Name:Aditya Dubey Div:D15C Roll no:10

STATIC HOSTING :

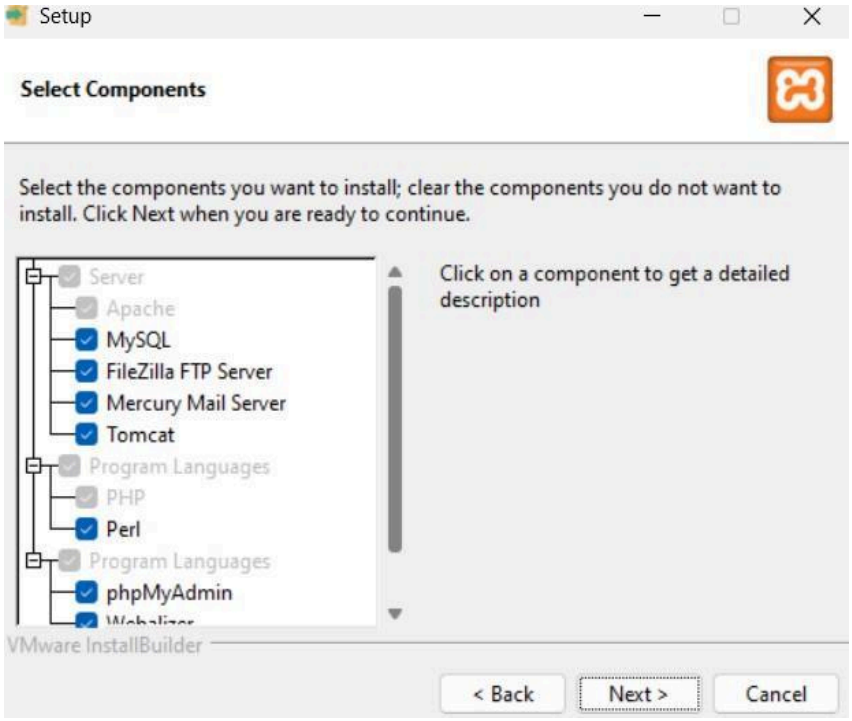
1) On local server (XAMPP)

Step 1: Install XAMPP from https://www.apachefriends.org/

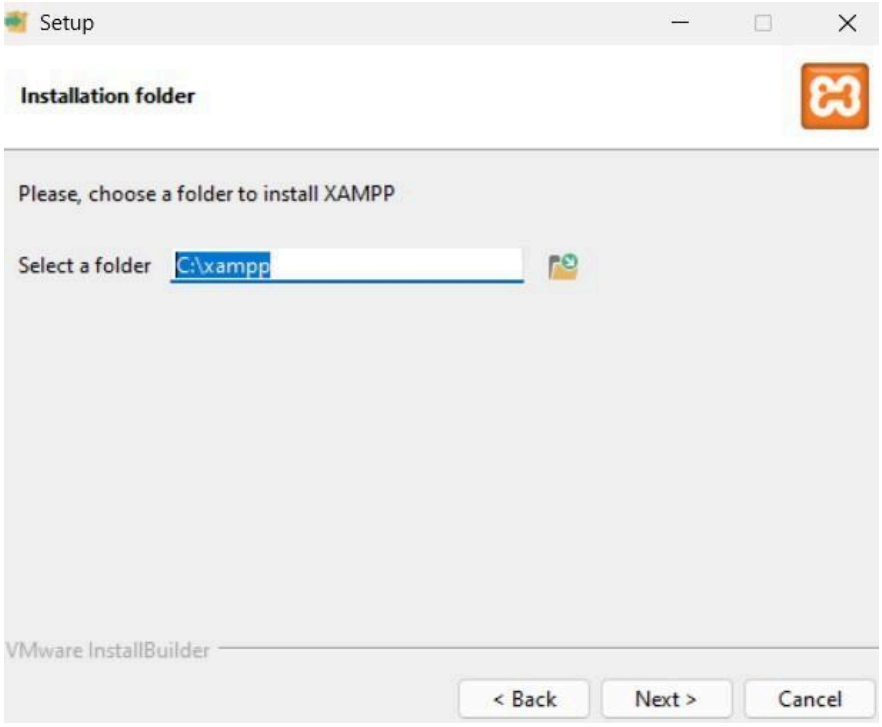
a) Download xampp

b) Open the setup file. Click on Next

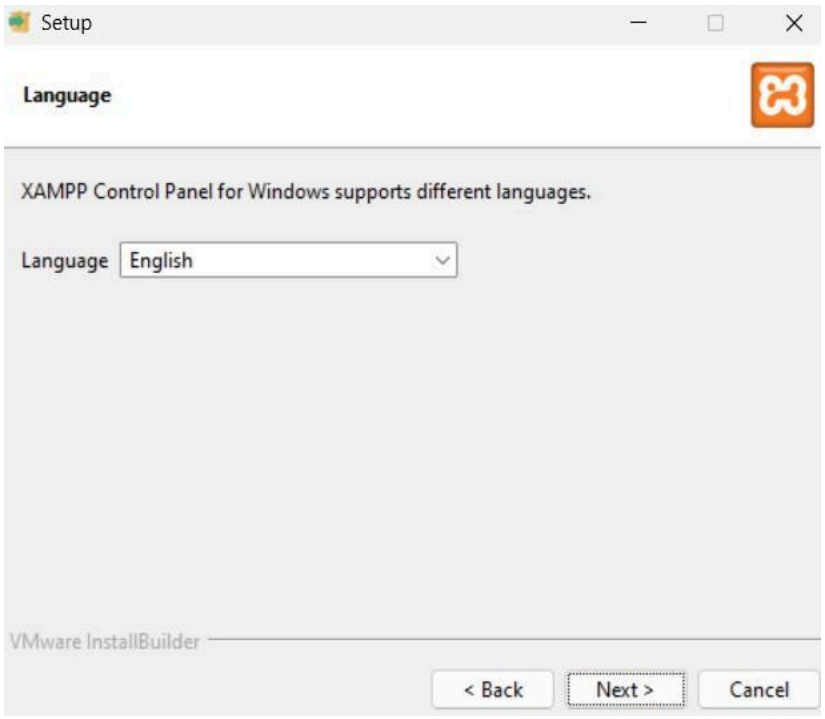
c) Select all the required components and click next



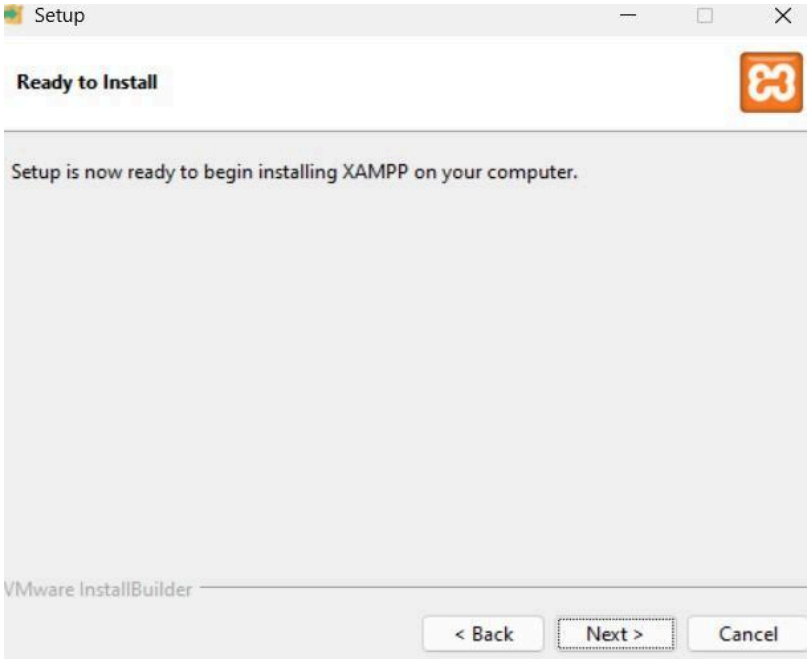
d) Choose the folder to install XAMPP in. Make sure the folder is empty. Click next



e) Select the language, click next. XAMPP starts to install

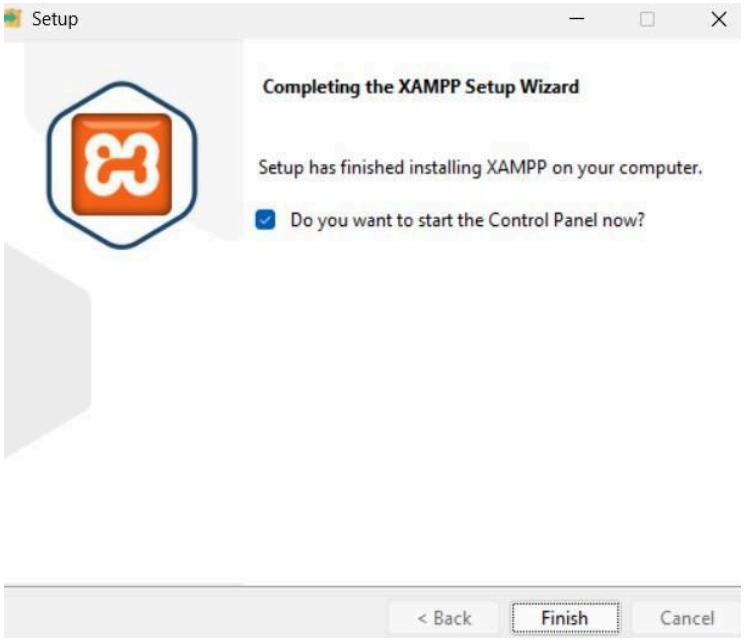


f) Click on Next

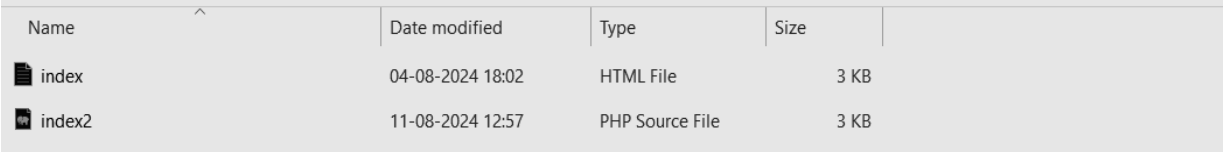


g) Wait until unpacking of all files is done

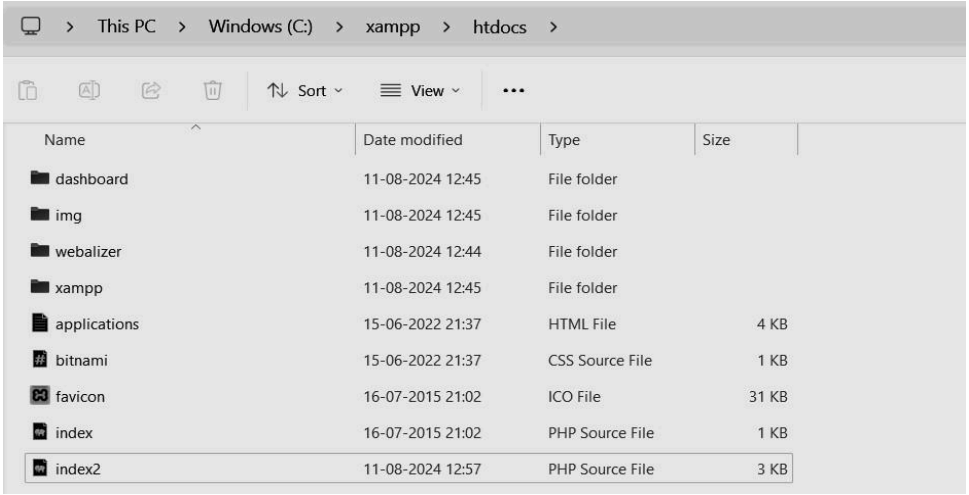
h) The installation is complete. Click Finish



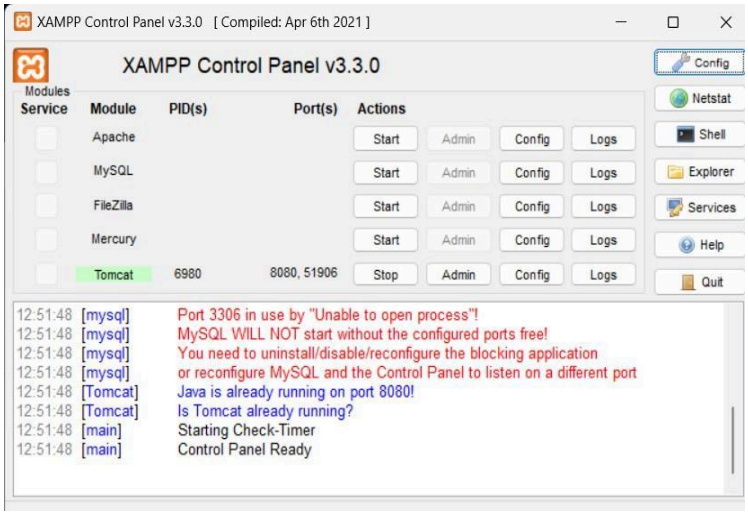
Step 2: Setup a file that is to be hosted on the server. Make sure the file has extension .php



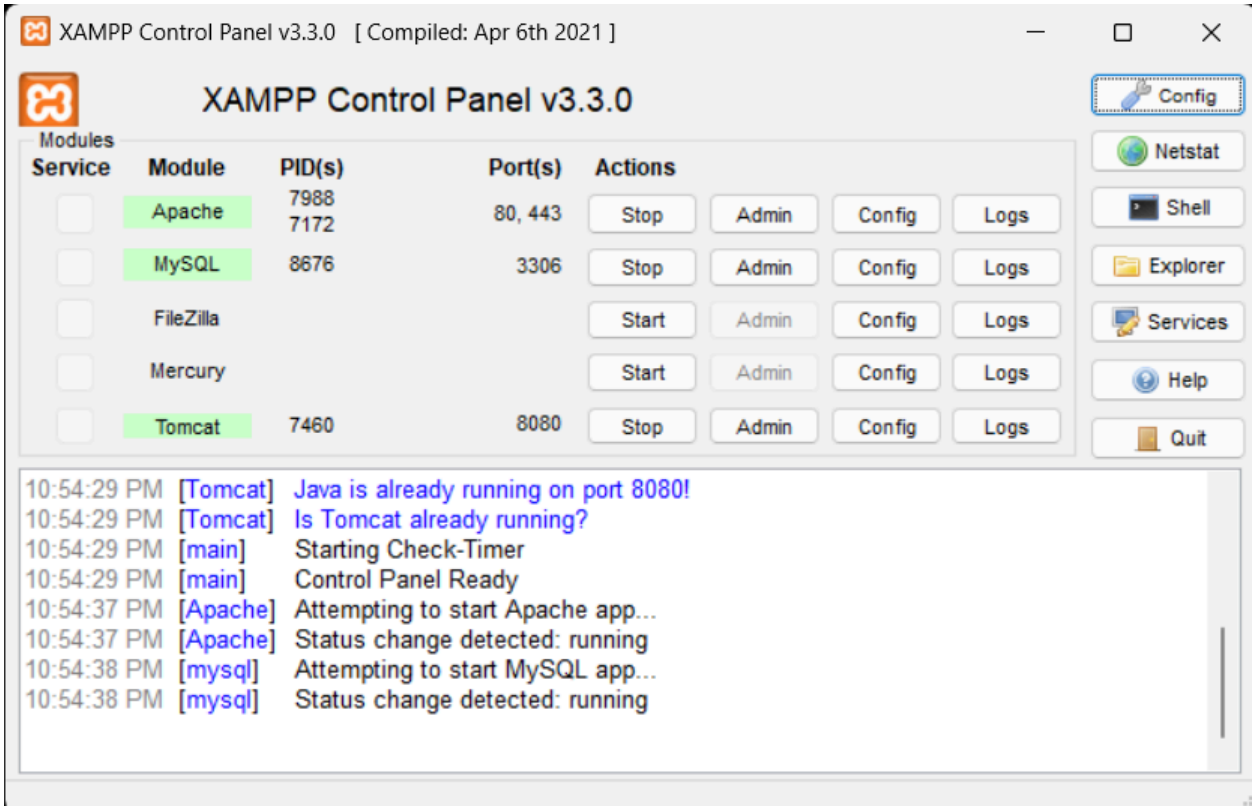
Step 3: Go to the directory where XAMPP was installed. Go to htdocs folder. Place your folder in this directory i.e Paste the index2.php here



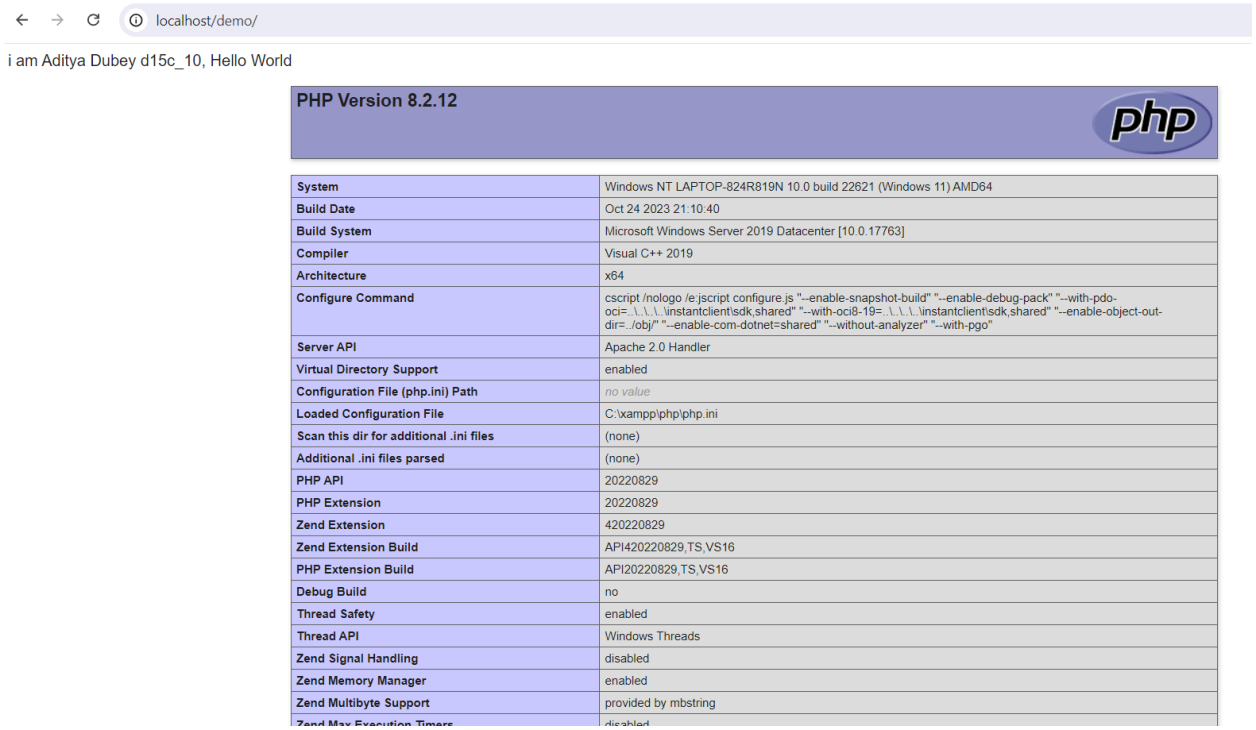
Step 4: Open XAMPP Control Panel, start the Apache service (Required) and mySQL service (if needed)



Name:Aditya Dubey Div:D15C Roll no:10



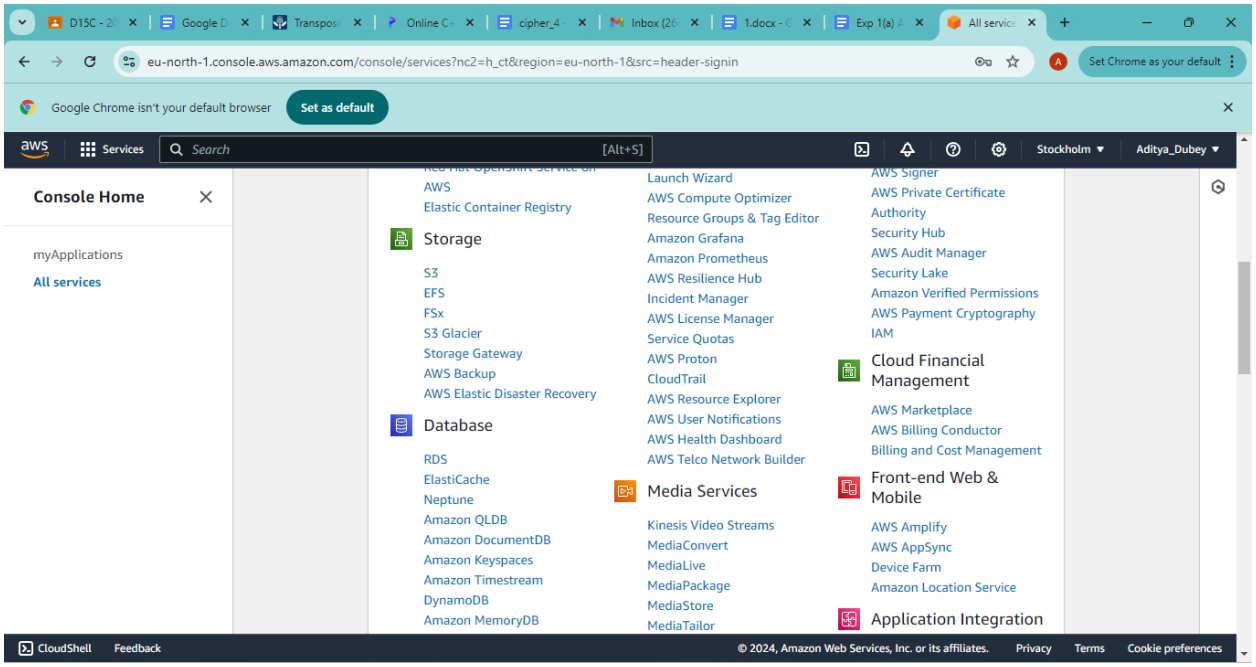
Step 5: Open your web browser. Type localhost/demo. This will open your website on your browser.



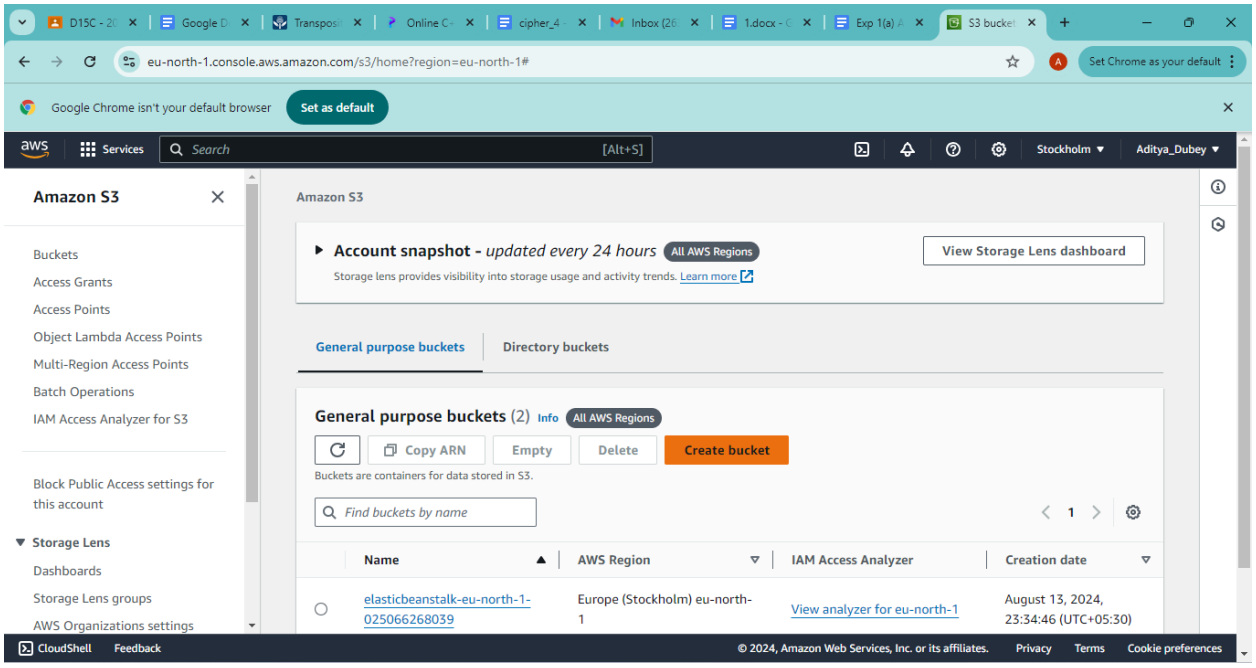
Name:Aditya Dubey Div:D15C Roll no:10

1) On AWS S3

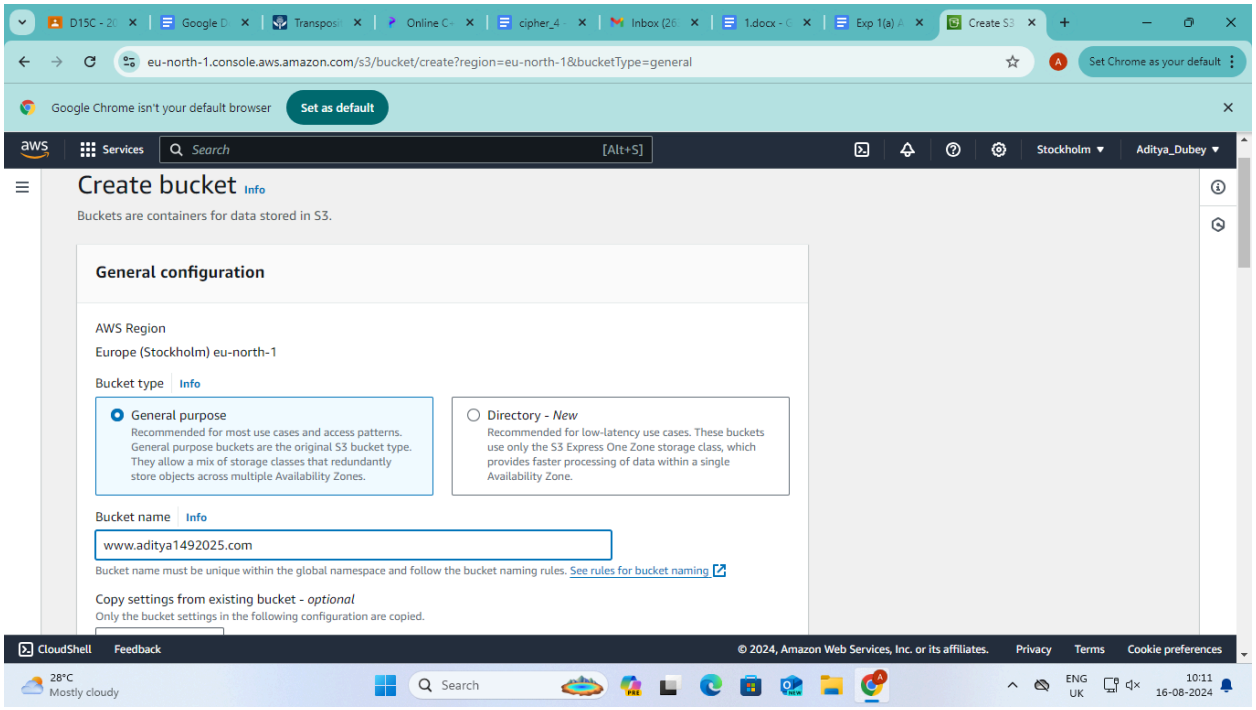
Step 1: Login to your AWS account. Go to services and open S3.

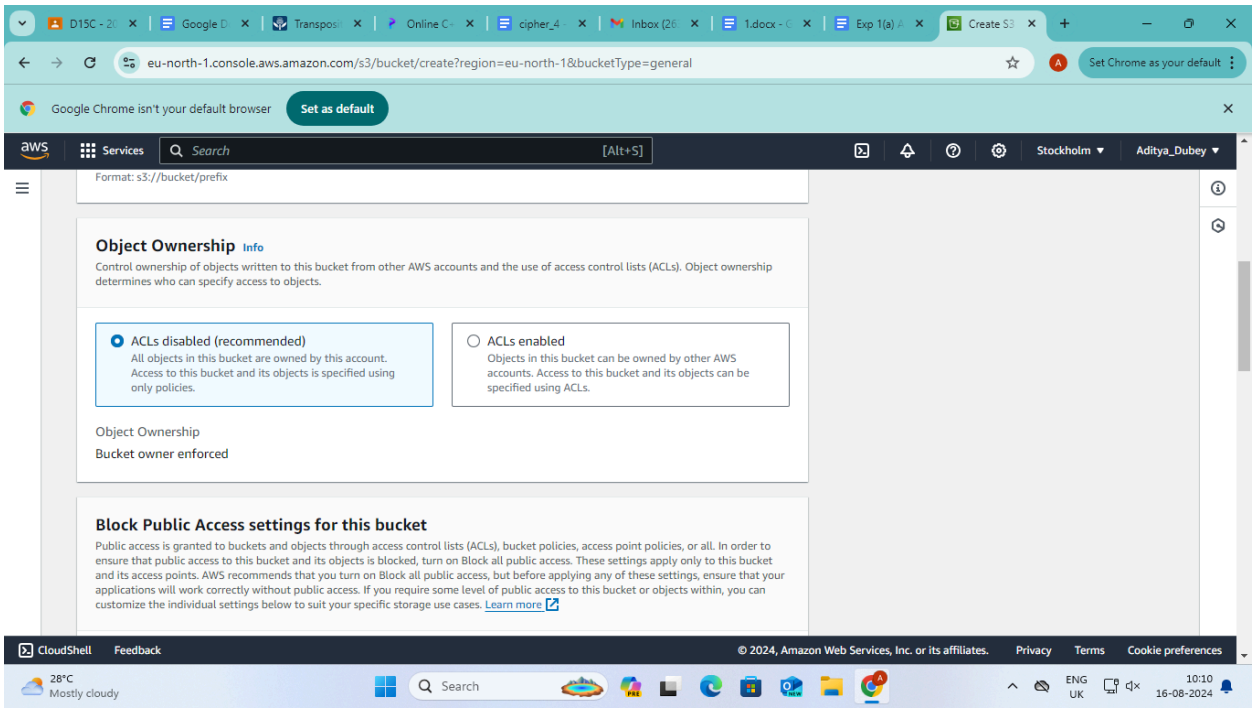


Step 2: Click on Create Bucket



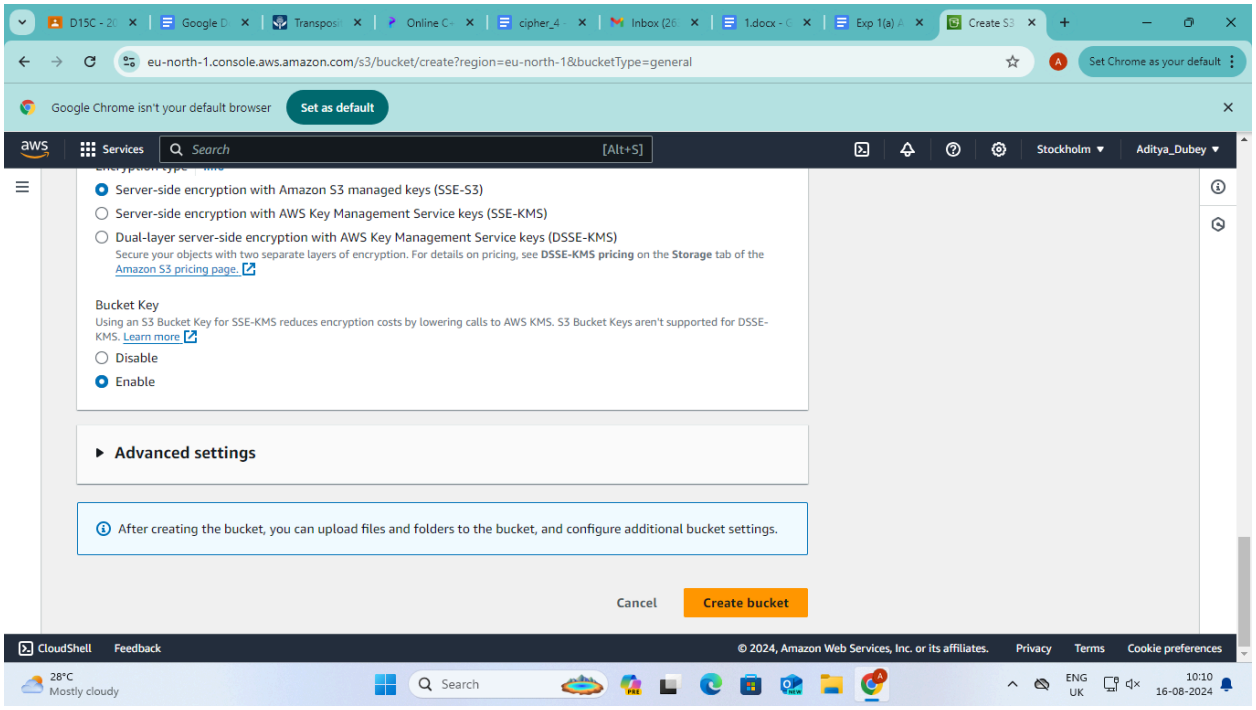
Step 3: Give a name to your bucket, keeping other options default, scroll down and click on Create Bucket

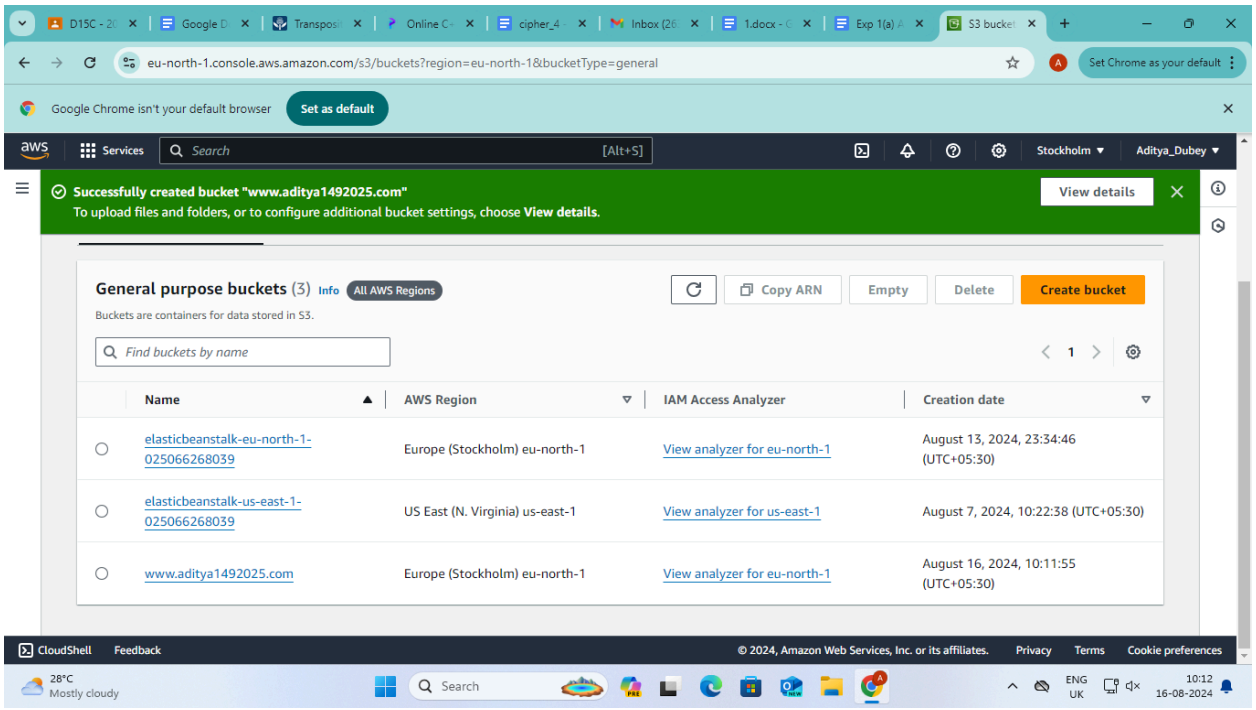




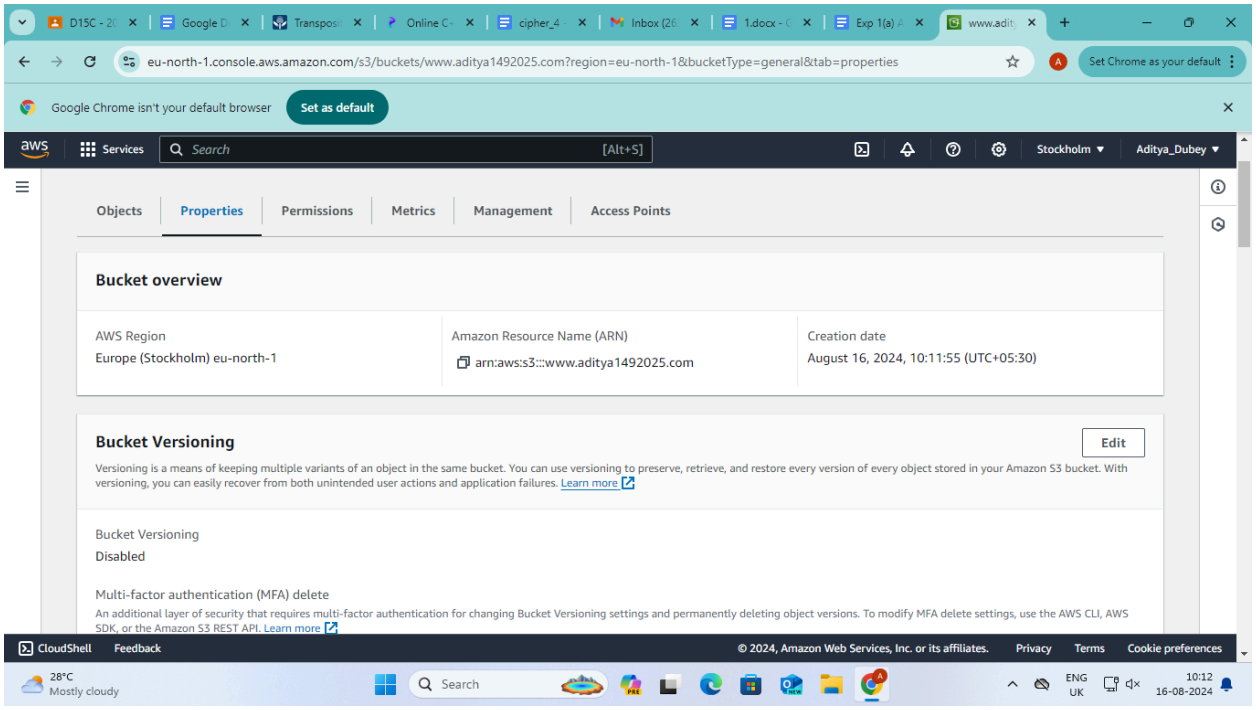
Step 4: Click on the name of your bucket and goto Properties

Name:Aditya Dubey Div:D15C Roll no:10

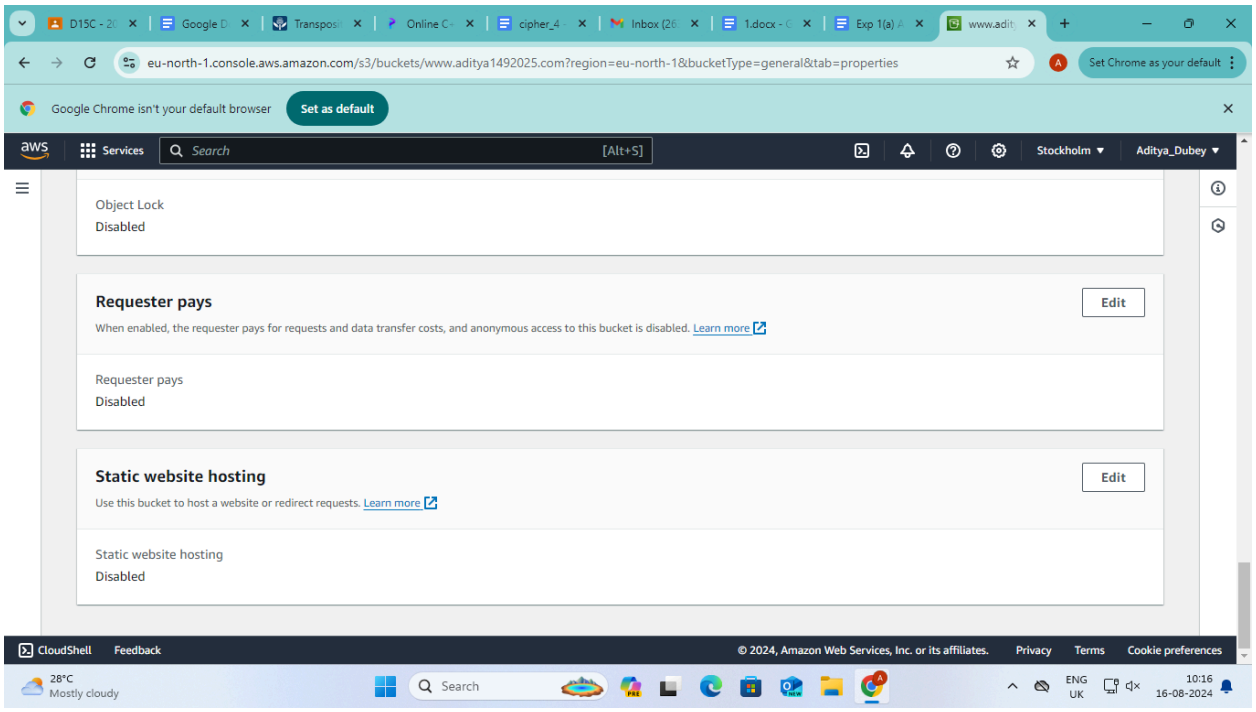




Name:Aditya Dubey Div:D15C Roll no:10

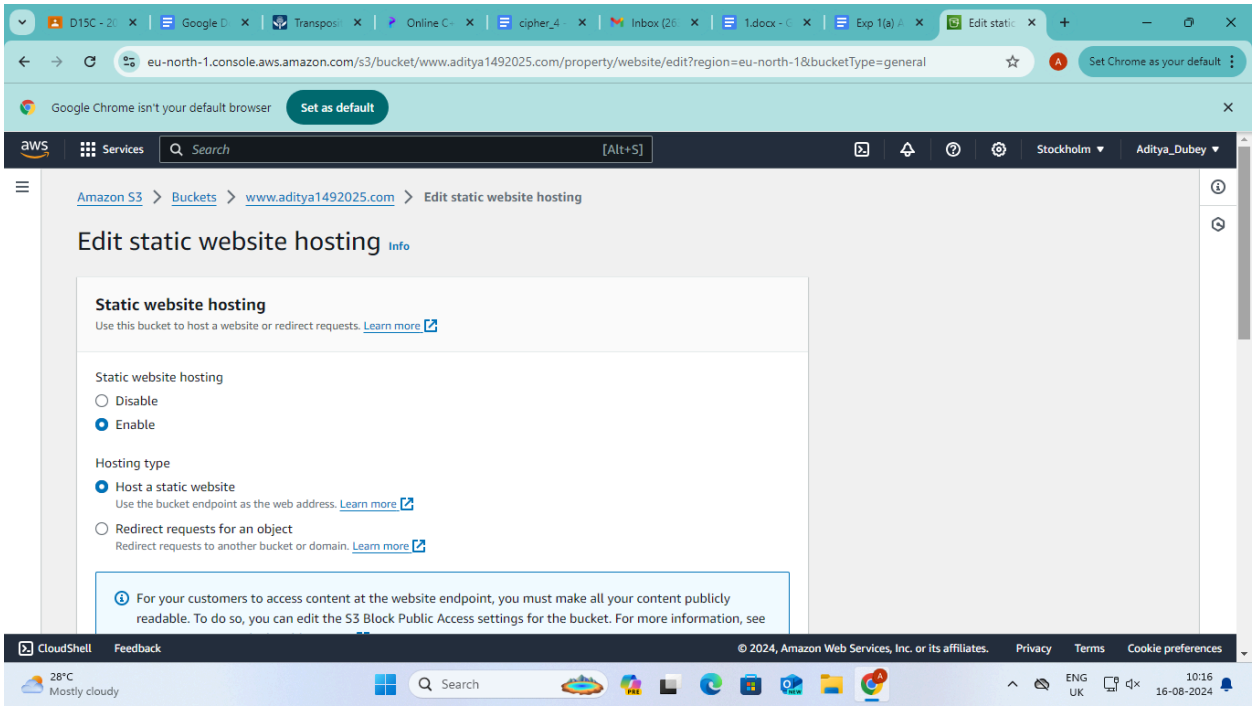


Step 5: Scroll down till you find Static website hosting, click on edit

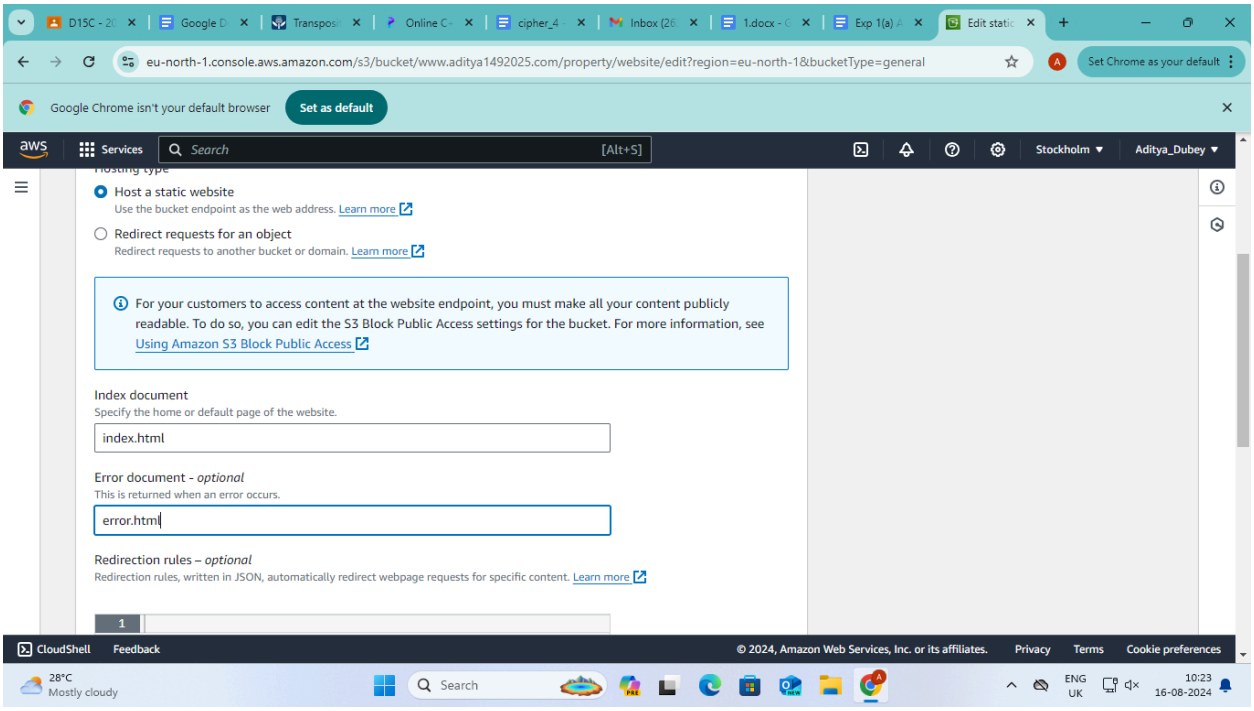


Step 6: Click on Enable static website hosting

Name:Aditya Dubey Div:D15C Roll no:10

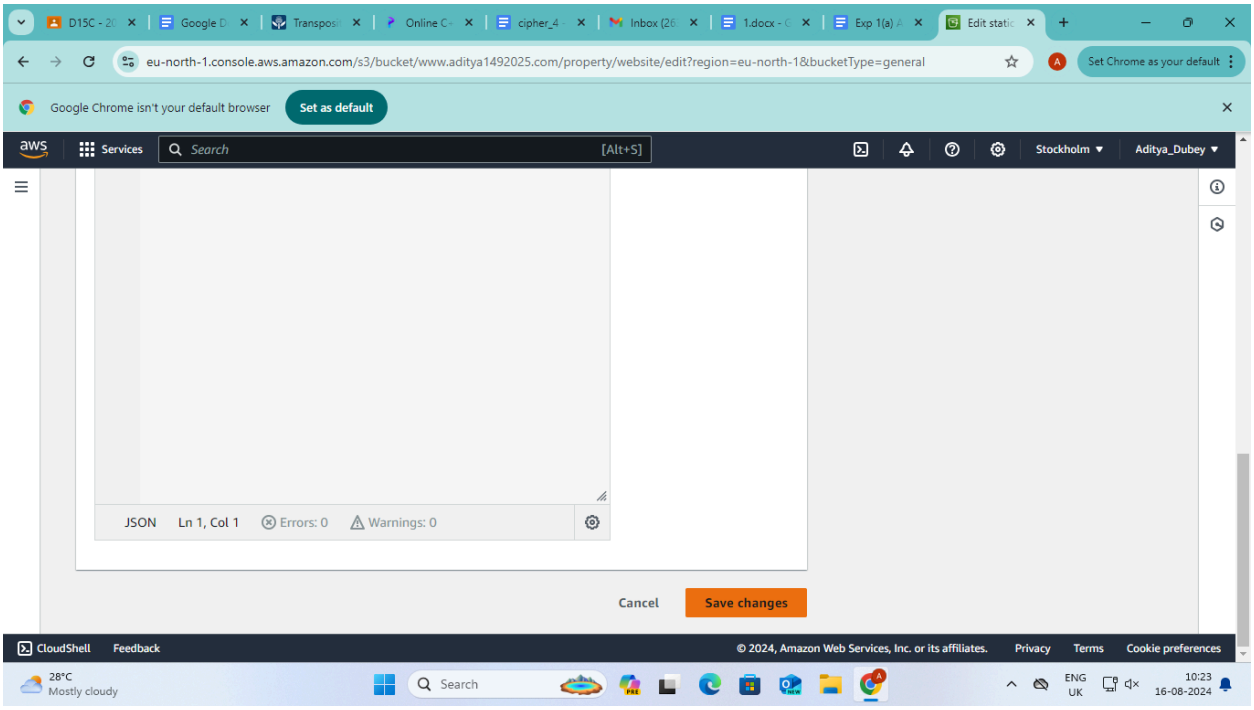


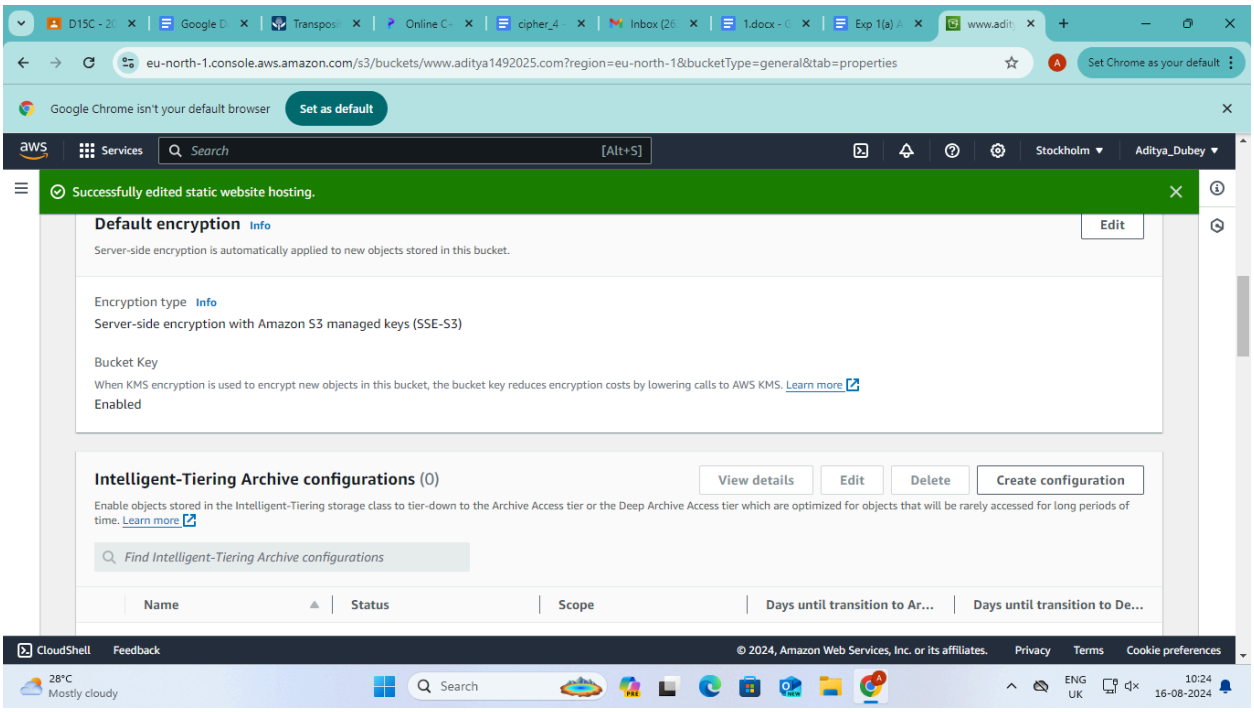
Step 7: Write the name of your document which you wanted to host on AWS from your local folder and in error document, give name as error..html. Save your changes.



Step 8: Save the Changes

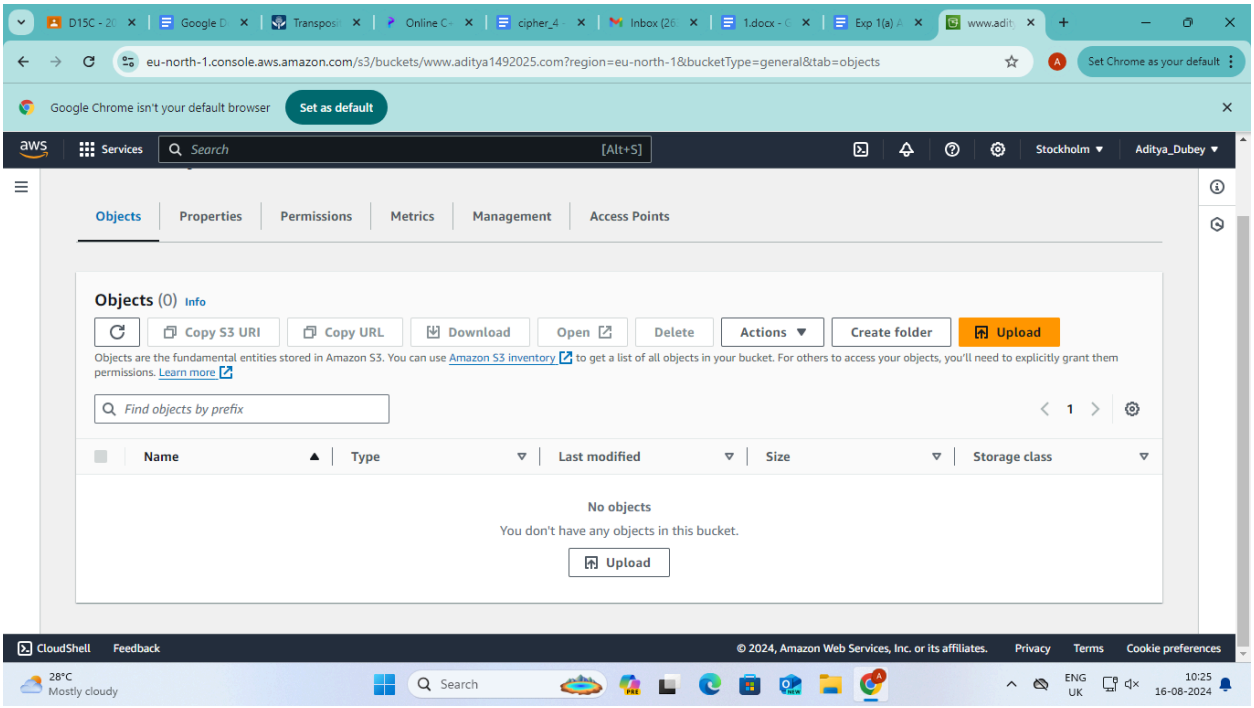
Name:Aditya Dubey Div:D15C Roll no:10



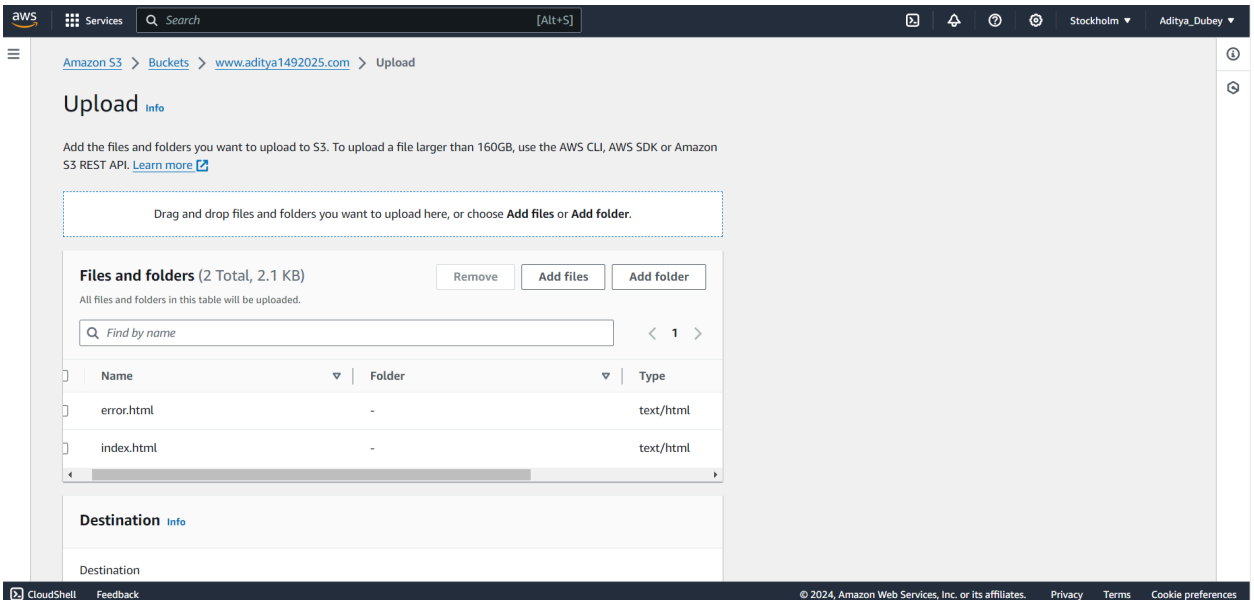


Step 9: Go to Objects tab and click on upload file

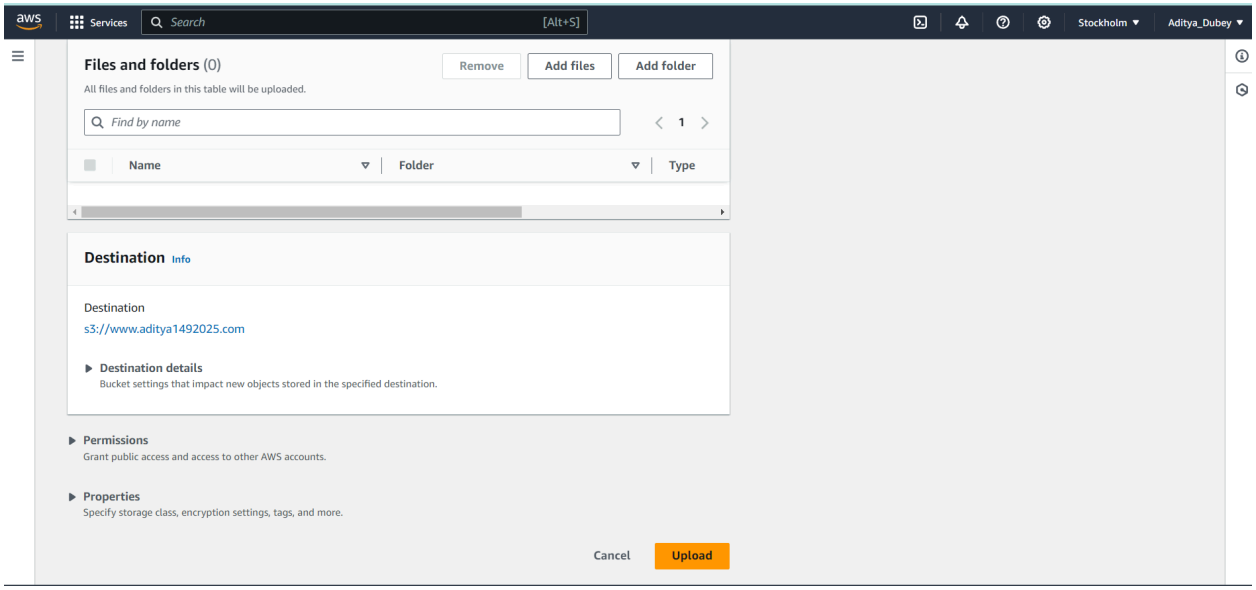
Name:Aditya Dubey Div:D15C Roll no:10

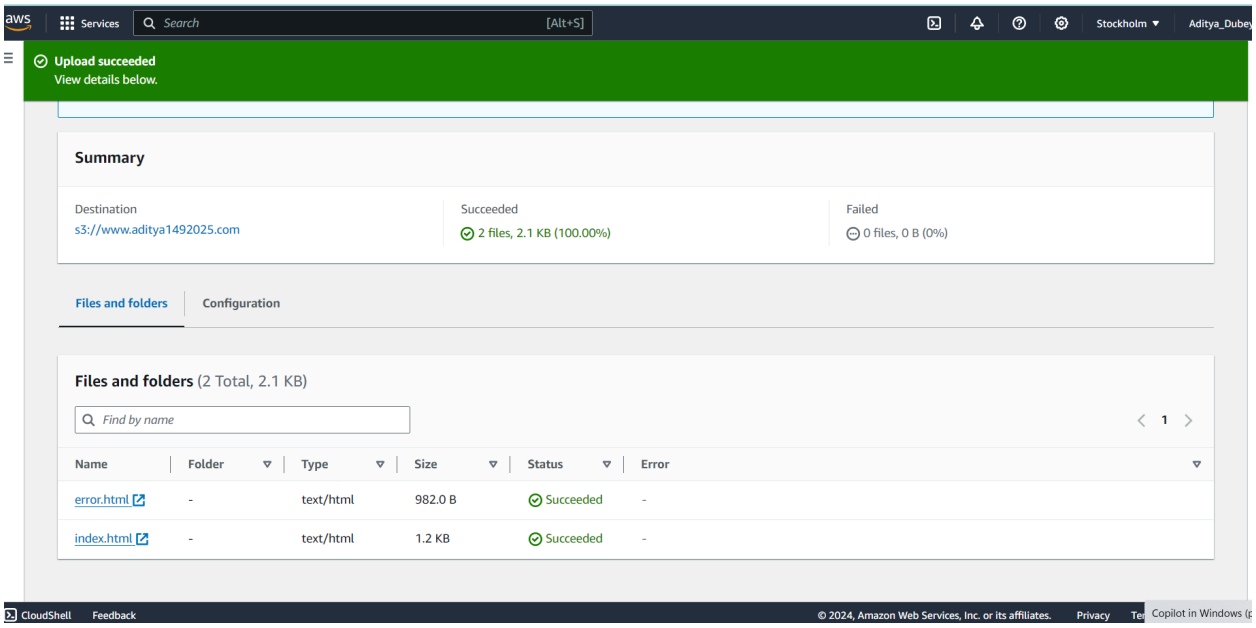


Step 10: Click on Add files. Add all the files you want to upload. Then scroll down and click on Upload

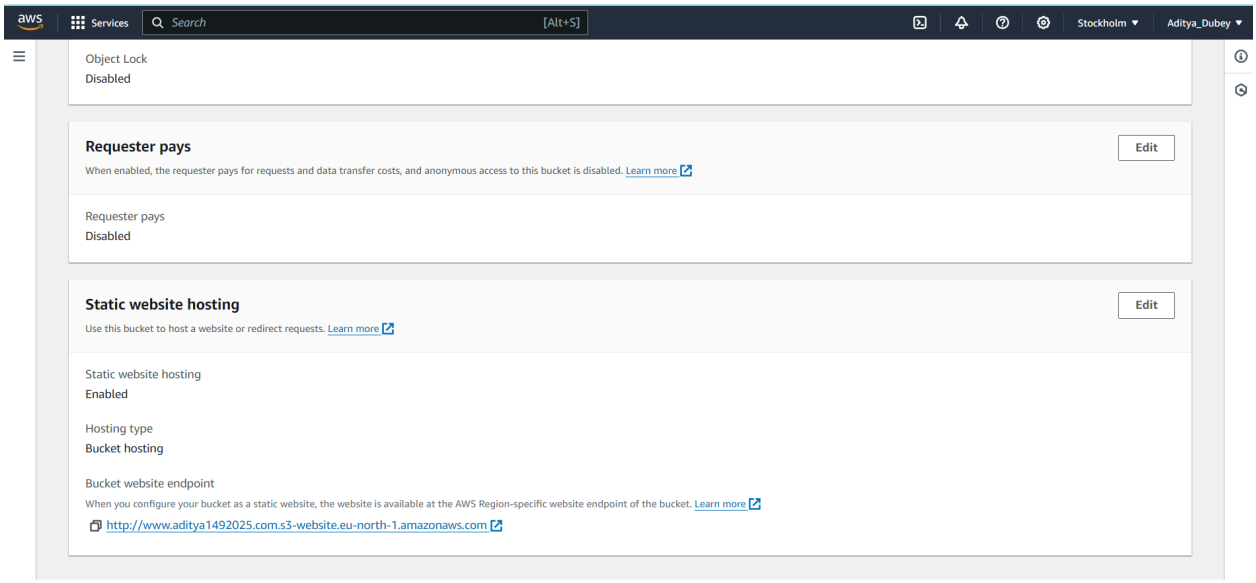


Name:Aditya Dubey Div:D15C Roll no:10





Step11: This will take you to the Objects screen. Switch to Properties, scroll down to Static web hosting. There you would find the link (Bucket website endpoint) to your website.

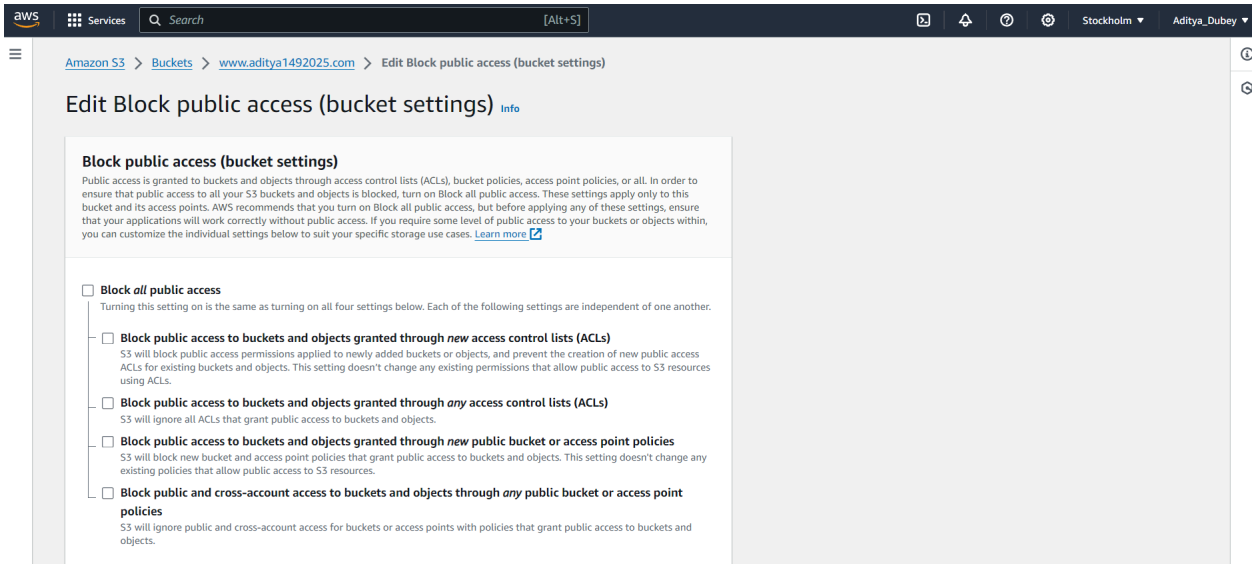


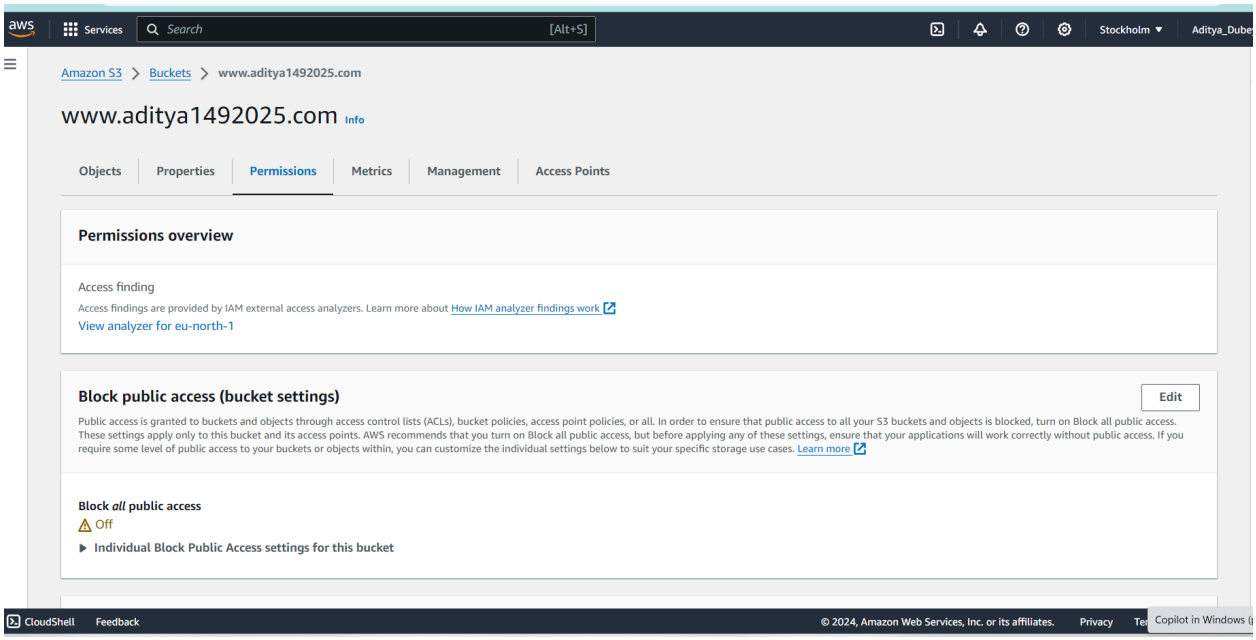
Step12: Open the link. It will show a 403 forbidden error screen as the contents of the bucket are not available for the public users. To change this, go to Permissions tab, go to Block public access and click on edit

Step 13:

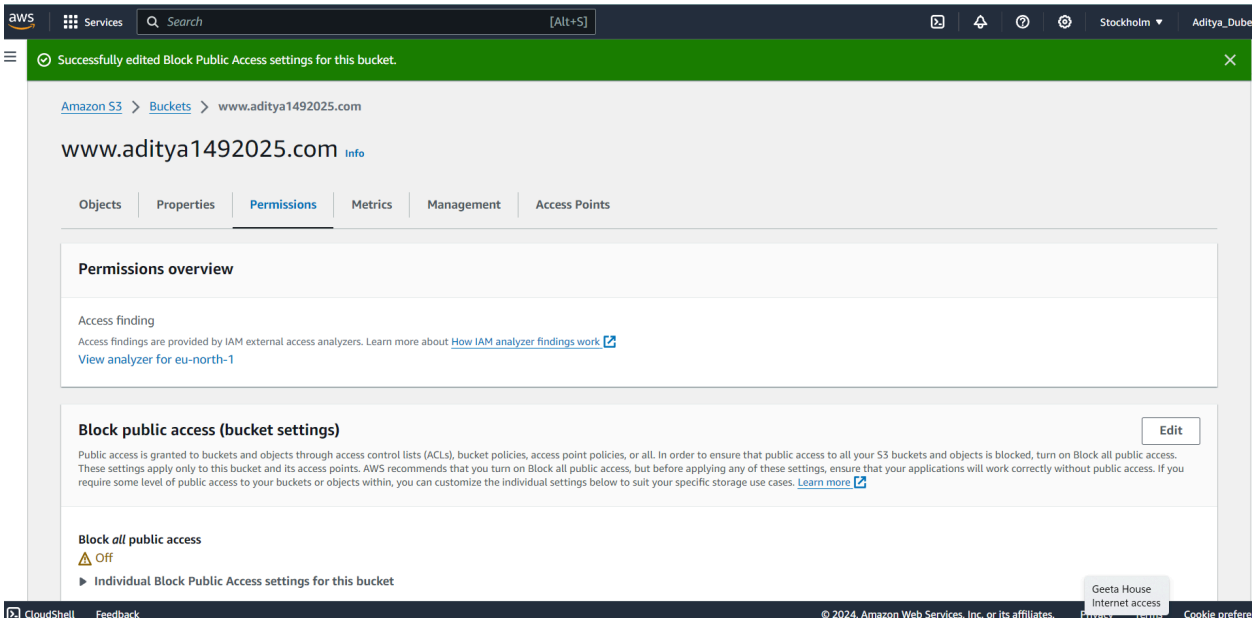
Uncheck the Block all public access checkbox and click on save changes

Name:Aditya Dubey Div:D15C Roll no:10





Step 14:Successfully Changed the Settings

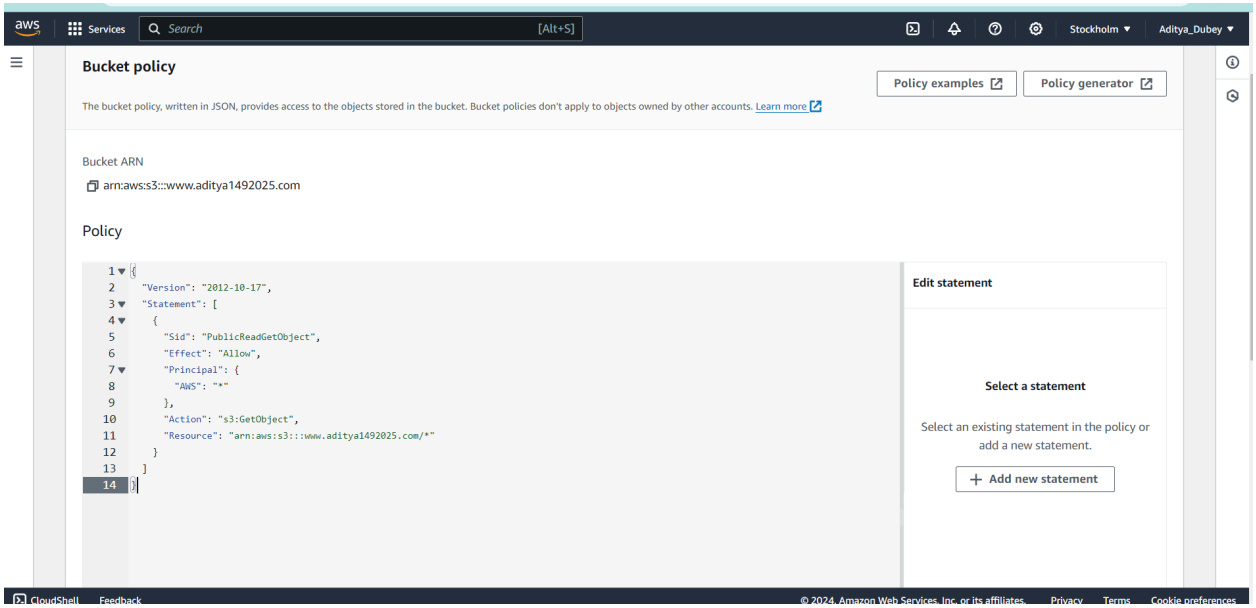


Step 15: Scroll down to bucket policy and click edit and paste the code from given Github Link https://gist.github.com/Savjee/b4b3a21d143a30e7dc07

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadGetObject",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::YOUR-BUCKET-NAME-HERE/*"
    }
  ]
}
```

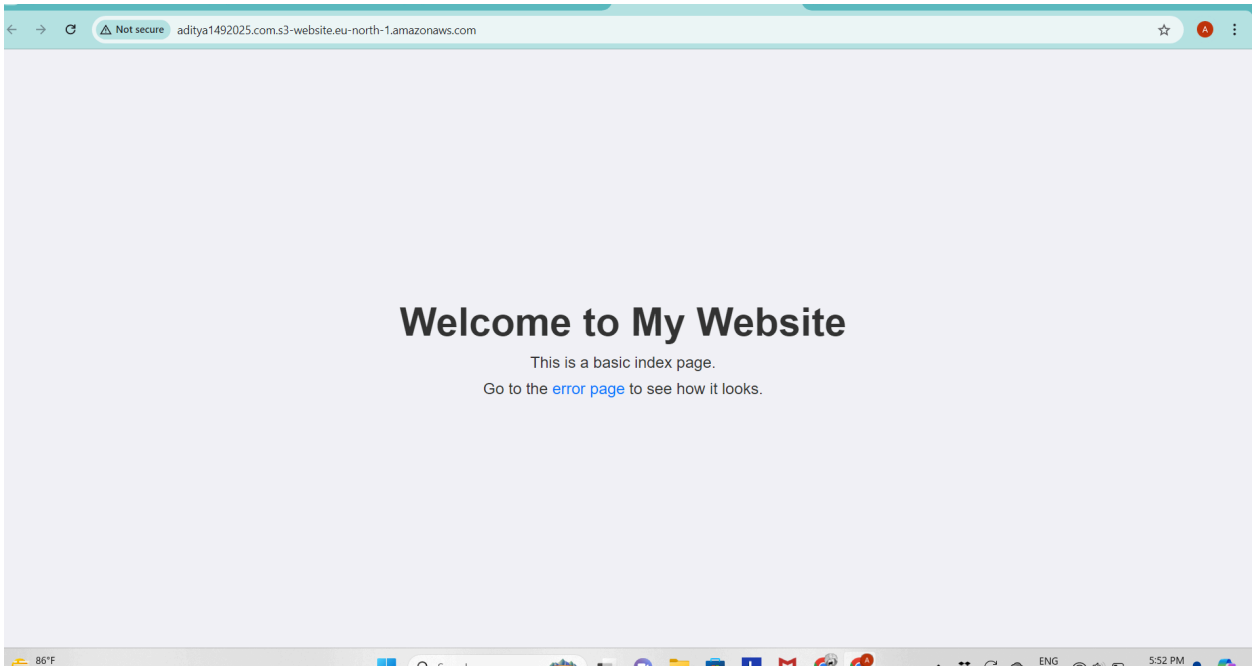
Paste this code snippet in the policy textarea. Replace YOUR-BUCKET-NAME-HERE with the name you have given to your bucket. Save the changes

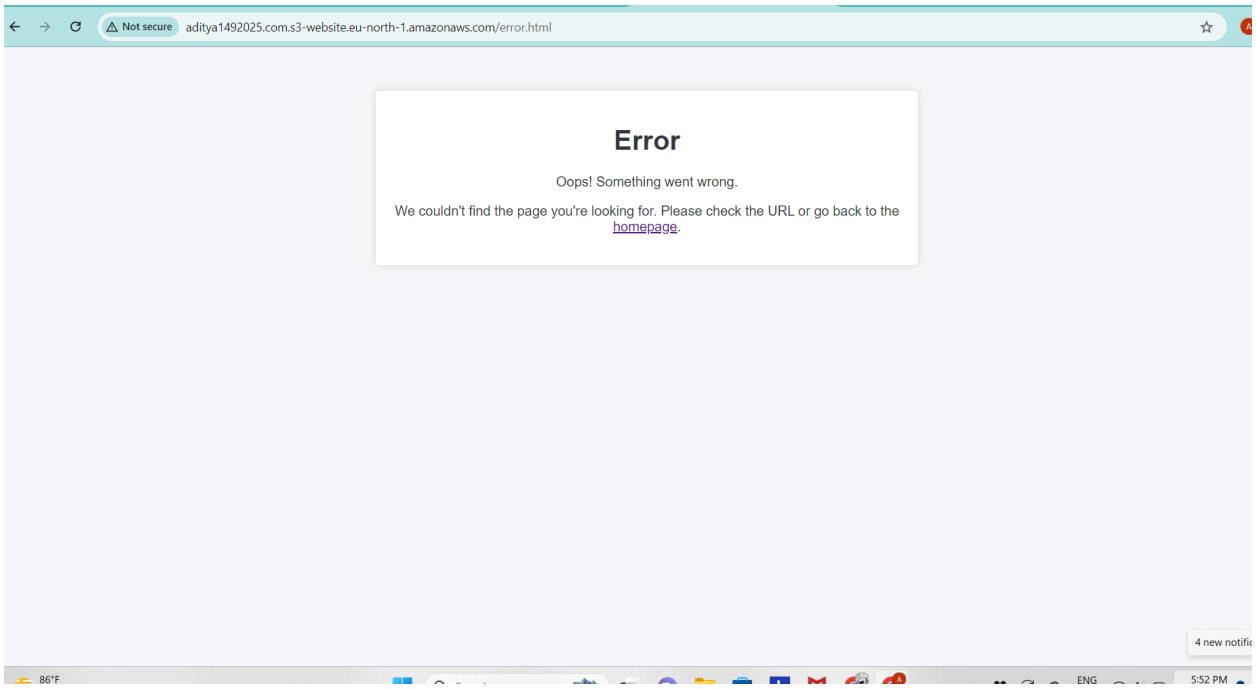
Name:Aditya Dubey Div:D15C Roll no:10



Step 16: Now reload the website. You can see your website

Name:Aditya Dubey Div:D15C Roll no:10

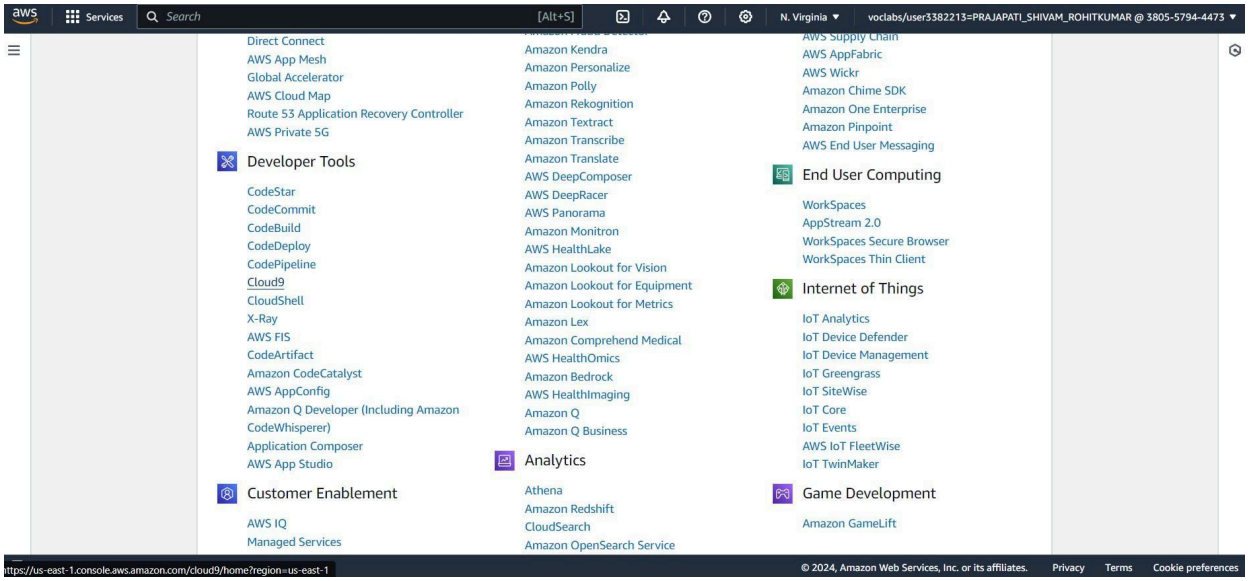




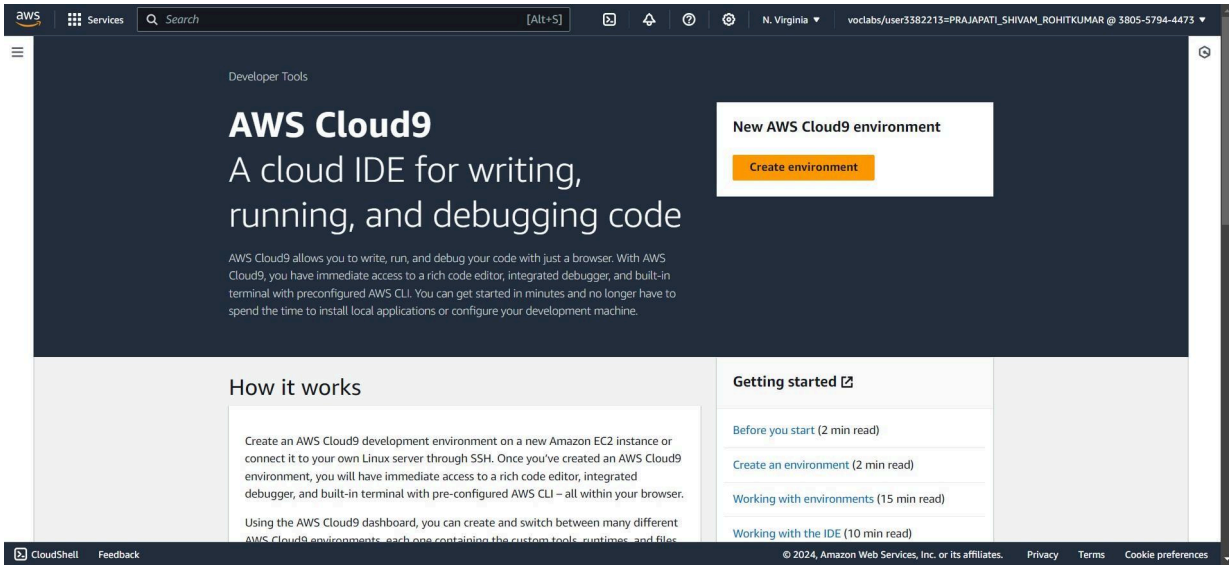
Experiment No: 1(B)

Step 1: Set up Cloud9 environment.

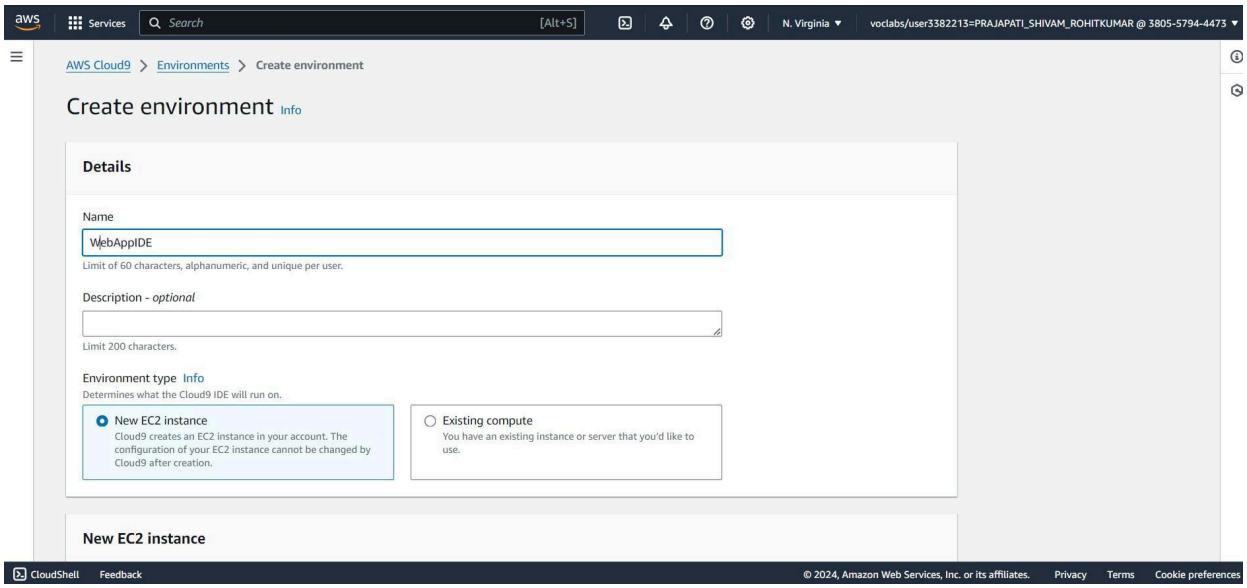
1) Go to Cloud9 services under developers tool in All services



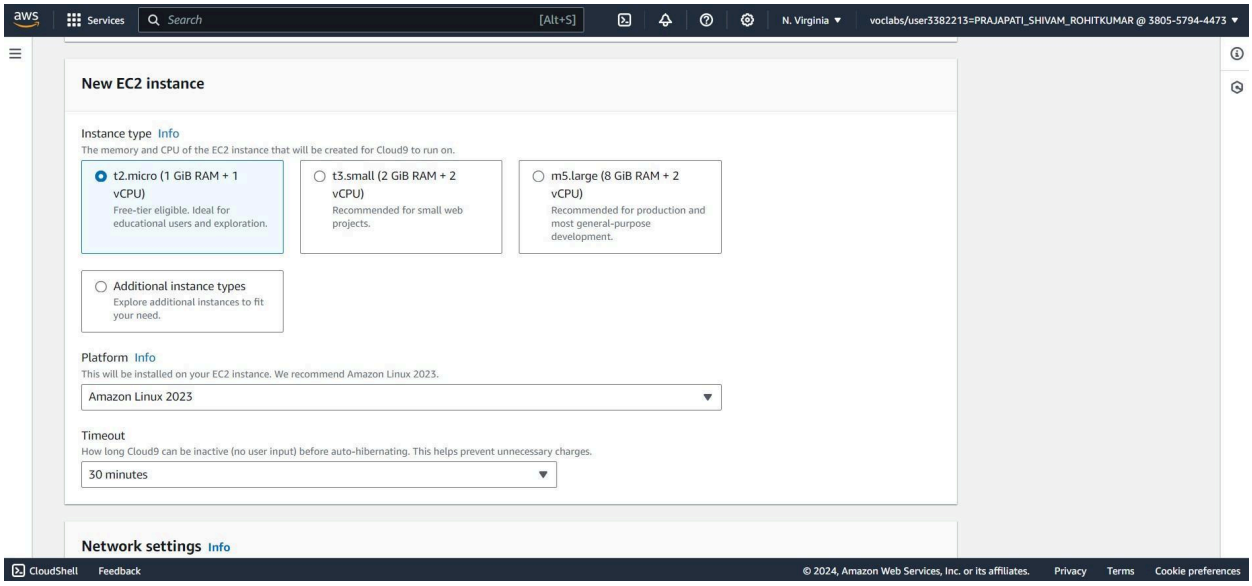
2) Click on create environment



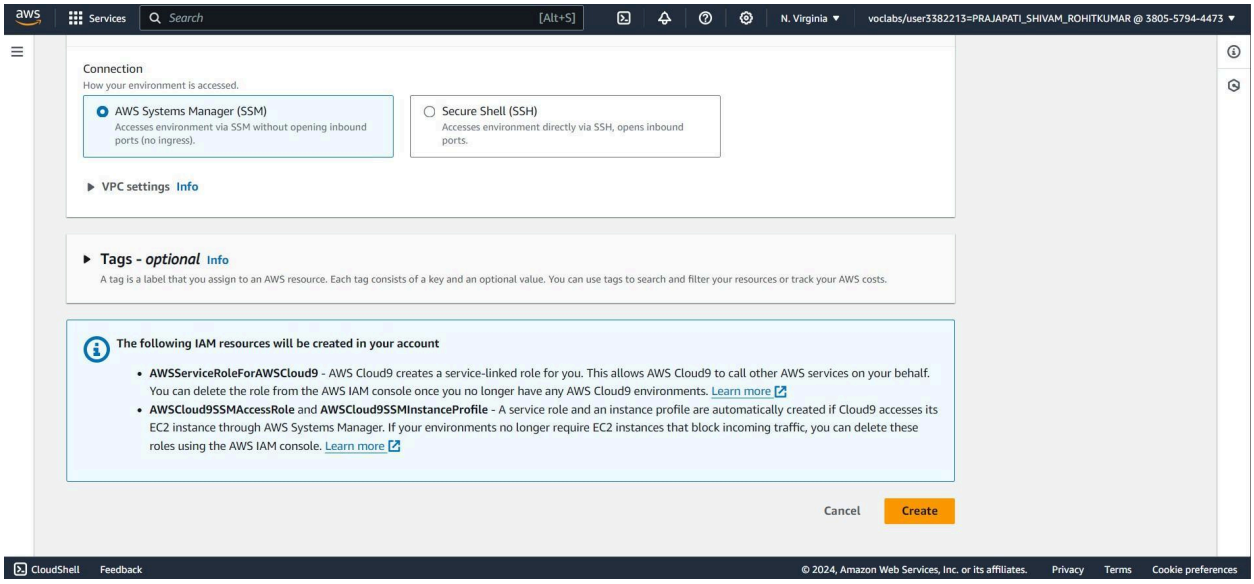
3) Give the name to your Environment ,keeping the other settings as default



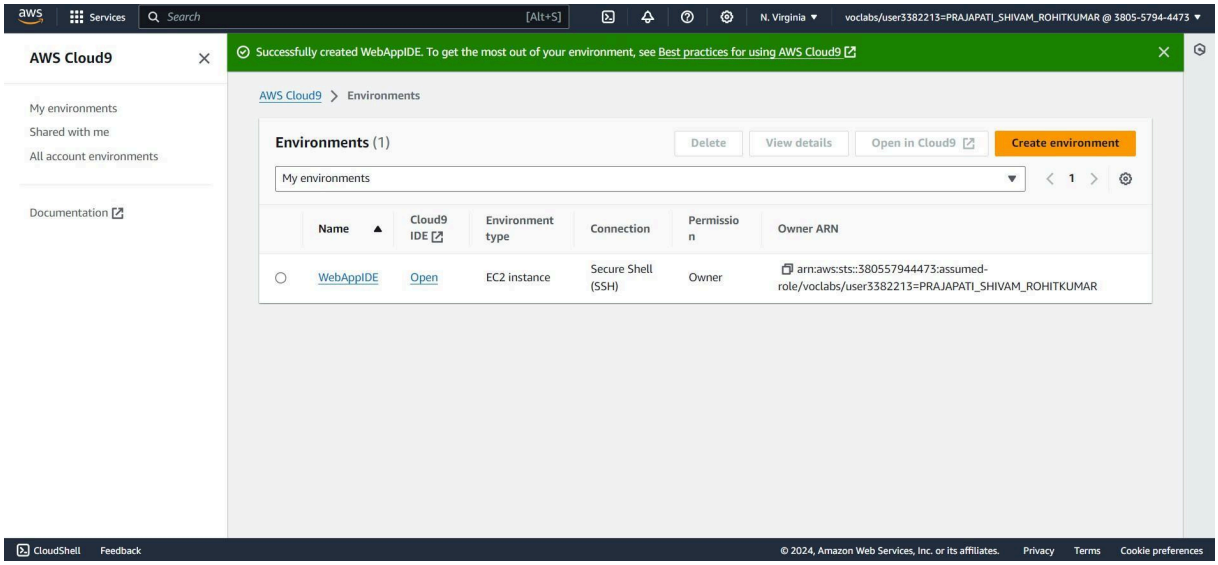
4) Select the correct platform type as shown below and keep the others details as default



5) Click on SSH under connection type in network settings and click on Create

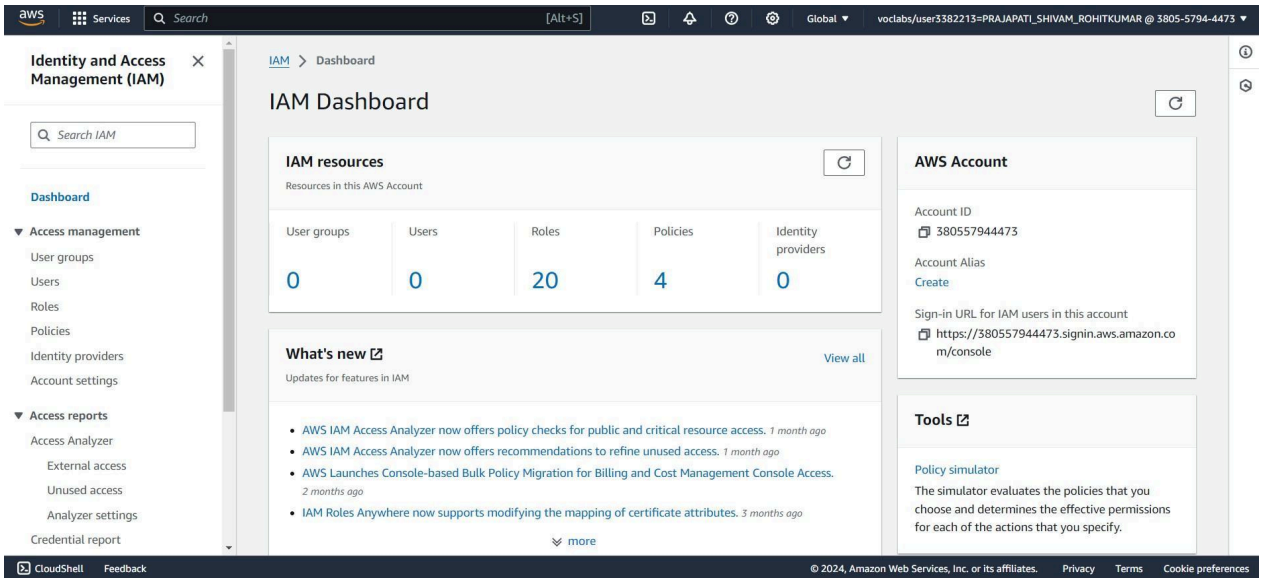


6) Successfully created the environment so now click on open

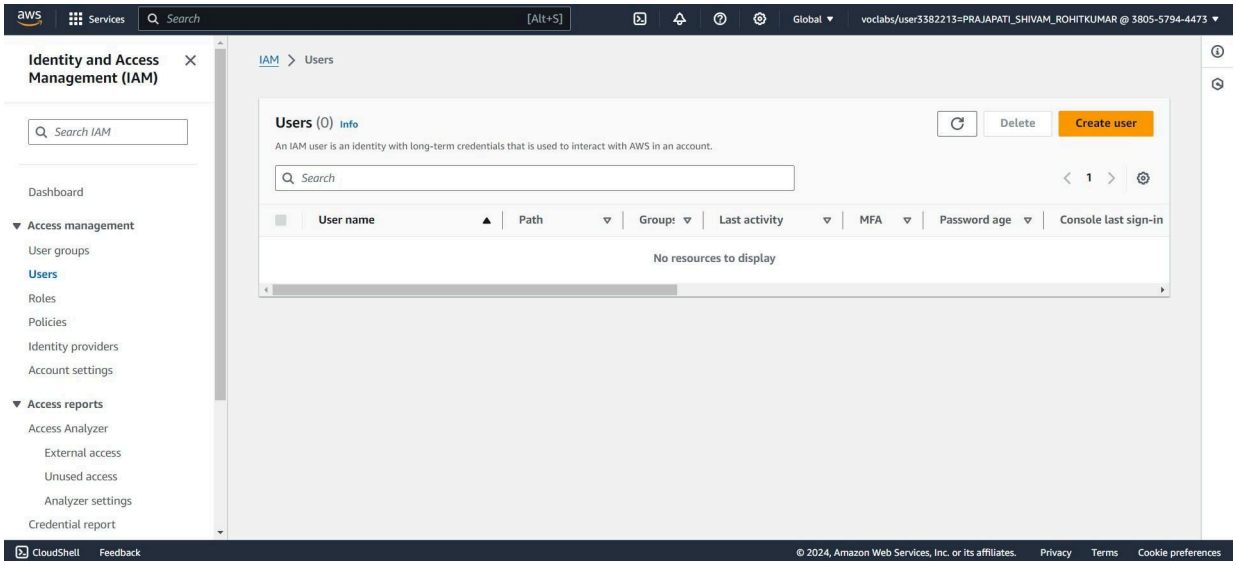


Step 2: Creating IAM user.

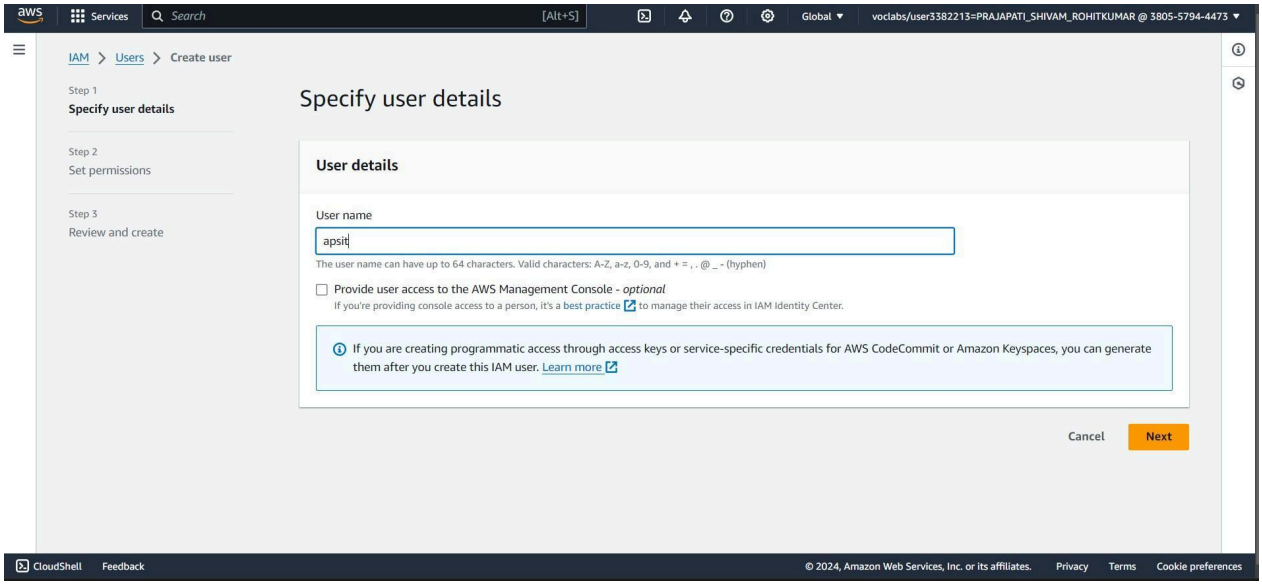
1) Search IAM on the services search bar and open it. Click on Create User



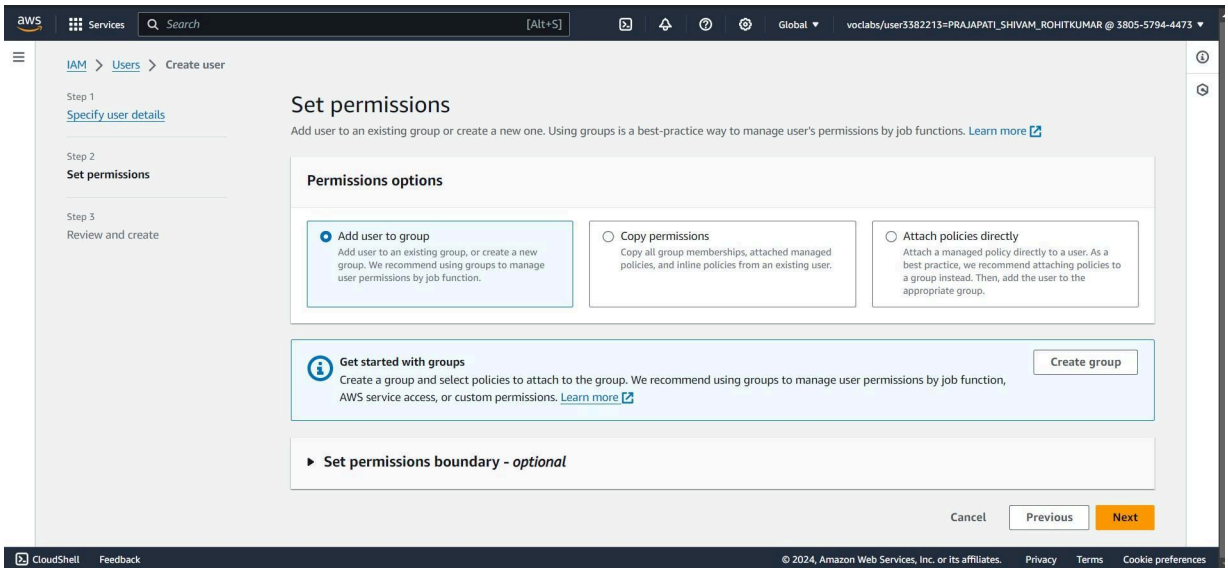
2) Click on the create user



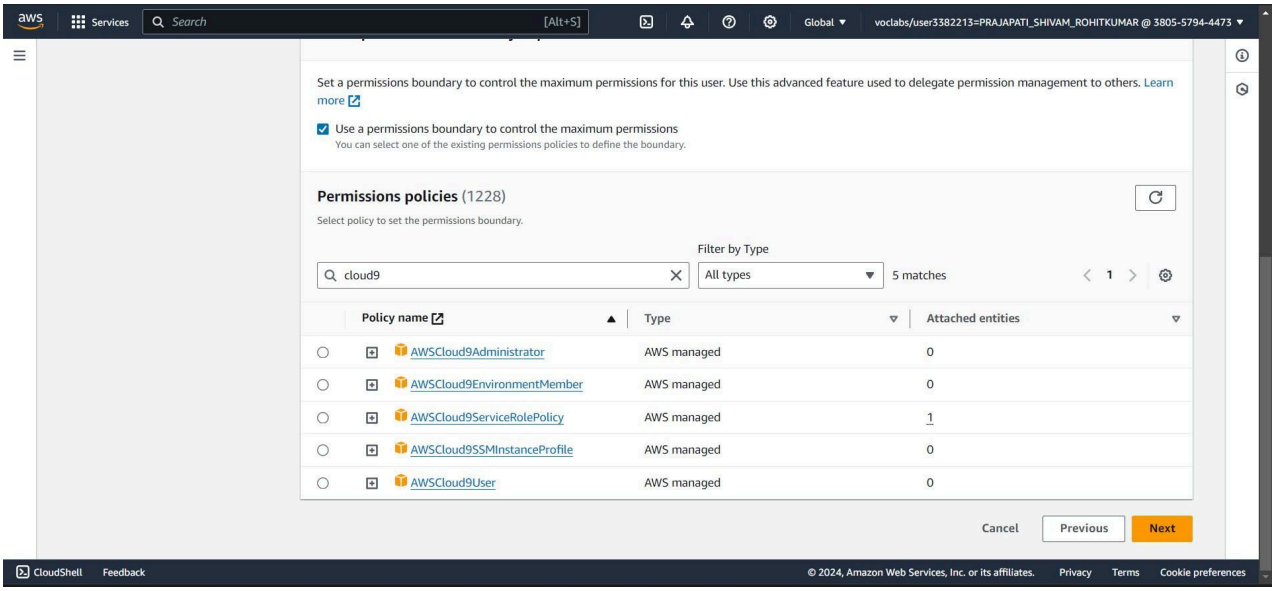
3) Write the name of the user you want to add and click on next



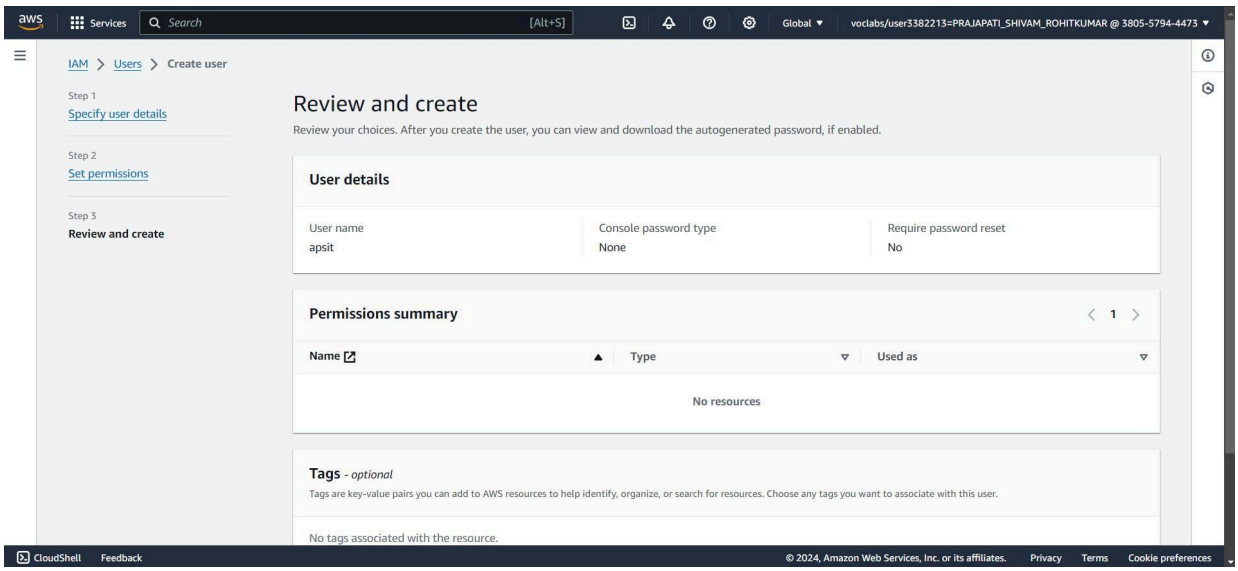
4) Click on the drop down menu of the set permissions boundary

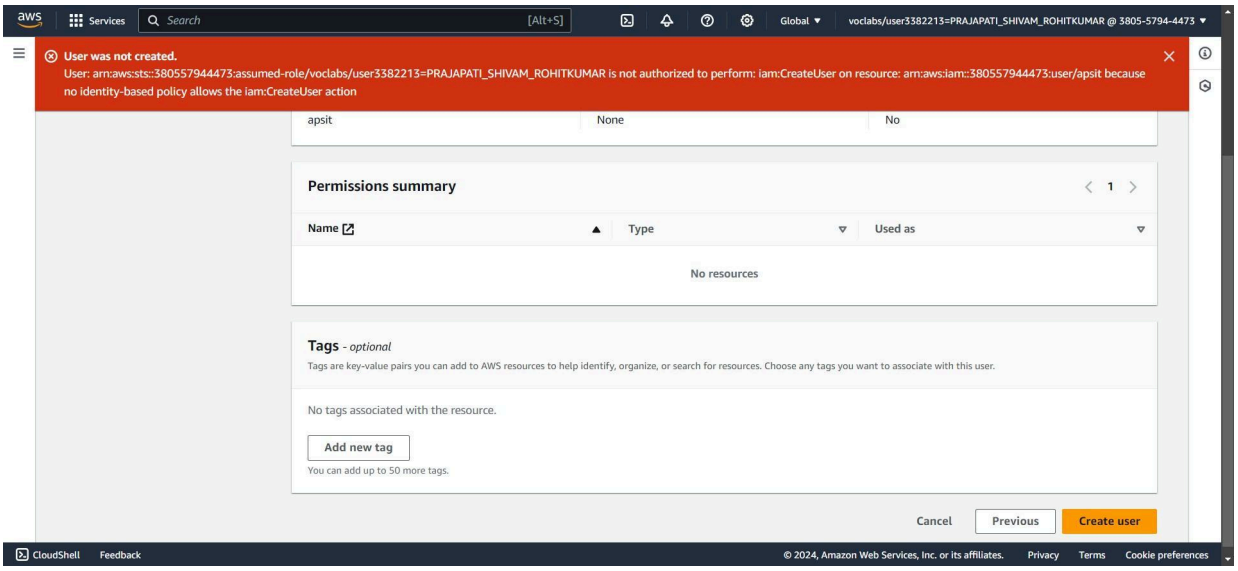


5) Click on the checkbox and search for cloud9 under permissions policies ,click on next



6) Scroll down and click on create user

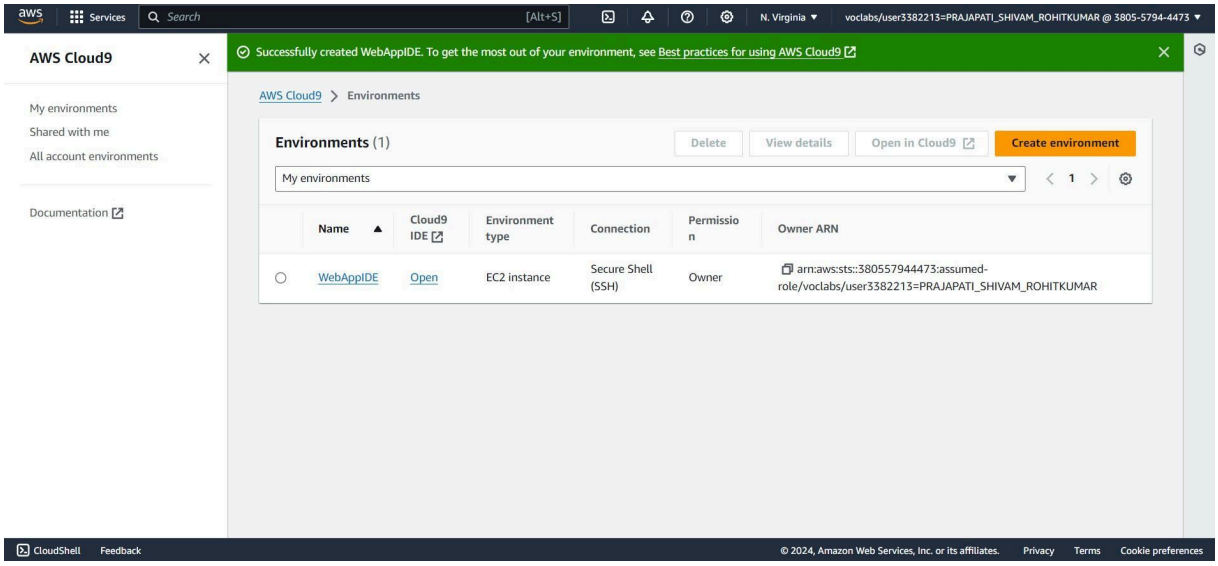




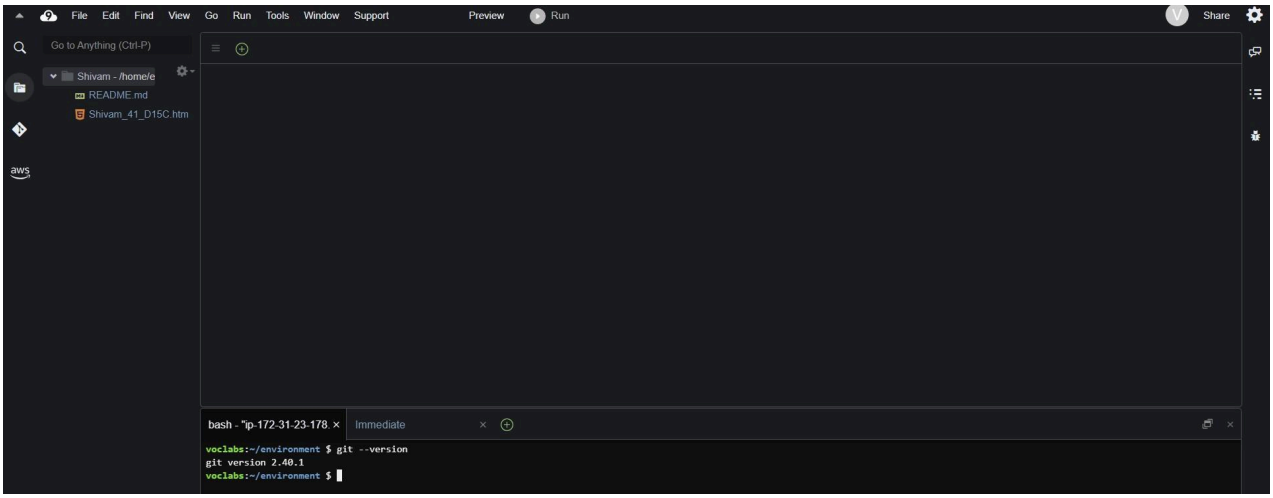
When we go to add user to a group, the AWS Academy account throws an error as we do not have the permissions to create a group. So we have to use our personal AWS account for this part.

Step 3: Working on Cloud9 IDE

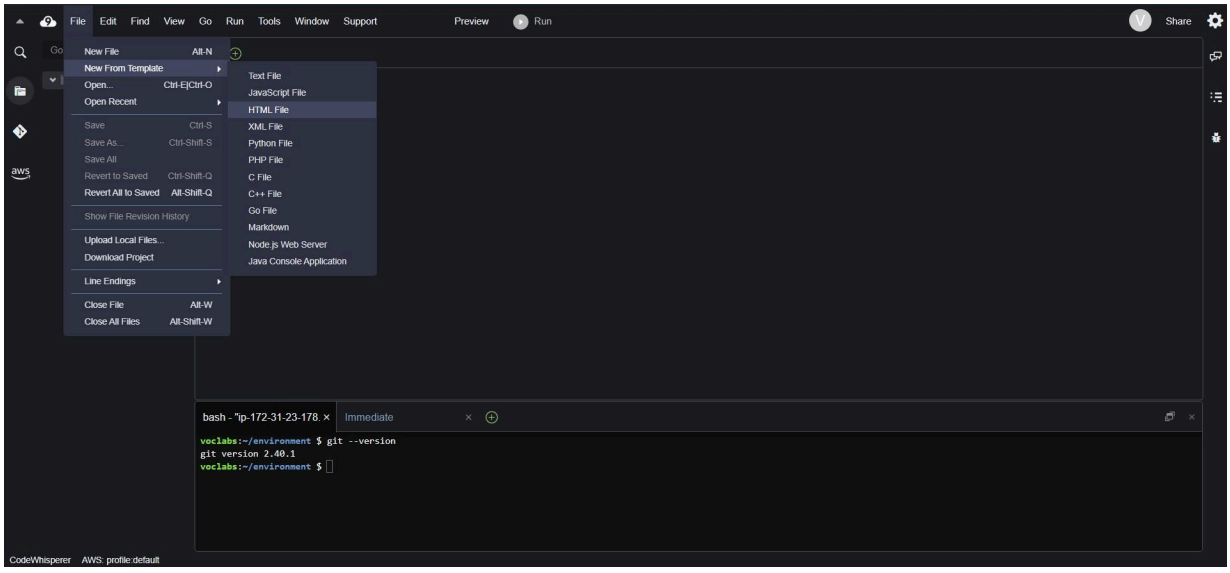
1) Go to Cloud9 services. Click on Open under Cloud9 IDE



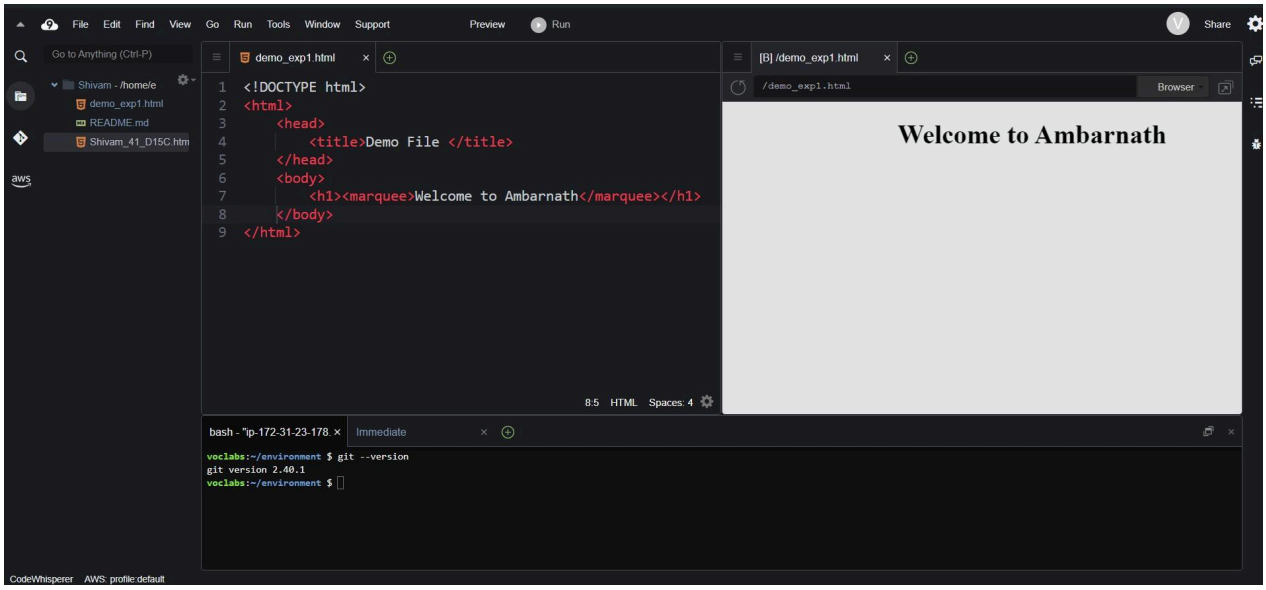
2) This is the Cloud9 IDE interface. The major part of the screen is the coding IDE. There is a command console just below it. For example, the command git --version is run.



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

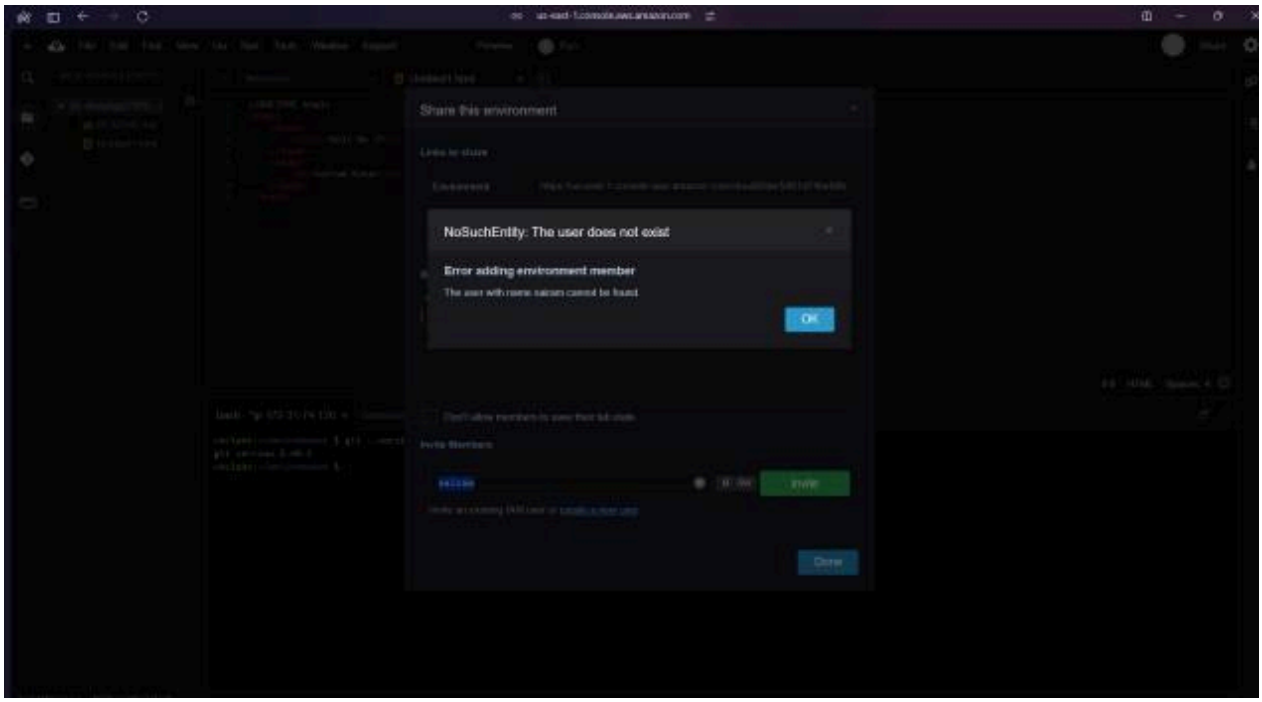


4) Make a basic website on the HTML template and save it.



.

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



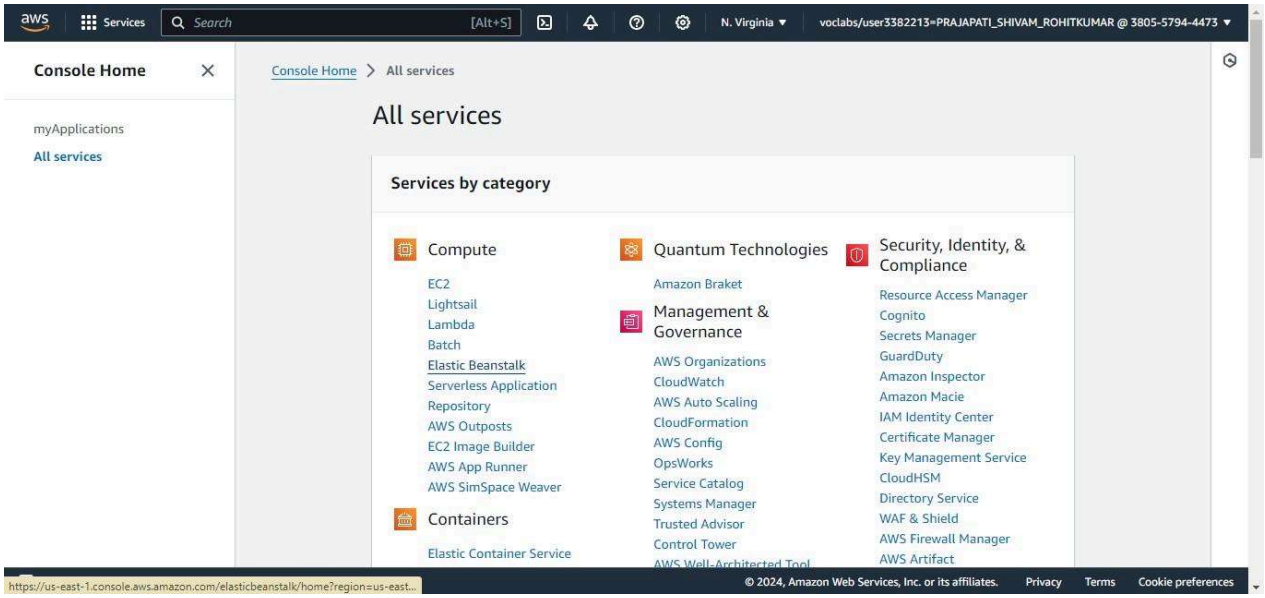
Here, it gives an error as Cloud9 was created on the academy account where creating an IAM group is not available, meanwhile on the personal account, the services of Cloud9 have been deprecated. So currently, it is not possible to integrate the cloud9 and IAM parts of the experiment.

Name:Aditya Dubey **Div:D15C** **Roll No:10**

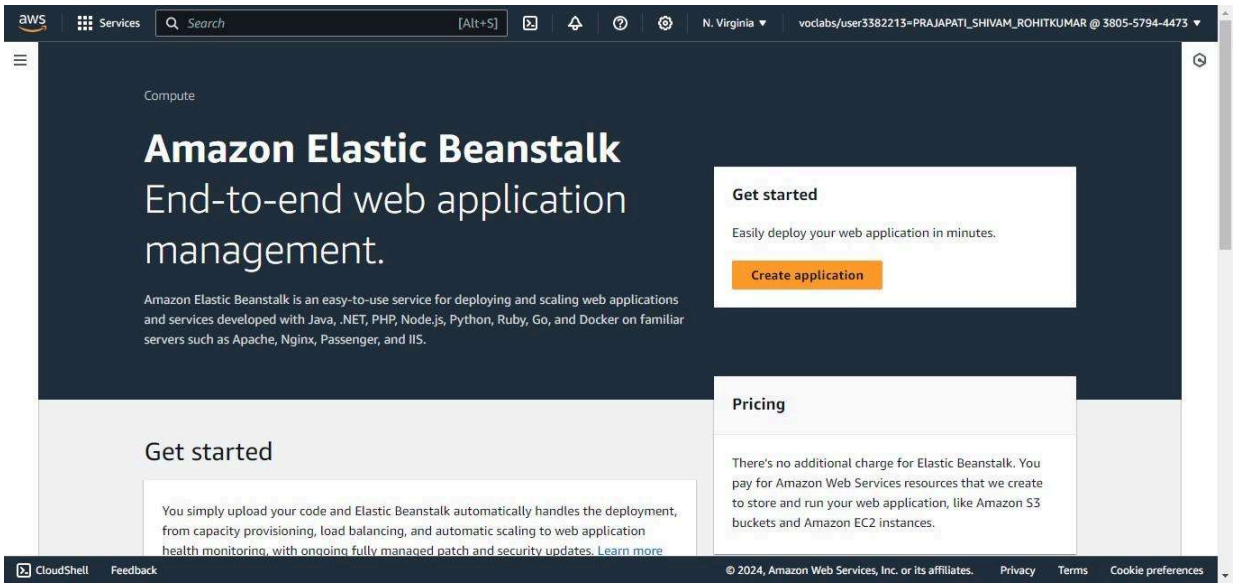
Name:Aditya Dubey **Div:D1 5C** **Roll No:10**

Experiment No :2

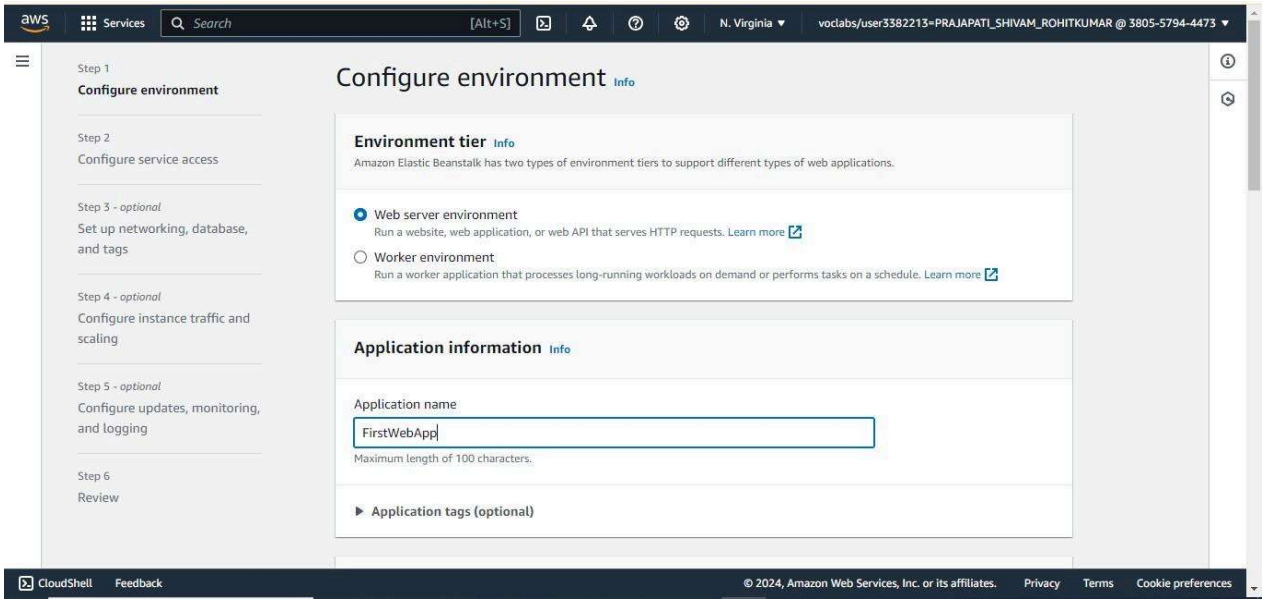
Step 1: Login to your AWS console. Search for Elastic Beanstalk in the searchbar near services.



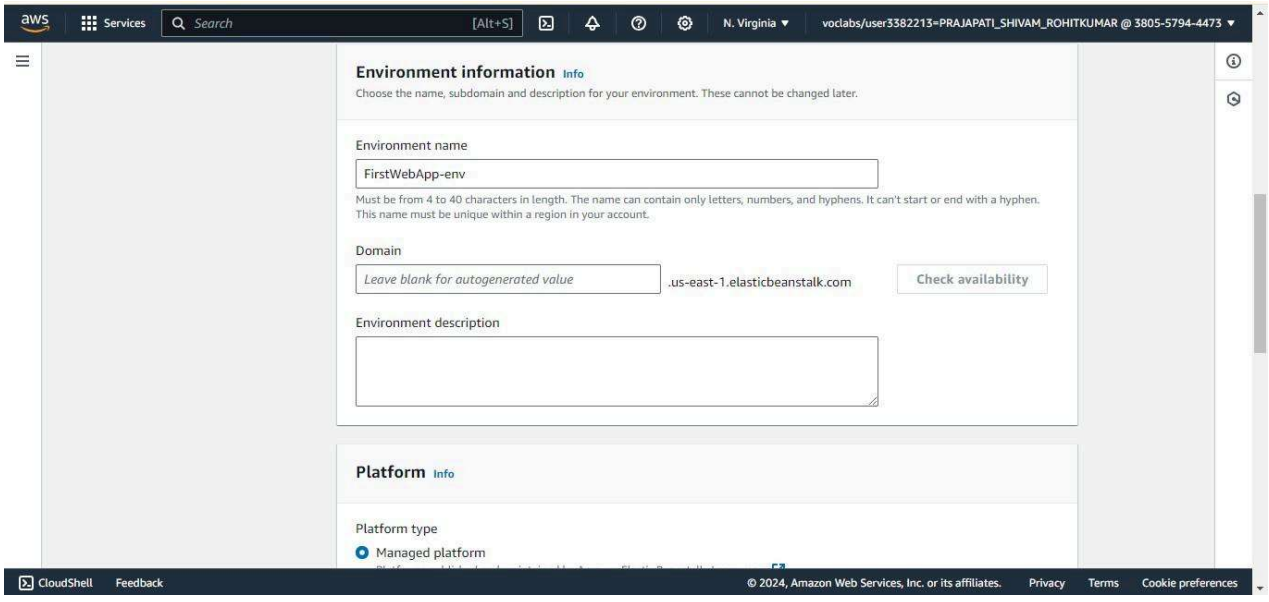
Step 2: Go to Elastic Beanstalk and click on Create Application



Step 3: Enter the name of your application. Scroll down and in the platform, select platform as PHP. Keep the application code as Sample Application. Set the instance to single instance. Click on NEXT.

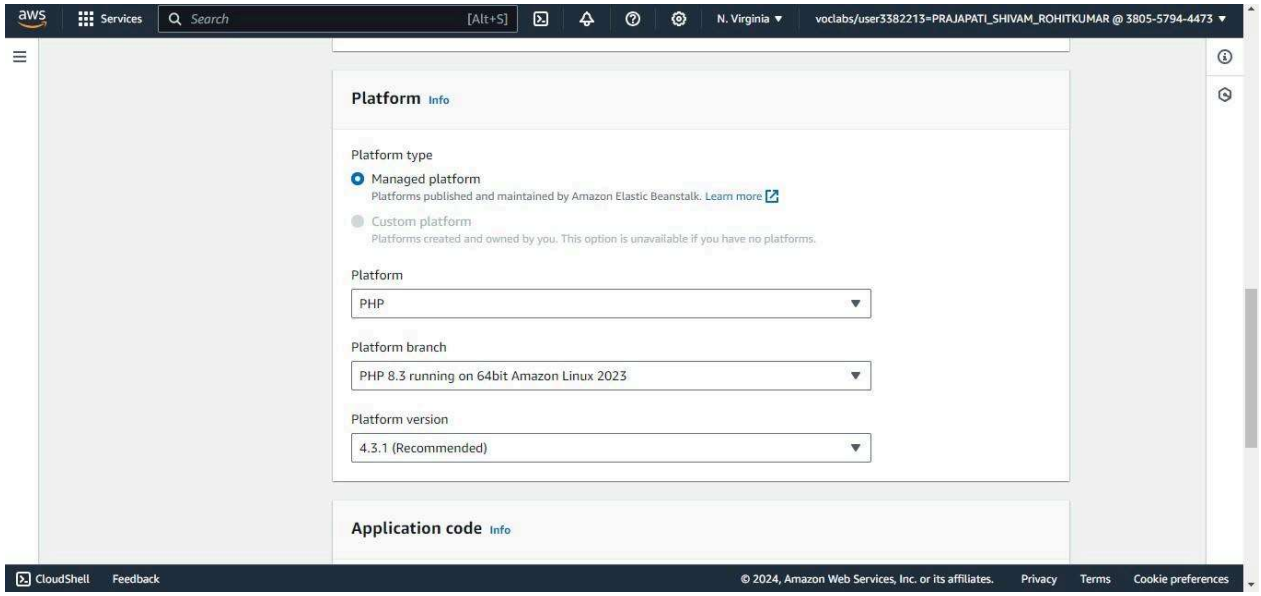


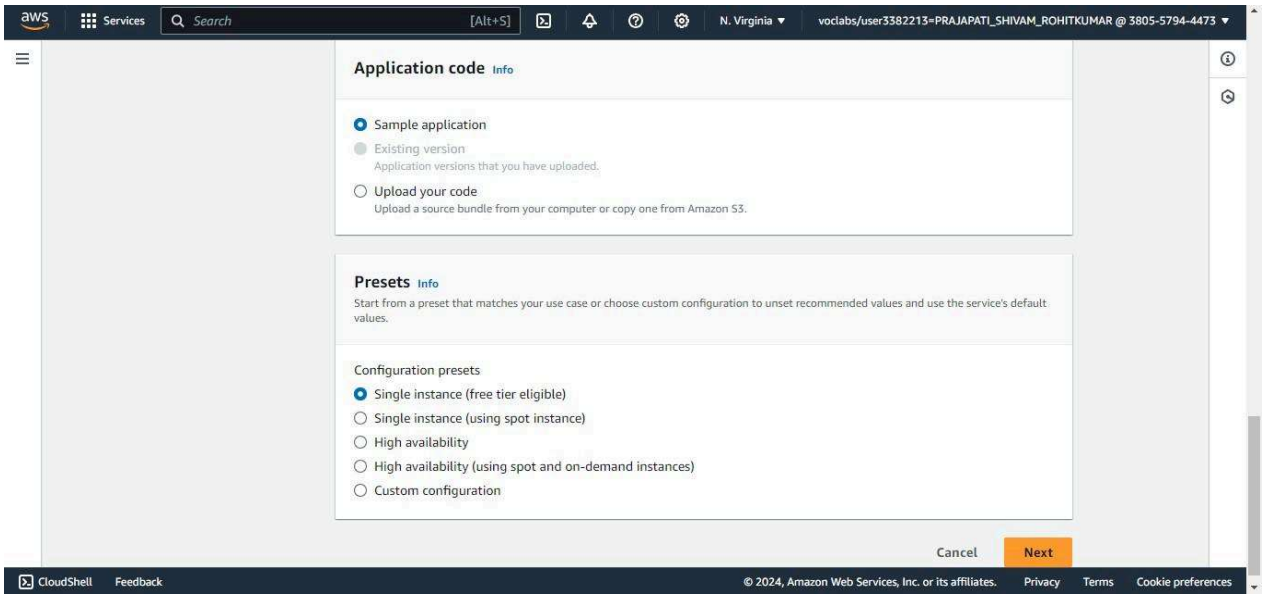
(Scroll Down)



(Scroll Down)

Name:Aditya Dubey **Div:D15C** **Roll No:10**

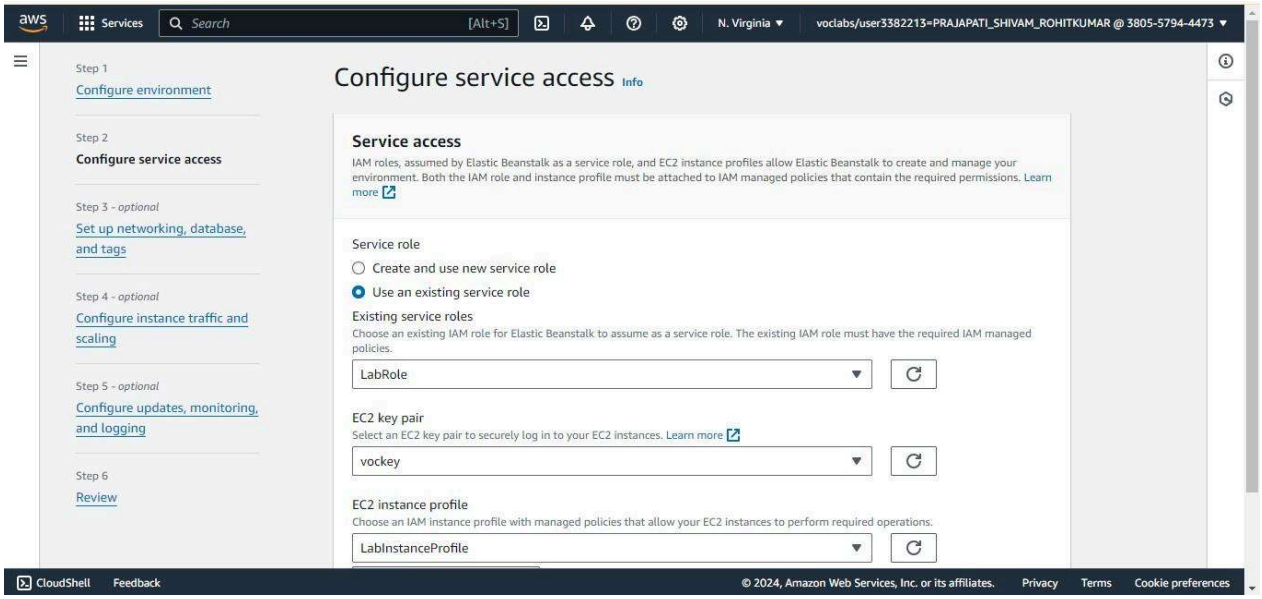




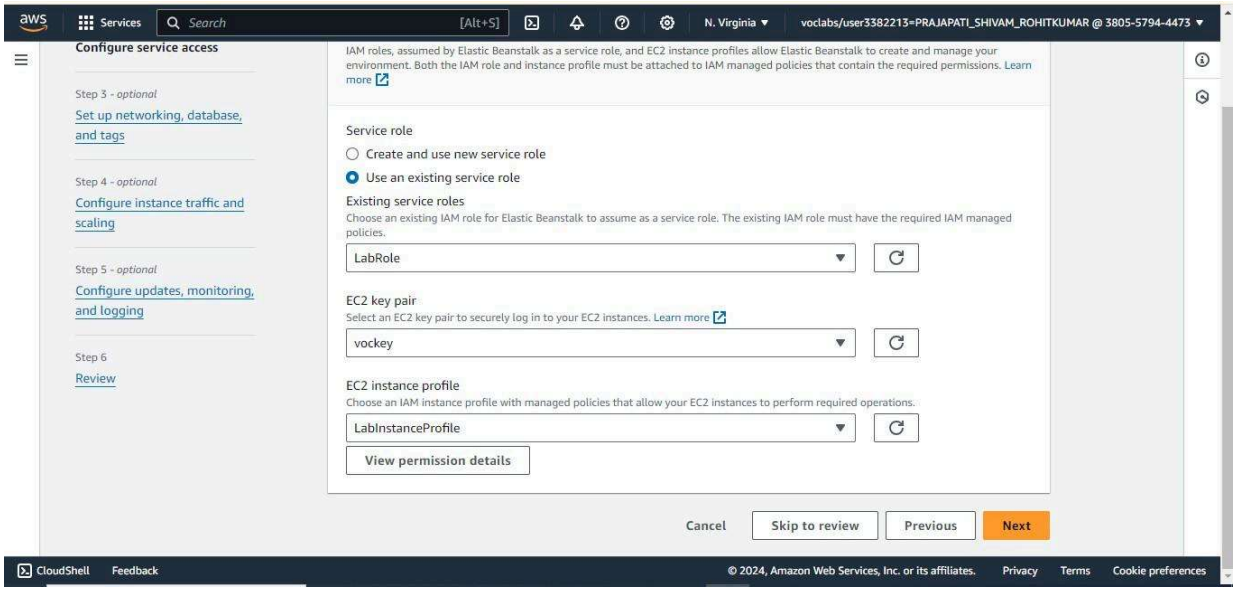
(Click on Next)

Name:Aditya Dubey **Div:D1 5C** **Roll No:10**

Step 4: Use an existing service role and choose whatever service role is present on your account

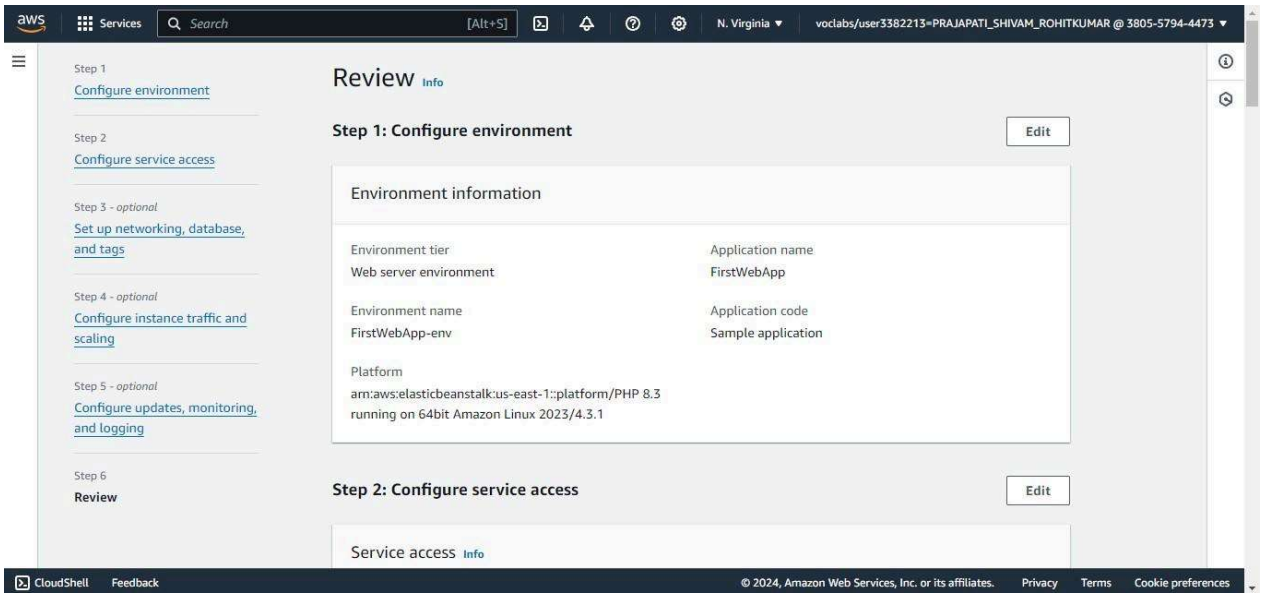


Step 5: Click on Skip to Review

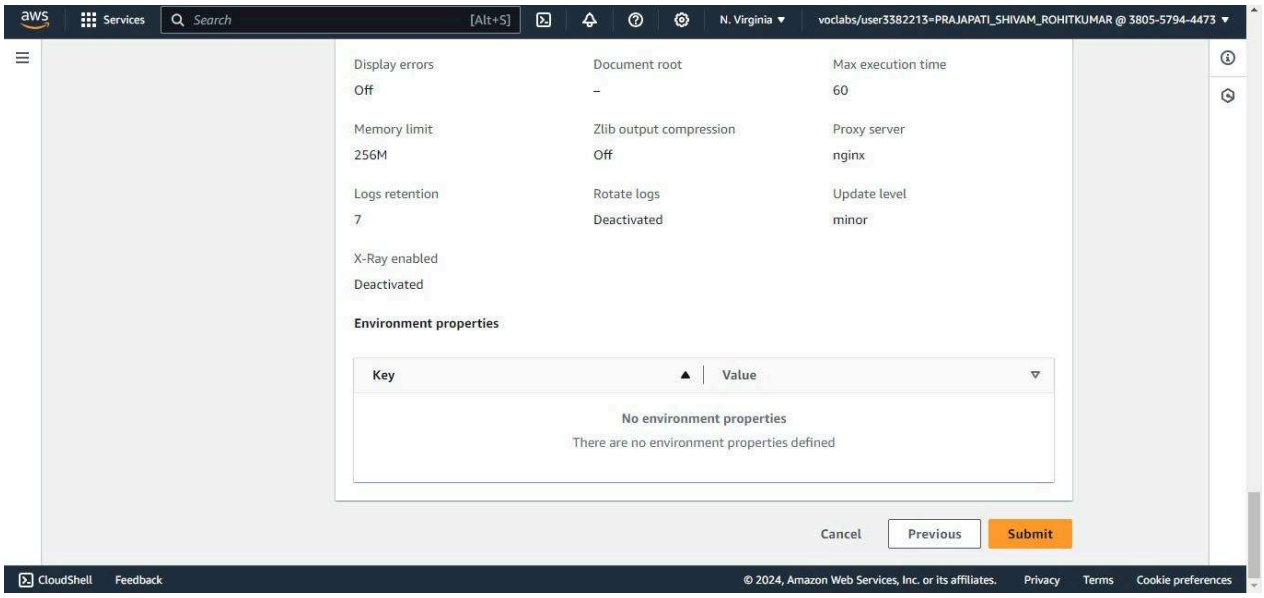


Name:Aditya Dubey **Div:D1 5C** **Roll No:10**

Step 6: Review the settings that you have set up for your application and submit your application

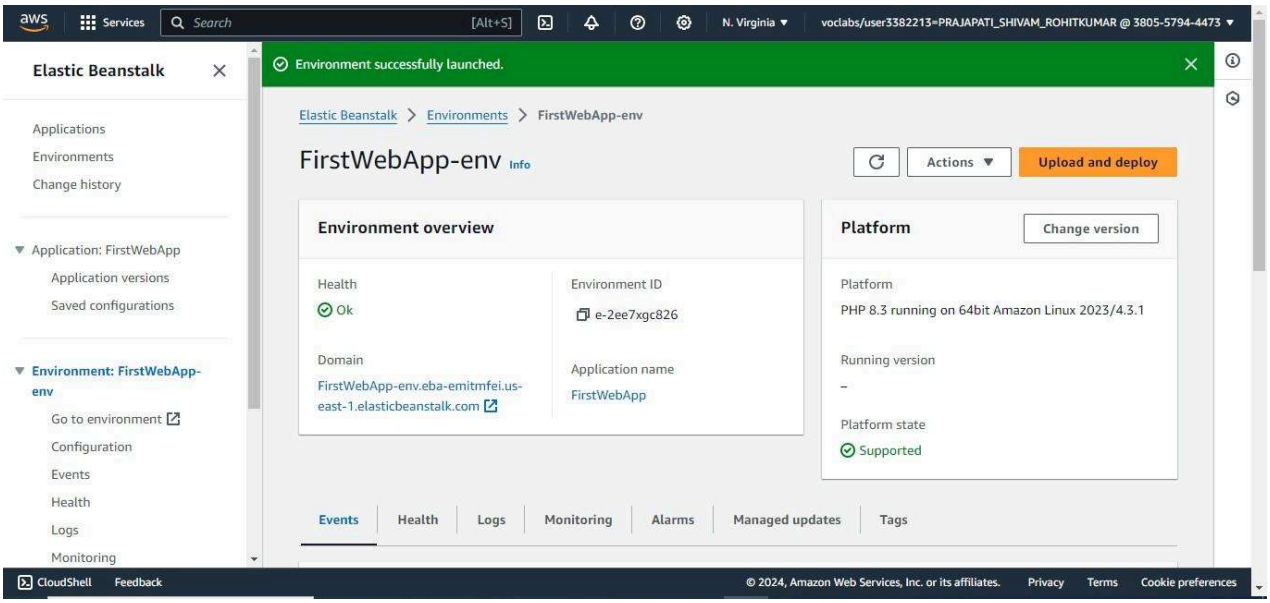


(Scroll Down)

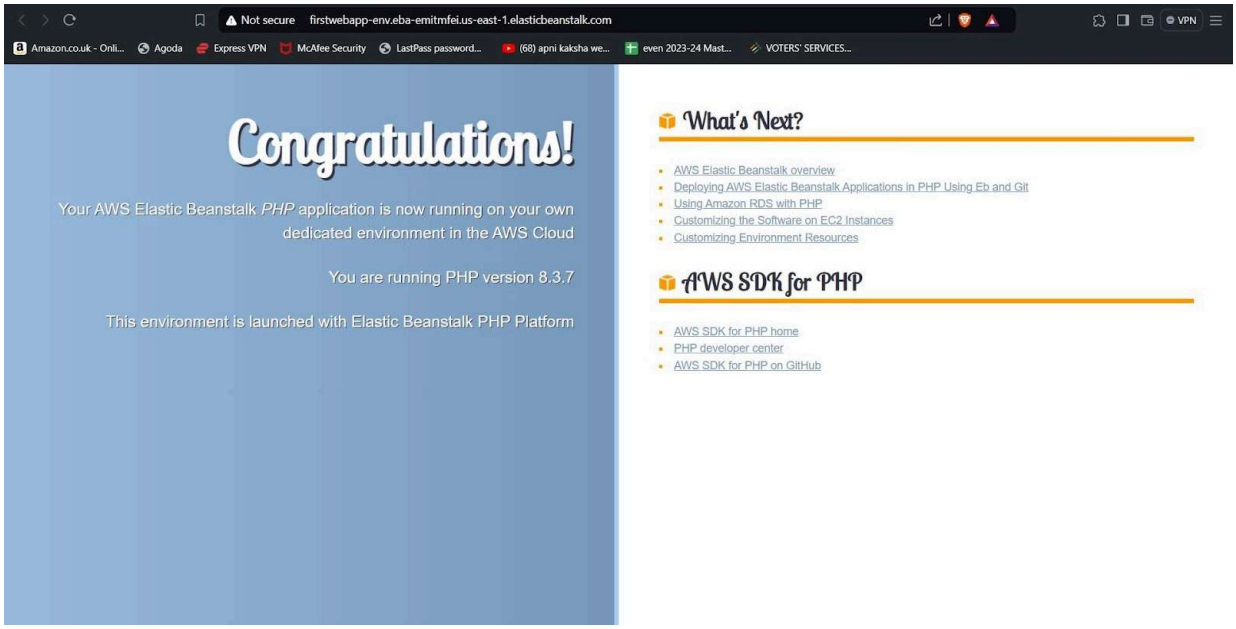


(Click on the Submit)

Name:Aditya Dubey **Div:D1 5C** **Roll No:10**



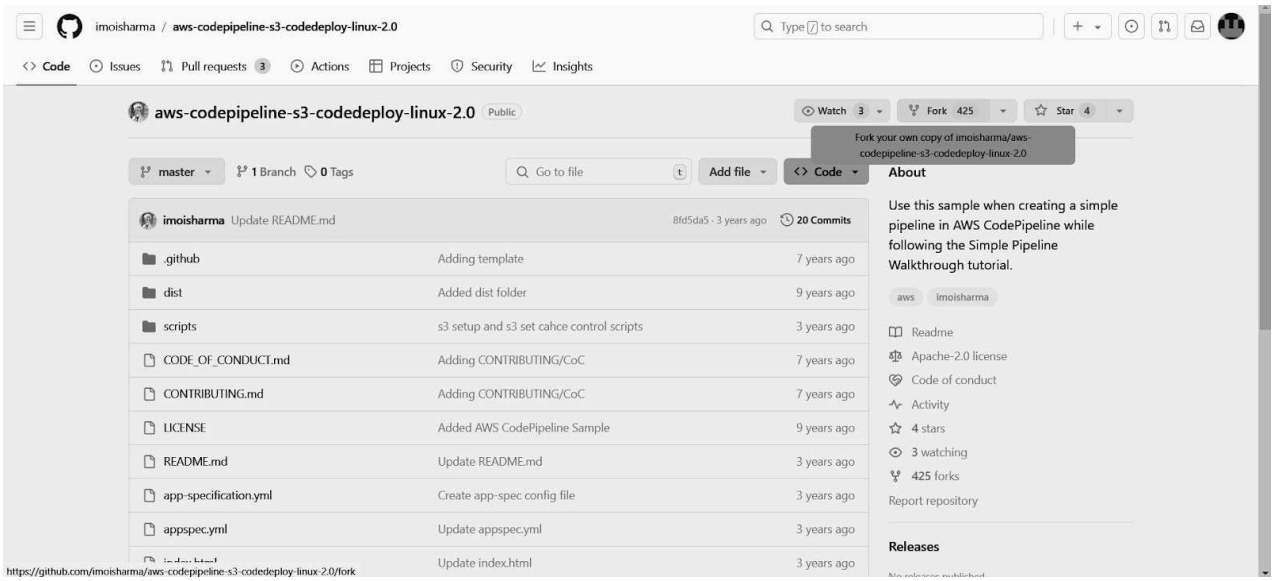
(Click on the link under domain it will redirect to a new page)

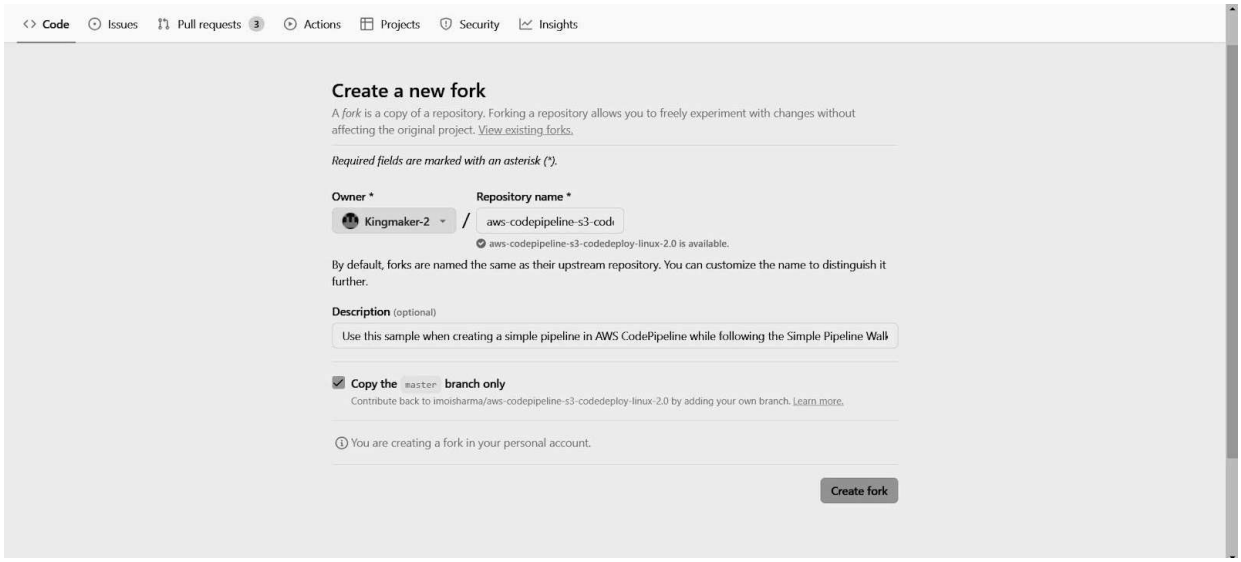


Name:Aditya Dubey **Div:D1 5C** **Roll No:10**

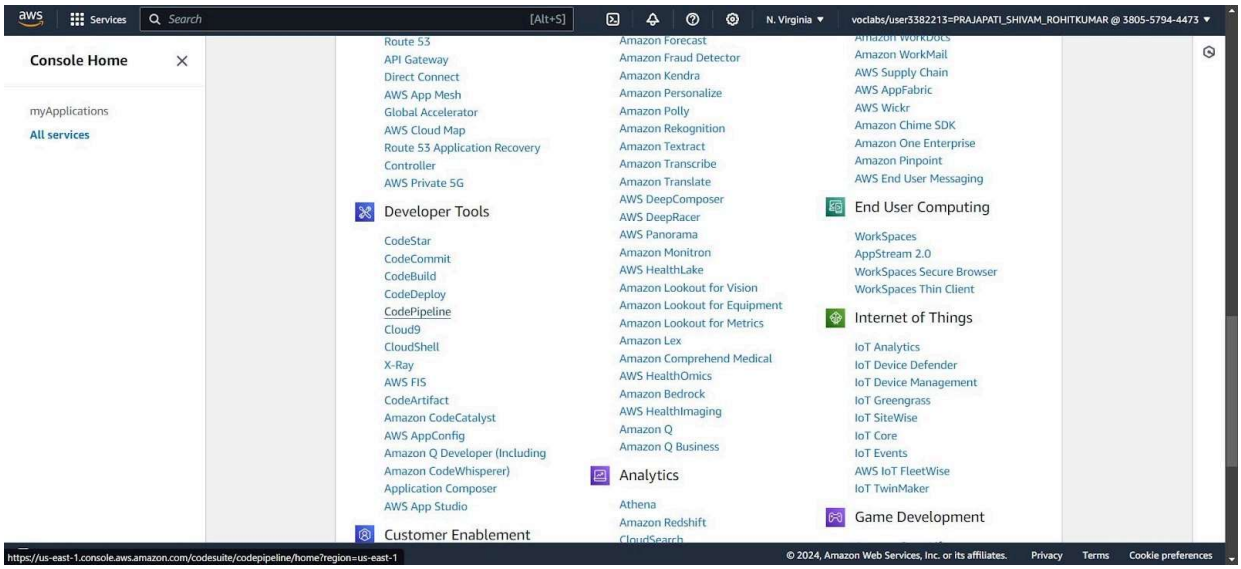
Step 7 : Go to the github link below. This is a github with a sample code for deploying a file on AWS CodePipeline. Fork this repository into your personal github.

https://github.com/aws-samples/aws-codepipeline-s3-codedeploy-linux

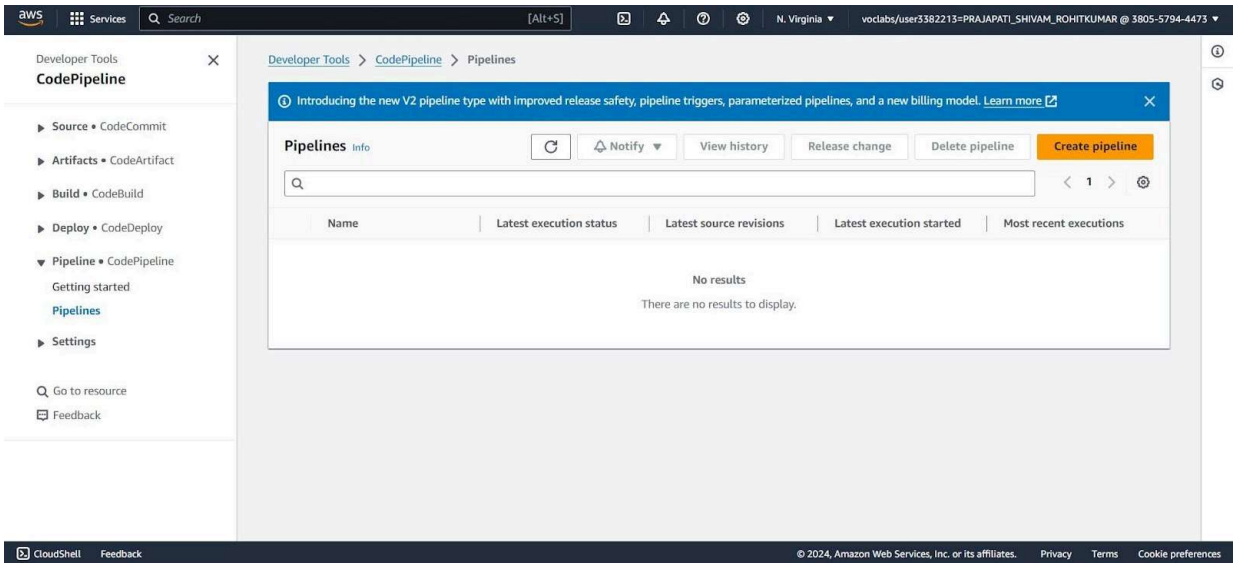




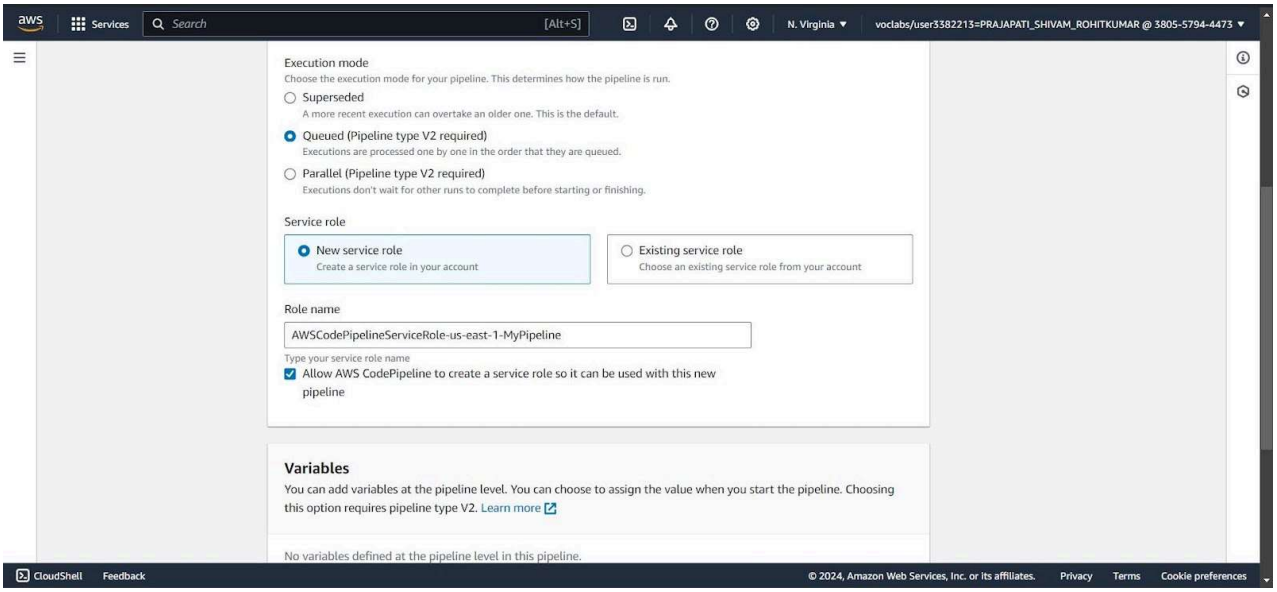
Step 8: Search CodePipeline in the services tab and click on it.

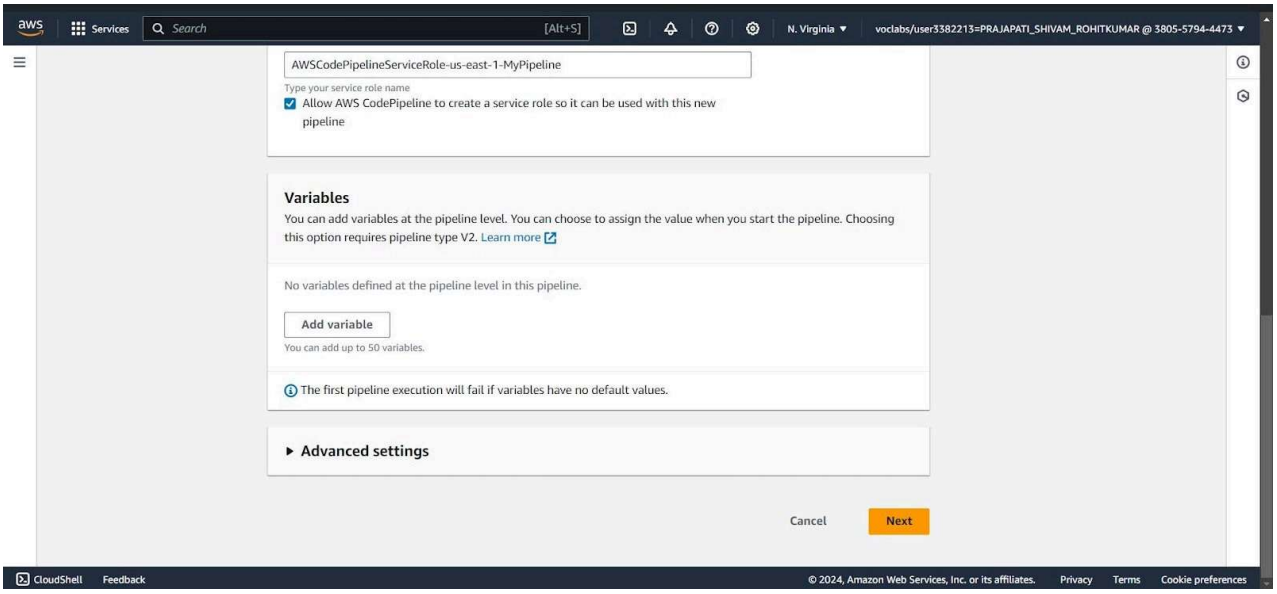


Step 9: Click on Create Pipeline.

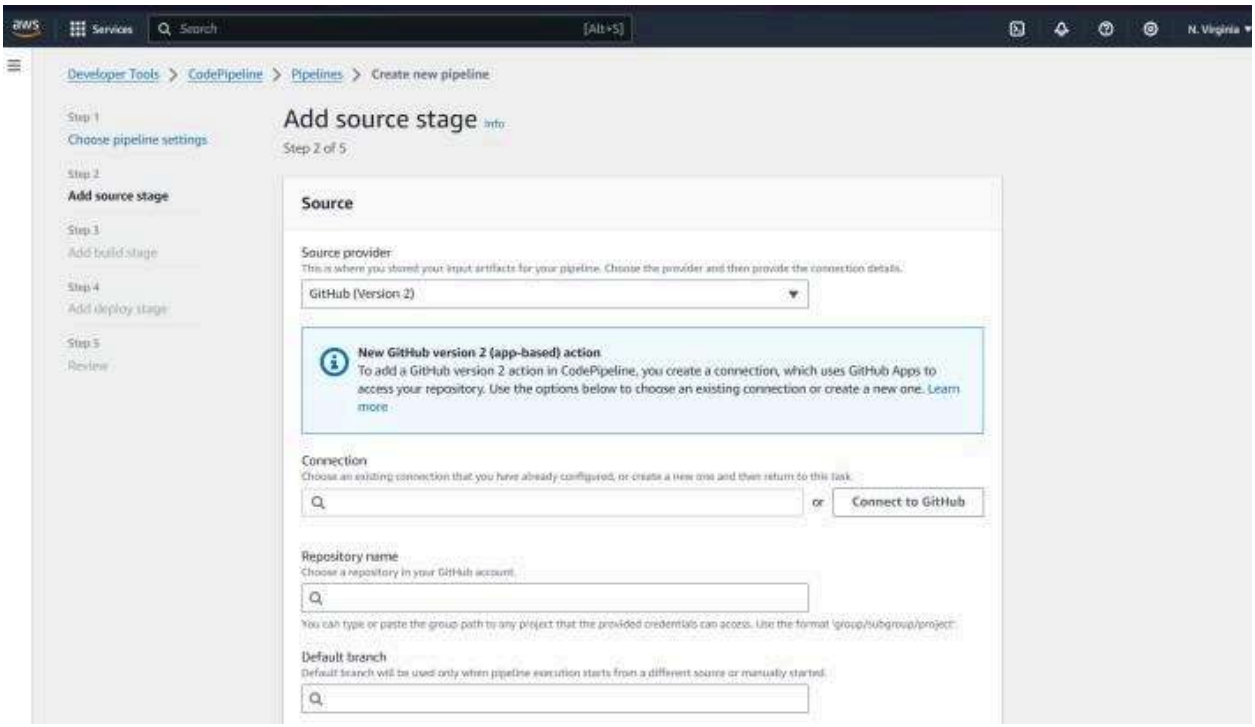


Step 10: Give a name to your Pipeline. A new service role would be created with the name of the pipeline.

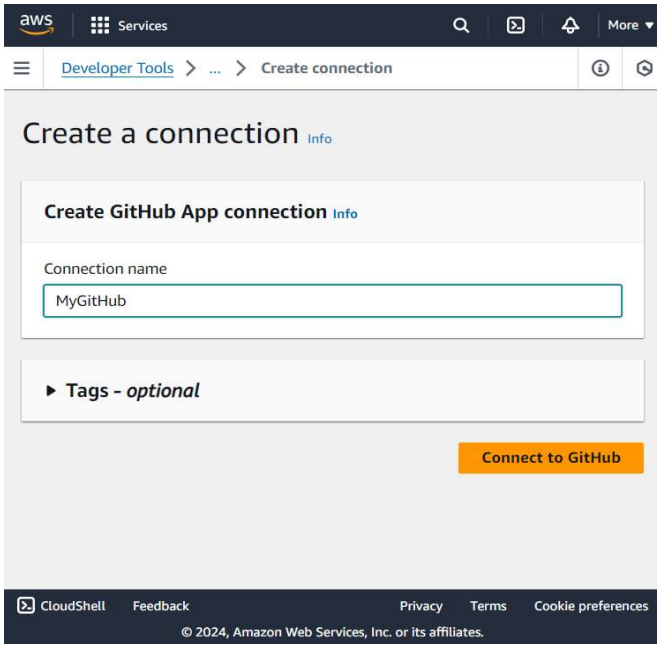




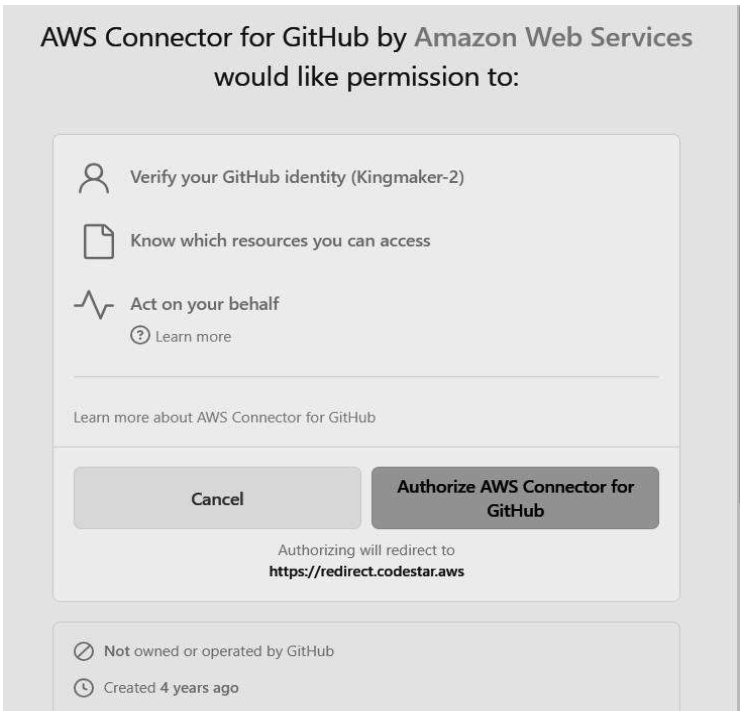
Step 11: Select a source provider (as GitHub Version (2)). Click on connect to Github (*This part have to be done in the personal account of aws as in academy account it wont allow you to create pipeline with github version 1 or 2*)



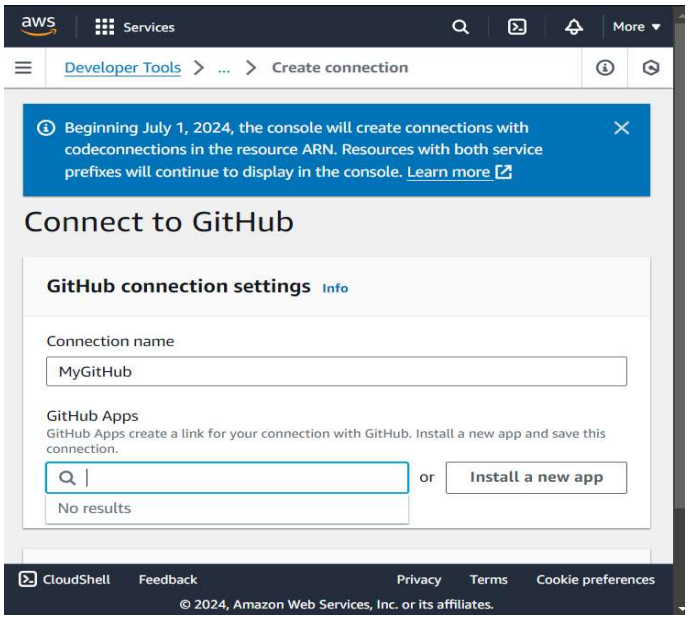
Step 12: Give a name to your GitHub app Connection and click on Connect. This will give you a prompt to either to select a GitHub app or to install a new app. If it is your first time, click on Install a new app.



Name:Aditya Dubey **Div:D15C** **Roll No:10**

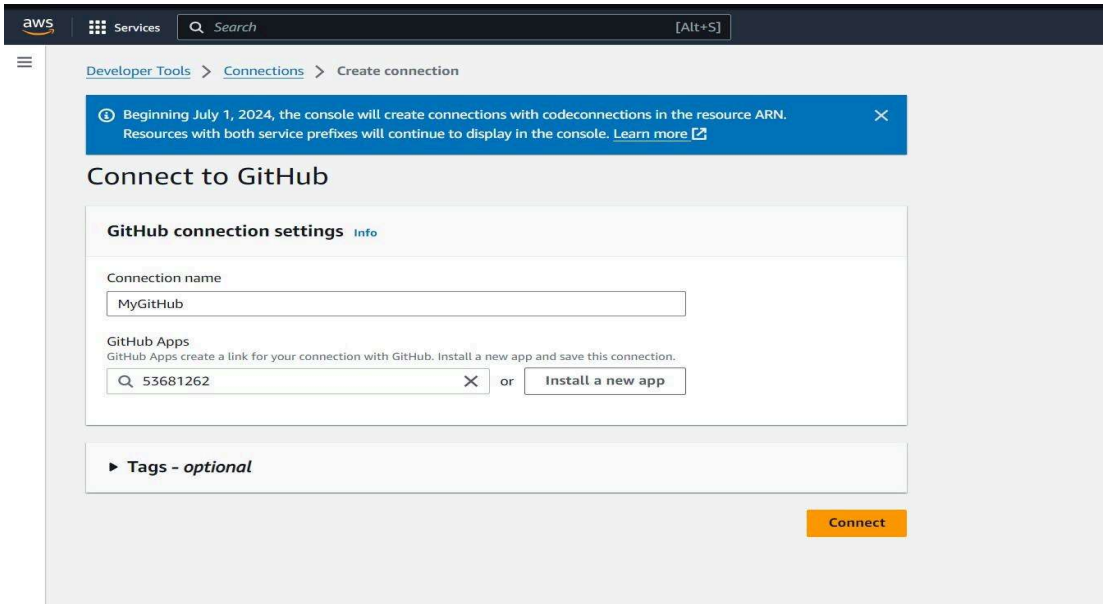


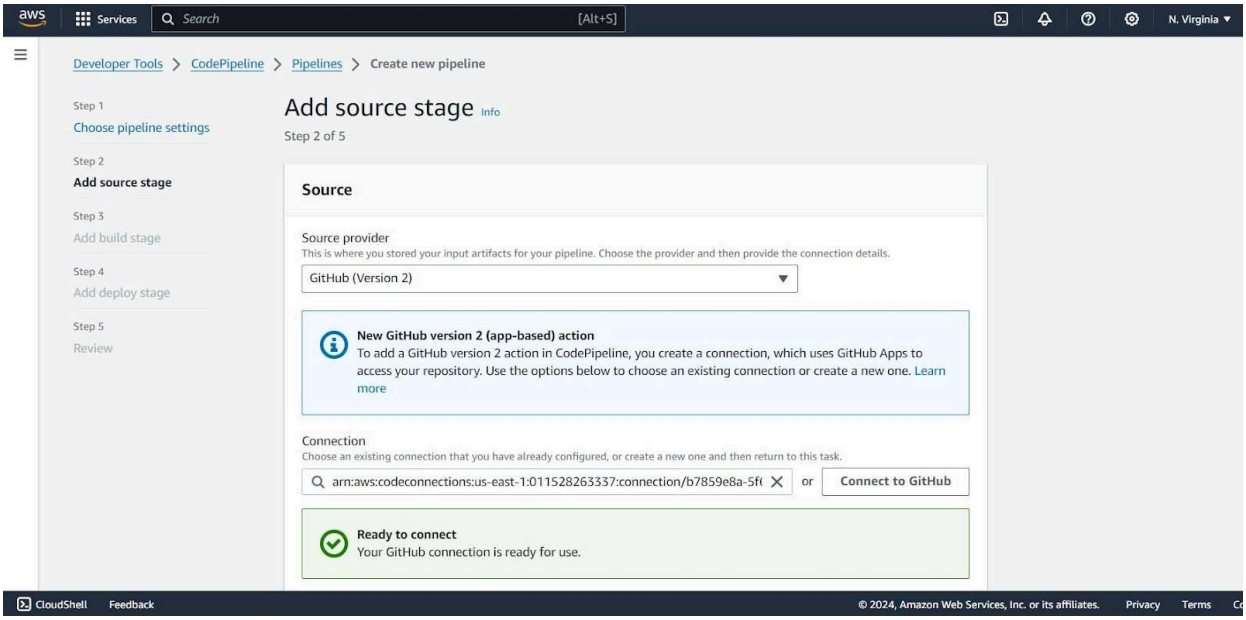
(Click on Authorize)



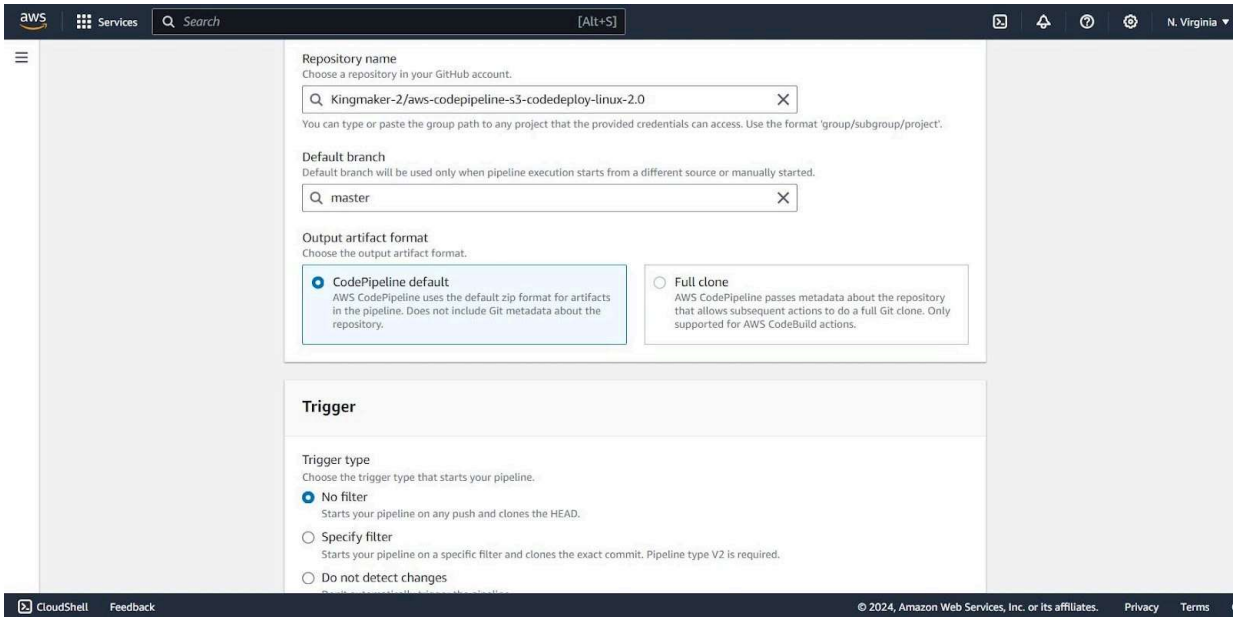
Step 13: This will direct you to install AWS connector on your GitHUb .Install it to your account and give it its permissions

Step 14: After the app is set up, it gives the number in the text field. Click on Connect. After clicking on connect, the link is shown in the connection field and AWS shows that GitHub connection is ready to use.

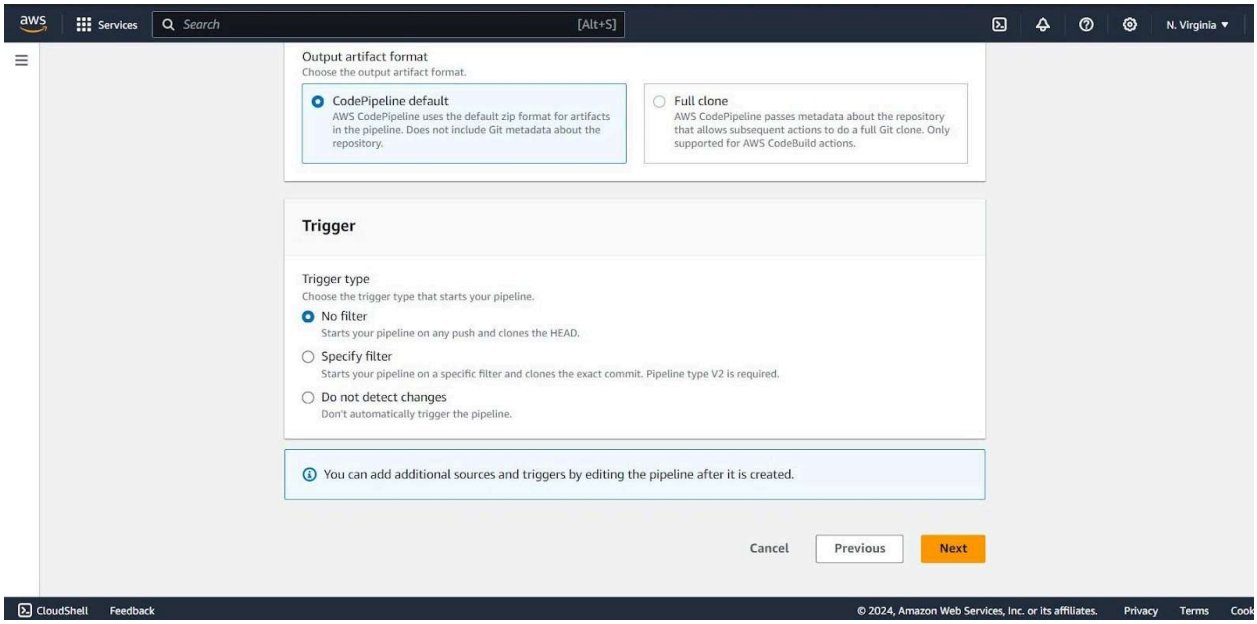




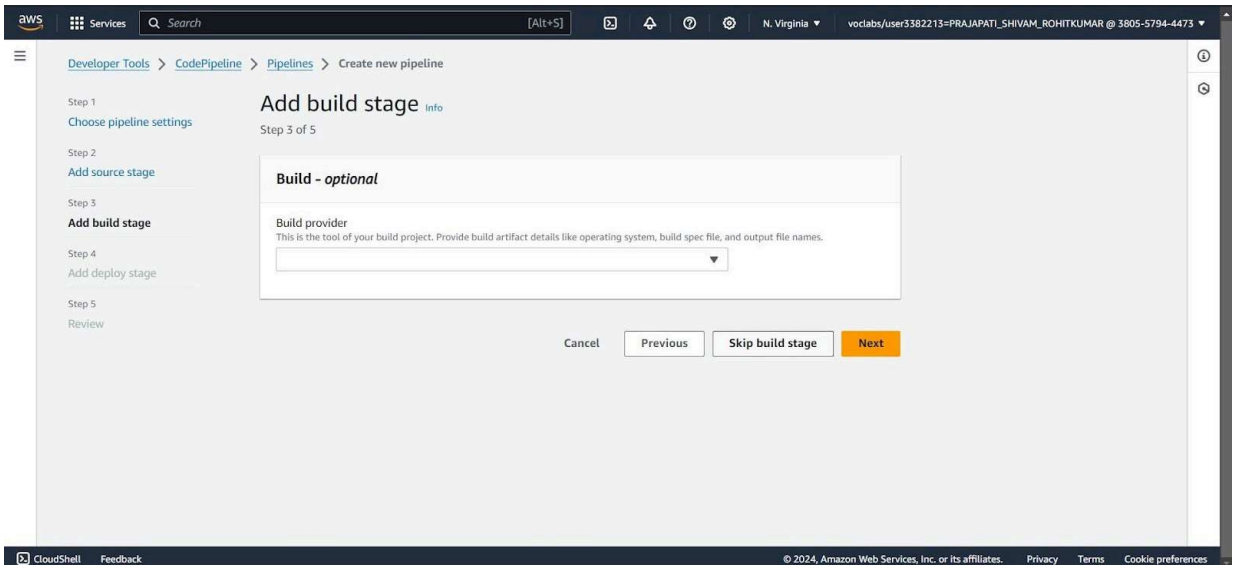
Step 15: Select the repository that you had forked to your GitHub. After that select the branch on which the files are present (default is Master).



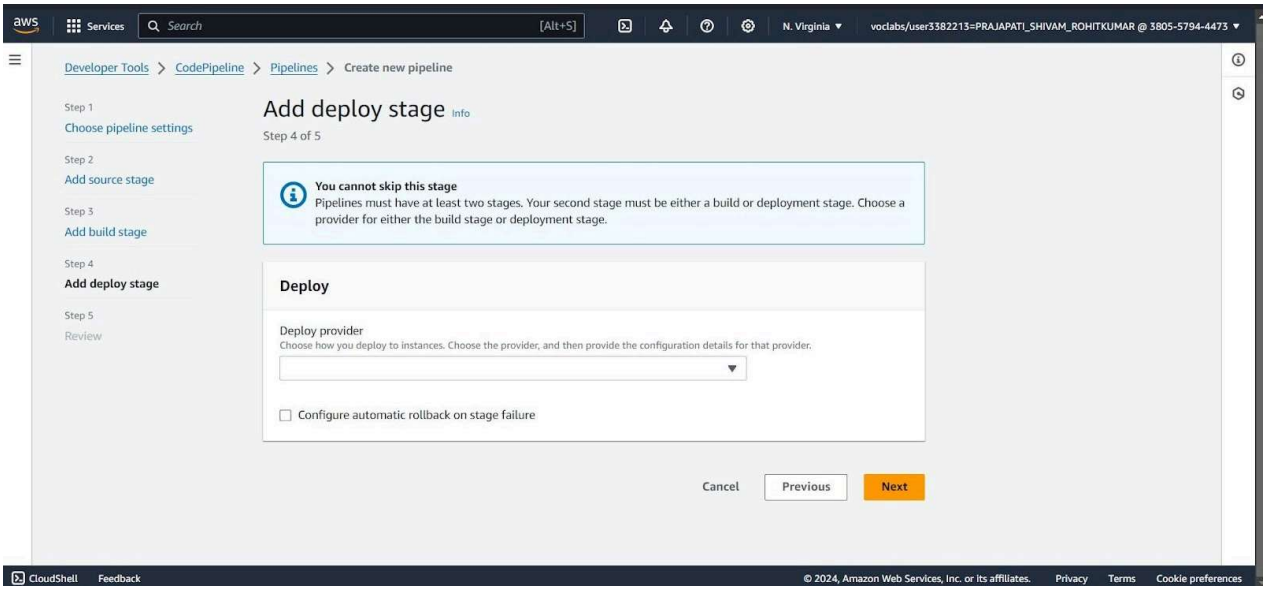
Step 16: Set the Trigger type as no filter. This would allow it to the website to update as soon as some change is made in the github.

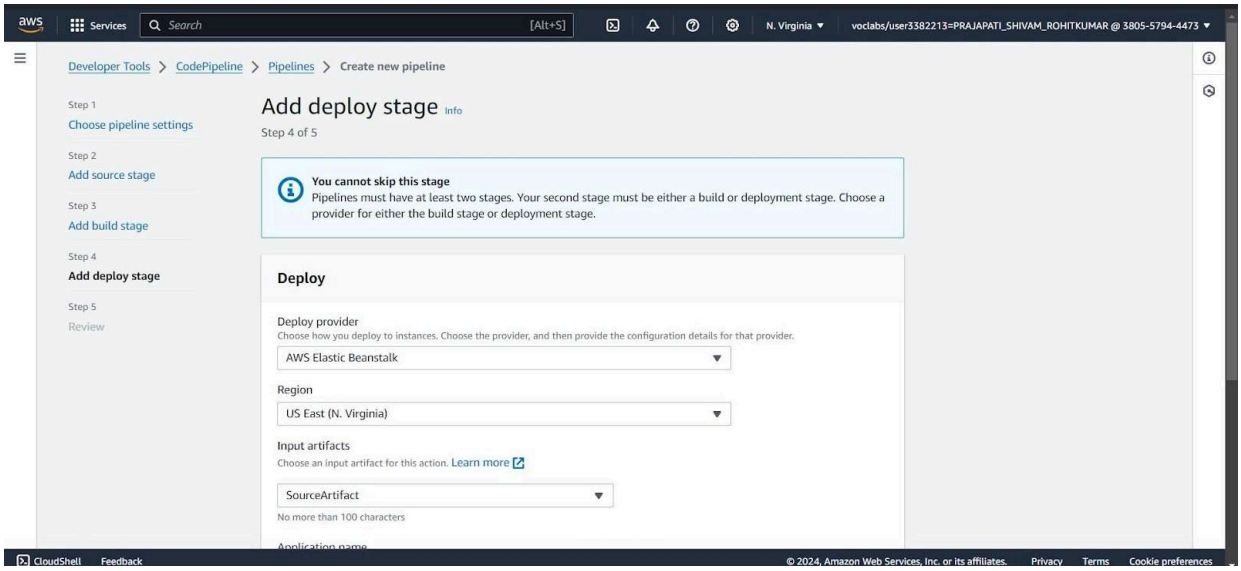


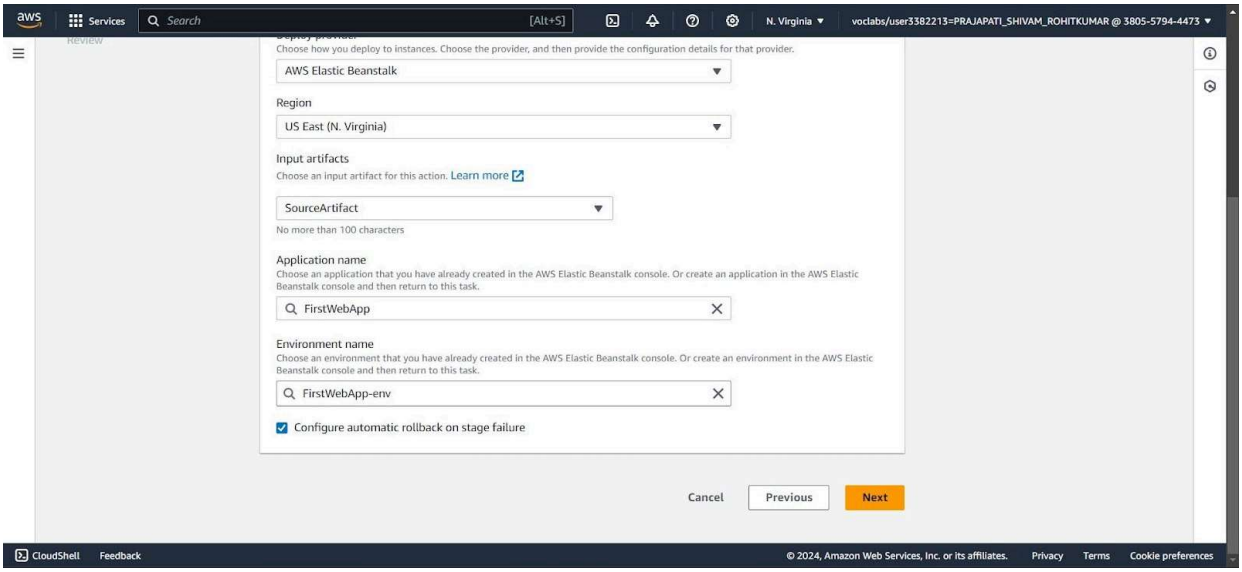
Step 17: Skip the build stage and go to Deploy. Select the deploy provider as AWS Elastic Beanstalk and Input Artifact as SourceArtifact. The application name would be the name of your Elastic Beanstalk. Then click on next.



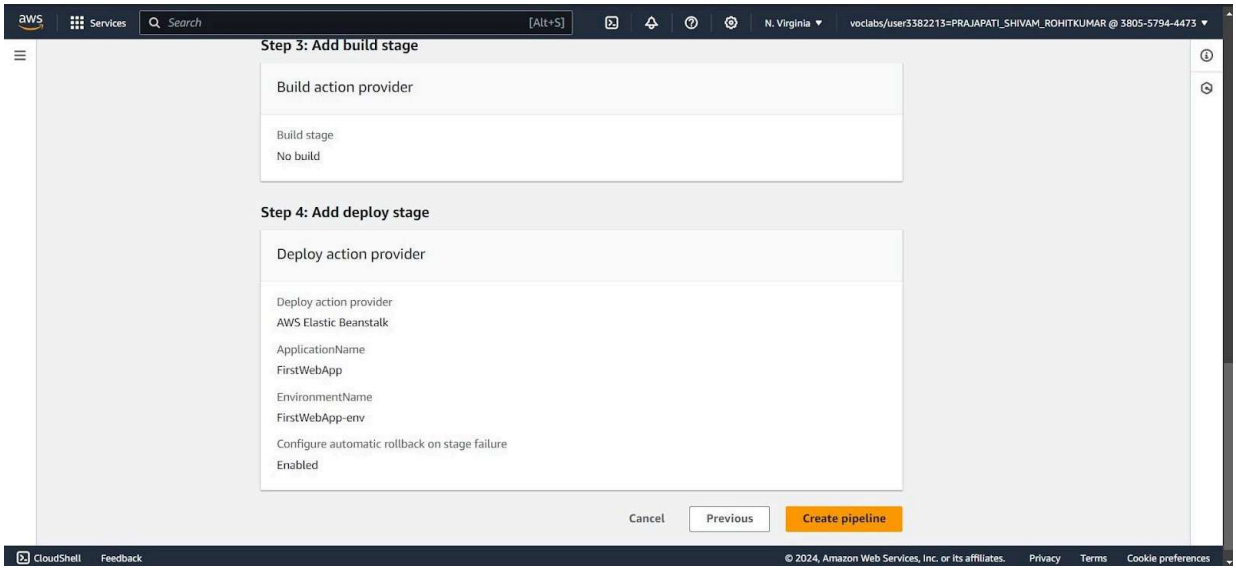
Name:Aditya Dubey **Div:D1 5C** **Roll No:10**



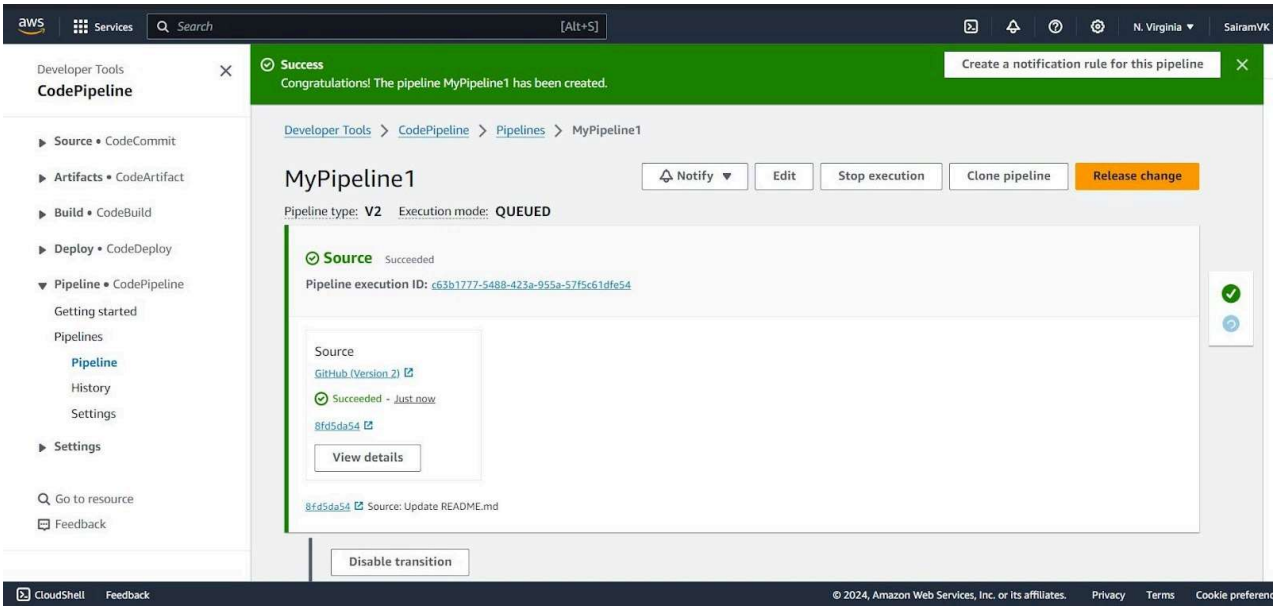


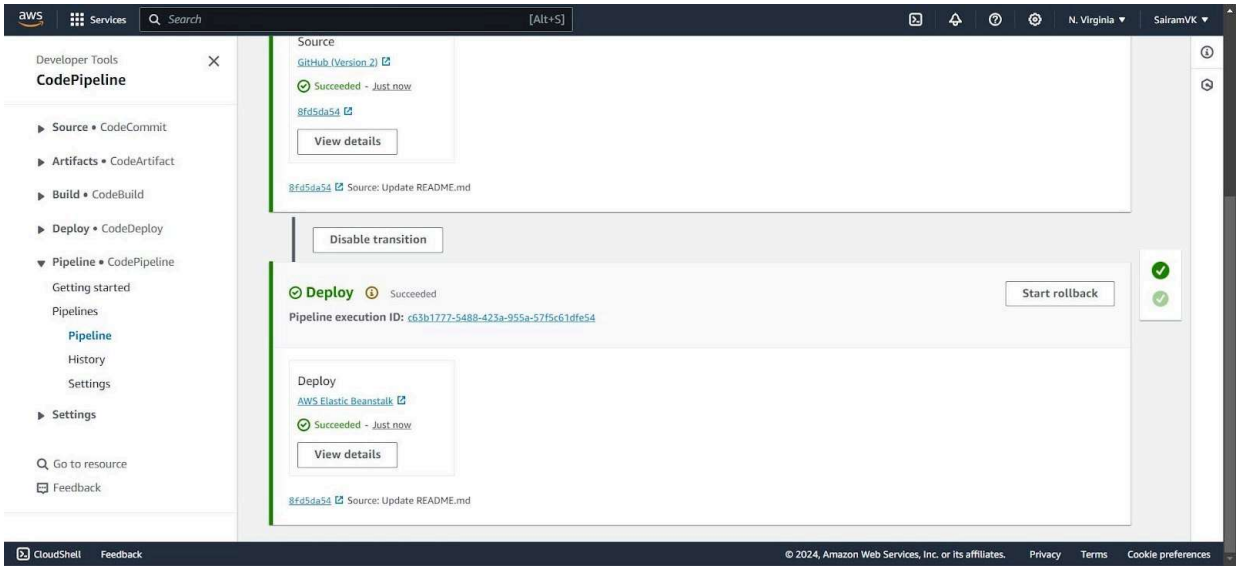


Step 18: Check all the information and click on create Pipeline

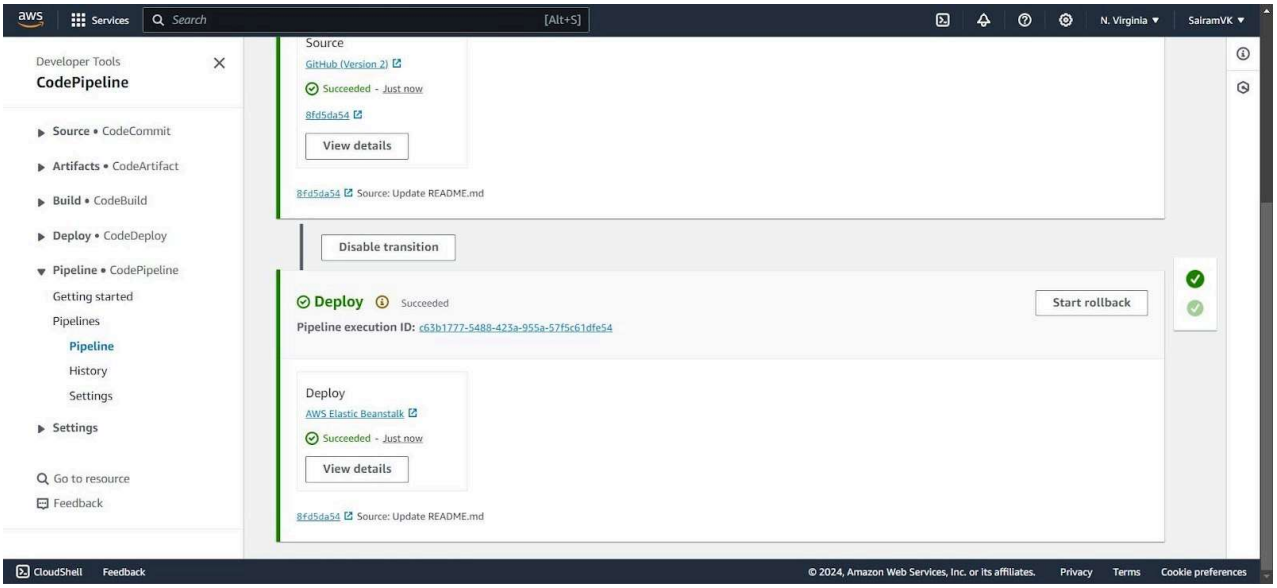


Step 19: If the pipeline is successfully deployed, this screen comes up where the source is set up and then it is transitioned to deploy

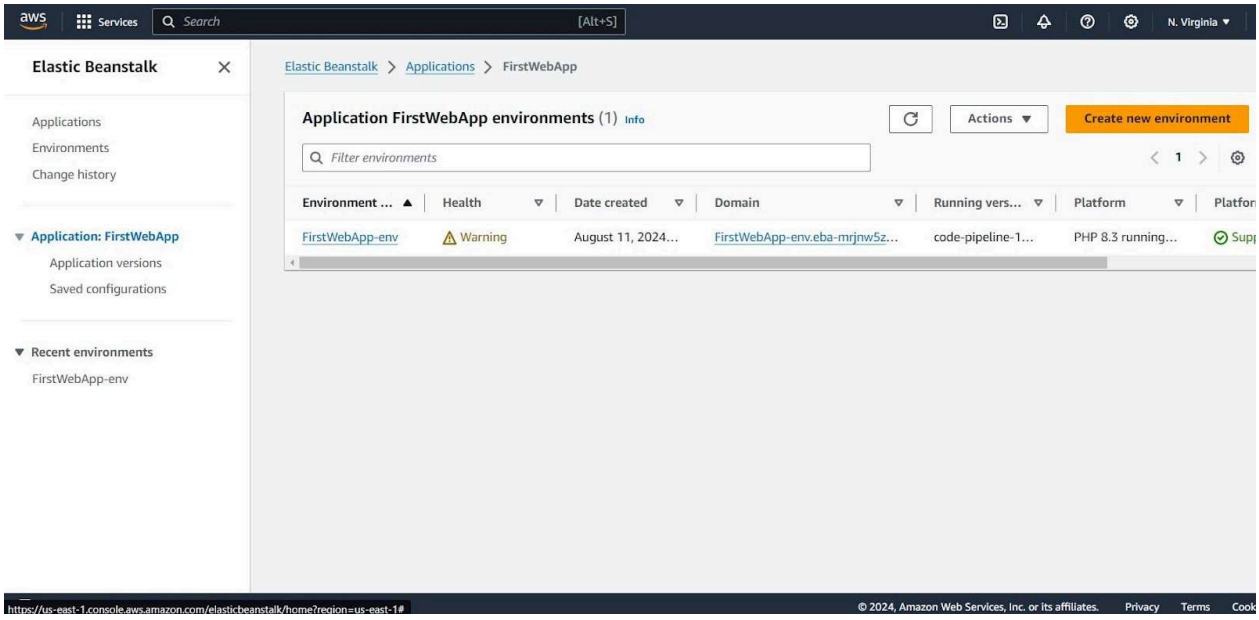




Step 20: Once the deployment is complete, click on the AWS Elastic Beanstalk under Deploy.

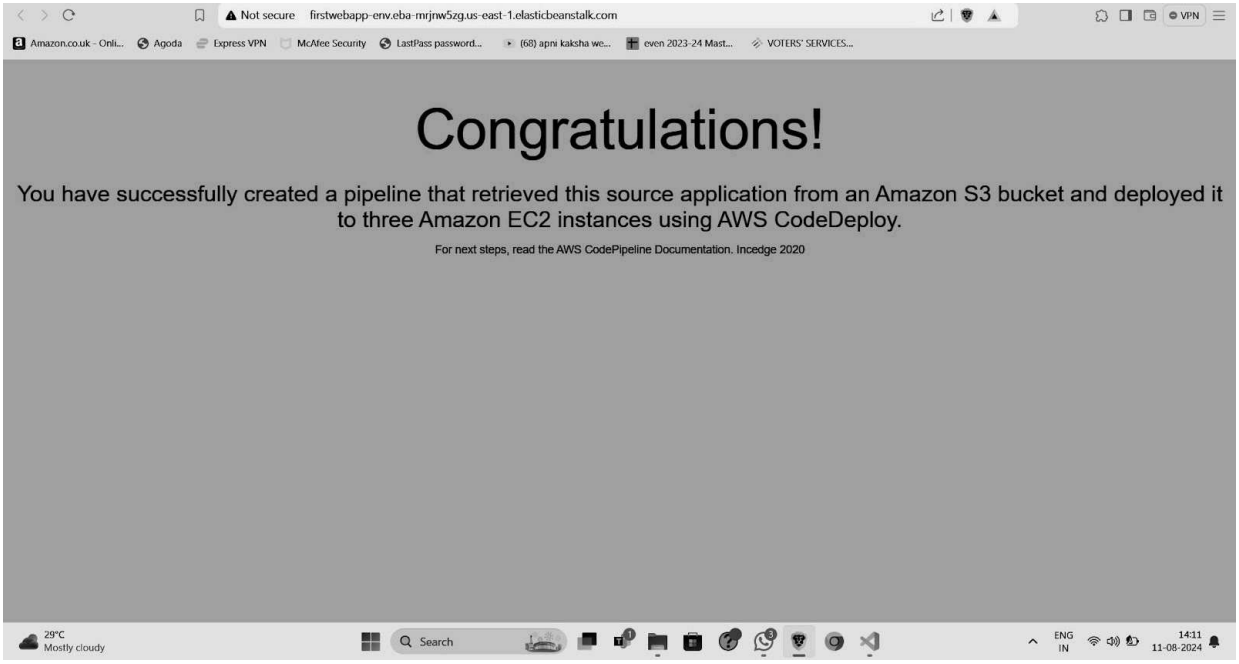


Step 21: This will redirect you to the application screen of Elastic Beanstalk. Click on the link shown under Domain

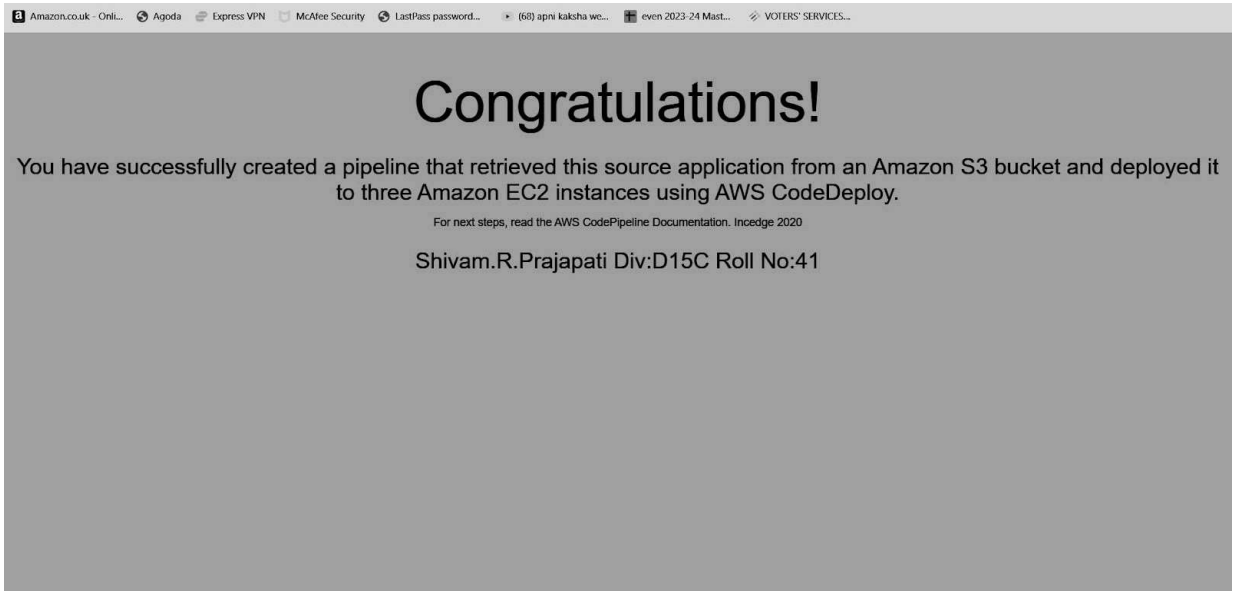


Step 22: This will successfully show the sample website hosted.

Name:Aditya Dubey **Div:D15C** **Roll No:10**



Step 23: Now, we make some changes to the index.html file in the github. For eg: If you make some changes to the tag .Once the changes are committed ,when the website is refreshed ,the
changes will be seen.



Name:Aditya Dubey Div:D15C Roll no:10

EXPERIMENT NO. 3

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

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

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Z
worker-2	i-03e737ce896f3e8ef	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1b
master	i-0650be926fe6acc7d	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1b
worker-1	i-00f42f05b3b3762ed	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1b

Set up Docker

Kubernetes requires a CRI-compliant container engine runtime such as Docker, containerd, or CRI-O. This article shows you how to deploy Kubernetes using Docker.

Install Docker on each server node by executing the steps below:

1. Update the package list:

```
sudo apt update
```

```
ubuntu@ip-172-31-92-255:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:7 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [351 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Components [35.0 kB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 c-n-f Metadata [8328 B]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [502 kB]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [123 kB]
```

2. Install Docker with the following command:

```
sudo apt install docker.io -y
```

```
root@ip-172-31-92-237:/home/ubuntu# 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 docke
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 130 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 1.7.1-1ubun
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 runc amd64 1.1.12-0ubu
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 containerd amd64 1.7.1
```

- 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

```
root@ip-172-31-92-237:/home/ubuntu# cd /etc/docker
root@ip-172-31-92-237:/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"
}
root@ip-172-31-92-237:/etc/docker#
```

- sudo systemctl enable docker
- sudo systemctl daemon-reload

```
root@ip-172-31-92-237:/home/ubuntu# sudo systemctl start docker
root@ip-172-31-92-237:/home/ubuntu# sudo systemctl enable docker
root@ip-172-31-92-237:/home/ubuntu#
```

Install Kubernetes

Setting up Kubernetes on an Ubuntu system involves adding the Kubernetes repository to the APT sources list and installing the relevant tools. Follow the steps below to install Kubernetes on all the nodes in your cluster.

Step 1: Add Kubernetes Signing Key

Since Kubernetes comes from a non-standard repository, download the signing key to ensure the software is authentic.

On each node, use the curl command to download the key and store it in a safe place (default is */etc/apt/keyrings/*:

```
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
```

```
root@ip-172-31-92-237:/etc/docker# curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | sudo gpg --dearmor -o /etc/apt/
ring.gpg
root@ip-172-31-92-237:/etc/docker# echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:
etc/apt/sources.list.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /
root@ip-172-31-92-237:/etc/docker# sudo apt 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 http://security.ubuntu.com/ubuntu noble-security InRelease
Ign:5 https://packages.cloud.google.com/apt kubernetes-xenial InRelease
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.30/deb  InRelease [1186 B]
Err:7 https://packages.cloud.google.com/apt kubernetes-xenial Release
  404  Not Found [IP: 172.253.122.100 443]
Get:8 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.30/deb  Packages [9318 B]
Reading package lists... Done
E: The repository 'http://apt.kubernetes.io kubernetes-xenial Release' does not have a Release file.
N: Updating from such a repository can't be done securely, and is therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user configuration details.
root@ip-172-31-92-237:/etc/docker# sudo apt install kubeadm kubelet kubectl
Reading package lists... Done
Building dependency tree... Done
```

Step 2: Add Software Repositories

Kubernetes is not included in the default Ubuntu repositories. To add the Kubernetes repository to your list, enter this command on each node:

```
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
```

```
root@master-node:/etc/docker# echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /' | sudo tee /etc/
pt/sources.list.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /
root@master-node:/etc/docker# sudo apt update
```

Ensure all packages are up to date:

```
sudo apt update
```

Step 3: Install Kubernetes Tools

Each Kubernetes deployment consists of three separate tools:

- Kubeadm. A tool that initializes a Kubernetes cluster by fast-tracking the setup using community-sourced best practices.
- Kubelet. The work package that runs on every node and starts containers. The tool gives you command-line access to clusters.
- Kubectl. The command-line interface for interacting with clusters.

Execute the following commands on each server node to install the Kubernetes tools:

1. Run the install command:

```
sudo apt install kubeadm kubelet kubectl
```

```
root@ip-172-31-92-237:/etc/docker# sudo apt install kubeadm kubelet 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 130 not upgraded.
Need to get 93.5 MB of archives.
After this operation, 341 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

2. Mark the packages as held back to prevent automatic installation, upgrade, or removal:

```
sudo apt-mark hold kubeadm kubelet kubectl
```

```
root@ip-172-31-92-237:/etc/docker# sudo apt-mark hold kubeadm kubelet kubectl
kubeadm set on hold.
kubelet set on hold.
kubectl set on hold.
```

> Note: The process presented in this tutorial prevents APT from automatically updating Kubernetes. For instructions on how to update, please see the official developers' instructions.

3. Verify the installation with:

```
kubeadm version
```

```
root@master-node:/etc/docker# kubeadm version
kubeadm version: &version.Info{Major:"1", Minor:"30", GitVersion:"v1.30.5", GitCommit:"74e84a90c725047b1328ff3d589fedb1cb7a
"2024-09-12T00:17:07Z", GoVersion:"go1.22.6", Compiler:"gc", Platform:"linux/amd64"}
```

The output of the `version` command shows basic deployment information.

> Note: BMC offers balanced and affordable server instances well suited for containerized services deployment. To simplify and streamline the process, deploy Kubernetes clusters on BMC using our Rancher solution.

Deploy Kubernetes

With the necessary tools installed, proceed to deploy the cluster. Follow the steps below to make the necessary system adjustments, initialize the cluster, and join worker nodes.

Step 1: Prepare for Kubernetes Deployment

This section shows you how to prepare the servers for a Kubernetes deployment. Execute the steps below on each server node:

1. Disable all swap spaces with the swapoff command:

```
sudo swapoff -a
```

Then use the sed command below to make the necessary adjustments to the */etc/fstab* file:

```
sudo sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab
```

```
root@ip-172-31-92-237:/home/ubuntu# sudo swapoff -a
root@ip-172-31-92-237:/home/ubuntu# sed -i '/ swap / s/^/#/' /etc/fstab
```

2. Load the required containerd modules. Start by opening the containerd configuration file in a text editor, such as nano:

```
sudo nano /etc/modules-load.d/containerd.conf
```

```
root@ip-172-31-92-237:/etc/docker# sudo nano /etc/modules-load.d/containerd.conf
root@ip-172-31-92-237:/etc/docker#
```

3. Add the following two lines to the file:

```
overlay
br_netfilter
```

```
  GNU nano 7.2
overlay
br_netfilter
```

Save the file and exit.

4. Next, use the modprobe command to add the modules:

```
sudo modprobe overlay
sudo modprobe br_netfilter
```

```
root@ip-172-31-92-237:/etc/docker# sudo modprobe overlay
root@ip-172-31-92-237:/etc/docker# sudo modprobe br_netfilter
```

5. Open the kubernetes.conf file to configure Kubernetes networking:

```
sudo nano /etc/sysctl.d/kubernetes.conf
```

```
root@ip-172-31-92-255:/etc/docker# sudo nano /etc/sysctl.d/kubernetes.conf
root@ip-172-31-92-255:/etc/docker#
```

6. Add the following lines to the file:

```
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
```

```
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
```

Save the file and exit.

7. Reload the configuration by typing:

```
sudo sysctl --system
```

```
root@ip-172-31-92-237:/etc/docker# sudo sysctl --system
* Applying /usr/lib/sysctl.d/10-apparmor.conf ...
* Applying /etc/sysctl.d/10-console-messages.conf ...
* Applying /etc/sysctl.d/10-ipv6-privacy.conf ...
* Applying /etc/sysctl.d/10-kernel-hardening.conf ...
* Applying /etc/sysctl.d/10-magic-sysrq.conf ...
* Applying /etc/sysctl.d/10-map-count.conf ...
* Applying /etc/sysctl.d/10-network-security.conf ...
* Applying /etc/sysctl.d/10-ptrace.conf ...
* Applying /etc/sysctl.d/10-zeropage.conf ...
* Applying /etc/sysctl.d/50-cloudimg-settings.conf ...
* Applying /usr/lib/sysctl.d/50-pid-max.conf ...
* Applying /etc/sysctl.d/99-cloudimg-ipv6.conf ...
* Applying /usr/lib/sysctl.d/99-protect-links.conf ...
* Applying /etc/sysctl.d/99-sysctl.conf ...
* Applying /etc/sysctl.d/kubernetes.conf ...
* Applying /etc/sysctl.conf ...
kernel.apparmor_restrict_unprivileged_userns = 1
kernel.printk = 4 4 1 7
net.ipv6.conf.all.use_tempaddr = 2
net.ipv6.conf.default.use_tempaddr = 2
kernel.kptr_restrict = 1
kernel.sysrq = 176
```

Step 2: Assign Unique Hostname for Each Server Node

1. Decide which server will be the master node. Then, enter the command on that node to name it accordingly:

```
sudo hostnamectl set-hostname master-node
```

```
net.ipv4.ip_forward = 1
root@ip-172-31-92-237:/etc/docker# sudo hostnamectl set-hostname master-node
root@ip-172-31-92-237:/etc/docker#
```

2. Next, set the hostname on the first worker node by entering the following command:

```
sudo hostnamectl set-hostname worker01 and worker02
```

```
root@ip-172-31-86-115:/etc/docker# sudo hostnamectl set-hostname worker01
root@ip-172-31-86-115:/etc/docker#
```

```
net.ipv4.ip_forward = 1
root@ip-172-31-92-255:/etc/docker# sudo hostnamectl set-hostname worker02
root@ip-172-31-92-255:/etc/docker#
```

3. Edit the hosts file on each node by adding the IP addresses and hostnames of the servers that will be part of the cluster.

```
root@ip-172-31-92-237:/etc/docker# sudo nano /etc/hosts
root@ip-172-31-92-237:/etc/docker#
```

```
  GNU nano 7.2
127.0.0.1 localhost
3.89.108.169 master-node
44.202.47.176 worker01
34.238.119.65 worker02
# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

4. Restart the terminal application to apply the hostname change.

Step 3: Initialize Kubernetes on Master Node

Once you finish setting up hostnames on cluster nodes, switch to the master node and follow the steps to initialize Kubernetes on it:

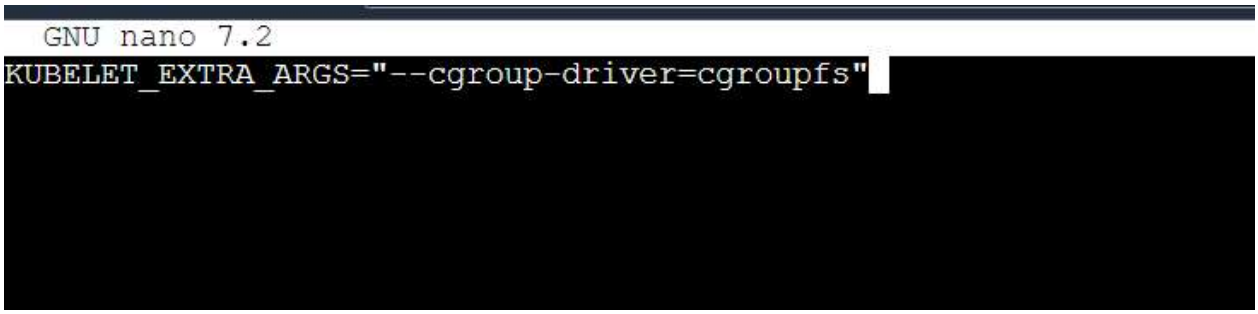
1. Open the kubelet file in a text editor.

```
sudo nano /etc/default/kubelet
```

```
root@ip-172-31-92-237:/etc/docker# sudo nano /etc/hosts
root@ip-172-31-92-237:/etc/docker# sudo nano /etc/default/kubelet
root@ip-172-31-92-237:/etc/docker#
```

2. Add the following line to the file:

```
KUBELET_EXTRA_ARGS="--cgroup-driver=cgroupfs"
```



Save and exit.

3. Reload the configuration and restart the kubelet:

```
sudo systemctl daemon-reload && sudo systemctl restart kubelet
```

```
root@ip-172-31-92-237:/# sudo systemctl daemon-reload && sudo systemctl restart kubelet
root@ip-172-31-92-237:/#
```

6. Open the kubeadm configuration file:

```
sudo nano /etc/systemd/system/kubelet.service.d/10-kubeadm.conf
```

```
root@ip-172-31-92-237:/# sudo nano /usr/lib/systemd/system/kubelet.service.d/10-kubeadm.conf
root@ip-172-31-92-237:/#
```

7. Add the following line to the file:

```
Environment="KUBELET_EXTRA_ARGS=--fail-swap-on=false"
```

```
  GNU nano 7.2                                  /usr/lib/systemd/system/kubelet.service.d/10-kubeadm.conf *
# Note: This dropin only works with kubeadm and kubelet v1.11+
[Service]
Environment="KUBELET_KUBECONFIG_ARGS=--bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf --kubeconfig=/etc/kubernet
Environment="KUBELET_CONFIG_ARGS=--config=/var/lib/kubelet/config.yaml"
Environment="KUBELET_EXTRA_ARGS=--fail-swap-on=false"
# This is a file that "kubeadm init" and "kubeadm join" generates at runtime, populating the KUBELET_KUBEADM_ARGS variable dy
EnvironmentFile=-/var/lib/kubelet/kubeadm-flags.env
# This is a file that the user can use for overrides of the kubelet args as a last resort. Preferably, the user should use
# the .NodeRegistration.KubeletExtraArgs object in the configuration files instead. KUBELET_EXTRA_ARGS should be sourced from
EnvironmentFile=-/etc/default/kubelet
ExecStart=
ExecStart=/usr/bin/kubelet $KUBELET_KUBECONFIG_ARGS $KUBELET_CONFIG_ARGS $KUBELET_KUBEADM_ARGS $KUBELET_EXTRA_ARGS
```

Save the file and exit.

9. Reload the configuration and restart the kubelet:

```
sudo systemctl daemon-reload && sudo systemctl restart kubelet
```

```
root@ip-172-31-92-237:/# sudo systemctl daemon-reload && sudo systemctl restart kubelet
root@ip-172-31-92-237:/#
```

10. Finally, initialize the cluster by typing:

```
sudo kubeadm init --control-plane-endpoint=master-node
--upload-certs —-ignore-preflight-errors=all
```

```
root@master-node:/etc/docker# sudo kubeadm init --control-plane-endpoint=master-node --upload-certs --ignore-preflight-errors=all
I0914 09:48:13.230610    9946 version.go:256] remote version is much newer: v1.31.0; falling back to: stable-1.30
[init] Using Kubernetes version: v1.30.4
[preflight] Running pre-flight checks
        [WARNING NumCPU]: the number of available CPUs 1 is less than the required 2
        [WARNING Mem]: the system RAM (957 MB) is less than the minimum 1700 MB
        [WARNING Port-6443]: Port 6443 is in use
        [WARNING Port-10259]: Port 10259 is in use
        [WARNING Port-10257]: Port 10257 is in use
        [WARNING FileAvailable--etc-kubernetes-manifests-kube-apiserver.yaml]: /etc/kubernetes/manifests/kube-apiserver.yaml already
        [WARNING FileAvailable--etc-kubernetes-manifests-kube-controller-manager.yaml]: /etc/kubernetes/manifests/kube-controller-ma
        [WARNING FileAvailable--etc-kubernetes-manifests-kube-scheduler.yaml]: /etc/kubernetes/manifests/kube-scheduler.yaml already
        [WARNING FileAvailable--etc-kubernetes-manifests-etcd.yaml]: /etc/kubernetes/manifests/etcd.yaml already exists
        [WARNING Port-10250]: Port 10250 is in use
        [WARNING Port-2379]: Port 2379 is in use
        [WARNING Port-2380]: Port 2380 is in use
```

Once the operation finishes, the output displays a `kubeadm join` command at the bottom. Make a note of this command, as you will use it to join the worker nodes to the cluster.

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

kubeadm join master-node:6443 --token fjlof5.8fnxwt2begoiwzrf \
        --discovery-token-ca-cert-hash sha256:d79687f16bbb4c7d8c78a4c02995b1f6a906afa8aaefc31dc66695800c31aed6
root@master-node:/etc/docker# ^C
root@master-node:/etc/docker#
```

11. Create a directory for the Kubernetes cluster:

```
mkdir -p $HOME/.kube
```

```
root@master-node:/# mkdir -p $HOME/.kube
root@master-node:/#
```

12. Copy the configuration file to the directory:

```
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
```

13. Change the ownership of the directory to the current user and group using the chown command:

```
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

```
root@master-node:/etc/docker# cd ../..
root@master-node:/# mkdir -p $HOME/.kube
root@master-node:/# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
root@master-node:/# sudo chown $(id -u):$(id -g) $HOME/.kube/config
root@master-node:/# kubectl apply -f https://github.com/flannel-io/flannel/releases/latest/download/kube-flannel.yml
namespace/kube-flannel created
serviceaccount/flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
root@master-node:/# kubectl taint nodes --all node-role.kubernetes.io/control-plane-
node/master-node untainted
root@master-node:/#
```

Step 4: Deploy Pod Network to Cluster

A pod network is a way to allow communication between different nodes in the cluster. This tutorial uses the Flannel node network manager to create a pod network.

Apply the Flannel manager to the master node by executing the steps below:

1. Use kubectl to install Flannel:

```
kubectl apply -f https://github.com/flannel-io/flannel/releases/latest/download/kube-flannel.yml
```

```
root@master-node:/# kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/maste
namespace/kube-flannel unchanged
clusterrole.rbac.authorization.k8s.io/flannel unchanged
clusterrolebinding.rbac.authorization.k8s.io/flannel unchanged
serviceaccount/flannel unchanged
configmap/kube-flannel-cfg unchanged
daemonset.apps/kube-flannel-ds created
```

2. Untaint the node:

```
kubectl taint nodes --all node-role.kubernetes.io/control-plane-
```

```
root@master-node:/# kubectl taint nodes --all node-role.kubernetes.io/control-plane-
node/master-node untainted
root@master-node:/#
```

Step 5: Join Worker Node to Cluster

Repeat the following steps on each worker node to create a cluster:

1. Stop and disable AppArmor:

```
sudo systemctl stop apparmor && sudo systemctl disable apparmor
```

2. Restart containerd:

```
sudo systemctl restart containerd.service
```

```
Removed "/etc/systemd/system/sysinit.target.wants/apparmor.service".
root@worker02:/etc/docker# sudo systemctl restart containerd.service
root@worker02:/etc/docker#
```

3. Apply the kubeadm join command from Step 3 on worker nodes to connect them to the master node. Prefix the command with `sudo`:

```
sudo kubeadm join [master-node-ip]:6443 --token [token]
--discovery-token-ca-cert-hash sha256:[hash]
```

```
root@worker01:/home/ubuntu# kubeadm join master-node:6443 --token fjlof5.8fnxwt2begoiwzrf
95b1f6a906afa8aaefc31dc66695800c31aed6
```

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

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

Replace [master-node-ip], [token], and [hash] with the values from the kubeadm join command output.

4. After a few minutes, switch to the master server and enter the following command to check the status of the nodes:

```
kubectl get 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
```

The system displays the master node and the worker nodes in the cluster.

Advanced DevOps Lab

Experiment 4

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

Theory:

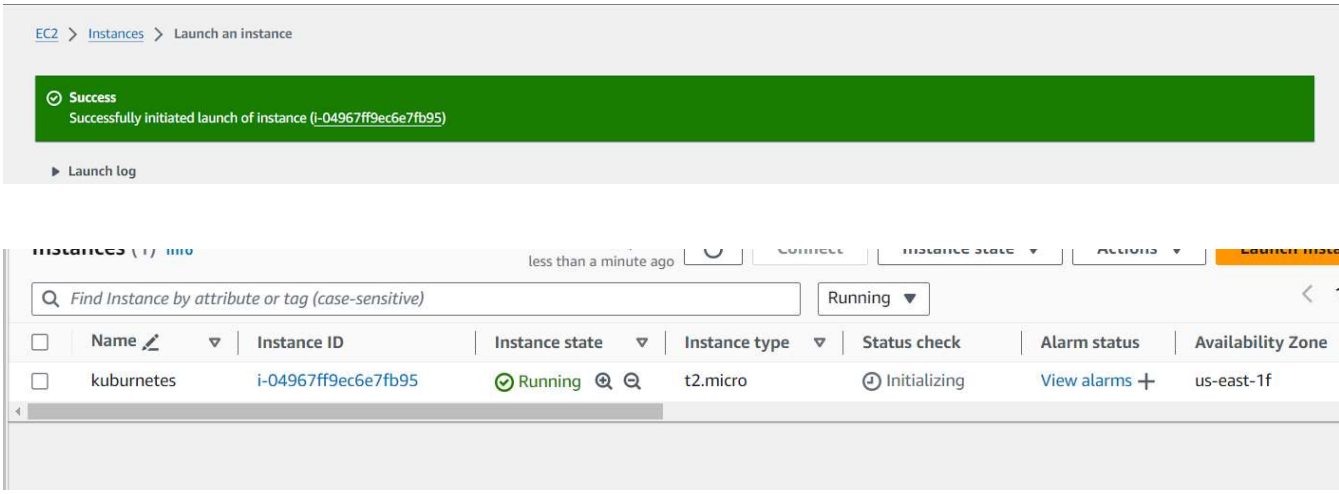
Originally developed by Google, Kubernetes is an open-source container orchestration platform designed to automate the deployment, scaling, and management of containerized applications. In fact, Kubernetes has established itself as the defacto standard for container orchestration and is the flagship project of the Cloud Native Computing Foundation (CNCF), backed by key players like Google, AWS, Microsoft, IBM, Intel, Cisco, and Red Hat.

Kubernetes Deployment

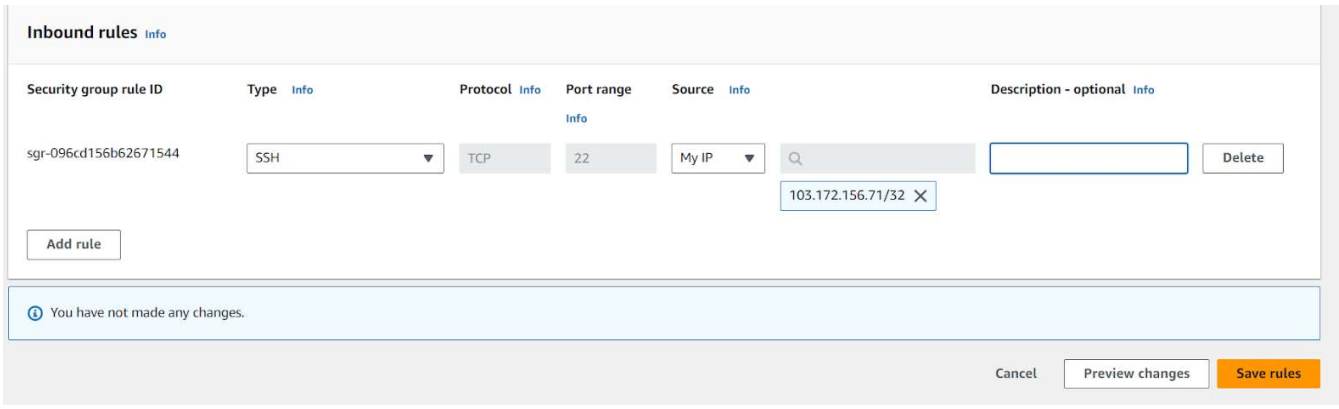
A Kubernetes Deployment is used to tell Kubernetes how to create or modify instances of the pods that hold a containerized application. Deployments can scale the number of replica pods, enable the rollout of updated code in a controlled manner, or roll back to an earlier deployment version if necessary.

Steps:

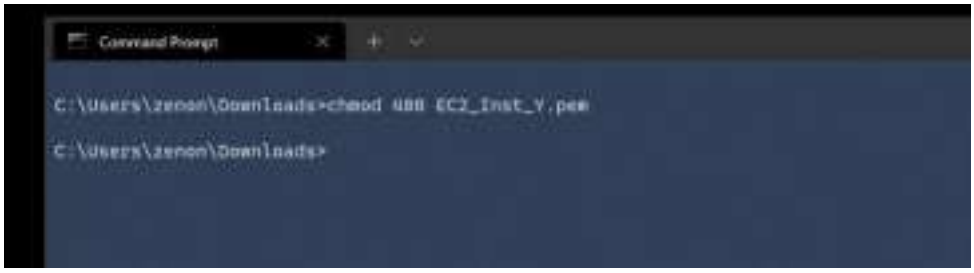
1. Create an EC2 Ubuntu Instance on AWS.



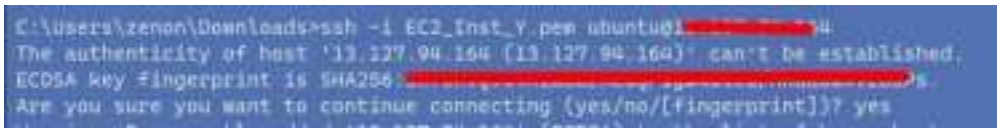
2. Edit the Security Group Inbound Rules to allow SSH



3. SSH into the machine



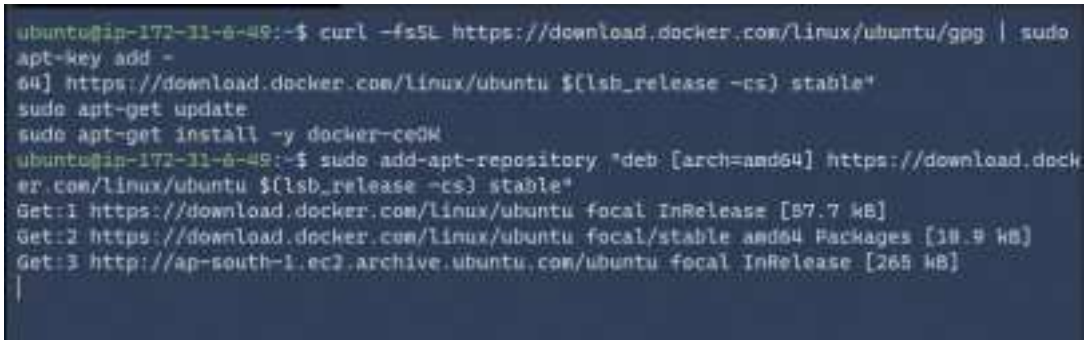
ssh -i <keyname>.pem ubuntu@<public_ip_address>



4. Install Docker

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key
add -
sudo add-apt-repository "deb [arch=amd64]
```

```
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
sudo apt-get update
sudo apt-get install -y docker-ce
```



Then, configure cgroup in a daemon.json file.

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

5. Install Kubernetes

```
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo
apt-key add -
cat << EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb https://apt.kubernetes.io/ kubernetes-xenial main
EOF
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
```

```
ubuntu@ip-172-31-8-49:~$ curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg |
 sudo apt-key add -
s.list.d/kubernetes.list
deb https://apt.kubernetes.io/ kubernetes-xenial main
EOF
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectlOK
ubuntu@ip-172-31-8-49:~$ cat << EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list
> deb https://apt.kubernetes.io/ kubernetes-xenial main
> EOF
deb https://apt.kubernetes.io/ kubernetes-xenial main
ubuntu@ip-172-31-8-49:~$ sudo apt-get update
Hit:1 https://download.docker.com/linux/ubuntu focal InRelease
0% [Waiting for headers] [Connecting to security.ubuntu.com (91.189.91.39)] [Waiting fo
```

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
```

6. Initialize the Kubecluster

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

```
ubuntu@i[redacted]0:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
```

```
Your Kubernetes control-plane has initialized successfully!

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

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

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

  export KUBECONFIG=/etc/kubernetes/admin.conf

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

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

kubeadm join [redacted]
        --discovery-token-ca-cert-hash sha256:ab3[redacted]
```

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
```

Then, add a common networking plugin called flannel as mentioned in the code.

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

7. Now that the cluster is up and running, we can deploy our nginx server on this cluster.

Apply this deployment file using this command to create a deployment

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

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

Next up, create a name alias for this pod.

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

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

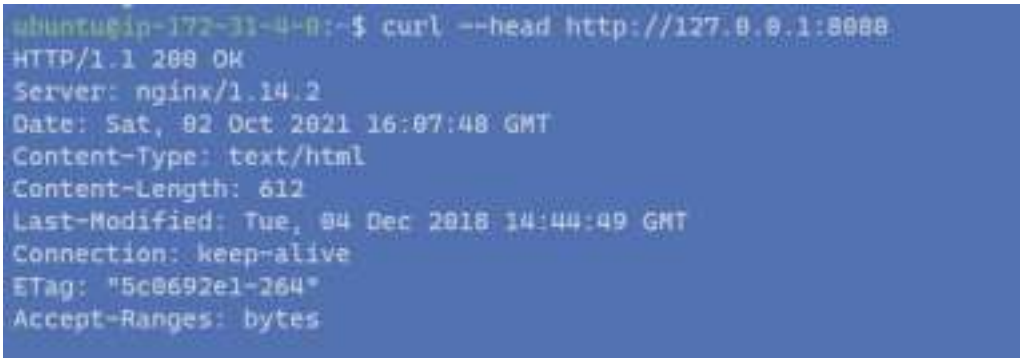
```
kubectl port-forward $POD_NAME 8080:80
```

9. Verify your deployment

Open up a new terminal and ssh to your EC2 instance.

Then, use this curl command to check if the Nginx server is running.

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



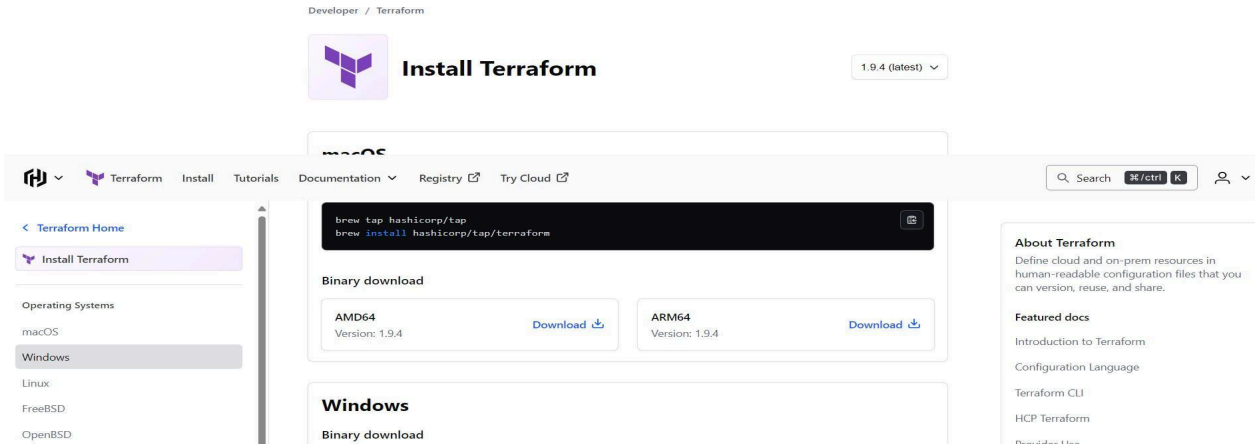
If the response is 200 OK and you can see the Nginx server name, your deployment was successful.

We have successfully deployed our Nginx server on our EC2 instance.

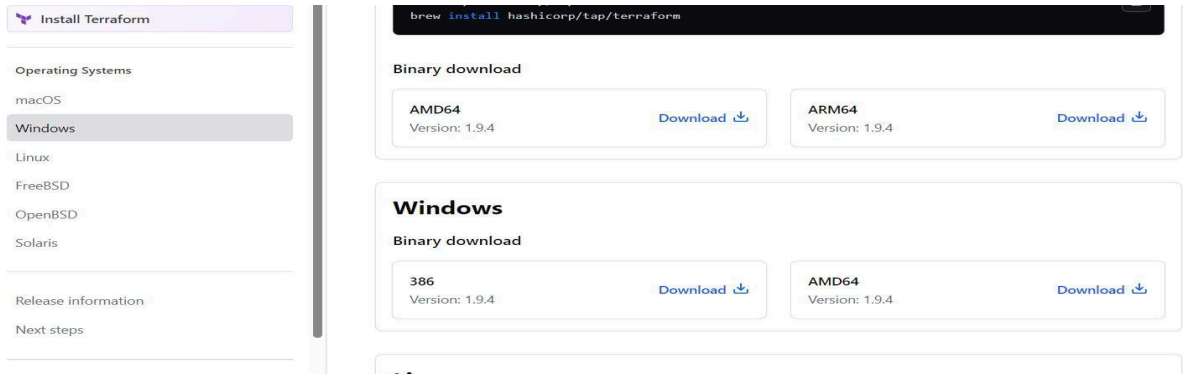
Name:Aditya Dubey **Div:D15C** **Roll no:10**

Step 1: Download terraform

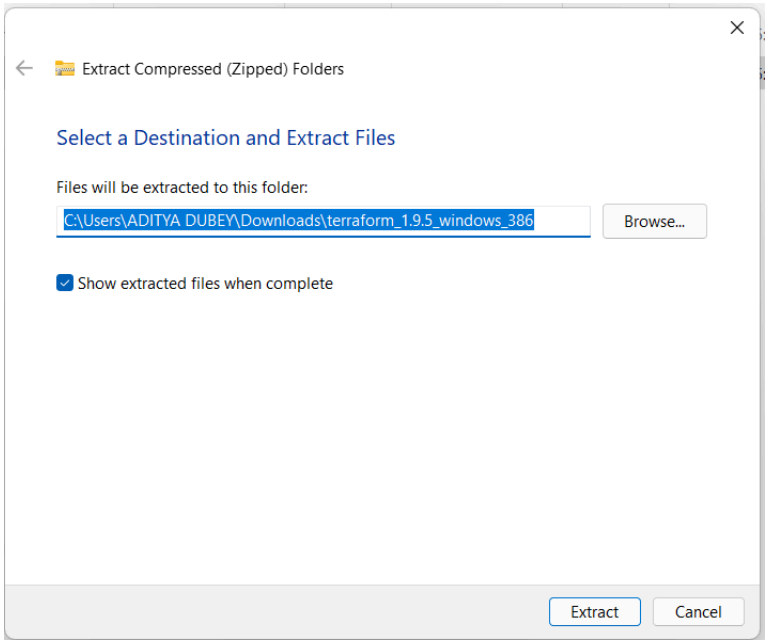
website:https://www.terraform.io/downloads.html



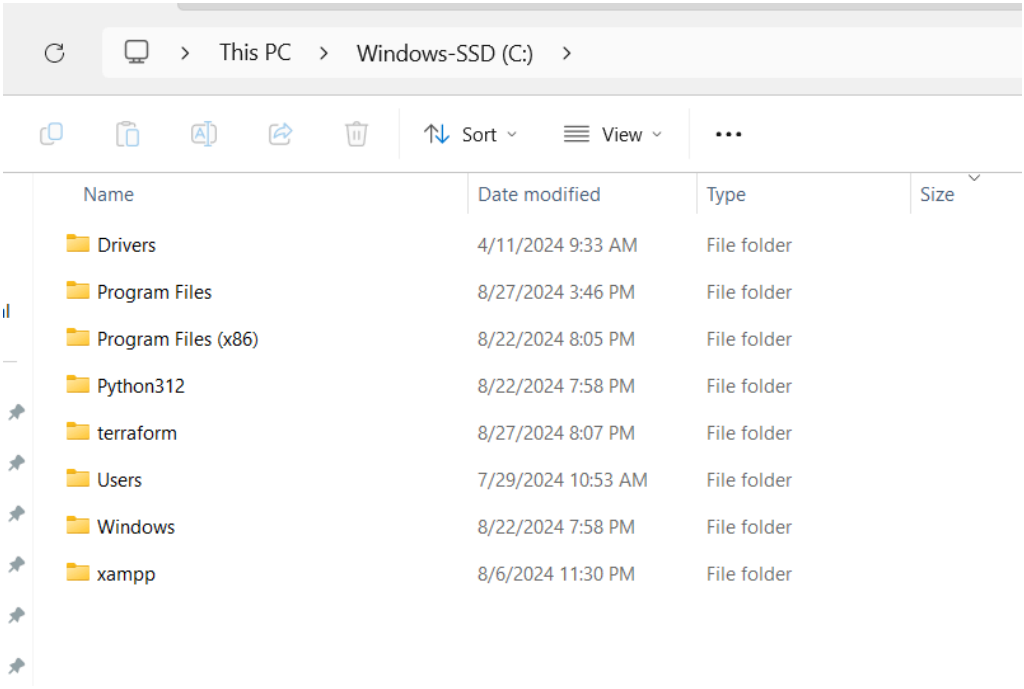
Step 2: Downlaod Windows Binary download 386



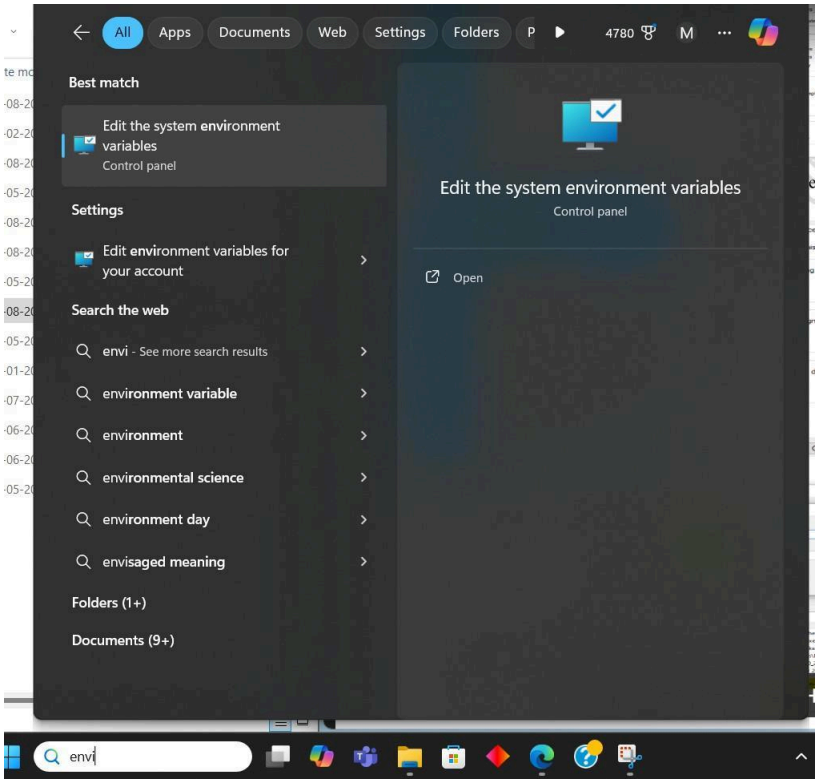
Step 3: Extract the downloaded setup file Terraform.exe in C:\Terraform directory

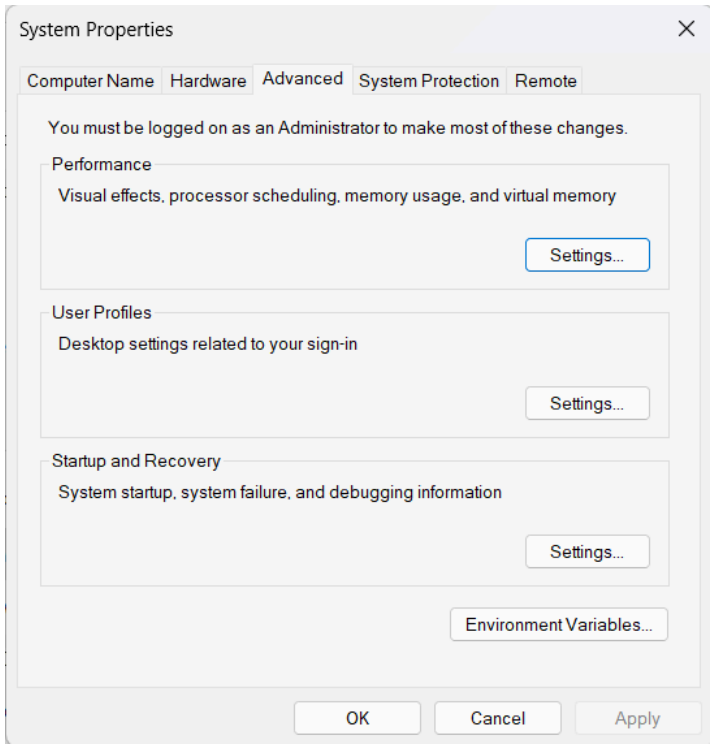


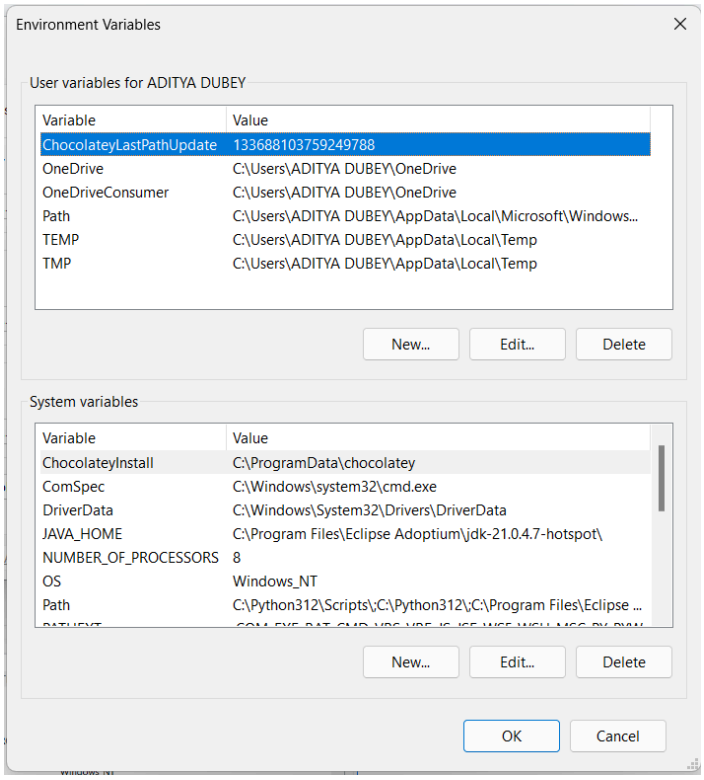
Step 4: Created folder in C: drive for terraform,

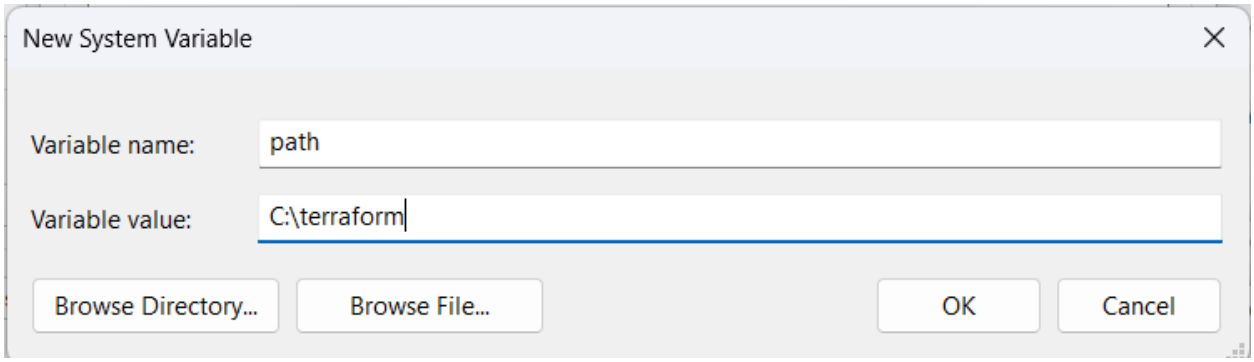


Step 5: Set the System path for Terraform in Environment Variable. Open Environment Variable.

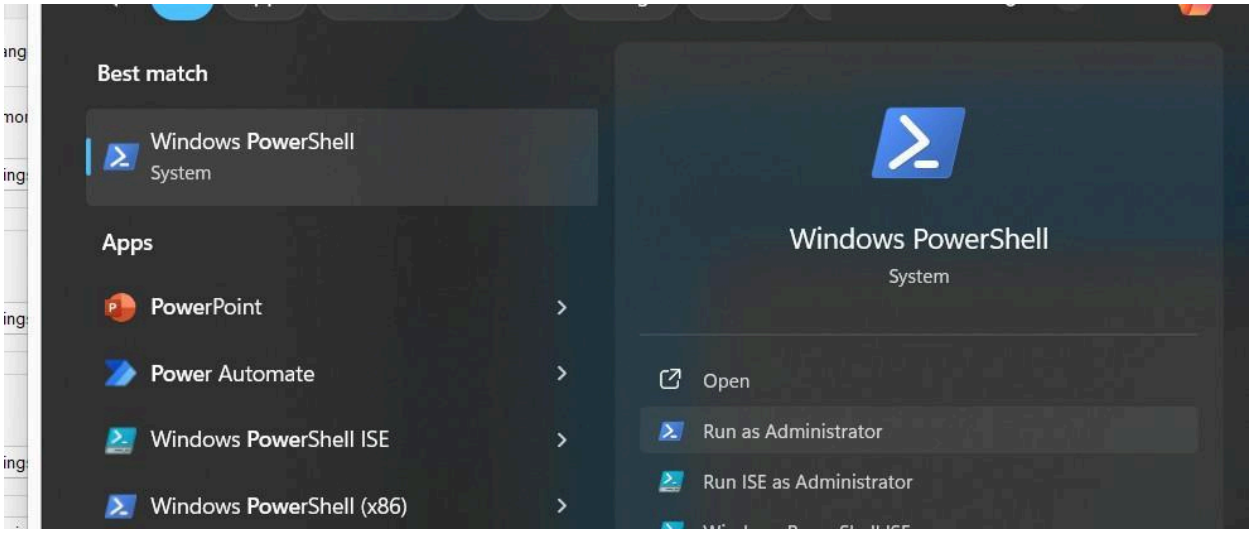



System Properties
Computer Name
Hardware
Advanced
System Protection
Remote
You must be logged on as an Administrator to make most of these changes.
Performance
Visual effects, processor scheduling, memory usage, and virtual memory
Settings...
User Profiles
Desktop settings related to your sign-in
Settings...
Startup and Recovery
System startup, system failure, and debugging information
Settings...
Environment Variables...
OK
Cancel
Apply

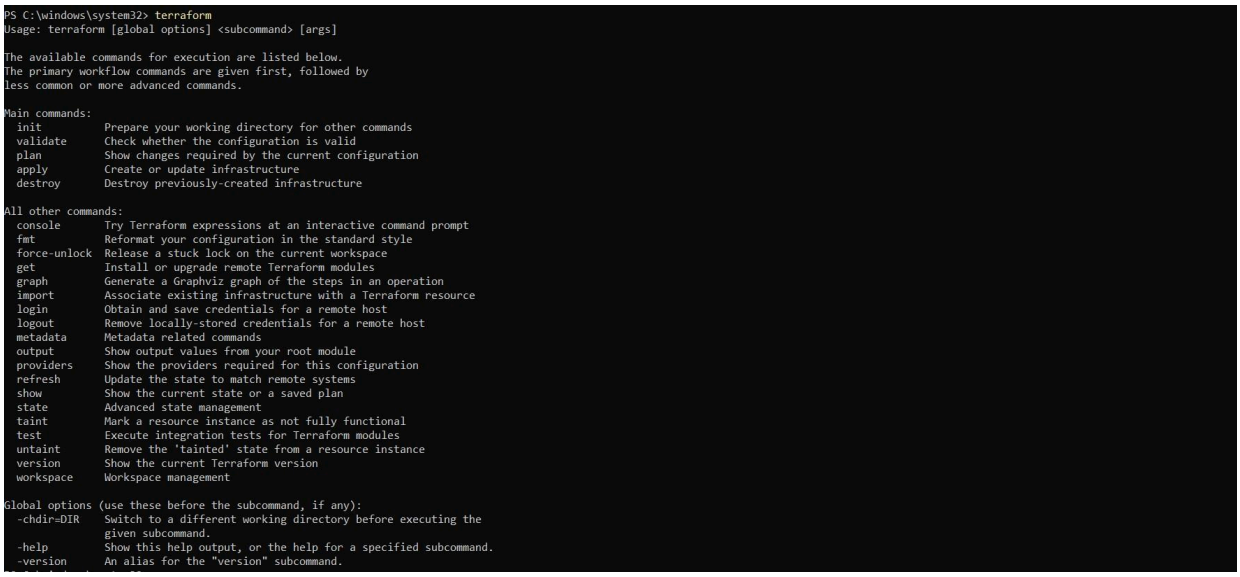

Environment Variables
User variables for ADITYA DUBEY
Variable
Value
ChocolateyLastPathUpdate
133688103759249788
OneDrive
C:\Users\ADITYA DUBEY\OneDrive
OneDriveConsumer
C:\Users\ADITYA DUBEY\OneDrive
Path
C:\Users\ADITYA DUBEY\AppData\Local\Microsoft\Windows...
TEMP
C:\Users\ADITYA DUBEY\AppData\Local\Temp
TMP
C:\Users\ADITYA DUBEY\AppData\Local\Temp
New...
Edit...
Delete
System variables
Variable
Value
ChocolateyInstall
C:\ProgramData\chocolatey
ComSpec
C:\Windows\system32\cmd.exe
DriverData
C:\Windows\System32\Drivers\DriverData
JAVA_HOME
C:\Program Files\Eclipse Adoptium\jdk-21.0.4.7-hotspot\
NUMBER_OF_PROCESSORS
8
OS
Windows_NT
Path
C:\Python312\Scripts\;C:\Python312\;C:\Program Files\Eclipse ...
New...
Edit...
Delete
OK
Cancel



Step 9: Open PowerShell with Admin Access



Step 10: Open Terraform in PowerShell and check its functionality



A. Creating docker image using terraform
Prerequisite:

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

```
C:\Users\ADITYA DUBEY>docker

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

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

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

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

C:\Users\ADITYA DUBEY>
```

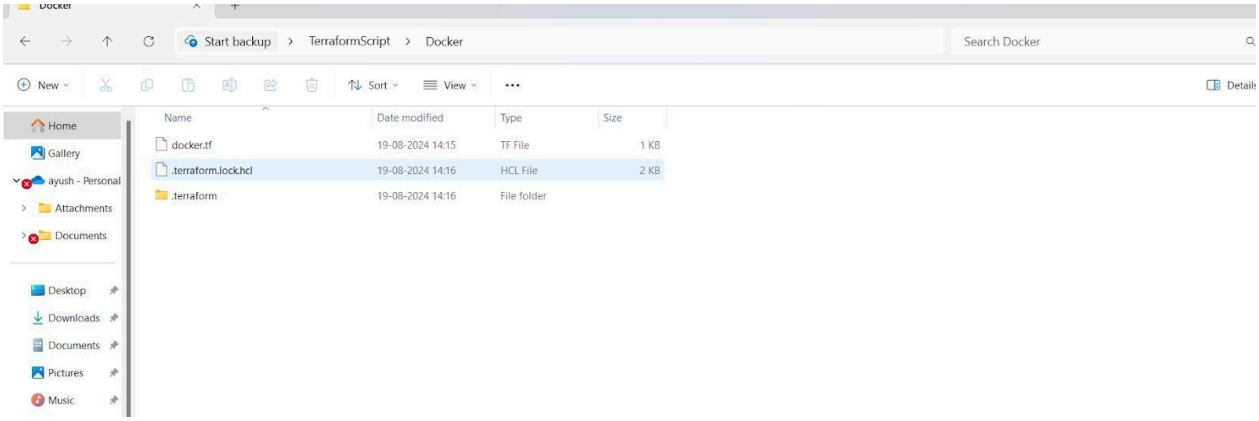
Now, create a folder named ‘Terraform Scripts’ in which we save our different types of scripts which will be further used in this experiment.

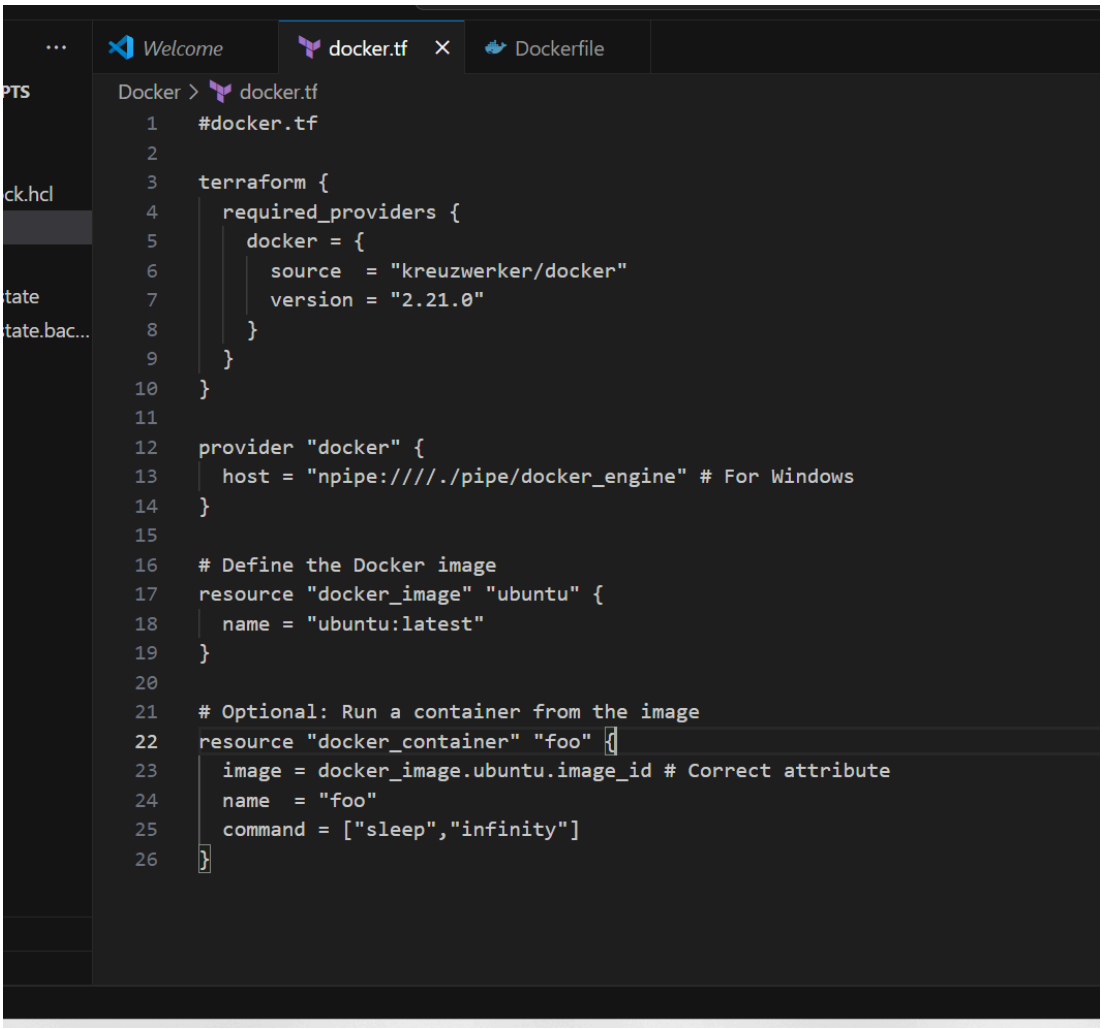
Step 2: Firstly create a new folder named ‘Docker’ in the ‘TerraformScripts’ folder. Then create a new docker.tf file using Atom editor and write the followingcontents into it to create a Ubuntu Linux container.

Script:
#docker.tf

```
#docker.tf

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

```
  }
}

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

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

# Optional: Run a container from the image
resource "docker_container" "foo" {
  image = docker_image.ubuntu.image_id # Correct attribute
  name  = "foo"
  command = ["sleep","infinity"]
}
```





Step 3: Execute Terraform Init command to initialize the resources

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

Terraform has been successfully initialized!

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

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

Step 4: Execute Terraform plan to see the available resources

```
C:\Users\ADITYA DUBEY\OneDrive\Desktop\terraformscripts\Docker>terraform plan

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

Terraform will perform the following actions:

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

      + healthcheck (known after apply)
```

Step 5: Execute Terraform apply to apply the configuration, which will automatically create and run the Ubuntu Linux container based on our configuration. Using command : “terraform apply”

```
C:\Users\ADITYA DUBEY\OneDrive\Desktop\terraformscripts\Docker>terraform plan
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]

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

Terraform will perform the following actions:

  # docker_container.foo will be created
  + resource "docker_container" "foo" {
      + attach           = false
      + bridge           = (known after apply)
      + command          = [
          + "sleep",
          + "infinity",
        ]
      + container_logs   = (known after apply)
      + entrypoint       = (known after apply)
      + env              = (known after apply)
      + exit_code        = (known after apply)
      + gateway          = (known after apply)
      + hostname         = (known after apply)
      + id               = (known after apply)
      + image            = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a"
      + init             = (known after apply)
      + ip_address       = (known after apply)
      + ip_prefix_length = (known after apply)
      + ipc_mode         = (known after apply)
      + log_driver       = (known after apply)
      + logs             = false
      + must_run         = true
      + name             = "foo"
      + network_data     = (known after apply)
      + read_only        = false
      + remove_volumes   = true
      + restart          = "no"
      + rm               = false
      + runtime          = (known after apply)
      + security_opts    = (known after apply)
      + shm_size         = (known after apply)
      + start            = true
```

```
      + stdin_open       = false
      + stop_signal      = (known after apply)
      + stop_timeout     = (known after apply)
      + tty              = false

      + healthcheck (known after apply)

      + labels (known after apply)
    }

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=25e618f8f29715a205c36e0d31bfdd4a326dc83f513f2c7779aebd492ce9f602]

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

Docker images, After Executing Apply step:

```
C:\Users\ADITYA DUBEY\OneDrive\Desktop\terraformscripts\Docker>docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
ubuntu       latest    edbfe74c41f8   3 weeks ago   78.1MB
```

Step 6: Execute Terraform destroy to delete the configuration, which will automatically delete the Ubuntu Container.

```
C:\Users\ADITYA DUBEY\OneDrive\Desktop\terraformscripts\Docker>terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Refreshing state... [id=e9d84fdf0bbada9cfa63df05a6ee74111495ea3805eb4afaef53501a0f0a9e56]

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",
          - "infinity",
        ] -> 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          = "e9d84fdf0bba" -> null
      - id                = "e9d84fdf0bbada9cfa63df05a6ee74111495ea3805eb4afaef53501a0f0a9e56" -> null
      - image             = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a" -> null
      - init              = false -> null
      - ip_address        = "172.17.0.2" -> null
      - ip_prefix_length  = 16 -> null
      - ipc_mode          = "private" -> null
      - links             = [] -> null
      - log_driver        = "json-file" -> null
      - log_opts          = {} -> null
      - logs              = false -> null
      - max_retry_count   = 0 -> null
      - memory            = 0 -> null
      - memory_swap       = 0 -> null
      - must_run          = true -> null
      - name              = "foo" -> null
      - network_data      = [
          - {
              - gateway                   = "172.17.0.1"
              - global_ipv6_prefix_length = 0
              - ip_address                = "172.17.0.2"
              - ip_prefix_length          = 16
              - network_name              = "bridge"
                # (2 unchanged attributes hidden)
            },
        ] -> null
      - network_mode      = "bridge" -> null
      - privileged        = false -> null
      - publish_all_ports = false -> null
      - read_only         = false -> null
```

```
      - rm                = false -> null
      - runtime           = "runc" -> null
      - security_opts     = [] -> null
      - shm_size          = 64 -> null
      - start             = true -> null
      - stdin_open        = false -> null
      - stop_timeout      = 0 -> null
      - storage_opts      = {} -> null
      - sysctls           = {} -> null
      - tmpfs             = {} -> null
      - tty               = false -> null
        # (8 unchanged attributes hidden)
    }

  # 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=601782417e17994f75628bb648b3e4cbbb989037e3a3d302303c4a675a39599b]
docker_container.foo: Destruction complete after 0s
docker_image.ubuntu: Destroying... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_image.ubuntu: Destruction complete after 1s

Destroy complete! Resources: 2 destroyed.
PS C:\Users\Ayush Maurya\Desktop\TerraformScript\Docker>
```

Docker images After Executing Destroy step

```
C:\Users\ADITYA DUBEY\OneDrive\Desktop\terraformscripts\Docker>docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE

C:\Users\ADITYA DUBEY\OneDrive\Desktop\terraformscripts\Docker>docker images
```

Name:Aditya Dubey Div:D15C Roll no:10

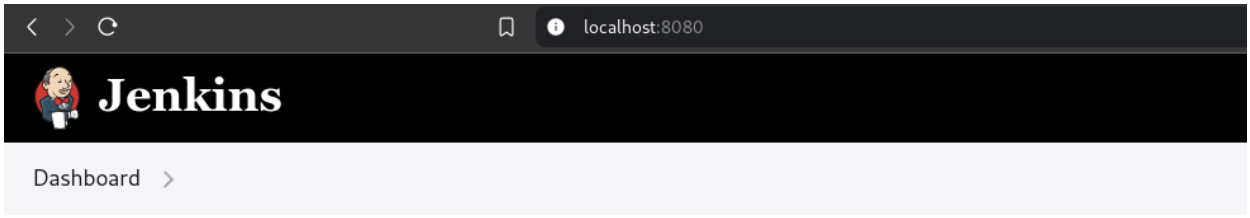
EXPERIMENT NO. 7

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

Prerequisites:
- Jenkins installed (Java JDK required)
- Docker Installed (for SonarQube)

Steps to integrate Jenkins with SonarQube

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



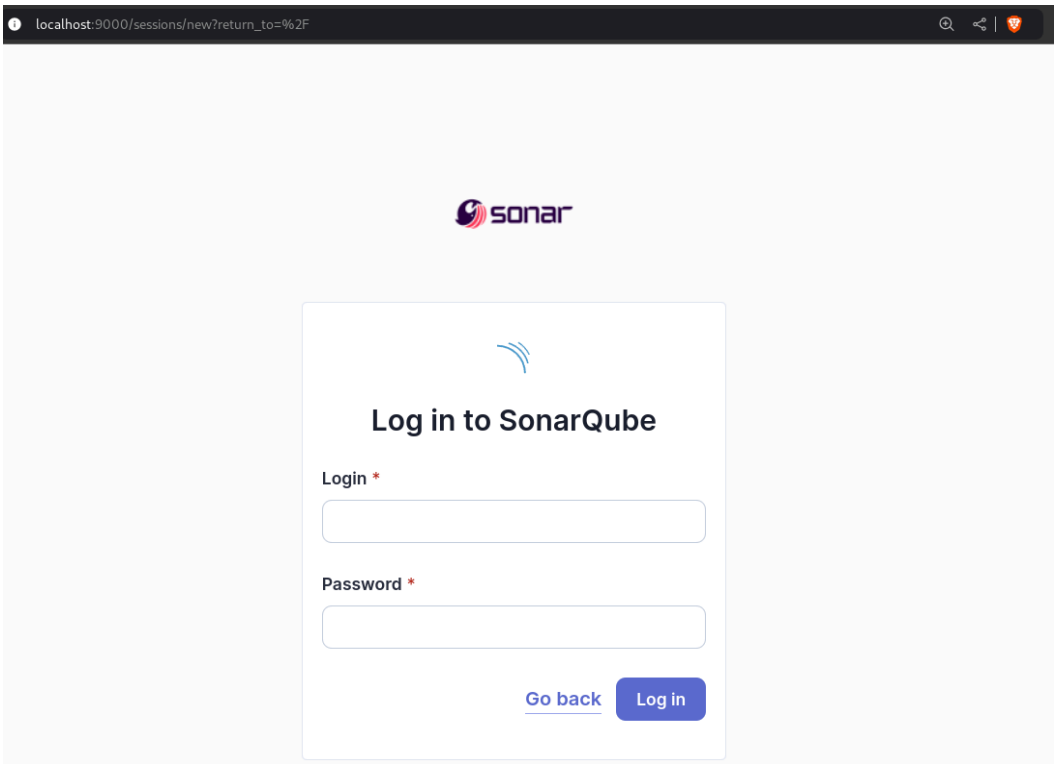
Our jenkins is running on port 8080

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

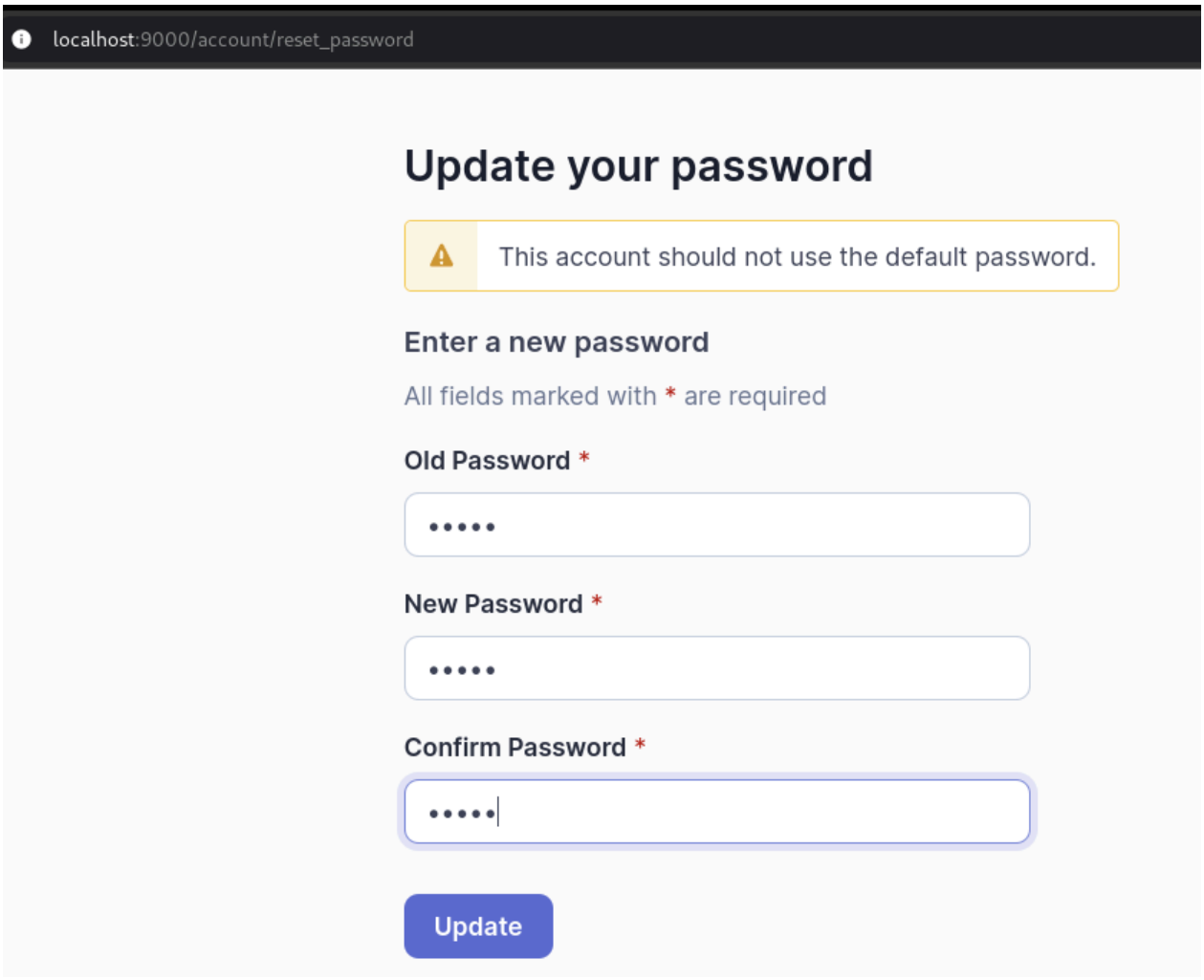
Warning: run below command only once

```
quantum@machine ~ sudo docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
[sudo] password for quantum:
f9c595308e368210e19e099256a47ec1fe44affdc778eb58ccb53174163ce057
```

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

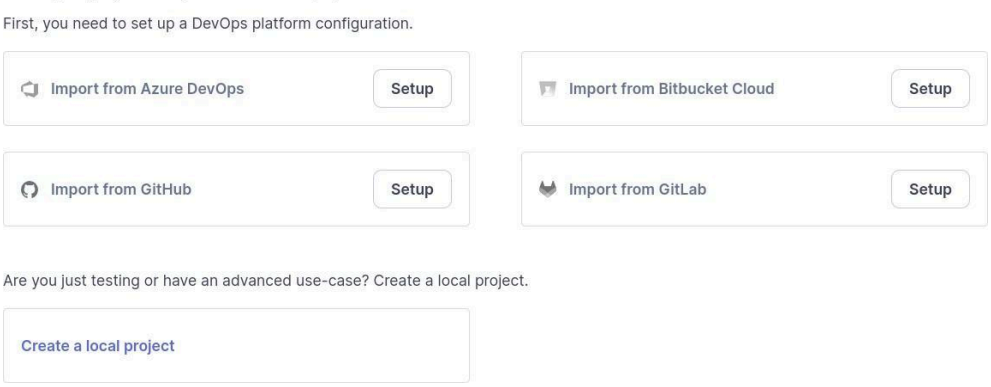


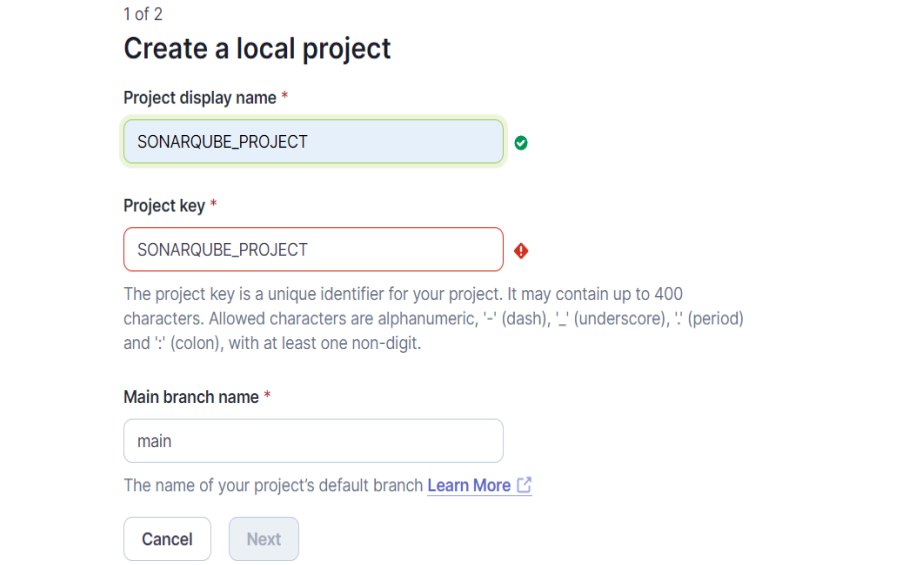
4. Login to SonarQube using username admin and password admin.

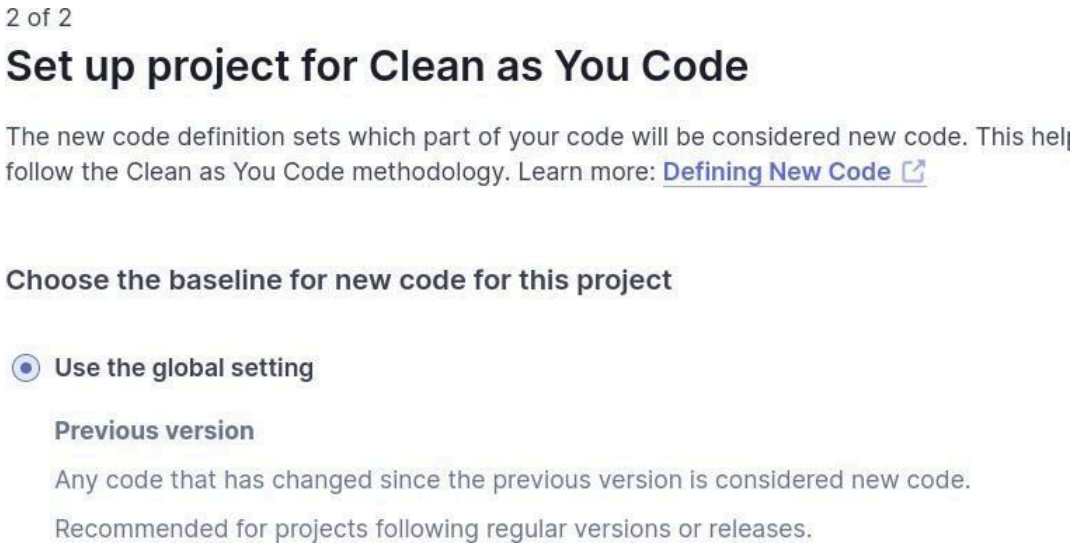


After logging, we have to change default password

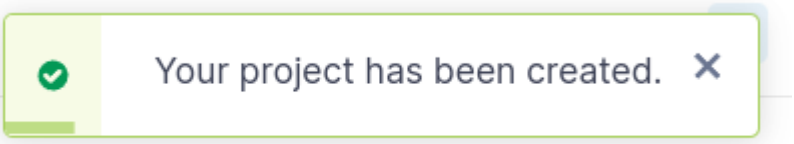
5. Create a manual project in SonarQube with the name sonarqube Click on **create a local project** on dashboard





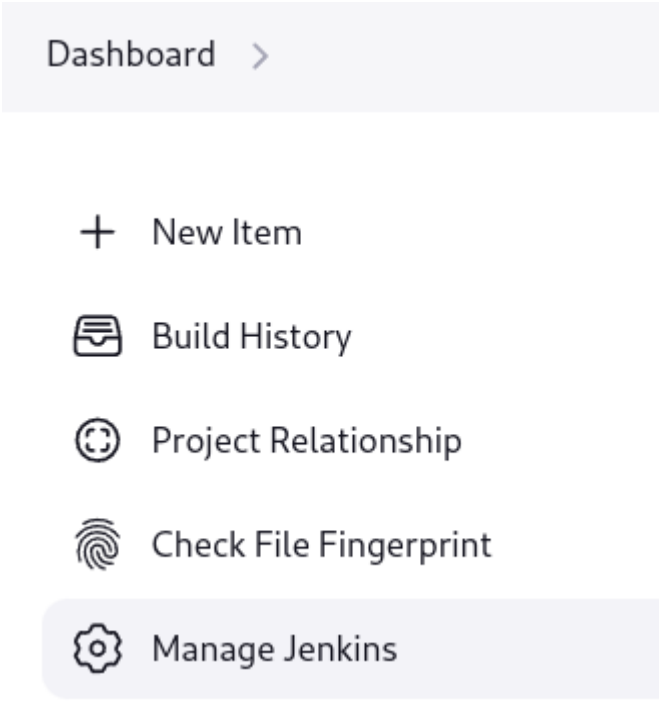


We can either use new custom settings for the project or use global settings
Here I'm using global setting

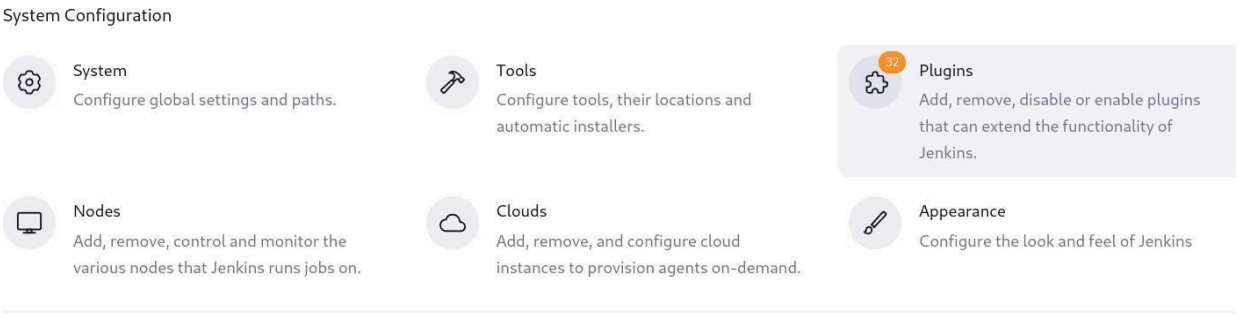


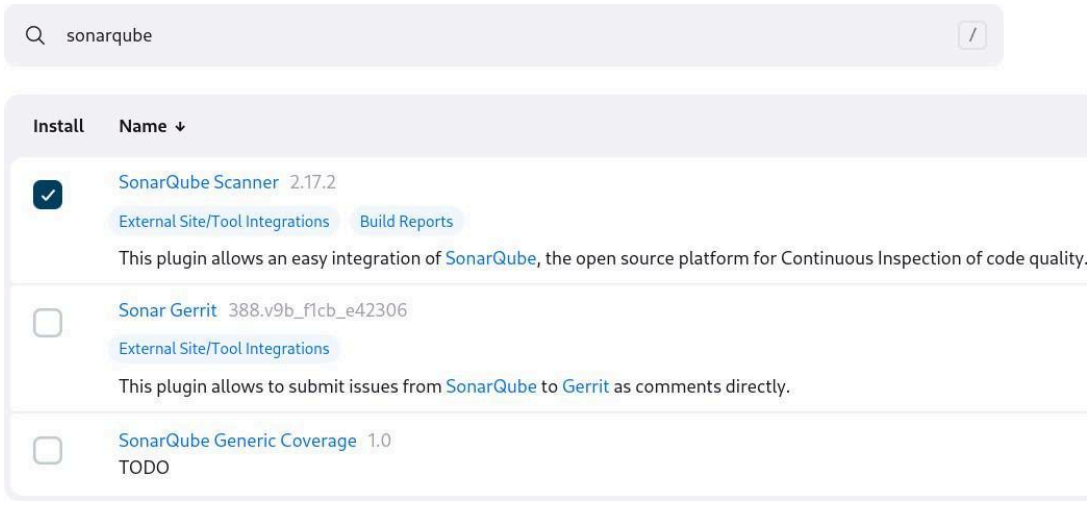
On successful creation of project, we a popup for the same

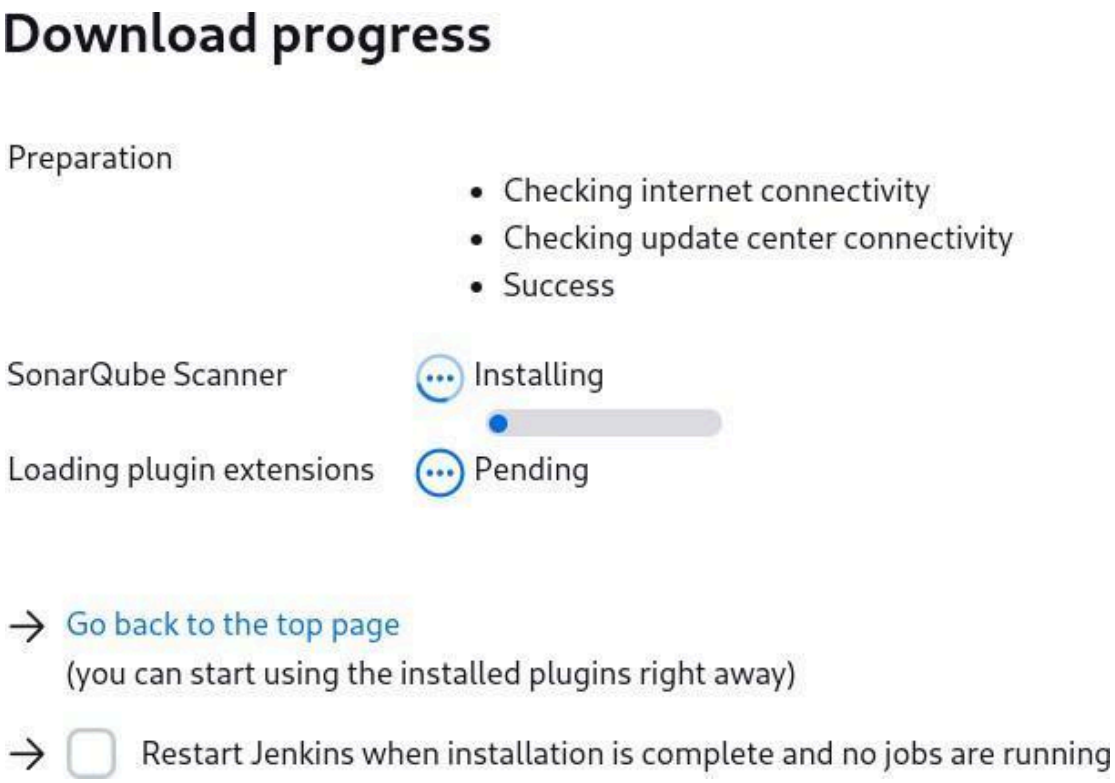
6. After setting project in sonarqube, go to **Jenkins Dashboard**



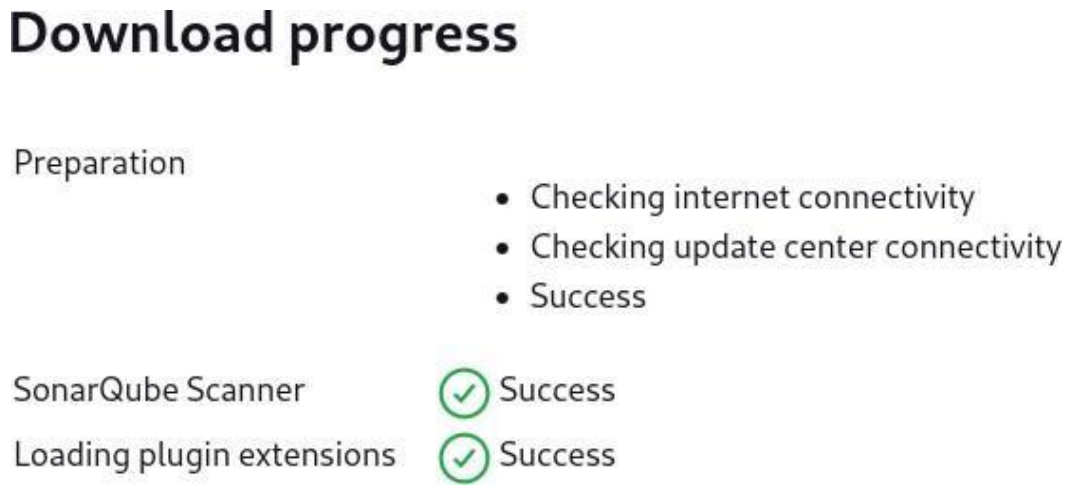
Go to Manage Jenkins and search for SonarQube Scanner in Plugins settings and install it.





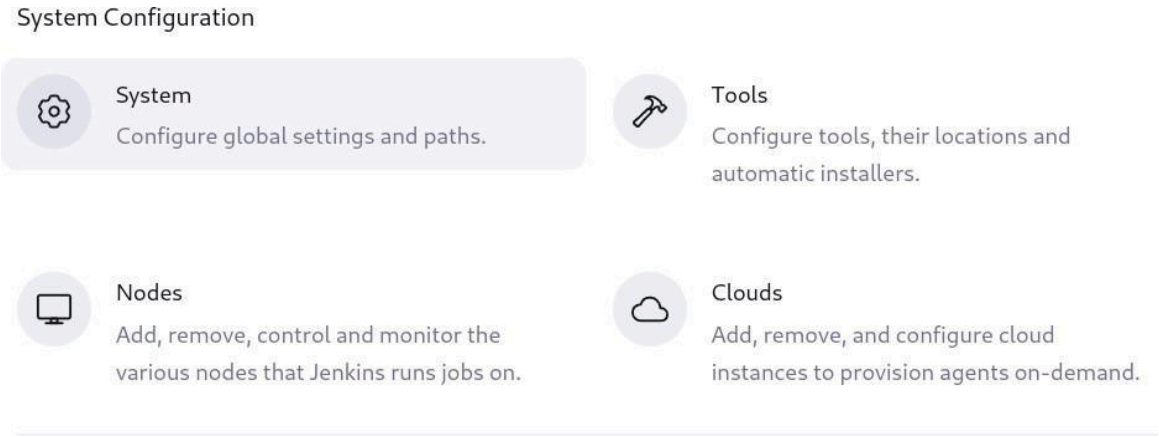


Our installation is in progress wait for it to download and install packages

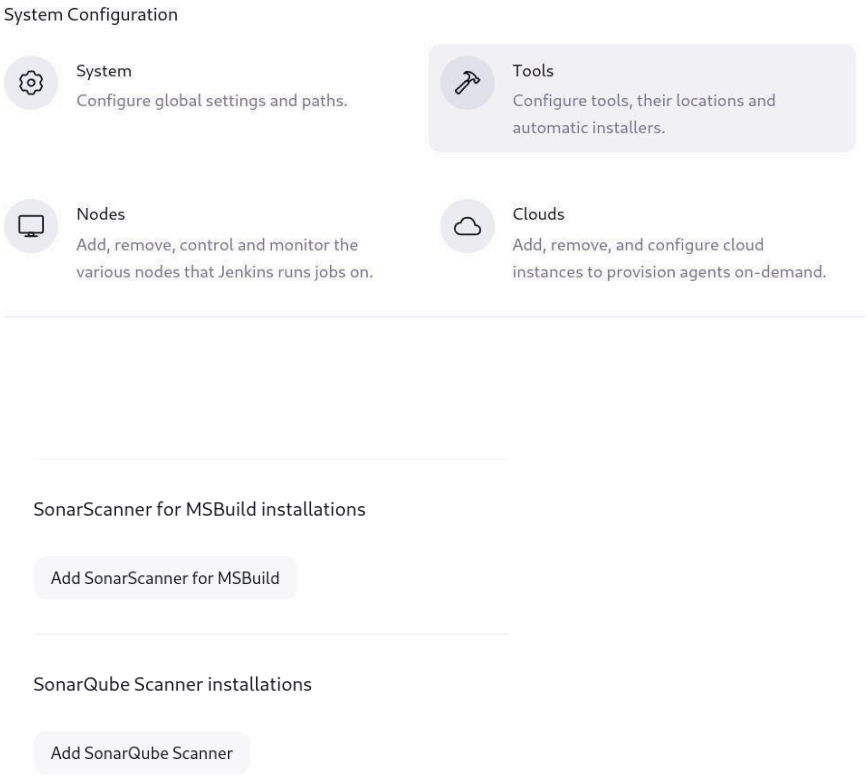


Plugin installed successfully

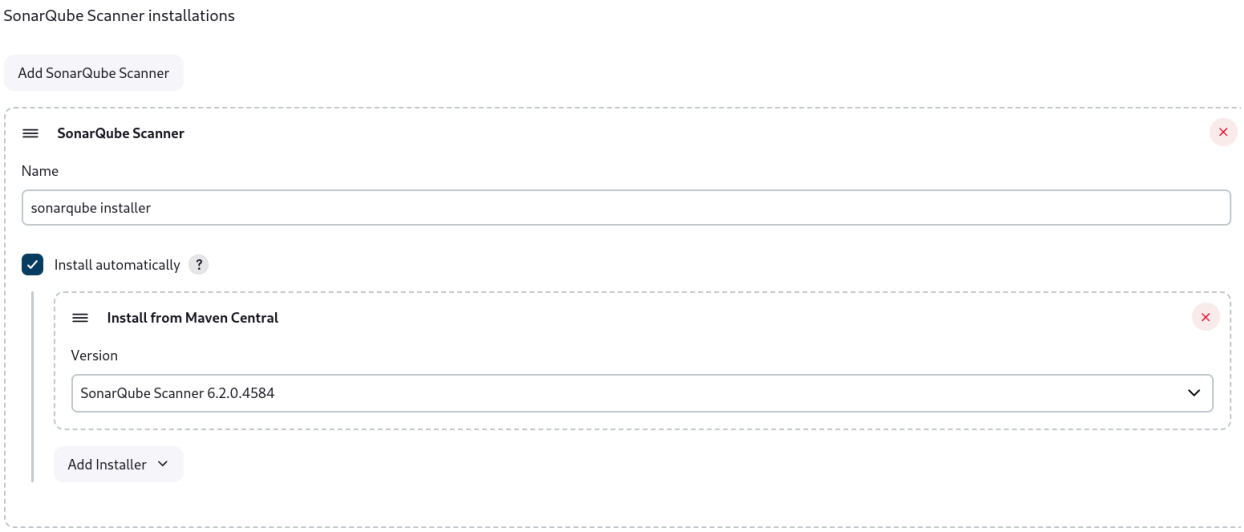
7. Under Jenkins dashboard ‘Configure System’, look for SonarQube Servers and enter the details.



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

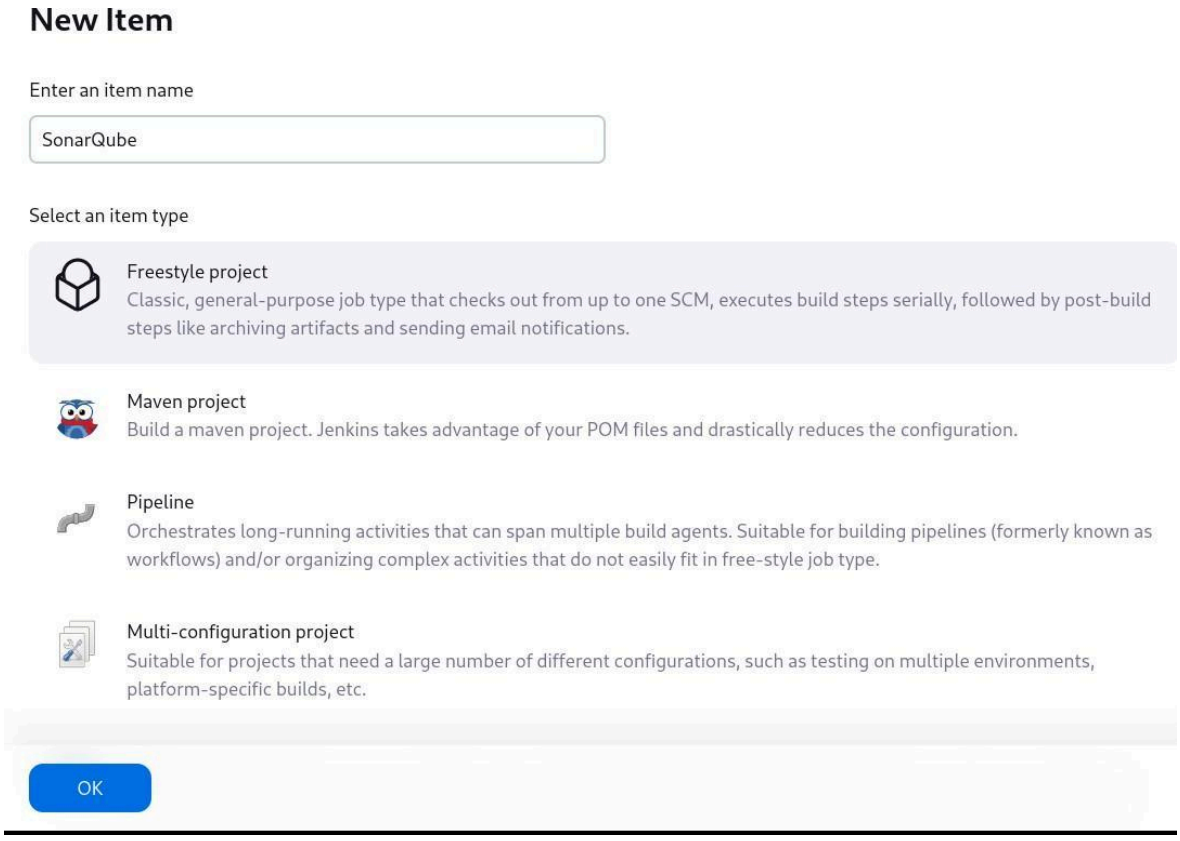


Click on **Add SonarQube Scanner**

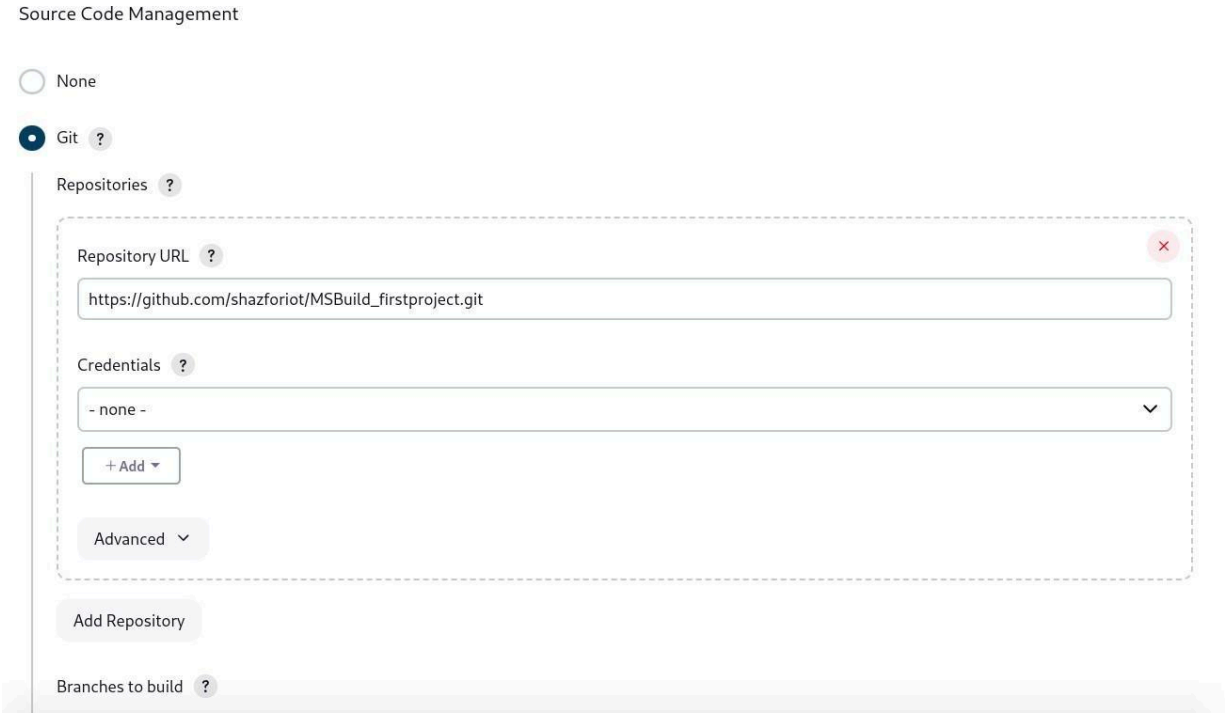


Select Latest version and save configuration

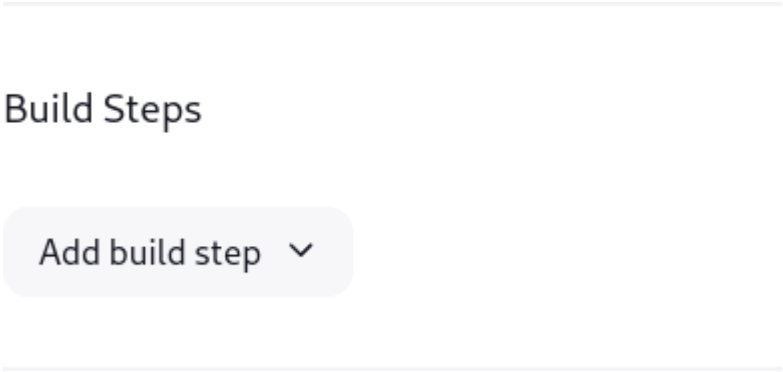
9. After the configuration, create a New Item in Jenkins, choose a freestyle project.



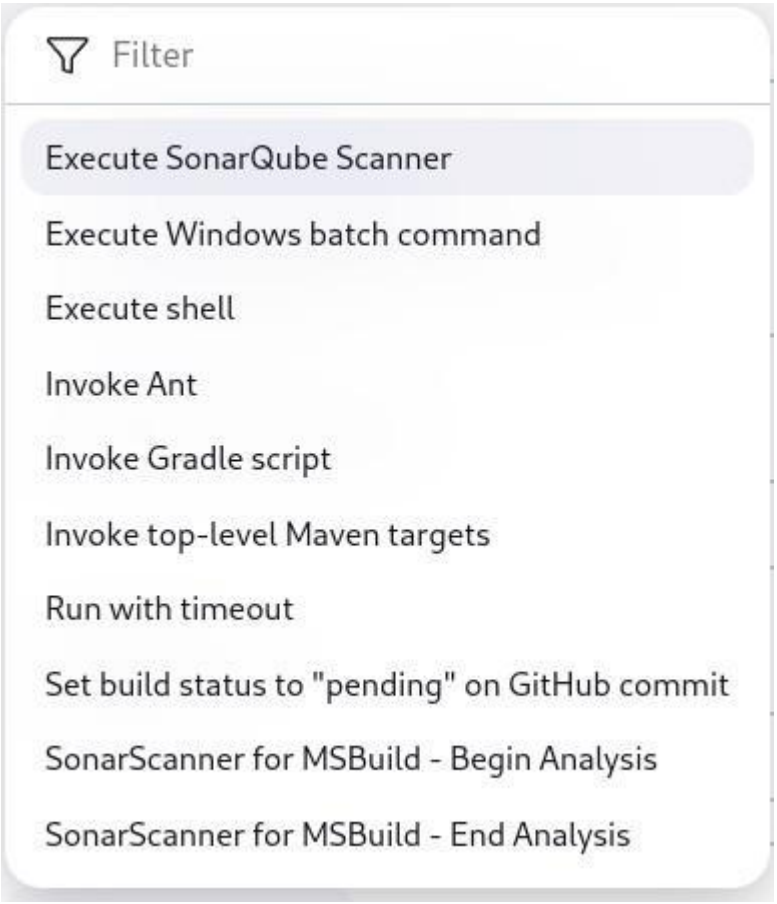
10. Choose this GitHub repository in Source Code Management. https://github.com/shazforiot/MSBuild_firstproject.git It is a sample hello-world project with no vulnerabilities and issues, just to test the integration



11. Under Build-> Execute SonarQube Scanner

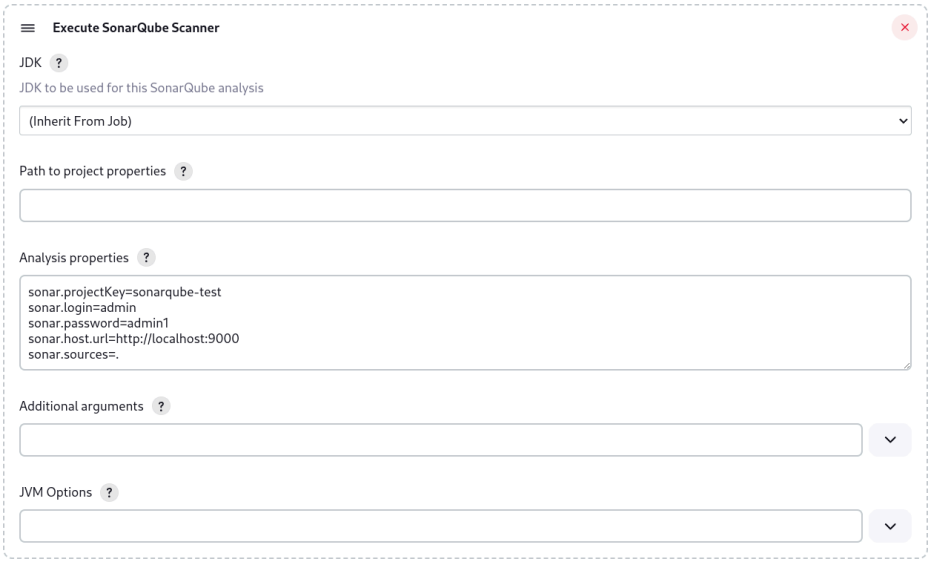


Click on add build steps

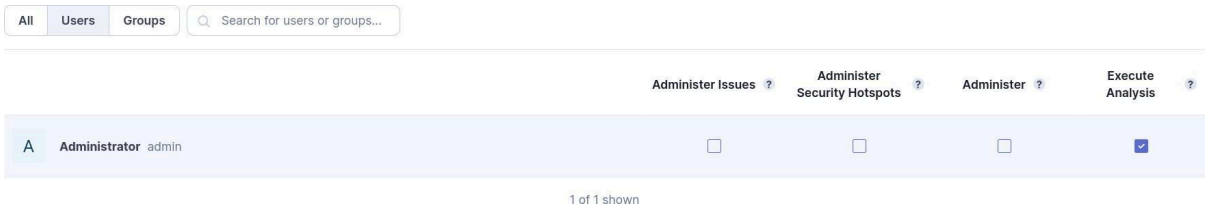


Then click on **Execute SonarQube Scanner**

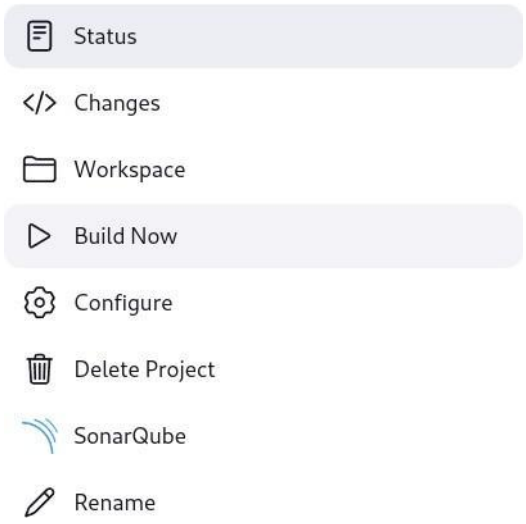
12. Mention the SonarQube Project Key, Login, Password, Source path and Host URL in Analysis properties



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



14. Run The Build.

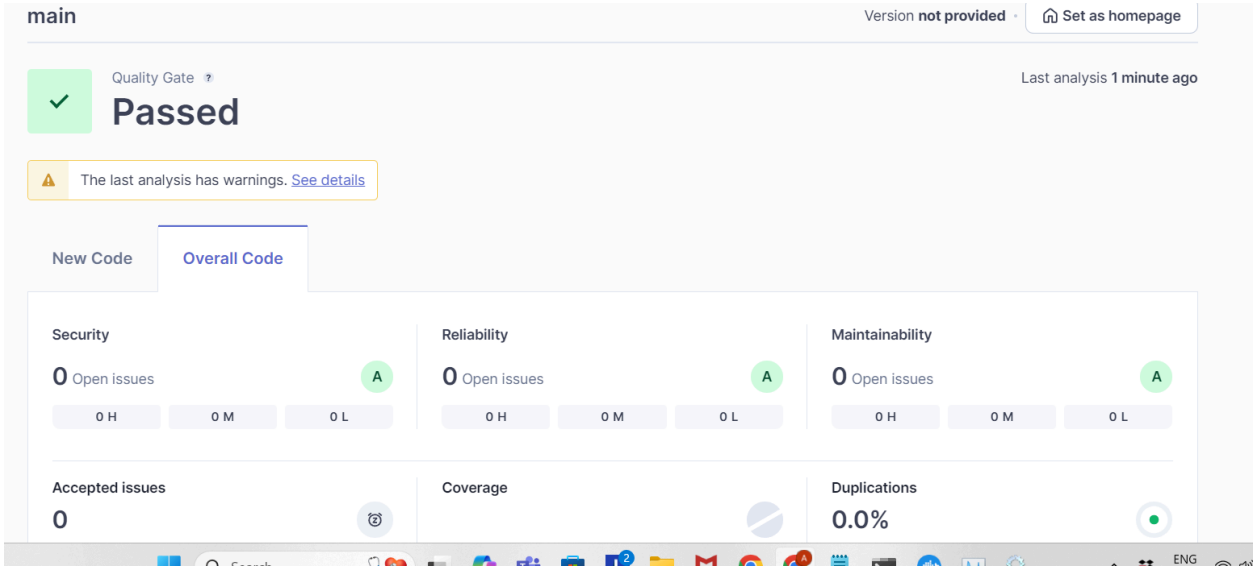


15. Check the console output

```
Started by user Aditya Dubey
Running as SYSTEM
Building on the built-in node in workspace C:\Users\ADITYA DUBEY\.jenkins\workspace\Sonarqube
The recommended git tool is: NONE
No credentials specified
 > git.exe rev-parse --resolve-git-dir C:\Users\ADITYA DUBEY\.jenkins\workspace\Sonarqube\.git # timeout=10
Fetching changes from the remote Git repository
 > git.exe config remote.origin.url https://github.com/shazforiot/MSBuild_firstproject.git # timeout=10
Fetching upstream changes from https://github.com/shazforiot/MSBuild_firstproject.git
 > git.exe --version # timeout=10
 > git --version # 'git version 2.46.0.windows.1'
 > git.exe fetch --tags --force --progress -- https://github.com/shazforiot/MSBuild_firstproject.git +refs/heads/*:refs/remotes/origin/* # tir
 > git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision f2bc042c04c6e72427c380bcaee6d6fee7b49adf (refs/remotes/origin/master)
 > git.exe config core.sparsecheckout # timeout=10
 > git.exe checkout -f f2bc042c04c6e72427c380bcaee6d6fee7b49adf # timeout=10
Commit message: "updated"
 > git.exe rev-list --no-walk f2bc042c04c6e72427c380bcaee6d6fee7b49adf # timeout=10
[Sonarqube] $ "C:\Users\ADITYA DUBEY\.jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\insaller\bin\sonar-scanner.bat" -
Dsonar.host.url=http://localhost:9000 -Dsonar.projectKey=SONARQUBE_PROJECT -Dsonar.projectName=SONARQUBE_PROJECT -
Dsonar.host.url=http://localhost:9000 -Dsonar.login=admin -Dsonar.sources=. -Dsonar.password=a1b2d4e5i9i10 "-Dsonar.projectBaseDir=C:\Users\AD
```

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

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



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

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

EXPERIMENT NO. 8

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

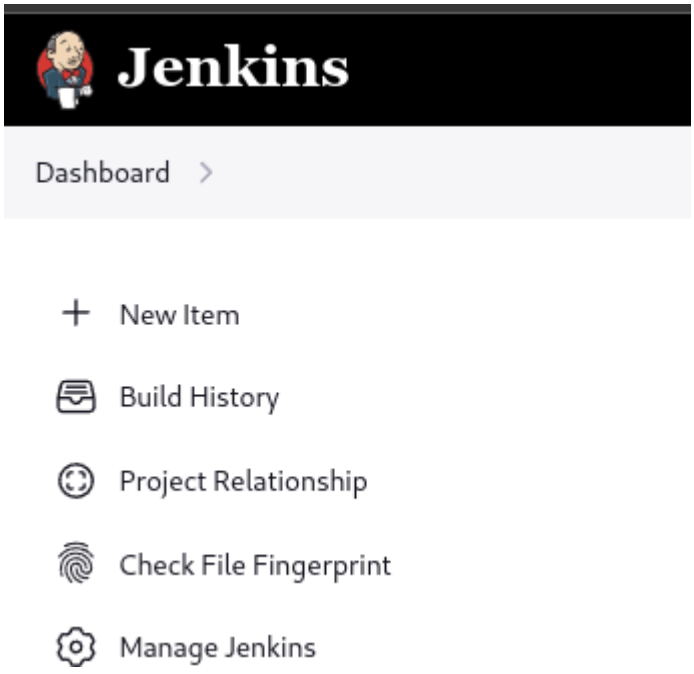
Integrating Jenkins with SonarQube:

Prerequisites:

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

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

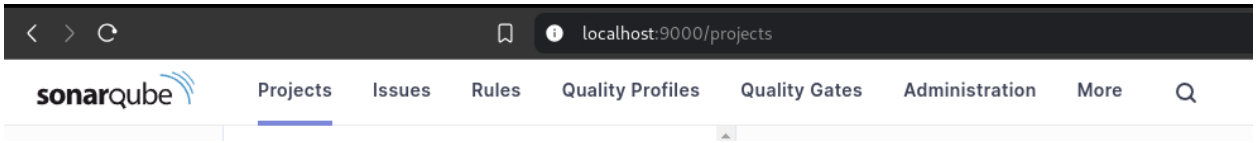
1. Open up Jenkins Dashboard on localhost, port 8080 or whichever port it is at for you.



2. Run SonarQube in a Docker container using this command

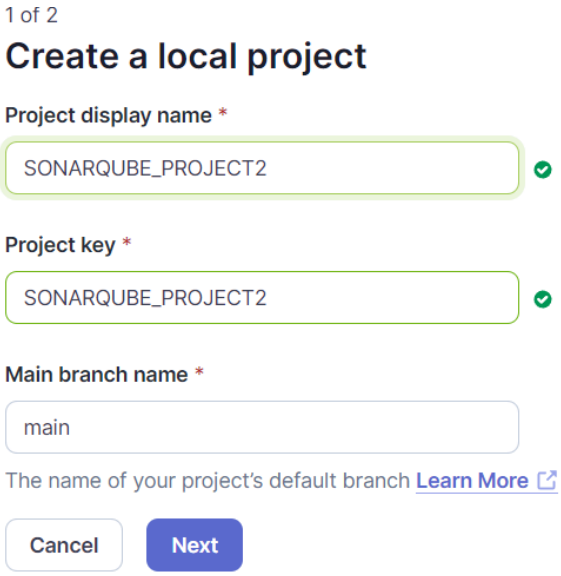
```
quantum@machine ~ sudo docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
```

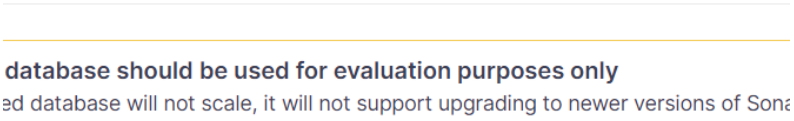
3. Once the container is up and running, you can check the status of SonarQube at localhost port 9000.



4. Login to SonarQube using your username and password

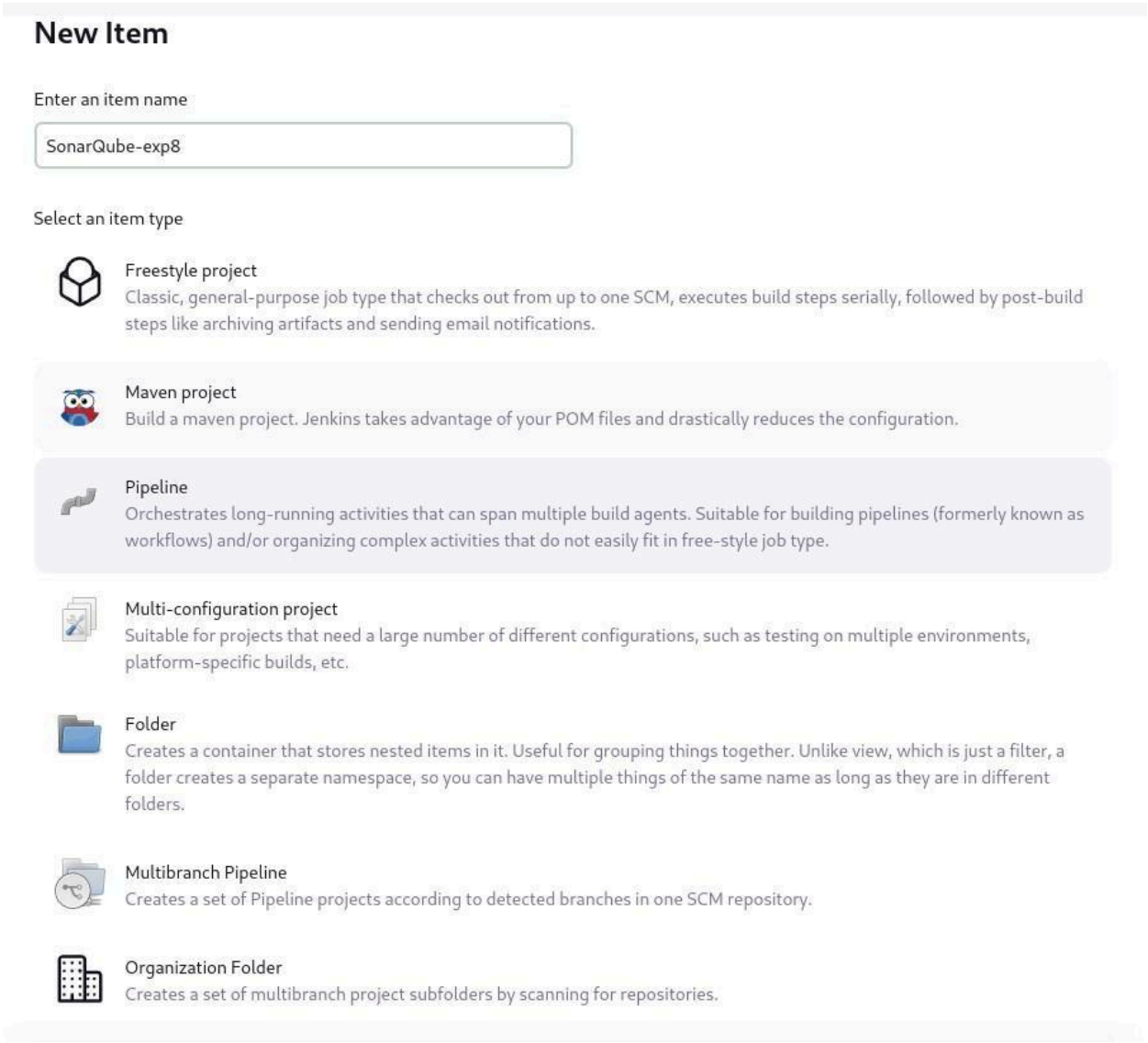
5. Create a manual project in SonarQube with the name sonarqube-test2





Setup the project and come back to Jenkins Dashboard.

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



7. Under Pipeline Script, enter the following -

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

```
}
}
}
```

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

8. Install sonar-scanner

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

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

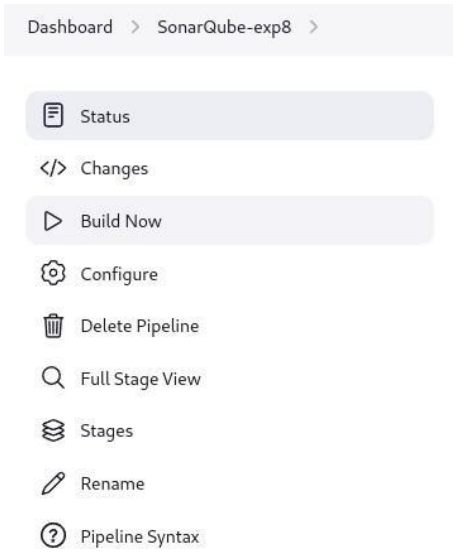
```
pipeline {
    agent any
    stages {
        stage('Cloning the GitHub Repo') {
            steps {
                git 'https://github.com/shazforiot/GOL.git'
            }
        }
        stage('SonarQube analysis') {
            steps {
                withSonarQubeEnv('sonarqube') {configuration
                    sh """
“C:\\Users\ADITYA DUBEY\Downloads\sonar-scanner-cli-6.2.0.4584-windows-x64\
                    sonar-scanner-cli-6.2.0.4584-windows-x64\bin\sonar-scanner” ^
                    -D sonar.projectKey=sonarqube-test2 ^
                    -D sonar.sources=. ^
                    -D sonar.exclusions=vendor/**,resources/**,**/*.java ^
```

```
                -D sonar.host.url=http://127.0.0.1:9000 ^
                -D sonar.login=admin ^
                -D sonar.password=my pass
            """
        }
    }
}
}
```

Script ?

```
pipeline {
    agent any
    stages {
        stage('Cloning the GitHub Repo') {
            steps {
                git 'https://github.com/shazforiot/GOL.git'
            }
        }
        stage('SonarQube Analysis') {
            steps {
                withSonarQubeEnv('sonarqube') {
                    bat """
                    "C:/Users/ADITYA DUBEY/Downloads/sonar-scanner-cli-6.2.0.4584-windows-x64/sonar-scanner-6.2.0.4584-windows-x64/bin/sona
                    -D sonar.projectKey=SONARQUBE_PROJECT2 ^
                    -D sonar.sources=. ^
                    -D sonar.exclusions=vendor/**,resources/**,**/*.java ^
                    -D sonar.host.url=http://127.0.0.1:9000 ^
                    -D sonar.login=admin ^
                    -D sonar.password=a1b2d4e5i9j10
                    """
                }
            }
        }
    }
}
```

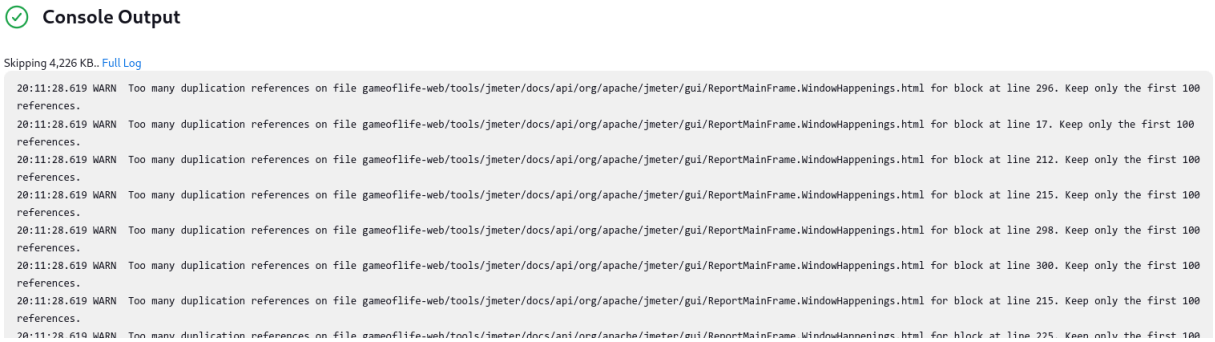
9. Run the build



Check the status of Build

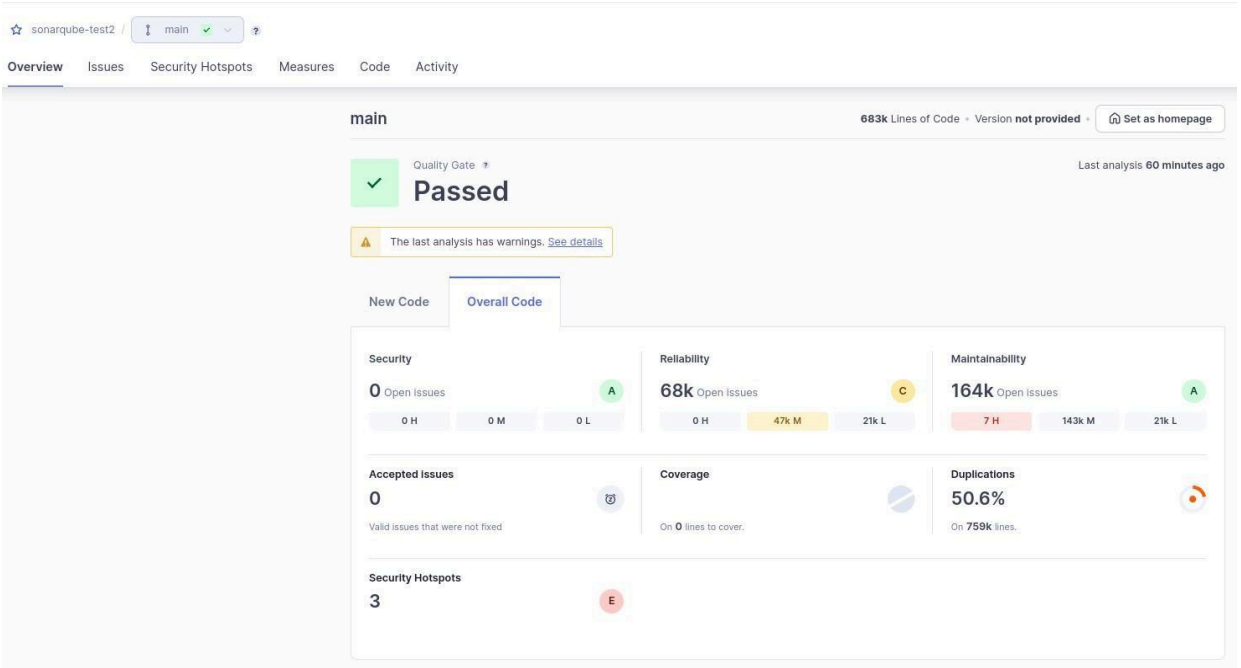
As we can see the SonarQube analysis is completed

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



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

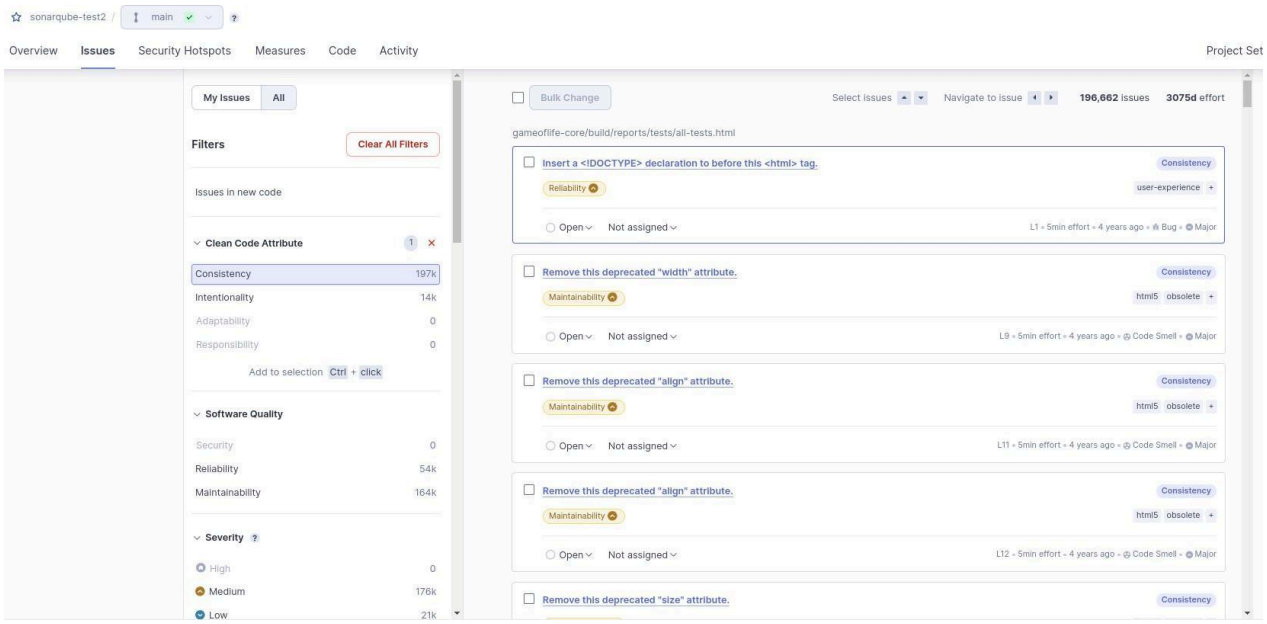
11. After that, check the project in SonarQube.



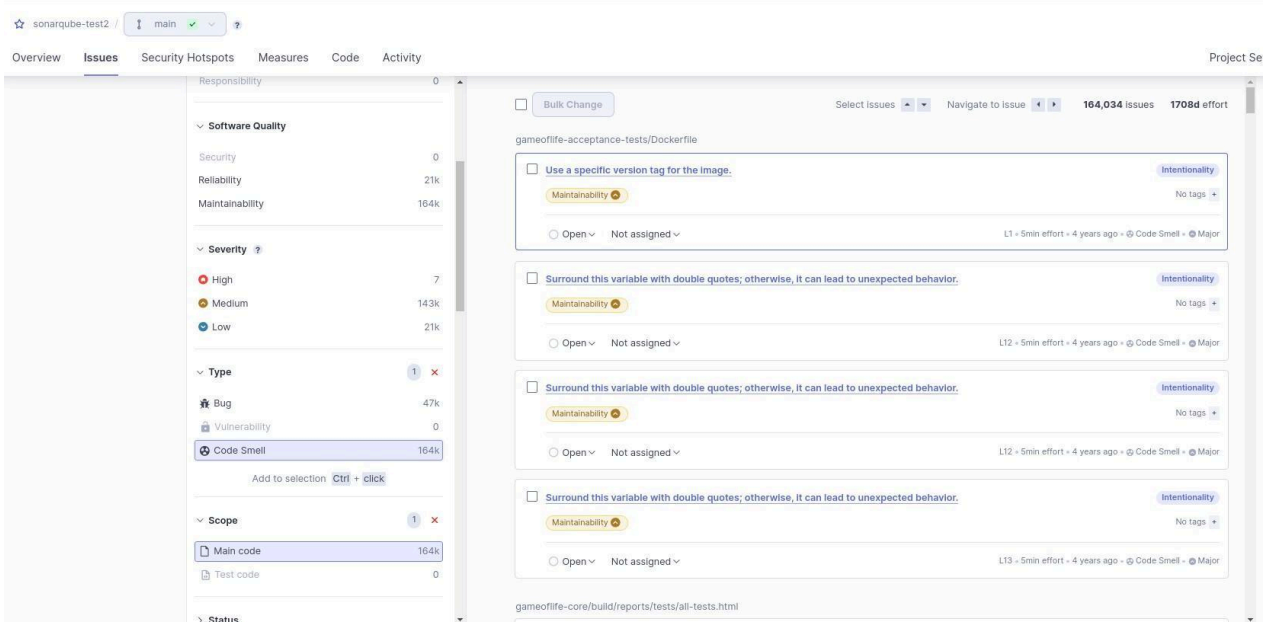
Under different tabs, check all different issues with the code

12. **Code Problems-**

Name:Aditya Dubey Div:D15C Roll no:10

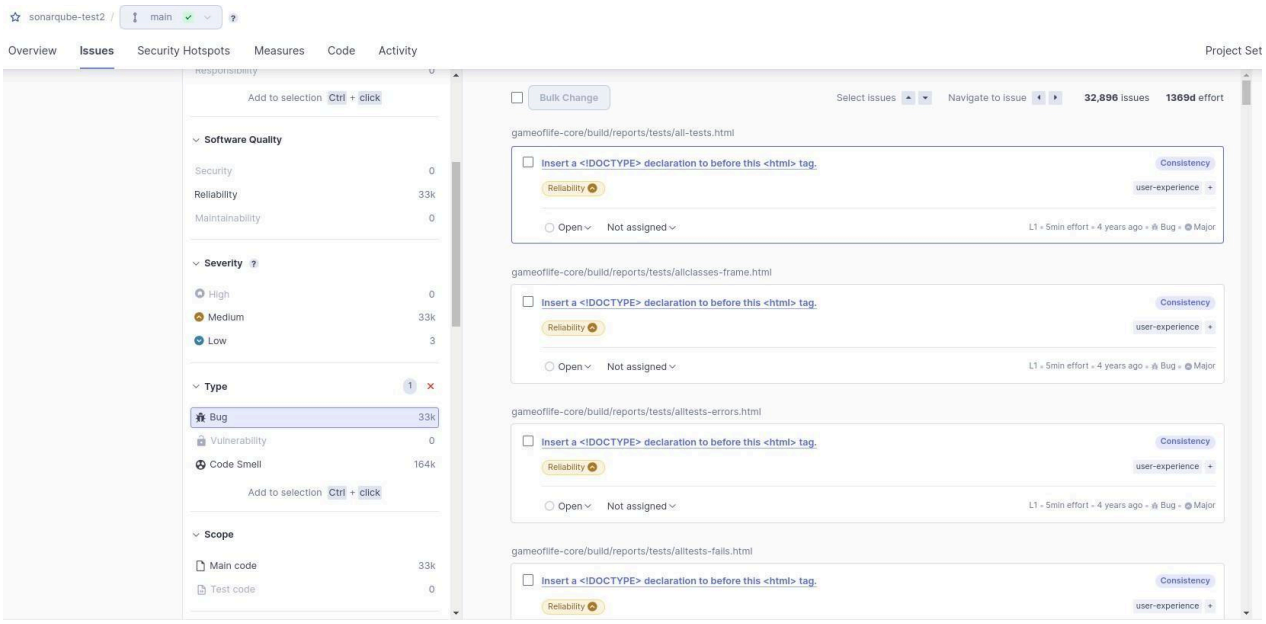


Code Smells



Bugs

Name:Aditya Dubey Div:D15C Roll no:10



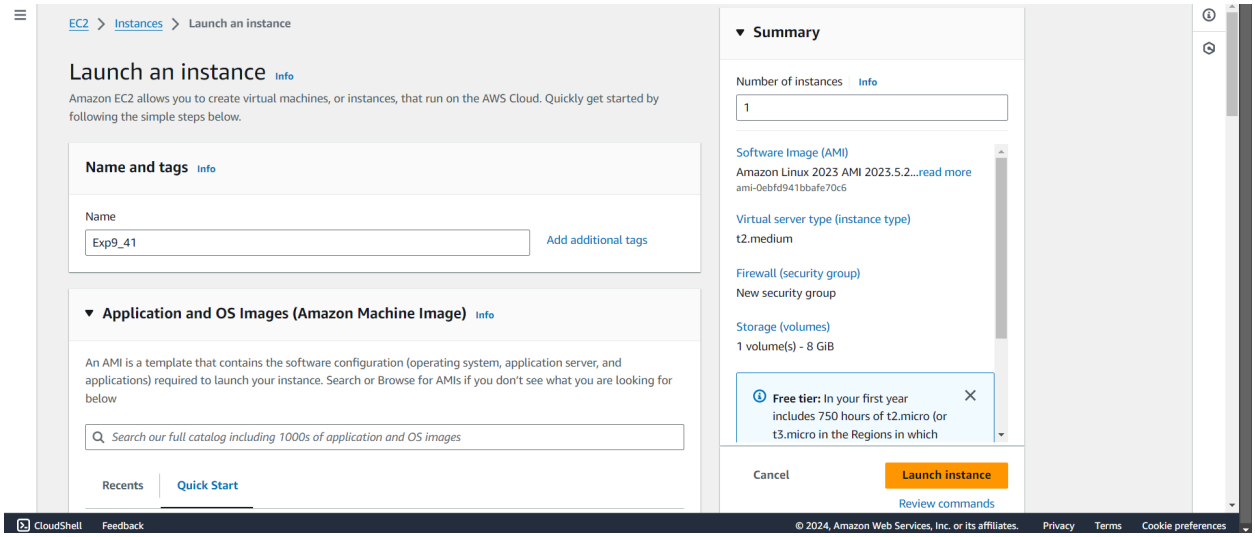
Conclusion: We began the experiment with creating a new project in SonarQube and setting up a new Pipeline in Jenkins with proper configuration of pipeline script. Then we installed Sonar Scanner CLI so that jenkins can do code analysis of Git Repository. We can also configure the pipeline to use the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

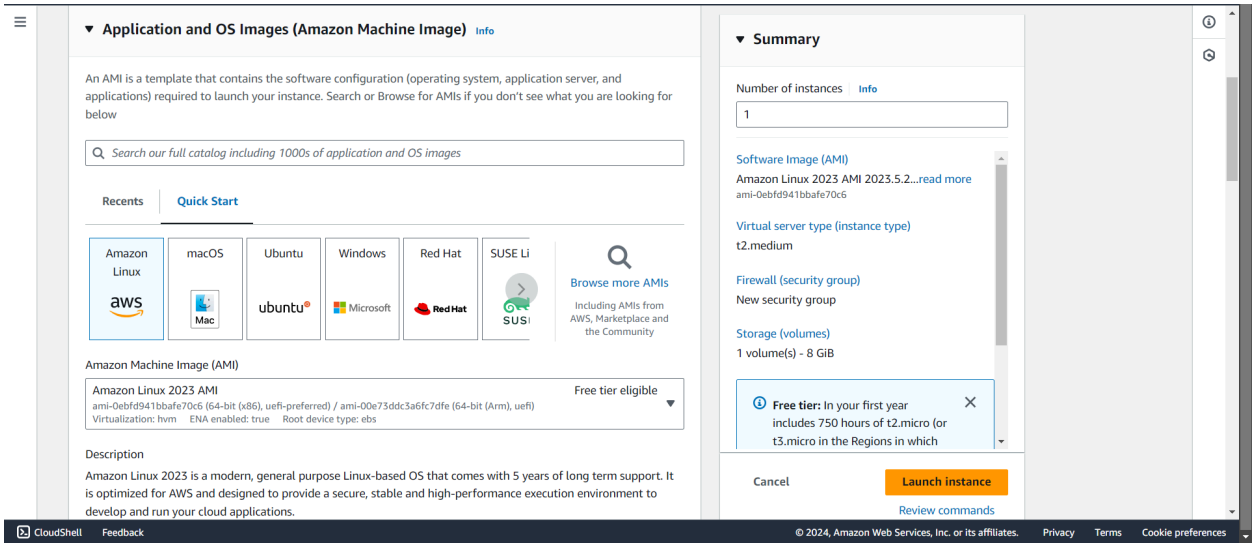
Name: Aditya Dubey **Div: D15C** **Roll No: 10**

Experiment No: 9

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

Step 1: Sign in to your AWS account. Look for EC2 in the services list. Open it and click on "Create Instance."

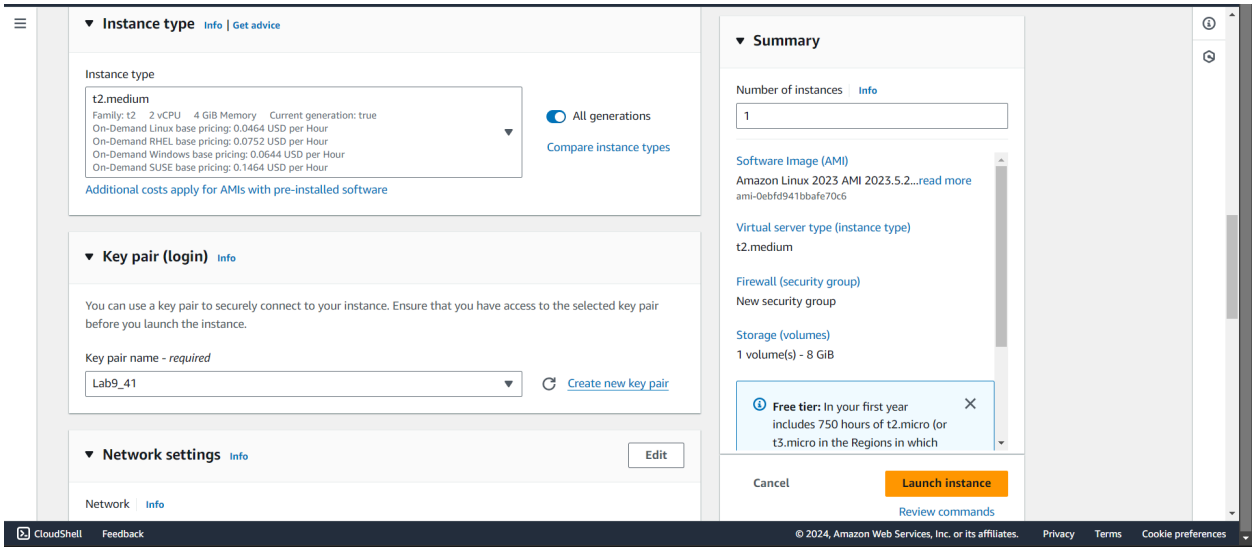


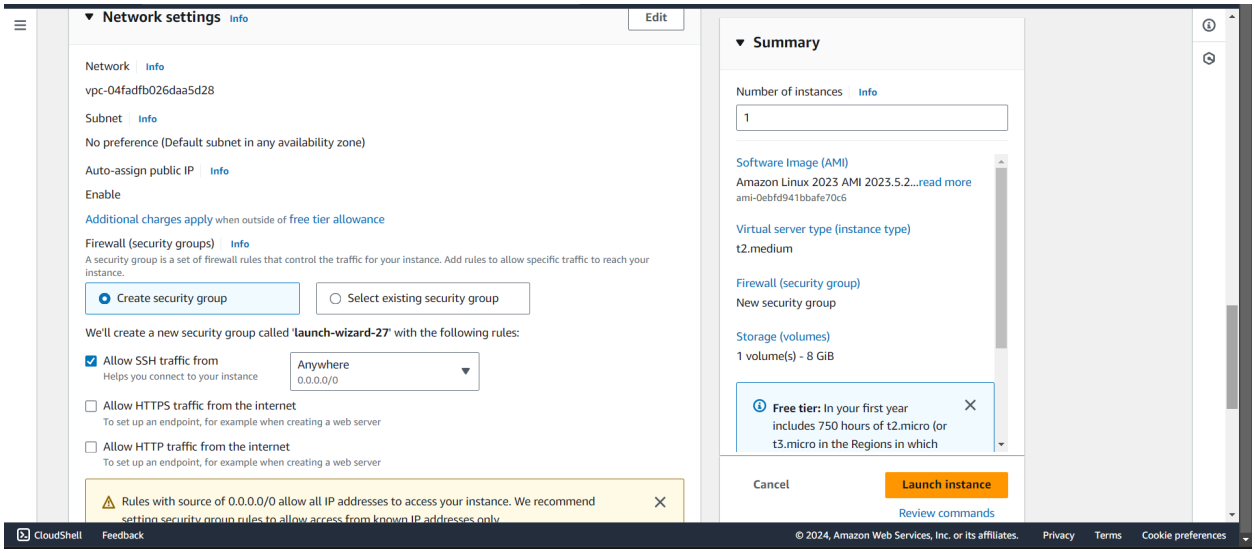


Select The OS Image as Amazon Linux.

Name: Aditya Dubey **Div: D15C** **Roll No: 10**

Step 2: If you haven't created a private key or a .pem file yet, click on "**Create a key pair**." Otherwise, choose the key pair you created earlier. (Be sure to remember where the .pem file for that key is located on your system.) .in my case i have created a new one



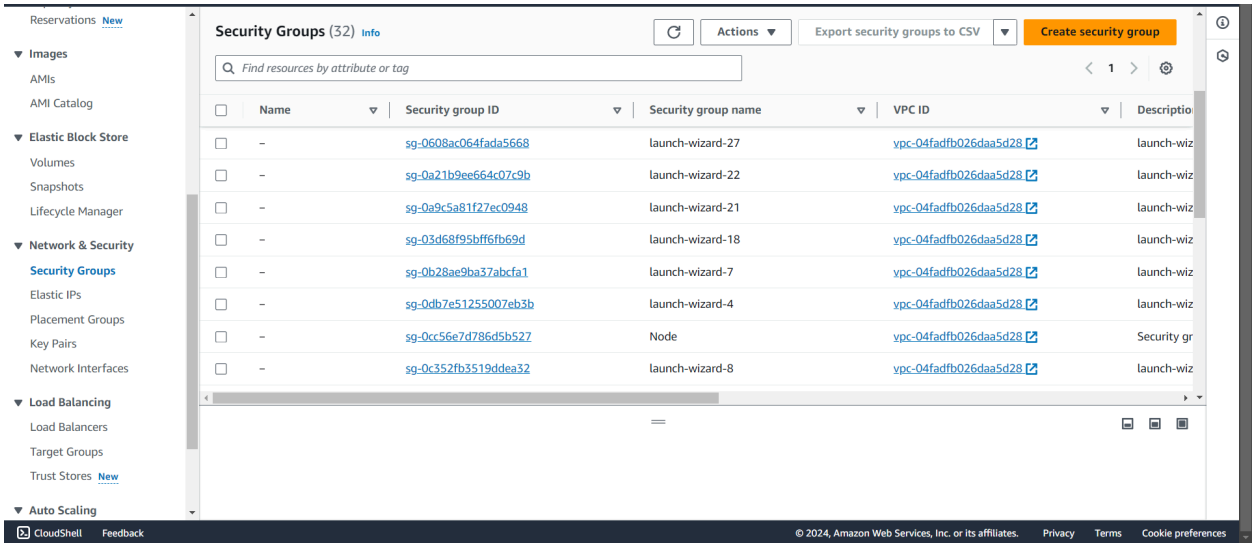


AWS will create a security group for this instance. Keep the name of that instance saved.

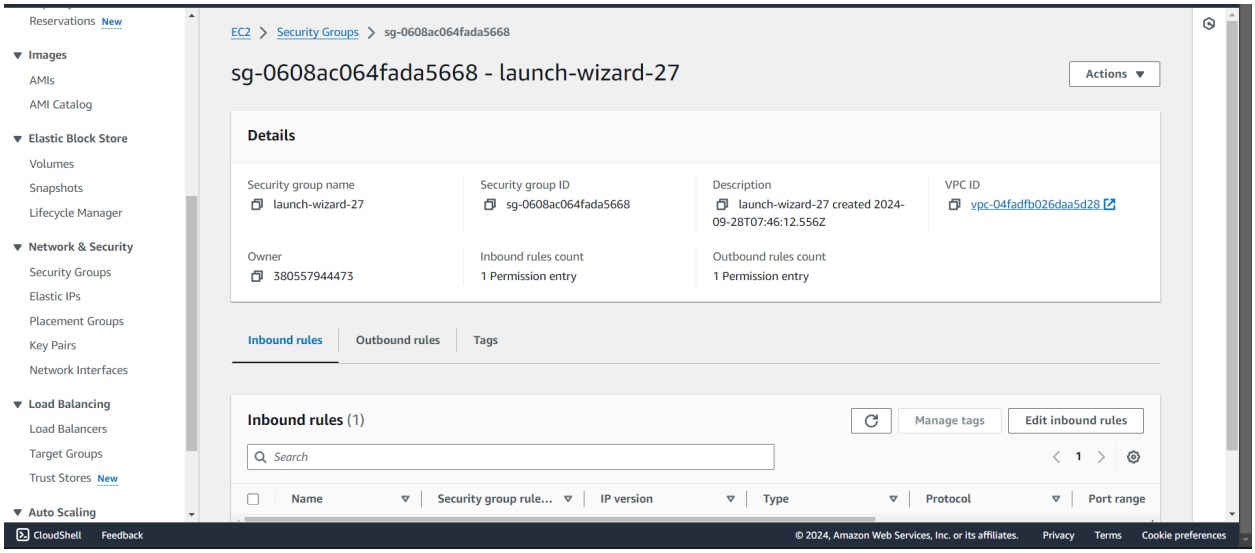
Instance:

Name: Aditya Dubey **Div: D15C** **Roll No: 10**

Step3: After you create the instance, click on "**Security Groups**" in the left sidebar. Look for the security group that corresponds to your instance, and then click on the security group ID for that group. (in my case launch-wizard-27 is the latest one.)

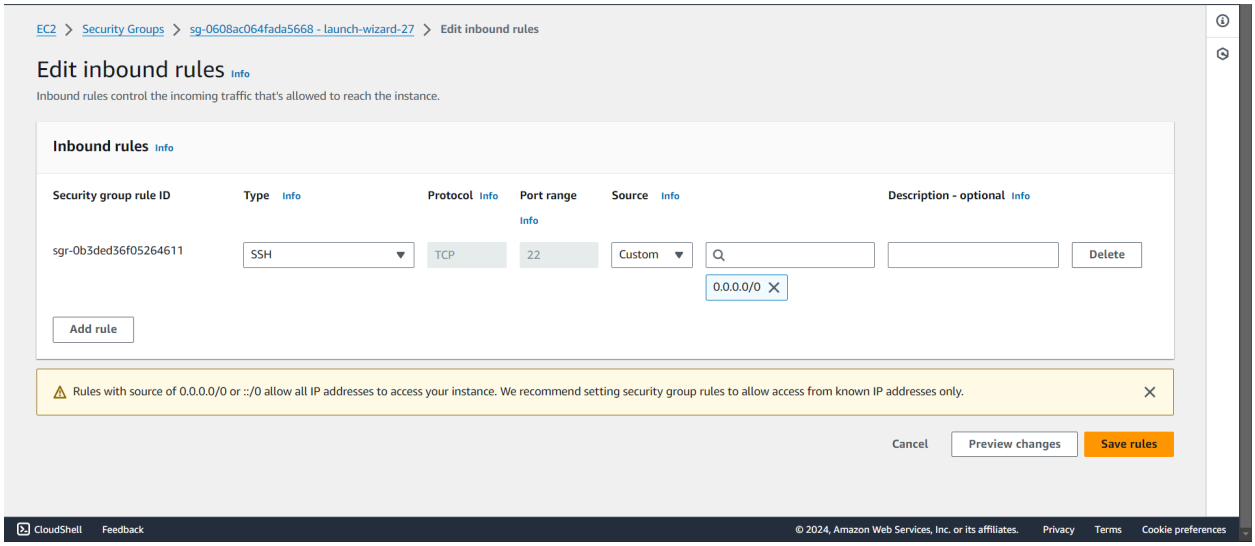


Click on Id

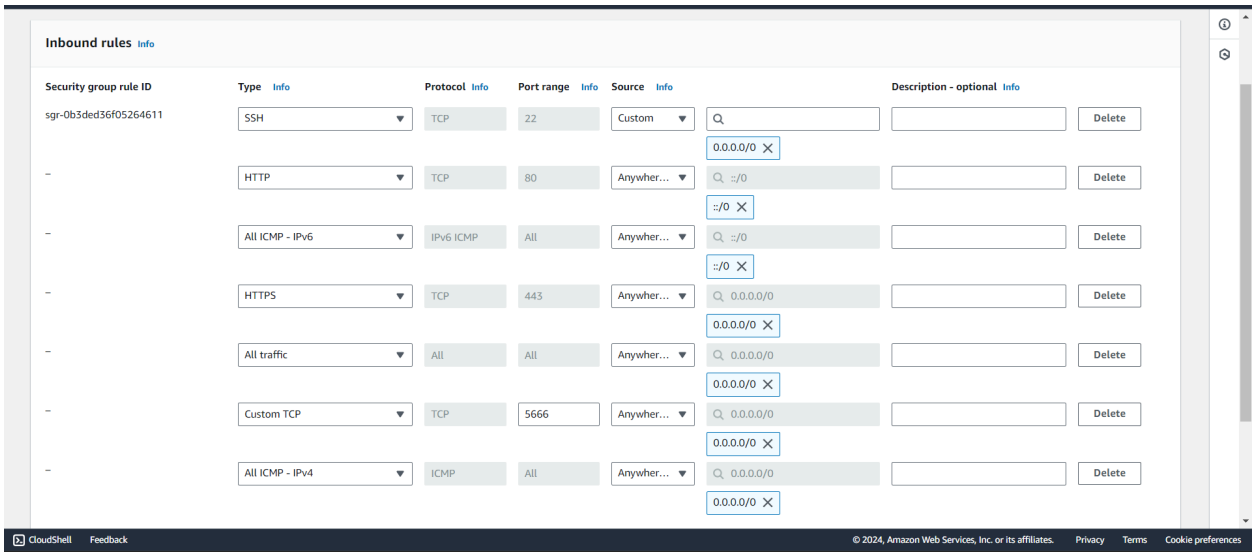


Click on the edit inbound rules

Name: Aditya Dubey **Div: D15C** **Roll No: 10**

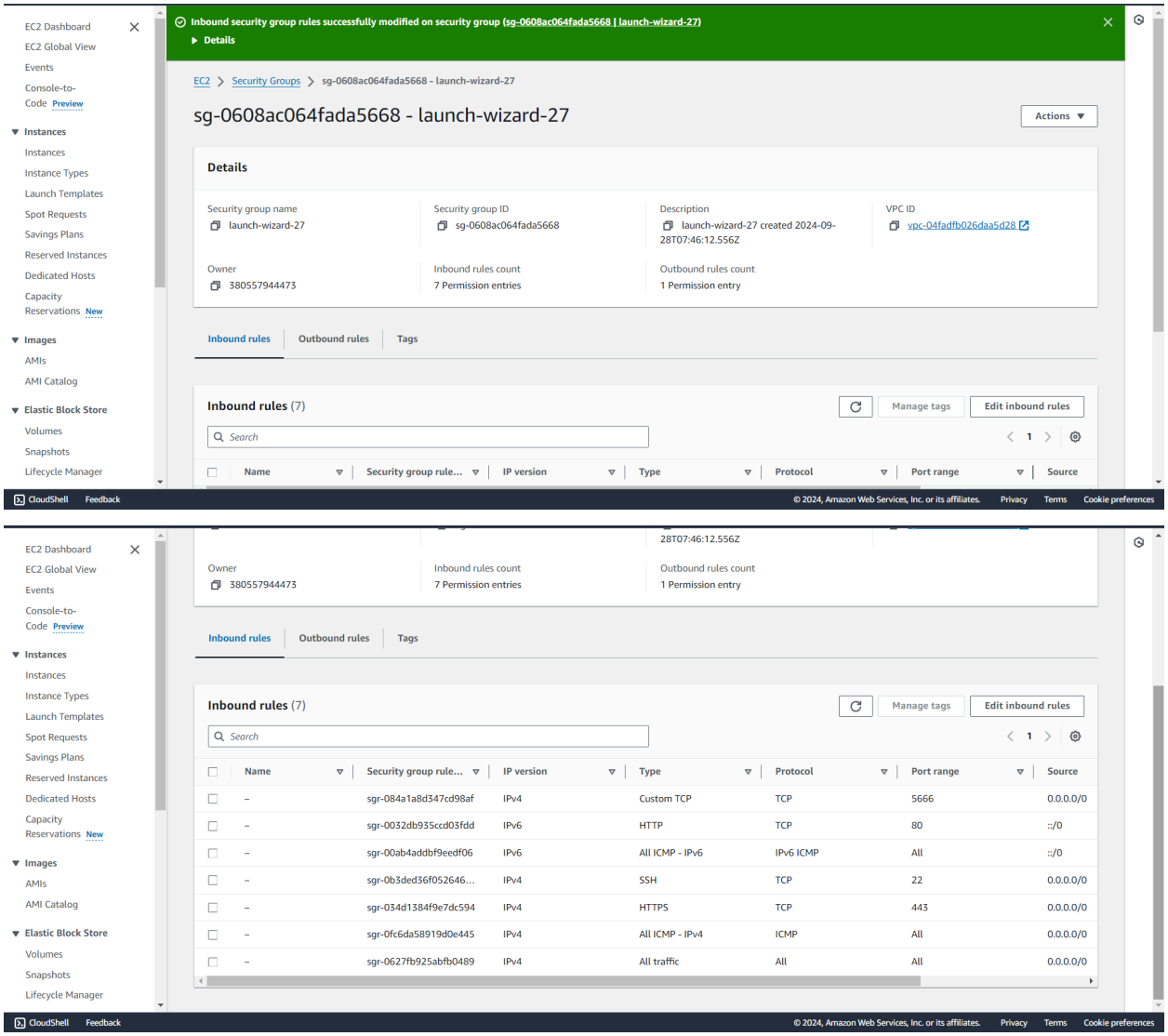


Next, click on "Add rules" and set up rules for the following protocols: HTTP, All ICMP (IPv6), HTTPS, All traffic, Custom TCP (Port 5666), and All ICMP (IPv4).

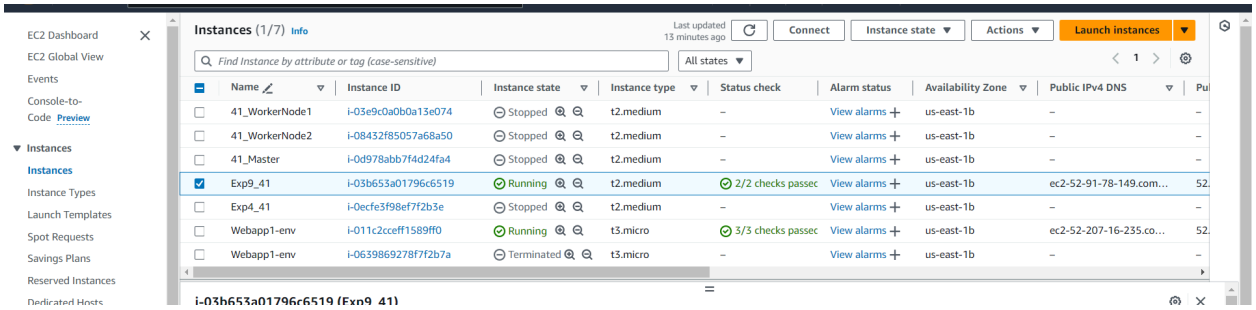


Click on save. This will add all the inbound rules to the security group.

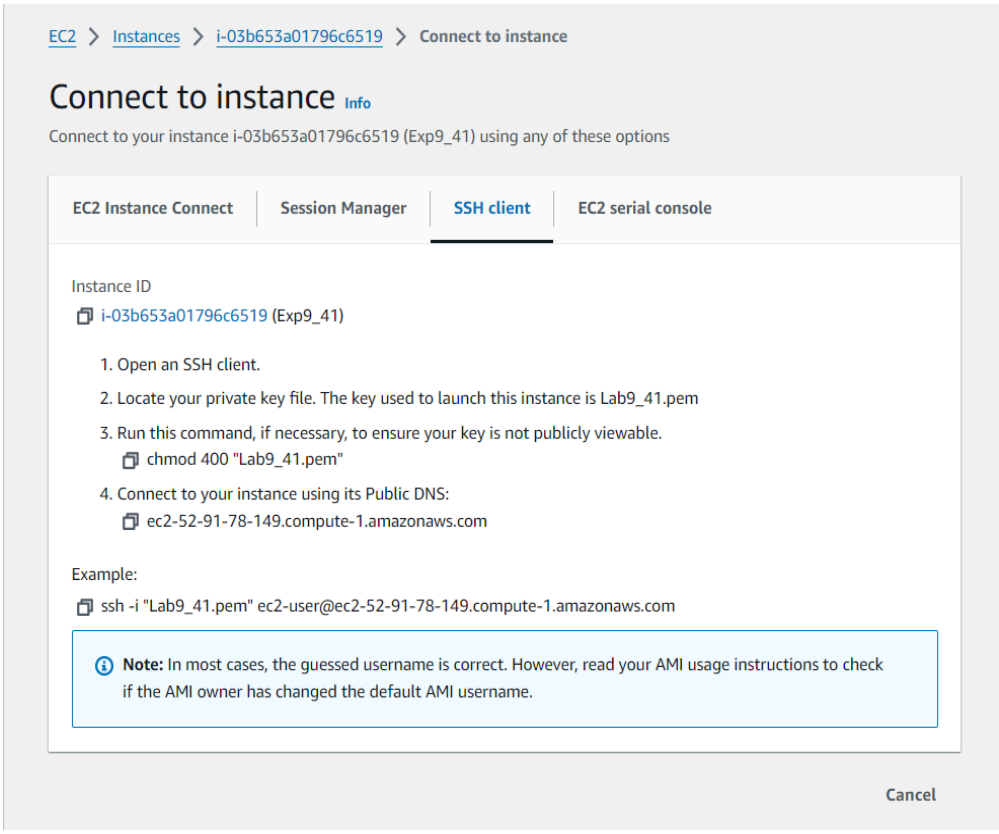
Name: Aditya Dubey **Div: D15C** **Roll No: 10**



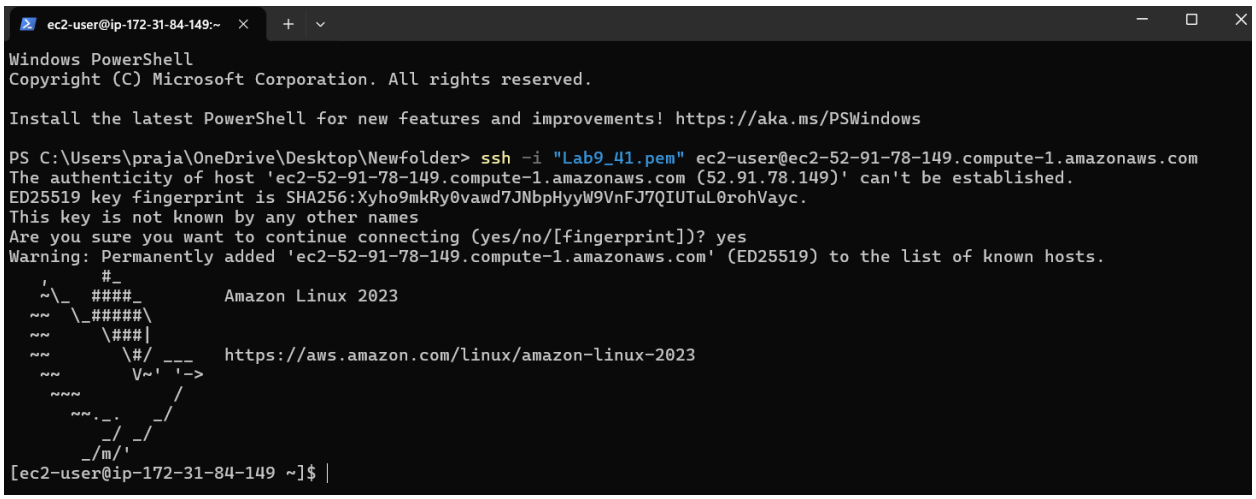
Step 4: Return to the instances screen and click on the instance ID of your instance. Then, click on "**Connect.**"



Click on "**SSH client**" and copy the example command provided.



Step 5: Now, we need to connect our local terminal to the instance using SSH. Open the terminal where your private key file (.pem) is located by actually going to the folder which has the .pem file, paste the copied SSH command, and run it.



Name: Aditya Dubey **Div: D15C** **Roll No: 10**

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 to ensure that all libraries are with up to date with the latest features and security fixes

```
[ec2-user@ip-172-31-84-149 ~]$ sudo yum update
Last metadata expiration check: 0:30:14 ago on Sat Sep 28 07:51:31 2024.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-84-149 ~]$
```

Step 7: We are going to set up a web server software application called **Apache** and a programming language called **PHP** in this instance . To do this, run this command in your terminal

sudo yum install httpd php

```
[ec2-user@ip-172-31-84-149 ~]$ sudo yum install httpd php
Last metadata expiration check: 0:31:34 ago on Sat Sep 28 07:51:31 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
```

```
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-172-31-84-149 ~]$
```

Step 8: Now, we will **install the GCC compiler**, which is used for compiling and running C and C++ programs, along with the essential C libraries. To do this, enter the following command: **sudo yum install gcc glibc glibc-common**

```
[ec2-user@ip-172-31-84-149 ~]$ sudo yum install gcc glibc glibc-common
Last metadata expiration check: 0:34:20 ago on Sat Sep 28 07:51:31 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

Transaction Summary
================================================================================
Install  13 Packages

Total download size: 52 M
```

```
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-172-31-84-149 ~]$
```

Step 9: Next, we need to **install the GD library**, along with its development tools. This library helps with creating and manipulating images. For that, run this command **sudo yum install gd gd-devel**

```
[ec2-user@ip-172-31-84-149 ~]$ sudo yum install gd gd-devel
Last metadata expiration check: 0:36:28 ago on Sat Sep 28 07:51:31 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
```

```
  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

Complete!
[ec2-user@ip-172-31-84-149 ~]$
```

Step 10: Now, we create a user called '**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-172-31-84-149 ~]$ sudo adduser -m nagios
sudo passwd nagios
Changing password for user nagios.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
Sorry, passwords do not match.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[ec2-user@ip-172-31-84-149 ~]$
```

Password :Shivam2@

Step 11: Now, we need to create a user group named **nagcmd**, which will be used to execute Nagios commands. To do this, run the following command: **sudo groupadd nagcmd**

```
[ec2-user@ip-172-31-84-149 ~]$ sudo groupadd nagcmd
[ec2-user@ip-172-31-84-149 ~]$
```

Step 12: Next, we'll **add the users apache and nagios to the nagcmd group**. This allows them to execute Nagios commands.
sudo usermod -a -G nagcmd nagios
sudo usermod -a -G nagcmd apache

```
[ec2-user@ip-172-31-84-149 ~]$ sudo usermod -a -G nagcmd nagios
sudo usermod -a -G nagcmd apache
[ec2-user@ip-172-31-84-149 ~]$
```

Step 13: We'll create a directory called **downloads** to store the files related to the Nagios server that we download.

mkdir ~/downloads

cd ~/downloads

```
[ec2-user@ip-172-31-84-149 ~]$ mkdir ~/downloads
cd ~/downloads
[ec2-user@ip-172-31-84-149 downloads]$
```

Step 14: Now we need to install the latest versions of nagios-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.

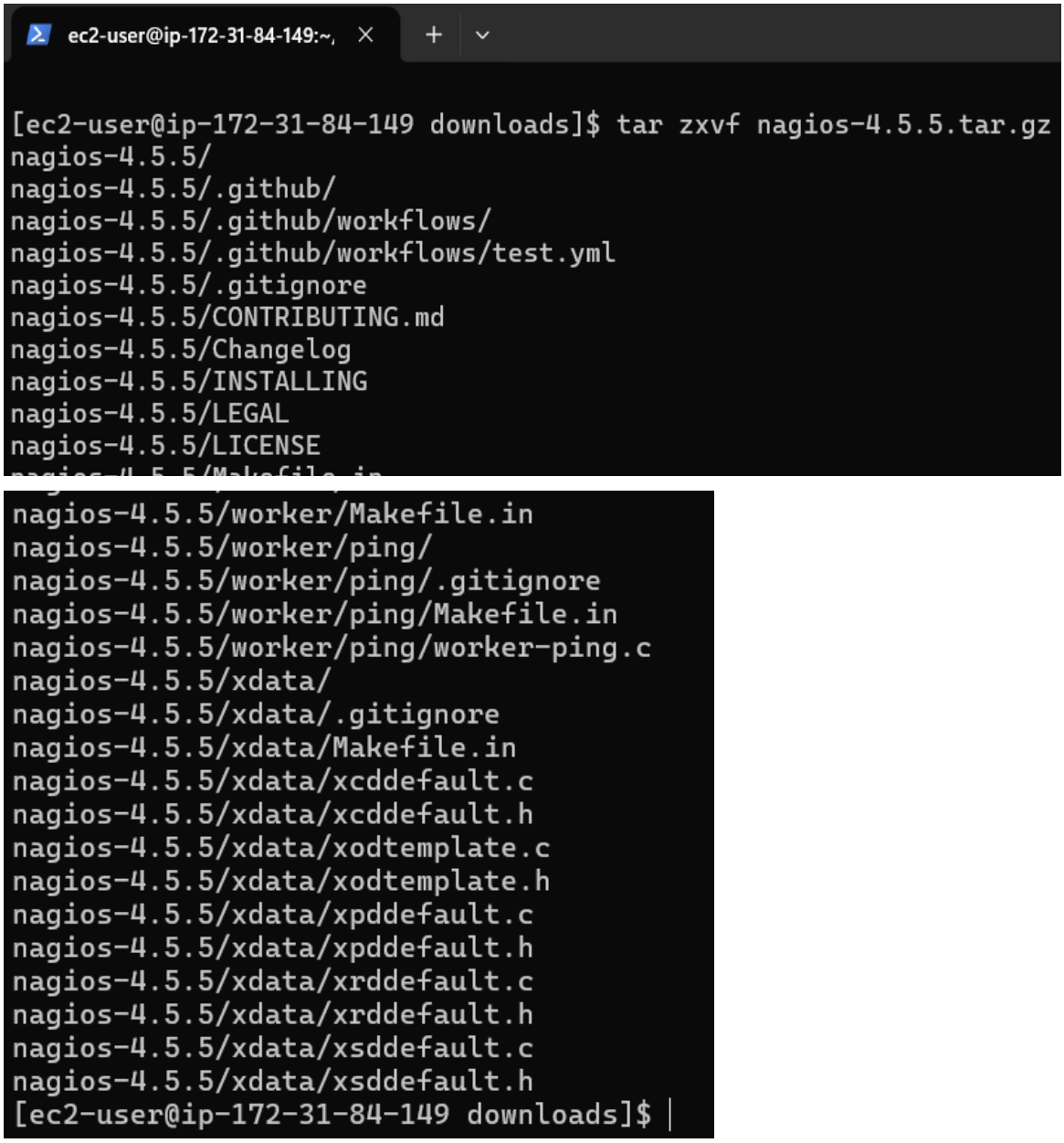
wget https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.5.tar.gz

```
[ec2-user@ip-172-31-84-149 downloads]$ wget https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.5.tar.gz
--2024-09-28 08:37:22--  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  5.07MB/s    in 0.4s

2024-09-28 08:37:23 (5.07 MB/s) - 'nagios-4.5.5.tar.gz' saved [2065473/2065473]

[ec2-user@ip-172-31-84-149 downloads]$
```

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

```
[ec2-user@ip-172-31-84-149 downloads]$ wget https://nagios-plugins.org/download/nagios-plugins-2.4.11.tar.gz
--2024-09-28 08:38:31--  https://nagios-plugins.org/download/nagios-plugins-2.4.11.tar.gz
Resolving nagios-plugins.org (nagios-plugins.org)... 45.56.123.251
Connecting to nagios-plugins.org (nagios-plugins.org)|45.56.123.251|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2753049 (2.6M) [application/x-gzip]
Saving to: 'nagios-plugins-2.4.11.tar.gz'

nagios-plugins-2.4.11.tar.gz  100%[=================================================>]   2.62M  7.16MB/s    in 0.4s

2024-09-28 08:38:32 (7.16 MB/s) - 'nagios-plugins-2.4.11.tar.gz' saved [2753049/2753049]

[ec2-user@ip-172-31-84-149 downloads]$
```

Step 15: Now, we need to extract the Nagios Core file into the same directory. We can do this using the tar command.
tar zxvf nagios-4.5.5.tar.gz



Step16: Now, we need to ensure that Nagios uses the nagcmd group for executing external commands.
./configure --with-command-group=nagcmd

Name: Aditya Dubey **Div: D15C** **Roll No: 10**

```
[ec2-user@ip-172-31-84-149 downloads]$ ./configure --with-command-group=nagcmd
-bash: ./configure: No such file or directory
[ec2-user@ip-172-31-84-149 downloads]$
```

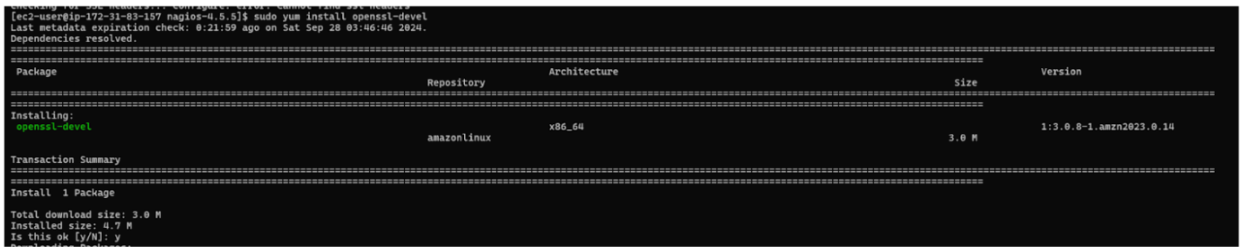
An error was encountered here: ./configure: no such path or directory . So Navigate to the nagios-4.5.5 folder in downloads. (version could vary)

ls :

```
[ec2-user@ip-172-31-84-149 downloads]$ ls
nagios-4.5.5  nagios-4.5.5.tar.gz  nagios-plugins-2.4.11.tar.gz
[ec2-user@ip-172-31-84-149 downloads]$
```

- 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 : **sudo yum install openssl-devel**




```
Last metadata expiration check: 0:58:10 ago on Sat Sep 28 07:51:31 2024.
Dependencies resolved.
================================================================================
 Package          Architecture    Version                    Repository    Size
================================================================================
Installing:
 openssl-devel    x86_64          1:3.0.8-1.amzn2023.0.14    amazonlinux   3.0 M

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

Total download size: 3.0 M
Installed size: 4.7 M
Is this ok [y/N]: y
Downloading Packages:
openssl-devel-3.0.8-1.amzn2023.0.14.x86_64.rpm   18 MB/s | 3.0 MB     00:00
--------------------------------------------------------------------------------
Total                                            14 MB/s | 3.0 MB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
```

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

```
[ec2-user@ip-172-31-84-149 nagios-4.5.5]$ ./configure --with-command-group=nagcmd
checking for a BSD-compatible install... /usr/bin/install -c
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether the compiler supports GNU C... yes
checking whether gcc accepts -g... yes
checking for gcc option to enable C11 features... none needed
checking whether make sets $(MAKE)... yes
checking whether ln -s works... yes
checking for strip... /usr/bin/strip
checking for sys/wait.h that is POSIX.1 compatible... yes
checking for stdio.h... yes
checking for stdlib.h... yes
checking for string.h... yes
checking for inttypes.h... yes
checking for stdint.h... yes
checking for strings.h... yes
checking for sys/stat.h... yes
checking for sys/types.h... 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-172-31-84-149 nagios-4.5.5]$
```

Step 17: Next, we need to compile all the components of the software based on the instructions in the Makefile. For that, use this command: **make all** Then,
sudo make install

Name: Aditya Dubey **Div: D15C** **Roll No: 10**

sudo make install-init
sudo make install-config
sudo make install-commandmode

```
[ec2-user@ip-172-31-84-149 nagios-4.5.5]$ make all
cd ./base && make
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/base'
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o nagios.o ./nagios.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o broker.o broker.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o nebmods.o nebmods.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o ../common/shared.o .
./common/shared.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o query-handler.o quer
y-handler.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o workers.o workers.c
In function 'get_wproc_list',
    inlined from 'get_worker' at workers.c:277:12:
workers.c:253:17: warning: '%s' directive argument is null [-Wformat-overflow=]
  253 |                 log_debug_info(DEBUGL_CHECKS, 1, "Found specialized worker(s) for '%s'", (slash && *slash != '/'
) ? slash : cmd_name);
      |                 ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o checks.o checks.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o config.o config.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o commands.o commands.
c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o events.o events.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o flapping.o flapping.
c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o logging.o logging.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o macros-base.o ../com
mon/macros.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o netutils.o netutils.
```

```
*** Support Notes *******************************************

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

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

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

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

For more information on obtaining support for Nagios, visit:

       https://support.nagios.com

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

Enjoy.

[ec2-user@ip-172-31-84-149 nagios-4.5.5]$ |
```

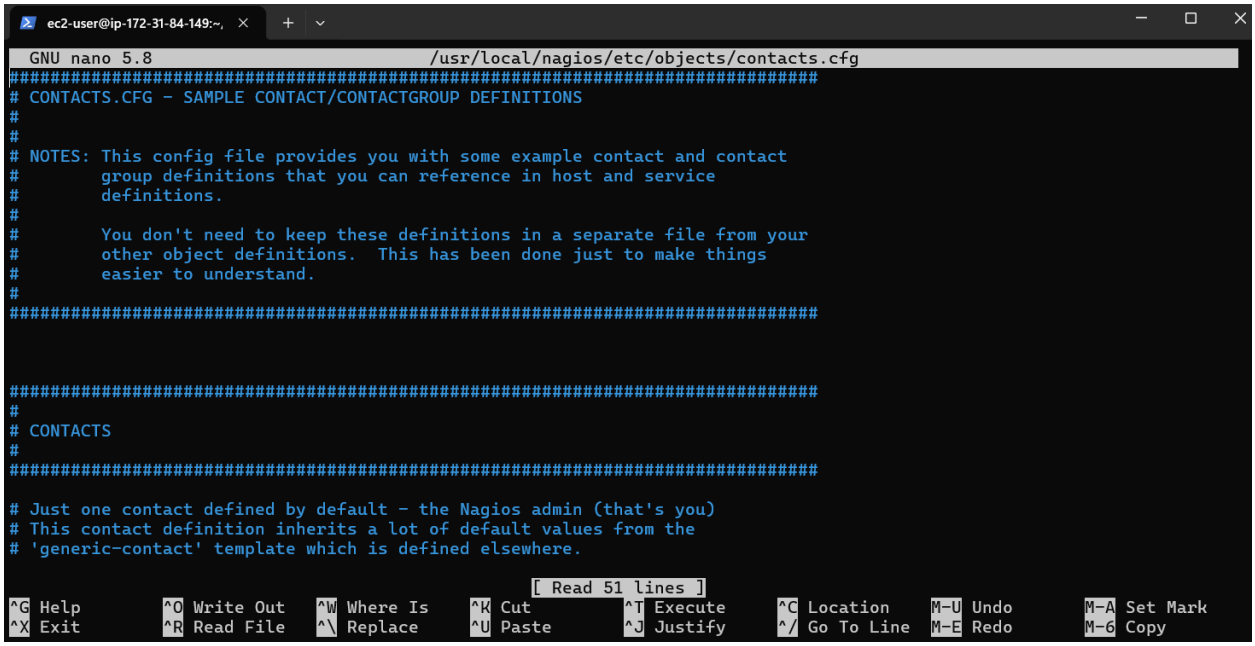
```
[ec2-user@ip-172-31-84-149 nagios-4.5.5]$ sudo make install
sudo make install-init
sudo make install-config
sudo make install-commandmode
cd ./base && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/base'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagios /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagiostats /usr/local/nagios/bin
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5/base'
cd ./cgi && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
make install-basic
make[2]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/sbin
for file in *.cgi; do \
        /usr/bin/install -c -s -m 775 -o nagios -g nagios $file /usr/local/nagios/sbin; \
done
make[2]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
cd ./html && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/html'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/media
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/stylesheets
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/contexthelp
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/docs
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/docs/images
```

```
*** 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 ***

[ec2-user@ip-172-31-84-149 nagios-4.5.5]$
```

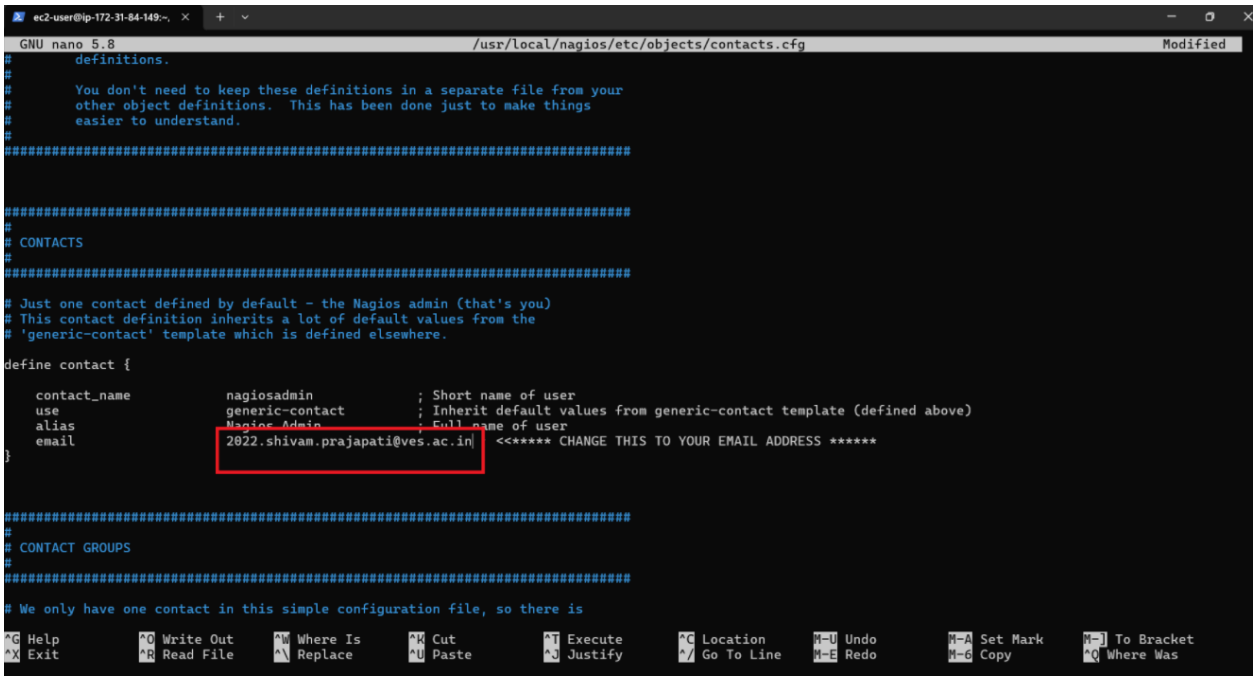
Step 18: We need to update the email linked with this server to our email for it to send notifications (if any needed). **sudo nano /usr/local/nagios/etc/objects/contacts.cfg**

```
[ec2-user@ip-172-31-84-149 nagios-4.5.5]$ sudo nano /usr/local/nagios/etc/objects/contacts.cfg
```

Name: Aditya Dubey **Div: D15C** **Roll No: 10**



Here, change the email under ‘define contact{}’ to your email address



To save this use the following shortcut sequence CTRL+O→Enter→CTRL+X.
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.
sudo make install-webconf

```
[ec2-user@ip-172-31-84-149 nagios-4.5.5]$ [ec2-user@ip-172-31-84-149 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-172-31-84-149 nagios-4.5.5]$
```

Step 20: Now we need to create a user to access the Nagios web interface. For that, run this command to create a user named '**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-172-31-84-149 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-172-31-84-149 nagios-4.5.5]$
```

Kingmaker is the password

Step 21: Now, we need to restart the Apache server to apply all the recent configurations. Use this command: **sudo service httpd restart**

```
[ec2-user@ip-172-31-84-149 nagios-4.5.5]$ sudo service httpd restart
Redirecting to /bin/systemctl restart httpd.service
[ec2-user@ip-172-31-84-149 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)

```
[ec2-user@ip-172-31-84-149 nagios-4.5.5]$ cd ~/downloads
[ec2-user@ip-172-31-84-149 downloads]$
```

```
[ec2-user@ip-172-31-84-149 downloads]$ tar zxvf nagios-plugins-2.4.11.tar.gz
nagios-plugins-2.4.11/
nagios-plugins-2.4.11/build-aux/
nagios-plugins-2.4.11/build-aux/compile
nagios-plugins-2.4.11/build-aux/config.guess
nagios-plugins-2.4.11/build-aux/config.rpath
nagios-plugins-2.4.11/build-aux/config.sub
nagios-plugins-2.4.11/build-aux/install-sh
nagios-plugins-2.4.11/build-aux/ltmain.sh
nagios-plugins-2.4.11/build-aux/missing
nagios-plugins-2.4.11/build-aux/mkinstalldirs
nagios-plugins-2.4.11/build-aux/depcomp
nagios-plugins-2.4.11/build-aux/snippet/
nagios-plugins-2.4.11/build-aux/snippet/_Noreturn.h
nagios-plugins-2.4.11/build-aux/snippet/arg-nonnull.h
nagios-plugins-2.4.11/build-aux/snippet/c++defs.h
nagios-plugins-2.4.11/build-aux/snippet/warn-on-use.h
nagios-plugins-2.4.11/build-aux/test-driver
```

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

```
[ec2-user@ip-172-31-84-149 downloads]$ cd nagios-plugins-2.4.11
[ec2-user@ip-172-31-84-149 nagios-plugins-2.4.11]$
```

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

```
[ec2-user@ip-172-31-84-149 nagios-plugins-2.4.11]$ ./configure --with-nagios-user=nagios --with-nagios-group=nagios
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /usr/bin/mkdir -p
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking whether make supports nested variables... yes
checking whether to enable maintainer-specific portions of Makefiles... yes
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking whether gcc understands -c and -o together... yes
checking whether make supports the include directive... yes (GNU style)
checking dependency style of gcc... gcc3
checking how to run the C preprocessor... gcc -E
checking for grep that handles long lines and -e... /usr/bin/grep
checking for egrep... /usr/bin/grep -E
```

Step 24: We need to compile all the components of this software based on the instructions in the Makefile.

make
sudo make install

```
[ec2-user@ip-172-31-84-149 nagios-plugins-2.4.11]$ make
make  all-recursive
make[1]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11'
Making all in gl
make[2]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11/gl'
make  all-recursive
make[3]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11/gl'
make[4]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11/gl'
make[4]: Nothing to be done for 'all-am'.
```

```
[ec2-user@ip-172-31-84-149 nagios-plugins-2.4.11]$ sudo make install
Making install in gl
make[1]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11/gl'
make  install-recursive
make[2]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11/gl'
make[3]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11/gl'
make[4]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11/gl'
if test yes = no; then \
  case 'linux-gnu' in \
    darwin[56]*) \
      need_charset_alias=true ;; \
    darwin* | cygwin* | mingw* | pw32* | cegcc*) \
      need_charset_alias=false ;; \
    *) \
      need_charset_alias=true ;; \
  esac ; \
```

```
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-172-31-84-149 nagios-plugins-2.4.11]$
```

Step 25: We need to register the Nagios service with the system to enable it to manage the server status
sudo chkconfig --add nagios
sudo chkconfig nagios on

```
[ec2-user@ip-172-31-84-149 nagios-plugins-2.4.11]$ sudo chkconfig --add nagios
sudo chkconfig nagios on
error reading information on service nagios: No such file or directory
Note: Forwarding request to 'systemctl enable nagios.service'.
Created symlink /etc/systemd/system/multi-user.target.wants/nagios.service → /usr/lib/systemd/system/nagios.service.
```

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

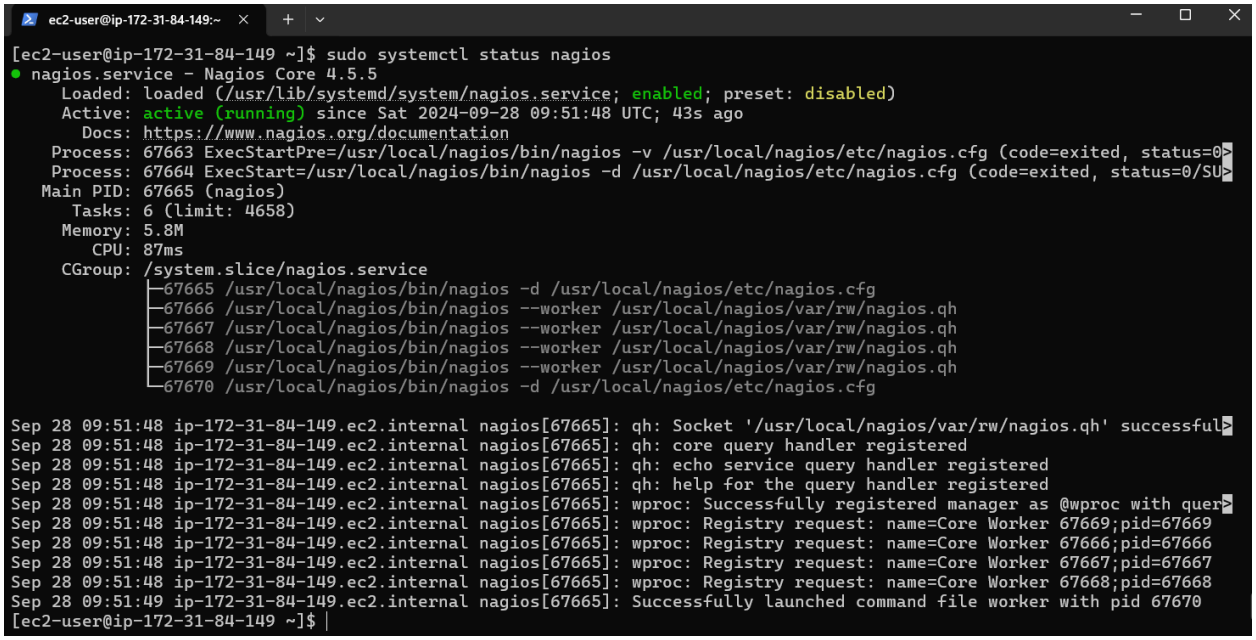
```
[ec2-user@ip-172-31-84-149 nagios-plugins-2.4.11]$ sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

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

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

Running pre-flight check on configuration data...

Checking objects...
        Checked 8 services.
        Checked 1 hosts.
        Checked 1 host groups.
        Checked 0 service groups.
        Checked 1 contacts.
        Checked 1 contact groups.
        Checked 24 commands.
        Checked 5 time periods.
        Checked 0 host escalations.
        Checked 0 service escalations.
Checking for circular paths...
```

sudo service nagios start

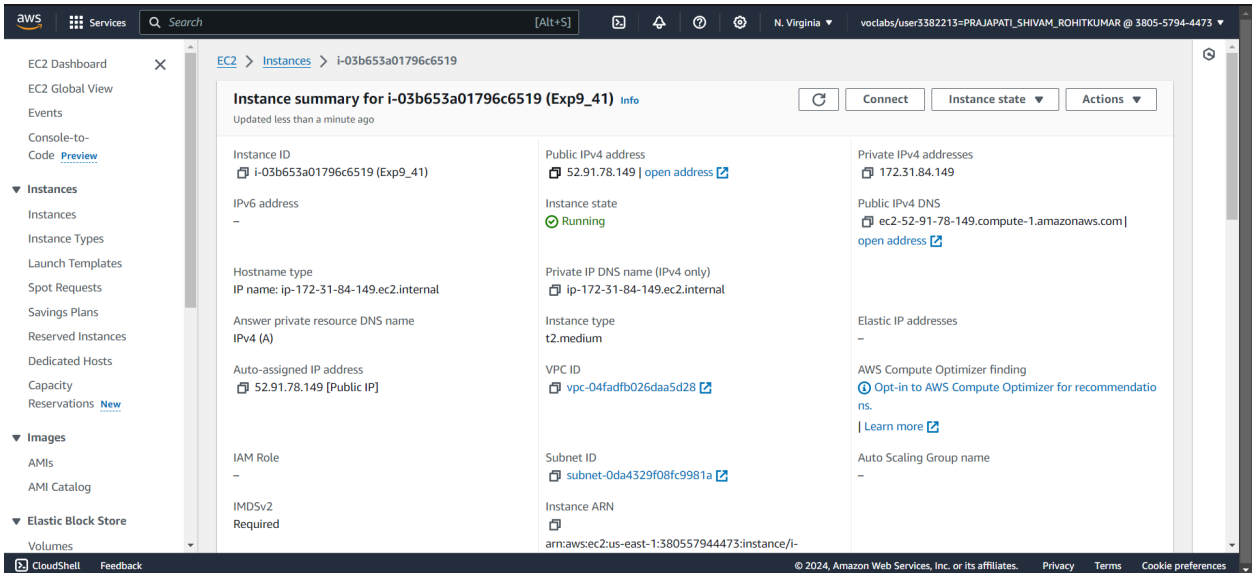
```
[ec2-user@ip-172-31-84-149 nagios-plugins-2.4.11]$ cd
[ec2-user@ip-172-31-84-149 ~]$ sudo service nagios start
Redirecting to /bin/systemctl start nagios.service
[ec2-user@ip-172-31-84-149 ~]$
```

Step 27: Check the status of the nagios.

Name: Aditya Dubey **Div: D15C** **Roll No: 10**

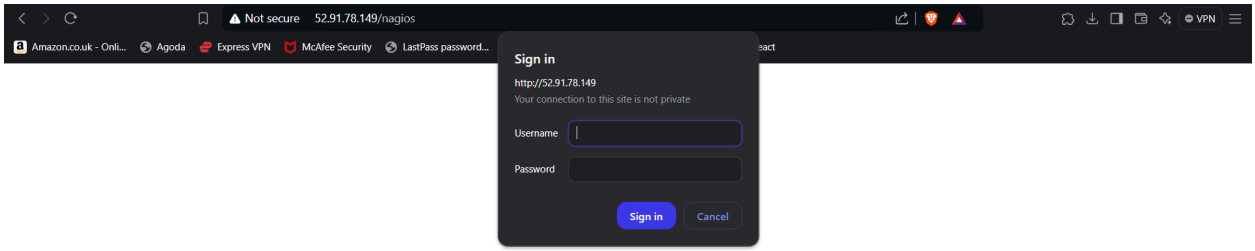


Step 28: Go back to EC2 Console and copy the Public IP address of this instance. Open up your browser and look for **http://<your_public_ip_address>/nagios**



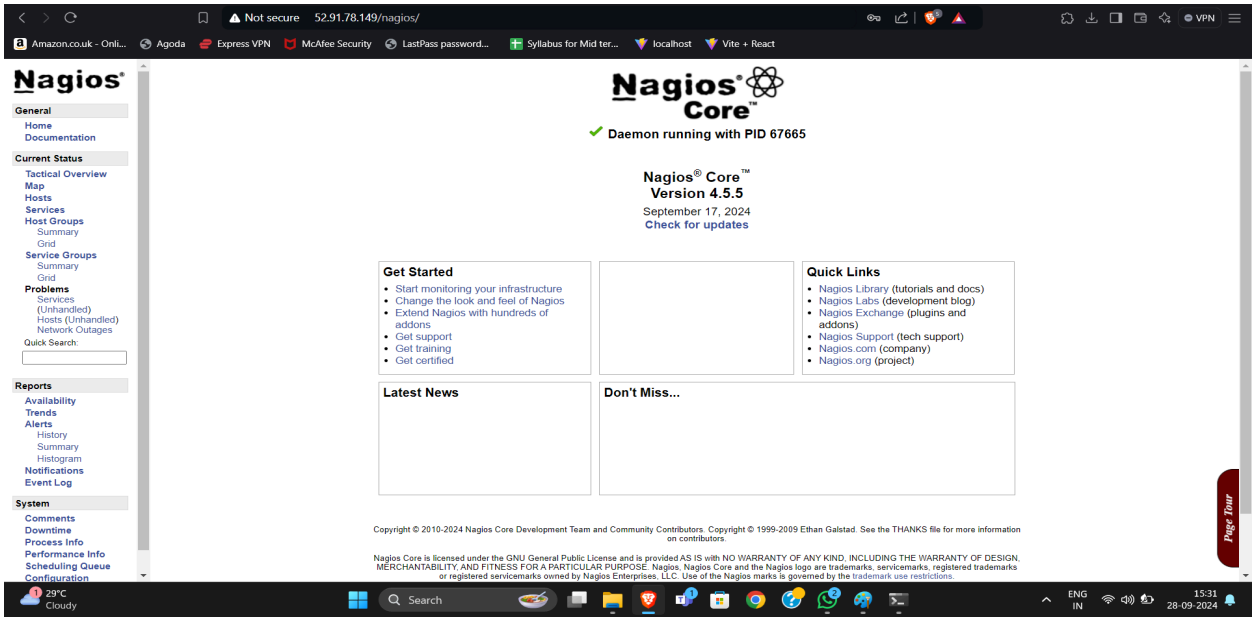
Name: Aditya Dubey **Div: D15C** **Roll No: 10**

http://52.91.78.149/nagios.



Enter **username as nagiosadmin** and **password as Kingmaker.**

Step 29: After entering the correct credentials, you will see this page



CONCLUSION:

In this experiment, we have learned how to install and configure Nagios Core, Nagios Plugins, and NRPE on a Linux machine. We used an Amazon Linux OS instance with the necessary security rules in place. It's important to ensure that the links for Nagios Core and Nagios Plugins are up to date (when using wget). After extracting and configuring these files, we should check for any issues before starting the server. Once everything is set up, we can start the Nagios server. By using the public IP address of the EC2 instance, we can access the Nagios dashboard by navigating to that IP followed by /nagios.

Name: Aditya Dubey **Div: D15C** **Roll No: 10**

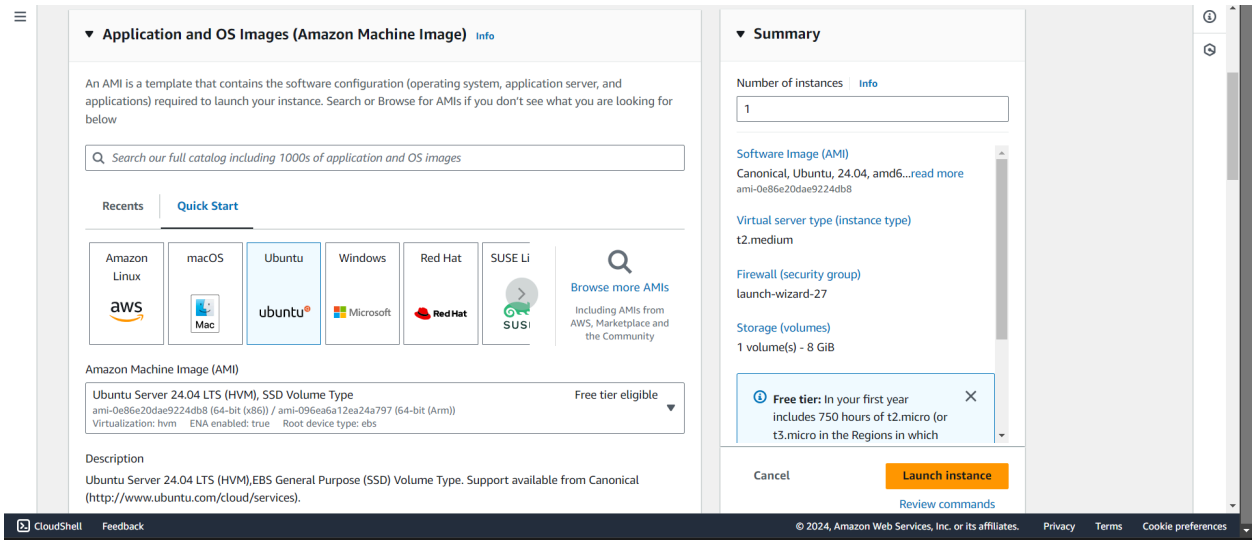
Experiment No:10

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

PREREQUISITES : We should have an Amazon Linux instance with nagios already set up.

Step 1: Set up ubuntu instance

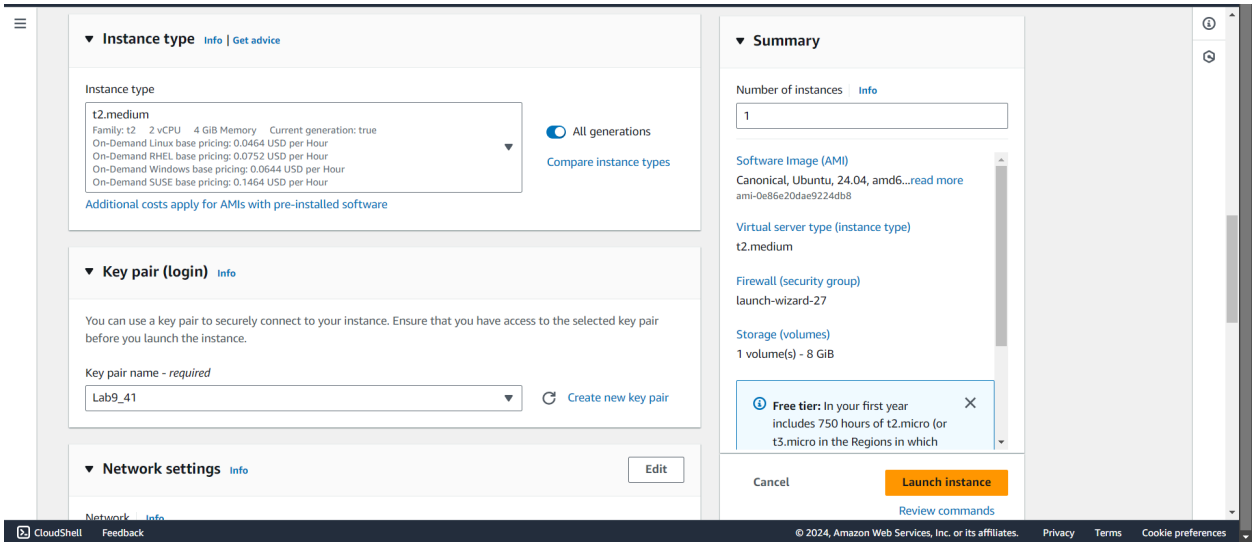
1) Log in to your AWS account. Look for EC2 in the services menu. Open the interface and select Create Instance.

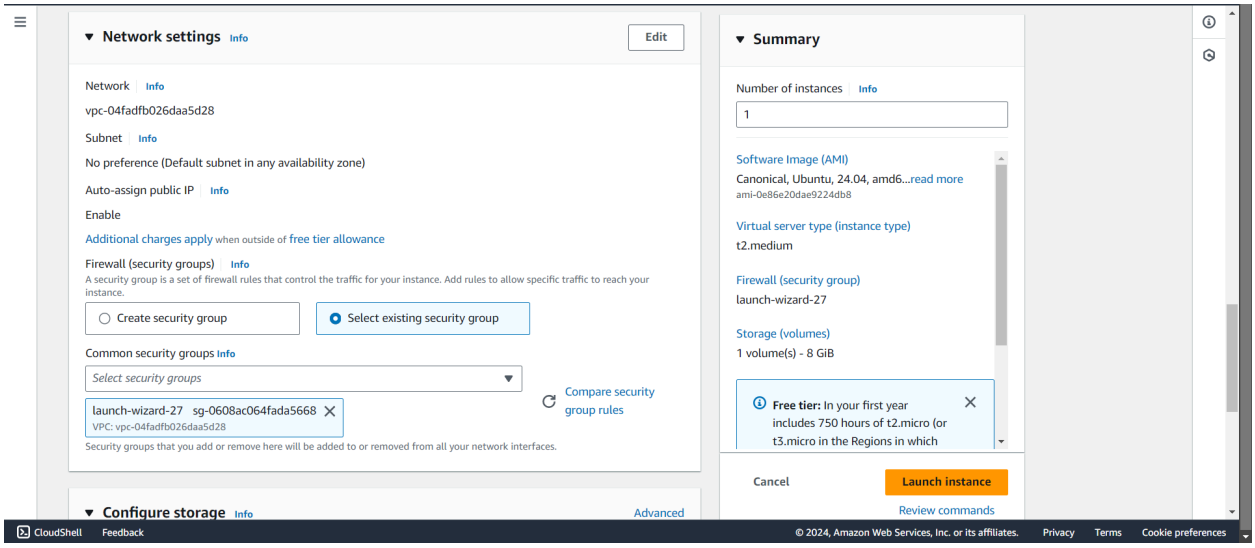


Select The OS Image as Ubuntu.

2) Ensure that you choose the same private key you created for the Amazon Linux instance. Additionally, select the same security group that you configured for the Linux instance.

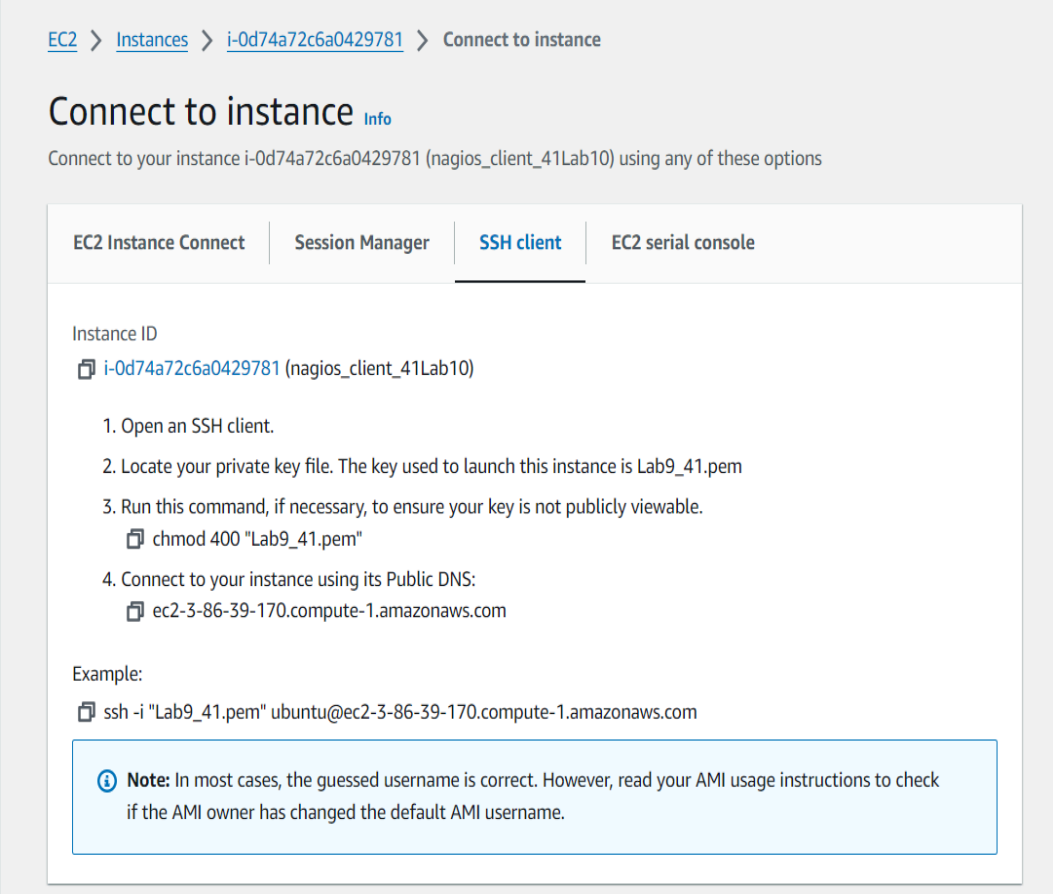
Name: Aditya Dubey **Div: D15C** **Roll No: 10**



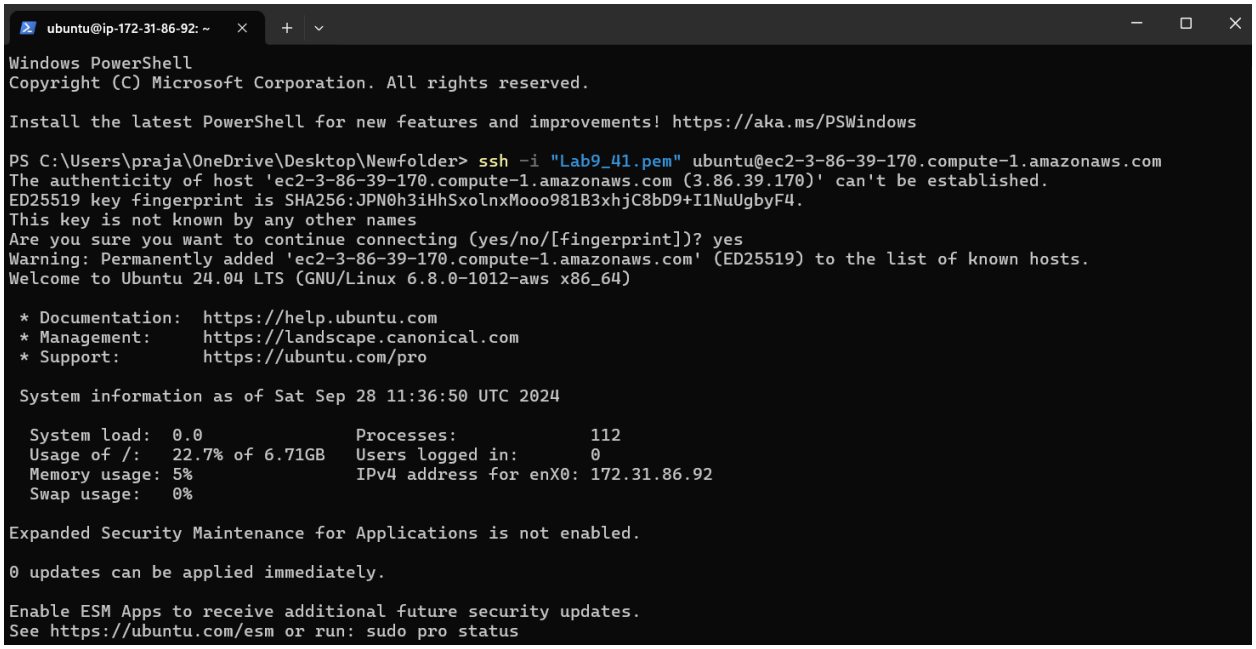


Instance is:

3) Now return to the instances screen. Click on the instance ID of your instance, then select Connect. Click on SSH client and copy the example command. Next, we need to connect our local OS terminal to the instance using SSH. To do this, open the terminal where the private key file (.pem) is stored. Paste the copied SSH command and execute it.



Copy the example command



*Successfully connected the instance via **SSH***

Step 2: On Nagios Host machine (Linux) execute the following which we have already created as a prerequisites:

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-172-31-84-149 ~]$ ps -ef | grep nagios
nagios     67665       1  0 09:51 ?        00:00:00 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
nagios     67666   67665  0 09:51 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios     67667   67665  0 09:51 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios     67668   67665  0 09:51 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios     67669   67665  0 09:51 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios     67670   67665  0 09:51 ?        00:00:00 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
ec2-user   70887    3276  0 10:48 pts/0    00:00:00 grep --color=auto nagios
[ec2-user@ip-172-31-84-149 ~]$
```

2) Next, switch to the root user and create a directory at the path '/usr/local/nagios/etc/objects/monitorhosts/linuxhosts'.

sudo su
mkdir -p /usr/local/nagios/etc/objects/monitorhosts/linuxhosts

```
[ec2-user@ip-172-31-84-149 ~]$ sudo su
mkdir /usr/local/nagios/etc/objects/monitorhosts
mkdir /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
[root@ip-172-31-84-149 ec2-user]#
```

3) We need to create a configuration file in this directory. To do this, copy the contents of the existing localhost configuration into the new file named **'linuxserver.cfg'.**

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

```
[root@ip-172-31-84-149 ec2-user]# cp /usr/local/nagios/etc/objects/localhost.cfg /usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg
cp: cannot create regular file '/usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg': No such file or directory
```

So make the second directory again and run the cp command

```
[root@ip-172-31-84-149 ec2-user]# mkdir -p /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
[root@ip-172-31-84-149 ec2-user]# cp /usr/local/nagios/etc/objects/localhost.cfg /usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg
[root@ip-172-31-84-149 ec2-user]#
```

We need to make some changes in this config file. Open it using a nano editor.

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

```
[root@ip-172-31-84-149 ec2-user]# 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


    host_name               linuxserver
    alias                   linuxserver
    address                 3.86.39.170
}
```

Change hostgroup_name to **linux-servers1**

```
# Define an optional hostgroup for Linux machines

define hostgroup {

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



##############################################################################
#
# SERVICE DEFINITIONS
#
##############################################################################
```

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

Now, we need to edit the nagios configuration file to add this directory. Run this command

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

```
[root@ip-172-31-84-149 ec2-user]# nano /usr/local/nagios/etc/nagios.cfg
```

and add the following line **cfg_dir=/usr/local/nagios/etc/objects/monitorhosts/**

```
  GNU nano 5.8                          /usr/local/nagios/etc/nagios.cfg                          Modified
# host groups, contacts, contact groups, services, etc.
# You can split your object definitions across several config files
# if you wish (as shown below), or keep them all in a single config file.

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

# 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/


# OBJECT CACHE FILE
# This option determines where object definitions are cached when

^G Help          ^O Write Out     ^W Where Is      ^K Cut           ^T Execute       ^C Location      M-U Undo         M-A Set Mark     M-] To Bracket
^X Exit          ^R Read File     ^\ Replace       ^U Paste         ^J Justify       ^/ Go To Line    M-E Redo         M-6 Copy         ^Q Where Was
```

Now we verify the configuration files. **/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg**

```
[root@ip-172-31-84-149 ec2-user]# /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

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

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

Running pre-flight check on configuration data...

Checking objects...
        Checked 16 services.
        Checked 2 hosts.
        Checked 2 host groups.
        Checked 0 service groups.
        Checked 1 contacts.
        Checked 1 contact groups.
```

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

Total Warnings: 0
Total Errors:   0

Things look okay - No serious problems were detected during the pre-flight check
[root@ip-172-31-84-149 ec2-user]#
```

Once the files are verified, we need to restart the server: **service nagios restart**

```
[root@ip-172-31-84-149 ec2-user]# service nagios restart
Redirecting to /bin/systemctl restart nagios.service
[root@ip-172-31-84-149 ec2-user]#
```

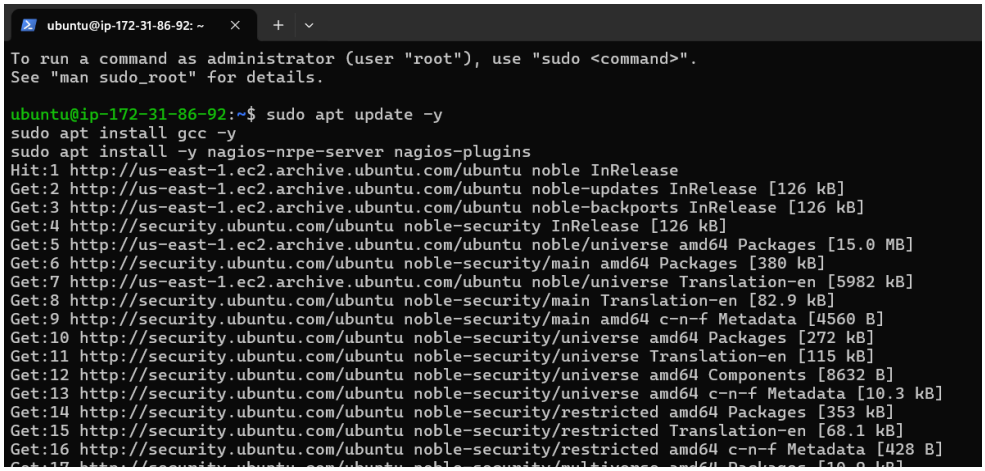
```
[root@ip-172-31-84-149 ec2-user]# service nagios restart
Redirecting to /bin/systemctl restart nagios.service
[root@ip-172-31-84-149 ec2-user]# sudo systemctl status nagios
● nagios.service - Nagios Core 4.5.5
     Loaded: loaded (/usr/lib/systemd/system/nagios.service; enabled; preset: disabled)
     Active: active (running) since Sat 2024-09-28 11:30:31 UTC; 3min 57s ago
       Docs: https://www.nagios.org/documentation
    Process: 73417 ExecStartPre=/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
    Process: 73418 ExecStart=/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
   Main PID: 73419 (nagios)
      Tasks: 6 (limit: 4658)
     Memory: 4.2M
        CPU: 113ms
     CGroup: /system.slice/nagios.service
             ├─73419 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─73420 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─73421 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─73422 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─73423 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             └─73425 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg

Sep 28 11:30:31 ip-172-31-84-149.ec2.internal nagios[73419]: wproc: Successfully registered manager as @wproc with query handler
Sep 28 11:30:31 ip-172-31-84-149.ec2.internal systemd[1]: Started nagios.service - Nagios Core 4.5.5.
Sep 28 11:30:31 ip-172-31-84-149.ec2.internal nagios[73419]: wproc: Registry request: name=Core Worker 73423;pid=73423
Sep 28 11:30:31 ip-172-31-84-149.ec2.internal nagios[73419]: wproc: Registry request: name=Core Worker 73421;pid=73421
Sep 28 11:30:31 ip-172-31-84-149.ec2.internal nagios[73419]: wproc: Registry request: name=Core Worker 73420;pid=73420
Sep 28 11:30:31 ip-172-31-84-149.ec2.internal nagios[73419]: wproc: Registry request: name=Core Worker 73422;pid=73422
Sep 28 11:30:31 ip-172-31-84-149.ec2.internal nagios[73419]: Successfully launched command file worker with pid 73425
Sep 28 11:32:23 ip-172-31-84-149.ec2.internal nagios[73419]: SERVICE ALERT: linuxserver;HTTP;CRITICAL;SOFT;1;connect to address 3.86.39.170 and port 80: Co>
Sep 28 11:33:23 ip-172-31-84-149.ec2.internal nagios[73419]: SERVICE ALERT: linuxserver;HTTP;CRITICAL;SOFT;2;connect to address 3.86.39.170 and port 80: Co>
Sep 28 11:34:23 ip-172-31-84-149.ec2.internal nagios[73419]: SERVICE ALERT: linuxserver;HTTP;CRITICAL;SOFT;3;connect to address 3.86.39.170 and port 80: Co>
lines 1-28/28 (END)
```

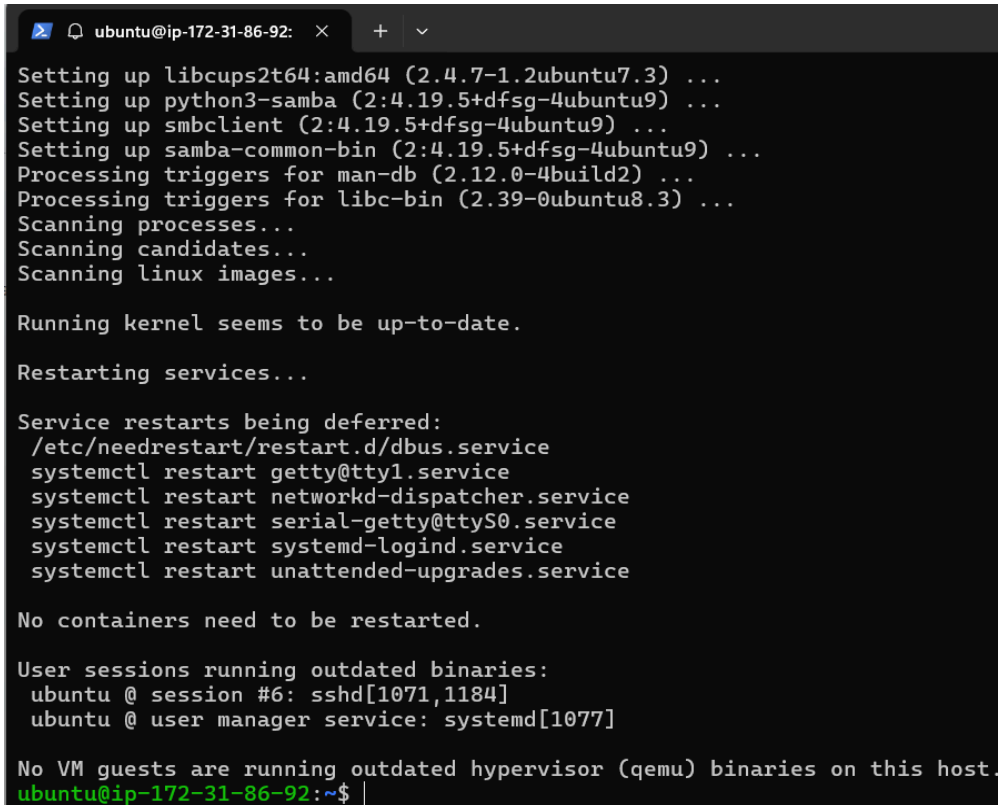
Name: Aditya Dubey **Div: D15C** **Roll No: 10**

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

1) First, check for any available updates, and then proceed to install gcc, the Nagios NRPE server, and Nagios plugins.

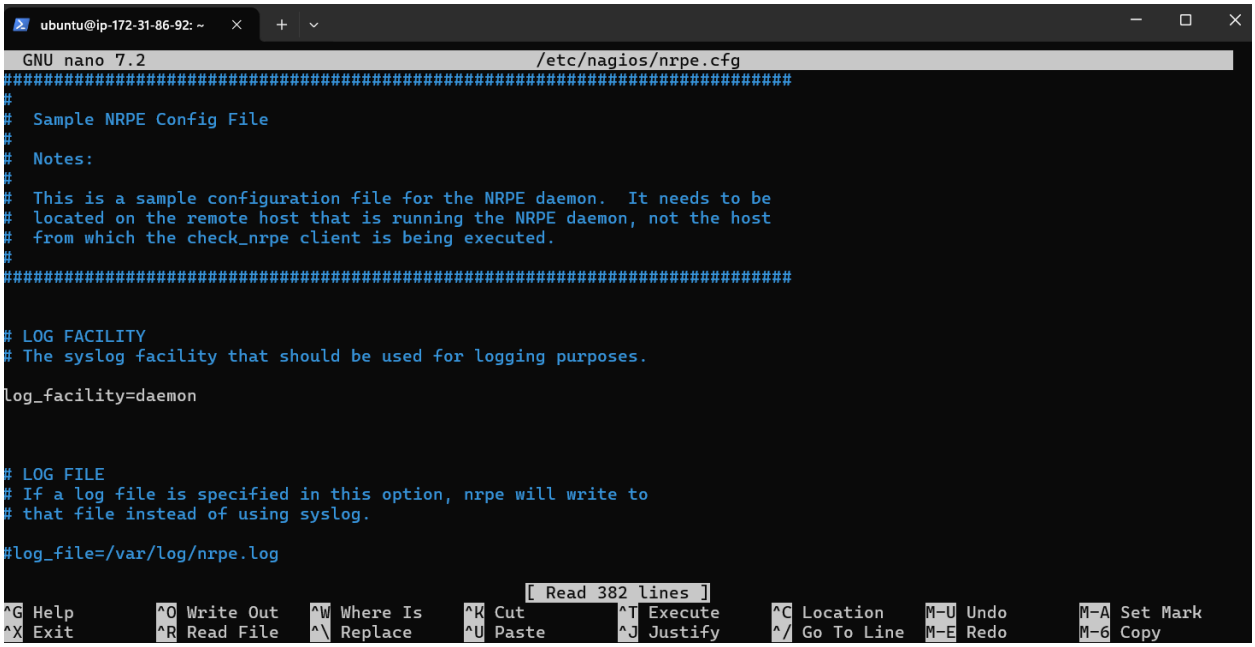
sudo apt update -y
sudo apt install gcc -y
sudo apt install -y nagios-nrpe-server nagios-plugins



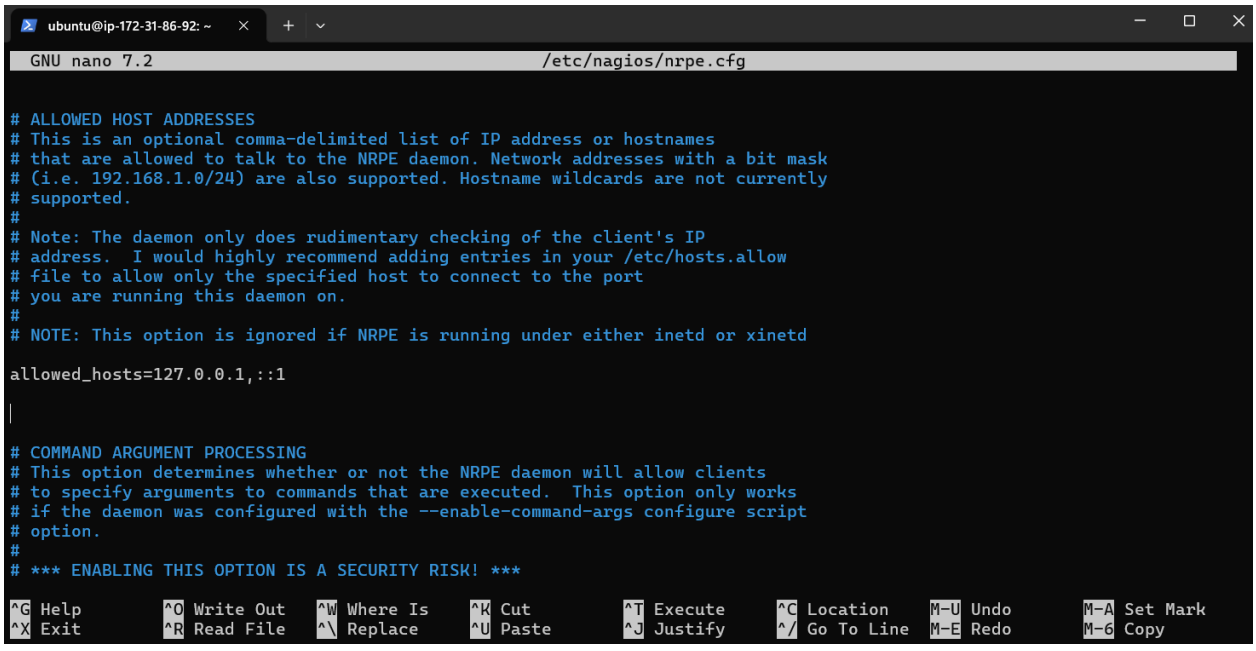


Name: Aditya Dubey **Div: D15C** **Roll No: 10**

2) We need to include the public IP address of our Nagios host machine (Linux) in the NRPE configuration file. **sudo nano /etc/nagios/nrpe.cfg**

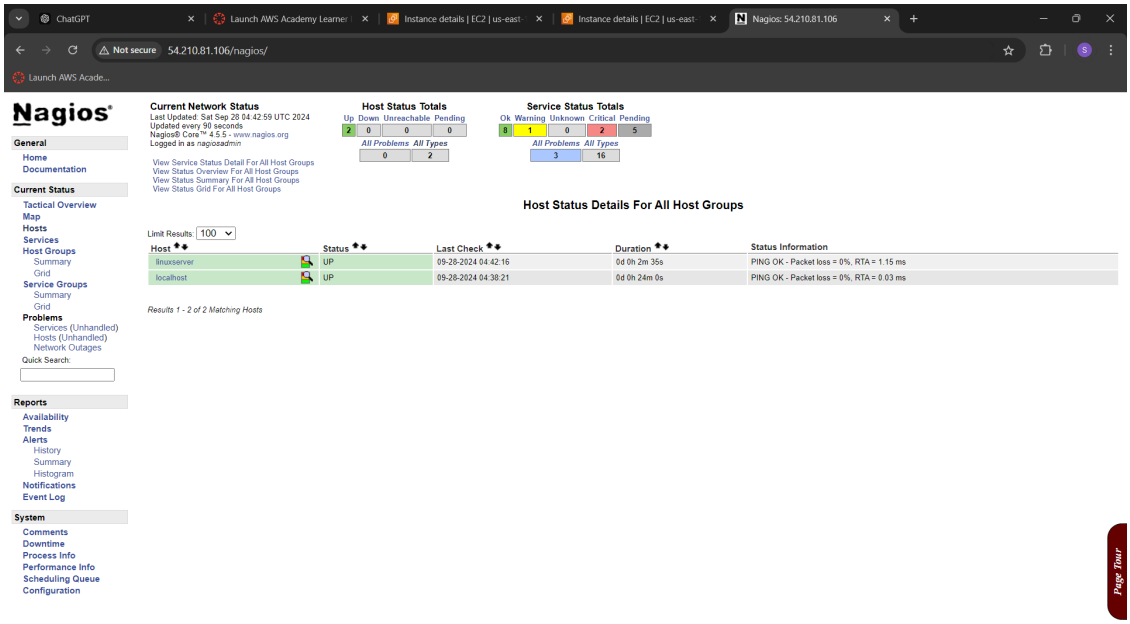


Under allowed_hosts, add the nagios host ip address (public)

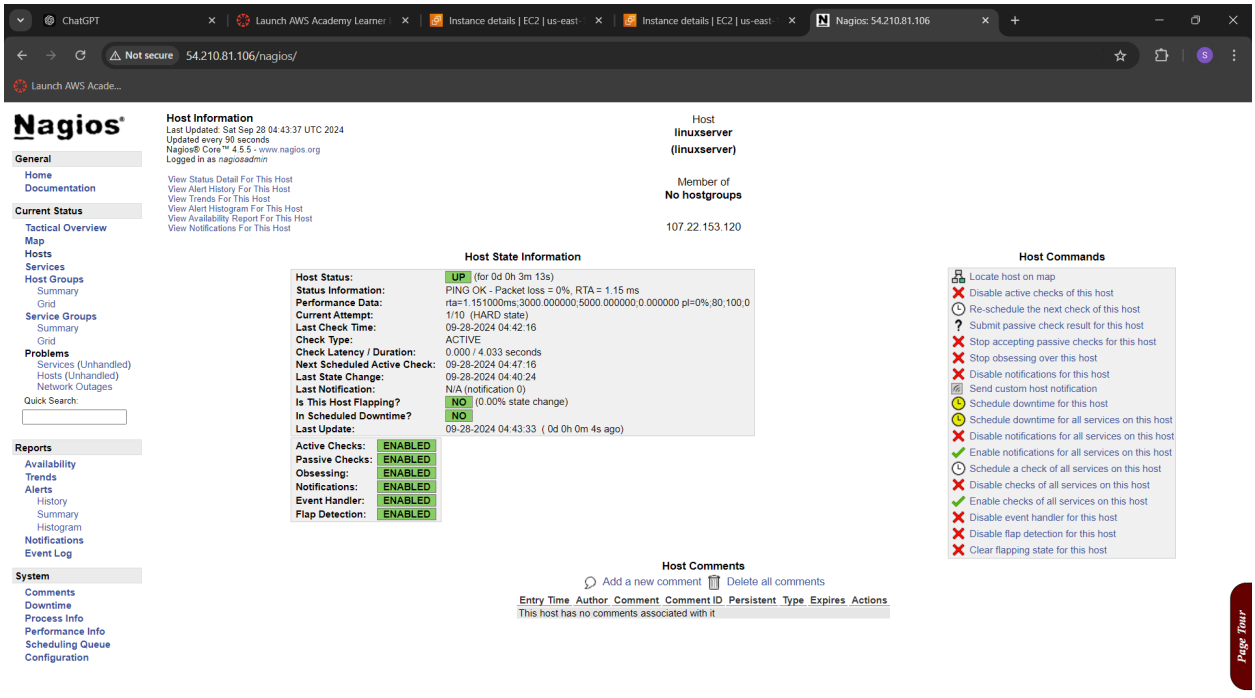


Name: Aditya Dubey **Div: D15C** **Roll No: 10**

Step 4: Check the Nagios Dashboard.Go to Nagios dashboard, click on hosts.Here, we can see that the linuxserver is also added as a host.

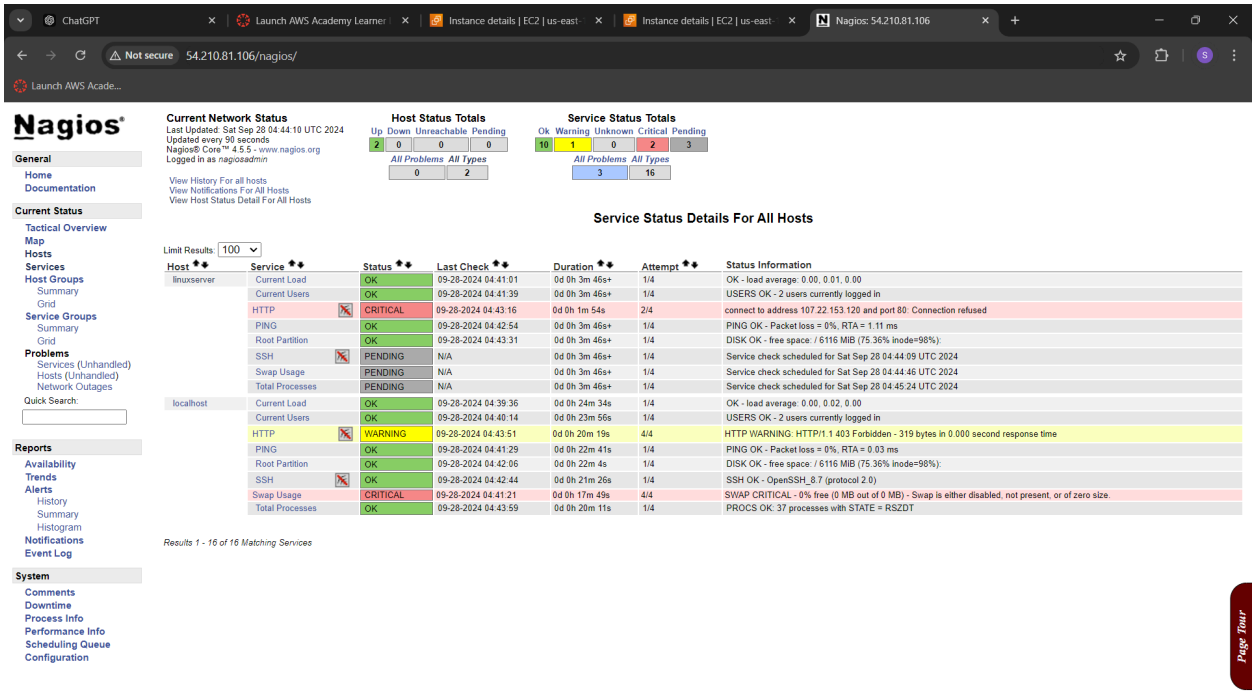


Click on linuxserver. we can check all the information about linuxserver host.



Name: Aditya Dubey **Div: D15C** **Roll No: 10**

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



CONCLUSION:

In this experiment, we learned to conduct port service monitoring and server monitoring using Nagios. To do this, we require a Linux instance to host the Nagios dashboard and a separate Ubuntu instance linked as a second host. We need to configure the Linux instance and include the IP address of the Ubuntu instance. Subsequently, we must replicate the initial setup from the Linux instance on the Ubuntu instance by adding the IP address of the Linux instance to the list of allowed hosts. After restarting the NRPE server, we should see the ‘linuxserver’ host listed.

Name: Aditya Dubey **Div: D15C** **Roll No: 10**

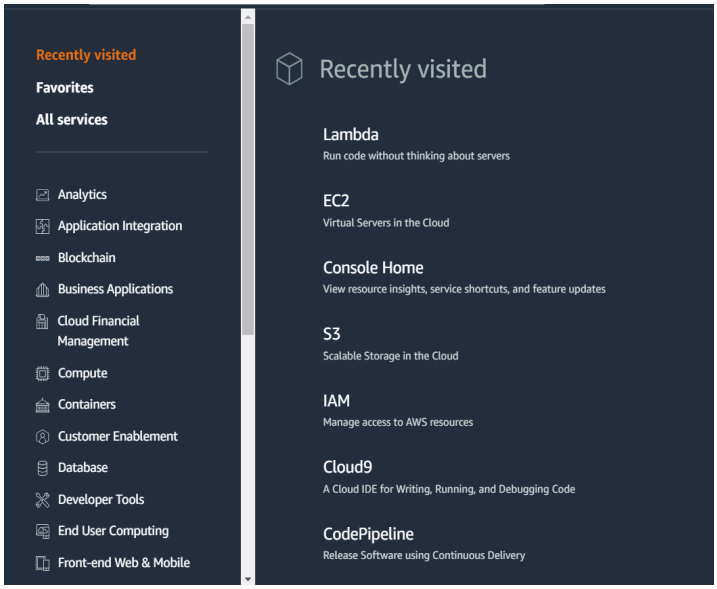
Name: Aditya Dubey **Div: D15C** **Roll No: 10**

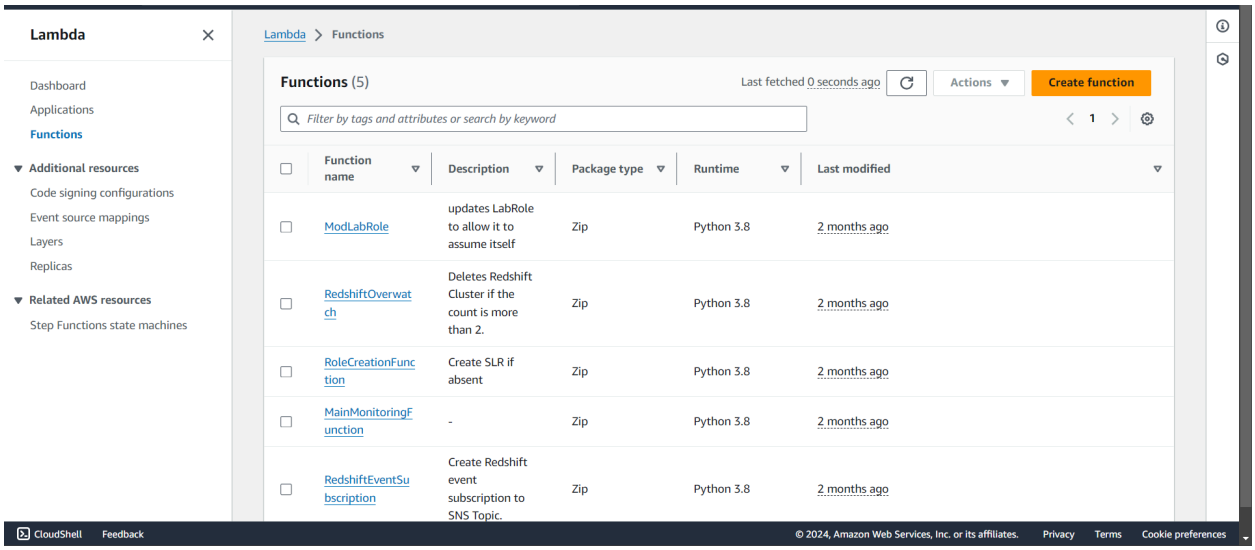
Experiment No: 11

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

CREATION OF LAMBDA FUNCTION:

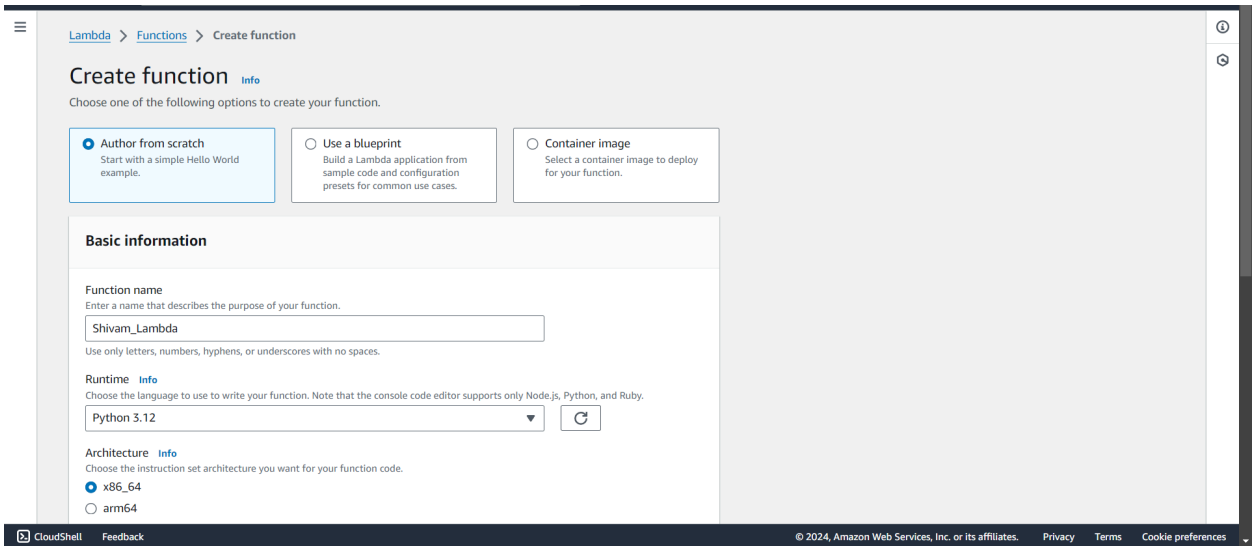
Step1: Log in to your AWS Personal or Academy account. Navigate to Lambda, then select the 'Create Function' button



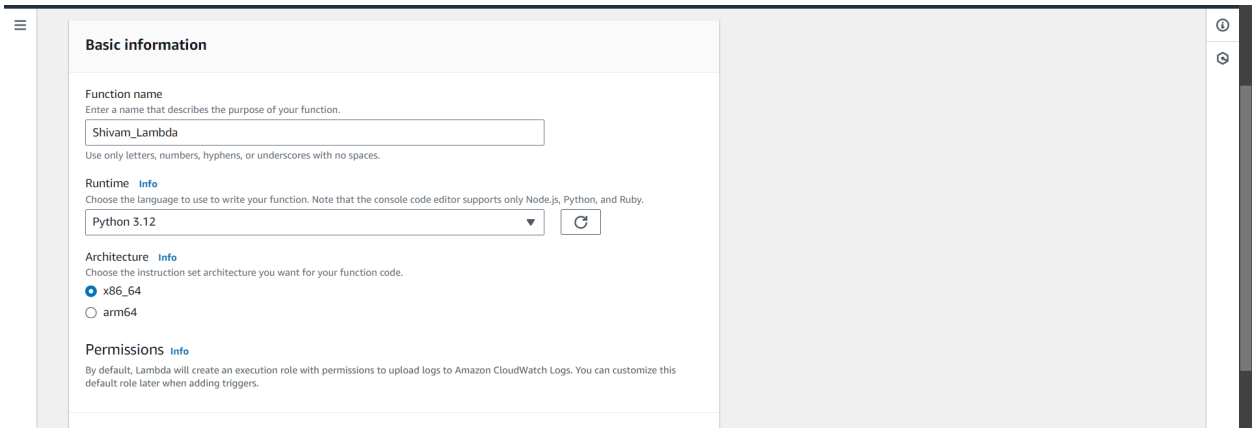


Name: Aditya Dubey **Div: D15C** **Roll No: 10**

Step 2: Give your Lambda function a name and choose a programming language. The code editor only supports Node.js, Python, and Ruby, so in my case I have chosen **Python 3.12**. Set the **architecture to x86**. For the execution role, select '**Use an existing role**,' then pick '**Lab role**' from the dropdown menu under existing roles .
(This is because the Lab role already has the permissions needed for Lambda to run properly, so you don't need to create a new role from scratch. It's a quicker and more convenient option)

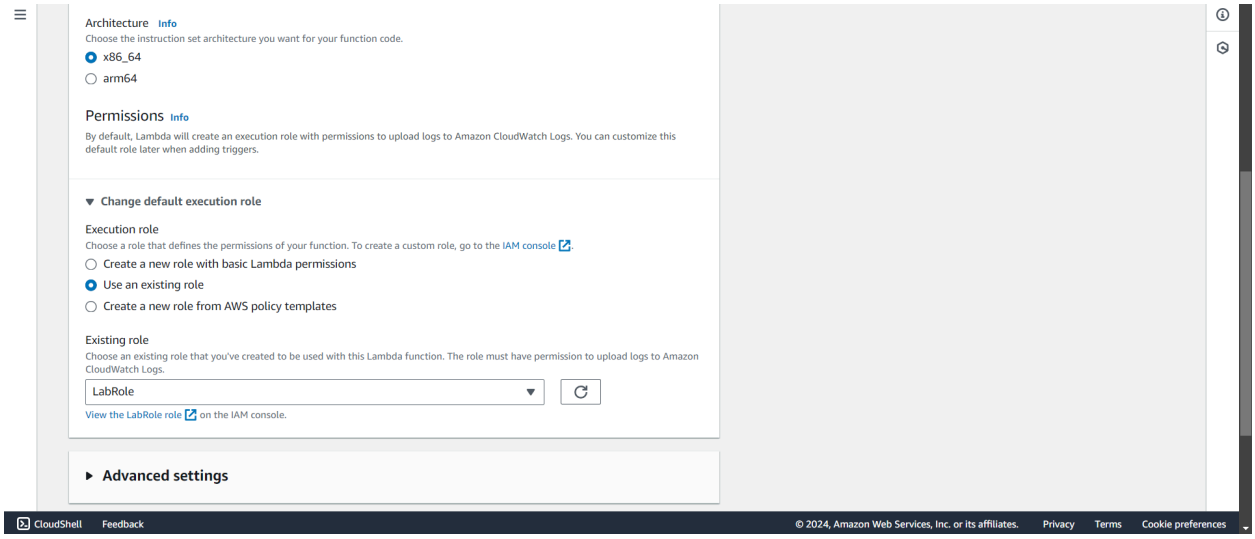


Give the function name and select required language for lambda function

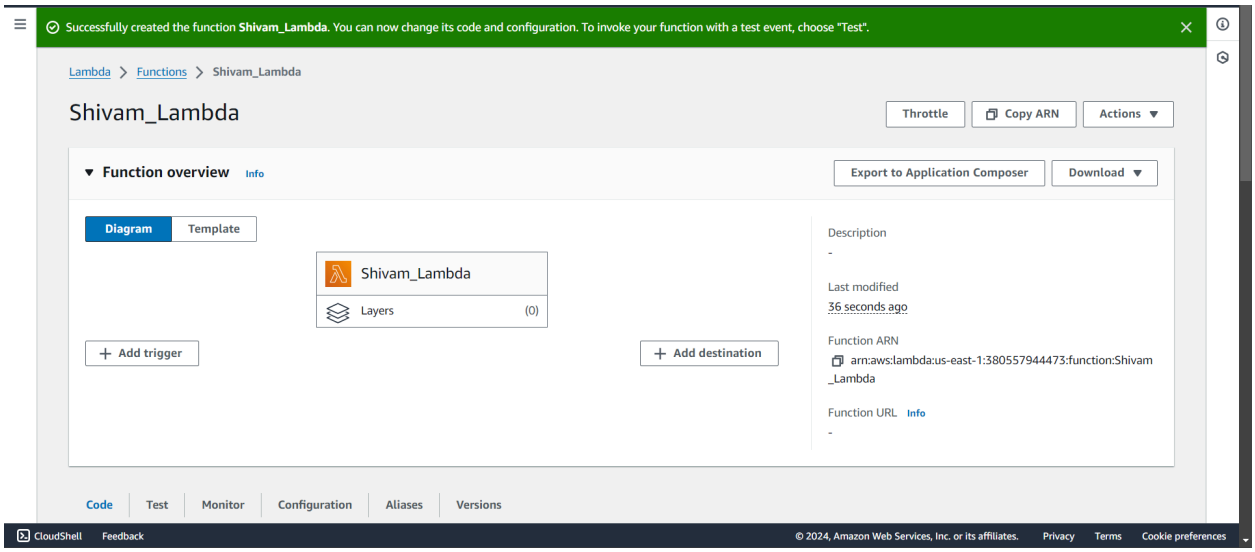


Architecture will be x86_64

Name: Aditya Dubey **Div: D15C** **Roll No: 10**



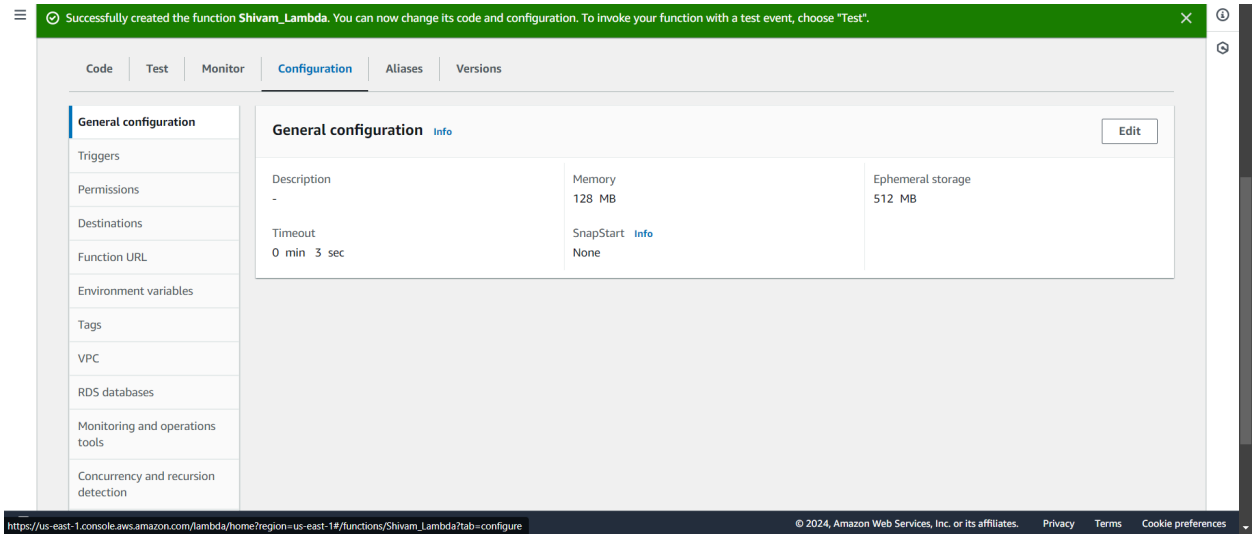
Select proper Execution role



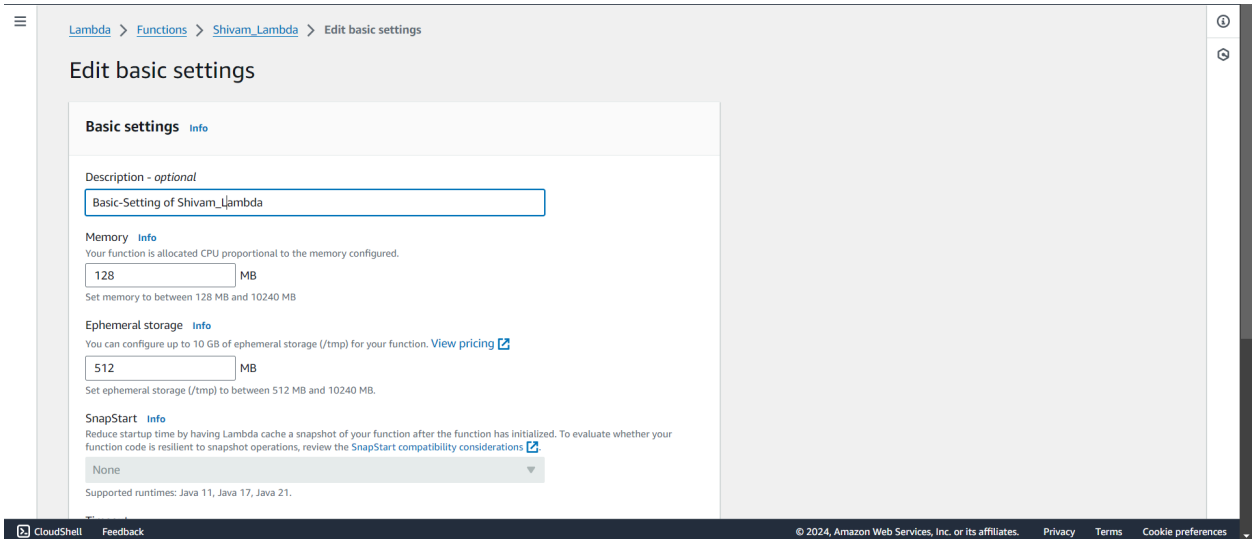
Successfully created Lambda function

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

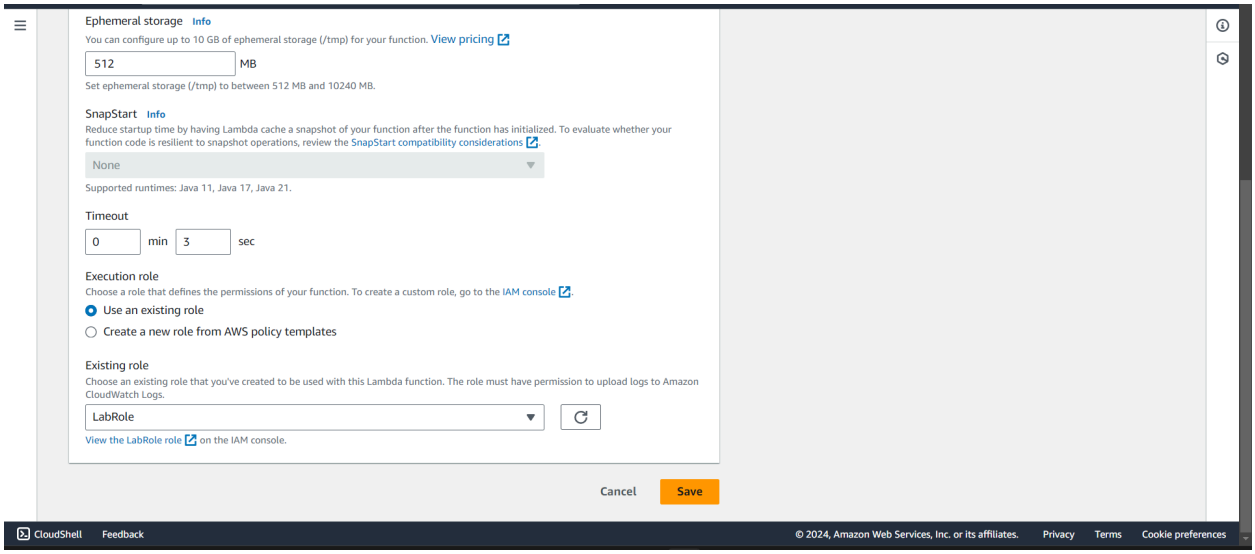
Name: Aditya Dubey **Div: D15C** **Roll No: 10**



You can add a description and adjust the memory and timeout settings. I've changed the timeout to 1 second, as that's enough for now.



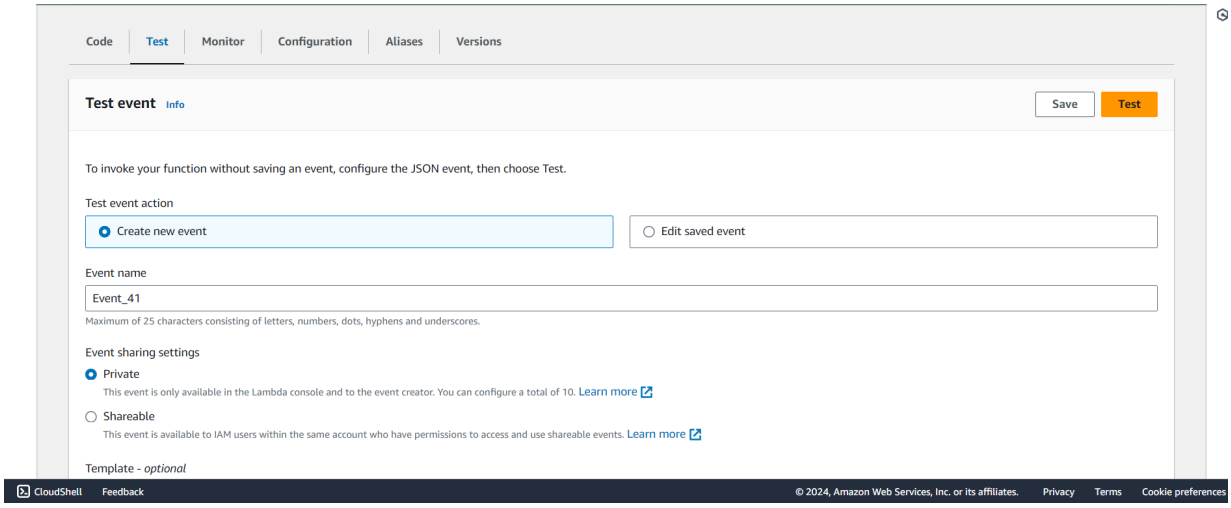
Name: Aditya Dubey **Div: D15C** **Roll No: 10**



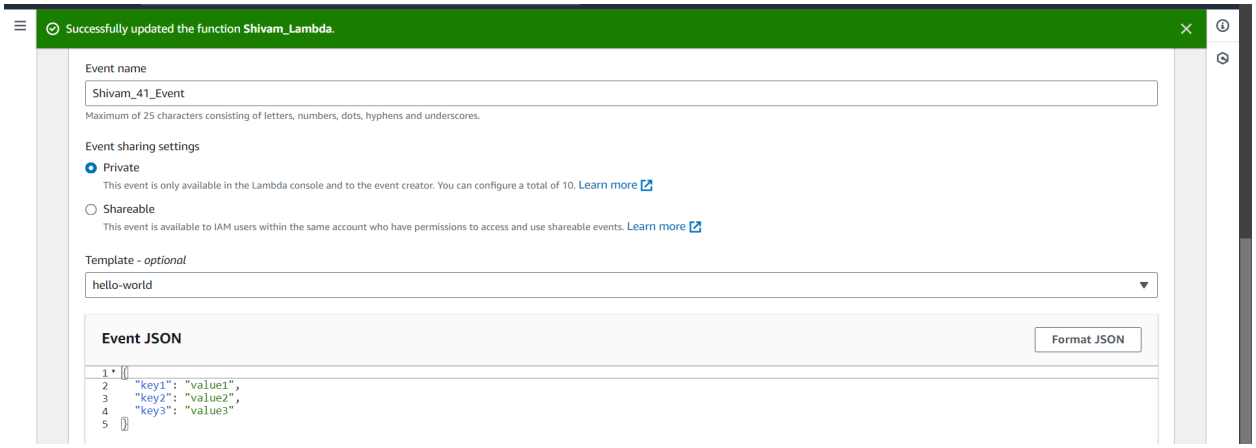
Click on Save

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

We basically create a new event to test and verify your Lambda function; setting Event Sharing to private keeps it secure and choosing the "hello-world" template provides a simple structure for testing without complex inputs.

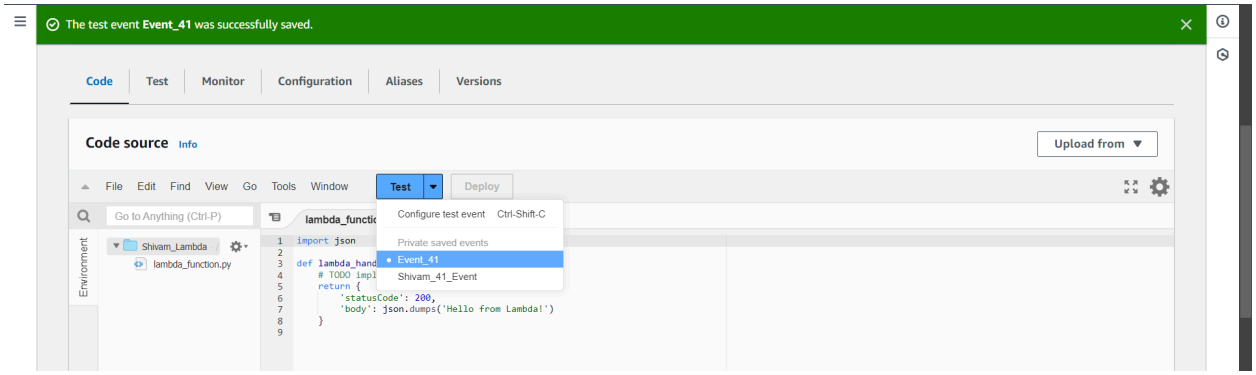


Name: Aditya Dubey **Div: D15C** **Roll No: 10**

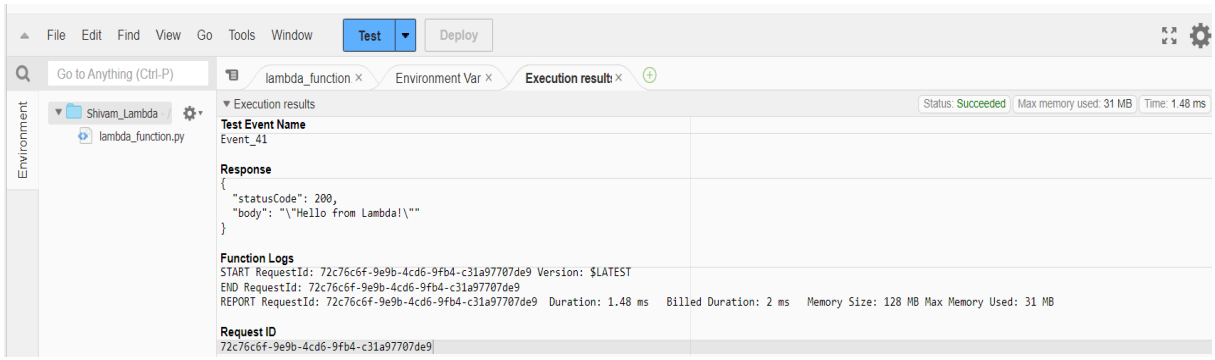


Click on save button above

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



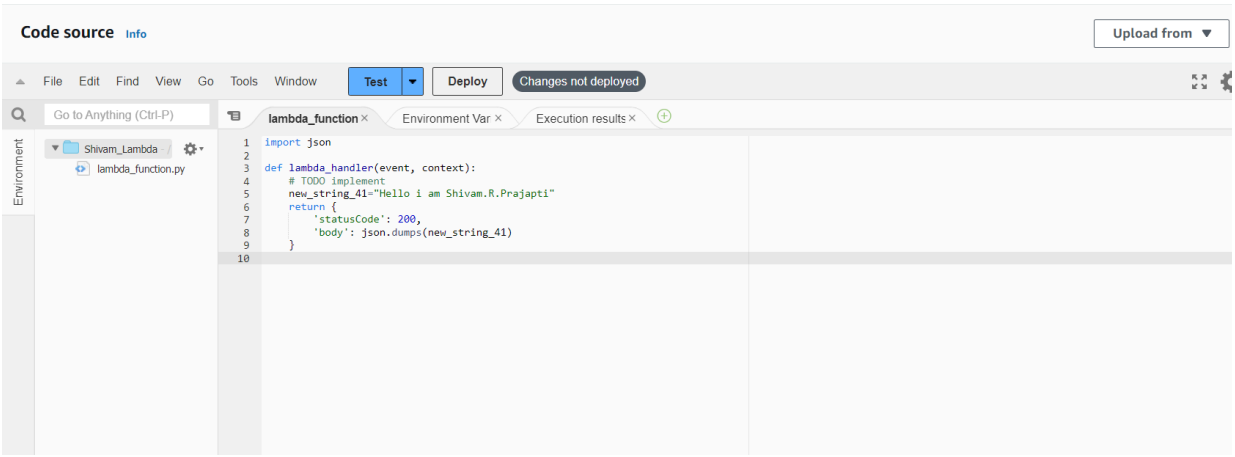
Output :

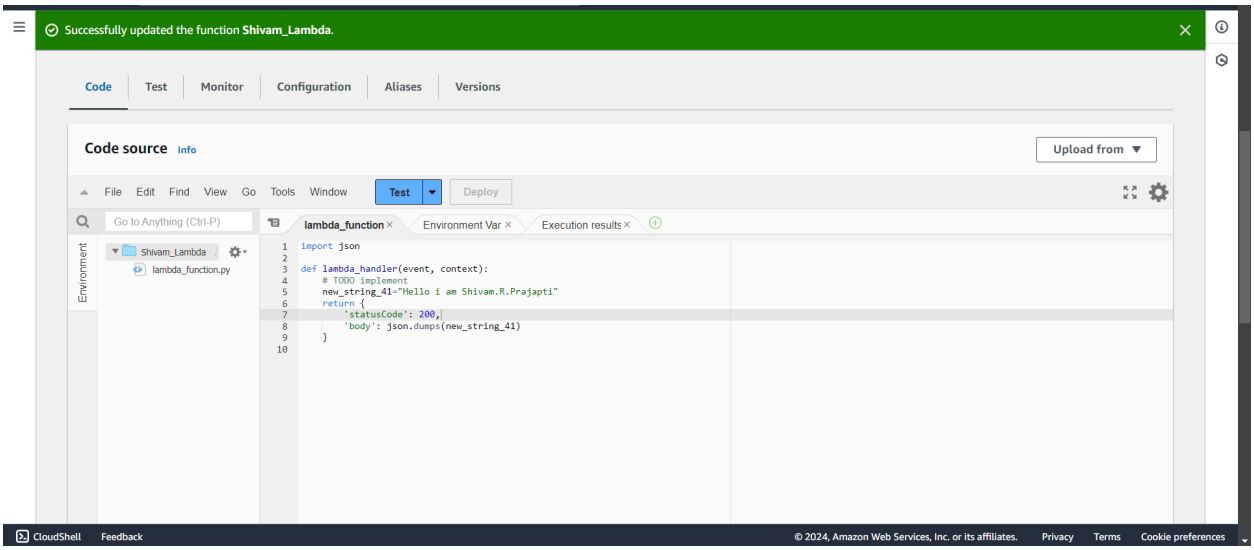


You select the created event to run the specific test you set up, and clicking 'Test' executes your Lambda function to check if it works as expected and produces the desired output.

Name: Aditya Dubey **Div: D15C** **Roll No: 10**

Step 6: You can edit your lambda function code. I have changed the code to display the new String. After Changing save it by Control + S and click on Deploy . Make sure you have internet connectivity while deploying or else it will show failed deployment

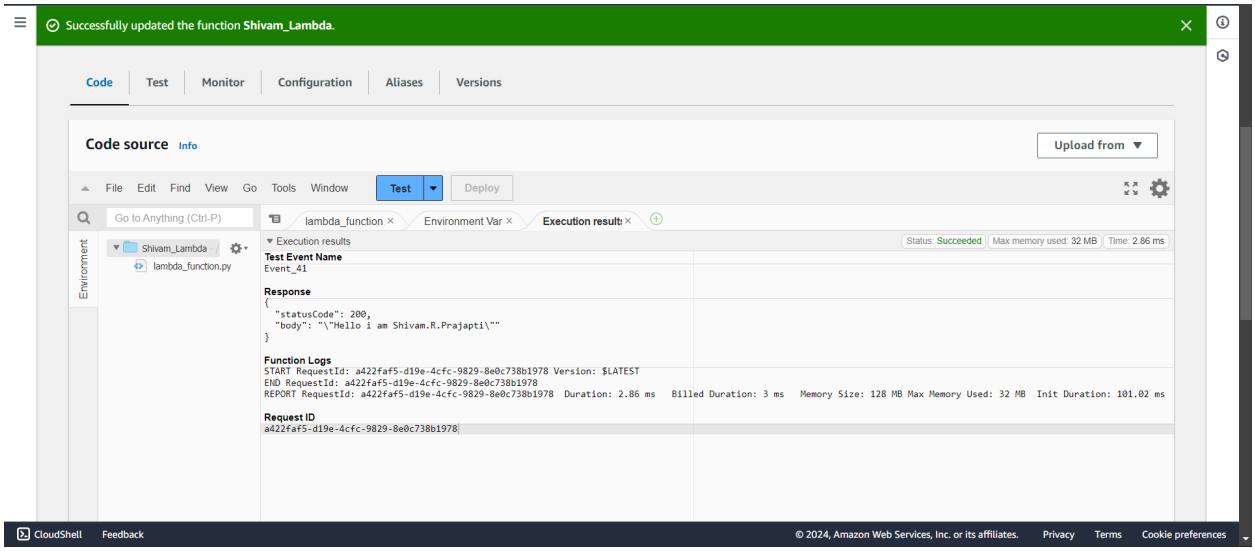




Successfully changed the function.

Name: Aditya Dubey **Div: D15C** **Roll No: 10**

Step 7: Click on '**Test**' to see the output. You'll get a status code of **200** which means "OK" and indicates that the request was successful, your string output, and the function logs, showing that it was deployed successfully.



CONCLUSION:

In this experiment, we successfully created an AWS Lambda function and followed the important steps involved. First, we set up the function using Python and adjusted the timeout setting to 1 second. Then, we created a test event to see how the function works and checked the output to ensure it was correct. We also modified the function's code and redeployed it to see the changes in real-time. So Lambda Function allows you to concentrate on writing code while AWS manages the infrastructure and automatically scales the service as needed. This makes it easier to develop and run applications without worrying about server management.

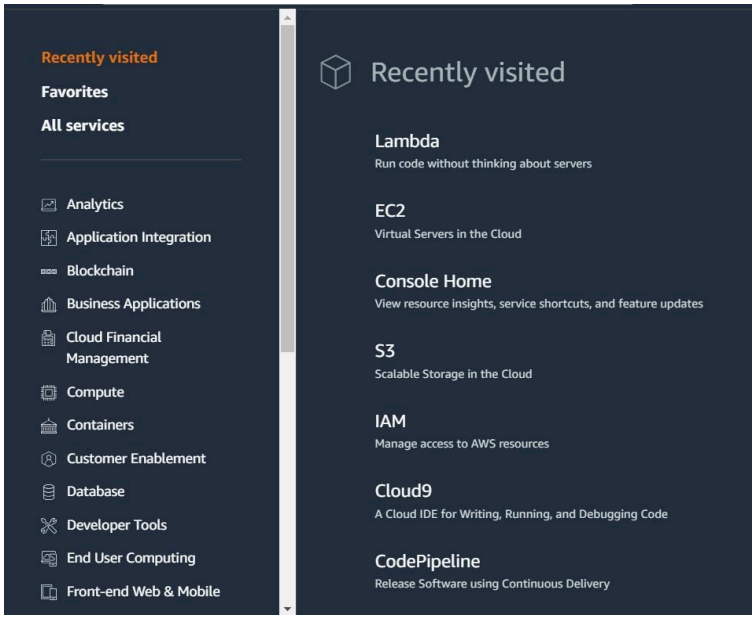
Name:Aditya Dubey **Div:D15C** **Roll NO :10**

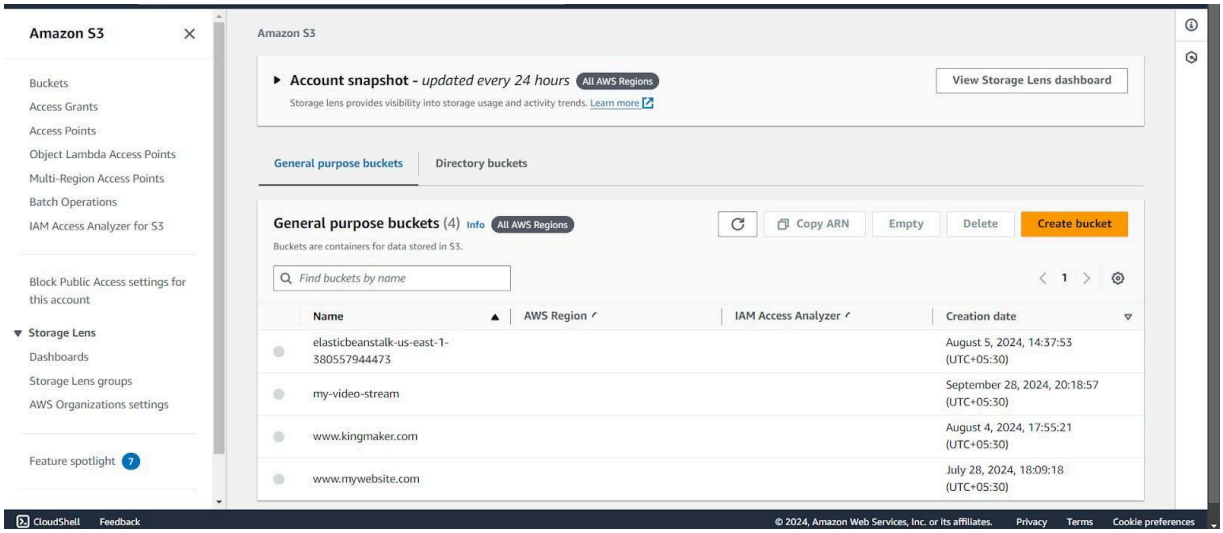
Experiment No :12

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

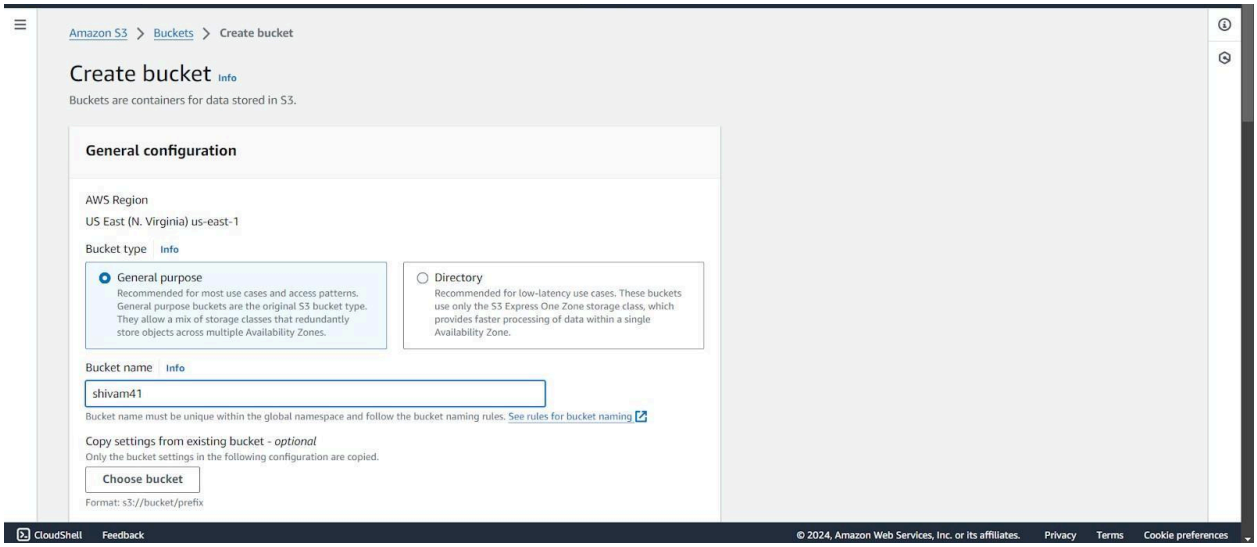
CREATING LAMBDA FUNCTION :

Step 1: Log in to your AWS Personal account. Then go to S3 in the services menu and click on "Create S3 Bucket."

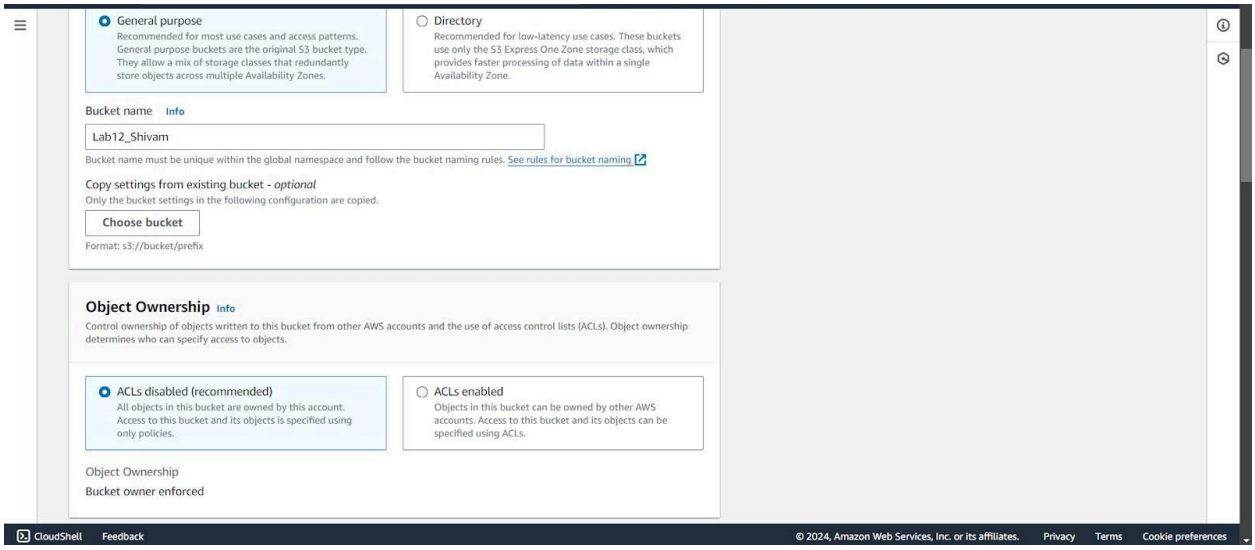




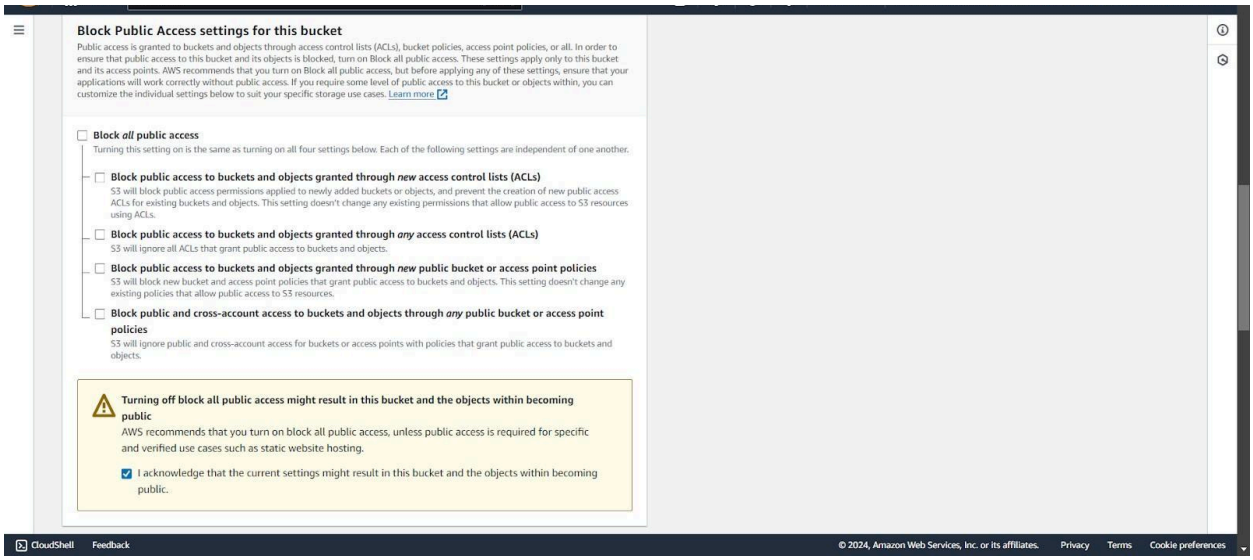
Step 2: Give your bucket a name, select "General purpose project," then uncheck "Block public access." Keep the other settings as they are.



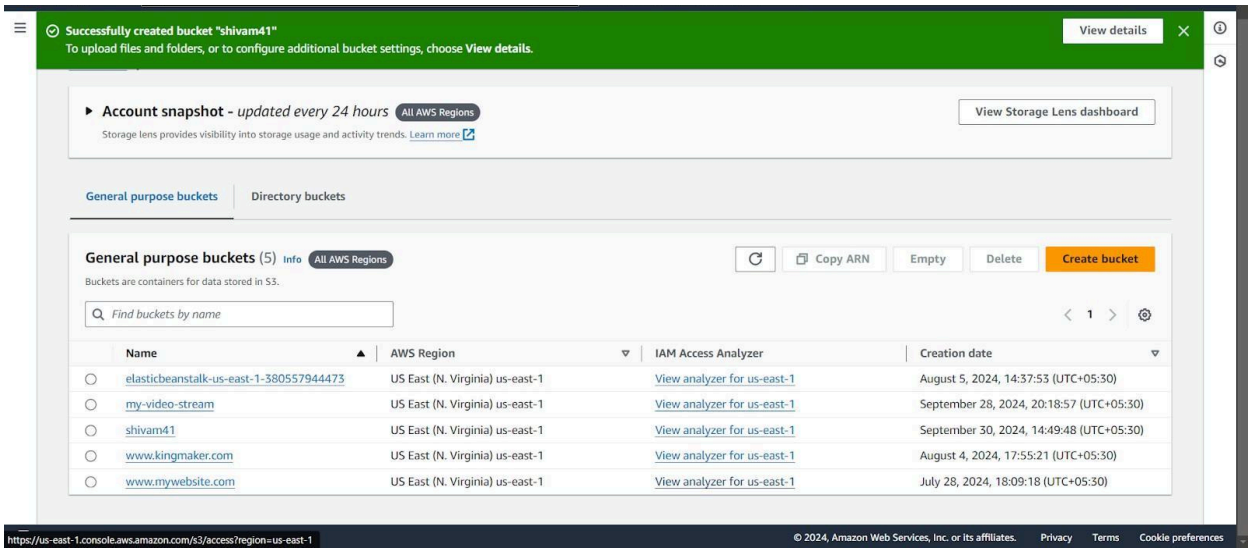
Given proper bucket Name



Selected Appropriate object Ownership

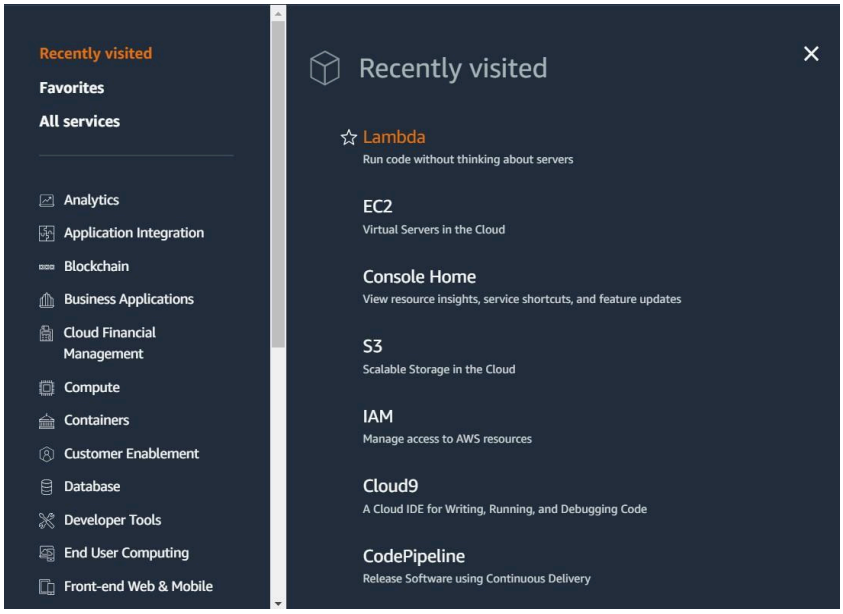


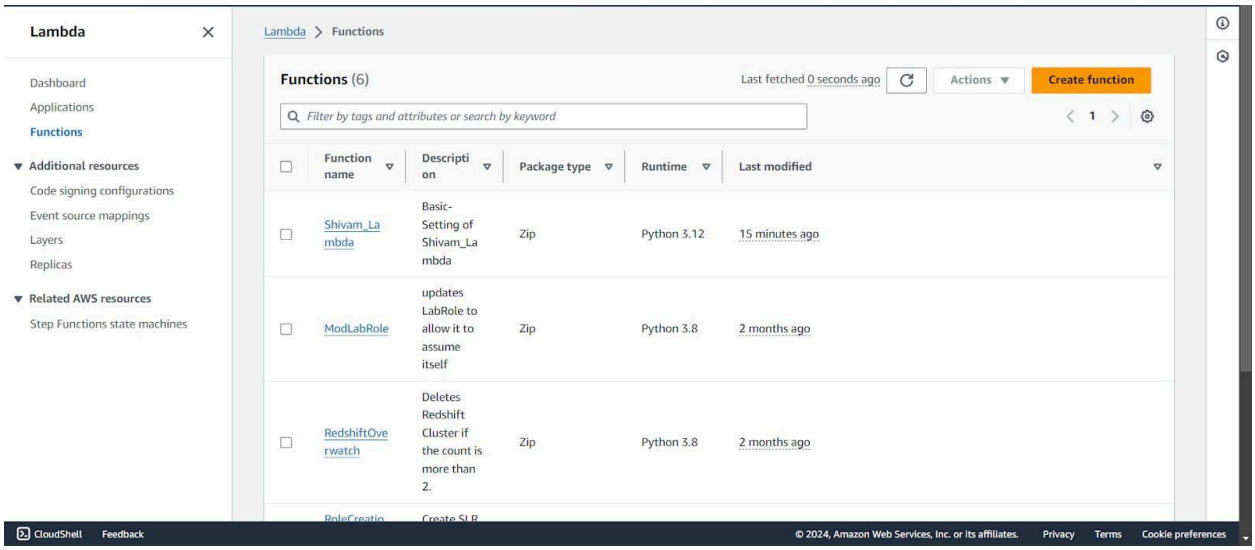
Unchecked the block all public access checkbox and checked the lower checkbox.



Successfully created the bucket

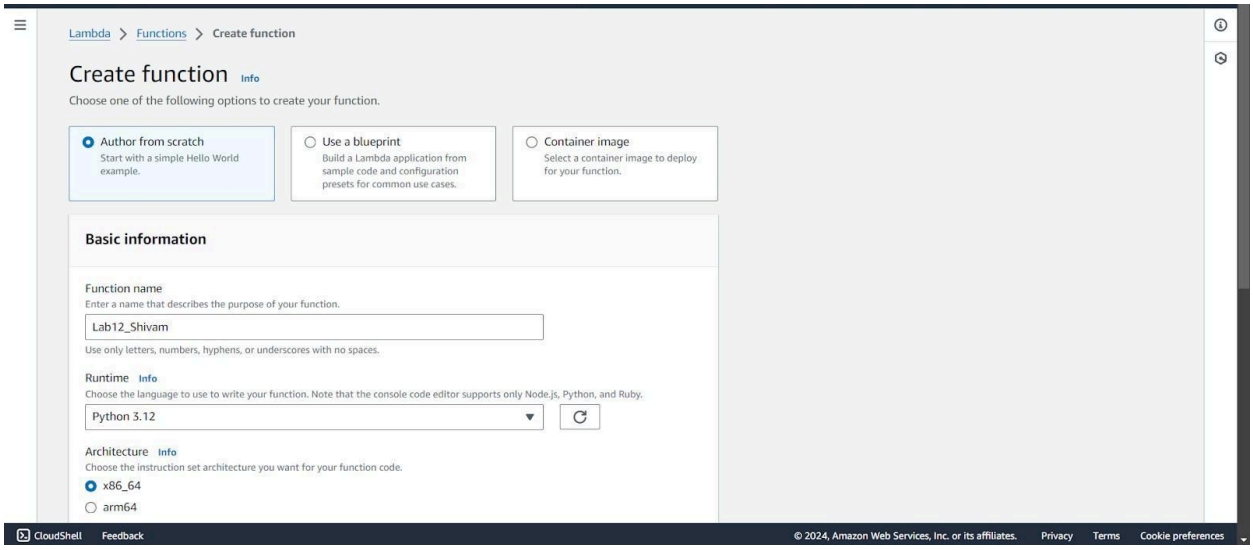
Step 3: Open lambda console and click on create function button.



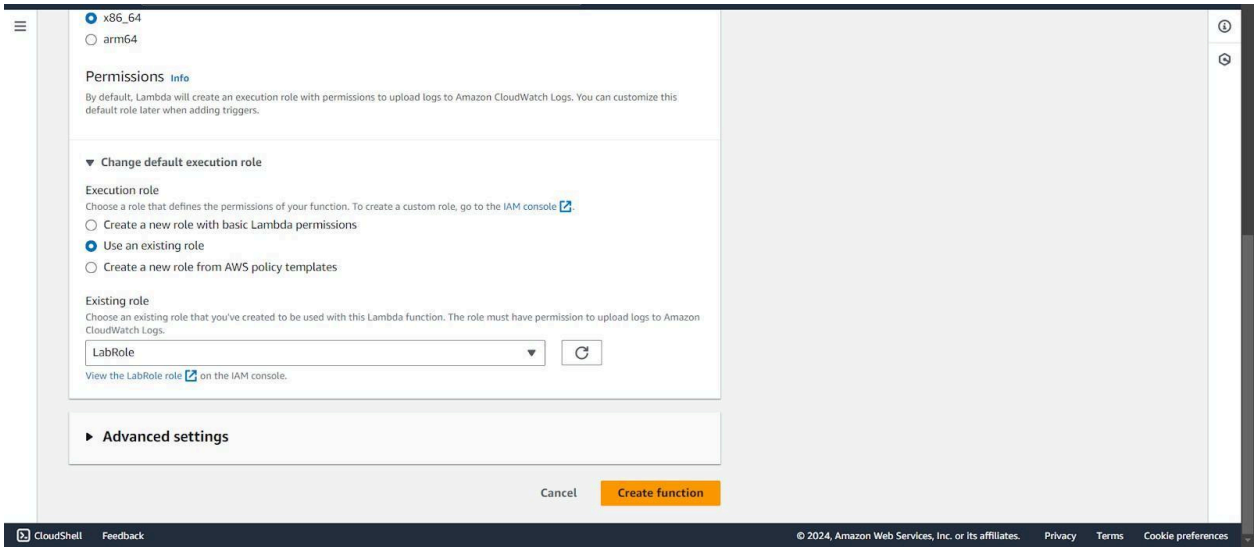


Step 4: Give your Lambda function a name and choose a programming language. The code editor only supports Node.js, Python, and Ruby, so in my case I have chosen **Python 3.12**. Set the **architecture to x86**. For the execution role, select '**Use an existing role**,' then pick '**Lab role**' from the dropdown menu under existing roles .

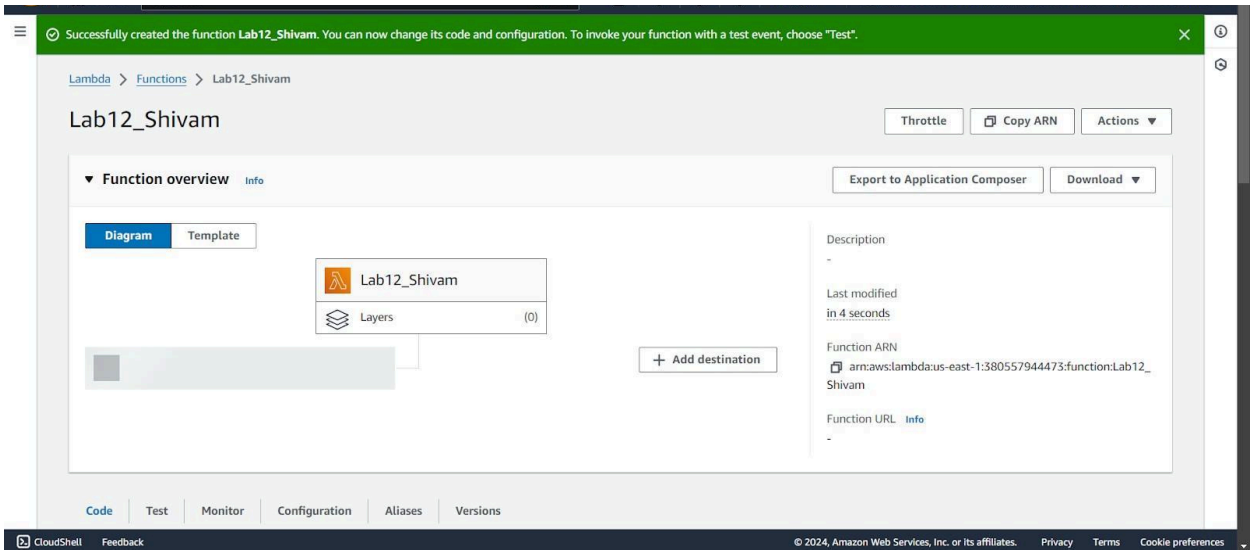
(This is because the Lab role already has the permissions needed for Lambda to run properly, so you don't need to create a new role from scratch. It's a quicker and more convenient option)



Given proper function name and selected language for Lambda function

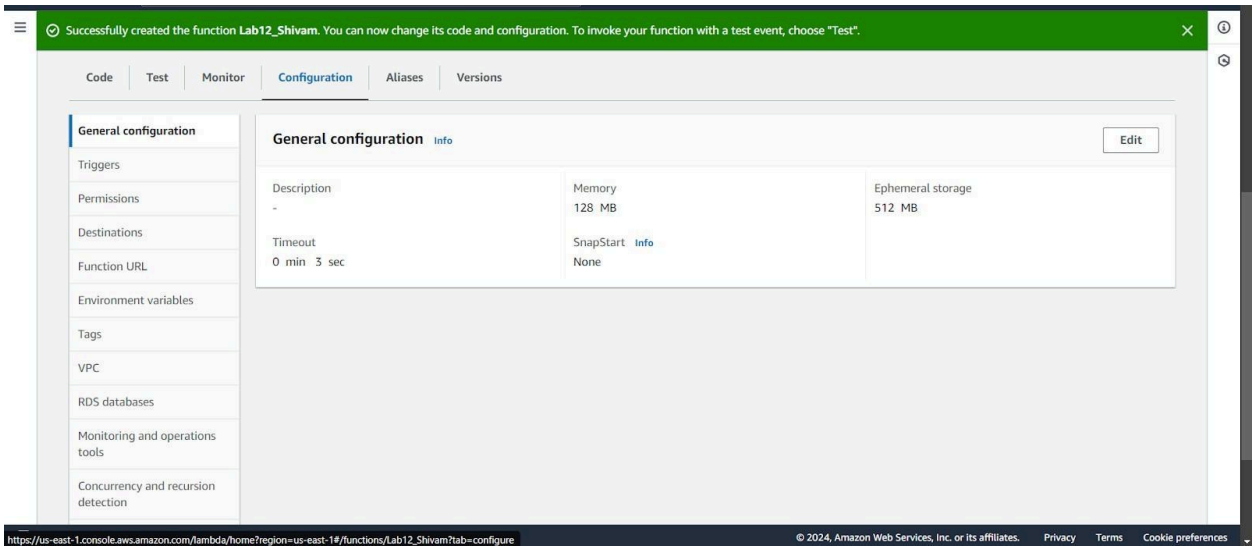


Selected Appropriate Execution role

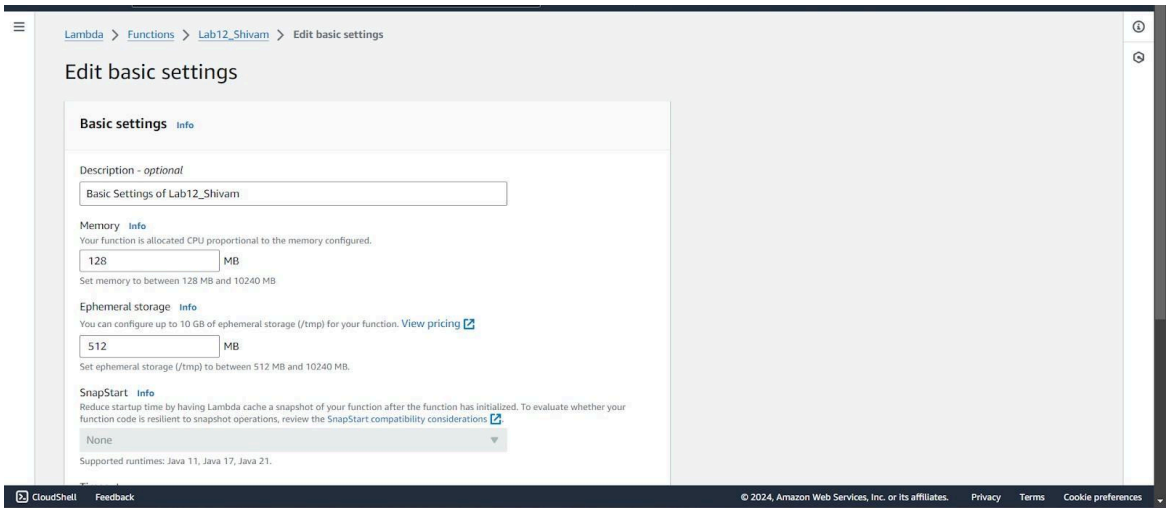


Successfully created the Lambda function.

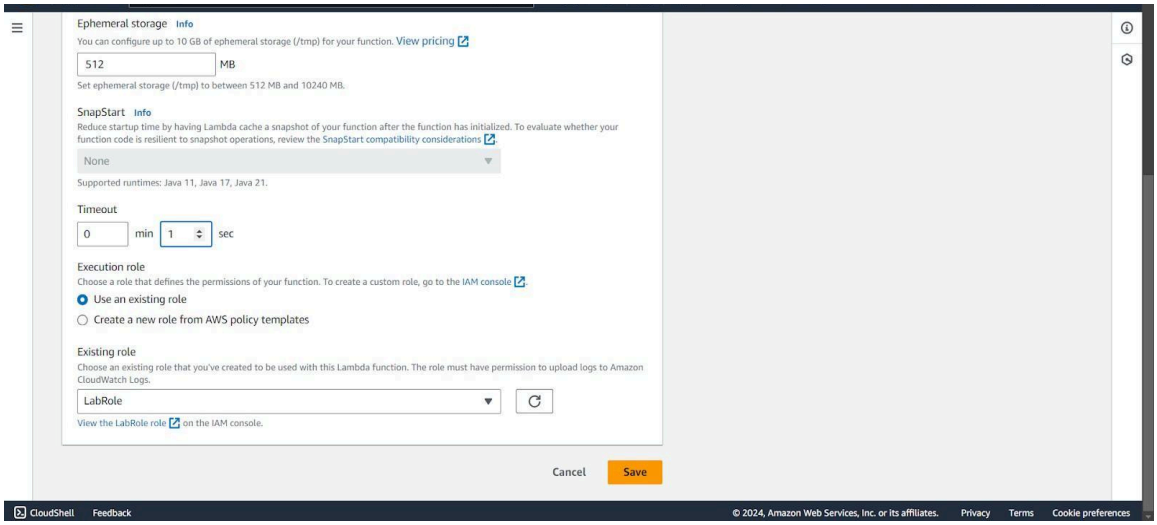
Step 5: To view or change the basic settings, go to the 'Configuration' tab and click 'Edit' under 'General settings.' (THIS STEP IS OPTIONAL)



You can add a description and adjust the memory and timeout settings. I've changed the timeout to 1 second, as that's enough for now.

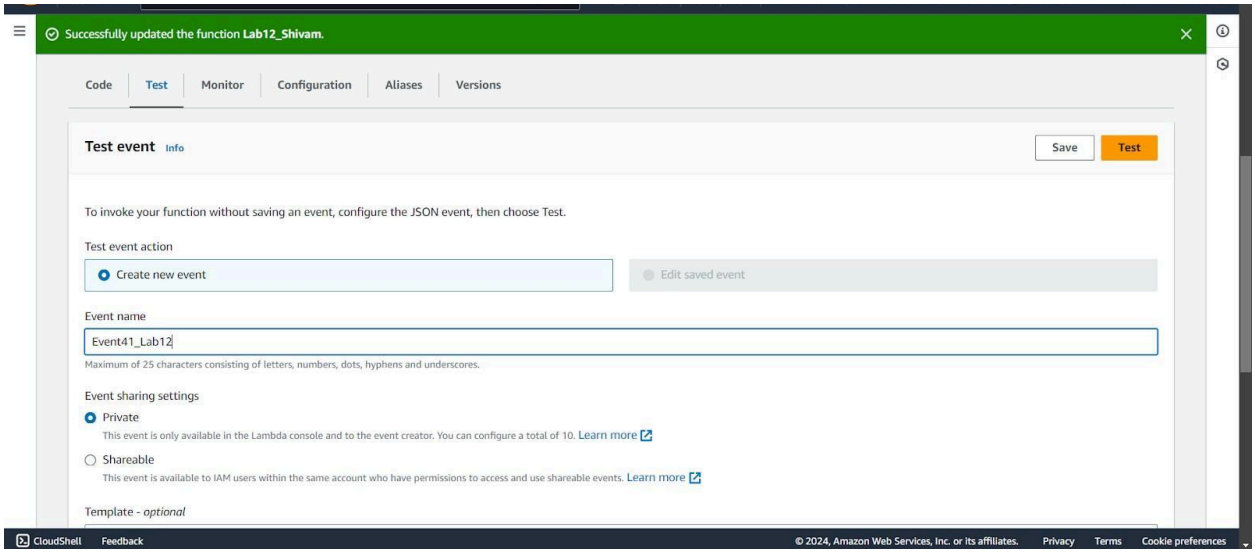


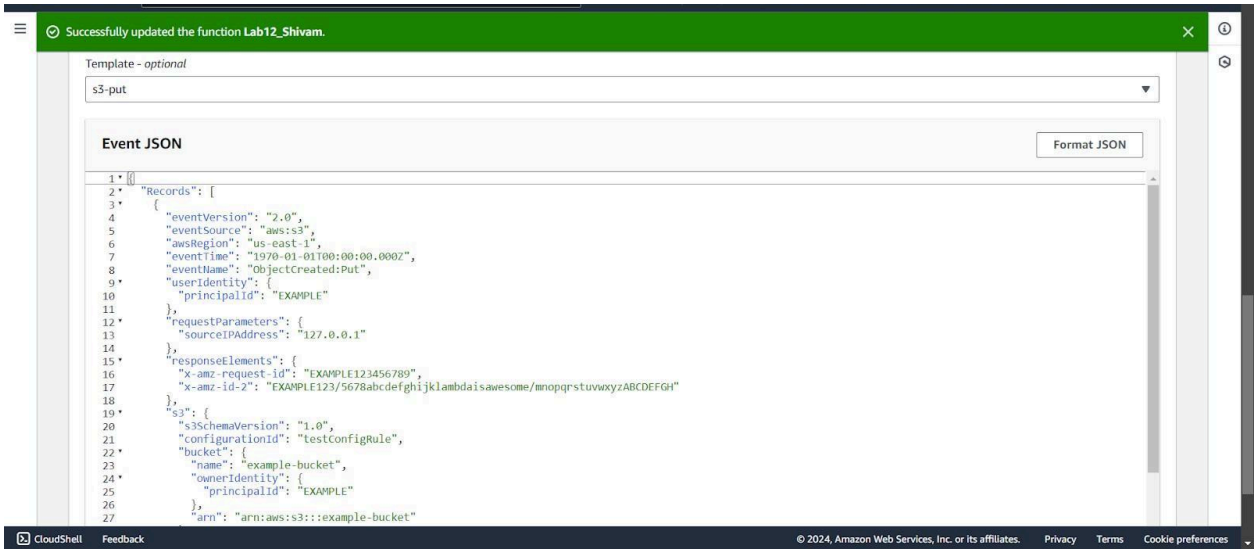
Given some description for your settings



Click on Save.

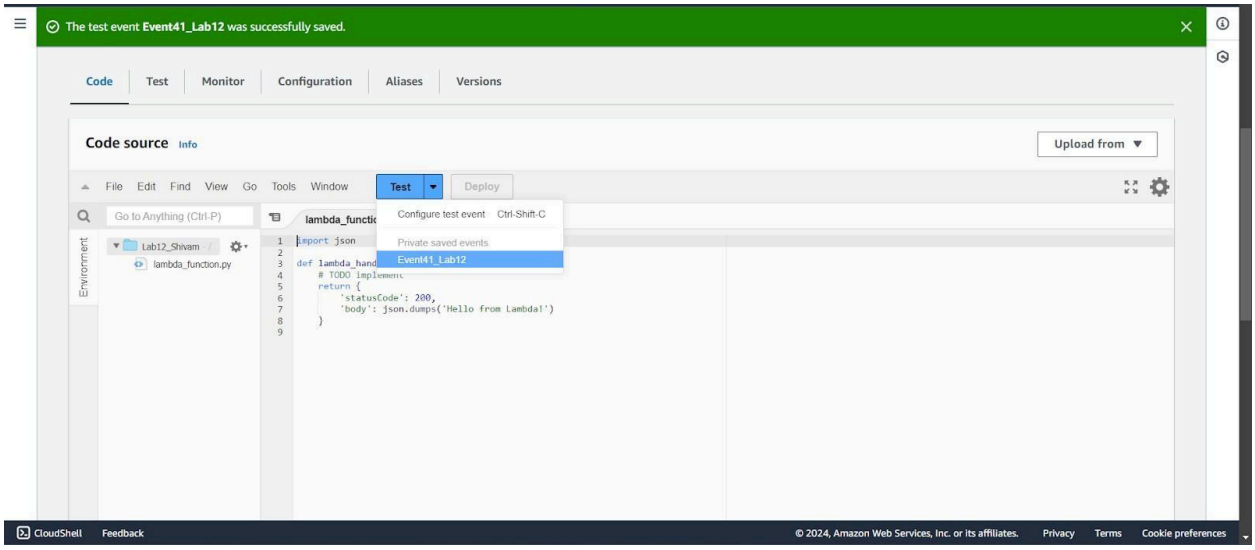
Step 6: Click on the "Test" tab, then select "Create a new event." Give the event a name, set "Event Sharing" to private, and choose the "S3 Put" template.S3 (Simple Storage Service) template allows you to test your Lambda function specifically for events related to uploading files to an S3 bucket.



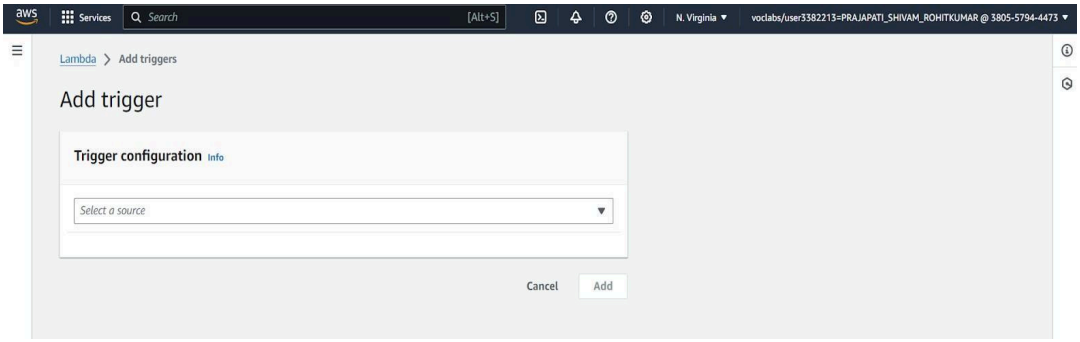


Event Jason code will be automatically generated once S3 -put is selected.

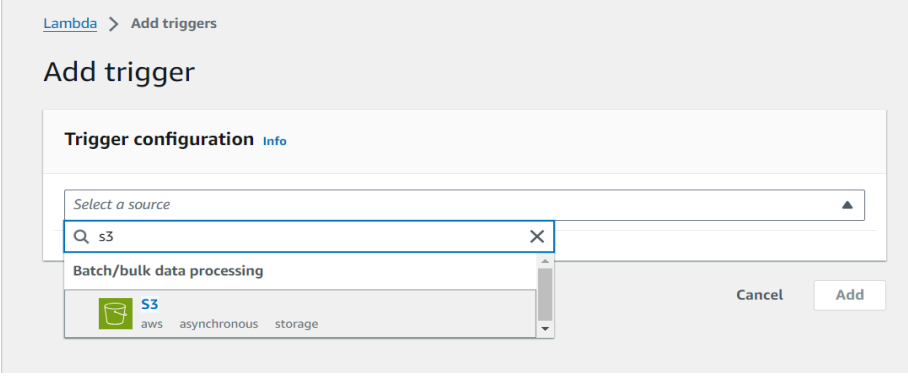
Step 7: Now In the Code section select the created event from the dropdown .

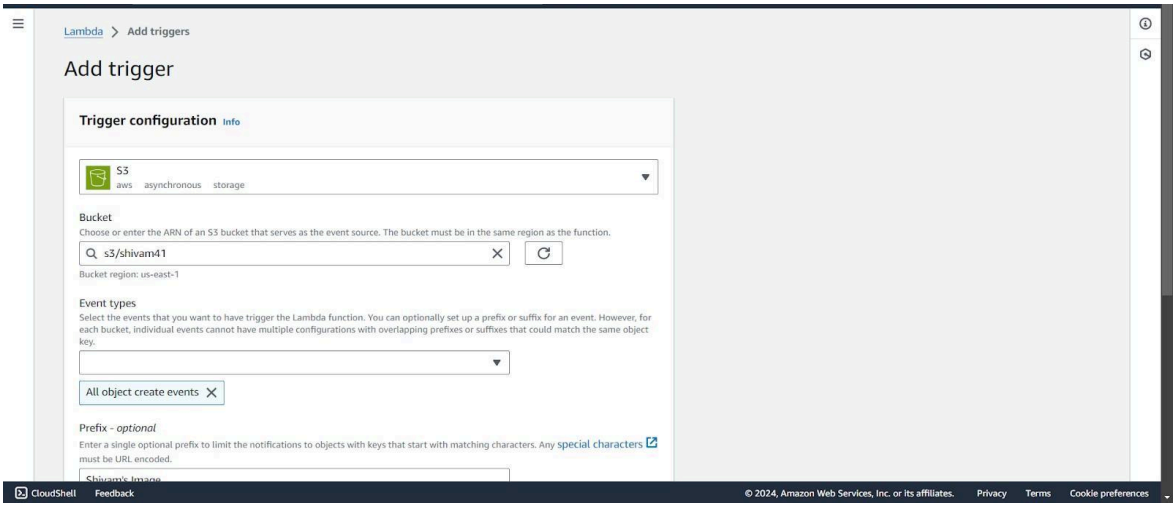


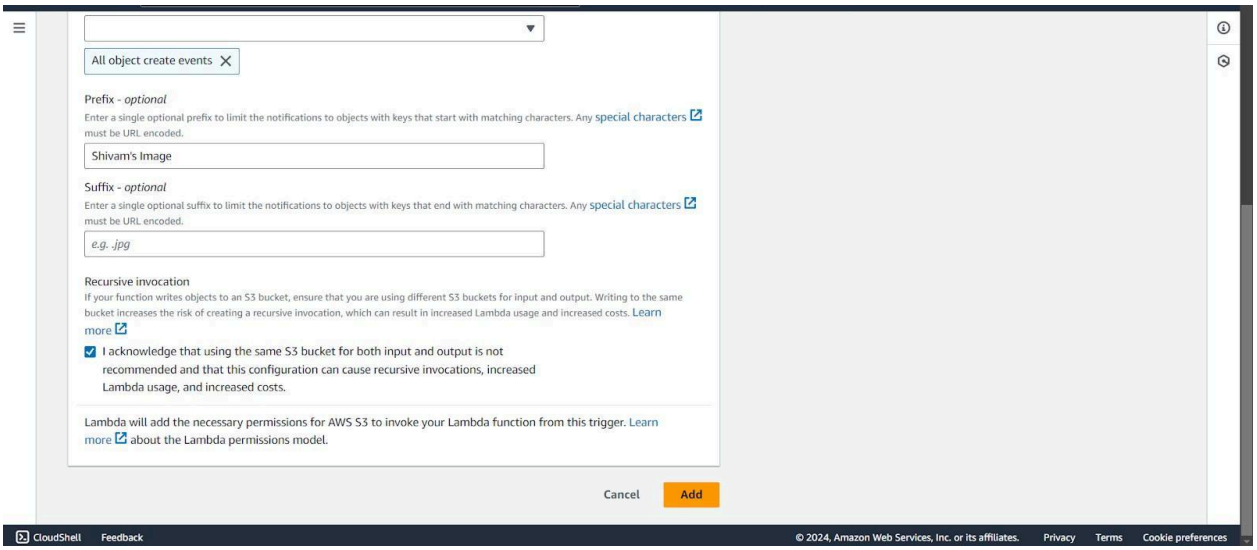
Step 8: In the Lambda function, click on "**Add Trigger.**" Adding a trigger allows your Lambda function to automatically run in response to specific events such as uploads to an S3 bucket



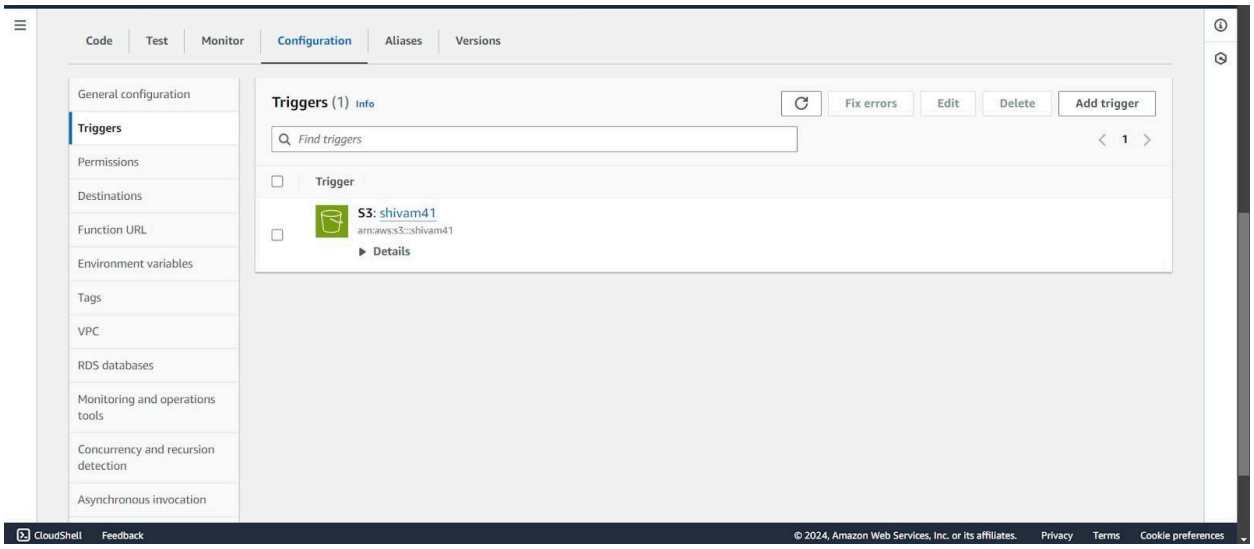
Now select the source as S3, then choose the bucket name from the dropdown menu. Keep the other settings as default, and you can also add a prefix for the image if you want.A prefix for an image (or any file) in S3 is a string that you can use to organize or filter files within a bucket. It acts like a folder name, helping to categorize your files.




Lambda > Add triggers
Add trigger
Trigger configuration Info
S3
aws asynchronous storage
Bucket
Choose or enter the ARN of an S3 bucket that serves as the event source. The bucket must be in the same region as the function.
s3/shivam41
Bucket region: us-east-1
Event types
Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.
All object create events
Prefix - optional
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters. Any special characters must be URL encoded.
Shivam's Image
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

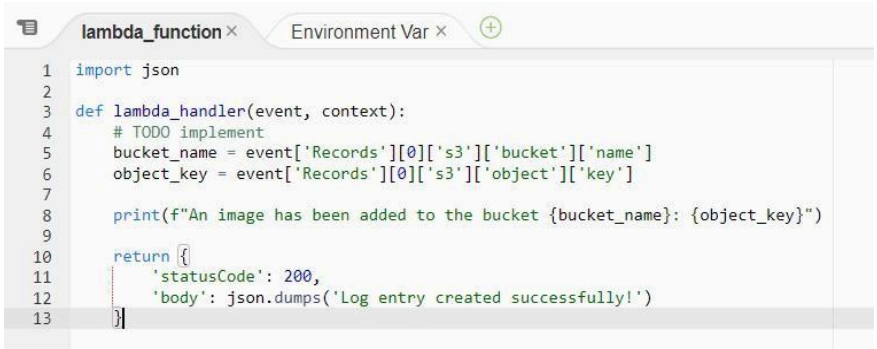


Click on save

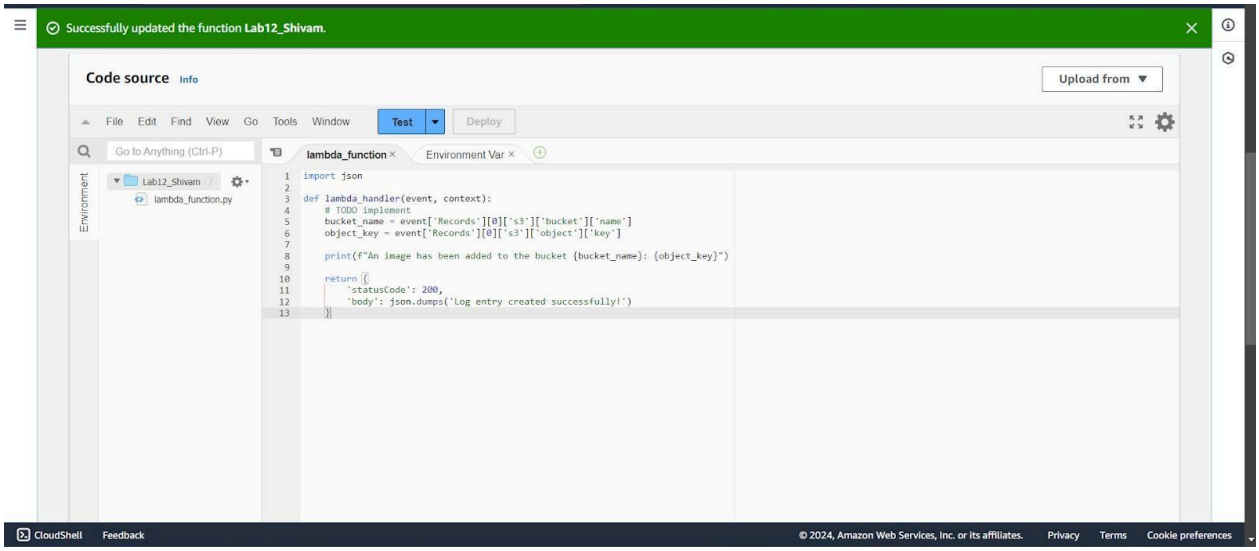


Step 9: Now Write code that logs a message like “An Image has been added” when triggered. Save the file and click on deploy.

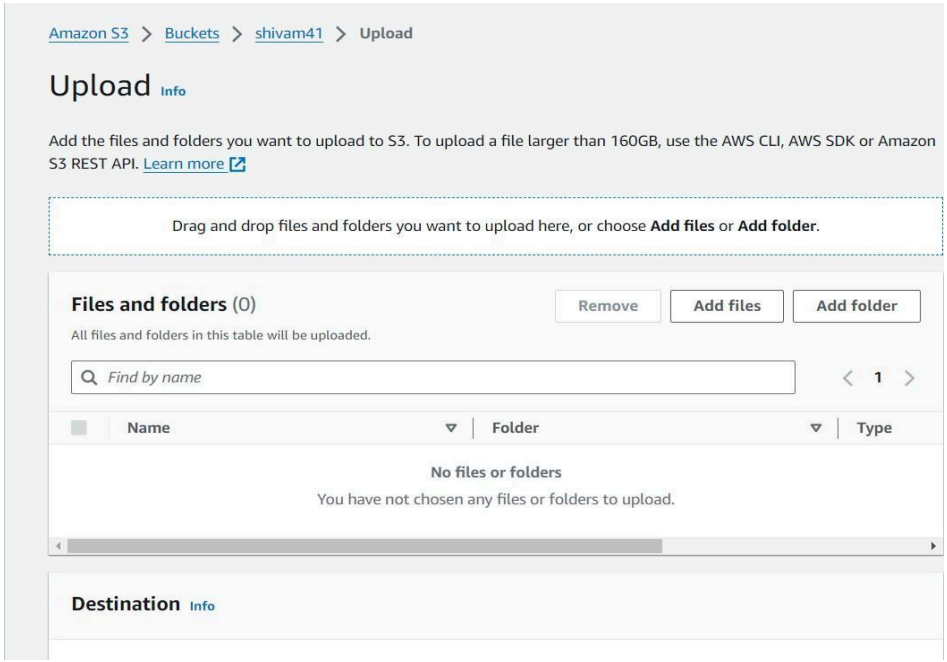
```
import json
def lambda_handler(event, context):
    # TODO implement
    bucket_name = event['Records'][0]['s3']['bucket']['name']
    object_key = event['Records'][0]['s3']['object']['key']
    print(f"An image has been added to the bucket {bucket_name}: {object_key}")
    return {
        'statusCode': 200,
        'body': json.dumps('Log entry created successfully!')
    }
```



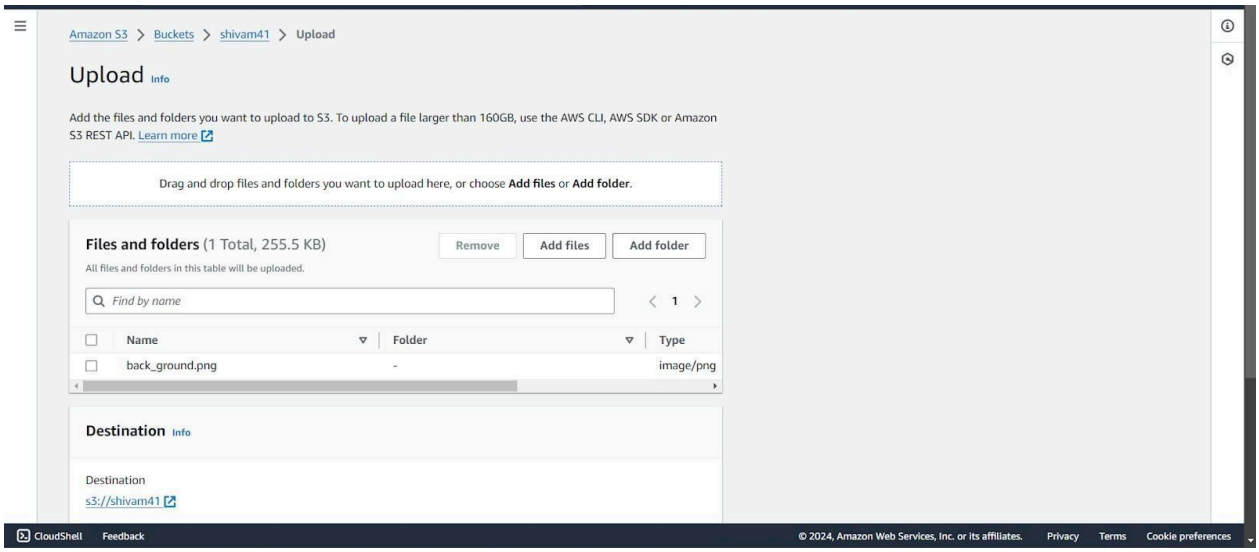
Save it by Control + S and deploy

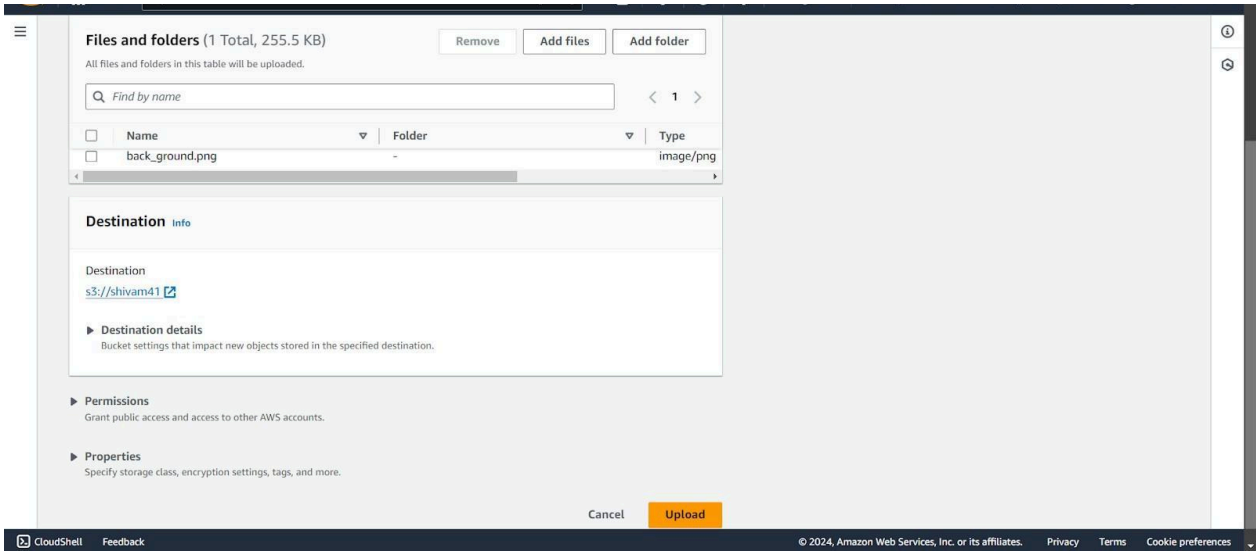


Step 10: Now we will upload any image to the bucket



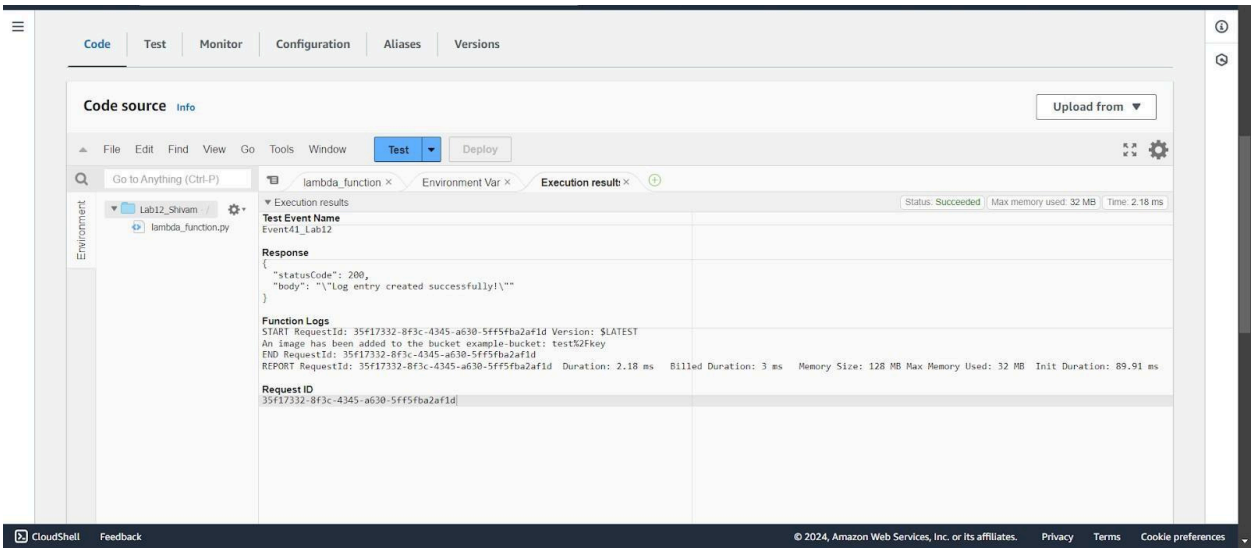
Click on add file where you can upload any image of your choice in your bucket

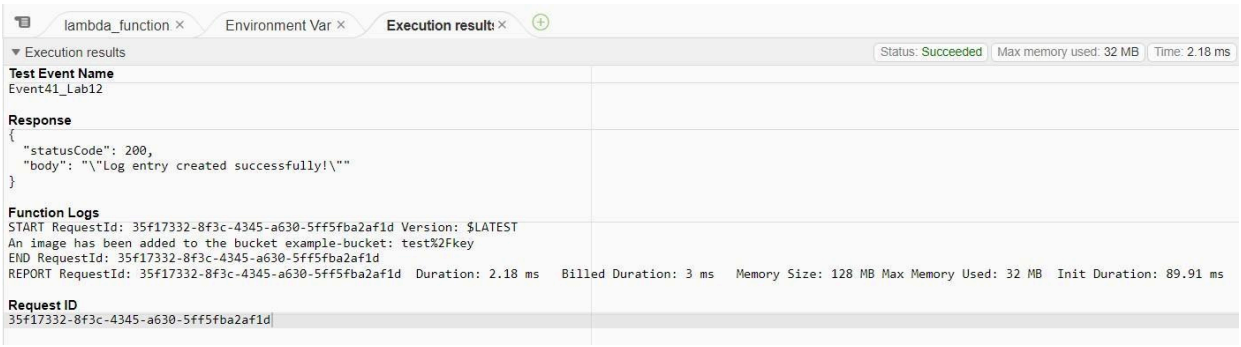




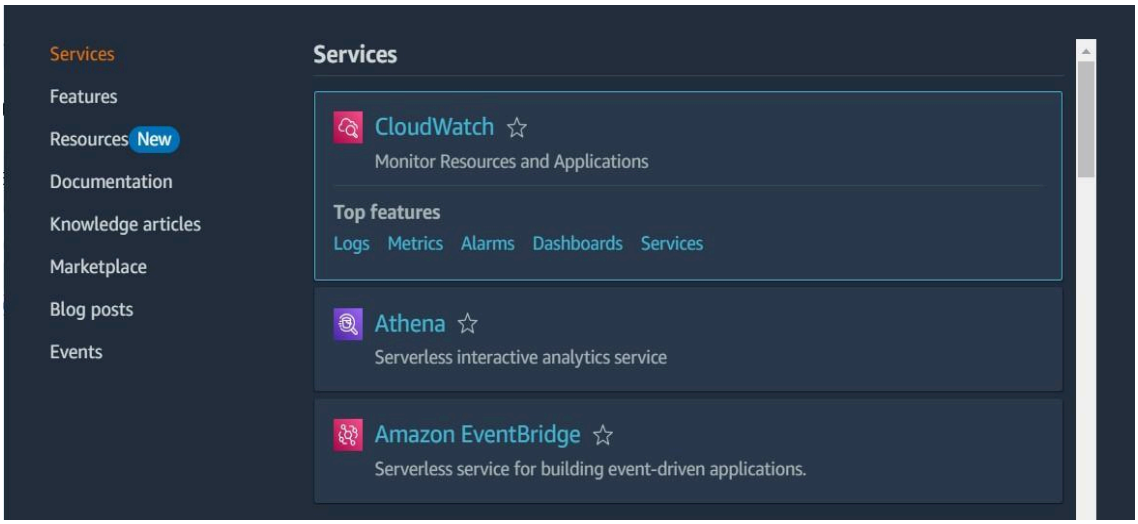
Click on Upload

Step 11: Now click on "Test" in Lambda to see if it logs the activity when an image is added to S3.



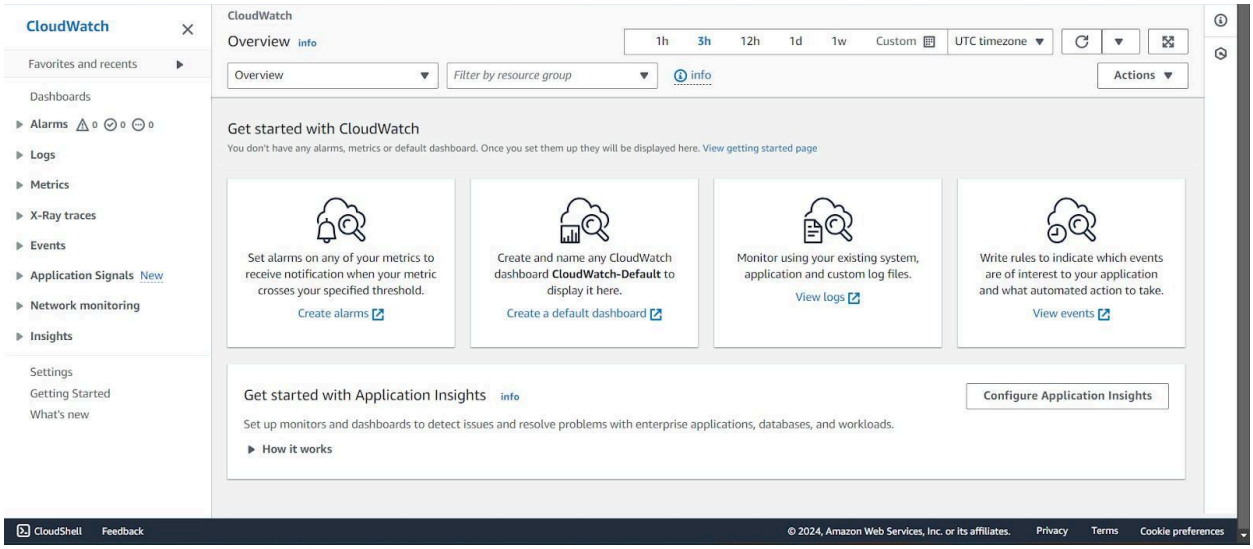


Step 12: Now let's check the logs on CloudWatch. Go to the "Monitor" section and click on "View CloudWatch Logs.

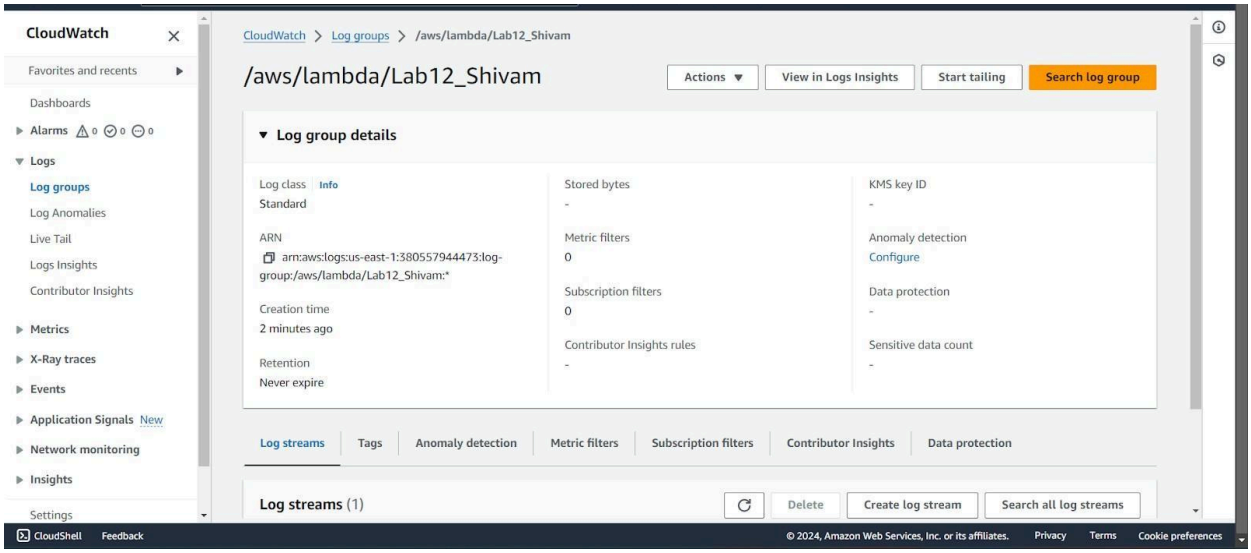


Name:Aditya Dubey **Div:D15C** **Roll NO :10**

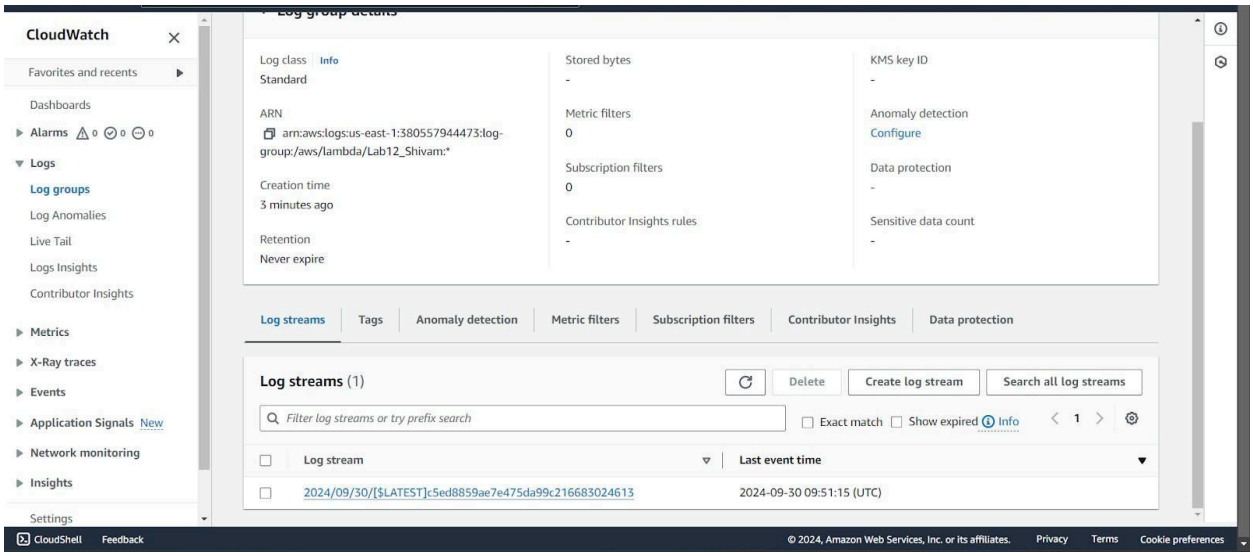
Click on the CloudWatch:



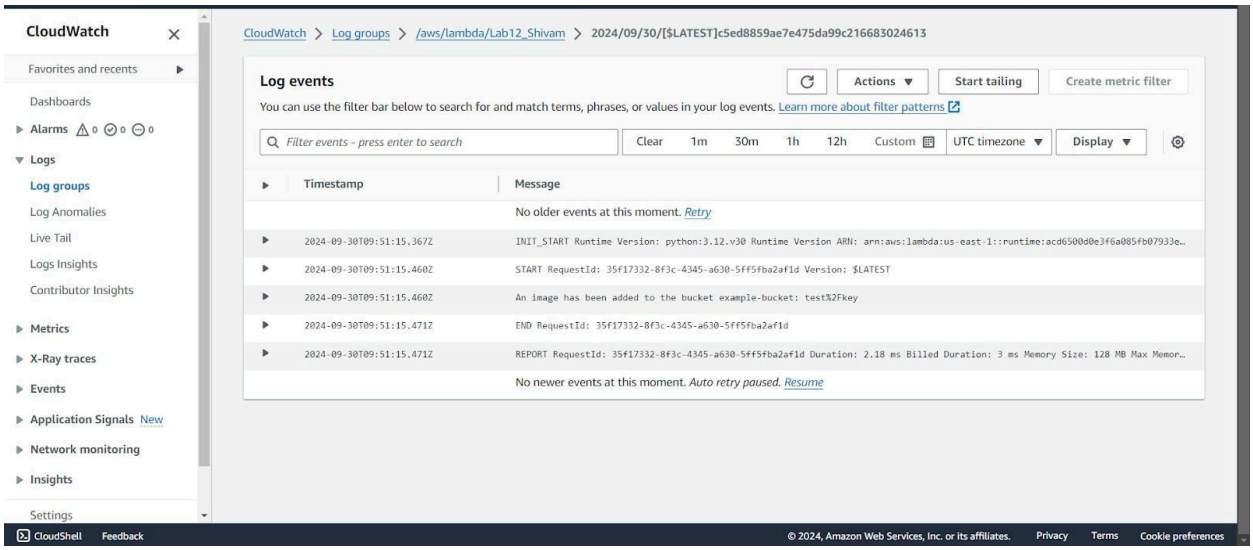
Click on the logs:



Click on the log group:



Scrolled down and click on the log stream:



CONCLUSION:

In this experiment, we successfully created an AWS Lambda function that logs a message when an image is uploaded to an S3 bucket. It's important to choose the S3 Put template for the event; otherwise, the code will give an error. The function was triggered correctly when files were uploaded to S3, showing that Lambda's event-driven design works well. This experiment showed how Lambda can respond to S3 events and how to fix common problems with the event setup.