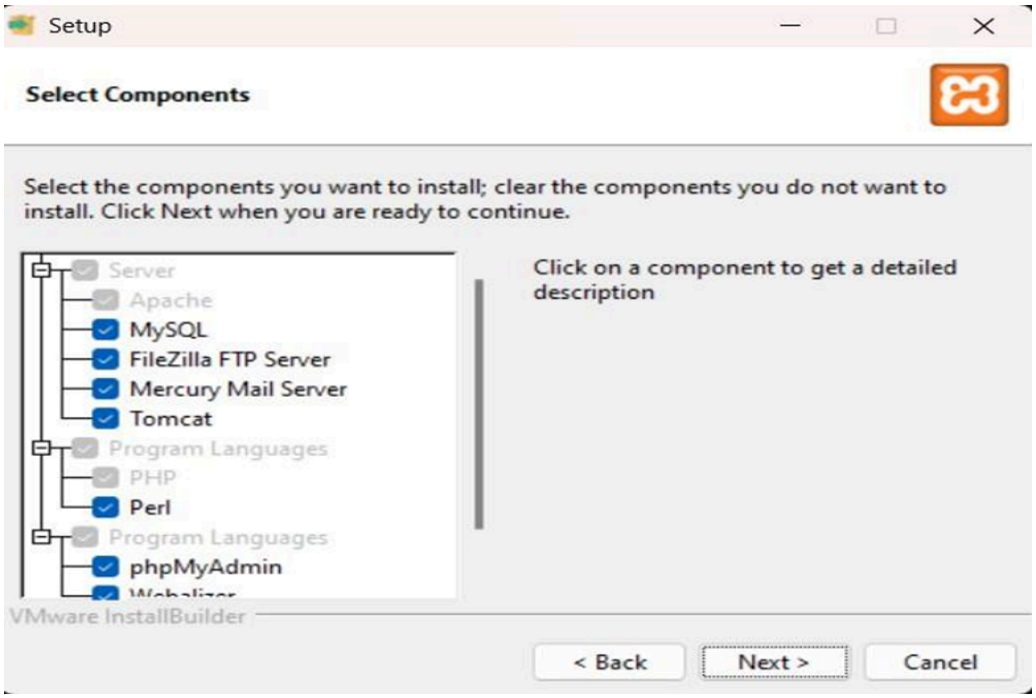
Name: Advik Hegde **Div/Roll No.: D15C/15**

Static Hosting:

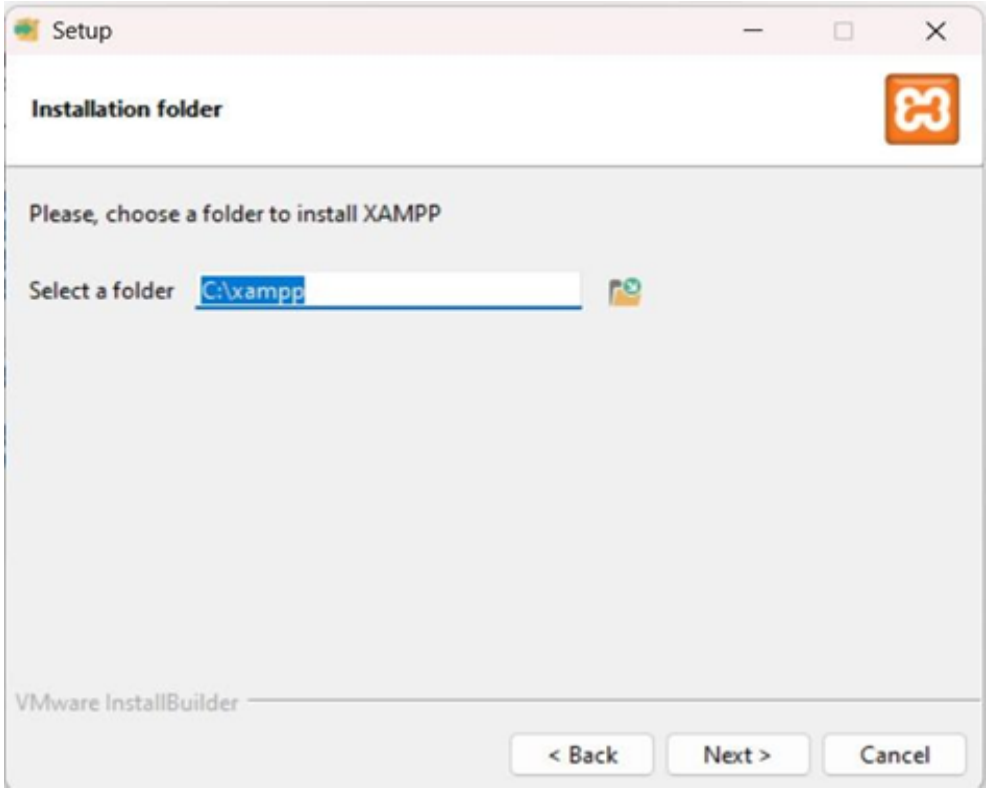
1) On local server (XAMPP)
Step 1: Install XAMPP from https://www.apachefriends.org/

1) Select your OS. It will automatically start downloading.

2) Open the setup file. Select all the required components and click next

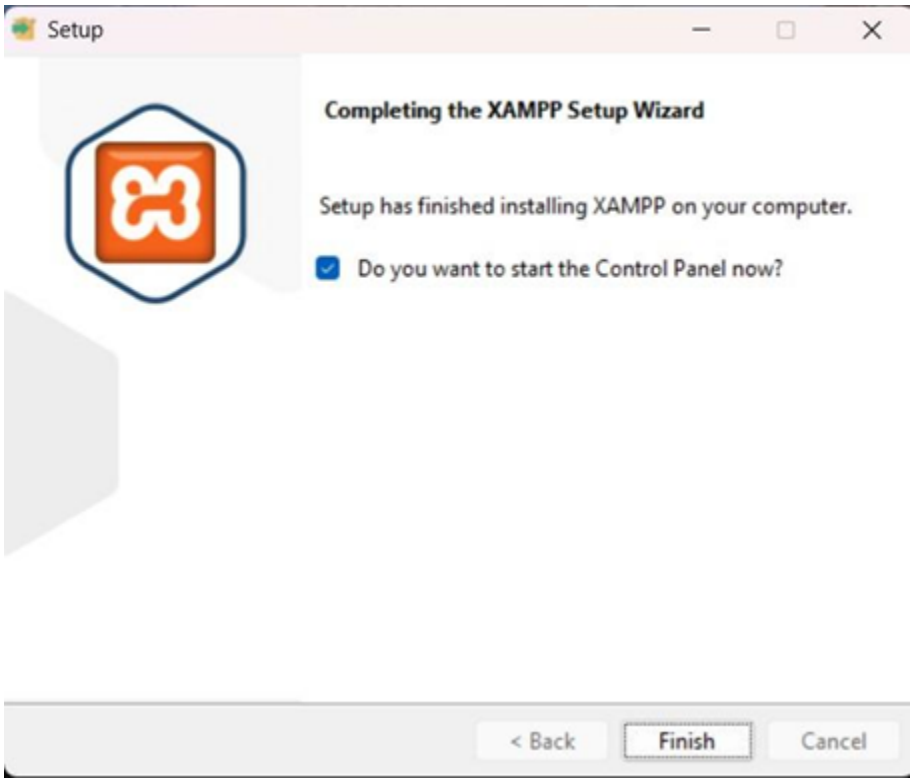


3) Choose the folder to install XAMPP in. Make sure the folder is empty. Click next

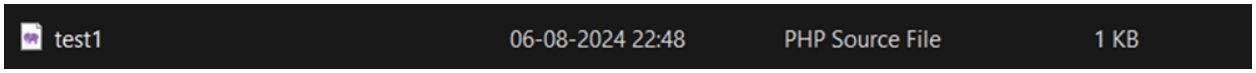


4) Select the language, click next. XAMPP starts to install

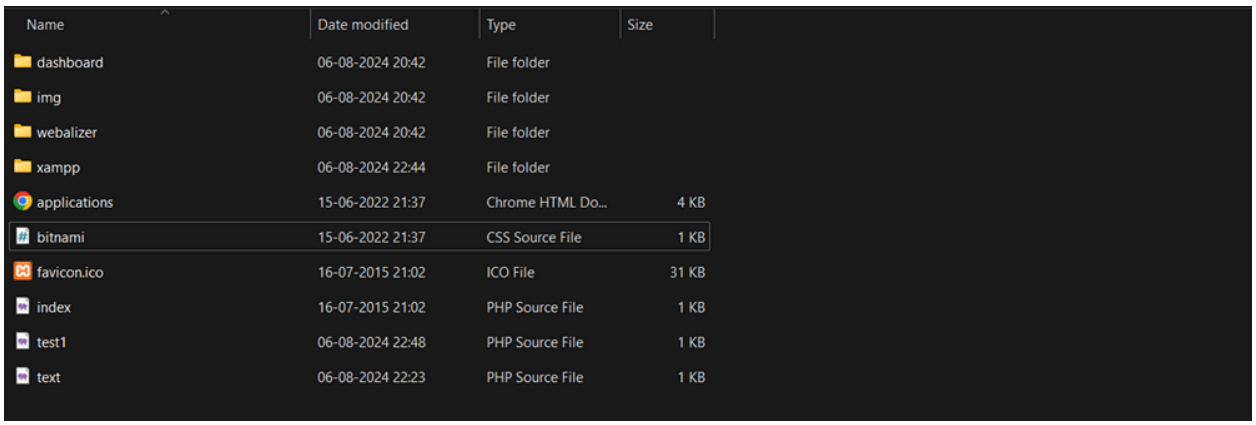
5) The installation is complete. Click Finish



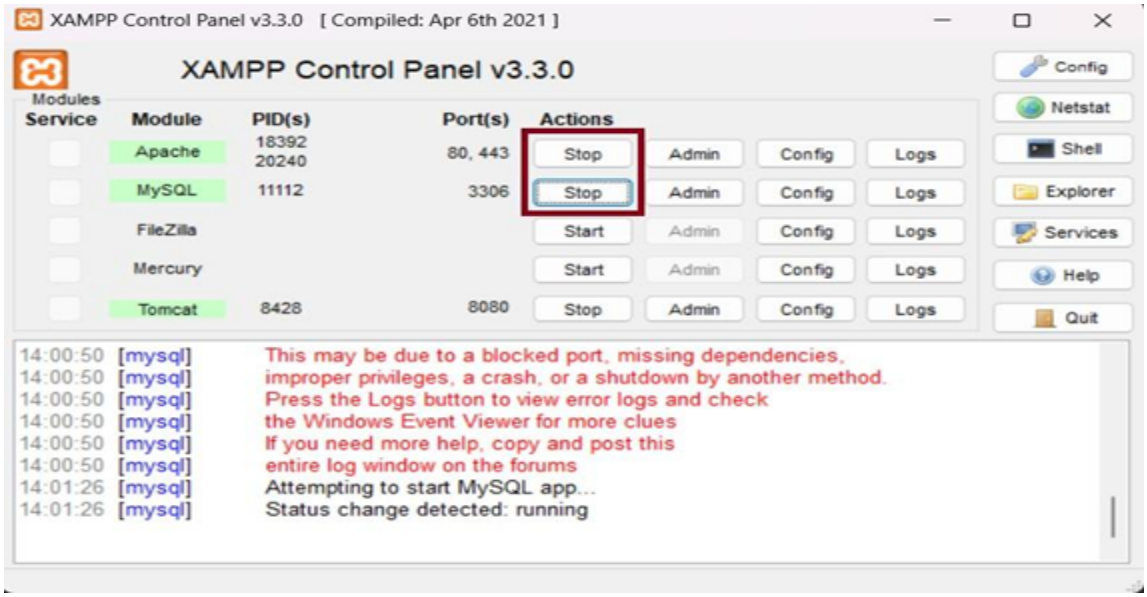
Step 2: Setup a file that is to be hosted on the server. Make sure the file has extension .php



Step 3: Go to the directory where XAMPP was installed. Go to htdocs folder. Place your folder in this directory.



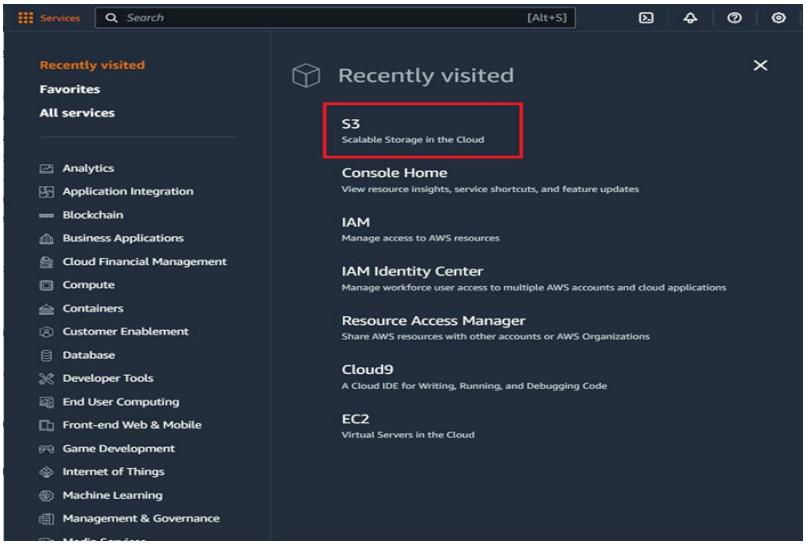
Step 4: Open XAMPP Control Panel, start the Apache service (Required) and mySQL service (if needed)



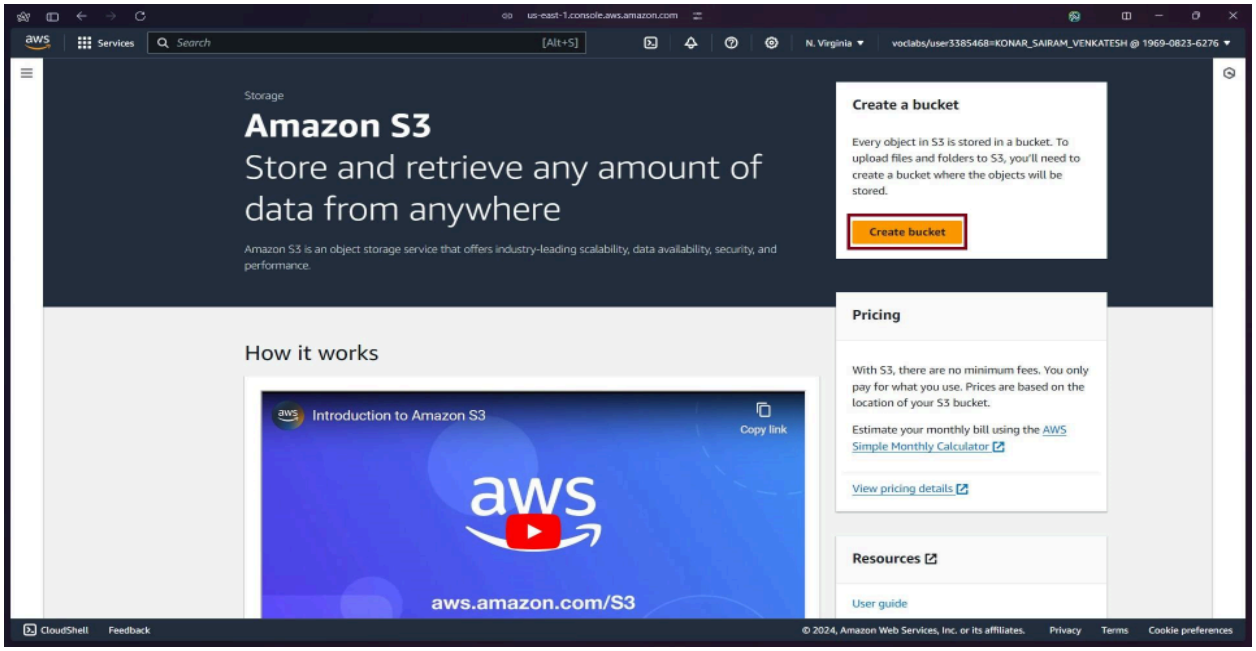
Step 5: Open your web browser. Type localhost/YOUR_FILENAME.php. This will open your website on your browser.

2) AWS S3

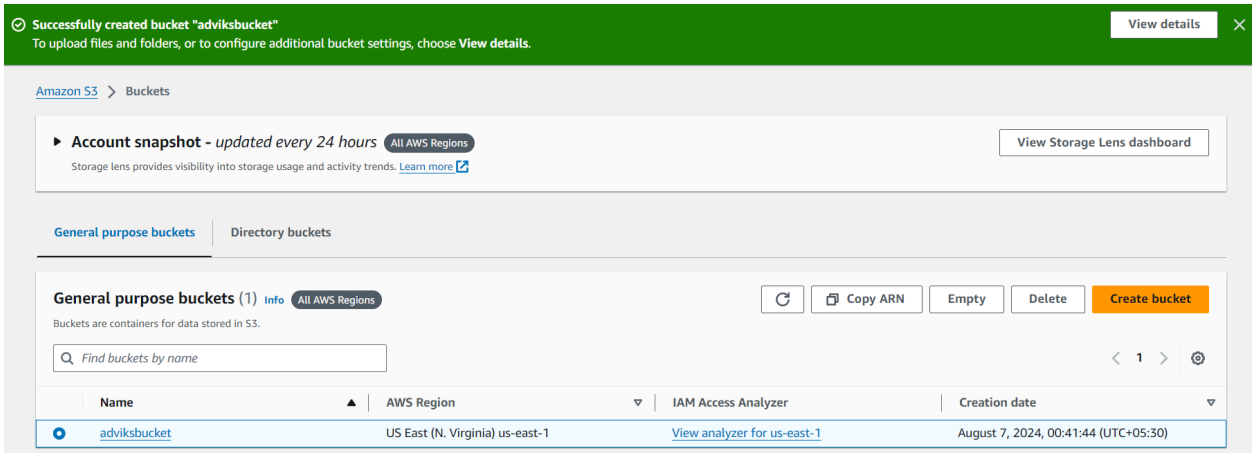
Step 1: Login to your AWS account. Go to services and open S3.



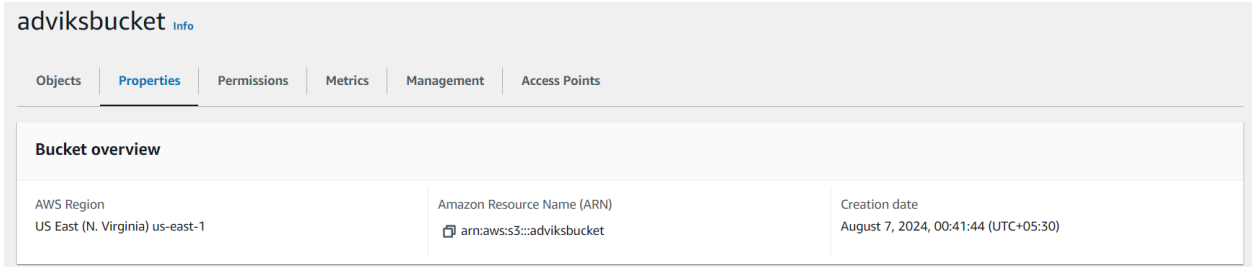
Step 2: Click on Create Bucket



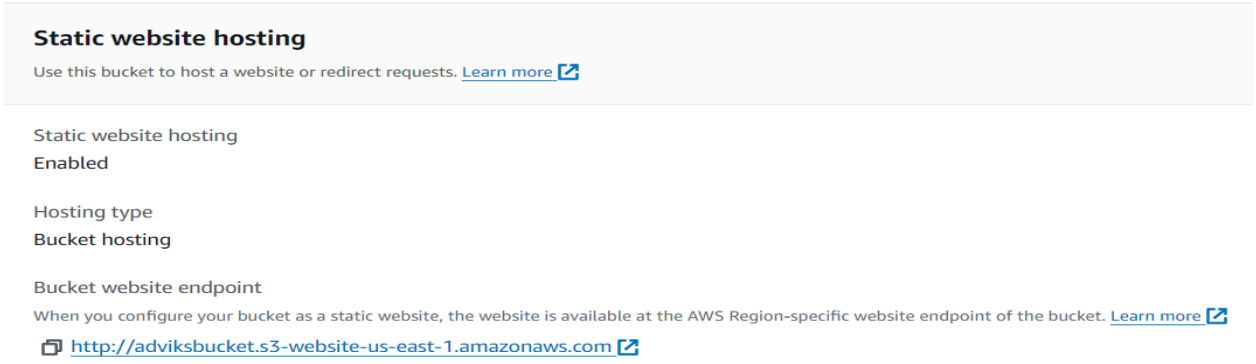
Step 3: Give a name to your bucket, keeping other options default, scroll down and click on Create Bucket



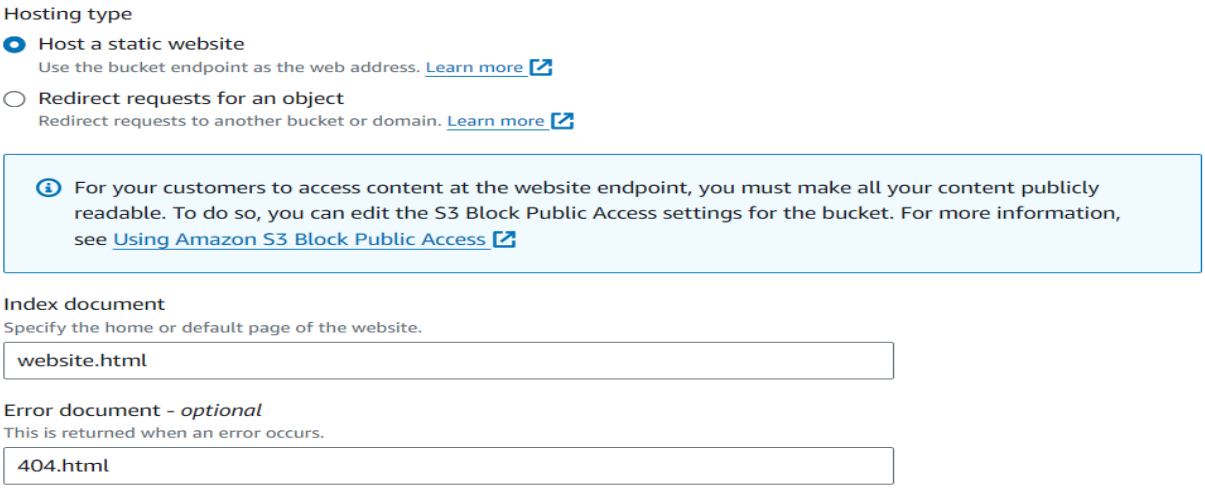
Step 4: Click on the name of your bucket and goto Properties



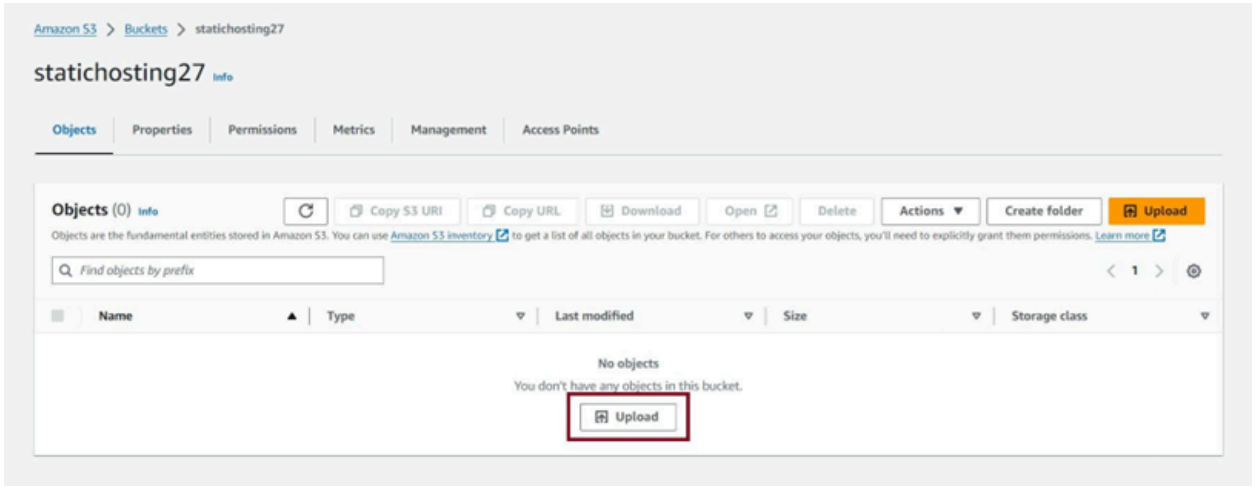
Step 5: Scroll down till you find Static website hosting, click on edit



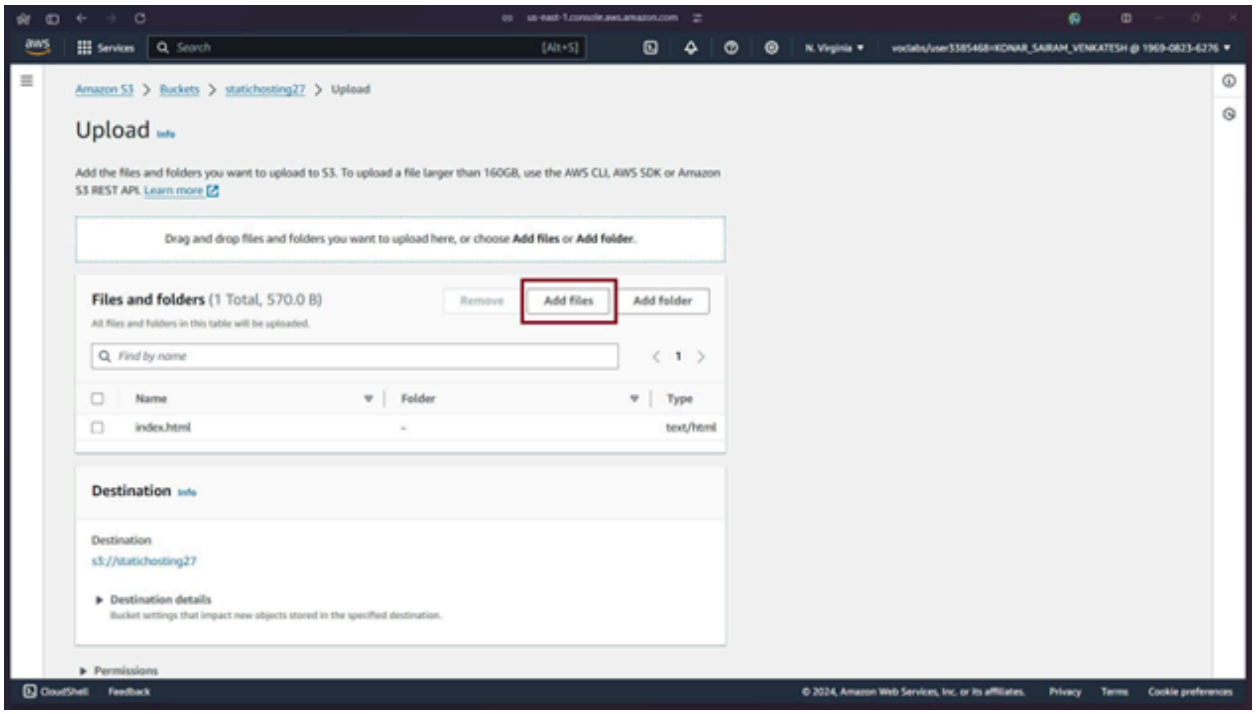
Step 6: Enable static website hosting, in Index document, write the name of your document and in error document, give name as 404.html. Save your changes.



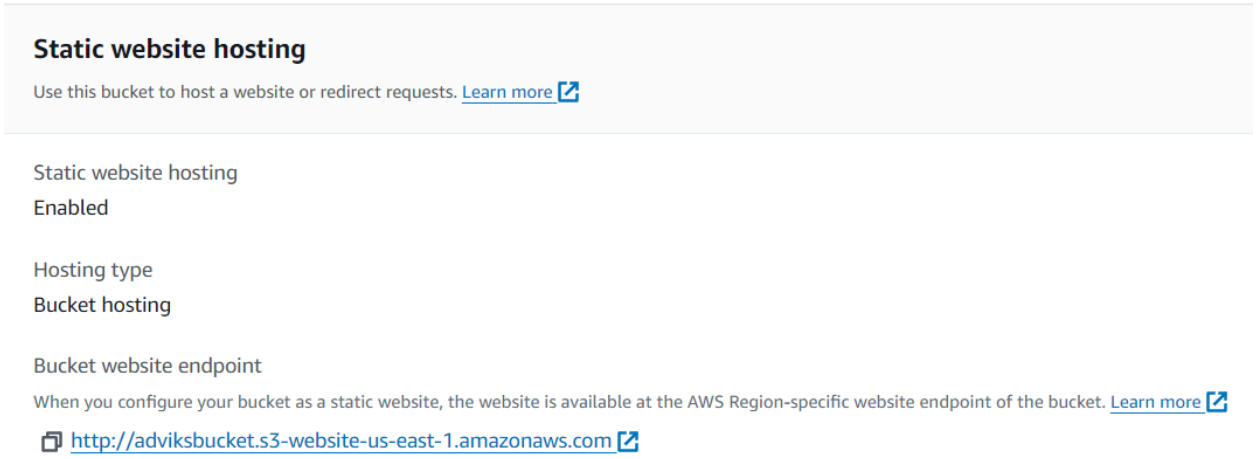
Step 7: Go to Objects tab and click on upload file.



Step 8: Click on Add files. Add all the files you want to upload. Then scroll down and click on Upload



Step 9: 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.



Step 10: 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

403 Forbidden

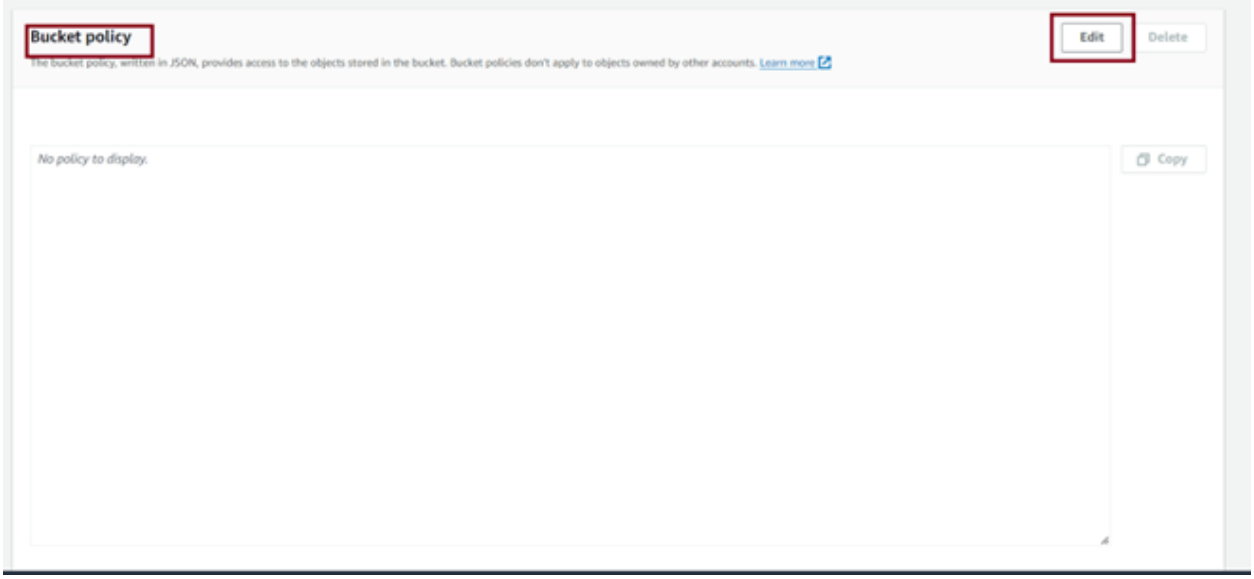
- Code: AccessDenied
- Message: Access Denied
- RequestId: QFWZ19ZNK2CSHN24
- HostId: 5tvtUKU1Jr0bR7xeXCiKQ2O89qapq//OQxXenu/o9umWWNH2blrLSZ11Sa3h0JMqG3CLAkZkB8U=

An Error Occurred While Attempting to Retrieve a Custom Error Document

- Code: AccessDenied
- Message: Access Denied

Step 11: Uncheck the Block all public access checkbox and click on save changes

Step 12: Scroll down to bucket policy and click edit



Step 13:

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

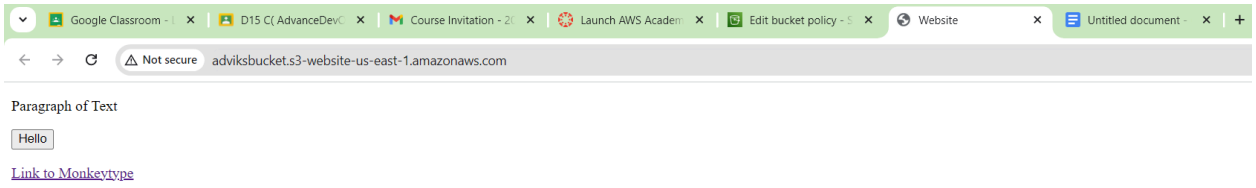
Paste this code snippet in the policy textarea. Replace YOUR-BUCKET-NAME-HERE with the name you have given to your bucket. Save the changes.

Bucket policy

The bucket policy, written in JSON, provides access to the objects stored in the buc

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

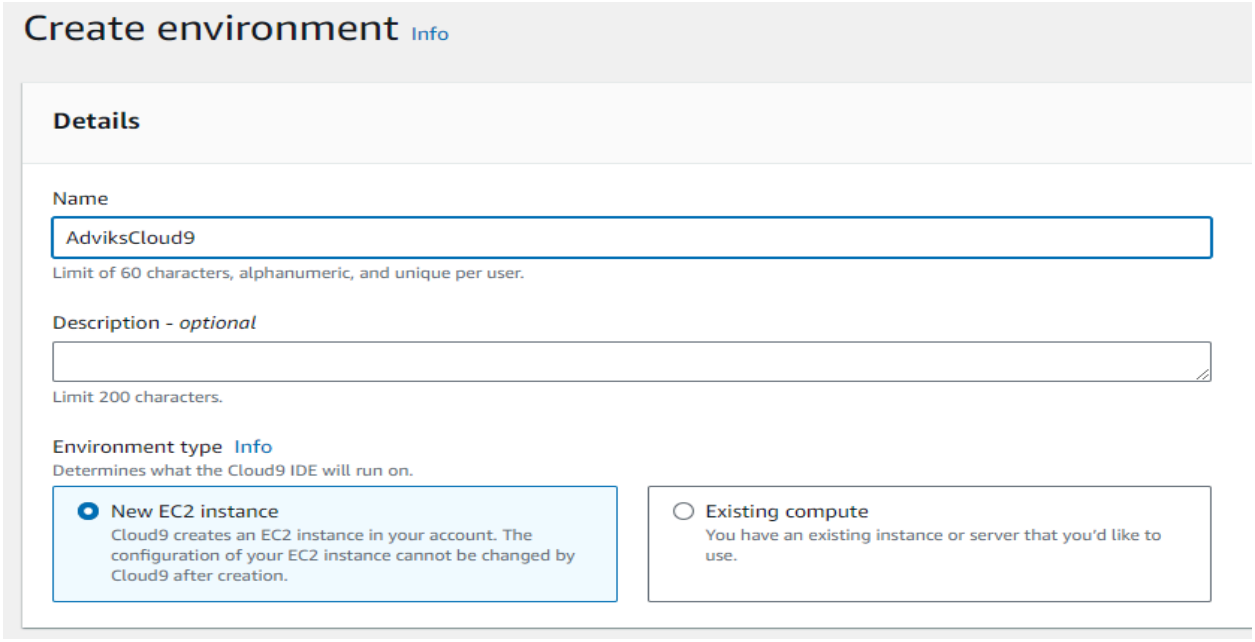
Step 14: Now reload the website. You can see your website

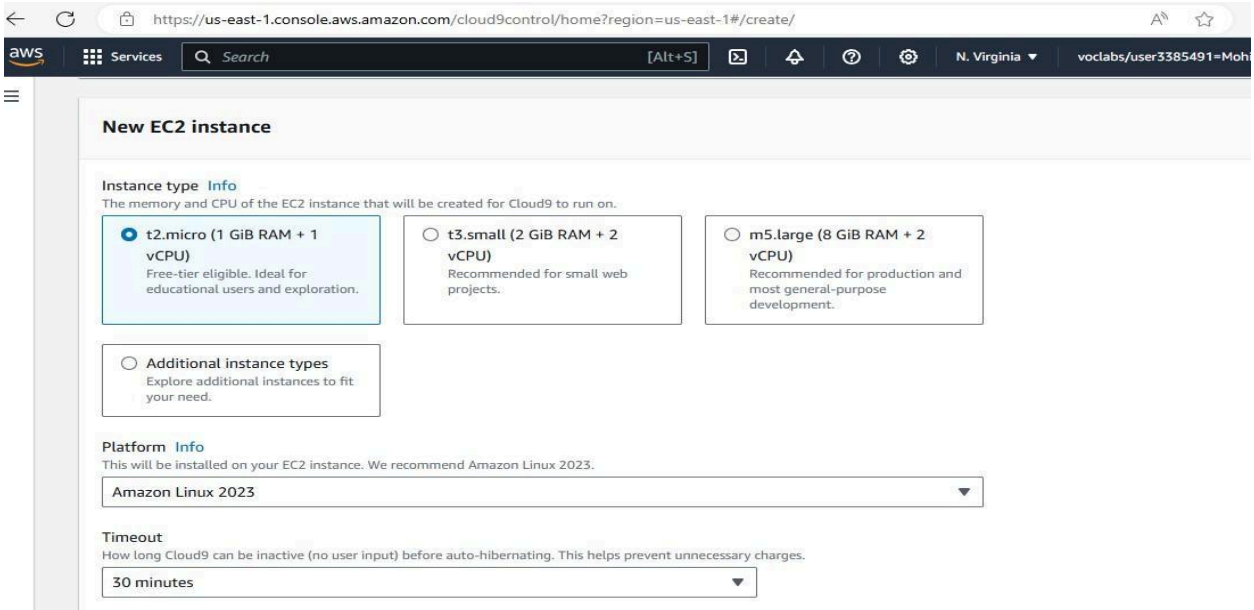


Name:Advik Hegde Div/Roll No. : D15C/15

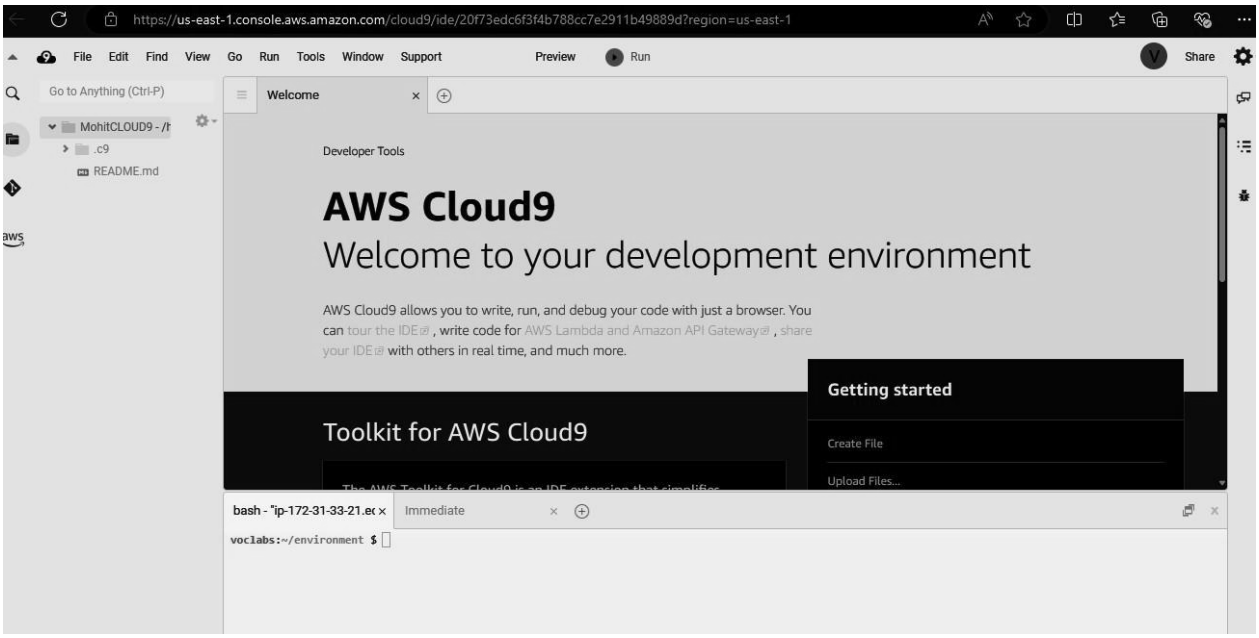
Exp 1b: Cloud9 Setup and Launch, Collaboration demonstration by creation of IAM groups and users.

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

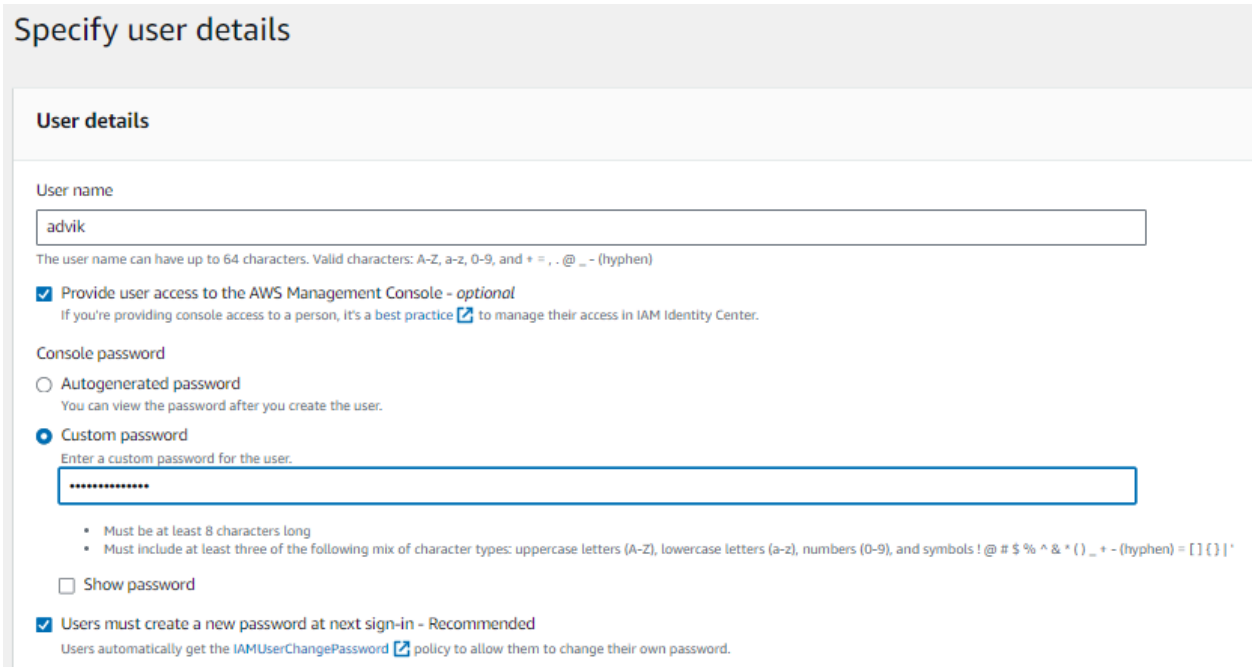


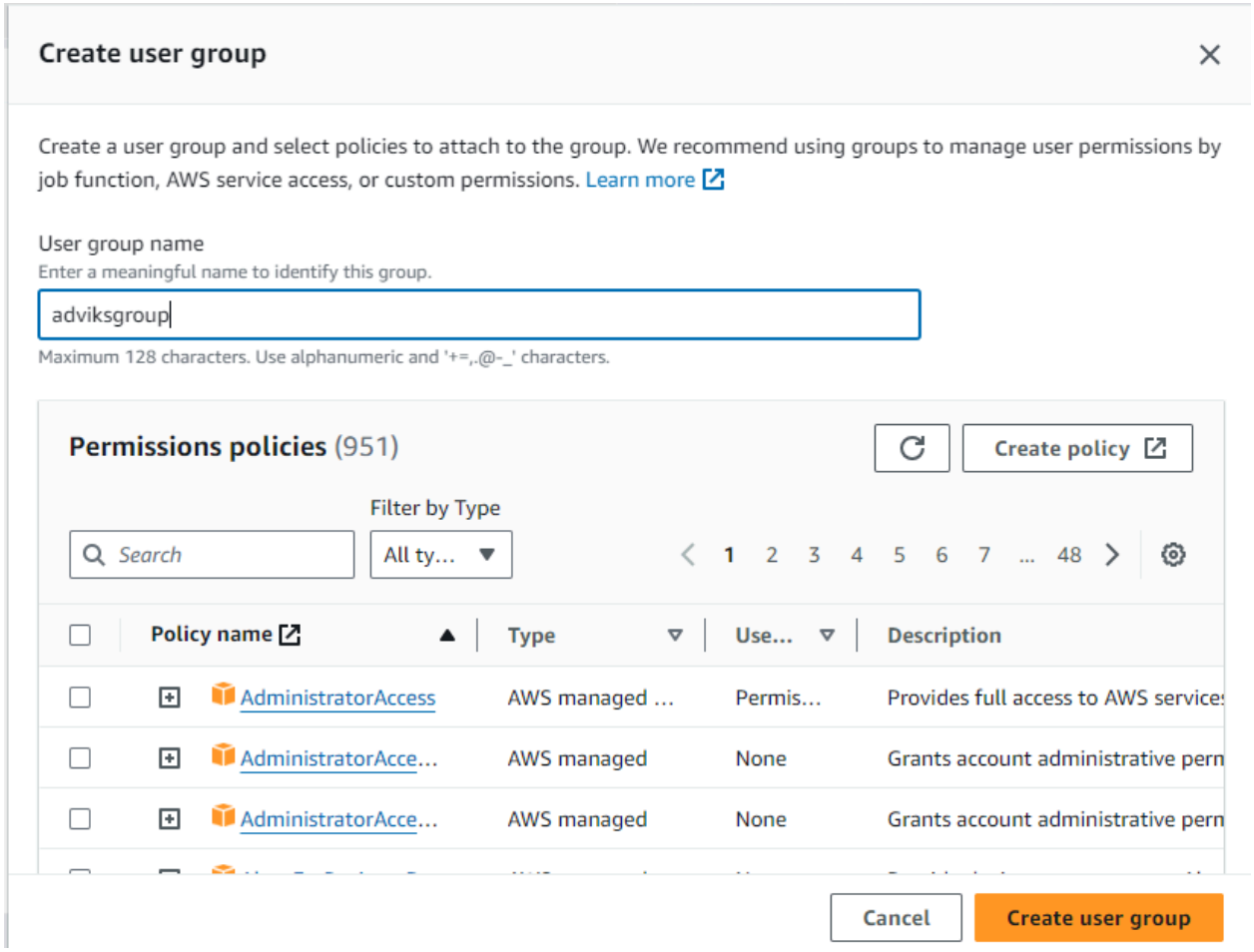


We have successfully set up and launched our Cloud9 environment. Over here, we can build and develop programs as per our desire. We are also allowed to collaborate with multiple other users and access shared resources.

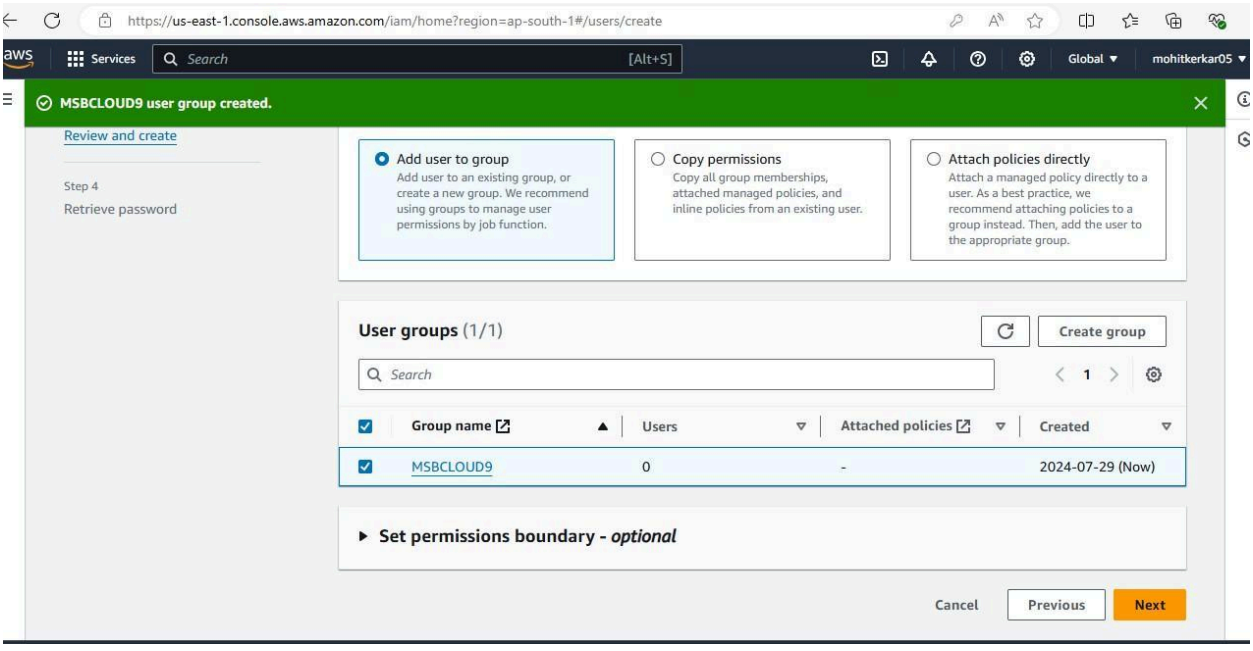


Moving on, we are supposed to create a new user. Give a suitable name to the user and decide the password for the same.

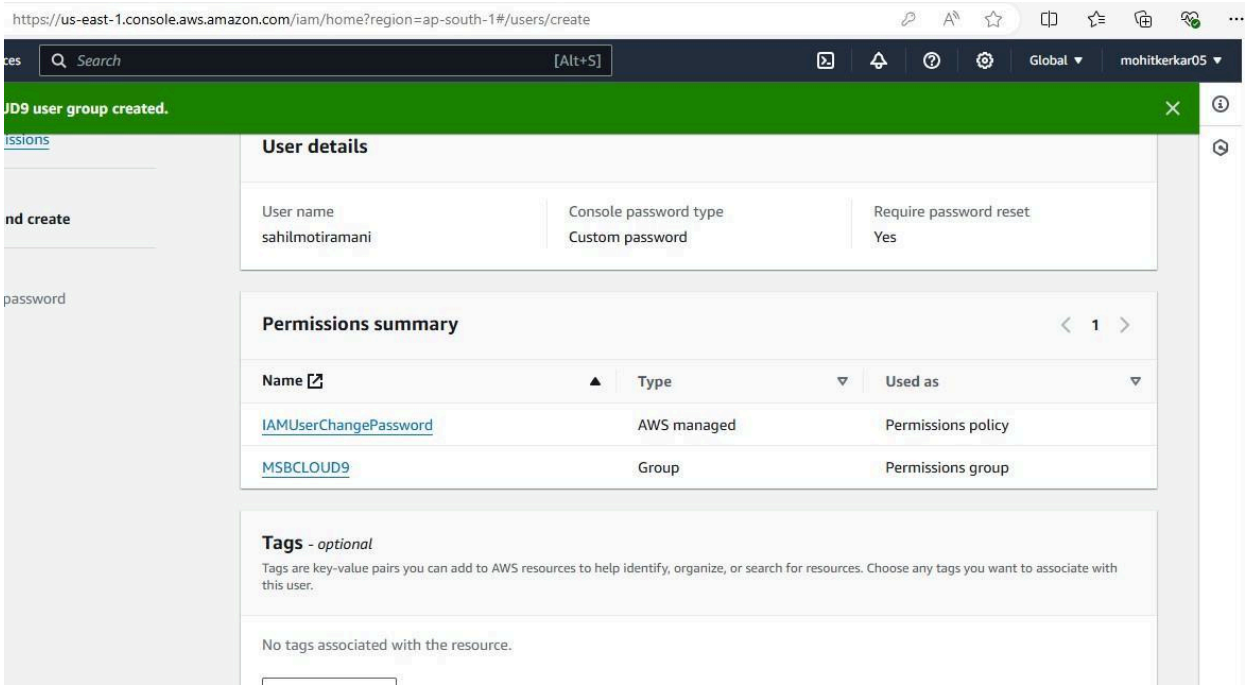




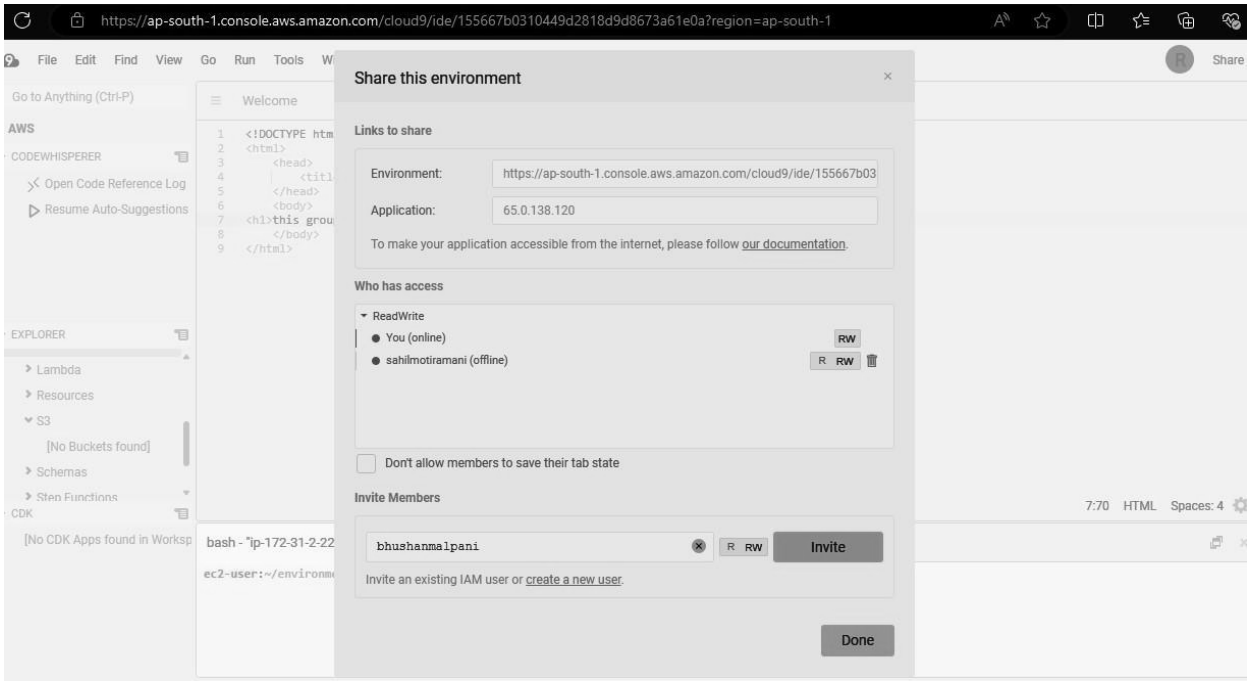
Similarly, create a new group and provide a suitable name for them. Include the IAM users in this group together for our convenience, that is, to provide similar kinds of permissions to the entire group rather than an individual user.

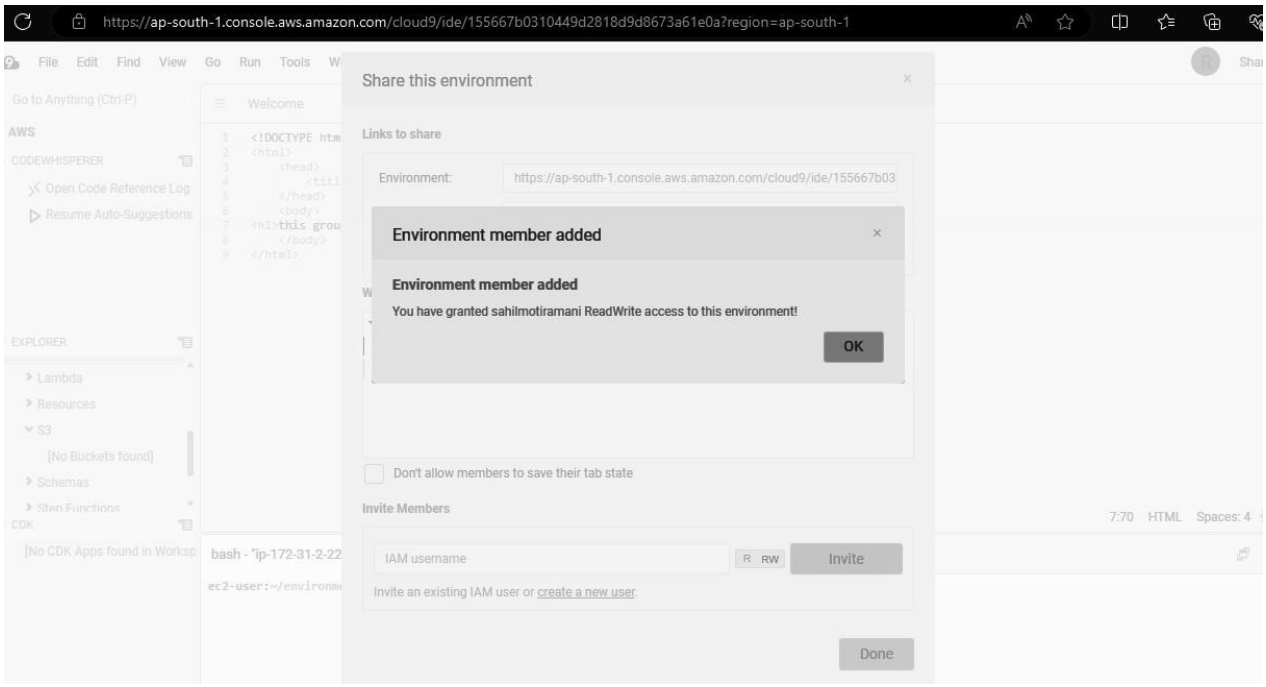


The user has successfully been created i.e. There is a custom-made username and a password for the IAM user.



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





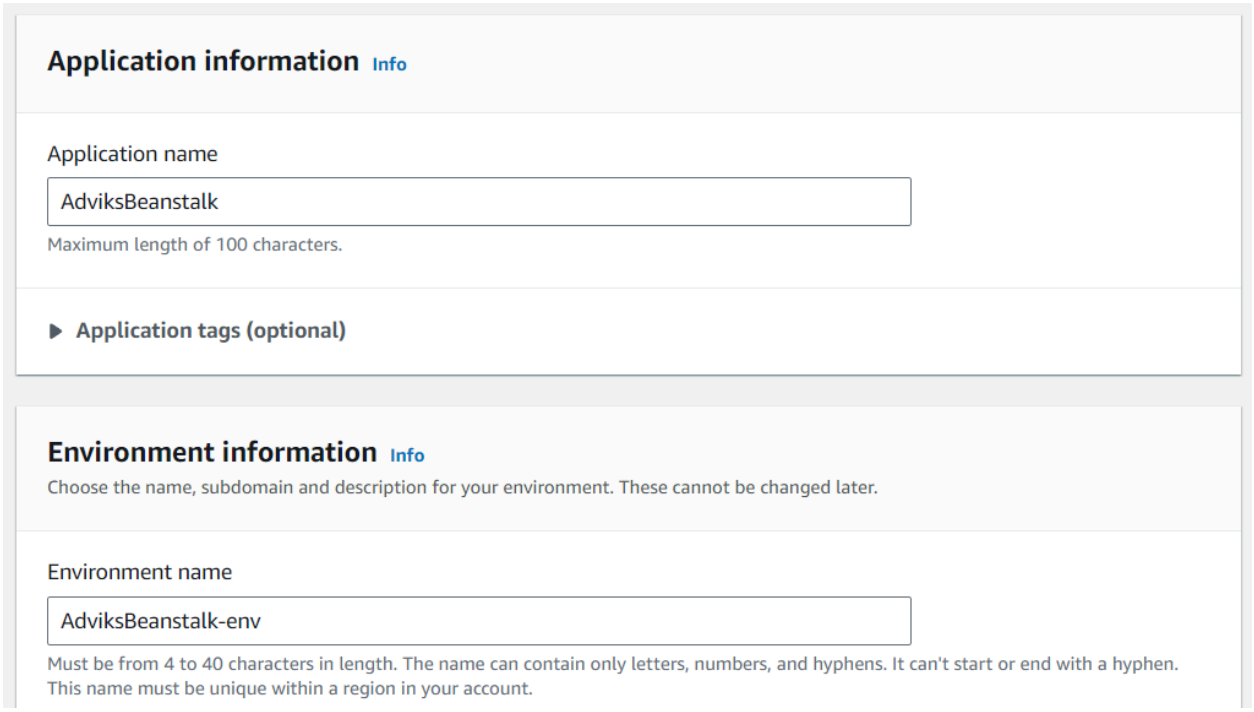
Further, we are supposed to login from another browser using the credentials of the IAM user, to access the shared cloud9 environment with us.
These steps could not be completed because Cloud9 services have been disrupted and there is no access to the IAM user from the remote login.

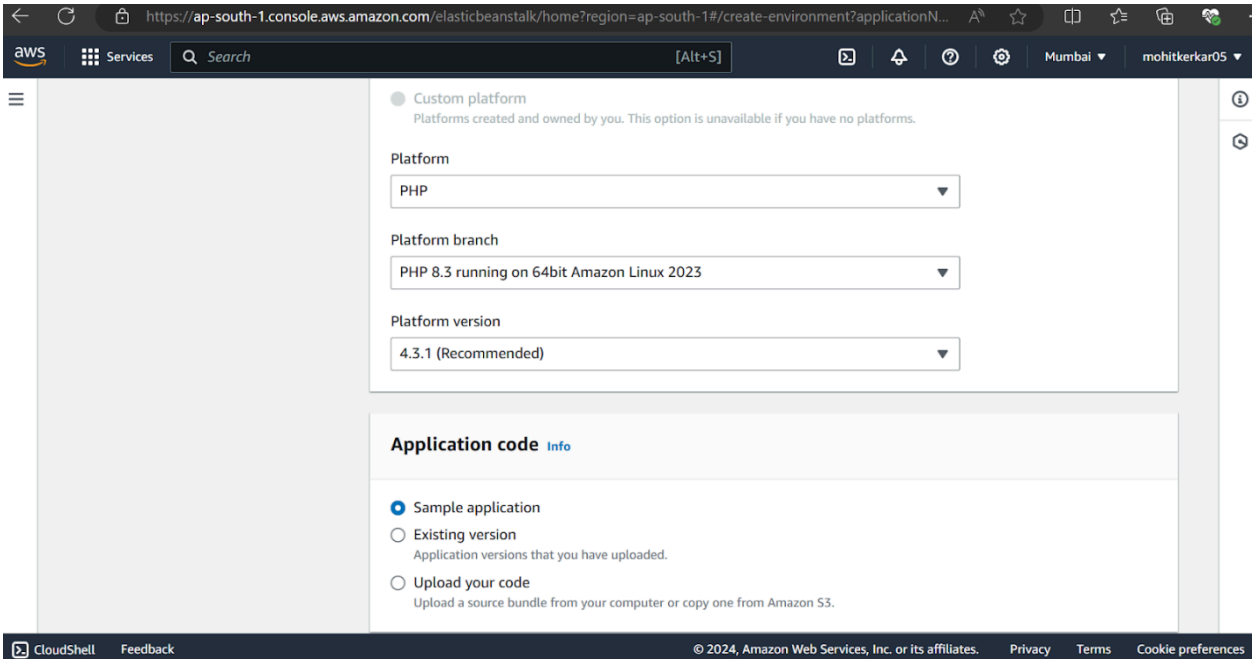
Name:Advik Hegde Div/Roll No. : D15C/15

Exp 02:To Build Your Application using AWS Code Build and Deploy on S3 / SEBS using AWS CodePipeline, deploy Sample Application on EC2 instance using AWS Code Deploy.

Step 1: Create our Elastic Beanstalk Environment

Login into your AWS account and navigate to services. Search for Elastic Beanstalk service and click on create application. Give your application a suitable name. For the platform, select PHP. The rest of the configuration settings are to be kept as default.

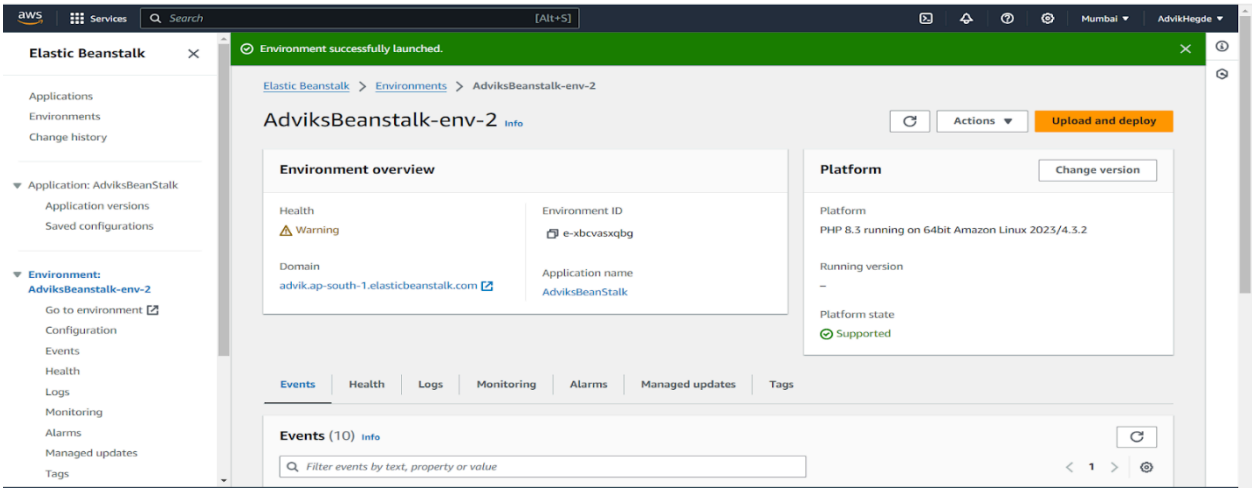




Now, while creating the environment, we are asked to provide an IAM role with the necessary EC2 permissions. We are supposed to make sure that we have made an existing IAM role with the following set of permissions:

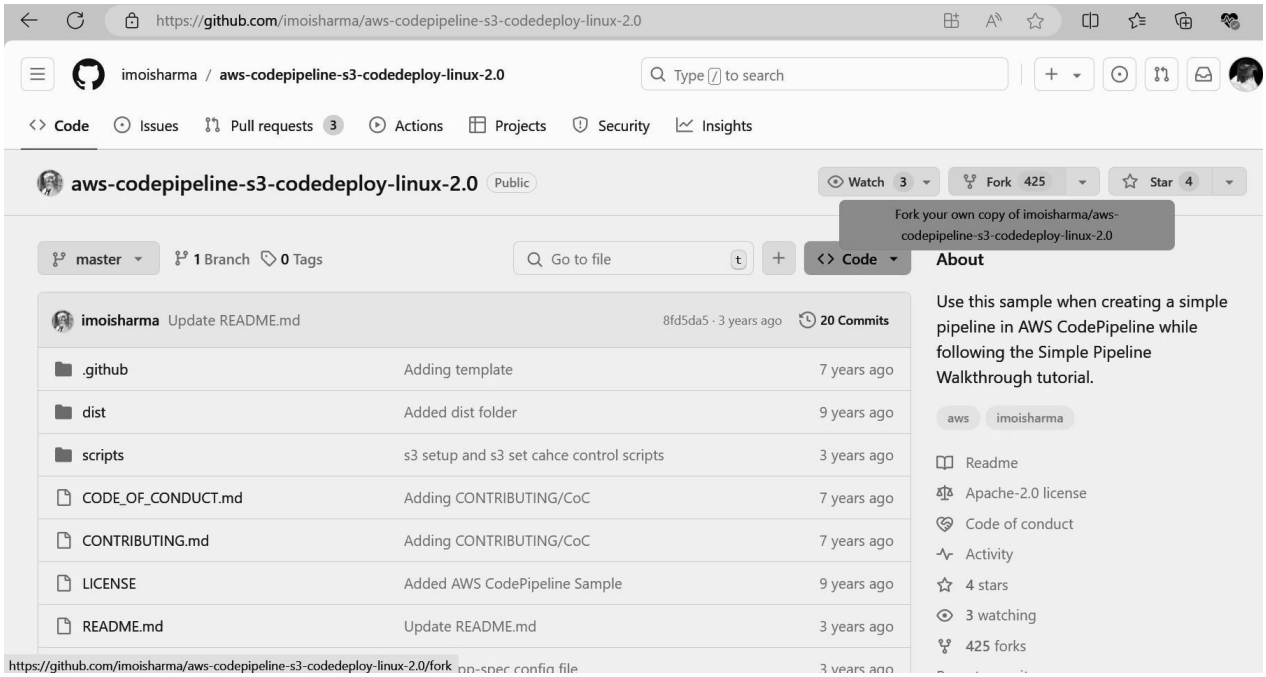
1. AWSElasticBeanStalkWebTier
2. AWSElasticBeanStalkWorkerTier
3. AWSElasticBeanStalkMulticontainerDocker

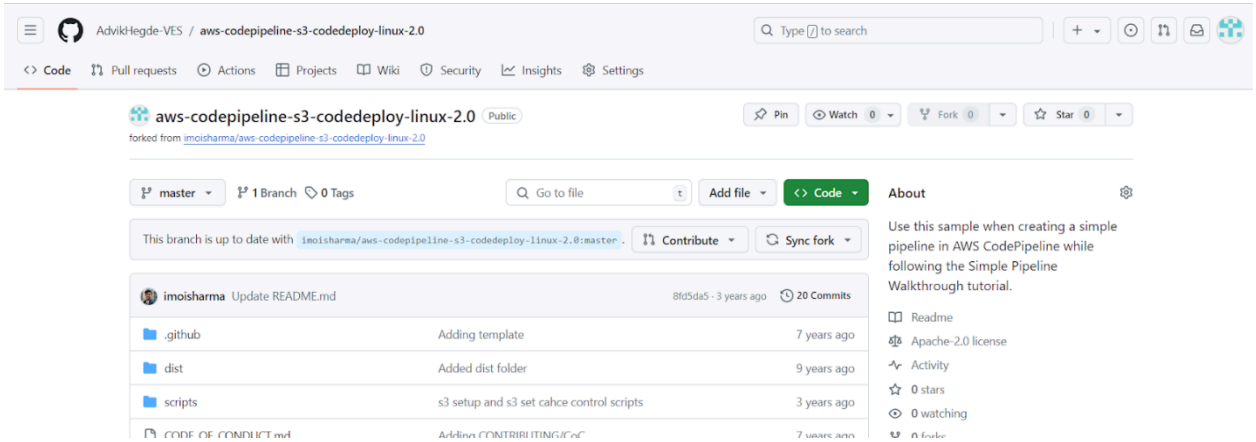
We can skip the steps to follow after the initial few steps mentioned above and move straight to review the settings of our environment. After reviewing everything properly, our environment can successfully be created.



Step 2: Fork the required repository onto our github account

The repository to be forked is - imoisharma/aws-codepipeline-s3-codedeploy-linux-2.0

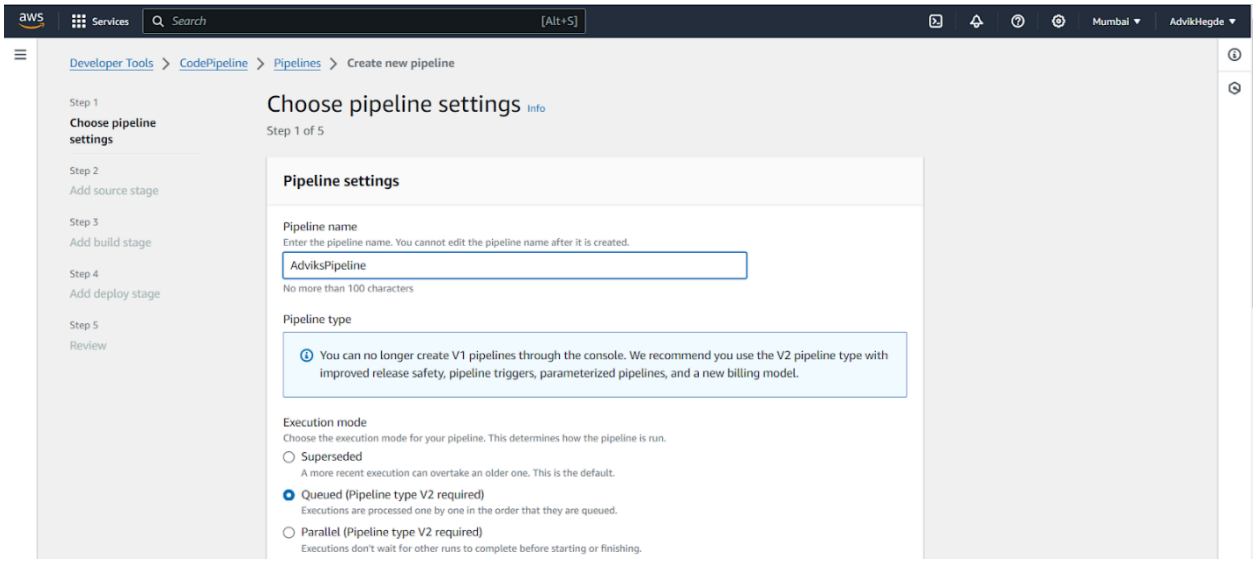




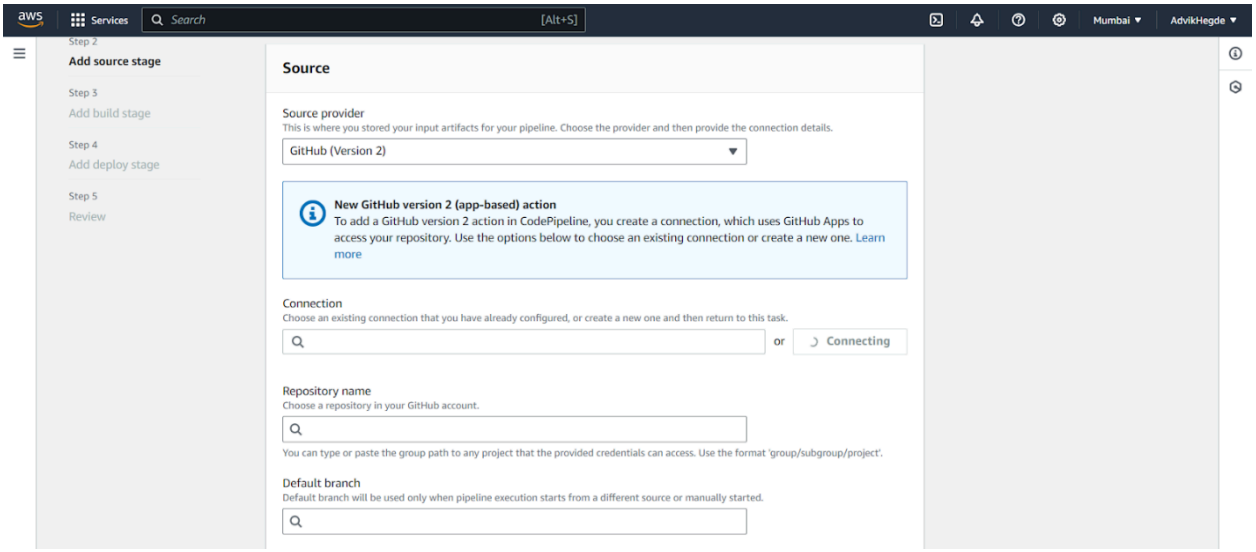
This step is necessary for the execution of the steps to follow. It will be helpful in the creation of a pipeline.

Step 3: Creation of the Pipeline

Navigate to Codepipeline inside Developer Tools. Give a suitable name to the pipeline you want to create.

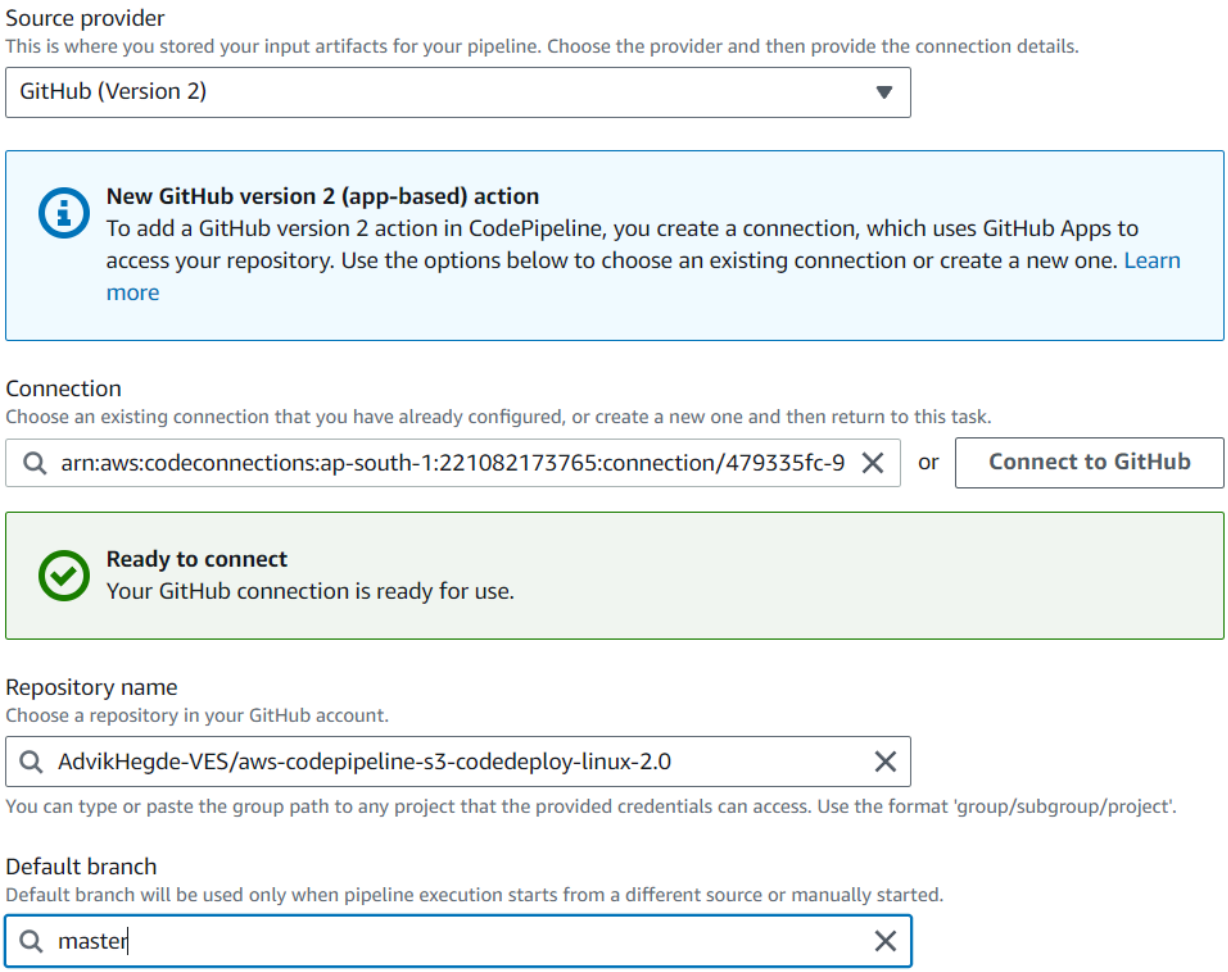


And click on next …

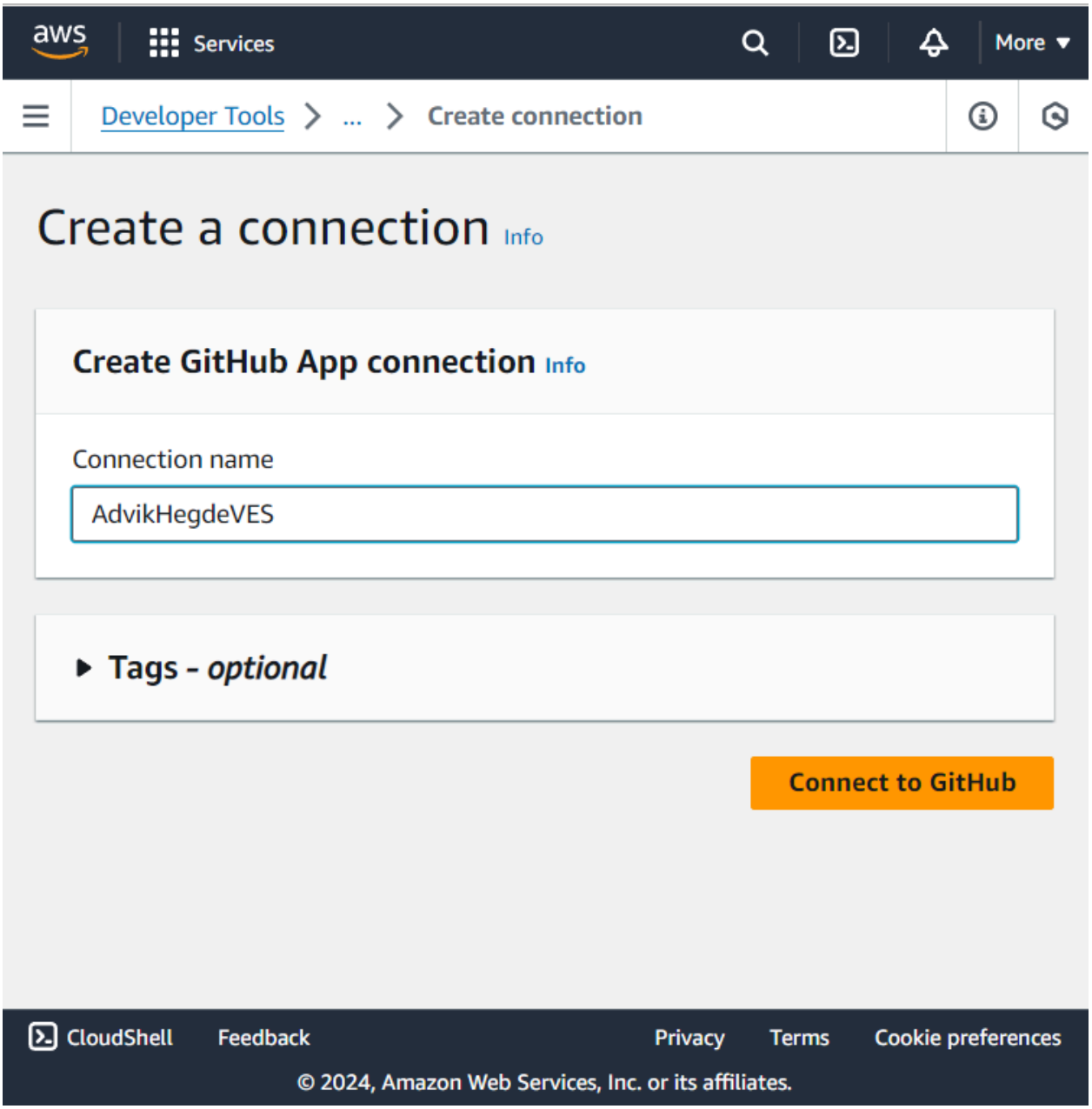


Step 4: GitHub connection

In this step, we are supposed to create a GitHub connection and add our existing repository over here i.e. the one we forked earlier.

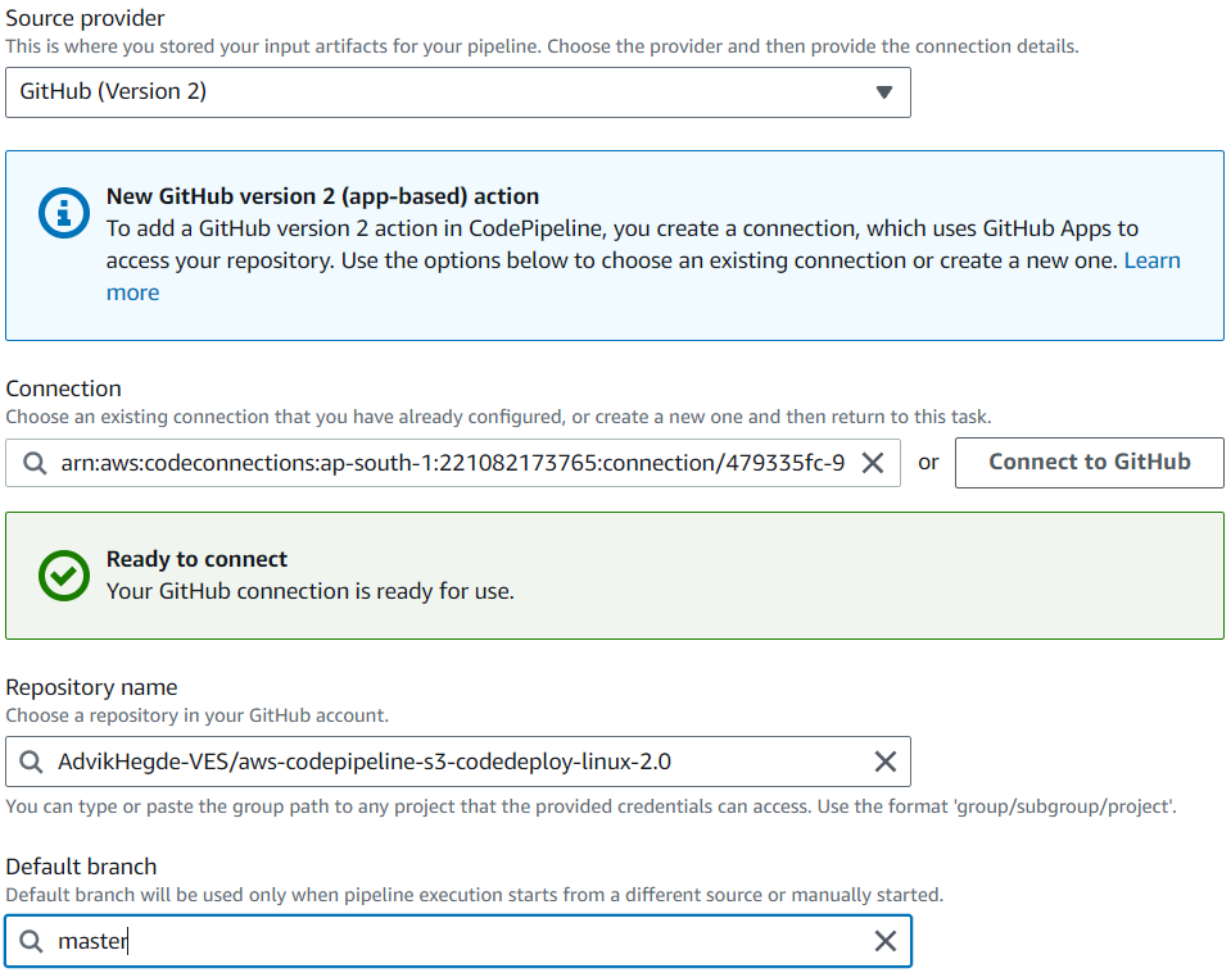


We are supposed to enter our GitHub username so as to proceed towards making the connection.


aws
Services
More
Developer Tools
...
Create connection
Create a connection
Info
Create GitHub App connection
Info
Connection name
AdvikHegdeVES
Tags - optional
Connect to GitHub
CloudShell
Feedback
Privacy
Terms
Cookie preferences
© 2024, Amazon Web Services, Inc. or its affiliates.

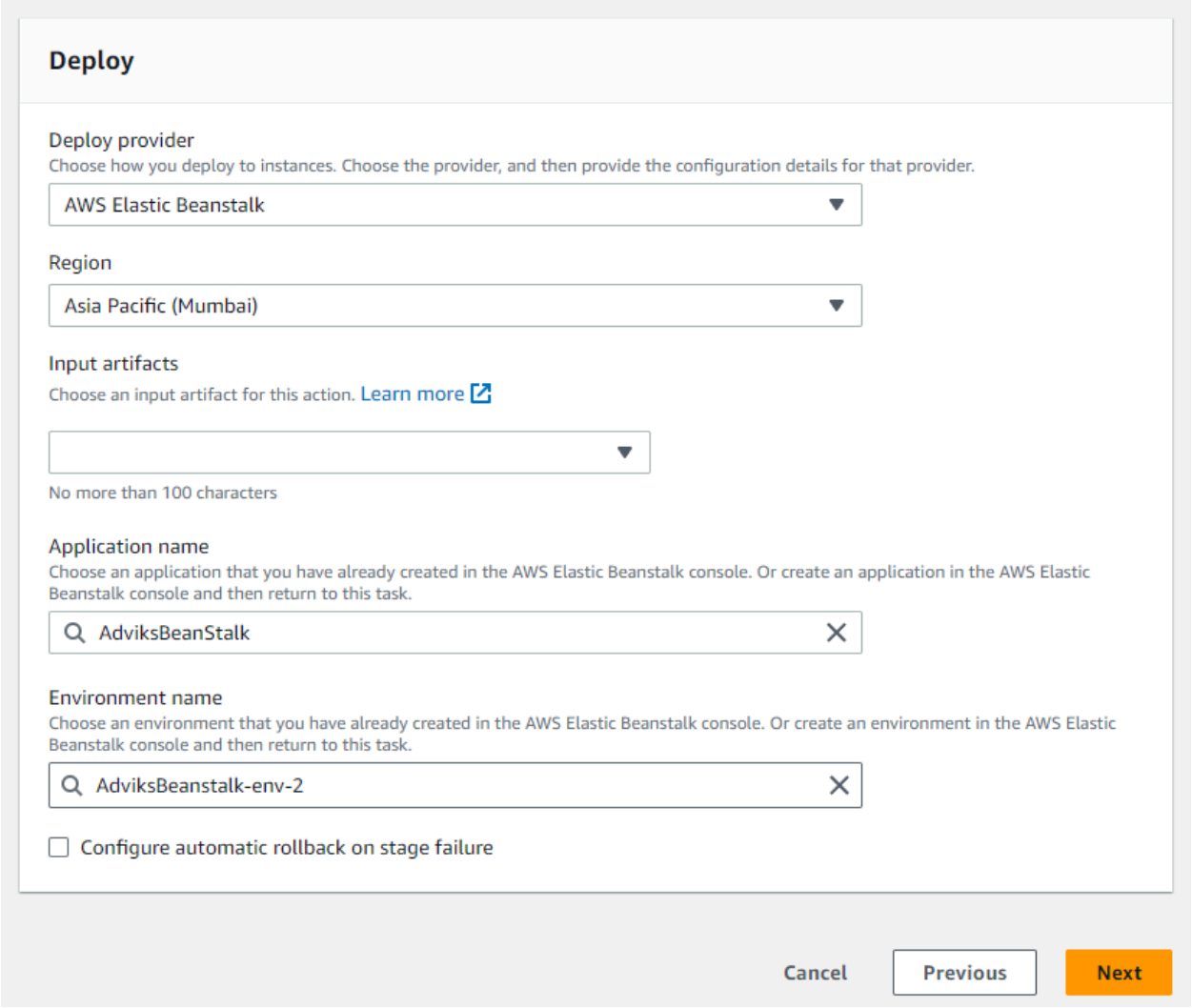
Now to finalize our connection, we are to install an application which connects AWS to our GitHub account and repository.

Post the establishment of the connection, this is the message that is displayed. We can further select the branch of our repository that we want to connect.

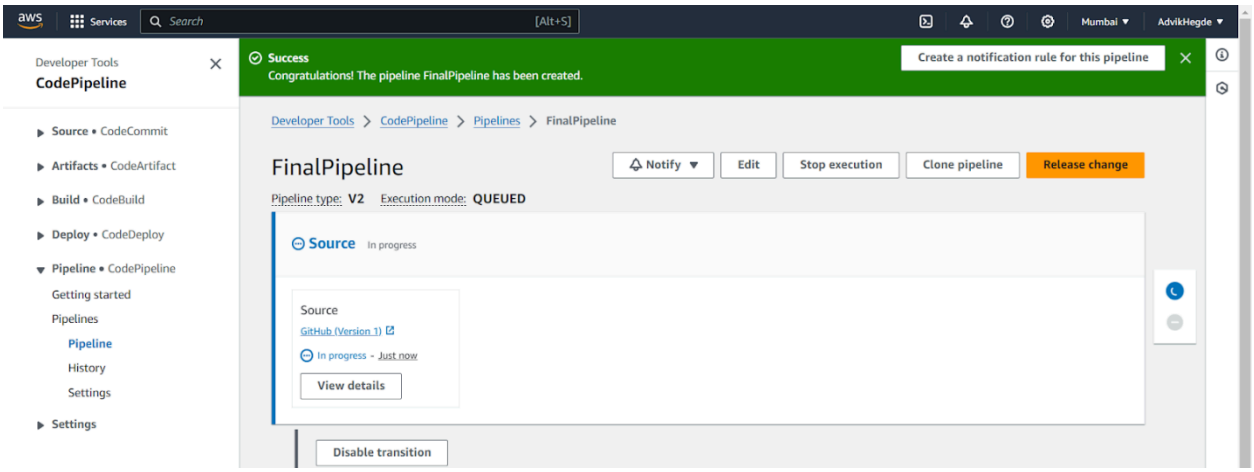


Step 5: Deployment stage:

We are expected to skip the build stage and move towards the deployment step. In the deployment step we are supposed to choose the Elastic Beanstalk application and the environment that we created earlier and proceed with our pipeline creation.

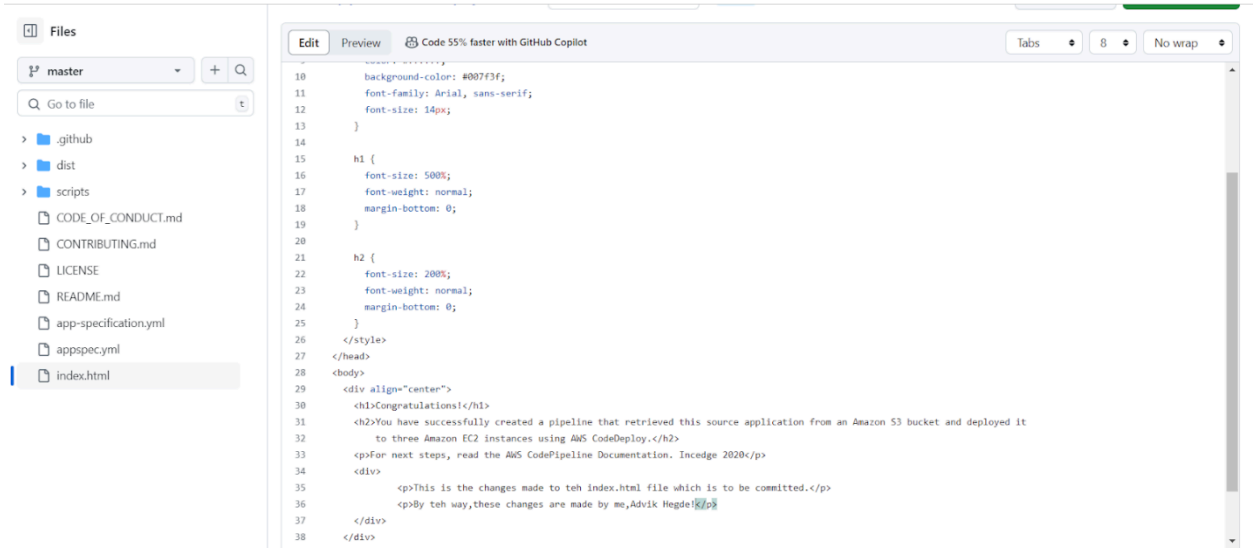


Step 6: Post deployment stage: When all the stages run successfully, this is what is displayed onto the screen. It shows us that our application and our environment have successfully been deployed using a dedicated pipeline created.



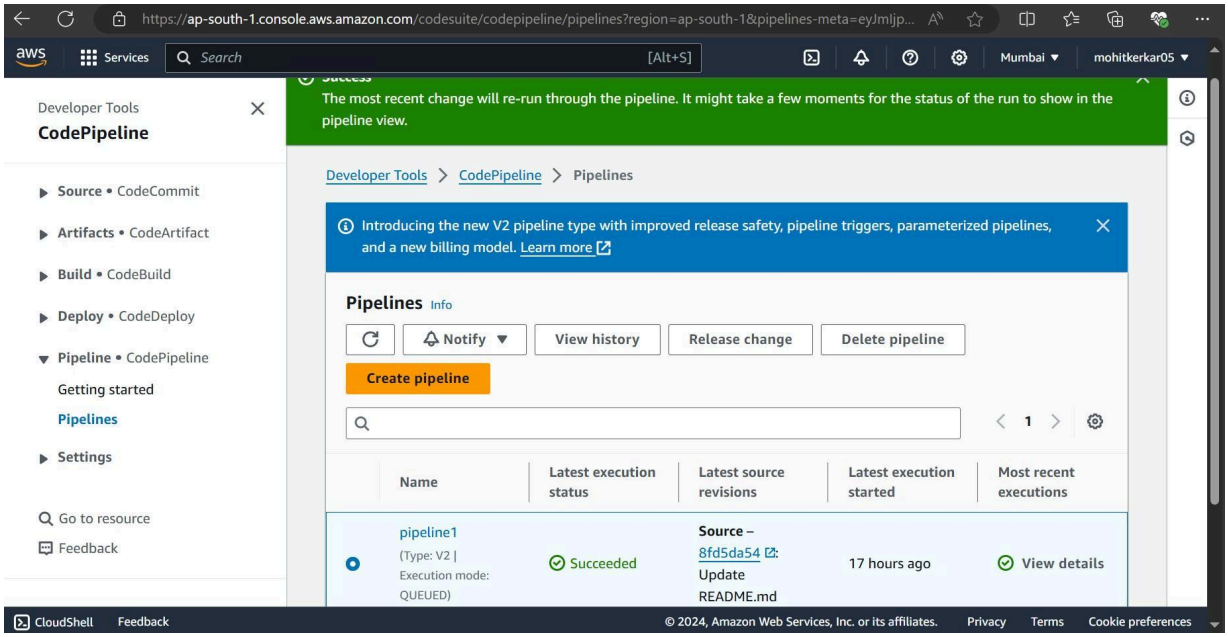
Step 7: Committing changes to your GitHub code

Now, we will go to our forked repository and make some changes to the index.html file. On making the desired changes, we are supposed to commit those changes on our forked repository. Write a good commit message so as to recognize it when it appears on the pipeline.

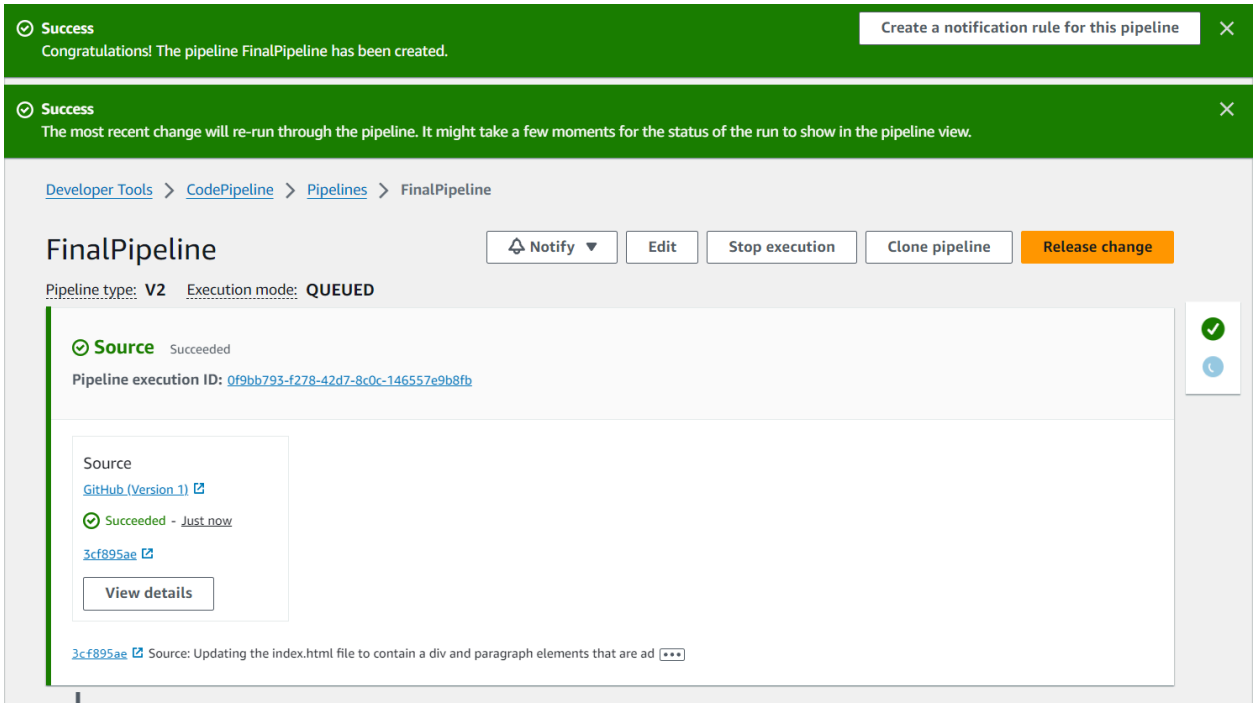


Step 8: Apply the newly made changes in index.html onto our pipeline

Come back to the Codepipeline section and select the pipeline through which we successfully created and deployed our application. Click on the release change option to apply the latest changes/commits from our GitHub repository to our pipeline.



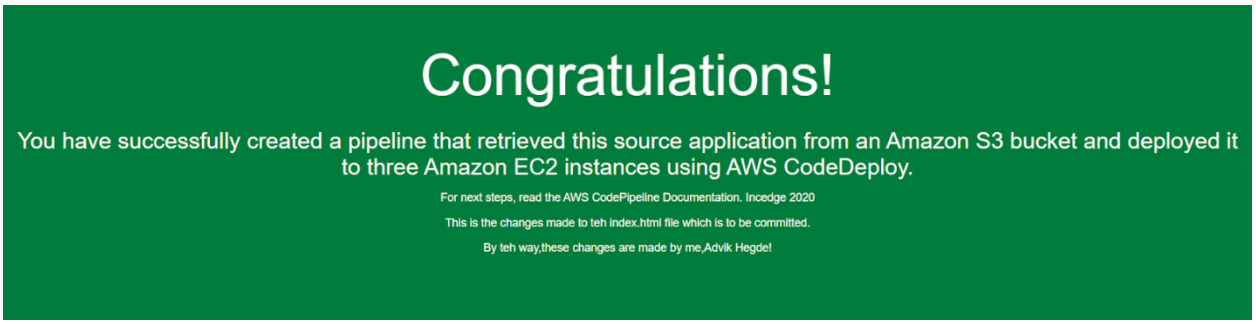
Once the changes have been applied, we see the commit message that we wrote for the latest commit on our repository being reflected on our pipeline. Over here, it would be seen somewhere near the bottom of the image that is attached. “Update index.html” was the latest commit message in the GitHub repository.



Step 9: Open the Domain of our Elastic Beanstalk environment

Now, we navigate back to our Elastic Beanstalk environment and open the environment domain of our deployed application.

The text in this image is clearly distinguishable from the earlier website’s text meaning that the changes that we made to our code in index.html has successfully been applied to the website that we deployed.



Name: **Advik Hegde** Div/Roll no.:**D15C/15**

Advanced DevOps Lab
Experiment No:3

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

Theory:

Container-based microservices architectures have profoundly changed the way development and operations teams test and deploy modern software. Containers help companies modernize by making it easier to scale and deploy applications, but containers have also introduced new challenges and more complexity by creating an entirely new infrastructure ecosystem.

Large and small software companies alike are now deploying thousands of container instances daily, and that's a complexity of scale they have to manage. So how do they do it?

Enter the age of Kubernetes.

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.

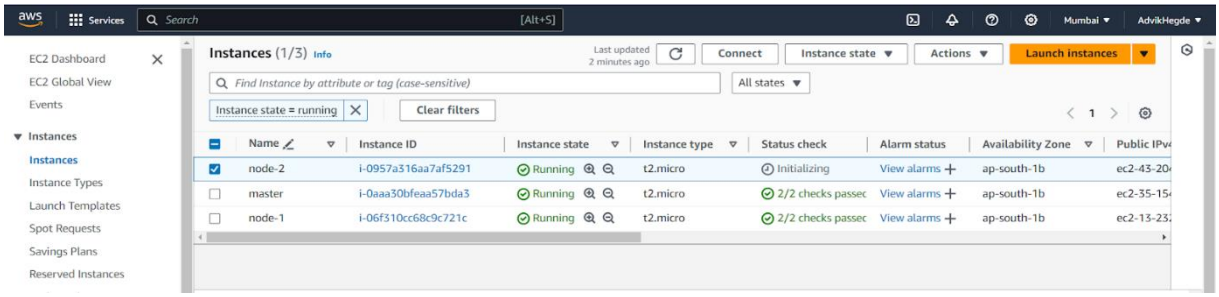
Kubernetes makes it easy to deploy and operate applications in a microservice architecture. It does so by creating an abstraction layer on top of a group of hosts so that development teams can deploy their applications and let Kubernetes manage the following activities:

- Controlling resource consumption by application or team
- Evenly spreading application load across a hosting infrastructure
- Automatically load balancing requests across the different instances of an application
- Monitoring resource consumption and resource limits to automatically stop applications from consuming too many resources and restarting the applications again
- Moving an application instance from one host to another if there is a shortage of resources in a host, or if the host dies
- Automatically leveraging additional resources made available when a new host is added to the cluster
- Easily performing canary deployments and rollbacks

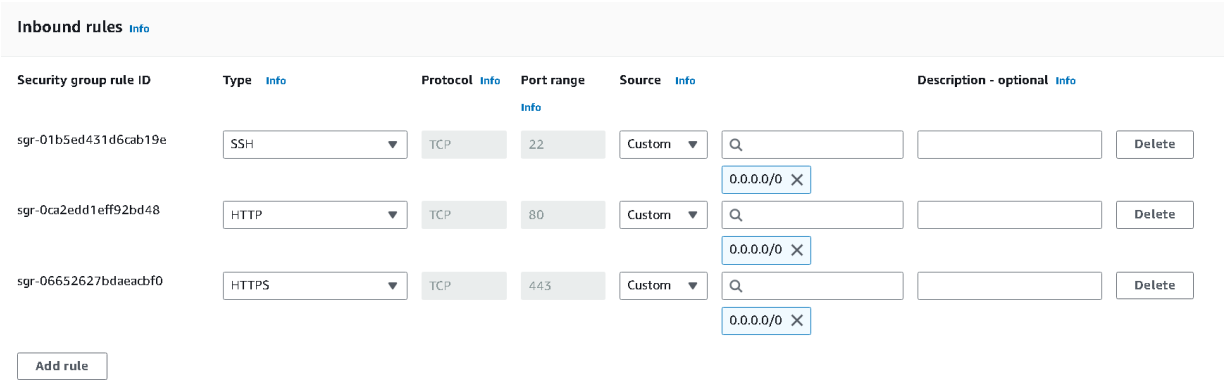
Steps:

1. Create 3 EC2 Ubuntu Instances on AWS.

(Name 1 as Master, the other 2 as worker-1 and worker-2)



2. Edit the Security Group Inbound Rules to allow SSH



3.SSH into all 3 machines

a. You can do it through the aws console directly

or

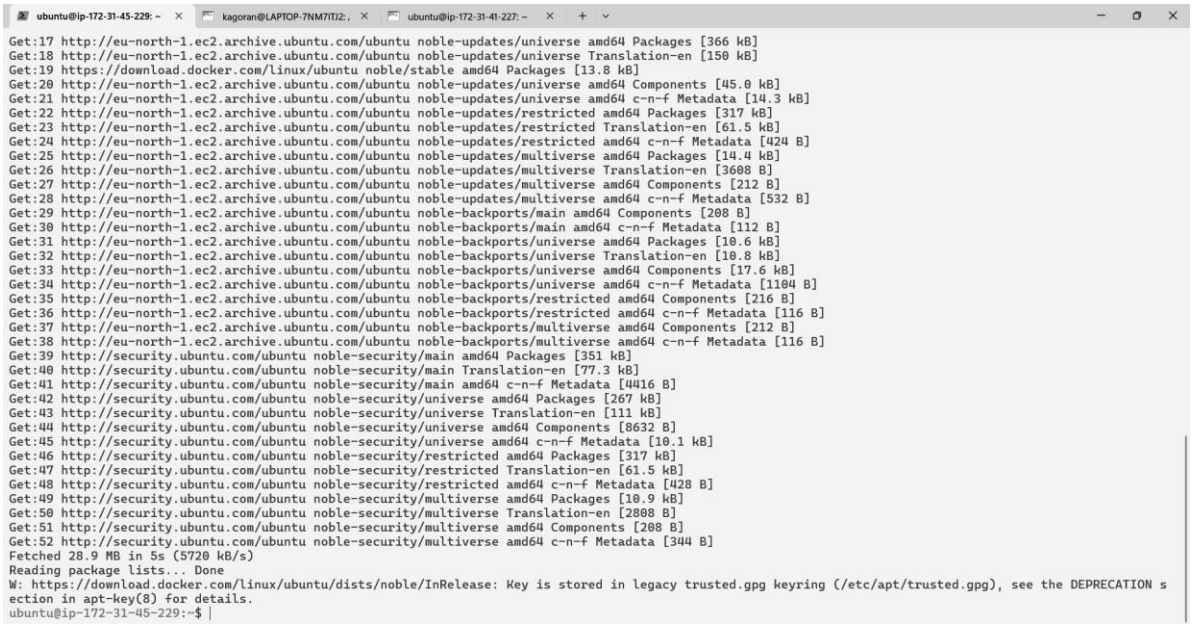
b. Locate your key from the Downloads folder and open it in cmd and paste this command

ssh -i <-your-key->.pem ec2-user<ip-address of instance>

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

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"], "log-driver":
    "json-file",
    "log-opts": {
       "max-size": "100m"
    },
    "storage-driver": "overlay2"
}
EOF
sudo systemctl enable docker sudo
systemctl daemon-reload sudo
systemctl restart  docker
```

Install Kubernetes on all 3 machines

```
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-172-31-40-255:~$ # Add Kubernetes GPG key
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -

# Add Kubernetes repository
sudo tee /etc/apt/sources.list.d/kubernetes.list <<EOF
deb https://apt.kubernetes.io/ kubernetes-xenial main
EOF

# Update package list
sudo apt-get update

# Install kubelet, kubeadm, and kubectl
sudo apt-get install -y kubelet kubeadm kubectl

# Hold the versions of Kubernetes components
sudo apt-mark hold kubelet kubeadm kubectl
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
deb https://apt.kubernetes.io/ kubernetes-xenial main
Hit:1 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Ign:6 https://packages.cloud.google.com/apt kubernetes-xenial InRelease
Err:7 https://packages.cloud.google.com/apt kubernetes-xenial Release
```

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

```
ubuntu@ip-172-31-45-229:~$ # Disable swap
sudo swapoff -a

# Allow bridging for iptables
echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf

# Apply sysctl changes
sudo sysctl -p
net.bridge.bridge-nf-call-iptables=1
net.bridge.bridge-nf-call-iptables = 1
ubuntu@ip-172-31-45-229:~$
```

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

Initialize the Kubecluster

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

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

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

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

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

  export KUBECONFIG=/etc/kubernetes/admin.conf

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

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

kubeadm join 172.31.45.229:6443 --token s9zq75.bsi7js5f62ridulc \
        --discovery-token-ca-cert-hash sha256:91eae090fdd49337bf70d5bf7478e60bc85820d0996651871129a082db6fa8f1
ubuntu@ip-172-31-45-229:~$
```

Copy the join command and keep it in a notepad, we'll need it later.

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 flammel file as mentioned in the code.

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

```
ubuntu@ip-172-31-45-229:~$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
namespace/kube-flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
```

Check the created pod using this command

Now, keep a watch on all nodes using the following command

watch kubectl get nodes

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

sudo kubeadm join <ip> --token <token> \
--discovery-token-ca-cert-hash <hash>

Now, notice the changes on the master terminal

```
Every 2.0s: kubectl get nodes                                                    ip-172-31-45-229: Sat Sep 14 12:19:42 20
24
NAME              STATUS   ROLES           AGE   VERSION
ip-172-31-45-229  Ready    control-plane   28m   v1.31.1
```

That's it, we now have a Kubernetes cluster running across 3 AWS EC2 Instances. This cluster can be used to further deploy applications and their loads being distributed across these machines.

Conclusion:

In this setup, we established a Kubernetes cluster utilizing three AWS EC2 instances, with Docker and Kubernetes components successfully deployed on each. While the master node is operational and the initial setup was completed, the worker nodes have experienced difficulties joining the cluster, which appear to be related to configuration or networking issues. To finalize the cluster setup and ensure its functionality, additional troubleshooting on the worker nodes is required to address these connectivity problems. Once these issues are resolved, the cluster will be fully functional, enabling efficient management and scaling of containerized applications across the instances.

Name: **Advik Hegde** Div/Roll no.:**D15C/15**

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:

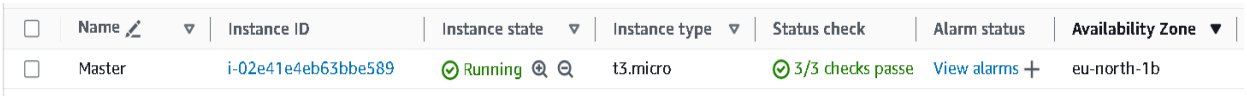
Originally developed by Google, Kubernetes is an open-source container orchestration platform designed to automate the deployment, scaling, and management of containerized applications. In fact, Kubernetes has established itself as the de facto standard for container orchestration and is the flagship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

Kubernetes Deployment

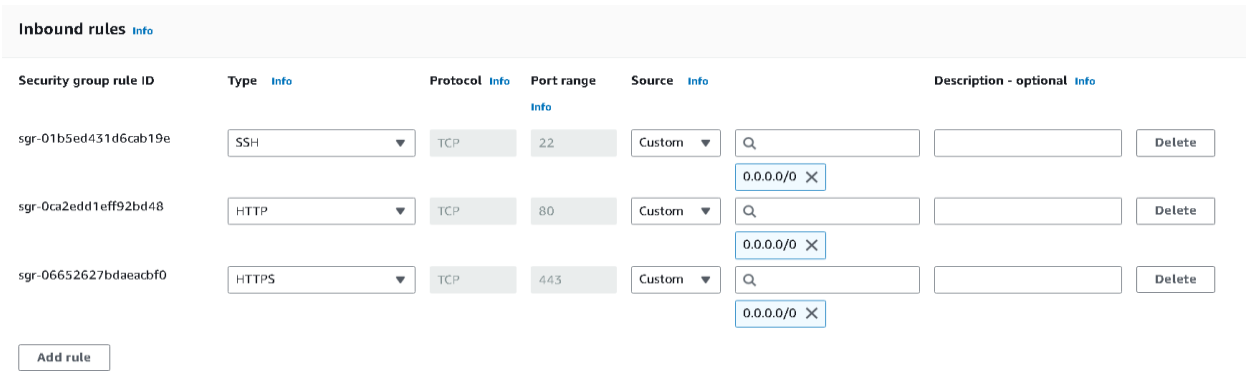
A Kubernetes Deployment is used to tell Kubernetes how to create or modify instances of the pods that hold a containerized application. Deployments can scale the number of replica pods, enable the rollout of updated code in a controlled manner, or roll back to an earlier deployment version if necessary.

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

```
sudo apt-get update
# apt-transport-https may be a dummy package; if so, you can skip that package
sudo apt-get install -y apt-transport-https ca-certificates curl gpg
# If the directory `/etc/apt/keyrings` does not exist, it should be created before the curl command, read the note below.
# sudo mkdir -p -m 755 /etc/apt/keyrings
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg # This overwrites any existing configuration in /etc/apt/sources.list.d/kubernetes.list
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl sudo apt-mark hold kubelet kubeadm kubectl
sudo systemctl enable --now kubelet
```

```
ubuntu@ip-172-31-40-255:~$ # Add Kubernetes GPG key
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -

# Add Kubernetes repository
sudo tee /etc/apt/sources.list.d/kubernetes.list <<EOF
deb https://apt.kubernetes.io/ kubernetes-xenial main
EOF

# Update package list
sudo apt-get update

# Install kubelet, kubeadm, and kubectl
sudo apt-get install -y kubelet kubeadm kubectl

# Hold the versions of Kubernetes components
sudo apt-mark hold kubelet kubeadm kubectl
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
deb https://apt.kubernetes.io/ kubernetes-xenial main
Hit:1 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Ign:6 https://packages.cloud.google.com/apt kubernetes-xenial InRelease
Err:7 https://packages.cloud.google.com/apt kubernetes-xenial Release
```

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

```
ubuntu@ip-172-31-45-229:~$ # Disable swap
sudo swapoff -a

# Allow bridging for iptables
echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf

# Apply sysctl changes
sudo sysctl -p
net.bridge.bridge-nf-call-iptables=1
net.bridge.bridge-nf-call-iptables = 1
ubuntu@ip-172-31-45-229:~$
```

6. Initialize the Kubecluster

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

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

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

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

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

  export KUBECONFIG=/etc/kubernetes/admin.conf

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

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

kubeadm join 172.31.45.229:6443 --token s9zq75.bsi7js5f62ridulc \
        --discovery-token-ca-cert-hash sha256:91eae090fdd49337bf70d5bf7478e60bc85820d0996651871129a082db6fa8f1
ubuntu@ip-172-31-45-229:~$
```

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

```
ubuntu@ip-172-31-45-229:~$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
namespace/kube-flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
```

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

Apply this deployment file using this command to create a deployment

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

```
ubuntu@ip-172-31-45-229:~$ kubectl apply -f https://k8s.io/examples/application/deployment.yaml
deployment.apps/nginx-deployment created
ubuntu@ip-172-31-45-229:~$
```

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

```
ubuntu@ip-172-31-45-229:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-krhbv    0/1     Pending   0          2m29s
nginx-deployment-d556bf558-mhlm2    0/1     Pending   0          2m29s
ubuntu@ip-172-31-45-229:~$
```

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
```

```
ubuntu@ip-172-31-45-229:~$ curl --head http://127.0.0.1:8080
HTTP/1.1 200 OK
Server: nginx/1.18.0
Date: Sat, 14 Sep 2024 7:20:53 GMT
Content-Type: text/html
Content-Length: 612
Connection: keep-alive
ETag: "5c0692e1-265"
Accept-Ranges: bytes
```

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

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

Conclusion:

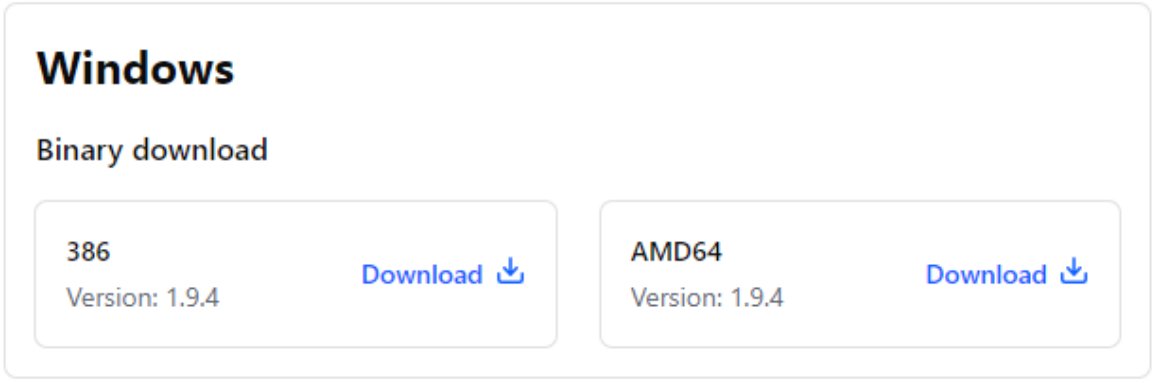
In this project, we effectively set up Kubernetes and Docker on an AWS EC2 Ubuntu instance, fine-tuned the necessary configurations, and initiated a Kubernetes cluster. We then deployed an Nginx server via a Kubernetes Deployment and utilized the Flannel networking plugin to enable pod communication. By checking the status of the pods and using port forwarding, we were able to access the Nginx server from our local environment. The successful `200 OK` response from the `curl` command verified the successful deployment. This exercise highlighted key Kubernetes functionalities, including cluster setup, application deployment, and operational validation, demonstrating Kubernetes' capability to manage and orchestrate containerized applications with efficiency.

Name:Advik Hegde Div/Roll No.:D15C/15

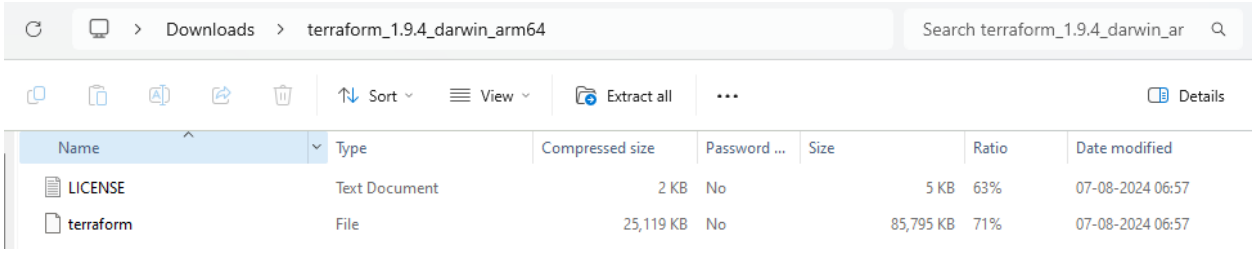
Exp 05:To Understand Terraform lifecycle,core concepts,technologies and install it on a Linux or Windows machine.

Installation and Configuration of Terraform in Windows

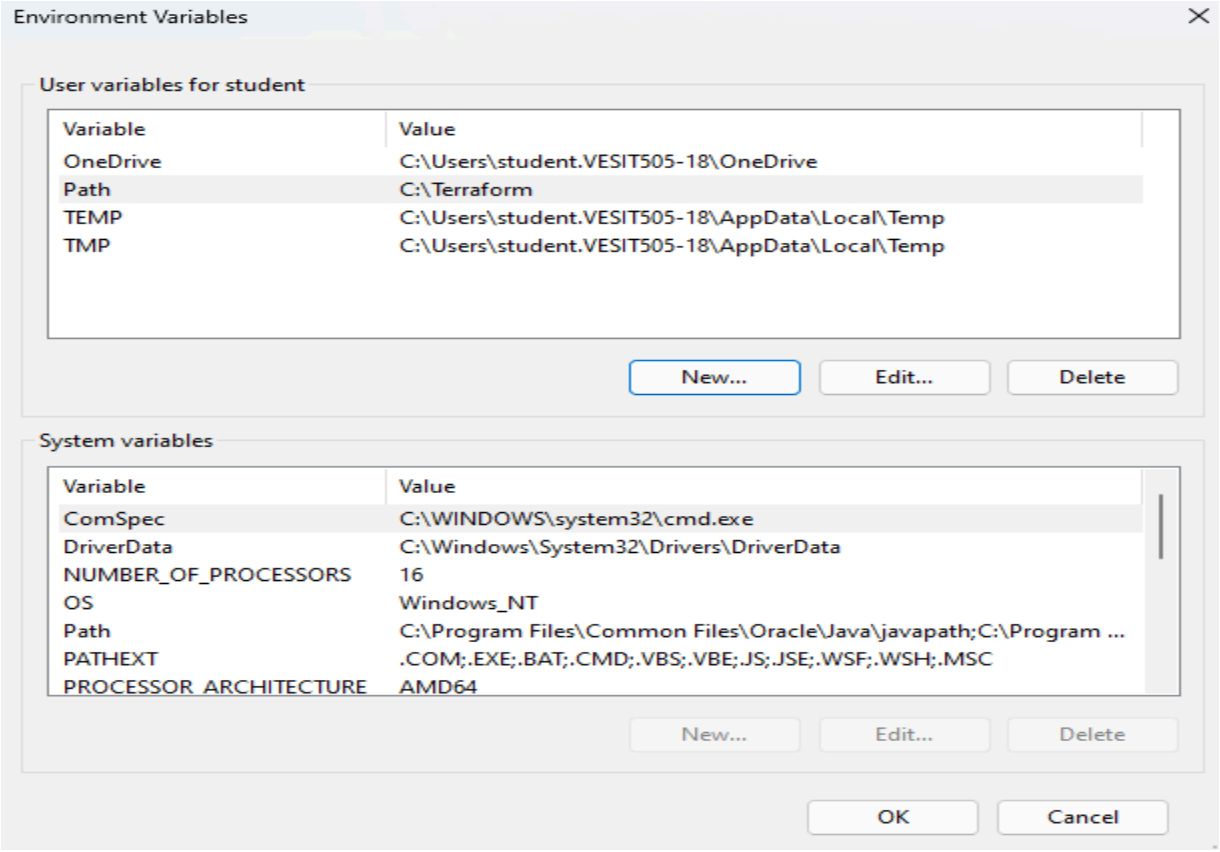
1: To install Terraform, First Download the Terraform Cli Utility for windows from terraforms official website: :https://www.terraform.io/downloads.html Select the Operating System Windows followed by either 32 bit or 64 bit based on your OS type.



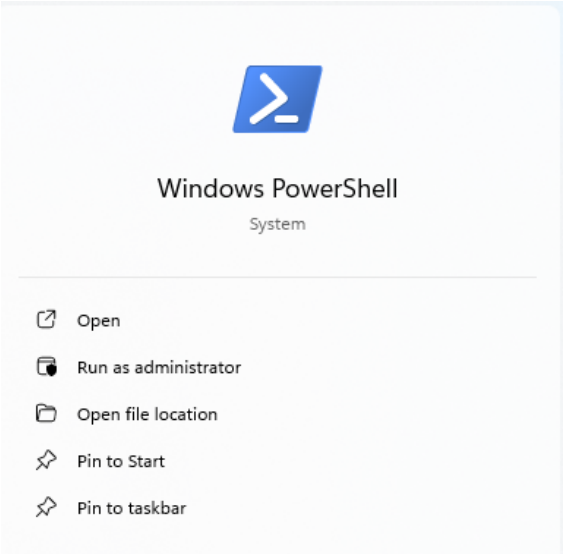
2. Extract the downloaded setup file Terraform.exe in C:\Terraformdirectory.



3: Set the System path for Terraform in Environment Variables.



4: Open PowerShell with Admin Access.



5 : Open Terraform in PowerShell and check its functionality.

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

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\student.VESIT505-18> terraform
Usage: terraform [global options] <subcommand> [args]

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

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

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

Global options (use these before the subcommand, if any):
  -chdir=DIR    Switch to a different working directory before executing the
                given subcommand.
  -help         Show this help output, or the help for a specified subcommand.
  -version      An alias for the "version" subcommand.
PS C:\Users\student.VESIT505-18>
```

Name: Advik Hegde **Div/Roll No.: D15C/15**

Experiment No : 6

Aim: To Build, change, and destroy AWS / GCP /Microsoft Azure/ DigitalOcean infrastructure using Terraform.(S3 bucket or Docker)

Implementation

A. Creating docker image using terraform

Prerequisite:

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

Step 1: Check the docker functionality

```
PS C:\Windows\system32> docker --version
Docker version 27.0.3, build 7d4bcd8
PS C:\Windows\system32> 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
```

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:

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

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

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

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

Docker > docker.tf

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

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

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

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

Step 3: Execute Terraform Init command to initialize the resources

```
PS D:\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.
PS D:\TerraformScripts\Docker>
```

\

Step 4: Execute Terraform plan to see the available resources.

```
PS D:\TerraformScripts\Docker> terraform plan

Terraform used the selected providers to generate the following execution plan.
  + 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
```

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"**

```
PS D:\TerraformScripts\Docker> terraform apply

Terraform used the selected providers to generate the following execution plan.
  + create

Terraform will perform the following actions:

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

      + healthcheck (known after apply)

      + labels (known after apply)
    }
```

Docker images, After Executing Apply step:

```
PS D:\TerraformScripts\Docker> docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
ubuntu       latest    edbfe74c41f8   2 weeks ago   78.1MB
PS D:\TerraformScripts\Docker>
```

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

```
PS D:\TerraformScripts\Docker> terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Refreshing state... [id=df0818fda9652d036abe76b261c69c8177df5c55da3b32310d6f09b66a654482]

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           = [
            "/bin/sh",
            "-c",
            "while true; do sleep 1000; done",
        ] -> 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          = "df0818fda965" -> null
        id                = "df0818fda9652d036abe76b261c69c8177df5c55da3b32310d6f09b66a654482" -> 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      = [
```

```
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=df0818fda9652d036abe76b261c69c8177df5c55da3b32310d6f09b66a654482]
docker_container.foo: Destruction complete after 1s
docker_image.ubuntu: Destroying... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_image.ubuntu: Destruction complete after 0s

Destroy complete! Resources: 2 destroyed.
```

Docker images After Executing Destroy step

```
PS D:\TerraformScripts\Docker> docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
PS D:\TerraformScripts\Docker>
```

Name : Advik Ashok Hegde **Div/Roll No. : D15C/15**

Experiment - 7 :

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

Integrating Jenkins with SonarQube (Prerequisites):

- **Jenkins installed :** To perform this experiment,it is necessary to have jenkins pre installed and up and running at some port like 8080.
- **Docker Installed (**for SonarQube) : It is also a requirement to have docker installed.

Steps to integrate Jenkins with SonarQube

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

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

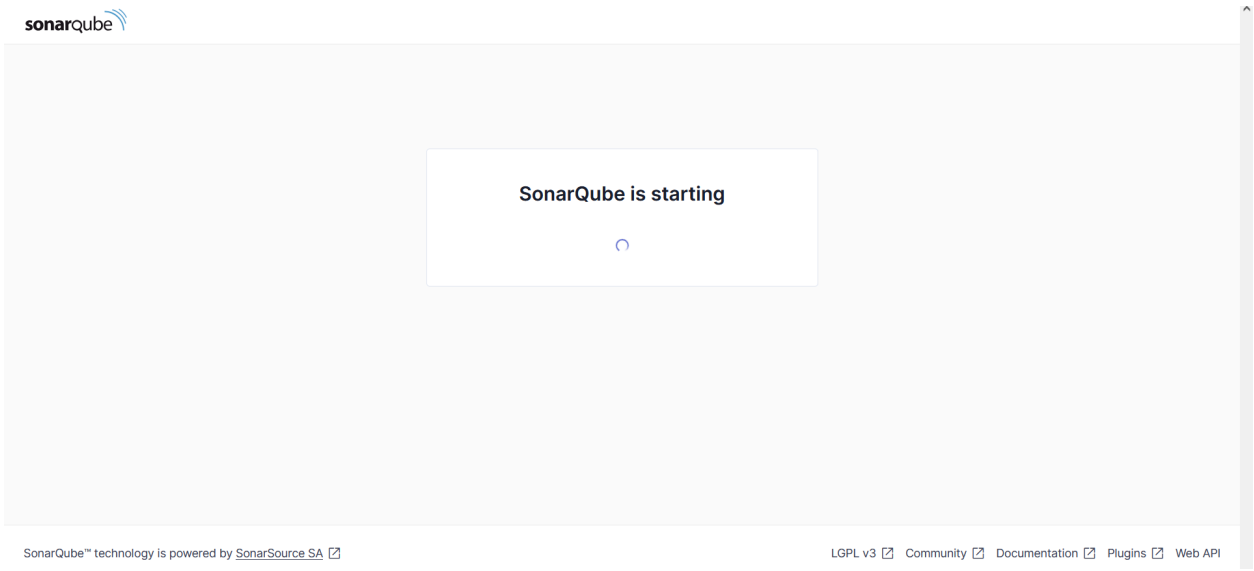
```
C:\Users\ADMIN>docker pull sonarqube
Using default tag: latest
latest: Pulling from library/sonarqube
7478e0ac0f23: Pull complete
90a925ab929a: Pull complete
7d9a34308537: Pull complete
80338217a4ab: Pull complete
1a5fd5c7e184: Pull complete
7b87d6fa783d: Pull complete
bd819c9b5ead: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:72e9feec71242af83faf65f95a40d5e3bb2822a6c3b2cda8568790f3d31aecde
Status: Downloaded newer image for sonarqube:latest
docker.io/library/sonarqube:latest

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

C:\Users\ADMIN>docker run -d --name sonarqube -p 9000:9000 sonarqube
dca969bc6baf139bf8d1d7517fd76b0ce90b18db59a88a8bf6f694a648c2d084
```

Run the given command only once and replace the name ‘sonarqube’ with any name that you prefer.In my case I have not changed the name.

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

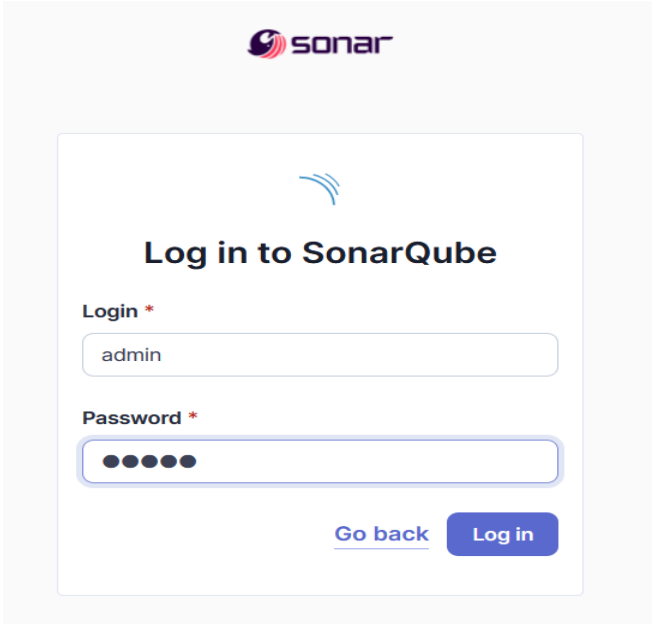
Note: Run the above command only once as once it is run,the sonarqube container is created.Running the same command again will result in an error as the container with that name will already have been created.

STEP: 3. Once the container is up and running,you can check the status ofSonarQube at localhost port 9000.

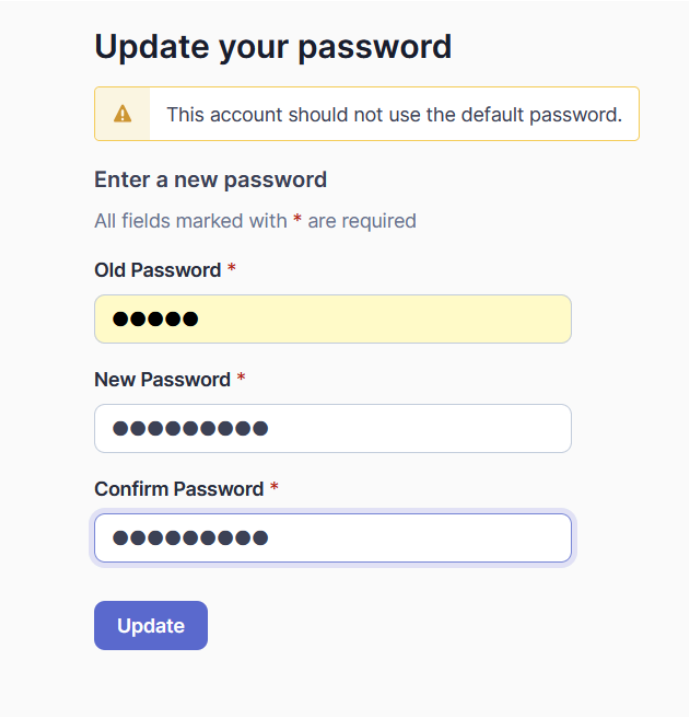


If you want to access this sonarqube dashboard again once you have closed this tab,you do not have to run the command again,rather,just open your docker and within it run the container that was created and then access the port 9000.

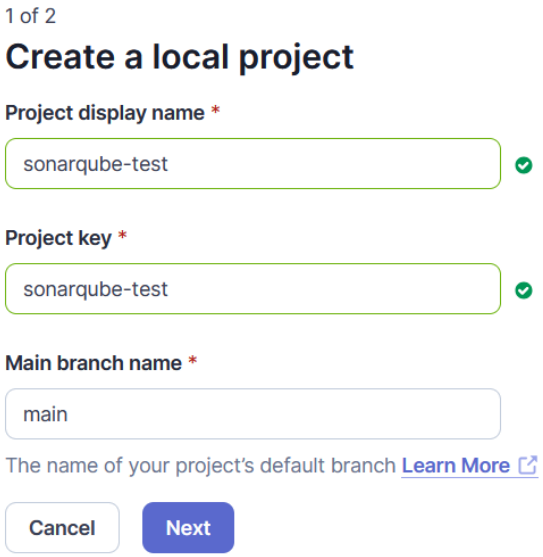
STEP: 4. Login to SonarQube using username *admin* and password *admin*.This is the default username and password for the first log-in into sonarqube.



After Logging in,you will be asked to reset the password,create a new personal password and store it for future use.

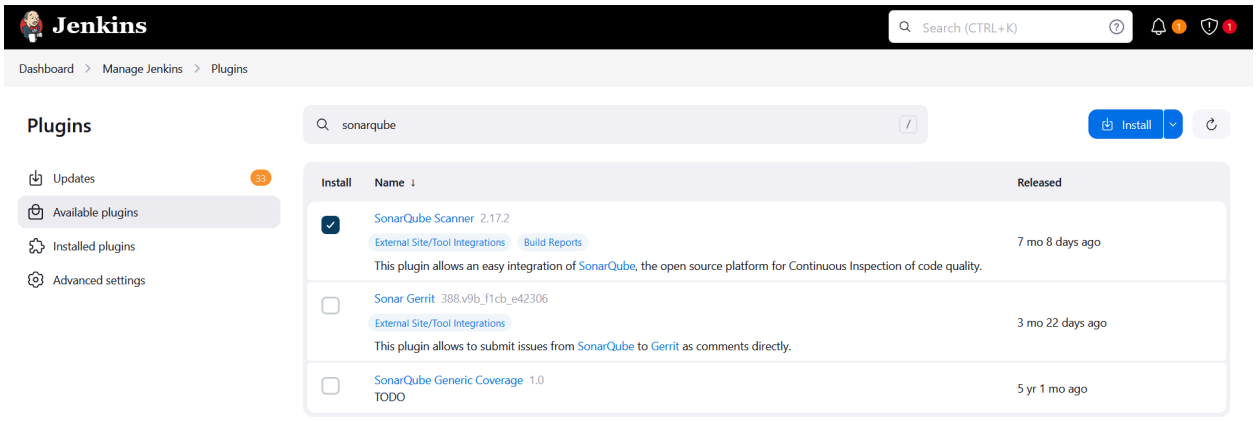


STEP:5. Create a manual project in SonarQube with the name **sonarqube-test.** Again,you can set the name as per your wish,I have decided to name the project as sonarqube-test.



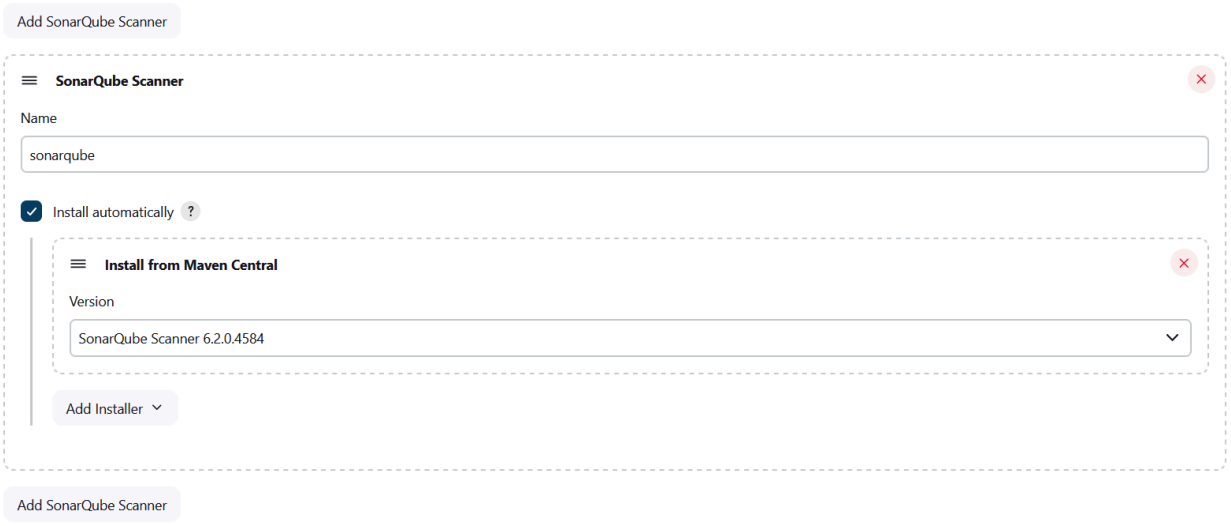
After completing the process of setting up the project,come back to the Jenkins Dashboard.

Go to Manage Jenkins and search for SonarQube Scanner for Jenkins.It is an available plugin that you can search for and then install.

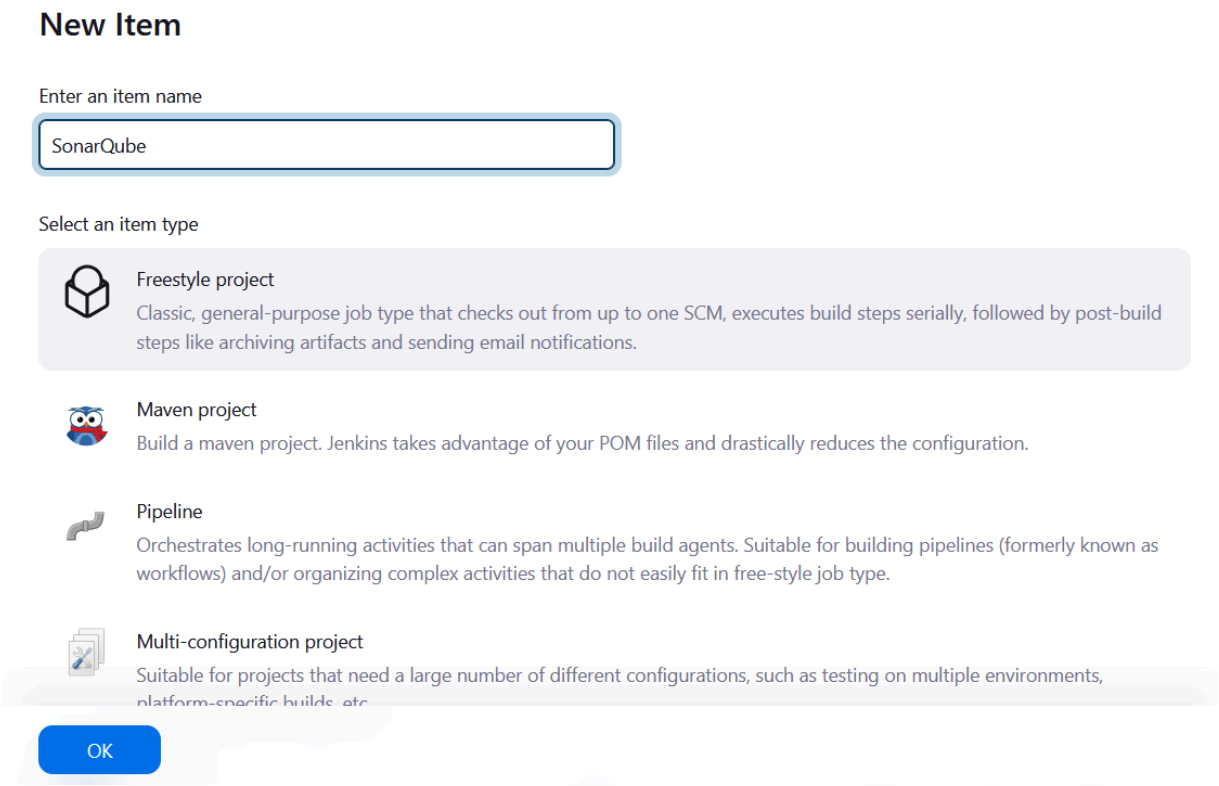


STEP : 6 . Under Jenkins 'Configure System', look for SonarQube Installations and enter the details. Provide the name for the project and enter the url based on the port that is used to run sonarqube which in my case is port **9000**.

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

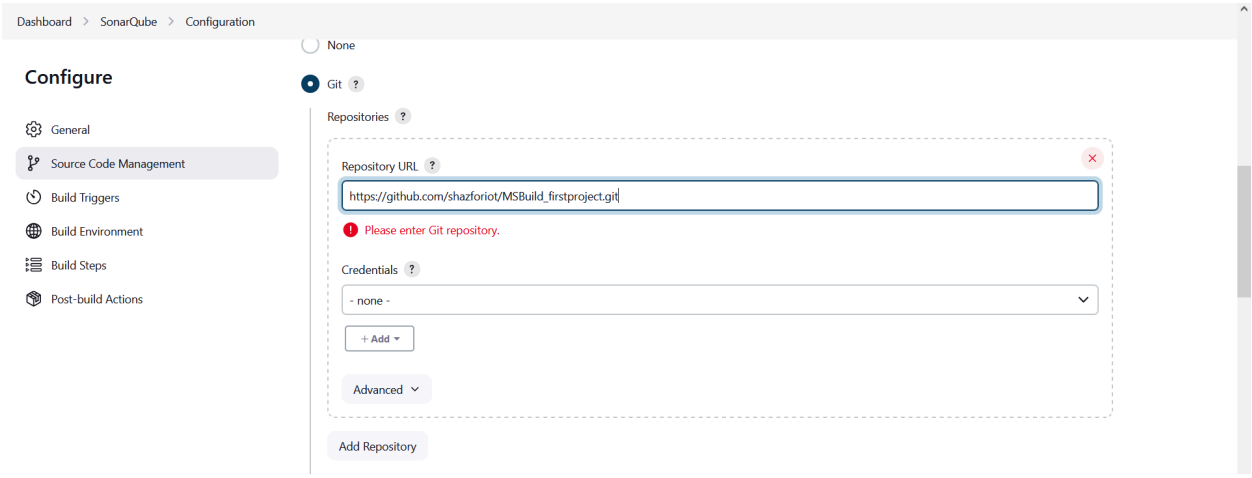


STEP : 8. After the configuration, create a New Item in Jenkins,choose a freestyle project.



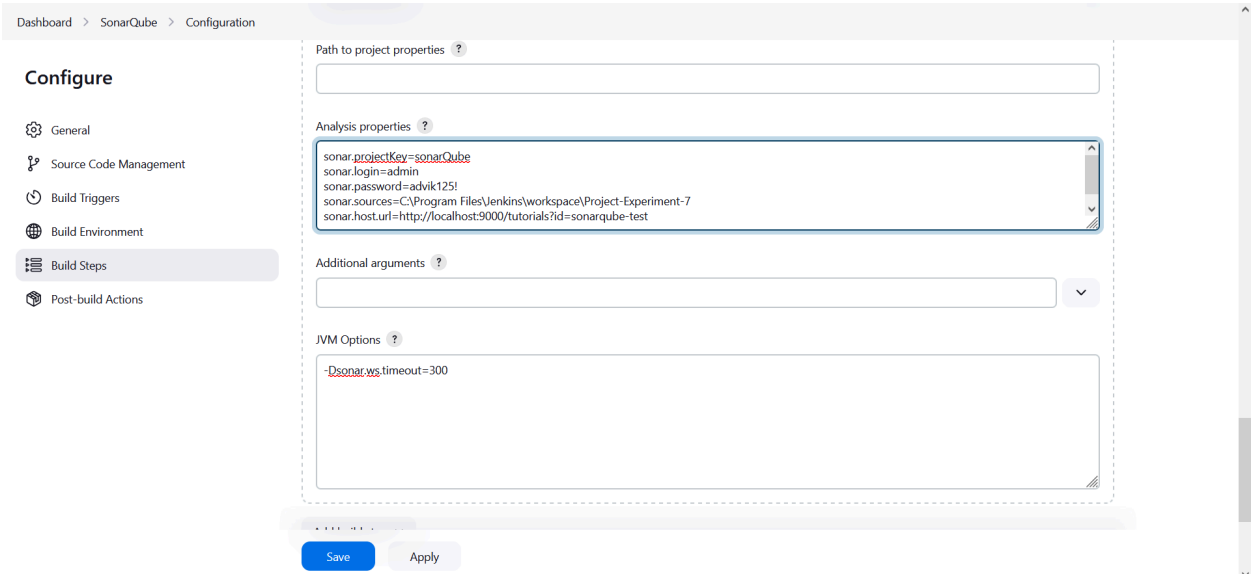
STEP : 9. Choose this GitHub repository in Source Code Management.
https://github.com/shazforiot/MSBuild_firstproject.git

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



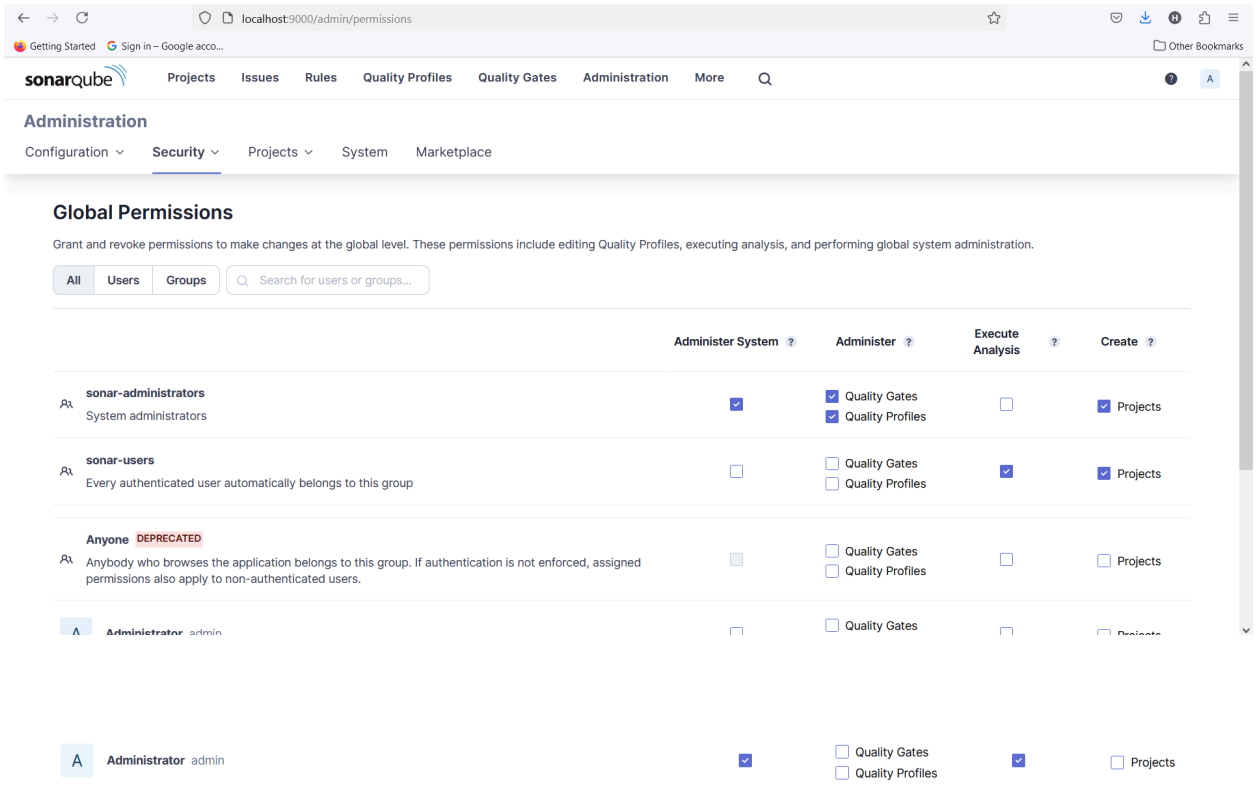
STEP : 10. Under Build-> Execute SonarQube Scanner, enter these Analysis properties. Mention the SonarQube Project Key, Login, Password, Source path and Host URL.

- Project-Key:sonarQube
- Login:admin
- Password:advik125!
- Host URL : http://localhost9000/

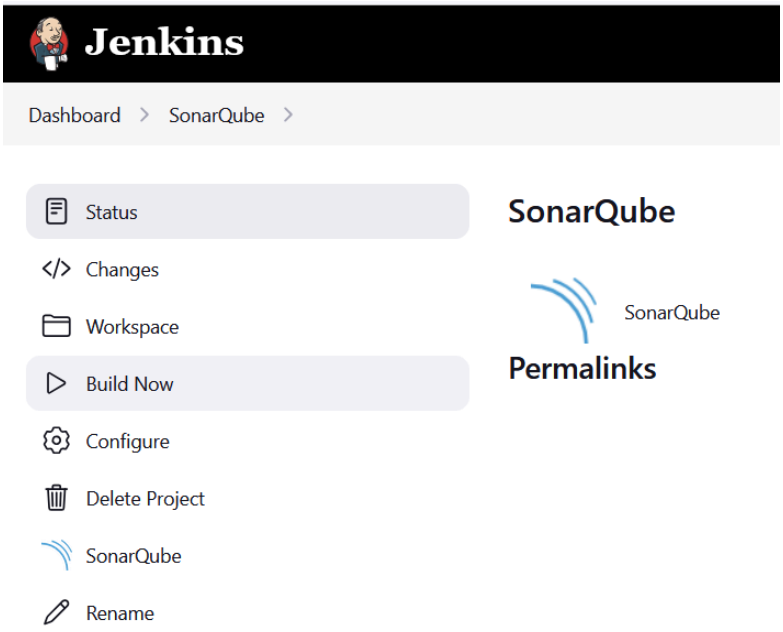


Note : Carefully enter the values for the Project Key,Login,Password,Source path and Host URL as any error will result in a **Failed Build Attempt.**

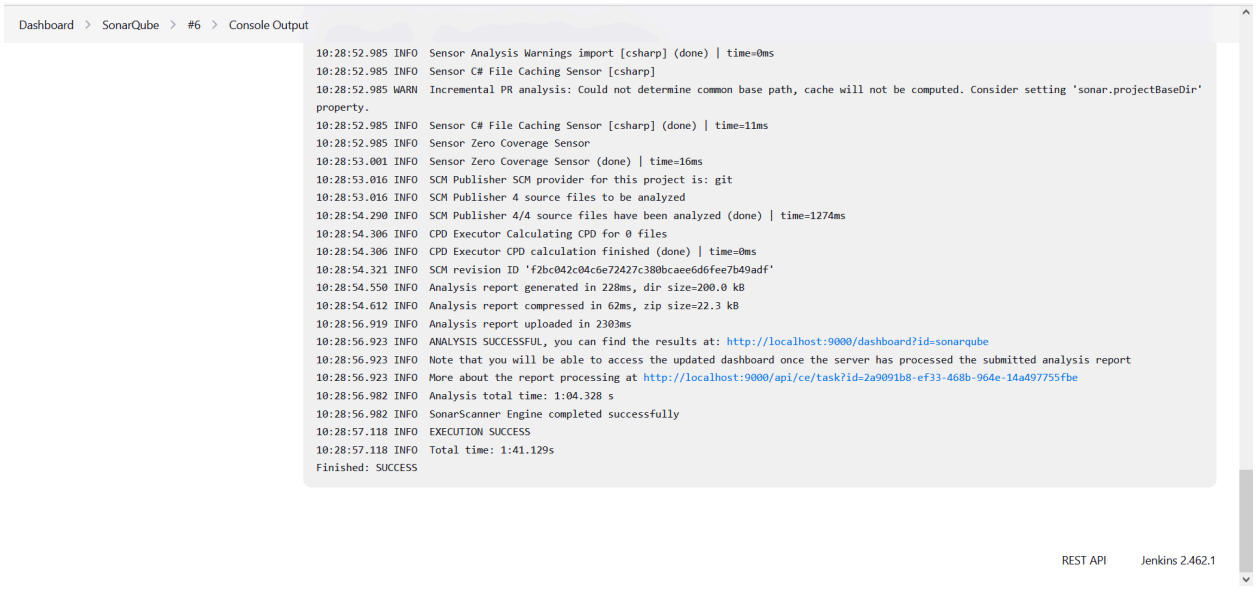
STEP : 11. Go to http://localhost:9000/<user_name>/permissions and allow Execute Permissions to the Admin user. In my case ,the user_name will be replaced with 'admin'.



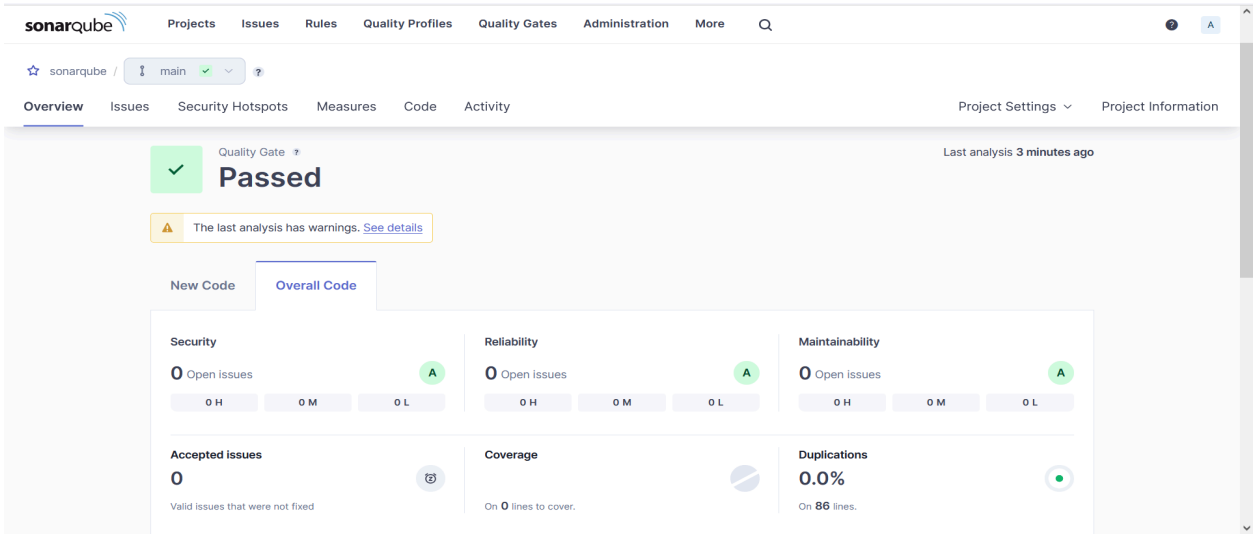
STEP : 12. Run The Build.



Once the build is complete,check the console output to determine whether the build was a success or a failure and identify the error in case of any errors.



STEP : 13. Once the build is complete and successful, check the project in SonarQube.The overview of the project should display Passed.



Conclusion : In this experiment, we integrated Jenkins with SonarQube using Docker to automate code quality analysis. SonarQube, deployed via Docker, efficiently performed static code checks, while Jenkins orchestrated the process through a pipeline that triggered analysis after every code update. This setup ensured continuous monitoring of code quality, providing immediate feedback on potential issues such as bugs and code smells. Docker simplified the management of SonarQube, making the entire process more efficient. Overall, this integration streamlined the workflow and enhanced code quality through automated analysis.

Name : Advik Ashok Hegde **Div/Roll No. : D15C/15**

Experiment - 8 :

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

Integrating Jenkins with SonarQube (Prerequisites)

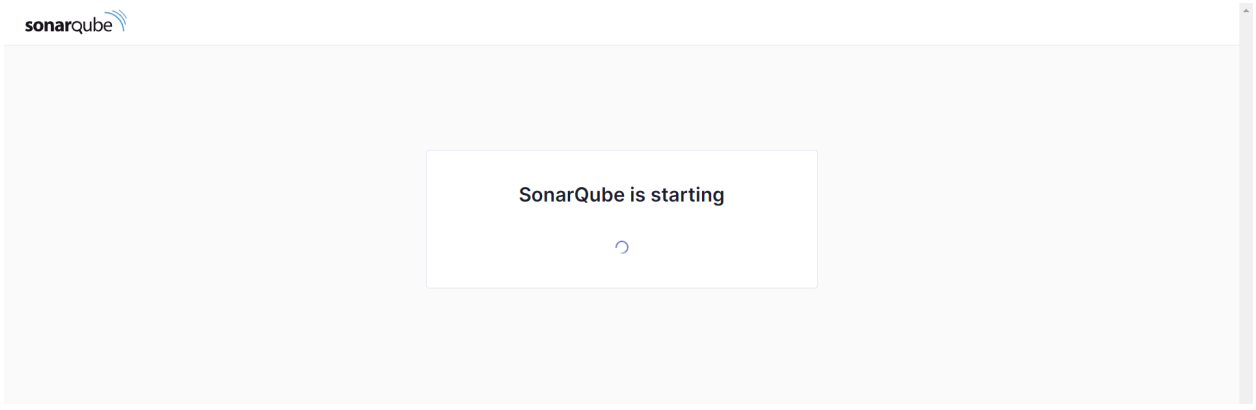
- Jenkins installed
- Docker Installed (for SonarQube)
- SonarQube Docker Image

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 -

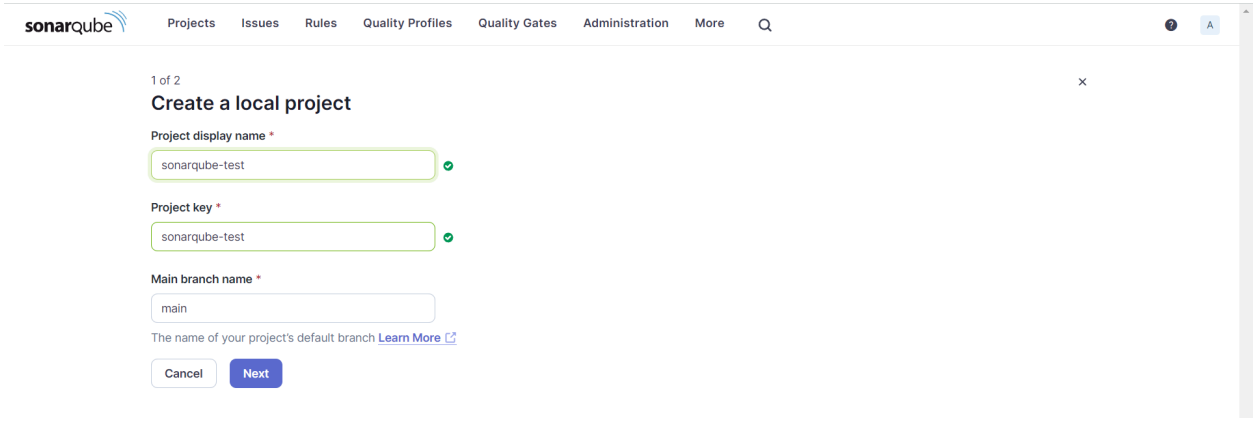
```
C:\Users\ADMIN>docker run -d --name sonarqube-2 -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
9f2e3f25d6fc6dac6175fc2feabc967ff04003b0f90b9e2a58bbc7870f093c50

C:\Users\ADMIN>
```

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 which we reset to advik125!

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

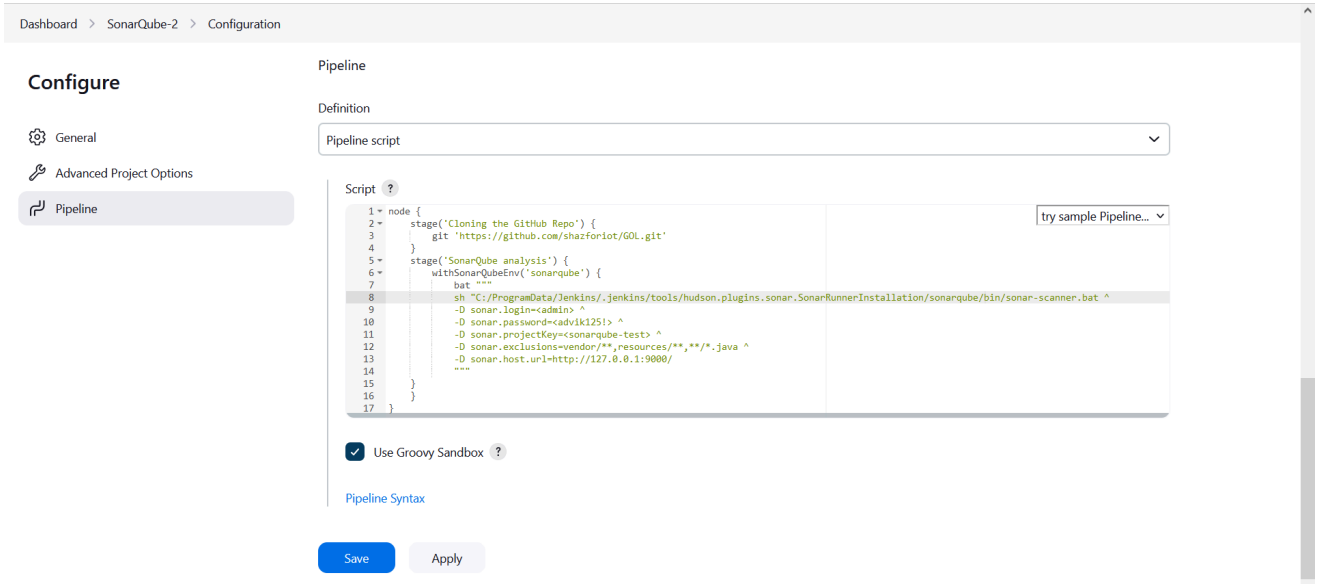


Setup the project and come back to Jenkins Dashboard.

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

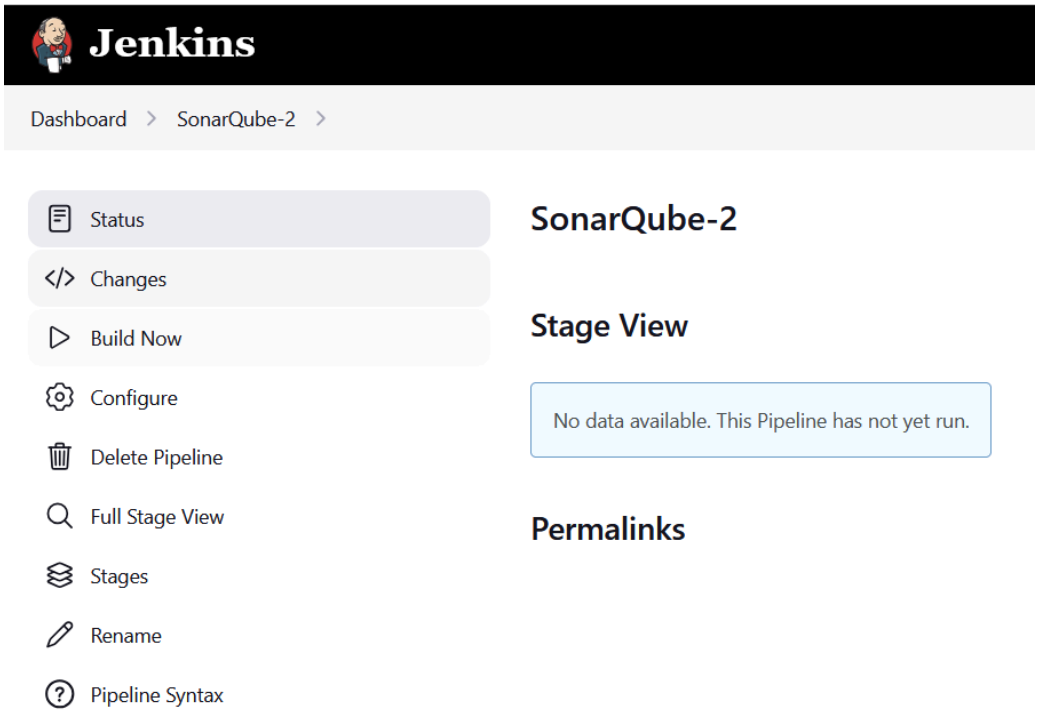
7. Under Pipeline Script, enter the following -

```
node {
        stage('Cloning the GitHub Repo') {
           git 'https://github.com/shazforiot/GOL.git'
        }
        stage('SonarQube analysis') {
          withSonarQubeEnv('sonarqube') {
              bat “””
              sh C:/ProgramData/Jenkins/.jenkins/tools/hudson.plugins.sonar.
              SonarRunnerInstallation/sonarqube/bin/sonar-scanner.bat ^
               -D sonar.login=<admin> ^
               -D sonar.password=<advik125!> ^
               -D sonar.projectKey=<sonarqube-test> ^
            -D sonar.exclusions=vendor/**,resources/**,**/*.java ^
               -D sonar.host.url=http://127.0.0.1:9000/"
               -D sonar.branch.name=main
           }
        }
    }
```

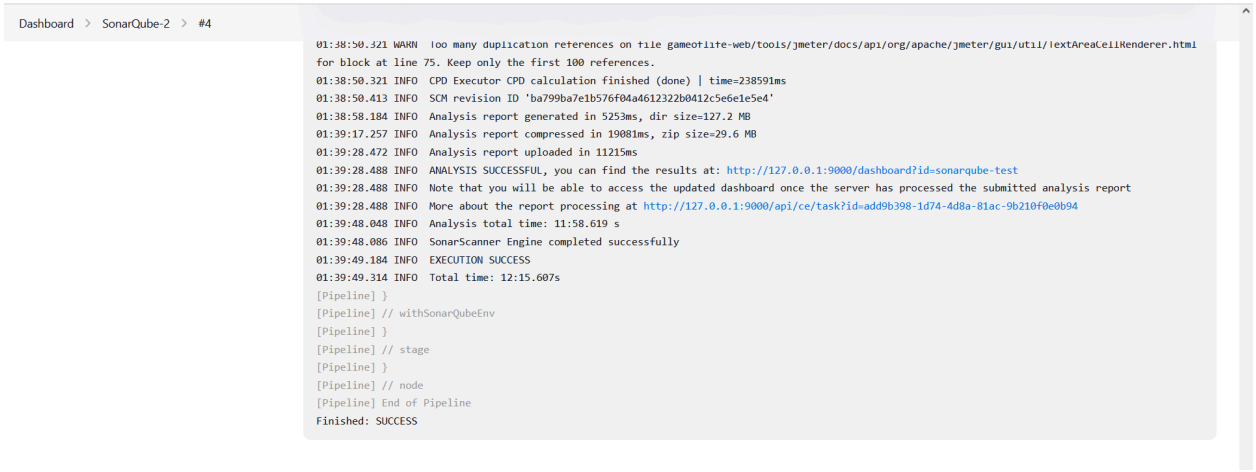


It is a java sample project which has a lot of repetitions and issues that will be detected by SonarQube.

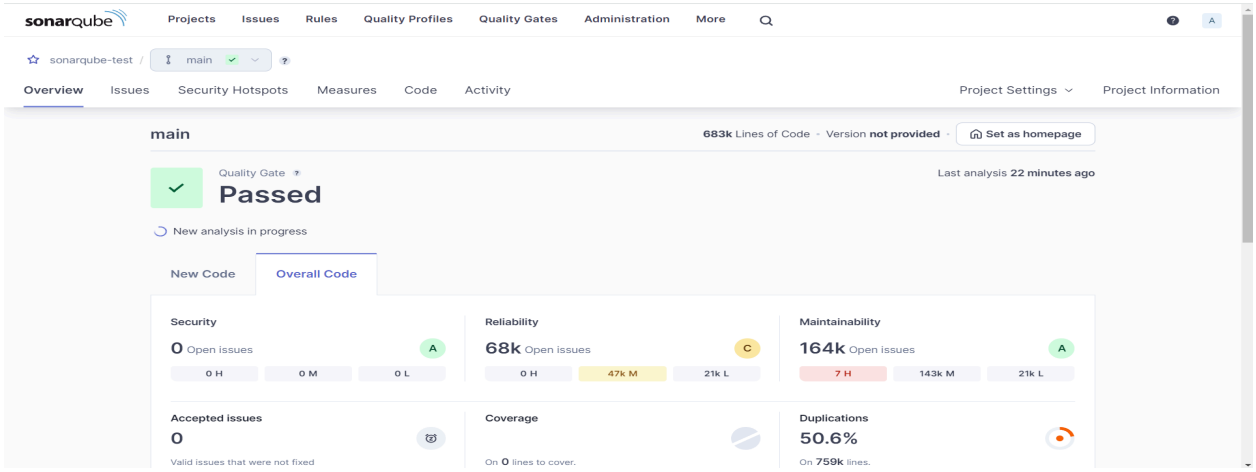
8. Run The Build.



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

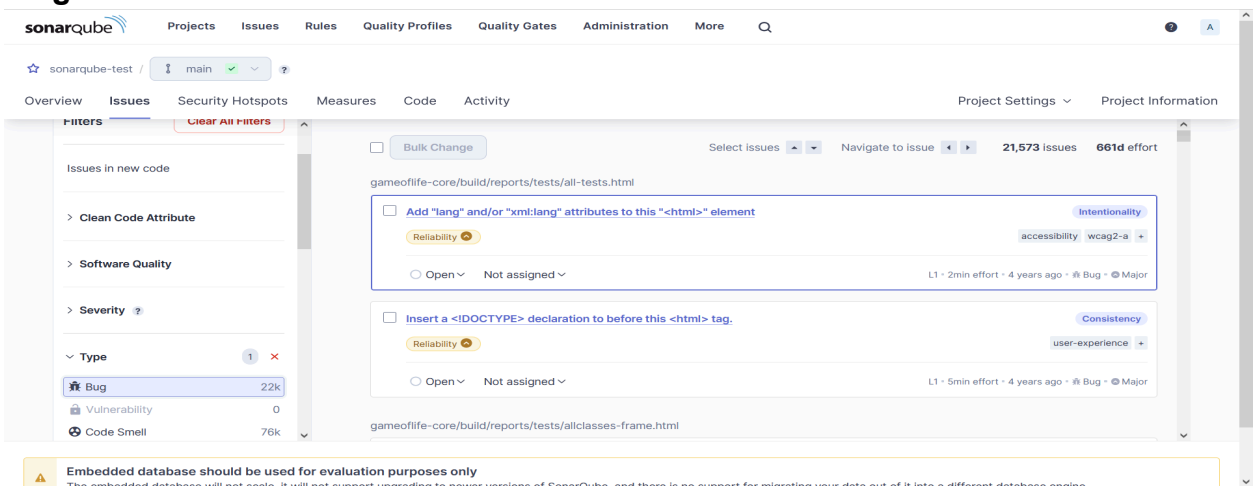


10. After that, check the project in SonarQube.

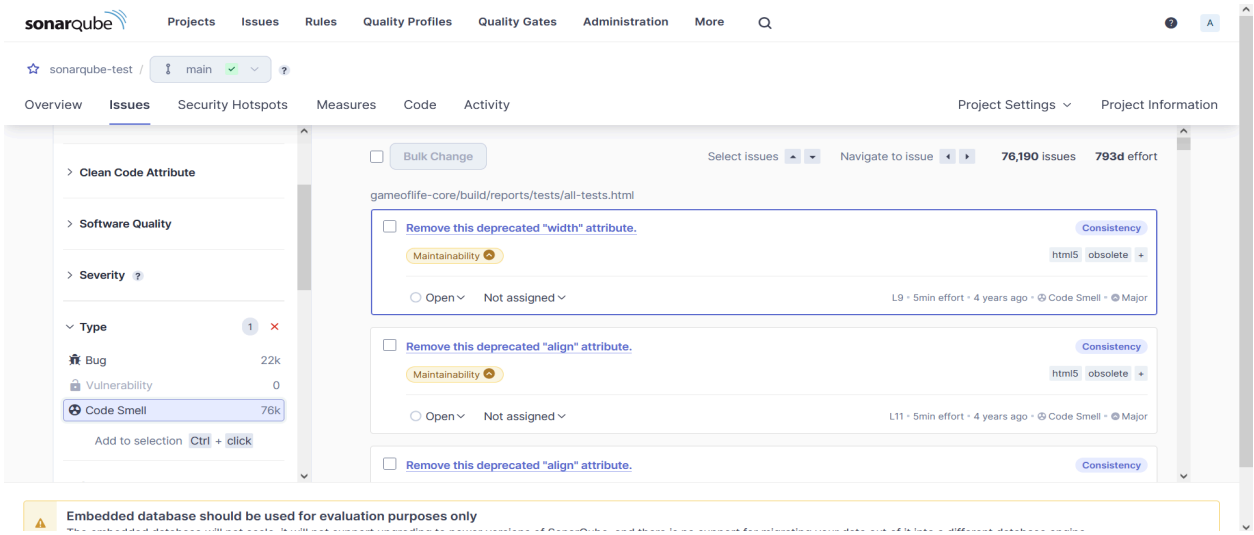


Under different tabs, check all different issues with the code.

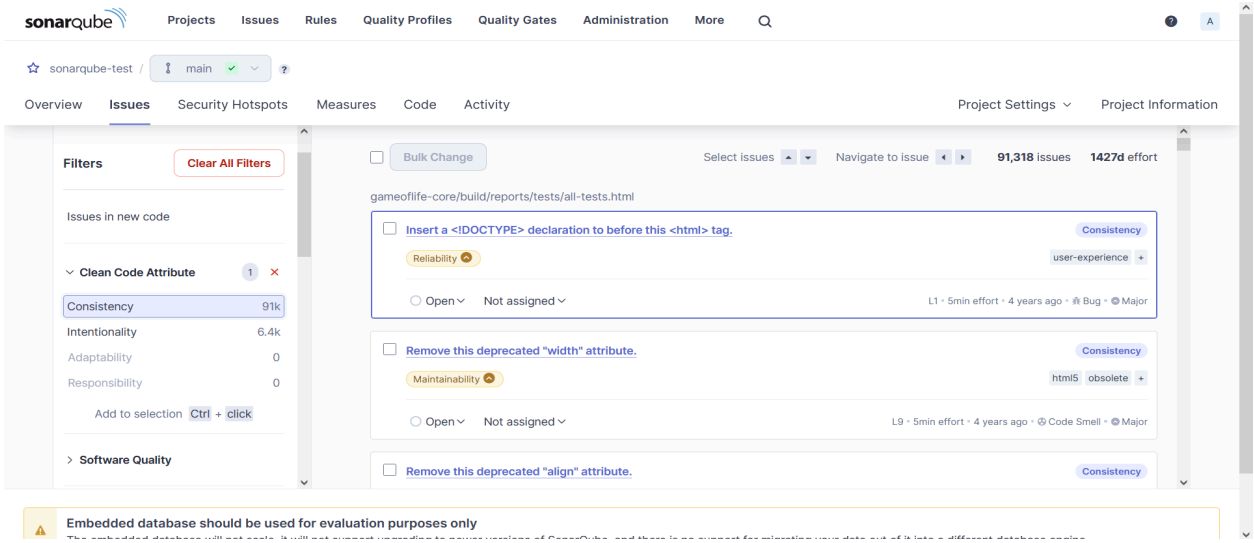
11. **Code Problems -**
Bugs :



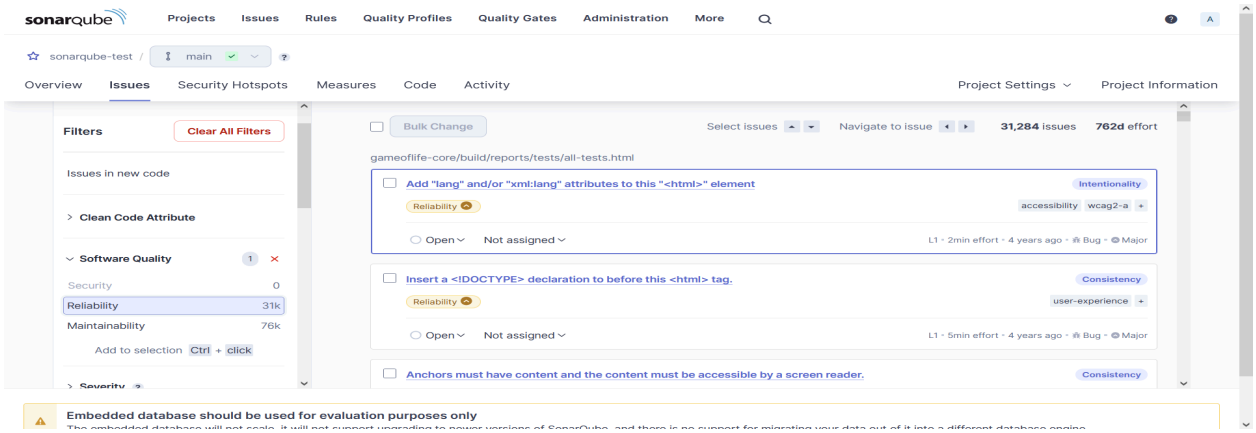
Code smells :



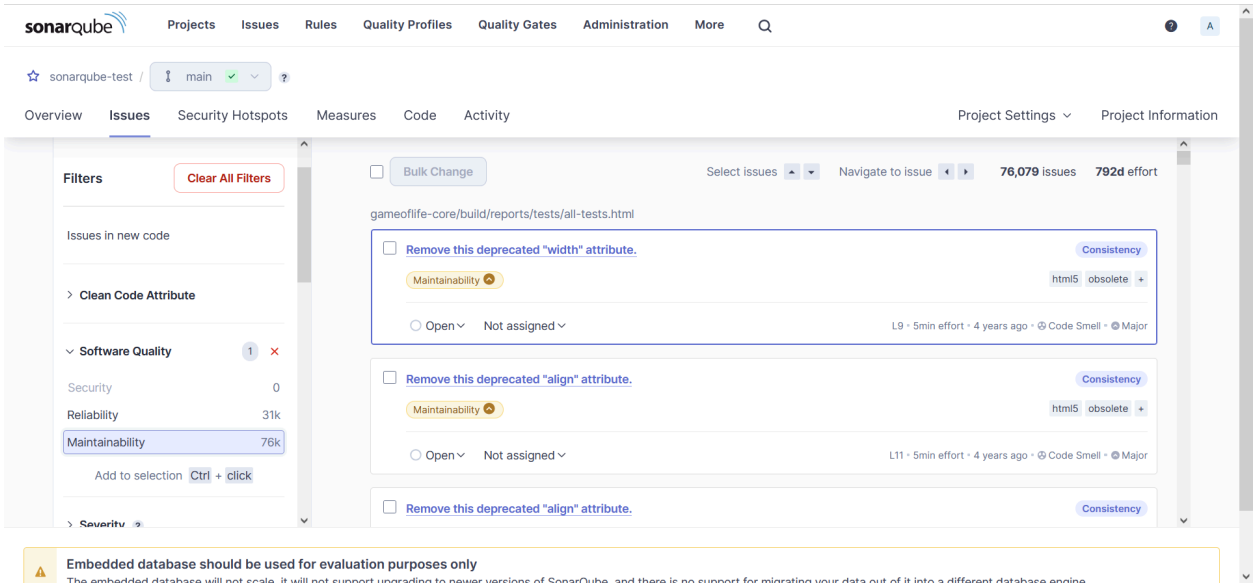
Clean Code Consistency issues:



Reliability Issues:



Maintainability Issues:



Conclusion:

In this experiment, we set up a Jenkins CI/CD pipeline integrated with SonarQube to automate static analysis on a sample application. Jenkins was configured to trigger builds and run SonarQube's analysis with every code change, detecting bugs, code smells, and security vulnerabilities. This pipeline provided continuous monitoring and ensured early detection of issues, improving code quality and security. The experiment showcased how integrating CI/CD pipelines with SonarQube enhances development efficiency and ensures better, more reliable software.

Name:Advik Hegde Div: D15C Roll No: 15

Adv DevOps Exp 09

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

Theory:

What is Nagios?

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

Why We Need Nagios tool?

Here are the important reasons to use Nagios monitoring tool:

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

Features of Nagios

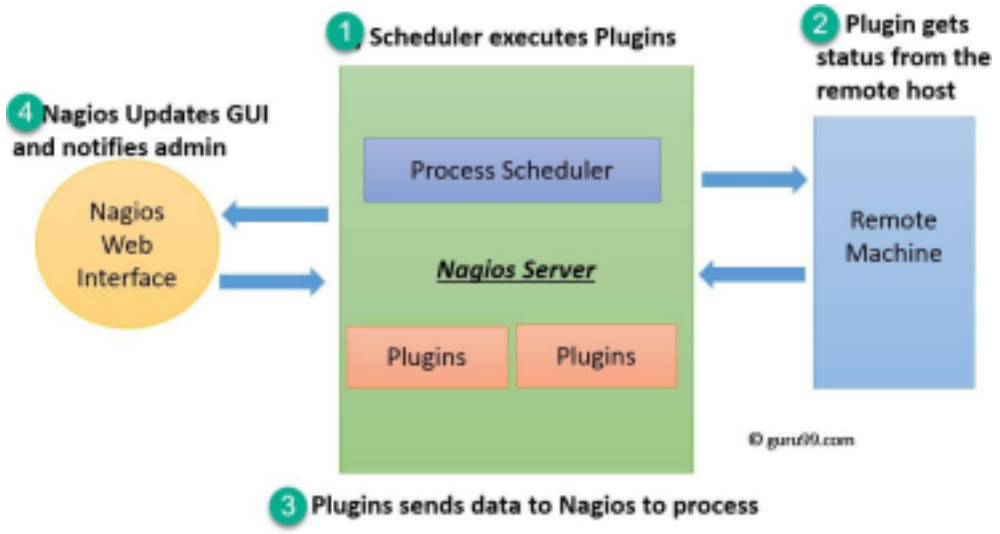
Following are the important features of Nagios monitoring tool:

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

problem resolution
- Support for implementing redundant monitoring hosts

Nagios Architecture

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

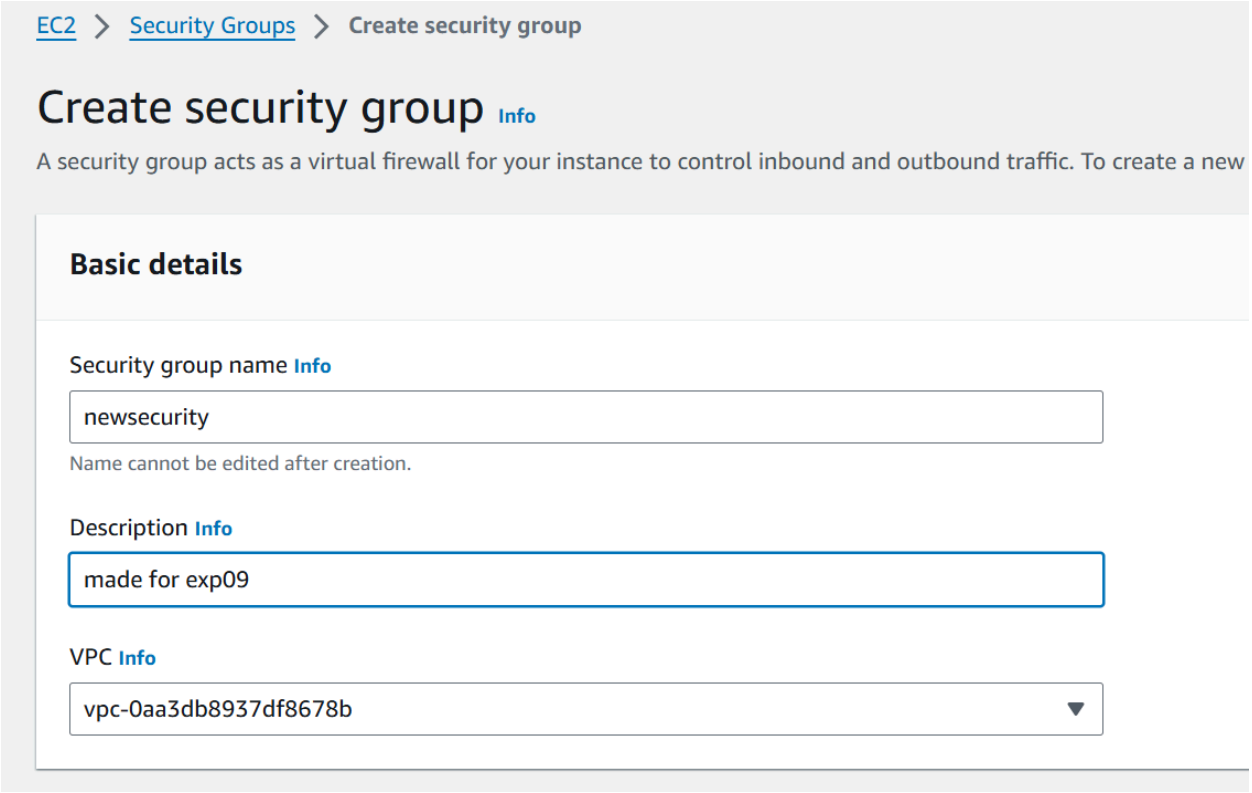


1. The scheduler is a component of the server part of Nagios. It sends a signal to execute the plugins at the remote host.
2. The plugin gets the status from the remote host
3. The plugin sends the data to the process scheduler
4. The process scheduler updates the GUI and notifications are sent to admins.

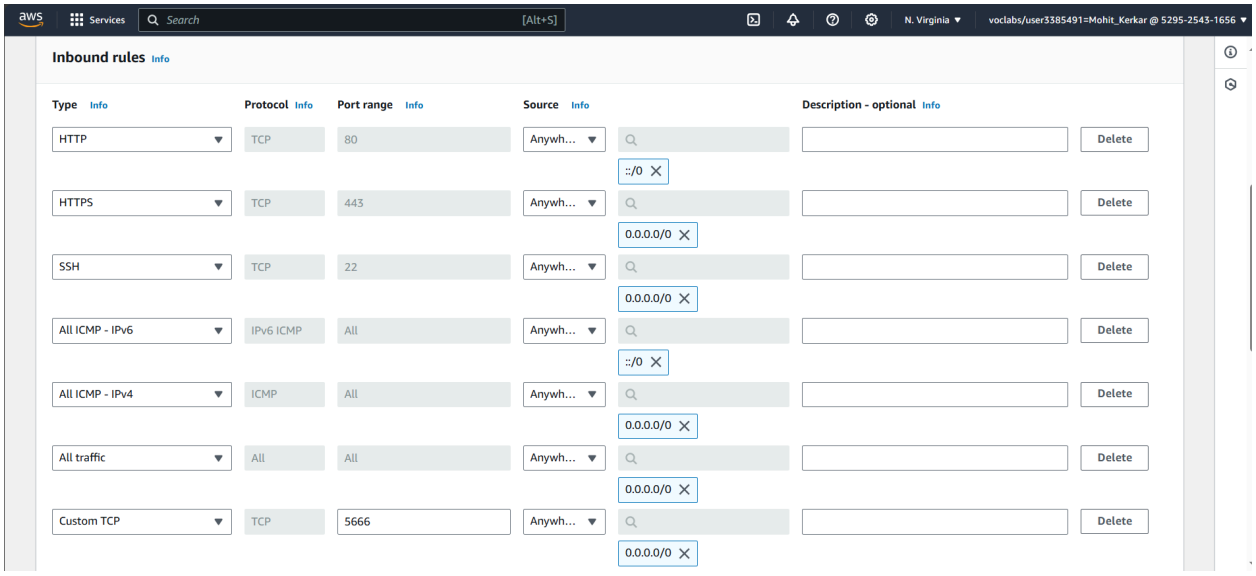
Name:Advik Hegde Div: D15C Roll No: 15

Step 1: Create a security group with the required configurations

I have created a new security group with a name ‘newsecurity’

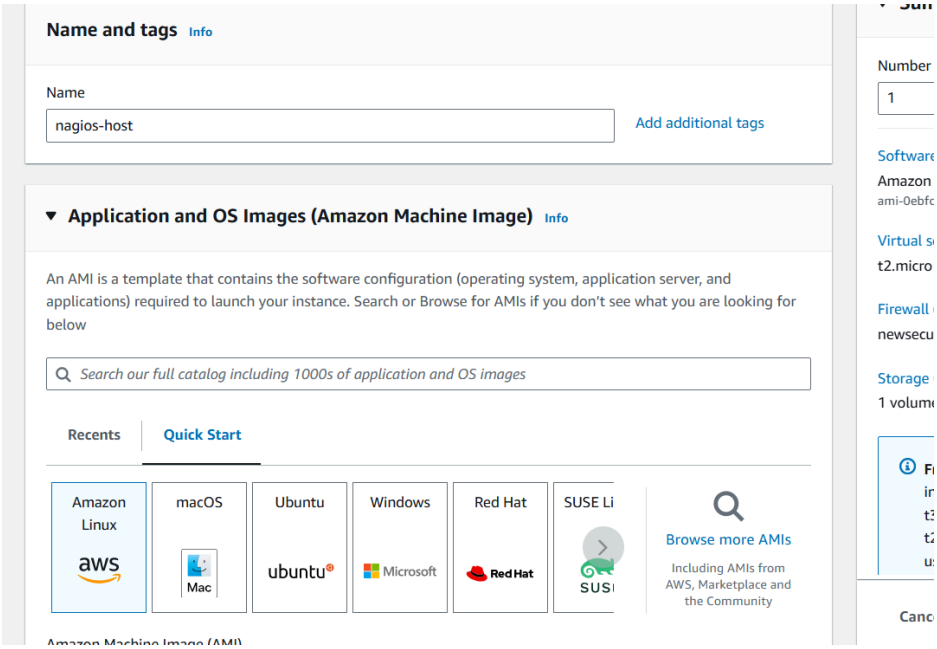


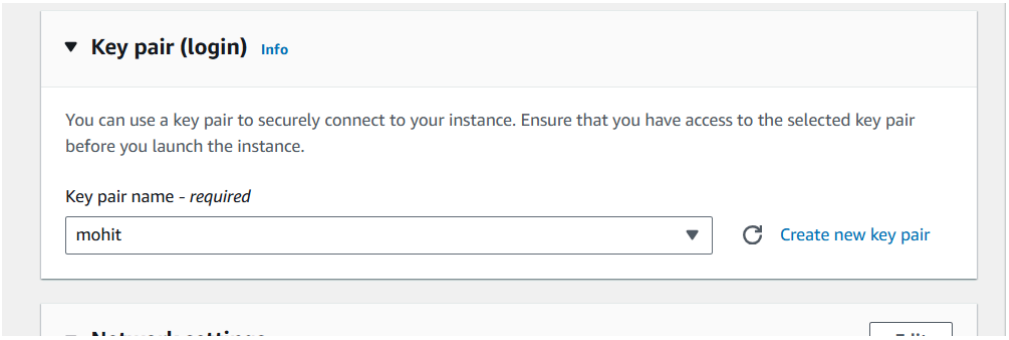
I have modified the INBOUND RULES as follows

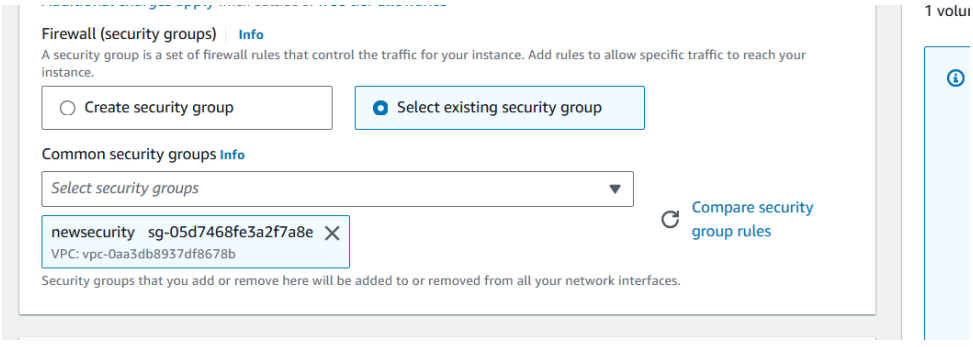


Step 2: Create ec2 instance

Name it as nagios-host. Select instance type as amazon-linux and choose the already created key pair and security group

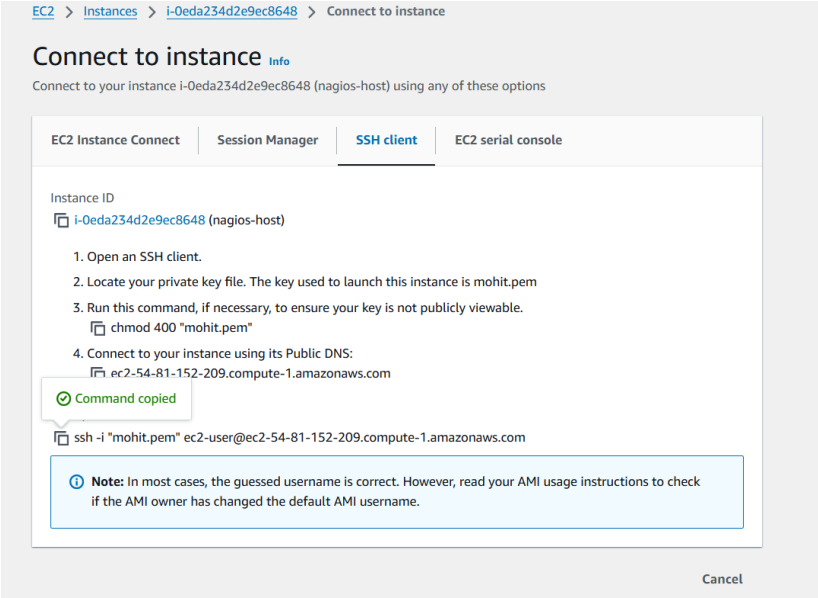




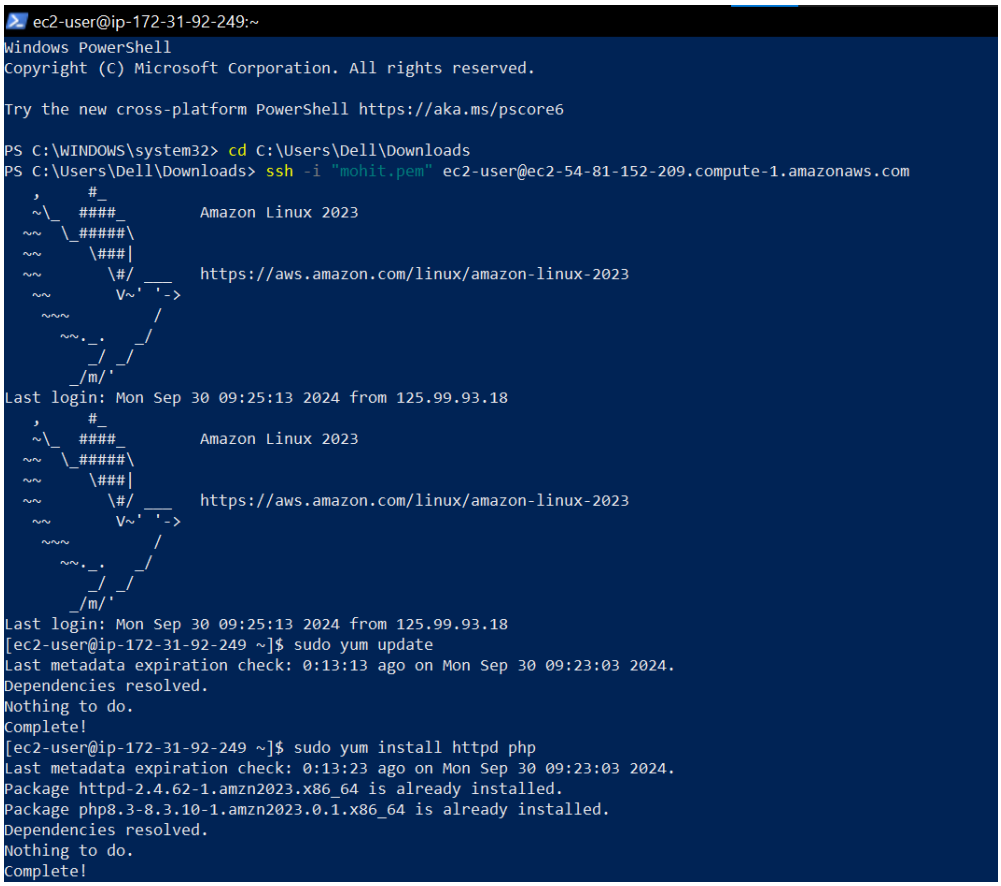


Name:Advik Hegde Div: D15C Roll No: 15

Copy the given ssh command, as we will require it for logging into our nagios-host instance from our windows powershell



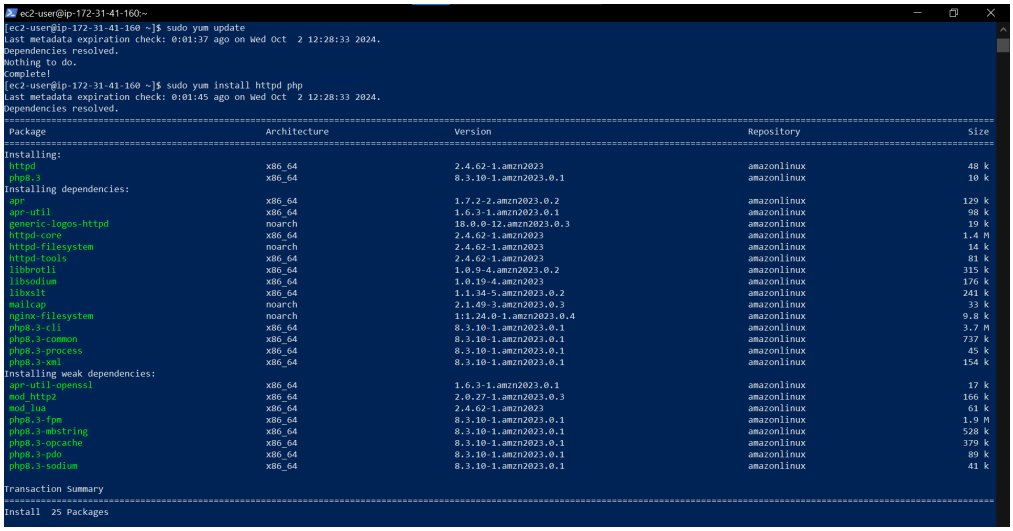
Step 3: Open an administrative powershell and remotely login using the above mentioned ssh command



And then run these commands

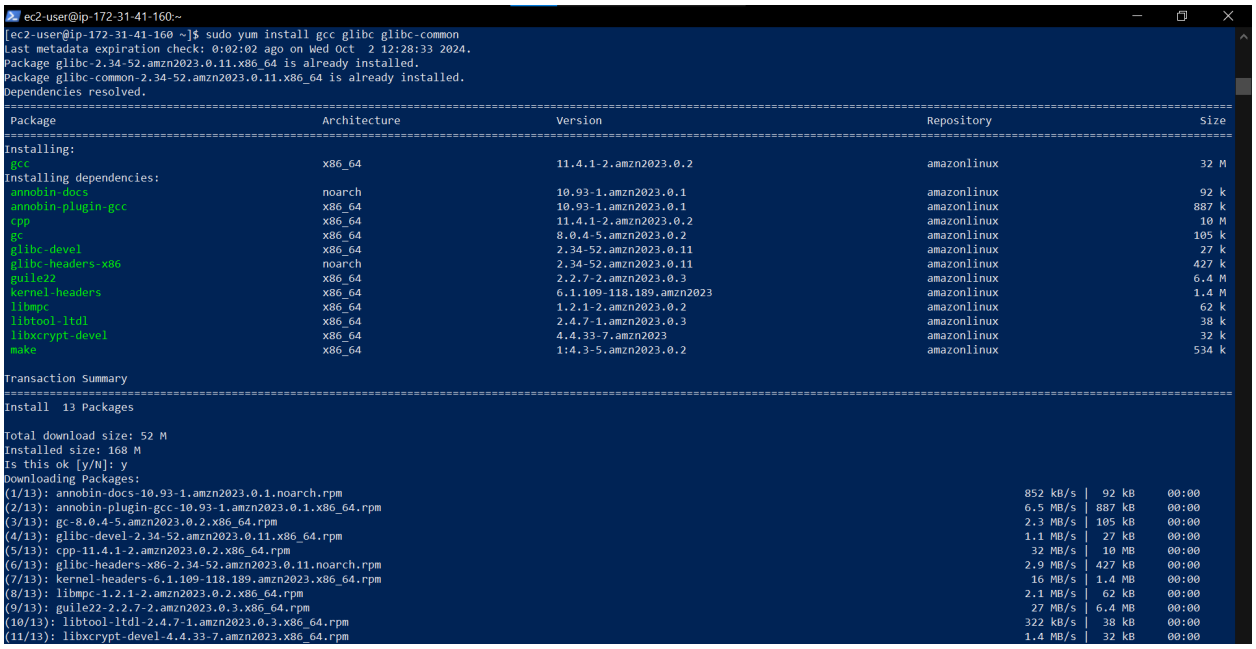
sudo yum update

sudo yum install httpd php



```
ec2-user@ip-172-31-41-160:~
[ec2-user@ip-172-31-41-160 ~]$ sudo yum update
Last metadata expiration check: 0:01:37 ago on Wed Oct  2 12:28:33 2024.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-41-160 ~]$ sudo yum install httpd php
Last metadata expiration check: 0:01:45 ago on Wed Oct  2 12:28:33 2024.
Dependencies resolved.
================================================================================
 Package                 Architecture    Version                 Repository  Size
================================================================================
Installing:
 httpd                   x86_64          2.4.62-1.amzn2023       amazonlinux 48 k
 php8.3                  x86_64          8.3.10-1.amzn2023.0.1   amazonlinux 10 k
Installing dependencies:
 apr                     x86_64          1.7.2-2.amzn2023.0.2    amazonlinux 129 k
 apr-util                x86_64          1.6.3-1.amzn2023.0.1    amazonlinux 98 k
 generic-logos-httpd     noarch          18.0.0-12.amzn2023.0.3  amazonlinux 19 k
 httpd-core              x86_64          2.4.62-1.amzn2023       amazonlinux 1.4 M
 httpd-filesystem        noarch          2.4.62-1.amzn2023       amazonlinux 14 k
 httpd-tools             x86_64          2.4.62-1.amzn2023       amazonlinux 81 k
 libbrotli               x86_64          1.0.9-4.amzn2023.0.2    amazonlinux 315 k
 libsodium               x86_64          1.0.19-4.amzn2023       amazonlinux 176 k
 libxslt                 x86_64          1.1.34-5.amzn2023.0.2   amazonlinux 241 k
 mailcap                 noarch          2.1.49-3.amzn2023.0.3   amazonlinux 33 k
 nginx-filesystem        noarch          1:1.24.0-1.amzn2023.0.4 amazonlinux 9.8 k
 php8.3-cli              x86_64          8.3.10-1.amzn2023.0.1   amazonlinux 3.7 M
 php8.3-common           x86_64          8.3.10-1.amzn2023.0.1   amazonlinux 737 k
 php8.3-process          x86_64          8.3.10-1.amzn2023.0.1   amazonlinux 45 k
 php8.3-xml              x86_64          8.3.10-1.amzn2023.0.1   amazonlinux 154 k
Installing weak dependencies:
 apr-util-openssl        x86_64          1.6.3-1.amzn2023.0.1    amazonlinux 17 k
 mod_http2               x86_64          2.0.27-1.amzn2023.0.3   amazonlinux 166 k
 mod_lua                 x86_64          2.4.62-1.amzn2023       amazonlinux 61 k
 php8.3-fpm              x86_64          8.3.10-1.amzn2023.0.1   amazonlinux 1.9 M
 php8.3-mbstring         x86_64          8.3.10-1.amzn2023.0.1   amazonlinux 528 k
 php8.3-opcache          x86_64          8.3.10-1.amzn2023.0.1   amazonlinux 379 k
 php8.3-pdo              x86_64          8.3.10-1.amzn2023.0.1   amazonlinux 89 k
 php8.3-sodium           x86_64          8.3.10-1.amzn2023.0.1   amazonlinux 41 k

Transaction Summary
================================================================================
Install  25 Packages
```

sudo yum install gcc glibc glibc-common



```
ec2-user@ip-172-31-41-160:~
[ec2-user@ip-172-31-41-160 ~]$ sudo yum install gcc glibc glibc-common
Last metadata expiration check: 0:02:02 ago on Wed Oct  2 12:28:33 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 size: 168 M
Is this ok [y/N]: y
Downloading Packages:
(1/13): annobin-docs-10.93-1.amzn2023.0.1.noarch.rpm          852 kB/s |  92 kB     00:00
(2/13): annobin-plugin-gcc-10.93-1.amzn2023.0.1.x86_64.rpm    6.5 MB/s | 887 kB     00:00
(3/13): gc-8.0.4-5.amzn2023.0.2.x86_64.rpm                    2.3 MB/s | 105 kB     00:00
(4/13): glibc-devel-2.34-52.amzn2023.0.11.x86_64.rpm          1.1 MB/s |  27 kB     00:00
(5/13): cpp-11.4.1-2.amzn2023.0.2.x86_64.rpm                   32 MB/s |  10 MB     00:00
(6/13): glibc-headers-x86-2.34-52.amzn2023.0.11.noarch.rpm    2.9 MB/s | 427 kB     00:00
(7/13): kernel-headers-6.1.109-118.189.amzn2023.x86_64.rpm     16 MB/s | 1.4 MB     00:00
(8/13): libmpc-1.2.1-2.amzn2023.0.2.x86_64.rpm                2.1 MB/s |  62 kB     00:00
(9/13): guile22-2.2.7-2.amzn2023.0.3.x86_64.rpm                27 MB/s | 6.4 MB     00:00
(10/13): libtool-ltdl-2.4.7-1.amzn2023.0.3.x86_64.rpm         322 kB/s |  38 kB     00:00
(11/13): libxcrypt-devel-4.4.33-7.amzn2023.x86_64.rpm         1.4 MB/s |  32 kB     00:00
```

sudo yum install gd gd-devel

```
ec2-user@ip-172-31-41-160:~
[ec2-user@ip-172-31-41-160 ~]$ sudo yum install gd gd-devel
Last metadata expiration check: 0:02:25 ago on Wed Oct  2 12:28:33 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
 glib2-devel             x86_64          2.74.7-689.amzn2023.0.2 amazonlinux 486 k
 google-noto-fonts-common noarch         20201206-2.amzn2023.0.2 amazonlinux  15 k
 google-noto-sans-vf-fonts noarch        20201206-2.amzn2023.0.2 amazonlinux 492 k
 graphite2               x86_64          1.3.14-7.amzn2023.0.2   amazonlinux  97 k
 graphite2-devel         x86_64          1.3.14-7.amzn2023.0.2   amazonlinux  21 k
 harfbuzz                x86_64          7.0.0-2.amzn2023.0.1    amazonlinux 868 k
 harfbuzz-devel          x86_64          7.0.0-2.amzn2023.0.1    amazonlinux 404 k
 harfbuzz-icu            x86_64          7.0.0-2.amzn2023.0.1    amazonlinux  18 k
 jbigkit-libs            x86_64          2.1-21.amzn2023.0.2     amazonlinux  54 k
 langpacks-core-font-en  noarch          3.0-21.amzn2023.0.4     amazonlinux  10 k
 libICE                  x86_64          1.0.10-6.amzn2023.0.2   amazonlinux  71 k
 libSM                   x86_64          1.2.3-8.amzn2023.0.2    amazonlinux  42 k
 libX11                  x86_64          1.7.2-3.amzn2023.0.4    amazonlinux 657 k
 libX11-common           noarch          1.7.2-3.amzn2023.0.4    amazonlinux 152 k
 libX11-devel            x86_64          1.7.2-3.amzn2023.0.4    amazonlinux 939 k
 libX11-xcb              x86_64          1.7.2-3.amzn2023.0.4    amazonlinux  12 k
 libXau                  x86_64          1.0.9-6.amzn2023.0.2    amazonlinux  31 k
 libXau-devel            x86_64          1.0.9-6.amzn2023.0.2    amazonlinux  14 k
 libXext                 x86_64          1.3.4-6.amzn2023.0.2    amazonlinux  41 k
 libXpm                  x86_64          3.5.15-2.amzn2023.0.3   amazonlinux  65 k
 libXpm-devel            x86_64          3.5.15-2.amzn2023.0.3   amazonlinux  59 k
 libXrender              x86_64          0.9.10-14.amzn2023.0.2  amazonlinux  28 k
 libXt                   x86_64          1.2.0-4.amzn2023.0.2    amazonlinux 181 k
 libblkid-devel          x86_64          2.37.4-1.amzn2023.0.4   amazonlinux  15 k
```

Create a new Nagios User with its password. You'll have to enter the password twice for confirmation.

sudo adduser -m nagios

sudo passwd nagios

```
[ec2-user@ip-172-31-41-160 ~]$ sudo adduser -m nagios
[ec2-user@ip-172-31-41-160 ~]$ sudo passwd nagios
Changing password for user nagios.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[ec2-user@ip-172-31-41-160 ~]$
```

Create a new user group & create a new directory for Nagios downloads using the following commands

sudo groupadd nagcmd

sudo usermod -a -G nagcmd nagios

sudo usermod -a -G nagcmd apache

mkdir ~/downloads

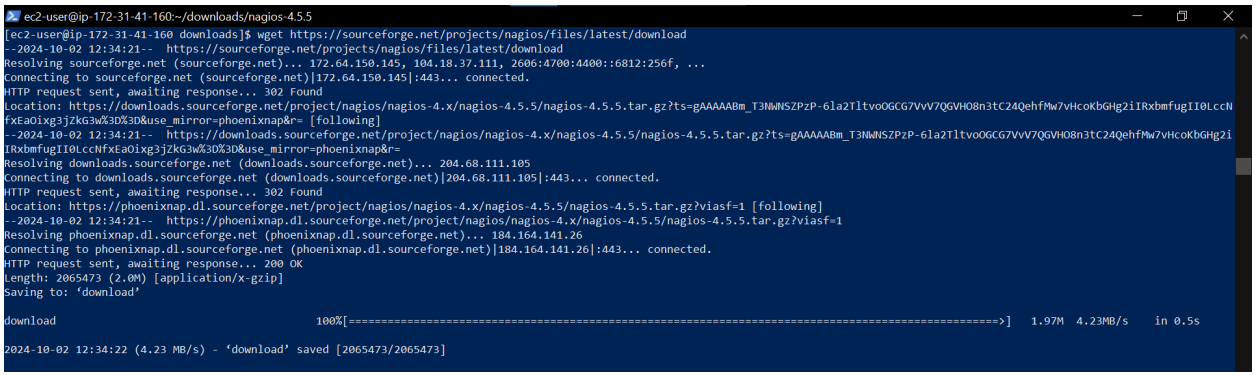
cd ~/downloads

Use **wget** to download the source zip files.

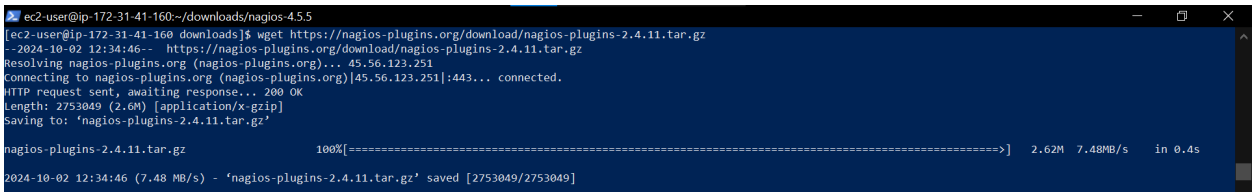
In this step, we are downloading, the latest version of nagios and the necessary plugins required to carry out the tasks of setting up a nagios server

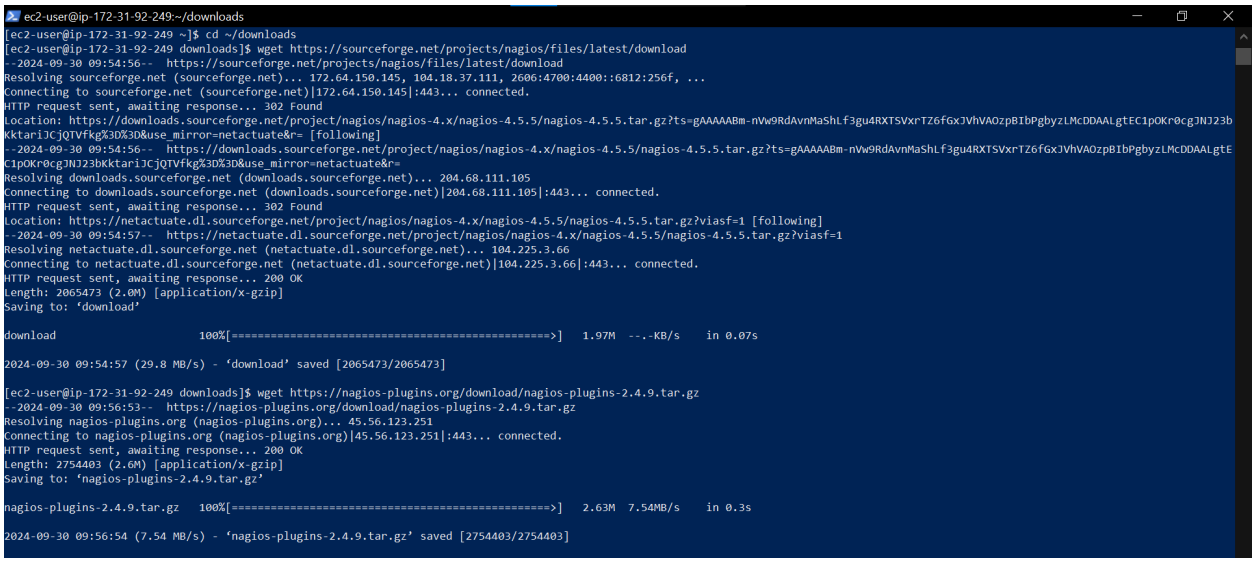
Name:Advik Hegde Div: D15C Roll No: 15

wget https://sourceforge.net/projects/nagios/files/latest/download



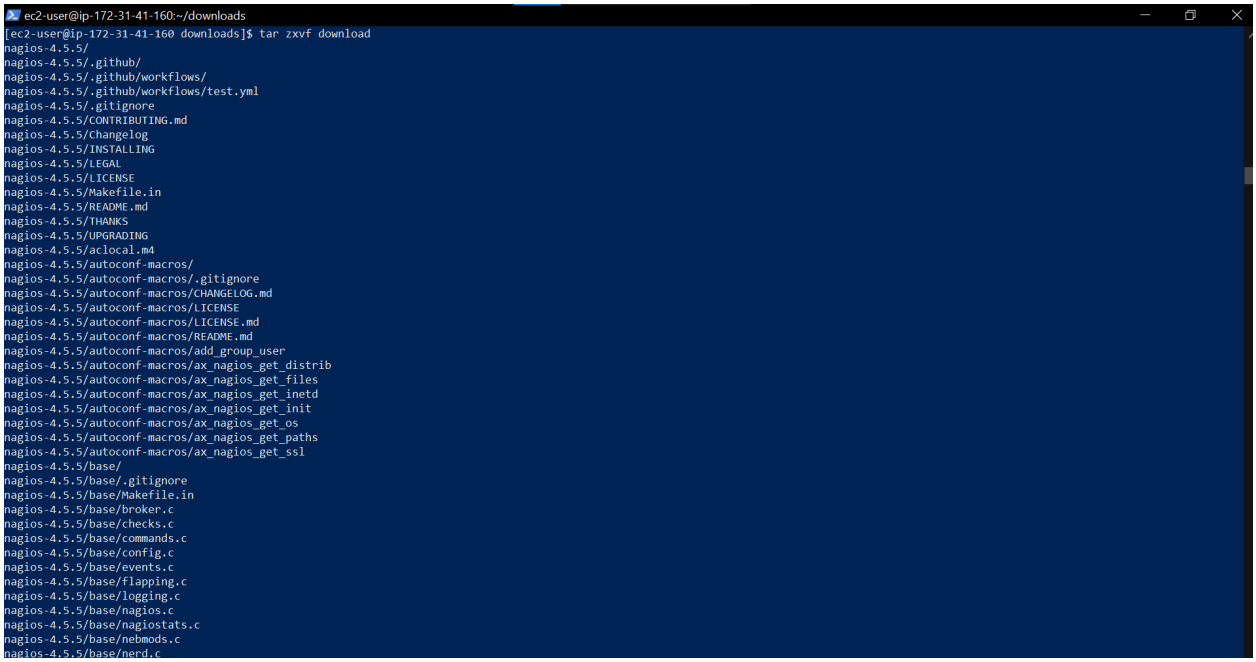
wget https://nagios-plugins.org/download/nagios-plugins-2.4.11.tar.gz





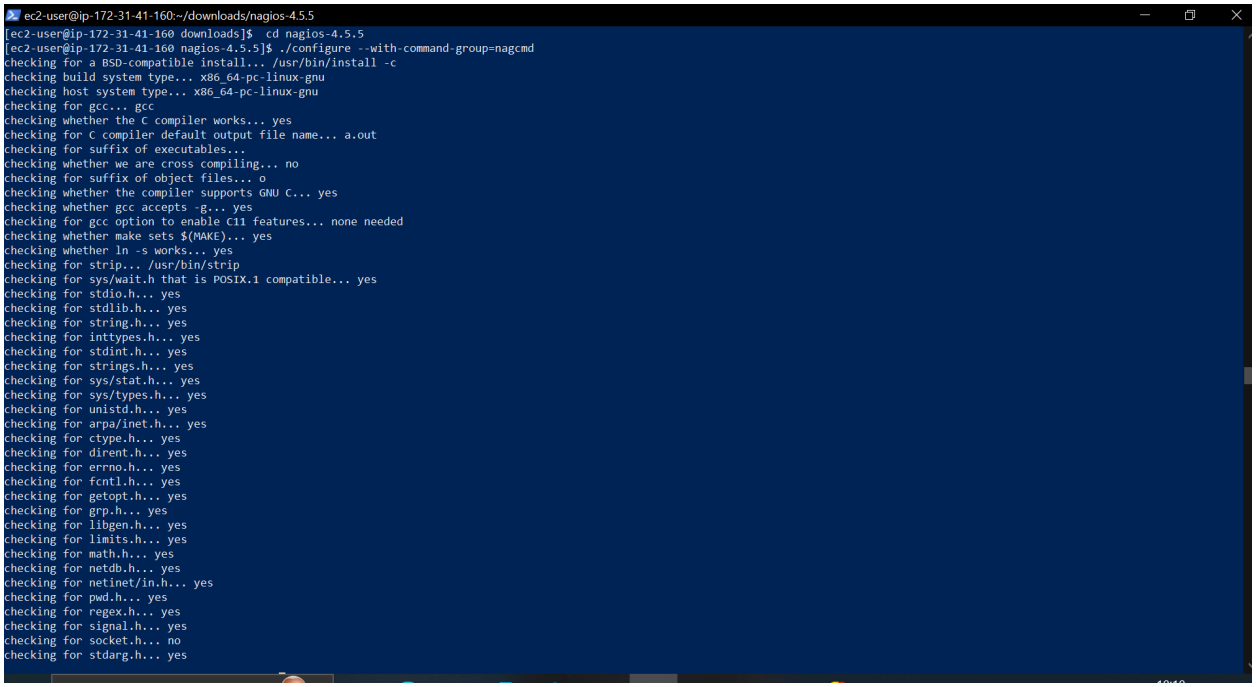
Now, we run the next command in the following manner
tar zxvf <nagios-4.5.5 version> (for me it has gotten saved as 'download')
So i wrote **tar zxvf download**

Name:Advik Hegde Div: D15C Roll No: 15



After which we are supposed to **change our directory** over there
For eg. **cd nagios-4.5.5**… depending on the version that we have downloaded

Next, Run this command (make sure that you are working inside nagios-4.x.x directory)
./configure --with-command-group=nagcmd



```
checking for strtoul... yes
checking for unsetenv... yes
checking for type of socket size... size_t
checking for Kerberos include files... configure: WARNING: could not find include files
checking for pkg-config... pkg-config
checking for SSL headers... configure: error: Cannot find ssl headers
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$
```

After running this command, we get an **error related to ssl header being absent**

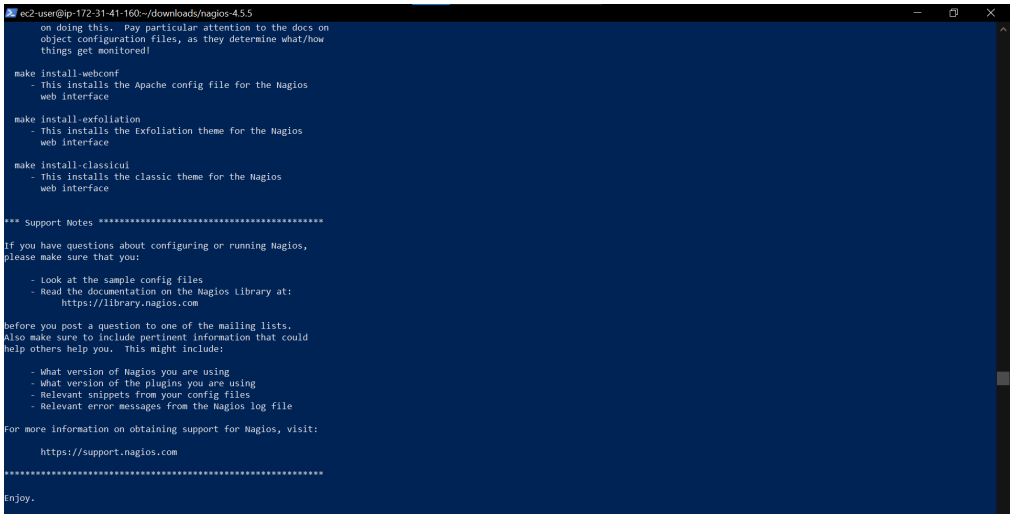
For that purpose, we are to run the following command.
sudo yum install openssl-devel (for ssl header)

```
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ sudo yum install openssl-devel
Last metadata expiration check: 0:12:11 ago on Wed Oct  2 12:28:33 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                                 26 MB/s | 3.0 MB     00:00
----------------------------------------------------------------------------------------------------------------
Total                                                                          17 MB/s | 3.0 MB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                                                       1/1
  Installing       : openssl-devel-1:3.0.8-1.amzn2023.0.14.x86_64                                           1/1
  Running scriptlet: openssl-devel-1:3.0.8-1.amzn2023.0.14.x86_64                                           1/1
  Verifying        : openssl-devel-1:3.0.8-1.amzn2023.0.14.x86_64                                           1/1

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

Complete!
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$
```

Now, Re-run **./configure --with-command-group=nagcmd**
After this, run **make all** command

```
ec2-user@ip-172-31-41-160:~/downloads/nagios-4.5.5
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ make all
cd ./base && make
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/base'
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o nagios.o ./nagios.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o broker.o broker.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o nebmods.o nebmods.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o ../common/shared.o ../common/shared.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o query-handler.o query-handler.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o workers.o workers.c
In function ‘get_wproc_list’,
    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 ../common/macros.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o netutils.o netutils.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o notifications.o notifications.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o sehandlers.o sehandlers.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o utils.o utils.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o retention-base.o ./sretention.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o xretention-base.o ../xdata/xrddefault.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o comments-base.o ../common/comments.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o xcomments-base.o ../xdata/xcddefault.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o objects-base.o ../common/objects.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o xobjects-base.o ../xdata/xodtemplate.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o statusdata-base.o ../common/statusdata.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o xstatusdata-base.o ../xdata/xsddefault.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o perfdata-base.o ./perfdata.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o xperfdata-base.o ../xdata/xpddefault.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o downtime-base.o ../common/downtime.c
make -C ../lib
make[2]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/lib'
gcc -Wall -g -O2  -I. -I../include -DHAVE_CONFIG_H -c squeue.c -o squeue.o
gcc -Wall -g -O2  -I. -I../include -DHAVE_CONFIG_H -c kvvec.c -o kvvec.o
gcc -Wall -g -O2  -I. -I../include -DHAVE_CONFIG_H -c iocache.c -o iocache.o
gcc -Wall -g -O2  -I. -I../include -DHAVE_CONFIG_H -c iobroker.c -o iobroker.o
gcc -Wall -g -O2  -I. -I../include -DHAVE_CONFIG_H -c bitmap.c -o bitmap.o
gcc -Wall -g -O2  -I. -I../include -DHAVE_CONFIG_H -c dkhash.c -o dkhash.o
```

Name:Advik Hegde Div: D15C Roll No: 15



Run the following set of commands to ensure that
sudo make install

```
ec2-user@ip-172-31-41-160:~/downloads/nagios-4.5.5
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ sudo make install
cd ./base && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/base'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagios /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagiostats /usr/local/nagios/bin
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5/base'
cd ./cgi && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
make install-basic
make[2]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/sbin
for file in *.cgi; do \
        /usr/bin/install -c -s -m 775 -o nagios -g nagios $file /usr/local/nagios/sbin; \
done
make[2]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
cd ./html && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/html'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/media
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/stylesheets
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/contexthelp
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/docs
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/docs/images
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/js
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/images
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/images/logos
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/includes
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/ssi
/usr/bin/install -c -m 664 -o nagios -g nagios ./robots.txt /usr/local/nagios/share
/usr/bin/install -c -m 664 -o nagios -g nagios ./jsonquery.html /usr/local/nagios/share
rm -f /usr/local/nagios/share/index.html
rm -f /usr/local/nagios/share/main.html
rm -f /usr/local/nagios/share/side.html
rm -f /usr/local/nagios/share/map.html
rm -f /usr/local/nagios/share/rss-*
rm -f /usr/local/nagios/share/graph-header.html
rm -f /usr/local/nagios/share/histogram.html
rm -f /usr/local/nagios/share/histogram-form.html
rm -f /usr/local/nagios/share/histogram-graph.html
rm -f /usr/local/nagios/share/histogram-links.html
rm -f /usr/local/nagios/share/infobox.html
rm -f /usr/local/nagios/share/map.php
```

sudo make install-init

```
ec2-user@ip-172-31-41-160:~/downloads/nagios-4.5.5
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ sudo make install-init
/usr/bin/install -c -m 755 -d -o root -g root /lib/systemd/system
/usr/bin/install -c -m 755 -o root -g root startup/default-service /lib/systemd/system/nagios.service
```

sudo make install-config

```
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ sudo make install-config
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc/objects
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/nagios.cfg /usr/local/nagios/etc/nagios.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/cgi.cfg /usr/local/nagios/etc/cgi.cfg
/usr/bin/install -c -b -m 660 -o nagios -g nagios sample-config/resource.cfg /usr/local/nagios/etc/resource.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/templates.cfg /usr/local/nagios/etc/objects/templates.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/commands.cfg /usr/local/nagios/etc/objects/commands.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/contacts.cfg /usr/local/nagios/etc/objects/contacts.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/timeperiods.cfg /usr/local/nagios/etc/objects/timeperiods.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/localhost.cfg /usr/local/nagios/etc/objects/localhost.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/windows.cfg /usr/local/nagios/etc/objects/windows.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/printer.cfg /usr/local/nagios/etc/objects/printer.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/switch.cfg /usr/local/nagios/etc/objects/switch.cfg

*** 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.
```

sudo make install-webconf

```
[ec2-user@ip-172-31-41-160 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-172-31-41-160 nagios-4.5.5]$
```

Next, we are supposed to create a nagiosadmin account for nagios login along with password. Specify the password twice.

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

```
[ec2-user@ip-172-31-41-160 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-172-31-41-160 nagios-4.5.5]$
```

Restart Apache

sudo service httpd restart

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

cd ~/downloads

tar zxvf nagios-plugins-2.4.11.tar.gz

```
ec2-user@ip-172-31-41-160:~/downloads
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ sudo service httpd restart
Redirecting to /bin/systemctl restart httpd.service
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ cd ~/downloads
[ec2-user@ip-172-31-41-160 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
```

Compile and install plugins
cd nagios-plugins-2.4.11
./configure --with-nagios-user=nagios --with-nagios-group=nagios

Run the following command:
sudo chkconfig --add nagios
On running the above command

```
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$ sudo chkconfig --add nagios
error reading information on service nagios: No such file or directory
```

If this is the output that one is getting, then it means that the init script is missing…
We can check this by running ls /etc/init.d/

```
[ec2-user@ip-172-31-92-249 nagios-plugins-2.4.9]$ ls /etc/init.d/
README  functions
[ec2-user@ip-172-31-92-249 nagios-plugins-2.4.9]$
```

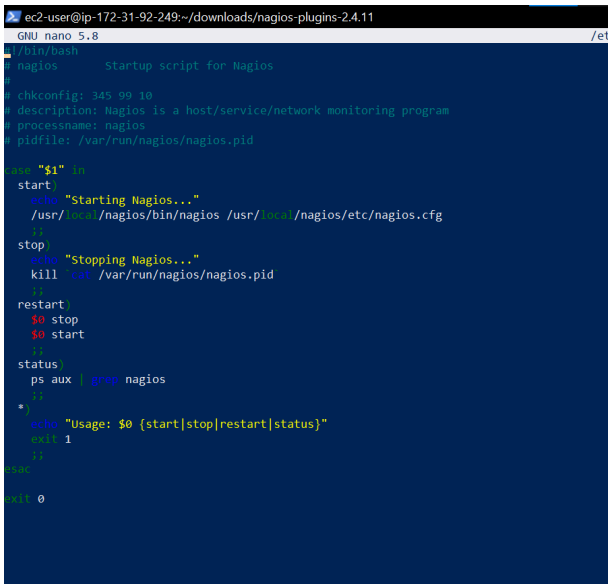
With ls command, we must see a file named nagios, which i was not able to see
If the Init Script is Missing i.e If you don't see the nagios script in /etc/init.d/, you can create it manually. Here's how:

Run the following command:
sudo nano /etc/init.d/nagios

Within this file, paste the following script

Name:Advik Hegde Div: D15C Roll No: 15

```
#!/bin/bash
# nagios       Startup script for Nagios
#
# chkconfig: 345 99 10
# description: Nagios is a host/service/network monitoring program
# processname: nagios
# pidfile: /var/run/nagios/nagios.pid
case "$1" in
  start)
    echo "Starting Nagios..."
    /usr/local/nagios/bin/nagios /usr/local/nagios/etc/nagios.cfg
    ;;
  stop)
    echo "Stopping Nagios..."
    kill `cat /var/run/nagios/nagios.pid`
    ;;
  restart)
    $0 stop
    $0 start
    ;;
  status)
    ps aux | grep nagios
    ;;
  *)
    echo "Usage: $0 {start|stop|restart|status}"
    exit 1
    ;;
esac
exit 0
```



Name:Advik Hegde Div: D15C Roll No: 15

Make the Script Executable: After saving the file, run the following command to make it executable:
sudo chmod +x /etc/init.d/nagios

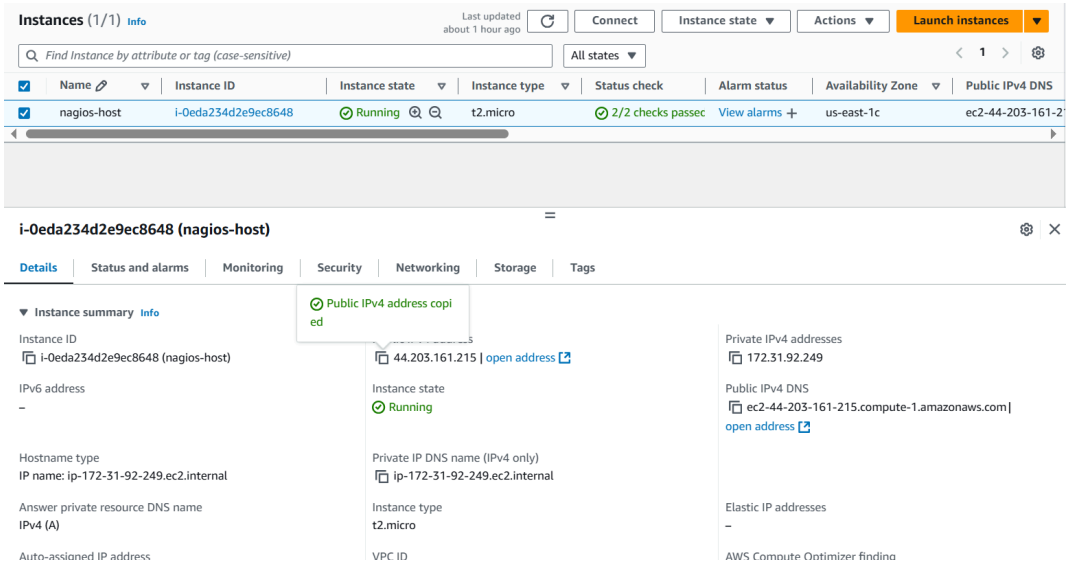
Run **sudo chkconfig --add nagios** again
And then run **sudo chkconfig nagios on**

```
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$ sudo nano /etc/init.d/nagios
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$ sudo chmod +x /etc/init.d/nagios
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$ sudo chkconfig --add nagios
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$ sudo chkconfig nagios on
Note: Forwarding request to 'systemctl enable nagios.service'.
Synchronizing state of nagios.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable nagios
Created symlink /etc/systemd/system/multi-user.target.wants/nagios.service → /usr/lib/systemd/system/nagios.service.
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$
```

sudo service nagios start

```
[ec2-user@ip-172-31-92-249 nagios-plugins-2.4.11]$ sudo service nagios start
Starting Nagios...

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
Nagios 4.5.5 starting... (PID=72261)
Local time is Tue Oct 01 20:59:58 UTC 2024
wproc: Successfully registered manager as @wproc with query handler
wproc: Registry request: name=Core Worker 72265;pid=72265
wproc: Registry request: name=Core Worker 72264;pid=72264
wproc: Registry request: name=Core Worker 72263;pid=72263
wproc: Registry request: name=Core Worker 72262;pid=72262
Successfully launched command file worker with pid 72266
wproc: NOTIFY job 4 from worker Core Worker 72262 is a non-check helper but exited with return code 127
wproc:   host=localhost; service=Swap Usage; contact=nagiosadmin
wproc:   early_timeout=0; exited_ok=1; wait_status=32512; error_code=0;
wproc:   stderr line 01: /bin/sh: line 1: /bin/mail: No such file or directory
wproc:   stderr line 02: /usr/bin/printf: write error: Broken pipe
```

Get your public IPv4 address from your instance. We will require it for connecting to our nginx server



Name:Advik Hegde Div: D15C Roll No: 15

Browse for this url: http://<your_public_ip_address>/nagios
The browser may ask you for your nagios credentials which set in the earlier steps
The username is nagiosadmin and enter the password that you set earlier

Conclusion:

In this experiment, we successfully installed and configured Nagios Core on an Amazon Linux EC2 instance, showcasing its role in continuous monitoring within a DevOps environment. We learned about user management and service configuration, emphasizing Nagios's ability to monitor systems and networks effectively. This experience laid the groundwork for enhancing infrastructure reliability and integrating advanced monitoring strategies in future projects.

Name: Advik Hegde Div: D15C Roll No: 15

Adv DevOps Exp 10

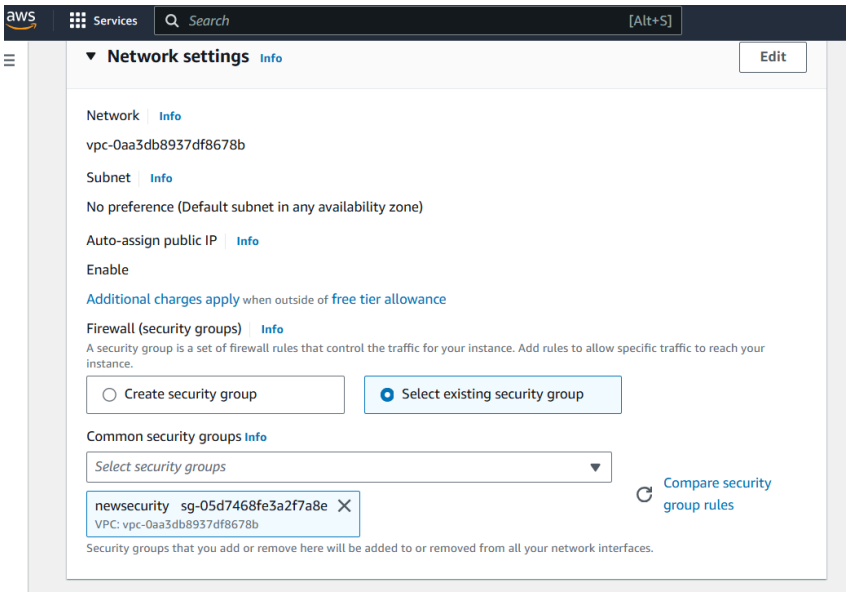
Aim: To perform Port, Service monitoring, Windows/Linux server monitoring using Nagios.

Monitoring Using Nagios:

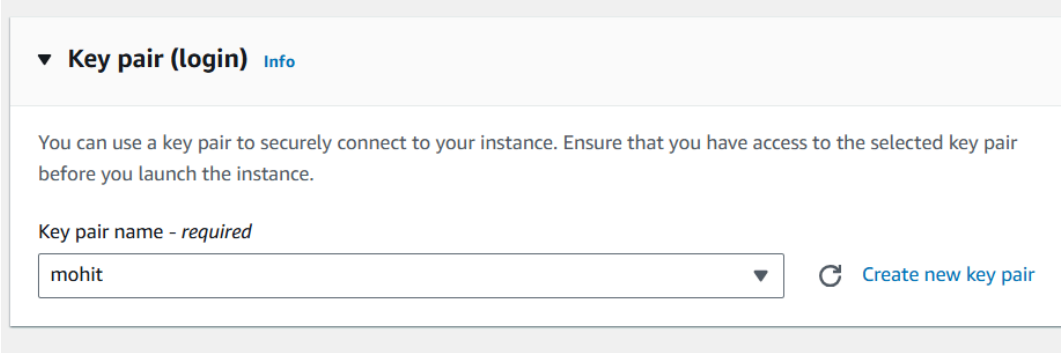
Step 1: To Confirm Nagios is running on the server side Perform the following command on your Amazon Linux Machine (Nagios-host).
Run this command **sudo systemctl status**

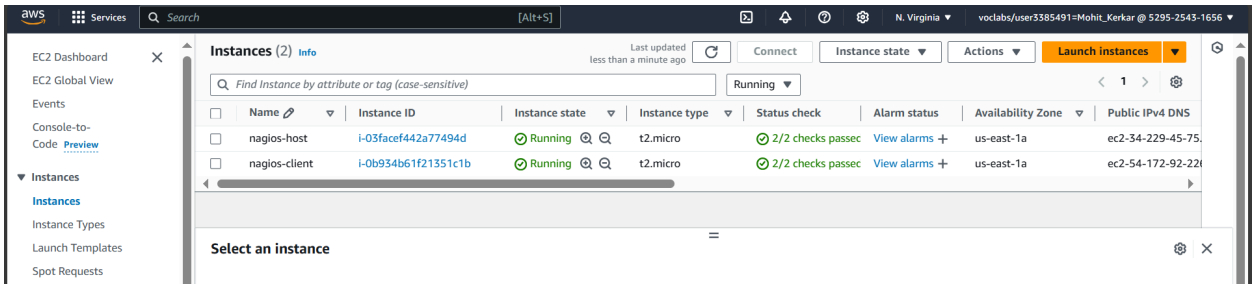
```
ec2-user@ip-172-31-41-160:~/downloads/nagios-plugins-2.4.11
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$ sudo systemctl status
● ip-172-31-41-160.ec2.internal
    State: running
    Units: 296 loaded (incl. loaded aliases)
     Jobs: 0 queued
   Failed: 0 units
    Since: Wed 2024-10-02 12:28:05 UTC; 33min ago
  systemd: 252.23-2.amzn2023
  CGroup: /
          ├─init.scope
          │ └─1 /usr/lib/systemd/systemd --switched-root --system --deserialize=32
          ├─system.slice
          │ ├─acpid.service
          │ │ ├─1938 /usr/bin/systemd-inhibit --what=handle-suspend-key:handle-hibernate-key --who=noah "--why=acpid instead" --mode=block /usr/sbin/acpid -f
          │ │ └─2059 /usr/sbin/acpid -f
          │ ├─amazon-ssm-agent.service
          │ │ └─2141 /usr/bin/amazon-ssm-agent
          │ ├─atd.service
          │ │ └─2152 /usr/sbin/atd -f
          │ ├─auditd.service
          │ │ └─1768 /sbin/auditd
          │ ├─chronyd.service
          │ │ └─2175 /usr/sbin/chronyd -F 2
          │ ├─dbus-broker.service
          │ │ ├─1946 /usr/bin/dbus-broker-launch --scope system --audit
          │ │ └─1954 dbus-broker --log 4 --controller 9 --machine-id ec2e4d759a3e2f6fe850b14e4cdacabe --max-bytes 536870912 --max-fds 4096 --max-matches 16384 --audit
          │ ├─gssproxy.service
          │ │ └─1959 /usr/sbin/gssproxy -D
          │ ├─httpd.service
          │ │ ├─49553 /usr/sbin/httpd -DFOREGROUND
          │ │ ├─49555 /usr/sbin/httpd -DFOREGROUND
          │ │ ├─49556 /usr/sbin/httpd -DFOREGROUND
          │ │ ├─49557 /usr/sbin/httpd -DFOREGROUND
          │ │ ├─49558 /usr/sbin/httpd -DFOREGROUND
          │ │ └─62800 /usr/sbin/httpd -DFOREGROUND
          │ ├─libstoragemgmt.service
          │ │ └─1940 /usr/bin/lsmd -d
```

Step 2: Before we begin,
To monitor a Linux machine, create an **Ubuntu 20.04 server** EC2 Instance in AWS. Provide it with the **same security group** as the Nagios Host and name it ‘nagios-client’ alongside the host.



Name: Advik Hegde Div: D15C Roll No: 15





Step 3: TO BE DONE IN THE Nagios-host TERMINAL
In the nagios-host terminal, run this command
ps -ef | grep nagios

```
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$ ps -ef | grep nagios
ec2-user   63115    2315  0 13:03 pts/0    00:00:00 grep --color=auto nagios
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$
```

To become a root user, run **'sudo su'** and make two directories using the following commands. If one is running these commands in windows powershell, make sure that he/she copies it line by line as powershell might make an error while interpreting multiple lines
mkdir /usr/local/nagios/etc/objects/monitorhosts
mkdir /usr/local/nagios/etc/objects/monitorhosts/linuxhosts

```
[ec2-user@ip-172-31-92-249 ~]$ sudo su
[root@ip-172-31-92-249 ec2-user]# mkdir /usr/local/nagios/etc/objects/monitorhosts
[root@ip-172-31-92-249 ec2-user]# mkdir /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
[root@ip-172-31-92-249 ec2-user]#
```

Copy the sample localhost.cfg file to linuxhost folder. Use the following mentioned command to achieve it
cp /usr/local/nagios/etc/objects/localhost.cfg
/usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg

Open linuxserver.cfg using nano and make the following changes. This is a conf type file in which we will have to modify the configurations in way which will help us specify the hosts and clients to be monitored

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

Changes to be made:

1. Change the hostname to linux-server (EVERYWHERE ON THE FILE)
2. Change address to the public IP address of your LINUX CLIENT.
3. Change hostgroup_name under hostgroup to linux-servers1

```
# HOST DEFINITION
#
###############################################################################

# Define a host for the local machine

define host {

    use                     linux-server            ; Name of host template to use
                                                    ; This host definition will inherit all variables that are defined
                                                    ; in (or inherited by) the linux-server host template definition.
    host_name               linux-server
    alias                   localhost
    address                 54.172.92.226
}



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

```
# 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
}
```

IMP: Everywhere else on the file, change the hostname to linux-server instead of localhost.

Open the Nagios Config file and add the following line

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

Add the following line in the file and save

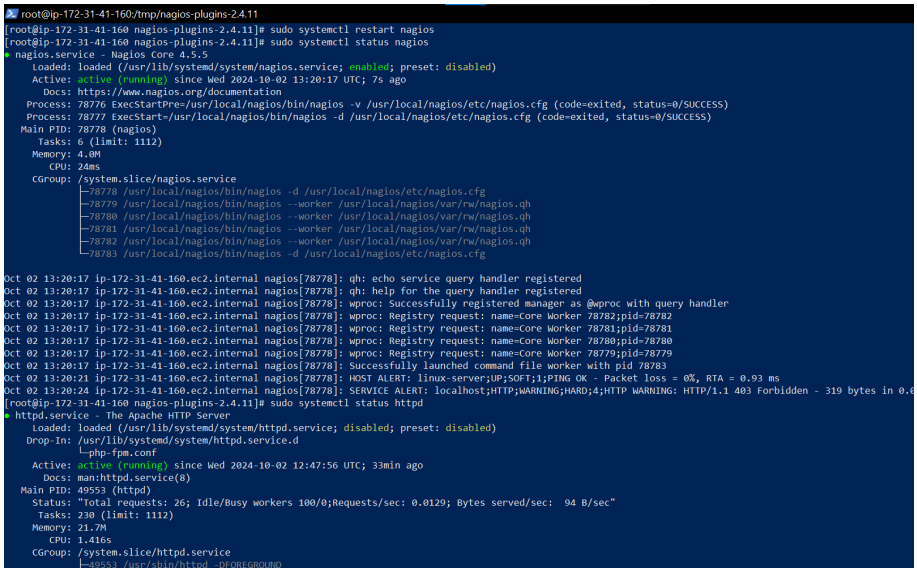
cfg_dir=/usr/local/nagios/etc/objects/monitorhosts/

```
# OBJECT CONFIGURATION FILE(S)
# These are the object configuration files in which you define hosts,
# 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
cfg_dir=/usr/local/nagios/etc/objects/monitorhosts/
# Definitions for monitoring a Windows machine
#cfg_file=/usr/local/nagios/etc/objects/windows.cfg
```

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

```
[root@ip-172-31-41-160 nagios-plugins-2.4.11]# /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.
        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-172-31-41-160 nagios-plugins-2.4.11]#
```

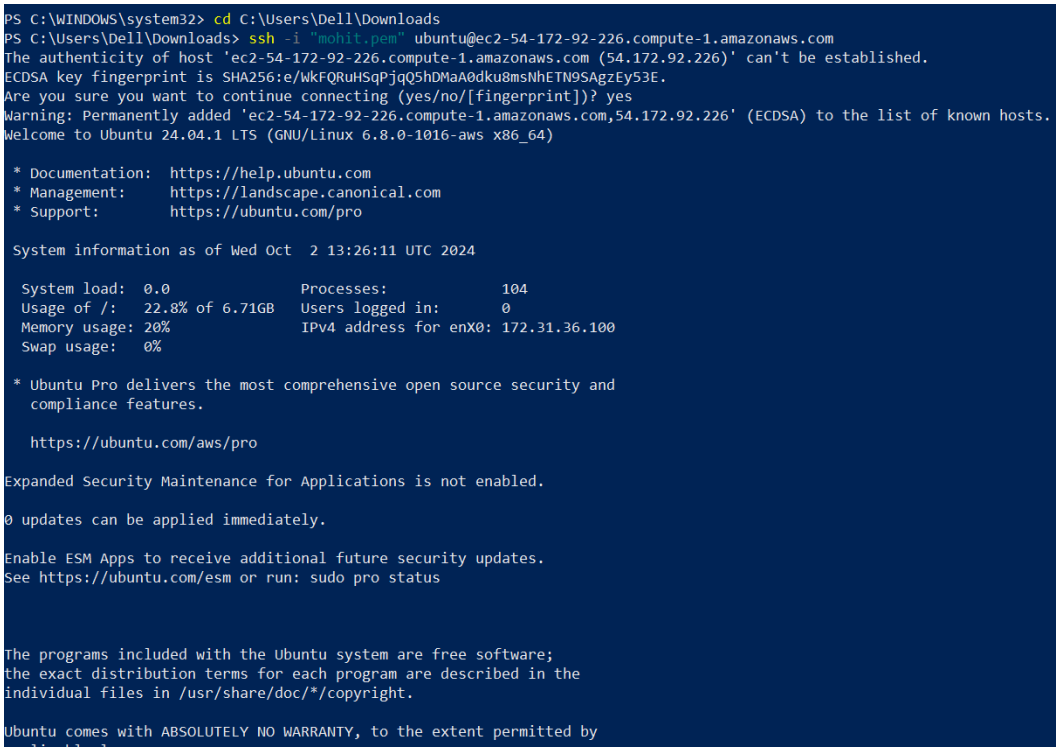
You are good to go if there are no errors.
Restart the nagios service
service nagios restart
And by running sudo systemctl status nagios, we can again check whether our server is running or not



Step 4: TO BE DONE IN THE Nagios-client TERMINAL

Now it is time to switch to the client machine.

SSH into the machine or simply use the EC2 Instance Connect feature.



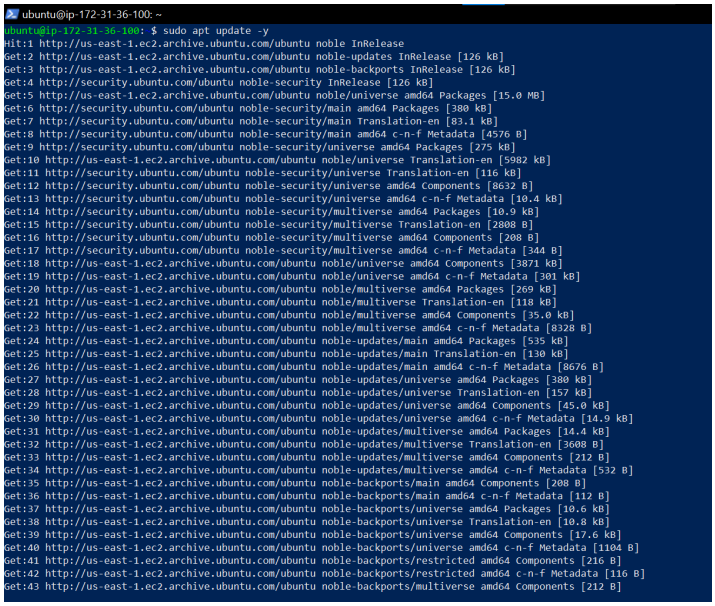
Make a package index update and install gcc, nagios-nrpe-server and the plugins. Run the following commands to achieve the same.

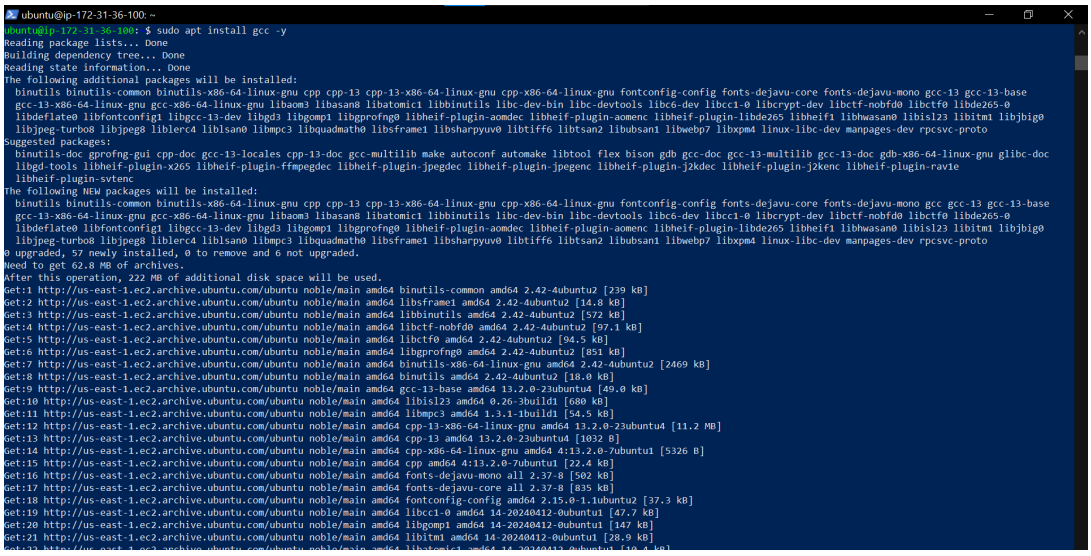
sudo apt update -y

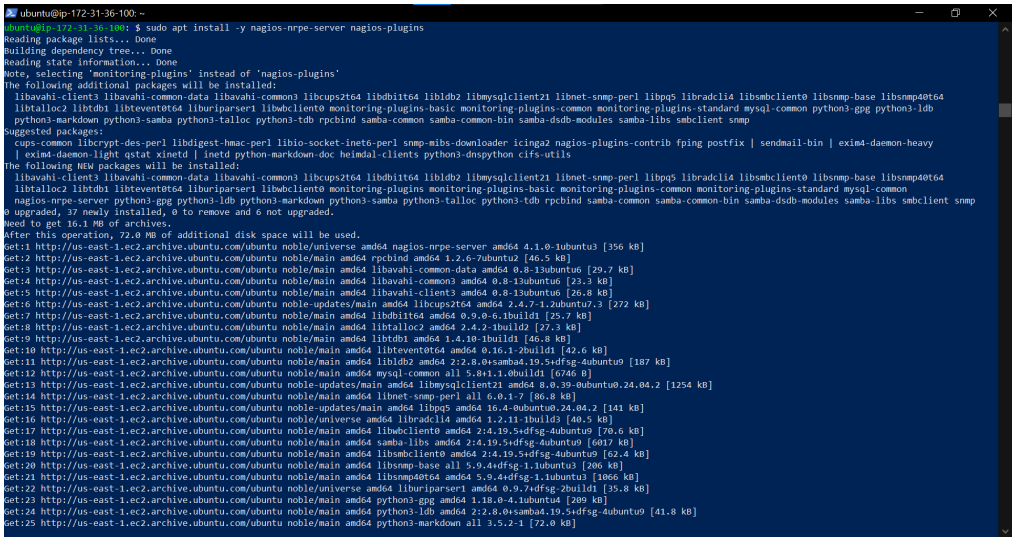
sudo apt install gcc -y

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

Name: Advik Hegde Div: D15C Roll No: 15





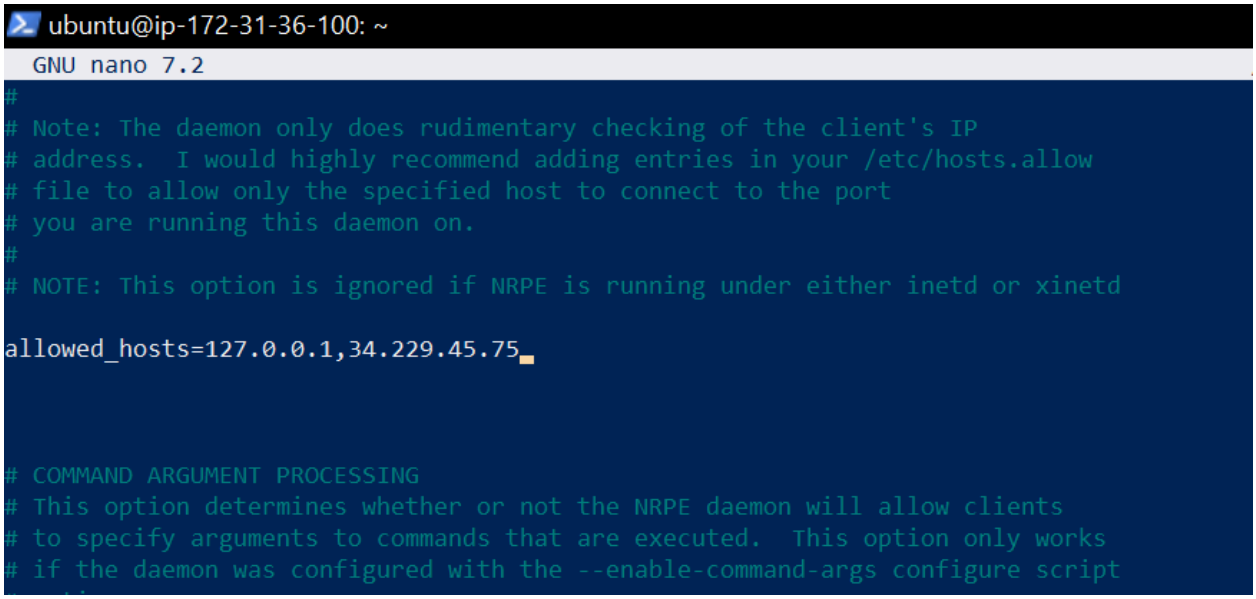


Name: Advik Hegde Div: D15C Roll No: 15

Open nrpe.cfg file to make changes.

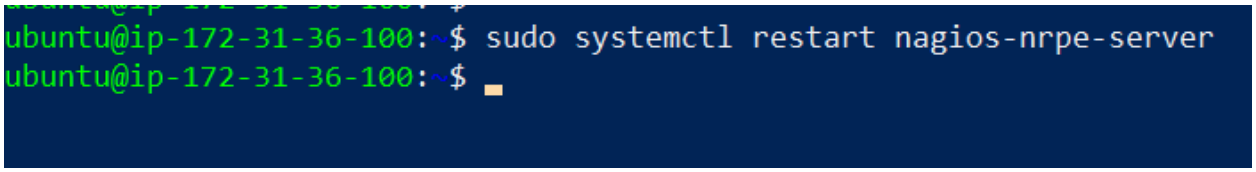
sudo nano /etc/nagios/nrpe.cfg

Under allowed_hosts, add your nagios host IP address like so



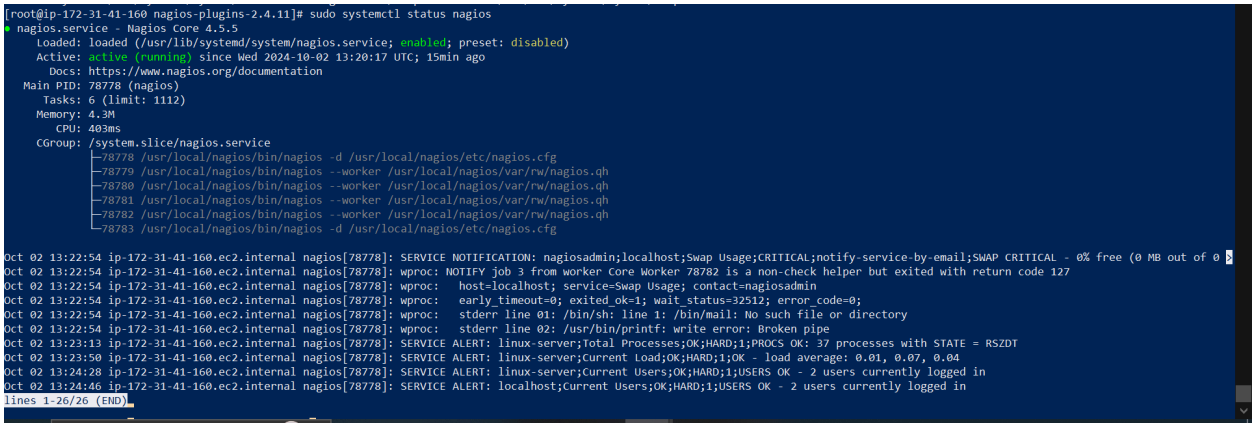
Now restart the NRPE server by this command.

sudo systemctl restart nagios-nrpe-server



Run the following command in the Nagios-host terminal

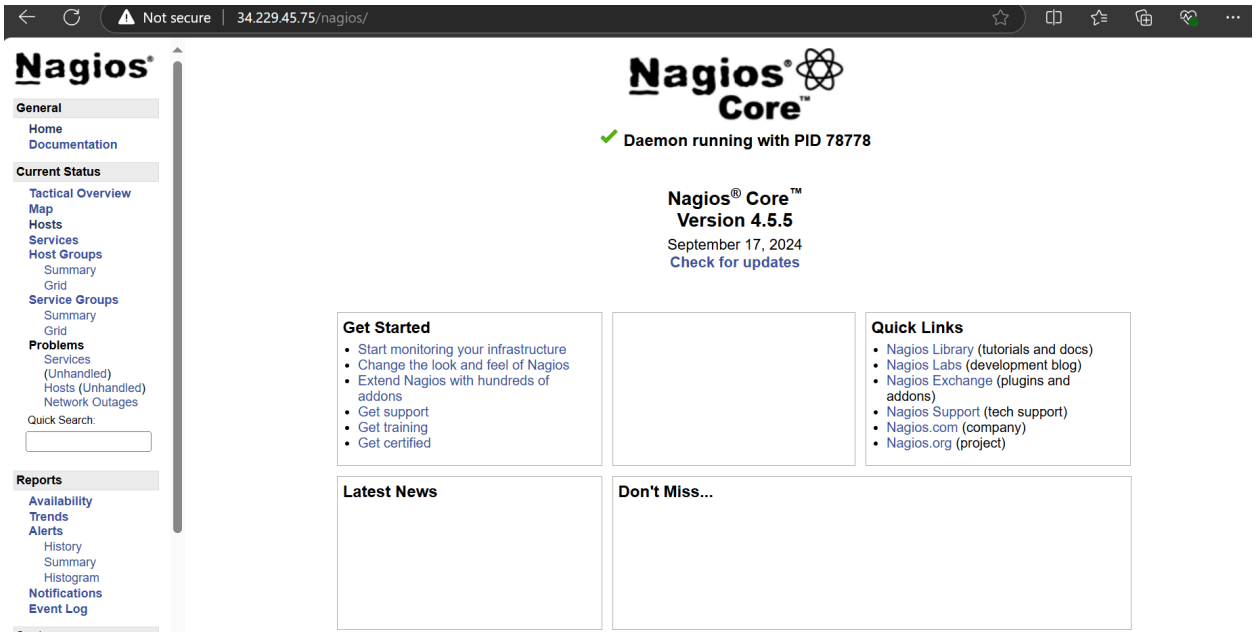
sudo systemctl status nagios



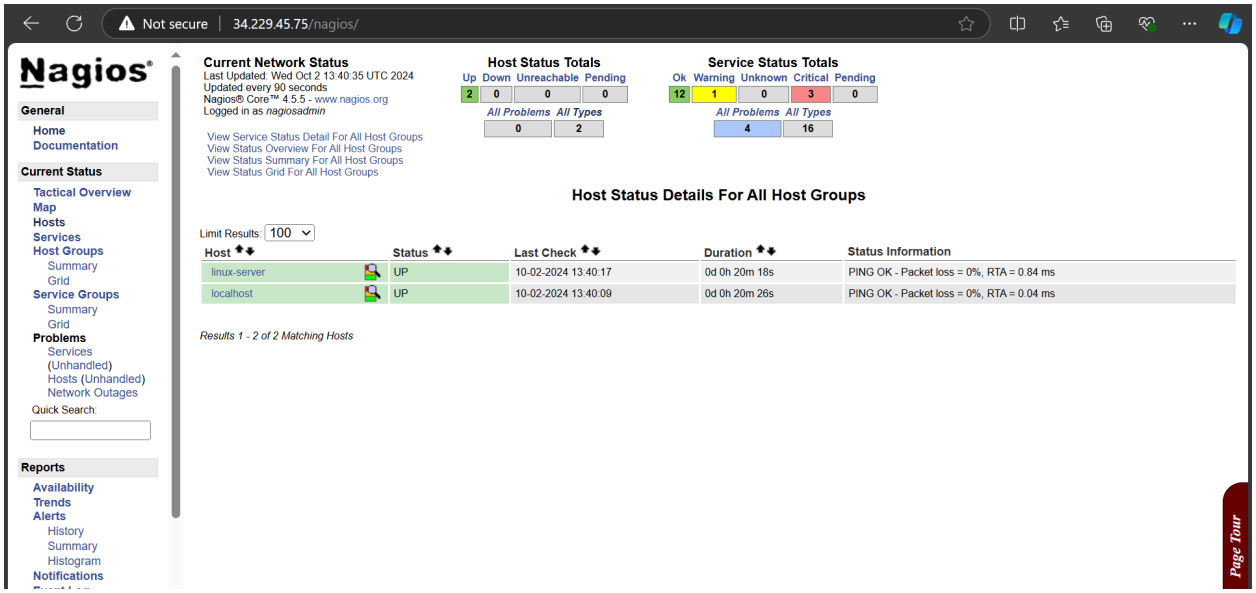
Name: Advik Hegde Div: D15C Roll No: 15

Step 5: Visiting your nagios server using your nagios-host ip address

Open up your browser and look for http://<public_ip_address_of_nagios-host>/nagios

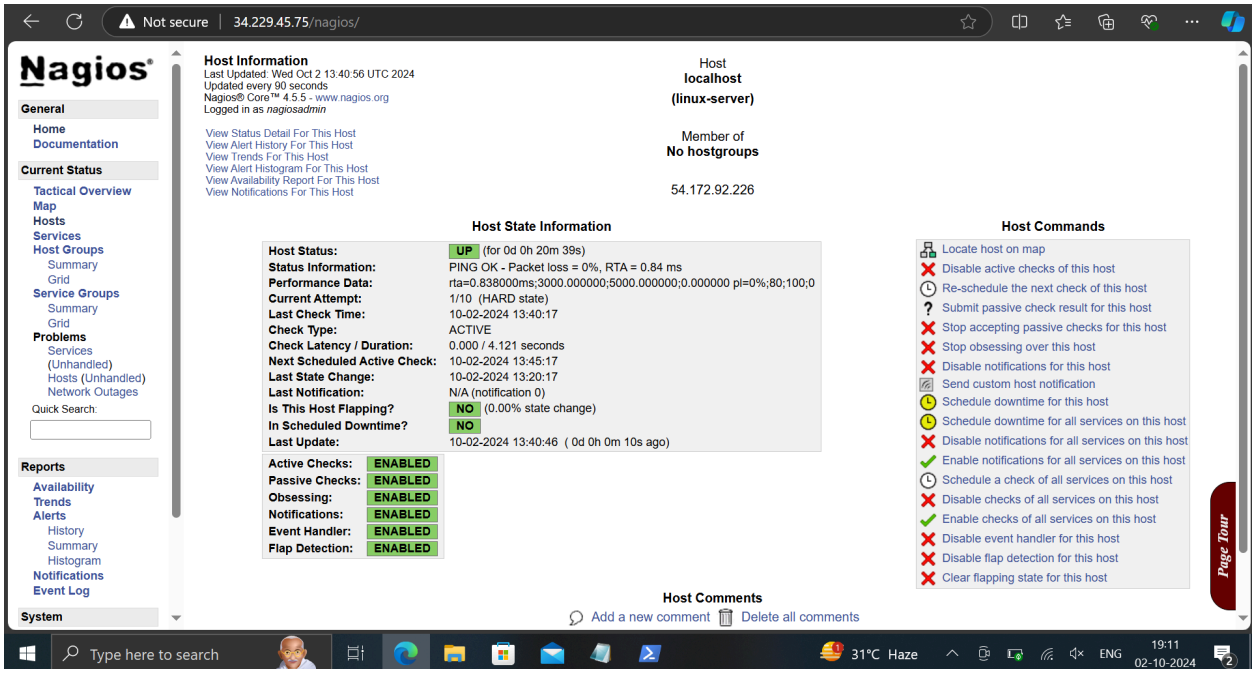


Click on Hosts.

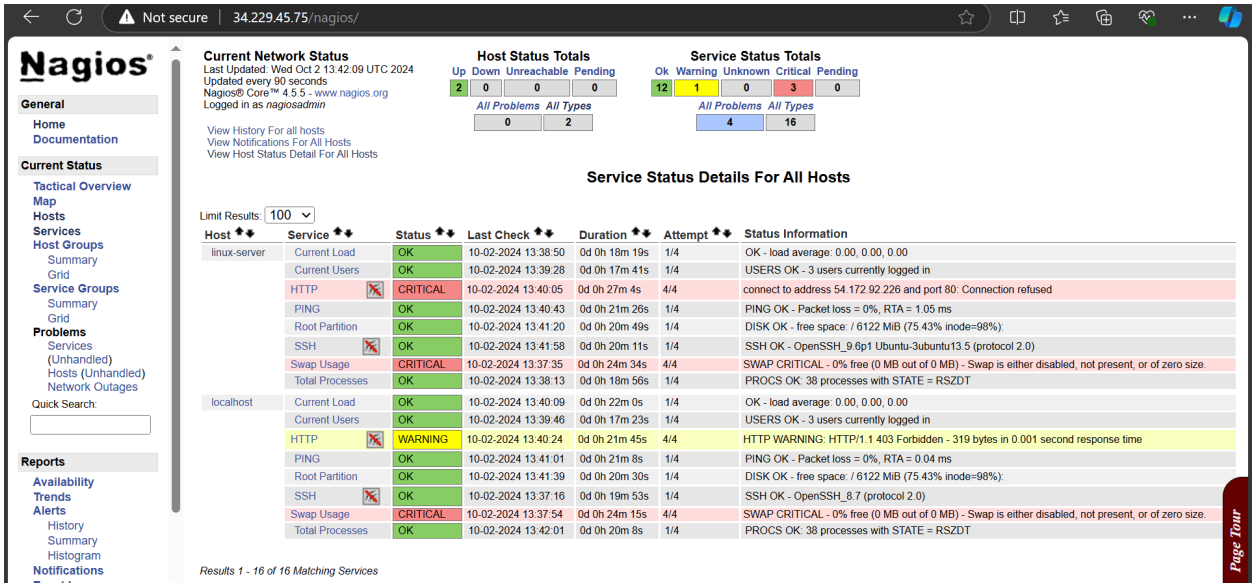


Name: Advik Hegde Div: D15C Roll No: 15

Click on linux-server to view host information



We can even navigate to the services section, which explicitly mentions the status, duration, checks, information about the numerous services present on our hosts



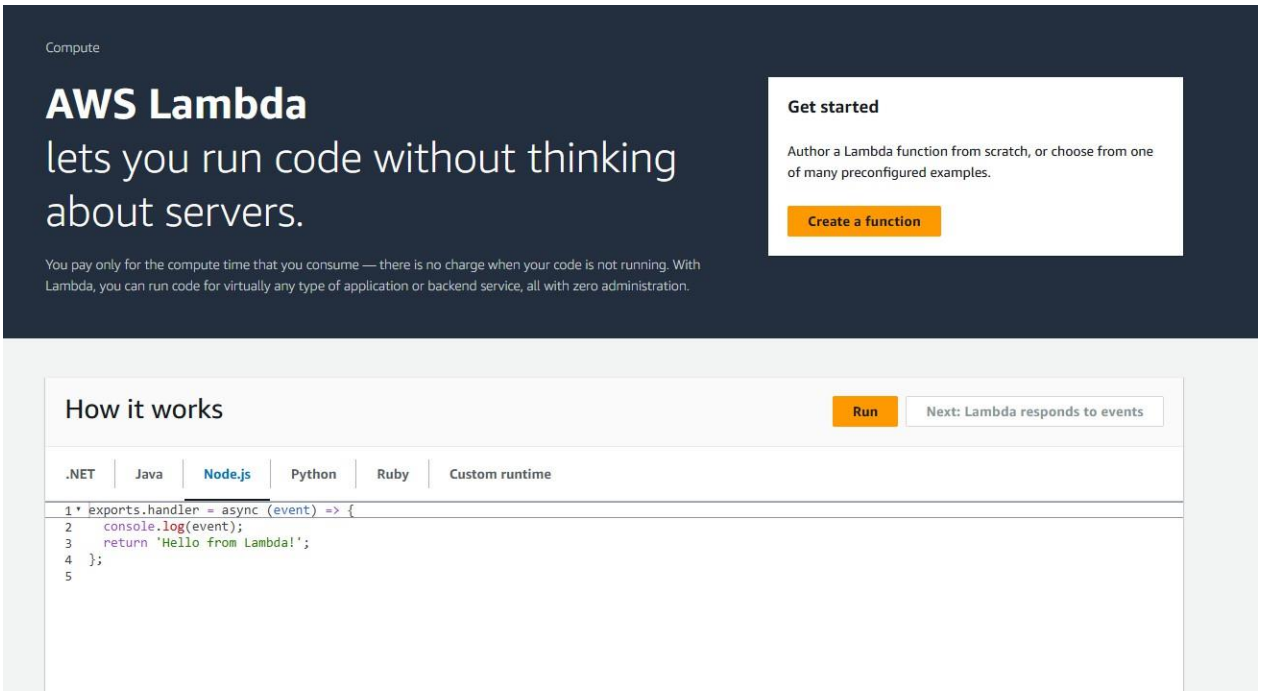
Conclusion: In conclusion, the experiment focused on monitoring ports, services, and a Linux server using Nagios. Through the step-by-step process, we successfully configured Nagios to monitor essential network services on the Linux server. By setting up both the Nagios host and client, we were able to track system performance, ensure service availability, and monitor key metrics like CPU and memory usage

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

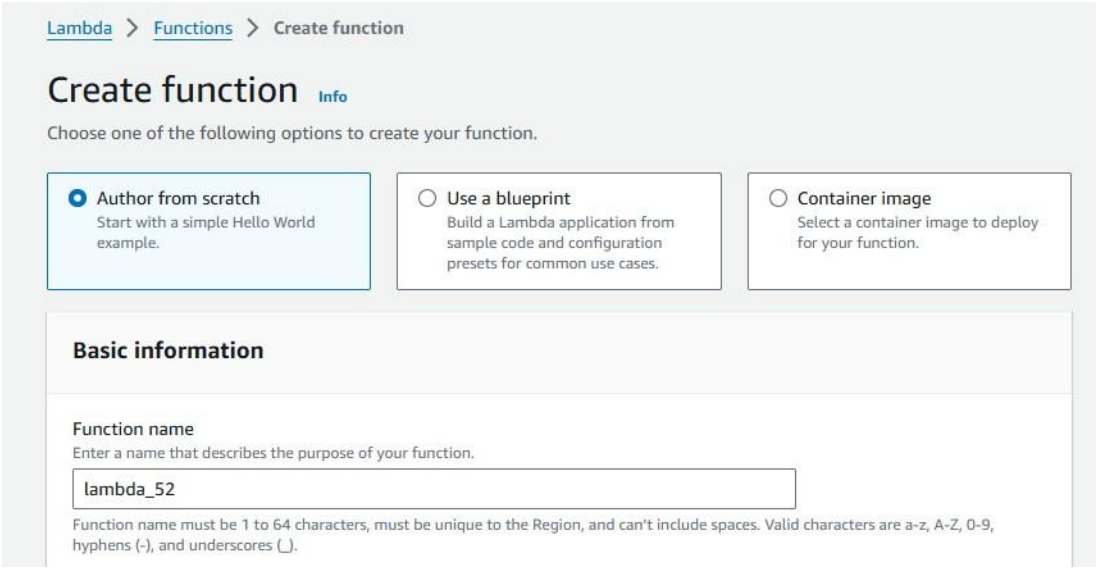
Steps:

1. Go to AWS ACADEMY.
2. **Create the lambda function:**

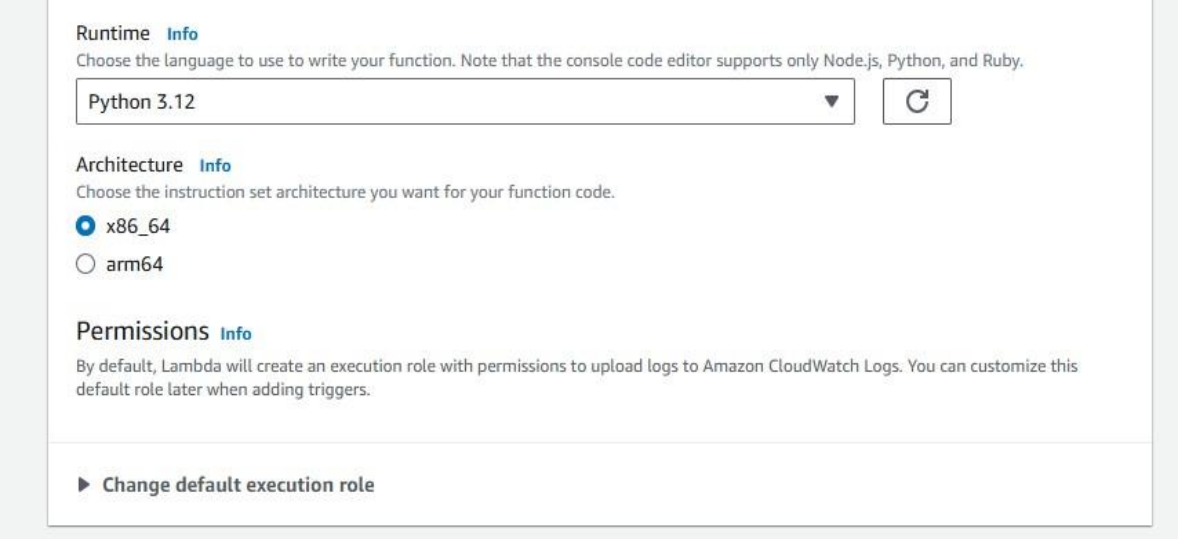
Firstly, Search lambda, then Open lambda and then click on create function button.



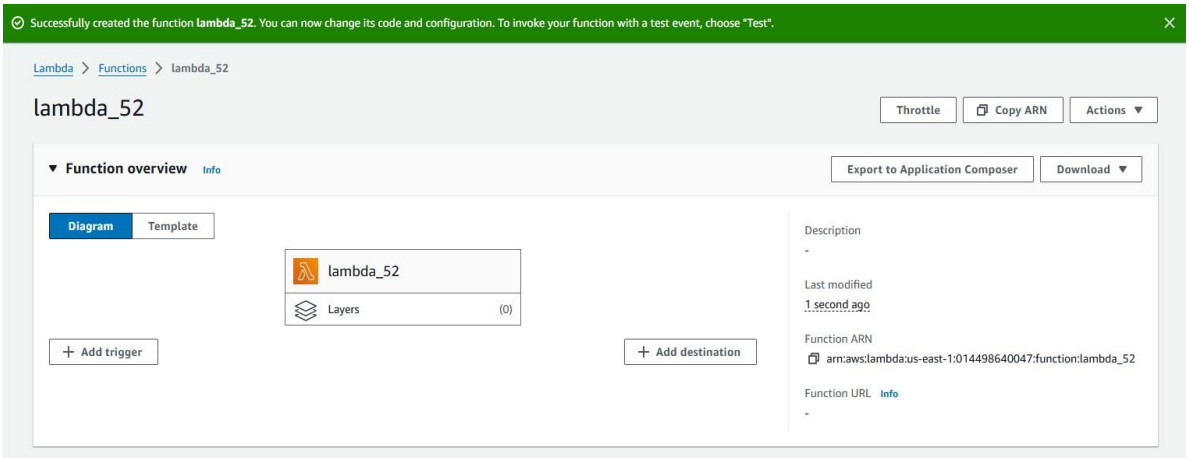
3. Now Give a name to your Lambda function,



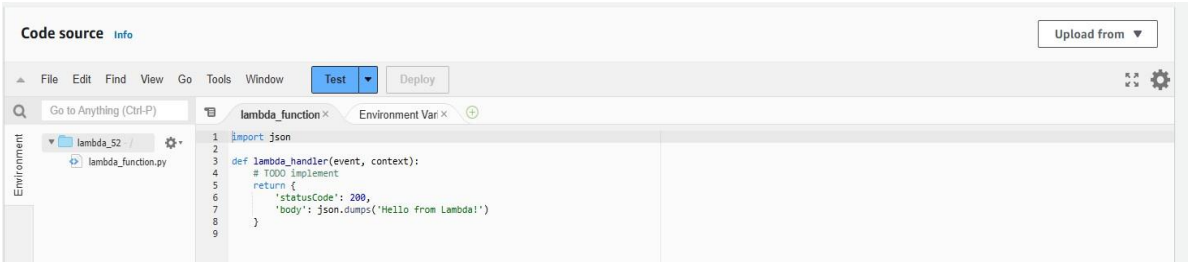
4. Select the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby. So will select Python 3.12, Architecture as x86, and Exceution role to Create a new role with basic Lambda permissions.



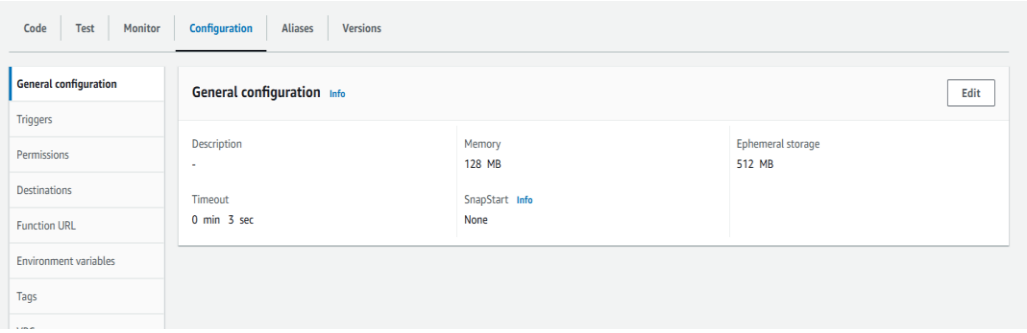
Thus the Lambda function was created successfully.



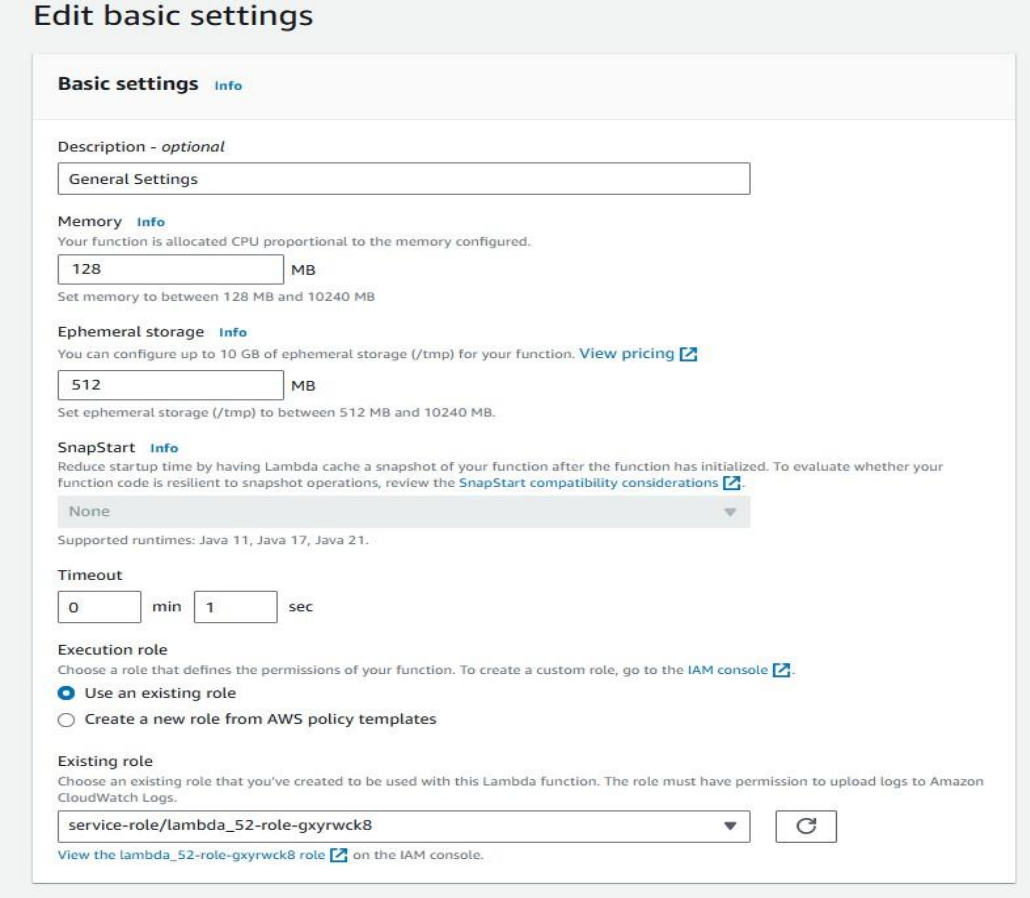
5. Code Source Section



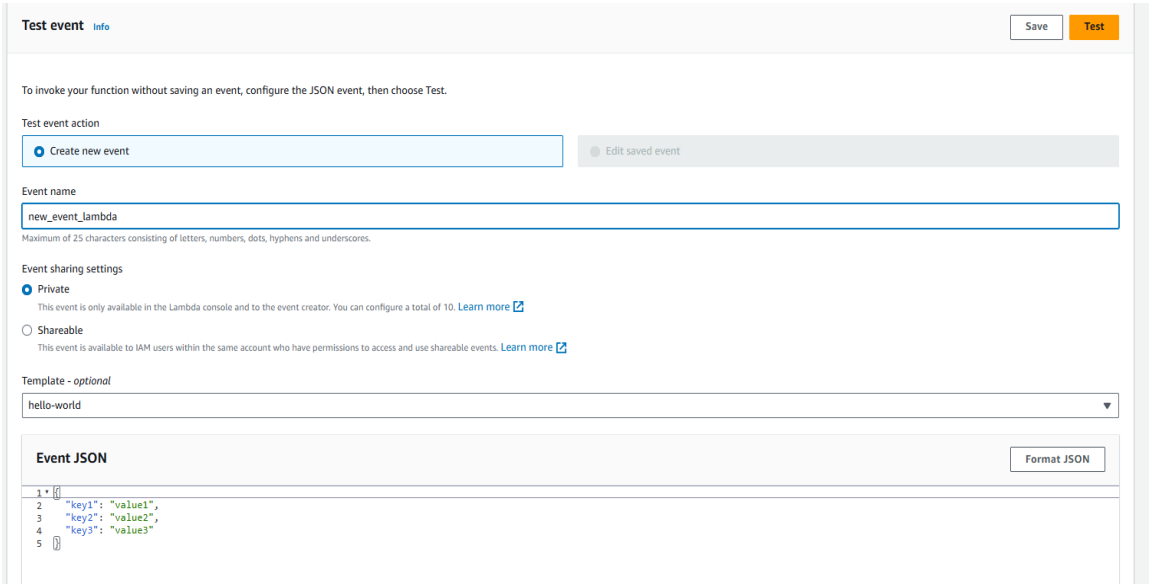
So now to edit the basic settings, go to configuration then click on edit.



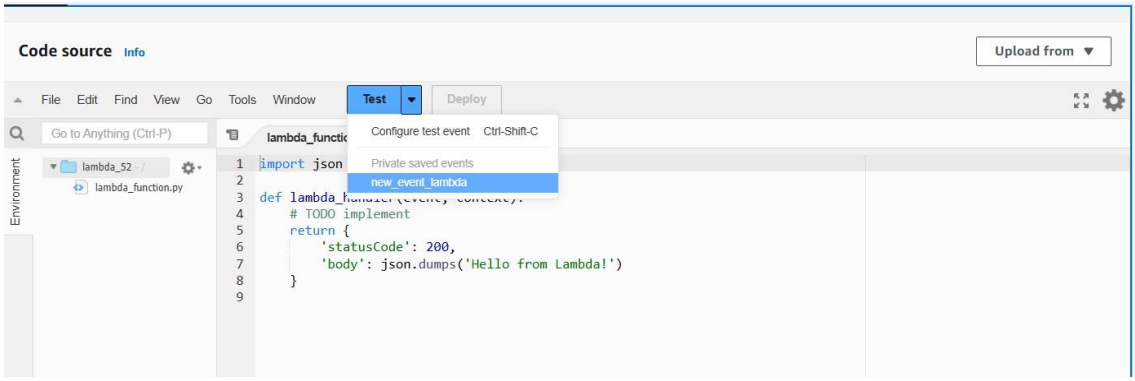
Now enter a description and change Memory and Timeout.Here, I've changed the Timeout period to 1 sec.



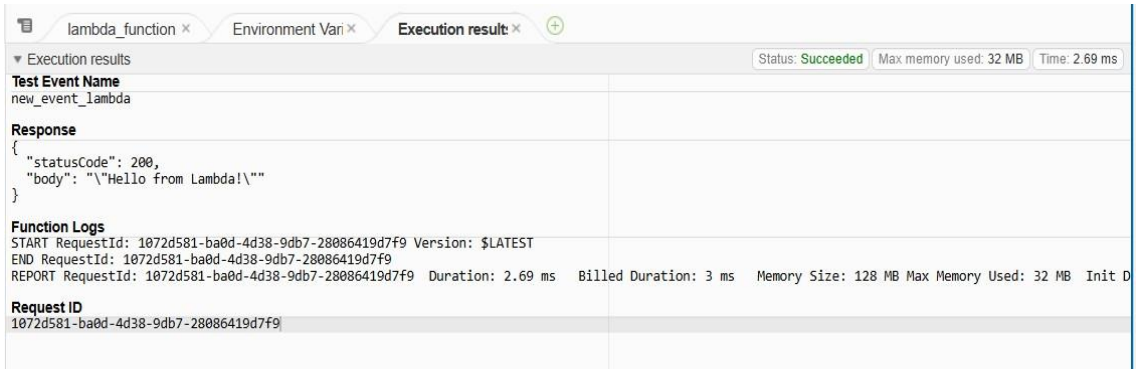
6. Now Click on the Test tab then select Create a new event, give a name to the event here i have given name as "new_event_lambda" and then select event sharing to private, and select hello-world template.



7. **Testing & Deployment:** Now In Code section select the created event (our_event) from the dropdown of test ,then click on test .



8. Now you will see the following output.

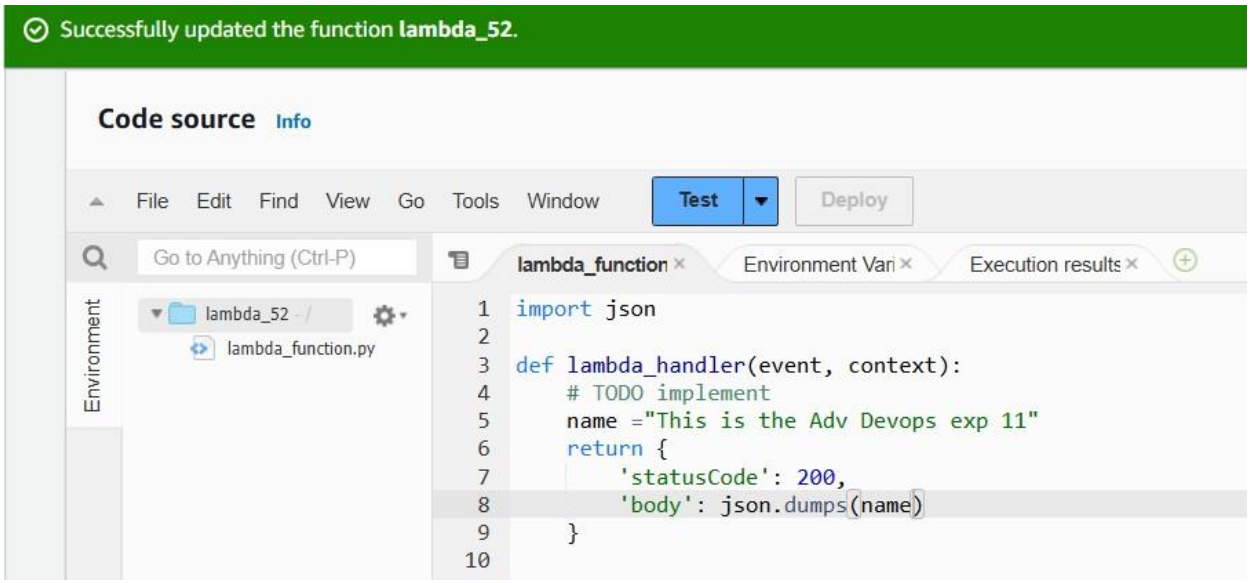


9. **Editing the given code into our own by adding a string or you can add anything**

You can edit your lambda function code.Here I have created a new string name “new_string” and

assigned a string to it.

Now save it by ctrl+s and then click finally on deploy to deploy the changes.



10. **Testing and redeploying changes** Now click on the test and observe the output. Thus Output gives status code 200. This deployment is done successfully.

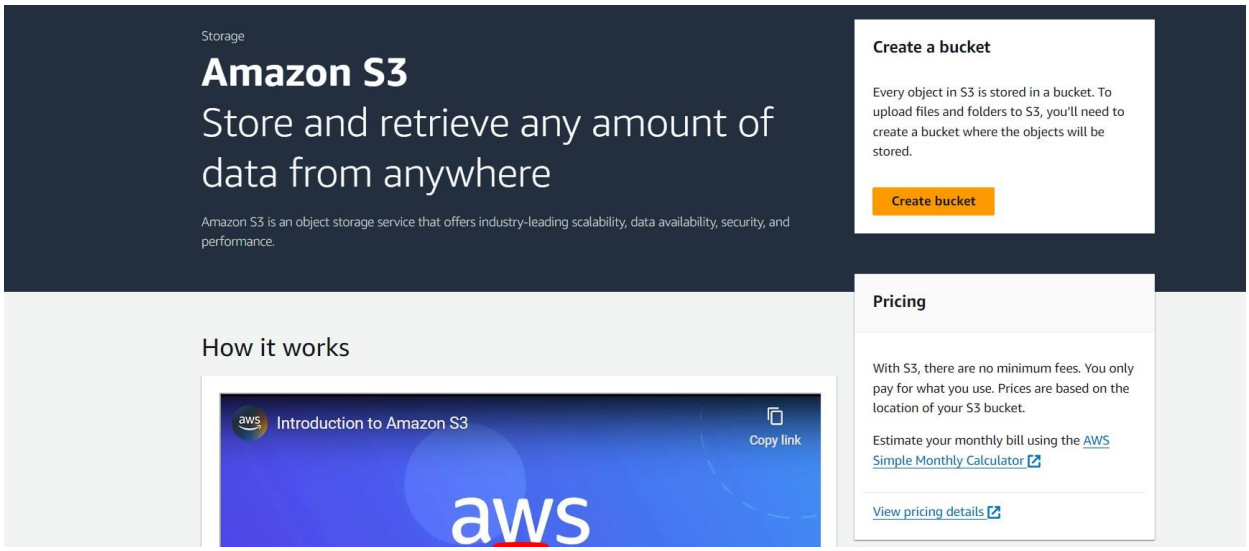
Conclusion:

In this experiment, we successfully created an AWS Lambda function using Python as the chosen language. After configuring the basic settings, we adjusted the timeout to 1 second, tested the function, and deployed it. The deployment was successful. We then modified the Lambda function's code and redeployed it to observe real-time changes. This process highlighted the ease of using AWS Lambda for building serverless applications. However, one issue we encountered was that, while we initially added the code source in Python, we could have chosen other supported languages as well.

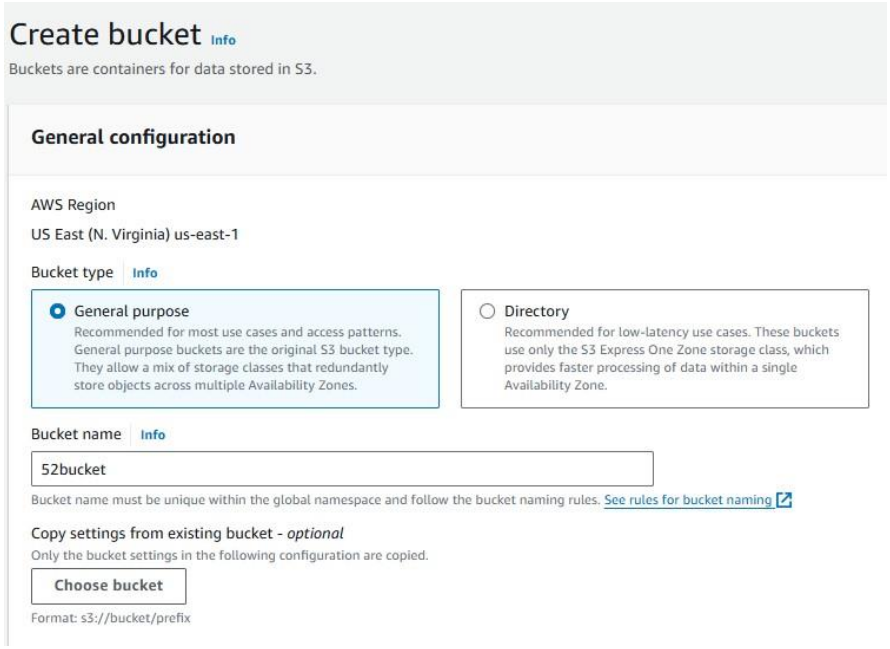
Aim: To create a Lambda function which will log “An Image has been added” once you add an object to a specific bucket in S3

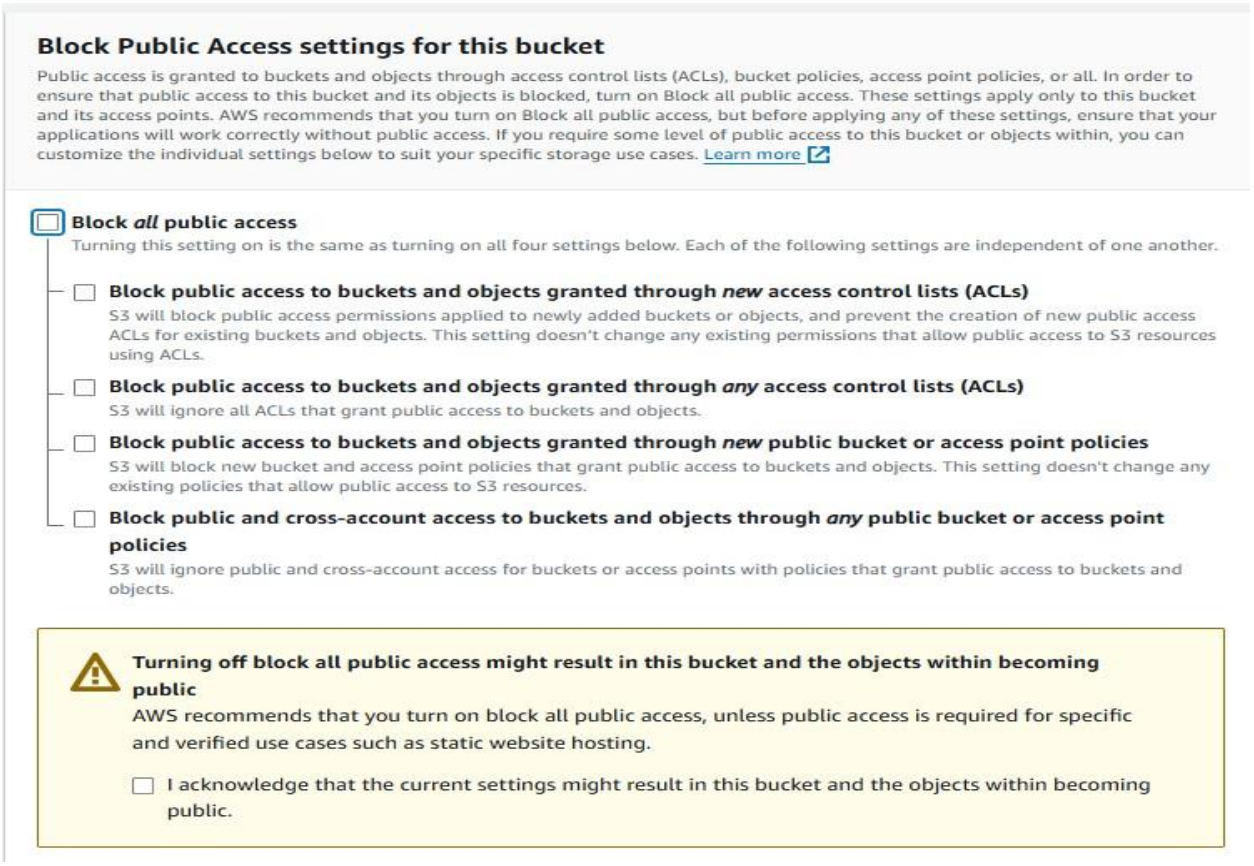
Steps :

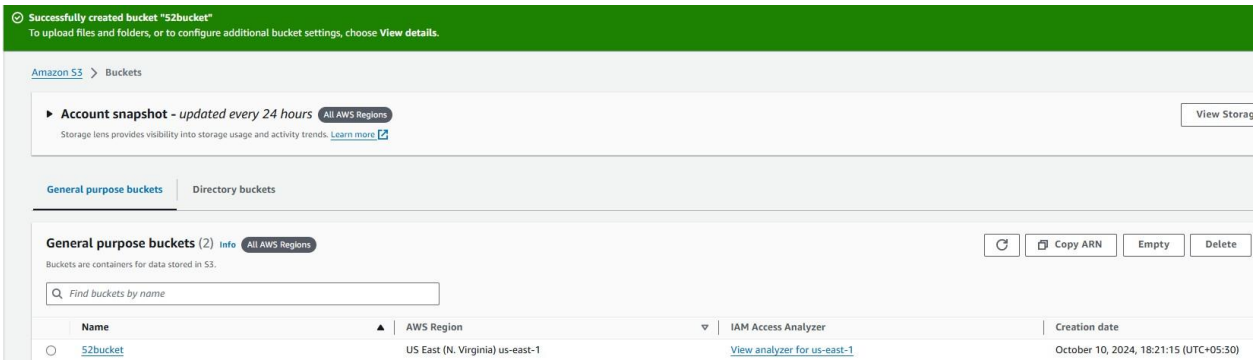
1: Login to your AWS Personal account. Now open S3 from services and click on create S3 bucket.



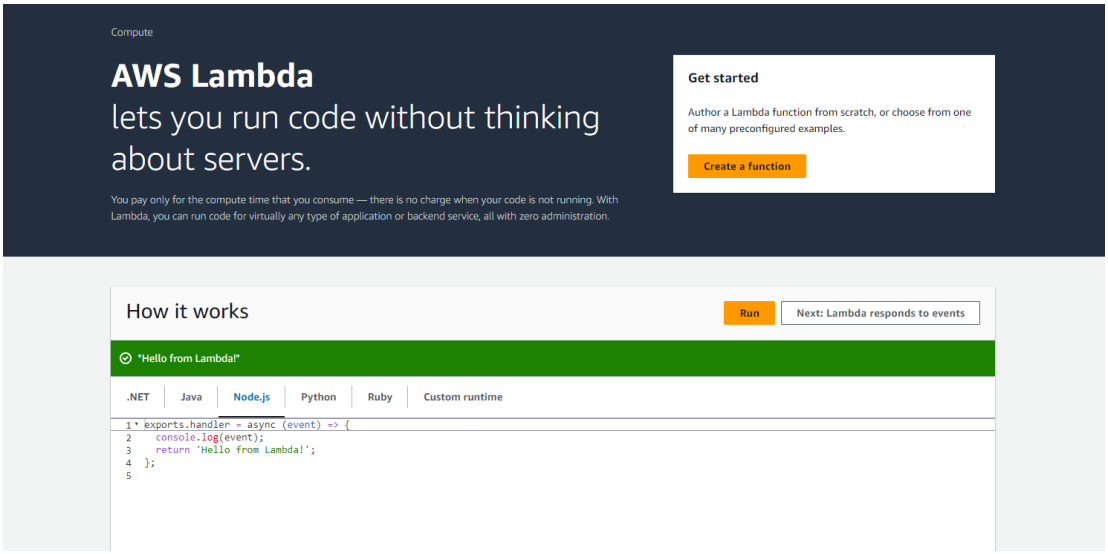
2: Now Give a name to the Bucket, select general purpose project and deselect the Block public access and keep other this to default.



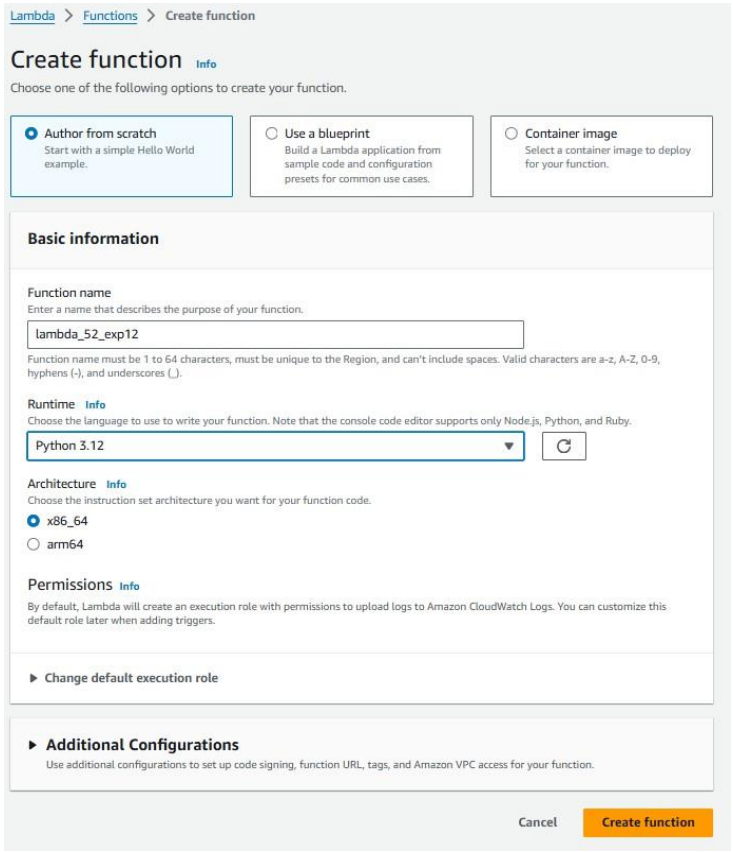




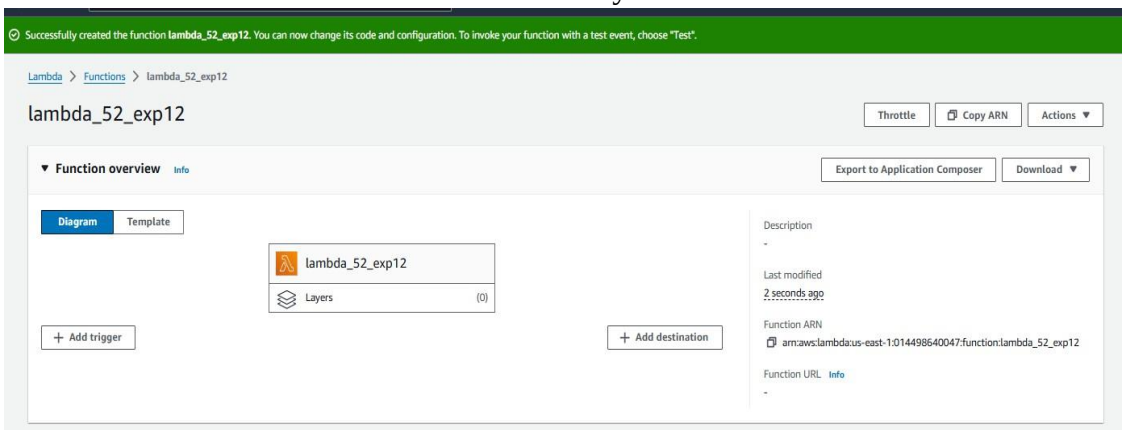
3. Search and Open lambda console and click on create function button.



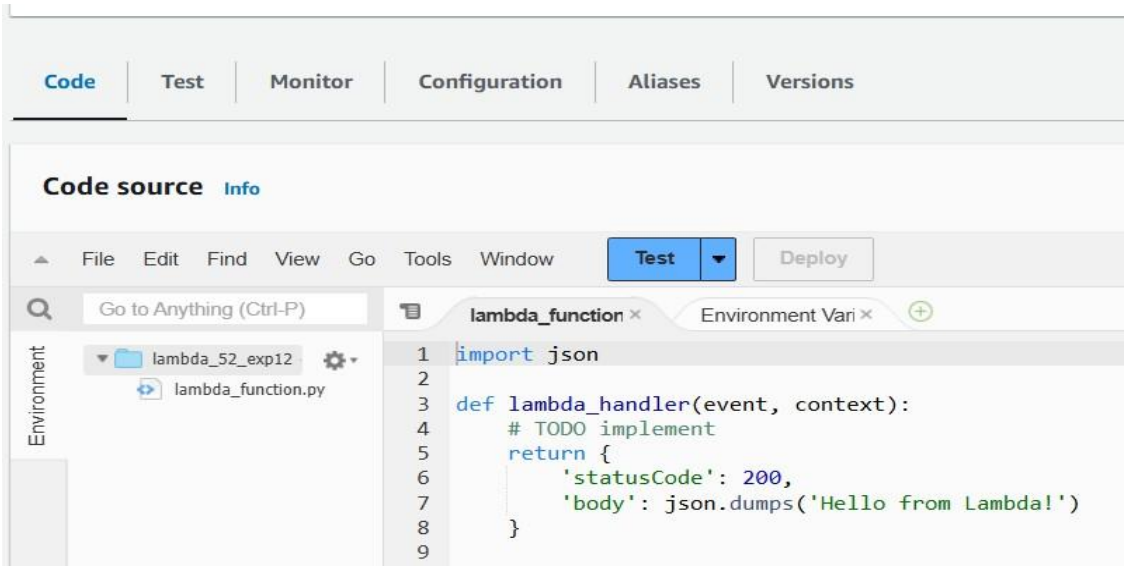
4. Now Give a name to your Lambda function, Select the language to write your function.Here I have chosen python3.12, Architecture as x86, and Execution role to Create a new role with basic Lambda permissions. Note that the console code editor supports only Node.js, Python, and Ruby



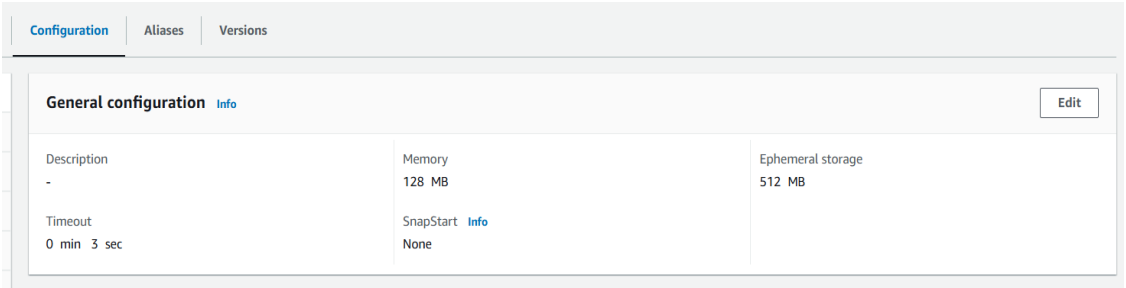
5. The Lambda function was created successfully.



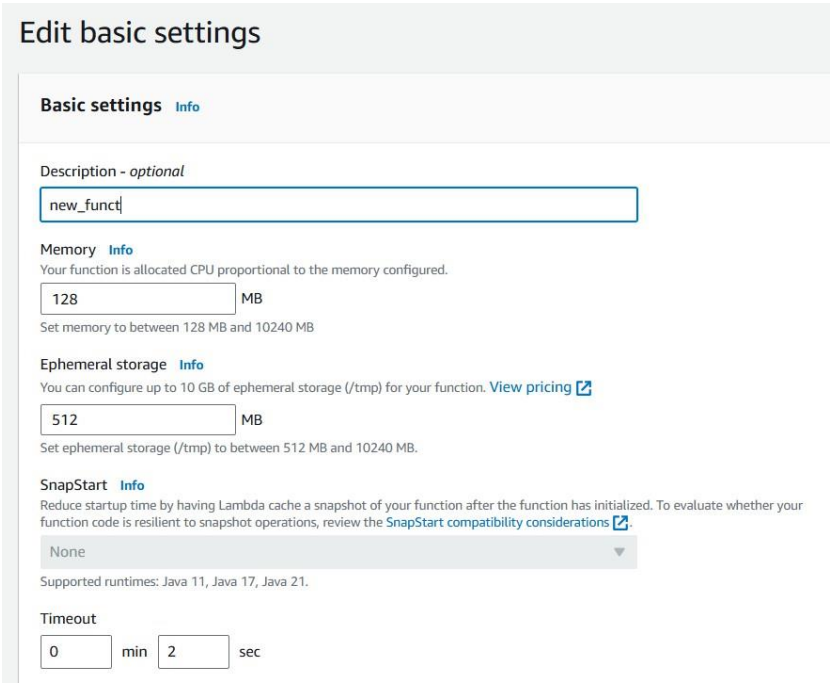
6. Then Go into the code section.You will see some default code there.



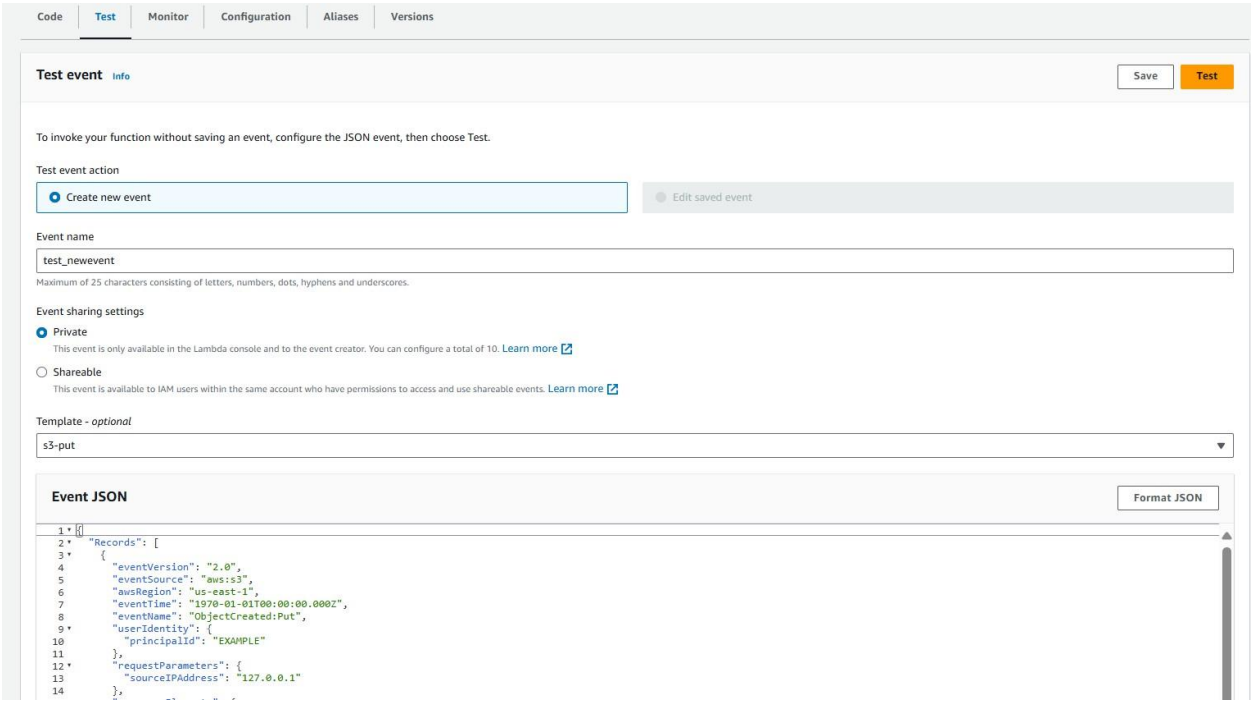
7. To Edit the basic settings go to configuration then click on edit setting.



8. Here, enter a description which is optional and change Memory and Timeout.
I've changed the Timeout period to 2 sec.

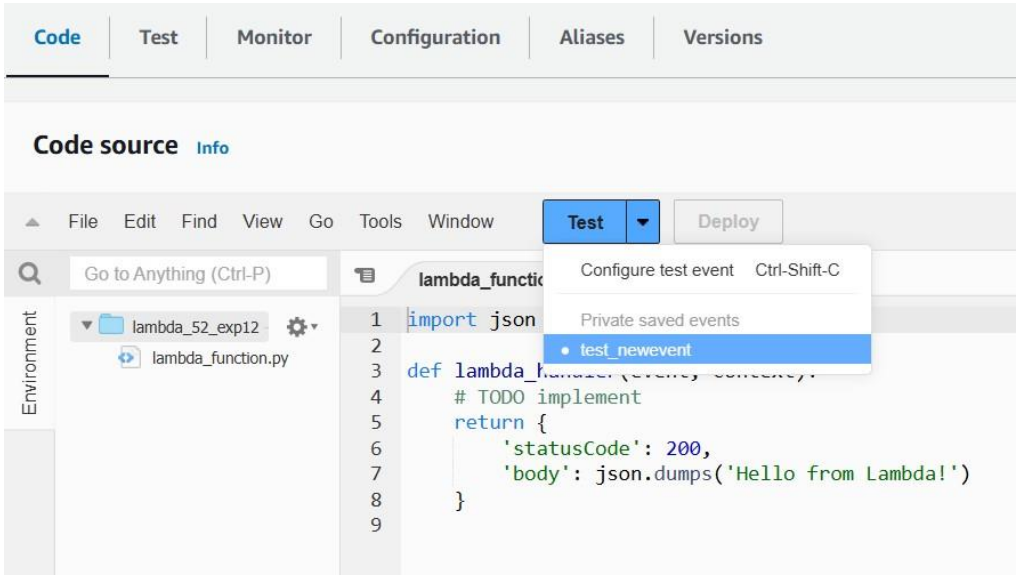


9. Now Click on the Test then select Create a new event, give a name to the event. Here I have given name as ‘test_newevent’ and then select Event Sharing to private, and select s3 put template.

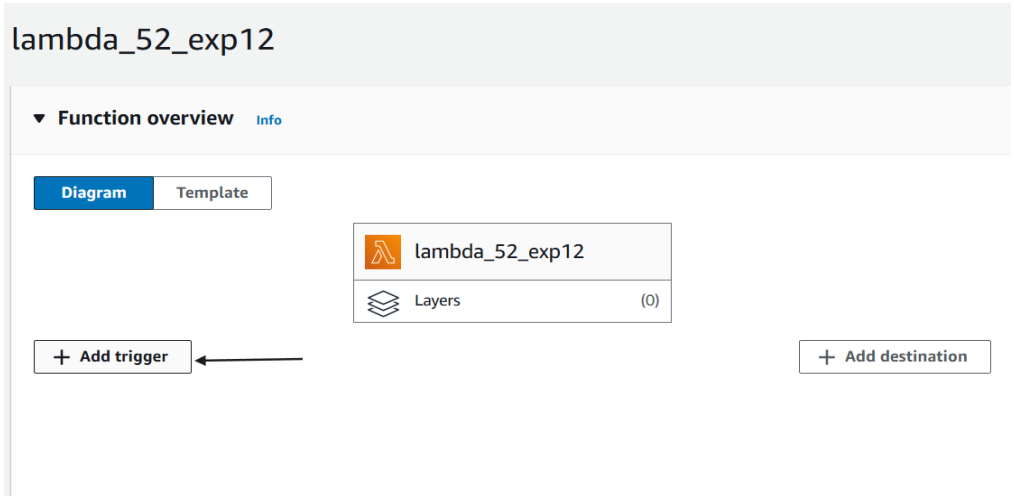


10. Now go to the Code section. Then click on the Test dropdown icon and select the event which

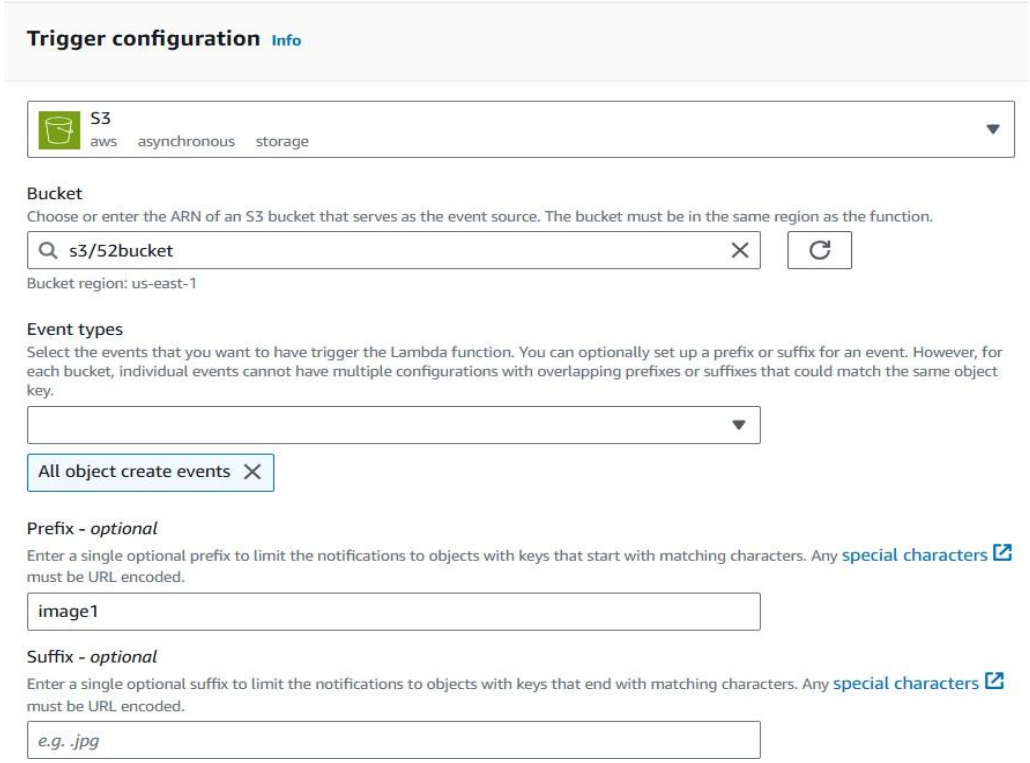
we have created now(‘test_newevent’).



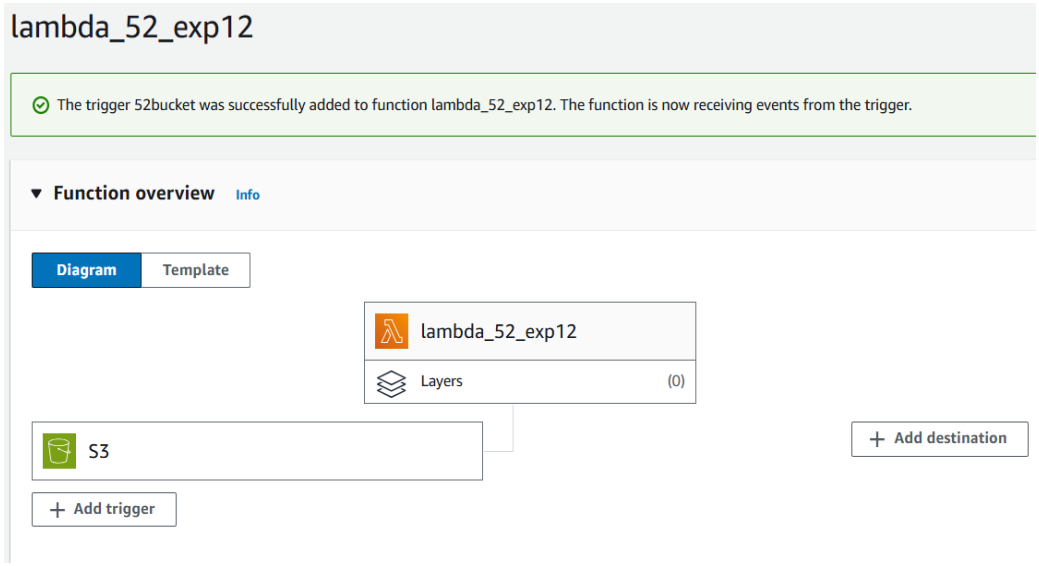
11. Now go into the Lambda function and then click on add tigger.



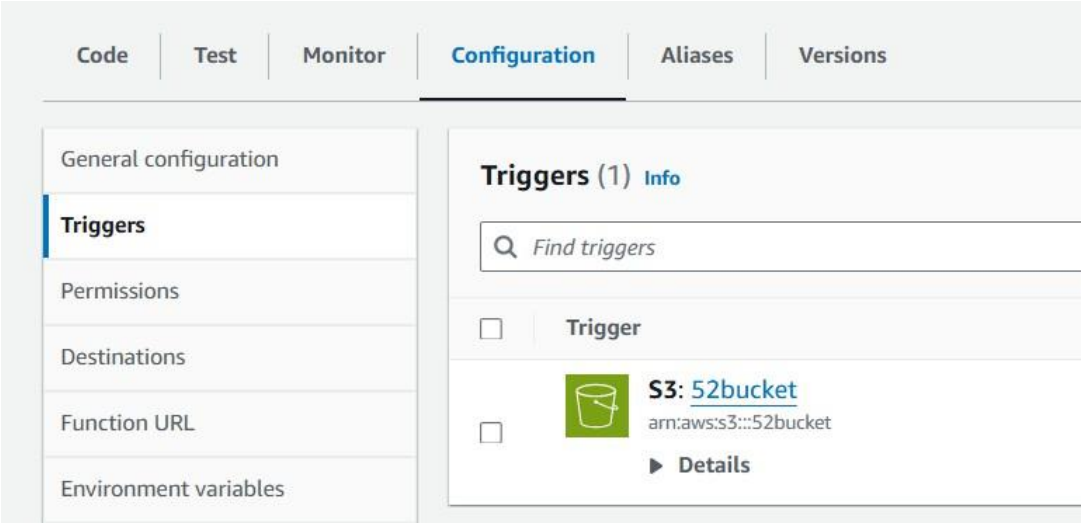
12. Now in the Trigger information. Select the source as S3.Then select the bucket which we have created now (52bucket{in my case}), keep other things default and also you can add prefix to image.



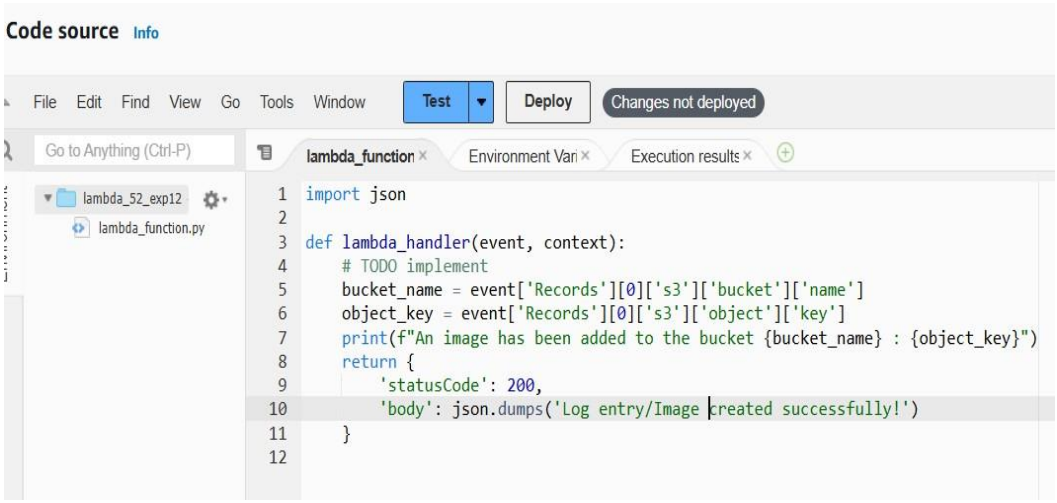
13. Thus, **trigger** is created successfully.



14. You can also check it in the configuration section.

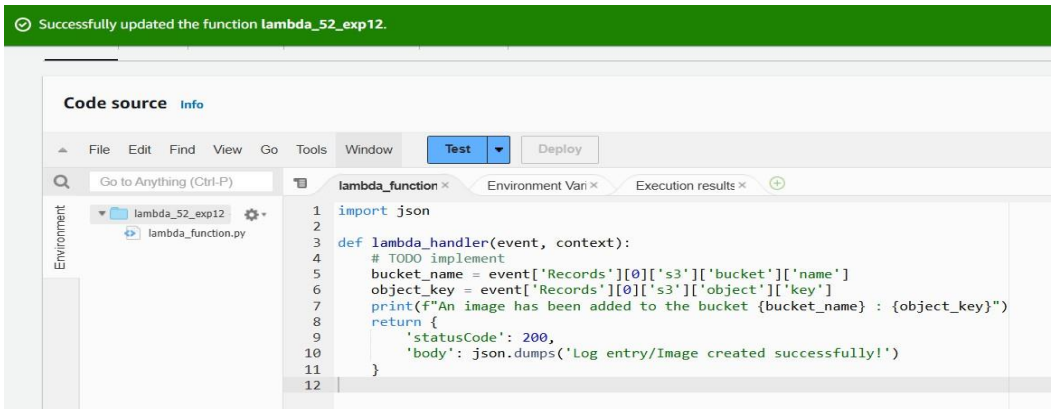


15. Now write a code which logs a message "Log entry/image created successfully" when triggered.

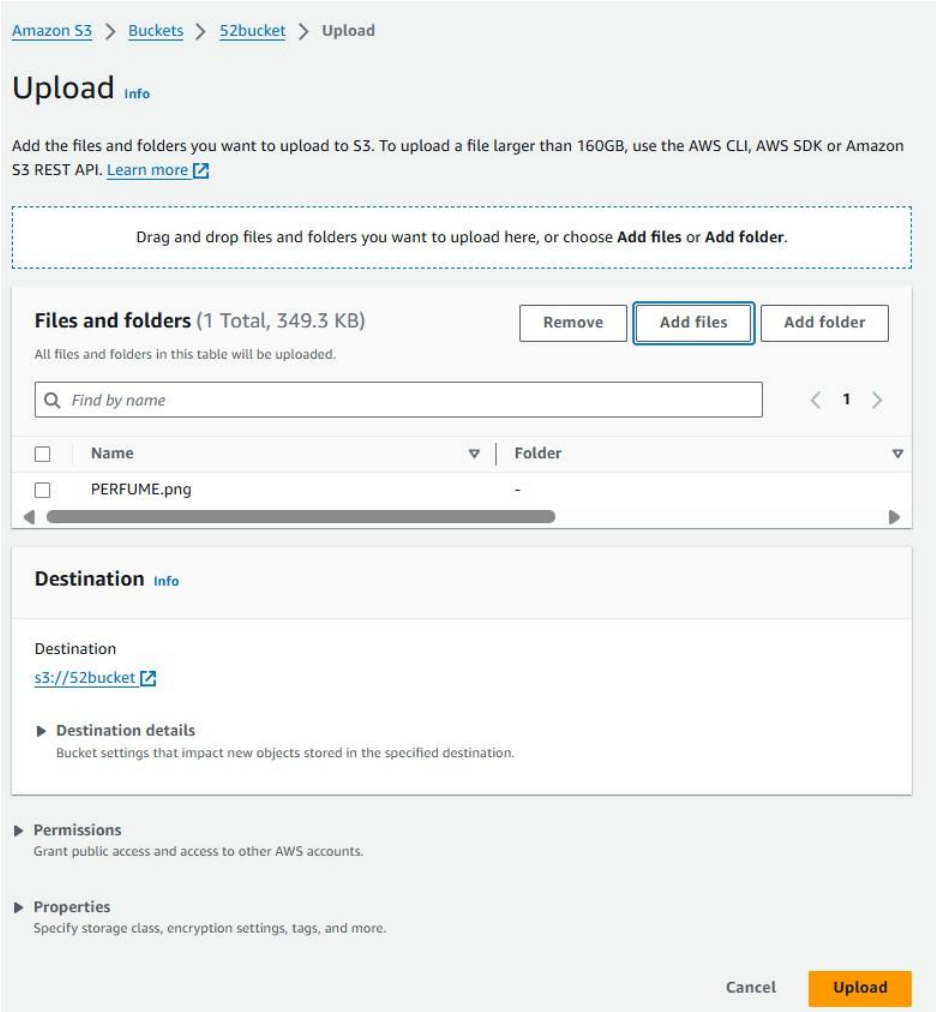


Here changes are not deployed.

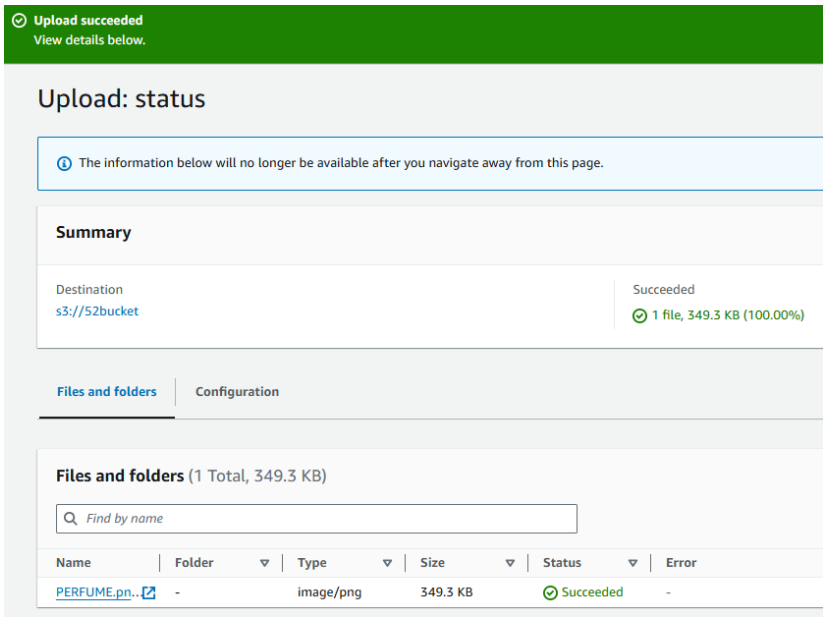
16. So now, Save the file by ctrl+s and then click on deploy.



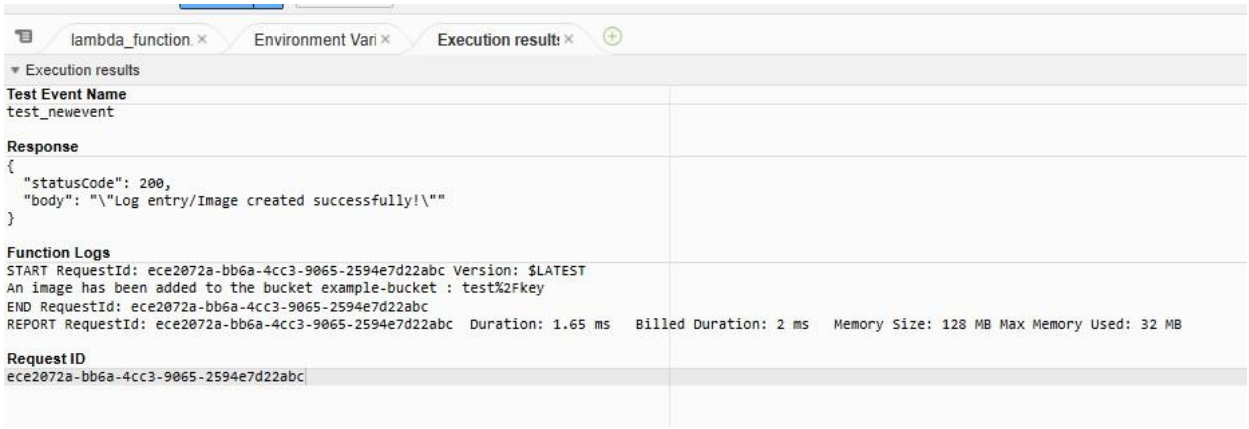
17. Go to S3 bucket,and there upload **any image** to the bucket.



18. Thus the image was uploaded successfully

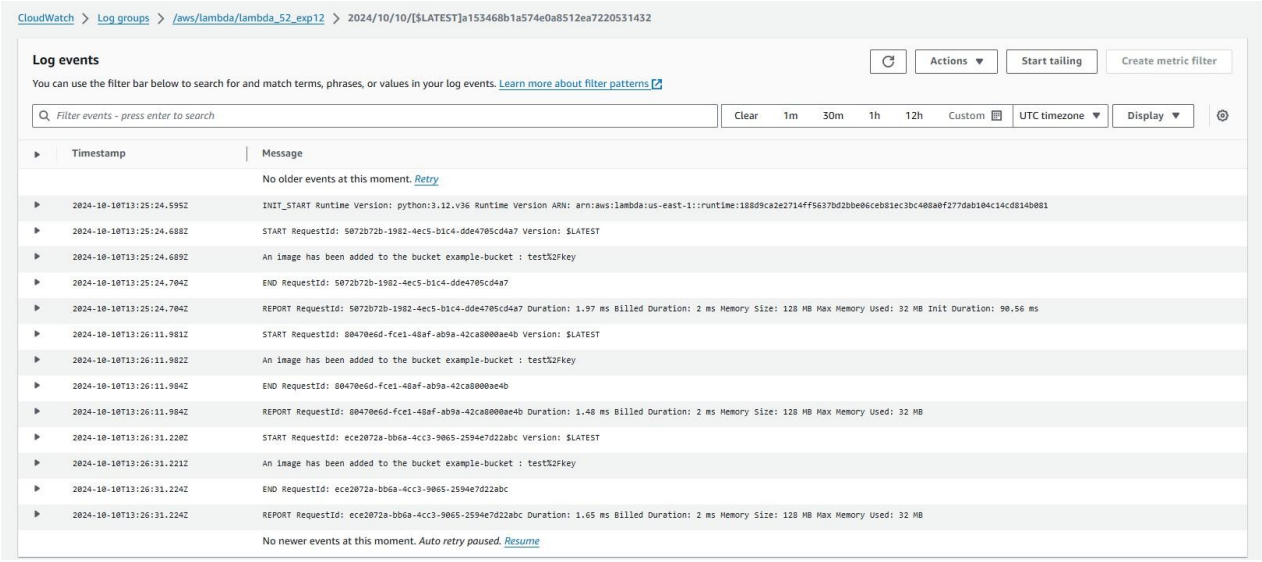


19. Now go to lambda function. Then click on test. This will give you log about the image that we have uploaded in S3 bucket.



(In response, It gives status 200 and also the message "Log entry/image created successfully" and also contains function Logs)

20. Now go to cloudwatch. Then go into log groups. Inside that you will get the lambda function name that we have created click on it. Here, you will get a detailed log of events.



Conclusion:

Through this project, I successfully set up a Lambda function and an S3 bucket. After configuring the settings, like adding a description and setting a 1-second timeout, I created a test event named 'test_newevent' and deployed the function without any issues. I also linked the Lambda function to the S3 bucket via a trigger and added a print statement in the code. After uploading an image to the bucket and redeploying, the function returned a status code of 200 with the expected message, and CloudWatch logs captured the entire process.This practical taught me how to effectively manage AWS Lambda settings, set up and test triggers, and monitor logs using CloudWatch. I also gained a deeper understanding of integrating S3 with Lambda to automate tasks.