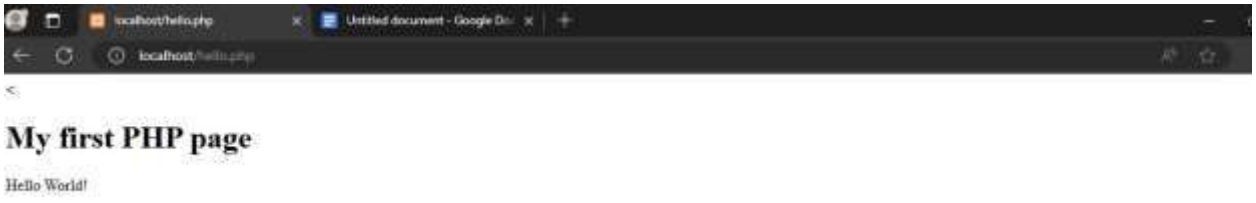
Name: Ganesh Gupta Div: D15C Roll No:13

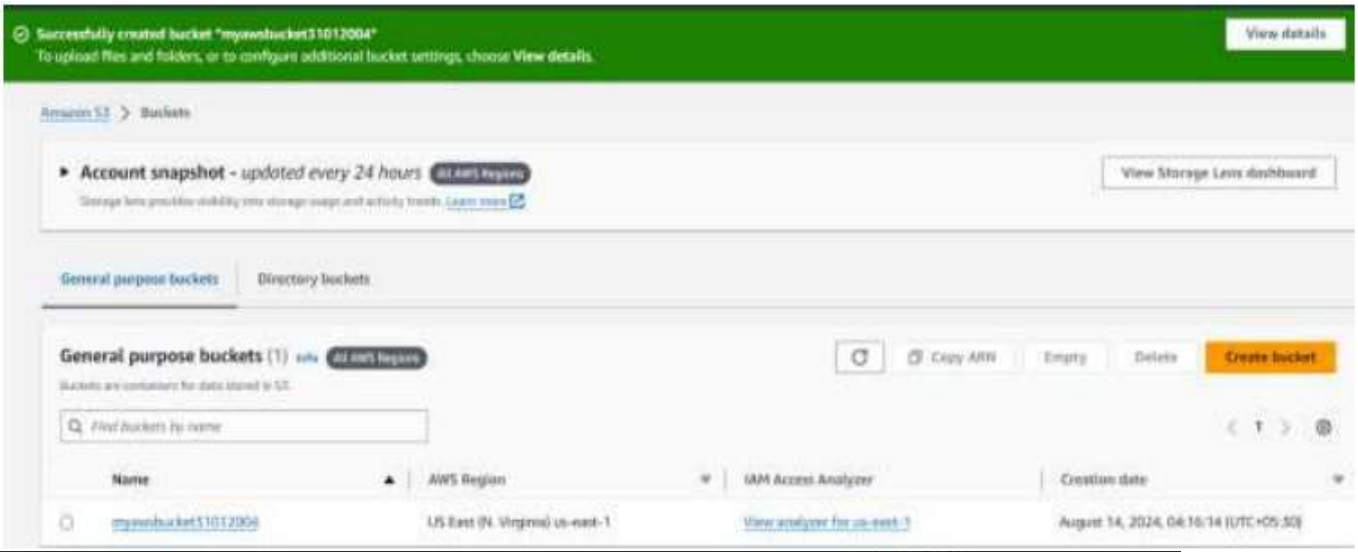
Exp 1a : Static Hosting

a) Hosting of a PHP file on Local virtual machine using Xampp



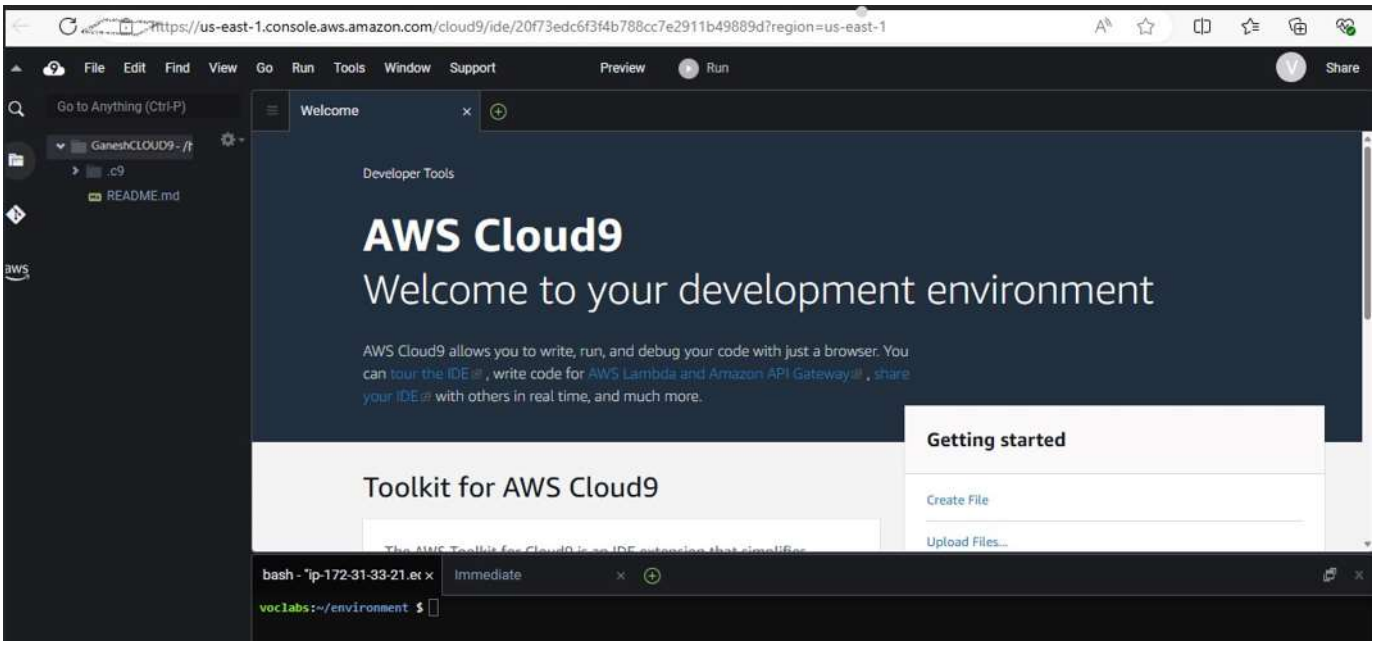
Name: Ganesh Gupta Div: D15C Roll No:13

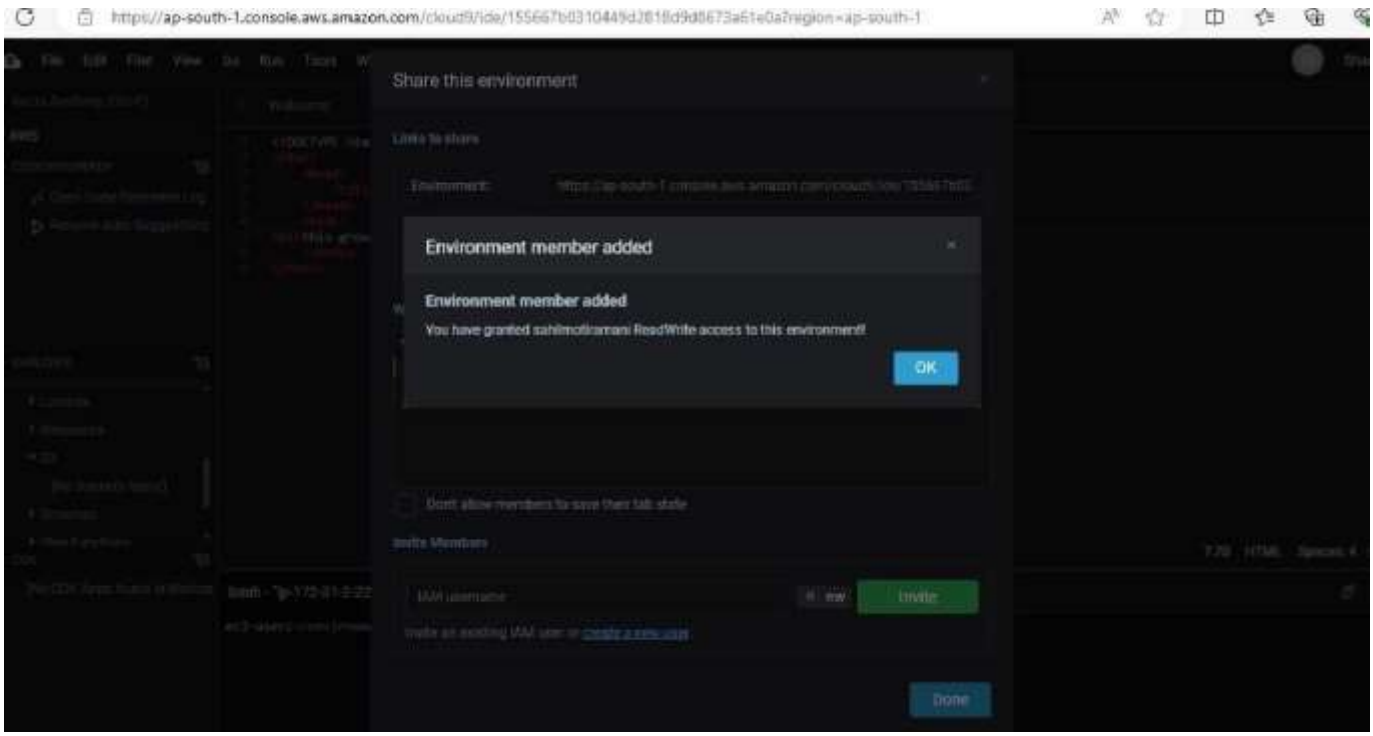
b) Static hosting using AWS S3 bucket



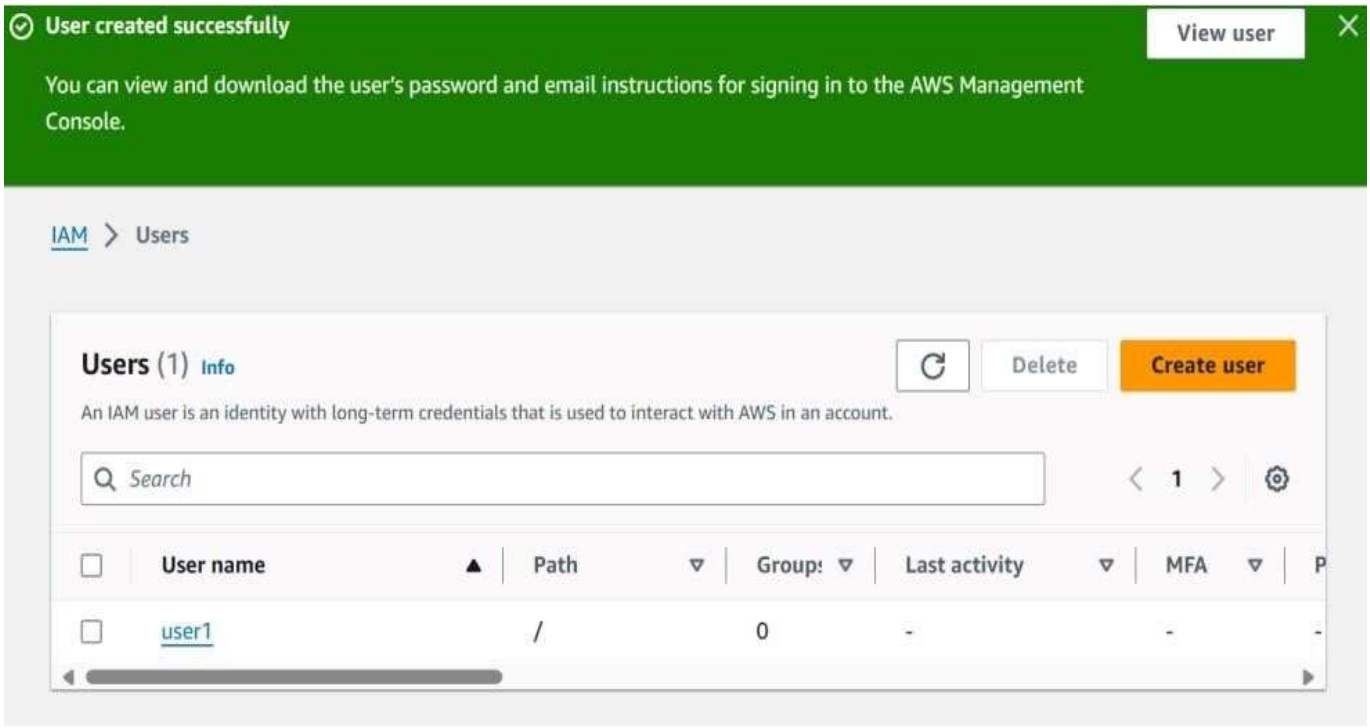
Experiment No. 1B

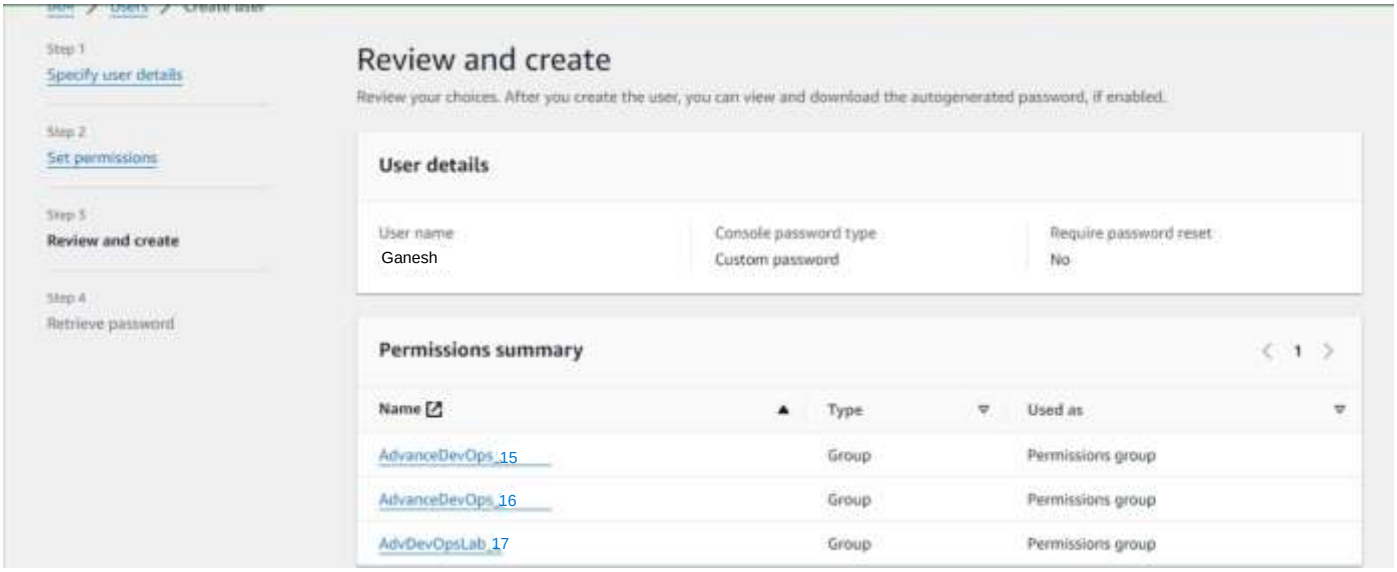
CLOUD 9 Environment:





IAM user and User Group:


User created successfully
View user
You can view and download the user's password and email instructions for signing in to the AWS Management Console.
IAM > Users
Users (1) Info
Delete
Create user
An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.
Search
1
User name
Path
Group:
Last activity
MFA
user1
/
0
-
-
-


Step 1
Specify user details
Step 2
Set permissions
Step 3
Review and create
Step 4
Retrieve password
Review and create
Review your choices. After you create the user, you can view and download the autogenerated password, if enabled.
User details
User name
Ganesh
Console password type
Custom password
Require password reset
No
Permissions summary
1
Name
Type
Used as
AdvanceDevOps_15
Group
Permissions group
AdvanceDevOps_16
Group
Permissions group
AdvDevOpsLab_17
Group
Permissions group

Experiment No 2

Step 1: Choose pipeline settings

Pipeline settings

Pipeline name
ganeshpipeline

Pipeline type
V2

Execution mode
QUEUED

Artifact location
codepipeline-us-east-1-602624750893

Service role name
AWSCodePipelineServiceRole-us-east-1-ganeshpipeline

Source Succeeded
Pipeline execution ID: a23b429a-7039-444c-8386-b03ba43a425f

Source
GitHub (Version 2)
Succeeded - 8 minutes ago
da19c44a

da19c44a Source: Update README.md

Disable transition

Deploy Succeeded
Pipeline execution ID: a23b429a-7039-444c-8386-b03ba43a425f

Deploy
AWS Elastic Beanstalk
Succeeded - 7 minutes ago

da19c44a Source: Update README.md

Congratulations!

You have successfully created a pipeline that retrieved this source application from an Amazon S3 bucket and deployed it to three Amazon EC2 instances using AWS CodeDeploy.

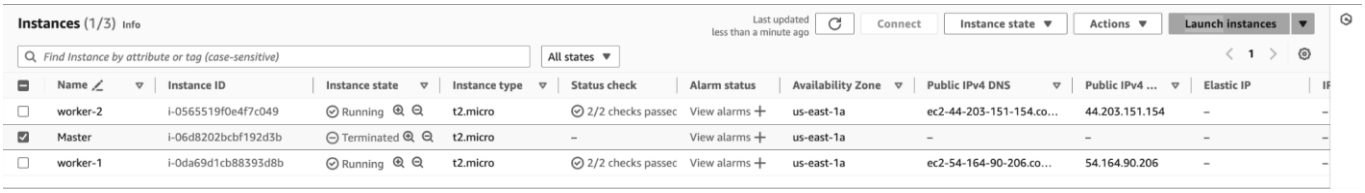
For next steps, read the AWS CodePipeline Documentation.

Advanced DevOps Lab
Experiment:3

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

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 **ssh -i <keyname>.pem**

ubuntu@<public_ip_address>

```
The authenticity of host 'ec2-44-203-151-154.compute-1.amazonaws.com (44.203.151.154)' can't be established.
ED25519 key fingerprint is SHA256:4cidwEvWyyoqWE0gGMsqDMjX2SlxkZVTUTIbDzMDdlc.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-44-203-151-154.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-31-95-91 ~]$ ssh -i "newkey.pem" ec2-user@ec2-54-164-90-206.compute-1.amazonaws.com
Warning: Identity file newkey.pem not accessible: No such file or directory.
The authenticity of host 'ec2-54-164-90-206.compute-1.amazonaws.com (172.31.88.50)' can't be established.
ED25519 key fingerprint is SHA256:RRoSz1NvNq9JLCAJhDKUn6FiRCRul+VtNbkijVO5M/I.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-54-164-90-206.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
ec2-user@ec2-54-164-90-206.compute-1.amazonaws.com: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
[ec2-user@ip-172-31-95-91 ~]$
```

4. From now on, until mentioned, perform these steps on all 3 machines.
   sudo yum install docker -y

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

Transaction Summary
```

Then, configure cgroup in a daemon.json file by using following commands

- 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
- docker -v

Install Kubernetes on all 3 machines

SELinux needs to be disable before configuring kubelet

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

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

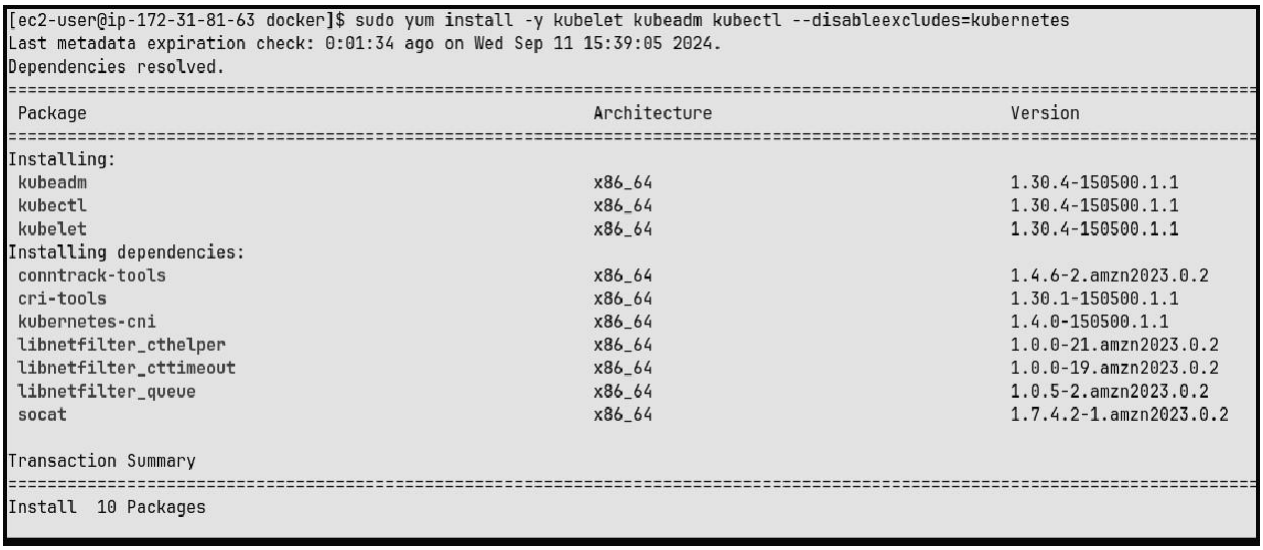
Add kubernetes repository (paste in terminal)

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

Type following commands:

- sudo yum update

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



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

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

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

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

1. Perform this ONLY on the Master machine

Initialize kubernetes by typing below command

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

Copy the mkdir and chown commands from the top and execute them

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

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

My personal join key:

```
kubeadm join 172.31.92.157:6443 --token x4sw6q.sbckmhm5gkoubquv \
        --discovery-token-ca-cert-hash
sha256:24c005691fcab2260667ee43384d46afd4b2b27401e82c01550798a0d8f98950
```

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

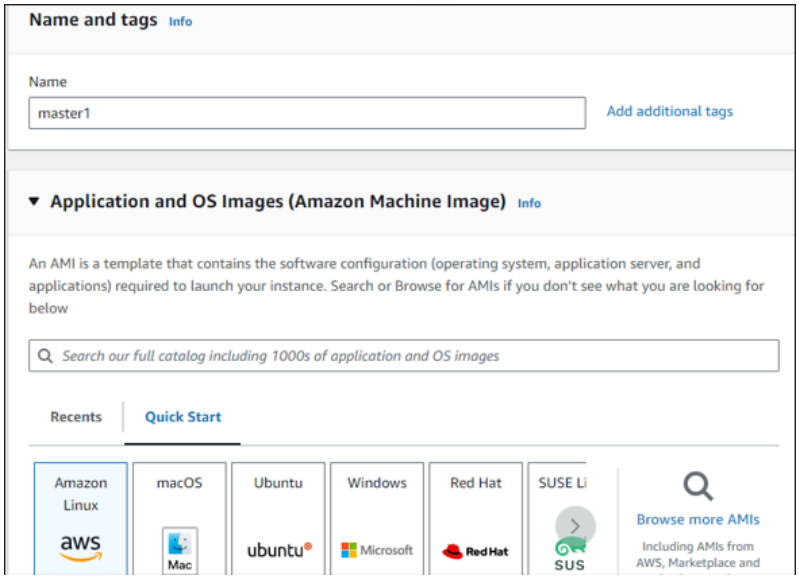
D15C
4

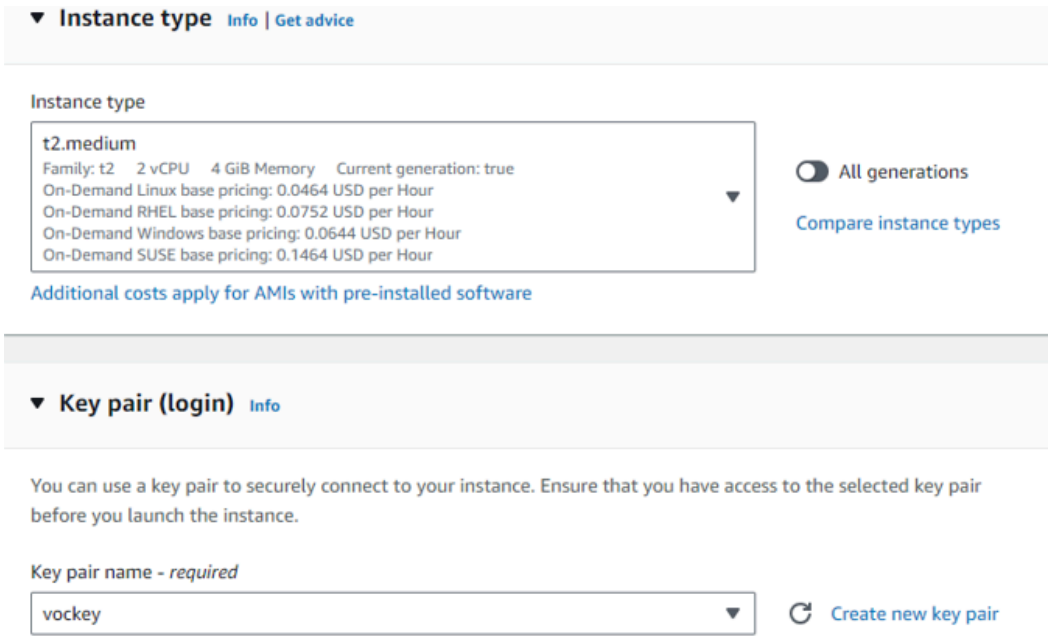
kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Document
at ion/kube-flannel.yml

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

Create 3 EC2 Ubuntu Instances on AWS.

Login to your AWS console.Go to services and in that search for EC2 and create 3 EC2 Ubuntu Instances as master 1 ,node1 and node 2.While making an instance make sure to select Amazon Linux and in linux type instead of default t2 .micro select t2.medium.





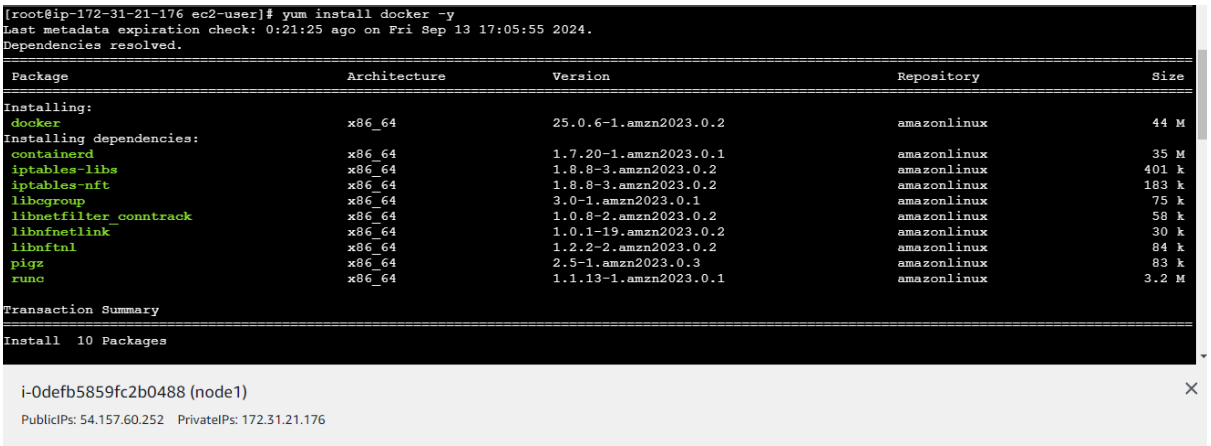
Setting SSH for establishing connections

INSTALLATION OF DOCKER

For installing docker we use the following steps:

STEP 1: In node 1 EC2 instance install docker and repeat the same step for master and node2 .

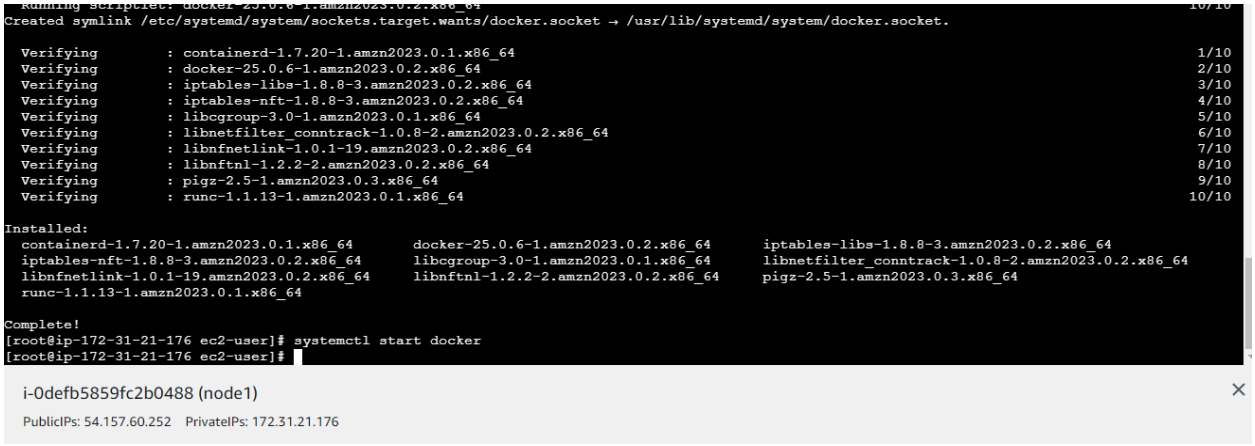
Syntax:yum install docker -y



STEP 2: After the installation of docker is successfully completed in all the three instances start the docker by the syntax given below:

Syntax :systemctl start docker.

Start the docker in master and node2 too .

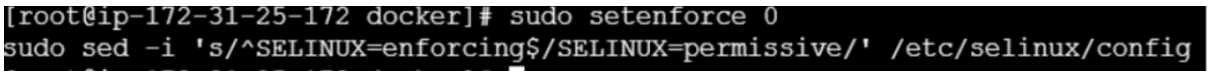


INSTALLATION OF KUBERNETES

After installing and starting the docker in all the three instances ,now lets install kubernetes for the installation we use the following steps:

STEP 1: Set SELinux to permissive mode:
Syntax:`sudo setenforce 0`

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



STEP 2: Add the Kubernetes yum repository. The exclude parameter in the repository definition ensures that the packages related to Kubernetes are not upgraded upon running yum update as there's a special procedure that must be followed for upgrading Kubernetes

```
[root@ip-172-31-21-176 ec2-user]# sudo su
[root@ip-172-31-21-176 ec2-user]# yum repolist
repo id                                                repo name
amazonlinux                                            Amazon Linux 2023 repository
kernel-livepatch                                       Amazon Linux 2023 Kernel Livepatch repository
[root@ip-172-31-21-176 ec2-user]# cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
[root@ip-172-31-21-176 ec2-user]#
```

i-0defb5859fc2b0488 (node1)

PublicIPs: 54.157.60.252 PrivateIPs: 172.31.21.176

STEP 3: Install kubelet, kubeadm and kubectl:

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

```
Last login: Fri Sep 13 17:58:28 2024 from 18.206.107.27
[ec2-user@ip-172-31-21-176 ~]$ sudo su
[root@ip-172-31-21-176 ec2-user]# sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
Kubernetes                                                                    60 kB/s | 9.4 kB     00:00
Dependencies resolved.
================================================================================
 Package                  Architecture  Version                 Repository    Size
================================================================================
Installing:
 kubeadm                  x86_64        1.31.1-150500.1.1       kubernetes    11 M
 kubectl                  x86_64        1.31.1-150500.1.1       kubernetes    11 M
 kubelet                  x86_64        1.31.1-150500.1.1       kubernetes    15 M
Installing dependencies:
 conntrack-tools          x86_64        1.4.6-2.amzn2023.0.2    amazonlinux  208 k
 cri-tools                x86_64        1.31.1-150500.1.1       kubernetes   6.9 M
 kubernetes-cni           x86_64        1.5.1-150500.1.1        kubernetes   7.1 M
 libnetfilter_cthelper    x86_64        1.0.0-21.amzn2023.0.2   amazonlinux   24 k
 libnetfilter_cttimeout   x86_64        1.0.0-19.amzn2023.0.2   amazonlinux   24 k
 libnetfilter_queue       x86_64        1.0.5-2.amzn2023.0.2    amazonlinux   30 k

Transaction Summary
================================================================================
Install  9 Packages
```

STEP 4: Enable the kubelet service before running kubeadm:

Syntax:`sudo systemctl enable --now kubelet`

```
[root@ip-172-31-21-176 ec2-user]# sudo systemctl enable --now kubelet
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service → /usr/lib/systemd/system/kubelet.service.
[root@ip-172-31-21-176 ec2-user]#
```

i-0defb5859fc2b0488 (node1)

PublicIPs: 54.157.60.252 PrivateIPs: 172.31.21.176

STEP 5: It can be seen from the repolist command which lists all the repository we can see that kubernetes in installed repeat all these steps on master1 and node2.

```
[root@ip-172-31-21-176 ec2-user]# yum repolist
repo id                                                repo name
amazonlinux                                            Amazon Linux 2023 repository
kernel-livepatch                                       Amazon Linux 2023 Kernel Livepatch repository
kubernetes                                             Kubernetes
[root@ip-172-31-21-176 ec2-user]#
```

i-0defb5859fc2b0488 (node1)

PublicIPs: 54.157.60.252 PrivateIPs: 172.31.21.176

STEP 6 : This command disable swap space and configure the system to use iptables for bridged network traffic, then apply these settings.

Syntax:`sudo swapoff -a`

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

```
[root@ip-172-31-16-56 ec2-user]# sudo swapoff -a
echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf
sudo sysctl -p
net.bridge.bridge-nf-call-iptables=1
```

STEP 7: Initialize Kubernetes in master instance .

```
Syntax: kubeadm init
```

```
[root@ip-172-31-16-56 ec2-user]# kubeadm init
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
        [WARNING FileExisting-socat]: socat not found in system path
        [WARNING FileExisting-tc]: tc not found in system path
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action beforehand using 'kubeadm config images pull'
W0913 18:58:26.902514   34809 checks.go:846] detected that the sandbox image "registry.k8s.io/pause:3.8" of the container runtime is inconsistent
 with that used by kubeadm.It is recommended to use "registry.k8s.io/pause:3.10" as the CRI sandbox image.
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [ip-172-31-16-56.ec2.internal kubernetes kubernetes.default kubernetes.default.svc kuberne
tes.default.svc.cluster.local] and IPs [10.96.0.1 172.31.16.56]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [ip-172-31-16-56.ec2.internal localhost] and IPs [172.31.16.56 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [ip-172-31-16-56.ec2.internal localhost] and IPs [172.31.16.56 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
```

i-0ddf50a232db19957 (master1)

PublicIPs: 3.88.204.138 PrivateIPs: 172.31.16.56

```
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 172.31.16.56:6443 --token oghyi3.fnspdro8pevgr0d5 \
        --discovery-token-ca-cert-hash sha256:ec71ffc0d9fd79263fb8909d938da8d29e5f15a21ab5e0a17ec93514e8c4ecb8
```

Use the mkdir and chown commands shown above

```
[root@ip-172-31-16-56 ec2-user]#  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Add a common networking plugin called flannel
Syntax:`kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml`

```
[root@ip-172-31-16-56 ~]# 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
```

STEP 8: Apply deployment of nginx server using the following command.
Syntax:
`kubectl apply -f https://k8s.io/examples/application/deployment.yaml`

```
[root@ip-172-31-16-56 ~]# kubectl apply -f https://k8s.io/examples/application/deployment.yaml

deployment.apps/nginx-deployment created
```

Check whether the pods is created or not by the following command
Syntax: kubectl get pods

```
[root@ip-172-31-16-56 ~]# kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-gw8v8    0/1     Pending   0          82s
nginx-deployment-d556bf558-rfk9n    0/1     Pending   0          82s
```

Kubectl describe pod nginx command describe the pods.

```
[root@ip-172-31-16-56 ~]# kubectl describe pod nginx
Name:             nginx-deployment-d556bf558-gw8v8
Namespace:        default
Priority:         0
Service Account:  default
Node:             <none>
Labels:           app=nginx
                  pod-template-hash=d556bf558
Annotations:      <none>
Status:           Pending
IP:
IPs:              <none>
Controlled By:    ReplicaSet/nginx-deployment-d556bf558
Containers:
  nginx:
    Image:        nginx:1.14.2
    Port:         80/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:
```

```
Conditions:
  Type           Status
  PodScheduled   False
Volumes:
  kube-api-access-f9k9s:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason            Age    From               Message
  ----     ------            ----   ----               -------
  Warning  FailedScheduling  114s   default-scheduler  0/1 nodes are available: 1 node(s) had untolerated taint {node-role.kubernetes.io/control-
: }. preemption: 0/1 nodes are available: 1 Preemption is not helpful for scheduling.
  Warning  FailedScheduling  3m18s  default-scheduler  0/1 nodes are available: 1 node(s) had untolerated taint {node-role.kubernetes.io/control-
: }. preemption: 0/1 nodes are available: 1 Preemption is not helpful for scheduling.
```

```
[ec2-user@ip-172-31-26-174 ~]$ kubectl taint nodes --all node-role.kubernetes.io
/control-plane-
node/ip-172-31-26-174.ec2.internal untainted
```

STEP 9:Check whether the pods are running or not.

```
[ec2-user@ip-172-31-26-174 ~]$ kubectl get pods
NAME    READY   STATUS    RESTARTS      AGE
nginx   1/1     Running   1 (6s ago)   90s
```

STEP 10: Mention the port that you want to host on
Syntax:port-forward nginx 8081:80

```
[ec2-user@ip-172-31-26-174 ~]$ kubectl port-forward nginx 8081:80
Forwarding from 127.0.0.1:8081 -> 80
Forwarding from [::1]:8081 -> 80
```

STEP 11: Then verify your deployment

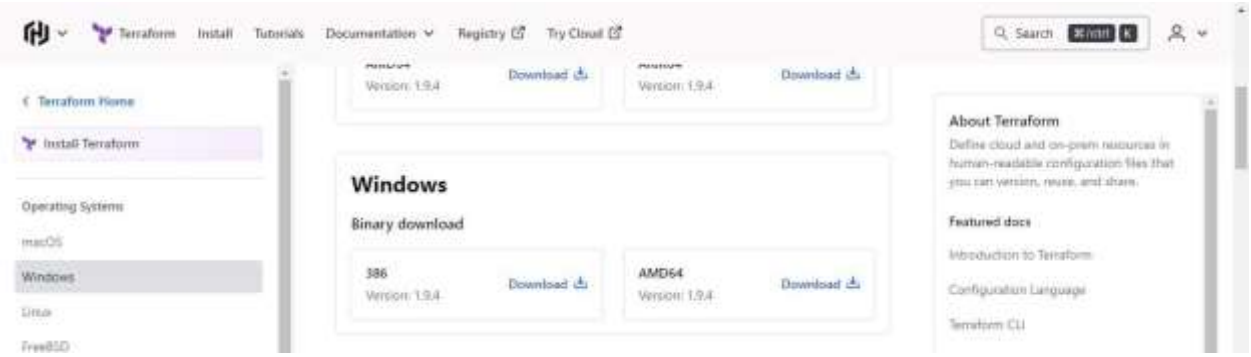
Conclusion : In this experiment we have created 3 EC2 instances. Setting SSH for establishing connections in that we have installed and started docker and kubernetes ,initialising kubernetes we use the mkdir and chown commands that we get by initializing the kubertenes then we add a common networking plugin called flannel then we apply deployment to nginx server we describe the pods and we check the status of the pods we mention the port that we want to host on and at the end we very the deployment of the kubernetes application by performing the following steps we learned to deploy the our Kubernetes Application.

Name:Ganesh Gupta D15C Roll No: 15

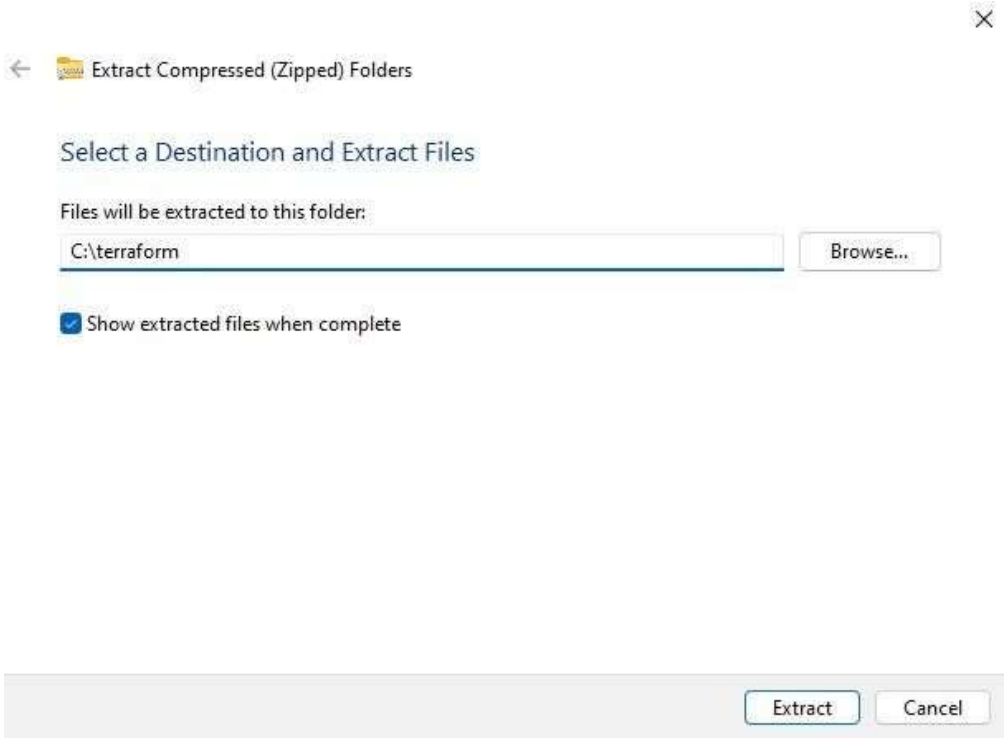
Experiment No. 5

Installation of Terraform

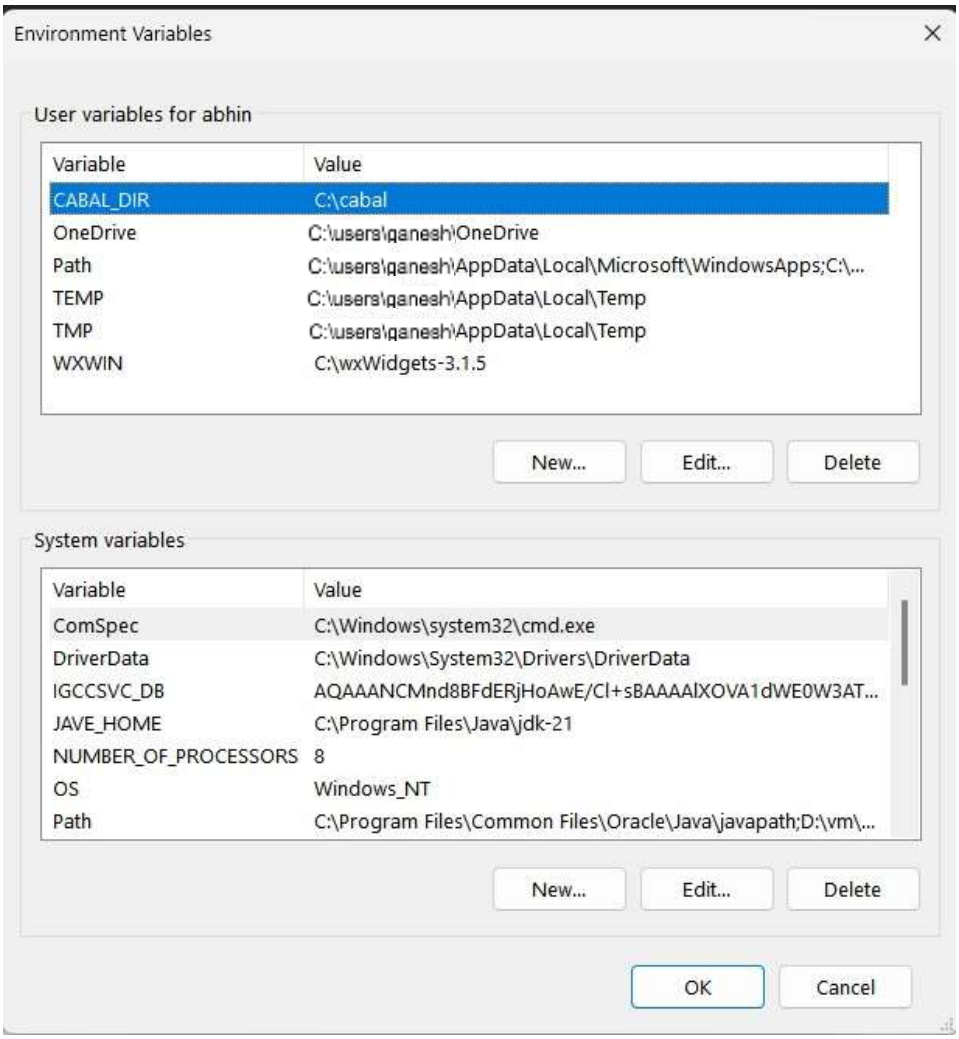
Step 1:To install Terraform, visit the official Terraform website mentioned below, go to the Downloads section, select Windows, and download the 64-bit version for your system. website:https://www.terraform.io/downloads.html



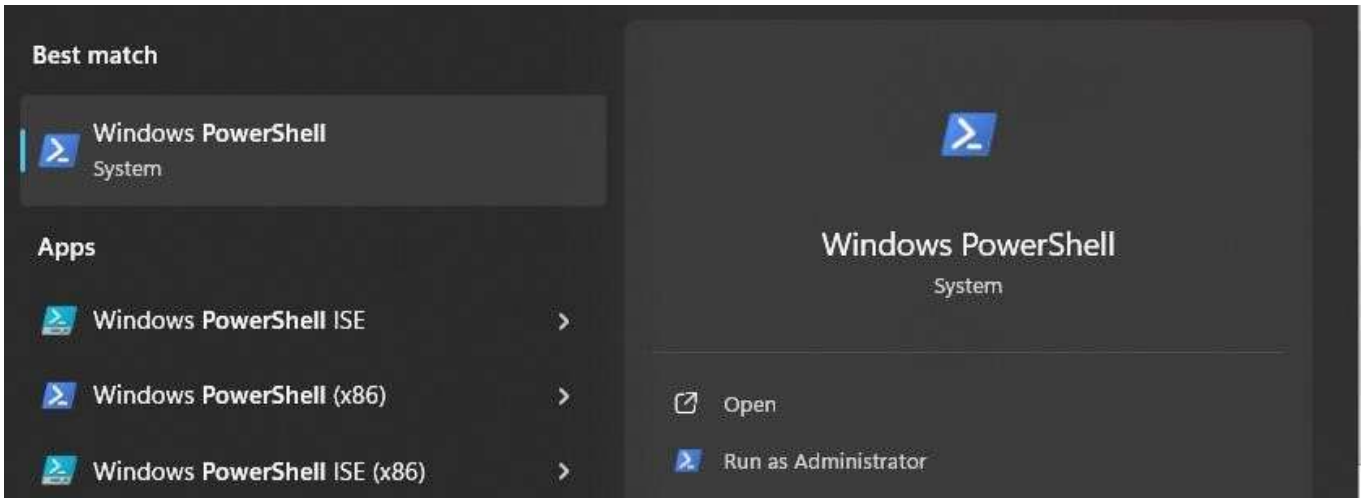
Step 2: Extract the downloaded `Terraform.exe` file to the `C:\Terraform` directory on your system.



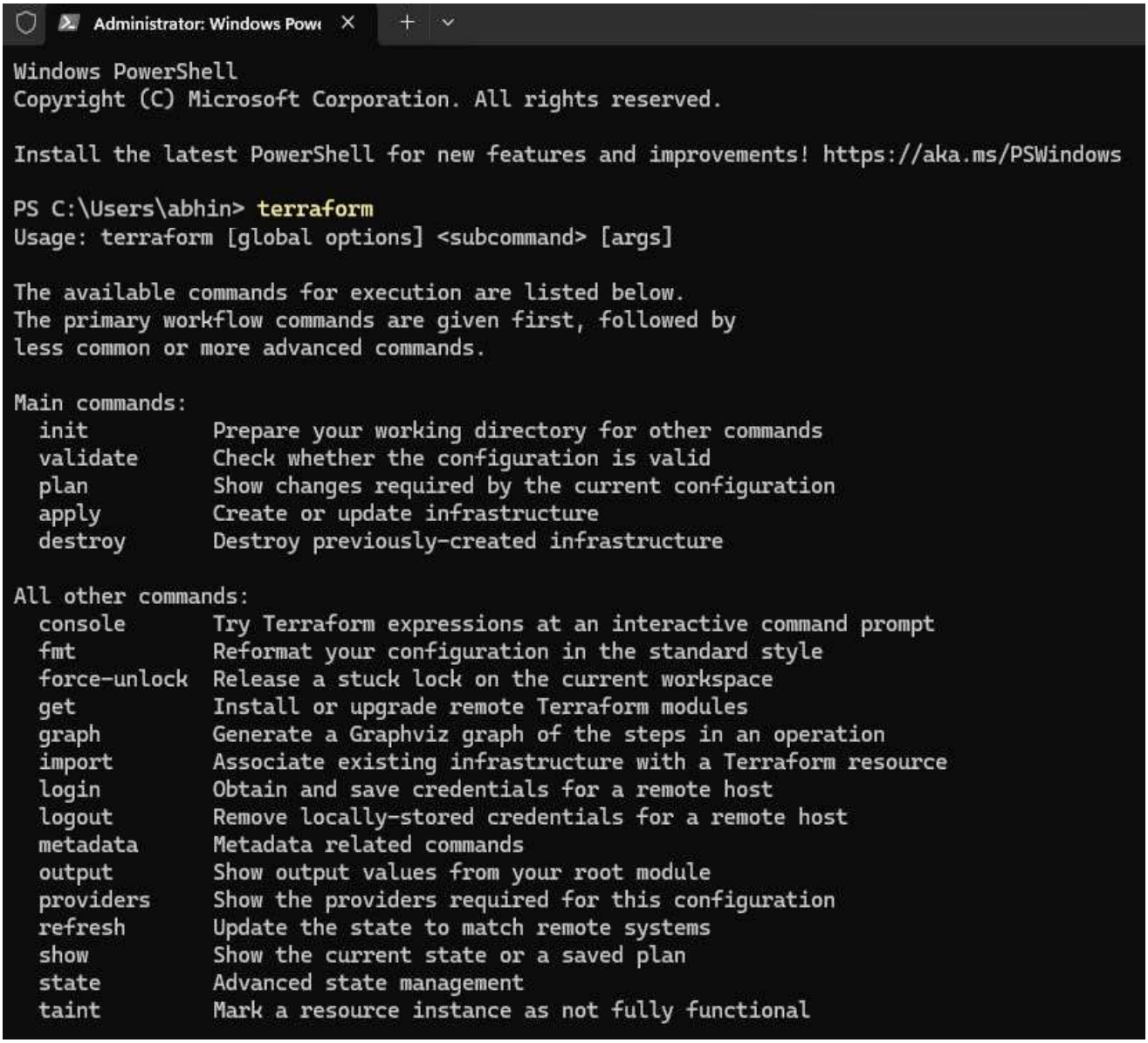
Step 3: Set the System path for Terraform in Environment Variables



Step 4: Run the windows powershell as administrator.



Step 5: Run "`terraform`" to verify its functionality. If you encounter any errors, double-check or update the Terraform path in your environment variables.



Name : Ganesh Gupta Div : D15C Roll no : 13

Creating docker image using terraform

Step 1: Download and install Docker Desktop by visiting https://www.docker.com. Run the installer and follow the prompts to complete the installation, then verify by launching Docker Desktop or using the `docker --version` command.

```
C:\Users\Admin>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
  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
```

```
C:\Users\Admin>docker --version
Docker version 27.1.1, build 6312585
```

Step 2: 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 3**:** First, create a new folder named `Docker` inside the `TerraformScripts` folder. Then, open Notepad and create a new file named `docker.tf` within the `Docker` folder. Write the following contents into the `docker.tf` file to create an Ubuntu Linux container. Save the file when done.

Script:

Name : Ganesh Gupta Div : D15C Roll no : 13

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

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

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

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

This Terraform script configures the Docker provider to communicate with the Docker Engine using a Windows named pipe.
It pulls the latest Ubuntu image from Docker Hub and creates a container named "foo."
The container runs the `sleep infinity` command, which keeps it active indefinitely.
This setup is useful for scenarios where the container needs to remain running continuously.

Name : Ganesh Gupta Div : D15C Roll no : 13

docker.tf

File Edit View

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

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

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

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

Step 4: Execute the `terraform init` command to initialize the working directory, download the necessary provider plugins, and set up the backend for managing Terraform state.

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

Terraform has been successfully initialized!

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

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
PS C:\Users\Admin\Desktop\Terraformscripts\docker>
```

Name : Ganesh Gupta Div : D15C Roll no : 13

Step 5: Run `**terraform plan**` to preview the actions Terraform will take to reach the desired state defined in your configuration, including creating, modifying, or deleting resources.

```
PS C:\Users\Admin\Desktop\Terraformscripts\docker> terraform plan

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

Terraform will perform the following actions:

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

```
      + read_only        = false
      + remove_volumes   = true
      + restart          = "no"
      + rm               = false
      + runtime          = (known after apply)
      + security_opts    = (known after apply)
      + shm_size         = (known after apply)
      + start            = true
      + stdin_open       = false
      + stop_signal      = (known after apply)
      + stop_timeout     = (known after apply)
      + tty              = false

      + healthcheck (known after apply)

      + labels (known after apply)
    }

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

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

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply"
now.
PS C:\Users\Admin\Desktop\Terraformscripts\docker>
```

Name : Ganesh Gupta Div : D15C Roll no : 13

Step 6: Execute "**terraform apply**" to apply the configuration, which will automatically create and run the Ubuntu container based on our configuration.

```
PS C:\Users\Admin\Desktop\Terraformscripts\docker> terraform apply

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

Terraform will perform the following actions:

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

      + healthcheck (known after apply)

      + labels (known after apply)
    }

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

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

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

  Enter a value: yes

docker_image.ubuntu: Creating...
docker_image.ubuntu: Still creating... [10s elapsed]
docker_image.ubuntu: Still creating... [20s elapsed]
docker_image.ubuntu: Still creating... [30s elapsed]
docker_image.ubuntu: Creation complete after 37s [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Creating...
docker_container.foo: Creation complete after 2s [id=76c6390ec277dc11f709997c4eb636ee8b45f90723d0f6ec07d85caeda18ead9]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
PS C:\Users\Admin\Desktop\Terraformscripts\docker>
```

Name : Ganesh Gupta Div : D15C Roll no : 13

Step 7: The command `**docker images**` lists all Docker images stored locally on your system, showing details like repository names, tags, image IDs, and creation dates.

Docker images, Before Executing Apply step:

```
PS C:\Users\Admin\Desktop\Terraformscripts\docker> docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
```

Docker images, After Executing Apply step:

```
PS C:\Users\Admin\Desktop\Terraformscripts\docker> docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
ubuntu       latest    edbfe74c41f8   3 weeks ago   78.1MB
```

Step 8: Execute Terraform destroy to delete the configuration, which will automatically delete the Ubuntu Container.

```
PS C:\Users\Admin\Desktop\Terraformscripts\docker> terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Refreshing state... [id=76c6390ec277dc11f709997c4eb636ee8b45f90723d0f6ec07d85caeda18ead9]

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

Terraform will perform the following actions:

  # docker_container.foo will be destroyed
  - resource "docker_container" "foo" {
      - attach            = false -> null
      - command           = [
          - "sleep",
          - "infinity",
        ] -> null
      - cpu_shares        = 0 -> null
      - dns               = [] -> null
      - dns_opts          = [] -> null
      - dns_search        = [] -> null
      - entrypoint        = [] -> null
      - env               = [] -> null
      - gateway           = "172.17.0.1" -> null
      - group_add         = [] -> null
      - hostname          = "76c6390ec277" -> null
      - id                = "76c6390ec277dc11f709997c4eb636ee8b45f90723d0f6ec07d85caeda18ead9" -> null
      - image             = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a" -> null
      - init              = false -> null
      - ip_address        = "172.17.0.2" -> null
      - ip_prefix_length  = 16 -> null
      - ipc_mode          = "private" -> null
      - links             = [] -> null
      - log_driver        = "json-file" -> null
      - log_opts          = {} -> null
      - logs              = false -> null
      - max_retry_count   = 0 -> null
      - memory            = 0 -> null
      - memory_swap       = 0 -> null
      - must_run          = true -> null
      - name              = "foo" -> null
      - network_data      = [
          - {
              - gateway                   = "172.17.0.1"
              - global_ipv6_prefix_length = 0
              - ip_address                = "172.17.0.2"
              - ip_prefix_length          = 16
              - network_name              = "bridge"
                # (2 unchanged attributes hidden)
```

Name : Ganesh Gupta Div : D15C Roll no : 13

```
            - network_name             = "bridge"
              # (2 unchanged attributes hidden)
          },
      ] -> null
      - network_mode      = "bridge" -> null
      - privileged        = false -> null
      - publish_all_ports = false -> null
      - read_only         = false -> null
      - remove_volumes    = true -> null
      - restart           = "no" -> null
      - rm                = false -> null
      - runtime           = "runc" -> null
      - security_opts     = [] -> null
      - shm_size          = 64 -> null
      - start             = true -> null
      - stdin_open        = false -> null
      - stop_timeout      = 0 -> null
      - storage_opts      = {} -> null
      - sysctls           = {} -> null
      - tmpfs             = {} -> null
      - tty               = false -> null
        # (8 unchanged attributes hidden)
    }

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

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

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

  Enter a value: yes

docker_container.foo: Destroying... [id=76c6390ec277dc11f709997c4eb636ee8b45f90723d0f6ec07d85caeda18ead9]
docker_container.foo: Destruction complete after 1s
docker_image.ubuntu: Destroying... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_image.ubuntu: Destruction complete after 1s

Destroy complete! Resources: 2 destroyed.
PS C:\Users\Admin\Desktop\Terraformscripts\docker>
```

Step 9: Docker images After Executing Destroy step

```
PS C:\Users\Admin\Desktop\Terraformscripts\docker> docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
```

Name: Ganesh Gupta Div:D15C Roll No.13

Adv DevOps Practical 7

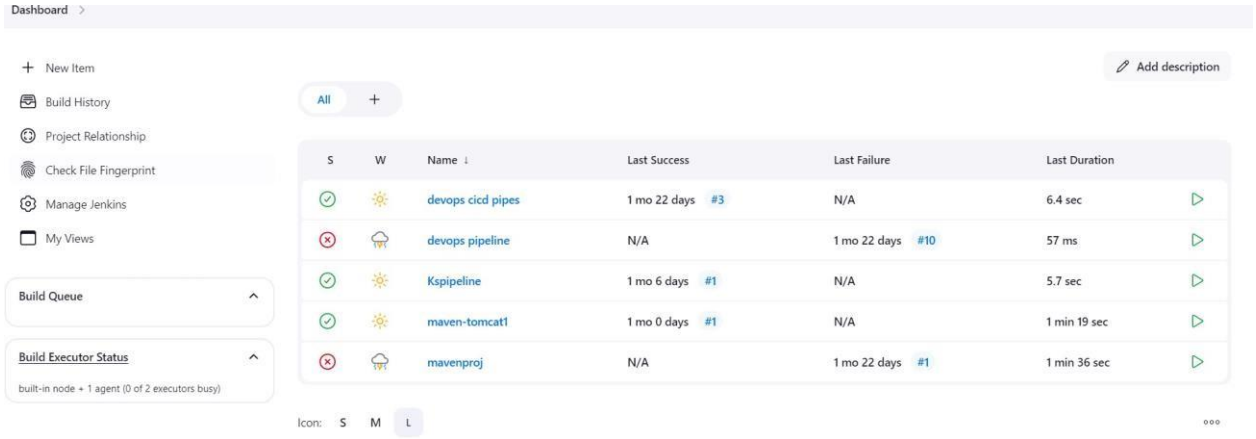
Aim: To understand Static Analysis SAST process and learn to integrate Jenkins SAST to SonarQube/GitLab.

Integrating Jenkins with SonarQube:

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

Steps to integrate Jenkins with SonarQube

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



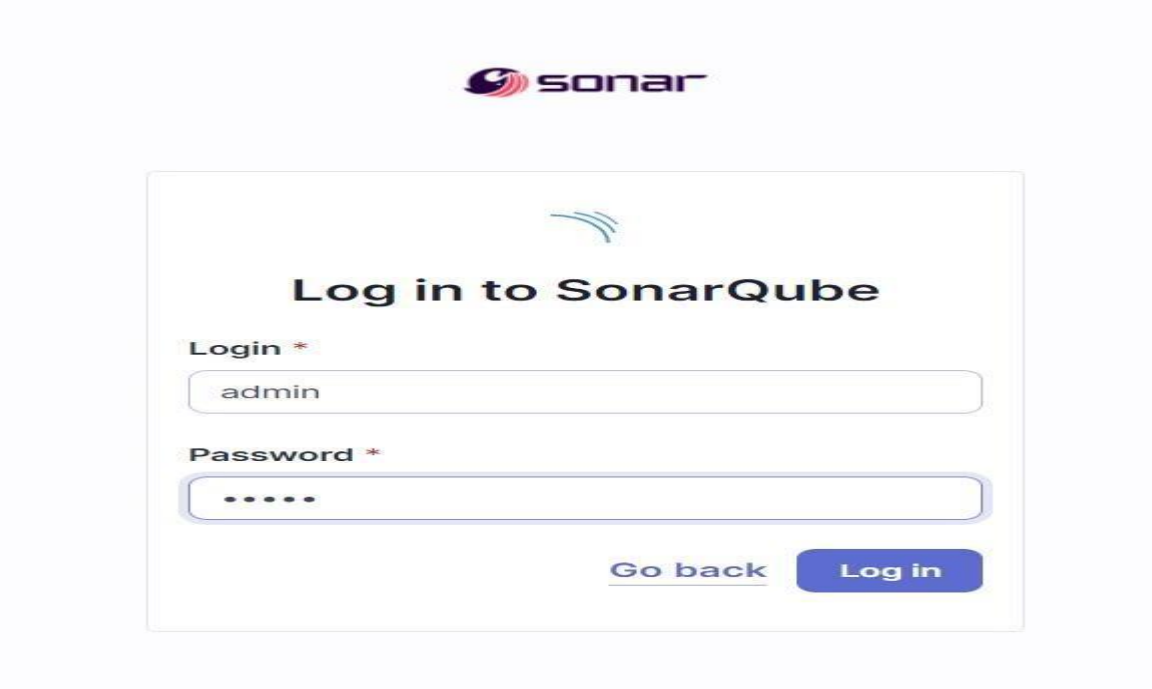
2. Run SonarQube in a Docker container using this command -

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

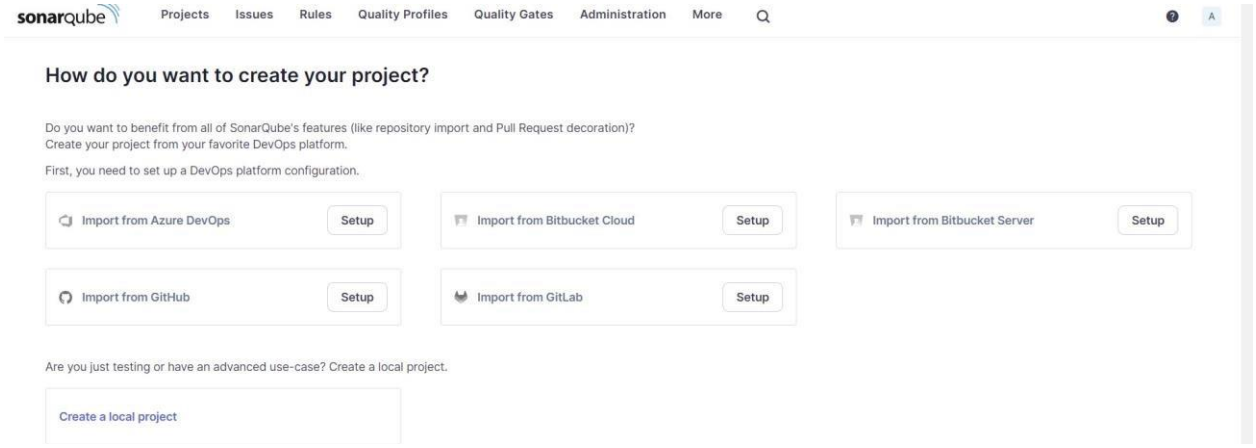
----------Warning: run below command only once

```
PS C:\Users\91773\Desktop\College Resources\Exp7 adv devops> docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS
_DISABLE=true -p 9000:9000 sonarqube:latest
Unable to find image 'sonarqube:latest' locally
latest: Pulling from library/sonarqube
7478e0ac0f23: Pull complete
90a925ab929a: Pull complete
7d9a34308537: Pull complete
80338217a4ab: Pull complete
1a5fd5c7e184: Pull complete
7b87d6fa783d: Pull complete
bd819c9b5ead: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:72e9feec71242af83faf65f95a40d5e3bb2822a6c3b2cda8568790f3d31aecde
Status: Downloaded newer image for sonarqube:latest
77e678cded2ef5f989912d3d9e6991dd548eac03faa1eed68dd906614be53acc
PS C:\Users\91773\Desktop\College Resources\Exp7 adv devops>
```

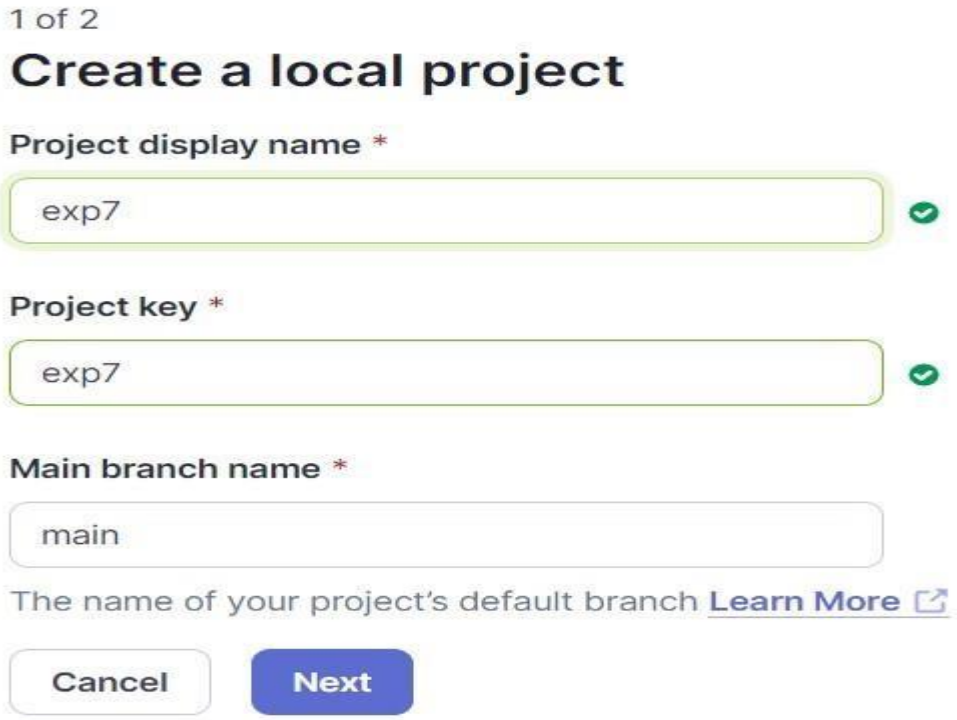
3. Once the container is up and running, you can check the status of SonarQube at localhost port 9000.



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

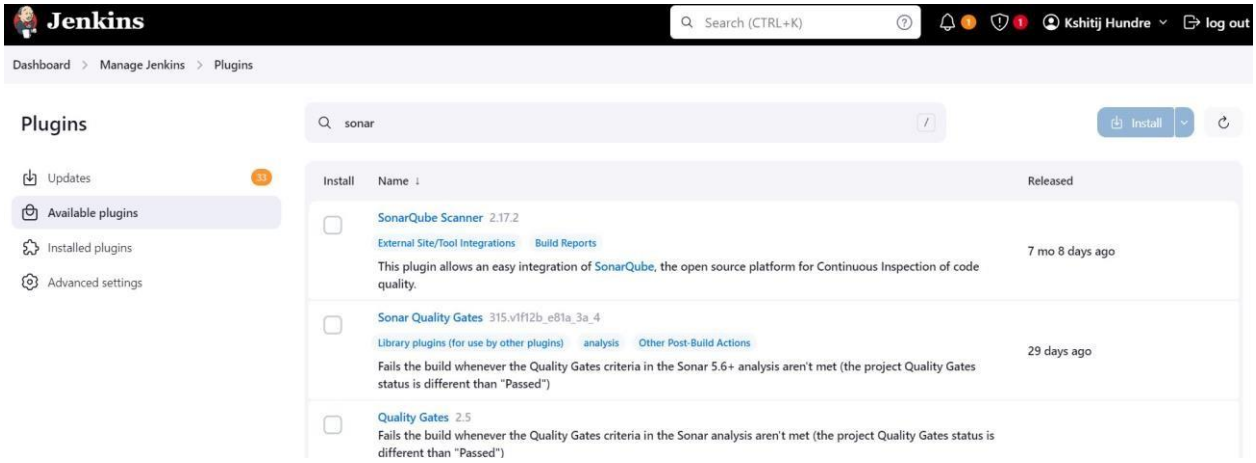


5. Create a manual project in SonarQube with the name sonarqube



Setup the project and come back to Jenkins Dashboard.

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



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

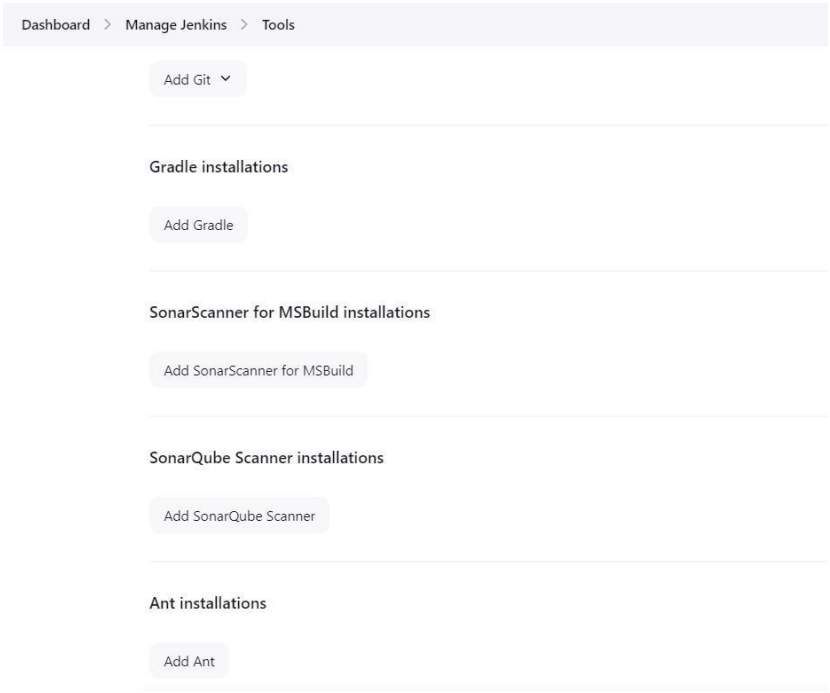
Enter the Server Authentication token if needed.

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

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

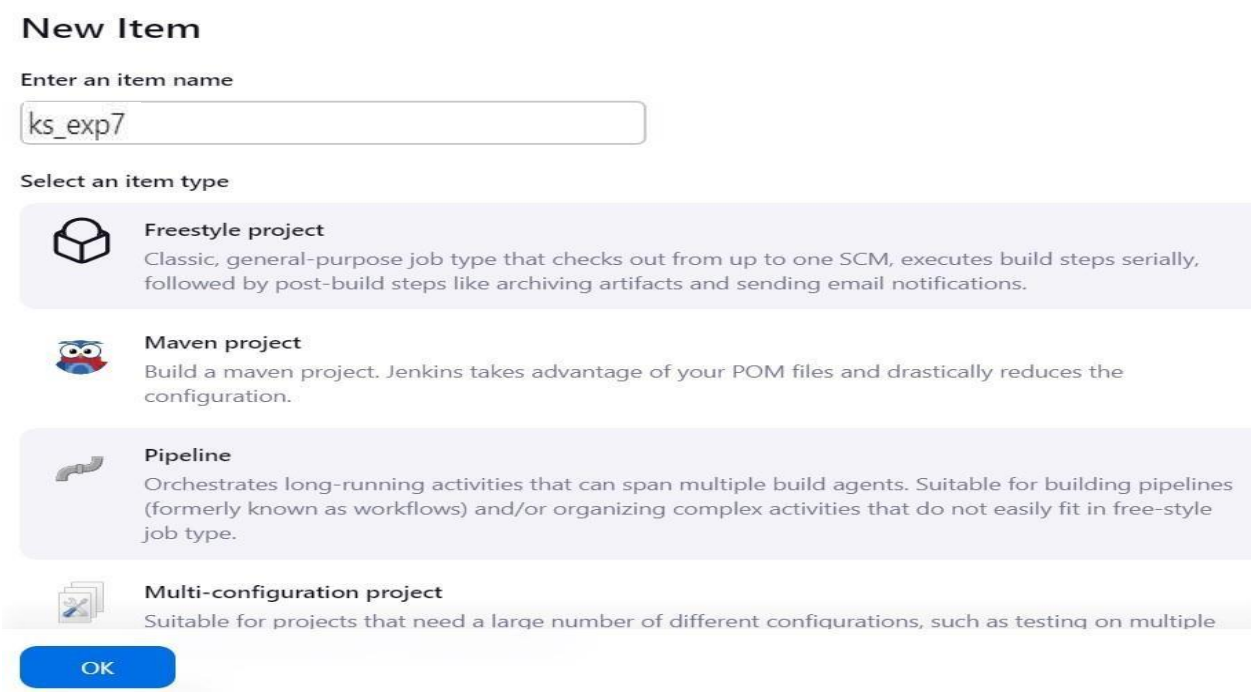
7. Search for SonarQube Scanner under Global Tool Configuration. Choose the latest configuration and choose Install automatically.

Dashboard > Manage Jenkins > Tools



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

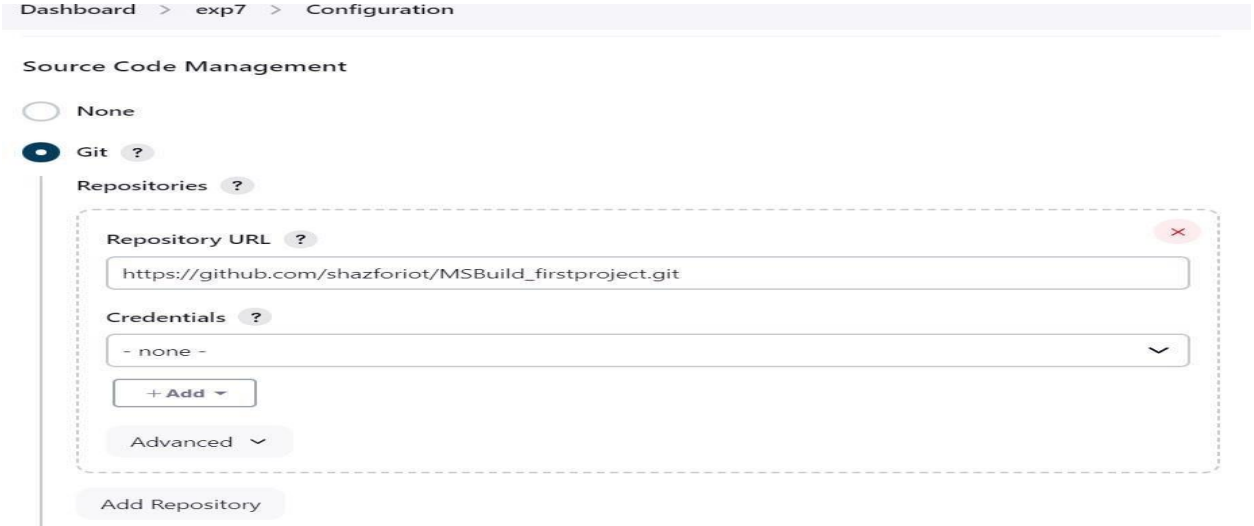
8. After the configuration, create a New Item in Jenkins, choose a freestyle project.ks



9. Choose this GitHub repository in Source Code Management.

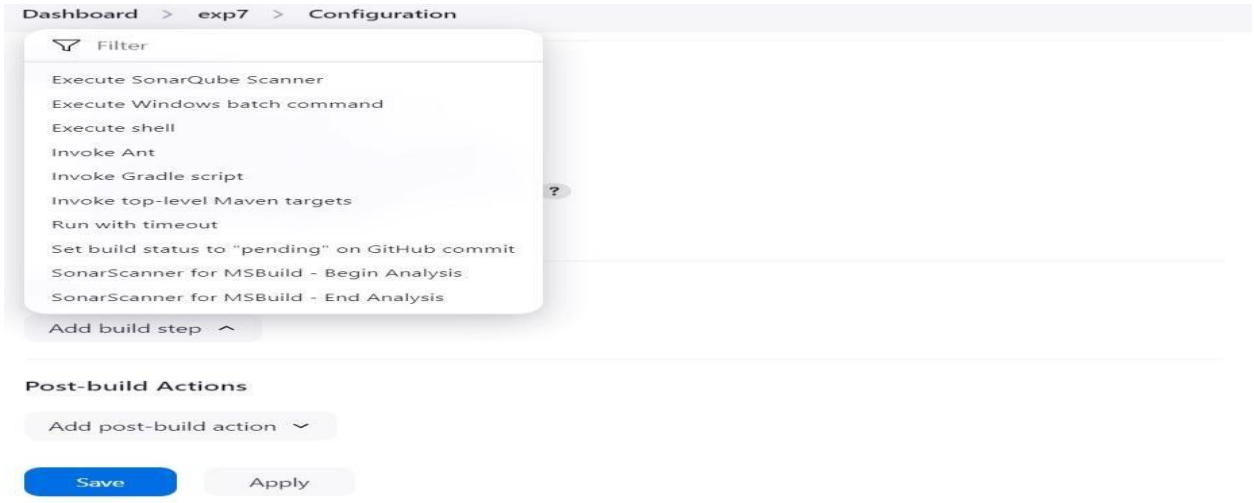
https://github.com/shazforiot/MSBuild_firstproject.git

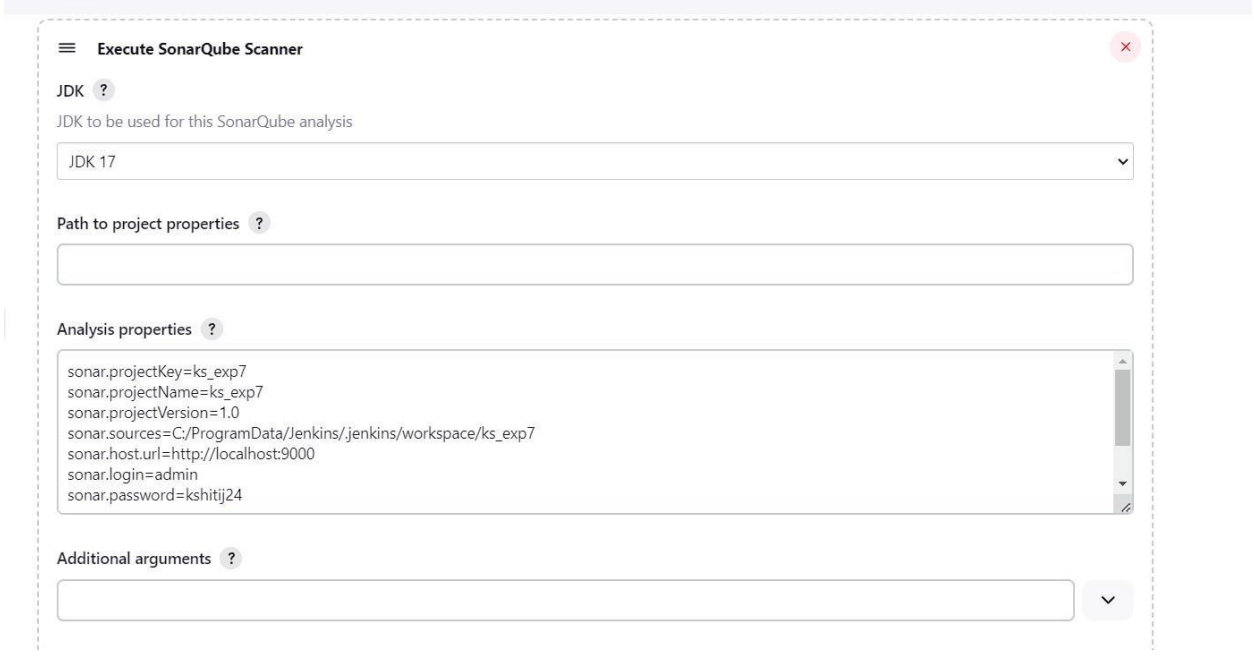
It is a sample hello-world project with no vulnerabilities and issues, just to test the integration.

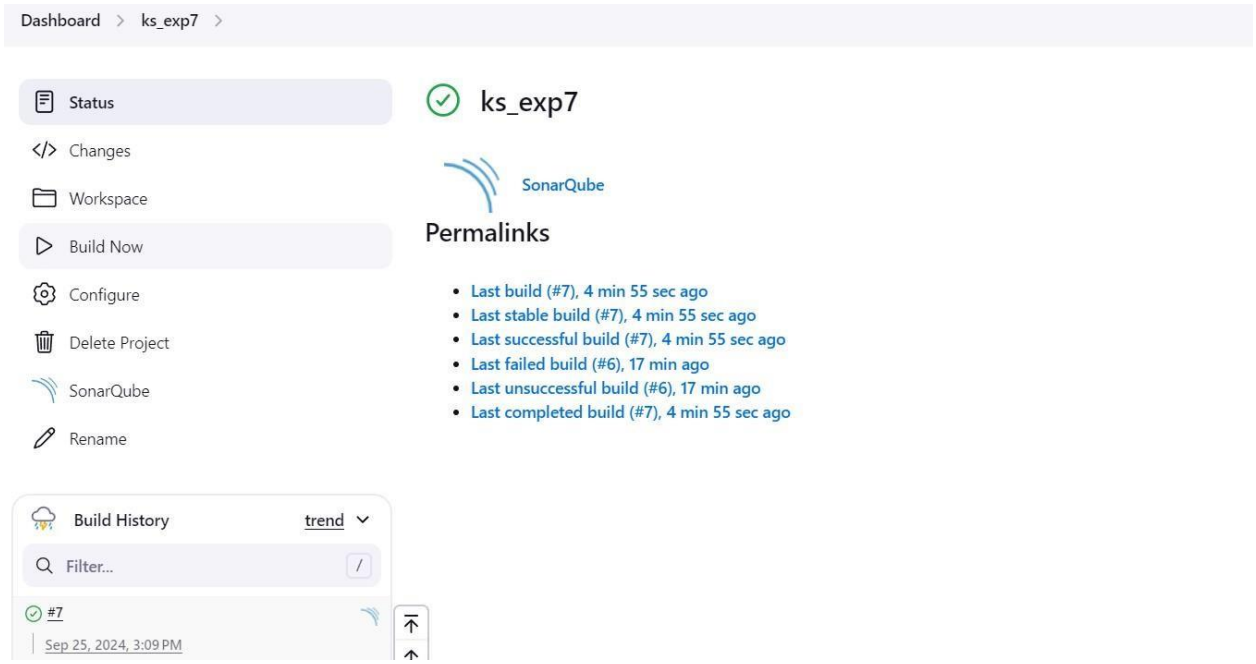


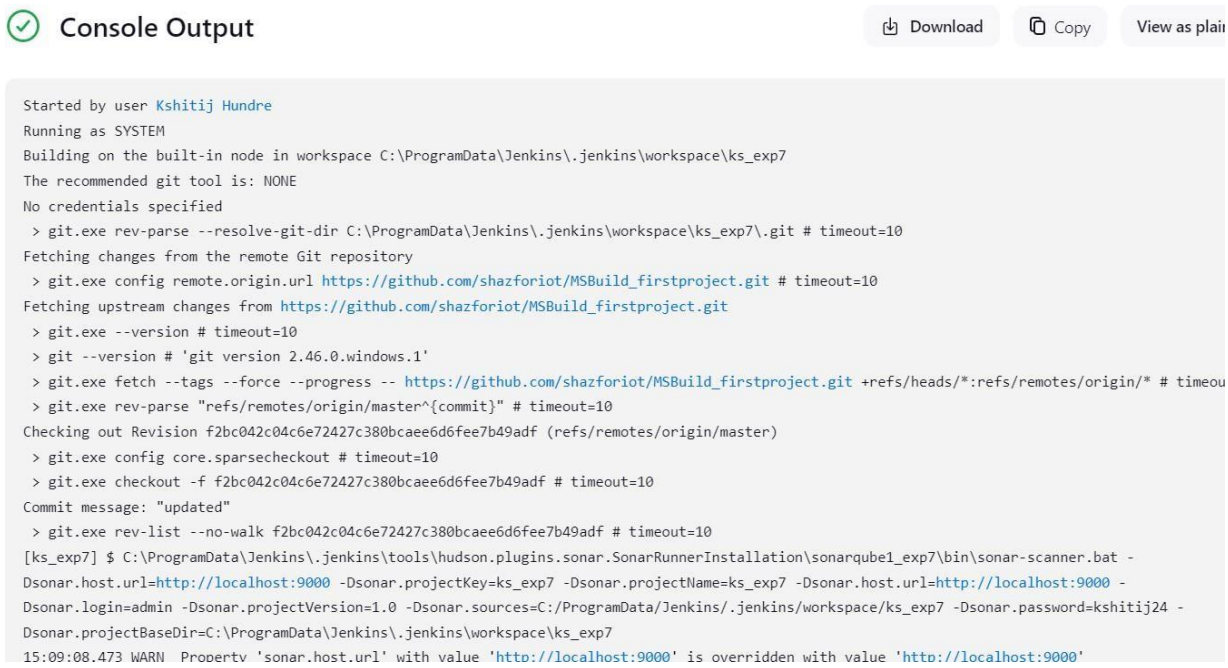
10. Under **Select project → Configuration → Build steps → Execute SonarQube Scanner**, enter these Analysis properties. Mention the SonarQube Project Key, Login, Password, Source path and Host URL.

Name: Ganesh Gupta Div:D15C Roll No.13

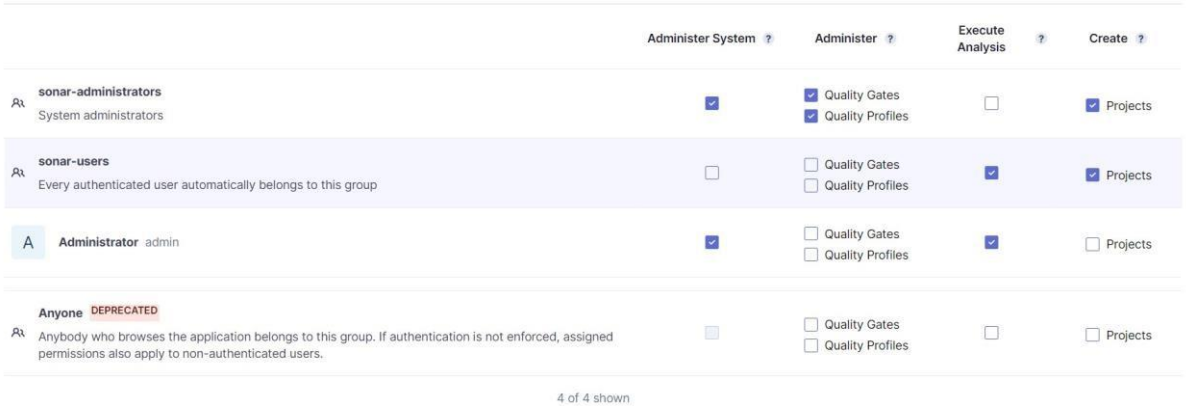




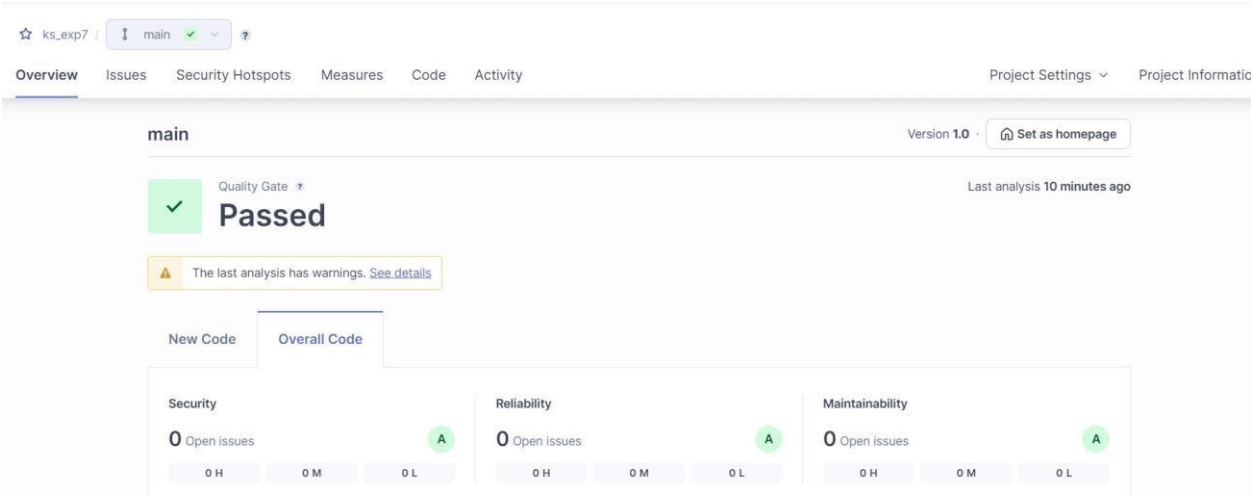




11. Go to http://localhost:9000/<user_name>/permissions and allow Execute Permissions to the Admin user.



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



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

Conclusion:

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

Expt No. 08 Advanced DevOps Lab

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.

Theory:

What is SAST?

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

What problems does SAST solve?

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

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

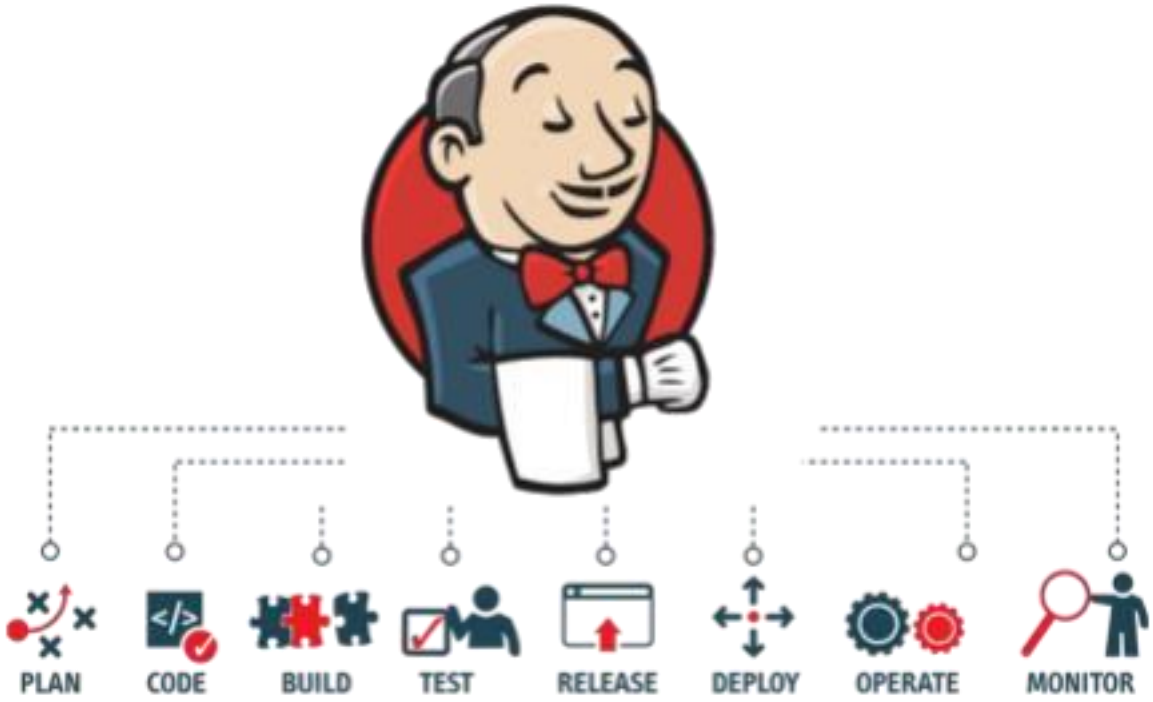
It's important to note that SAST tools must be run on the application on a regular basis, such as during daily/monthly builds, every time code is checked in, or during a code release.

Why is SAST important?

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

CI/CD pipeline refers to the Continuous Integration/Continuous Delivery pipeline. Before we dive deep into this segment, let's first understand what is meant by the term ‘pipeline’?

A pipeline is a concept that introduces a series of events or tasks that are connected in a sequence to make quick software releases. For example, there is a task, that task has got five different stages, and each stage has got some steps. All the steps in phase one have to be completed, to mark the latter stage to be complete.



Now, consider the CI/CD pipeline as the backbone of the DevOps approach. This Pipeline is responsible for building codes, running tests, and deploying new software versions. The Pipeline executes the job in a defined manner by first coding it and then structuring it inside several blocks that may include several steps or tasks.

What is SonarQube?

SonarQube is an open-source platform developed by SonarSource for continuous inspection of code quality. Sonar does static code analysis, which provides a detailed report of bugs, code smells, vulnerabilities, code duplications.
It supports 25+ major programming languages through built-in rulesets and can also be extended with various plugins.

Benefits of SonarQube

- **Sustainability -** Reduces complexity, possible vulnerabilities, and code duplications, optimising the life of applications.
- **Increase productivity -** Reduces the scale, cost of maintenance, and risk of the application; as such, it removes the need to spend more time changing the code
- **Quality code -** Code quality control is an inseparable part of the process of software development.
- **Detect Errors -** Detects errors in the code and alerts developers to fix them automatically before submitting them for output.
- **Increase consistency -** Determines where the code criteria are breached and enhances the quality
- **Business scaling -** No restriction on the number of projects to be evaluated
- **Enhance developer skills -** Regular feedback on quality problems helps developers to improve their coding skills

Integrating Jenkins with SonarQube:

Prerequisites:

- Jenkins installed
- Docker Installed (for SonarQube)

- SonarQube Docker Image

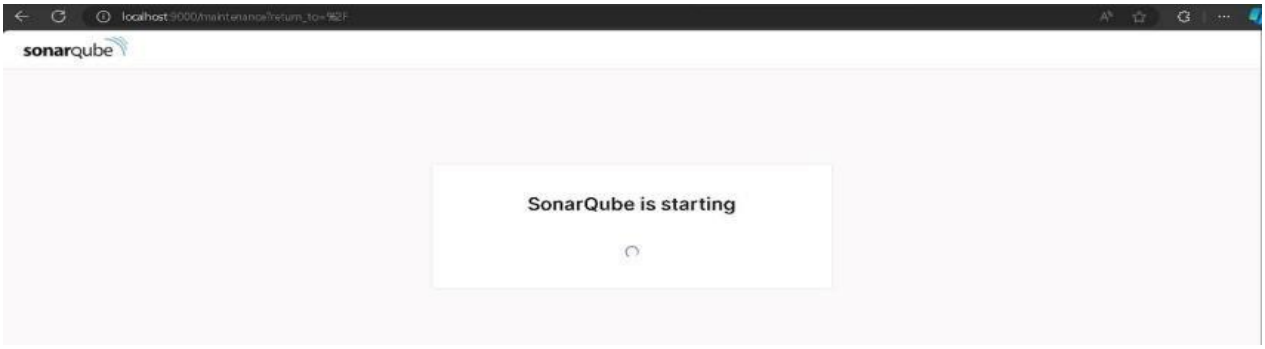
Steps to create a Jenkins CI/CD Pipeline and use SonarQube to perform SAST

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 -

```
PS C:\Users\91773\Desktop\College Resources\Advdevops Exp8> docker run -d --name sonarqube2 -e SONAR_ES_BOOTSTRAP_CHECKS
_DISABLE=true -p 9000:9000 sonarqube:latest
71fc67f0b15baa5be5bdcdd66966938e18682683d020beadcbc909dd027cfe7a
PS C:\Users\91773\Desktop\College Resources\Advdevops Exp8>
```

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

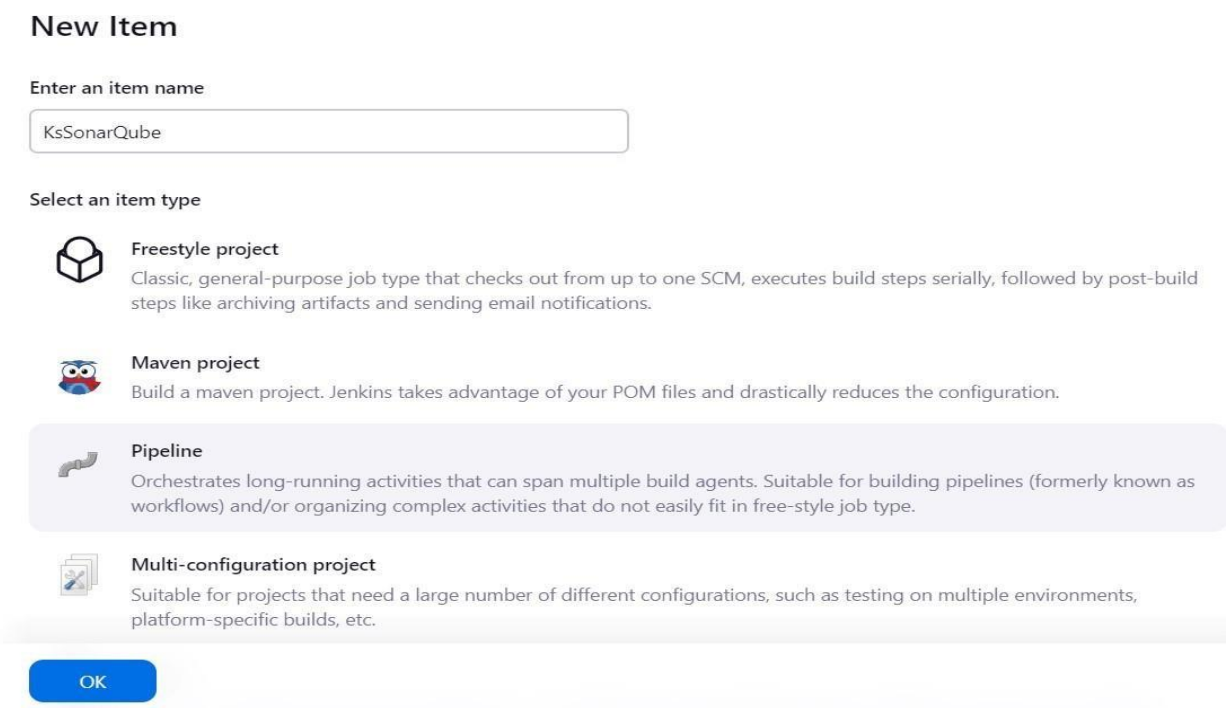


4. Login to SonarQube using username *admin* and password *admin*.

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

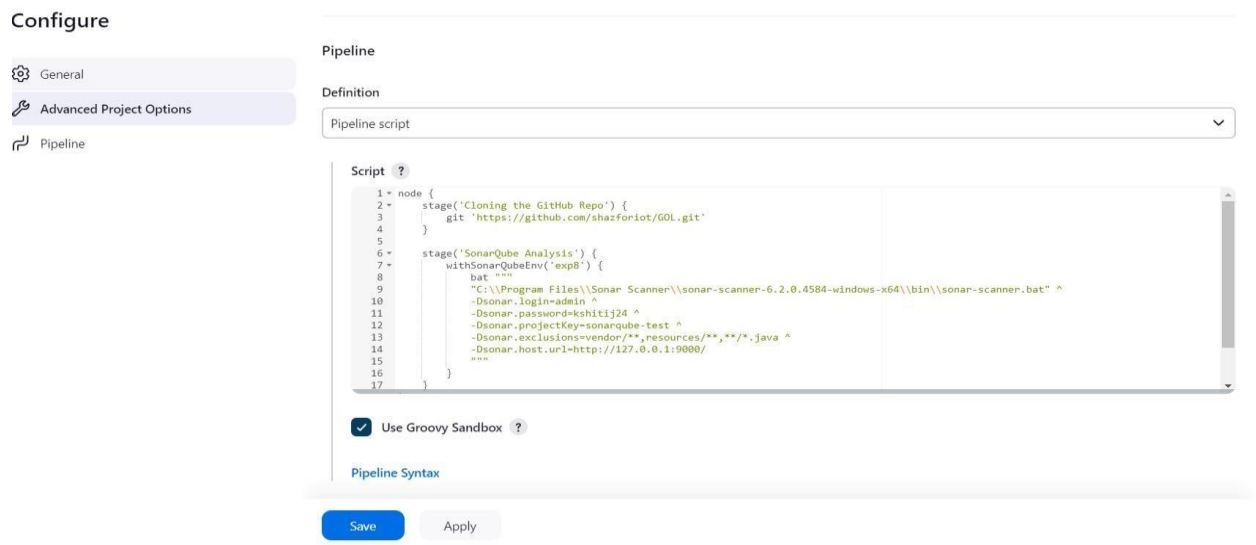
Setup the project and come back to Jenkins Dashboard.

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



7. Under Pipeline Script, enter the following - node {

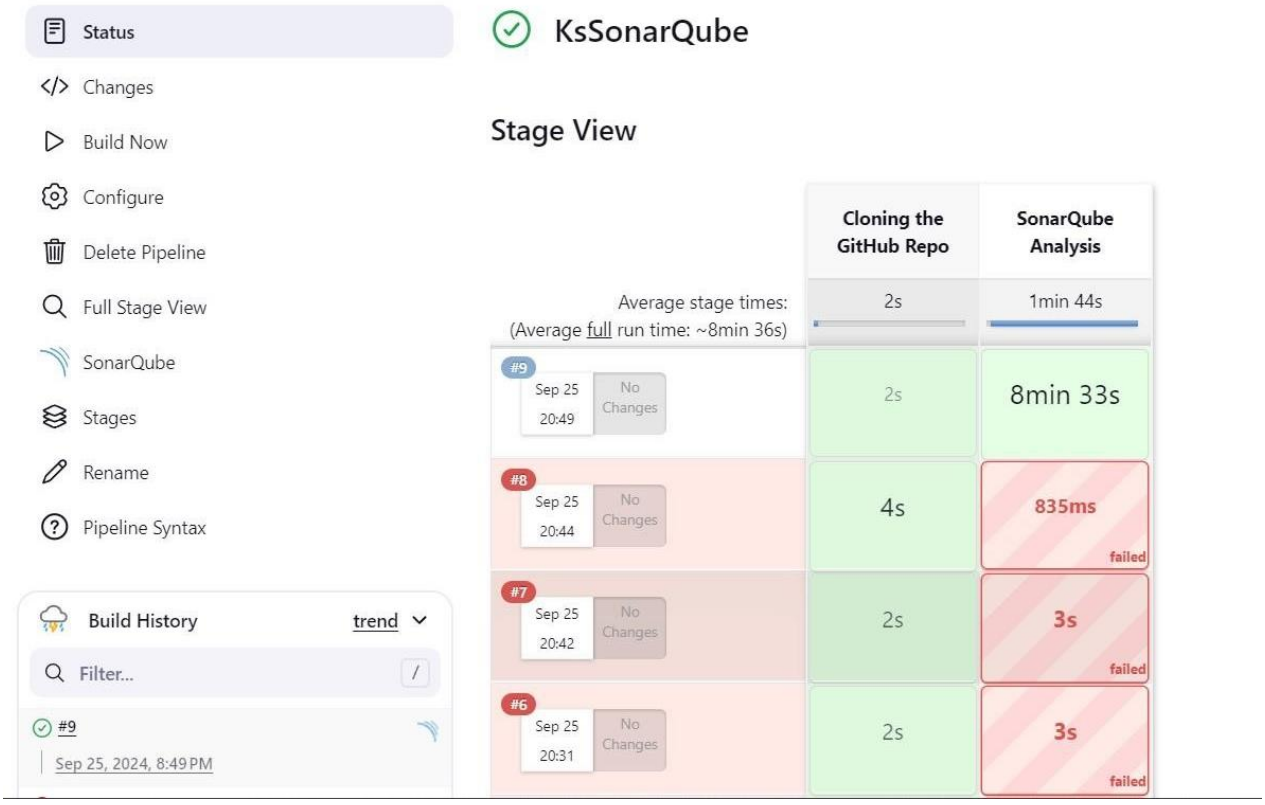
```
    stage('Cloning the GitHub Repo') {
      git 'https://github.com/shazforiot/GOL.git'
    } stage('SonarQube analysis')
    {
      withSonarQubeEnv('sonarqube') { sh
        "<PATH_TO_SONARQUBE_FOLDER>//bin//sonar-scanner \
        -D sonar.login=<SonarQube_USERNAME> \
        -D sonar.password=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



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

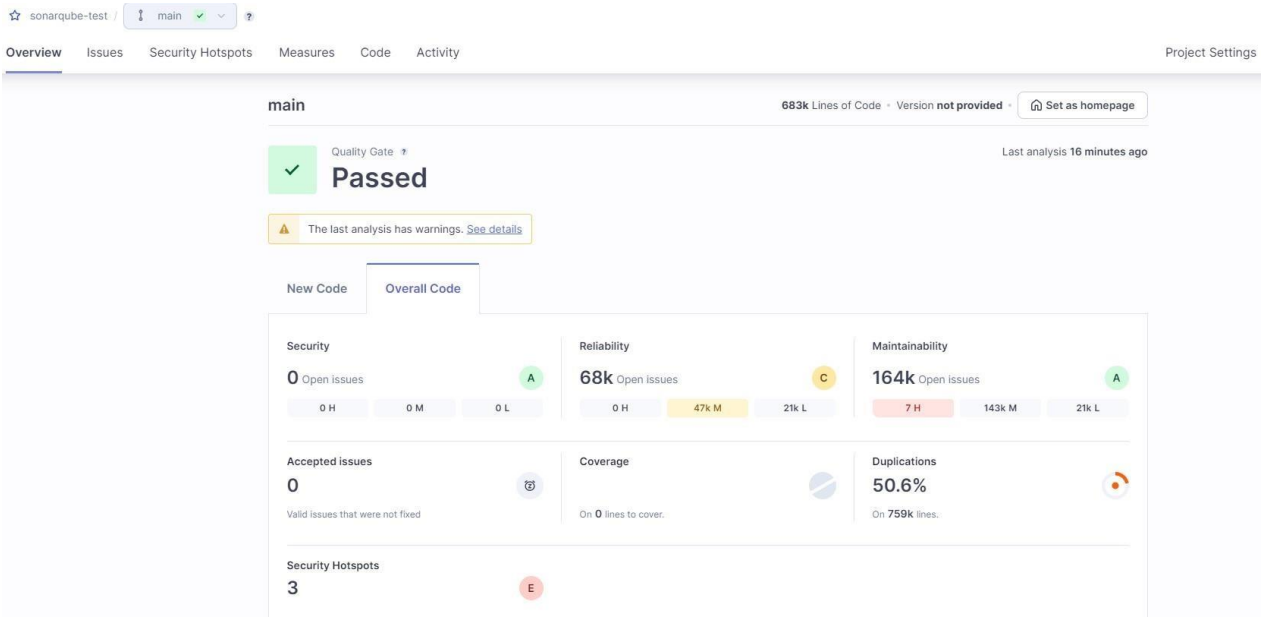
8. Run The Build.

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



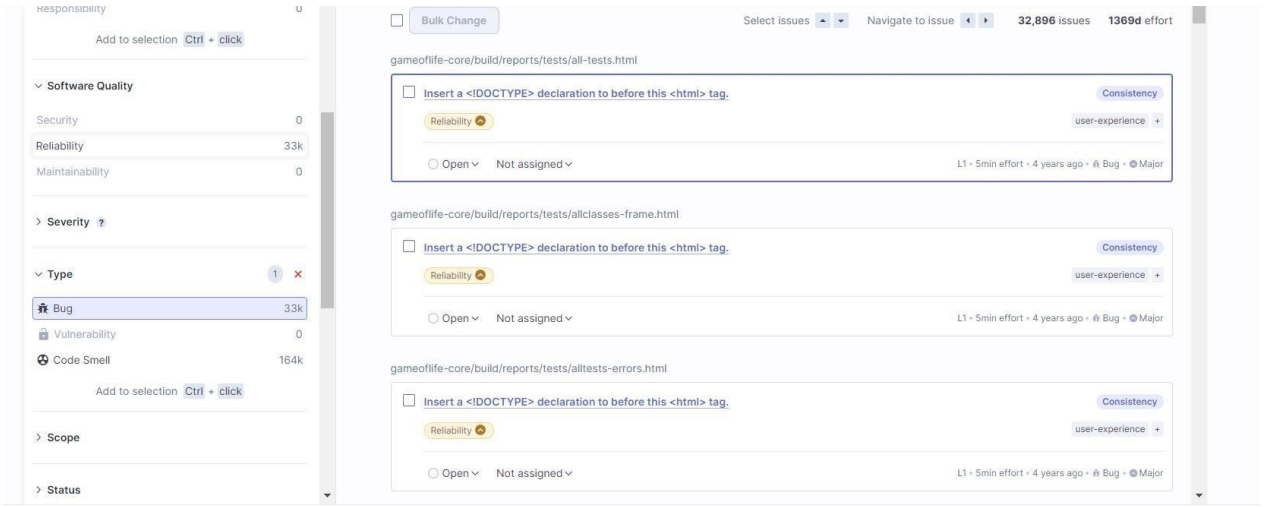

```
for block at line 17. Keep only the first 100 references.
20:56:18.455 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/gui/util/TextAreaCellRenderer.html
for block at line 296. Keep only the first 100 references.
20:56:18.455 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/gui/util/TextAreaCellRenderer.html
for block at line 75. Keep only the first 100 references.
20:56:18.456 INFO  CPD Executor CPD calculation finished (done) | time=107093ms
20:56:18.490 INFO  SCM revision ID 'ba799ba7e1b576f04a4612322b0412c5e6e1e5e4'
20:57:50.106 INFO  Analysis report generated in 3149ms, dir size=127.2 MB
20:57:56.943 INFO  Analysis report compressed in 6828ms, zip size=29.6 MB
20:57:58.685 INFO  Analysis report uploaded in 1732ms
20:57:58.688 INFO  ANALYSIS SUCCESSFUL, you can find the results at: http://127.0.0.1:9000/dashboard?id=sonarqube-test
20:57:58.688 INFO  Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
20:57:58.688 INFO  More about the report processing at http://127.0.0.1:9000/api/ce/task?id=18847db4-4f06-4766-9ad4-ee006448353c
20:58:06.225 INFO  Analysis total time: 8:22.672 s
20:58:06.231 INFO  SonarScanner Engine completed successfully
20:58:06.824 INFO  EXECUTION SUCCESS
20:58:06.857 INFO  Total time: 8:31.713s
[Pipeline] }
[Pipeline] // withSonarQubeEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

10. After that, check the project in SonarQube.

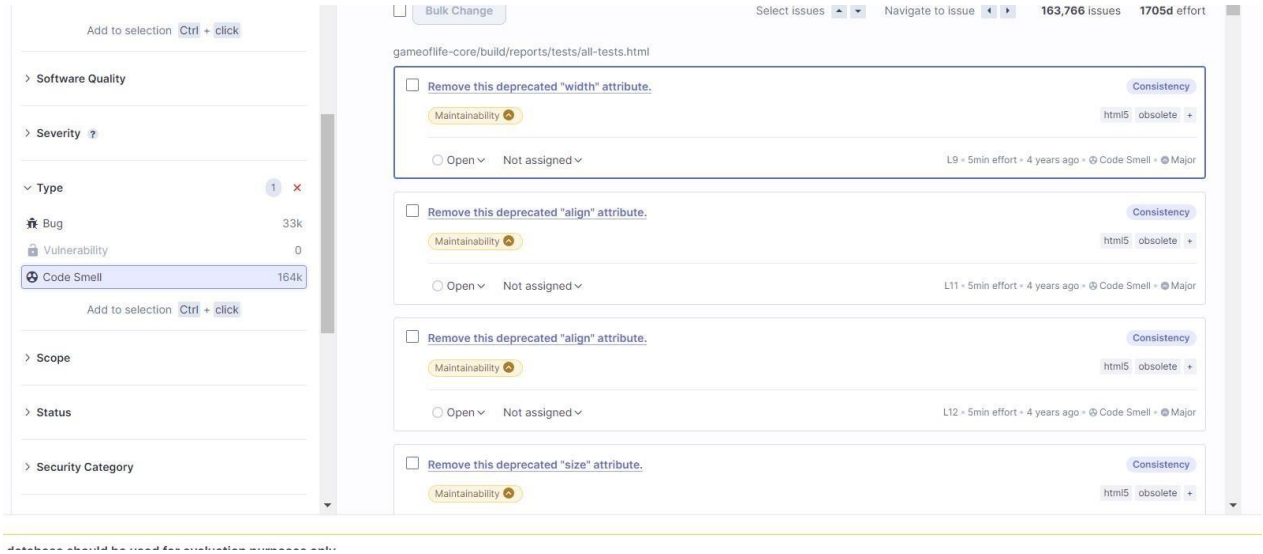


Under different tabs, check all different issues with the code. 11.

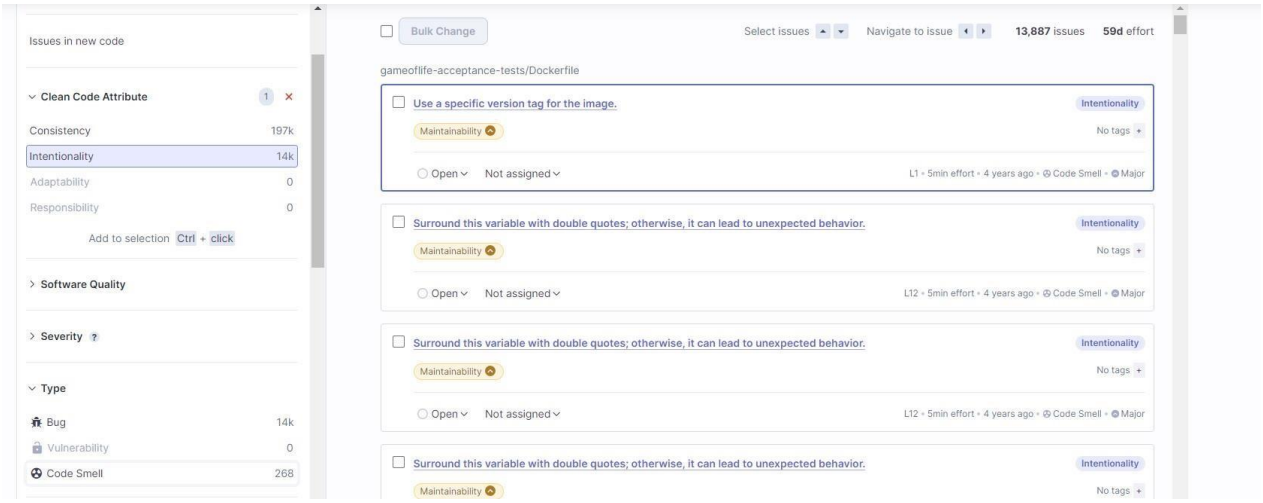
Bugs



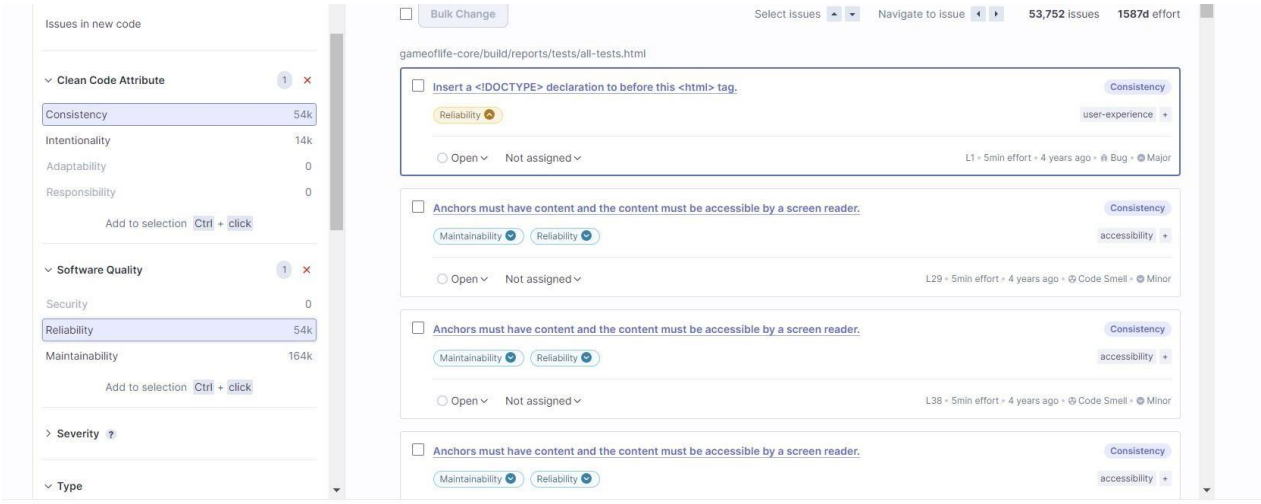
Code Smells



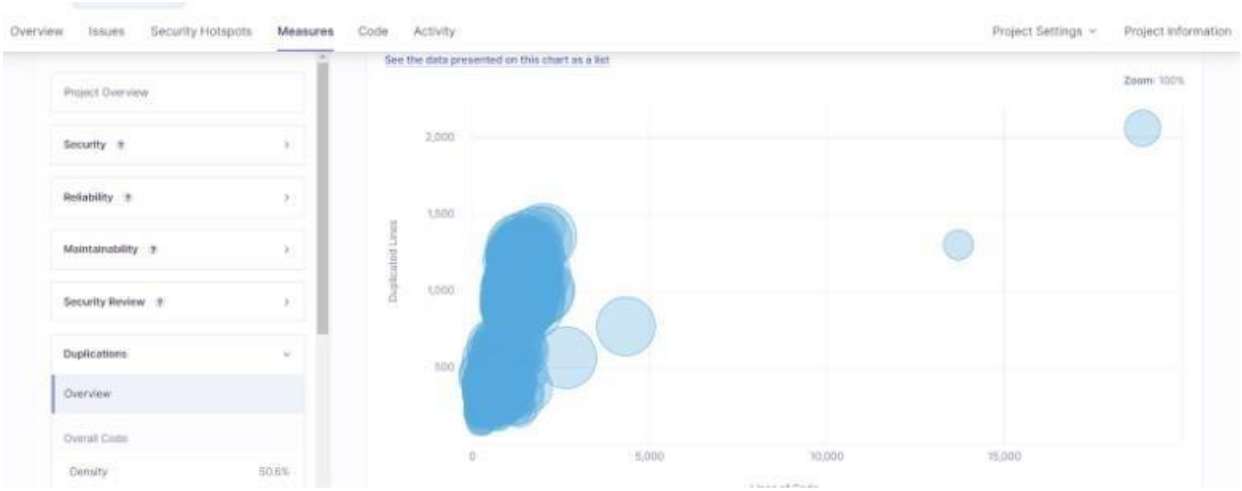
Intentional issues



Reliabilities issue



Duplicates



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

Conclusion:

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

Name:Ganesh Gupta Div: D15C Roll No: 13

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

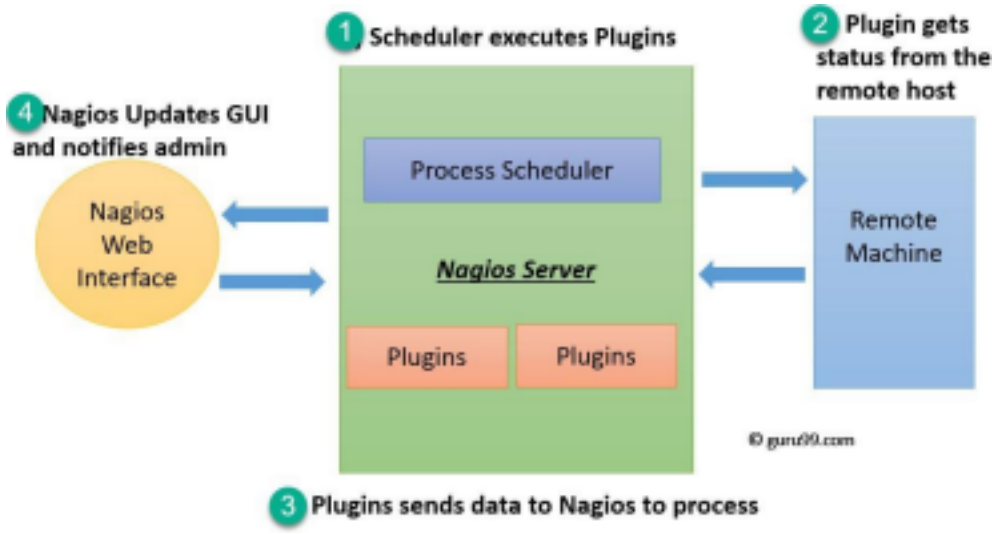
Following are the important features of Nagios monitoring tool:

- 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.

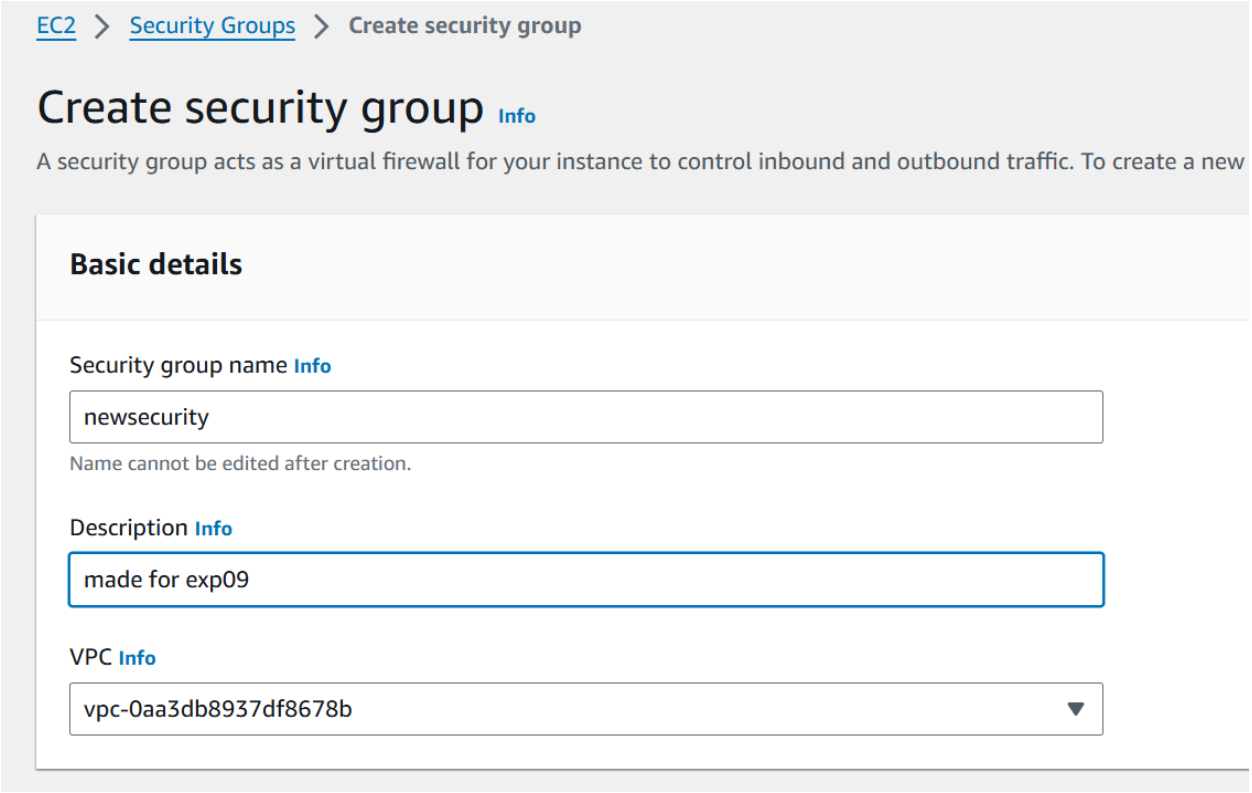


1. The scheduler is a component of the server part of Nagios. It sends a signal to execute the plugins at the remote host.
2. The plugin gets the status from the remote host
3. The plugin sends the data to the process scheduler
4. The process scheduler updates the GUI and notifications are sent to admins.

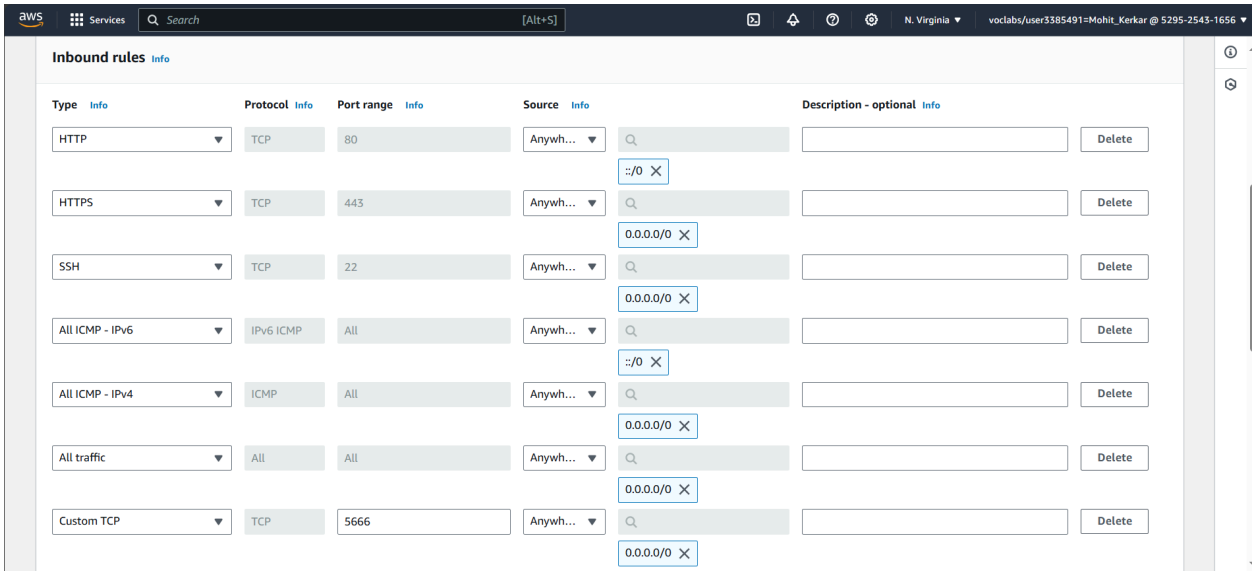
Name:Ganesh Gupta Div: D15C Roll No: 13

Step 1: Create a security group with the required configurations

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

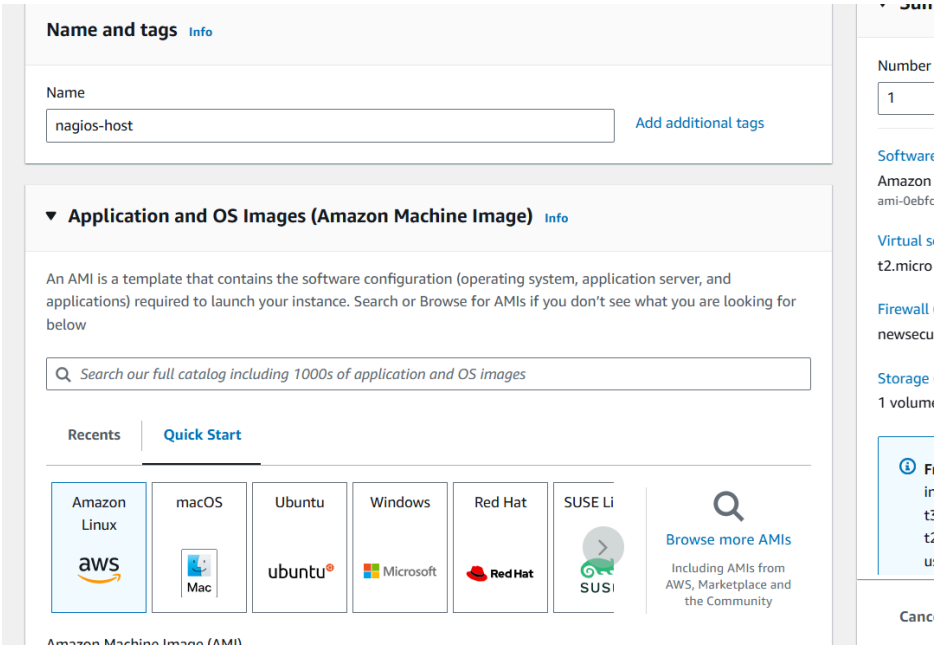


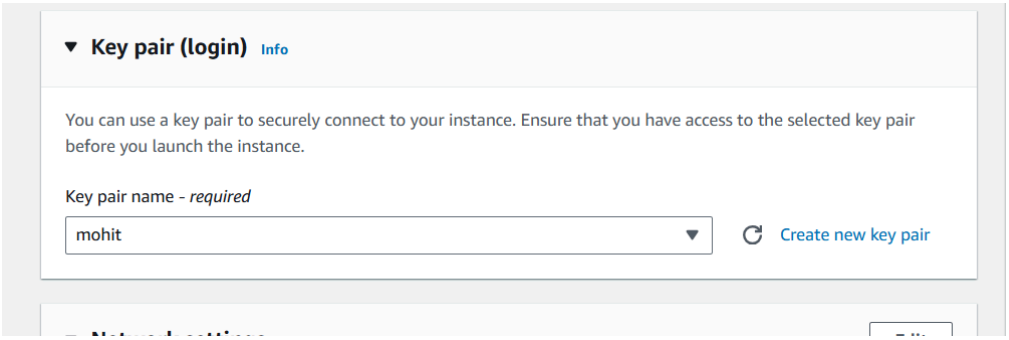
I have modified the INBOUND RULES as follows

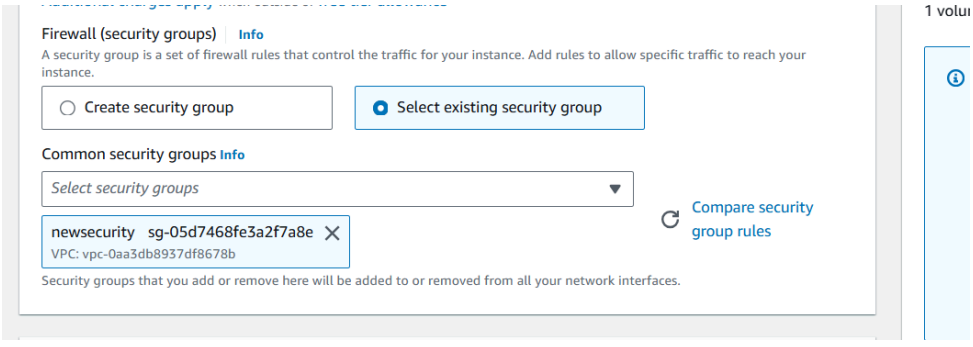


Step 2: Create ec2 instance

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

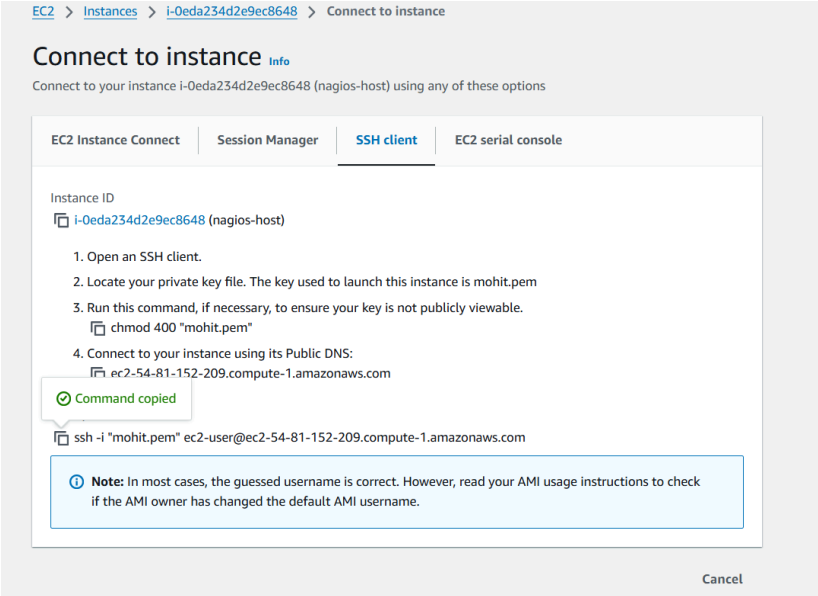




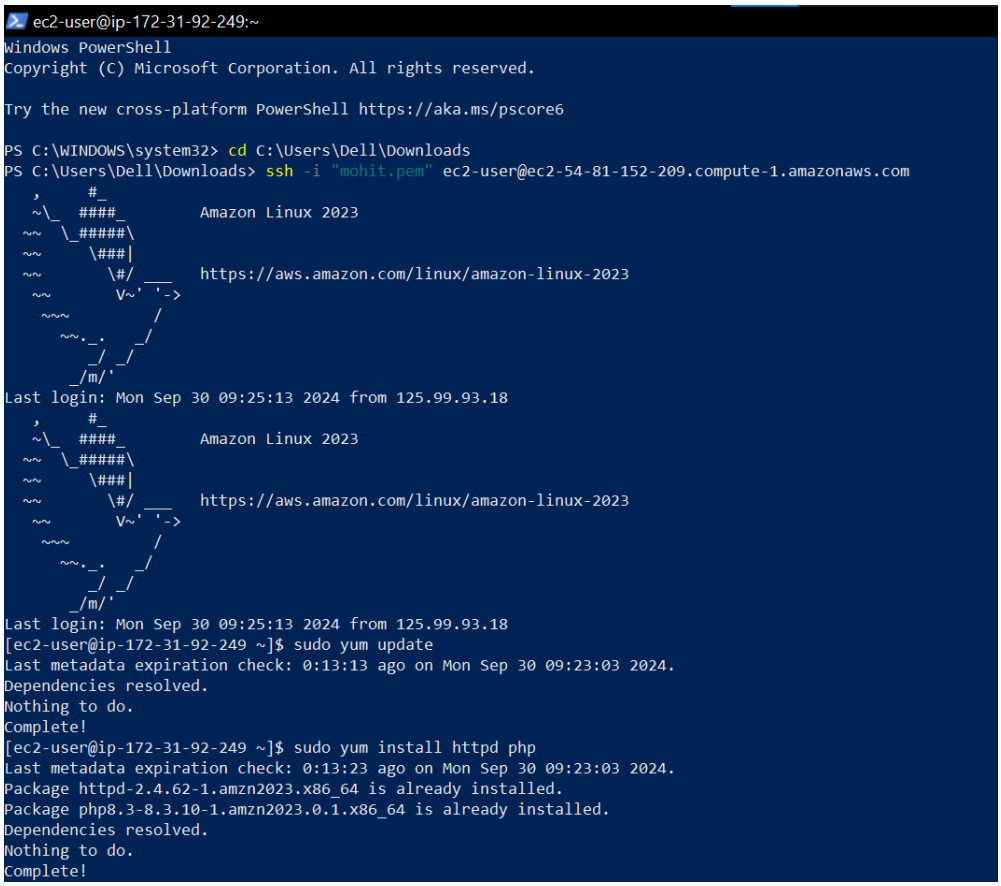


Name:Ganesh Gupta Div: D15C Roll No: 13

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



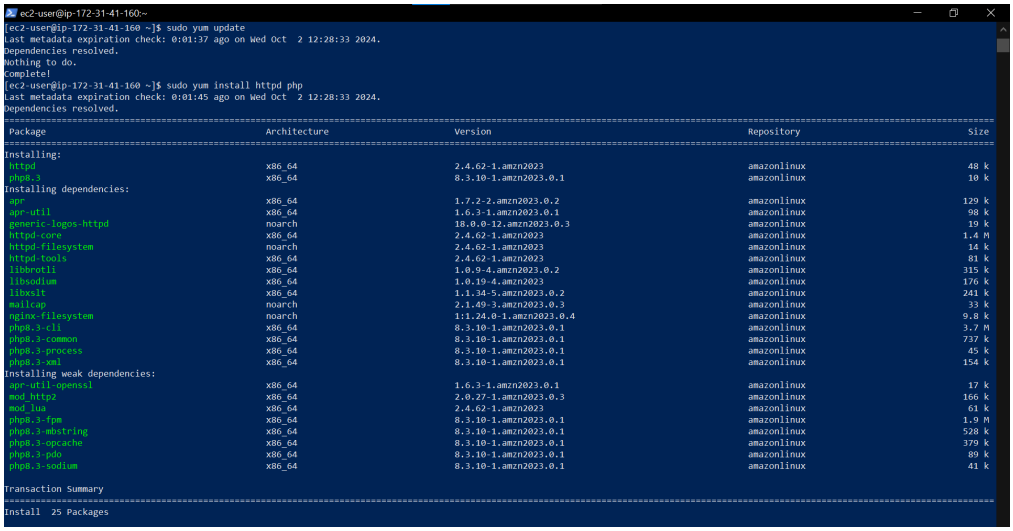
Step 3: Open an administrative powershell and remotely login using the above mentioned ssh command



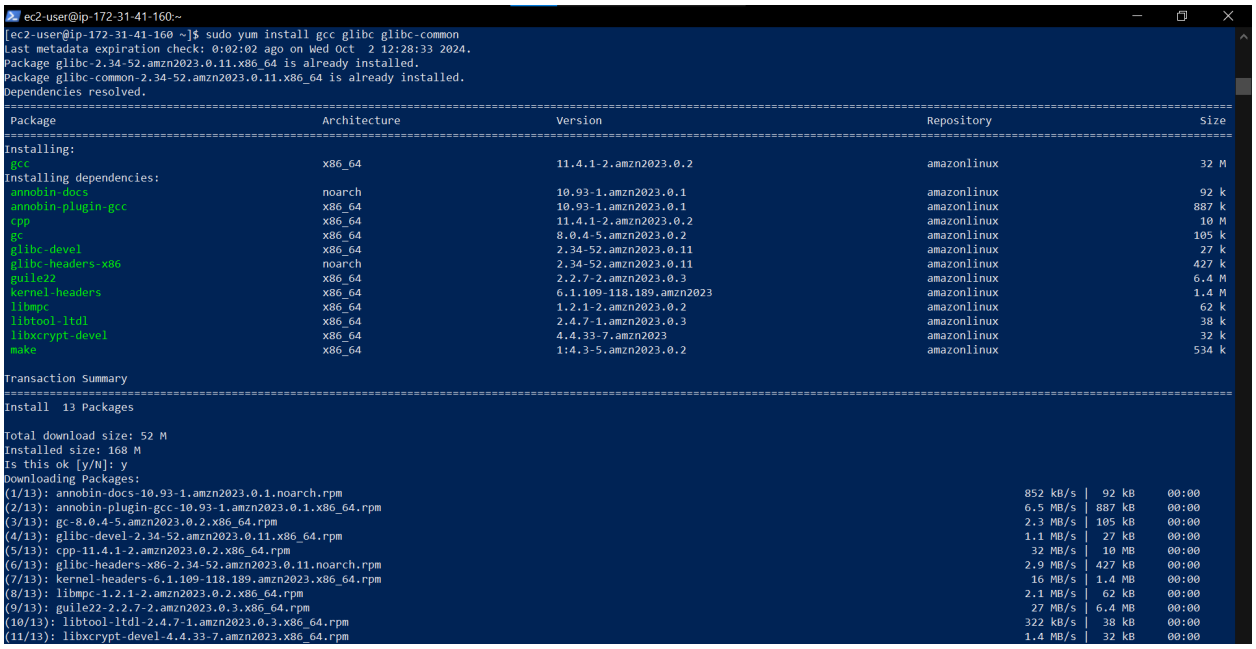
And then run these commands

sudo yum update

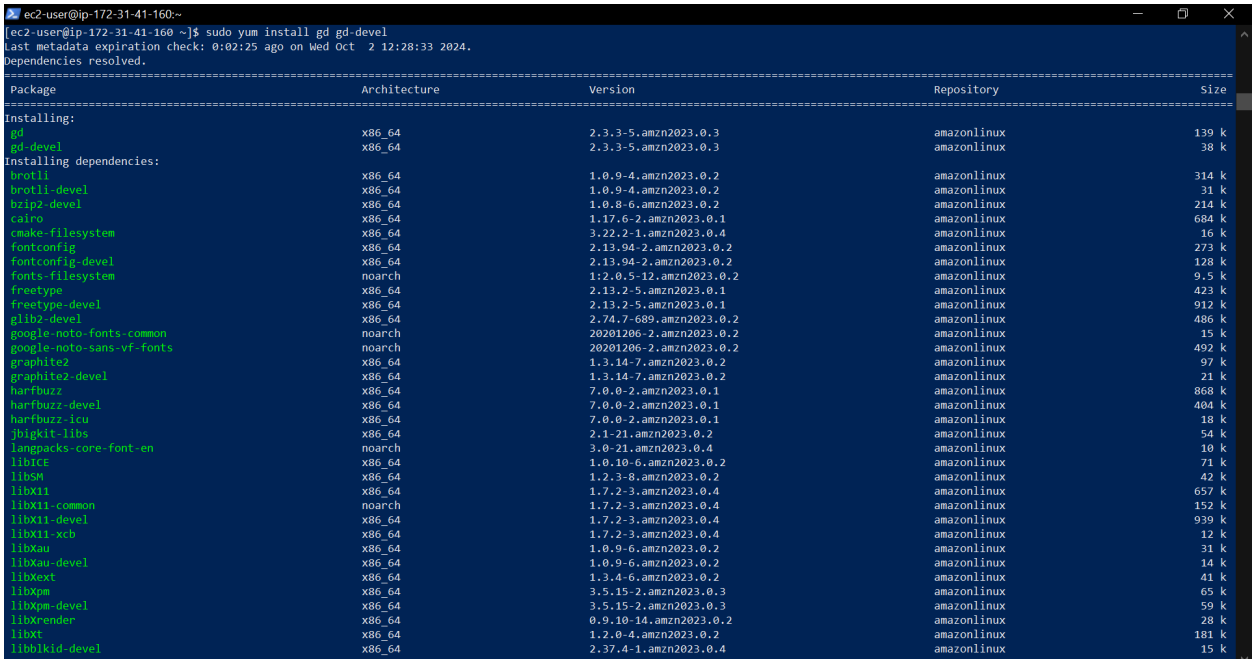
sudo yum install httpd php



sudo yum install gcc glibc glibc-common



sudo yum install gd gd-devel

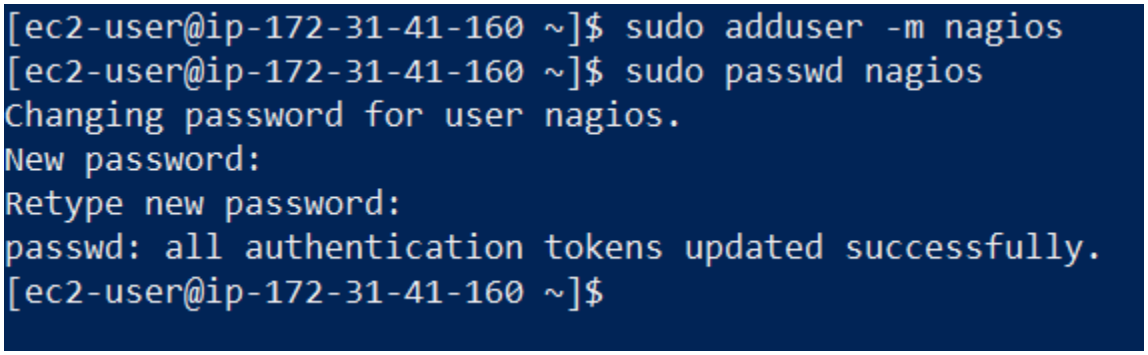


```
ec2-user@ip-172-31-41-160:~
[ec2-user@ip-172-31-41-160 ~]$ sudo yum install gd gd-devel
Last metadata expiration check: 0:02:25 ago on Wed Oct  2 12:28:33 2024.
Dependencies resolved.
================================================================================
 Package                 Architecture    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

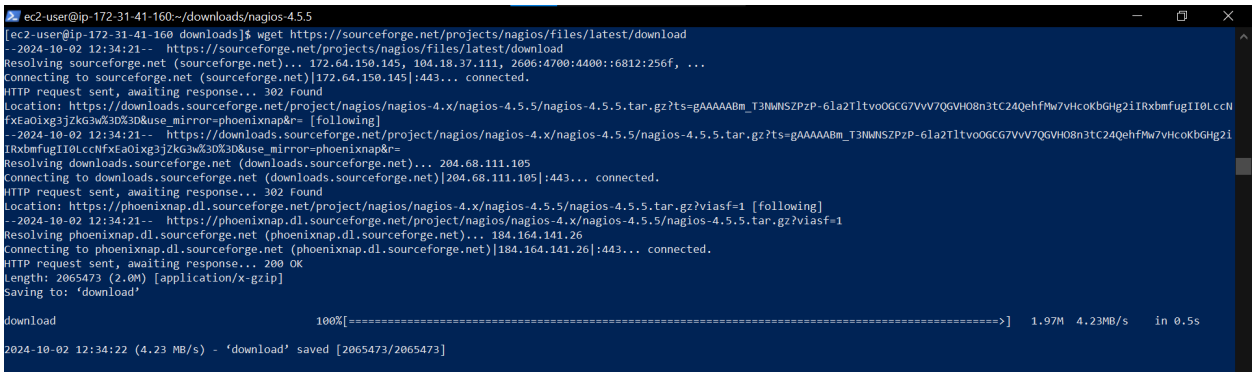
cd ~/downloads

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

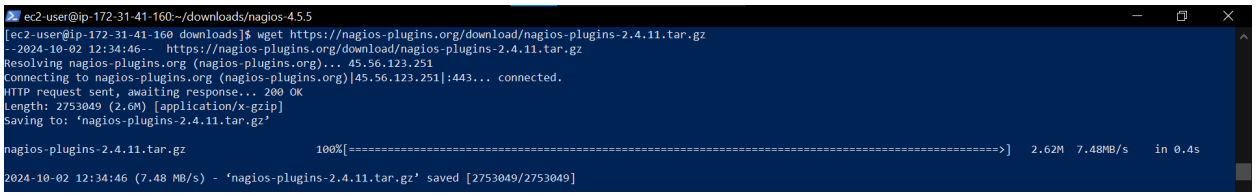
In this step, we are downloading, the latest version of nagios and the necessary plugins required to carry out the tasks of setting up a nagios server

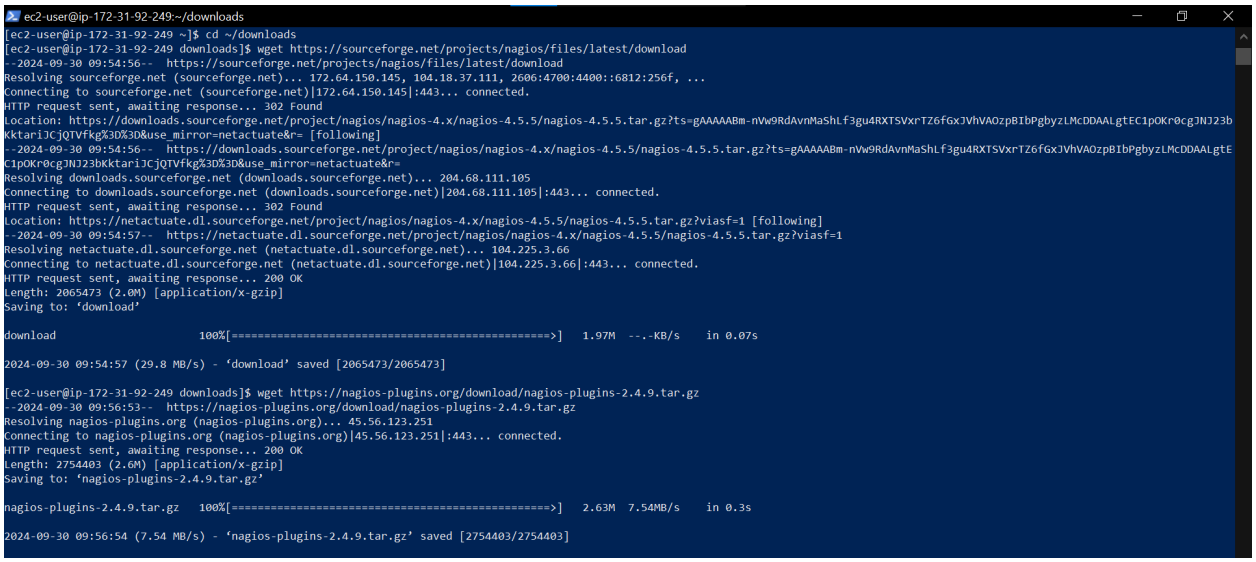
Name:Ganesh Gupta Div: D15C Roll No: 13

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



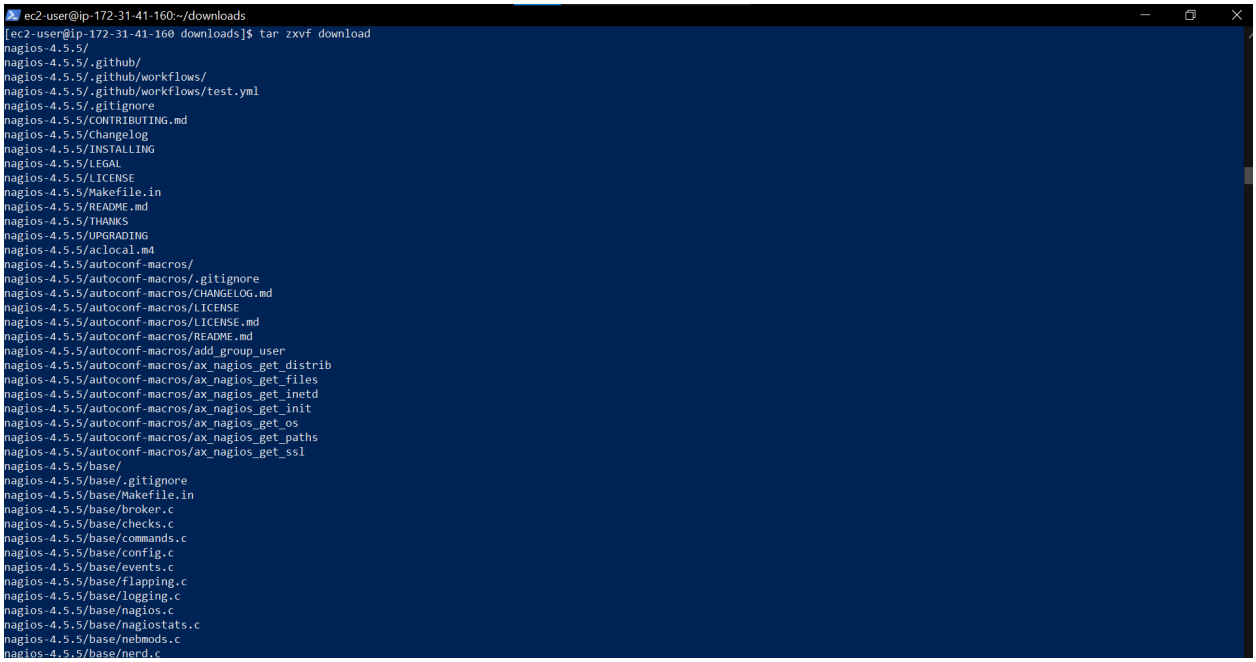
wget https://nagios-plugins.org/download/nagios-plugins-2.4.11.tar.gz





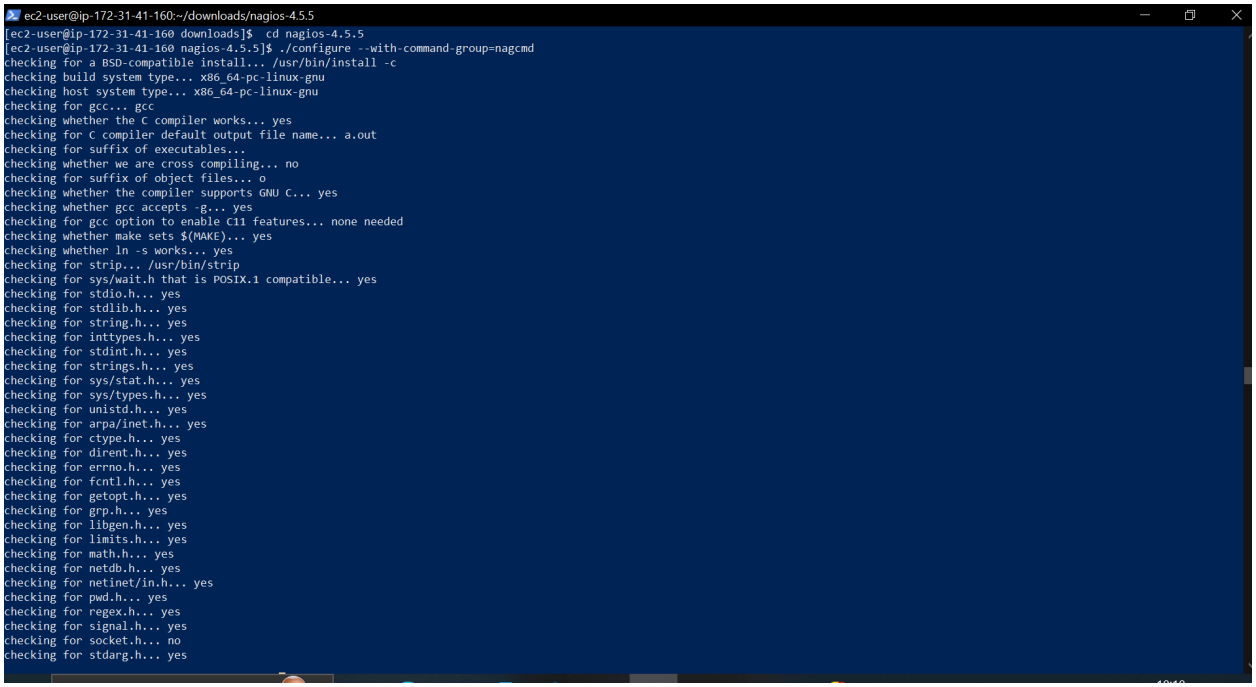
Now, we run the next command in the following manner
tar zxvf <nagios-4.5.5 version> (for me it has gotten saved as 'download')
So i wrote **tar zxvf download**

Name:Ganesh Gupta Div: D15C Roll No: 13



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 strstr... yes
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**

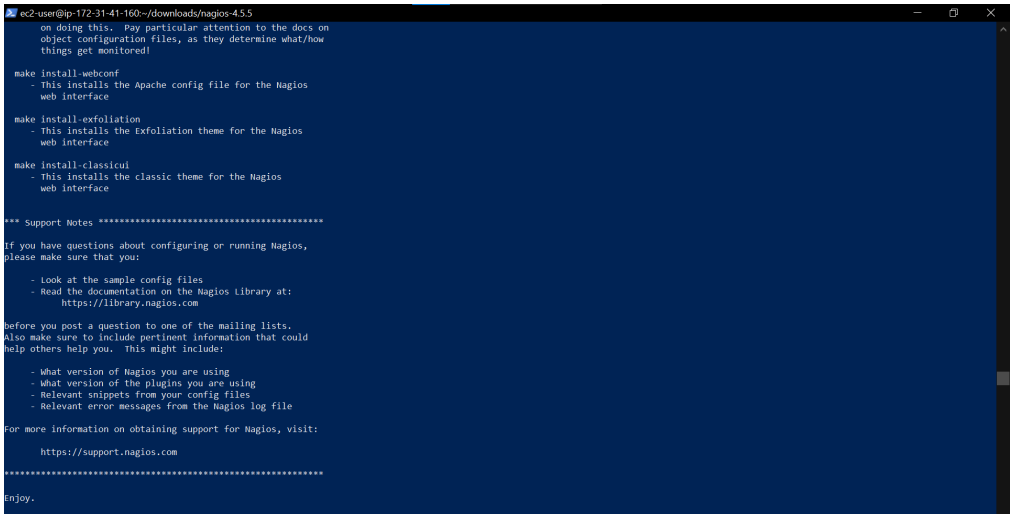
For that purpose, we are to run the following command.
sudo yum install openssl-devel (for ssl header)

```
[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:Ganesh Gupta Div: D15C Roll No: 13



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]$
```

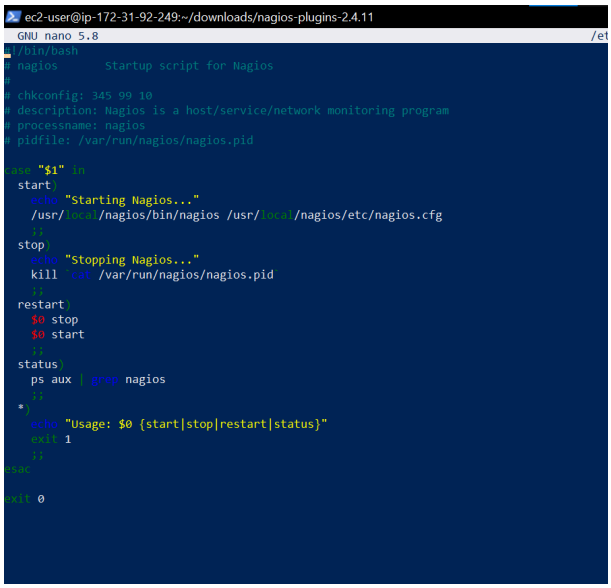
With ls command, we must see a file named nagios, which i was not able to see
If the Init Script is Missing i.e If you don't see the nagios script in /etc/init.d/, you can create it manually. Here's how:

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

Within this file, paste the following script

Name:Ganesh Gupta Div: D15C Roll No: 13

```
#!/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
```



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

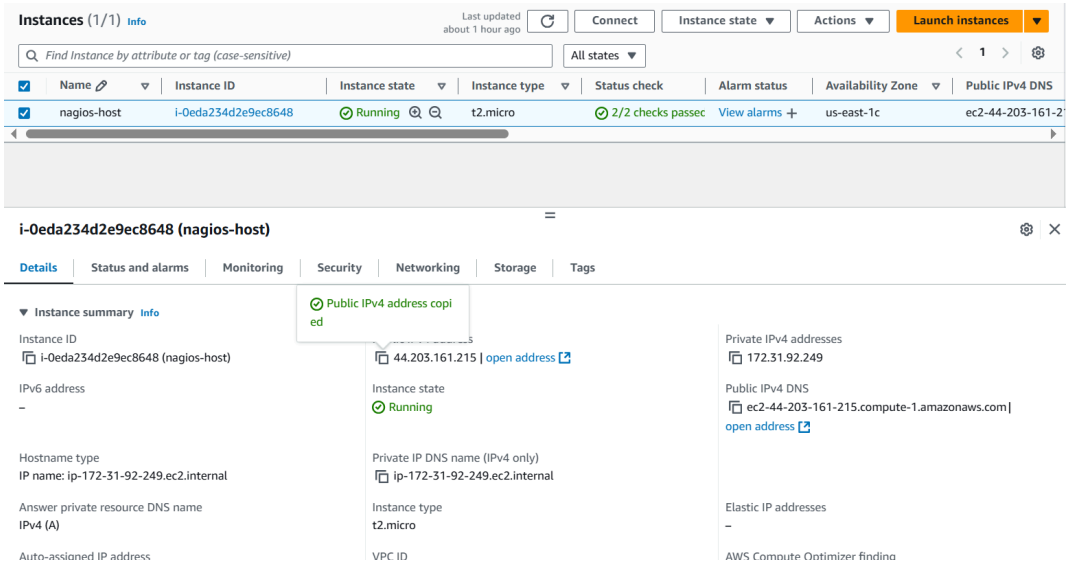
Run **sudo chkconfig --add nagios** again
And then run **sudo chkconfig nagios on**

```
[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:Ganesh Gupta Div: D15C Roll No: 13

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:Ganesh Gupta Div: D15C Roll No: 13

Adv DevOps Exp 10

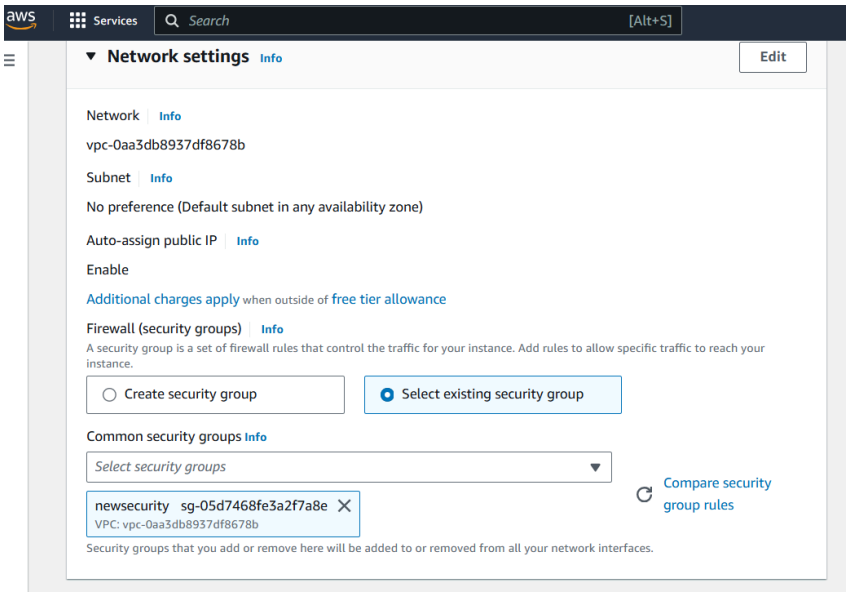
Aim: To perform Port, Service monitoring, Windows/Linux server monitoring using Nagios.

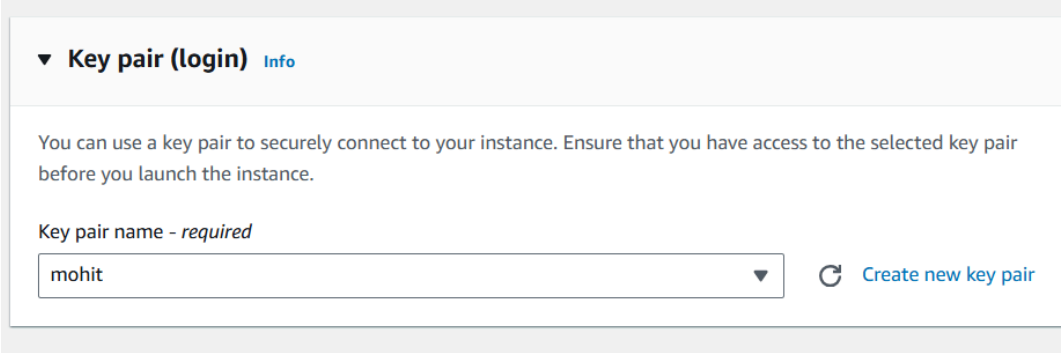
Monitoring Using Nagios:

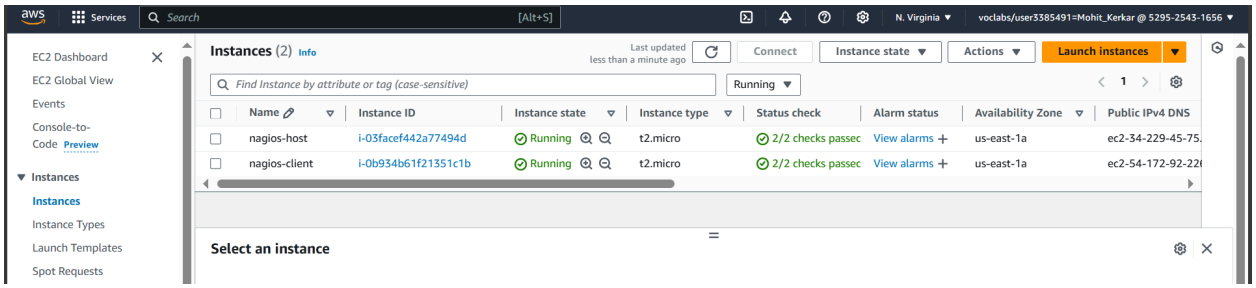
Step 1: To Confirm Nagios is running on the server side Perform the following command on your Amazon Linux Machine (Nagios-host).
Run this command **sudo systemctl status**

```
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.







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

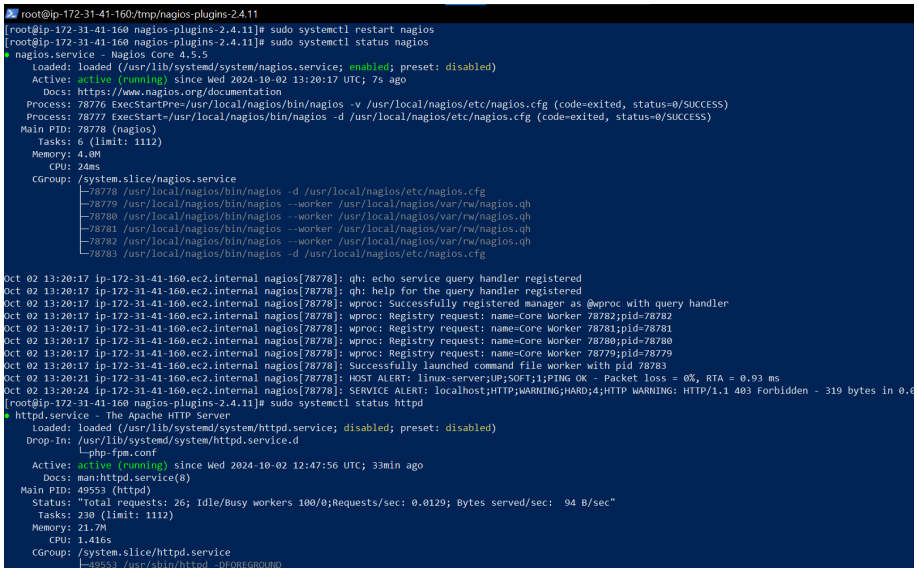
```
[root@ip-172-31-41-160 nagios-plugins-2.4.11]# /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

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

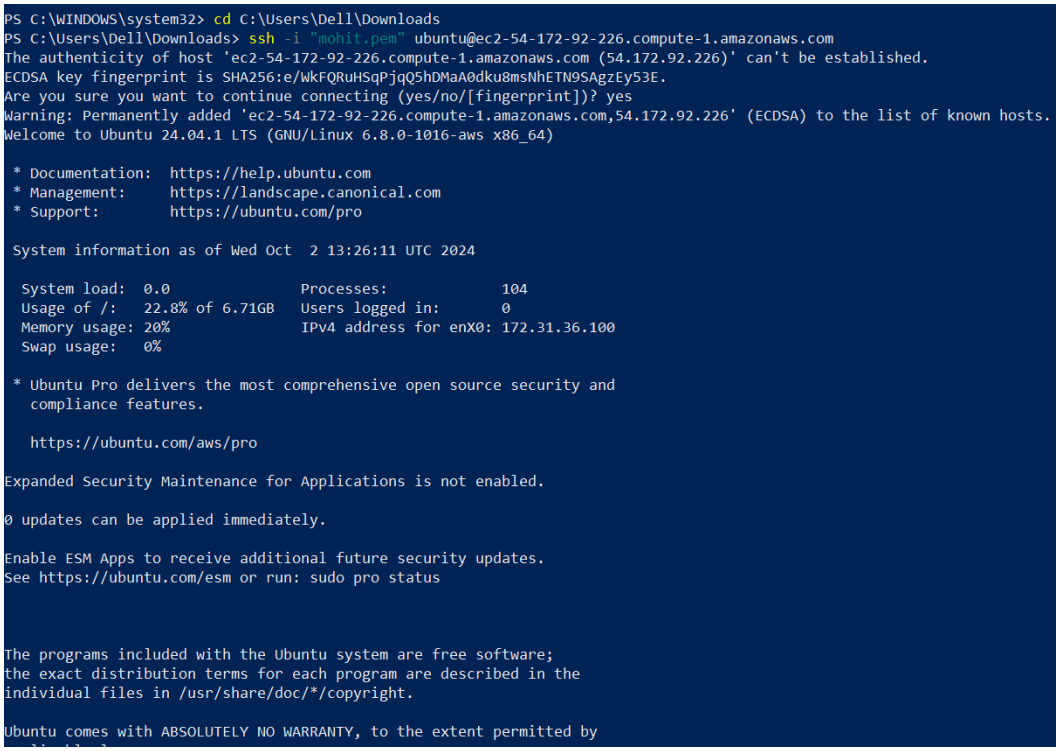
And by running sudo systemctl status nagios, we can again check whether our server is running or not



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:Ganesh Gupta Div: D15C Roll No: 13

```
ubuntu@ip-172-31-36-100: ~
ubuntu@ip-172-31-36-100: $ sudo apt update -y
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:6 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [380 kB]
Get:7 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [83.1 kB]
Get:8 http://security.ubuntu.com/ubuntu noble-security/main amd64 c-n-f Metadata [4576 B]
Get:9 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [275 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:11 http://security.ubuntu.com/ubuntu noble-security/universe Translation-en [116 kB]
Get:12 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [8632 B]
Get:13 http://security.ubuntu.com/ubuntu noble-security/universe amd64 c-n-f Metadata [10.4 kB]
Get:14 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Packages [10.9 kB]
Get:15 http://security.ubuntu.com/ubuntu noble-security/multiverse Translation-en [2808 B]
Get:16 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [208 B]
Get:17 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 c-n-f Metadata [344 B]
Get:18 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:19 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
Get:20 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
Get:21 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
Get:22 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Components [35.0 kB]
Get:23 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 c-n-f Metadata [8328 B]
Get:24 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [535 kB]
Get:25 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [130 kB]
Get:26 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 c-n-f Metadata [8676 B]
Get:27 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [380 kB]
Get:28 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [157 kB]
Get:29 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [45.0 kB]
Get:30 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 c-n-f Metadata [14.9 kB]
Get:31 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Packages [14.4 kB]
Get:32 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse Translation-en [3608 B]
Get:33 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [212 B]
Get:34 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 c-n-f Metadata [532 B]
Get:35 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 Components [208 B]
Get:36 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 c-n-f Metadata [112 B]
Get:37 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Packages [10.6 kB]
Get:38 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe Translation-en [10.8 kB]
Get:39 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Components [17.6 kB]
Get:40 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 c-n-f Metadata [1104 B]
Get:41 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/restricted amd64 Components [216 B]
Get:42 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/restricted amd64 c-n-f Metadata [116 B]
Get:43 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 Components [212 B]
```

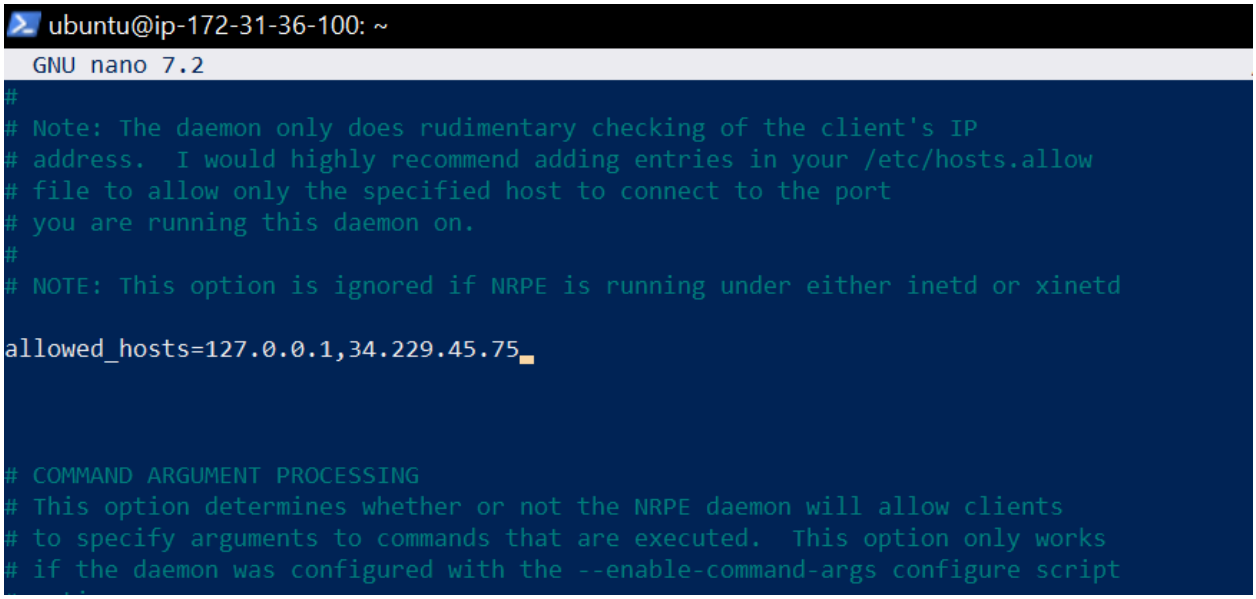
```
ubuntu@ip-172-31-36-100: ~
ubuntu@ip-172-31-36-100: $ sudo apt install gcc -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu cpp cpp-13 cpp-13-x86-64-linux-gnu cpp-x86-64-linux-gnu fontconfig-config fonts-dejavu-core fonts-dejavu-mono gcc-13 gcc-13-base
  gcc-13-x86-64-linux-gnu gcc-x86-64-linux-gnu libasan8 libatomic1 libbinutils libc-dev-bin libc-devtools libc6-dev libcc1-0 libcrypt-dev libctf-nobfd0 libctf0 libde265-0
  libdeflate0 libfontconfig1 libgcc-13-dev libgd3 libgomp1 libgprofng0 libheif-plugin-aomdec libheif-plugin-aomenc libheif-plugin-libde265 libheif1 libhwasan0 libisl23 libitm1 libjbig0
  libjpeg-turbo8 libjpeg8 liblerc4 liblsan0 libmpc3 libquadmath0 libsframe1 libsharpyuv0 libtiff6 libtsan2 libubsan1 libwebp7 libxpm4 linux-libc-dev manpages-dev rpcsvc-proto
Suggested packages:
  binutils-doc gprofng-gui cpp-doc gcc-13-locales cpp-13-doc gcc-multilib make autoconf automake libtool flex bison gdb gcc-doc gcc-13-multilib gcc-13-doc gdb-x86-64-linux-gnu glibc-doc
  libgd-tools libheif-plugin-x265 libheif-plugin-ffmpegdec libheif-plugin-jpegdec libheif-plugin-jpegenc libheif-plugin-j2kdec libheif-plugin-j2kenc libheif-plugin-rav1e
  libheif-plugin-svtenc
The following NEW packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu cpp cpp-13 cpp-13-x86-64-linux-gnu cpp-x86-64-linux-gnu fontconfig-config fonts-dejavu-core fonts-dejavu-mono gcc gcc-13 gcc-13-base
  gcc-13-x86-64-linux-gnu gcc-x86-64-linux-gnu libaom3 libasan8 libatomic1 libbinutils libc-dev-bin libc-devtools libc6-dev libcc1-0 libcrypt-dev libctf-nobfd0 libctf0 libde265-0
  libdeflate0 libfontconfig1 libgcc-13-dev libgd3 libgomp1 libgprofng0 libheif-plugin-aomdec libheif-plugin-aomenc libheif-plugin-libde265 libheif1 libhwasan0 libisl23 libitm1 libjbig0
  libjpeg-turbo8 libjpeg8 liblerc4 liblsan0 libmpc3 libquadmath0 libsframe1 libsharpyuv0 libtiff6 libtsan2 libubsan1 libwebp7 libxpm4 linux-libc-dev manpages-dev rpcsvc-proto
0 upgraded, 57 newly installed, 0 to remove and 6 not upgraded.
Need to get 62.8 MB of archives.
After this operation, 222 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 binutils-common amd64 2.42-4ubuntu2 [239 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libsframe1 amd64 2.42-4ubuntu2 [14.8 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libbinutils amd64 2.42-4ubuntu2 [572 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libctf-nobfd0 amd64 2.42-4ubuntu2 [97.1 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libctf0 amd64 2.42-4ubuntu2 [94.5 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libgprofng0 amd64 2.42-4ubuntu2 [851 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 binutils-x86-64-linux-gnu amd64 2.42-4ubuntu2 [2469 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 binutils amd64 2.42-4ubuntu2 [18.0 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 gcc-13-base amd64 13.2.0-23ubuntu4 [49.0 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libisl23 amd64 0.26-3build1 [680 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libmpc3 amd64 1.3.1-1build1 [54.5 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 cpp-13-x86-64-linux-gnu amd64 13.2.0-23ubuntu4 [11.2 MB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 cpp-13 amd64 13.2.0-23ubuntu4 [1032 B]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 cpp-x86-64-linux-gnu amd64 4:13.2.0-7ubuntu1 [5326 B]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 cpp amd64 4:13.2.0-7ubuntu1 [22.4 kB]
Get:16 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 fonts-dejavu-mono all 2.37-8 [502 kB]
Get:17 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 fonts-dejavu-core all 2.37-8 [835 kB]
Get:18 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 fontconfig-config amd64 2.15.0-1.1ubuntu2 [37.3 kB]
Get:19 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libcc1-0 amd64 14-20240412-0ubuntu1 [47.7 kB]
Get:20 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libgomp1 amd64 14-20240412-0ubuntu1 [147 kB]
Get:21 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libitm1 amd64 14-20240412-0ubuntu1 [28.9 kB]
```

```
ubuntu@ip-172-31-36-100: ~
ubuntu@ip-172-31-36-100: $ sudo apt install -y nagios-nrpe-server nagios-plugins
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'monitoring-plugins' instead of 'nagios-plugins'
The following additional packages will be installed:
  libavahi-client3 libavahi-common-data libavahi-common3 libcups2t64 libdbi1t64 libldb2 libmysqlclient21 libnet-snmp-perl libpq5 libradcli4 libsmbclient0 libsnmp-base libsnmp40t64
  libtalloc2 libtdb1 libtevent0t64 liburiparser1 libwbclient0 monitoring-plugins-basic monitoring-plugins-common monitoring-plugins-standard mysql-common python3-gpg python3-ldb
  python3-markdown python3-samba python3-talloc python3-tdb rpcbind samba-common samba-common-bin samba-dsdb-modules samba-libs smbclient snmp
Suggested packages:
  cups-common libcrypt-des-perl libdigest-hmac-perl libio-socket-inet6-perl snmp-mibs-downloader icinga2 nagios-plugins-contrib fping postfix | sendmail-bin | exim4-daemon-heavy
  | exim4-daemon-light qstat xinetd | inetd python-markdown-doc heimdal-clients pytho3-dnspython cifs-utils
The following NEW packages will be installed:
  libavahi-client3 libavahi-common-data libavahi-common3 libcups2t64 libdbi1t64 libldb2 libmysqlclient21 libnet-snmp-perl libpq5 libradcli4 libsmbclient0 libsnmp-base libsnmp40t64
  libtalloc2 libtdb1 libtevent0t64 liburiparser1 libwbclient0 monitoring-plugins monitoring-plugins-basic monitoring-plugins-common monitoring-plugins-standard mysql-common
  nagios-nrpe-server python3-gpg python3-ldb python3-markdown python3-samba python3-talloc python3-tdb rpcbind samba-common samba-common-bin samba-dsdb-modules samba-libs smbclient snmp
0 upgraded, 37 newly installed, 0 to remove and 6 not upgraded.
Need to get 16.1 MB of archives.
After this operation, 72.0 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 nagios-nrpe-server amd64 4.1.0-1ubuntu3 [356 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 rpcbind amd64 1.2.6-7ubuntu2 [46.5 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libavahi-common-data amd64 0.8-13ubuntu6 [29.7 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libavahi-common3 amd64 0.8-13ubuntu6 [23.3 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libavahi-client3 amd64 0.8-13ubuntu6 [26.8 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 libcups2t64 amd64 2.4.7-1.2ubuntu7.3 [272 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libdbi1t64 amd64 0.9.0-6.1build1 [25.7 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libtalloc2 amd64 2.4.2-1build2 [27.3 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libtdb1 amd64 1.4.10-1build1 [46.8 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libtevent0t64 amd64 0.16.1-2build1 [42.6 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libldb2 amd64 2:2.8.0+samba4.19.5+dfsg-4ubuntu9 [187 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 mysql-common all 5.8+1.1.0build1 [6746 B]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 libmysqlclient21 amd64 8.0.39-0ubuntu0.24.04.2 [1254 kB]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libnet-snmp-perl all 6.0.1-7 [86.8 kB]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 libpq5 amd64 16.4-0ubuntu0.24.04.2 [141 kB]
Get:16 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 libradcli4 amd64 1.2.11-1build3 [40.5 kB]
Get:17 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libwbclient0 amd64 2:4.19.5+dfsg-4ubuntu9 [70.6 kB]
Get:18 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 samba-libs amd64 2:4.19.5+dfsg-4ubuntu9 [6017 kB]
Get:19 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libsmbclient0 amd64 2:4.19.5+dfsg-4ubuntu9 [62.4 kB]
Get:20 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libsnmp-base all 5.9.4+dfsg-1.1ubuntu3 [206 kB]
Get:21 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libsnmp40t64 amd64 5.9.4+dfsg-1.1ubuntu3 [1066 kB]
Get:22 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 liburiparser1 amd64 0.9.7+dfsg-2build1 [35.8 kB]
Get:23 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 python3-gpg amd64 1.18.0-4.1ubuntu4 [209 kB]
Get:24 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 python3-ldb amd64 2:2.8.0+samba4.19.5+dfsg-4ubuntu9 [41.8 kB]
Get:25 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 python3-markdown all 3.5.2-1 [72.0 kB]
```

Open nrpe.cfg file to make changes.

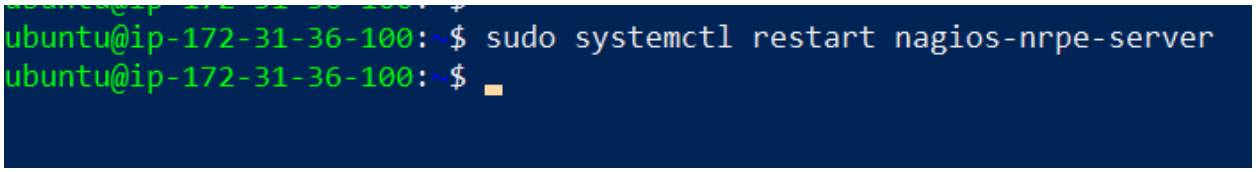
sudo nano /etc/nagios/nrpe.cfg

Under allowed_hosts, add your nagios host IP address like so



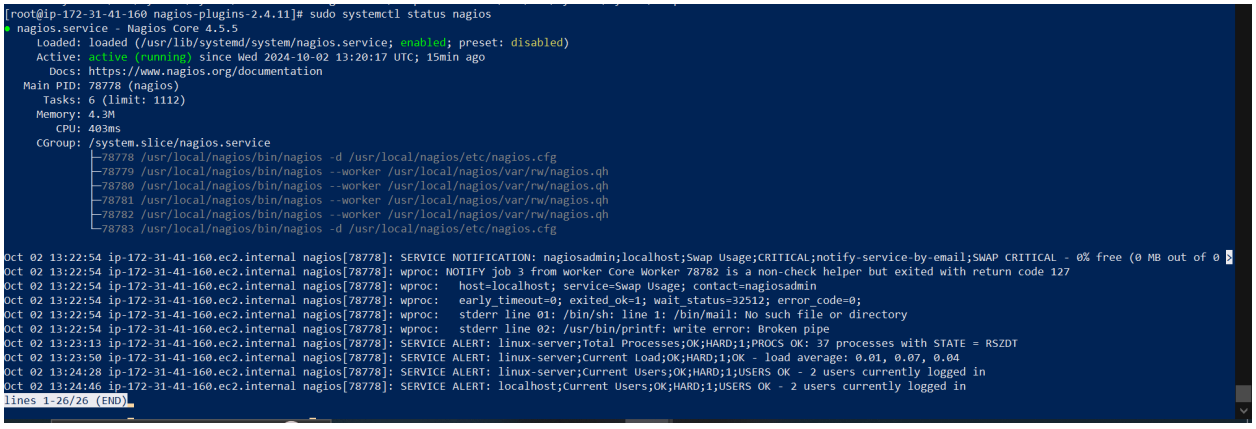
Now restart the NRPE server by this command.

sudo systemctl restart nagios-nrpe-server



Run the following command in the Nagios-host terminal

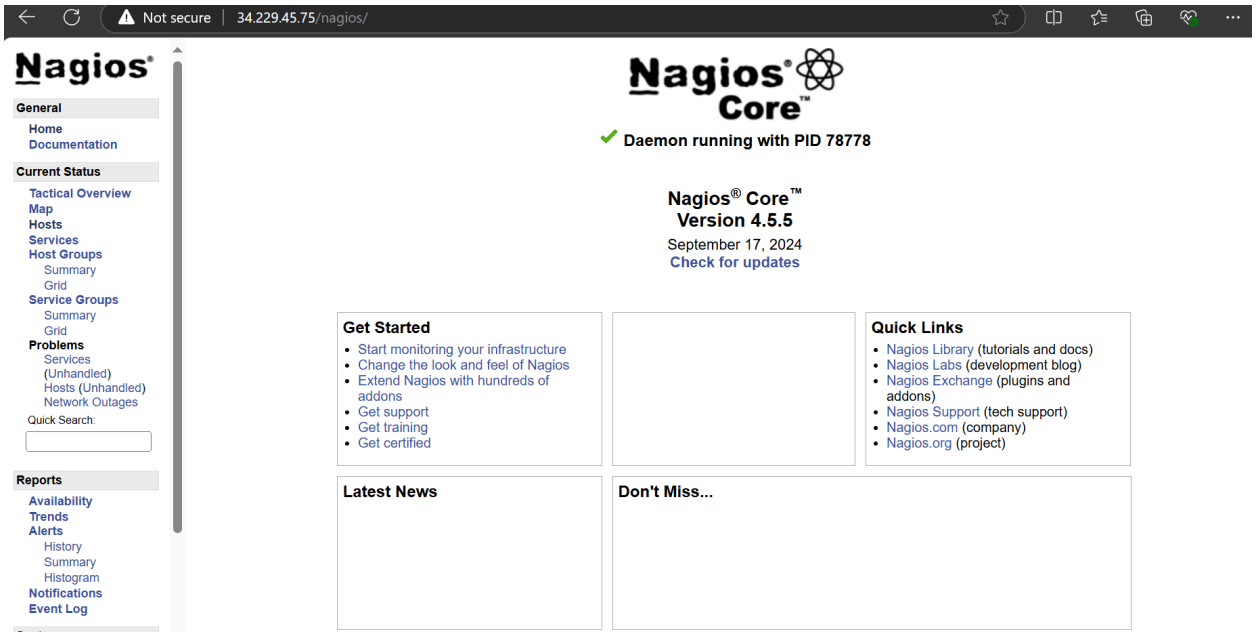
sudo systemctl status nagios



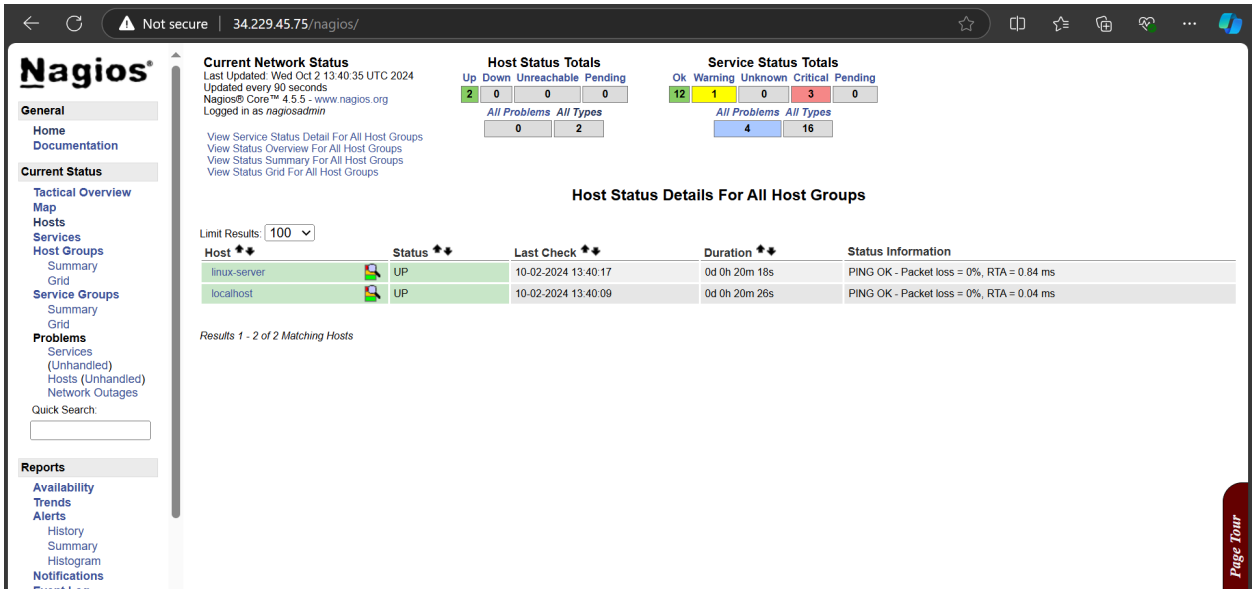
Name:Ganesh Gupta Div: D15C Roll No: 13

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

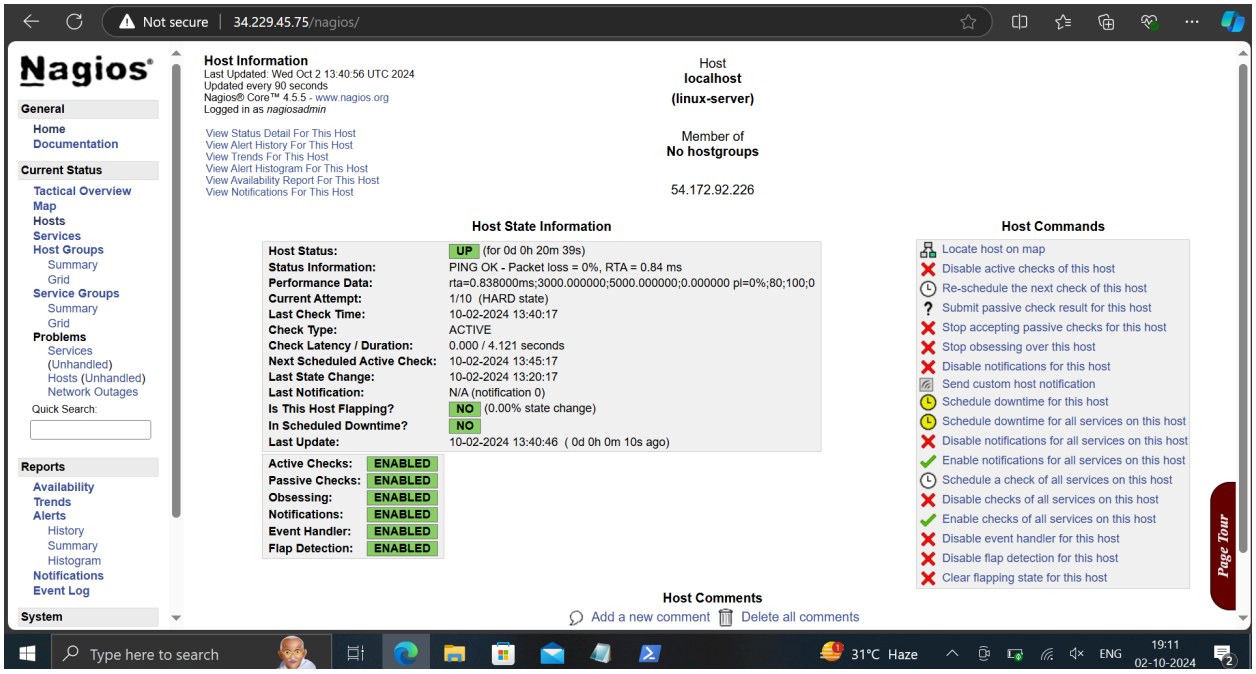


Click on Hosts.

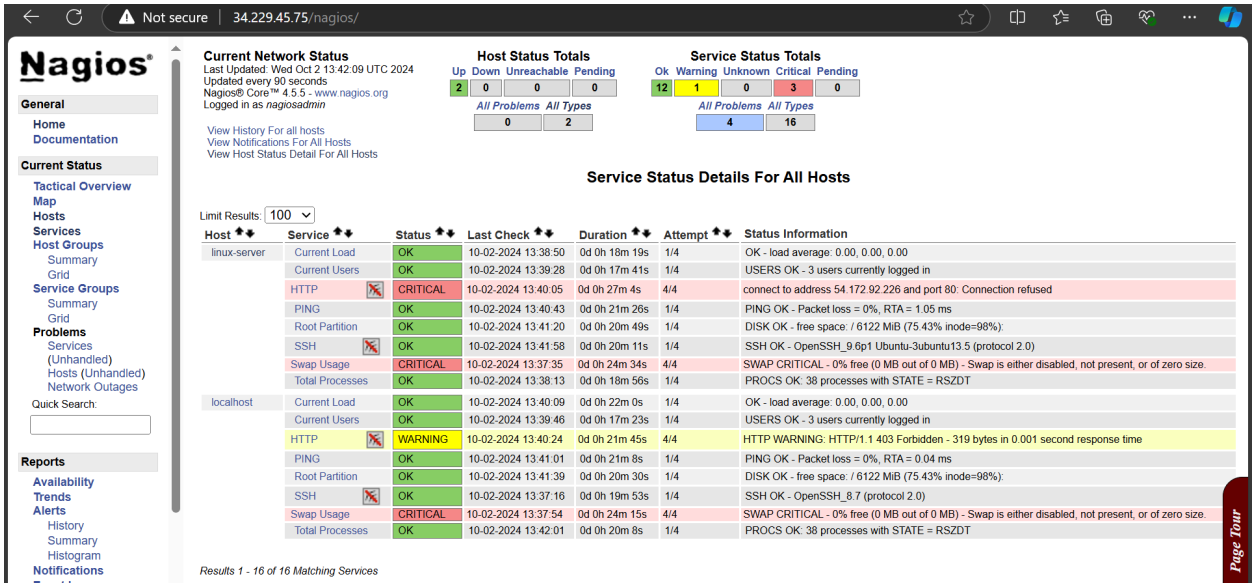


Name:Ganesh Gupta Div: D15C Roll No: 13

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

Name : Ganesh Gupta **RollNo**./Div:13/D15C

Experiment 11

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

Theory:

AWS Lambda

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

Lambda Workflow

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

AWS Lambda Functions

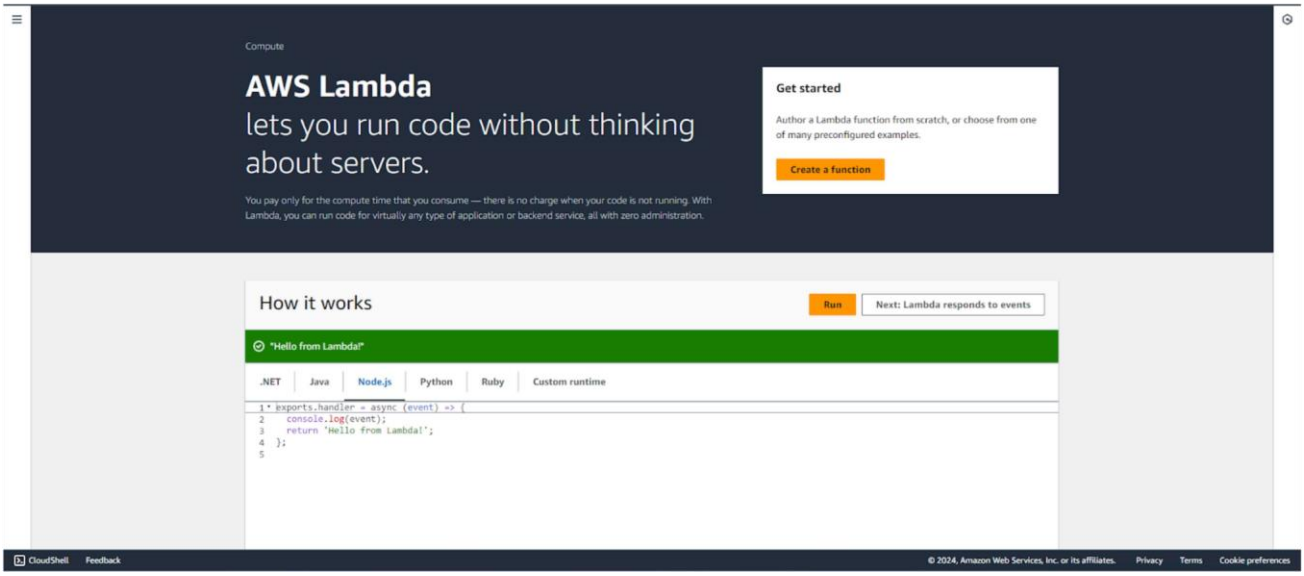
- **Python**: Great for quick development with its rich standard library and support for lightweight tasks.
- **Java**: Typically used for more complex, compute-intensive tasks. While it's robust, cold start times can be higher.
- **Node.js**: Excellent for I/O-bound tasks like handling APIs or streaming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

Prerequisites: AWS Personal/Academy Account

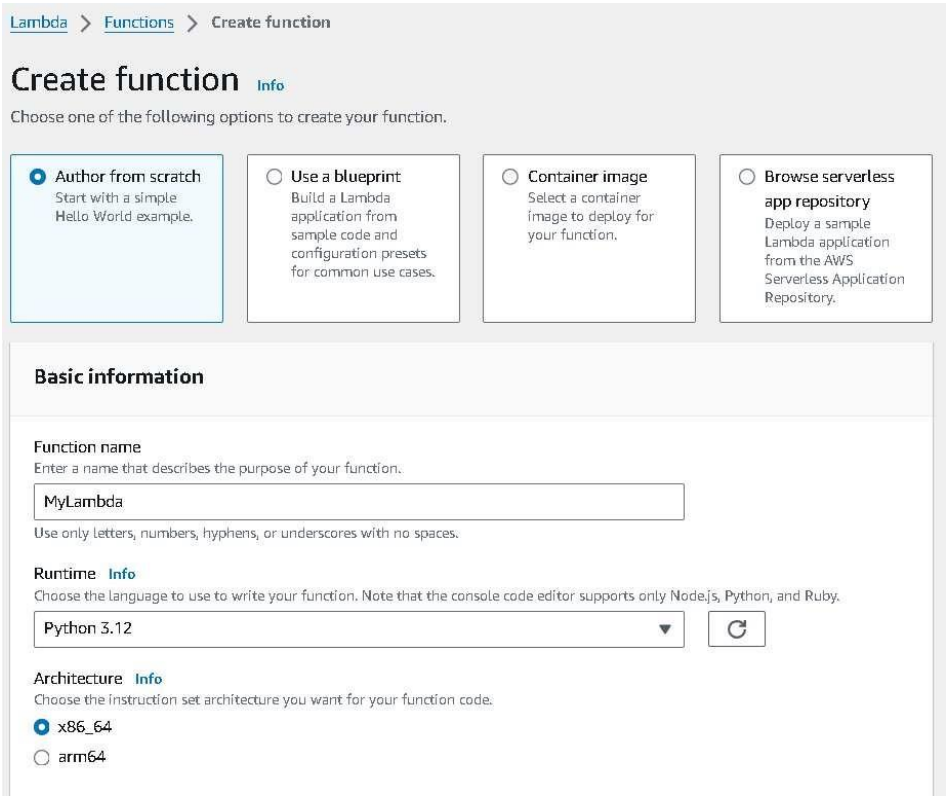
Name : Ganesh Gupta **RollNo**./Div:13/D15C

Steps To create the lambda function:

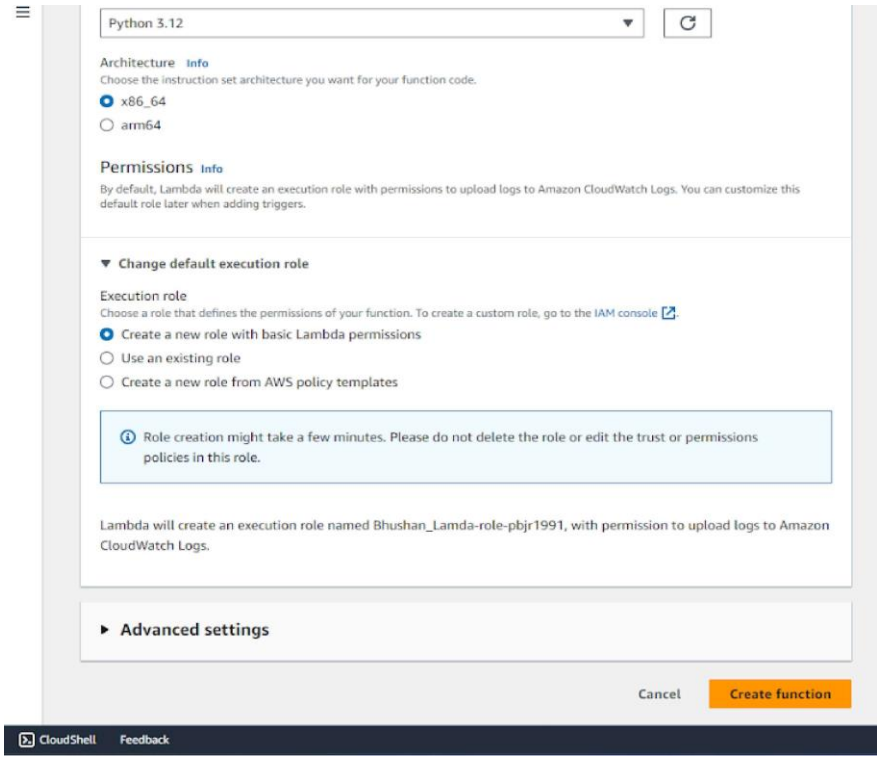
Step 1: Login to your AWS Personal/Academy Accout.Open lambda and click on create function button.

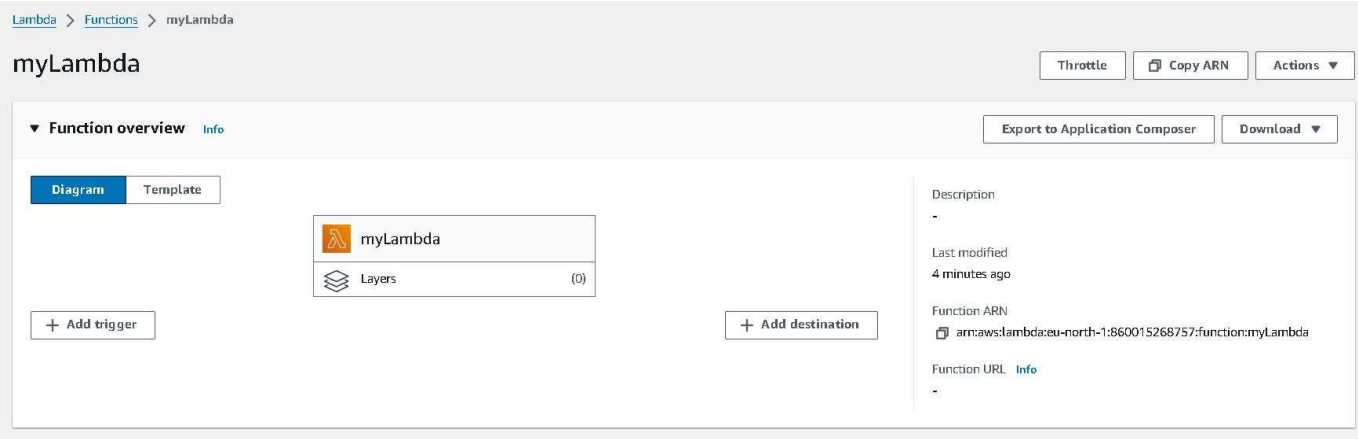


Step 2: Now Give a name to your Lambda function, Select the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby. So will select Python 3.12, Architecture as x86, and Execution role to Create a new role with basic Lambda permissions.



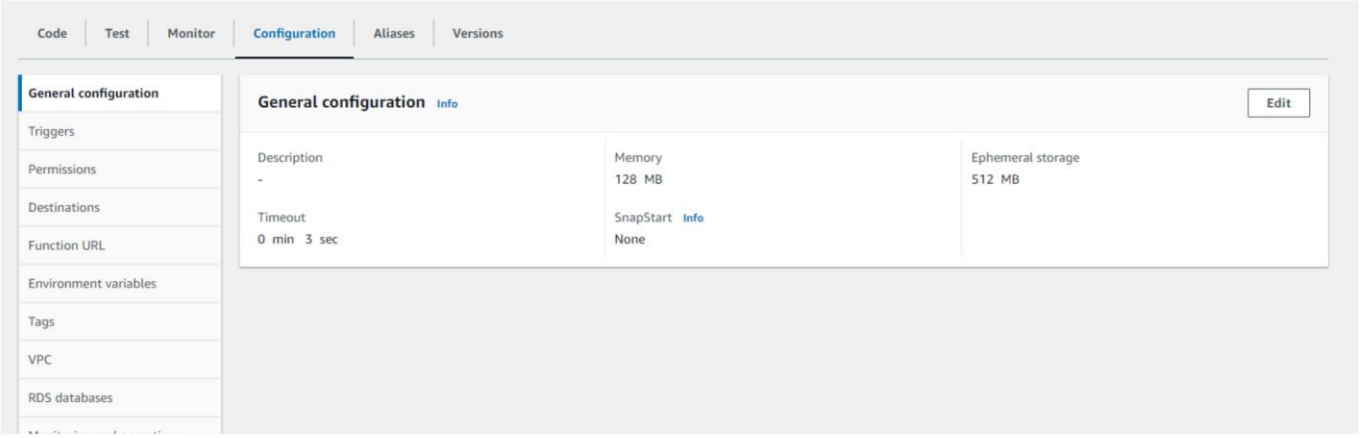
Name : Ganesh Gupta **RollNo**./Div:13/D15C



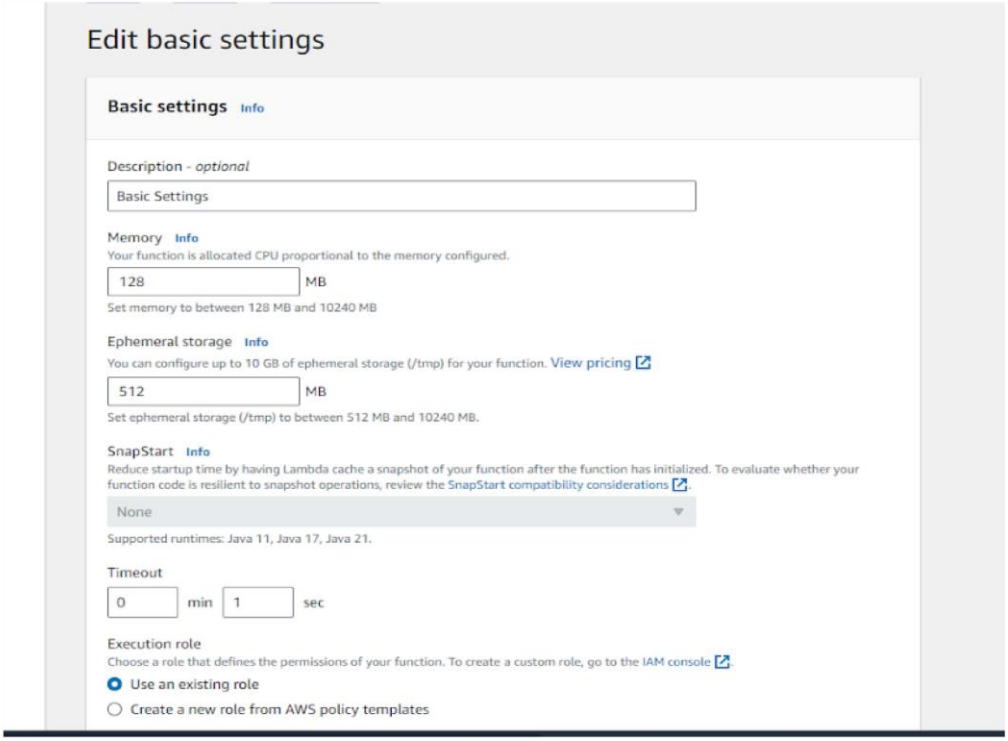


Name : Ganesh Gupta **RollNo**./Div:13/D15C

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

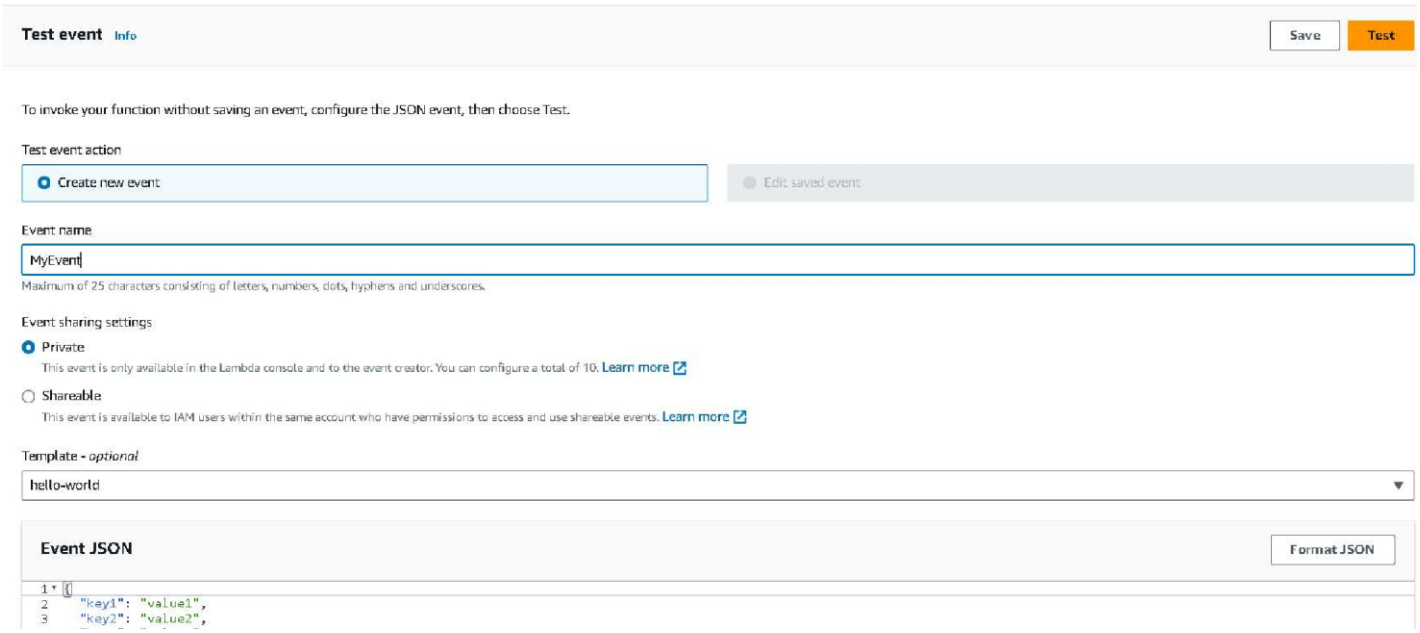


Here, you can enter a description and change Memory and Timeout. I've changed the Timeout period to 1 sec since that is sufficient for now.

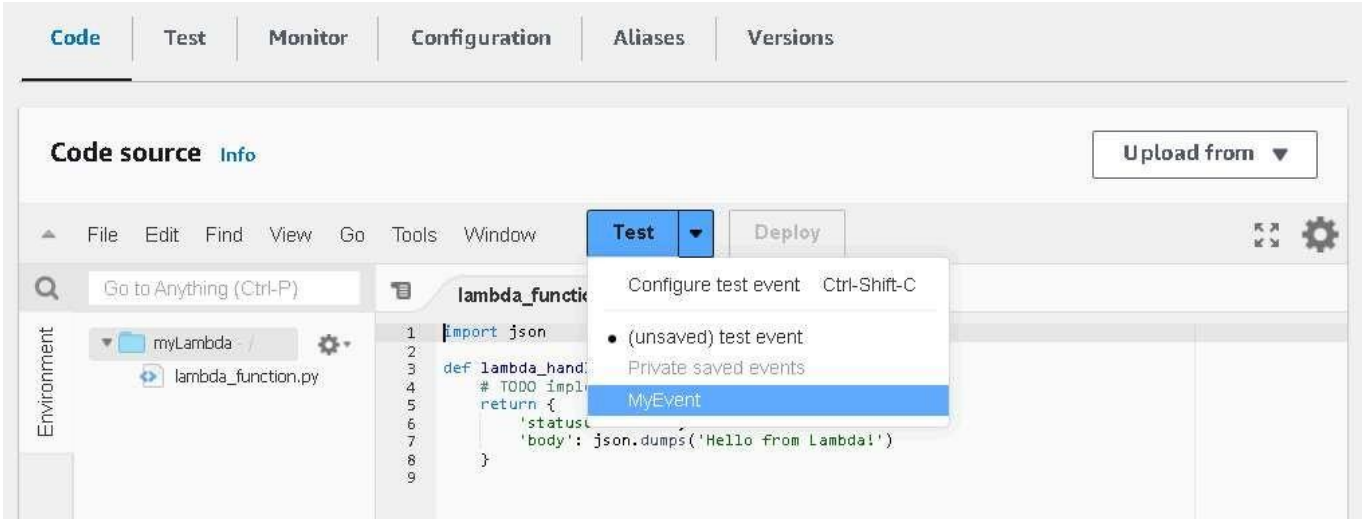


Name : Ganesh Gupta **RollNo**./Div:13/D15C

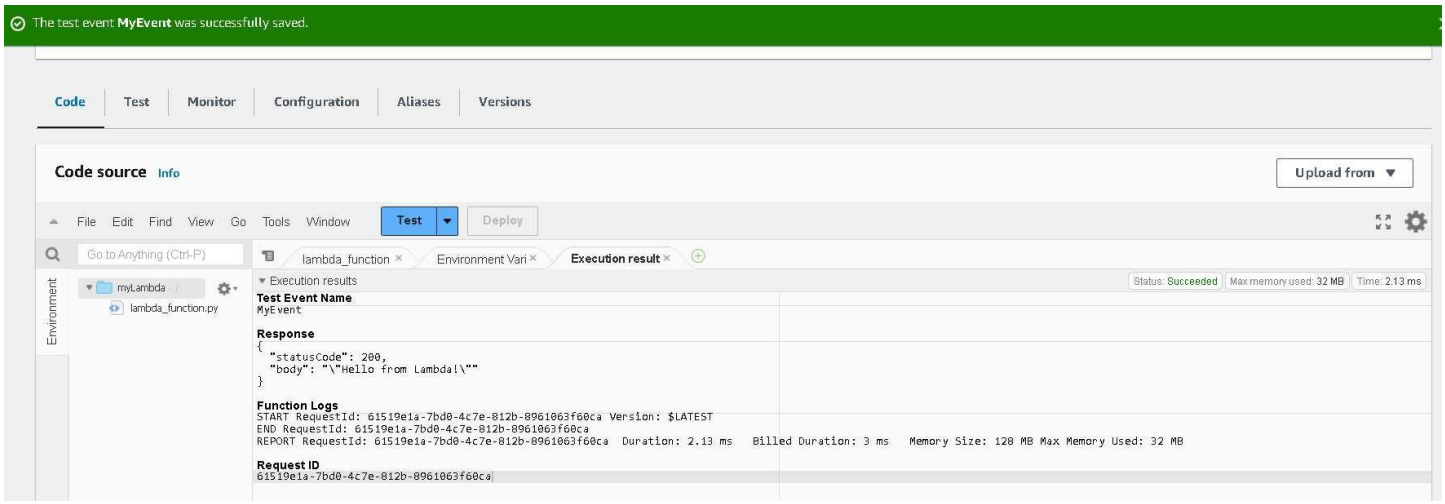
Step 3: Now Click on the Test tab then select Create a new event, give a name to the event and select Event Sharing to private, and select hello-world template.



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

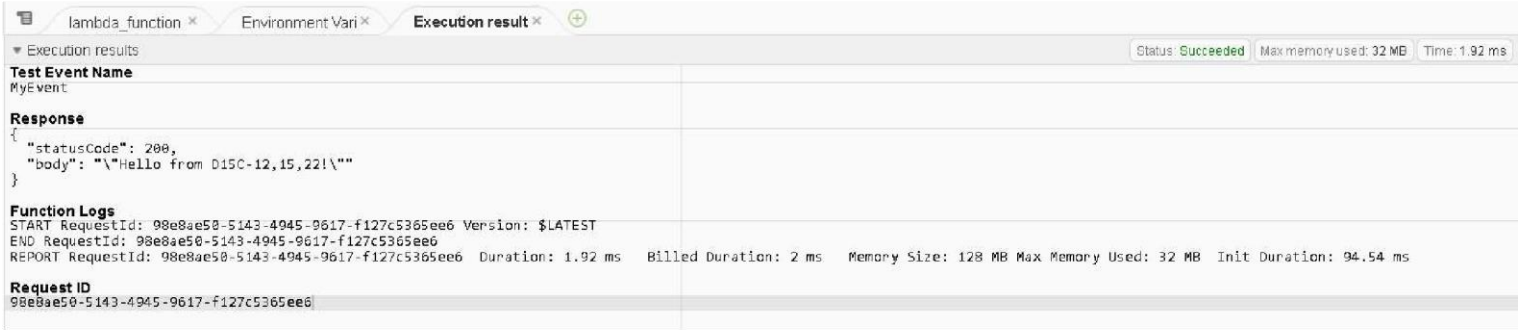


Name : Ganesh Gupta **RollNo**./Div:13/D15C



Step 5: You can edit your lambda function code. I have changed the code to display the new String.

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



Conclusion:

In this experiment, we successfully implemented an AWS Lambda function, covering all the key steps involved. Starting with the function's setup in Python, we configured essential settings such as adjusting the timeout to 1 second. A test event was then created, followed by deploying the function and verifying its output. We also made code modifications to the Lambda function, redeployed it, and observed the real-time effects of these changes. This hands-on experience highlighted the ease and adaptability of AWS Lambda for building serverless applications, enabling developers to concentrate on writing code while AWS handles infrastructure and scalability.

Experiment 12

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

Theory:

AWS Lambda and S3 Integration:

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

Workflow:

1. **Create an S3 Bucket:**
   - First, create an S3 bucket that will store the objects. This bucket will act as the trigger source for the Lambda function.

2. **Create the Lambda Function:**
   - Set up a new Lambda function using AWS Lambda’s console. You can choose a runtime environment like Python, Node.js, or Java.
   - Write code that logs a message like “An Image has been added” when triggered.

3. **Set Up Permissions:**
   - Ensure that the Lambda function has the necessary permissions to access S3. You can do this by attaching an IAM role with policies that allow reading from the bucket and writing logs to CloudWatch.

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

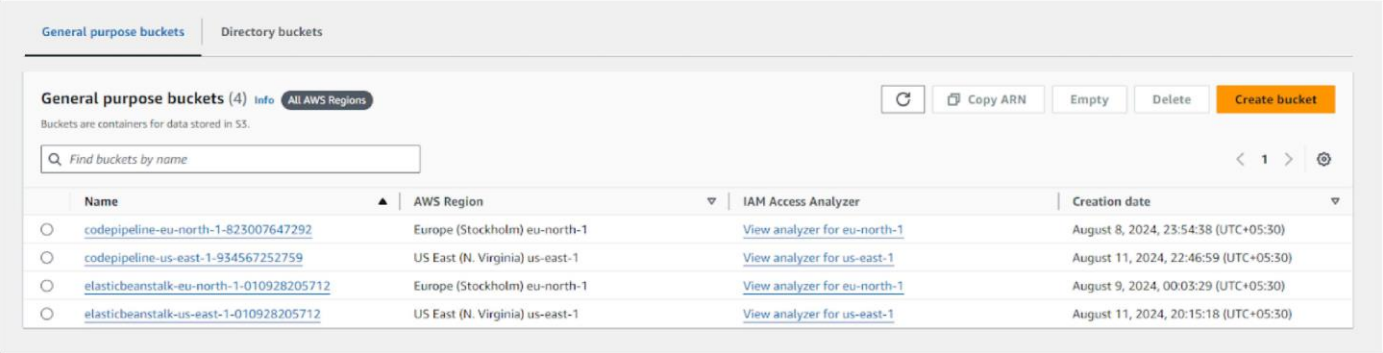
5. **Test the Setup:**
   - Upload an object (e.g., an image) to the S3 bucket to test the trigger. The Lambda function should execute and log the message “An Image has been added” in AWS CloudWatch Logs.

Prerequisites: AWS Personal Account

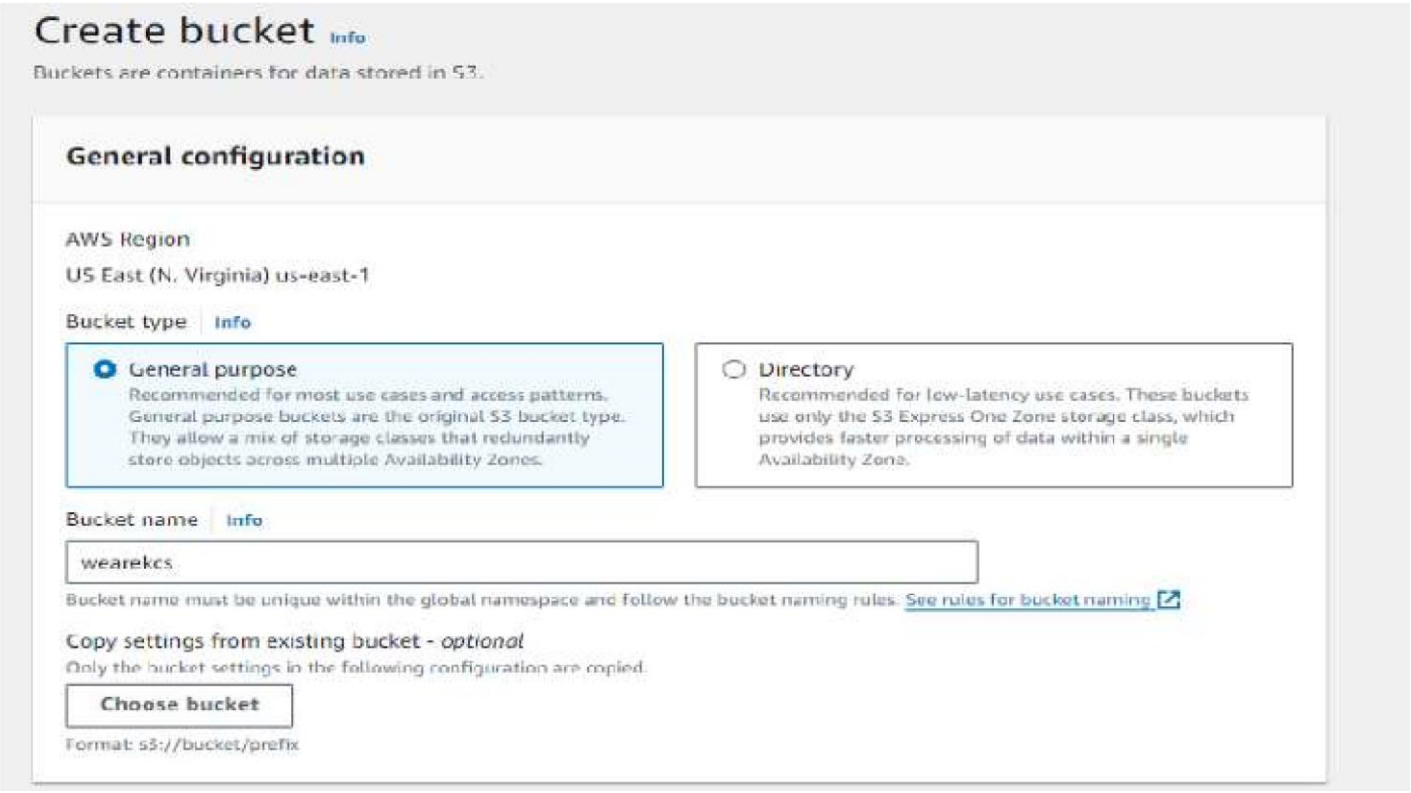
Name: Ganesh Gupta **DIV/Roll no.:D15C/13**

Steps To create the lambda function:

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



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



Name: Ganesh Gupta **DIV/Roll no.:D15C/13**

Object Ownership Info

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

- (●) ACLs disabled (recommended)
  All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.
- ( ) ACLs enabled
  Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Object Ownership

Bucket owner enforced

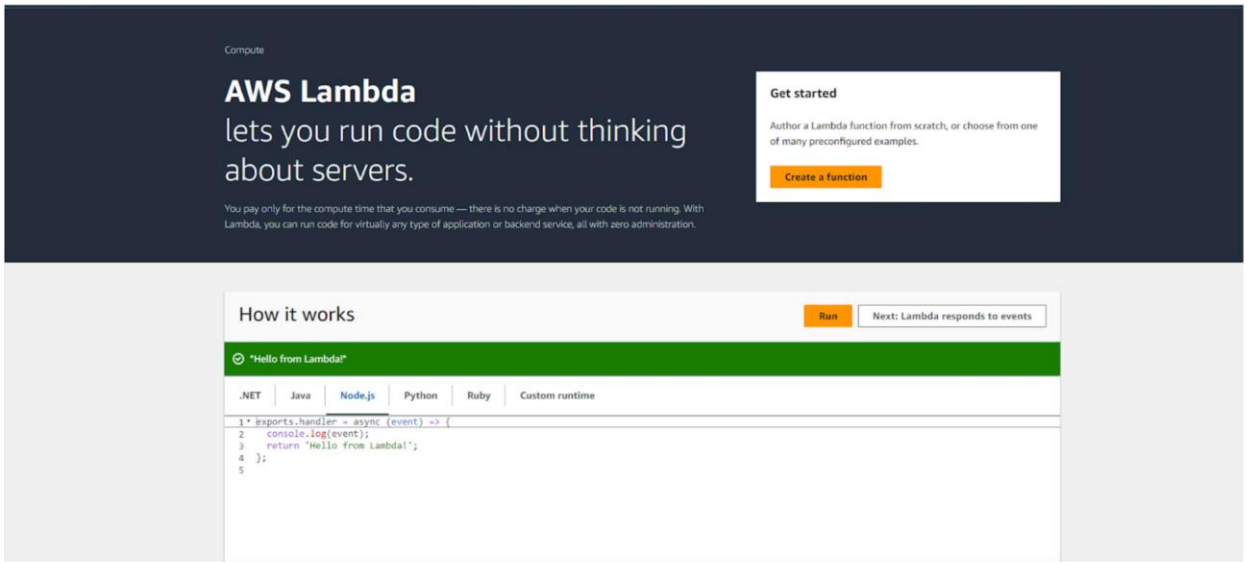
Block Public Access settings for this bucket

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

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

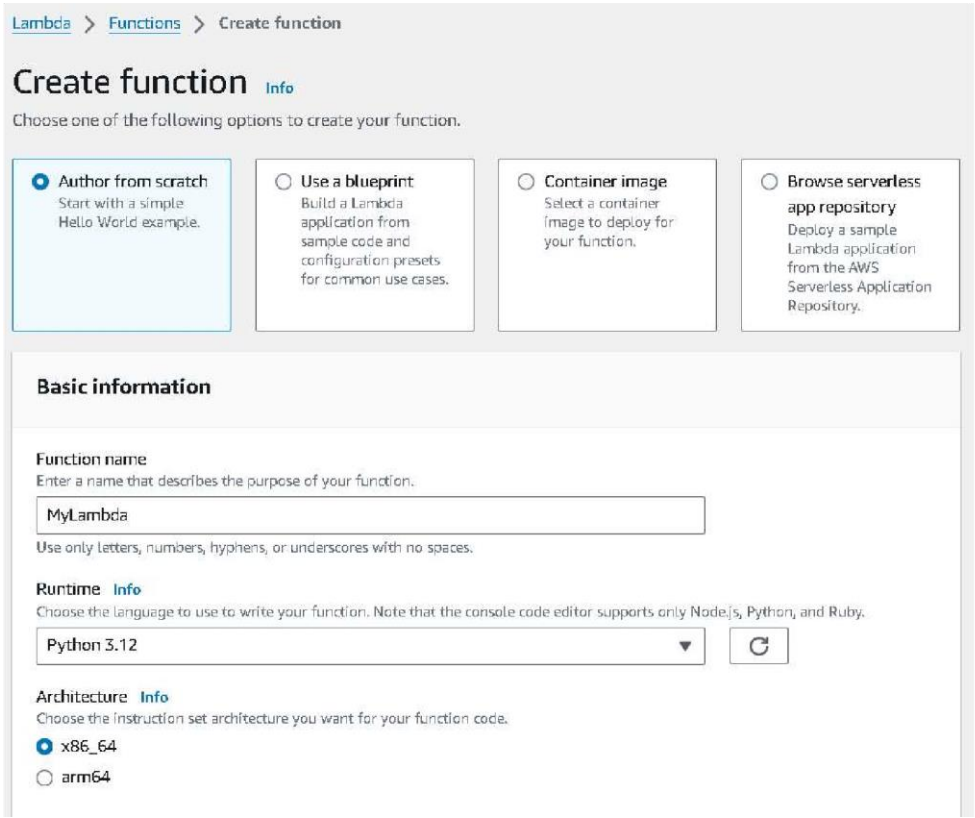
Name: Ganesh Gupta **DIV/Roll no.:D15C/13**

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

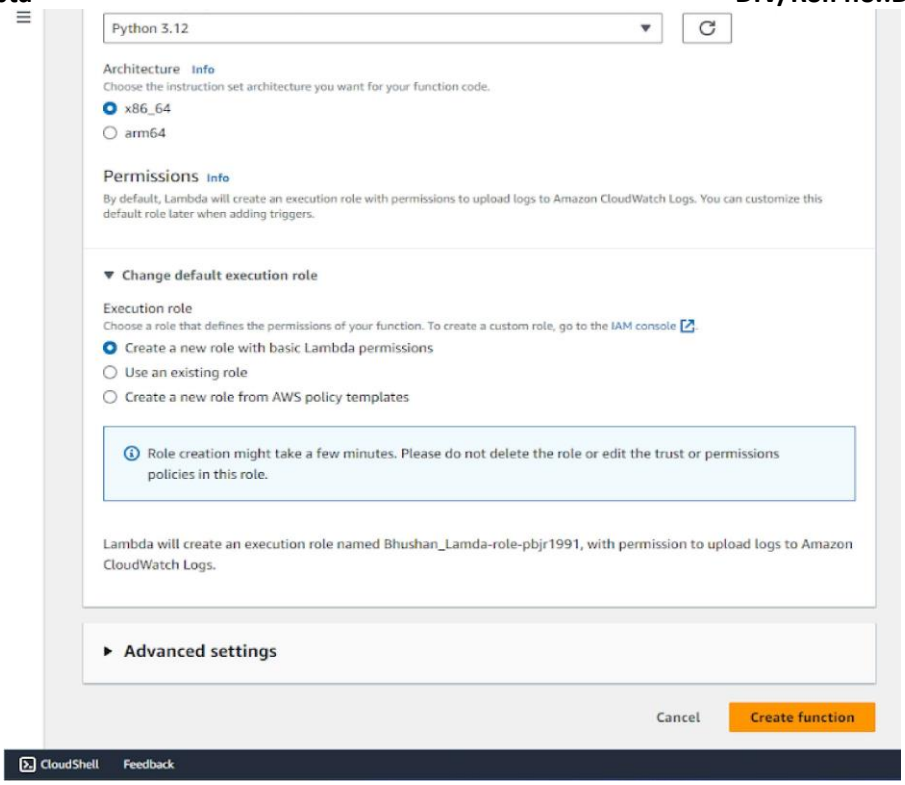


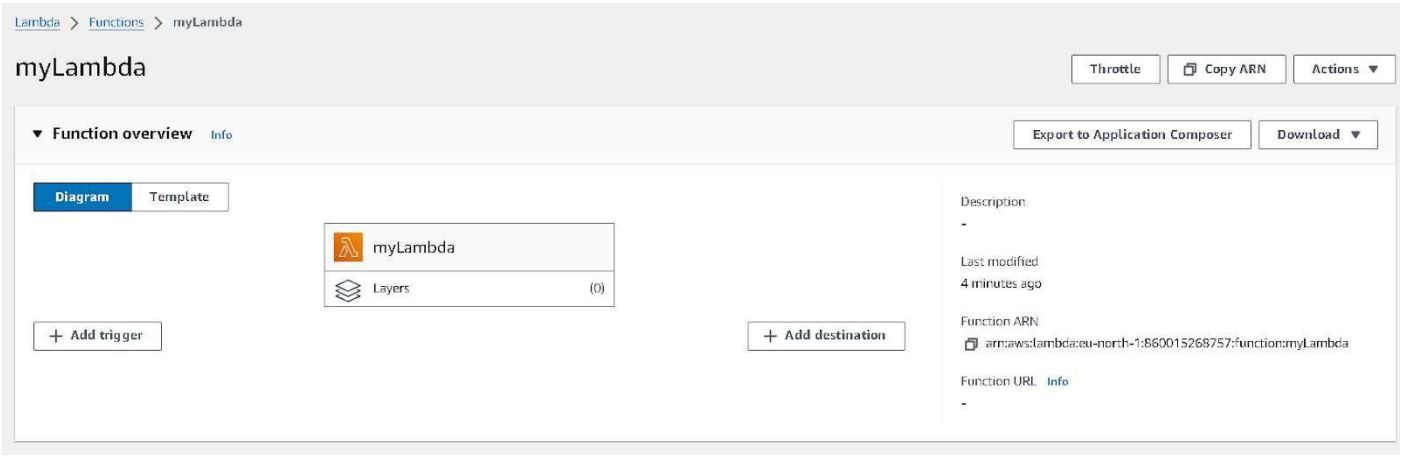
Step 4: Now Give a name to your Lambda function, Select the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby. So will select Python 3.12

, Architecture as x86, and Exceution role to Create a new role with basic Lambda permissions.

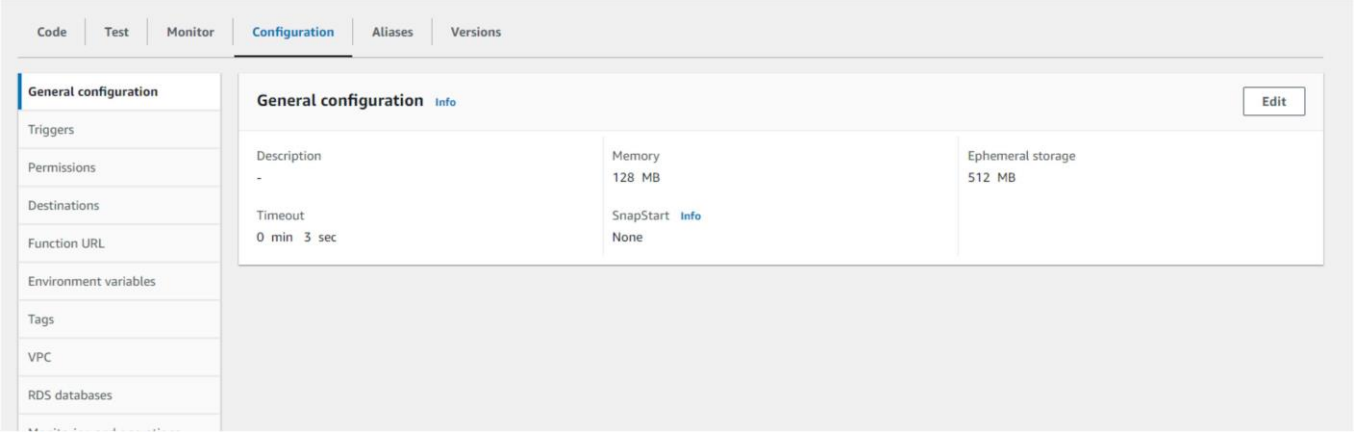


Name: Ganesh Gupta **DIV/Roll no.:D15C/13**

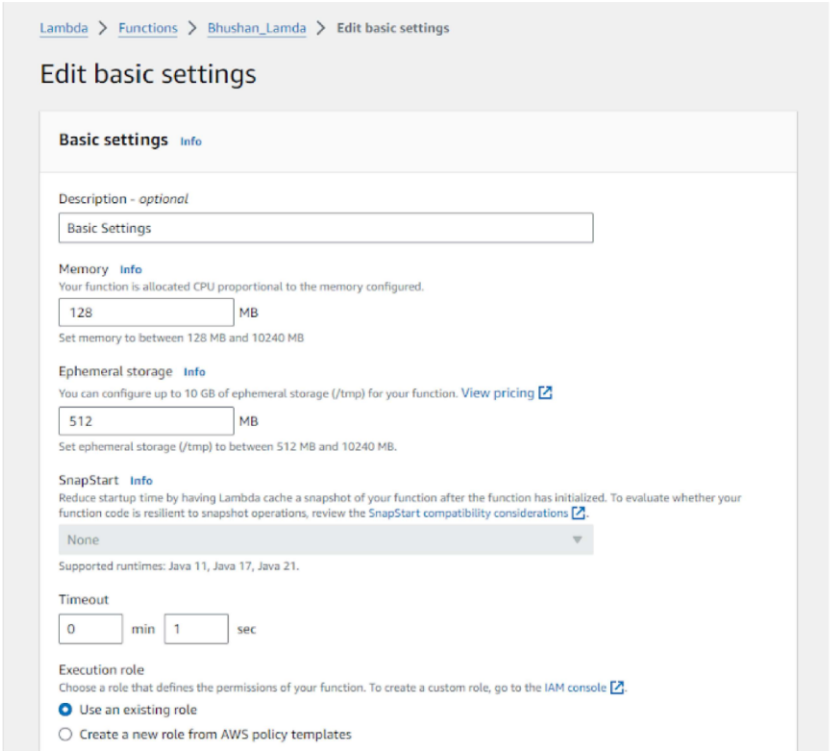




So See or Edit the basic settings go to configuration then click on edit general setting.

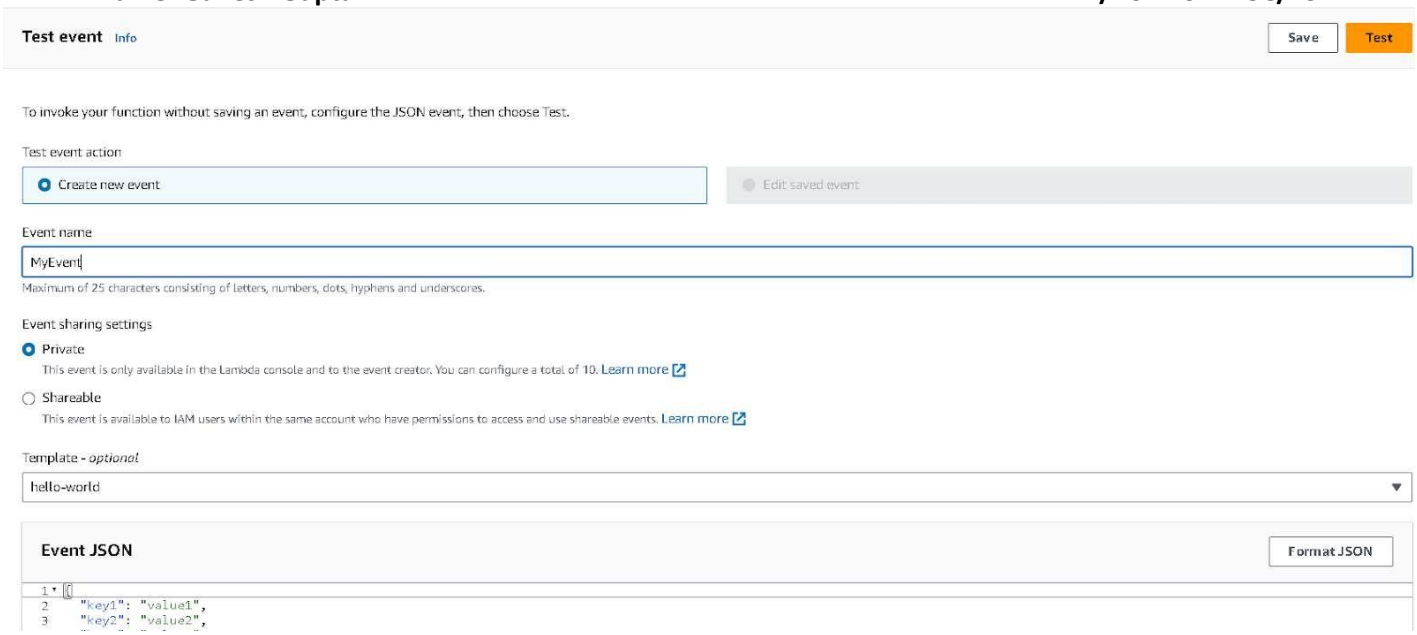


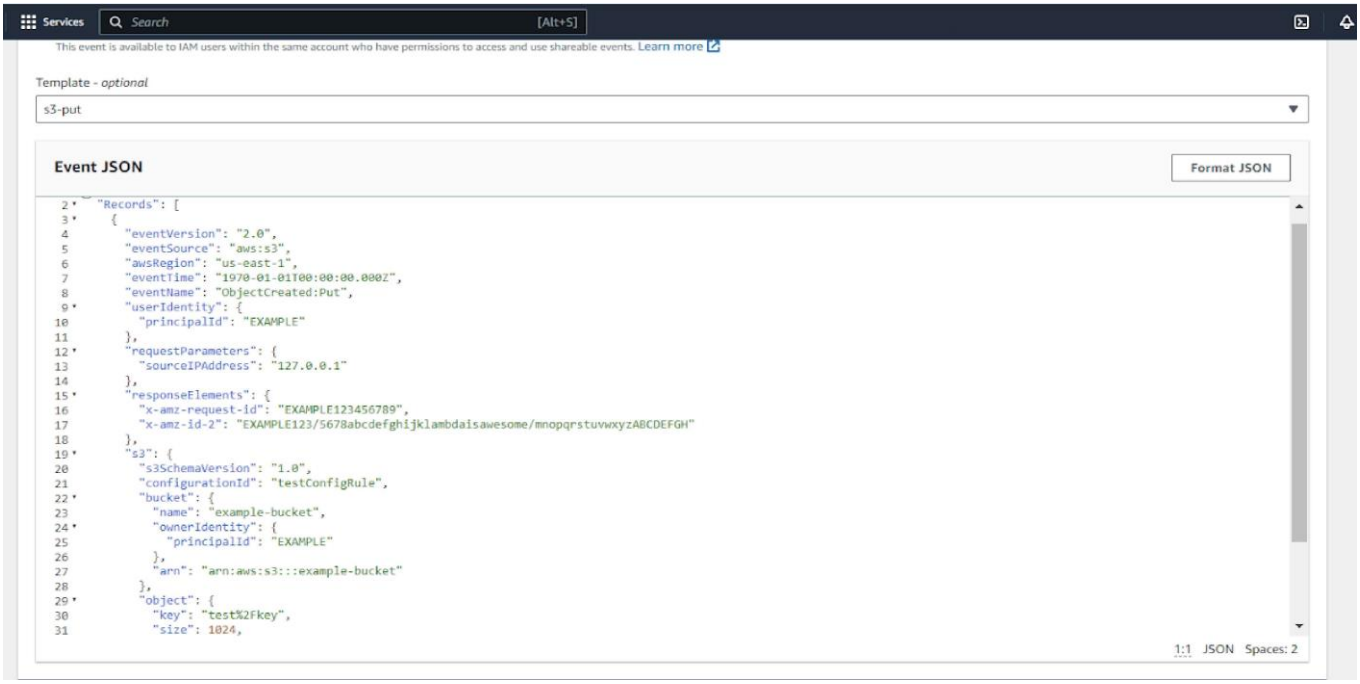
Here, you can enter a description and change Memory and Timeout. I've changed the Timeout period to 1 sec since that is sufficient for now.



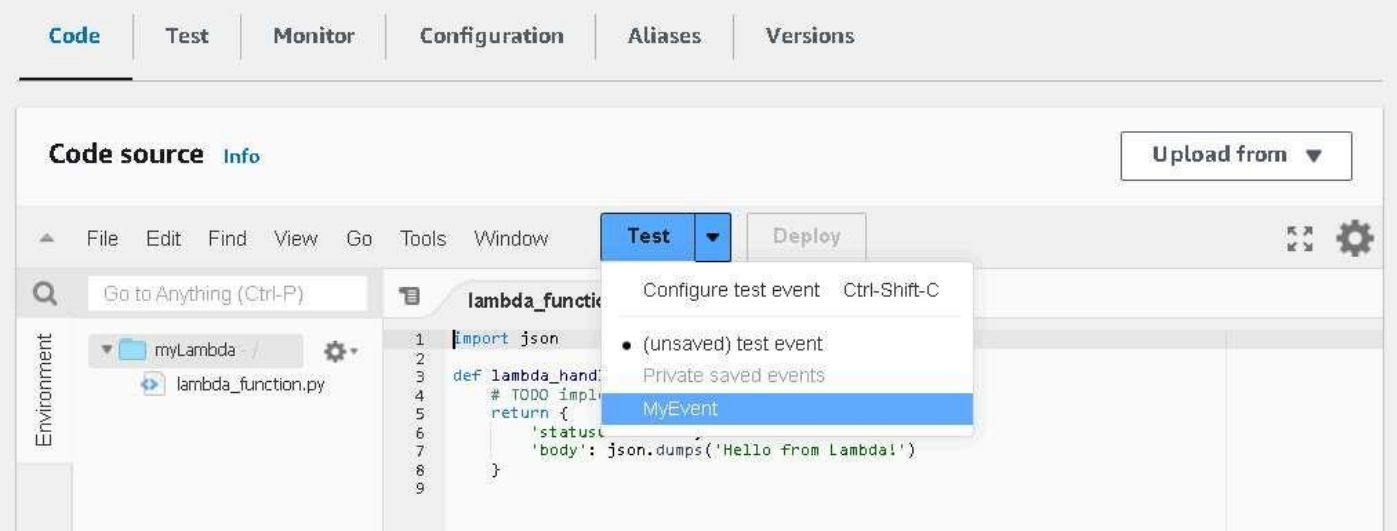
Step 5: Now Click on the Test tab then select Create a new event, give a name to the event and select Event Sharing to private, and select s3 put template.

Name: Ganesh Gupta **DIV/Roll no.:D15C/13**

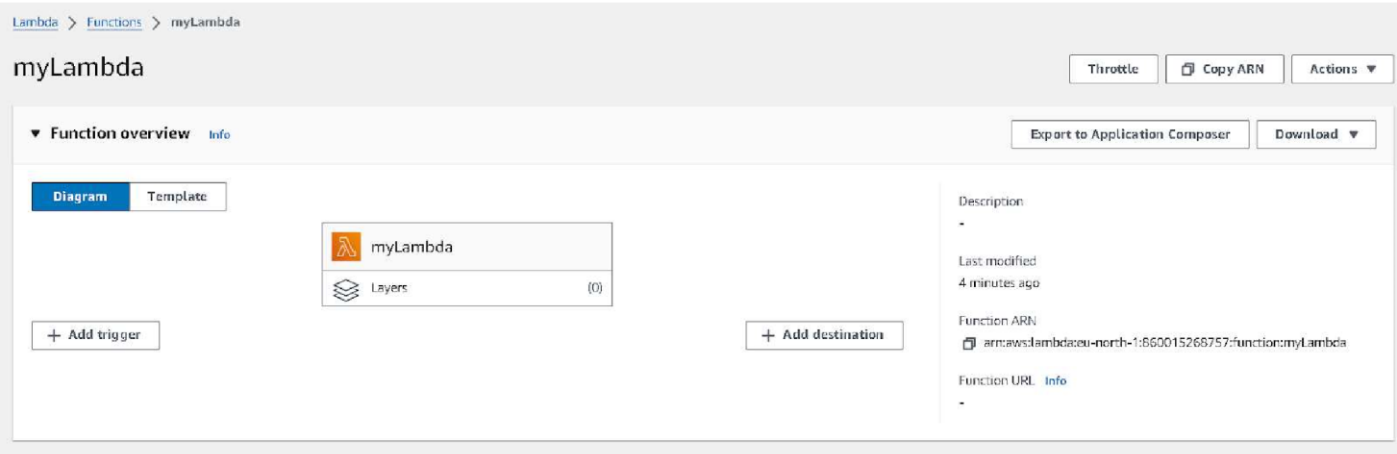




Step 6: Now In Code section select the created event from the dropdown .

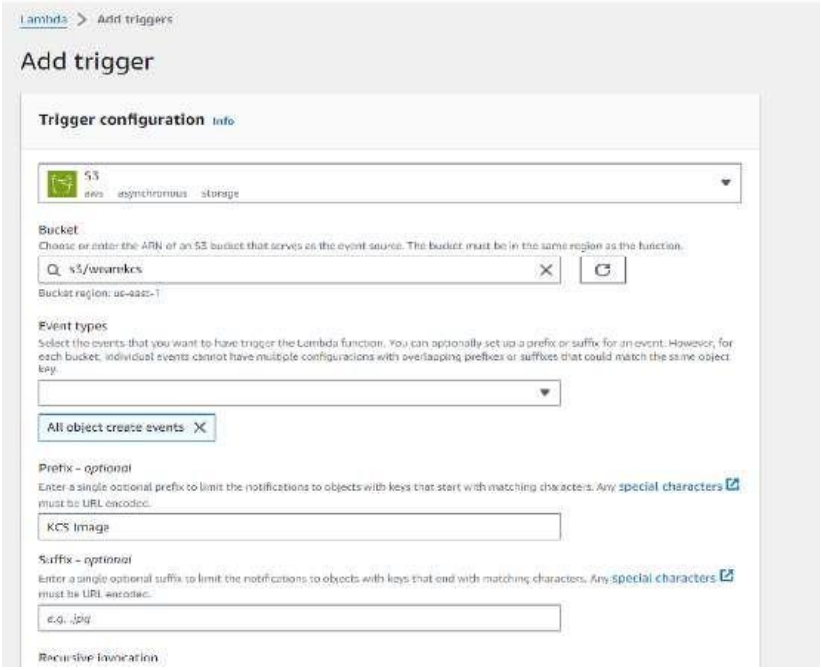


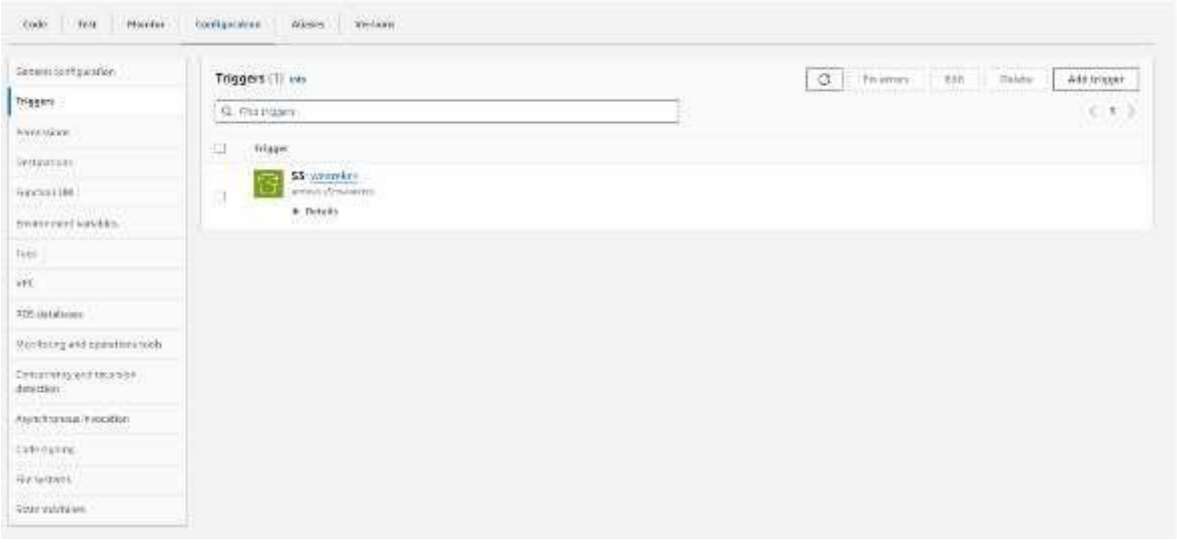
Step 7: Now In the Lambda function click on add tigger.



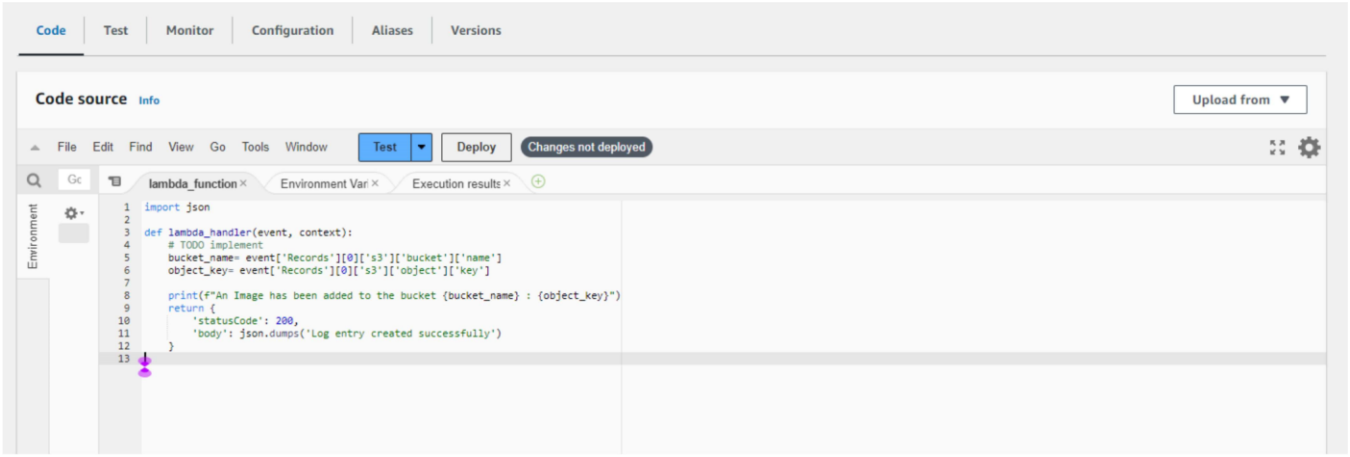
Now select the source as S3 then select the bucket name from the dropdown, keep other things to default and also you can add prefix to image.

Name: Ganesh Gupta **DIV/Roll no.:D15C/13**



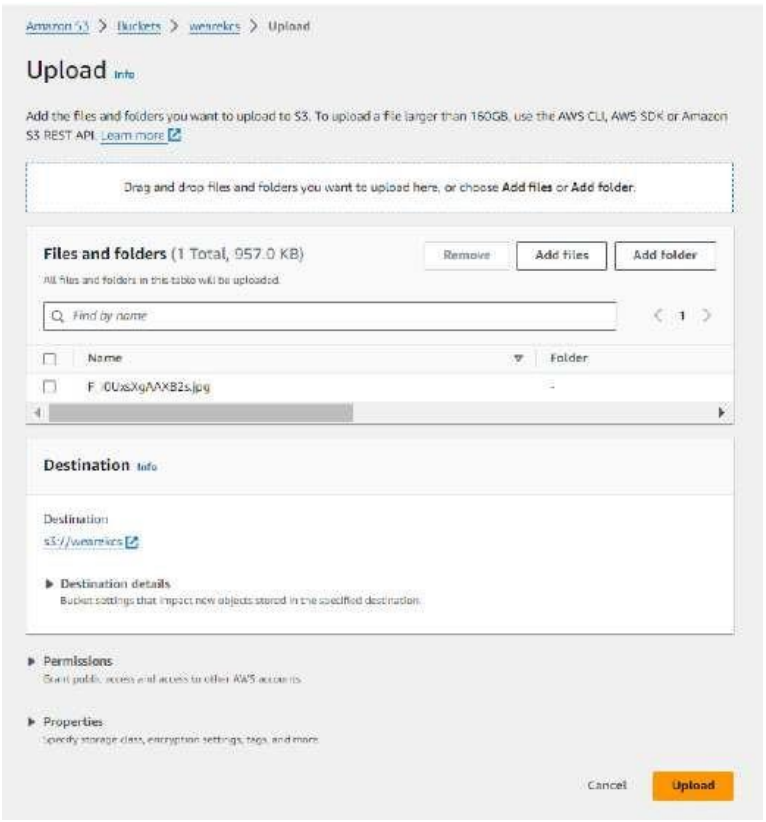


Step 8: Now Write code that logs a message like “An Image has been added” when triggered. Save the file and click on deploy.

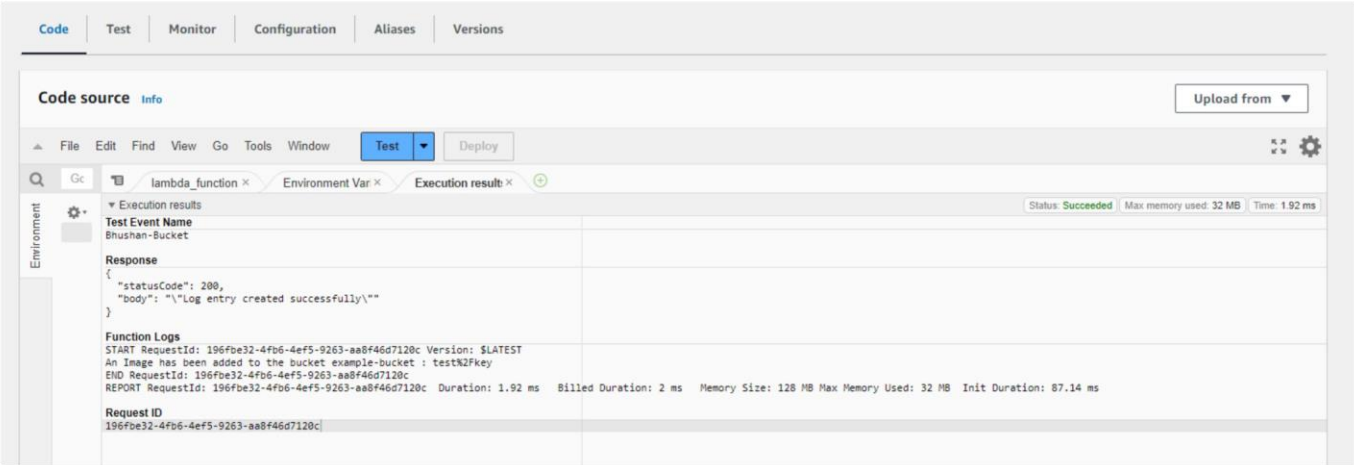


Name: Ganesh Gupta **DIV/Roll no.:D15C/13**

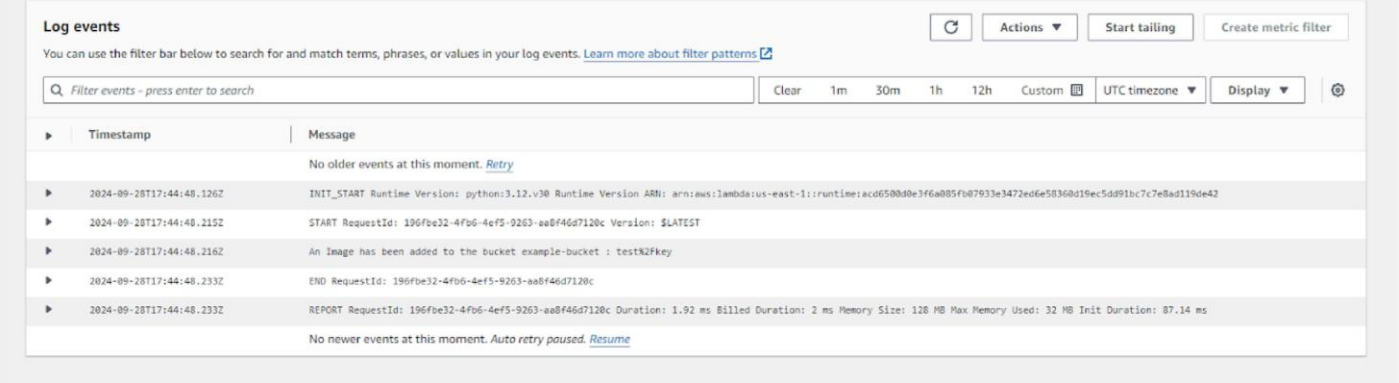
Step 9: Now upload any image to the bucket.



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



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



Conclusion:

In this experiment, we successfully created an AWS Lambda function that logs a message when an image is uploaded to an S3 bucket. It is important to note that we have to select S3-put template in the event otherwise code will give an error.The function was successfully triggered by S3 object uploads, validating the functionality of Lambda's event-driven architecture. This experiment demonstrated how Lambda can efficiently respond to S3 events and how to troubleshoot common issues with event structure.