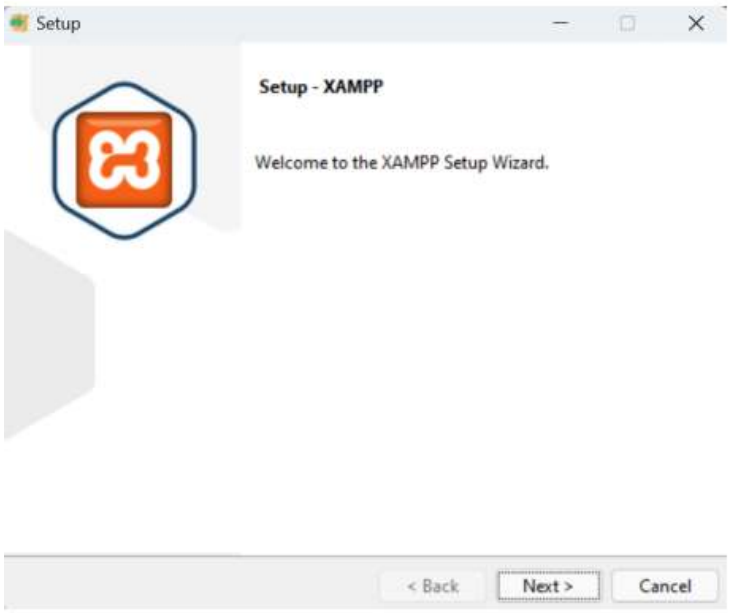
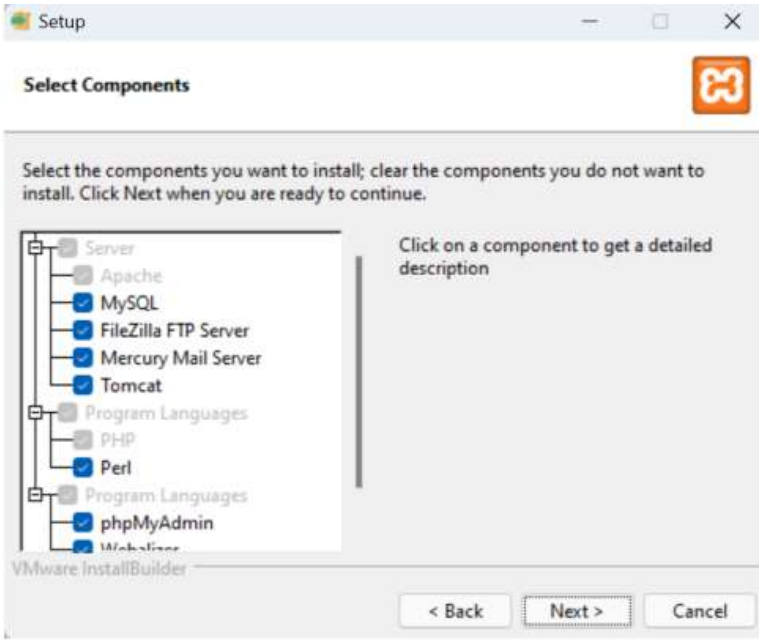
Aim: To develop a website and host it on your local machine and Amazon S3

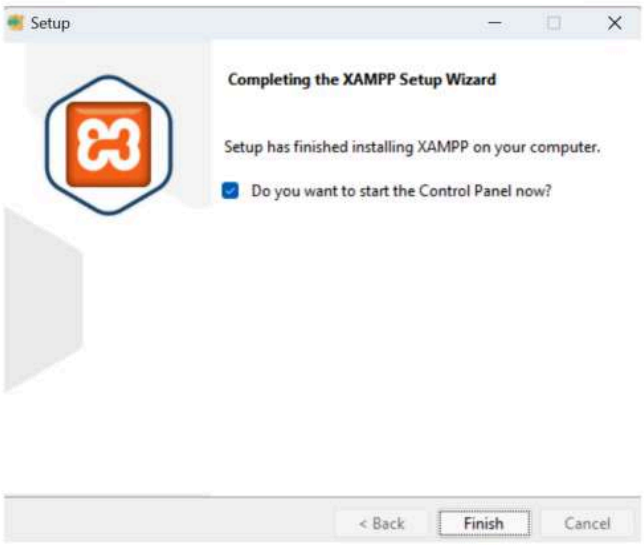
1. Hosting Website using Xampp

Install the required files from the internet and open up the setup

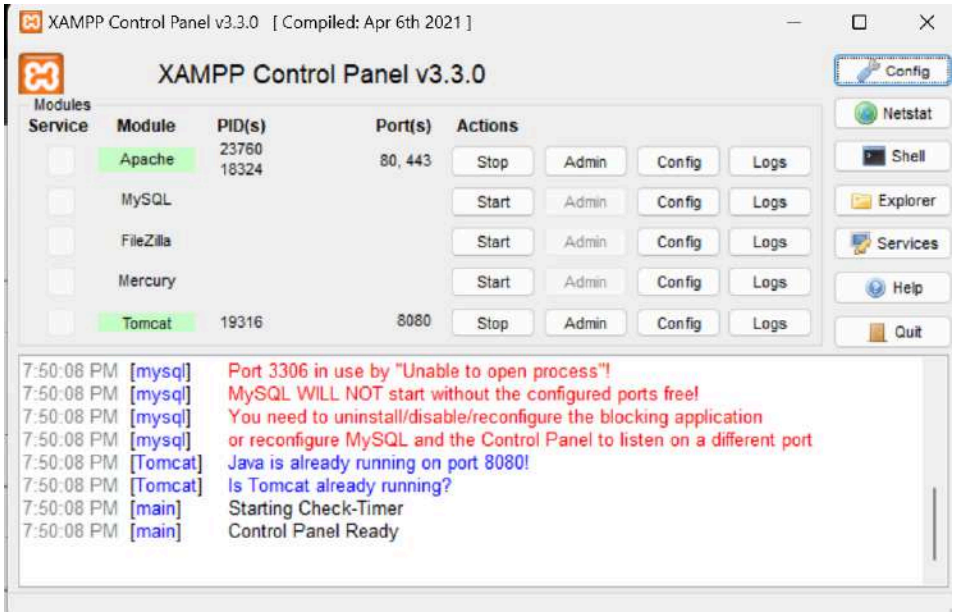


Make the changes according to your requirements





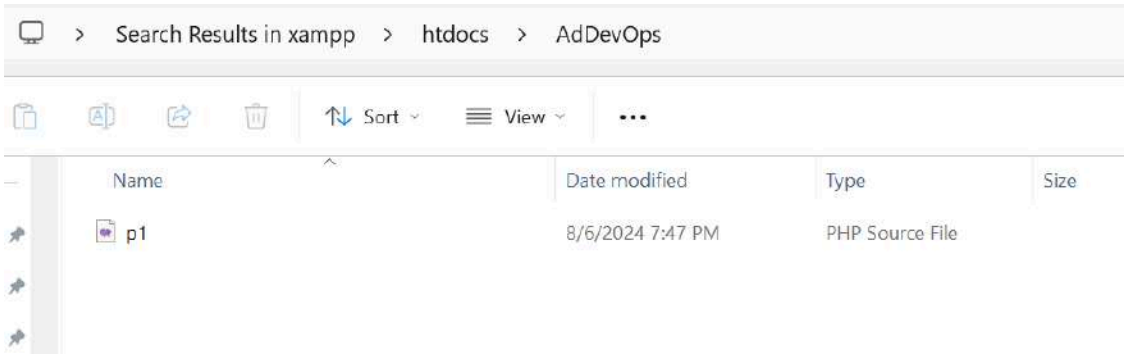
Open the control panel and click start for Apache



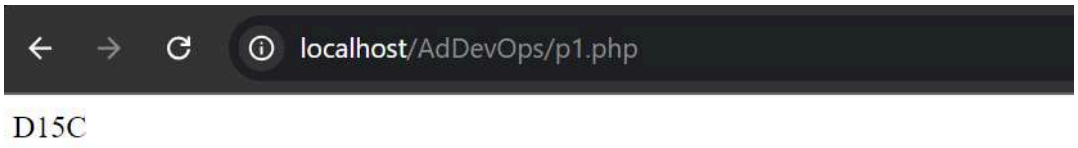
Write down your code and save it inside the htdocs folder in xampp files

C: > xampp > htdocs > AdDevOps > p1.php

```
1   <html>
2   <title> LAB 1 NIKITA CHHABRIA </title>
3   <body>
4   <?php
5   echo"D15C";
6   ?>
7
8
```

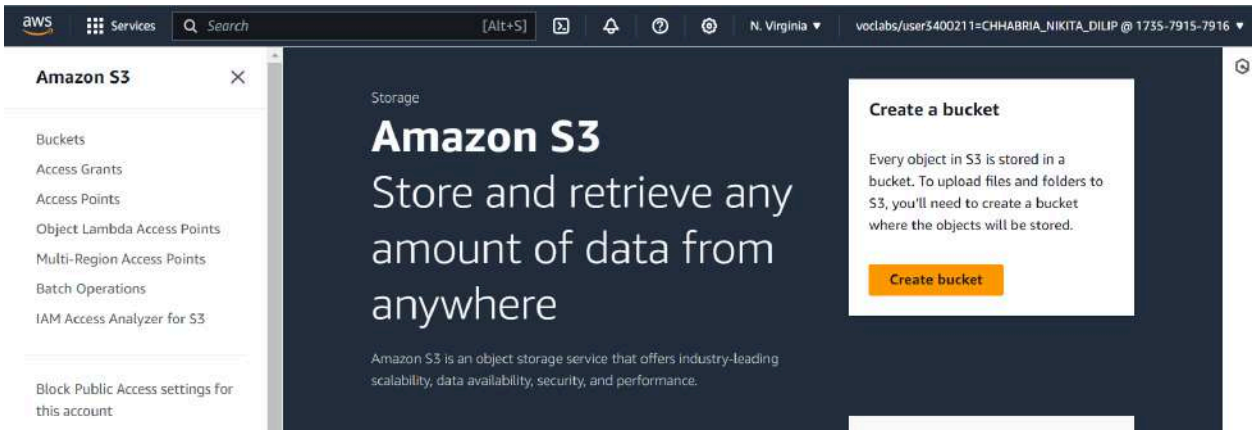


Go to your browser and type <localhost/filename/>

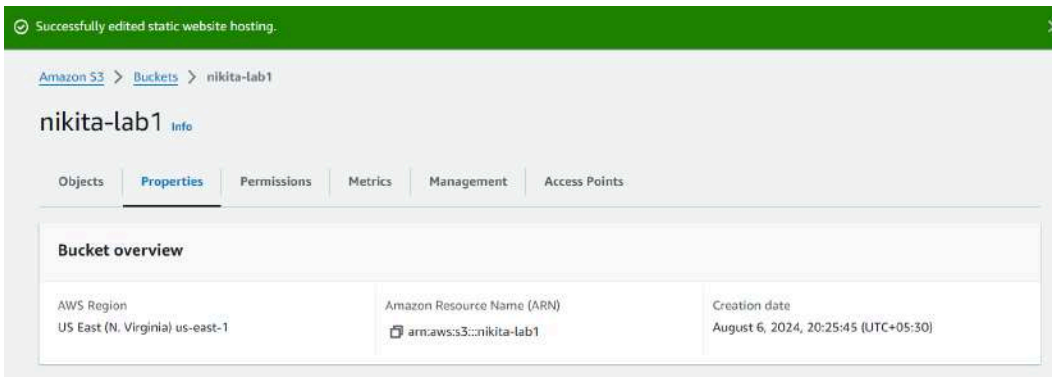


Hosting website with Amazon S3

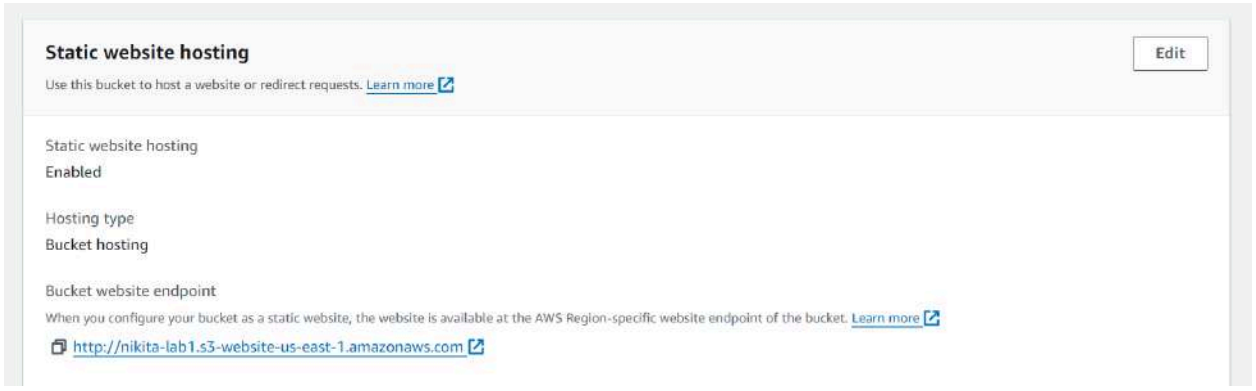
Open your learner lab's and Go to Amazon S3 in services



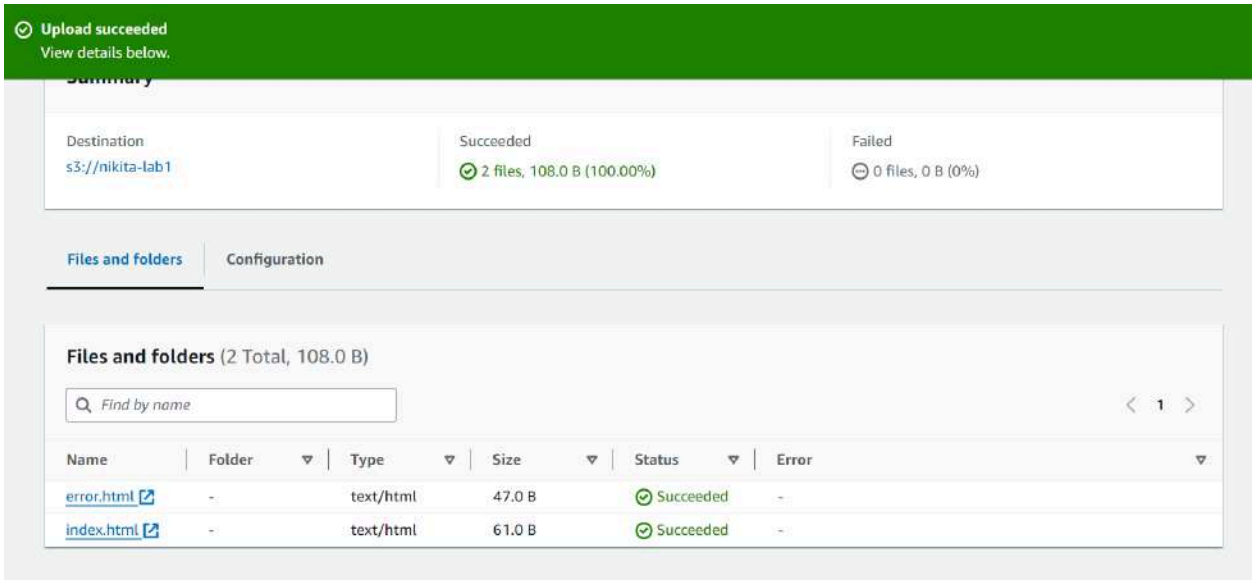
Click on create bucket and make a bucket with any name



In properties go to static website hosting and enable it



Go to Uploads option and upload the code files in them



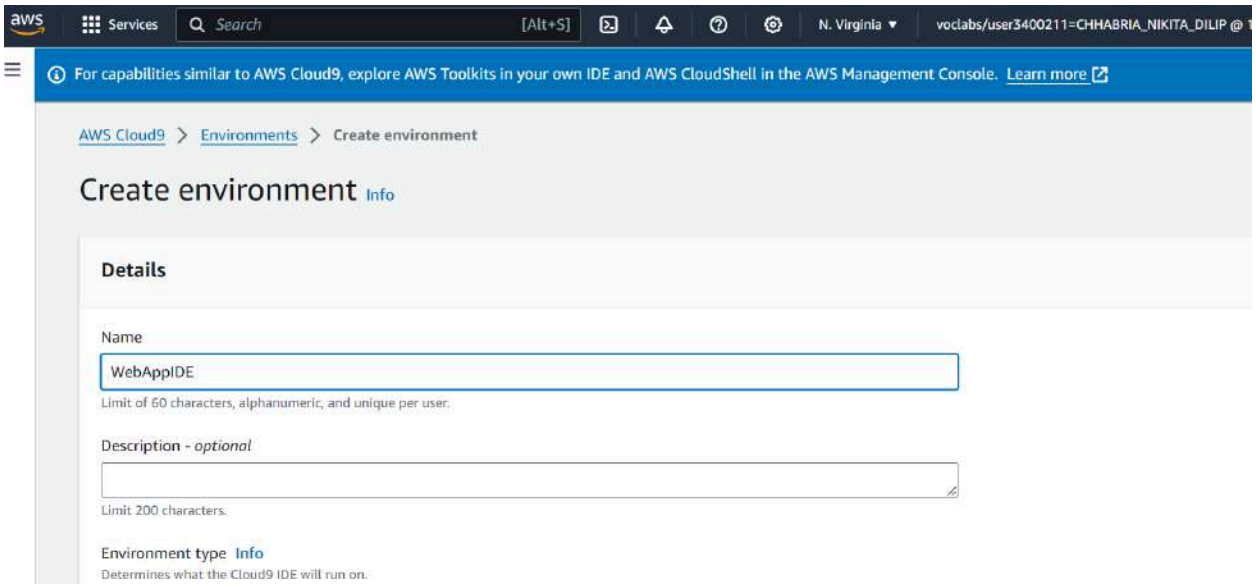
To host the website we need some permissions which can be changed using the following code

lab 1

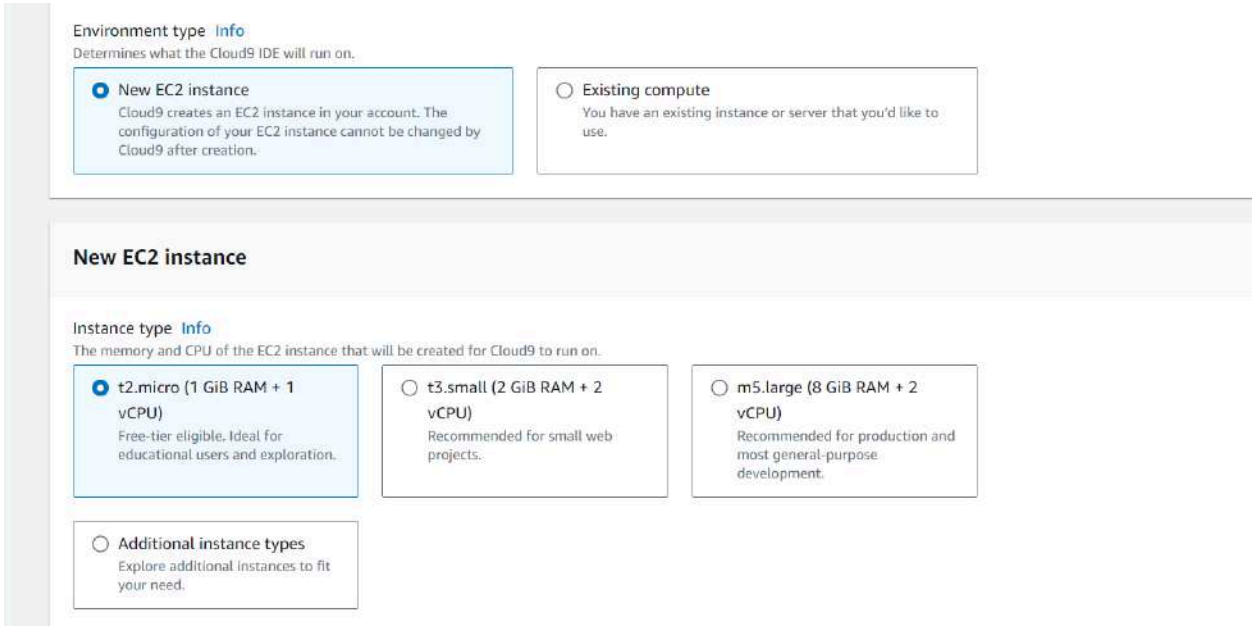

Not secure | www.lab1a.com.s3-website-us-east-1.amazonaws.com
Error Message

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

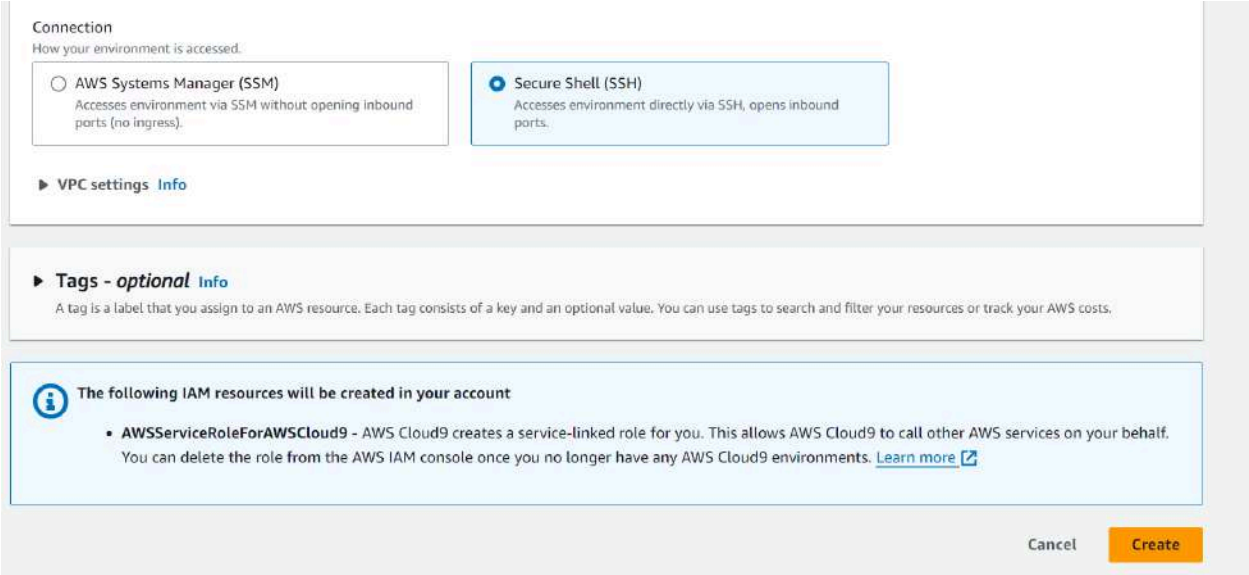
Login in to your Aws canva account and search for AWS Cloud9 in services



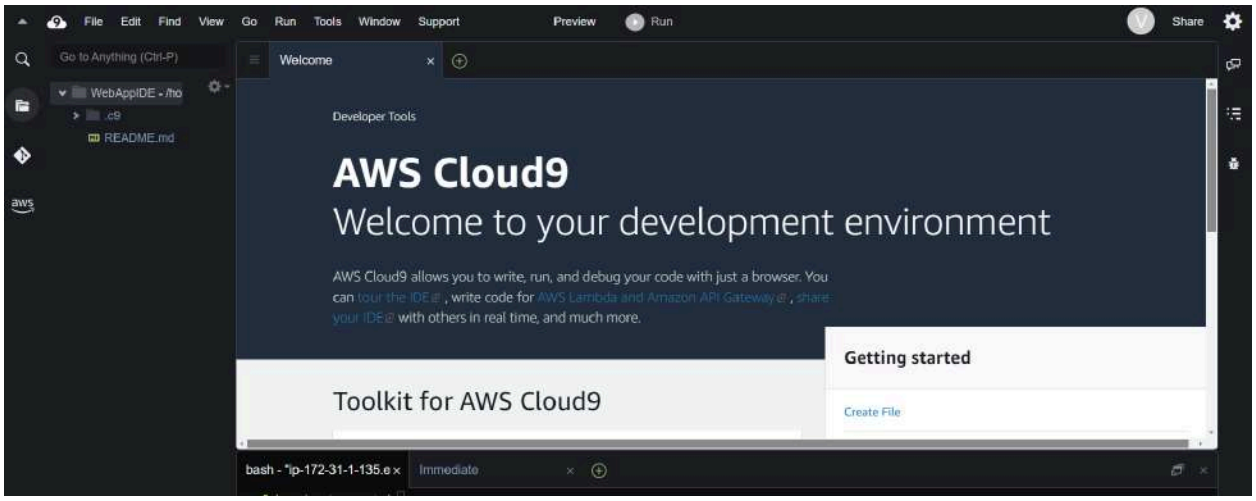
Select Environment type to be EC2



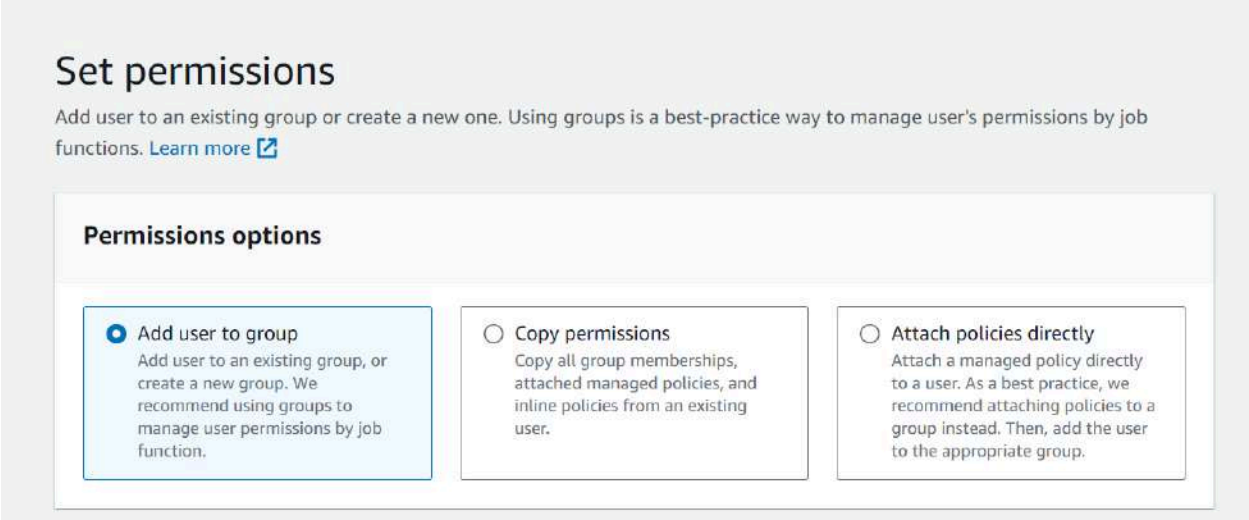
A few other options depending on your use case and then **Create environment**



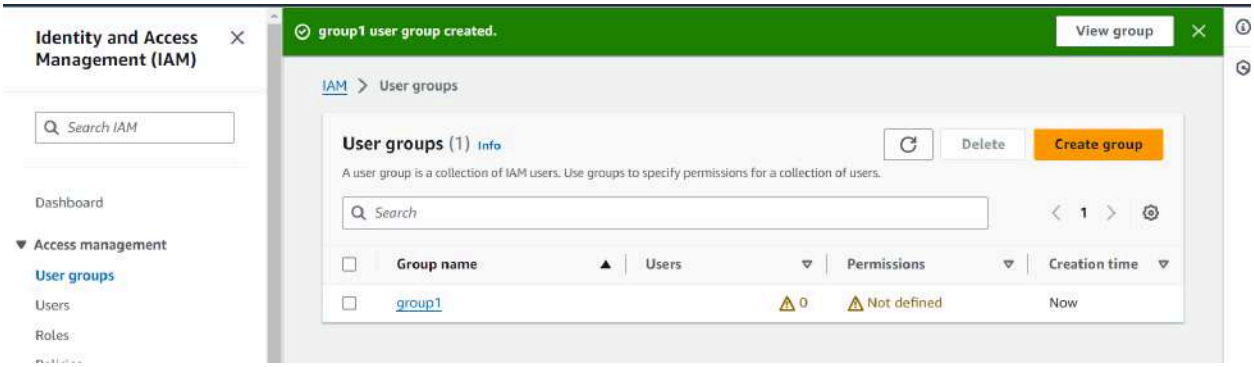
The environment is ready

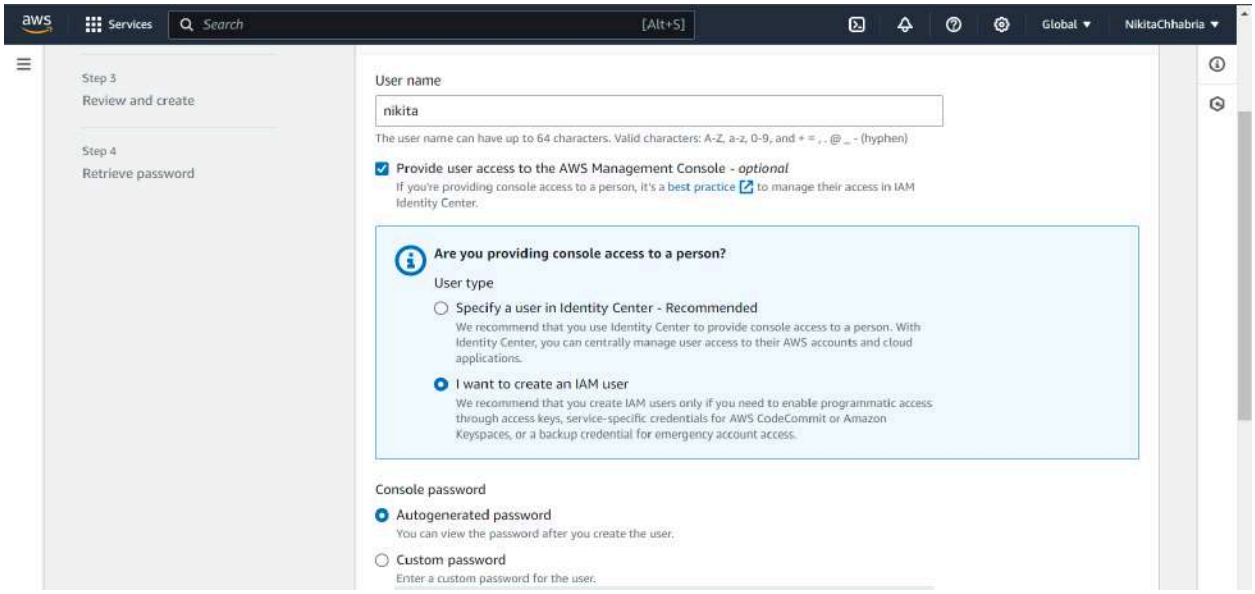


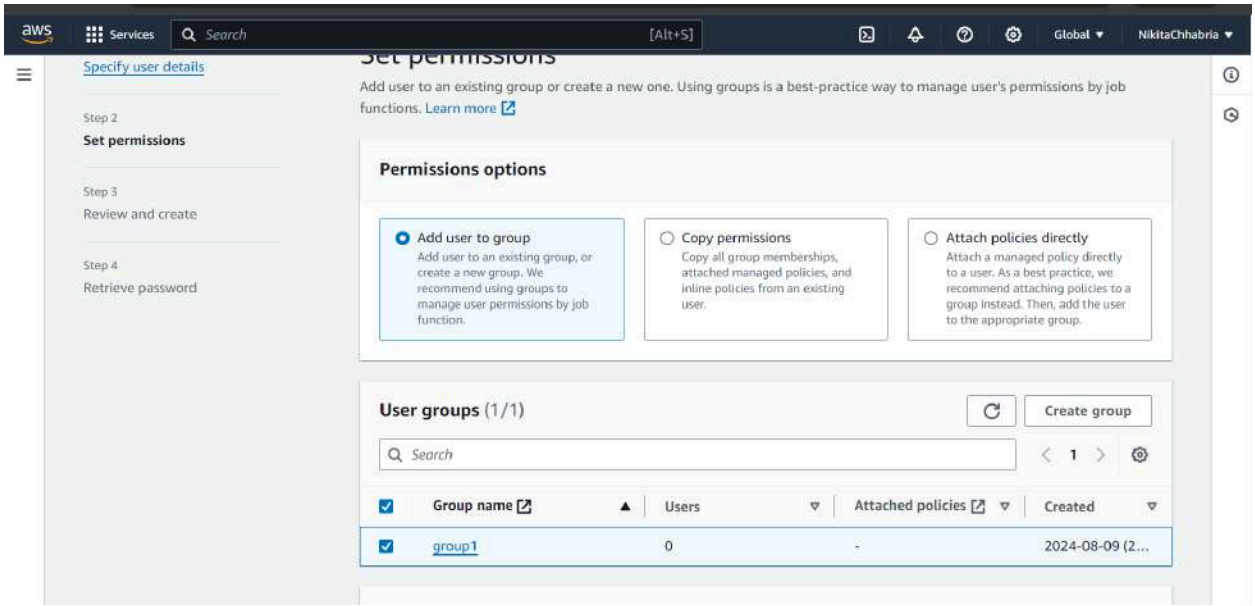
Now, to enable collaboration in cloud environment, we need to add a user group. Only IAM users can be part of this group.



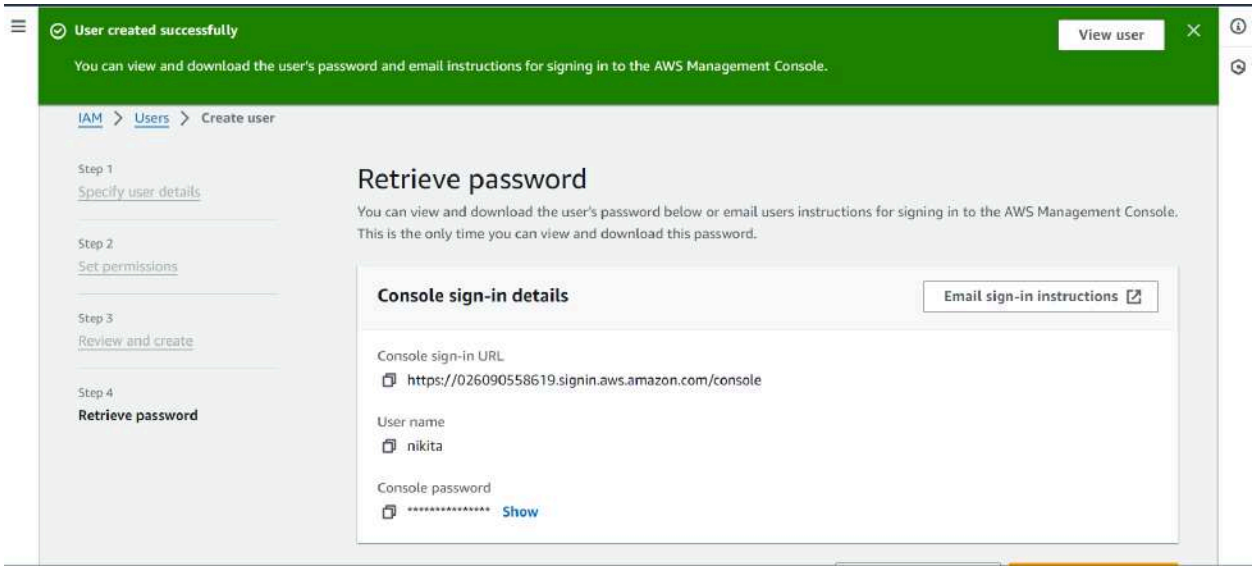
Group created without any hassle



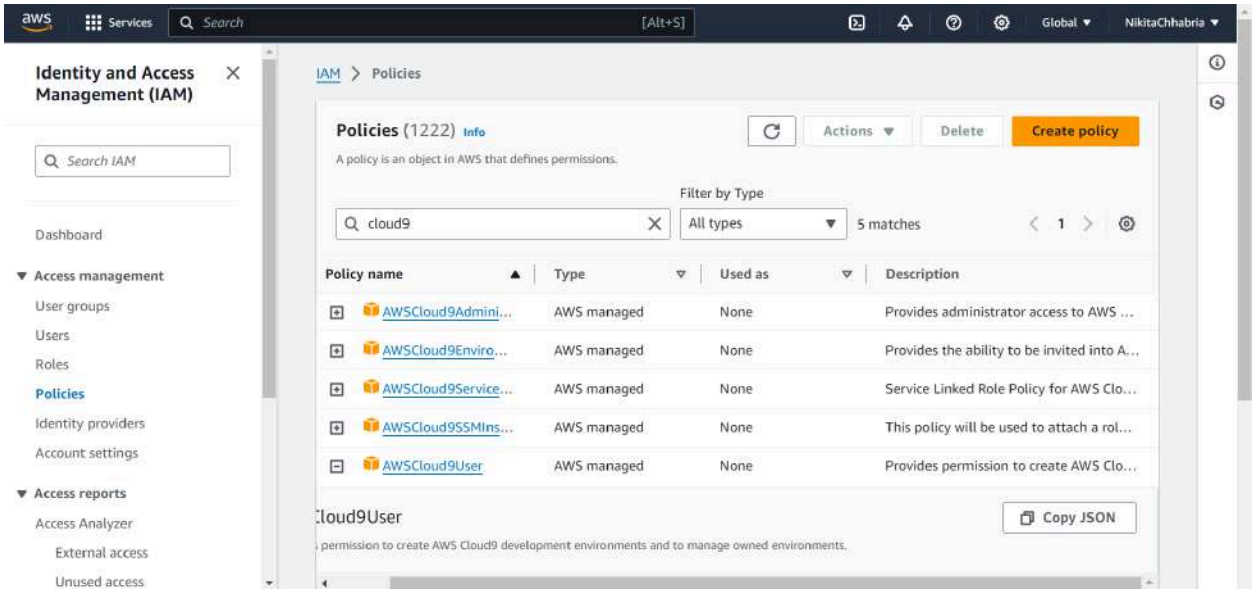




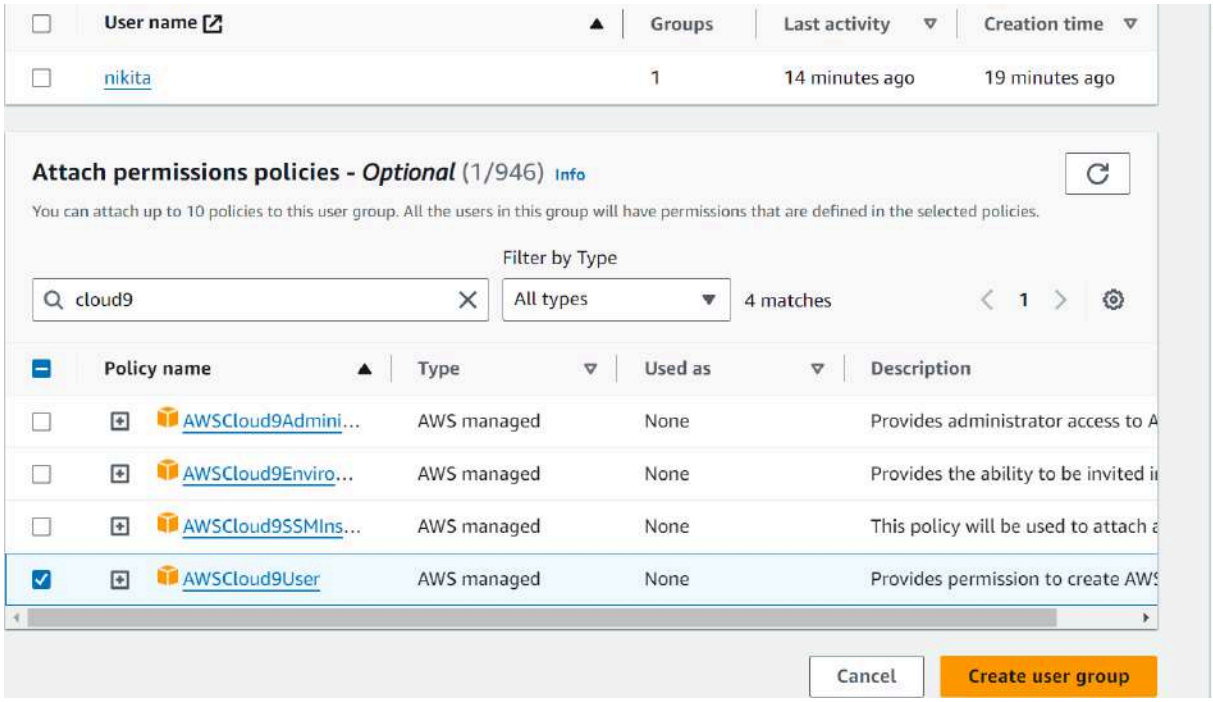
Now for creating an IAM user, you need to create an account as in the academy you don't the access to create one.



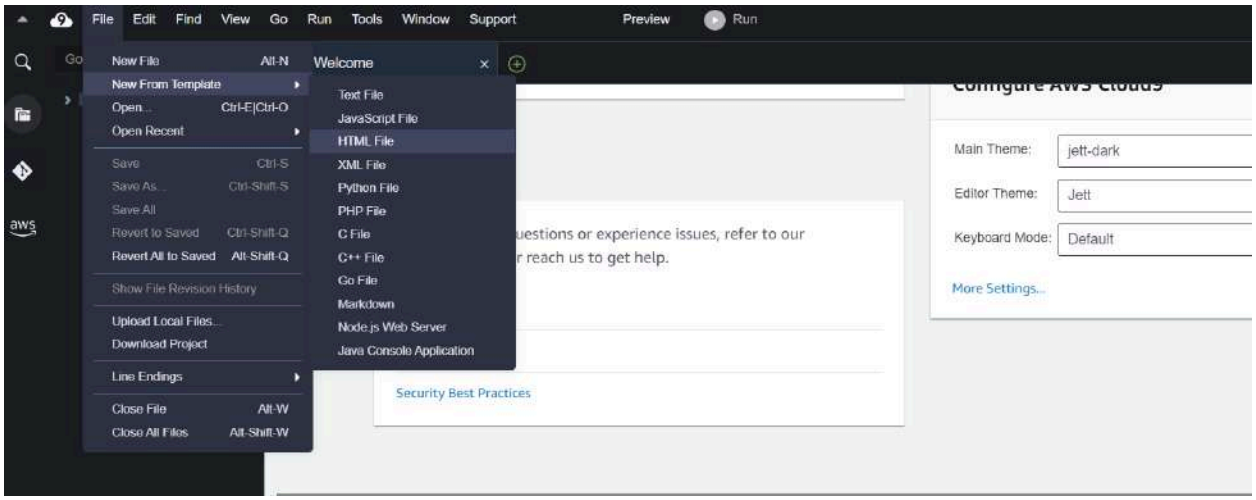
Add policies as per your requirement> I've added AWSCloud9 services

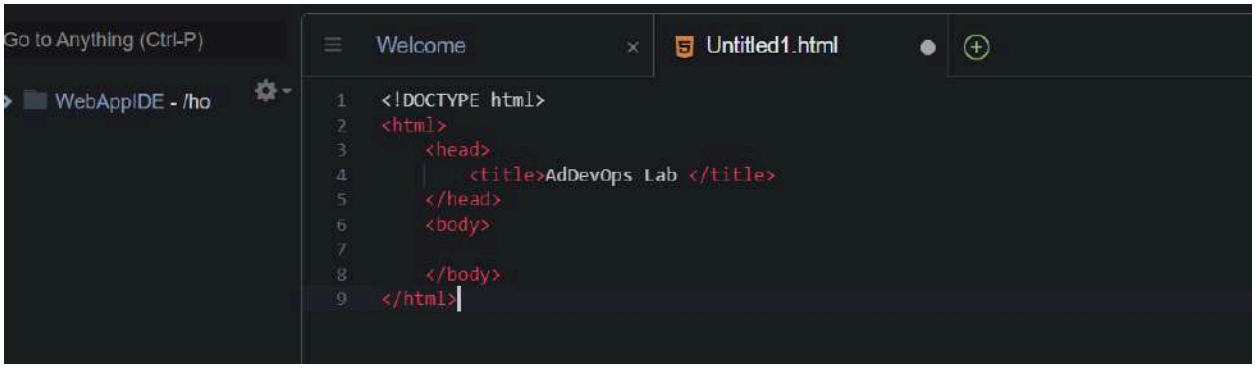


Click on create and done

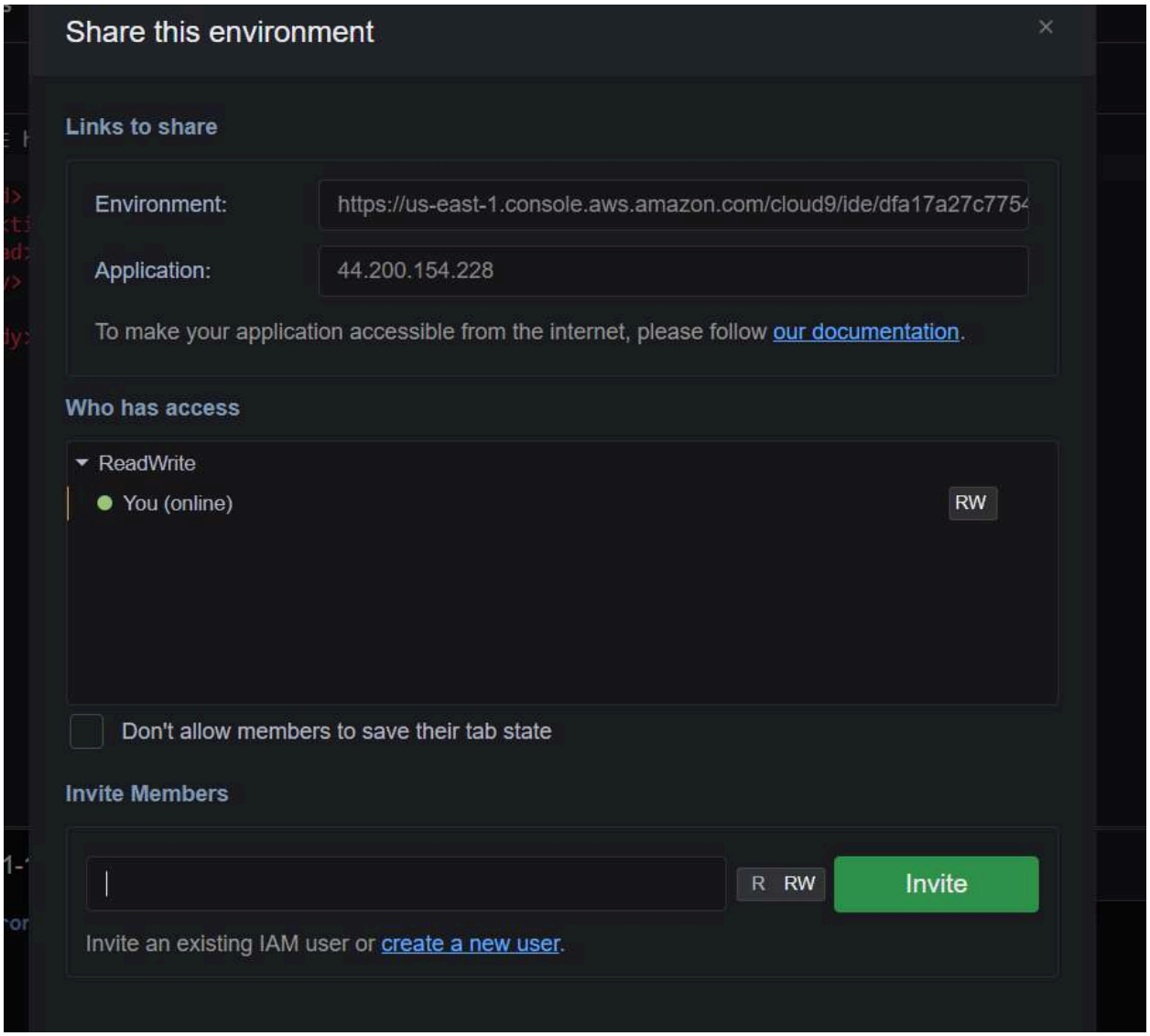


Now, go back to the academy account and deploy your code





Click on share this environment and update the <who can access> section

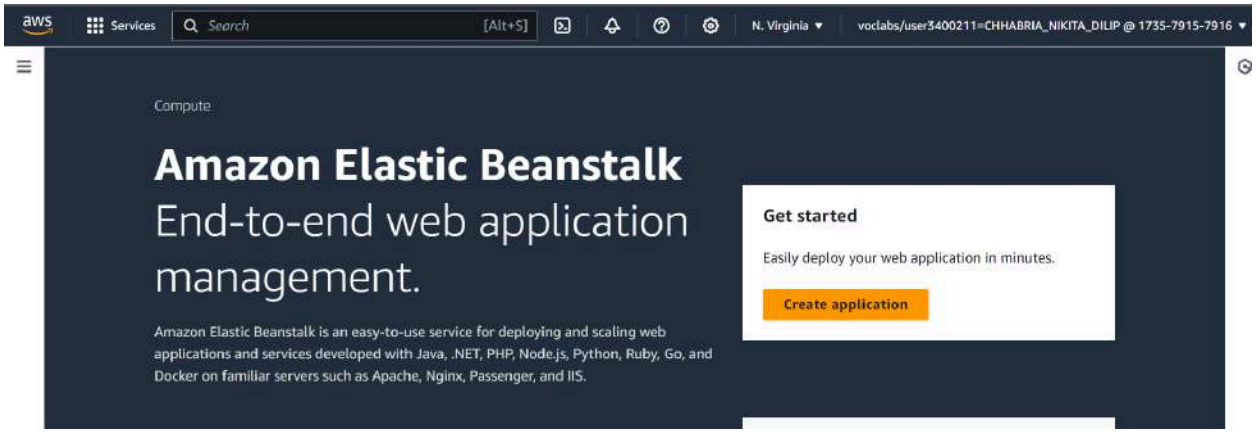


And done!

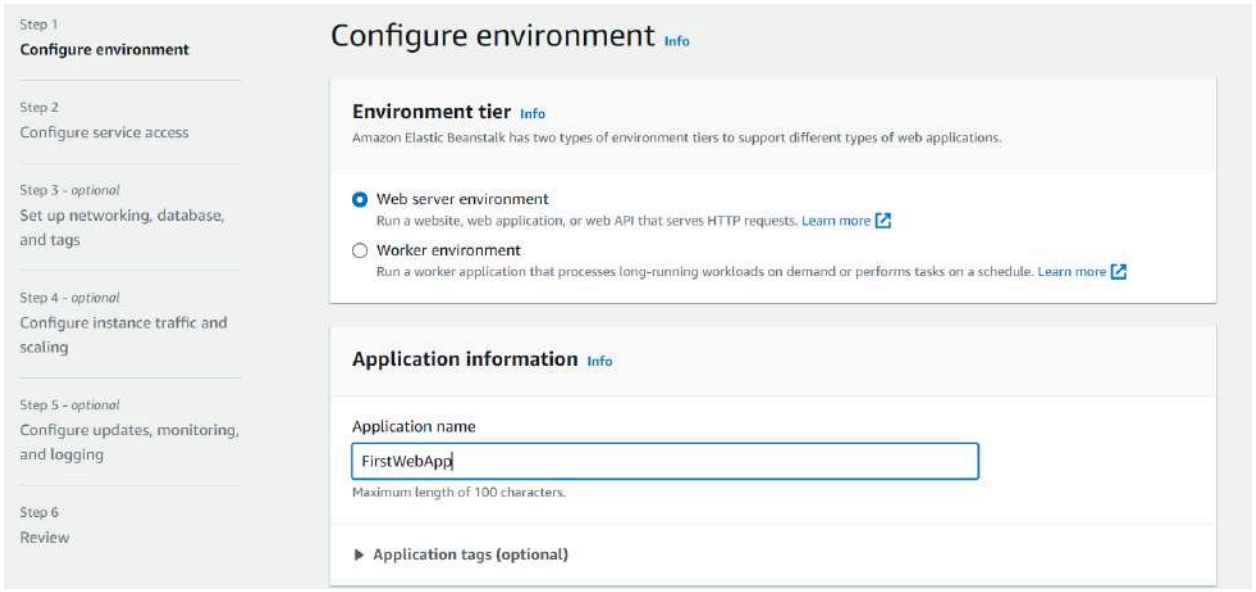
Nikita Chhabria **D15C** **07**

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

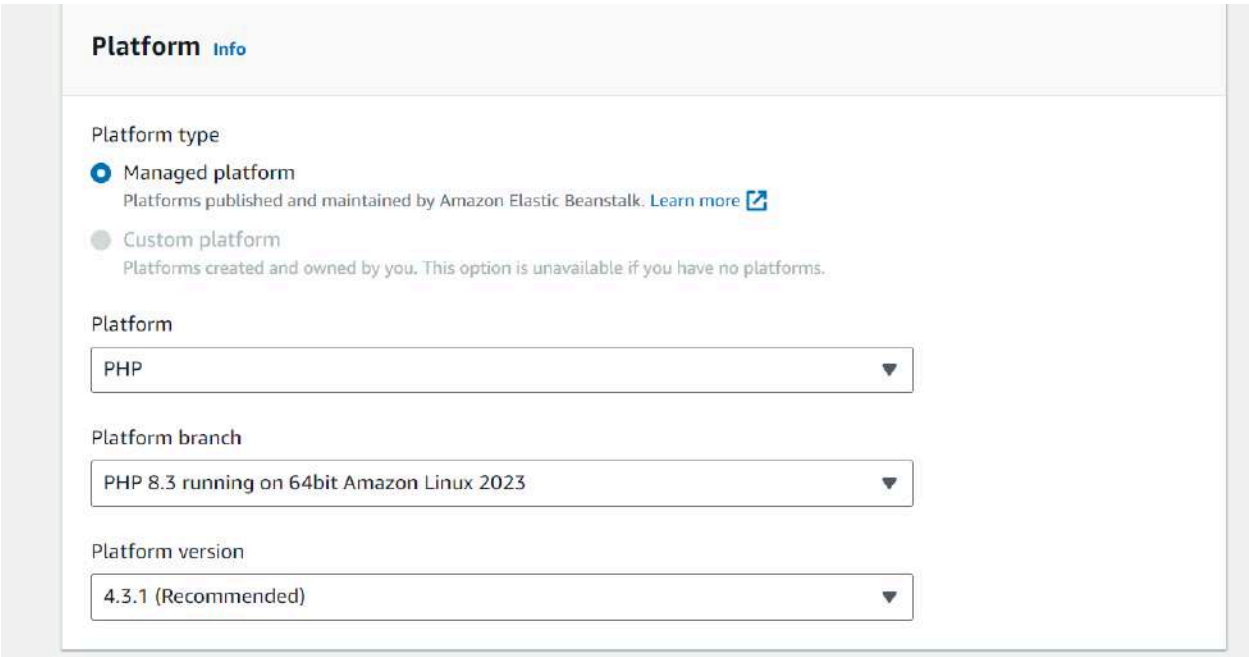
Go to your amazon academy account and search for <Elastic Beanstalk>



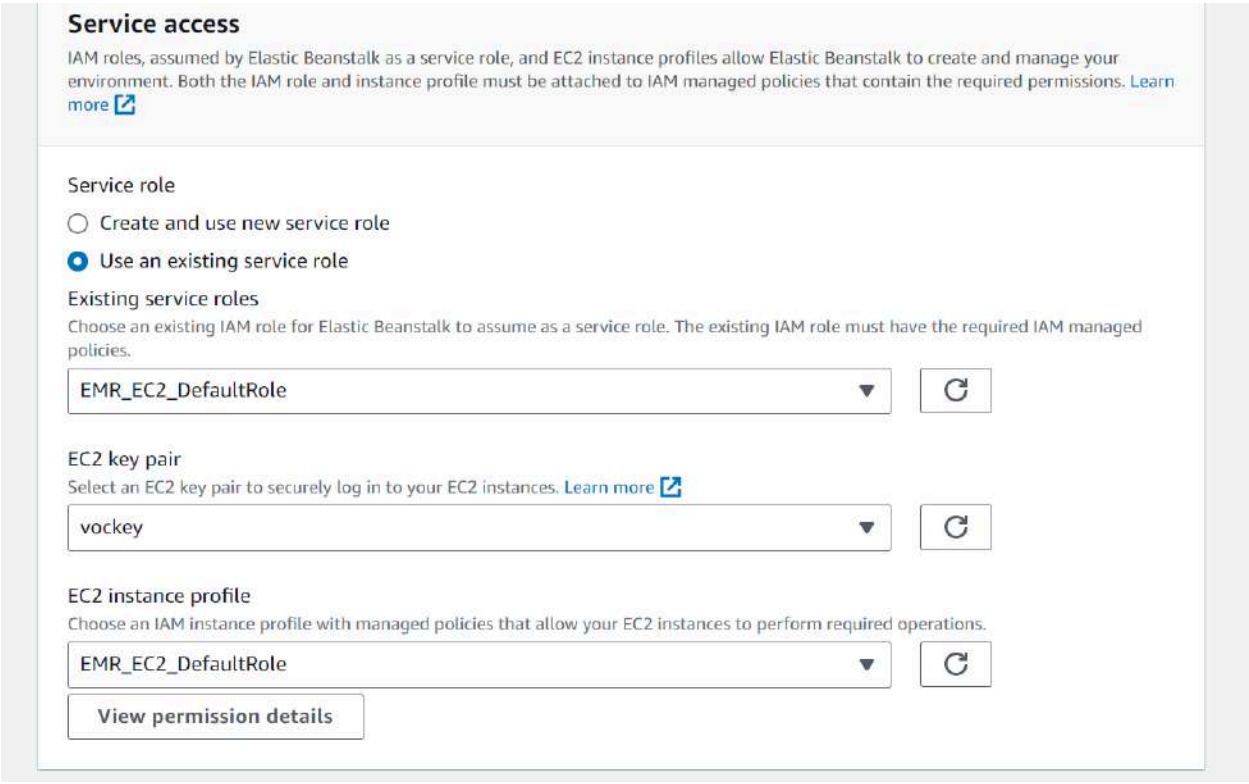
Click on create application and configure the environment



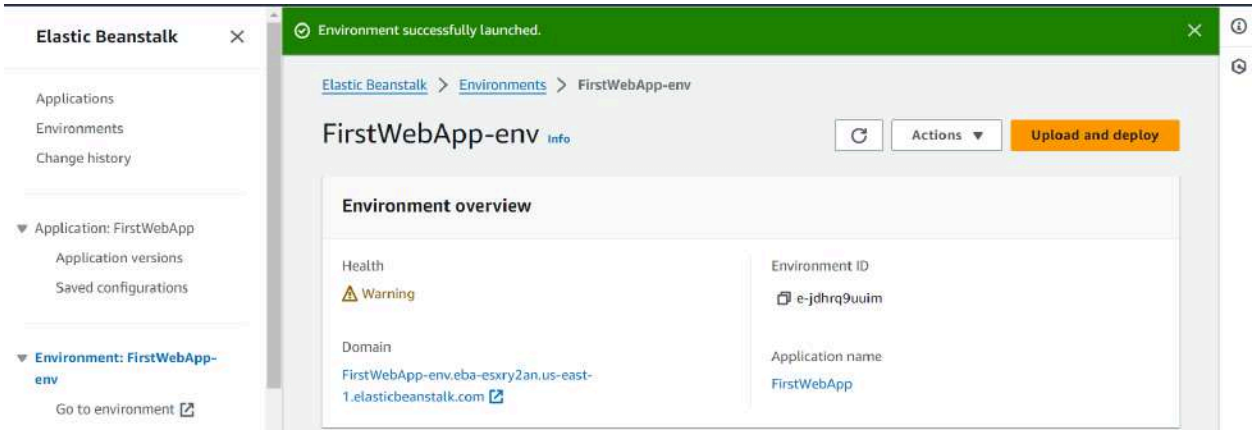
Now, manage the platform type as shown below



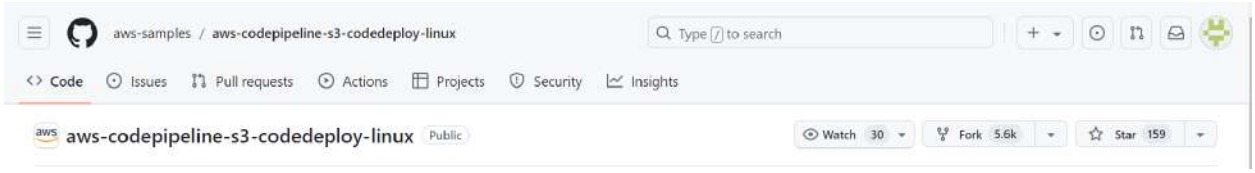
A few more customizations



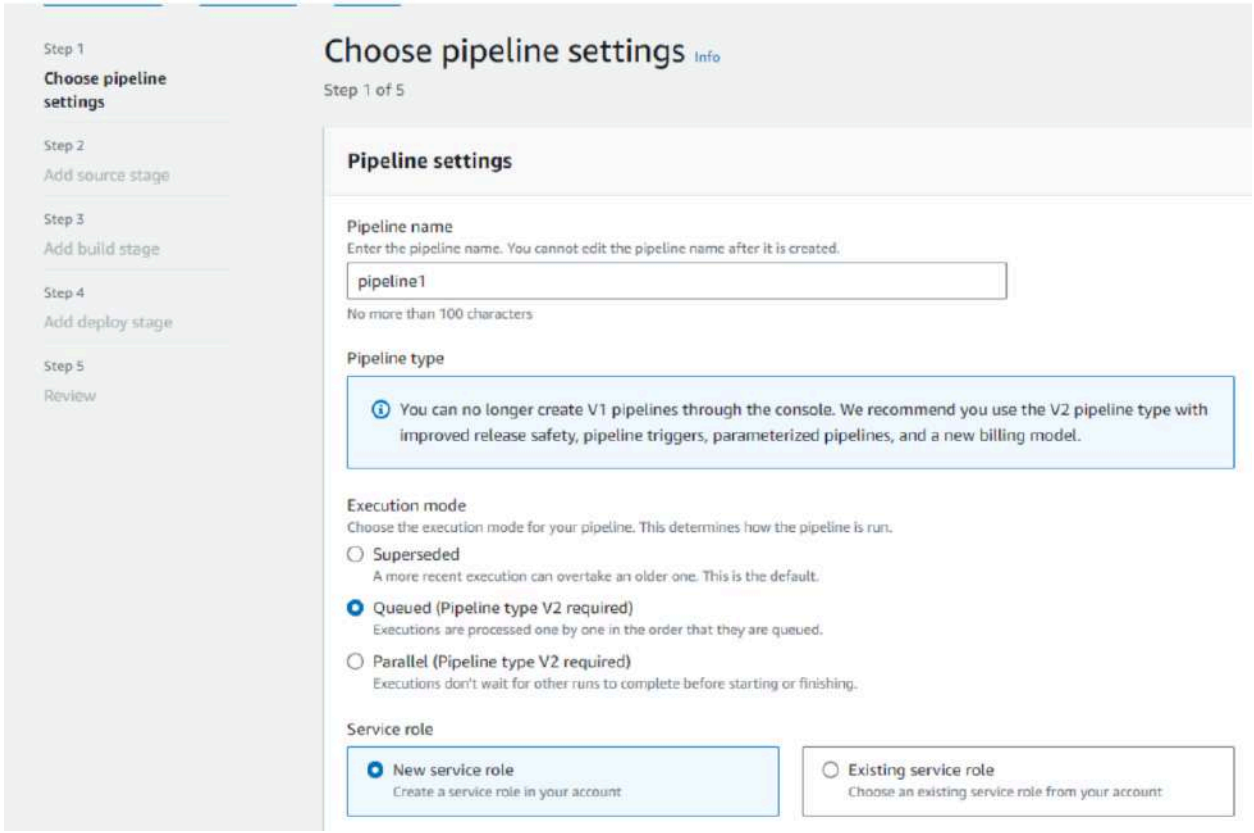
And environment created successfully!



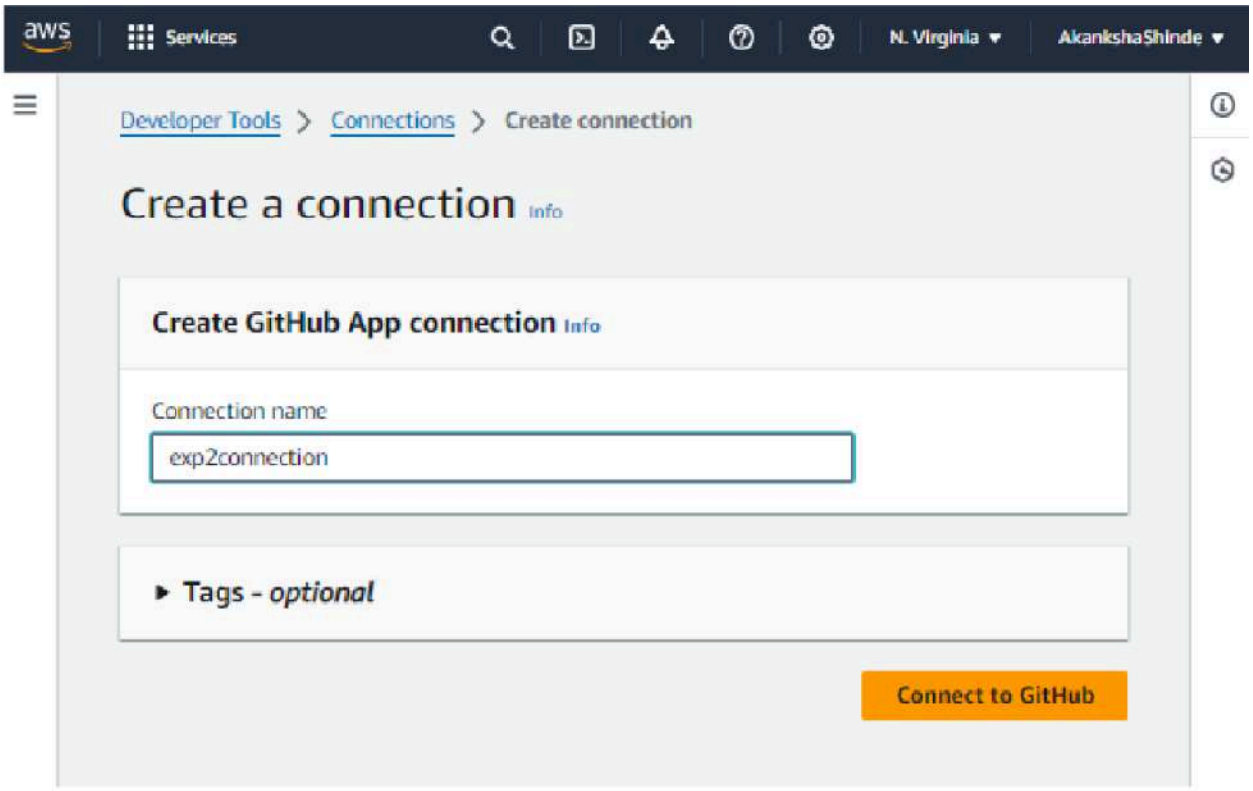
Now, go to github> https://github.com/aws-samples/aws-codepipeline-s3-codedeploy-linux



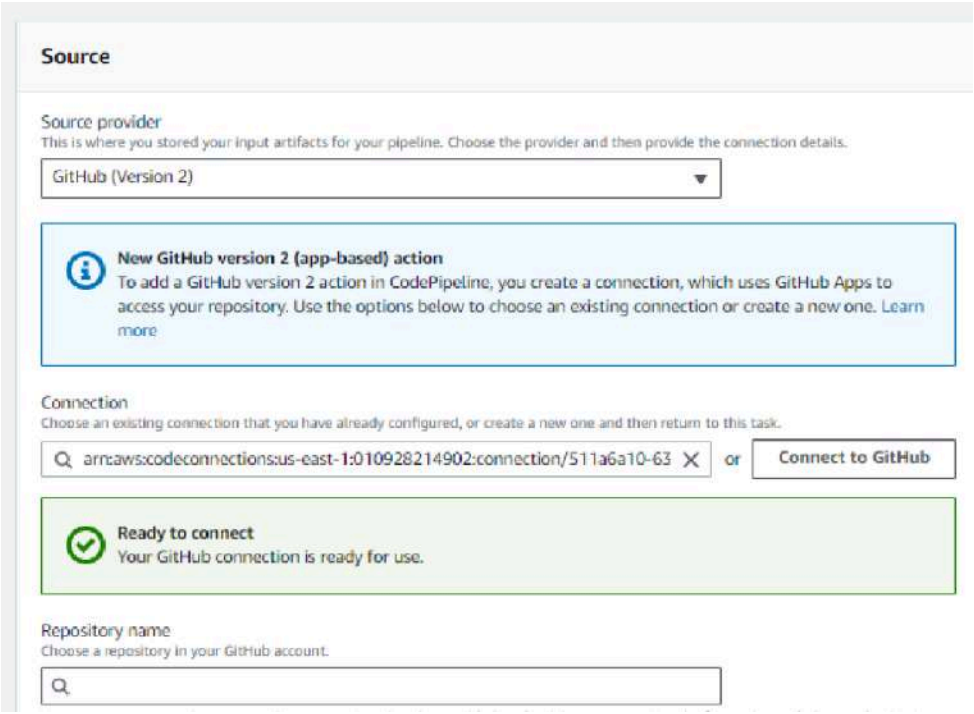
Fork this Repository and come back to the account and go to code pipeline

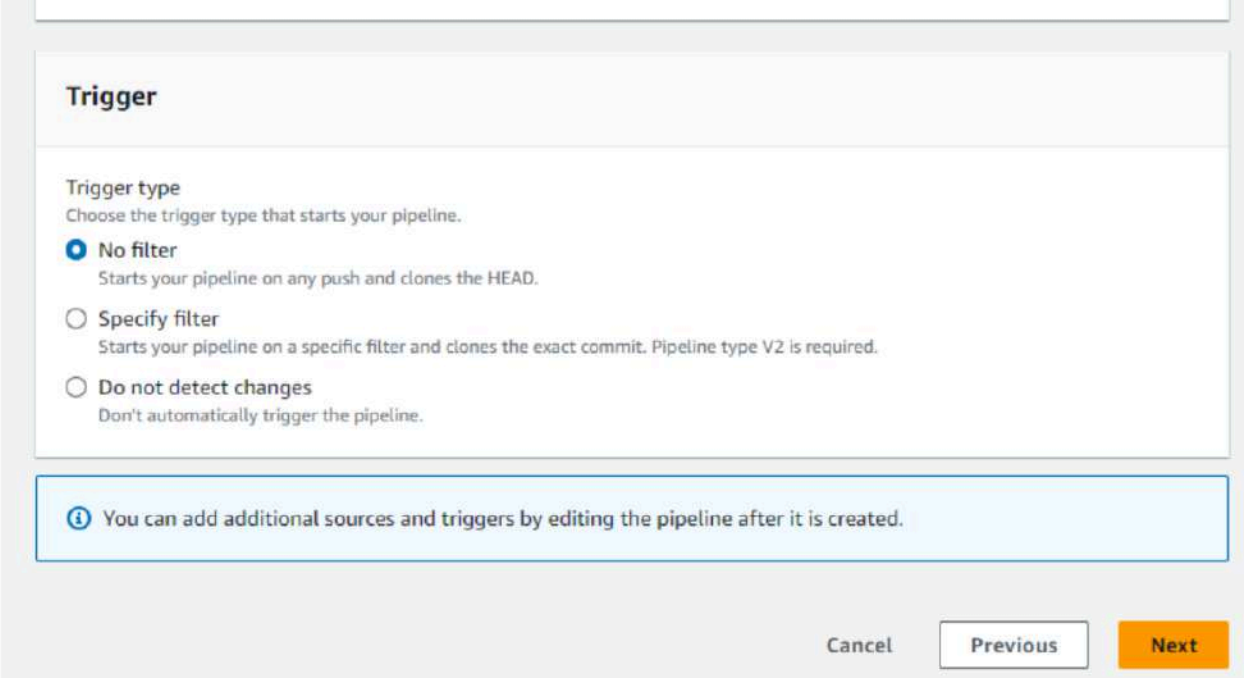


Follow through all the steps given

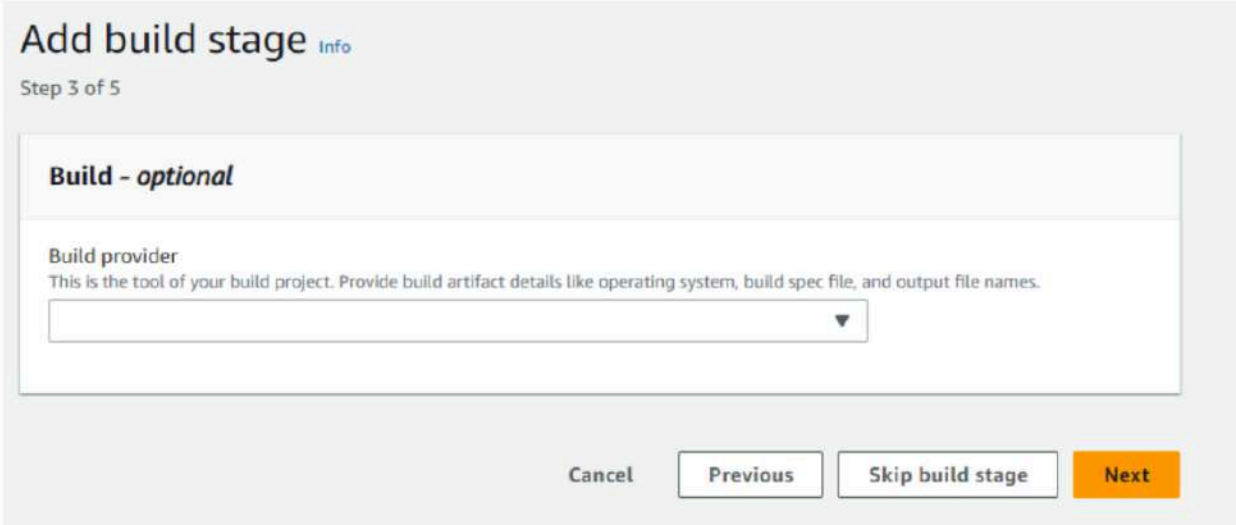


2. In the source stage, choose GitHub v2 as the provider, then connect your GitHub account to AWS by creating a connection. You'd need your GitHub credentials and then you'd need to authorize and install AWS on the forked GitHub Repository.

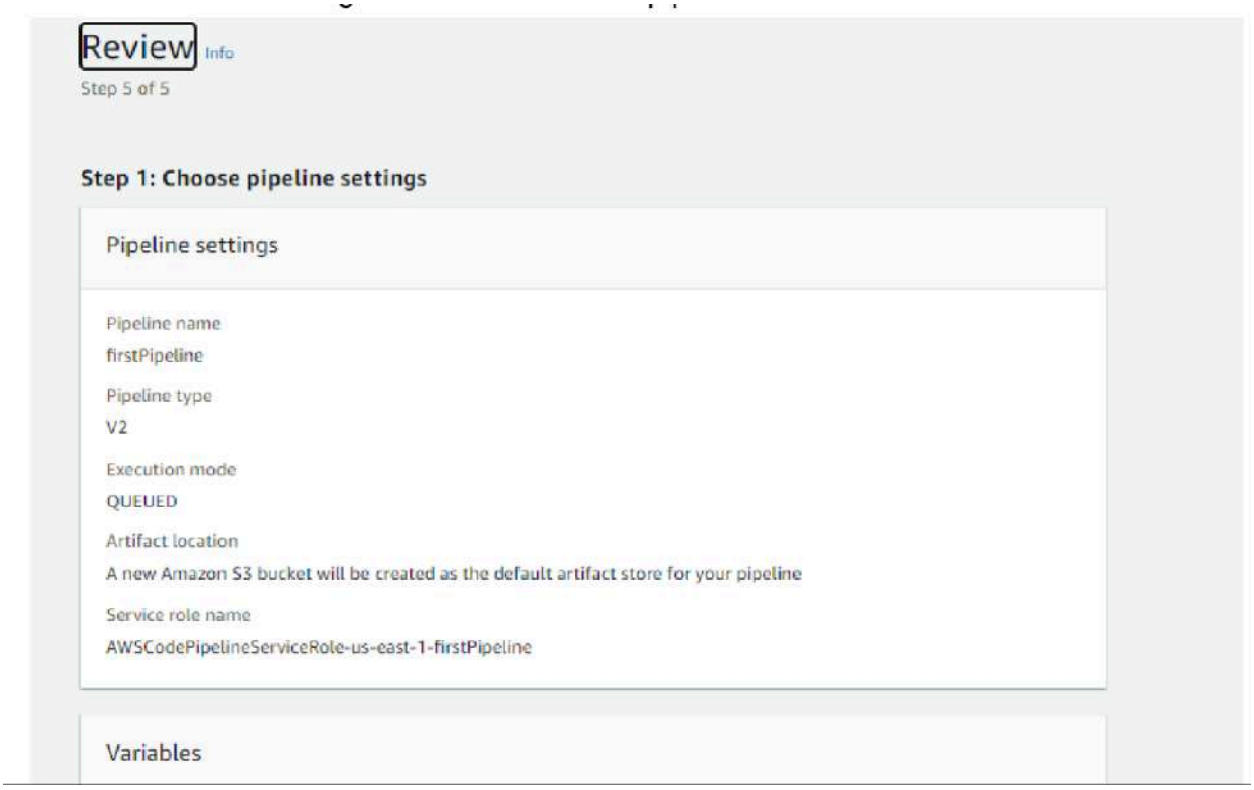

Source
Source provider
This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.
GitHub (Version 2)
New GitHub version 2 (app-based) action
To add a GitHub version 2 action in CodePipeline, you create a connection, which uses GitHub Apps to access your repository. Use the options below to choose an existing connection or create a new one. Learn more
Connection
Choose an existing connection that you have already configured, or create a new one and then return to this task.
arn:aws:codeconnections:us-east-1:010928214902:connection/511a6a10-63
or
Connect to GitHub
Ready to connect
Your GitHub connection is ready for use.
Repository name
Choose a repository in your GitHub account.


Trigger
Trigger type
Choose the trigger type that starts your pipeline.
No filter
Starts your pipeline on any push and clones the HEAD.
Specify filter
Starts your pipeline on a specific filter and clones the exact commit. Pipeline type V2 is required.
Do not detect changes
Don't automatically trigger the pipeline.
You can add additional sources and triggers by editing the pipeline after it is created.
Cancel
Previous
Next

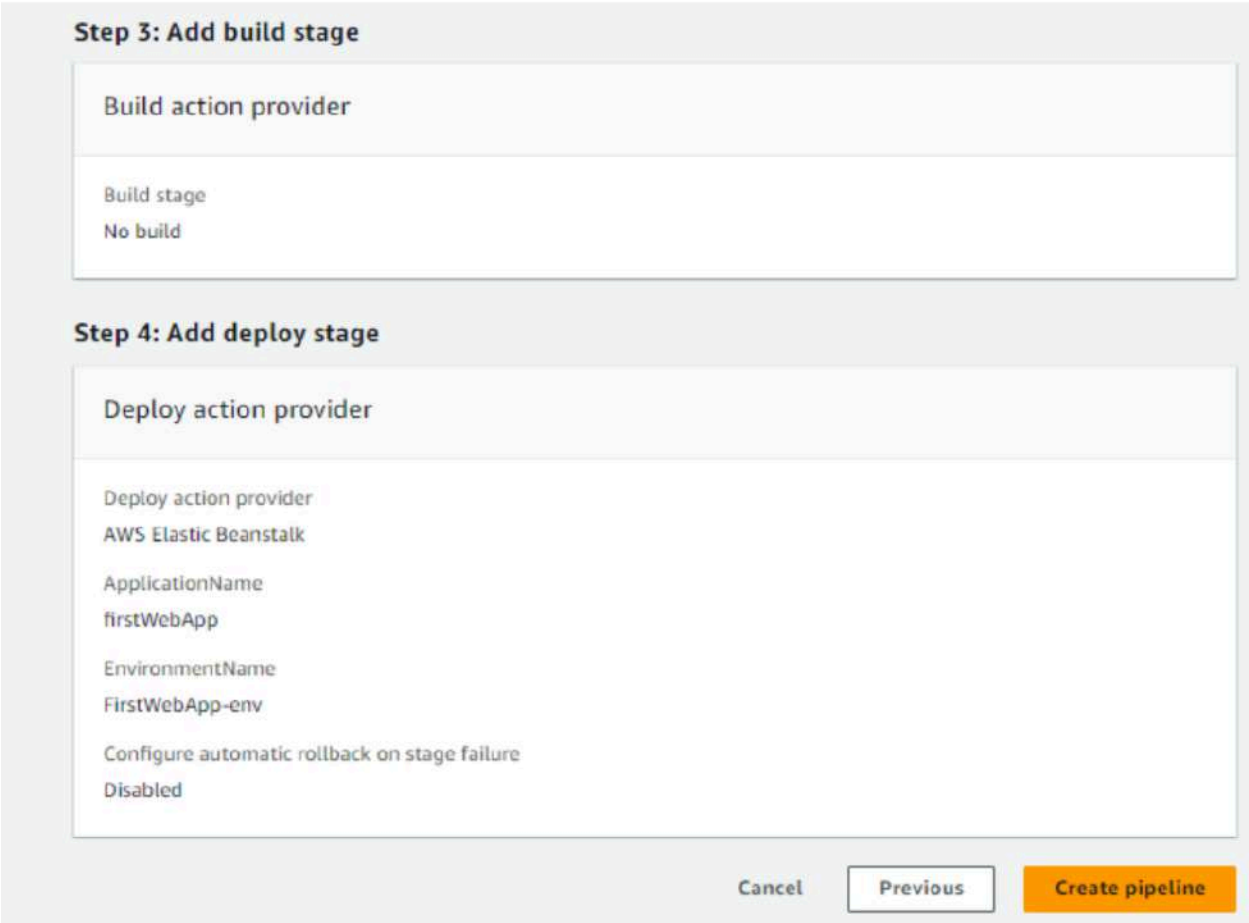
3. Then, simply choose this forked repository and the branch which you will be able to find in the search box. After that, click Continue and skip the build stage. Proceed to the Deployment stage.



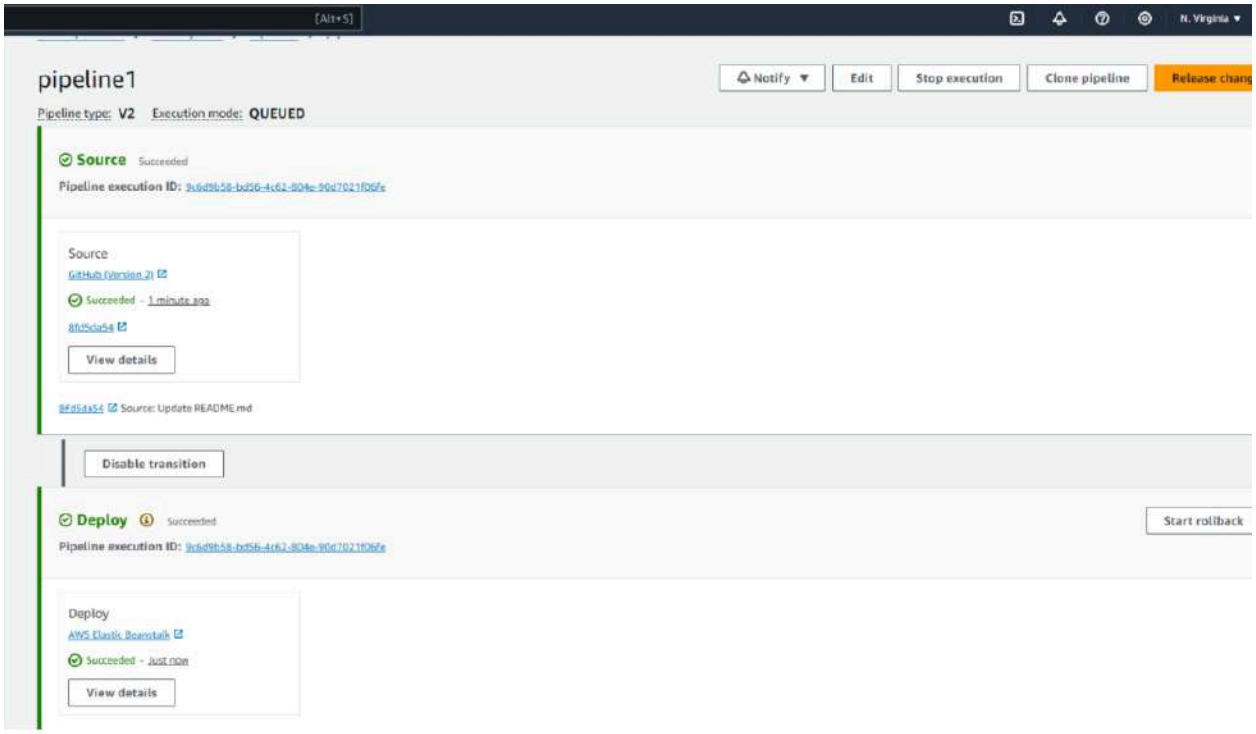
Review all the settings



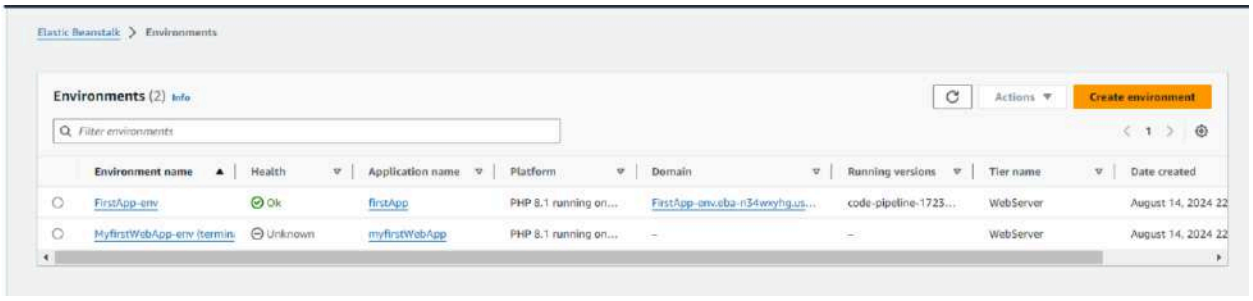
The pipeline is created successfully

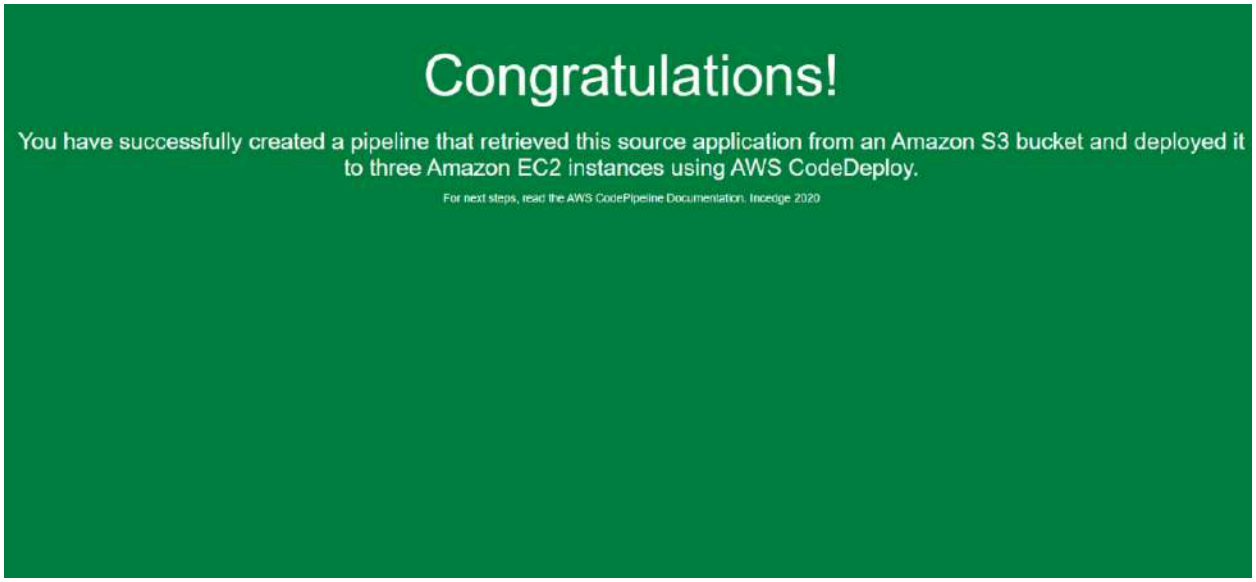


Now, you will be able to see the code deployed



Now we can see the URL provided for our application, we can simply click on it to lead us to our app




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. Incedge 2020

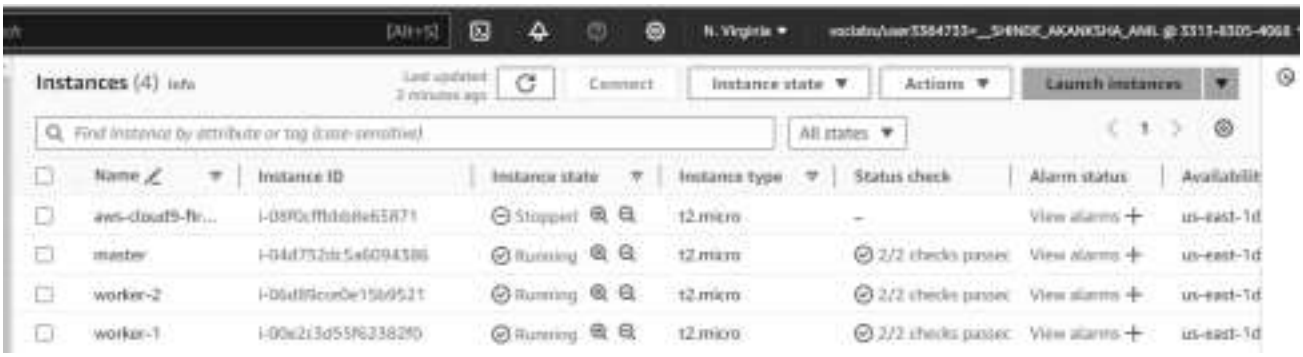
Experiment: 3

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 and do it for all the three machines.

```
Akanksha Shinde@AkankshaShinde MINGW64 ~/downloads (master)
$ ssh -i "server.pem" ec2-user@ec2-54-174-206-93.compute-1.amazonaws.com
The authenticity of host 'ec2-54-174-206-93.compute-1.amazonaws.com (54.174.206
93)' can't be established.
ED25519 key fingerprint is SHA256:T+tsGyI15gAvUvjeAZ7GjDIWXHOaI4EPF5g5oICrkoQ.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-54-174-206-93.compute-1.amazonaws.com' (ED25519)
 to the list of known hosts.
       #_
       ####_          Amazon Linux 2023
      \_#####\
      \###|
        \#/ ___      https://aws.amazon.com/linux/amazon-linux-2023
```

3. From now on, until mentioned, perform these steps on all 3 machines. Install Docker for all the 3 machines

```
[root@ip-172-31-90-172 ec2-user]# yum install docker -y
Last metadata expiration check: 0:21:16 ago on Fri Aug 30 04:01:12 2024.
Dependencies resolved.
```

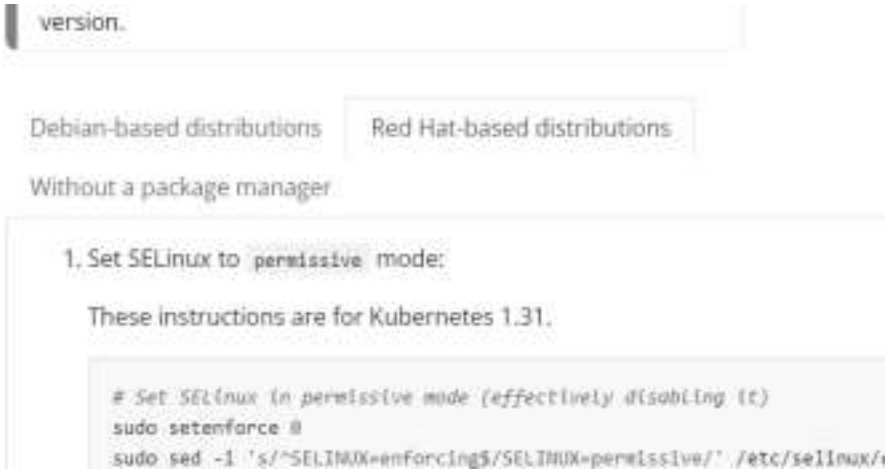
Package	Architecture	Version
Installing:		
docker	x86_64	25.0.6-1.amzn2023.0.1
Installing dependencies:		
containerd	x86_64	1.7.20-1.amzn2023.0.1
iptables-libs	x86_64	1.8.8-3.amzn2023.0.2
iptables-nft	x86_64	1.8.8-3.amzn2023.0.2
libcgroup	x86_64	3.0-1.amzn2023.0.1
libnetfilter_conntrack	x86_64	1.0.8-2.amzn2023.0.2
libnfnetlink	x86_64	1.0.1-19.amzn2023.0.2
libnftnl	x86_64	1.2.2-2.amzn2023.0.2
pigz	x86_64	2.5-1.amzn2023.0.3

Start the docker by running the command systemctl start docker in the terminal of all the ec2 instance.

```
Complete!
[root@ip-172-31-82-133 ec2-user]# systemctl start docker
[root@ip-172-31-82-133 ec2-user]#
```

4. Install the kubernetes on all 3 machines by searching for kubeadm and click on install kubernetes.

Select the red hat based distribution. This process will automatically disable SELinux before configuring kubelet so no need to run it separately in terminal**.**



Copy the below script, to install kubernetes we need a kubernetes repo so this script helps us in getting that and paste it in the terminal.

```
Installed:
  containerd-1.7.20-1.amzn2023.0.1.x86_64       docker-25.0.6-1.amzn2023.0.1.x86_64      iptables-libs-1
  iptables-nft-1.8.8-3.amzn2023.0.2.x86_64      libcgroup-3.0-1.amzn2023.0.1.x86_64      libnetfilter_cc
  libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64     libnftnl-1.2.2-2.amzn2023.0.2.x86_64     pigz-2.5-1.amzn
  runc-1.1.11-1.amzn2023.0.1.x86_64

Complete!
[root@ip-172-31-90-172 ec2-user]# systemctl start docker
[root@ip-172-31-90-172 ec2-user]# sudo su
[root@ip-172-31-90-172 ec2-user]# yum repolist
repo id                                                    repo name
amazonlinux                                                Amazon Linux 2023 repository
```

Run the command yum repolist to check whether the kubernetes repo has installed or not if successful installed then you can see a repo named as kubernetes

```
[root@ip-172-31-90-172 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-172-31-90-172 ec2-user]#
```

Do the above steps for all the instances i.e for worker-1 and worker-2.

5. Perform this ONLY on the Master machine. Initialize the Kubecluster

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

```
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!

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

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

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 172.31.12.28:6443 --token 4bqwb8.lua2ud01lr02uu55 \
    --discovery-token-ca-cert-hash sha256:b4edc7948be9bca50767f623b58e0612feedc144a7364f95be8dbd8c4614a169
```

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

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

```
[ec2-user@ip-172.31.12.28 docker]$ mkdir -p $HOME/.kube
sudo cp -i  /etc/kubernetes/admin.conf  $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Then, add a common networking plugin called flammel file as mentioned in the code.
kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/ k ube-flannel.yml

```
[ec2-user@ip-172.31.12.28 docker]$ kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
```

Check the created pod using this command

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

6. Perform this ONLY on the worker machines

Run the following command

sudo yum install iproute-tc -y

sudo systemctl enable kubelet
sudo systemctl restart kubelet

Check the status of the pods using the following command
This command will show the status of all the pods.
kubectl get pods -n kube-system
Following command will show the status of the pod named daemonset.
kubectl get daemonset -n kube-system

```
[ec2-user@ip-172.31.12.28 docker]$ kubectl get pods -n kube-system
NAME                                                READY   STATUS             RESTARTS   AGE
coredns-55cb5b8774-fx12f                            1/1     Running            0          100s
coredns-55cb5b8774-xn14v                            1/1     Running            0          100s
etcd-ip-172.31.12.28.ec2.internal                   1/1     Running            0          75s
kube-apiserver-ip-172.31.12.28.ec2.internal         1/1     Running            1          2m
kube-controller-manager-ip-172.31.12.28.ec2.internal 0/1 CrashLoopBackOff   1          70s
kube-proxy-4dv8m                                    1/1     Running            2          100s
kube-scheduler-ip-172.31.12.28.ec2.internal         1/1     Running            1          76s

[ec2-user@ip-172.31.12.28 docker]$ kubectl get daemonset -n kube-system
NAME         DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR            AGE
kube-proxy   1         1         1       1            1           kubernetes.io/os=linux   3m
```

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

Conclusion:

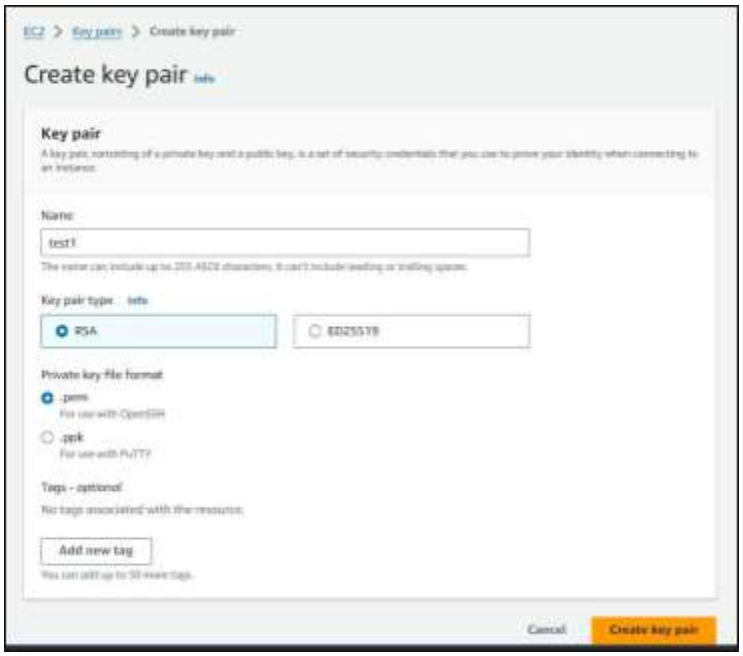
Kubernetes cluster was successfully established using three AWS EC2 instances, which includes one Master and two Worker nodes. The process began with the creation of instances and configuration of settings to begin the communication. Docker was installed on all machines followed by the installation of Kubernetes components and the necessary repositories. The Master node was initialized with the `kubeadm init` command, and a plugin called Flannel was deployed to enable pod communication. Error incurred during initialization can be solved by changing the instance type to t3.medium or t3.large and thus 3 nodes were connected successfully

Experiment 4

Aim:

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

Create a key pair



The .pem file will be downloaded on your machine and will be required in the further steps.

1. Now we will create an EC2 Ubuntu instance. Select the key pair which you just created while creating this instance.

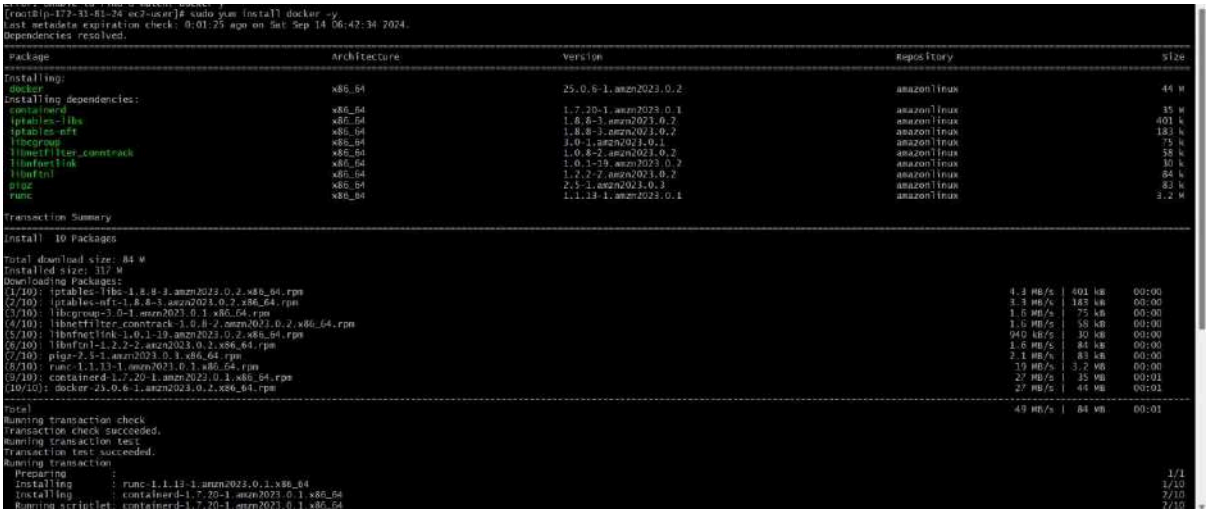
2. Open git bash and go to the directory where pem file is located and use chmod to provide permissions.

```
Dell@DESKTOP-OVNTA1M MINGW64 ~/Downloads (master)
$ chmod 400 test1.pem
```

3. Now use this command on the terminal: ssh -i <keyname>.pem ubuntu@<public_ip_address> and replace

```
Dell@DESKTOP-OVNTA1M MINGW64 ~/Downloads (master)
$ ssh -i "test1.pem" ec2-user@ec2-44-204-14-37.compute-1.amazonaws.com
The authenticity of host 'ec2-44-204-14-37.compute-1.amazonaws.com (44.204.14.37
)' can't be established.
ED25519 key fingerprint is SHA256:CtxhAZnv4MFbUaiO3z96MQzMKK6JxuN/nWlIerDSazI.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-44-204-14-37.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-172-31-81-24 ~]$
```

4. Docker installation:
   We will be installing docker by using “sudo yum install docker -y”



5. Then to configure cgroup in a daemon json file we will run

cd /etc/docker

cat <<EOF | sudo tee /etc/docker/daemon.json

{

"exec-opts": ["native.cgroupdriver=systemd"]

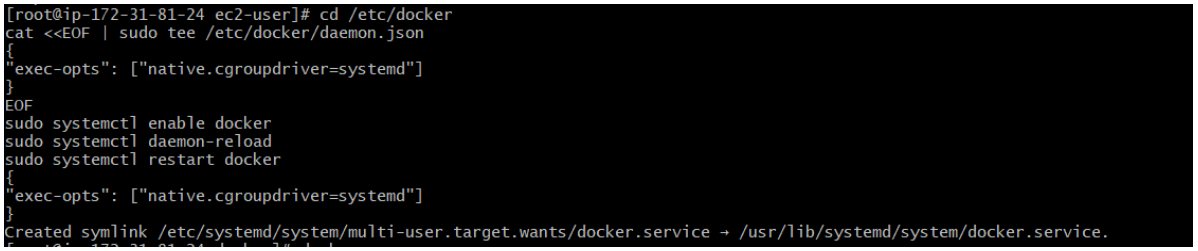
}

EOF

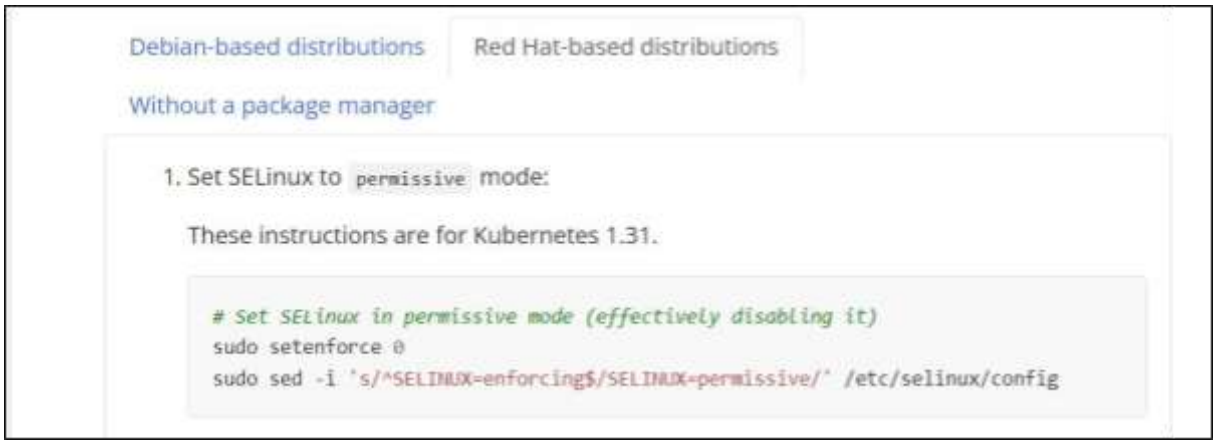
sudo systemctl enable docker

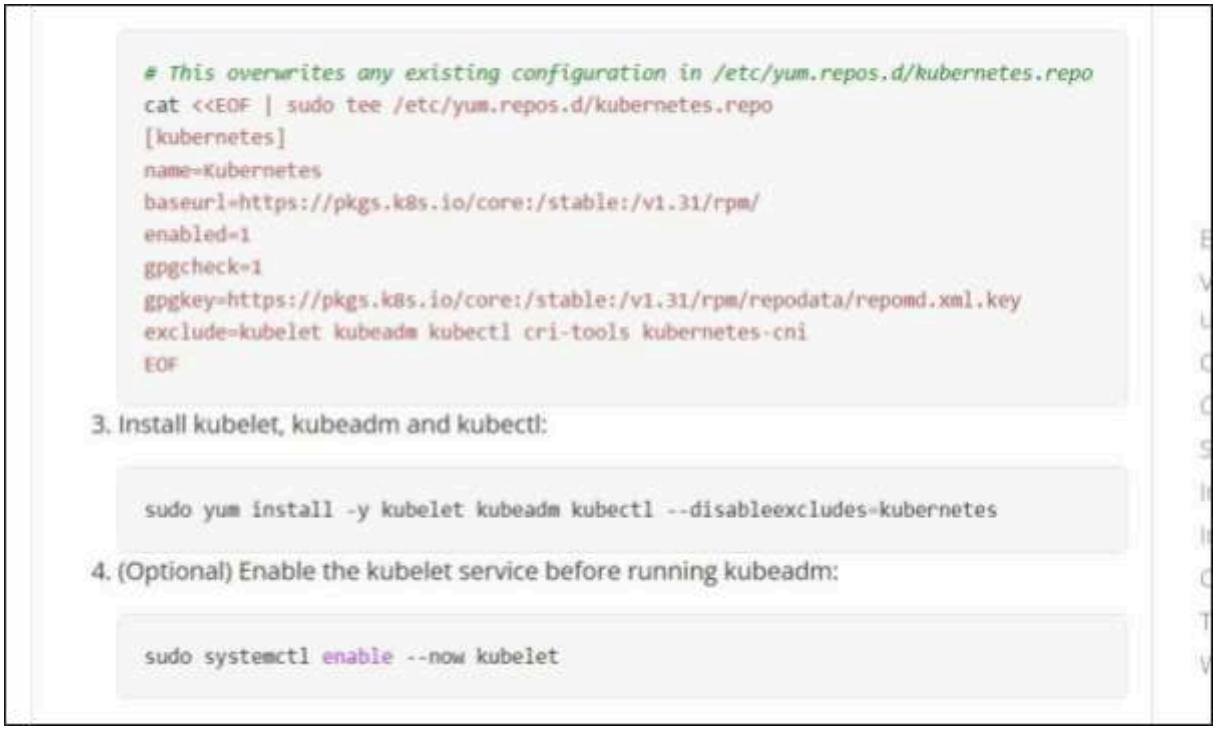
sudo systemctl daemon-reload

sudo systemctl restart docker

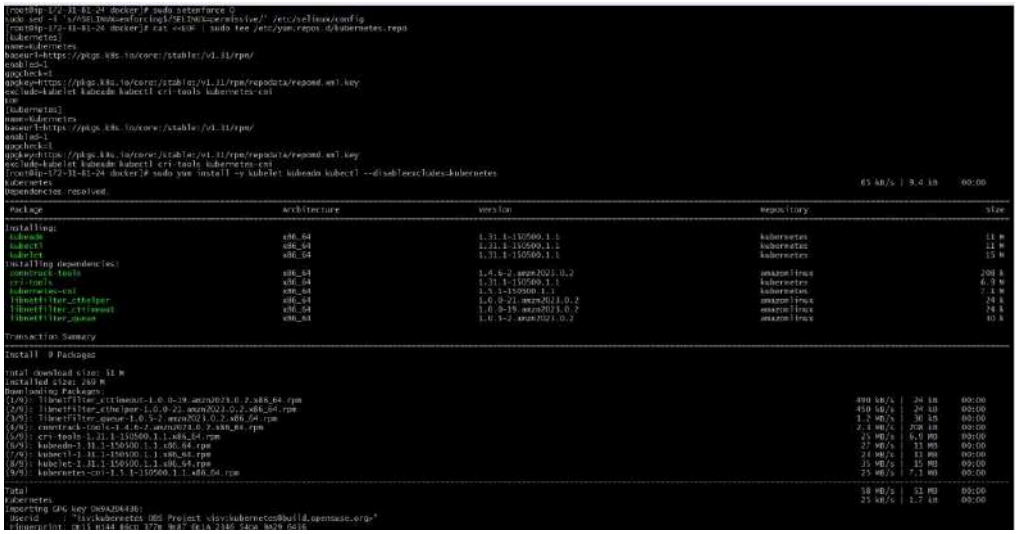


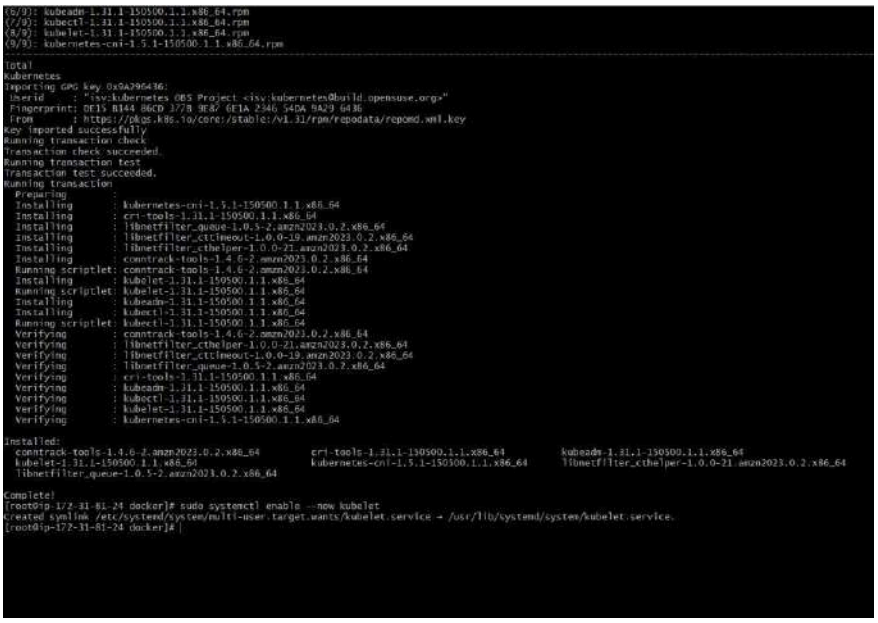
6. Kubernetes installation:
   Search kubeadm installation on your browser and scroll down and select red hat-based distributions.





Copy the above given steps and paste in the terminal. This will create a Kubernetes repository, install kubelet,kubeadm and kubectl and also enable the services.





7. After installing Kubernetes, we need to configure internet options to allow bridging.
   sudo swapoff -a
   echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a
   /etc/sysctl.conf
   sudo sysctl -p

```
[root@ip-172-31-81-24 docker]# sudo swapoff -a
[root@ip-172-31-81-24 docker]# echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf
net.bridge.bridge-nf-call-iptables=1
[root@ip-172-31-81-24 docker]# sudo sysctl -p
net.bridge.bridge-nf-call-iptables = 1
[root@ip-172-31-81-24 docker]#
```

8. Initializing kubecluster:
   sudo kubeadm init --pod-network-cidr=10.244.0.0/16

```
[root@ip-172-31-81-24 docker]# sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[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
error execution phase preflight: [preflight] Some fatal errors occurred:
        [ERROR NumCPU]: the number of available CPUs 1 is less than the required 2
        [ERROR Mem]: the system RAM (949 MB) is less than the minimum 1700 MB
[preflight] If you know what you are doing, you can make a check non-fatal with `--ignore-preflight-errors=...`
To see the stack trace of this error execute with --v=5 or higher
[root@ip-172-31-81-24 docker]# sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errors=NumCPU,Mem
[init] Using Kubernetes version: v1.31.0
```

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

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

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

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

  export KUBECONFIG=/etc/kubernetes/admin.conf

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

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

kubeadm join 172.31.81.24:6443 --token 4a91z3.yz6rwmmkf9yncyd2 \
        --discovery-token-ca-cert-hash sha256:3404bd1bcdd9cf90a003673f622d1672acb4c6ce7c15c4738c80a0a1560fe70d
[root@ip-172-31-81-24 docker]#
```

9. The mkdir command that is generated after initialization has to be copy pasted in the terminal.

```
[root@ip-172-31-81-24 docker]#  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

10. Then, add a common networking plugin called flannel:
    kubectl apply -f
    https://raw.githubusercontent.com/coreos/flannel/master/Documentation/
    kube-flannel.yml

```
[root@ip-172-31-81-24 docker]# 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
[root@ip-172-31-81-24 docker]#
```

11. Apply this deployment file using this command to create a deployment kubectl apply -f
    https://k8s.io/examples/application/deployment.yaml

```
[root@ip-172-31-81-24 docker]# kubectl apply -f https://k8s.io/examples/application/deployment.yaml
deployment.apps/nginx-deployment created
[root@ip-172-31-81-24 docker]#
```

11. Use kubectl get pods to check if pod is working correctly

```
[root@ip-172-31-81-24 docker]# kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-8jdlf    0/1     Pending   0          18s
```

12. To change status from pending to running use following command: kubectl describe pod nginx.

```
[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.
```

Use the below command to remove taints

```
[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
```

13. Check the pod status

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

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

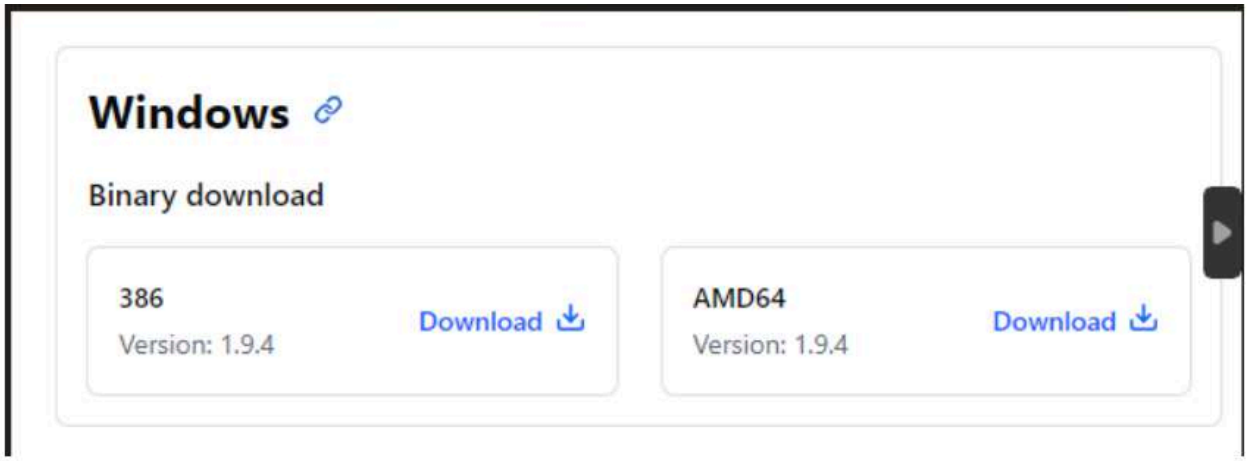
```
[ec2-user@ip-172-31-26-174 ~]$ kubectl port-forward nginx 8081:80
Forwarding from 127.0.0.1:8081 -> 80
Forwarding from [::1]:8081 -> 80
```

15. Verify your deployment
    Open up a new terminal and ssh to your EC2 instance.
    Then, use this curl command to check if the Nginx server is running.
    curl --head http://127.0.0.1:8080

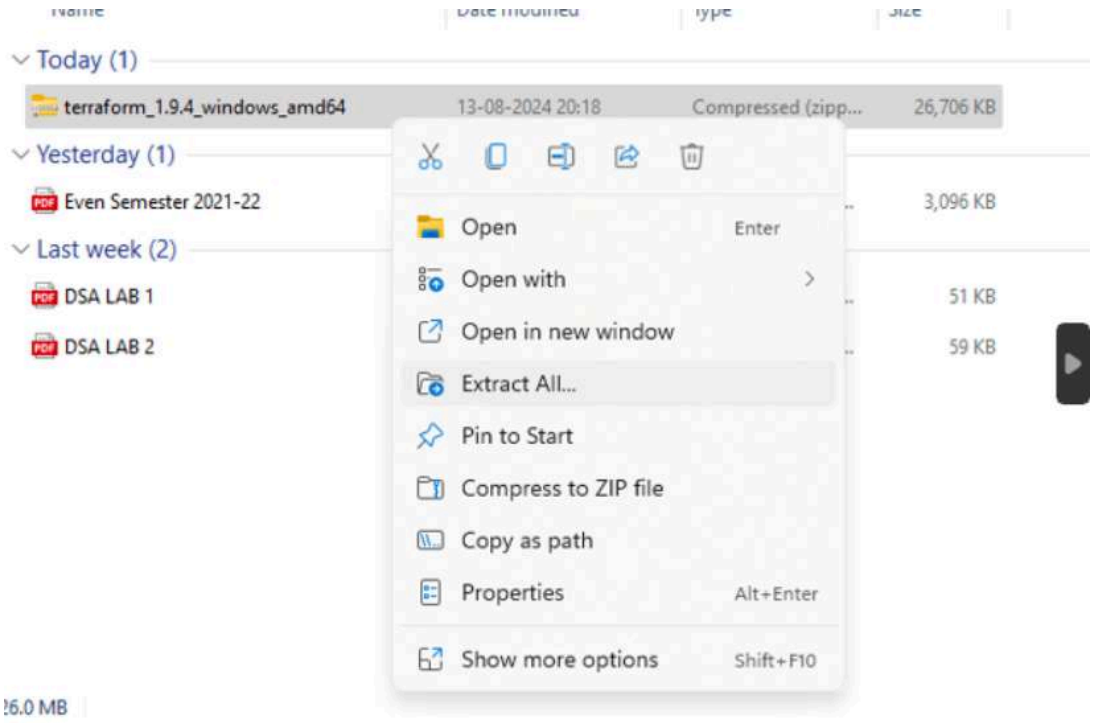
Conclusion: :Firstly I created an EC2 AWS Linux instance successfully.then installed docker and kubernetes successfully. Initialized kubernetes and execute mkdir and chown command successfully. Then I tried to deploy nginx which initially gave an error. Then I deployed (simple-pod.yml) nginx successfully and also checked by using the get pods command.

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

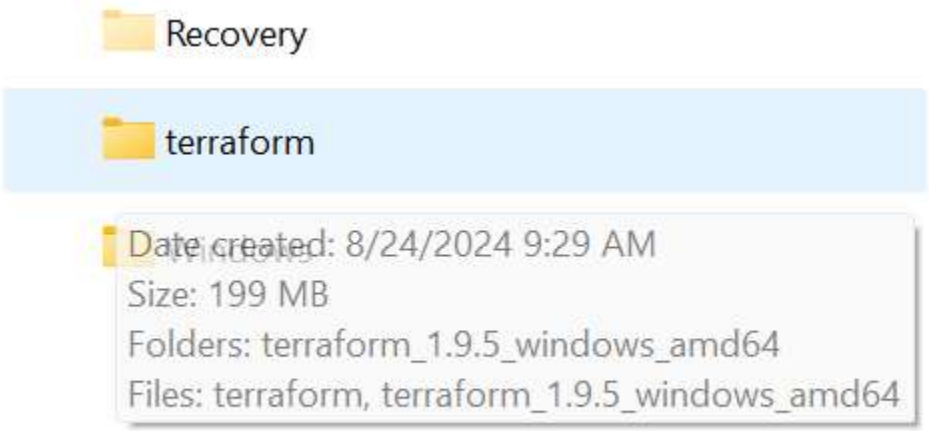
Step 1: Go to <:www.terraform.io/downloads.html> and download the amd file compatible with your device



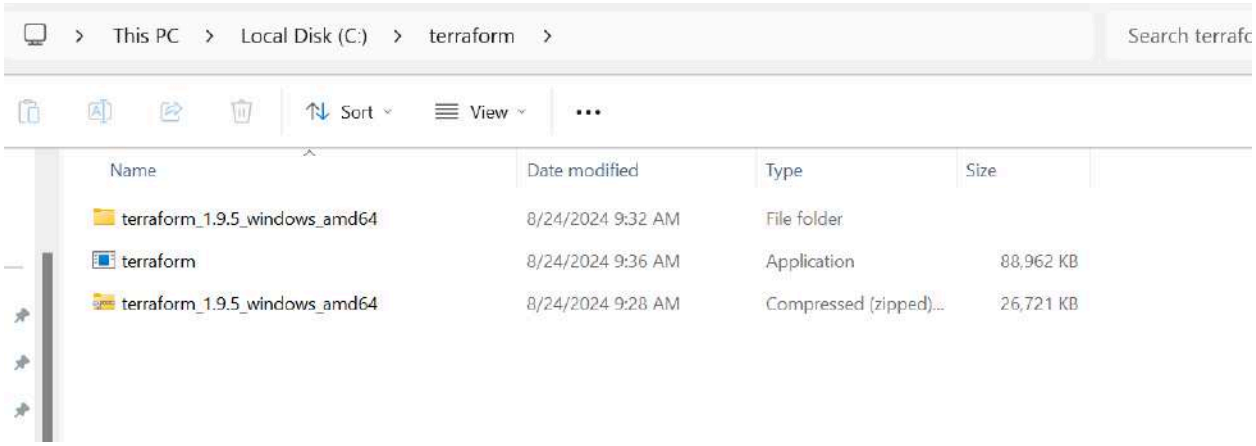
Step 2: It will download a zip file



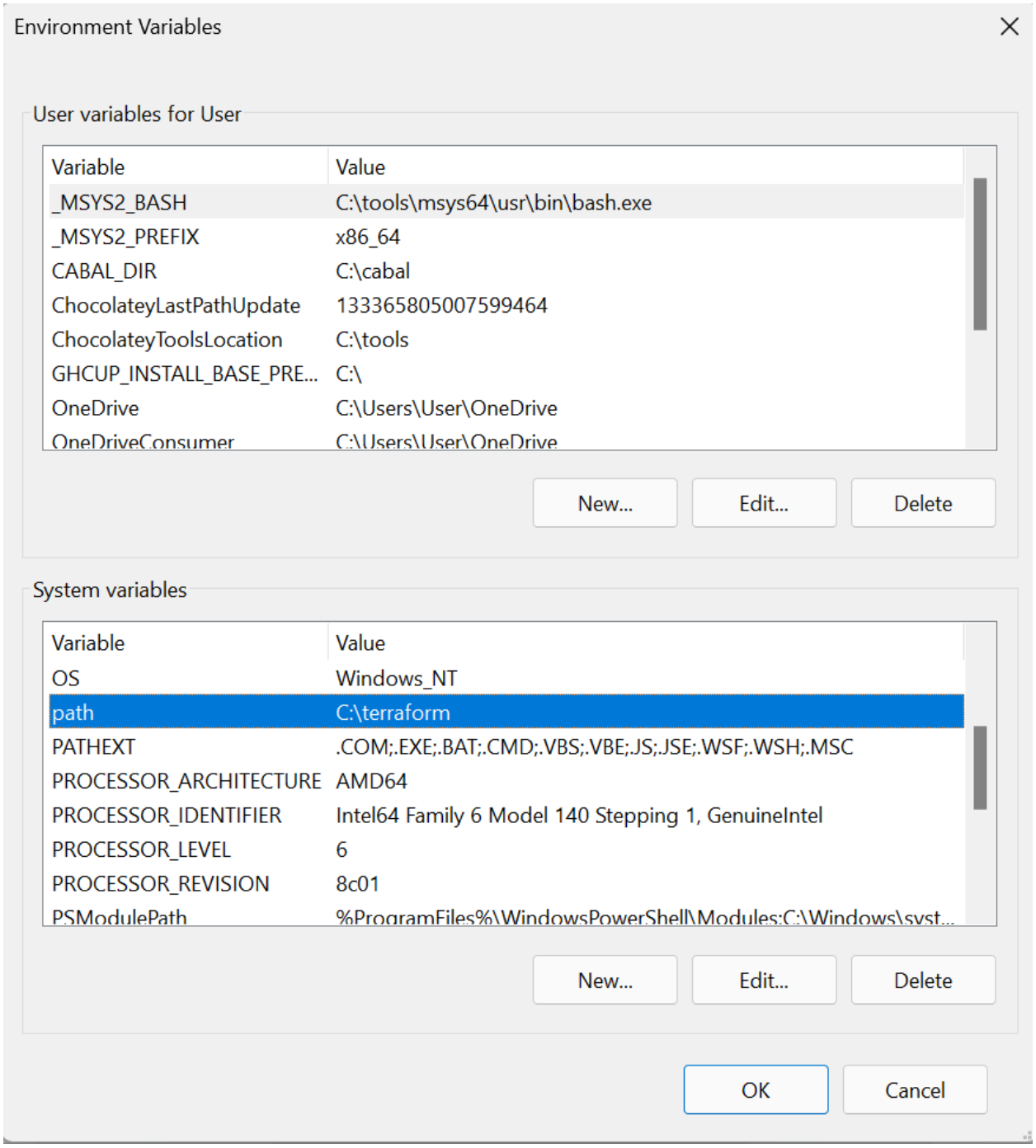
Step 3: Create a folder and name it <terraform> on C drive



Step 4: Move the extracted files in this folder



Step 5: Go to Settings> System Properties> Environment variables and then add a new system variable



Step 6: To confirm proper installation, Open Windows Powershell as administrator and run terraform

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

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

PS C:\Windows\system32> terraform
Usage: terraform [global options] <subcommand> [args]

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

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

All other commands:
  console       Try Terraform expressions at an interactive command prompt
  fmt           Reformat your configuration in the standard style
  force-unlock  Release a stuck lock on the current workspace
  get           Install or upgrade remote Terraform modules
  graph         Generate a Graphviz graph of the steps in an operation
  import        Associate existing infrastructure with a Terraform resource
  login         Obtain and save credentials for a remote host
  logout        Remove locally-stored credentials for a remote host
  metadata      Metadata related commands
  output        Show output values from your root module
```

Experiment No. 6

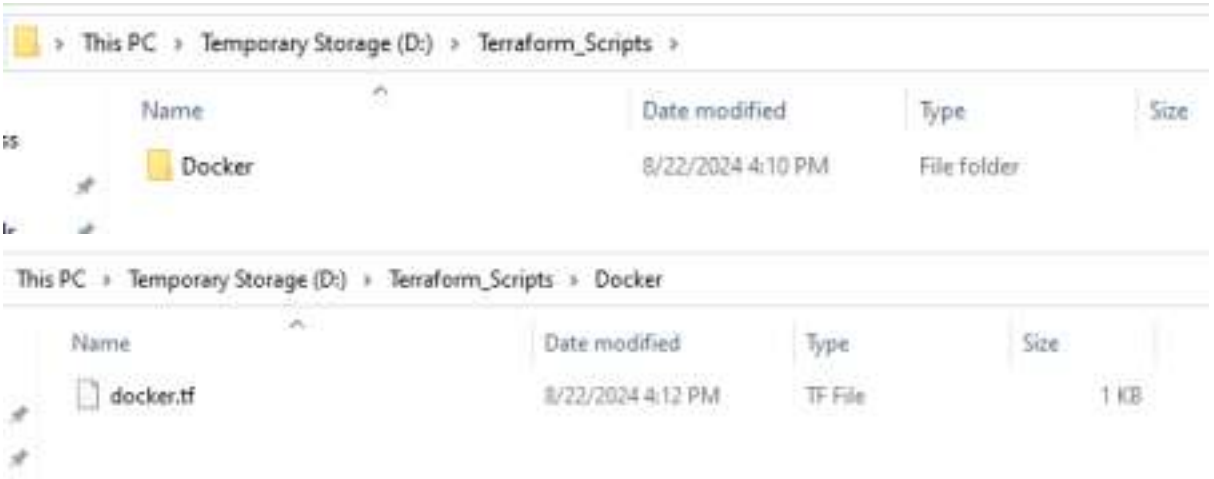
Aim: To create a docker image using terraform

1. Check if the docker is functioning properly on your device by running some commands

```
PS C:\ProgramData\Microsoft\Windows\Start Menu> Docker --version
Docker version 27.1.1, build 6312585
PS C:\ProgramData\Microsoft\Windows\Start Menu> docker

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

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

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

2. To write the code for our pipeline create a new folder

3. Create a new folder named ‘Docker’ in the ‘TerraformScripts’ folder. Then create a new docker.tf file using a text editor and write the following contents into it to create a Ubuntu Linux container.



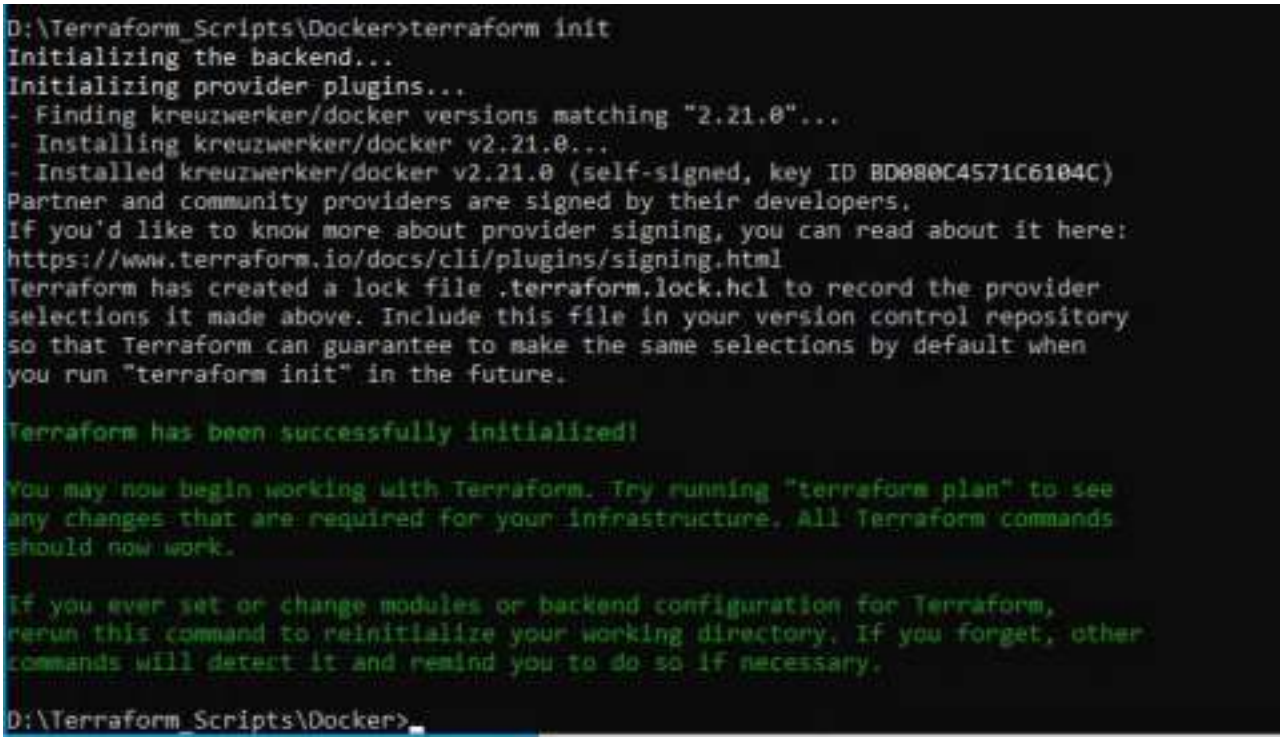
4. Write the code in that docker.tf file and save it

docker.tf - Notepad

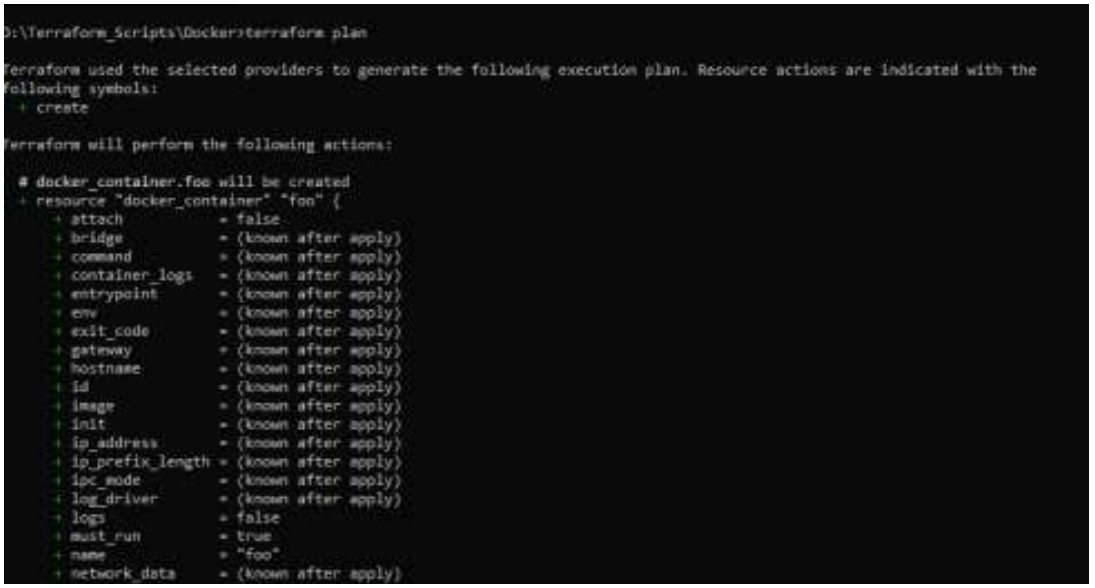
File Edit Format View Help

```
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"
}
```

5. Open the folder in command prompt to run terraform commands. The first step is to initialize terraform using the command > terraform init



5. The next command to run > terraform plan to check the available resources



```
    + healthcheck (known after apply)

    + labels (known after apply)
  }

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

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

6. Now to finally execute the code, run the command terraform apply

```
D:\Terraform_Scripts\Docker>terraform apply

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

Terraform will perform the following actions:

  # docker_container.foo will be created
  + resource "docker_container" "foo" {
      + attach           = false
      + bridge           = (known after apply)
      + command          = (known after apply)
      + container_logs   = (known after apply)
      + entrypoint       = (known after apply)
      + env              = (known after apply)
      + exit_code        = (known after apply)
      + gateway          = (known after apply)
      + hostname         = (known after apply)
      + id               = (known after apply)
      + image            = (known after apply)
      + init             = (known after apply)
      + ip_address       = (known after apply)
      + ip_prefix_length = (known after apply)
      + ipc_mode         = (known after apply)
      + log_driver       = (known after apply)
      + logs             = false
      + must_run         = true
      + name             = "foo"
      + network_data     = (known after apply)
      + read_only        = false
      + remove_volumes   = true
      + restart          = "no"
      + rm               = false
      + 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 30s [id=sha256:2639065960d42ad38ae991477166927770a9ff8779a62ad93a7f4fe82e3e1f
ubuntu:latest] docker_container.foo: Creating...
```

7. Here you can see the ubuntu image created

```
D:\Terraform_Scripts\Docker>docker images
REPOSITORY                                  TAG                               IMAGE ID       CREATED          SIZE
mcr.microsoft.com/dotnet/framework/aspnet   4.8-windowsservercore-ltsc2022   0b1ef1176a57   6 weeks ago      5.43GB
mcr.microsoft.com/dotnet/framework/sdk      4.8-windowsservercore-ltsc2022   c3f8c2735565   6 weeks ago      9.04GB
mcr.microsoft.com/dotnet/framework/runtime  4.8-windowsservercore-ltsc2022   e69ea8a5ec1b   6 weeks ago      5.1GB
mcr.microsoft.com/windows/servercore        ltsc2022                          e60f47e635b7   7 weeks ago      4.84GB
mcr.microsoft.com/windows/nanoserver        ltsc2022                          f0ca29645006   7 weeks ago      292MB
ubuntu                                      latest                            2dc39ba859dc   2 minutes ago    77.8MB
```

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

```
docker_image.ubuntu: Refreshing state... [id=sha256:[illegible]ubuntu:latest]
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following sy
mbols:
  - destroy
Terraform will perform the following actions:

  # docker_image.ubuntu will be destroyed
  - resource "docker_image" "ubuntu" {
      - id           = "sha256:[illegible]ubuntu:latest" -> null
      - image_id     = "sha256:[illegible]" -> null
      - latest       = "sha256:[illegible]" -> null
      - name         = "ubuntu:latest" -> null
      - repo_digest  = "ubuntu@sha256:[illegible]" -> null
    }

Plan: 0 to add, 0 to change, 1 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_image.ubuntu: Destroying... [id=sha256:[illegible]ubuntu:latest]
docker_image.ubuntu: Destruction complete after 1s

Destroy complete! Resources: 1 destroyed.
```

9. The docker image is destroyed

```
D:\Terraform_Scripts\Docker>docker images
REPOSITORY                                  TAG                               IMAGE ID       CREATED       SIZE
mcr.microsoft.com/dotnet/framework/aspnet   4.8-windowsservercore-ltsc2022   0b1ef1176a57   6 weeks ago   5.43GB
mcr.microsoft.com/dotnet/framework/sdk      4.8-windowsservercore-ltsc2022   c3f8c2735565   6 weeks ago   9.04GB
mcr.microsoft.com/dotnet/framework/runtime  4.8-windowsservercore-ltsc2022   e69ea8a5ec1b   6 weeks ago   5.1GB
mcr.microsoft.com/windows/servercore        ltsc2022                          e60f47e635b7   7 weeks ago   4.84GB
mcr.microsoft.com/windows/nanoserver        ltsc2022                          f0ca29645006   7 weeks ago   292MB

D:\Terraform_Scripts\Docker>
```

Advance DevOps

Experiment 7

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

Steps:

1. Firstly, we will ensure whether docker is installed or not by running docker -v in the command prompt.

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

2. Run docker login command and add your username and password for docker.

```
C:\Users\Dell>docker login
Authenticating with existing credentials...
Stored credentials invalid or expired
Log in with your Docker ID or email address to push and pull images from Docker Hub. If you don't have a Docker ID, head
over to https://hub.docker.com/ to create one.
You can log in with your password or a Personal Access Token (PAT). Using a limited-scope PAT grants better security and
is required for organizations using SSO. Learn more at https://docs.docker.com/go/access-tokens/

Username (dimple866): dimple866
Password:

Login Succeeded
```

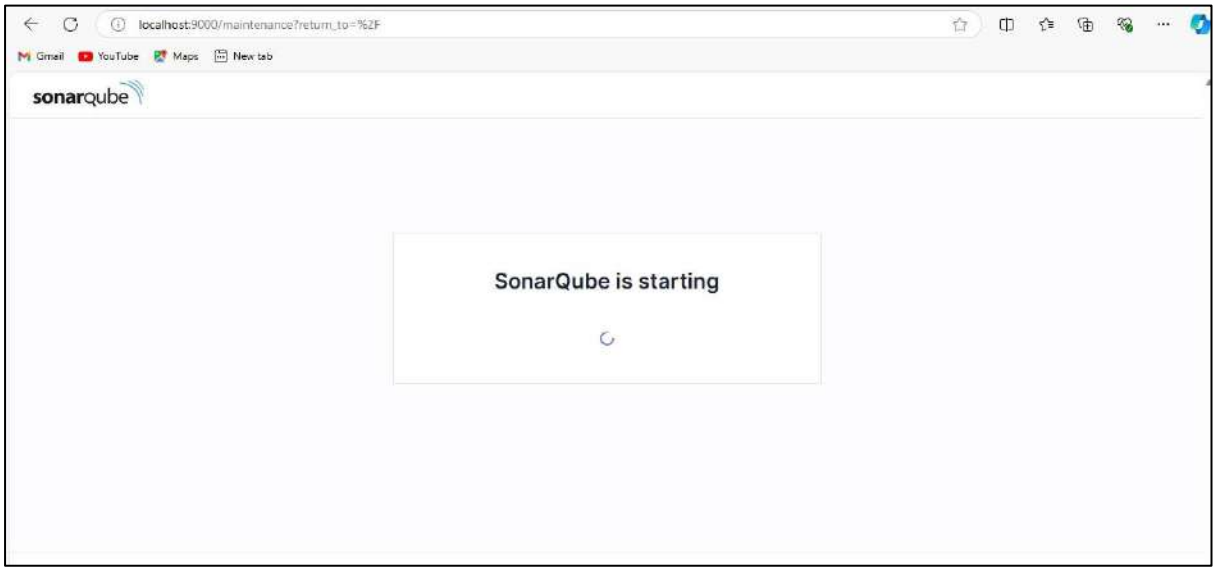
3. Run docker pull SonarQube command to install SonarQube image.

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

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

4. Run docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest Command to run the sonarqube.

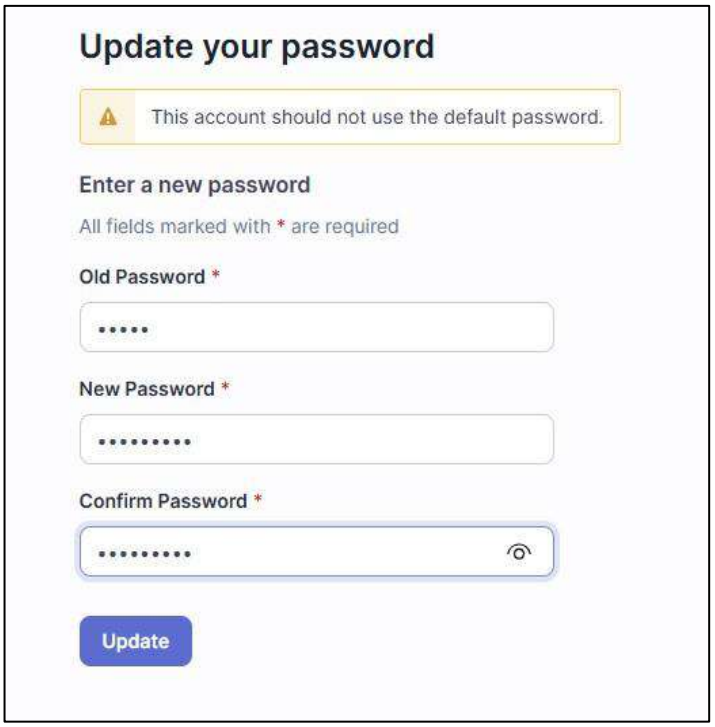
```
C:\Users\Dell>docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
ac1f985dedebc00a642a4c69a502d611389e8f9fa46610febe75aa5021767cab
```

5. Once the container is running go to your web browser and check status of SonarQube at port 9000.

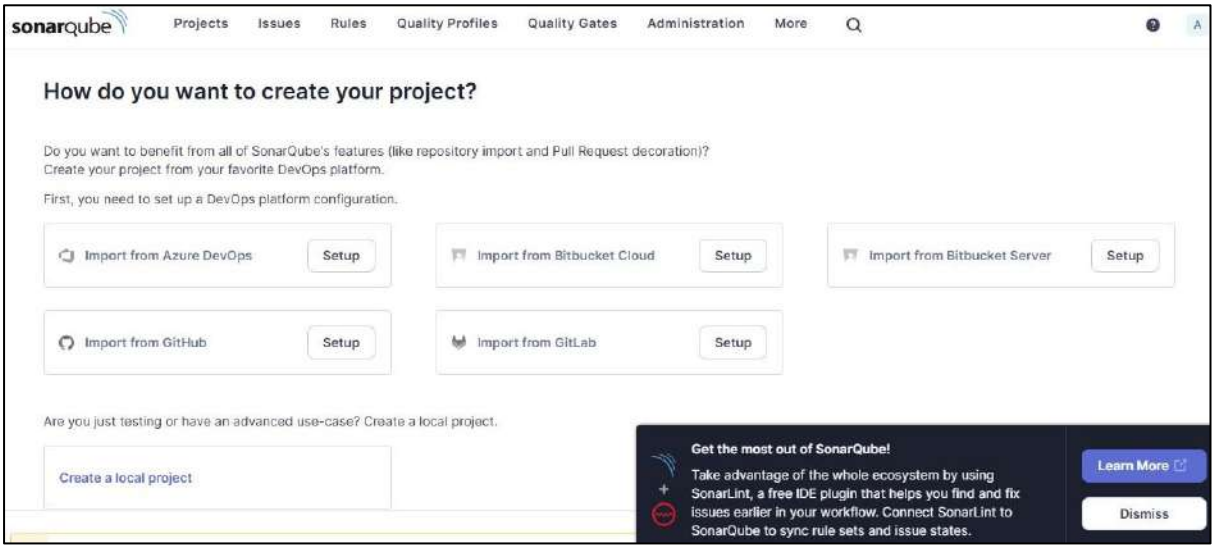


6. Once SonarQube is started it will redirect you to login page. The login and password both for SonarQube is ‘admin’

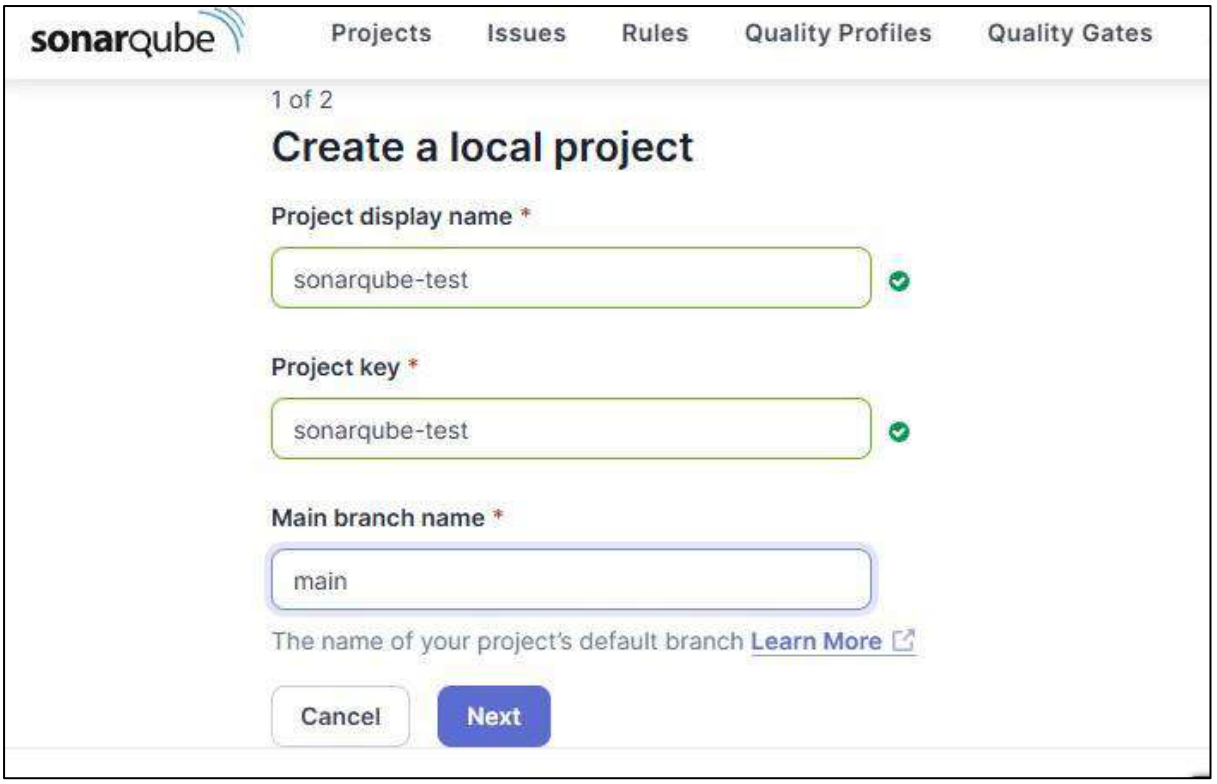
7. Change the password for your SonarQube account.



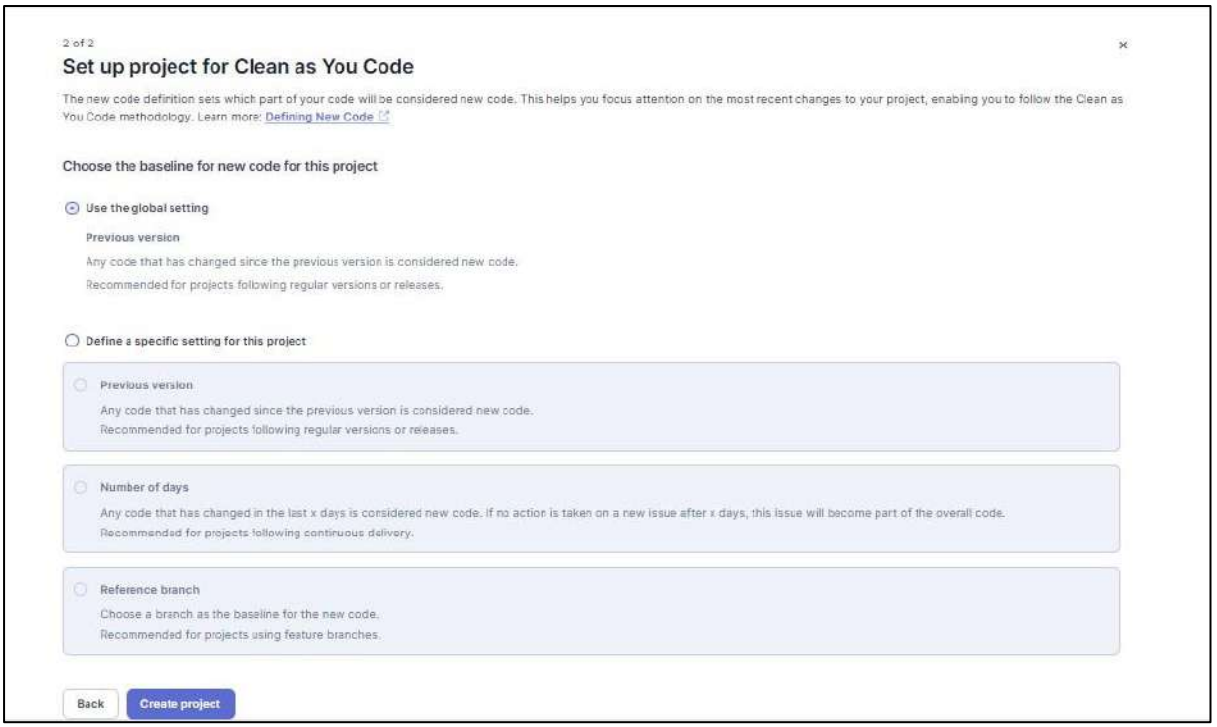
8. After changing the password, you will be directed to this screen. Click on Create a Local Project.



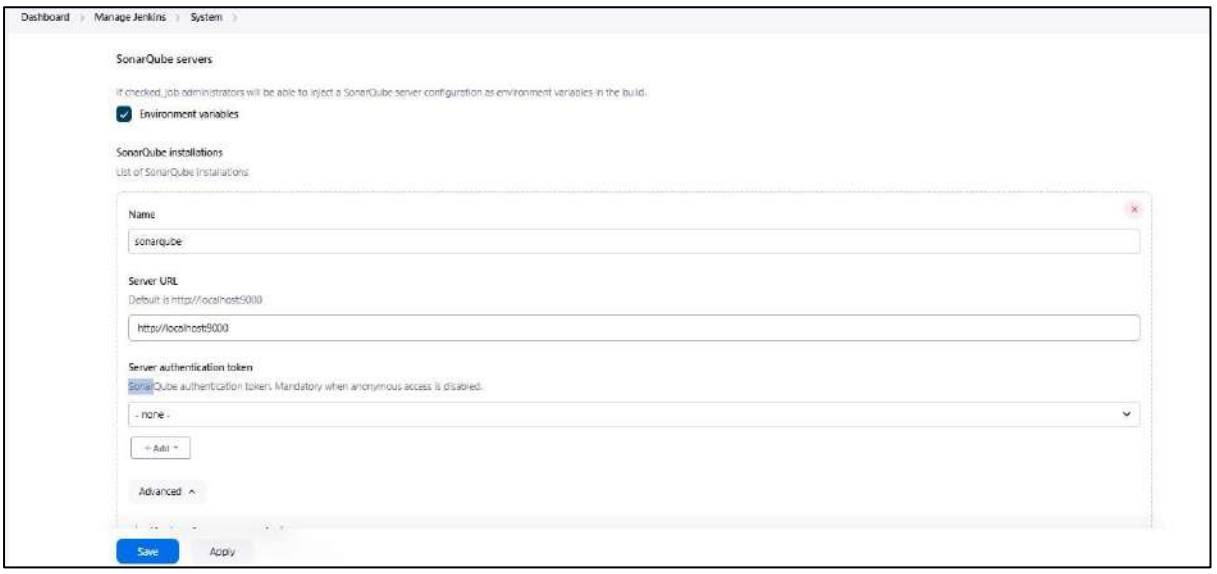
9. Add name of the project and project key and select the main branch name and click on next.



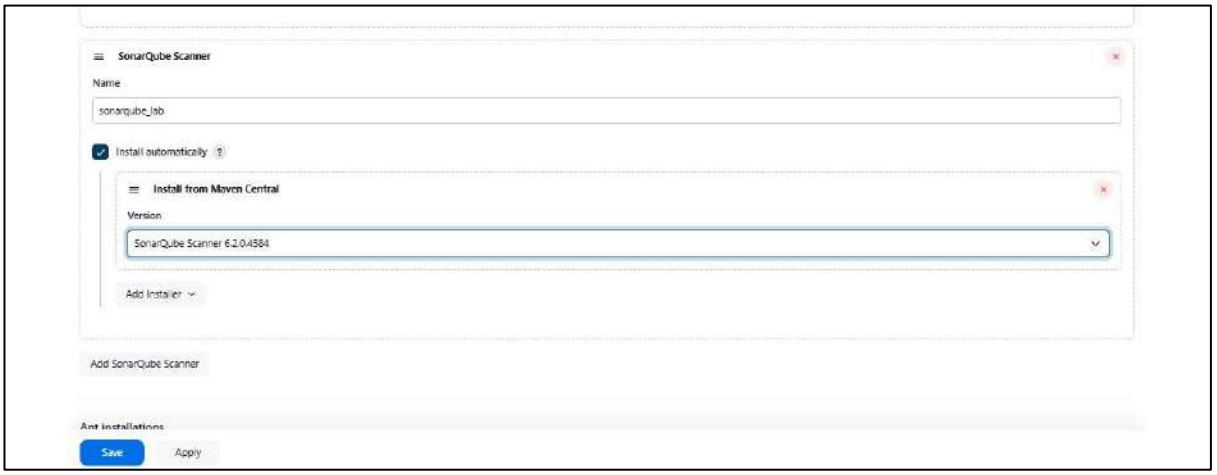
10. Set up the project as required and click on create.



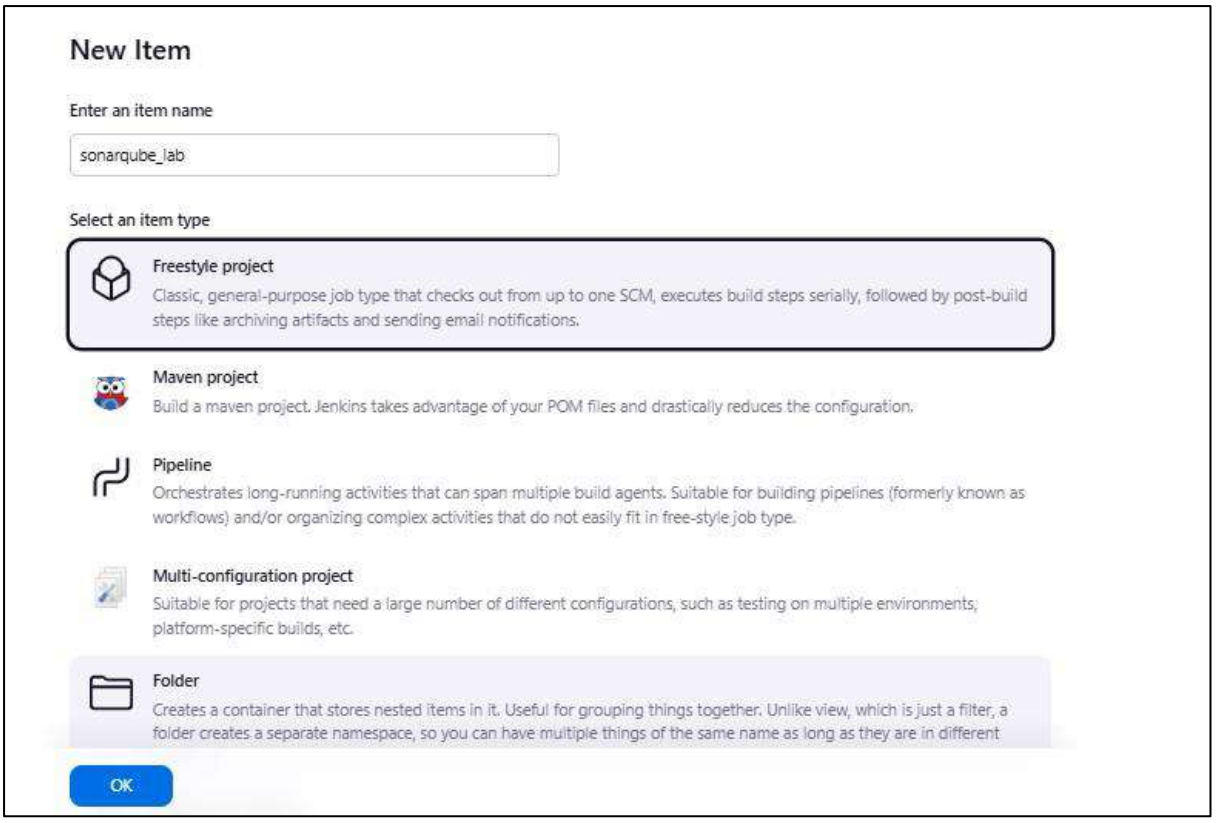
11. Go to Jenkins dashboard->Manage Jenkins->System and scroll down to SonarQube installations. Enter the name and URL in the fields and save the changes.



12. In SonarQube Scanner add the latest version then apply the changes and save it.



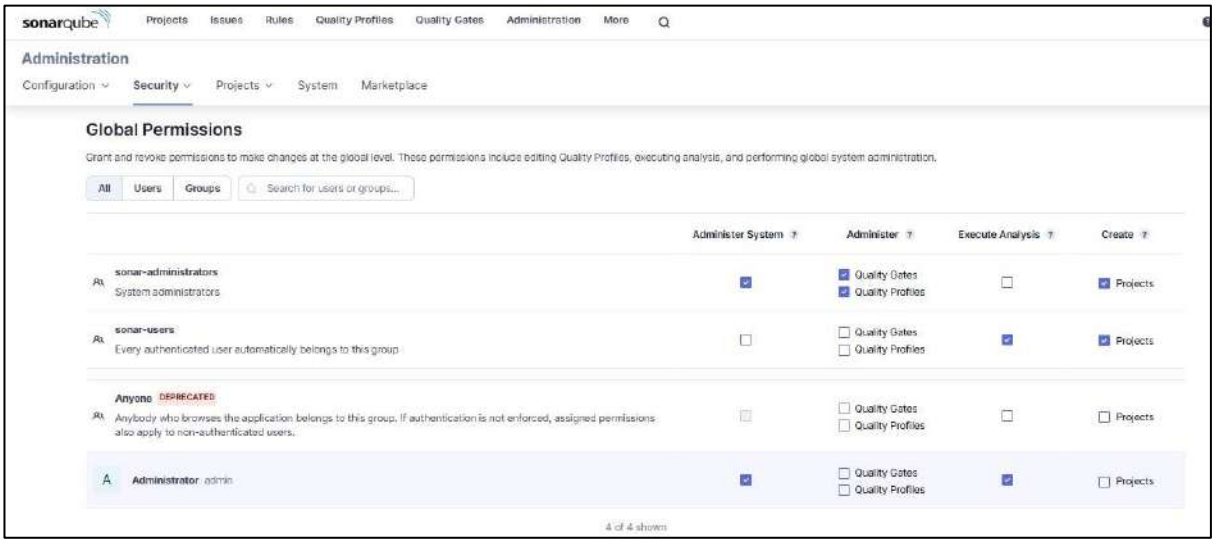
13. Go to Jenkins and then create a new item, enter the item name and select “Freestyle project” and then click on ok.



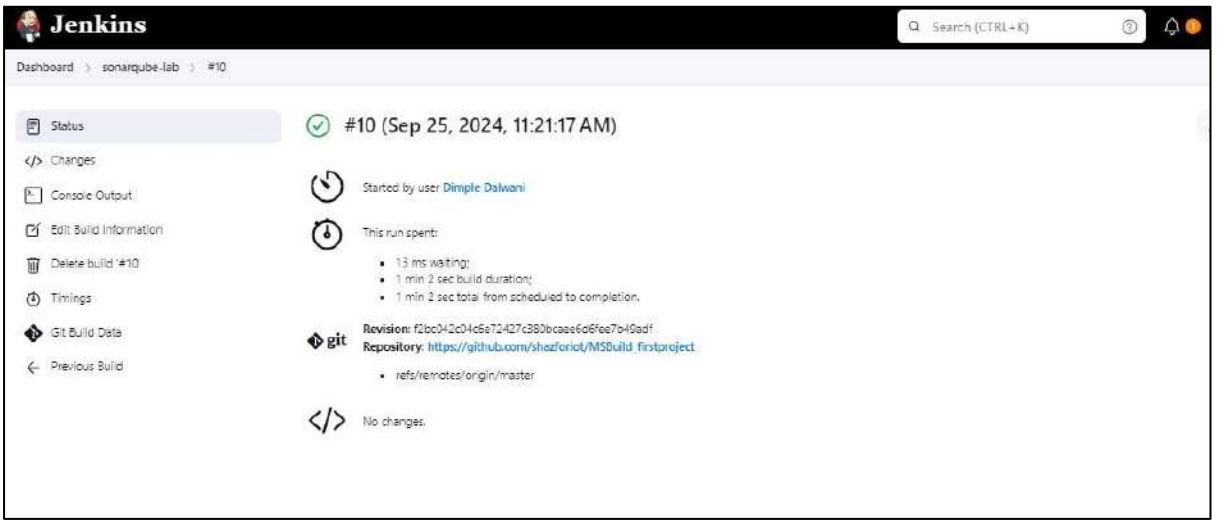
14. Use this GitHub repository in Source Code Management.
https://github.com/shazforiot/MSBuild_firstproject

15. In Analysis properties, mention the SonarQube Project Key, Login, Password, Source path and Host URL.

16. Now, you need to grant the local user (here admin user) permissions to Execute the Analysis stage on SonarQube. For this go to http://localhost:<port_number>/admin/permissions and check the ‘Execute Analysis’ checkbox under Administrator.



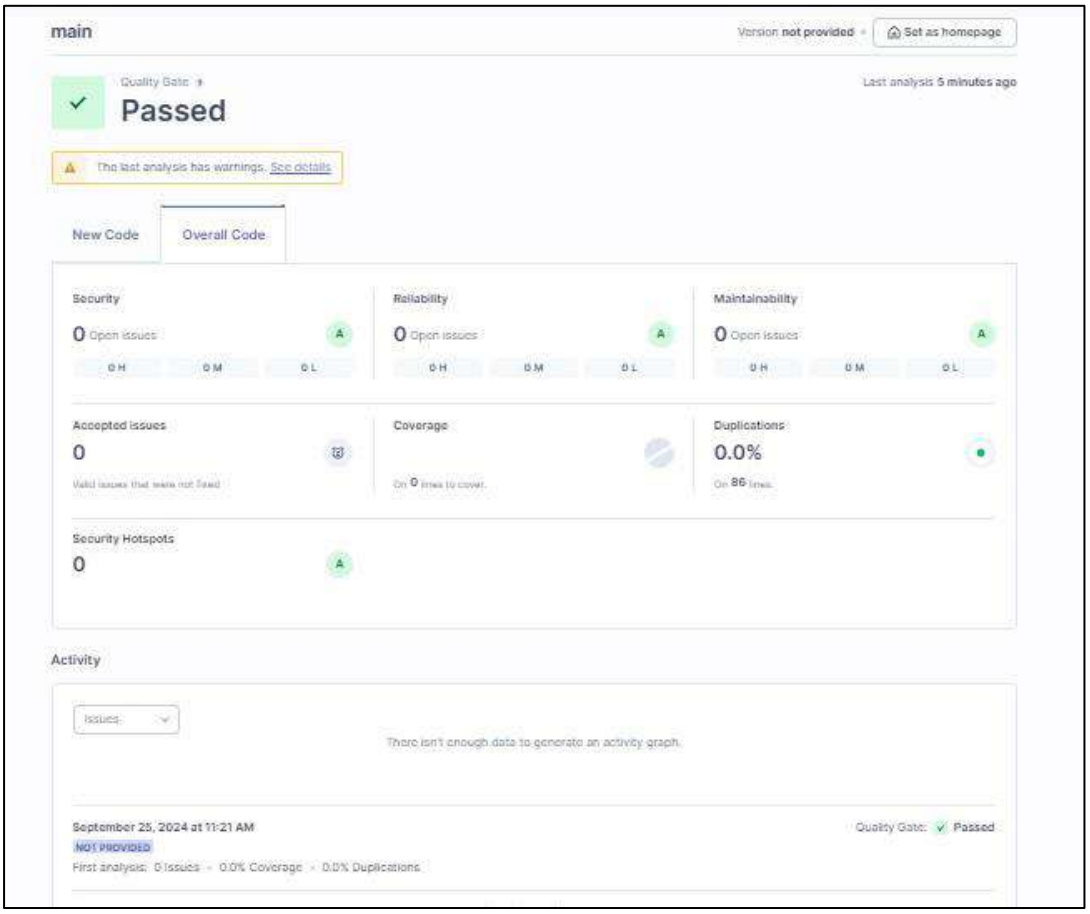
17. Go to the job you have just built and click on Build Now.



18. Check the console Output

```
for block at line 17. Keep only the first 100 references.
23:13:58.632 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.
23:13:58.632 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.
23:13:58.632 INFO  CPD Executor CPD calculation finished (done) | time=94361ms
23:13:58.695 INFO  SCM revision ID 'ba799ba7e1b576f04a4612322b0412c5e6e1e5e4'
23:15:46.177 INFO  Analysis report generated in 14542ms, dir size=127.2 MB
23:15:55.734 INFO  Analysis report compressed in 9547ms, zip size=29.6 MB
23:15:59.127 INFO  Analysis report uploaded in 3391ms
23:15:59.132 INFO  ANALYSIS SUCCESSFUL, you can find the results at: http://127.0.0.1:9000/dashboard?id=sonarqube-test
23:15:59.132 INFO  Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
23:15:59.132 INFO  More about the report processing at http://127.0.0.1:9000/api/ce/task?id=fbad731f-dcba-45c3-bfdd-b2ed2fec3a9e
23:16:05.629 INFO  Analysis total time: 10:30.120 s
23:16:05.636 INFO  SonarScanner Engine completed successfully
23:16:06.248 INFO  EXECUTION SUCCESS
23:16:06.273 INFO  Total time: 10:47.728s
[Pipeline] }
[Pipeline] // withSonarQubeEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

19. Go back to SonarQube and check the project.



Conclusion: While performing this experiment there was an issue in creating sonarqube docker image and we resolved it by logging in to the docker desktop and performing it through the terminal. Other than this we created a freestyle project and entered the sonarqube credentials and then performed build.

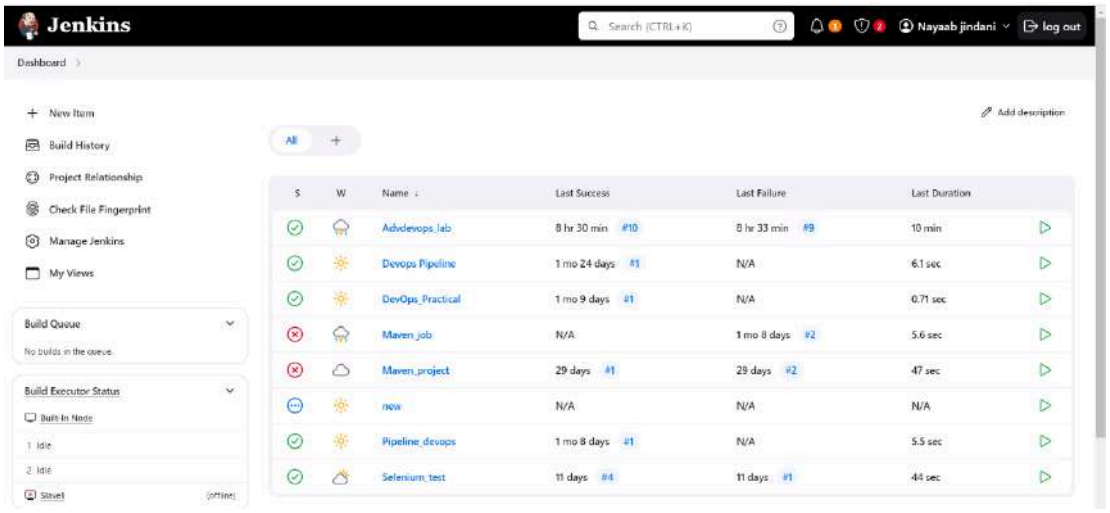
Advance DevOps

Experiment 8

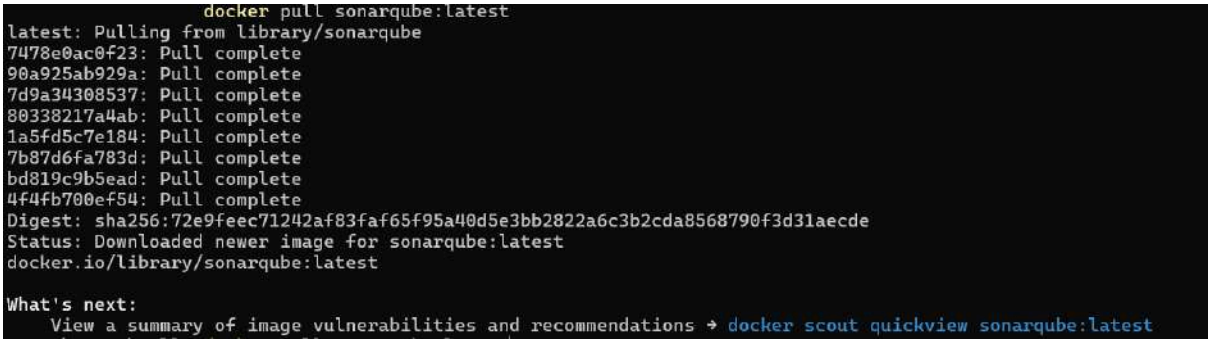
Aim: Create a Jenkins CICD Pipeline with SonarQube / GitLab Integration to perform a static analysis of the code to detect bugs, code smells, and security vulnerabilities on a sample Web / Java / Python application.

Steps:

1. Open the Jenkins dashboard.



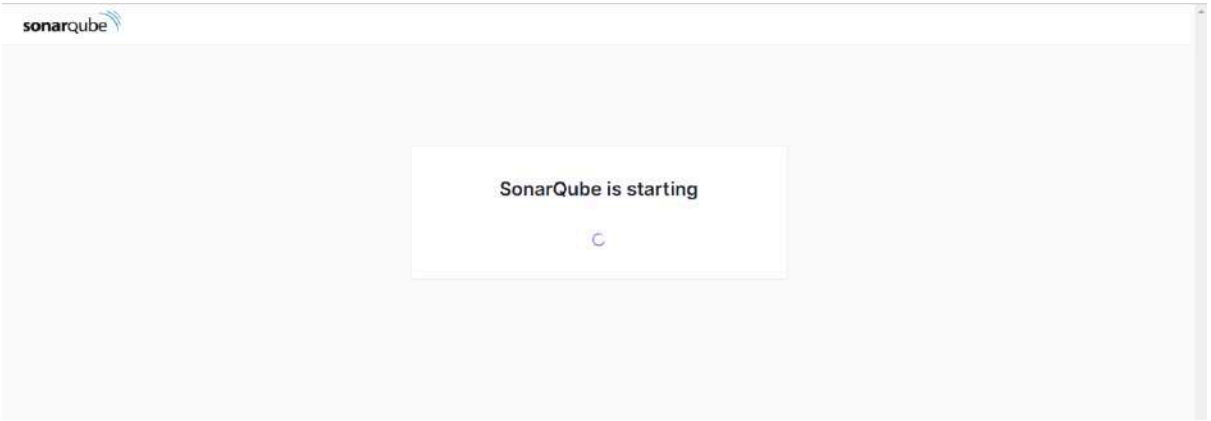
2. First, we will pull the latest version of sonar qube image from the docker hub using the command:
   docker pull sonarqube:latest



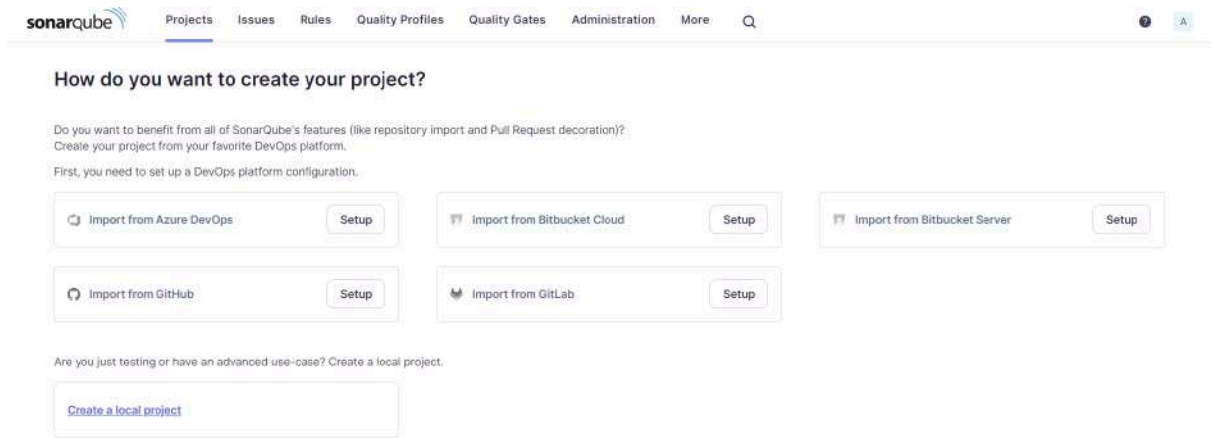
3. Next, we will run SonarQube in a docker container

```
PS C:\Users\Dell> docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
67aeea599cfc48e12d50da592eff01d8257f58e6c1bffdf50446066e5f2a8844
```

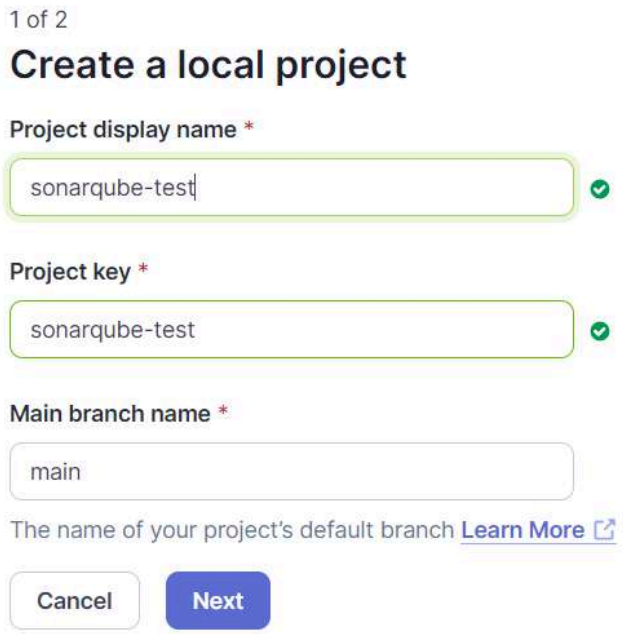
4. Once the container is running, we will check the status of sonar qube on the port 9000. It will show “Sonar qube is starting”



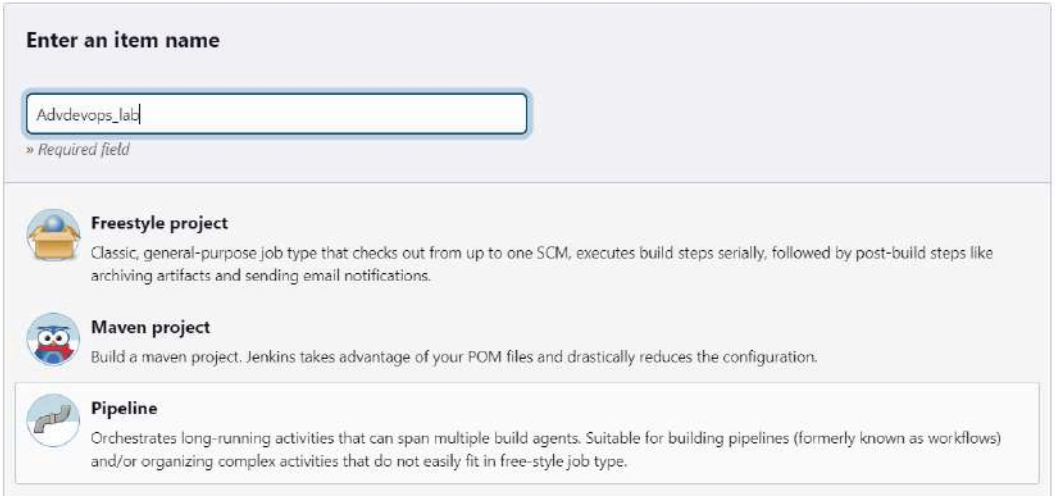
5. Now login to SonarQube using username and password.



6. Click on create a local project option from the dashboard and give a name to the project, click on next and complete the setup.



7. Go to Jenkins dashboard and create a new item by giving a name and select pipeline option.



8. Scroll down to pipeline script and enter the following script:
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/"
}
}
}
```

(Change the path and credentials)

9. Now run the build.

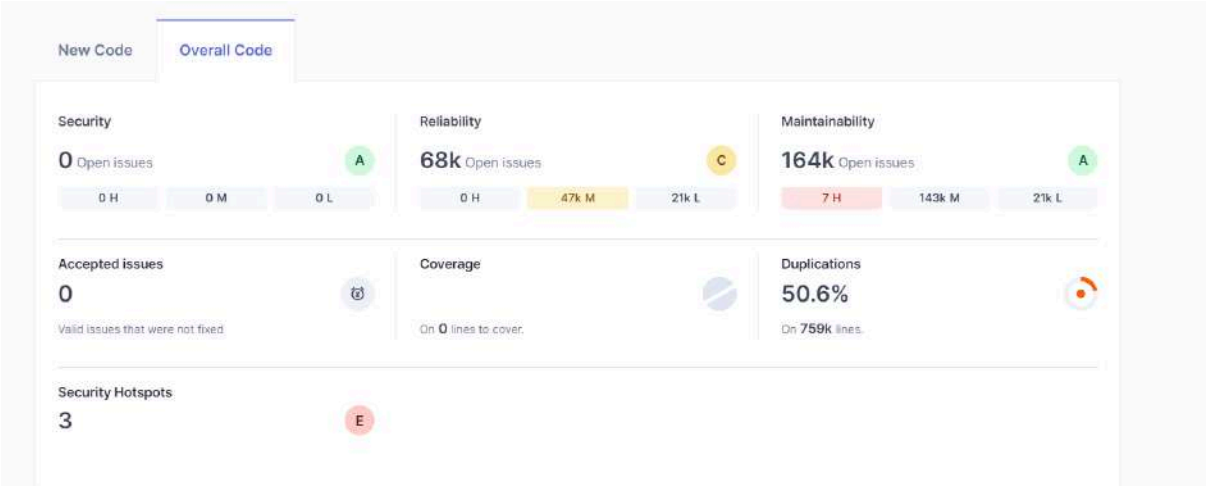
Console Output

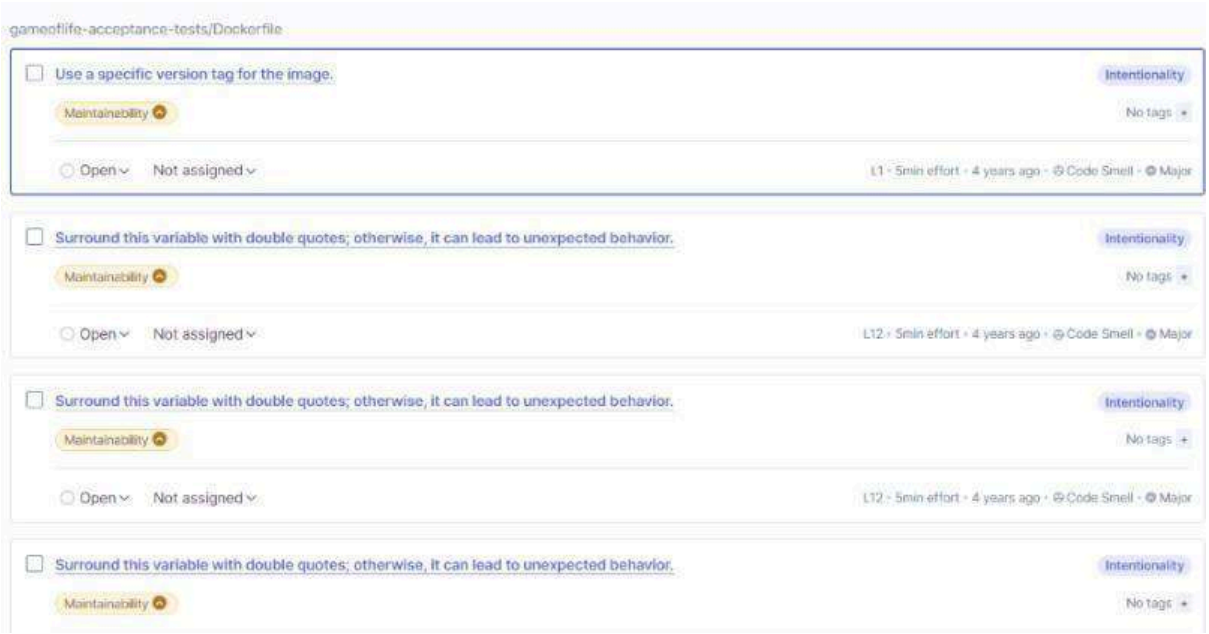
```
Started by user Nayaab jindani
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in C:\ProgramData\Jenkins\.jenkins\workspace\Advdevops_lab
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Cloning the GitHub Repo)
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
 > git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\Advdevops_lab\.git # timeout=10
Fetching changes from the remote Git repository
 > git.exe config remote.origin.url https://github.com/shazforiot/GOL.git # timeout=10
Fetching upstream changes from https://github.com/shazforiot/GOL.git
 > git.exe --version # timeout=10
 > git --version # 'git version 2.45.2.windows.1'
 > git.exe fetch --tags --force --progress -- https://github.com/shazforiot/GOL.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision ba799ba7e1b576f04a4612322b0412c5e6e1e5e4 (refs/remotes/origin/master)
 > git.exe config core.sparsecheckout # timeout=10
 > git.exe checkout -f ba799ba7e1b576f04a4612322b0412c5e6e1e5e4 # timeout=10
 > git.exe branch -a -v --no-abbrev # timeout=10
 > git.exe branch -D master # timeout=10
```

The build is successful.

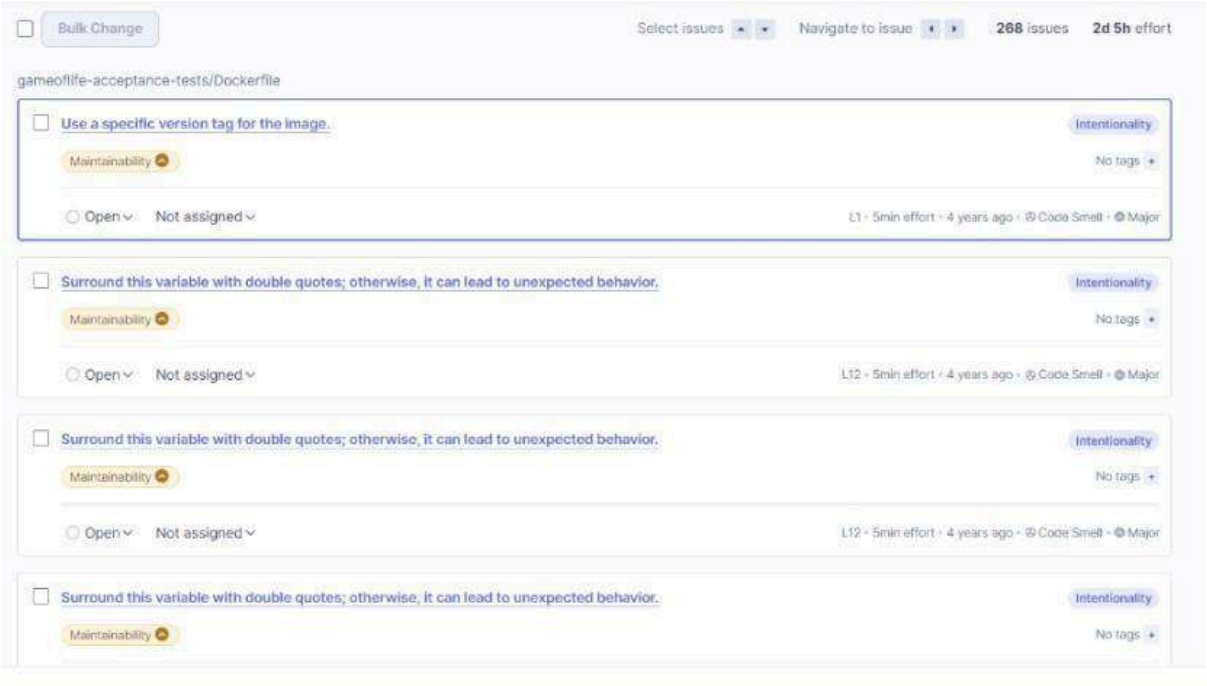
```
for block at line 17. Keep only the first 100 references.
23:13:58.632 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/gui/util/TextAreaCellRenderer.html
for block at line 296. Keep only the first 100 references.
23:13:58.632 WARN  Too many duplication 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.
23:13:58.632 INFO  CPD Executor CPD calculation finished (done) | time=94361ms
23:13:58.695 INFO  SCM revision ID 'ba799ba7e1b576f04a4612322b0412c5e6e1e5e4'
23:15:46.177 INFO  Analysis report generated in 14542ms, dir size=127.2 MB
23:15:55.734 INFO  Analysis report compressed in 9547ms, zip size=29.6 MB
23:15:59.127 INFO  Analysis report uploaded in 3391ms
23:15:59.132 INFO  ANALYSIS SUCCESSFUL, you can find the results at: http://127.0.0.1:9000/dashboard?id=sonarqube-test
23:15:59.132 INFO  Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
23:15:59.132 INFO  More about the report processing at http://127.0.0.1:9000/api/ce/task?id=fbad731f-dcba-45c3-bfdd-b2ed2fec3a9e
23:16:05.629 INFO  Analysis total time: 10:30.120 s
23:16:05.636 INFO  SonarScanner Engine completed successfully
23:16:06.248 INFO  EXECUTION SUCCESS
23:16:06.273 INFO  Total time: 10:47.728s
[Pipeline] }
[Pipeline] // withSonarQubeEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

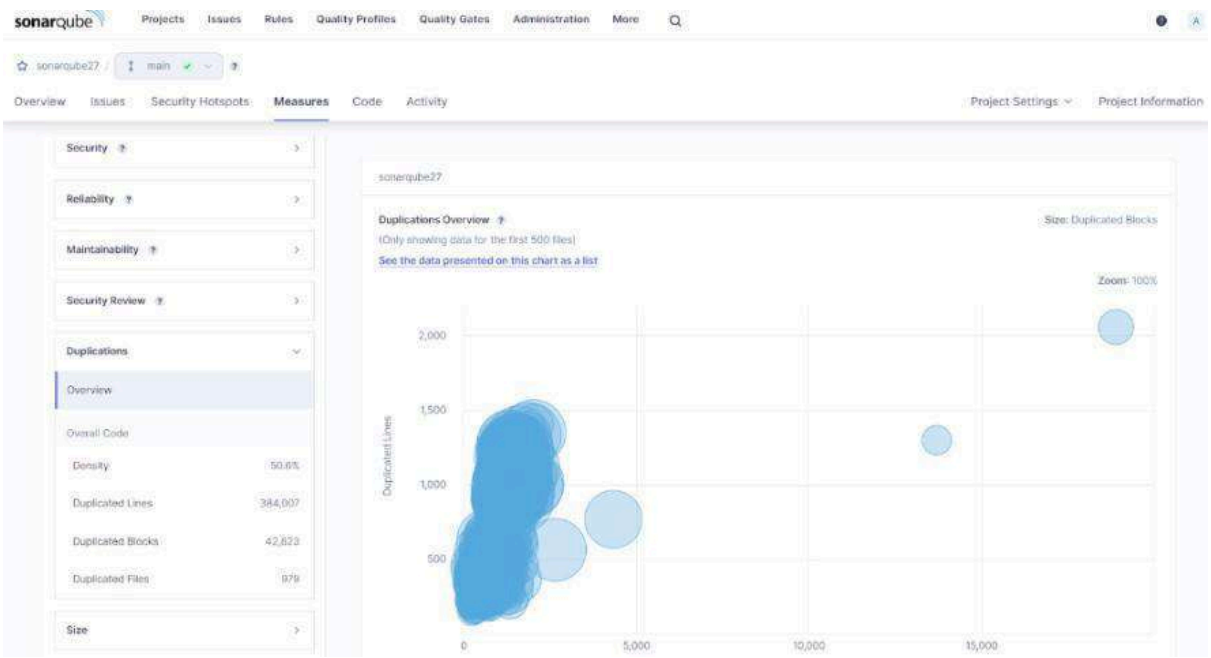
10. Go to sonar qube and check the different errors, code problems, bugs present in the code.


New Code
Overall Code
Security
0 Open issues
A
0 H
0 M
0 L
Reliability
68k Open issues
C
0 H
47k M
21k L
Maintainability
164k Open issues
A
7 H
143k M
21k L
Accepted issues
0
Valid issues that were not fixed
Coverage
On 0 lines to cover.
Duplications
50.6%
On 759k lines.
Security Hotspots
3
E


gameoflife-acceptance-tests/Dockerfile
Use a specific version tag for the image.
Intentionality
Maintainability
No tags
Open
Not assigned
L1 · 5min effort · 4 years ago · Code Smell · Major
Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.
Intentionality
Maintainability
No tags
Open
Not assigned
L12 · 5min effort · 4 years ago · Code Smell · Major
Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.
Intentionality
Maintainability
No tags
Open
Not assigned
L12 · 5min effort · 4 years ago · Code Smell · Major
Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.
Intentionality
Maintainability
No tags


Bulk Change
Select issues
Navigate to issue
13,619 issues
56d effort
gameoflife-core/build/reports/tests/all-tests.html
Add "lang" and/or "xml:lang" attributes to this "<html>" element
Intentionality
Reliability
accessibility wcag2-a
Open
Not assigned
L1 · 2min effort · 4 years ago · Bug · Major
Add "<th>" headers to this "<table>".
Intentionality
Reliability
accessibility wcag2-a
Open
Not assigned
L9 · 2min effort · 4 years ago · Bug · Major
gameoflife-core/build/reports/tests/allclasses-frame.html
Add "lang" and/or "xml:lang" attributes to this "<html>" element
Intentionality
Reliability
accessibility wcag2-a
Open
Not assigned
L1 · 2min effort · 4 years ago · Bug · Major
Add "<th>" headers to this "<table>".
Intentionality


Bulk Change
Select issues
Navigate to issue
268 issues
2d 5h effort
gameoflife-acceptance-tests/Dockerfile
Use a specific version tag for the image.
Intentionality
Maintainability
No tags
Open
Not assigned
L1 · 5min effort · 4 years ago · Code Smell · Major
Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.
Intentionality
Maintainability
No tags
Open
Not assigned
L12 · 5min effort · 4 years ago · Code Smell · Major
Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.
Intentionality
Maintainability
No tags
Open
Not assigned
L12 · 5min effort · 4 years ago · Code Smell · Major
Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.
Intentionality
Maintainability
No tags



Conclusion:

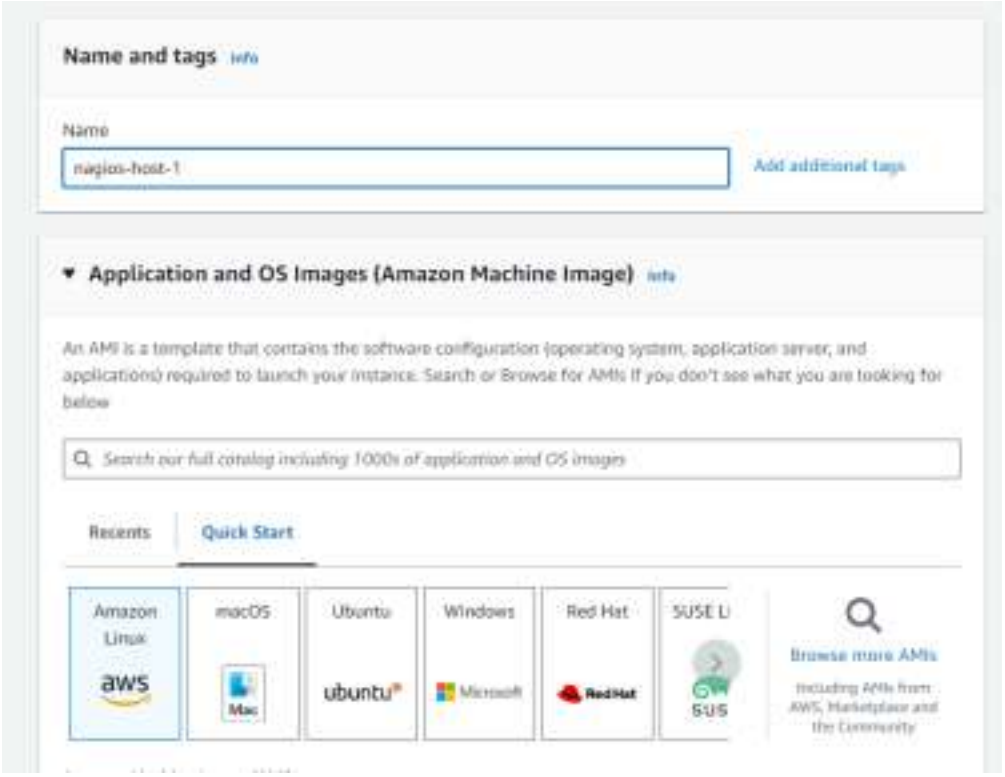
In this experiment we created a Jenkins CICD Pipeline to perform a static analysis of the code to detect bugs, code smells, and security vulnerabilities on a sample code. It is to be checked whether the sonar scanner plugin is installed in Jenkins or not and also provide the correct path and credentials in the pipeline script or else it leads to the failure of the build.

Experiment 9

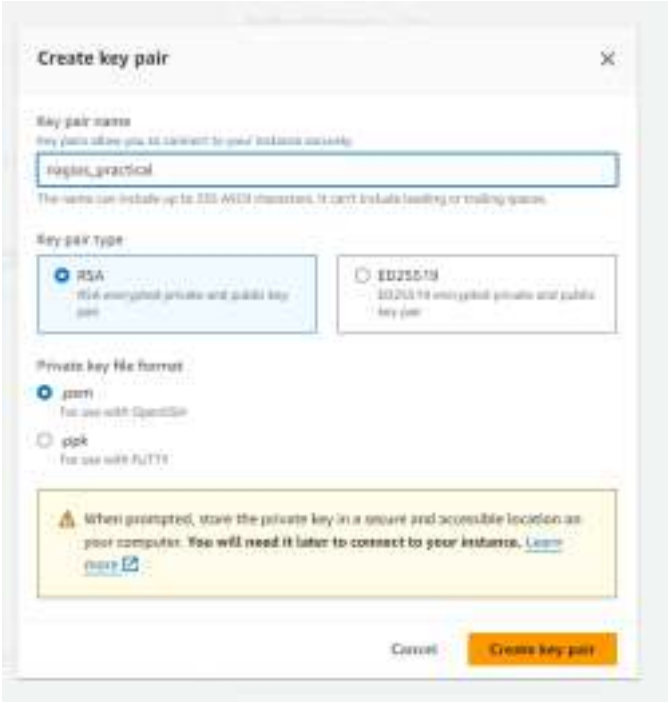
Aim: To understand continuous monitoring and installation and configuration of Nagios Core, Nagios Plugins and NRPE (Nagios Remote Plugin Executor) on Linux Machine.

Steps:

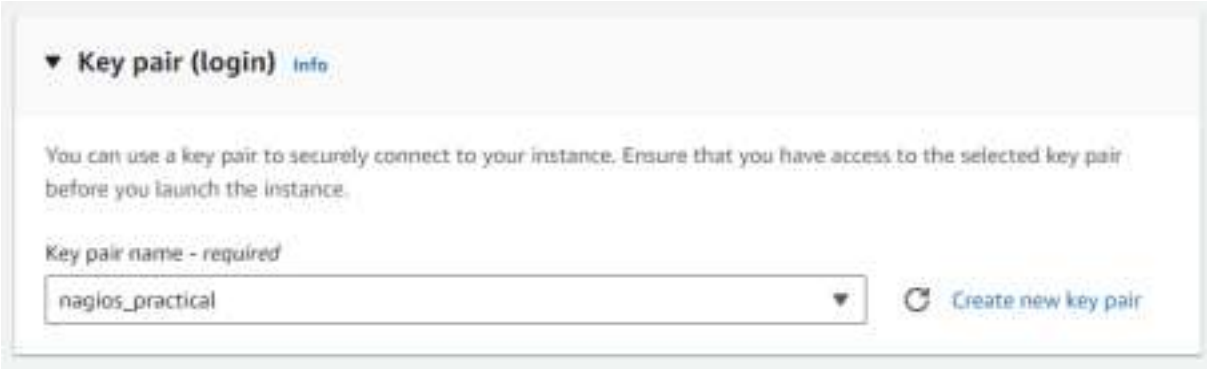
1. Create an ec2 instance and select amazon Linux as the OS



2. Now we will create a key pair.

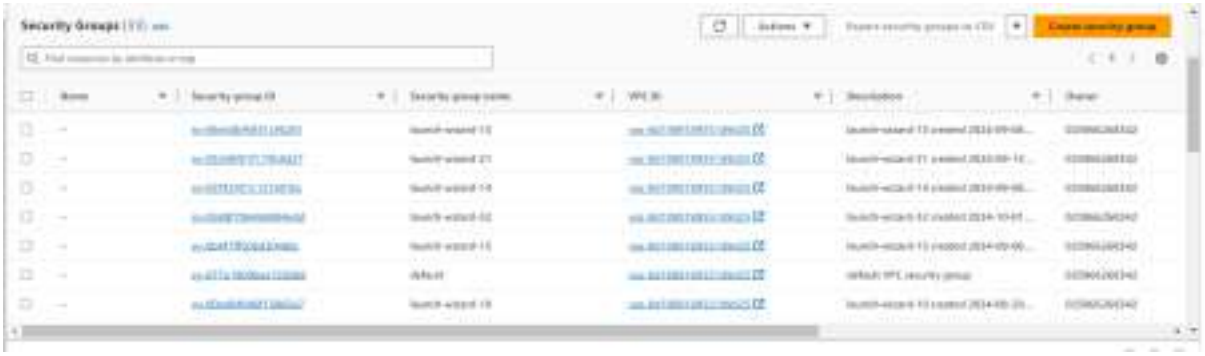


3. Use the created key pair while creating the instance



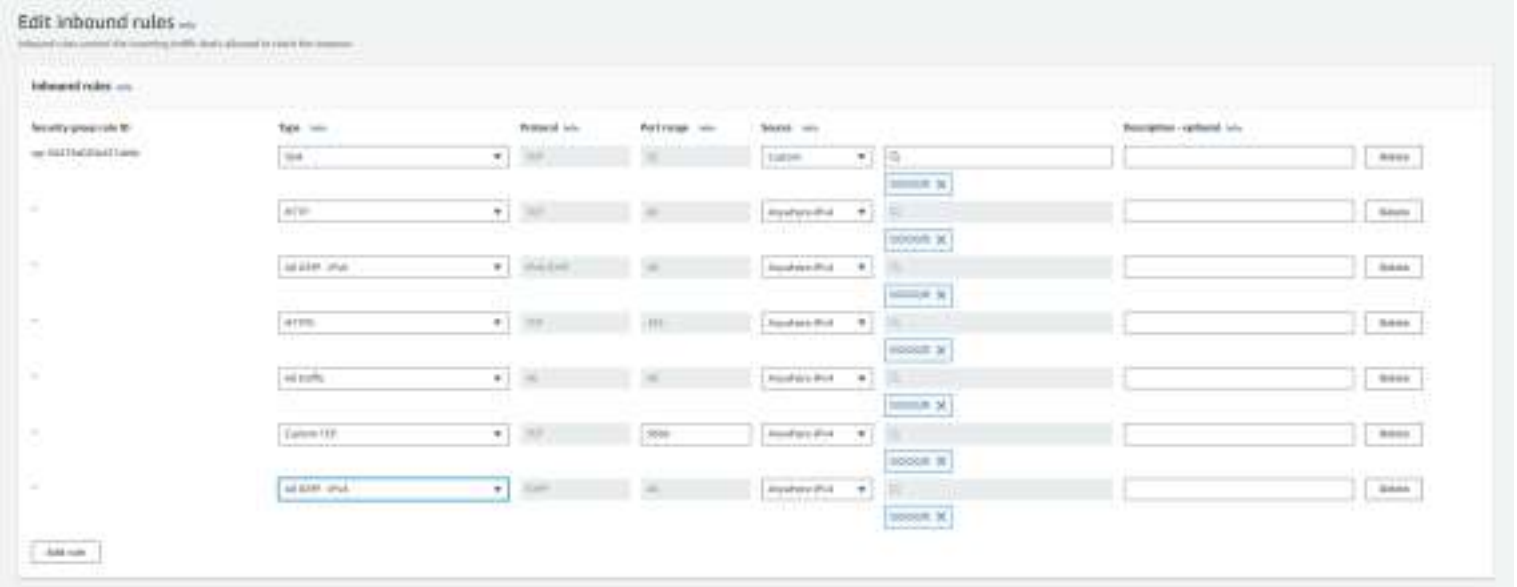
4. Once the instance is successfully initiated go to security groups and select the security group id of the instance you just created.

   (The security group name is visible during instance creation and also on the ec2 instances dashboard)



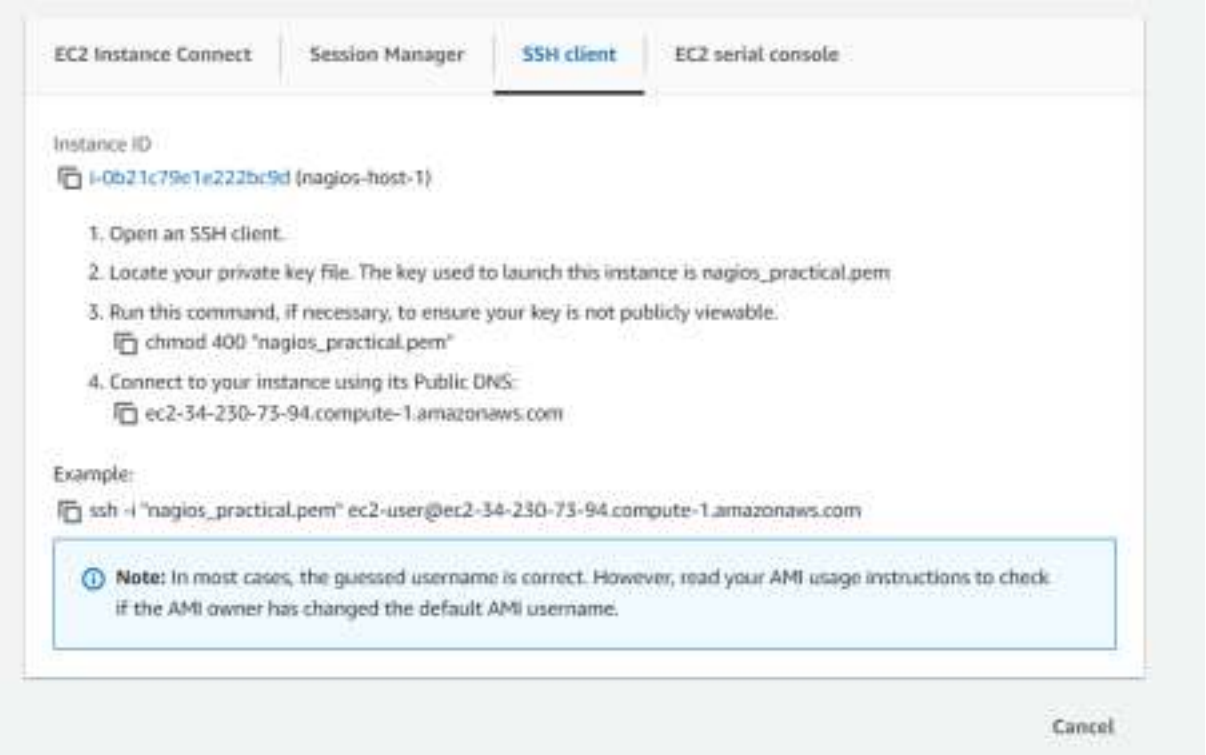
5. To edit the inbound rules select the “Edit inbound rules” button

6. Add the rules as given in the screenshot below



7. Connect the instance

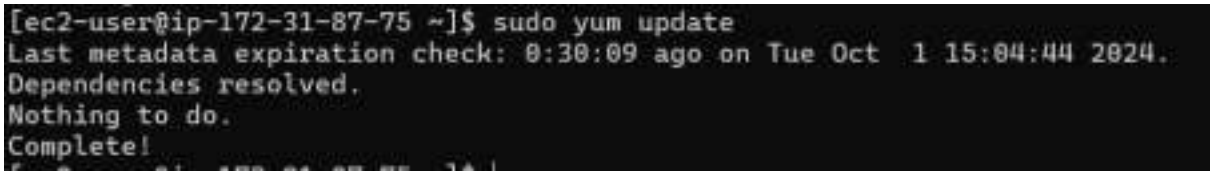
8. Copy the ssh command given in the ssh client section.



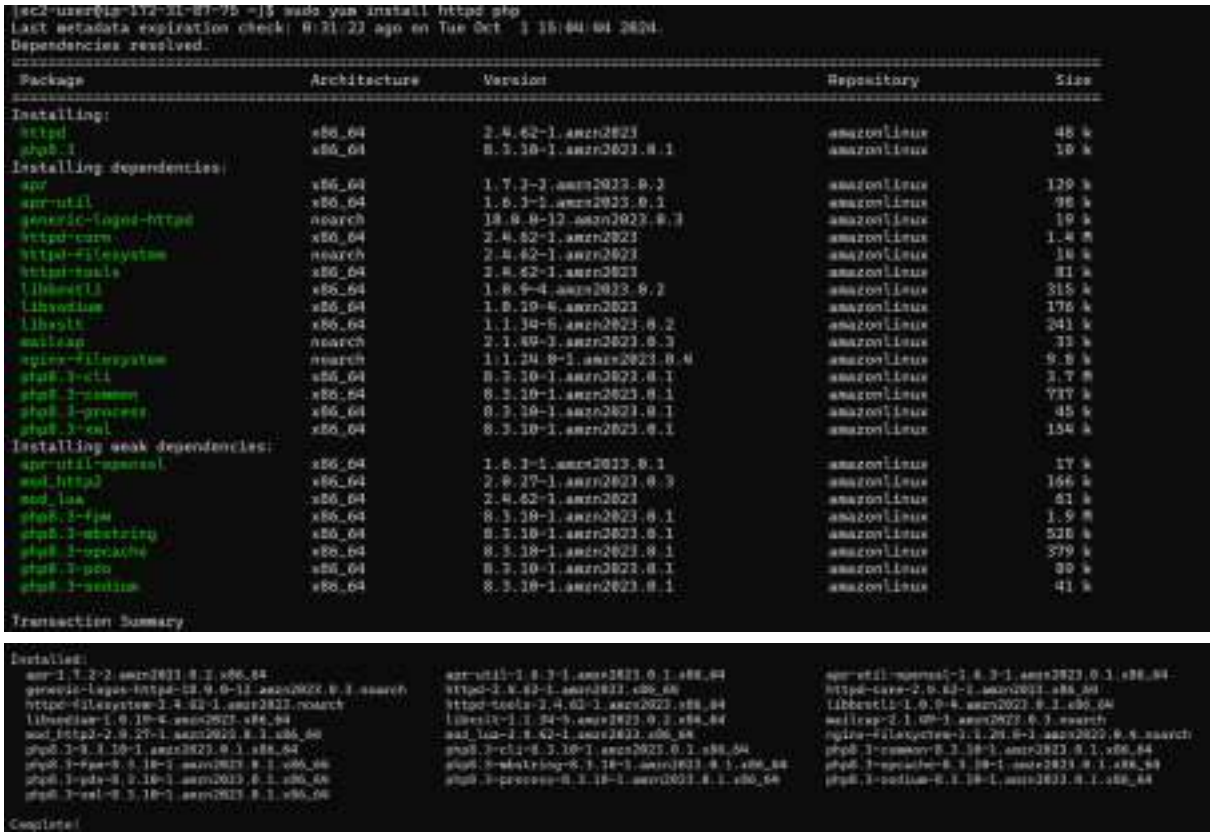
9. In your terminal paste the copied command, just replace the .pem file name with the actual location where the .pem file is downloaded in your system

10. Now install the following packages using yum:

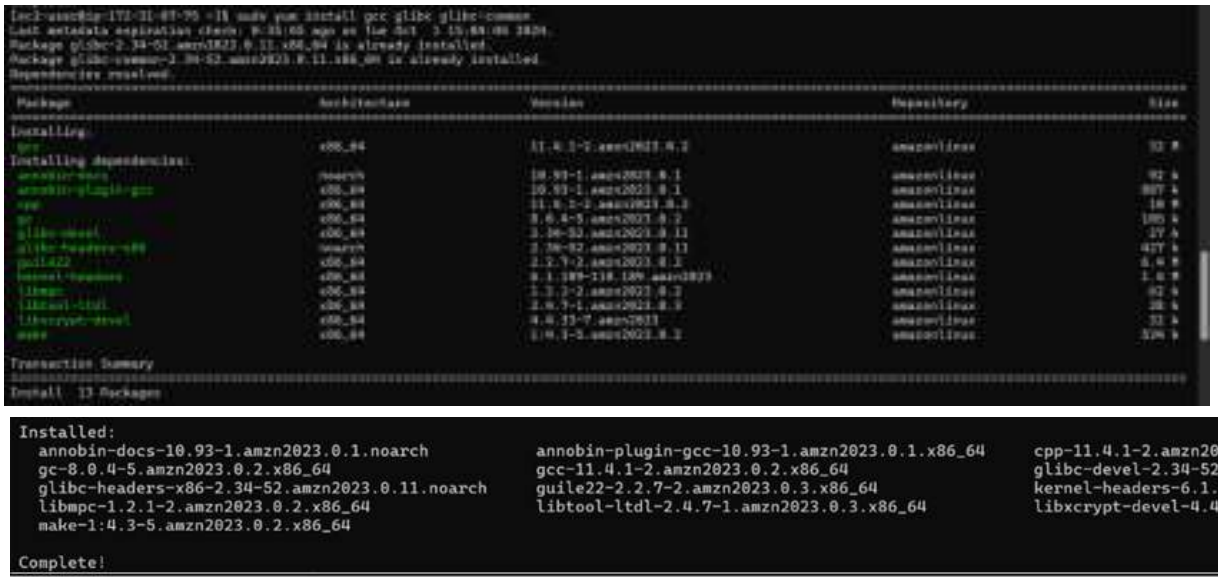
sudo yum update



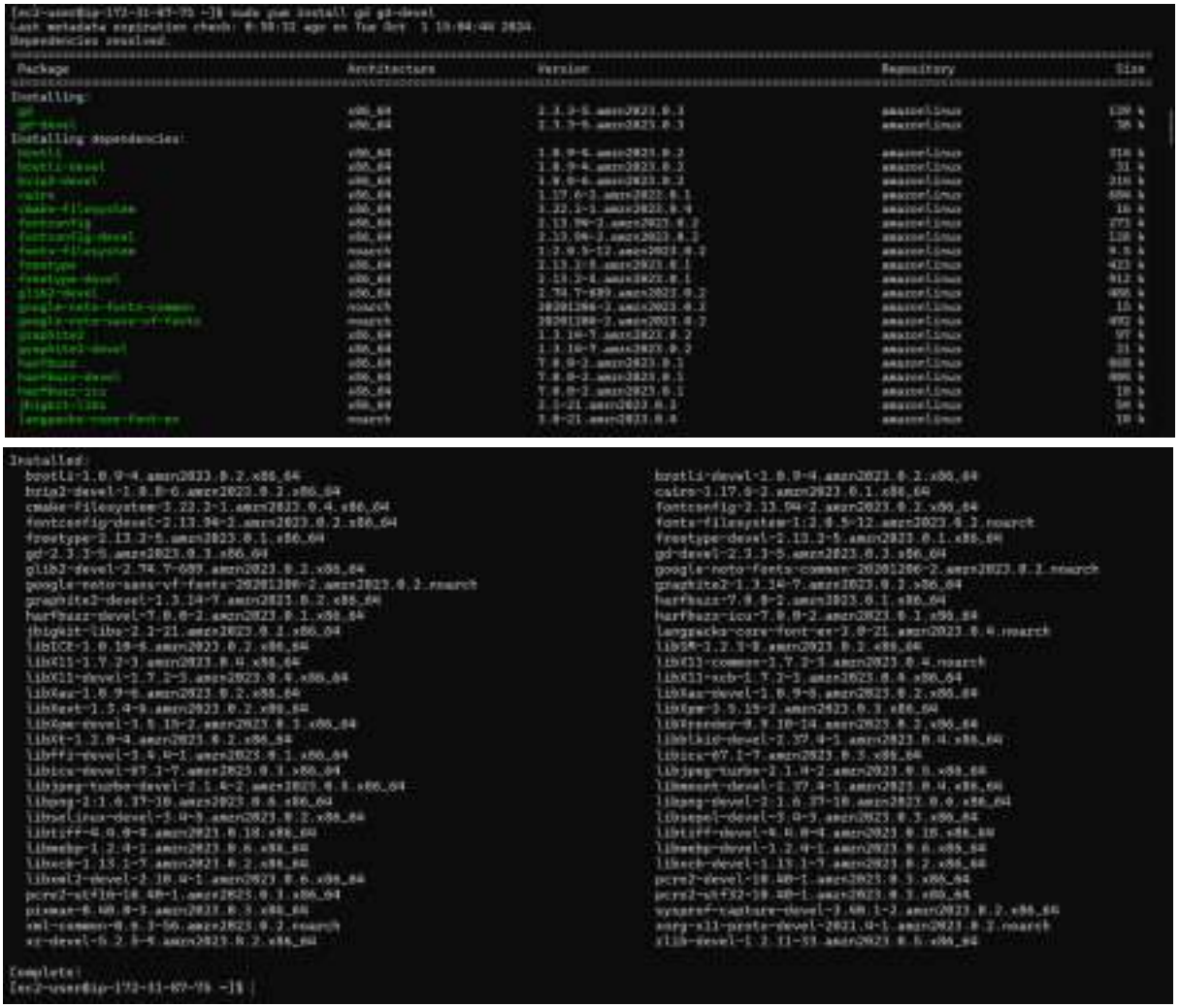
sudo yum install httpd php



sudo yum install gcc glibc glibc-common



sudo yum install gd gd-devel



11. Create a new Nagios User with its password using the below given commands.

sudo adduser -m nagios
sudo passwd nagios

```
[ec2-user@ip-172-31-87-75 ~]$ sudo adduser -m nagios
sudo passwd nagios
Changing password for user nagios.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[ec2-user@ip-172-31-87-75 ~]$
```

12. Create a new user group

sudo groupadd nagcmd

```
[ec2-user@ip-172-31-87-75 ~]$ sudo groupadd nagcmd
[ec2-user@ip-172-31-87-75 ~]$
```

13. Next execute these commands so that you don't have to use sudo for Apache and Nagios:

sudo usermod -a -G nagcmd nagios

sudo usermod -a -G nagcmd apache

```
[ec2-user@ip-172-31-87-75 ~]$ sudo usermod -a -G nagcmd nagios
sudo usermod -a -G nagcmd apache
[ec2-user@ip-172-31-87-75 ~]$
```

14. Create a new directory for Nagios downloads

mkdir ~/downloads
cd ~/downloads

```
[ec2-user@ip-172-31-87-75 ~]$ mkdir ~/downloads
cd ~/downloads
[ec2-user@ip-172-31-87-75 downloads]$
```

15. Use wget to download the source zip files.

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

```
[ec2-user@ip-172-31-87-75 downloads]$ wget https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.5.tar.gz
--2024-10-01 15:55:47--  https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.5.tar.gz
Resolving assets.nagios.com (assets.nagios.com)... 45.79.49.120, 2600:3c00::f03c:92ff:fef7:45ce
Connecting to assets.nagios.com (assets.nagios.com)|45.79.49.120|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2065473 (2.0M) [application/x-gzip]
Saving to: 'nagios-4.5.5.tar.gz'

nagios-4.5.5.tar.gz                 100%[=============================================================================>]   1.97M  5.54MB/s    in 0.4s

2024-10-01 15:55:47 (5.54 MB/s) - 'nagios-4.5.5.tar.gz' saved [2065473/2065473]

[ec2-user@ip-172-31-87-75 downloads]$
```

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

```
[ec2-user@ip-172-31-87-75 downloads]$ wget https://nagios-plugins.org/download/nagios-plugins-2.4.11.tar.gz
--2024-10-01 15:57:19--  https://nagios-plugins.org/download/nagios-plugins-2.4.11.tar.gz
Resolving nagios-plugins.org (nagios-plugins.org)... 45.56.123.251
Connecting to nagios-plugins.org (nagios-plugins.org)|45.56.123.251|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2753049 (2.6M) [application/x-gzip]
Saving to: 'nagios-plugins-2.4.11.tar.gz'

nagios-plugins-2.4.11.tar.gz          100%[=============================================================================>]   2.62M  4.14MB/s    in 0.6s

2024-10-01 15:57:20 (4.14 MB/s) - 'nagios-plugins-2.4.11.tar.gz' saved [2753049/2753049]
```

16. Use tar to unzip and change to that directory.

tar zxvf nagios-4.5.5.tar.gz

```
[ec2-user@ip-172-31-87-75 downloads]$ tar zxvf nagios-4.5.5.tar.gz
nagios-4.5.5/
nagios-4.5.5/.github/
nagios-4.5.5/.github/workflows/
nagios-4.5.5/.github/workflows/test.yml
nagios-4.5.5/.gitignore
nagios-4.5.5/CONTRIBUTING.md
nagios-4.5.5/Changelog
nagios-4.5.5/INSTALLING
nagios-4.5.5/LEGAL
nagios-4.5.5/LICENSE
nagios-4.5.5/Makefile.in
nagios-4.5.5/README.md
nagios-4.5.5/THANKS
nagios-4.5.5/UPGRADING
nagios-4.5.5/aclocal.m4
nagios-4.5.5/autoconf-macros/
nagios-4.5.5/autoconf-macros/.gitignore
nagios-4.5.5/autoconf-macros/CHANGELOG.md
nagios-4.5.5/autoconf-macros/LICENSE
nagios-4.5.5/autoconf-macros/LICENSE.md
nagios-4.5.5/autoconf-macros/README.md
nagios-4.5.5/autoconf-macros/add_group_user
nagios-4.5.5/autoconf-macros/ax_nagios_get_distrib
nagios-4.5.5/autoconf-macros/ax_nagios_get_files
```

17. We have to now change the directory to nagios-4.5.5, for this first verify whether nagios-4.5.5 exists by using ls command.

```
[ec2-user@ip-172-31-87-75 downloads]$ ls
nagios-4.5.5  nagios-4.5.5.tar.gz  nagios-plugins-2.4.11.tar.gz
```

18. As nagios-4.5.5 is present we will now use cd command to change directory.

```
[ec2-user@ip-172-31-87-75 downloads]$ cd nagios-4.5.5
[ec2-user@ip-172-31-87-75 nagios-4.5.5]$
```

19. Now we will install openssl dev library by using the command:
sudo yum install openssl-devel

```
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          23 MB/s | 3.0 MB     00:00
--------------------------------------------------------------------------------
Total                                                   18 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
```

20. Run the configuration script

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

```
[ec2-user@ip-172-31-87-75 nagios-4.5.5]$ ./configure --with-command-group=nagcmd
checking for a BSD-compatible install... /usr/bin/install -c
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether the compiler supports GNU C... yes
checking whether gcc accepts -g... yes
checking for gcc option to enable C11 features... none needed
checking whether make sets $(MAKE)... yes
checking whether ln -s works... yes
checking for strip... /usr/bin/strip
checking for sys/wait.h that is POSIX.1 compatible... yes
checking for stdio.h... yes
checking for stdlib.h... yes
checking for string.h... yes
```

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

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

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


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

21. To compile the source code run “make all”

```
[ec2-user@ip-172-31-87-75 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
```

22. To install binaries, init script and sample config files run

    sudo make install

    sudo make install-init

    sudo make install-config

    sudo make install-commandmode

```
[ec2-user@ip-172-31-87-75 nagios-4.5.5]$ sudo make install
sudo make install-init
sudo make install-config
sudo make install-commandmode
cd ./base && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/base'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagios /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagiostats /usr/local/nagios/bin
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5/base'
cd ./cgi && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
make install-basic
make[2]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/sbin
for file in *.cgi; do \
        /usr/bin/install -c -s -m 775 -o nagios -g nagios $file /usr/local/nagios/sbin; \
done
make[2]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
cd ./html && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/html'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/media
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/stylesheets
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/contexthelp
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/docs
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/docs/images
/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
```

23. In the config file edit the email address

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

```
##############################################################################
#
# CONTACTS
#
##############################################################################

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

define contact {

    contact_name            nagiosadmin             ; Short name of user
    use                     generic-contact         ; Inherit default values from generic-contact template (defined abov
    alias                   Nagios Admin            ; Full name of user
    email                   d2022.nayaab.jindani@ves.ac.in ; <****** CHANGE THIS TO YOUR EMAIL ADDRESS ******
}
```

24. To configure the web interface run:
    sudo make install-webconf

```
[ec2-user@ip-172-31-87-75 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.con
fi

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

[ec2-user@ip-172-31-87-75 nagios-4.5.5]$
```

25. Create a nagios admin account and password
    sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin

```
[ec2-user@ip-172-31-87-75 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
```

26. Restart apache
    sudo service httpd restart

```
[ec2-user@ip-172-31-87-75 nagios-4.5.5]$ sudo service httpd restart
Redirecting to /bin/systemctl restart httpd.service
[ec2-user@ip-172-31-87-75 nagios-4.5.5]$
```

27. Go back to the downloads folder by using “cd ~/downloads” and unzip the plugins zip file using

    tar zxvf nagios-plugins-2.4.11.tar.gz

```
[ec2-user@ip-172-31-87-75 nagios-4.5.5]$ cd ~/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
nagios-plugins-2.4.11/config_test/
nagios-plugins-2.4.11/config_test/Makefile
nagios-plugins-2.4.11/config_test/run_tests
nagios-plugins-2.4.11/config_test/child_test.c
nagios-plugins-2.4.11/gl/
```

28. Compile and install the plugins

    cd nagios-plugins-2.0.3

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

```
[ec2-user@ip-172-31-87-75 downloads]$ cd nagios-plugins-2.4.11
./configure --with-nagios-user=nagios --with-nagios-group=nagios
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /usr/bin/mkdir -p
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking whether make supports nested variables... yes
checking whether to enable maintainer-specific portions of Makefiles
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
```

make

sudo make install

```
[ec2-user@ip-172-31-87-75 nagios-plugins-2.4.11]$ sudo make install
Making install in gl
make[1]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11/gl'
make  install-recursive
make[2]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11/gl'
make[3]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11/gl'
make[4]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11/gl'
if test yes = no; then \
  case 'linux-gnu' in \
    darwin[56]*) \
      need_charset_alias=true ;; \
    darwin* | cygwin* | mingw* | pw32* | cegcc*) \
      need_charset_alias=false ;; \
    *) \
      need_charset_alias=true ;; \
  esac ; \
else \
  need_charset_alias=false ; \
fi ; \
if $need_charset_alias; then \
  /bin/sh ../build-aux/mkinstalldirs /usr/local/nagios/lib ; \
fi ; \
if test -f /usr/local/nagios/lib/charset.alias; then \
  sed -f ref-add.sed /usr/local/nagios/lib/charset.alias > /usr/local/nagios/lib/charset.tmp ; \
  /usr/bin/install -c -o nagios -g nagios -m 644 /usr/local/nagios/lib/charset.tmp /usr/local/nagios/lib/charset.alias ; \
  rm -f /usr/local/nagios/lib/charset.tmp ; \
else \
  if $need_charset_alias; then \
    sed -f ref-add.sed charset.alias > /usr/local/nagios/lib/charset.tmp ; \
    /usr/bin/install -c -o nagios -g nagios -m 644 /usr/local/nagios/lib/charset.tmp /usr/local/nagios/lib/charset.alias ; \
```

29. Run below given commands to start nagios:
    sudo chkconfig --add nagios
    sudo chkconfig nagios on

```
[ec2-user@ip-172-31-87-75 nagios-plugins-2.4.11]$ sudo chkconfig --add nagios
sudo chkconfig nagios on
error reading information on service nagios: No such file or directory
Note: Forwarding request to 'systemctl enable nagios.service'.
Created symlink /etc/systemd/system/multi-user.target.wants/nagios.service → /usr/lib/systemd/system/nagios.service.
```

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

```
[ec2-user@ip-172-31-87-75 nagios-plugins-2.4.11]$ sudo /usr/local/nagios/bin/nagios -v /usr/local/nagi

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

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

Running pre-flight check on configuration data...

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

Total Warnings: 0
Total Errors:   0

Things look okay - No serious problems were detected during the pre-flight check
[ec2-user@ip-172-31-87-75 nagios-plugins-2.4.11]$
```

If the message says no errors detected then run “sudo service nagios start”

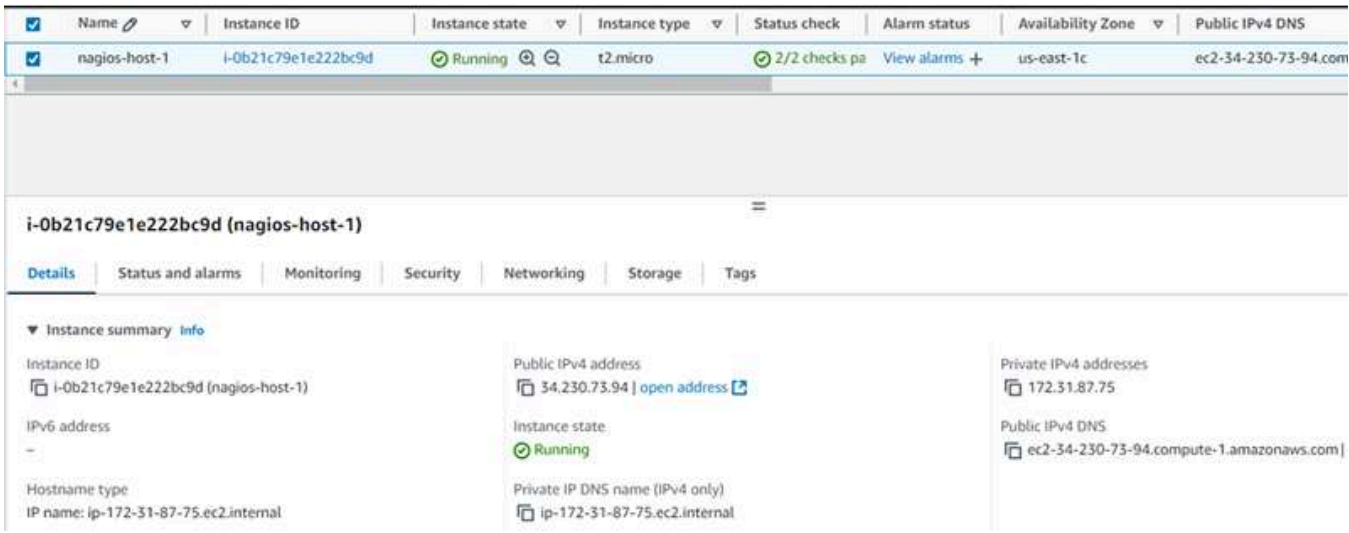
```
[ec2-user@ip-172-31-87-75 nagios-plugins-2.4.11]$ sudo service nagios s
Redirecting to /bin/systemctl start nagios.service
[ec2-user@ip-172-31-87-75 nagios-plugins-2.4.11]$
```

sudo systemctl status nagios

```
Checking global event handlers...
Checking obsessive compulsive processor commands...
Checking misc settings...

Total Warnings: 0
Total Errors:   0

Things look okay - No serious problems were detected during the pre-flight check
[ec2-user@ip-172-31-87-75 nagios-plugins-2.4.11]$ sudo service nagios start
Redirecting to /bin/systemctl start nagios.service
[ec2-user@ip-172-31-87-75 nagios-plugins-2.4.11]$ sudo systemctl status nagios
● nagios.service - Nagios Core 4.5.5
     Loaded: loaded (/usr/lib/systemd/system/nagios.service; enabled; preset: disabled)
     Active: active (running) since Tue 2024-10-01 16:32:27 UTC; 48s ago
       Docs: https://www.nagios.org/documentation
    Process: 66684 ExecStartPre=/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/
    Process: 66693 ExecStart=/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUC
   Main PID: 66694 (nagios)
      Tasks: 6 (limit: 1112)
     Memory: 5.7M
        CPU: 82ms
     CGroup: /system.slice/nagios.service
             ├─66694 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─66695 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─66696 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─66697 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─66698 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             └─66739 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg

Oct 01 16:32:27 ip-172-31-87-75.ec2.internal nagios[66694]: qh: Socket '/usr/local/nagios/var/rw/nagios.qh' successfully
Oct 01 16:32:27 ip-172-31-87-75.ec2.internal nagios[66694]: qh: core query handler registered
Oct 01 16:32:27 ip-172-31-87-75.ec2.internal nagios[66694]: qh: echo service query handler registered
Oct 01 16:32:27 ip-172-31-87-75.ec2.internal nagios[66694]: qh: help for the query handler registered
Oct 01 16:32:27 ip-172-31-87-75.ec2.internal nagios[66694]: wproc: Successfully registered manager as @wproc with query
Oct 01 16:32:27 ip-172-31-87-75.ec2.internal nagios[66694]: wproc: Registry request: name=Core Worker 66696;pid=66696
Oct 01 16:32:27 ip-172-31-87-75.ec2.internal nagios[66694]: wproc: Registry request: name=Core Worker 66695;pid=66695
Oct 01 16:32:27 ip-172-31-87-75.ec2.internal nagios[66694]: wproc: Registry request: name=Core Worker 66697;pid=66697
Oct 01 16:32:27 ip-172-31-87-75.ec2.internal nagios[66694]: wproc: Registry request: name=Core Worker 66698;pid=66698
Oct 01 16:32:27 ip-172-31-87-75.ec2.internal nagios[66694]: Successfully launched command file worker with pid 66739
[ec2-user@ip-172-31-87-75 nagios-plugins-2.4.11]$
```

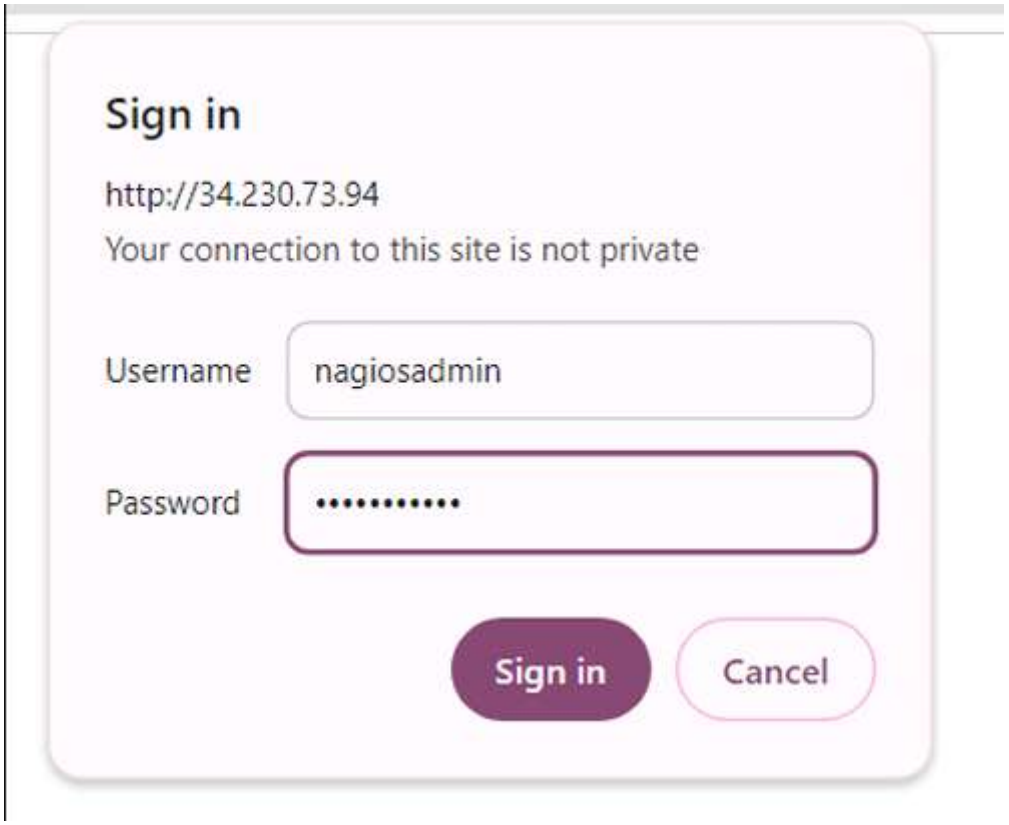
We can see that Nagios has been initialized correctly and its status is active.

30. Go back to your instances and copy the public IPv4 address

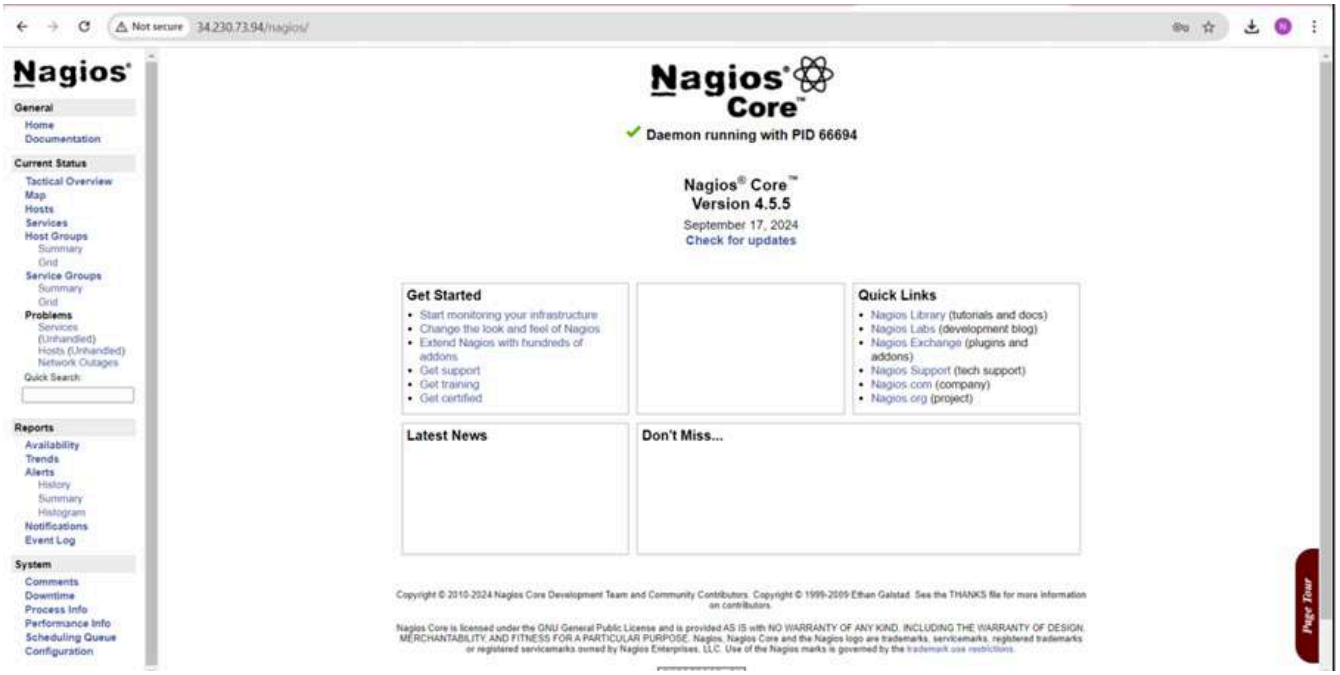


31. Lastly, go to your web browser and type “http://<public-IPv4-address>/nagios” Replace public-IPv4-address with the public ip address of your instance which you copied.

You will get a prompt to enter the username and password that have been set for nagios admin in step 25.



You will see the below shown page after entering credentials.



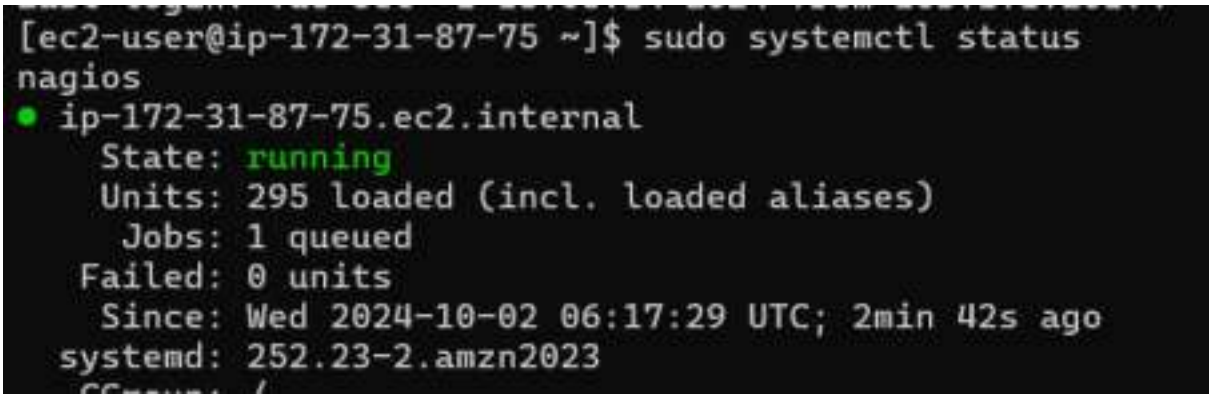
Conclusion:In this experiment, the primary challenge I encountered was accessing the Nagios web interface due to a "Forbidden: You do not have permission to access this resource" error. This issue was resolved by modifying the inbound security rules and verifying that all necessary files were installed in the correct directories. Additionally, it is crucial to restart Apache after making any changes to ensure they take effect. Once Nagios was activated without any errors, the output was successfully displayed on the web interface.

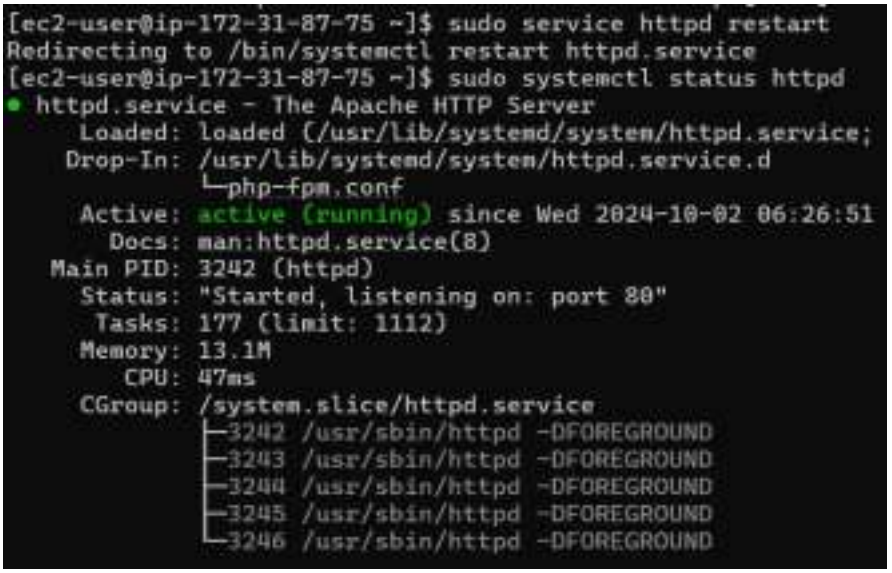
Experiment 10

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

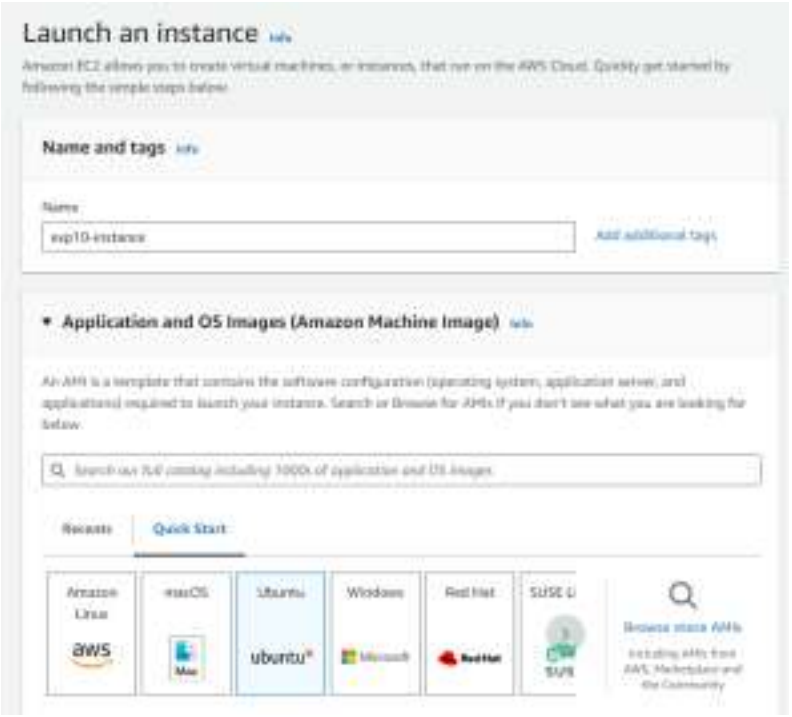
Nagios. Steps:

1. Firstly, we will check whether nagios is running on the server side by using the command “sudo systemctl status nagios” on the host machine (host machine is the instance connected to the terminal in experiment 9, ensure that you have started the instance created for exp9, also check status of apache).

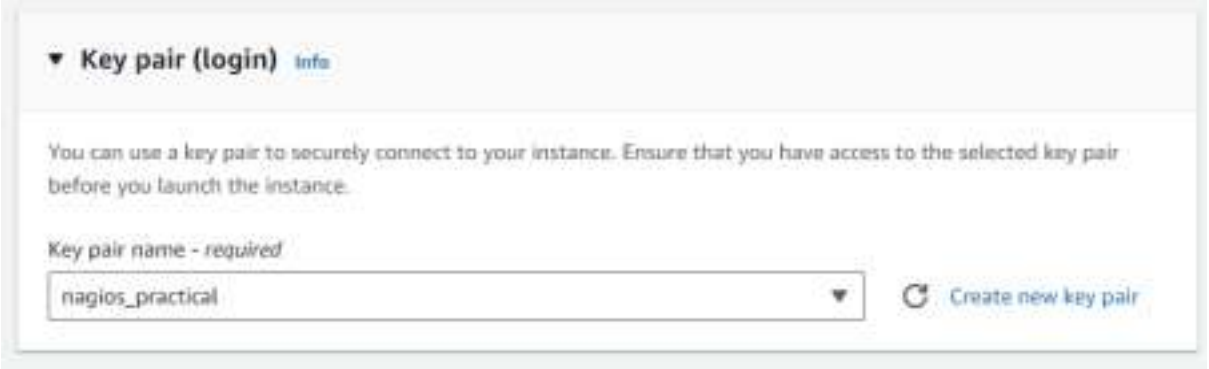




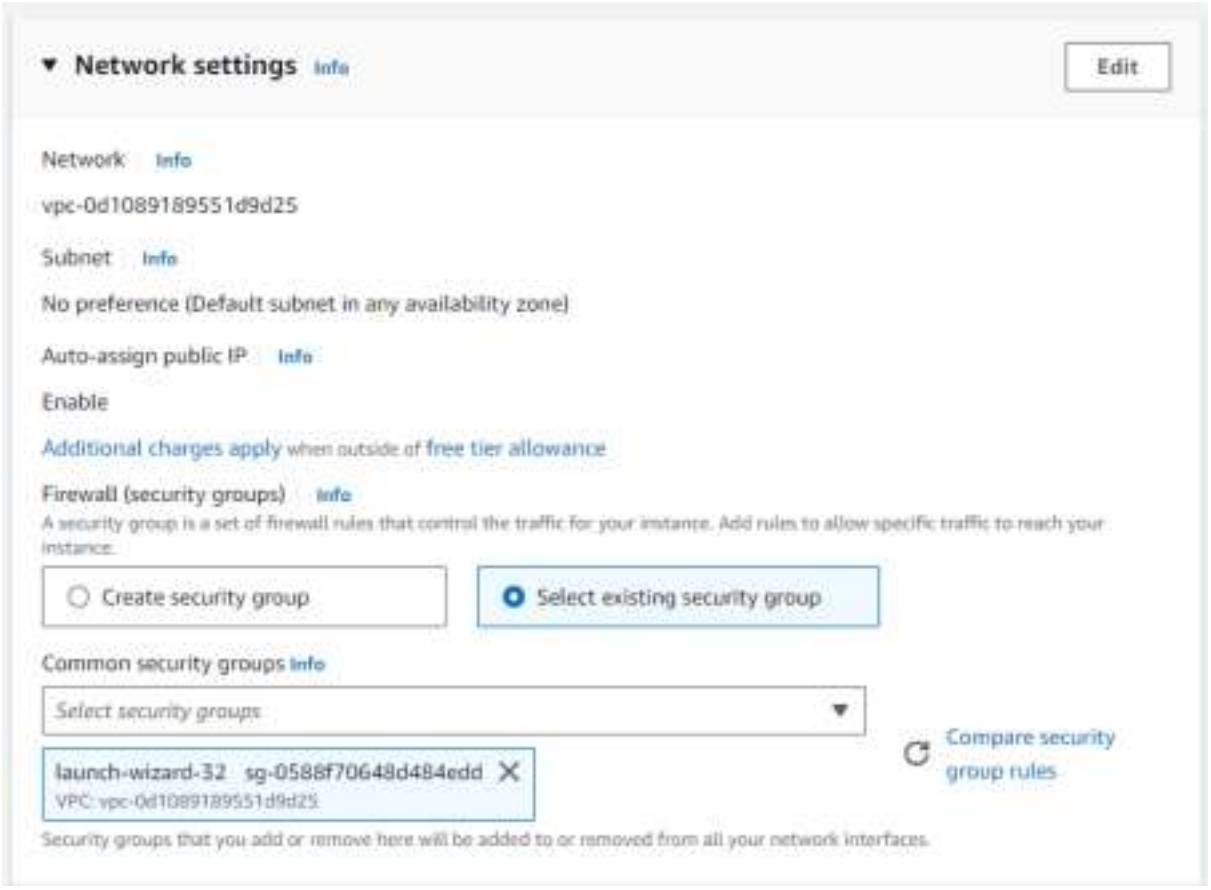
2. Now we will launch a new instance. Select ubuntu for the OS.



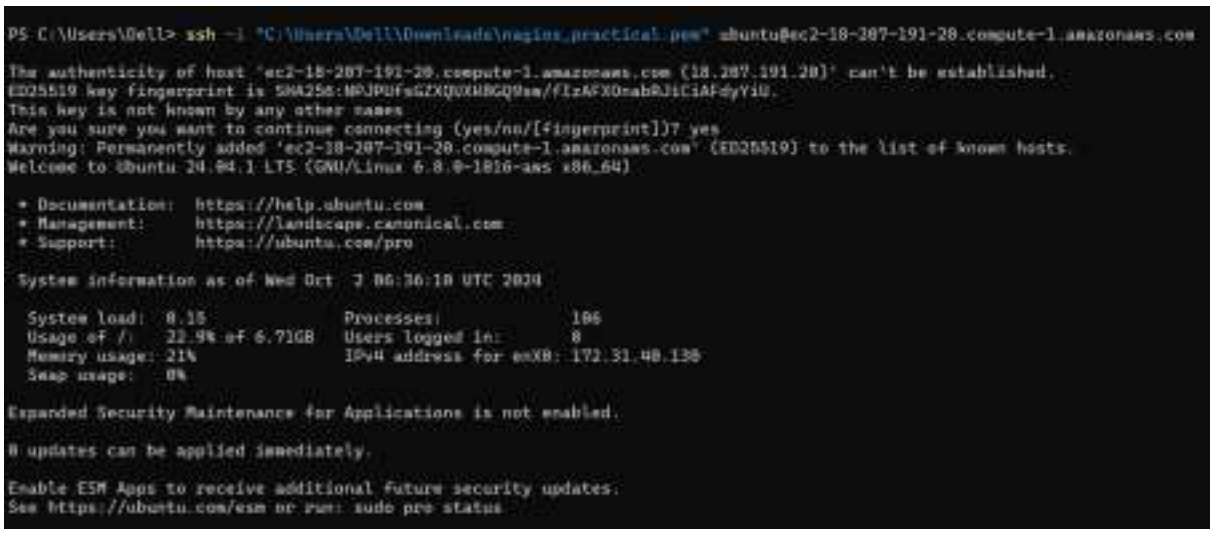
3. Select the key pair which was created and used in the exp 9.



4. Select existing security group and from the list of options select the security group created for exp 9. Previously it was launch wizard 32 and so here I have selected the same.



5. Open a new terminal to connect to the client machine. Copy the SSH command provided in the SSH client section during connection of instance. When pasting the command into your terminal, ensure you specify the full path to your .pem file instead of just the file name.



6. Now go back to your host machine and run the following command

ps -ef | grep nagios

```
[ec2-user@ip-172-31-87-75 ~]$ ps -ef | grep nagios
nagios    2002    1  0 06:17 ?        00:00:00 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
nagios    2003 2002  0 06:17 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios    2004 2002  0 06:17 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios    2005 2002  0 06:17 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios    2006 2002  0 06:17 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios    2007 2002  0 06:17 ?        00:00:00 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
ec2-user  4580 2306  0 06:43 pts/0    00:00:00 grep --color=auto nagios
[ec2-user@ip-172-31-87-75 ~]$
```

7. Now perform these commands on the host terminal
sudo su
mkdir -p /usr/local/nagios/etc/objects/monitorhosts/linuxhosts

```
[root@ip-172-31-87-75 ec2-user]# mkdir -p /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
[root@ip-172-31-87-75 ec2-user]#
```

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

```
[root@ip-172-31-87-75 ec2-user]# cp /usr/local/nagios/etc/objects/localhost.cfg /usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linux
[root@ip-172-31-87-75 ec2-user]#
```

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

The above given command will open the nano text editor wherein you have to do the following changes:

i. Change the hostgroup name to linux-servers1

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

# Define an optional hostgroup for Linux machines

define hostgroup {

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

ii. Change host name and alias from localhost to linuxserver everywhere in the file

```
# Define a service to "ping" the local machine

define service {

    use                     local-service           ; Name of service template to use
    host_name               linuxserver
    service_description     PING
    check_command           check_ping!100.0,20%!500.0,60%
}
```

iii. Change the address to the public IPv4 address of the ubuntu instance (You will find the ip address when you select the instance on the ec2 instances dashboard)

```
# Define a host for the local machine

define host {

    use                     linux-server            ; Name of host template to use
                                                    ; This host definition will inherit all variables that are defined
                                                    ; in (or inherited by) the linux-server host template definition.
    host_name               linuxserver
    alias                   linuxserver
    address                 10.207.191.20
}
```

8. Open the Nagios Config file by using this command:
   nano /usr/local/nagios/etc/nagios.cfg
   nano text editor will get opened

```
##############################################################################
#
# NAGIOS.CFG - Sample Main Config File for Nagios 4.5.5
#
# Read the documentation for more information on this configuration
# file.  I've provided some comments here, but things may not be so
# clear without further explanation.
#
#
##############################################################################


# LOG FILE
# This is the main log file where service and host events are logged
# for historical purposes.  This should be the first option specified
# in the config file!!!

log_file=/usr/local/nagios/var/nagios.log



# 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

# Definitions for monitoring a Windows machine
```

9. In the text editor add “cfg_dir=/usr/local/nagios/etc/objects/monitorhosts/” this line

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

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

10. Now we will verify the configuration files

    /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-172-31-87-75 ec2-user]#
```

If there are no errors we can proceed further

11. We will now restart the nagios service
    service nagios restart

```
[root@ip-172-31-87-75 ec2-user]# service nagios restart
Redirecting to /bin/systemctl restart nagios.service
[root@ip-172-31-87-75 ec2-user]#
```

12. Now on the client machine (The ubuntu machine we created for this experiment) run the following command:

sudo apt update -y

```
ubuntu@ip-172-31-40-130:~$ 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://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:7 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [380 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Components [35.0 kB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 c-n-f Metadata [8328 B]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [535 kB]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [130 kB]
Get:16 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 c-n-f Metadata [8676 B]
Get:17 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [380 kB]
Get:18 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [156 kB]
Get:19 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [45.0 kB]
Get:20 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 c-n-f Metadata [14.9 kB]
Get:21 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Packages [14.4 kB]
Get:22 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse Translation-en [3608 B]
Get:23 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [212 B]
Get:24 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 c-n-f Metadata [532 B]
```

sudo apt install gcc -y

```
ubuntu@ip-172-31-40-130:~$ 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 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
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.
```

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

```
ubuntu@ip-172-31-40-130:~$ 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 python3-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]
```

13. Open nrpe.cfg file to make changes.
    sudo nano /etc/nagios/nrpe.cfg
    Under allowed_hosts, add your nagios host public IPv4 address:

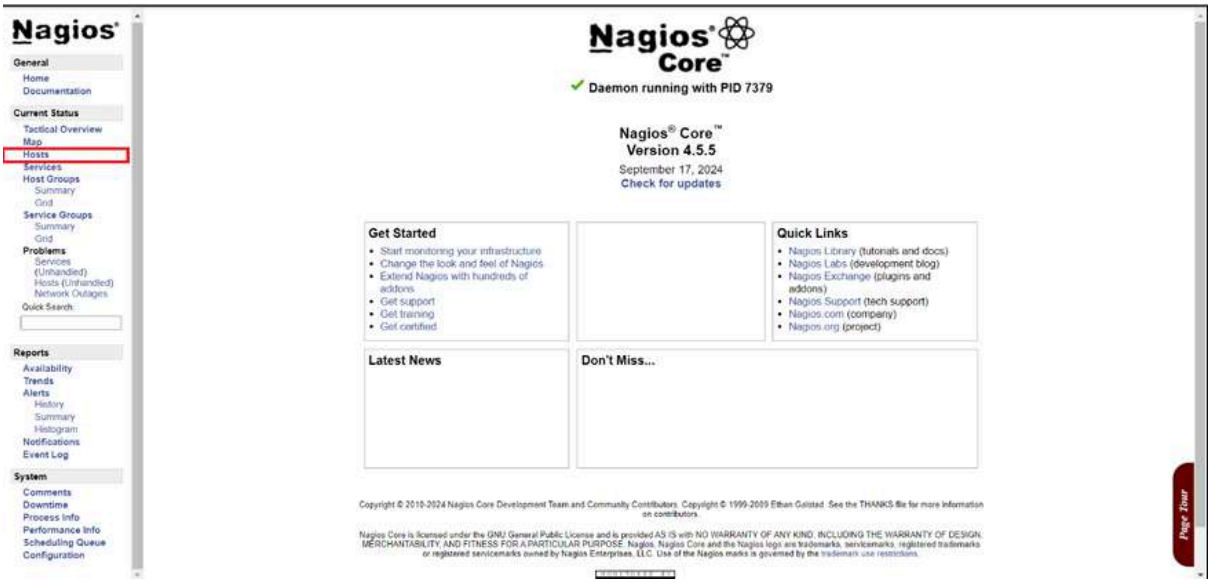
```
#
# Note: The daemon only does rudimentary checking of the client's IP
# address.  I would highly recommend adding entries in your /etc/hosts.allow
# file to allow only the specified host to connect to the port
# you are running this daemon on.
#
# NOTE: This option is ignored if NRPE is running under either inetd or xinetd

allowed_hosts=127.0.0.1,54.163.184.143


# COMMAND ARGUMENT PROCESSING
# This option determines whether or not the NRPE daemon will allow clients
# to specify arguments to commands that are executed.  This option only works
# if the daemon was configured with the --enable-command-args configure script
# option.
#
# *** ENABLING THIS OPTION IS A SECURITY RISK! ***
# Read the SECURITY file for information on some of the security implications
# of enabling this variable.
#
# Values: 0=do not allow arguments, 1=allow command arguments
```

14. Now restart the NRPE server

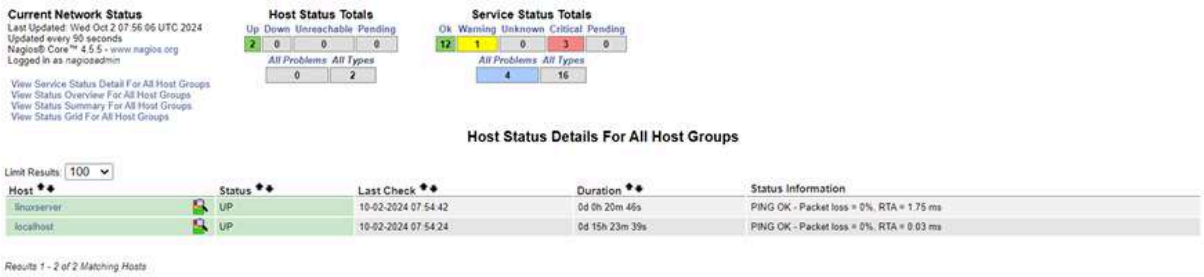
    sudo systemctl restart nagios-nrpe-server

```
ubuntu@ip-172-31-40-130:~$ sudo systemctl restart nagios-nrpe-server
ubuntu@ip-172-31-40-130:~$
```

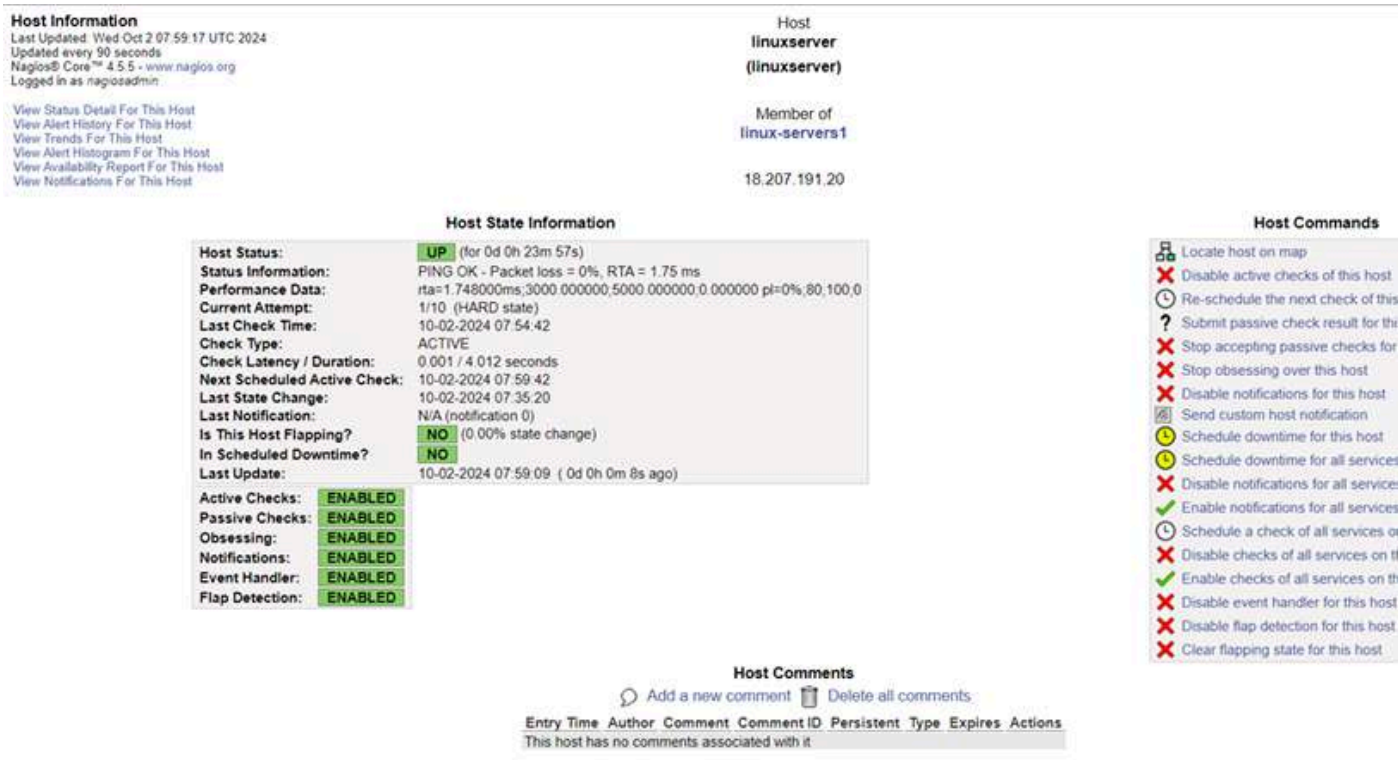
15. Go to the nagios dashboard and click on hosts



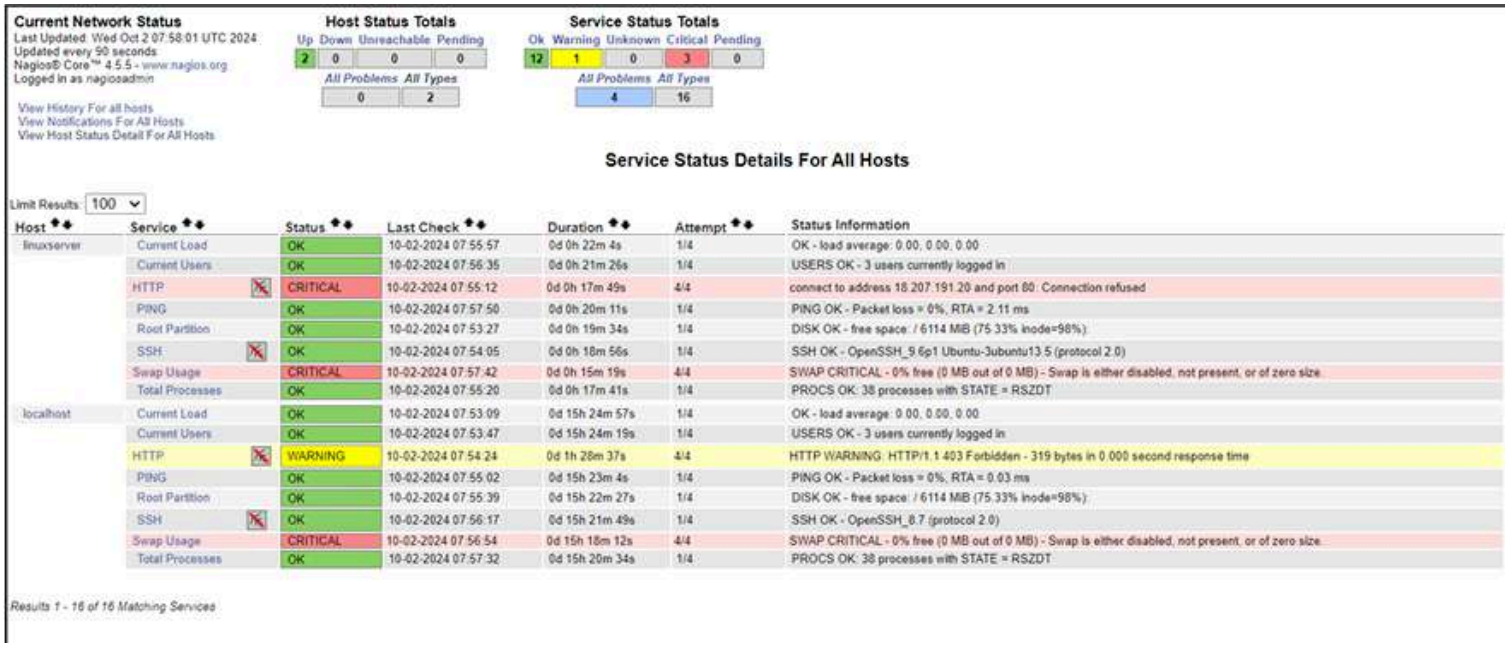
Click on linux server



We can see the host state information:



If you want to see all the services and ports being monitored then select the services option and you will see the page as shown below:



Current Network Status
Last Updated: Wed Oct 2 07:58:01 UTC 2024
Updated every 90 seconds
Nagios® Core™ 4.5.5 - www.nagios.org
Logged in as nagiosadmin

View History For all hosts
View Notifications For All Hosts
View Host Status Detail For All Hosts

Host Status Totals

Up	Down	Unreachable	Pending
2	0	0	0

All Problems	All Types
0	2

Service Status Totals

Ok	Warning	Unknown	Critical	Pending
12	1	0	3	0

All Problems	All Types
4	16

Service Status Details For All Hosts

Limit Results: 100

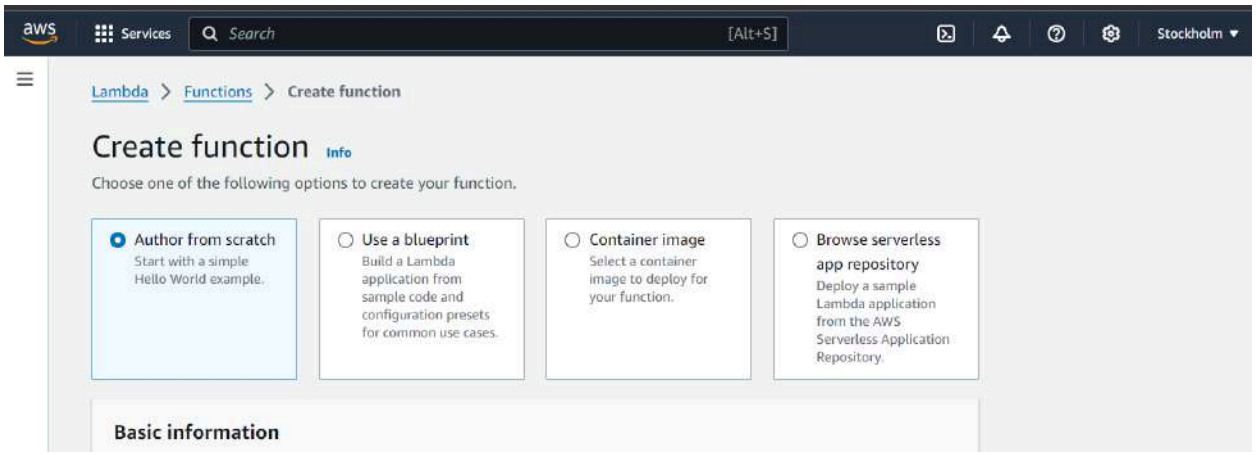
Host	Service	Status	Last Check	Duration	Attempt	Status Information
linuxserver	Current Load	OK	10-02-2024 07:55:57	0d 0h 22m 4s	1/4	OK - load average: 0.00, 0.00, 0.00
	Current Users	OK	10-02-2024 07:56:35	0d 0h 21m 26s	1/4	USERS OK - 3 users currently logged in
	HTTP	CRITICAL	10-02-2024 07:55:12	0d 0h 17m 49s	4/4	connect to address 18.207.191.20 and port 80: Connection refused
	PING	OK	10-02-2024 07:57:50	0d 0h 20m 11s	1/4	PING OK - Packet loss = 0%, RTA = 2.11 ms
	Root Partition	OK	10-02-2024 07:53:27	0d 0h 19m 34s	1/4	DISK OK - free space: / 6114 MiB (75.33% inode=98%):
	SSH	OK	10-02-2024 07:54:05	0d 0h 18m 56s	1/4	SSH OK - OpenSSH_9.6p1 Ubuntu-3ubuntu13.5 (protocol 2.0)
	Swap Usage	CRITICAL	10-02-2024 07:57:42	0d 0h 15m 19s	4/4	SWAP CRITICAL - 0% free (0 MB out of 0 MB) - Swap is either disabled, not present, or of zero size.
	Total Processes	OK	10-02-2024 07:55:20	0d 0h 17m 41s	1/4	PROCS OK: 38 processes with STATE = RSZDT
localhost	Current Load	OK	10-02-2024 07:53:09	0d 15h 24m 57s	1/4	OK - load average: 0.00, 0.00, 0.00
	Current Users	OK	10-02-2024 07:53:47	0d 15h 24m 19s	1/4	USERS OK - 3 users currently logged in
	HTTP	WARNING	10-02-2024 07:54:24	0d 1h 28m 37s	4/4	HTTP WARNING: HTTP/1.1 403 Forbidden - 319 bytes in 0.000 second response time
	PING	OK	10-02-2024 07:55:02	0d 15h 23m 4s	1/4	PING OK - Packet loss = 0%, RTA = 0.03 ms
	Root Partition	OK	10-02-2024 07:55:39	0d 15h 22m 27s	1/4	DISK OK - free space: / 6114 MiB (75.33% inode=98%):
	SSH	OK	10-02-2024 07:56:17	0d 15h 21m 49s	1/4	SSH OK - OpenSSH_8.7 (protocol 2.0)
	Swap Usage	CRITICAL	10-02-2024 07:56:54	0d 15h 18m 12s	4/4	SWAP CRITICAL - 0% free (0 MB out of 0 MB) - Swap is either disabled, not present, or of zero size.
	Total Processes	OK	10-02-2024 07:57:32	0d 15h 20m 34s	1/4	PROCS OK: 38 processes with STATE = RSZDT

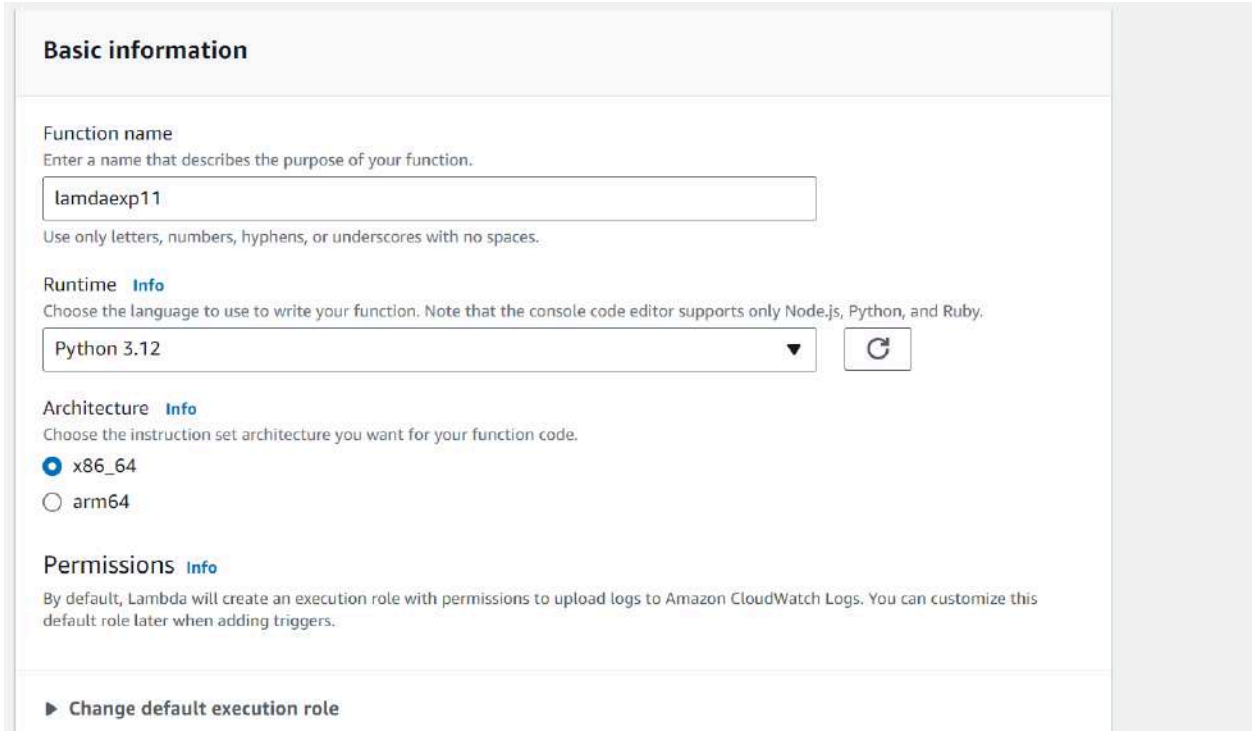
Results 1 - 16 of 16 Matching Services

Conclusion: To conduct this experiment, it's necessary to start the instance from the previous experiment, as it will serve as the host, while the instance created in this experiment will act as the client machine. When I attempted to run the command to verify the Nagios configuration file, I encountered errors. To resolve these errors, I reinstalled the Nagios plugins and restarted the Nagios service, which fixed the issues.

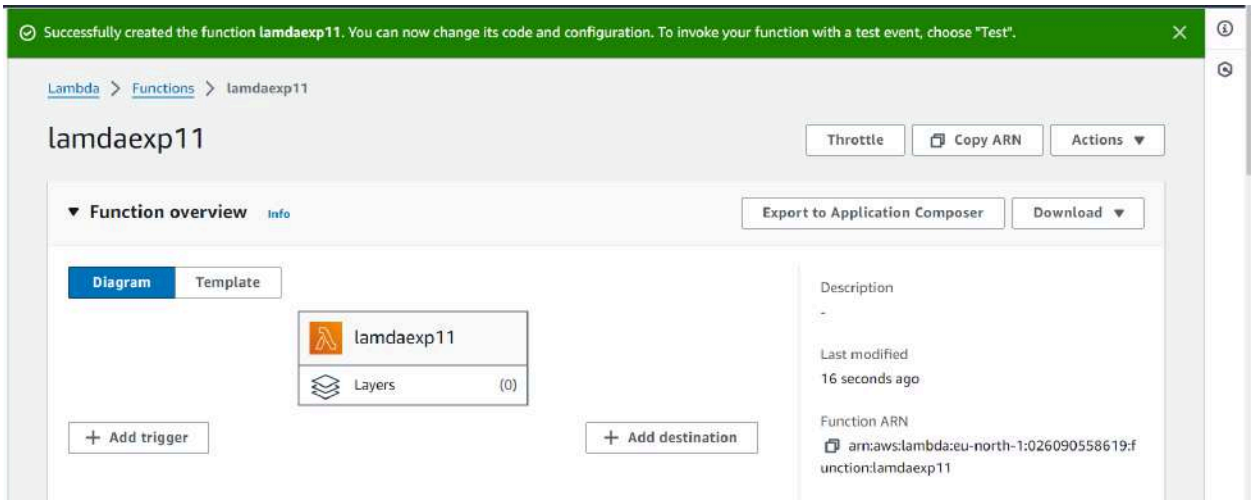
AIM:To understand AWS Lambda, its workflow, various functions and create your first Lambda functions using Python / Java / Nodejs.

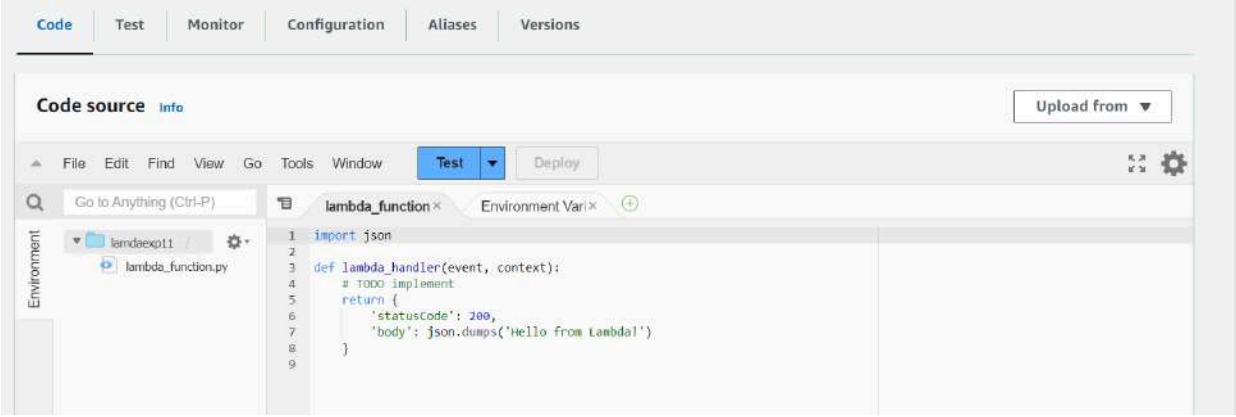
STEP1:Go on your AWS console account and search for lamba and then go on create function Select the author from scratch, add function name and then, choose a runtime env for your function, under the dropdown, you can see all the options AWS supports, Python, Nodejs, .NET and Java being the most popular ones.

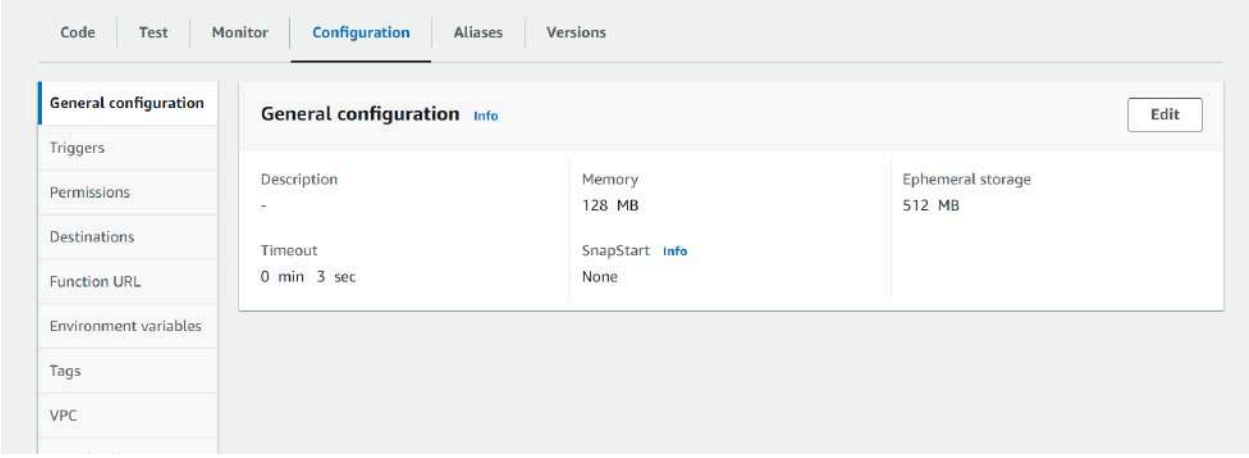




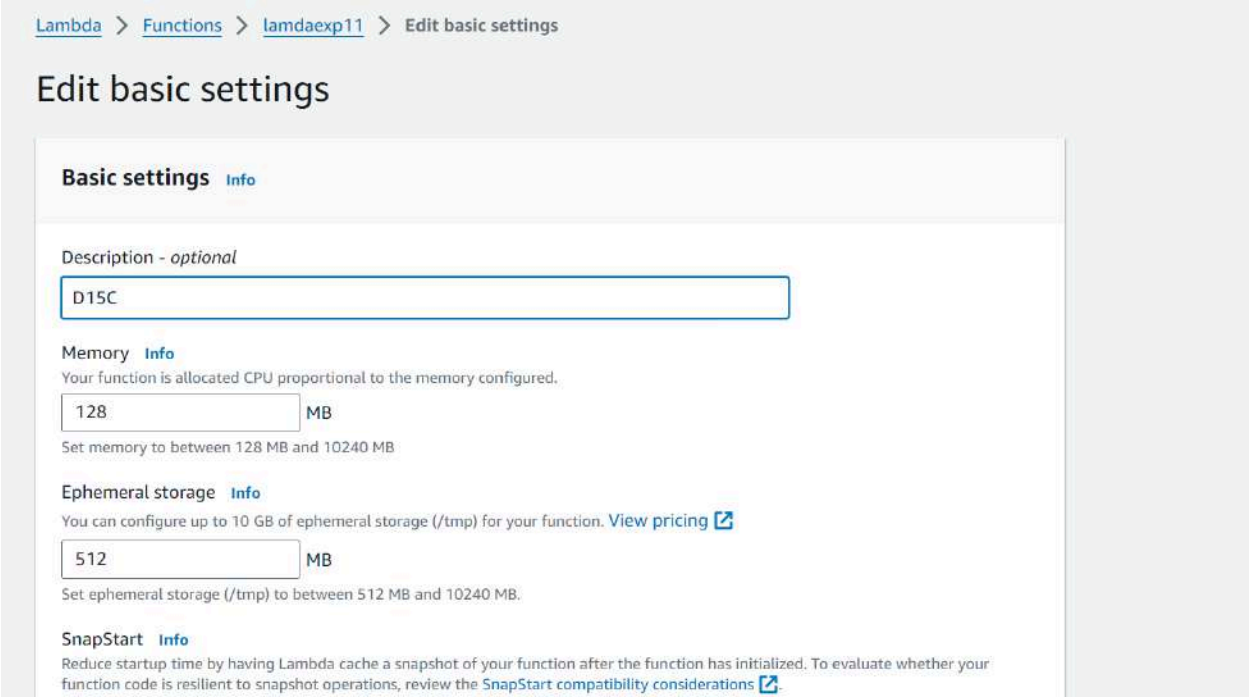
STEP 2:After the function is created successfully go on code write the default code and then configure them.



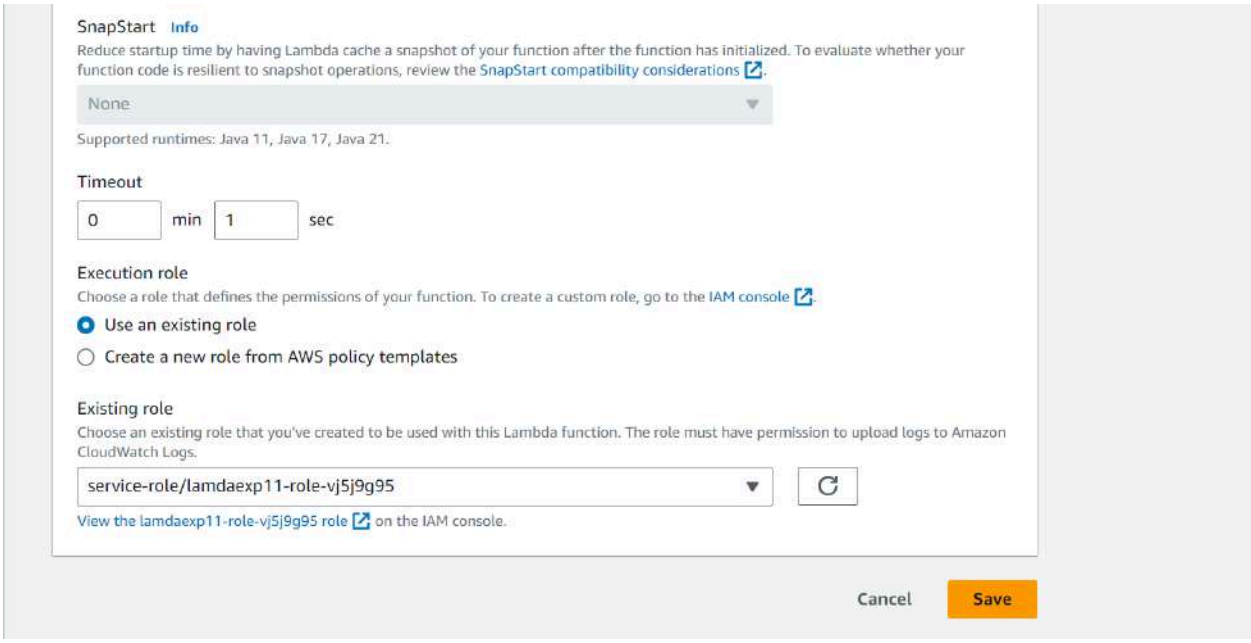




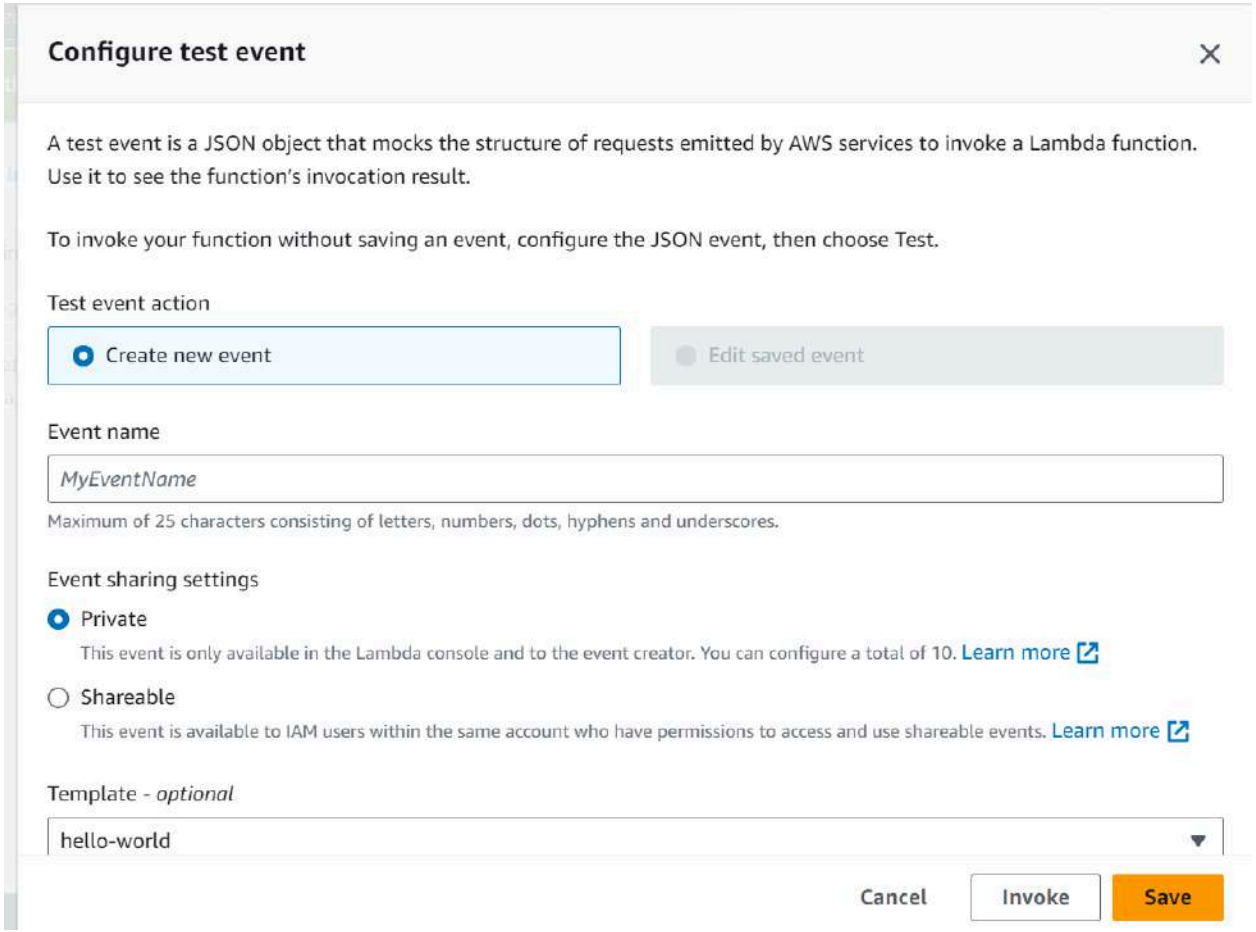
STEP 3: Then go on edit basic settings and add the description and then save it .



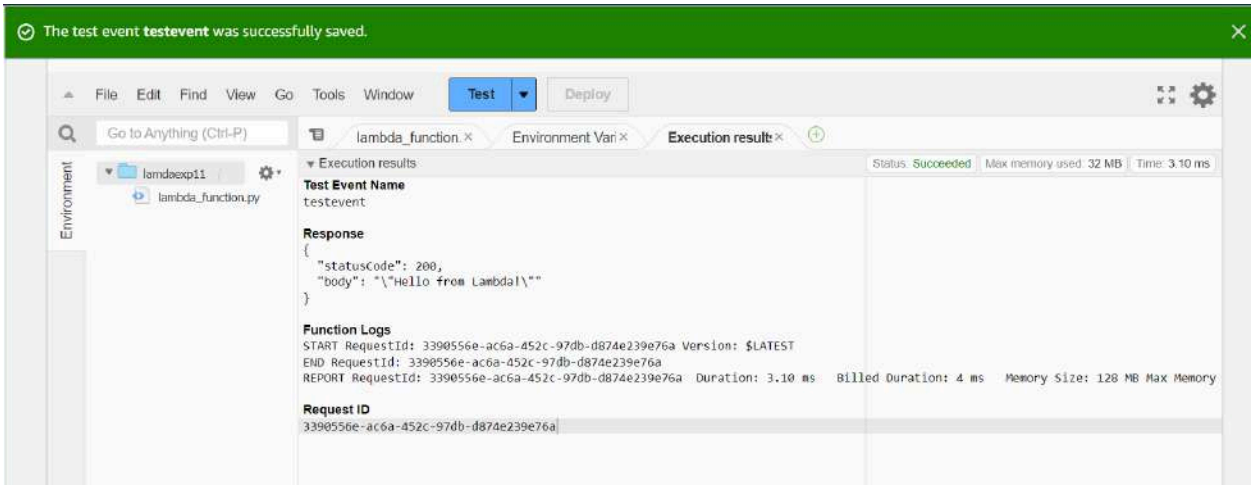
STEP 4:Click on “use an existing role “option and then ahead add the role and save it.



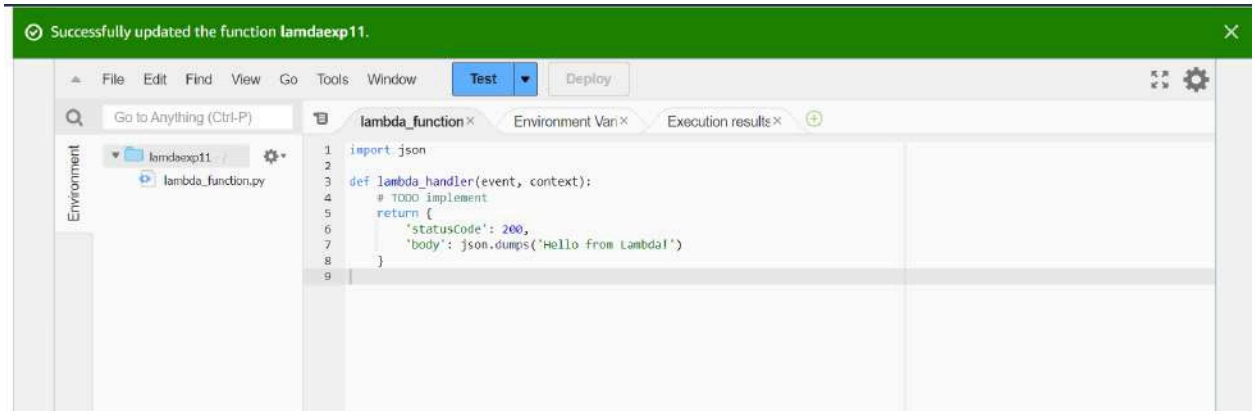
STEP 5:Go on configure test event click on “create new event” edit the event sharing accordingly and select hello world template for template option and then save it.



STEP 6: Click on the test and test the code.



STEP 7: The function is successfully added .

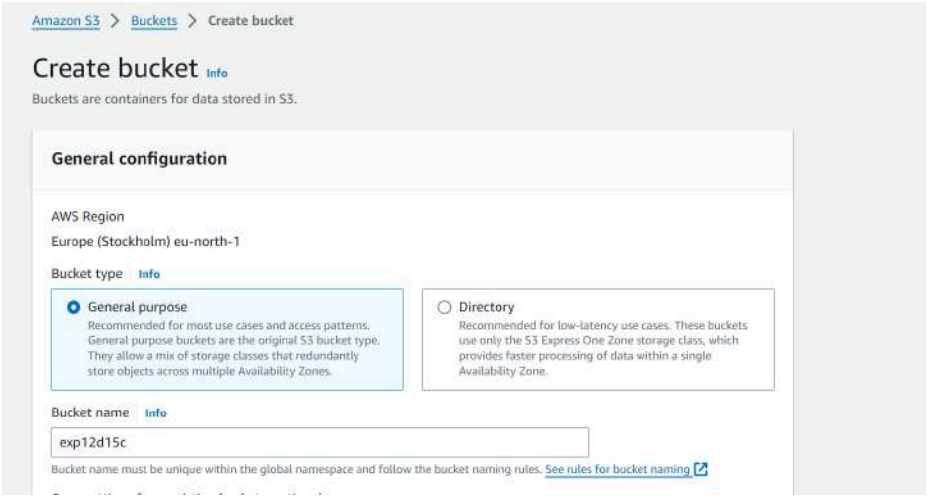


Conclusion:In conclusion, the experiment successfully involved the creation, coding, and deployment of AWS Lambda function. By writing and refining the source code, we demonstrated the ability to implement specific functionality within the Lambda environment. The successful testing of the function confirmed its operational integrity and effectiveness in executing the desired tasks

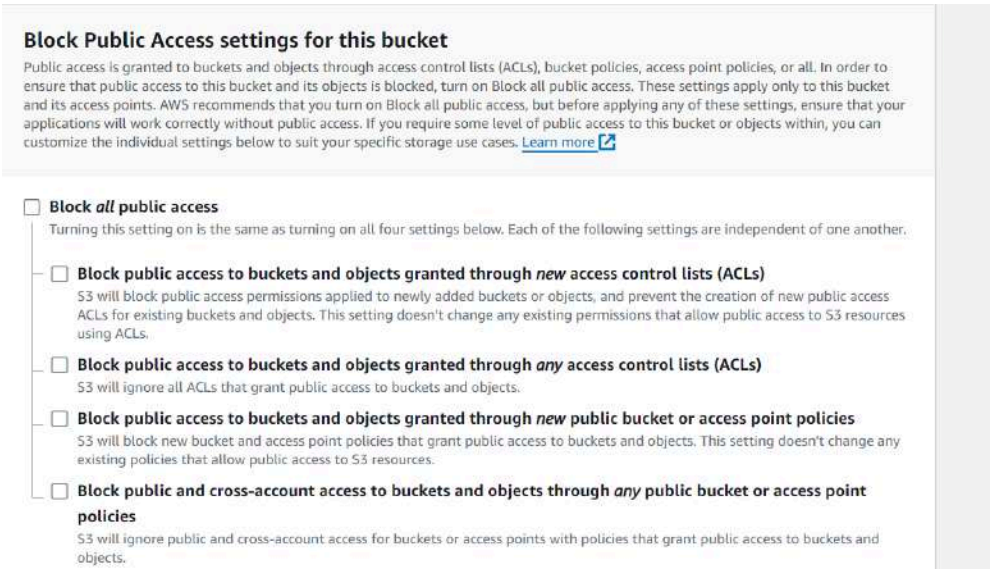
Aim: To create a Lambda function which will log “An Image has been added” once you add an object to a specific bucket in S3

STEPS:

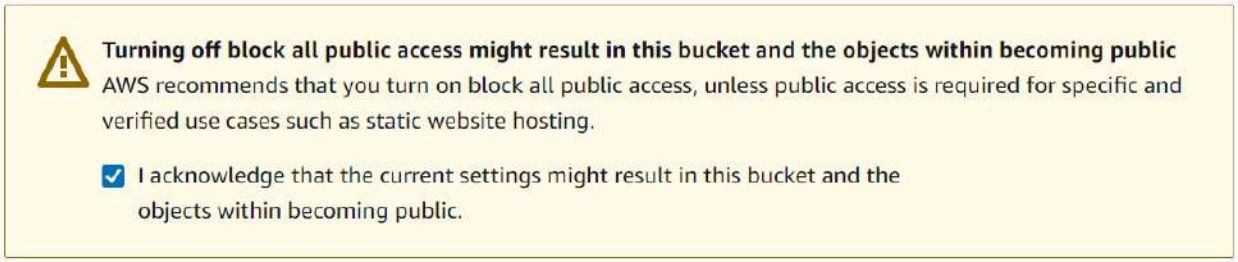
1. Create a S3 bucket and give it a bucket name



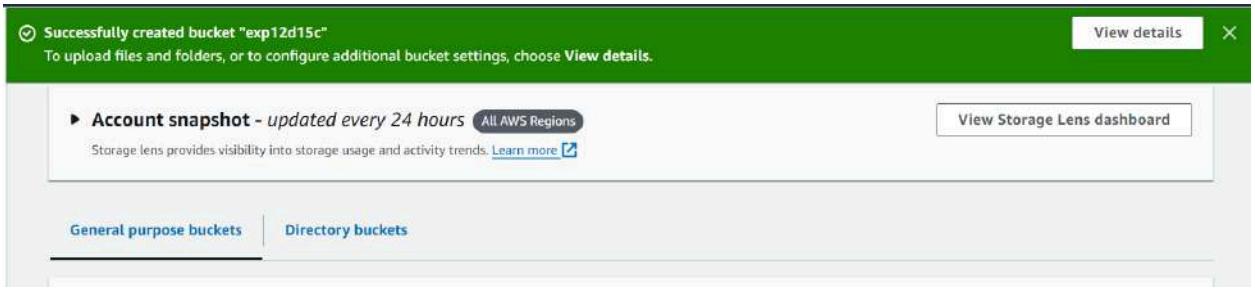
2. Allow public access to the bucket as we are going to add this bucket as a trigger for our lamda function



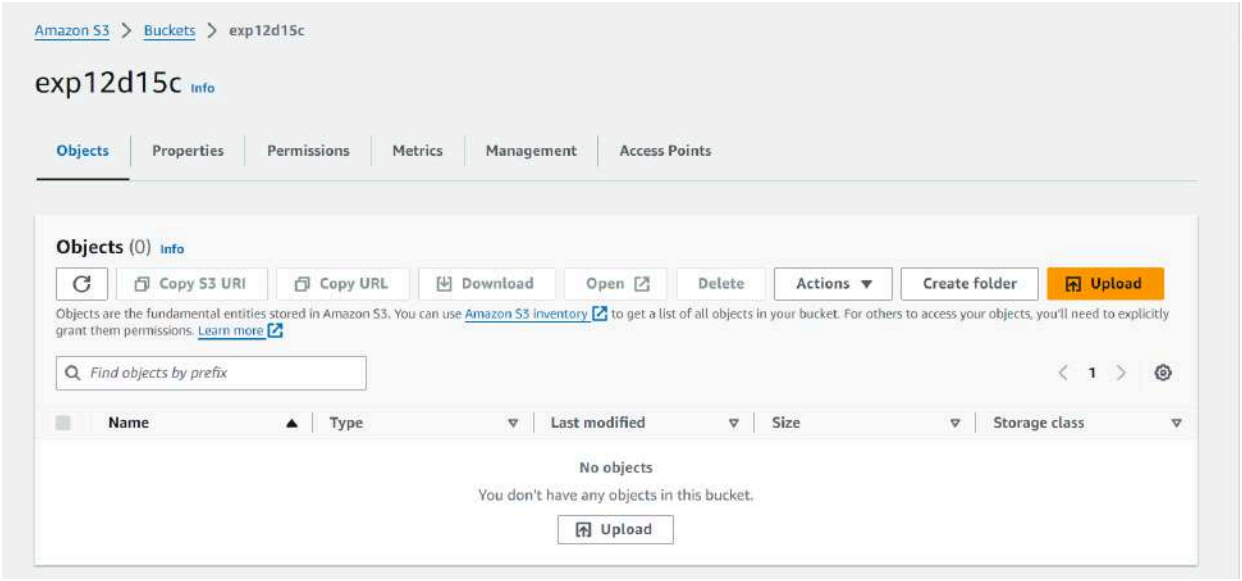
3. Give confirmation that you want to allow full public access and create the bucket



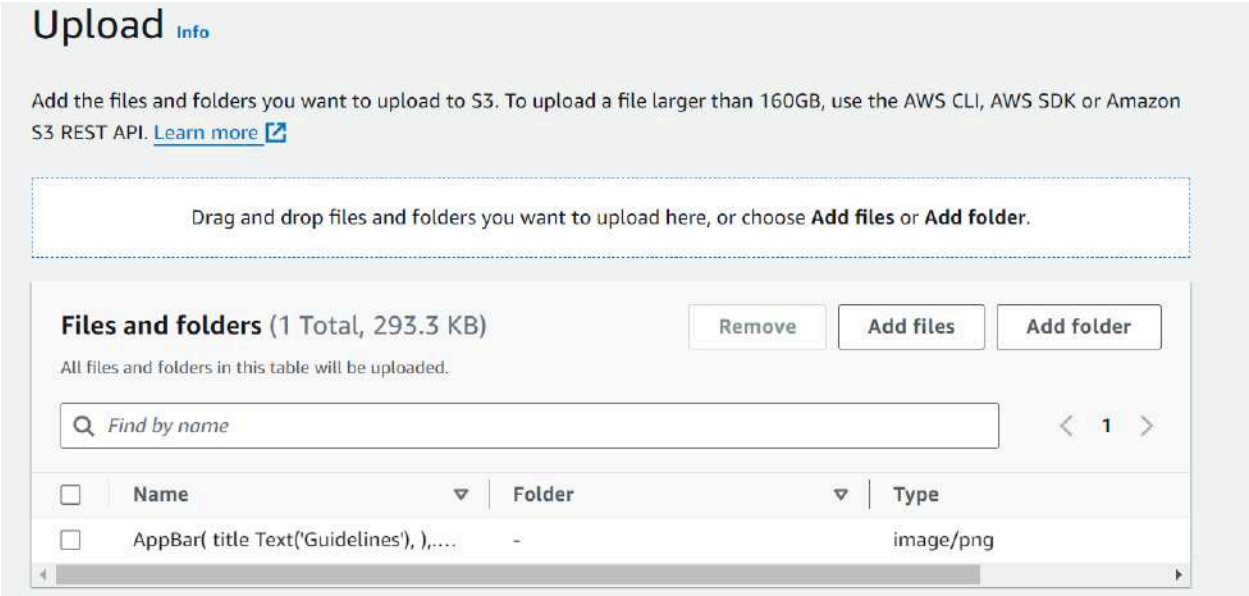
4. You will see the confirmation that the bucket is created successfully



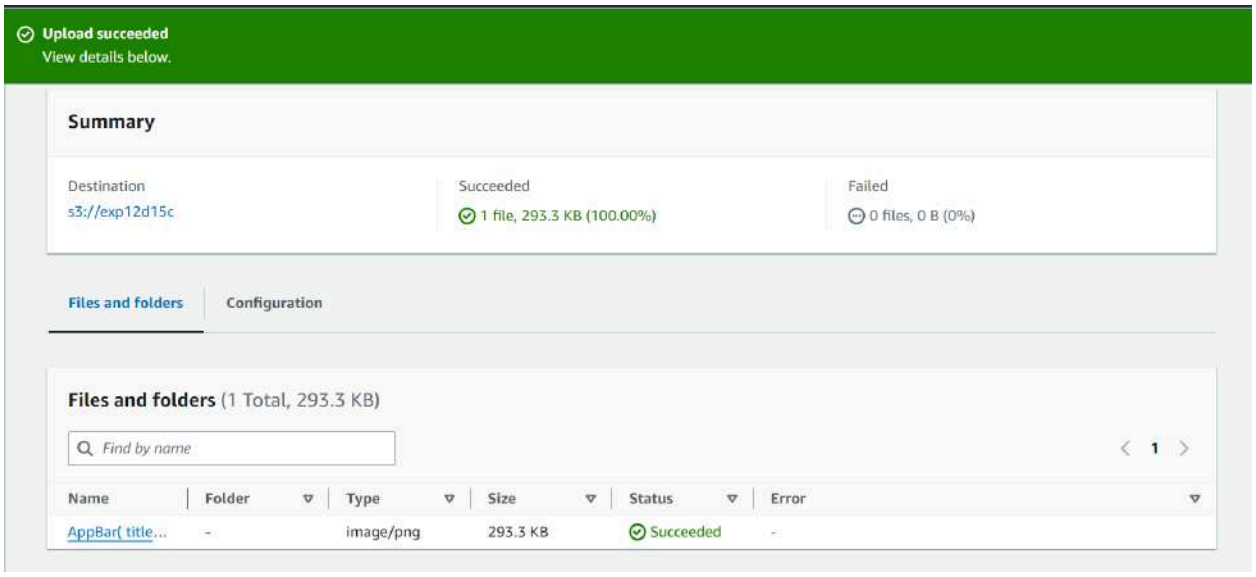
5. Now we need to upload something in the bucket so click on the upload button and add a file



6. I have added a .png extension file; You can upload a .txt file as well

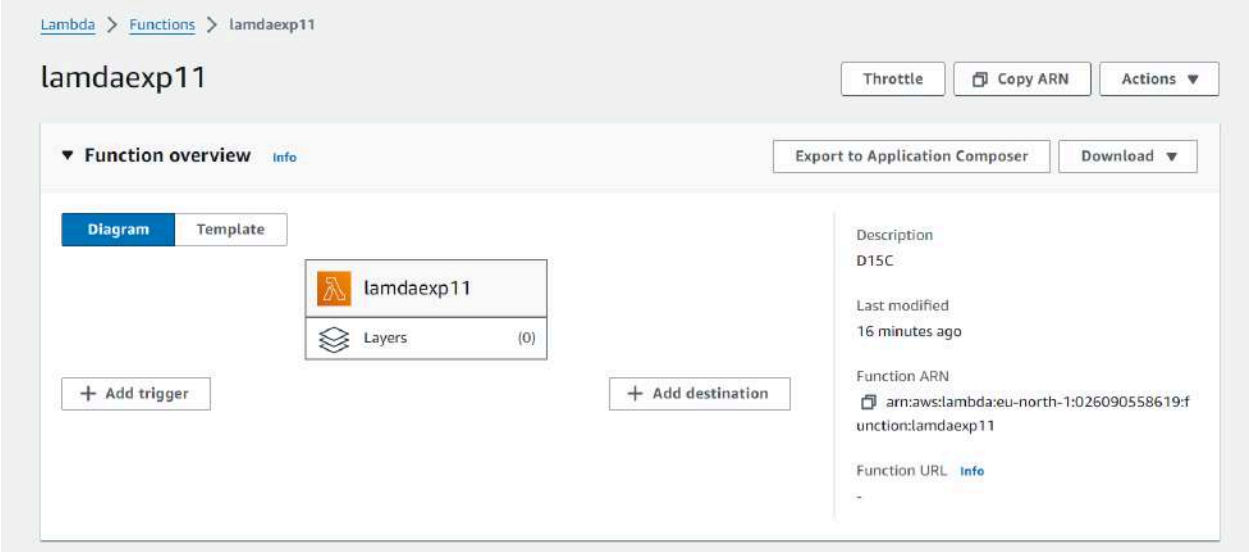


7. Here you can see the confirmation that the upload was a success

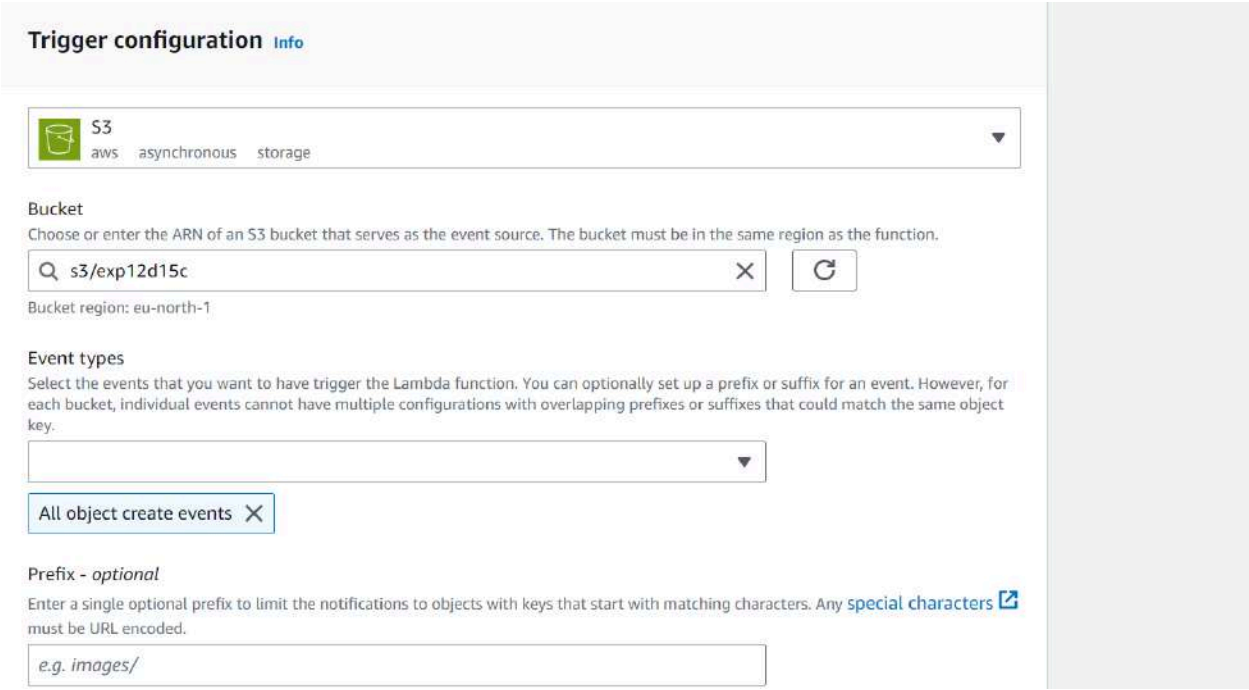


8. Now go back to the aws dashboard and search for lamda function service, Open the function we created in experiment 10. We are going to add this bucket as a trigger to this function

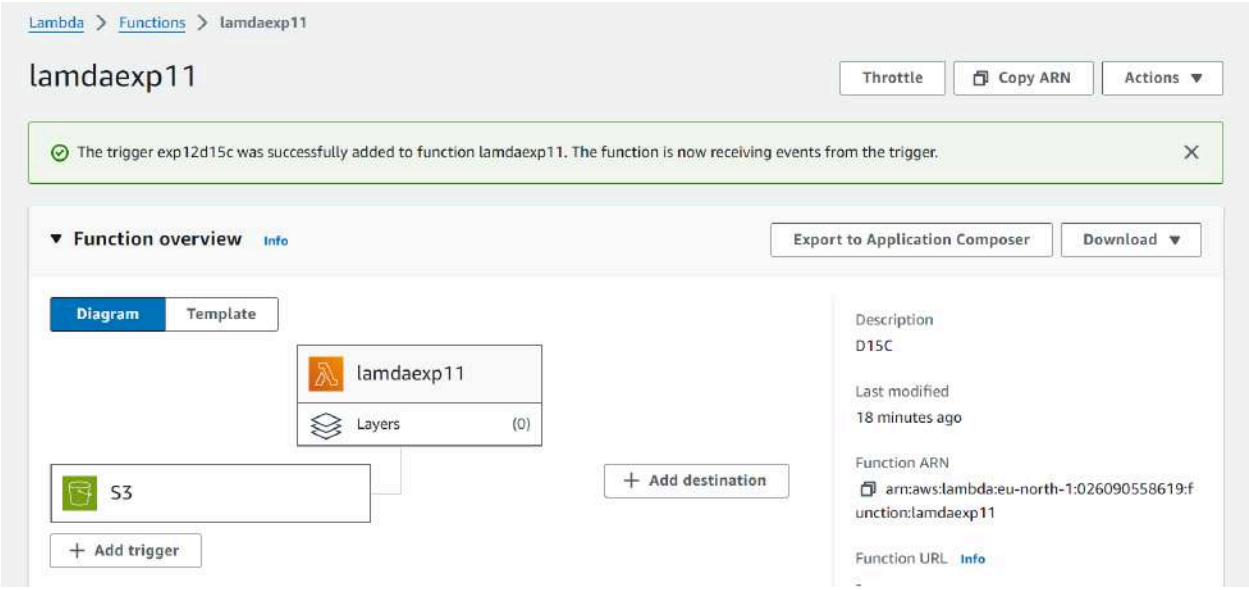
9. On the function overview section of the dashboard you can see the “Add trigger” button. Click on that



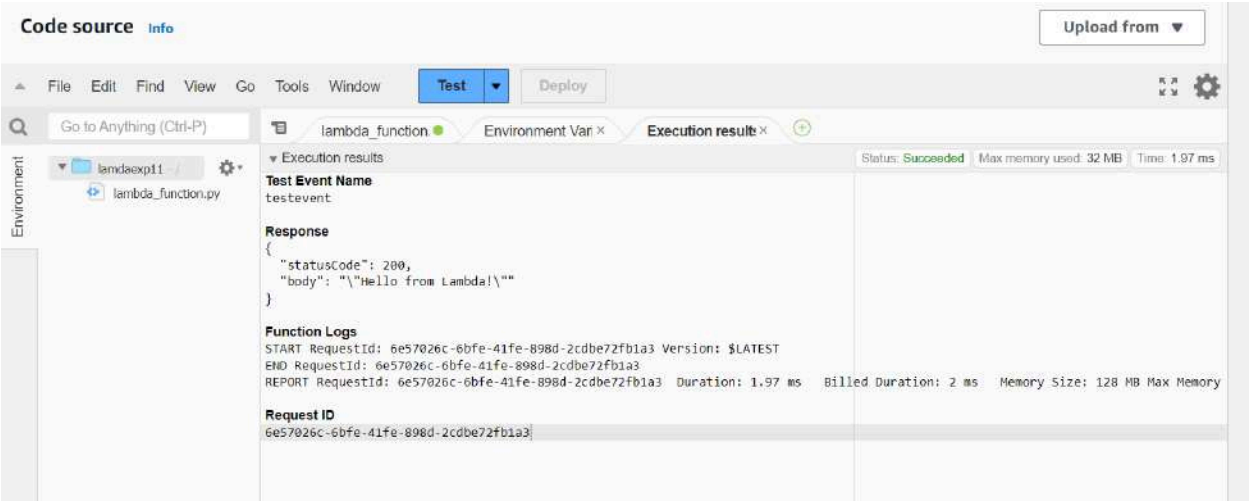
10. It will lead you to the trigger configuration tab; Where you have to select the service and the bucket you created. Add the required configuration information and then save.



11. Here you can see we have the confirmation message as well the the s3 bucket added to our triggers



12. Test the code by clicking on the Test tab ; Here as you can see our code ran successfully



Conclusion: In conclusion, the experiment successfully demonstrated the integration of an S3 bucket with an AWS Lambda function as a trigger. By creating the S3 bucket and configuring it to invoke the Lambda function upon object uploads, we established a seamless workflow for automated processing.

Nikita Chhabaria
D15C (07)

04/05

Adv. DevOps

Assignment 1

Q1 Use S3 bucket and host Video Streaming

Sol:-

Step 1: Set up S3 bucket

(a) Search for S3 bucket on your AWS account. Click on create bucket.

(b) Maintain all the options as default; give your bucket a name

(c) Bucket is created. Now, add a video to that bucket. For that, click on the name of the bucket, this will redirect you to the objects screen which shows the object of your bucket

(d) Click on upload, Here now select an .mp4 extension file for video; Once selected click on upload

Step 2: Set up CloudFront

(a) Search for CloudFront on the services tab. On the left dashboard you will find <origin access> under the <security> tab

(b) Under origin access, click on create origin access identity; Give identity a name & click create

(c) Now go to Distribution from the left panel and click on create a cloudfront distribution

(d) In the origin field select the S3 bucket we created

(e) Under Origin Access, select legacy access identities

(f) Under Origin access identities, select the identity that we created

(g) Update the bucket policy : In default cache beahviour > review > Redirect HTTP or HTTPS naky hosting secure

(h) Keep the remaining options as default & click on create distribution

Step 3: Accessing the hosted video

(a) Once the distribution is deployed, copy the domain name

(b) Go to S3 bucket, copy the key for the video

(c) Combine them like < domain name > / < key of video >

(d) The video is now streaming.

Q2 Discuss BMW & Hotstar case study using AWS

Sol:-

BMW and Hotstar are prime examples of how organisations can leverage the power of Amazon Web Services (AWS) to enhance their operations and deliver exceptional customer experiences. Let's explore how these companies have used AWS to achieve their goals.

BMW: Driving Innovation in the Automotive Industry

BMW faced the challenge of managing vast amounts of vehicle data, ensuring data security, and accelerating the development of new connected car features. By migrating its on-premises data lake to AWS, BMW created a centralized platform for processing & analyzing data from millions of vehicles. This enabled them to:

- Gain deeper insights into driving patterns, vehicle performance & customer preferences
- Develop innovative connected car features like over-the-air updates, voice-activated assistants, and predictive maintenance

- Strengthen data security with robust measures on the AWS platform

Hotstar: Revolutionizing Digital Entertainment

Hotstar, a leading digital entertainment platform in India, sought to deliver high-quality streaming content to millions of users with varying internet speeds and devices. By adopting AWS, Hotstar built a scalable and reliable streaming platform capable of:

- Delivering seamless streaming experiences to users worldwide, regardless of their internet connections or devices
- Scaling to meet peak demand during major sporting events & other popular content creator releases
- Reducing latency and improving user experience through AWS's global network.

Key Takeaways:

(1) AWS empowers innovations: Both BMW & Hotstar have used AWS to drive innovation & enhance their offerings

(2) Scalability is essential: AWS's elastic infrastructure has enabled these companies to scale their operations to meet increasing demands

(3) Data-driven insights: By leveraging AWS, BMW has been able to gain valuable insights from its vehicle data

(4) Exceptional user experiences: Hotstar has used AWS to deliver a superior streaming experience to its customers.

In conclusion, BMW & Hotstar demonstrate how organizations can effectively leverage AWS to achieve their business objectives & provide exceptional value to their customers. By embracing cloud technology, these companies have been able to stay ahead of the curve & drive innovation in their respective industries.

3 Why Kubernetes and advantages and disadvantages of Kubernetes. How adidas uses Kubernetes Lol:-

Kubernetes, an open-source platform for managing containerized applications, has revolutionized the way businesses deploy, scale and operate their software. For Adidas, a global sportswear giant, Kubernetes has been a game-changer enabling them to enhance their e-commerce operations and deliver a superior customer experience

Advantages of Kubernetes

- Scalability: Kubernetes can effortlessly scale applications up or down to meet fluctuating demands. For Adidas, this means their e-commerce platform can handle peak traffic during sales seasons like Black Friday without compromising performance
- Portability: Kubernetes applications can be deployed across various cloud platforms and on-premises infrastructure. This flexibility allows Adidas to choose the most suitable environment based on their specific needs & cost considerations

Q3

- Efficiency: Kubernetes optimizes resource utilization by dynamically allocating resources based on application demands. This helps Adidas reduce costs & improve overall performance.

Disadvantages of Kubernetes

- Complexity: Kubernetes can be complex to learn & manage, especially for organizations new to containerization
- Operational Overhead: Managing Kubernetes requires specialized knowledge and can introduce additional operational costs
- Vendor lock-in: Some Kubernetes distributions may be tied to specific cloud providers or vendors, potentially limiting flexibility & increasing dependency

How Adidas leverages Kubernetes

- Scalability: Adidas can dynamically scale their e-commerce platform to handle peak traffic during sale seasons
- Reliability: Kubernetes ensures high availability & minimizes downtime, guaranteeing a consistent

customer experiences

- portability: Adidas can deploy their e-commerce platform and on-premises infrastructure
- Efficiency: Kubernetes optimizes resources utilization, reducing costs & improving overall performance

Q4 What are Nagios and explain how Nagios are used in E-services

Sol:-

Nagios is a popular open-source network monitoring system that has become an essential tool for organization that rely on e-services. It provides real-time information on the health and performance of computer systems, applications & network infrastructure, allowing administrators to proactively identify & resolve potential issues

Key Features of Nagios

① Comprehensive Monitoring: Nagios can monitor a wide range of systems and services, including servers, networks, applications and databases

Q 4

(3) Customizable Alerts: Users can configure Nagios to send alerts via email, SMS or other Notification methods

(4) Detailed Reporting: Nagios generates detailed reports on system performance, allowing administrators to analyze trends and identify areas for improvement

(5) Extensibility: Nagios supports a vast ecosystem of plugins, extending its capabilities to monitor specific systems & applications

Uses in E-services

In world of e-services, Nagios plays a crucial role in ensuring the reliability and performance of online applications. Here are some common use cases

- Website Monitoring: Nagios can monitor the availability & response time of websites, ensuring that they are accessible to users
- Application performance Monitoring: Nagios can track the performance of key application metrics such as CPU usage, memory consumption & database query times

04/05

Assignment 2

Create a REST API with the serverless framework

Step 1 : Install Prerequisites

(a) Install Node.js from the official website

(b) Install the serverless framework globally using npm

```
npm install -g serverless
```

Step 2 : Create a Serverless project

(a) Create a new directory for your project & navigate into it

(b) Initialise a new serverless service using the AWS Node.js template

Step 3 : Setup AWS credentials

(a) Configure AWS CLI if not already set up

(b) Set up your AWS Access Key, Secret Key, Region & Output Format

Step 4: Modify serverless.yml configuration

(a) Open serverless.yml & define your REST endpoints, Lambda functions, and even

(b) Add HTTP event triggers for the CRUD operations (POST, GET, PUT, DELETE)

Step 5: Write Lambda Function Handlers

(a) In handler.js, write the function to handle the API requests (eg. create, retrieve, update, delete)

Step 6: Deploy your service

(a) Deploy the REST API to AWS

(b) After deployment, note down the API gateway endpoints URLs from the console

Step 7: Test the API

Use tools like curl, Postman or browser to test the deployed API using the provided URLs

Case Study for Sonarqube.

- Create your own profile in sonarqube
- Use sonarcloud to analyse your github code
- Install sonarlint in your Java intellij id or eclipse ide & analyse your java code
- Analyse python project with sonarqube
- Analyse node.js project with sonarqube
- Analyse node.js project with sonarqube

Sol:

In modern software development, maintaining code quality is paramount for ensuring the long term sustainability & reliability of a project. Sonarqube & sonarcloud offer powerful tools for static code analysis, helping developers identify and fix issues in their code.

① Setting up Sonarqube for local testing

- Installation: Download & install Sonarqube from sonarqube.org
- set up a database for sonarqube
- start the sonarqube server by running the sonar.sh script in the sonarqube installation directory
- Open http://localhost:9000 in your browser & login and create a new project.

(2) Use sonarcloud to Analyse your github code

- Go to sonarcloud
- Sign up with your github account & give sonarcloud access to your repositories
- Create a new project & select the github repository to analyse
- Configure the project key and organization
- Follow the provided instruction to set up sonar-project.properties file in the root of your repository
- Use github actions, Jenkins or manually trigger the analysis using sonarscanner analyse the code quality in the cloud

(3) Install sonarlint in IntelliJ IDEA or eclipse for Java code Analysis

- Open IntelliJ IDEA and navigate to File → Settings → Plugins
- Search for "sonarlint" and install it
- Restart IntelliJ after the installation is complete
- Open your java project, right-click and select Analyse → sonarlint. This will analyse the code and highlight any issues

(2) Use sonarcloud to Analyse your github code

- Go to sonarcloud
- Sign up with your github account & give sonarcloud access to your repositories
- Create a new project & select the github repository to analyse
- Configure the project key and organization
- Follow the provided instruction to set up sonar-project.properties file in the root of repository
- Use Github actions, Jenkins or manually trigger the analysis using sonarscanner to analyse the code quality in the cloud

(3) Install sonarlint in IntelliJ IDEA or Eclipse for Java code Analysis

- Open IntelliJ IDEA and navigate to File → Settings → Plugins
- Search for "sonarlint" and install it
- Restart IntelliJ after the installation is complete
- Open your java project, right-click and select Analyse → sonarlint. This will analyse the code and highlight any issues

Analyse Python Project with sonarqube

Install the sonarqube scanner on your machine
In your Python project directory, create a
sonar-project.properties file with the
following content

```
sonar.projectKey = your-project-key
sonar.sources = .
sonar.language = py
sonar.pythonversion = 3.x
sonar.sourceEncoding = UTF-8
```

Run the analysis by executing the following
command in the terminal

```
- sonar-scanner
```

After analysis, view the results on your
sonarqube dashboard

Analyse Node.js project with sonarqube

- In your Node.js project directory, create a
sonar-project.properties file with the same
content as mentioned in the previous case
- Install SonarScanner globally if not installed
- Run the analysis using sonar-scanner
- Check the analysis results on the SonarQube
dashboard to identify and fix potential
issues

Q: Creating a self-service infrastructure model using Terraform for a large organisation involves the following steps:

- Step 1:- Define Infrastructure standards:-
  Establish clear standards and best practices for infrastructure deployment, including naming conventions, resource types, tagging policies and security compliance. This foundation ensures consistency across the organisation.

- Step 2:- Create Terraform modules:-
  Develop reusable Terraform modules based on your organisation's standards. Each module defines resources and configurations, allowing teams to deploy infrastructure efficiently & consistently.

- Step 3:- Set up terraform cloud or enterprise:
  Use terraform cloud for centralized management of configurations & state files, enabling collaboration & access control for infrastructure changes.

- Step 4:- Configure version control:
  Integrate terraform modules with version control (eg. Github). This tracks changes, facilitates collaboration & ensures proper versioning for updates & compatibility.

Step 5:- Integrate with service now to automate infrastructure requests. This trigger runs terraform and thus streamlining the process for teams

Step 6:- Provide Documentation & training :-

Create Documentation & training for using terraform modules & submitting requests helping teams understand & follow best practices

Step 7:- Monitor & support :-

Monitor the usage of the self-service model & provide ongoing support to users. Gathering feedbacks helps identify that the infrastructure remains compliant and super efficient.

By following above steps, organisations can enable product teams to manage their own infrastructure through a standardised terraform approach