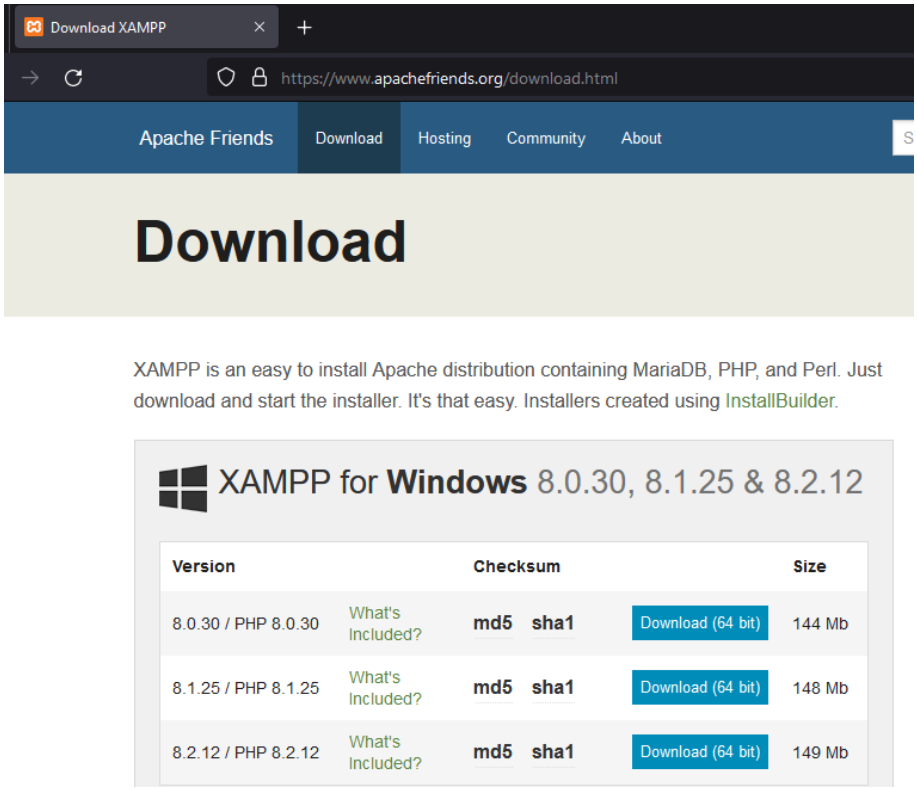
Experiment No. 1(A)

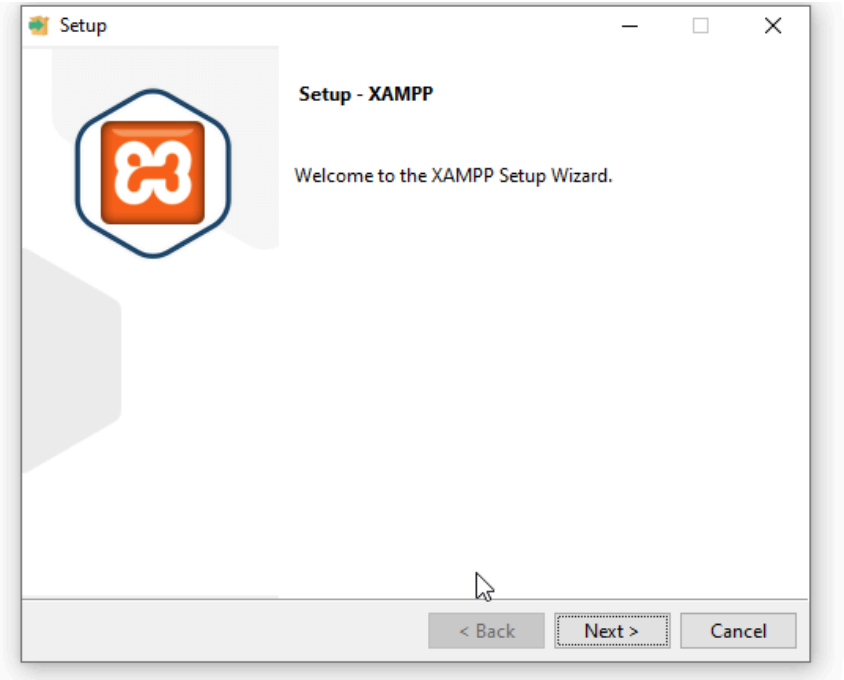
Aim: To develop a website and host it on local machine or virtual machine and Hosting a static website using Amazon S3 Bucket

1. To develop a website and host it on your local machine on a VM using XAMPP

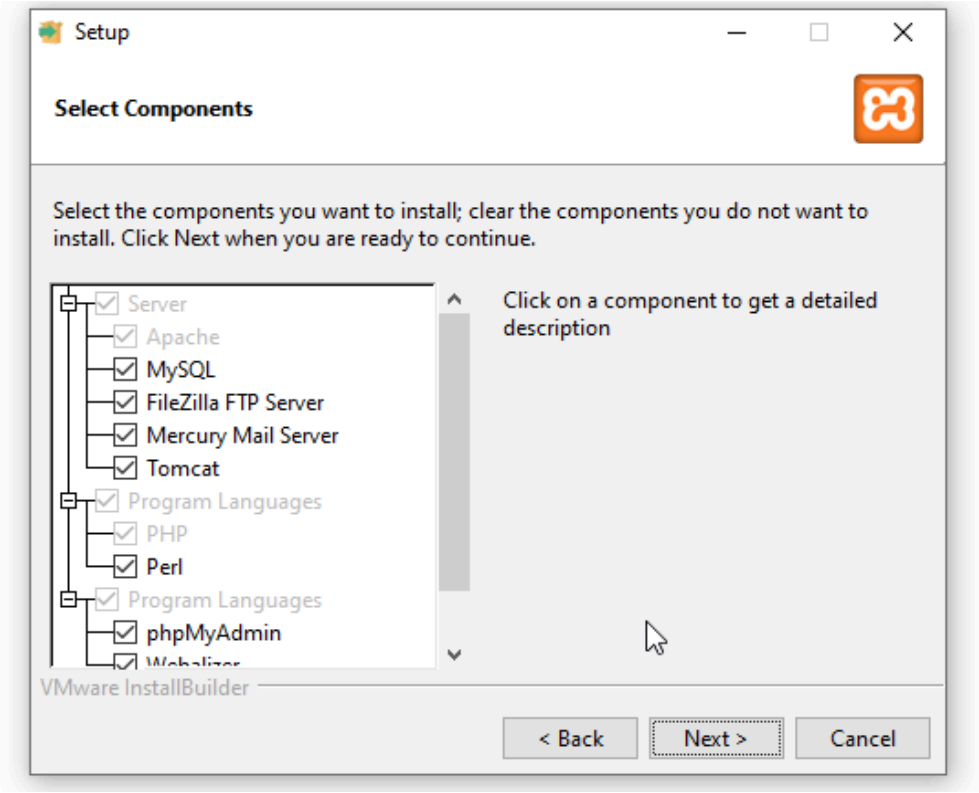
1) Goto official website of XAMPP and download the software as per your OS



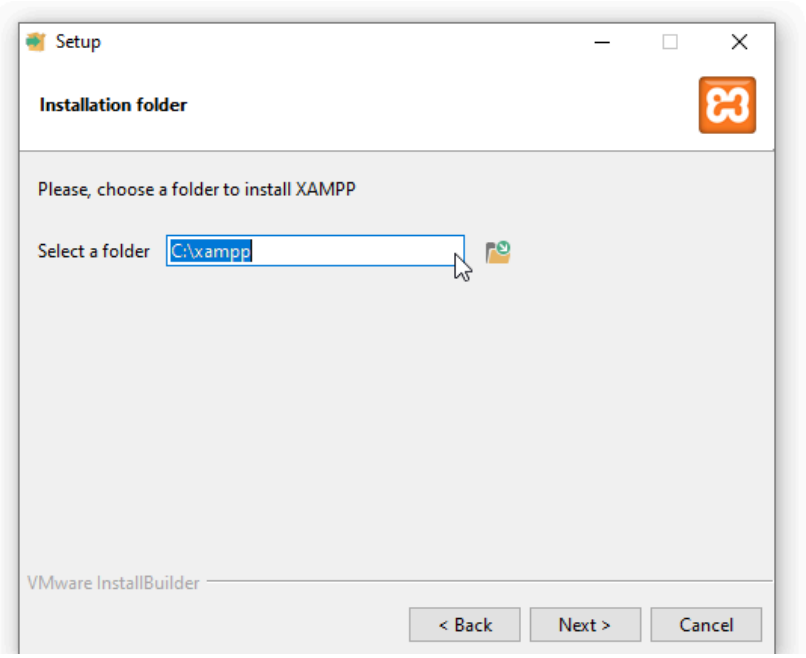
2) After Downloading, open it for installation, Click on next



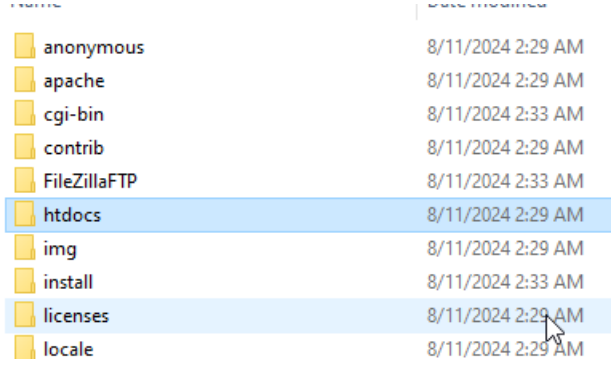
3) Select components to install



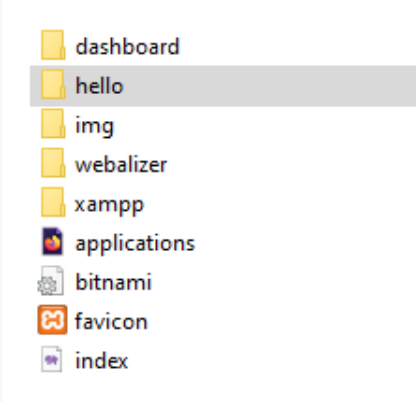
4) Select installation location, and confirm it



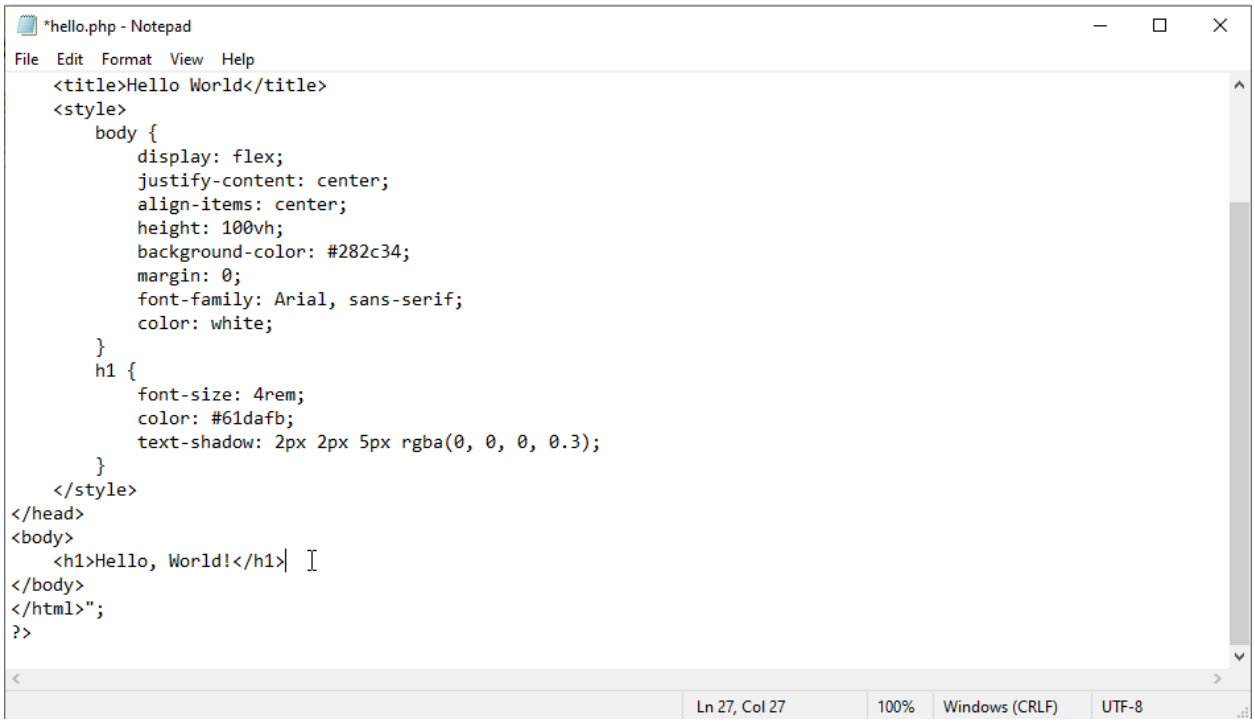
5) After installation, goto folder where XAMPP is installed, go to htdocs



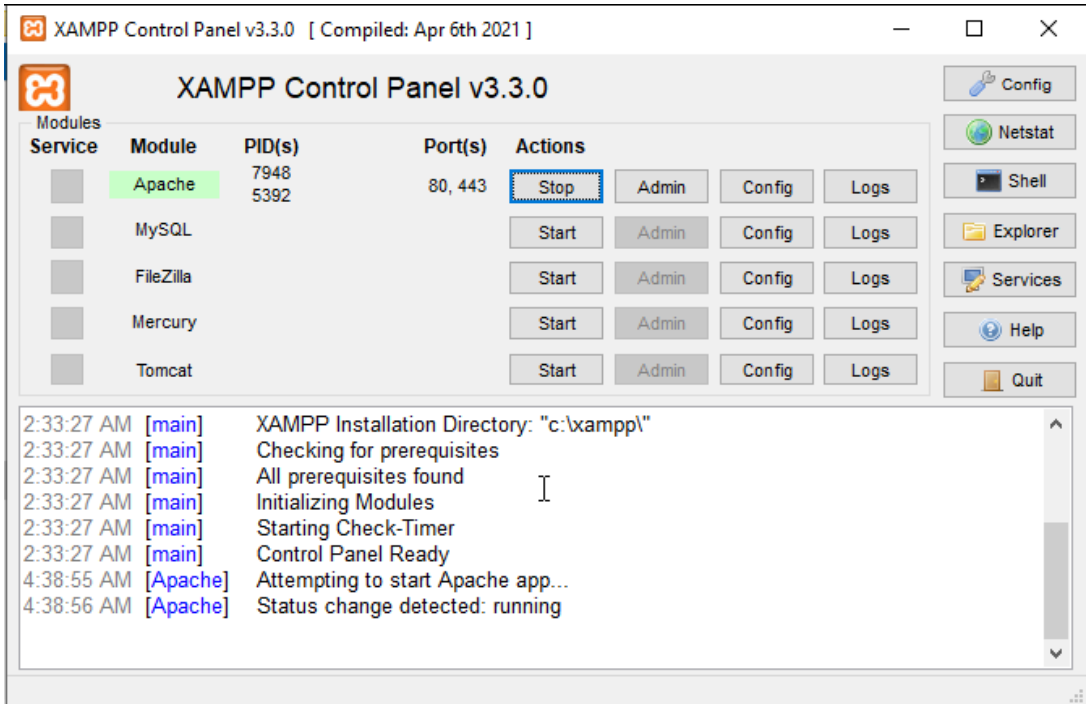
6) Create new folder named anything we want our URI to be.



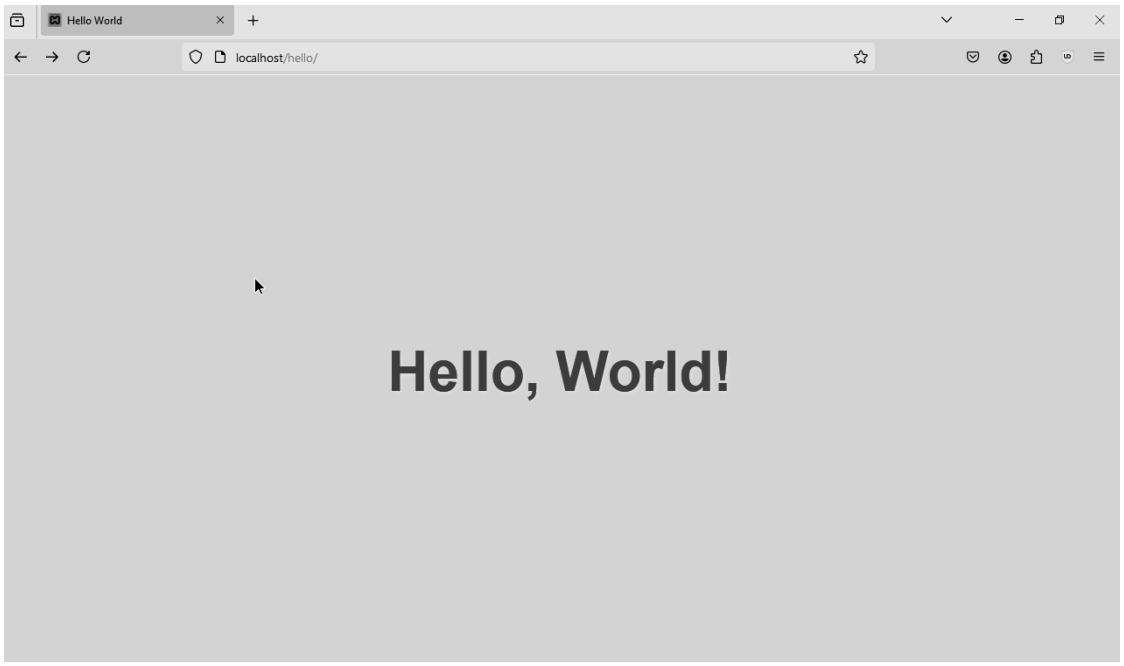
7) Create a file index.php and paste the following code.



8) Now go to XAMPP control panel and start Apache server

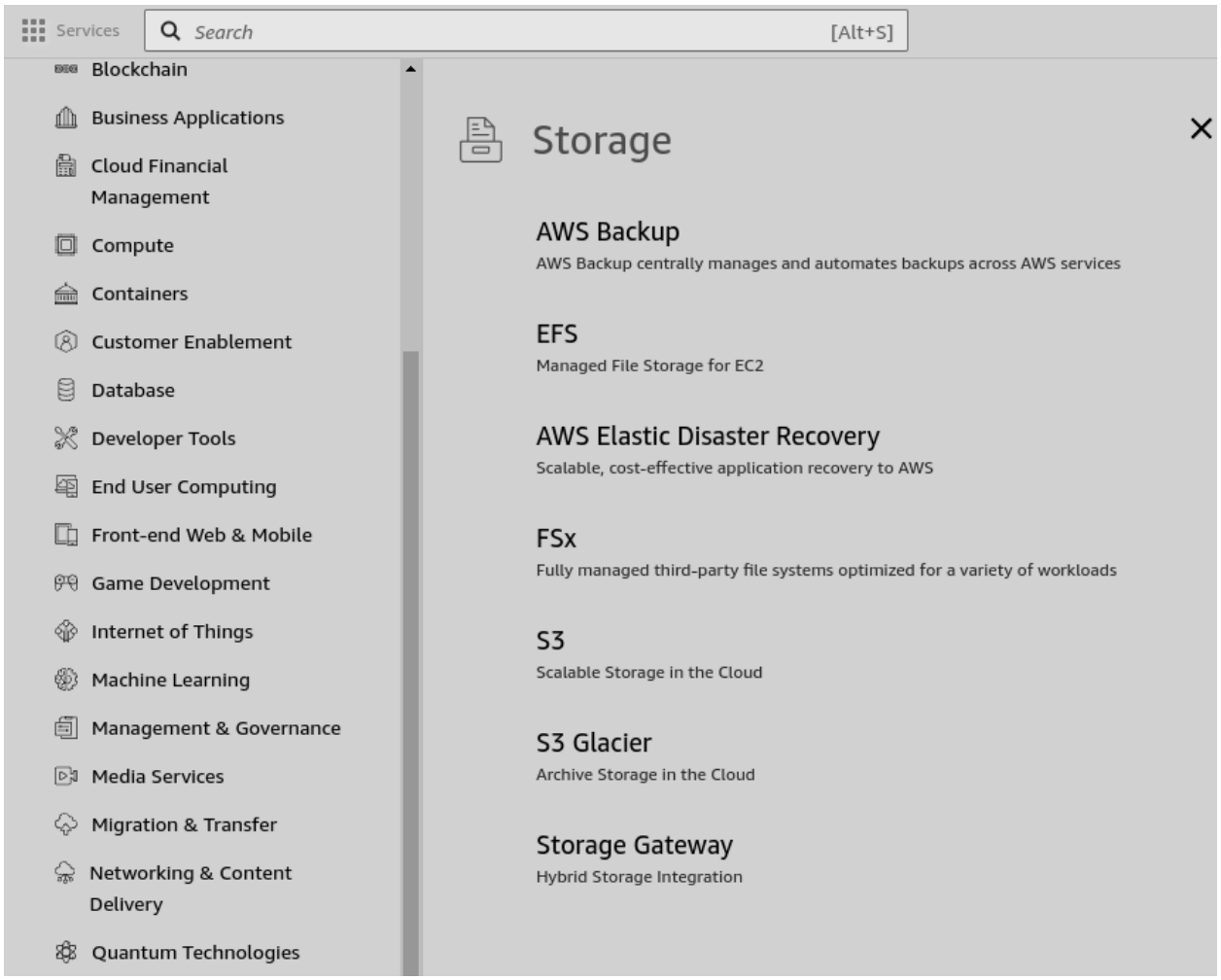


9) Now open browser and enter http://localhost/folder_name. This will open webpage locally on our machine



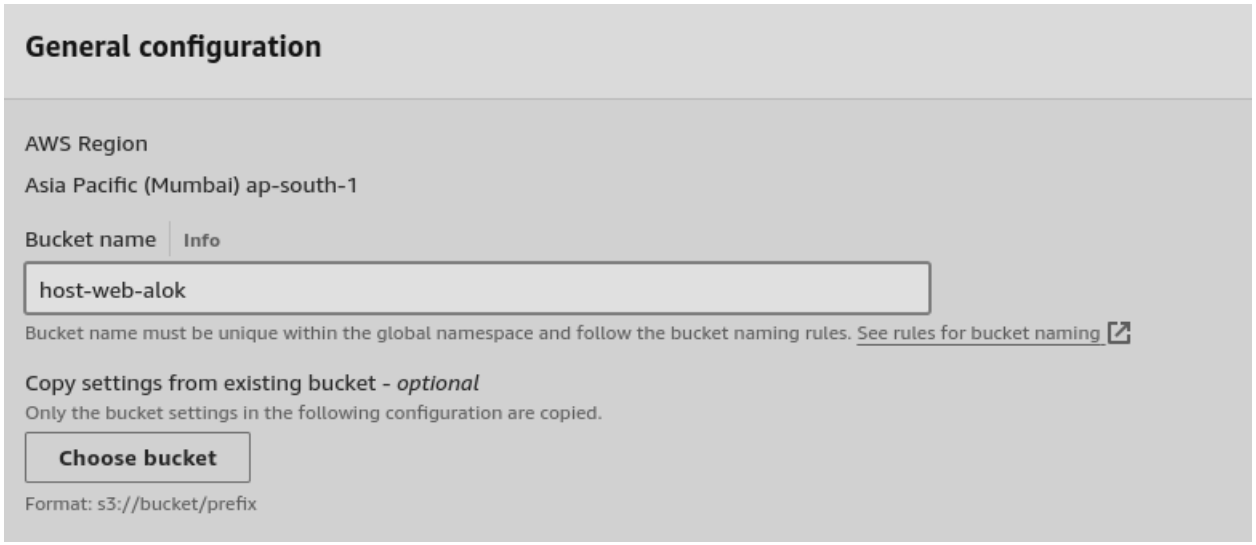
2. Hosting a static website on Amazon S3

1) Go to services then under Storage, select S3

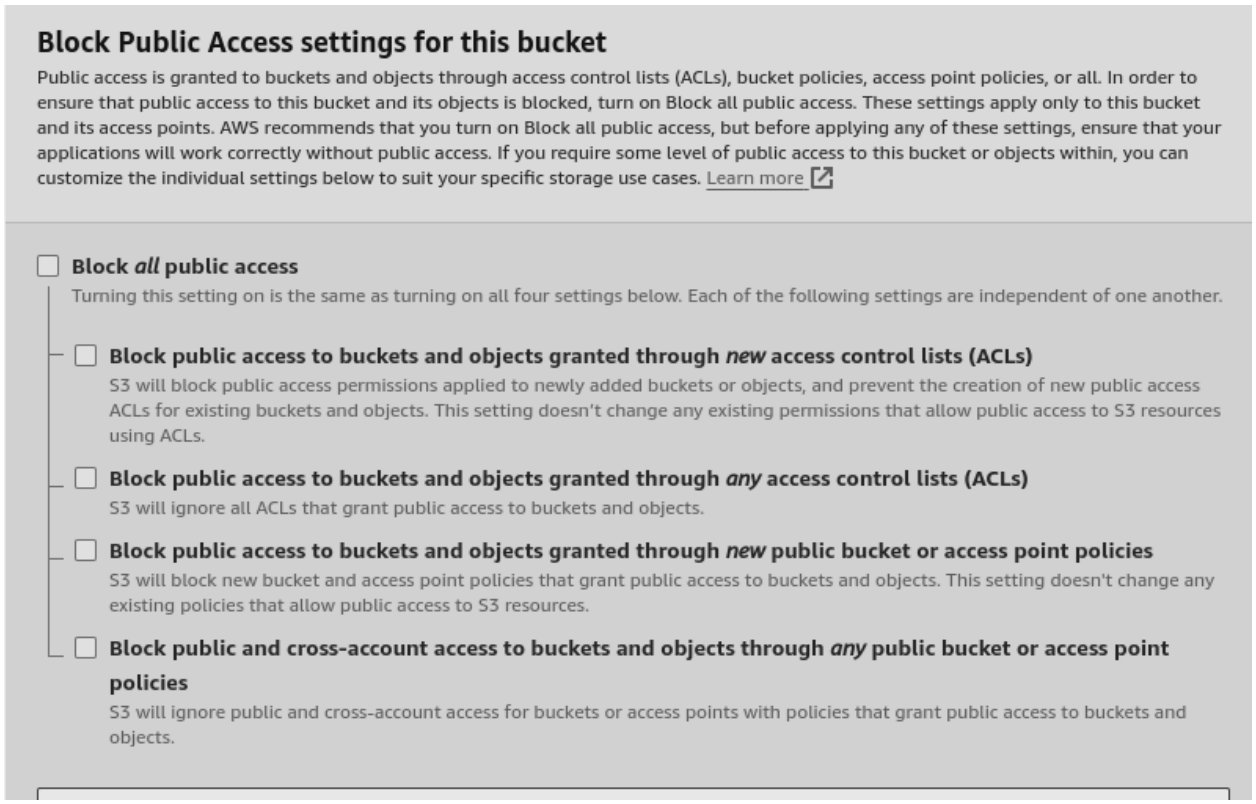


2) Create a new bucket

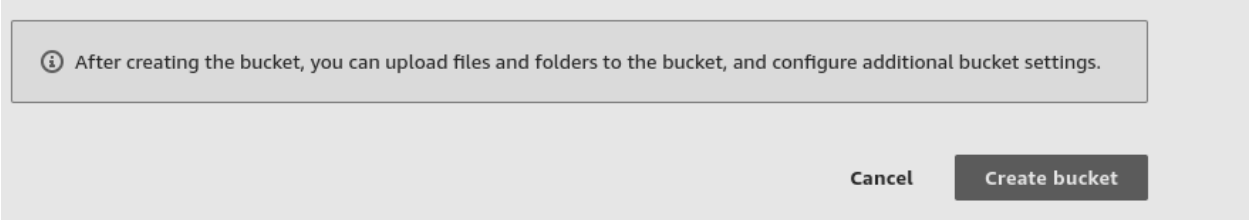
3) Enter Name of the bucket



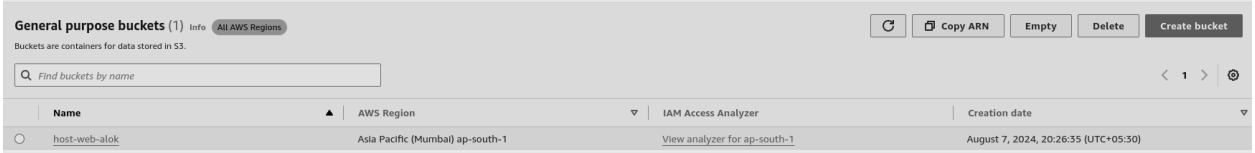
4) Disable “Block all public access” to make the contents of bucket publicly accessible



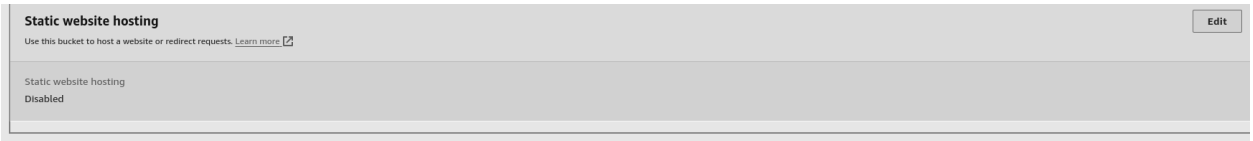
5) Leaving the remaining configuration to it’s default, click on Create Bucket



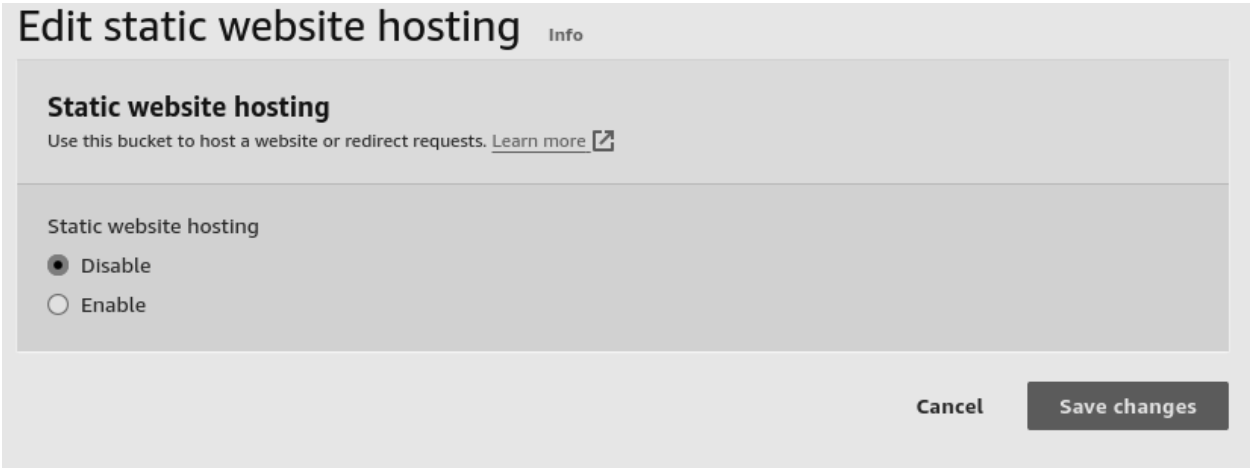
6) A new bucket will be created



7) Now to host a static web page, open the bucket we created, goto Properties, edit "Static Web hosting settings"

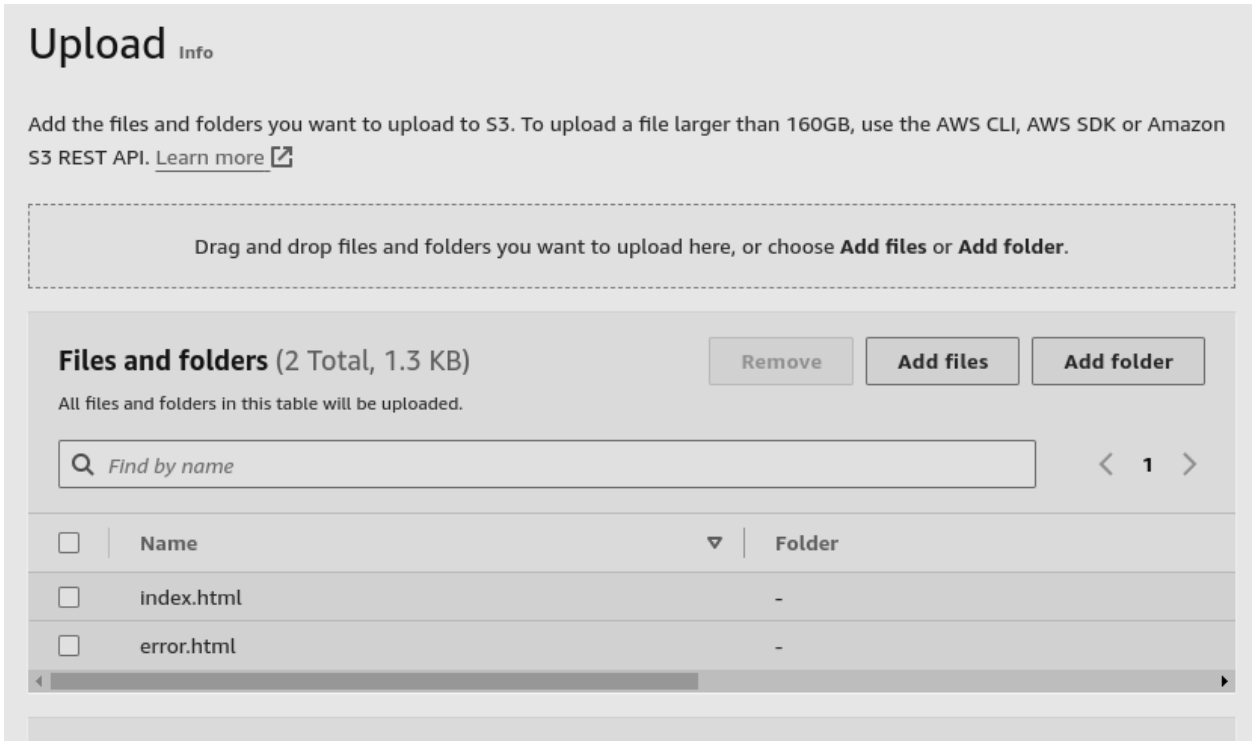


8) Enable Static web Hosting



9) After enabling, we need to enter the names of files two files, one for main file that would be displayed when page loads and the other when some error occurs. Name those files and save it

10) After naming, go to objects and upload the files you mentioned in the static web hosting settings page



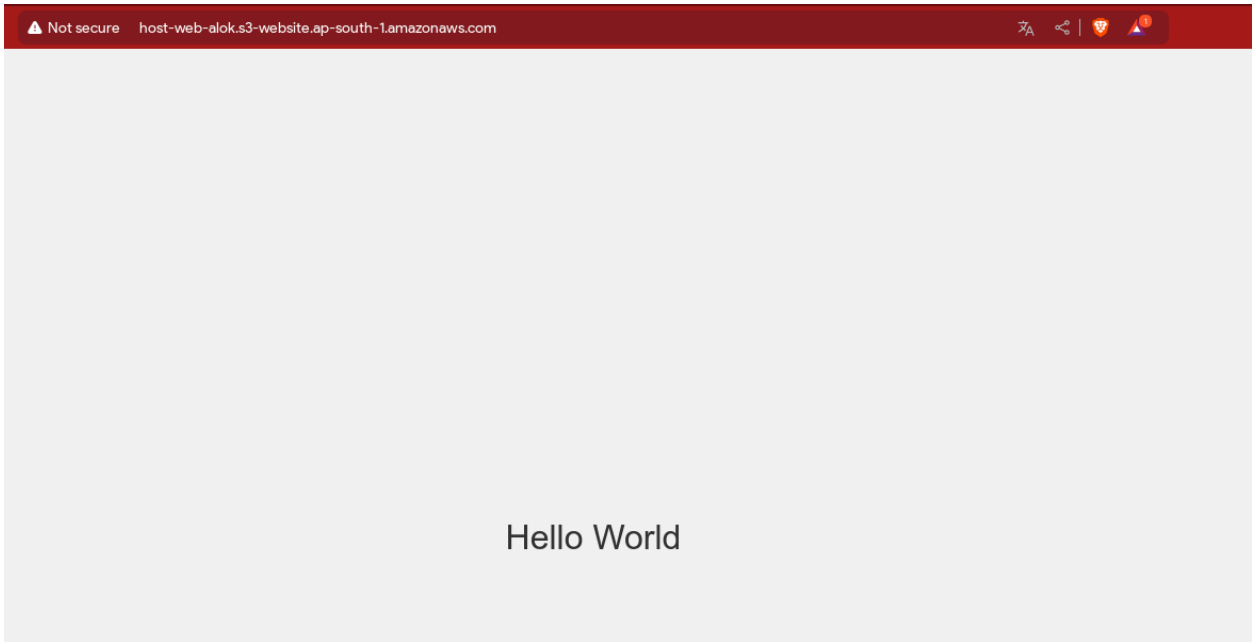
11) Now to make sure our page is accessible to public, we need to change bucket policy. To change it, go to permissions and edit Bucket policy.
Change resource arn as per your bucket name

Policy

```
1
2  {
3    "Version": "2012-10-17",
4    "Statement": [
5      {
6        "Sid": "PublicReadGetObject",
7        "Effect": "Allow",
8        "Principal": {
9          "AWS": "*"
10       },
11       "Action": "s3:GetObject",
12       "Resource": "arn:aws:s3:::host-web-alok/*"
13     }
14   ]
15 }
```

12) Now we can go to home page of our bucket and our static page will be loaded

Name: Alok Yadav Div: D15C Roll No.: 59



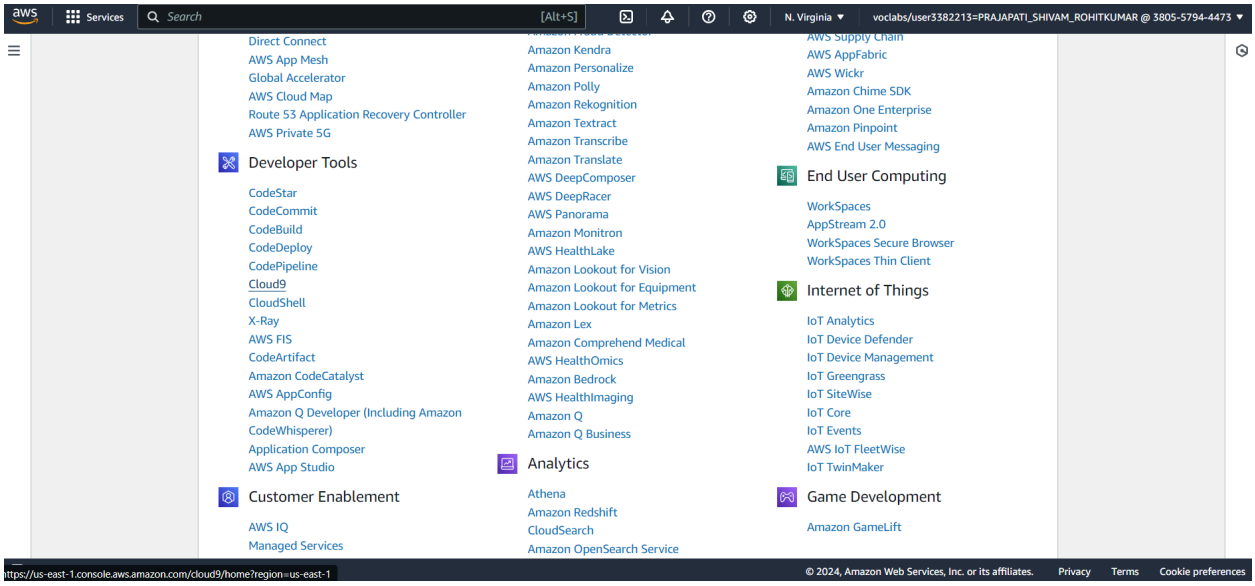
Name: Alok Yadav Div: D15C Roll No.: 59

Experiment No: 1(B)

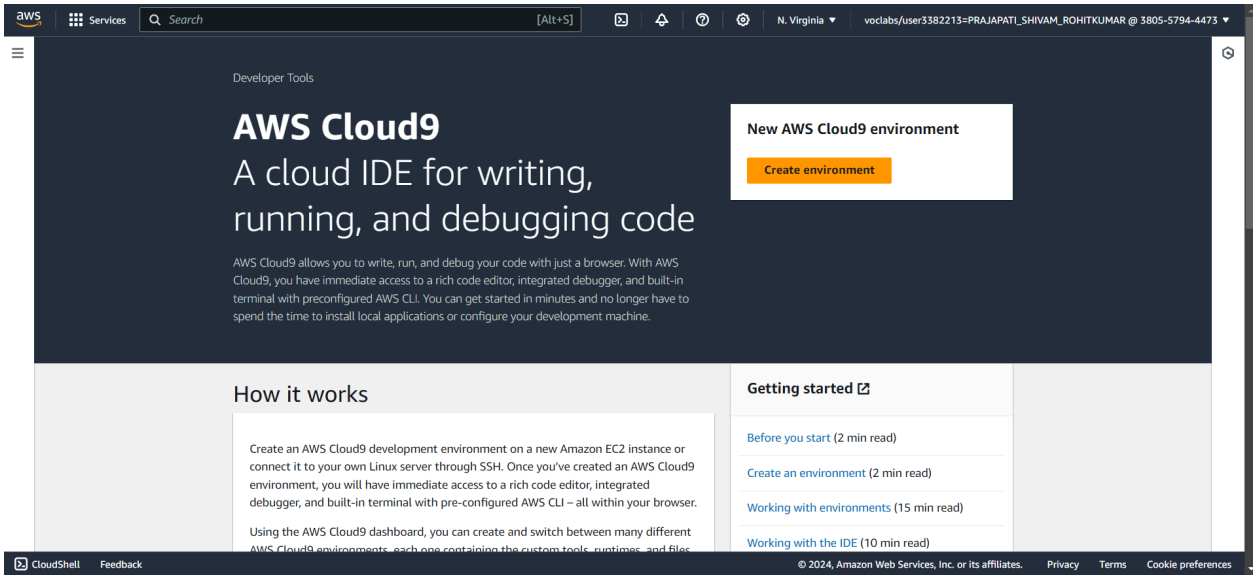
Aim: To understand the benefits of Cloud Infrastructure and Setup AWS Cloud9 IDE, Launch AWS Cloud9 IDE and Perform Collaboration Demonstration.

Step 1: Set up a Cloud9 environment.

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

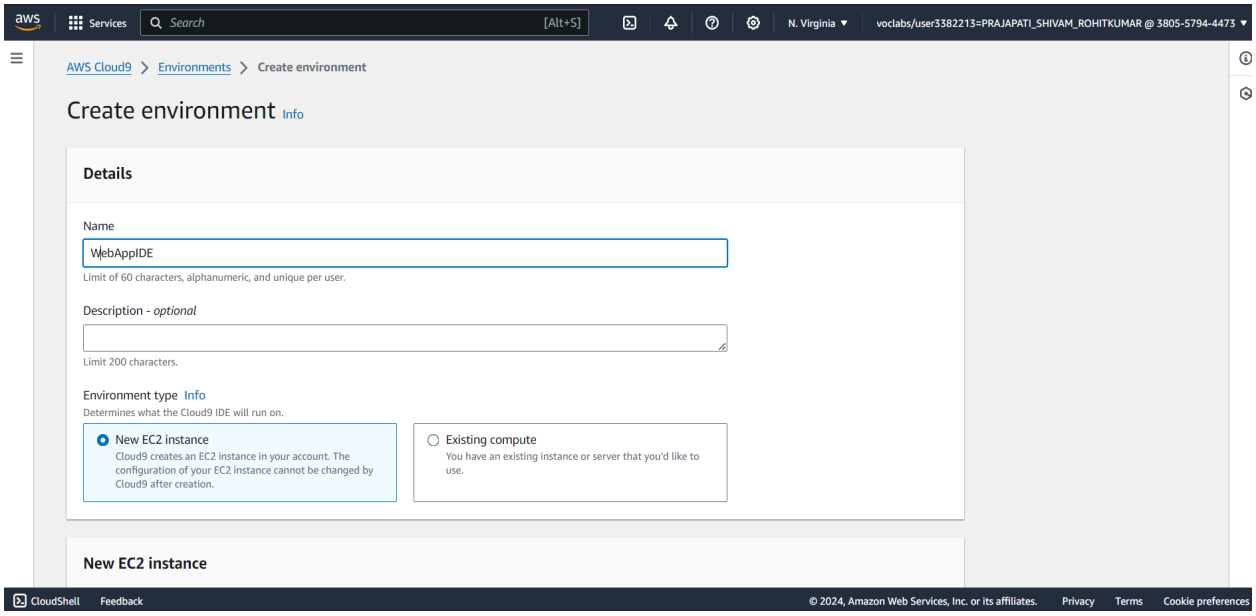


2) Click on create environment

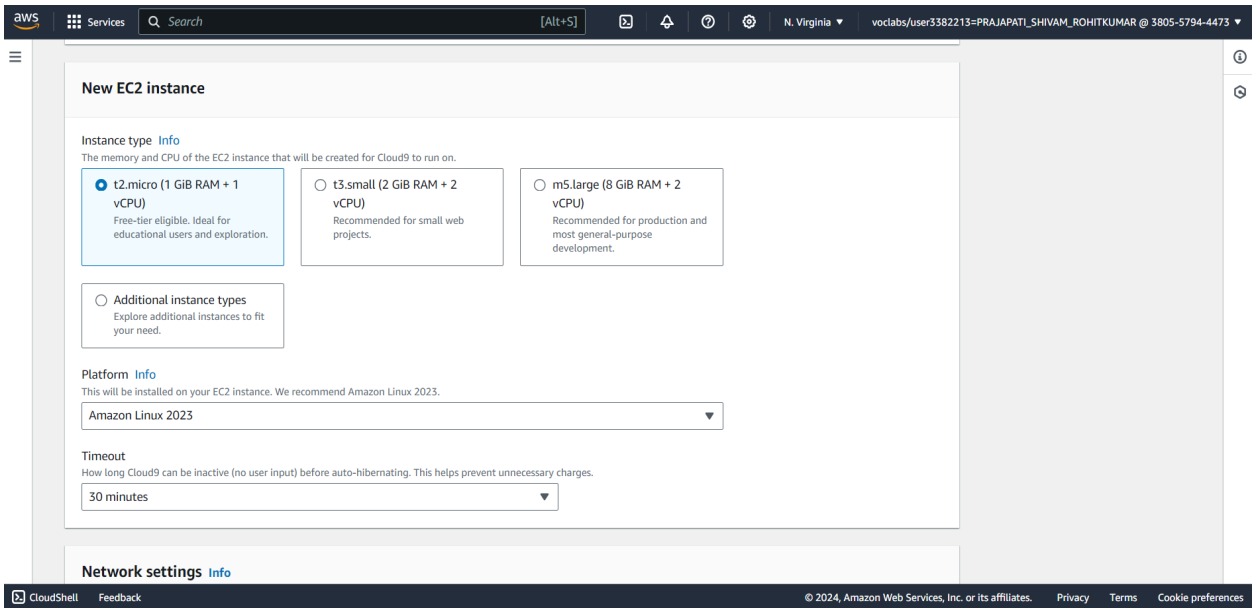


Name: Alok Yadav Div: D15C Roll No.: 59

3) Give the name to your Environment ,keeping the other settings as default like environment type should be New EC2 instance

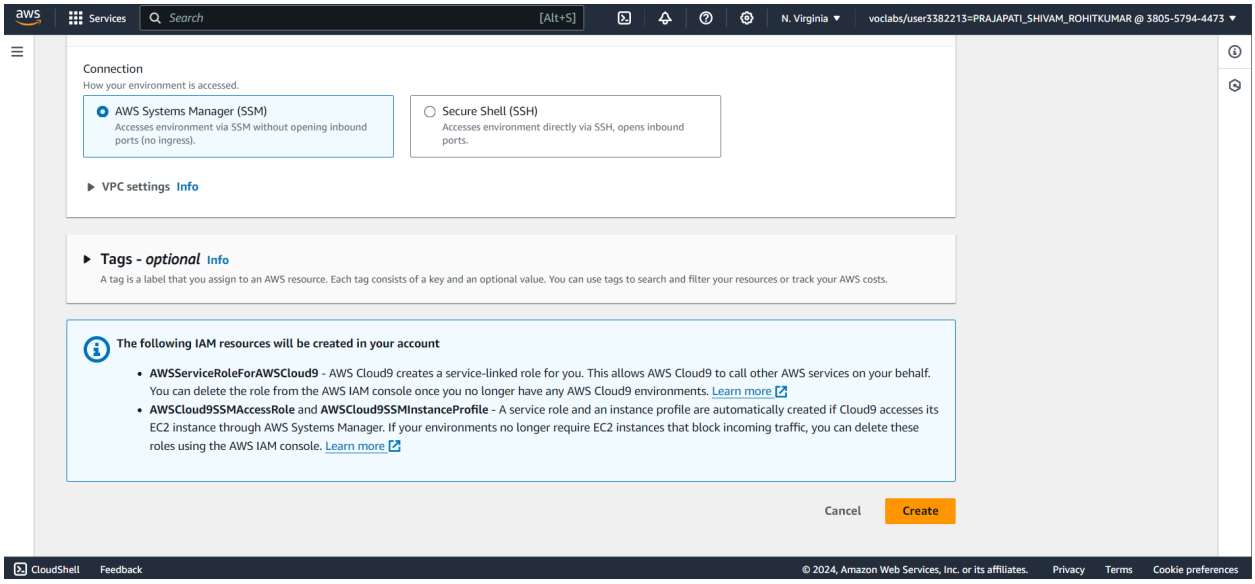


4) Select the correct platform type as shown below and keep the others details as default like instance type as t2.micro which gives the user 1GB RAM + 1 Virtual CPU

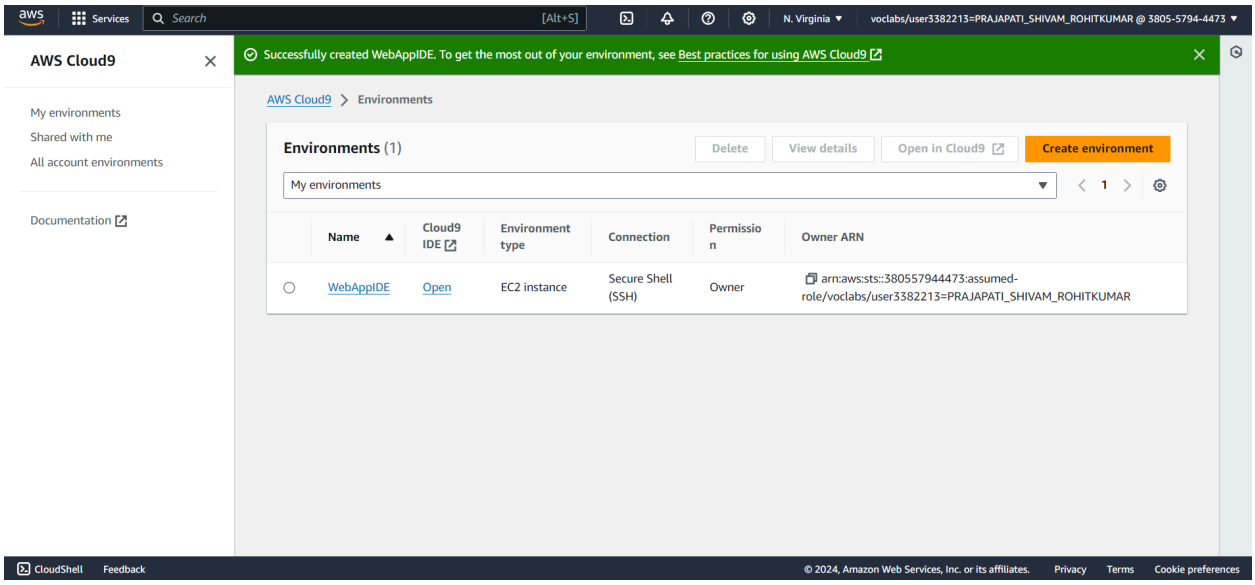


Name: Alok Yadav Div: D15C Roll No.: 59

5) Click on SSH under connection type in network settings if we go for AWS Manager(SSM) then it won't allow to create an environment then click on Create



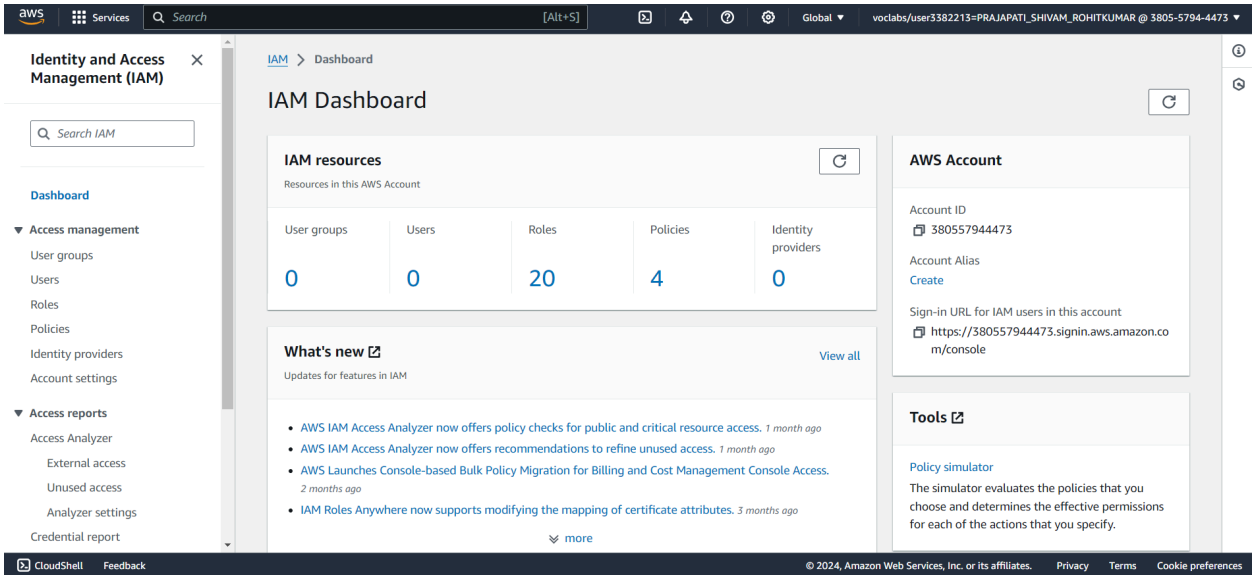
6) Successfully created the environment so now click on open



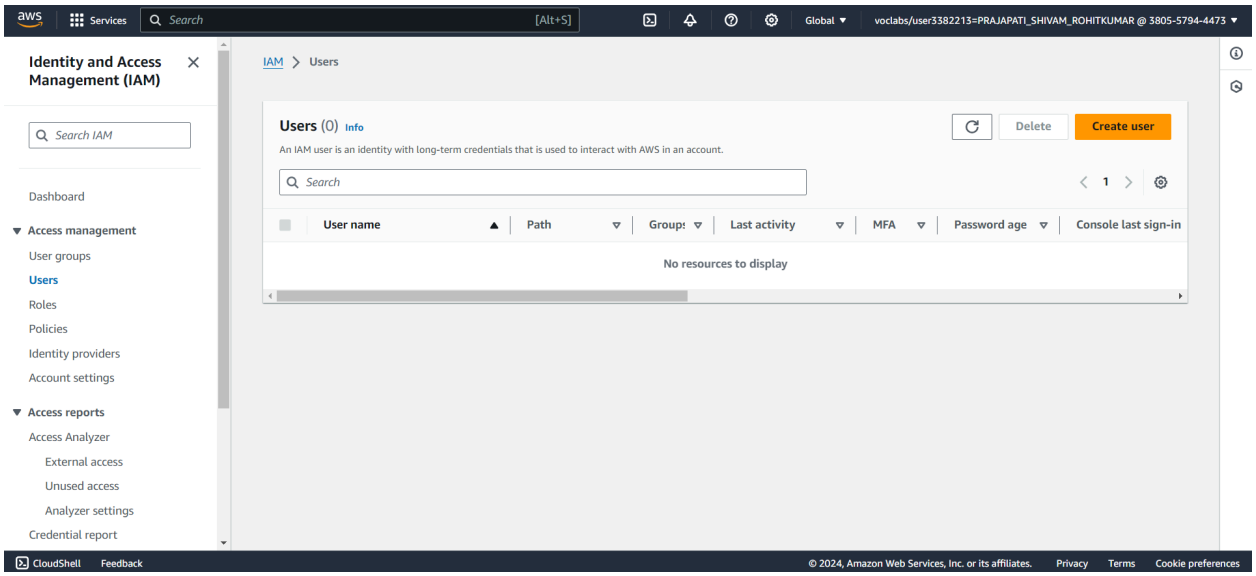
Name: Alok Yadav Div: D15C Roll No.: 59

Step 2: Creating IAM user.

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

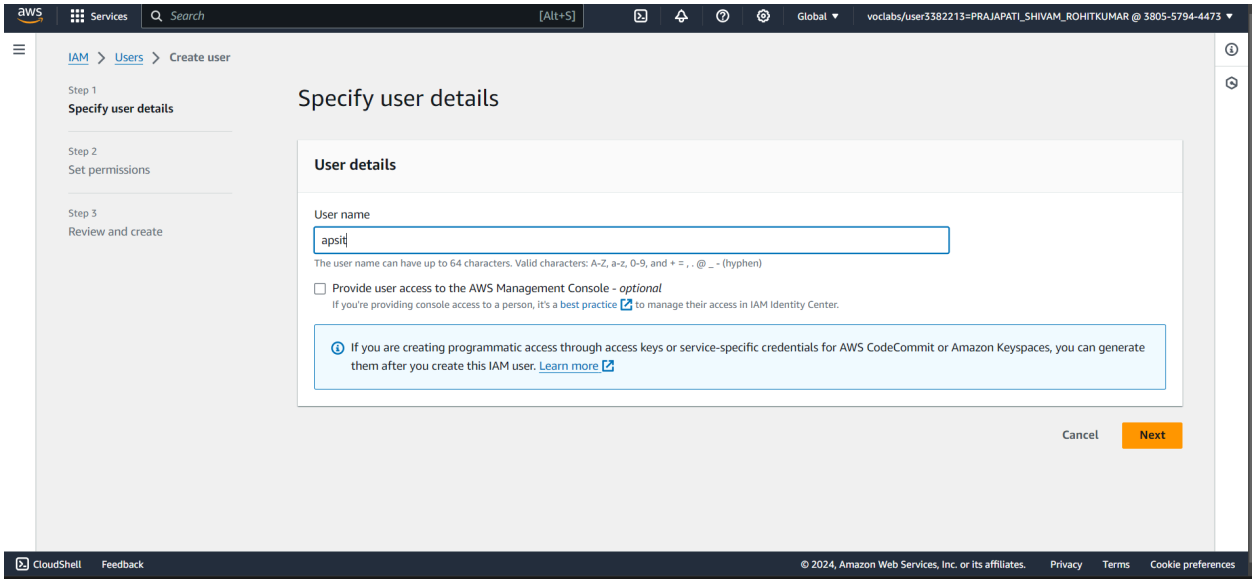


2) Click on the create user

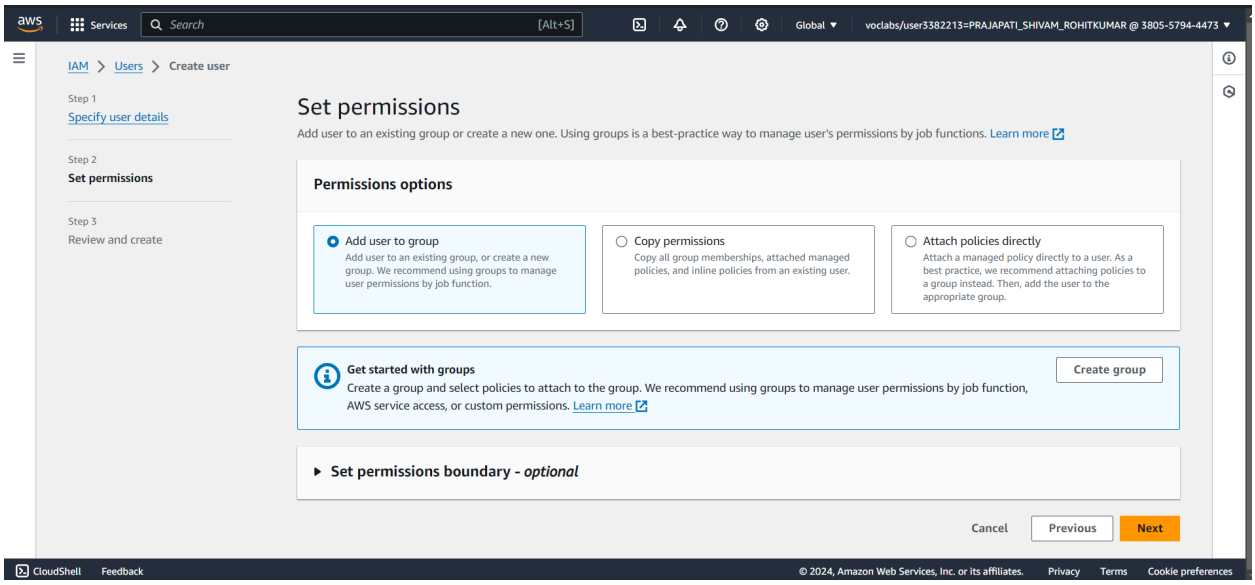


Name: Alok Yadav Div: D15C Roll No.: 59

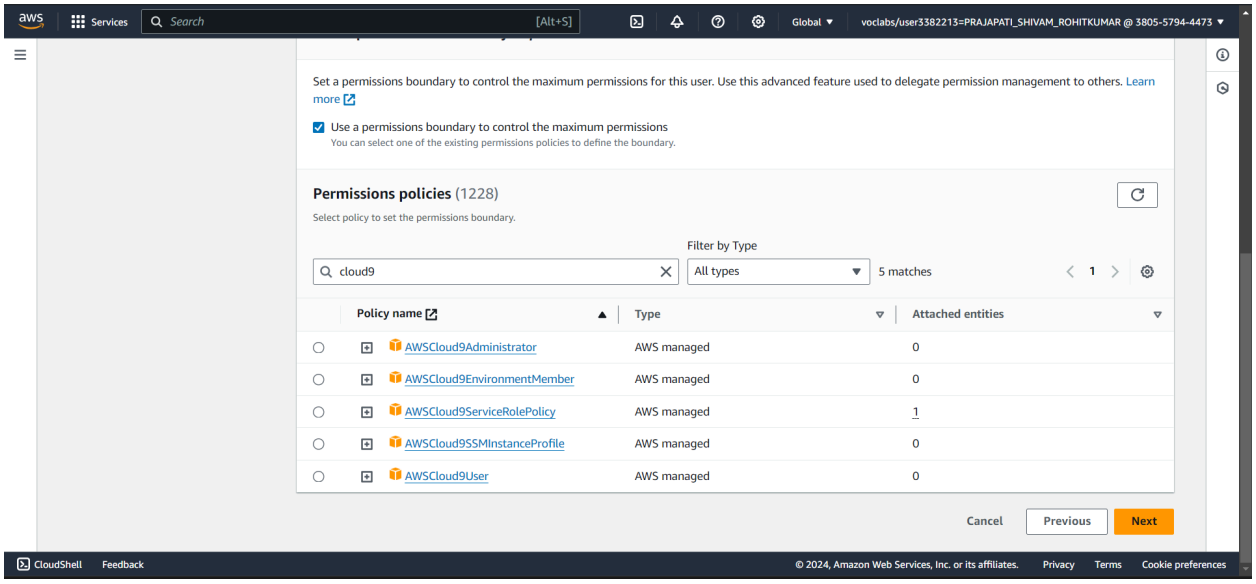
3) Write the name of the user you want to add and click on next



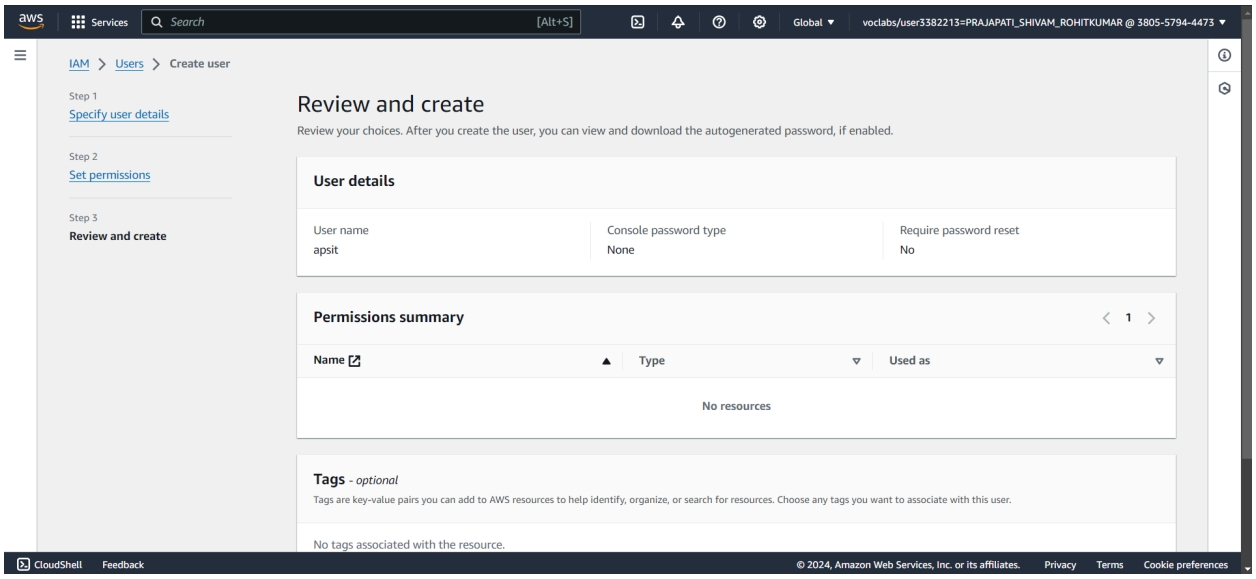
4) Select add User to Group. If there are no user groups on your accounts, you will have to create one. Click on Create Group. Click on the drop down menu of the set permissions boundary



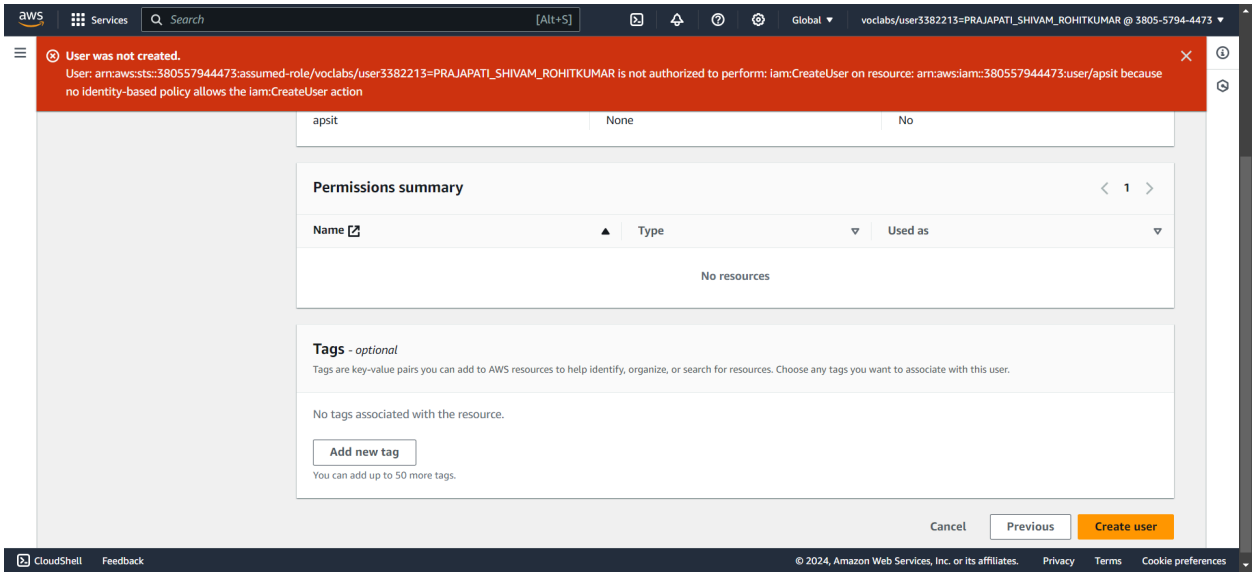
5)Click on the checkbox and search for cloud9 under permissions policies ,click on next



6) Scroll down and click on create user



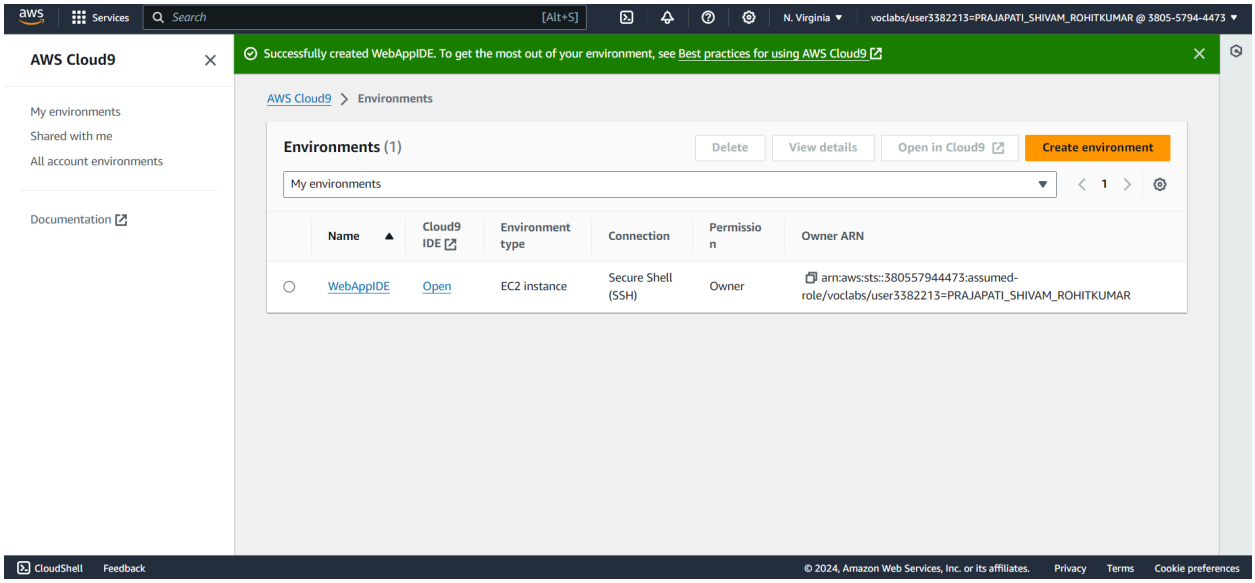
Name: Alok Yadav Div: D15C Roll No.: 59



When we go to add user to a group, the AWS Academy account throws an error as we do not have the permissions to create a group. So we have to use our personal AWS account for this part.

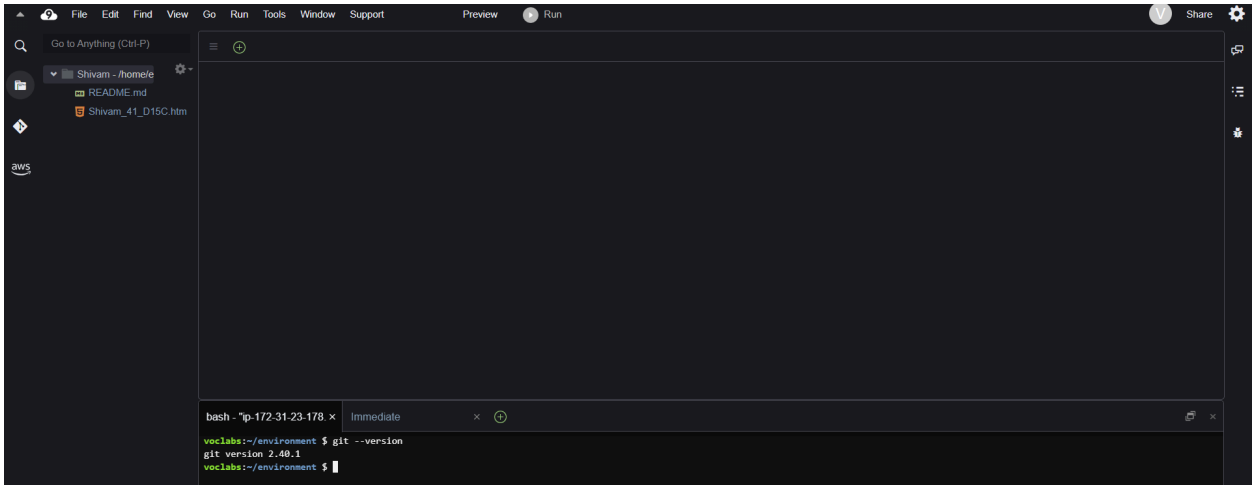
Step 3: Working on Cloud9 IDE

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

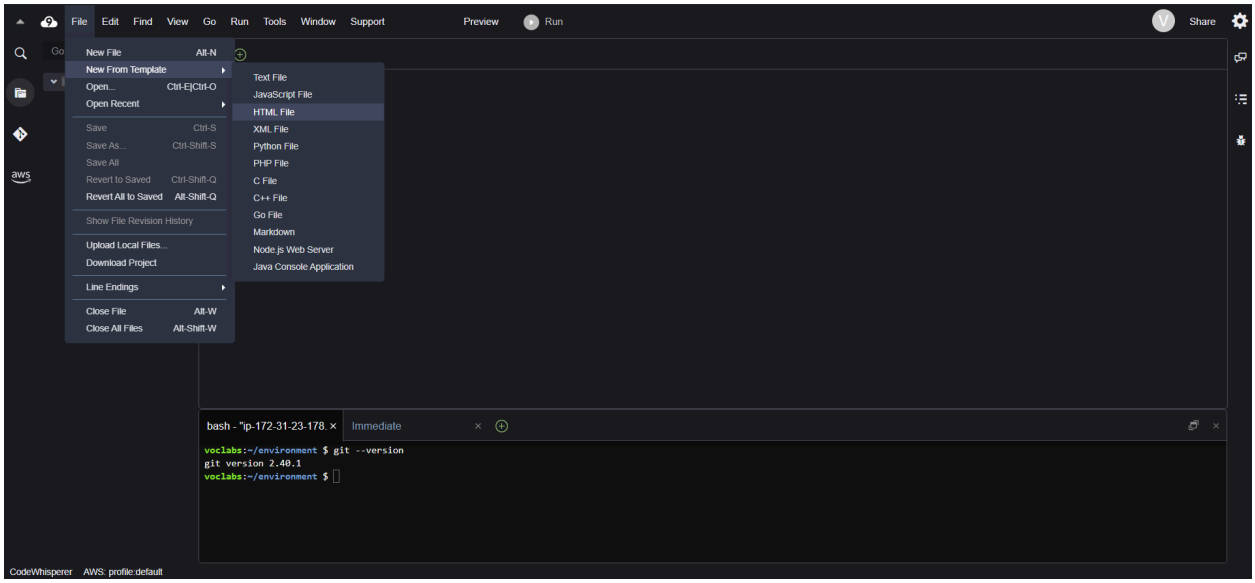


Name: Alok Yadav Div: D15C Roll No.: 59

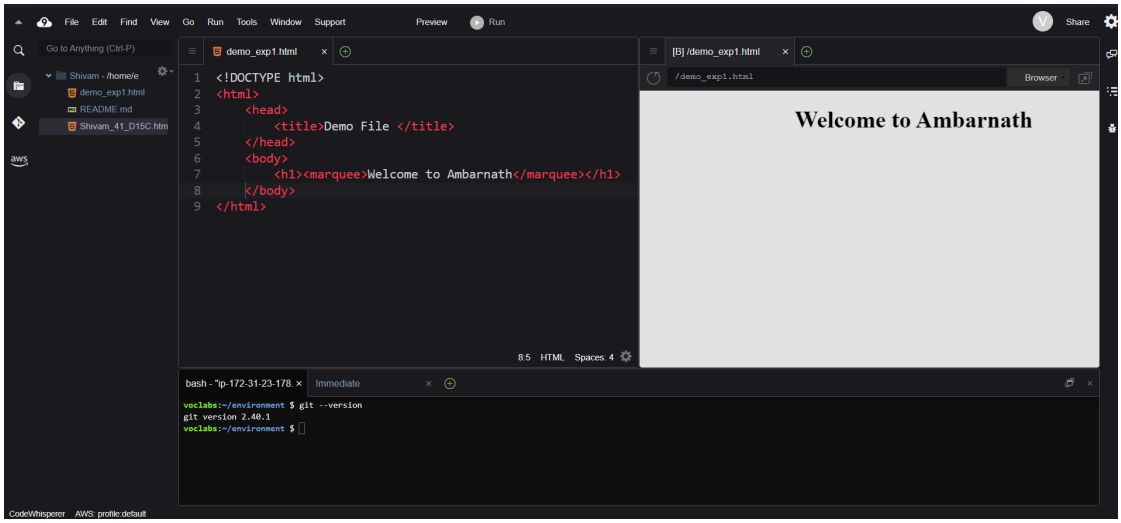
2) This is the Cloud9 IDE interface. The major part of the screen is the coding IDE. There is a command console just below it. For example, the command git --version is run.



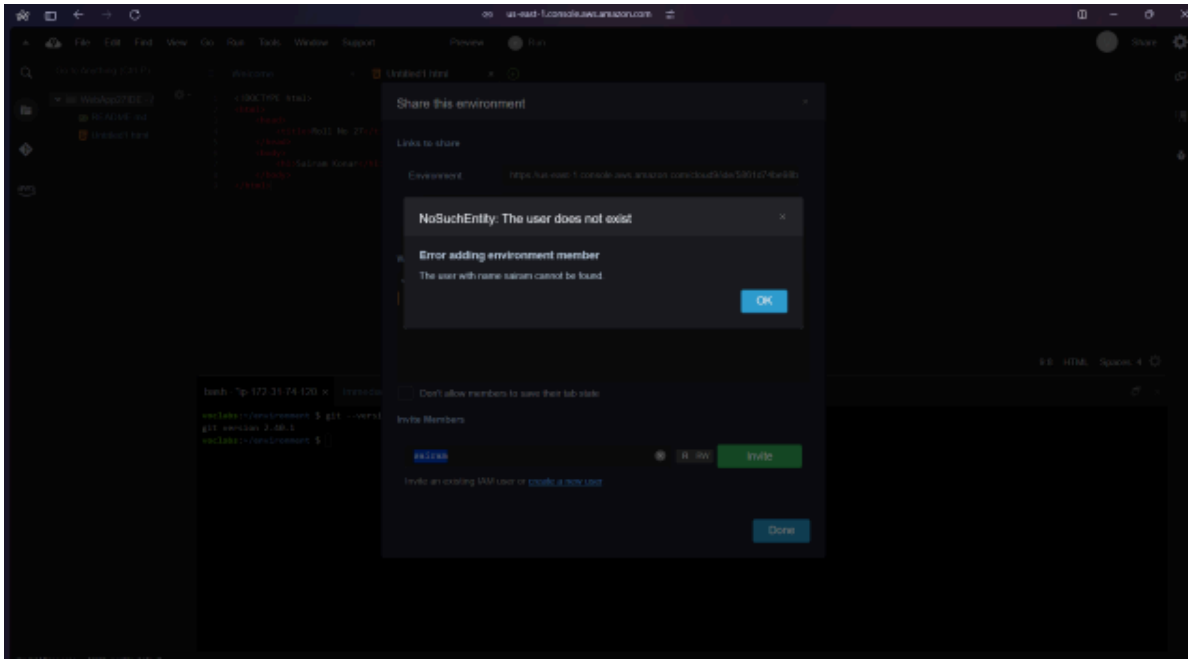
3) To add a file, click on file. For this experiment, we are to add an HTML file. So go to File → New From Template → HTML file. This gives a basic HTML template on the coding IDE



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



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



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

Name: Alok Yadav Div: D15C Roll No.: 59

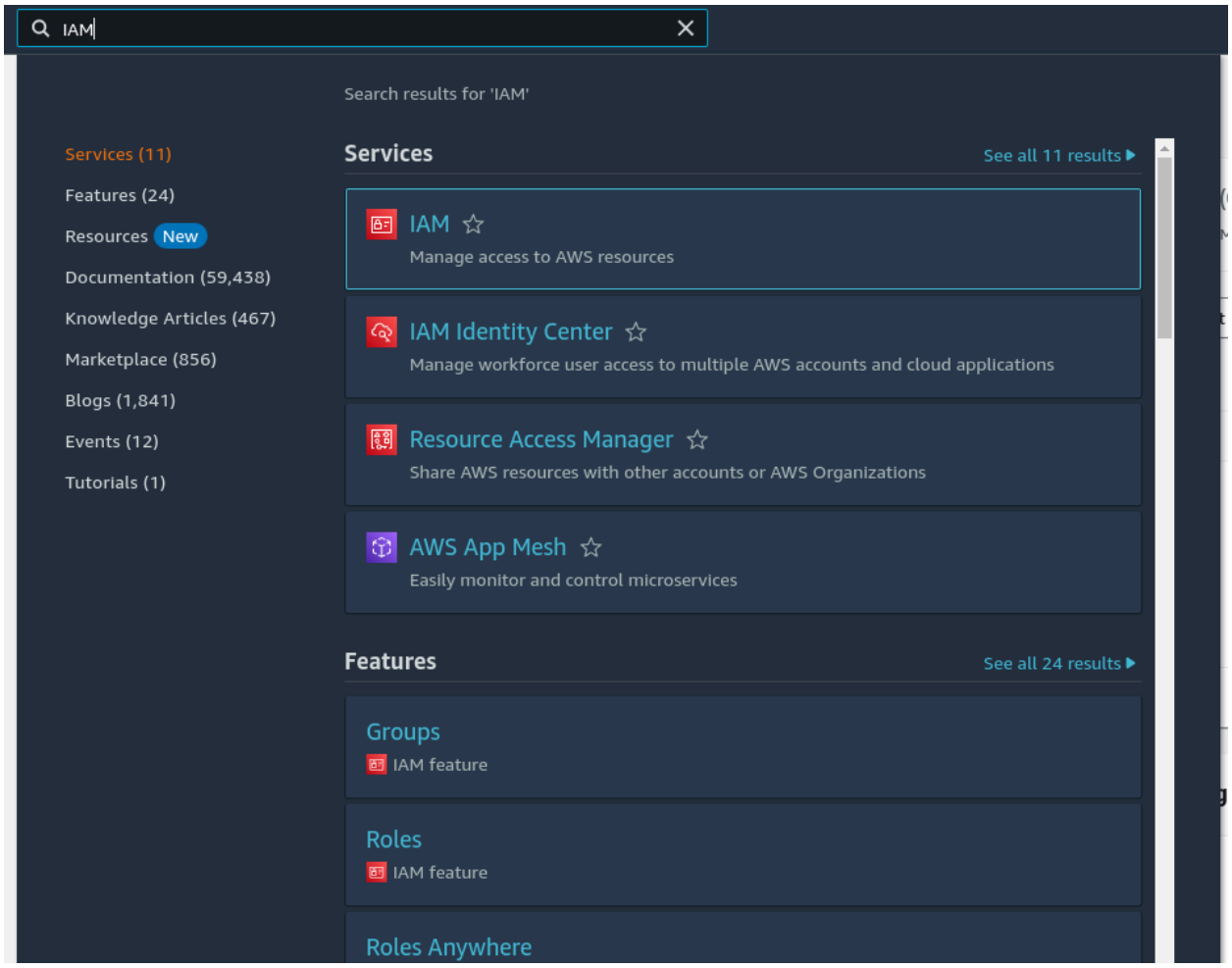
Experiment No. 2

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

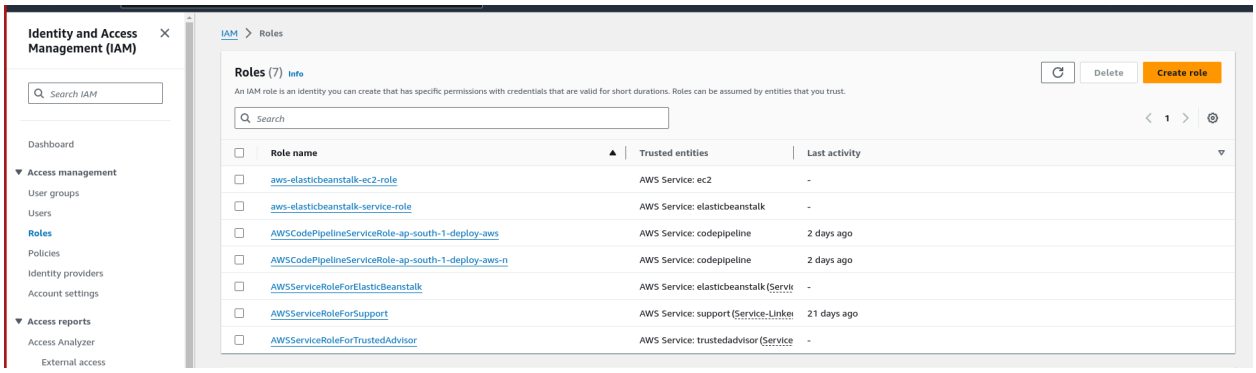
Steps:

Initially we will create a new Role in IAM

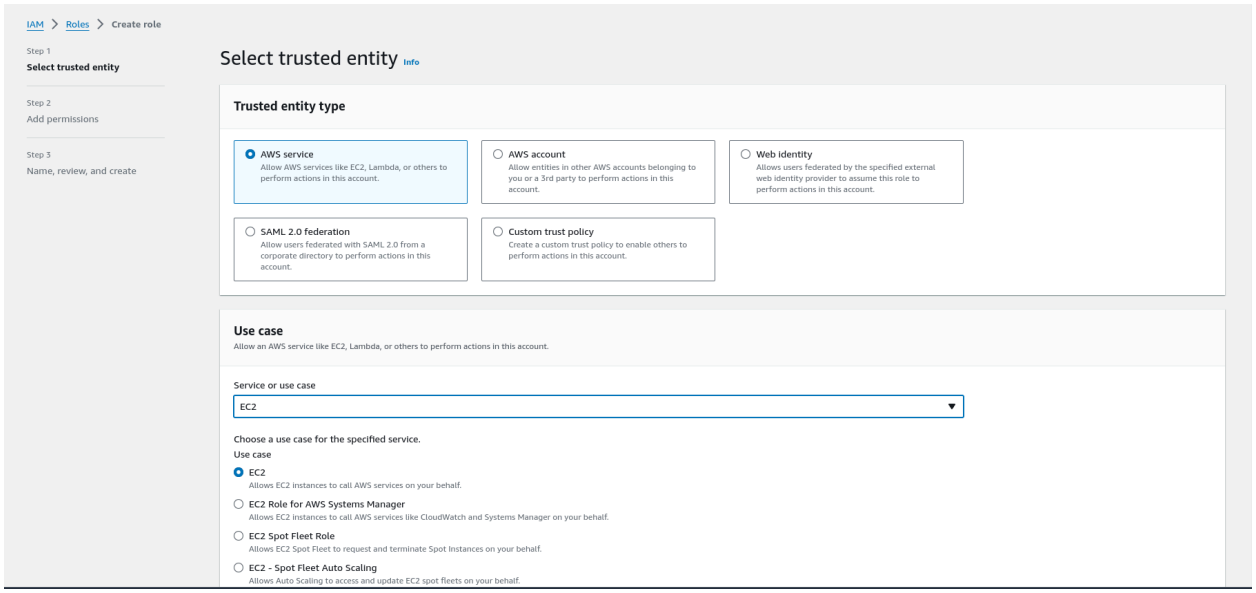
1. Search for IAM in search box

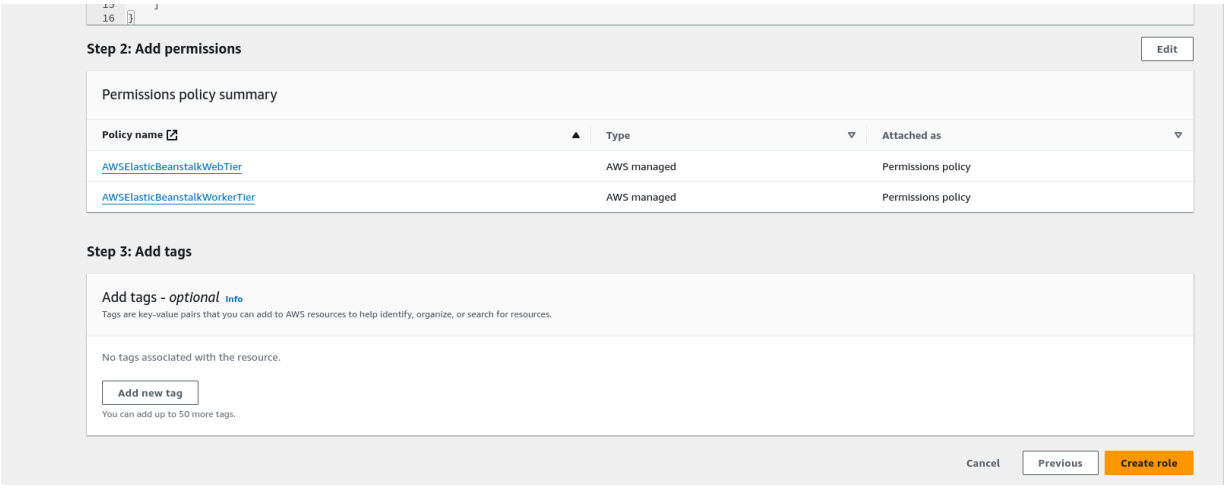


2. Go to Roles, and select AWSElasticBeanstalkWebtier and AWSElasticBeanstalkWorkertier, and create a role

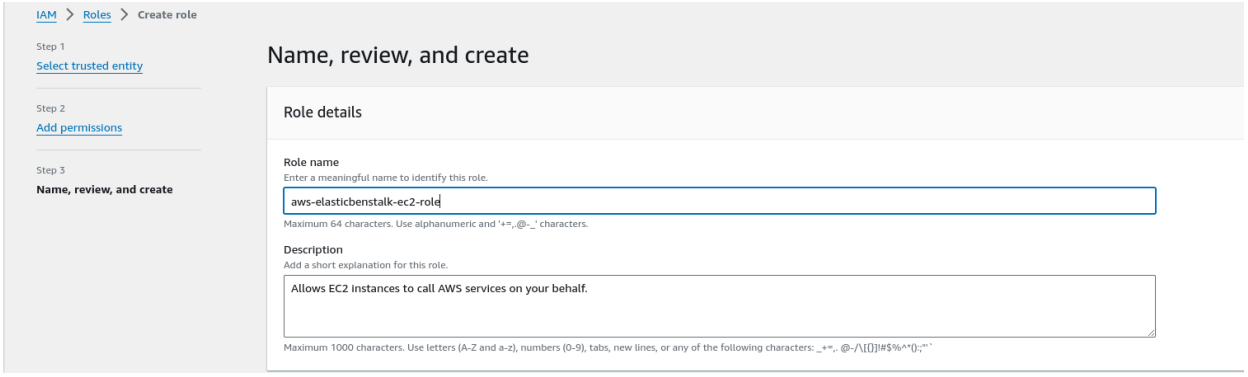


3. Select entity type as **AWS Service**

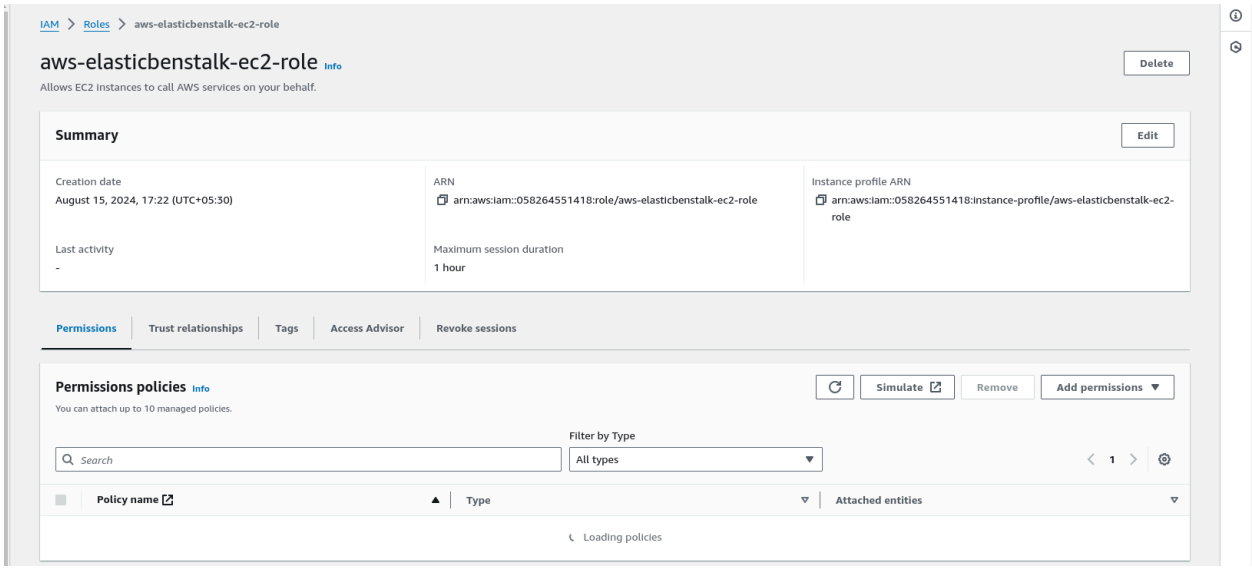




4. Give name to newly created role such as **aws-elasticbenstalk-ec2-role**

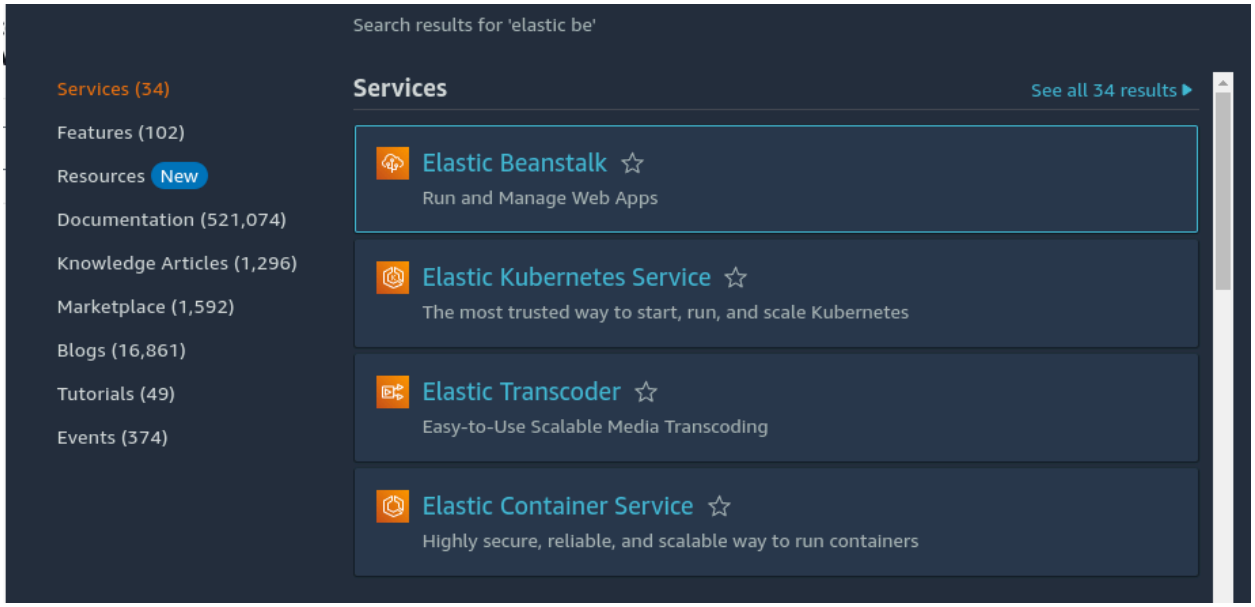


5. Summary of newly created role

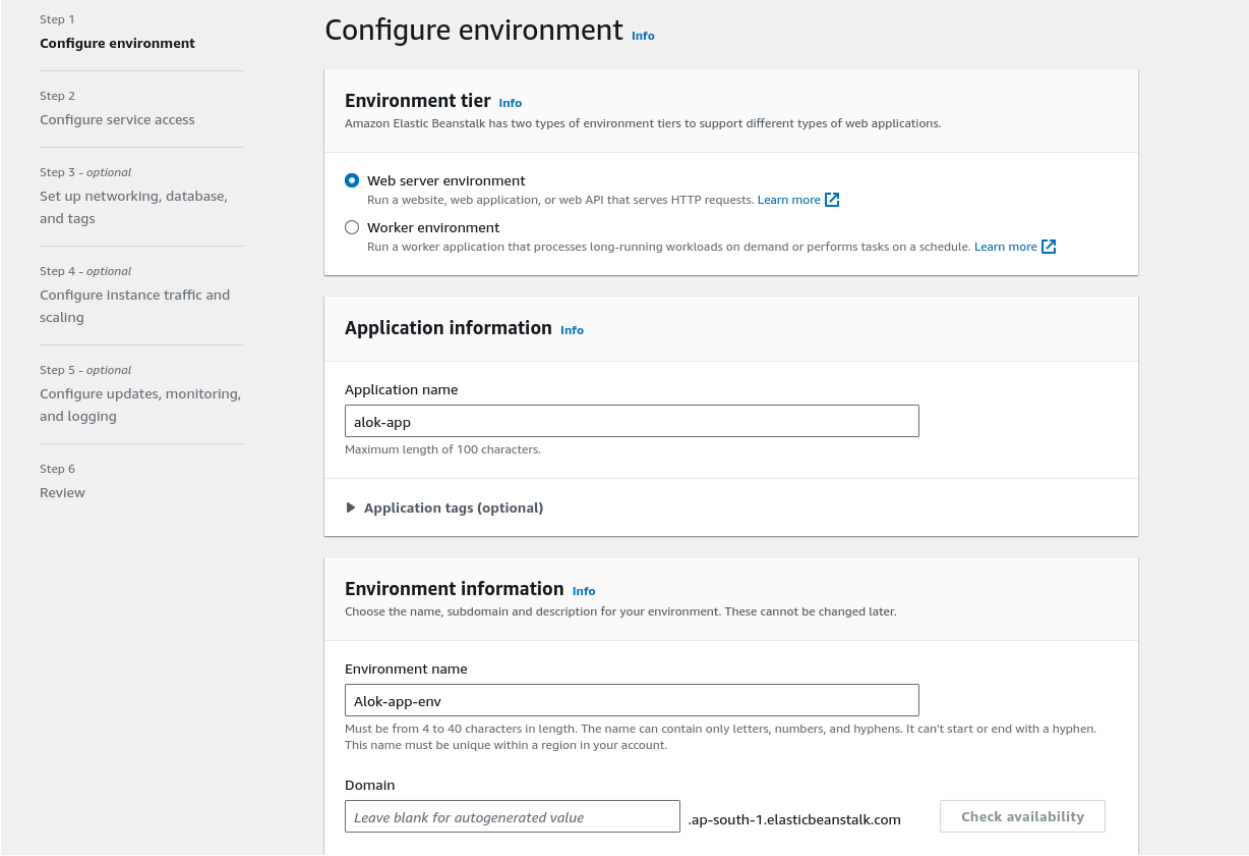


Now we have to create Elastic Beanstalk Environment

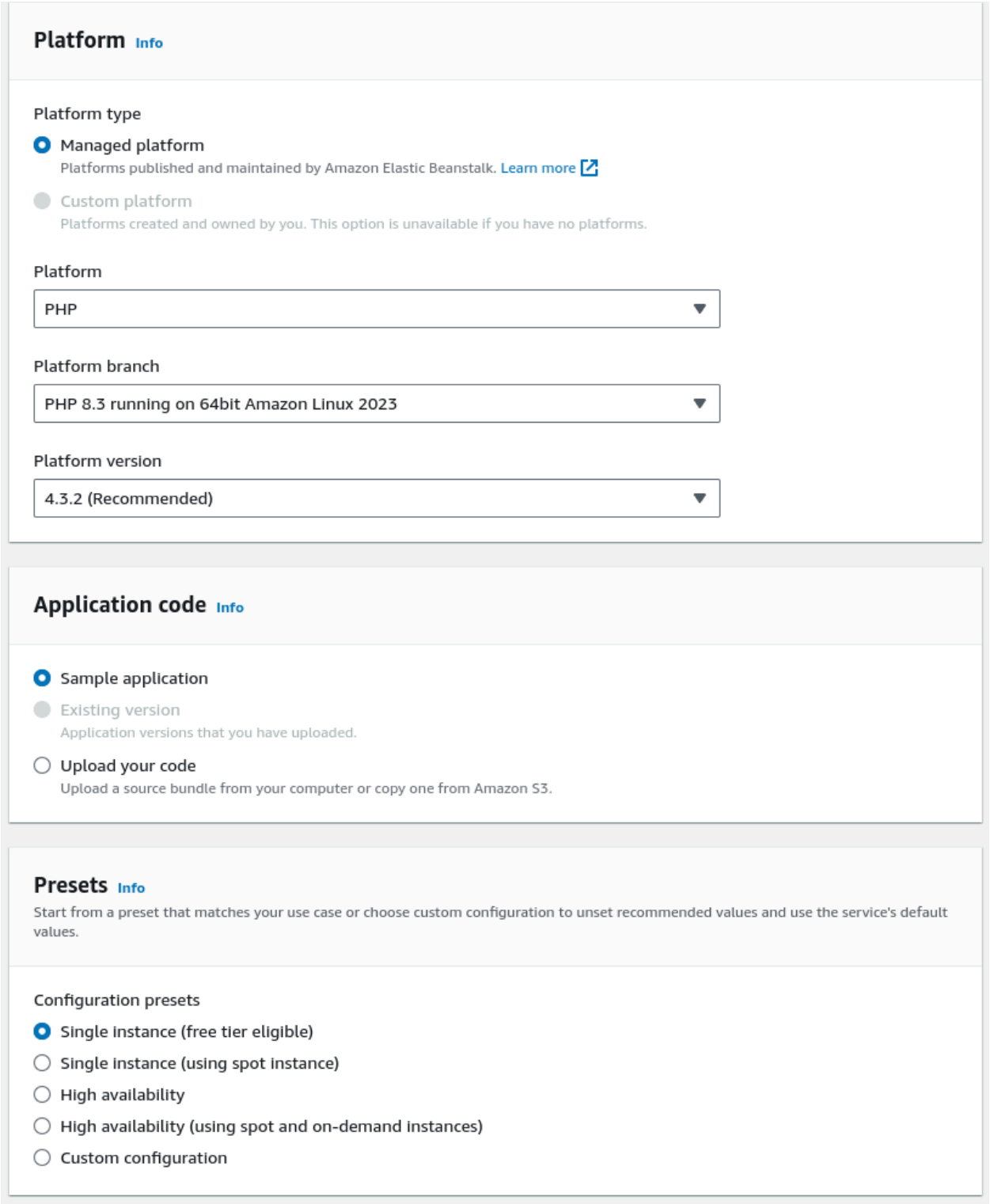
6. Search **Elastic Beanstalk** and proceed with it



7. Create a new Environment and name it.

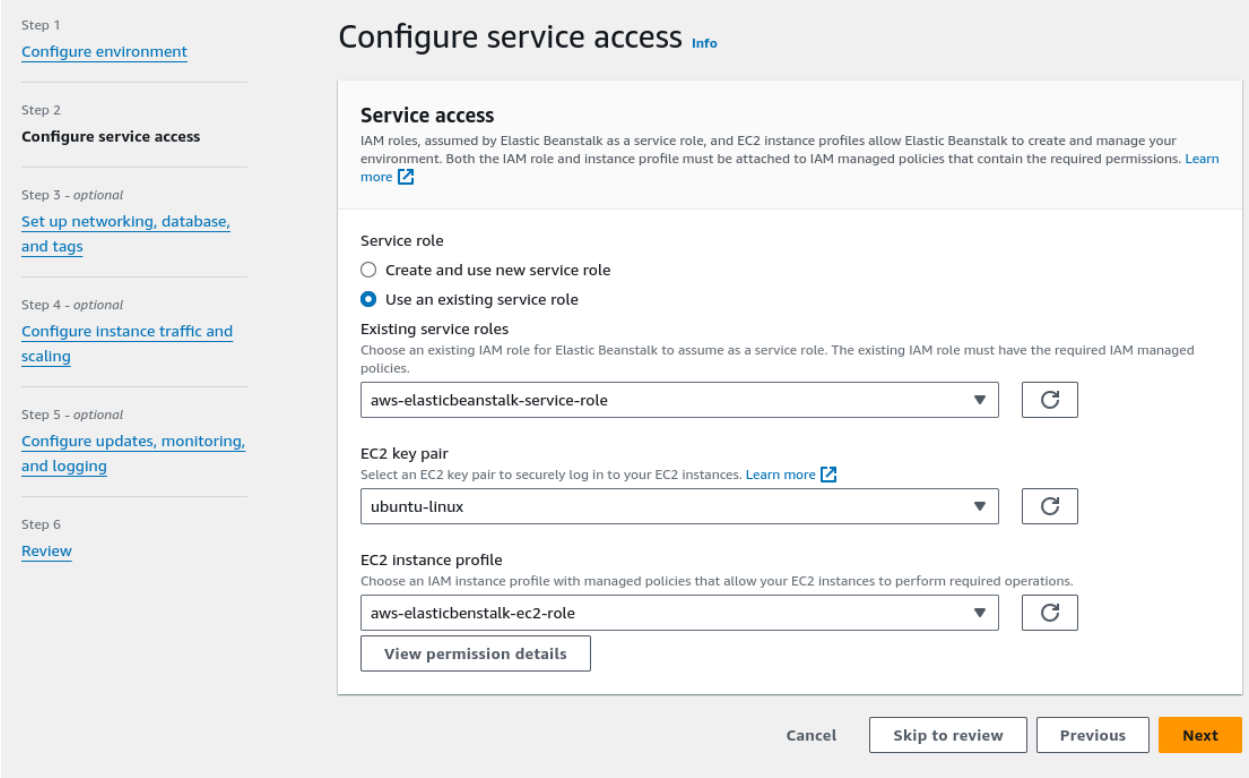


8. Select Platform as PHP, Application code as **Sample Application**, presets **Single Instance**



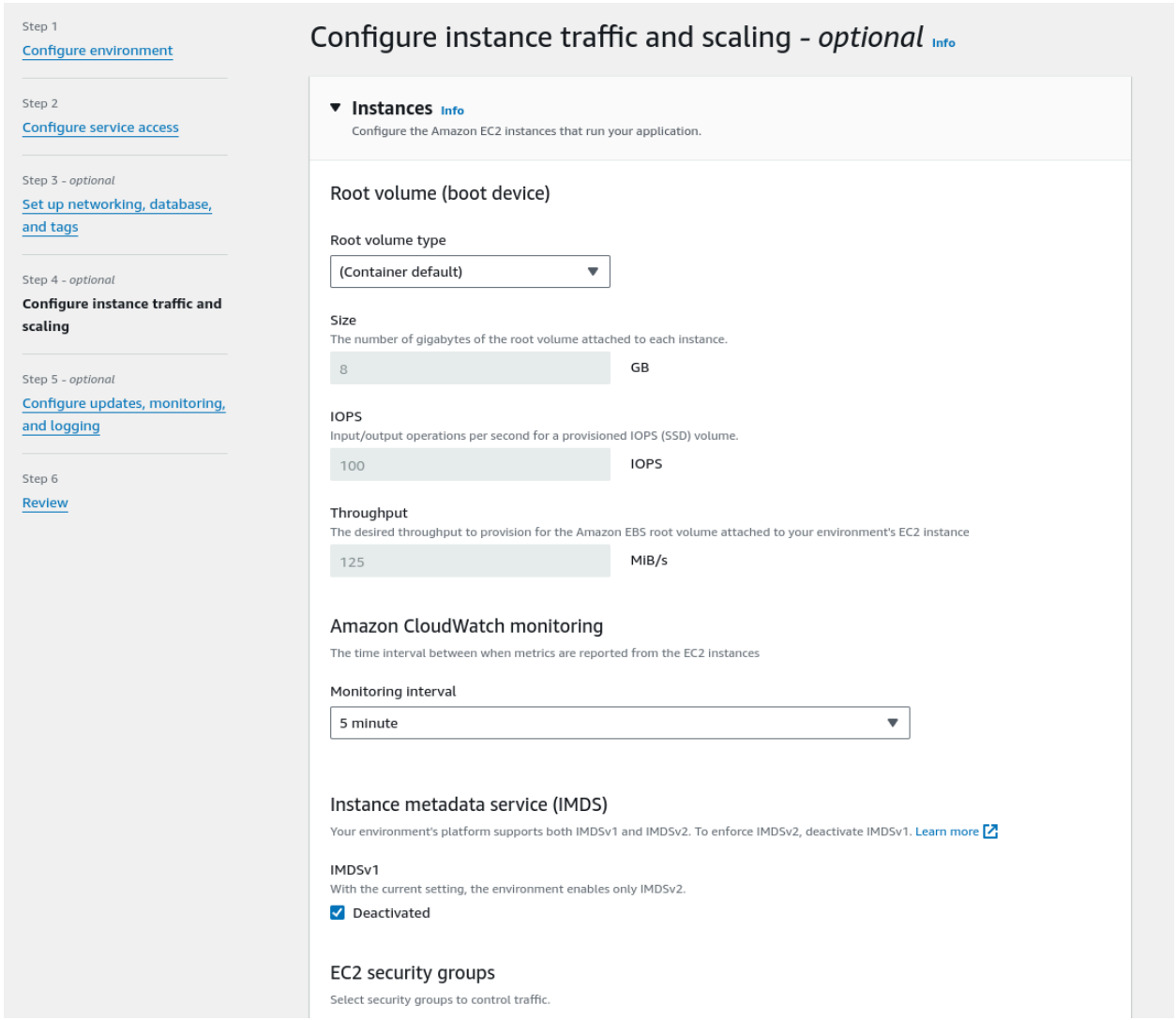
9. Under Service access settings, select **Use an existing service role**. Name service role as **aws-elasticbeanstalk-service-role** and EC2

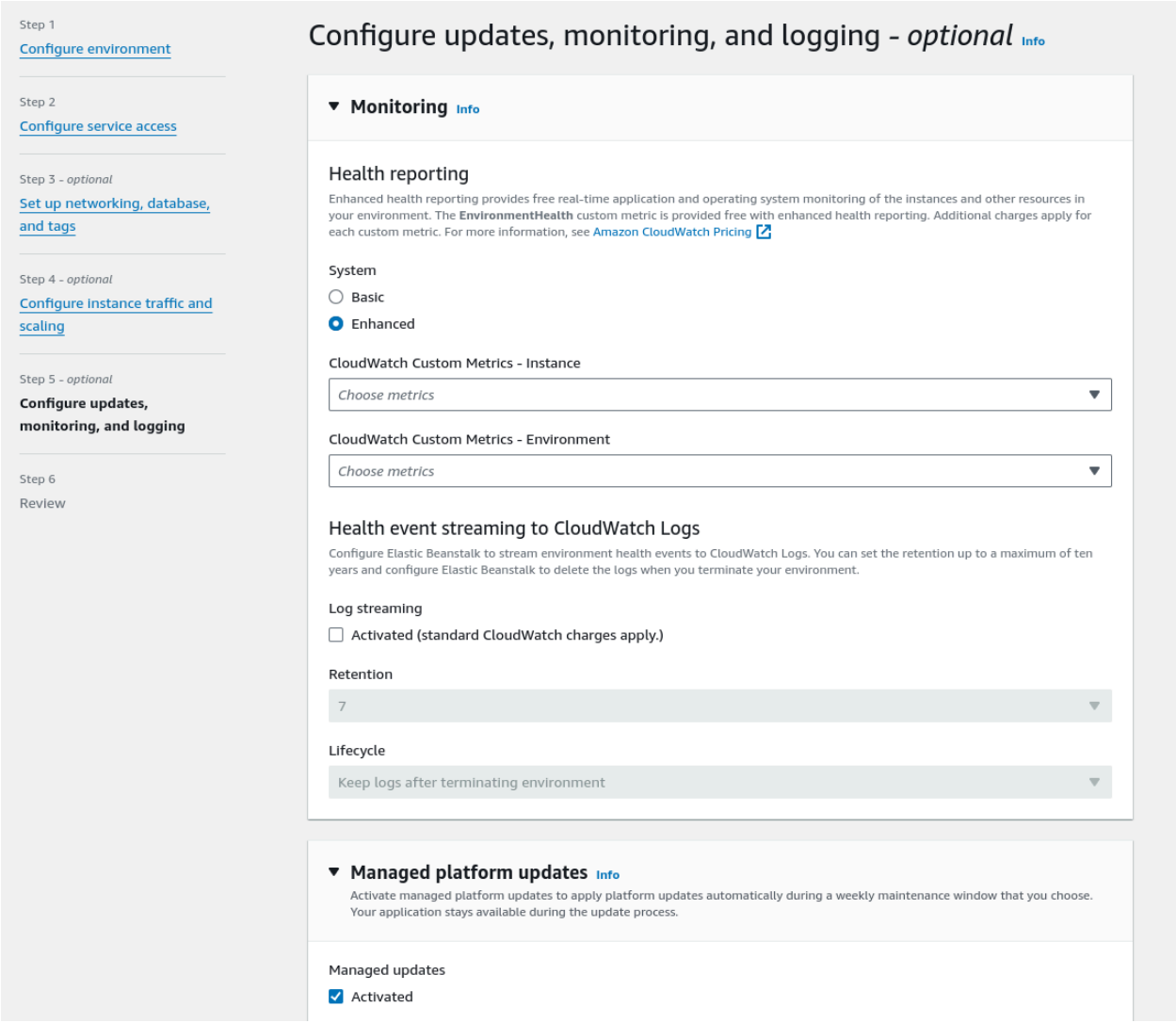
instance profile as **aws-elasticbenstalk-ec2-role** which we had created earlier



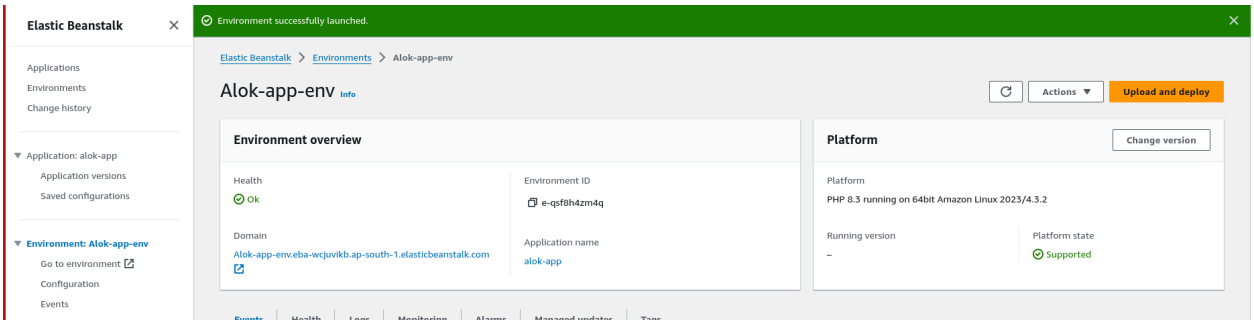
10. Keep below settings as it as

Name: Alok Yadav Div: D15C Roll No.: 59



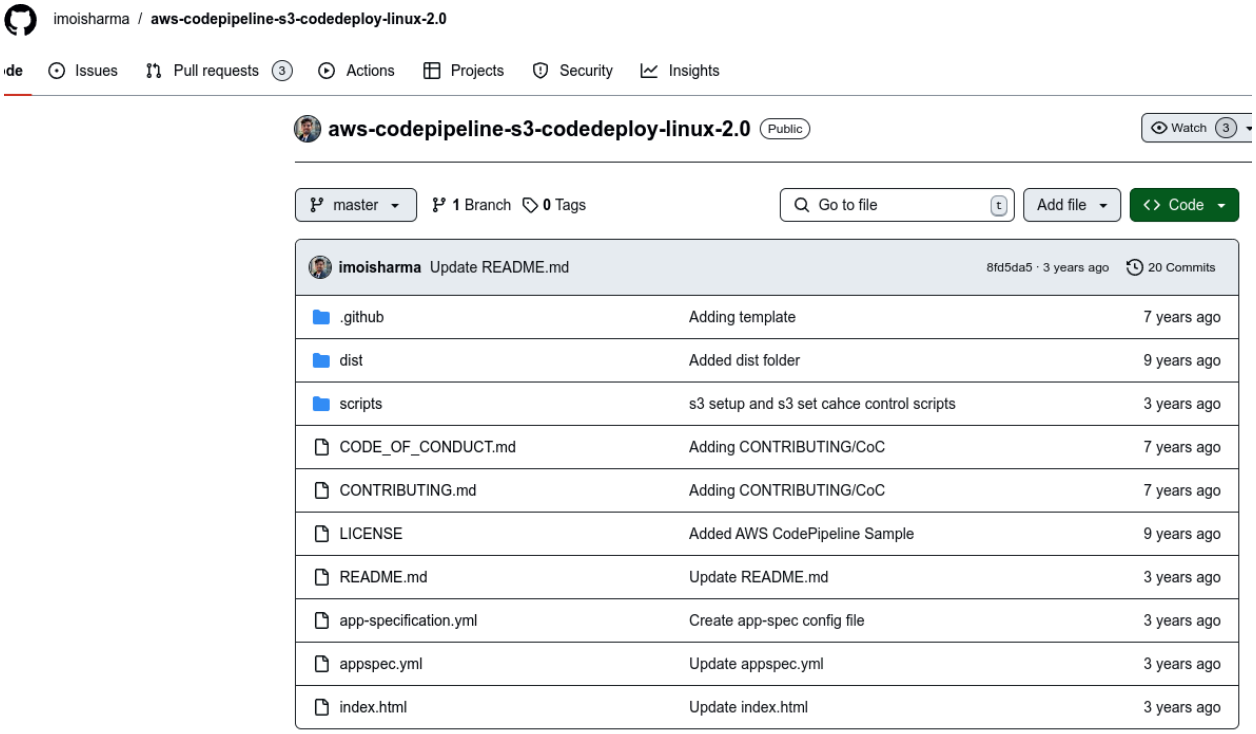


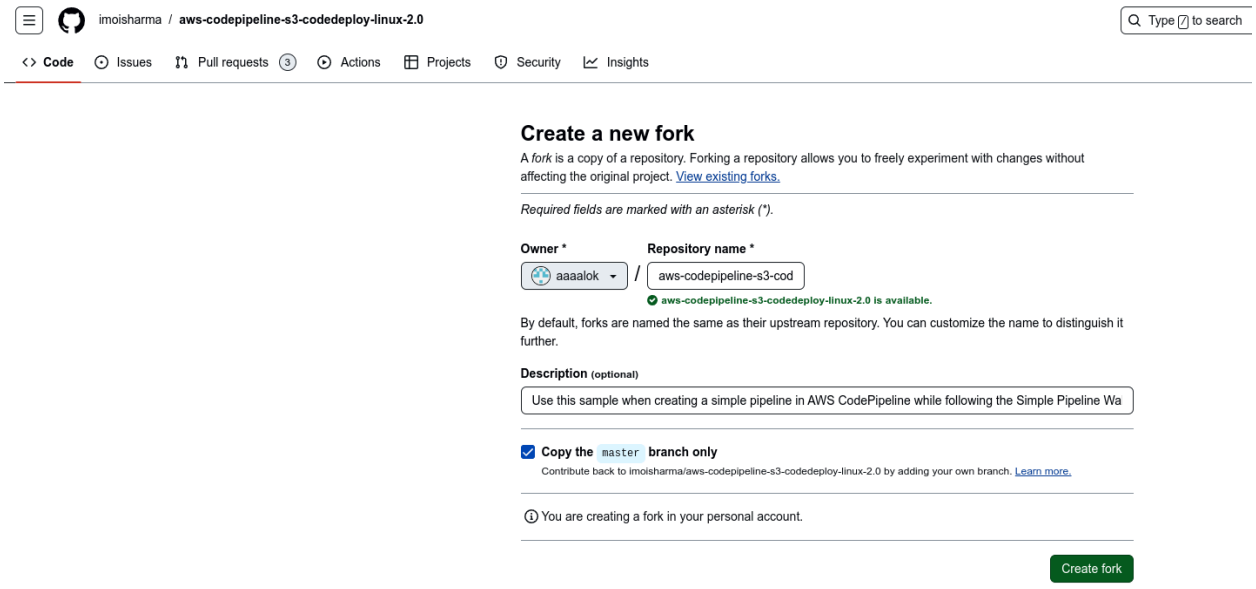
11. Our environment is created successfully



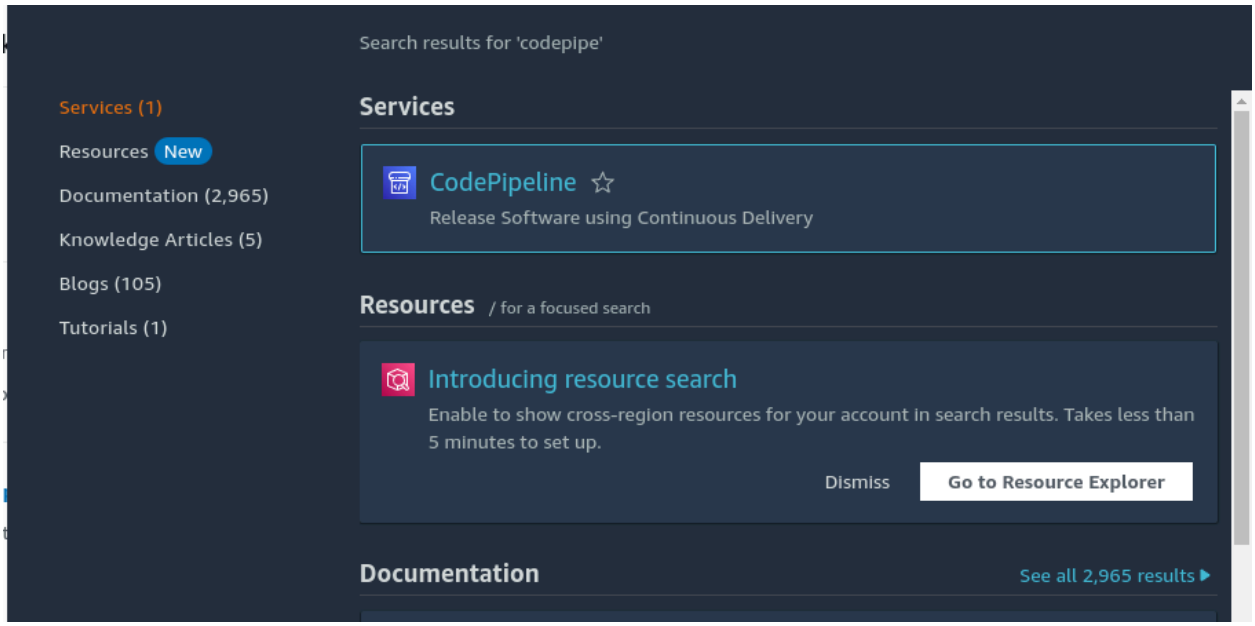
12. Now we would deploy a codepipeline. For below repo in your github

Fork this repository in your github.

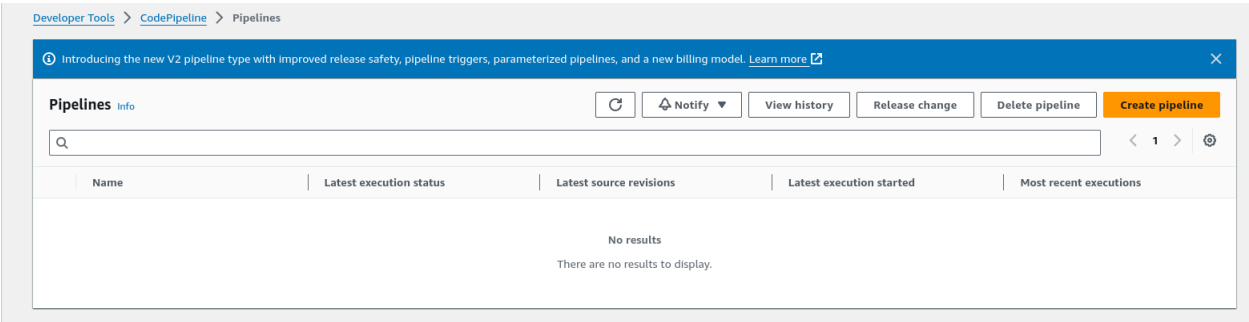




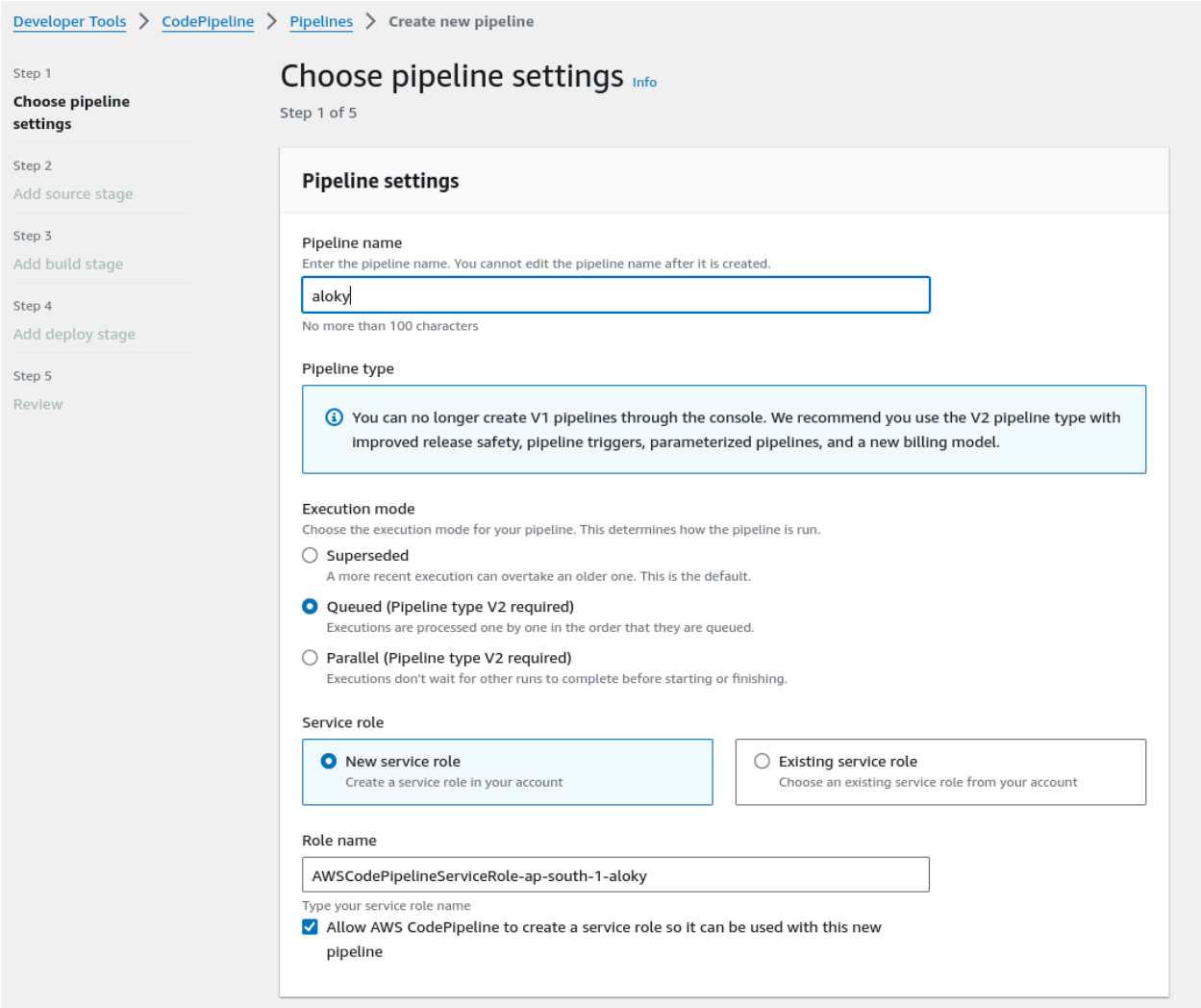
13. After forking, Now we would create a CodePipeline. Goto CodePipeline under services section



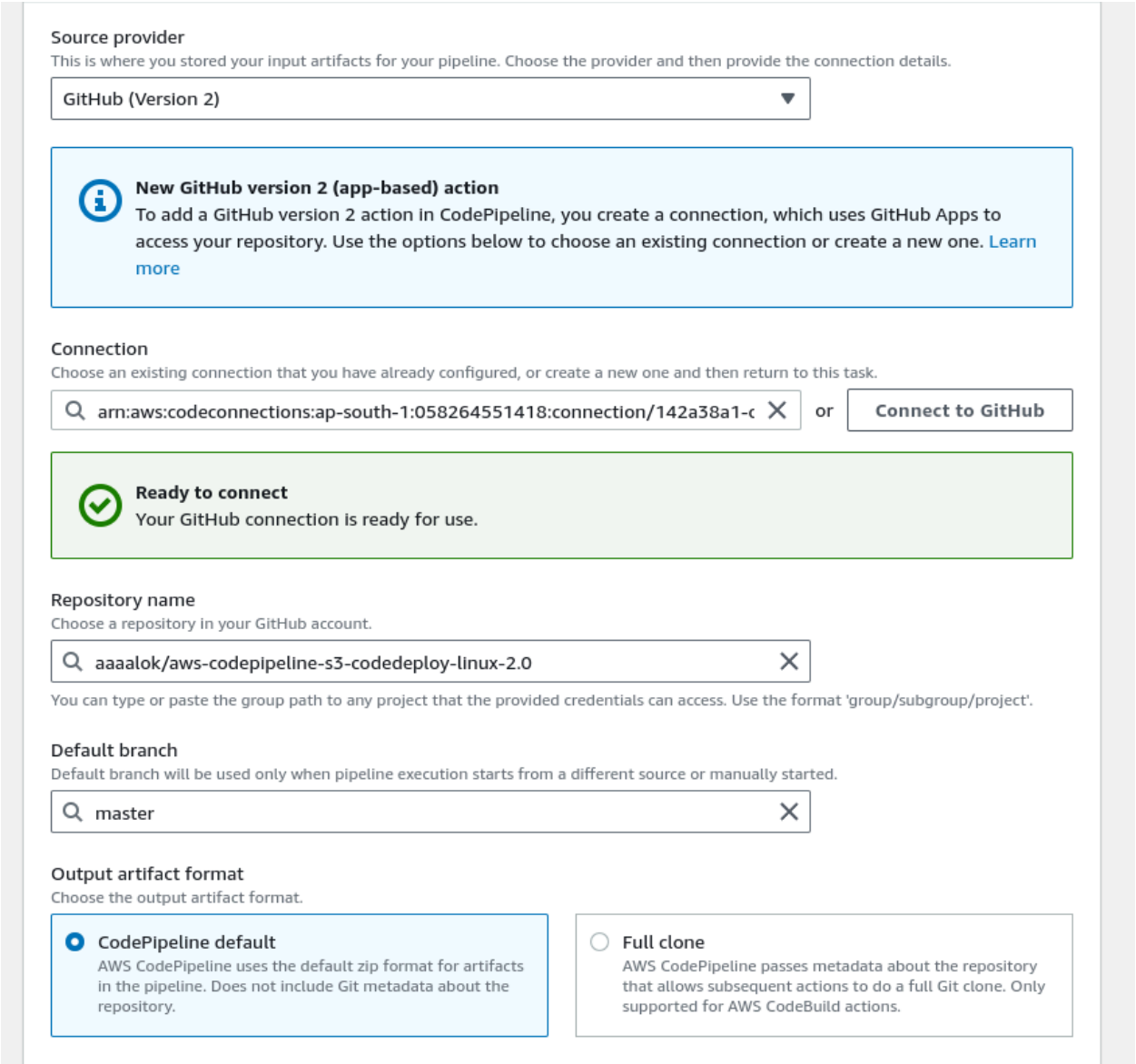
14. Create a new PipeLine



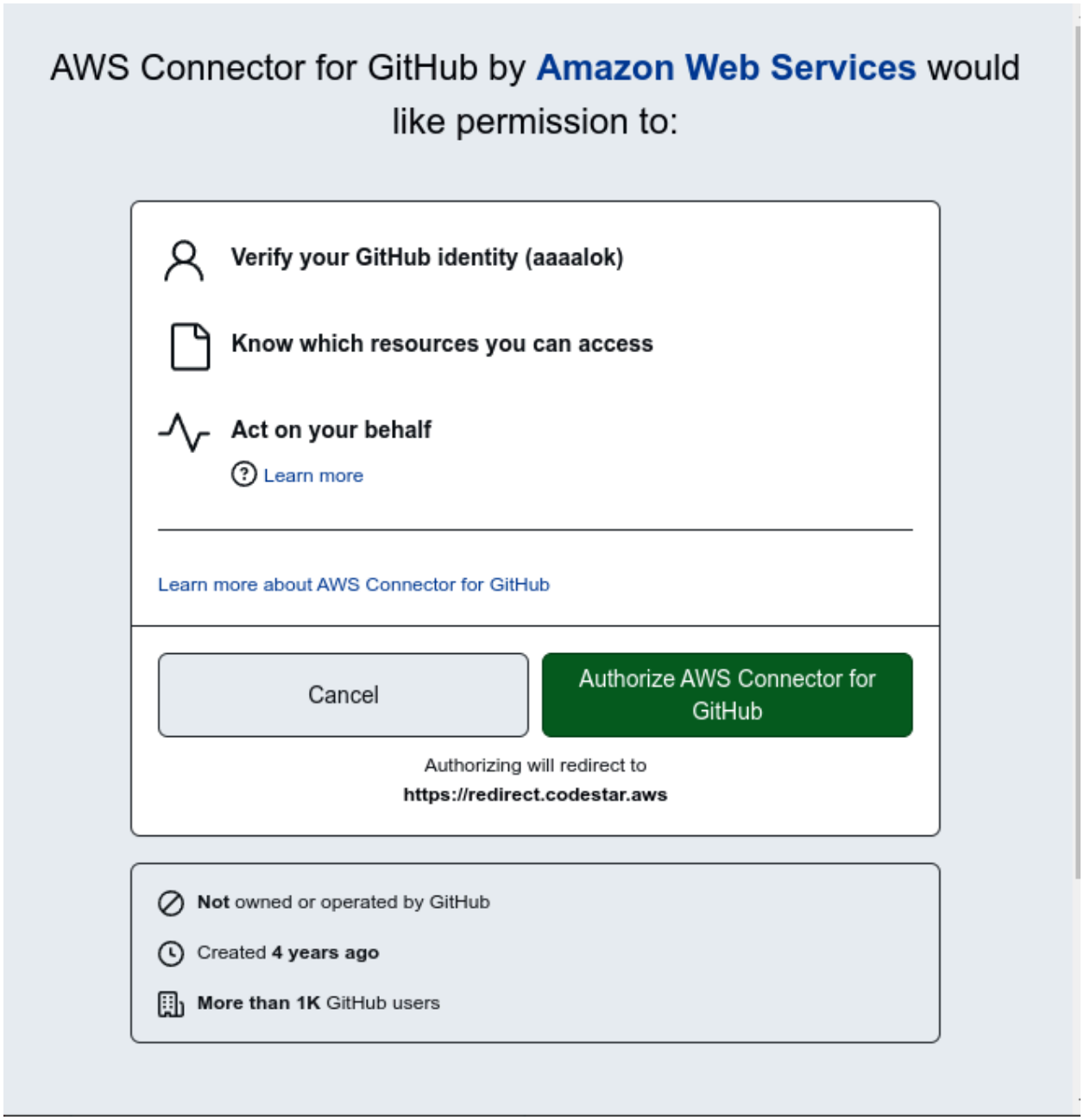
15. Name the pipeline leaving rest settings to its default



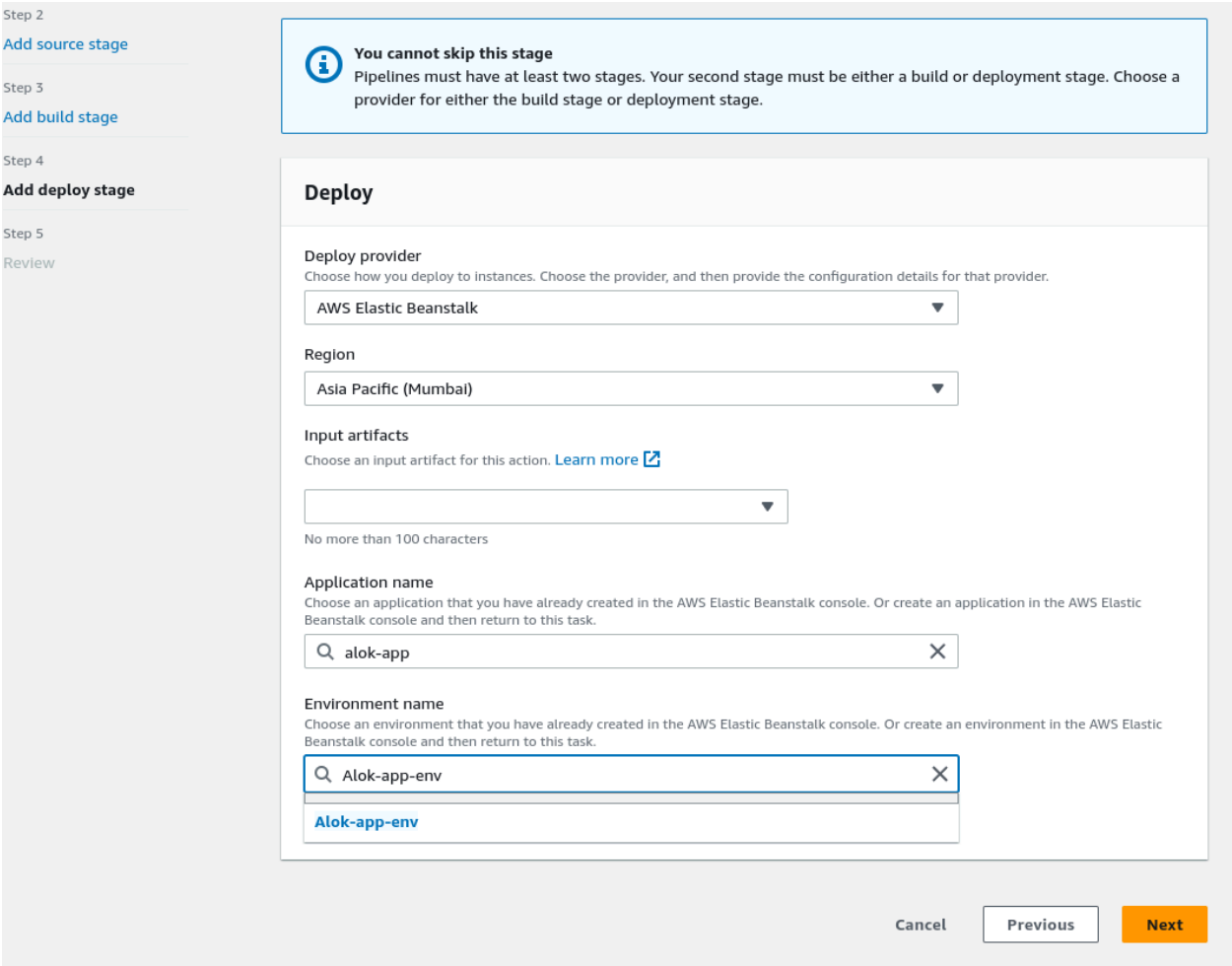
16. Under Source stage, select **Source Provider** as **GitHub (Version 2)**



17. Connect your GitHub account to AWS for it to build and deploy and track changes on repo

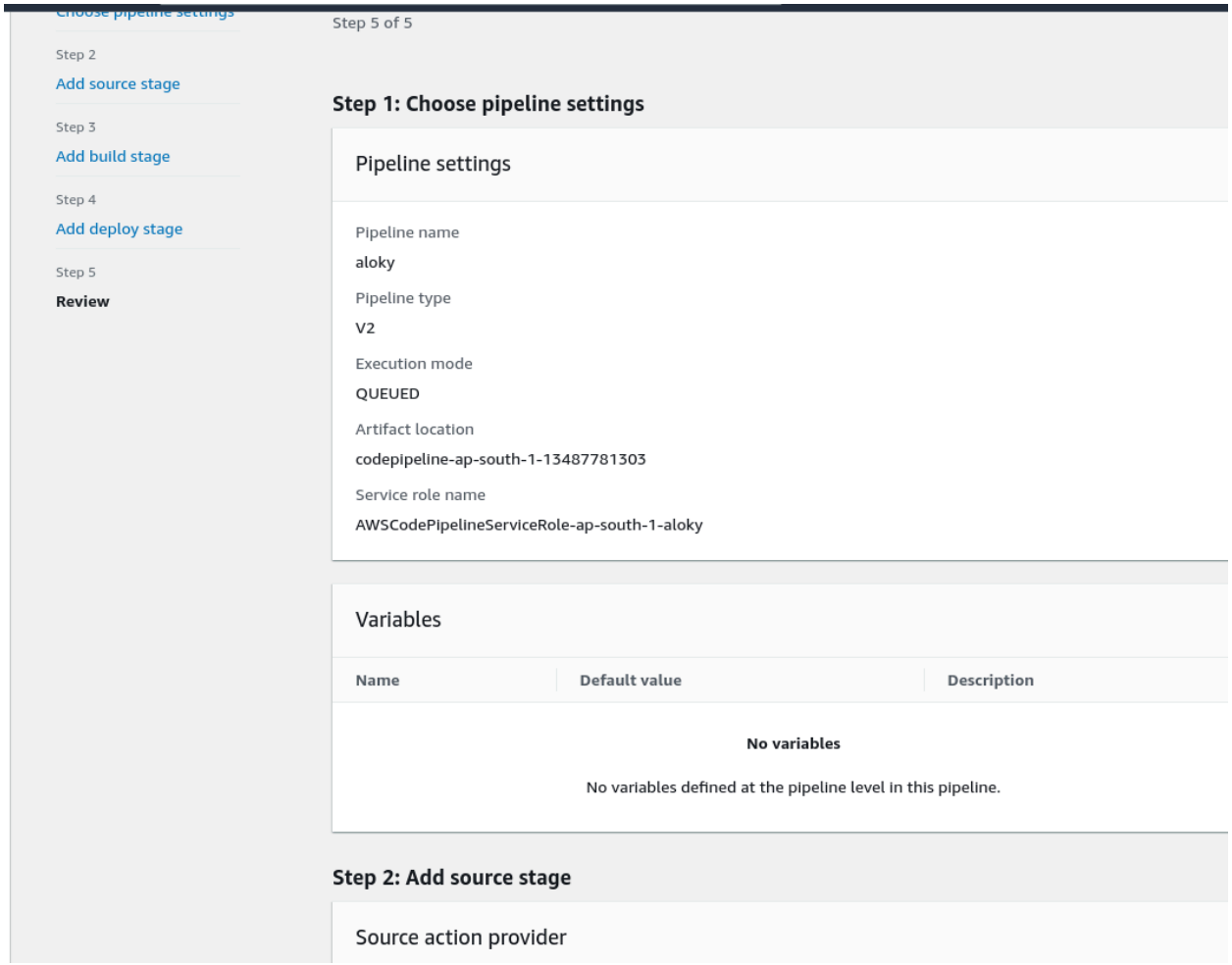


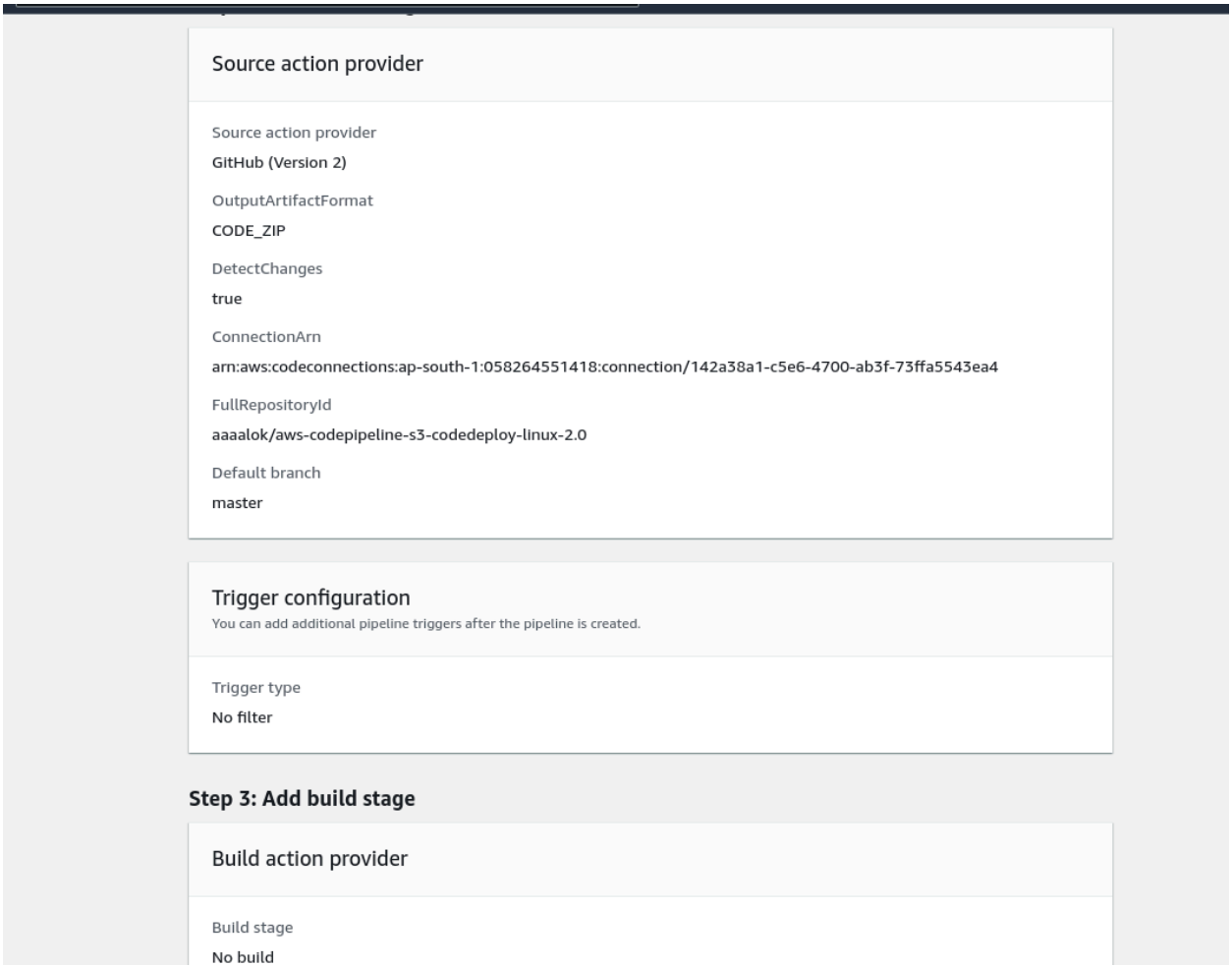
18. As of Now Skip **Build stage** under **Deploy Stage** enter AWS Elastic Beanstalk as Deploy Provider



19. Review the summary of Pipeline created and click on **Create Pipeline**

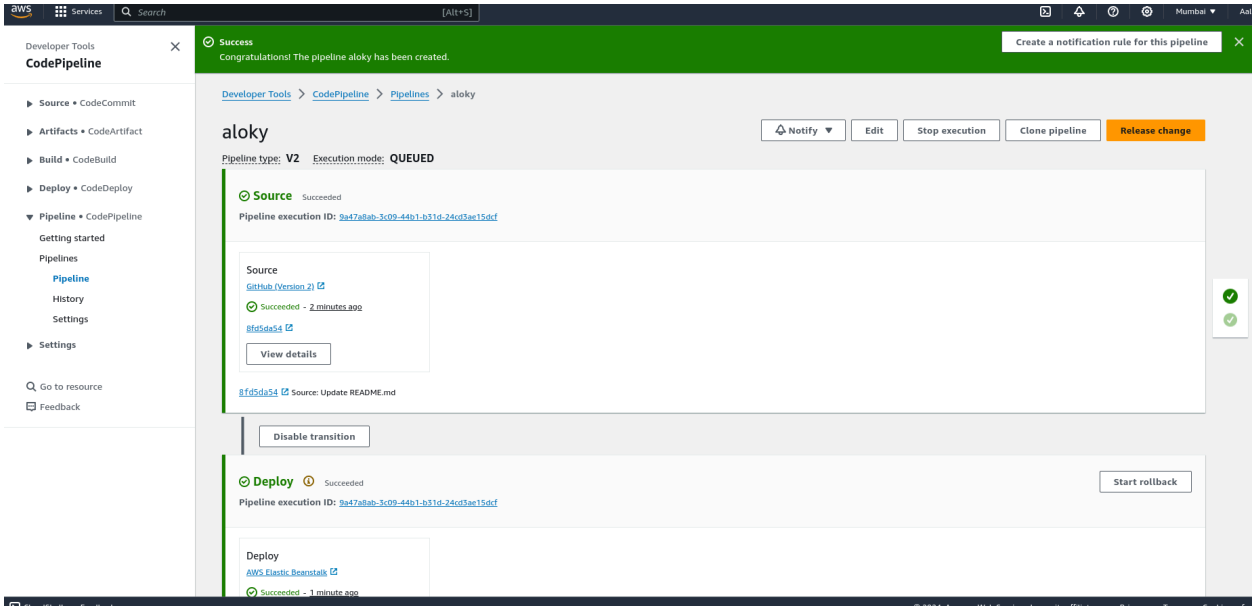
Name: Alok Yadav Div: D15C Roll No.: 59



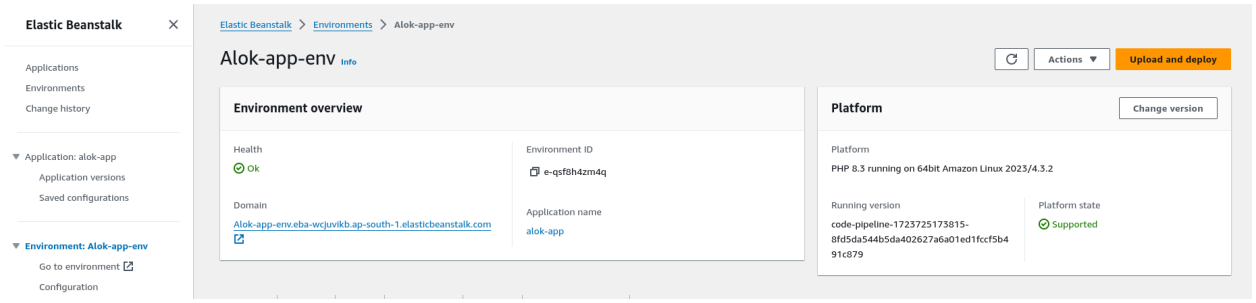


20. Our pipeline will be created and deployed in few minutes

Name: Alok Yadav Div: D15C Roll No.: 59

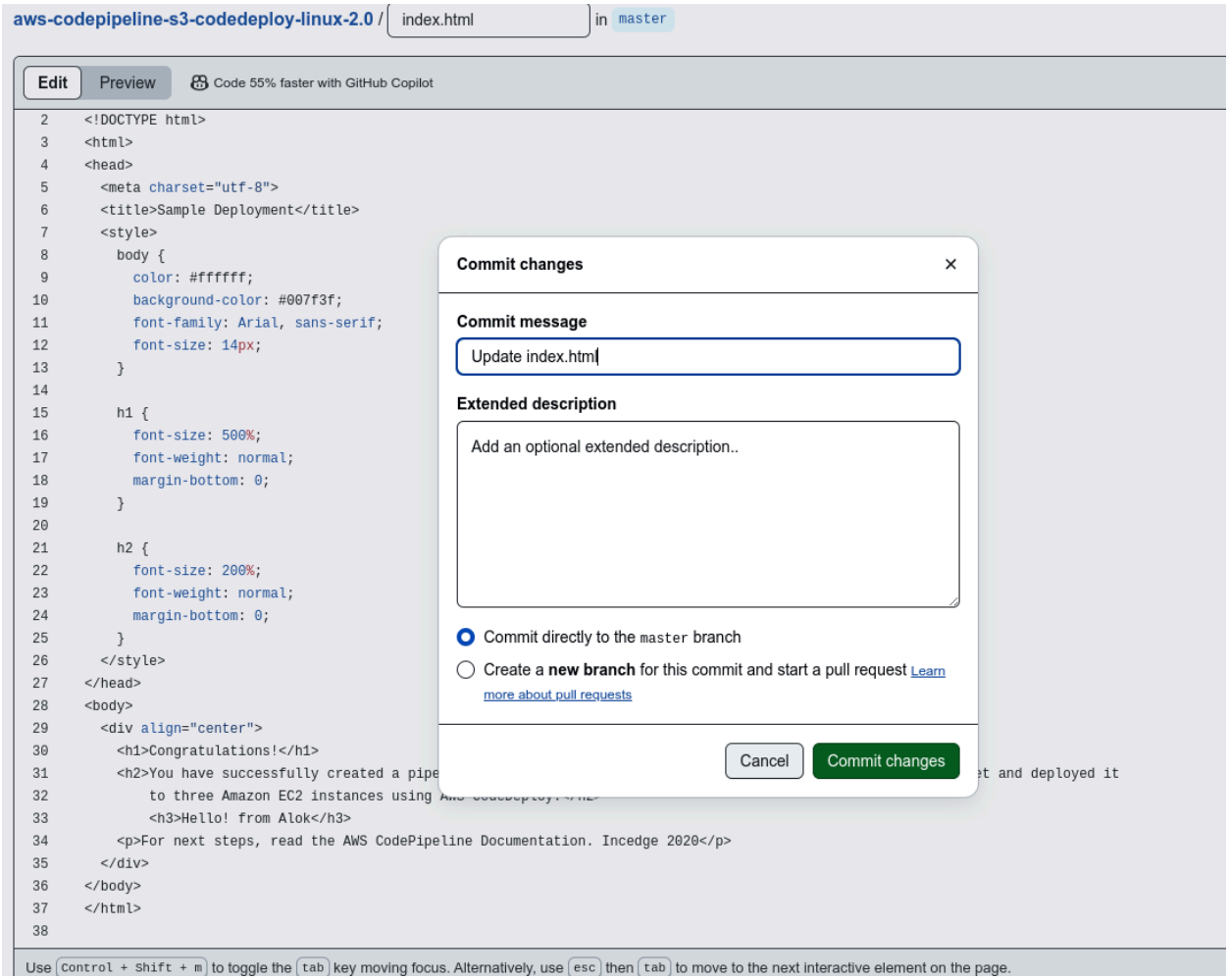


21. Go to Elastic Beanstalk environment that we had created previously and click on the URL given

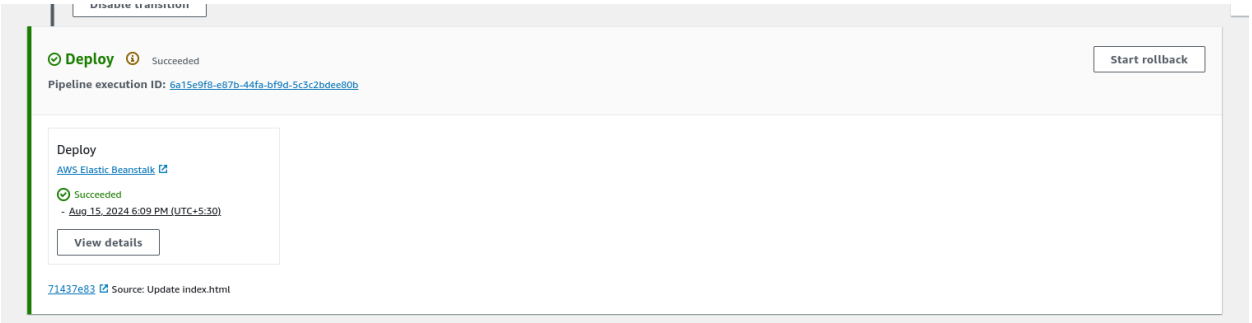


22. Our pipeline is successfully deployed

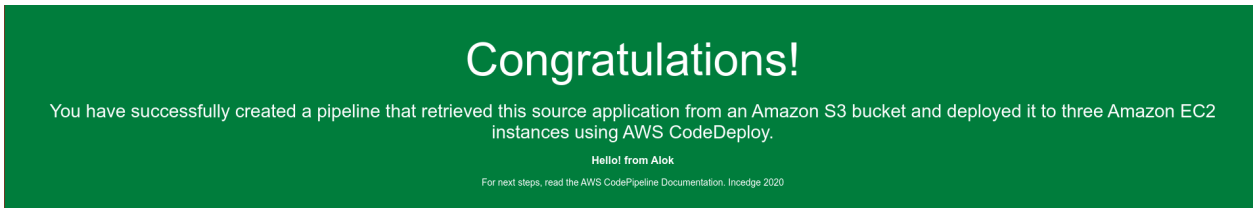
23. Now to test our pipeline, we would do some changes in files in the repository



24. Deploy process is automatically started on detecting change in repository



25. A new text **Hello! from Alok** is shown in the website.



Name: Alok Yadav Div: D15C Roll No.: 59

EXPERIMENT NO. 3

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

1. Create 3 EC-2 instances with all running on Amazon Linux as OS with inbound SSH allowed
   To efficient run kubernetes cluster select instance type of at least t2.medium as kubernetes recommends at least 2 vCPU to run smoothly

Name	Instance ID	Instance state	Instance type
kube-master	i-00aa79ac09d7462c0	Running	t2.medium
kube-worker1	i-0bab86cd3fbfcb40a	Running	t2.medium
kube-worker2	i-00dcfd302ffd80dda	Running	t2.medium

2. SSH into all 3 machines each in separate terminal

```
quantum@machine ~/Downloads ssh -i "ec-2-ubuntu.pem" ec2-user@ec2-3-88-111-183.compute-1.amazonaws.com
The authenticity of host 'ec2-3-88-111-183.compute-1.amazonaws.com (3.88.111.183)' can't be established.
ED25519 key fingerprint is SHA256:pQu+xs9foYbY3de1twjZcVVAOzmGwGv6PHmVruF/Q1s.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-88-111-183.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
```

3. From now on, until mentioned, perform these steps on all 3 machines.

   Install Docker

sudo yum install docker -y

```
[ec2-user@ip-172-31-92-18 ~]$ sudo yum install docker -y
Last metadata expiration check: 0:09:56 ago on Wed Sep 11 15:19:39 2024.
Dependencies resolved.
================================================================================
 Package                                                    Architecture
================================================================================
Installing:
 docker                                                     x86_64
Installing dependencies:
 containerd                                                 x86_64
 iptables-libs                                              x86_64
 iptables-nft                                               x86_64
 libcgroup                                                  x86_64
 libnetfilter_conntrack                                     x86_64
 libnfnetlink                                               x86_64
 libnftnl                                                   x86_64
 pigz                                                       x86_64
 runc                                                       x86_64

Transaction Summary
```

Then, configure cgroup in a daemon.json file by using following commands. This allows kubernetes to manage host more efficiently

- cd /etc/docker
- cat <<EOF | sudo tee /etc/docker/daemon.json
  {
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
  "max-size": "100m"
  },
  "storage-driver": "overlay2"
  }
  EOF

After configuring restart docker service service :

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

```
[ec2-user@ip-172-31-81-63 docker]$ sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker
docker -v

Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
Docker version 25.0.5, build 5dc9bcc
[ec2-user@ip-172-31-81-63 docker]$
```

4. Install Kubernetes on all 3 machines

SELinux needs to be disabled before configuring kubelet to avoid interference with kubernetes api server

- `sudo setenforce 0`
- `sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config`

```
[ec2-user@ip-172-31-81-63 docker]$ sudo setenforce 0
[ec2-user@ip-172-31-81-63 docker]$ sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
```

Add kubernetes repository (paste in terminal)

```
cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
```

Type following commands to install set of kubernetes packages:

- sudo yum update

- `sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes`

```
[ec2-user@ip-172-31-81-63 docker]$ sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
Last metadata expiration check: 0:01:34 ago on Wed Sep 11 15:39:05 2024.
Dependencies resolved.
================================================================================================================================
 Package                                    Architecture                          Version
================================================================================================================================
Installing:
 kubeadm                                    x86_64                                1.30.4-150500.1.1
 kubectl                                    x86_64                                1.30.4-150500.1.1
 kubelet                                    x86_64                                1.30.4-150500.1.1
Installing dependencies:
 conntrack-tools                            x86_64                                1.4.6-2.amzn2023.0.2
 cri-tools                                  x86_64                                1.30.1-150500.1.1
 kubernetes-cni                             x86_64                                1.4.0-150500.1.1
 libnetfilter_cthelper                      x86_64                                1.0.0-21.amzn2023.0.2
 libnetfilter_cttimeout                     x86_64                                1.0.0-19.amzn2023.0.2
 libnetfilter_queue                         x86_64                                1.0.5-2.amzn2023.0.2
 socat                                      x86_64                                1.7.4.2-1.amzn2023.0.2

Transaction Summary
================================================================================================================================
Install  10 Packages
```

After installing Kubernetes, we need to configure internet options to allow bridging.

- sudo swapoff -a
- echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf
- sudo sysctl -p

5. **Perform this ONLY on the Master machine**

   Initialize kubernetes by typing below command

   - sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errors=all

```
[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 172.31.81.63:6443 --token zh5jbb.a6ty3eujzc51d15d \
        --discovery-token-ca-cert-hash sha256:0822f656bf52a17a2b6686c123f811306f41495ca650a0aed9bf6cd2d2f6f8c5
[ec2-user@ip-172-31-81-63 docker]$  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config
[ec2-user@ip-172-31-81-63 docker]$
```

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
```

Copy this join link and save it in clipboard (copy from your output as it different for each instance)

```
kubeadm join 172.31.81.63:6443 --token
zh5jbb.a6ty3eujzc51d15d \
        --discovery-token-ca-cert-hash
sha256:0822f656bf52a17a2b6686c123f811306f41495ca650a0aed9bf6c
d2d2f6f8c5
```

`Then, add a common networking plugin called flammel file as mentioned in the code.`

`kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml`

Check the created pod using this command

- kubectl get pods

6. **Perform this ONLY on the worker machines**

   Paste the below command on all 2 worker machines
   - sudo yum install iproute-tc socat -y (necessary packages required by kubernetes)
   - sudo systemctl enable kubelet
   - sudo systemctl restart kubelet

- kubeadm join 172.31.81.63:6443 --token zh5jbb.a6ty3eujzc51d15d \
  --discovery-token-ca-cert-hash sha256:0822f656bf52a17a2b6686c123f811306f41495ca650a0aed9bf6cd2d2f6f8c5

With the help of command the worker nodes are connected master node and is ready to do task assigned by master node

Now we can see in the master/control node of kubernetes that worker nodes are connected by typing **watch kubectl get nodes** in the **master node instance**

```
Every 2.0s: kubectl get nodes

NAME                              STATUS   ROLES           AGE     VERSION
ip-172-31-81-63.ec2.internal      Ready    control-plane   29m     v1.30.4
ip-172-31-87-137.ec2.internal     Ready    <none>          5m58s   v1.30.4
ip-172-31-92-18.ec2.internal      Ready    <none>          5m53s   v1.30.4
```

Conclusion: We began with installation and configuration of necessary packages required by kubernetes. Some of them were available in the repository of the distribution of linux but some of them were not available so had to add their repository for installation. Even after setting up, the nodes were tainted which was the reason kubernetes api server was crashing, we then fixed it by making them untainted.We even disabled SELINUX to prevent any interference. In this experiment we successfully connected worker nodes with master nodes of kubernetes.

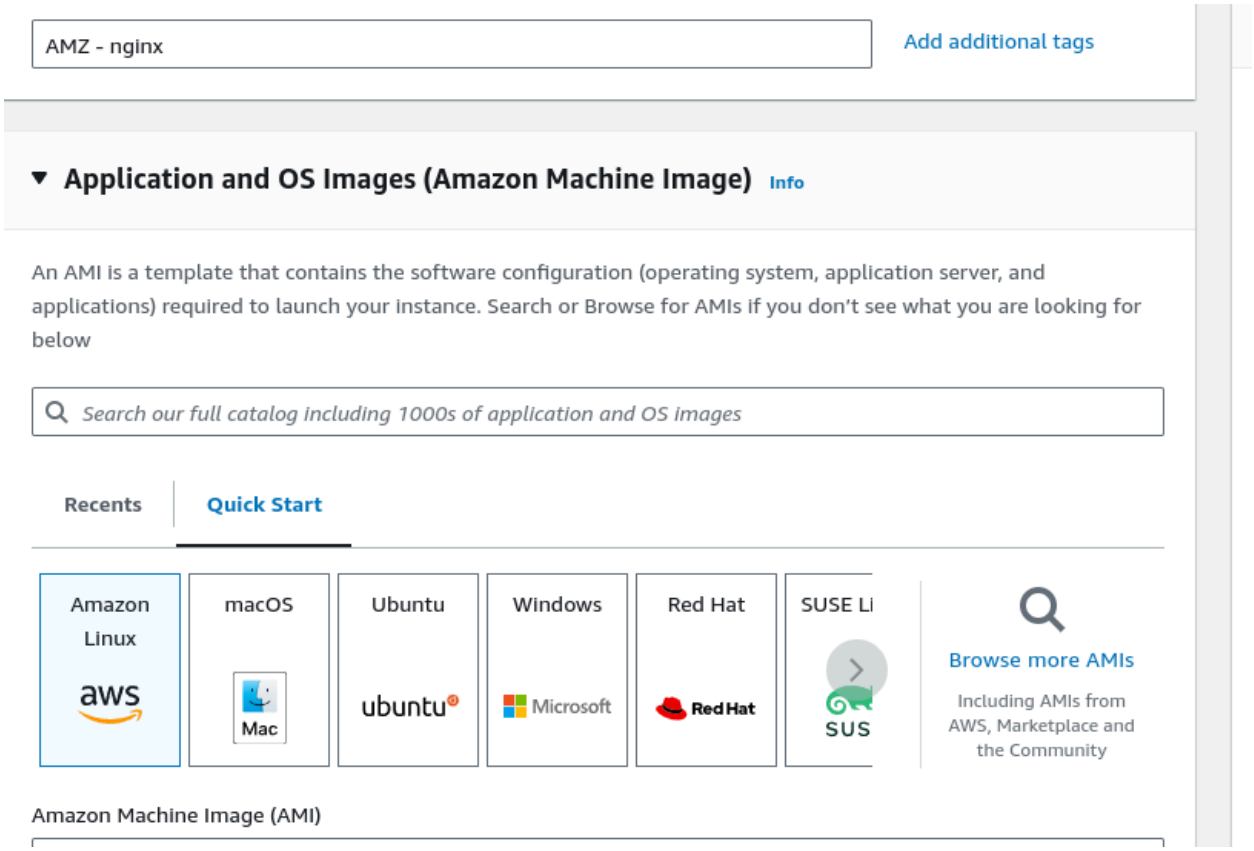
Name: Alok Yadav Div: D15C Roll No.: 59

EXPERIMENT No.4

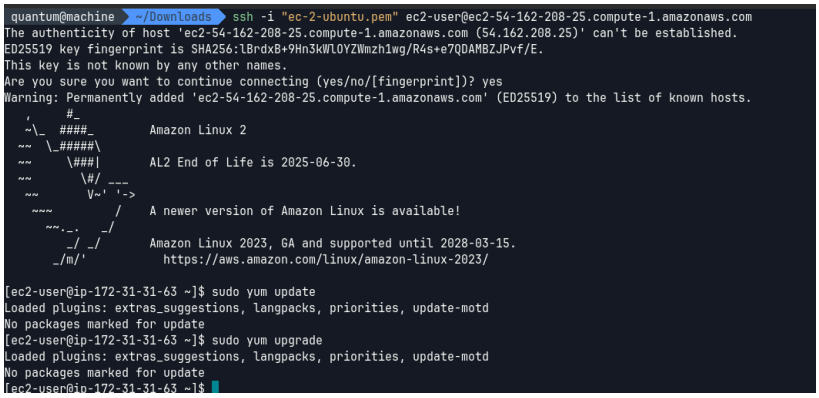
Aim: To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deploy Your First Kubernetes Application.

STEPS:

1. Select Amazon linux as OS image (You can use any but then modify commands accordingly)



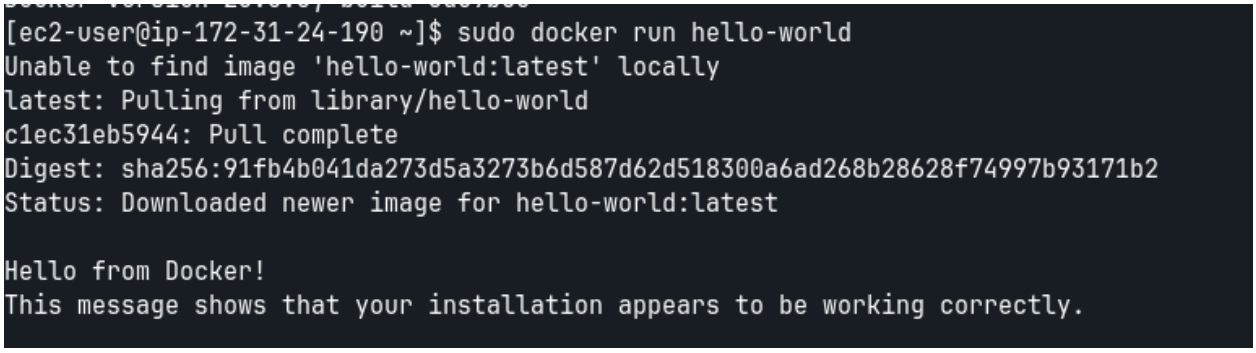
2. Make ssh connection in terminal
   Note: If you have directly made connection through browser then skip this part



3. **Install Docker**

sudo dnf update
sudo dnf install docker
sudo systemctl enable docker
sudo systemctl start docker

To test whether docker is successfully running, use command **sudo docker run hello-world**



Then, configure cgroup in a daemon.json file.This allows kubernetes to manage host more efficiently

cd /etc/docker

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

4. Install Kubernetes

Note: I'm directly installing binary package you may install from package repository of your distribution

Install CNI plugins (required for most pod network):

```
CNI_PLUGINS_VERSION="v1.3.0"
ARCH="amd64"
DEST="/opt/cni/bin"
sudo mkdir -p "$DEST"
curl -L
"https://github.com/containernetworking/plugins/releases/download/${CNI_PLUGINS_V
ERSION}/cni-plugins-linux-${ARCH}-${CNI_PLUGINS_VERSION}.tgz" | sudo tar -C
"$DEST" -xz
```

Define the directory to download command files:

```
DOWNLOAD_DIR="/usr/local/bin"
sudo mkdir -p "$DOWNLOAD_DIR"
```

Optionally install crictl (required for interaction with the Container Runtime Interface (CRI), optional for kubeadm):

```
CRICTL_VERSION="v1.31.0"
ARCH="amd64"
curl -L
"https://github.com/kubernetes-sigs/cri-tools/releases/download/${CRICTL_VERSION}/c
rictl-${CRICTL_VERSION}-linux-${ARCH}.tar.gz" | sudo tar -C $DOWNLOAD_DIR -xz
```

Install `kubeadm`, `kubelet` and add a `kubelet` systemd service:

```
RELEASE="$(curl -sSL https://dl.k8s.io/release/stable.txt)"
ARCH="amd64"
cd $DOWNLOAD_DIR
sudo curl -L --remote-name-all
https://dl.k8s.io/release/${RELEASE}/bin/linux/${ARCH}/{kubeadm,kubelet}
sudo chmod +x {kubeadm,kubelet}


RELEASE_VERSION="v0.16.2"
curl -sSL
"https://raw.githubusercontent.com/kubernetes/release/${RELEASE_VERSION}/cmd/kr
el/templates/latest/kubelet/kubelet.service" | sed "s:/usr/bin:${DOWNLOAD_DIR}:g" |
sudo tee /usr/lib/systemd/system/kubelet.service
sudo mkdir -p /usr/lib/systemd/system/kubelet.service.d
curl -sSL
"https://raw.githubusercontent.com/kubernetes/release/${RELEASE_VERSION}/cmd/kr
el/templates/latest/kubeadm/10-kubeadm.conf" | sed "s:/usr/bin:${DOWNLOAD_DIR}:g"
| sudo tee /usr/lib/systemd/system/kubelet.service.d/10-kubeadm.conf
```

Now we need to install kubectl

```
Set up repository:

cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/repodata/repo
md.xml.key
```

```
EOF

sudo yum install -y kubectl
```

```
ec2-user@ip-172-31-24-190 ~ $ kubectl version
Client Version: v1.31.1
Kustomize Version: v5.4.2
```

```
We have installed successfully installed  kubernetes


After installing Kubernetes, we need to configure internet
options to allow bridging.

sudo swapoff -a
echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a
/etc/sysctl.conf
sudo sysctl -p
```

```
[root@ip-172-31-24-190 bin]# sudo swapoff -a
echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf
sudo sysctl -p
net.bridge.bridge-nf-call-iptables=1
net.bridge.bridge-nf-call-iptables = 1
```

Disable SELINUX

Type **sudo nano /etc/selinux/config** and set the value to **SELINUX=disabled** instead of **SELINUX=permissive**
Save the file by pressing ctrl+o then press enter then press ctrl+x

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
# See also:
# https://docs.fedoraproject.org/en-US/quick-docs/getting-started-with-selinux/#getting-started-with-selinux-selinux-states-and-modes
#
# NOTE: In earlier Fedora kernel builds, SELINUX=disabled would also
# fully disable SELinux during boot. If you need a system with SELinux
# fully disabled instead of SELinux running with no policy loaded, you
# need to pass selinux=0 to the kernel command line. You can use grubby
# to persistently set the bootloader to boot with selinux=0:
#
#    grubby --update-kernel ALL --args selinux=0
#
# To revert back to SELinux enabled:
#
#    grubby --update-kernel ALL --remove-args selinux
#
SELINUX=disabled
# SELINUXTYPE= can take one of these three values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

Then reboot the system using **sudo reboot**
After rebooting we need to make ssh connection with machine after it gets disconnected

Now if we type command **sestatus,** then it show disabled

```
ec2-user@ip-172-31-24-190 ~ $ sestatus
SELinux status:                 disabled
```

5. Initialize the Kubecluster

Install packages socat and iproute-tc and conntrack to avoid prelight errors
sudo dnf install socat iproute-tc conntrack-tools -y

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

```
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 172.31.24.190:6443 --token xsbsq1.6ro11sawnvttbsvu \
        --discovery-token-ca-cert-hash sha256:10d2b67f4f4749b51854065a554c74e6a956e4782d9ab4bb79b8591648b3edef
ec2-user@ip-172-31-24-190 ~ $ kubectl get nodes
```

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

sudo systemctl restart kubelet

Then, 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

```
ec2-user@ip-172-31-24-190 ~ $ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
namespace/kube-flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
```

Now type **kubectl get nodes**

```
The connection to the server 172.31.24.190:6443 was refused - did you specify the right host or port?
ec2-user@ip-172-31-24-190 ~ $ kubectl get nodes
The connection to the server 172.31.24.190:6443 was refused - did you specify the right host or port?
ec2-user@ip-172-31-24-190 ~ $ kubectl get nodes
^[[AError from server (Forbidden): nodes is forbidden: User "kubernetes-admin" cannot list resource "nodes" in
ec2-user@ip-172-31-24-190 ~ $ kubectl get nodes
NAME                           STATUS   ROLES           AGE   VERSION
ip-172-31-24-190.ec2.internal   Ready    control-plane   34m   v1.31.0
ec2-user@ip-172-31-24-190 ~ $ kubectl get nodes
NAME                           STATUS   ROLES           AGE   VERSION
ip-172-31-24-190.ec2.internal   Ready    control-plane   34m   v1.31.0
ec2-user@ip-172-31-24-190 ~ $
```

Note: If any time of get error of connection refused just restart the kubelet service (sudo systemctl restart kubelet)

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

```
ec2-user@ip-172-31-24-190 ~ $ kubectl apply -f https://k8s.io/examples/application/deployment.yaml
deployment.apps/nginx-deployment created
```

Use ‘kubectl get pods’ to verify if the deployment was properly created and the pod is working correctly.

```
ec2-user@ip-172-31-24-190 ~ $ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-mwd8p    0/1     Pending   0          7s
nginx-deployment-d556bf558-zc25s    0/1     Pending   0          7s
```

As we can see our pods are in pending state
On checking logs to we came to know the pods are in tainted state (using command **kubectl describe pod nginx-deployment-d556bf558-mwd8p**)

```
Events:
  Type     Reason            Age   From               Message
  ----     ------            ----  ----               -------
  Warning  FailedScheduling  56s   default-scheduler  0/1 nodes are available: 1 node(s) had untolerated taint {node-role.kubernetes.io/control-plane: }. preemption: 0/1 nodes are available:
```

To make pods untainted
Type kubectl get nodes to see name of node

```
ec2-user@ip-172-31-24-190 ~ $ kubectl get nodes
NAME                              STATUS   ROLES           AGE   VERSION
ip-172-31-24-190.ec2.internal     Ready    control-plane   43m   v1.31.0
ec2-user@ip-172-31-24-190 ~ $
```

Copy the name of the node (ip-172-31-24-190.ec2.internal)

Then type command **kubectl taint nodes <NODE_NAME> - -all**

In my case **kubectl taint nodes ip-172-31-24-190.ec2.internal node-role.kubernetes.io/control-plane-**

```
ec2-user@ip-172-31-24-190 ~ $ kubectl taint nodes ip-172-31-24-190.ec2.internal node-role.kubernetes.io/control-plane-
node/ip-172-31-24-190.ec2.internal untainted
```

After executing above command, check again status of pods if still pending then restart kubelet wait for 1-2 minutes and check again

```
ec2-user@ip-172-31-24-190 ~ $ kubectl get pods
NAME                                READY   STATUS    RESTARTS      AGE
nginx-deployment-d556bf558-mwd8p    1/1     Running   2 (73s ago)   12m
nginx-deployment-d556bf558-zc25s    1/1     Running   2 (73s ago)   12m
```

As we can see our pods are running

Lastly, port forward the deployment to your localhost so that you can view it.

kubectl port-forward <POD_NAME> 8080:80
In my case : **kubectl port-forward nginx-deployment-d556bf558-mwd8p 8080:80**

Note: if you are getting connection refused error then restart kubelet

```
ec2-user@ip-172-31-24-190 ~ $  kubectl port-forward  nginx-deployment-d556bf558-mwd8p 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
```

As port forwarding is active so we cannot type other commands.
Open new terminal window and make ssh connection to same machine OR we can open instance of same machine in new browser tab
And type command **curl --head http://127.0.0.1:8080**

```
ec2-user@ip-172-31-24-190 ~ $ curl --head http://127.0.0.1:8080
HTTP/1.1 200 OK
Server: nginx/1.14.2
Date: Sat, 14 Sep 2024 06:54:21 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

ec2-user@ip-172-31-24-190 ~ $
 3  1:ec2-user@ip-172-31-24-190:~#-  2:ec2-user@ip-172-31-24-190:~*  3:~/Downloads
```

Response status 200 (OK) indicates that our nginx server is running successfully on kubernetes

Conclusion: We began with installation and setup of docker and kubernetes. The kubernetes api server had some issues but were resolved after restarting kubelet service. The pods created were not running because nodes were tainted so we had to make them untainted. After solving all errors the nginx server pods were deployed successfully and can be accessed with the forwarded port. The nginx server can be accessed on different terminals or by making port forward process as background process by appending & after the command.

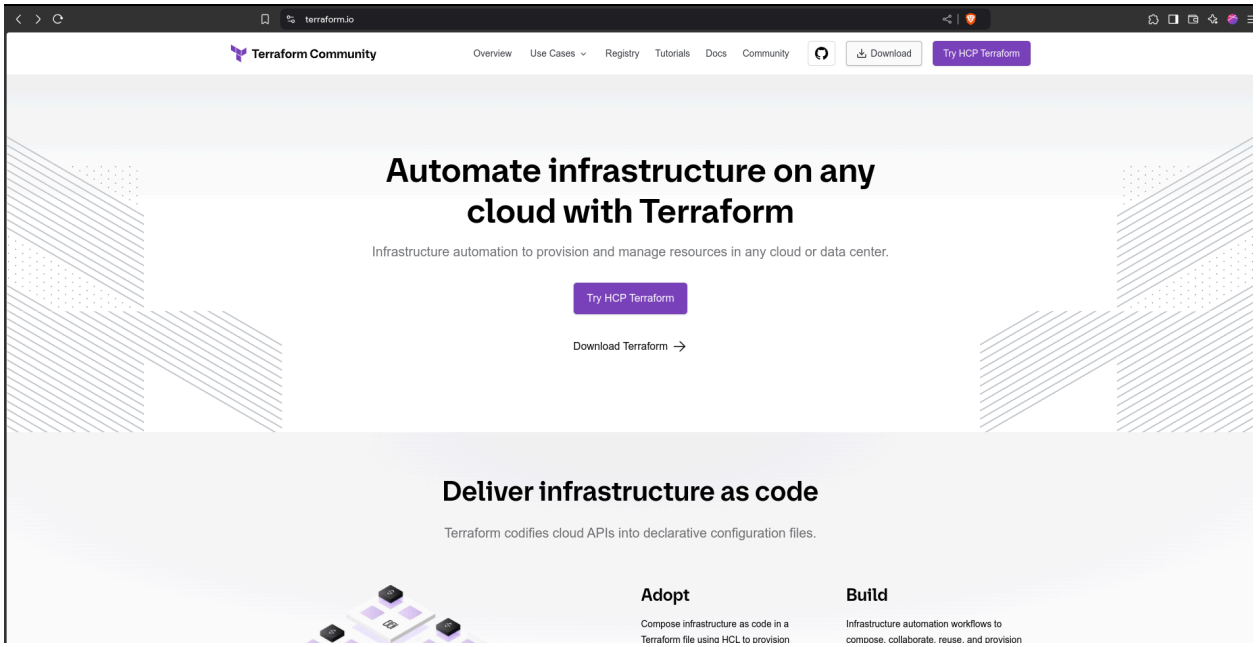
EXPERIMENT NO. 5

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

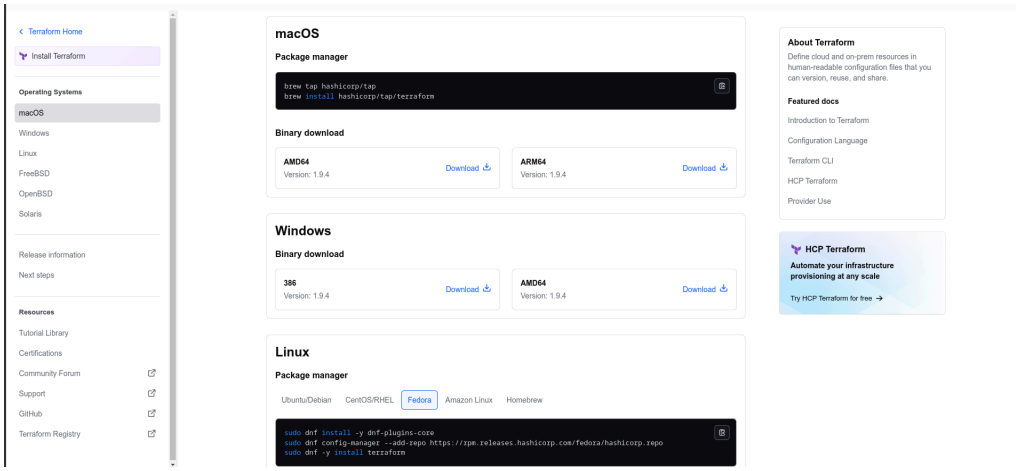
Name: Alok Yadav Div: D15C Roll No.: 59

Steps:

1. Go to the official website of **Terraform (https://www.terraform.io/).** Click on Downloads located at top right corner of the page



2. Follow procedure according to your operating system (I'm installing on Linux)



3. Update your repository cache

```
quantum@machine ~ sudo dnf5 update --refresh
Updating and loading repositories:
```

4. Install **Plugins Core**

```
quantum@machine ~ sudo dnf install -y dnf-plugins-core

Fedora 40 - x86_64 - Updates
Package dnf-plugins-core-4.8.0-1.fc40.noarch is already installed.
Dependencies resolved.
Nothing to do.
Complete!
```

5. Add official package repository to your system

```
quantum@machine ~ sudo dnf config-manager --add-repo https://rpm.releases.hashicorp.com/fedora/hashicorp.repo
Adding repo from: https://rpm.releases.hashicorp.com/fedora/hashicorp.repo
```

6. Install it by typing **sudo dnf -y install terraform** in the terminal

```
✘ quantum@machine ~ sudo dnf5  install terraform
```

```
Repositories loaded.
Package                                    Arch          Version                Repository                Size
Installing:
 terraform                                 x86_64        1.9.4-1                hashicorp                 84.9 MiB

Transaction Summary:
 Installing:       1 packages

Total size of inbound packages is 27 MiB. Need to download 27 MiB.
After this operation 85 MiB will be used (install 85 MiB, remove 0 B).
Is this ok [y/N]: y
[1/1] terraform-0:1.9.4-1.x86_64                                                  100% |   1.8 MiB/s |  26.7 MiB |  00m14s
----------------------------------------------------------------------------------------------------------------------------
[1/1] Total                                                                       100% |   1.8 MiB/s |  26.7 MiB |  00m14s
[1/2] https://rpm.releases.hashicorp.com/gpg                                      100% |   1.7 KiB/s |   3.9 KiB |  00m02s
----------------------------------------------------------------------------------------------------------------------------
[2/2] Total                                                                       100% |   1.8 MiB/s |  26.7 MiB |  00m14s
Importing PGP key 0xA621E701:
 Userid     : "HashiCorp Security (HashiCorp Package Signing) <security+packaging@hashicorp.com>"
 Fingerprint: 798AEC654E5C15428C8E42EEAA16FCBCA621E701
 From       : https://rpm.releases.hashicorp.com/gpg
Is this ok [y/N]: y
The key was successfully imported.
[1/3] Verify package files                                                        100% |  50.0   B/s |   1.0   B |  00m00s
[2/3] Prepare transaction                                                         100% |   1.0   B/s |   1.0   B |  00m01s
[3/3] Installing terraform-0:1.9.4-1.x86_64                                       100% | 108.3 MiB/s |  84.9 MiB |  00m01s
```

7. Verify installation by checking version of Terraform installed

```
quantum@machine ~ terraform -v
Terraform v1.9.4
on linux_amd64
```

EXPERIMENT NO. 6 (A)

Name: Alok Yadav Div: D15C Roll No.: 59

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

Steps:

1. Check docker installation of looking at it's help page

```
quantum@machine ~ docker -h
Flag shorthand -h has been deprecated, use --help

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

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

Management Commands:
  builder     Manage builds
  buildx*     Docker Buildx
  checkpoint  Manage checkpoints
  compose*    Docker Compose
  container   Manage containers
  context     Manage contexts
  dev*        Docker Dev Environments
  extension*  Manages Docker extensions
  feedback*   Provide feedback, right in your terminal!
  image       Manage images
  init*       Creates Docker-related starter files for your project
  manifest    Manage Docker image manifests and manifest lists
  network     Manage networks
  plugin      Manage plugins
  sbom*       View the packaged-based Software Bill Of Materials (SBOM) for an image
  scan*       Docker Scan
  scout*      Docker Scout
  system      Manage Docker
  trust       Manage trust on Docker images
  volume      Manage volumes
```

```
quantum@machine ~ docker -v
Docker version 27.1.2, build d01f264
```

2. Create a new folder named **"Terraform Scripts"**, inside it create a new folder **docker** and create a new file named **docker.tf** with following contents inside it.

```
  GNU nano 7.2                                    docker.tf
terraform {
  required_providers {
    docker = {
      source  = "kreuzwerker/docker"
      version = "2.21.0"
    }
  }
}

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

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

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

3. To see list of providers for current configuration file use command **terraform providers**

```
quantum@machine ~/Downloads/advdevops/TerraformScripts/docker terraform providers

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

4. Execute command **terraform init** in the current directory

```
quantum@machine ~/Downloads/advdevops/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.
```

5. Use command **terraform validate** to check for validation and syntax errors of config file

```
quantum@machine ~/Downloads/advdevops/TerraformScripts/docker terraform validate
Success! The configuration is valid.
```

5. Execute command **terraform plan** to see the changes that will be made

```
quantum@machine ~/Downloads/advdevops/TerraformScripts/docker sudo terraform plan

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

Terraform will perform the following actions:

  # docker_container.foo will be created
  + resource "docker_container" "foo" {
      + attach           = false
      + bridge           = (known after apply)
      + command          = (known after apply)
      + container_logs   = (known after apply)
      + entrypoint       = (known after apply)
      + env              = (known after apply)
      + exit_code        = (known after apply)
      + gateway          = (known after apply)
      + hostname         = (known after apply)
      + id               = (known after apply)
      + image            = (known after apply)
      + init             = (known after apply)
      + ip_address       = (known after apply)
      + ip_prefix_length = (known after apply)
      + ipc_mode         = (known after apply)
      + log_driver       = (known after apply)
      + logs             = false
      + must_run         = true
      + name             = "foo"
      + network_data     = (known after apply)
      + read_only        = false
      + remove_volumes   = true
      + restart          = "no"
      + rm               = false
      + runtime          = (known after apply)
      + security_opts    = (known after apply)
      + shm_size         = (known after apply)
      + start            = true
      + stdin_open       = false
      + stop_signal      = (known after apply)
      + stop_timeout     = (known after apply)
      + tty              = false

      + healthcheck (known after apply)
```

```
      + labels (known after apply)
    }

  # docker_image.ubuntu will be created
  + resource "docker_image" "ubuntu" {
      + id          = (known after apply)
      + image_id    = (known after apply)
      + latest      = (known after apply)
      + name        = "ubuntu:latest"
      + output      = (known after apply)
      + repo_digest = (known after apply)
    }

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

6. Check list of running docker containers using command **docker ps**

```
✘ quantum@machine ~/Downloads/advdevops/TerraformScripts/docker sudo docker ps

CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

As we can see there are no active containers running

7. Execute **terraform apply** to apply configuration, which will automatically create and run the Ubuntu Linux container based on our configuration.

```
✘ quantum@machine ~/Downloads/advdevops/TerraformScripts/docker sudo terraform apply
docker_image.ubuntu: Refreshing state... [id=sha256:17c0145030df106e60e5d99149d69810db23b869ff0d3c9d23627
a5a7bbb6b3ubuntu:latest]

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          = [
          + "/bin/bash",
          + "-c",
          + "while true; do sleep 3600; done",
        ]
      + container_logs   = (known after apply)
      + entrypoint       = (known after apply)
      + env              = (known after apply)
      + exit_code        = (known after apply)
      + gateway          = (known after apply)
      + hostname         = (known after apply)
      + id               = (known after apply)
      + image            = "sha256:17c0145030df106e60e5d99149d69810db23b869ff0d3c9d236279a5a7bbb6b3"
      + init             = (known after apply)
      + ip_address       = (known after apply)
      + ip_prefix_length = (known after apply)
      + ipc_mode         = (known after apply)
```

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

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

  Enter a value: yes

docker_container.foo: Creating...
docker_container.foo: Creation complete after 1s [id=d912cf0a2579f9b8c958d2d33426ed11e8251a553f85a0c08022d19ddf9eeecd]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

8. After executing **terraform apply**

```
quantum@machine ~/Downloads/advdevops/TerraformScripts/docker sudo docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS             PORTS     NAMES
d912cf0a2579   17c0145030df   "/bin/bash -c 'while…"   2 minutes ago   Up About a minute            foo
```

After execution, a new docker image and container will be created

9. Execute **terraform destroy** to delete the configuration, which will automatically delete the Container.

```
  # docker_image.ubuntu will be destroyed
  - resource "docker_image" "ubuntu" {
      - id          = "sha256:35a88802559dd2077e584394471ddaa1a2c5bfd16893b829ea57619301eb3908ubuntu:latest
" -> null
      - image_id    = "sha256:35a88802559dd2077e584394471ddaa1a2c5bfd16893b829ea57619301eb3908" -> null
      - latest      = "sha256:35a88802559dd2077e584394471ddaa1a2c5bfd16893b829ea57619301eb3908" -> null
      - name        = "ubuntu:latest" -> null
      - repo_digest = "ubuntu@sha256:2e863c44b718727c860746568e1d54afd13b2fa71b160f5cd9058fc436217b30" -> n
ull
    }

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=8329fd298f0ccf122a391d74108a2809c40b5fbe383ffc31ee4fb6aeb6f0e3c7]
docker_container.foo: Destruction complete after 0s
docker_image.ubuntu: Destroying... [id=sha256:35a88802559dd2077e584394471ddaa1a2c5bfd16893b829ea57619301eb3
908ubuntu:latest]
docker_image.ubuntu: Destruction complete after 0s

Destroy complete! Resources: 2 destroyed.
```

10. Check whether docker image and container is removed or not

```
quantum@machine ~/Downloads/advdevops/TerraformScripts/docker sudo docker images
REPOSITORY     TAG       IMAGE ID       CREATED         SIZE
alpine         latest    05455a08881e   6 months ago    7.38MB
busybox        latest    3f57d9401f8d   7 months ago    4.26MB
sxcurity/gau   latest    5a5fc3cf7aa4   9 months ago    23.3MB
hello-world    latest    d2c94e258dcb   15 months ago   13.3kB
quantum@machine ~/Downloads/advdevops/TerraformScripts/docker sudo docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

As we can see both containers and images are removed

Name: Alok Yadav Div: D15C Roll No.: 59

EXPERIMENT NO. 6 (B)

AIM: Creating S3 Bucket using terraform

Prerequisite:

1) Any text editor to write and save scripts
2) Must have an AWS Access Key ID and Secret Access Key

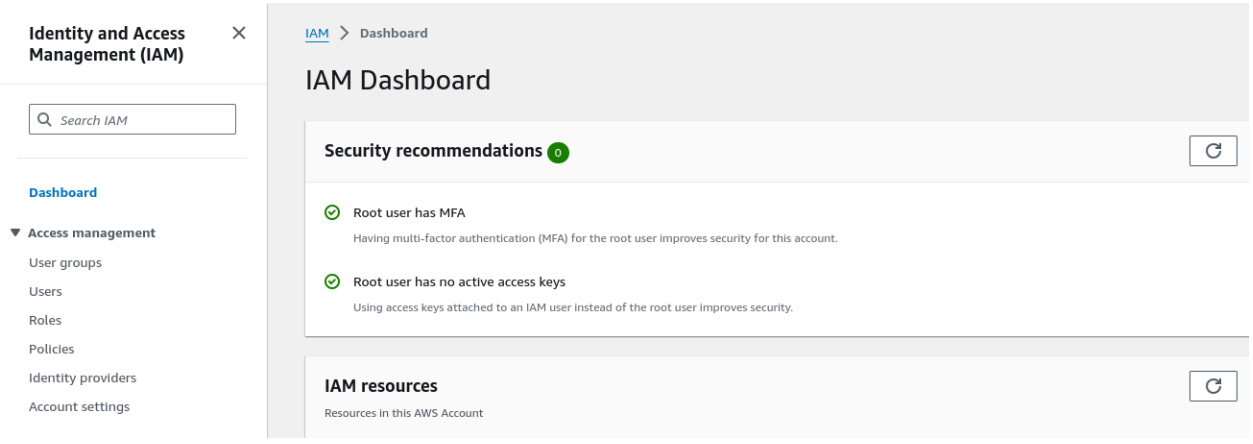
Step 1: Write a Terraform Script in Atom for creating S3 Bucket on Amazon AWS

```
resource "aws_s3_bucket" "aloky" {
  bucket = "my-bj-terraform-test-bucket-aloky"

  tags = {
    Name        = "My bucket"
    Environment = "Dev"
  }
}
```

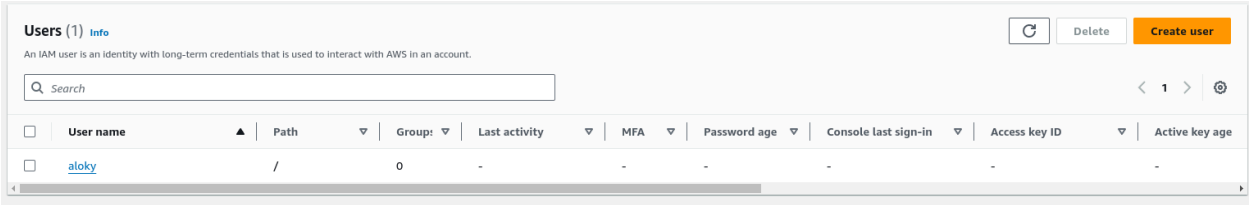
Ensure your bucket name is globally unique

Step 2: Get secret key and access key from AWS account

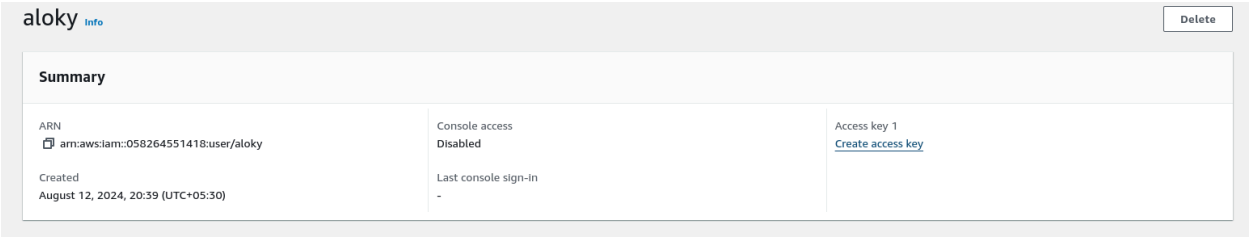
1. Go to IAM dashboard



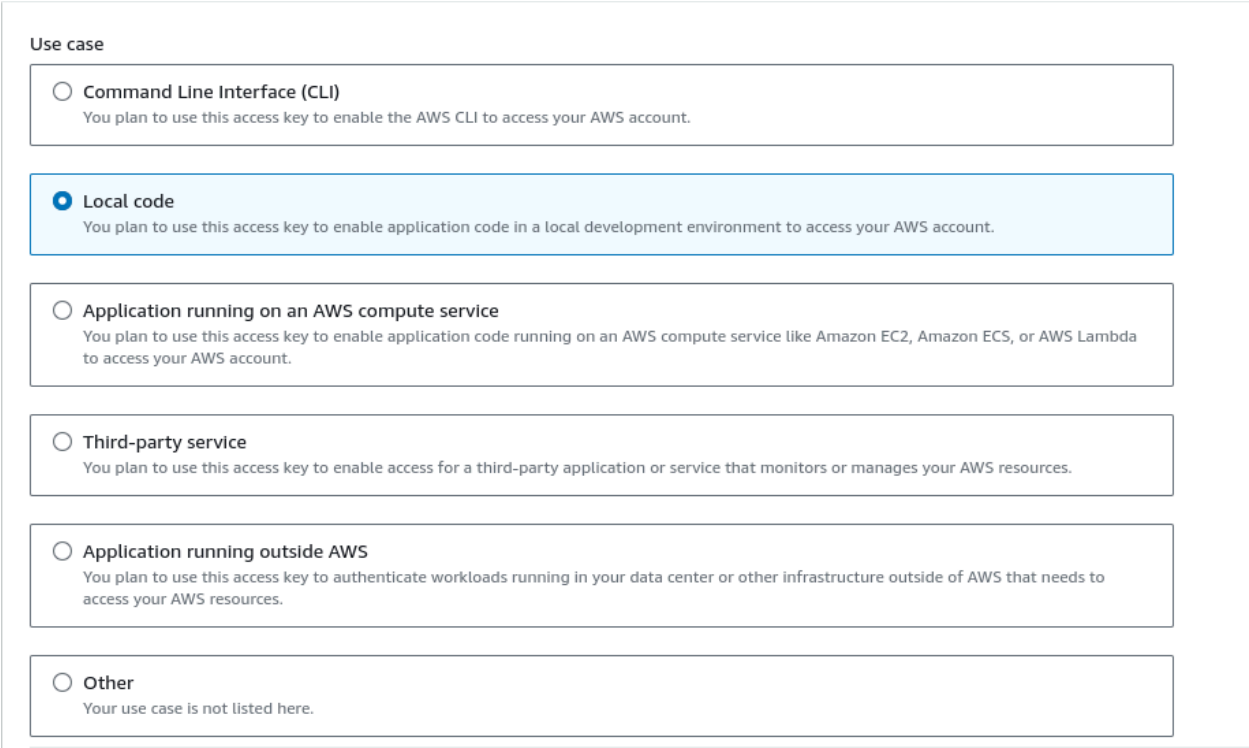
2. Go to Users and select an existing user if it exists or create a new one. I'm selecting existing one as i had already created before



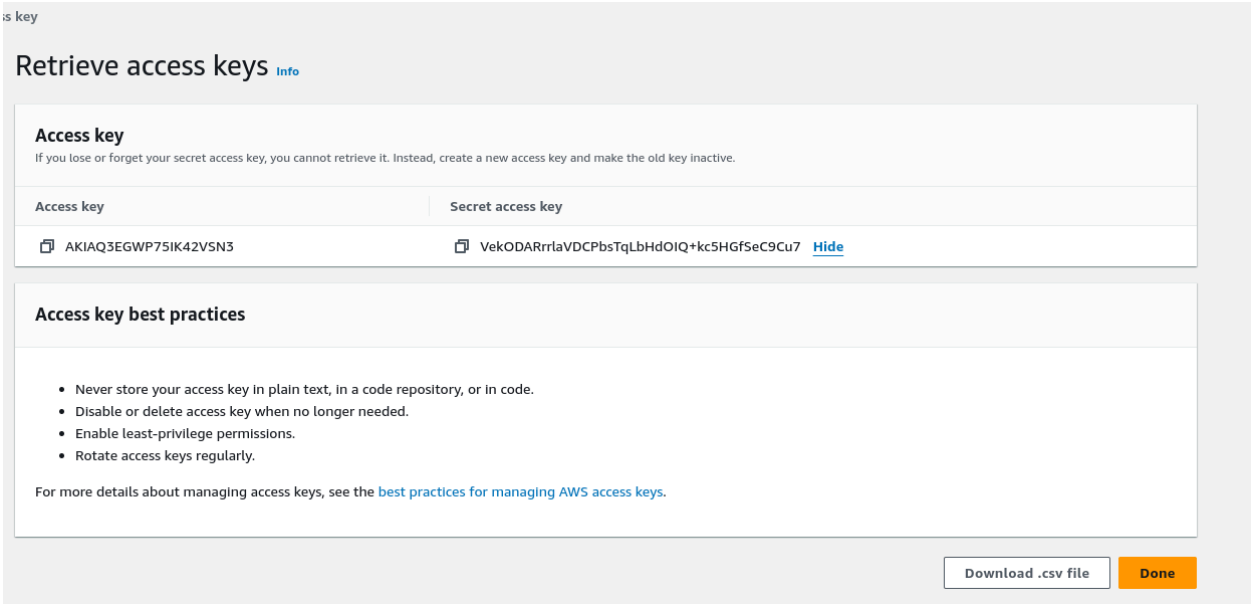
3. On the user dashboard, click on **create access key**



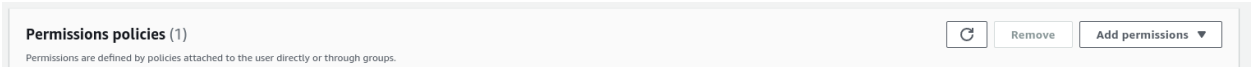
4. Select **Use case** as **Local Code** then Create access key



5. After all above steps, access key is generated

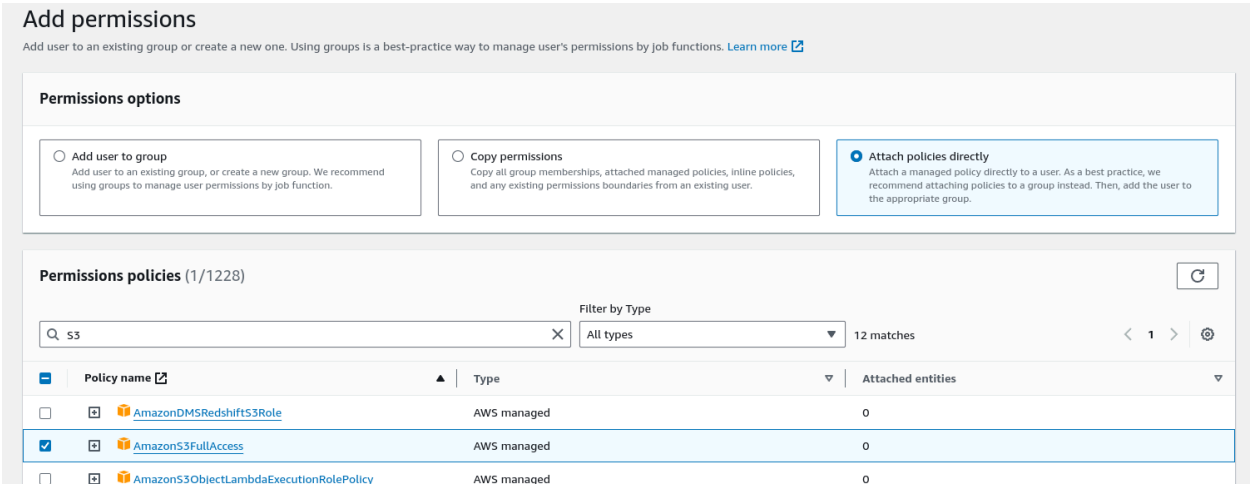


6. Ensure newly created user has necessary permissions to edit and create S3 buckets to do so, click on **Add permissions**



7. Select permission option as **Attach policies directly** and select **AmazonS3FullAccess** as permission policy from the list

\



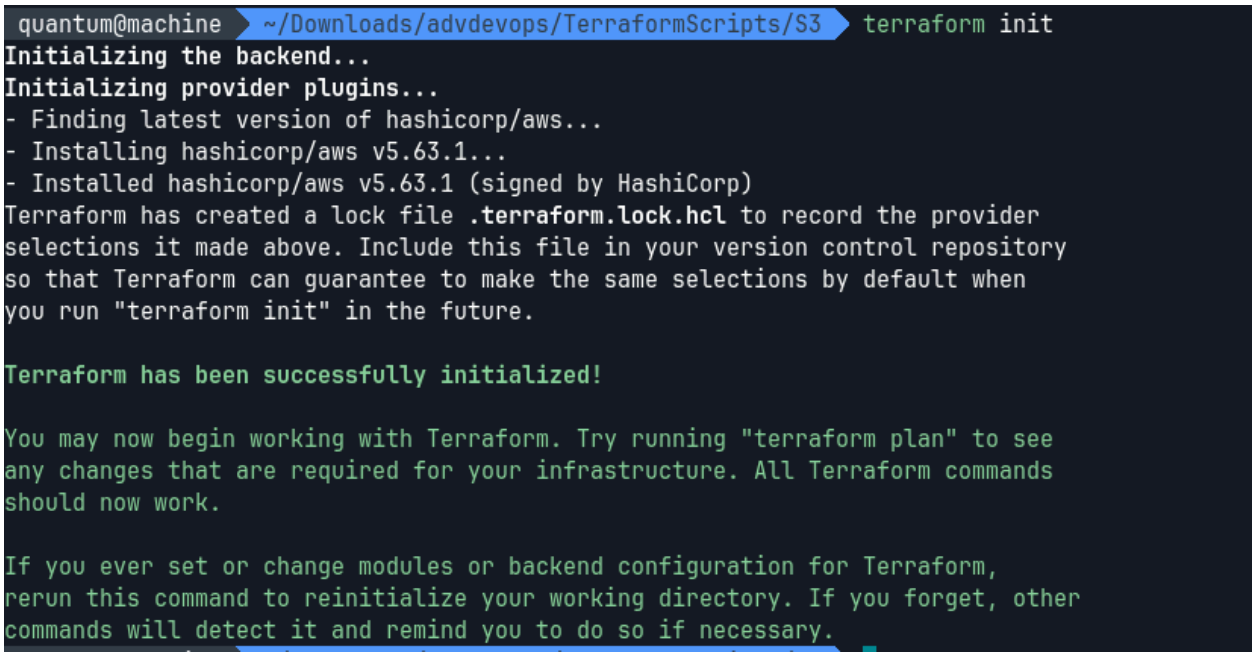
Step 3: Create a new provider.tf file and write the following contents into it.

```
provider "aws" {
  access_key = "AKIAQ3EGWP75IK42VSN3"
  secret_key = "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"
  region     = "ap-south-1"
}
```

Save both the files in same directory TerraformScripts/S3

Step 4: Open terminal ang go to TerraformScripts/S3 directory where our .tf files are stored

Step 5: Execute **terraform init** to initialize the resources



Step 6: Execute Terraform plan to see the available resources

```
quantum@machine ~/Downloads/advdevops/TerraformScripts/S3 sudo terraform plan
[sudo] password for quantum:

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:

  # aws_s3_bucket.kajal will be created
  + resource "aws_s3_bucket" "kajal" {
      + acceleration_status         = (known after apply)
      + acl                         = "public-read"
      + arn                         = (known after apply)
      + bucket                      = "my-bj-terraform-test-bucket"
      + bucket_domain_name          = (known after apply)
      + bucket_prefix               = (known after apply)
      + bucket_regional_domain_name = (known after apply)
      + force_destroy               = false
      + hosted_zone_id              = (known after apply)
      + id                          = (known after apply)
      + object_lock_enabled         = (known after apply)
      + policy                      = (known after apply)
      + region                      = (known after apply)
      + request_payer               = (known after apply)
      + tags                        = {
          + "Environment" = "Dev"
          + "Name"        = "My bucket"
        }
      + tags_all                    = {
          + "Environment" = "Dev"
          + "Name"        = "My bucket"
        }
      + website_domain              = (known after apply)
      + website_endpoint            = (known after apply)

      + cors_rule (known after apply)

      + grant (known after apply)

      + lifecycle_rule (known after apply)
```

```
      + logging (known after apply)

      + object_lock_configuration (known after apply)

      + replication_configuration (known after apply)

      + server_side_encryption_configuration (known after apply)

      + versioning (known after apply)

      + website (known after apply)
    }

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

  Warning: Argument is deprecated

    with aws_s3_bucket.kajal,
    on s3.tf line 3, in resource "aws_s3_bucket" "kajal":
     3:   acl    = "public-read"

  Use the aws_s3_bucket_acl resource instead

  (and one more similar warning elsewhere)


Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.
```

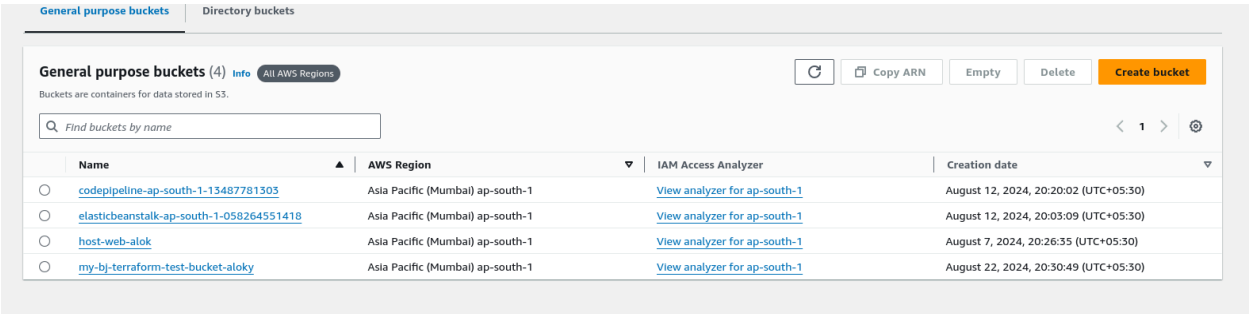
Step 7: Execute **terraform apply** to apply the configuration, which will automatically create an S3 bucket based on our configuration.

Name: Alok Yadav Div: D15C Roll No.: 59

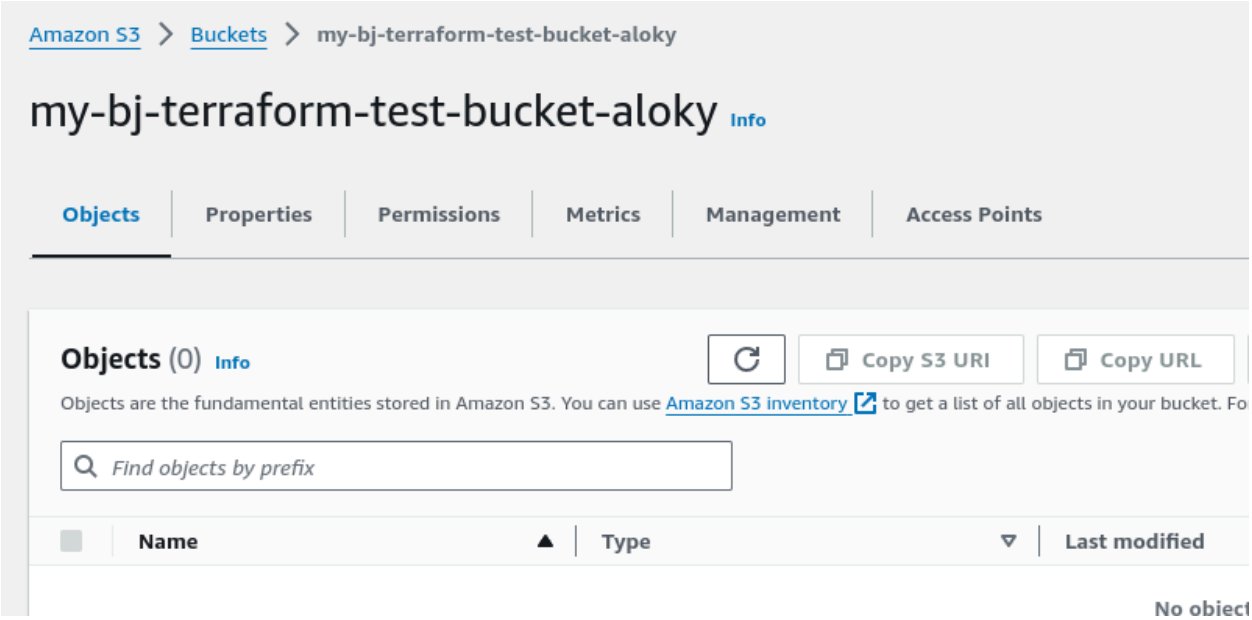
```
quantum@machine ~/Downloads/advdevops/TerraformScripts/S3 sudo terraform apply

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:

  # aws_s3_bucket.aloky will be created
  + resource "aws_s3_bucket" "aloky" {
      + acceleration_status         = (known after apply)
      + acl                         = (known after apply)
      + arn                         = (known after apply)
      + bucket                      = "my-bj-terraform-test-bucket-aloky"
      + bucket_domain_name          = (known after apply)
      + bucket_prefix               = (known after apply)
      + bucket_regional_domain_name = (known after apply)
      + force_destroy               = false
      + hosted_zone_id              = (known after apply)
      + id                          = (known after apply)
      + object_lock_enabled         = (known after apply)
      + policy                      = (known after apply)
      + region                      = (known after apply)
      + request_payer               = (known after apply)
      + tags                        = {
          + "Environment" = "Dev"
          + "Name"        = "My bucket"
        }
      + tags_all                    = {
          + "Environment" = "Dev"
          + "Name"        = "My bucket"
        }
      + website_domain              = (known after apply)
      + website_endpoint            = (known after apply)

      + cors_rule (known after apply)

      + grant (known after apply)
```

```
      + grant (known after apply)

      + lifecycle_rule (known after apply)

      + logging (known after apply)

      + object_lock_configuration (known after apply)

      + replication_configuration (known after apply)

      + server_side_encryption_configuration (known after apply)

      + versioning (known after apply)

      + website (known after apply)
    }

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

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

  Enter a value: yes

aws_s3_bucket.aloky: Creating...
aws_s3_bucket.aloky: Creation complete after 4s [id=my-bj-terraform-test-bucket-aloky]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

Step 8: Go to **S3 Buckets** dashboard to see newly created S3 bucket using terraform



Name: Alok Yadav Div: D15C Roll No.: 59



Name: Alok Yadav Div: D15C Roll No.: 59

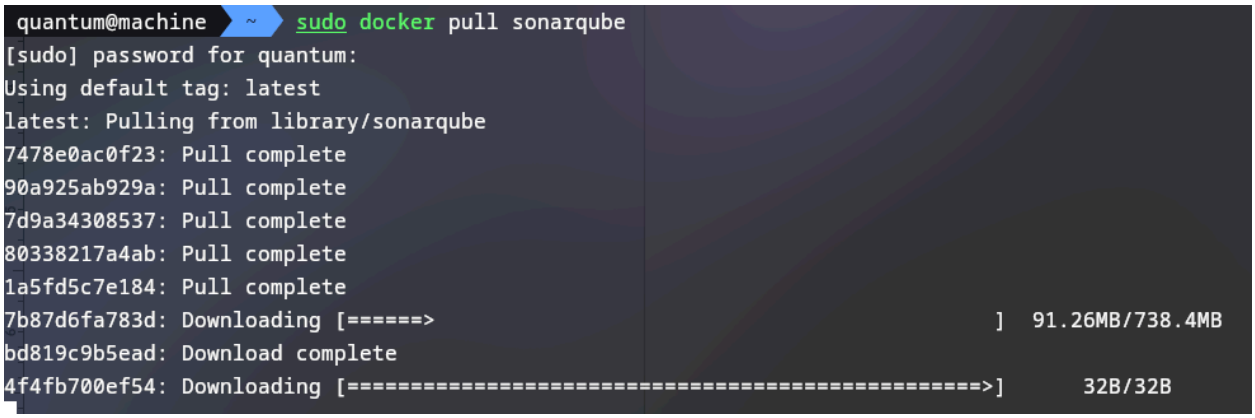
EXPERIMENT NO. 7

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

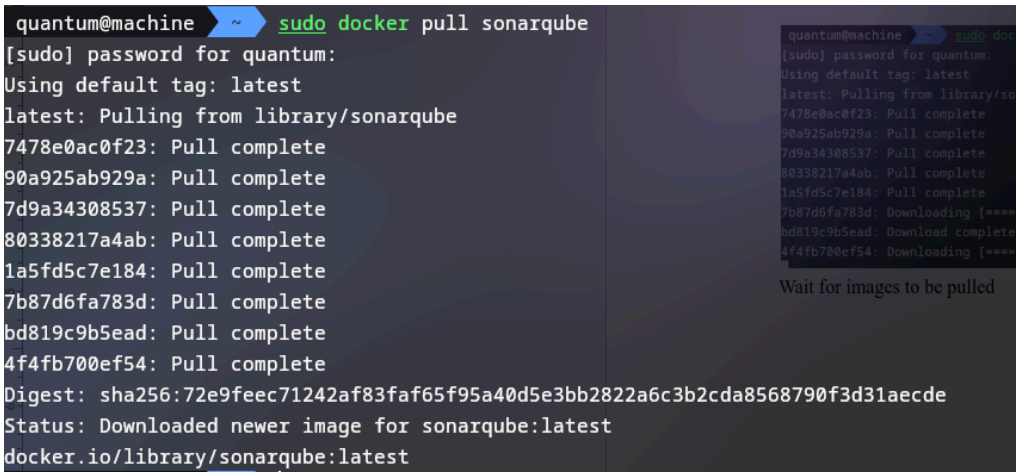
Prerequisites:
● Jenkins installed (Java JDK required)
● Docker Installed (for SonarQube)
docker engine service should be running (sudo systemctl start docker)

1. Install SonarQube Docker image

   sudo docker pull sonarqube

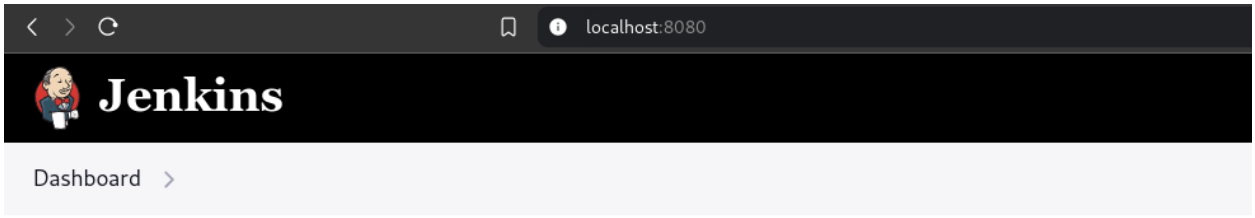


   Wait for images to be pulled



SonarQube installed successfully

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

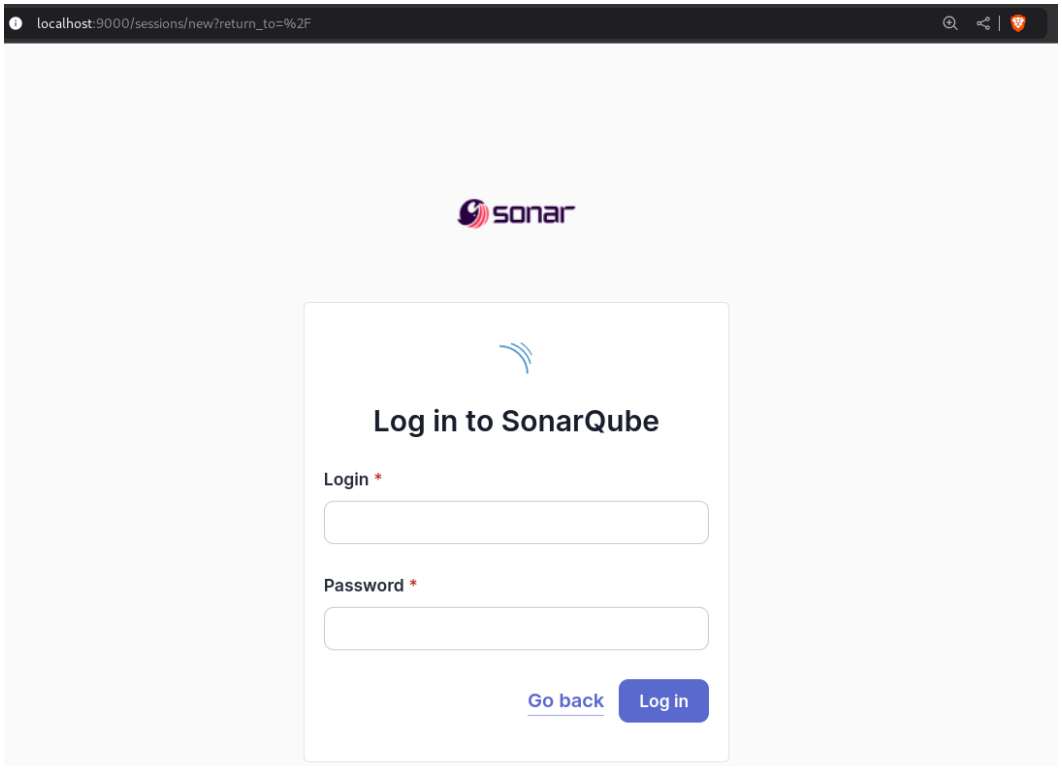


Our jenkins is running on port 8080

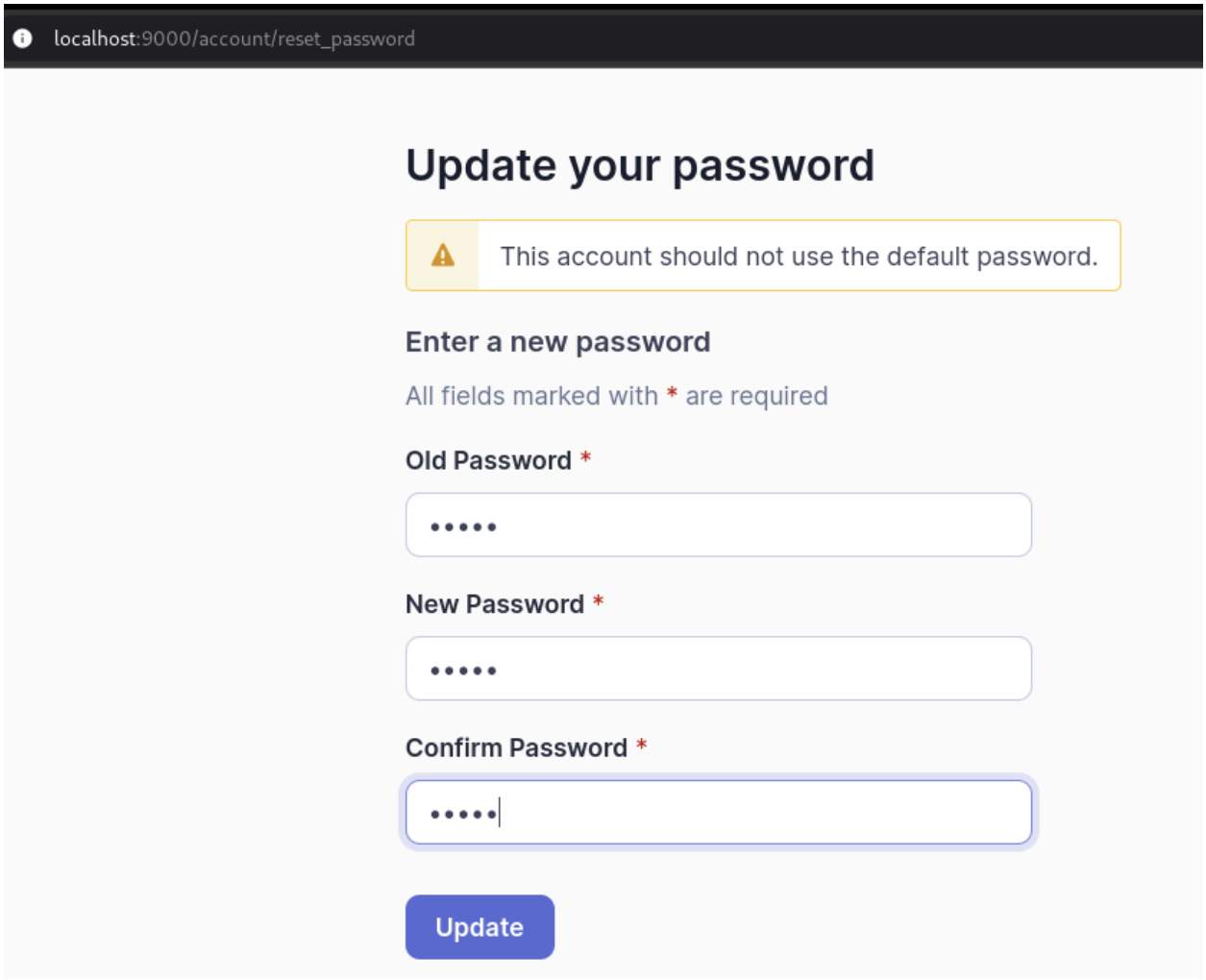
3. Run SonarQube in a Docker container using this command
sudo docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest

Warning: run below command only once

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

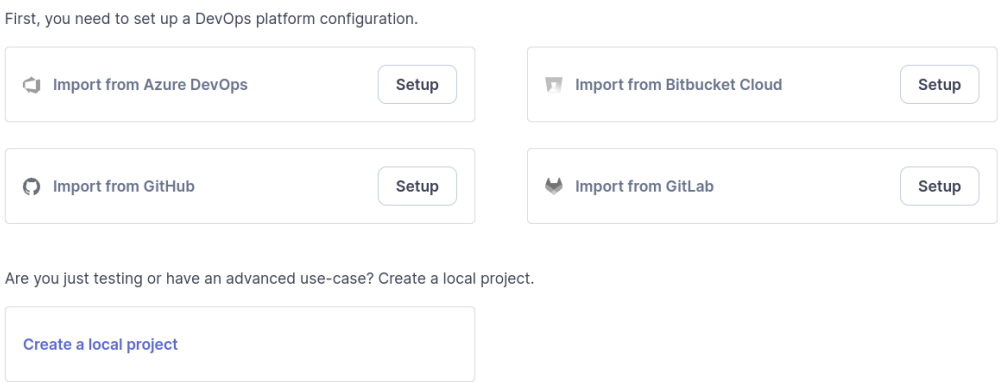


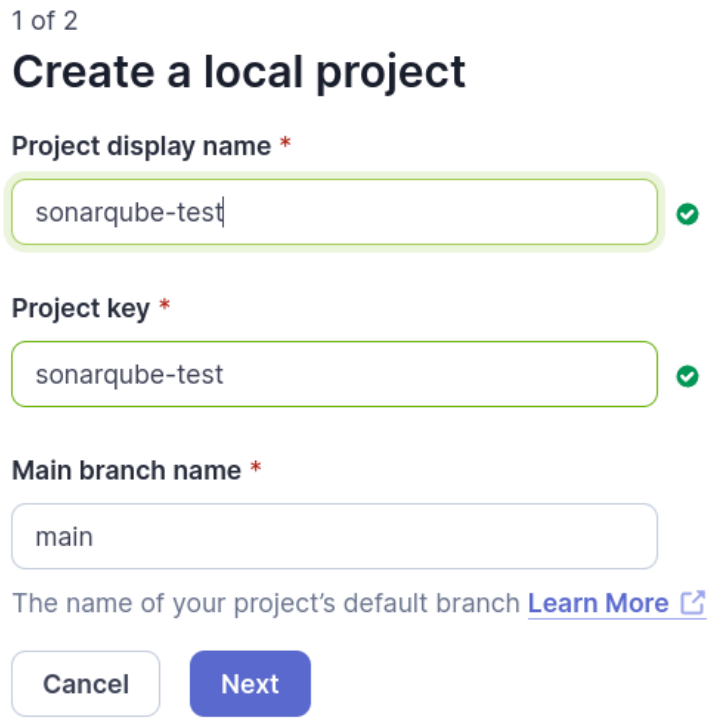
5. Login to SonarQube using username admin and password admin.

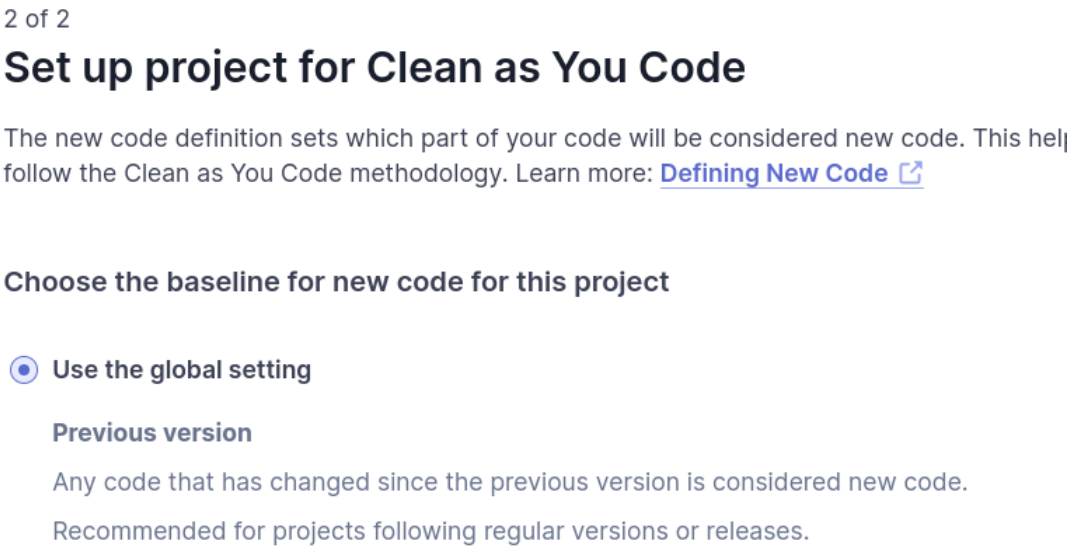


After logging, we have to change default password

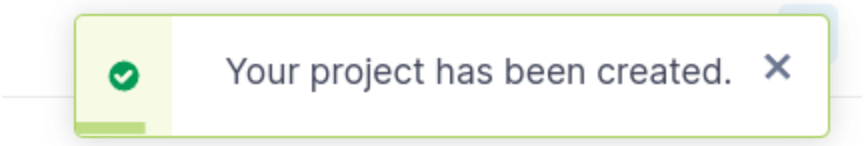
6. Create a manual project in SonarQube with the name sonarqube
   Click on **create a local project** on dashboard





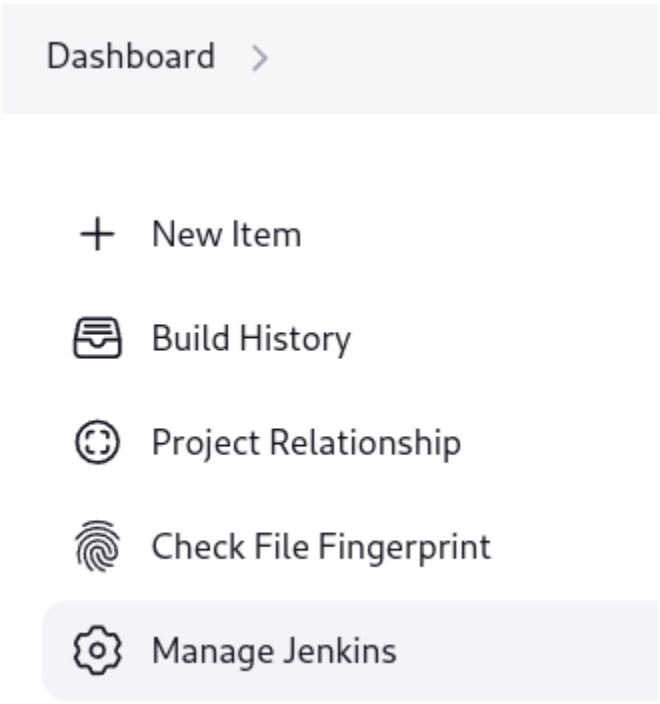


We can either use new custom settings for the project or use global settings
Here I'm using global setting

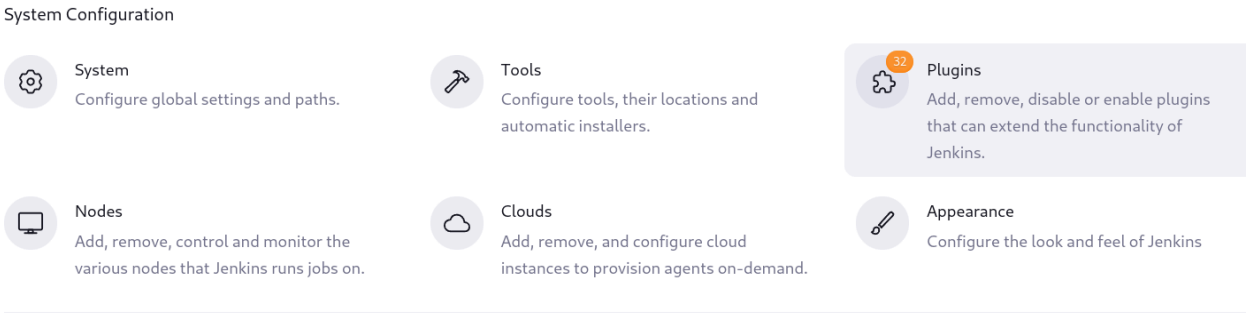


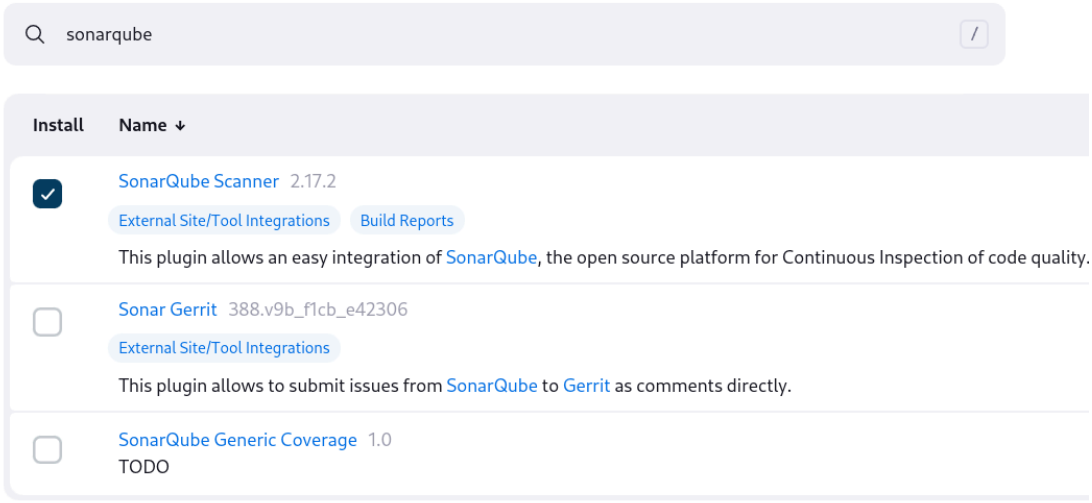
On successful creation of project, we a popup for the same

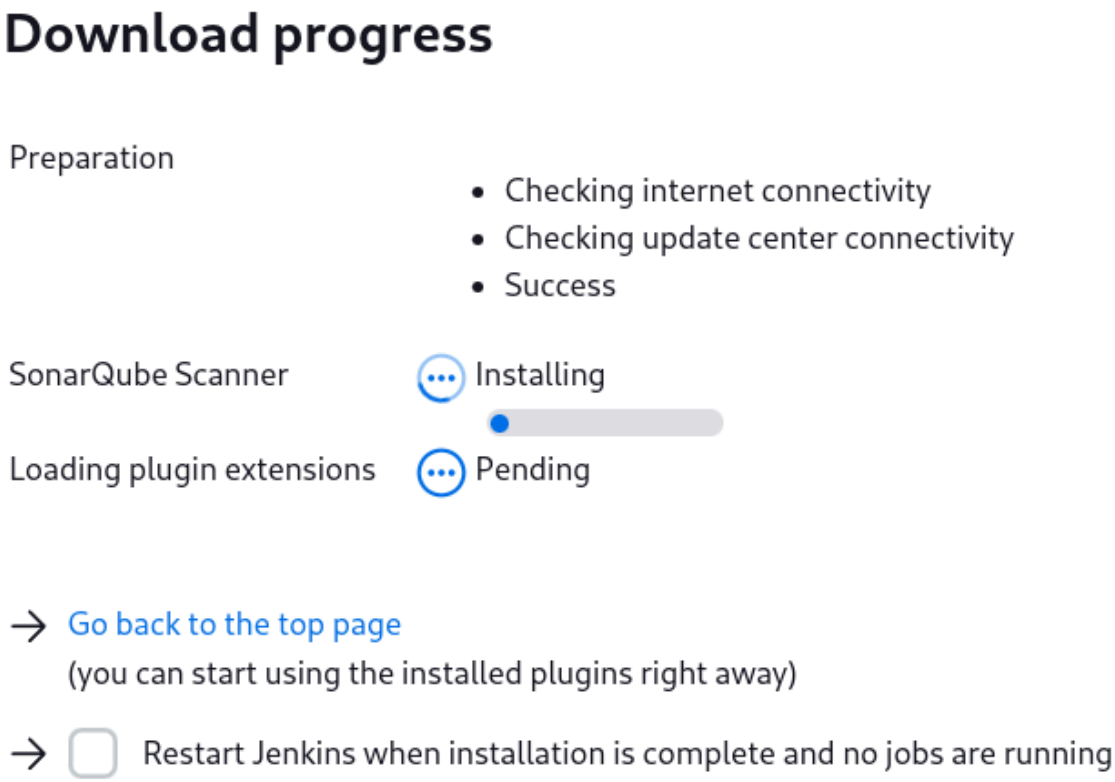
7. After setting project in sonarqube, go to **Jenkins Dashboard**



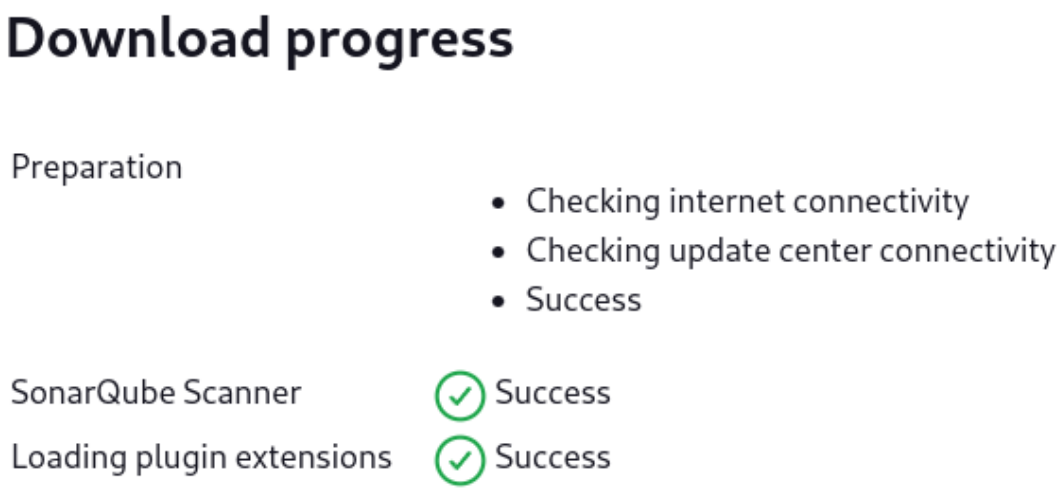
Go to Manage Jenkins and search for SonarQube Scanner in Plugins settings and install it.





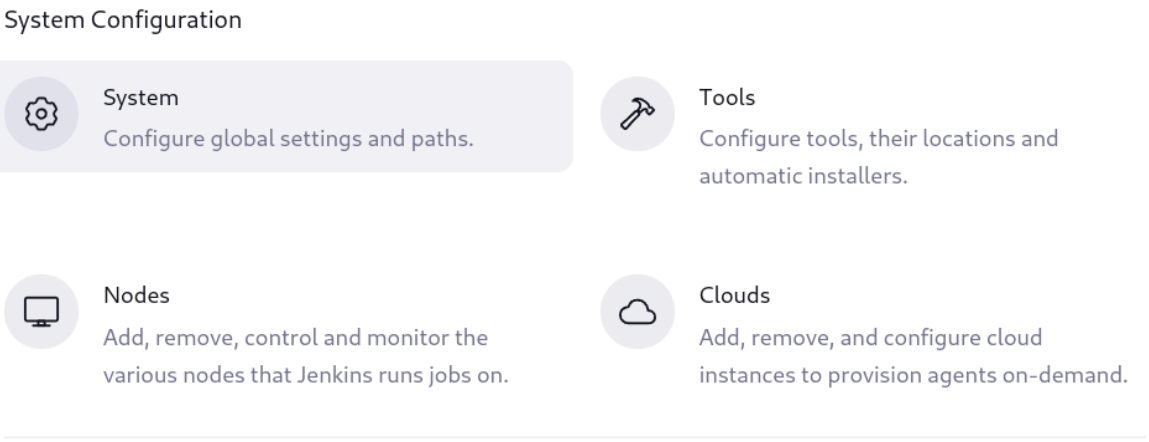


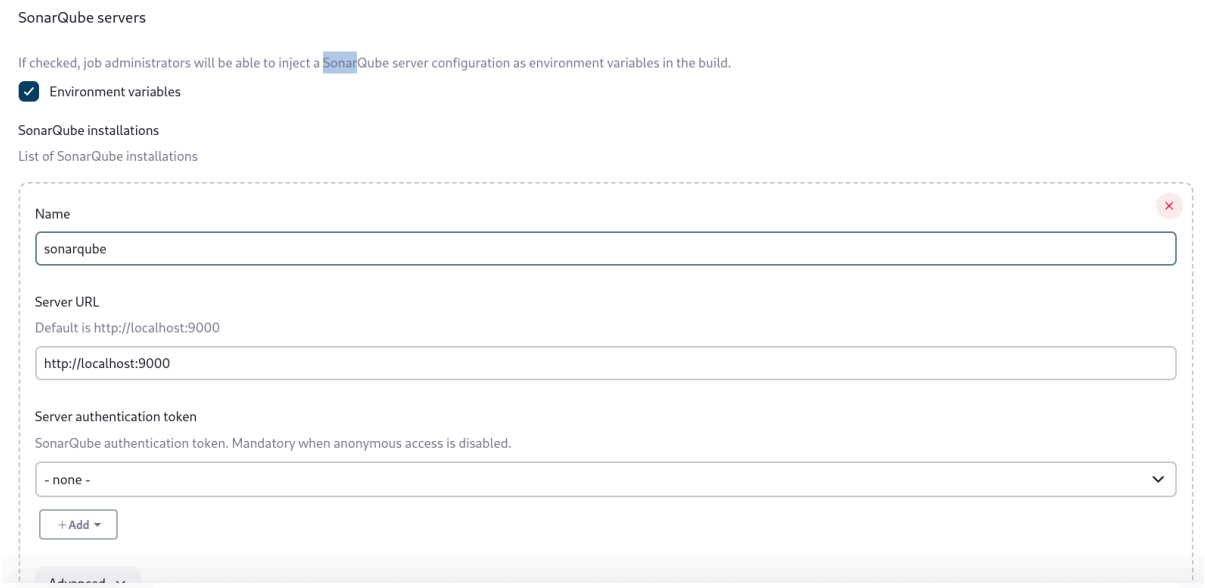
Our installation is in progress wait for it to download and install packages



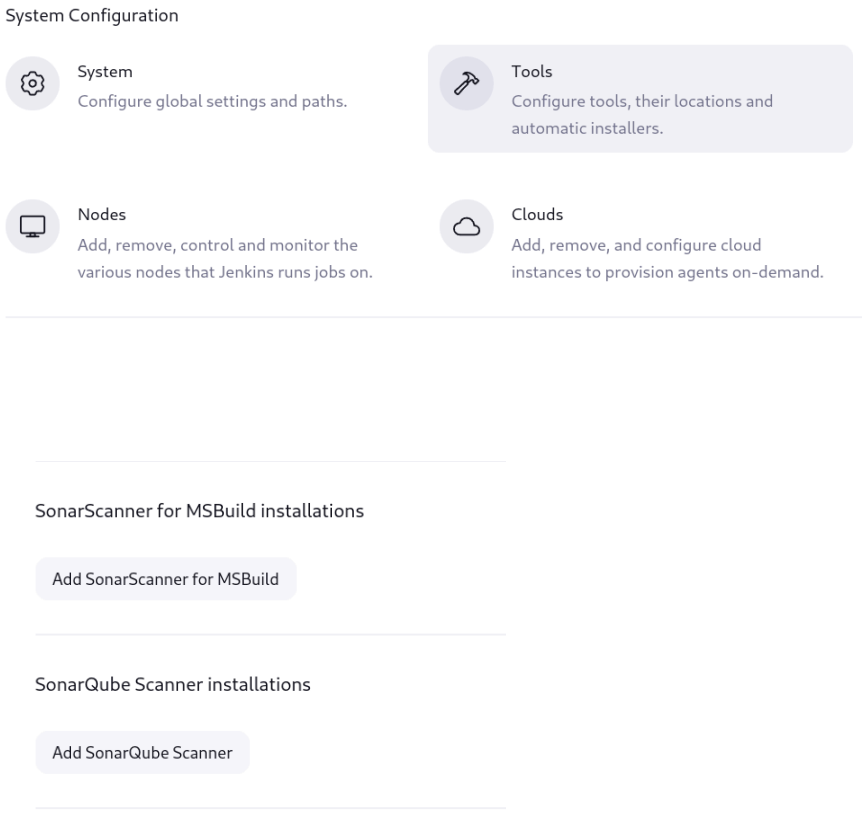
Plugin installed successfully

8. Under Jenkins dashboard ‘Configure System’, look for SonarQube Servers and enter the details.

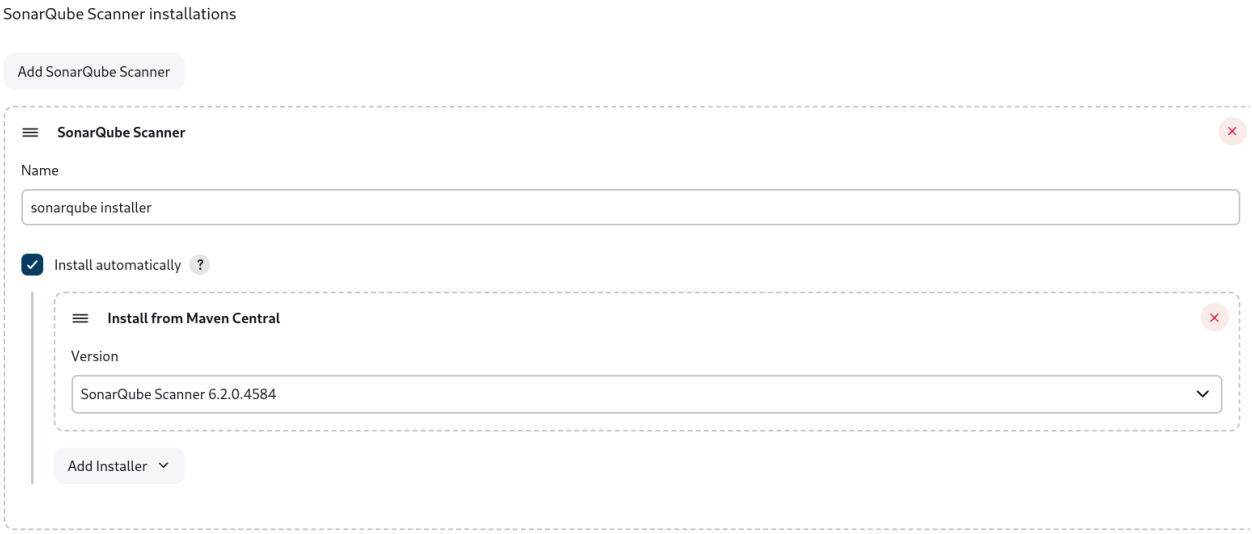




9. Search for SonarQube Scanner under Global Tool Configuration. Choose the latest configuration and choose Install automatically.

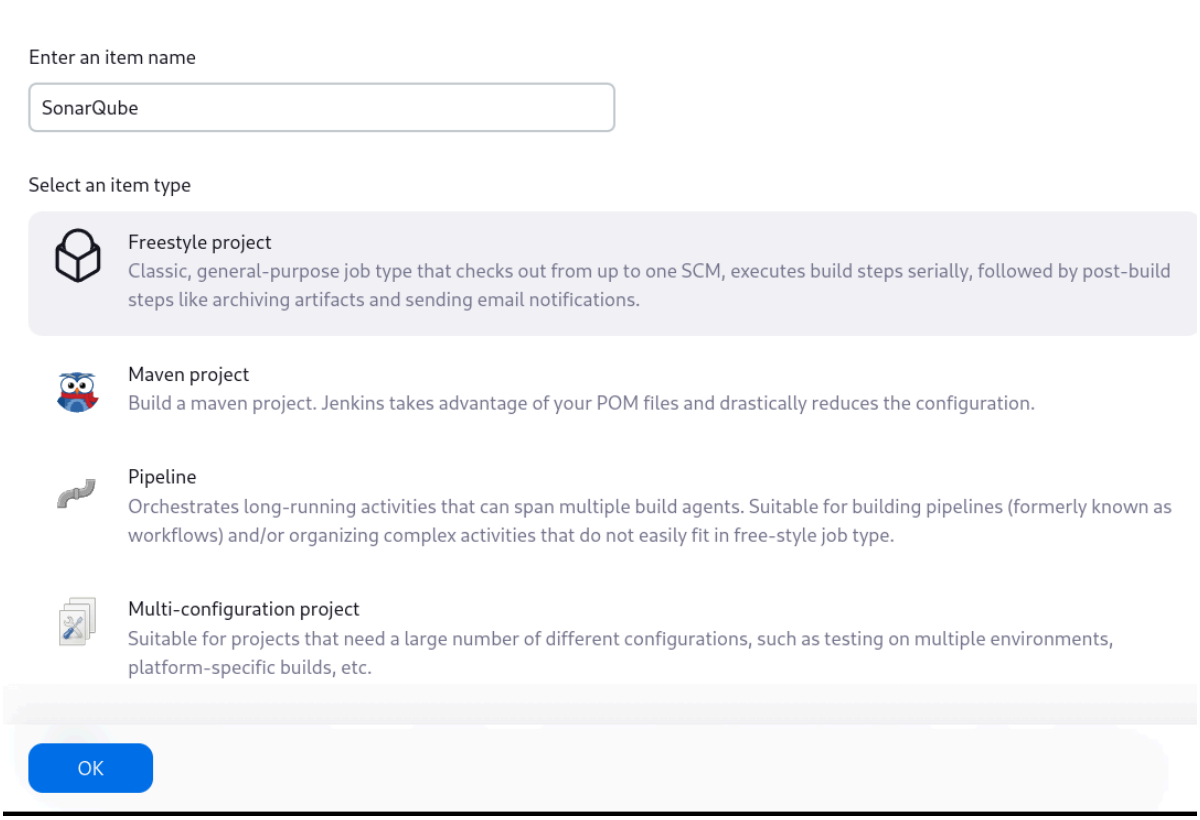


Click on **Add SonarQube Scanner**

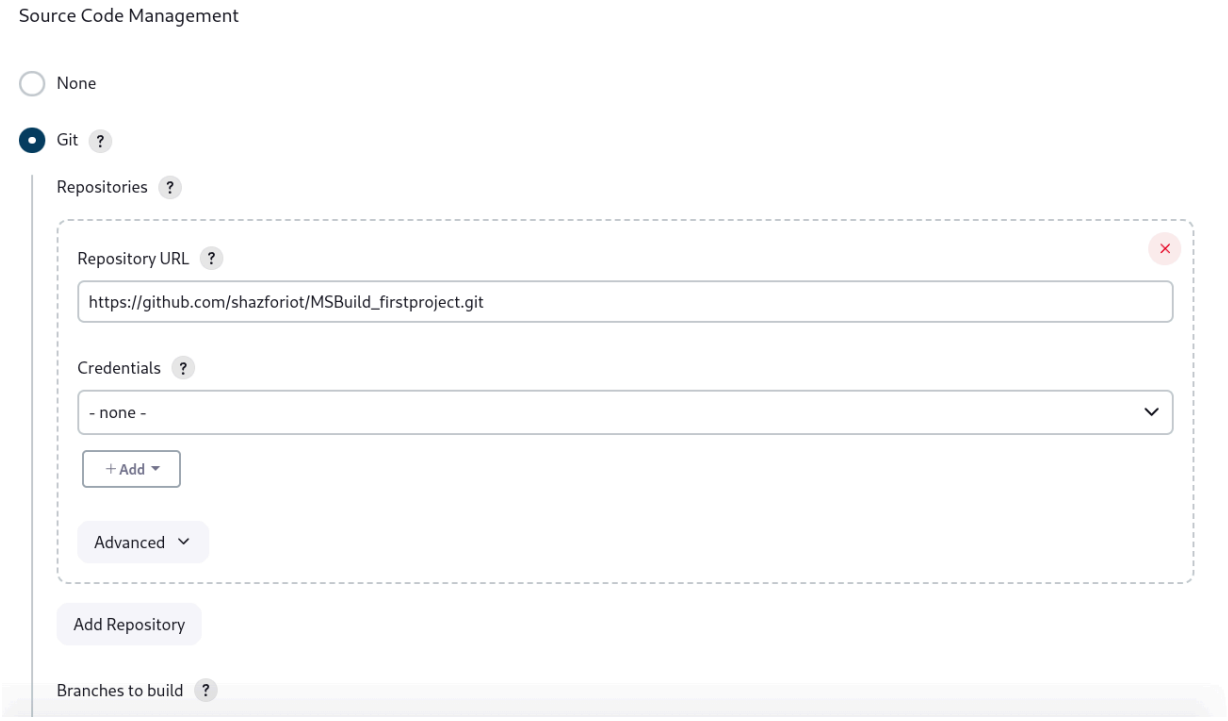


Select Latest version and save configuration

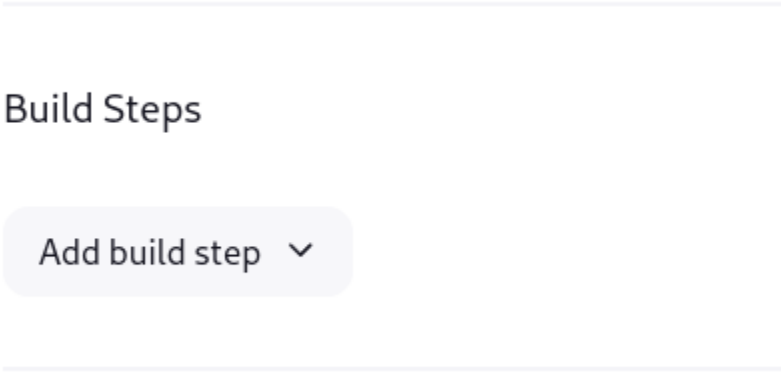
10. After the configuration, create a New Item in Jenkins, choose a freestyle project.



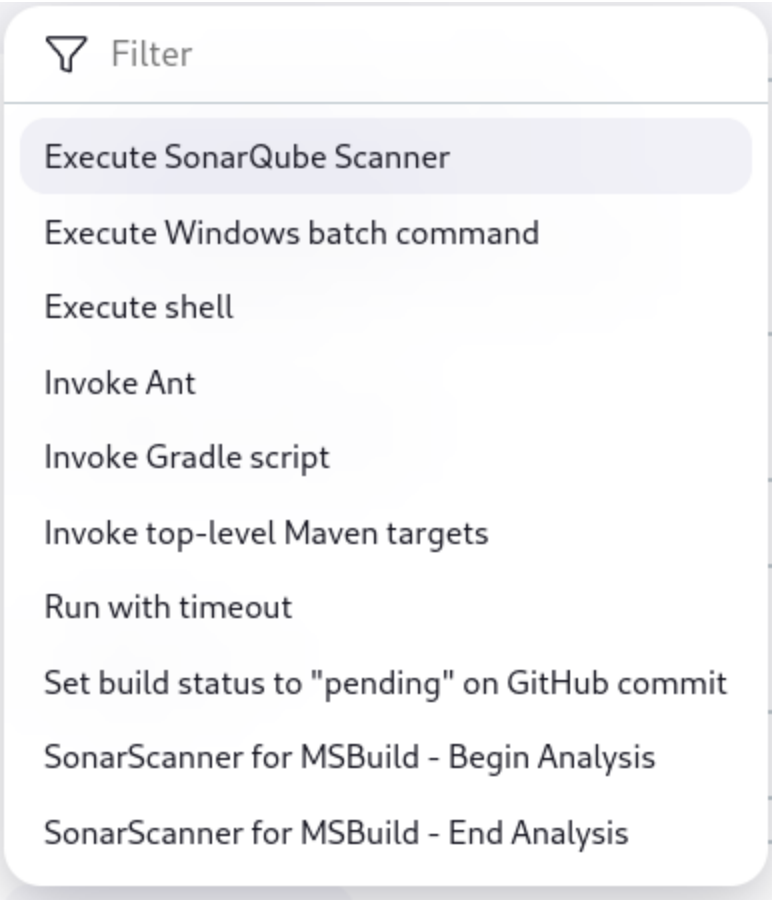
11. Choose this GitHub repository in Source Code Management. https://github.com/shazforiot/MSBuild_firstproject.git It is a sample hello-world project with no vulnerabilities and issues, just to test the integration



12. Under Build-> Execute SonarQube Scanner

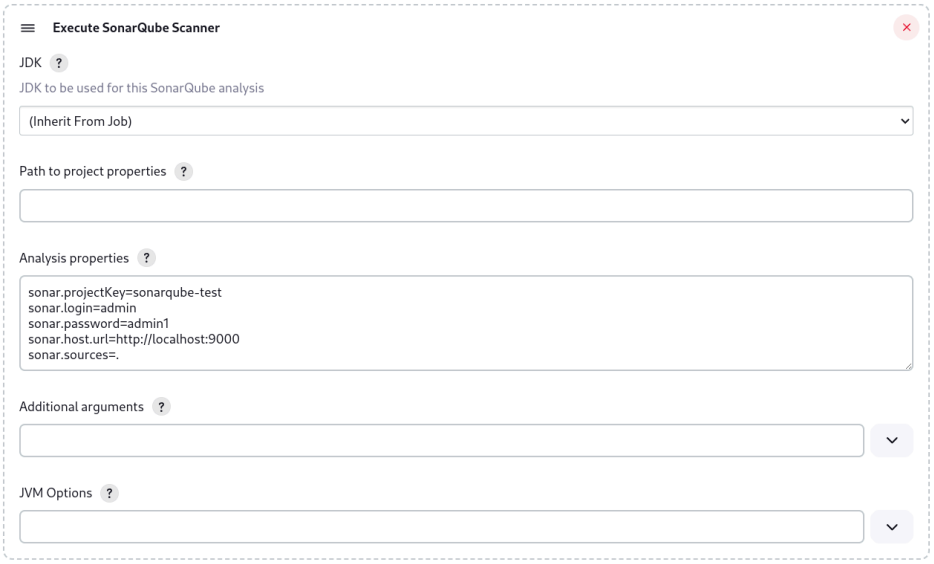


Click on add build steps

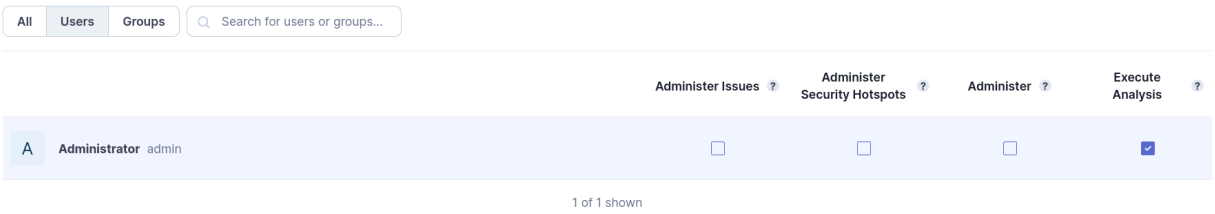


Then click on **Execute SonarQube Scanner**

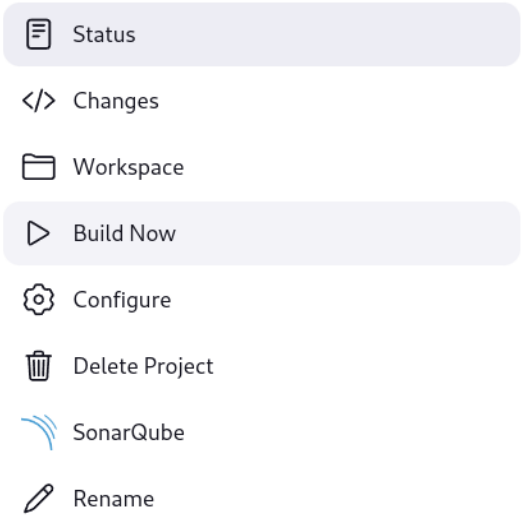
13. Mention the SonarQube Project Key, Login, Password, Source path and Host URL in Analysis properties



14. Go to http://localhost:9000/project_roles?id=<project_key> and allow Execute Permissions to the Admin user.



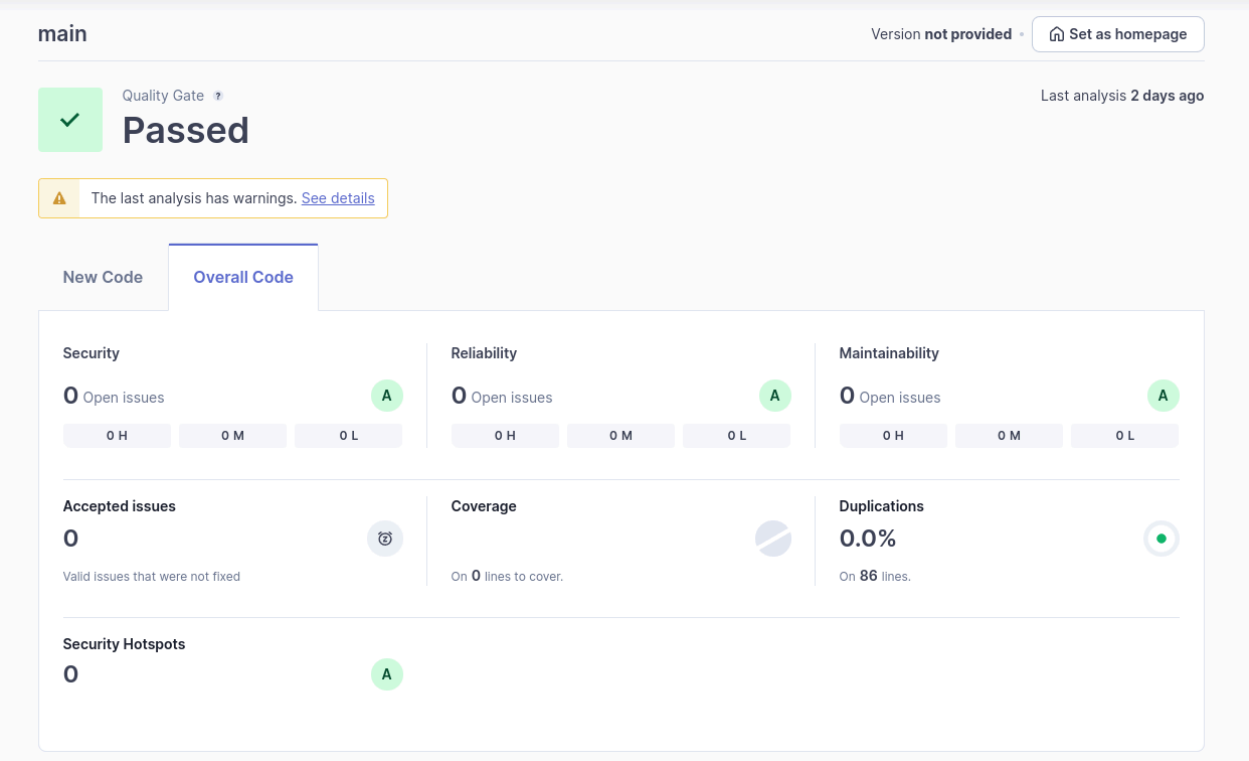
15. Run The Build.



16. Check the console output

```
10:30:32.234 INFO  ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=sonarqube-test
10:30:32.234 INFO  Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
10:30:32.235 INFO  More about the report processing at http://localhost:9000/api/ce/task?id=96795973-9667-4456-a15c-3311dbc9d067
10:30:32.244 INFO  Analysis total time: 9.533 s
10:30:32.245 INFO  SonarScanner Engine completed successfully
10:30:32.300 INFO  EXECUTION SUCCESS
10:30:32.301 INFO  Total time: 14.090s
Finished: SUCCESS
```

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



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

Conclusion: We began the experiment with installation of SonarQube Docker Image followed by setting up a new project in SonarQube. Then we installed the SonarQube scanner plugin and then created a new freestyle project in Jenkins with a Git repository for code analysis. Then we configured the jenkins with appropriate settings to work with sonarqube. Gave permissions to Jenkins to perform code analysis. It is essential to provide correct properties in **Analysis Properties** for Jenkins to run correctly.The jenkins project ran successfully with all tests passed in SonarQube

EXPERIMENT NO. 8

Aim: Create a Jenkins CI/CD Pipeline with SonarQube / GitLab Integration to perform a static
analysis of the code to detect bugs, code smells, and security vulnerabilities on a sample Web /Java / Python application.

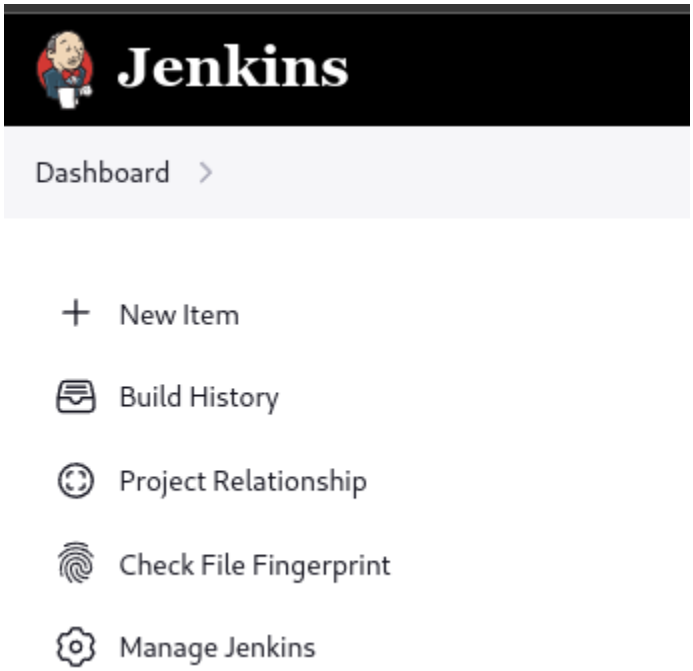
Integrating Jenkins with SonarQube:

Prerequisites:

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

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

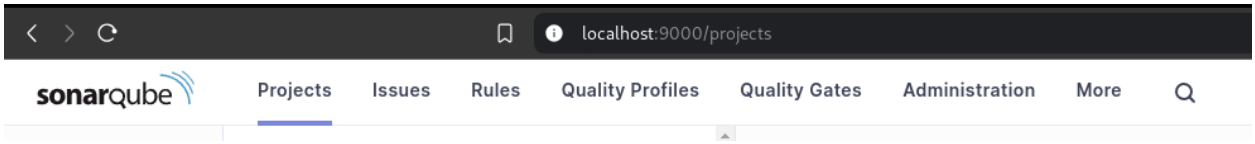
1. Open up Jenkins Dashboard on localhost, port 8080 or whichever port it is at for you.



2. Run SonarQube in a Docker container using this command

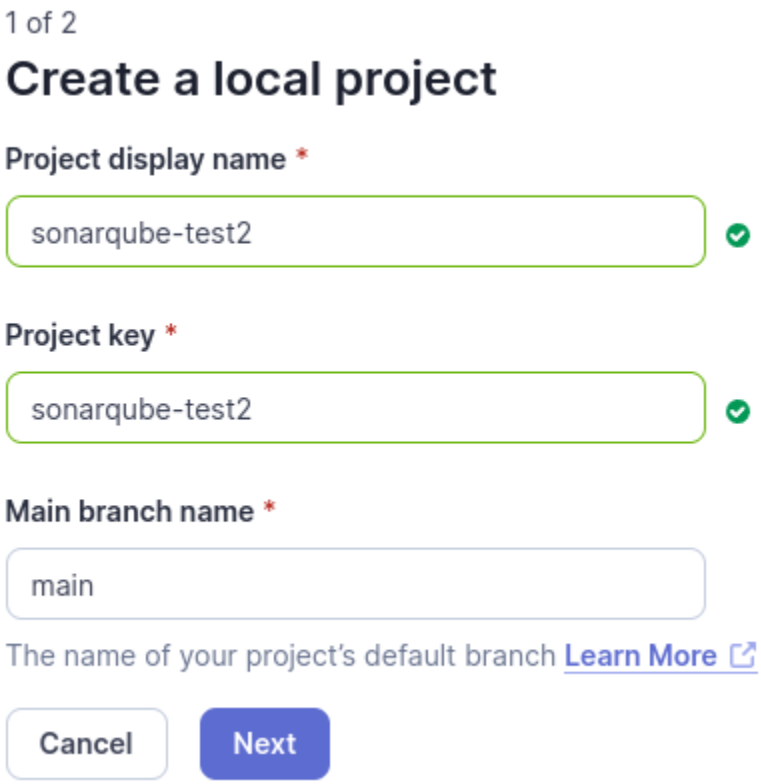
```
quantum@machine ~ sudo docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
```

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



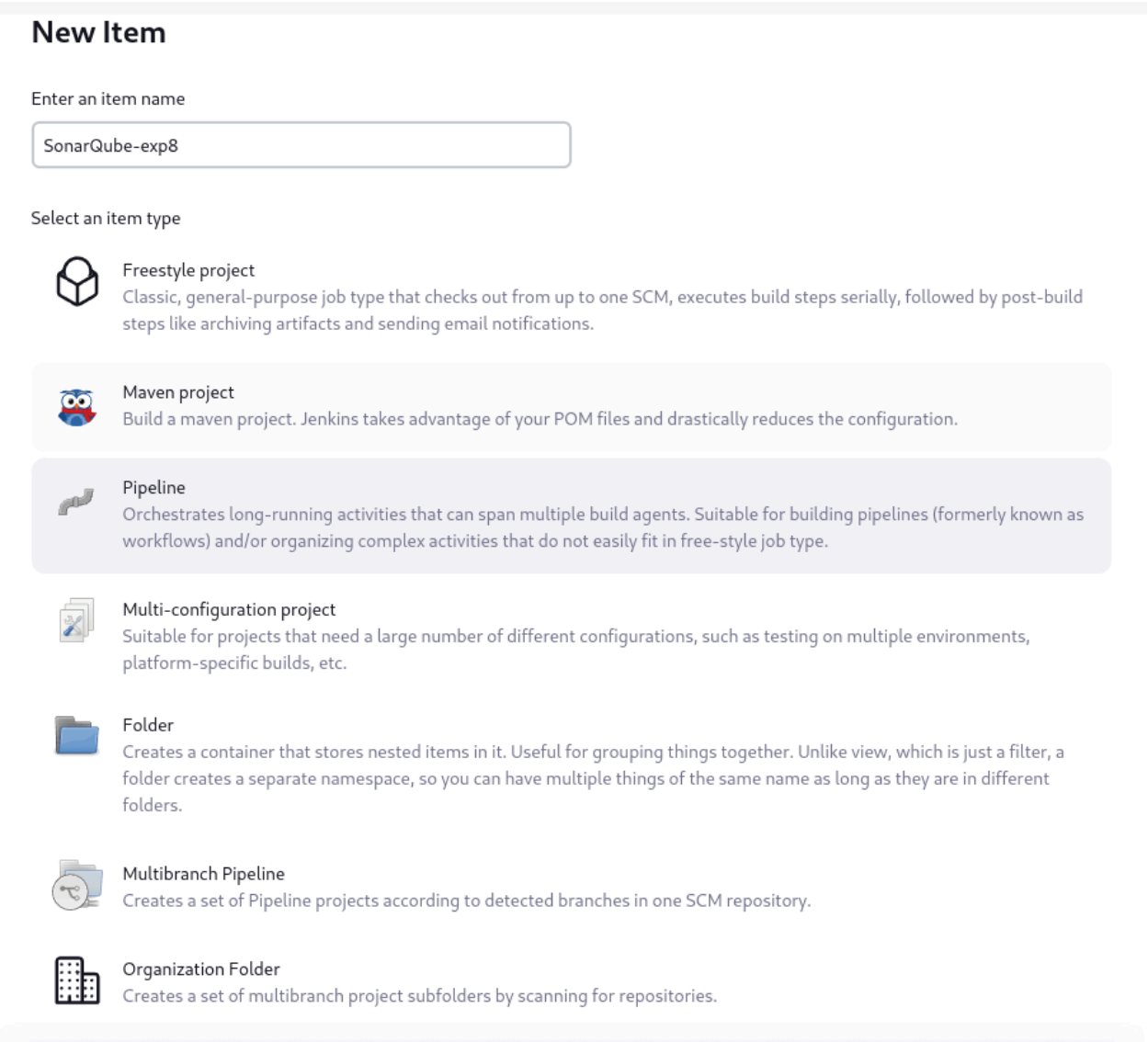
4. Login to SonarQube using your username and password

5. Create a manual project in SonarQube with the name sonarqube-test2



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. Install sonar-scanner

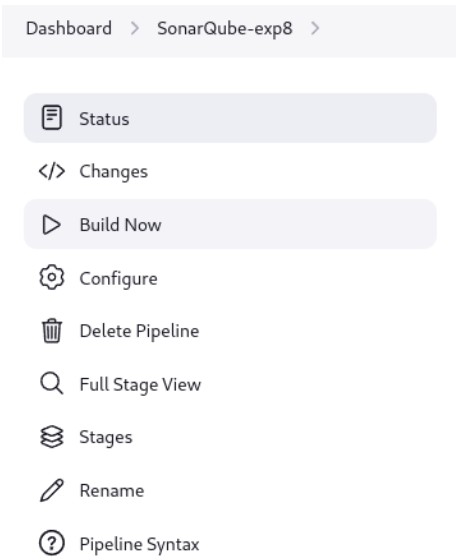
Now we need to install **sonar-scanner** binary to perform code analysis
To do so, go to
https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar-scanner-cli-6.2.0.4584-linux-x64.zip for linux OS
(https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar-scanner-cli-6.2.0.4584-windows-x64.zip for windows OS)
You will be prompted to download the zip file, download it and extract it and copy the path (absolute path) to <ROOT_DIRECTORY>/bin/sonar-scanner in my case it is /opt/sonar-scanner/bin/sonar-scanner (For linux)
For windows path would be
C:\\Users\<USER_NAME>\Downloads\sonar-scanner-cli-6.2.0.4584-windows-x64\bin\sonar-scanner

Now we need to update the required details in the pipeline script as :

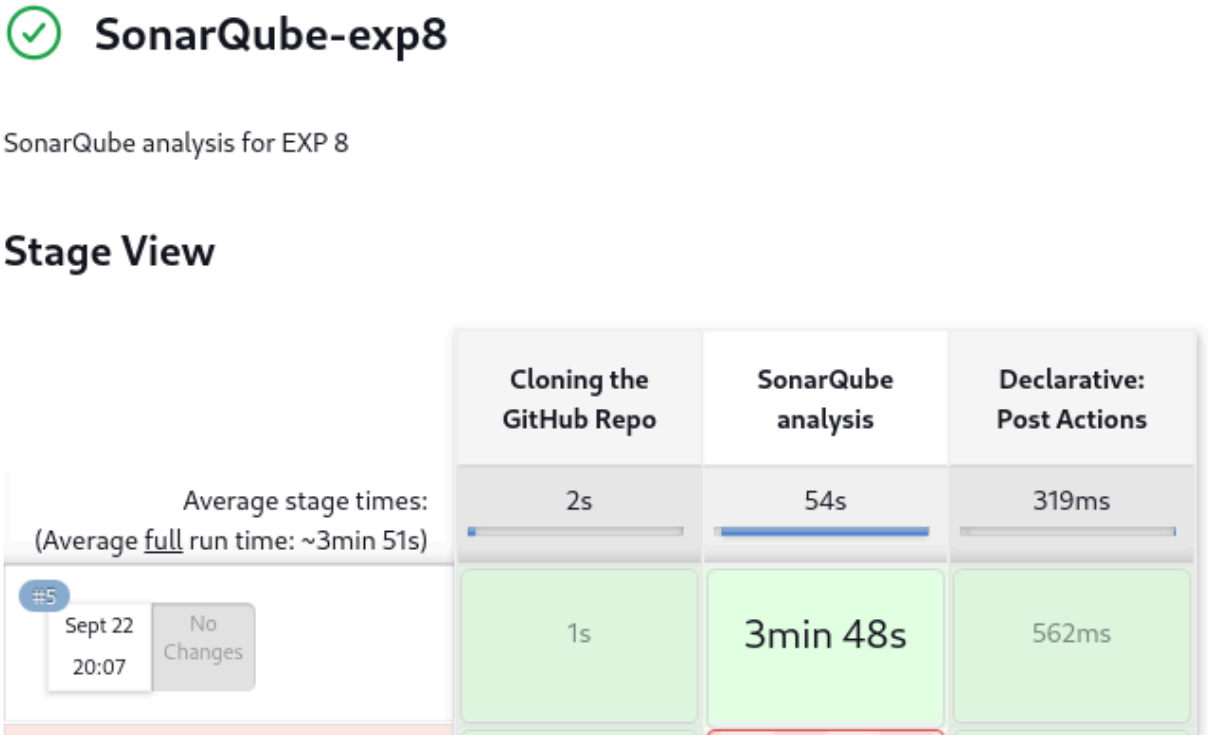
```
pipeline {
    agent any
    stages {
        stage('Cloning the GitHub Repo') {
            steps {
                git 'https://github.com/shazforiot/GOL.git'
            }
        }
        stage('SonarQube analysis') {
            steps {
                withSonarQubeEnv('sonarqube') {configuration
                    sh """
                        /opt/sonar-scanner/bin/sonar-scanner \   //replace with your actual path to sonar-scanner
                        -D sonar.projectKey=sonarqube-test2 \
                        -D sonar.sources=. \
                        -D sonar.exclusions=vendor/**,resources/**,**/*.java \
```

```
                -D sonar.host.url=http://127.0.0.1:9000 \
                -D sonar.login=admin \
                -D sonar.password=admin1
            """
        }
    }
  }
}
```

9. Run the build

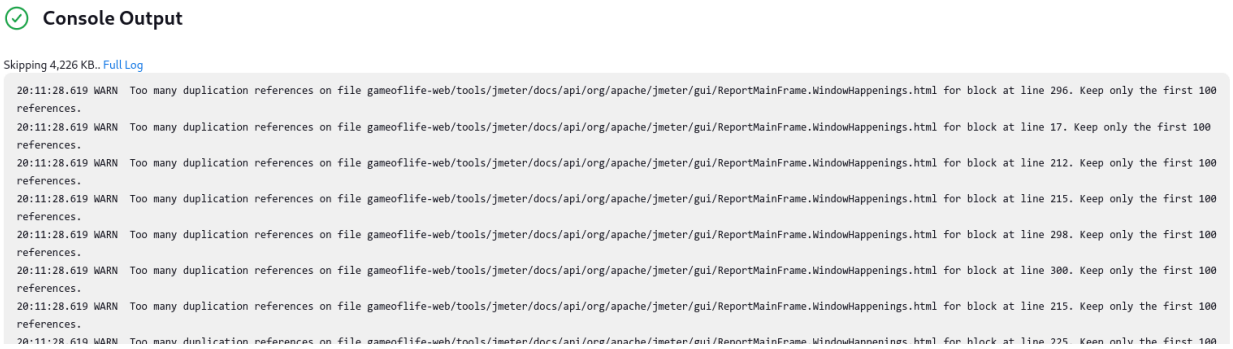


Check the status of Build



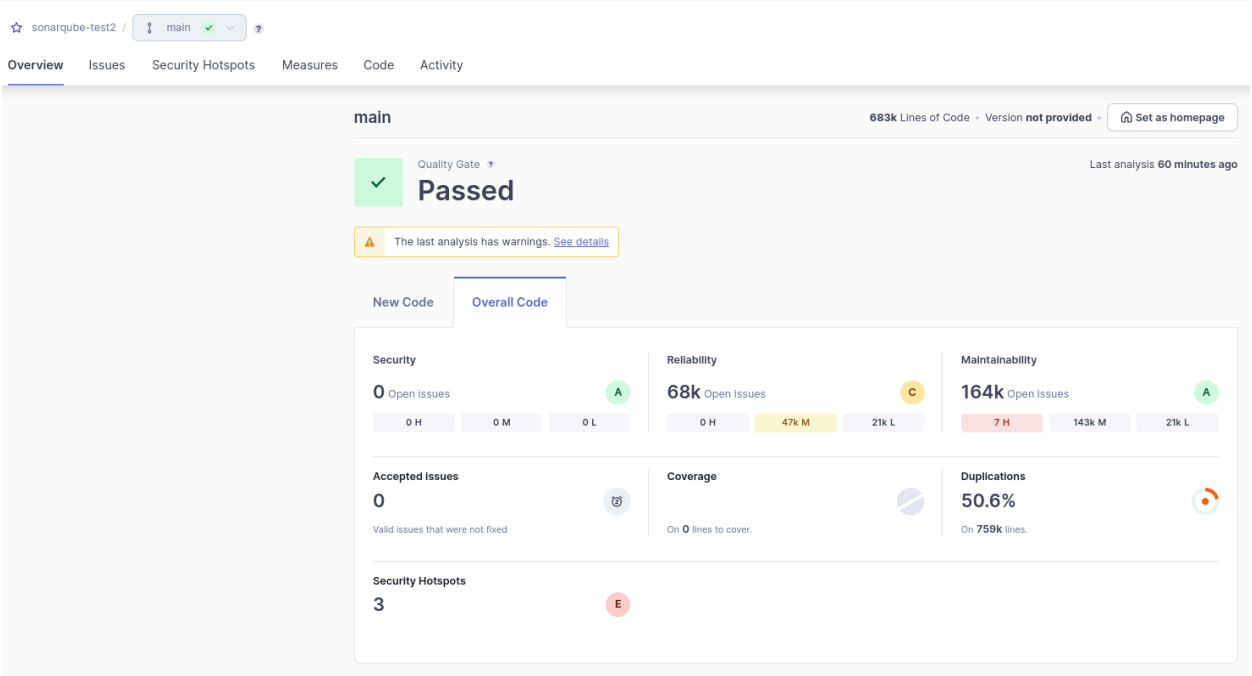
As we can see the SonarQube analysis is completed

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



```
20:11:31.504 INFO  CPD Executor CPD calculation finished (done) | time=104775ms
20:11:31.748 INFO  SCM revision ID 'ba799ba7e1b576f04a4612322b0412c5e6e1e5e4'
20:11:33.306 INFO  Analysis report generated in 1507ms, dir size=127.2 MB
20:11:40.819 INFO  Analysis report compressed in 7511ms, zip size=29.6 MB
20:11:42.147 INFO  Analysis report uploaded in 1322ms
20:11:42.154 INFO  ANALYSIS SUCCESSFUL, you can find the results at: http://127.0.0.1:9000/dashboard?id=sonarqube-test2
20:11:42.154 INFO  Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
20:11:42.154 INFO  More about the report processing at http://127.0.0.1:9000/api/ce/task?id=670aec3c-6c20-460b-ad52-ec6ce041fb1f
20:11:46.802 INFO  Analysis total time: 3:42.216 s
20:11:46.835 INFO  SonarScanner Engine completed successfully
20:11:48.173 INFO  EXECUTION SUCCESS
20:11:48.296 INFO  Total time: 3:47.113s
[Pipeline] }
[Pipeline] // withSonarQubeEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Declarative: Post Actions)
[Pipeline] echo
Pipeline completed.
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

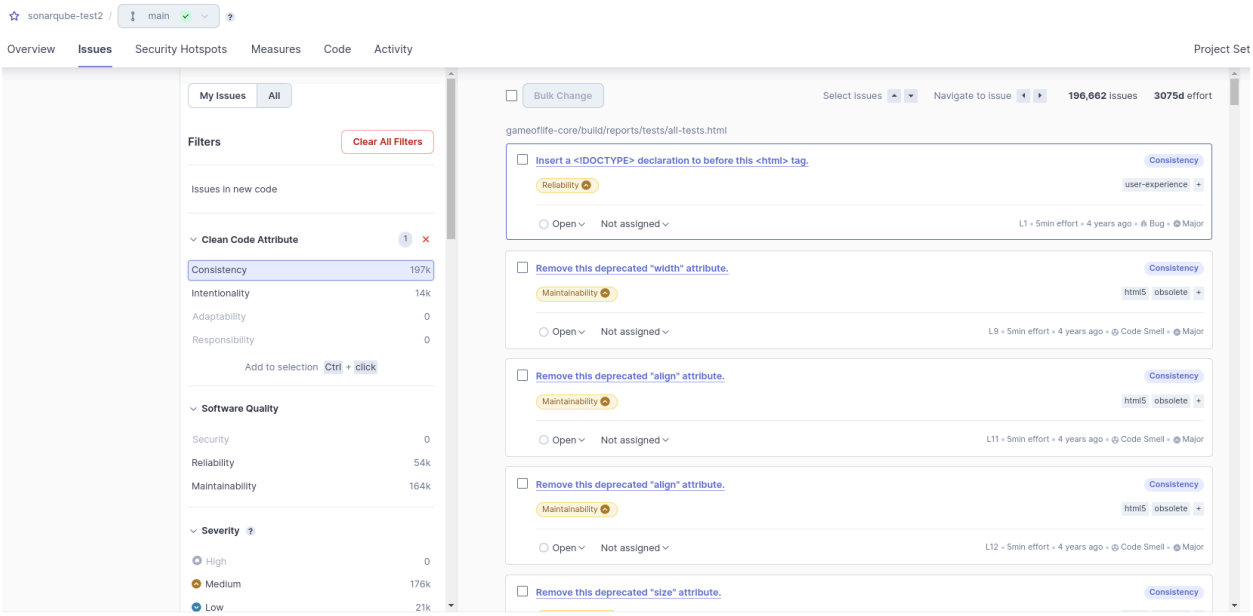
11. After that, check the project in SonarQube.



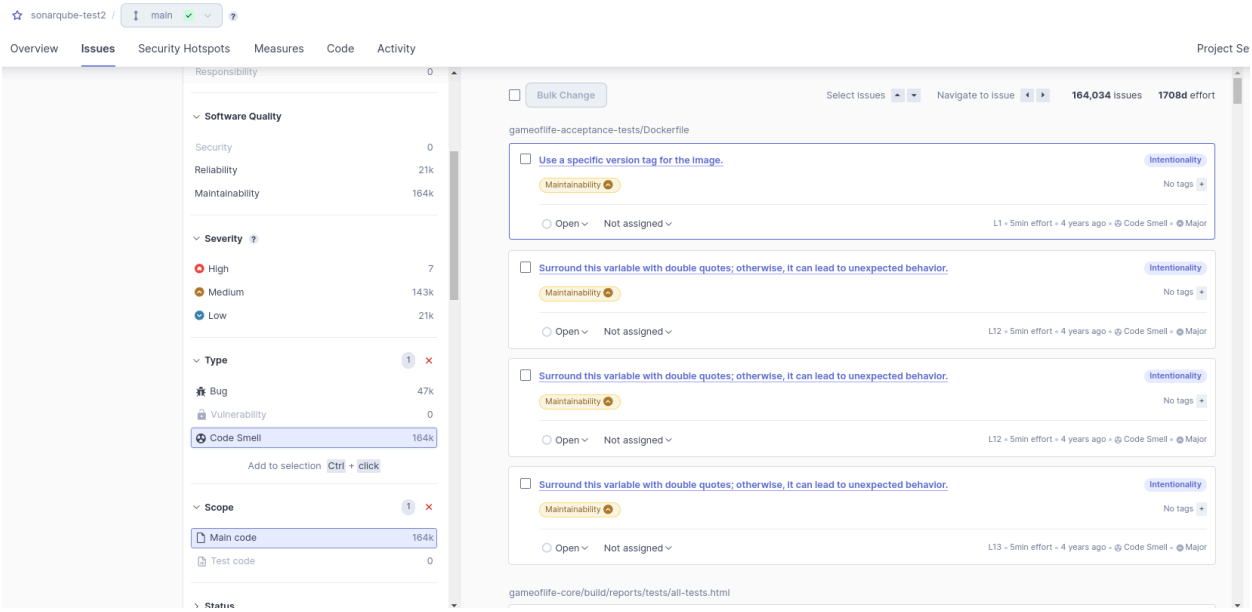
Under different tabs, check all different issues with the code

12. **Code Problems-**

Name: Alok Yadav Div: D15C Roll No.: 59

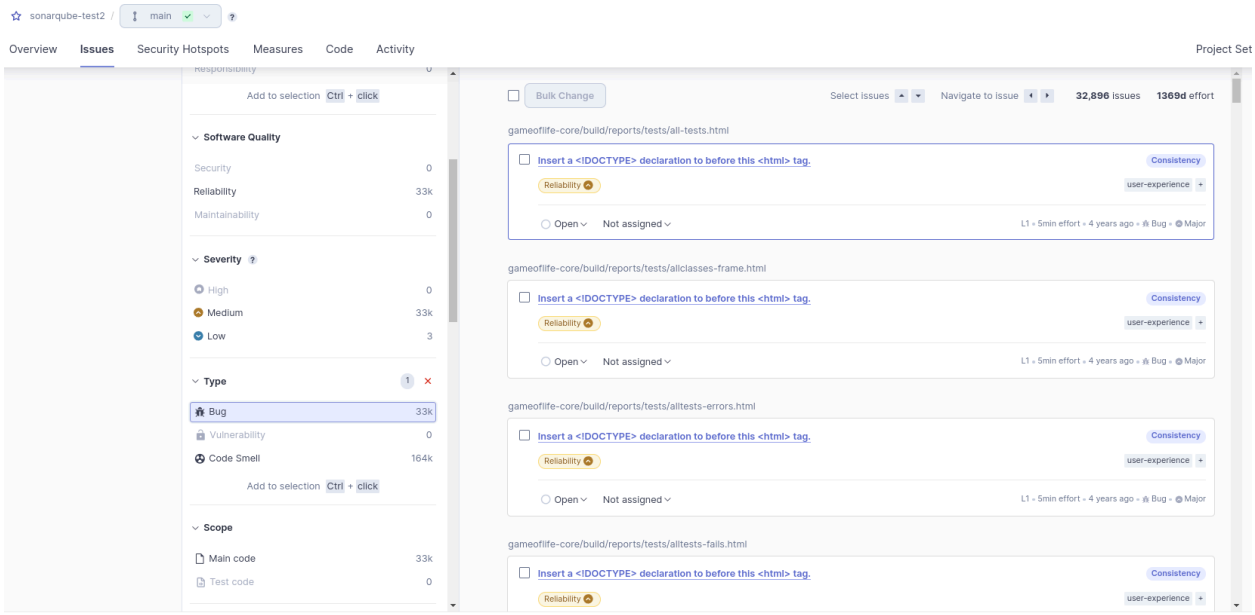


Code Smells



Bugs

Name: Alok Yadav Div: D15C Roll No.: 59



Conclusion: We began the experiment with creating a new project in SonarQube and setting up a new Pipeline in Jenkins with proper configuration of pipeline script. Then we installed Sonar Scanner CLI so that jenkins can do code analysis of Git Repository. We can also configure the pipeline to use the installed Sonar Scanner plugin instead of locally installed Binary of Sonar Scanner.The pipeline ran successfully with all tests passed in SonarQube

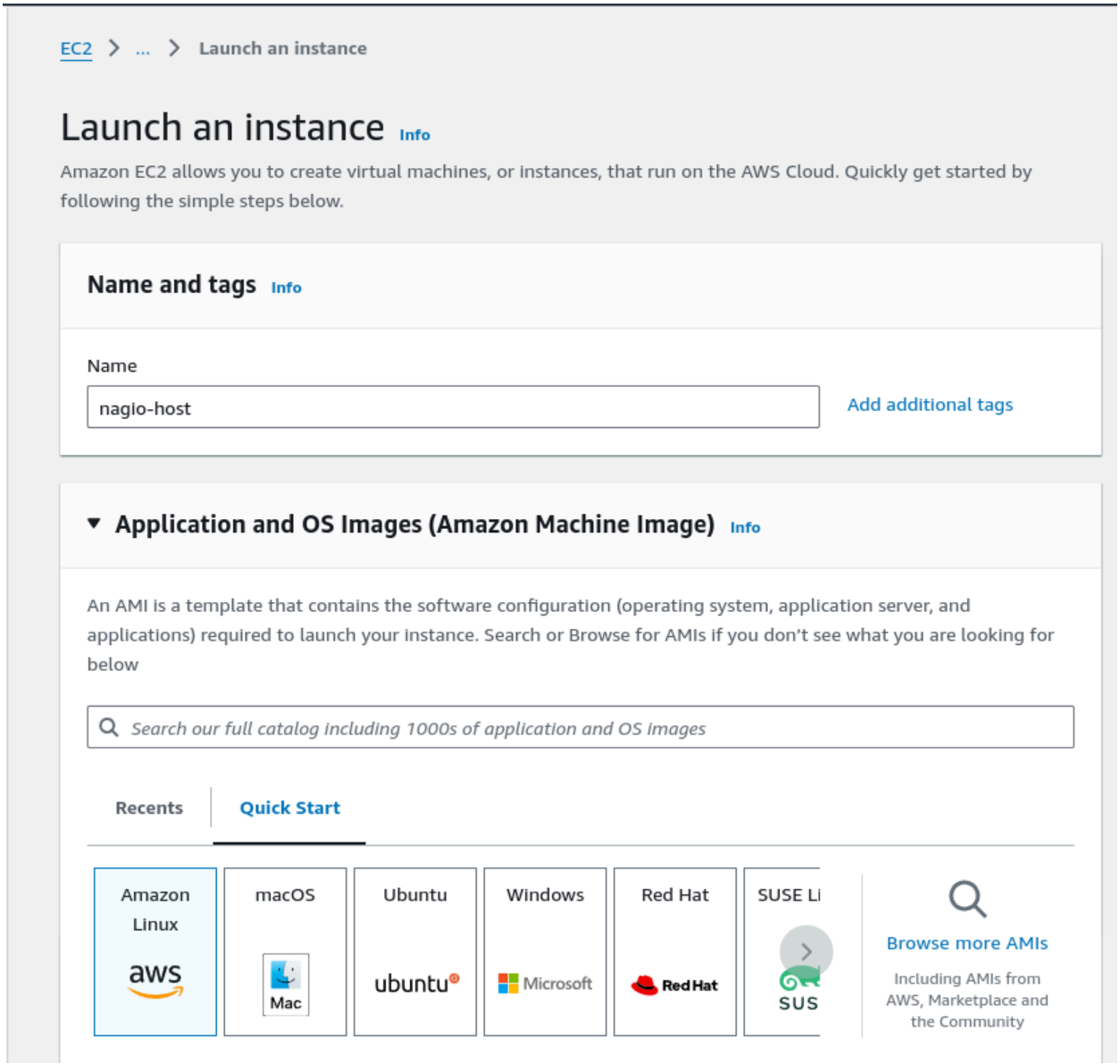
EXPERIMENT NO. 9

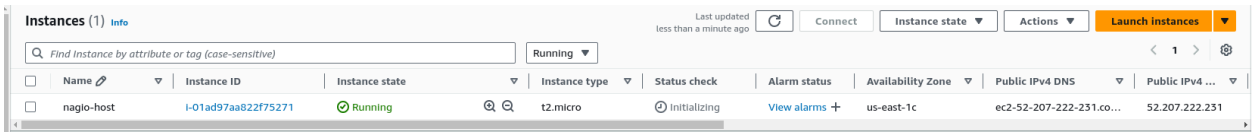
Aim: To Understand Continuous monitoring and Installation and configuration of Nagios Core,

Nagios Plugins and NRPE (Nagios Remote Plugin Executor) on Linux Machine.

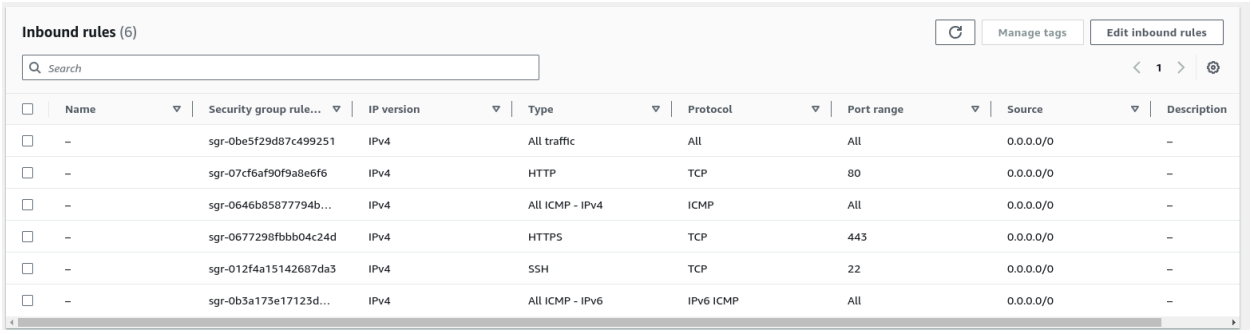
STEPS TO INSTALL AND SETUP NAGIOS ON EC-2 INSTANCE

1. Create an Amazon Linux EC-2 instance and select either existing key pair or create new





2. Under Security Group, make sure HTTP, HTTPS, SSH, ICMP are open from everywhere.



Inbound rules (6)

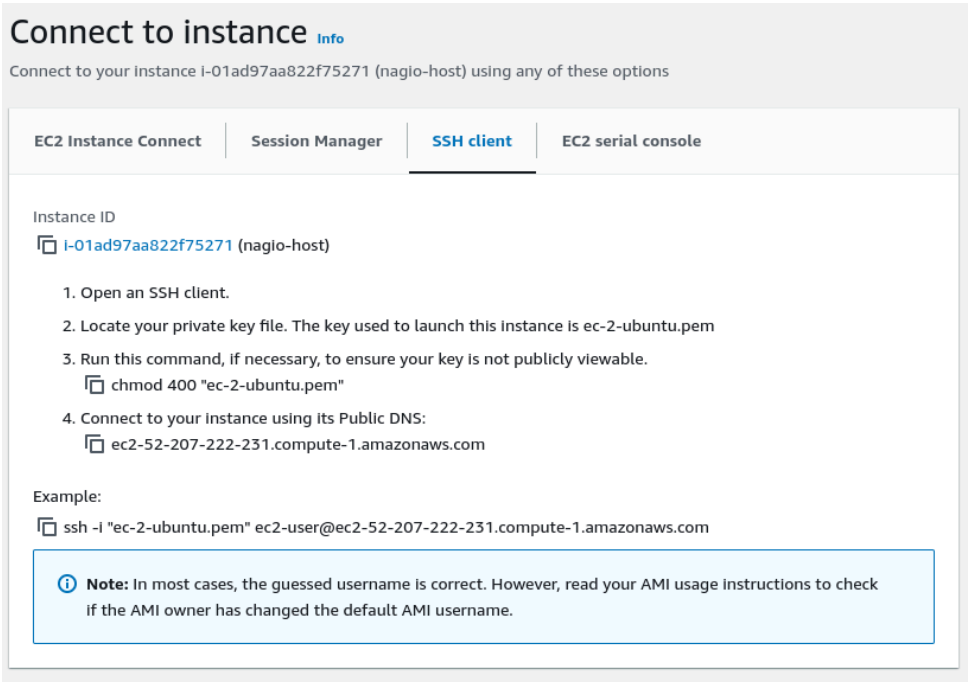
Search

Manage tags | Edit inbound rules

1

Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Description
–	sgr-0be5f29d87c499251	IPv4	All traffic	All	All	0.0.0.0/0	–
–	sgr-07cf6af90f9a8e6f6	IPv4	HTTP	TCP	80	0.0.0.0/0	–
–	sgr-0646b85877794b...	IPv4	All ICMP - IPv4	ICMP	All	0.0.0.0/0	–
–	sgr-0677298fbbb04c24d	IPv4	HTTPS	TCP	443	0.0.0.0/0	–
–	sgr-012f4a15142687da3	IPv4	SSH	TCP	22	0.0.0.0/0	–
–	sgr-0b3a173e17123d...	IPv4	All ICMP - IPv6	IPv6 ICMP	All	0.0.0.0/0	–

3. SSH into Your EC2 instance or simply use **EC2 Instance Connect** from the browser.



Connect to instance Info

Connect to your instance i-01ad97aa822f75271 (nagio-host) using any of these options

EC2 Instance Connect | Session Manager | SSH client | EC2 serial console

Instance ID

i-01ad97aa822f75271 (nagio-host)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is ec2-ubuntu.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
   chmod 400 "ec2-ubuntu.pem"
4. Connect to your instance using its Public DNS:
   ec2-52-207-222-231.compute-1.amazonaws.com

Example:

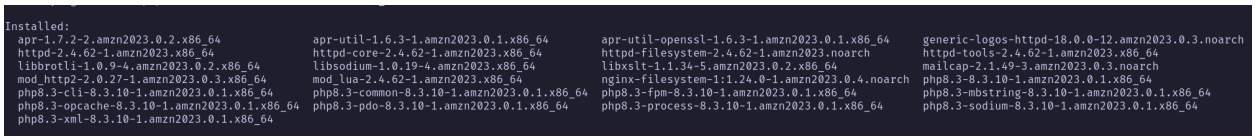
ssh -i "ec2-ubuntu.pem" ec2-user@ec2-52-207-222-231.compute-1.amazonaws.com

Note: In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

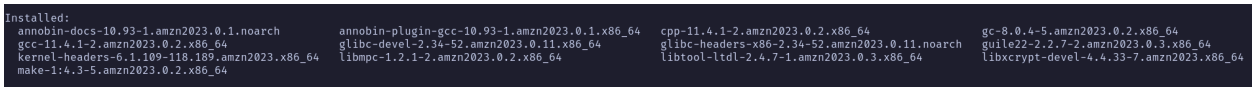
```
❯ ssh -i "ec2-ubuntu.pem" ec2-user@ec2-52-207-222-231.compute-1.amazonaws.com
The authenticity of host 'ec2-52-207-222-231.compute-1.amazonaws.com (52.207.222.231)' can't be established.
ED25519 key fingerprint is SHA256:/s4gfn/CHOBlXNdwi7aFEQPNXd+F3gfT+c8lsjldby8.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-52-207-222-231.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
X11 forwarding request failed on channel 0
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-31-33-179 ~]$
```

Now, we need to install necessary packages that are required to run nagios properly

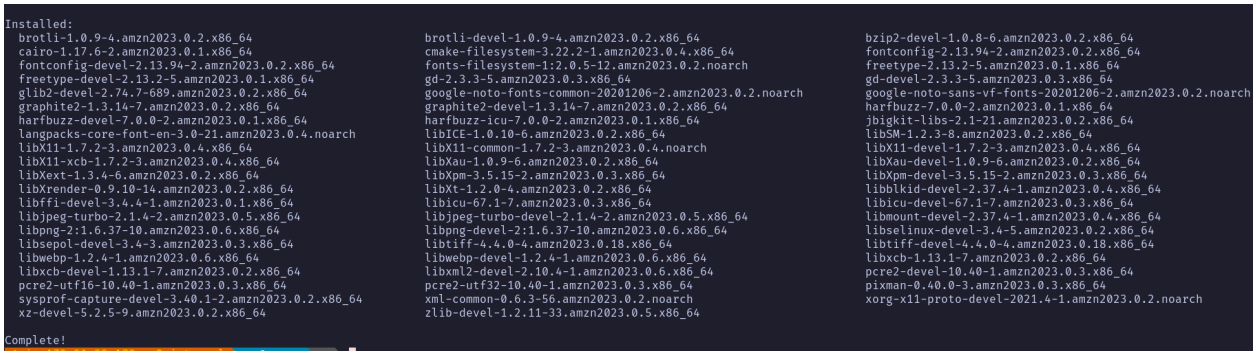
sudo yum install httpd php



sudo yum install gcc glibc glibc-common



sudo yum install gd gd-devel



4. Create a new Nagios User with its password. You'll have to enter the password twice for confirmation. This is require as to give separate permissions for nagios

   sudo adduser -m nagios

   sudo passwd nagios

```
ip-172-31-33-179.ec2.internal ec2-user ~ sudo passwd nagios
Changing password for user nagios.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
Sorry, passwords do not match.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: all authentication tokens updated successfully.
```

6. Create a new user group

sudo groupadd nagcmd

7. Use these commands so that you don't have to use sudo for Apache and Nagios

sudo usermod -a -G nagcmd nagios

sudo usermod -a -G nagcmd apache

```
ip-172-31-33-179.ec2.internal ec2-user ~ sudo usermod -a -G nagcmd nagios
sudo usermod -a -G nagcmd apache
```

8. Create a new directory for Nagios downloads

mkdir ~/downloads

cd ~/downloads

9. Use wget to download the installation source zip files.

wget https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.5.tar.gz

```
ip-172-31-33-179.ec2.internal ec2-user ~ downloads wget https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.5.tar.gz
--2024-10-04 13:40:59--  https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.5.tar.gz
Resolving assets.nagios.com (assets.nagios.com)... 45.79.49.120, 2600:3c00::f03c:92ff:fef7:45ce
Connecting to assets.nagios.com (assets.nagios.com)|45.79.49.120|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2065473 (2.0M) [application/x-gzip]
Saving to: 'nagios-4.5.5.tar.gz'

nagios-4.5.5.tar.gz                 100%[=============================================================================================================>]

2024-10-04 13:41:00 (4.15 MB/s) - 'nagios-4.5.5.tar.gz' saved [2065473/2065473]
```

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

```
ip-172-31-33-179.ec2.internal ec2-user ~ downloads wget https://nagios-plugins.org/download/nagios-plugins-2.4.11.tar.gz
--2024-10-04 13:41:25--  https://nagios-plugins.org/download/nagios-plugins-2.4.11.tar.gz
Resolving nagios-plugins.org (nagios-plugins.org)... 45.56.123.251
Connecting to nagios-plugins.org (nagios-plugins.org)|45.56.123.251|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2753049 (2.6M) [application/x-gzip]
Saving to: 'nagios-plugins-2.4.11.tar.gz'

nagios-plugins-2.4.11.tar.gz                  100%[=========================================================================================================>]

2024-10-04 13:41:26 (7.55 MB/s) - 'nagios-plugins-2.4.11.tar.gz' saved [2753049/2753049]
```

10. Use tar to unzip the downloaded archive and change to that directory.

tar zxvf nagios-4.5.5.tar.gz

```
ip-172-31-33-179.ec2.internal ec2-user ~ downloads tar zxvf nagios-4.5.5.tar.gz
nagios-4.5.5/
nagios-4.5.5/.github/
nagios-4.5.5/.github/workflows/
nagios-4.5.5/.github/workflows/test.yml
nagios-4.5.5/.gitignore
nagios-4.5.5/CONTRIBUTING.md
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
```

11. Navigate to the extracted folder and Run the configuration script with the same group name you previously created.

./configure --with-command-group=nagcmd

```
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
```

We got error, because ssl headers library is not installed

It can be installed using **sudo yum install openssl-devel**

```
ip-172-31-33-179.ec2.internal ec2-user ~ downloads nagios-4.5.5 sudo yum install openssl-devel
Last metadata expiration check: 0:28:57 ago on Fri Oct  4 13:16:25 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
```

Now rerun

./configure --with-command-group=nagcmd

```
Creating sample config files in sample-config/ ...


*** Configuration summary for nagios 4.5.5 2024-09-17 ***:

 General Options:
 -------------------------
        Nagios executable:  nagios
        Nagios user/group:  nagios,nagios
       Command user/group:  nagios,nagcmd
             Event Broker:  yes
        Install ${prefix}:  /usr/local/nagios
    Install ${includedir}:  /usr/local/nagios/include/nagios
                Lock file:  /run/nagios.lock
   Check result directory:  /usr/local/nagios/var/spool/checkresults
           Init directory:  /lib/systemd/system
  Apache conf.d directory:  /etc/httpd/conf.d
             Mail program:  /bin/mail
                  Host OS:  linux-gnu
          IOBroker Method:  epoll

 Web Interface Options:
 ------------------------
                 HTML URL:  http://localhost/nagios/
                  CGI URL:  http://localhost/nagios/cgi-bin/
 Traceroute (used by WAP):  /usr/bin/traceroute


Review the options above for accuracy.  If they look okay,
type 'make all' to compile the main program and CGIs.
```

12. Compile the source code.

make all

```
ip-172-31-33-179.ec2.internal ec2-user ~ downloads nagios-4.5.5 make all
cd ./base && make
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/base'
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o nagios.o ./nagios.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o broker.o broker.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o nebmods.o nebmods.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o ../common/shared.o ../common/shared.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o query-handler.o query-handler.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o workers.o workers.c
In function 'get_wproc_list',
```

13. Install binaries, init script and sample config files. Lastly, set permissions on the external

command directory.

sudo make install

sudo make install-init

sudo make install-config

sudo make install-commandmode

```
ip-172-31-33-179.ec2.internal ec2-user ~ downloads 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
```

```
ip-172-31-33-179.ec2.internal ec2-user ~ downloads 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
```

```
ip-172-31-33-179.ec2.internal ec2-user ~ downloads nagios-4.5.5 sudo make install-config
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc/objects
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/nagios.cfg /usr/local/nagios/etc/nagios.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/cgi.cfg /usr/local/nagios/etc/cgi.cfg
/usr/bin/install -c -b -m 660 -o nagios -g nagios sample-config/resource.cfg /usr/local/nagios/etc/resource.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/templates.cfg /usr/local/nagios/etc/objects/templates.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/commands.cfg /usr/local/nagios/etc/objects/commands.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/contacts.cfg /usr/local/nagios/etc/objects/contacts.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/timeperiods.cfg /usr/local/nagios/etc/objects/timeperiods.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/localhost.cfg /usr/local/nagios/etc/objects/localhost.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/windows.cfg /usr/local/nagios/etc/objects/windows.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/printer.cfg /usr/local/nagios/etc/objects/printer.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/switch.cfg /usr/local/nagios/etc/objects/switch.cfg
```

```
ip-172-31-33-179.ec2.internal ec2-user ~ downloads nagios-4.5.5 sudo make install-commandmode
/usr/bin/install -c -m 775 -o nagios -g nagcmd -d /usr/local/nagios/var/rw
chmod g+s /usr/local/nagios/var/rw

*** External command directory configured ***
```

14. Edit the config file and change the email address so that we can receive timely alerts about the status of our system.

sudo nano /usr/local/nagios/etc/objects/contacts.cfg

```
define contact {

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


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

And change email with your email

15. Configure the web interface. This is used to set up web server configuration of nagios dashboard.

sudo make install-webconf

```
ip-172-31-33-179.ec2.internal ec2-user ~ downloads 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 ***
```

16. Create a nagiosadmin account for nagios login along with password. You'll have to specify the password twice.

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

```
ip-172-31-33-179.ec2.internal ec2-user ~ downloads nagios-4.5.5 sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
New password:
Re-type new password:
Adding password for user nagiosadmin
```

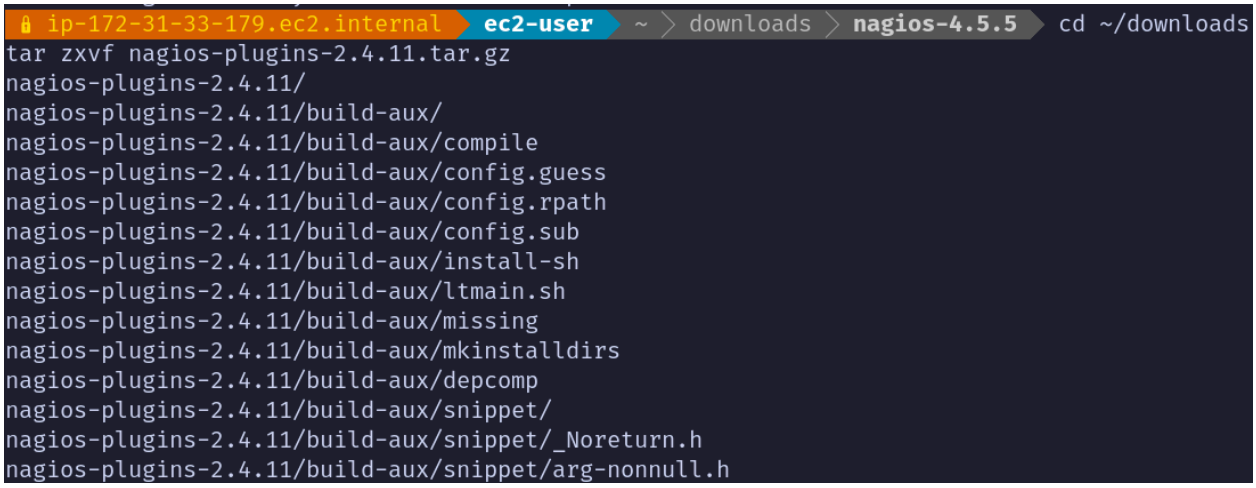
17. Restart Apache

sudo service httpd restart

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

cd ~/downloads

tar zxvf nagios-plugins-2.4.11.tar.gz



19. Compile and install plugins

cd nagios-plugins-2.4.11

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

```
checking that generated files are newer than configure... done
configure: creating ./config.status
config.status: creating gl/Makefile
config.status: creating nagios-plugins.spec
config.status: creating tools/build_perl_modules
config.status: creating Makefile
config.status: creating tap/Makefile
config.status: creating lib/Makefile
config.status: creating plugins/Makefile
config.status: creating lib/tests/Makefile
config.status: creating plugins-root/Makefile
config.status: creating plugins-scripts/Makefile
config.status: creating plugins-scripts/utils.pm
config.status: creating plugins-scripts/utils.sh
config.status: creating perlmods/Makefile
config.status: creating test.pl
config.status: creating pkg/solaris/pkginfo
config.status: creating po/Makefile.in
config.status: creating config.h
config.status: config.h is unchanged
config.status: executing depfiles commands
config.status: executing libtool commands
config.status: executing po-directories commands
config.status: creating po/POTFILES
config.status: creating po/Makefile
```

make

sudo make install

```
ip-172-31-33-179.ec2.internal ec2-user ~ downloads nagios-plugins-2.4.11 sudo make install
Making install in gl
make[1]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11/gl'
make  install-recursive
make[2]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11/gl'
make[3]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11/gl'
make[4]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11/gl'
if test yes = no; then \
  case 'linux-gnu' in \
    darwin[56]*) \
      need_charset_alias=true ;; \
    darwin* | cygwin* | mingw* | pw32* | cegcc*) \
      need_charset_alias=false ;; \
    *) \
      need_charset_alias=true ;; \
  esac ; \
else \
  need_charset_alias=false ; \
fi ; \
if $need_charset_alias; then \
  /bin/sh ../build-aux/mkinstalldirs /usr/local/nagios/lib ; \
fi ; \
```

20. Start Nagios and Add Nagios to the list of system services

sudo chkconfig --add nagios

sudo chkconfig nagios on

```
ip-172-31-33-179.ec2.internal ec2-user ~ downloads nagios-plugins-2.4.11 sudo chkconfig --add nagios
error reading information on service nagios: No such file or directory
ip-172-31-33-179.ec2.internal ec2-user ~ downloads nagios-plugins-2.4.11 sudo chkconfig nagios on
Note: Forwarding request to 'systemctl enable nagios.service'.
Created symlink /etc/systemd/system/multi-user.target.wants/nagios.service → /usr/lib/systemd/system/nagios.service.
```

Verify the sample configuration files

```
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
```

As we can see no errors were detected

sudo service nagios start

```
ip-172-31-33-179.ec2.internal ec2-user ~ downloads nagios-plugins-2.4.11 sudo service nagios start
Redirecting to /bin/systemctl start nagios.service
```

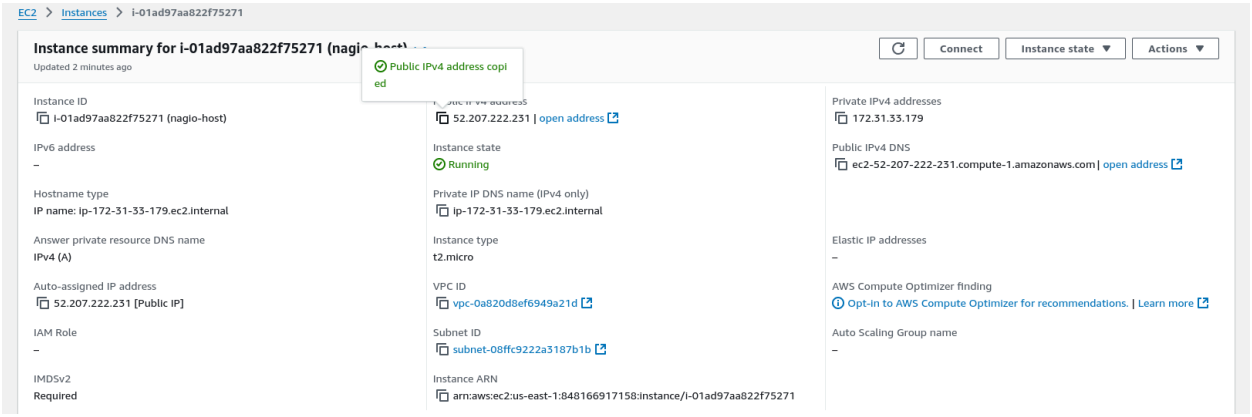
21. Check the status of Nagios

sudo systemctl status nagios

```
ip-172-31-33-179.ec2.internal ec2-user ~ downloads nagios-plugins-2.4.11 sudo systemctl status nagios
● nagios.service - Nagios Core 4.5.5
     Loaded: loaded (/usr/lib/systemd/system/nagios.service; enabled; preset: disabled)
     Active: active (running) since Fri 2024-10-04 14:05:21 UTC; 33s ago
       Docs: https://www.nagios.org/documentation
    Process: 67432 ExecStartPre=/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
    Process: 67433 ExecStart=/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
   Main PID: 67434 (nagios)
      Tasks: 6 (limit: 1112)
     Memory: 5.6M
        CPU: 67ms
     CGroup: /system.slice/nagios.service
             ├─67434 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─67438 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─67439 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─67440 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─67441 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             └─67446 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg

Oct 04 14:05:21 ip-172-31-33-179.ec2.internal nagios[67434]: qh: Socket '/usr/local/nagios/var/rw/nagios.qh' successfully initialized
Oct 04 14:05:21 ip-172-31-33-179.ec2.internal nagios[67434]: qh: core query handler registered
Oct 04 14:05:21 ip-172-31-33-179.ec2.internal nagios[67434]: qh: echo service query handler registered
Oct 04 14:05:21 ip-172-31-33-179.ec2.internal nagios[67434]: qh: help for the query handler registered
Oct 04 14:05:21 ip-172-31-33-179.ec2.internal nagios[67434]: wproc: Successfully registered manager as @wproc with query handler
Oct 04 14:05:21 ip-172-31-33-179.ec2.internal nagios[67434]: wproc: Registry request: name=Core Worker 67441;pid=67441
Oct 04 14:05:21 ip-172-31-33-179.ec2.internal nagios[67434]: wproc: Registry request: name=Core Worker 67440;pid=67440
Oct 04 14:05:21 ip-172-31-33-179.ec2.internal nagios[67434]: wproc: Registry request: name=Core Worker 67438;pid=67438
Oct 04 14:05:21 ip-172-31-33-179.ec2.internal nagios[67434]: wproc: Registry request: name=Core Worker 67439;pid=67439
Oct 04 14:05:22 ip-172-31-33-179.ec2.internal nagios[67434]: Successfully launched command file worker with pid 67446
```

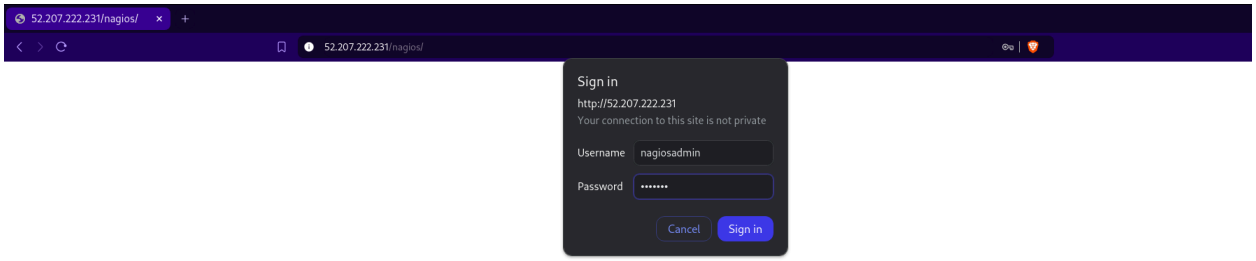
The nagios service is running and working normally

22. Go back to EC2 Console and copy the Public IP address of this instance

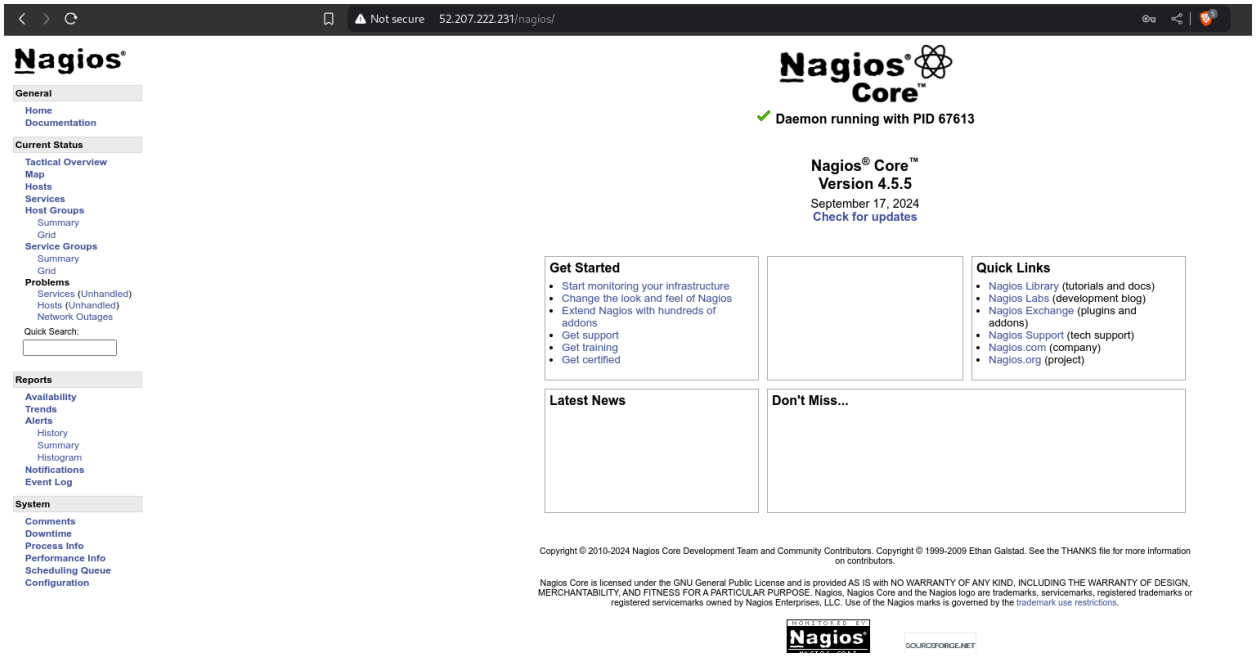


23. Open up your browser and look for http://<your_public_ip_address>/nagios

Enter username as nagiosadmin and password which we had set previously

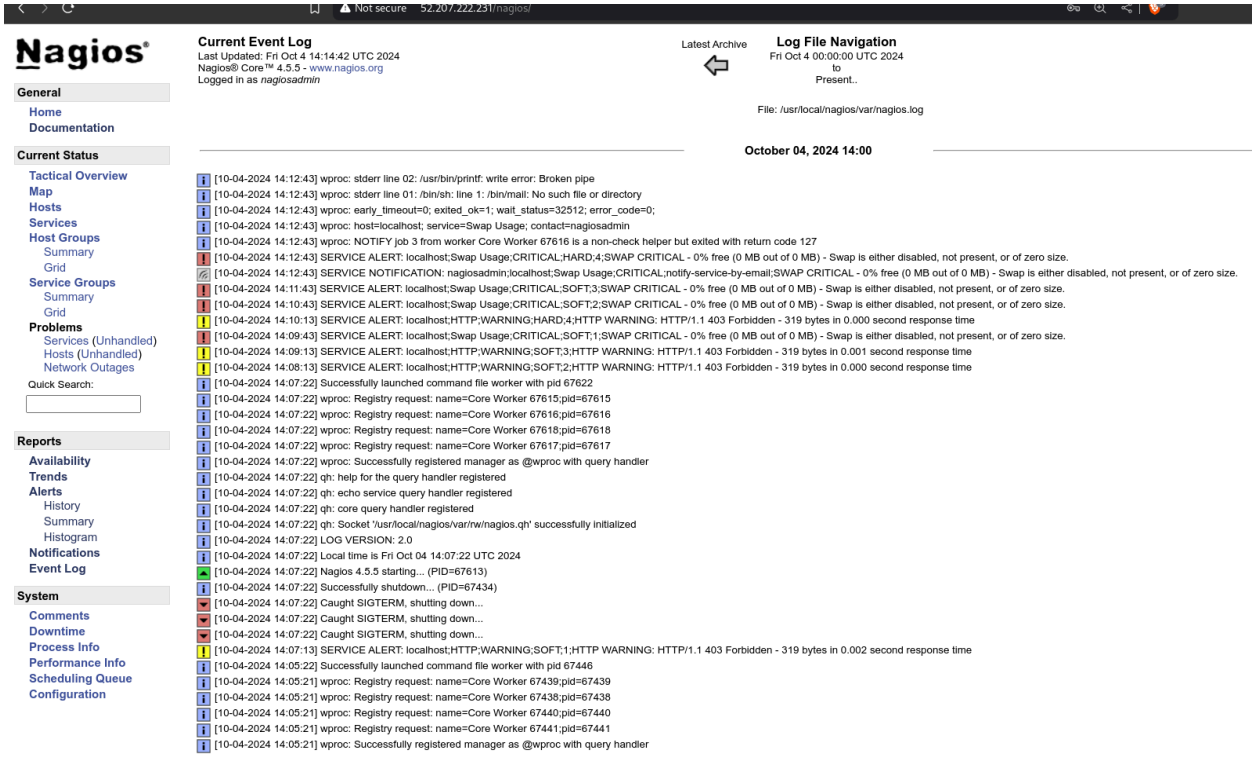


24. After entering the correct credentials, you will see the Home page of Nagios.

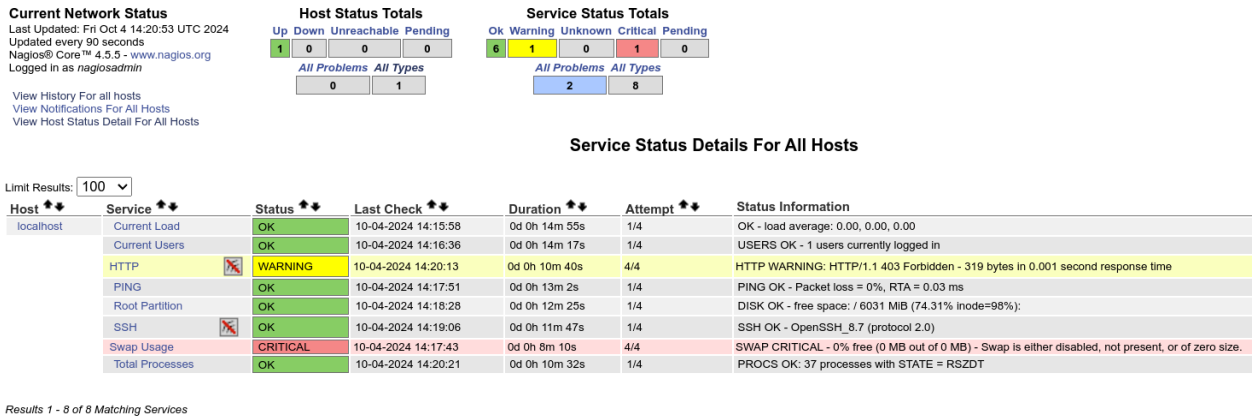


This means that Nagios was correctly installed and configured with its plugins so far

Name: Alok Yadav Div: D15C Roll No.: 59



Now we can see system logs using nagios



Above is the status of all the services running on Host Machine

Conclusion:

We began the experiment by installing all the necessary packages required for Nagios. Next, we created a new user and group for Nagios, followed by installing the Nagios software through local compilation. Proper compilation is crucial to avoid errors during operation. After installation, we started both the `httpd` and Nagios services, enabling access to the Nagios dashboard, where critical system information is displayed.

Name: Alok Yadav Div: D15C Roll No.: 59

EXPERIMENT NO. 10

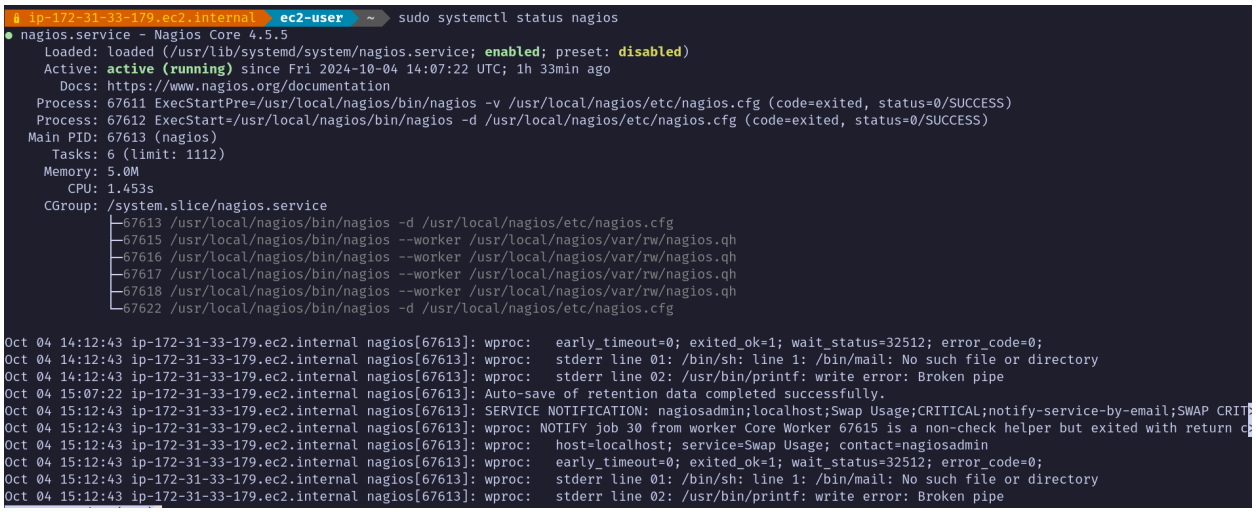
Aim: To perform Port, Service monitoring, Windows/Linux server monitoring using Nagios.

Prerequisites:
Nagios Server running on Amazon Linux Machine.

STEPS:

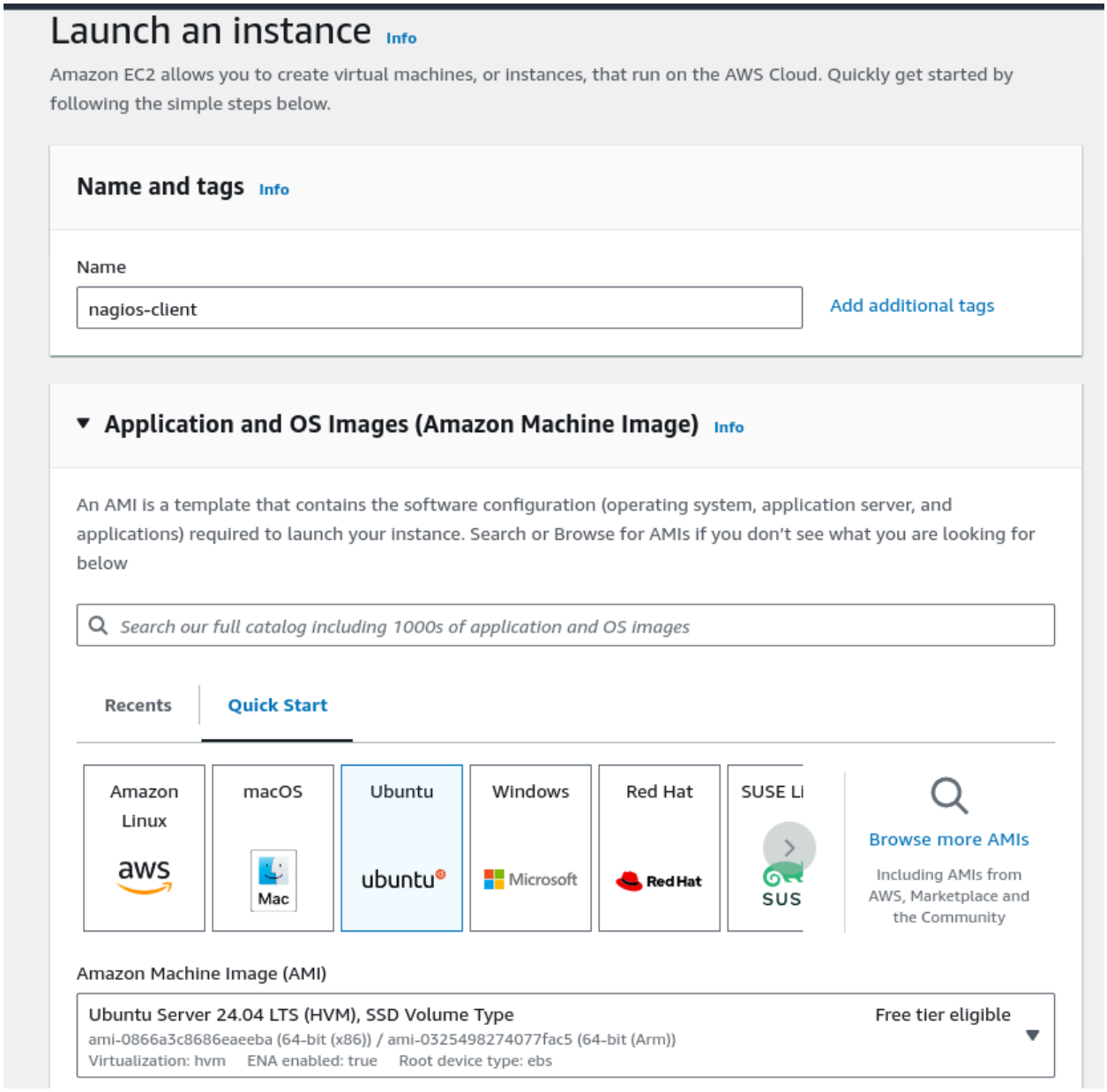
1. To Confirm Nagios is running on the server side Perform the following command on your Amazon Linux Machine (Nagios-host).

   sudo systemctl status nagios

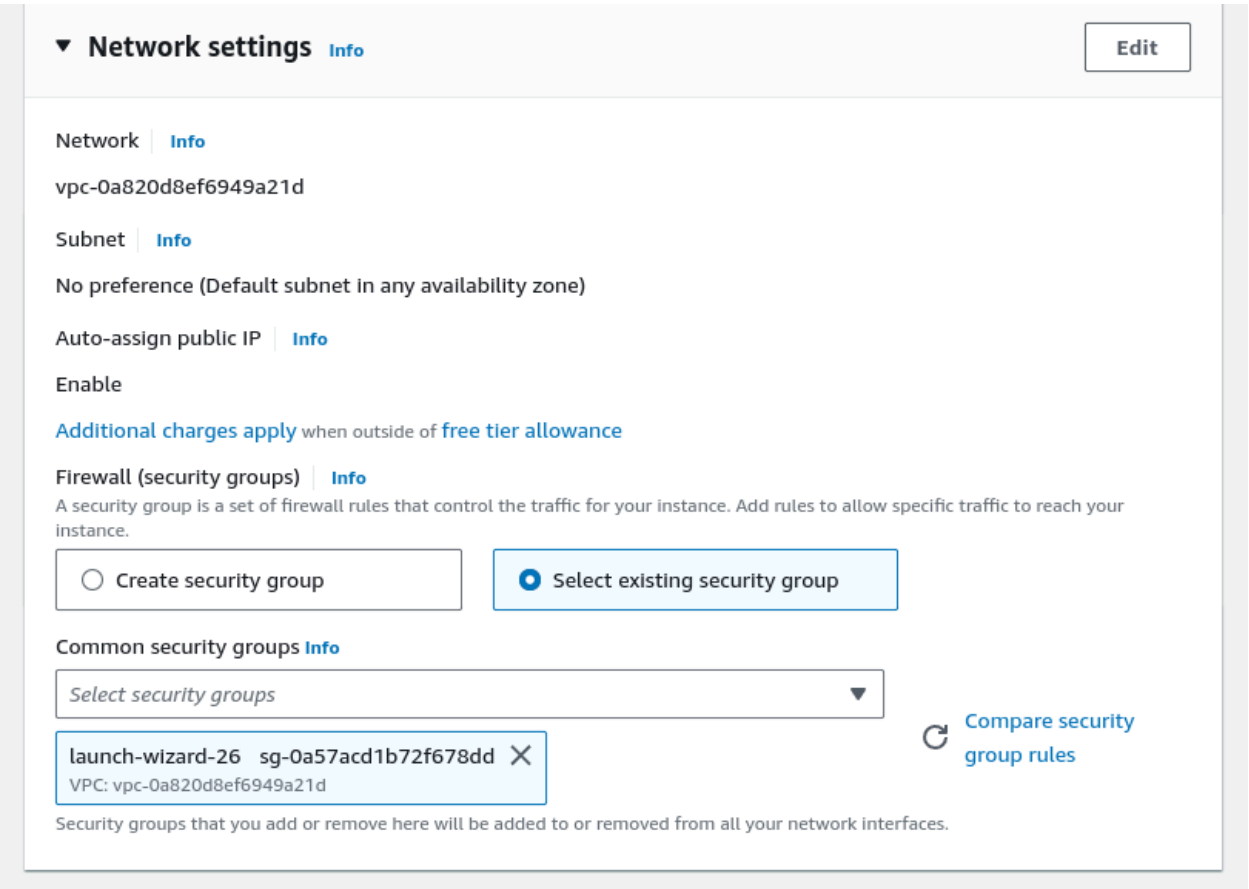


   As nagios service on main machine is running, we can proceed further

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



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).



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

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

ps -ef | grep nagios

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

```
ip-172-31-33-179.ec2.internal ec2-user ~ sudo su
[root@ip-172-31-33-179 ec2-user]# mkdir /usr/local/nagios/etc/objects/monitorhosts
[root@ip-172-31-33-179 ec2-user]# mkdir /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
[root@ip-172-31-33-179 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

7. Open linuxserver.cfg using nano and make the following changes in all Positions everywhere in file

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

Change hostname to **linuxserver.**
Change address to the **public IP of your Linux client**.
Set hostgroup_name to **linux-servers1**.

```
# Define a host for the local machine

define host {

    use                     linux-server            ; Name of host template to use
                                                    ; This host definition will inherit all variables that are defined
                                                    ; in (or inherited by) the linux-server host template definition.
    host_name               linuxserver
    alias                   localhost
    address                 34.204.79.231
}



###############################################################################
#
# HOST GROUP DEFINITION
#
###############################################################################

# Define an optional hostgroup for Linux machines

define hostgroup {

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

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/

nano /usr/local/nagios/etc/nagios.cfg

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

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

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

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

Total Warnings: 0
Total Errors:   0

Things look okay - No serious problems were detected during the pre-flight check
```

We got no errors and warnings for current configuration

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

```
[root@ip-172-31-33-179 ec2-user]# service nagios restart
Redirecting to /bin/systemctl restart nagios.service
```

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

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

```
ubuntu@ip-172-31-45-81:~$ sudo apt update -y
sudo apt install gcc -y
sudo apt install -y nagios-nrpe-server nagios-plugins
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:7 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [382 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Components [35.0 kB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 c-n-f Metadata [8328 B]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [537 kB]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [132 kB]
Get:16 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 c-n-f Metadata [8860 B]
Get:17 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [384 kB]
```

```
Creating config file /etc/nagios-plugins/config/netware.cfg with new version

Creating config file /etc/nagios-plugins/config/nt.cfg with new version

Creating config file /etc/nagios-plugins/config/pgsql.cfg with new version

Creating config file /etc/nagios-plugins/config/radius.cfg with new version

Creating config file /etc/nagios-plugins/config/rpc-nfs.cfg with new version

Creating config file /etc/nagios-plugins/config/snmp.cfg with new version
Setting up monitoring-plugins (2.3.5-1ubuntu3) ...
Setting up libldb2:amd64 (2:2.8.0+samba4.19.5+dfsg-4ubuntu9) ...
Setting up libavahi-client3:amd64 (0.8-13ubuntu6) ...
Setting up samba-libs:amd64 (2:4.19.5+dfsg-4ubuntu9) ...
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...
```

12. Open nrpe.cfg file to make changes.Under allowed_hosts, add your nagios host public IP address.

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

allowed_hosts=127.0.0.1,::1,52.207.222.231
```

13. Now restart the NRPE server by this command.

sudo systemctl restart nagios-nrpe-server

```
ubuntu@ip-172-31-45-81:~$ sudo nano /etc/nagios/nrpe.cfg
ubuntu@ip-172-31-45-81:~$ sudo systemctl restart nagios-nrpe-server
ubuntu@ip-172-31-45-81:~$
```

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-172-31-33-179 ec2-user]# sudo systemctl status nagios
● nagios.service - Nagios Core 4.5.5
     Loaded: loaded (/usr/lib/systemd/system/nagios.service; enabled; preset: disabled)
     Active: active (running) since Fri 2024-10-04 16:01:14 UTC; 6min ago
       Docs: https://www.nagios.org/documentation
    Process: 74405 ExecStartPre=/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
    Process: 74406 ExecStart=/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
   Main PID: 74407 (nagios)
      Tasks: 6 (limit: 1112)
     Memory: 4.2M
        CPU: 161ms
     CGroup: /system.slice/nagios.service
             ├─74407 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─74408 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─74409 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─74410 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─74411 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             └─74412 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
```

sudo systemctl status httpd

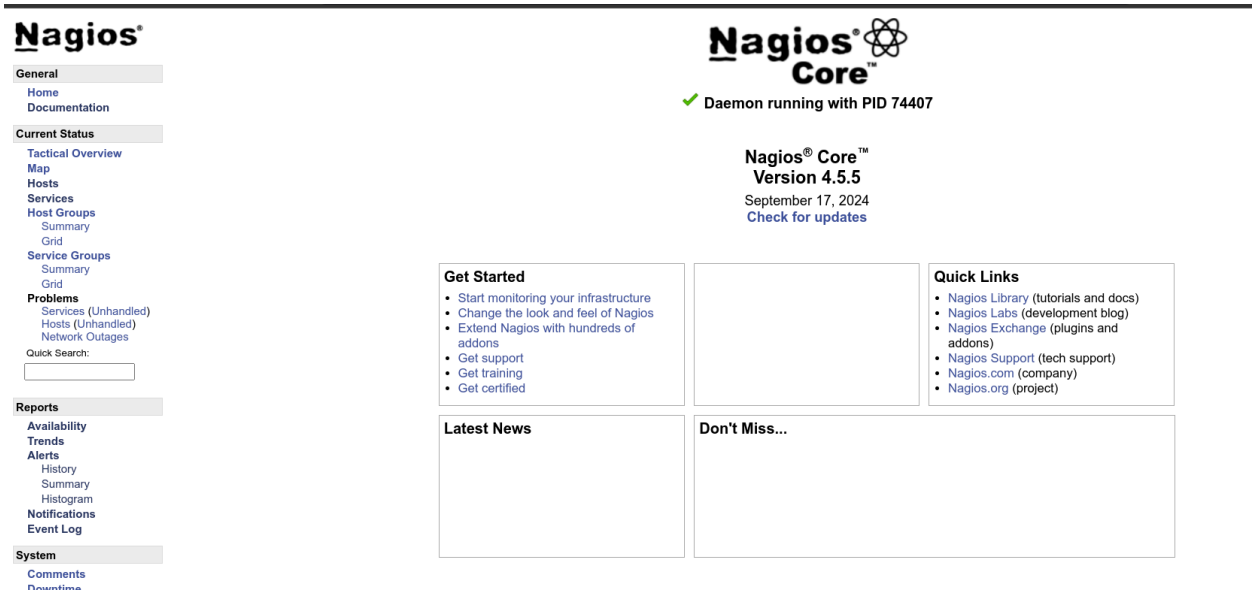
```
[root@ip-172-31-33-179 ec2-user]# sudo systemctl status httpd
● httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset: disabled)
    Drop-In: /usr/lib/systemd/system/httpd.service.d
             └─php-fpm.conf
     Active: active (running) since Fri 2024-10-04 13:56:26 UTC; 2h 11min ago
       Docs: man:httpd.service(8)
   Main PID: 52059 (httpd)
     Status: "Total requests: 220; Idle/Busy workers 100/0;Requests/sec: 0.0279; Bytes served/sec: 122 B/sec"
      Tasks: 230 (limit: 1112)
     Memory: 23.7M
        CPU: 5.641s
     CGroup: /system.slice/httpd.service
             ├─52059 /usr/sbin/httpd -DFOREGROUND
             ├─52061 /usr/sbin/httpd -DFOREGROUND
             ├─52065 /usr/sbin/httpd -DFOREGROUND
             ├─52066 /usr/sbin/httpd -DFOREGROUND
             ├─52067 /usr/sbin/httpd -DFOREGROUND
             └─67725 /usr/sbin/httpd -DFOREGROUND

Oct 04 13:56:25 ip-172-31-33-179.ec2.internal systemd[1]: Starting httpd.service - The Apache HTTP Server...
Oct 04 13:56:26 ip-172-31-33-179.ec2.internal systemd[1]: Started httpd.service - The Apache HTTP Server.
Oct 04 13:56:26 ip-172-31-33-179.ec2.internal httpd[52059]: Server configured, listening on: port 80
[root@ip-172-31-33-179 ec2-user]#
```

Both nagios and httpd service is running fine on host system

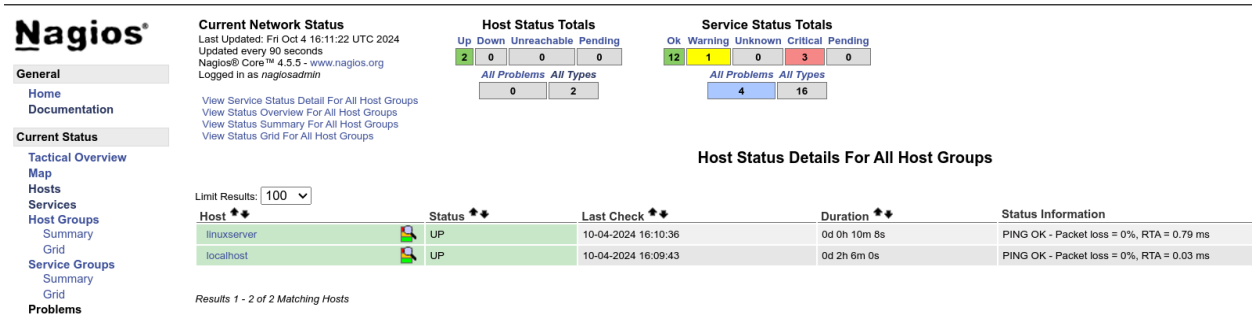
15. Now to check Nagios dashboard go to http://<nagios host ip>/nagios
Eg. http://34.207.68.187/nagios

Enter username as nagiosadmin and password which you set in Exp 9 if prompted.

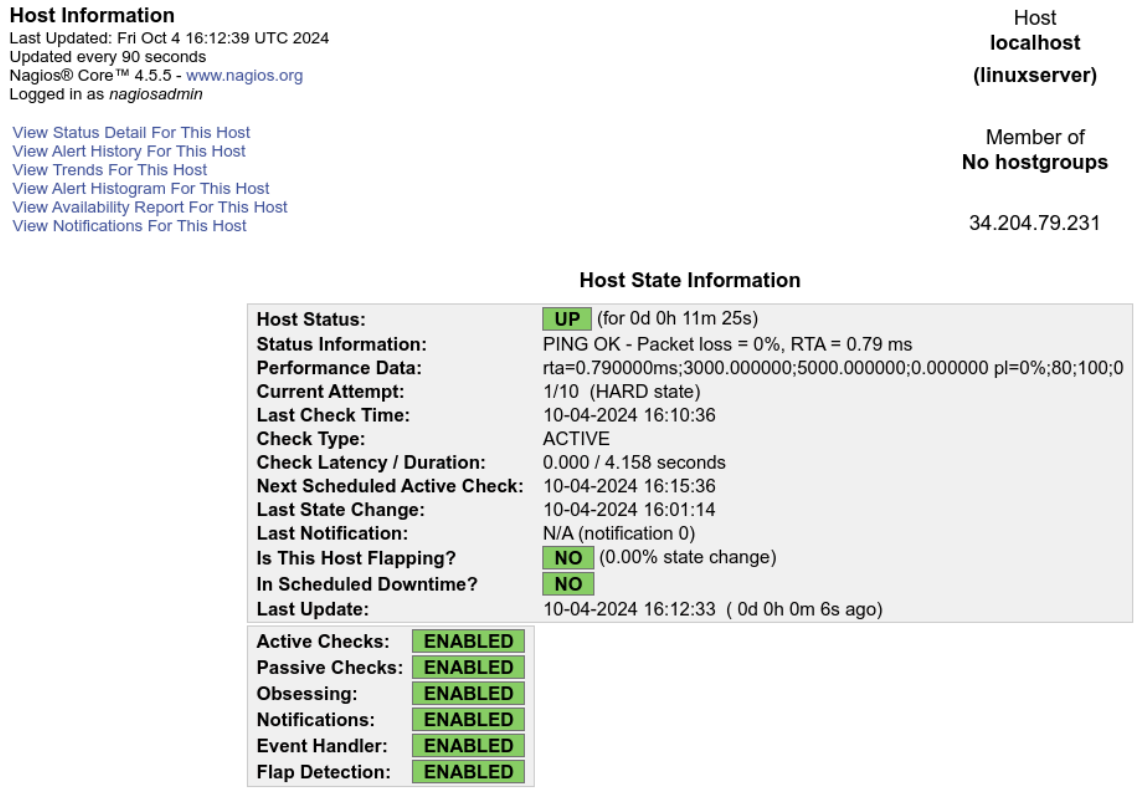
Name: Alok Yadav Div: D15C Roll No.: 59



Now Click on Hosts from left side panel



Our nagios client is showing up on nagios host dashboard



Host Information
Last Updated: Fri Oct 4 16:12:39 UTC 2024
Updated every 90 seconds
Nagios® Core™ 4.5.5 - www.nagios.org
Logged in as *nagiosadmin*

View Status Detail For This Host
View Alert History For This Host
View Trends For This Host
View Alert Histogram For This Host
View Availability Report For This Host
View Notifications For This Host

Host
localhost
(linuxserver)

Member of
No hostgroups

34.204.79.231

Host State Information

Host Status:	UP (for 0d 0h 11m 25s)
Status Information:	PING OK - Packet loss = 0%, RTA = 0.79 ms
Performance Data:	rta=0.790000ms;3000.000000;5000.000000;0.000000 pl=0%;80;100;0
Current Attempt:	1/10 (HARD state)
Last Check Time:	10-04-2024 16:10:36
Check Type:	ACTIVE
Check Latency / Duration:	0.000 / 4.158 seconds
Next Scheduled Active Check:	10-04-2024 16:15:36
Last State Change:	10-04-2024 16:01:14
Last Notification:	N/A (notification 0)
Is This Host Flapping?	NO (0.00% state change)
In Scheduled Downtime?	NO
Last Update:	10-04-2024 16:12:33 (0d 0h 0m 6s ago)

Active Checks:	ENABLED
Passive Checks:	ENABLED
Obsessing:	ENABLED
Notifications:	ENABLED
Event Handler:	ENABLED
Flap Detection:	ENABLED

Here we can see current status of nagios client mahine which is up and running fine

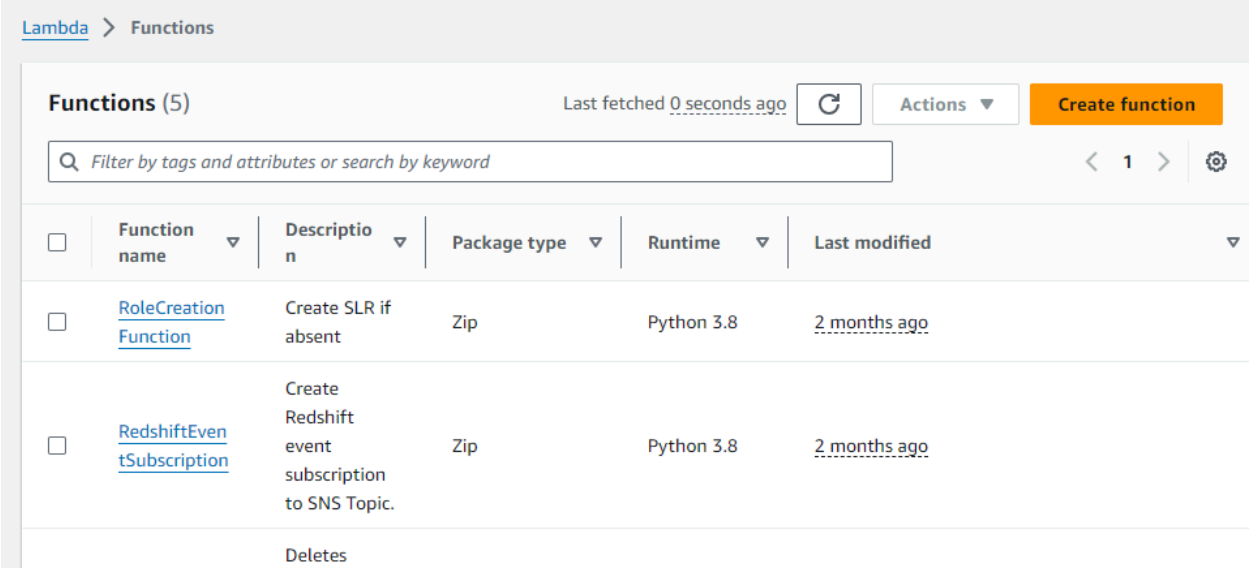
Conclusion:
In this experiment, we created a new EC2 Linux instance and set up the Nagios client on it. We then connected the client to the Nagios host machine, allowing us to monitor alerts and status for both the host and clients on a single dashboard. It is important to configure the Nagios client with the correct host IP, as failure to do so will prevent the connection. After completing the setup, we were able to successfully view the system health details of both the client and host on the unified Nagios dashboard.

EXPERIMENT NO.11

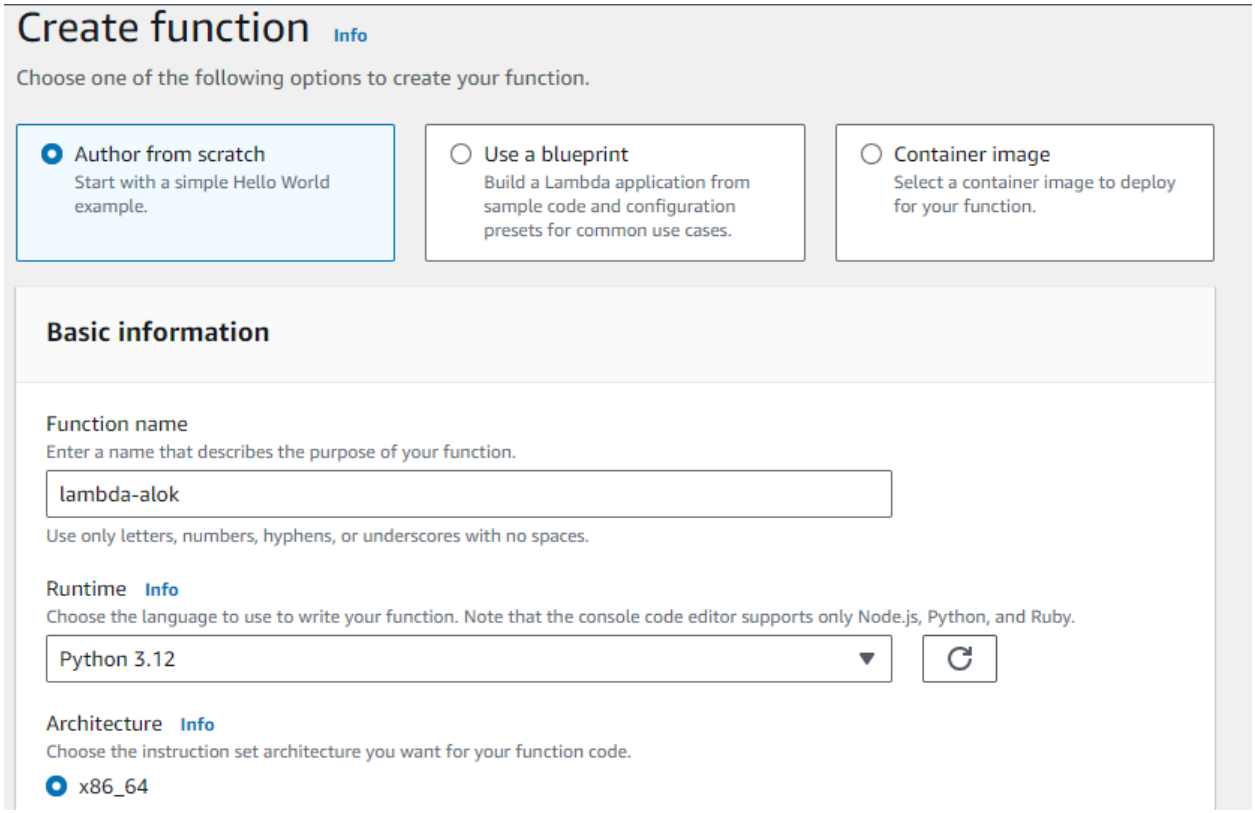
Aim: To understand AWS Lambda, its workflow, various functions and create your first Lambda functions using Python / Java / Nodejs.

Steps to create Lambda function in AWS :

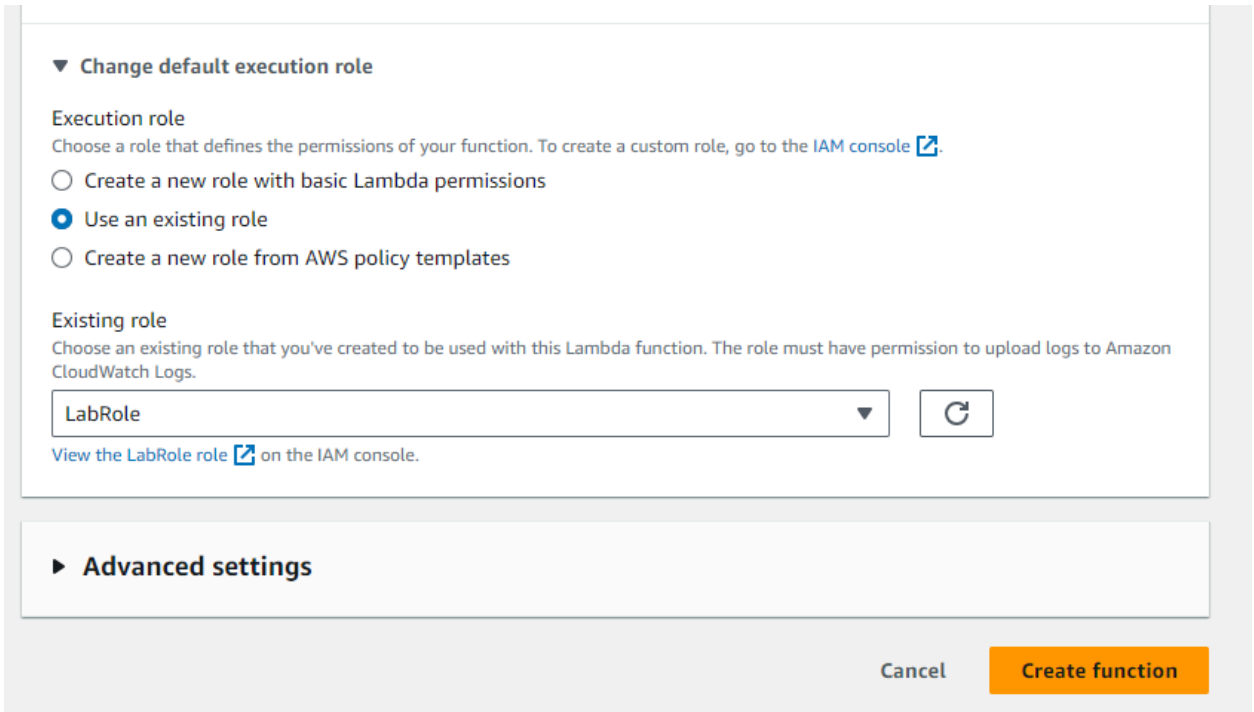
1. Open up the Lambda Console and click on the Create button.
   Be mindful of where you create your functions since Lambda is region-dependent.

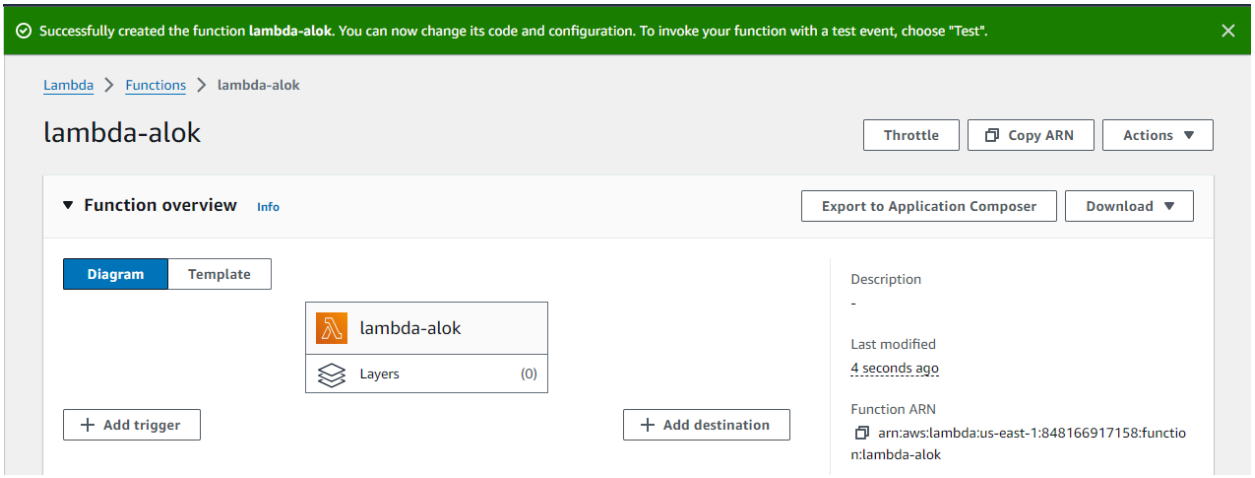


2. You can either create a function from scratch or select a blueprint, which is a pre-defined template by AWS that includes configuration settings for common use cases. Next, choose a runtime environment for your function; the dropdown will display all supported options, with Python, Node.js, .NET, and Java being the most popular. Finally, if you don't have an existing role, opt to create a new role with basic Lambda permissions.



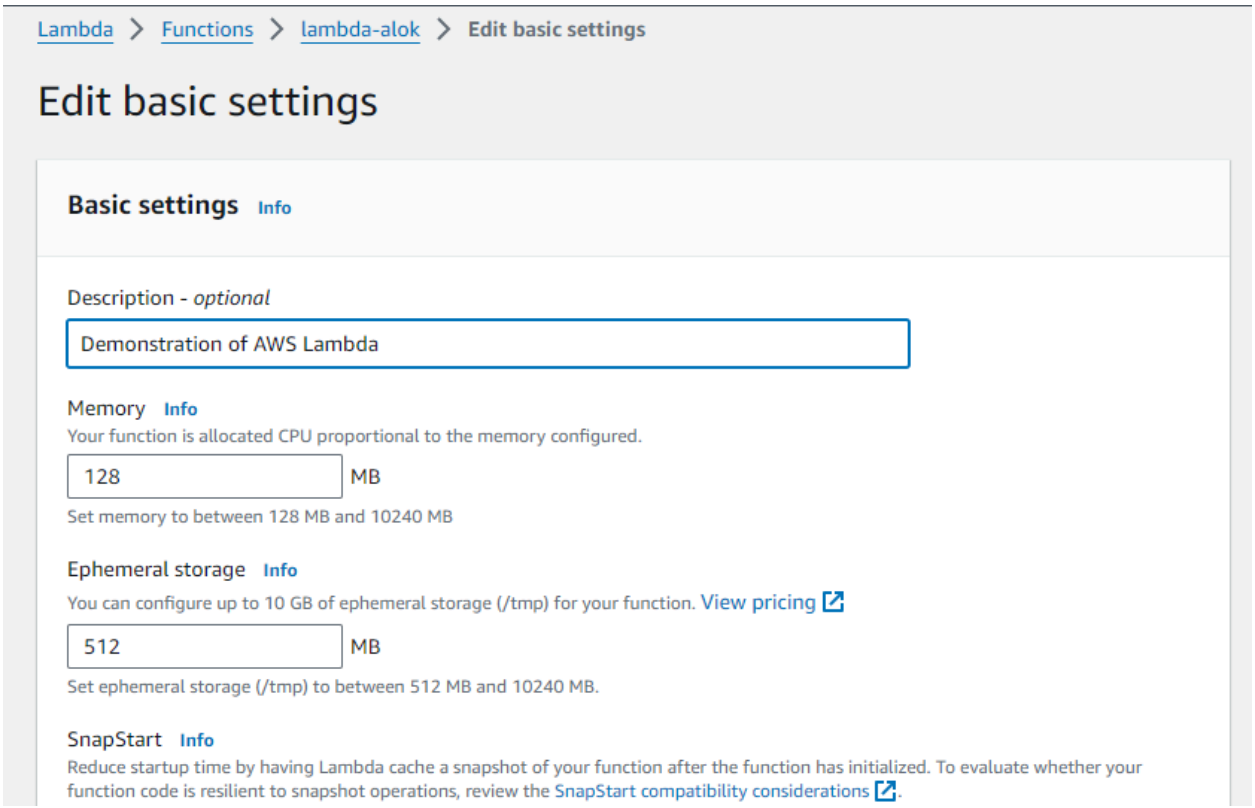
Select proper Execution role





Successfully created Lambda function

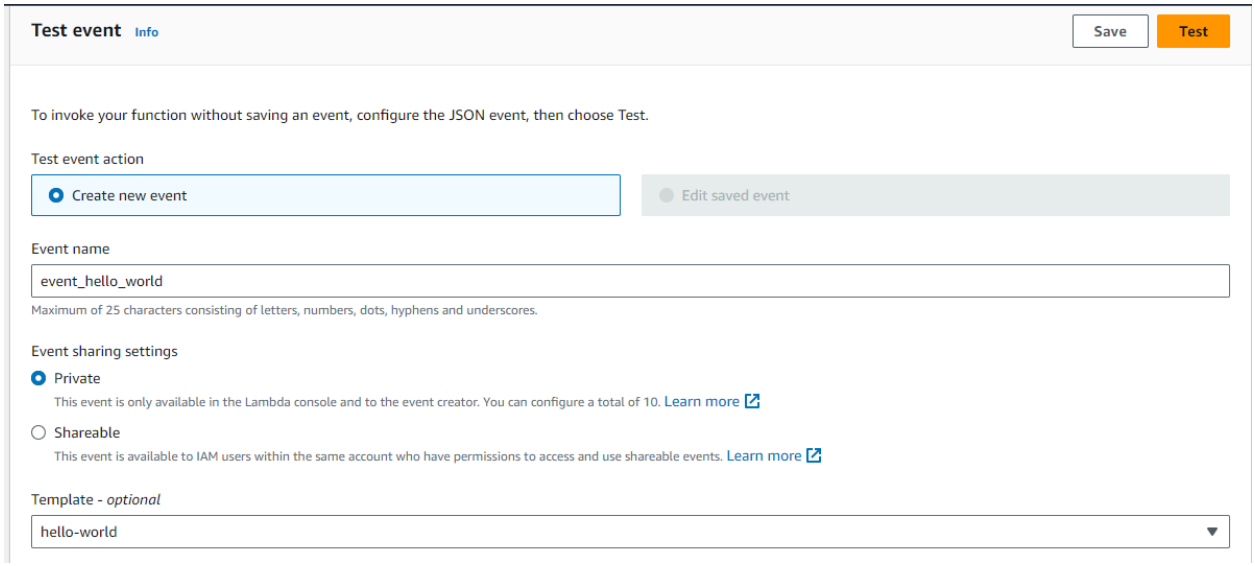
3. To view or change the basic settings, go to the 'Configuration' tab and click 'Edit' under 'General settings.' (THIS STEP IS OPTIONAL)



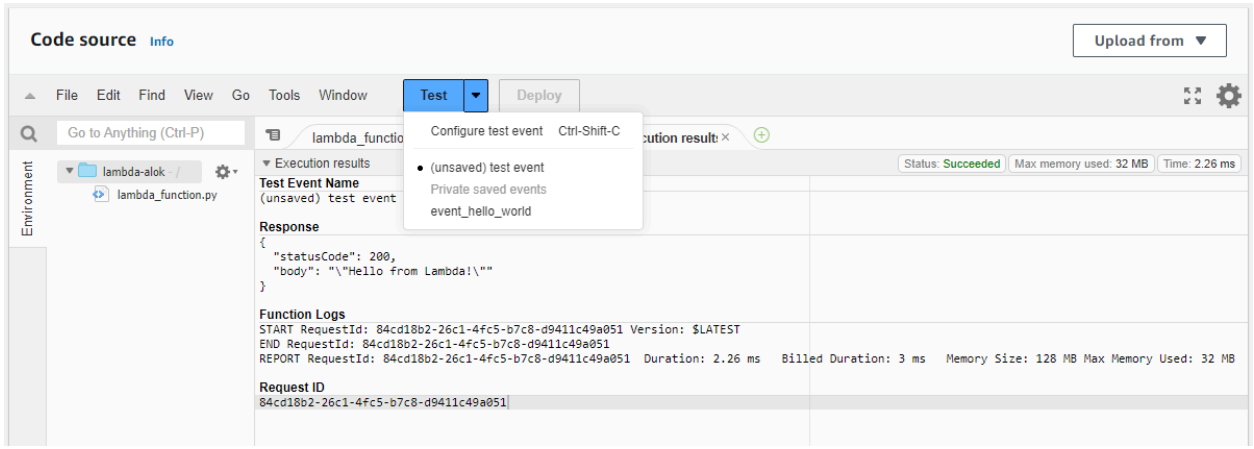
Added description.

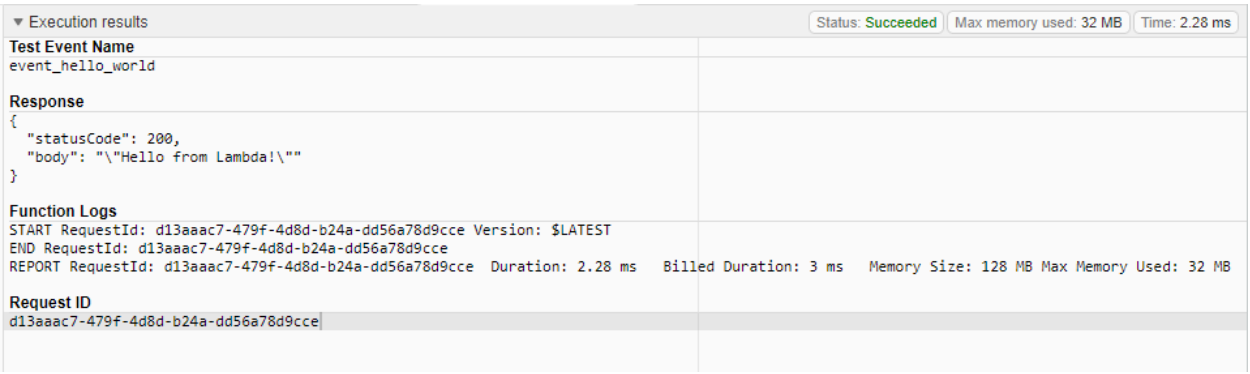
4. Go to the 'Test' tab and click 'Create a new event.' Give the event a name, set 'Event Sharing' to private, and choose the 'hello-world' template.

   We can create a new event to test and validate your Lambda function. By setting Event Sharing to private, we ensure its security, and selecting the "hello-world" template offers a straightforward framework for testing with minimal input complexity.



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





Got above output from Lambda function

6. You can modify your Lambda function code. I've updated it to display the current Date and Time of AWS Server. After making your changes, save them using Control + S and then click on Deploy.

   Function to get current Date and Time of Server:

```
import json
from datetime import datetime

def lambda_handler(event, context):
    # Get the current date and time
    current_time = datetime.utcnow().isoformat() + 'Z'

    return {
        'statusCode': 200,
        'body': json.dumps(f'Current server date and time: {current_time}')
    }
```

   Note: After making change in lambda function, ensure you deploy it before running test

lambda_function × Environment Vari × Execution results ×

```python
import json
from datetime import datetime

def lambda_handler(event, context):
    # Get the current date and time
    current_time = datetime.utcnow().isoformat() + 'Z'

    return {
        'statusCode': 200,
        'body': json.dumps(f'Current server date and time: {current_time}')
    }
```

lambda_function × Environment Var × Execution result ×

Execution results — Status: Succeeded | Max memory used: 32 MB | Time: 2.02 ms

Test Event Name

event_hello_world

Response

```
{
  "statusCode": 200,
  "body": "\"Current server date and time: 2024-10-01T05:42:14.718370Z\""
}
```

Function Logs

```
START RequestId: 1db8fc49-2f86-46bd-82a8-1337ffd8c40e Version: $LATEST
END RequestId: 1db8fc49-2f86-46bd-82a8-1337ffd8c40e
REPORT RequestId: 1db8fc49-2f86-46bd-82a8-1337ffd8c40e  Duration: 2.02 ms   Billed Duration: 3 ms   Memory Size: 128 MB Max Memory Used: 32 MB
```

Request ID

1db8fc49-2f86-46bd-82a8-1337ffd8c40e

After changing the lambda function, we are getting Date and Time from the Server.

Conclusion:

In conclusion, I conducted an experiment using the **Hello World** template and achieved the expected results. Initially, I encountered issues with my custom Lambda function, which was designed to retrieve the server's date and time, due to it not being deployed. After deploying the function, it executed correctly and returned the desired output. This experience underscored the critical importance of deploying changes to ensure they are properly reflected and take effect in production.

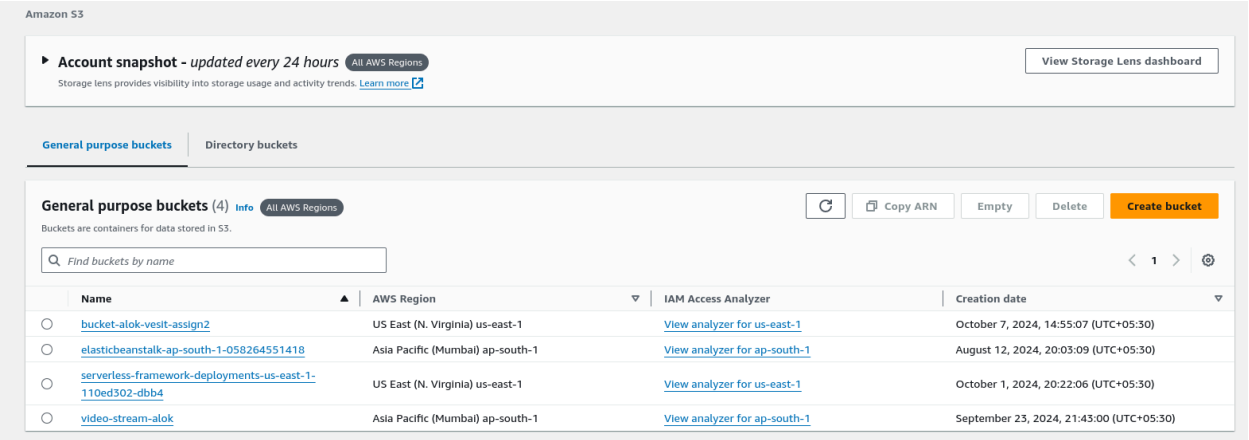
Name: Alok Yadav Div: D15C Roll No.: 59

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

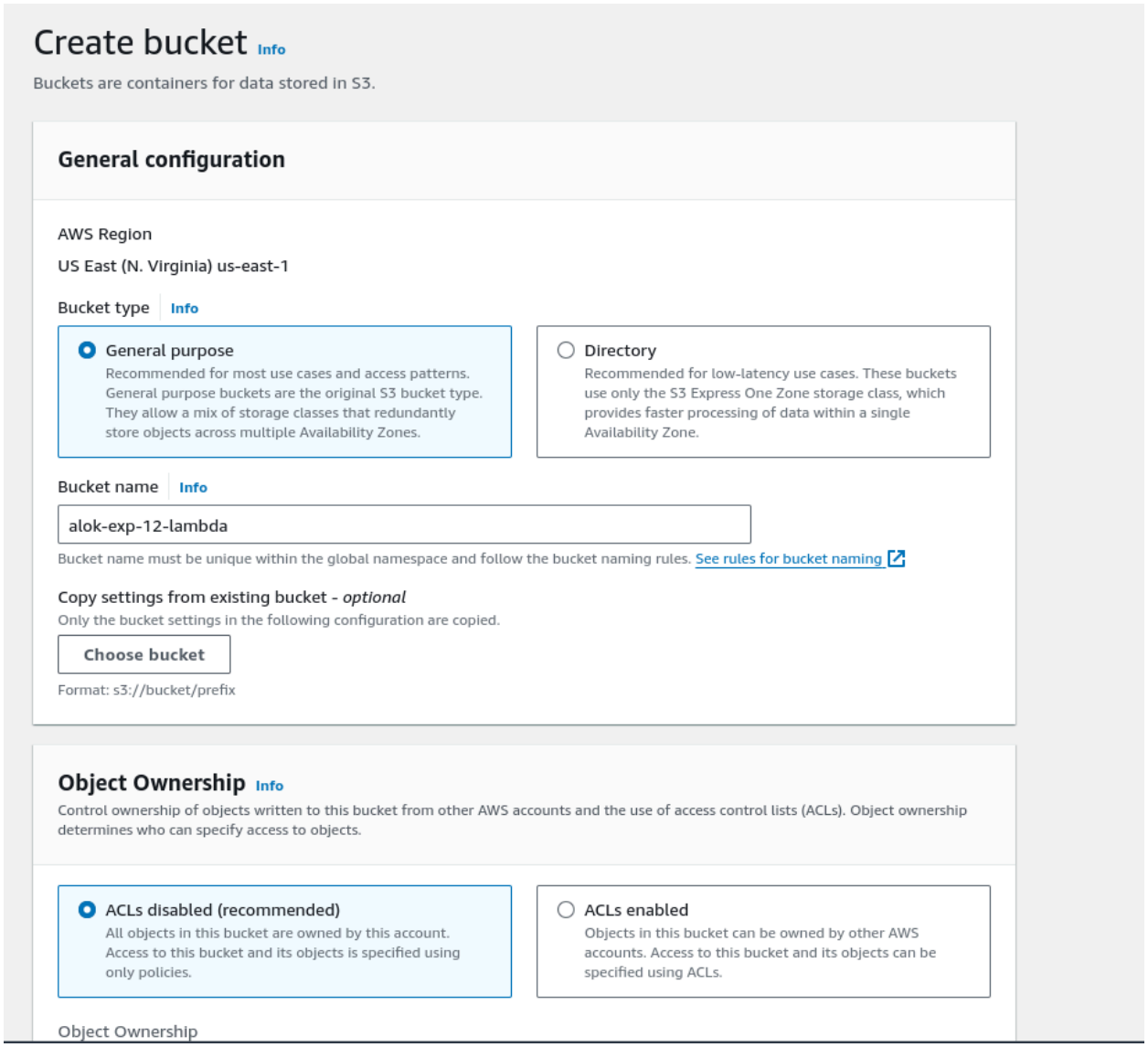
Steps To create the lambda function:

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



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

Name: Alok Yadav Div: D15C Roll No.: 59



Bucket owner enforced

Block Public Access settings for this bucket

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

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

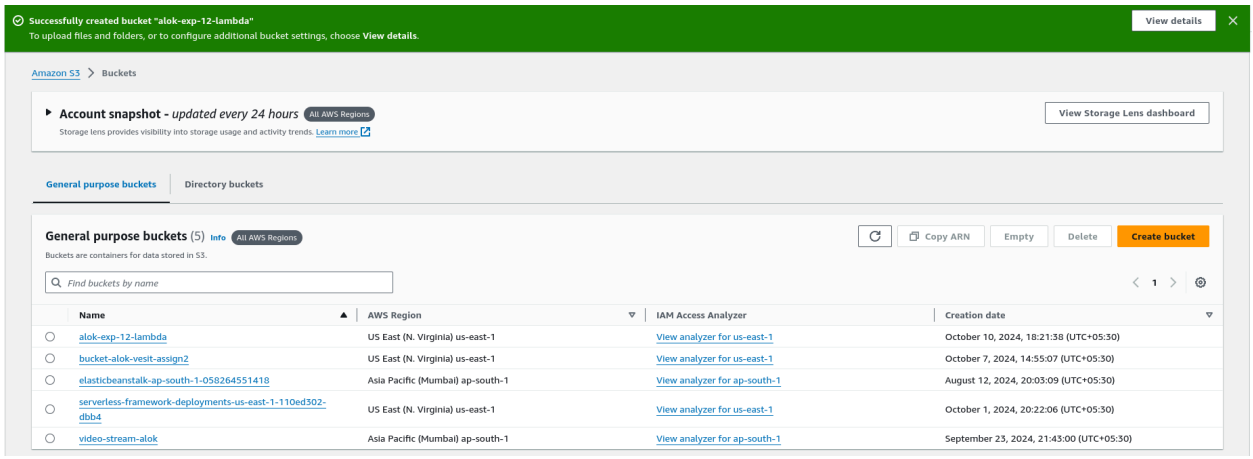
Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. Learn more

Bucket Versioning

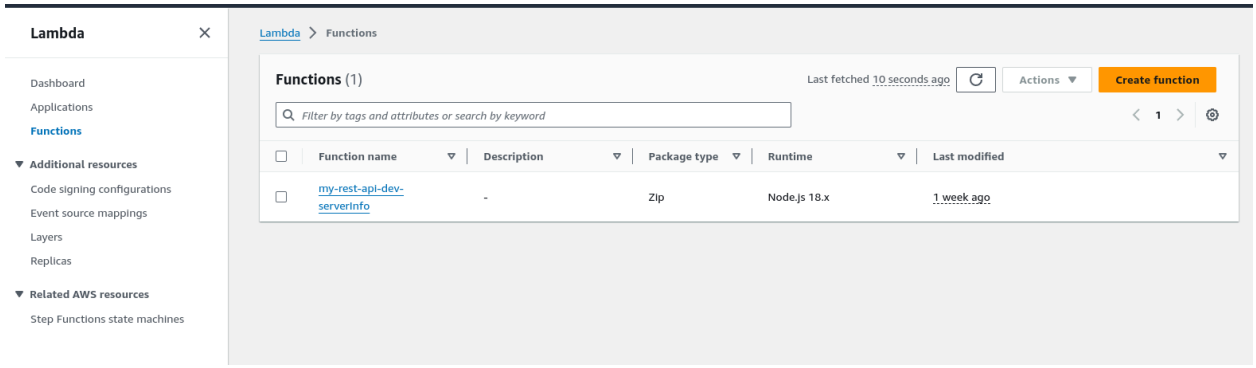
- (•) Disable
- ( ) Enable

Tags - *optional* (0)



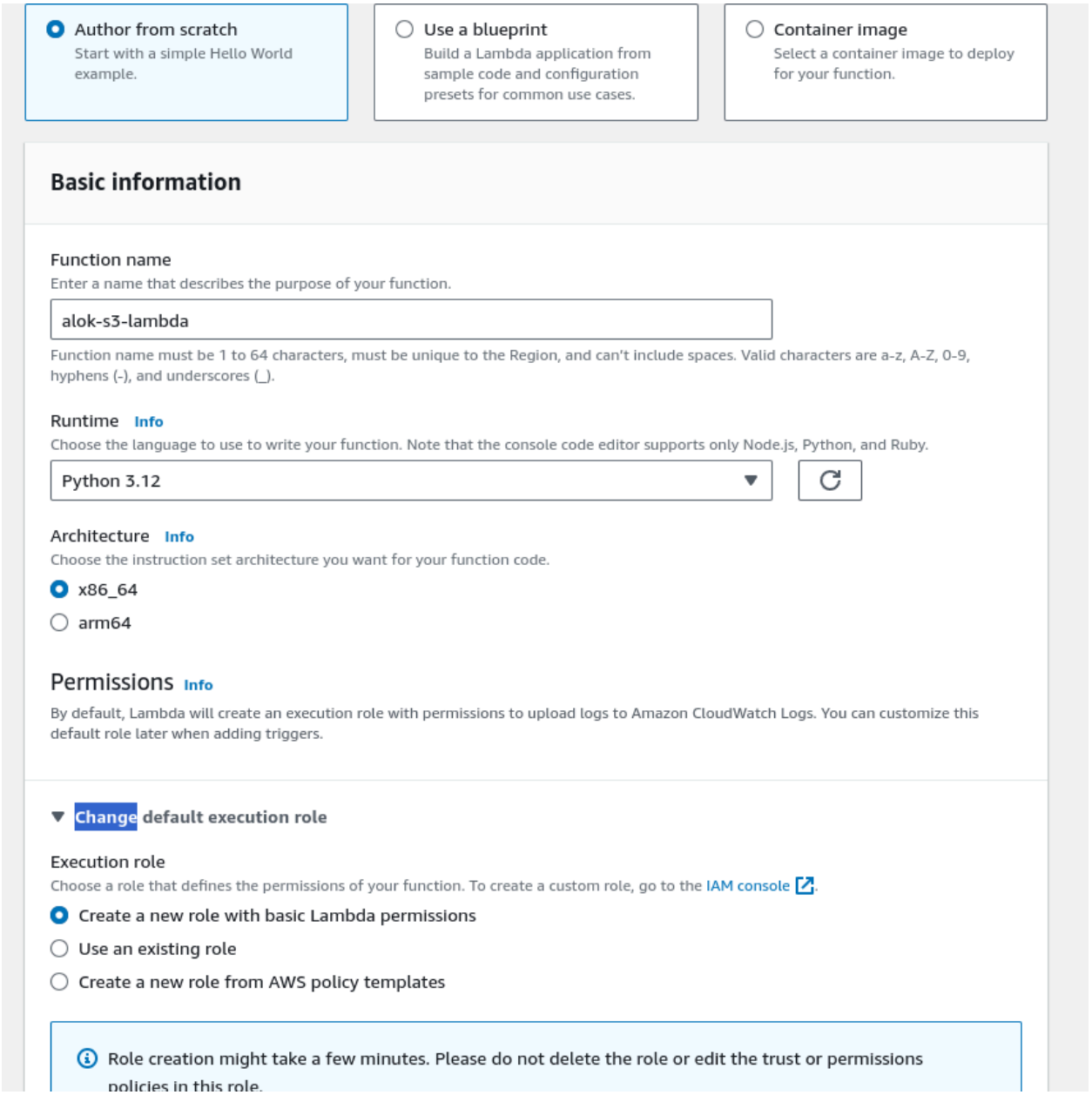
Bucket created successfully

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

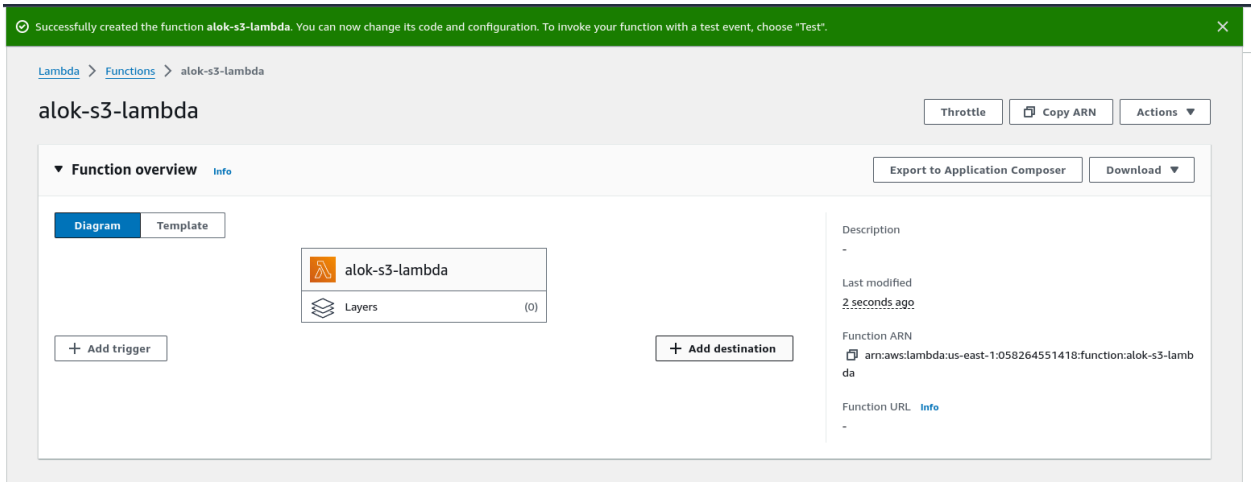


4. Now Give a name to your Lambda function, Select the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby. So will select Python 3.12 Architecture as x86, and Exceution role to Create a new role with basic Lambda permissions.

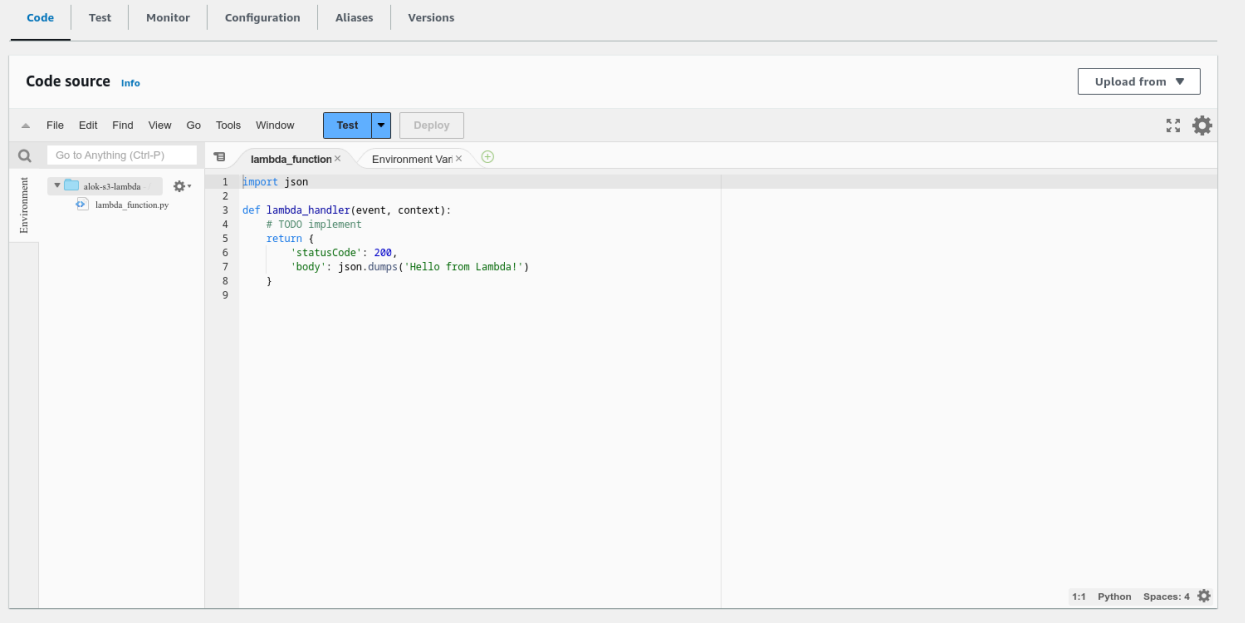
Name: Alok Yadav Div: D15C Roll No.: 59



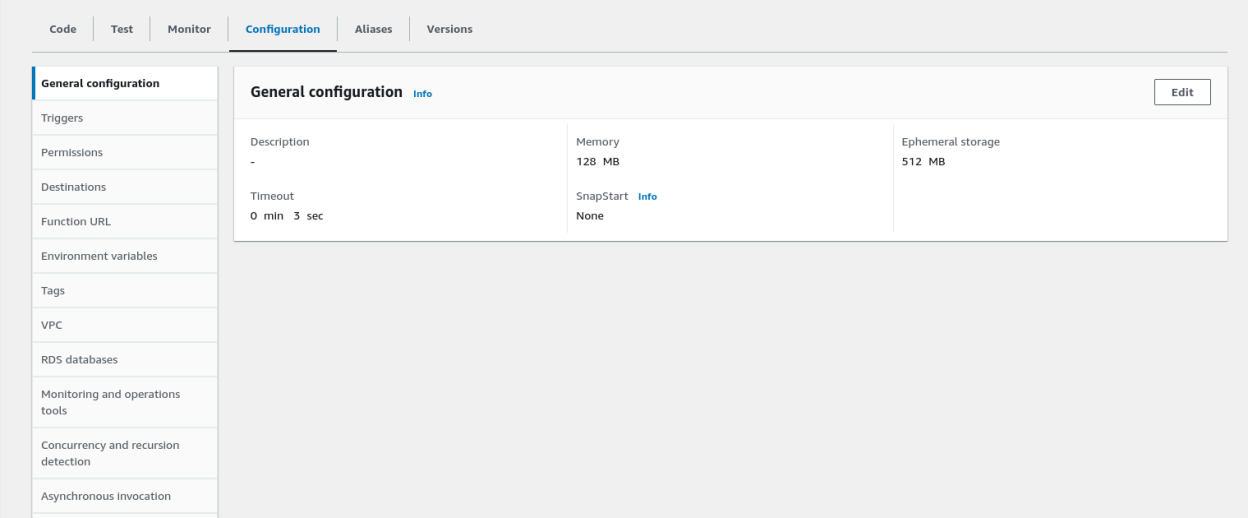
Name: Alok Yadav Div: D15C Roll No.: 59



Lambda function created successfully

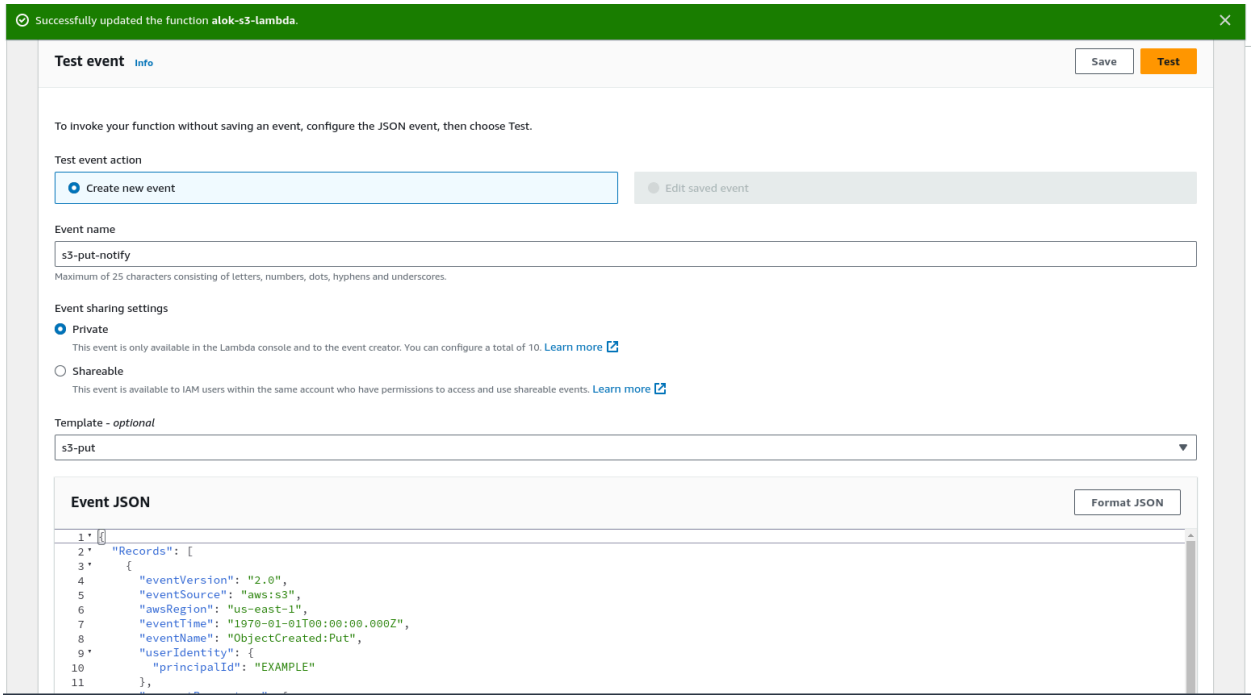


So to See or Edit the basic settings go to configuration then click on edit general setting.



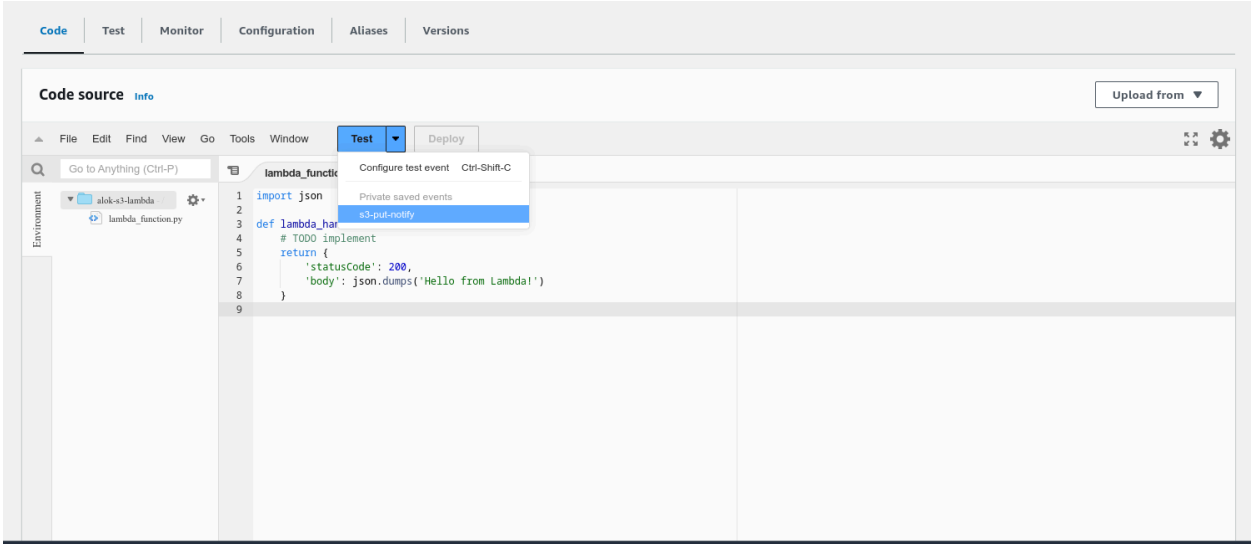
Here, you can enter a description and change Memory and Timeout. I've changed the
Timeout period to 1 sec since that is sufficient for now.

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.

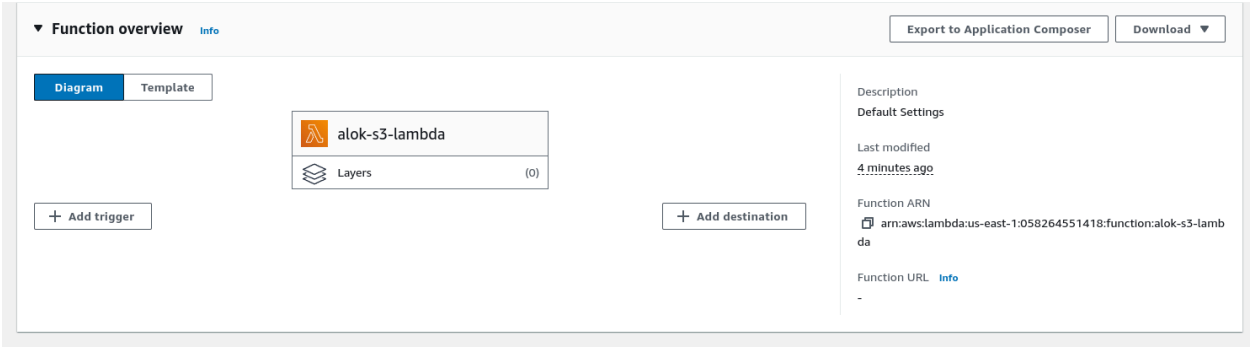


6. Now In Code section select the created event from the dropdown .

Name: Alok Yadav Div: D15C Roll No.: 59

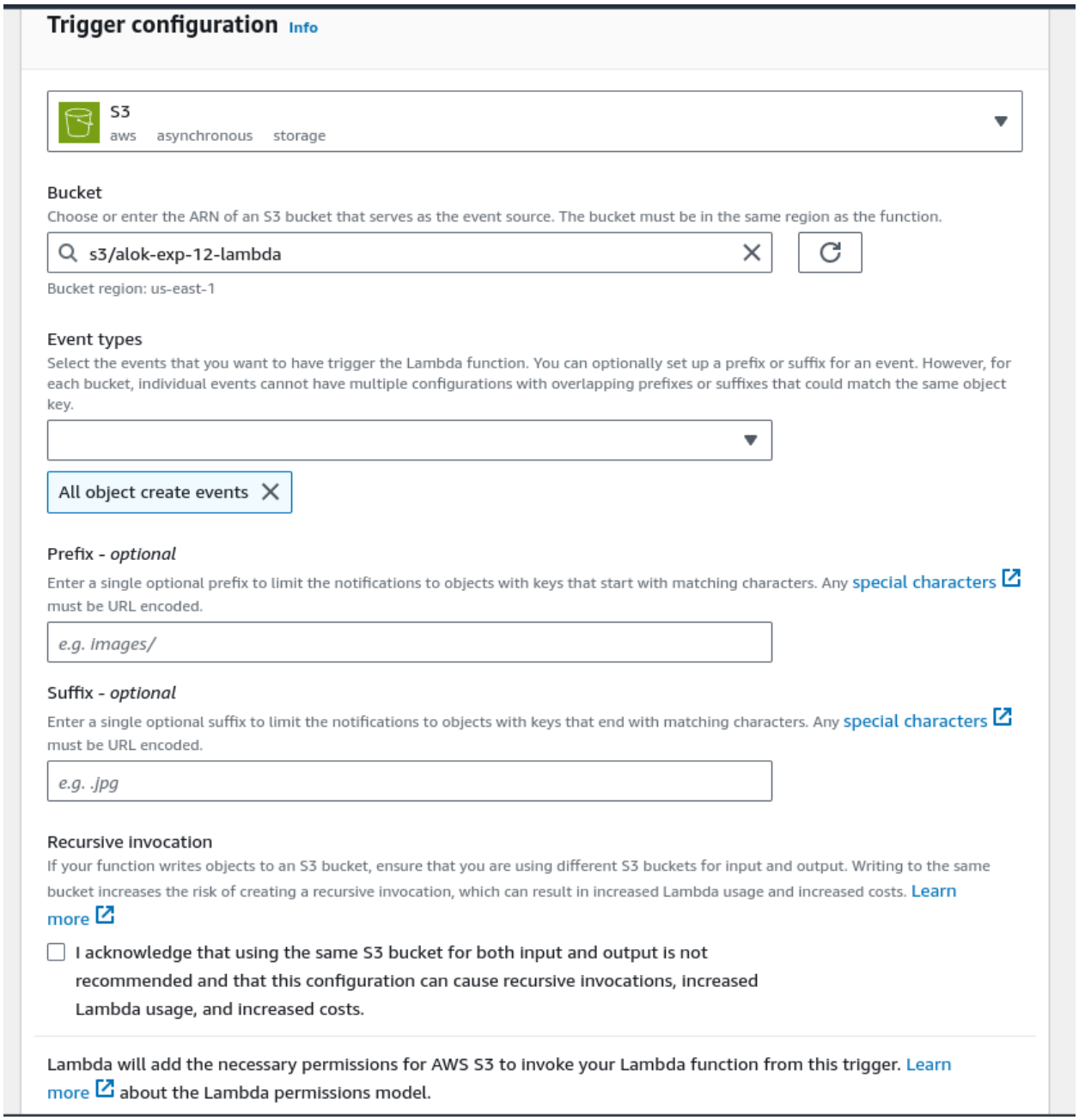


7. Now In the Lambda function click on add tigger.

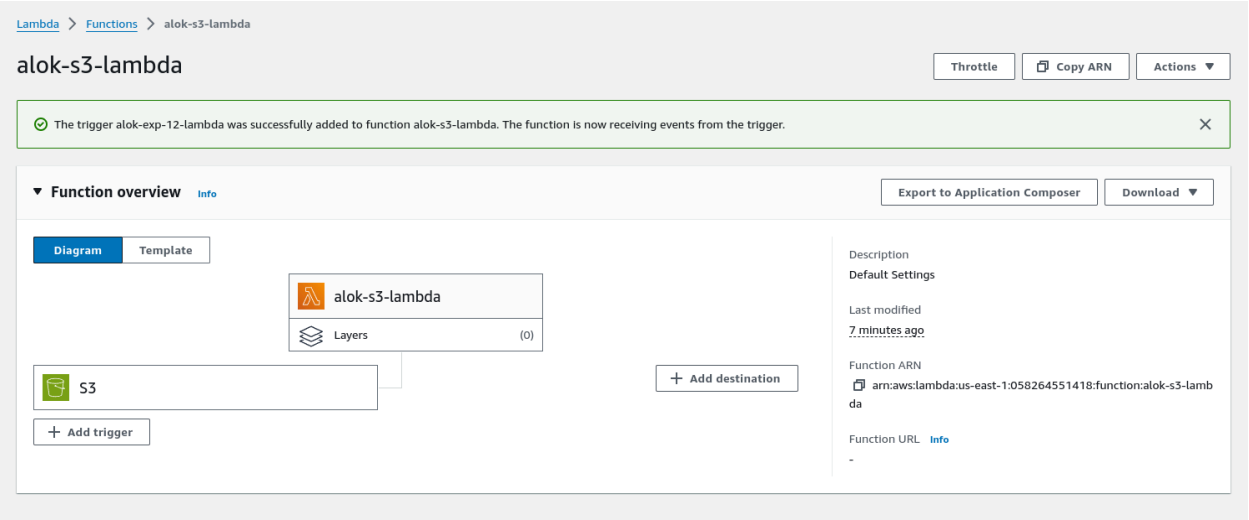


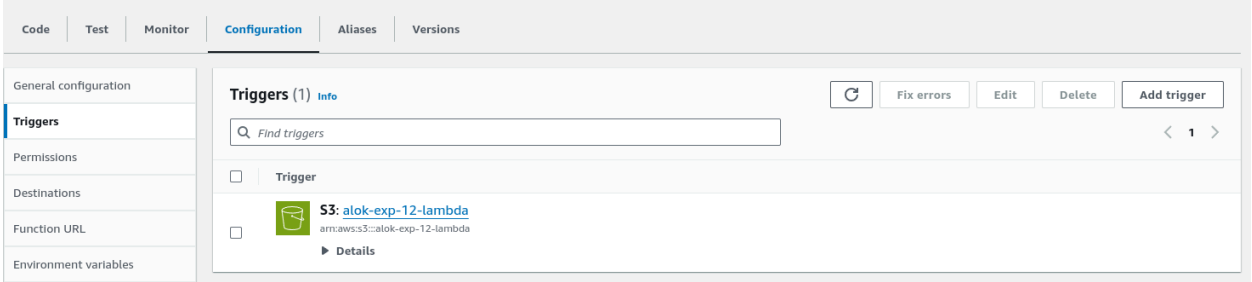
Now select the source as S3 then select the bucket name from the dropdown, keep other things to default and also you can add prefix to image.

Name: Alok Yadav Div: D15C Roll No.: 59



Name: Alok Yadav Div: D15C Roll No.: 59

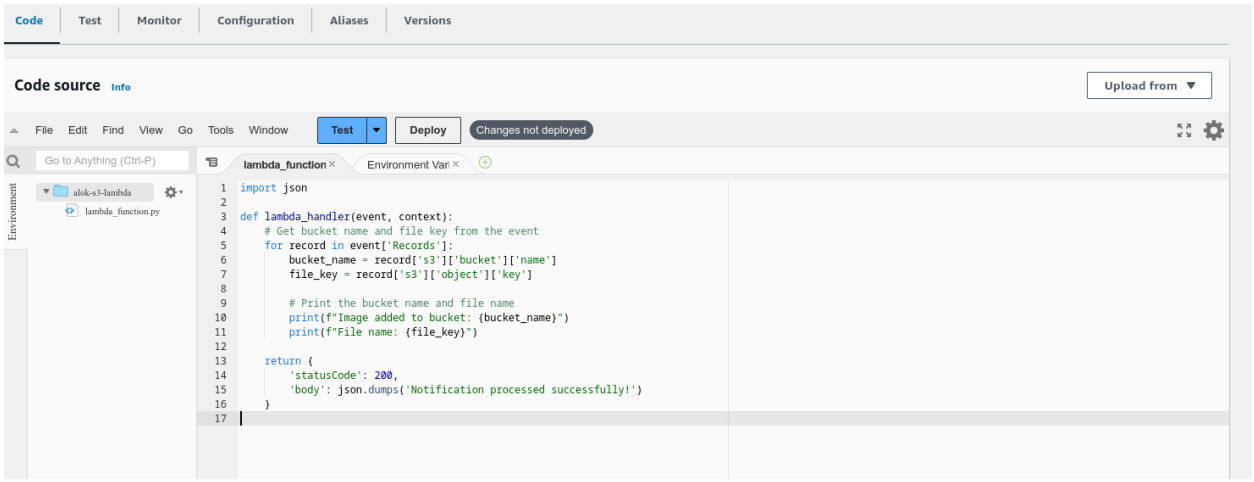




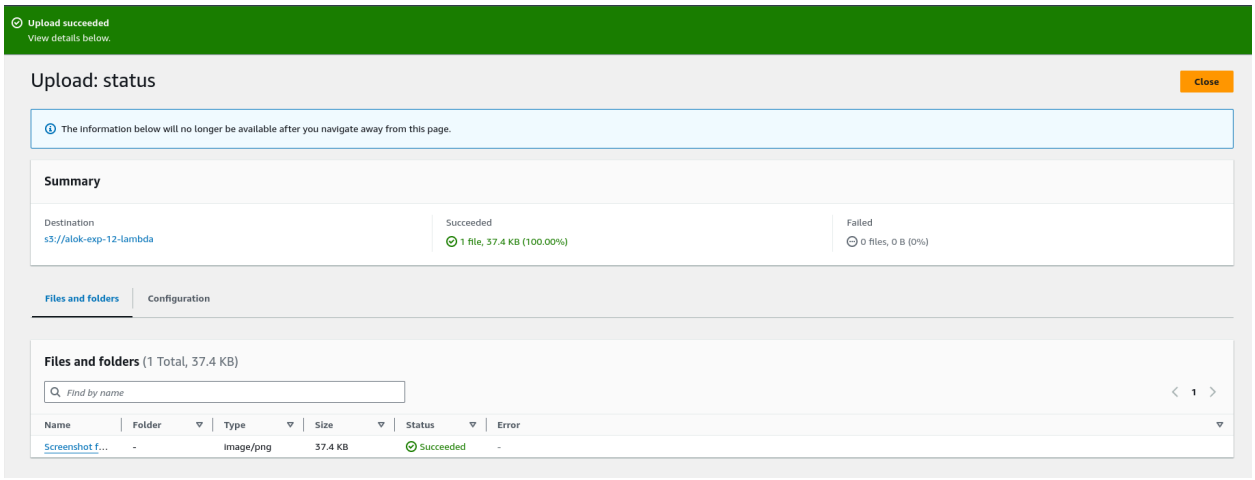
8. Now Write code that logs a message like “Image added to bucket” when triggered. Save the file and click on deploy.

```
import json

def lambda_handler(event, context):
    # Get bucket name and file key from the event
    for record in event['Records']:
        bucket_name = record['s3']['bucket']['name']
        file_key = record['s3']['object']['key']

        # Print the bucket name and file name
        print(f"Image added to bucket: {bucket_name}")
        print(f"File name: {file_key}")

    return {
        'statusCode': 200,
        'body': json.dumps('Notification processed successfully!')
    }
```

Name: Alok Yadav Div: D15C Roll No.: 59

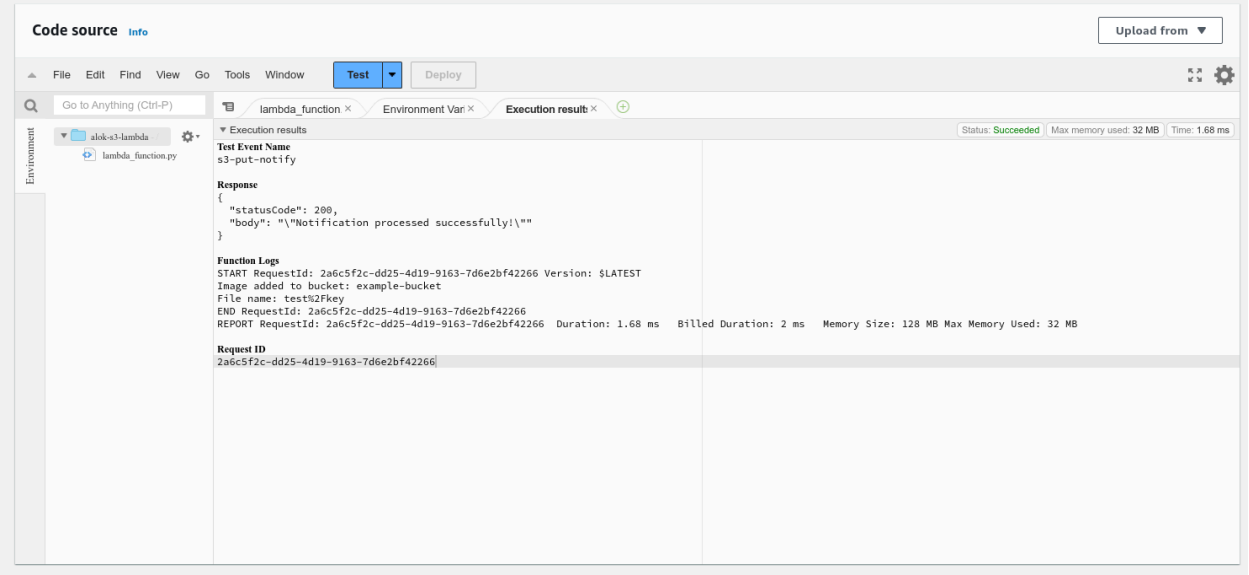


9. Now upload any image to the bucket.

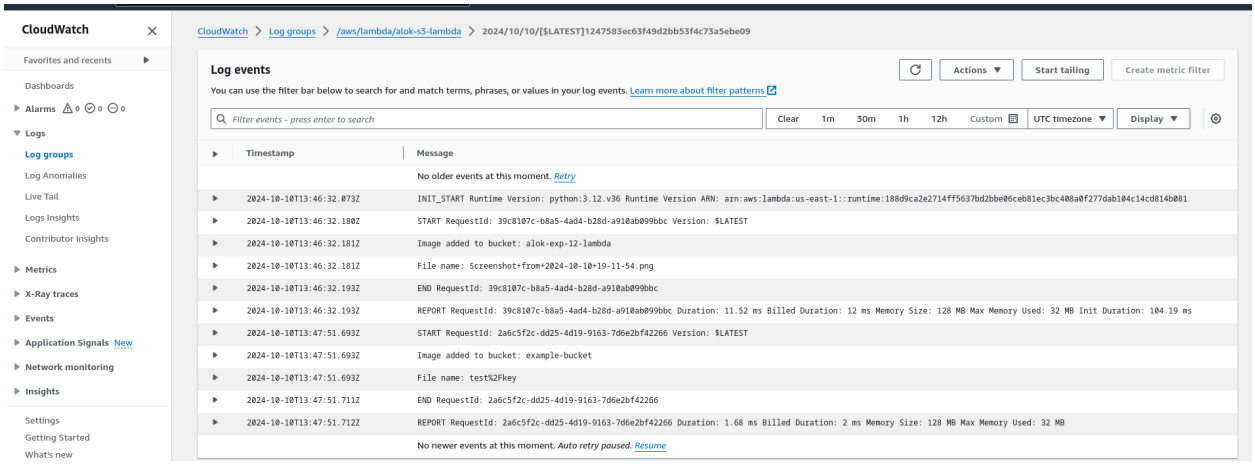


10. Now to click on test in lambda to check whether it is giving log when image is added to S3.

Name: Alok Yadav Div: D15C Roll No.: 59



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



As we can see our lambda function activity is recorded by Cloudwatch

Conclusion:

In this experiment, we successfully implemented an AWS Lambda function that logs a message when an image is uploaded to an S3 bucket. A key aspect was selecting the **S3 Object Created (Put)** event template, as failing to do so would result in errors due to an incompatible event structure. The Lambda function was successfully triggered by S3 object uploads, demonstrating the functionality and efficiency of AWS Lambda's event-driven architecture. This experiment highlighted Lambda's ability to seamlessly respond to S3 events and emphasized

the importance of correctly configuring event triggers to avoid common issues with event data.