Name: Abhinav Swaminathan **Div: D15C** **Roll No: 01**

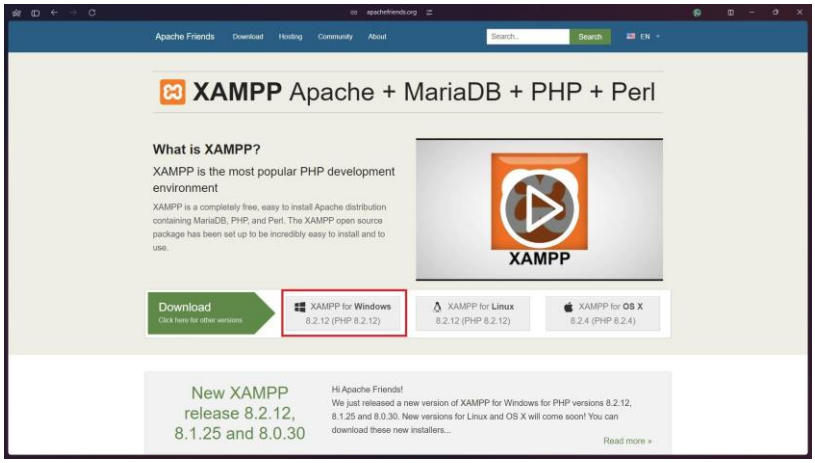
Experiment No. 1A

Static Hosting

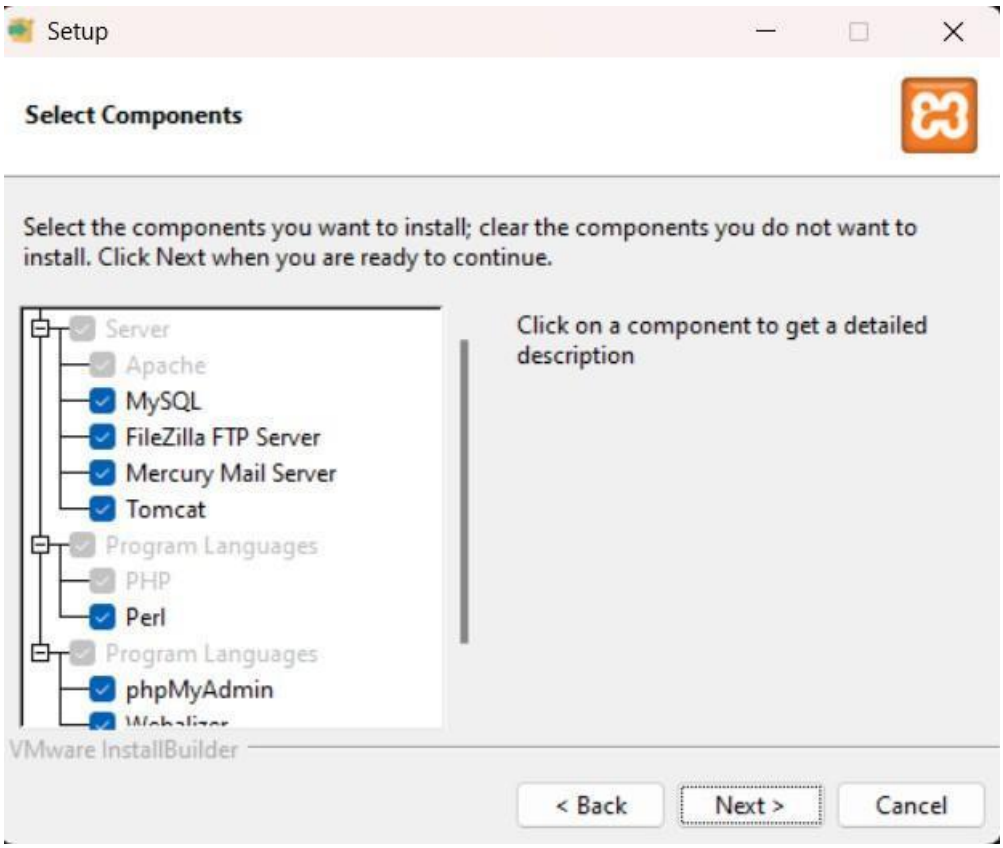
On local server (XAMPP)

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

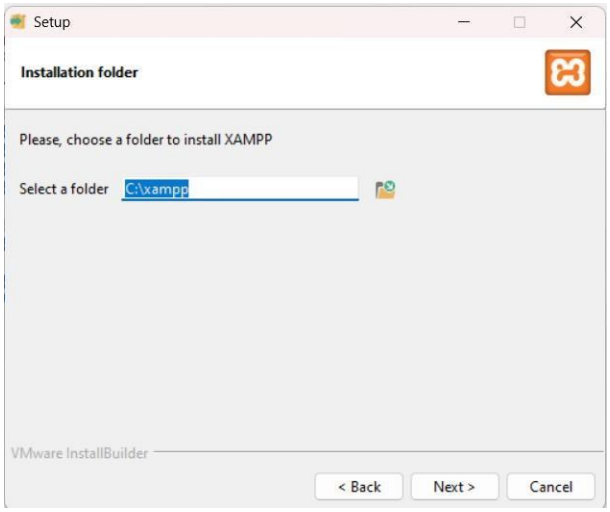
1) Select your OS. It will automatically start downloading.



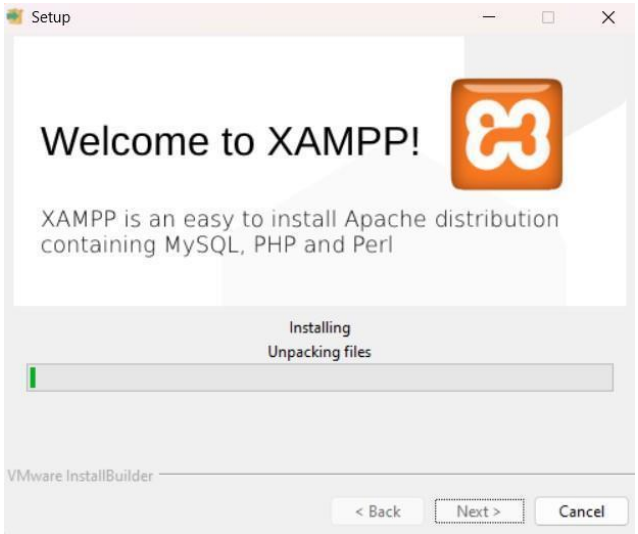
2) Open the setup file. Select all the required components and click next



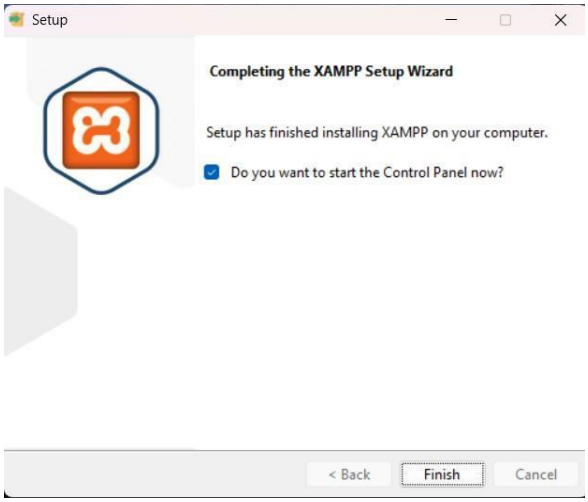
3) Choose the folder to install XAMPP in. Make sure the folder is empty. Click next



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

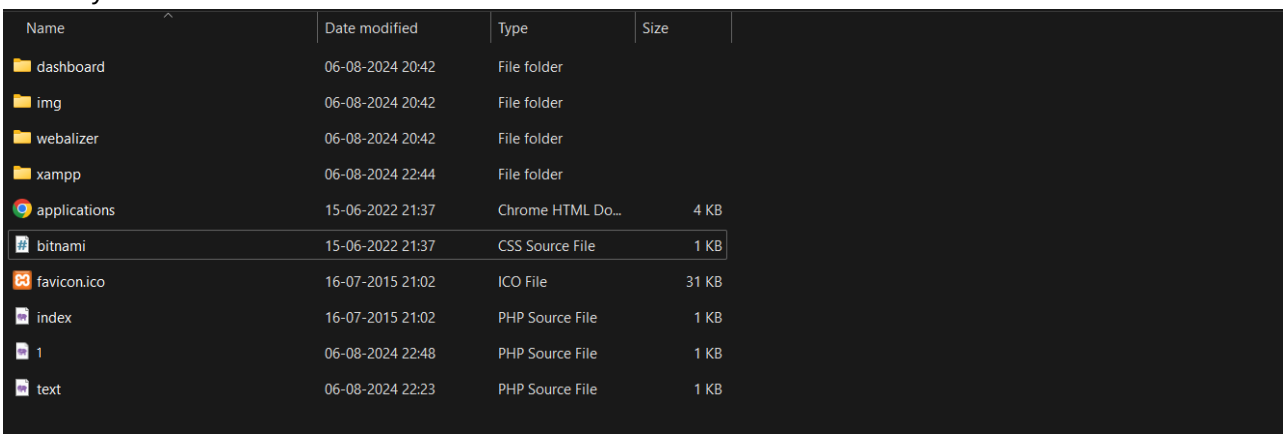


5) The installation is complete. Click Finish

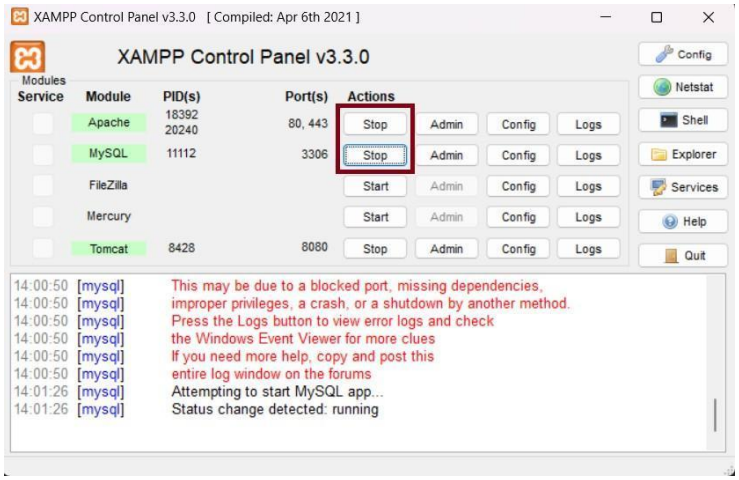


Step 2: Setup a file that is to be hosted on the server. Make sure the file has extension **.php**

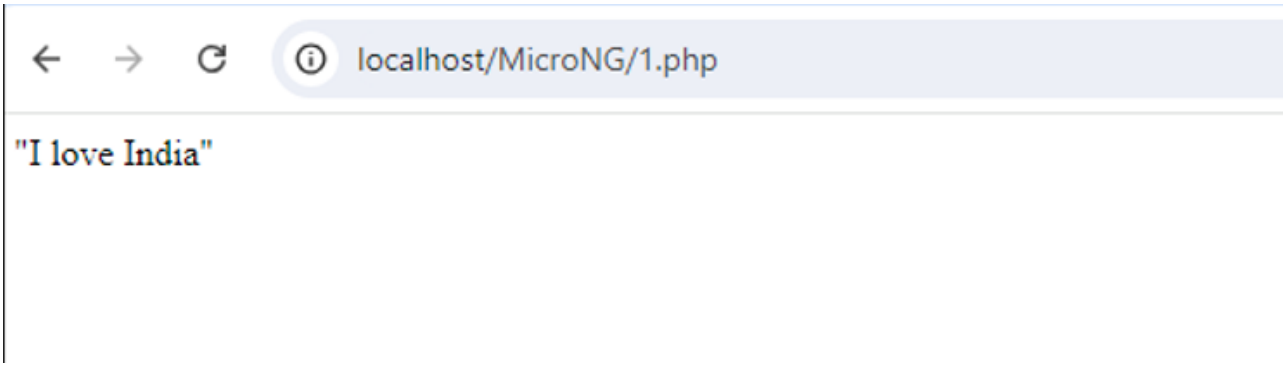
Step 3: Go to the directory where XAMPP was installed. Go to **htdocs** folder. Place your folder in this directory.



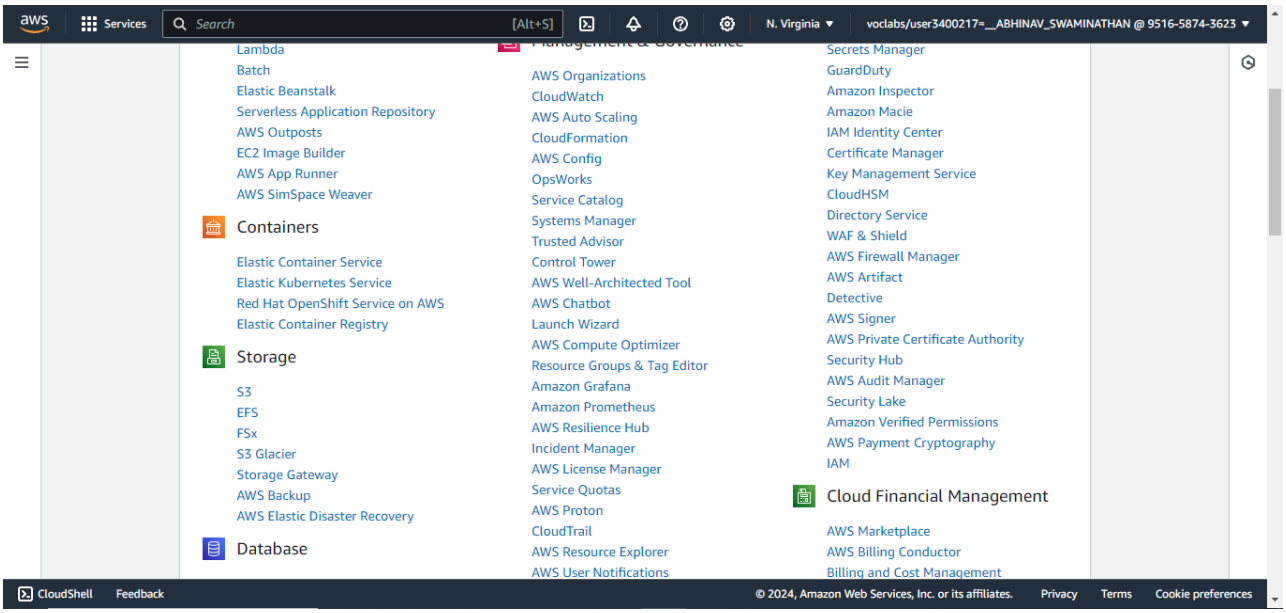
Step 4: Open XAMPP Control Panel, start the Apache service (Required) and mySQL service (if needed)



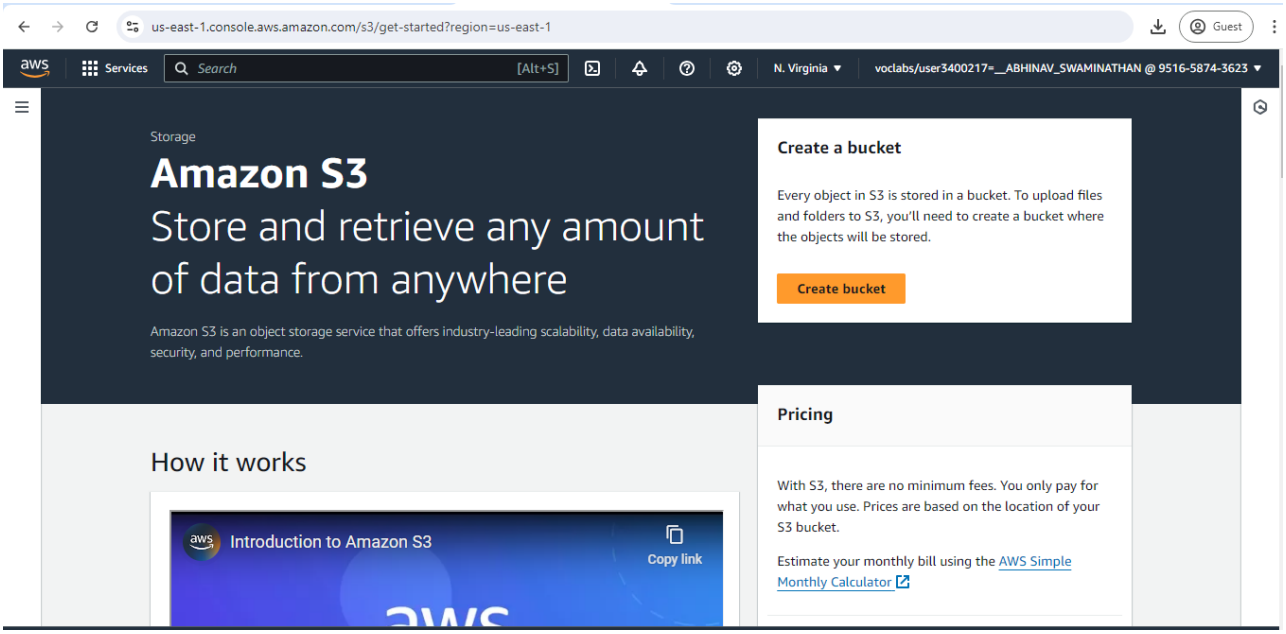
Step 5: Open your web browser. Type localhost/YOUR_FILENAME.php. This will open your website on your browser.



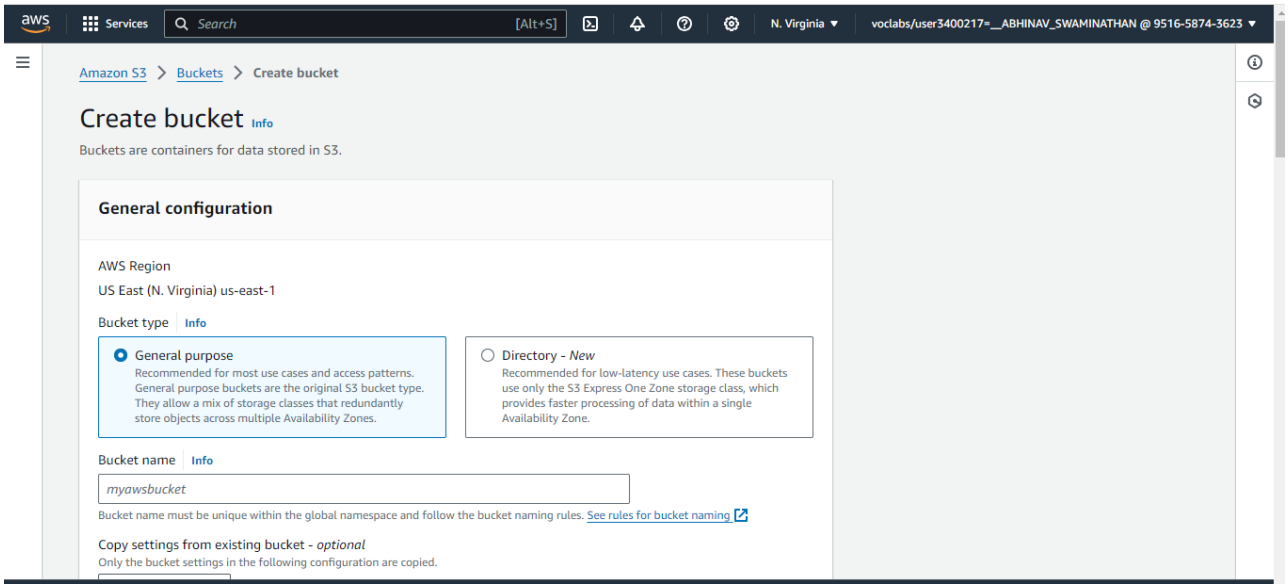
AWS S3



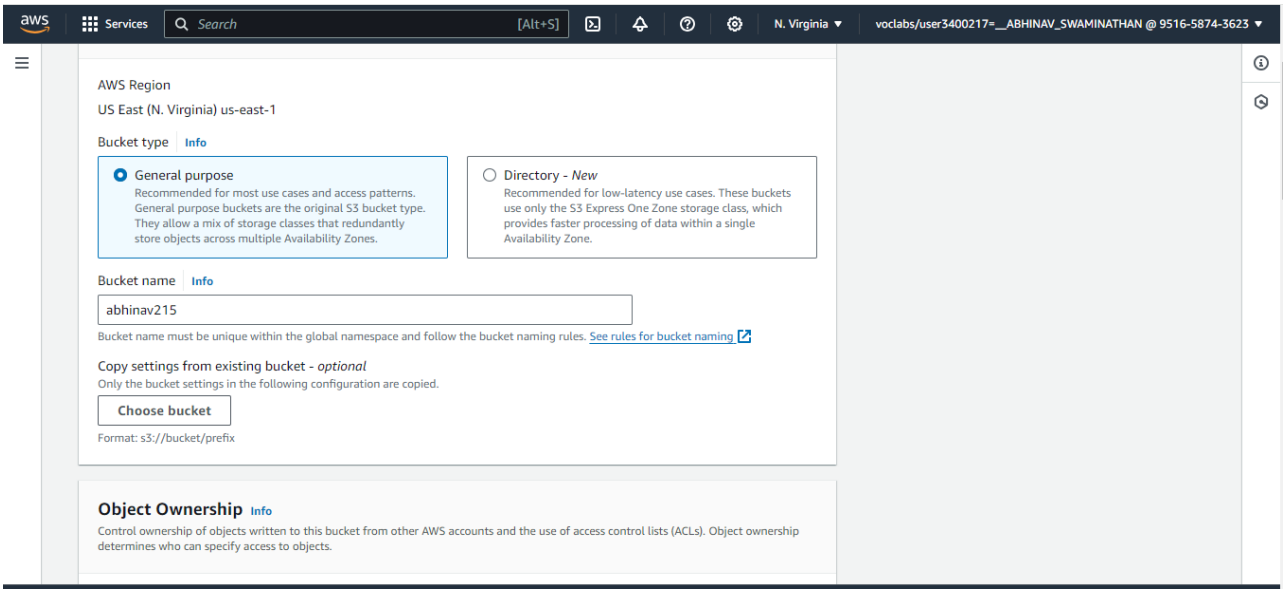
Step 1: Login to your AWS account. Go to services and open **S3.**



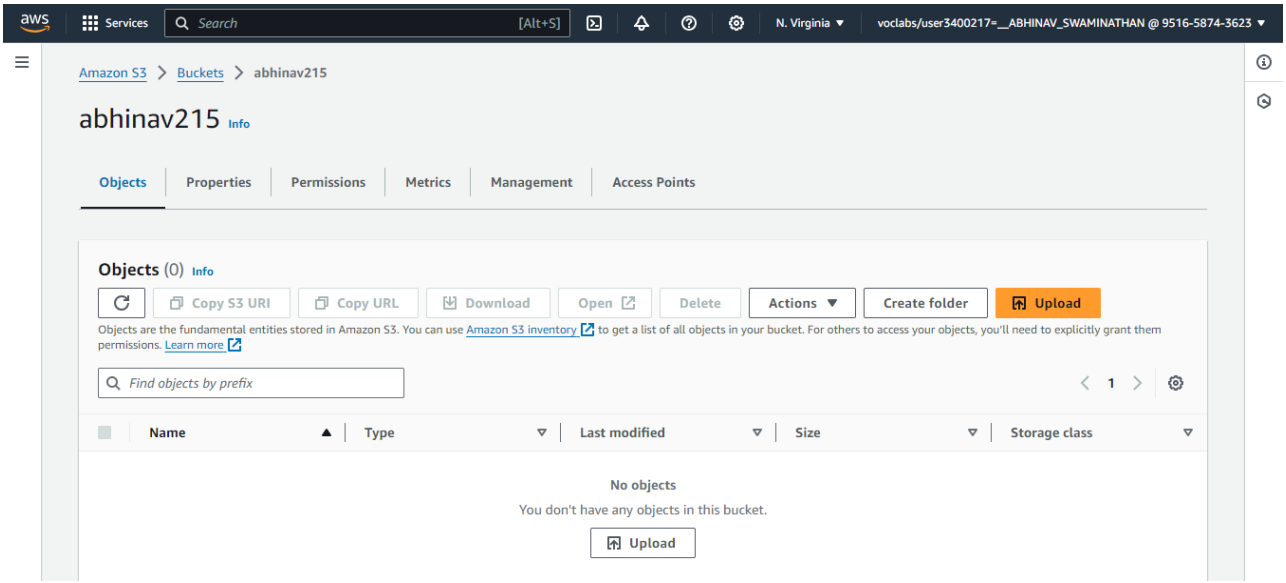
Step 2: Click on Create Bucket

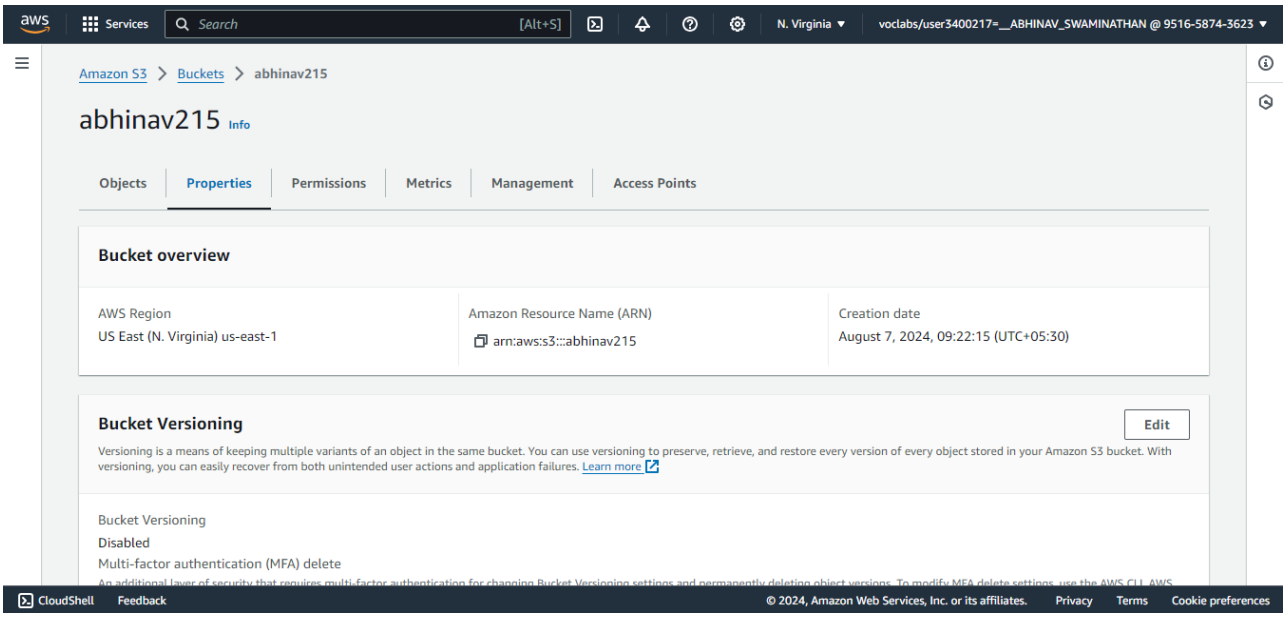


Step 3: Give a name to your bucket, keeping other options default, scroll down and click on Create Bucket

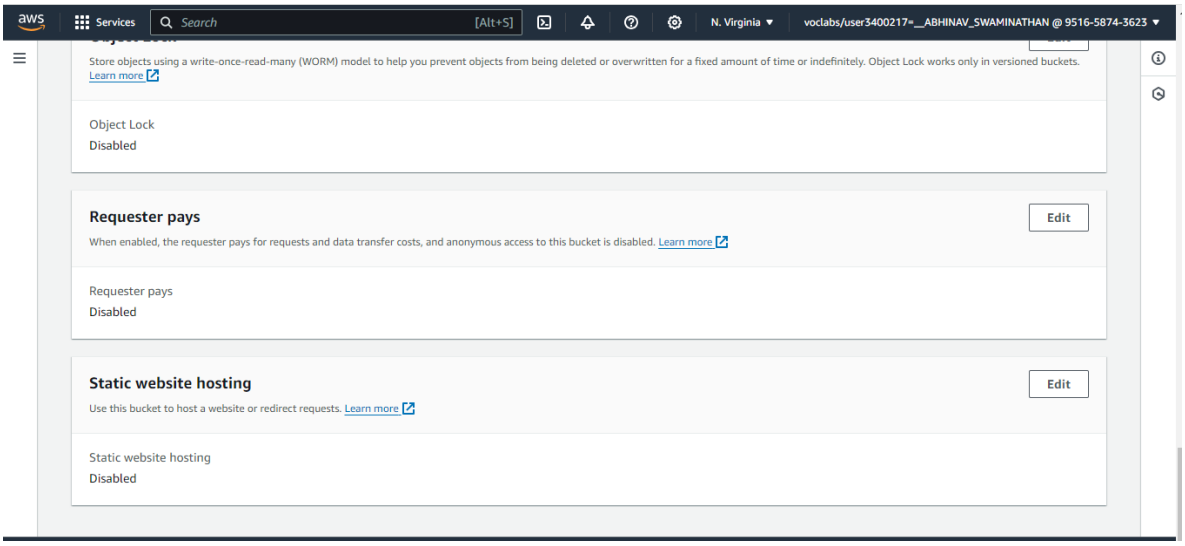


Step 4: Click on the name of your bucket and goto Properties

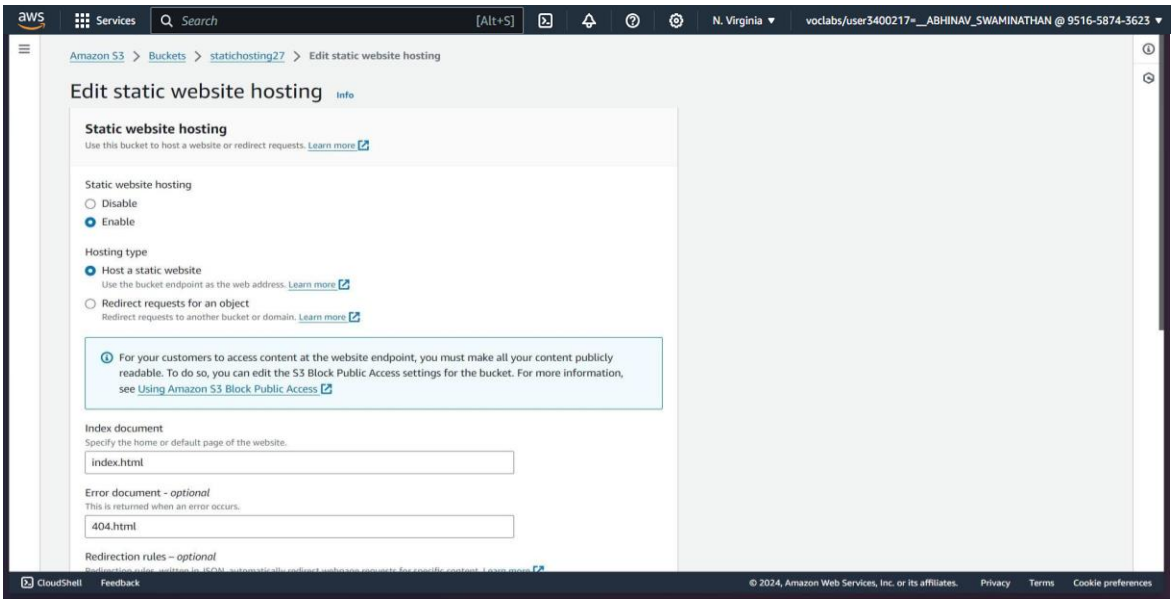




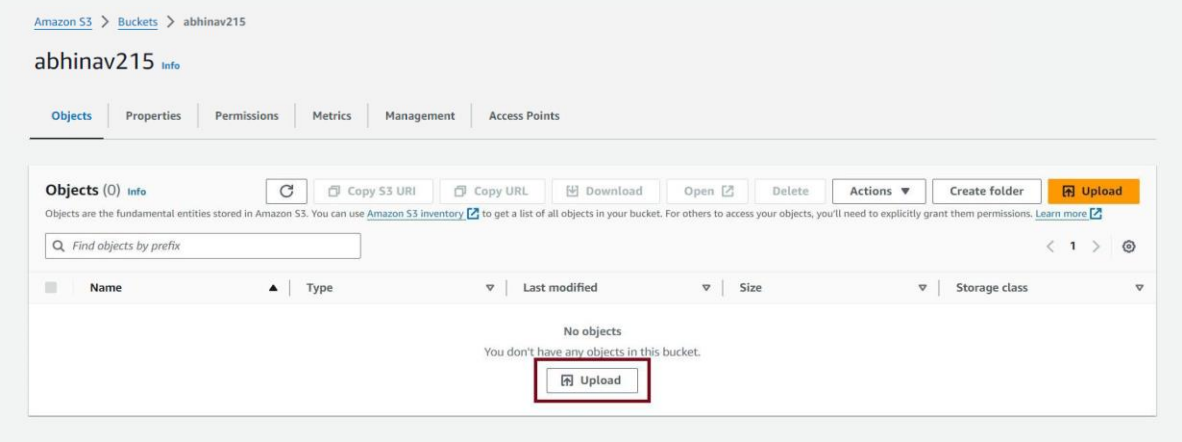
Step 5: Scroll down till you find Static website hosting, click on edit



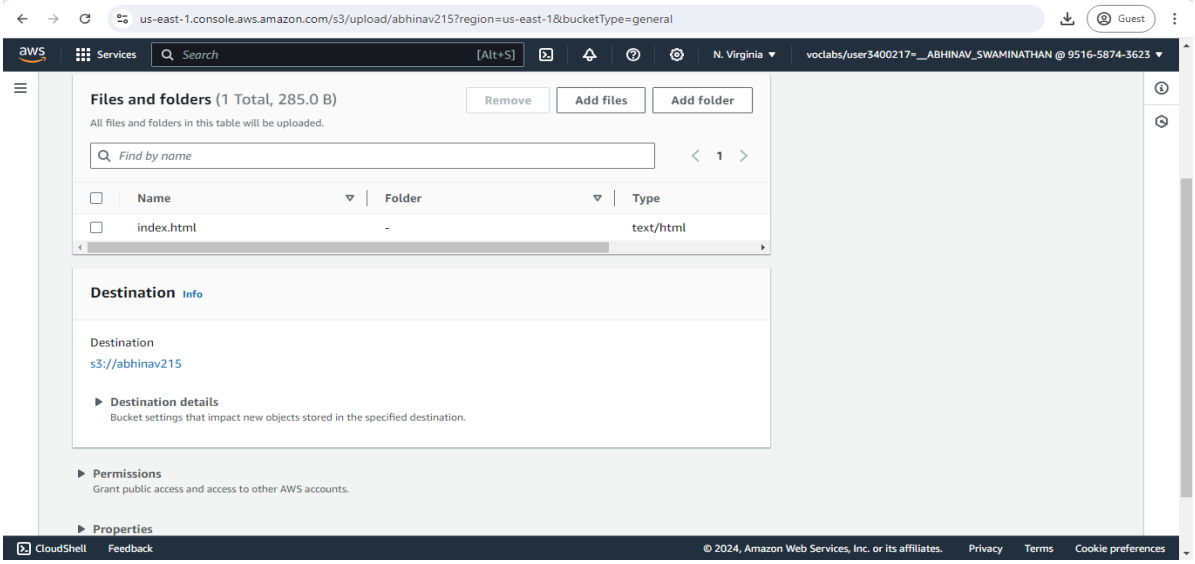
Step 6: Enable static website hosting, in Index document, write the name of your document and in error document, give name as 404.html. Save your changes.



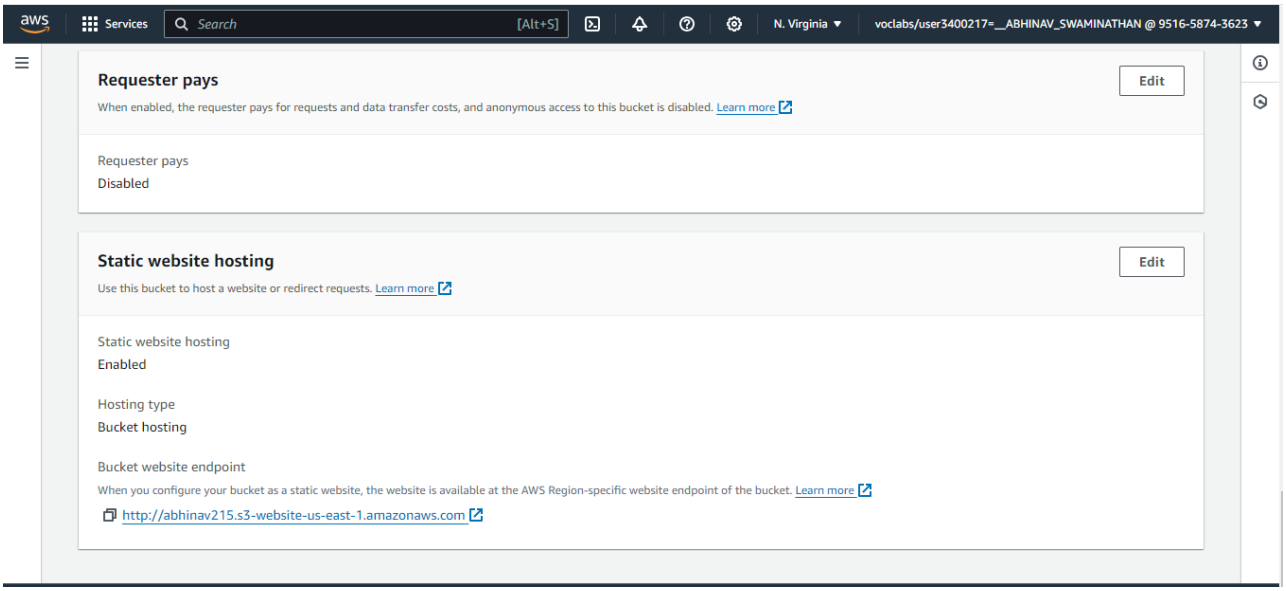
Step 7: Go to Objects tab and click on upload file.



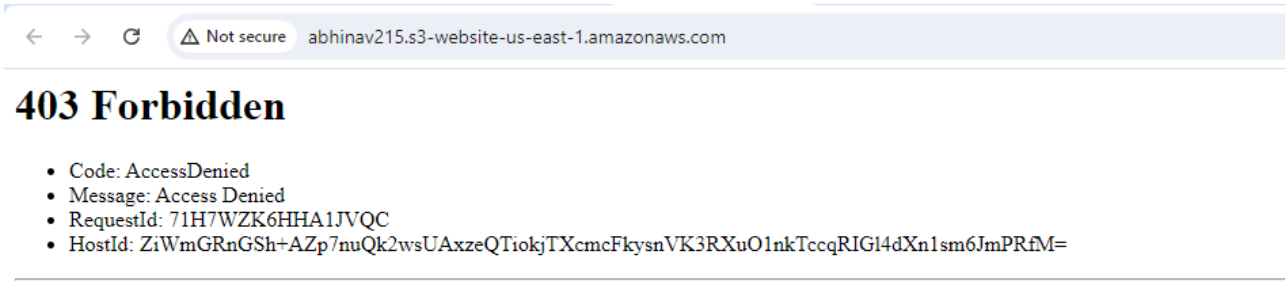
Step 8: Click on Add files. Add all the files you want to upload. Then scroll down and click on Upload



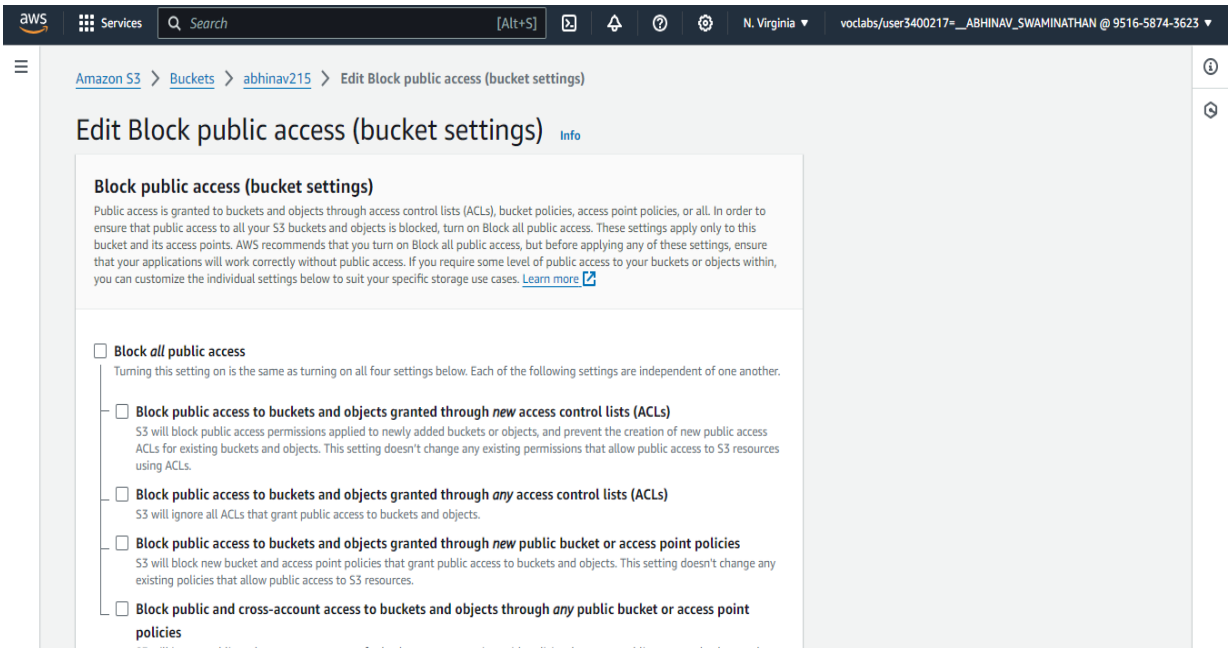
Step 9: This will take you to the Objects screen. Switch to Properties, scroll down to Static web hosting. There you would find the link (Bucket website endpoint) to your website.



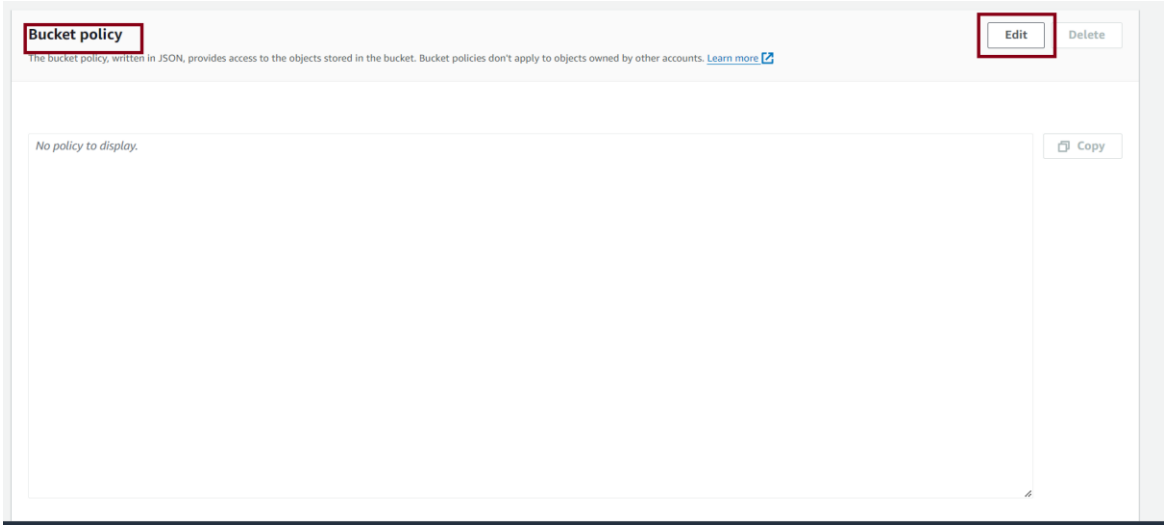
Step 10: Open the link. It will show a 403 forbidden error screen as the contents of the bucket are not available for the public users. To change this, go to Permissions tab, go to Block public access and click on edit



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



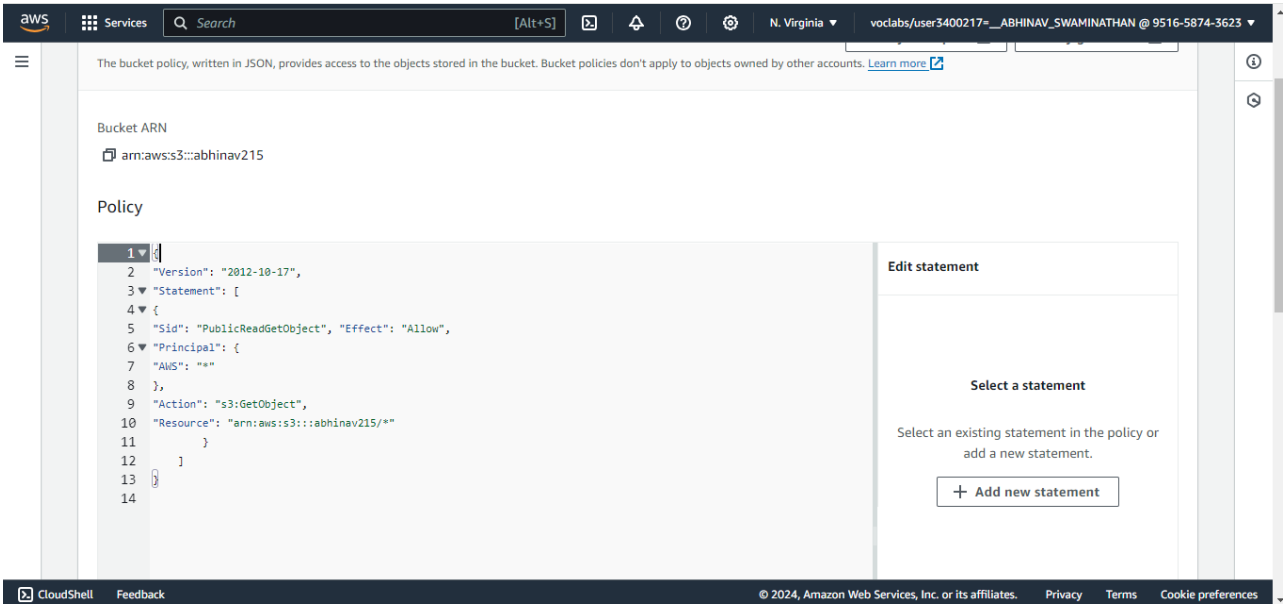
Step 12: Scroll down to bucket policy and click edit



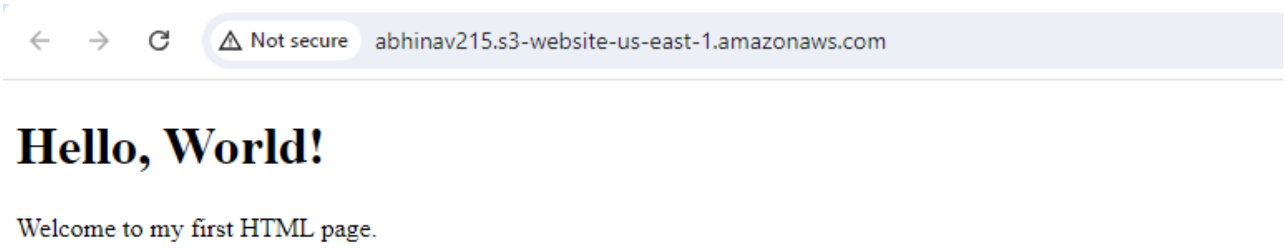
Step 13:

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

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



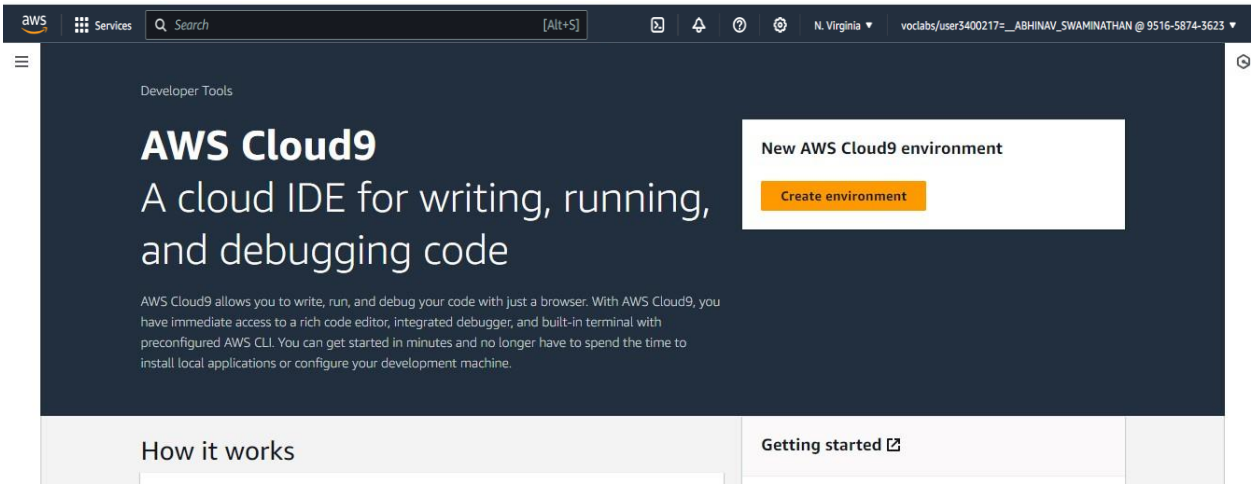
Step 14: Now reload the website. You can see your website



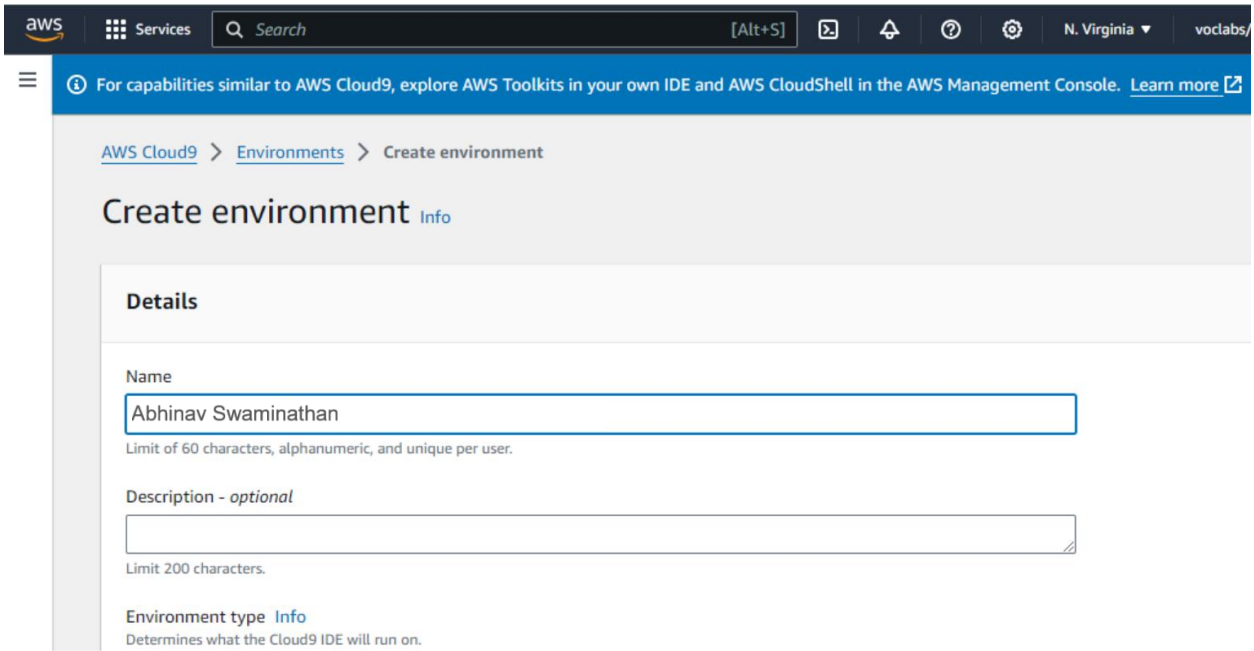
Name: Abhinav Swaminathan D15C Roll No:01

Experiment No. 1B

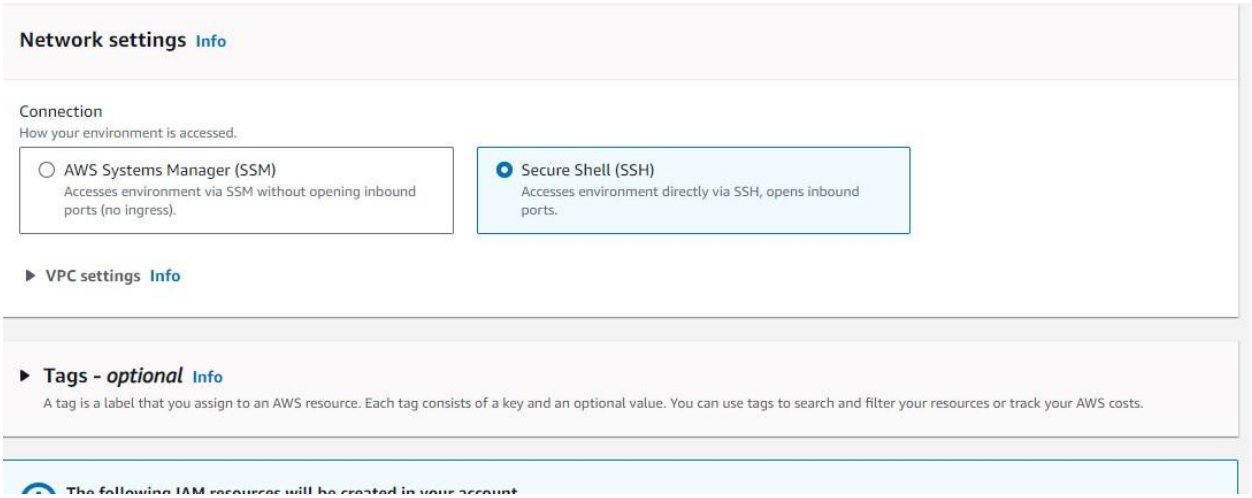
Open the AWS account and search for Cloud9. Click on create environment.



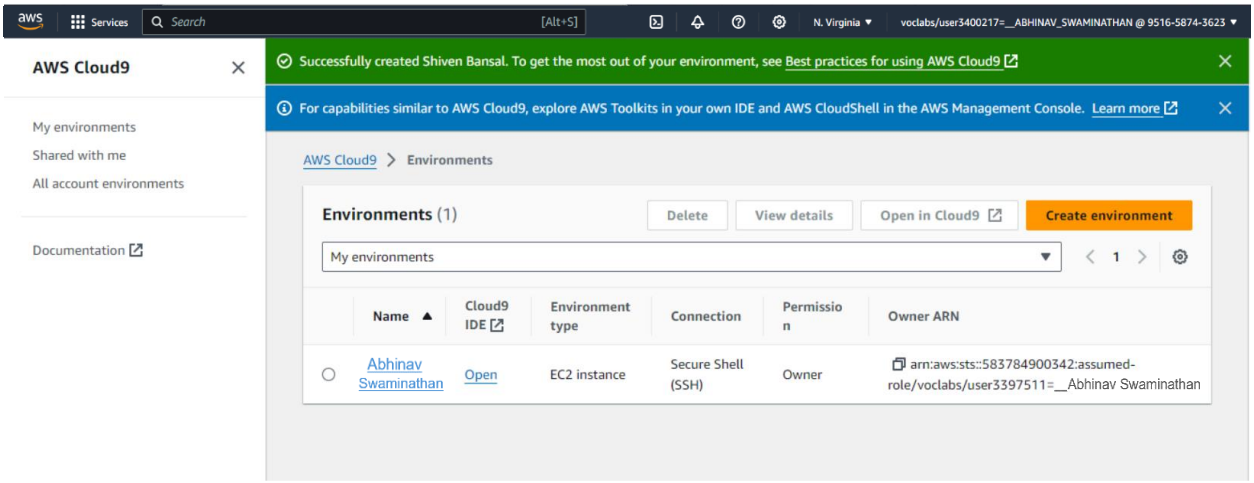
Enter the name and other required configuration for creating an environment. In network settings, using the AWS system manager gives an error while creating the environment. It states there was an error creating IAM resources needed for SSM.



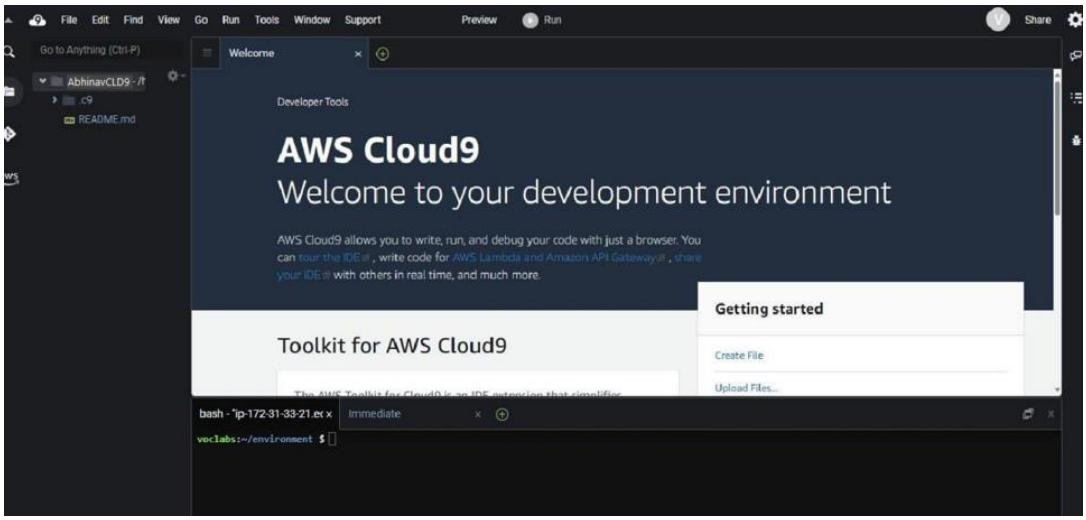
Use the Secure Shell option in Network settings



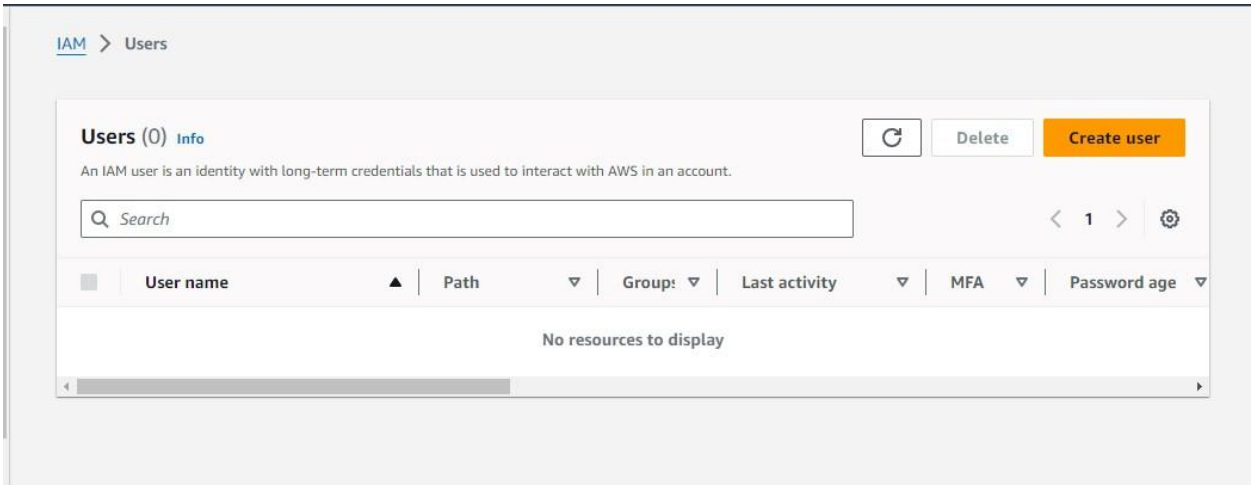
Once the configuration is complete, click on create environment to create a Cloud9 environment.



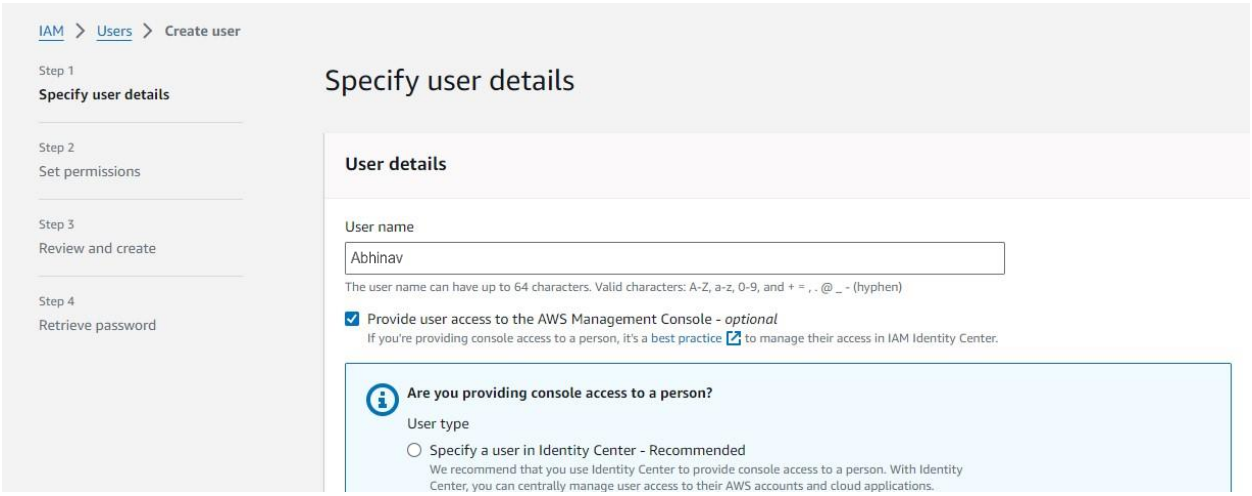
Click on the environment name to open the created Cloud9 Environment.



Open the aws account and search for IAM service. Then go to users tab and click on create user to create a new user.



Give the user name, select the “provide user access” checkbox. Also select the option “I want to create an IAM user”. Otherwise we will have to enable the Identity center and specify a user there.



Provide user access to the AWS Management Console - *optional*

If you're providing console access to a person, it's a best practice to manage their access in IAM Identity Center.

Are you providing console access to a person?

User type

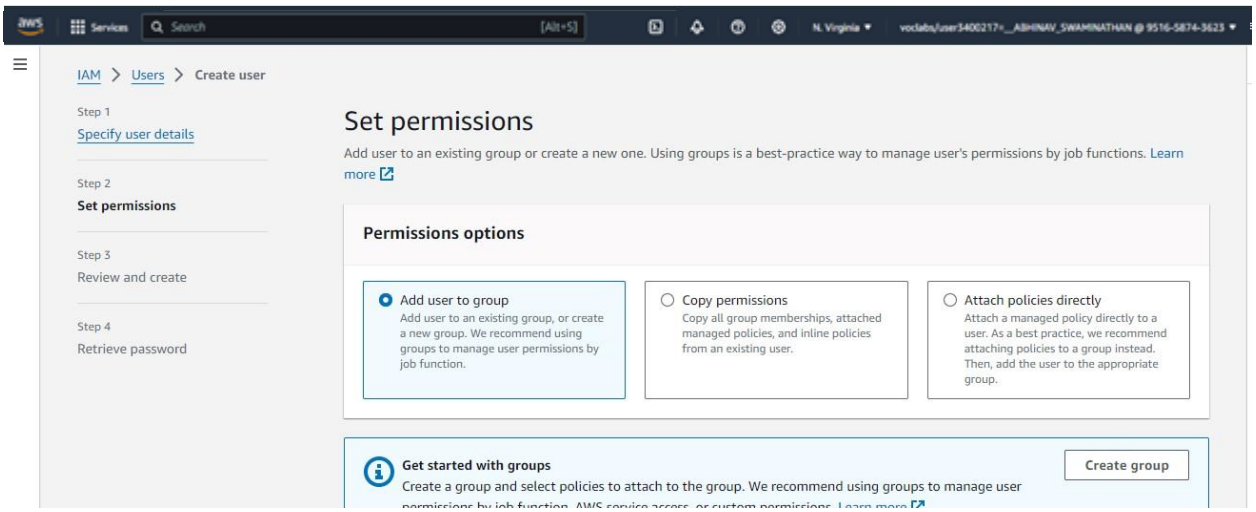
- Specify a user in Identity Center - Recommended
  We recommend that you use Identity Center to provide console access to a person. With Identity Center, you can centrally manage user access to their AWS accounts and cloud applications.
- I want to create an IAM user
  We recommend that you create IAM users only if you need to enable programmatic access through access keys, service-specific credentials for AWS CodeCommit or Amazon Keyspaces, or a backup credential for emergency account access.

Console password

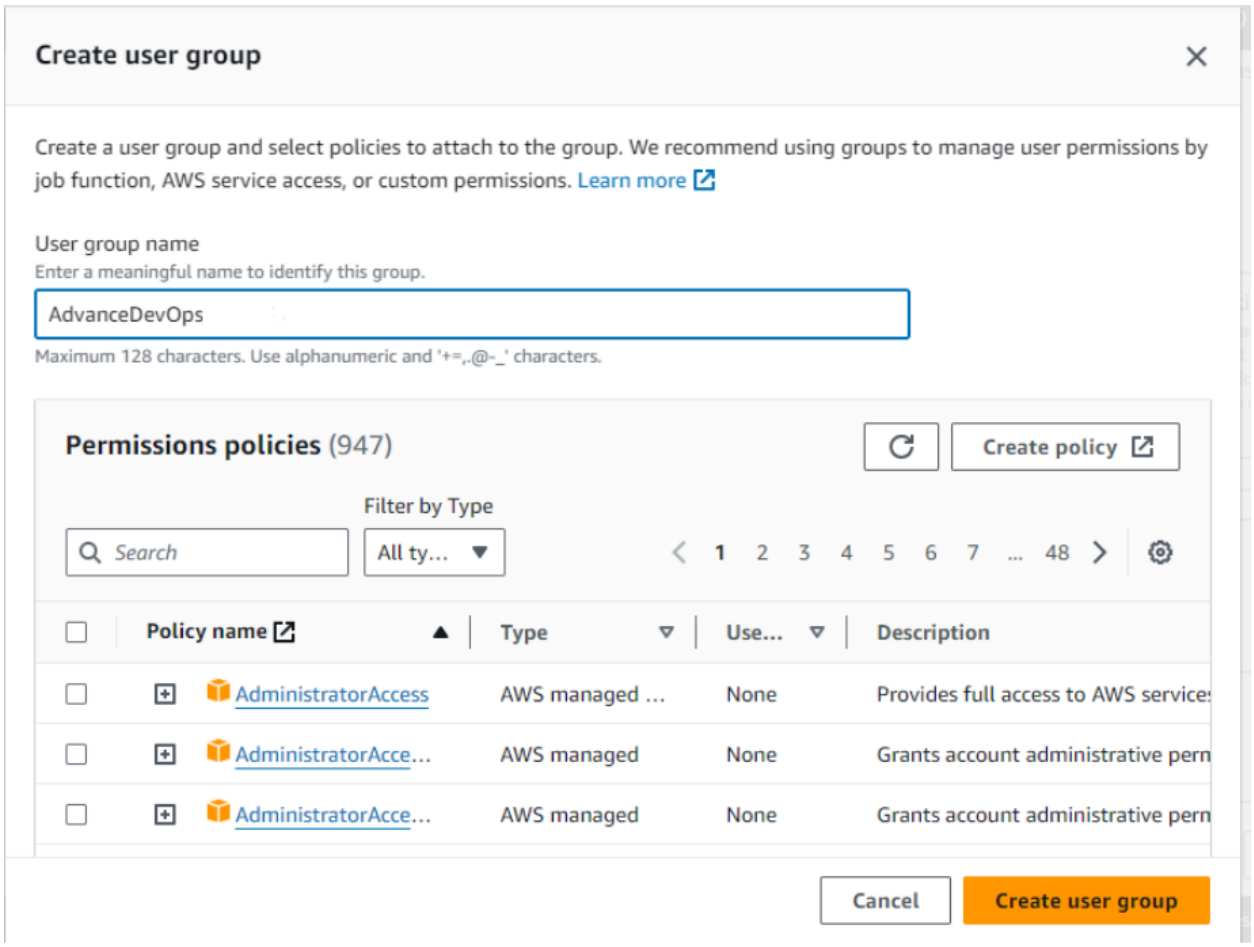
- Autogenerated password
  You can view the password after you create the user.
- Custom password
  Enter a custom password for the user.

  ••••••••••

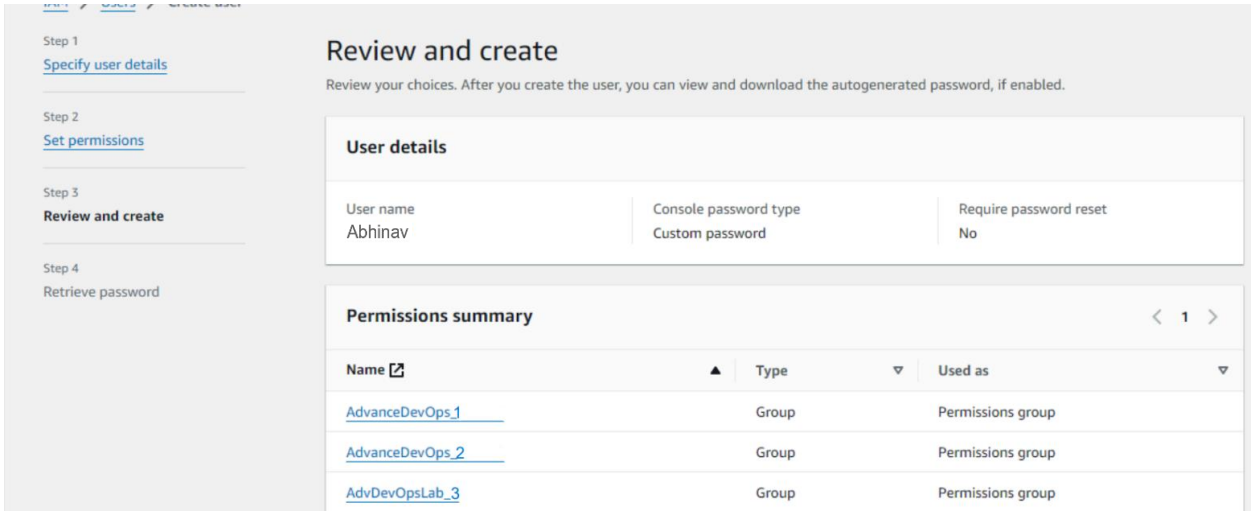
  - Must be at least 8 characters long



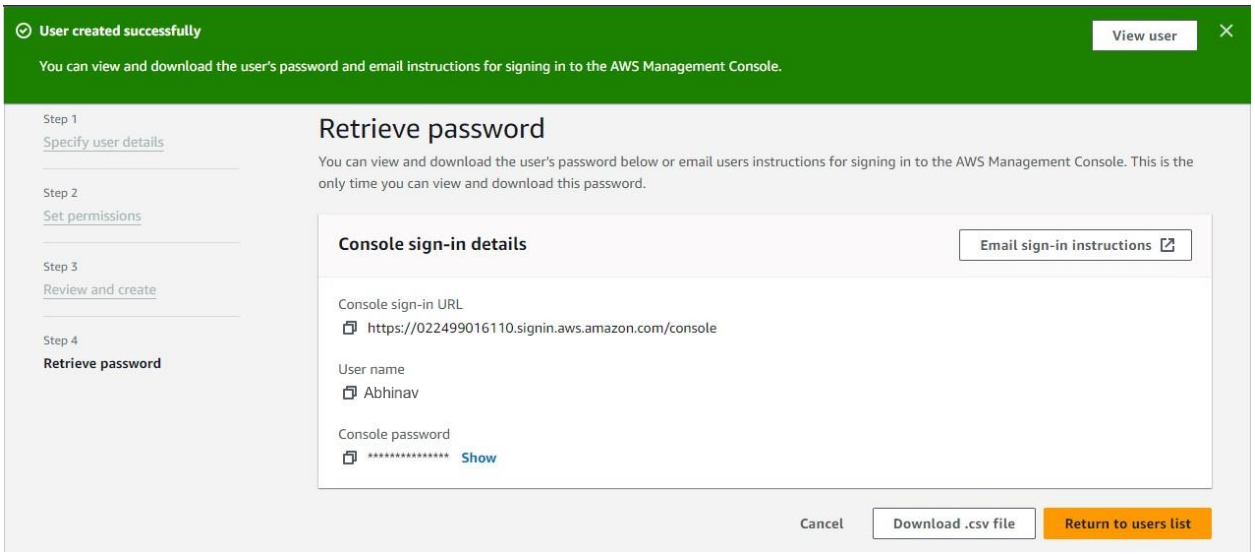
Next click on add user to group. If you do not have an existing group, select create group. Then Give the group name and policies if required and create a group.



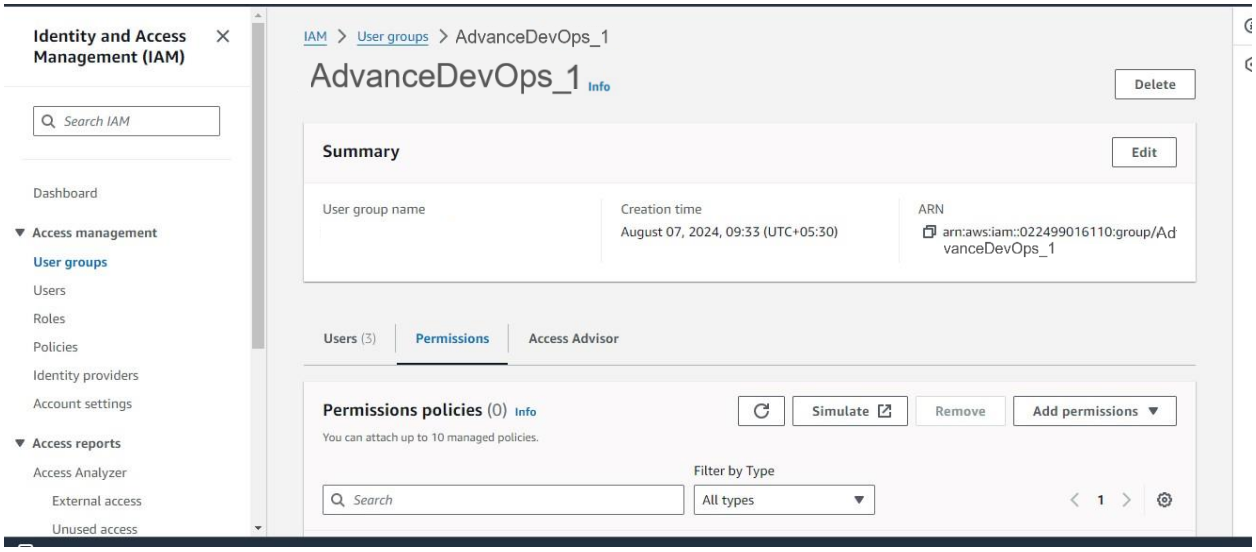
Once the group is created, select the group in which the user should be added.



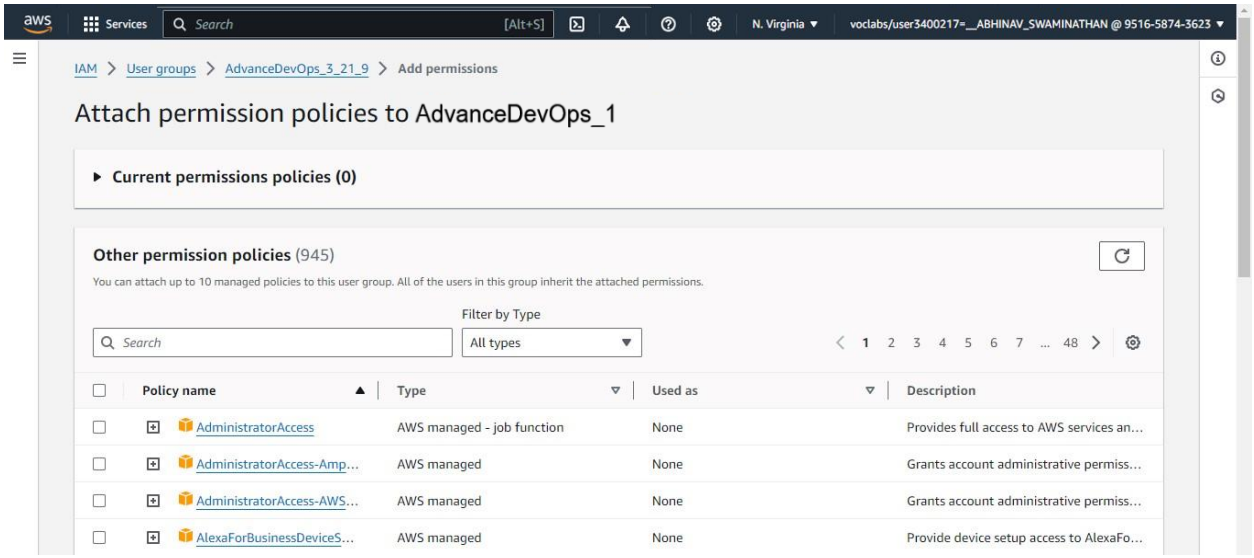
Recheck all the configuration and details of the user and click on create user. Then you will this page.

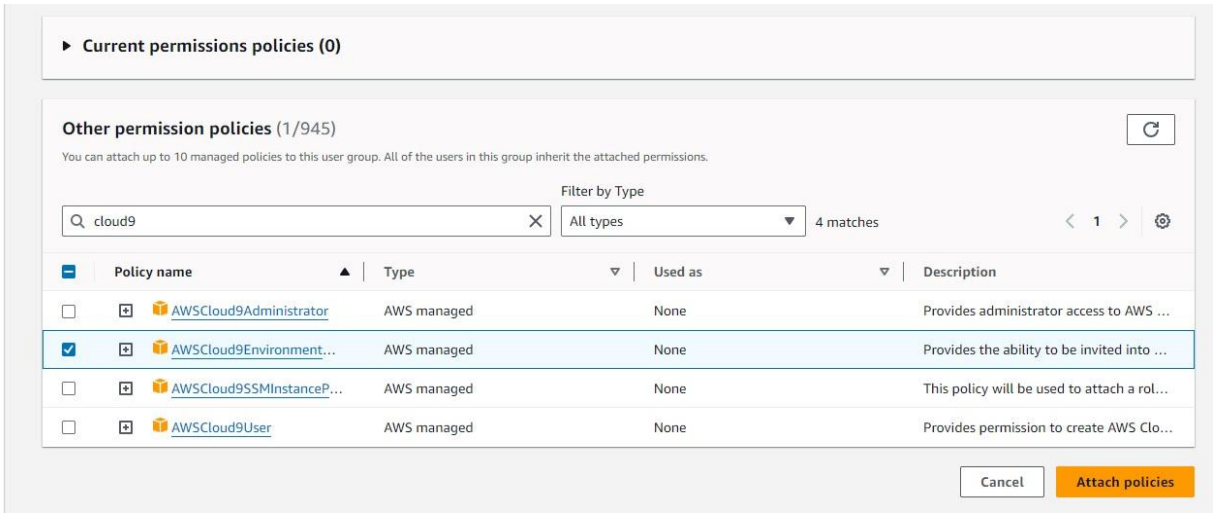


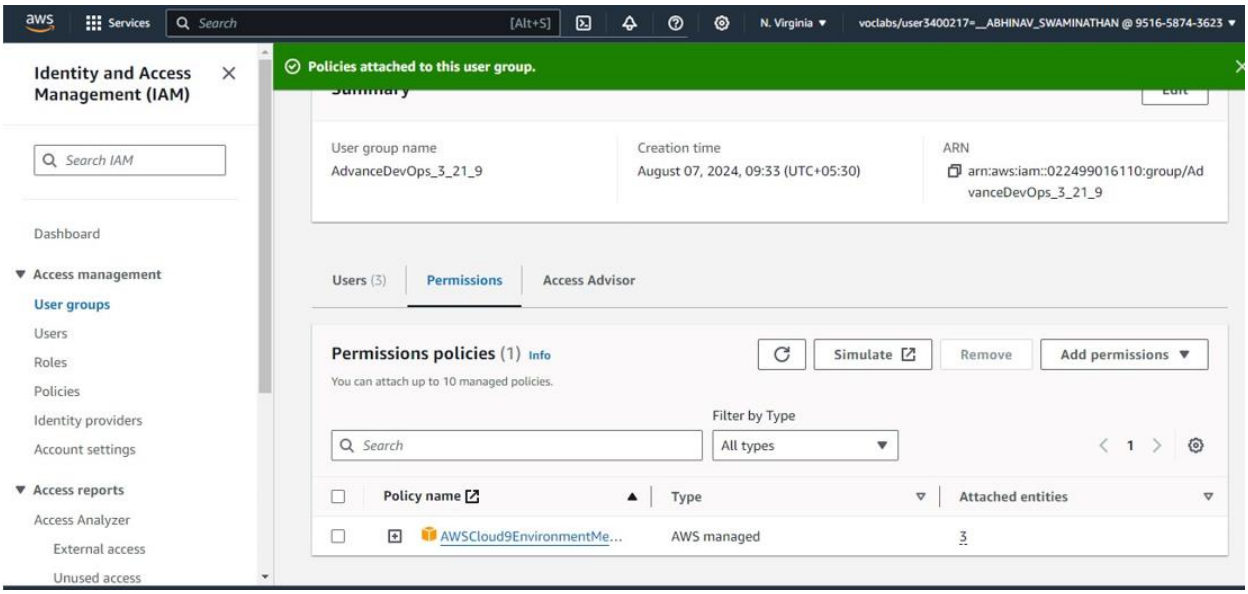
After creation of the user, go to the user groups tab and select the group in which the user has been added. Navigate to the permissions tab. Click on add permissions and attach policy.



Search for the “AWSCloud9EnvironmentMember” policy and attach it.





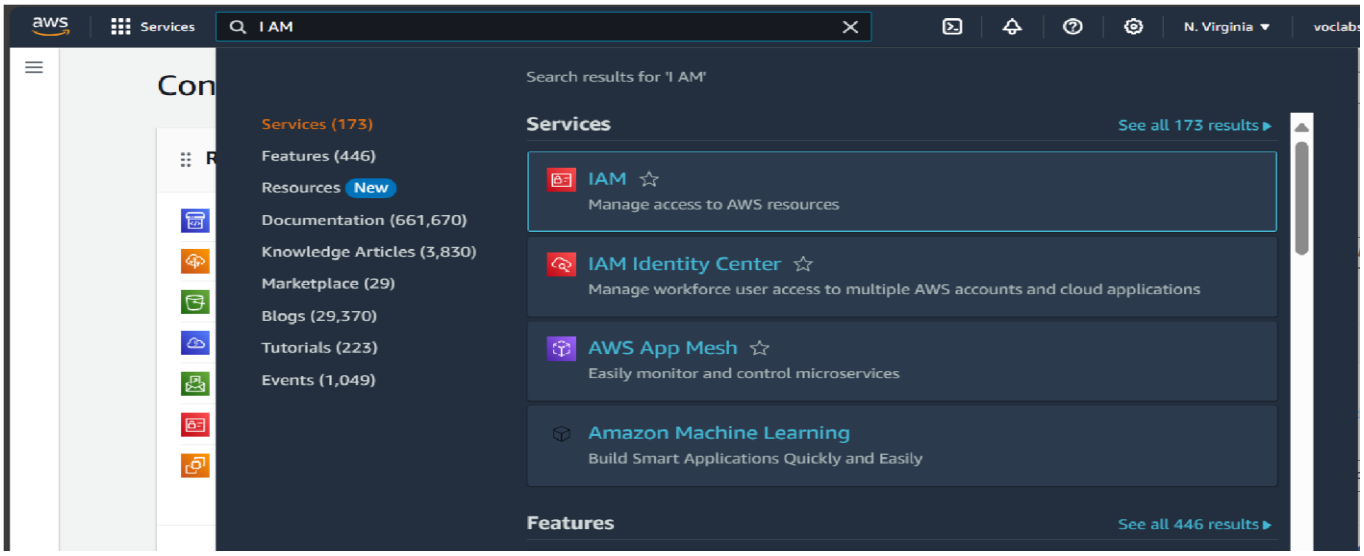


Name: Abhinav Swaminathan **D15C** **Roll No: 01**

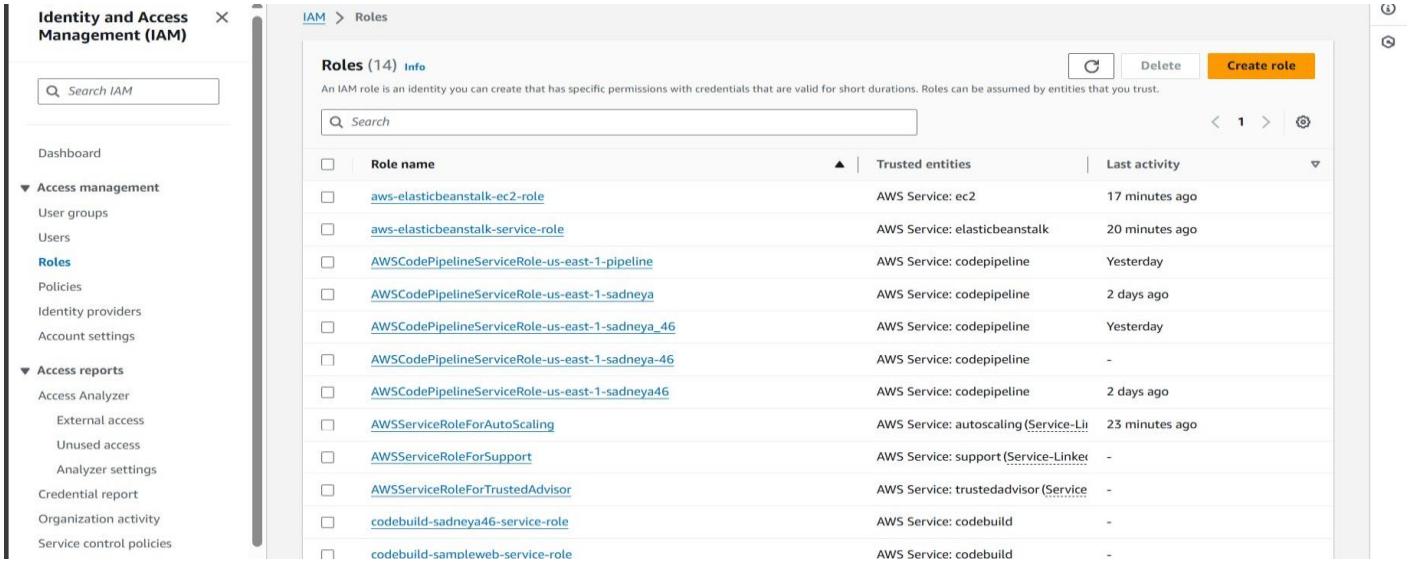
Experiment No: 2

Step1 :- Creation of role:

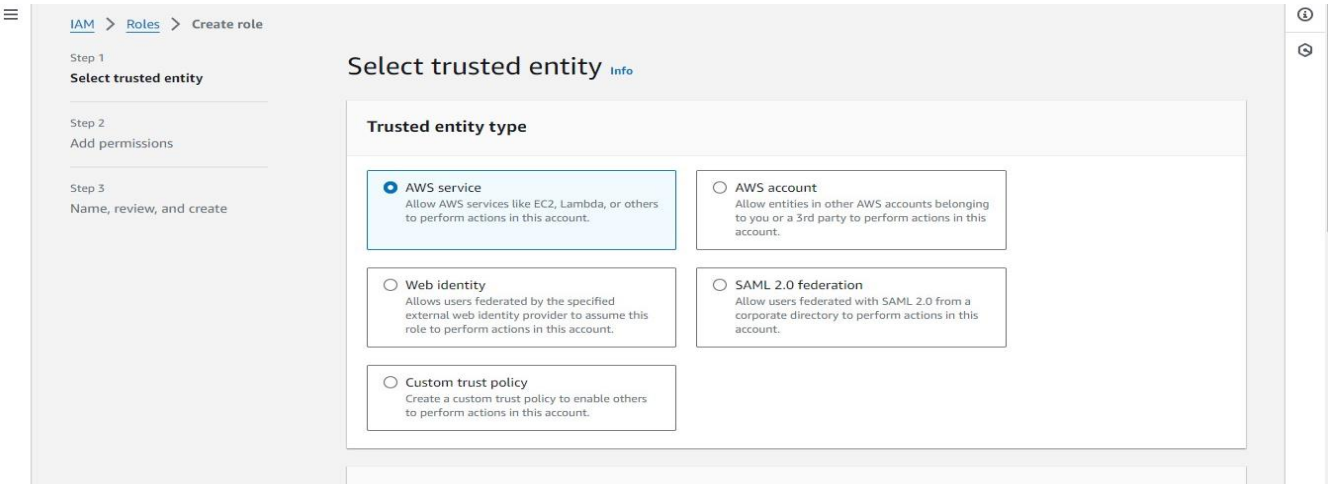
1. Login to your AWS account and search for IAM



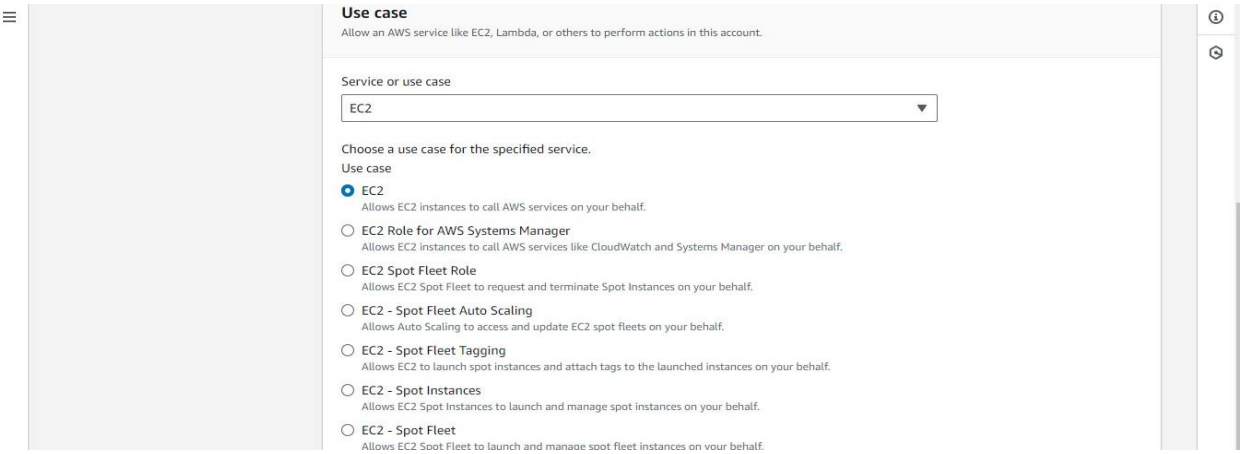
2. Then go into the role section and click on create role.



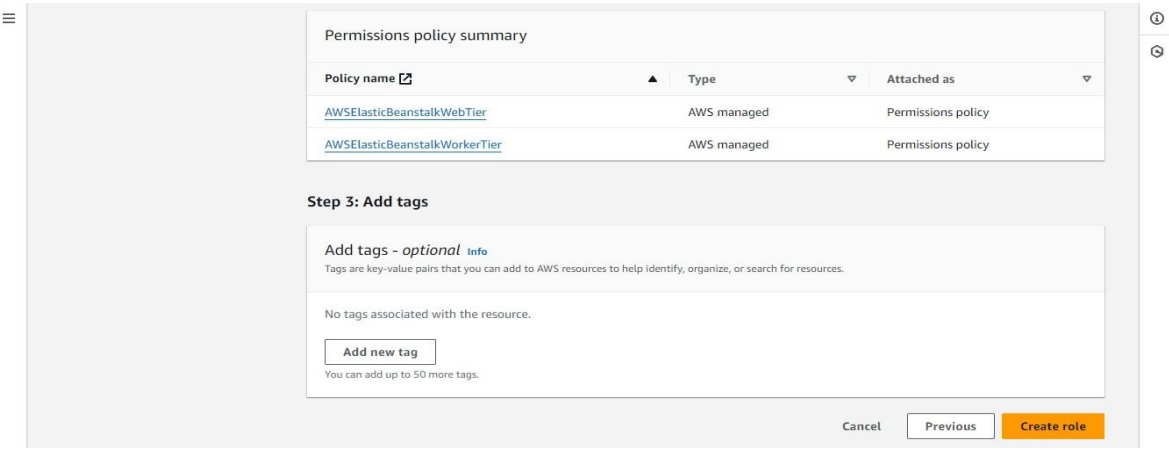
3. Then select a trusted entity as AWS service.



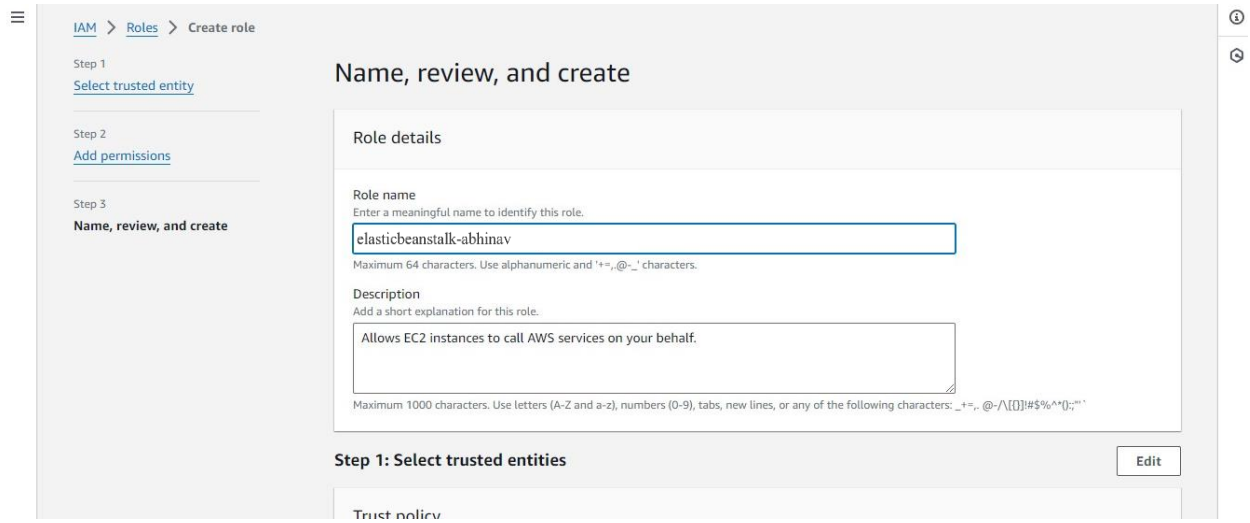
4. Select use case as EC2.



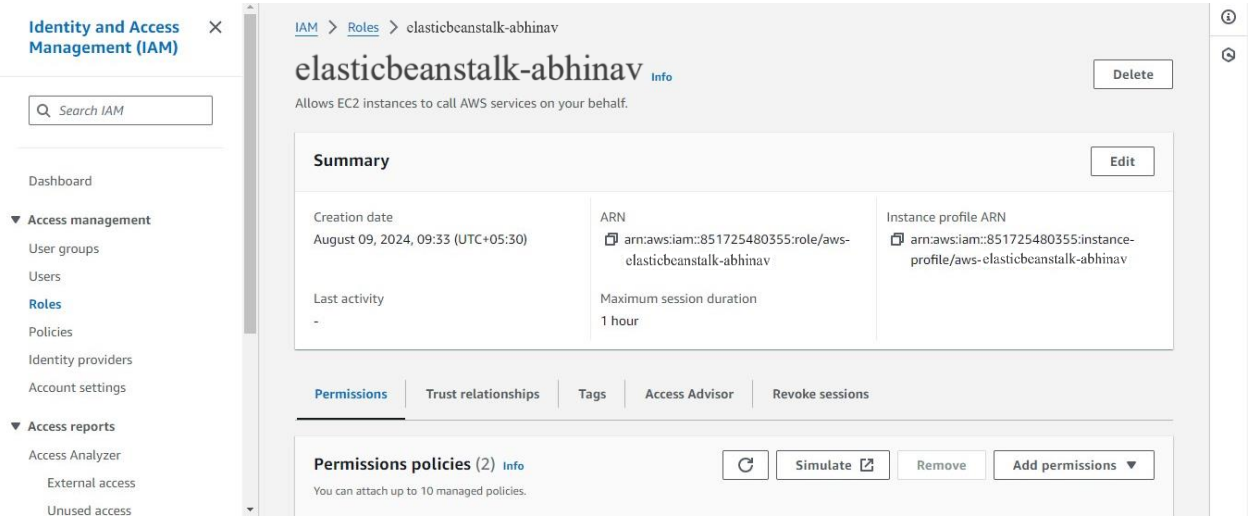
5. Select permissions as AWS Elastic Beanstalk Web Tier and AWS elastic Beanstalk worker tier.



6. Give a name to Role.

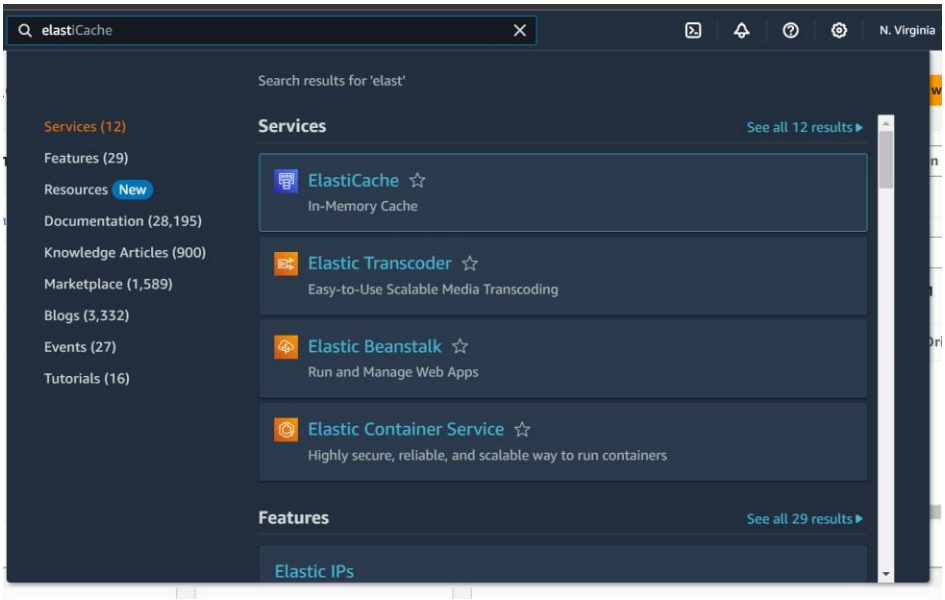


7. Then the role gets created

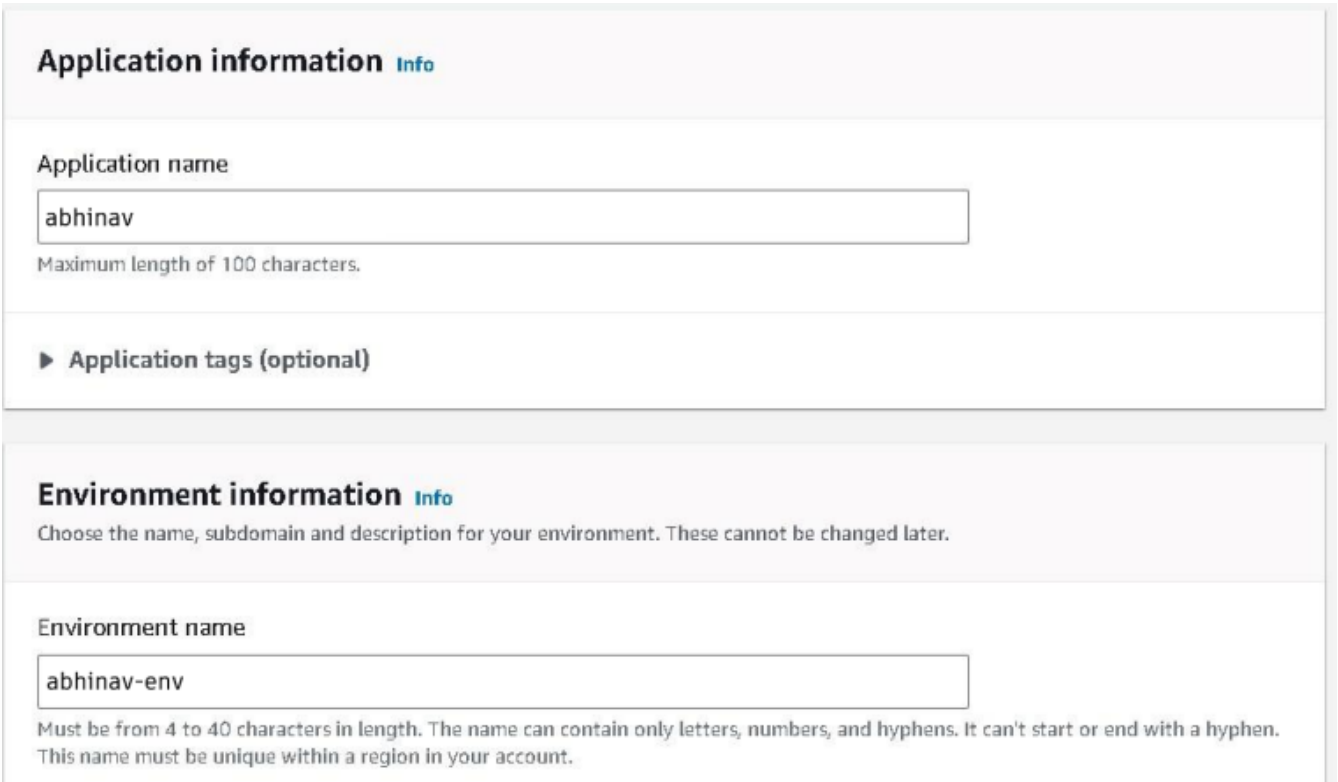


Step 2 :- Creation Elastic Beanstalk Environment

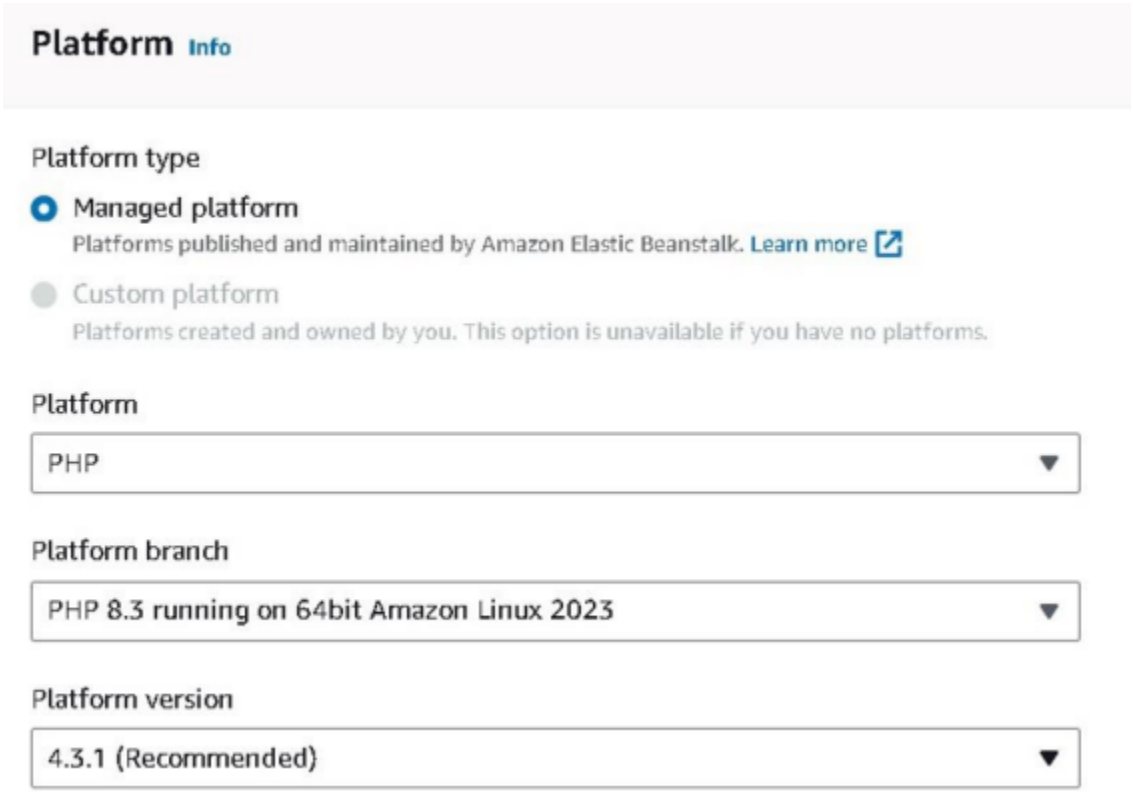
1. search for Elastic Beanstalk in the search box.



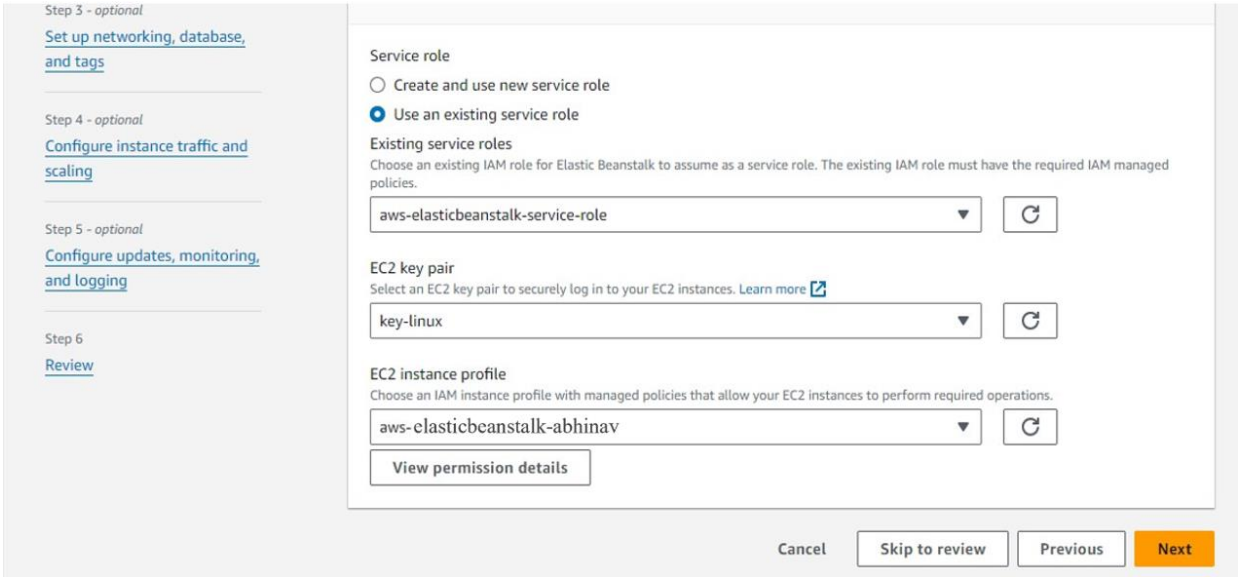
2. Open up Elastic Beanstalk and name your web app.



3. Select platform as PHP.

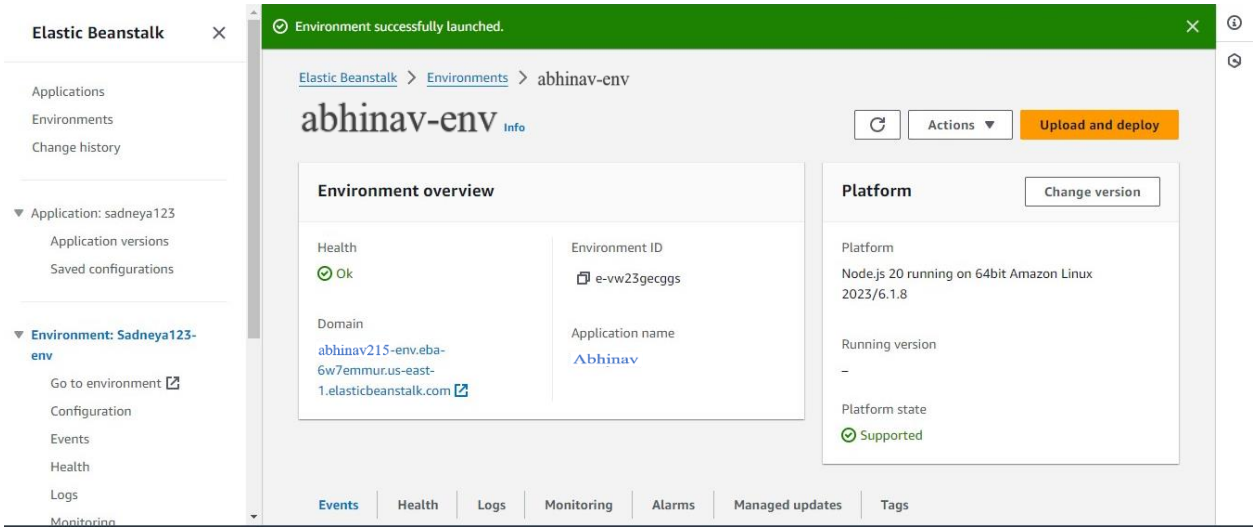


4. After clicking on next you need to select the use existing role. Then you will see the existing role select it like here it is aws-elasticbeanstalk-service-role. Which we created in 1st part. Select role, then select key you have created then profile will be automatically selected according to role. then click on create application by keeping all the remaining settings as it is.



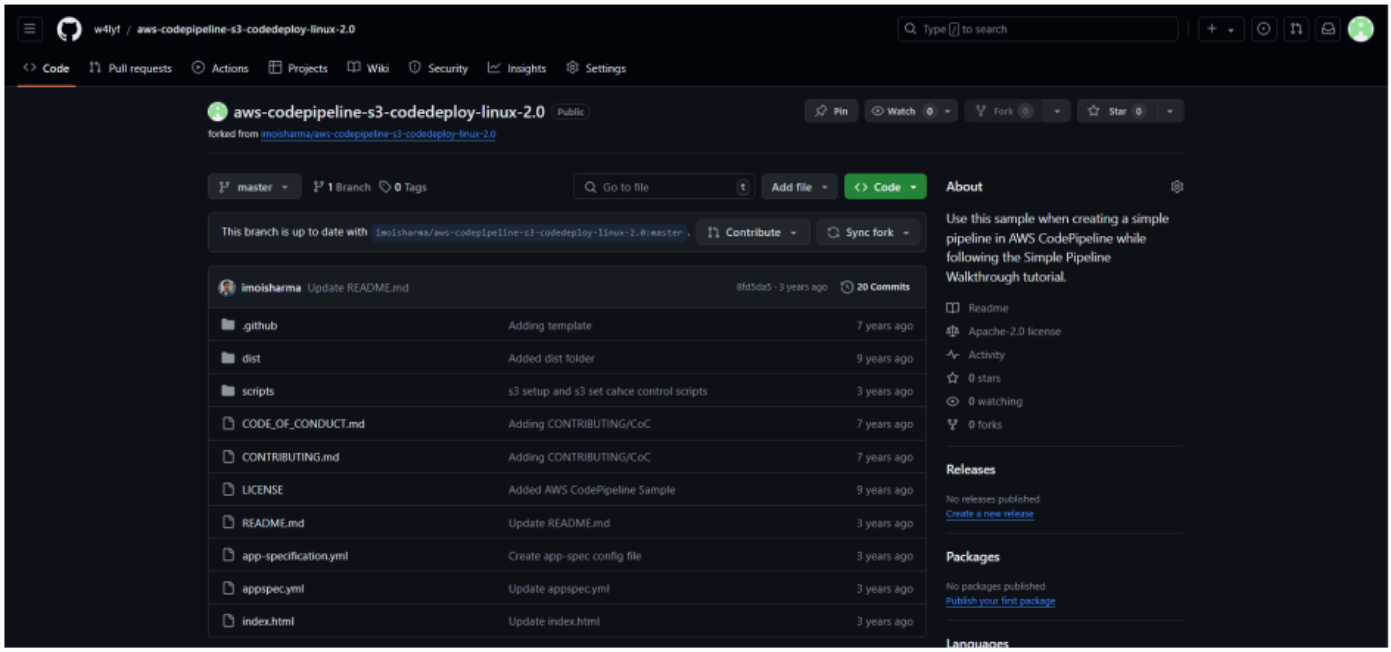
Keep Set up networking, database and tags, Configure instance traffic and scaling, Configure updates, monitoring and logging all these default.

5. Beanstalk creates a sample environment for you to deploy your application. By default, it creates an EC2 instance, a security group, an Auto Scaling group, an Amazon S3 Bucket, Amazon CloudWatch alarms and a domain name for your Application.



Step 3: Get a copy of your sample code

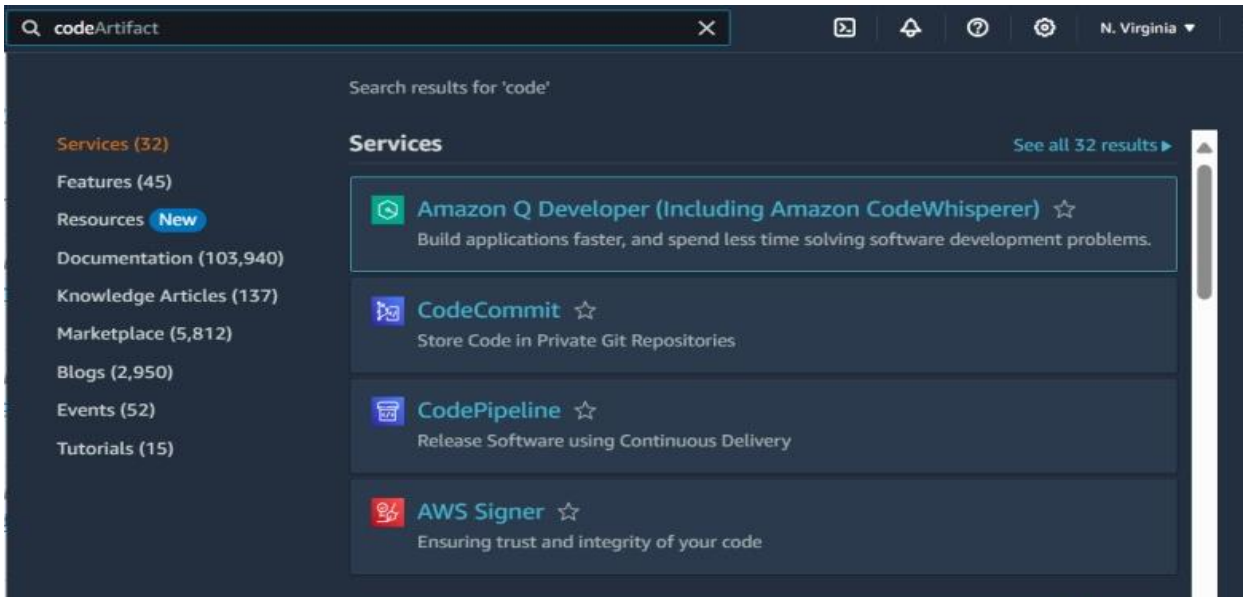
In this step, we will get the sample code from this GitHub Repository to later host it. The pipeline takes code from the source and then performs actions on it.
For this experiment, as a source, we will use this forked GitHub repository. We can alternatively also use Amazon S3 and AWS CodeCommit.
Go to the repository shared above and simply fork it.



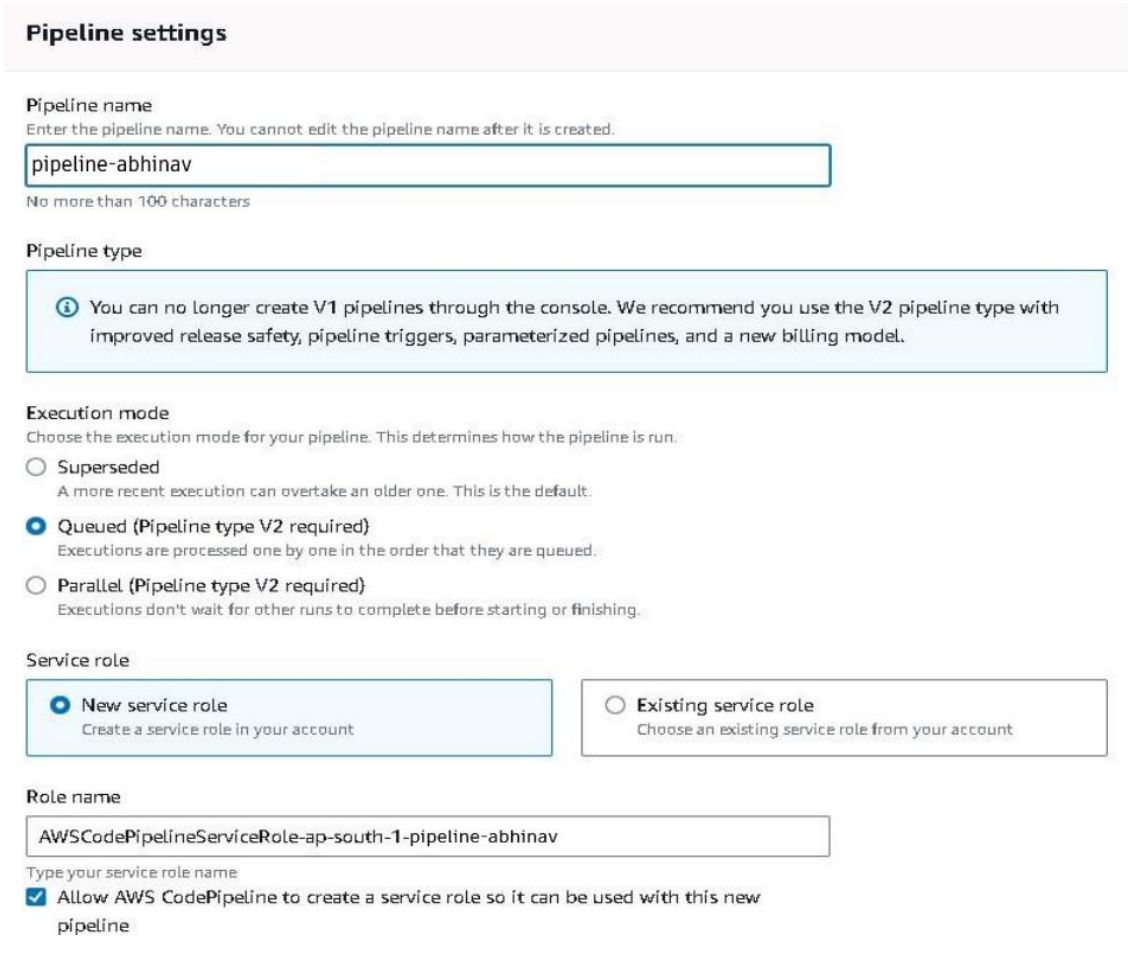
Step 4: Creating a CodePipeline

In this step, we'll create a simple pipeline that has its source and deployment information. In this case, however, we will skip the build stage where you get to plug in our preferred build provider.

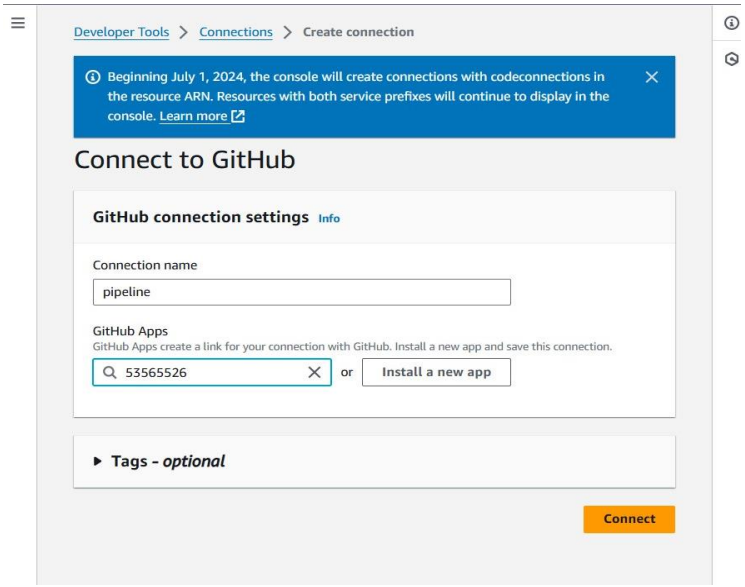
1. Search CodePipeline in the search bar and click on create a new Pipeline.

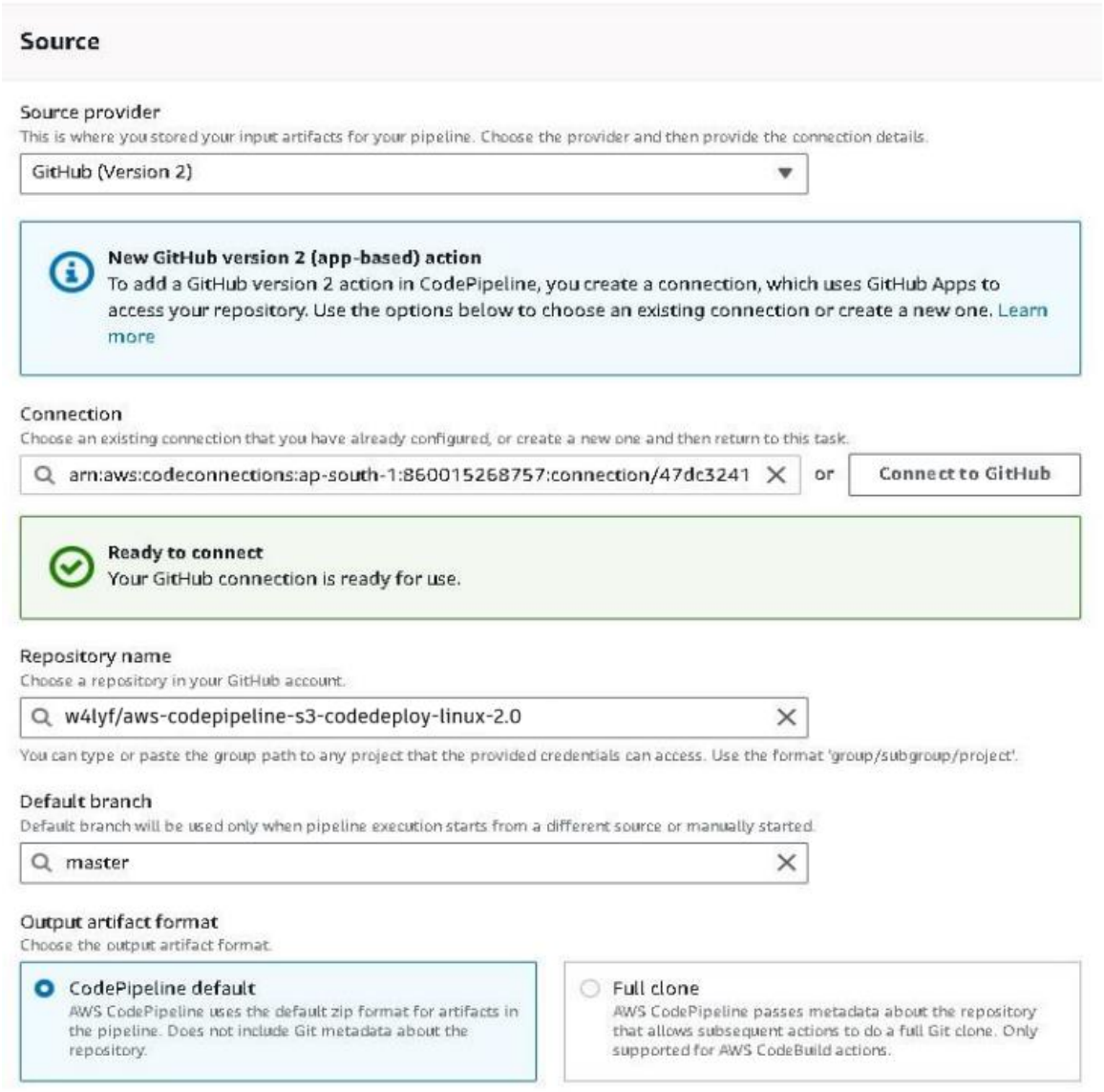


2. Give a name to your pipeline.

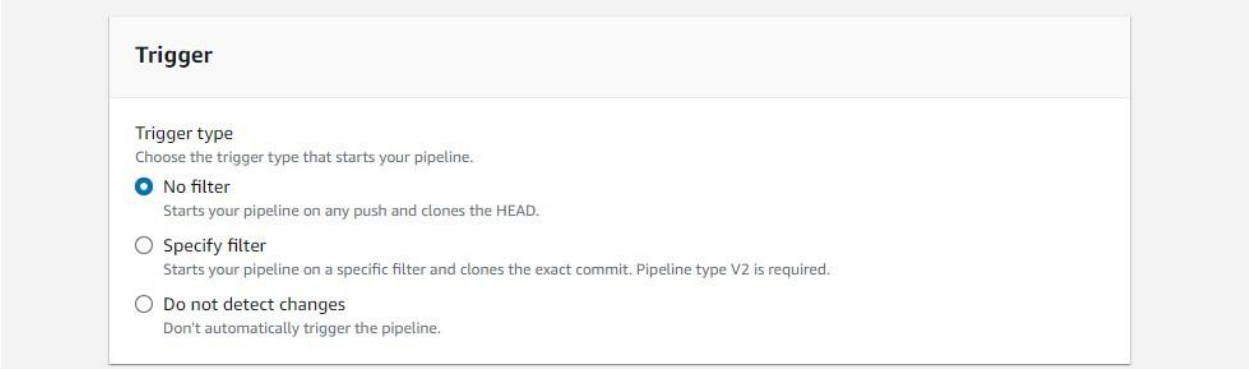


3. In the source stage, choose GitHub v2 as the provider, then connect your GitHub account to AWS by creating a connection. You'd need your GitHub credentials and then you'd need to authorize and install AWS on the forked GitHub Repository.





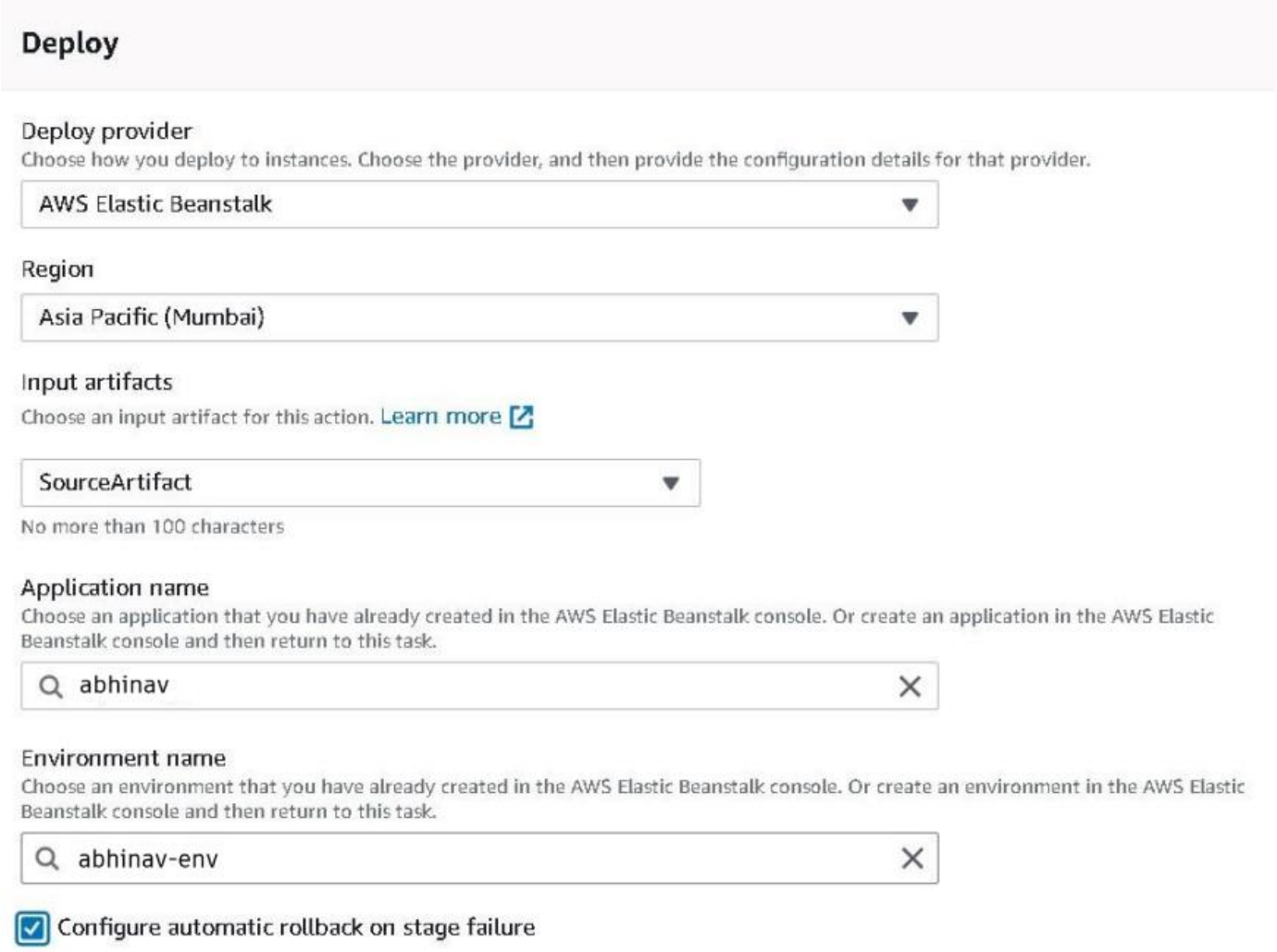
4. Then select trigger type none.



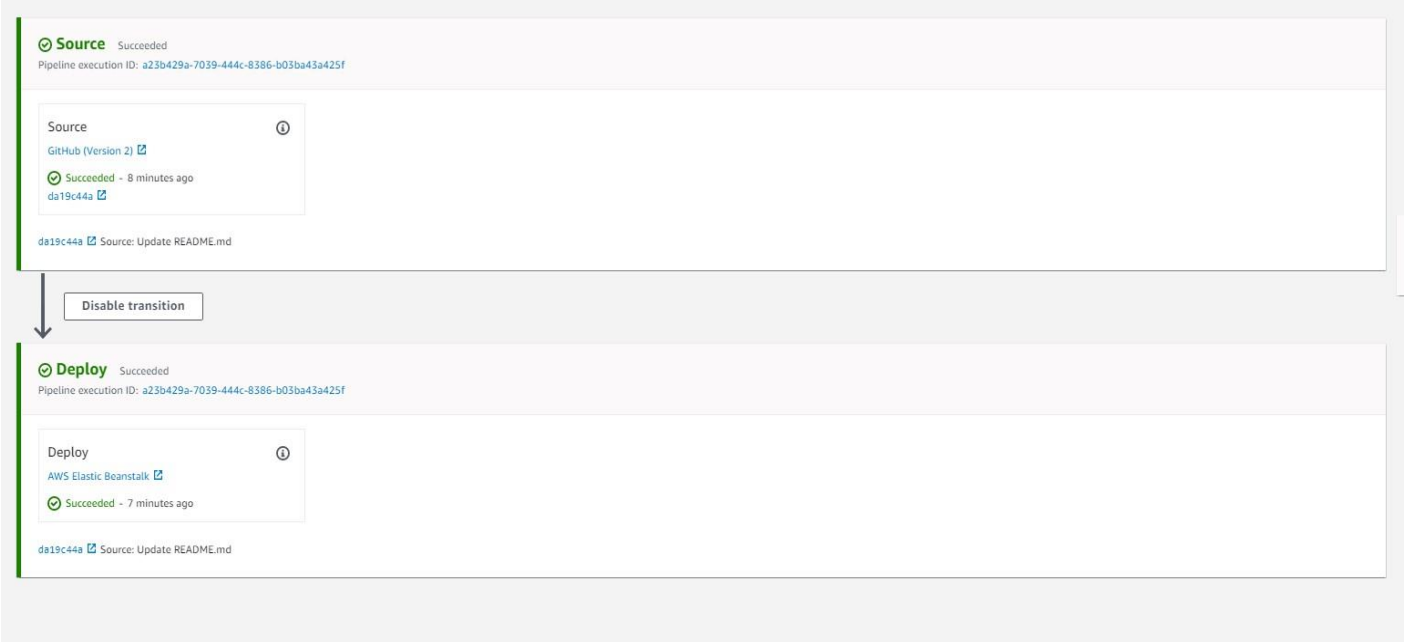
After that, click Continue and skip the build stage. Proceed to the Deployment stage.

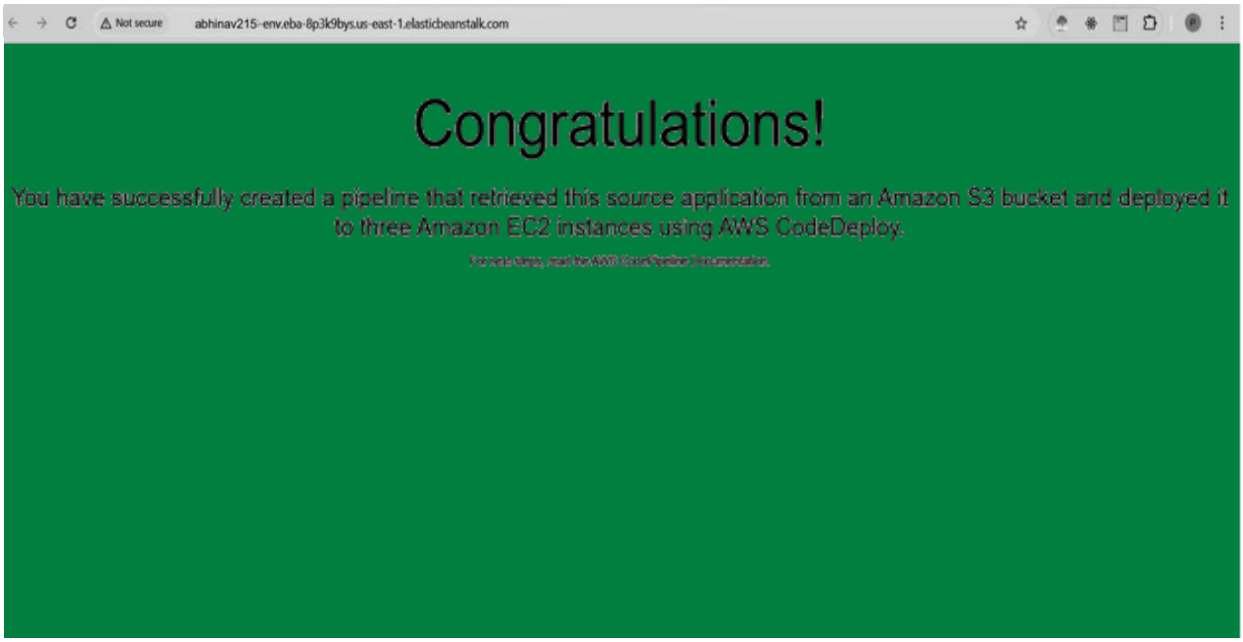
Step 5: Deployment

1. Choose Beanstalk as the Deploy Provider, same region as the Bucket and Beanstalk, name and environment name.



2. Then it will give you this result on screen. i.e. deployed successfully.

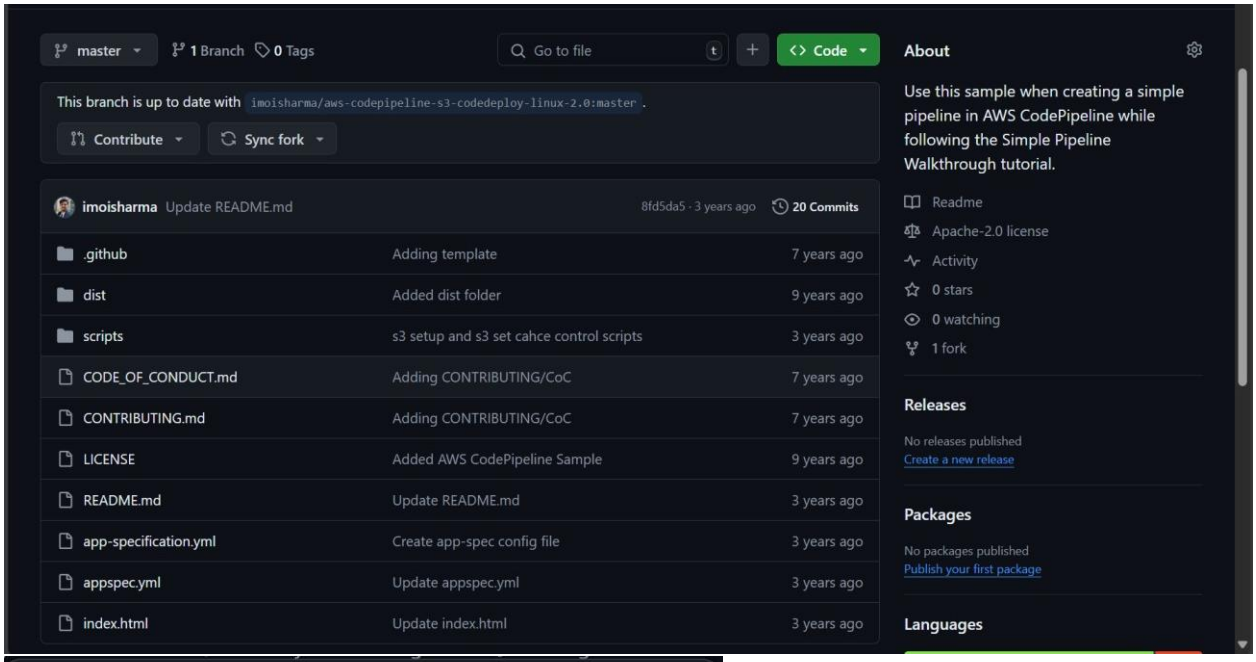


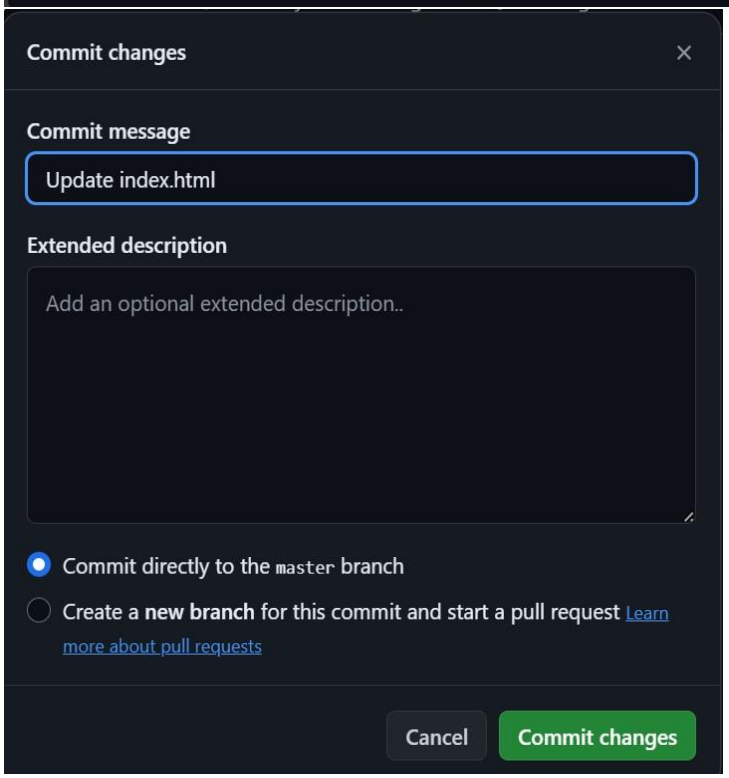


If you can see this, that means that you successfully created an automated software using CodePipeline.

Step 6: Committing changes to update app

1. In this we make some changes in the file. Open github.com then open the forked repository. Then update the changes in the index.html file and finally commit those changes.





2. Then again start the deployment of the pipeline. And check the changes in the website

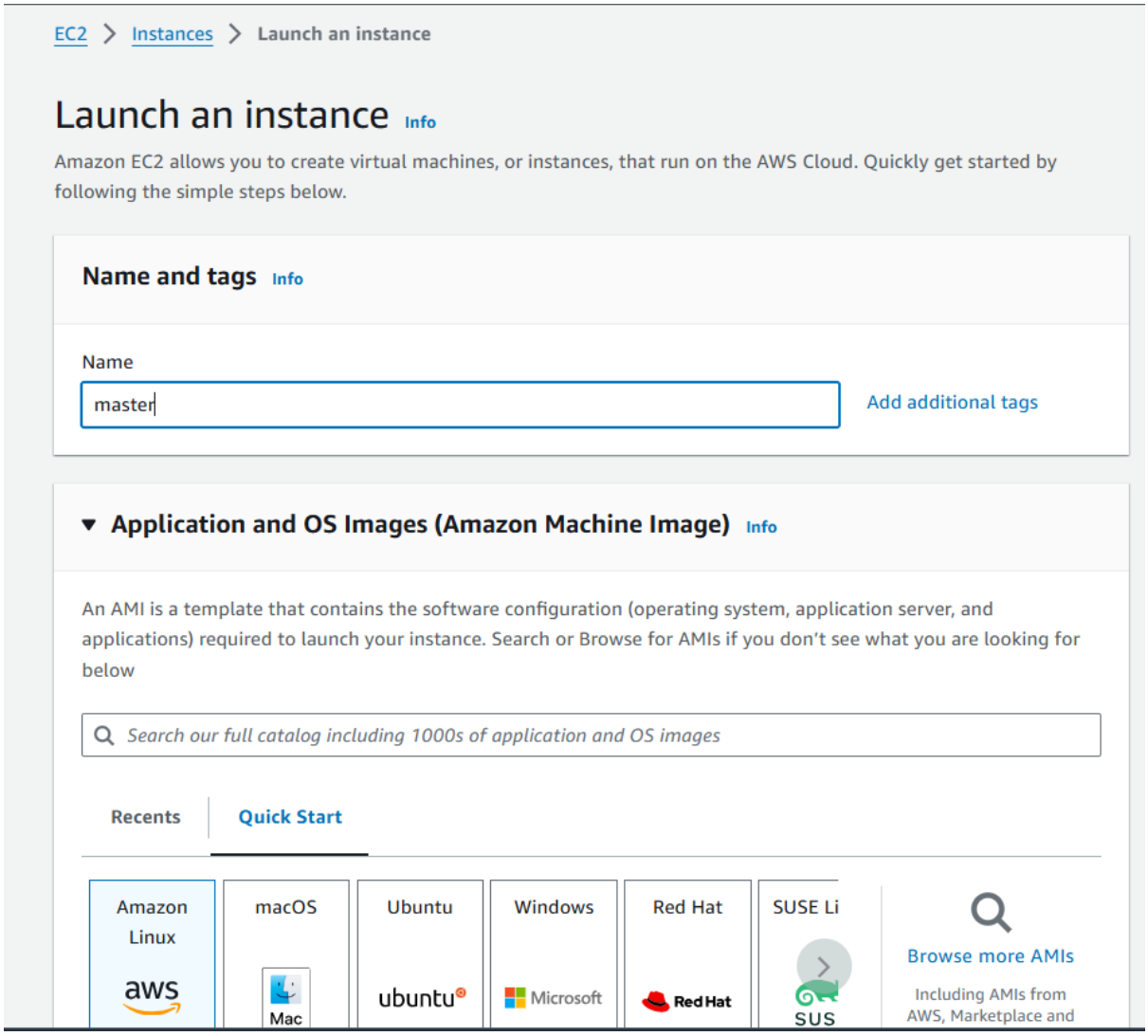
Name: Abhinav Swaminathan **D15C** **Roll No: 01**

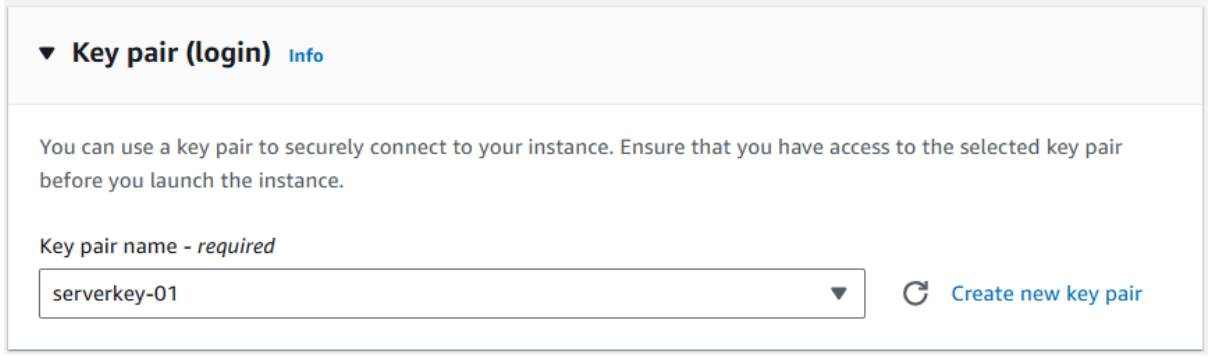
Experiment No. 3

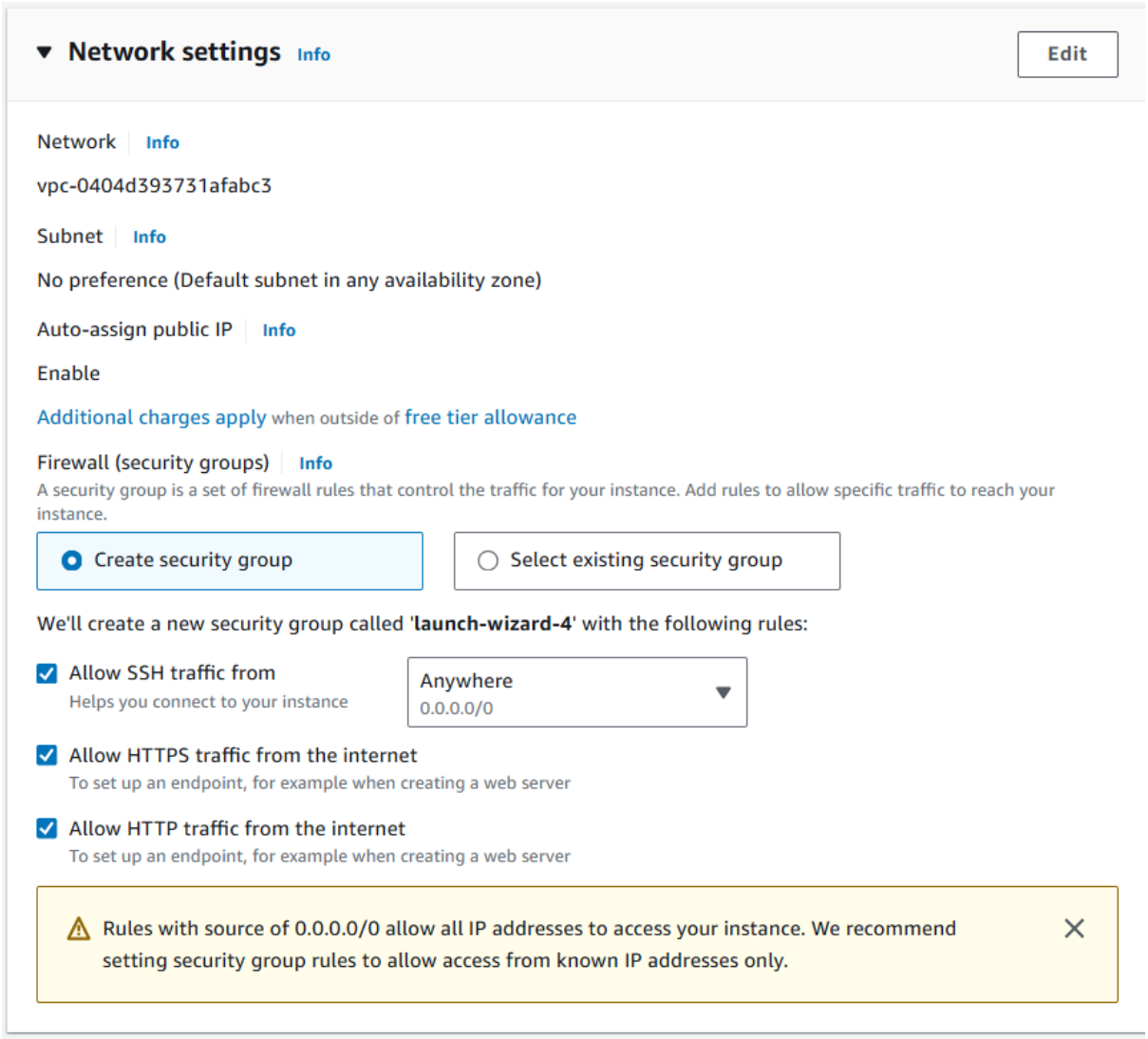
Aim: To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud.

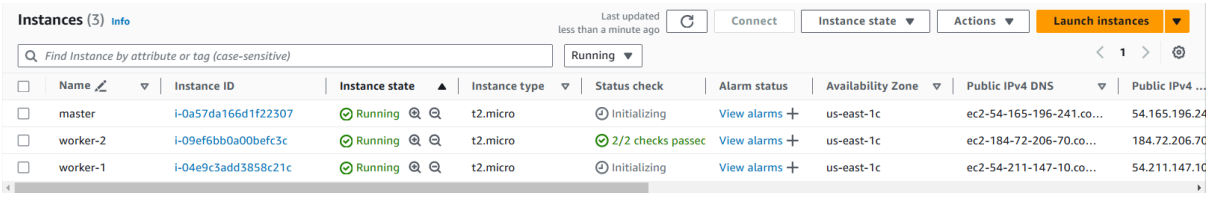
Steps:

1. **Create 3 EC2 Ubuntu Instances on AWS. (Name 1 as Master, the other 2 as worker-1 and worker-2)**



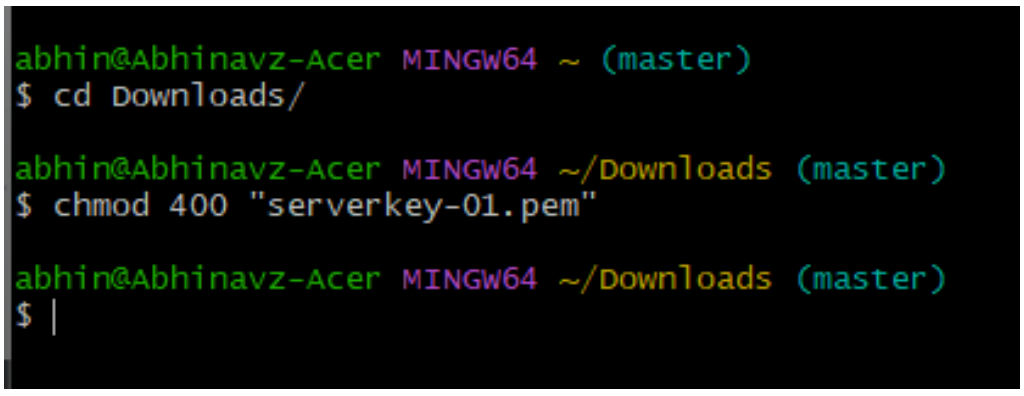






2. SSH into all 3 machines

- Give permissions to the current user to the downloaded pem file using - **chmod 400 <security_filename.pem>**



- ssh into all the three machines using –
  ssh -i (keyname).pem (username)@(public ipv4 dns address)
  where keyname is name of the key you created. (server-01.pem). Other details can be found on the Instance dashboard.

```
abhin@Abhinavz-Acer MINGW64 ~/Downloads (master)
$ ssh -i "serverkey-01.pem" ec2-user@ec2-54-165-196-241.compute-1.amazonaws.com
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-31-90-103 ~]$
```

3. Installation Of Docker on three machines

- **sudo yum install docker -y**

```
[ec2-user@ip-172-31-90-103 ~]$ sudo yum install docker -y
Last metadata expiration check: 0:13:41 ago on Sat Sep 14 03:42:27 2024.
Dependencies resolved.
================================================================================
 Package                 Arch     Version                  Repository     Size
================================================================================
Installing:
 docker                  x86_64   25.0.6-1.amzn2023.0.2    amazonlinux    44 M
Installing dependencies:
 containerd              x86_64   1.7.20-1.amzn2023.0.1    amazonlinux    35 M
 iptables-libs           x86_64   1.8.8-3.amzn2023.0.2     amazonlinux   401 k
 iptables-nft            x86_64   1.8.8-3.amzn2023.0.2     amazonlinux   183 k
 libcgroup               x86_64   3.0-1.amzn2023.0.1       amazonlinux    75 k
 libnetfilter_conntrack  x86_64   1.0.8-2.amzn2023.0.2     amazonlinux    58 k
 libnfnetlink            x86_64   1.0.1-19.amzn2023.0.2    amazonlinux    30 k
 libnftnl                x86_64   1.2.2-2.amzn2023.0.2     amazonlinux    84 k
 pigz                    x86_64   2.5-1.amzn2023.0.3       amazonlinux    83 k
 runc                    x86_64   1.1.13-1.amzn2023.0.1    amazonlinux   3.2 M

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

Total download size: 84 M
Installed size: 317 M
Downloading Packages:
(1/10): iptables-libs-1.8.8-3.amzn2023.0.2.x86_ 2.2 MB/s | 401 kB     00:00
(2/10): iptables-nft-1.8.8-3.amzn2023.0.2.x86_6 2.5 MB/s | 183 kB     00:00
(3/10): libcgroup-3.0-1.amzn2023.0.1.x86_64.rpm 1.3 MB/s |  75 kB     00:00
(4/10): libnetfilter_conntrack-1.0.8-2.amzn2023 1.3 MB/s |  58 kB     00:00
(5/10): libnfnetlink-1.0.1-19.amzn2023.0.2.x86_ 938 kB/s |  30 kB     00:00
(6/10): libnftnl-1.2.2-2.amzn2023.0.2.x86_64.rp 1.6 MB/s |  84 kB     00:00
(7/10): pigz-2.5-1.amzn2023.0.3.x86_64.rpm      1.7 MB/s |  83 kB     00:00
(8/10): runc-1.1.13-1.amzn2023.0.1.x86_64.rpm    21 MB/s | 3.2 MB     00:00
(9/10): containerd-1.7.20-1.amzn2023.0.1.x86_64  31 MB/s |  35 MB     00:01
(10/10): docker-25.0.6-1.amzn2023.0.2.x86_64.rp  28 MB/s |  44 MB     00:01
--------------------------------------------------------------------------------
Total                                            52 MB/s |  84 MB     00:01
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :
  Installing       : runc-1.1.13-1.amzn2023.0.1.x86_64
  Installing       : containerd-1.7.20-1.amzn2023.0.1.x86_64
  Running scriptlet: containerd-1.7.20-1.amzn2023.0.1.x86_64
  Installing       : pigz-2.5-1.amzn2023.0.3.x86_64
  Installing       : libnftnl-1.2.2-2.amzn2023.0.2.x86_64
  Installing       : libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64
  Installing       : libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64
  Installing       : iptables-libs-1.8.8-3.amzn2023.0.2.x86_64
  Installing       : iptables-nft-1.8.8-3.amzn2023.0.2.x86_64
```

```
Installed:
  containerd-1.7.20-1.amzn2023.0.1.x86_64                docker-25.0.6-1.amzn2023.0.2.x86_64
  libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64     libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64

Complete!
[ec2-user@ip-172-31-90-103 ~]$
```

- Configure cgroup in a daemon.json
  (this can be done by creating the file and using **nano** text editor)

```
{
        "exec-opts": ["native.cgroupdriver=systemd"],
        "log-driver": "json-file",
        "log-opts": {
        "max-size": "100m"
        },
        "storage-driver": "overlay2"
}
```

```
[ec2-user@ip-172-31-90-103 docker]$ sudo nano daemon.json
[ec2-user@ip-172-31-90-103 docker]$
```

```
ec2-user@ip-172-31-90-103:/etc/docker
  GNU nano 5.8                          daemon.json                          Modified
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line
```

- Enable and start docker and also load the daemon.json
  sudo systemctl enable docker
  sudo systemctl daemon-reload
  sudo systemctl restart docker

```
[ec2-user@ip-172-31-90-103 docker]$ sudo systemctl enable docker
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /us
r/lib/systemd/system/docker.service.
[ec2-user@ip-172-31-90-103 docker]$ sudo systemctl daemon-reload
[ec2-user@ip-172-31-90-103 docker]$
sudo systemctl restart docker
[ec2-user@ip-172-31-90-103 docker]$
```

- Check if docker is installed

```
[ec2-user@ip-172-31-90-103 docker]$ docker --version
Docker version 25.0.5, build 5dc9bcc
[ec2-user@ip-172-31-90-103 docker]$
```

4. Install Kubernetes on all 3 machines

- SELinux needs to be disabled before configuring kubelet
  sudo setenforce 0
  sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config

```
[ec2-user@ip-172-31-90-103 docker]$ sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
[ec2-user@ip-172-31-90-103 docker]$
```

- Add Kubernetes using the repo
  (this is done by creating **kubernetes.repo** file in **/etc/yum.repos.d** and configuring it using **nano** editor)
  [kubernetes]
  name=Kubernetes
  baseurl=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/
  enabled=1
  gpgcheck=1
  gpgkey=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/repodata/repomd.xml.key
  exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni

```
[ec2-user@ip-172-31-90-103 docker]$ cd /etc/yum.repos.d/
[ec2-user@ip-172-31-90-103 yum.repos.d]$ ls
amazonlinux.repo  kernel-livepatch.repo
[ec2-user@ip-172-31-90-103 yum.repos.d]$ sudo nano kubernetes.repo
[ec2-user@ip-172-31-90-103 yum.repos.d]$ ls
amazonlinux.repo  kernel-livepatch.repo  kubernetes.repo
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

```
ec2-user@ip-172-31-90-103:/etc/yum.repos.d
 GNU nano 5.8                    kubernetes.repo                    Modified
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line
```

- Update packages list using **sudo yum update**

```
[ec2-user@ip-172-31-90-103 yum.repos.d]$ sudo yum update
Kubernetes                                      125 kB/s |  17 kB     00:00
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

- Install kubelet kubeadm kubectl

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

```
[ec2-user@ip-172-31-90-103 yum.repos.d]$ sudo yum install -y kubelet kubeadm kub
ectl --disableexcludes=kubernetes
Last metadata expiration check: 0:00:42 ago on Sat Sep 14 04:08:20 2024.
Dependencies resolved.
================================================================================
 Package                 Arch     Version                 Repository     Size
================================================================================
Installing:
 kubeadm                 x86_64   1.30.5-150500.1.1       kubernetes     10 M
 kubectl                 x86_64   1.30.5-150500.1.1       kubernetes     10 M
 kubelet                 x86_64   1.30.5-150500.1.1       kubernetes     17 M
Installing dependencies:
 conntrack-tools         x86_64   1.4.6-2.amzn2023.0.2    amazonlinux   208 k
 cri-tools               x86_64   1.30.1-150500.1.1       kubernetes    8.6 M
 kubernetes-cni          x86_64   1.4.0-150500.1.1        kubernetes    6.7 M
 libnetfilter_cthelper   x86_64   1.0.0-21.amzn2023.0.2   amazonlinux    24 k
 libnetfilter_cttimeout  x86_64   1.0.0-19.amzn2023.0.2   amazonlinux    24 k
 libnetfilter_queue      x86_64   1.0.5-2.amzn2023.0.2    amazonlinux    30 k

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

Total download size: 53 M
Installed size: 292 M
Downloading Packages:
(1/9): libnetfilter_cttimeout-1.0.0-19.amzn2023 448 kB/s |  24 kB     00:00
(2/9): libnetfilter_cthelper-1.0.0-21.amzn2023. 409 kB/s |  24 kB     00:00
(3/9): libnetfilter_queue-1.0.5-2.amzn2023.0.2. 1.5 MB/s |  30 kB     00:00
(4/9): conntrack-tools-1.4.6-2.amzn2023.0.2.x86 1.8 MB/s | 208 kB     00:00
(5/9): cri-tools-1.30.1-150500.1.1.x86_64.rpm    28 MB/s | 8.6 MB     00:00
(6/9): kubectl-1.30.5-150500.1.1.x86_64.rpm      23 MB/s |  10 MB     00:00
(7/9): kubeadm-1.30.5-150500.1.1.x86_64.rpm      18 MB/s |  10 MB     00:00
(8/9): kubelet-1.30.5-150500.1.1.x86_64.rpm      37 MB/s |  17 MB     00:00
(9/9): kubernetes-cni-1.4.0-150500.1.1.x86_64.r  20 MB/s | 6.7 MB     00:00
--------------------------------------------------------------------------------
Total                                            56 MB/s |  53 MB     00:00
```

```
Installed:
  conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64
  cri-tools-1.30.1-150500.1.1.x86_64
  kubeadm-1.30.5-150500.1.1.x86_64
  kubectl-1.30.5-150500.1.1.x86_64
  kubelet-1.30.5-150500.1.1.x86_64
  kubernetes-cni-1.4.0-150500.1.1.x86_64
  libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64
  libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64
  libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64

Complete!
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

- 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

```
[ec2-user@ip-172-31-90-103 yum.repos.d]$ . sudo swapoff -a
-bash: sudo: No such file or directory
[ec2-user@ip-172-31-90-103 yum.repos.d]$ sudo swapoff -a
[ec2-user@ip-172-31-90-103 yum.repos.d]$ echo "net.bridge.bridge-nf-call-iptable
s=1" | sudo tee -a /etc/sysctl.conf
net.bridge.bridge-nf-call-iptables=1
[ec2-user@ip-172-31-90-103 yum.repos.d]$ sudo sysctl -p
net.bridge.bridge-nf-call-iptables = 1
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

5. **Perform this ONLY on the Master machine Initialize the Kubecluster**
   sudo kubeadm init --podnetwork-cidr=10.244.0.0/16

```
[ec2-user@ip-172-31-90-103 yum.repos.d]$ sudo kubeadm init --pod-network-cidr=10
.244.0.0/16 --ignore-preflight-errors=all
I0914 04:12:17.448521   27990 version.go:256] remote version is much newer: v1.3
1.0; falling back to: stable-1.30
[init] Using Kubernetes version: v1.30.4
[preflight] Running pre-flight checks
        [WARNING NumCPU]: the number of available CPUs 1 is less than the requir
ed 2
        [WARNING Mem]: the system RAM (949 MB) is less than the minimum 1700 MB
        [WARNING FileExisting-socat]: socat not found in system path
        [WARNING FileExisting-tc]: tc not found in system path
        [WARNING Service-Kubelet]: kubelet service is not enabled, please run 's
ystemctl enable kubelet.service'
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your inte
rnet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config
 images pull'
W0914 04:12:17.711154   27990 checks.go:844] detected that the sandbox image "re
gistry.k8s.io/pause:3.8" of the container runtime is inconsistent with that used
 by kubeadm.It is recommended to use "registry.k8s.io/pause:3.9" as the CRI sand
box image.
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
```

```
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 172.31.90.103:6443 --token 0zk8w3.xyegkydsy42vfscm \
        --discovery-token-ca-cert-hash sha256:31c672892b19dcb869fc46362d189234128f5bfc302bd41ae8c6078c56173f00
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

- Save the token

```
kubeadm join 172.31.90.103:6443 --token 0zk8w3.xyegkydsy42vfscm \
        --discovery-token-ca-cert-hash
sha256:31c672892b19dcb869fc46362d189234128f5bfc302bd41ae8c6078c56173f00
```

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

mkdir -p $HOME/.kube

```
[ec2-user@ip-172-31-90-103 yum.repos.d]$ mkdir -p $HOME/.kube
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

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

```
[ec2-user@ip-172-31-90-103 yum.repos.d]$ sudo cp -i /etc/kubernetes/admin.conf $
HOME/.kube/config
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

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

```
[ec2-user@ip-172-31-90-103 yum.repos.d]$
sudo chown $(id -u):$(id -g) $HOME/.kube/config
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

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

```
[ec2-user@ip-172-31-90-103 yum.repos.d]$ kubectl apply -f https://raw.githubuser
content.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
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

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

- Paste the below command on all 2 worker machines
  sudo yum install iproute-tc -y
  sudo systemctl enable kubelet
  sudo systemctl restart kubelet

- Now use the token from earlier to join into worker instances

```
kubeadm join 172.31.90.103:6443 --token 0zk8w3.xyegkydsy42vfscm \
        --discovery-token-ca-cert-hash
sha256:31c672892b19dcb869fc46362d189234128f5bfc302bd41ae8c6078c56173f00
```

- **kubectl get nodes** to check whether master and worker nodes are connected successfully

```
[ec2-user@ip-54-211-147-10 docker]$ kubectl get nodes
NAME                           STATUS   ROLES           AGE     VERSION
ip-172-31-90-103.ec2.internal  Ready    control-plane   3m21s   v1.30.5
```

Conclusion:

An EC2 instance was created on AWS Linux, and Docker, Kubernetes, Kubelet, Kubeadm, and Kubectl were installed. Kubernetes was initialized on the master node, which provided a token for connecting the master and worker nodes. On the slave node, `iproute` was installed, and Kubelet was enabled and restarted. However, there was an issue with joining the slave node to the cluster, resulting in only the master node being listed when running `kubectl get nodes`

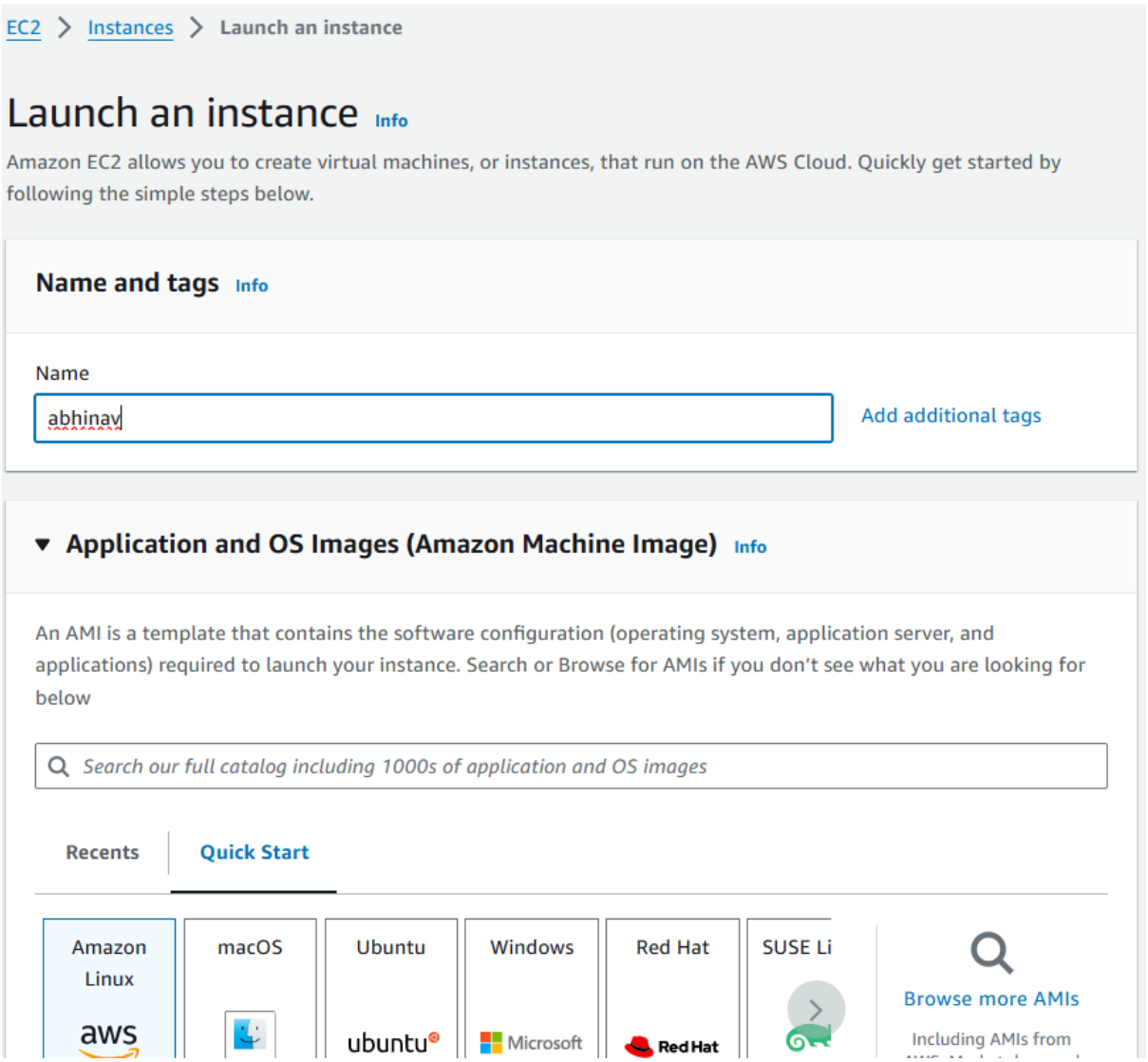
Name: Abhinav Swaminathan **D15C** **Roll No: 01**

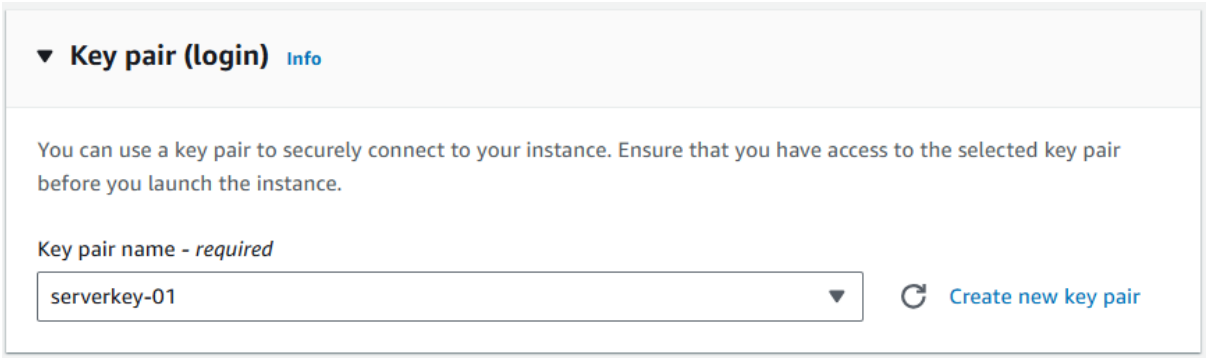
Experiment No. 4

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

Steps:

1. **Create an EC2 Linux Instances on AWS.**





Network settings Info

Edit

Network Info

vpc-0404d393731afabc3

Subnet Info

No preference (Default subnet in any availability zone)

Auto-assign public IP Info

Enable

Additional charges apply when outside of free tier allowance

Firewall (security groups) Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

(•) Create security group

( ) Select existing security group

We'll create a new security group called '**launch-wizard-4**' with the following rules:

[x] Allow SSH traffic from
Helps you connect to your instance

Anywhere
0.0.0.0/0

[x] Allow HTTPS traffic from the internet
To set up an endpoint, for example when creating a web server

[x] Allow HTTP traffic from the internet
To set up an endpoint, for example when creating a web server

⚠ Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only. ✕

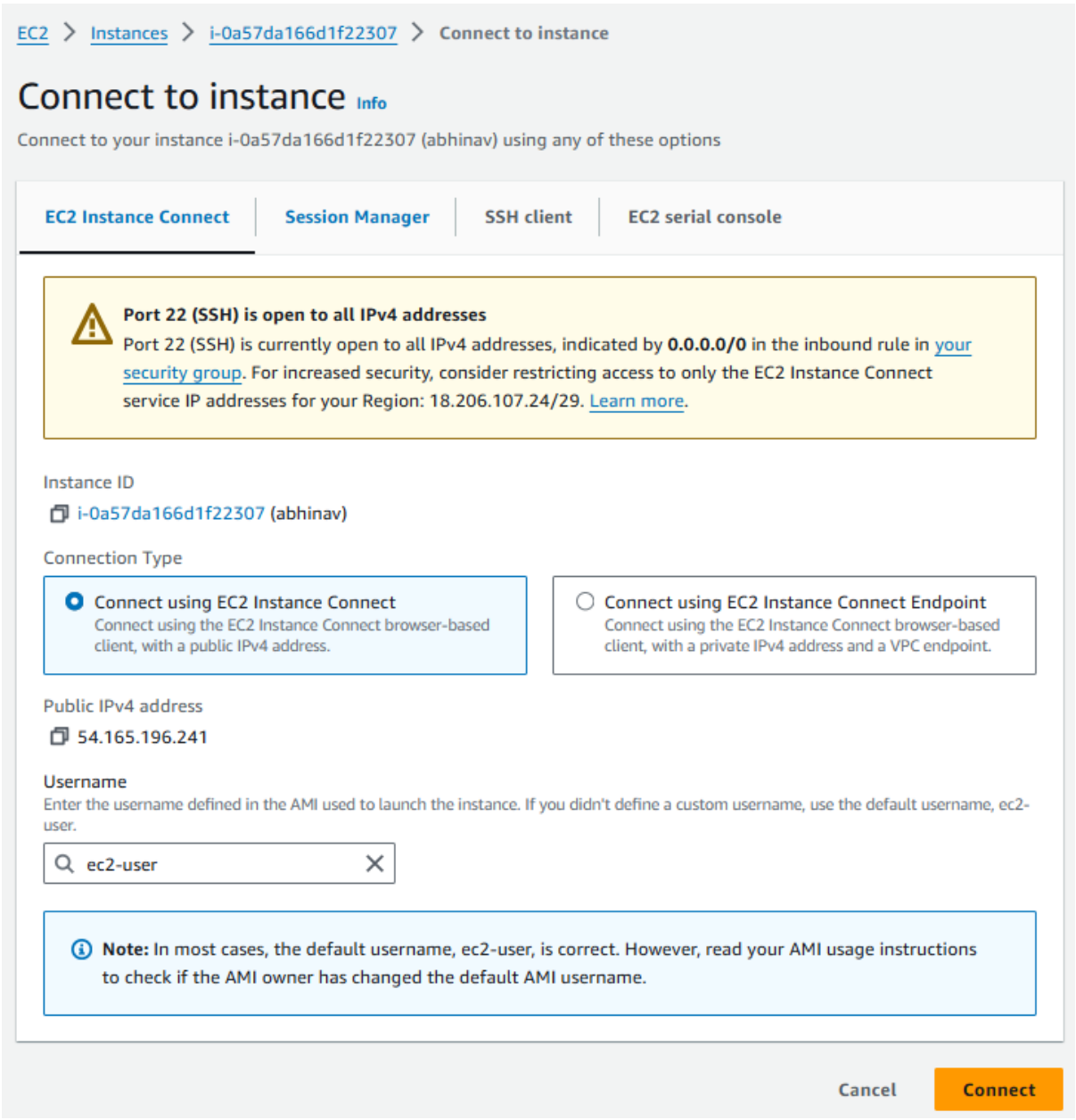
Instances (1) Info

Last updated less than a minute ago | Connect | Instance state | Actions | Launch instances

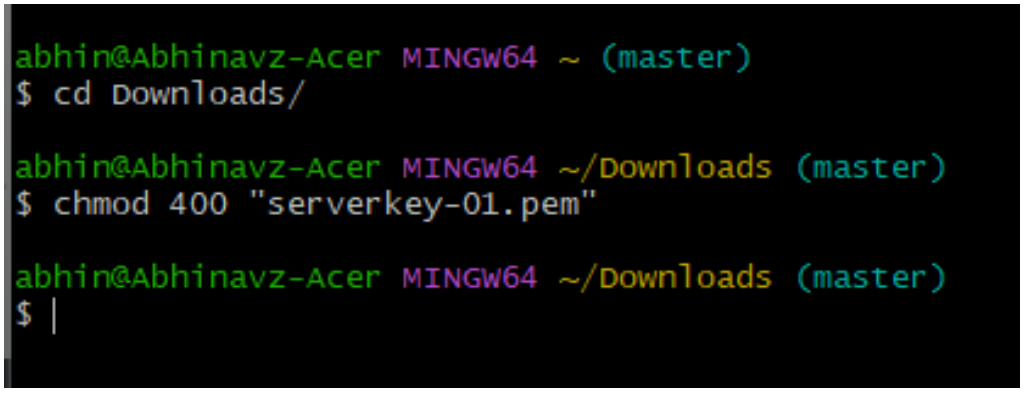
Find Instance by attribute or tag (case-sensitive) | Running | 1

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4
abhinav	i-0a57da166d1f22307	Running	t2.micro	2/2 checks passed	View alarms	us-east-1c	ec2-54-165-196-241.co...	54.165.196

2. **Then click on Id of that instance then click on connect**



3. **SSH into the created machine instance**
   - Give permissions to the current user to the downloaded pem file using - **chmod 400 <security_filename.pem>**



- Ssh using –
  ssh -i (keyname).pem (username)@(public ipv4 dns address)
  where keyname is name of the key you created. (server-01.pem). Other details can be found on the Instance dashboard.

```
abhin@Abhinavz-Acer MINGW64 ~/Downloads (master)
$ ssh -i "serverkey-01.pem" ec2-user@ec2-54-165-196-241.compute-1.amazonaws.com
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-31-90-103 ~]$
```

4. Installation Of Docke007

- **sudo yum install docker -y**

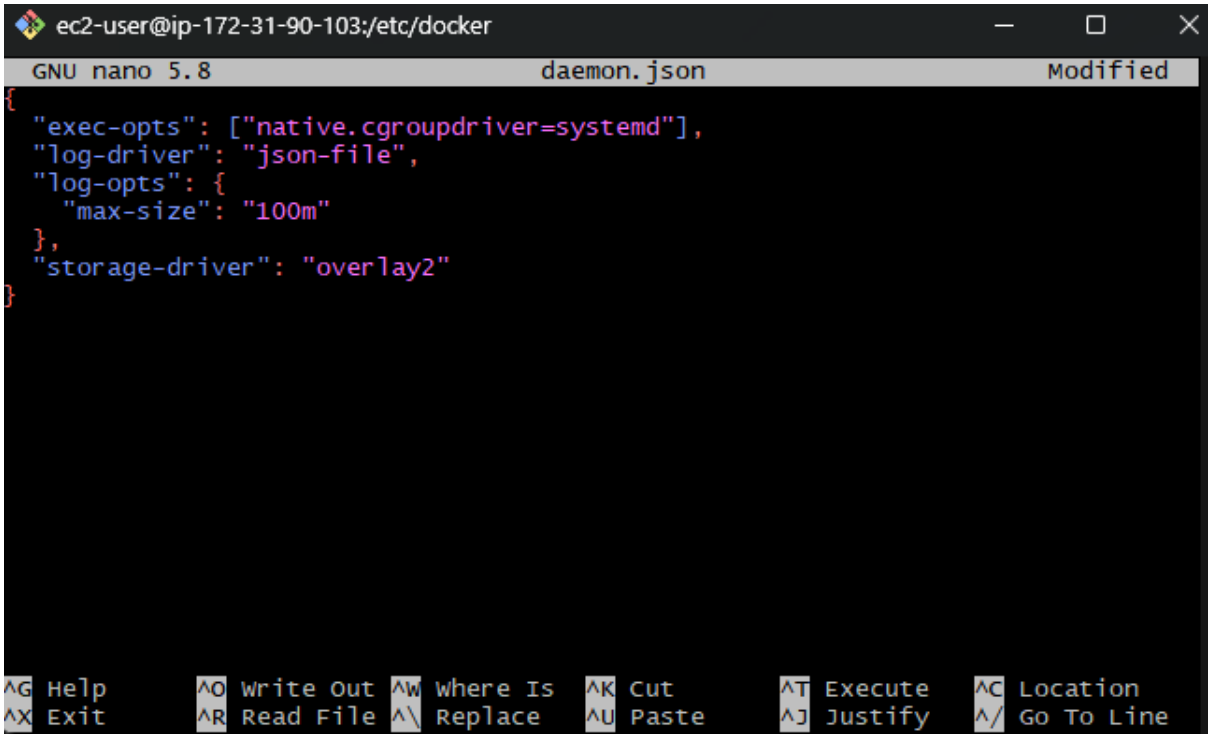
```
[ec2-user@ip-172-31-90-103 ~]$ sudo yum install docker -y
Last metadata expiration check: 0:13:41 ago on Sat Sep 14 03:42:27 2024.
Dependencies resolved.
================================================================================
 Package                 Arch     Version                  Repository     Size
================================================================================
Installing:
 docker                  x86_64   25.0.6-1.amzn2023.0.2    amazonlinux    44 M
Installing dependencies:
 containerd              x86_64   1.7.20-1.amzn2023.0.1    amazonlinux    35 M
 iptables-libs           x86_64   1.8.8-3.amzn2023.0.2     amazonlinux   401 k
 iptables-nft            x86_64   1.8.8-3.amzn2023.0.2     amazonlinux   183 k
 libcgroup               x86_64   3.0-1.amzn2023.0.1       amazonlinux    75 k
 libnetfilter_conntrack  x86_64   1.0.8-2.amzn2023.0.2     amazonlinux    58 k
 libnfnetlink            x86_64   1.0.1-19.amzn2023.0.2    amazonlinux    30 k
 libnftnl                x86_64   1.2.2-2.amzn2023.0.2     amazonlinux    84 k
 pigz                    x86_64   2.5-1.amzn2023.0.3       amazonlinux    83 k
 runc                    x86_64   1.1.13-1.amzn2023.0.1    amazonlinux   3.2 M

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

Total download size: 84 M
Installed size: 317 M
Downloading Packages:
(1/10): iptables-libs-1.8.8-3.amzn2023.0.2.x86_ 2.2 MB/s | 401 kB     00:00
(2/10): iptables-nft-1.8.8-3.amzn2023.0.2.x86_6 2.5 MB/s | 183 kB     00:00
(3/10): libcgroup-3.0-1.amzn2023.0.1.x86_64.rpm 1.3 MB/s |  75 kB     00:00
(4/10): libnetfilter_conntrack-1.0.8-2.amzn2023 1.3 MB/s |  58 kB     00:00
(5/10): libnfnetlink-1.0.1-19.amzn2023.0.2.x86_ 938 kB/s |  30 kB     00:00
(6/10): libnftnl-1.2.2-2.amzn2023.0.2.x86_64.rp 1.6 MB/s |  84 kB     00:00
(7/10): pigz-2.5-1.amzn2023.0.3.x86_64.rpm      1.7 MB/s |  83 kB     00:00
(8/10): runc-1.1.13-1.amzn2023.0.1.x86_64.rpm    21 MB/s | 3.2 MB     00:00
(9/10): containerd-1.7.20-1.amzn2023.0.1.x86_64  31 MB/s |  35 MB     00:01
(10/10): docker-25.0.6-1.amzn2023.0.2.x86_64.rp  28 MB/s |  44 MB     00:01
--------------------------------------------------------------------------------
Total                                            52 MB/s |  84 MB     00:01
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :
  Installing       : runc-1.1.13-1.amzn2023.0.1.x86_64
  Installing       : containerd-1.7.20-1.amzn2023.0.1.x86_64
  Running scriptlet: containerd-1.7.20-1.amzn2023.0.1.x86_64
  Installing       : pigz-2.5-1.amzn2023.0.3.x86_64
  Installing       : libnftnl-1.2.2-2.amzn2023.0.2.x86_64
  Installing       : libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64
  Installing       : libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64
  Installing       : iptables-libs-1.8.8-3.amzn2023.0.2.x86_64
  Installing       : iptables-nft-1.8.8-3.amzn2023.0.2.x86_64
```

```
Installed:
  containerd-1.7.20-1.amzn2023.0.1.x86_64                docker-25.0.6-1.amzn2023.0.2.x86_64
  libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64     libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64

Complete!
[ec2-user@ip-172-31-90-103 ~]$
```

- Configure cgroup in a daemon.json
  (this can be done by creating the file and using **nano** text editor)

```
{
        "exec-opts": ["native.cgroupdriver=systemd"],
        "log-driver": "json-file",
        "log-opts": {
        "max-size": "100m"
        },
        "storage-driver": "overlay2"
}
```

```
[ec2-user@ip-172-31-90-103 docker]$ sudo nano daemon.json
[ec2-user@ip-172-31-90-103 docker]$
```



- Enable and start docker and also load the daemon.json
  sudo systemctl enable docker
  sudo systemctl daemon-reload
  sudo systemctl restart docker

```
[ec2-user@ip-172-31-90-103 docker]$ sudo systemctl enable docker
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /us
r/lib/systemd/system/docker.service.
[ec2-user@ip-172-31-90-103 docker]$ sudo systemctl daemon-reload
[ec2-user@ip-172-31-90-103 docker]$
sudo systemctl restart docker
[ec2-user@ip-172-31-90-103 docker]$
```

- Check if docker is installed

```
[ec2-user@ip-172-31-90-103 docker]$ docker --version
Docker version 25.0.5, build 5dc9bcc
[ec2-user@ip-172-31-90-103 docker]$
```

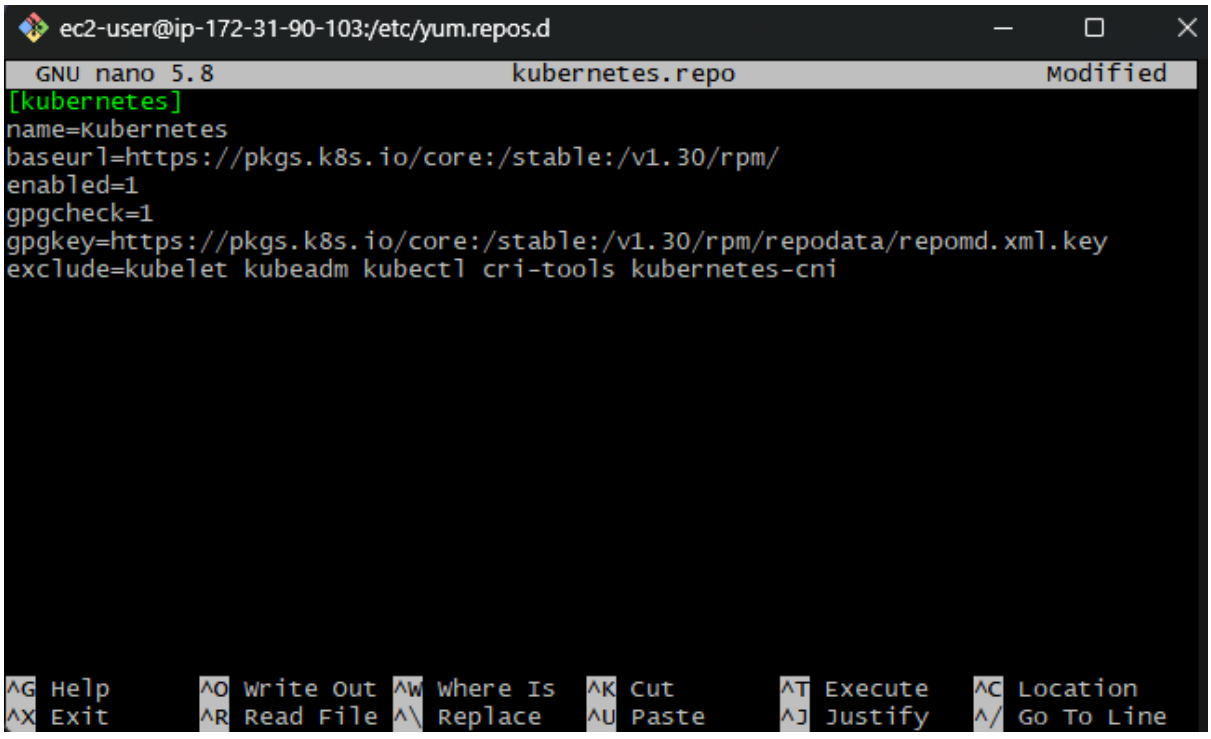
5. Install Kubernetes

- SELinux needs to be disabled before configuring kubelet
  sudo setenforce 0
  sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config

```
[ec2-user@ip-172-31-90-103 docker]$ sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
[ec2-user@ip-172-31-90-103 docker]$
```

- Add Kubernetes using the repo
  (this is done by creating **kubernetes.repo** file in **/etc/yum.repos.d** and configuring it using **nano** editor)
  [kubernetes]
  name=Kubernetes
  baseurl=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/
  enabled=1
  gpgcheck=1
  gpgkey=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/repodata/repomd.xml.key
  exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni

```
[ec2-user@ip-172-31-90-103 docker]$ cd /etc/yum.repos.d/
[ec2-user@ip-172-31-90-103 yum.repos.d]$ ls
amazonlinux.repo  kernel-livepatch.repo
[ec2-user@ip-172-31-90-103 yum.repos.d]$ sudo nano kubernetes.repo
[ec2-user@ip-172-31-90-103 yum.repos.d]$ ls
amazonlinux.repo  kernel-livepatch.repo  kubernetes.repo
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```



- Update packages list using **sudo yum update**

```
[ec2-user@ip-172-31-90-103 yum.repos.d]$ sudo yum update
Kubernetes                                      125 kB/s |  17 kB     00:00
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

- Install kubelet kubeadm kubectl

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

```
[ec2-user@ip-172-31-90-103 yum.repos.d]$ sudo yum install -y kubelet kubeadm kub
ectl --disableexcludes=kubernetes
Last metadata expiration check: 0:00:42 ago on Sat Sep 14 04:08:20 2024.
Dependencies resolved.
================================================================================
 Package                 Arch     Version                 Repository     Size
================================================================================
Installing:
 kubeadm                 x86_64   1.30.5-150500.1.1       kubernetes     10 M
 kubectl                 x86_64   1.30.5-150500.1.1       kubernetes     10 M
 kubelet                 x86_64   1.30.5-150500.1.1       kubernetes     17 M
Installing dependencies:
 conntrack-tools         x86_64   1.4.6-2.amzn2023.0.2    amazonlinux   208 k
 cri-tools               x86_64   1.30.1-150500.1.1       kubernetes    8.6 M
 kubernetes-cni          x86_64   1.4.0-150500.1.1        kubernetes    6.7 M
 libnetfilter_cthelper   x86_64   1.0.0-21.amzn2023.0.2   amazonlinux    24 k
 libnetfilter_cttimeout  x86_64   1.0.0-19.amzn2023.0.2   amazonlinux    24 k
 libnetfilter_queue      x86_64   1.0.5-2.amzn2023.0.2    amazonlinux    30 k

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

Total download size: 53 M
Installed size: 292 M
Downloading Packages:
(1/9): libnetfilter_cttimeout-1.0.0-19.amzn2023 448 kB/s |  24 kB     00:00
(2/9): libnetfilter_cthelper-1.0.0-21.amzn2023. 409 kB/s |  24 kB     00:00
(3/9): libnetfilter_queue-1.0.5-2.amzn2023.0.2. 1.5 MB/s |  30 kB     00:00
(4/9): conntrack-tools-1.4.6-2.amzn2023.0.2.x86 1.8 MB/s | 208 kB     00:00
(5/9): cri-tools-1.30.1-150500.1.1.x86_64.rpm    28 MB/s | 8.6 MB     00:00
(6/9): kubectl-1.30.5-150500.1.1.x86_64.rpm      23 MB/s |  10 MB     00:00
(7/9): kubeadm-1.30.5-150500.1.1.x86_64.rpm      18 MB/s |  10 MB     00:00
(8/9): kubelet-1.30.5-150500.1.1.x86_64.rpm      37 MB/s |  17 MB     00:00
(9/9): kubernetes-cni-1.4.0-150500.1.1.x86_64.r  20 MB/s | 6.7 MB     00:00
--------------------------------------------------------------------------------
Total                                            56 MB/s |  53 MB     00:00
```

```
Installed:
  conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64
  cri-tools-1.30.1-150500.1.1.x86_64
  kubeadm-1.30.5-150500.1.1.x86_64
  kubectl-1.30.5-150500.1.1.x86_64
  kubelet-1.30.5-150500.1.1.x86_64
  kubernetes-cni-1.4.0-150500.1.1.x86_64
  libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64
  libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64
  libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64

Complete!
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

- 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

```
[ec2-user@ip-172-31-90-103 yum.repos.d]$ . sudo swapoff -a
-bash: sudo: No such file or directory
[ec2-user@ip-172-31-90-103 yum.repos.d]$ sudo swapoff -a
[ec2-user@ip-172-31-90-103 yum.repos.d]$ echo "net.bridge.bridge-nf-call-iptable
s=1" | sudo tee -a /etc/sysctl.conf
net.bridge.bridge-nf-call-iptables=1
[ec2-user@ip-172-31-90-103 yum.repos.d]$ sudo sysctl -p
net.bridge.bridge-nf-call-iptables = 1
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

6. **Initialize the Kubecluster**

   sudo kubeadm init --podnetwork-cidr=10.244.0.0/16

```
[ec2-user@ip-172-31-90-103 yum.repos.d]$ sudo kubeadm init --pod-network-cidr=10
.244.0.0/16 --ignore-preflight-errors=all
I0914 04:12:17.448521   27990 version.go:256] remote version is much newer: v1.3
1.0; falling back to: stable-1.30
[init] Using Kubernetes version: v1.30.4
[preflight] Running pre-flight checks
        [WARNING NumCPU]: the number of available CPUs 1 is less than the requir
ed 2
        [WARNING Mem]: the system RAM (949 MB) is less than the minimum 1700 MB
        [WARNING FileExisting-socat]: socat not found in system path
        [WARNING FileExisting-tc]: tc not found in system path
        [WARNING Service-Kubelet]: kubelet service is not enabled, please run 's
ystemctl enable kubelet.service'
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your inte
rnet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config
 images pull'
W0914 04:12:17.711154   27990 checks.go:844] detected that the sandbox image "re
gistry.k8s.io/pause:3.8" of the container runtime is inconsistent with that used
 by kubeadm.It is recommended to use "registry.k8s.io/pause:3.9" as the CRI sand
box image.
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
```

```
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 172.31.90.103:6443 --token 0zk8w3.xyegkydsy42vfscm \
        --discovery-token-ca-cert-hash sha256:31c672892b19dcb869fc46362d189234128f5bfc302bd41ae8c6078c56173f00
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

- Save the token

```
kubeadm join 172.31.90.103:6443 --token 0zk8w3.xyegkydsy42vfscm \
        --discovery-token-ca-cert-hash
sha256:31c672892b19dcb869fc46362d189234128f5bfc302bd41ae8c6078c56173f00
```

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

mkdir -p $HOME/.kube

```
[ec2-user@ip-172-31-90-103 yum.repos.d]$ mkdir -p $HOME/.kube
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

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

```
[ec2-user@ip-172-31-90-103 yum.repos.d]$ sudo cp -i /etc/kubernetes/admin.conf $
HOME/.kube/config
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

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

```
[ec2-user@ip-172-31-90-103 yum.repos.d]$
sudo chown $(id -u):$(id -g) $HOME/.kube/config
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

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

```
[ec2-user@ip-172-31-90-103 yum.repos.d]$ kubectl apply -f https://raw.githubuser
content.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
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

7. Deploy nginx server

- Apply deployment using this following command:
  kubectl apply -f https://k8s.io/examples/pods/simple-pod.yaml

```
[ec2-user@ip-172-31-90-103 docker]$ kubectl apply -f https://k8s.io/examples/pods/s
imple-pod.yaml
pod/nginx created
```

- use **kubectl get nodes** to check whether the pod gets created or not

```
[ec2-user@ip-172-31-90-103 docker]$ kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
nginx   0/1     Pending   0          12s
```

- To convert state from pending to running use following command:
  kubectl describe pod nginx (This command will help to describe the pods it gives reason for failure as it shows the untolerated taints which need to be untainted.)

```
[ec2-user@ip-172-31-90-103 docker]$ kubectl describe pod nginx
Name:             nginx
Namespace:        default
Priority:         0
Service Account:  default
Node:             <none>
Labels:           <none>
Annotations:      <none>
Status:           Pending
IP:
IPs:              <none>
Containers:
  nginx:
    Image:        nginx:1.14.2
    Port:         80/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-k4lj6 (ro)
    Type:                     Projected (a volume that contains injected data from m
ultiple 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 3
00s
                              node.kubernetes.io/unreachable:NoExecute op=Exists for
 300s
Events:
  Type     Reason            Age   From               Message
  ----     ------            ----  ----               -------
  Warning  FailedScheduling  7s    default-scheduler  0/1 nodes are available: 1 no
de(s) had untolerated taint {node-role.kubernetes.io/control-plane: }. preemption:
0/1 nodes are available: 1 Preemption is not helpful for scheduling.
```

- check pod status

```
[ec2-user@ip-172-31-90-103 ~]$ kubectl get pods
NAME    READY   STATUS    RESTARTS      AGE
nginx   1/1     Running   1 (6s ago)    90s
```

- mention the port you want to host

```
[ec2-user@ip-172-31-90-103 ~]$ kubectl port-forward nginx 8081:80
Forwarding from 127.0.0.1:8081 -> 80
Forwarding from [::1]:8081 -> 80
```

8. Verify your deployment

- Open up a new terminal and ssh to your EC2 instance. Then, use this curl command to check if the Nginx server is running. **curl --head http://127.0.0.1:8080** If the response is 200 OK and you can see the Nginx server name, your deployment was successful. We have successfully deployed our Nginx server on our EC2 instance.

```
[ec2-user@172-31-90-103 ~]$ curl --head http://127.0.0.1:8080
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Linux)
Date: Sat, 14 Sep 2024 12:31:53 GMT
Content-Type: text/html
Content-Length: 612
Connection: keep-alive
```

Conclusion:

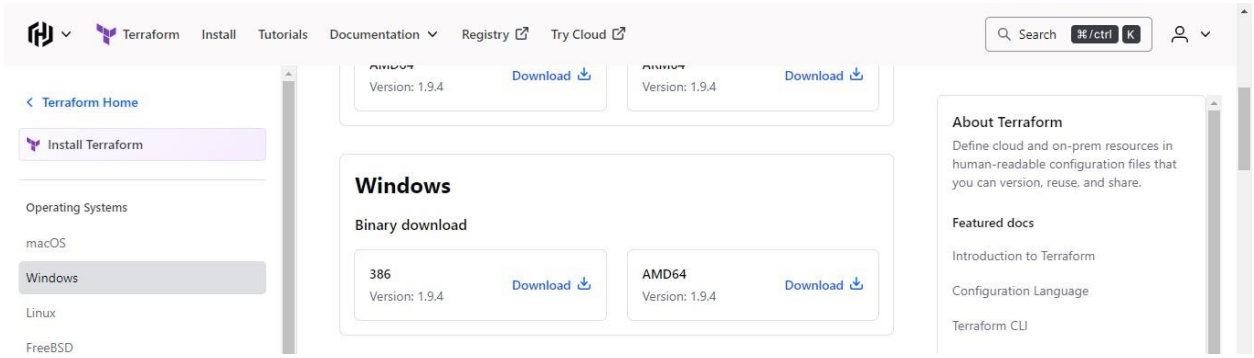
An AWS EC2 Linux instance was set up, and Docker and Kubernetes were installed. Kubernetes was initialized successfully, and the required commands were executed. Flannel was installed as a networking plugin. Although there was an initial error with the Nginx deployment, it was eventually deployed successfully using the `simple-pod.yml` file and accessed via localhost on port 8080.

Name: Abhinav Swaminathan D15C Roll No: 01

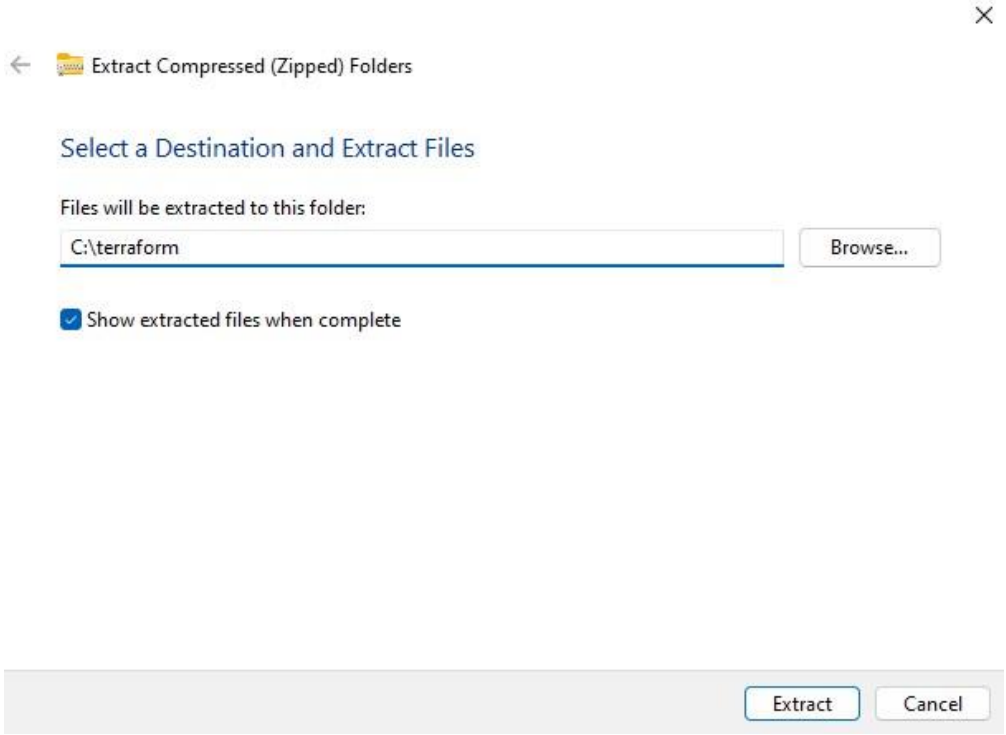
Experiment No. 5

Installation of Terraform

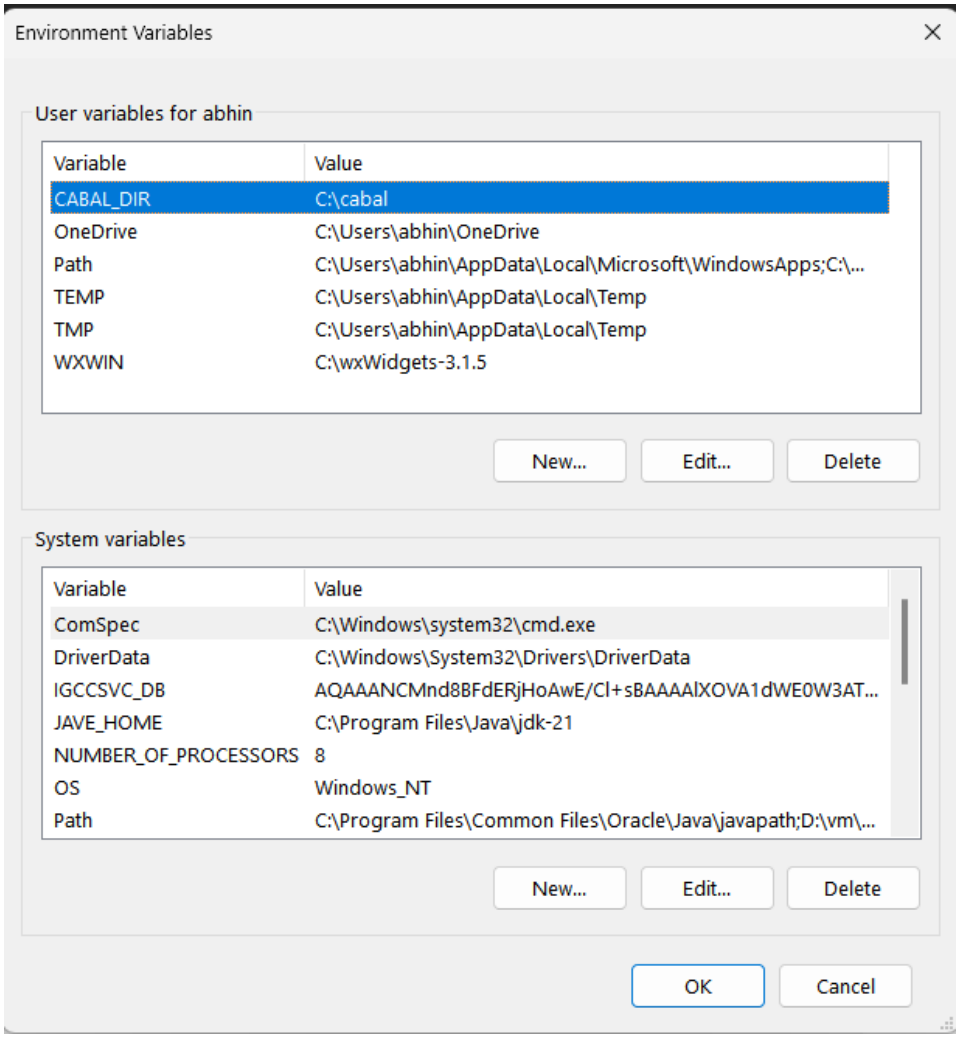
Step 1:To install Terraform, visit the official Terraform website mentioned below, go to the Downloads section, select Windows, and download the 64-bit version for your system. website:https://www.terraform.io/downloads.html



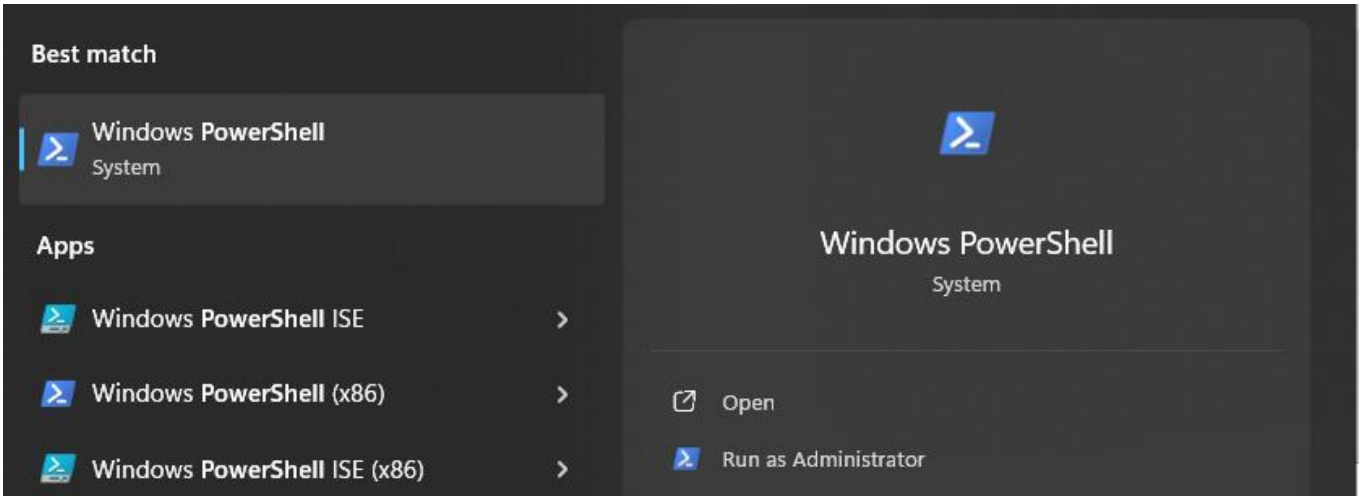
Step 2: Extract the downloaded `Terraform.exe` file to the `C:\Terraform` directory on your system.



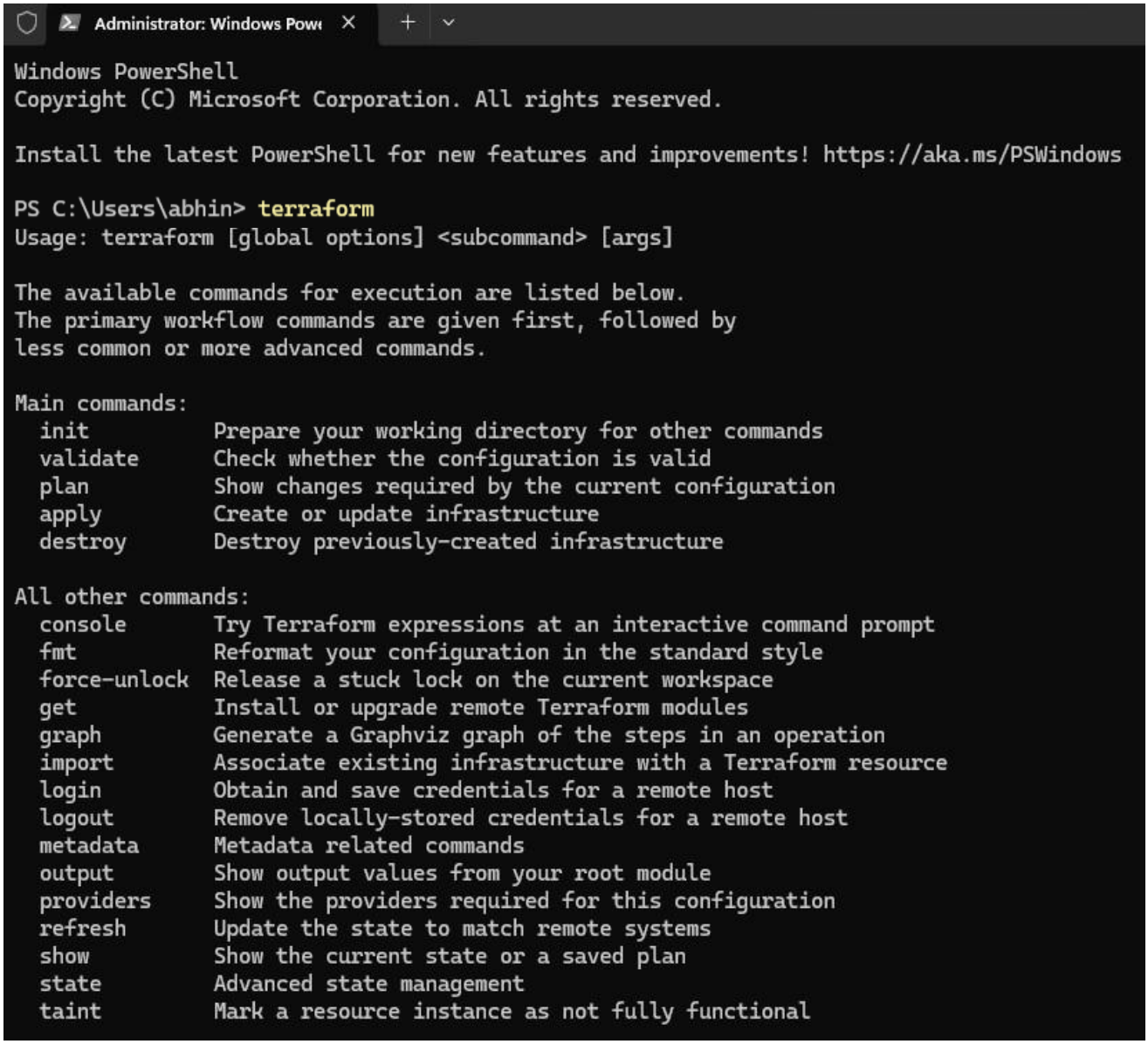
Step 3: Set the System path for Terraform in Environment Variables



Step 4: Run the windows powershell as administrator.



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



Name: Abhinav Swaminathan D15C Roll No: 01

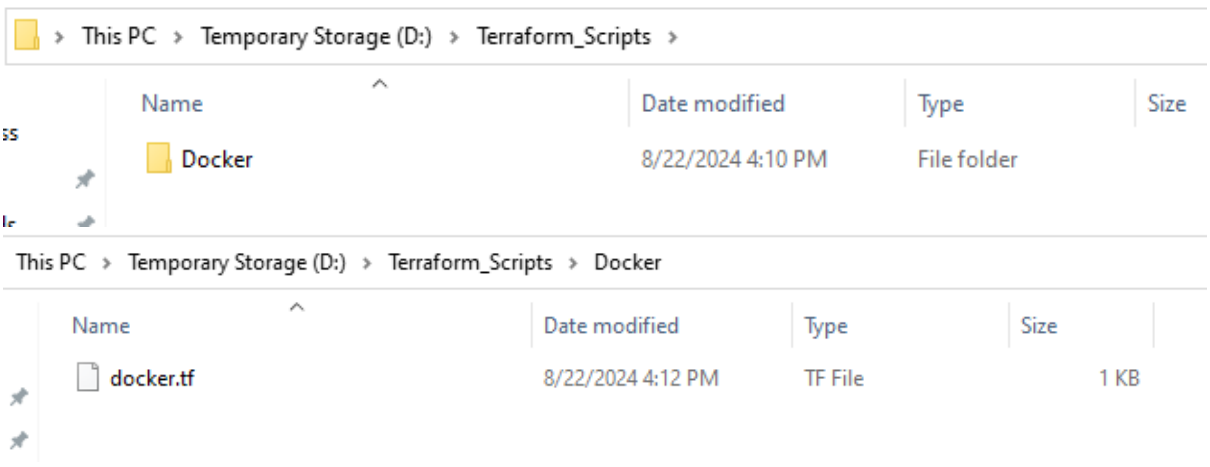
Experiment No. 6

1. Creating a docker image using terraform

```
C:\Users\abhinav>docker --version
Docker version 26.1.3, build b72abbb

C:\Users\abhinav>docker

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

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

Management Commands:
  builder     Manage builds
  buildx*     Docker Buildx
  compose*    Docker Compose
  container   Manage containers
  context     Manage contexts
```

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

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.



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"
}
```

4. Execute Terraform Init command to initialize the resources

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

Terraform has been successfully initialized!

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

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

D:\Terraform_Scripts\Docker>
```

5. Execute Terraform plan to see the available resources

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

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

Terraform will perform the following actions:

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

      + healthcheck (known after apply)

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

```
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... [19s elapsed] docker_image.ubuntu: Still creating... (20s elapsed) docker_image.
ubuntu: Still creating... [30s elapsed]
docker_image.ubuntu: Creation complete after 30s [id=sha256:263966596d42ad38ae9914716692777ba9ff8779a62ad93a74fe82e3e1f
ubuntu:latest] docker_container.foo: Creating...
```

7. Check Docker images, Before and After Executing Apply step

```
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>
```

```
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.

```
D:\Terraform_Scripts\Docker>terraform destroy
docker_image.ubuntu: Refreshing state... [idrsha256:2dc29b50dcd2d30101475692777ba087762d92de0221fubuntu:latest)
Terrafore 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           = "sha256:2dc39b859dc2ade8ae30147b5692777ba9ff9779a62ad93a78de82e3elfubuntu:latest" -> null
           - image_id     = "sha256:2dc39b859dc2ade8ae30147b5692777ba9ff9779a62ad93a78de82e3elf" -> null
           - Latest       = "sha256:2dc39b859dc2ade8ae30147b5692777ba9ff9779a62ad93a78de82e3elf" -> null
           - name         = "ubuntu:latest" -> null
           - repo digest  = "ubuntu@sha256:204a3d7bb4d7723452be3923b06cd7043704030041c83c#7856c1" -> null
       }

Plan: to add, 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 so unde. Only 'yes' will be accepted to confirm.

  Enter a value: yes

docker image.ubuntu: Destroying... [id-sha256:2de99b59cd42ade83814765692777ba5ff8779a62ad93ad62e3e1fubuntu:latest]
docker image.ubuntu: Destruction complete after is

Destroy complete! Resources: 1 destroyed.
```

9. Check Docker images, After Executing Destroy step

```
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>
```

Name : Abhinav Swaminthan **Roll No : 01** **Class : D15C**

Experiment 7

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

Theory:

What is SAST?

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

What problems does SAST solve?

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

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

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

Why is SAST important?

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

What are the key steps to run SAST effectively?

There are six simple steps needed to perform SAST efficiently in organizations that have a very large number of applications built with different languages, frameworks, and platforms.

1. **Finalize the tool.** Select a static analysis tool that can perform code reviews of applications written in the programming languages you use. The tool should also be able to comprehend the underlying framework used by your software.

2. **Create the scanning infrastructure, and deploy the tool.** This step involves handling the licensing requirements, setting up access control and authorization, and procuring the resources required (e.g., servers and databases) to deploy the tool.

3. **Customize the tool.** Fine-tune the tool to suit the needs of the organization. For example, you might configure it to reduce false positives or find additional security vulnerabilities by writing new rules or updating existing ones. Integrate the tool into the build environment, create dashboards for tracking scan results, and build custom reports.

4. **Prioritize and onboard applications.** Once the tool is ready, onboard your applications. If you have a large number of applications, prioritize the high-risk applications to scan first. Eventually, all your applications should be onboarded and scanned regularly, with application scans synced with release cycles, daily or monthly builds, or code check-ins.

5. **Analyze scan results.** This step involves triaging the results of the scan to remove false positives. Once the set of issues is finalized, they should be tracked and provided to the deployment teams for proper and timely remediation.

6. **Provide governance and training.** Proper governance ensures that your development teams are employing the scanning tools properly. The software security touchpoints should be present within the SDLC. SAST should be incorporated as part of your application development and deployment process.

Integrating Jenkins with SonarQube:

Prerequisites:

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

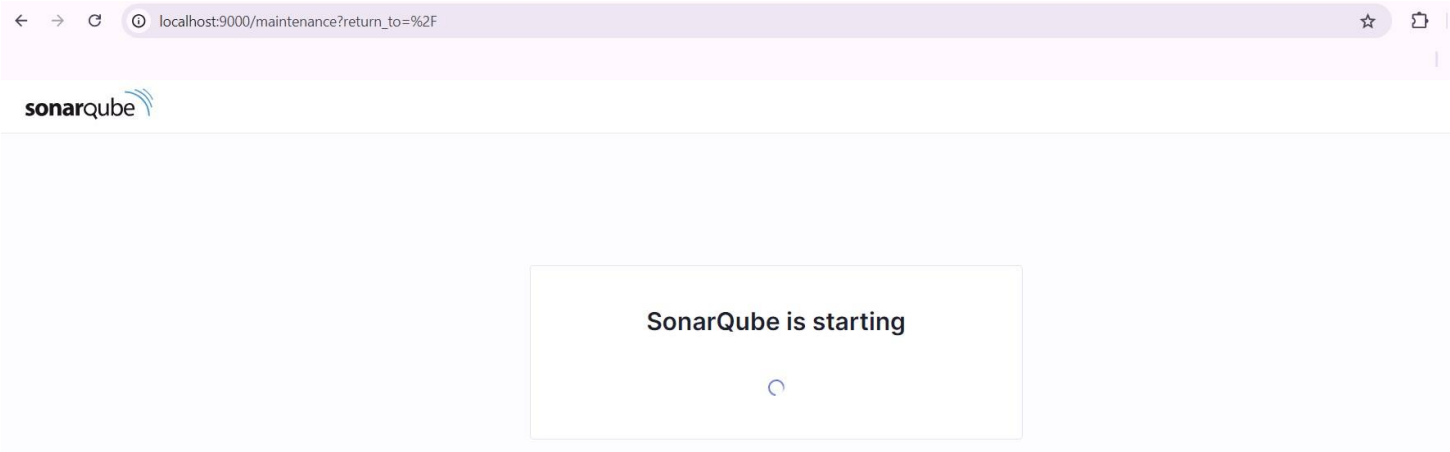
Steps to integrate Jenkins with SonarQube

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

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

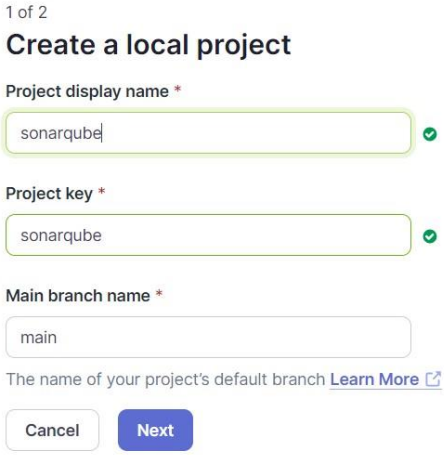
```
C:\Users\ADMIN>docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
Unable to find image 'sonarqube:latest' locally
latest: Pulling from library/sonarqube
7478e0ac0f23: Pull complete
90a925ab929a: Pull complete
7d9a34308537: Pull complete
80338217a4ab: Pull complete
1a5fd5c7e184: Pull complete
7b87d6fa783d: Pull complete
bd819c9b5ead: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:72e9feec71242af83faf65f95a40d5e3bb2822a6c3b2cda8568790f3d31aecde
Status: Downloaded newer image for sonarqube:latest
de76efbeef2054aeb442b86ba54c2916039b8757b388482d9780ffc69f5d8bbe
```

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



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

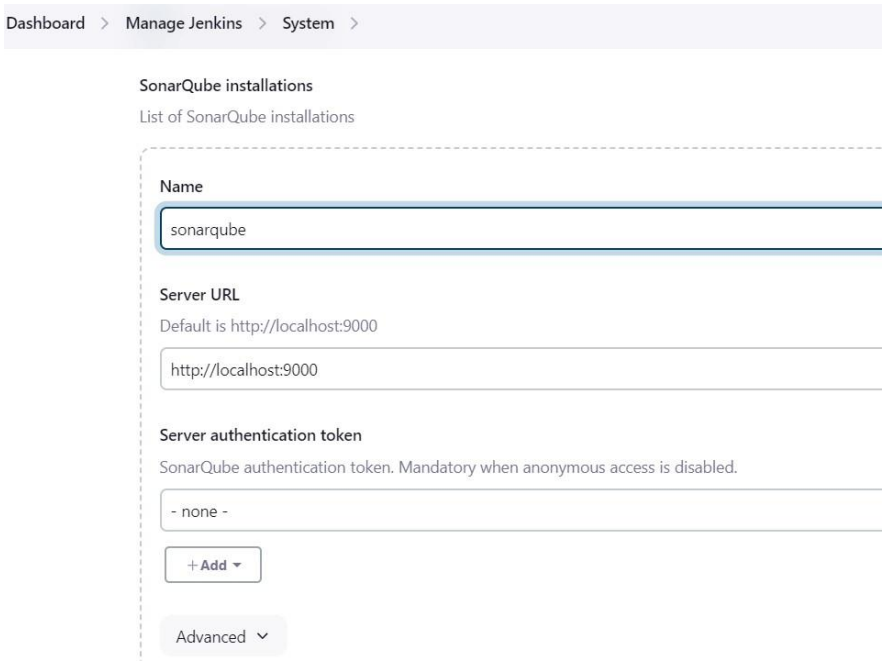
5. Create a manual project in SonarQube with the name **sonarqube**



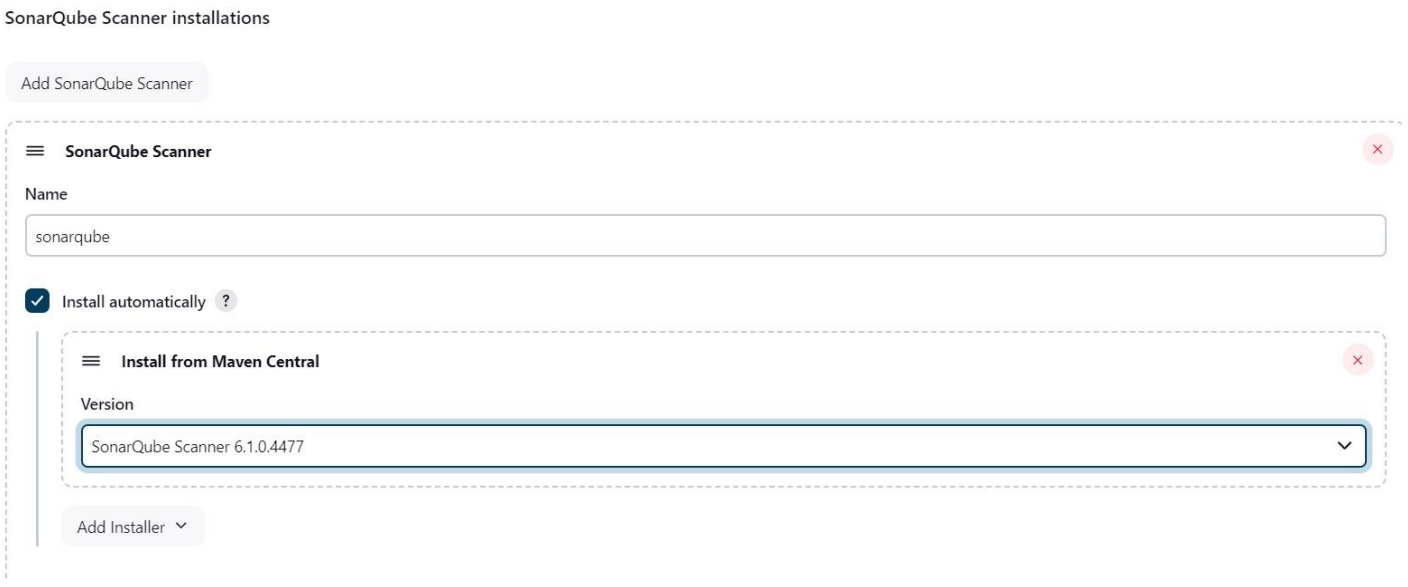
Setup the project and come back to Jenkins Dashboard.

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

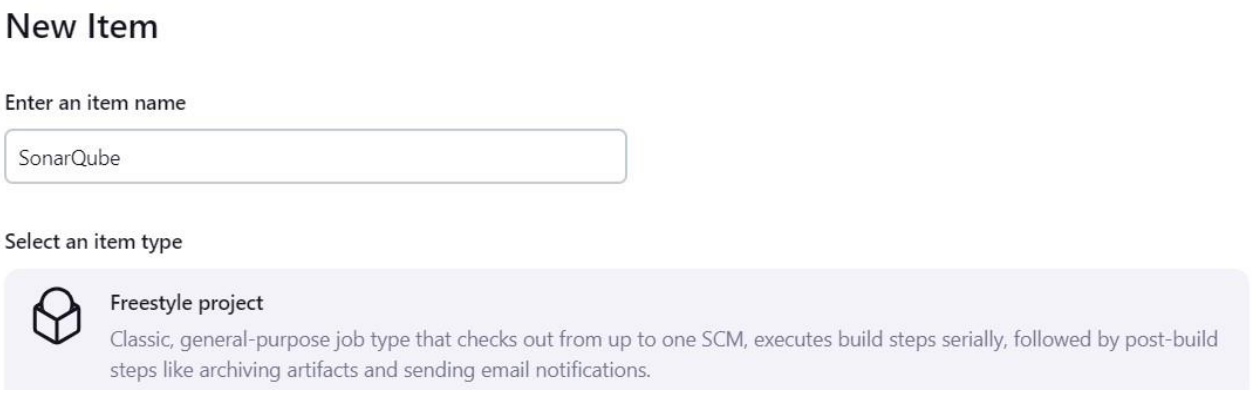
7. Under Jenkins 'Configure System', look for SonarQube Servers and enter the details. Enter the Server Authentication token if needed.



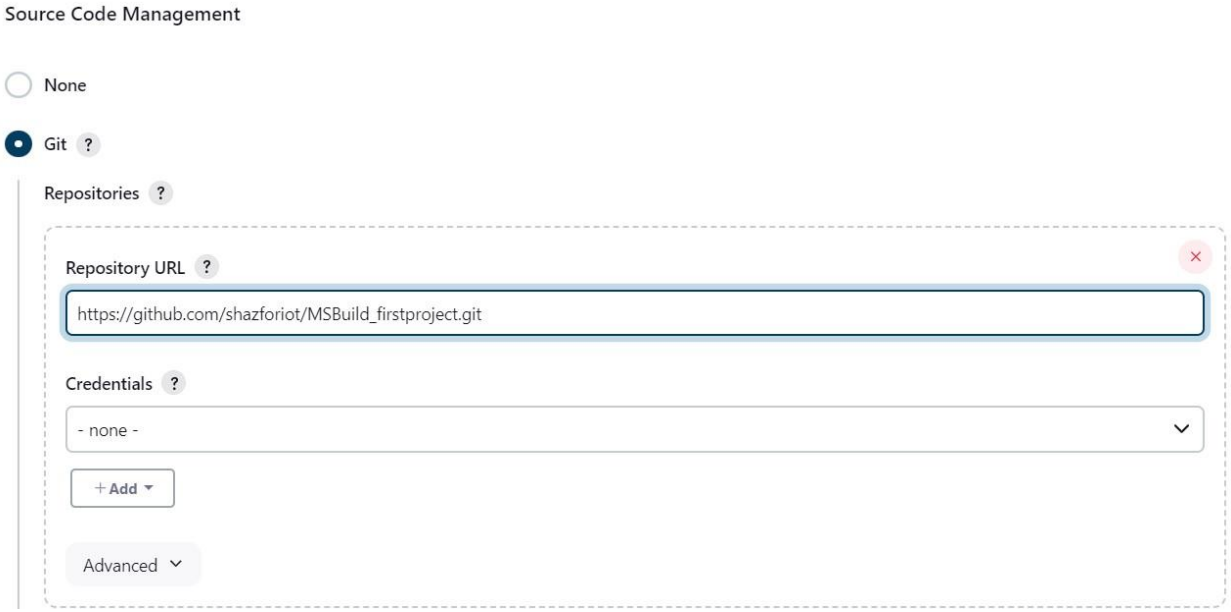
8. Search for SonarQube Scanner under Global Tool Configuration. Choose the latest configuration and choose Install automatically.



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

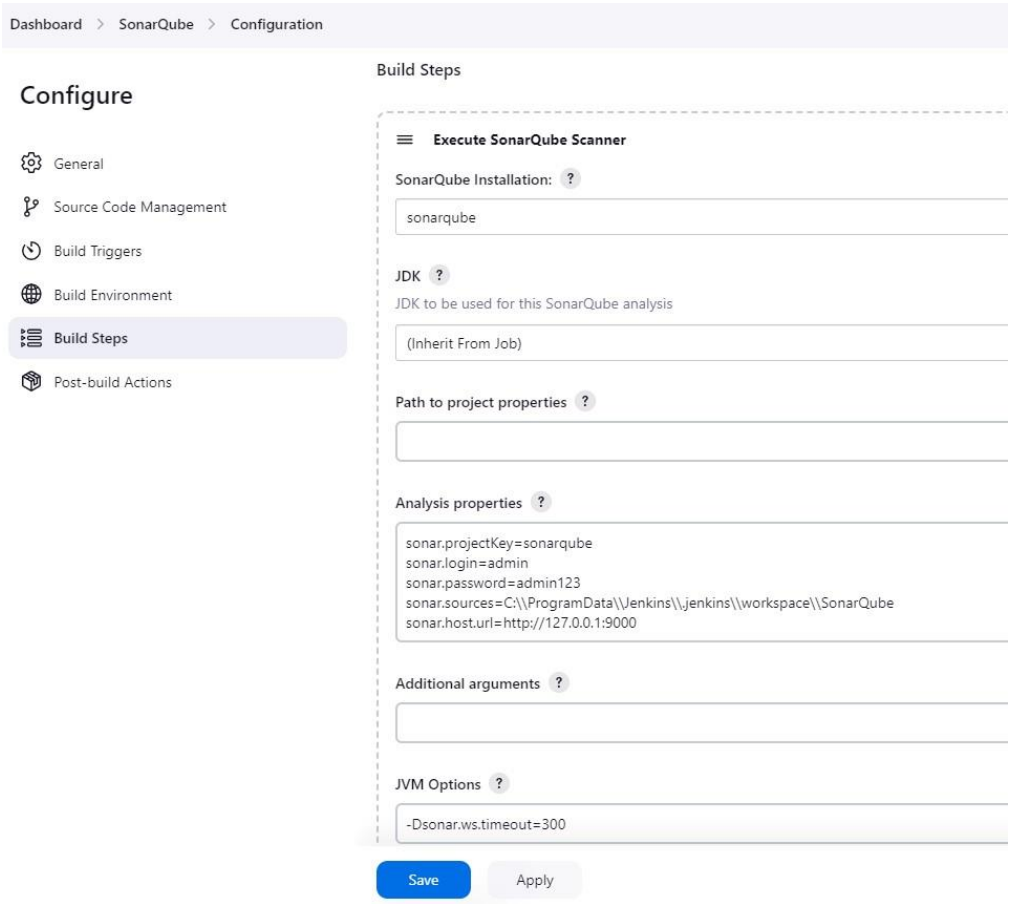


10. Choose this GitHub repository in Source Code Management.
    https://github.com/shazforiot/MSBuild_firstproject.git

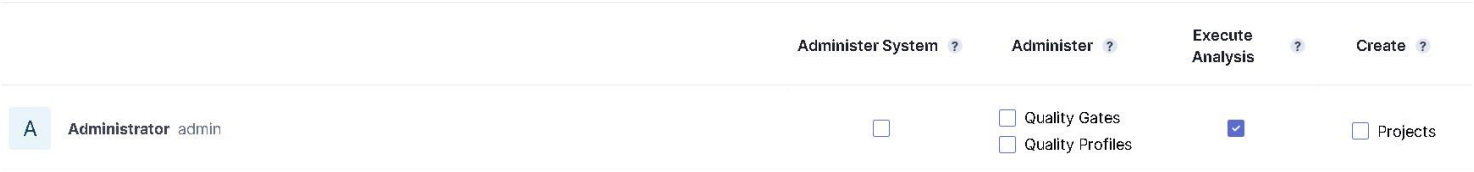


It is a sample hello-world project with no vulnerabilities and issues, just to test the integration.

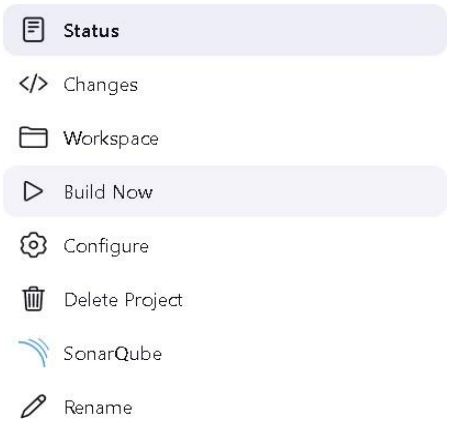
11. Under Build-> Execute SonarQube Scanner, enter these Analysis properties. Mention the SonarQube Project Key, Login, Password, Source path and Host URL.



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



13. Run The Build.



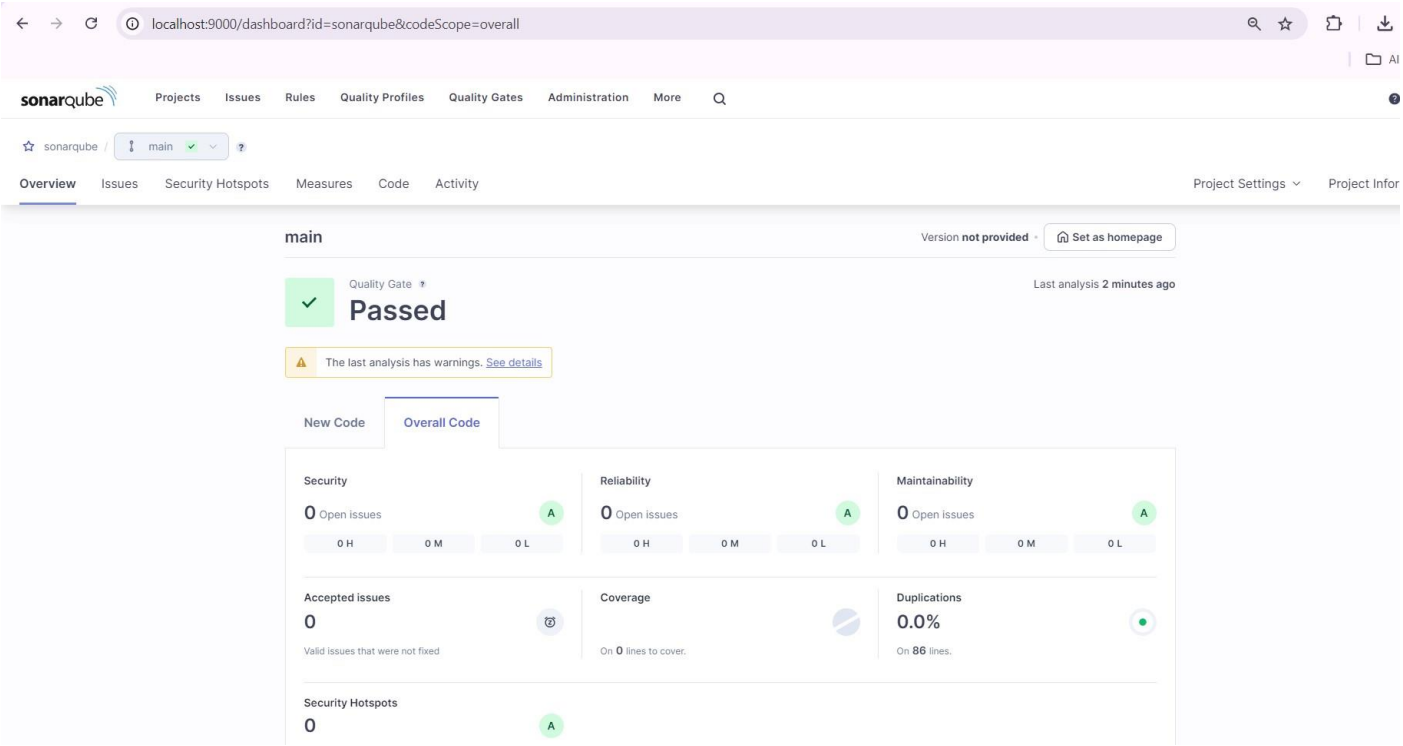
Check the console output.

Console Output

Download Copy View as plain tex

```
Started by user Abhinav Swaminathan
Running as SYSTEM
Building on the built-in node in workspace C:\ProgramData\Jenkins\.jenkins\workspace\SonarQube
The recommended git tool is: NONE
No credentials specified
 > git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\SonarQube\.git # timeout=10
Fetching changes from the remote Git repository
 > git.exe config remote.origin.url https://github.com/shazforiot/MSBuild_firstproject.git # timeout=10
Fetching upstream changes from https://github.com/shazforiot/MSBuild_firstproject.git
 > git.exe --version # timeout=10
 > git --version # 'git version 2.45.2.windows.1'
 > git.exe fetch --tags --force --progress -- https://github.com/shazforiot/MSBuild_firstproject.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision f2bc042c04c6e72427c380bcaee6d6fee7b49adf (refs/remotes/origin/master)
 > git.exe config core.sparsecheckout # timeout=10
 > git.exe checkout -f f2bc042c04c6e72427c380bcaee6d6fee7b49adf # timeout=10
Commit message: "updated"
 > git.exe rev-list --no-walk f2bc042c04c6e72427c380bcaee6d6fee7b49adf # timeout=10
[SonarQube] $ C:\ProgramData\Jenkins\.jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\sonarqube\bin\sonar-scanner.bat -Dsonar.host.url=http://localhost:9000 -Dsonar.projectKey=sonarqube -
Dsonar.login=admin -Dsonar.host.url=http://127.0.0.1:9000 -Dsonar.sources=C:\ProgramData\Jenkins\.jenkins\workspace\SonarQube -Dsonar.password=admin123 -
Dsonar.projectBaseDir=C:\ProgramData\Jenkins\.jenkins\workspace\SonarQube
16:16:39.198 WARN  Property 'sonar.host.url' with value 'http://localhost:9000' is overridden with value 'http://127.0.0.1:9000'
16:16:39.206 INFO  Scanner configuration file: C:\ProgramData\Jenkins\.jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\sonarqube\bin\..\conf\sonar-scanner.properties
16:16:39.206 INFO  Project root configuration file: NONE
16:16:39.230 INFO  SonarScanner CLI 6.1.0.4477
16:16:39.230 INFO  Java 21.0.4 Eclipse Adoptium (64-bit)
16:16:39.230 INFO  Windows 11 10.0 amd64
16:16:39.230 INFO  SONAR_SCANNER_OPTS=-Dsonar.ws.timeout=300
16:16:39.254 INFO  User cache: C:\Windows\system32\config\systemprofile\.sonar\cache
16:16:58.734 INFO  Using git CLI to retrieve untracked files
16:16:58.791 INFO  Analyzing language associated files and files included via "sonar.text.inclusions" that are tracked by git
16:16:58.856 INFO  14 source files to be analyzed
16:16:59.154 INFO  14/14 source files have been analyzed
16:16:59.154 INFO  Sensor TextAndSecretsSensor [text] (done) | time=1306ms
16:16:59.163 INFO  ------------- Run sensors on project
16:16:59.373 INFO  Sensor C# [csharp]
16:16:59.373 WARN  Your project contains C# files which cannot be analyzed with the scanner you are using. To analyze C# or VB.NET, you must use the SonarScanner for .NET 5.x or higher, see
https://redirect.sonarsource.com/doc/install-configure-scanner-msbuild.html
16:16:59.373 INFO  Sensor C# [csharp] (done) | time=0ms
16:16:59.373 INFO  Sensor Analysis Warnings import [csharp]
16:16:59.379 INFO  Sensor Analysis Warnings import [csharp] (done) | time=0ms
16:16:59.379 INFO  Sensor C# File Caching Sensor [csharp]
16:16:59.379 WARN  Incremental PR analysis: Could not determine common base path, cache will not be computed. Consider setting 'sonar.projectBaseDir' property.
16:16:59.379 INFO  Sensor C# File Caching Sensor [csharp] (done) | time=6ms
16:16:59.379 INFO  Sensor Zero Coverage Sensor
16:16:59.389 INFO  Sensor Zero Coverage Sensor (done) | time=10ms
16:16:59.389 INFO  SCM Publisher SCM provider for this project is: git
16:16:59.389 INFO  SCM Publisher 4 source files to be analyzed
16:16:59.838 INFO  SCM Publisher 4/4 source files have been analyzed (done) | time=449ms
16:16:59.846 INFO  CPD Executor Calculating CPD for 0 files
16:16:59.846 INFO  CPD Executor CPD calculation finished (done) | time=0ms
16:16:59.854 INFO  SCM revision ID 'f2bc042c04c6e72427c380bcaee6d6fee7b49adf'
16:17:00.121 INFO  Analysis report generated in 120ms, dir size=201.1 kB
16:17:00.195 INFO  Analysis report compressed in 57ms, zip size=22.4 kB
16:17:00.393 INFO  Analysis report uploaded in 195ms
16:17:00.394 INFO  ANALYSIS SUCCESSFUL, you can find the results at: http://127.0.0.1:9000/dashboard?id=sonarqube
16:17:00.395 INFO  Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
16:17:00.395 INFO  More about the report processing at http://127.0.0.1:9000/api/ce/task?id=acd819f5-9e70-42ab-bff7-3cc893e2cae4
16:17:00.405 INFO  Analysis total time: 18.743 s
16:17:00.408 INFO  SonarScanner Engine completed successfully
16:17:00.494 INFO  EXECUTION SUCCESS
16:17:00.494 INFO  Total time: 21.288s
Finished: SUCCESS
```

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



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

Conclusion:

In this experiment, the integration of Jenkins with SonarQube for performing Static Application Security Testing (SAST) was explored. SonarQube was set up within a Docker container, and Jenkins was configured to utilize the SonarQube scanner.

By creating a manual project in SonarQube and configuring the necessary authentication and tools in Jenkins, a seamless connection was established between Jenkins and SonarQube for static code analysis.

The integration was tested using a sample GitHub repository, successfully executing a build and analyzing the project's code quality through SonarQube.

This process provided valuable insights into the SAST process, Jenkins automation, and SonarQube's capabilities in identifying potential code vulnerabilities.

Name: Abhinav Swaminathan **D15C** **Roll No: 01**

Experiment No. 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.

Theory:

What is SAST?

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

What problems does SAST solve?

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

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

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

Why is SAST important?

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

What is a CI/CD Pipeline?

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

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

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

What is SonarQube?

SonarQube is an open-source platform developed by SonarSource for continuous inspection of code quality. Sonar does static code analysis, which provides a detailed report of bugs, code smells, vulnerabilities, code duplications.

It supports 25+ major programming languages through built-in rulesets and can also be extended with various plugins.

Benefits of SonarQube

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

Integrating Jenkins with SonarQube:

Prerequisites:

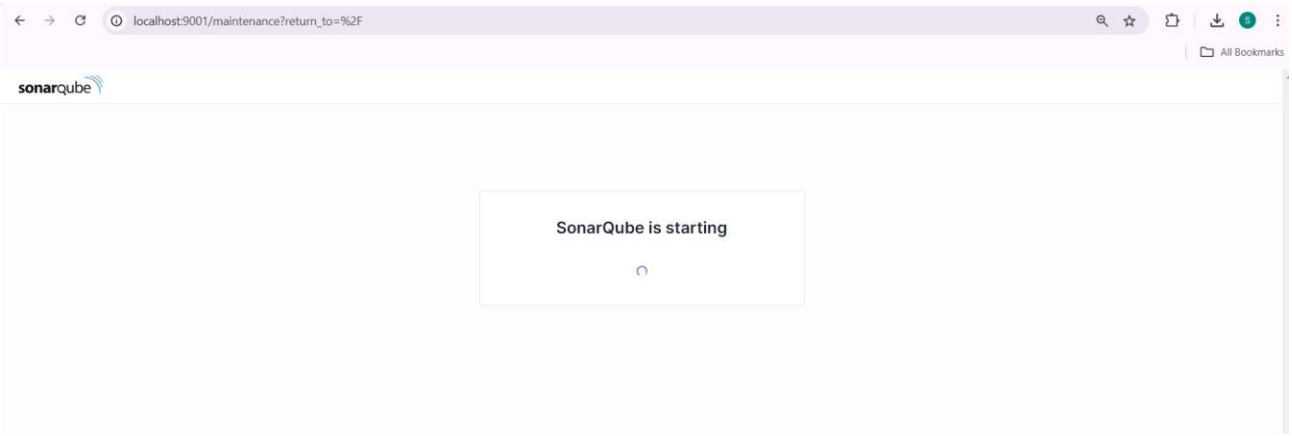
- Jenkins installed
- Docker Installed (for SonarQube)
- SonarQube Docker Image

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

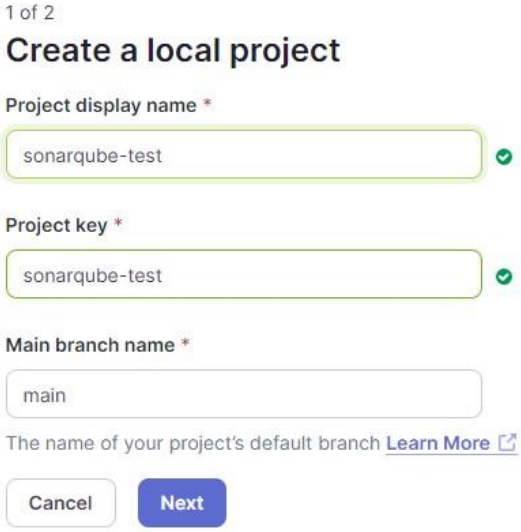
1. Open up Jenkins Dashboard on localhost, port 8080 or whichever port it is at for you.
2. Run SonarQube in a Docker container using this command –

```
C:\Users\ADMIN>docker run -d --name sonarqube2 -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9001:9000 sonarqube:latest
fda86b00e3989f3eb5aca8396b29b2a0adc95bcfe0fc5d85cf1237491e7678b9
```

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

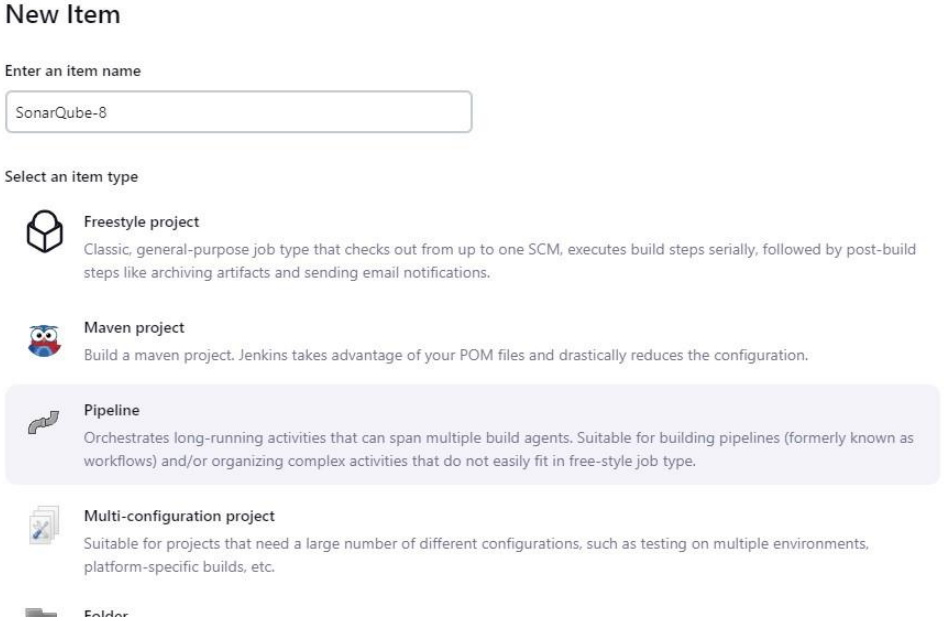


4. Login to SonarQube using username *admin* and password *admin*.
5. Create a manual project in SonarQube with the name **sonarqube-test**



Setup the project and come back to Jenkins Dashboard.

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



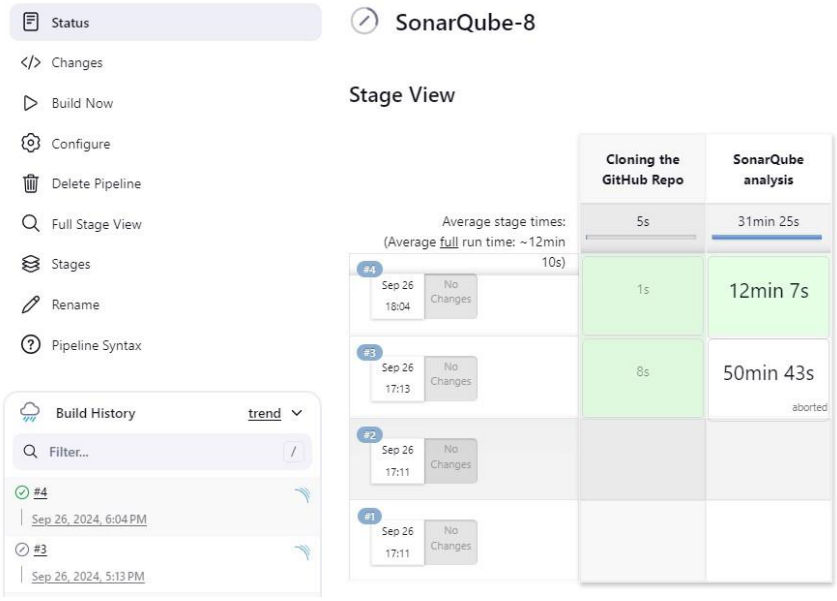
7. Under Pipeline Script, enter the following -

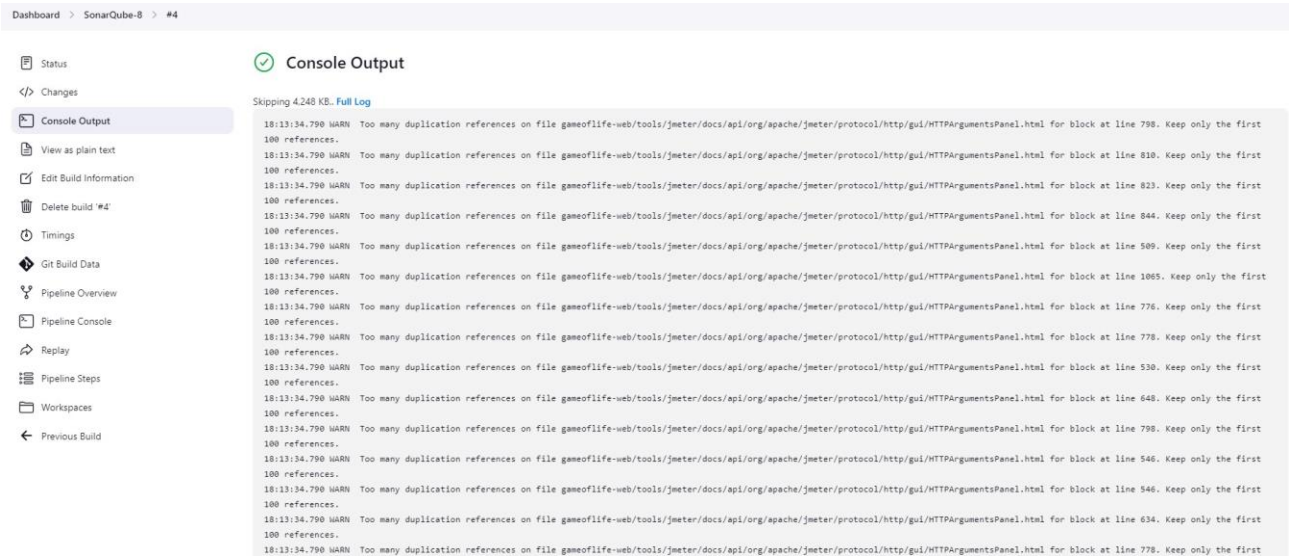
```
node {
  stage('Cloning the GitHub Repo') {
    git 'https://github.com/shazforiot/GOL.git'
  } stage('SonarQube
  analysis') {
    withSonarQubeEnv('sonarqube') {
      sh "<PATH_TO_SONARQUBE_FOLDER>//bin//sonar-scanner \
      -D sonar.login=<SonarQube_USERNAME> \
      -D sonar.password=<SonarQube_PASSWORD>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

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

8. Run The Build.

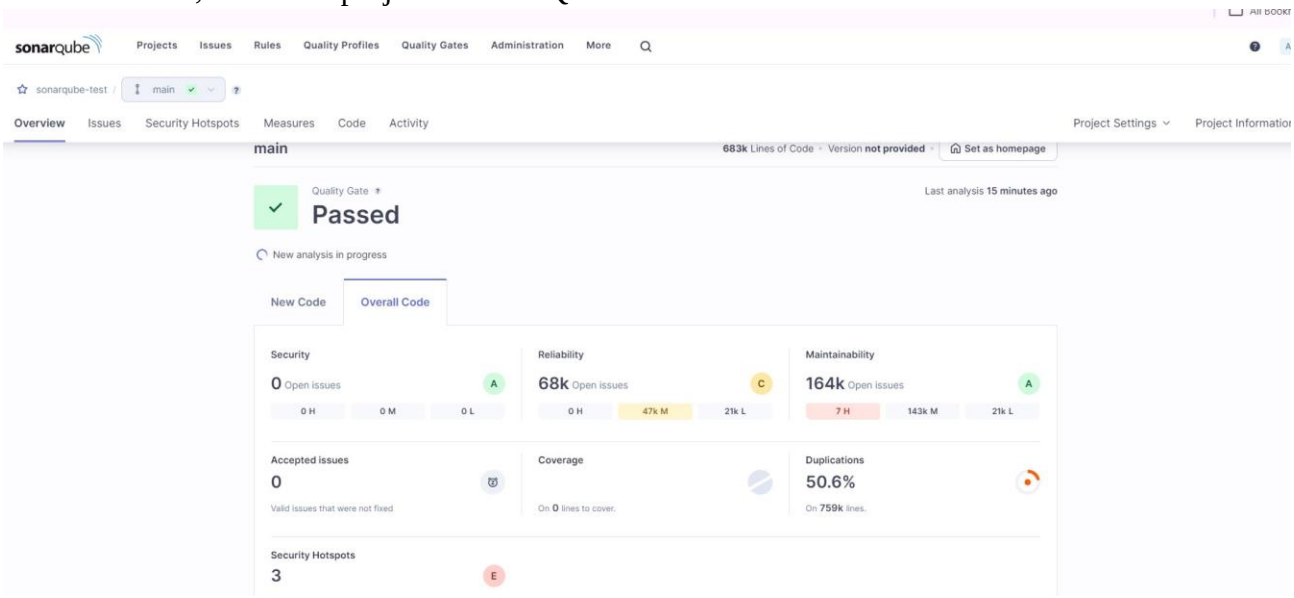
9. Check the console output once the build is complete.






```
references.
18:13:39.657 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.
18:13:39.657 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/gui/util/TextAreaCellRenderer.html for block at line 41. Keep only the first 100
references.
18:13:39.657 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/gui/util/TextAreaCellRenderer.html for block at line 17. Keep only the first 100
references.
18:13:39.657 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.
18:13:39.657 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.
18:13:39.657 INFO  CPD Executor CPD calculation finished (done) | time=158971ms
18:13:39.674 INFO  SCM revision ID 'ba799ba7e1b576f04a4612322b0412c5e6e1e5e4'
18:15:49.696 INFO  Analysis report generated in 5022ms, dir size=127.2 MB
18:16:08.759 INFO  Analysis report compressed in 19048ms, zip size=29.6 MB
18:16:09.884 INFO  Analysis report uploaded in 1125ms
18:16:09.887 INFO  ANALYSIS SUCCESSFUL, you can find the results at: http://127.0.0.1:9001/dashboard?id=sonarqube-test
18:16:09.887 INFO  Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
18:16:09.887 INFO  More about the report processing at http://127.0.0.1:9001/api/ce/task?id=6f22c333-3777-4a21-b058-0ab4c049625c
18:16:22.970 INFO  Analysis total time: 12:02.242 s
18:16:22.975 INFO  SonarScanner Engine completed successfully
18:16:23.699 INFO  EXECUTION SUCCESS
18:16:23.706 INFO  Total time: 12:05.758s
[Pipeline] }
[Pipeline] // withSonarQubeEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

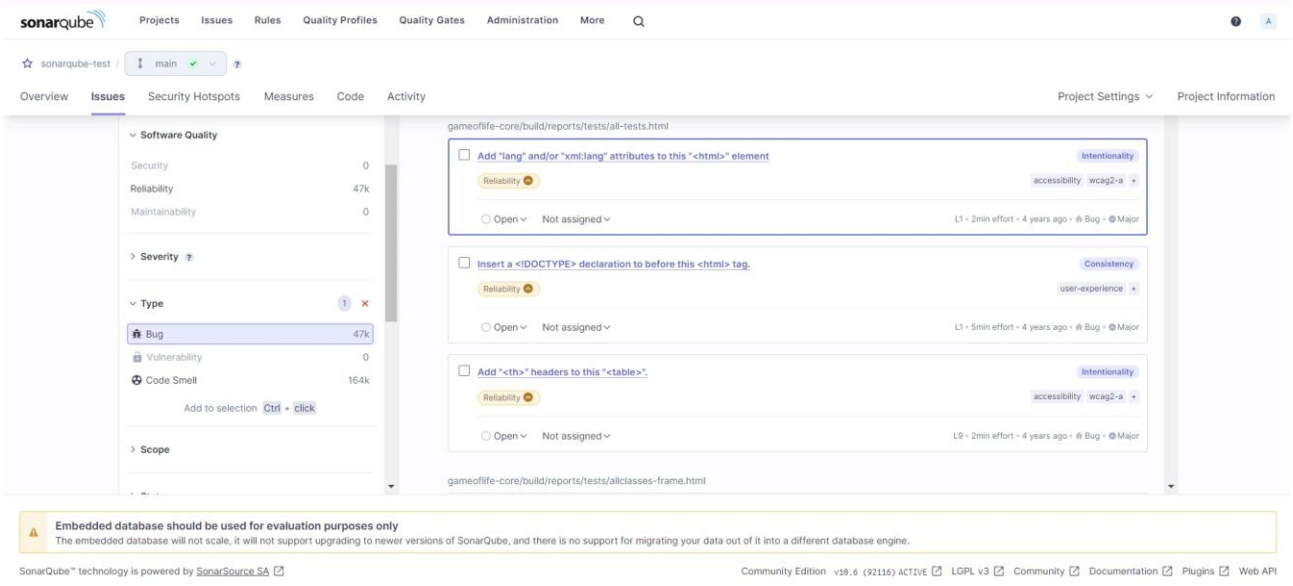
10. After that, check the project in SonarQube.



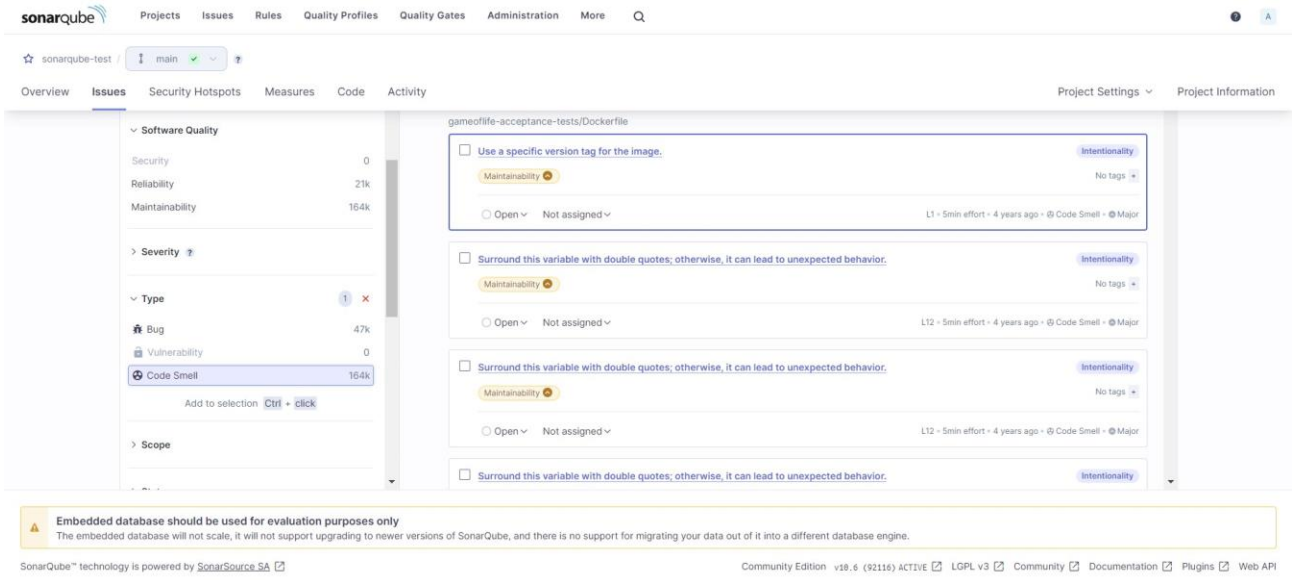
Under different tabs, check all different issues with the code.

11. **Code Problems -**

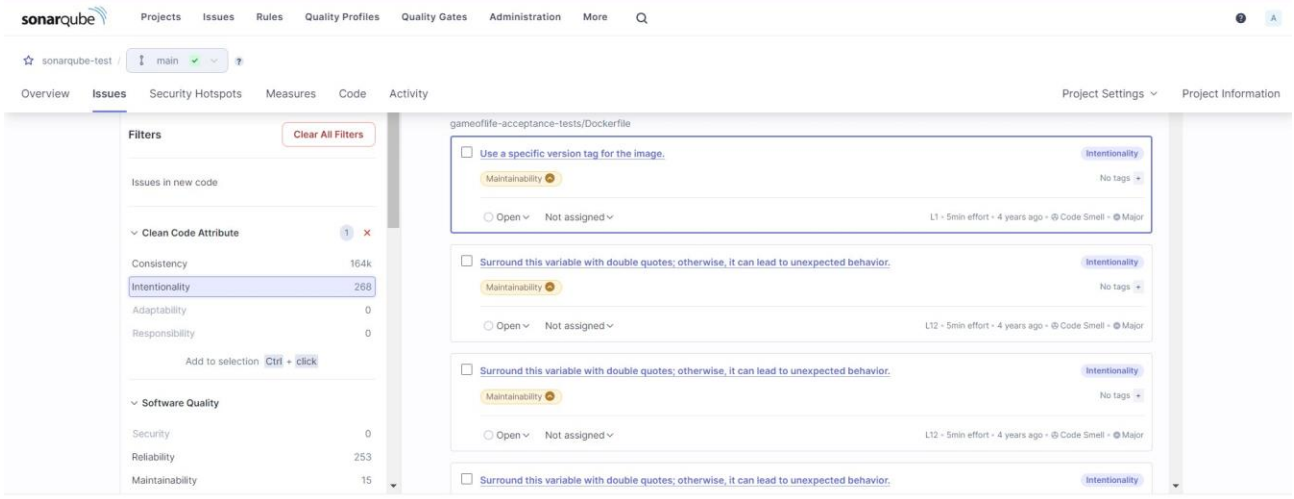
Bugs



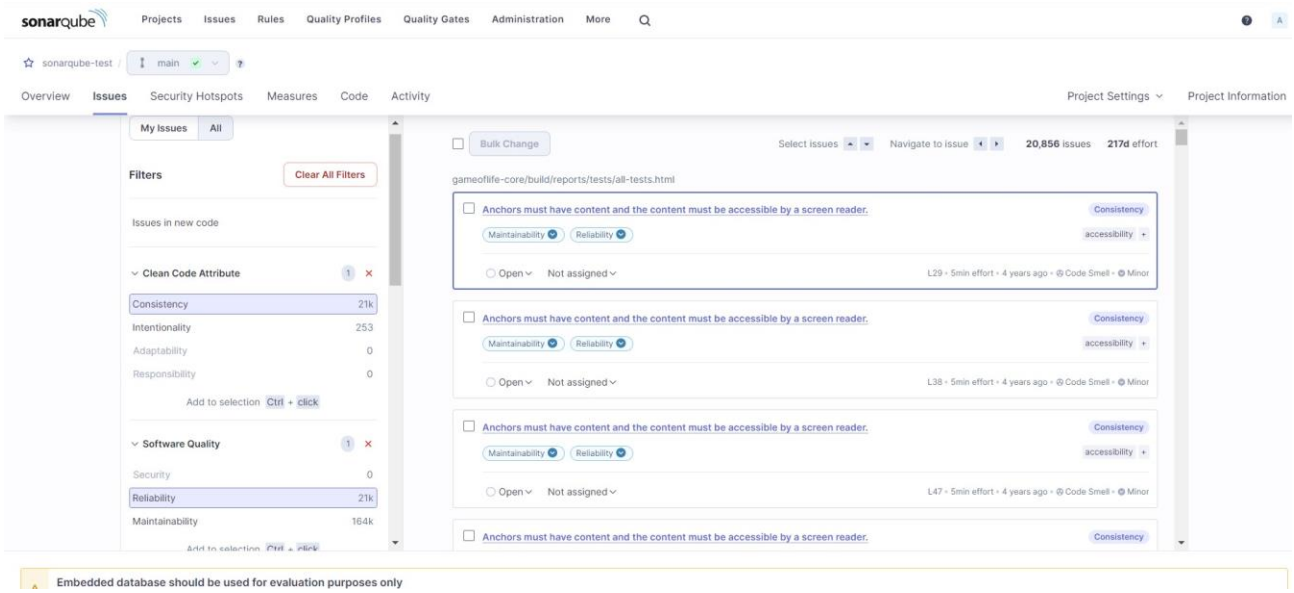
Code Smells



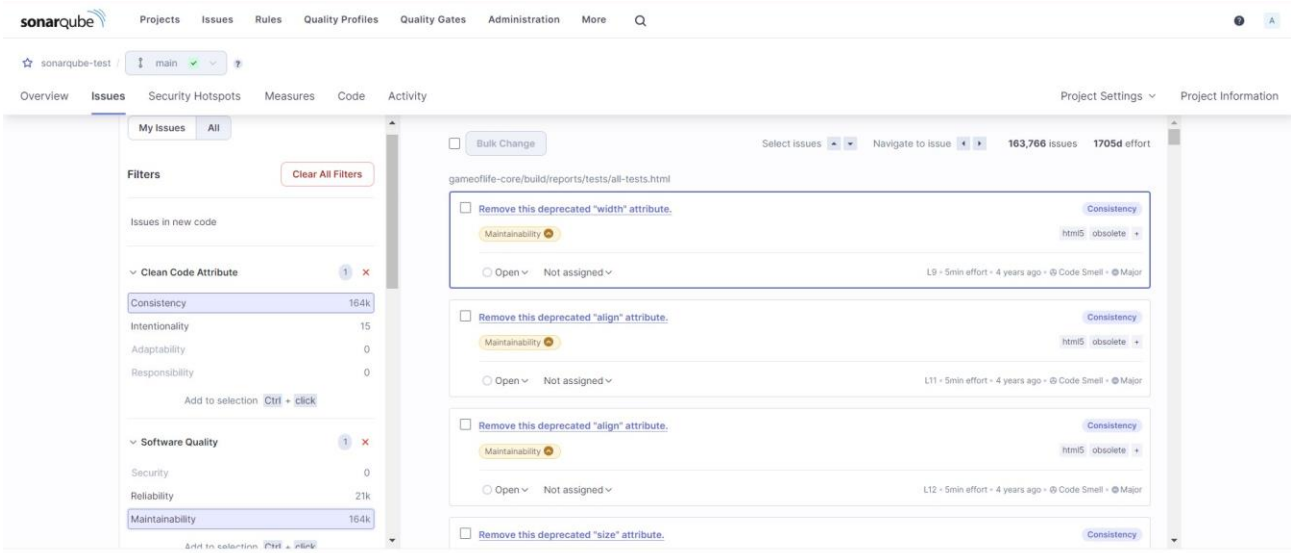
Intentional Issues



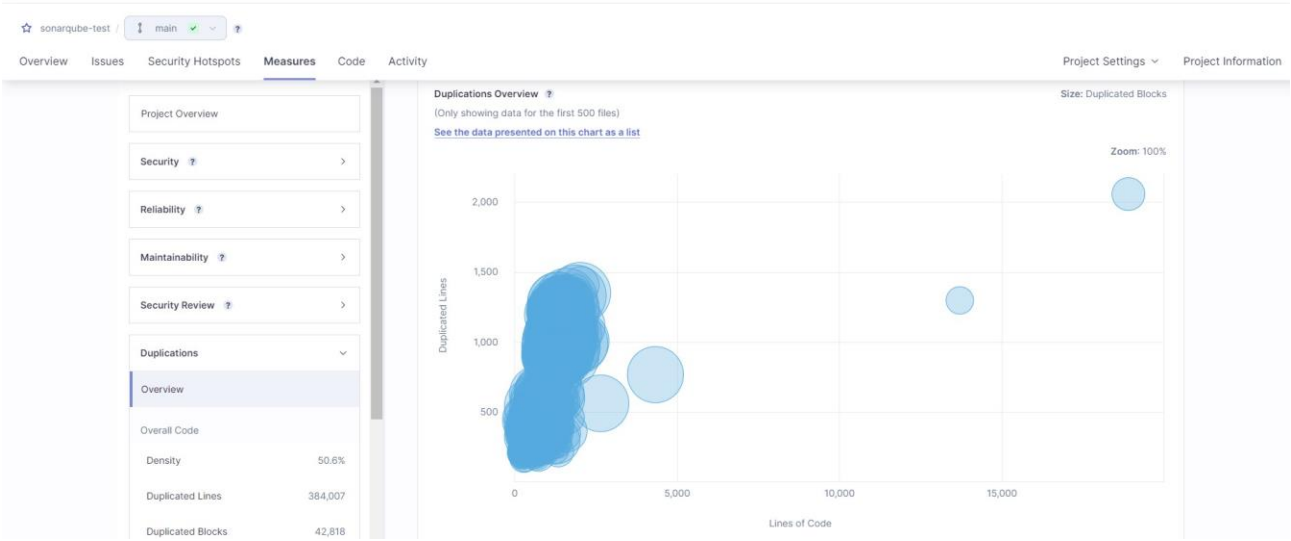
Reliability Issue



Maintainability Issue



Duplicates



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

Conclusion:

In this experiment, I successfully created a CI/CD pipeline using Jenkins integrated with SonarQube for static code analysis on a sample Java application. I set up SonarQube in a Docker container and configured Jenkins to clone the GitHub repository and perform the analysis. The pipeline detected various issues, including bugs, code smells, and security vulnerabilities, which I reviewed in SonarQube. This experience enhanced my skills in configuring CI/CD tools and highlighted the importance of maintaining code quality through automation. Overall, I gained valuable insights into integrating tools for effective software development practices.

Name: Abhinav Swaminathan Div: D15C Roll No: 01

Adv DevOps Exp 09

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

Theory:

What is Nagios?

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

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

Why We Need Nagios tool?

Here are the important reasons to use Nagios monitoring tool:

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

Features of Nagios

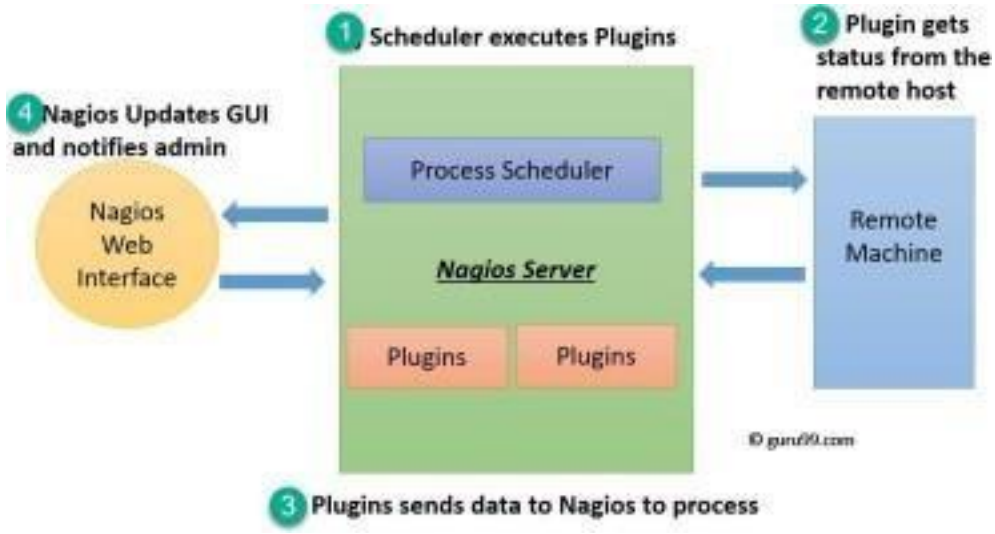
Following are the important features of Nagios monitoring tool:

- Relatively scalable, Manageable, and Secure
- Good log and database system
- Informative and attractive web interfaces
- Automatically send alerts if condition changes
- If the services are running fine, then there is no need to do check that host is an alive
- Helps you to detect network errors or server crashes ● You can troubleshoot the performance issues of the server.
- The issues, if any, can be fixed automatically as they are identified during the monitoring process
- You can monitor the entire business process and IT infrastructure with a single pass
- The product's architecture is easy to write new plugins in the language of your choice
- Nagios allows you to read its configuration from an entire directory which helps you to decide how to define individual files

- Utilizes topology to determine dependencies
- Monitor network services like HTTP, SMTP, HTTP, SNMP, FTP, SSH, POP, etc.
- Helps you to define network host hierarchy using parent hosts
- Ability to define event handlers that runs during service or host events for proactive problem resolution
- Support for implementing redundant monitoring hosts

Nagios Architecture

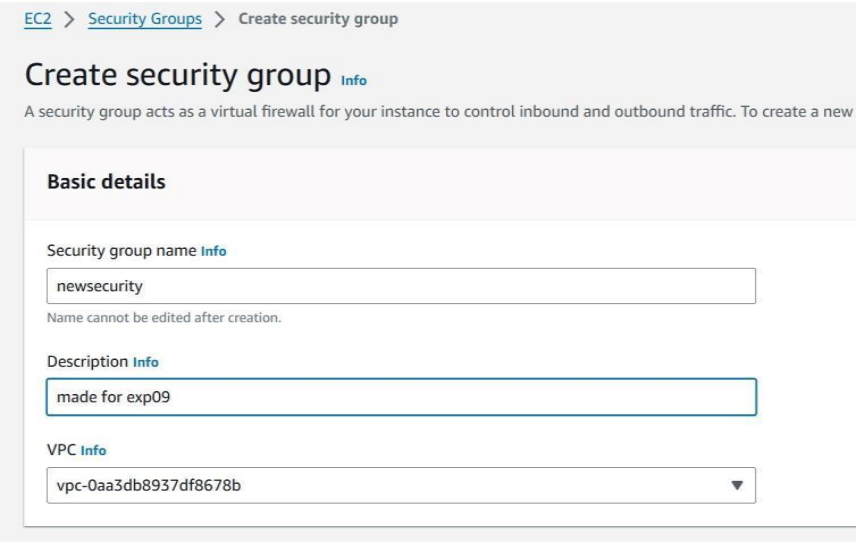
Nagios is a client-server architecture. Usually, on a network, a Nagios server is running on a host, and plugins are running on all the remote hosts which should be monitored.



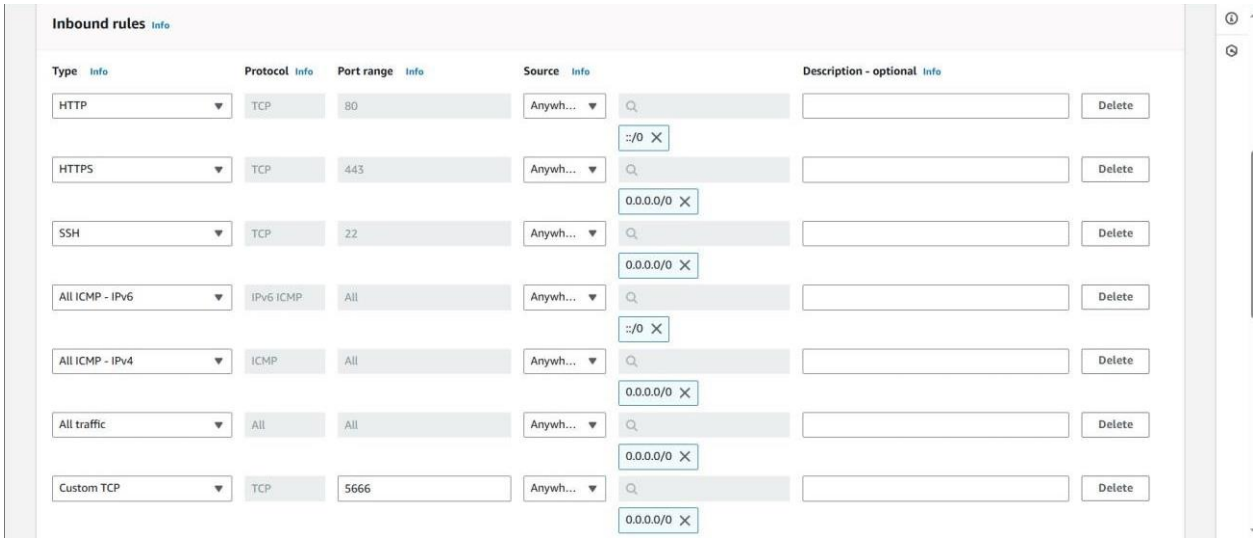
1. The scheduler is a component of the server part of Nagios. It sends a signal to execute the plugins at the remote host.
2. The plugin gets the status from the remote host
3. The plugin sends the data to the process scheduler
4. The process scheduler updates the GUI and notifications are sent to admins.

Name: Abhinav Swaminathan Div: D15C Roll No: 01

Step 1: Create a security group with the required configurations I have created a new security group with a name ‘newsecurity’



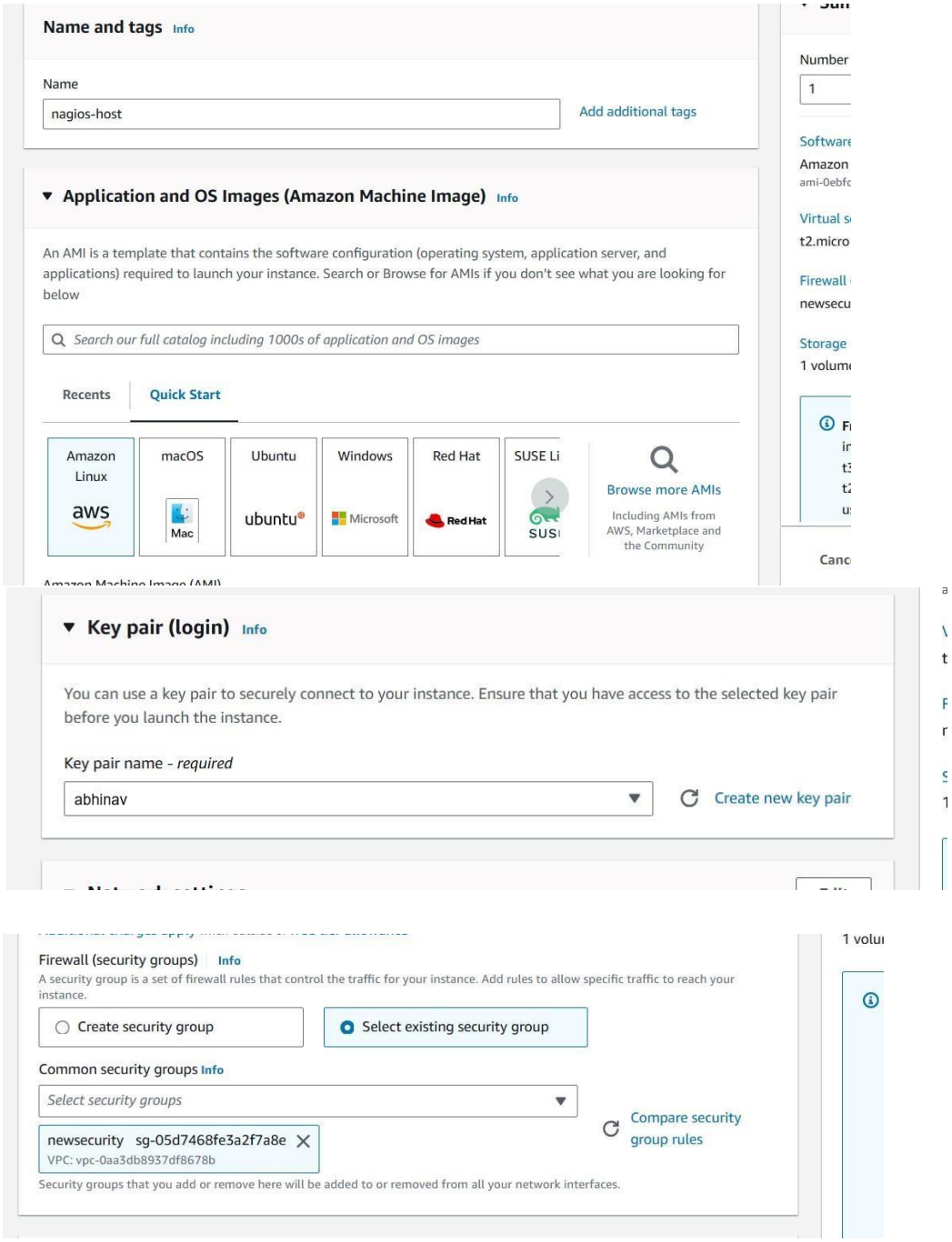
I have modified the INBOUND RULES as follows



Step 2: Create ec2 instance

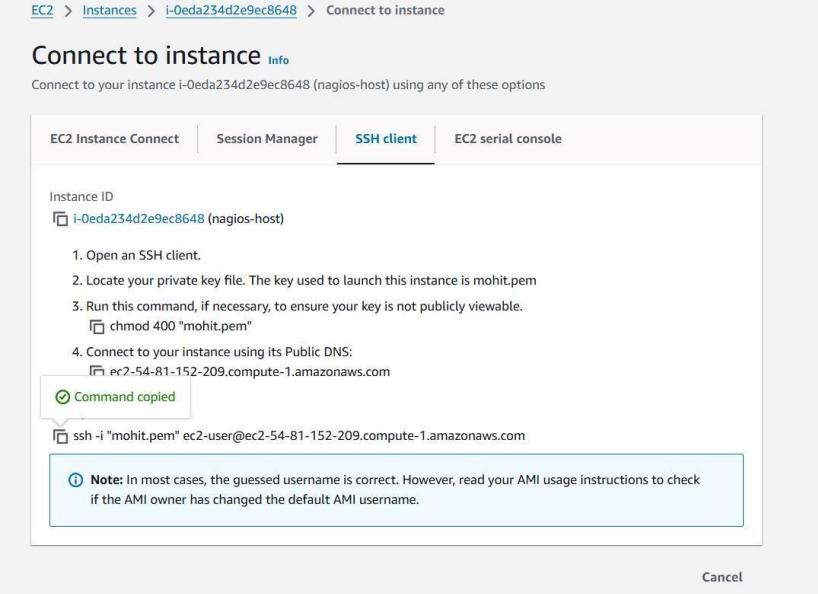
Name it as nagios-host. Select instance type as amazon-linux and choose the already created key pair and security group

Name: Abhinav Swaminathan Div: D15C Roll No: 01

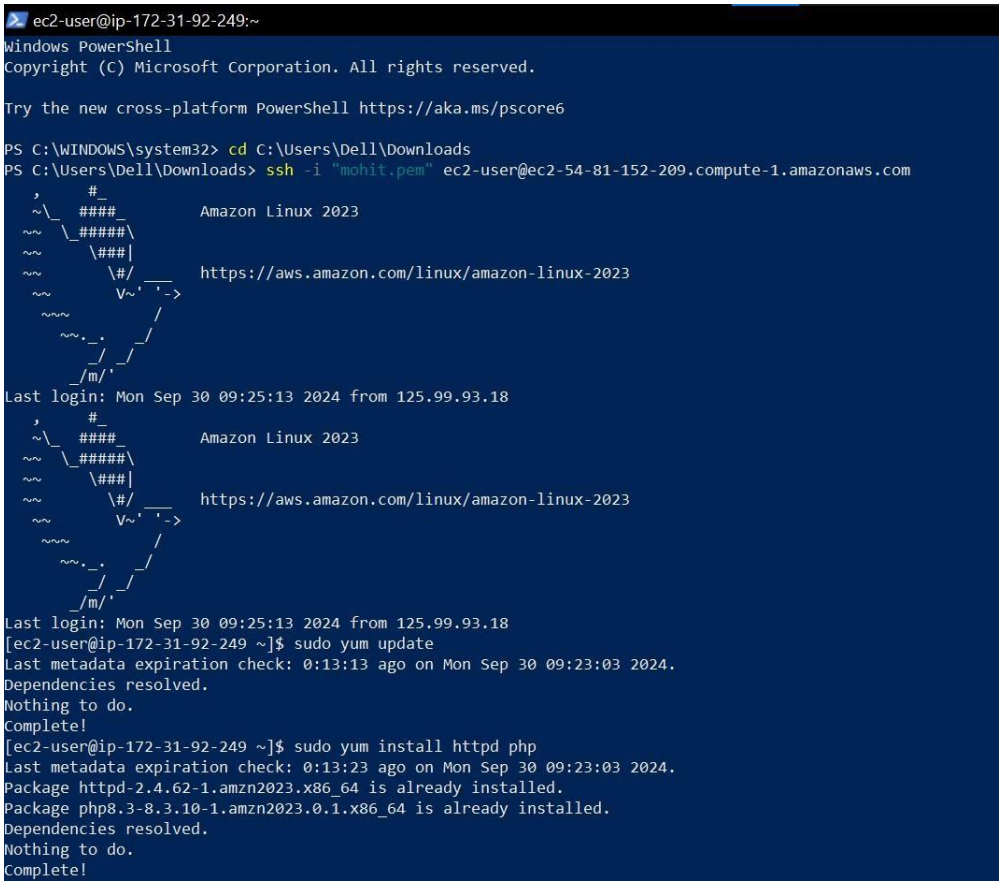


Copy the given ssh command, as we will require it for logging into our nagios-host instance from our windows powershell

Name: Abhinav Swaminathan Div: D15C Roll No: 01



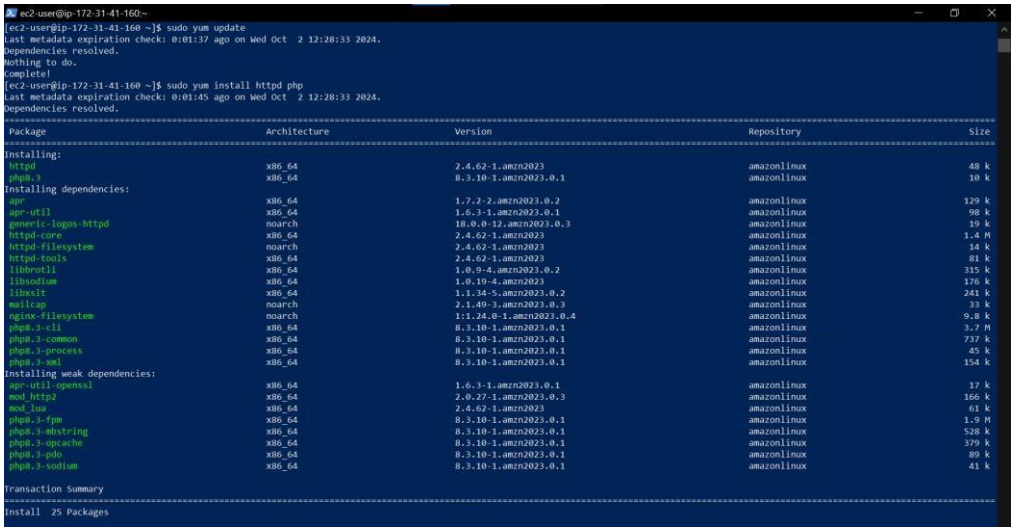
Step 3: Open an administrative powershell and remotely login using the above mentioned ssh command



And then run these commands

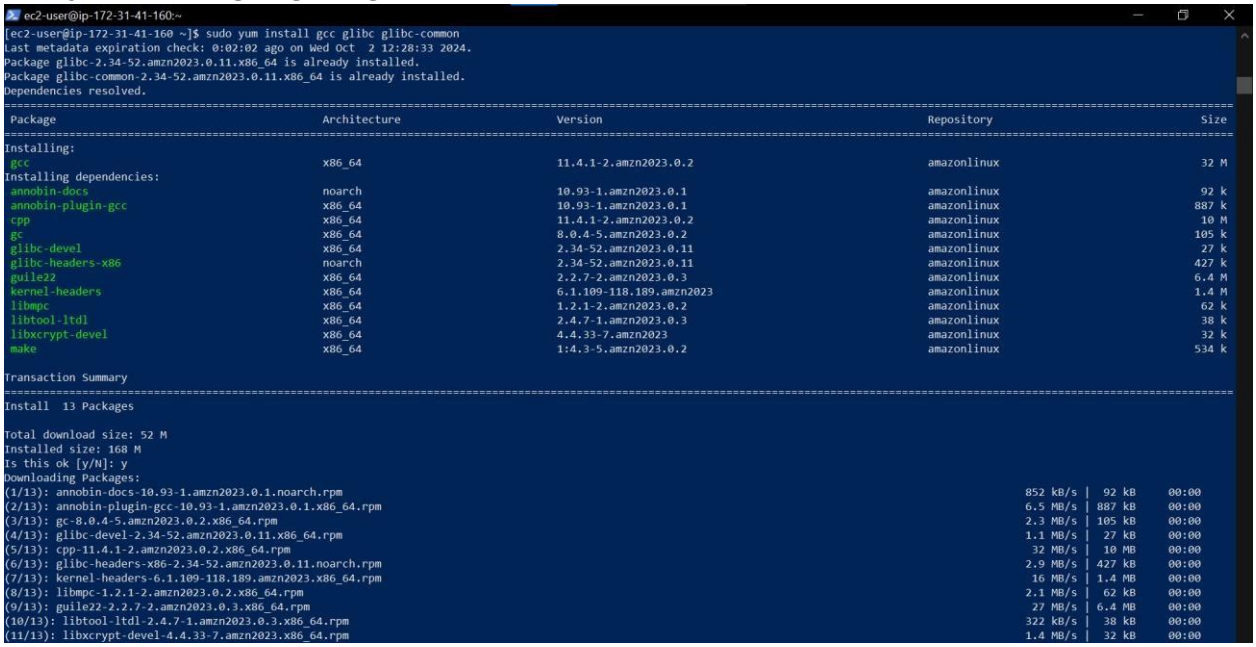
Name: Abhinav Swaminathan Div: D15C Roll No: 01

sudo yum update sudo yum install httpd php



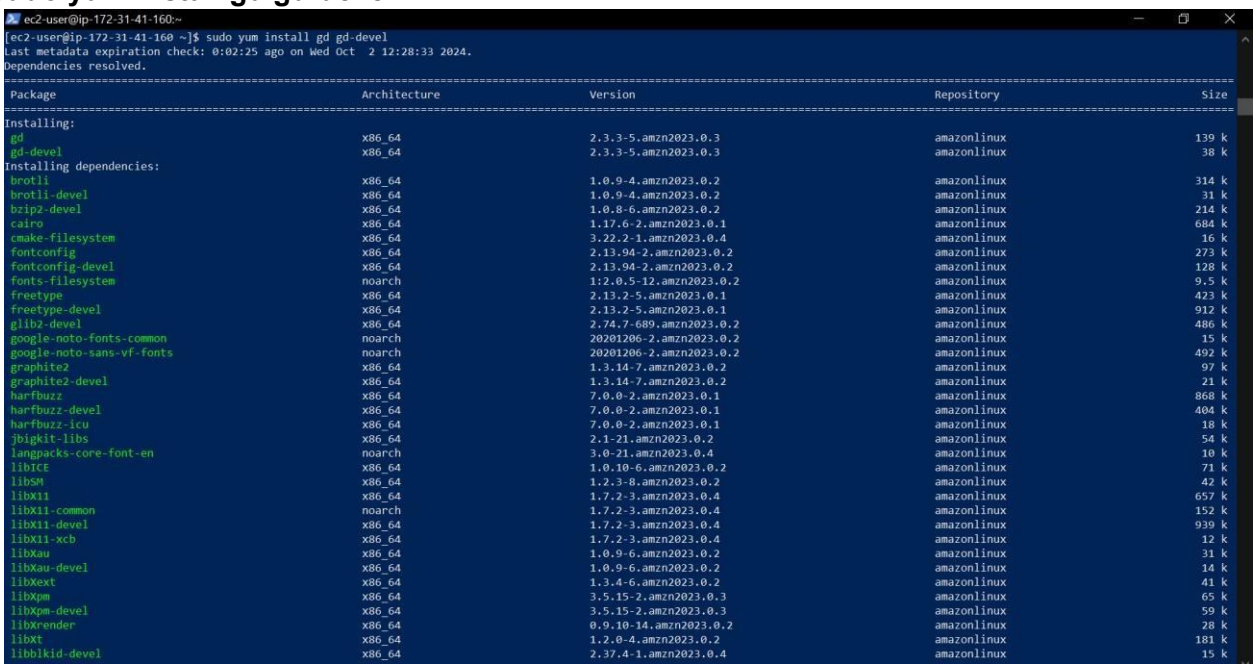
```
ec2-user@ip-172-31-41-160:~
[ec2-user@ip-172-31-41-160 ~]$ sudo yum update
Last metadata expiration check: 0:01:37 ago on Wed Oct  2 12:28:33 2024.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-41-160 ~]$ sudo yum install httpd php
Last metadata expiration check: 0:01:45 ago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

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

sudo yum install gcc glibc glibc-common



```
ec2-user@ip-172-31-41-160:~
[ec2-user@ip-172-31-41-160 ~]$ sudo yum install gcc glibc glibc-common
Last metadata expiration check: 0:02:02 ago on Wed Oct  2 12:28:33 2024.
Package glibc-2.34-52.amzn2023.0.11.x86_64 is already installed.
Package glibc-common-2.34-52.amzn2023.0.11.x86_64 is already installed.
Dependencies resolved.
================================================================================
 Package                 Architecture  Version                      Repository      Size
================================================================================
Installing:
 gcc                     x86_64        11.4.1-2.amzn2023.0.2        amazonlinux     32 M
Installing dependencies:
 annobin-docs            noarch        10.93-1.amzn2023.0.1         amazonlinux     92 k
 annobin-plugin-gcc      x86_64        10.93-1.amzn2023.0.1         amazonlinux    887 k
 cpp                     x86_64        11.4.1-2.amzn2023.0.2        amazonlinux     10 M
 gc                      x86_64        8.0.4-5.amzn2023.0.2         amazonlinux    105 k
 glibc-devel             x86_64        2.34-52.amzn2023.0.11        amazonlinux     27 k
 glibc-headers-x86       noarch        2.34-52.amzn2023.0.11        amazonlinux    427 k
 guile22                 x86_64        2.2.7-2.amzn2023.0.3         amazonlinux    6.4 M
 kernel-headers          x86_64        6.1.109-118.189.amzn2023     amazonlinux    1.4 M
 libmpc                  x86_64        1.2.1-2.amzn2023.0.2         amazonlinux     62 k
 libtool-ltdl            x86_64        2.4.7-1.amzn2023.0.3         amazonlinux     38 k
 libxcrypt-devel         x86_64        4.4.33-7.amzn2023            amazonlinux     32 k
 make                    x86_64        1:4.3-5.amzn2023.0.2         amazonlinux    534 k

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

Total download size: 52 M
Installed size: 168 M
Is this ok [y/N]: y
Downloading Packages:
(1/13): annobin-docs-10.93-1.amzn2023.0.1.noarch.rpm          852 kB/s |  92 kB     00:00
(2/13): annobin-plugin-gcc-10.93-1.amzn2023.0.1.x86_64.rpm    6.5 MB/s | 887 kB     00:00
(3/13): gc-8.0.4-5.amzn2023.0.2.x86_64.rpm                    2.3 MB/s | 105 kB     00:00
(4/13): glibc-devel-2.34-52.amzn2023.0.11.x86_64.rpm          1.1 MB/s |  27 kB     00:00
(5/13): cpp-11.4.1-2.amzn2023.0.2.x86_64.rpm                   32 MB/s |  10 MB     00:00
(6/13): glibc-headers-x86-2.34-52.amzn2023.0.11.noarch.rpm    2.9 MB/s | 427 kB     00:00
(7/13): kernel-headers-6.1.109-118.189.amzn2023.x86_64.rpm     16 MB/s | 1.4 MB     00:00
(8/13): libmpc-1.2.1-2.amzn2023.0.2.x86_64.rpm                2.1 MB/s |  62 kB     00:00
(9/13): guile22-2.2.7-2.amzn2023.0.3.x86_64.rpm                27 MB/s | 6.4 MB     00:00
(10/13): libtool-ltdl-2.4.7-1.amzn2023.0.3.x86_64.rpm         322 kB/s |  38 kB     00:00
(11/13): libxcrypt-devel-4.4.33-7.amzn2023.x86_64.rpm         1.4 MB/s |  32 kB     00:00
```

sudo yum install gd gd-devel

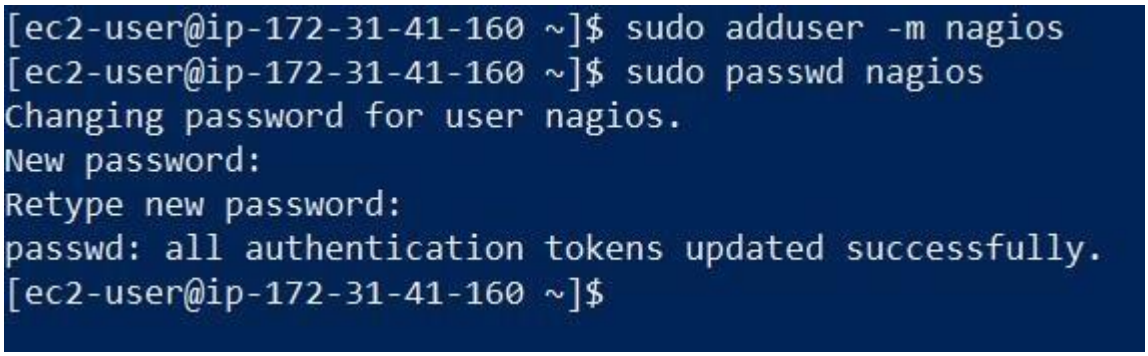


```
ec2-user@ip-172-31-41-160:~
[ec2-user@ip-172-31-41-160 ~]$ sudo yum install gd gd-devel
Last metadata expiration check: 0:02:25 ago on Wed Oct  2 12:28:33 2024.
Dependencies resolved.
```

Package	Architecture	Version	Repository	Size
Installing:				
gd	x86_64	2.3.3-5.amzn2023.0.3	amazonlinux	139 k
gd-devel	x86_64	2.3.3-5.amzn2023.0.3	amazonlinux	38 k
Installing dependencies:				
brotli	x86_64	1.0.9-4.amzn2023.0.2	amazonlinux	314 k
brotli-devel	x86_64	1.0.9-4.amzn2023.0.2	amazonlinux	31 k
bzip2-devel	x86_64	1.0.8-6.amzn2023.0.2	amazonlinux	214 k
cairo	x86_64	1.17.6-2.amzn2023.0.1	amazonlinux	684 k
cmake-filesystem	x86_64	3.22.2-1.amzn2023.0.4	amazonlinux	16 k
fontconfig	x86_64	2.13.94-2.amzn2023.0.2	amazonlinux	273 k
fontconfig-devel	x86_64	2.13.94-2.amzn2023.0.2	amazonlinux	128 k
fonts-filesystem	noarch	1:2.0.5-12.amzn2023.0.2	amazonlinux	9.5 k
freetype	x86_64	2.13.2-5.amzn2023.0.1	amazonlinux	423 k
freetype-devel	x86_64	2.13.2-5.amzn2023.0.1	amazonlinux	912 k
glib2-devel	x86_64	2.74.7-689.amzn2023.0.2	amazonlinux	486 k
google-noto-fonts-common	noarch	20201206-2.amzn2023.0.2	amazonlinux	15 k
google-noto-sans-vf-fonts	noarch	20201206-2.amzn2023.0.2	amazonlinux	492 k
graphite2	x86_64	1.3.14-7.amzn2023.0.2	amazonlinux	97 k
graphite2-devel	x86_64	1.3.14-7.amzn2023.0.2	amazonlinux	21 k
harfbuzz	x86_64	7.0.0-2.amzn2023.0.1	amazonlinux	868 k
harfbuzz-devel	x86_64	7.0.0-2.amzn2023.0.1	amazonlinux	404 k
harfbuzz-icu	x86_64	7.0.0-2.amzn2023.0.1	amazonlinux	18 k
jbigkit-libs	x86_64	2.1-21.amzn2023.0.2	amazonlinux	54 k
langpacks-core-font-en	noarch	3.0-21.amzn2023.0.4	amazonlinux	10 k
libICE	x86_64	1.0.10-6.amzn2023.0.2	amazonlinux	71 k
libSM	x86_64	1.2.3-8.amzn2023.0.2	amazonlinux	42 k
libX11	x86_64	1.7.2-3.amzn2023.0.4	amazonlinux	657 k
libX11-common	noarch	1.7.2-3.amzn2023.0.4	amazonlinux	152 k
libX11-devel	x86_64	1.7.2-3.amzn2023.0.4	amazonlinux	939 k
libX11-xcb	x86_64	1.7.2-3.amzn2023.0.4	amazonlinux	12 k
libXau	x86_64	1.0.9-6.amzn2023.0.2	amazonlinux	31 k
libXau-devel	x86_64	1.0.9-6.amzn2023.0.2	amazonlinux	14 k
libXext	x86_64	1.3.4-6.amzn2023.0.2	amazonlinux	41 k
libXpm	x86_64	3.5.15-2.amzn2023.0.3	amazonlinux	65 k
libXpm-devel	x86_64	3.5.15-2.amzn2023.0.3	amazonlinux	59 k
libXrender	x86_64	0.9.10-14.amzn2023.0.2	amazonlinux	28 k
libXt	x86_64	1.2.0-4.amzn2023.0.2	amazonlinux	181 k
libblkid-devel	x86_64	2.37.4-1.amzn2023.0.4	amazonlinux	15 k

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

sudo adduser -m nagios sudo
passwd nagios

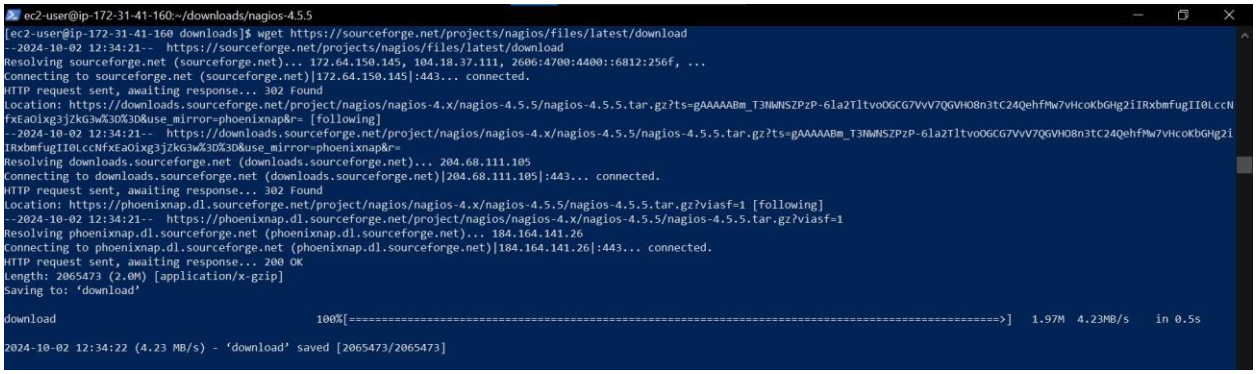


```
[ec2-user@ip-172-31-41-160 ~]$ sudo adduser -m nagios
[ec2-user@ip-172-31-41-160 ~]$ sudo passwd nagios
Changing password for user nagios.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[ec2-user@ip-172-31-41-160 ~]$
```

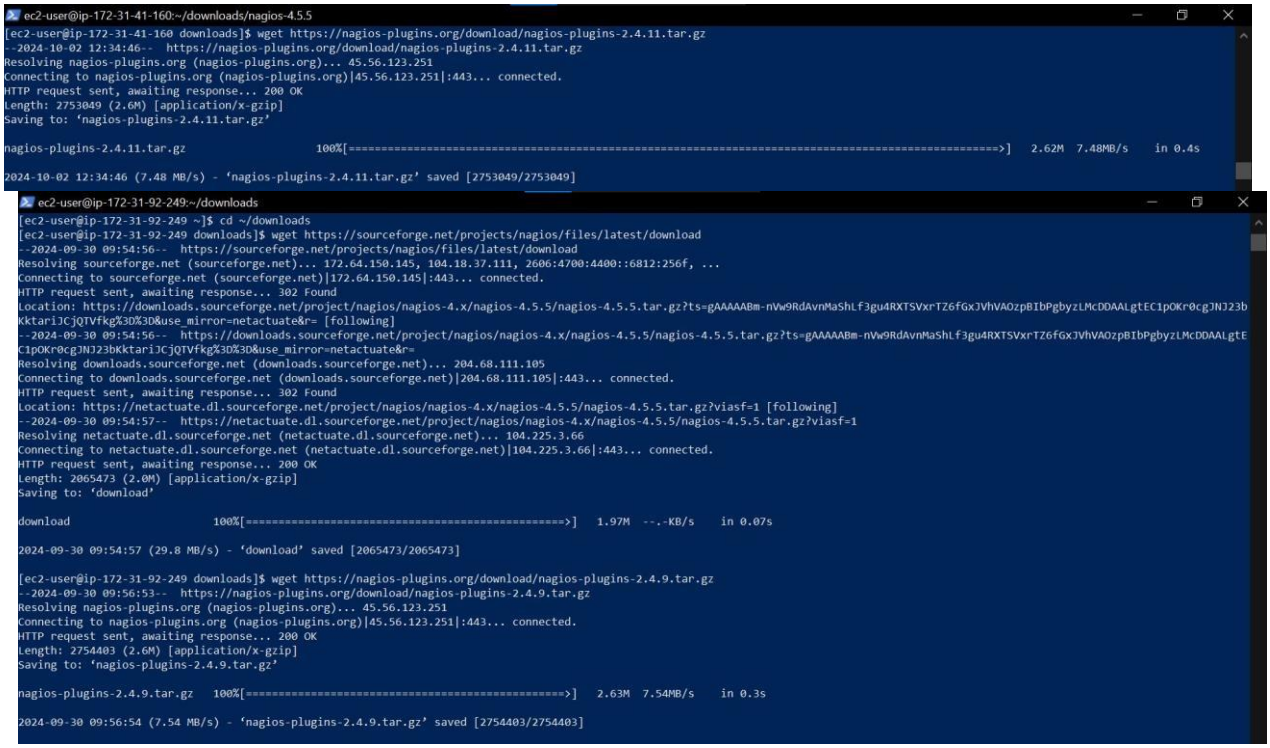
Create a new user group & create a new directory for Nagios downloads using the following commands **sudo groupadd nagcmd sudo usermod -a -G nagcmd nagios sudo usermod -a -G nagcmd apache mkdir ~/downloads cd ~/downloads**

Use **wget** to download the source zip files.
In this step, we are downloading, the latest version of nagios and the necessary plugins required to carry out the tasks of setting up a nagios server wget
https://sourceforge.net/projects/nagios/files/latest/download

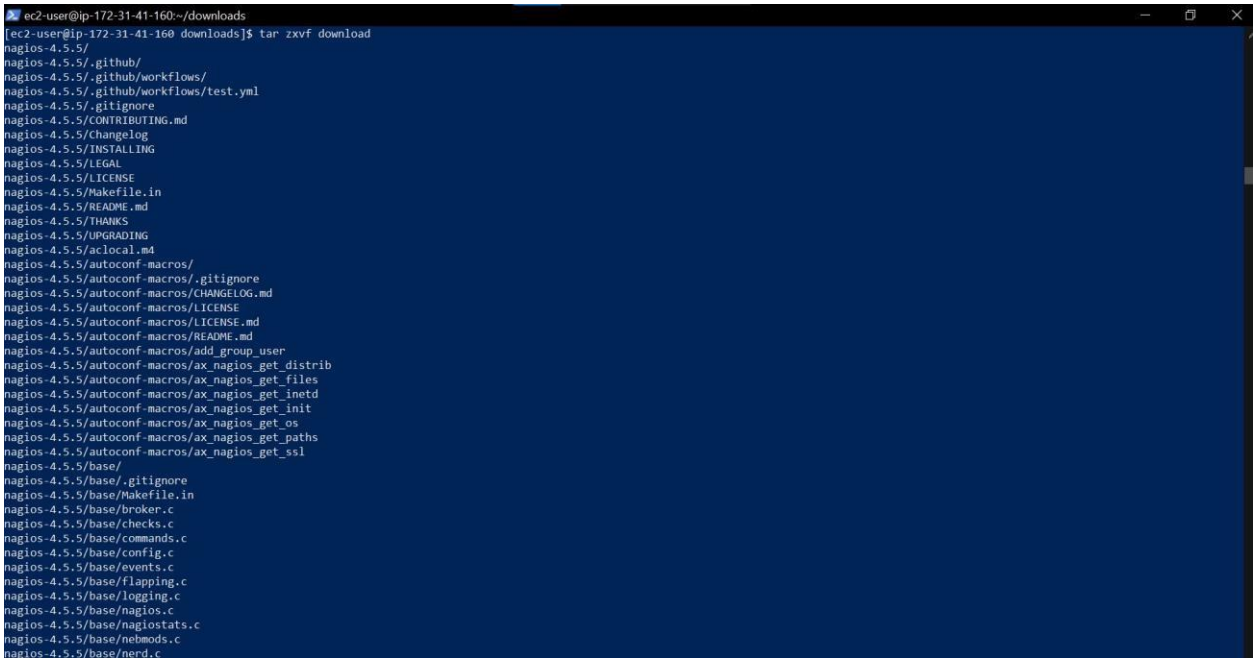
Name: Abhinav Swaminathan Div: D15C Roll No: 01



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



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



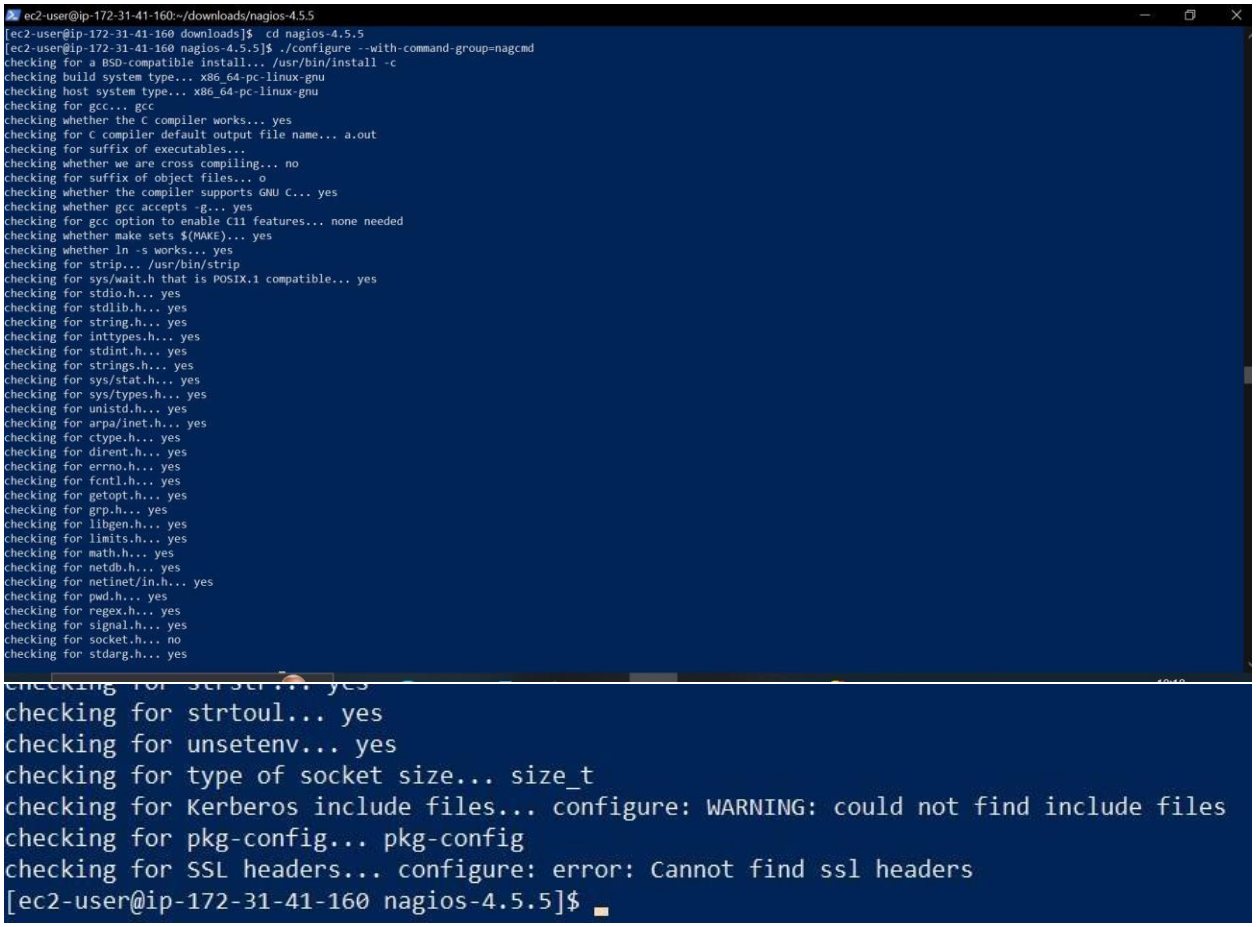
After which we are supposed to **change our directory** over there

For eg. **cd nagios-4.5.5**… depending on the version that we have downloaded

Next, Run this command (make sure that you are working inside nagios-4.x.x directory)

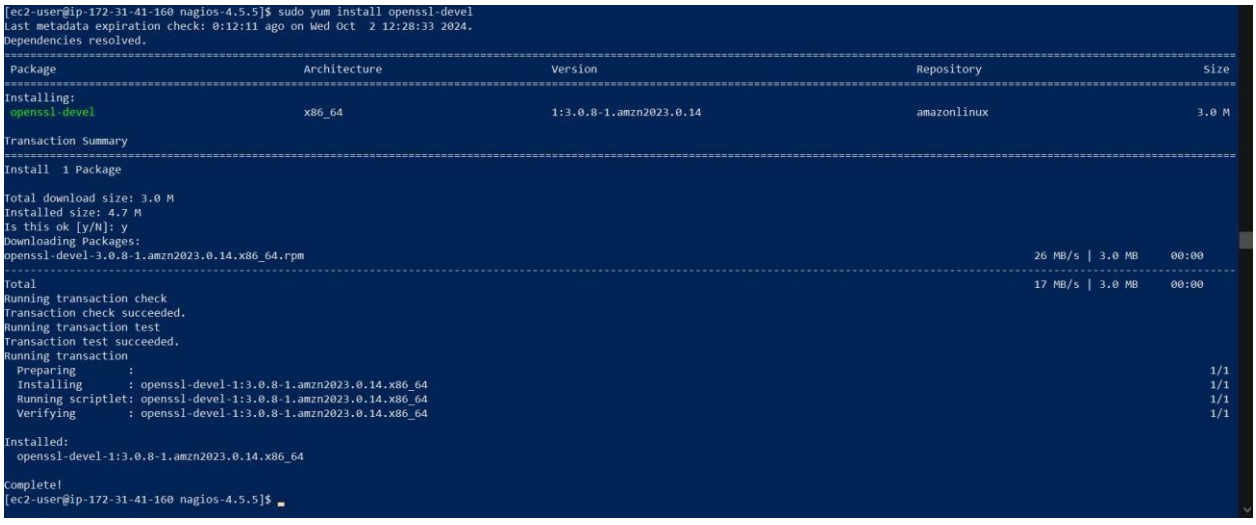
./configure --with-command-group=nagcmd

Name: Abhinav Swaminathan Div: D15C Roll No: 01



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

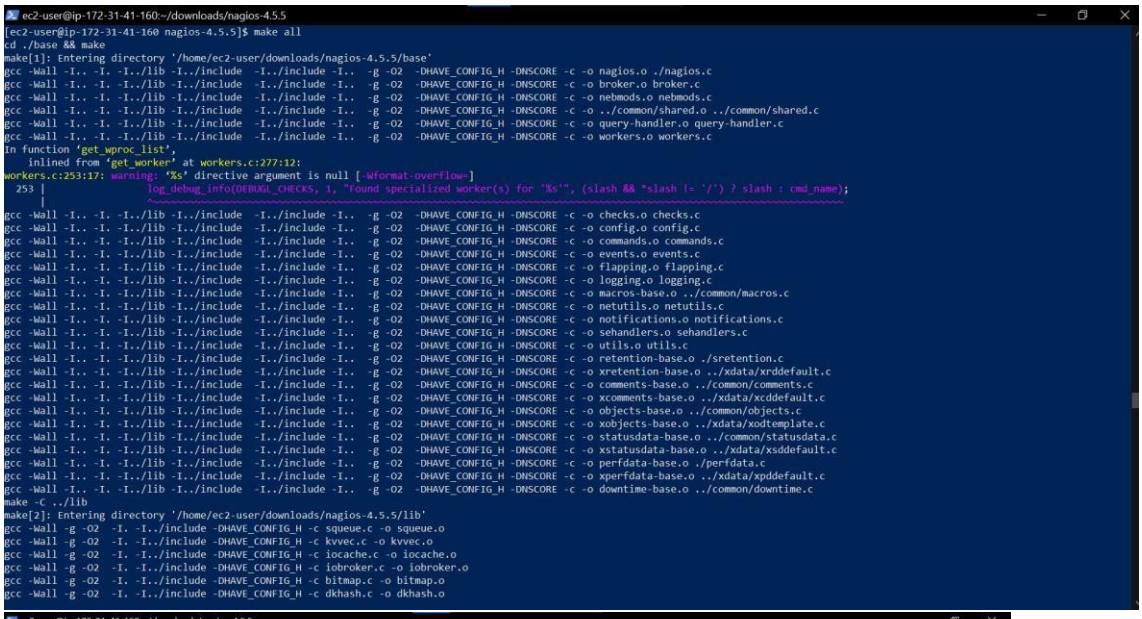
For that purpose, we are to run the following command. **sudo yum install openssl-devel** (for ssl header)

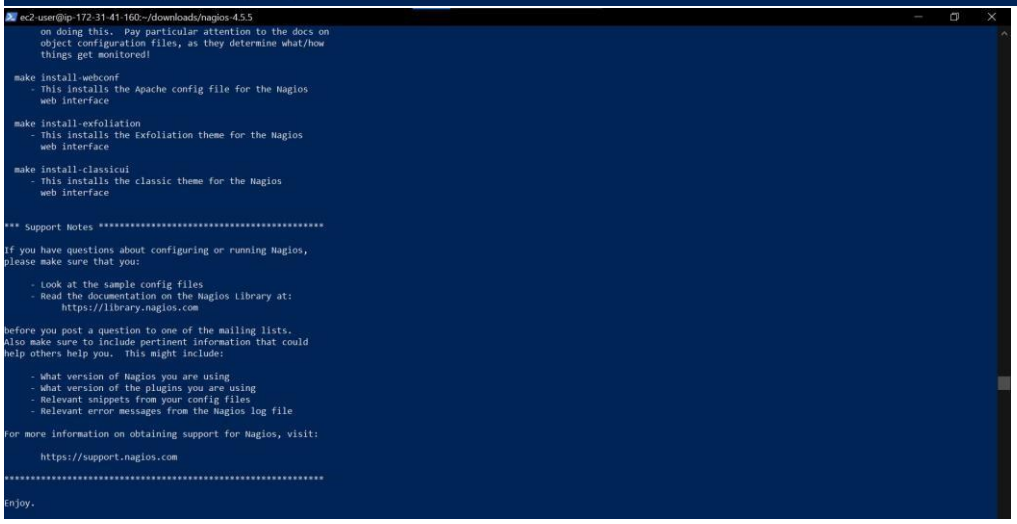


Now, Re-run **./configure --with-command-group=nagcmd**

After this, run **make all** command

Name: Abhinav Swaminathan Div: D15C Roll No: 01





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

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

sudo make install-init

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

sudo make install-config

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

*** Config files installed ***

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

sudo make install-webconf

```
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ sudo make install-webconf
/usr/bin/install -c -m 644 sample-config/httpd.conf /etc/httpd/conf.d/nagios.conf
if [ 0 -eq 1 ]; then \
        ln -s /etc/httpd/conf.d/nagios.conf /etc/apache2/sites-enabled/nagios.conf; \
fi

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

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

Next, we are supposed to create a nagiosadmin account for nagios login along with password. Specify the password twice. **sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin**

```
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
New password:
Re-type new password:
Adding password for user nagiosadmin
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$
```

Restart Apache **sudo service httpd restart**

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

cd ~/downloads tar zxvf nagios-plugins-2.4.11.tar.gz

```
ec2-user@ip-172-31-41-160:~/downloads
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ sudo service httpd restart
Redirecting to /bin/systemctl restart httpd.service
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ cd ~/downloads
[ec2-user@ip-172-31-41-160 downloads]$ tar zxvf nagios-plugins-2.4.11.tar.gz
nagios-plugins-2.4.11/
nagios-plugins-2.4.11/build-aux/
nagios-plugins-2.4.11/build-aux/compile
nagios-plugins-2.4.11/build-aux/config.guess
nagios-plugins-2.4.11/build-aux/config.rpath
nagios-plugins-2.4.11/build-aux/config.sub
nagios-plugins-2.4.11/build-aux/install-sh
nagios-plugins-2.4.11/build-aux/ltmain.sh
nagios-plugins-2.4.11/build-aux/missing
nagios-plugins-2.4.11/build-aux/mkinstalldirs
nagios-plugins-2.4.11/build-aux/depcomp
nagios-plugins-2.4.11/build-aux/snippet/
nagios-plugins-2.4.11/build-aux/snippet/_Noreturn.h
nagios-plugins-2.4.11/build-aux/snippet/arg-nonnull.h
nagios-plugins-2.4.11/build-aux/snippet/c++defs.h
nagios-plugins-2.4.11/build-aux/snippet/warn-on-use.h
nagios-plugins-2.4.11/build-aux/test-driver
```

Compile and install plugins **cd nagios-plugins-2.4.11**

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

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

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

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

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

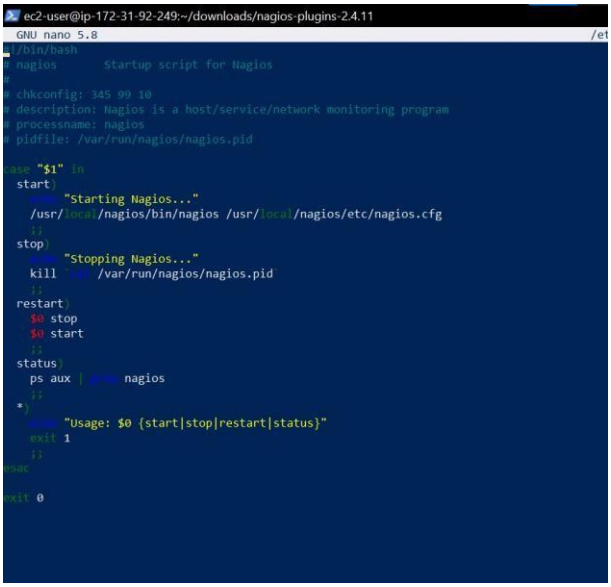
With ls command, we must see a file named nagios, which i was not able to see
If the Init Script is Missing i.e If you don't see the nagios script in /etc/init.d/, you can create it manually. Here's how:

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

Within this file, paste the following script

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

```
  status)
    ps aux | grep nagios
    ;;
  *)
    echo "Usage: $0 {start|stop|restart|status}"
    exit 1
    ;;
esac
exit 0
```



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

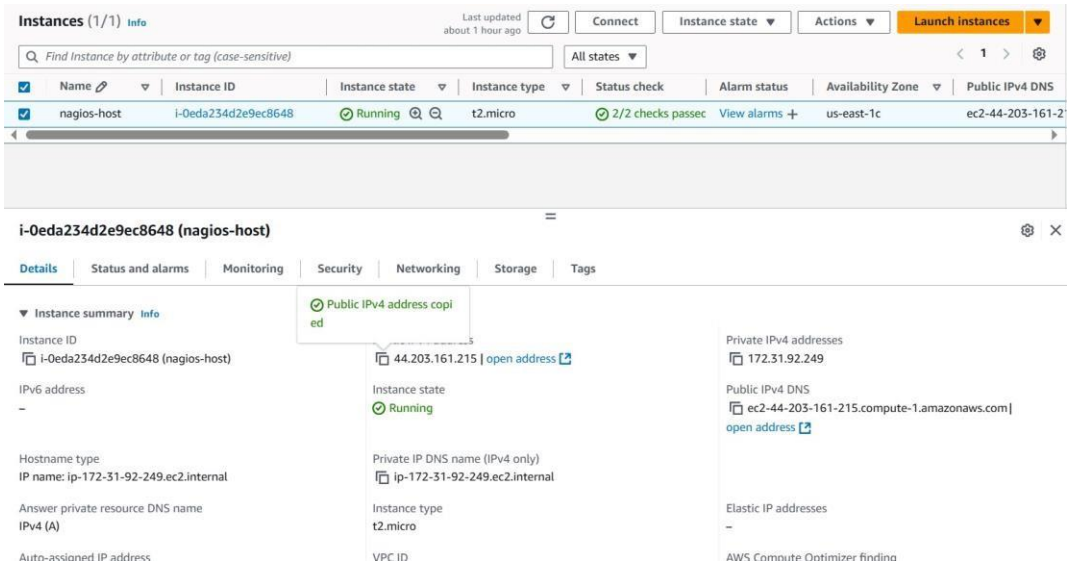
Run **sudo chkconfig --add nagios** again And then run **sudo chkconfig nagios on**

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

sudo service nagios start

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

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

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

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



Browse for this url: http://<your_public_ip_address>/nagios

The browser may ask you for your nagios credentials which set in the earlier steps

The username is nagiosadmin and enter the password that you set earlier

Name: Abhinav Swaminathan Div: D15C Roll No: 01

Conclusion:

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

Name: Abhinav Swaminathan Div: D15C Roll No: 01

Adv DevOps Exp 10

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

Monitoring Using Nagios:

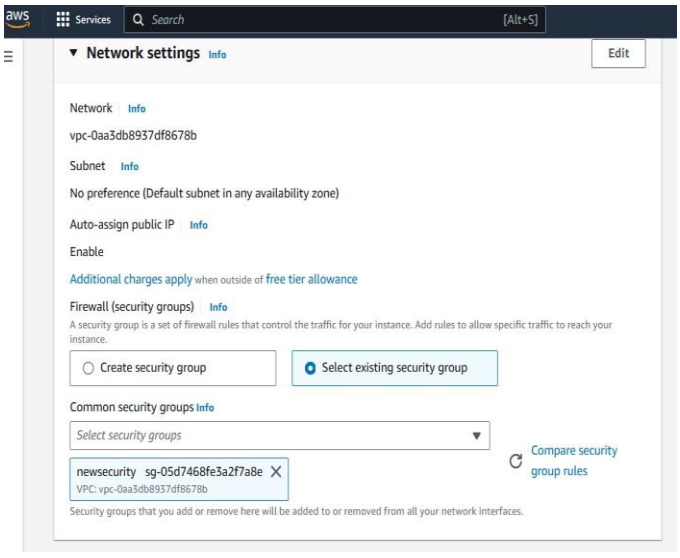
Step 1: To Confirm Nagios is running on the server side Perform the following command on your Amazon Linux Machine (Nagios-host).

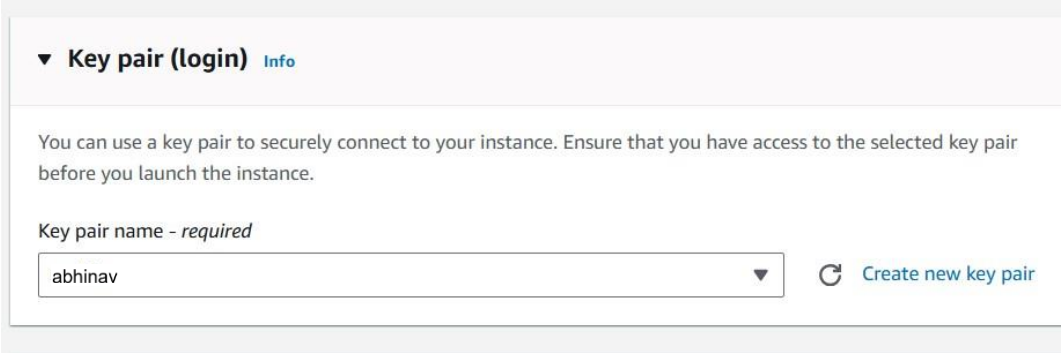
Run this command **sudo systemctl status**

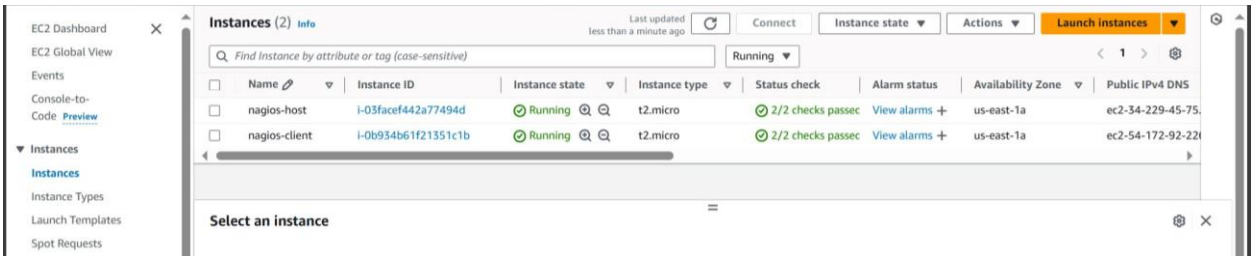
```
ec2-user@ip-172-31-41-160:~/downloads/nagios-plugins-2.4.11
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$ sudo systemctl status
● ip-172-31-41-160.ec2.internal
    State: running
    Units: 296 loaded (incl. loaded aliases)
     Jobs: 0 queued
   Failed: 0 units
    Since: Wed 2024-10-02 12:28:05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

Step 2: Before we begin,

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







Step 3: TO BE DONE IN THE Nagios-host TERMINAL

In the nagios-host terminal, run this command

ps -ef | grep nagios

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

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

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

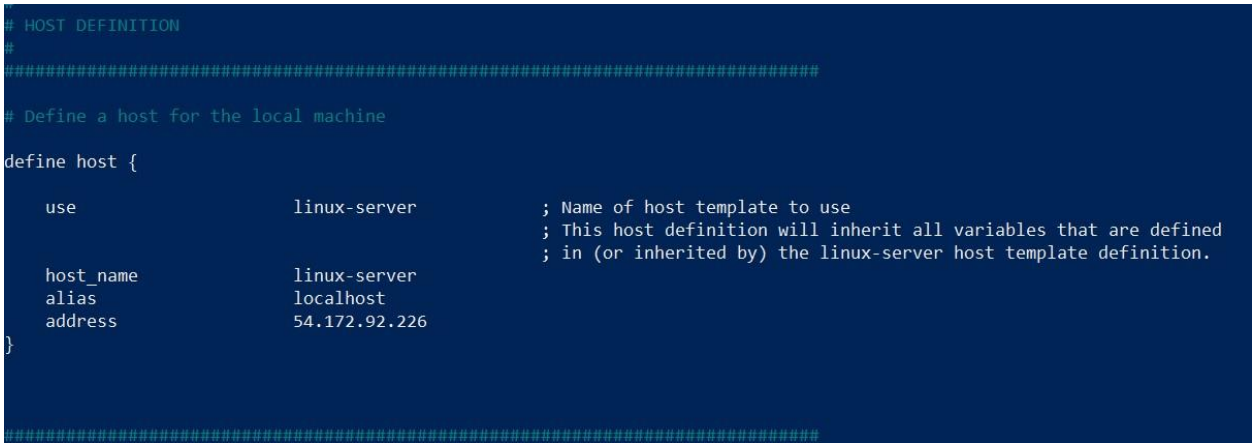
Copy the sample localhost.cfg file to linuxhost folder. Use the following mentioned command to achieve it **cp /usr/local/nagios/etc/objects/localhost.cfg /usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg**

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

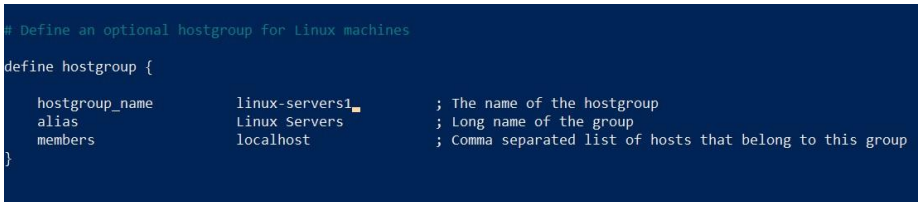
nano /usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg

Changes to be made:

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



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

# Define a host for the local machine

define host {

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



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



```
# Define an optional hostgroup for Linux machines

define hostgroup {

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

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

Open the Nagios Config file and add the following line

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

Add the following line in the file and save

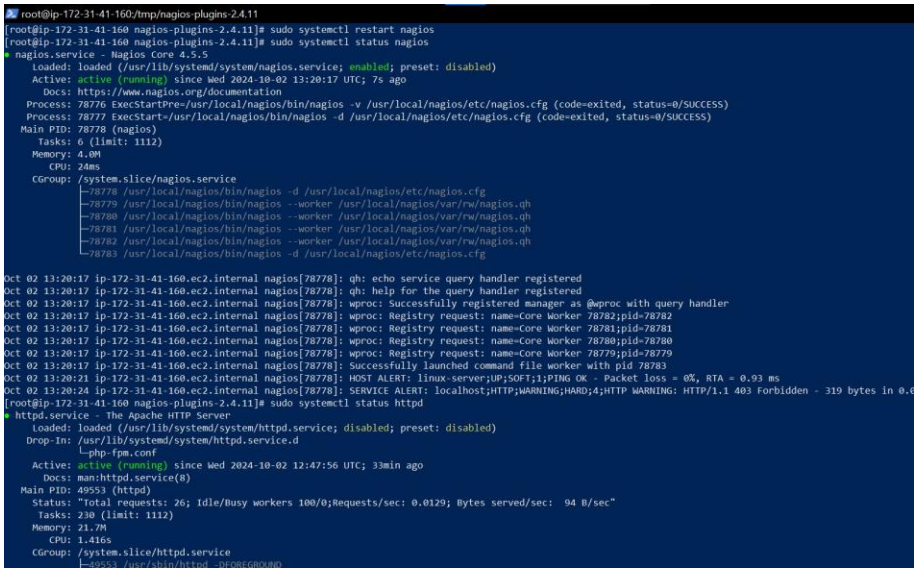
cfg_dir=/usr/local/nagios/etc/objects/monitorhosts/

```
# OBJECT CONFIGURATION FILE(S)
# These are the object configuration files in which you define hosts,
# host groups, contacts, contact groups, services, etc.
# You can split your object definitions across several config files
# if you wish (as shown below), or keep them all in a single config file.

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

# Definitions for monitoring the local (Linux) host
cfg_file=/usr/local/nagios/etc/objects/localhost.cfg
cfg_dir=/usr/local/nagios/etc/objects/monitorhosts/
# Definitions for monitoring a Windows machine
#cfg_file=/usr/local/nagios/etc/objects/windows.cfg
```

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

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

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

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

Running pre-flight check on configuration data...

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

Total Warnings: 0
Total Errors:   0

Things look okay - No serious problems were detected during the pre-flight check
[root@ip-172-31-41-160 nagios-plugins-2.4.11]#
```

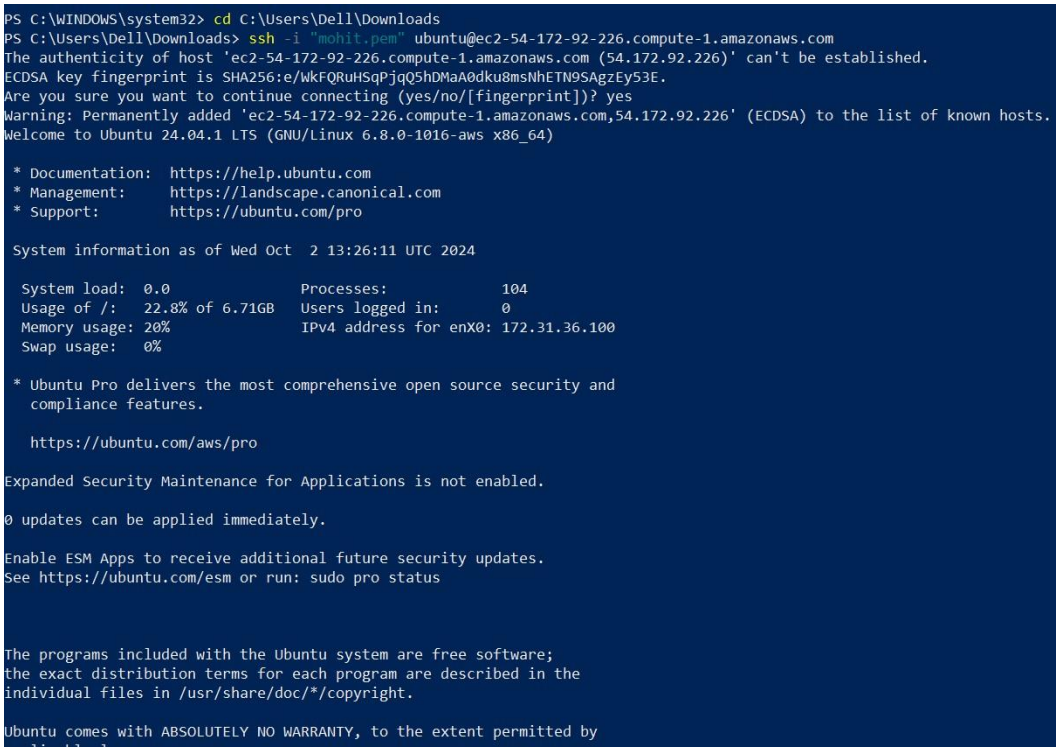
You are good to go if there are no errors.
Restart the nagios service **service**
nagios restart
And by running sudo systemctl status nagios, we can again check whether our server is running or not



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

Now it is time to switch to the client machine.

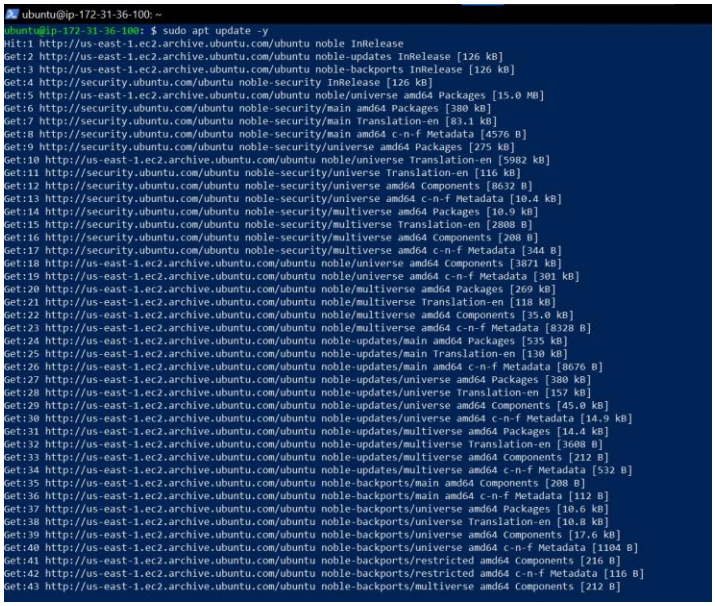
SSH into the machine or simply use the EC2 Instance Connect feature.



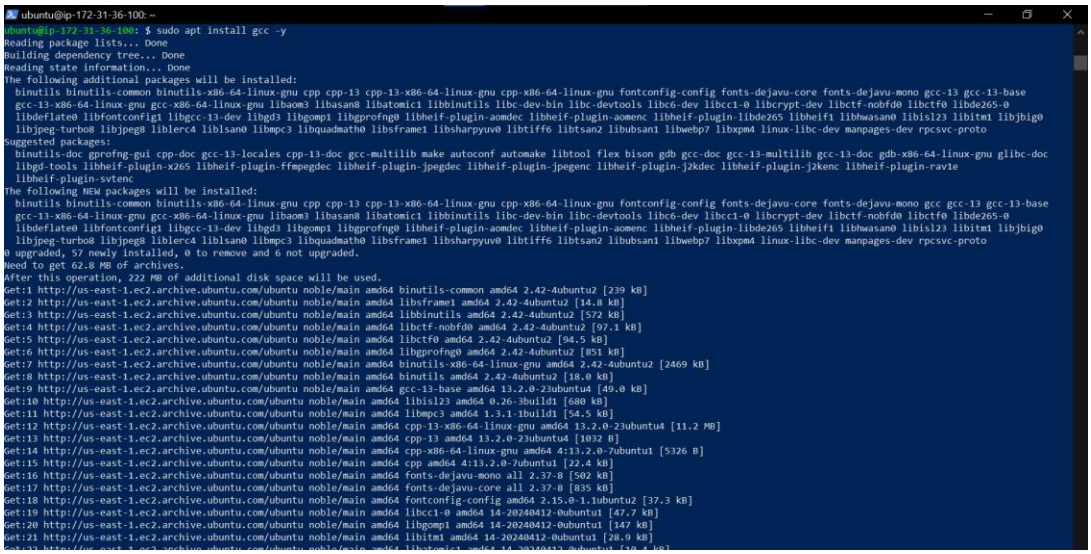
Make a package index update and install gcc, nagios-nrpe-server and the plugins. Run the following commands to achieve the same.

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

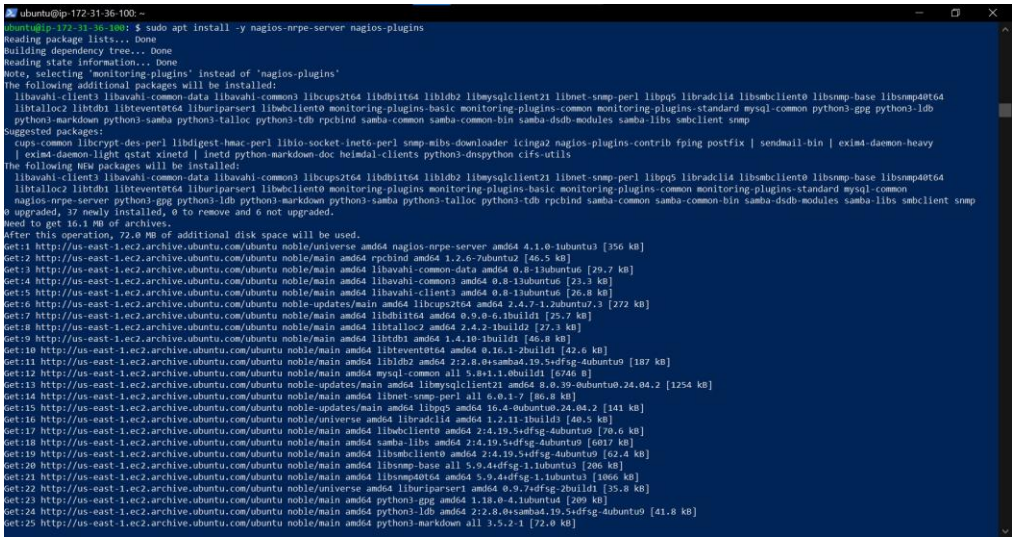
Name: Abhinav Swaminathan Div: D15C Roll No: 01



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



```
ubuntu@ip-172-31-36-100: ~
ubuntu@ip-172-31-36-100: $ sudo apt install gcc -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu cpp cpp-13 cpp-13-x86-64-linux-gnu cpp-x86-64-linux-gnu fontconfig-config fonts-dejavu-core fonts-dejavu-mono gcc-13 gcc-13-base
  gcc-13-x86-64-linux-gnu gcc-x86-64-linux-gnu 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.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 binutils-common amd64 2.42-4ubuntu2 [239 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libsframe1 amd64 2.42-4ubuntu2 [14.8 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libbinutils amd64 2.42-4ubuntu2 [572 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libctf-nobfd0 amd64 2.42-4ubuntu2 [97.1 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libctf0 amd64 2.42-4ubuntu2 [94.5 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libgprofng0 amd64 2.42-4ubuntu2 [851 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 binutils-x86-64-linux-gnu amd64 2.42-4ubuntu2 [2469 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 binutils amd64 2.42-4ubuntu2 [18.0 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 gcc-13-base amd64 13.2.0-23ubuntu4 [49.0 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libisl23 amd64 0.26-3build1 [680 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libmpc3 amd64 1.3.1-1build1 [54.5 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 cpp-13-x86-64-linux-gnu amd64 13.2.0-23ubuntu4 [11.2 MB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 cpp-13 amd64 13.2.0-23ubuntu4 [1032 B]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 cpp-x86-64-linux-gnu amd64 4:13.2.0-7ubuntu1 [5326 B]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 cpp amd64 4:13.2.0-7ubuntu1 [22.4 kB]
Get:16 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 fonts-dejavu-mono all 2.37-8 [502 kB]
Get:17 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 fonts-dejavu-core all 2.37-8 [835 kB]
Get:18 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 fontconfig-config amd64 2.15.0-1.1ubuntu2 [37.3 kB]
Get:19 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libcc1-0 amd64 14-20240412-0ubuntu1 [47.7 kB]
Get:20 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libgomp1 amd64 14-20240412-0ubuntu1 [147 kB]
Get:21 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libitm1 amd64 14-20240412-0ubuntu1 [28.9 kB]
```



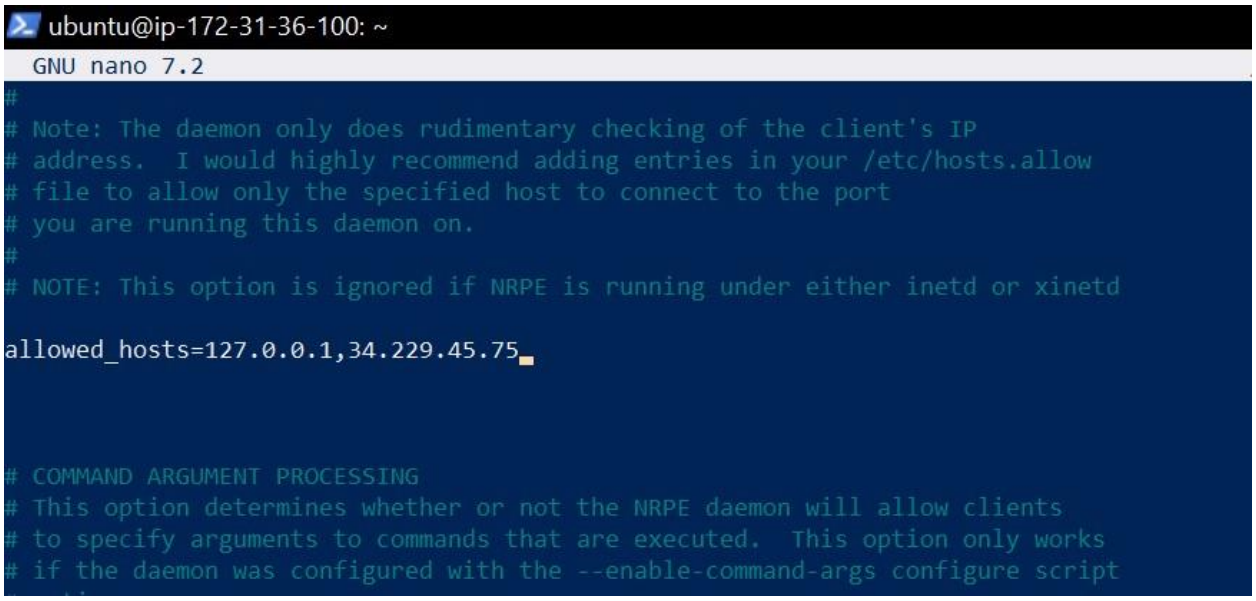
```
ubuntu@ip-172-31-36-100: ~
ubuntu@ip-172-31-36-100: $ sudo apt install -y nagios-nrpe-server nagios-plugins
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'monitoring-plugins' instead of 'nagios-plugins'
The following additional packages will be installed:
  libavahi-client3 libavahi-common-data libavahi-common3 libcups2t64 libdbi1t64 libldb2 libmysqlclient21 libnet-snmp-perl libpq5 libradcli4 libsmbclient0 libsnmp-base libsnmp40t64
  libtalloc2 libtdb1 libtevent0t64 liburiparser1 libwbclient0 monitoring-plugins-basic monitoring-plugins-common monitoring-plugins-standard mysql-common python3-gpg python3-ldb
  python3-markdown python3-samba python3-talloc python3-tdb rpcbind samba-common samba-common-bin samba-dsdb-modules samba-libs smbclient snmp
Suggested packages:
  cups-common libcrypt-des-perl libdigest-hmac-perl libio-socket-inet6-perl snmp-mibs-downloader icinga2 nagios-plugins-contrib fping postfix | sendmail-bin | exim4-daemon-heavy
  | exim4-daemon-light qstat xinetd | inetd python-markdown-doc heimdal-clients 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]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libavahi-common3 amd64 0.8-13ubuntu6 [23.3 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libavahi-client3 amd64 0.8-13ubuntu6 [26.8 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 libcups2t64 amd64 2.4.7-1.2ubuntu7.3 [272 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libdbi1t64 amd64 0.9.0-6.1build1 [25.7 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libtalloc2 amd64 2.4.2-1build2 [27.3 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libtdb1 amd64 1.4.10-1build1 [46.8 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libtevent0t64 amd64 0.16.1-2build1 [42.6 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libldb2 amd64 2:2.8.0+samba4.19.5+dfsg-4ubuntu9 [187 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 mysql-common all 5.8+1.1.0build1 [6746 B]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 libmysqlclient21 amd64 8.0.39-0ubuntu0.24.04.2 [1254 kB]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libnet-snmp-perl all 6.0.1-7 [86.8 kB]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 libpq5 amd64 16.4-0ubuntu0.24.04.2 [141 kB]
Get:16 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 libradcli4 amd64 1.2.11-1build3 [40.5 kB]
Get:17 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libwbclient0 amd64 2:4.19.5+dfsg-4ubuntu9 [70.6 kB]
Get:18 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 samba-libs amd64 2:4.19.5+dfsg-4ubuntu9 [6017 kB]
Get:19 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libsmbclient0 amd64 2:4.19.5+dfsg-4ubuntu9 [62.4 kB]
Get:20 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libsnmp-base all 5.9.4+dfsg-1.1ubuntu3 [206 kB]
Get:21 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libsnmp40t64 amd64 5.9.4+dfsg-1.1ubuntu3 [1066 kB]
Get:22 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 liburiparser1 amd64 0.9.7+dfsg-2build1 [35.8 kB]
Get:23 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 python3-gpg amd64 1.18.0-4.1ubuntu4 [209 kB]
Get:24 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 python3-ldb amd64 2:2.8.0+samba4.19.5+dfsg-4ubuntu9 [41.8 kB]
Get:25 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 python3-markdown all 3.5.2-1 [72.0 kB]
```

Name: Abhinav Swaminathan Div: D15C Roll No: 01

Open nrpe.cfg file to make changes.

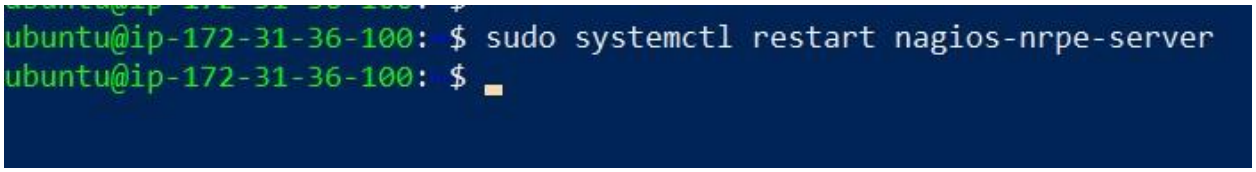
sudo nano /etc/nagios/nrpe.cfg

Under allowed_hosts, add your nagios host IP address like so



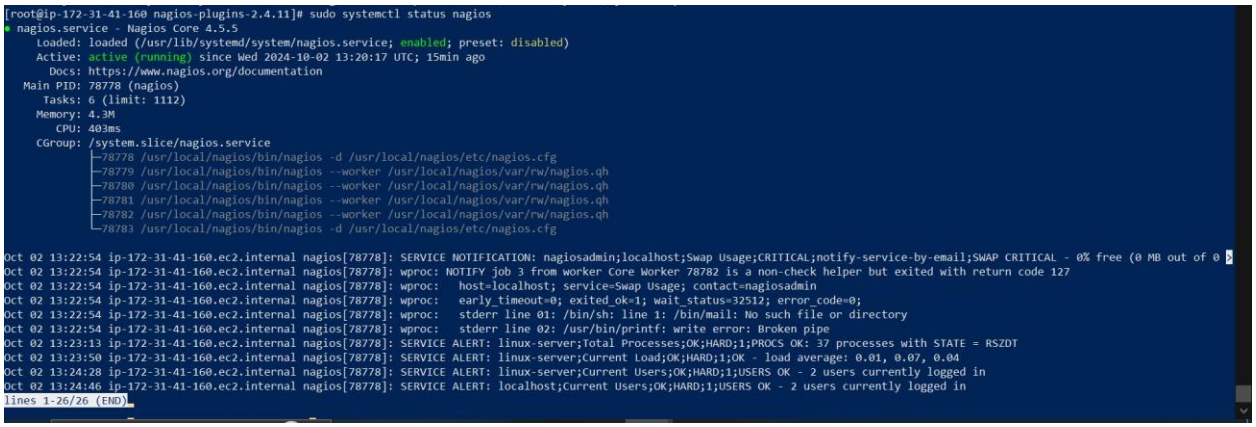
Now restart the NRPE server by this command.

sudo systemctl restart nagios-nrpe-server



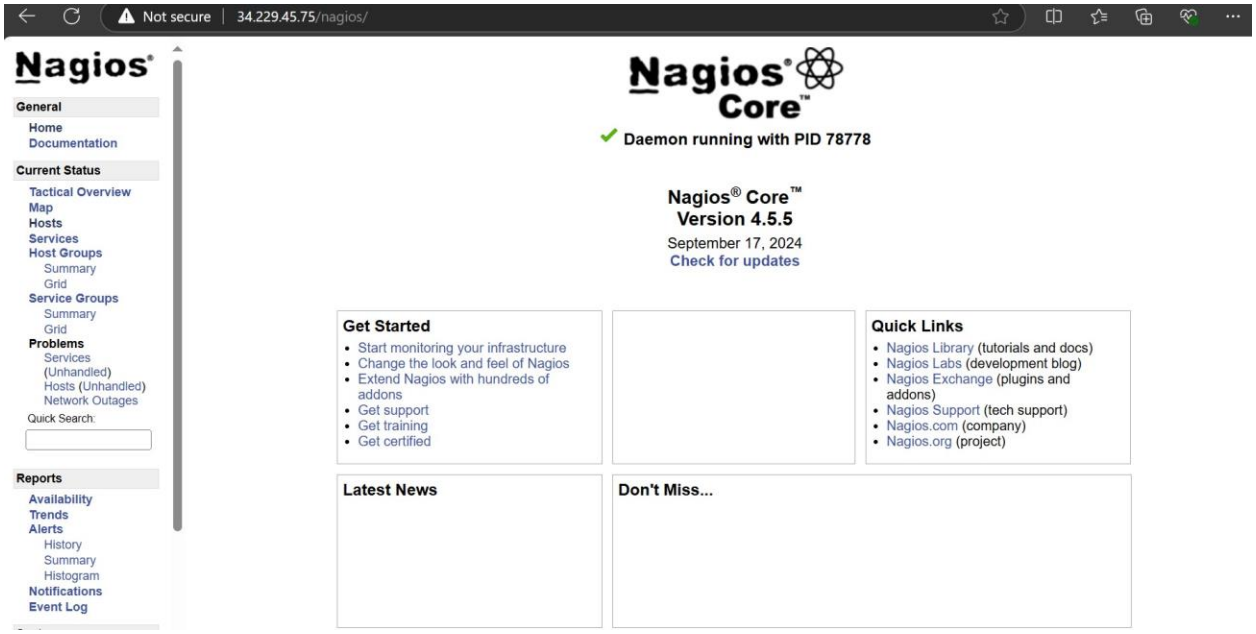
Run the following command in the Nagios-host terminal

sudo systemctl status nagios

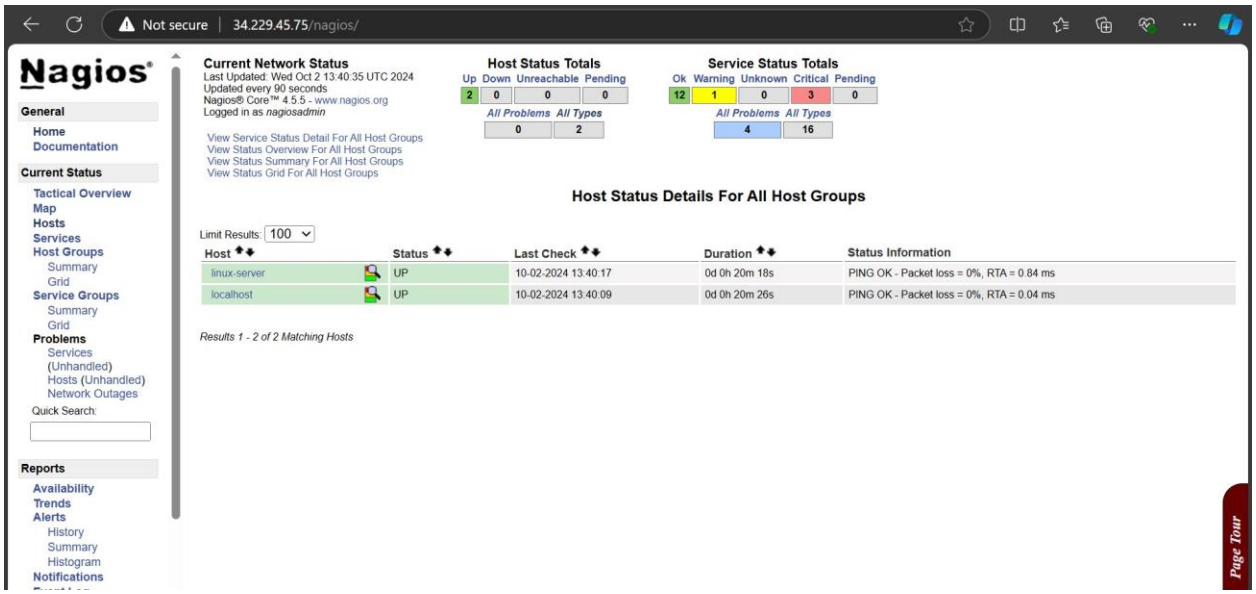


Step 5: Visiting your nagios server using your nagios-host ip address Open up your browser and look for http://<public_ip_address_of_nagios-host>/nagios

Name: Abhinav Swaminathan Div: D15C Roll No: 01

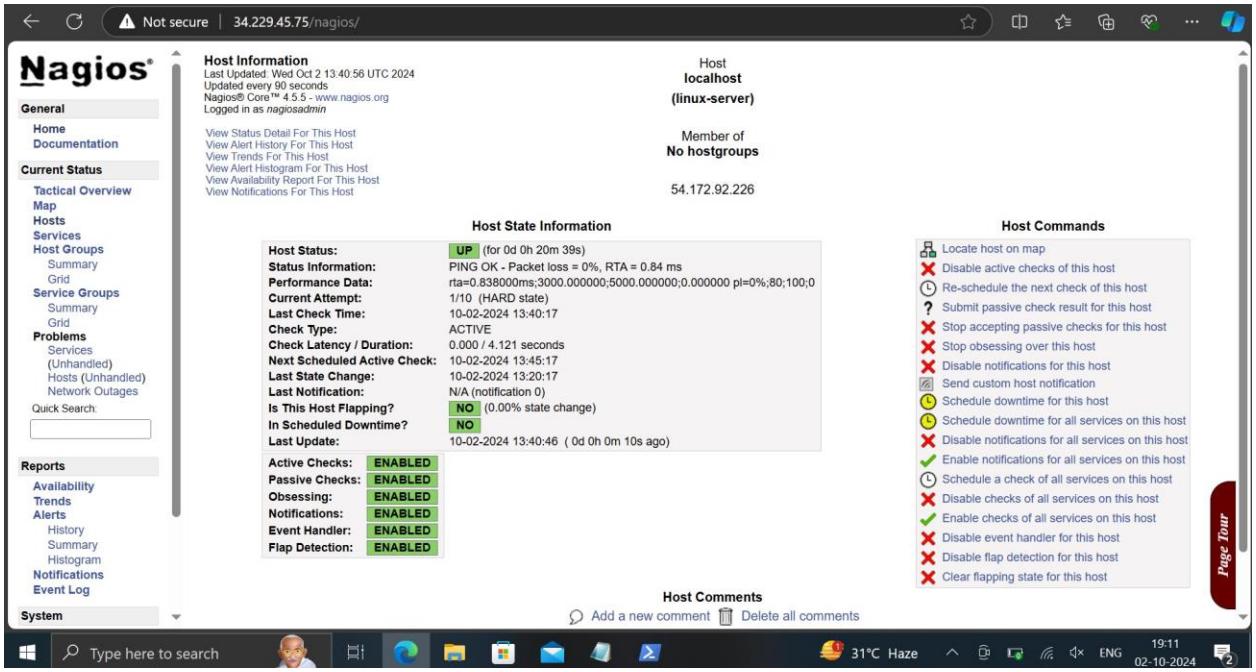


Click on Hosts.

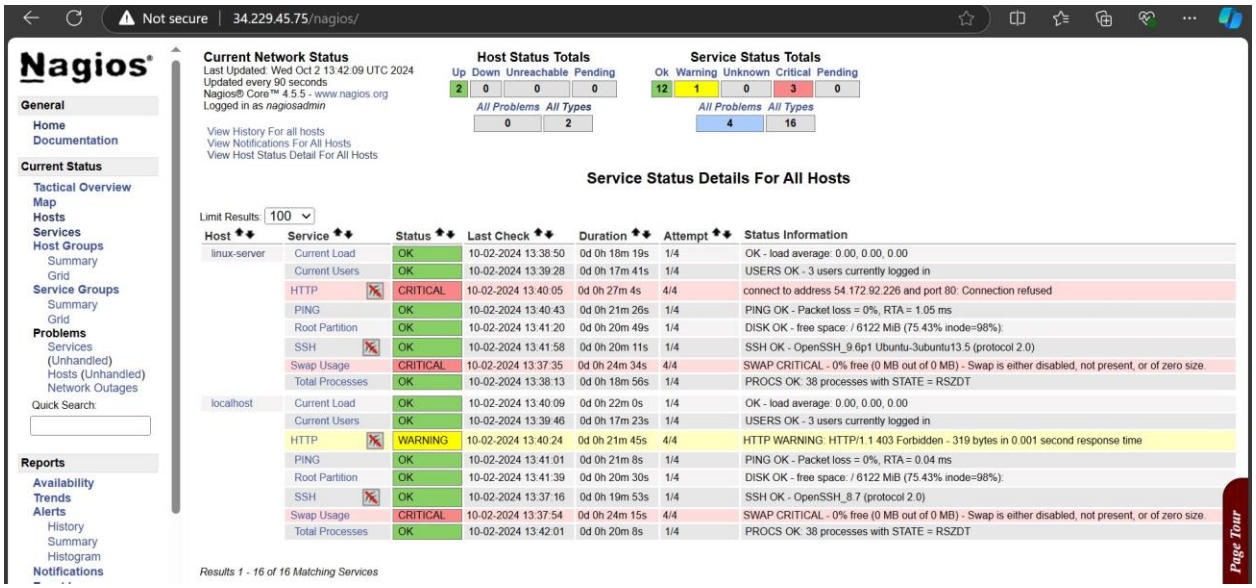


Click on linux-server to view host information

Name: Abhinav Swaminathan Div: D15C Roll No: 01



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



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

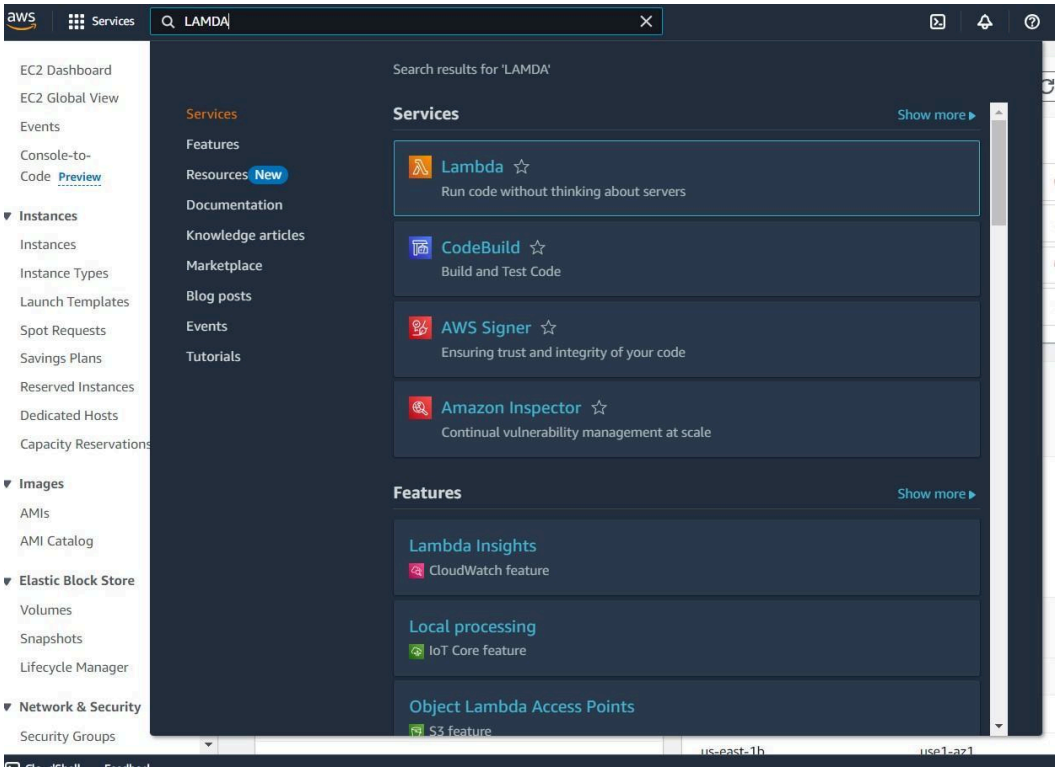
Name : Abhinav Swaminathan D15C Roll No : 01

Experiment 11

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

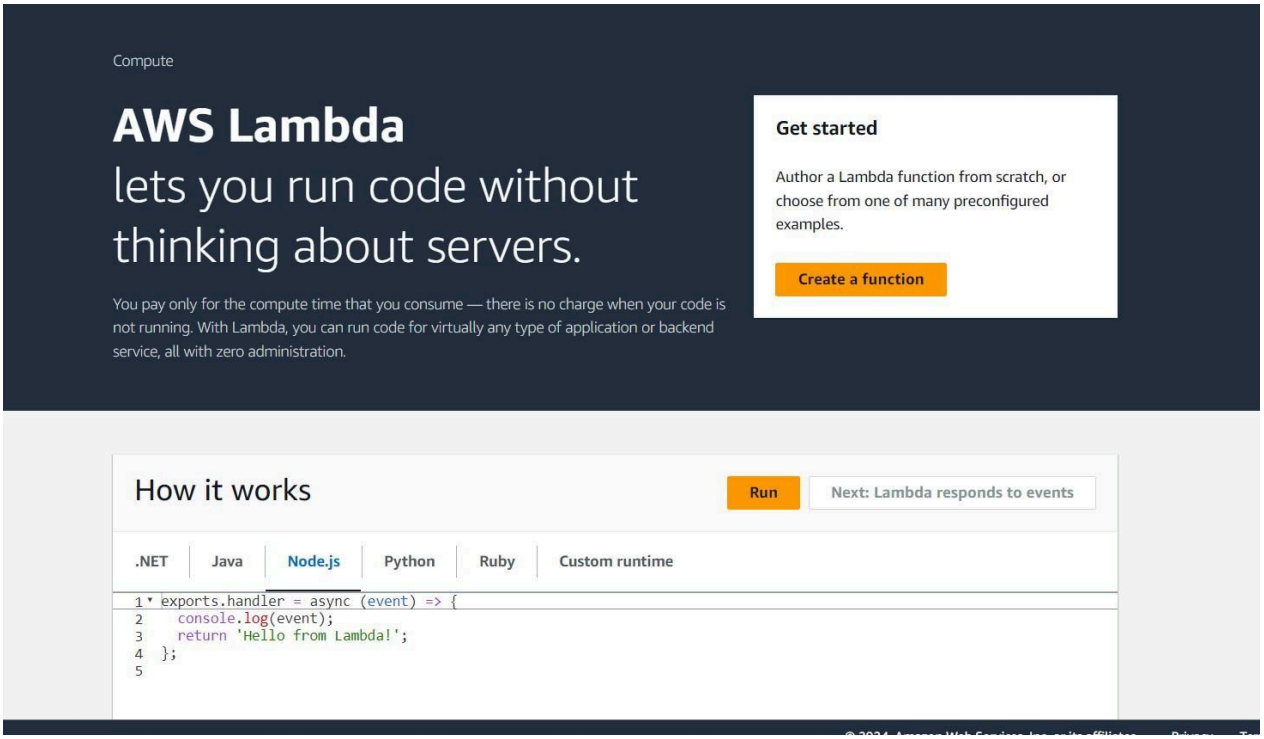
Step 1: Accessing AWS

Log in to your AWS Personal/Academy account. Navigate to the Lambda service by searching for "Lambda" in the AWS Management Console.



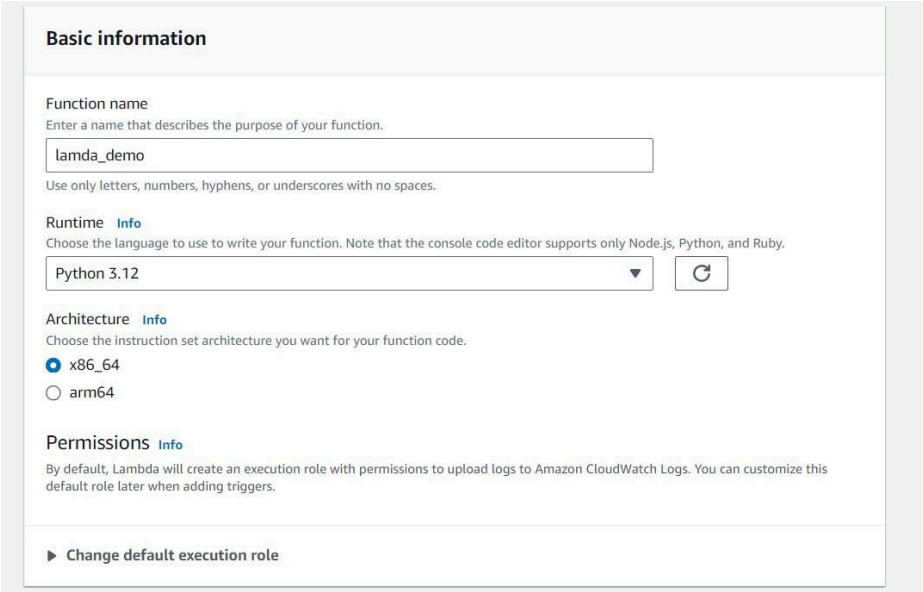
Step 2: Creating a New Lambda Function

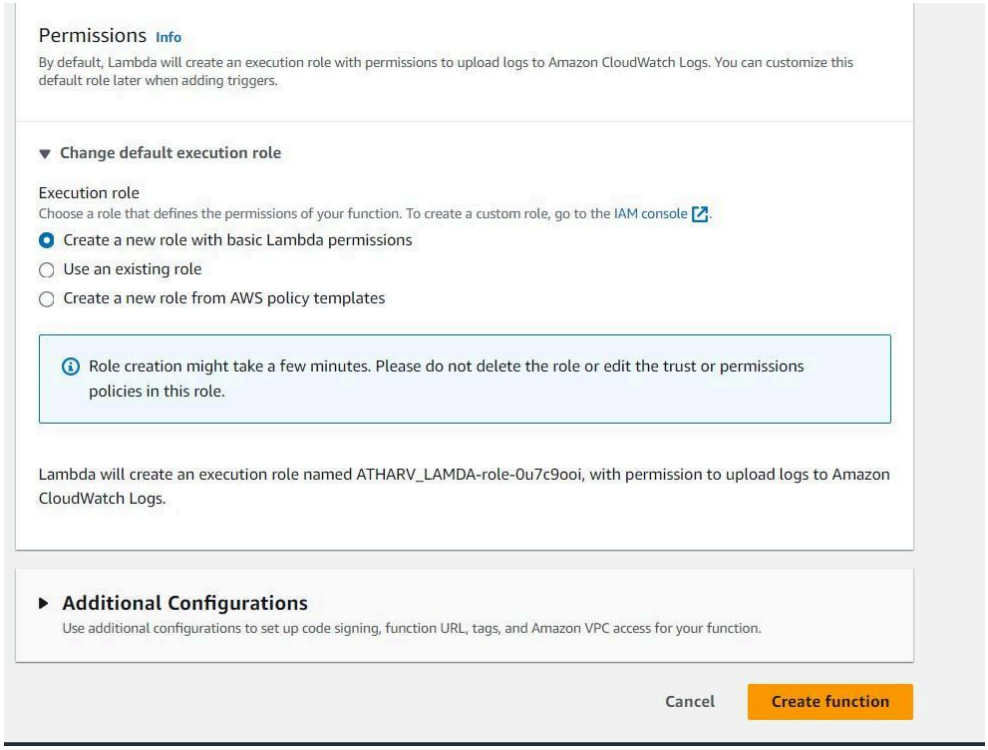
Click on the "Create function" button. Provide a name for your Lambda function and select the language you wish to use, such as Python 3.12. For architecture, choose x86, and for execution role, opt to create a new role with basic Lambda g permissions.

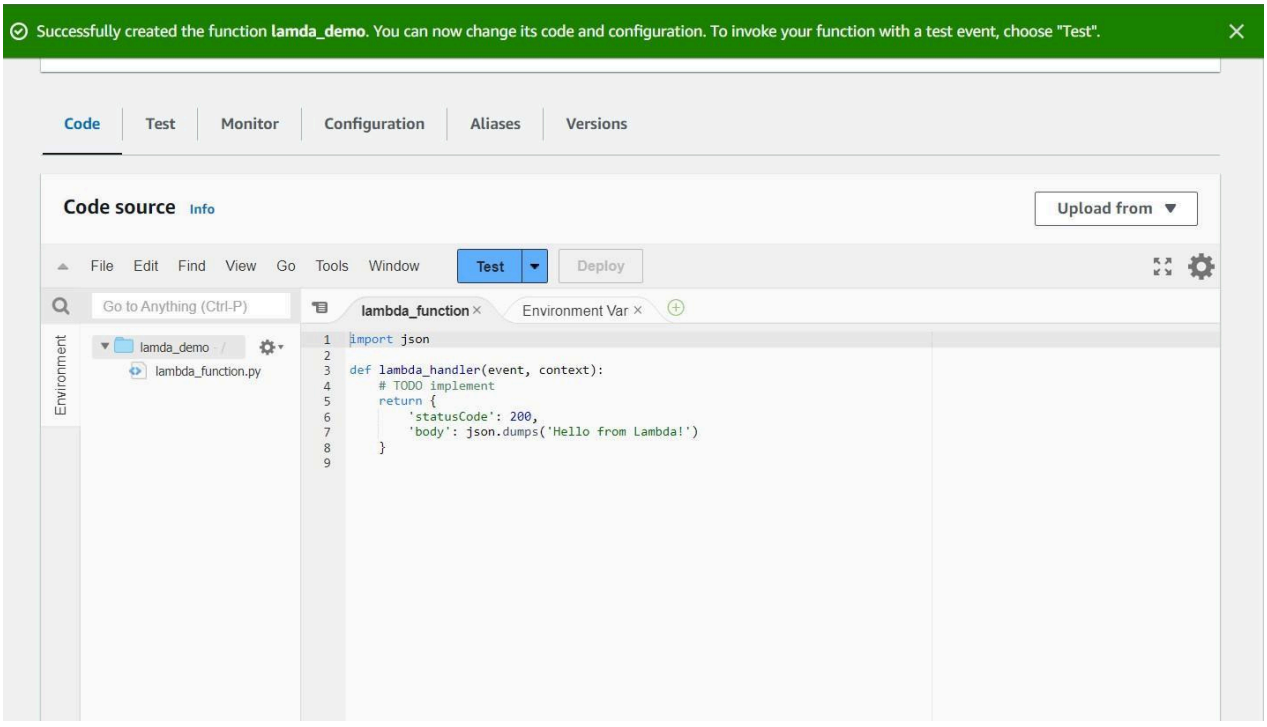


Step 3: Configuring Basic Settings

To modify the basic settings, navigate to the "Configuration" tab and click on "Edit" under General Settings. Here, you can add a description and adjust the memory and timeout settings. For this experiment, I set the timeout to 1 second, which is sufficient for testing.

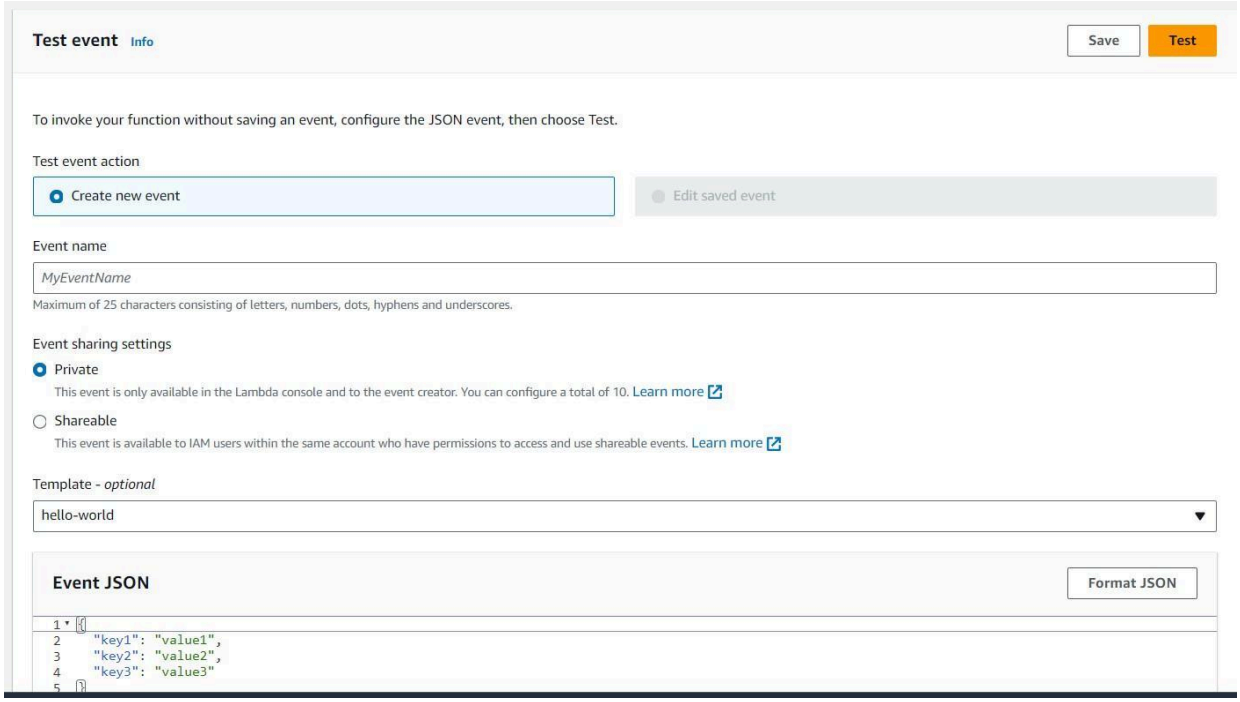


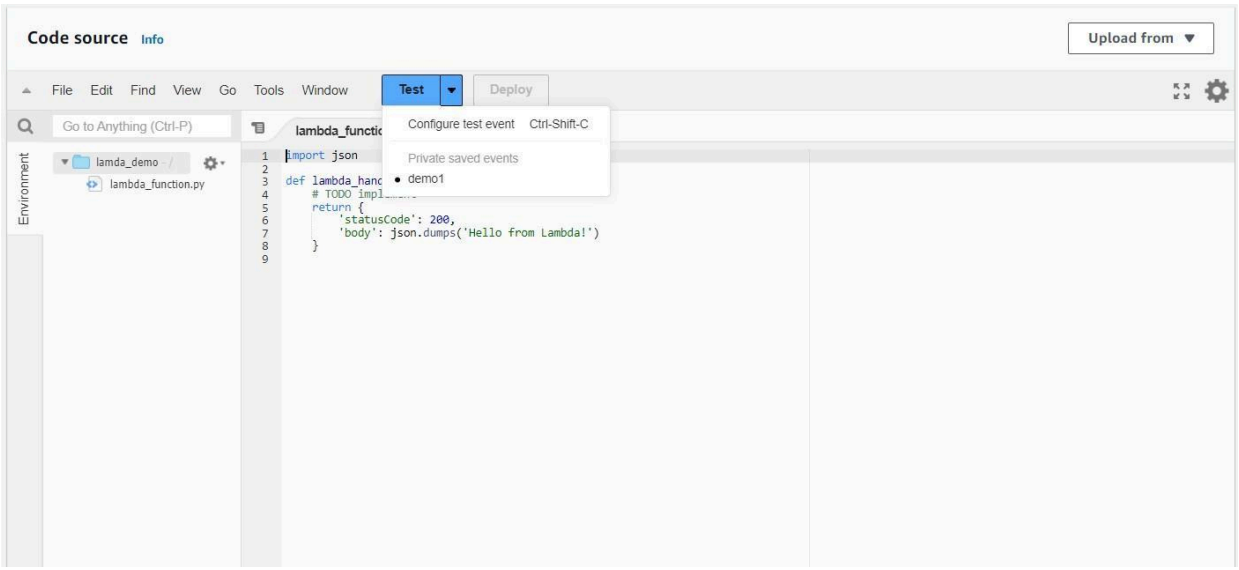

Permissions Info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.
Change default execution role
Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.
Create a new role with basic Lambda permissions
Use an existing role
Create a new role from AWS policy templates
Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.
Lambda will create an execution role named ATHARV_LAMDA-role-0u7c9ooi, with permission to upload logs to Amazon CloudWatch Logs.
Additional Configurations
Use additional configurations to set up code signing, function URL, tags, and Amazon VPC access for your function.
Cancel
Create function

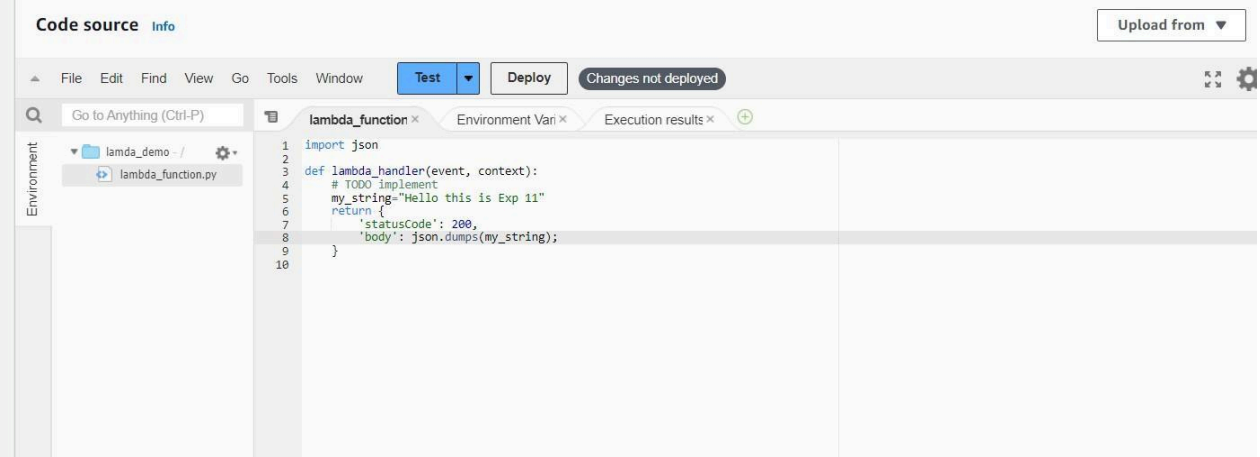

Successfully created the function lamda_demo. You can now change its code and configuration. To invoke your function with a test event, choose "Test".
Code
Test
Monitor
Configuration
Aliases
Versions
Code source Info
Upload from
File Edit Find View Go Tools Window
Test
Deploy
Go to Anything (Ctrl-P)
lambda_function
Environment Var
Environment
lamda_demo
lambda_function.py
import json
def lambda_handler(event, context):
# TODO implement
return {
'statusCode': 200,
'body': json.dumps('Hello from Lambda!')
}

Step 4: Testing the Function

Click on the "Test" tab and select "Create a new event." Name your event, set the event sharing to private, and choose the "hello-world" template.

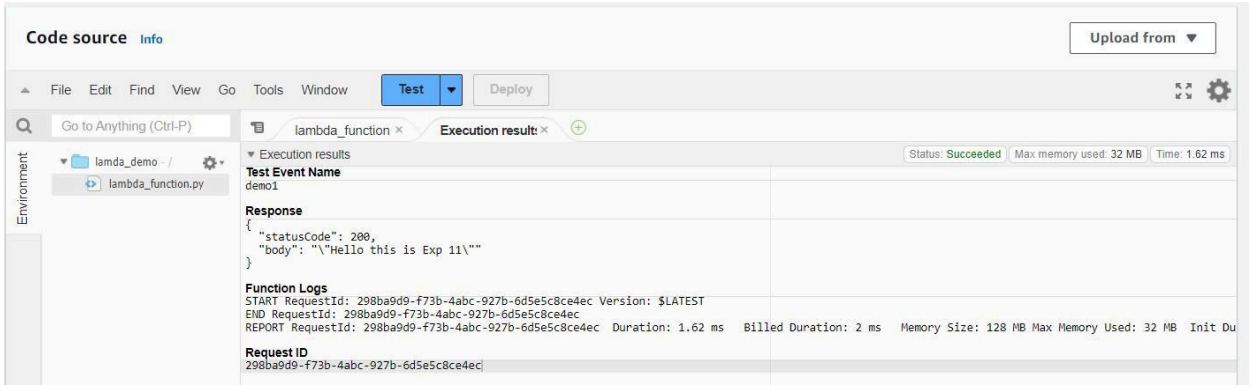






Step 5: Running the Test

In the Code section, select the newly created event from the dropdown menu and click on "Test." You should see the output displayed below.

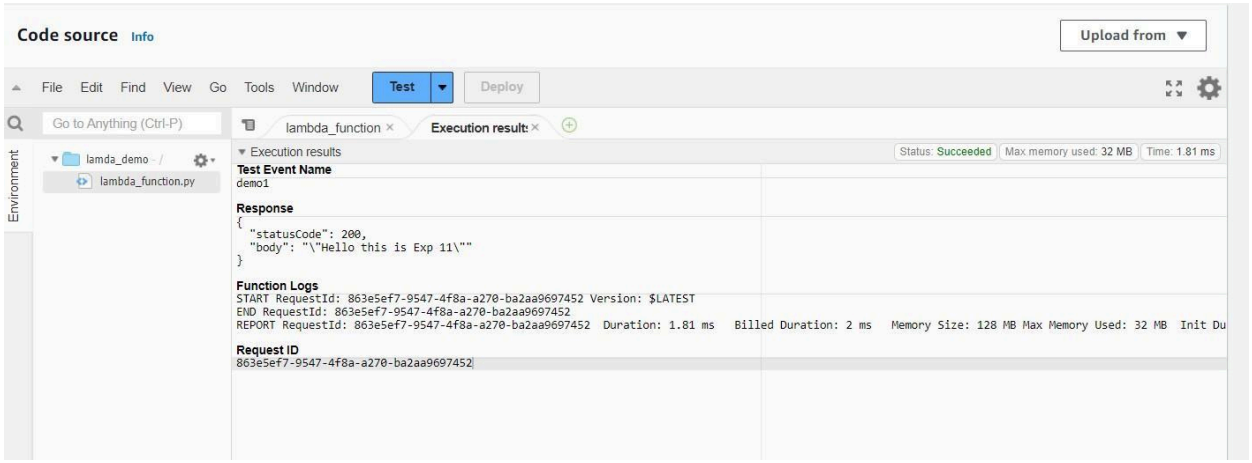


Step 6: Editing and Deploying the Code

You can modify your Lambda function's code as needed. I updated the code to display a new string. After making changes, press `Ctrl + S` to save and then click on "Deploy" to apply the updates.

Step 7: Final Testing

Return to the "Test" tab and execute the test again to observe the output. You should see a status code of 200 along with your string output and function logs confirming a successful deployment.



Conclusion:

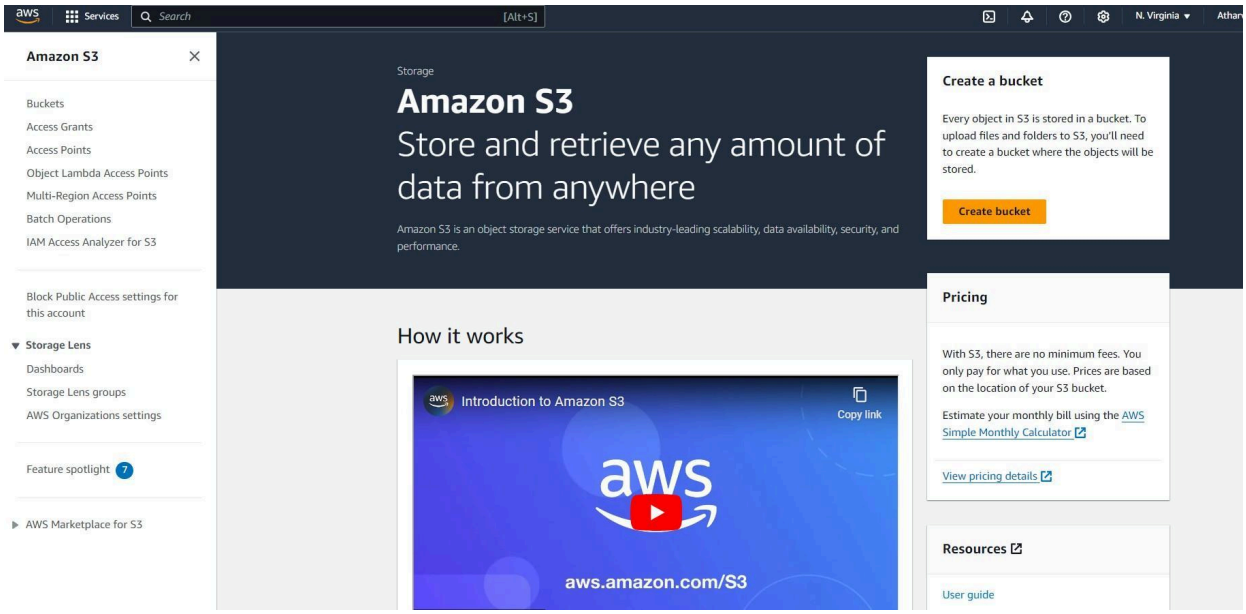
In this experiment, i created and tested your first AWS Lambda function using Python. I learned to navigate the AWS Management Console, configure basic settings, and modify function's code. This experience highlights the ease of deploying serverless applications with Lambda, allowing you to focus on coding rather than infrastructure management. I now have a foundational understanding to explore more complex serverless solutions and integrate AWS services for greater functionality.

Name: Abhinav Swaminathan **D15C** **Roll No: 01**

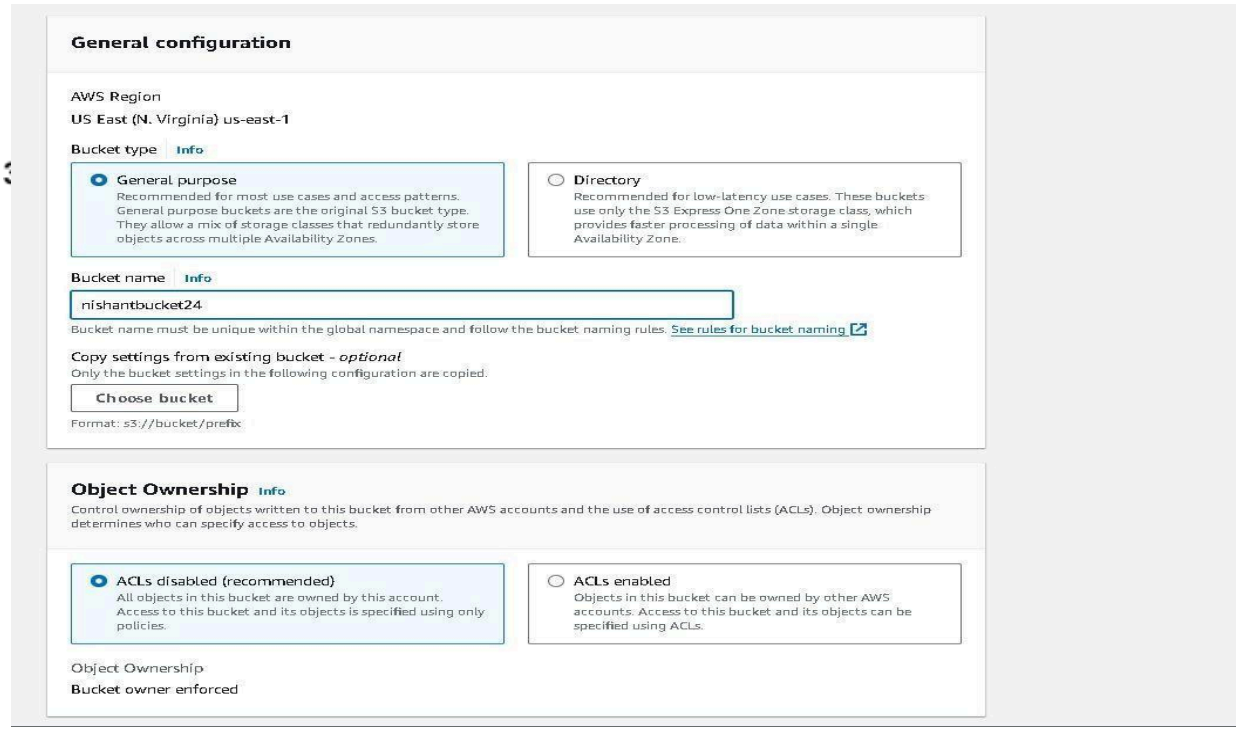
Experiment 12

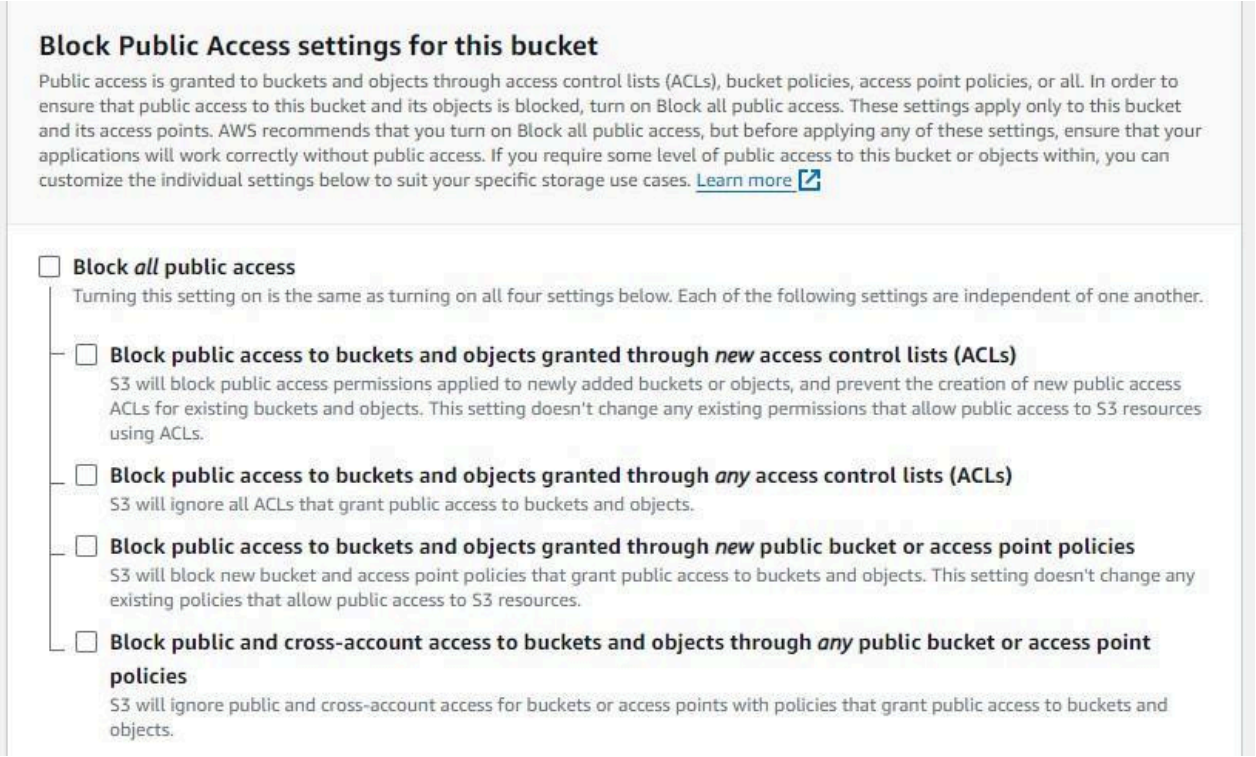
Aim: To create a Lambda function which will log “An Image has been added” once you add an object to a specific bucket in S3.

Step 1: Create a s3 bucket. 1) Search for S3 bucket in the services search. Then click on create bucket.

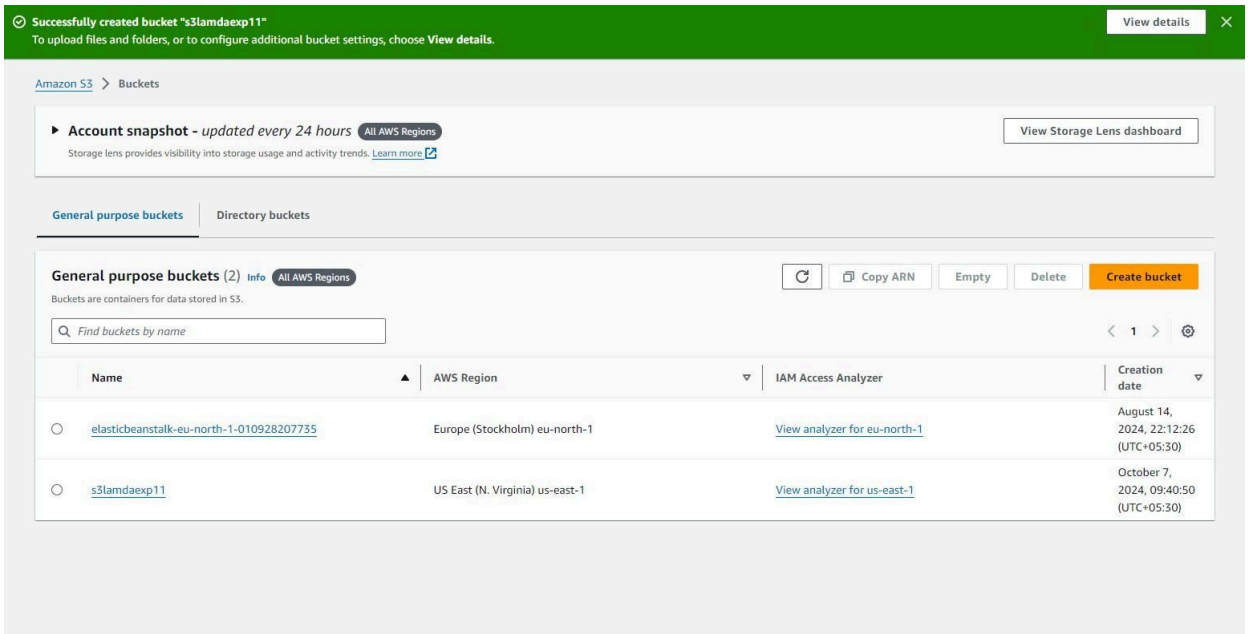


2) Keep the bucket as a general purpose bucket. Give a name to your bucket.

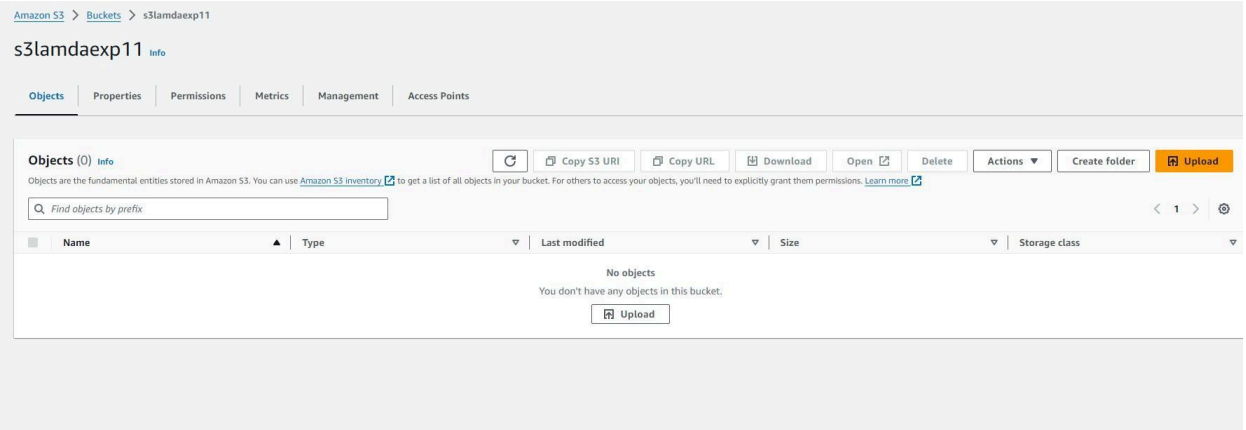


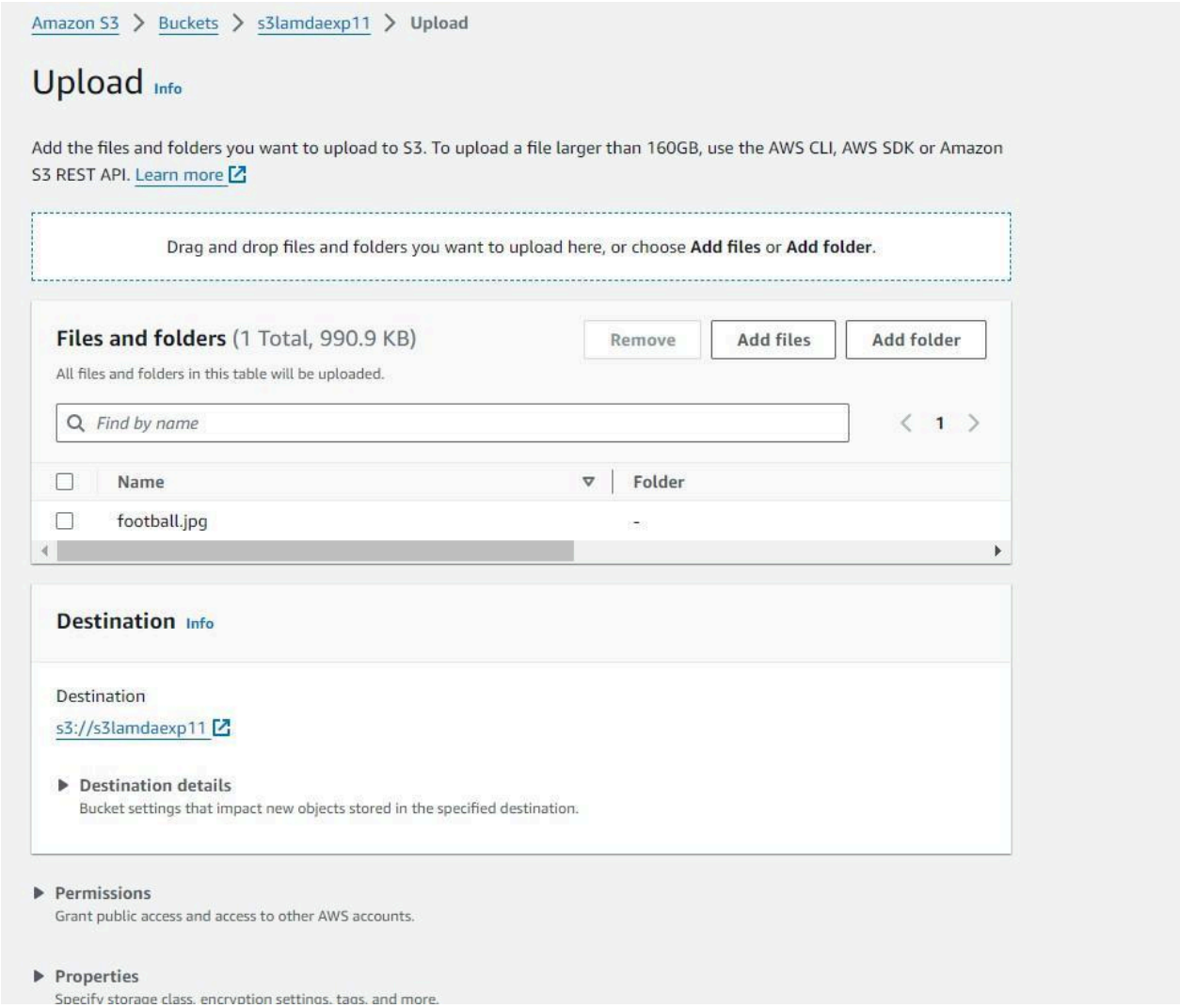


4) Keeping all other options the same, click on create. This would create your bucket. Now click on the name of the bucket.

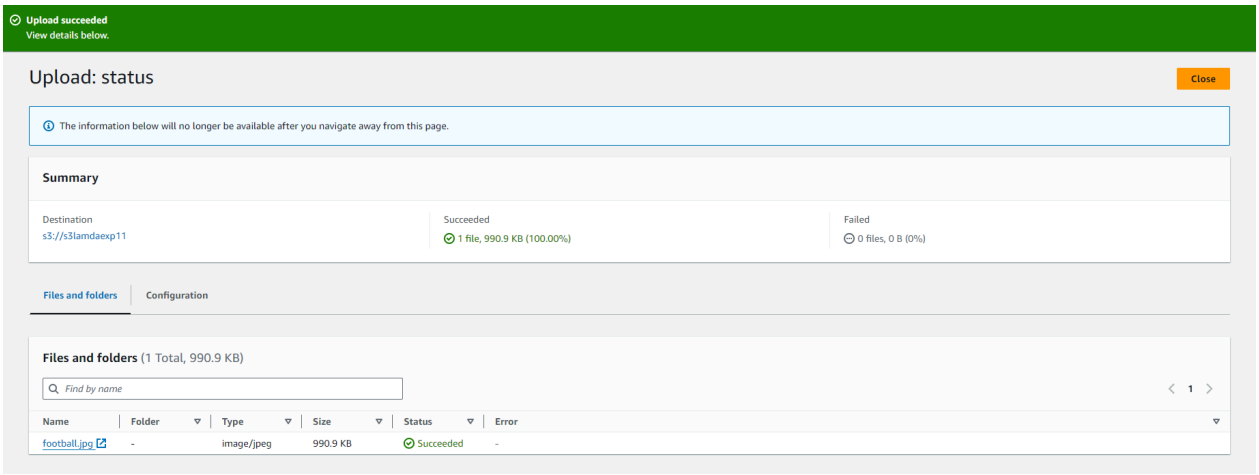


5) Here, click on upload, then add files. Select any image that you want to upload in the bucket and click on upload.



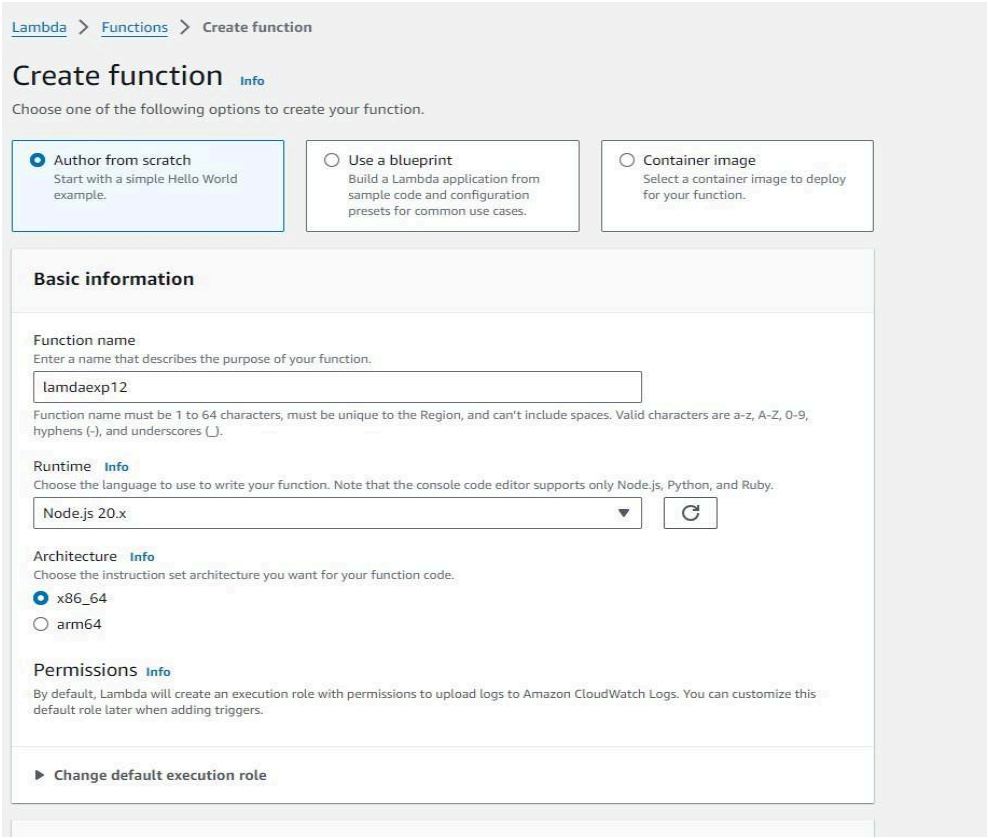


6) The image has been uploaded to the bucket.

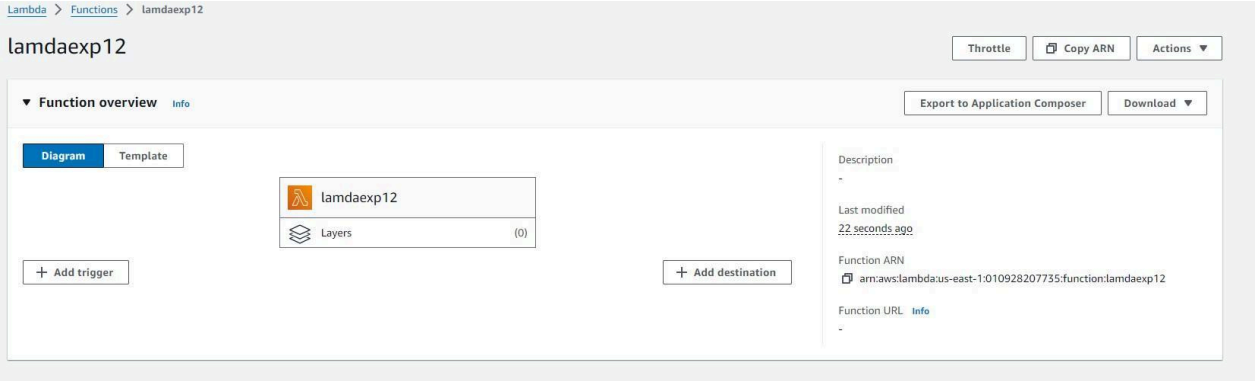


Step 2: Configure Lambda function

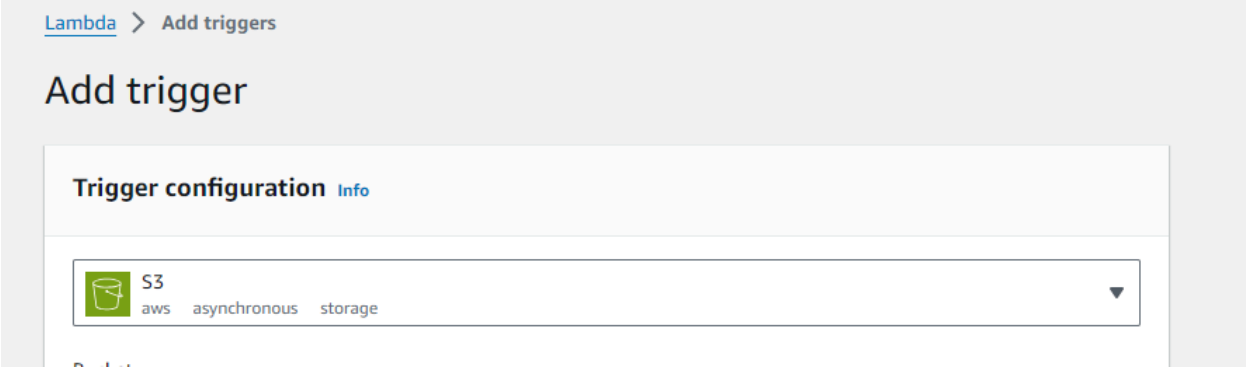
1) Go to the lambda function you had created berfor. (Services → Lambda → Click on name of function). Here, click on add trigger

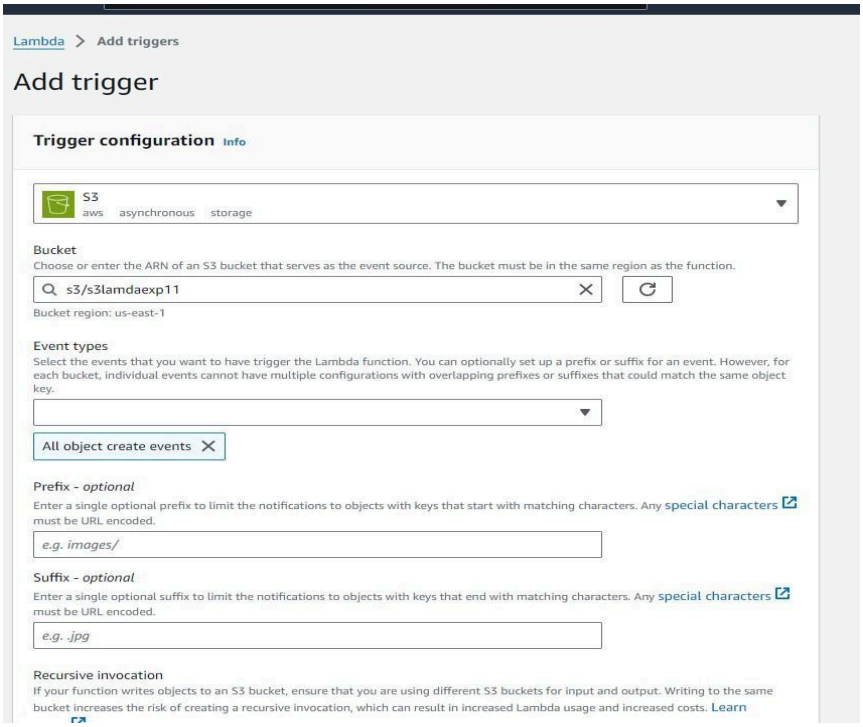


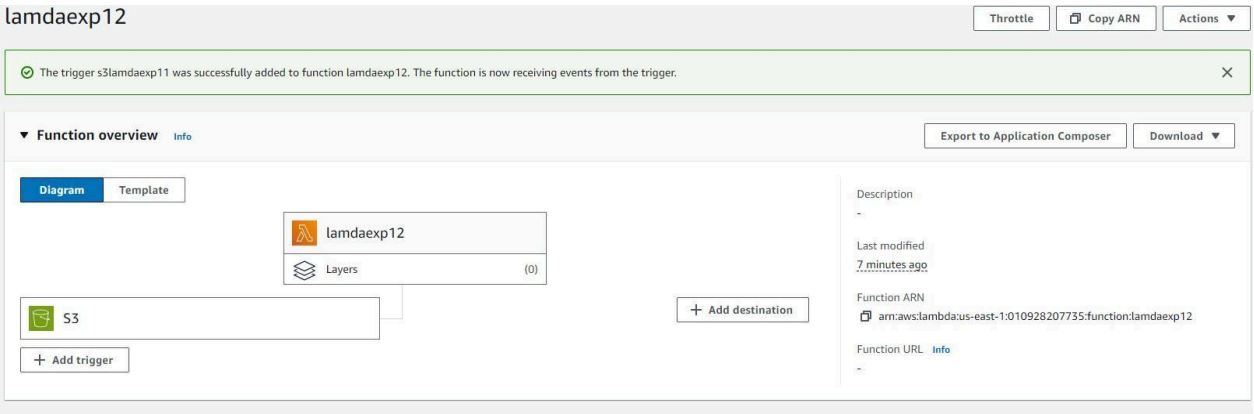
2) Under trigger configuration, search for S3 and select it.



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



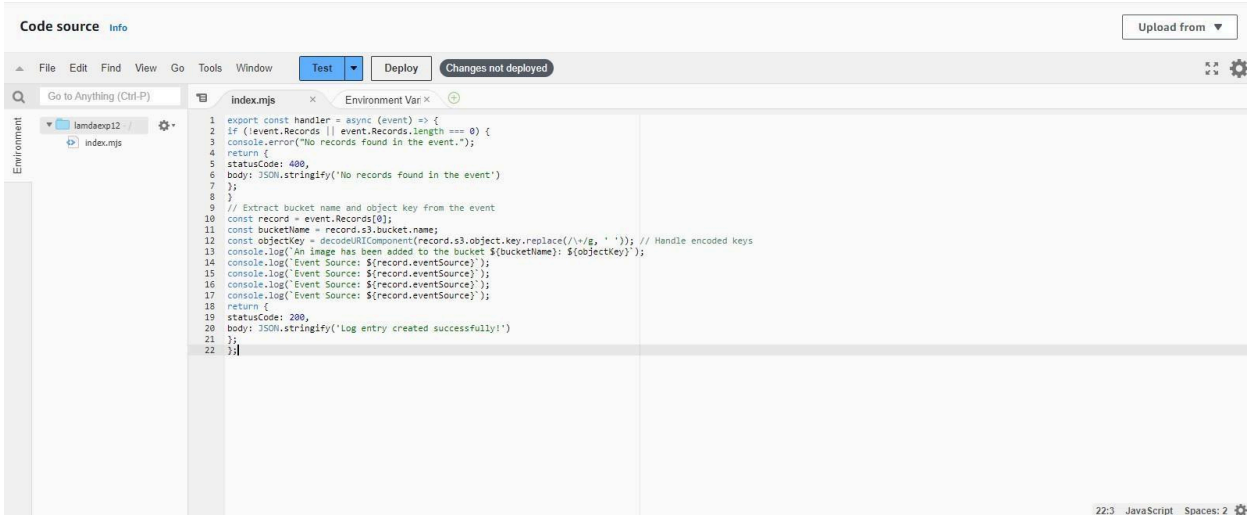


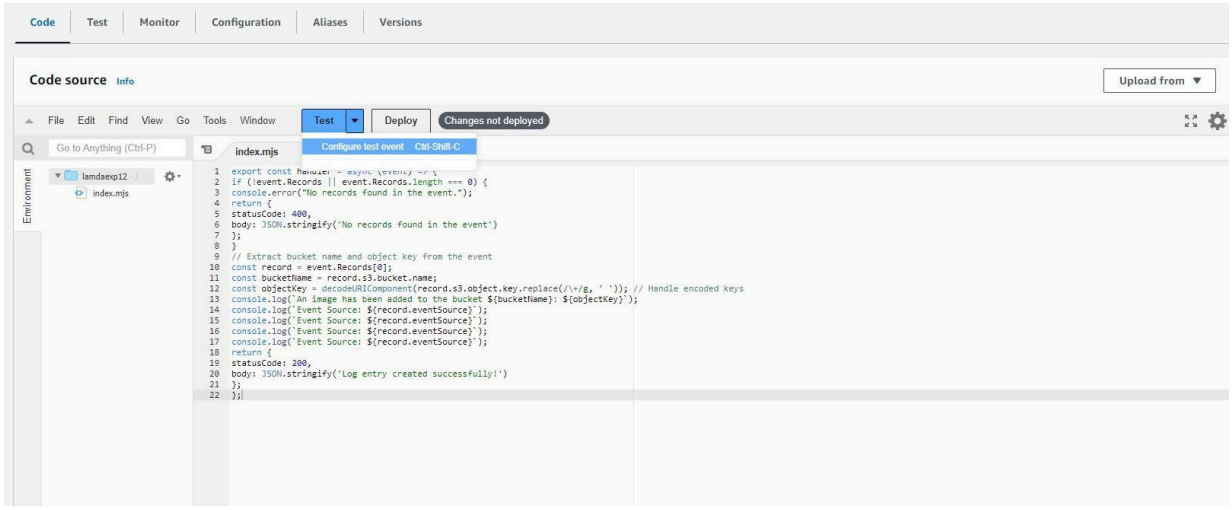


4) Scroll down to the code section of the function. Add the following javascript code to the code area by replacing the existing code

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

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

    console.log(`An image has been added to the bucket ${bucketName}: ${objectKey}`);
    console.log(`Event Source: ${record.eventSource}`);

    return {
        statusCode: 200,
        body: JSON.stringify('Log entry created successfully!')
    };
};
```

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





Private

This event is only available in the Lambda console and to the event creator. You can configure a total of 10. Learn more

Shareable

This event is available to IAM users within the same account who have permissions to access and use shareable events. Learn more

Template - *optional*

hello-world

Event JSON Format JSON

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

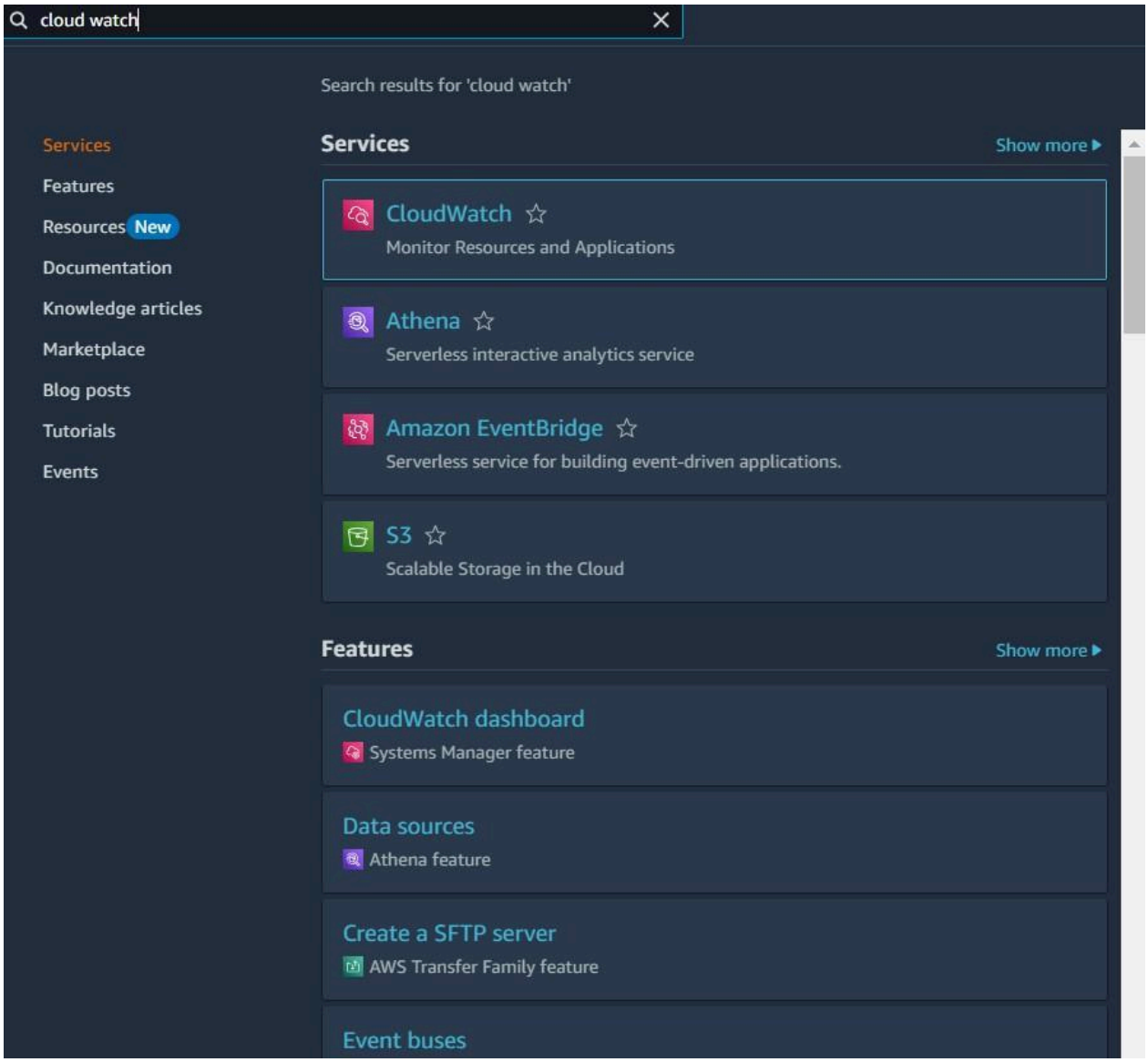
1:1 JSON Spaces: 2

Cancel Invoke Save

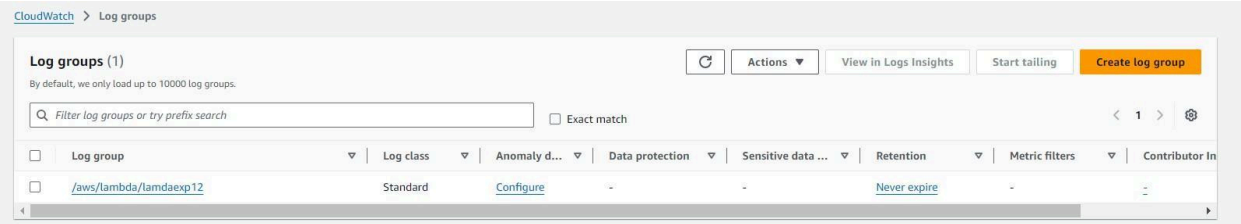
The test event **myevent1** was successfully saved.

Step 3: Check the logs

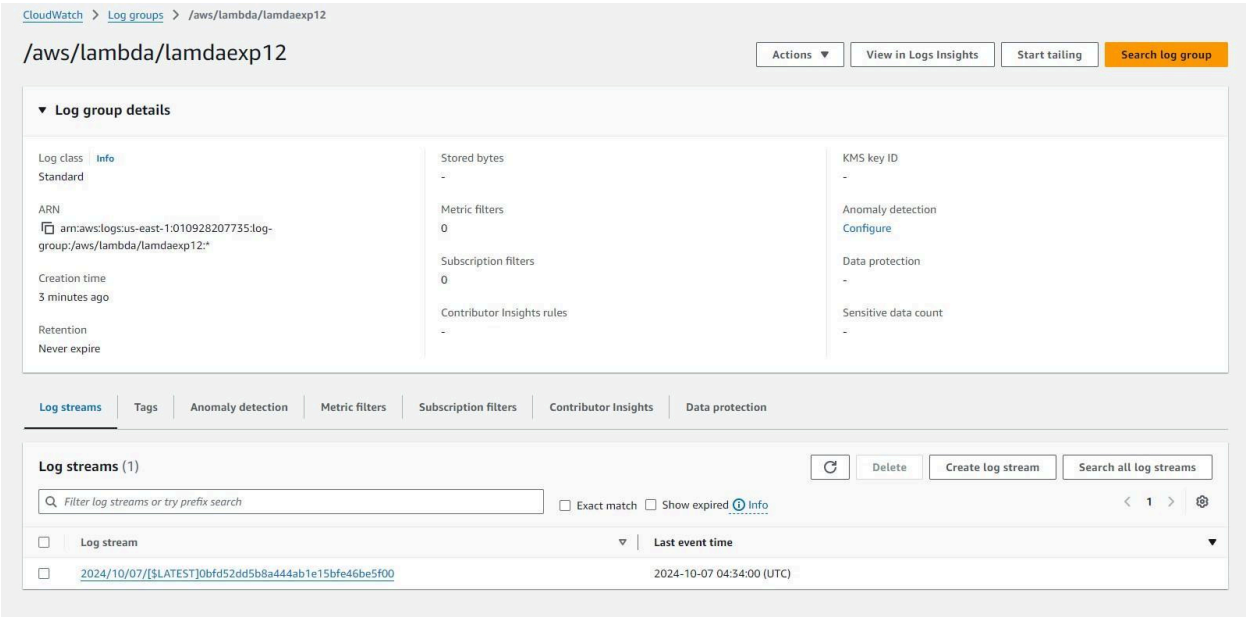
1) To check the logs explicitly, search foe CloudWatch on services and open it in a new tab



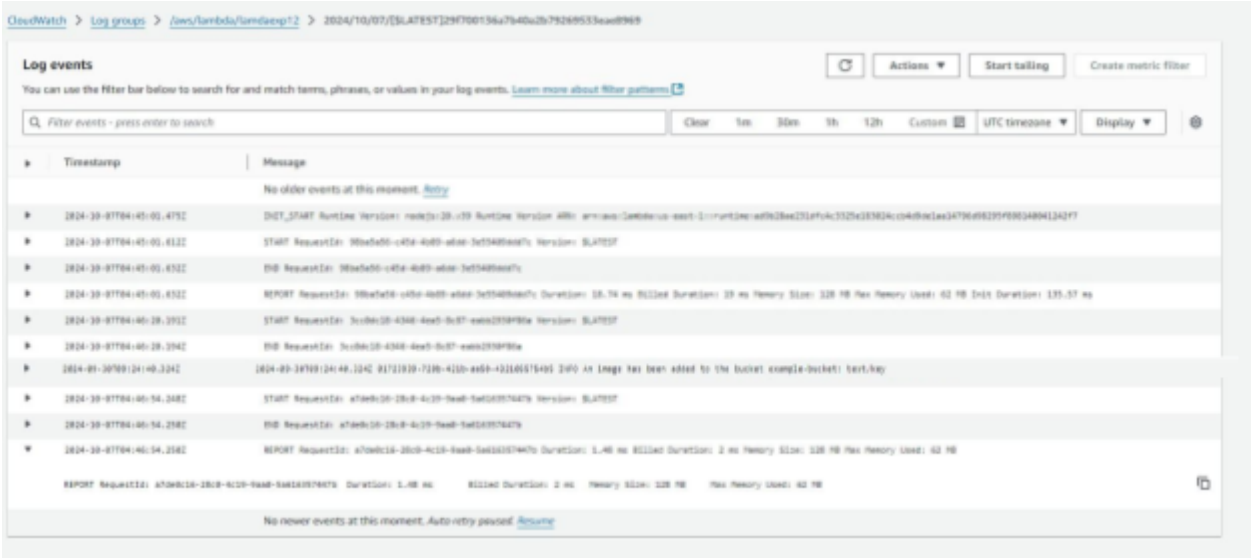
2) Here, Click on Logs → Log Groups. Select the log that has the lambda function name you just ran.



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



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



Conclusion:

In this experiment, In addition to demonstrating the integration of AWS Lambda with S3, this experiment showcases the scalability and flexibility of serverless architectures. By leveraging these services, we can build applications that respond in real-time to changes in data, such as the addition of files to S3 buckets, without the need for managing underlying server infrastructure. This not only enhances efficiency but also reduces operational costs, allowing developers to focus on building features rather than maintaining systems. Furthermore, the ability to log and monitor events through CloudWatch opens opportunities for further automation and analytics, paving the way for more complex workflows and data processing solutions.