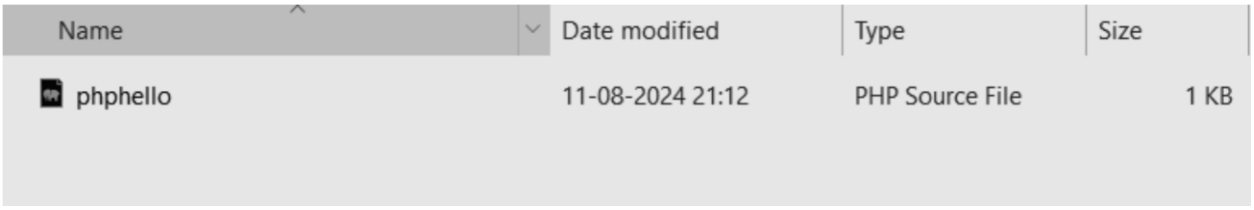
Name:Sahil Motiramani Roll no 35 Div D15C

Exp No.1A

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

Hosting Using Xampp

1.Create a File in your local directory.

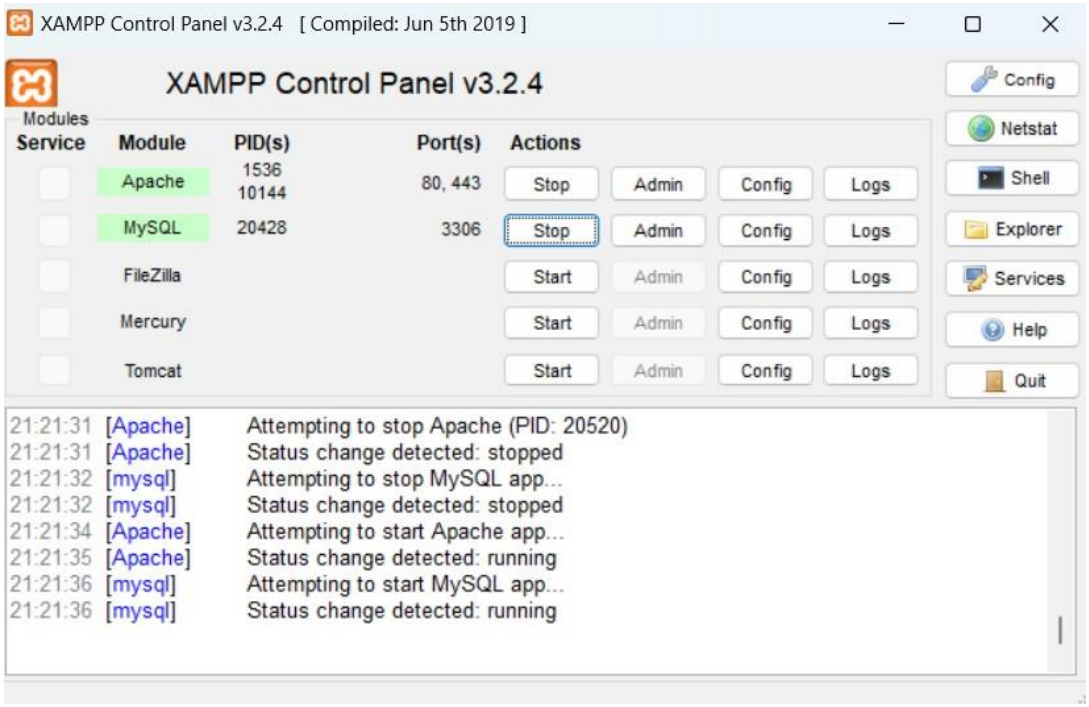


2 .Write the PHP in the File.

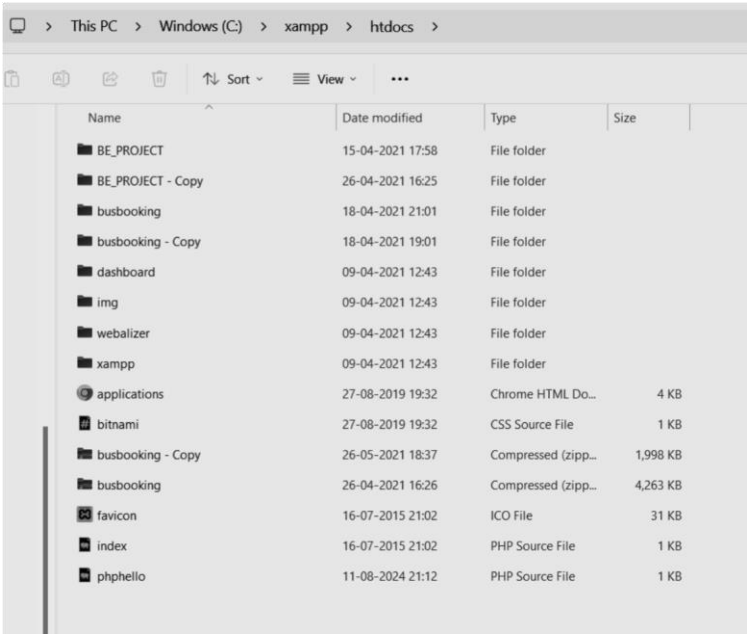
```
<!DOCTYPE html>
<html>
<body>

<h1>My first PHP page</h1>

<?php
echo "Hello World!";
?>

</body>
</html>
```

3. Turn on Xampp Control panel.
And Start the Apache Module and MySQL Module.

Name:Sahil Motiramani Roll no 35 Div D15C

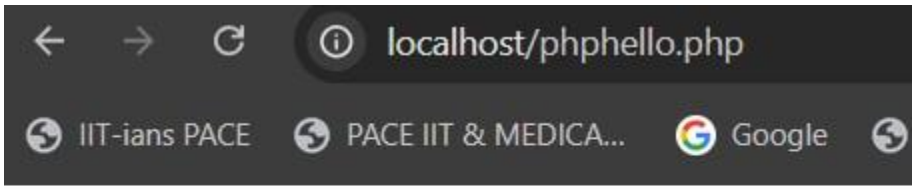


4. Paste the File Which you have made in This PC → Windows(c) →Xammp →htdocs folder



5.Type localhost.<filename>.php in the browser
Here the File name is “phphello”.

Name:Sahil Motiramani Roll no 35 Div D15C



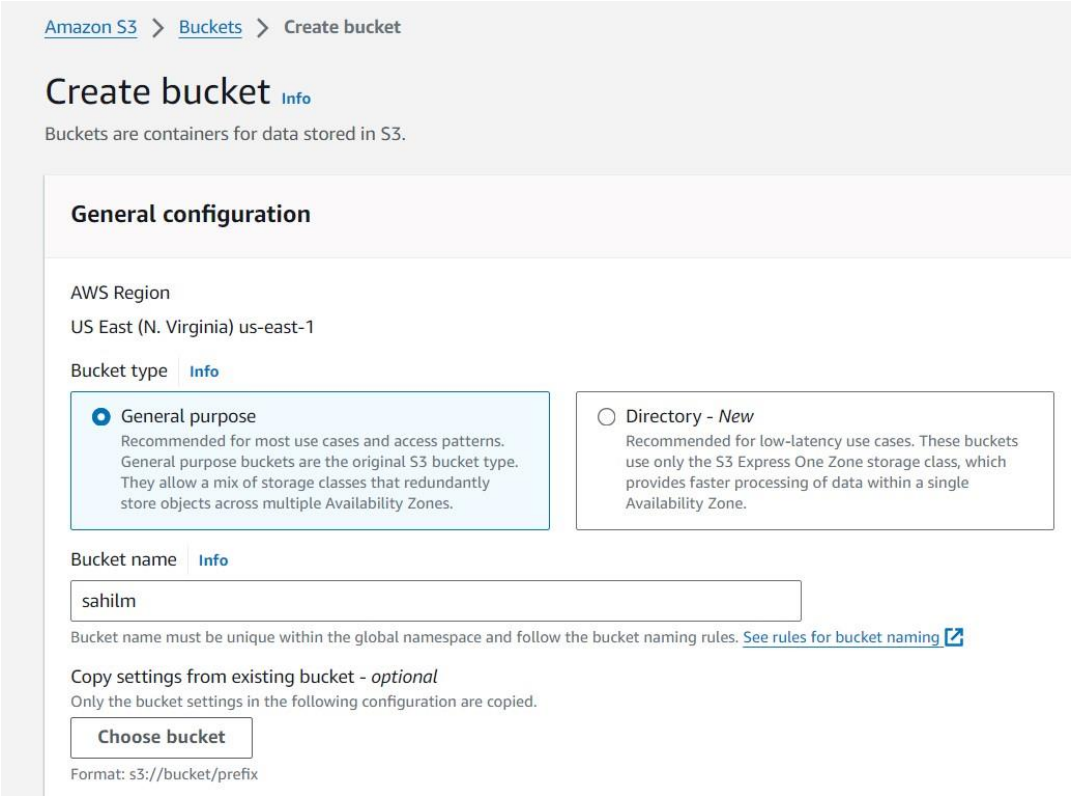
My first PHP page

Hello Sahil

Name:Sahil Motiramani Roll no 35 Div D15C

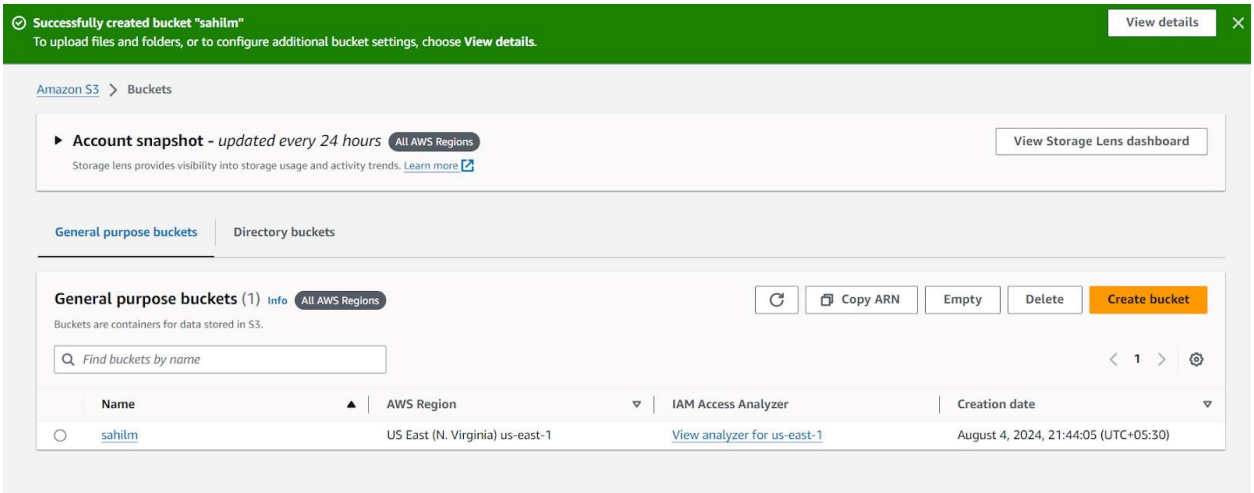
Exp 1A Part 2 :Static Hosting Using AWS

1.Start the AWS lab and Navigate to the services and choose S3 bucket and Click on create Bucket.
Name the bucket according to you.

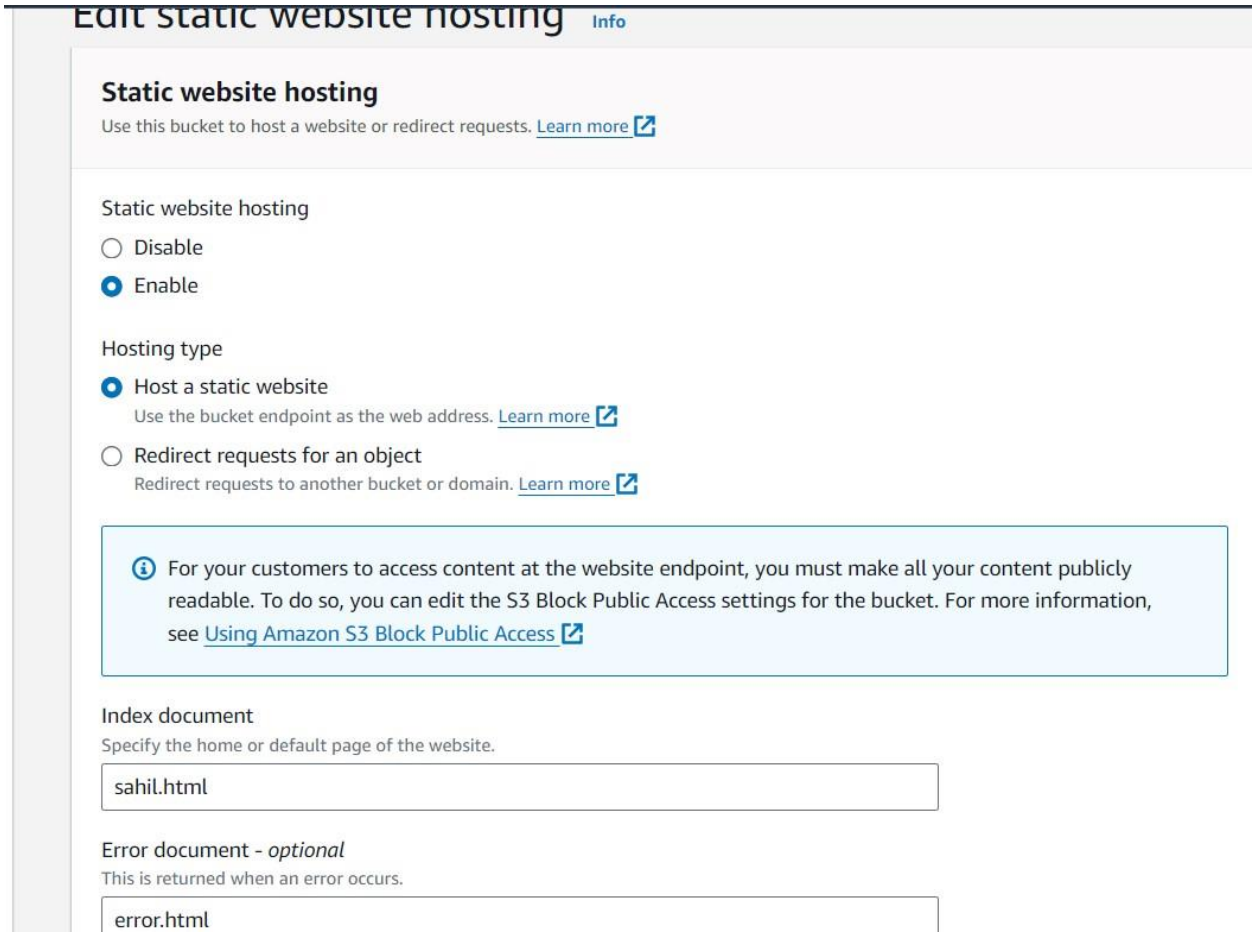


2. Bucket will be created.

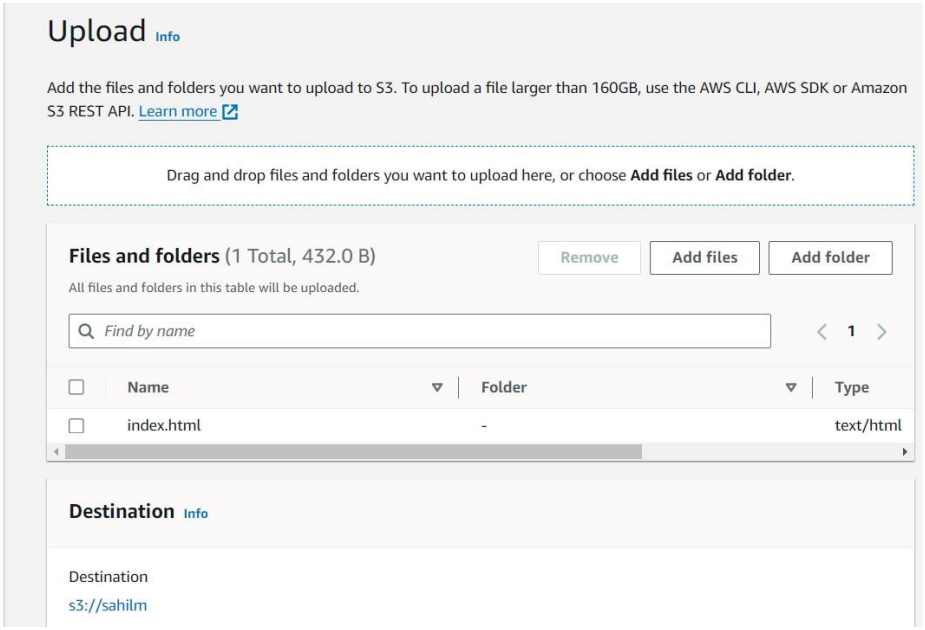
3. Configure the Edit Static website hosting page.



Name:Sahil Motiramani Roll no 35 Div D15C

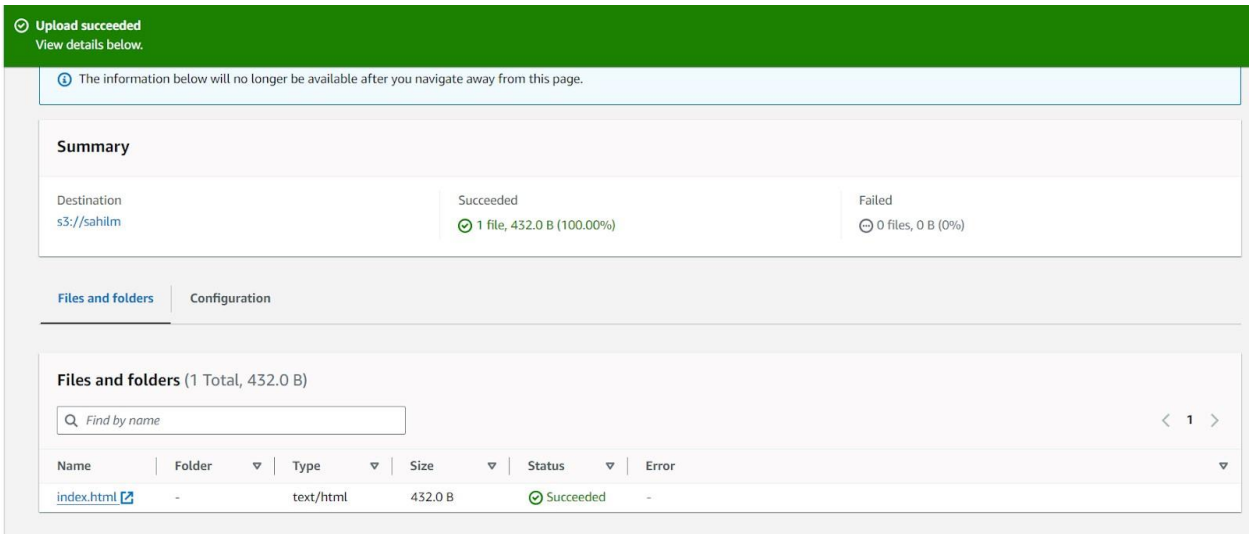


4. Upload the File Which want to Host.

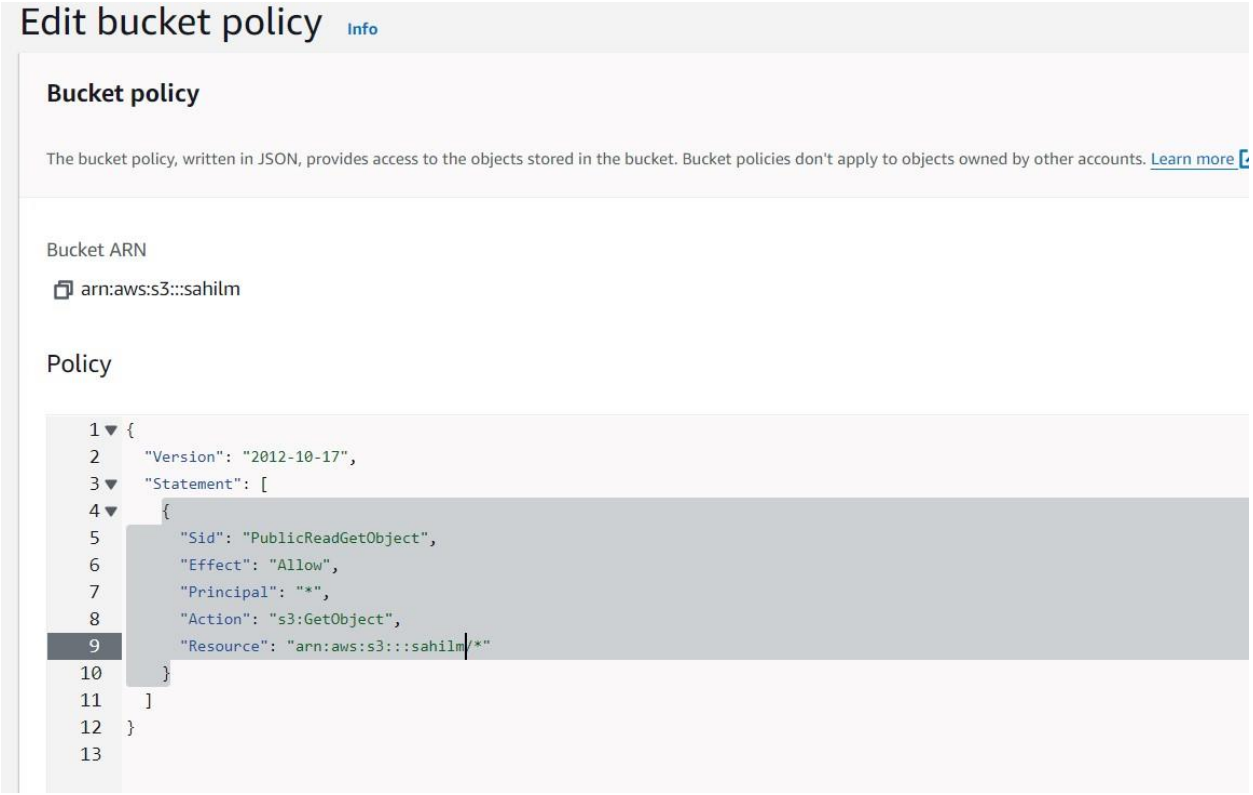


Name:Sahil Motiramani Roll no 35 Div D15C

5.The file Will be uploaded.

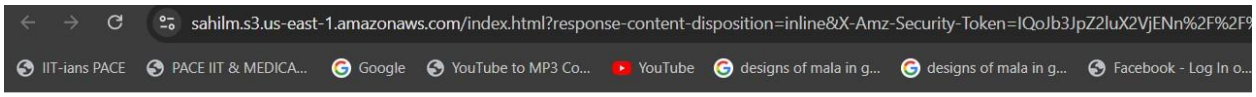


6. Edit the Bucket Policies and use this code replace the sahilm with the bucket name Here my bucket name is sahilm.



Name:Sahil Motiramani Roll no 35 Div D15C

7. After editing Policy the link will be generated of static hosting.

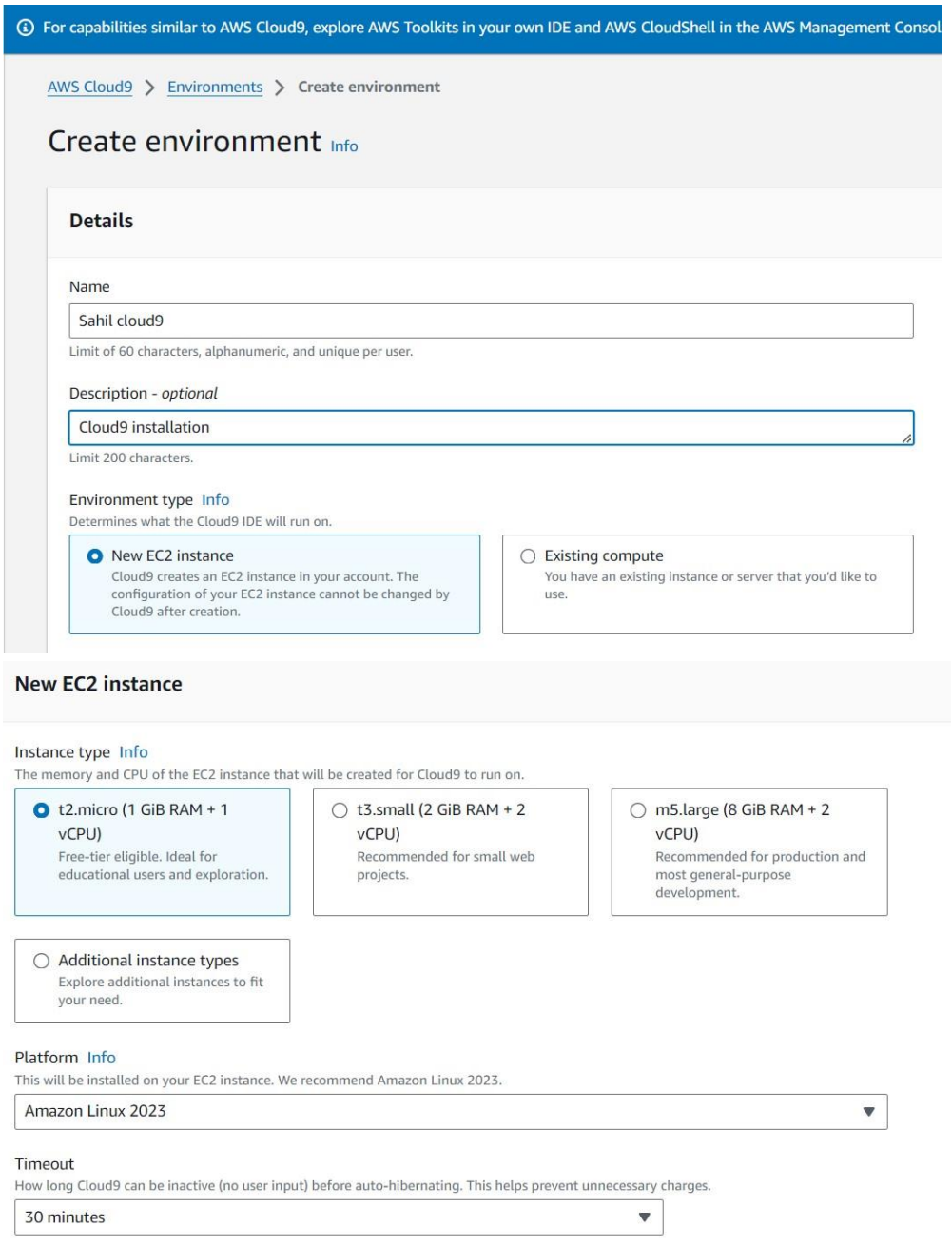


My Portfolio

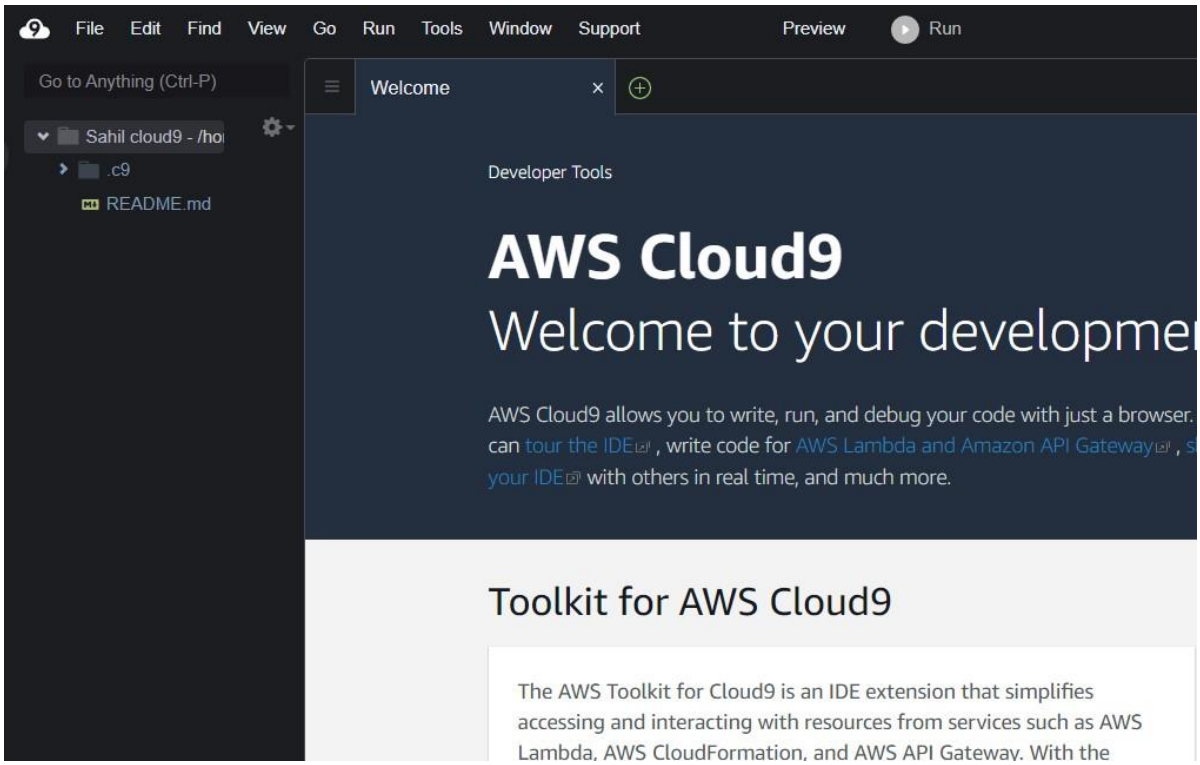
Name:Sahil Motiramani

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

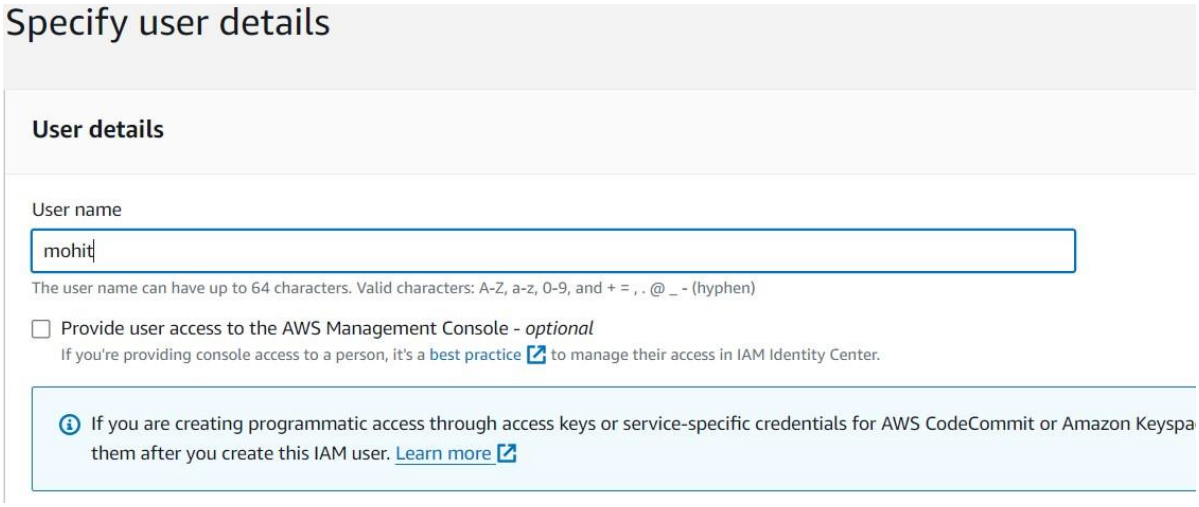
1. Open your AWS account and search for Cloud9 service inside Developer tools. Create a new Cloud9 environment by filling in the required details. Make sure you use an EC2 instance to create your environment.

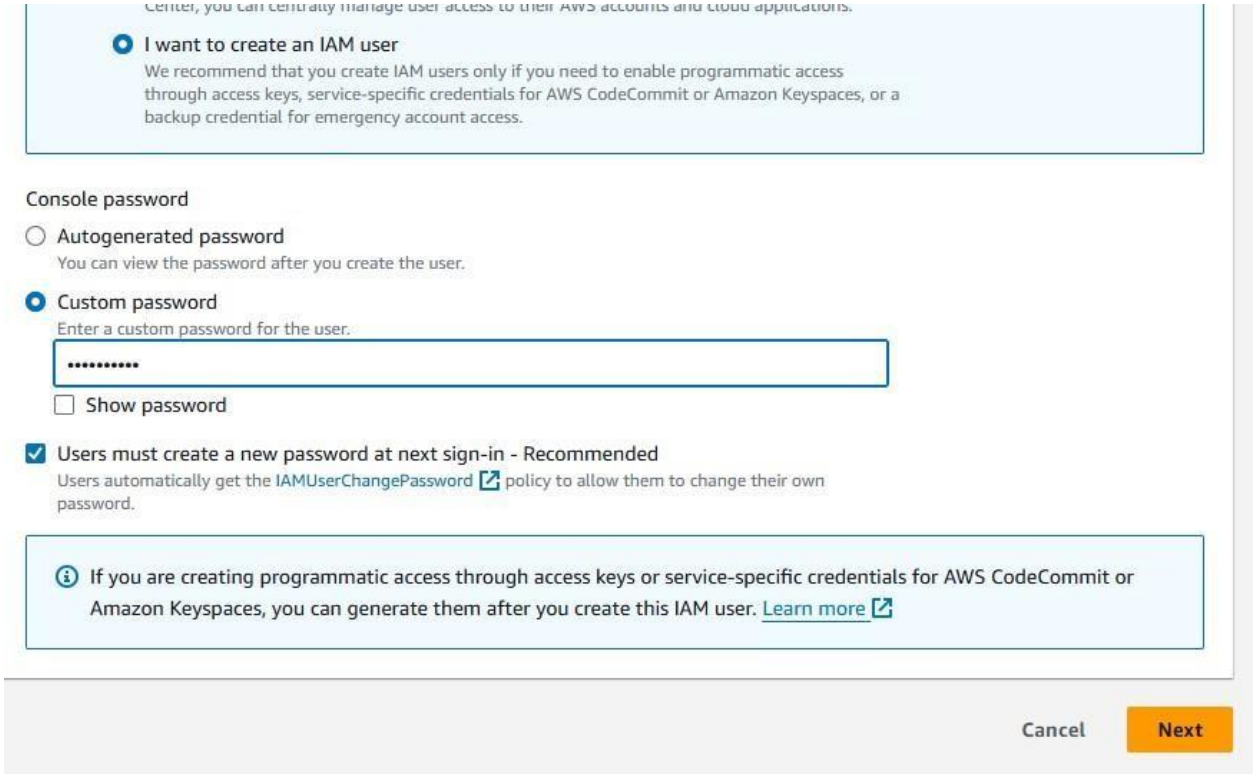


2. We have successfully setup and launched our Cloud9 environment. Over here, we can build and develop programs as per our desire. We are also allowed to collaborate with multiple other users and access shared resources.

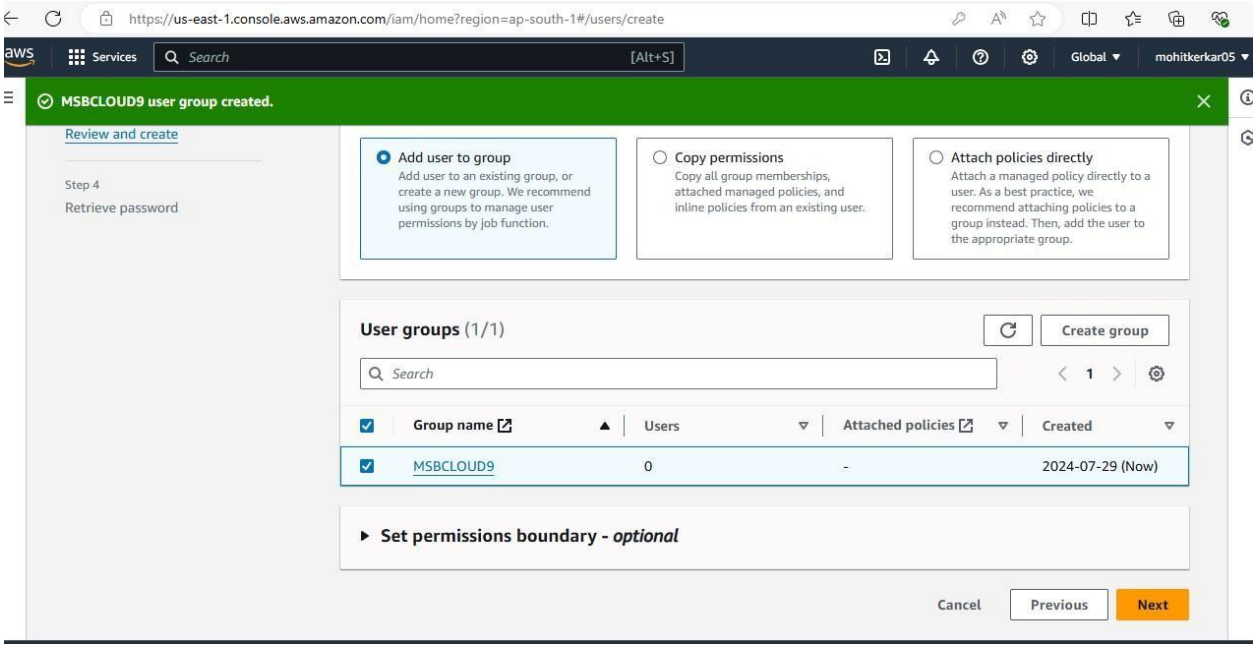


3. Moving on, we are supposed to create a new user. Give a suitable name to the user and decide the password for the same.

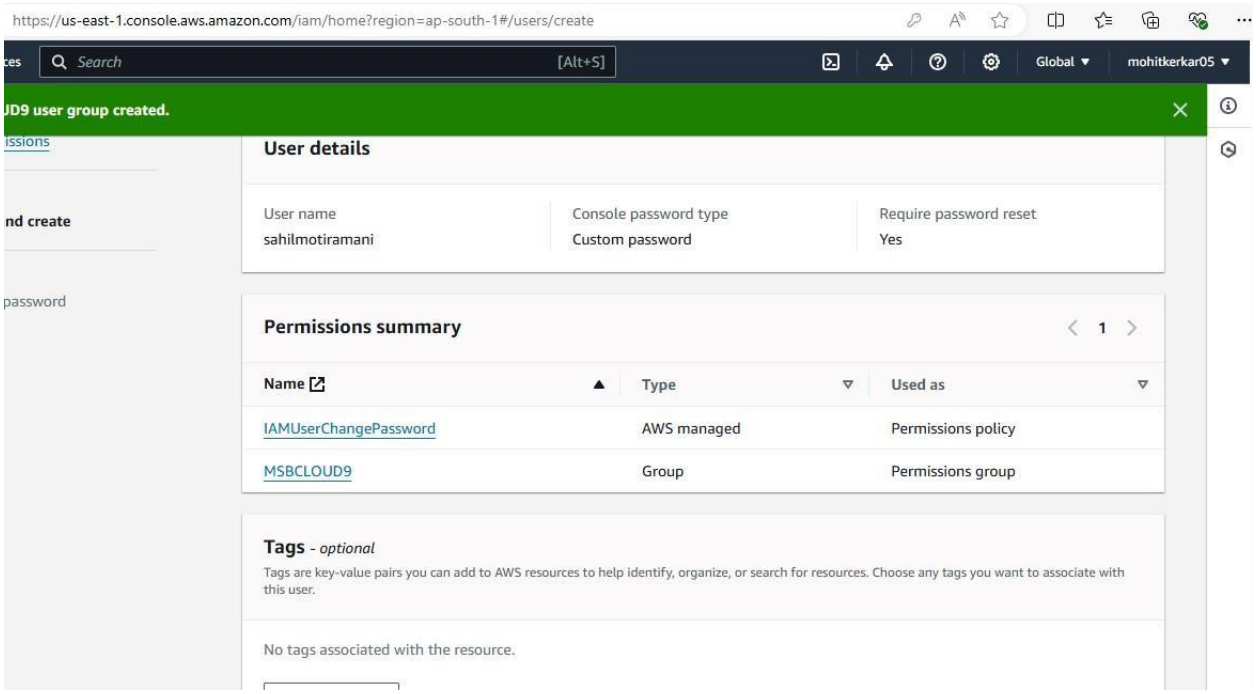




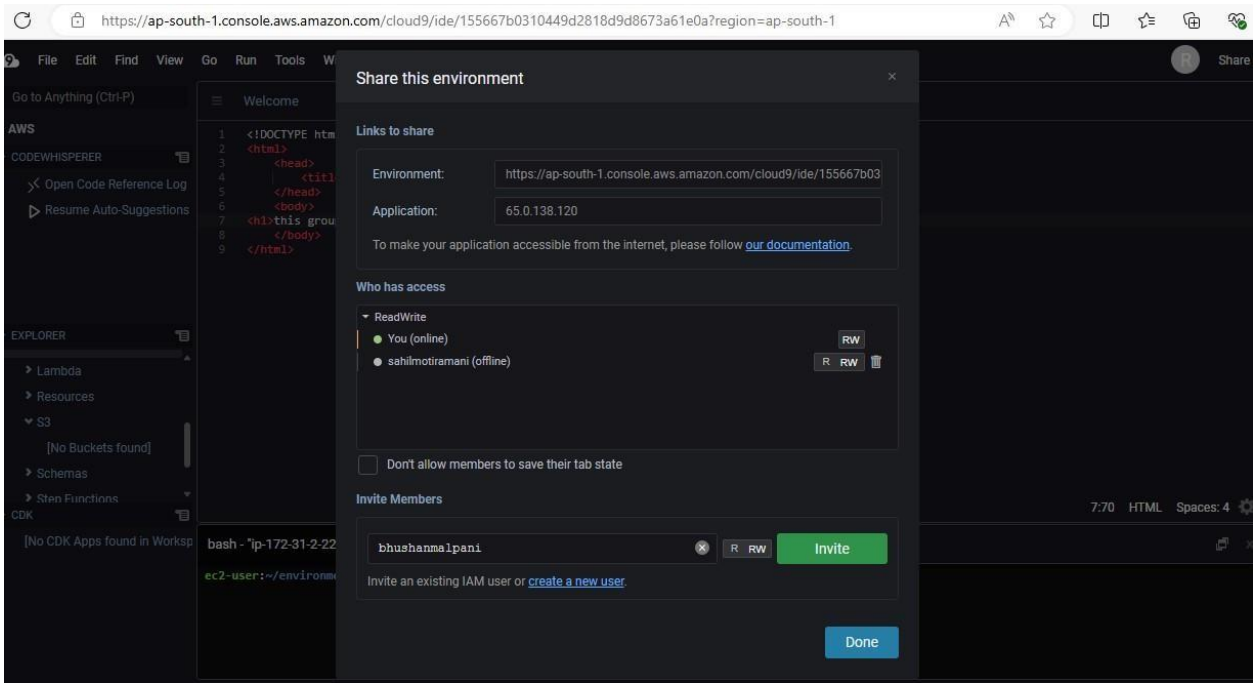
4. Similarly, create a new group and provide a suitable name for the same. Include the IAM users in this group together for our convenience i.e to provide similar kinds of permissions to the entire group rather than an individual user.



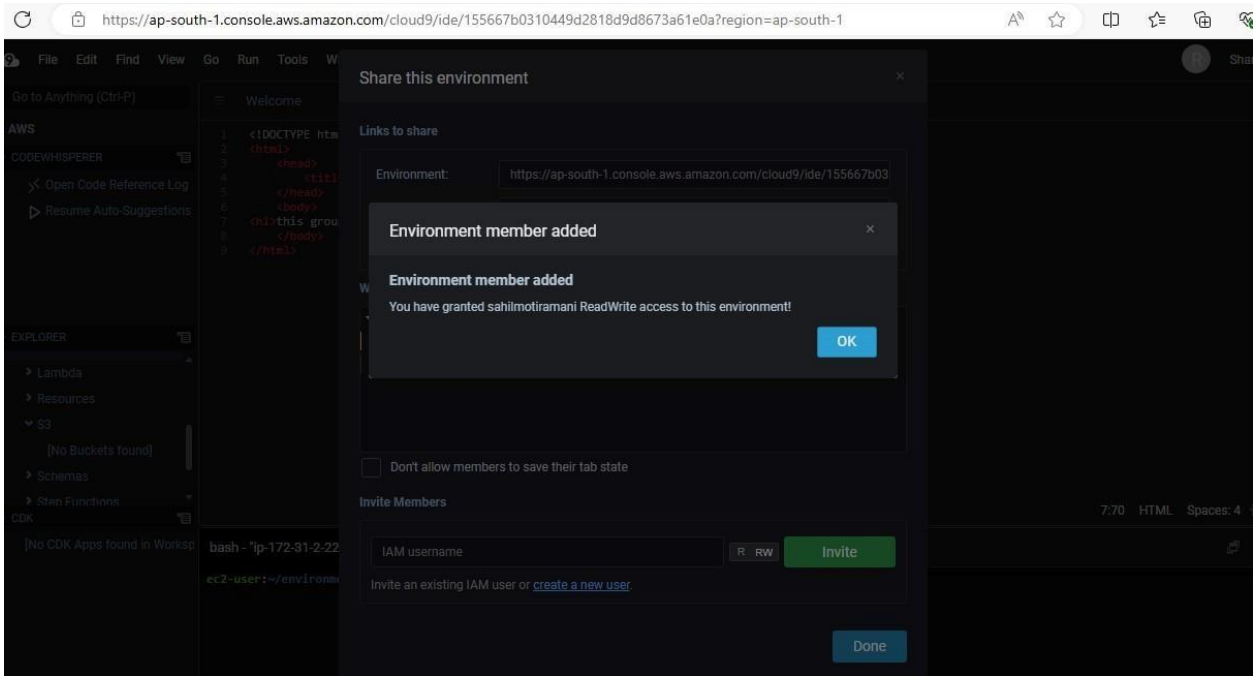
5. The user has successfully been created i.e There is a custom made username and a password for the IAM user.



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



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

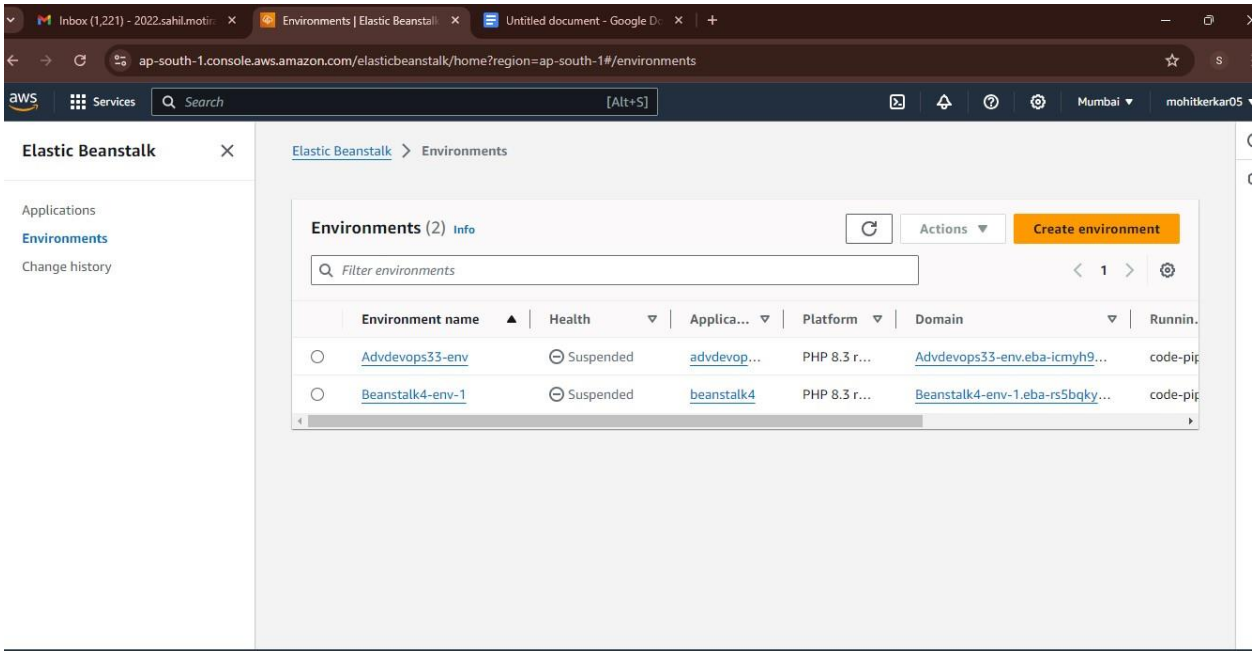


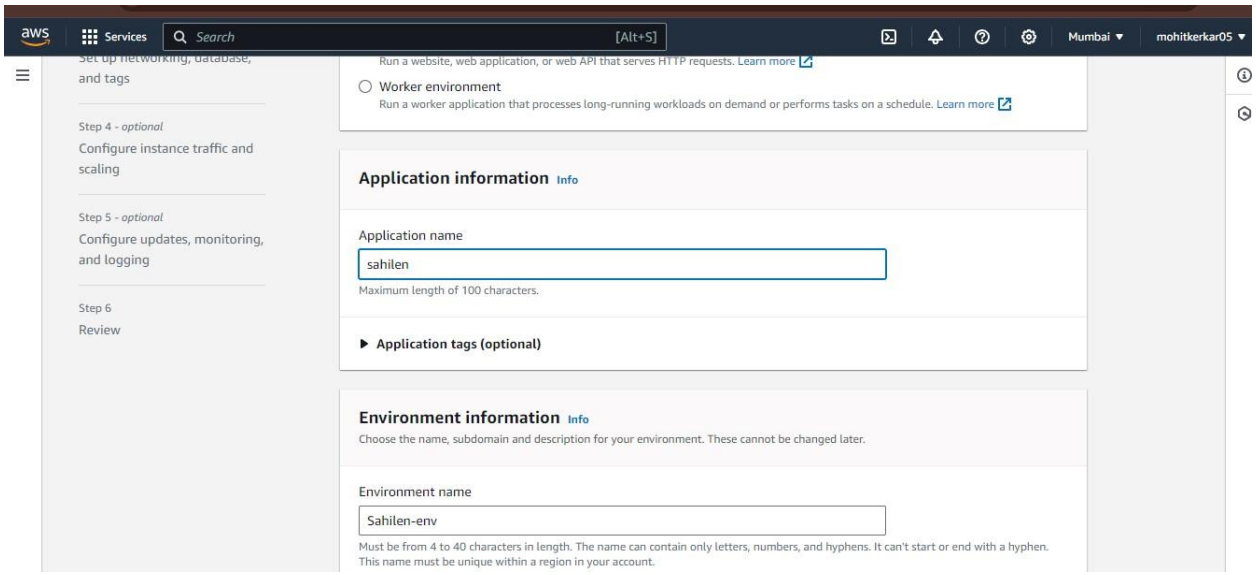
Name:Sahil Motiramani Roll No:35 Div:D15C

Exp:2

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

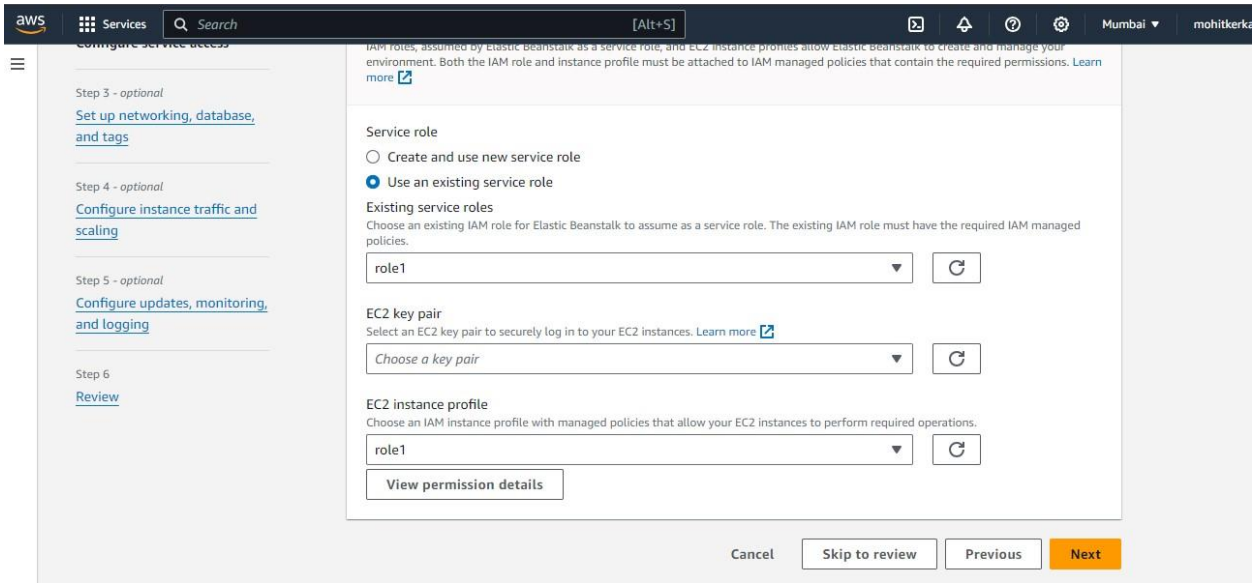
1.Login to your AWS account and got to services , search for Elastic Beanstalk in search box open up Elastic Beanstalk and Create Your environment.





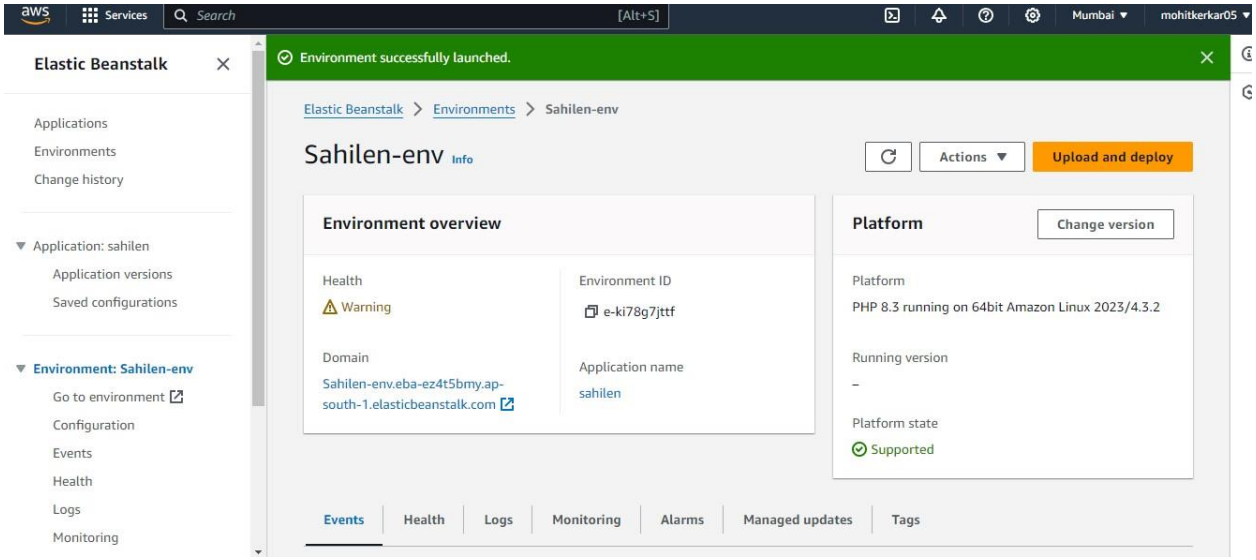
Name:Sahil Motiramani Roll No:35 Div:D15C

2. Give Name to tour application.
3.Configure the Service roles.



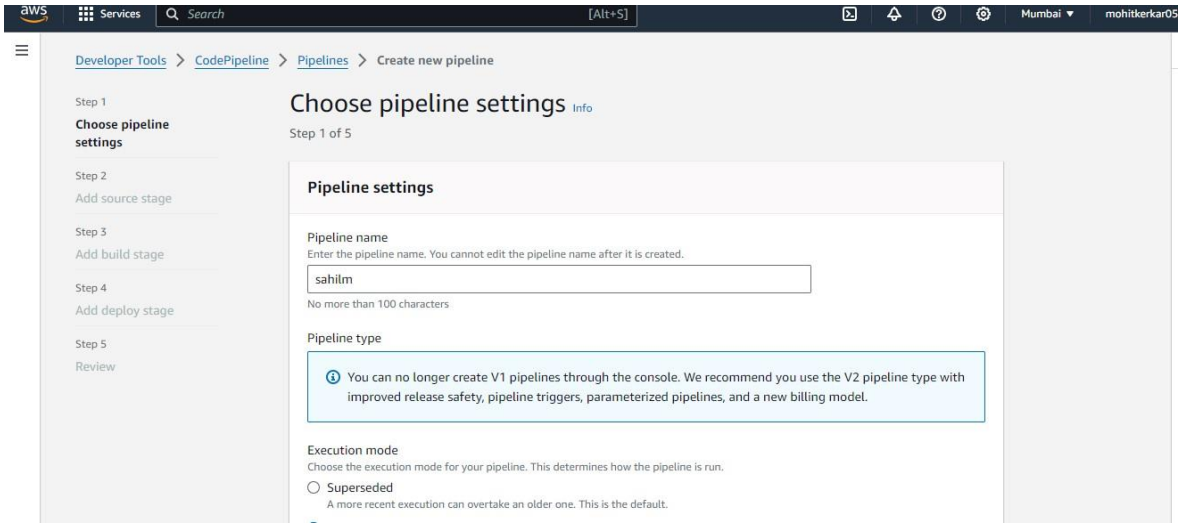
Skip the Further steps and directly go to the review part.

4.Click on Create environment,Then your environment will be created.

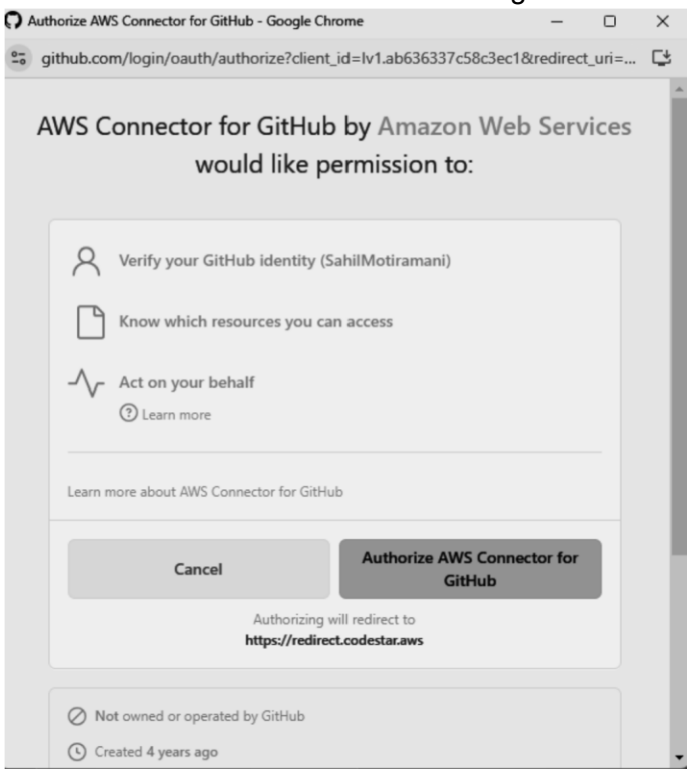


Step 2: Create an new Pipline.

1. Now, Navigate to the services and Search for Code Pipeline → Piplines → Create bew pipeline.

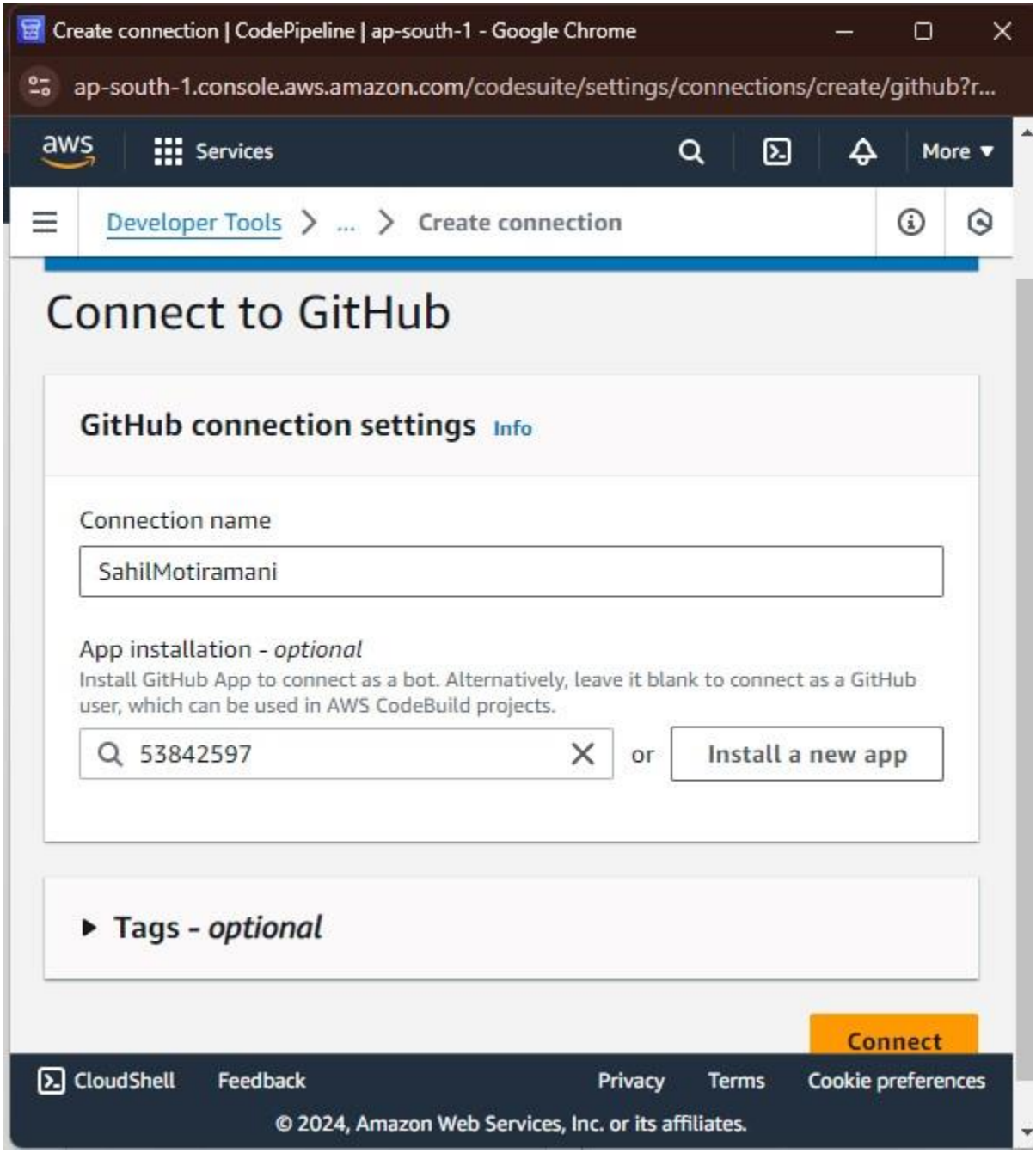


2. Connect AWS with Git Hub and give the access.



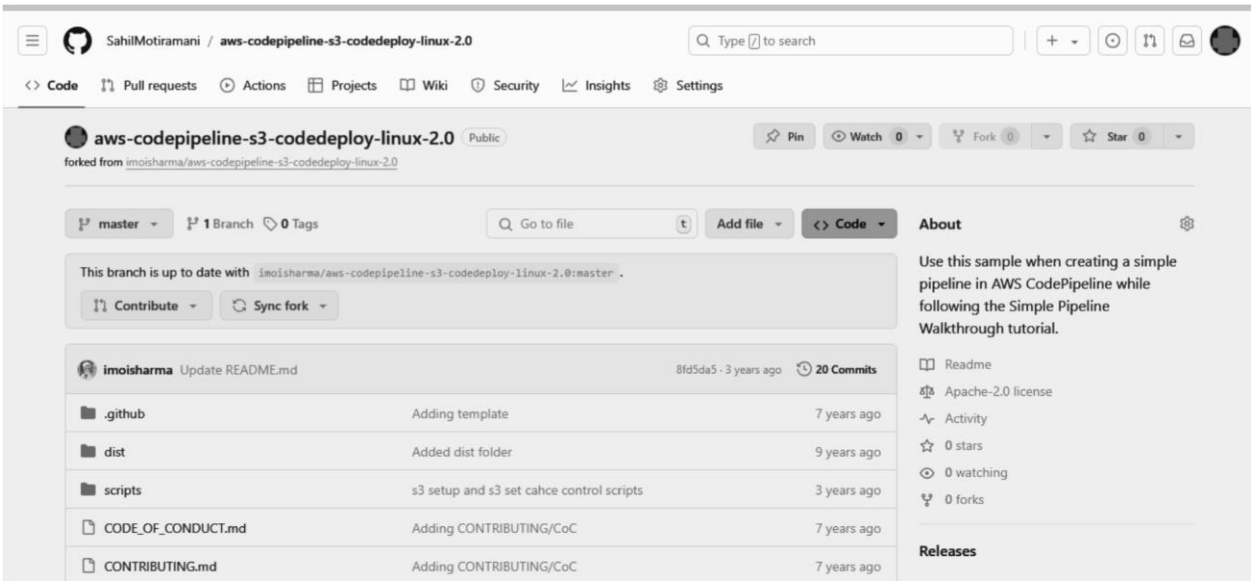
Name:Sahil Motiramani Roll No:35 Div:D15C

3.Connect to GitHub.

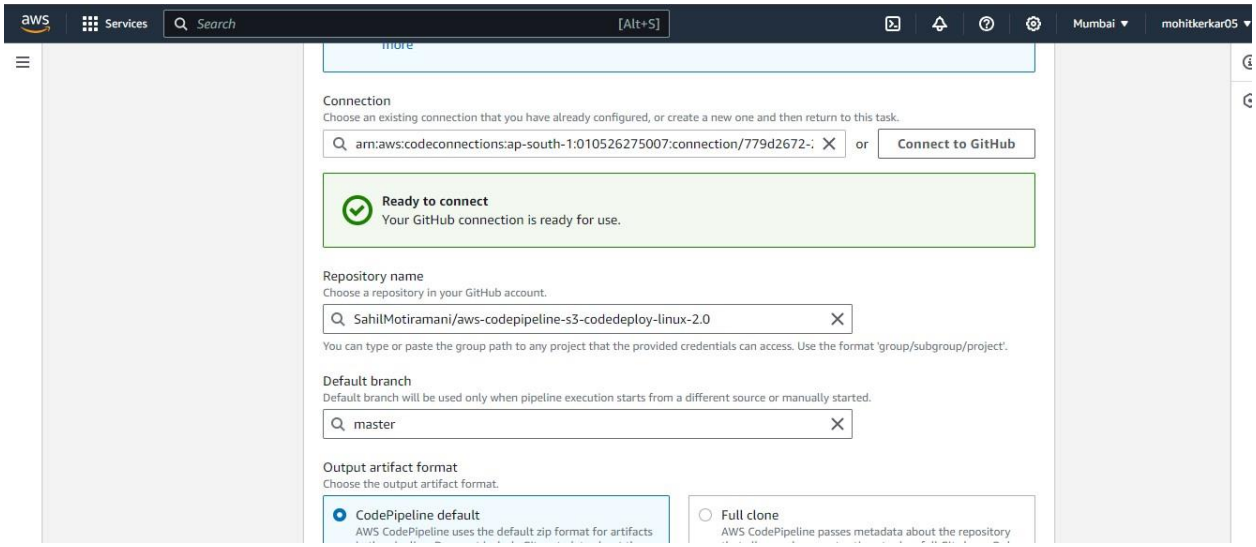


Name:Sahil Motiramani Roll No:35 Div:D15C

4.Fork the repository named aws-codepipline-s3-codeplay-linux-2.0.

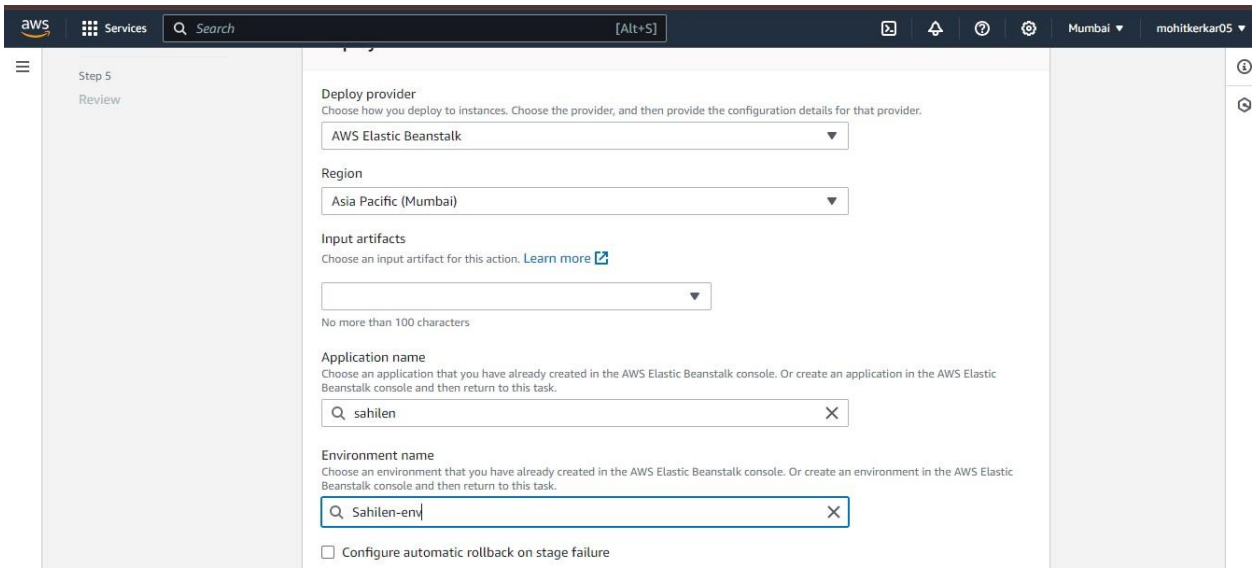


5.select the repository which you have forked.

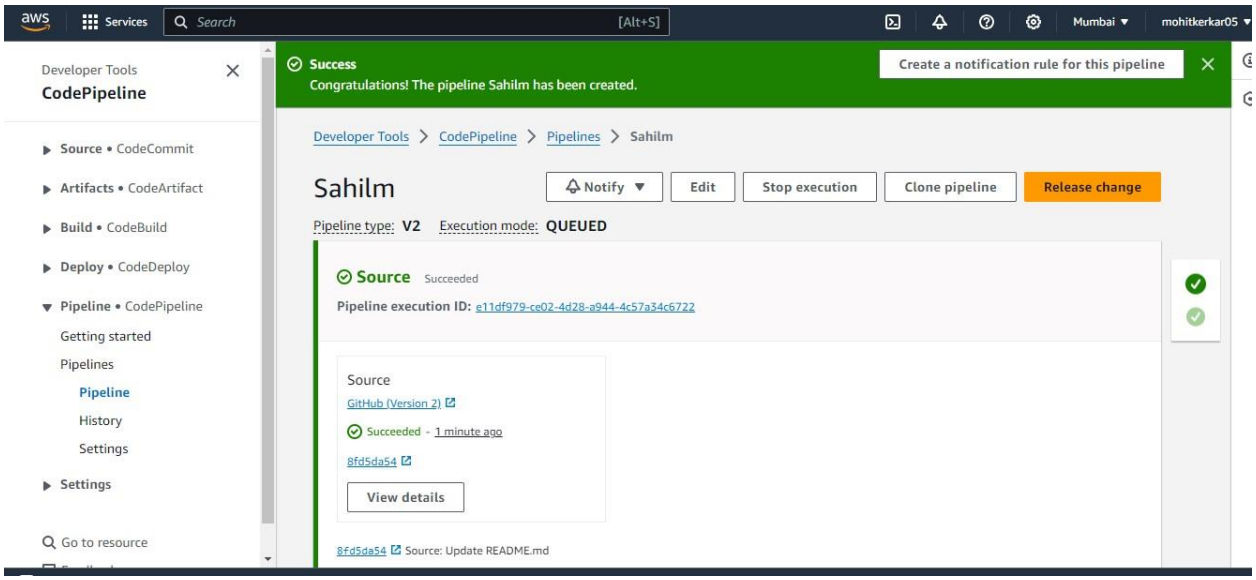


Name:Sahil Motiramani Roll No:35 Div:D15C

6.select the application and environment name.



7.The pipeline will be created.



Name:Sahil Motiramani Roll No:35 Div:D15C

8. Make some changes in the forked repository using github
Here, I have made a change by adding my name “Sahil” after “Congratulations”.

```
      h1 {
        font-size: 500%;
        font-weight: normal;
        margin-bottom: 0;
      }

      h2 {
        font-size: 200%;
        font-weight: normal;
        margin-bottom: 0;
      }
    </style>
  </head>
  <body>
    <div align="center">
      <h1>Congratulations Sahil!</h1>
      <h2>You have successfully created a pipeline that retrieved this source application from an Amazon S3 bucket and deployed it
          to three Amazon EC2 instances using AWS CodeDeploy.</h2>
      <p>For next steps, read the AWS CodePipeline Documentation. Incedge 2020</p>
    </div>
  </body>
  </html>
```

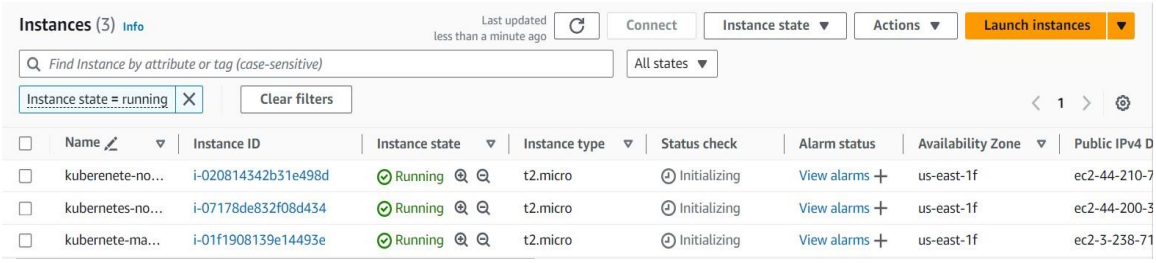
9.Commit the changes and Reload the environment the changes will be seen.

Name: Sahil Motiramani Div:D15C Roll No:35

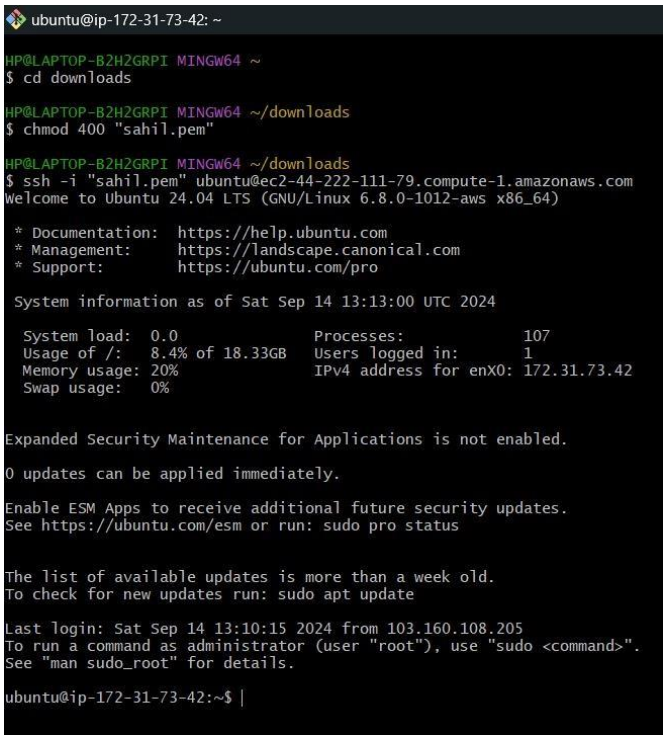
EXP No.:3

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

Step 1:Create 3 EC-2 instances with all running on Amazon Linux as OS Kubernete-master,kuberenete-node1,kubernete-node2.



Step 2:SSH into all 3 machines each in separate terminal for each instance,



Step 3:Now install docker in all 3 instances;

sudo apt-get intall -y docker.io

```
ubuntu@ip-172-31-73-42:~$ sudo apt-get install -y docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-buildx docker-compose-v2 docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc ubuntu-fan
0 upgraded, 8 newly installed, 0 to remove and 7 not upgraded.
Need to get 76.8 MB of archives.
After this operation, 289 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 pigz amd64 2.8-1 [65.6 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 bridge-utils amd64 1.7.1-1ubuntu2 [33.9 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 runc amd64 1.1.12-0ubuntu3.1 [8599 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 containerd amd64 1.7.12-0ubuntu4.1 [38.6 MB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 dns-root-data all 2023112702~willsync1 [4450 B]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 dnsmasq-base amd64 2.90-2build2 [375 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 docker.io amd64 24.0.7-0ubuntu4.1 [29.1 MB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 ubuntu-fan all 0.12.16 [35.2 kB]
```

Then, Configure File daemon.json;

```
ubuntu@ip-172-31-73-42:~$ cd /etc/docker
ubuntu@ip-172-31-73-42:/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
{
"exec-opts": ["native.cgroupdriver=systemd"],
"log-driver": "json-file",
"log-opts": {
"max-size": "100m"
},
"storage-driver": "overlay2"
}
ubuntu@ip-172-31-73-42:/etc/docker$
```

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

```
ubuntu@ip-172-31-73-42:/etc/docker$ sudo systemctl enable docker
ubuntu@ip-172-31-73-42:/etc/docker$ sudo systemctl daemon-reload
ubuntu@ip-172-31-73-42:/etc/docker$ sudo systemctl restart docker
ubuntu@ip-172-31-73-42:/etc/docker$ docker -v
Docker version 24.0.7, build 24.0.7-0ubuntu4.1
ubuntu@ip-172-31-73-42:/etc/docker$
```

Step 4:Install Kubernetes in all three instances:

```
[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/r epomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni EOF

- 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

Step 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

Step:6. For nodes only;

Use the below command on all 2 node machines

- Sudo yum install iproute-tc -y
- sudo systemctl enable kubelet
- sudo systemctl restart kubelet

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

sha256:0822f656bf52a17a2b6686c123f811306f41495ca650a0aed9bf6cd2d2f6f8

c5 Master control nodes;

```
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
```

Name:Sahil Motiramani Roll No.-35 Div:D15C

Exp:04

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

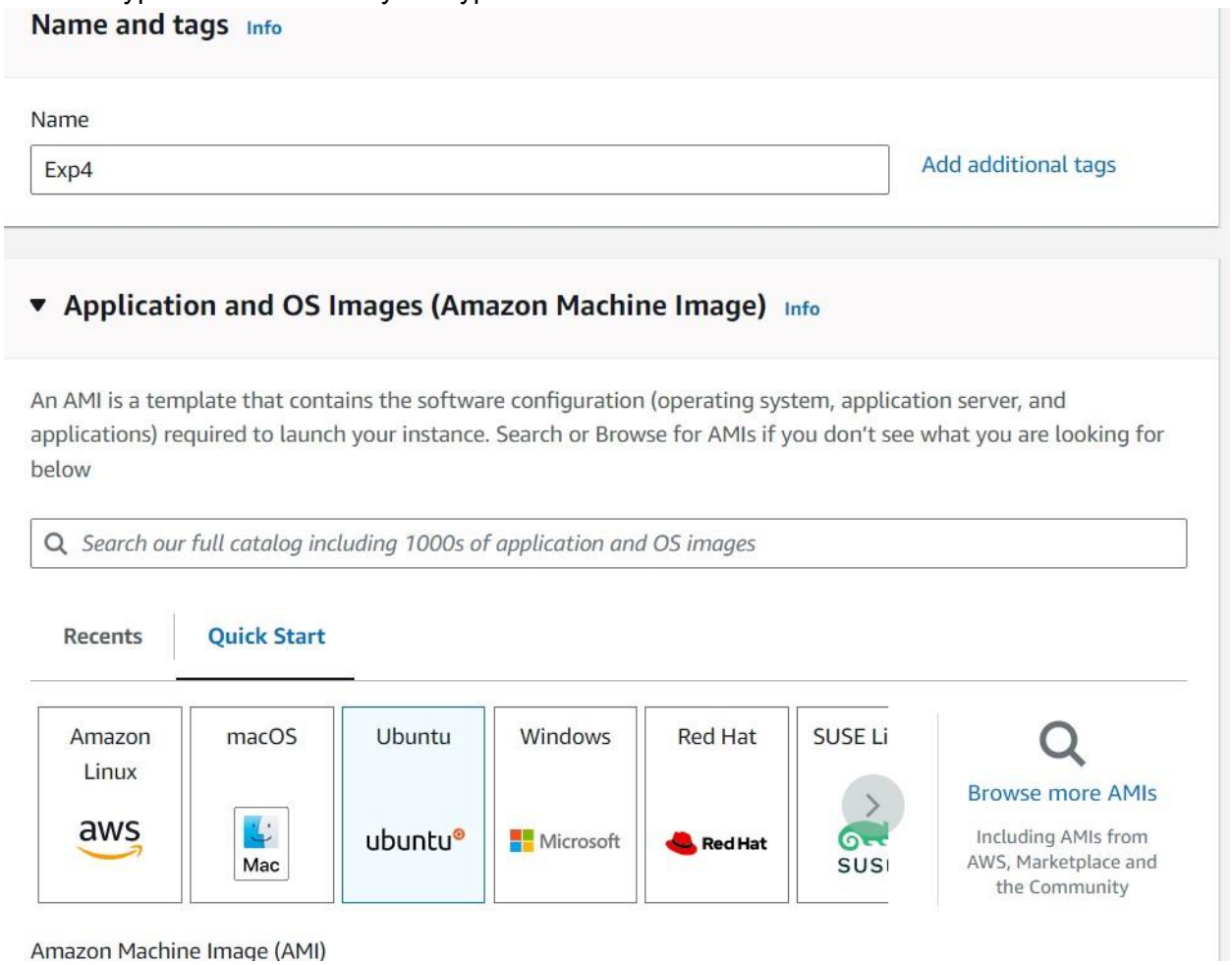
Theory `kubectl` is the command-line tool used to interact with Kubernetes clusters. It serves as the primary interface for managing and orchestrating containers in a Kubernetes environment. By sending commands to the Kubernetes API server, `kubectl` allows you to control clusters, manage workloads, and inspect resource states.

To begin using Kubernetes, installing `kubectl` is essential. The installation process varies based on the operating system (Linux, Windows, or macOS). After installing, `kubectl` connects to the Kubernetes cluster using the `kubeconfig` file, which stores details like cluster name, server address, and access credentials. With this connection established, you can use `kubectl` to perform a variety of operations, such as creating, updating, scaling, and deleting applications.

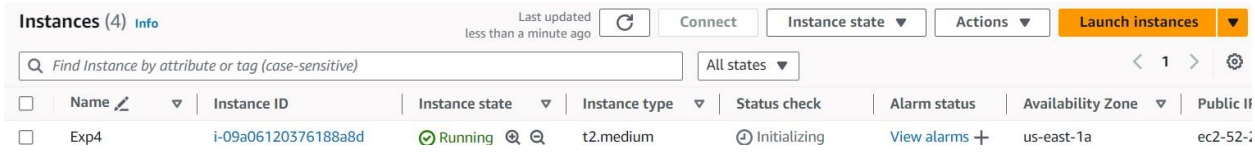
When deploying your first application with Kubernetes, the process involves defining the application in a configuration file (usually YAML) that specifies its requirements, such as images, replicas, and networking settings. `kubectl` interprets this configuration and relays it to the Kubernetes cluster, which then manages the lifecycle of the application.

Name:Sahil Motiramani Roll No.-35 Div:D15C

Step 1:Create an EC2 instance use ubuntu application and select t2 .medium category in instance type create a new key rsa type save it in local machine in an folder:

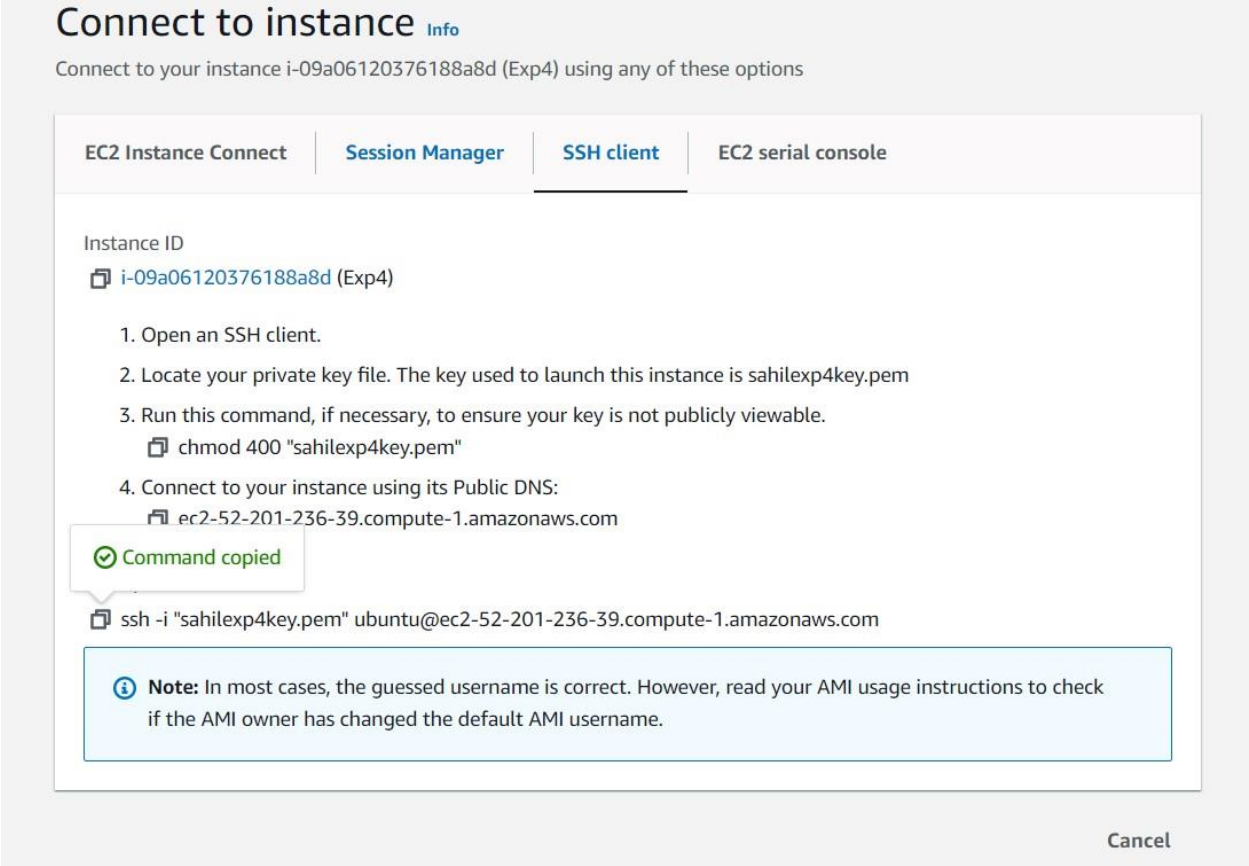


Step 2:Click create to create the instance:



Name:Sahil Motiramani Roll No.-35 Div:D15C

Step 3:
Navigate to ssh client copy the key:



Step 4:navigate to the folder open the terminal and paste the ssh command:
ssh -i "sahilexp4key.pem" ubuntu@ec2-52-201-236-39.compute-1.amazonaws.com

```
PS C:\Users\HP\Desktop\Adv devops exp4> ssh -i "sahilexp4key.pem" ubuntu@ec2-52-201-236-39.compute-1.amazonaws.com
The authenticity of host 'ec2-52-201-236-39.compute-1.amazonaws.com (52.201.236.39)' can't be established.
ED25519 key fingerprint is SHA256:nITHk14RA95WSu1ku3jdi5jaDWtYkXby4850GXZFDU8.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-52-201-236-39.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

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

 System information as of Sun Sep 15 20:25:29 UTC 2024

  System load:  0.02              Processes:             120
  Usage of /:   22.8% of 6.71GB   Users logged in:       0
  Memory usage: 5%                IPv4 address for enX0: 172.31.21.243
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

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


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

Last login: Sun Sep 15 20:25:31 2024 from 18.206.107.29
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
```

Name:Sahil Motiramani Roll No.-35 Div:D15C

Step 5:Install docker

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

```
ubuntu@ip-172-31-21-243:~$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
$(lsb_release -cs) stable"
Repository: 'deb [arch=amd64] https://download.docker.com/linux/ubuntu noble stable'
Description:
Archive for codename: noble components: stable
More info: https://download.docker.com/linux/ubuntu
Adding repository.
Press [ENTER] to continue or Ctrl-c to cancel.
Adding deb entry to /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-noble.list
Adding disabled deb-src entry to /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-noble.list
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
```

Use:

sudo apt-get update

```
ubuntu@ip-172-31-21-243:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored
ection in apt-key(8) for details.
```

Use:

sudo apt-get install -y docker-ce

```
ubuntu@ip-172-31-21-243:~$ sudo apt-get install -y docker-ce
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  containerd.io docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0 pigz slirp4netns
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite
The following NEW packages will be installed:
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0 pigz slirp4netns
0 upgraded, 10 newly installed, 0 to remove and 133 not upgraded.
Need to get 122 MB of archives.
After this operation, 440 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 pigz amd64 2.8-1 [65.6 kB]
```

Now the docker is installed;

Now lets enable the docker:

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

```
ubuntu@ip-172-31-21-243:~$ sudo mkdir -p /etc/docker
ubuntu@ip-172-31-21-243:~$ cat <<EOF | sudo tee /etc/docker/daemon.json {
"exec-opts": ["native.cgroupdriver=systemd"] } EOF
```

sudo systemctl enable docker

Name:Sahil Motiramani Roll No.-35 Div:D15C

```
ubuntu@ip-172-31-21-243:~$ sudo systemctl enable docker
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
ubuntu@ip-172-31-21-243:~$
```

sudo systemctl daemon-reload
sudo systemctl restart docker

```
ubuntu@ip-172-31-21-243:~$ sudo systemctl daemon-reload
ubuntu@ip-172-31-21-243:~$
sudo systemctl restart docker
ubuntu@ip-172-31-21-243:~$
```

Step 6:Now lets install kubernetes; curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

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

```
ubuntu@ip-172-31-21-243:~$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyri
ng.gpg
ubuntu@ip-172-31-21-243:~$ echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /et
c/apt/sources.list.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
ubuntu@ip-172-31-21-243:~$
```

sudo apt-get update

```
ubuntu@ip-172-31-21-243:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  InRelease [1186 B]
Hit:6 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:7 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  Packages [4865 B]
Fetched 6051 B in 0s (12.6 kB/s)
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring (/etc
ection in apt-key(8) for details.
ubuntu@ip-172-31-21-243:~$
```

sudo apt-get install -y kubelet kubeadm kubectl

```
ubuntu@ip-172-31-21-243:~$ sudo apt-get install -y kubelet kubeadm kubectl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  conntrack cri-tools kubernetes-cni
The following NEW packages will be installed:
  conntrack cri-tools kubeadm kubectl kubelet kubernetes-cni
0 upgraded, 6 newly installed, 0 to remove and 133 not upgraded.
Need to get 87.4 MB of archives.
After this operation, 314 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 conntrack amd64 1:1.4.8-1ubuntu1 [37.9 kB]
Get:2 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  cri-tools 1.31.1-1.1 [15.7 MB]
Get:3 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  kubeadm 1.31.1-1.1 [11.4 MB]
Get:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  kubectl 1.31.1-1.1 [11.2 MB]
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  kubernetes-cni 1.5.1-1.1 [33.9 MB]
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  kubelet 1.31.1-1.1 [15.2 MB]
Fetched 87.4 MB in 1s (71.7 MB/s)
Selecting previously unselected package conntrack.
(Reading database ... 68007 files and directories currently installed.)
Preparing to unpack .../0-conntrack_1%3a1.4.8-1ubuntu1_amd64.deb ...
```

sudo apt-mark hold kubelet kubeadm kubectl

Name:Sahil Motiramani Roll No.-35 Div:D15C

```
ubuntu@ip-172-31-21-243:~$ sudo apt-mark hold kubelet kubeadm kubectl
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
ubuntu@ip-172-31-21-243:~$
```

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

```
ubuntu@ip-172-31-21-243:~$ sudo systemctl enable --now kubelet
ubuntu@ip-172-31-21-243:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
W0915 20:41:32.398638    4595 checks.go:1080] [preflight] WARNING: Couldn't create
 new CRI runtime service: validate service connection: validate CRI v1 runtime API
e = Unimplemented desc = unknown service runtime.v1.RuntimeService
        [WARNING FileExisting-socat]: socat not found in system path
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your interne
[preflight] You can also perform this action beforehand using 'kubeadm config image
error execution phase preflight: [preflight] Some fatal errors occurred:
failed to create new CRI runtime service: validate service connection: validate CRI
: rpc error: code = Unimplemented desc = unknown service runtime.v1.RuntimeService[
 with `--ignore-preflight-errors=...`
To see the stack trace of this error execute with --v=5 or higher
ubuntu@ip-172-31-21-243:~$
```

sudo apt-get install -y containerd

```
ubuntu@ip-172-31-21-243:~$ sudo apt-get install -y containerd
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0 pigz slirp4netn
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  runc
The following packages will be REMOVED:
  containerd.io docker-ce
The following NEW packages will be installed:
  containerd runc
0 upgraded, 2 newly installed, 2 to remove and 133 not upgraded.
```

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

Name:Sahil Motiramani Roll No.-35 Div:D15C

```
ubuntu@ip-172-31-21-243:~$ sudo mkdir -p /etc/containerd
ubuntu@ip-172-31-21-243:~$ sudo containerd config default | sudo tee /etc/containerd/config.toml
disabled_plugins = []
imports = []
oom_score = 0
plugin_dir = ""
required_plugins = []
root = "/var/lib/containerd"
state = "/run/containerd"
temp = ""
version = 2

[cgroup]
  path = ""

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

[grpc]
  address = "/run/containerd/containerd.sock"
  gid = 0
  max_recv_message_size = 16777216
  max_send_message_size = 16777216
  tcp_address = ""
  tcp_tls_ca = ""
  tcp_tls_cert = ""
  tcp_tls_key = ""
  uid = 0
```

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

```
ubuntu@ip-172-31-21-243:~$ sudo systemctl restart containerd
sudo systemctl enable containerd
sudo systemctl status containerd
● containerd.service - containerd container runtime
     Loaded: loaded (/usr/lib/systemd/system/containerd.service; enabled; preset: enabled)
     Active: active (running) since Sun 2024-09-15 20:45:11 UTC; 228ms ago
       Docs: https://containerd.io
   Main PID: 5089 (containerd)
      Tasks: 7
     Memory: 13.9M (peak: 14.3M)
        CPU: 58ms
     CGroup: /system.slice/containerd.service
             └─5089 /usr/bin/containerd

Sep 15 20:45:11 ip-172-31-21-243 containerd[5089]: time="2024-09-15T20:45:11.275924779Z" level=info msg=servi
Sep 15 20:45:11 ip-172-31-21-243 containerd[5089]: time="2024-09-15T20:45:11.275966419Z" level=info msg=servi
Sep 15 20:45:11 ip-172-31-21-243 containerd[5089]: time="2024-09-15T20:45:11.275967171Z" level=info msg="Star
Sep 15 20:45:11 ip-172-31-21-243 containerd[5089]: time="2024-09-15T20:45:11.276021207Z" level=info msg="Star
Sep 15 20:45:11 ip-172-31-21-243 containerd[5089]: time="2024-09-15T20:45:11.276080044Z" level=info msg="Star
Sep 15 20:45:11 ip-172-31-21-243 containerd[5089]: time="2024-09-15T20:45:11.276092900Z" level=info msg="Star
Sep 15 20:45:11 ip-172-31-21-243 containerd[5089]: time="2024-09-15T20:45:11.276101205Z" level=info msg="Star
Sep 15 20:45:11 ip-172-31-21-243 containerd[5089]: time="2024-09-15T20:45:11.276111208Z" level=info msg="Star
Sep 15 20:45:11 ip-172-31-21-243 containerd[5089]: time="2024-09-15T20:45:11.276200348Z" level=info msg="conta
Sep 15 20:45:11 ip-172-31-21-243 systemd[1]: Started containerd.service - containerd container runtime.
lines 1-21/21 (END)
```

sudo apt-get install -y socat

Name:Sahil Motiramani Roll No.-35 Div:D15C

```
ubuntu@ip-172-31-21-243:~$ sudo apt-get install -y socat
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plugin
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  socat
0 upgraded, 1 newly installed, 0 to remove and 133 not upgraded.
Need to get 374 kB of archives.
After this operation, 1649 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 socat amd64 1.
Fetched 374 kB in 0s (10.9 MB/s)
Selecting previously unselected package socat.
(Reading database ... 68107 files and directories currently installed.)
Preparing to unpack .../socat_1.8.0.0-4build3_amd64.deb ...
Unpacking socat (1.8.0.0-4build3) ...
Setting up socat (1.8.0.0-4build3) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

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

Step 7: Intitialize the kubernete:
sudo kubeadm init --pod-network-cidr=10.244.0.0/16

```
ubuntu@ip-172-31-21-243:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action beforehand using 'kubeadm config images pull'
W0915 20:47:01.807699    5328 checks.go:846] detected that the sandbox image "registry.k8s.io/pa
used by kubeadm.It is recommended to use "registry.k8s.io/pause:3.10" as the CRI sandbox image.
```

```
ubuntu@ip-172-31-21-243:~$ mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
ubuntu@ip-172-31-21-243:~$
```

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

Name:Sahil Motiramani Roll No.-35 Div:D15C

```
ubuntu@ip-172-31-21-243:~$ 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
ubuntu@ip-172-31-21-243:~$
```

Step 8:Now we can deploy our nginx server on this cluster using following steps:
kubectl apply -f https://k8s.io/examples/application/deployment.yaml

```
ubuntu@ip-172-31-21-243:~$ kubectl apply -f https://k8s.io/examples/application/deployment.yaml
deployment.apps/nginx-deployment created
ubuntu@ip-172-31-21-243:~$
```

kubectl get pods

```
ubuntu@ip-172-31-21-243:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-54h6b    0/1     Pending   0          28s
nginx-deployment-d556bf558-jw5xg    0/1     Pending   0          28s
ubuntu@ip-172-31-21-243:~$
```

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

```
ubuntu@ip-172-31-21-243:~$ POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
ubuntu@ip-172-31-21-243:~$ kubectl port-forward $POD_NAME 8080:80
error: unable to forward port because pod is not running. Current status=Pending
ubuntu@ip-172-31-21-243:~$
```

kubectl taint nodes --all node-role.kubernetes.io/control-plane-node/ip-172-31-20-171 untainted
kubectl get nodes

```
ubuntu@ip-172-31-21-243:~$ kubectl taint nodes --all node-role.kubernetes.io/control-plane-node/ip-172-31-20-171 untainted
error: at least one taint update is required
ubuntu@ip-172-31-21-243:~$ kubectl get nodes
NAME               STATUS   ROLES           AGE     VERSION
ip-172-31-21-243   Ready    control-plane   6m14s   v1.31.1
ubuntu@ip-172-31-21-243:~$
```

kubectl get pods

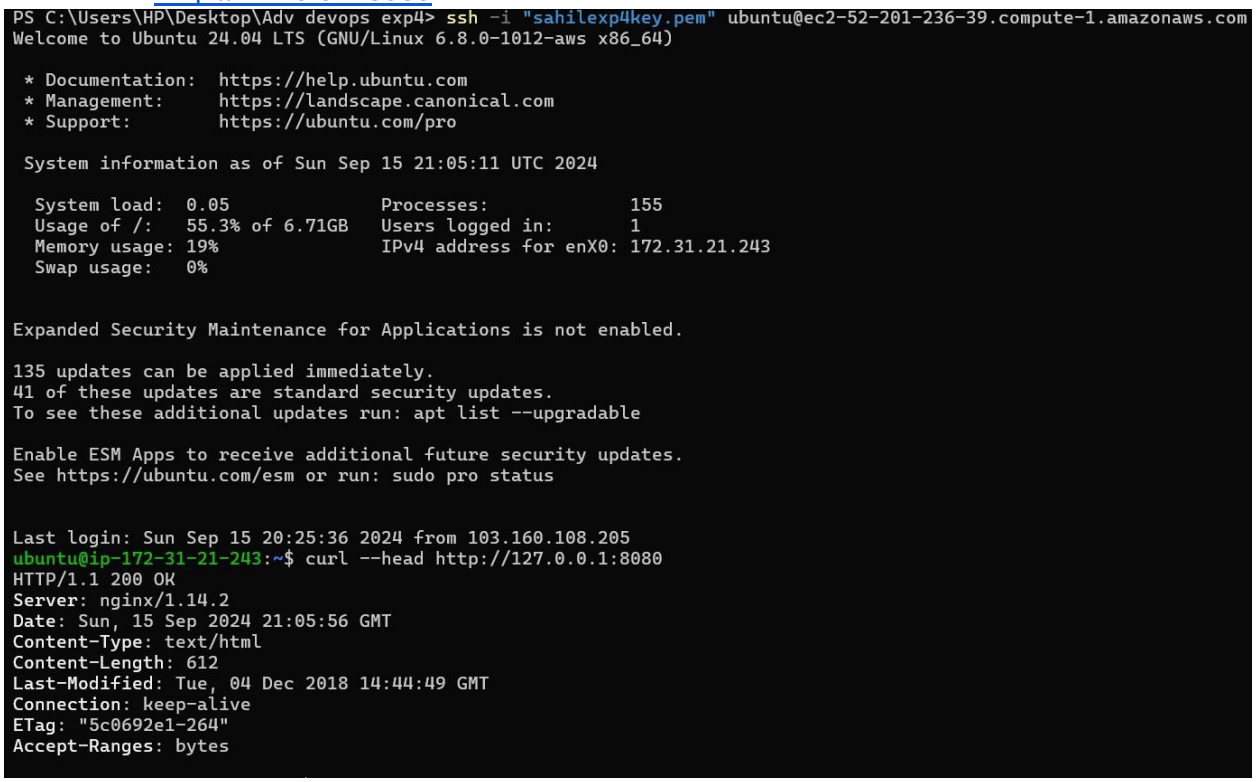
```
ubuntu@ip-172-31-21-243:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-54h6b    0/1     Pending   0          2m31s
nginx-deployment-d556bf558-jw5xg    0/1     Pending   0          2m31s
ubuntu@ip-172-31-21-243:~$
```

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

```
ubuntu@ip-172-31-21-243:~$ POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
kubectl port-forward $POD_NAME 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
```

Name:Sahil Motiramani Roll No.-35 Div:D15C

Step 9 :check deployment:
Open new terminal in folder,
Paste the ssh key,
Type
curl --head http://127.0.0.1:8080



Now we have successfully deployed our nginx server on our ec2 instance.

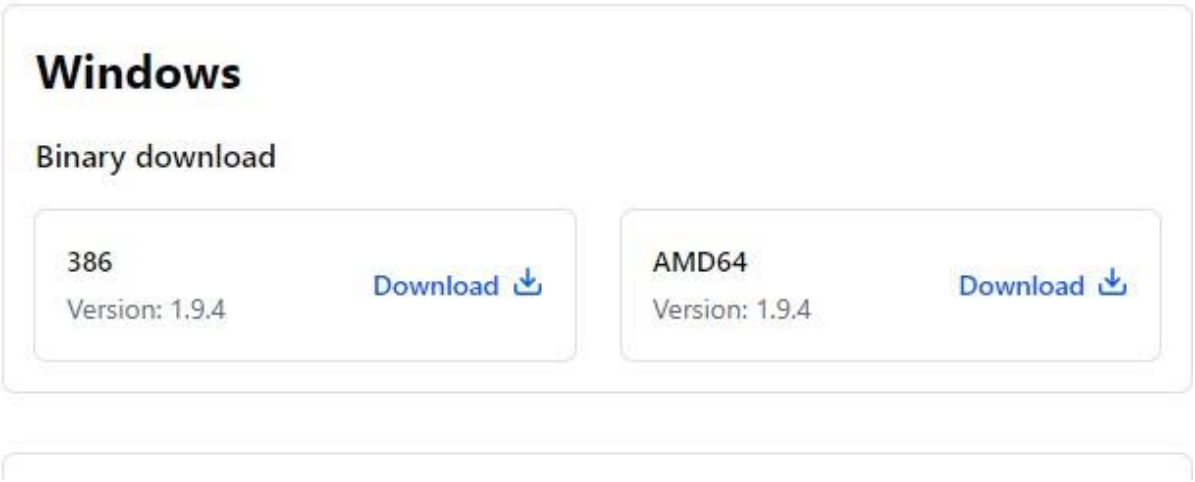
Conclusion

Installing `kubectl` and using it effectively is a crucial part of managing Kubernetes clusters. As the main interface to Kubernetes, `kubectl` empowers you to deploy, monitor, and troubleshoot applications, providing full control over cluster resources. Deploying an application for the first time serves as an introduction to Kubernetes' ability to orchestrate containers seamlessly. It showcases the power of declarative configurations and automated scaling, which are central to Kubernetes' efficiency in managing modern applications. Understanding how to install and operate `kubectl` lays the foundation for more advanced interactions with Kubernetes, enabling both developers and administrators to harness its full potential in creating resilient, scalable, and portable containerized applications. This knowledge is fundamental for any professional aiming to work within a Kubernetes-driven infrastructure.

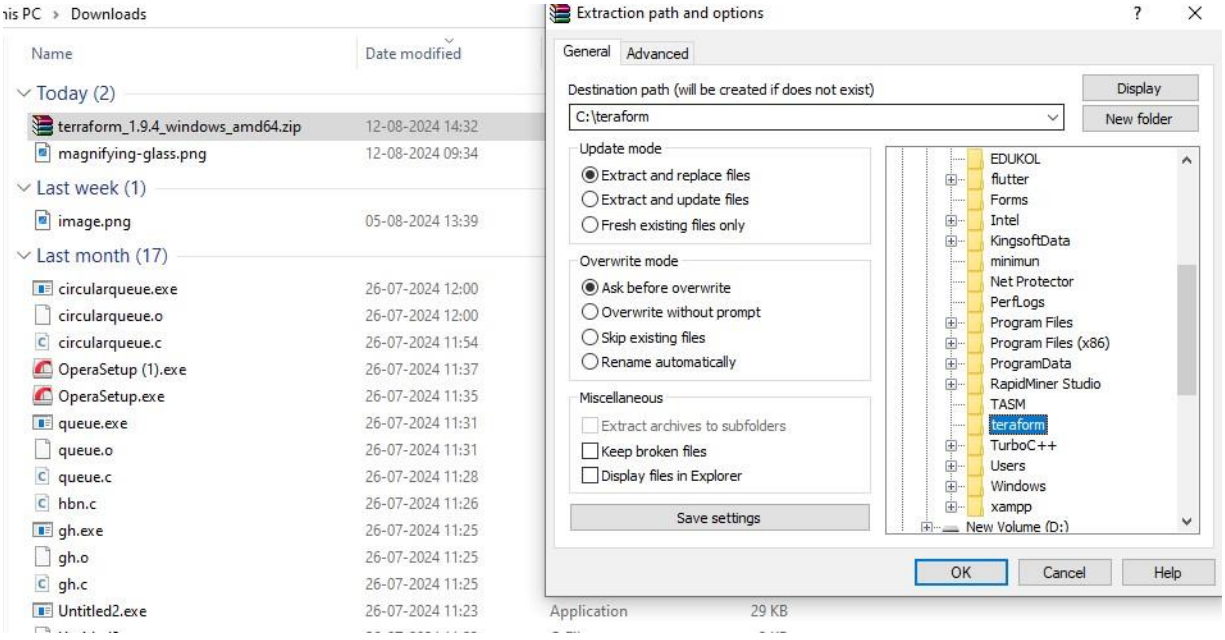
Name:Sahil Motiramani Roll No. 35 Div:D15C

Exp 05: To understand terraform lifecycle, core concepts/terminologies and install it on a Linux Machine.

Step1: Go to hashicorp.com page and download the AMD 64 file for terraform.Install it in Windows.



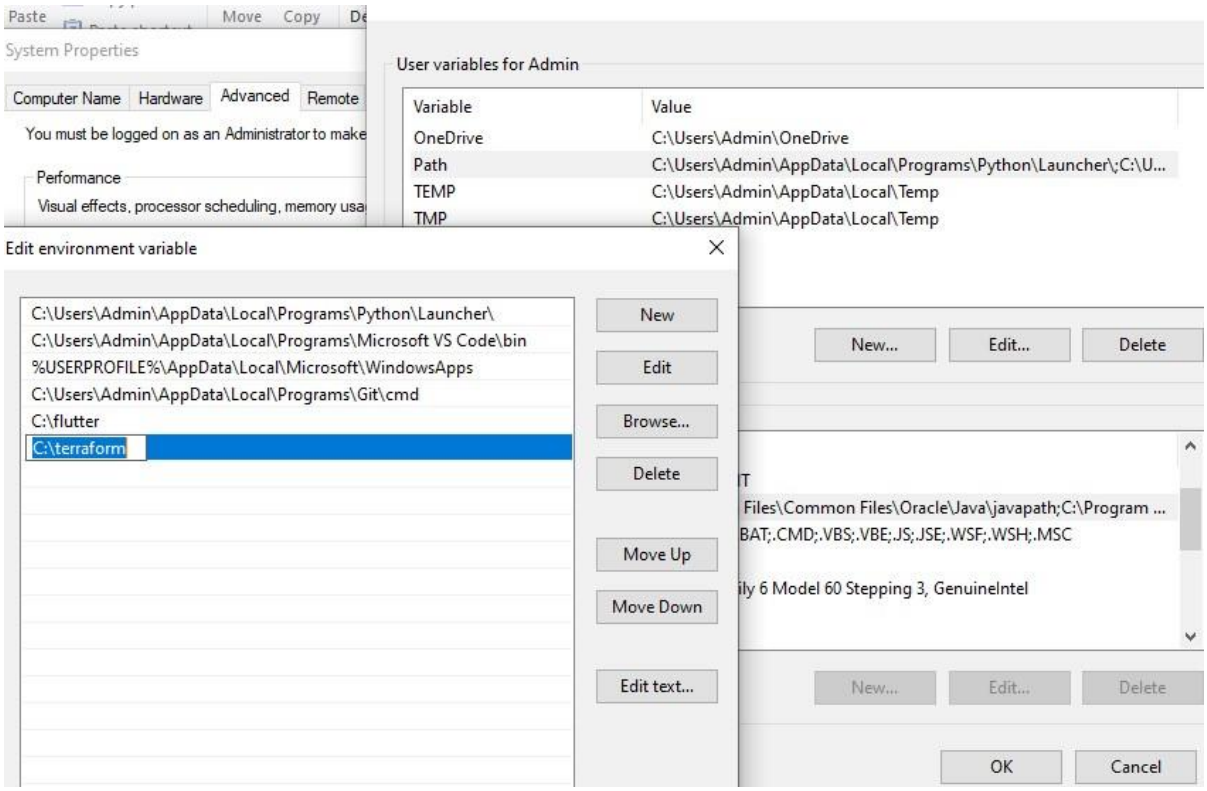
Step 2:After Downloading Extract the file and Save it inside the C drive by naming the folder terraform.



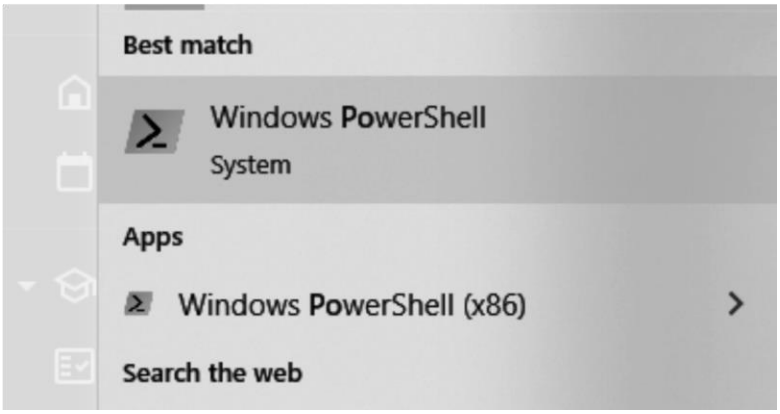
\

Name:Sahil Motiramani Roll No. 35 Div:D15C

Step 3: Edit Path inside Environment Variables Add the new path of terraform I.e C:\terraform



Step 4: Open Windows PowerShell



Name:Sahil Motiramani Roll No. 35 Div:D15C

Step 5. Type terraform in PowerShell if There are no errors then that means the Terraform is installed successfully or else there will be a error in Environment Variables.

Name:Sahil Motiramani Roll No.:35 Div:D15C

A. Creating docker image using terraform

Step 1: Download Docker and Check the docker functionality

```
C:\Users\student>docker

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

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

Management Commands:
  builder     Manage builds
  buildx*     Docker Buildx
  checkpoint  Manage checkpoints
  compose*    Docker Compose
  container   Manage containers
  context     Manage contexts
  debug*      Get a shell into any image or container
  desktop*    Docker Desktop commands (Alpha)
  dev*        Docker Dev Environments
  extension*  Manages Docker extensions
  feedback*   Provide feedback, right in your terminal!
```

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

C:\Users\student>
```

Step 2:Create a folder named ‘TerraformScripts’

Then,go in that folder and again create a folder named ‘Docker’ Then,create a file named docker.tf And write the following code in it.

```
terraform{
        required_providers{
                docker = {
                        source = "kreuzwerker/docker"
                        version = "2.21.0"
                        }
                }
        }
        provider "docker" {
        host = "npipe:////.//pipe//docker_engine"
        }
        # Pulls the image
        resource "docker_image" "ubuntu"{
                name = "ubuntu:latest"
        }
# Create a container
        resource "docker_container" "foo"{
                image =docker_image.ubuntu.image_id
                name = "foo"
                command = ["sleep", "3600"]
        }
```

Name:Sahil Motiramani Roll No.:35 Div:D15C

Step 3: Run the command terraform init.

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

Terraform has been successfully initialized!

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

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

Step 4:Run the terraform plan to see all the files initialized.

```
C:\Users\INFT\Desktop\TerraformScript\Docker>terraform plan

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

Terraform will perform the following actions:

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

Name:Sahil Motiramani Roll No.:35 Div:D15C

```
      + 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.

------------------------------------------------------------------------

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

C:\Users\INFT\Desktop\TerraformScript\Docker>
```

This is image before apply:

```
C:\Users\INFT\Desktop\TerraformScript\Docker>docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE

C:\Users\INFT\Desktop\TerraformScript\Docker>
```

Step 5:execute the command terraform apply:
It will apply the configuration

```
C:\Users\INFT\Desktop\TerraformScript\Docker>docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
ubuntu       latest    edbfe74c41f8   2 weeks ago   78.1MB

C:\Users\INFT\Desktop\TerraformScript\Docker>
```

Name:Sahil Motiramani Roll No.:35 Div:D15C

```
C:\Users\INFT\Desktop\TerraformScript\Docker>terraform apply

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

Terraform will perform the following actions:

  # docker_container.foo will be created
  + resource "docker_container" "foo" {
      + attach           = false
      + bridge           = (known after apply)
      + command          = (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
```

```
      + 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)
    }

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=6b4b627b5597135995916aaa25dad8226b7961c9b0db969caf119b02ffbc06ea]

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

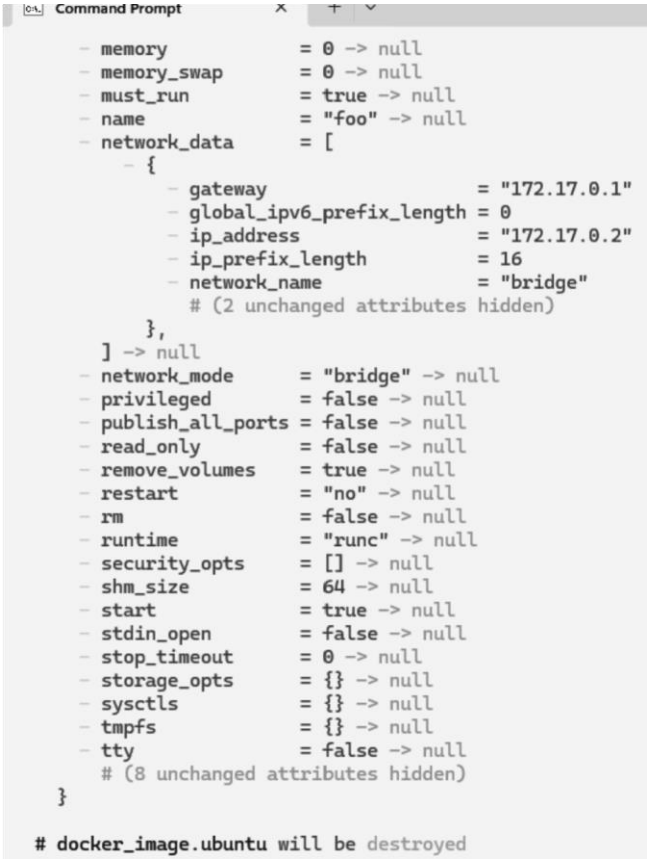
Docker images,After executing terraform apply:

Step 6:Execute terraform destroy to delete the configurations;

It will automatically delete the container:

Name:Sahil Motiramani Roll No.:35 Div:D15C

```
C:\Users\INFT\Desktop\TerraformScript\Docker>terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Refreshing state... [id=6b4b627b5597135995916aaa25dad8226b7961c9b0db969caf119b02ffbc06ea]

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

Terraform will perform the following actions:

  # docker_container.foo will be destroyed
  - resource "docker_container" "foo" {
      - attach            = false -> null
      - command           = [
          - "sleep",
          - "3600",
        ] -> null
      - cpu_shares        = 0 -> null
      - dns               = [] -> null
      - dns_opts          = [] -> null
      - dns_search        = [] -> null
      - entrypoint        = [] -> null
      - env               = [] -> null
      - gateway           = "172.17.0.1" -> null
      - group_add         = [] -> null
      - hostname          = "6b4b627b5597" -> null
      - id                = "6b4b627b5597135995916aaa25dad8226b7961c9b0db969caf119b02ffbc06ea" -> null
      - image             = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a" -> null
      - init              = false -> null
      - ip_address        = "172.17.0.2" -> null
      - ip_prefix_length  = 16 -> null
      - ipc_mode          = "private" -> null
      - links             = [] -> null
      - log_driver        = "json-file" -> null
      - log_opts          = {} -> null
      - logs              = false -> null
```



Name:Sahil Motiramani Roll No.:35 Div:D15C

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

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

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

  Enter a value: yes

docker_container.foo: Destroying... [id=6b4b627b5597135995916aaa25dad8226b7961c9b0db969caf119b02ffbc06ea]
docker_container.foo: Destruction complete after 0s
docker_image.ubuntu: Destroying... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_image.ubuntu: Destruction complete after 0s

Destroy complete! Resources: 2 destroyed.
```

Docker images After destroying,

```
C:\Users\INFT\Desktop\TerraformScript\Docker>docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE

C:\Users\INFT\Desktop\TerraformScript\Docker>
```

Name:Sahil Motiramani Div:D15C Roll No.35

Adv DevOps Practical 7

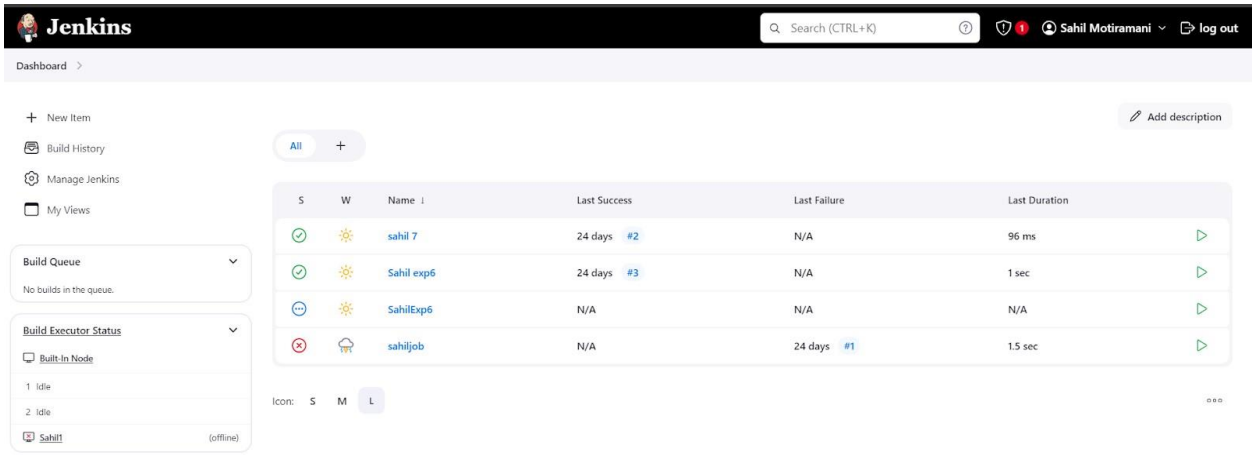
Aim: To understand Static Analysis SAST process and learn to integrate Jenkins SAST to SonarQube/GitLab.

Integrating Jenkins with SonarQube:

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

Steps to integrate Jenkins with SonarQube

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



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

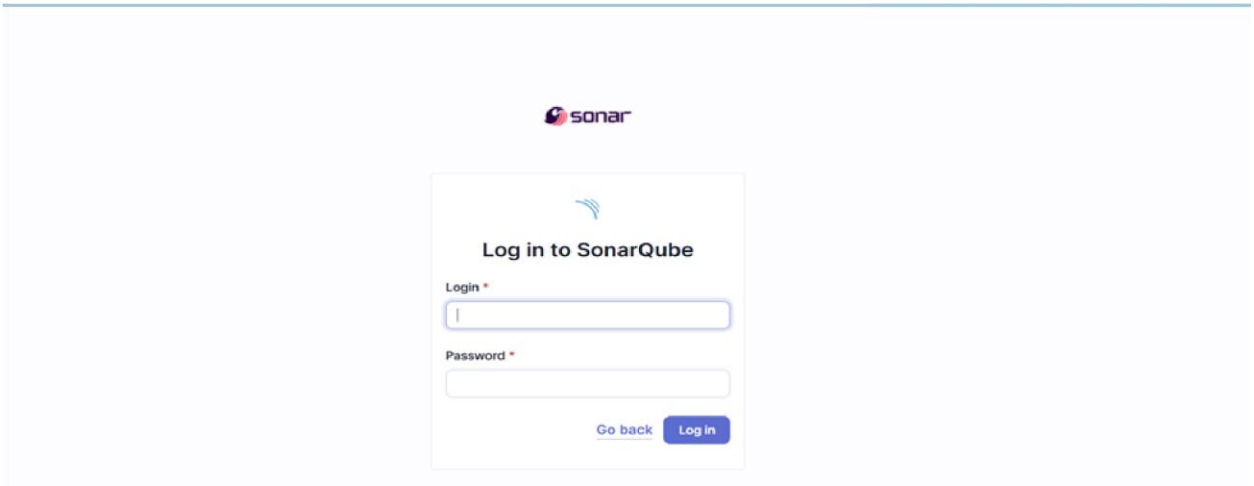
docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest

Name:Sahil Motiramani Div:D15C Roll No.35

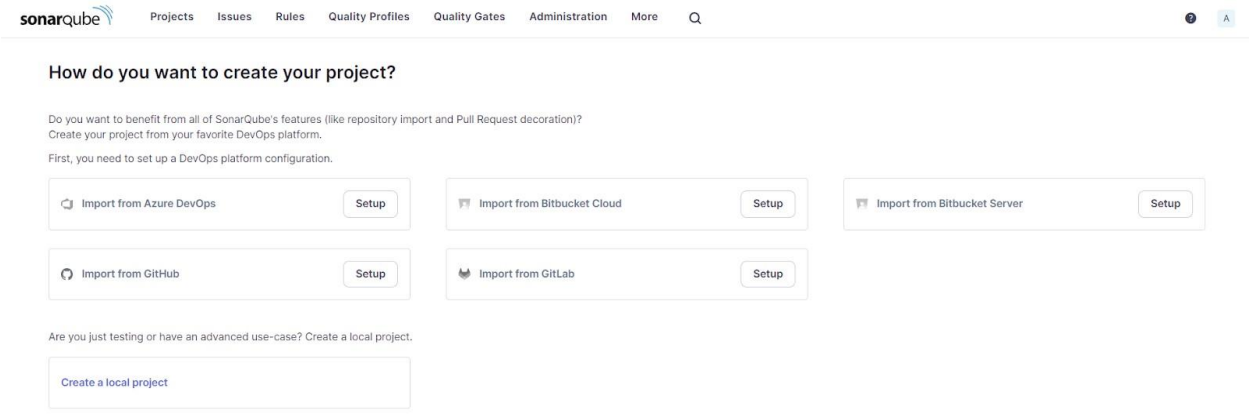
----------Warning: run below command only once

```
C:\Windows\System32>docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:late
st
Unable to find image 'sonarqube:latest' locally
latest: Pulling from library/sonarqube
7478e0ac0f23: Pull complete
90a925ab929a: Pull complete
7d9a34308537: Pull complete
80338217a4ab: Pull complete
1a5fd5c7e184: Pull complete
7b87d6fa783d: Pull complete
bd819c9b5ead: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:72e9feec71242af83faf65f95a40d5e3bb2822a6c3b2cda8568790f3d31aecde
Status: Downloaded newer image for sonarqube:latest
c1c57f6ace6123c4160745bf57640c75fcbfe70225eabb02b7bf7a40450e8001
```

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

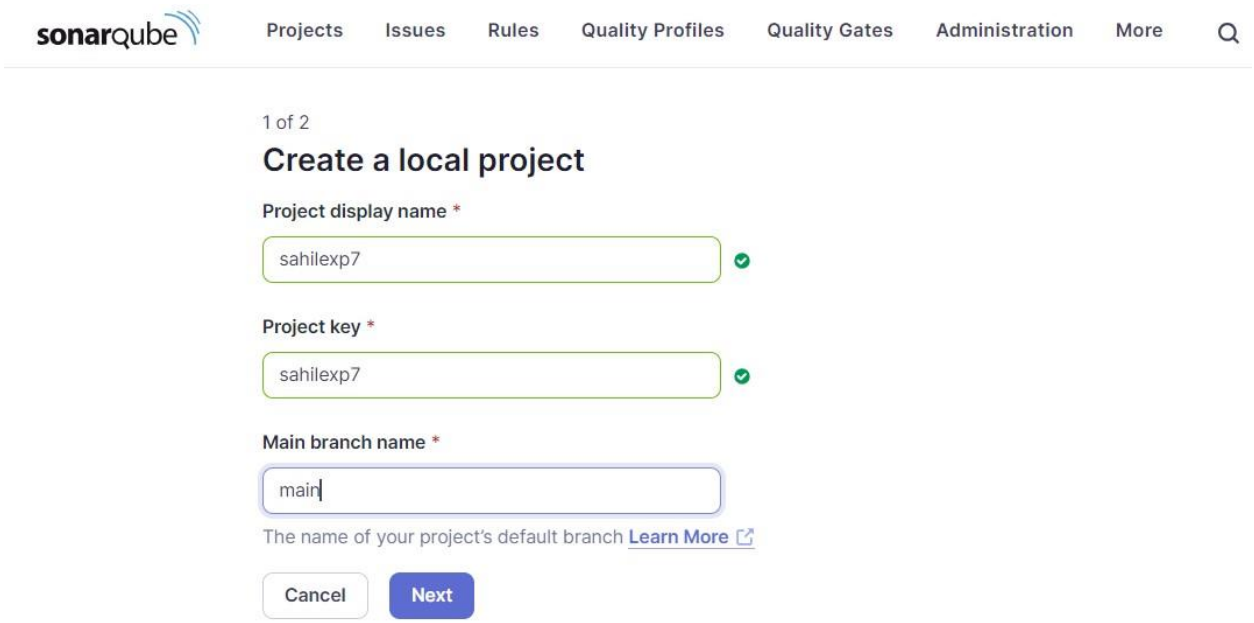


4. Login to SonarQube using username admin and password admin.



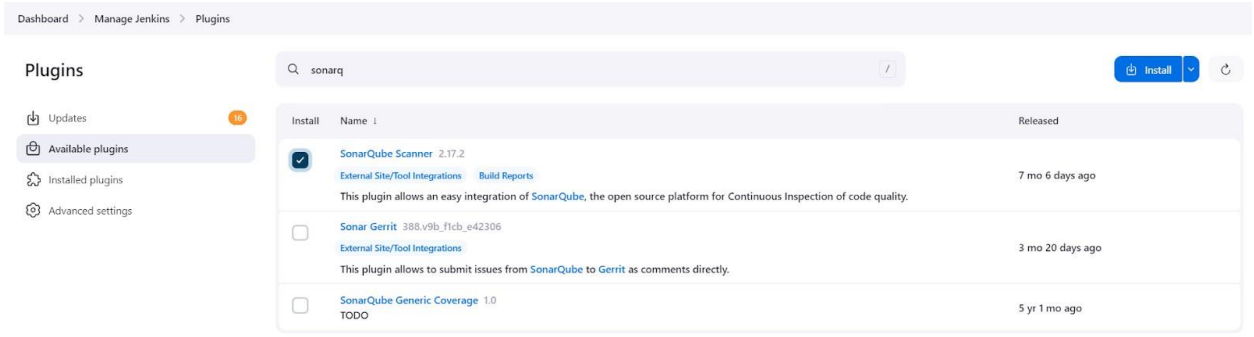
5. Create a manual project in SonarQube with the name sonarqube

Name:Sahil Motiramani Div:D15C Roll No.35



Setup the project and come back to Jenkins Dashboard.

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

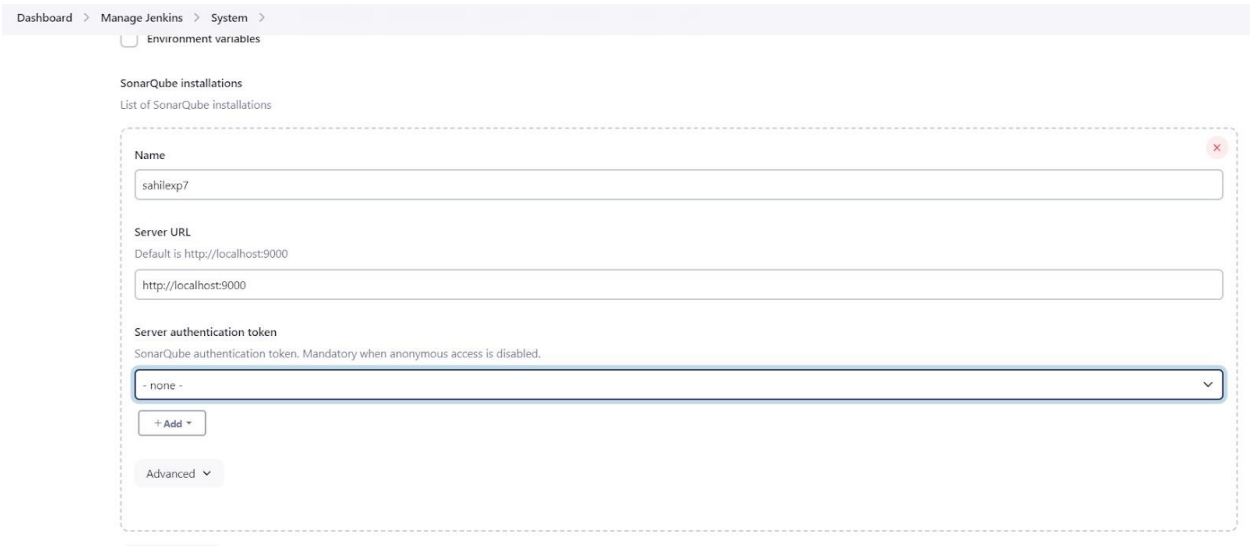


6. Under Jenkins 'Manage Jenkins' then go to 'system', scroll and look for **SonarQube Servers** and enter the details.

Enter the Server Authentication token if needed.

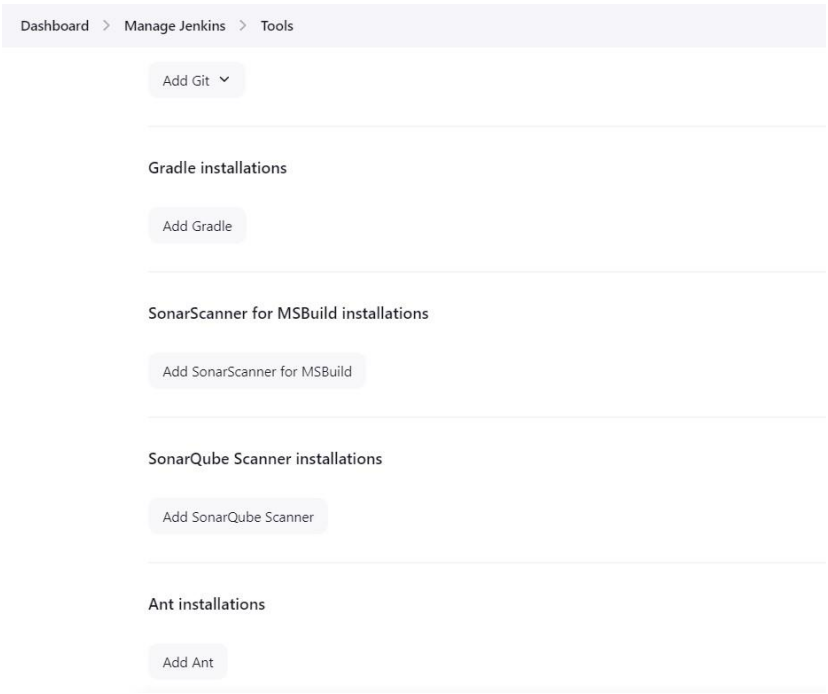
In SonarQube installations: Under **Name** add <project name of sonarqube> for me **sahilexp7**

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



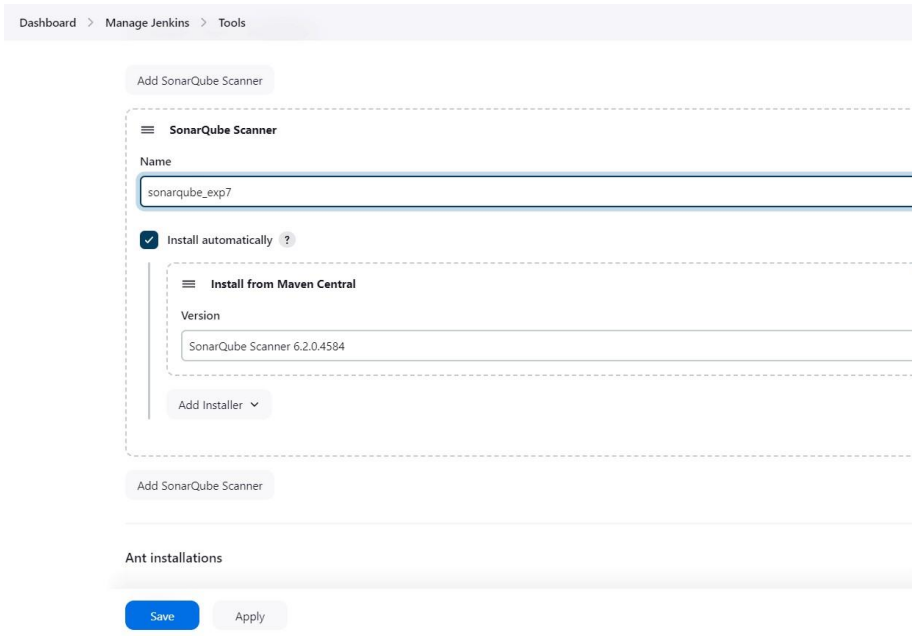
7. Search for SonarQube Scanner under Global Tool Configuration. Choose the latest configuration and choose Install automatically.

Dashboard > Manage Jenkins > Tools

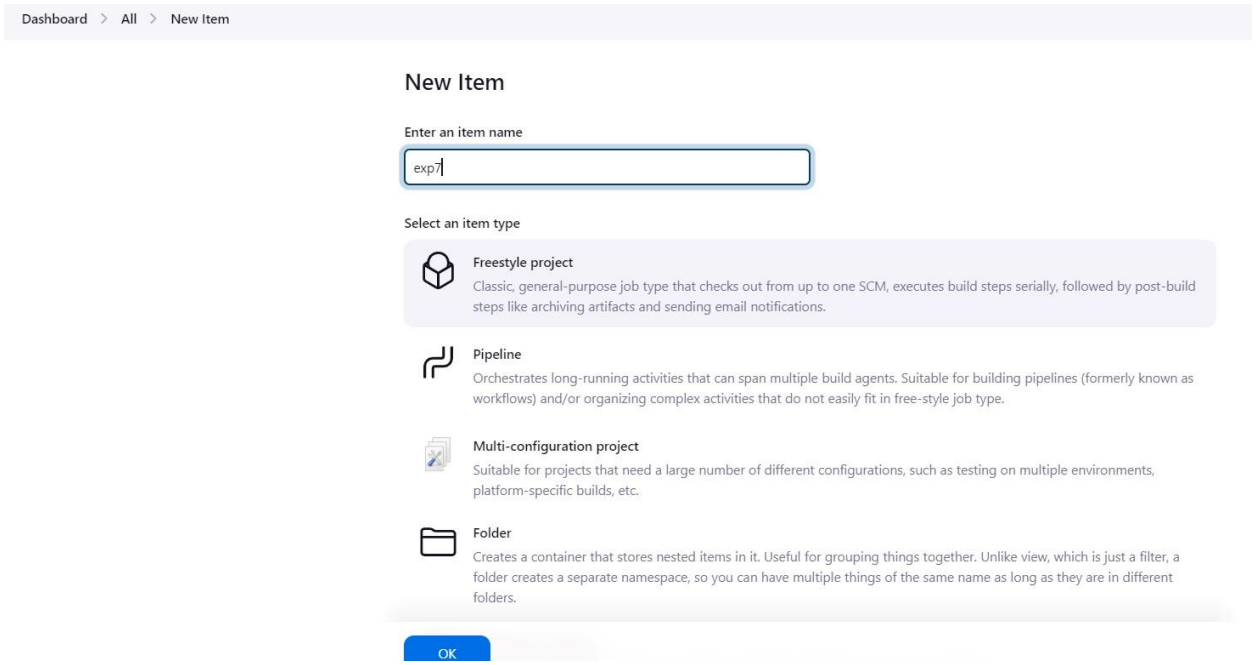


Check the “Install automatically” option. → Under name any name as identifier → Check the “Install automatically” option.

Name:Sahil Motiramani Div:D15C Roll No.35



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

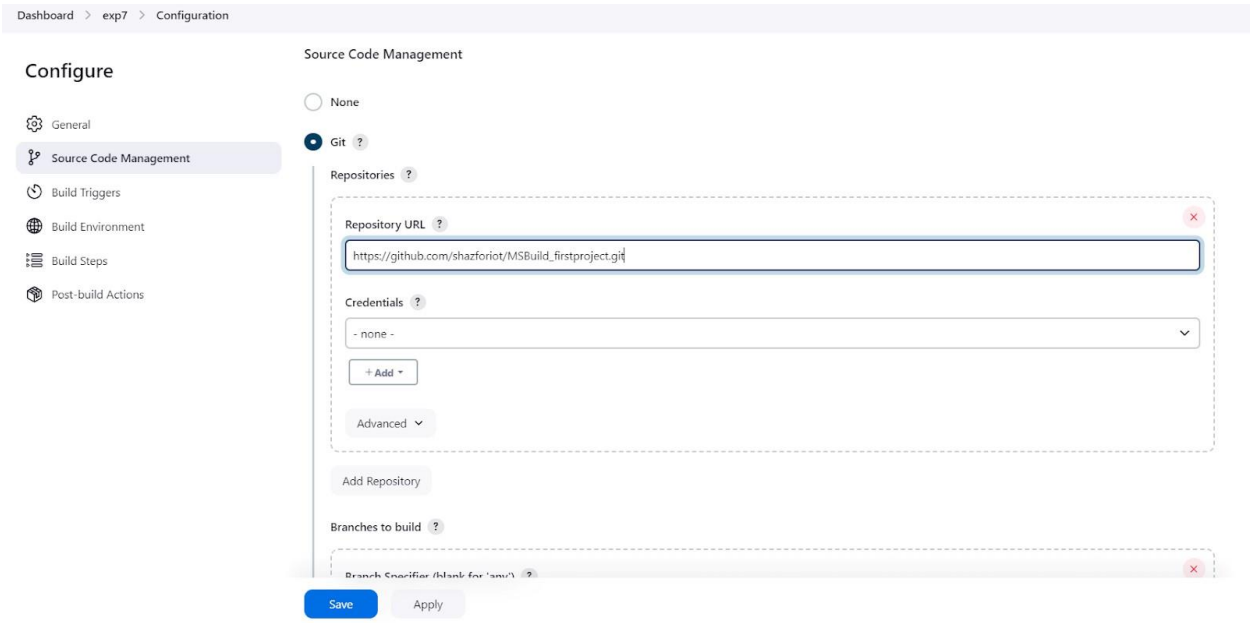


9. Choose this GitHub repository in Source Code Management.

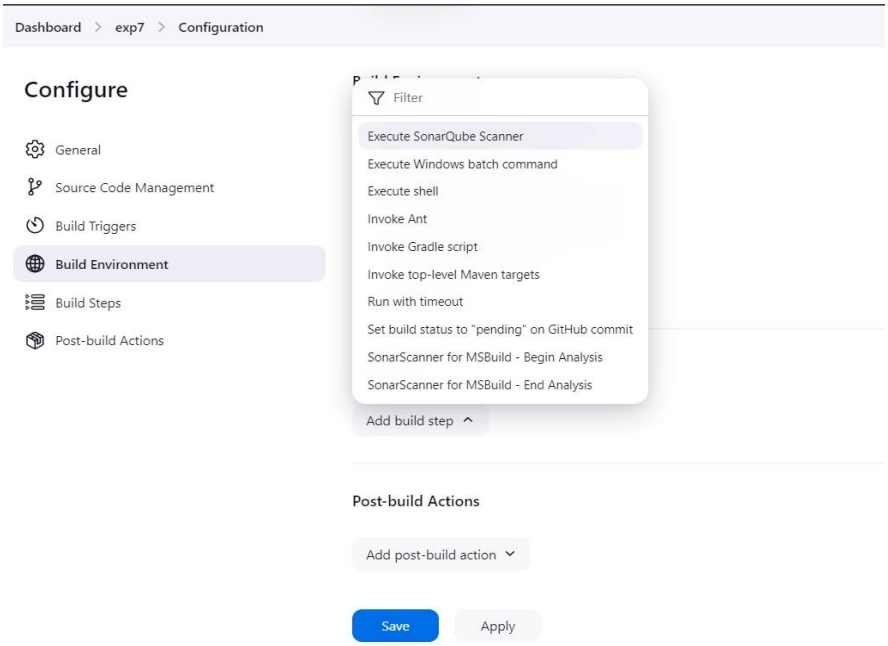
https://github.com/shazforiot/MSBuild_firstproject.git

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

Name:Sahil Motiramani Div:D15C Roll No.35

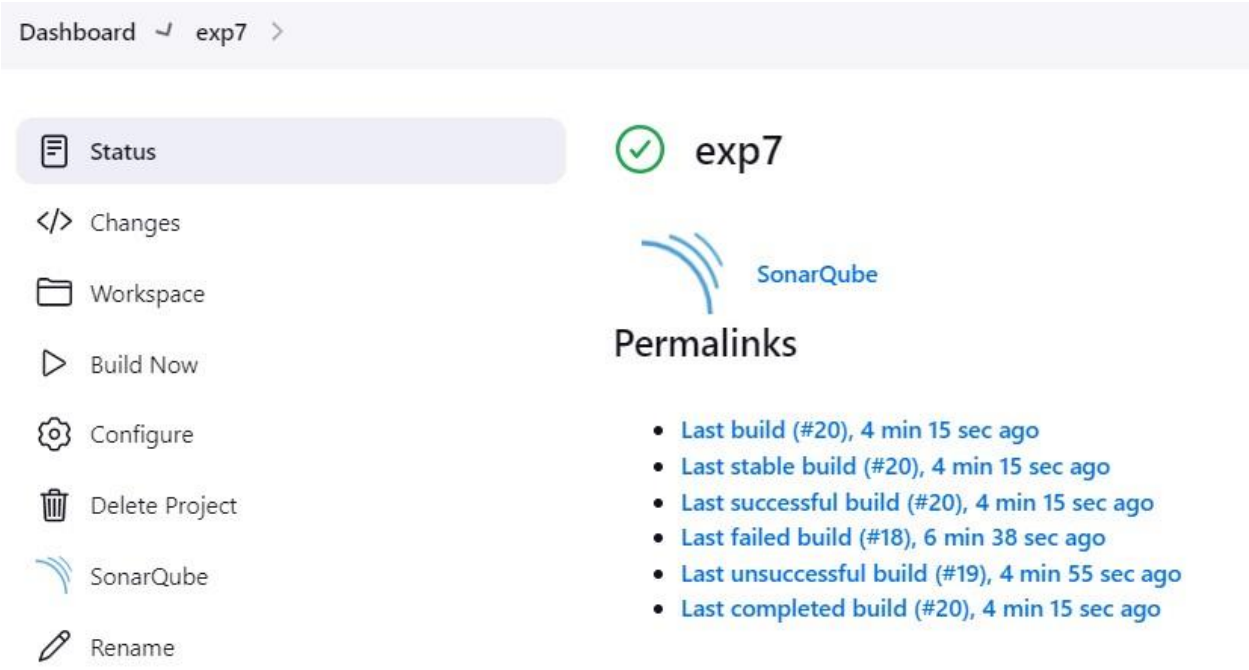


10. Under **Select project → Configuration → Build steps → Execute SonarQube Scanner** , enter these Analysis properties. Mention the SonarQube Project Key, Login, Password, Source path and Host URL.

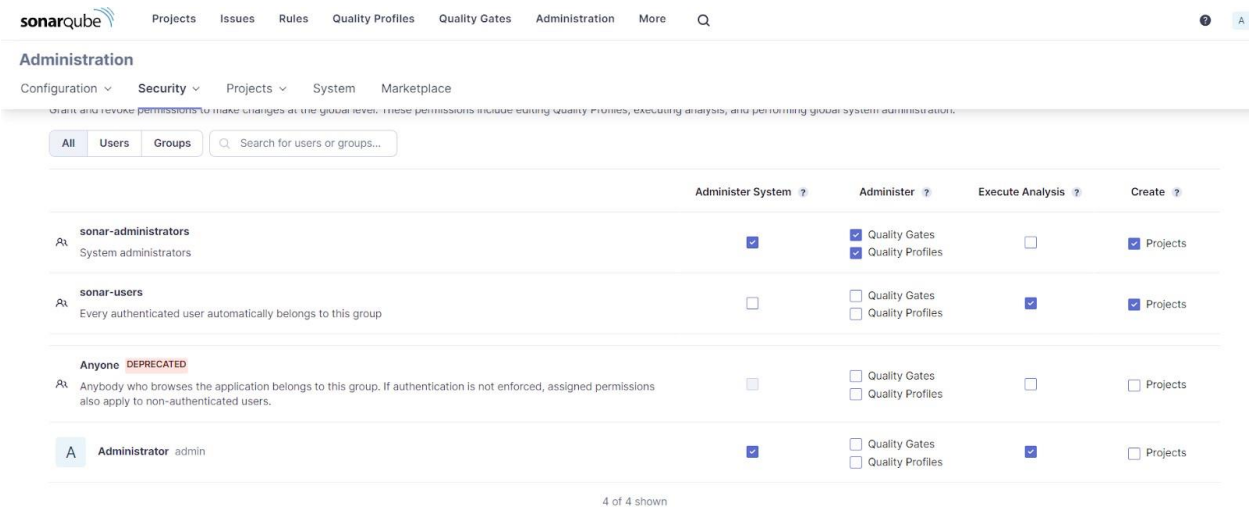


Then save

Name:Sahil Motiramani Div:D15C Roll No.35



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



IF CONSOLE OUTPUT FAILED:

Name:Sahil Motiramani Div:D15C Roll No.35

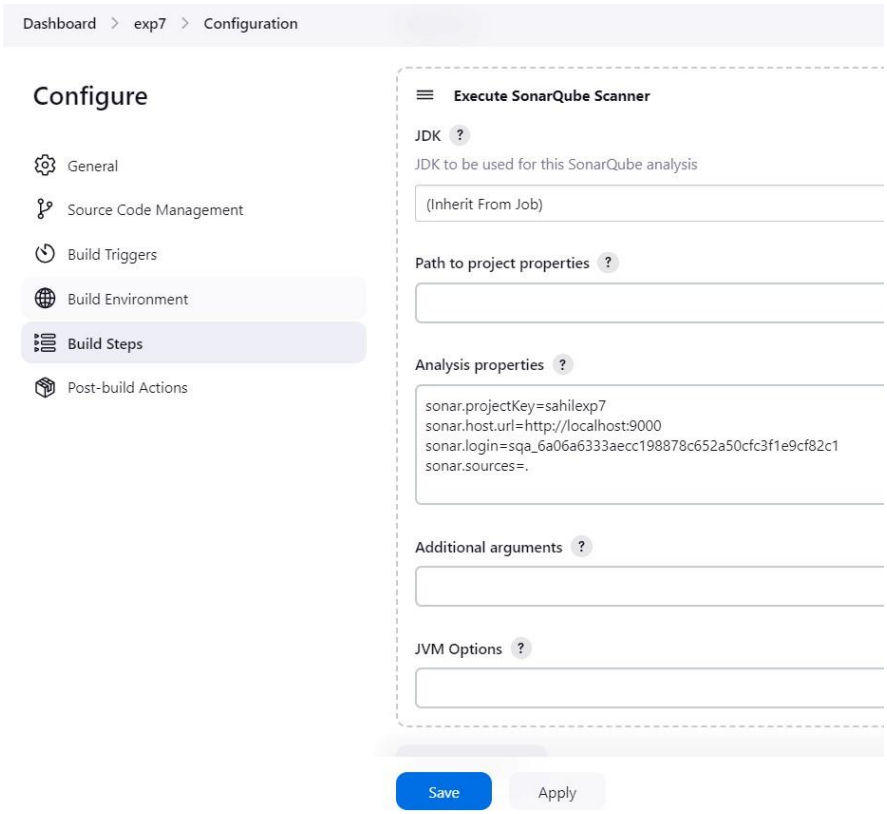
Step 1: Generate a New Authentication Token in SonarQube

1. **Login to SonarQube:**
    - Open your browser and go to `http://localhost:9000` .
    - Log in with your admin credentials (default username is `admin` , and the password is either `admin` or your custom password if it was changed).
2. **Generate a New Token:**
    - Click on your **username** in the top-right corner of the SonarQube dashboard.
    - Select **My Account** from the dropdown menu.
    - Go to the **Security** tab.
    - Under **Generate Tokens** , type a name for the token (e.g., "Jenkins-SonarQube").
    - Click **Generate** .
    - Copy the token and save it securely. You will need it in Jenkins.

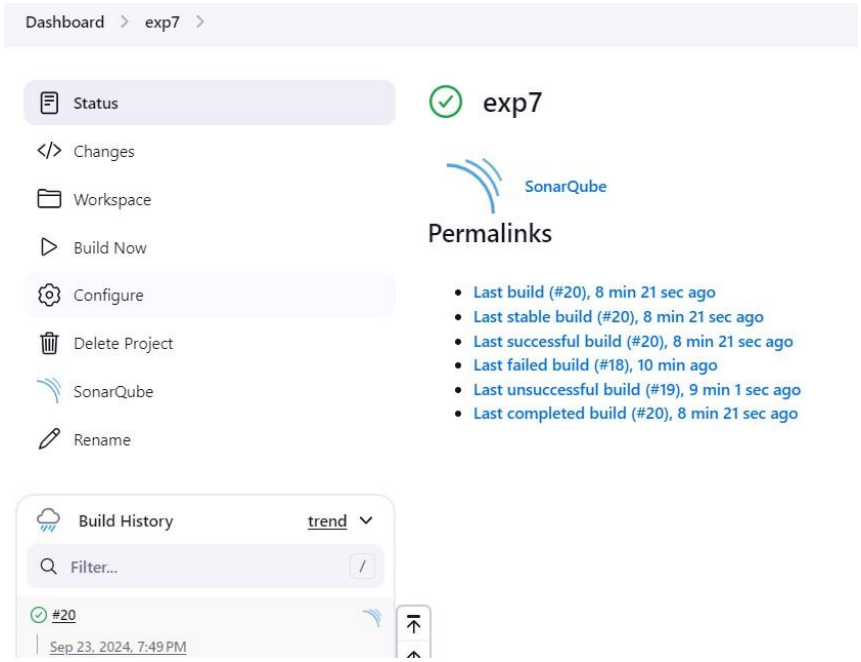
Step 2: Update the Token in Jenkins

1. **Go to Jenkins Dashboard:**
    - Open Jenkins and log in with your credentials.
2. **Configure the Jenkins Job:**
    - Go to the job that is running the SonarQube scanner (`adv_devops_exp7`).
    - Click **Configure** .
3. **Update the SonarQube Token:**
    - In the SonarQube analysis configuration (either in the pipeline script or under "Build" section, depending on your job type), update the `sonar.login` parameter with the new token.

Name:Sahil Motiramani Div:D15C Roll No.35



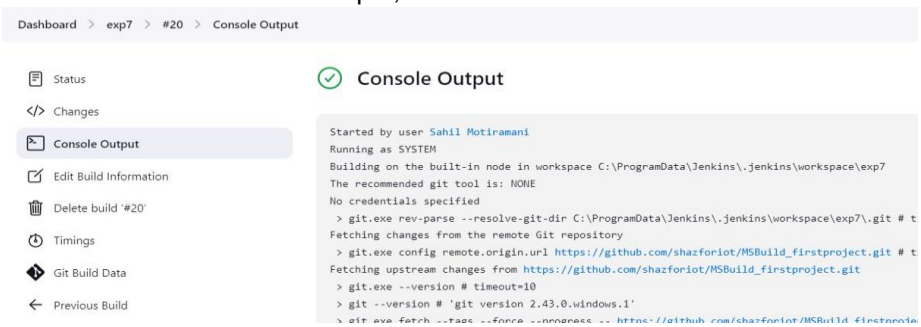
12. Run the Jenkins build.



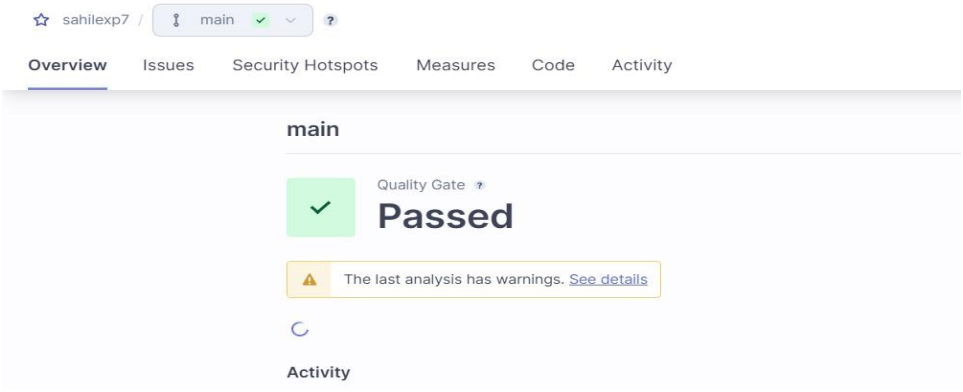
Name:Sahil Motiramani Div:D15C Roll No.35

Check the console Output

Console output;



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



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

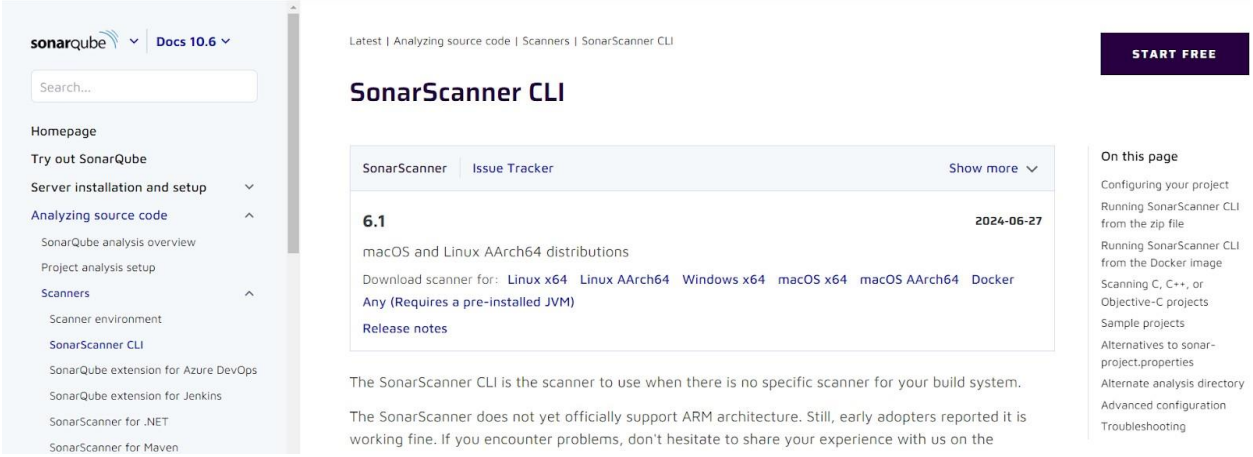
Conclusion:

In this project, we integrated Jenkins with SonarQube for automated static application security testing (SAST). We set up SonarQube using Docker, configured Jenkins with the necessary plugins and authentication, and linked it to a GitHub repository. The SonarQube scanner was added as a build step, enabling continuous code analysis for vulnerabilities, code smells, and quality issues, ensuring automated reporting and continuous code quality improvement.

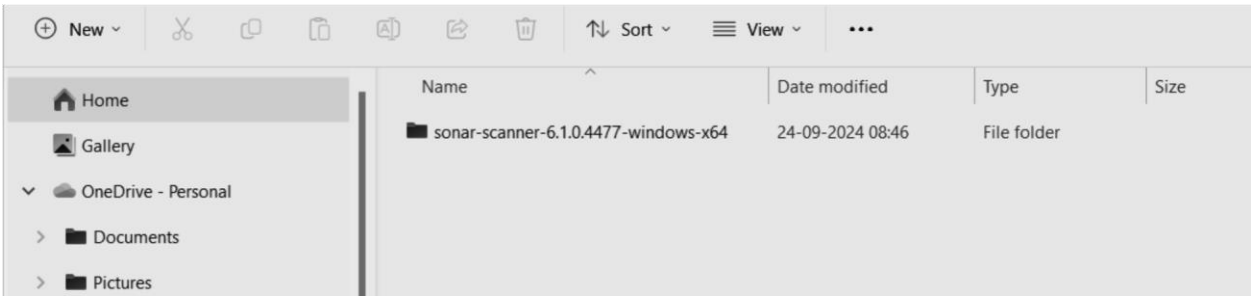
Name:Sahil Motiramani Roll No:35 Div:D15C

Exp:8

Aim: Create a Jenkins CICD Pipeline with SonarQube / GitLab Integration to perform a static analysis of the code to detect bugs, code smells, and security vulnerabilities on a sample Web / Java / Python application. Step 1: Download sonar scanner
https://docs.sonarsource.com/sonarqube/latest/analyzing-source-code/scanners/sonarscanner/



Visit this link and download the sonarqube scanner CLI.
Extract the downloaded zip file in a folder.



1. Install sonarqube image

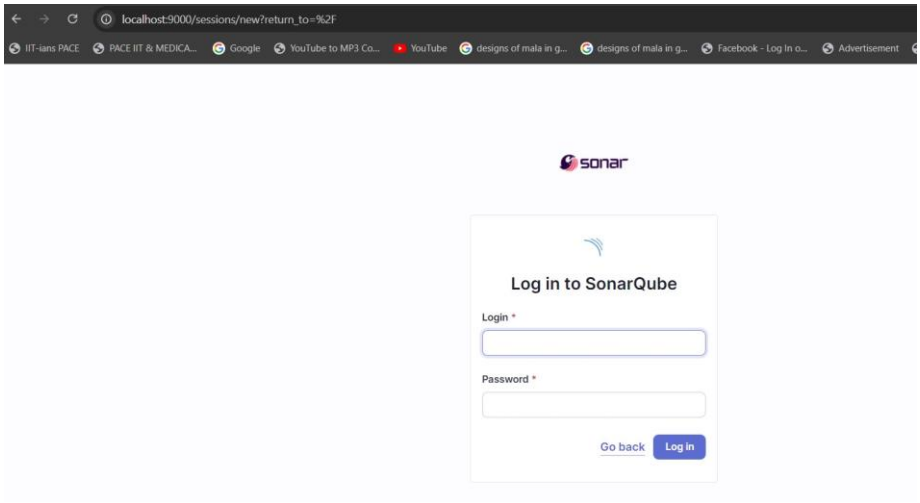
Command: docker pull sonarqube

```
C:\Users\HP\Desktop\sem5\advdevops8>docker pull sonarqube
Using default tag: latest
latest: Pulling from library/sonarqube
Digest: sha256:72e9feec71242af83faf65f95a40d5e3bb2822a6c3b2cda8568790f3d31aecde
Status: Image is up to date for sonarqube:latest
docker.io/library/sonarqube:latest

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

C:\Users\HP\Desktop\sem5\advdevops8>
```

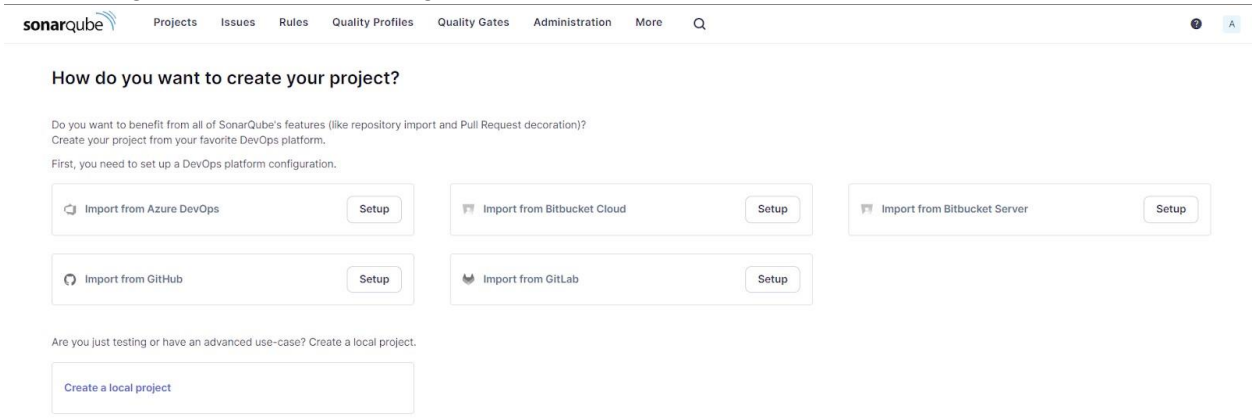
```
C:\Users\HP\Desktop\sem5\advdevops8>docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:900
0 sonarqube:latest
a57154161e14bed00ec141b755fa197a52321bf5c0688b825ff4dfbeaf712099
```

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

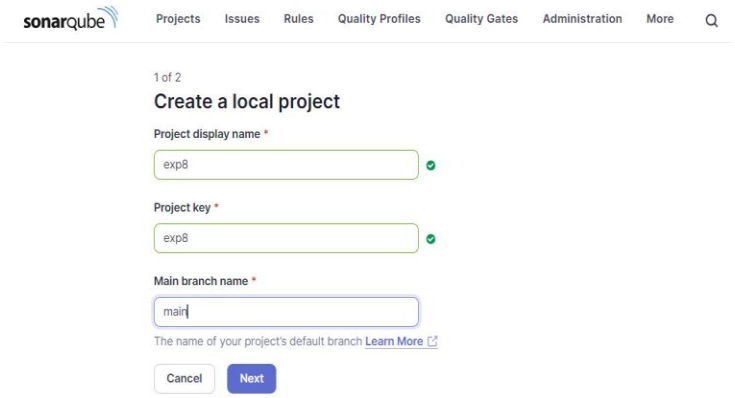
Name:Sahil Motiramani Roll No:35 Div:D15C



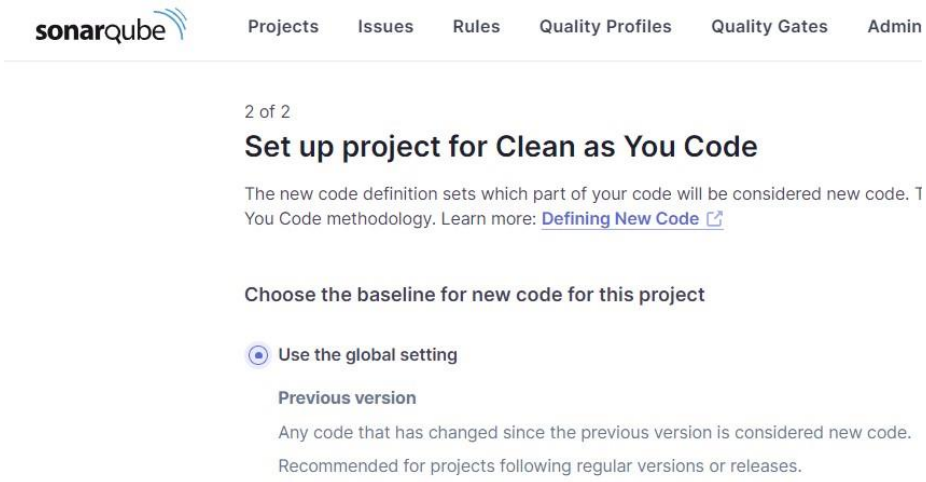
3. Login to SonarQube using username admin and password admin.



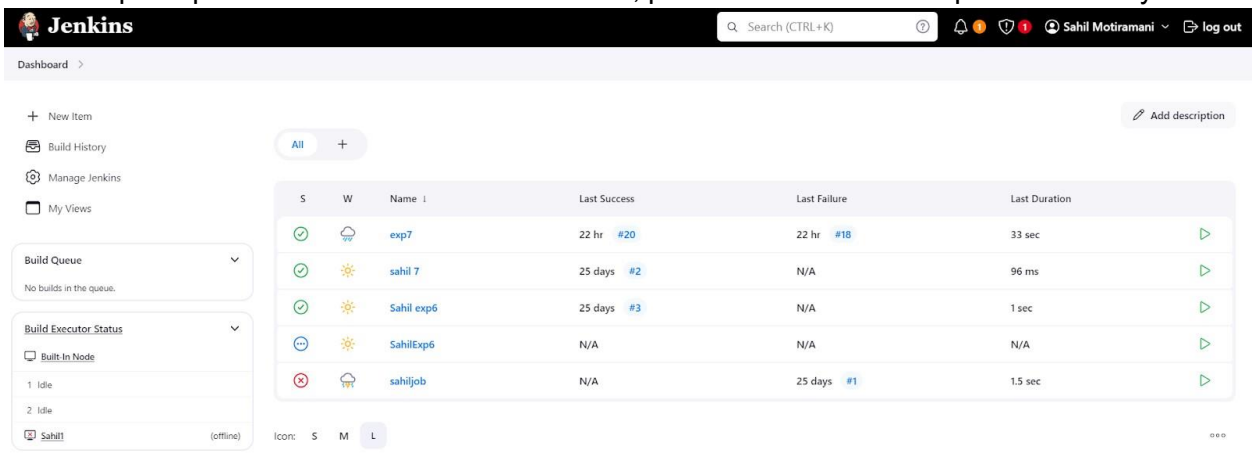
4. Create a manual project in SonarQube with the name sonarqube



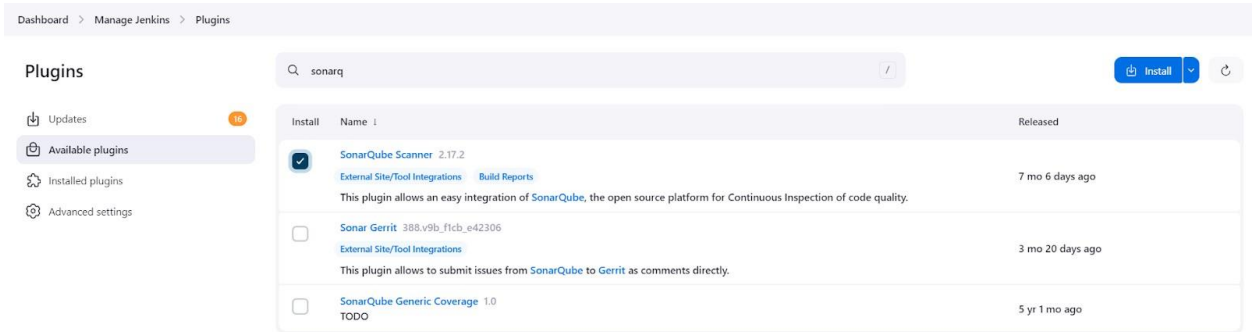
Name:Sahil Motiramani Roll No:35 Div:D15C



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



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



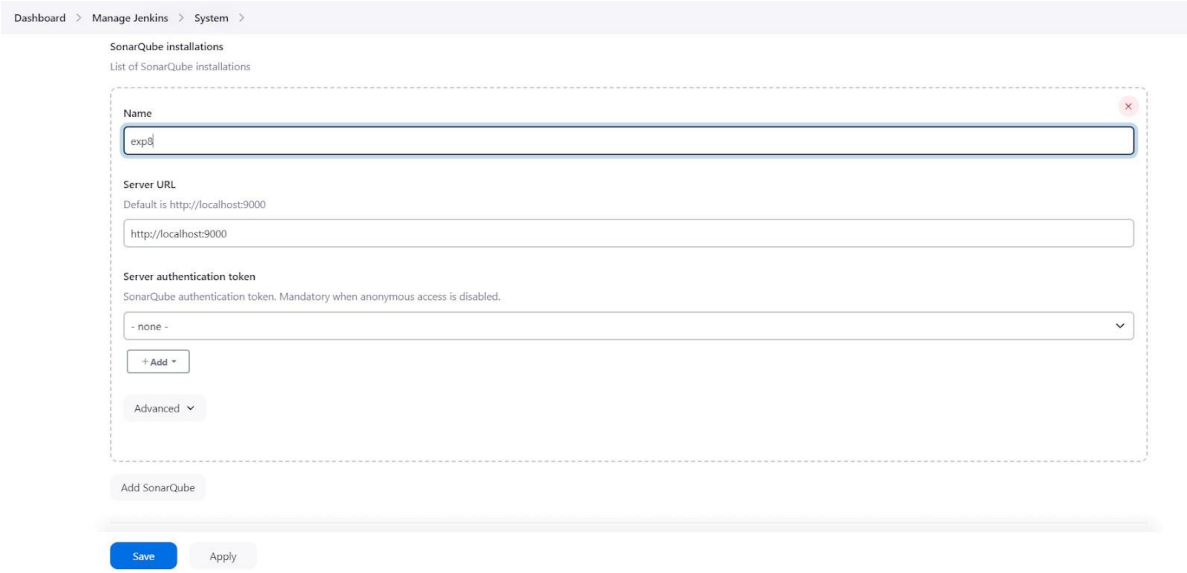
Name:Sahil Motiramani Roll No:35 Div:D15C

7. Under Jenkins ‘Manage Jenkins’ then go to ‘system’, scroll and look for SonarQube Servers and enter the details.

Enter the Server Authentication token if needed.

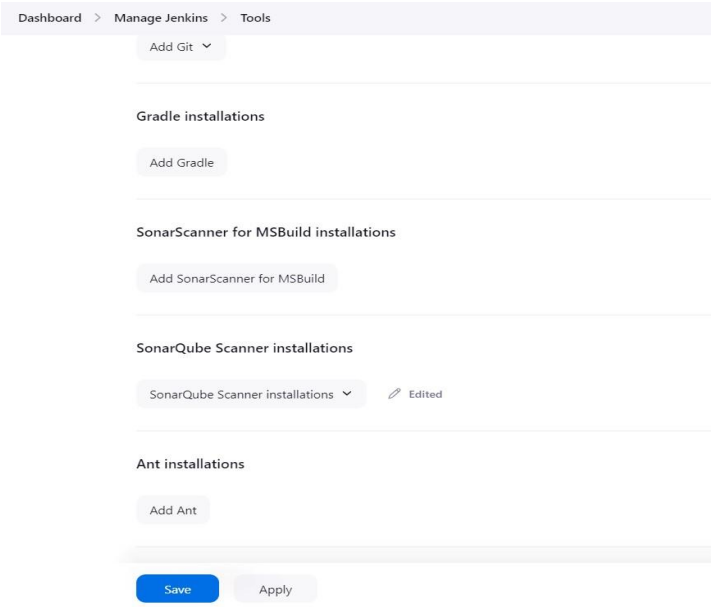
In SonarQube installations: Under Name add <project name of sonarqube> for me adv_devops_7_sonarqube

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

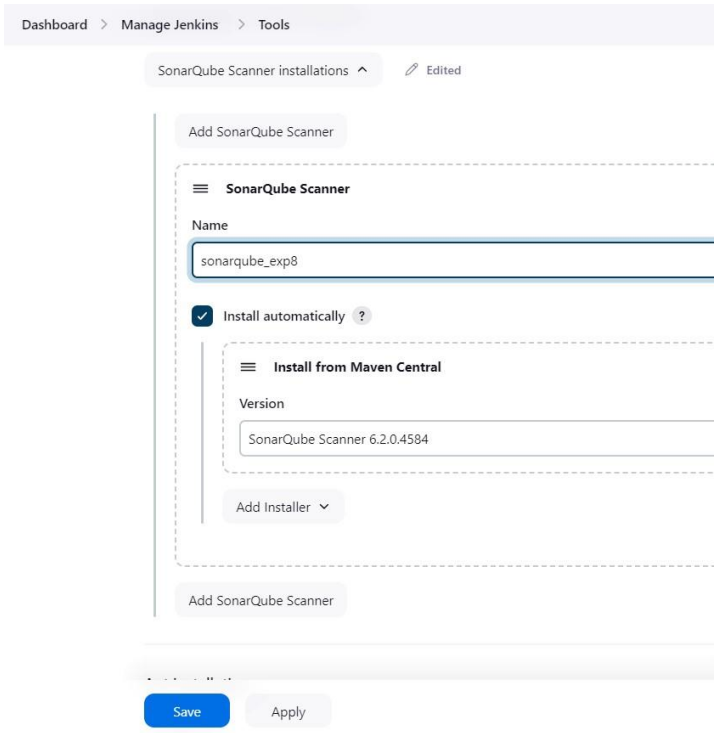


8. Search for SonarQube Scanner under Global Tool Configuration. Choose the latest configuration and choose Install automatically.

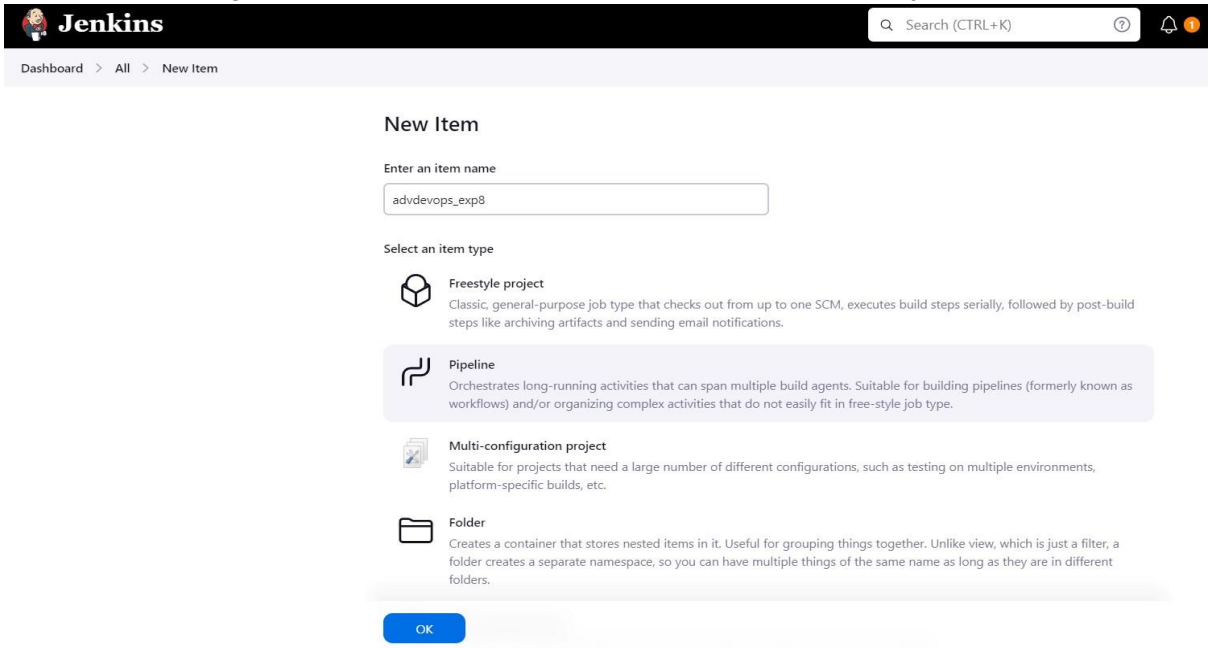
Dashboard > Manage Jenkins > Tools



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



9. After configuration, create a New Item → choose a pipeline project.

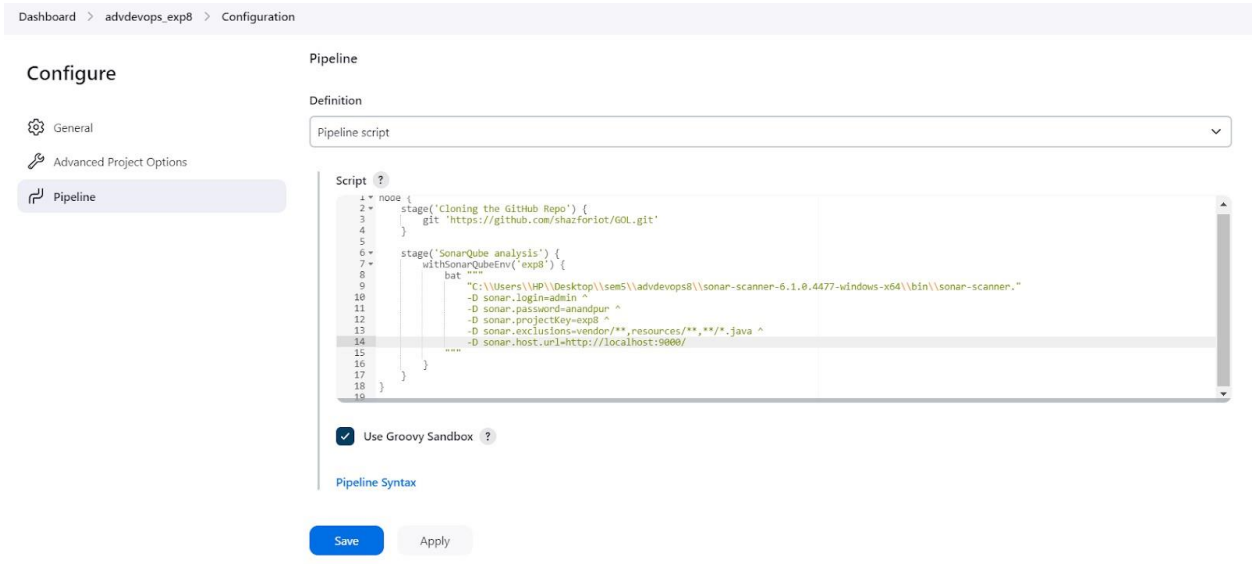


Name:Sahil Motiramani Roll No:35 Div:D15C

10. Under Pipeline script, enter the following:

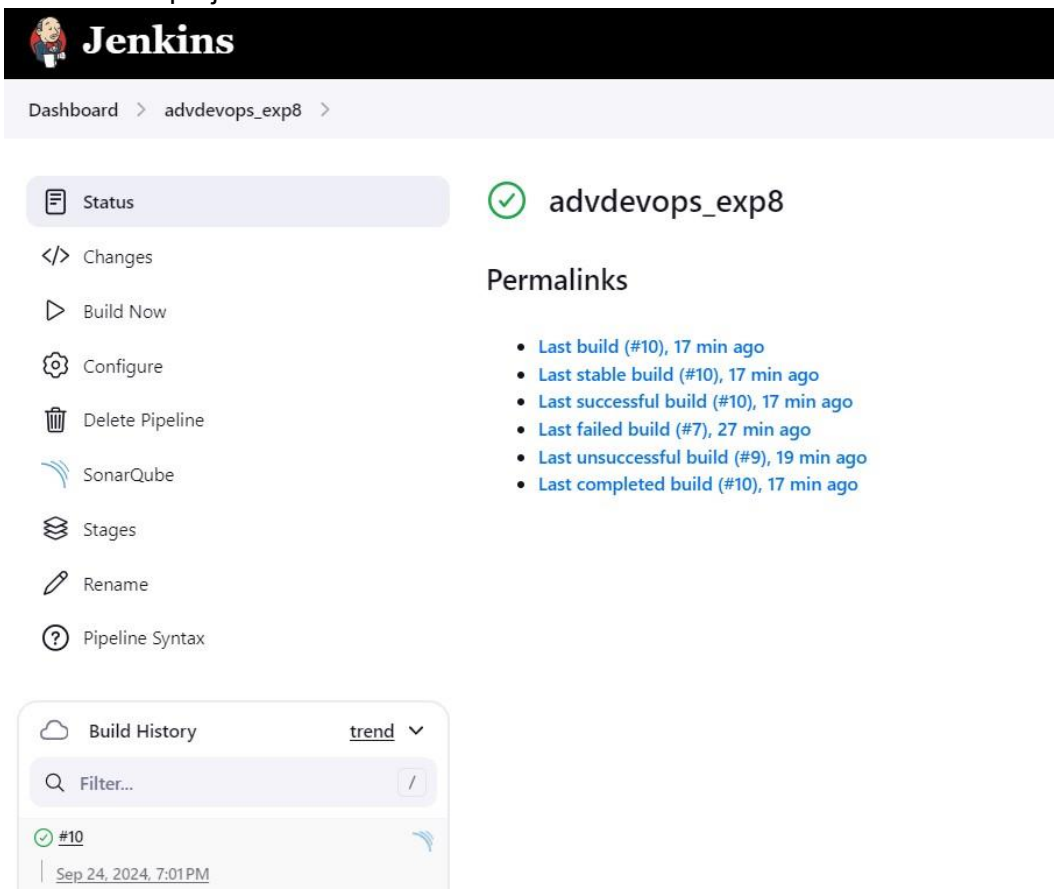
```
        node {
    stage('Cloning the GitHub Repo') {  git
      'https://github.com/shazforiot/GOL.git'
    }

    stage('SonarQube analysis') {
       withSonarQubeEnv('<Name_of_SonarQube_environment_on_Jenkins>') {
         sh """
             <PATH_TO_SONARQUBE_SCANNER_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=<SonarQube_URL>(default: http://localhost:9000/)
          """
       }
    }
}
```

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

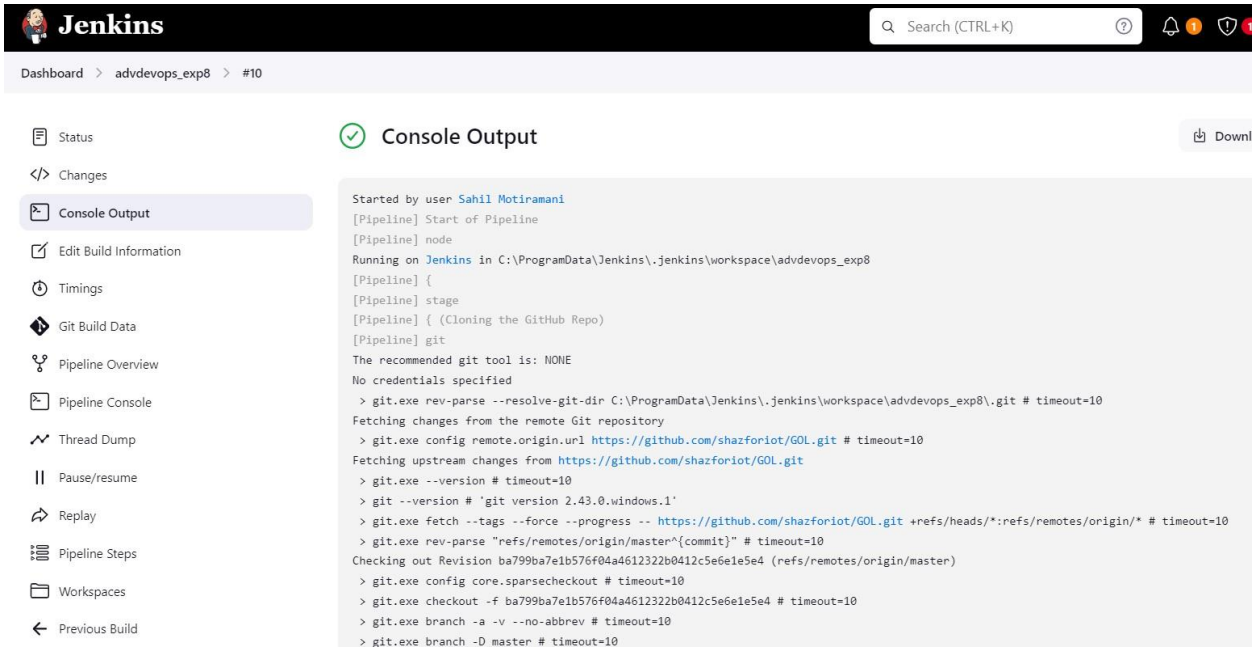


Name:Sahil Motiramani Roll No:35 Div:D15C

11. Build project

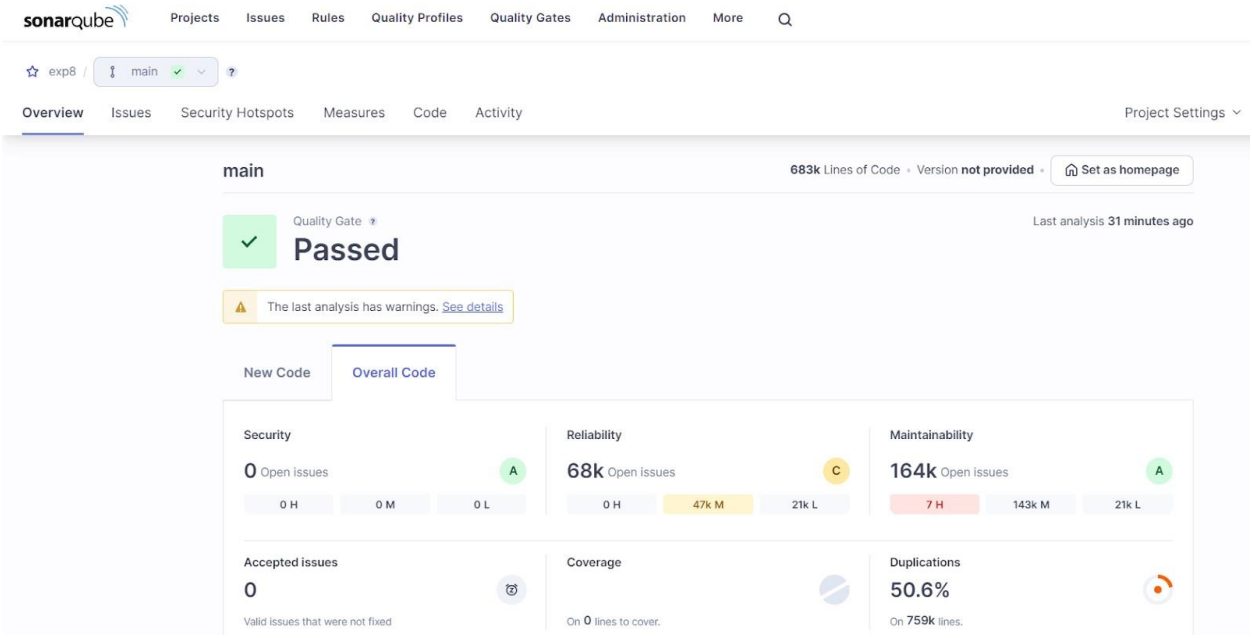


12. Check console



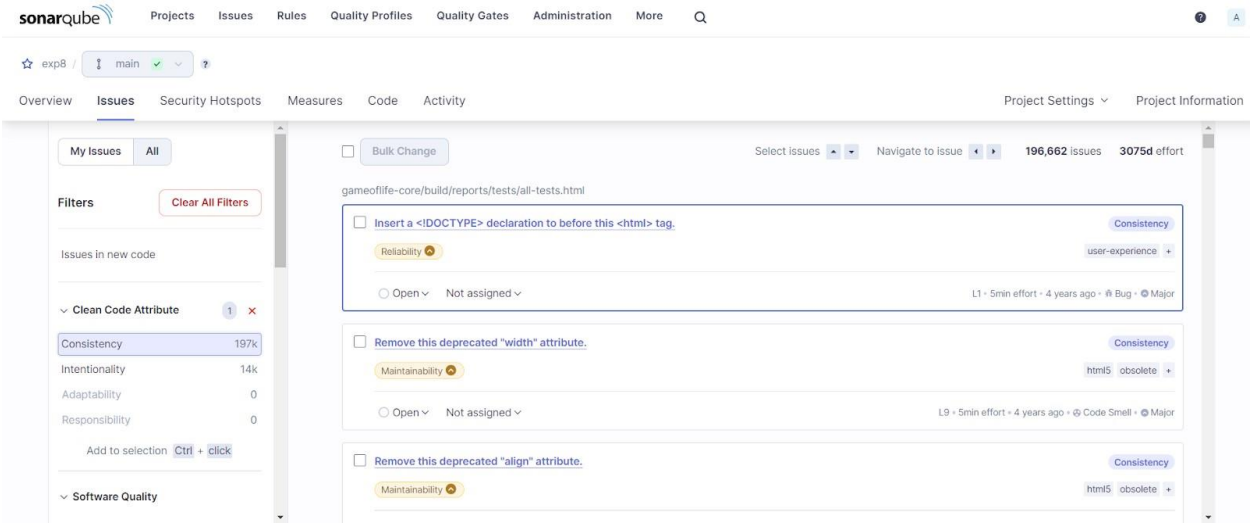
Name:Sahil Motiramani Roll No:35 Div:D15C

13. Now, check the project in SonarQube



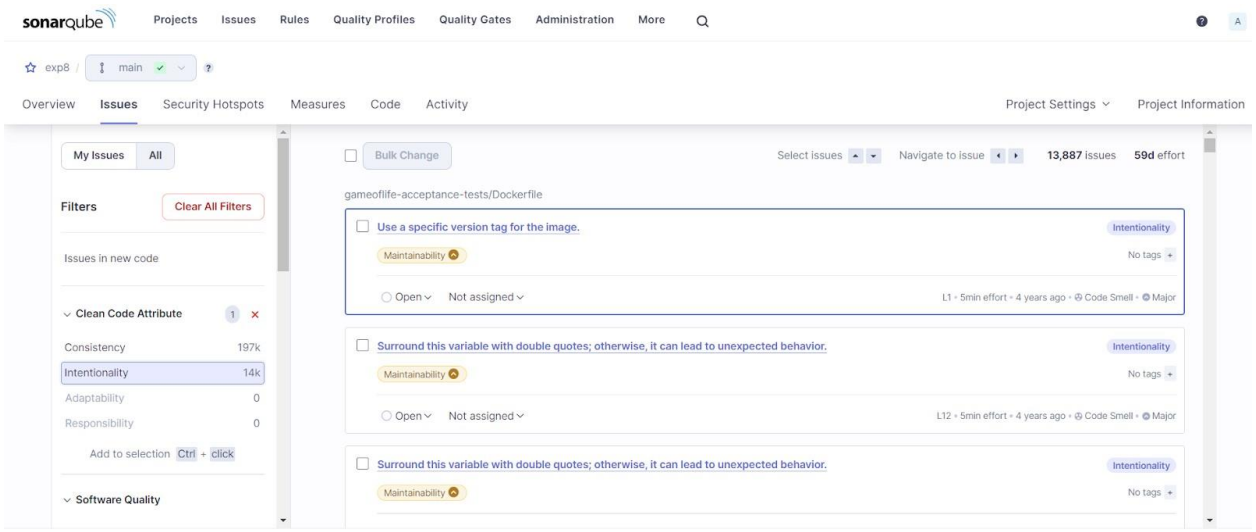
14. Code Problems

- Consistency

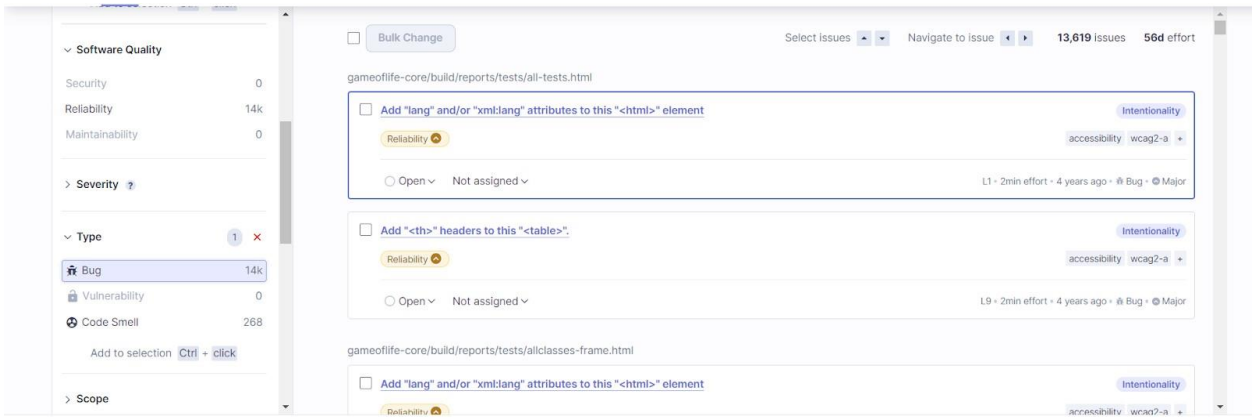


- Intentionality

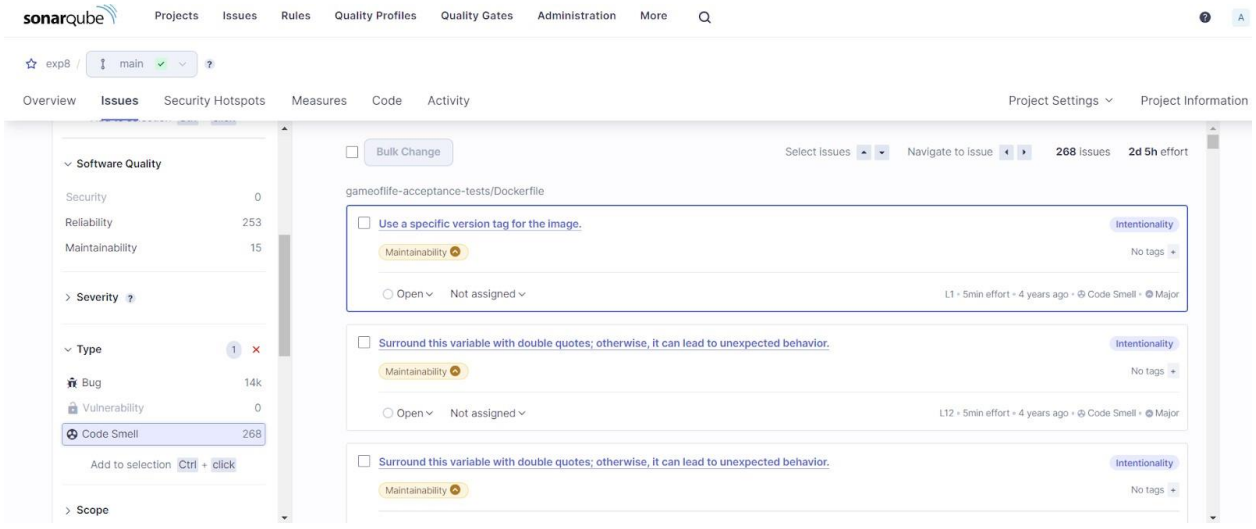
Name:Sahil Motiramani Roll No:35 Div:D15C



- Bugs

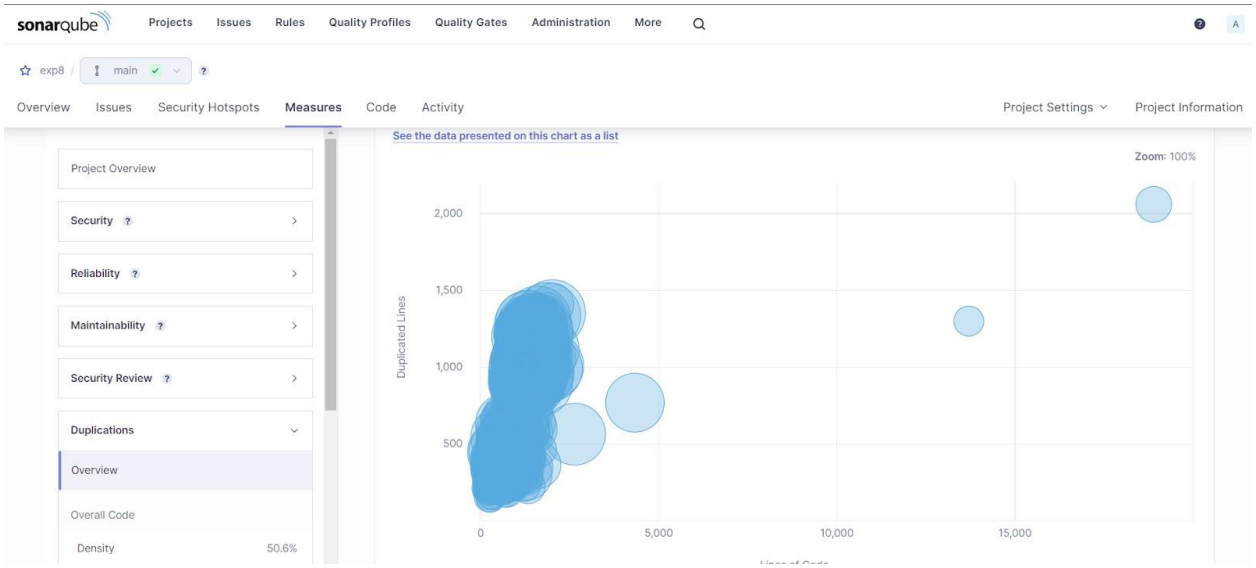


- Code Smells

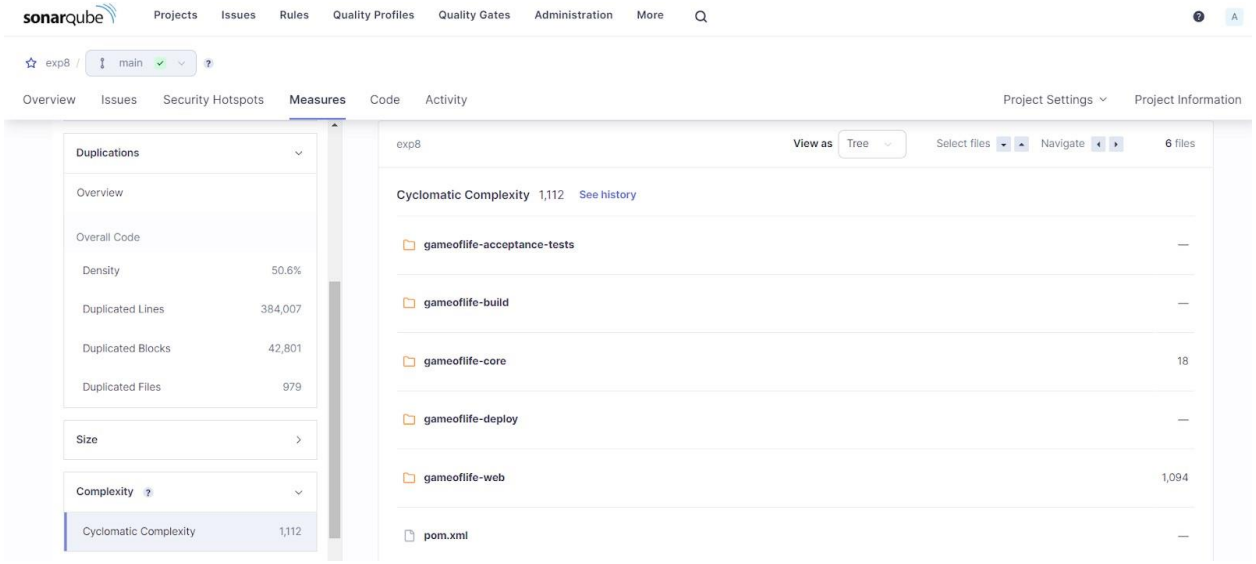


- Duplications

Name:Sahil Motiramani Roll No:35 Div:D15C



- Cyclomatic Complexities



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

Conclusion:

In this experiment, we integrated Jenkins with SonarQube to enable automated code quality checks within our CI/CD pipeline. We started by deploying SonarQube using Docker, setting up a project, and configuring it to analyze code quality. Next, we configured Jenkins by installing the SonarQube Scanner plugin, adding SonarQube server details, and setting up the scanner tool. We then developed a Jenkins pipeline to automate the process of cloning a GitHub repository and running SonarQube analysis on the code. This integration helps ensure continuous monitoring of code quality, detecting issues such as bugs, code smells, and security vulnerabilities throughout the development process.

Name:Sahil Motiramani Div:D15C Rollno:35

Exp:09

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

Theory :

What is Nagios?

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

Nagios is used for continuous monitoring of systems, applications, service and business processes in a DevOps culture.

Why We Need Nagios tool?

Here are the important reasons to use Nagios monitoring tool:

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

Features of Nagios

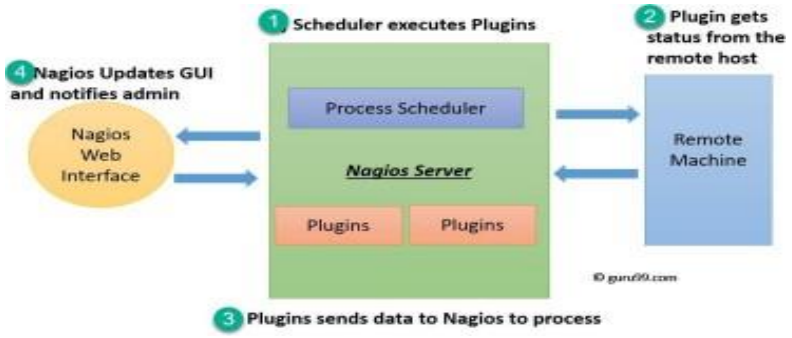
Following are the important features of Nagios monitoring tool:

- Relatively scalable, Manageable, and Secure
- Good log and database system
- Informative and attractive web interfaces
- Automatically send alerts if condition changes

- If the services are running fine, then there is no need to do check that host is an alive ● Helps you to detect network errors or server crashes ● You can troubleshoot the performance issues of the server.
- The issues, if any, can be fixed automatically as they are identified during the monitoring process
- You can monitor the entire business process and IT infrastructure with a single pass ● The product's architecture is easy to write new plugins in the language of your choice ● Nagios allows you to read its configuration from an entire directory which helps you to decide how to define individual files
- Utilizes topology to determine dependencies
- Monitor network services like HTTP, SMTP, HTTP, SNMP, FTP, SSH, POP, etc. ● Helps you to define network host hierarchy using parent hosts
- Ability to define event handlers that runs during service or host events for proactive problem resolution
- Support for implementing redundant monitoring hosts

Nagios Architecture

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



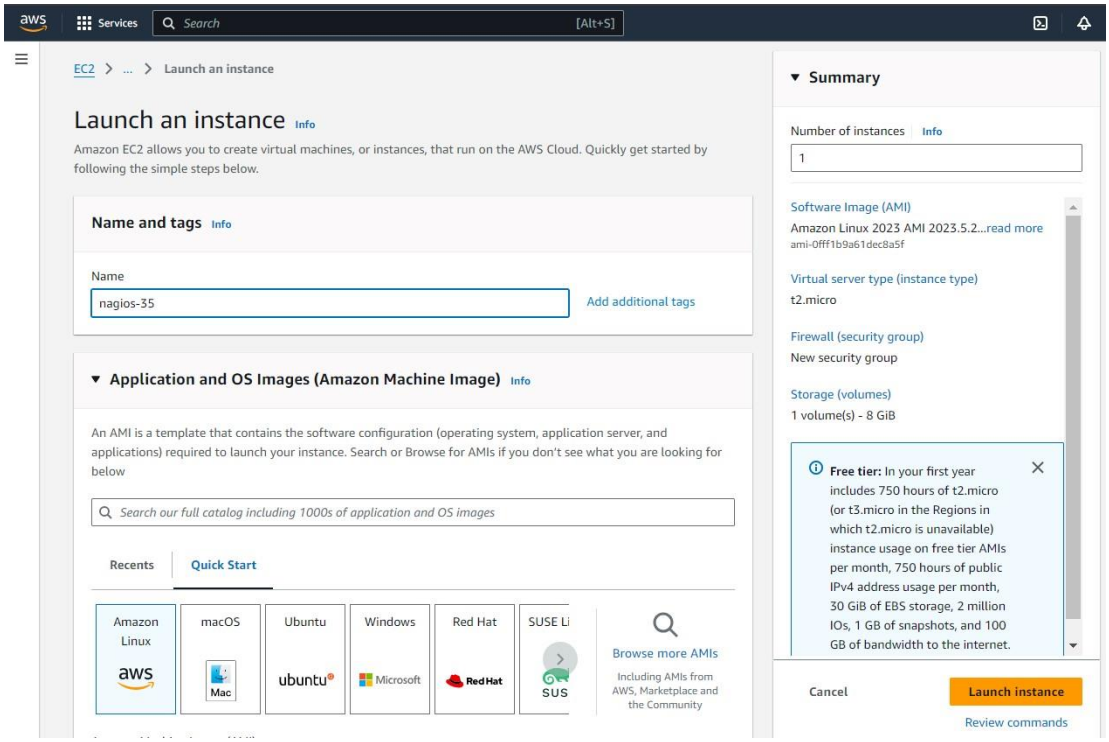
1. The scheduler is a component of the server part of Nagios. It sends a signal to execute the plugins at the remote host.
2. The plugin gets the status from the remote host
3. The plugin sends the data to the process scheduler
4. The process scheduler updates the GUI and notifications are sent to admins.

Installation of Nagios

Prerequisites: AWS Free Tier

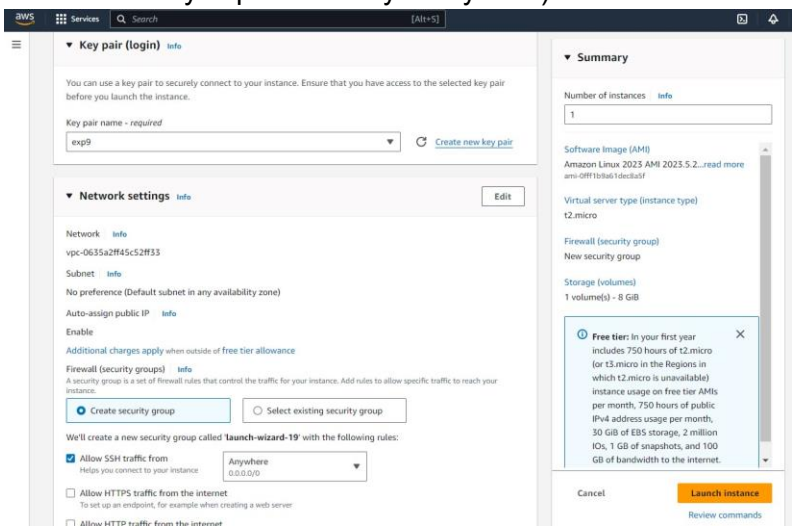
Name:Sahil Motiramani Div:D15C Rollno:35

Step 1: Login to your AWS account. Search for EC2 on services. Open the interface and click on Create Instance.



Select The OS Image as Amazon Linux.

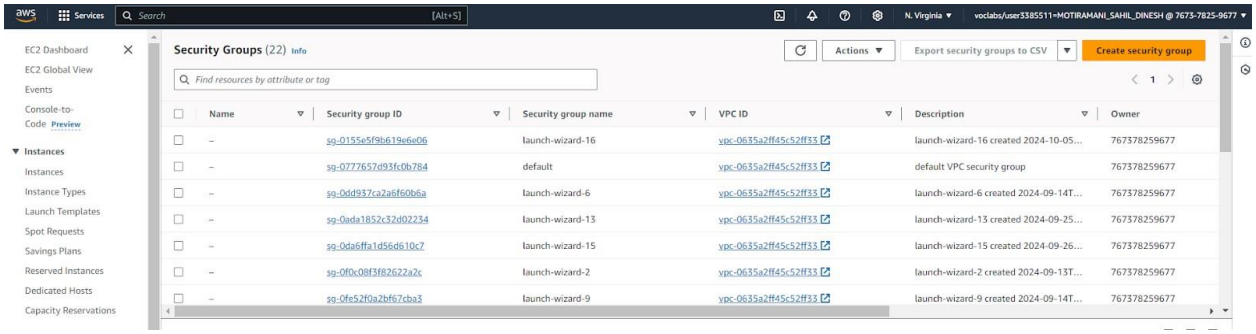
Step 2: If you do not have a private key created or a .pem file created, click on create a key pair. Else select the key pair that you had created before. (Make sure you know where the .pem file for that key is present on your system)



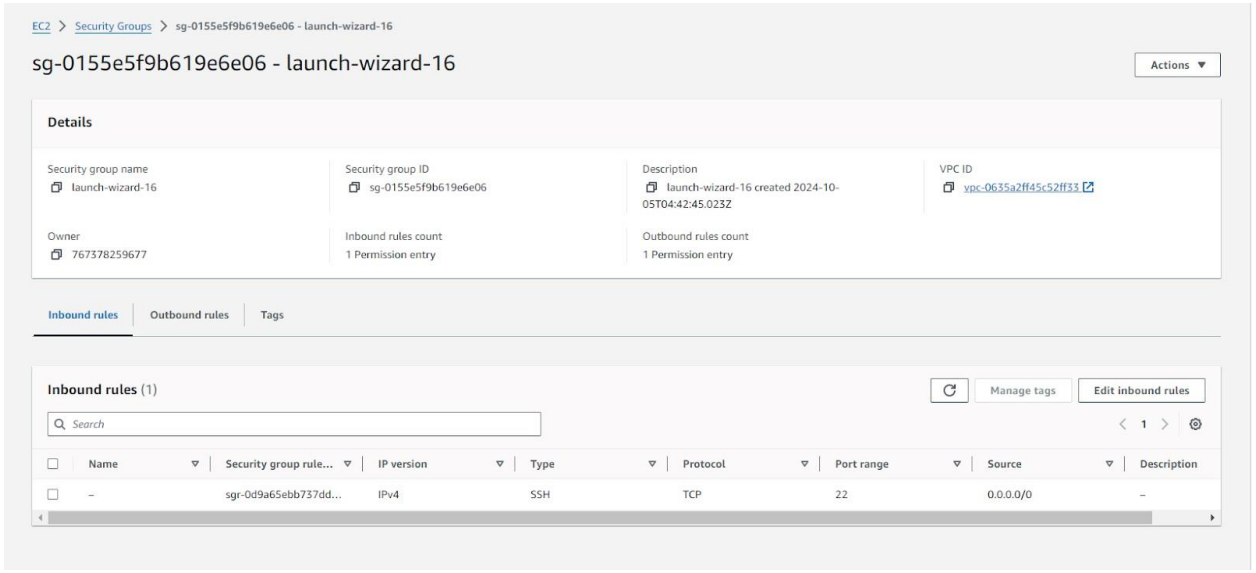
AWS will create a security group for this instance. Keep the name of that instance saved.

Name:Sahil Motiramani Div:D15C Rollno:35

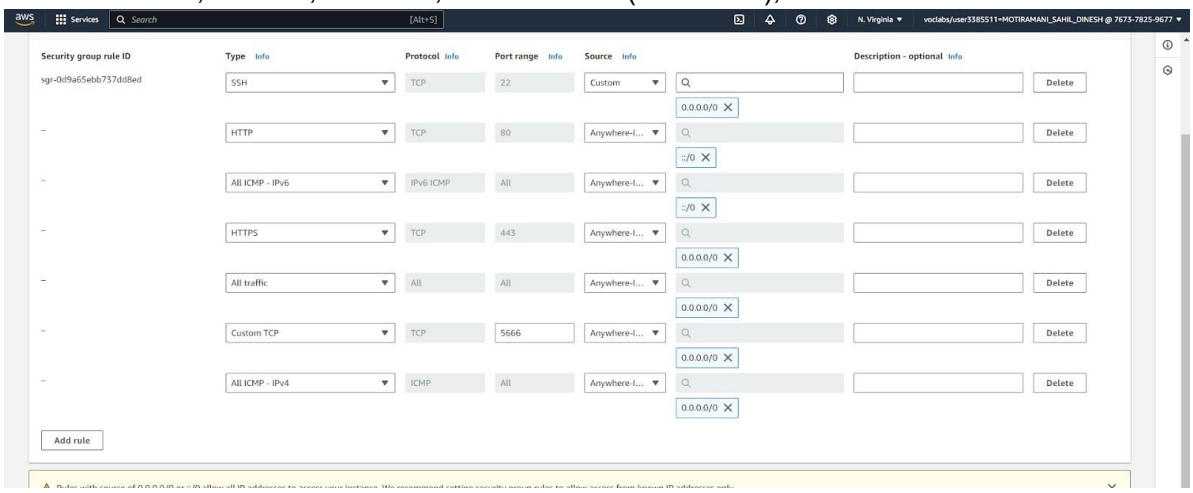
Step 3: After creating the instance, click on Security Groups from the left side pane. Find the security group that was created for your instance. Click on the instance ID for that group.



Here, click on Edit Inbound Rules.

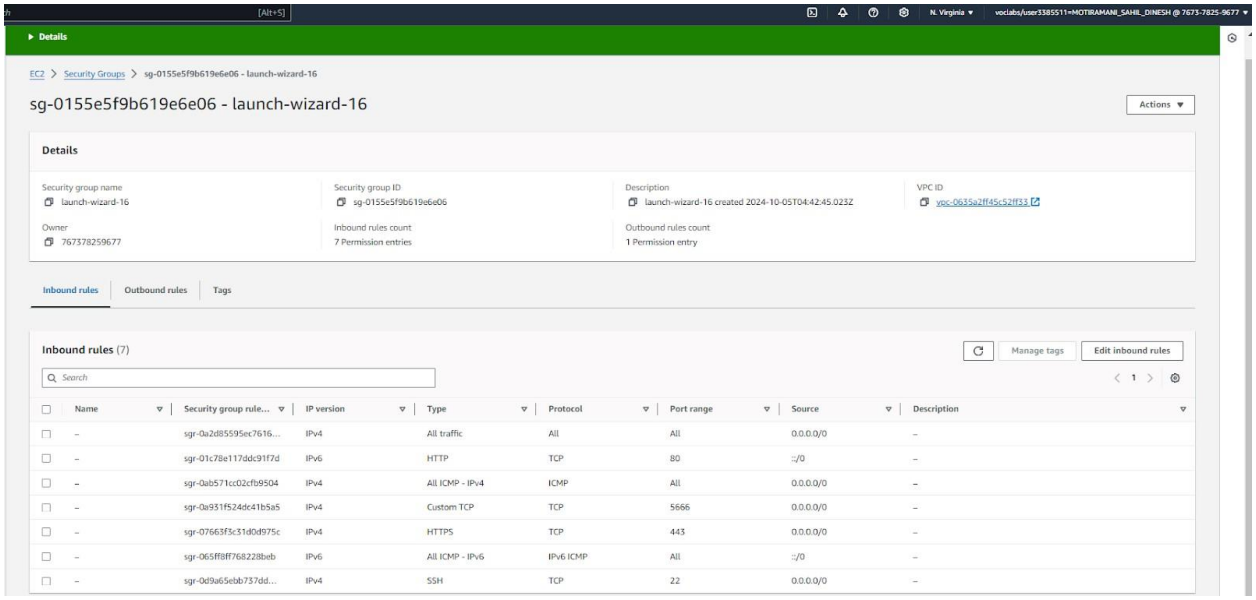


Now, click on add rules, and add teh rules for the following protocols:
HTTP, All ICMP - IPv6, HTTPS, All traffic, Custom TCP (Port 5666), All ICMP - IPv4

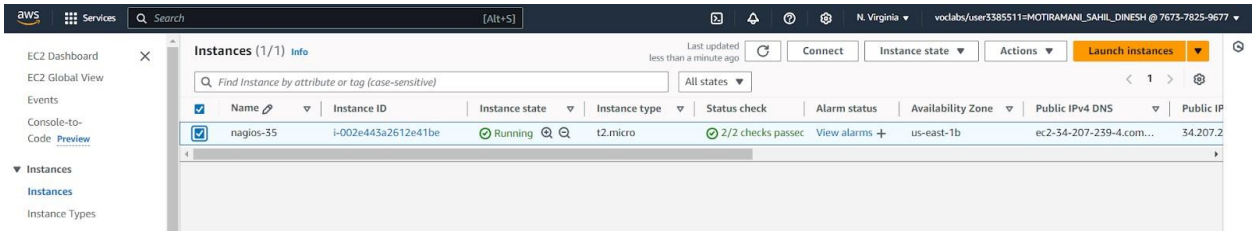


Click on save. This will add all the inbound rules to the security group.

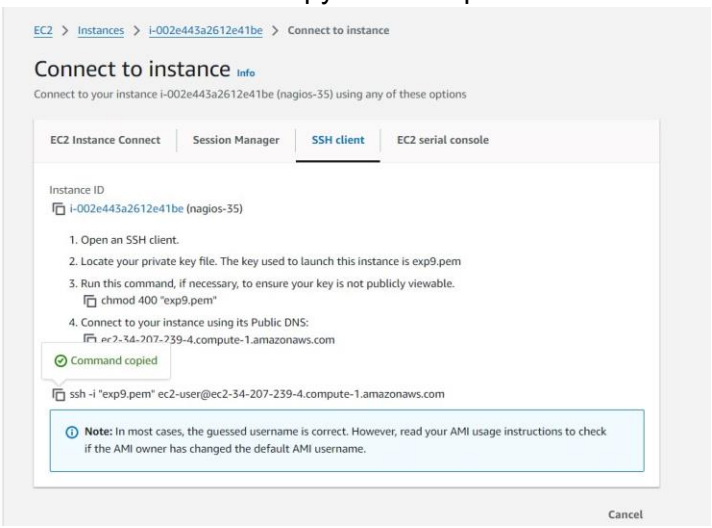
Name:Sahil Motiramani Div:D15C Rollno:35



Step 4: Now come back to the instances screen. Click on the instance ID of your instance. Then click on Connect.



Click on SSH client. Copy the example command.



Name:Sahil Motiramani Div:D15C Rollno:35

Step 5: Now, we have to connect our local OS terminal to the instance using SSH. For this, Open terminal wher the private key file is located (.pem) Paste the copied SSH command and run it.

```
Microsoft Windows [Version 10.0.22631.4249]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP\Desktop\sem5\advdevops 9>ssh -i "exp9.pem" ec2-user@ec2-34-207-239-4.compute-1.amazonaws.com
The authenticity of host 'ec2-34-207-239-4.compute-1.amazonaws.com (34.207.239.4)' can't be established.
ED25519 key fingerprint is SHA256:UuegxDkRL6R+3iPxqY2jG36UhsDv85Y3Wp21RVcZobo.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-34-207-239-4.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
       #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-31-39-94 ~]$
```

Step 6: Now we start working on this terminal. First run the command
sudo yum update
This command will check for any updates for the YUM library.

```
[ec2-user@ip-172-31-39-94 ~]$ sudo yum update
Last metadata expiration check: 0:19:35 ago on Sun Oct  6 10:11:16 2024.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-39-94 ~]$
```

Step 7: We are going to install an Apache server and a PHP on this instance. For that, run this command.
sudo yum install httpd php

```
[ec2-user@ip-172-31-39-94 ~]$ sudo yum install httpd php
Last metadata expiration check: 0:20:07 ago on Sun Oct  6 10:11:16 2024.
Dependencies resolved.
================================================================================
 Package                 Architecture    Version                 Repository    Size
================================================================================
Installing:
 httpd                   x86_64          2.4.62-1.amzn2023       amazonlinux   48 k
 php8.3                  x86_64          8.3.10-1.amzn2023.0.1   amazonlinux   10 k
Installing dependencies:
 apr                     x86_64          1.7.2-2.amzn2023.0.2    amazonlinux  129 k
 apr-util                x86_64          1.6.3-1.amzn2023.0.1    amazonlinux   98 k
 generic-logos-httpd     noarch          18.0.0-12.amzn2023.0.3  amazonlinux   19 k
 httpd-core              x86_64          2.4.62-1.amzn2023       amazonlinux  1.4 M
 httpd-filesystem        noarch          2.4.62-1.amzn2023       amazonlinux   14 k
 httpd-tools             x86_64          2.4.62-1.amzn2023       amazonlinux   81 k
 libbrotli               x86_64          1.0.9-4.amzn2023.0.2    amazonlinux  315 k
 libsodium               x86_64          1.0.19-4.amzn2023       amazonlinux  176 k
 libxslt                 x86_64          1.1.34-5.amzn2023.0.2   amazonlinux  241 k
 mailcap                 noarch          2.1.49-3.amzn2023.0.3   amazonlinux   33 k
 nginx-filesystem        noarch          1:1.24.0-1.amzn2023.0.4 amazonlinux  9.8 k
 php8.3-cli              x86_64          8.3.10-1.amzn2023.0.1   amazonlinux  3.7 M
 php8.3-common           x86_64          8.3.10-1.amzn2023.0.1   amazonlinux  737 k
 php8.3-process          x86_64          8.3.10-1.amzn2023.0.1   amazonlinux   45 k
 php8.3-xml              x86_64          8.3.10-1.amzn2023.0.1   amazonlinux  154 k
Installing weak dependencies:
 apr-util-openssl        x86_64          1.6.3-1.amzn2023.0.1    amazonlinux   17 k
 mod_http2               x86_64          2.0.27-1.amzn2023.0.3   amazonlinux  166 k
 mod_lua                 x86_64          2.4.62-1.amzn2023       amazonlinux   61 k
 php8.3-fpm              x86_64          8.3.10-1.amzn2023.0.1   amazonlinux  1.9 M
 php8.3-mbstring         x86_64          8.3.10-1.amzn2023.0.1   amazonlinux  528 k
 php8.3-opcache          x86_64          8.3.10-1.amzn2023.0.1   amazonlinux  379 k
 php8.3-pdo              x86_64          8.3.10-1.amzn2023.0.1   amazonlinux   89 k
 php8.3-sodium           x86_64          8.3.10-1.amzn2023.0.1   amazonlinux   41 k
```

Name:Sahil Motiramani Div:D15C Rollno:35

```
Installed:
  apr-1.7.2-2.amzn2023.0.2.x86_64                  apr-util-1.6.3-1.amzn2023.0.1.x86_64          apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64
  generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch httpd-2.4.62-1.amzn2023.x86_64              httpd-core-2.4.62-1.amzn2023.x86_64
  httpd-filesystem-2.4.62-1.amzn2023.noarch        httpd-tools-2.4.62-1.amzn2023.x86_64          libbrotli-1.0.9-4.amzn2023.0.2.x86_64
  libsodium-1.0.19-4.amzn2023.x86_64               libxslt-1.1.34-5.amzn2023.0.2.x86_64          mailcap-2.1.49-3.amzn2023.0.3.noarch
  mod_http2-2.0.27-1.amzn2023.0.3.x86_64           mod_lua-2.4.62-1.amzn2023.x86_64              nginx-filesystem-1:1.24.0-1.amzn2023.0.4.noarch
  php8.3-8.3.10-1.amzn2023.0.1.x86_64              php8.3-cli-8.3.10-1.amzn2023.0.1.x86_64       php8.3-common-8.3.10-1.amzn2023.0.1.x86_64
  php8.3-fpm-8.3.10-1.amzn2023.0.1.x86_64          php8.3-mbstring-8.3.10-1.amzn2023.0.1.x86_64  php8.3-opcache-8.3.10-1.amzn2023.0.1.x86_64
  php8.3-pdo-8.3.10-1.amzn2023.0.1.x86_64          php8.3-process-8.3.10-1.amzn2023.0.1.x86_64   php8.3-sodium-8.3.10-1.amzn2023.0.1.x86_64
  php8.3-xml-8.3.10-1.amzn2023.0.1.x86_64

Complete!
[ec2-user@ip-172-31-39-94 ~]$
```

Step 8: Next we install C/C++ compiler (GCC) along with the necessary C libraries required for compiling and running C programs. Use the following command. sudo yum install gcc glibc glibc-common

```
[ec2-user@ip-172-31-39-94 ~]$ sudo yum install gcc glibc glibc-common
Last metadata expiration check: 0:21:56 ago on Sun Oct  6 10:11:16 2024.
Package glibc-2.34-52.amzn2023.0.11.x86_64 is already installed.
Package glibc-common-2.34-52.amzn2023.0.11.x86_64 is already installed.
Dependencies resolved.
================================================================================
 Package                 Architecture  Version                    Repository  Size
================================================================================
Installing:
 gcc                     x86_64        11.4.1-2.amzn2023.0.2      amazonlinux  32 M
Installing dependencies:
 annobin-docs            noarch        10.93-1.amzn2023.0.1       amazonlinux  92 k
 annobin-plugin-gcc      x86_64        10.93-1.amzn2023.0.1       amazonlinux 887 k
 cpp                     x86_64        11.4.1-2.amzn2023.0.2      amazonlinux  10 M
 gc                      x86_64        8.0.4-5.amzn2023.0.2       amazonlinux 105 k
 glibc-devel             x86_64        2.34-52.amzn2023.0.11      amazonlinux  27 k
 glibc-headers-x86       noarch        2.34-52.amzn2023.0.11      amazonlinux 427 k
 guile22                 x86_64        2.2.7-2.amzn2023.0.3       amazonlinux 6.4 M
 kernel-headers          x86_64        6.1.109-118.189.amzn2023   amazonlinux 1.4 M
 libmpc                  x86_64        1.2.1-2.amzn2023.0.2       amazonlinux  62 k
 libtool-ltdl            x86_64        2.4.7-1.amzn2023.0.3       amazonlinux  38 k
 libxcrypt-devel         x86_64        4.4.33-7.amzn2023          amazonlinux  32 k
 make                    x86_64        1:4.3-5.amzn2023.0.2       amazonlinux 534 k
```

```
Installed:
  annobin-docs-10.93-1.amzn2023.0.1.noarch      annobin-plugin-gcc-10.93-1.amzn2023.0.1.x86_64  cpp-11.4.1-2.amzn2023.0.2.x86_64
  gc-8.0.4-5.amzn2023.0.2.x86_64                gcc-11.4.1-2.amzn2023.0.2.x86_64                glibc-devel-2.34-52.amzn2023.0.11.x86_64
  glibc-headers-x86-2.34-52.amzn2023.0.11.noarch guile22-2.2.7-2.amzn2023.0.3.x86_64            kernel-headers-6.1.109-118.189.amzn2023.x86_64
  libmpc-1.2.1-2.amzn2023.0.2.x86_64            libtool-ltdl-2.4.7-1.amzn2023.0.3.x86_64        libxcrypt-devel-4.4.33-7.amzn2023.x86_64
  make-1:4.3-5.amzn2023.0.2.x86_64

Complete!
[ec2-user@ip-172-31-39-94 ~]$
```

Step 9: We would also need GD library and its development tools. For that, run this command sudo yum install gd gd-devel

```
[ec2-user@ip-172-31-39-94 ~]$ sudo yum install gd gd-devel
Last metadata expiration check: 0:23:03 ago on Sun Oct  6 10:11:16 2024.
Dependencies resolved.
================================================================================
 Package                 Architecture  Version                    Repository  Size
================================================================================
Installing:
 gd                      x86_64        2.3.3-5.amzn2023.0.3       amazonlinux 139 k
 gd-devel                x86_64        2.3.3-5.amzn2023.0.3       amazonlinux  38 k
Installing dependencies:
 brotli                  x86_64        1.0.9-4.amzn2023.0.2       amazonlinux 314 k
 brotli-devel            x86_64        1.0.9-4.amzn2023.0.2       amazonlinux  31 k
 bzip2-devel             x86_64        1.0.8-6.amzn2023.0.2       amazonlinux 214 k
 cairo                   x86_64        1.17.6-2.amzn2023.0.1      amazonlinux 684 k
 cmake-filesystem        x86_64        3.22.2-1.amzn2023.0.4      amazonlinux  16 k
 fontconfig              x86_64        2.13.94-2.amzn2023.0.2     amazonlinux 273 k
 fontconfig-devel        x86_64        2.13.94-2.amzn2023.0.2     amazonlinux 128 k
 fonts-filesystem        noarch        1:2.0.5-12.amzn2023.0.2    amazonlinux 9.5 k
 freetype                x86_64        2.13.2-5.amzn2023.0.1      amazonlinux 423 k
 freetype-devel          x86_64        2.13.2-5.amzn2023.0.1      amazonlinux 912 k
```

```
Installed:
  brotli-1.0.9-4.amzn2023.0.2.x86_64                brotli-devel-1.0.9-4.amzn2023.0.2.x86_64          bzip2-devel-1.0.8-6.amzn2023.0.2.x86_64
  cairo-1.17.6-2.amzn2023.0.1.x86_64                cmake-filesystem-3.22.2-1.amzn2023.0.4.x86_64     fontconfig-2.13.94-2.amzn2023.0.2.x86_64
  fontconfig-devel-2.13.94-2.amzn2023.0.2.x86_64    fonts-filesystem-1:2.0.5-12.amzn2023.0.2.noarch   freetype-2.13.2-5.amzn2023.0.1.x86_64
  freetype-devel-2.13.2-5.amzn2023.0.1.x86_64       gd-2.3.3-5.amzn2023.0.3.x86_64                    gd-devel-2.3.3-5.amzn2023.0.3.x86_64
  glib2-devel-2.74.7-689.amzn2023.0.2.x86_64        google-noto-fonts-common-20201206-2.amzn2023.0.2.noarch google-noto-sans-vf-fonts-20201206-2.amzn2023.0.2.noarch
  graphite2-1.3.14-7.amzn2023.0.2.x86_64            graphite2-devel-1.3.14-7.amzn2023.0.2.x86_64      harfbuzz-7.0.0-2.amzn2023.0.1.x86_64
  harfbuzz-devel-7.0.0-2.amzn2023.0.1.x86_64        harfbuzz-icu-7.0.0-2.amzn2023.0.1.x86_64          jbigkit-libs-2.1-21.amzn2023.0.2.x86_64
  langpacks-core-font-en-3.0-21.amzn2023.0.4.noarch libICE-1.0.10-6.amzn2023.0.2.x86_64              libSM-1.2.3-8.amzn2023.0.2.x86_64
  libX11-1.7.2-3.amzn2023.0.4.x86_64                libX11-common-1.7.2-3.amzn2023.0.4.noarch         libX11-devel-1.7.2-3.amzn2023.0.4.x86_64
  libX11-xcb-1.7.2-3.amzn2023.0.4.x86_64            libXau-1.0.9-6.amzn2023.0.2.x86_64                libXau-devel-1.0.9-6.amzn2023.0.2.x86_64
  libXext-1.3.4-6.amzn2023.0.2.x86_64               libXpm-3.5.15-2.amzn2023.0.3.x86_64               libXpm-devel-3.5.15-2.amzn2023.0.3.x86_64
  libXrender-0.9.10-14.amzn2023.0.2.x86_64          libXt-1.2.0-4.amzn2023.0.2.x86_64                 libblkid-devel-2.37.4-1.amzn2023.0.4.x86_64
  libffi-devel-3.4.4-1.amzn2023.0.1.x86_64          libicu-67.1-7.amzn2023.0.3.x86_64                 libicu-devel-67.1-7.amzn2023.0.3.x86_64
  libjpeg-turbo-2.1.4-2.amzn2023.0.5.x86_64         libjpeg-turbo-devel-2.1.4-2.amzn2023.0.5.x86_64   libmount-devel-2.37.4-1.amzn2023.0.4.x86_64
  libpng-2:1.6.37-10.amzn2023.0.6.x86_64            libpng-devel-2:1.6.37-10.amzn2023.0.6.x86_64      libselinux-devel-3.4-5.amzn2023.0.2.x86_64
  libsepol-devel-3.4-3.amzn2023.0.3.x86_64          libtiff-4.4.0-4.amzn2023.0.18.x86_64              libtiff-devel-4.4.0-4.amzn2023.0.18.x86_64
  libwebp-1.2.4-1.amzn2023.0.6.x86_64               libwebp-devel-1.2.4-1.amzn2023.0.6.x86_64         libxcb-1.13.1-7.amzn2023.0.2.x86_64
  libxcb-devel-1.13.1-7.amzn2023.0.2.x86_64         libxml2-devel-2.10.4-1.amzn2023.0.6.x86_64        pcre2-devel-10.40-1.amzn2023.0.3.x86_64
  pcre2-utf16-10.40-1.amzn2023.0.3.x86_64           pcre2-utf32-10.40-1.amzn2023.0.3.x86_64           pixman-0.40.0-3.amzn2023.0.3.x86_64
  sysprof-capture-devel-3.40.1-2.amzn2023.0.2.x86_64 xml-common-0.6.3-56.amzn2023.0.2.noarch         xorg-x11-proto-devel-2021.4-1.amzn2023.0.2.noarch
  xz-devel-5.2.5-9.amzn2023.0.2.x86_64              zlib-devel-1.2.11-33.amzn2023.0.5.x86_64
```

Name:Sahil Motiramani Div:D15C Rollno:35

Step 10: Now, we create a user called as ‘nagios’ and make sure that it has a home directory, and set up a password for it. sudo adduser -m nagios sudo passwd nagios

```
[ec2-user@ip-172-31-39-94 ~]$ sudo adduser -m nagios
sudo passwd nagios
Changing password for user nagios.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[ec2-user@ip-172-31-39-94 ~]$
```

Step 11: Create a user group called as ‘nagcmd’ to execute nagios commands.
sudo groupadd nagcmd

```
[ec2-user@ip-172-31-39-94 ~]$ sudo groupadd nagcmd
[ec2-user@ip-172-31-39-94 ~]$
```

Step 12: Add users apache and nagios to this user group.
sudo usermod -a -G nagcmd nagios
sudo usermod -a -G nagcmd apache

```
[ec2-user@ip-172-31-39-94 ~]$ sudo usermod -a -G nagcmd nagios
sudo usermod -a -G nagcmd apache
[ec2-user@ip-172-31-39-94 ~]$
```

Step 13: We create a directory downloads, to store the files of nagios server that are downloaded. mkdir ~/downloads cd ~/downloads

```
[ec2-user@ip-172-31-39-94 ~]$ mkdir ~/downloads
cd ~/downloads
[ec2-user@ip-172-31-39-94 downloads]$
```

Step 14:Now we need to install the latest versions of nogios-core and nagios-plugins. Go to the respective websites and check whether a better version is available.If newer versions are available, then right click on the download button → Copy link address.
Paste this link address in place of the current link in command.
If not run these commands.

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

Name:Sahil Motiramani Div:D15C Rollno:35

```
[ec2-user@ip-172-31-83-157 downloads]$ wget https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.5.tar.gz
--2024-09-28 04:04:23--  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%[=========================================================================>]   1.97M  5.36MB/s    in 0.4s

2024-09-28 04:04:24 (5.36 MB/s) - 'nagios-4.5.5.tar.gz' saved [2065473/2065473]

[ec2-user@ip-172-31-83-157 downloads]$
```

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

```
[ec2-user@ip-172-31-39-94 downloads]$ wget https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.5.tar.gz
--2024-10-06 10:37:57--  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%[=========================================================================>]   1.97M  6.17MB/s    in 0.3s

2024-10-06 10:37:58 (6.17 MB/s) - 'nagios-4.5.5.tar.gz' saved [2065473/2065473]

[ec2-user@ip-172-31-39-94 downloads]$
```

Step 15:Now, we need to extract nagios-core file into the same directory. For that, we will use tar command.

tar zxvf nagios-4.5.5.tar.gz

```
[ec2-user@ip-172-31-39-94 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
nagios-4.5.5/README.md
nagios-4.5.5/THANKS
```

```
nagios-4.5.5/xdata/.gitignore
nagios-4.5.5/xdata/Makefile.in
nagios-4.5.5/xdata/xcddefault.c
nagios-4.5.5/xdata/xcddefault.h
nagios-4.5.5/xdata/xodtemplate.c
nagios-4.5.5/xdata/xodtemplate.h
nagios-4.5.5/xdata/xpddefault.c
nagios-4.5.5/xdata/xpddefault.h
nagios-4.5.5/xdata/xrddefault.c
nagios-4.5.5/xdata/xrddefault.h
nagios-4.5.5/xdata/xsddefault.c
nagios-4.5.5/xdata/xsddefault.h
[ec2-user@ip-172-31-39-94 downloads]$
```

Step 16: We need to ensure that Nagios uses a specific group (in this case, `nagcmd`) for executing external commands.

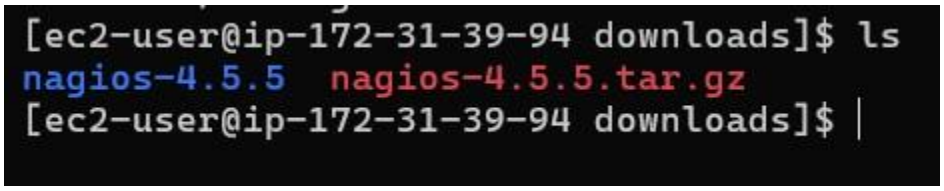
Name:Sahil Motiramani Div:D15C Rollno:35

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

An error could be encountered here: ./configure: no such path or directory

Solution: Navigate to the nagios-4.5.5 folder in downloads. (version could vary)

Steps: ls

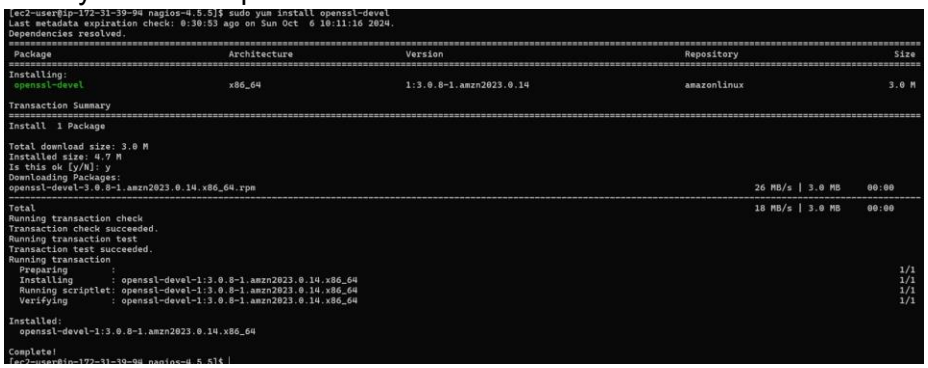


- cd nagios-4.5.5 (use the version shown by your ls command)
- ./configure --with-command-group=nagcmd

Another error could be Cannot find SSL headers .

To solve this, we need to install OpenSSL Dev Library

Steps:
sudo yum install openssl-devel



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

Name:Sahil Motiramani Div:D15C Rollno:35

```
[ec2-user@ip-172-31-39-94 nagios-4.5.5]$ ./configure --with-command-group=nagcmd
checking for a BSD-compatible install... /usr/bin/install -c
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether the compiler supports GNU C... yes
checking whether gcc accepts -g... yes
checking for gcc option to enable C11 features... none needed
checking whether make sets $(MAKE)... yes
```

```
*** 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.

[ec2-user@ip-172-31-39-94 nagios-4.5.5]$
```

Step 17: We need to compile all components of this software according to the instruction in the Makefile. For that, use this command:
make all

Then, sudo make install sudo make install-init sudo make install-config sudo make install-commandmode

Name:Sahil Motiramani Div:D15C Rollno:35

```
[ec2-user@ip-172-31-39-94 nagios-4.5.5]$ sudo make install
sudo make install-init
sudo make install-config
sudo make install-commandmode
cd ./base && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/base'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagios /usr/local/nagios/bin
/usr/bin/install: cannot stat 'nagios': No such file or directory
make[1]: *** [Makefile:188: install] Error 1
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5/base'
```

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

Remember, these are *SAMPLE* config files.  You'll need to read
the documentation for more information on how to actually define
services, hosts, etc. to fit your particular needs.

/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 ***
```

Step 18: We need to update the email linked with this server to our email for it to send notifications (if any needed). sudo nano /usr/local/nagios/etc/objects/contacts.cfg

```
  GNU nano 5.8                                  /usr/local/nagios/etc/objects/contacts.cfg
##############################################################################
# CONTACTS.CFG - SAMPLE CONTACT/CONTACTGROUP DEFINITIONS
#
#
# NOTES: This config file provides you with some example contact and contact
#        group definitions that you can reference in host and service
#        definitions.
#
#        You don't need to keep these definitions in a separate file from your
#        other object definitions.  This has been done just to make things
#        easier to understand.
#
##############################################################################



##############################################################################
#
# CONTACTS
#
##############################################################################

# Just one contact defined by default - the Nagios admin (that's you)
# This contact definition inherits a lot of default values from the
# 'generic-contact' template which is defined elsewhere.

define contact {

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



##############################################################################
#
# CONTACT GROUPS
#
##############################################################################
```

Here, change the email under ‘define contact{}’ to your email address.
To save this use the following shortcut sequence CTRL+O→Enter→CTRL+X.

Name:Sahil Motiramani Div:D15C Rollno:35

CTRL+O: Overwrite the existing file with edited file
CTRL+X: Exit nano editor.

Step 19: We need to install the necessary configuration files for the Nagios web interface. Run the following command. sudo make install-webconf

```
[ec2-user@ip-172-31-39-94 nagios-4.5.5]$ sudo make install-webconf
/usr/bin/install -c -m 644 sample-config/httpd.conf /etc/httpd/conf.d/nagios.conf
if [ 0 -eq 1 ]; then \
        ln -s /etc/httpd/conf.d/nagios.conf /etc/apache2/sites-enabled/nagios.conf; \
fi

*** Nagios/Apache conf file installed ***

[ec2-user@ip-172-31-39-94 nagios-4.5.5]$ S
```

Step 20: Now we need to setup a user to access this nagios web interface. So we run this command to create a user called ‘nagiosadmin’.

Keep this username and password saved as it is needed to login to the web interface.
sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin

```
[ec2-user@ip-172-31-39-94 nagios-4.5.5]$ sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
New password:
Re-type new password:
Adding password for user nagiosadmin
[ec2-user@ip-172-31-39-94 nagios-4.5.5]$
```

Step 21: Restart the apache server to apply all the recent configurations.
sudo service httpd restart

```
[ec2-user@ip-172-31-39-94 nagios-4.5.5]$ sudo service httpd restart
Redirecting to /bin/systemctl restart httpd.service
[ec2-user@ip-172-31-39-94 nagios-4.5.5]$
```

Step 22: Now we go back to the downloads folder and extract the files of nagios plugin. cd ~/downloads tar zxvf nagios-plugins-2.4.11.tar.gz (Version may vary)

Name:Sahil Motiramani Div:D15C Rollno:35

```
[ec2-user@ip-172-31-39-94 downloads]$ tar zxvf nagios-plugins-2.4.11.tar.gz
nagios-plugins-2.4.11/
nagios-plugins-2.4.11/build-aux/
nagios-plugins-2.4.11/build-aux/compile
nagios-plugins-2.4.11/build-aux/config.guess
nagios-plugins-2.4.11/build-aux/config.rpath
nagios-plugins-2.4.11/build-aux/config.sub
nagios-plugins-2.4.11/build-aux/install-sh
nagios-plugins-2.4.11/build-aux/ltmain.sh
nagios-plugins-2.4.11/build-aux/missing
nagios-plugins-2.4.11/build-aux/mkinstalldirs
nagios-plugins-2.4.11/build-aux/depcomp
nagios-plugins-2.4.11/build-aux/snippet/
```

```
nagios-plugins-2.4.11/po/fr.gmo
nagios-plugins-2.4.11/po/de.gmo
nagios-plugins-2.4.11/po/nagios-plugins.pot
nagios-plugins-2.4.11/po/stamp-po
nagios-plugins-2.4.11/po/ChangeLog
nagios-plugins-2.4.11/po/LINGUAS
nagios-plugins-2.4.11/release
[ec2-user@ip-172-31-39-94 downloads]$
```

Step 23: Again, we need to install the configurations for these files.
cd nagios-plugins-2.4.11 (version may vary)
./configure --with-nagios-user=nagios --with-nagios-group=nagios

```
[ec2-user@ip-172-31-39-94 downloads]$ cd nagios-plugins-2.4.11
[ec2-user@ip-172-31-39-94 nagios-plugins-2.4.11]$ ./configure --with-nagios-user=nagios --with-nagios-group=nagios
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /usr/bin/mkdir -p
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking whether make supports nested variables... yes
checking whether to enable maintainer-specific portions of Makefiles... yes
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking for gcc... gcc
```

```
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
[ec2-user@ip-172-31-39-94 nagios-plugins-2.4.11]$
```

Step 24: We need to compile all components of this software according to the instruction in the Makefile. For that, use the commands:
make sudo
make install

Name:Sahil Motiramani Div:D15C Rollno:35

```
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-plugins-2.4.11/po'
make[1]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11'
make[2]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11'
make[2]: Nothing to be done for 'install-exec-am'.
make[2]: Nothing to be done for 'install-data-am'.
make[2]: Leaving directory '/home/ec2-user/downloads/nagios-plugins-2.4.11'
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-plugins-2.4.11'
[ec2-user@ip-172-31-39-94 nagios-plugins-2.4.11]$
```

Step 25: We need to register the Nagios service with the system, which would make it able to manage the server status. So run the following commands sudo chkconfig --add nagios sudo chkconfig nagios on

```
[ec2-user@ip-172-31-39-94 nagios-4.4.6]$ sudo systemctl enable nagios
[ec2-user@ip-172-31-39-94 nagios-4.4.6]$ sudo systemctl start nagios
```

Step 26: We need to verify the Nagios configuration for any syntax errors or issues before starting or restarting the Nagios service.
sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

```
[ec2-user@ip-172-31-39-94 nagios-4.4.6]$ sudo make install-webconf
/usr/bin/install -c -m 644 sample-config/httpd.conf /etc/httpd/conf.d/nagios.conf
if [ 0 -eq 1 ]; then \
        ln -s /etc/httpd/conf.d/nagios.conf /etc/apache2/sites-enabled/nagios.conf; \
fi

*** Nagios/Apache conf file installed ***

[ec2-user@ip-172-31-39-94 nagios-4.4.6]$ sudo systemctl enable nagios
[ec2-user@ip-172-31-39-94 nagios-4.4.6]$ sudo systemctl start nagios
[ec2-user@ip-172-31-39-94 nagios-4.4.6]$ sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

Nagios Core 4.4.6
Copyright (c) 2009-present Nagios Core Development Team and Community Contributors
Copyright (c) 1999-2009 Ethan Galstad
Last Modified: 2020-04-28
License: GPL

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

Running pre-flight check on configuration data...

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

Total Warnings: 0
Total Errors:   0

Things look okay - No serious problems were detected during the pre-flight check
[ec2-user@ip-172-31-39-94 nagios-4.4.6]$
```

Step 27: sudo service nagios start

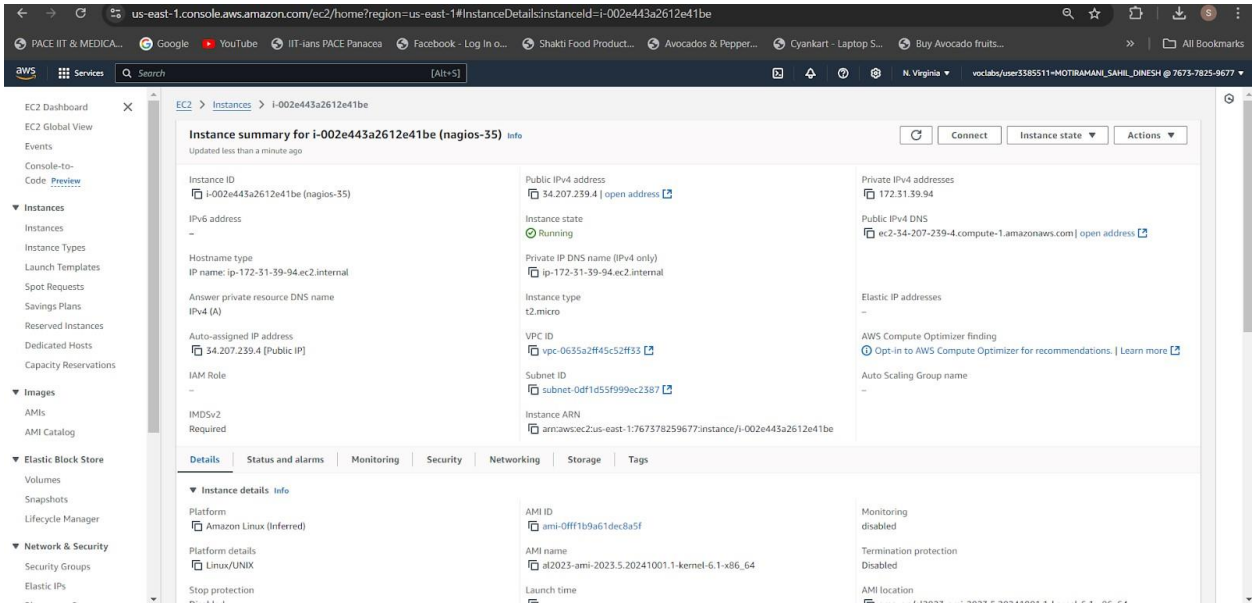
```
[ec2-user@ip-172-31-39-94 nagios-4.4.6]$ sudo service nagios start
Redirecting to /bin/systemctl start nagios.service
```

sudo systemctl status nagios

Name:Sahil Motiramani Div:D15C Rollno:35

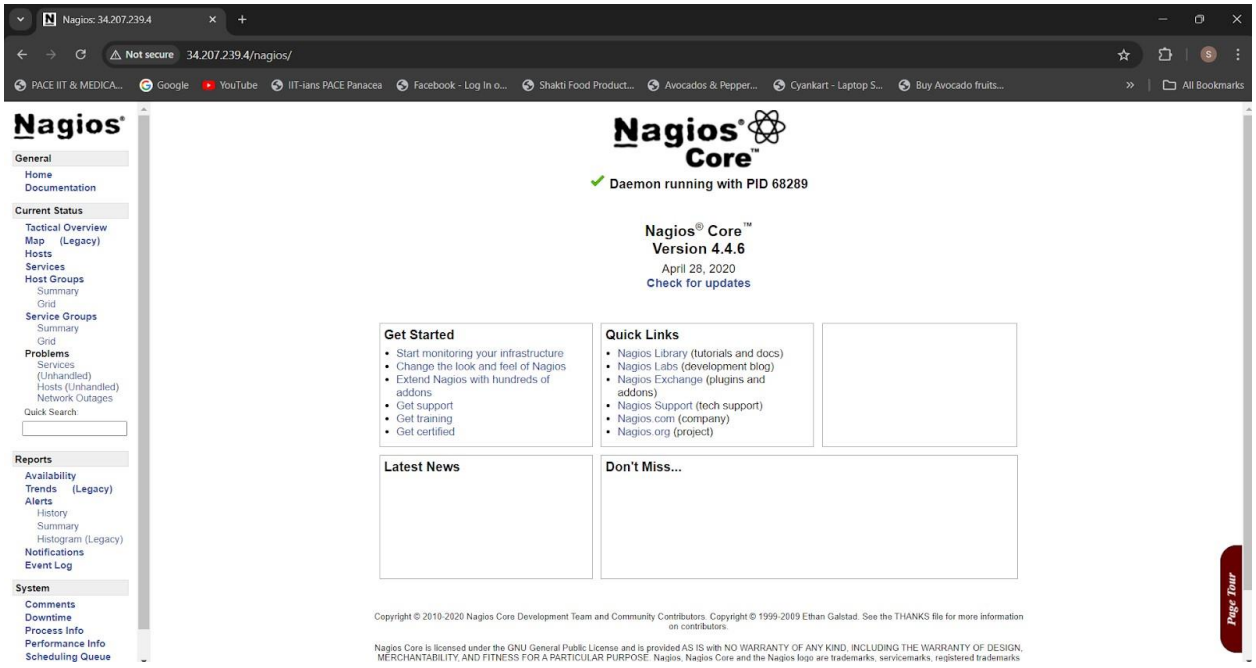
```
[ec2-user@ip-172-31-39-94 nagios-4.4.6]$ sudo systemctl status nagios
● nagios.service - Nagios Core 4.4.6
     Loaded: loaded (/usr/lib/systemd/system/nagios.service; enabled; preset: disabled)
     Active: active (running) since Sun 2024-10-06 11:02:08 UTC; 2min 4s ago
       Docs: https://www.nagios.org/documentation
    Process: 68287 ExecStartPre=/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
    Process: 68288 ExecStart=/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
   Main PID: 68289 (nagios)
      Tasks: 6 (limit: 1112)
     Memory: 2.4M
        CPU: 42ms
     CGroup: /system.slice/nagios.service
             ├─68289 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─68290 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─68291 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─68292 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─68293 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             └─68294 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg

Oct 06 11:02:08 ip-172-31-39-94.ec2.internal nagios[68289]: qh: core query handler registered
Oct 06 11:02:08 ip-172-31-39-94.ec2.internal nagios[68289]: qh: echo service query handler registered
Oct 06 11:02:08 ip-172-31-39-94.ec2.internal nagios[68289]: qh: help for the query handler registered
Oct 06 11:02:08 ip-172-31-39-94.ec2.internal nagios[68289]: wproc: Successfully registered manager as @wproc with query handler
Oct 06 11:02:08 ip-172-31-39-94.ec2.internal nagios[68289]: wproc: Registry request: name=Core Worker 68291;pid=68291
Oct 06 11:02:08 ip-172-31-39-94.ec2.internal nagios[68289]: wproc: Registry request: name=Core Worker 68292;pid=68292
Oct 06 11:02:08 ip-172-31-39-94.ec2.internal nagios[68289]: wproc: Registry request: name=Core Worker 68293;pid=68293
Oct 06 11:02:08 ip-172-31-39-94.ec2.internal nagios[68289]: wproc: Registry request: name=Core Worker 68290;pid=68290
Oct 06 11:02:10 ip-172-31-39-94.ec2.internal nagios[68289]: Successfully launched command file worker with pid 68294
Oct 06 11:04:00 ip-172-31-39-94.ec2.internal nagios[68289]: SERVICE ALERT: localhost;HTTP;WARNING;SOFT;1;HTTP WARNING: HTTP/1.1 403 Forbidden - 319 bytes in 0.003 second response time
[ec2-user@ip-172-31-39-94 nagios-4.4.6]$
```

Step 28: Now, go to EC2 instance and click on instance id. Then, click on the copy icon just before the public ip address on public IP.



Step 29: Open a new tab. In the address bar type http://<publicipaddress>/nagios.
This would be in the output

Name:Sahil Motiramani Div:D15C Rollno:35



Conclusion:

In this experiment, we have learned how to install and set up Nagios Core, Nagios Plugins, and NRPE on a Linux machine. We are working with an Amazon Linux OS instance that has been configured with the necessary security rules. It's crucial to ensure that the links for Nagios Core and Nagios Plugins used in the wget commands are current and up-to-date. After downloading, we need to extract and configure these files to prevent any issues when starting the server. Once the setup is complete, we can launch the Nagios server. By using the public IP address of the EC2 instance, we can access the Nagios dashboard by entering that IP into a web browser.

Name:Sahil Motiramani Div:D15C RollNo:35

Exp:10

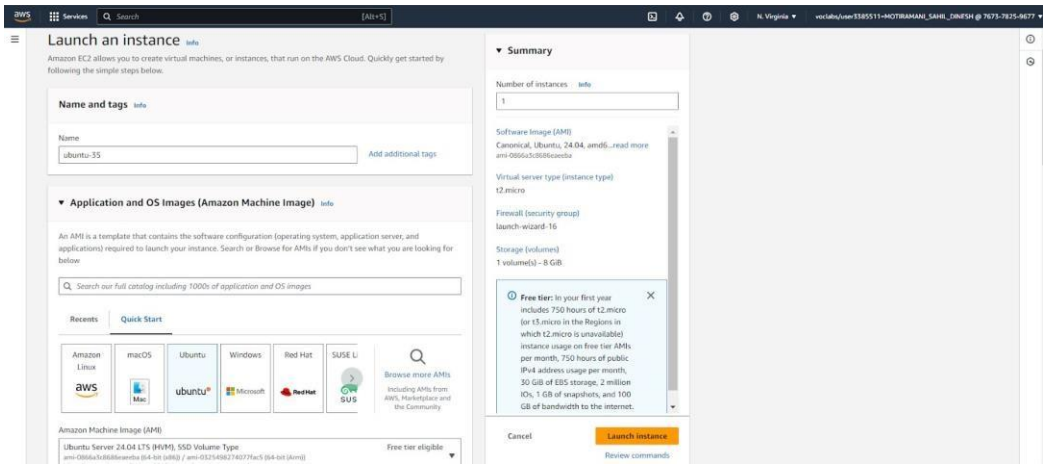
Aim: To perform Port, Service monitoring, Windows/Linux server monitoring using Nagios.

Prerequisites:

1) An Amazon Linux instance with nagios already set up.

Step 1: Set up ubuntu instance

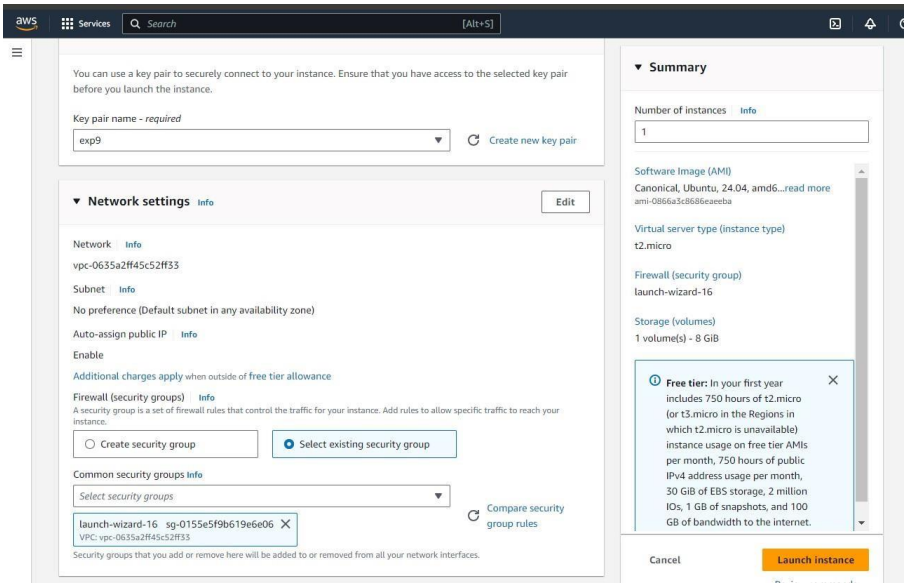
1) Login to your AWS account. Search for EC2 on services. Open the interface and click on Create Instance.



Select The OS Image as Ubuntu.

2) Make sure to select the same private key that you created for the Amazon Linux instance. Also select the same security group as you created for the Linux instance.

Name:Sahil Motiramani Div:D15C RollNo:35



3) Now come back to the instances screen. Click on the instance ID of your instance. Then click on Connect. Click on SSH client. Copy the example command. Now, we have to connect our local OS terminal to the instance using SSH. For this, open terminal wher the private key file is located (.pem). Paste the copied SSH command and run it.

Step 2: Execute the following on Nagios Host machine (Linux)

1) We need to verify whether the nagios service is running or not. Fo that, run this command.

ps -ef | grep nagios

```
[ec2-user@ip-172-31-39-94 nagios-4.4.6]$ ps -ef | grep nagios
nagios     68289       1  0 11:02 ?        00:00:00 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
nagios     68290   68289  0 11:02 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios     68291   68289  0 11:02 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios     68292   68289  0 11:02 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios     68293   68289  0 11:02 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios     68294   68289  0 11:02 ?        00:00:00 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
ec2-user   71786    2942  0 11:51 pts/0    00:00:00 grep --color=auto nagios
[ec2-user@ip-172-31-39-94 nagios-4.4.6]$
```

2) Now, make yourself as the root user, and create a folder with the path ‘/usr/local/nagios/etc/objects/monitorhosts/linuxhosts’ **sudo su mkdir -p /usr/local/nagios/etc/objects/monitorhosts/linuxhosts**

```
[ec2-user@ip-172-31-39-94 nagios-4.4.6]$ sudo su
mkdir -p /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
[root@ip-172-31-39-94 nagios-4.4.6]#
```

3) We need to create a config file in this folder. So, copy the contents of the existing localhost config to the new file ‘linuxserver.cfg’. **cp /usr/local/nagios/etc/objects/localhost.cfg**

Name:Sahil Motiramani Div:D15C RollNo:35

/usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg 4) We need to make some changes in this config file. Open it using nano editor.

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

Change **hostname** and **alias** to **linuxserver**

Change address to **public ip address of client instance** (Ubuntu instance)

```
# 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               localhost
    alias                   localhost
    address                 3.80.168.49
}
```

Change hostgroup_name to **linux-servers1**

```
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 >
        }
```

Change the **occurrences of hostname** further in the document from **localhost** to **linuxserver**

5) Now, we need to edit the nagios configuration file to add this directory. **nano /usr/local/nagios/etc/nagios.cfg** Run this command and add the following line **cfg_dir=/usr/local/nagios/etc/objects/monitorhosts/**

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

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

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

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


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

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

cfg_dir=/usr/local/nagios/etc/objects/monitorhosts/
```

6) Now we verify the configuration files.

Name:Sahil Motiramani Div:D15C RollNo:35

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

```
[root@ip-172-31-39-94 nagios-4.4.6]# /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

Nagios Core 4.4.6
Copyright (c) 2009-present Nagios Core Development Team and Community Contributors
Copyright (c) 1999-2009 Ethan Galstad
Last Modified: 2020-04-28
License: GPL

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

Running pre-flight check on configuration data...

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

Total Warnings: 0
Total Errors:   0

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

7) Once the files are verified, we need to restart the server.

service nagios restart

```
[root@ip-172-31-39-94 nagios-4.4.6]# service nagios restart
Redirecting to /bin/systemctl restart nagios.service
[root@ip-172-31-39-94 nagios-4.4.6]#
```

Step 3: Execute the following on Nagios Client machine (Ubuntu)

1) First, we check for any new updates, then we install gcc, nagios nrpe server and nagios plugins.

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

Name:Sahil Motiramani Div:D15C RollNo:35

```
ubuntu@ip-172-31-44-65:~$ sudo apt update -y
sudo apt install gcc -y
sudo apt install -y nagios-nrpe-server nagios-plugins
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:6 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [382 kB]
Get:7 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [83.9 kB]
Get:8 http://security.ubuntu.com/ubuntu noble-security/main amd64 c-n-f Metadata [4704 B]
Get:9 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [277 kB]
```

```
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...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-44-65:~$
```

2) We need to add the public IP address of our host Nagios machine (Linux) to the nrpe configuration file.

   sudo nano /etc/nagios/nrpe.cfg

   Under allowed_hosts, add the nagios host ip address (public)

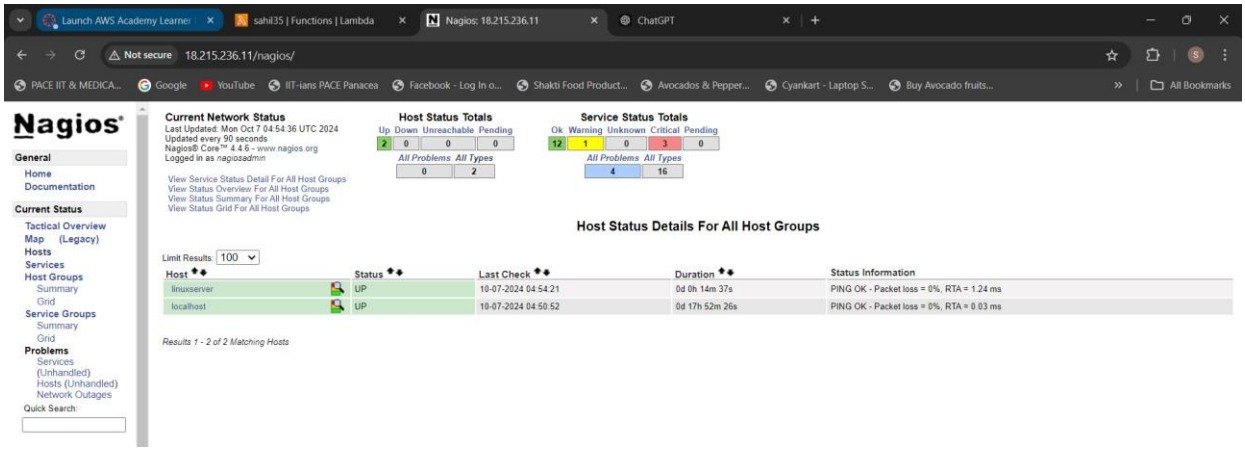
Name:Sahil Motiramani Div:D15C RollNo:35

```
# NRPE USER
# This determines the effective user that the NRPE daemon should run as.
# You can either supply a username or a UID.
#
# NOTE: This option is ignored if NRPE is running under either inetd or xinetd

nrpe_user=nagios



# NRPE GROUP
# This determines the effective group that the NRPE daemon should run as.
# You can either supply a group name or a GID.
#
# NOTE: This option is ignored if NRPE is running under either inetd or xinetd

nrpe_group=nagios



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

allowed_hosts=127.0.0.1,34.207.239.4
```

Step 4: Check the Nagios Dashboard 1)

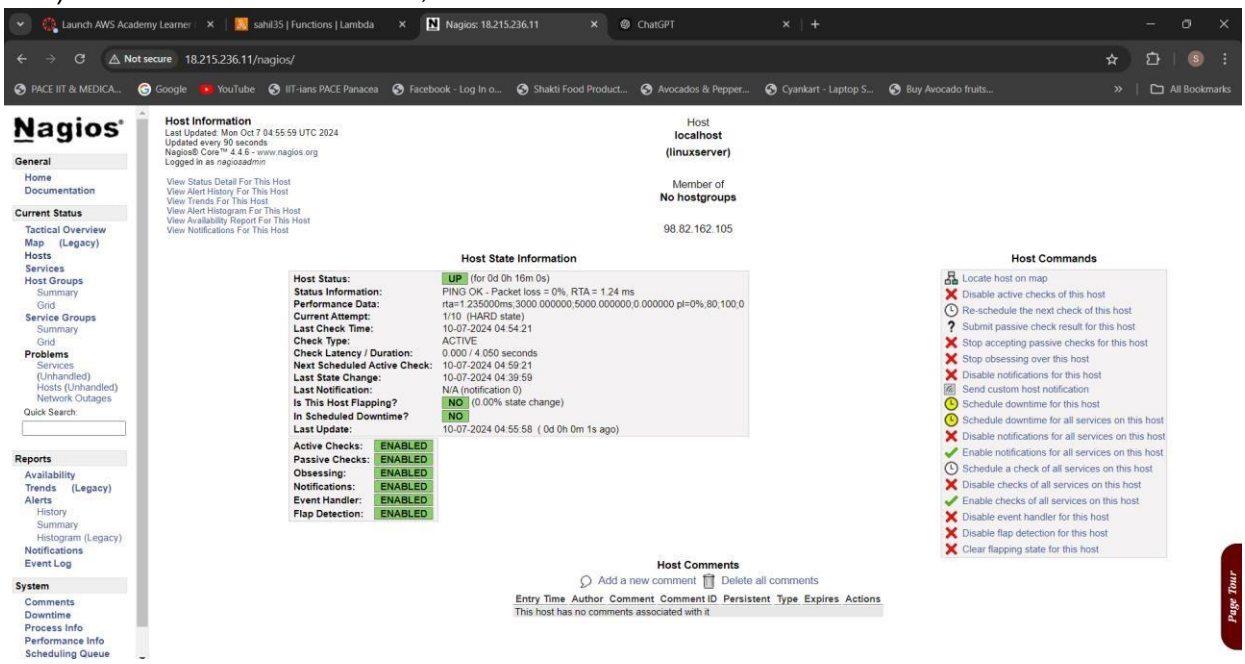
Go to Nagios dashboard, click on hosts.

Here, we can see that the linuxserver is also added as a host.

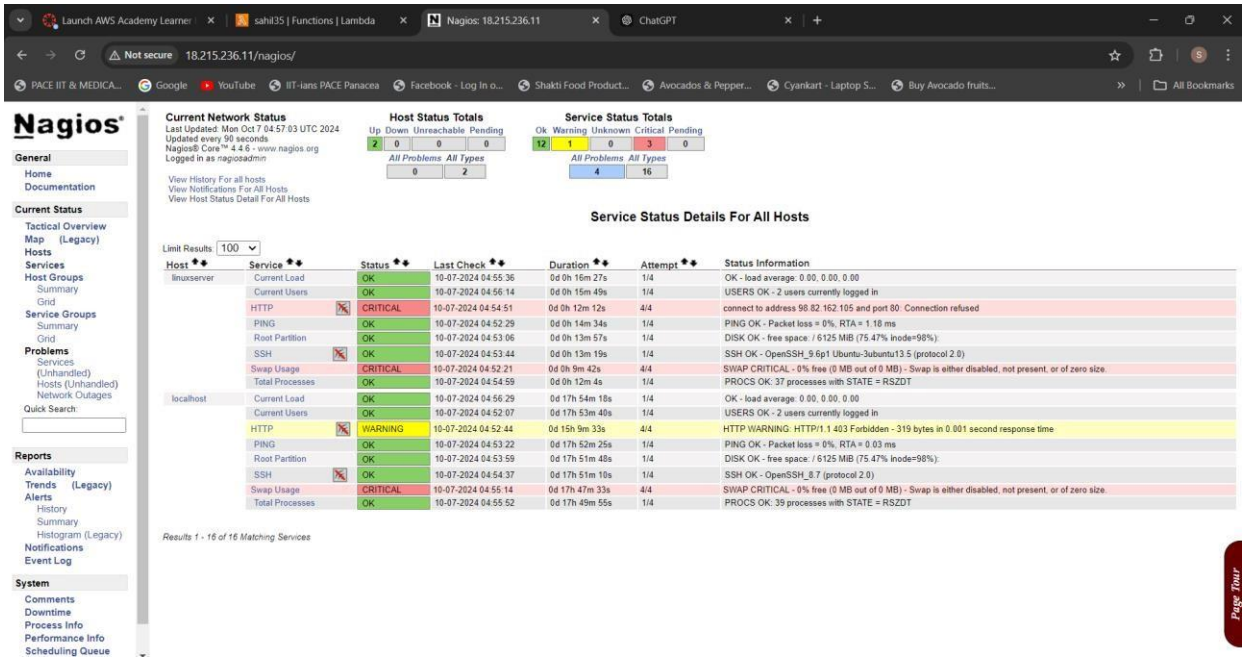


Name:Sahil Motiramani Div:D15C RollNo:35

2) Click on linuxserver. Here, we can check all the information about linuxserver host.



3) Click on services. Here we an see all the services that are being monitored by linuxserver.



Name:Sahil Motiramani Div:D15C RollNo:35

In this case, we have monitored -
Servers: 1 linux server
Services: swap
Ports: 22, 80 (ssh, http)
Processes: User status, Current load, total processes, root partition, etc.

Conclusion:

In this experiment, we set up port and server monitoring using Nagios. 1.

Linux Instance: Hosts the Nagios dashboard and server.

2. Ubuntu Instance : Acts as the second monitored host.
3. Configuration:
   - Add the Ubuntu instance’s IP to the Nagios server’s configuration.
   - On the Ubuntu instance, configure the NRPE server and allow the Nagios server's IP.
4. Restart NRPE: After configuration, restart the NRPE service on Ubuntu.
5. Monitor: The Ubuntu instance will appear as "linuxserver" on the Nagios dashboard for monitoring.

Name:Sahil Motiramani Div:D15C Roll no:35

Exp:11

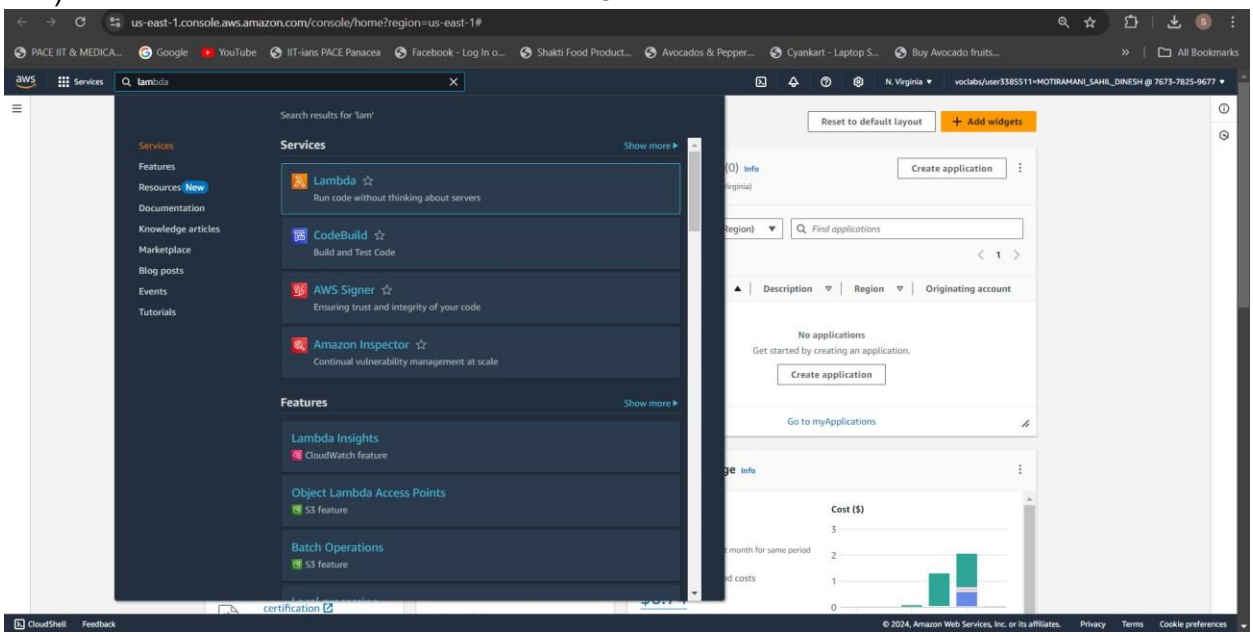
Aim: To understand AWS Lambda, its workflow, various functions and create your first Lambda functions using Python / Java / Nodejs.

Prerequisites:

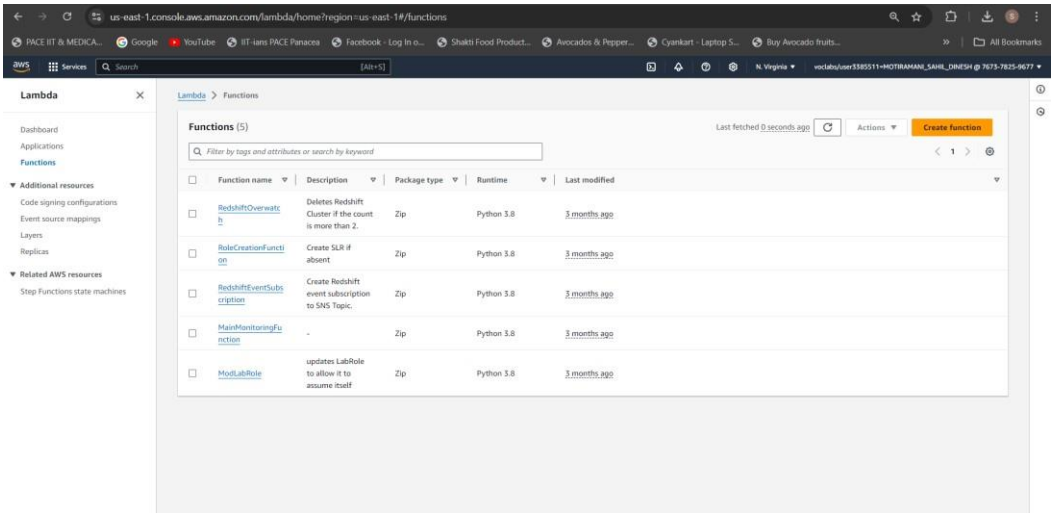
1) AWS account (academy recommended)

Step 1: Set up AWS Lambda Function

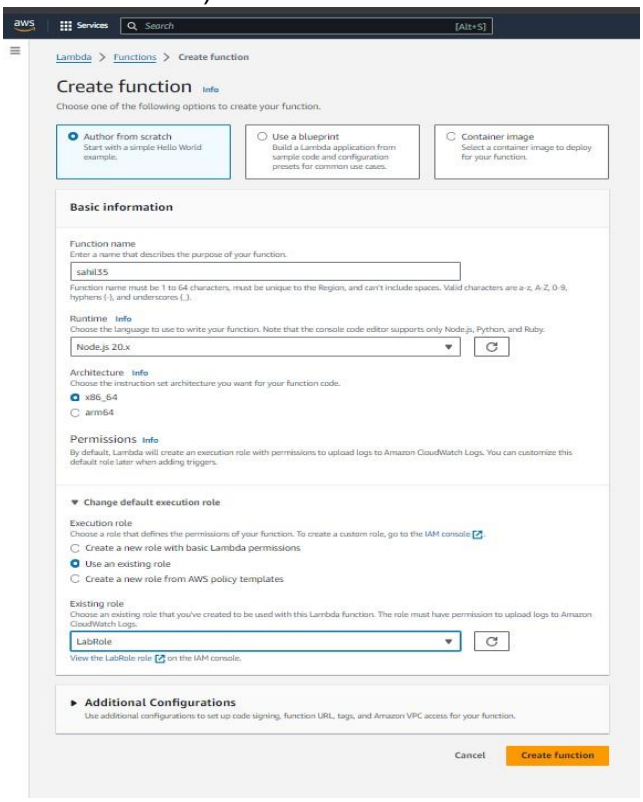
1) Search for Lambda in the services tab. Click on it once found.



Name:Sahil Motiramani Div:D15C Roll no:35

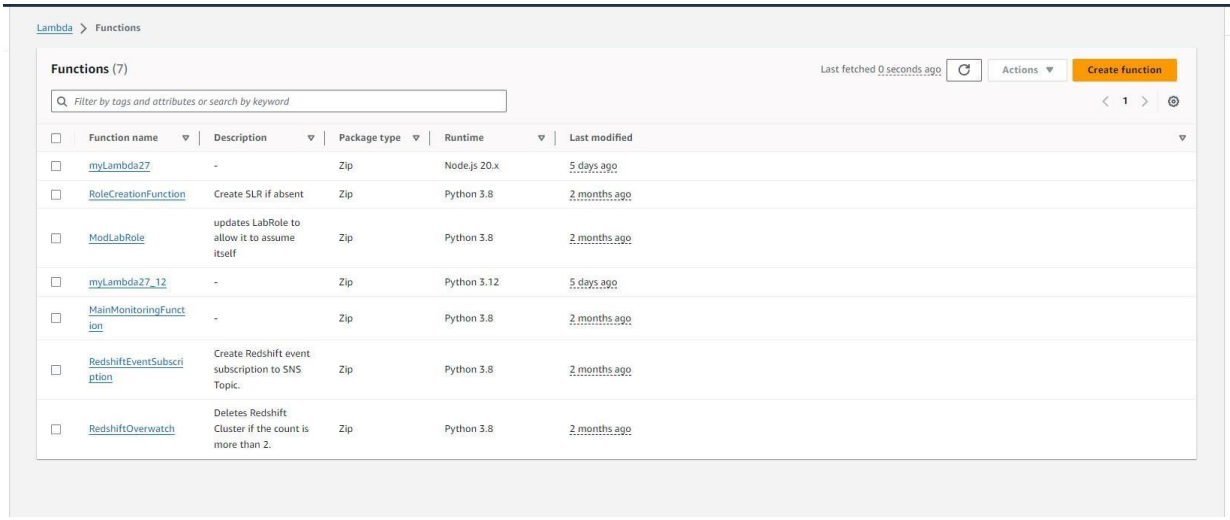


2) Click on create functions.
3) Give a name to your Lambda function. Select the runtime as Node.js 20.x (You can also use python). Select the architecture as x86_64. Set the default execution role as LabRole if you are doing this on your academy account. (Use an existing role → LabRole)

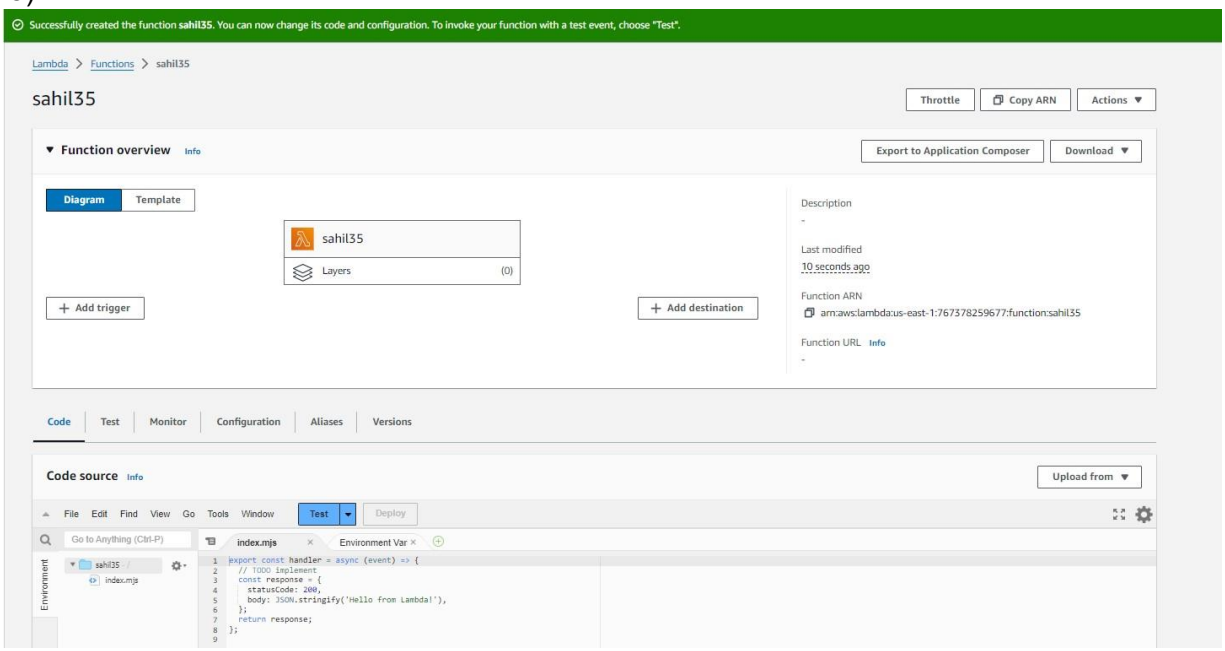


4) Once the function is created, click on the name of the function (myLambda27 in my case).

Name:Sahil Motiramani Div:D15C Roll no:35

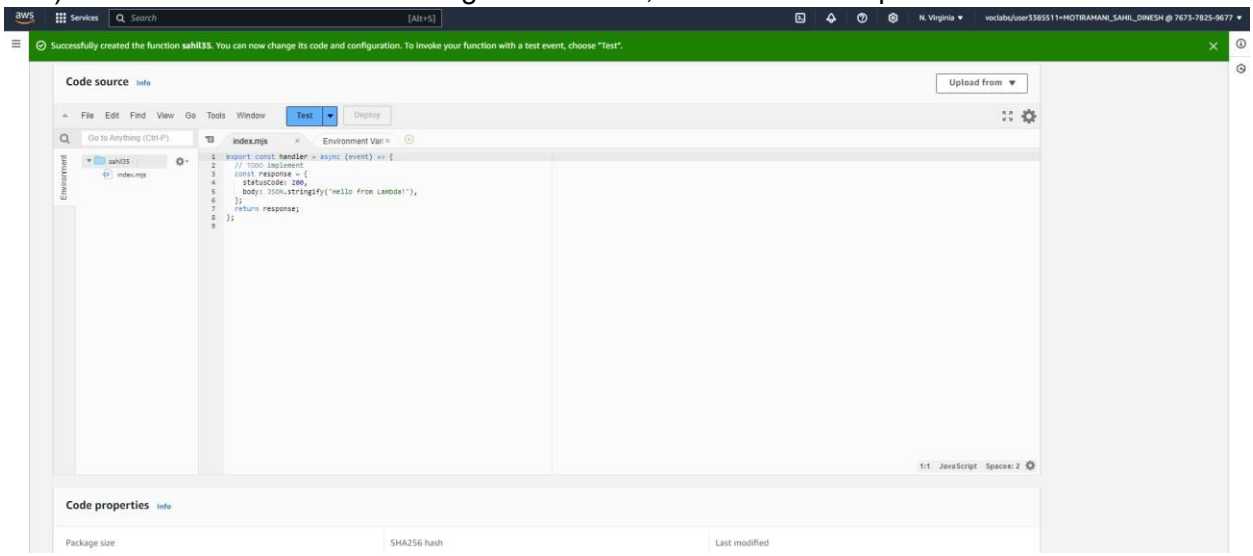


5) This is the dashboard of our lambda function.



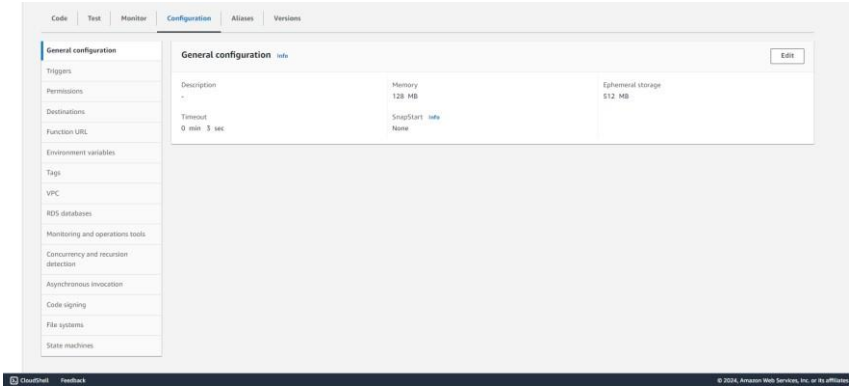
Name:Sahil Motiramani Div:D15C Roll no:35

6) This function has the following default code, which is used to print “Hello form Lambda!”.



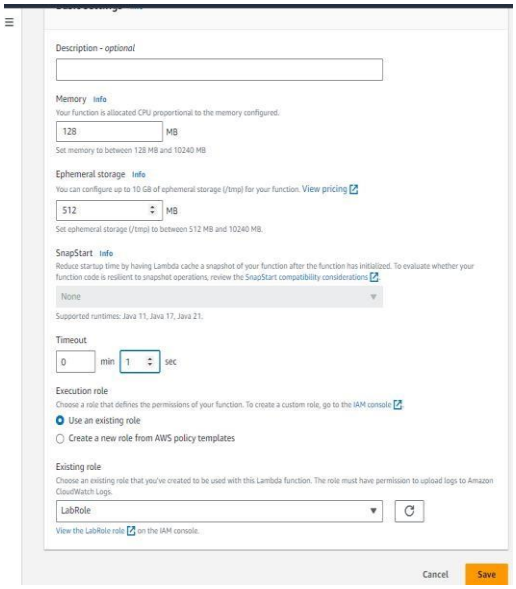
Step 3: Set up configurations and test events

1) Just above the test code, you would find Configuration, click on it. Then click on Edit.

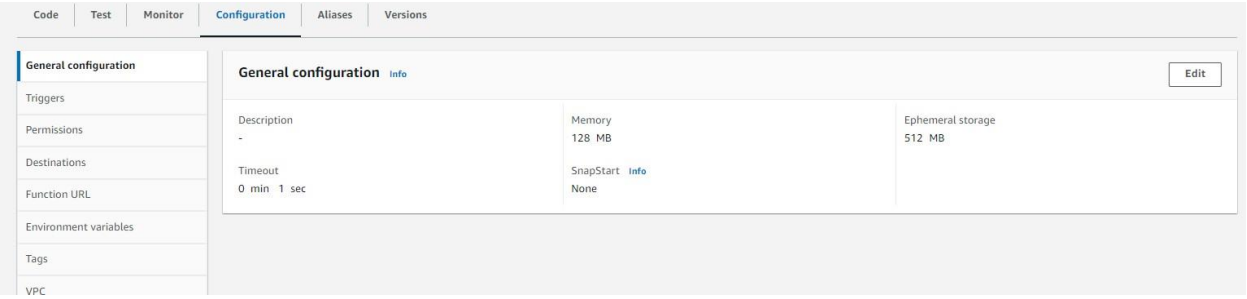


2) Here, change the Timeout to 1 sec. This is the time for which the function can be running before it is forcibly terminated.

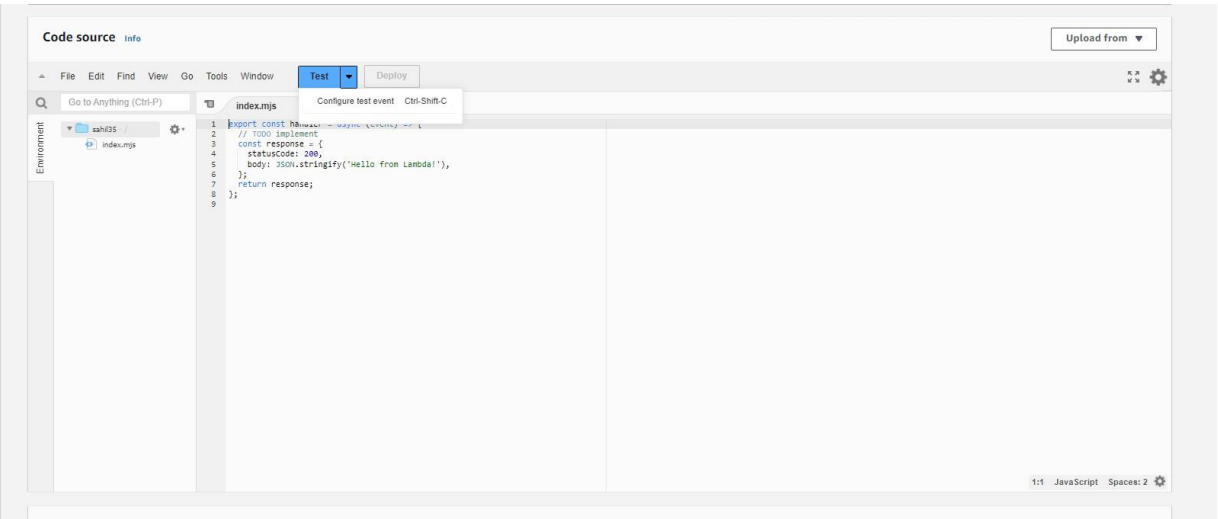
Name:Sahil Motiramani Div:D15C Roll no:35



3) We can see the executed changes.

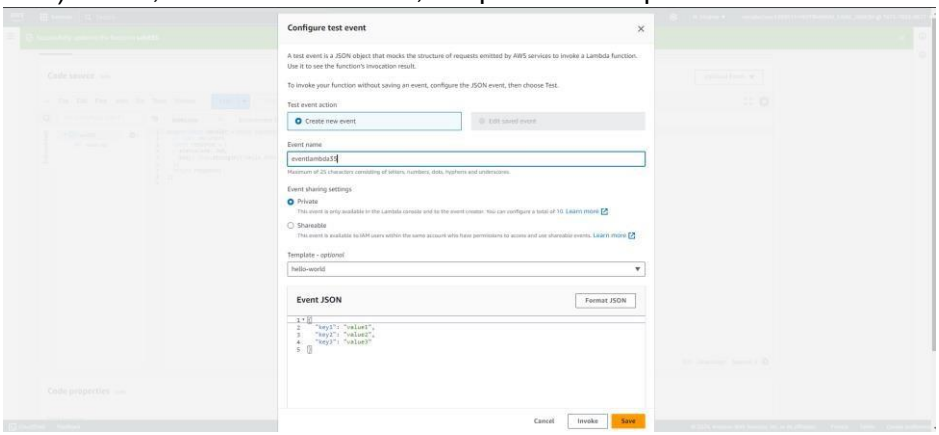


4) Switch back to the code tab. Click on the dropdown arrow near test. Then select configure test event.

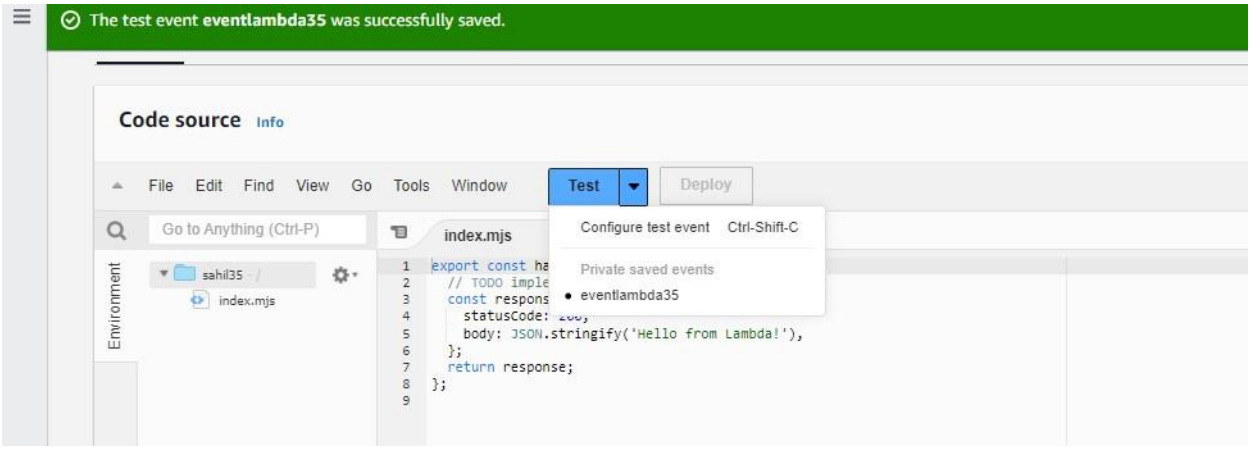


Name:Sahil Motiramani Div:D15C Roll no:35

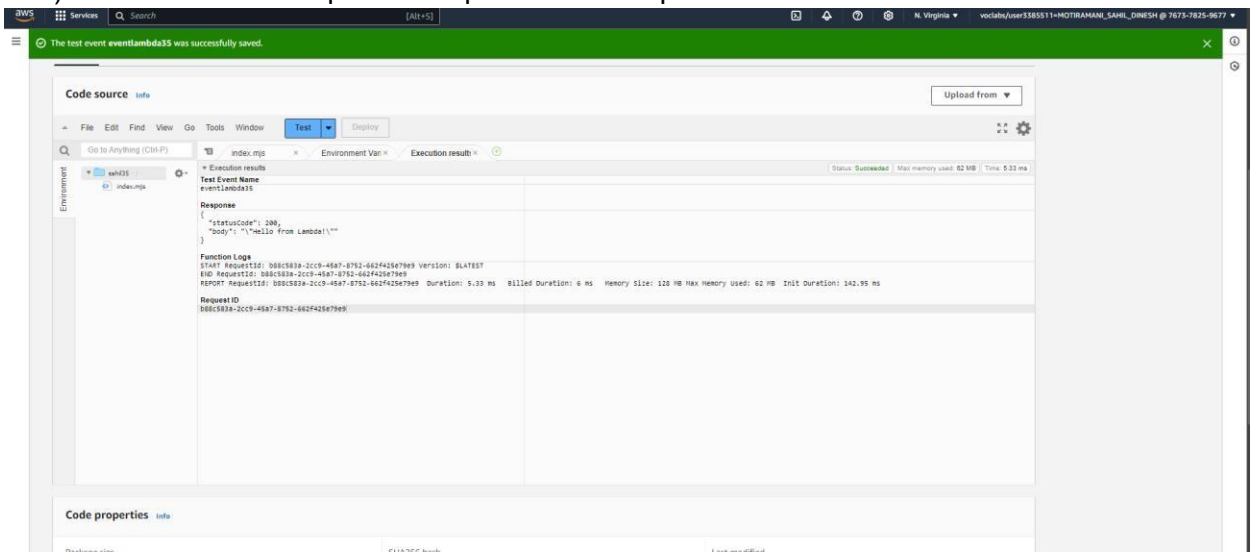
5) Here, create a new event, keep the other options default and save the event.



6) Now, again click on the dropdown. This time, select the event you have created. Then, click on TEST.



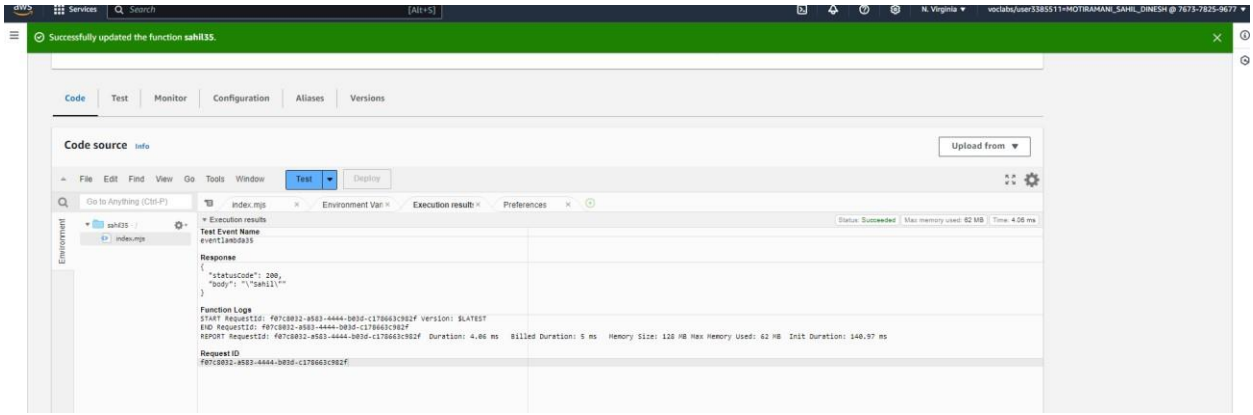
7) We can see the expected output for the sample code.



Name:Sahil Motiramani Div:D15C Roll no:35

8) For a test, declare a string and call it in line 6. After making the changes click on deplpoy.

9) Run the test. We can see that the string we declared has come in the output.



Conclusion:

In this experiment, we effectively investigated the AWS Lambda service by developing and configuring Lambda functions using Python, Java, or Node.js. We discovered how to establish a Lambda function, tweak its settings (like modifying the timeout duration), and evaluate the function with personalized events. Throughout this experience, we noted how Lambda manages executions, including timeout handling and producing expected results according to modifications in the code.

Name:Sahil Motiramani Div:D15C Roll No:35

Exp:12

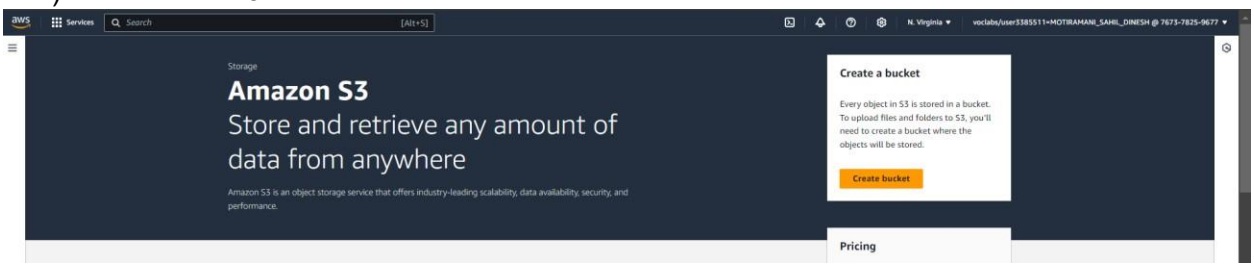
Aim: To create a Lambda function which will log “An Image has been added” once you add an object to a specific bucket in S3.

Prerequisites:

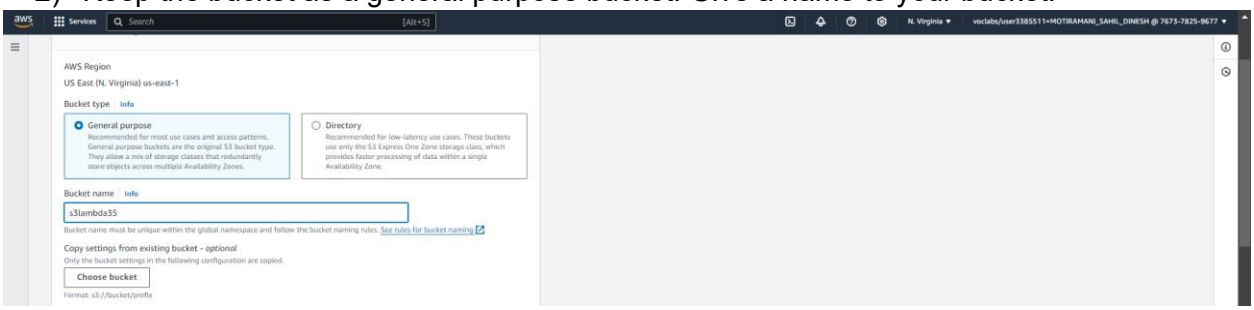
1) AWS account (acacdemy preferable)
2) Lambda function (created in the previous experiment).

Step 1: Create a s3 bucket.

1) Search for S3 bucket in the services search. Then click on create bucket.

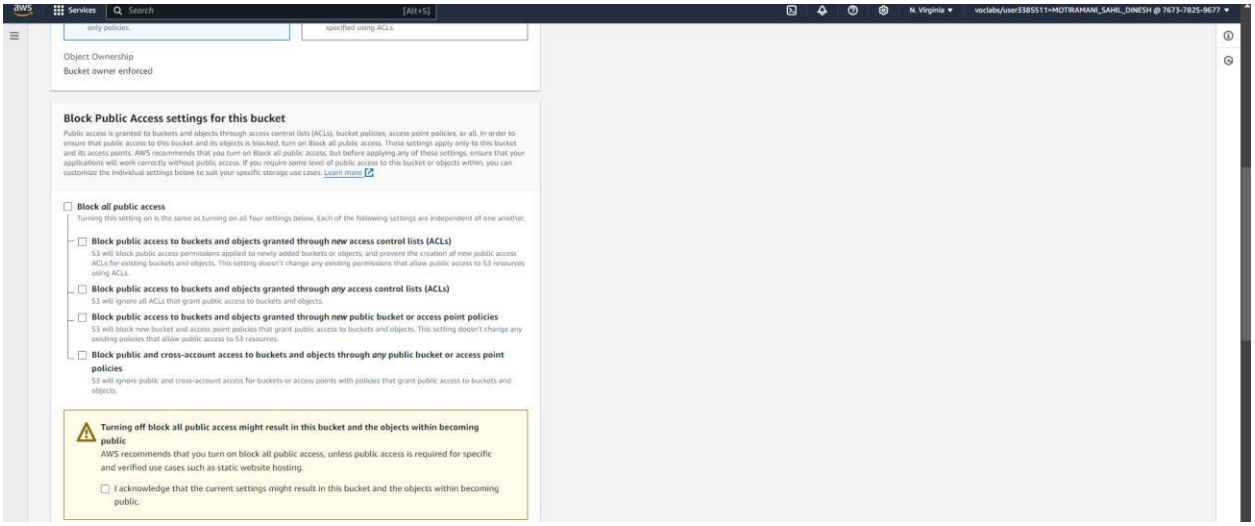


2) Keep the bucket as a general purpose bucket. Give a name to your bucket.

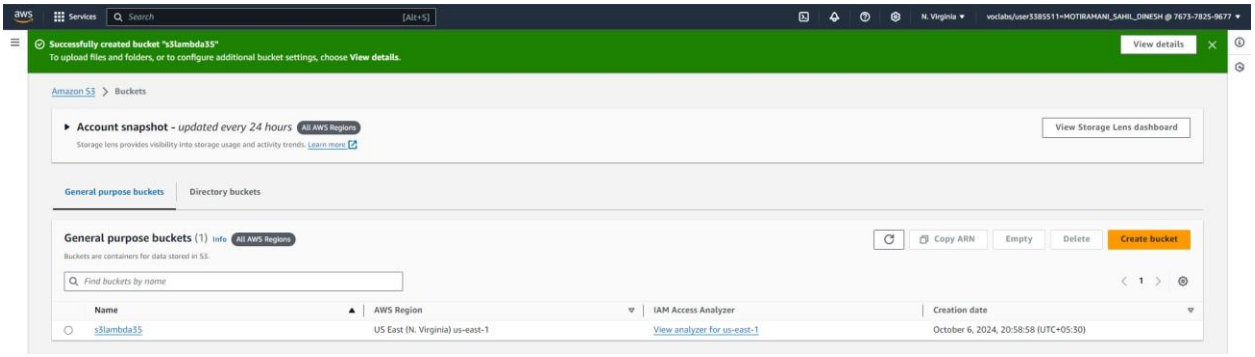


Name:Sahil Motiramani Div:D15C Roll No:35

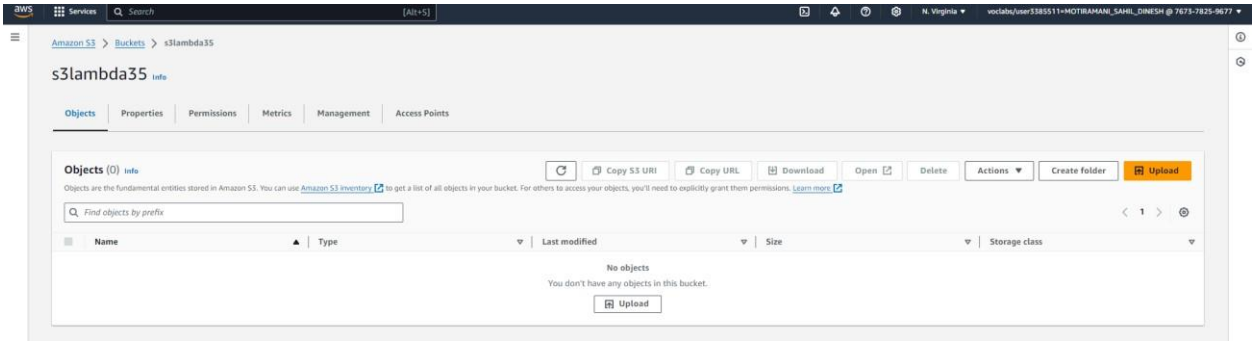
3) Uncheck block all public access.



4) Keeping all other options same, click on create. This would create your bucket. Now click on the name of the bucket.

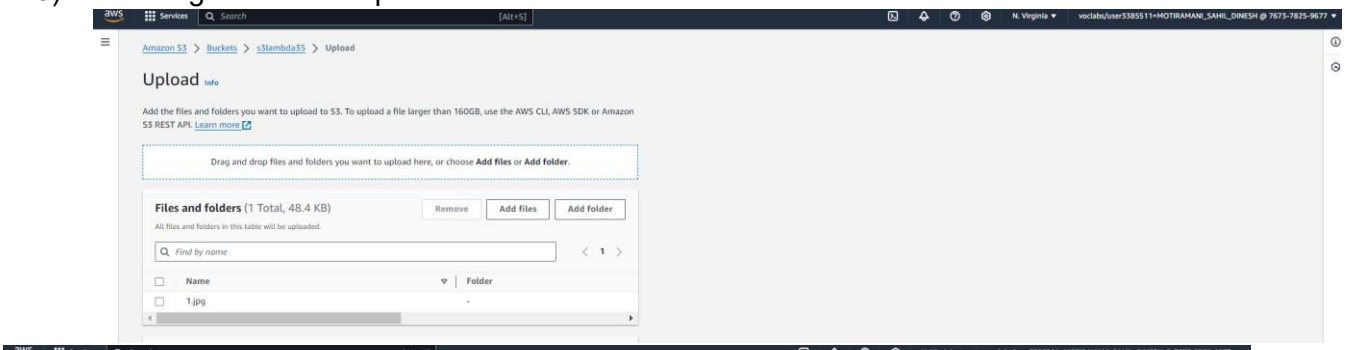


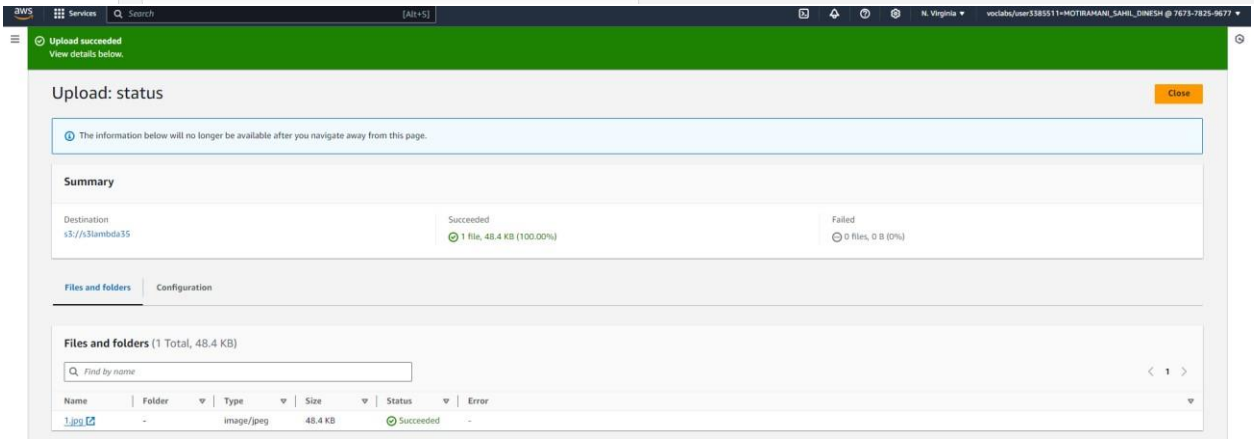
5) Here, click on upload, then add files. Select any image that you want to upload in the bucket and click on upload.



Name:Sahil Motiramani Div:D15C Roll No:35

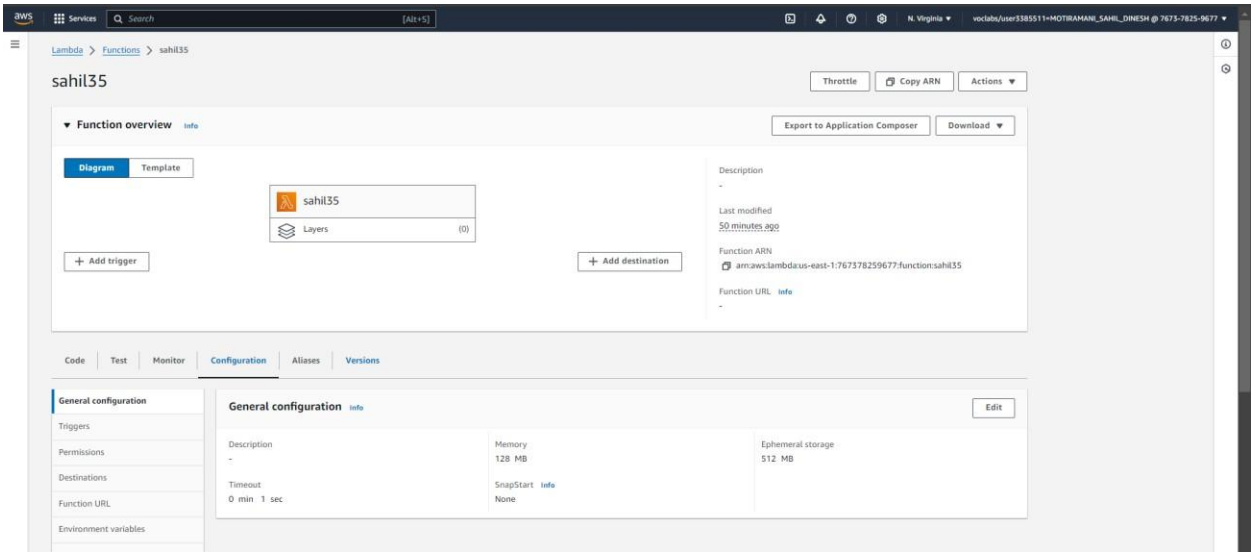
6) The image has been uploaded to the bucket.





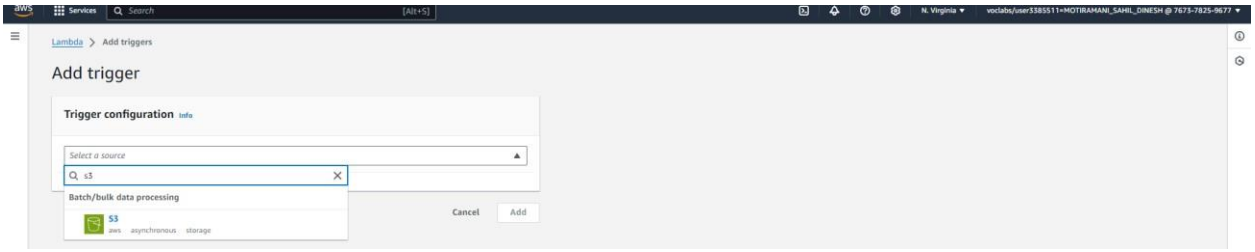
Step 2: Configure Lambda function

1) Go to the lambda function you had created berfor. (Services → Lambda → Click on name of function). Here, click on add trigger.

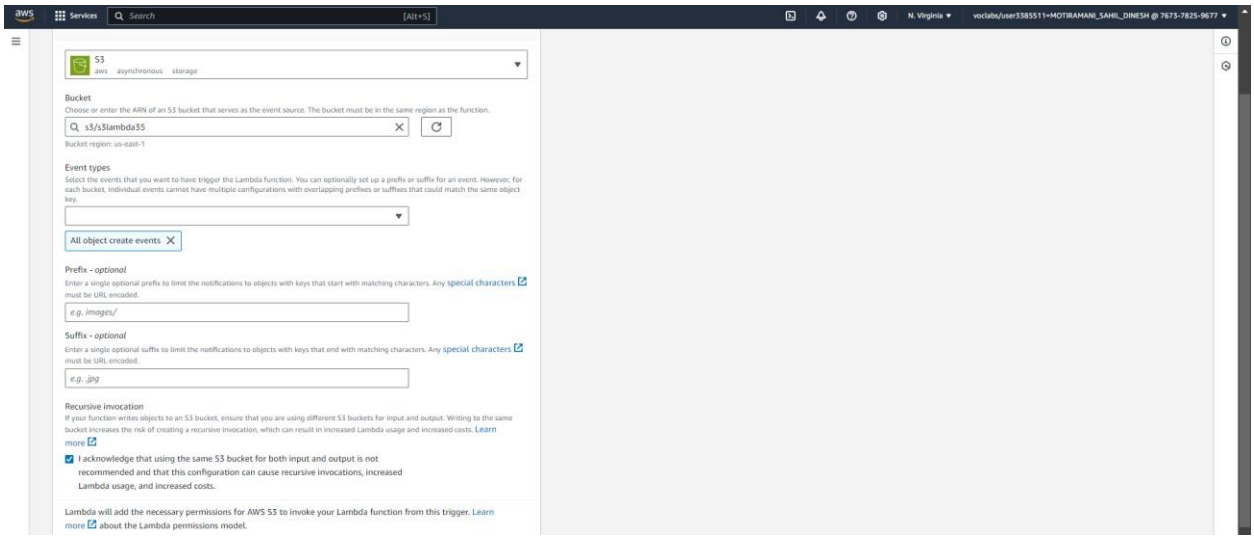


Name:Sahil Motiramani Div:D15C Roll No:35

2) Under trigger configuration, search for S3 and select it.



3) Here, select teh S3 bucket you created for this experiment. Acknowledge the condition given by AWS. then click on Add. This will add the S3 bucket trigger to your function.

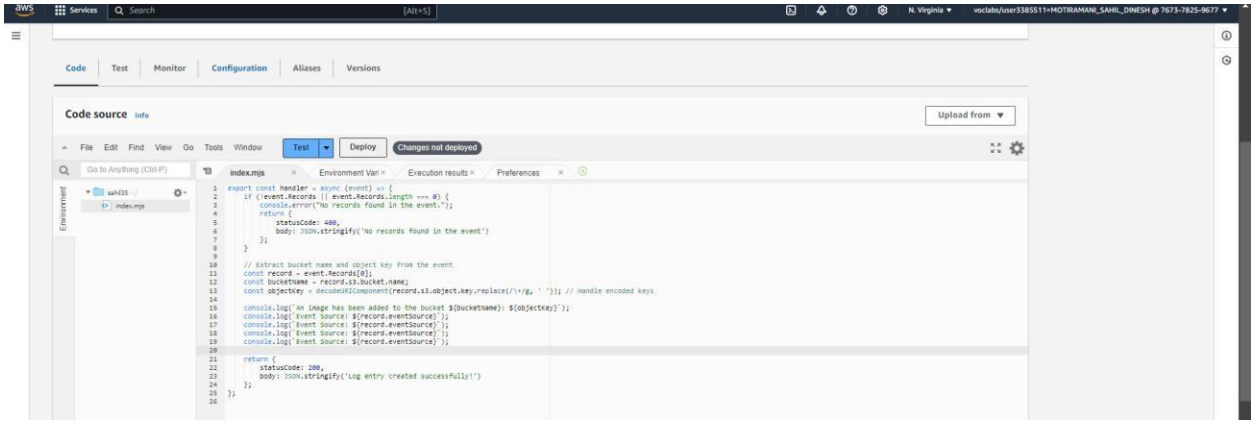


4. Scroll down to the code section of the function. Add the following javascript code to the code area by replacingthe existing code

```
export const handler = async (event) => { if
  (!event.Records || event.Records.length === 0) {
  console.error("No records found in the event.");
     return { statusCode: 400, body: JSON.stringify('No records
        found in the event')
     };
```

```
    }

    // Extract bucket name and object key from the event const record = event.Records[0]; const
    bucketName = record.s3.bucket.name; const objectKey =
    decodeURIComponent(record.s3.object.key.replace(/\+/g, ' ')); // Handle
encoded keys

    console.log(`An image has been added to the bucket ${bucketName}: ${objectKey}`);
    console.log(`Event Source: ${record.eventSource}`); console.log(`Event Source:
    ${record.eventSource}`); console.log(`Event Source: ${record.eventSource}`);
    console.log(`Event Source: ${record.eventSource}`); return {
        statusCode: 200, body: JSON.stringify('Log entry
        created successfully!')
    };
};
```

This code checks for records in the event, extracts the bucket name and object key, logs the details, and returns a success message if an image is added to the bucket.



Now, click on the dropdown near test, then click on configure test event.

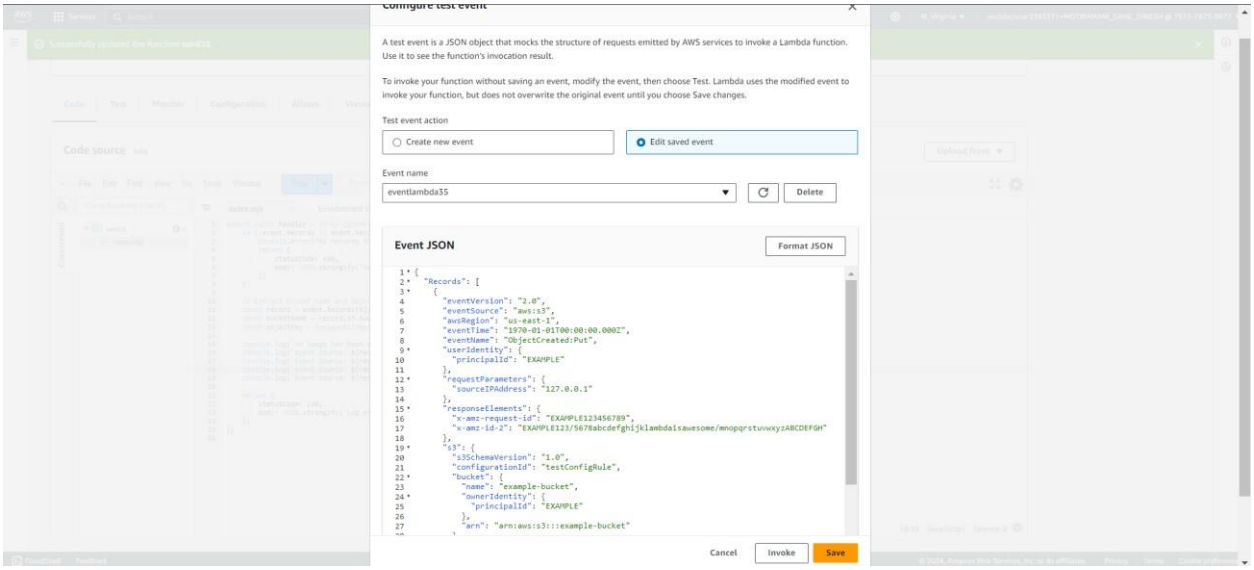
Name:Sahil Motiramani Div :D15C Roll no:35

6) Here, select edit saved event. Select the event taht you had created before. Under Event JSON, paste the following code.


```
{
  "Records": [
    {
      "eventVersion": "2.0",
      "eventSource": "aws:s3",
      "awsRegion": "us-east-1",
      "eventTime": "1970-01-01T00:00:00.000Z",
      "eventName": "ObjectCreated:Put",
      "userIdentity": {
        "principalId": "EXAMPLE"
      },
      "requestParameters": {
        "sourceIPAddress": "127.0.0.1"
      },
      "responseElements": {
        "x-amz-request-id": "EXAMPLE123456789", "x-amz-id-
        2":
"EXAMPLE123/5678abcdefghijklambdaisawesome/mnopqrstuvwxyzABCDEFGH"
      },
      "s3": {
        "s3SchemaVersion": "1.0",
        "configurationId": "testConfigRule",

"bucket": {
          "name": "example-bucket", "ownerIdentity":
          {
            "principalId": "EXAMPLE"
          },
          "arn": "arn:aws:s3:::example-bucket"
```

```
      },
      "object": {
        "key": "test%2Fkey",
        "size": 1024,
        "eTag": "0123456789abcdef0123456789abcdef",
        "sequencer": "0A1B2C3D4E5F678901"
      }
    }
  }
 ]
}
```


This JSON structure represents an S3 event notification triggered when an object is uploaded to an S3 bucket. It contains details about the event, including the bucket name (example-bucket), the object key (test/key), and metadata like the object's size, the event source (aws:s3), and the event time.

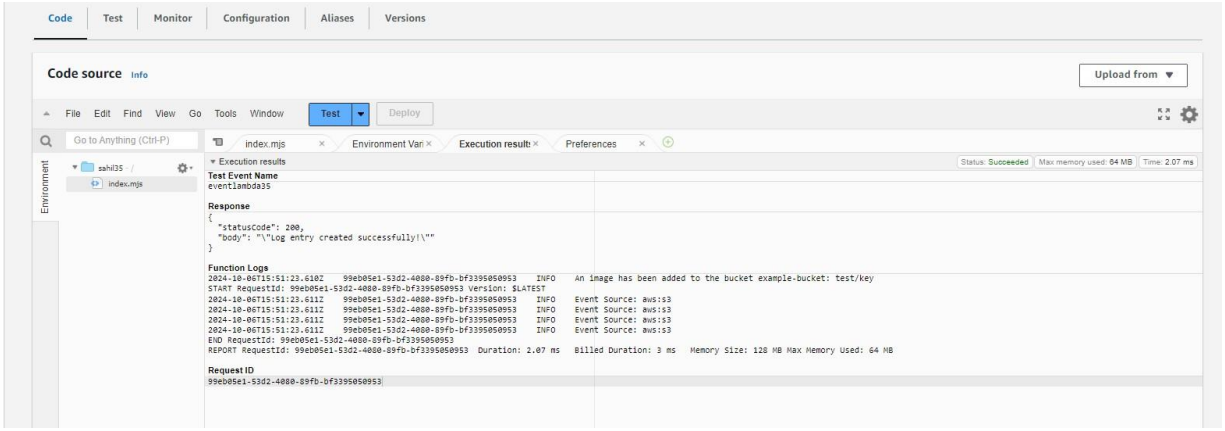


Save the changes. Then deploy the code changes by clicking on deploy.

Name:Sahil Motiramani Div :D15C Roll no:35

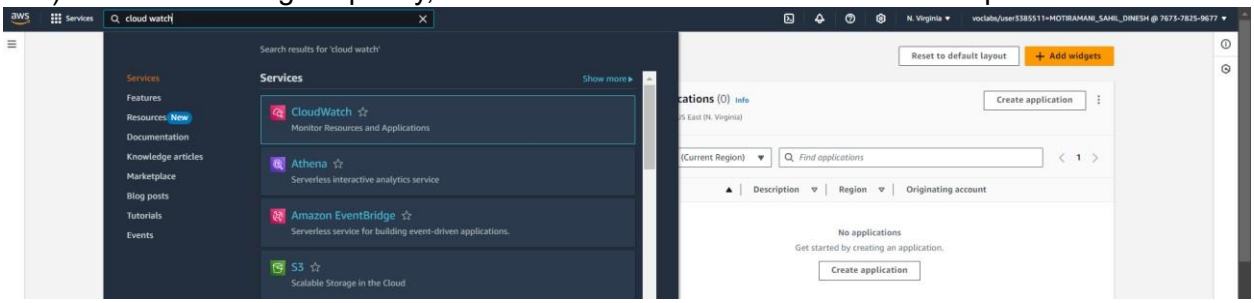
7) After deploying, click on Test. The console output shows that ‘an image has been added to the bucket’

The JSON response shows that the log entry was created successfully.

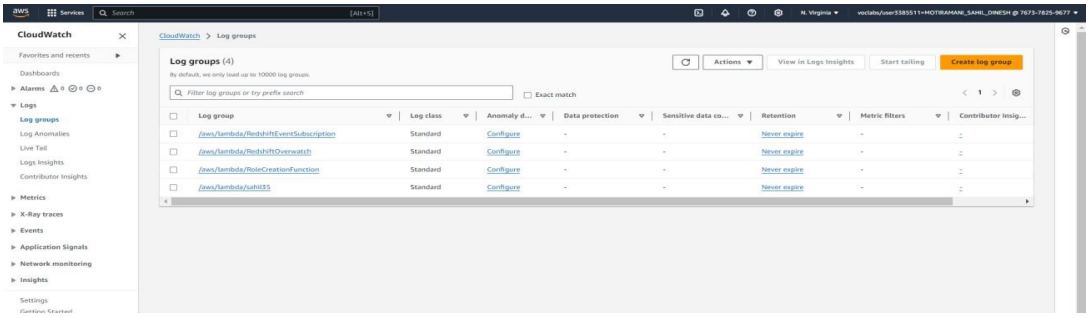


Step 3: Check the logs

1) To check the logs explicitly, search foe CloudWatch on services and open it in a new tab.

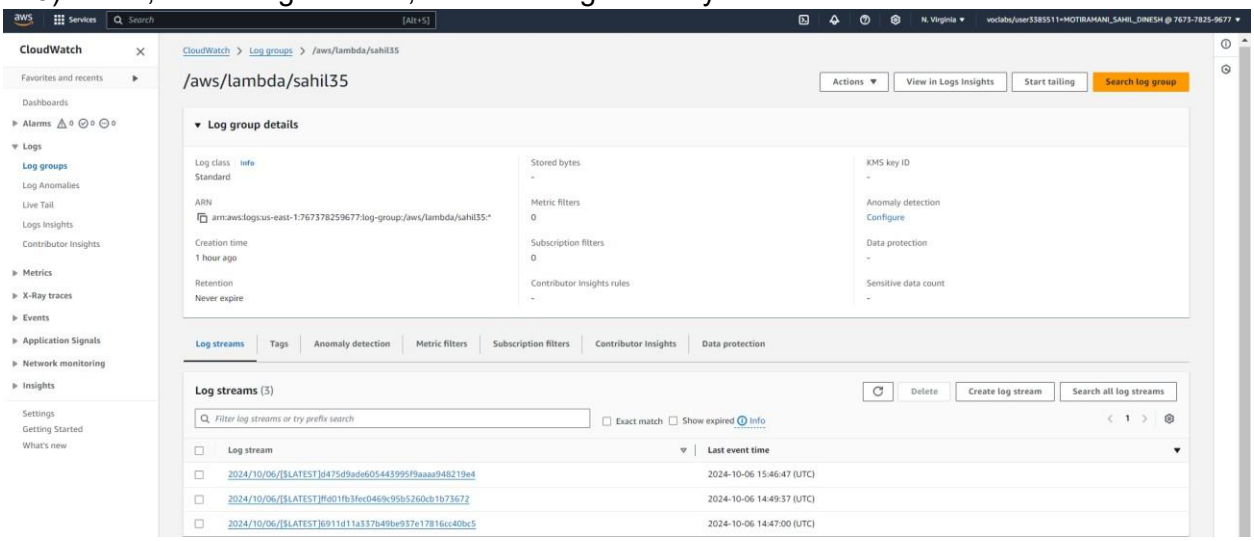


2) Here, Click on Logs → Log Groups. Select the log that has the lambda function name you just ran.

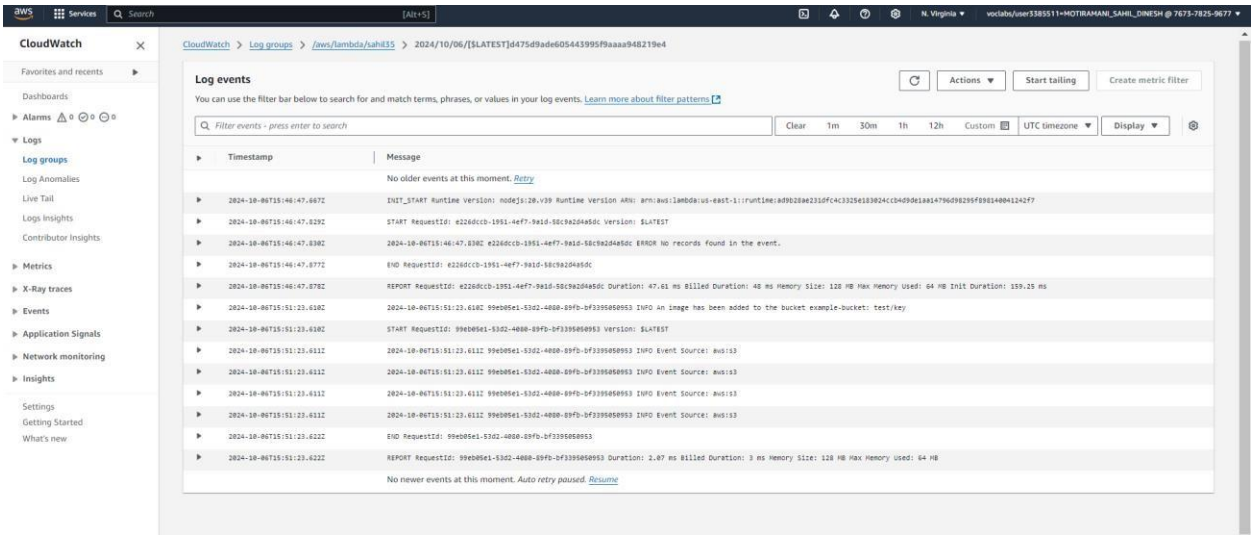


Name:Sahil Motiramani Div :D15C Roll no:35

3) Here, under Log streams, select the log stream you want to check.



4) Here again, we can see that ‘An image has been added to the bucket’.



Conclusion:

In this experiment, we developed and deployed a Lambda function designed to respond to file uploads in an S3 bucket. The function was triggered automatically whenever a new object was added to the bucket, illustrating how AWS services can efficiently automate workflows. The Lambda function extracted and logged key details from the event, such as the bucket's name and the object's key. We tested this by uploading a sample file, and upon reviewing the logs in CloudWatch, we confirmed that the function executed successfully, capturing the upload event.