Name: Shubham Jha **Roll_no: 19**

Div: D15C

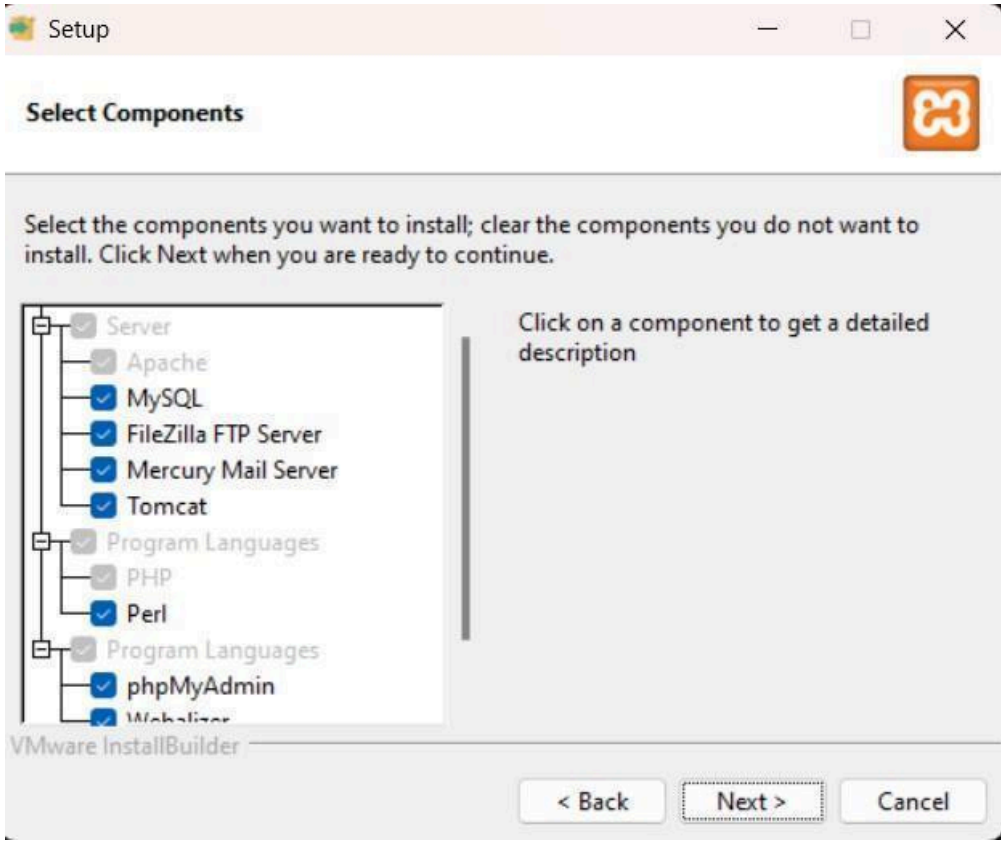
Static Hosting:

1) On local server (XAMPP)

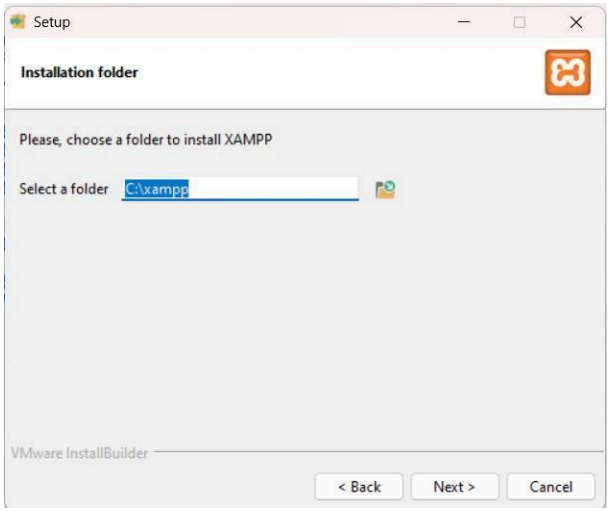
Step 1: Install XAMPP from https://www.apachefriends.org/

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

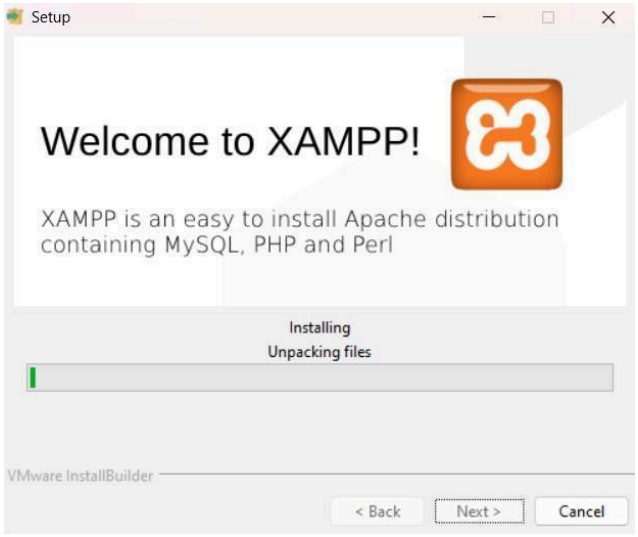
2) Open the setup file. Select all the required components and click next


Setup
Select Components
Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.
Server
Apache
MySQL
FileZilla FTP Server
Mercury Mail Server
Tomcat
Program Languages
PHP
Perl
Program Languages
phpMyAdmin
Click on a component to get a detailed description
VMware InstallBuilder
< Back
Next >
Cancel

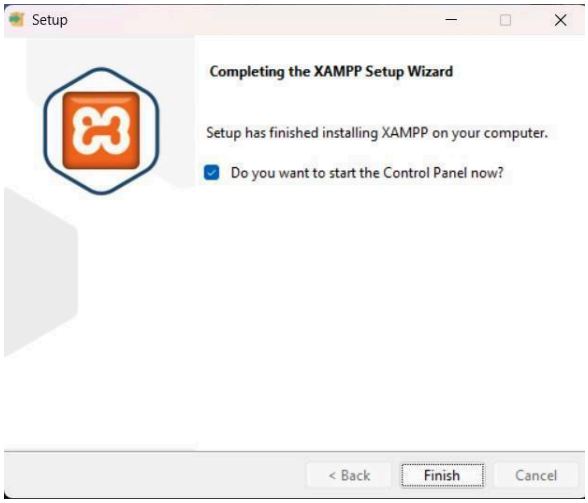
3) Choose the folder to install XAMPP in. Make sure the folder is empty. Click next



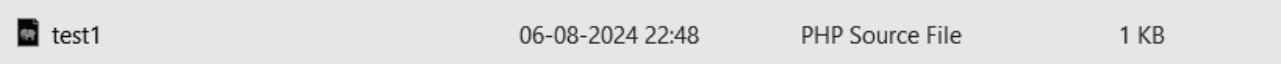
4) Select the language, click next. XAMPP starts to install



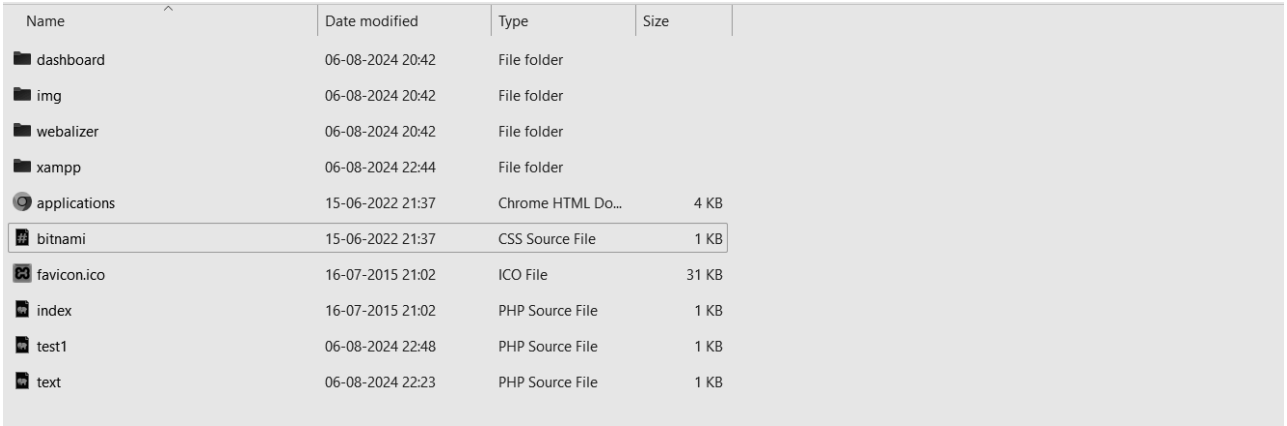
5) The installation is complete. Click Finish



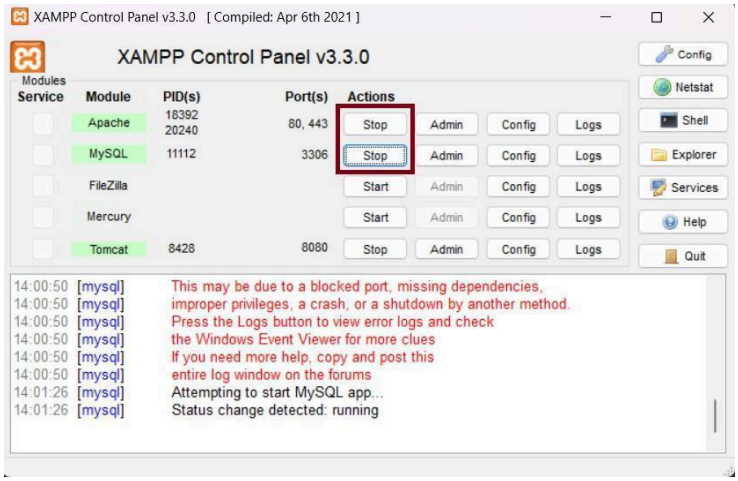
Step 2: Setup a file that is to be hosted on the server. Make sure the file has extension **.php**



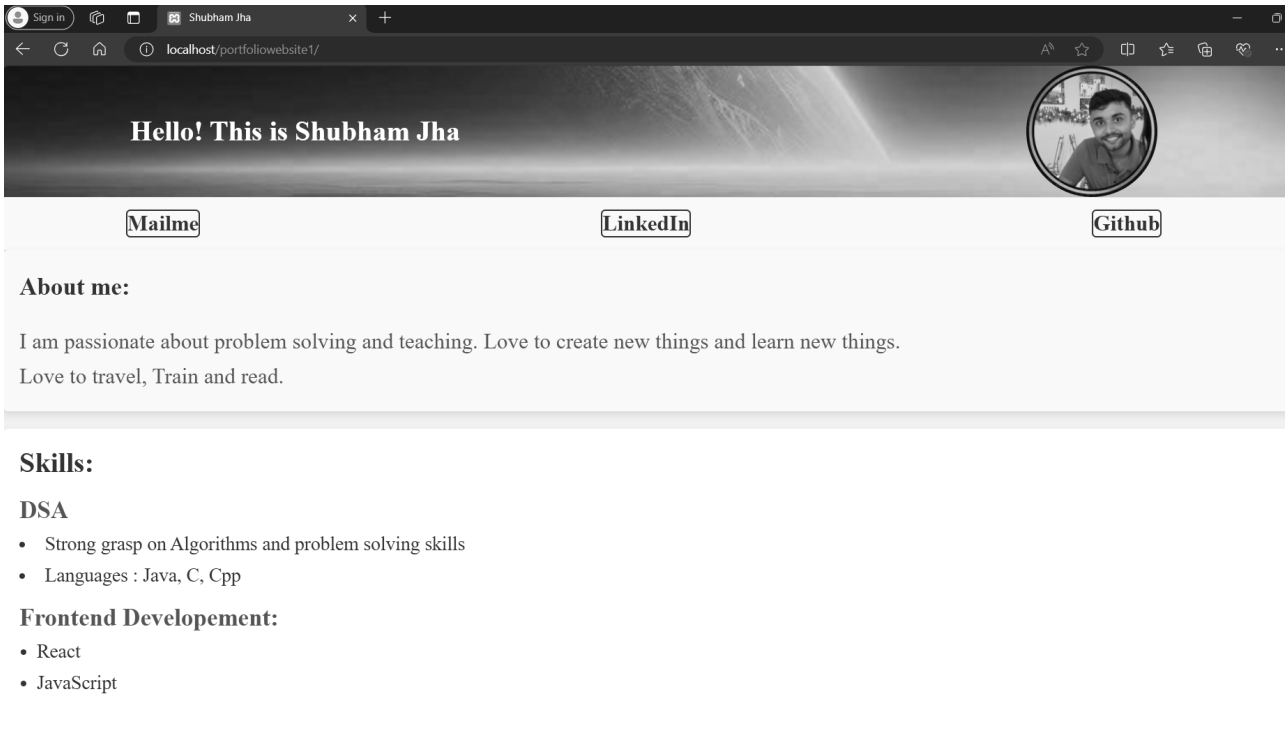
Step 3: Go to the directory where XAMPP was installed. Go to **htdocs** folder. Place your folder in this directory.



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

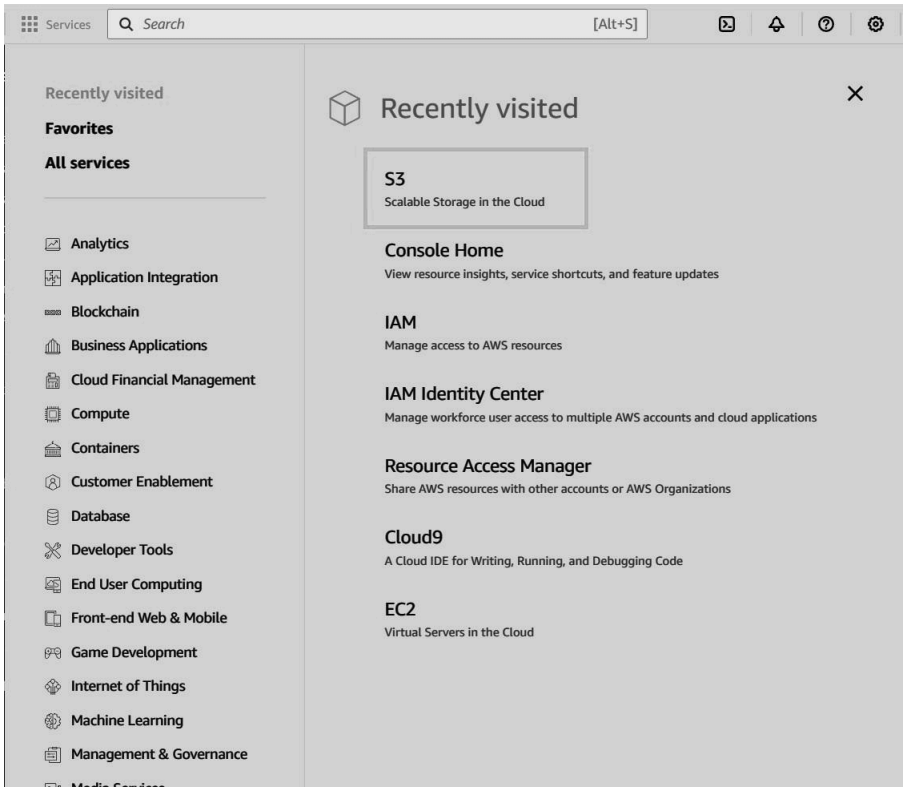


Step 5: Open your web browser. Type localhost/YOUR_FILENAME.php. This will open your website on your browser.

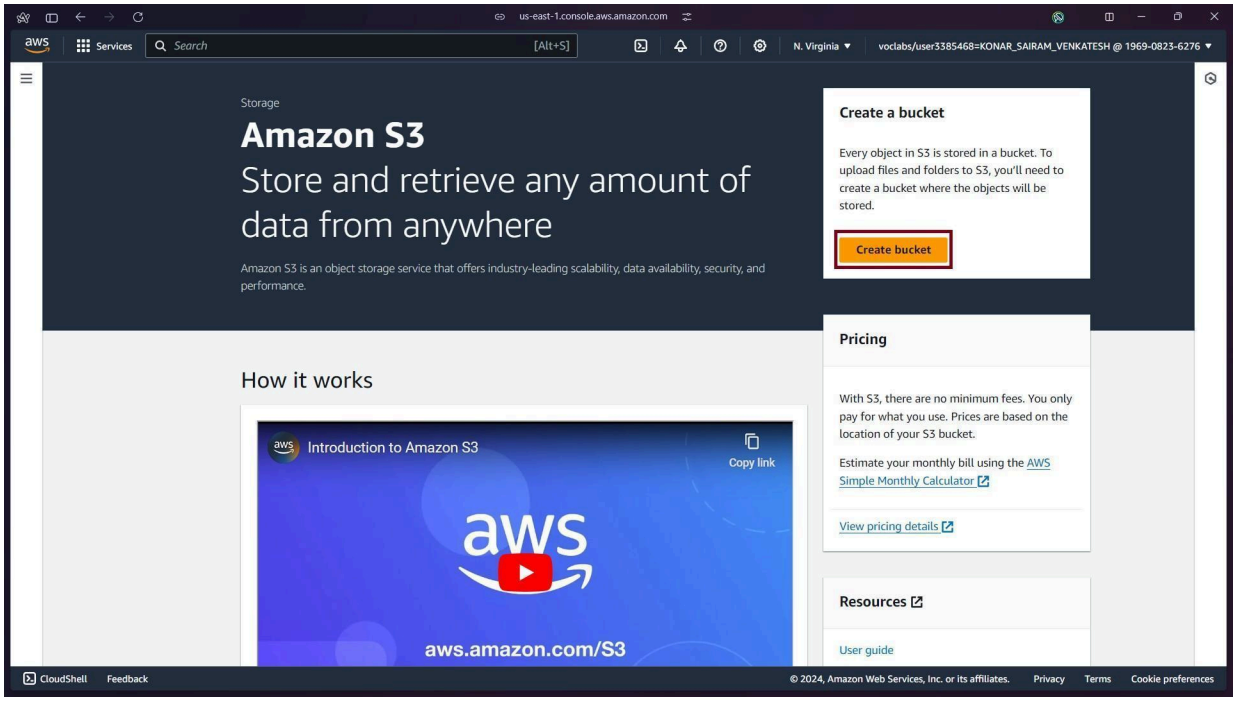


2) AWS S3

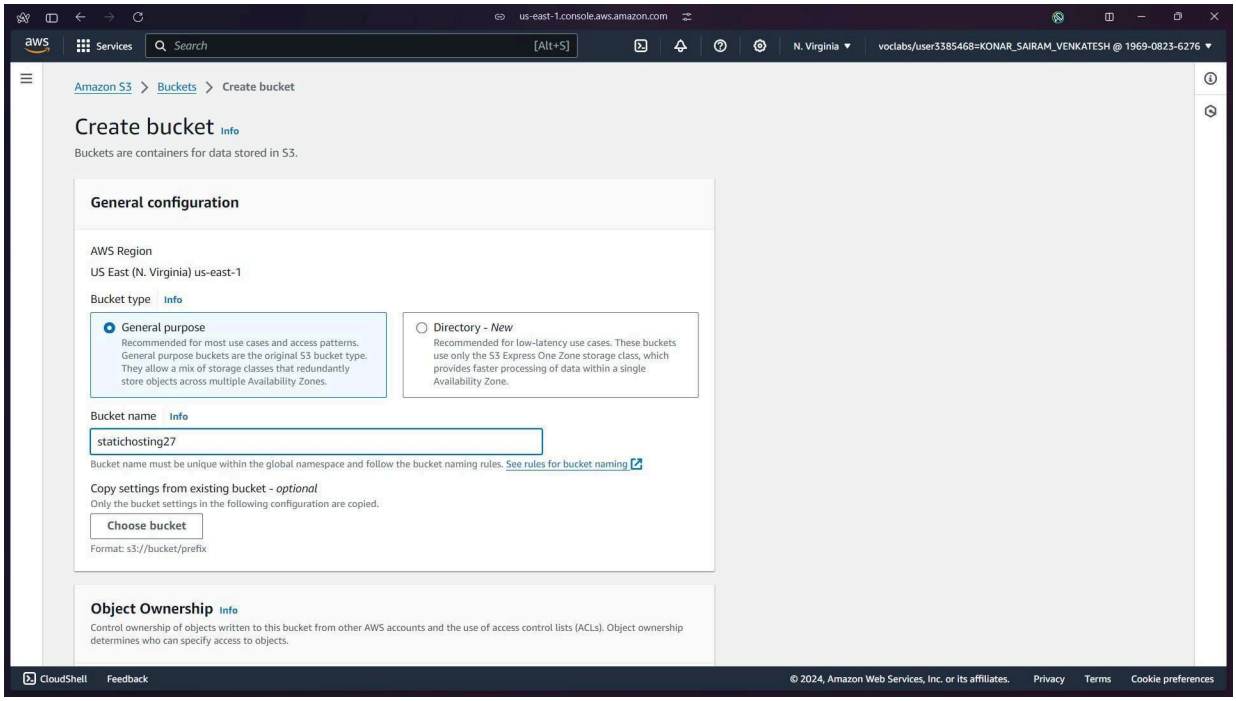
Step 1: Login to your AWS account. Go to services and open **S3.**



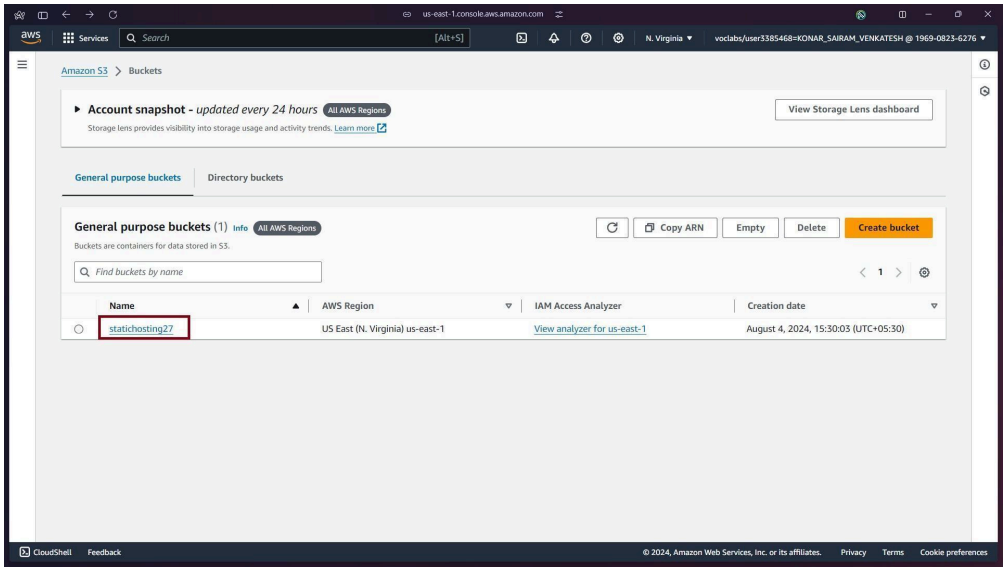
Step 2: Click on Create Bucket

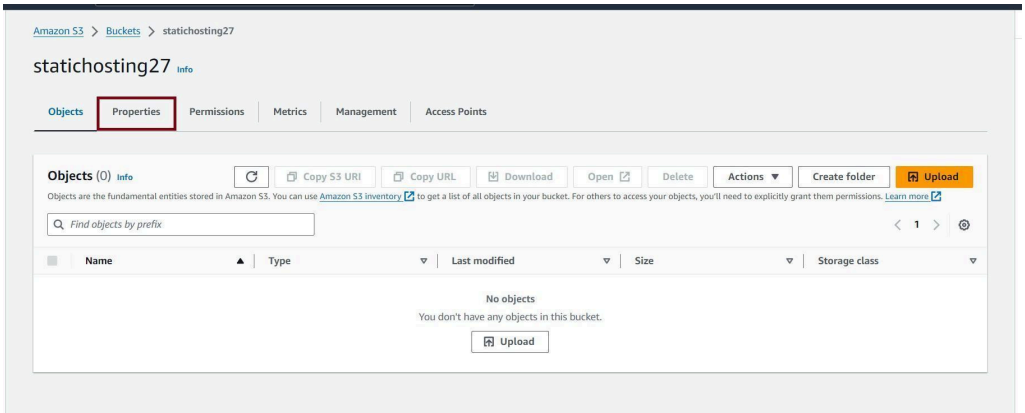


Step 3: Give a name to your bucket, keeping other options default, scroll down and click on Create Bucket

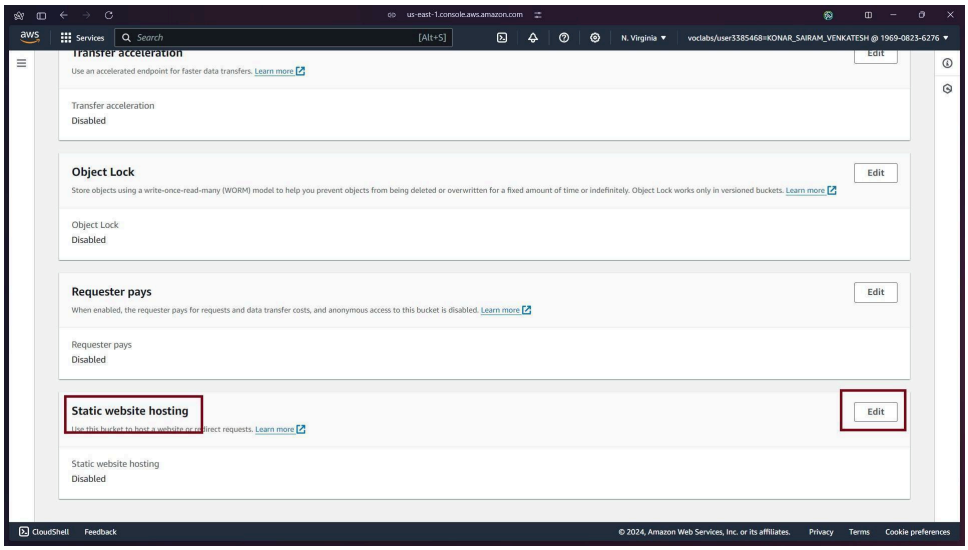


Step 4: Click on the name of your bucket and goto Properties

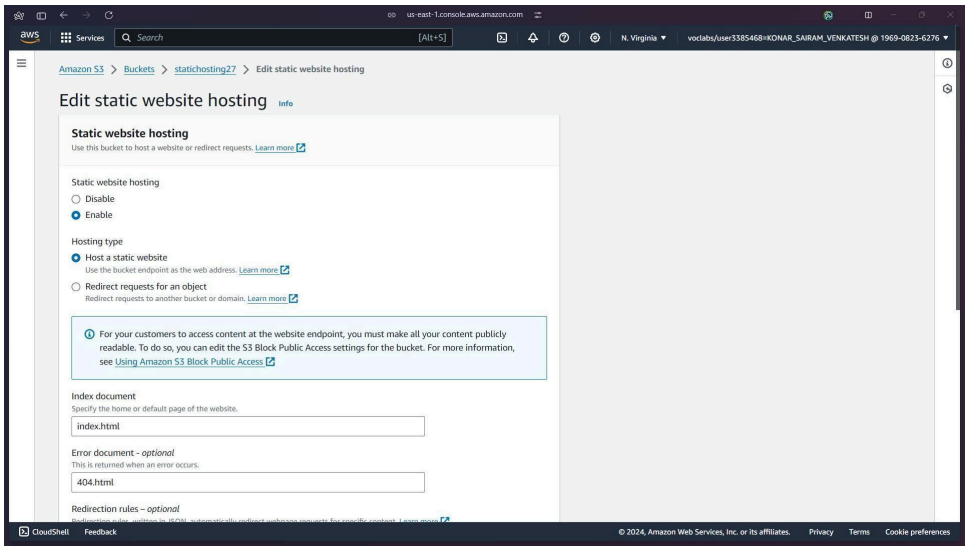




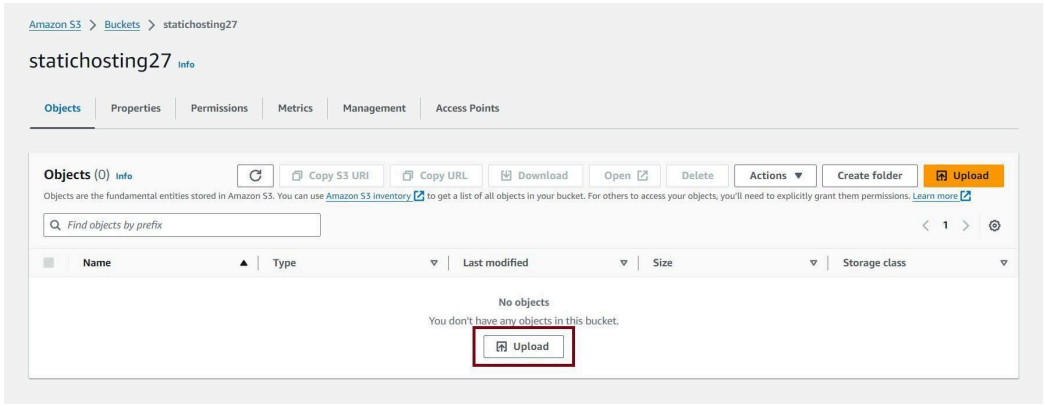
Step 5: Scroll down till you find Static website hosting, click on edit



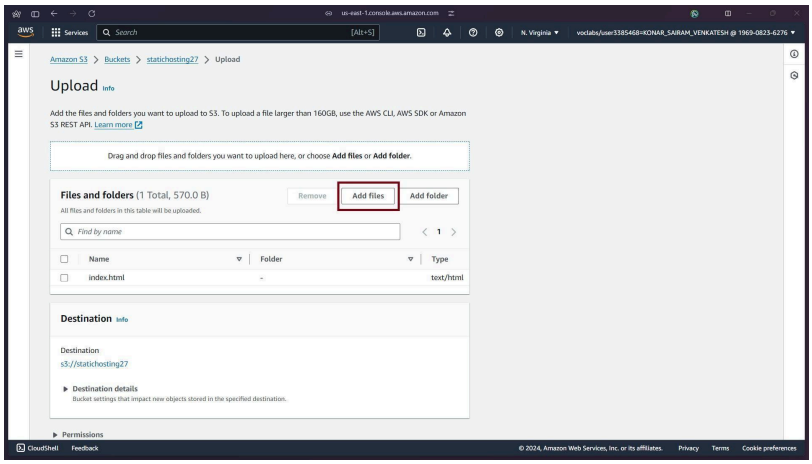
Step 6: Enable static website hosting, in Index document, write the name of your document and in error document, give name as 404.html. Save your changes.



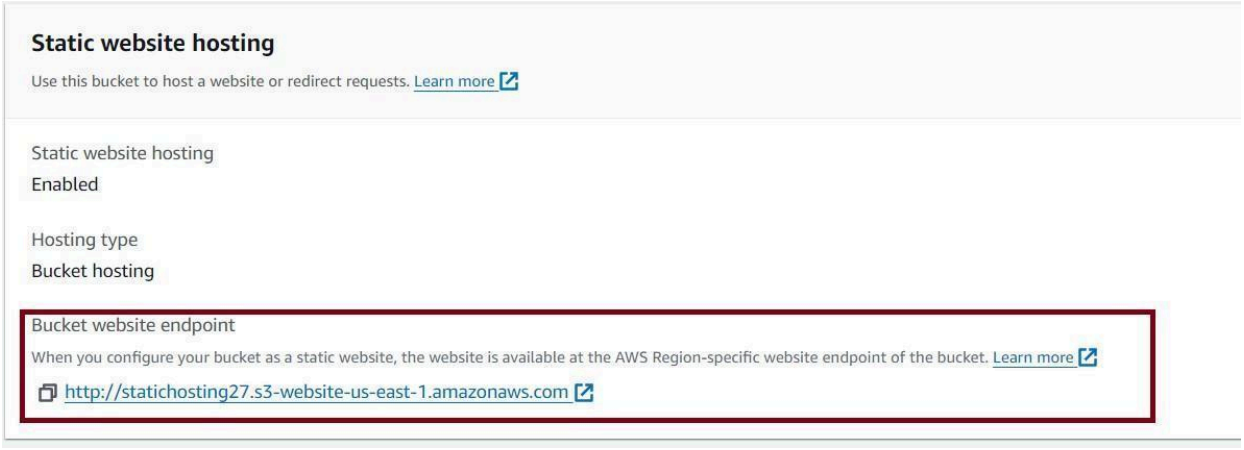
Step 7: Go to Objects tab and click on upload file.



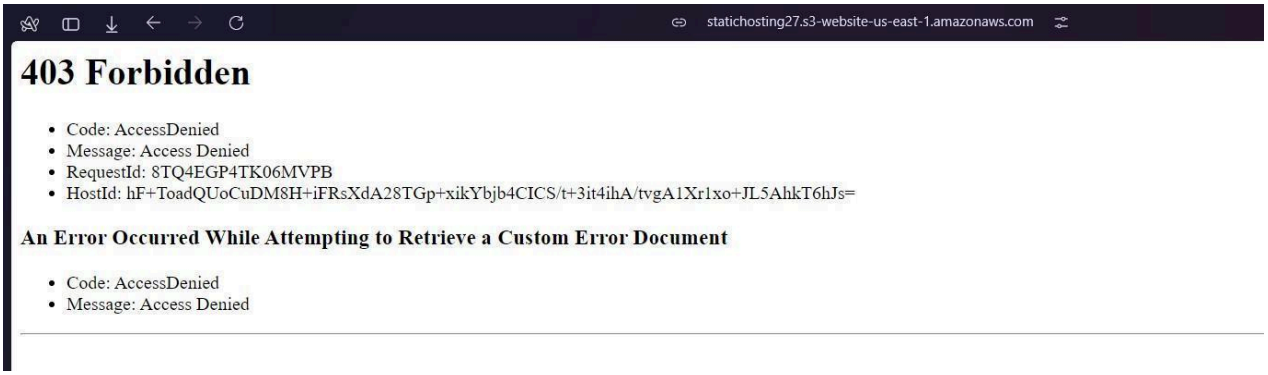
Step 8: Click on Add files. Add all the files you want to upload. Then scroll down and click on Upload



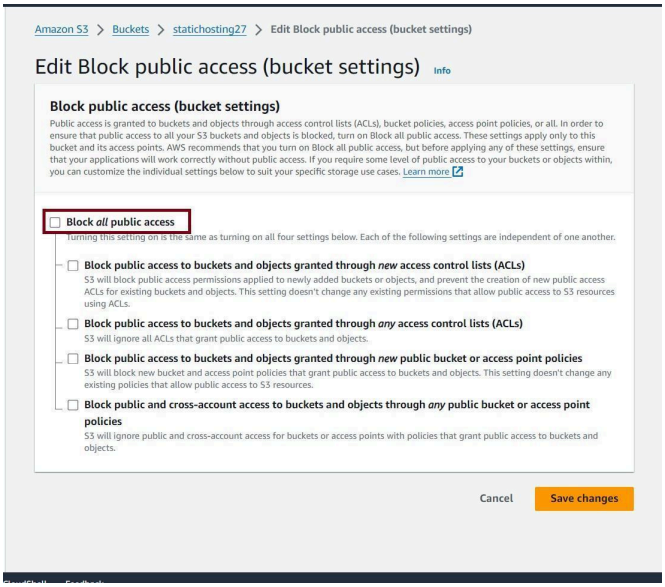
Step 9: This will take you to the Objects screen. Switch to Properties, scroll down to Static web hosting. There you would find the link (Bucket website endpoint) to your website.



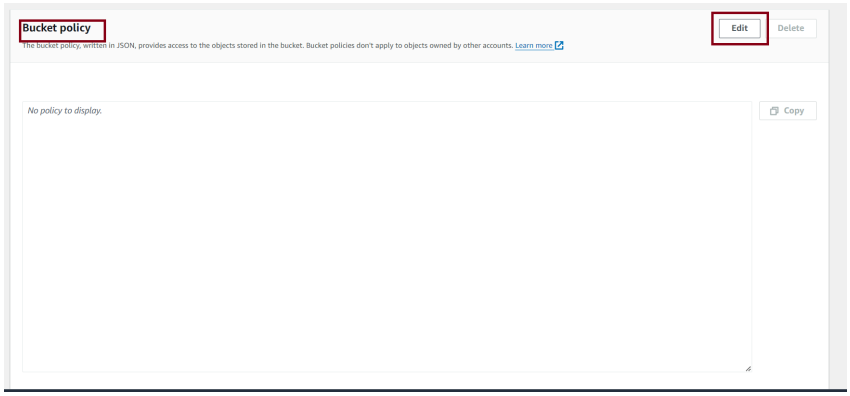
Step 10: Open the link. It will show a 403 forbidden error screen as the contents of the bucket are not available for the public users. To change this, go to Permissions tab, go to Block public access and click on edit



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



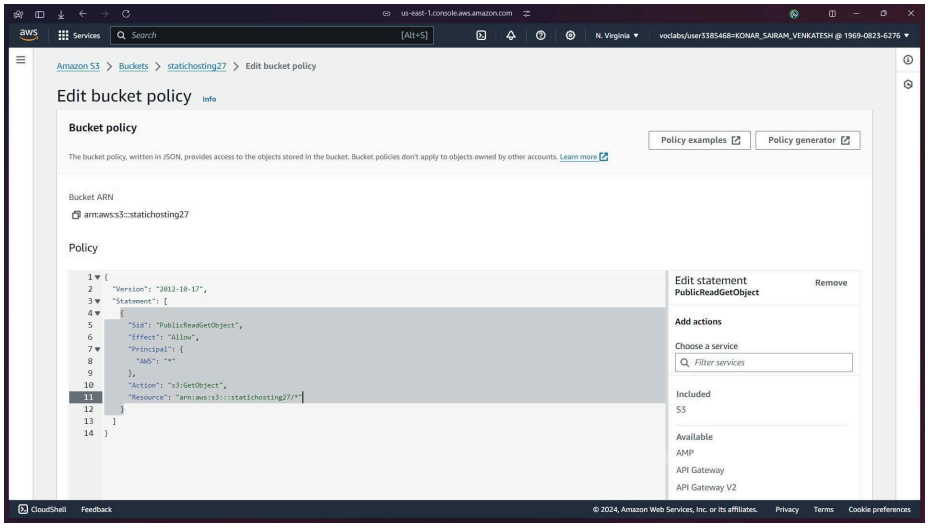
Step 12: Scroll down to bucket policy and click edit



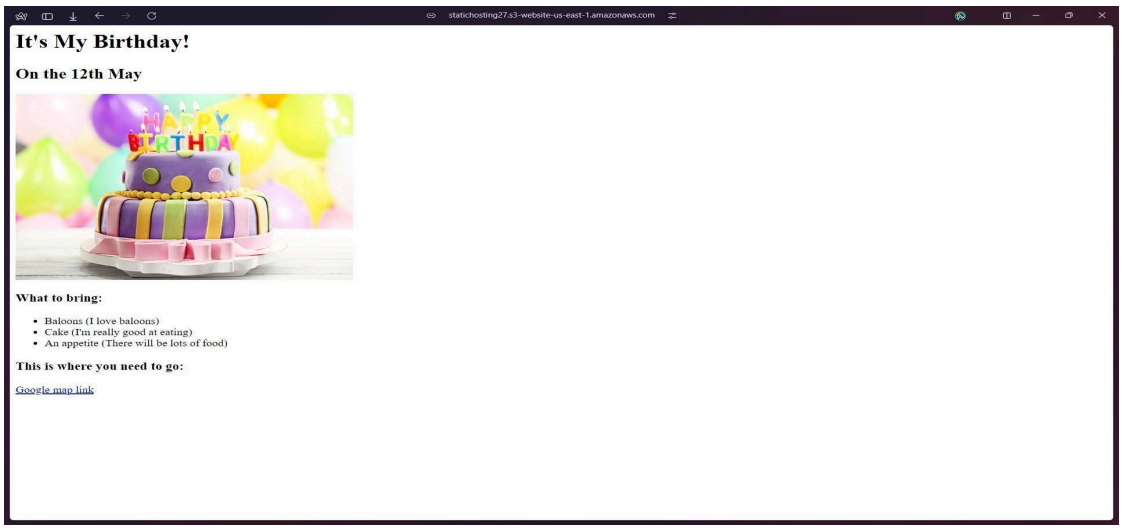
Step 13:

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

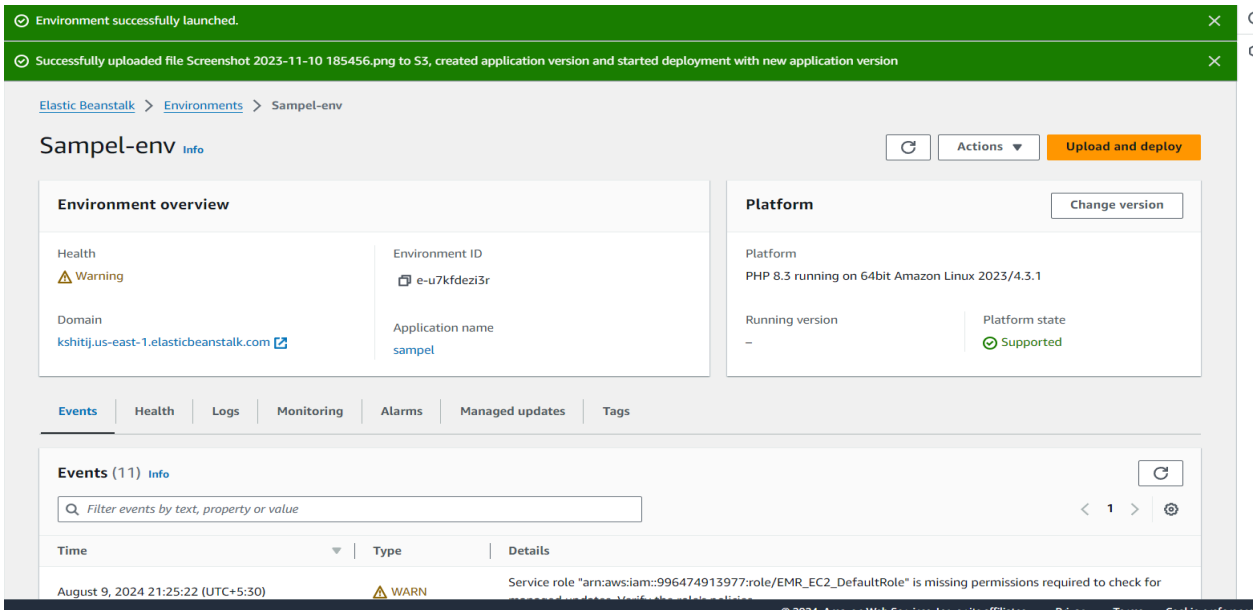
Paste this code snippet in the policy textarea. Replace YOUR-BUCKET-NAME-HERE with the name you have given to your bucket. Save the changes.

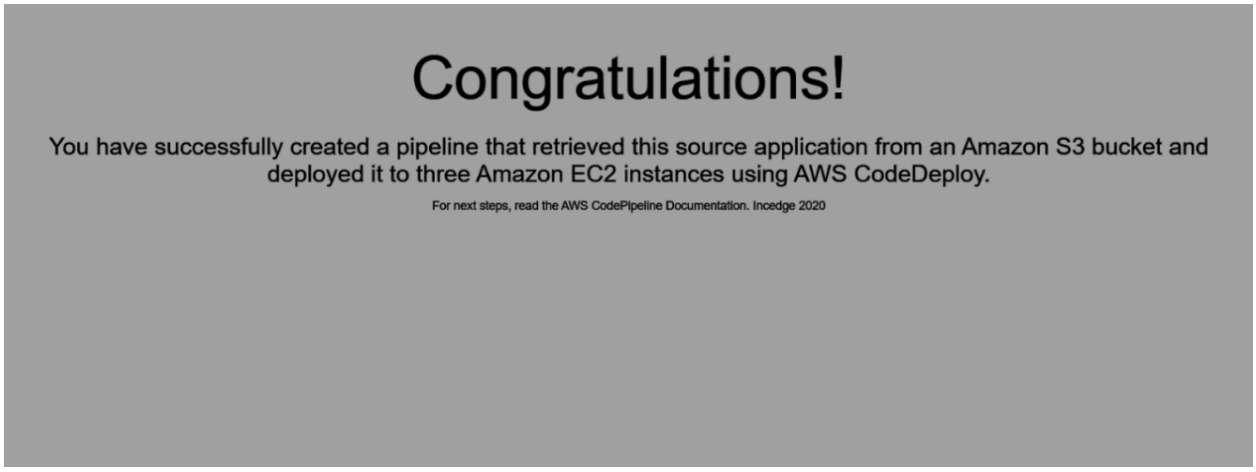


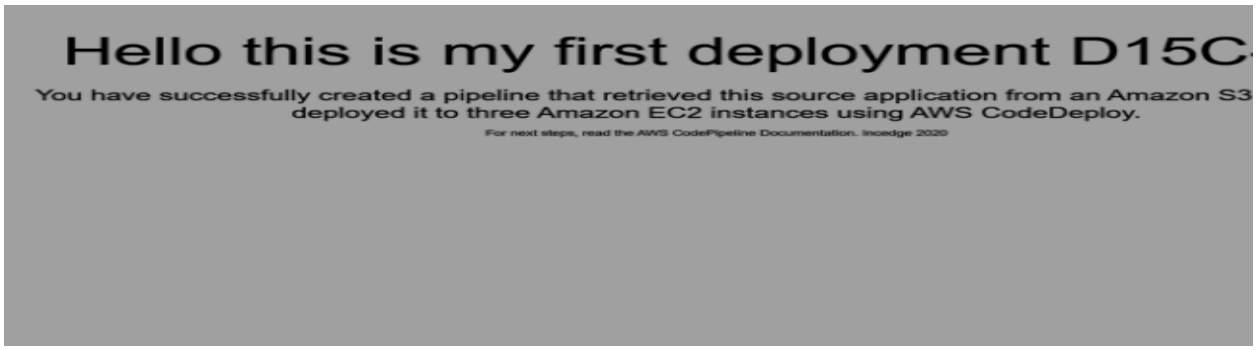
Step 14: Now reload the website. You can see your website



Practical No 2 : Elastic Beanstalk

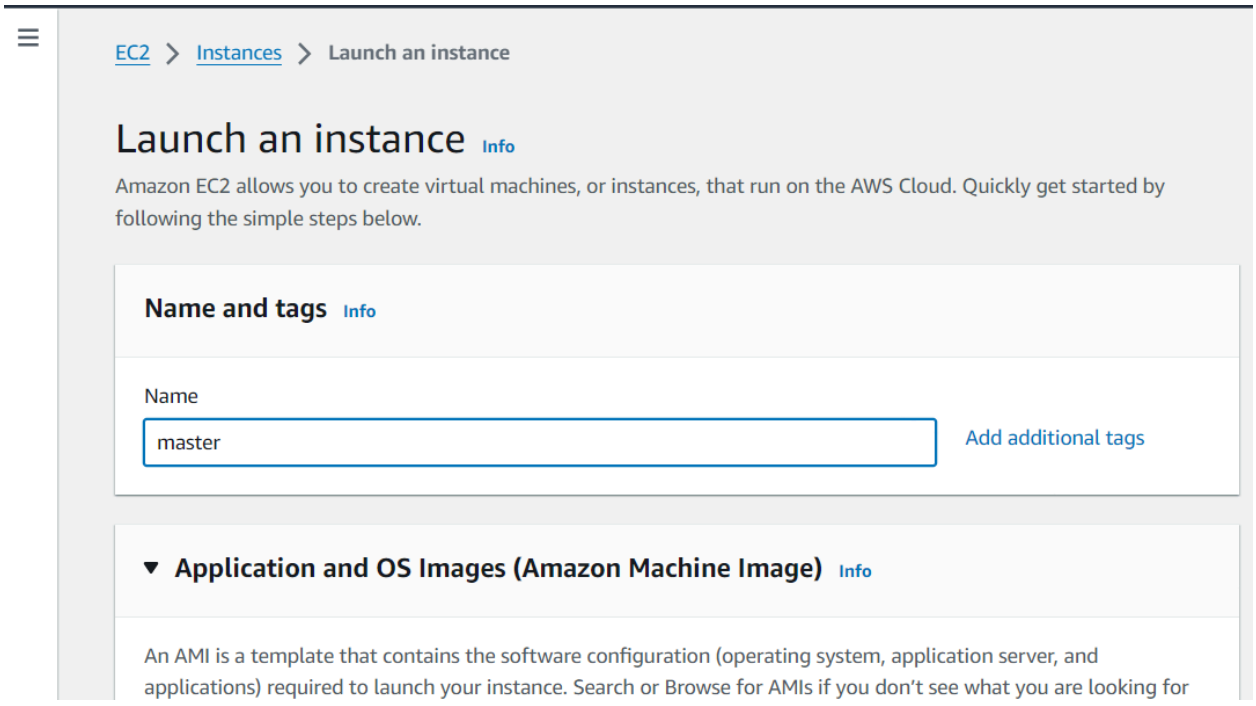




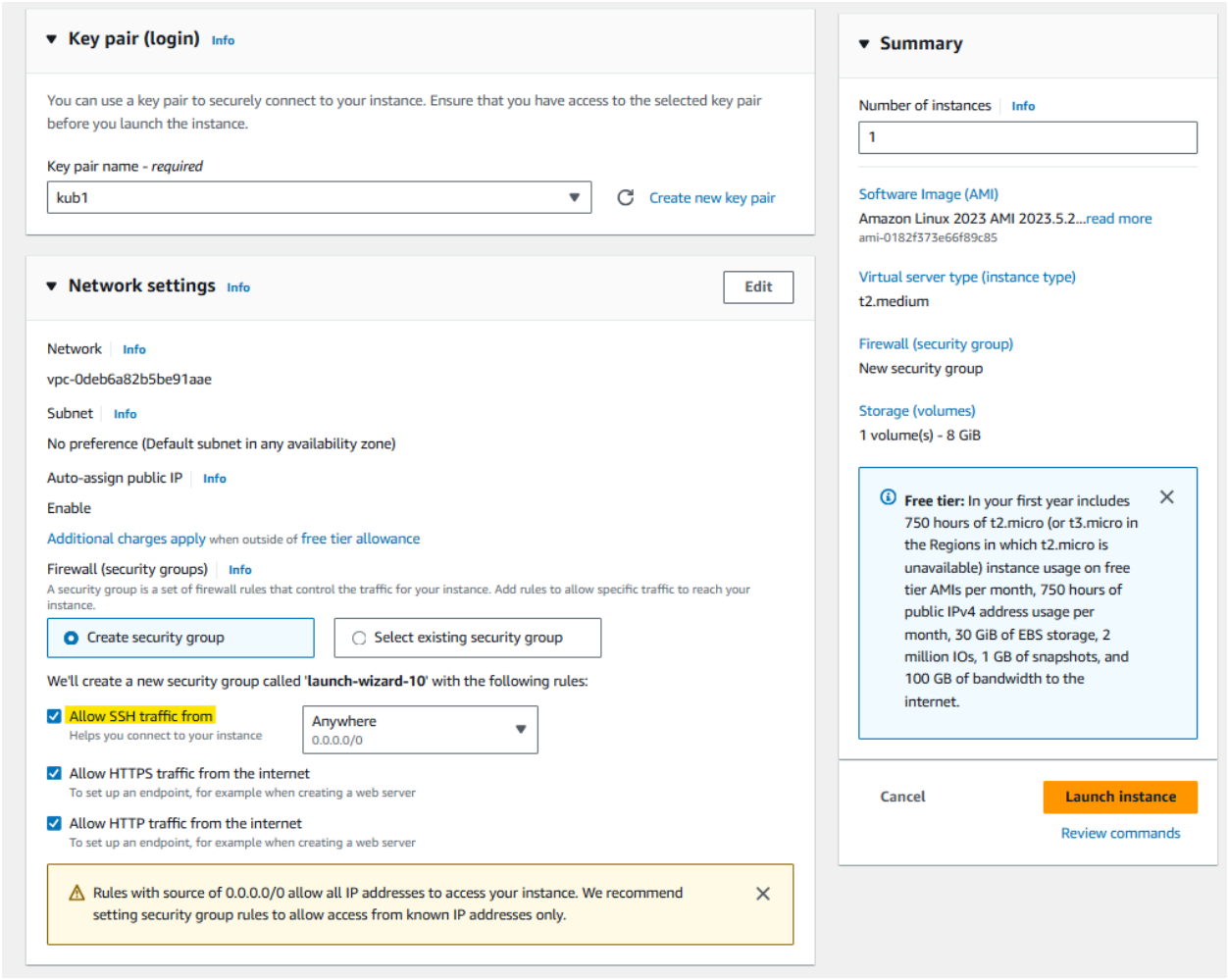


Aim: To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud

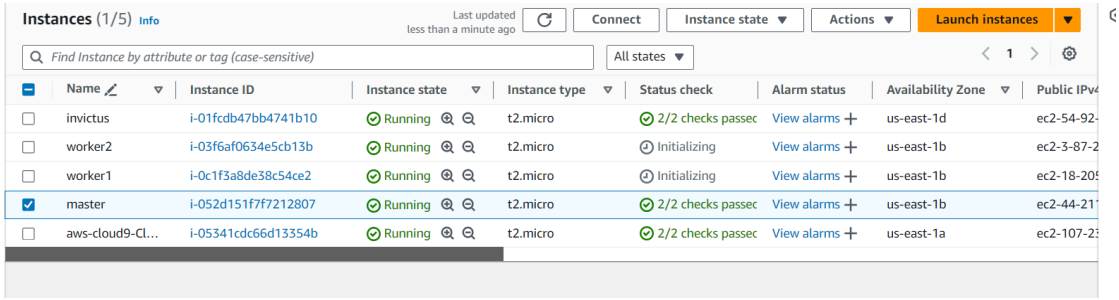
1. Create 3 EC-2 instances with all running on Amazon Linux as OS with inbound SSH allowed and the proper key



To efficient run kubernetes cluster select instance type of at least t2.medium as kubernetes recommends at least 2 vCPU to run smoothly



In this way create 3 instances namely master, worker-1 and worker-2



2. SSH into all 3 machines each in separate terminal

a. You can do it through the aws console directly

```
C:\Users\Avan>ssh -i "C:\Users\Avan\Downloads\kub1.pem" ec2-user@3.85.237.93
The authenticity of host '3.85.237.93 (3.85.237.93)' can't be established.
ED25519 key fingerprint is SHA256:Nz2iC26abFyhATf/8i4F0IgWmDoxTXBbzY9/NkMwYyM.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '3.85.237.93' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Thu Sep 12 13:11:49 2024 from 18.206.107.29
```

Or

b. Locate your key from the Downloads folder and open it in cmd and paste this command
ssh -i <-your-key->.pem ec2-user<ip-address of instance>

```
C:\Users\Avan\Downloads>ssh -i C:\Users\Avan\Downloads\kub1.pem ec2-user@54.85.79.186
The authenticity of host '54.85.79.186 (54.85.79.186)' can't be established.
ED25519 key fingerprint is SHA256:ffz946cqlxbNvBsPqtNcxLlfXmhW8VJhyLD4n4jStto.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '54.85.79.186' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Sat Sep 14 06:03:27 2024 from 18.206.107.27
[ec2-user@ip-172-31-20-75 ~]$
```

With this you can continue your commands through local terminal

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

Install Docker
sudo yum install docker -y

```
[ec2-user@ip-172-31-31-212 ~]$ sudo yum install docker -y
Last metadata expiration check: 0:33:43 ago on Thu Sep 12 13:11:13 2024.
Dependencies resolved.
================================================================================
 Package                 Architecture            Version
================================================================================
Installing:
 docker                  x86_64                  25.0.6-1.amzn2023.0.2
Installing dependencies:
 containerd              x86_64                  1.7.20-1.amzn2023.0.1
 iptables-libs           x86_64                  1.8.8-3.amzn2023.0.2
 iptables-nft            x86_64                  1.8.8-3.amzn2023.0.2
 libcgroup               x86_64                  3.0-1.amzn2023.0.1
 libnetfilter_conntrack  x86_64                  1.0.8-2.amzn2023.0.2
 libnfnetlink            x86_64                  1.0.1-19.amzn2023.0.2
 libnftnl                x86_64                  1.2.2-2.amzn2023.0.2
 pigz                    x86_64                  2.5-1.amzn2023.0.3
 runc                    x86_64                  1.1.13-1.amzn2023.0.1

Transaction Summary
```

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

- cd /etc/docker

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

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

```
[ec2-user@ip-172-31-20-75 ~]$ cd /etc/docker
[ec2-user@ip-172-31-20-75 docker]$ cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"],
"log-driver": "json-file",
"log-opts": {
"max-size": "100m"
},
"storage-driver": "overlay2"
}
EOF
{
"exec-opts": ["native.cgroupdriver=systemd"],
"log-driver": "json-file",
"log-opts": {
"max-size": "100m"
},
"storage-driver": "overlay2"
}
[ec2-user@ip-172-31-20-75 docker]$ ls
daemon.json  kubectl
```

- sudo systemctl enable docker
- sudo systemctl daemon-reload
- sudo systemctl restart docker
- docker -v

```
[ec2-user@ip-172-31-31-212 docker]$ sudo systemctl enable docker
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
[ec2-user@ip-172-31-31-212 docker]$ sudo systemctl daemon-reload
[ec2-user@ip-172-31-31-212 docker]$ sudo systemctl restart docker
[ec2-user@ip-172-31-31-212 docker]$ docker -v
Docker version 25.0.5, build 5dc9bcc
[ec2-user@ip-172-31-31-212 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-172-31-26-2 docker]$ sudo setenforce 0
[ec2-user@ip-172-31-26-2 docker]$ sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
[ec2-user@ip-172-31-26-2 docker]$
```

Add kubernetes repository (paste in terminal)

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

Type following commands to install set of kubernetes packages:

- sudo yum update

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

```
[ec2-user@ip-172-31-31-212 docker]$ sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
Last metadata expiration check: 0:00:23 ago on Thu Sep 12 14:09:10 2024.
Dependencies resolved.
================================================================================
 Package                 Architecture            Version
================================================================================
Installing:
 kubeadm                 x86_64                  1.30.5-150500.1.1
 kubectl                 x86_64                  1.30.5-150500.1.1
 kubelet                 x86_64                  1.30.5-150500.1.1
Installing dependencies:
 conntrack-tools         x86_64                  1.4.6-2.amzn2023.0.2
 cri-tools               x86_64                  1.30.1-150500.1.1
 kubernetes-cni          x86_64                  1.4.0-150500.1.1
 libnetfilter_cthelper   x86_64                  1.0.0-21.amzn2023.0.2
 libnetfilter_cttimeout  x86_64                  1.0.0-19.amzn2023.0.2
 libnetfilter_queue      x86_64                  1.0.5-2.amzn2023.0.2

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

Total download size: 53 M
Installed size: 292 M
Downloading Packages:
(1/9): libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64.rpm
(2/9): libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64.rpm
(3/9): libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64.rpm
(4/9): conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64.rpm
(5/9): kubeadm-1.30.5-150500.1.1.x86_64.rpm
(6/9): kubectl-1.30.5-150500.1.1.x86_64.rpm
(7/9): cri-tools-1.30.1-150500.1.1.x86_64.rpm
(8/9): kubernetes-cni-1.4.0-150500.1.1.x86_64.rpm
(9/9): kubelet-1.30.5-150500.1.1.x86_64.rpm
--------------------------------------------------------------------------------
Total
Kubernetes
Importing GPG key 0x9A296436:
 Userid     : "isv:kubernetes OBS Project <isv:kubernetes@build.opensuse.org>"
 Fingerprint: DE15 B144 86CD 377B 9E87 6E1A 2346 54DA 9A29 6436
 From       : https://pkgs.k8s.io/core:/stable:/v1.30/rpm/repodata/repomd.xml.key
Key imported successfully
Running transaction check
Transaction check succeeded.
```

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

5. Perform this ONLY on the Master machine

Initialize kubernetes by typing below command

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

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

Your Kubernetes control-plane has initialized successfully!

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

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

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 172.31.26.2:6443 --token 6cjsz0.8ei243v0zn9k7erg \
        --discovery-token-ca-cert-hash sha256:abd917ec30e12c5616bf647a3d174bef3d271e92c30b8f2f7768cfb3181341d4
[ec2-user@ip-172-31-26-2 docker]$
```

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

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

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

Example :

```
kubeadm join 172.31.20.75:6443 --token 66kg9u.2bc0kze31hrwbzvr \
        --discovery-token-ca-cert-hash
sha256:5e478da328b199e17d9b5da68e78bc9a6daab2043b05860552f4c184a7b3cb66
```

Then, add a common networking plugin called flamel file as mentioned in the code.
Command:

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

```
[ec2-user@ip-172-31-26-2 docker]$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
namespace/kube-flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
```

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

```
ec2-user@ip-172-31-31-212 docker]$ sudo yum install iproute-tc -y
ast metadata expiration check: 0:15:14 ago on Thu Sep 12 14:09:10 2024.
ependencies resolved.
================================================================================
 Package              Architecture        Version                 Repository     Size
================================================================================
nstalling:
 iproute-tc           x86_64              5.10.0-2.amzn2023.0.5   amazonlinux   455 k

ransaction Summary
================================================================================
nstall  1 Package

otal download size: 455 k
nstalled size: 928 k
ownloading Packages:
proute-tc-5.10.0-2.amzn2023.0.5.x86_64.rpm                  4.0 MB/s | 455 kB     00:00
--------------------------------------------------------------------------------
otal                                                        2.8 MB/s | 455 kB     00:00
unning transaction check
ransaction check succeeded.
unning transaction test
ransaction test succeeded.
unning transaction
  Preparing        :                                                         1/1
  Installing       : iproute-tc-5.10.0-2.amzn2023.0.5.x86_64                 1/1
  Running scriptlet: iproute-tc-5.10.0-2.amzn2023.0.5.x86_64                 1/1
  Verifying        : iproute-tc-5.10.0-2.amzn2023.0.5.x86_64                 1/1

nstalled:
  iproute-tc-5.10.0-2.amzn2023.0.5.x86_64

omplete!
ec2-user@ip-172-31-31-212 docker]$
```

```
[ec2-user@ip-172-31-17-184 docker]$ sudo systemctl enable kubelet
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service → /usr/lib/systemd/system/kubelet.service.
[ec2-user@ip-172-31-17-184 docker]$ sudo systemctl restart kubelet
[ec2-user@ip-172-31-17-184 docker]$
```

Now paste the hash that yo copied in these worker note to connect to master cluster

- kubeadm join 172.31.20.75:6443 --token 66kg9u.2bc0kze31hrwbzvr \
  --discovery-token-ca-cert-hash
sha256:5e478da328b199e17d9b5da68e78bc9a6daab2043b05860552f4c184a7b3cb66

Now we can see in the master/control node of kubernetes that worker nodes are connected by this command

- **watch kubectl get nodes**

(in the master node instance)

Errors faced during the execution :

1. In the end kubelet might not respond or the connectivity of nodes to master might not happen
2. You can see this error

```
[ec2-user@ip-172-31-20-75 docker]$ kubectl get nodes
E0914 06:14:55.956919    3650 memcache.go:265] couldn't get current server API group list: Get "https://172.31.20.75:6443/api?timeout=32s":
connection refused
E0914 06:14:55.957758    3650 memcache.go:265] couldn't get current server API group list: Get "https://172.31.20.75:6443/api?timeout=32s":
connection refused
E0914 06:14:55.959507    3650 memcache.go:265] couldn't get current server API group list: Get "https://172.31.20.75:6443/api?timeout=32s":
connection refused
E0914 06:14:55.960160    3650 memcache.go:265] couldn't get current server API group list: Get "https://172.31.20.75:6443/api?timeout=32s":
connection refused
E0914 06:14:55.961526    3650 memcache.go:265] couldn't get current server API group list: Get "https://172.31.20.75:6443/api?timeout=32s":
connection refused
```

3. Try to restart the kubelet from worker instance and try the commands again

Conclusion :

- The experiment aimed to deploy Kubernetes on Docker by connecting a master node to two worker nodes.
- Encountered issues with misconfigured SSH inbound rules.
- Resolved the SSH issue by correctly enabling necessary access rules.
- Realized that using t2.medium or t3 instances was essential to provide sufficient resources for Kubernetes.
- Despite adjustments, the worker nodes could not join the cluster.
- Master node was successfully initialized, but worker nodes faced issues.
- Possible causes of worker node problems included:
  - Misconfigurations in kubelet setup.
  - Networking challenges.
  - Inability of worker nodes to communicate with the master node's API server.
  - Potential incorrect firewall settings.
  - Missing API server certificates.
  - Errors during the kubeadm join process on the worker nodes.

Aim

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

Theory:

Kubernetes, originally developed by Google, is an open-source container orchestration platform. It automates the deployment, scaling, and management of containerized applications, ensuring high availability and fault tolerance. Kubernetes is now the industry standard for container orchestration and is governed by the **Cloud Native Computing Foundation (CNCF)**, with contributions from major cloud and software providers like Google, AWS, Microsoft, IBM, Intel, Cisco, and Red Hat.

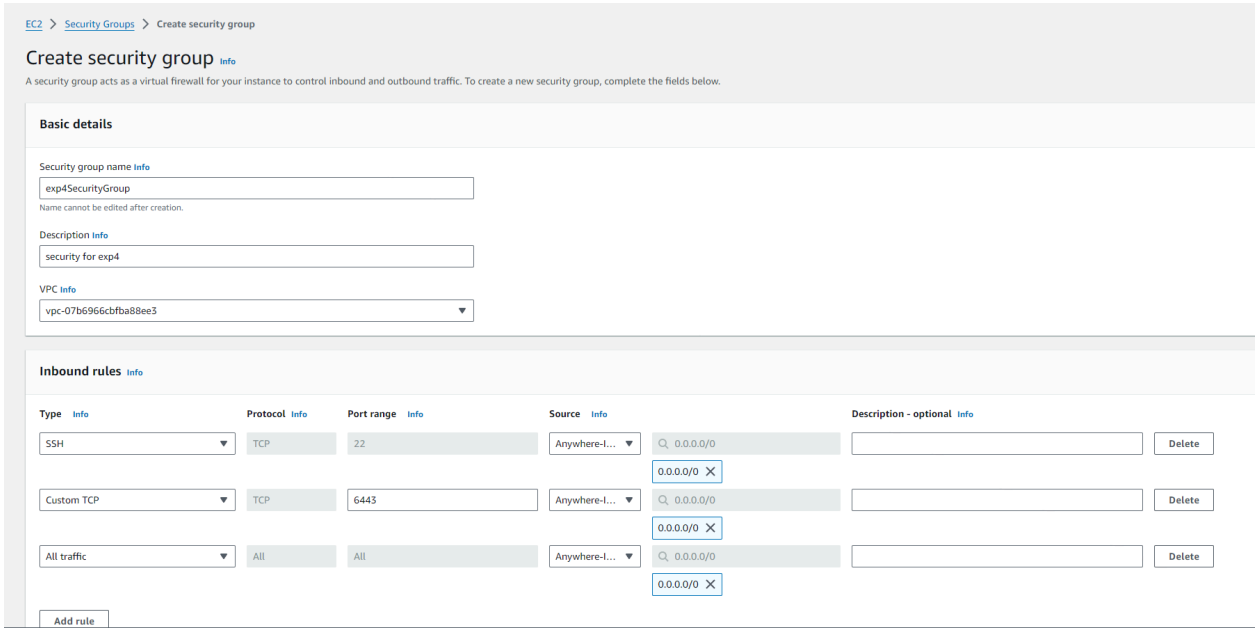
Kubernetes Deployment: Is a resource in Kubernetes that provides declarative updates for Pods and ReplicaSets. With a Deployment, you can define how many replicas of a pod should run, roll out new versions of an application, and roll back to previous versions if necessary. It ensures that the desired number of pod replicas are running at all times.

Steps:

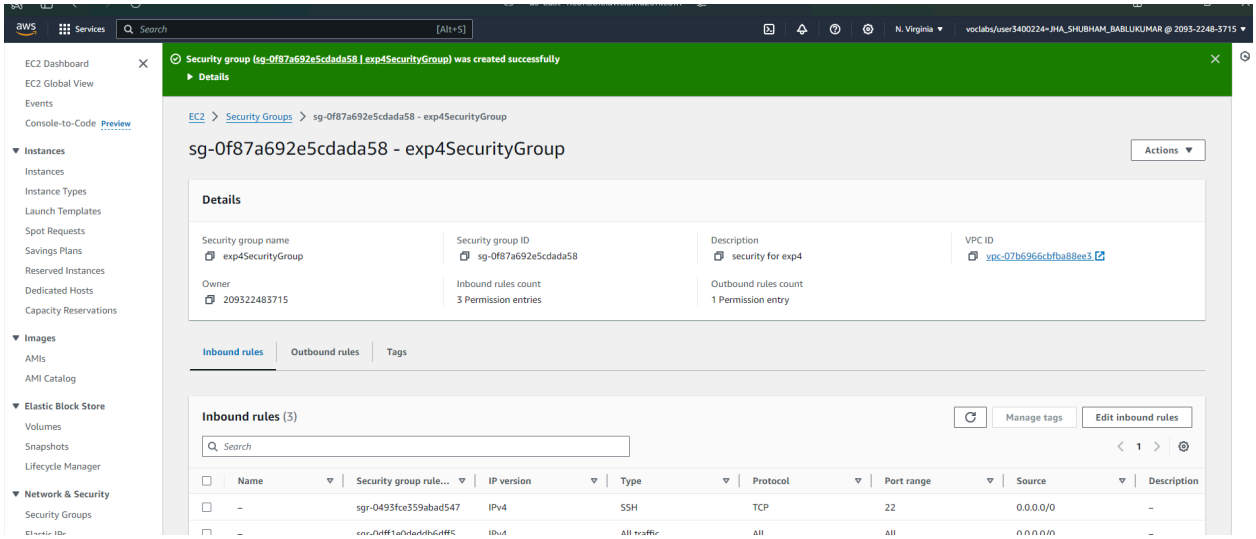
Log in to your AWS Academy/personal account.

1. Create security group

create security group with following configuration lets name it as exp4.

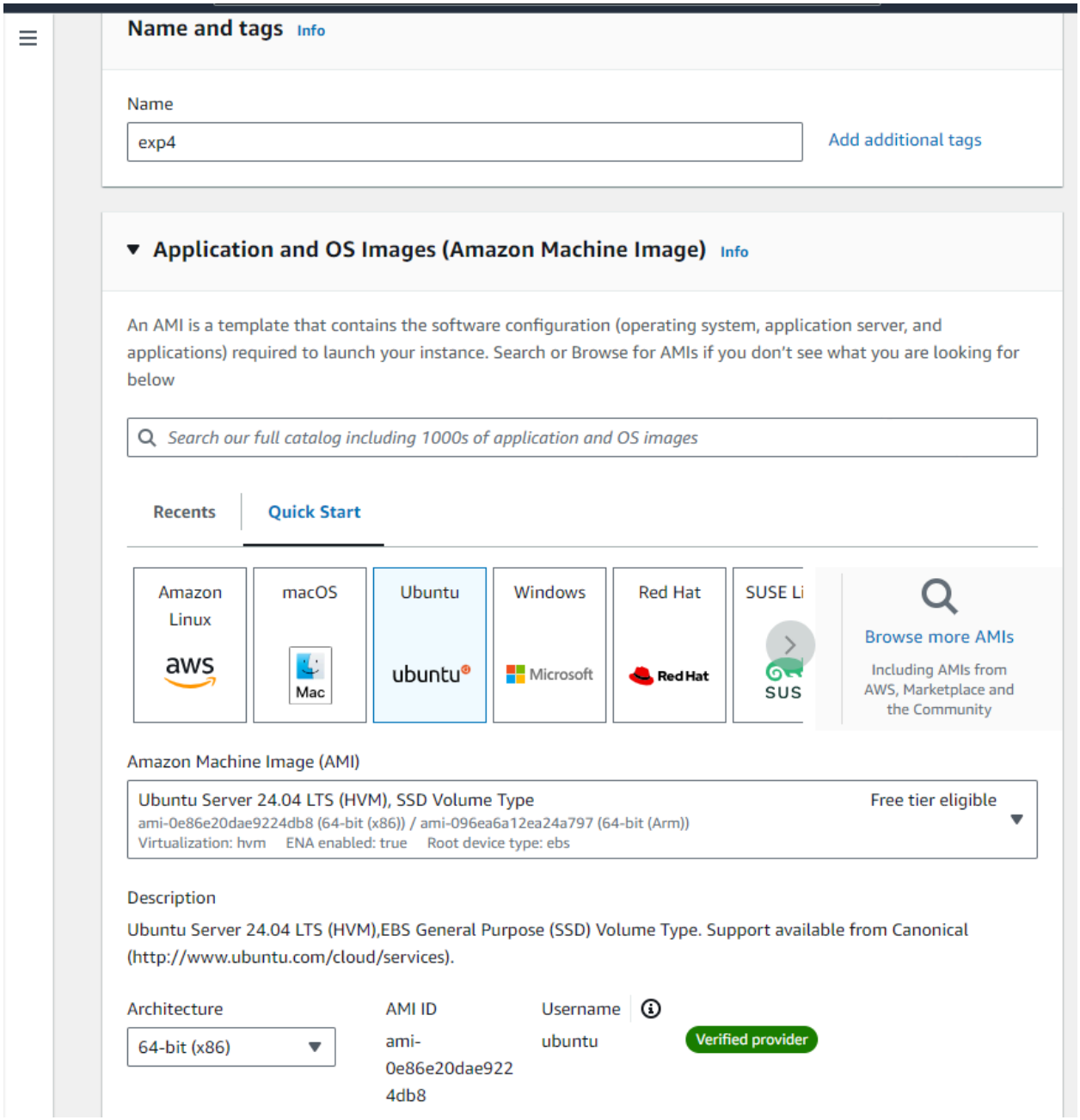


click on create security group.

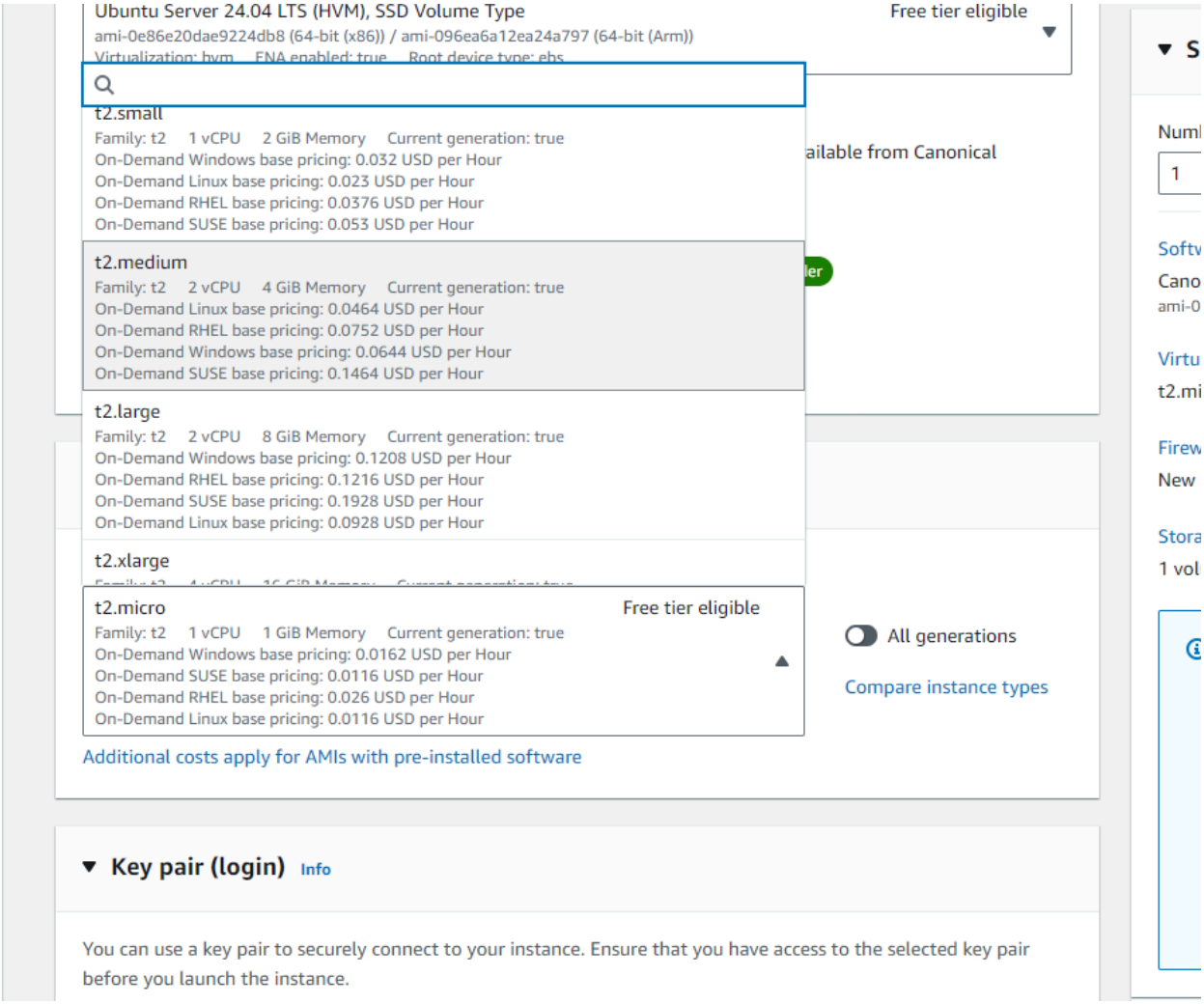


2. Create Instance

Launch an ec2 instance.

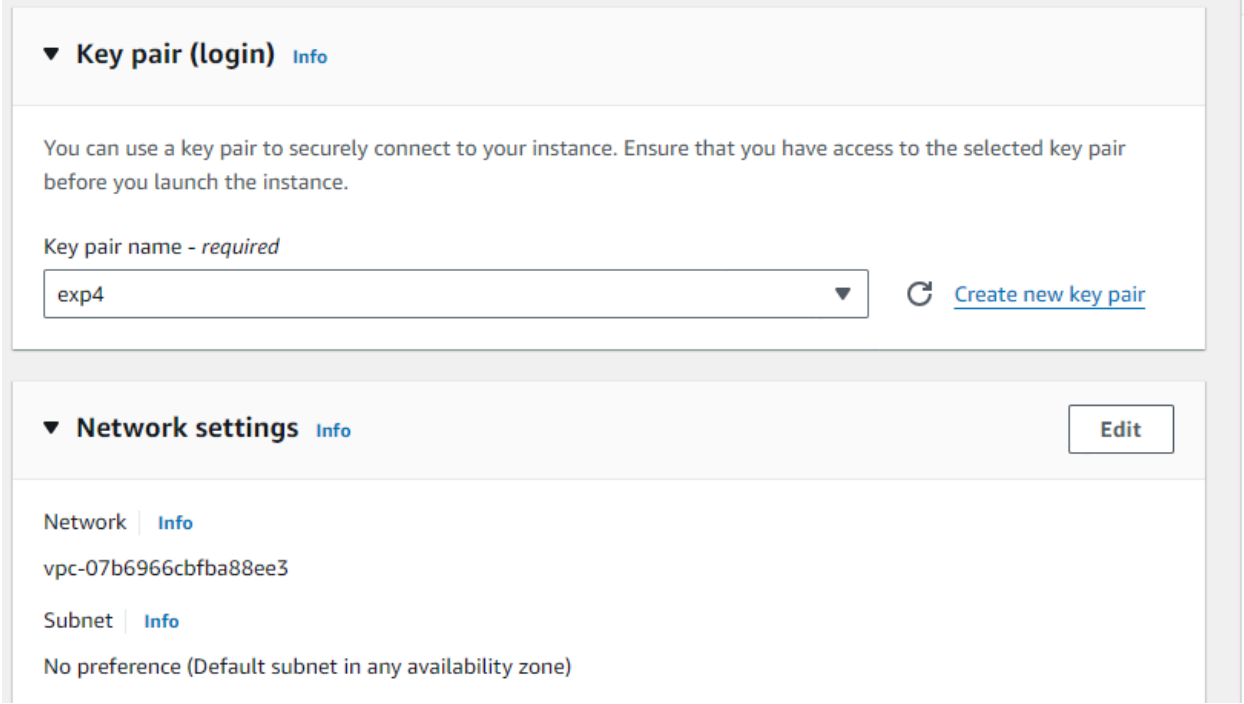


Select Ubuntu 22.04 as AMI and t2.medium as Instance Type, create a key of type RSA with .pem extension, and move the downloaded key to the new folder.

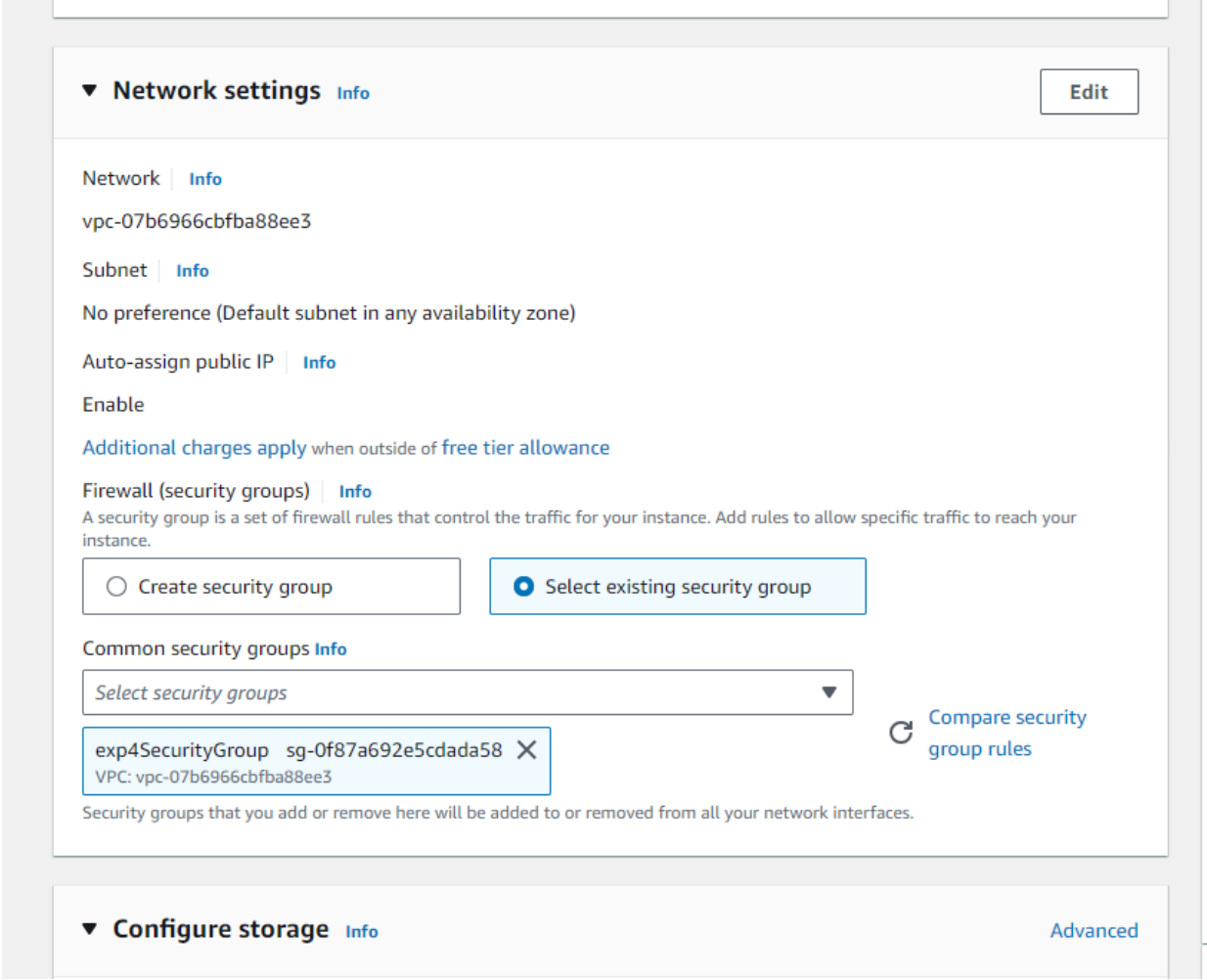


Note: A minimum of 2 CPUs are required so Please select t2.medium and do not forget to stop the instance after the experiment because it is not available in the free tier.

Create a key-pair to login to the machine remotely and then select this newly generated key-pair.



Select existing security group and select the security group we created at the start.

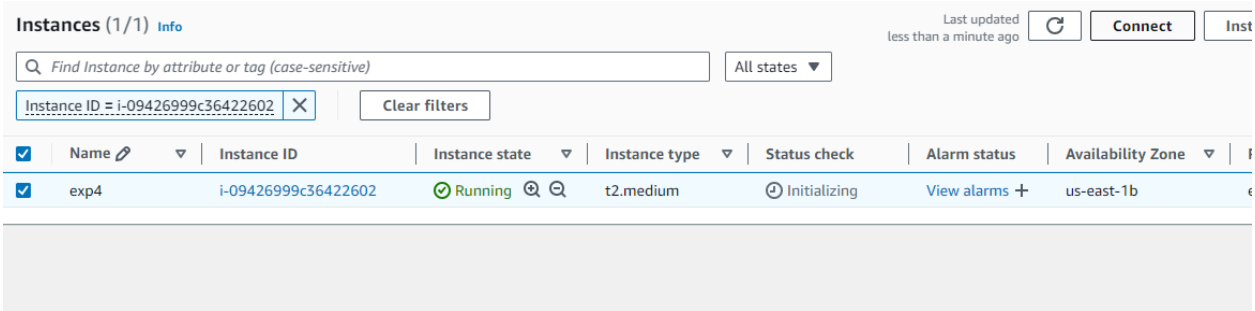


Launch the instance.

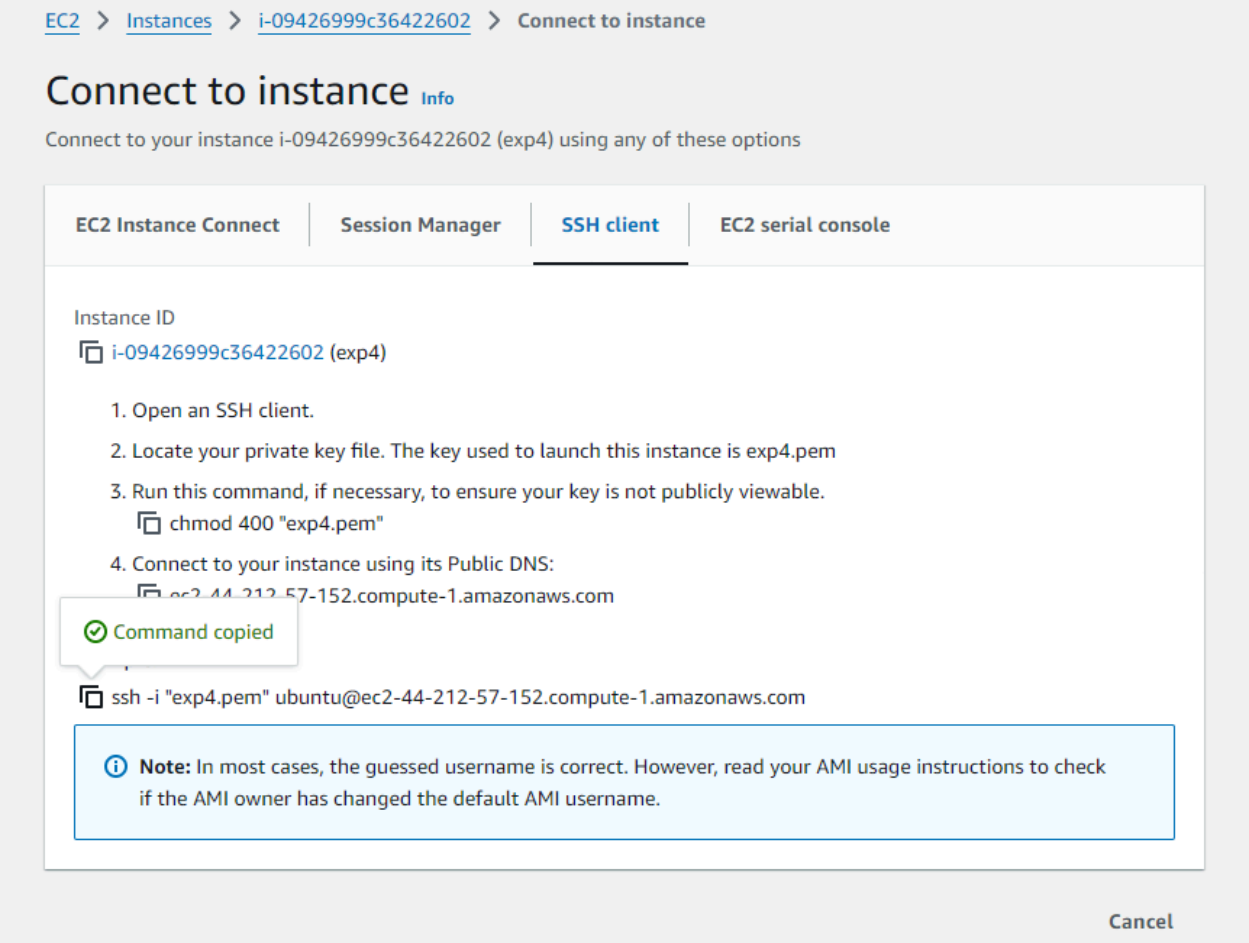
3. Connect to the instance.

Select the instance created

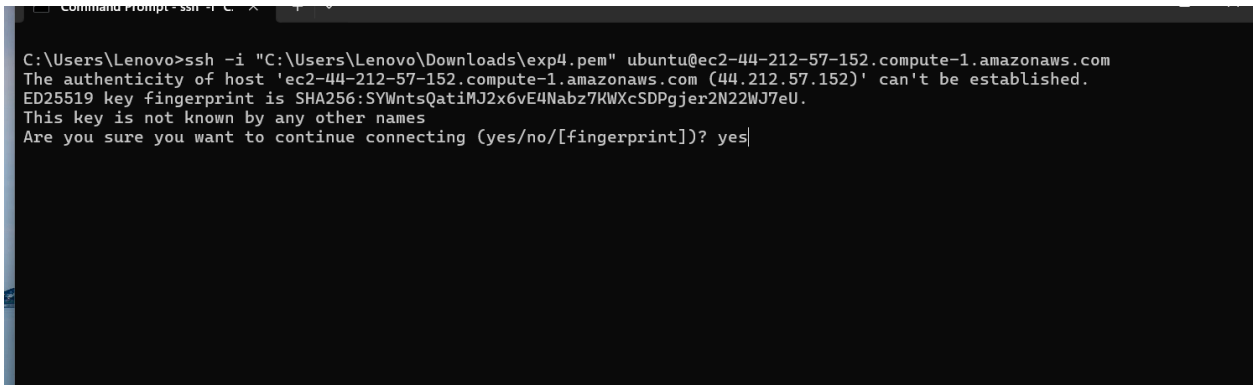
click on Connect the instance and navigate to SSH Client.



Copy the command that comes to your dashboard at the bottom.



Now copy the path to the file where our .pem key is stored and replace the pem file in the command copied from the ssh dashboard.

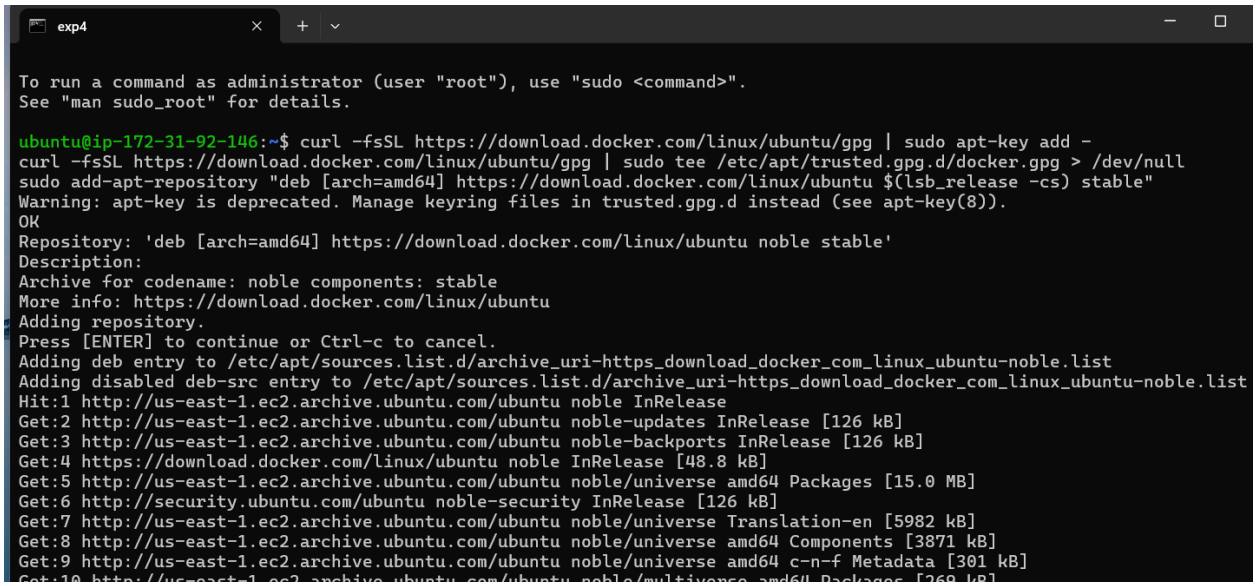


4. Install and set-up Docker

Run the following commands:

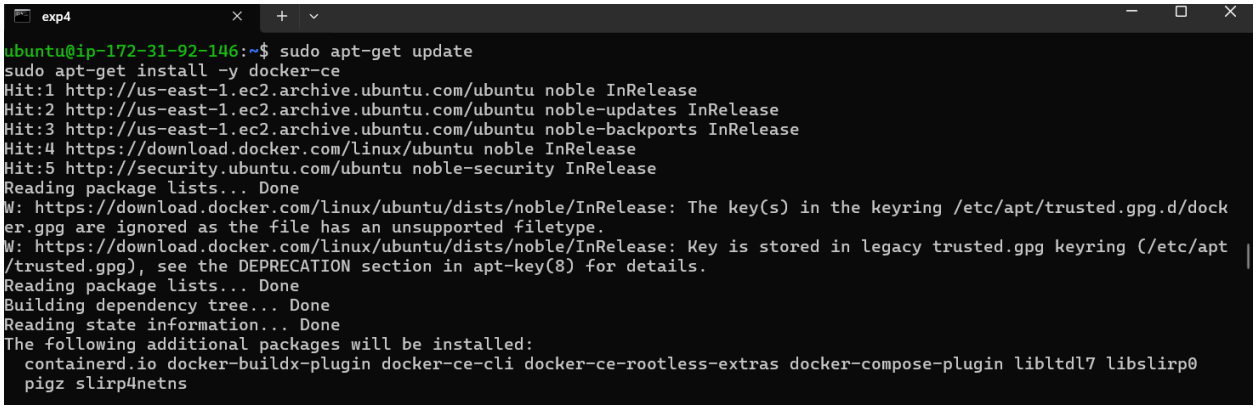
1. We have to install and setup Docker. Run these commands

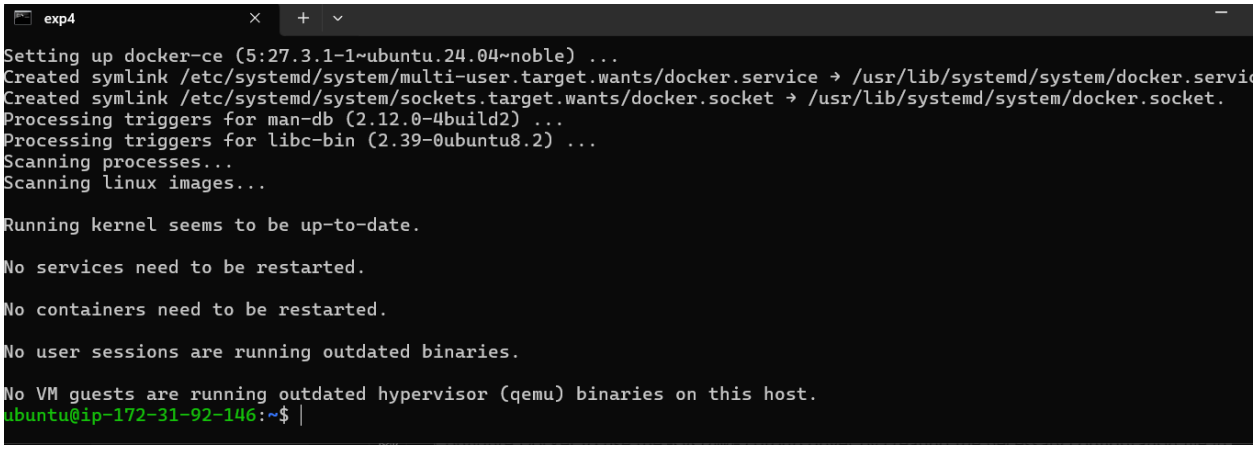
```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/trusted.gpg.d/docker.gpg > /dev/null
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
```



2. Update

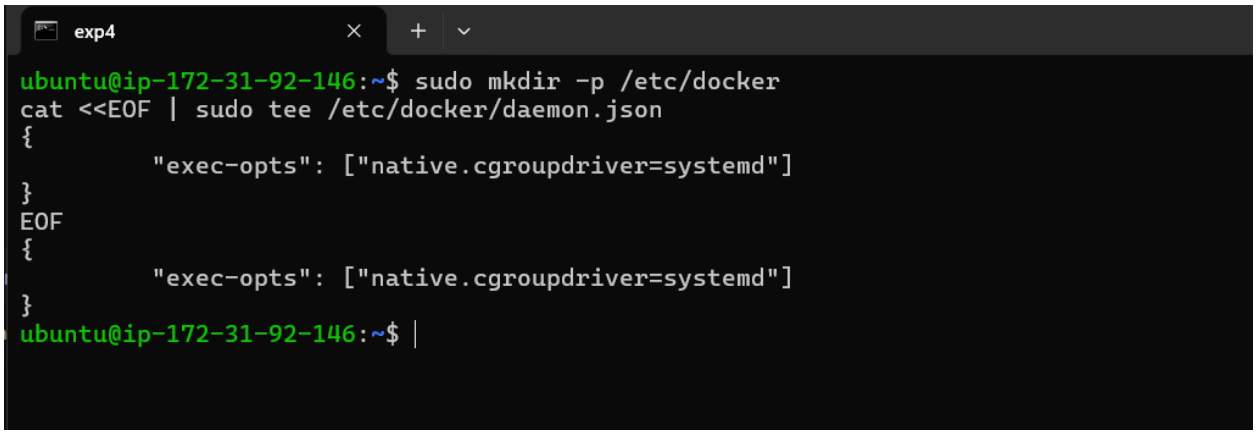
```
sudo apt-get update
sudo apt-get install -y docker-ce
```





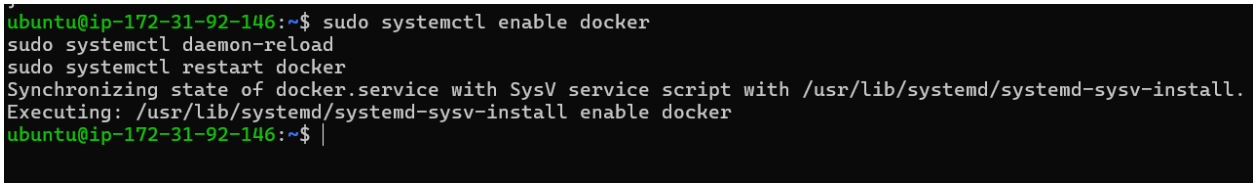
3. Configure Docker to use the systemd cgroup driver by creating the necessary configuration file in the /etc/docker directory.

```
sudo mkdir -p /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
         "exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
```



4. Restart and enable docker:

```
        sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker
```



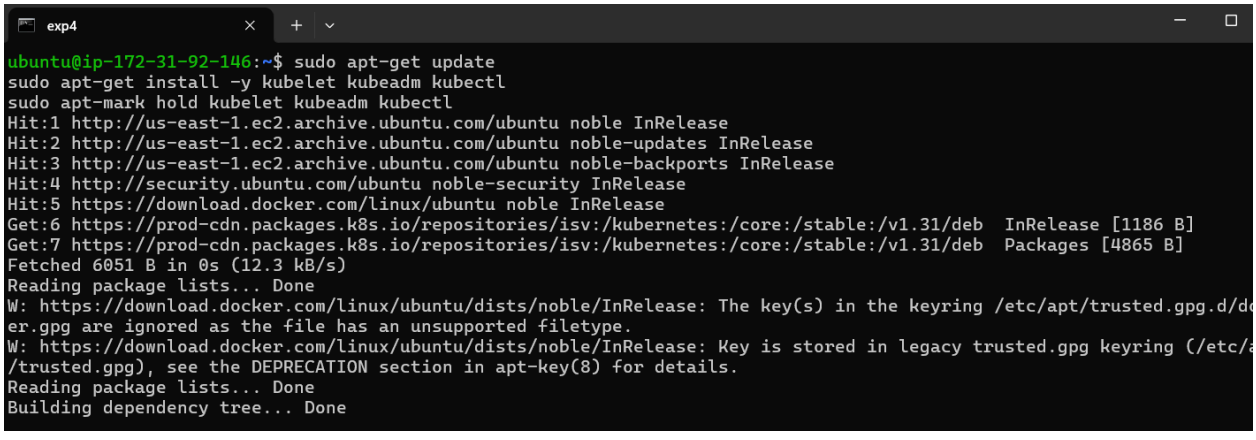
5. Set-up Kubernetes

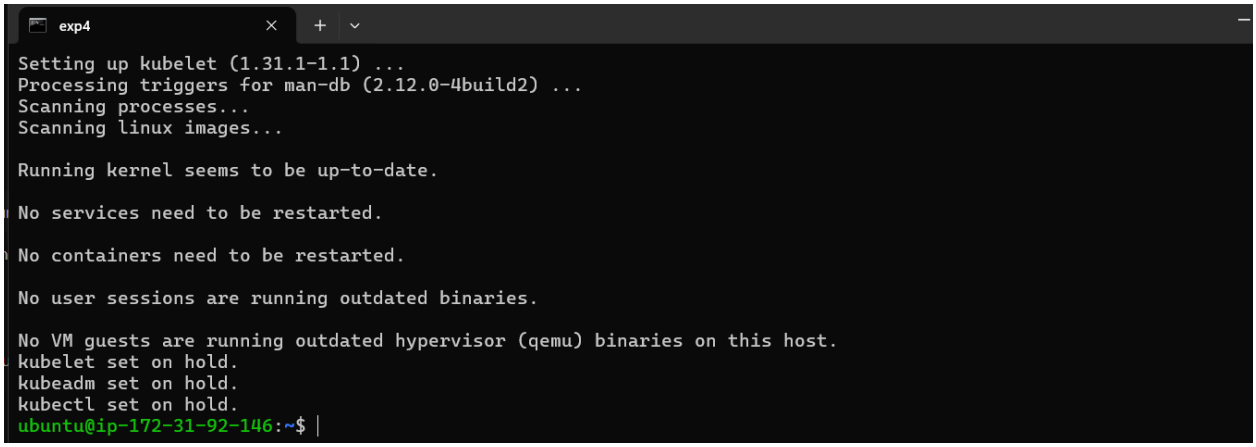
1. Add the Kubernetes signing key and repository to your APT sources for package installation.

```
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
```

2. Update APT package lists, install Kubernetes tools (kubelet, kubeadm, kubectl), and mark them to prevent automatic updates.

```
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
```

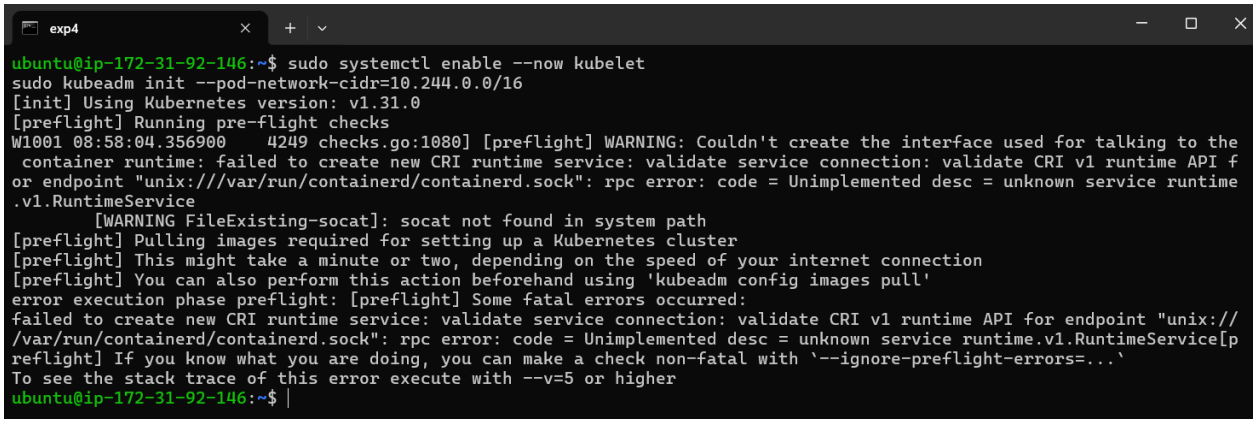




6. Initialize the kubecluster

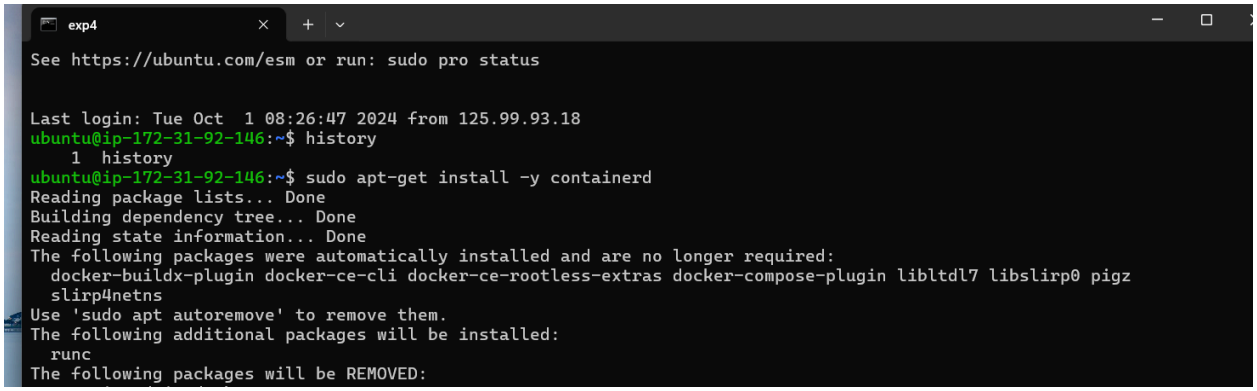
1. Enable and start the `kubelet` service, then initialize the Kubernetes cluster with a specified pod network CIDR.

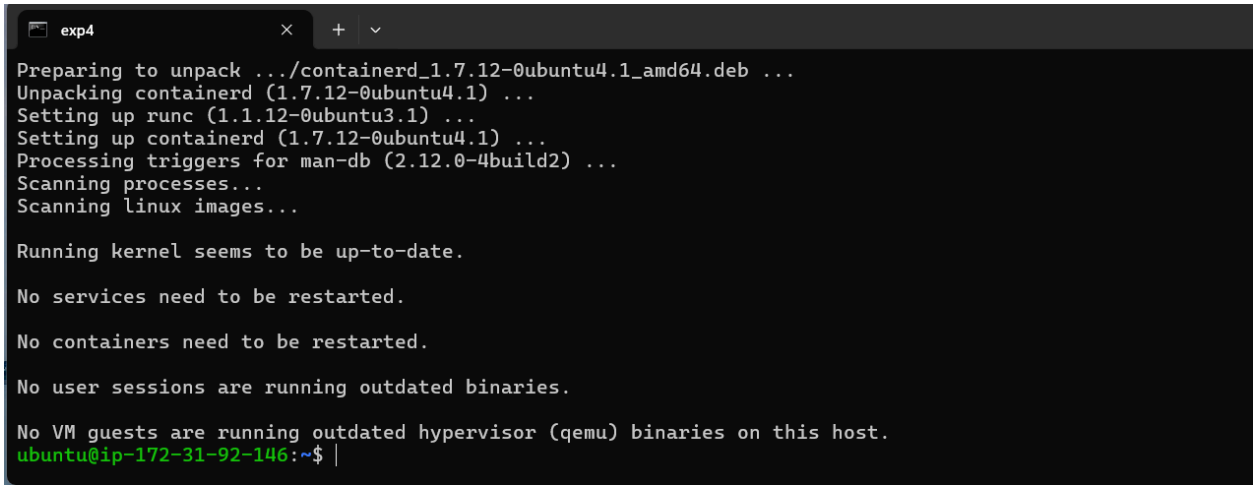
```
sudo systemctl enable --now kubelet
sudo kubeadm init --pod-network-cidr=10.244.0.0/16
```



Here, we encounter an error as a few of the dependencies for running the command are not installed. So, run the following commands

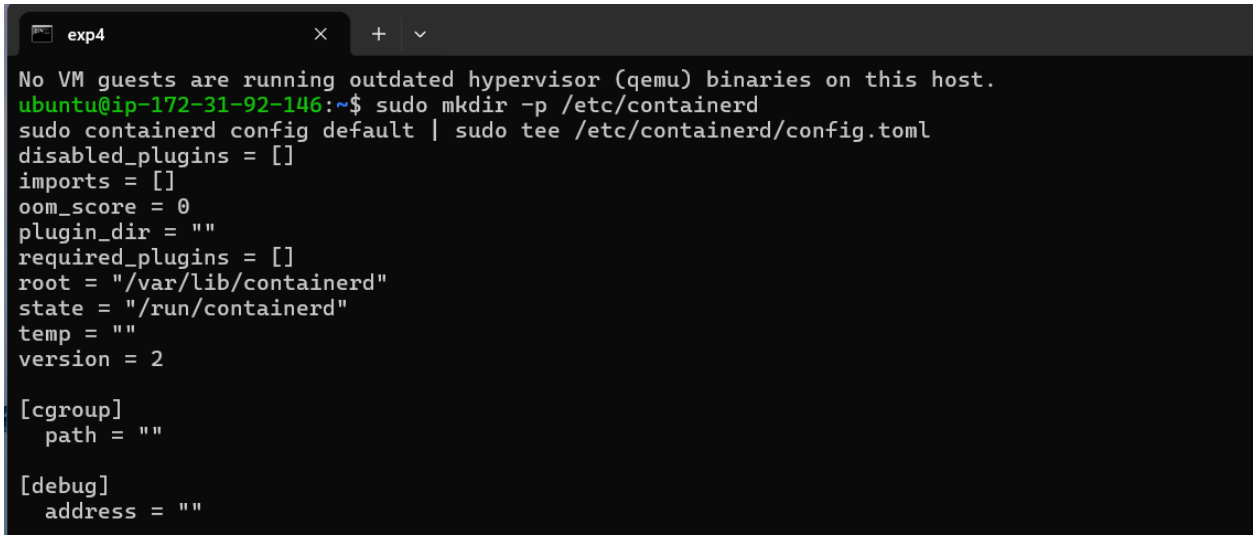
```
sudo apt-get install -y containerd
```





2. Create the /etc/containerd directory and generate the default containerd configuration file (config.toml).

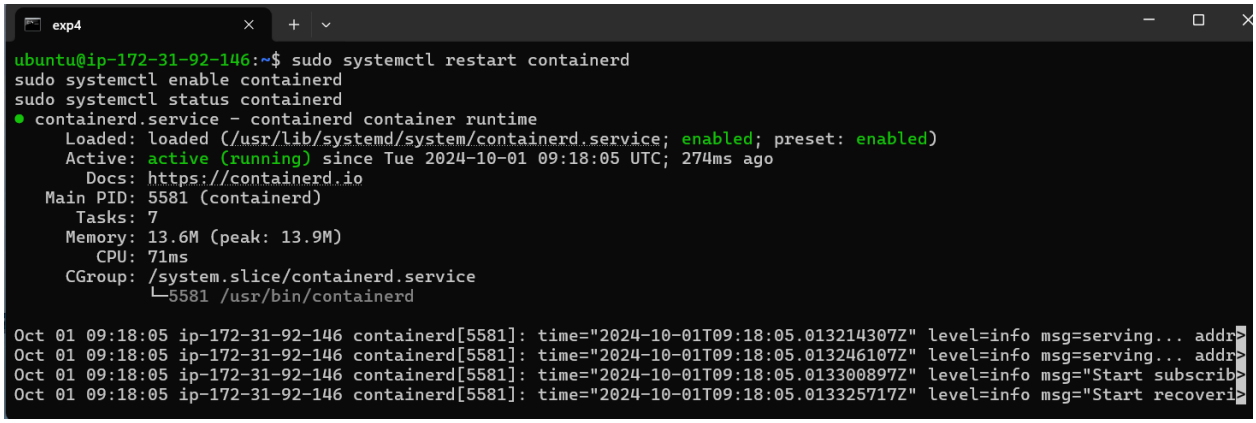
```
sudo mkdir -p /etc/containerd
sudo containerd config default | sudo tee /etc/containerd/config.toml
```



```
[timeouts]
  "io.containerd.timeout.bolt.open" = "0s"
  "io.containerd.timeout.metrics.shimstats" = "2s"
  "io.containerd.timeout.shim.cleanup" = "5s"
  "io.containerd.timeout.shim.load" = "5s"
  "io.containerd.timeout.shim.shutdown" = "3s"
  "io.containerd.timeout.task.state" = "2s"

[ttrpc]
  address = ""
  gid = 0
  uid = 0
ubuntu@ip-172-31-92-146:~$
```

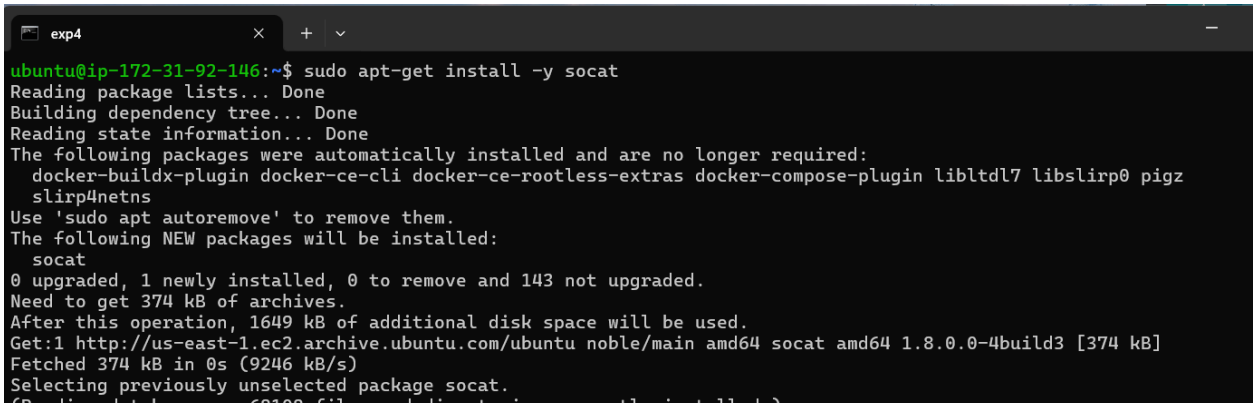
3. Restart, enable, and check the status of the containerd service.

```
sudo systemctl restart containerd
sudo systemctl enable containerd
sudo systemctl status containerd
```



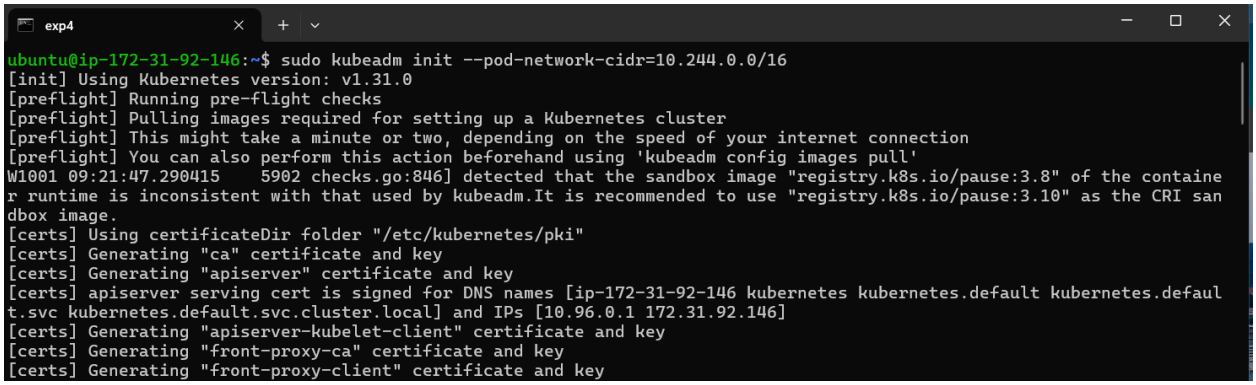
4. Install the socat package using APT with no prompts.

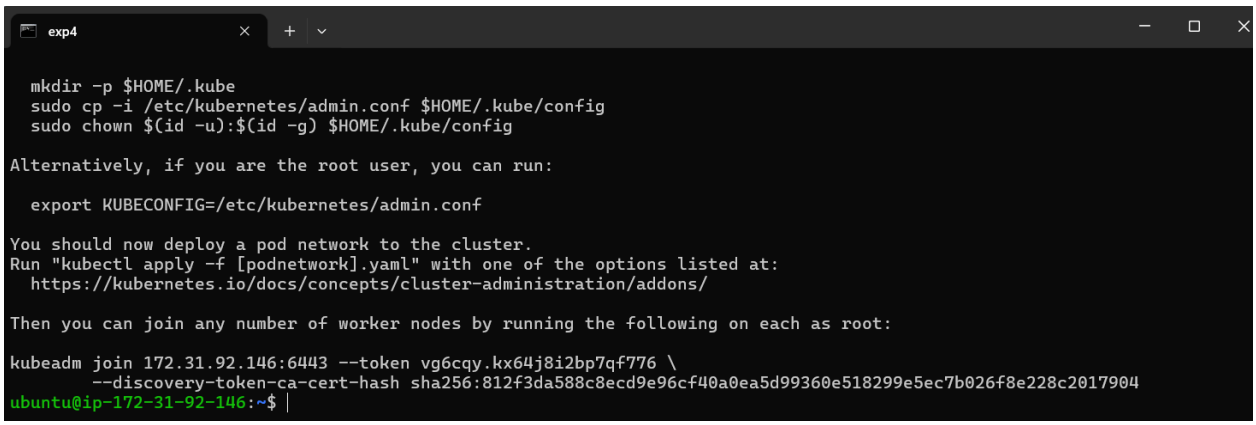
```
sudo apt-get install -y socat
```



5. Initialize the Kubernetes cluster with a specified pod network CIDR of 10.244.0.0/16.

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

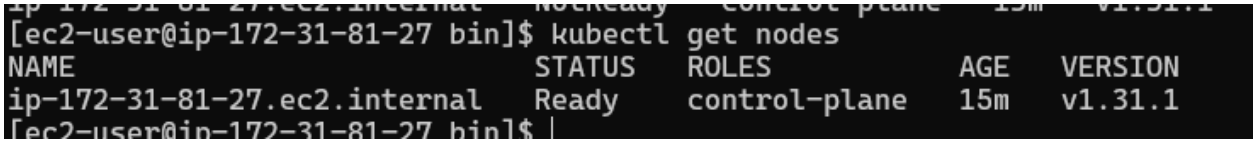




6. Deploy the Flannel network add-on for Kubernetes by applying the specified YAML configuration file from the provided URL.

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

7. Connect nginx server to pod.



```
[ec2-user@ip-172-31-81-27 bin]$ kubectl apply -f https://k8s.io/examples/application/deployment.yaml
deployment.apps/nginx-deployment created
[ec2-user@ip-172-31-81-27 bin]$
```

```
[ec2-user@ip-172-31-81-27 bin]$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-4prm9    0/1     Pending   0          94s
nginx-deployment-d556bf558-d6dld    0/1     Pending   0          94s
[ec2-user@ip-172-31-81-27 bin]$
```

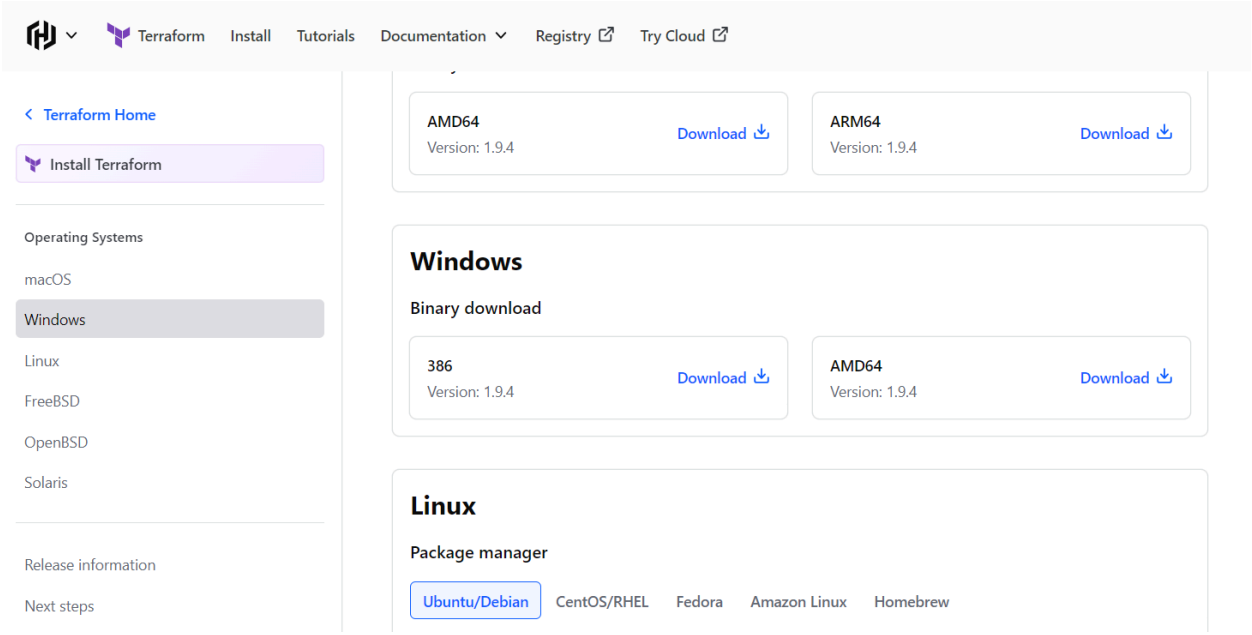
Conclusion:

In this experiment, we have learned how to deploy an nginx server to a kubernetes cluster. We also learned how to tackle any intolerable taints that tend to give issues while deploying the server. We also learned how to set the port on which you want to host the server.

Installation guide of Teraform (Windows).

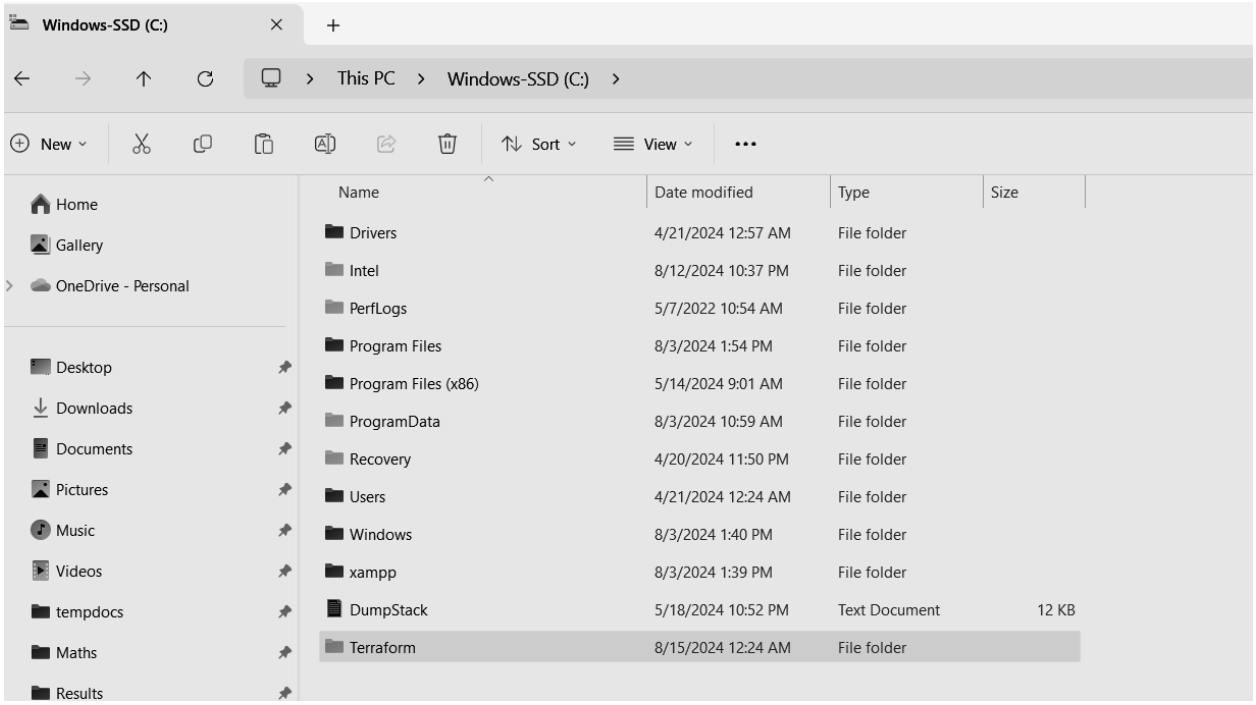
Step 1: Download terraform

To install Terraform, First Download the Terraform Cli Utility for windows from terraforms official website
website:https://www.terraform.io/downloads.html
Select the Operating System Windows followed by either 32bit (386) or 64 bit (AMD64) based on your OS type.



Step 2: Create a folder for Terraform:

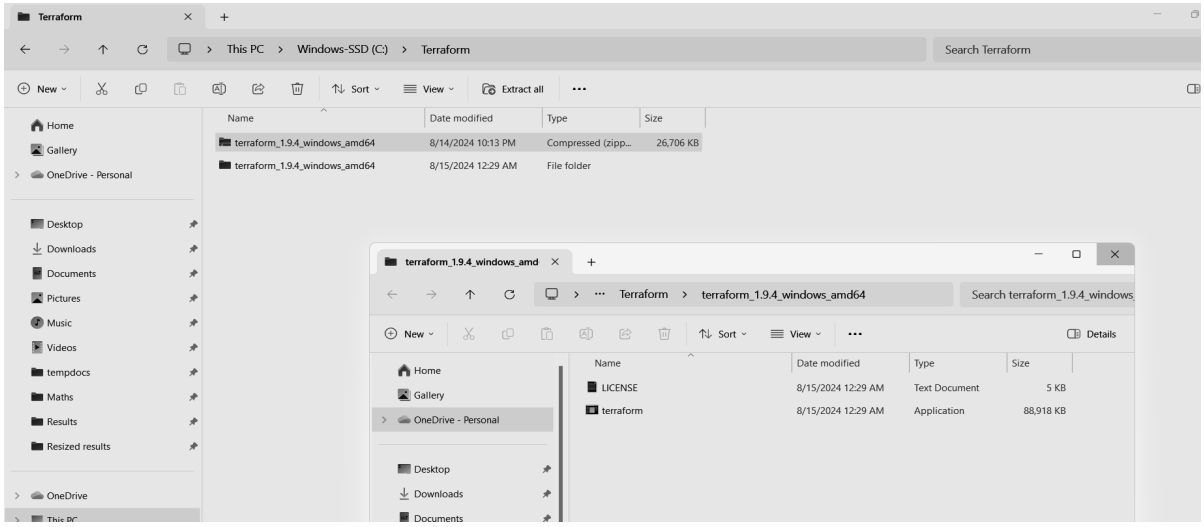
- Open **File Explorer** and navigate to `C:\` or another location where you want to store Terraform.
- Right-click in the folder, select **New > Folder**, and name the folder something like `Terraform`.
- For example, the folder might be `C:\Terraform`.



Step 3: Extract Terraform into this folder:

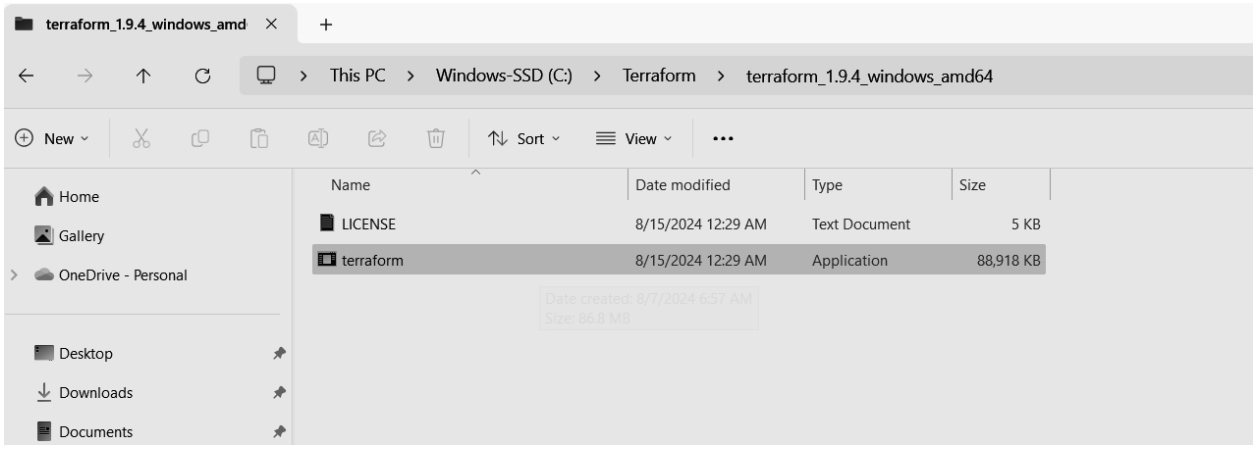
- Make sure to move the zip file from downloads to this new folder if it is not there already.
- After downloading the `.zip` file, right-click the `.zip` and choose **Extract All**.
- Extract the contents of the `.zip` file into the folder you just created (e.g., `C:\Terraform`).

*the background is the terraform folder in the c drive where we unzip the file and the forefront image is of contents inside this unzipped folder.

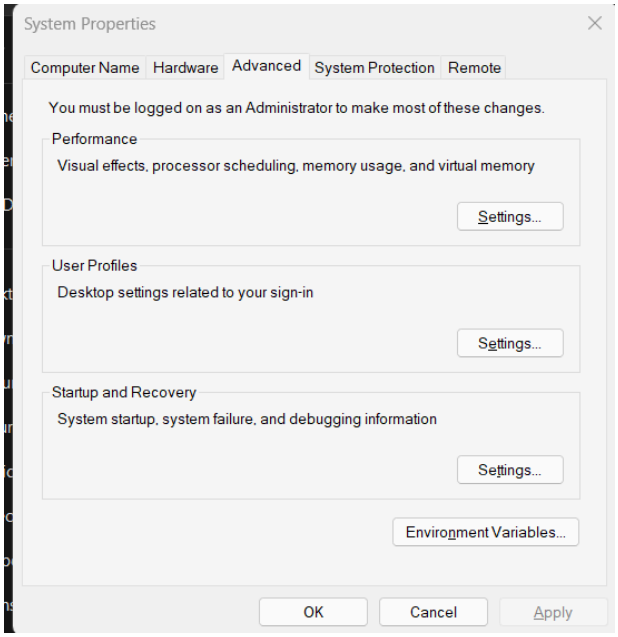


Step 4: Add to path

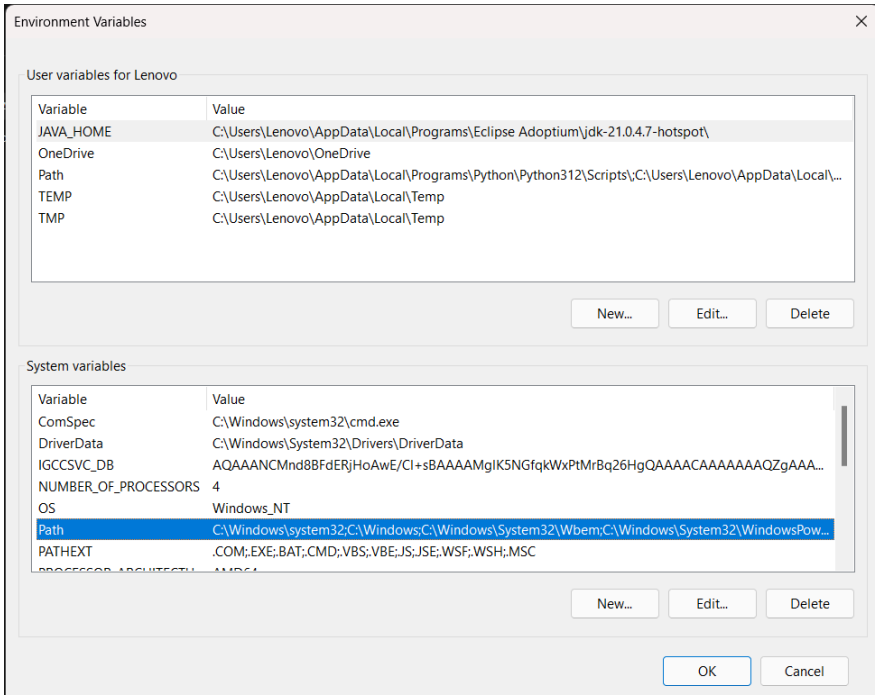
- Theory : the system uses top level search for finding the executable files, so rule of thumb is to add the directory which is a direct parent to the terraform exe.
- Hence, we first locate the terraform.exe and only copy the path till its parent directory. the highlighted file is the executable file and we will just copy that terraform_1.9.4_windows_amd64 visible in the path above.



- Now search for edit environment variables and click on it.
- Now click on environment variables.

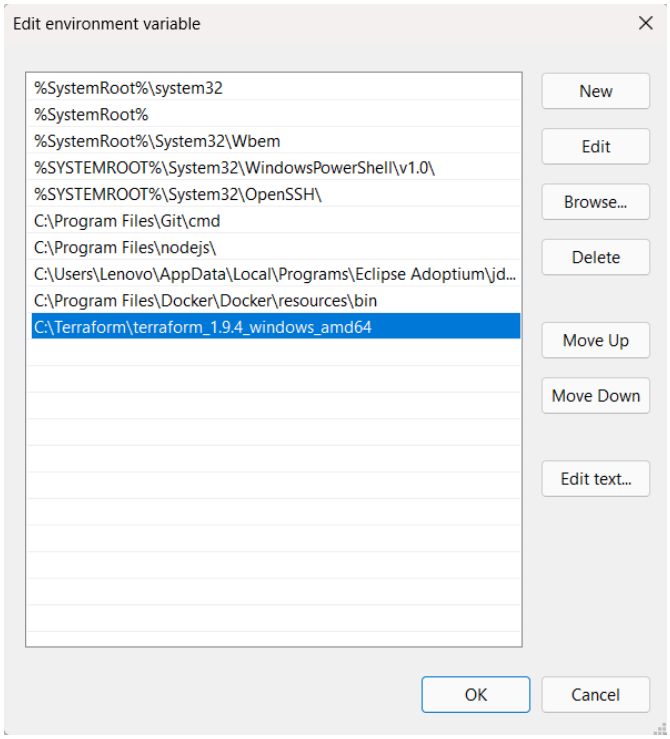


- Now select path from the system variables and click on edit.

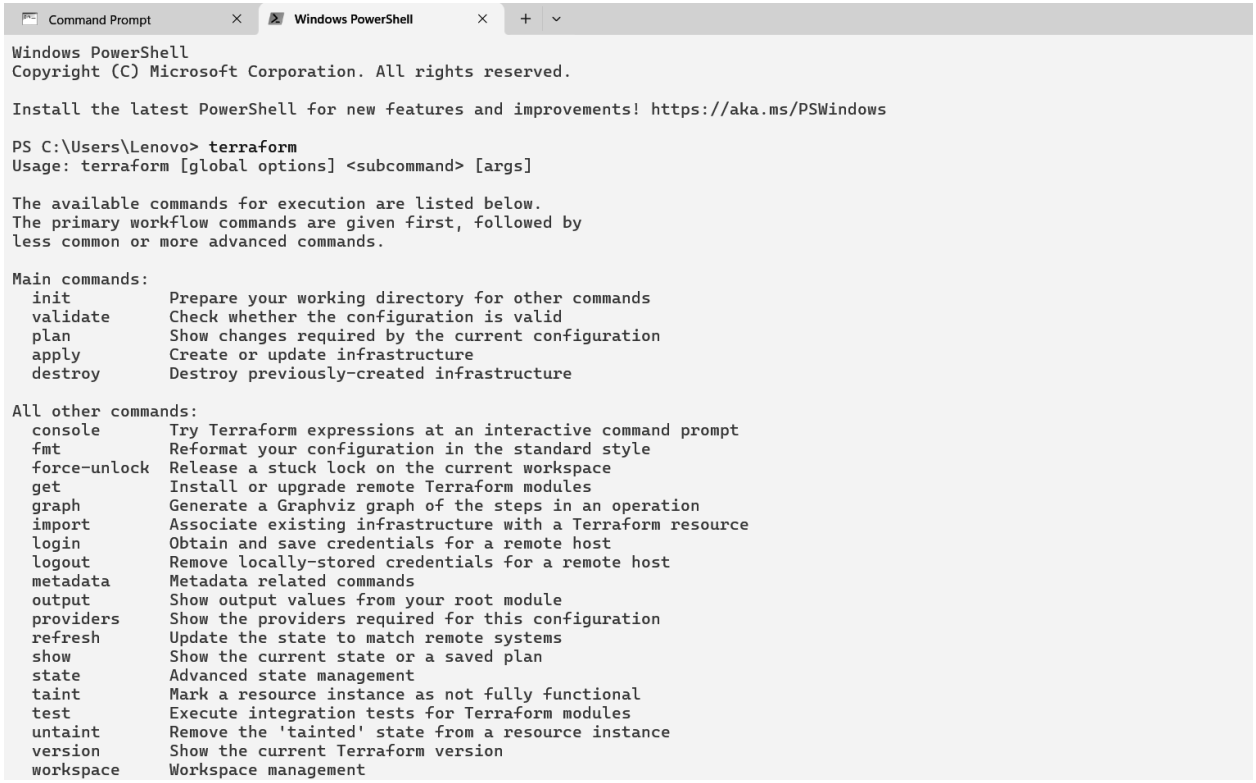


*Note click on edit and not new!

- Now paste the path to the directory that we copied.



Step 5: Run terraform command in shell to see if it works



Name: Shubham Jha
Div: D15C

Roll NO:19

Experiment No.: 6

Implementation:

A. Creating docker image using terraform

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

Step 1: Check the docker functionality

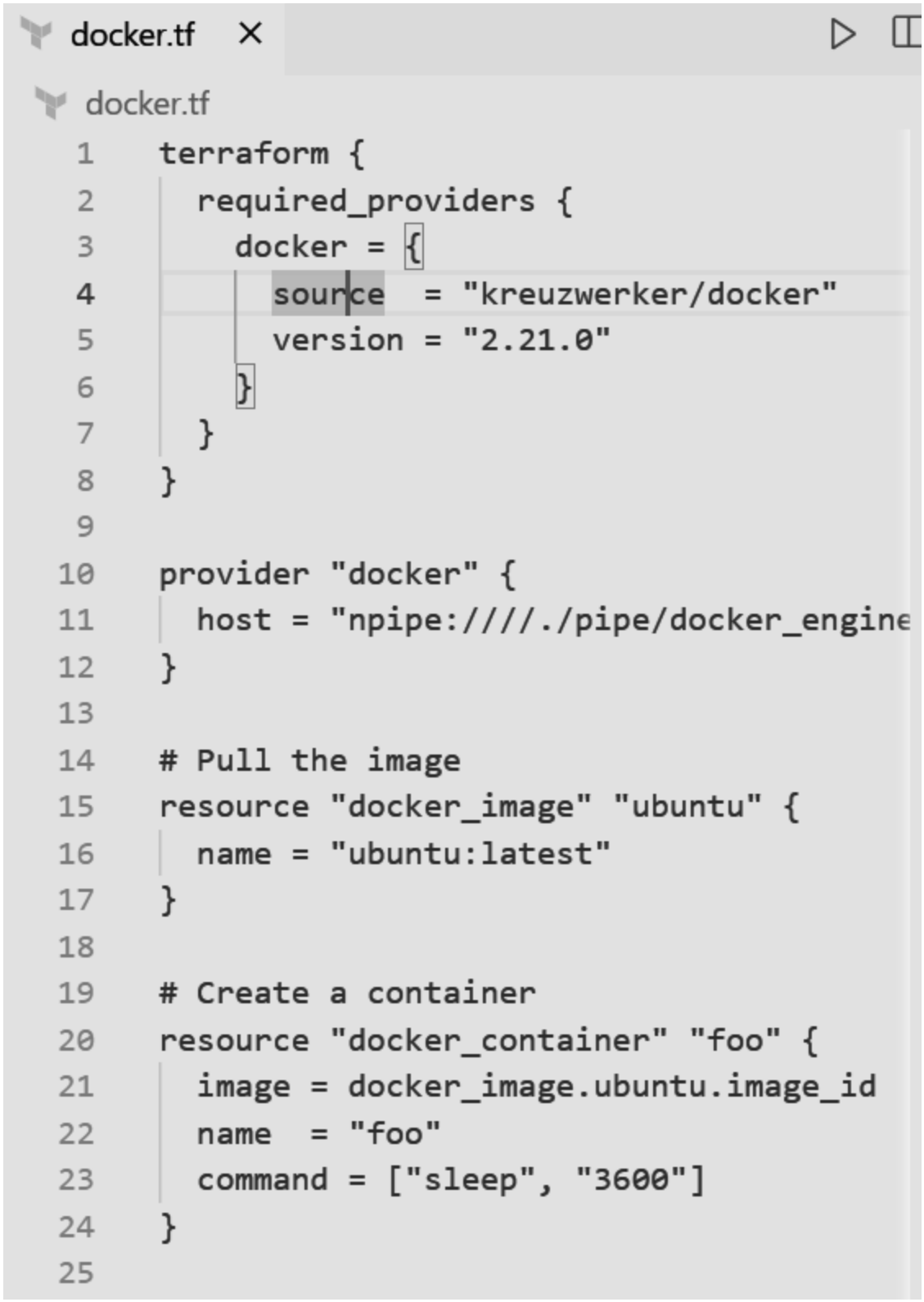
```
PS C:\Users\91773> docker

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

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

Management Commands:
  builder     Manage builds
  buildx*     Docker Buildx
  compose*    Docker Compose
  container   Manage containers
  context     Manage contexts
  debug*      Get a shell into any image or container
  desktop*    Docker Desktop commands (Alpha)
  dev*        Docker Dev Environments
```

```
PS C:\Users\91773> docker --version
Docker version 27.0.3, build 7d4bcd8
PS C:\Users\91773>
```

Now, create a folder named 'Terraform Scripts' in which we save our different types of scripts which will be further used in this experiment.
Step 2: Firstly create a new folder named 'Docker' in the 'TerraformScripts' folder. Then create a new docker.tf file using Atom editor and write the

followingcontents into it to create a Ubuntu Linux container.
Script:

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

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

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

# Create a container
resource "docker_container" "foo" {
  image = docker_image.ubuntu.image_id
  name  = "foo"
  command = ["sleep", "3600"]
}
```


docker.tf
docker.tf
terraform {
required_providers {
docker = {
source = "kreuzwerker/docker"
version = "2.21.0"
}
}
}
provider "docker" {
host = "npipe:////./pipe/docker_engine
}
# Pull the image
resource "docker_image" "ubuntu" {
name = "ubuntu:latest"
}
# Create a container
resource "docker_container" "foo" {
image = docker_image.ubuntu.image_id
name = "foo"
command = ["sleep", "3600"]
}

Step 3: Execute Terraform Init command to initialize the resources

```
PS C:\Users\91773\Desktop\College Resources\TerraformScripts\Docker> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding kreuzwerker/docker versions matching "2.21.0"...
- Installing kreuzwerker/docker v2.21.0...
- Installed kreuzwerker/docker v2.21.0 (self-signed, key ID BD080C4571C6104C)
Partner and community providers are signed by their developers.
If you'd like to know more about provider signing, you can read about it here:
https://www.terraform.io/docs/cli/plugins/signing.html
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

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

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

Step 4: Execute Terraform plan to see the available resources

```
PS C:\Users\91773\Desktop\College Resources\TerraformScripts\Docker> terraform plan

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

Terraform will perform the following actions:

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

```
      + 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.
```

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

```
PS C:\Users\91773\Desktop\College Resources\TerraformScripts\Docker> terraform apply
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubun
tu:latest]

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

Terraform will perform the following actions:

  # docker_container.foo will be created
  + resource "docker_container" "foo" {
      + attach           = false
      + bridge           = (known after apply)
      + command          = [
          + "sleep",
          + "3600",
        ]
```

Docker images, Before Executing Apply step:

```
PS C:\Users\91773\Desktop\College Resources\TerraformScripts\Docker> docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
```

Docker images, After Executing Apply step:

```
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\91773\Desktop\College Resources\TerraformScripts\Docker> docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
ubuntu       latest    edbfe74c41f8   2 weeks ago   78.1MB
PS C:\Users\91773\Desktop\College Resources\TerraformScripts\Docker>
```

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

```
PS C:\Users\91773\Desktop\College Resources\TerraformScripts\Docker> terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Refreshing state... [id=f03a28e4658896c23c9992f7a98eb1011befc7d014e997ea9fc6372da70b7903]

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

Terraform will perform the following actions:

  # docker_container.foo will be destroyed
  - resource "docker_container" "foo" {
      - attach            = false -> null
      - command           = [
          - "sleep",
          - "3600",
        ] -> null
      - cpu_shares        = 0 -> null
```

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

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

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

  Enter a value: yes

docker_container.foo: Destroying... [id=f03a28e4658896c23c9992f7a98eb1011befc7d014e997ea9fc6372da70b7903]
docker_container.foo: Destruction complete after 0s
docker_image.ubuntu: Destroying... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_image.ubuntu: Destruction complete after 1s

Destroy complete! Resources: 2 destroyed.
```

Docker images After Executing Destroy step

```
PS C:\Users\91773\Desktop\College Resources\TerraformScripts\Docker> docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
```

Adv DevOps Practical 7

Aim:

To understand Static Analysis SAST process and learn to integrate Jenkins SAST to

SonarQube/GitLab.

Integrating Jenkins with SonarQube:

Prerequisites:

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

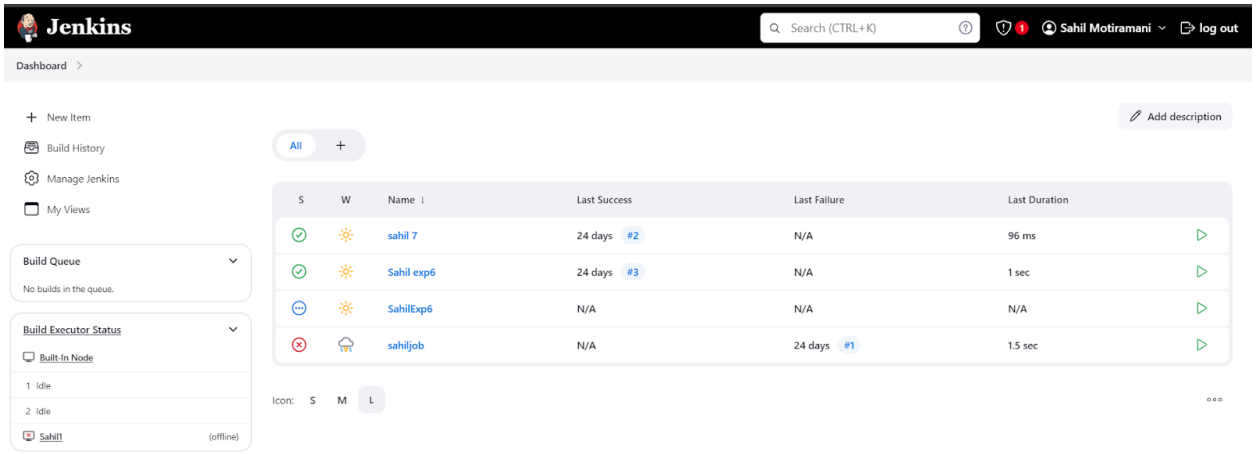
Steps

to integrate Jenkins with SonarQube

Prerequisites: Make sure you have docker and jenkins installed.
Run **docker -v** to check the docker installation.

Run

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



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

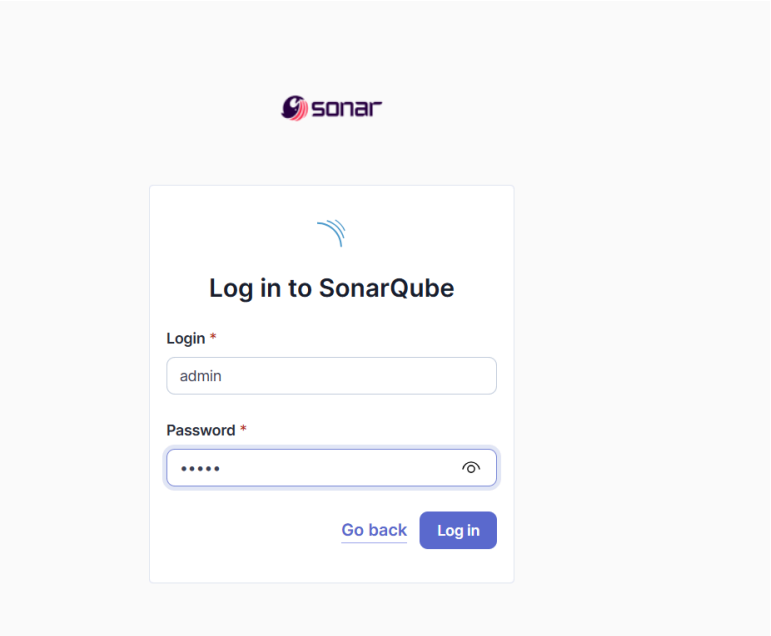
docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest

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

```
C:\Users\Lenovo>docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
47f6db8dbf2ed99dbe304bc0ebdf47b9d4144c4e4add42055ba44ce231058272
```

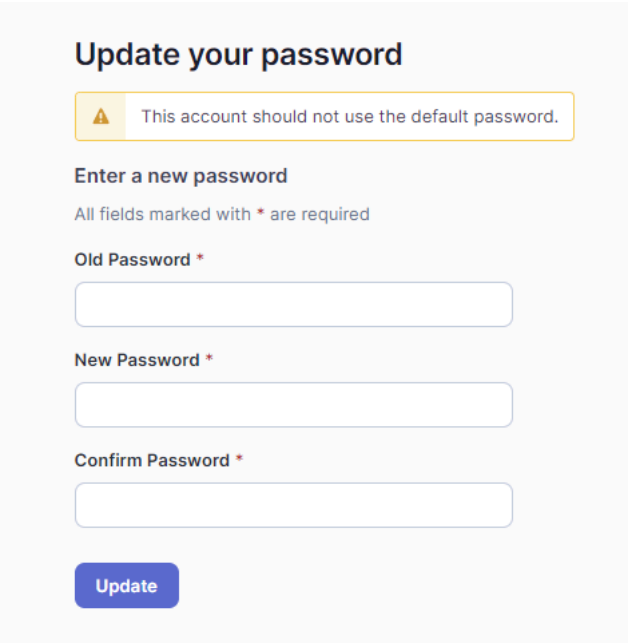
3. Once the container is up and running, you can check the status of SonarQube at

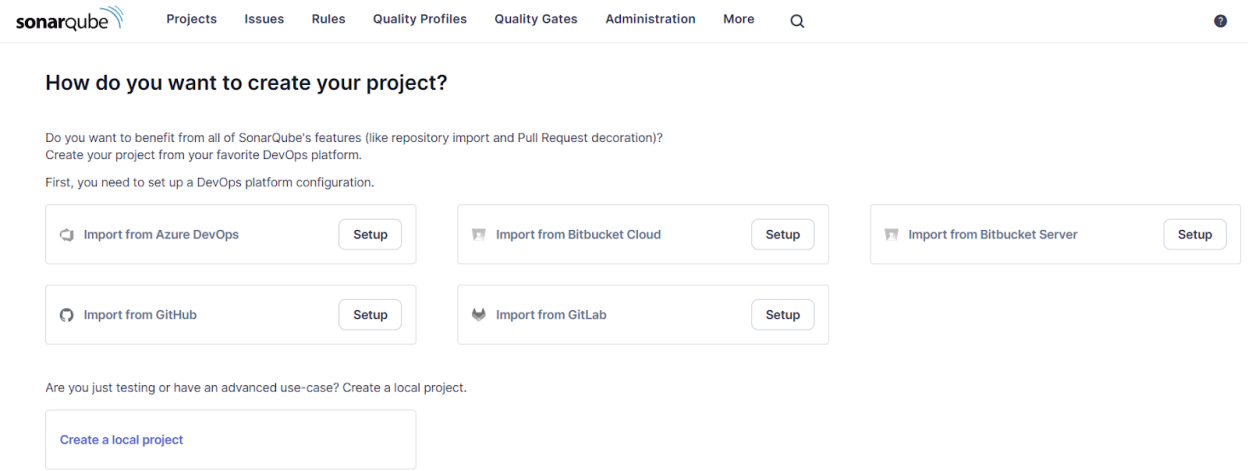
localhost port 9000.



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

(do change the password as you cannot use the default one)





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

1 of 2

Create a local project

Project display name *

exp7

Project key *

exp7

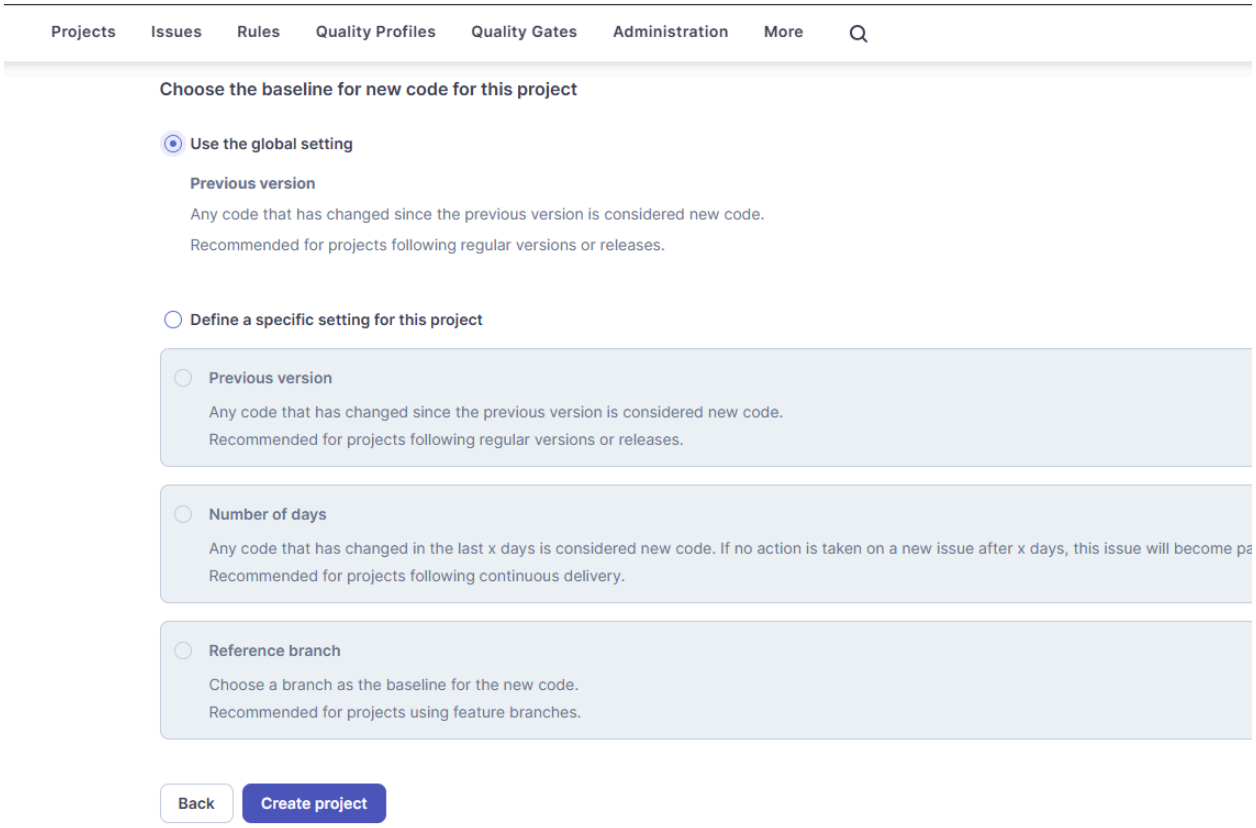
Main branch name *

main

The name of your project's default branch Learn More

Cancel Next

Setup the project



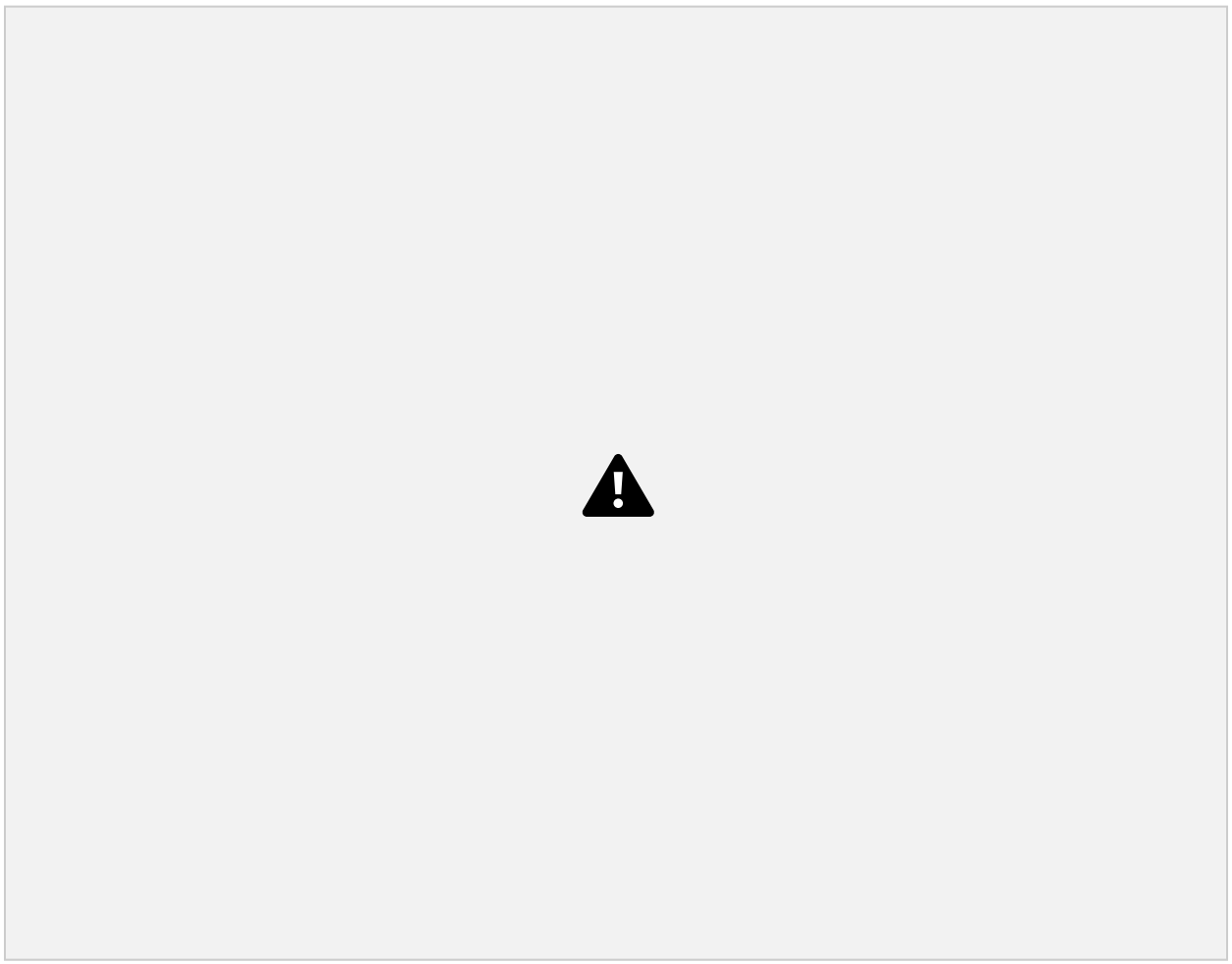
And come back to Jenkins Dashboard.
Go to Manage Jenkins and search for SonarQube Scanner for Jenkins and install it.

6. Under Jenkins ‘Manage Jenkins’ then go to ‘system’, scroll and look for **SonarQube Servers** and click on **add SonarQube** and then enter the details.

Enter the Server Authentication token if needed.(I didn’t do it)

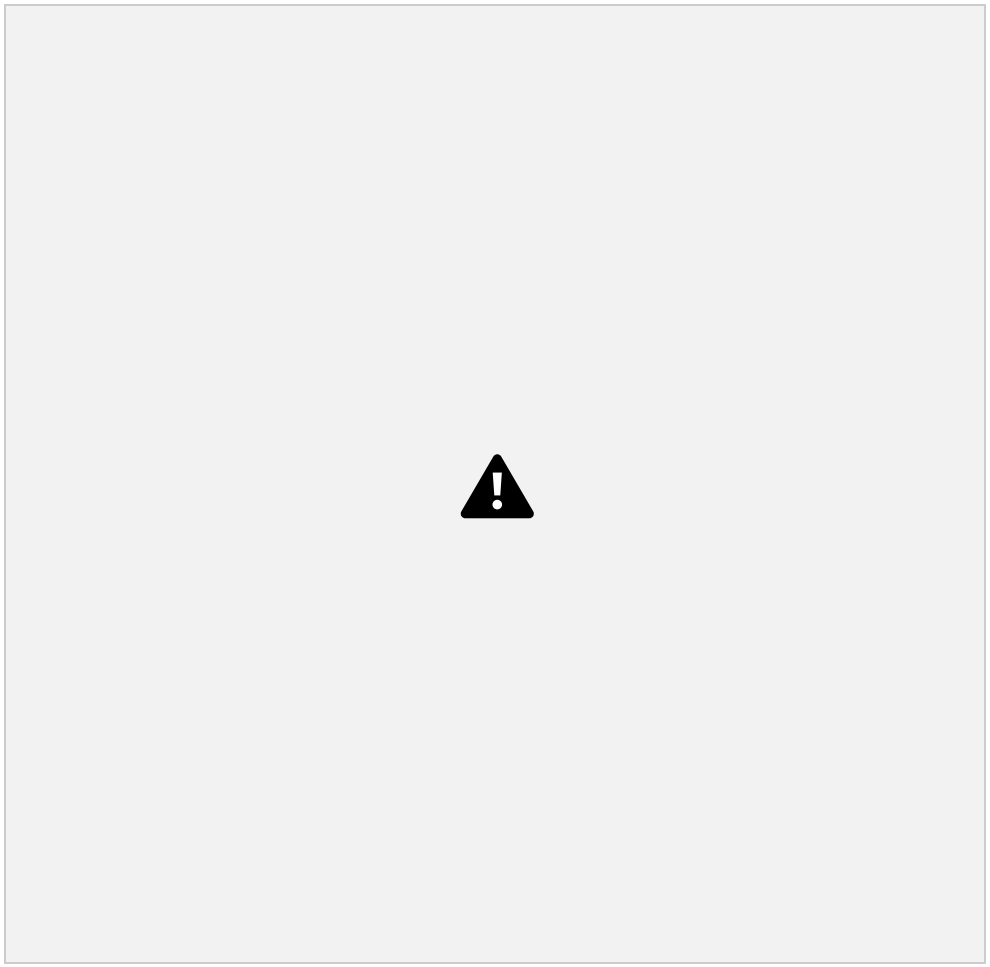
In SonarQube installations: Under **Name** add <project name of sonarqube> for me its sonarqube_exp7

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



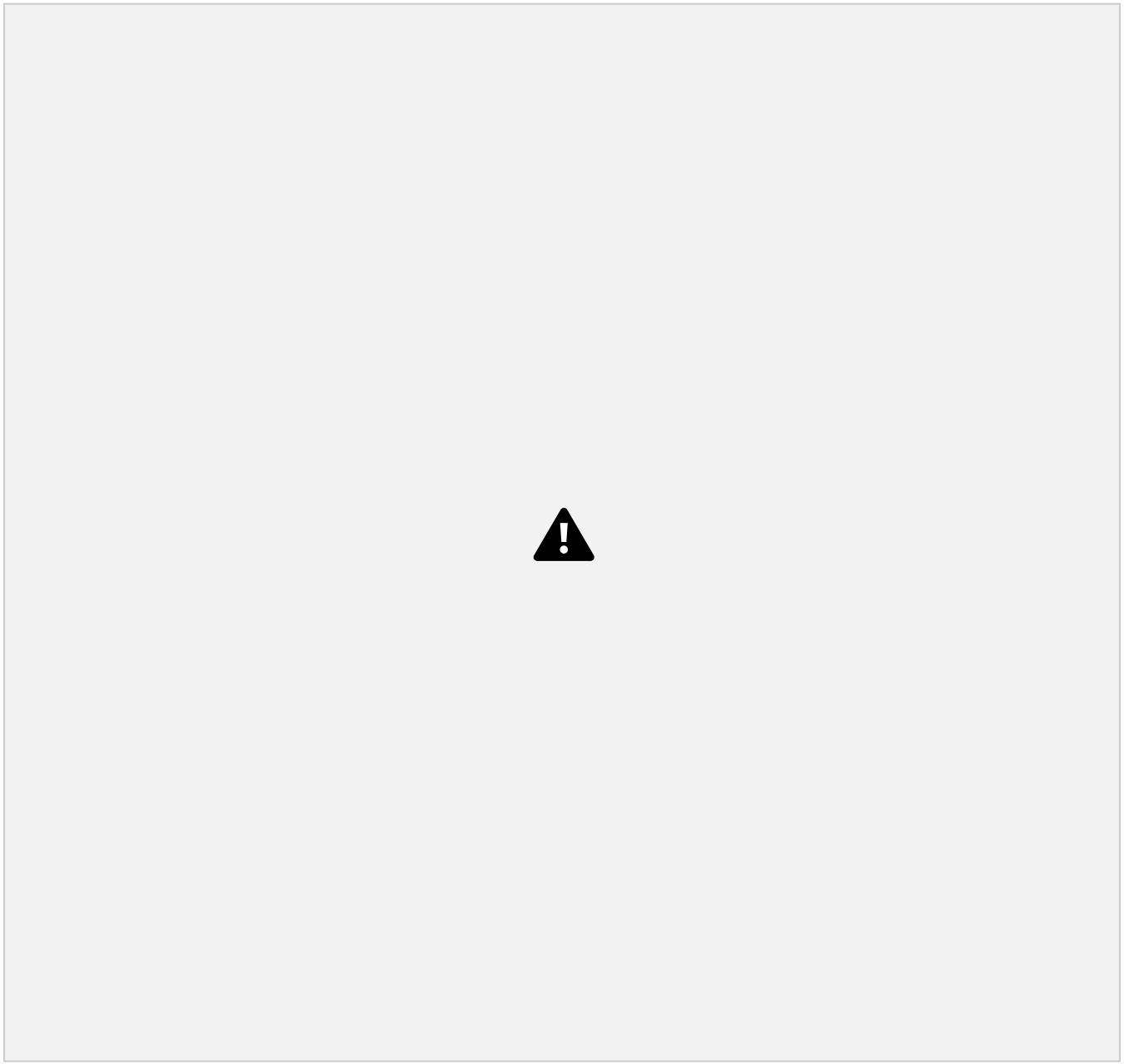
7. Search for SonarQube Scanner under Global Tool Configuration. Choose the latest configuration and choose Install automatically.

Dashboard > Manage Jenkins > Tools



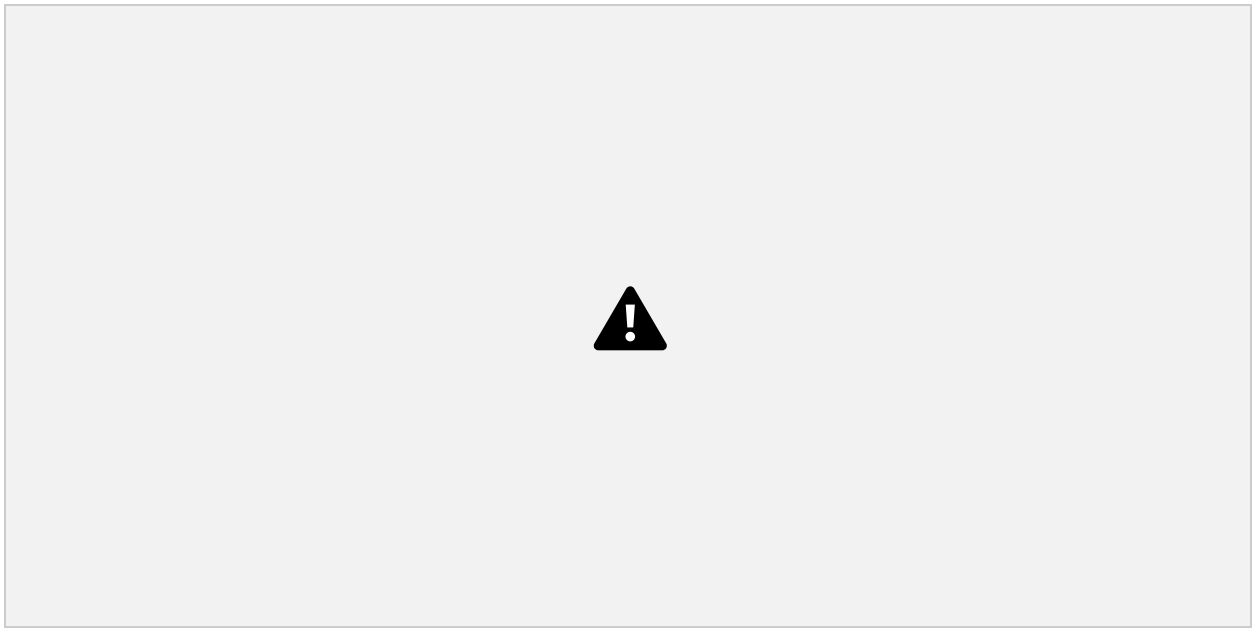
Click on **Add SonarQube Scanner** .

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

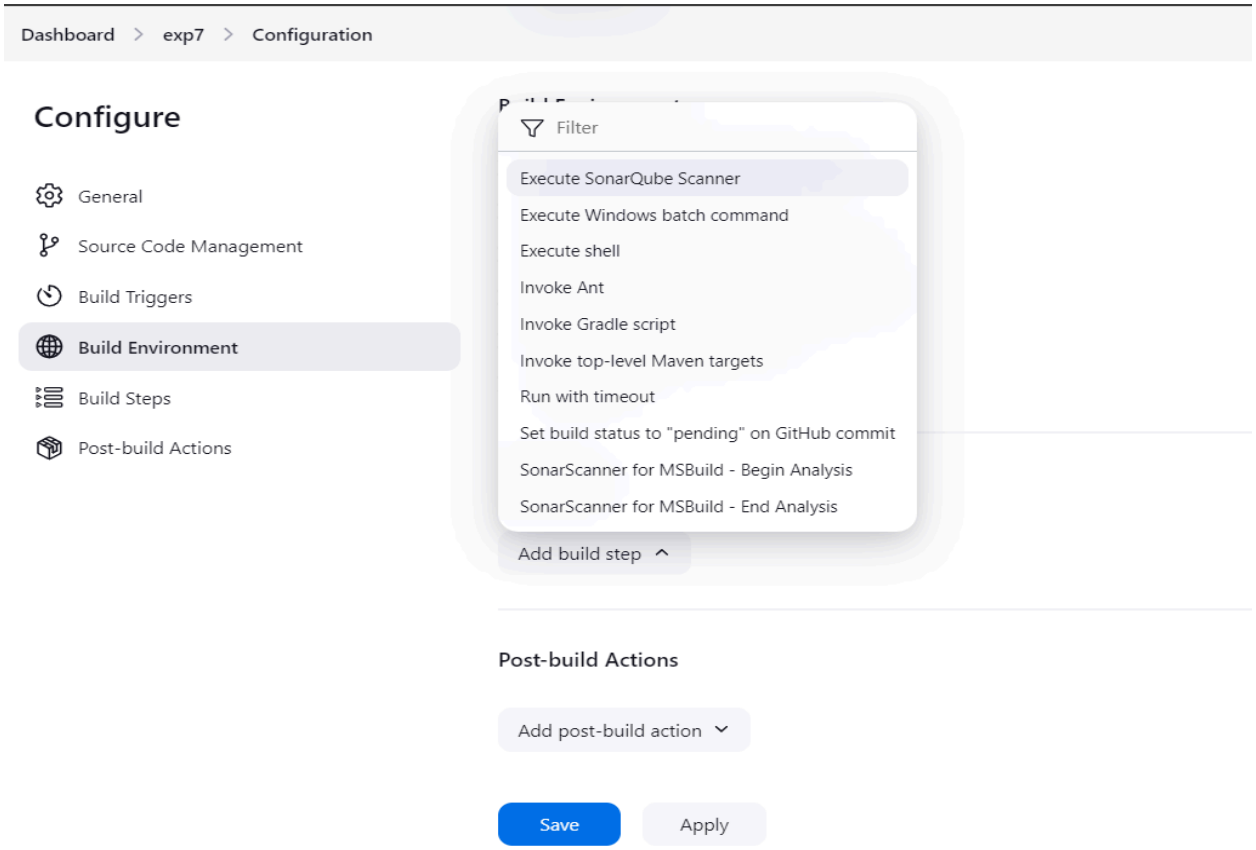


8. After the configuration, create a New Item in Jenkins, choose a freestyle project.

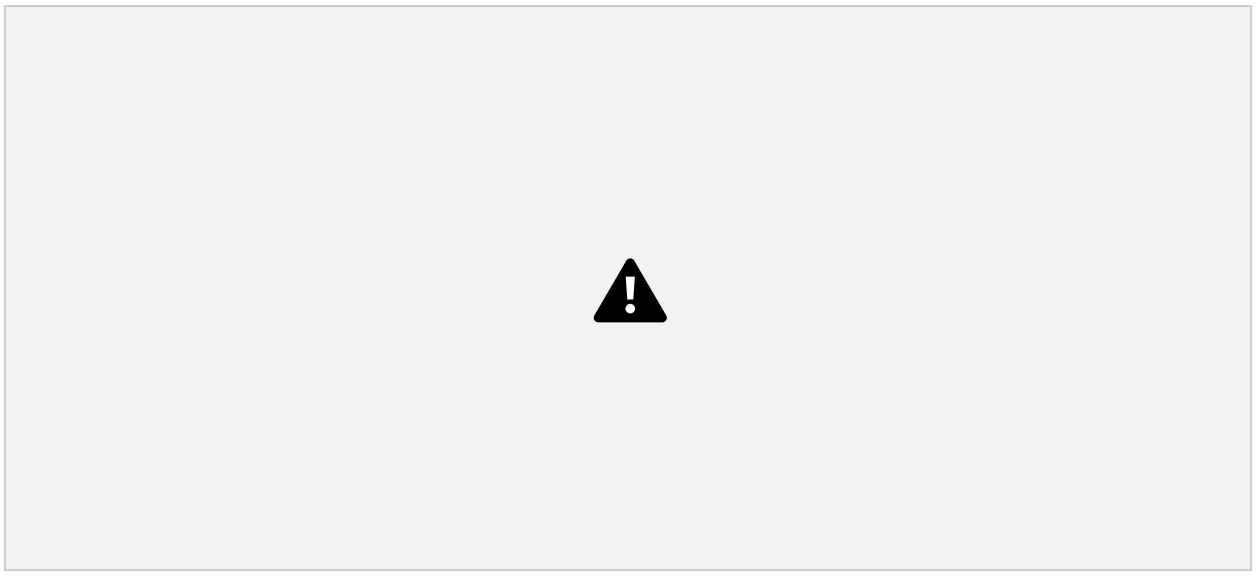
9. Choose this GitHub repository in Source Code Management.
https://github.com/shazforiot/MSBuild_firstproject
It is a sample hello-world project with no vulnerabilities and issues, just to test the integration.



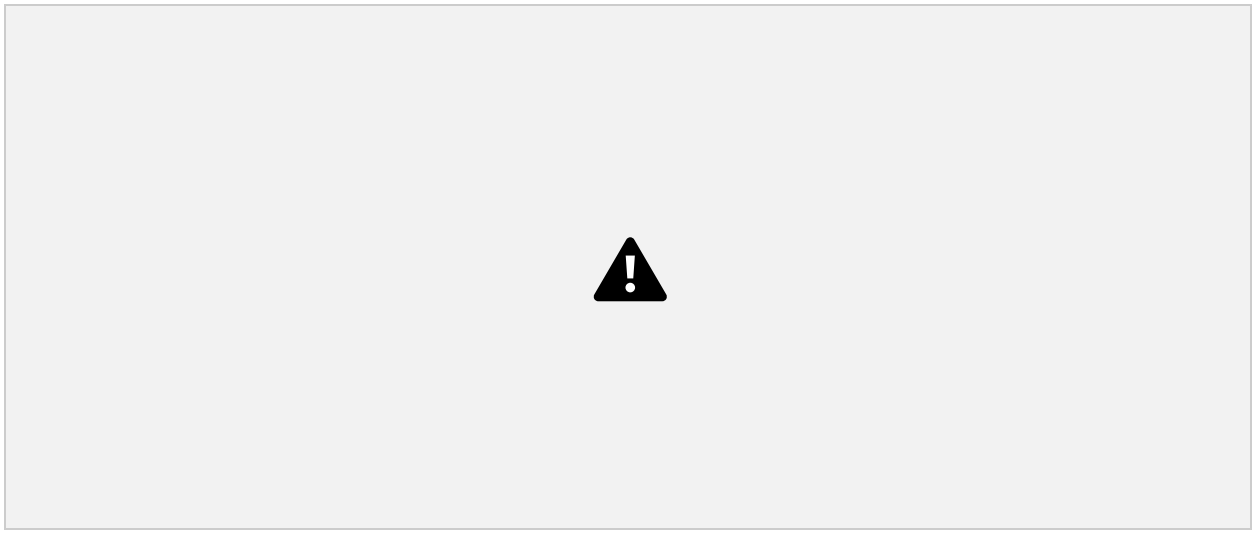
10. Under **Select project** → **Configuration** → **Build steps** → **Execute SonarQube Scanner**, enter these Analysis properties. Mention the SonarQube Project Key, Login, Password, Source path and Host URL.



You will see something like this:



Open sonarQube again and go to Project Information appearing in the right side. Click on it and you can copy the project key from About the Project Section.



Use this key in place of <projectKey> in the following code

sonar.projectKey =<projectKey>

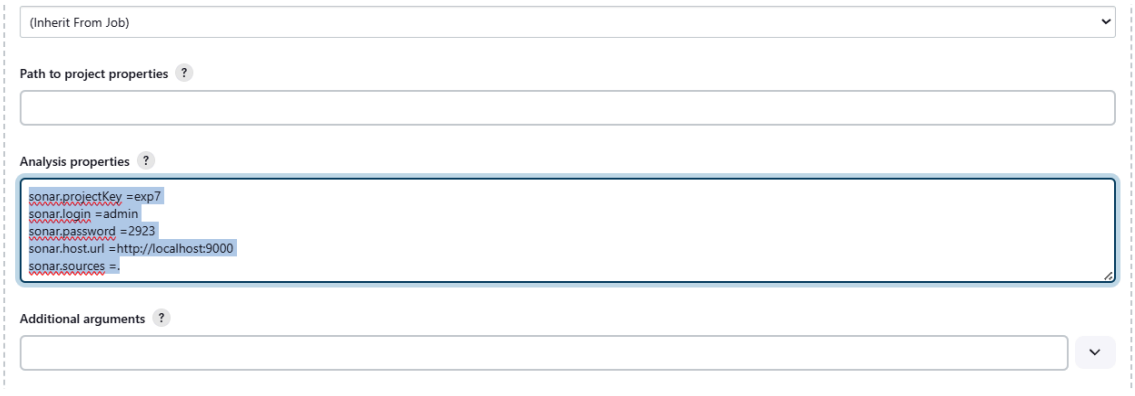
sonar.login =admin

sonar.password =<yourpassword for sonar qube>

sonar.host.url =http://localhost:9000

sonar.sources =.

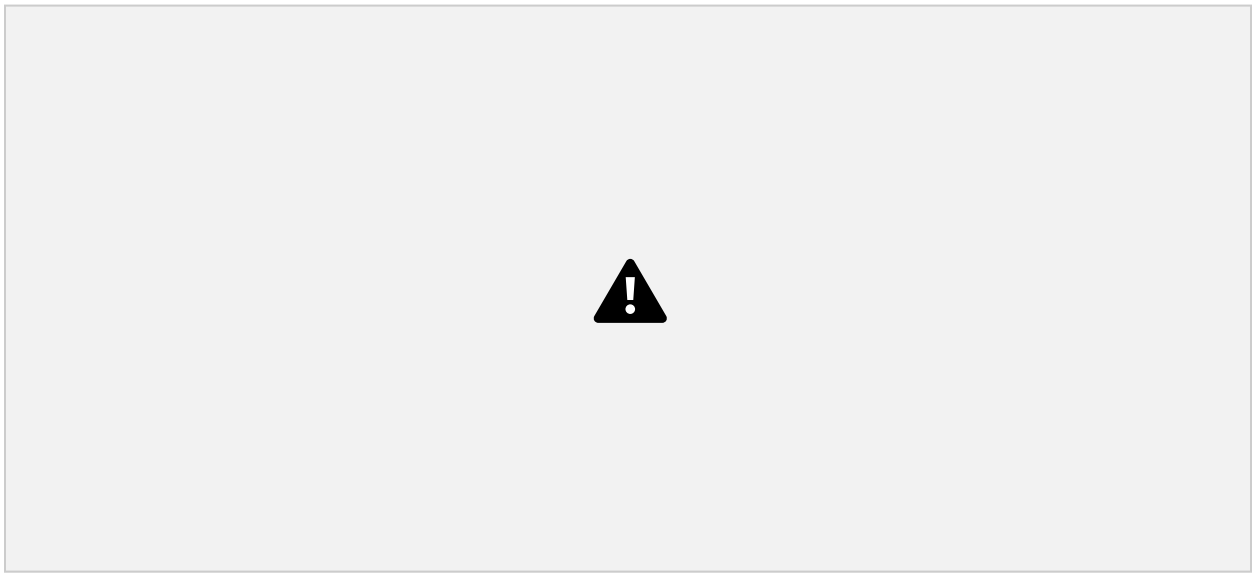
I personally preferred not keeping any spaces after '=' .



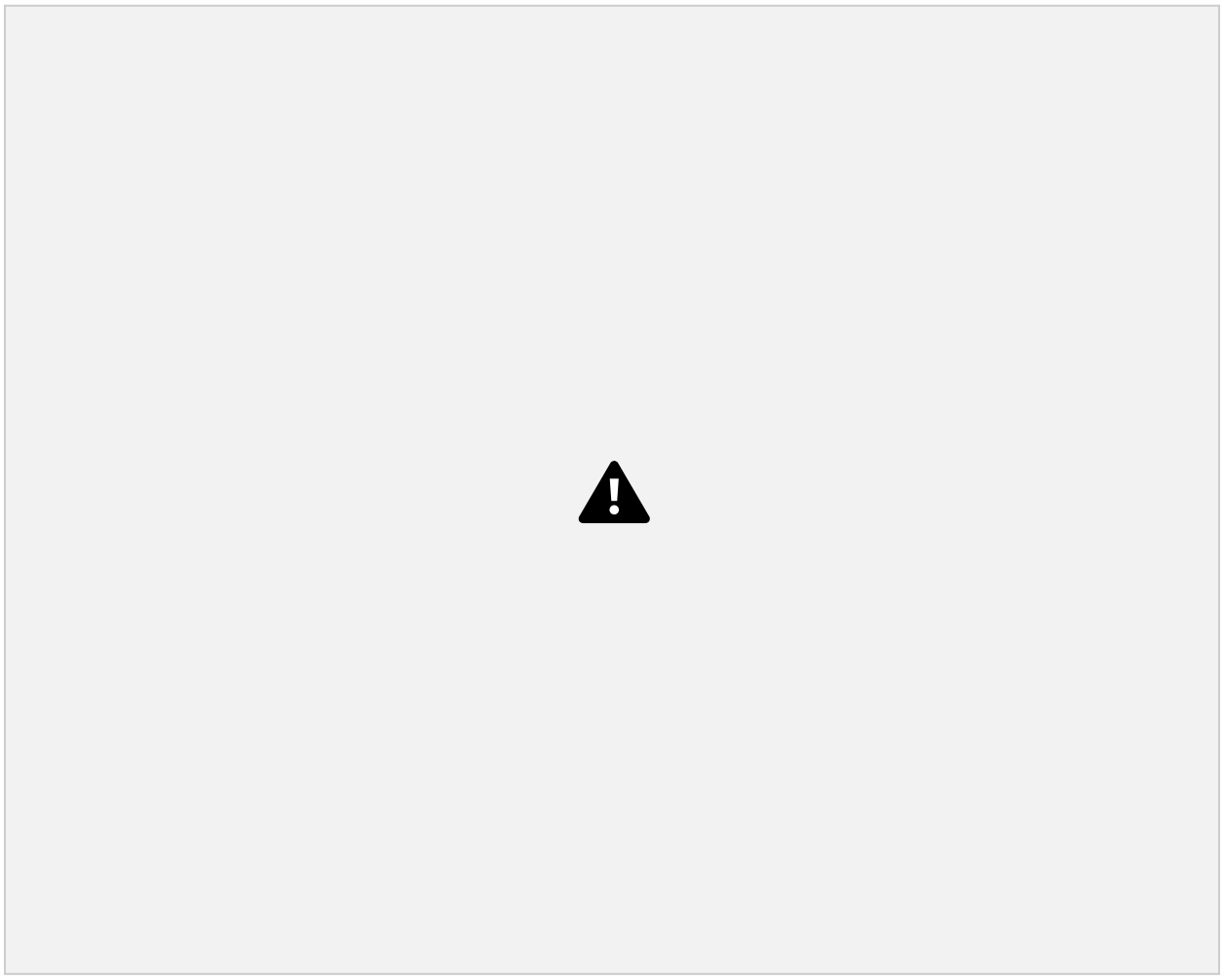
Apply and save.

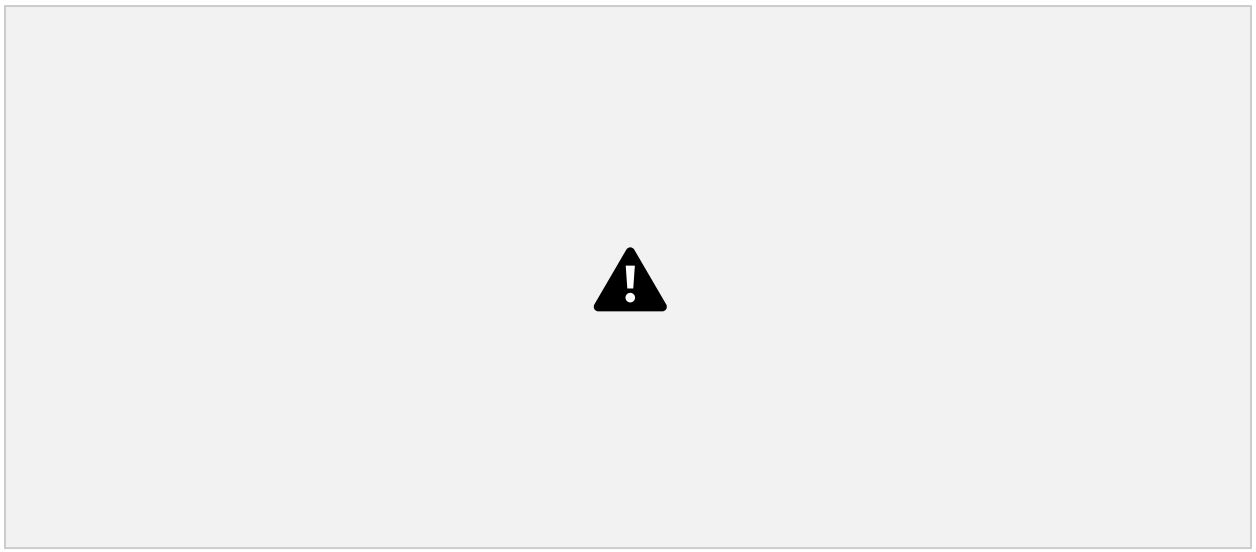
11. Go to sonarQube and go to administration → Security (dropdown) → Global Permissions.

See the administrator below and check the boxes which i checked.



12. Go to jenkins and click build:





Conclusion:

In this project, I successfully integrated Jenkins with SonarQube to establish a robust automated static application security testing (SAST) pipeline. The setup involved deploying SonarQube using Docker, ensuring smooth container orchestration and efficient resource management. A key component was configuring Jenkins with the appropriate SonarQube plugins, authentication mechanisms, and linking it to a GitHub repository for continuous integration.

One of the challenges I faced was configuring Docker on the Jenkins environment, which required resolving networking issues between the Docker containers and ensuring that the SonarQube server was reachable from Jenkins. Additionally, setting up secure authentication between Jenkins and SonarQube involved troubleshooting token-based authentication and resolving environment path issues, particularly with the **`JAVA_HOME`** setup for the SonarQube scanner.

After overcoming these obstacles, I integrated the SonarQube scanner as a build step, allowing for continuous code analysis. This setup provided automated detection of code vulnerabilities, code smells, and quality issues. It helped ensure that any new commits triggered immediate analysis, generating detailed reports and promoting continuous improvement in code quality.

08 Advanced DevOps Lab

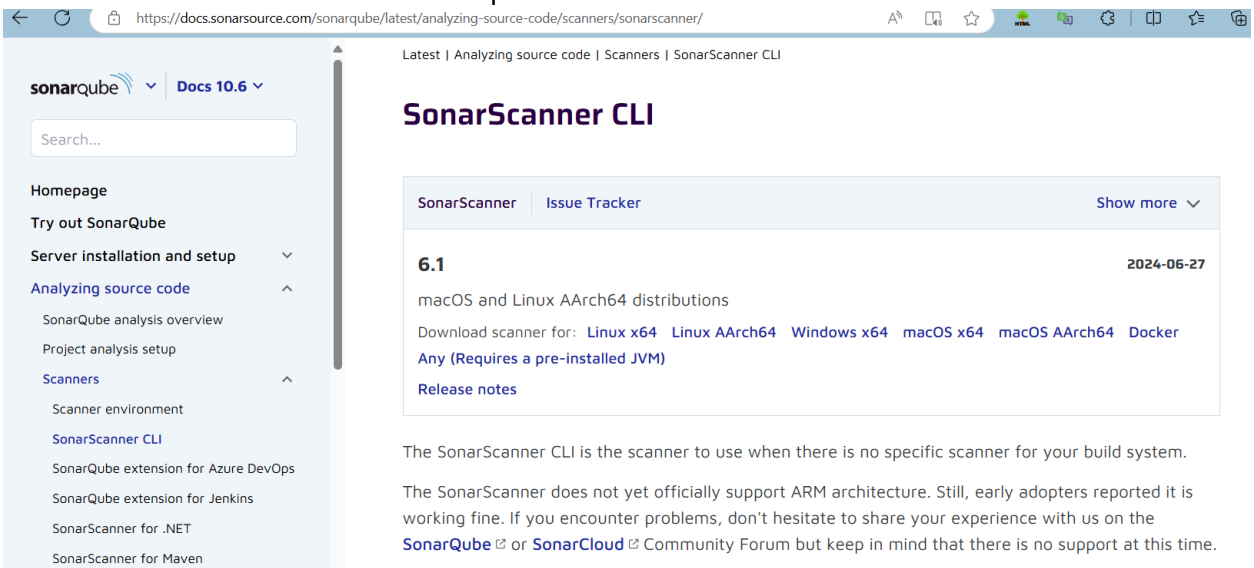
Aim:

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

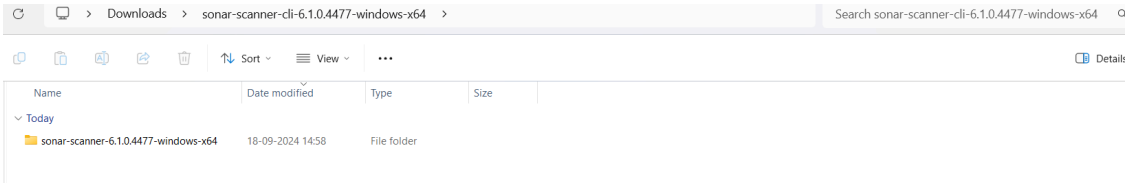
Steps

Step 1: Download sonar scanner

https://docs.sonarsource.com/sonarqube/latest/analyzing-source-code/scanners/sonarscanner/
Visit this link and download the sonarqube scanner CLI.

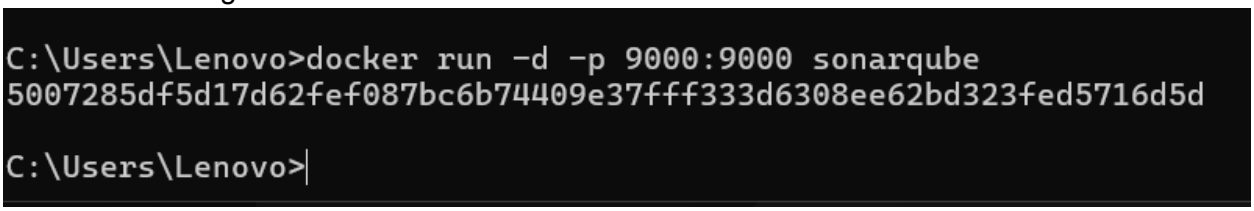


Extract the downloaded zip file in a folder.

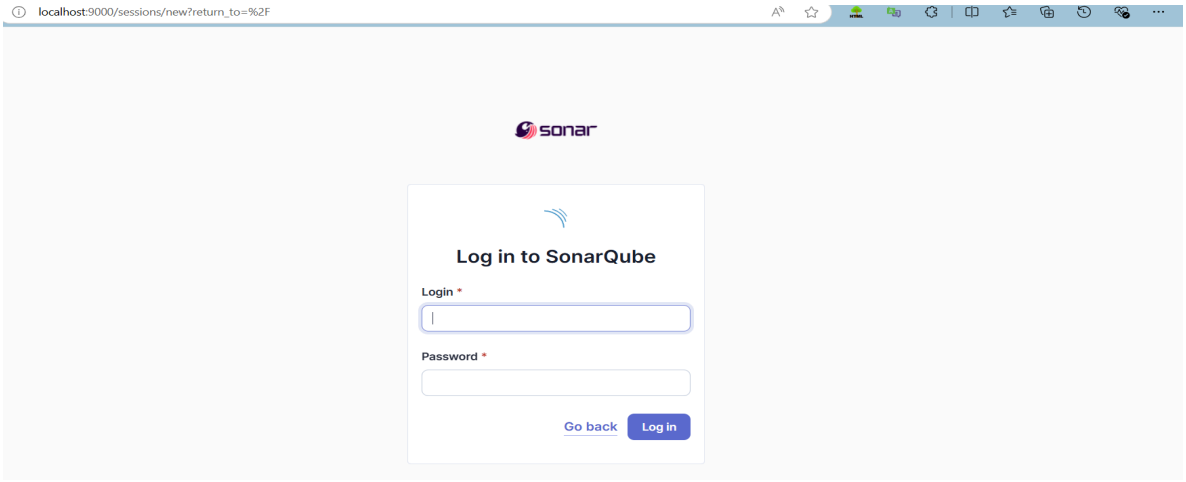


1. Install sonarqube image

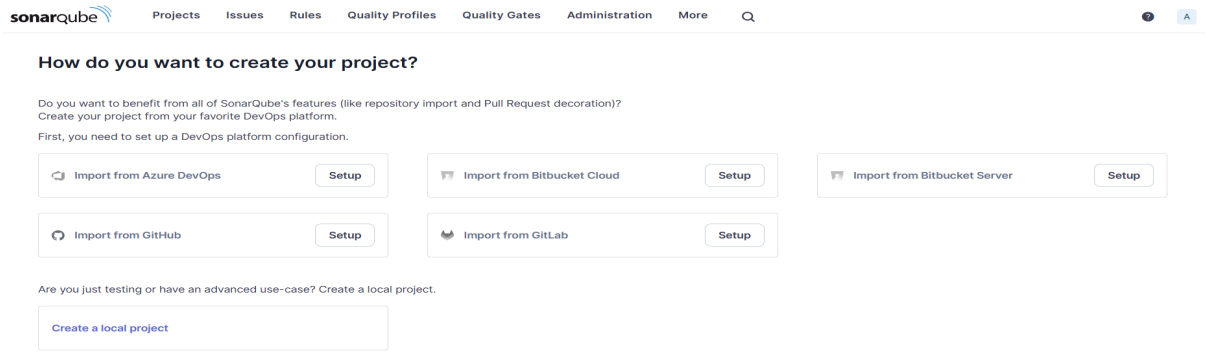
Command: **docker pull sonarqube** (skip if already installed we did install it in exp 7)
Then run the image



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



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



4. Create a local project in SonarQube with the name sonarqube

1 of 2

Create a local project

Project display name *

sonarqube

Project key *

sonarqube

Main branch name *

main

The name of your project's default branch Learn More

Cancel Next

2 of 2

Set up project for Clean as You Code

The new code definition sets which part of your code will be considered new code. This helps you focus att You Code methodology. Learn more: Defining New Code

Choose the baseline for new code for this project

Use the global setting

Previous version

Any code that has changed since the previous version is considered new code.

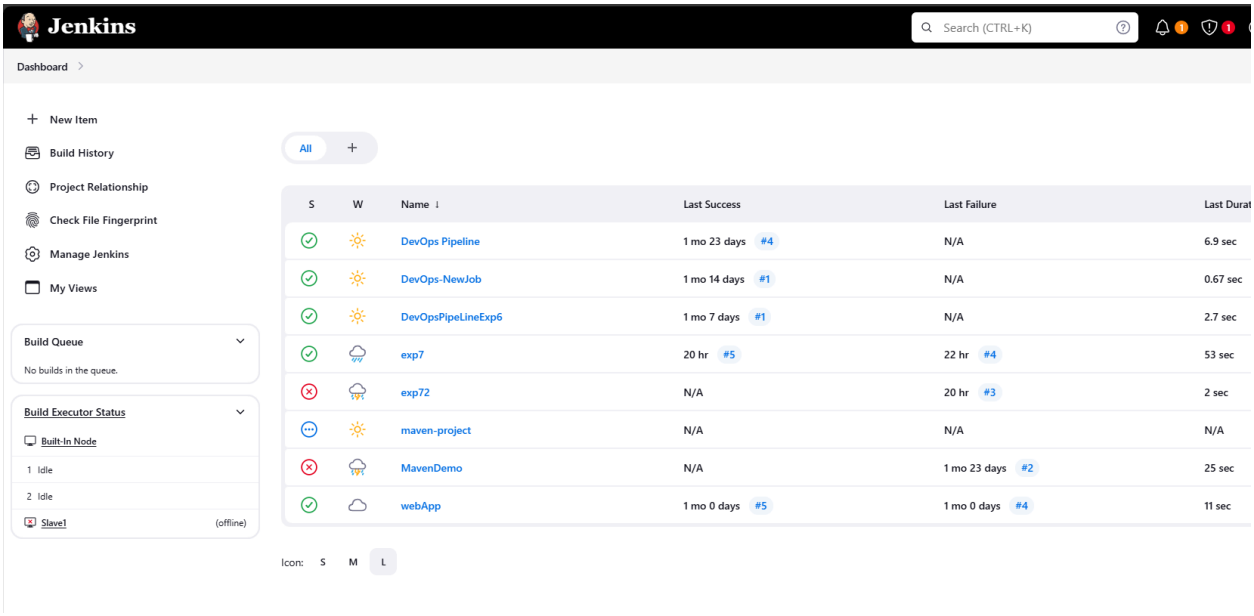
Recommended for projects following regular versions or releases.

Define a specific setting for this project

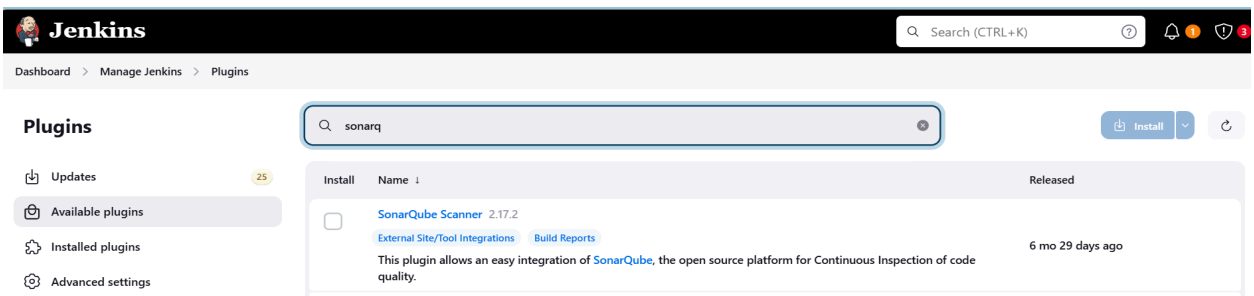
Previous version

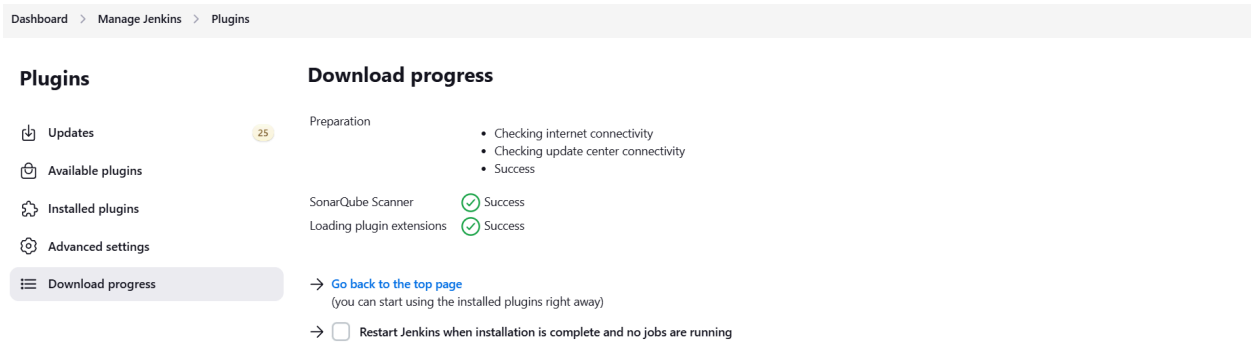
Any code that has changed since the previous version is considered new code.

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



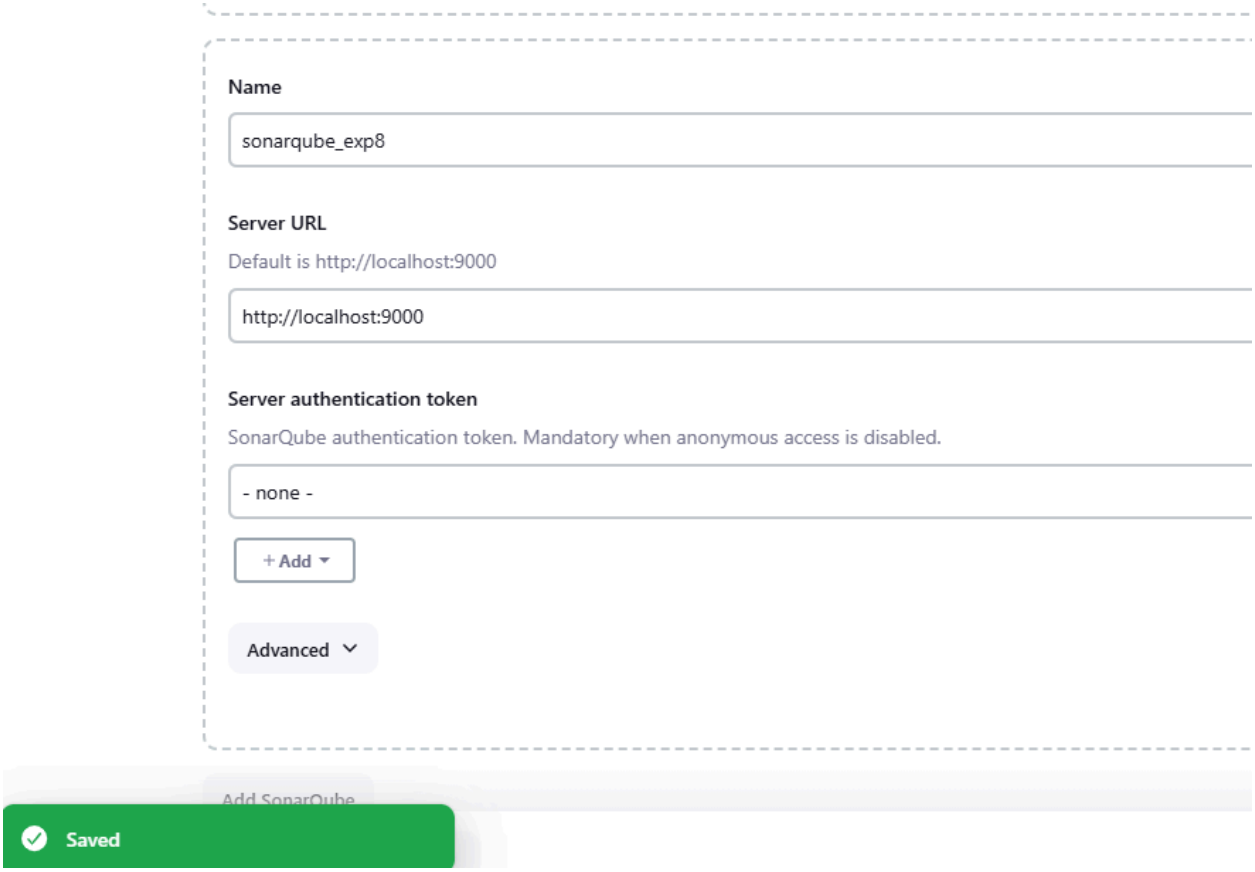
6. Go to Manage Jenkins and search for SonarQube Scanner for Jenkins and install it.(we already installed it for exp 7 so you can skip)



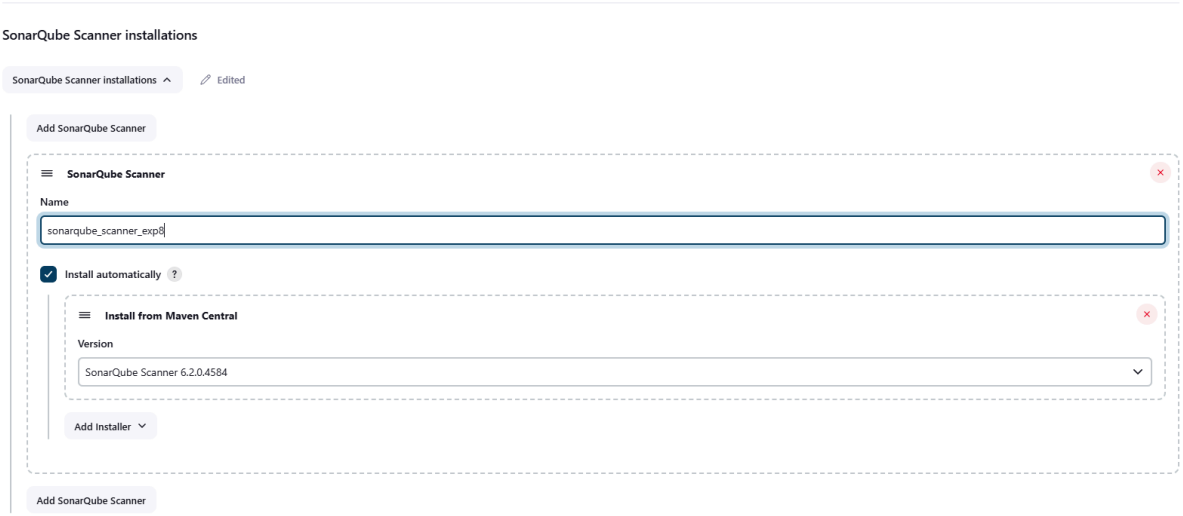


7. Under Jenkins ‘Manage Jenkins’ then go to ‘system’, scroll and look for **SonarQube Servers** and enter the details. Enter the Server Authentication token if needed.(we dont need the token and this step was done in previous exp but jic)

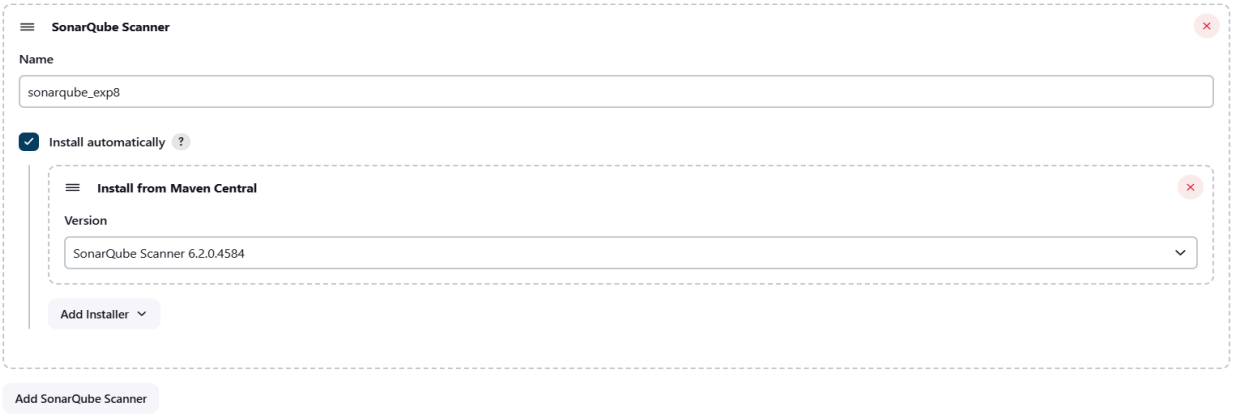
In SonarQube installations: Under **Name** add <project name of sonarqube> for me **sonarqube_exp8**
In **Server URL** Default is **http://localhost:9000**



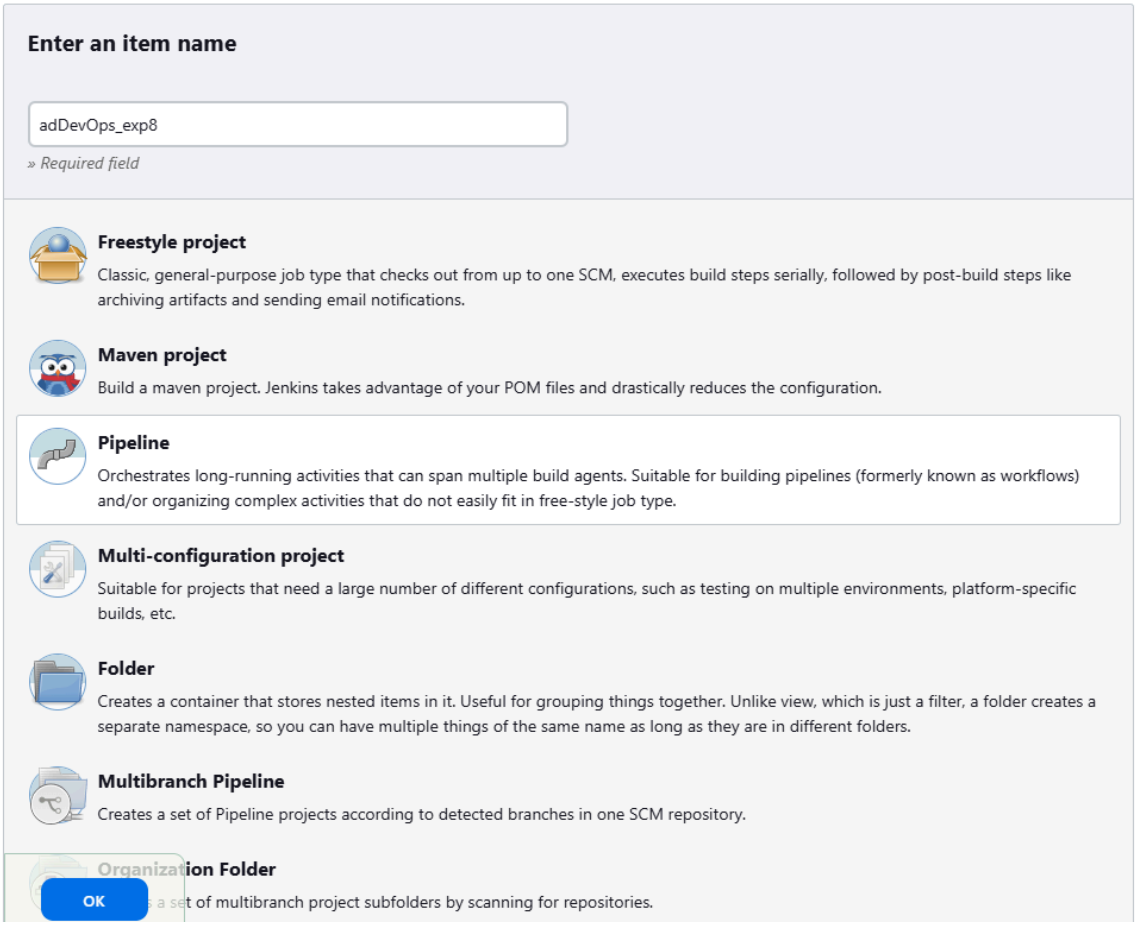
8. Search for SonarQube Scanner under Global Tool Configuration. Choose the latest configuration and choose Install automatically.
Dashboard > Manage Jenkins > Tools > SonarQube Scanner (i kept the default setting from last experiment and just changed the name.



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



9. After configuration, create a New Item → choose a pipeline project.



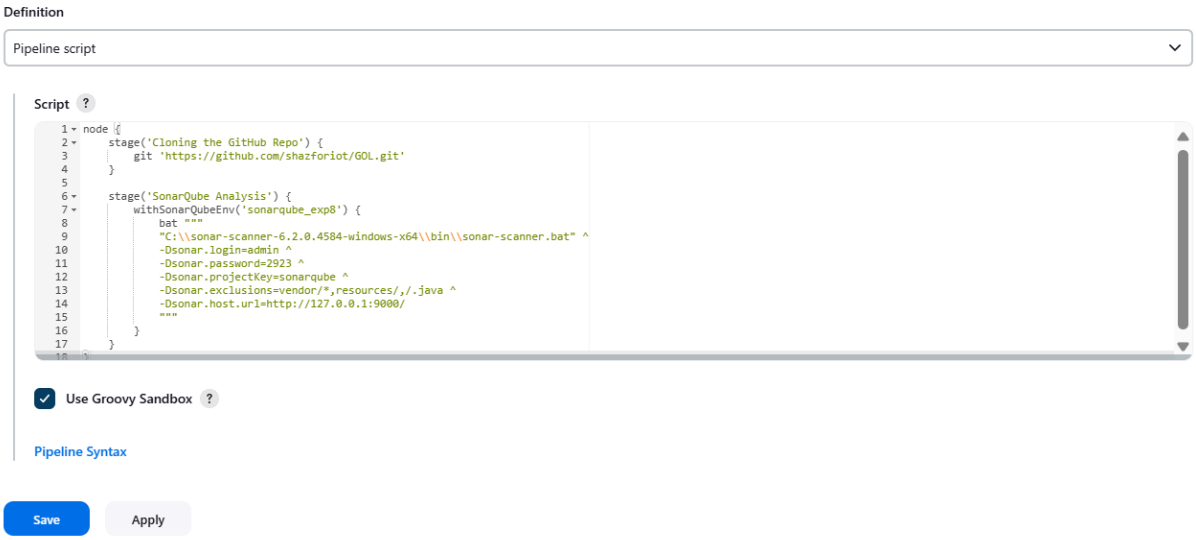
10. Under Pipeline script, enter the following:

```
node {
   stage('Cloning the GitHub Repo') {
      git 'https://github.com/shazforiot/GOL.git'
   }

   stage('SonarQube Analysis') {
      withSonarQubeEnv('exp8') {
         bat """
         "C:\\Program Files\\Sonar Scanner\\sonar-scanner-6.2.0.4584-windows-x64\\bin\\sonar-scanner.bat" ^
         -Dsonar.login=<username> ^
         -Dsonar.password=<password> ^
         -Dsonar.projectKey=<project-key> ^
         -Dsonar.exclusions=vendor/*,resources/,/.java ^
         -Dsonar.host.url=http://127.0.0.1:9000/
```

```
        """
    }
  }
}
```

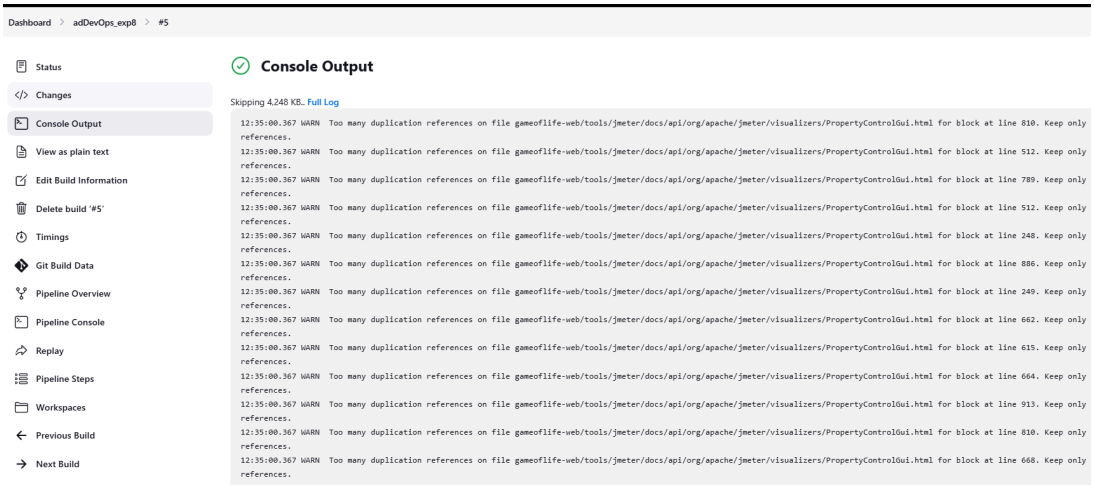
*Note that the code has placeholders

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

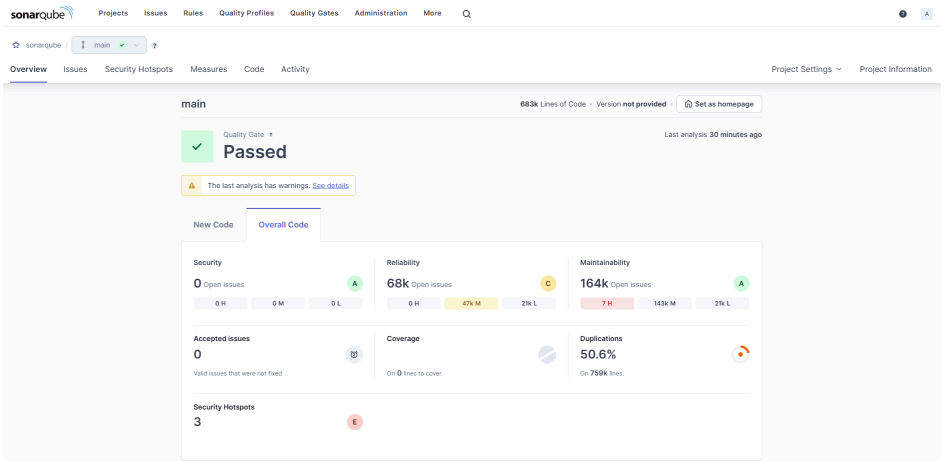


11. Build project

12. Check console

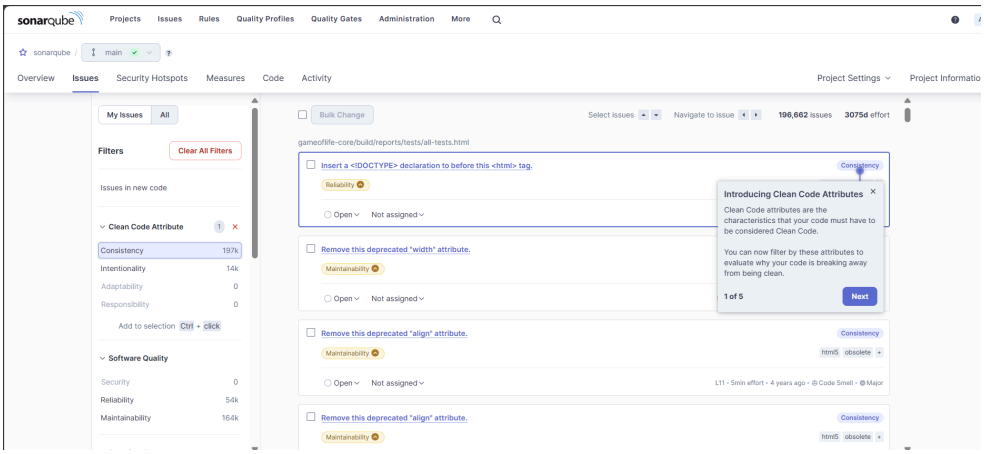


13. Now, check the project in SonarQube

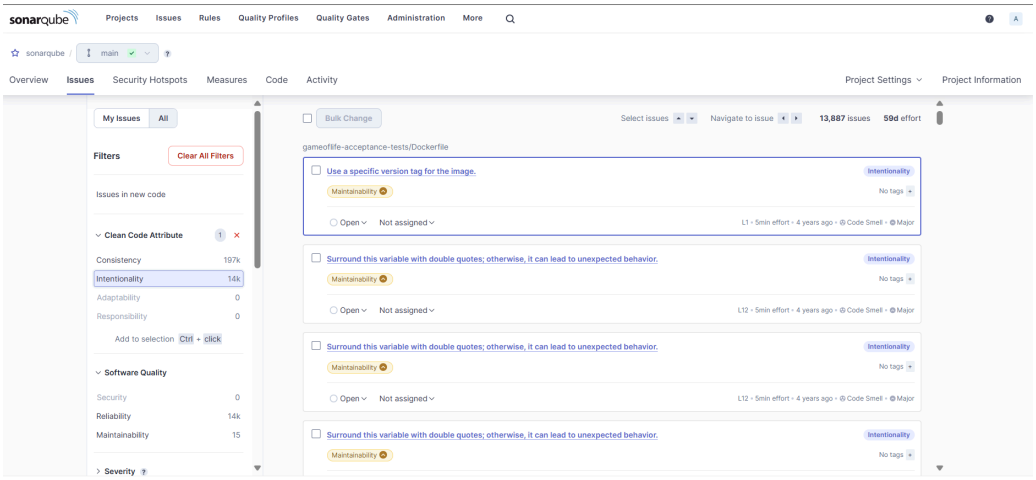


14. Code Problems

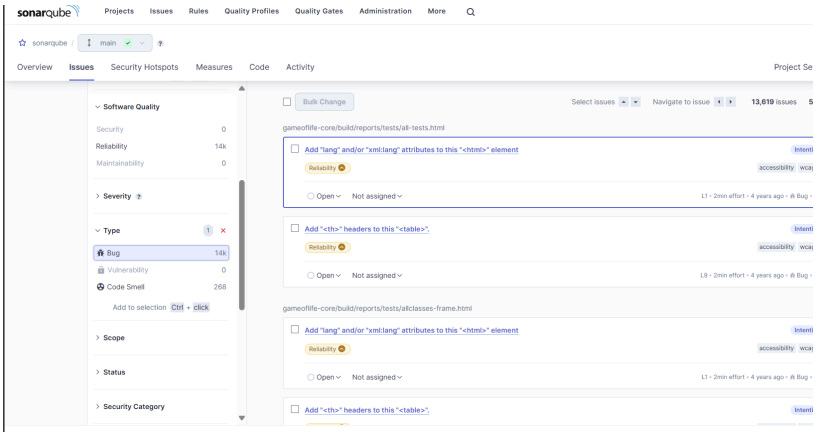
- Consistency



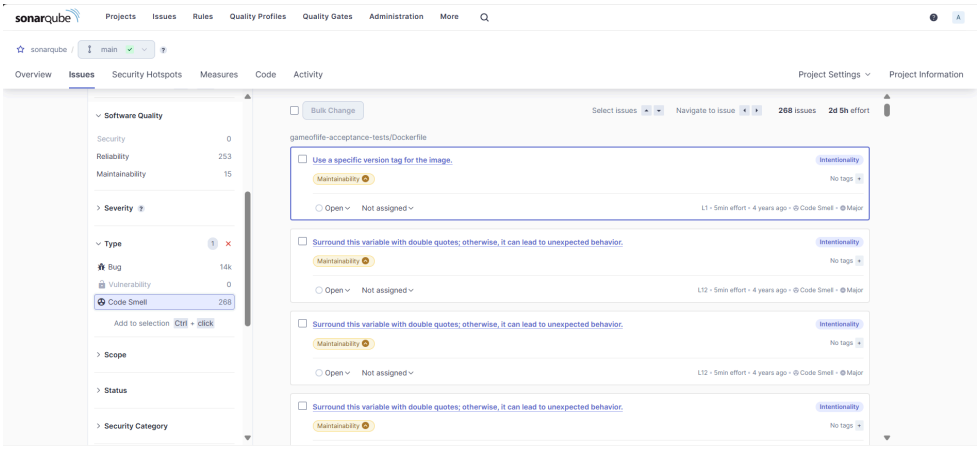
- Intentionality



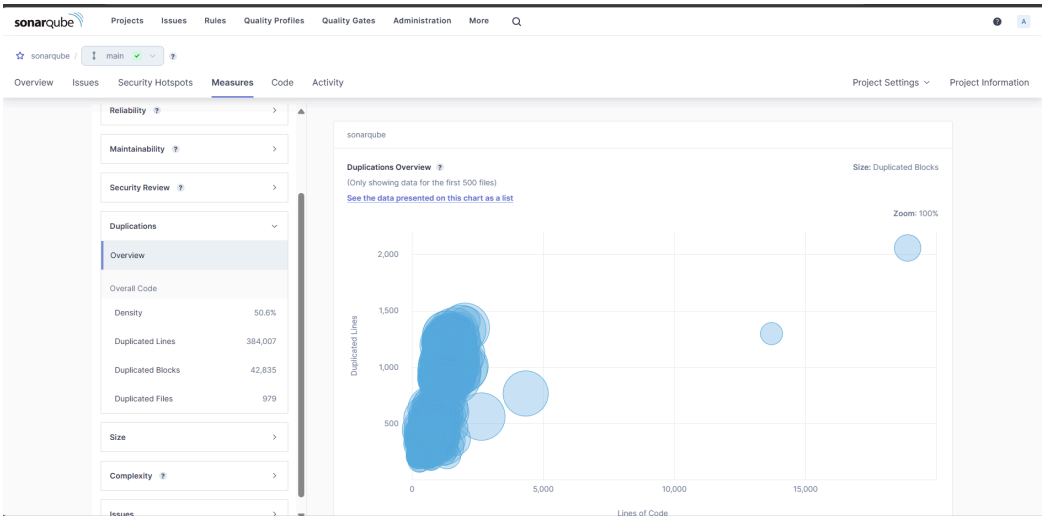
- Bugs



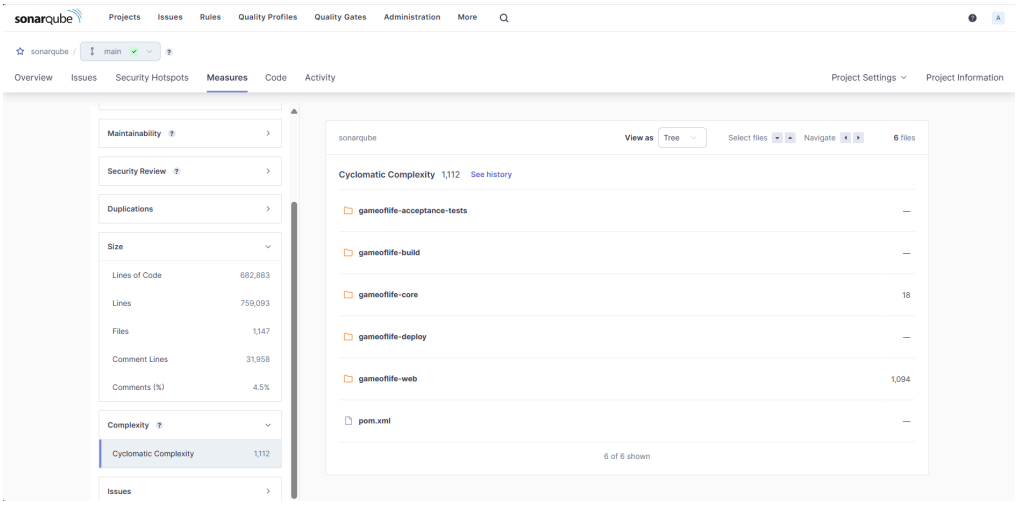
- Code Smells



- Duplications



- Cyclomatic Complexities



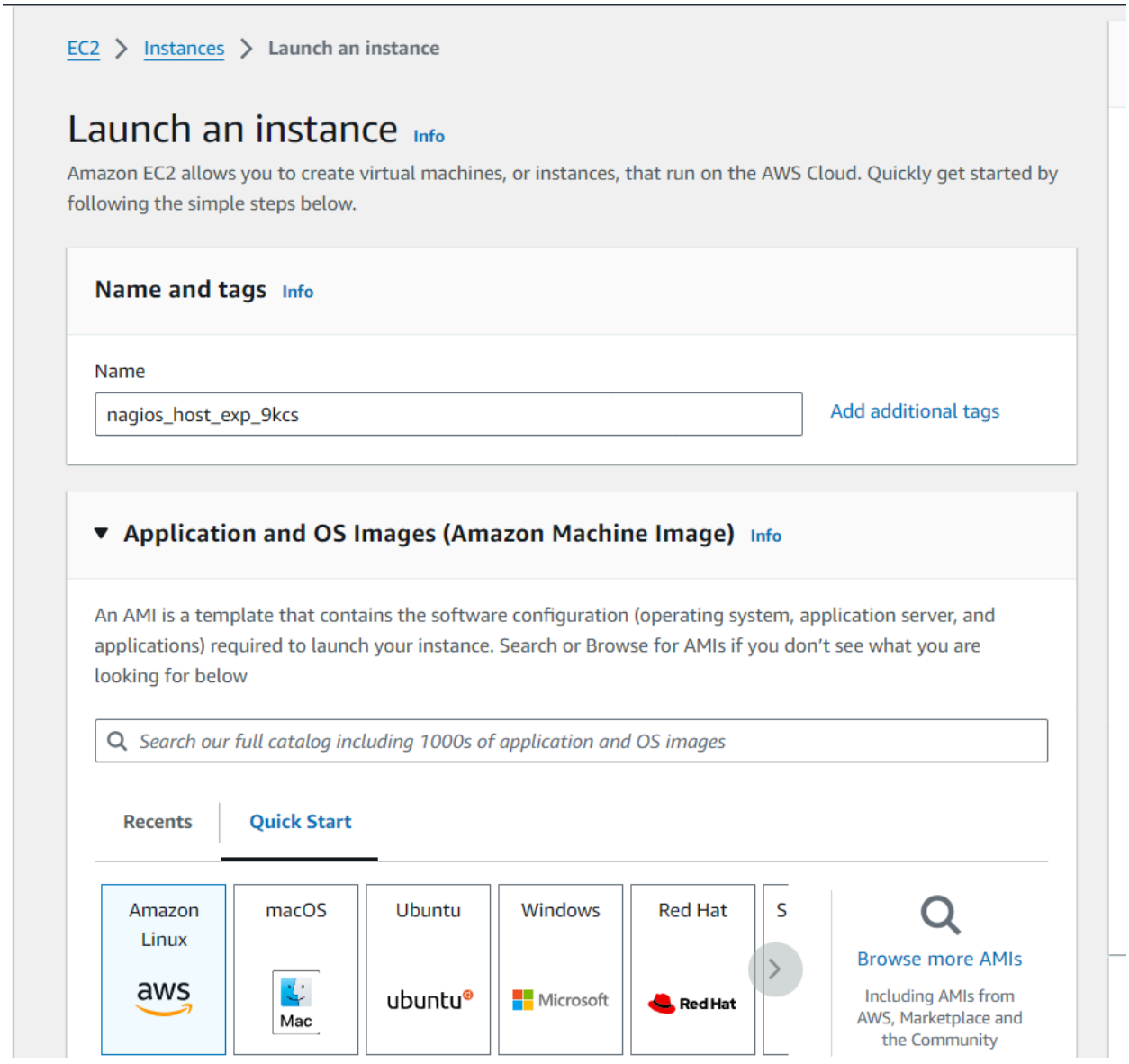
In this way, we have integrated Jenkins with SonarQube for SAST.

Conclusion:

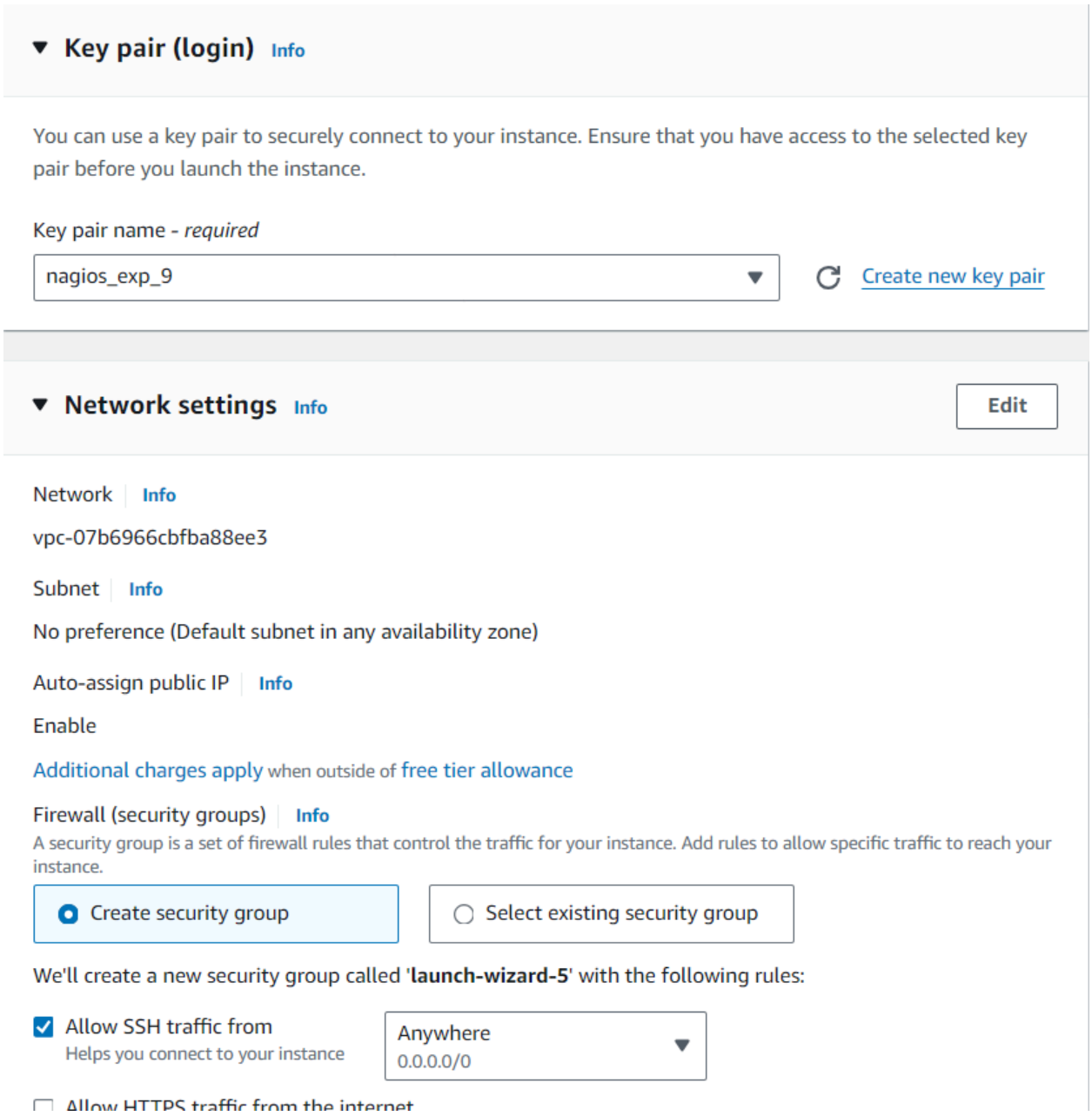
In this experiment, we successfully integrated Jenkins with SonarQube to automate continuous code quality monitoring within our CI/CD pipeline. This process involved deploying SonarQube via Docker, setting up a project for analysis, and configuring Jenkins with the SonarQube Scanner plugin. After configuring the necessary tools and adding SonarQube server details, we created a Jenkins pipeline that automates cloning from GitHub and running static analysis on the code. This integration allows us to detect potential bugs, code smells, and security vulnerabilities at every stage of development, ensuring improved code quality and streamlined development workflows.

Steps

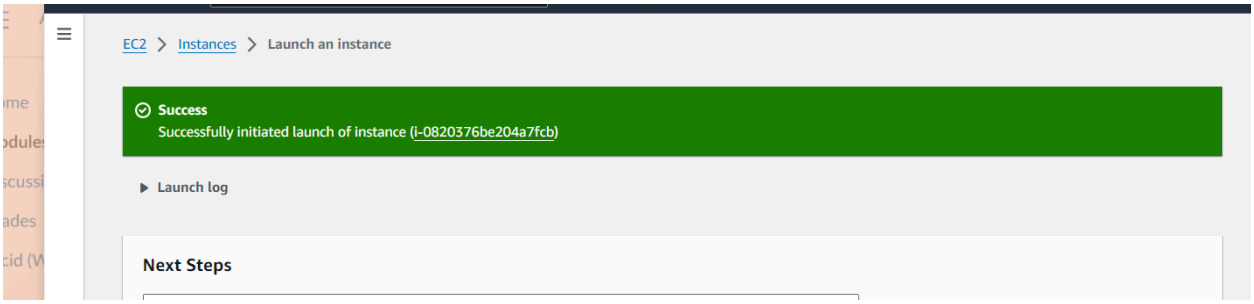
Launch an ec2 instance
Give name use the default OS



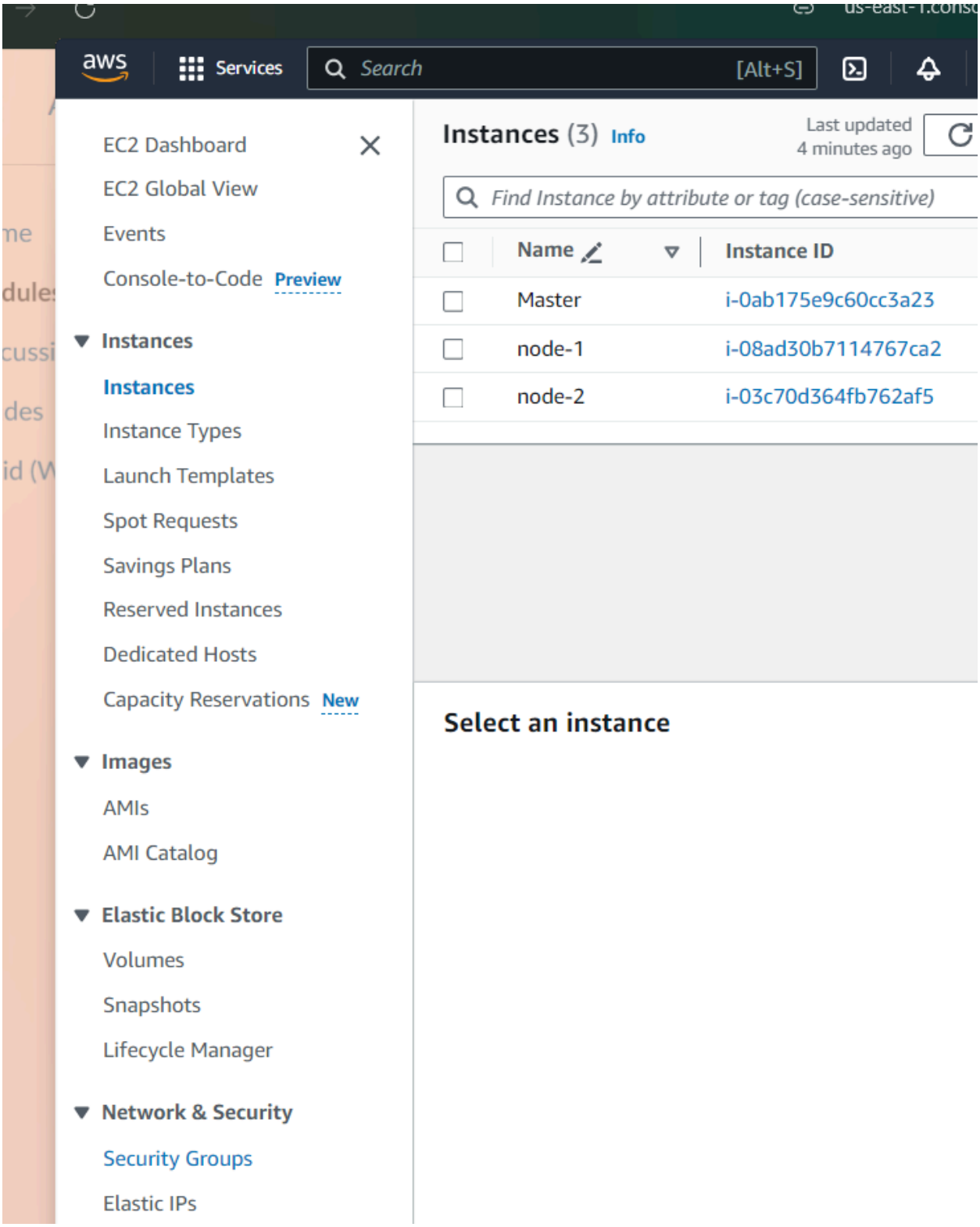
Make a key pair and use it.



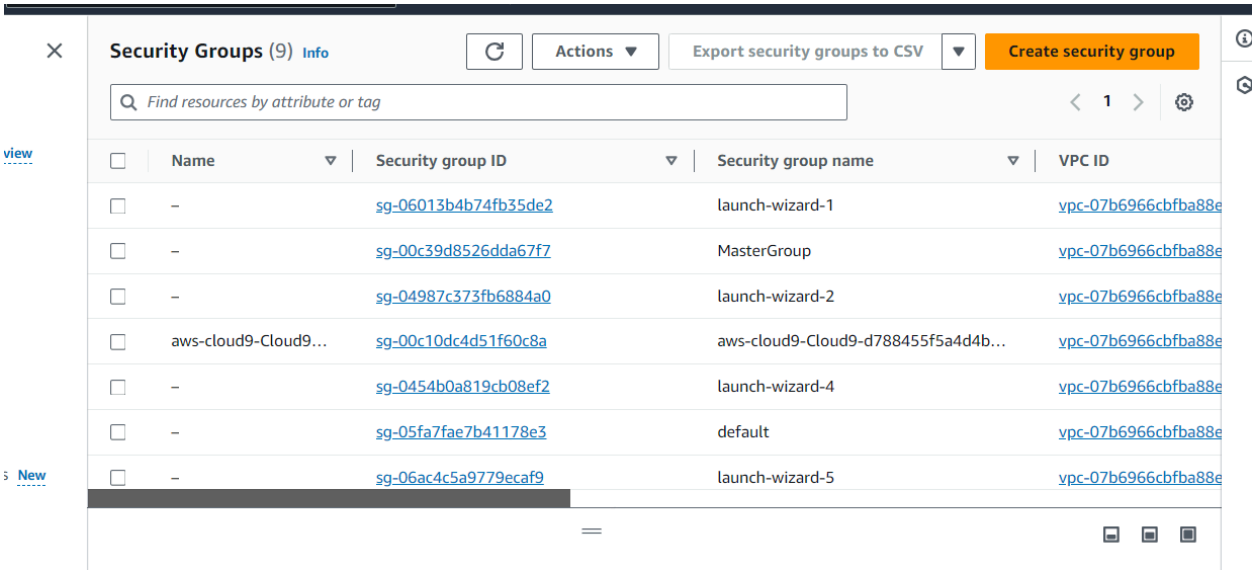
Note the name of the security group that was created for future use:
here it is ‘ launch-wizard-5 ’


EC2 > Instances > Launch an instance
Success
Successfully initiated launch of instance (i-0820376be204a7fcb)
Launch log
Next Steps

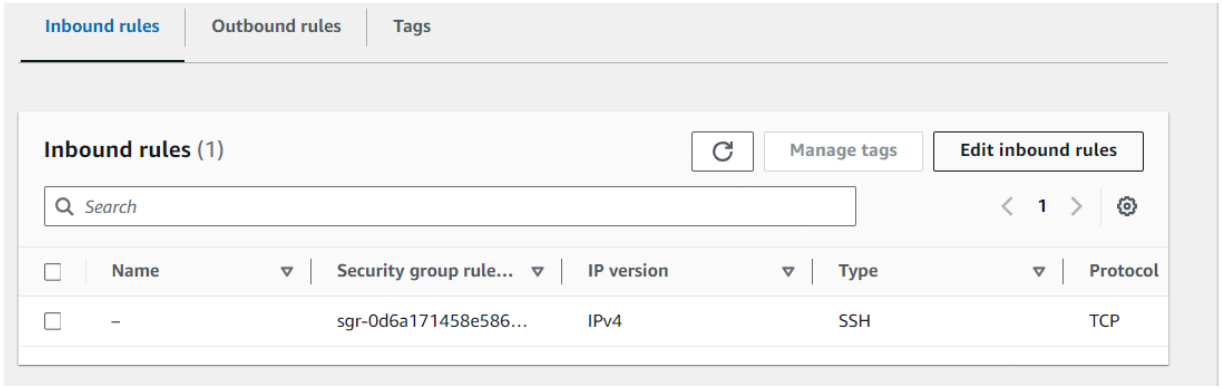
go to security groups:



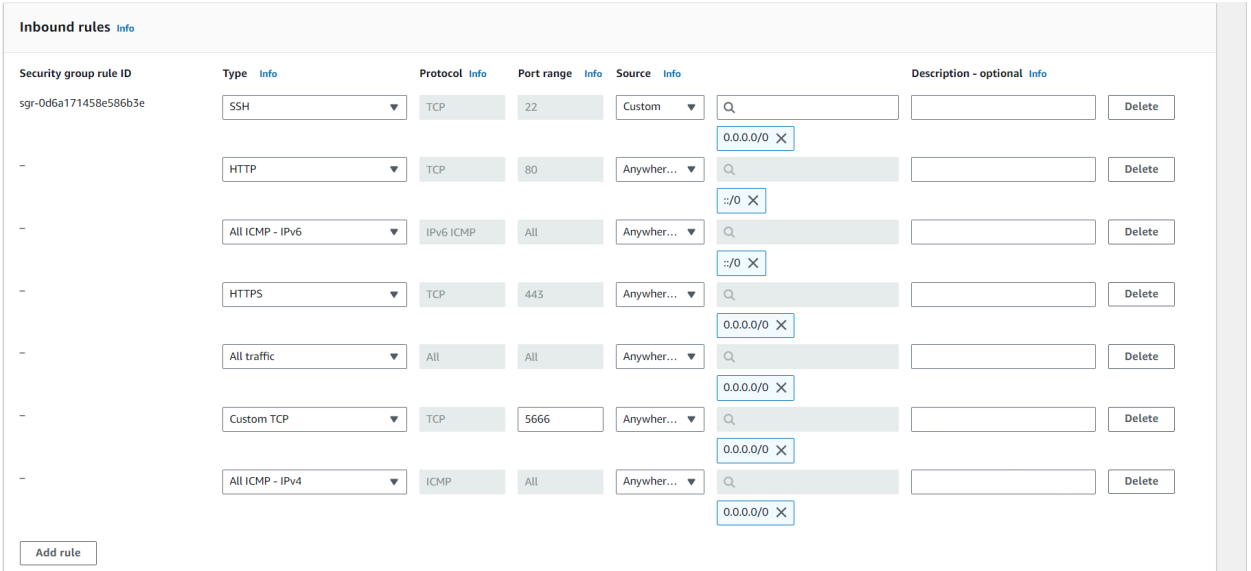
click the security group id which was created while you created the ec2 instance of this experiment.



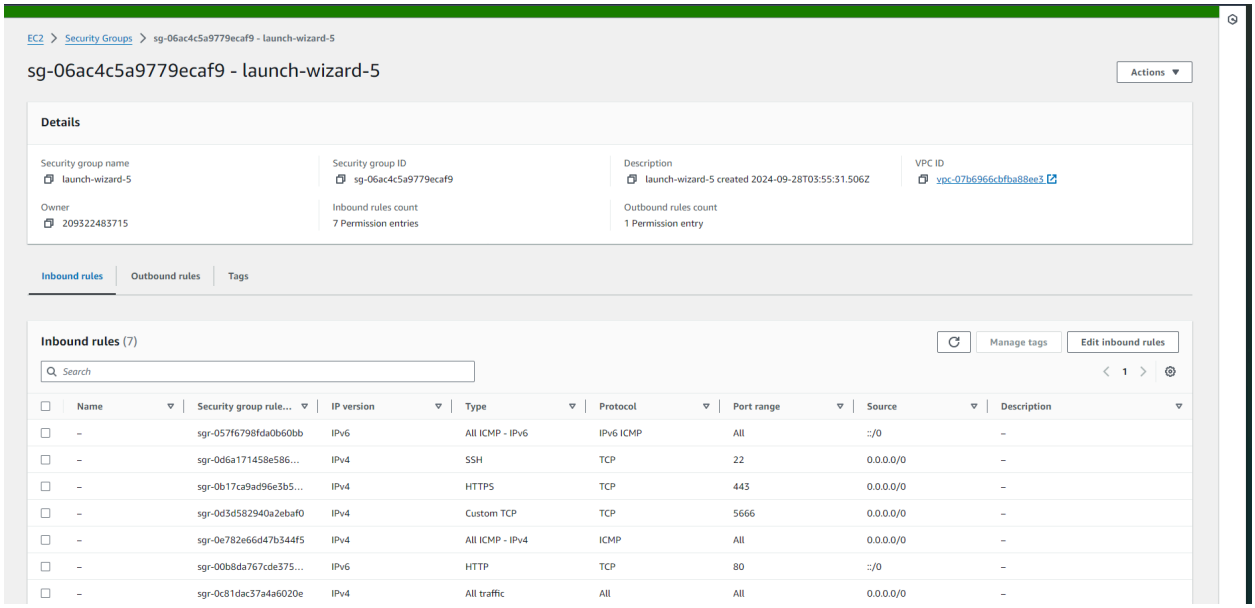
now click on edit inbound rules



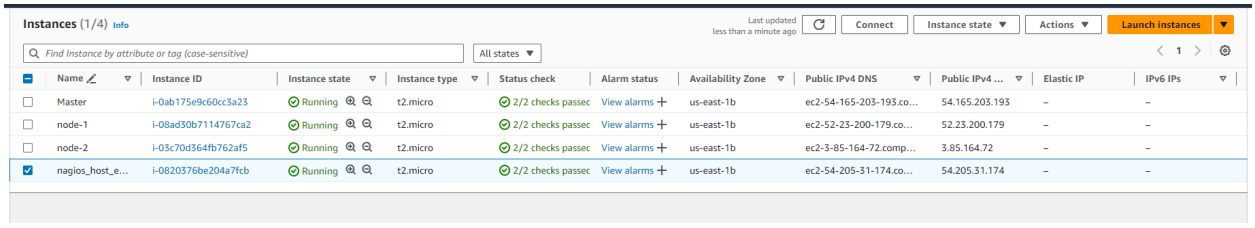
now do the following configurations:



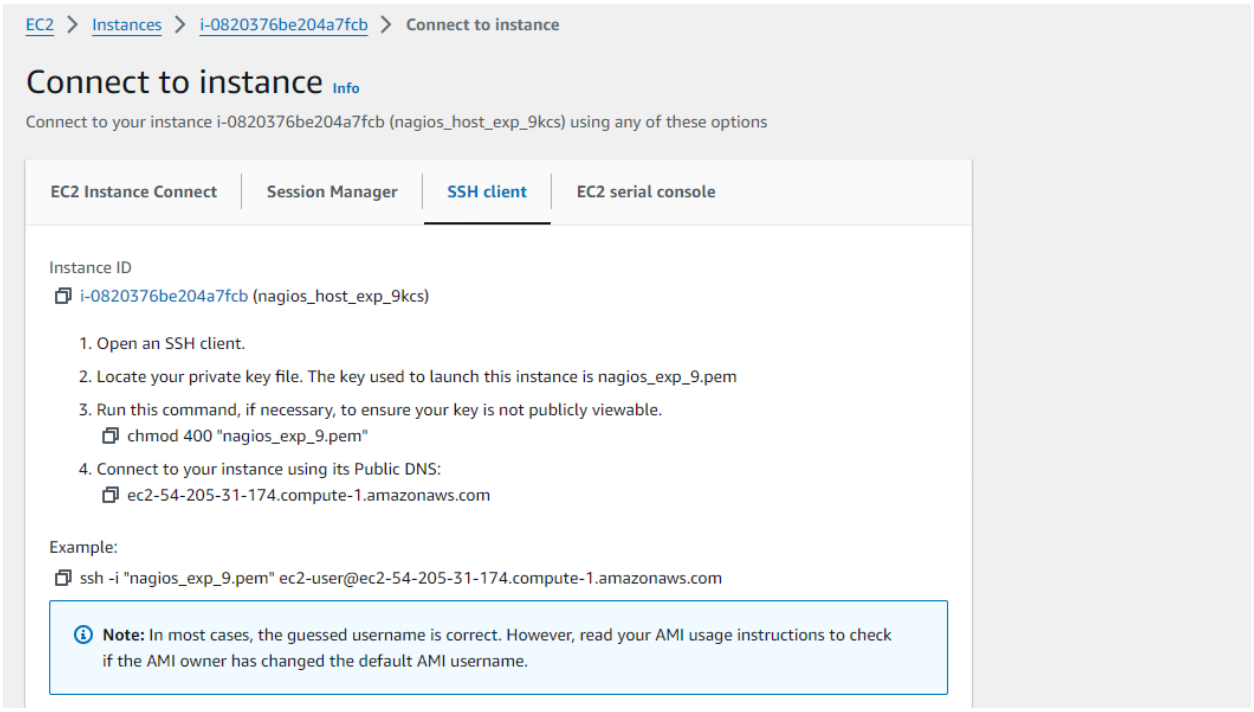
by clicking “add rules”
then click on save rules.



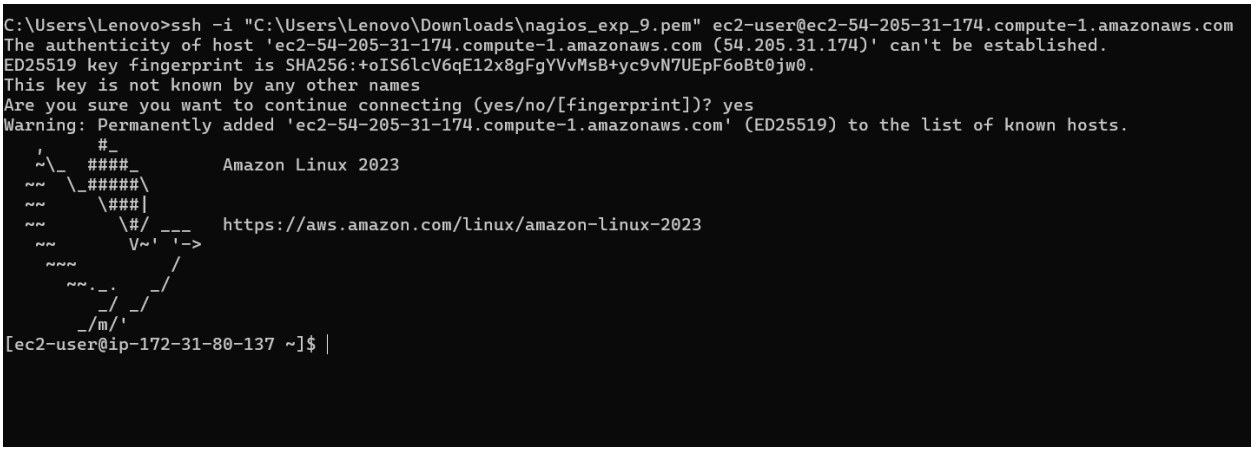
now navigate to instances, click on the instance which was created earlier and click on connect.



now copy the ssh command and just replace the .pem file with its actual location in your computer.



paste the command in your terminal and enter after replacing the .pem file with its actual location in your system.



now paste the following commands in your connected terminal:

sudo yum update

```
       _/m/'
[ec2-user@ip-172-31-80-137 ~]$ sudo yum update
Last metadata expiration check: 2:21:45 ago on Sat Sep 28 03:59:04 2024.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-80-137 ~]$
```

sudo yum install httpd php

```
[ec2-user@ip-172-31-80-137 ~]$ sudo yum install httpd php
Last metadata expiration check: 2:22:53 ago on Sat Sep 28 03:59:04 2024.
Dependencies resolved.
================================================================================
 Package                 Architecture       Version                    Repository
================================================================================
Installing:
 httpd                   x86_64             2.4.62-1.amzn2023          amazonlinux
 php8.3                  x86_64             8.3.10-1.amzn2023.0.1      amazonlinux
Installing dependencies:
 apr                     x86_64             1.7.2-2.amzn2023.0.2       amazonlinux
 apr-util                x86_64             1.6.3-1.amzn2023.0.1       amazonlinux
 generic-logos-httpd     noarch             18.0.0-12.amzn2023.0.3     amazonlinux
 httpd-core              x86_64             2.4.62-1.amzn2023          amazonlinux
 httpd-filesystem        noarch             2.4.62-1.amzn2023          amazonlinux
```

(type y when prompted)

sudo yum install gcc glibc glibc-common

```
Dependencies resolved.
================================================================================
 Package                 Architecture       Version                    Repository
================================================================================
Installing:
 gcc                     x86_64             11.4.1-2.amzn2023.0.2      amazonlinux
Installing dependencies:
 annobin-docs            noarch             10.93-1.amzn2023.0.1       amazonlinux
 annobin-plugin-gcc      x86_64             10.93-1.amzn2023.0.1       amazonlinux
 cpp                     x86_64             11.4.1-2.amzn2023.0.2      amazonlinux
 gc                      x86_64             8.0.4-5.amzn2023.0.2       amazonlinux
 glibc-devel             x86_64             2.34-52.amzn2023.0.11      amazonlinux
 glibc-headers-x86       noarch             2.34-52.amzn2023.0.11      amazonlinux
 guile22                 x86_64             2.2.7-2.amzn2023.0.3       amazonlinux
 kernel-headers          x86_64             6.1.109-118.189.amzn2023   amazonlinux
 libmpc                  x86_64             1.2.1-2.amzn2023.0.2       amazonlinux
 libtool-ltdl            x86_64             2.4.7-1.amzn2023.0.3       amazonlinux
 libxcrypt-devel         x86_64             4.4.33-7.amzn2023          amazonlinux
 make                    x86_64             1:4.3-5.amzn2023.0.2       amazonlinux

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

Total download size: 52 M
Installed size: 168 M
Is this ok [y/N]: y
```

sudo yum install gd gd-devel

```
google-noto-sans-vf-fonts-20201206-2.amzn2023.0.2.noarch    graphite2-1.3.14-7.amzn2023.0.2.x86_64
graphite2-devel-1.3.14-7.amzn2023.0.2.x86_64                harfbuzz-7.0.0-2.amzn2023.0.1.x86_64
harfbuzz-devel-7.0.0-2.amzn2023.0.1.x86_64                  harfbuzz-icu-7.0.0-2.amzn2023.0.1.x86_64
jbigkit-libs-2.1-21.amzn2023.0.2.x86_64                     langpacks-core-font-en-3.0-21.amzn2023.0.4.noarch
libICE-1.0.10-6.amzn2023.0.2.x86_64                         libSM-1.2.3-8.amzn2023.0.2.x86_64
libX11-1.7.2-3.amzn2023.0.4.x86_64                          libX11-common-1.7.2-3.amzn2023.0.4.noarch
libX11-devel-1.7.2-3.amzn2023.0.4.x86_64                    libX11-xcb-1.7.2-3.amzn2023.0.4.x86_64
libXau-1.0.9-6.amzn2023.0.2.x86_64                          libXau-devel-1.0.9-6.amzn2023.0.2.x86_64
libXext-1.3.4-6.amzn2023.0.2.x86_64                         libXpm-3.5.15-2.amzn2023.0.3.x86_64
libXpm-devel-3.5.15-2.amzn2023.0.3.x86_64                   libXrender-0.9.10-14.amzn2023.0.2.x86_64
libXt-1.2.0-4.amzn2023.0.2.x86_64                           libblkid-devel-2.37.4-1.amzn2023.0.4.x86_64
libffi-devel-3.4.4-1.amzn2023.0.1.x86_64                    libicu-67.1-7.amzn2023.0.3.x86_64
libicu-devel-67.1-7.amzn2023.0.3.x86_64                     libjpeg-turbo-2.1.4-2.amzn2023.0.5.x86_64
libjpeg-turbo-devel-2.1.4-2.amzn2023.0.5.x86_64             libmount-devel-2.37.4-1.amzn2023.0.4.x86_64
libpng-2:1.6.37-10.amzn2023.0.6.x86_64                      libpng-devel-2:1.6.37-10.amzn2023.0.6.x86_64
libselinux-devel-3.4-5.amzn2023.0.2.x86_64                  libsepol-devel-3.4-3.amzn2023.0.3.x86_64
libtiff-4.4.0-4.amzn2023.0.18.x86_64                        libtiff-devel-4.4.0-4.amzn2023.0.18.x86_64
libwebp-1.2.4-1.amzn2023.0.6.x86_64                         libwebp-devel-1.2.4-1.amzn2023.0.6.x86_64
libxcb-1.13.1-7.amzn2023.0.2.x86_64                         libxcb-devel-1.13.1-7.amzn2023.0.2.x86_64
libxml2-devel-2.10.4-1.amzn2023.0.6.x86_64                  pcre2-devel-10.40-1.amzn2023.0.3.x86_64
pcre2-utf16-10.40-1.amzn2023.0.3.x86_64                     pcre2-utf32-10.40-1.amzn2023.0.3.x86_64
pixman-0.40.0-3.amzn2023.0.3.x86_64                         sysprof-capture-devel-3.40.1-2.amzn2023.0.2.x86_64
xml-common-0.6.3-56.amzn2023.0.2.noarch                     xorg-x11-proto-devel-2021.4-1.amzn2023.0.2.noarch
xz-devel-5.2.5-9.amzn2023.0.2.x86_64                        zlib-devel-1.2.11-33.amzn2023.0.5.x86_64

omplete!
```

sudo adduser -m nagios
sudo passwd nagios

```
Complete!
[ec2-user@ip-172-31-80-137 ~]$ sudo adduser -m nagios
sudo passwd nagios
Changing password for user nagios.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: all authentication tokens updated successfully.
[ec2-user@ip-172-31-80-137 ~]$
```

(add a password here)

sudo groupadd nagcmd

```
[ec2-user@ip-172-31-80-137 ~]$ sudo groupadd nagcmd
[ec2-user@ip-172-31-80-137 ~]$
```

sudo usermod -a -G nagcmd nagios
sudo usermod -a -G nagcmd apache

```
[ec2-user@ip-172-31-80-137 ~]$ sudo groupadd nagcmd
[ec2-user@ip-172-31-80-137 ~]$ sudo usermod -a -G nagcmd nagios
sudo usermod -a -G nagcmd apache
[ec2-user@ip-172-31-80-137 ~]$
```

mkdir ~/downloads
cd ~/downloads

```
[ec2-user@ip-172-31-80-137 ~]$ mkdir ~/downloads
cd ~/downloads
[ec2-user@ip-172-31-80-137 downloads]$
```

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

```
cd ~/downloads
[ec2-user@ip-172-31-80-137 downloads]$ wget https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.5.tar.gz
--2024-09-28 06:27:51--  https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.5.tar.gz
Resolving assets.nagios.com (assets.nagios.com)... 45.79.49.120, 2600:3c00::f03c:92ff:fef7:45ce
Connecting to assets.nagios.com (assets.nagios.com)|45.79.49.120|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2065473 (2.0M) [application/x-gzip]
Saving to: ‘nagios-4.5.5.tar.gz’

nagios-4.5.5.tar.gz           100%[==================================================>]   1.97M  5.30MB/s    in 0.4s

2024-09-28 06:27:52 (5.30 MB/s) - ‘nagios-4.5.5.tar.gz’ saved [2065473/2065473]

[ec2-user@ip-172-31-80-137 downloads]$
```

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

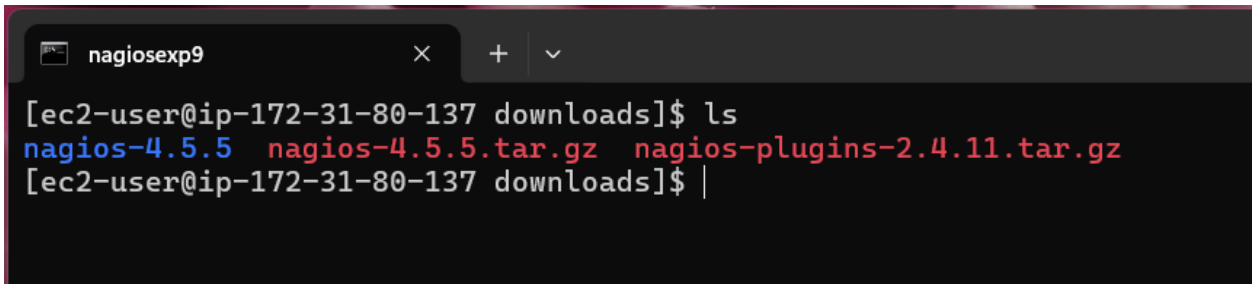
```
[ec2-user@ip-172-31-80-137 downloads]$ wget https://nagios-plugins.org/download/nagios-plugins-2.4.11.tar.gz
--2024-09-28 06:28:14--  https://nagios-plugins.org/download/nagios-plugins-2.4.11.tar.gz
Resolving nagios-plugins.org (nagios-plugins.org)... 45.56.123.251
Connecting to nagios-plugins.org (nagios-plugins.org)|45.56.123.251|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2753049 (2.6M) [application/x-gzip]
Saving to: ‘nagios-plugins-2.4.11.tar.gz’

nagios-plugins-2.4.11.tar.gz  100%[==================================================>]   2.62M  5.90MB/s    in 0.4s

2024-09-28 06:28:15 (5.90 MB/s) - ‘nagios-plugins-2.4.11.tar.gz’ saved [2753049/2753049]

[ec2-user@ip-172-31-80-137 downloads]$
```

tar zxvf nagios-4.5.5.tar.gz

```
[ec2-user@ip-172-31-80-137 downloads]$ tar zxvf nagios-4.5.5.tar.gz
nagios-4.5.5/
nagios-4.5.5/.github/
nagios-4.5.5/.github/workflows/
nagios-4.5.5/.github/workflows/test.yml
nagios-4.5.5/.gitignore
nagios-4.5.5/CONTRIBUTING.md
nagios-4.5.5/Changelog
nagios-4.5.5/INSTALLING
nagios-4.5.5/LEGAL
nagios-4.5.5/LICENSE
nagios-4.5.5/Makefile.in
```

Now we have to first navigate to the nagios-4.5.5 folder in downloads.

- **commands to enter:**
  ls (verify whether nagios-4.5.5 exists)

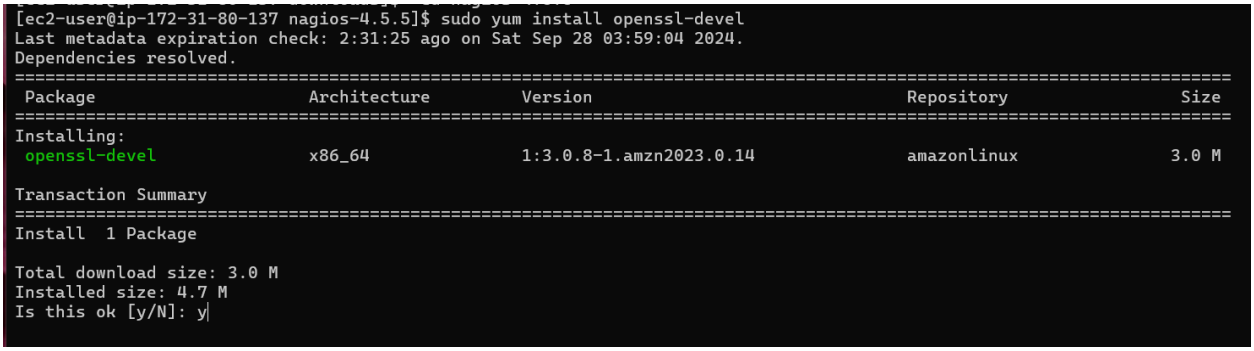


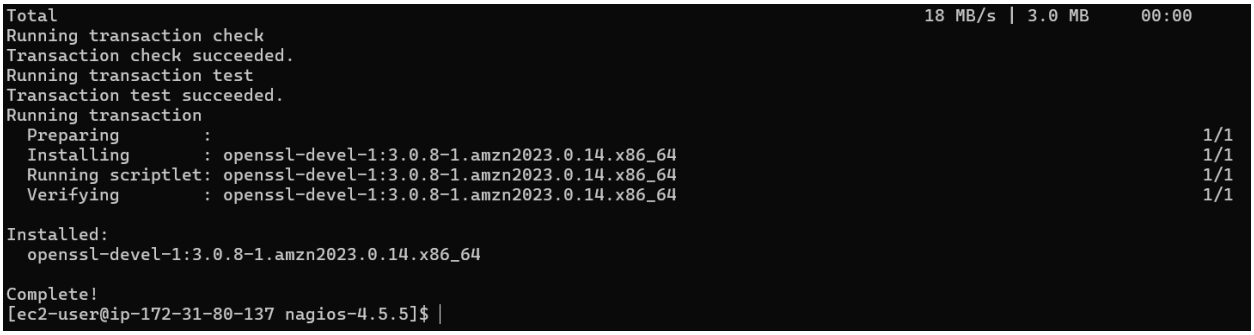
cd nagios-4.5.5

we now have to install openssl dev library

commands to enter:

sudo yum install openssl-devel





Then finally we can run the commands like usual.

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

```
nagiosexp9
[ec2-user@ip-172-31-80-137 nagios-4.5.5]$ ./configure --with-command-group=nagcmd
checking for a BSD-compatible install... /usr/bin/install -c
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether the compiler supports GNU C... yes
checking whether gcc accepts -g... yes
checking for gcc option to enable C11 features... none needed
checking whether make sets $(MAKE)... yes
checking whether ln -s works... yes
checking for strip... /usr/bin/strip
checking for sys/wait.h that is POSIX.1 compatible... yes
checking for stdio.h... yes
checking for stdlib.h... yes
checking for string.h... yes
checking for inttypes.h... yes
checking for stdint.h... yes
```

make all

```
[ec2-user@ip-172-31-80-137 nagios-4.5.5]$ make all
cd ./base && make
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/base'
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o nagios.o ./nagi
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o broker.o broker
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o nebmods.o nebmo
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o ../common/share
./common/shared.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o query-handler.o
y-handler.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o workers.o worke
In function 'get_wproc_list':
```

```
      - Relevant snippets from your config files
      - Relevant error messages from the Nagios log file

  For more information on obtaining support for Nagios, visit:

         https://support.nagios.com

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

  Enjoy.
```

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

```
[ec2-user@ip-172-31-80-137 nagios-4.5.5]$ sudo make install
sudo make install-init
sudo make install-config
sudo make install-commandmode
cd ./base && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/base'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagios /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagiostats /usr/local/nagios/bin
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5/base'
cd ./cgi && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
make install-basic
make[2]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/sbin
for file in *.cgi; do \
```

Now the next command will take us to nano editor:

---

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

```
nagiosexp9
  GNU nano 5.8                          /usr/local/nagios/etc/objects/contacts.cfg
##############################################################################
# CONTACTS.CFG - SAMPLE CONTACT/CONTACTGROUP DEFINITIONS
#
#
# NOTES: This config file provides you with some example contact and contact
#        group definitions that you can reference in host and service
#        definitions.
#
#        You don't need to keep these definitions in a separate file from your
#        other object definitions.  This has been done just to make things
#        easier to understand.
#
##############################################################################



##############################################################################
#
# CONTACTS
#
##############################################################################

# Just one contact defined by default - the Nagios admin (that's you)
                                [ Read 51 lines ]
^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo
```

navigate down to email: and change it to your email address.

```
# This contact definition inherits a lot of default values from the
# 'generic-contact' template which is defined elsewhere.

define contact {

    contact_name            nagiosadmin             ; Short name of user
    use                     generic-contact         ; Inherit default values from generic-contact template (defined abo
    alias                   Nagios Admin            ; Full name of user
    email                   nagios@localhost ; <****** CHANGE THIS TO YOUR EMAIL ADDRESS ******
}



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

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

define contact {

    contact_name            nagiosadmin             ; Short name of user
    use                     generic-contact         ; Inherit default values from generic-contact template (define
    alias                   Nagios Admin            ; Full name of user
    email                   2022.shubham.jha@ves.ac.in ; <<***** CHANGE THIS TO YOUR EMAIL ADDRESS ******
}



###############################################################################
#
# CONTACT GROUPS

^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo       M-A Set M
```

press Ctrl+O and then enter.

then press Ctrl +X

```
chmod g+s /usr/local/nagios/var/rw

*** External command directory configured ***

[ec2-user@ip-172-31-80-137 nagios-4.5.5]$ sudo nano /usr/local/nagios/etc/objects/contacts.cfg
[ec2-user@ip-172-31-80-137 nagios-4.5.5]$
```

---

sudo make install-webconf

```
[ec2-user@ip-172-31-80-137 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-172-31-80-137 nagios-4.5.5]$
```

Adding password for nagios admin

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

```
[ec2-user@ip-172-31-80-137 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-172-31-80-137 nagios-4.5.5]$
```

sudo service httpd restart

```
[ec2-user@ip-172-31-80-137 nagios-4.5.5]$ sudo service httpd restart
Redirecting to /bin/systemctl restart httpd.service
[ec2-user@ip-172-31-80-137 nagios-4.5.5]$
```

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

```
[ec2-user@ip-172-31-80-137 downloads]$ cd ~/downloads
tar zxvf nagios-plugins-2.4.11.tar.gz
nagios-plugins-2.4.11/
nagios-plugins-2.4.11/build-aux/
nagios-plugins-2.4.11/build-aux/compile
nagios-plugins-2.4.11/build-aux/config.guess
nagios-plugins-2.4.11/build-aux/config.rpath
nagios-plugins-2.4.11/build-aux/config.sub
nagios-plugins-2.4.11/build-aux/install-sh
nagios-plugins-2.4.11/build-aux/ltmain.sh
nagios-plugins-2.4.11/build-aux/missing
nagios-plugins-2.4.11/build-aux/mkinstalldirs
nagios-plugins-2.4.11/build-aux/depcomp
nagios-plugins-2.4.11/build-aux/snippet/
```

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

```
nagiosexp9
[ec2-user@ip-172-31-80-137 downloads]$ cd nagios-plugins-2.4.11
./configure --with-nagios-user=nagios --with-nagios-group=nagios
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /usr/bin/mkdir -p
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking whether make supports nested variables... yes
checking whether to enable maintainer-specific portions of Makefiles... yes
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking whether gcc understands -c and -o together... yes
checking whether make supports the include directive... yes (GNU style)
```

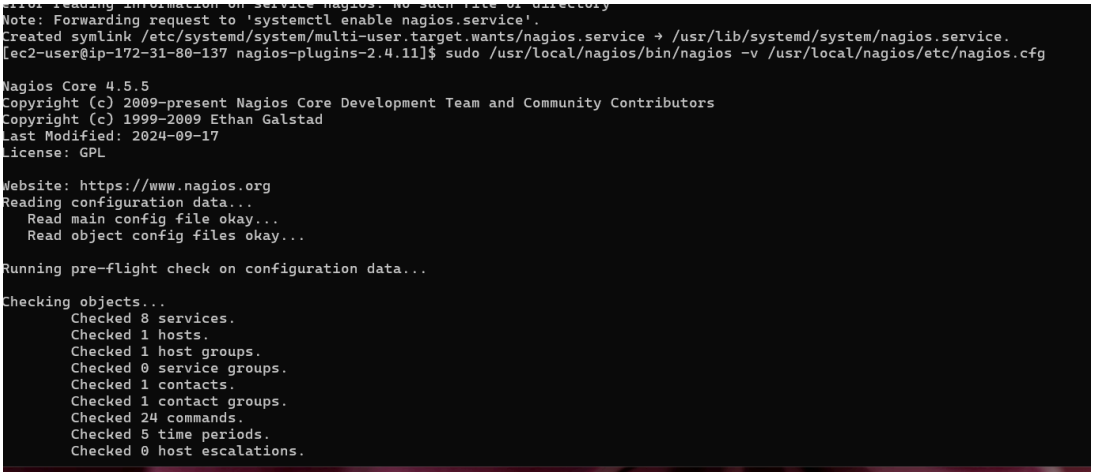
make
sudo make install

```
ake[1]: Leaving directory '/home/ec2-user/downloads/nagios-plugins-2.4.11/plugins-root'
aking install in po
ake[1]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11/po'
usr/bin/mkdir -p /usr/local/nagios/share
nstalling fr.gmo as /usr/local/nagios/share/locale/fr/LC_MESSAGES/nagios-plugins.mo
nstalling de.gmo as /usr/local/nagios/share/locale/de/LC_MESSAGES/nagios-plugins.mo
f test "nagios-plugins" = "gettext-tools"; then \
 /usr/bin/mkdir -p /usr/local/nagios/share/gettext/po; \
 for file in Makefile.in.in remove-potcdate.sin    Makevars.template; do \
   /usr/bin/install -c -o nagios -g nagios -m 644 ./$file \
                    /usr/local/nagios/share/gettext/po/$file; \
 done; \
 for file in Makevars; do \
   rm -f /usr/local/nagios/share/gettext/po/$file; \
 done; \
lse \
 : ; \
i
ake[1]: Leaving directory '/home/ec2-user/downloads/nagios-plugins-2.4.11/po'
ake[1]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11'
ake[2]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11'
ake[2]: Nothing to be done for 'install-exec-am'.
ake[2]: Nothing to be done for 'install-data-am'.
ake[2]: Leaving directory '/home/ec2-user/downloads/nagios-plugins-2.4.11'
ake[1]: Leaving directory '/home/ec2-user/downloads/nagios-plugins-2.4.11'
```

sudo chkconfig --add nagios
sudo chkconfig nagios on

```
[ec2-user@ip-172-31-80-137 nagios-plugins-2.4.11]$ sudo chkconfig --add nagios
sudo chkconfig nagios on
error reading information on service nagios: No such file or directory
Note: Forwarding request to 'systemctl enable nagios.service'.
Created symlink /etc/systemd/system/multi-user.target.wants/nagios.service → /usr/lib/systemd/system/nagios.service.
[ec2-user@ip-172-31-80-137 nagios-plugins-2.4.11]$
```

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

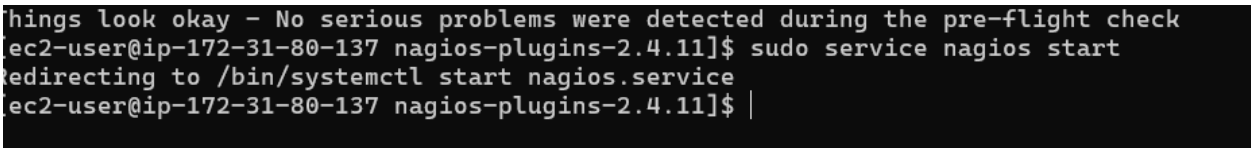


If this command is giving error! (Error in configuration file '/usr/local/nagios/etc/nagios.cfg' - Line 452 (Check result path '/usr/local/nagios/var/spool/checkresults' is not a valid directory)
Error processing main config file!**)**
The solution:
Create the missing directory, set the permissions, verify it.

sudo mkdir -p /usr/local/nagios/var/spool/checkresults (this is for creation)
sudo chown nagios:nagios /usr/local/nagios/var/spool/checkresults
sudo chmod 775 /usr/local/nagios/var/spool/checkresults (this is for permissions)

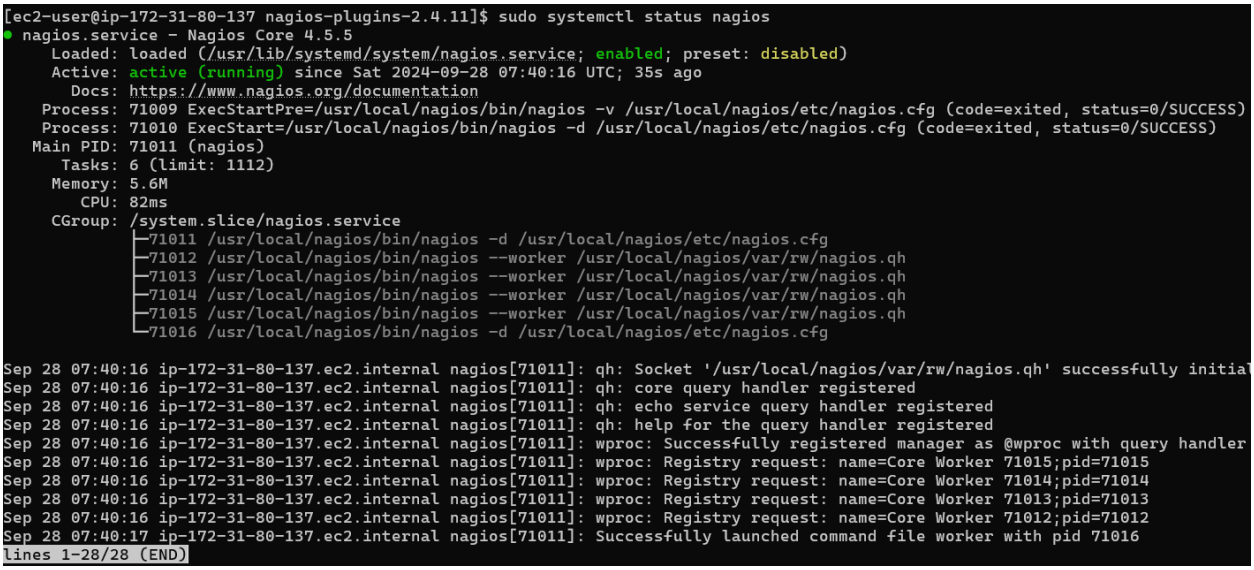
Now rerun the commmad (also given below) and continue:
sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

sudo service nagios start



---

sudo systemctl status nagios



(ignore if no error was found)
Again if this is giving an error then it is primarily because Nagios monitoring tool is unable to create or write to a temporary file in the "`/usr/local/nagios/var/`"

`To debug it lets start by checking the permissions:`
ls -ld /usr/local/nagios/var

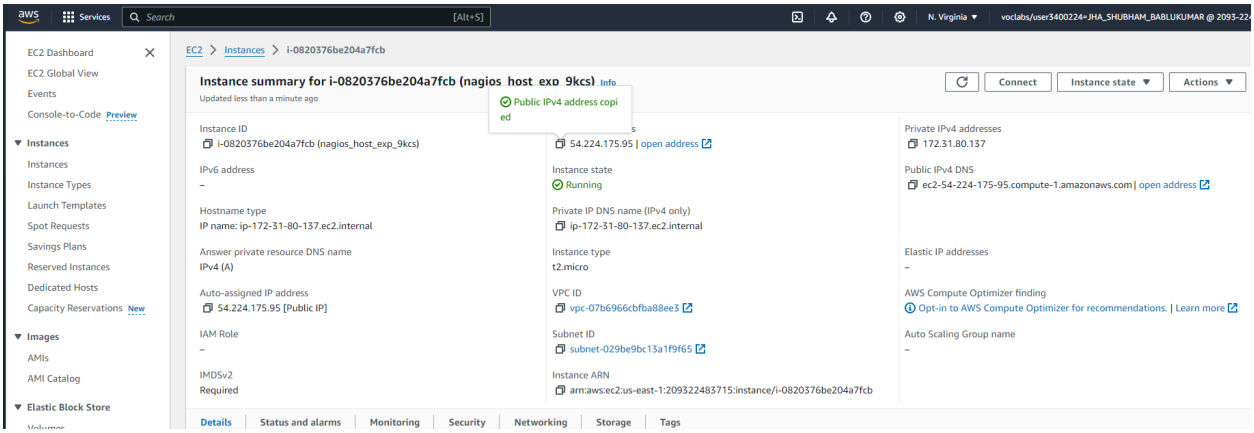
`Changing the ownership`
sudo chown -R nagios:nagios /usr/local/nagios/var

`Modify permissions`
sudo chmod -R 755 /usr/local/nagios/var

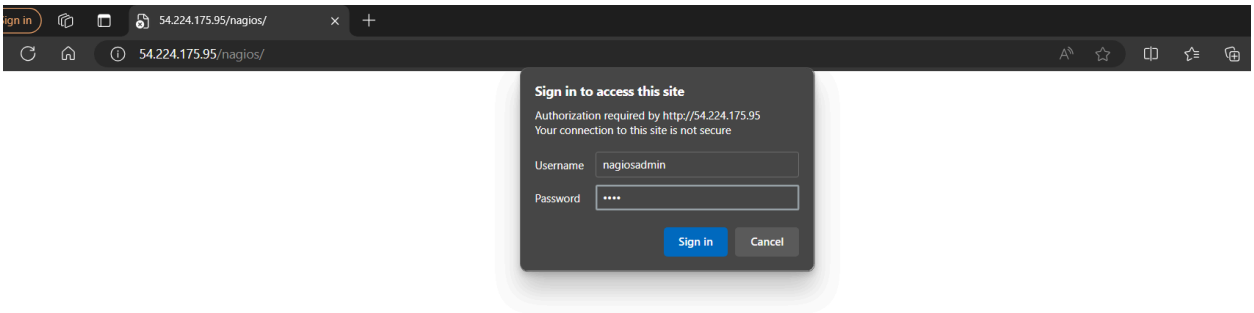
`Restart Nagios service`
sudo systemctl restart nagios

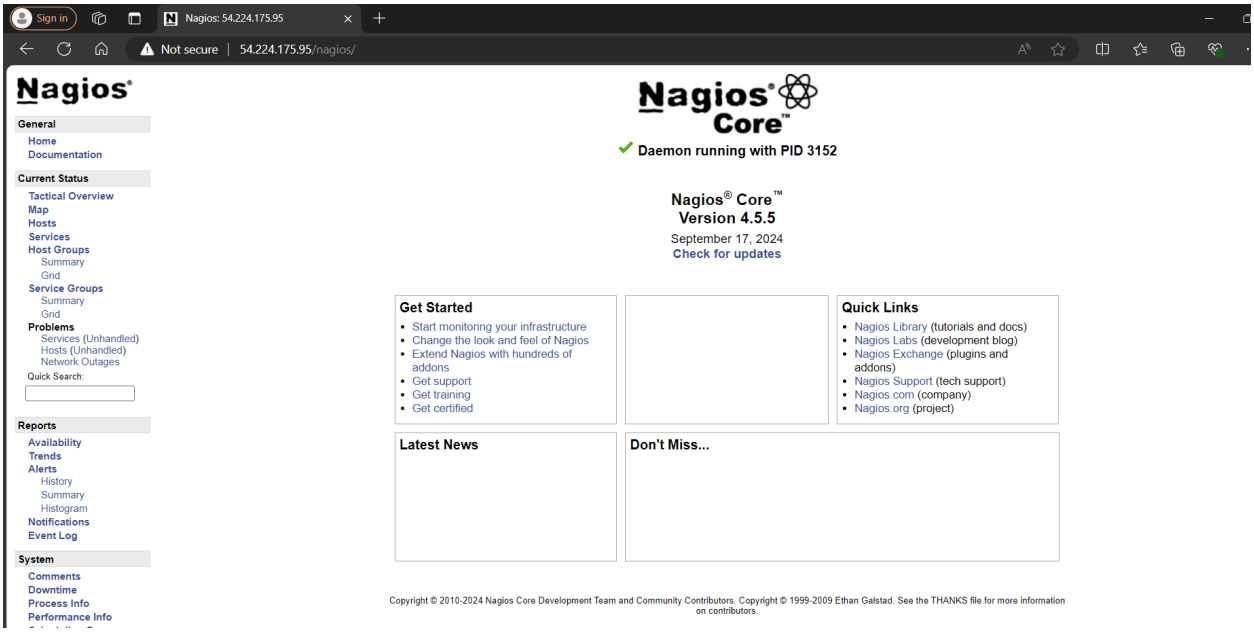
`check status of nagios,` `Rerun the command (the command which gave the recent error)`
sudo systemctl status nagios

Now, go to EC2 instance and click on instance id. Then, click on the copy icon just before the public ip address on public IP.



Enter the username password set above. (in the section of adding password for nagios admin)





Conclusion:

Setting up Nagios on an EC2 instance was a rewarding yet challenging experience for me. I began by launching an instance using the default operating system and configuring it to monitor my network. The installation process went smoothly at first; I installed essential packages, created users, and configured Nagios as planned.

However, I encountered a few hurdles along the way. One significant issue arose when the Apache server was not running, which prevented me from accessing the Nagios web interface. After some troubleshooting, I realized that restarting the Apache service was necessary to resolve this.

Additionally, I faced permission issues that initially hindered Nagios from creating or writing to temporary files. By checking the ownership and permissions of the necessary directories, I managed to address this issue effectively.

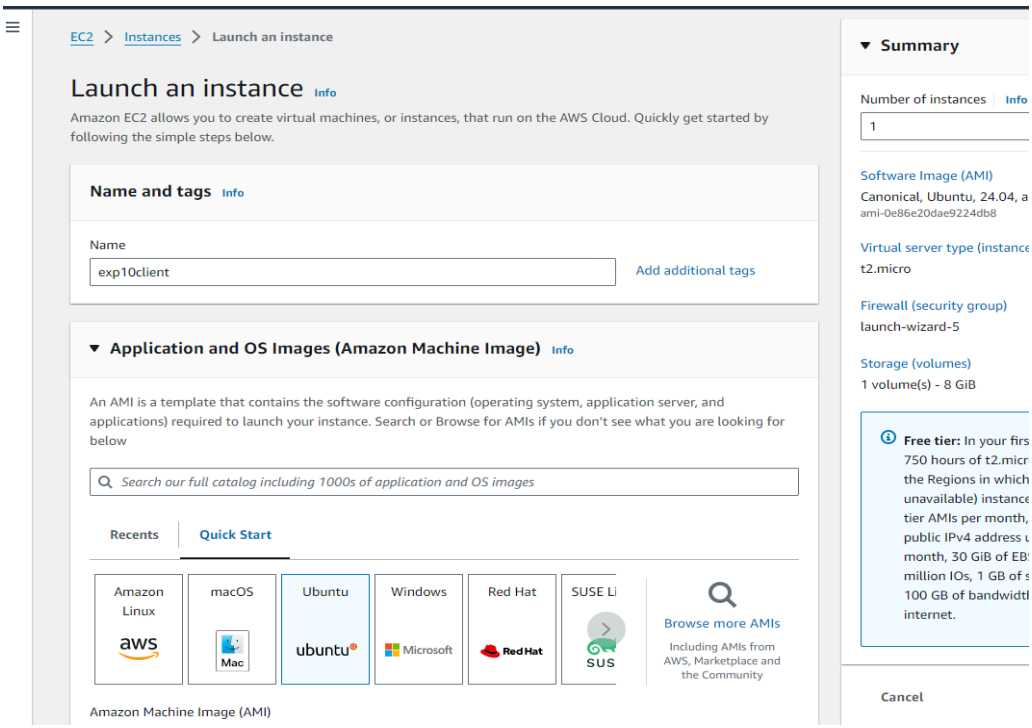
Here in the document nagios host machine (interchangibly also referred as exp9 machine or host machine) refers to the instance which was connected to the terminal in previous experiment.
(so if the previous instance was closed do connect with that instance and run the httpd status command to check whether the apache server was closed. if its closed run the start httpd command (google it or use ctrl+f to search for the key word in previous doc).)
And the client machine refers to the machine created just for this experiment.

Steps

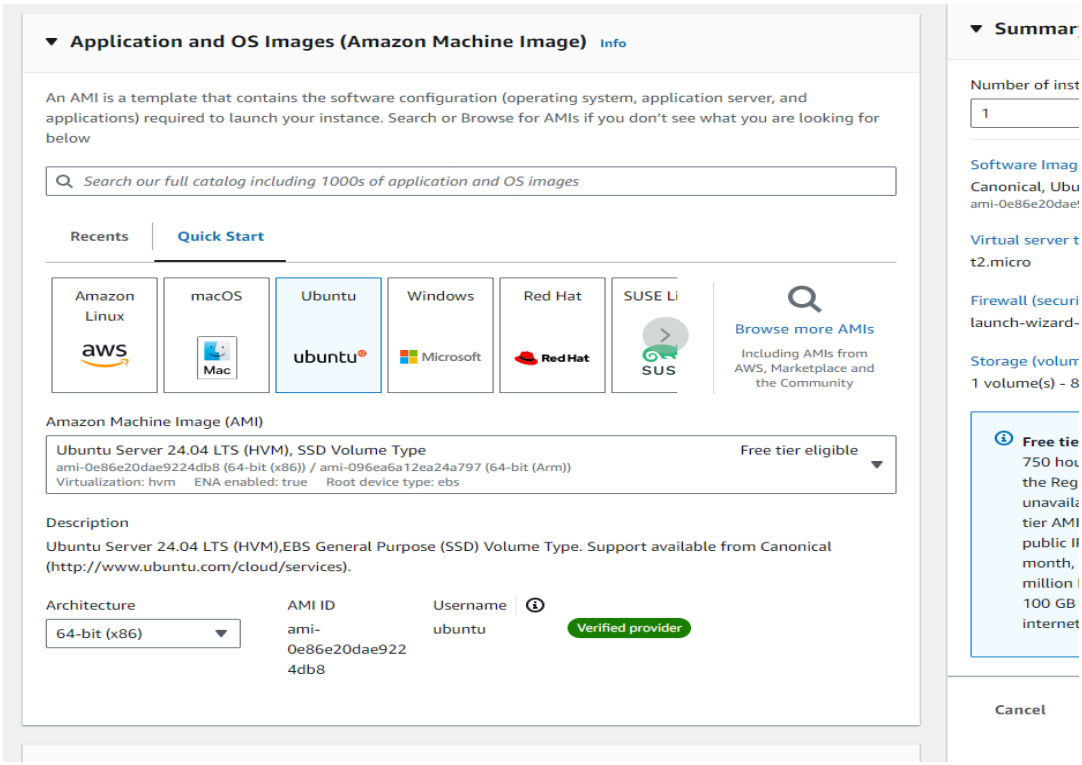
1) Launch an instance

Launch an ec2 instance.

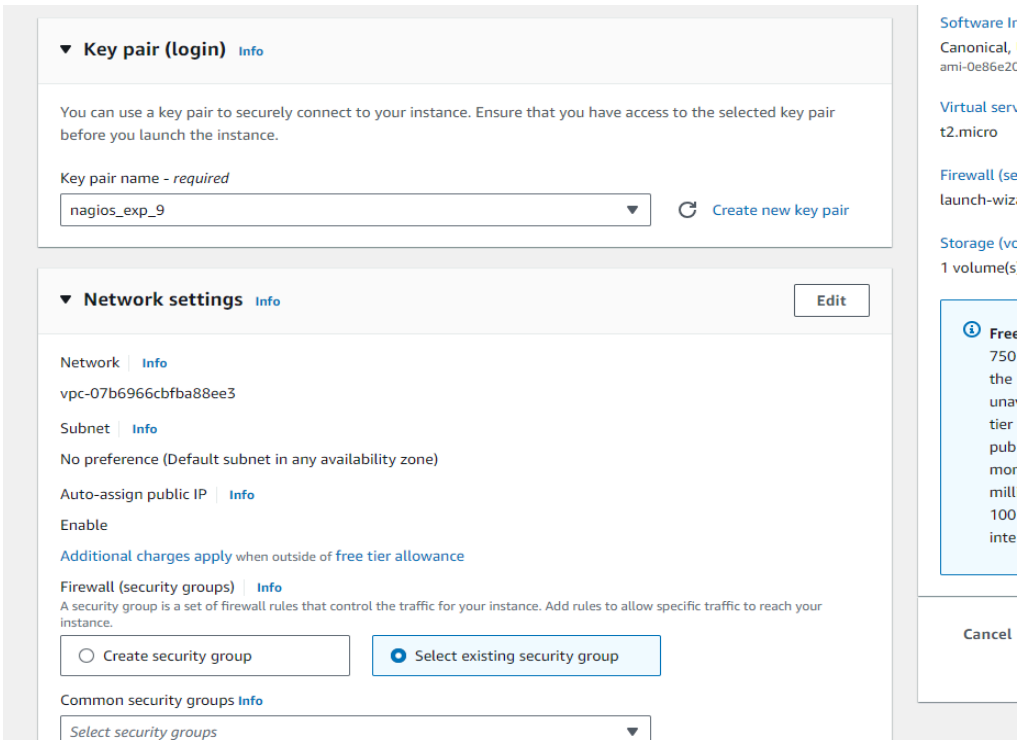
`Select Ubuntu as the os give a meaningful name of the instance.`



`Select the same security group as given to the exp9 machine.`

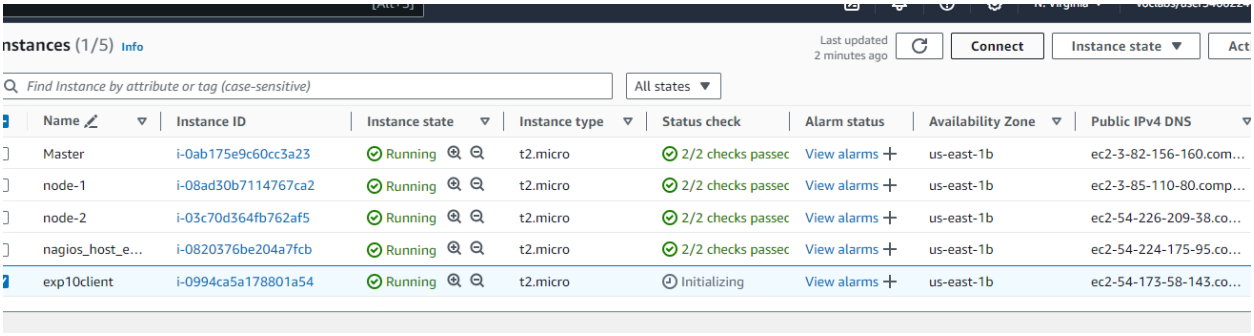


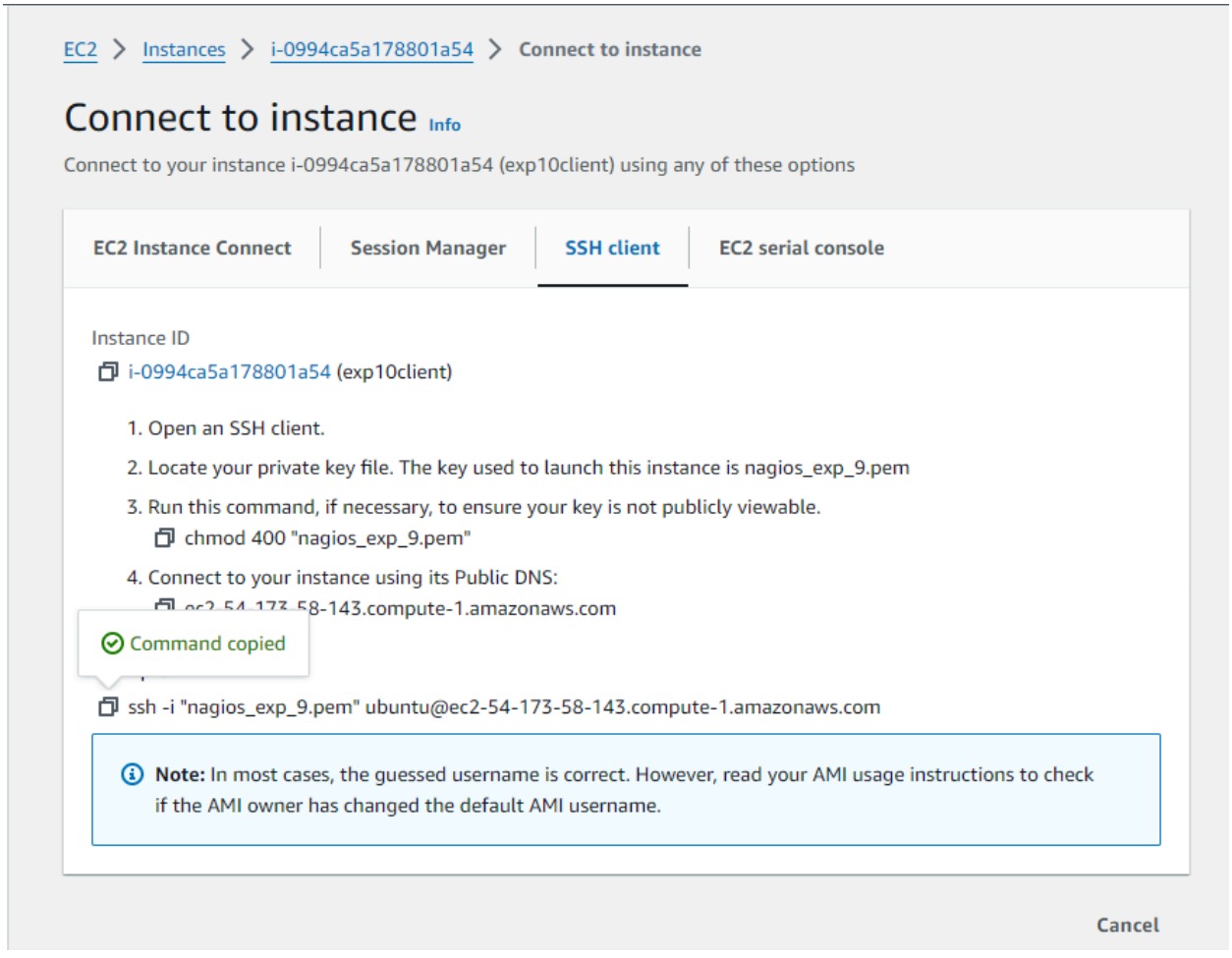
Make sure to select the same key-pair login used in the exp9 machine.



click on launch instance.

Now connect with this client machine using the ssh through your terminal(open a new terminal in your local machine and we will need both of the terminals open)





Note to change the path of the .pem file.

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

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

PS C:\Users\Lenovo> ssh -i "C:\Users\Lenovo\Downloads\nagios_exp_9.pem" ubuntu@ec2-54-173-58-143.compute-1.amazonaws.com

The authenticity of host 'ec2-54-173-58-143.compute-1.amazonaws.com (54.173.58.143)' can't be established.
ED25519 key fingerprint is SHA256:IA3XH7fO11spKO84wDcZFmqRgNn0iJZ7itI2pBMmHP4.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-54-173-58-143.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Sat Sep 28 10:43:28 UTC 2024

  System load:  0.01             Processes:             107
  Usage of /:   22.8% of 6.71GB  Users logged in:       0
  Memory usage: 19%              IPv4 address for enX0: 172.31.82.77
```

2) Go to nagios host machine (Host machine)

Perform the following commands

ps -ef | grep nagios

```
Host  CLient
[ec2-user@ip-172-31-80-137 ~]$ ps -ef | grep nagios
nagios      3152       1  0 08:36 ?        00:00:00 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
nagios      3153    3152  0 08:36 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios      3154    3152  0 08:36 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios      3155    3152  0 08:36 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios      3156    3152  0 08:36 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios      3160    3152  0 08:36 ?        00:00:00 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
ec2-user   11528    2972  0 10:44 pts/0    00:00:00 grep --color=auto nagios
[ec2-user@ip-172-31-80-137 ~]$
```

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

```
[root@ip-172-31-80-137 ec2-user]# mkdir -p /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
[root@ip-172-31-80-137 ec2-user]# ls
```

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

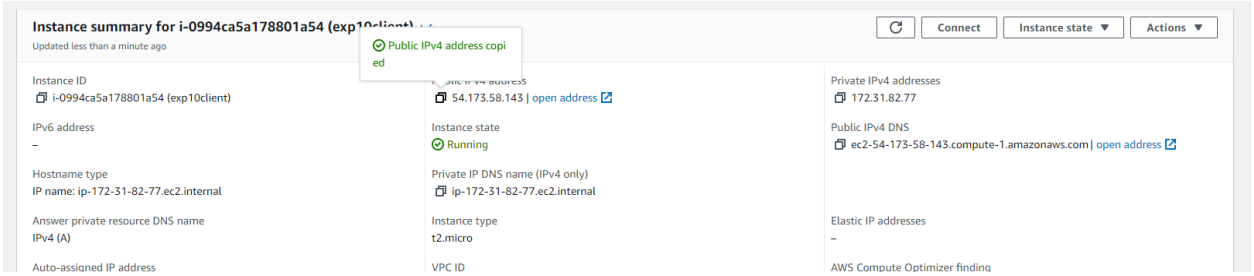
```
[root@ip-172-31-80-137 ec2-user]# cp /usr/local/nagios/etc/objects/localhost.cfg /usr/local/nagios/etc/objects/monitorhosts/linuxho
sts/linuxserver.cfg
```

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

```
[root@ip-172-31-80-137 ec2-user]# nano /usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg
```

Change hostname and alias to linuxserver
Change address to public ip address of client instance (Ubuntu instance) you can get

the ip address by clicking on the instance id on the instances section there you will get the public ipv4 address




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

# Define a host for the local machine

define host {

    use                     linux-server            ; Name of host template to use
                                                    ; This host definition will in>
                                                    ; in (or inherited by) the lin>
    host_name               linuxserver
    alias                   linuxserver
    address                 54.173.58.143
}
```

Change hostgroup_name to linux-servers1

```
# Define an optional hostgroup for Linux machines

define hostgroup {

    hostgroup_name          linux-servers1          ; The name of the hostgroup
    alias                   Linux Servers           ; Long name of the group
    members                 localhost               ; Comma separated list of host>
}
```

Change the occurrences of hostname further in the document from localhost to linuxserver example like:

```
    host_name                       localhost
```

changed to

```
define service {

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

This is the last one

```
define service {

    use                     local-service           ; Name of service template to >
    host_name               linuxserver
    service_description     HTTP
    check_command           check_http
    notifications_enabled   0
```

now ctrl+O and enter to save and then ctrl+X for exiting.

Open nagios configuration file and add the line shown below
nano /usr/local/nagios/etc/nagios.cfg

```
[root@ip-172-31-80-137 ec2-user]# nano /usr/local/nagios/etc/nagios.cfg
```

##Add this line below the opened nano interface where similar lines are commented.
cfg_dir=/usr/local/nagios/etc/objects/monitorhosts/

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

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

# Definitions for monitoring the local (Linux) host
cfg_file=/usr/local/nagios/etc/objects/localhost.cfg

# Definitions for monitoring a Windows machine
#cfg_file=/usr/local/nagios/etc/objects/windows.cfg

# Definitions for monitoring a router/switch
#cfg_file=/usr/local/nagios/etc/objects/switch.cfg

# Definitions for monitoring a network printer
#cfg_file=/usr/local/nagios/etc/objects/printer.cfg


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

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



# OBJECT CACHE FILE
# This option determines where object definitions are cached when
# Nagios starts/restarts.  The CGIs read object definitions from
```

ctrl+o and enter for saving and ctrl+x to exit nano editor.

`Verify configuration files`

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

```
[root@ip-172-31-80-137 ec2-user]# /usr/local/nagios/bin/nagios -v /usr/local/nagios
/etc/nagios.cfg

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

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

Running pre-flight check on configuration data...

Checking objects
```

```
        Checked 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-172-31-80-137 ec2-user]#
```

Restart nagios service.

service nagios restart

```
Things look okay - No serious problems were detected during the pre-flight check
[root@ip-172-31-80-137 ec2-user]# service nagios restart
Redirecting to /bin/systemctl restart nagios.service
[root@ip-172-31-80-137 ec2-user]#
```

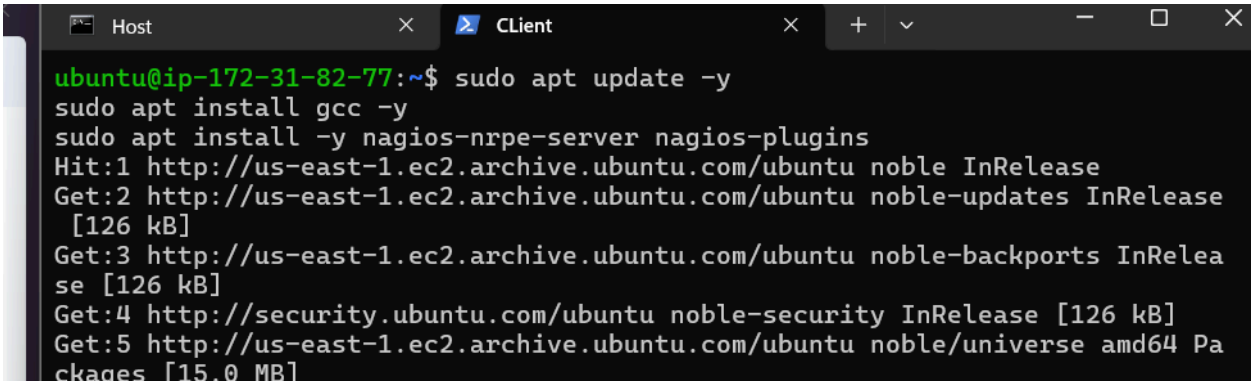
3) Go to client machine (ubuntu machine)

Perform the following commands

sudo apt update -y

sudo apt install gcc -y

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

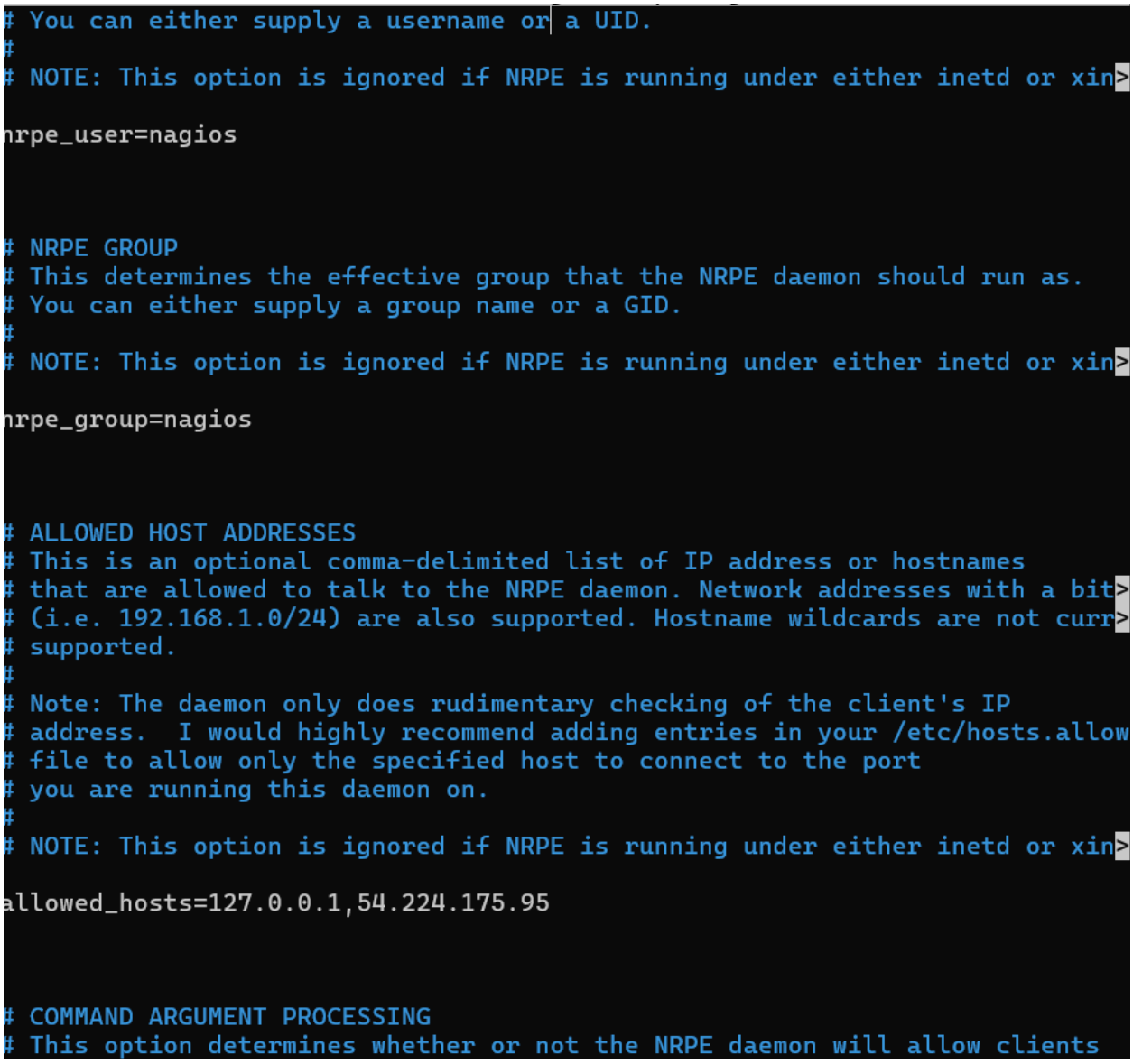


```
Running kernel seems to be up-to-date.

Restarting services...

Service restarts being deferred:
 /etc/needrestart/restart.d/dbus.service
 systemctl restart getty@tty1.service
 systemctl restart networkd-dispatcher.service
 systemctl restart serial-getty@ttyS0.service
 systemctl restart systemd-logind.service
 systemctl restart unattended-upgrades.service

No containers need to be restarted.

User sessions running outdated binaries:
 ubuntu @ session #1: sshd[990,1101]
 ubuntu @ user manager service: systemd[996]

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-82-77:~$
```

Open the nrpe.cfg file in nano editor

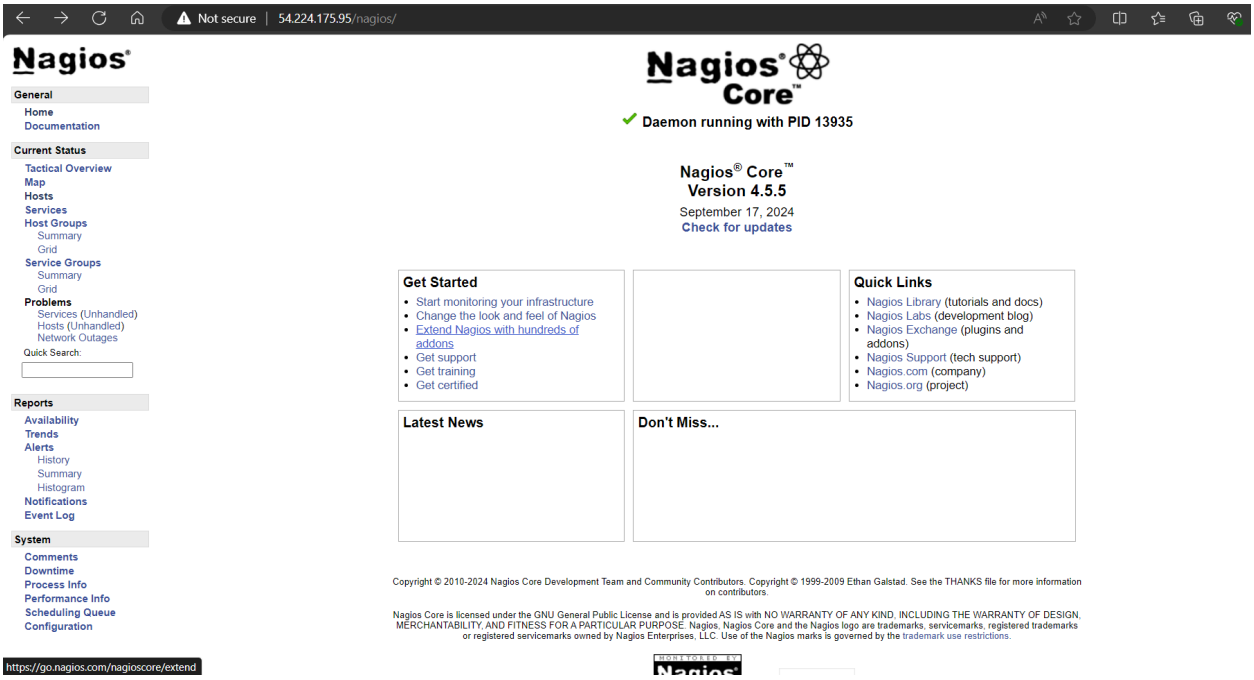
sudo nano /etc/nagios/nrpe.cfg

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

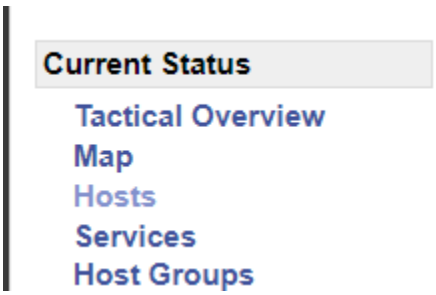


again save and exit the nano editor.

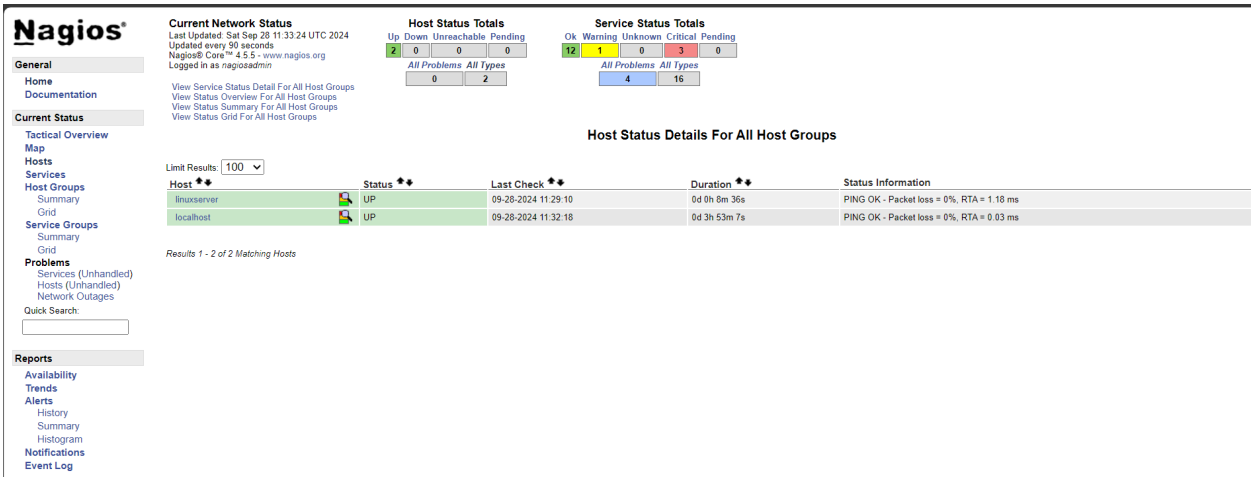
4) Go to nagios dashboard and click on hosts

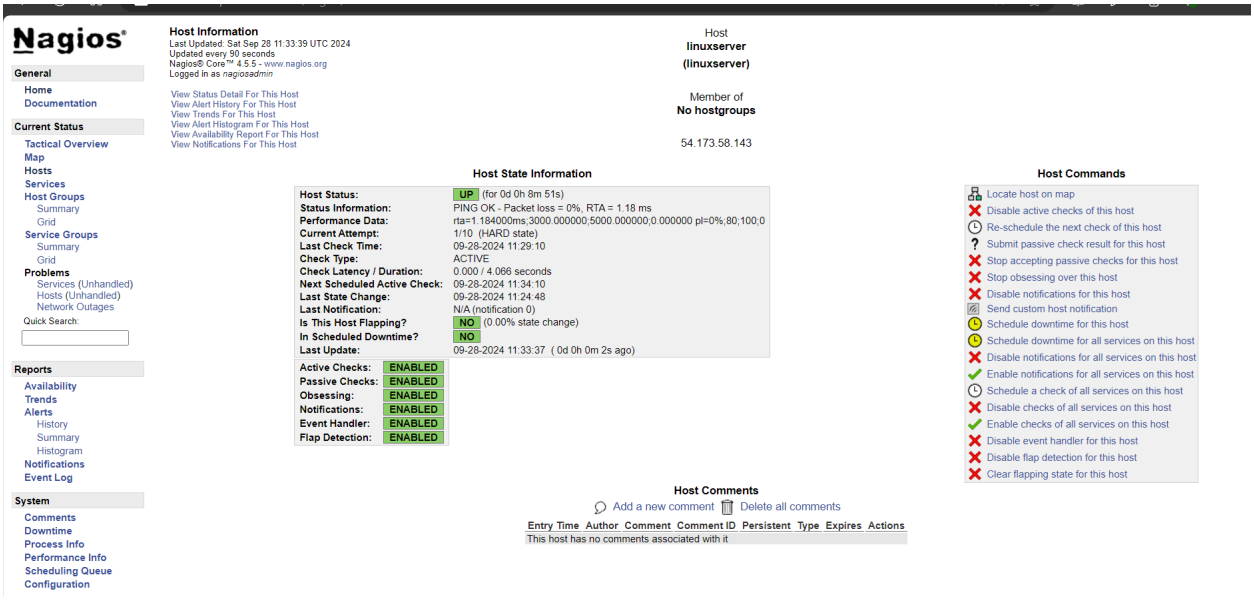


Click on hosts

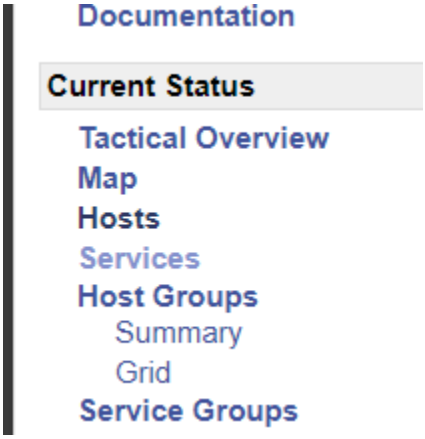


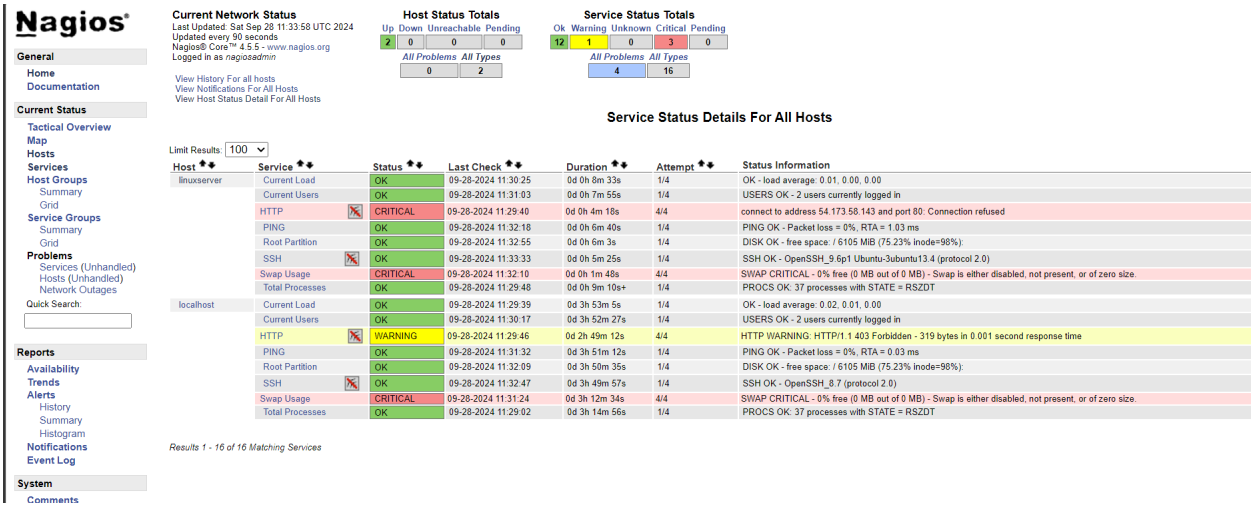
5) Click on linux server





6) Click on nagios services





Conclusion:

In this lab, we successfully configured a monitoring setup between a Nagios host machine (referred to as "exp9 machine") and a client machine (created specifically for this experiment). The goal was to set up Nagios to monitor a remote Linux server, which involved configuring both the Nagios host and client machine (Ubuntu instance) in an EC2 environment.

We started by launching an Ubuntu EC2 instance as the client machine, ensuring that we used the same security group and key-pair as the Nagios host machine to maintain consistent access and permissions. After establishing SSH connections to both machines, we worked in parallel, using one terminal for the host and another for the client.

On the Nagios host machine, we created a new directory structure, then copied and modified the `localhost.cfg` file to set up a configuration for monitoring the remote client machine. This included specifying the public IP address of the client

machine and updating the hostgroup and hostname. After editing the Nagios configuration file to recognize the new monitoring host directory, we verified the changes and restarted the Nagios service.

On the client machine, we installed the necessary Nagios packages (`nagios-nrpe-server` and `nagios-plugins`), configured the `nrpe.cfg` file, and allowed communication between the Nagios host and client by updating the allowed_hosts configuration.

After these steps, we were able to successfully monitor the remote Linux server from the Nagios dashboard, confirming that our setup was correct. This experiment demonstrated the core concepts of configuring Nagios to monitor a remote machine, providing practical insight into network monitoring and server management in a real-world scenario.

1.

Adv DevOps Exp-11

Aim:

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

Theory:

AWS Lambda

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

Lambda Workflow:

● **Create a Function**: Write the function code and define its handler (entry point). You can use the AWS Console, CLI, or upload a deployment package.
● **Set Event Sources**: Define how the function is triggered (e.g., when an object is uploaded to S3 or a DynamoDB table is updated).
● **Execution**: When triggered, Lambda runs your function, executes the logic, and automatically scales to handle the incoming event volume.
● **Scaling and Concurrency**: Lambda scales automatically by launching more instances of the function to handle simultaneous invocations. There are also options for configuring reserved

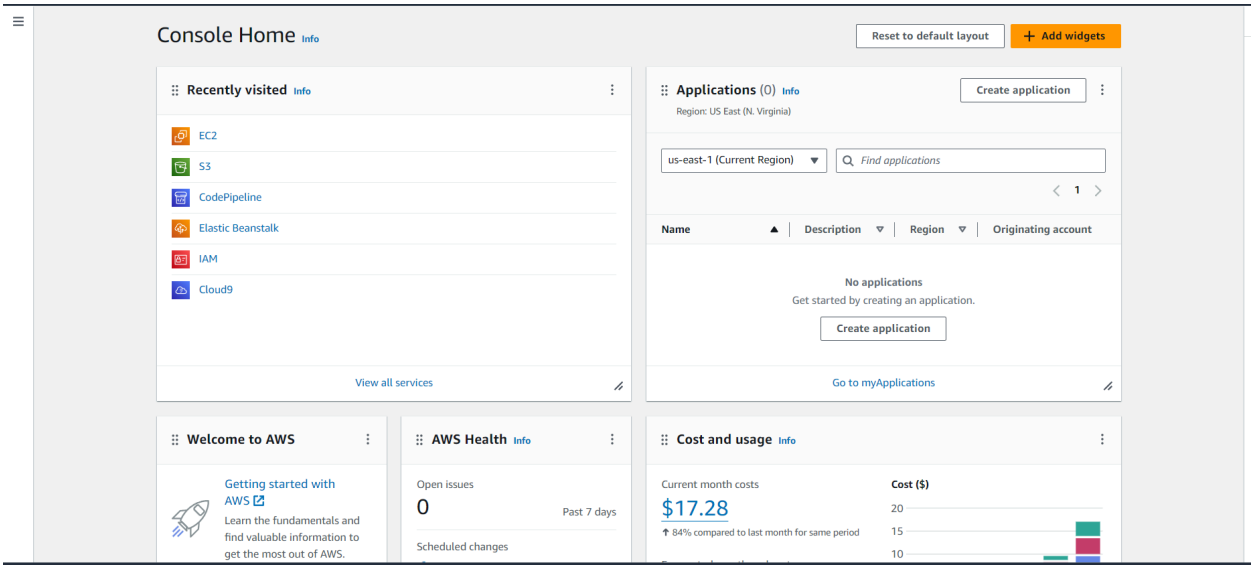
concurrency to manage traffic.
• **Monitoring and Logging**: Lambda integrates with Amazon CloudWatch for logging and monitoring. Logs for each invocation are sent to CloudWatch, allowing you to track performance and troubleshoot errors.

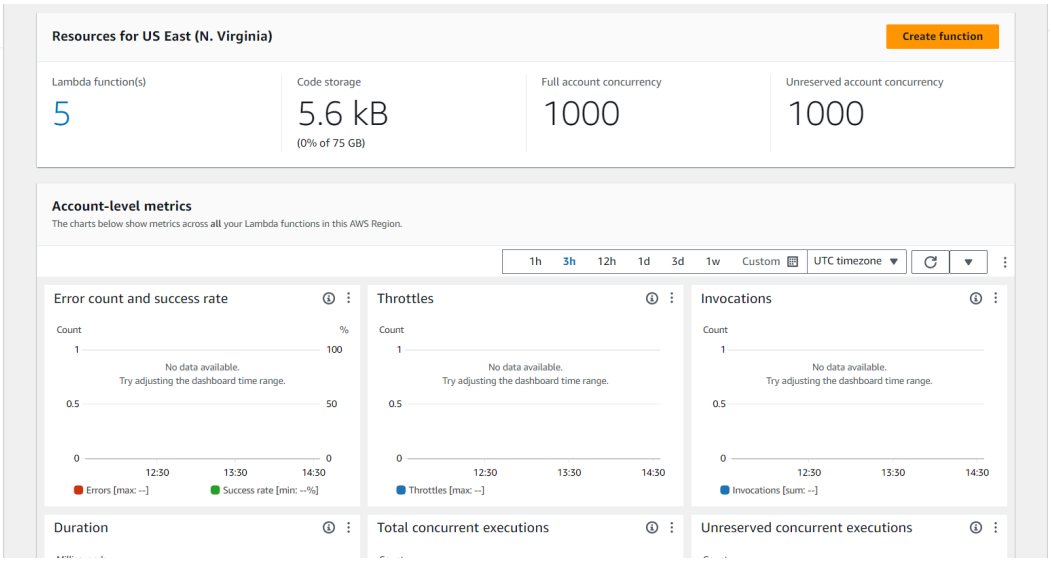
AWS Lambda Functions:

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

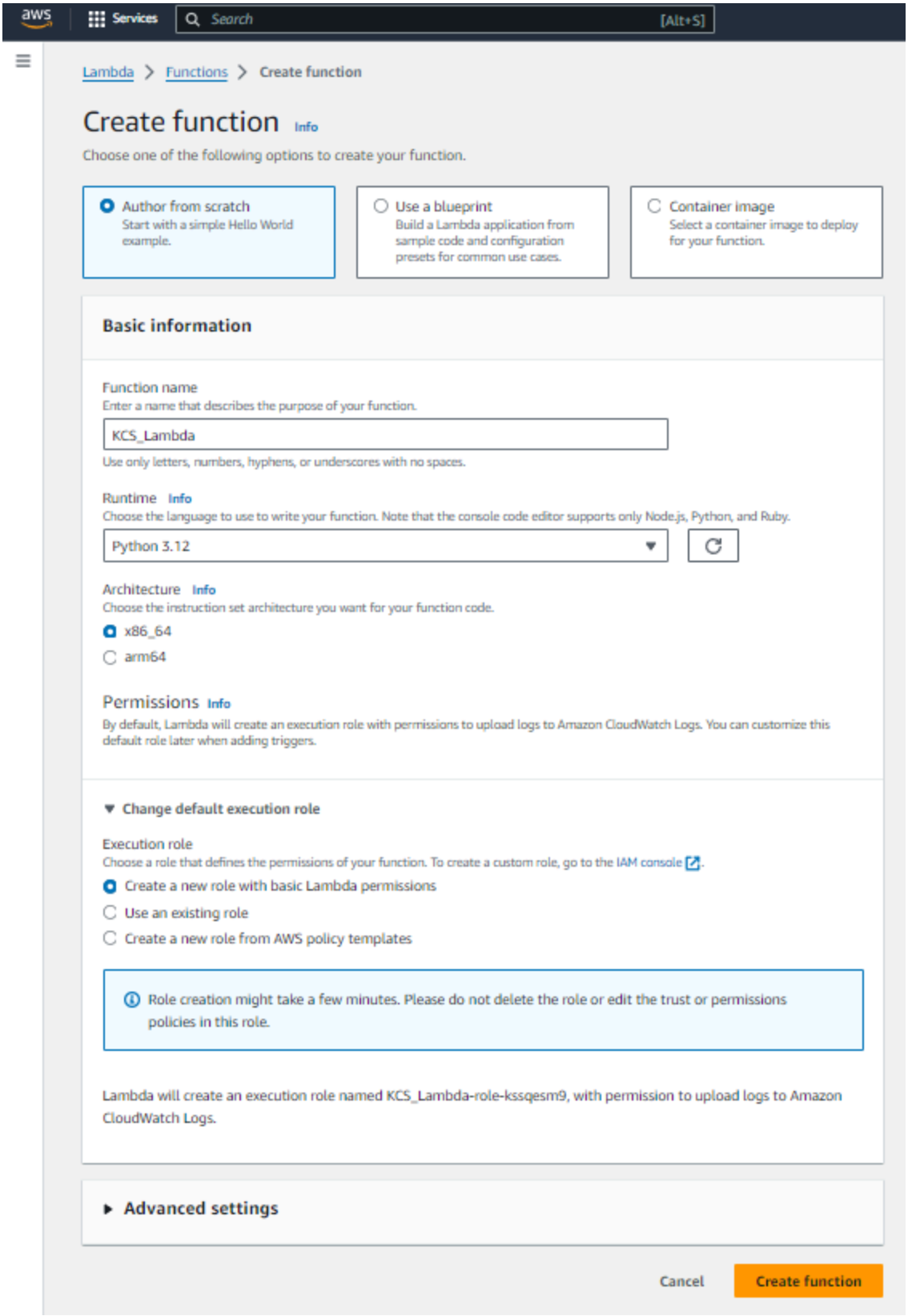
Steps To create the lambda function:

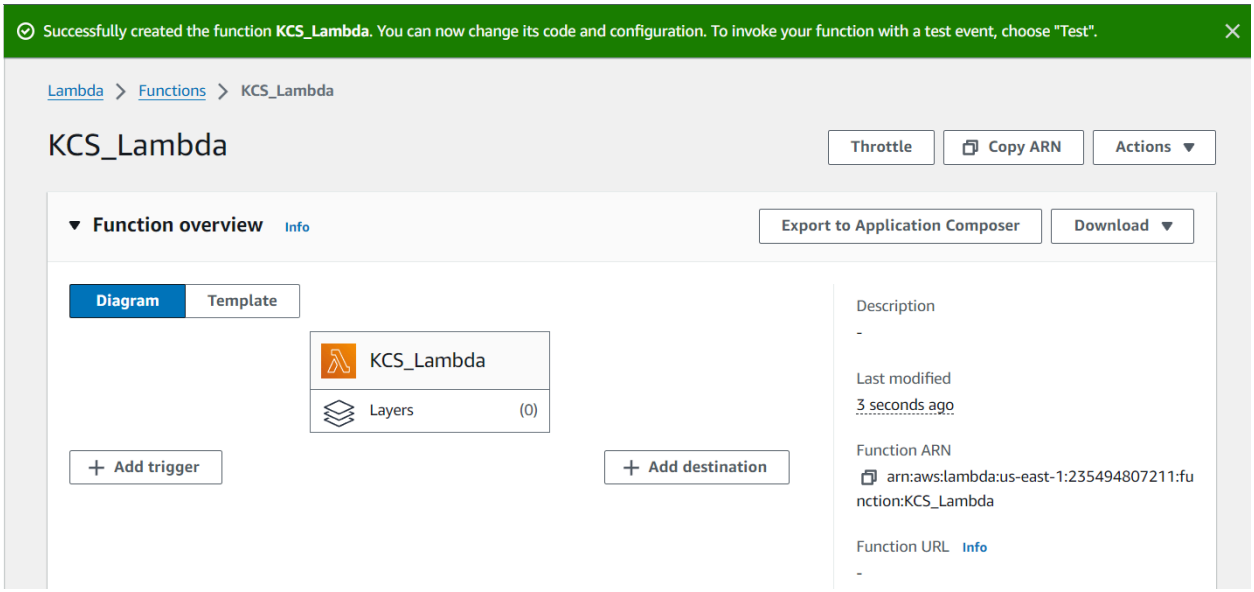
Step 1: Login to your AWS Personal/Academy Accout.Open lambda and click on create function button.

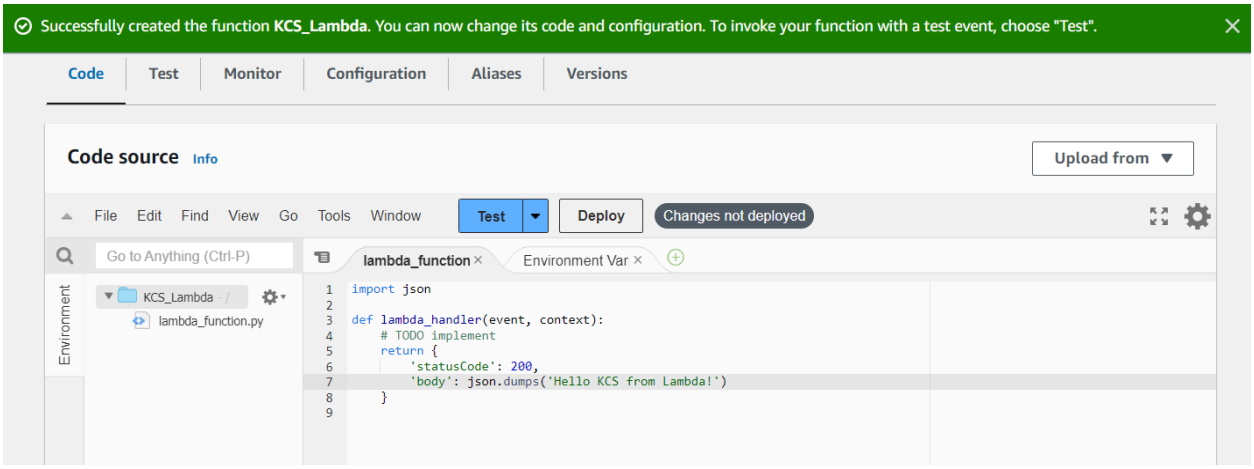




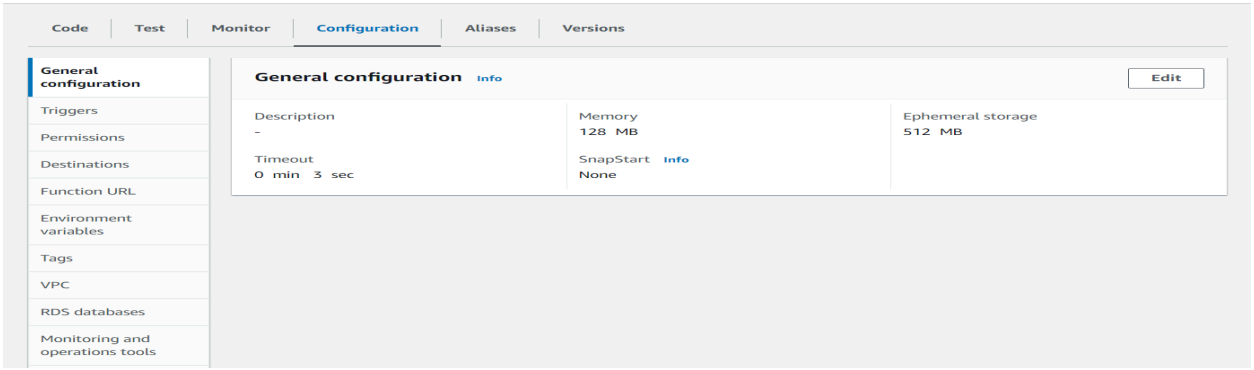
Step 2: Now Give a name to your Lambda function, Select the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby. So will select Python 3.12 , Architecture as x86, and Execution role to Create a new role with basic Lambda permissions.


aws
Services
Search
[Alt+S]
Lambda > Functions > Create function
Create function Info
Choose one of the following options to create your function.
Author from scratch
Start with a simple Hello World example.
Use a blueprint
Build a Lambda application from sample code and configuration presets for common use cases.
Container image
Select a container image to deploy for your function.
Basic information
Function name
Enter a name that describes the purpose of your function.
KCS_Lambda
Use only letters, numbers, hyphens, or underscores with no spaces.
Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Python 3.12
Architecture Info
Choose the instruction set architecture you want for your function code.
x86_64
arm64
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 KCS_Lambda-role-kssqesm9, with permission to upload logs to Amazon CloudWatch Logs.
Advanced settings
Cancel
Create function

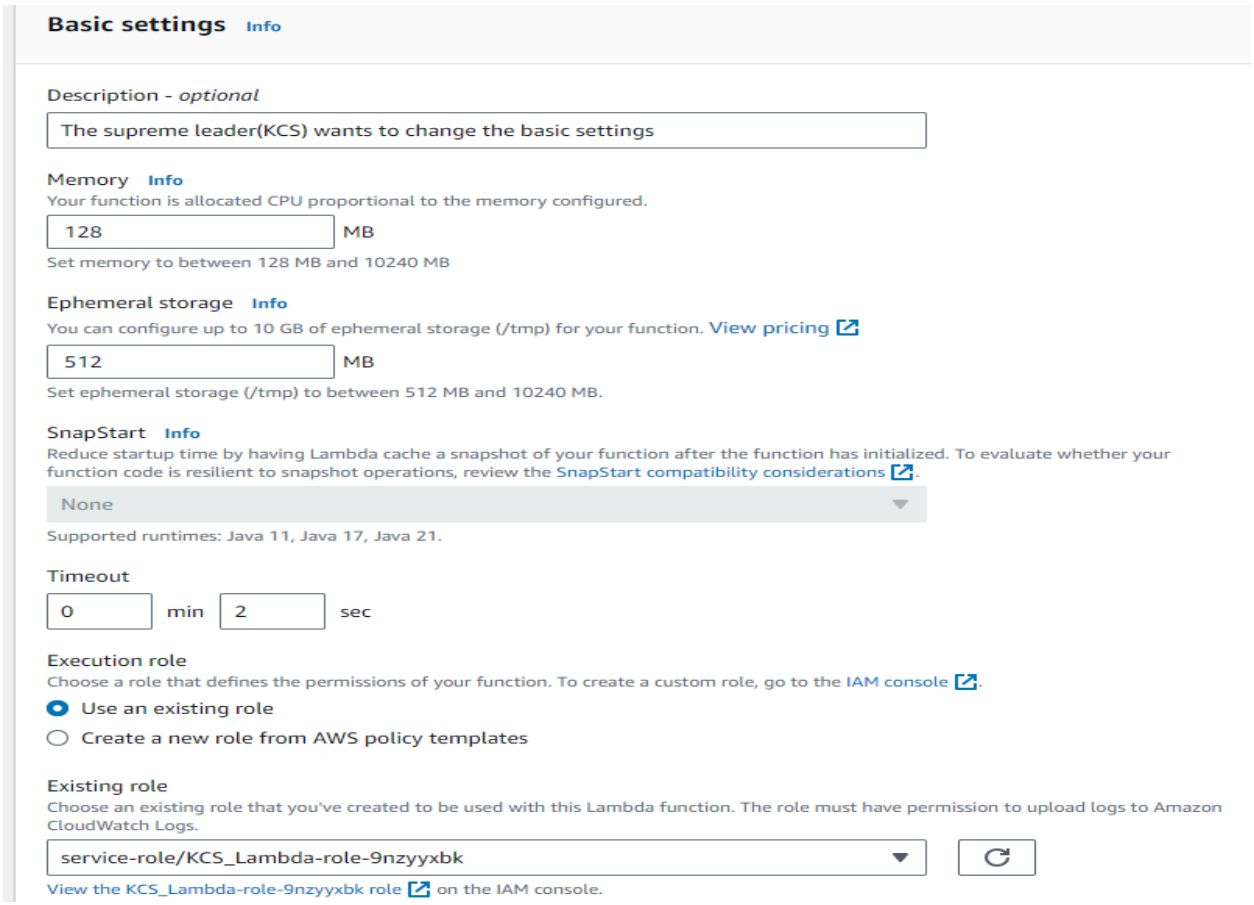


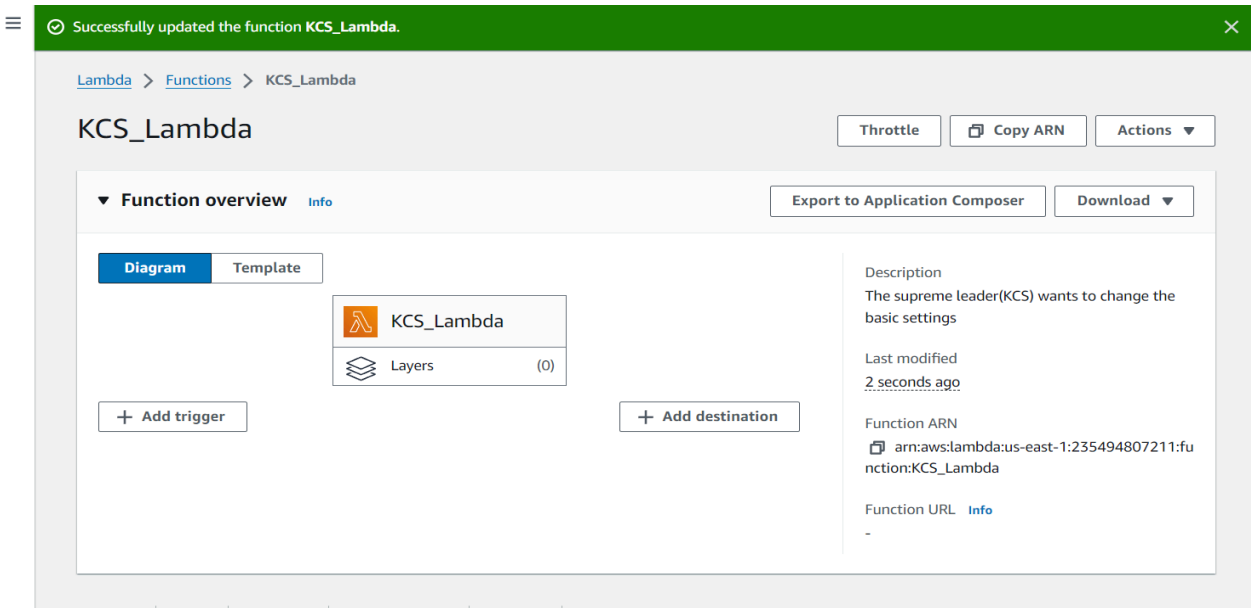


To See or Edit the basic settings go to configuration then click on edit general configuration.

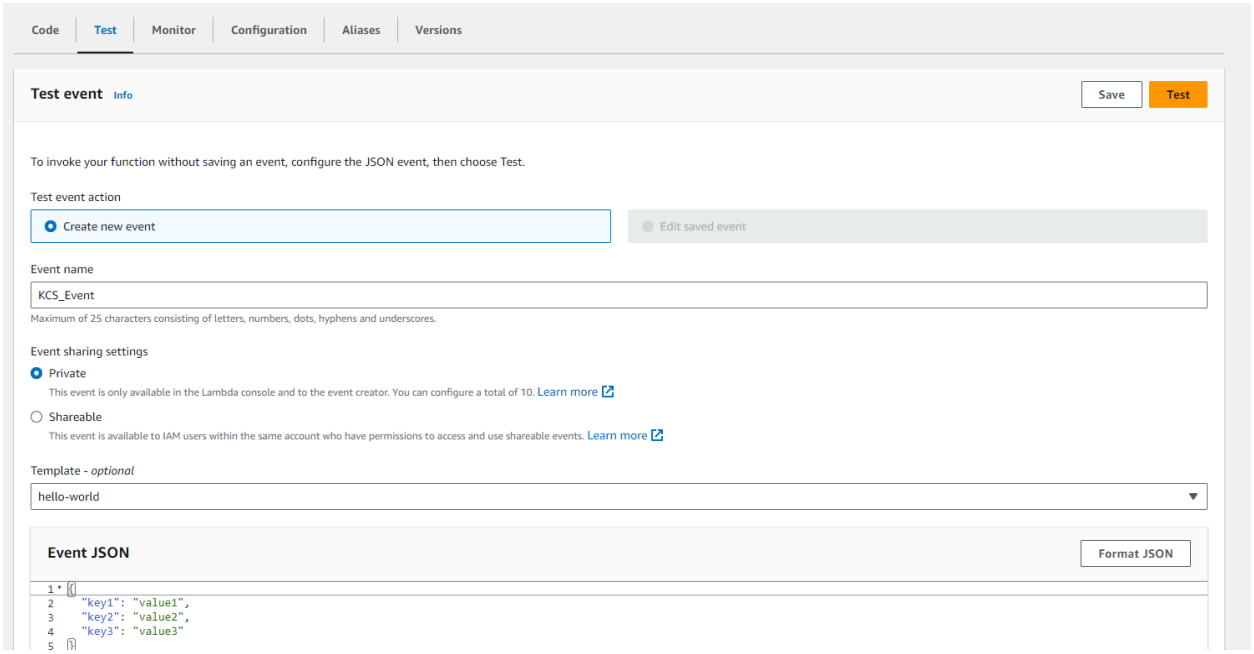


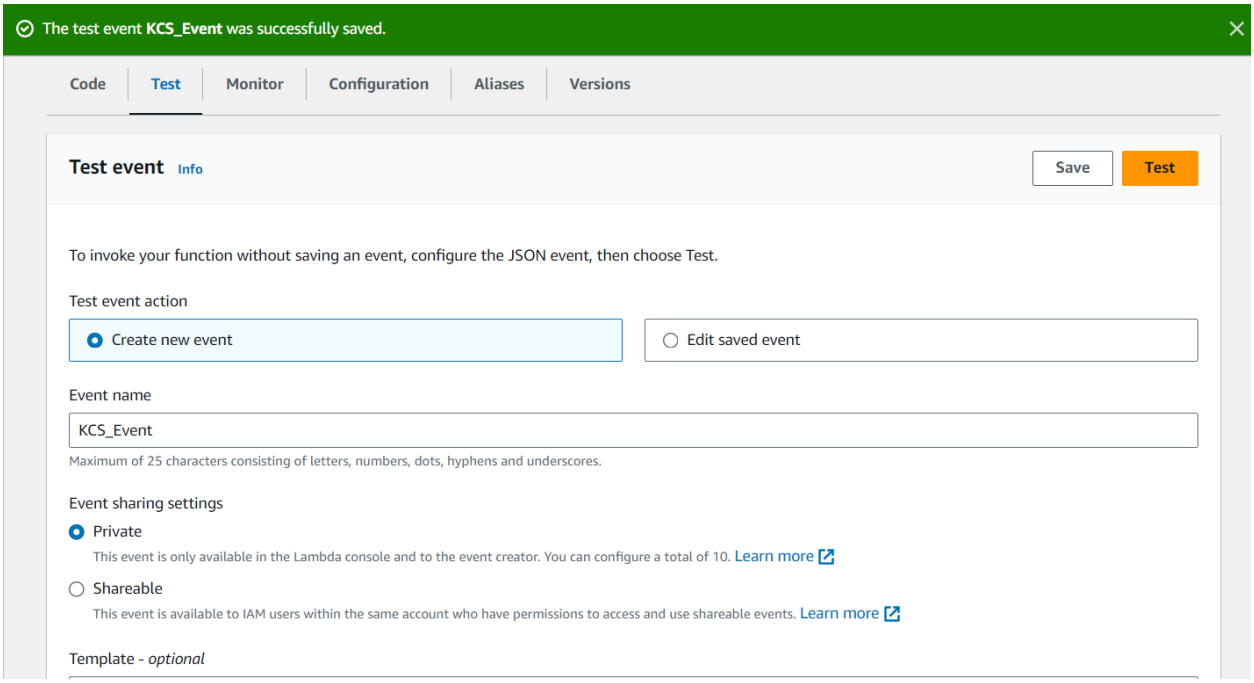
Here, you can enter a description and change Memory and Timeout. I've changed the Timeout period to 2 sec since that is sufficient for now.



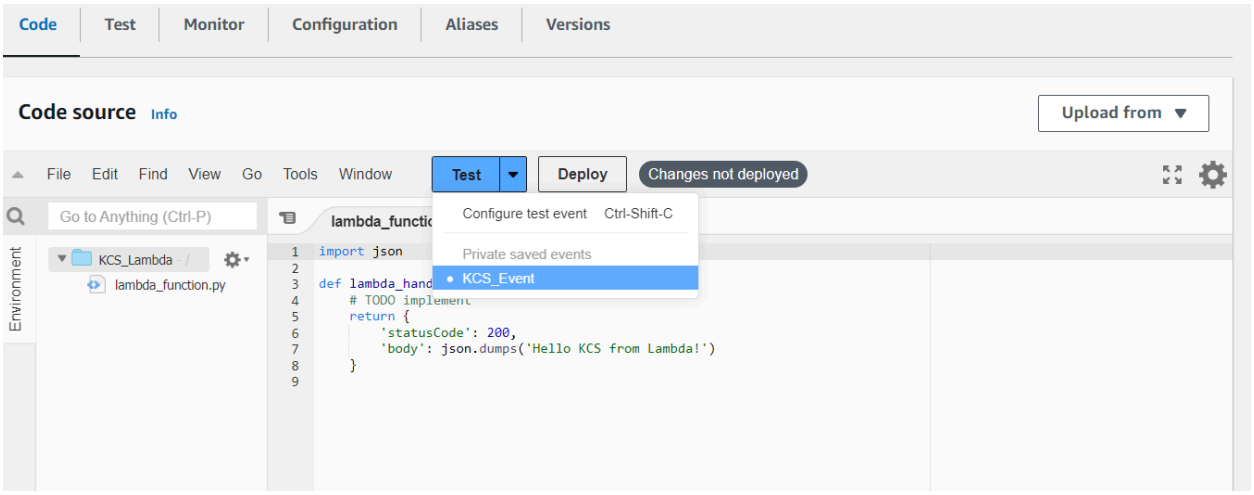


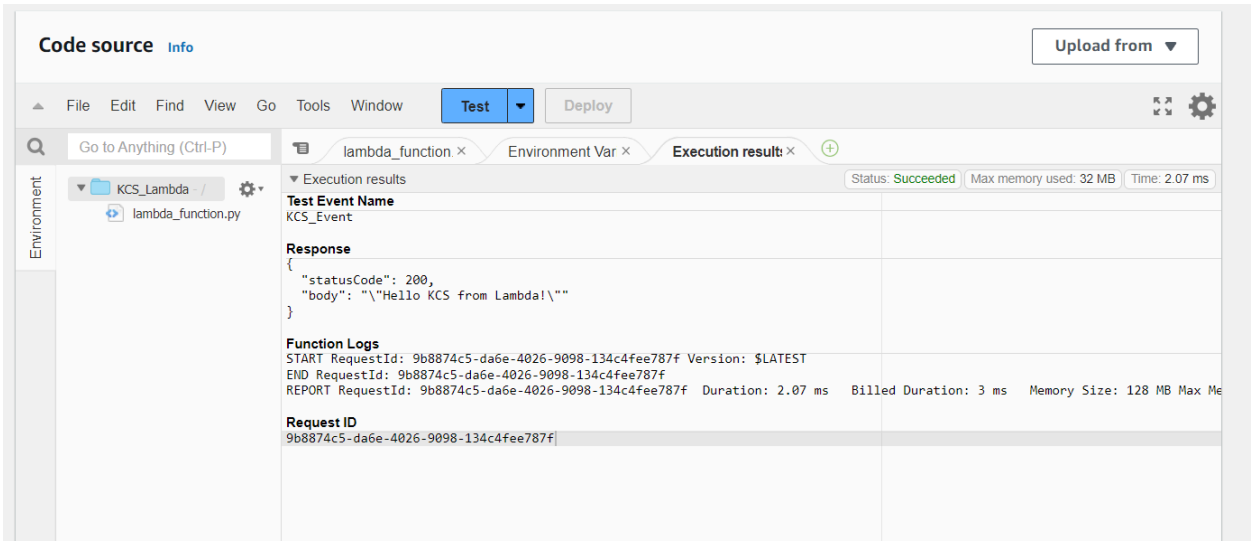
Step 3: Now Click on the Test tab then select Create a new event, give a name to the event and select Event Sharing to private, and select hello-world template.

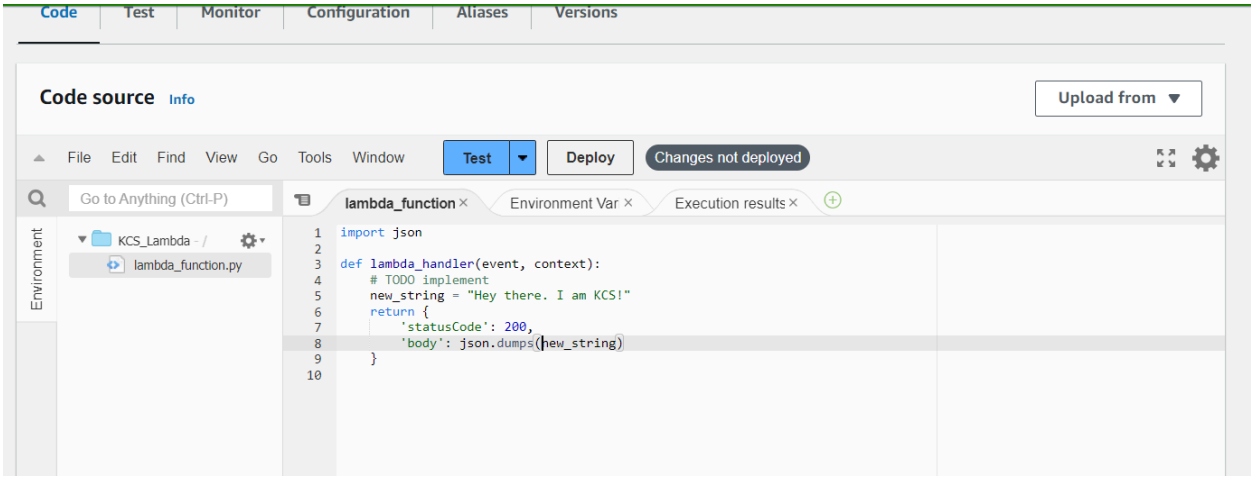




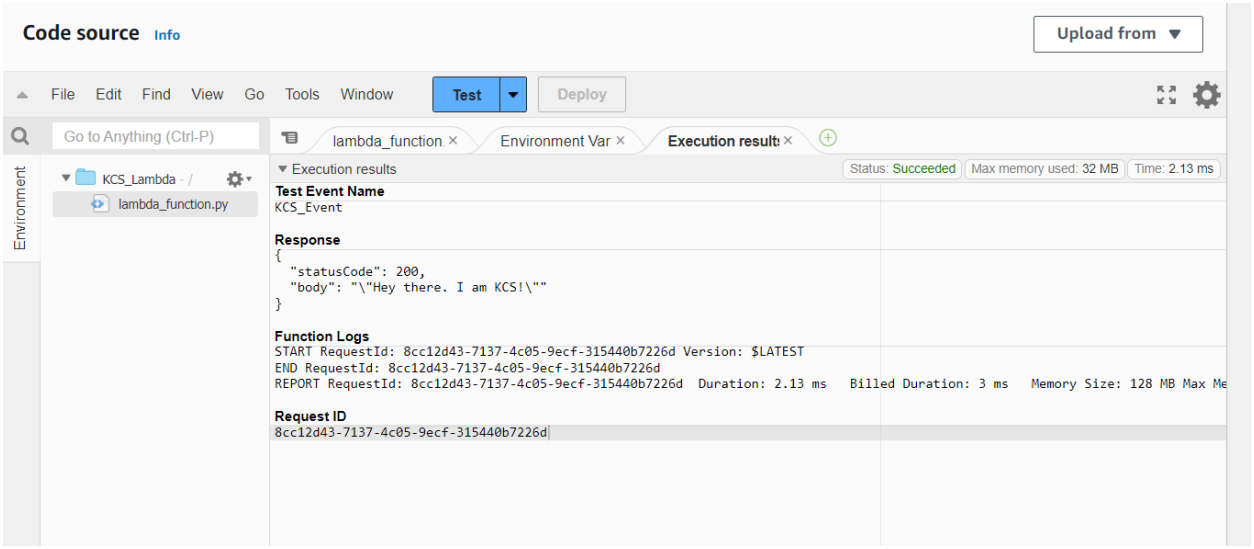
Step 4: Now In Code section select the created event from the dropdown of test then click on test . You will see the below output.







Now ctrl+s to save and click on deploy to deploy the changes



You can see the desired output.

Conclusion: In this experiment, we successfully developed an AWS Lambda function, covering the key steps involved. Starting with the Python-based setup, we configured the function's fundamental settings, including setting the timeout to 1 second. We proceeded to create a test event, deployed the function, and verified its output. Additionally, we made updates to the Lambda function's code and redeployed it, observing the real-time changes. This hands-on experience highlighted AWS Lambda's

efficiency and adaptability, enabling rapid serverless application development while AWS handles infrastructure and scaling effortlessly.

Adv DevOps Exp-12

Aim:

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

Theory: Exp12

AWS Lambda and S3 Integration:

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

Workflow:

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

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

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

bucket and writing logs to CloudWatch.

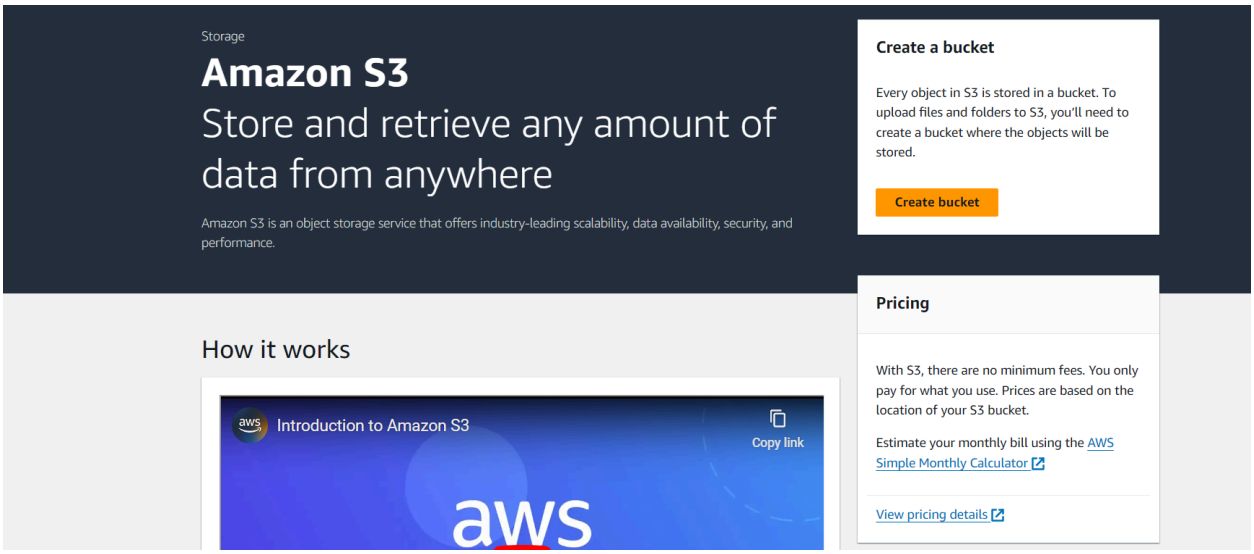
4. Configure S3 Trigger:
○ Link the S3 bucket to the Lambda function by setting up a trigger. Specify that the function should be triggered when an object is created in the bucket (e.g., when an image is uploaded).

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

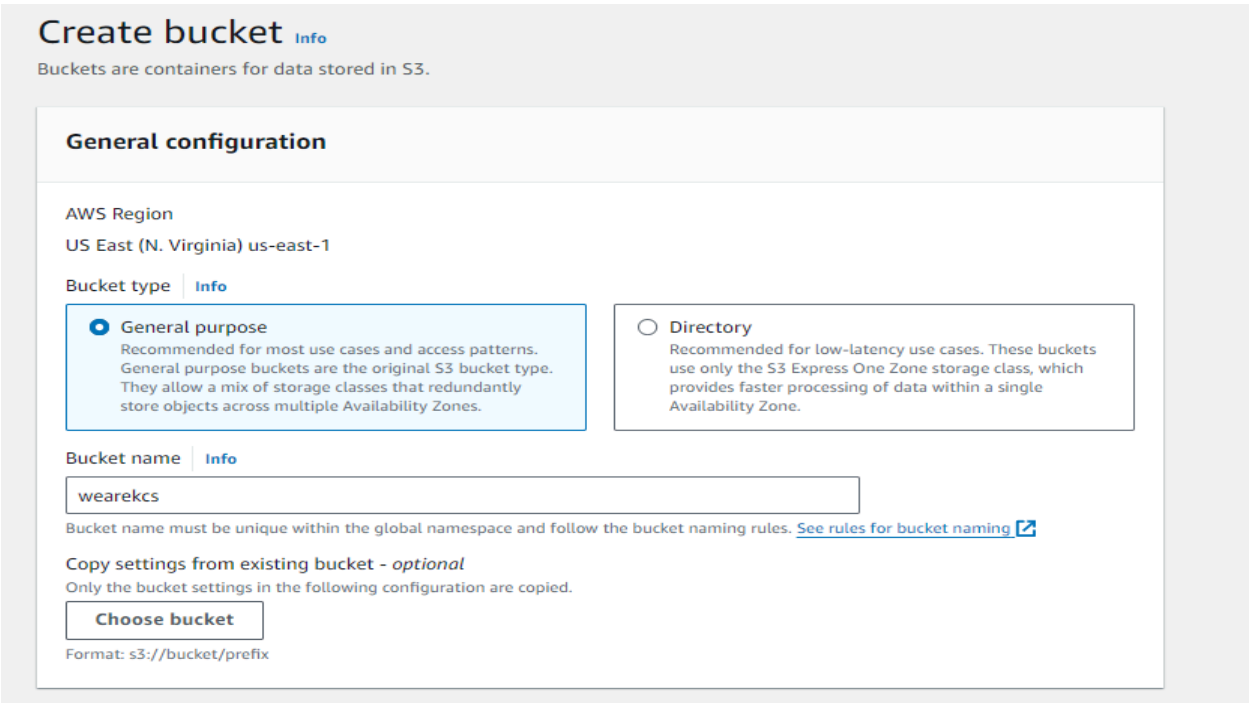
Prerequisites: AWS Personal Account

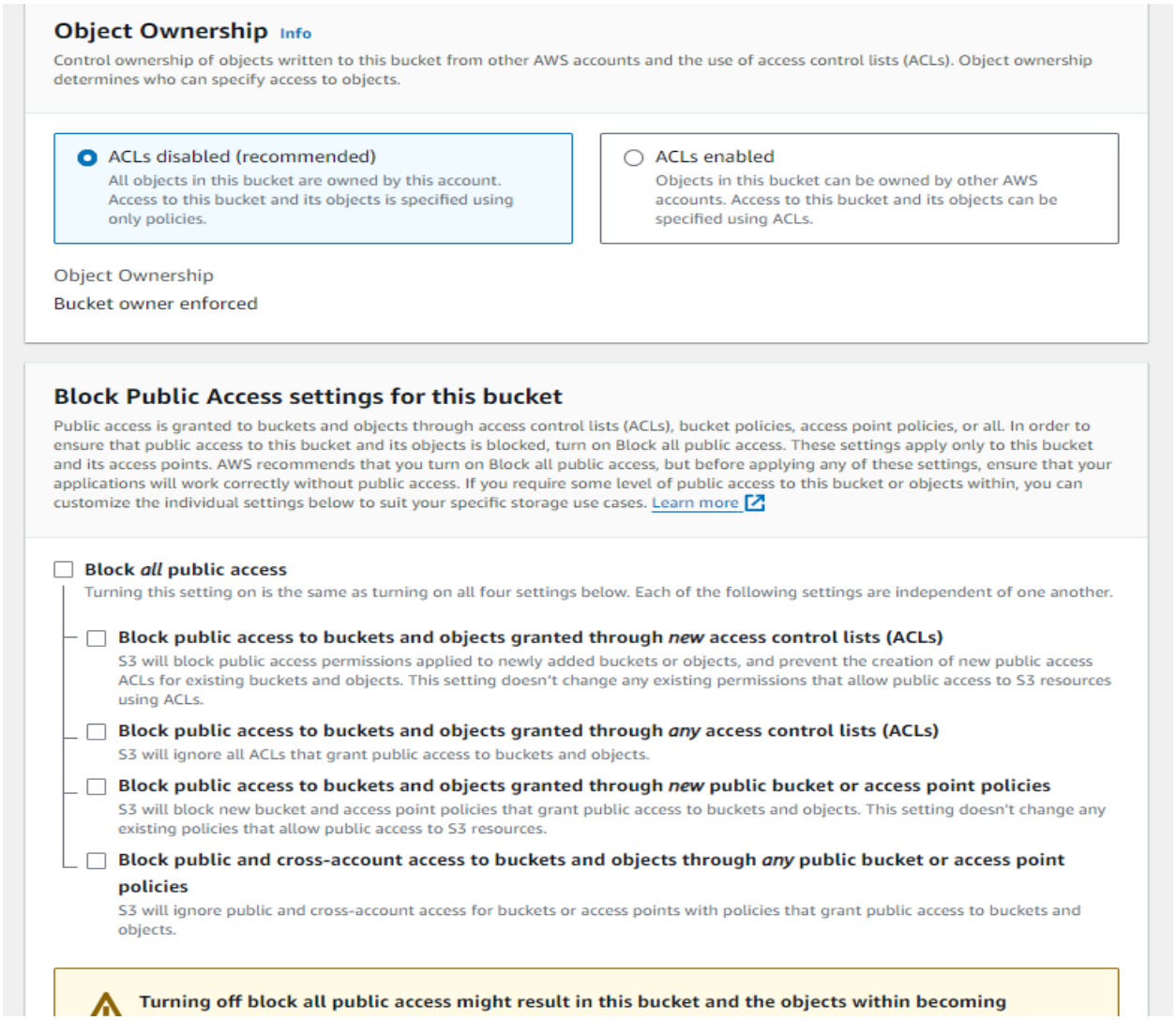
Steps To create the lambda function:

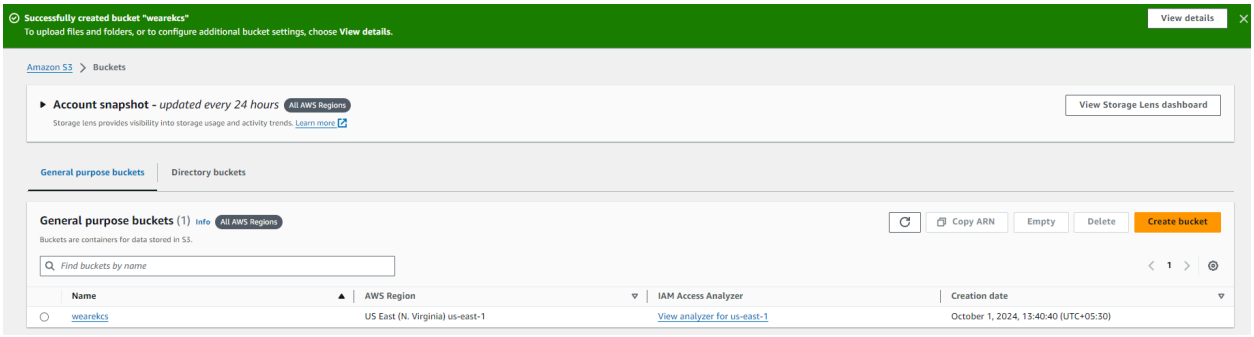
Step 1: Login to your AWS Personal account. Now open S3 from services and click on create S3 bucket.



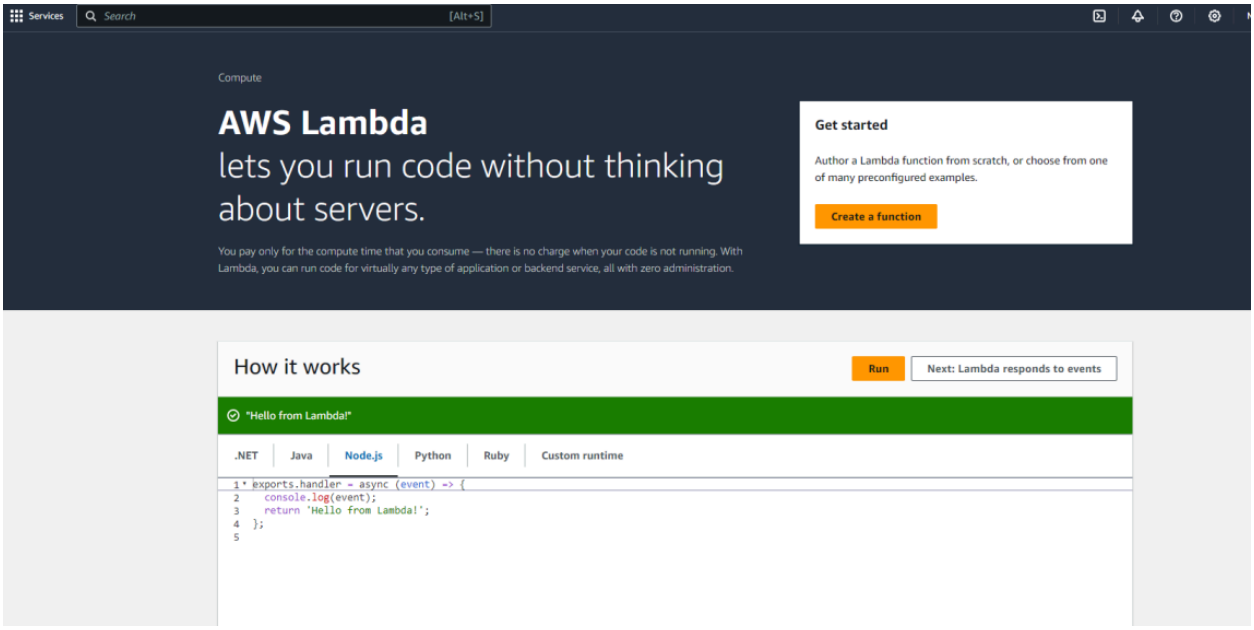
Step 2: Now Give a name to the Bucket, select general purpose project and deselect the Block public access and keep other this to default.



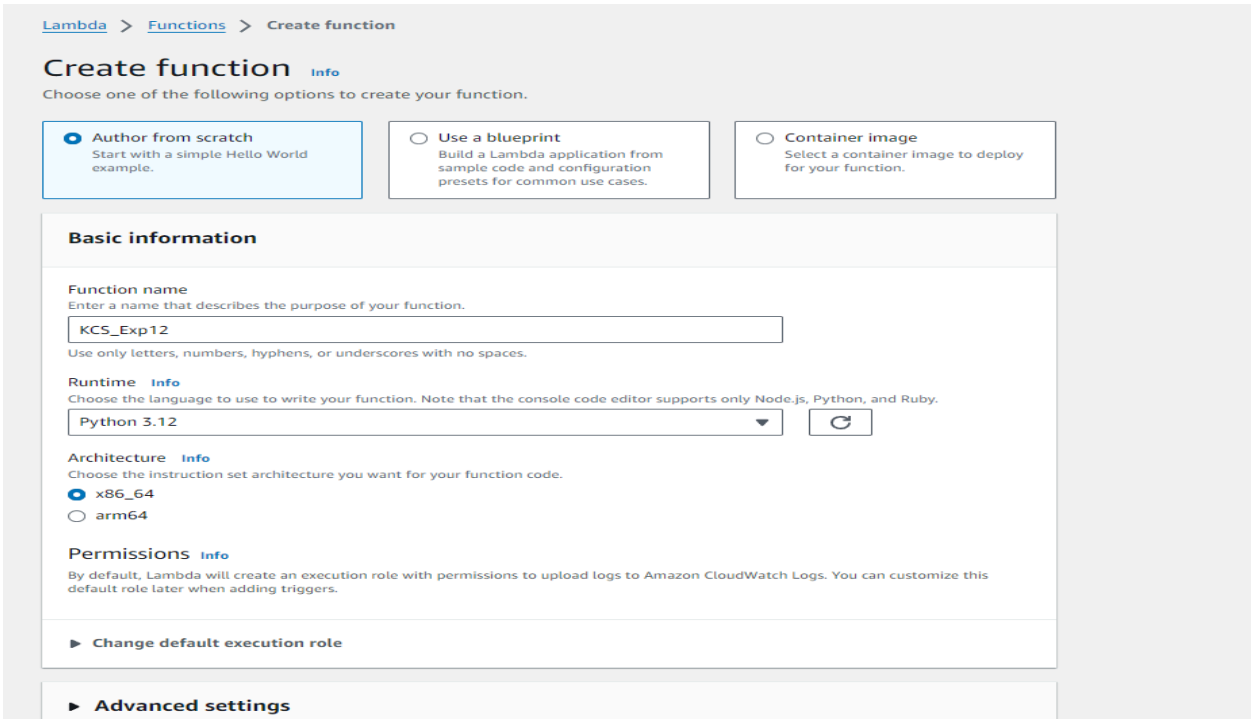


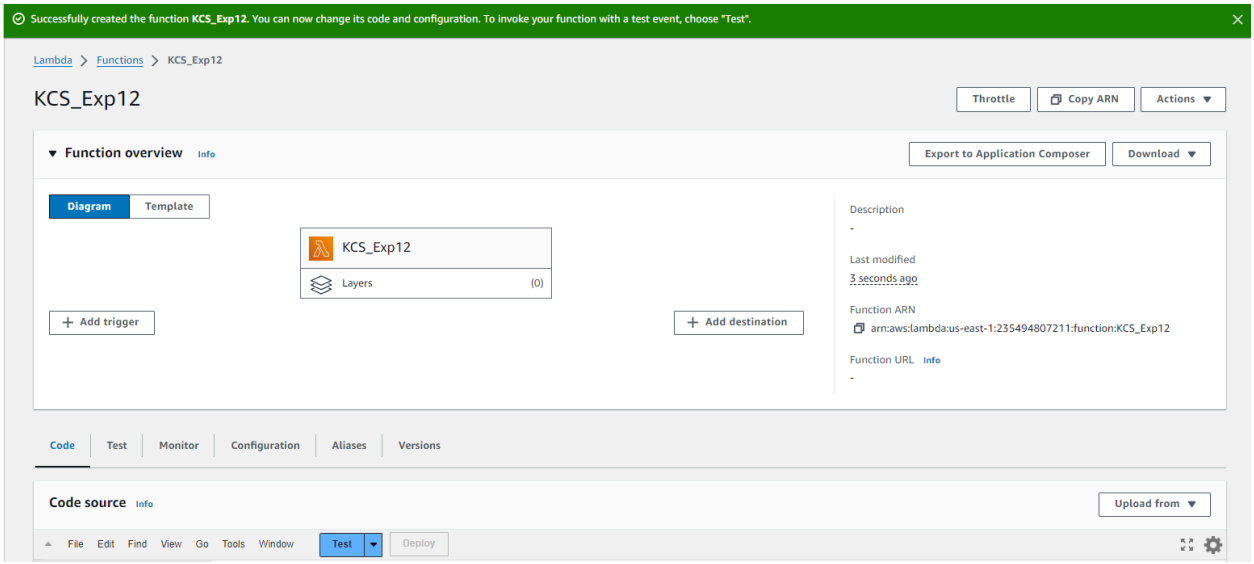


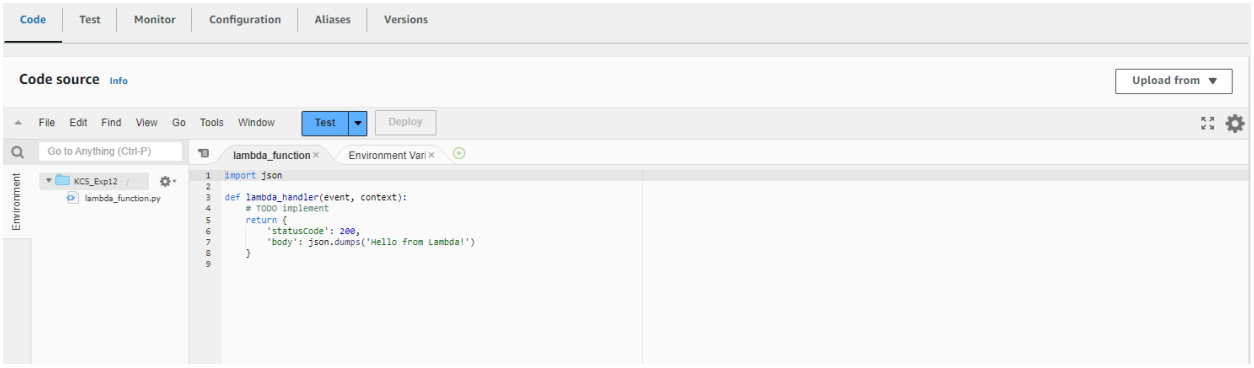
Step 3: Open lambda console and click on create function button



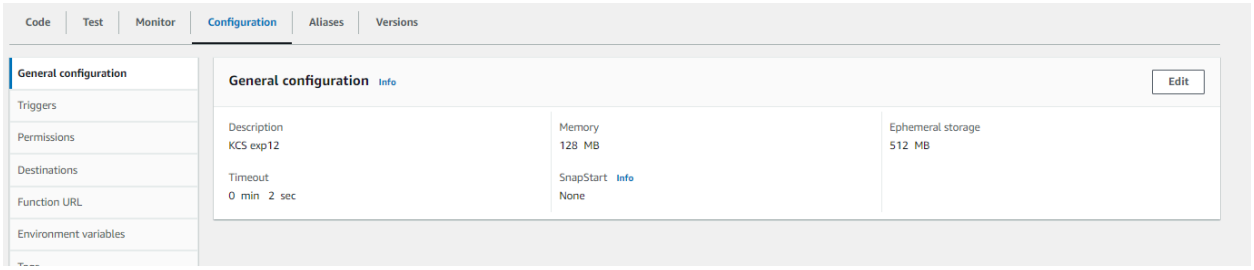
Step 4: Now Give a name to your Lambda function, Select the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby. So will select Python 3.12 , Architecture as x86, and Execution role to Create a new role with basic Lambda permissions.



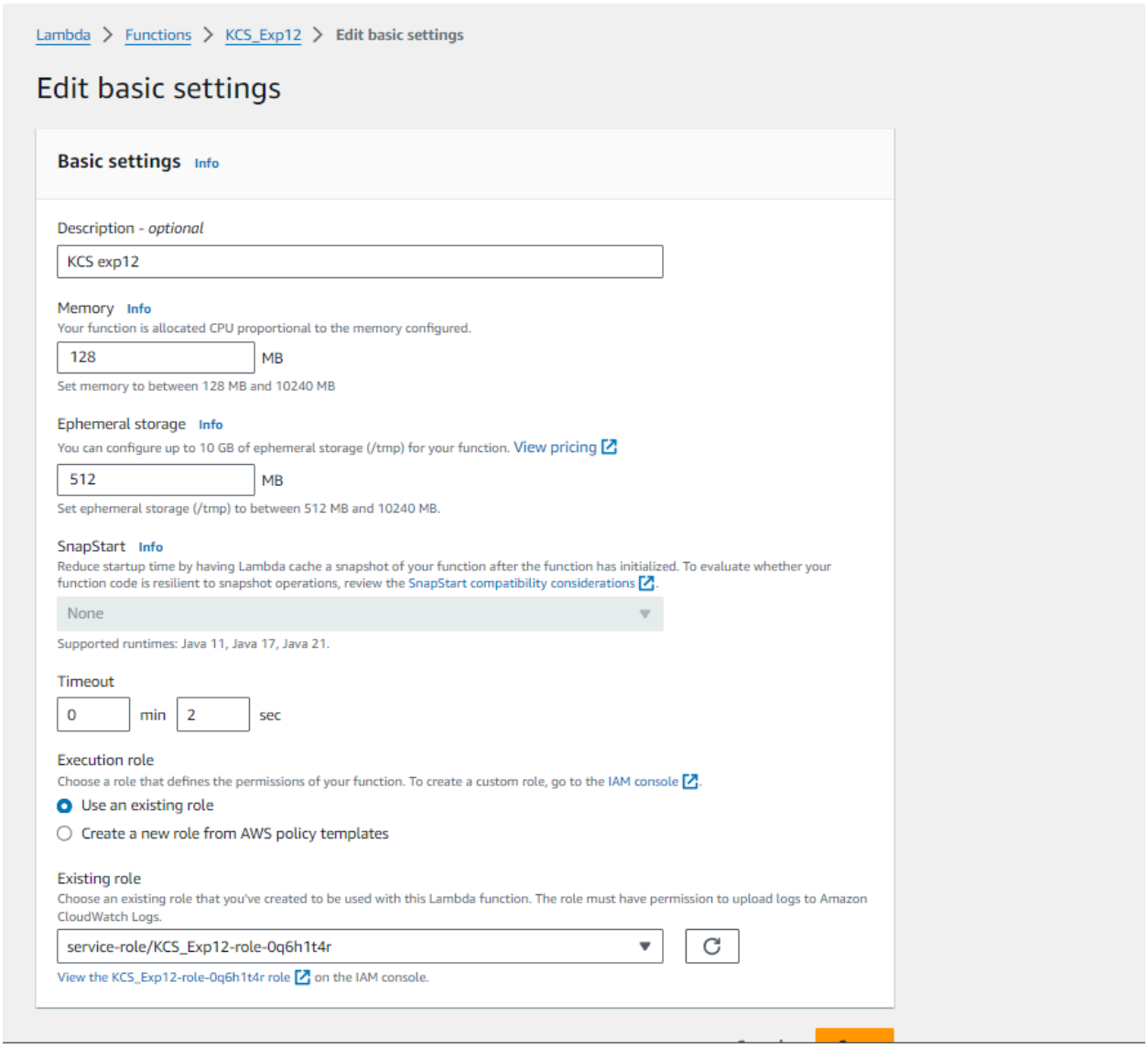




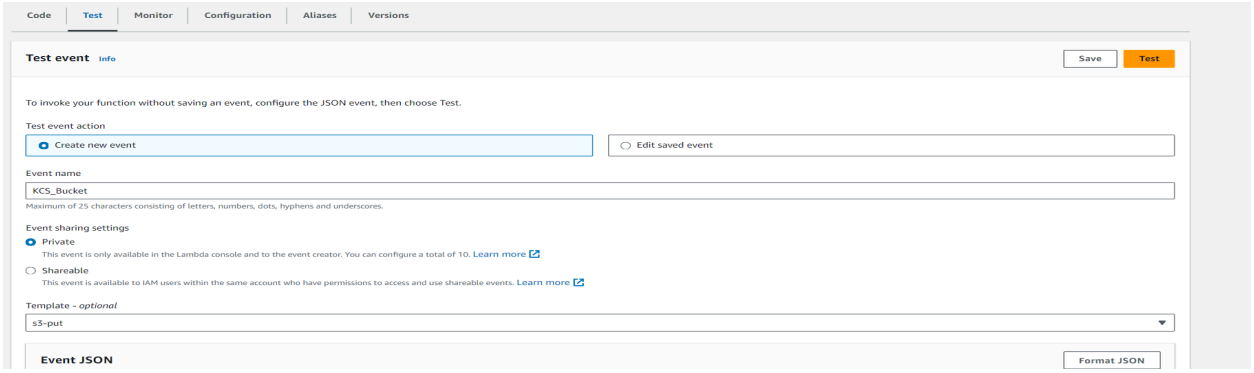
To See or Edit the basic settings go to configuration then click on edit general setting

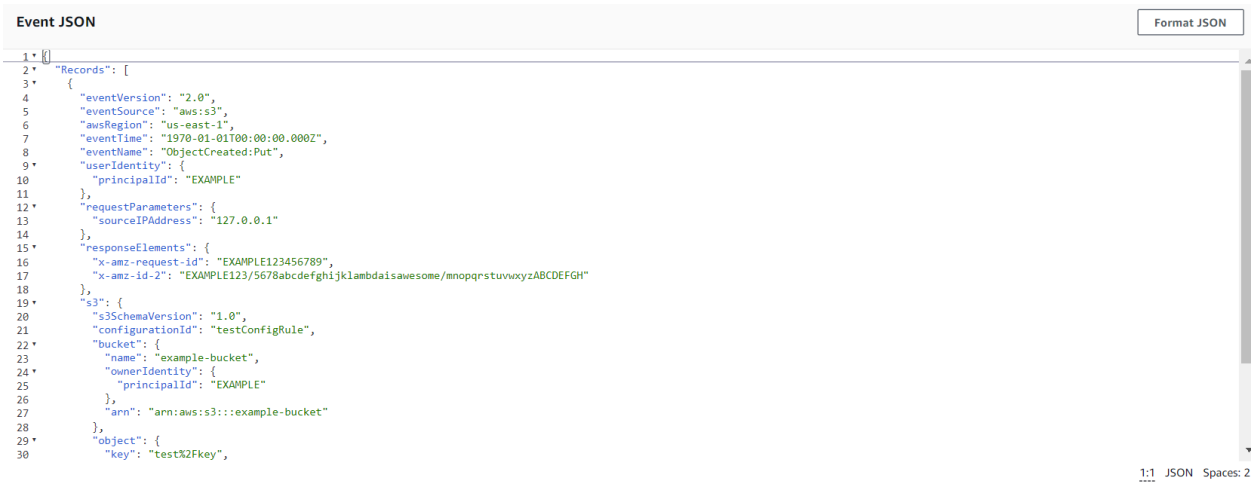


Change any setting of your choice. Here I have set a timeout of 2 secs.Then save changes

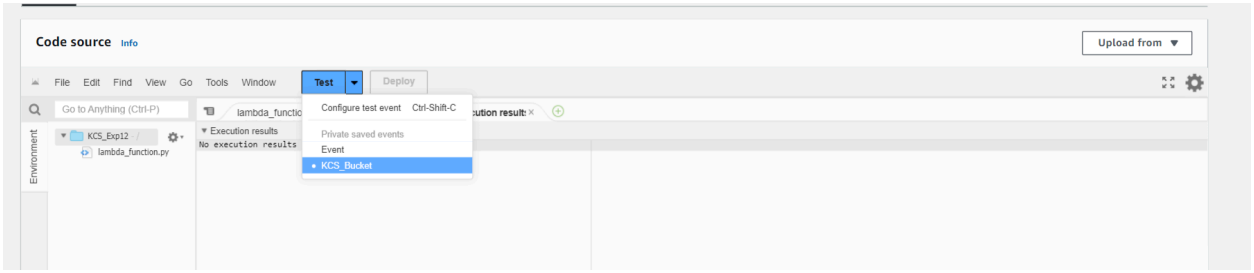


Step 5: Now Click on the Test tab then select Create a new event, give a name to the event and select Event Sharing to private, and select s3 put template.

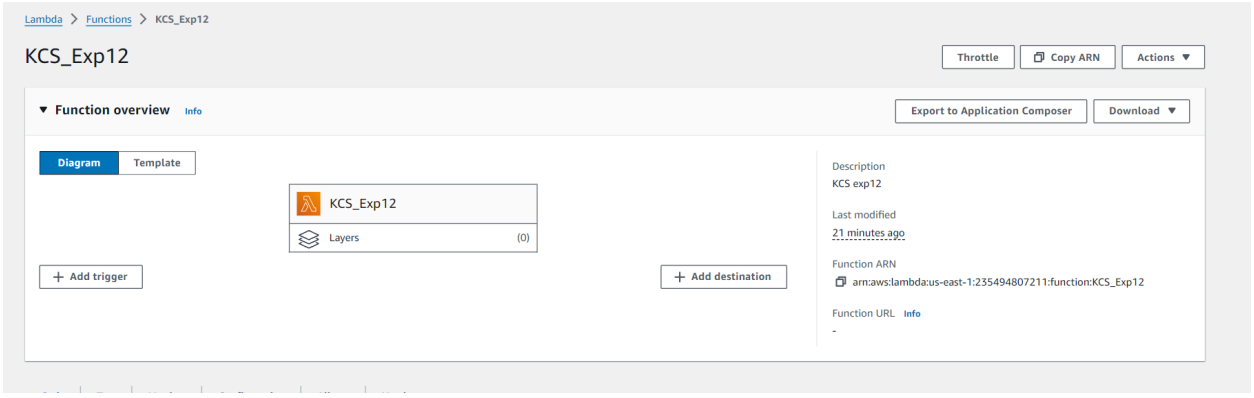




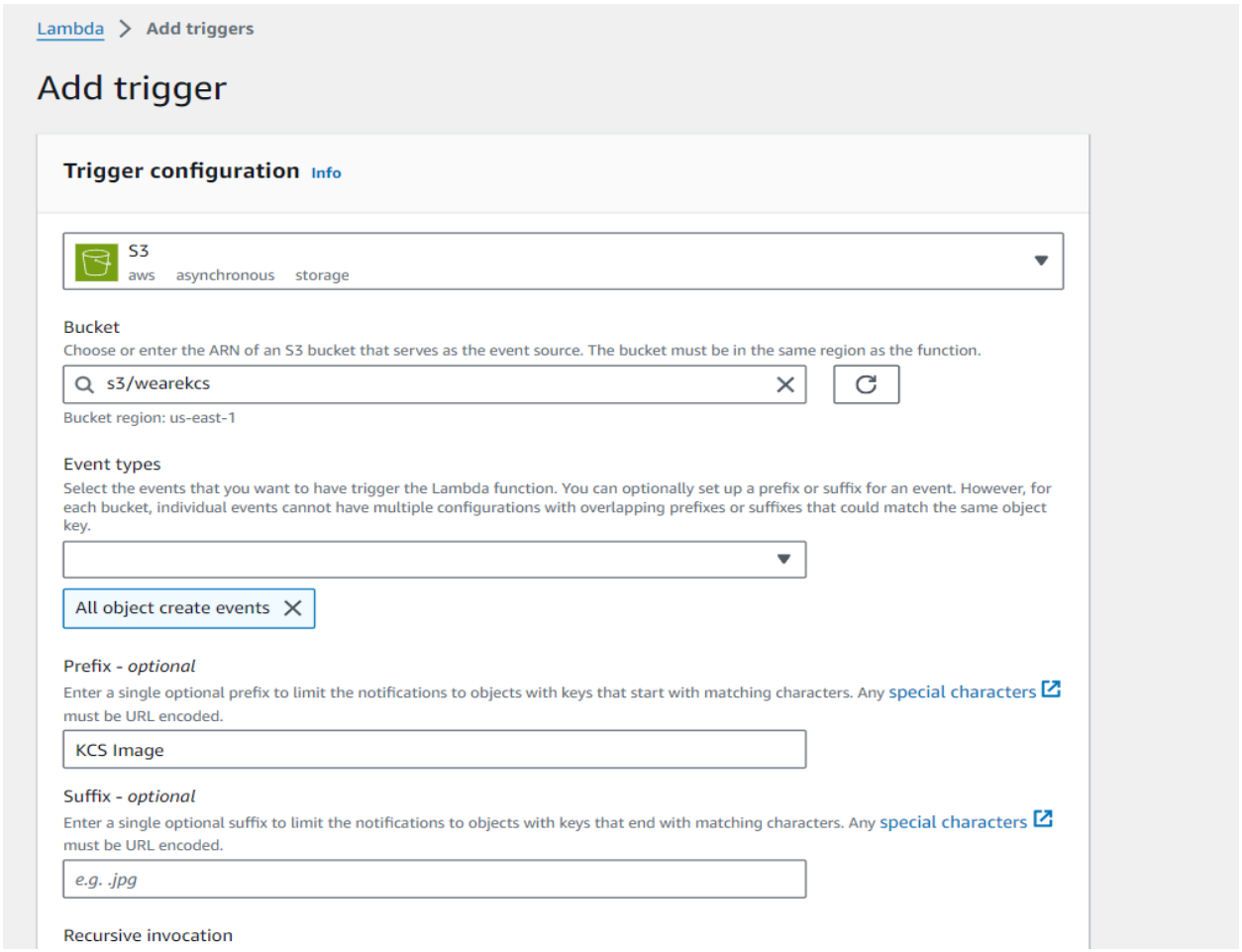
Step 6: Now In the Code section select the created event from the dropdown .

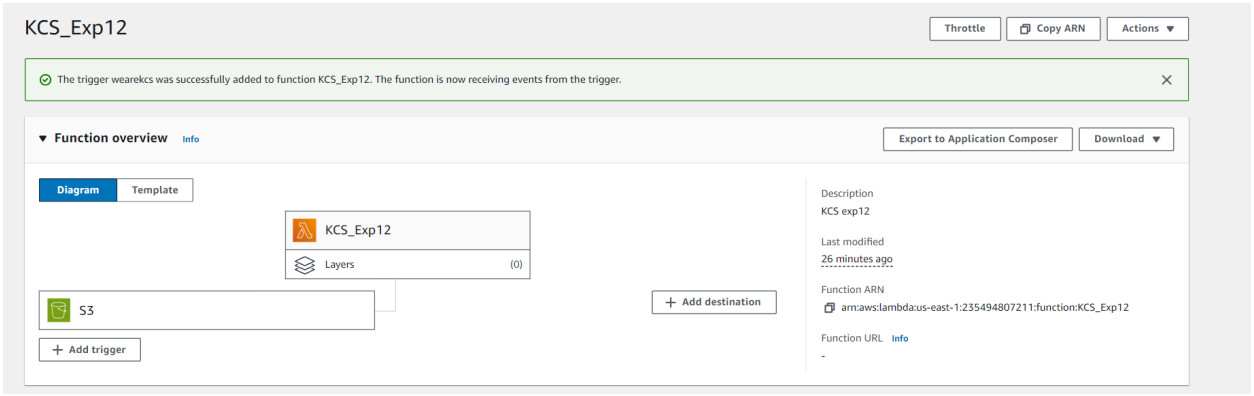


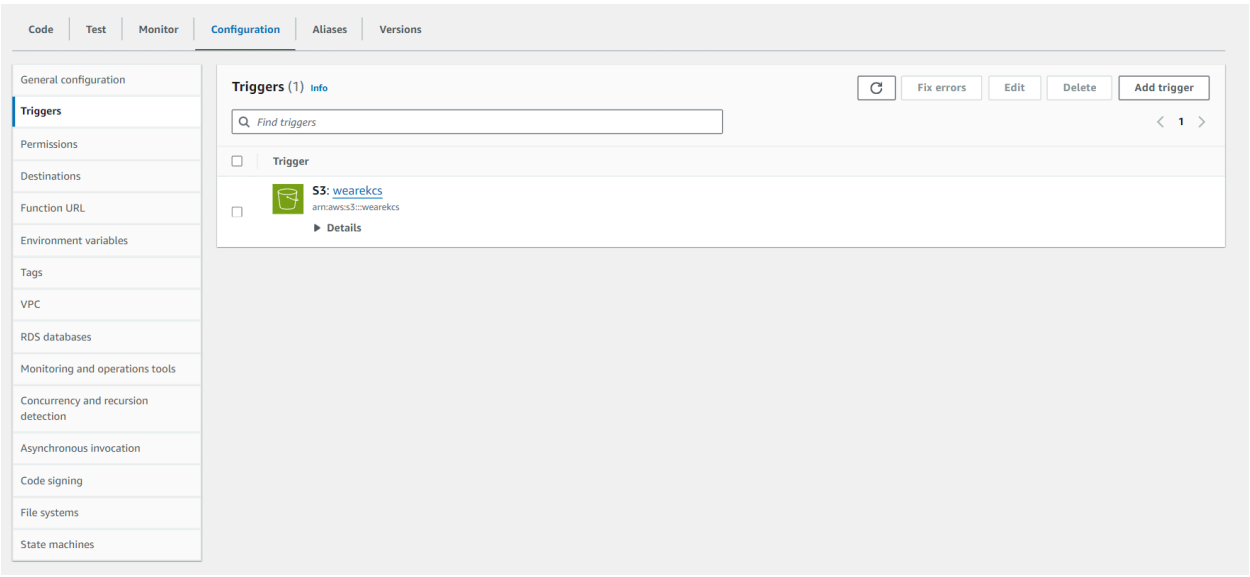
Step 7: Now In the Lambda function click on add tigger



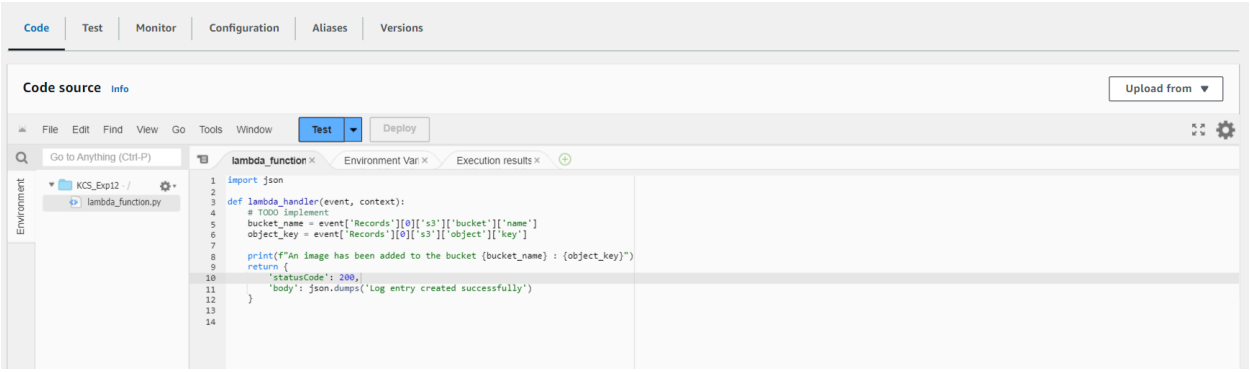
Now select the source as S3 then select the bucket name from the dropdown, keep other things to default and also you can add prefix to image.

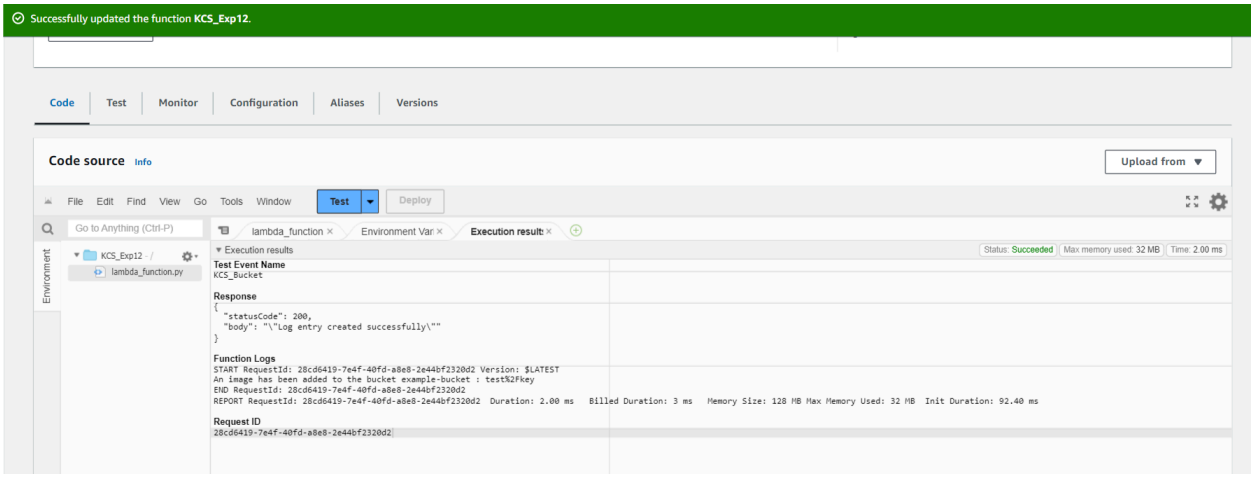

Lambda > Add triggers
Add trigger
Trigger configuration Info
S3
aws asynchronous storage
Bucket
Choose or enter the ARN of an S3 bucket that serves as the event source. The bucket must be in the same region as the function.
s3/wearekcs
Bucket region: us-east-1
Event types
Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.
All object create events
Prefix - optional
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters. Any special characters must be URL encoded.
KCS Image
Suffix - optional
Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters. Any special characters must be URL encoded.
e.g. .jpg
Recursive invocation


KCS_Exp12
Throttle
Copy ARN
Actions
The trigger wearekcs was successfully added to function KCS_Exp12. The function is now receiving events from the trigger.
Function overview Info
Export to Application Composer
Download
Diagram
Template
KCS_Exp12
Layers (0)
S3
Add destination
Add trigger
Description
KCS exp12
Last modified
26 minutes ago
Function ARN
arn:aws:lambda:us-east-1:235494807211:function:KCS_Exp12
Function URL Info
-



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





Step 9: Now upload any image to the bucket.

Amazon S3 > Buckets > wearekcs > Upload

Upload Info

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. Learn more

Drag and drop files and folders you want to upload here, or choose **Add files** or **Add folder**.

Files and folders (1 Total, 957.0 KB)

Remove | Add files | Add folder

All files and folders in this table will be uploaded.

Find by name

1

Name	Folder
F_i0UxsXgAAXB2s.jpg	-

Destination Info

Destination

s3://wearekcs

▶ **Destination details**

Bucket settings that impact new objects stored in the specified destination.

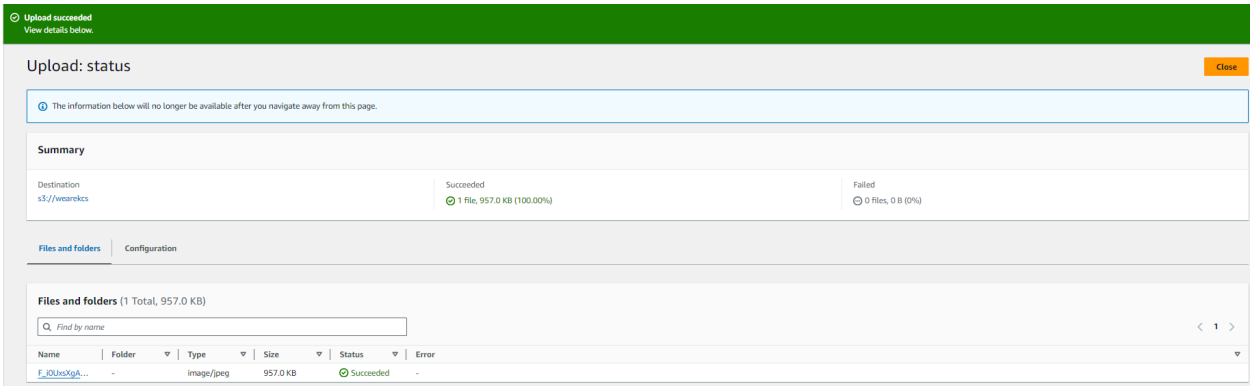
▶ **Permissions**

Grant public access and access to other AWS accounts.

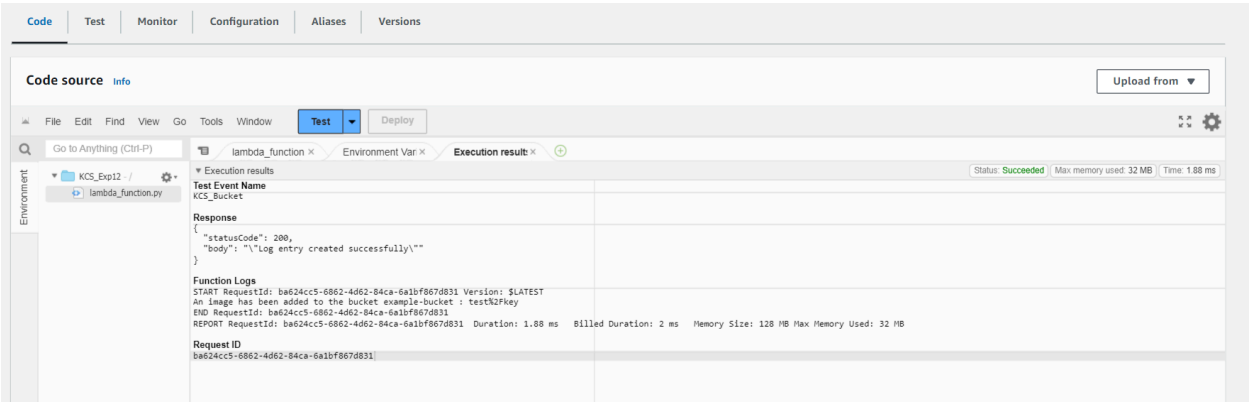
▶ **Properties**

Specify storage class, encryption settings, tags, and more.

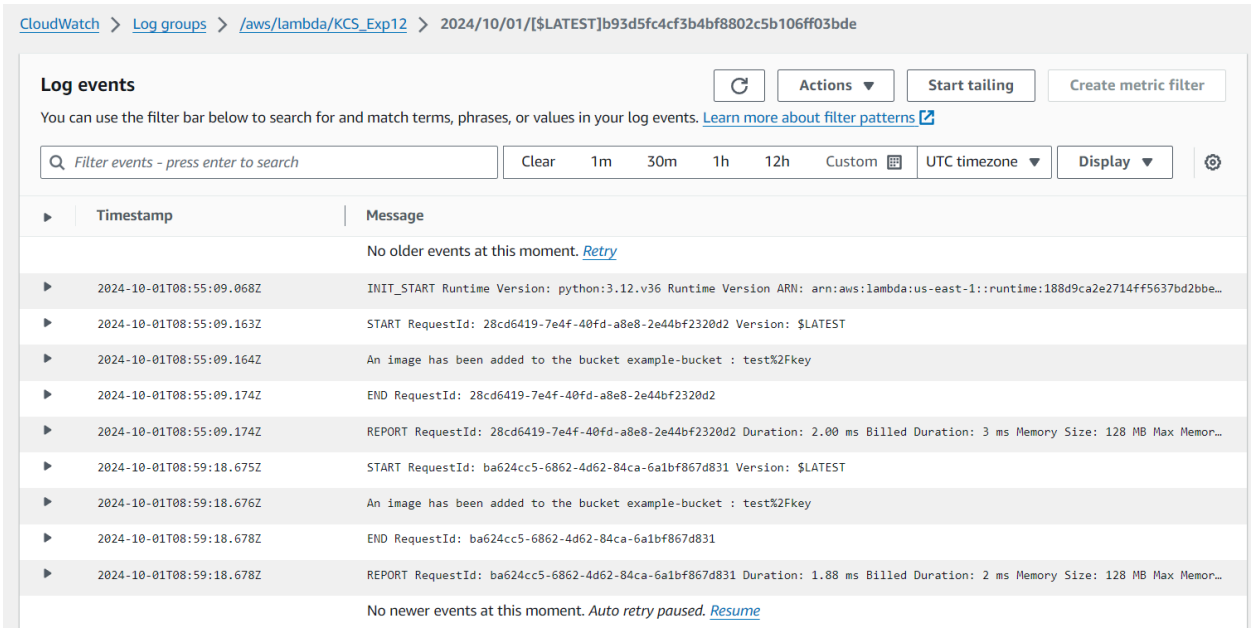
Cancel | Upload



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



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


CloudWatch > Log groups > /aws/lambda/KCS_Exp12 > 2024/10/01/[$LATEST]b93d5fc4cf3b4bf8802c5b106ff03bde
Log events
Actions
Start tailing
Create metric filter
You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns
Filter events - press enter to search
Clear
1m
30m
1h
12h
Custom
UTC timezone
Display
Timestamp
Message
No older events at this moment. Retry
2024-10-01T08:55:09.068Z INIT_START Runtime Version: python:3.12.v36 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:188d9ca2e2714ff5637bd2bbe…
2024-10-01T08:55:09.163Z START RequestId: 28cd6419-7e4f-40fd-a8e8-2e44bf2320d2 Version: $LATEST
2024-10-01T08:55:09.164Z An image has been added to the bucket example-bucket : test%2Fkey
2024-10-01T08:55:09.174Z END RequestId: 28cd6419-7e4f-40fd-a8e8-2e44bf2320d2
2024-10-01T08:55:09.174Z REPORT RequestId: 28cd6419-7e4f-40fd-a8e8-2e44bf2320d2 Duration: 2.00 ms Billed Duration: 3 ms Memory Size: 128 MB Max Memor…
2024-10-01T08:59:18.675Z START RequestId: ba624cc5-6862-4d62-84ca-6a1bf867d831 Version: $LATEST
2024-10-01T08:59:18.676Z An image has been added to the bucket example-bucket : test%2Fkey
2024-10-01T08:59:18.678Z END RequestId: ba624cc5-6862-4d62-84ca-6a1bf867d831
2024-10-01T08:59:18.678Z REPORT RequestId: ba624cc5-6862-4d62-84ca-6a1bf867d831 Duration: 1.88 ms Billed Duration: 2 ms Memory Size: 128 MB Max Memor…
No newer events at this moment. Auto retry paused. Resume