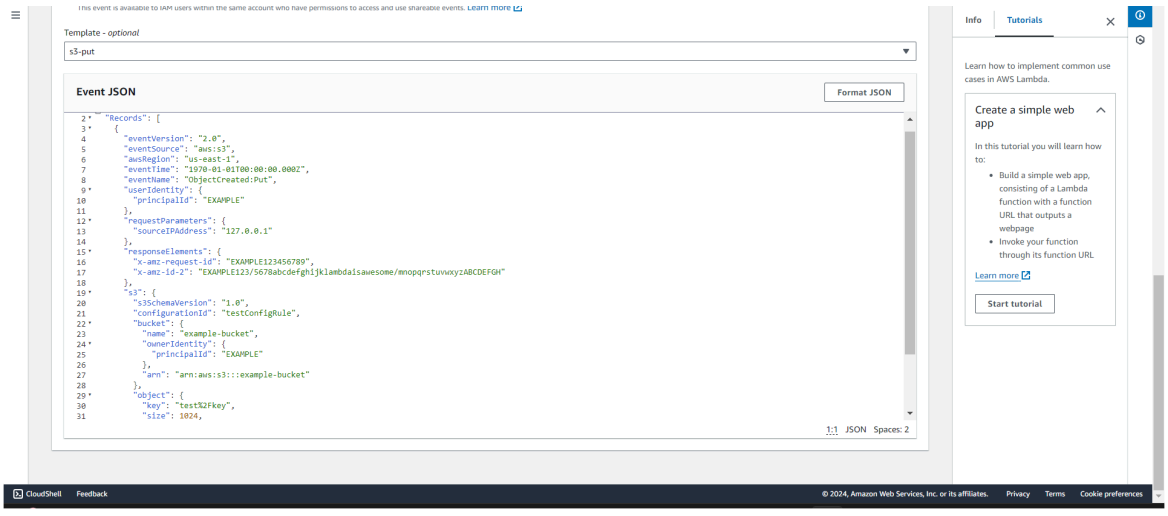



Here, you can enter a description and change Memory and Timeout. I've changed the Timeout period to 1 sec since that is sufficient for now.

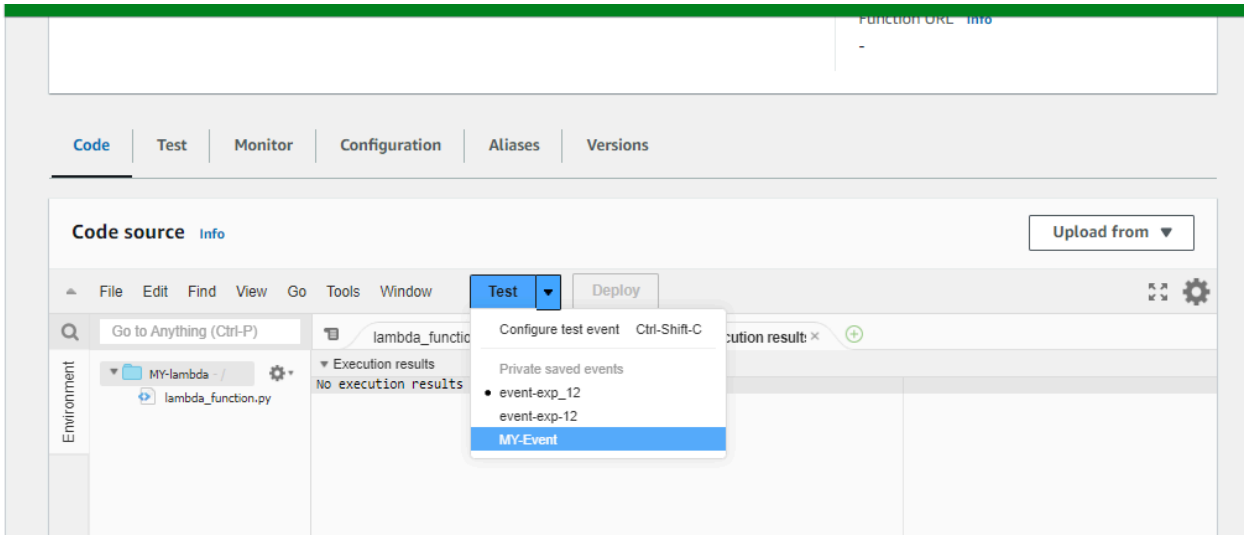


**Step 5:** Now Click on the Test tab then select Create a new event, give a name to the event and select Event Sharing to private, and select s3 put template.



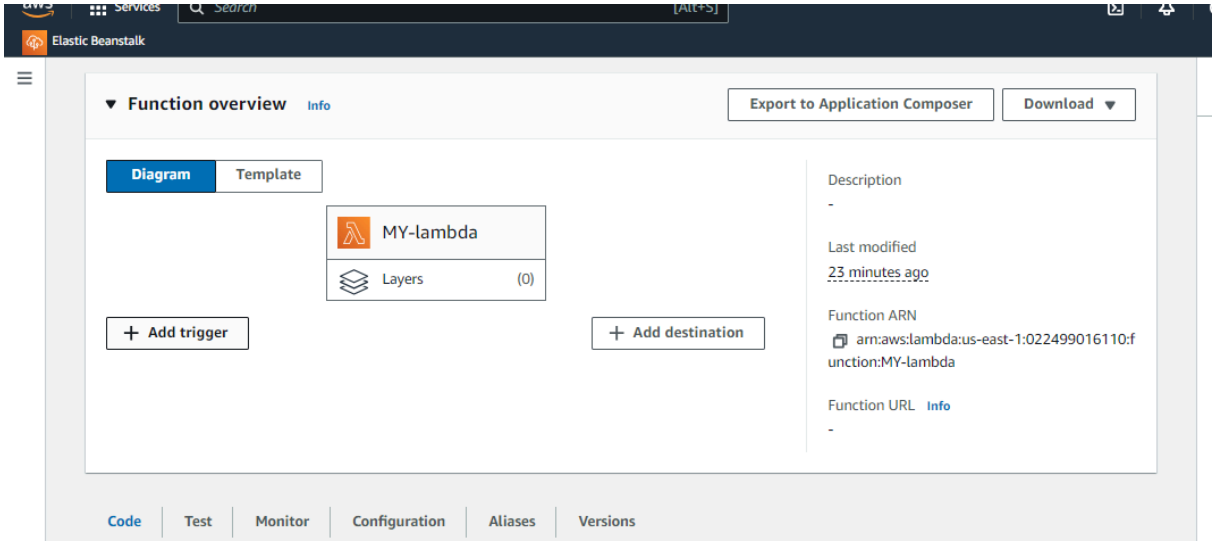



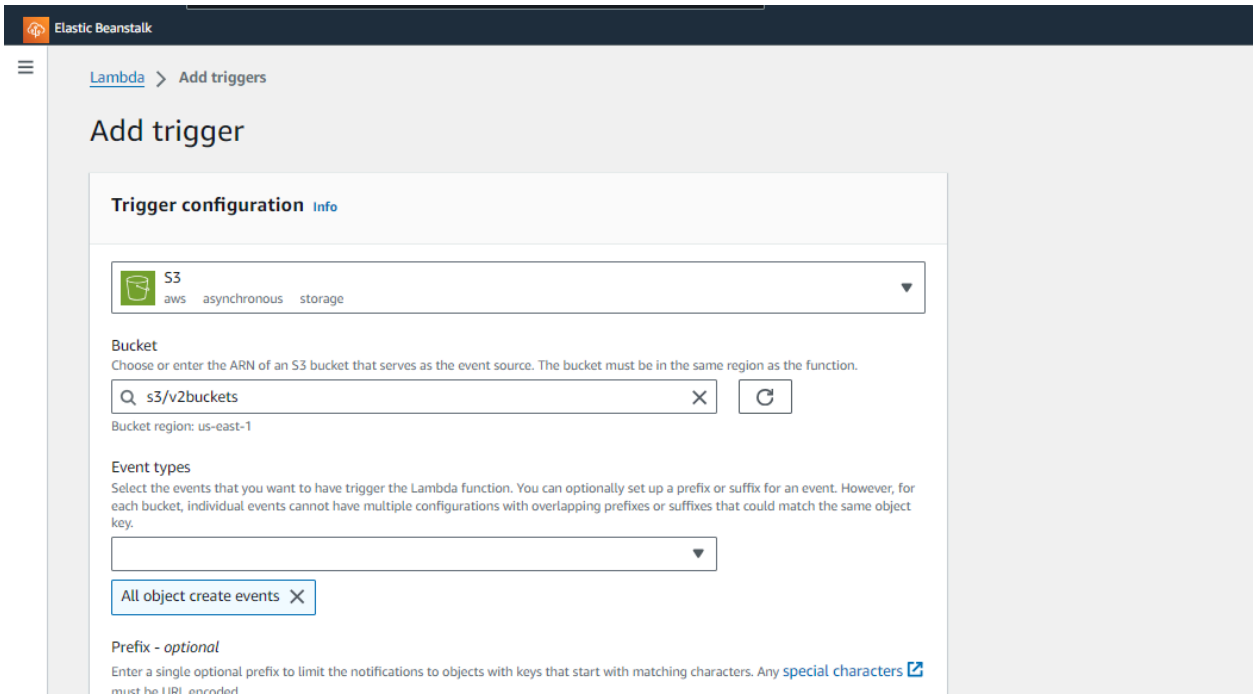
**Step 6:** Now In Code section select the created event from the dropdown .

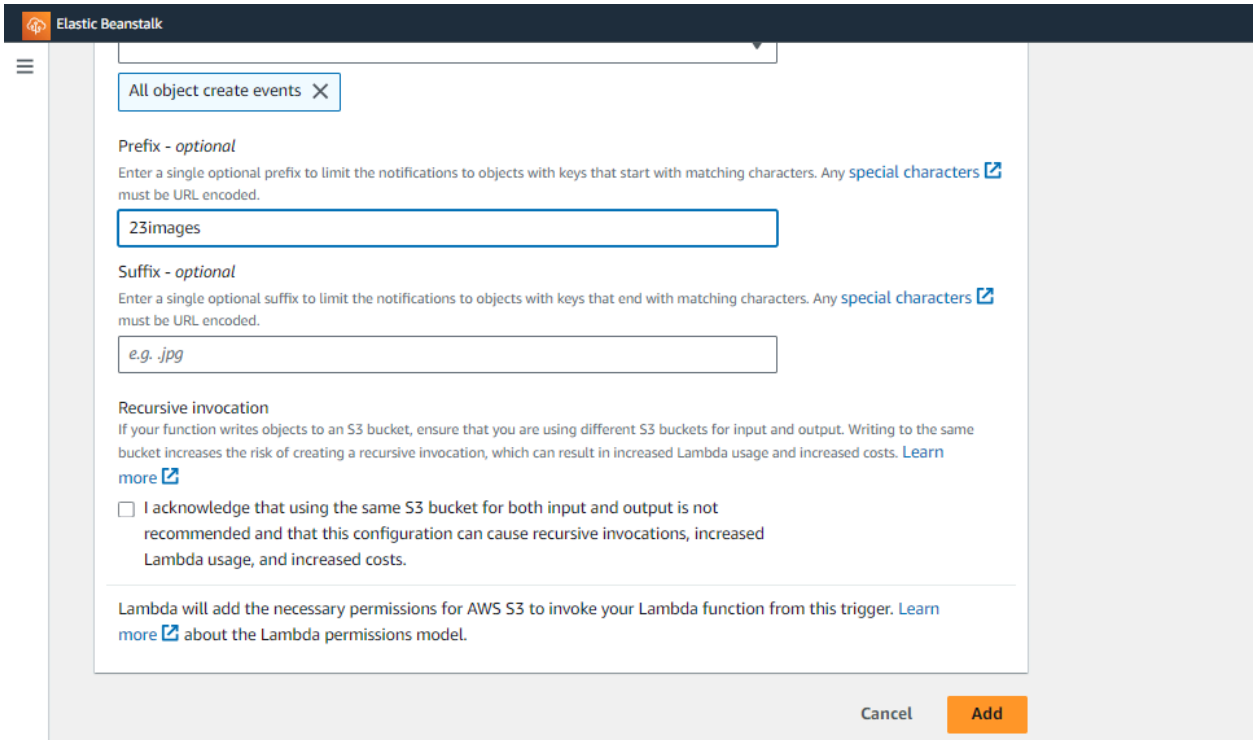


NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/ 9

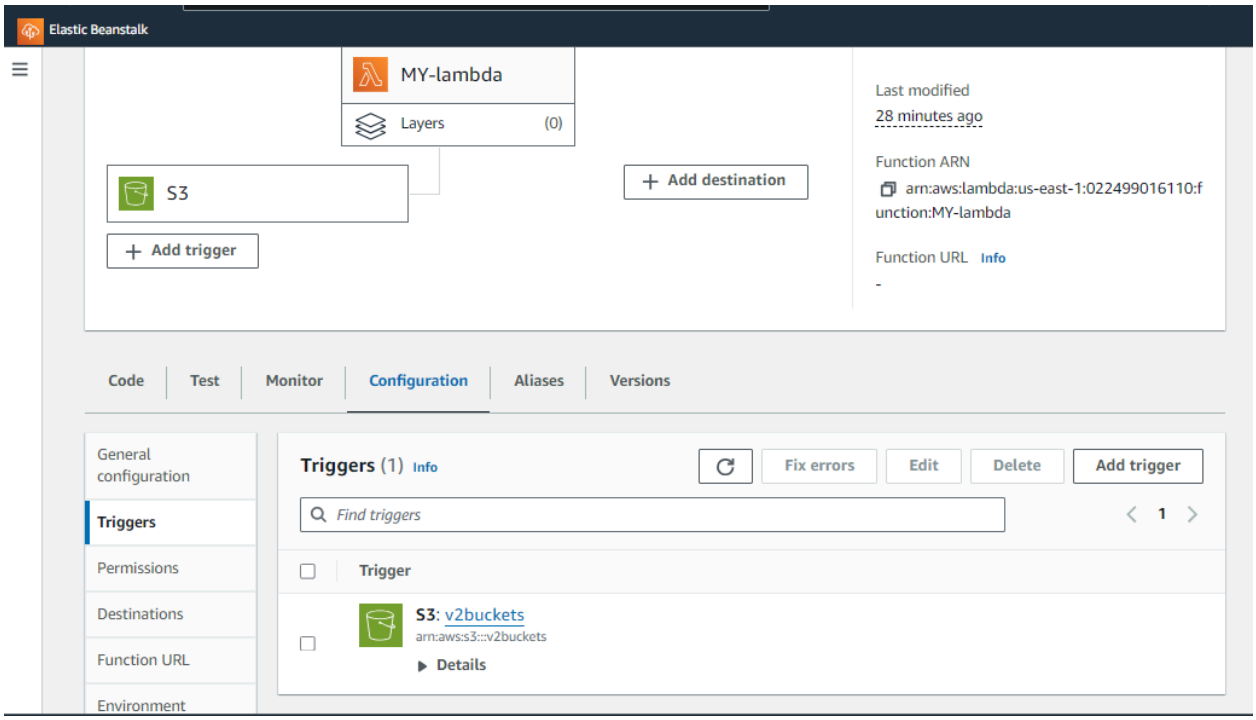
**Step 7:** Now In the Lambda function click on add tigger.
Now select the source as S3 then select the bucket name from the dropdown, keep other things to default and also you can add prefix to



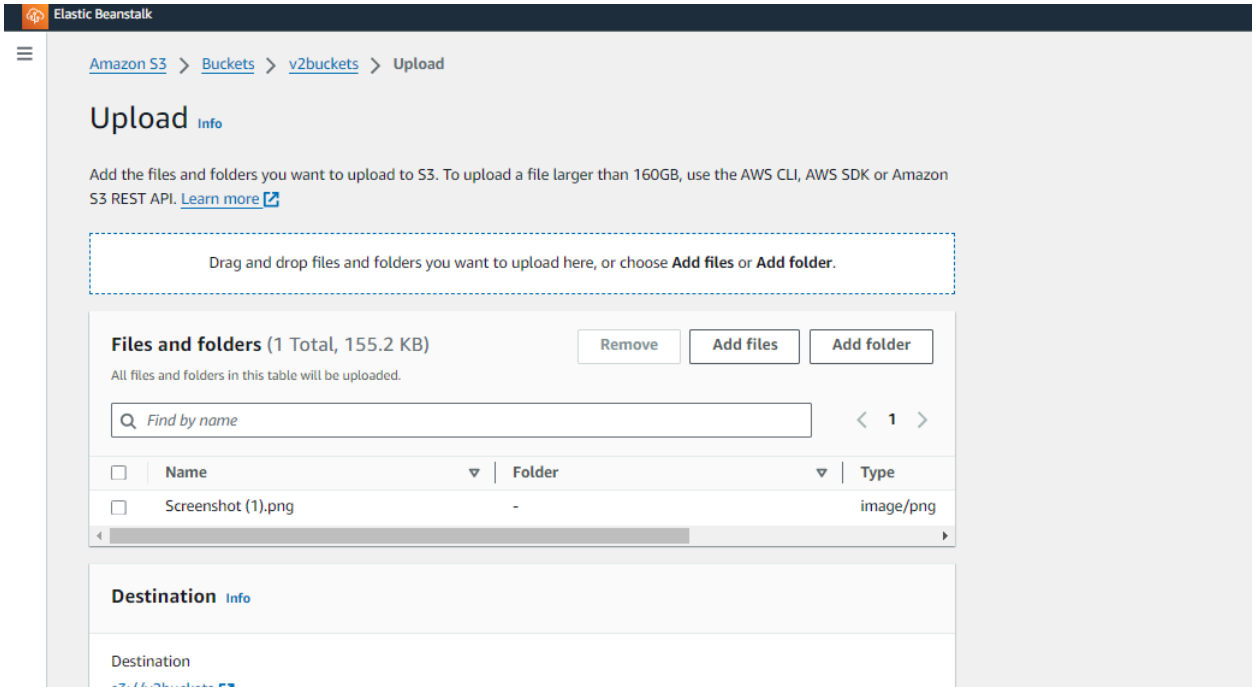





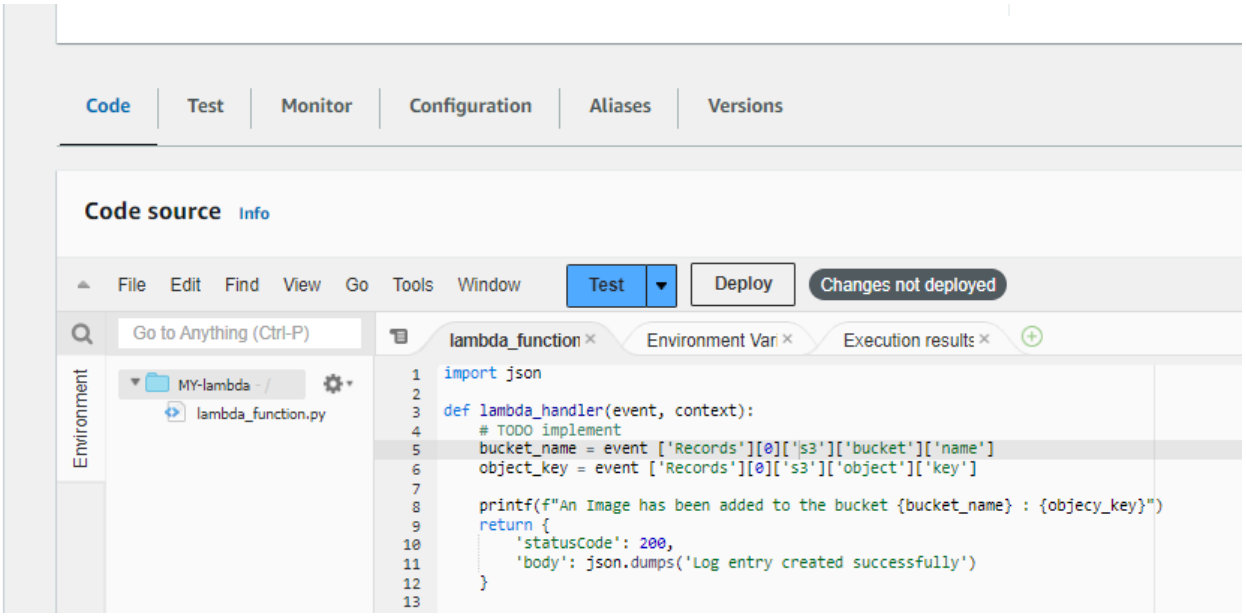

**Step 8:** Now Write code that logs a message like "An Image has been added" when triggered. Save the file and click on deploy.

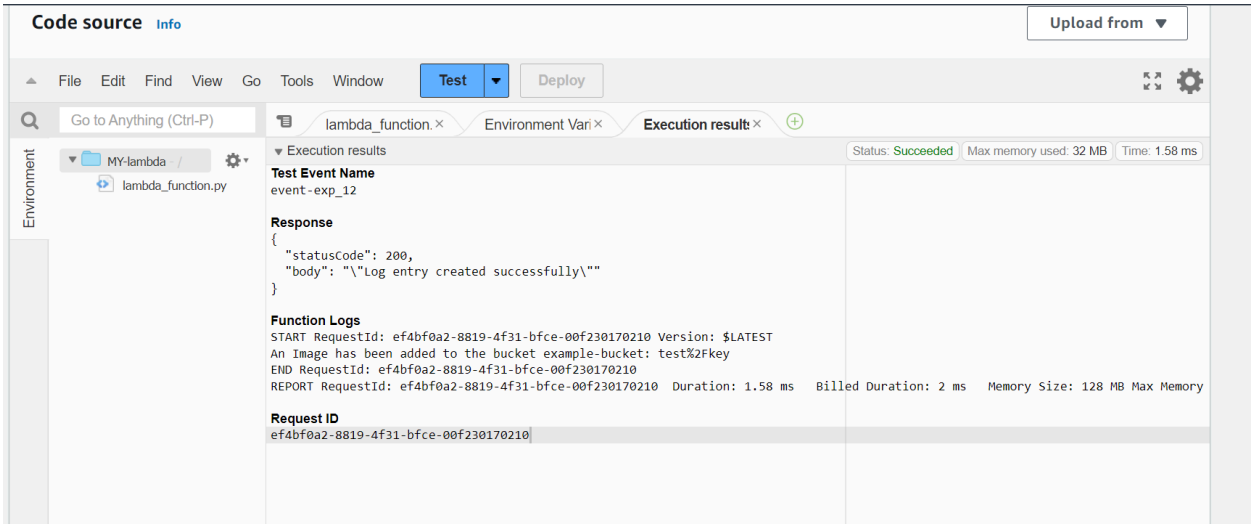


**Step 9:** Now upload any image to the bucket.

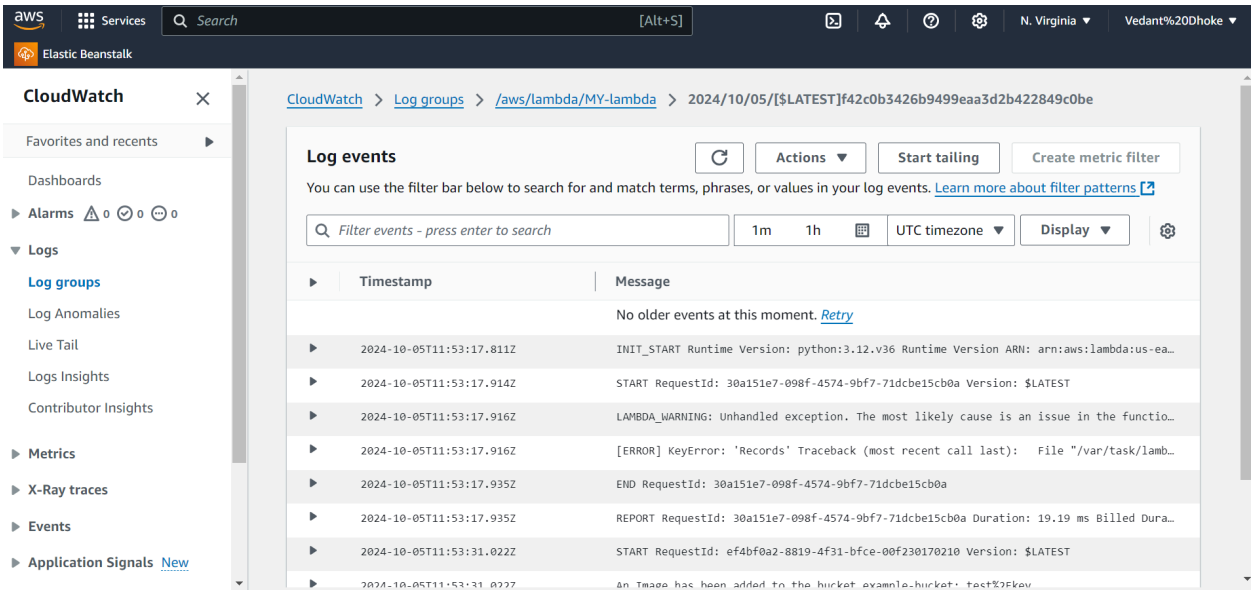



**Step 10:** Now to click on test in lambda to check whether it is giving log when image is added to S3.






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




## Conclusion:

In this experiment, we successfully created a Lambda function designed to log the message “An Image has been added” upon the addition of an object to a specified S3 bucket. This process deepened our understanding of integrating AWS Lambda with S3 for event-driven applications. However, we encountered several challenges that provided valuable learning experiences:

- **S3 Bucket Configuration:** Setting up the S3 bucket correctly was crucial. We had to ensure that the bucket's public access settings were appropriately configured to allow the Lambda function to trigger without compromising security. Misconfigurations here could have led to access issues.

- **Trigger Setup:** Configuring the S3 bucket as a trigger for the Lambda function required careful attention. Selecting the correct bucket and event type was essential to ensure that the function executed as intended. Any oversight could result in the function not being triggered.

- **Testing and Monitoring:** After uploading images to the S3 bucket, verifying that the Lambda function logged the appropriate messages required us to navigate CloudWatch logs. Initially, we struggled to find the correct log group, which delayed our testing process.