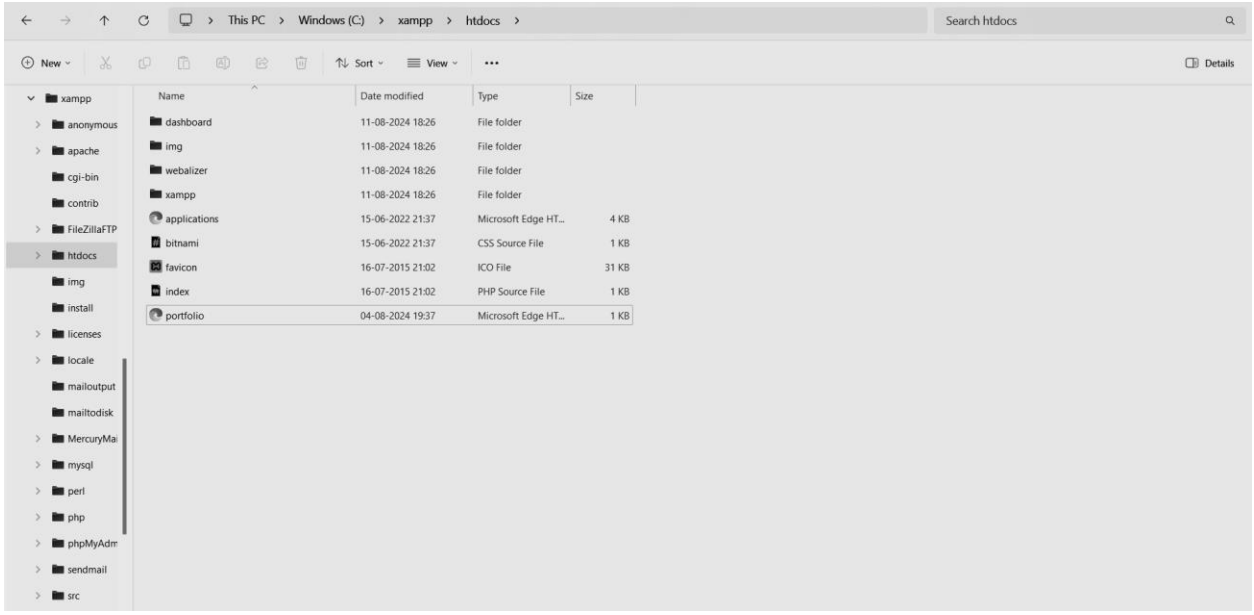
NAME:Bhushan Malpani **DIV:D15C** **Roll no.:33**

a)Hosting of html file on a local virtual machine using Xampp

1. Create an portfolio file with name portfolio.html.

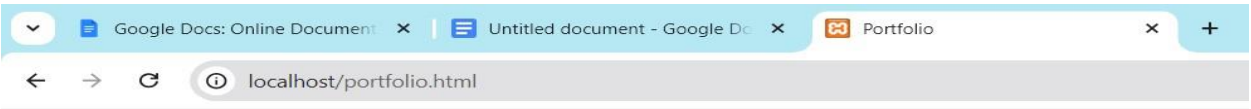
```
<> portfolio.html > html > body
  1   <!DOCTYPE html>
  2   <html lang="en">
  3   <head>
  4       <meta charset="UTF-8">
  5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
  6       <title>Portfolio</title>
  7   </head>
  8   <body>
  9       <h1>Bhushan Malpani</h1>
 10       <br>
 11       
 12       <h2>Education</h2>
 13       <ul>
 14           <li><b>10th class (CBSE)</b>: 94.8%</li>
 15           <li><b>12th class (HSC)</b>: 90%</li>
 16           <li><b>Bachelor in Engineering(IT)</b></li>
 17       </ul>
 18       <h2>Skills</h2>
 19       <ul>
 20           <li>HTML,CSS,JS</li>
 21           <li>Java,C++</li>
 22       </ul>
 23       <h2>Hobbies</h2>
 24       <ul>
 25           <li>Cricket</li>
 26           <li>Music</li>
 27       </ul>
 28       <h2>Contact Me</h2>
 29       <ul>
 30           <li><b>Email</b>:bhushanmalpani45@gmail.com</li>
 31           <li><b>Contact No.</b>:9423244006/;</li>
 32       </ul>
 33   </body>
 34   </html>
```

2. Save this file in htdocs file in Xampp folder in C drive:



3. Make sure Xampp is installed and then start Xampp control panel and start the module Apache and Mysql.

4. Access the content of the portfolio page using **localhost/portfolio.html** in any web browser. Your portfolio should be hosted successfully on local machine using Xampp.



Bhushan Malpani

Education

- **10th class (CBSE)**: 94.8%
- **12th class (HSC)**: 90%
- **Bachelor in Engineering(IT)**

Skills

- HTML,CSS,JS
- Java,C++

Hobbies

- Cricket
- Music

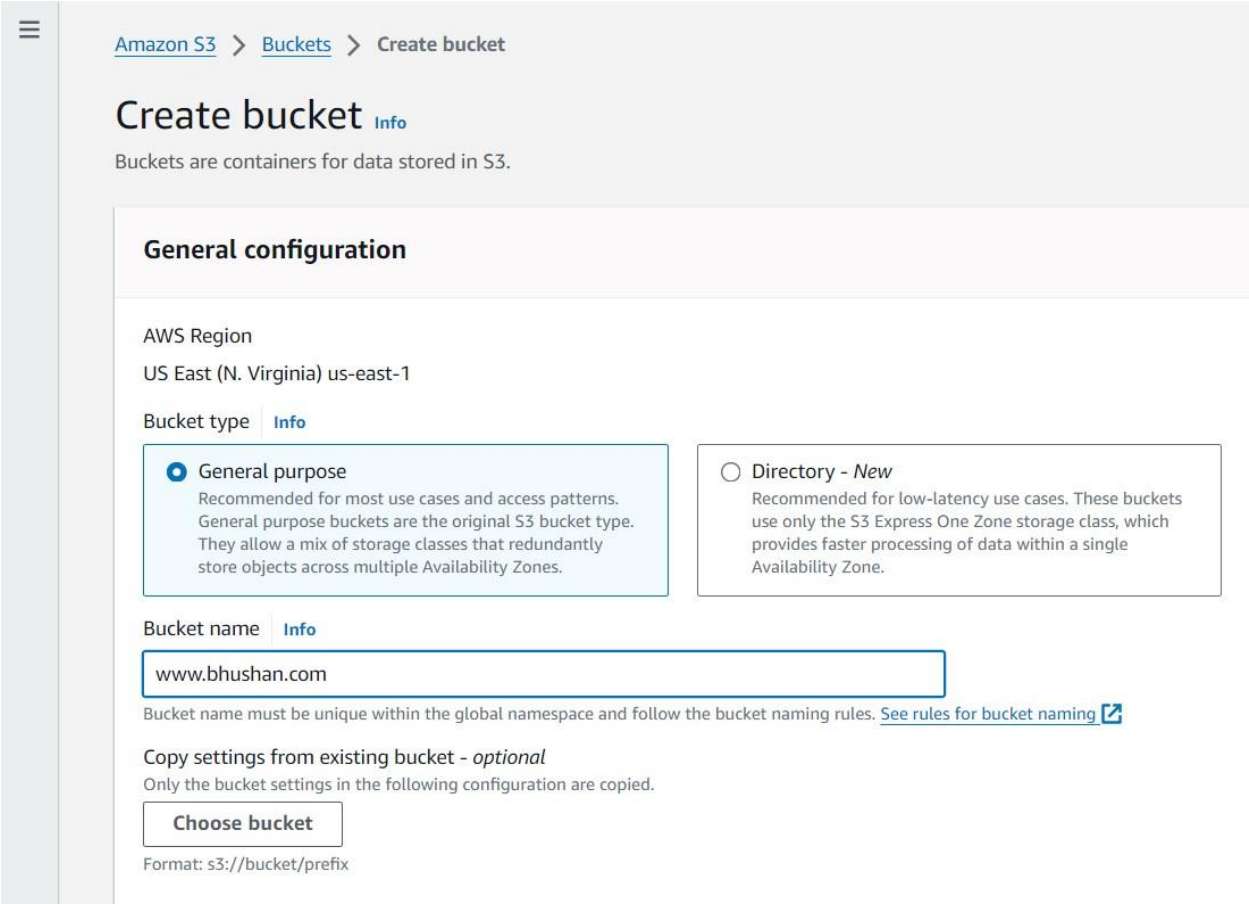
Contact Me

- **Email**:bhushanmalpani45@gmail.com
- **Contact No.**:9423244006/;

Static Hosting Using AWS

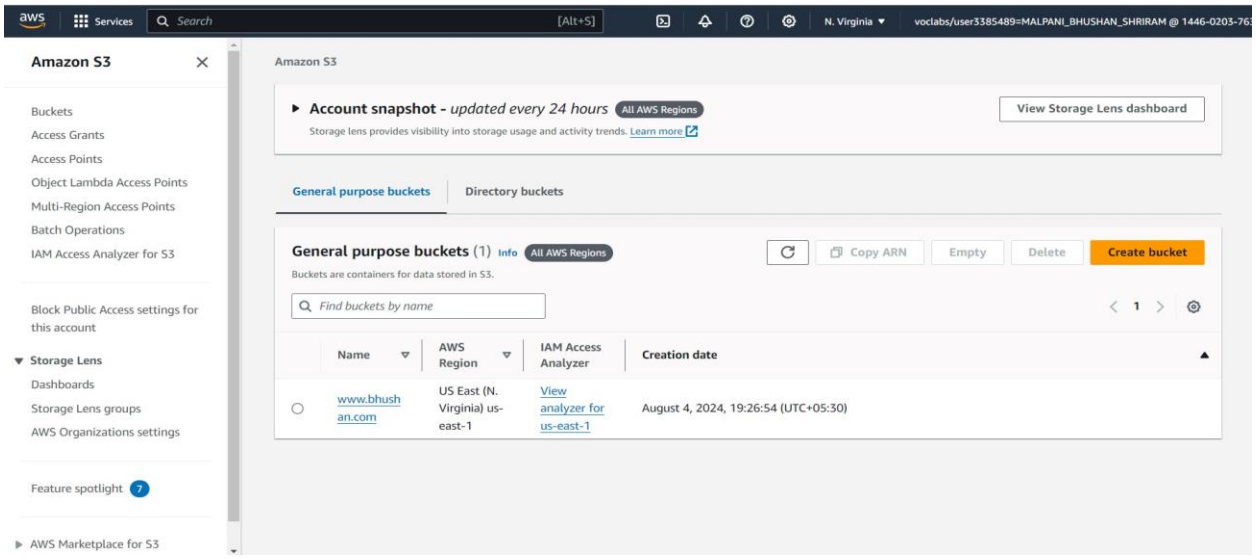
Navigate to S3 Service:

In the AWS Management Console, select Services and then S3 under Storage.



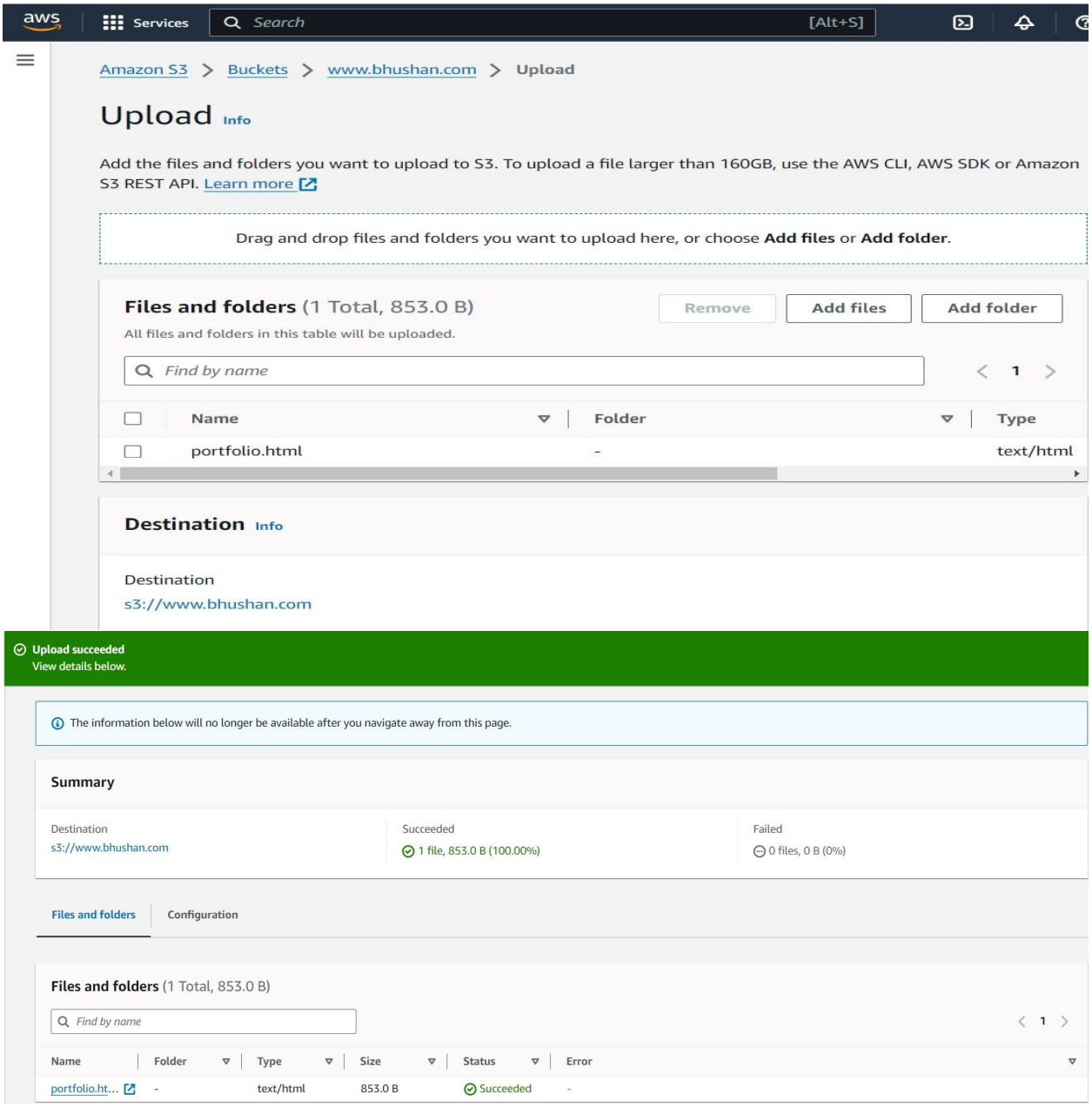
Create a Bucket:

- Click on **Create bucket**.
- **Bucket name**: Enter a unique name that is globally unique ● **Region**: Select a region (usually closest to your users).
- **Block Public Access Settings**: Uncheck **Block all public access**. Confirm the warning that appears about the public access settings. ● Click **Create bucket**.



Upload Files:

- Click on **Upload**.
- Drag and drop your static website files or use the **Add files** and **Add folder** buttons to upload your HTML, CSS, JavaScript, and image files. ● Click **Upload** to start the upload process.

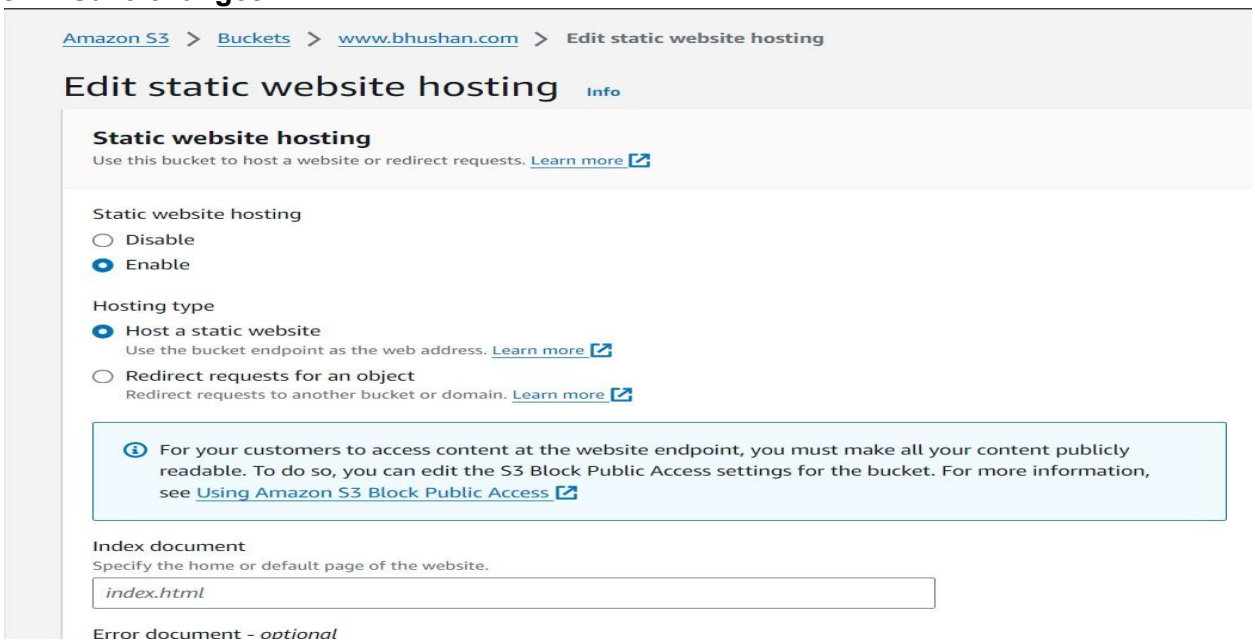


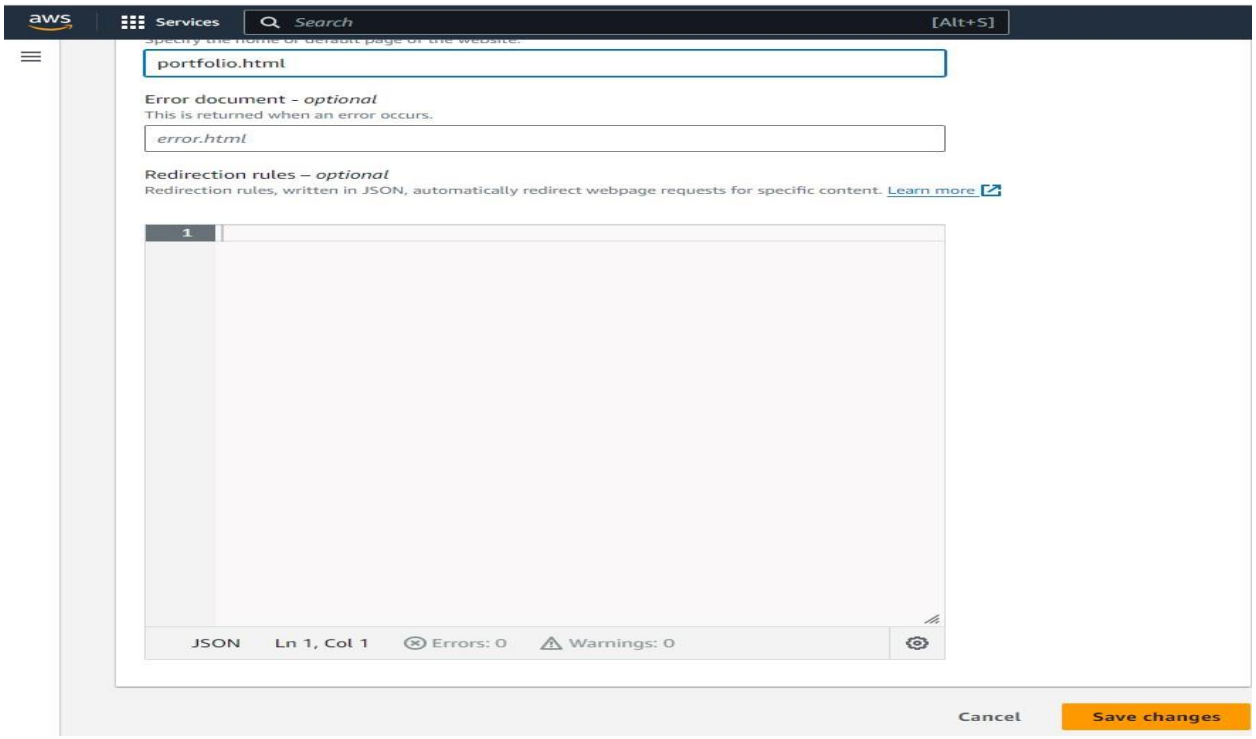
4. Configure Bucket for Static Website Hosting

1. **Access Bucket Properties**:
    - Go to the **Properties** tab of your bucket.

Enable Static Website Hosting:

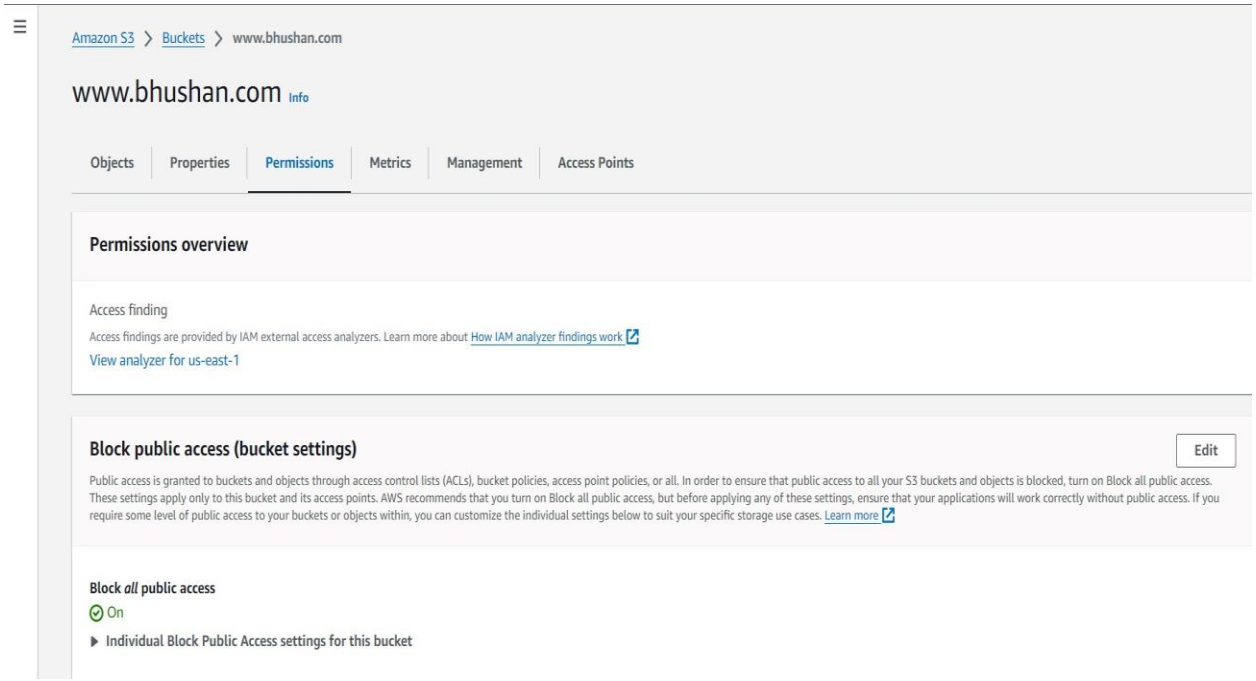
- Scroll down to **Static website hosting** and click **Edit**.
- Select **Enable**.
- **Index document**: Enter `index.html` (or the name of your main HTML file).
- **Error document**: Optionally, enter `error.html` (for handling errors).
- Click **Save changes**.



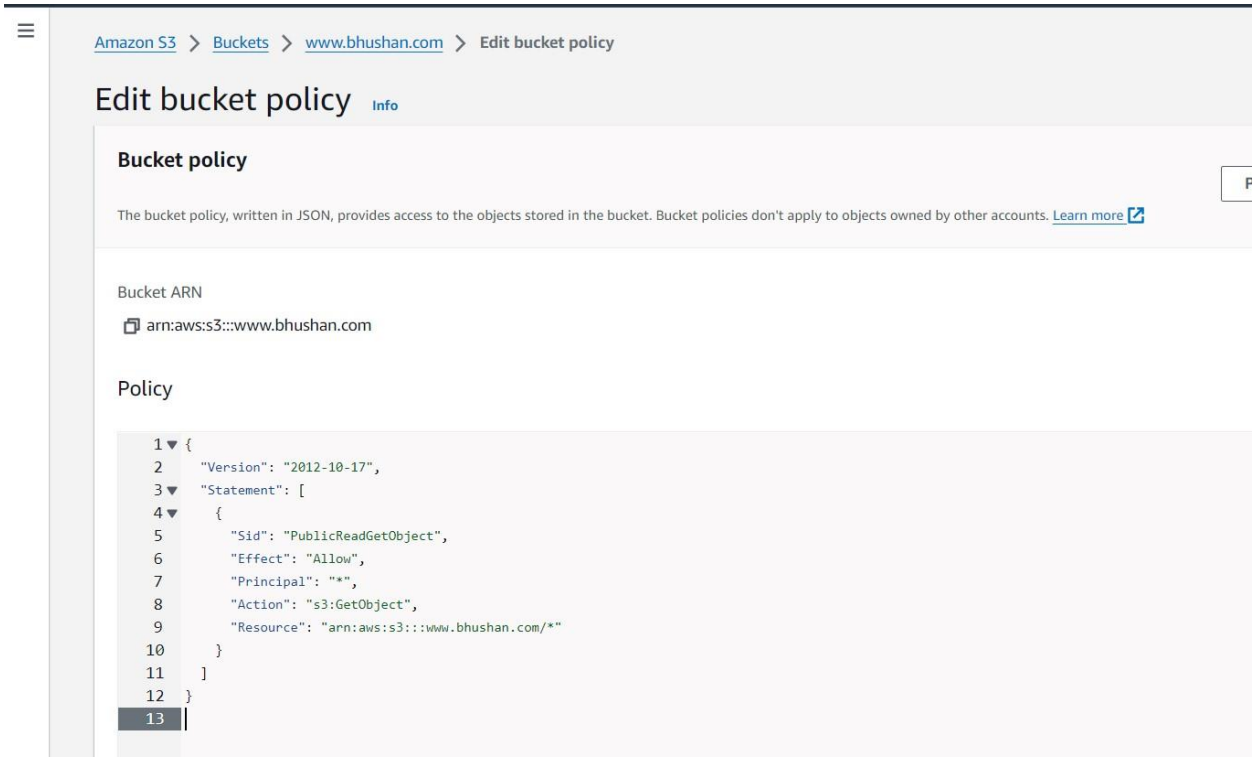


Access Permissions Tab: ● Go to the

Permissions tab of your bucket.



Click **Save** to apply the policy.

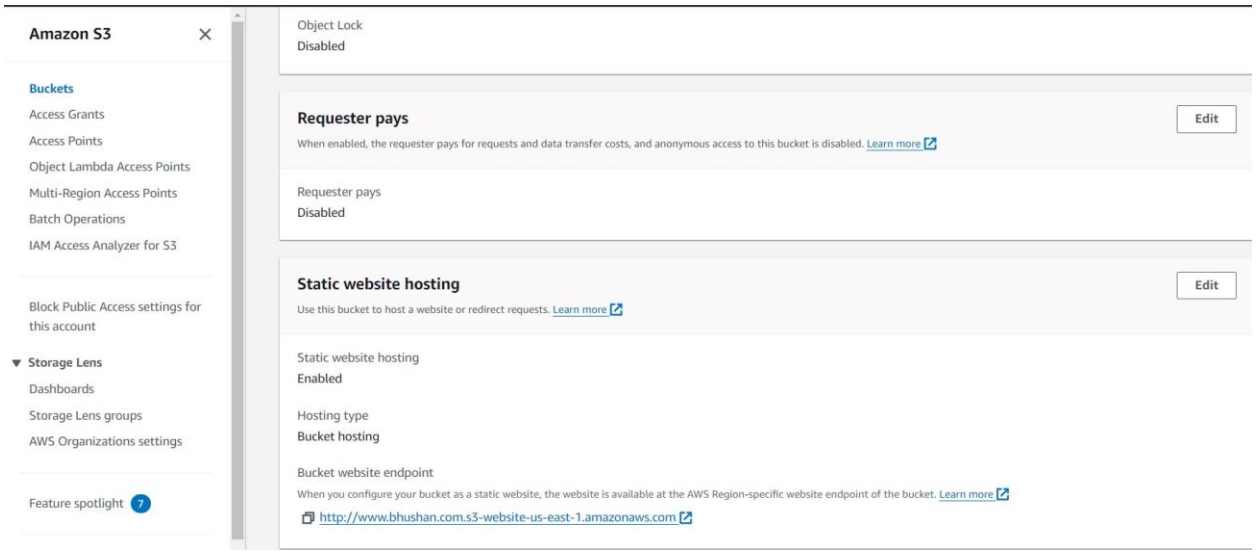


Test Your Website:

- Use the **Endpoint** URL from the static website hosting configuration to access and test your website.

Check Functionality:

- Ensure that all website pages and resources load correctly.



Bhushan Malpani

Education

- **10th class (CBSE)**: 94.8%
- **12th class (HSC)**: 90%
- **Bachelor in Engineering(IT)**

Skills

- HTML,CSS,JS
- Java,C++

Hobbies

- Cricket
- Music

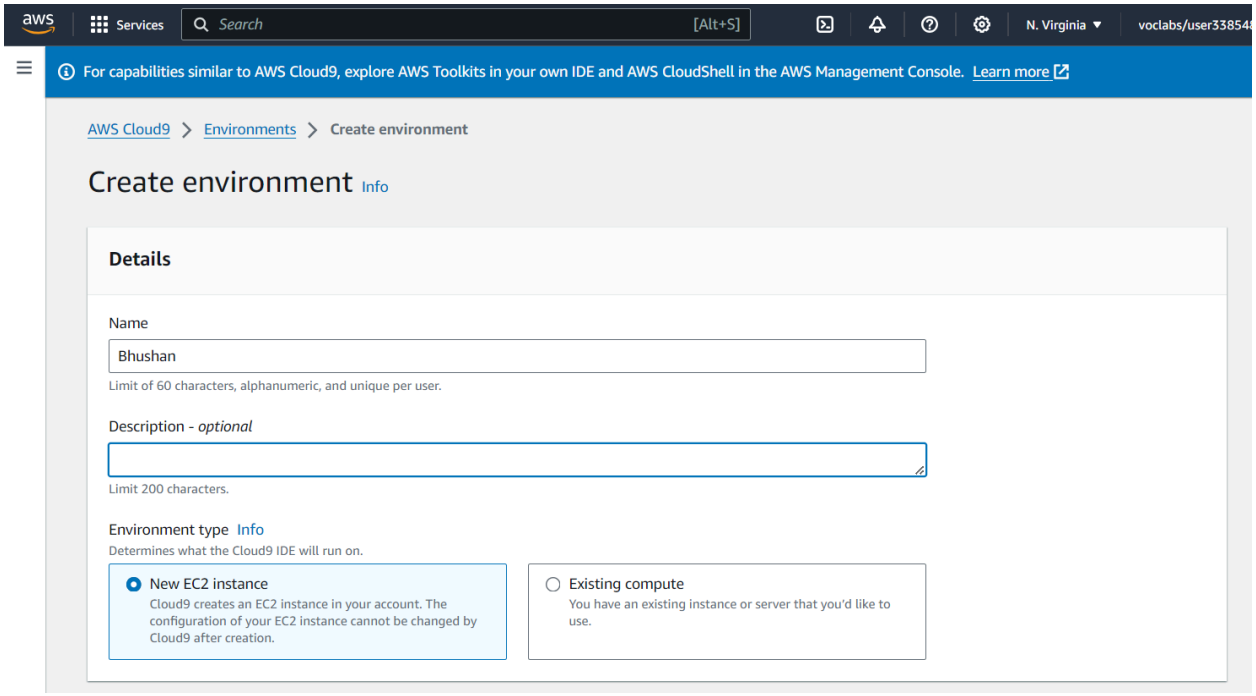
Contact Me

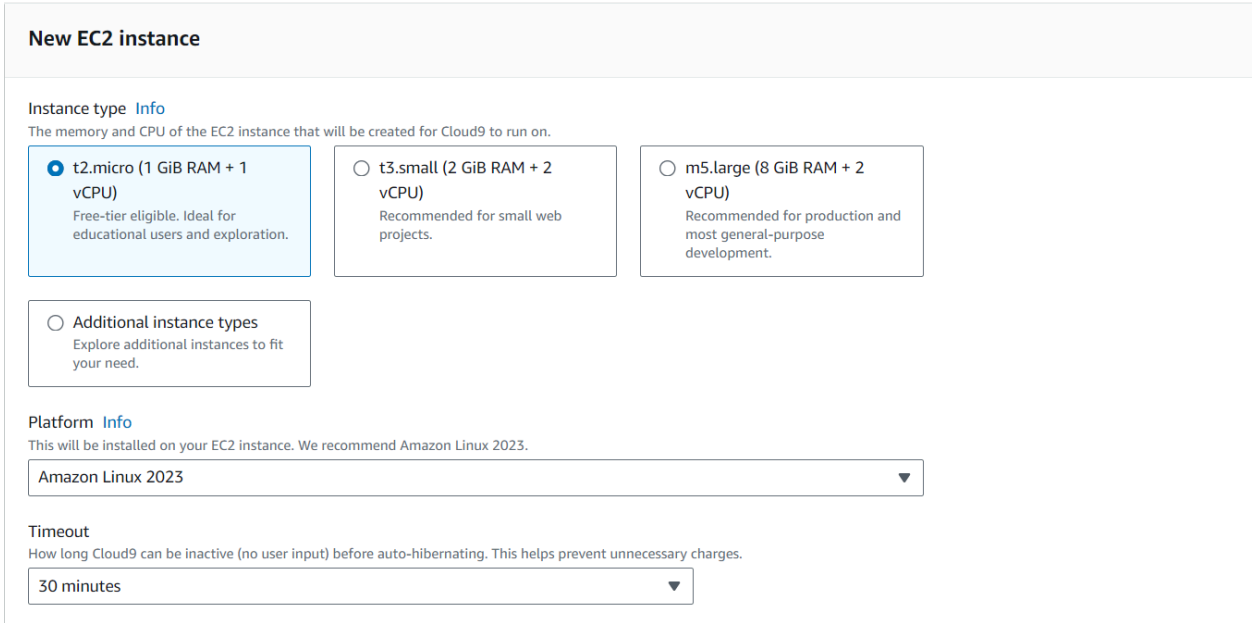
- **Email**:bhushanmalpani45@gmail.com
- **Contact No.**:9423244006/;

NAME: Bhushan Malpani **DIV : D15C** **Roll no: 33**

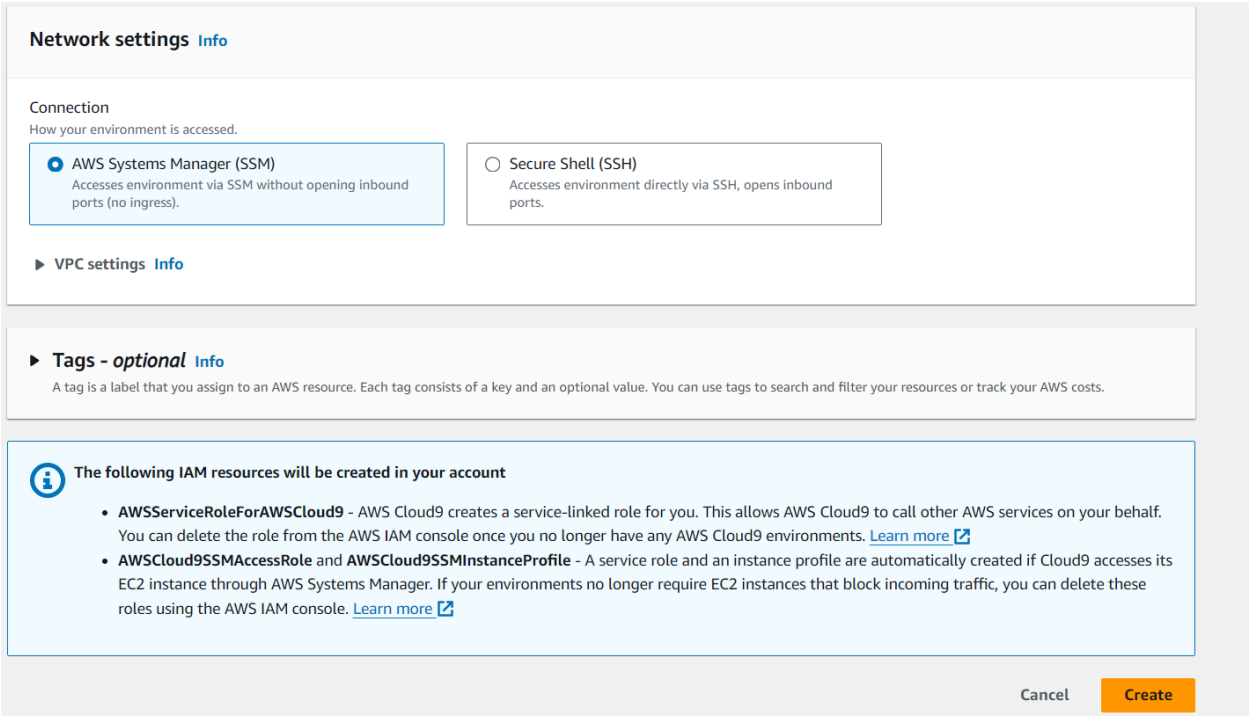
Exp 1b: Cloud9 Setup and Launch, Collaboration demonstration by creation of IAM groups and users.

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

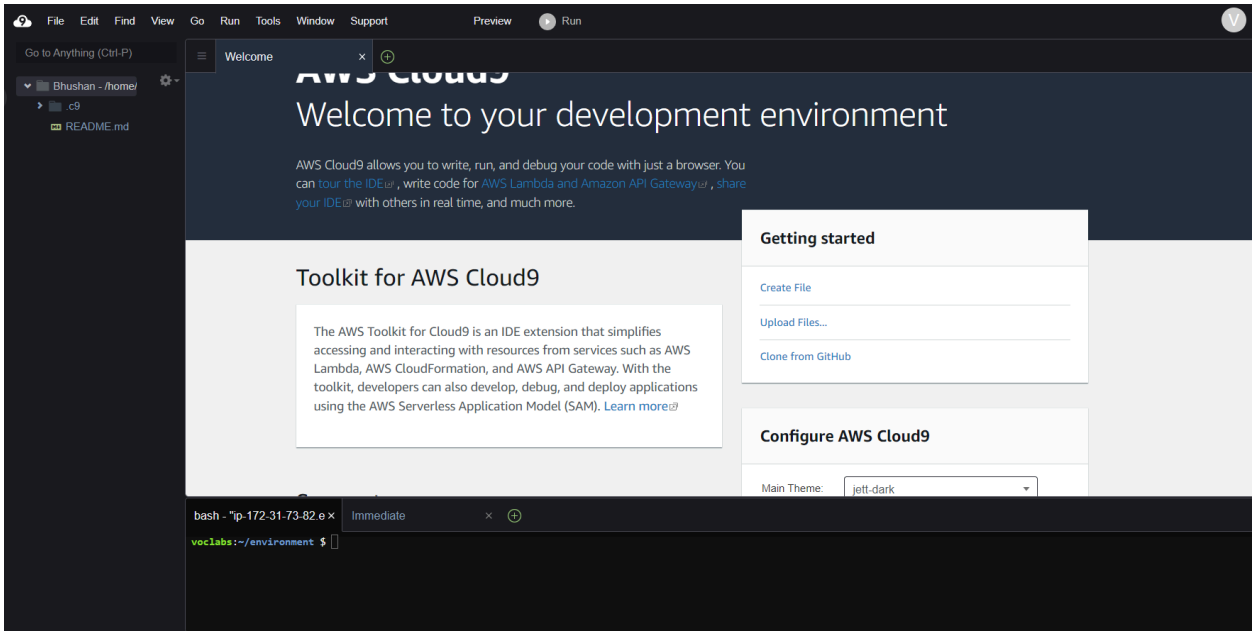




NAME: Bhushan Malpani DIV : D15C Roll no: 33

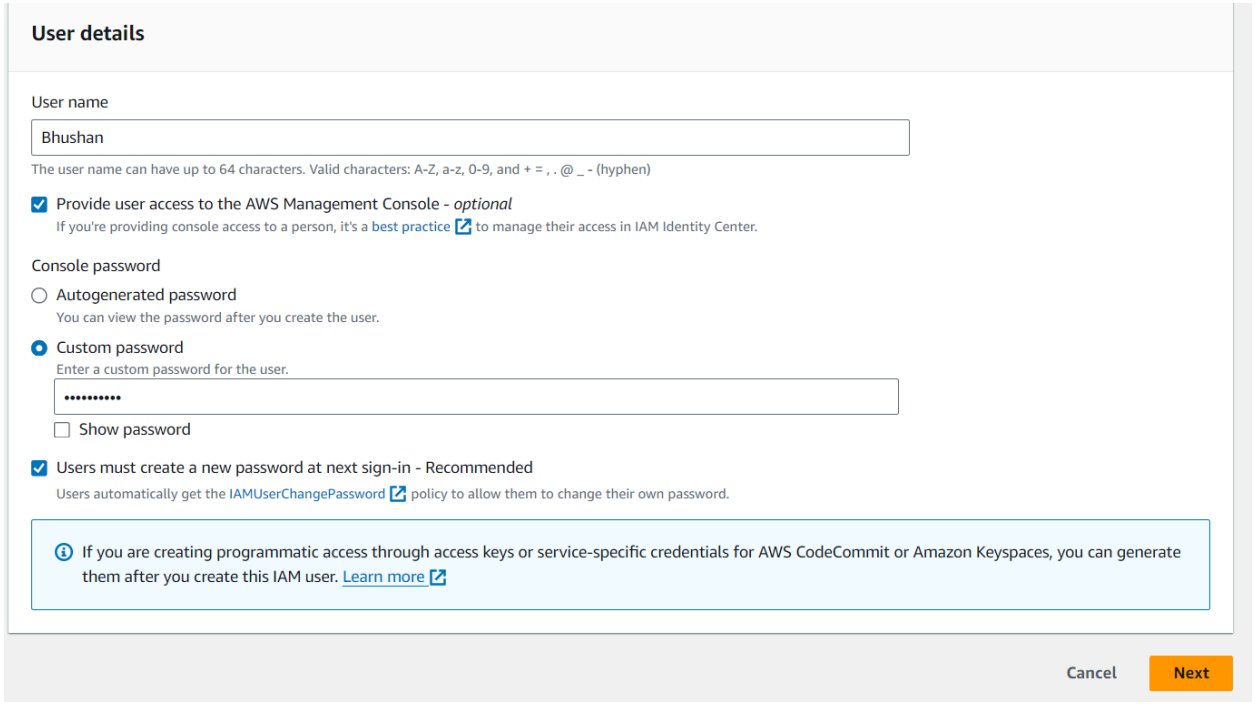


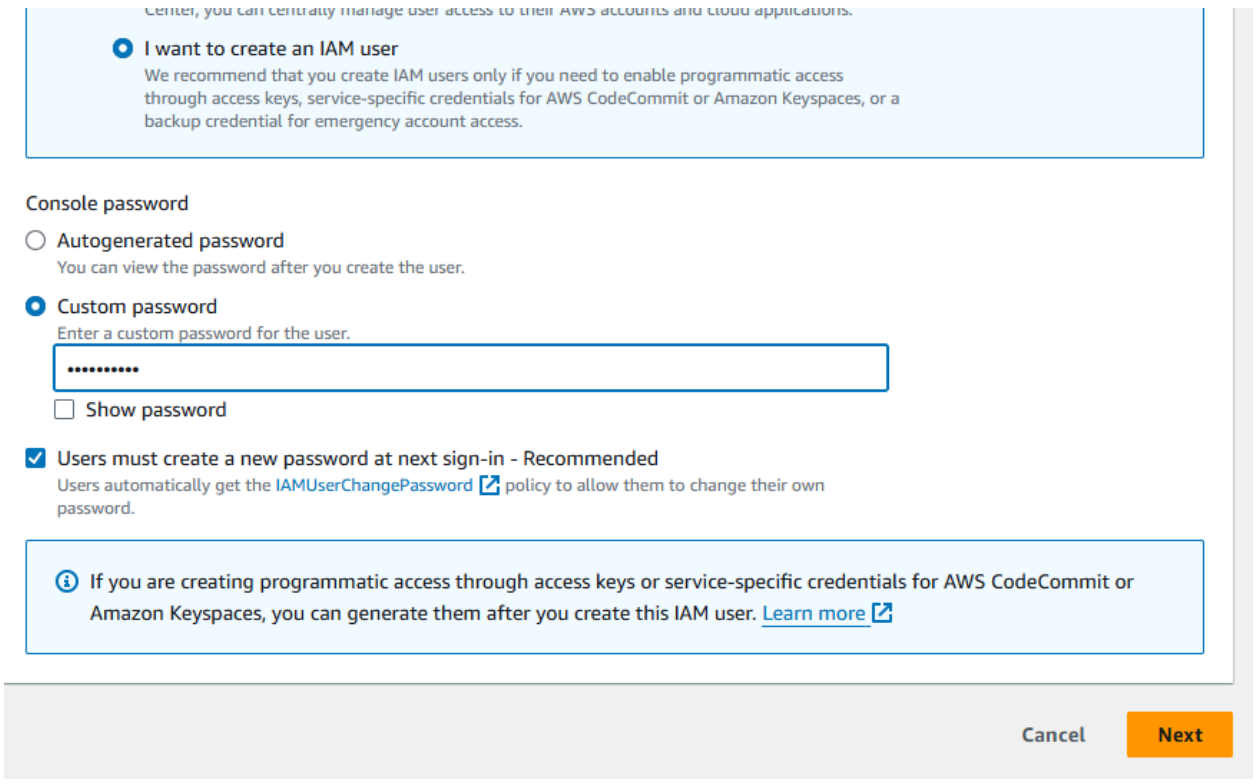
2. We have successfully setup and launched our Cloud9 environment. Over here, we can build and develop programs as per our desire. We are also allowed to collaborate with multiple other users and access shared resources.



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

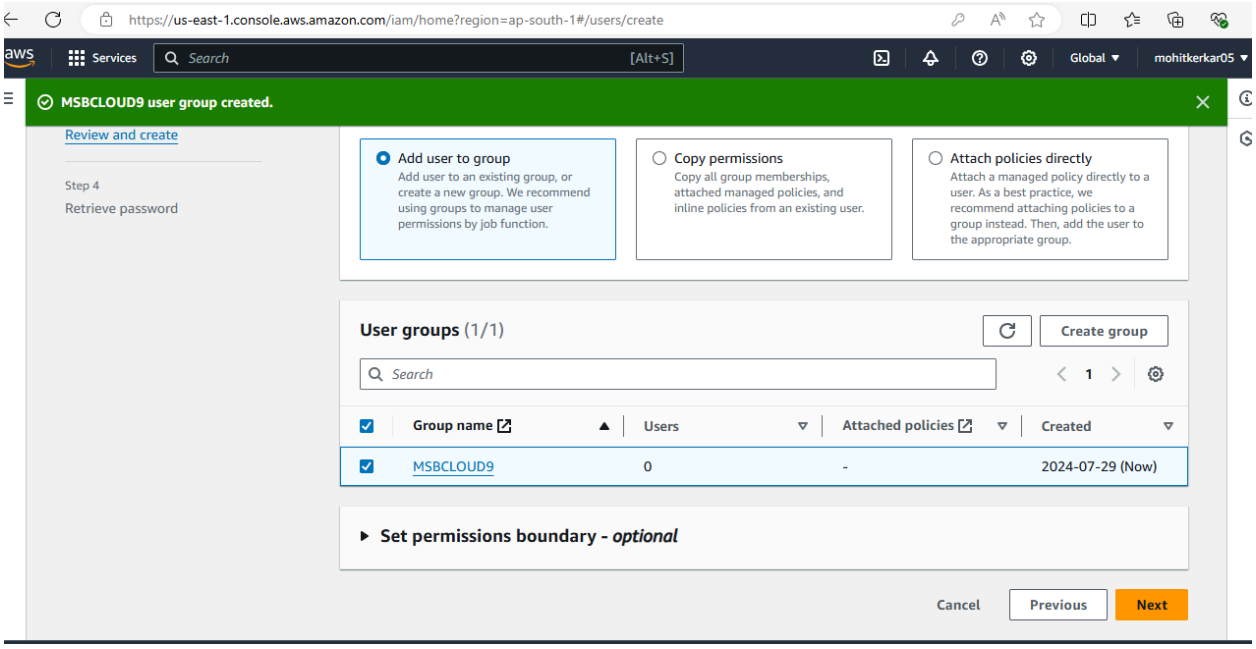
NAME: Bhushan Malpani DIV : D15C Roll no: 33



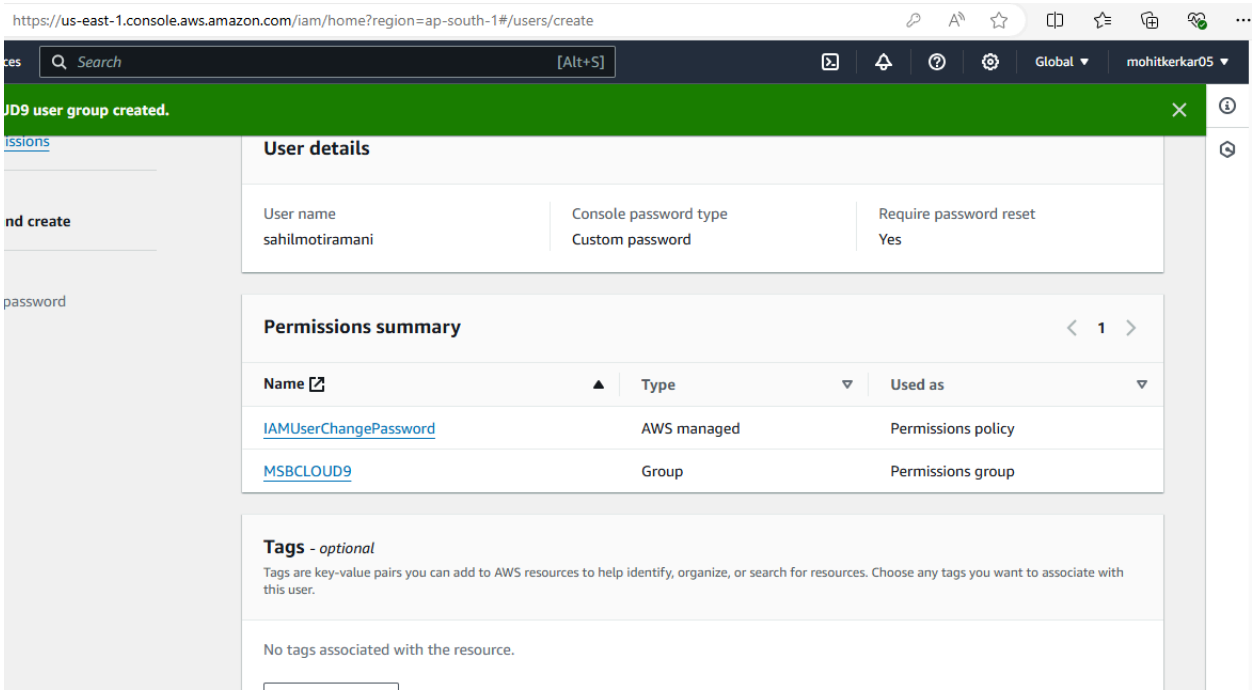


4. Similarly, create a new group and provide a suitable name for the same. Include the IAM users in this group together for our convenience i.e to provide similar kinds of permissions to the entire group rather than an individual user.

NAME: Bhushan Malpani DIV : D15C Roll no: 33

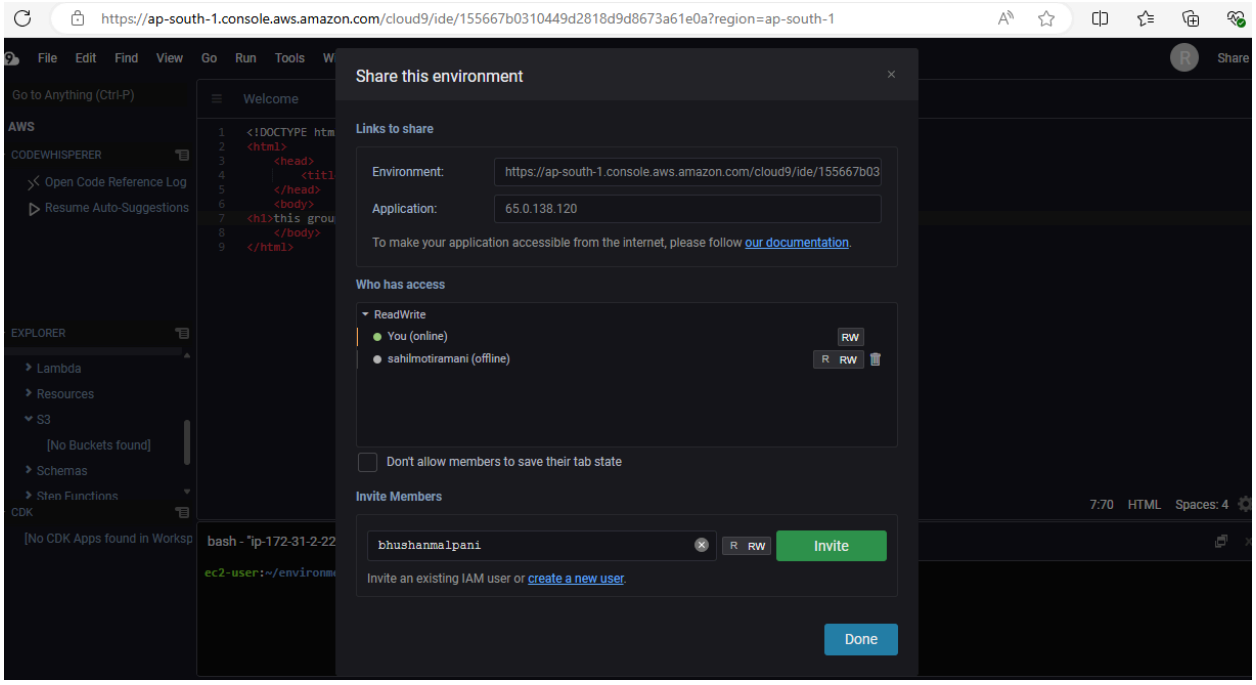


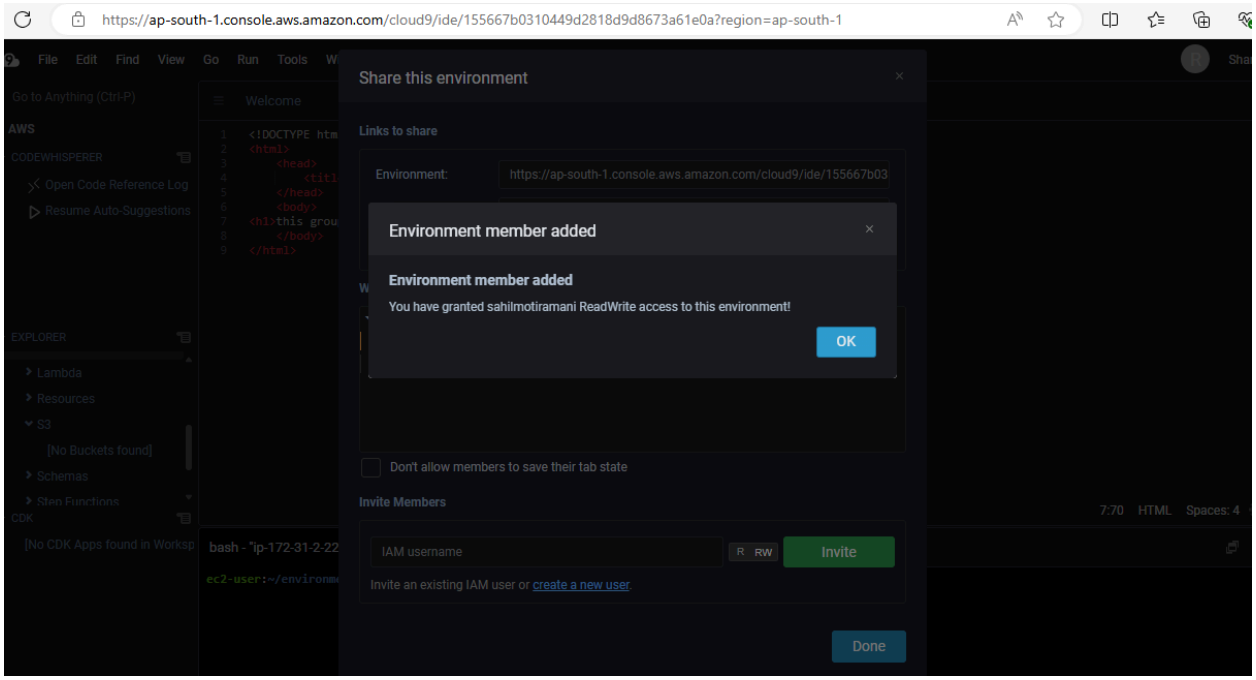
5. The user has successfully been created i.e There is a custom made username and a password for the IAM user.



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

NAME: Bhushan Malpani DIV : D15C Roll no: 33





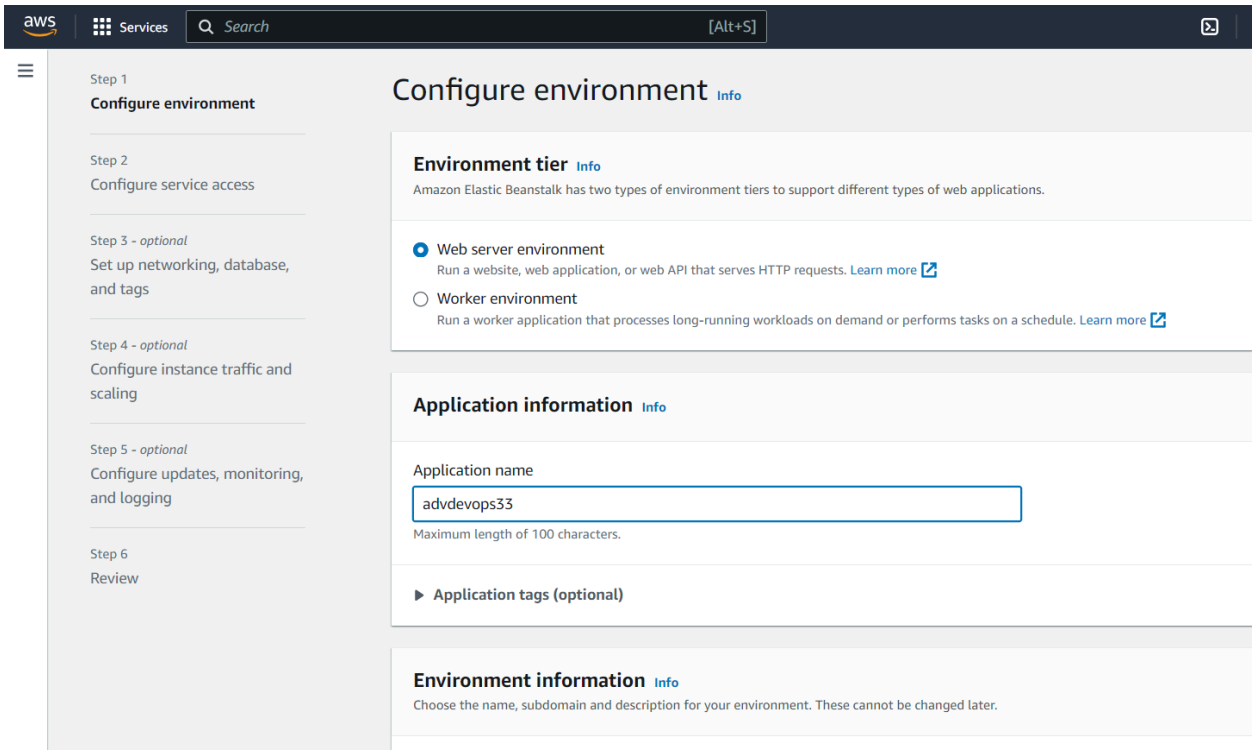
7. Further, we are supposed to login from another browser using the credentials of the IAM user, so as to access the shared cloud9 environment with us.
These steps could not be completed because Cloud9 services have been disrupted and there is no access to the IAM user from the remote login

Name:Bhushan Malpani Roll No. 33 Div:D15C

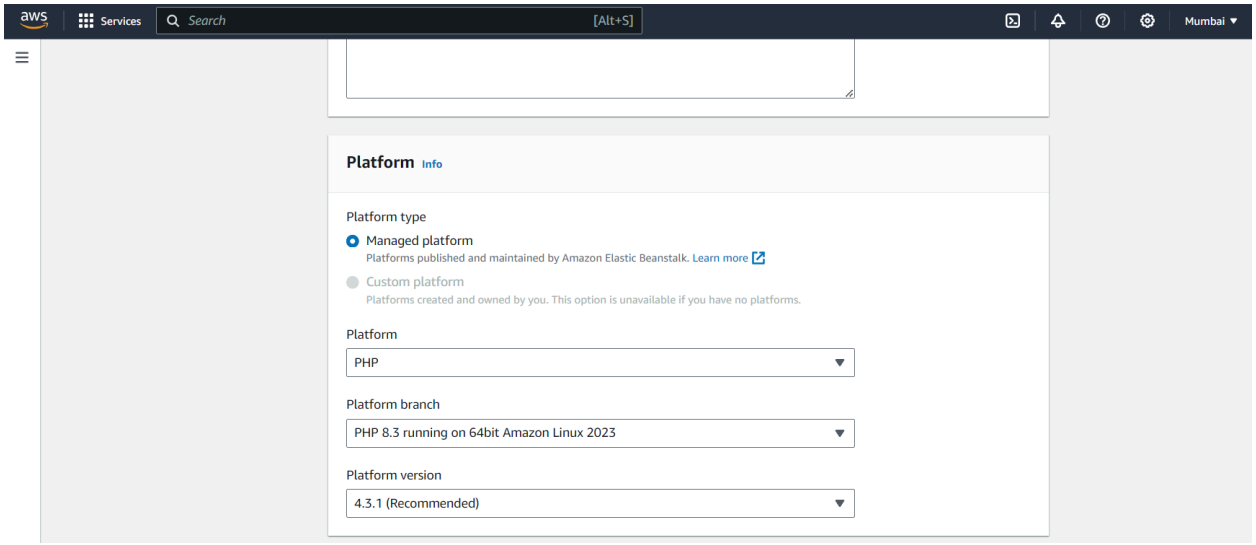
EXP 2

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

1. Create an Elastic Beanstalk environment. Give a suitable to your environment.

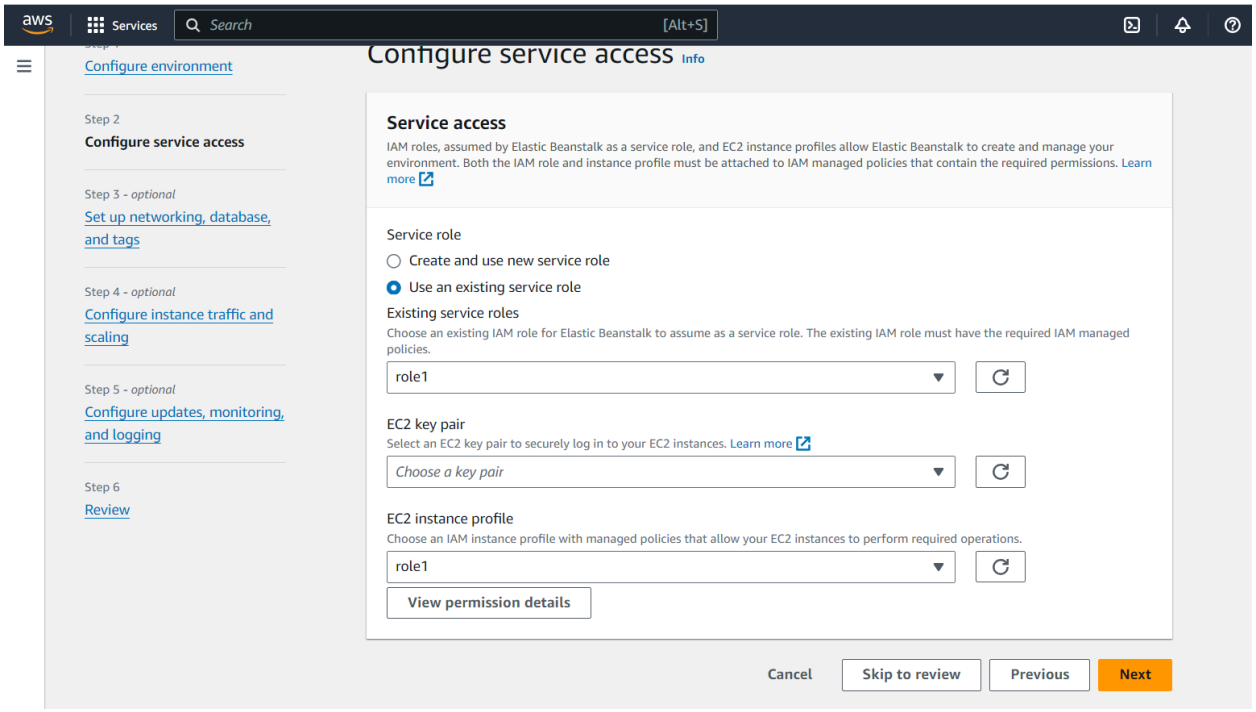


2. Select a suitable platform for your Elastic beanstalk environment. Here, we will select PHP file.

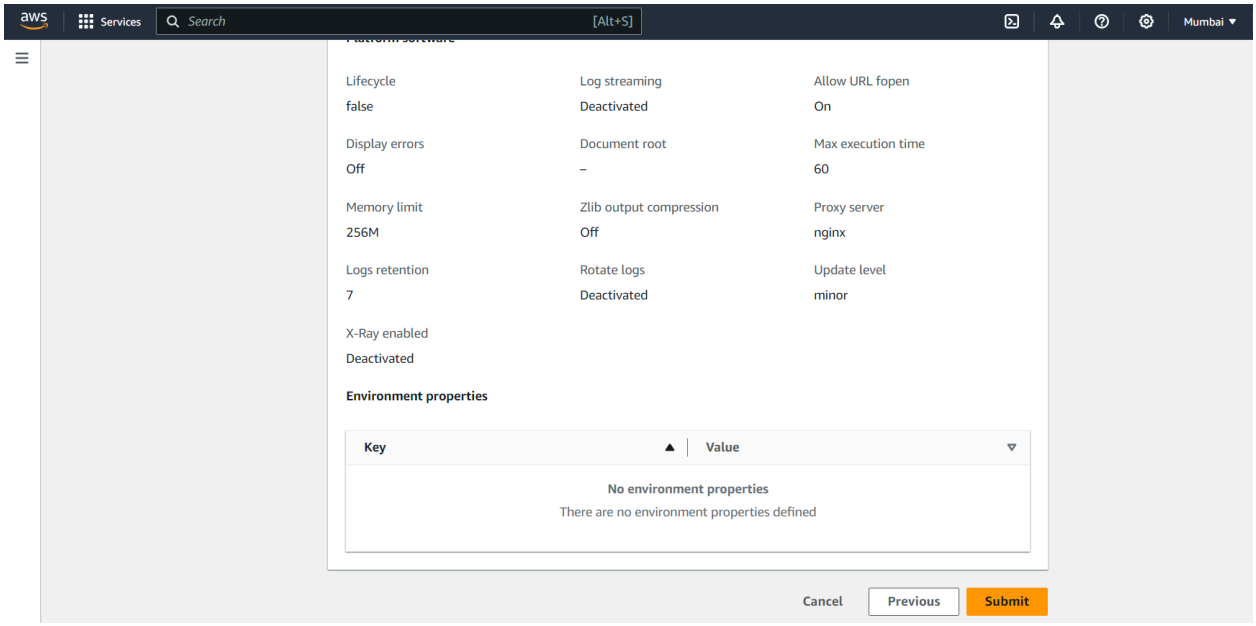


Name:Bhushan Malpani Roll No. 33 Div:D15C

3. Next, we will have to give access to our Elastic beanstalk environment to carry out it's tasks. For this, we have to grant certain permissions to the role, component or the entity that we will create.
   Following our permissions that we are supposed to grant to our role.
   - AWSElasticBeanStalkWebTier
   - AWSElasticBeanStalkWorkerTier
   - AWSElasticBeanStalkMulticontainerDocker

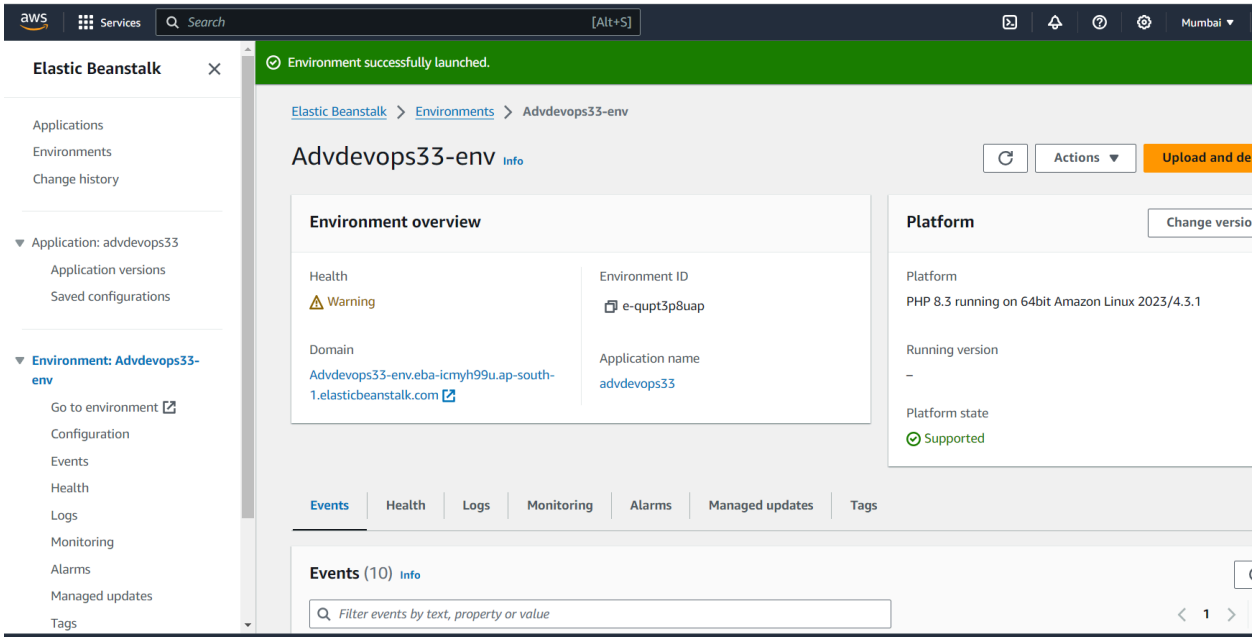


4. Click submit to create our Elastic beanstalk environment.



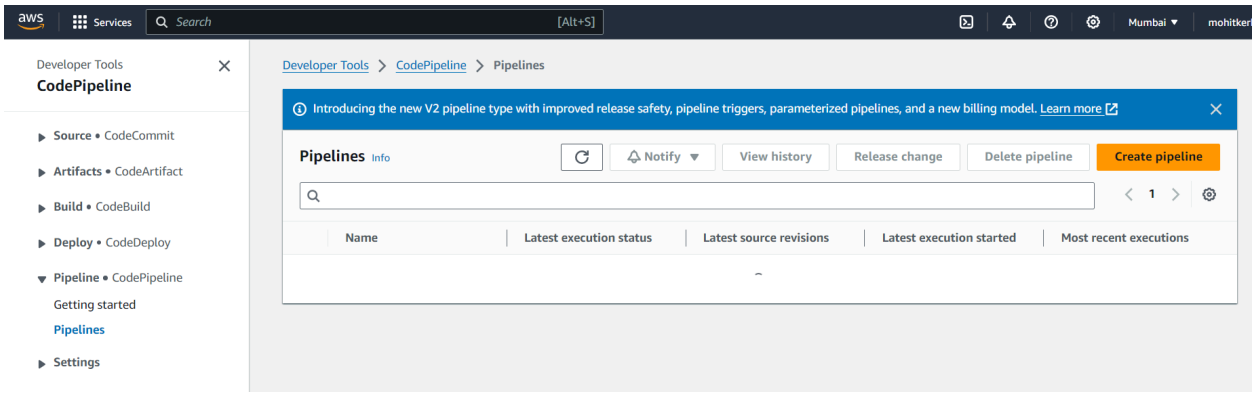
Name:Bhushan Malpani Roll No. 33 Div:D15C

5. When the creation is successful, we will see a message like this.



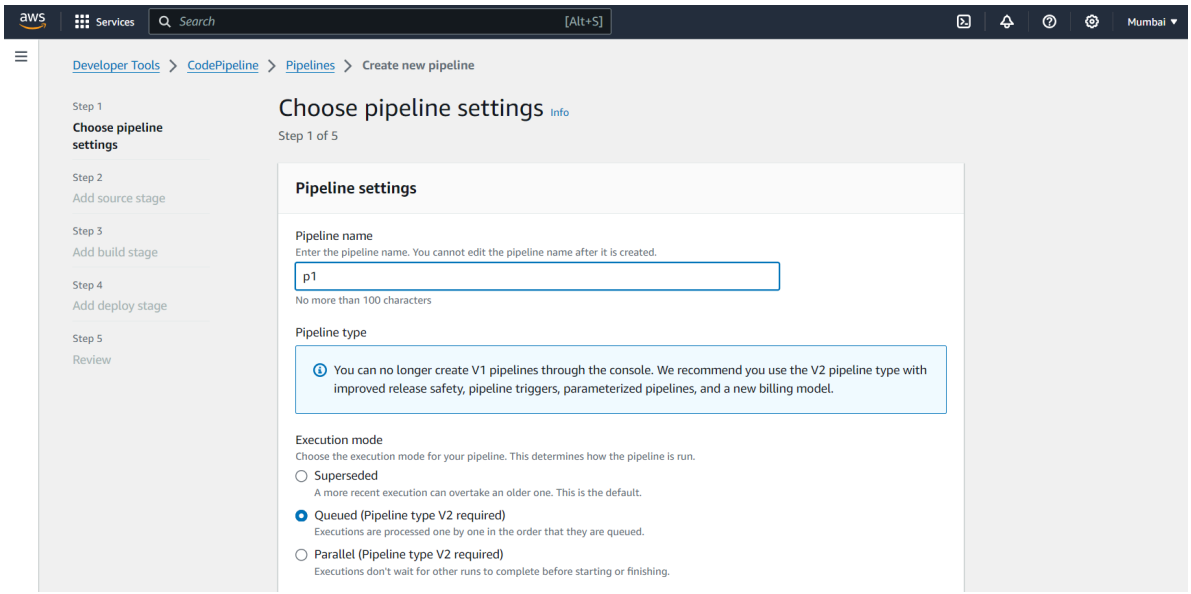
Now, We need to create a pipeline

6.To create a pipeline go to services →pipeline→create pipeline

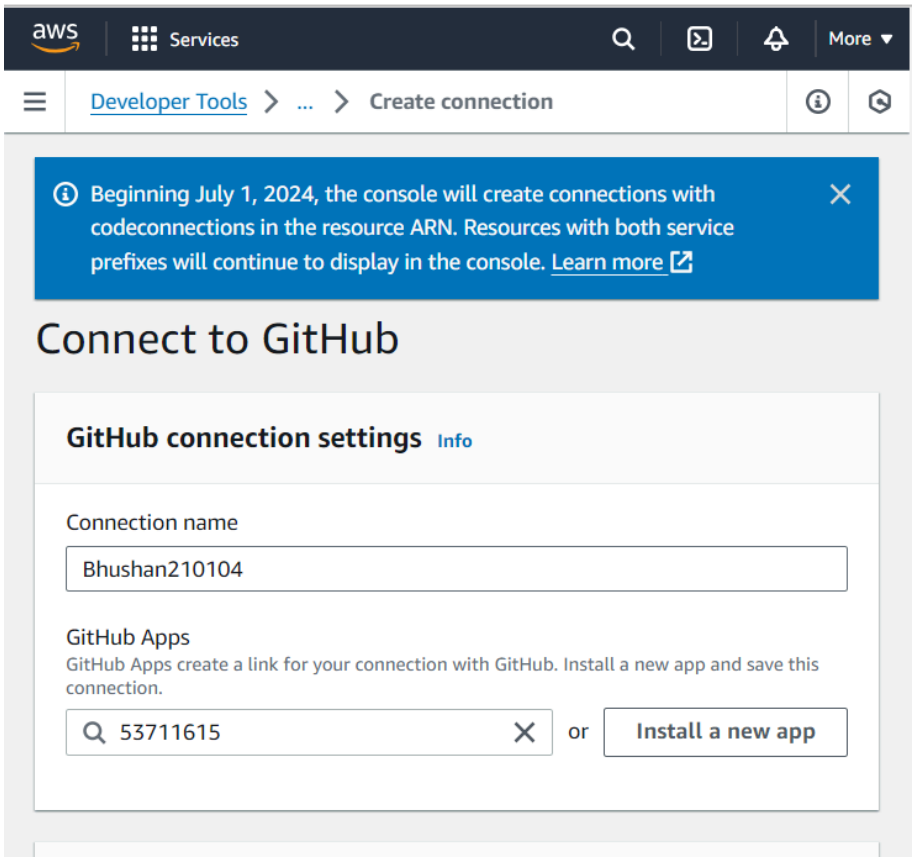


Name:Bhushan Malpani Roll No. 33 Div:D15C

7. Give name to the pipline

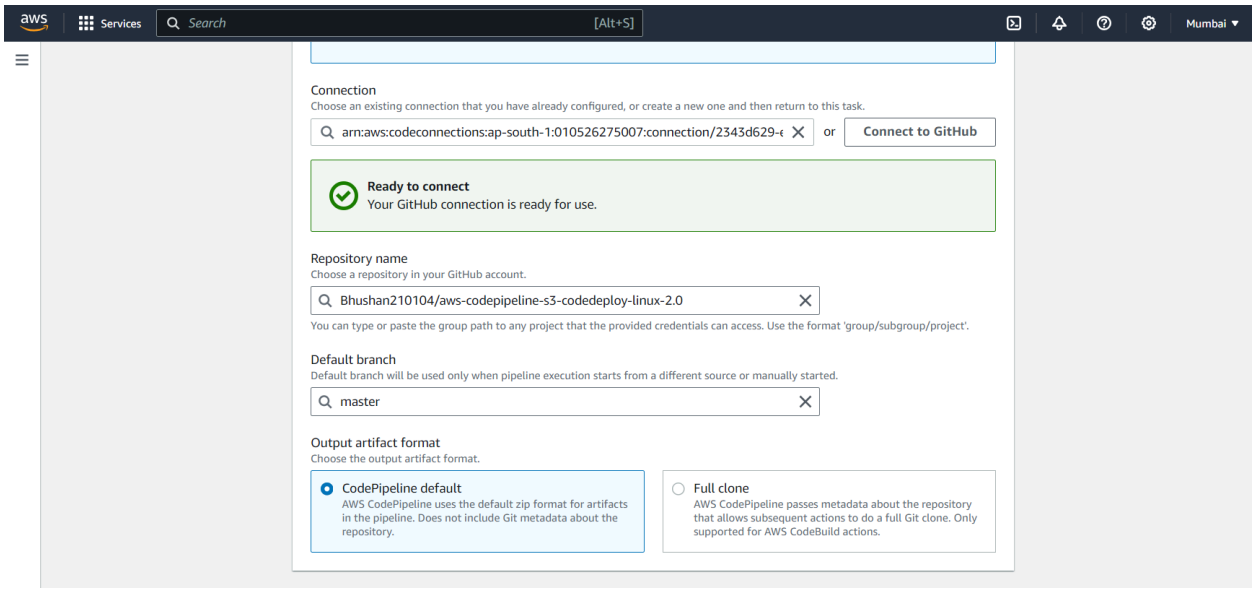


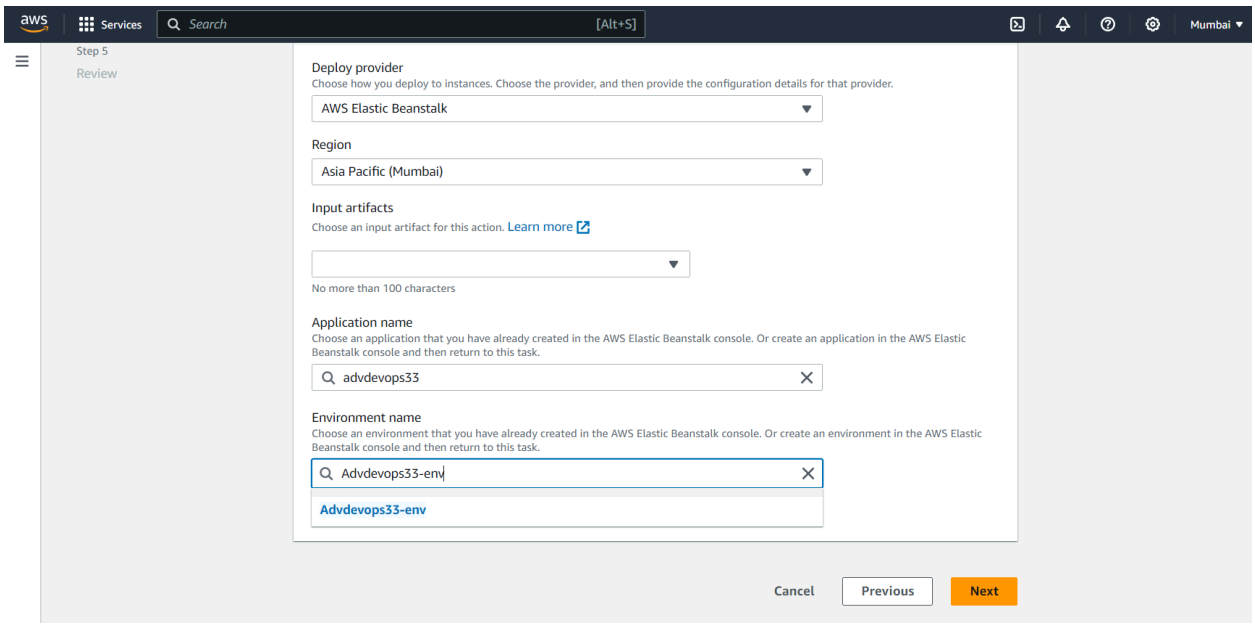
8. Connect it to git hub



Name:Bhushan Malpani Roll No. 33 Div:D15C

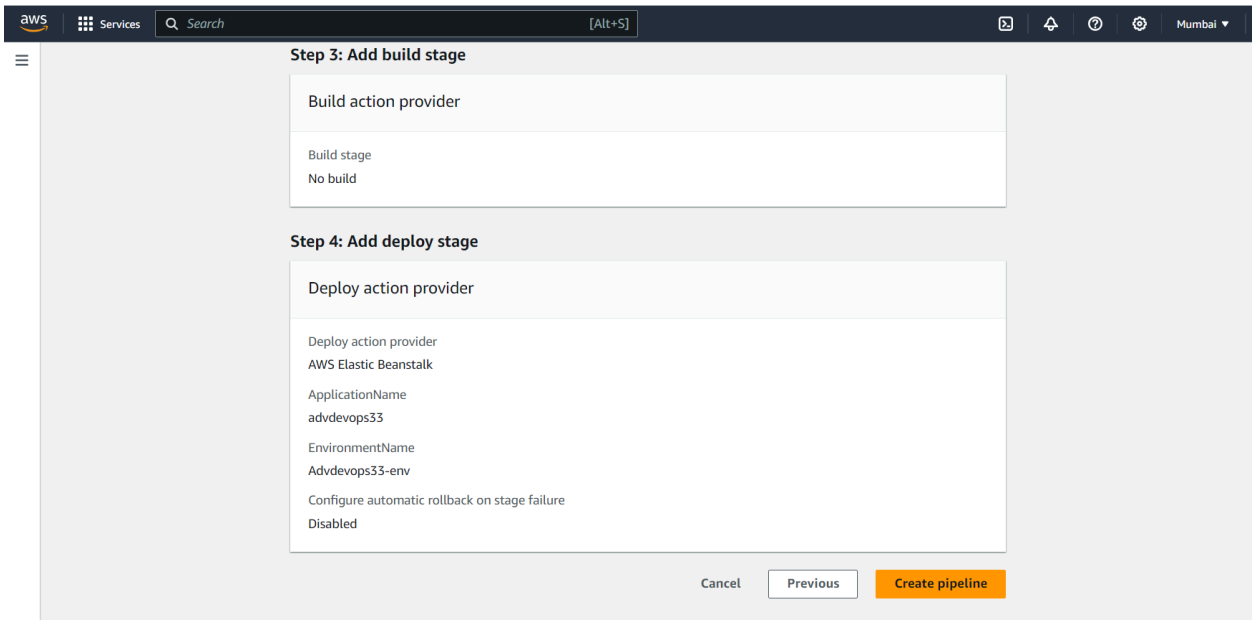
9. After connecting it to github select the repository and configure other settings



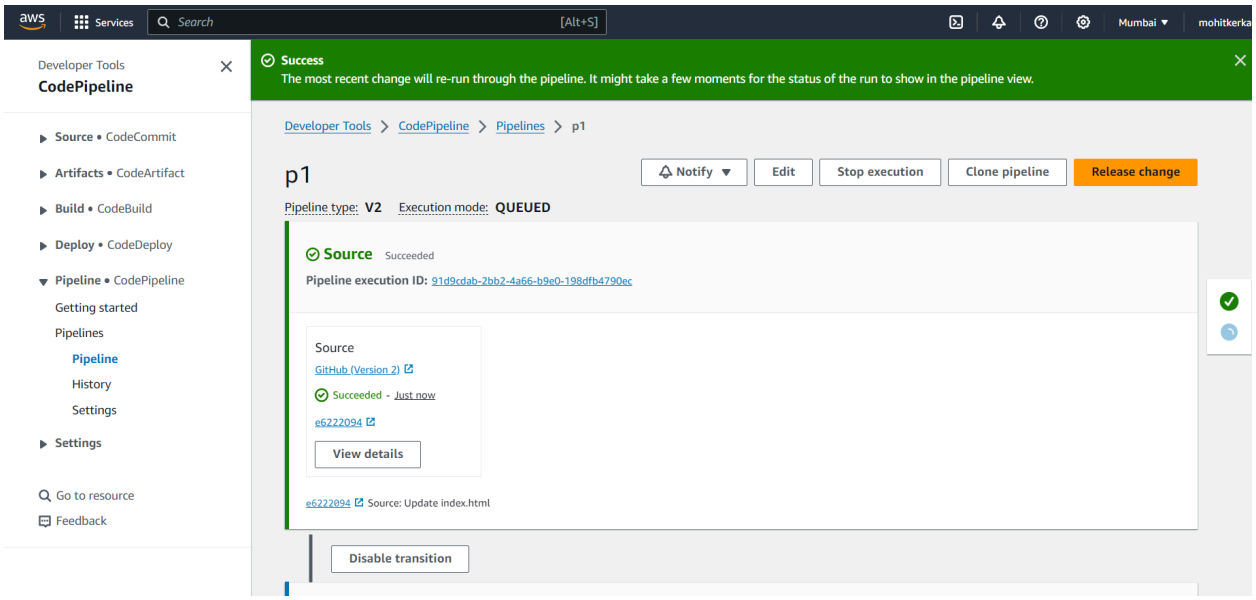


Name:Bhushan Malpani Roll No. 33 Div:D15C

10.Click on create pipline

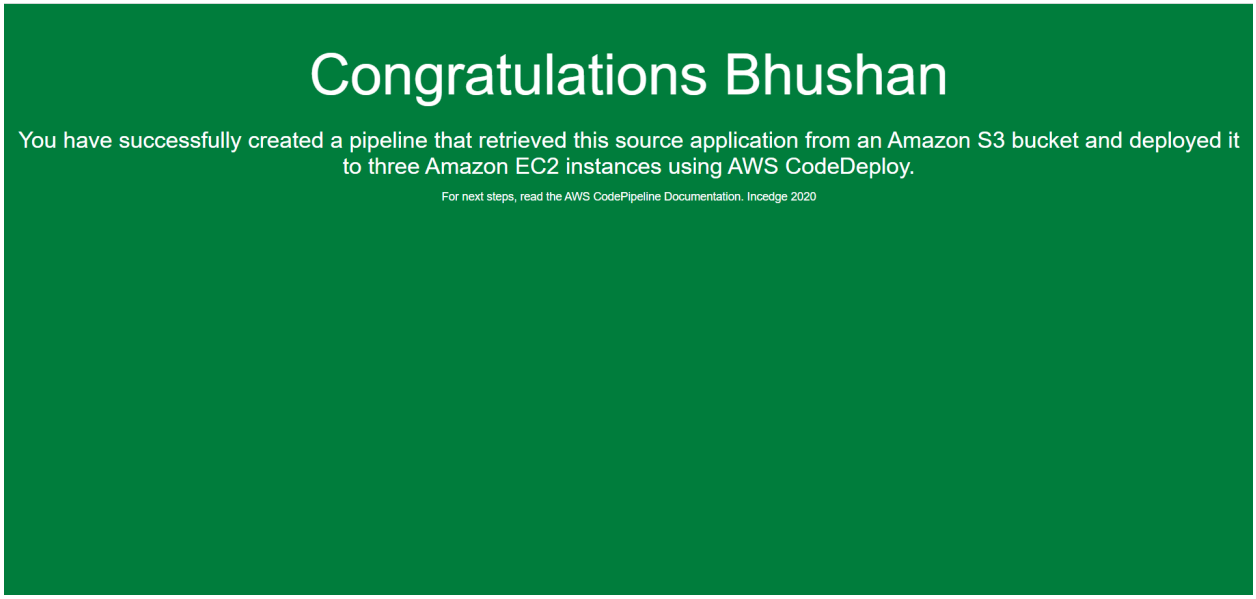


11.The pipeline will be created in few minutes



Name:Bhushan Malpani Roll No. 33 Div:D15C

12.The output will show after successful deployment.

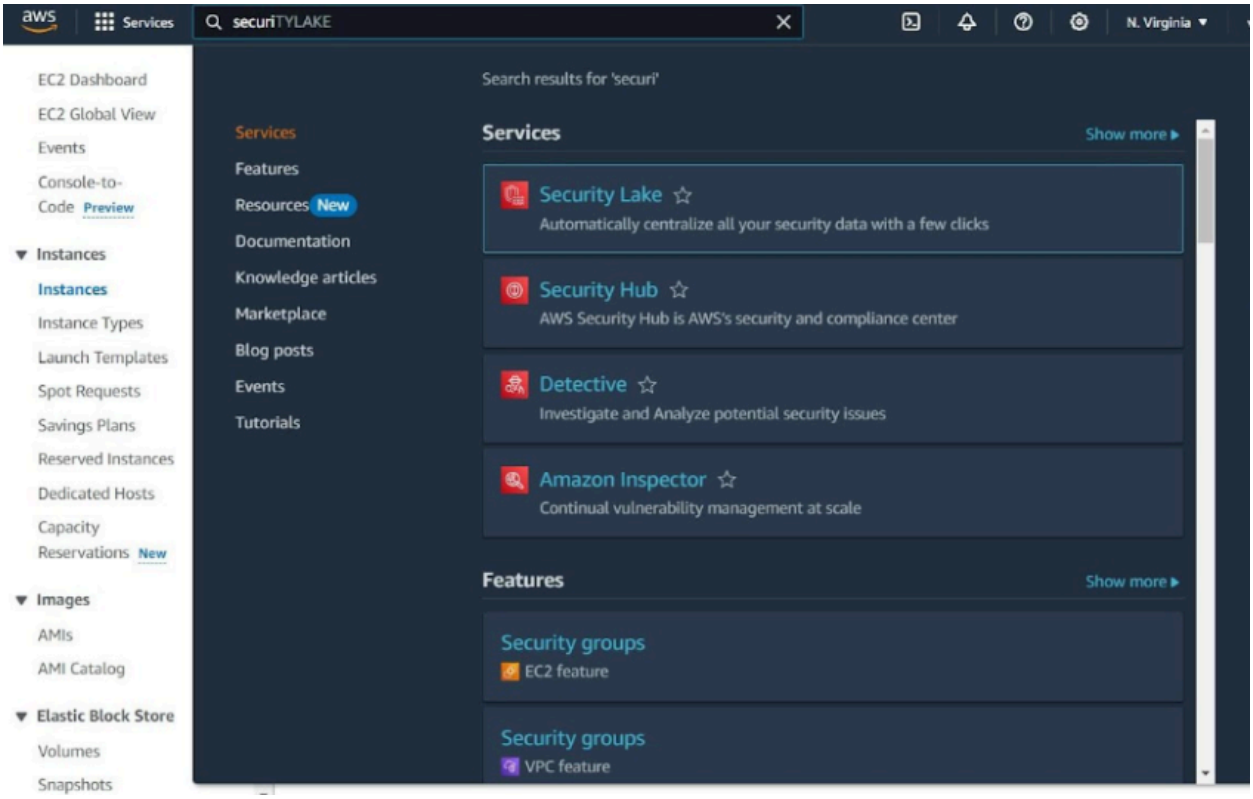


NAME:BHUSHAN MALPANI DIV: D15C ROLL NO:33

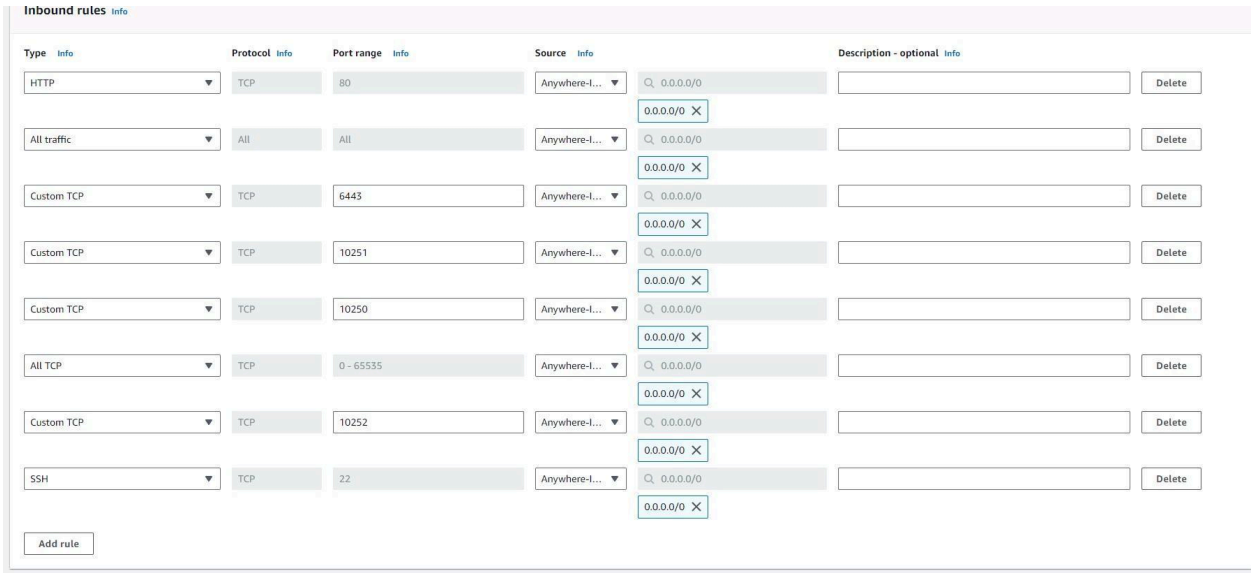
Aim: To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud Platforms.

Prerequisites:

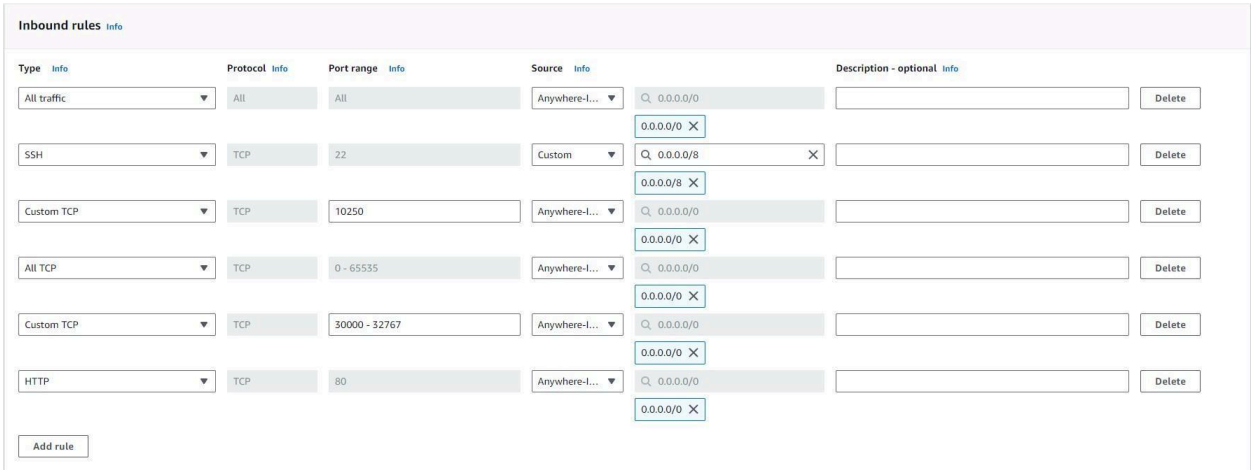
Create 2 security groups by navigating to Search → Security Groups.



Group 1: Master

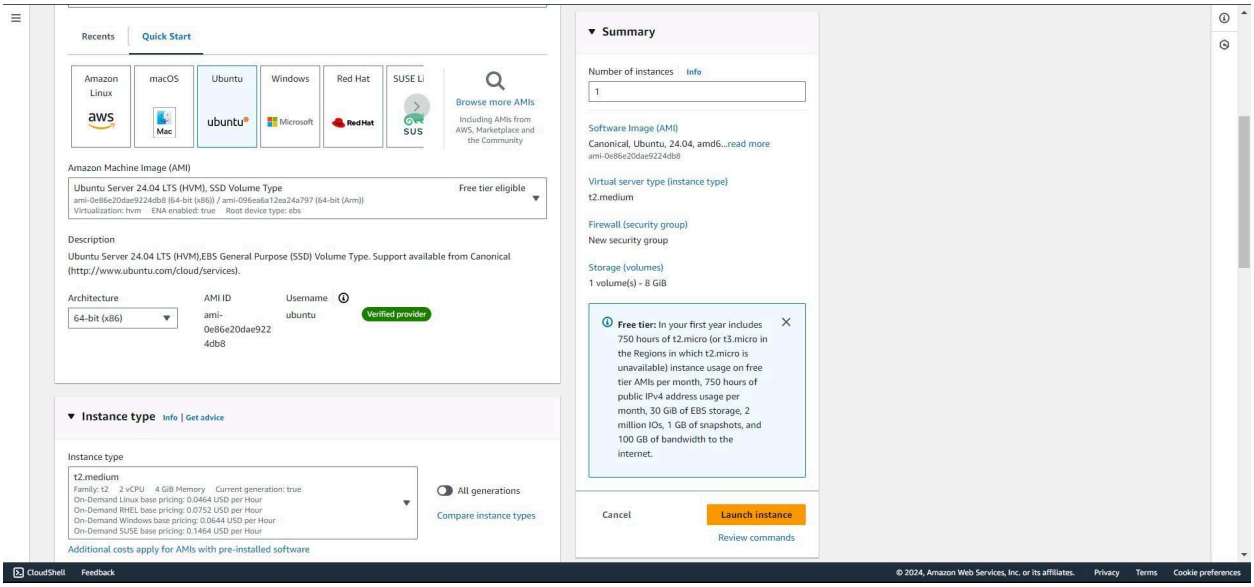


Group 2: Nodes



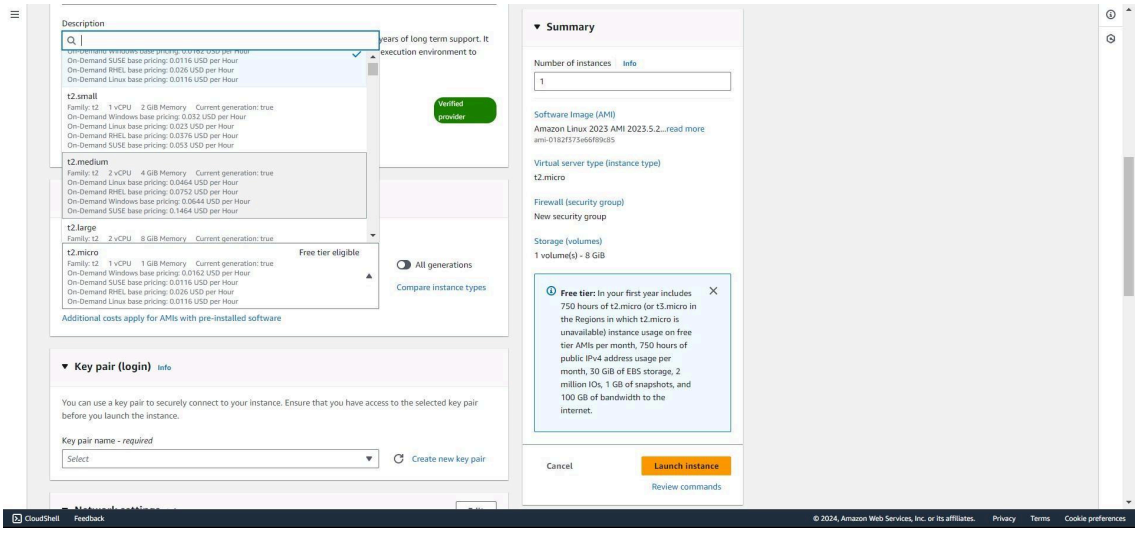
Step 1: Set Up EC2 Instances.

1) Set up 3 EC2 instances called master, node1, node2 Select Amazon Linux as the OS image.

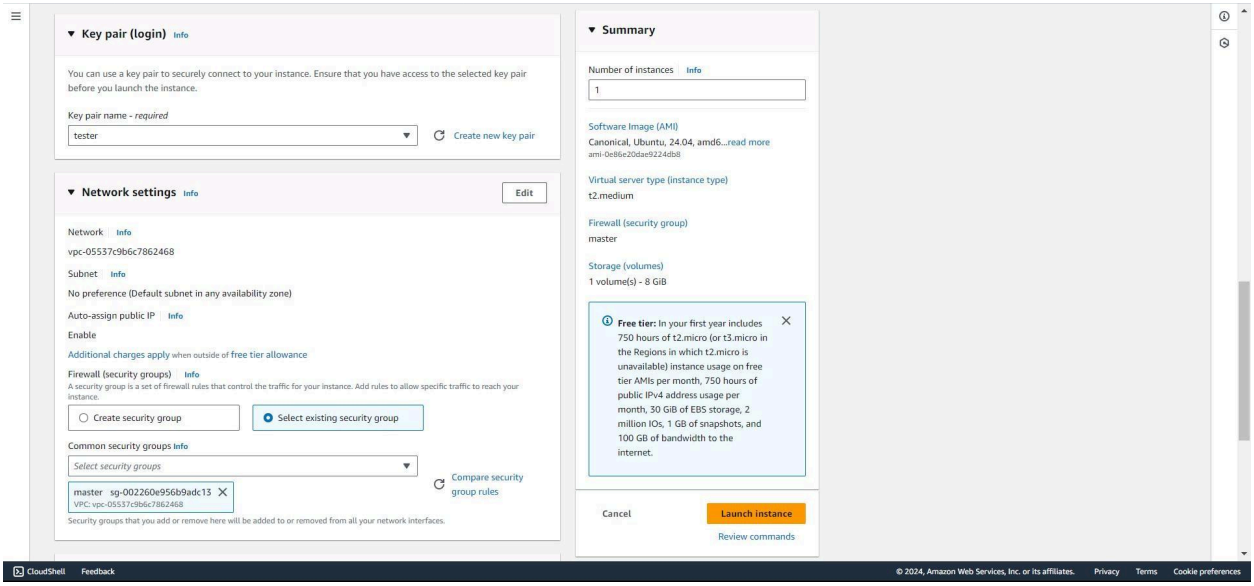


IMPORTANT: The default instance type and free one provided by AWS is t2.micro, which provides only 1CPU and 1 GiB of memory. For running Kubernetes, a minimum of 2 CPUs and 2GiB of RAM is required, hence change **t2.micro** to **t2.medium.**

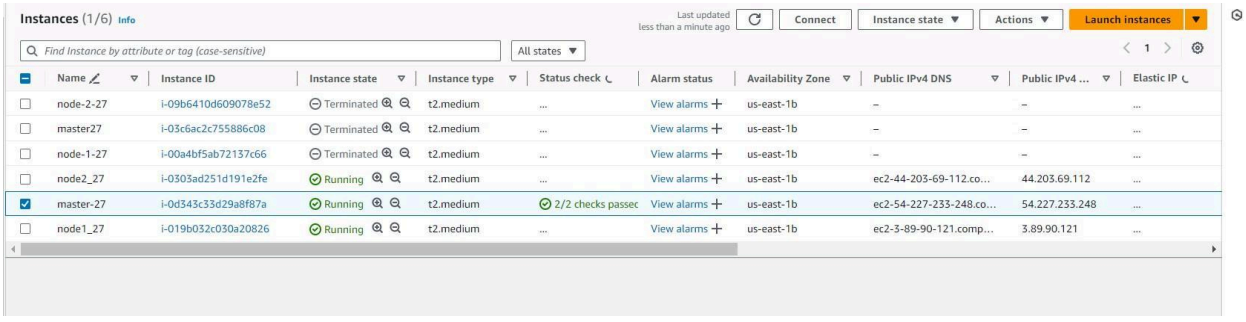
NAME:BHUSHAN MALPANI DIV: D15C ROLL NO:33



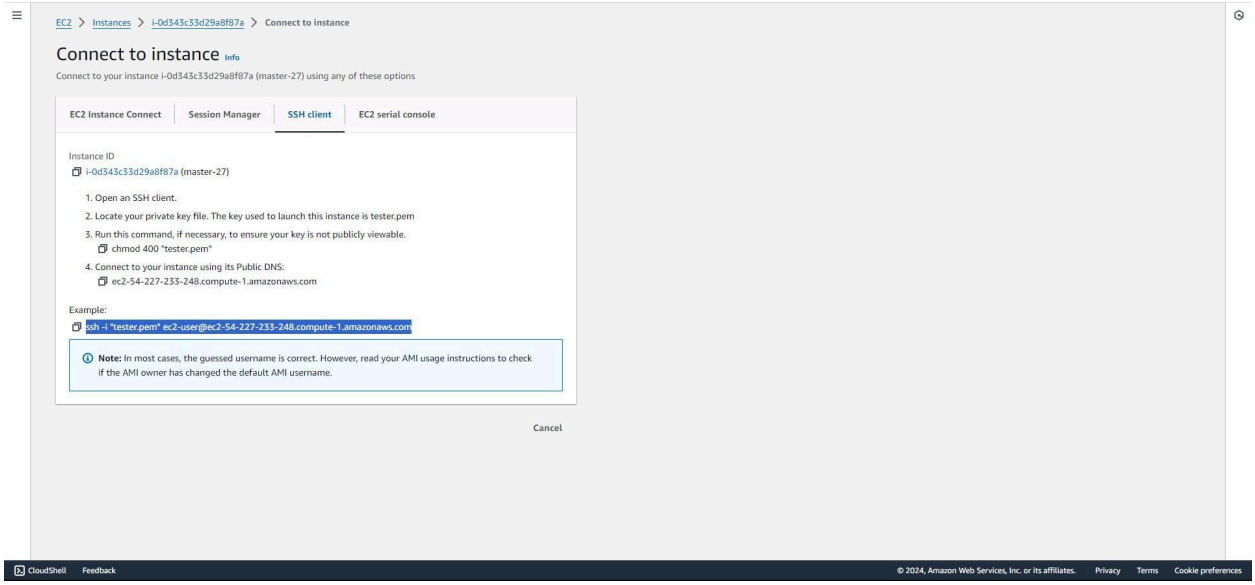
2) Create a key pair as you need the .pem file (private key file) on your system.
3) Click on ‘Select existing security group’ and select the master group for master instance, node group for both node instances.



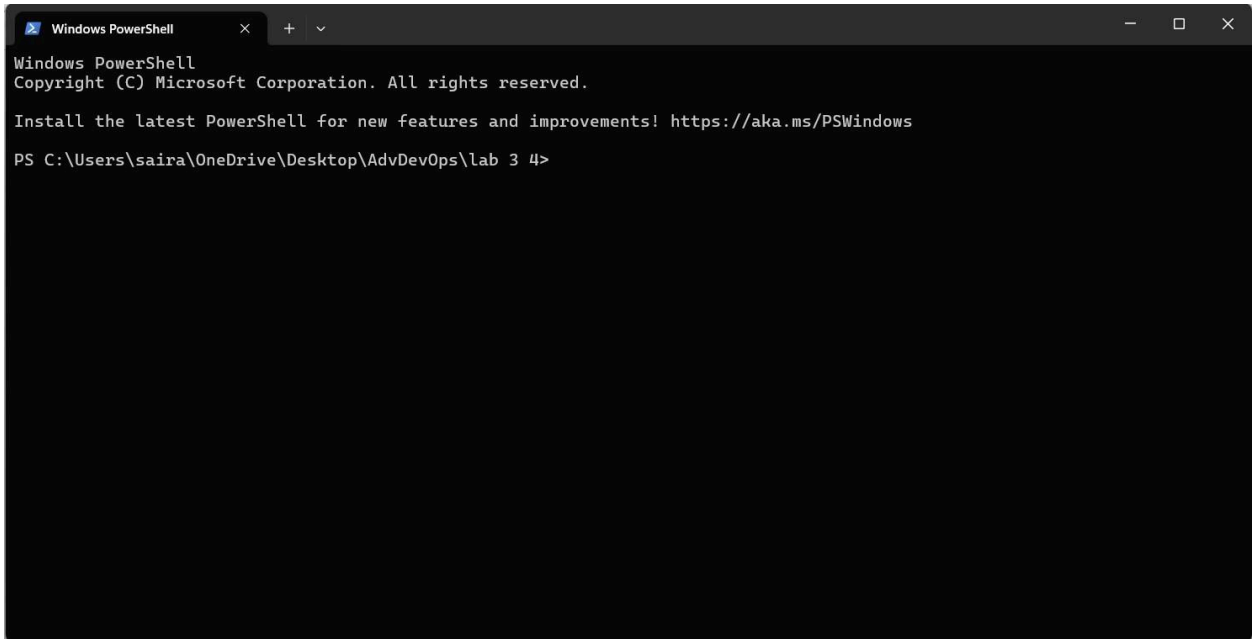
4) These are the instances that are created. Click on the Instance ID of master.



5) Click on Connect. This directs you to a connect dashboard. Click on SSH client, you get a SSH command. Use this command to access the instance terminal on your local system.



6) Go to the folder where your private key file (.pem file) is installed. Right click → Open in terminal.



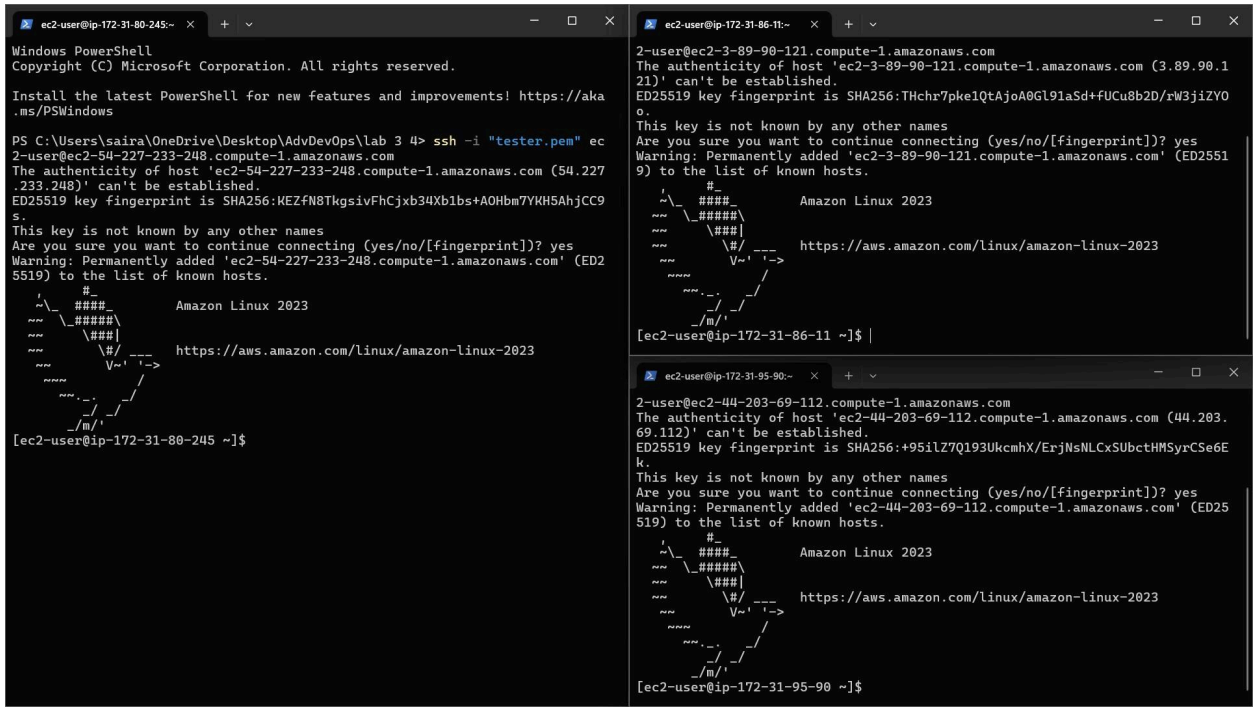
Paste the SSH command here and run it.

You might get the error of UNPROTECTED KEY FILE. This is because the .pem access Is with all users of teh system. Run the following commands to change the access to only the current user.

- icacls "C:\Users\saira\OneDrive\Desktop\AdvDevOps\lab 3 4\tester.pem" /inheritance:r
- icacls "C:\Users\saira\OneDrive\Desktop\AdvDevOps\lab 3 4\tester.pem" /grant:r "%USERNAME%:F"

Now rerun the SSH command.

7) After doing the above steps, the terminal for all 3 nodes can be seen.



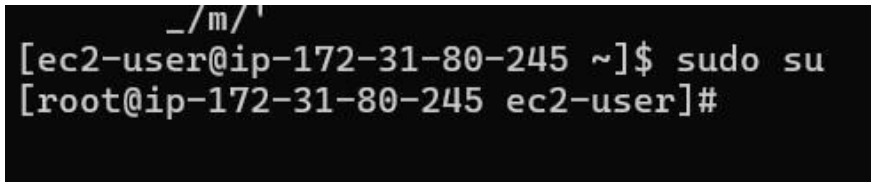
PERFORM THE FOLLOWING STEPS ON ALL 3 MACHINES

Step 2: Installation of Docker

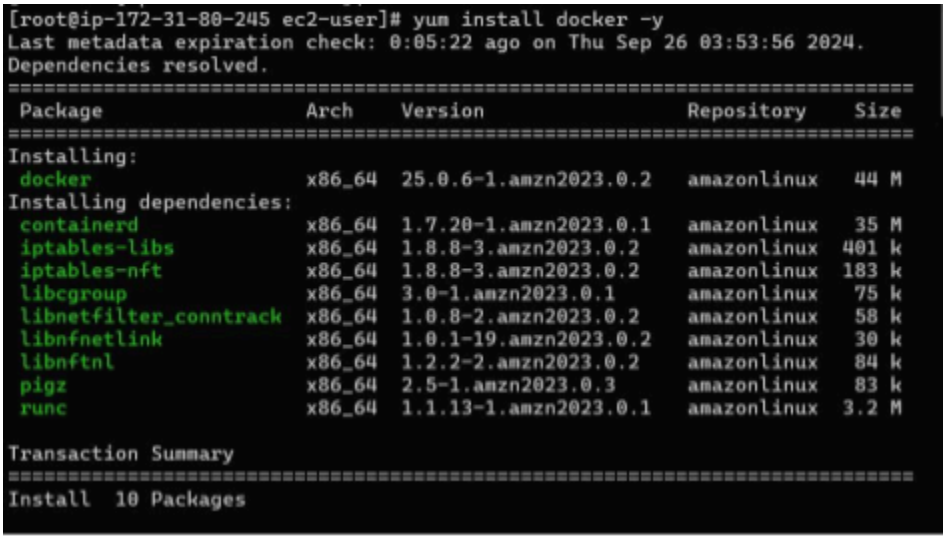
1) Use command

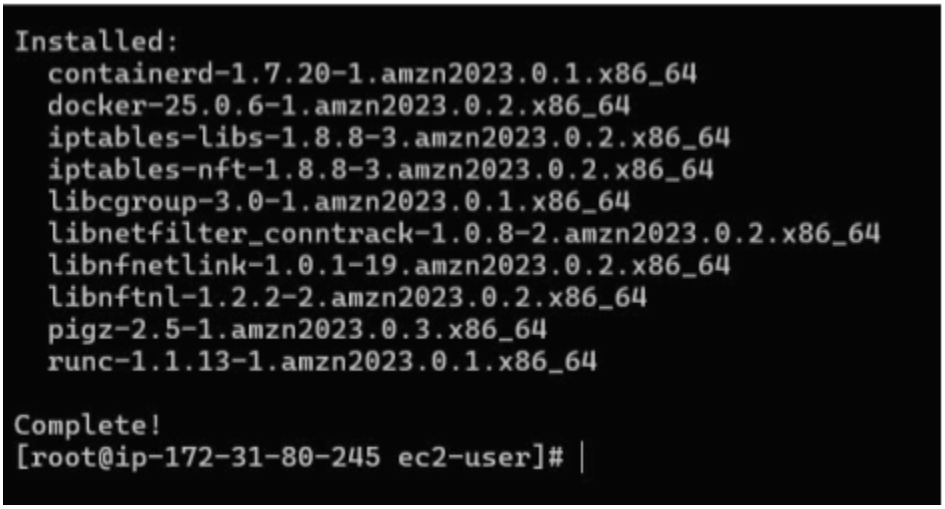
‘sudo su’

This allows you to act as the root user of the terminal



2) We can install docker using YUM(Yellowdog Updater, Modified). Use the command ‘yum install docker -y’





3) Now, configure a daemon.json file by using the following chain of commands.

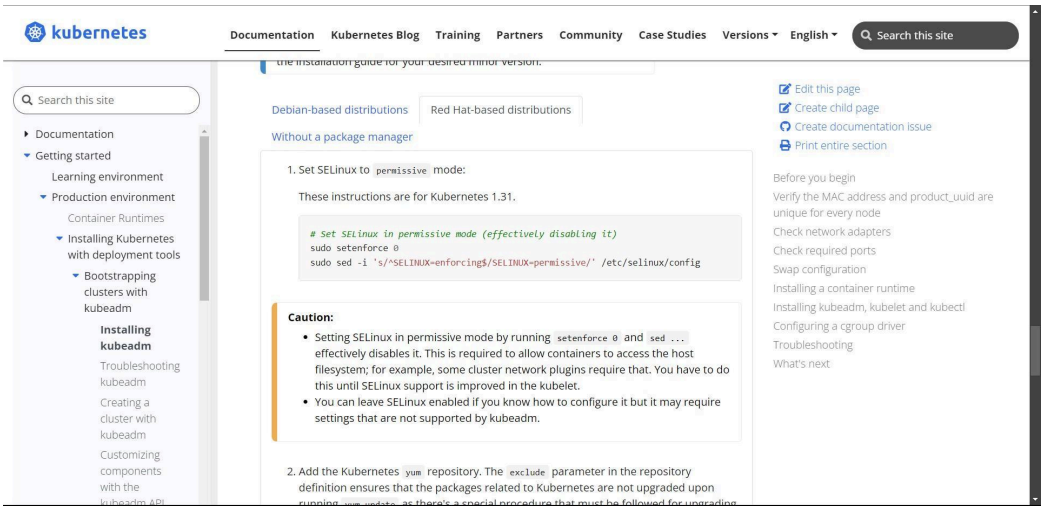
- cd /etc/docker
- cat <<EOF | sudo tee /etc/docker/daemon.json
  {
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
  "max-size": "100m"
  },
  "storage-driver": "overlay2"
  }
  EOF
- sudo systemctl enable docker

- sudo systemctl daemon-reload ● sudo systemctl restart docker

```
[root@ip-172-31-80-245 ec2-user]# cd /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"],
"log-driver": "json-file",
"log-opts": {
"max-size": "100m"
},
"storage-driver": "overlay2"
}
EOF
sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker
{
"exec-opts": ["native.cgroupdriver=systemd"],
"log-driver": "json-file",
"log-opts": {
"max-size": "100m"
},
"storage-driver": "overlay2"
}
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service →
 /usr/lib/systemd/system/docker.service.
[root@ip-172-31-80-245 docker]#
```

Step 3: Installing Kubernetes

1) For installing kubernetes, we will be using kubeadm, a framework used for creating kubernetes clusters using command line.
   https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/ The following will be visibe when you visit the website.



2) Select **red hat-based distributions** as amazon linux is based on red hat.

   sudo setenforce 0

→ sets SELinux to permissive mode

sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config → edits the SELinux configuration file (/etc/selinux/config) to make the change persistent across reboots. If not used, SELinux reverts to enforcing mode after reboot.

Setting SELinux to permissive mode during Kubernetes installation prevents permission-related issues with container runtimes and components that may not function correctly under SELinux's enforcing policies.

Run the following commands:

- sudo setenforce 0
- sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config ● cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo

    [kubernetes]
    name=Kubernetes
    baseurl=https://pkgs.k8s.io/core:/stable:/v1.31/
    rpm/ enabled=1 gpgcheck=1
    gpgkey=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/repodata/repomd.xml.key
    exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
    EOF

This comamnd is a repository script to create a kubernetes repository

```
/usr/lib/systemd/system/docker.service.
[root@ip-172-31-80-245 docker]# sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
[root@ip-172-31-80-245 docker]# cat <<EOF | sudo tee /etc/yum.repos.d/kubern
etes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
[root@ip-172-31-80-245 docker]#
```

- yum repolist

This command shows the repositories created on the machine.

```
[root@ip-172-31-80-245 docker]# yum repolist
repo id                    repo name
amazonlinux                Amazon Linux 2023 repository
kernel-livepatch           Amazon Linux 2023 Kernel Livepatch repository
kubernetes                 Kubernetes
[root@ip-172-31-80-245 docker]#
```

Next step is to install kubelet, kubeadm, kubectl

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

```
[root@ip-172-31-80-245 docker]# sudo yum install -y kubelet kubeadm kubectl
--disableexcludes=kubernetes
Kubernetes                                      65 kB/s | 9.4 kB     00:00
Dependencies resolved.
================================================================================
 Package                 Arch    Version                 Repository    Size
================================================================================
Installing:
 kubeadm                 x86_64  1.31.1-150500.1.1       kubernetes    11 M
 kubectl                 x86_64  1.31.1-150500.1.1       kubernetes    11 M
 kubelet                 x86_64  1.31.1-150500.1.1       kubernetes    15 M
Installing dependencies:
 conntrack-tools         x86_64  1.4.6-2.amzn2023.0.2    amazonlinux  208 k
 cri-tools               x86_64  1.31.1-150500.1.1       kubernetes   6.9 M
 kubernetes-cni          x86_64  1.5.1-150500.1.1        kubernetes   7.1 M
 libnetfilter_cthelper   x86_64  1.0.0-21.amzn2023.0.2   amazonlinux   24 k
 libnetfilter_cttimeout  x86_64  1.0.0-19.amzn2023.0.2   amazonlinux   24 k
 libnetfilter_queue      x86_64  1.0.5-2.amzn2023.0.2    amazonlinux   30 k

Transaction Summary
================================================================================
```

```
Installed:
  conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64
  cri-tools-1.31.1-150500.1.1.x86_64
  kubeadm-1.31.1-150500.1.1.x86_64
  kubectl-1.31.1-150500.1.1.x86_64
  kubelet-1.31.1-150500.1.1.x86_64
  kubernetes-cni-1.5.1-150500.1.1.x86_64
  libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64
  libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64
  libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64

Complete!
[root@ip-172-31-80-245 docker]#
```

Now, we need to enable the kubelet service. Run the command

- sudo systemctl enable --now kubelet

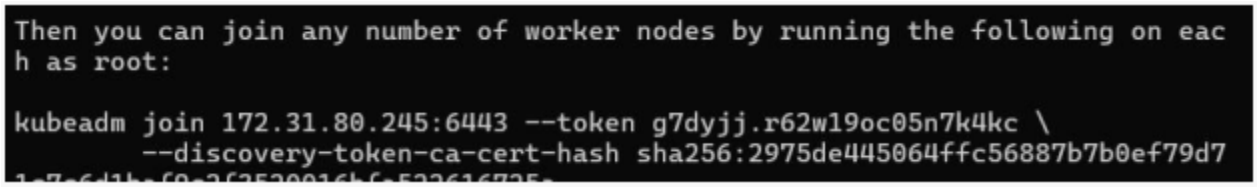
```
[root@ip-172-31-80-245 docker]# sudo systemctl enable --now kubelet
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service
→ /usr/lib/systemd/system/kubelet.service.
[root@ip-172-31-80-245 docker]#
```

NAME:BHUSHAN MALPANI DIV: D15C ROLL NO:33

PERFORM THE FOLLOWING ON ONLY THE MASTER MACHINE

1) Firstly, we need to initialize kubernetes. For this, run the command: ● kubeadm init

From the output, we will receive a command that is used to link the nodes to the master. Copy it and save it somewhere local.



2) From the output, we receive the following commands:

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

```
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
```

3) To check whether nodes are connected, run the command
   - kubectl get nodes

   This output shows only master is connected right now.

```
[root@ip-172-31-80-245 docker]# kubectl get nodes
NAME                            STATUS     ROLES           AGE    VERSION
ip-172-31-80-245.ec2.internal   NotReady   control-plane   112s   v1.31.1
[root@ip-172-31-80-245 docker]#
```

PERFORM THE FOLLOWING ONLY ON THE NODE MACHINES

Use the command that you had copied before and use them on the node machines.

```
root@ip-172-31-86-11:/etc/do
em get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/conf
ig.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/li
b/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-check] Waiting for a healthy kubelet at http://127.0.0.1:10248/heal
thz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 501.57413ms
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was recei
ved.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the clust
er.

[root@ip-172-31-86-11 docker]#
```

```
root@ip-172-31-95-90:/etc/dc
em get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/conf
ig.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/li
b/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-check] Waiting for a healthy kubelet at http://127.0.0.1:10248/heal
thz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 1.0012978465
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was recei
ved.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the clust
er.

[root@ip-172-31-95-90 docker]#
```

NOW GO BACK TO THE MASTER MACHINE AND RERUN ‘kubectl get nodes’

```
[root@ip-172-31-80-245 docker]# kubectl get nodes
NAME                              STATUS     ROLES           AGE     VERSION
ip-172-31-80-245.ec2.internal     NotReady   control-plane   2m18s   v1.31.1
ip-172-31-86-11.ec2.internal      NotReady   <none>          9s      v1.31.1
ip-172-31-95-90.ec2.internal      NotReady   <none>          2s      v1.31.1
[root@ip-172-31-80-245 docker]#
```

As we see, the status of the nodes are <NOT READY>. To change it, we need to install a network CNI plugin.

Use the following command only on the master machine::

kubectl apply -f

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

Now run ‘kubectl get nodes’ again.

```
[root@ip-172-31-80-245 docker]# kubectl get nodes
NAME                              STATUS   ROLES           AGE     VERSION
ip-172-31-80-245.ec2.internal     Ready    control-plane   9m33s   v1.31.1
ip-172-31-86-11.ec2.internal      Ready    <none>          7m24s   v1.31.1
ip-172-31-95-90.ec2.internal      Ready    <none>          7m17s   v1.31.1
```

Conclusion:

In this experiment, we have learned how to create kubernetes clusters on a linux terminal, how the ssh command works on a local terminal and what requirements are necessary to create kubernetes clusters. We have used many command inline tools of Kubernetes like kubecmd, kubectl to set up the clusters and work with docker container to perform the connection of the nodes with master machine.

NAME: Bhushan Malpani **DIV: D15C** **Roll No:33**

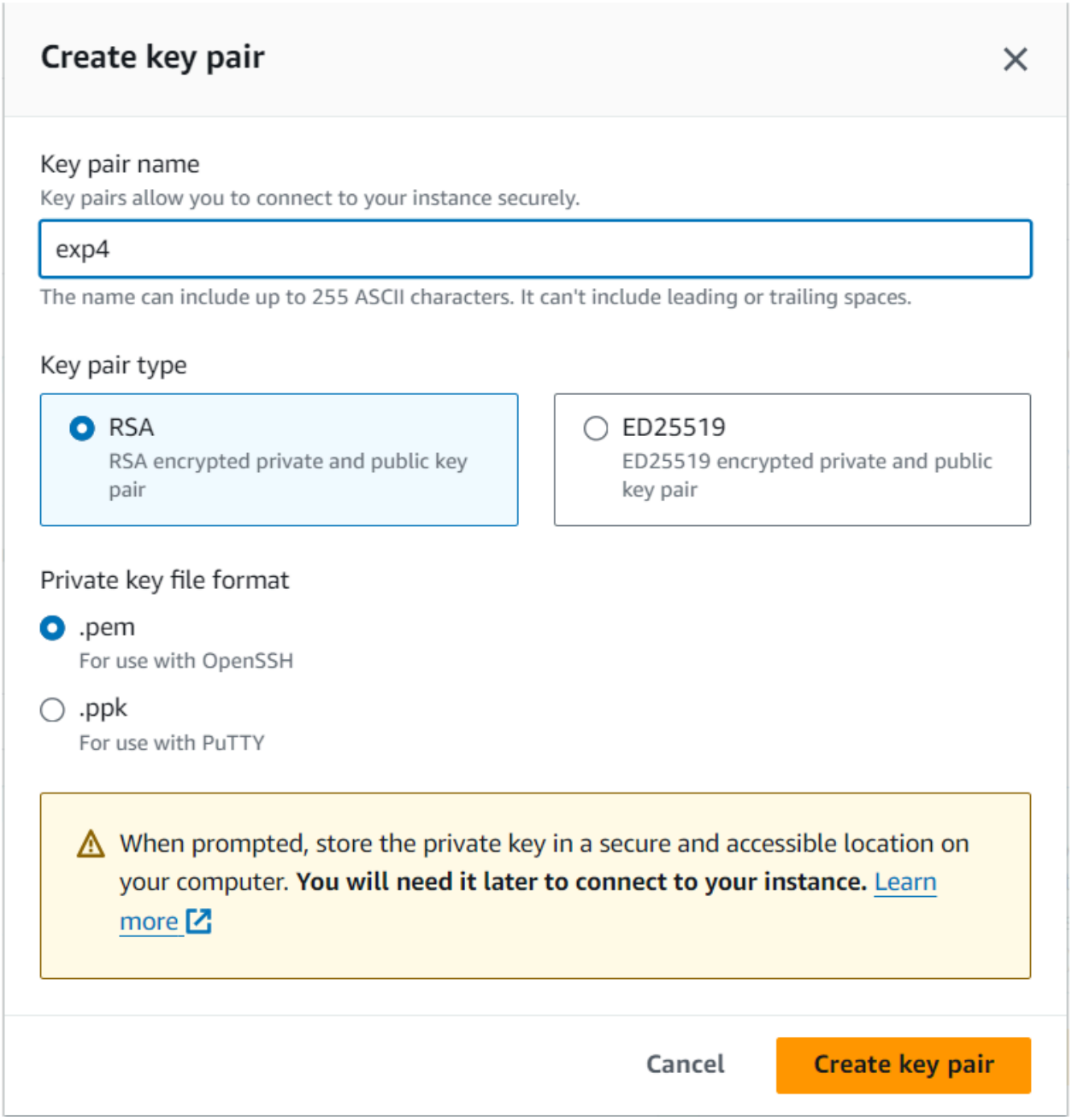
Aim: To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deploy Your First Kubernetes Application.

Theory

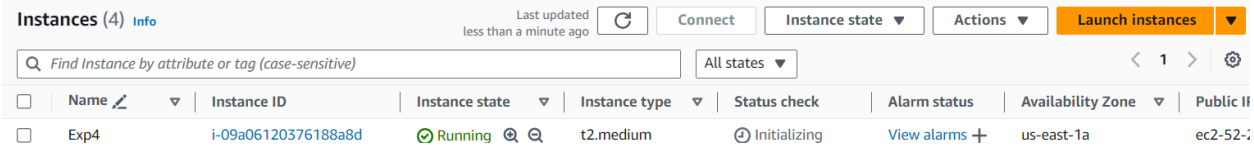
kubectl is the command-line tool used to interact with Kubernetes clusters. It serves as the primary interface for managing and orchestrating containers in a Kubernetes environment. By sending commands to the Kubernetes API server, kubectl allows you to control clusters, manage workloads, and inspect resource states.

To begin using Kubernetes, installing `kubectl` is essential. The installation process varies based on the operating system (Linux, Windows, or macOS). After installing, kubectl connects to the Kubernetes cluster using the kubeconfig file, which stores details like cluster name, server address, and access credentials. With this connection established, you can use kubectl to perform a variety of operations, such as creating, updating, scaling, and deleting applications.

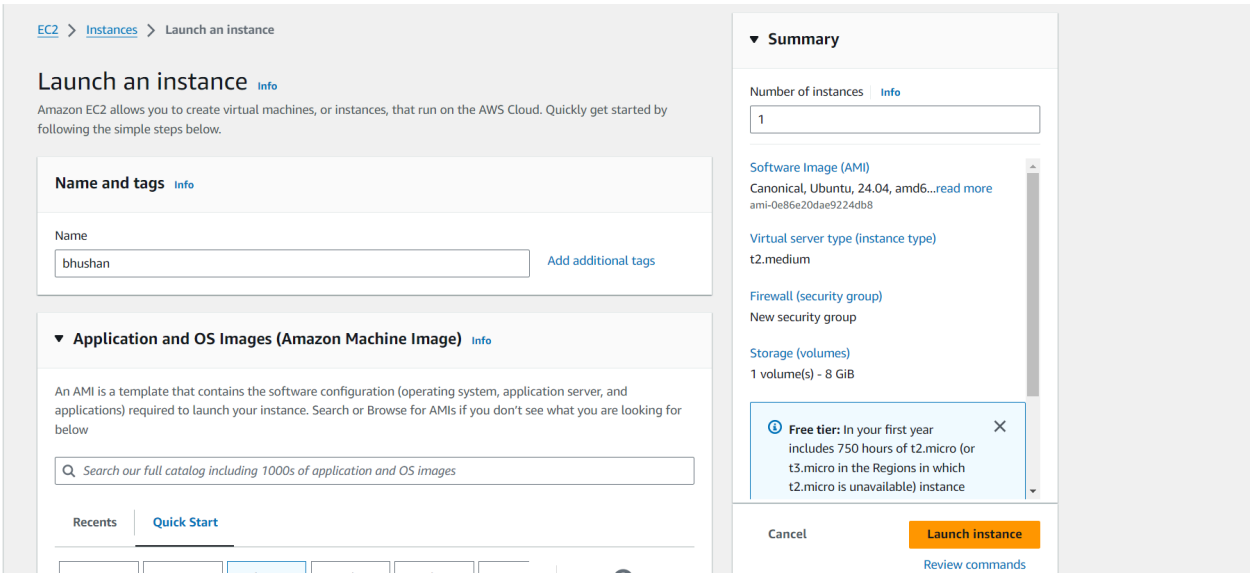
Step 1:Create an EC2 instance use ubuntu application and select t2 .medium category in instance type create a new key rsa type save it in local machine in an folder:

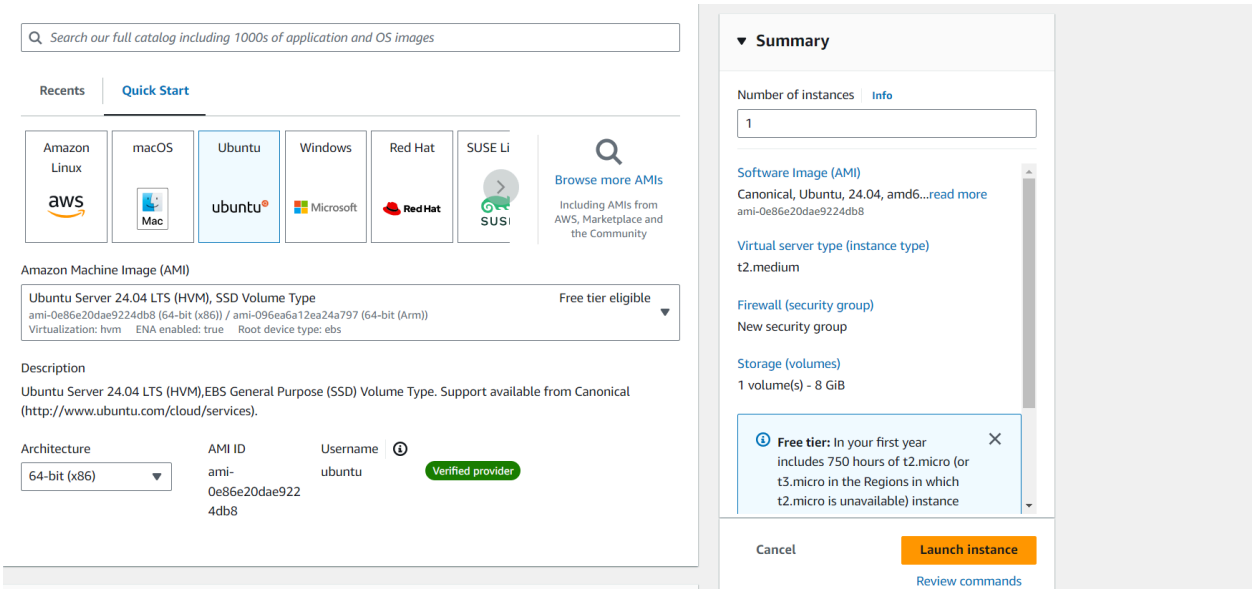


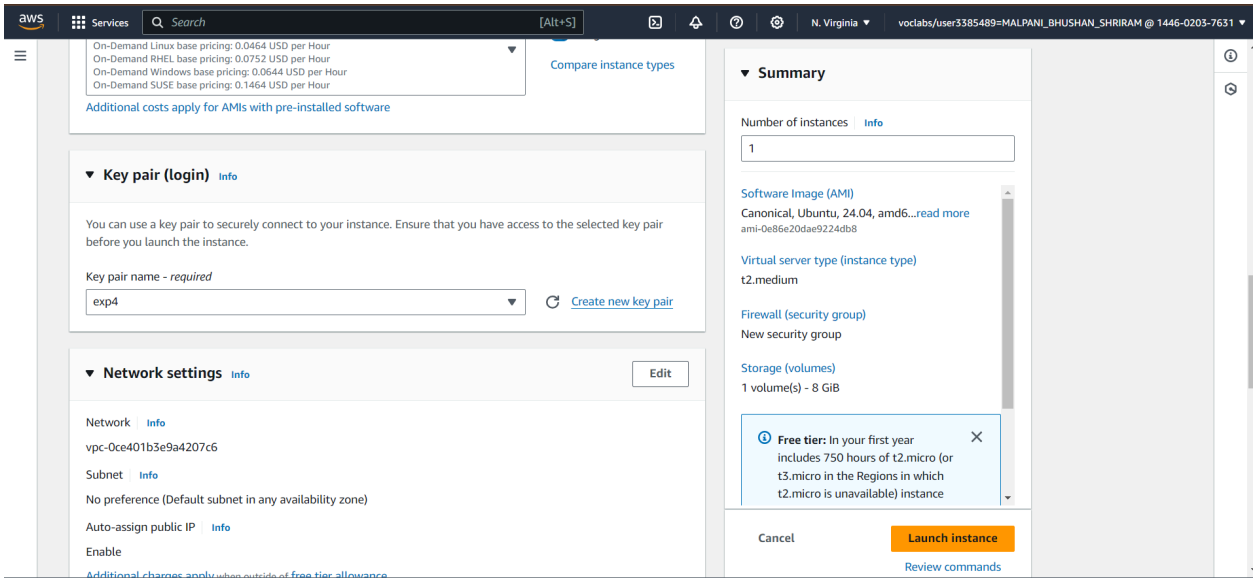
Step 2:Click create to create the instance:

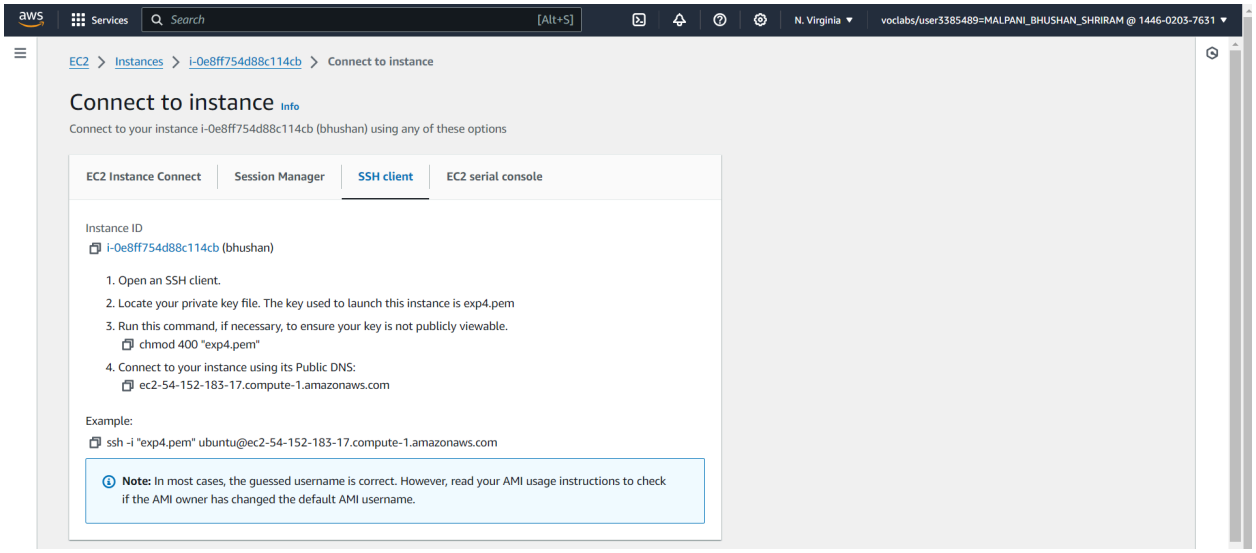


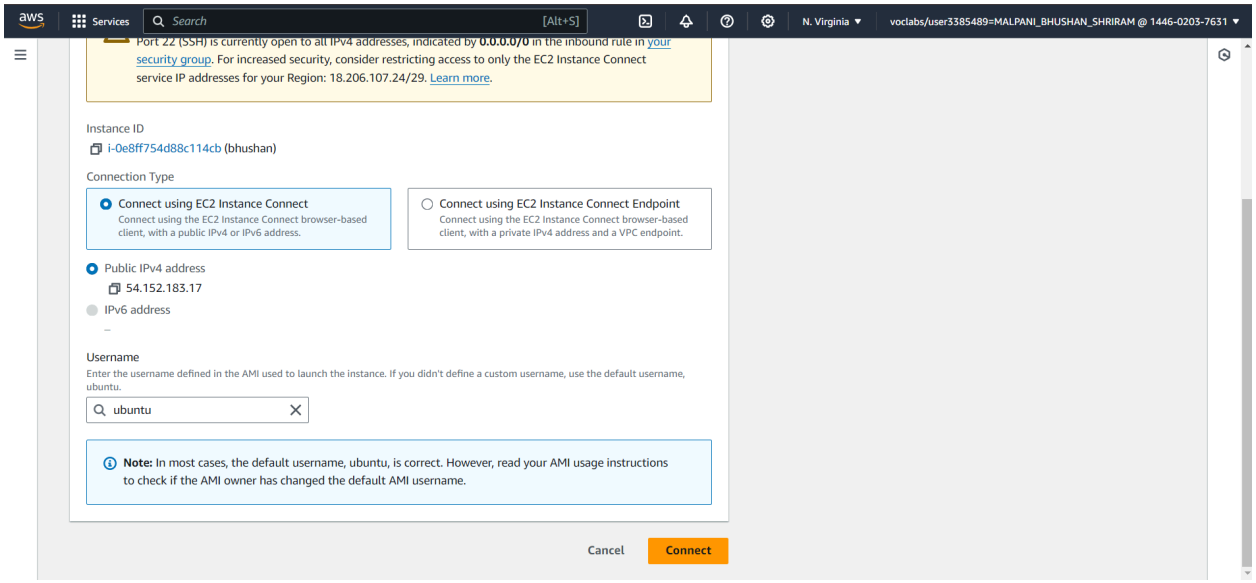
Step 3:
Navigate to ssh client copy the key:






aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3385489=MALPANI_BHUSHAN_SHRIRAM @ 1446-0203-7631
On-Demand Linux base pricing: 0.0464 USD per Hour
On-Demand RHEL base pricing: 0.0752 USD per Hour
On-Demand Windows base pricing: 0.0644 USD per Hour
On-Demand SUSE base pricing: 0.1464 USD per Hour
Compare instance types
Additional costs apply for AMIs with pre-installed software
Key pair (login) Info
You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.
Key pair name - required
exp4
Create new key pair
Network settings Info
Edit
Network Info
vpc-0ce401b3e9a4207c6
Subnet Info
No preference (Default subnet in any availability zone)
Auto-assign public IP Info
Enable
Summary
Number of instances Info
1
Software Image (AMI)
Canonical, Ubuntu, 24.04, amd6...read more
ami-0e86e20dae9224db8
Virtual server type (instance type)
t2.medium
Firewall (security group)
New security group
Storage (volumes)
1 volume(s) - 8 GiB
Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance
Cancel
Launch instance
Review commands


aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3385489=MALPANI_BHUSHAN_SHRIRAM @ 1446-0203-7631
EC2 > Instances > i-0e8ff754d88c114cb > Connect to instance
Connect to instance Info
Connect to your instance i-0e8ff754d88c114cb (bhushan) using any of these options
EC2 Instance Connect
Session Manager
SSH client
EC2 serial console
Instance ID
i-0e8ff754d88c114cb (bhushan)
1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is exp4.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
chmod 400 "exp4.pem"
4. Connect to your instance using its Public DNS:
ec2-54-152-183-17.compute-1.amazonaws.com
Example:
ssh -i "exp4.pem" ubuntu@ec2-54-152-183-17.compute-1.amazonaws.com
Note: In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.



Step 4:navigate to the folder open the terminal and paste the ssh command:
ssh -i "sahilexp4key.pem" ubuntu@ec2-52-201-236-39.compute-1.amazonaws.com

```
PS C:\Users\HP\Desktop\Exp4> ssh -i "exp4.pem" ubuntu@ec2-54-152-183-17.compute-1.amazonaws.com
The authenticity of host 'ec2-54-152-183-17.compute-1.amazonaws.com (64:ff9b::3698:b711)' can't be established.
ED25519 key fingerprint is SHA256:sK0m7P+Ism4YHKKxY7e6EXOS6KsyVWlghbvcGiH1v+s.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-54-152-183-17.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 Thu Sep 26 16:31:30 UTC 2024

  System load:  0.03              Processes:             122
  Usage of /:   22.9% of 6.71GB   Users logged in:       1
  Memory usage: 6%                IPv4 address for enX0: 172.31.85.133
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status
```

Step 5:Install docker
Use the commands given below to install docker
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"

```
ubuntu@ip-172-31-85-133:~$ sudo mkdir -p /etc/apt/trusted.gpg.d/
ubuntu@ip-172-31-85-133:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/trusted.gpg.d/doc
ker.asc
```

```
ubuntu@ip-172-31-85-133:~$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
$(lsb_release -cs) stable"
Repository: 'deb [arch=amd64] https://download.docker.com/linux/ubuntu noble stable'
Description:
Archive for codename: noble components: stable
More info: https://download.docker.com/linux/ubuntu
Adding repository.
Press [ENTER] to continue or Ctrl-c to cancel.
Adding deb entry to /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-noble.list
Adding disabled deb-src entry to /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-noble.list
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 https://download.docker.com/linux/ubuntu noble InRelease [48.8 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:7 https://download.docker.com/linux/ubuntu noble/stable amd64 Packages [15.3 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:9 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [378 kB]
Get:10 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [82.0 kB]
Get:11 http://security.ubuntu.com/ubuntu noble-security/main amd64 c-n-f Metadata [4548 B]
Get:12 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [271 kB]
Get:13 http://security.ubuntu.com/ubuntu noble-security/universe Translation-en [115 kB]
Get:14 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [8632 B]
Get:15 http://security.ubuntu.com/ubuntu noble-security/universe amd64 c-n-f Metadata [10.2 kB]
Get:16 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Packages [353 kB]
```

Use:
sudo apt-get update

```
ubuntu@ip-172-31-85-133:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... Done
```

Use:
sudo apt-get install -y docker-ce

```
ubuntu@ip-172-31-85-133:~$ sudo apt-get install -y docker-ce
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  containerd.io docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0
  pigz slirp4netns
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite
The following NEW packages will be installed:
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7
  libslirp0 pigz slirp4netns
0 upgraded, 10 newly installed, 0 to remove and 142 not upgraded.
Need to get 123 MB of archives.
After this operation, 442 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 pigz amd64 2.8-1 [65.6 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libltdl7 amd64 2.4.7-7build1 [40.3 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libslirp0 amd64 4.7.0-1ubuntu3 [63.8 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 slirp4netns amd64 1.2.1-1build2 [34.9 kB]
Get:5 https://download.docker.com/linux/ubuntu noble/stable amd64 containerd.io amd64 1.7.22-1 [29.5 MB]
Get:6 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-buildx-plugin amd64 0.17.1-1~ubuntu.24.04~noble
 [30.3 MB]
Get:7 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-ce-cli amd64 5:27.3.1-1~ubuntu.24.04~noble [15.
0 MB]
Get:8 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-ce amd64 5:27.3.1-1~ubuntu.24.04~noble [25.6 MB
```

Now the docker is installed;

Now lets enable the docker:

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

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

```
ubuntu@ip-172-31-85-133:~$ sudo mkdir -p /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
```

sudo systemctl enable docker

```
ubuntu@ip-172-31-85-133:~$ sudo systemctl enable docker
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
```

sudo systemctl daemon-reload

```
ubuntu@ip-172-31-85-133:~$ sudo systemctl daemon-reload
sudo systemctl restart docker
```

sudo systemctl restart docker

```
ubuntu@ip-172-31-85-133:~$ sudo mkdir -p /etc/apt/keyrings
ubuntu@ip-172-31-85-133:~$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /e
tc/apt/keyrings/kubernetes-apt-keyring.gpg
```

Step 6:Now lets install kubernetes;
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

```
ubuntu@ip-172-31-85-133:~$ 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
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
```

sudo apt-get update

```
ubuntu@ip-172-31-85-133:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:3 https://download.docker.com/linux/ubuntu noble InRelease
Hit:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  InRelease [1186 B]
Get:7 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  Packages [4865 B]
Fetched 6051 B in 0s (12.3 kB/s)
Reading package lists... Done
```

sudo apt-get install -y kubelet kubeadm kubectl

```
ubuntu@ip-172-31-85-133:~$ sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  conntrack cri-tools kubernetes-cni
The following NEW packages will be installed:
  conntrack cri-tools kubeadm kubectl kubelet kubernetes-cni
0 upgraded, 6 newly installed, 0 to remove and 142 not upgraded.
Need to get 87.4 MB of archives.
After this operation, 314 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 conntrack amd64 1:1.4.8-1ubuntu1 [37.9 kB]
Get:2 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  cri-tools 1.31.1-1.1 [15.7
 MB]
Get:3 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  kubeadm 1.31.1-1.1 [11.4 M
B]
Get:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  kubectl 1.31.1-1.1 [11.2 M
```

sudo apt-mark hold kubelet kubeadm kubectl

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

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
```

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

```
ubuntu@ip-172-31-85-133:~$ sudo systemctl enable --now kubelet
sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
W0926 16:54:00.720484    4952 checks.go:1080] [preflight] WARNING: Couldn't create the interface used for talking to the
 container runtime: failed to create new CRI runtime service: validate service connection: validate CRI v1 runtime API f
or endpoint "unix:///var/run/containerd/containerd.sock": rpc error: code = Unimplemented desc = unknown service runtime
.v1.RuntimeService
        [WARNING FileExisting-socat]: socat not found in system path
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action beforehand using 'kubeadm config images pull'
error execution phase preflight: [preflight] Some fatal errors occurred:
failed to create new CRI runtime service: validate service connection: validate CRI v1 runtime API for endpoint "unix://
/var/run/containerd/containerd.sock": rpc error: code = Unimplemented desc = unknown service runtime.v1.RuntimeService[p
reflight] If you know what you are doing, you can make a check non-fatal with `--ignore-preflight-errors=...`
To see the stack trace of this error execute with --v=5 or higher
ubuntu@ip-172-31-85-133:~$
```

sudo apt-get install -y containerd

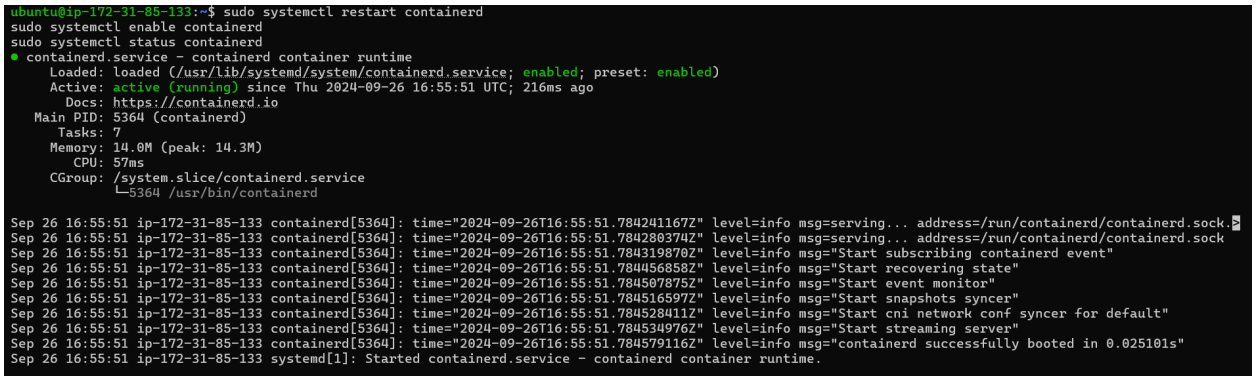
```
ubuntu@ip-172-31-85-133:~$ sudo apt-get install -y containerd
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0 pigz
  slirp4netns
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  runc
The following packages will be REMOVED:
  containerd.io docker-ce
The following NEW packages will be installed:
  containerd runc
0 upgraded, 2 newly installed, 2 to remove and 142 not upgraded.
Need to get 47.2 MB of archives.
After this operation, 53.1 MB disk space will be freed.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 runc amd64 1.1.12-0ubuntu3.1 [8599 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 containerd amd64 1.7.12-0ubuntu4.1 [38.6 MB]
Fetched 47.2 MB in 1s (72.0 MB/s)
(Reading database ... 68064 files and directories currently installed.)
Removing docker-ce (5:27.3.1-1~ubuntu.24.04~noble) ...
Removing containerd.io (1.7.22-1) ...
Selecting previously unselected package runc.
(Reading database ... 68044 files and directories currently installed.)
Preparing to unpack .../runc_1.1.12-0ubuntu3.1_amd64.deb ...
Unpacking runc (1.1.12-0ubuntu3.1) ...
Selecting previously unselected package containerd.
Preparing to unpack .../containerd_1.7.12-0ubuntu4.1_amd64.deb ...
Unpacking containerd (1.7.12-0ubuntu4.1) ...
Setting up runc (1.1.12-0ubuntu3.1) ...
Setting up containerd (1.7.12-0ubuntu4.1) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.
```

```
  [plugins."io.containerd.transfer.v1.local"]
    config_path = ""
    max_concurrent_downloads = 3
    max_concurrent_uploaded_layers = 3

    [[plugins."io.containerd.transfer.v1.local".unpack_config]]
      differ = ""
      platform = "linux/amd64"
      snapshotter = "overlayfs"

[proxy_plugins]

[stream_processors]

  [stream_processors."io.containerd.ocicrypt.decoder.v1.tar"]
    accepts = ["application/vnd.oci.image.layer.v1.tar+encrypted"]
    args = ["--decryption-keys-path", "/etc/containerd/ocicrypt/keys"]
    env = ["OCICRYPT_KEYPROVIDER_CONFIG=/etc/containerd/ocicrypt/ocicrypt_keyprovider.conf"]
    path = "ctd-decoder"
    returns = "application/vnd.oci.image.layer.v1.tar"

  [stream_processors."io.containerd.ocicrypt.decoder.v1.tar.gzip"]
    accepts = ["application/vnd.oci.image.layer.v1.tar+gzip+encrypted"]
    args = ["--decryption-keys-path", "/etc/containerd/ocicrypt/keys"]
    env = ["OCICRYPT_KEYPROVIDER_CONFIG=/etc/containerd/ocicrypt/ocicrypt_keyprovider.conf"]
    path = "ctd-decoder"
    returns = "application/vnd.oci.image.layer.v1.tar+gzip"

[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-172-31-85-133:~$
```

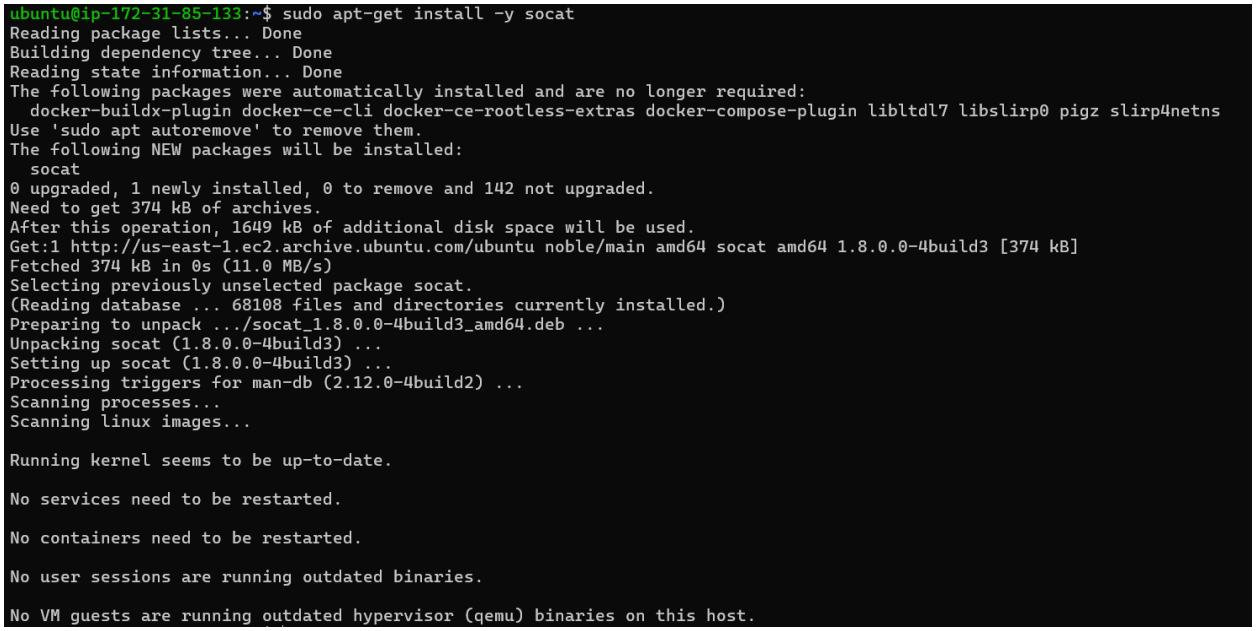
sudo mkdir -p /etc/containerd
sudo containerd config default | sudo tee /etc/containerd/config.toml

```
ubuntu@ip-172-31-85-133:~$ sudo mkdir -p /etc/containerd
sudo containerd config default | sudo tee /etc/containerd/config.toml
disabled_plugins = []
imports = []
oom_score = 0
plugin_dir = ""
required_plugins = []
root = "/var/lib/containerd"
state = "/run/containerd"
temp = ""
version = 2

[cgroup]
  path = ""

[debug]
  address = ""
  format = ""
  gid = 0
  level = ""
  uid = 0

[grpc]
  address = "/run/containerd/containerd.sock"
  gid = 0
  max_recv_message_size = 16777216
  max_send_message_size = 16777216
  tcp_address = ""
  tcp_tls_ca = ""
  tcp_tls_cert = ""
```

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



sudo apt-get install -y socat



Step 7: Intitialize the kubernete:
sudo kubeadm init --pod-network-cidr=10.244.0.0/16

```
ubuntu@ip-172-31-85-133:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action beforehand using 'kubeadm config images pull'
W0926 16:57:08.588594    5580 checks.go:846] detected that the sandbox image "registry.k8s.io/pause:3.8" of the container runtime is inconsistent with that
used by kubeadm.It is recommended to use "registry.k8s.io/pause:3.10" as the CRI sandbox image.
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [ip-172-31-85-133 kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster
.local] and IPs [10.96.0.1 172.31.85.133]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [ip-172-31-85-133 localhost] and IPs [172.31.85.133 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [ip-172-31-85-133 localhost] and IPs [172.31.85.133 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "super-admin.conf" kubeconfig file
[kubeconfig] Writing "kubelet.conf" kubeconfig file
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
[kubeconfig] Writing "scheduler.conf" kubeconfig file
[etcd] Creating static Pod manifest for local etcd in "/etc/kubernetes/manifests"
[control-plane] Using manifest folder "/etc/kubernetes/manifests"
[control-plane] Creating static Pod manifest for "kube-apiserver"
[control-plane] Creating static Pod manifest for "kube-controller-manager"
[control-plane] Creating static Pod manifest for "kube-scheduler"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Starting the kubelet
[wait-control-plane] Waiting for the kubelet to boot up the control plane as static Pods from directory "/etc/kubernetes/manifests"
```

```
ubuntu@ip-172-31-85-133:~$ mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
ubuntu@ip-172-31-85-133:~$
```

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

```
ubuntu@ip-172-31-85-133:~$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
namespace/kube-flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
```

Step 8:Now we can deploy our nginx server on this cluster using following steps:
kubectl apply -f https://k8s.io/examples/application/deployment.yaml

```
ubuntu@ip-172-31-85-133:~$ kubectl apply -f https://k8s.io/examples/application/deployment.yaml
deployment.apps/nginx-deployment created
```

kubectl get pods

```
ubuntu@ip-172-31-85-133:~$ kubectl get pods
NAME                               READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-bq75m   0/1     Pending   0          16s
nginx-deployment-d556bf558-t7mgm   0/1     Pending   0          16s
```

POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
kubectl port-forward $POD_NAME 8080:80

```
ubuntu@ip-172-31-85-133:~$ kubectl port-forward $POD_NAME 8080:80
error: unable to forward port because pod is not running. Current status=Pending
ubuntu@ip-172-31-85-133:~$
```

kubectl taint nodes --all node-role.kubernetes.io/control-plane-node/ip-172-31-20-171 untainted

kubectl get nodes

```
ubuntu@ip-172-31-85-133:~$ kubectl taint nodes ip-172-31-85-133 node-role.kubernetes.io/control-plane:NoSchedule-
node/ip-172-31-85-133 untainted
ubuntu@ip-172-31-85-133:~$ kubectl get nodes
kubectl get pods
NAME              STATUS   ROLES           AGE     VERSION
ip-172-31-85-133   Ready    control-plane   6m53s   v1.31.1
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-bq75m   1/1     Running   0          4m26s
nginx-deployment-d556bf558-t7mgm   1/1     Running   0          4m26s
```

kubectl get pods

```
ubuntu@ip-172-31-85-133:~$ kubectl port-forward $POD_NAME 8081:80
Forwarding from 127.0.0.1:8081 -> 80
Forwarding from [::1]:8081 -> 80
Handling connection for 8081
```

POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
kubectl port-forward $POD_NAME 8080:80

```
^Cubuntu@ip-172-31-85-133:~kubectl port-forward pod/nginx-deployment-d556bf558-t7mgm 8081:8080
Forwarding from 127.0.0.1:8081 -> 80
Forwarding from [::1]:8081 -> 80
Handling connection for 8081
```

Step 9 :check deployment:
Open new terminal in folder,
Paste the ssh key,
Type
curl --head http://127.0.0.8081

```
ubuntu@ip-172-31-85-133:~$ curl --head http://127.0.0.1:8081
HTTP/1.1 200 OK
Server: nginx/1.14.2
Date: Thu, 26 Sep 2024 17:24:21 GMT
Content-Type: text/html
Content-Length: 612
Last-Modified: Tue, 04 Dec 2018 14:44:49 GMT
Connection: keep-alive
ETag: "5c0692e1-264"
Accept-Ranges: bytes
```

Status Code 200 tells us that we have successfully deployed our nginx server on ec2 instance

Now we have successfully deployed our nginx server on our ec2 instance.

Conclusion

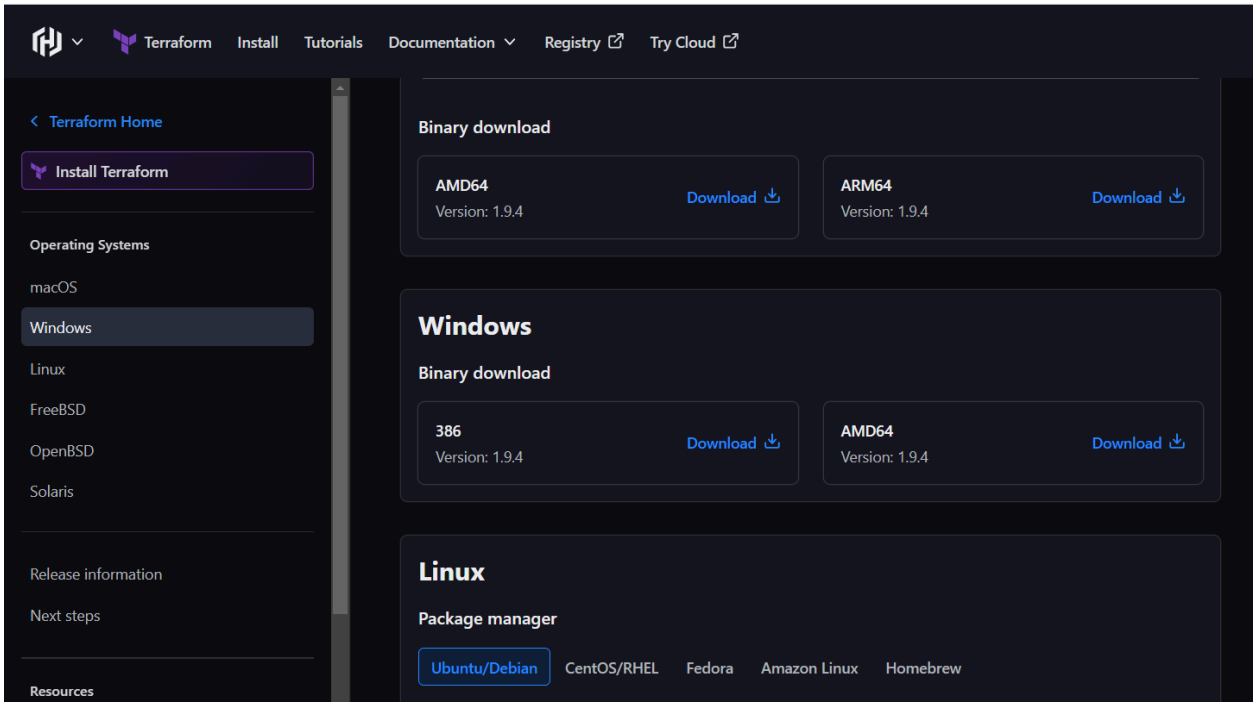
In this experiment, we installed kubectl, the command-line tool for managing Kubernetes clusters, using the appropriate package manager for our system. After setting up the kubeconfig file to connect with the Kubernetes cluster, we verified the connection and the cluster status using kubectl get nodes. We then deployed our first Kubernetes application by writing the necessary configuration in files, ensuring the application was defined correctly. Following the deployment, we inspected the resources and application status with kubectl get pods to confirm the application was running successfully in the cluster. This experiment demonstrated the basic interaction with Kubernetes using kubectl for managing resources and monitoring deployments.

NAME: Bhushan Malpani **DIV: D15C** **Roll No:33**

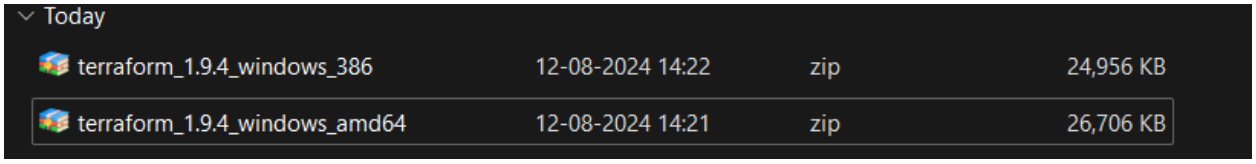
EXPERIMENT NO. 05

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

1)Install terraform in the system. You can choose the Operating System according to your requirement.

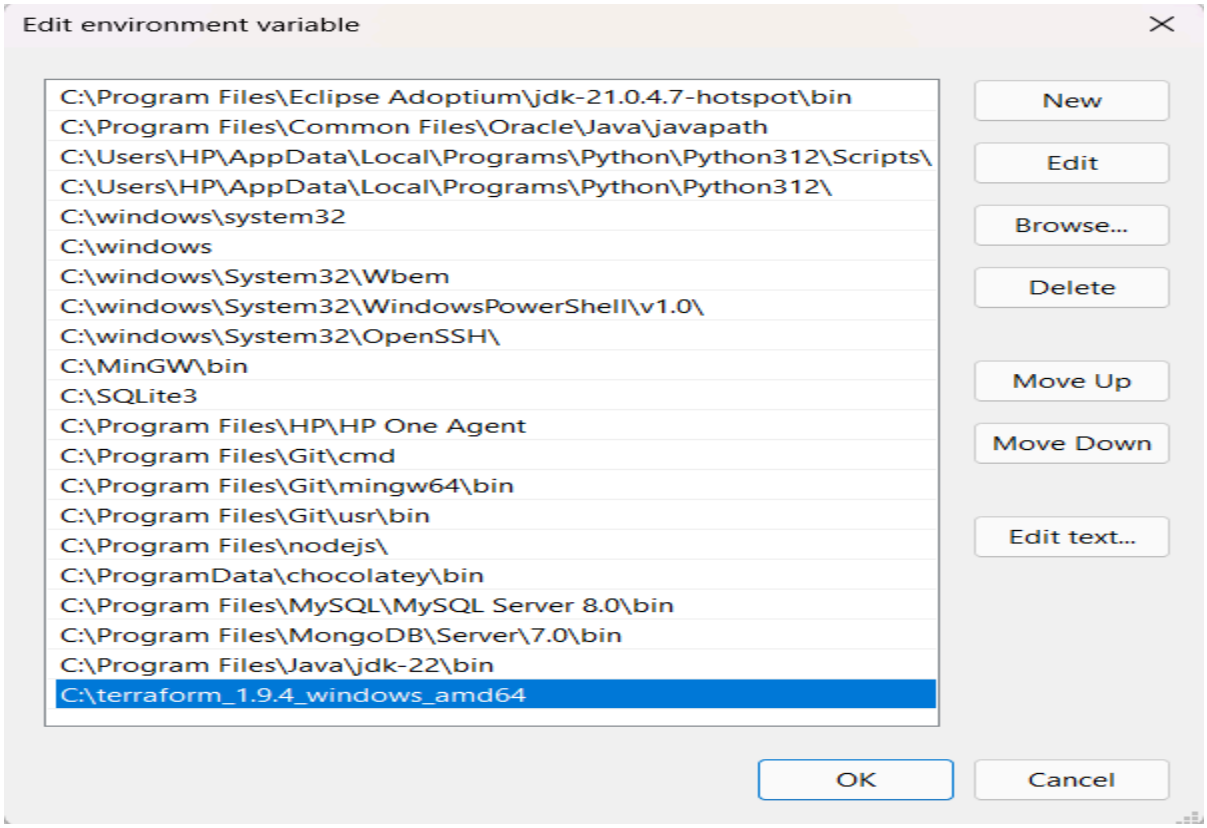


2) See the downloads file the zipped file will be there in the folder.

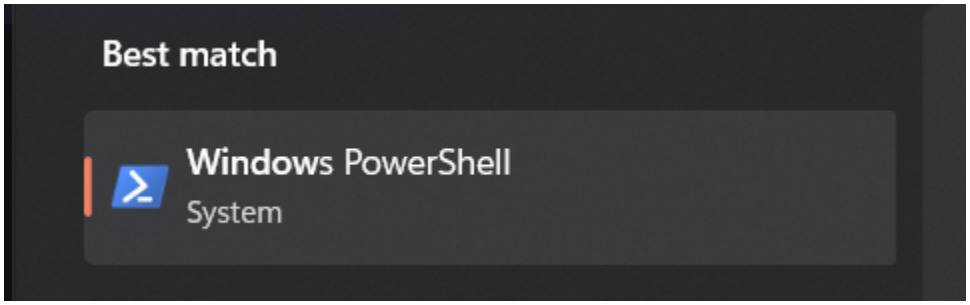


3)Extract this file in the C drive, copy the path of the file in the environment variable from settings

NAME: Bhushan Malpani **DIV: D15C** **Roll No:33**



4) Open the windows PowerShell and run the command terraform



NAME: Bhushan Malpani **DIV: D15C** **Roll No:33**

5)It will show all the properties of terraform:

```
PS C:\Users\HP> terraform
Usage: terraform [global options] <subcommand> [args]

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

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

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

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

NAME: Bhushan Malpani **DIV: D15C** **ROLL NO:33**

A. Creating docker image using terraform

Step 1: Check docker is working well using command docker.

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

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

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

Management Commands:
  builder     Manage builds
  buildx*     Docker Buildx
  checkpoint  Manage checkpoints
  compose*    Docker Compose
  container   Manage containers
  context     Manage contexts
  debug*      Get a shell into any image or container
  desktop*    Docker Desktop commands (Alpha)
  dev*        Docker Dev Environments
  extension*  Manages Docker extensions
  feedback*   Provide feedback, right in your terminal!
  image       Manage images
  init*       Creates Docker-related starter files for your project
  manifest    Manage Docker image manifests and manifest lists
  network     Manage networks
```

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

Step 2:**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

```
terraform{
    required_providers{
        docker = {
            source = "kreuzwerker/docker"
            version = "2.21.0"
        }
    }
}
provider "docker" {
    host = "npipe:////.//pipe//docker_engine"
}
# Pulls the image
resource "docker_image" "ubuntu"{
    name = "ubuntu:latest"
}
# Create a container
resource "docker_container" "foo"{
    image =docker_image.ubuntu.image_id
    name ="foo"
}
```

Step 3: Execute Terraform Init command to initialize the resources

```
PS C:\Users\HP\Desktop\Terraform\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.
```

NAME: Bhushan Malpani **DIV: D15C** **ROLL NO:33**

Step 4)Execute Terraform plan to see the available resources

```
PS C:\Users\HP\Desktop\Terraform\Docker> terraform plan

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

Terraform will perform the following actions:

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

      + labels (known after apply)
    }

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

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

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

This is docker images before apply:

```
PS C:\Users\HP\Desktop\Terraform\Docker> docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
```

Step5) 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\HP\Desktop\Terraform\Docker> terraform apply

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

Terraform will perform the following actions:

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

```
      + healthcheck (known after apply)

      + labels (known after apply)
    }

Plan: 1 to add, 0 to change, 0 to destroy.

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

  Enter a value: yes

docker_container.foo: Creating...
docker_container.foo: Creation complete after 0s [id=db3fd3cf50f3c1565b4b8d26c7d876a6cdfc8228abf9af04039673e59cb0b0f7]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

Docker images, After Executing Apply step:

```
PS C:\Users\HP\Desktop\Terraform\Docker> docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
ubuntu       latest    edbfe74c41f8   2 weeks ago   78.1MB
PS C:\Users\HP\Desktop\Terraform\Docker>
```

NAME: Bhushan Malpani **DIV: D15C** **ROLL NO:33**

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

```
PS C:\Users\HP\Desktop\Terraform\Docker> terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Refreshing state... [id=db3fd3cf50f3c1565b4b8d26c7d876a6cdfc8228abf9af04039673e59cb0b0f7]

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

Terraform will perform the following actions:

  # docker_container.foo will be destroyed
  - resource "docker_container" "foo" {
      - attach            = false -> null
      - command           = [
          - "sleep",
          - "infinity",
        ] -> null
      - cpu_shares        = 0 -> null
      - dns               = [] -> null
      - dns_opts          = [] -> null
      - dns_search        = [] -> null
      - entrypoint        = [] -> null
      - env               = [] -> null
      - gateway           = "172.17.0.1" -> null
      - group_add         = [] -> null
      - hostname          = "db3fd3cf50f3" -> null
      - id                = "db3fd3cf50f3c1565b4b8d26c7d876a6cdfc8228abf9af04039673e59cb0b0f7" -> null
      - image             = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a" -> null
      - init              = false -> null
      - ip_address        = "172.17.0.2" -> null
      - ip_prefix_length  = 16 -> null
      - ipc_mode          = "private" -> null
      - links             = [] -> null
      - log_driver        = "json-file" -> null
      - log_opts          = {} -> null
      - logs              = false -> null
      - max_retry_count   = 0 -> null
      - memory            = 0 -> null
      - memory_swap       = 0 -> null
      - must_run          = true -> null
      - name              = "foo" -> null
      - network_data      = [
```

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

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

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

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

  Enter a value: yes

docker_container.foo: Destroying... [id=db3fd3cf50f3c1565b4b8d26c7d876a6cdfc8228abf9af04039673e59cb0b0f7]
docker_container.foo: Destruction complete after 0s
docker_image.ubuntu: Destroying... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_image.ubuntu: Destruction complete after 0s

Destroy complete! Resources: 2 destroyed.
PS C:\Users\HP\Desktop\Terraform\Docker>
```

This is docker images after destroying:

```
PS C:\Users\HP\Desktop\Terraform\Docker> docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
PS C:\Users\HP\Desktop\Terraform\Docker>
```

NAME: BHUSHAN MALPANI DIV: D15C ROLL NO: 33

Adv DevOps Practical 7

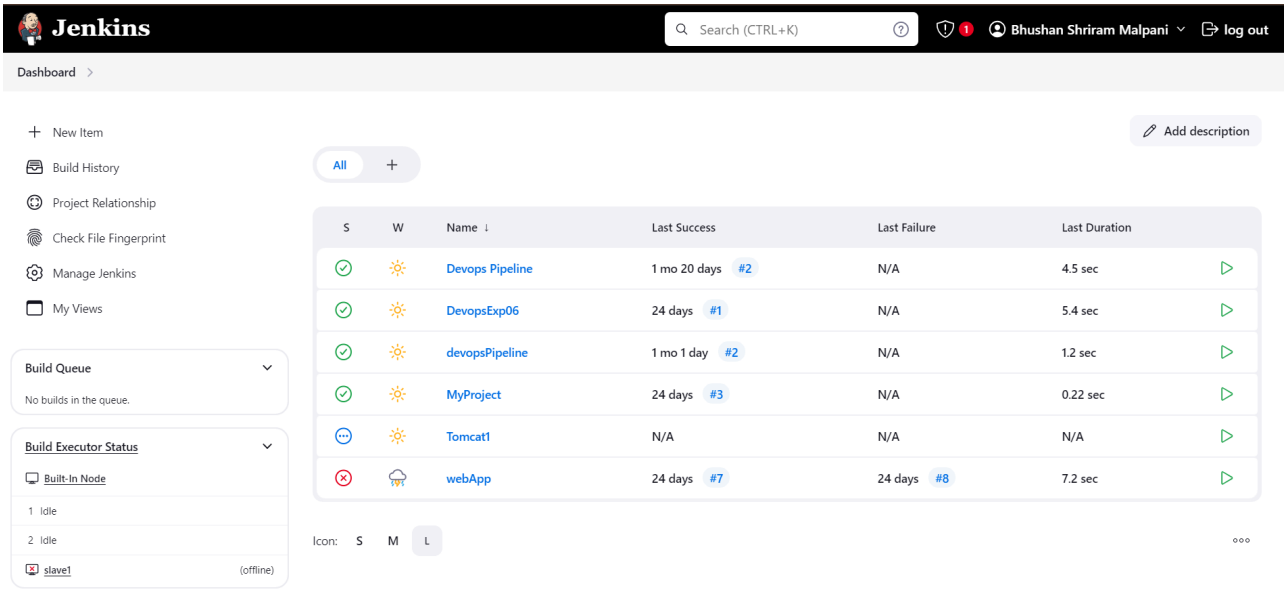
Aim: To understand Static Analysis SAST process and learn to integrate Jenkins SAST to SonarQube/GitLab.

Integrating Jenkins with SonarQube:

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

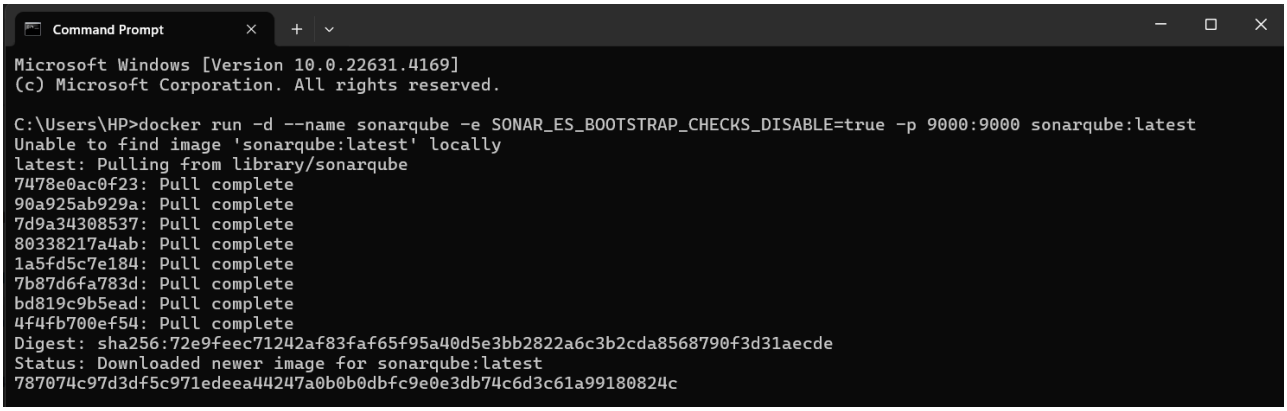
Steps to integrate Jenkins with SonarQube

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

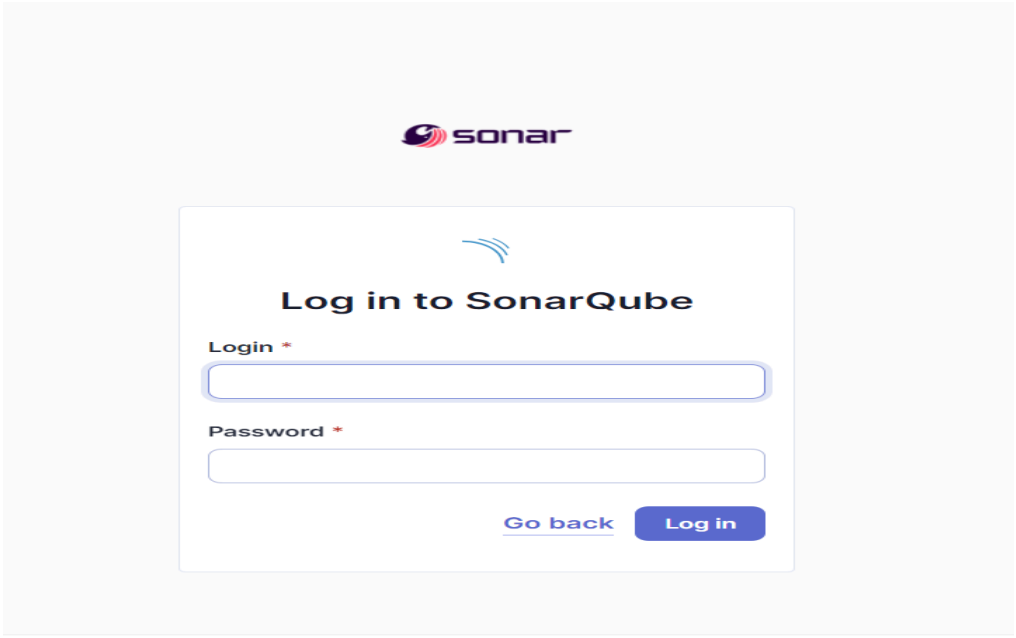


2. Run SonarQube in a Docker container using this command -
docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest

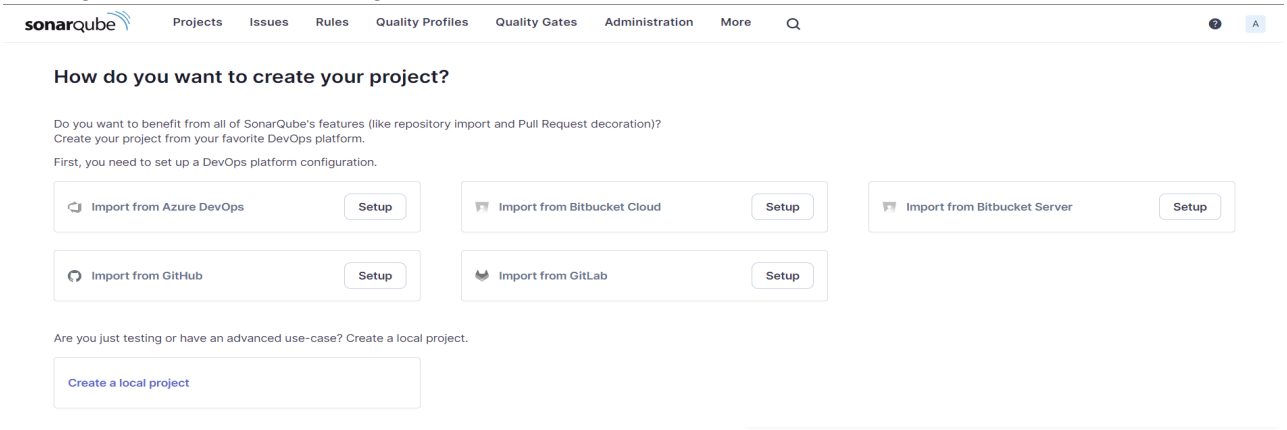
- **Warning: run below command only once**



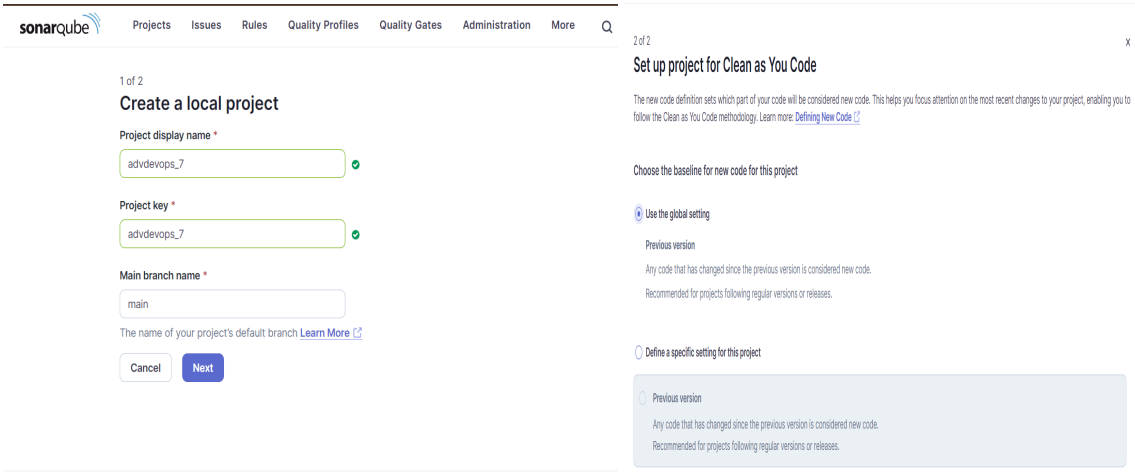
3. Once the container is up and running, you can check the status of SonarQube at localhost port 9000.



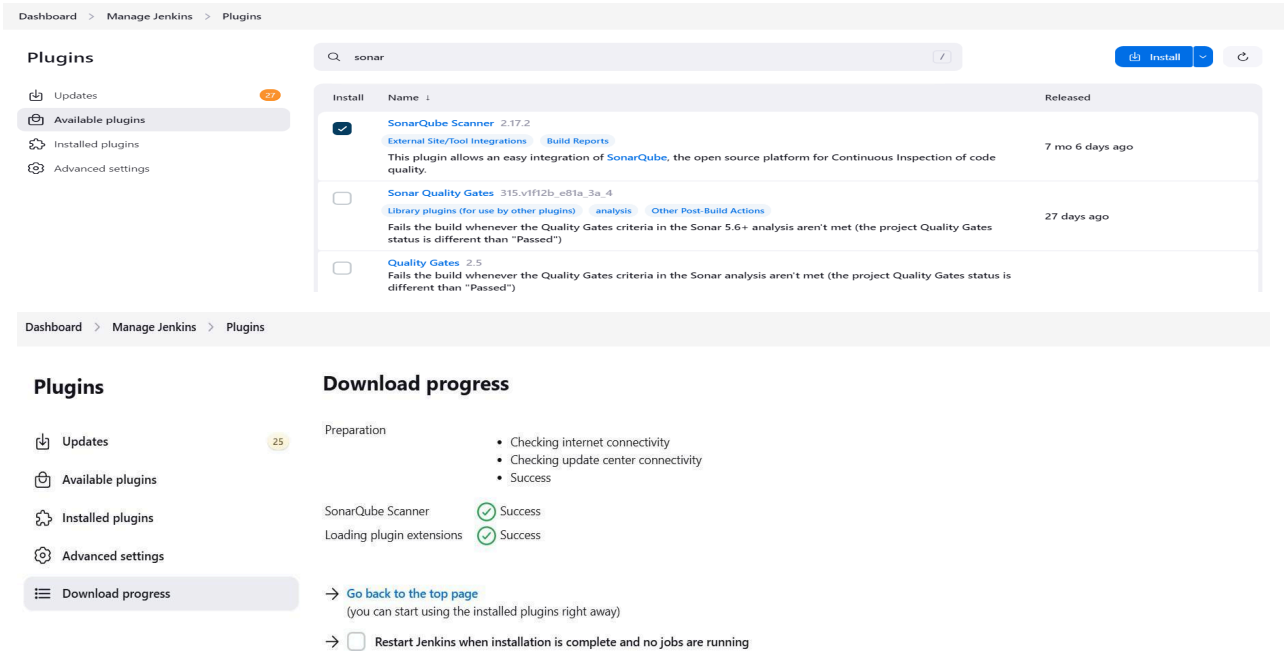
4. Login to SonarQube using username admin and password admin.



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



Setup the project and come back to Jenkins Dashboard.
Go to Manage Jenkins and search for SonarQube Scanner for Jenkins and install it.

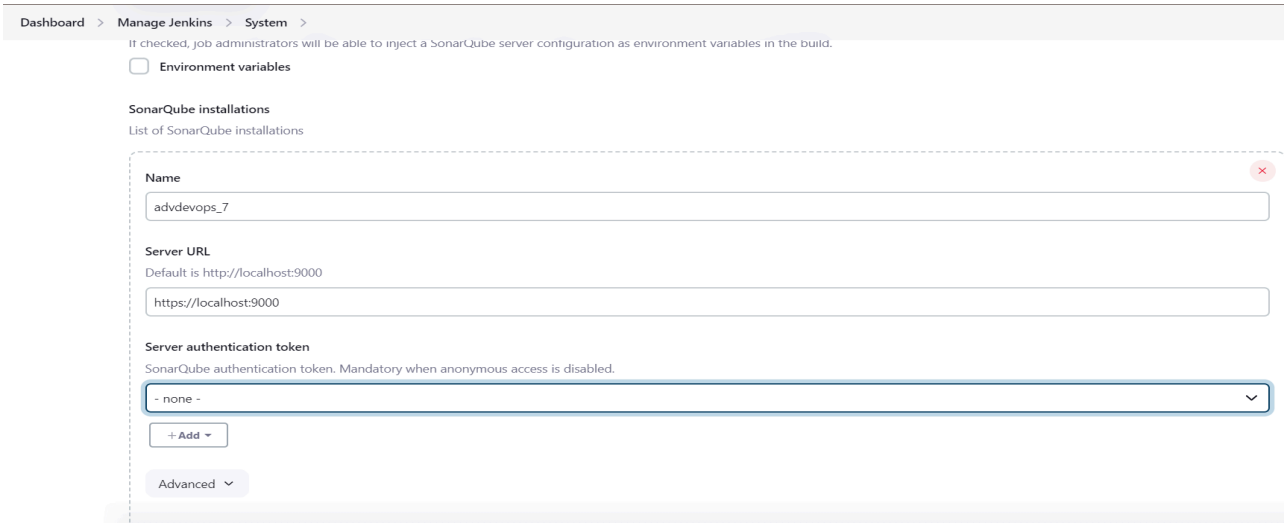


6. Under Jenkins 'Manage Jenkins' then go to 'system', scroll and look for **SonarQube Servers** and enter the details.
Enter the Server Authentication token if needed.

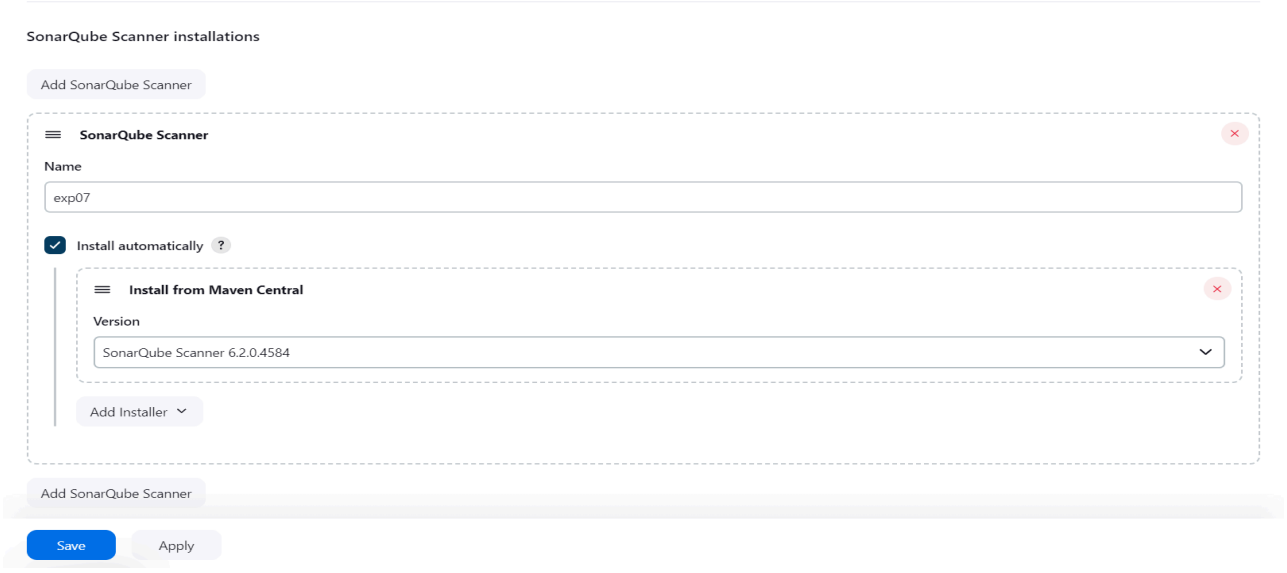
In SonarQube installations: Under **Name** add <project name of sonarqube> for me **adv_devops_7_sonarqube**
In **Server URL** Default is **http://localhost:9000**

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

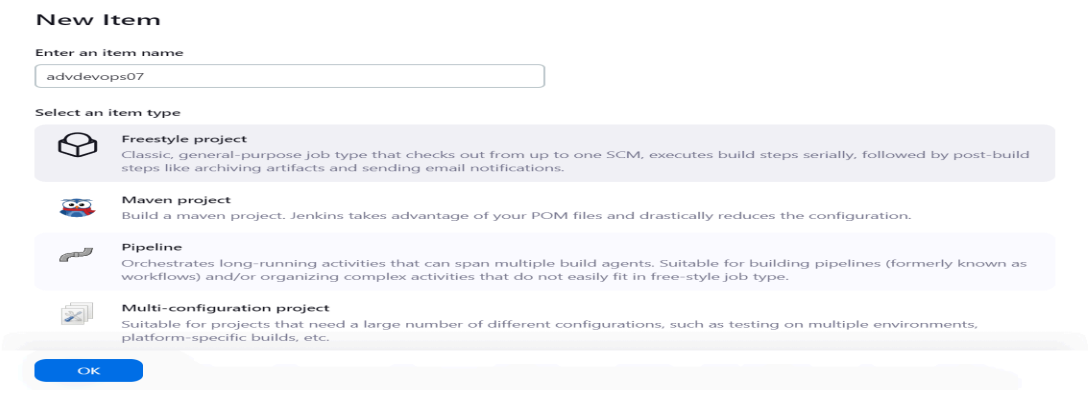
Dashboard > Manage Jenkins > Tools



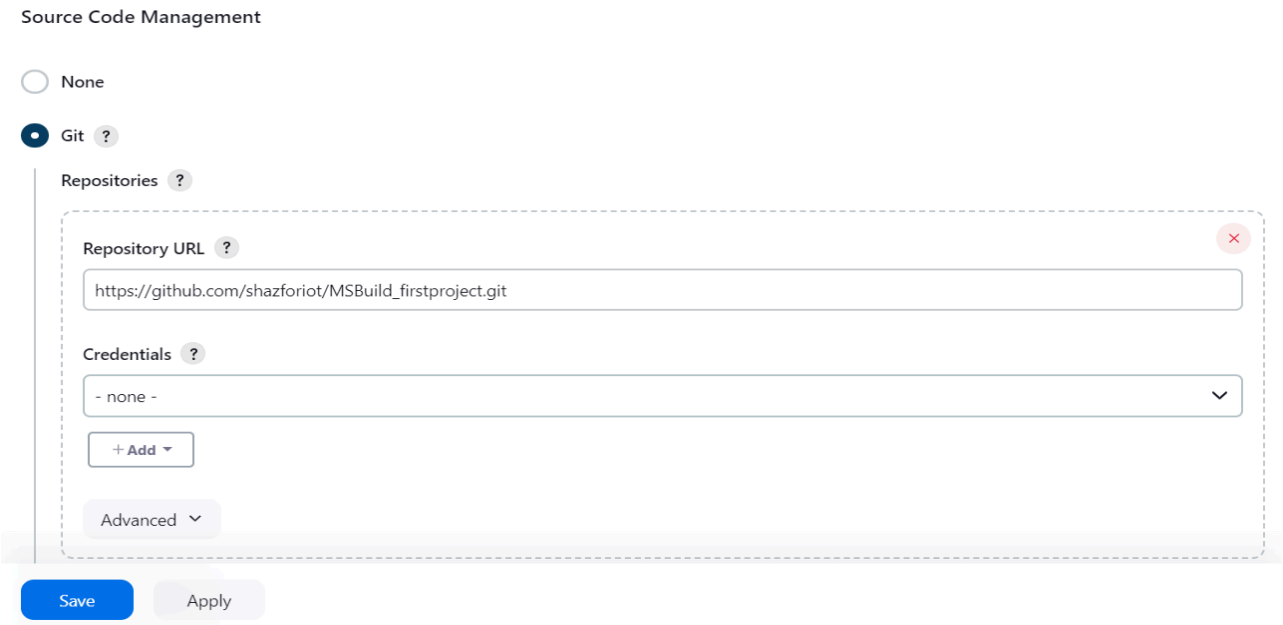
Check the “Install automatically” option. → Under name any name as identifier → Check the “Install automatically” option.



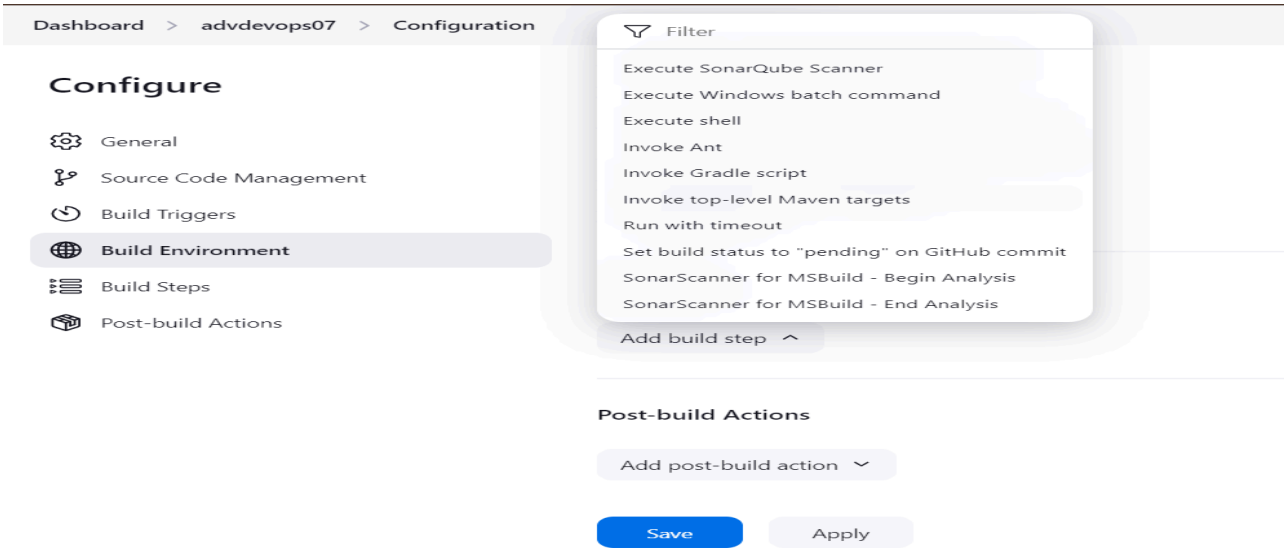
8. After the configuration, create a New Item in Jenkins, choose a freestyle project.

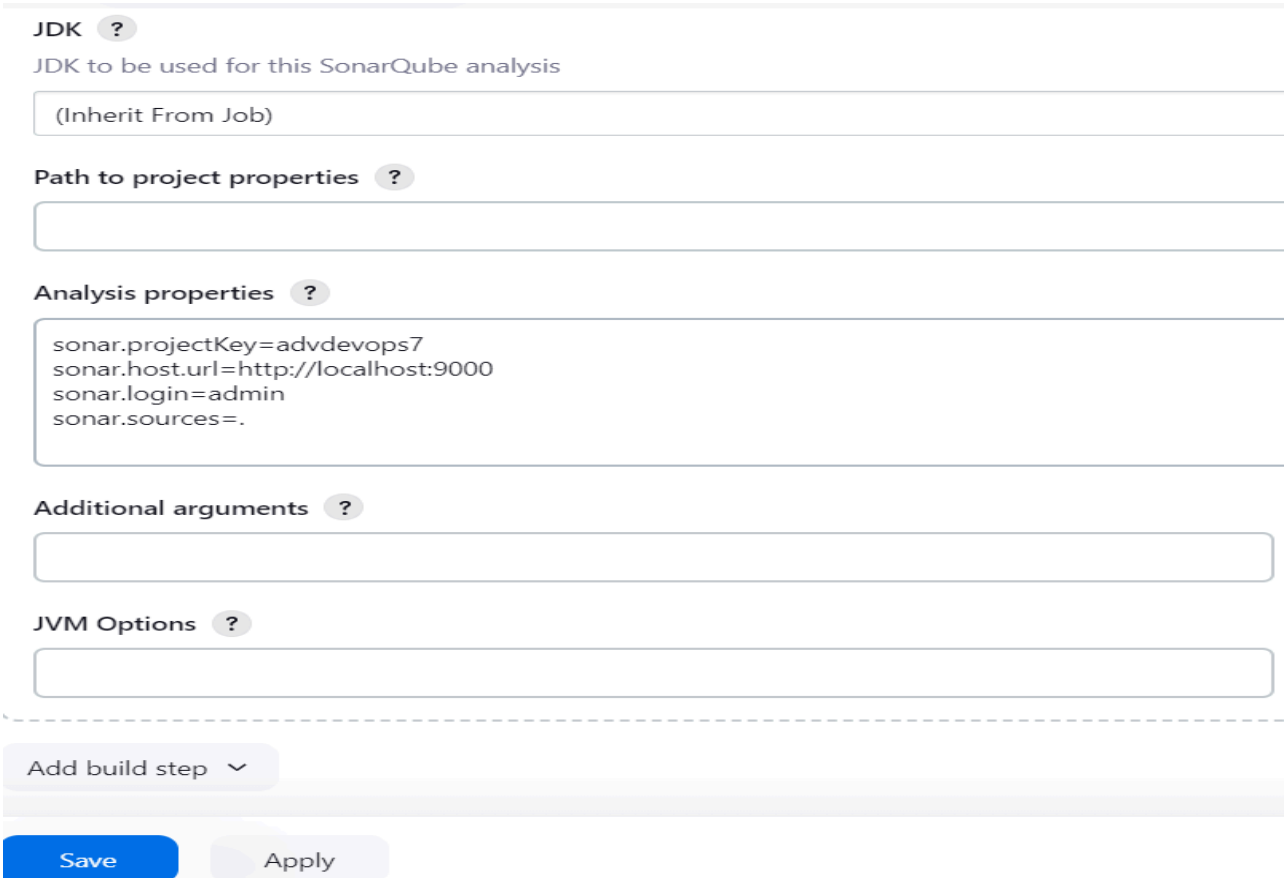


9. Choose this GitHub repository in Source Code Management.
https://github.com/shazforiot/MSBuild_firstproject.git
It is a sample hello-world project with no vulnerabilities and issues, just to test the integration.



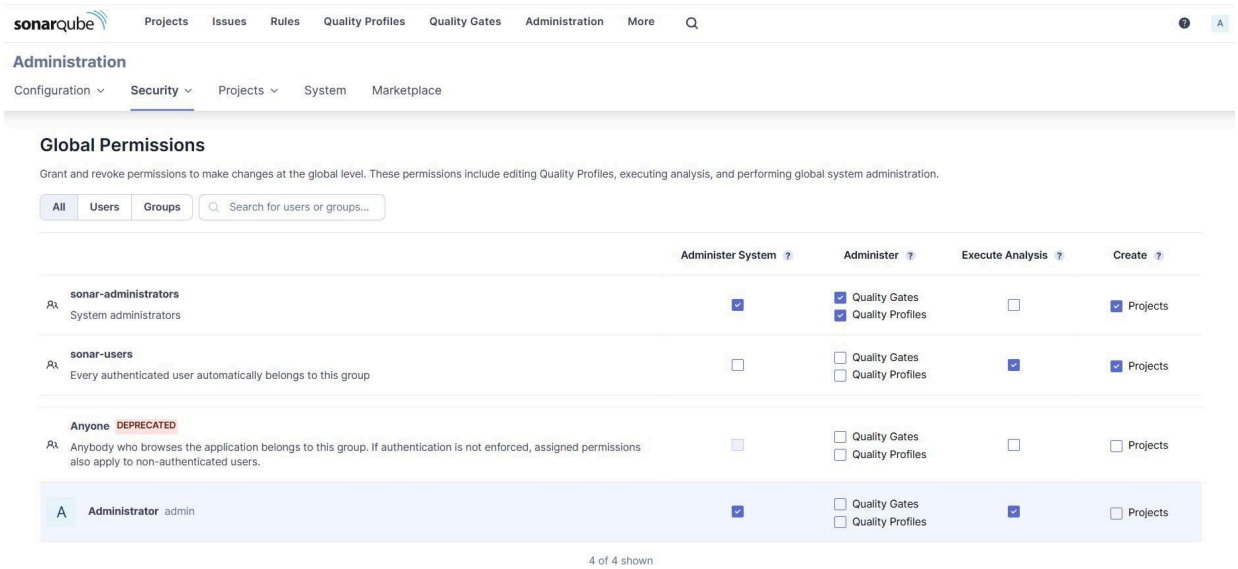
10. Under **Select project → Configuration → Build steps → Execute SonarQube Scanner**, enter these Analysis properties. Mention the SonarQube Project Key, Login, Password, Source path and Host URL.





Then save

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



IF CONSOLE OUTPUT FAILED:

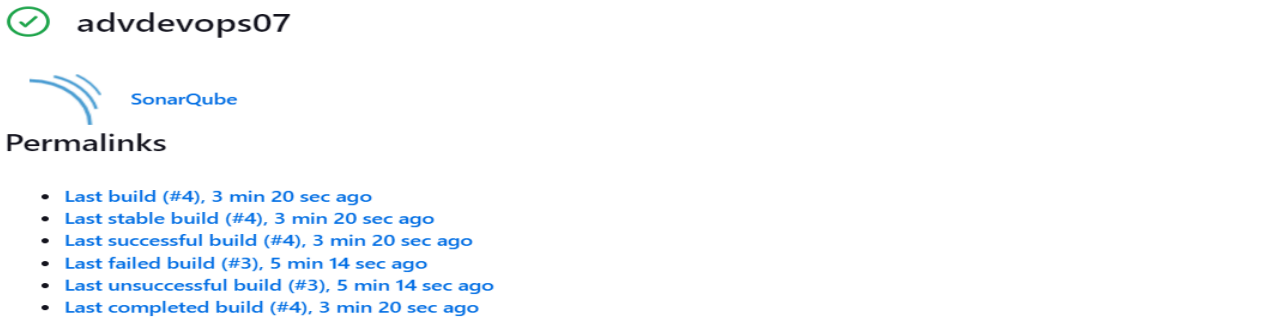
Step 1: Generate a New Authentication Token in SonarQube

1. **Login to SonarQube:**
   - Open your browser and go to `http://localhost:9000`.
   - Log in with your admin credentials (default username is `admin`, and the password is either `admin` or your custom password if it was changed).
2. **Generate a New Token:**
   - Click on your **username** in the top-right corner of the SonarQube dashboard.
   - Select **My Account** from the dropdown menu.
   - Go to the **Security** tab.
   - Under **Generate Tokens**, type a name for the token (e.g., "Jenkins-SonarQube").
   - Click **Generate**.
   - Copy the token and save it securely. You will need it in Jenkins.

Step 2: Update the Token in Jenkins

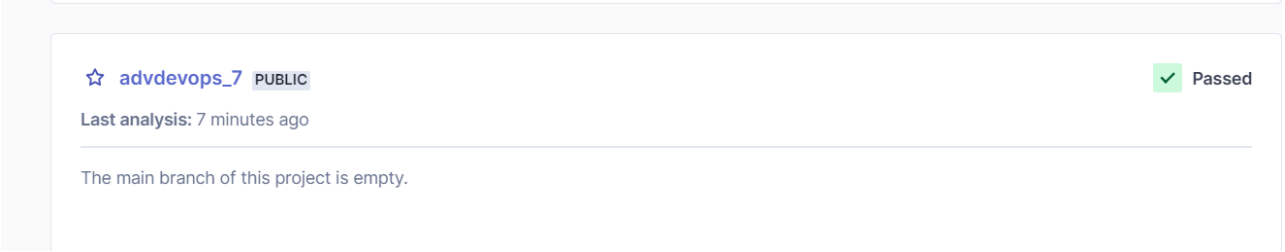
1. **Go to Jenkins Dashboard:**
   - Open Jenkins and log in with your credentials.
2. **Configure the Jenkins Job:**
   - Go to the job that is running the SonarQube scanner (`adv_devops_exp7`).
   - Click **Configure**.
3. **Update the SonarQube Token:**
   - In the SonarQube analysis configuration (either in the pipeline script or under "Build" section, depending on your job type), update the `sonar.login` parameter with the new token.

12. Run the Jenkins build.



Check the console Output

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



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

Conclusion:

In this project, we combined Jenkins with SonarQube to automate static application security testing (SAST). SonarQube was deployed via Docker, while Jenkins was configured with the required plugins and authentication. It was then connected to a GitHub repository. The SonarQube scanner was incorporated as a build step, facilitating ongoing code analysis for vulnerabilities, code quality issues, and other concerns, ensuring continuous improvement through automated reports.

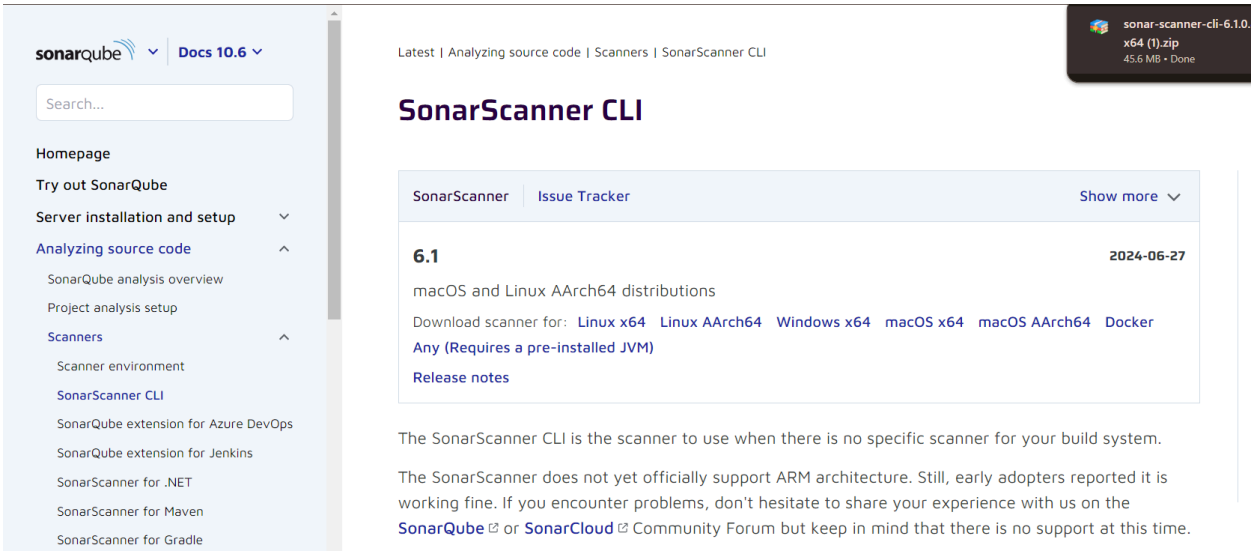
NAME: BHUSHAN MALPANI DIV: D15C ROLL NO: 33

08 Advanced DevOps Lab

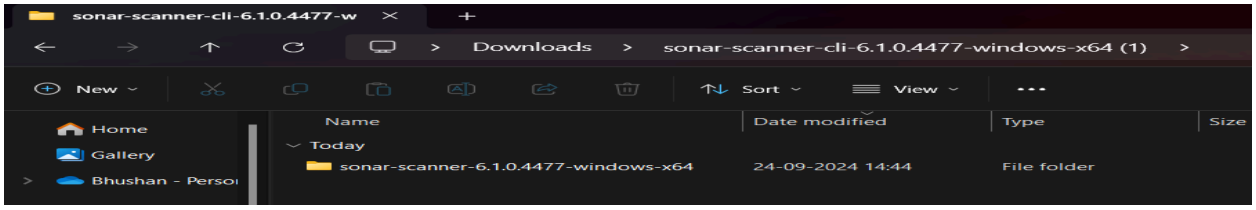
Aim: Create a Jenkins CICD Pipeline with SonarQube / GitLab Integration to perform a static analysis of the code to detect bugs, code smells, and security vulnerabilities on a sample Web / Java / Python application.

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.

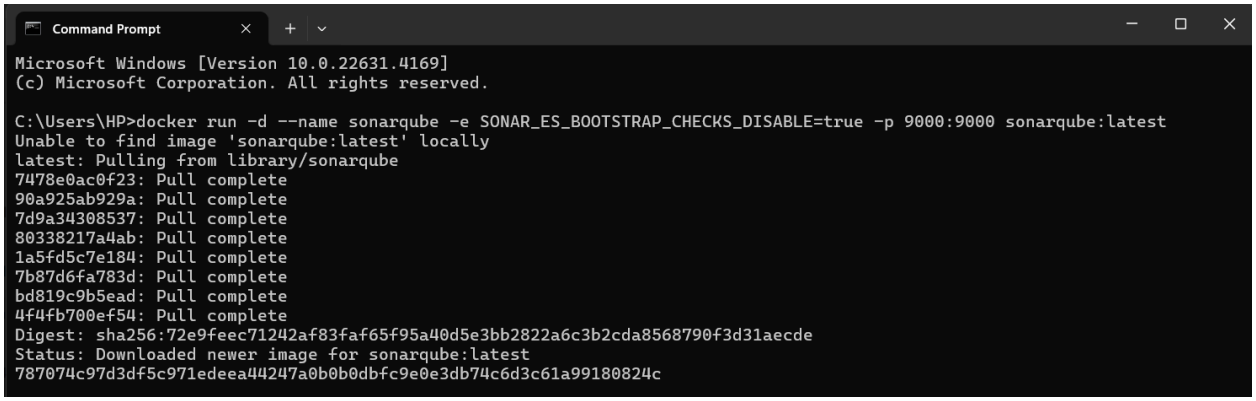


Extract the downloaded zip file in a folder.

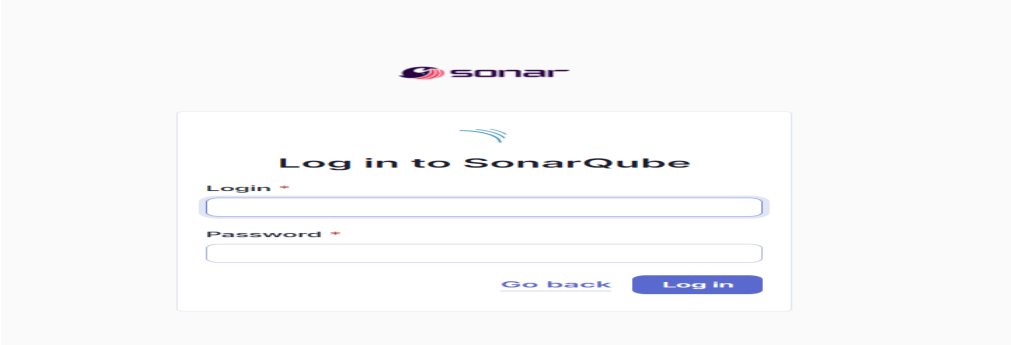


1. Install sonarqube image

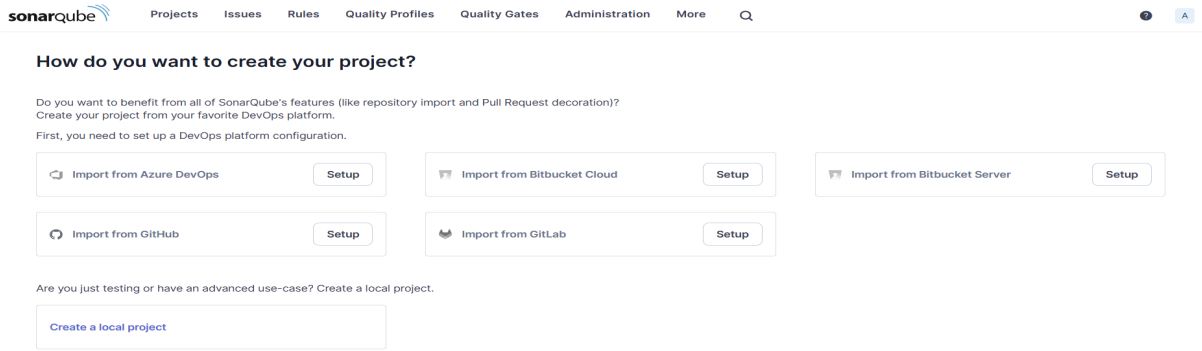
Command: **docker pull sonarqube**



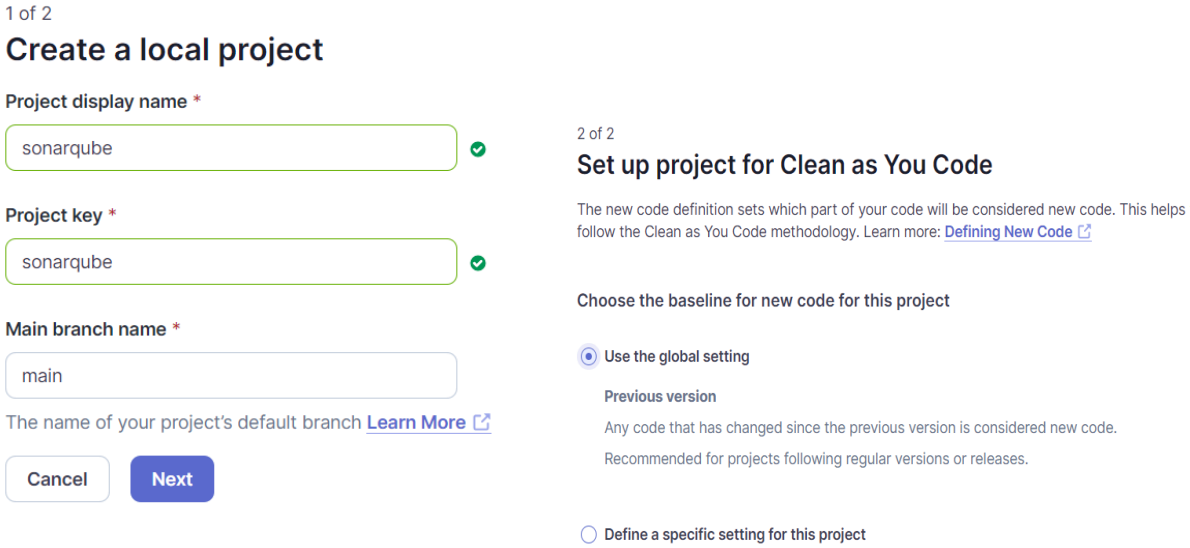
2. Once the container is up and running, you can check the status of SonarQube at localhost port 9000.



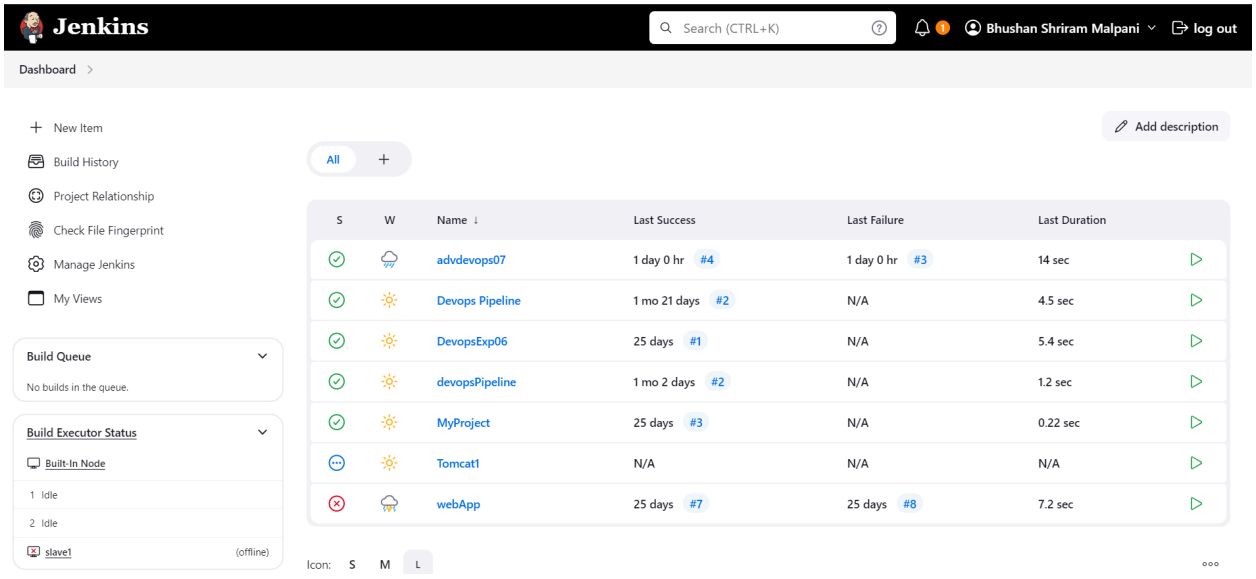
3. Login to SonarQube using username admin and password admin.



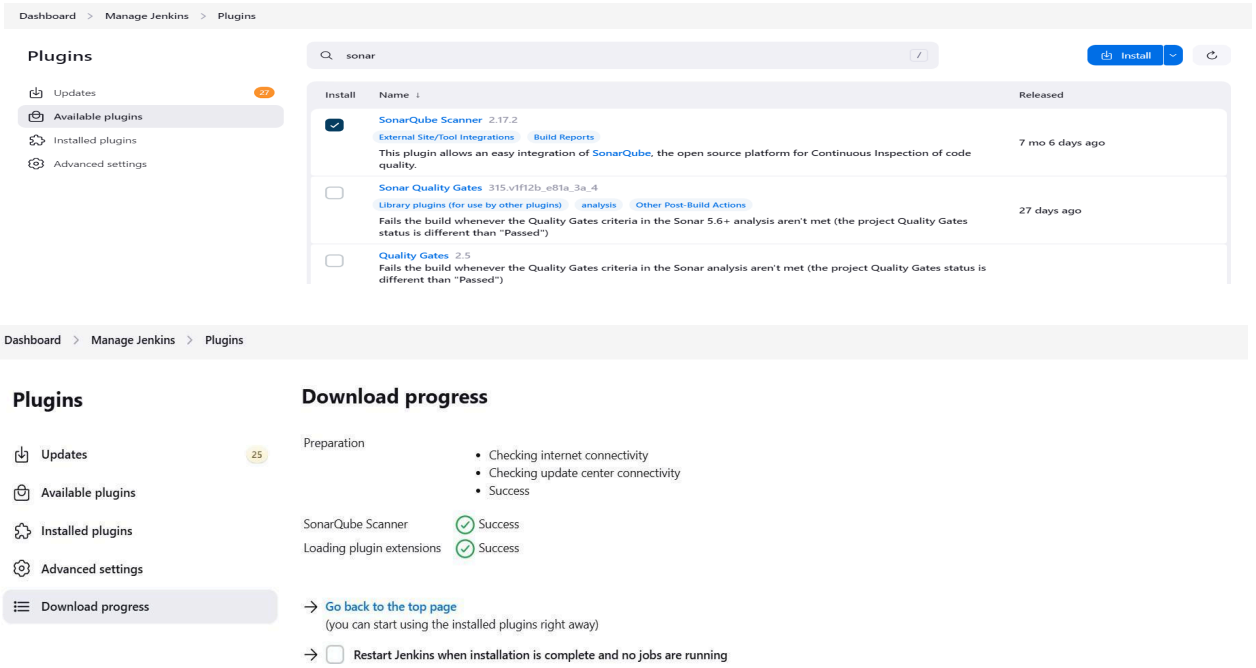
4. Create a manual project in SonarQube with the name sonarqube



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

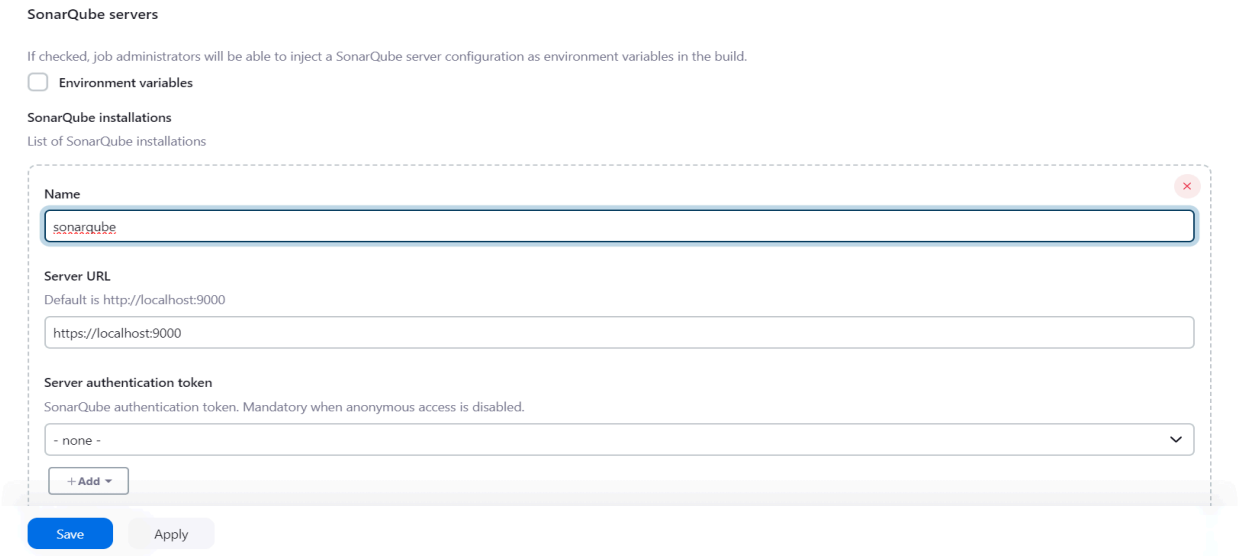


6. Go to Manage Jenkins and search for SonarQube Scanner for Jenkins and install it.



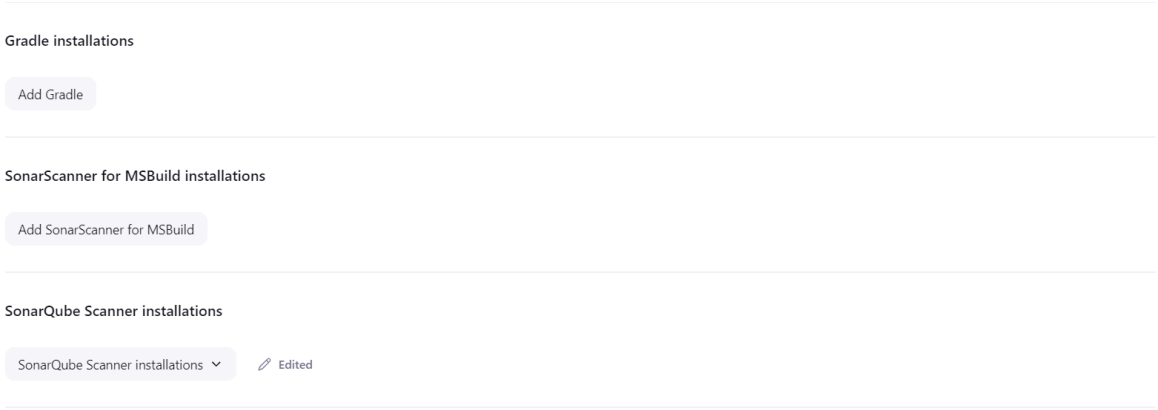
7. Under Jenkins ‘Manage Jenkins’ then go to ‘system’, scroll and look for **SonarQube Servers** and enter the details.
Enter the Server Authentication token if needed.

In SonarQube installations: Under **Name** add <project name of sonarqube> for me **adv_devops_7_sonarqube**
In **Server URL** Default is **http://localhost:9000**



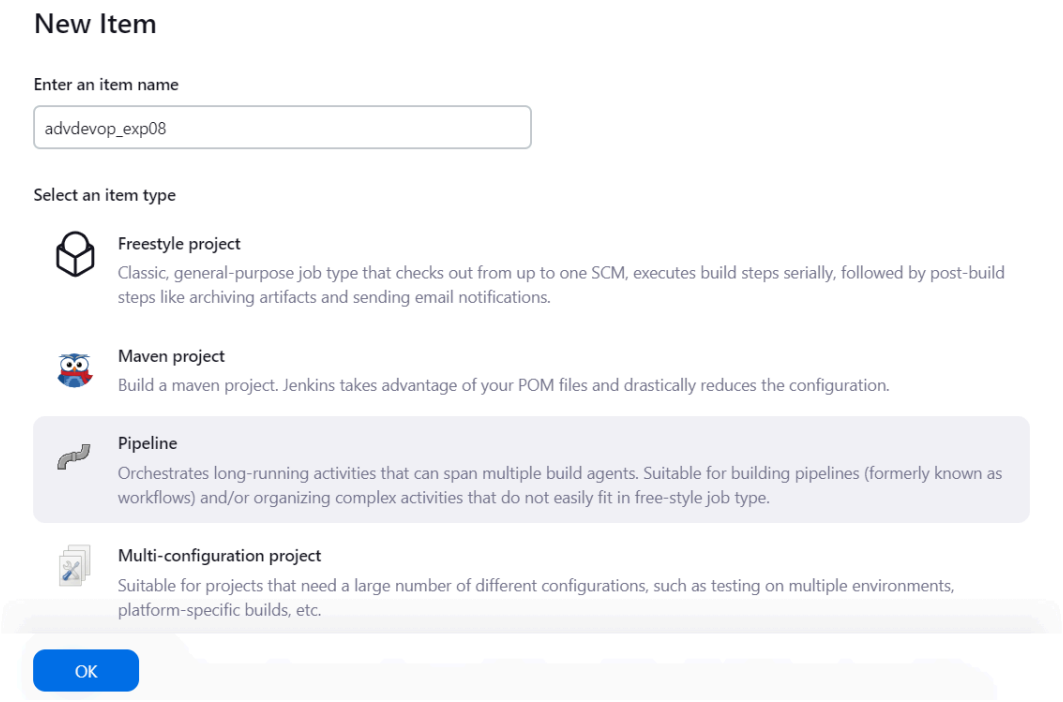
8. Search for SonarQube Scanner under Global Tool Configuration. Choose the latest configuration and choose Install automatically.

Dashboard > Manage Jenkins > Tools



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.

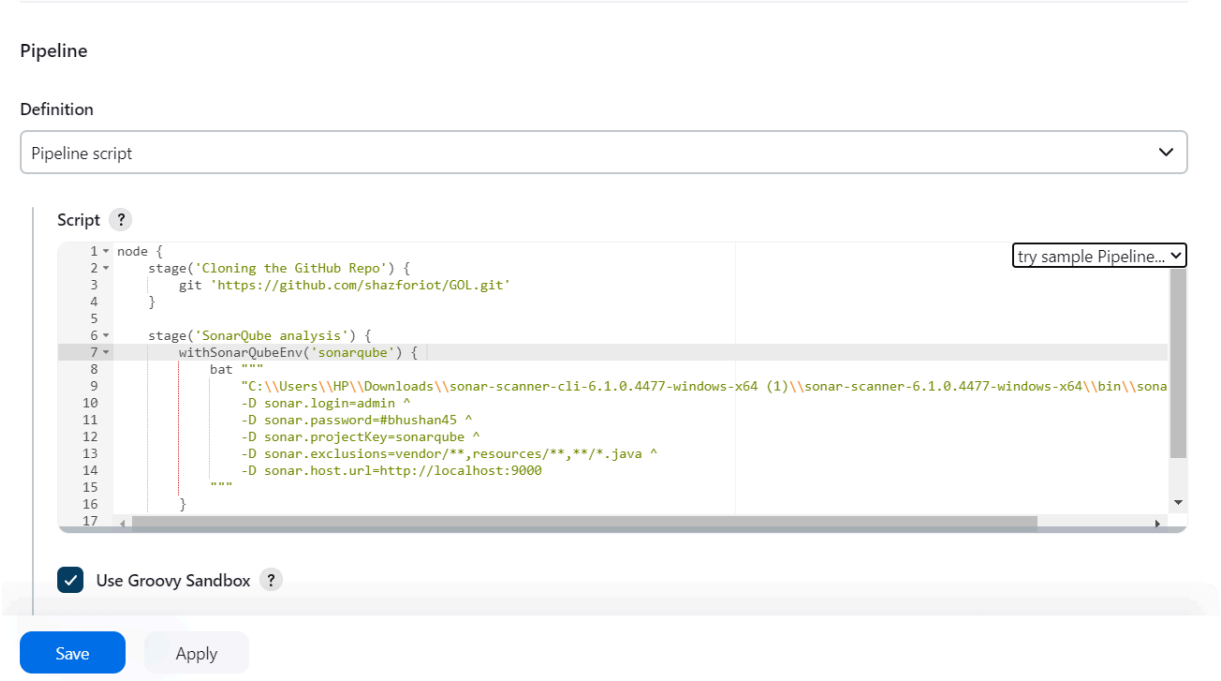


10. Under Pipeline script, enter the following:

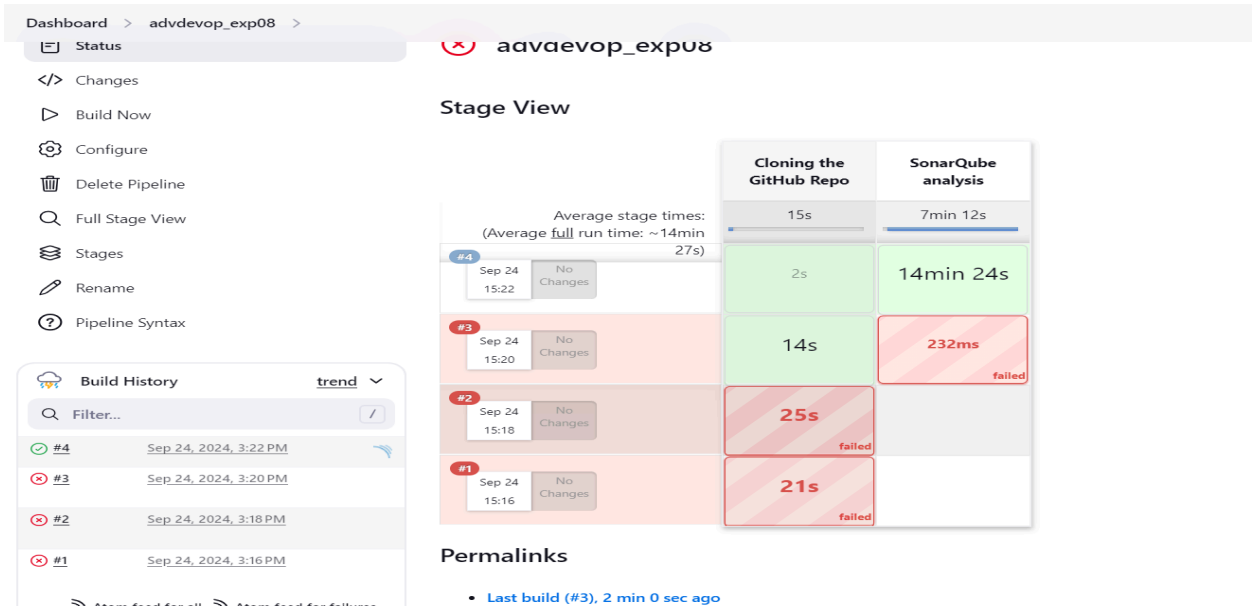
```
        node {
    stage('Cloning the GitHub Repo') {
        git 'https://github.com/shazforiot/GOL.git'
```

```
    }

    stage('SonarQube analysis') {
        withSonarQubeEnv('<Name_of_SonarQube_environment_on_Jenkins>') {
            sh """
                <PATH_TO_SONARQUBE_SCANNER_FOLDER>/bin/sonar-scanner \
                -D sonar.login=<SonarQube_USERNAME> \
                -D sonar.password=<SonarQube_PASSWORD> \
                -D sonar.projectKey=<Project_KEY> \
                -D sonar.exclusions=vendor/**,resources/**,**/*.java \
                -D sonar.host.url=<SonarQube_URL>(default: http://localhost:9000/)
            """
        }
    }
}
```

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

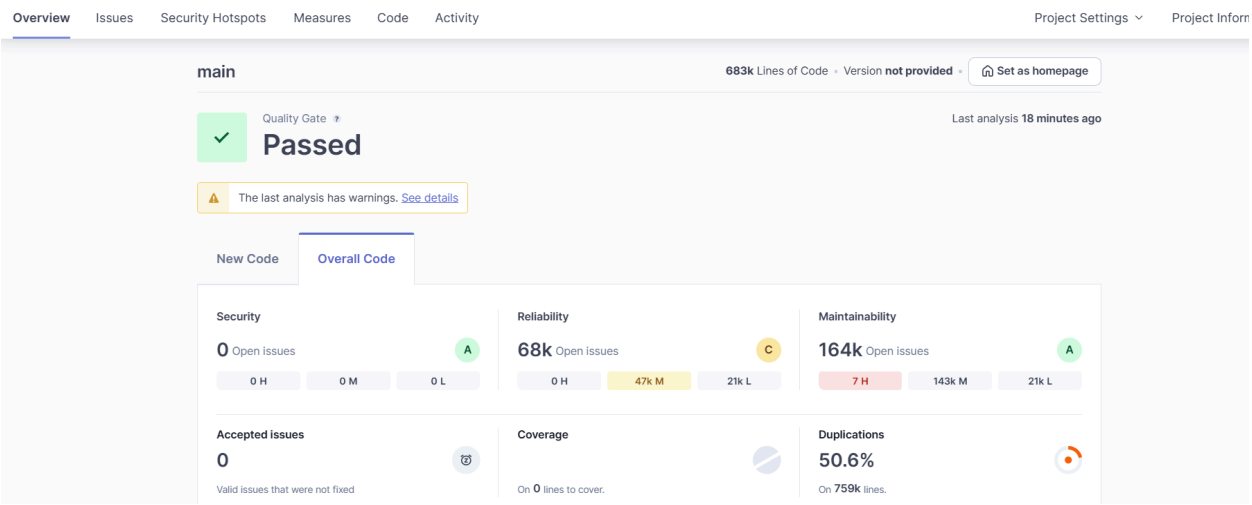


11. Build project



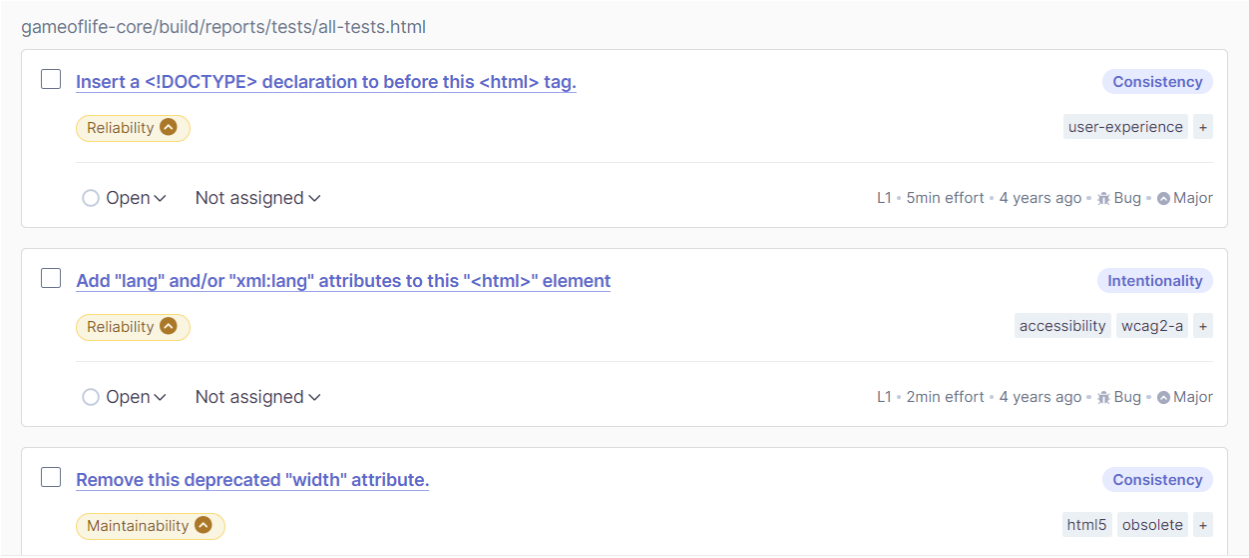
12. Check console

13. Now, check the project in SonarQube

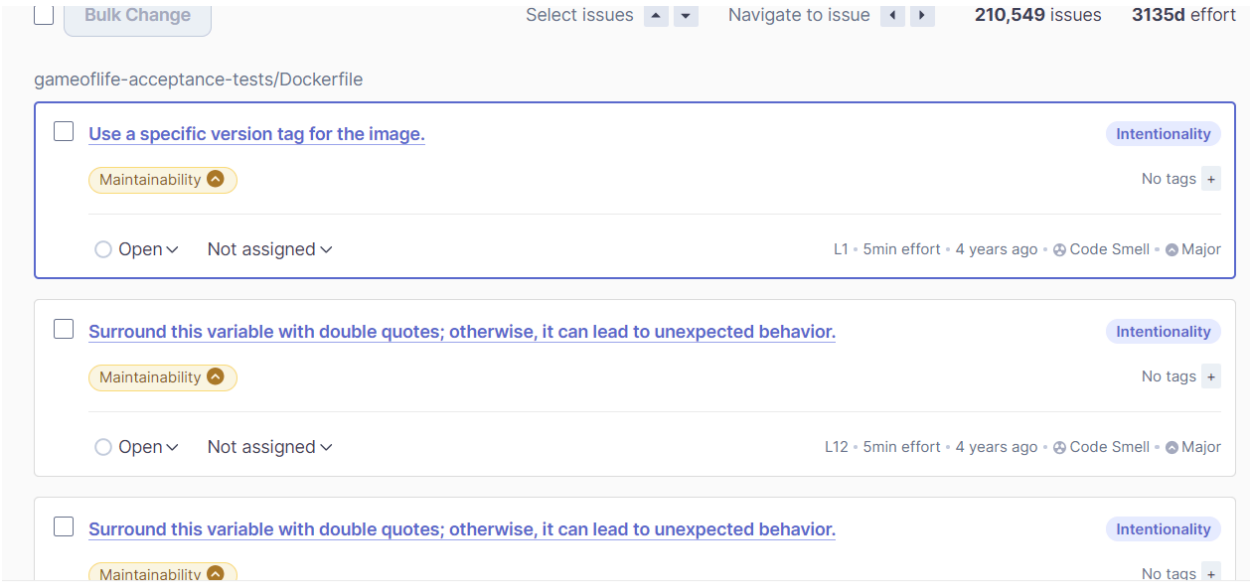


14. Code Problems

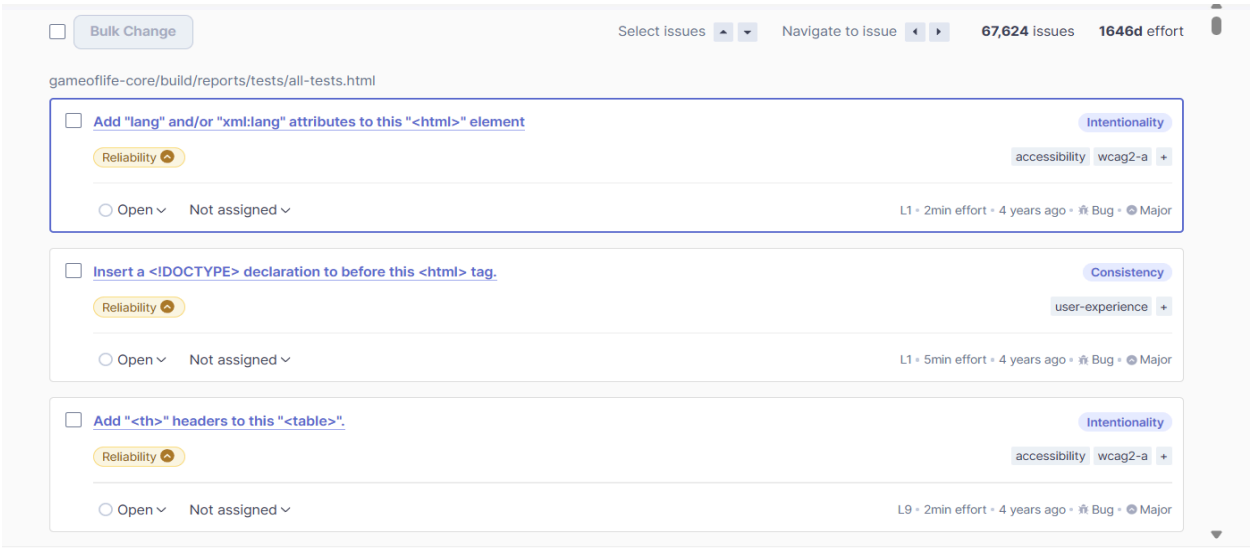
- Consistency



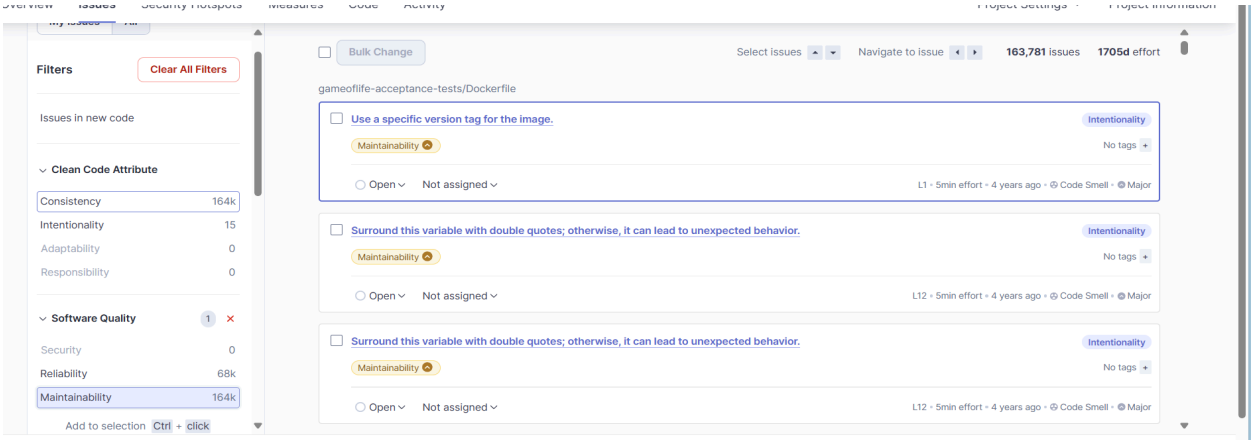
- Intentionality



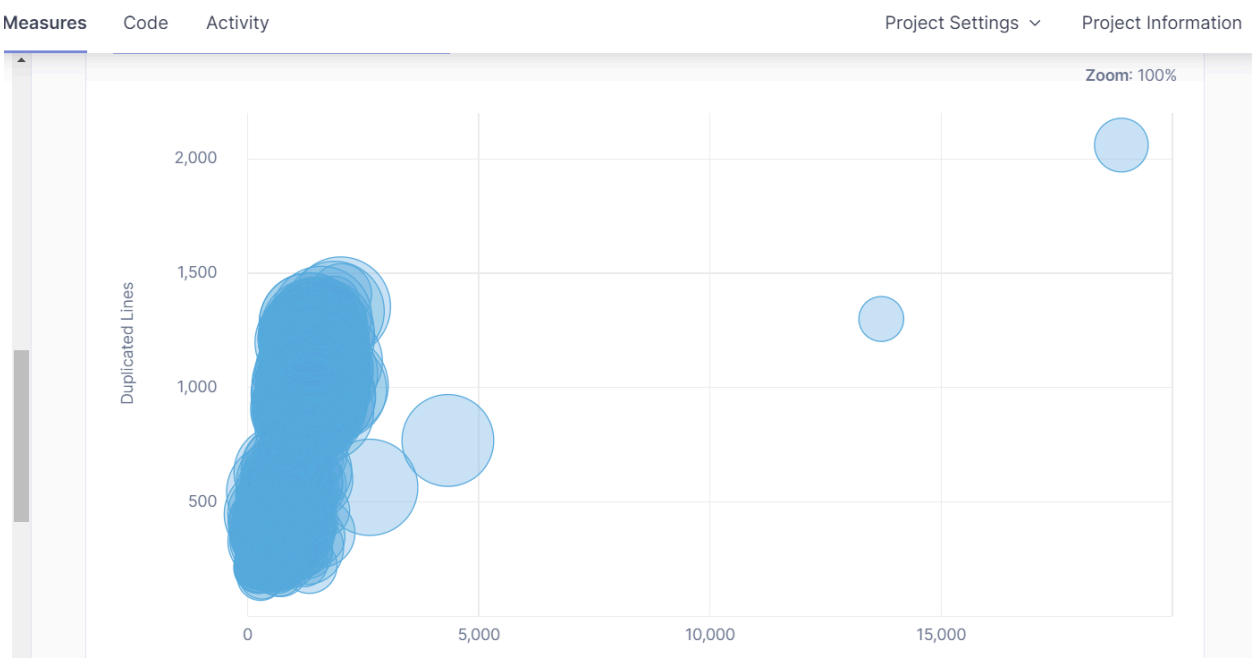
- Bugs



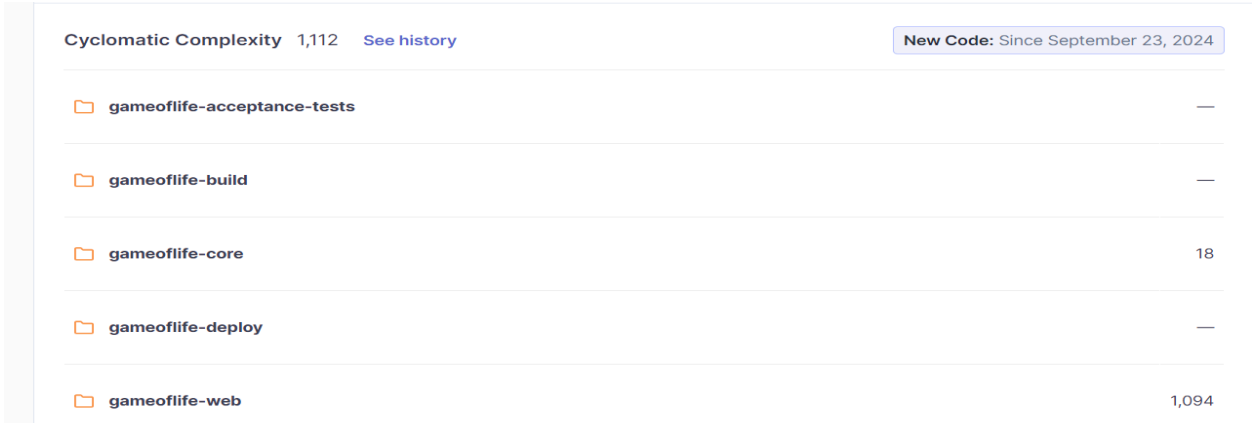
- Code Smells



- Duplications



- Cyclomatic Complexities



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

Conclusion:

In this experiment, we integrated Jenkins with SonarQube to implement automated code quality checks within our CI/CD pipeline. The process began with deploying SonarQube using Docker, followed by setting up a dedicated project within SonarQube to perform detailed code quality analysis. We then configured Jenkins by installing the SonarQube Scanner plugin, which allowed Jenkins to communicate with SonarQube. This included adding the necessary SonarQube server details and configuring the scanner tool to analyze code efficiently.

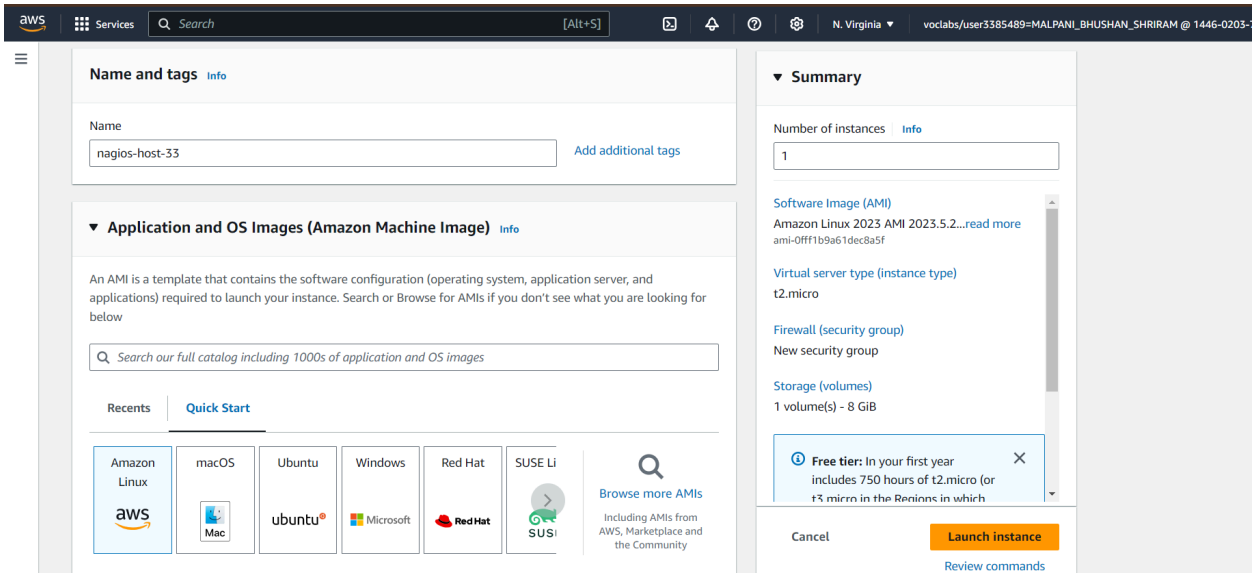
A Jenkins pipeline was developed to automate key tasks, such as cloning the code from a GitHub repository and triggering the SonarQube analysis on the codebase. Through this

integration, the pipeline continuously monitors code quality, providing detailed reports on issues such as bugs, code smells, and security vulnerabilities. This setup not only ensures early detection of potential problems but also promotes consistent improvements in code quality throughout the development lifecycle.

NAME: Bhushan Malpani **DIV: D15C** **ROLL NO:33**

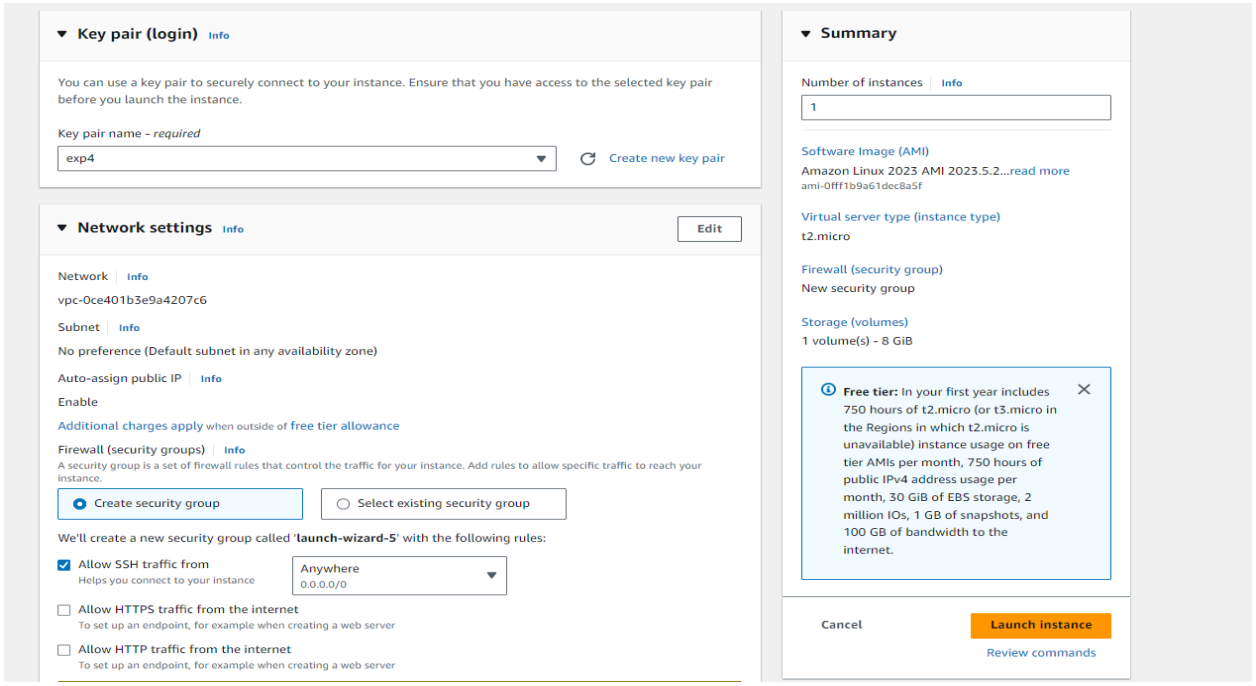
Aim: To Understand Continuous monitoring and Installation and configuration of Nagios Core, Nagios Plugins and NRPE (Nagios Remote Plugin Executor) on Linux Machine.

Step 1: Login to your AWS account. Search for EC2 on services. Open the interface and click on Create Instance.

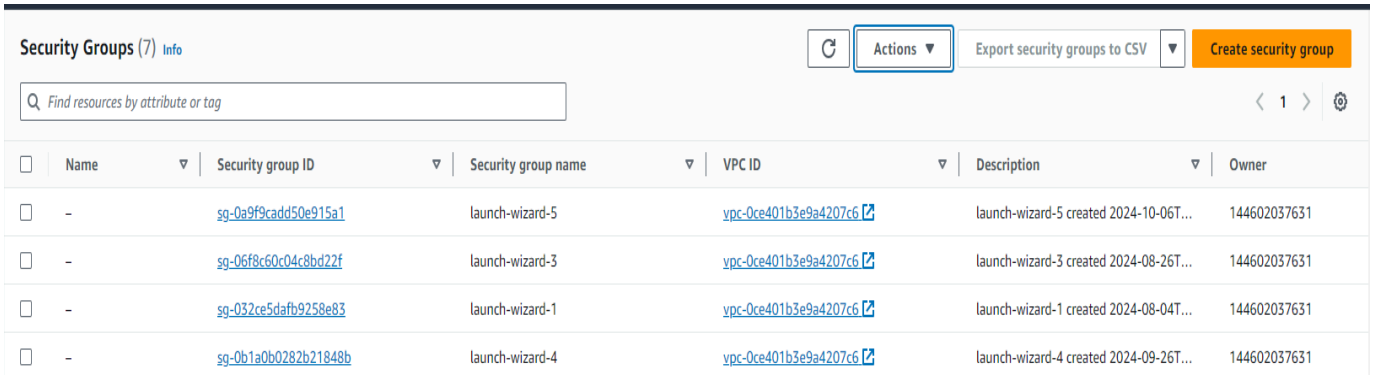


Select The OS Image as Amazon Linux.

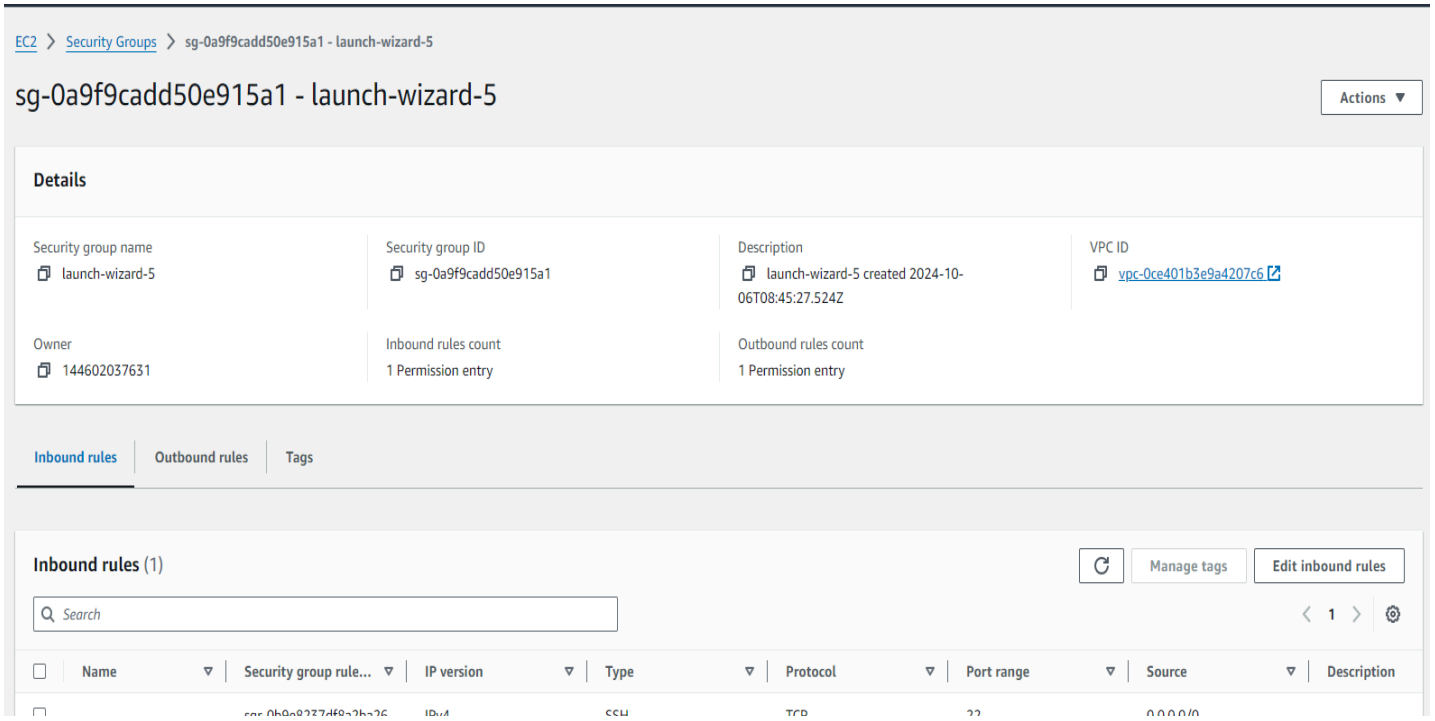
Step 2: If you do not have a private key created or a .pem file created, click on create a key pair. Else select the key pair that you had created before. (Make sure you know where the .pem file for that key is present on your system)



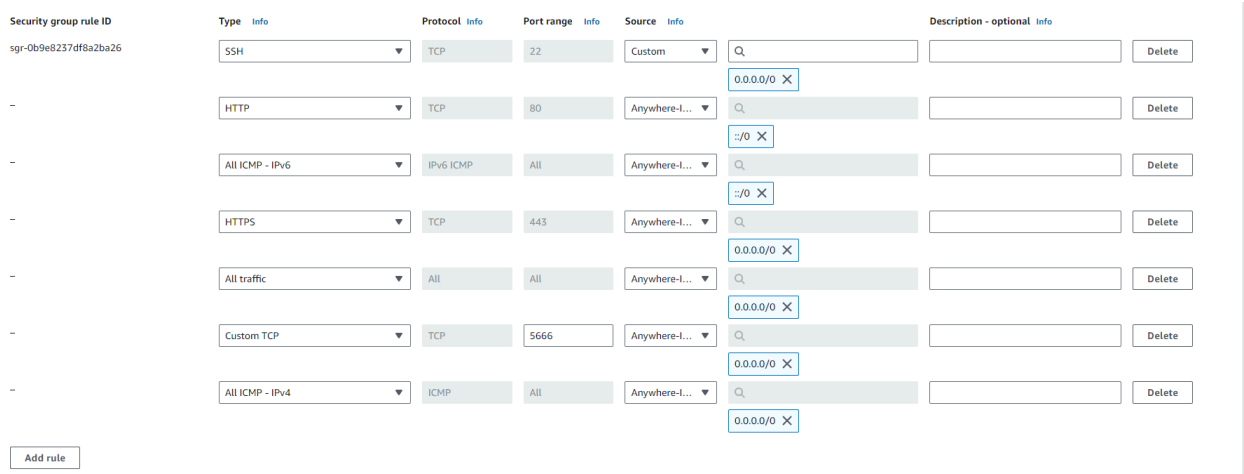
AWS will create a security group for this instance. Keep the name of that instance saved.
Step 3: After creating the instance, click on Security Groups from the left side pane. Find the security group that was created for your instance. Click on the instance ID for that group.



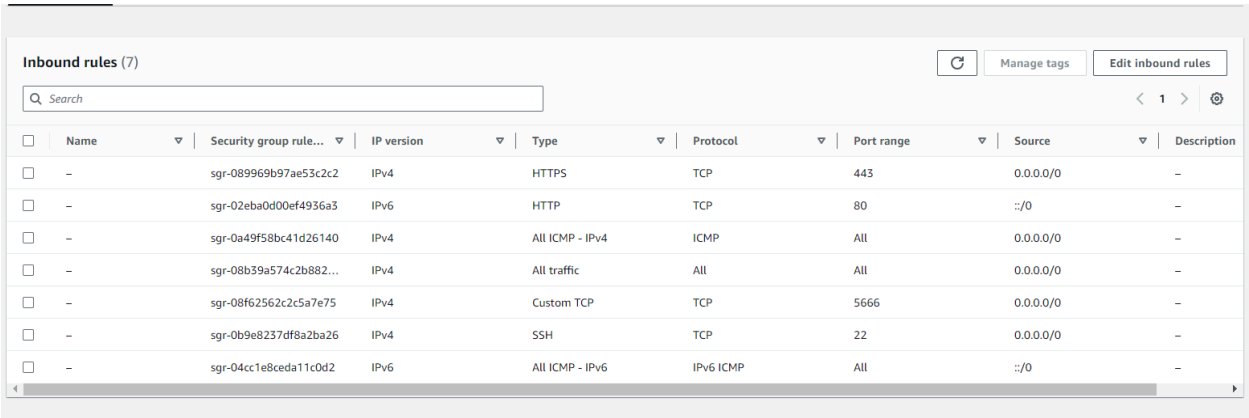
Here, click on Edit Inbound Rules.



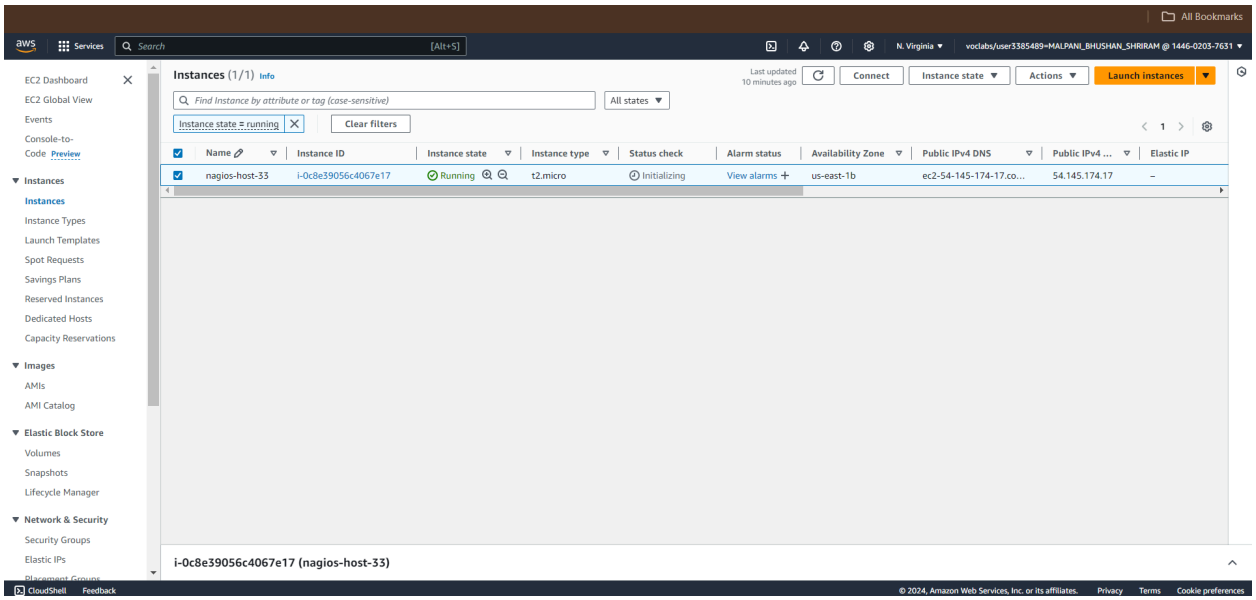
Now, click on add rules, and add teh rules for the following protocols:
HTTP, All ICMP - IPv6, HTTPS, All traffic, Custom TCP (Port 5666), All ICMP - IPv4



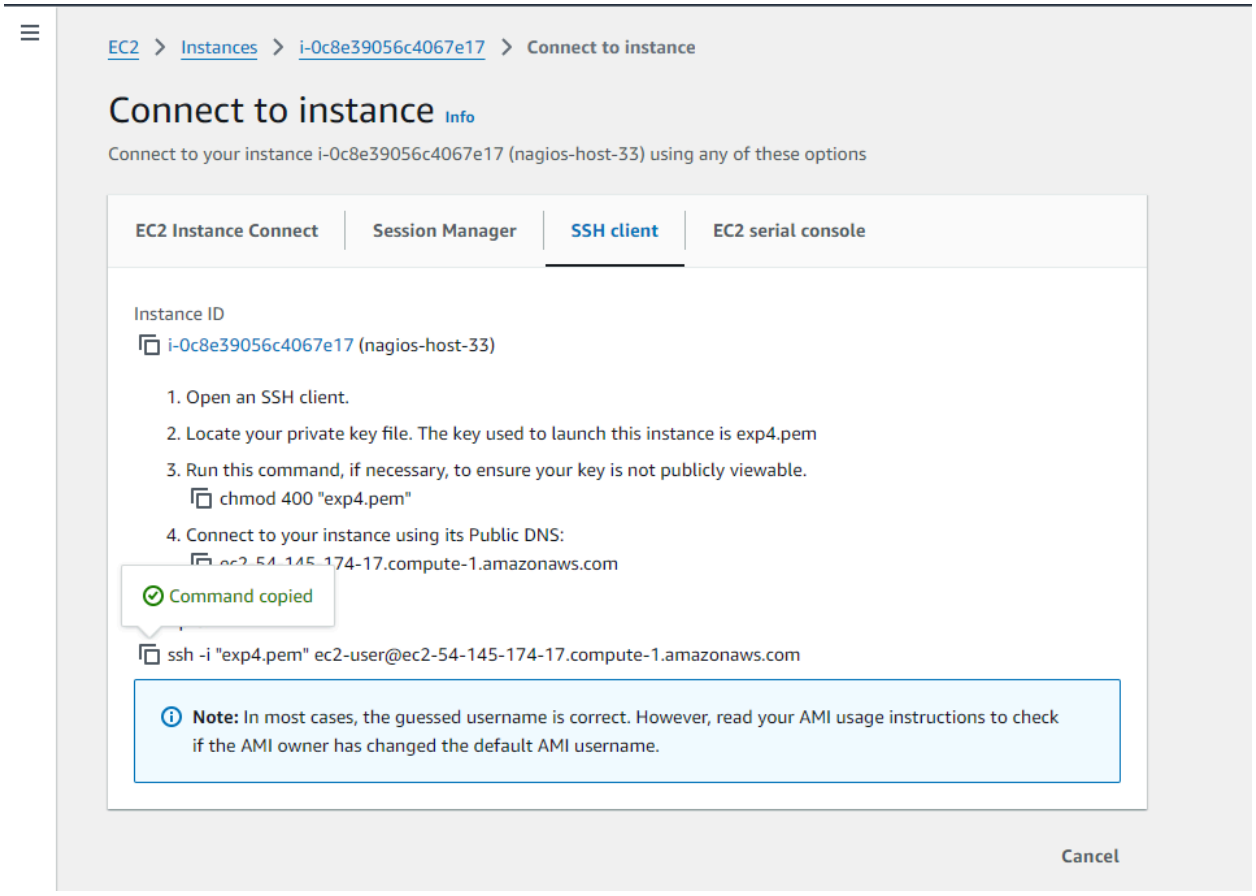
Click on save. This will add all the inbound rules to the security group.



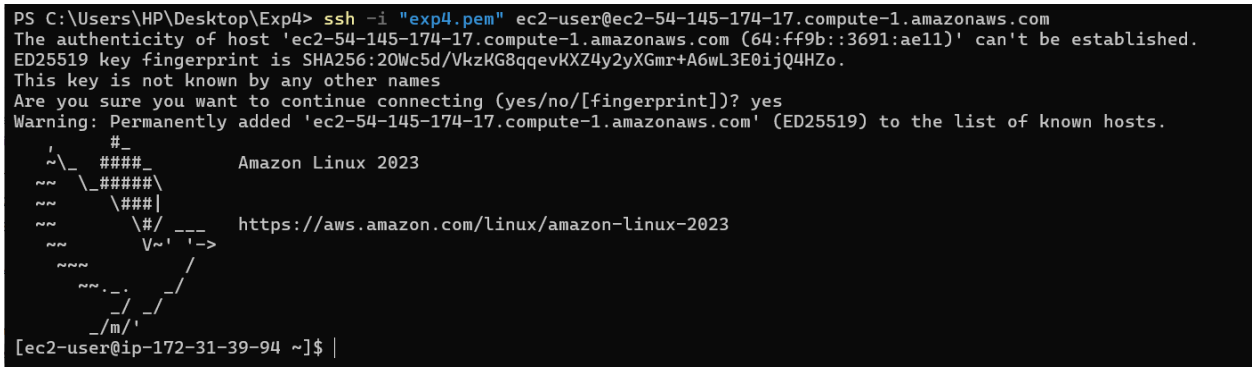
Step 4: Now come back to the instances screen. Click on the instance ID of your instance. Then click on Connect.



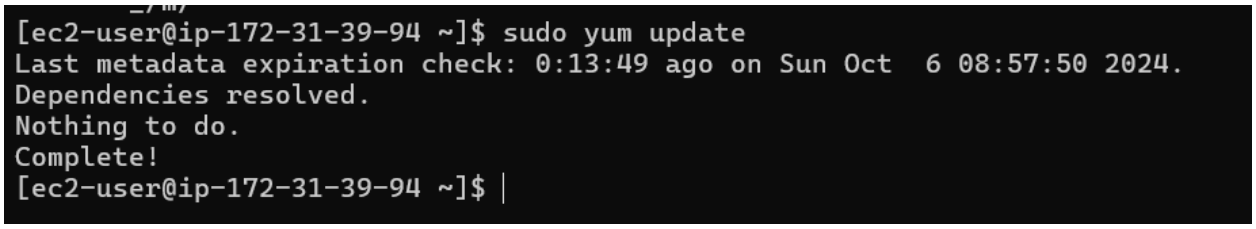
Click on SSH client. Copy the example command.



Step 5: Now, we have to connect our local OS terminal to the instance using SSH. For this,
Open terminal wher the private key file is located (.pem)
Paste the copied SSH command and run it.



Step 6: Now we start working on this terminal. First run the command
sudo yum update
This command will check for any updates for the YUM library.



Step 7: We are going to install an Apache server and a PHP on this instance. For that, run this command.
sudo yum install httpd php

```
[ec2-user@ip-172-31-39-94 ~]$ sudo yum install httpd php
Last metadata expiration check: 0:14:13 ago on Sun Oct  6 08:57:50 2024.
Dependencies resolved.
================================================================================
 Package                 Architecture    Version                 Repository    Size
================================================================================
Installing:
 httpd                   x86_64          2.4.62-1.amzn2023       amazonlinux   48 k
 php8.3                  x86_64          8.3.10-1.amzn2023.0.1   amazonlinux   10 k
Installing dependencies:
 apr                     x86_64          1.7.2-2.amzn2023.0.2    amazonlinux  129 k
 apr-util                x86_64          1.6.3-1.amzn2023.0.1    amazonlinux   98 k
 generic-logos-httpd     noarch          18.0.0-12.amzn2023.0.3  amazonlinux   19 k
 httpd-core              x86_64          2.4.62-1.amzn2023       amazonlinux  1.4 M
 httpd-filesystem        noarch          2.4.62-1.amzn2023       amazonlinux   14 k
 httpd-tools             x86_64          2.4.62-1.amzn2023       amazonlinux   81 k
 libbrotli               x86_64          1.0.9-4.amzn2023.0.2    amazonlinux  315 k
 libsodium               x86_64          1.0.19-4.amzn2023       amazonlinux  176 k
 libxslt                 x86_64          1.1.34-5.amzn2023.0.2   amazonlinux  241 k
 mailcap                 noarch          2.1.49-3.amzn2023.0.3   amazonlinux   33 k
 nginx-filesystem        noarch          1:1.24.0-1.amzn2023.0.4 amazonlinux  9.8 k
 php8.3-cli              x86_64          8.3.10-1.amzn2023.0.1   amazonlinux  3.7 M
 php8.3-common           x86_64          8.3.10-1.amzn2023.0.1   amazonlinux  737 k
 php8.3-process          x86_64          8.3.10-1.amzn2023.0.1   amazonlinux   45 k
 php8.3-xml              x86_64          8.3.10-1.amzn2023.0.1   amazonlinux  154 k
Installing weak dependencies:
 apr-util-openssl        x86_64          1.6.3-1.amzn2023.0.1    amazonlinux   17 k
 mod_http2               x86_64          2.0.27-1.amzn2023.0.3   amazonlinux  166 k
 mod_lua                 x86_64          2.4.62-1.amzn2023       amazonlinux   61 k
 php8.3-fpm              x86_64          8.3.10-1.amzn2023.0.1   amazonlinux  1.9 M
```

```
Installed:
  apr-1.7.2-2.amzn2023.0.2.x86_64                apr-util-1.6.3-1.amzn2023.0.1.x86_64
  apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64   generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch
  httpd-2.4.62-1.amzn2023.x86_64                 httpd-core-2.4.62-1.amzn2023.x86_64
  httpd-filesystem-2.4.62-1.amzn2023.noarch      httpd-tools-2.4.62-1.amzn2023.x86_64
  libbrotli-1.0.9-4.amzn2023.0.2.x86_64          libsodium-1.0.19-4.amzn2023.x86_64
  libxslt-1.1.34-5.amzn2023.0.2.x86_64           mailcap-2.1.49-3.amzn2023.0.3.noarch
  mod_http2-2.0.27-1.amzn2023.0.3.x86_64         mod_lua-2.4.62-1.amzn2023.x86_64
  nginx-filesystem-1:1.24.0-1.amzn2023.0.4.noarch php8.3-8.3.10-1.amzn2023.0.1.x86_64
  php8.3-cli-8.3.10-1.amzn2023.0.1.x86_64        php8.3-common-8.3.10-1.amzn2023.0.1.x86_64
  php8.3-fpm-8.3.10-1.amzn2023.0.1.x86_64        php8.3-mbstring-8.3.10-1.amzn2023.0.1.x86_64
  php8.3-opcache-8.3.10-1.amzn2023.0.1.x86_64    php8.3-pdo-8.3.10-1.amzn2023.0.1.x86_64
  php8.3-process-8.3.10-1.amzn2023.0.1.x86_64    php8.3-sodium-8.3.10-1.amzn2023.0.1.x86_64
  php8.3-xml-8.3.10-1.amzn2023.0.1.x86_64

Complete!
[ec2-user@ip-172-31-39-94 ~]$
```

Step 8: Next we install C/C++ compiler (GCC) along with the necessary C libraries required for compiling and running C programs. Use the following command.
sudo yum install gcc glibc glibc-common

```
[ec2-user@ip-172-31-39-94 ~]$ sudo yum install gcc glibc glibc-common
Last metadata expiration check: 0:15:54 ago on Sun Oct  6 08:57:50 2024.
Package glibc-2.34-52.amzn2023.0.11.x86_64 is already installed.
Package glibc-common-2.34-52.amzn2023.0.11.x86_64 is already installed.
Dependencies resolved.
================================================================================
 Package                 Architecture    Version                 Repository    Size
================================================================================
Installing:
 gcc                     x86_64          11.4.1-2.amzn2023.0.2   amazonlinux   32 M
Installing dependencies:
 annobin-docs            noarch          10.93-1.amzn2023.0.1    amazonlinux   92 k
 annobin-plugin-gcc      x86_64          10.93-1.amzn2023.0.1    amazonlinux  887 k
 cpp                     x86_64          11.4.1-2.amzn2023.0.2   amazonlinux   10 M
 gc                      x86_64          8.0.4-5.amzn2023.0.2    amazonlinux  105 k
 glibc-devel             x86_64          2.34-52.amzn2023.0.11   amazonlinux   27 k
 glibc-headers-x86       noarch          2.34-52.amzn2023.0.11   amazonlinux  427 k
 guile22                 x86_64          2.2.7-2.amzn2023.0.3    amazonlinux  6.4 M
 kernel-headers          x86_64          6.1.109-118.189.amzn2023 amazonlinux 1.4 M
 libmpc                  x86_64          1.2.1-2.amzn2023.0.2    amazonlinux   62 k
 libtool-ltdl            x86_64          2.4.7-1.amzn2023.0.3    amazonlinux   38 k
 libxcrypt-devel         x86_64          4.4.33-7.amzn2023       amazonlinux   32 k
 make                    x86_64          1:4.3-5.amzn2023.0.2    amazonlinux  534 k

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

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

Step 9: We would also need GD library and its development tools. For that, run this command sudo yum install gd gd-devel

```
[ec2-user@ip-172-31-39-94 ~]$ sudo yum install gd gd-devel
Last metadata expiration check: 0:17:02 ago on Sun Oct  6 08:57:50 2024.
Dependencies resolved.
================================================================================================================================
 Package                          Architecture     Version                                Repository              Size
================================================================================================================================
Installing:
 gd                               x86_64           2.3.3-5.amzn2023.0.3                   amazonlinux            139 k
 gd-devel                         x86_64           2.3.3-5.amzn2023.0.3                   amazonlinux             38 k
Installing dependencies:
 brotli                           x86_64           1.0.9-4.amzn2023.0.2                   amazonlinux            314 k
 brotli-devel                     x86_64           1.0.9-4.amzn2023.0.2                   amazonlinux             31 k
 bzip2-devel                      x86_64           1.0.8-6.amzn2023.0.2                   amazonlinux            214 k
 cairo                            x86_64           1.17.6-2.amzn2023.0.1                  amazonlinux            684 k
 cmake-filesystem                 x86_64           3.22.2-1.amzn2023.0.4                  amazonlinux             16 k
 fontconfig                       x86_64           2.13.94-2.amzn2023.0.2                 amazonlinux            273 k
 fontconfig-devel                 x86_64           2.13.94-2.amzn2023.0.2                 amazonlinux            128 k
 fonts-filesystem                 noarch           1:2.0.5-12.amzn2023.0.2                amazonlinux            9.5 k
 freetype                         x86_64           2.13.2-5.amzn2023.0.1                  amazonlinux            423 k
 freetype-devel                   x86_64           2.13.2-5.amzn2023.0.1                  amazonlinux            912 k
 glib2-devel                      x86_64           2.74.7-689.amzn2023.0.2                amazonlinux            486 k
 google-noto-fonts-common         noarch           20201206-2.amzn2023.0.2                amazonlinux             15 k
 google-noto-sans-vf-fonts        noarch           20201206-2.amzn2023.0.2                amazonlinux            492 k
 graphite2                        x86_64           1.3.14-7.amzn2023.0.2                  amazonlinux             97 k
 graphite2-devel                  x86_64           1.3.14-7.amzn2023.0.2                  amazonlinux             21 k
 harfbuzz                         x86_64           7.0.0-2.amzn2023.0.1                   amazonlinux            868 k
 harfbuzz-devel                   x86_64           7.0.0-2.amzn2023.0.1                   amazonlinux            404 k
 harfbuzz-icu                     x86_64           7.0.0-2.amzn2023.0.1                   amazonlinux             18 k
 jbigkit-libs                     x86_64           2.1-21.amzn2023.0.2                    amazonlinux             54 k
 langpacks-core-font-en           noarch           3.0-21.amzn2023.0.4                    amazonlinux             10 k

Installed:
  brotli-1.0.9-4.amzn2023.0.2.x86_64                          brotli-devel-1.0.9-4.amzn2023.0.2.x86_64
  bzip2-devel-1.0.8-6.amzn2023.0.2.x86_64                     cairo-1.17.6-2.amzn2023.0.1.x86_64
  cmake-filesystem-3.22.2-1.amzn2023.0.4.x86_64               fontconfig-2.13.94-2.amzn2023.0.2.x86_64
  fontconfig-devel-2.13.94-2.amzn2023.0.2.x86_64              fonts-filesystem-1:2.0.5-12.amzn2023.0.2.noarch
  freetype-2.13.2-5.amzn2023.0.1.x86_64                       freetype-devel-2.13.2-5.amzn2023.0.1.x86_64
  gd-2.3.3-5.amzn2023.0.3.x86_64                              gd-devel-2.3.3-5.amzn2023.0.3.x86_64
  glib2-devel-2.74.7-689.amzn2023.0.2.x86_64                  google-noto-fonts-common-20201206-2.amzn2023.0.2.noarch
  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

Complete!
[ec2-user@ip-172-31-39-94 ~]$
```

Step 10: Now, we create a user called as ‘nagios’ and make sure that it has a home directory, and set up a password for it.
sudo adduser -m nagios
sudo passwd nagios

```
[ec2-user@ip-172-31-39-94 ~]$ sudo adduser -m nagios
sudo passwd nagios
Changing password for user nagios.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[ec2-user@ip-172-31-39-94 ~]$
```

Step 11: Create a user group called as ‘nagcmd’ to execute nagios commands.
sudo groupadd nagcmd

```
[ec2-user@ip-172-31-39-94 ~]$ sudo groupadd nagcmd
[ec2-user@ip-172-31-39-94 ~]$
```

Step 12: Add users apache and nagios to this user group.
sudo usermod -a -G nagcmd nagios
sudo usermod -a -G nagcmd apache

```
[ec2-user@ip-172-31-39-94 ~]$ sudo usermod -a -G nagcmd nagios
sudo usermod -a -G nagcmd apache
```

Step 13: We create a directory downloads, to store the files of nagios server that are downloaded.
mkdir ~/downloads
cd ~/downloads

```
[ec2-user@ip-172-31-39-94 ~]$ mkdir ~/downloads
cd ~/downloads
```

Step 14:Now we need to install the latest versions of nogios-core and nagios-plugins. Go to the respective websites and check whether a better version is available.If newer versions are available, then right click on the download button → Copy link address.
Paste this link address in place of the current link in command.
If not run these commands.

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

```
[ec2-user@ip-172-31-39-94 downloads]$ wget https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.5.tar.gz
--2024-10-06 09:18:48--  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  5.66MB/s    in 0.3s

2024-10-06 09:18:49 (5.66 MB/s) - ‘nagios-4.5.5.tar.gz’ saved [2065473/2065473]
```

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

```
[ec2-user@ip-172-31-39-94 downloads]$ wget https://nagios-plugins.org/download/nagios-plugins-2.4.11.tar.gz
--2024-10-06 09:19:14--  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  6.48MB/s    in 0.4s

2024-10-06 09:19:15 (6.48 MB/s) - ‘nagios-plugins-2.4.11.tar.gz’ saved [2753049/2753049]
```

Step 15:Now, we need to extract nagios-core file into the same directory. For that, we will use tar command.
tar zxvf nagios-4.5.5.tar.gz

```
[ec2-user@ip-172-31-39-94 downloads]$ tar zxvf nagios-4.5.5.tar.gz
nagios-4.5.5/
nagios-4.5.5/.github/
nagios-4.5.5/.github/workflows/
nagios-4.5.5/.github/workflows/test.yml
nagios-4.5.5/.gitignore
nagios-4.5.5/CONTRIBUTING.md
nagios-4.5.5/Changelog
nagios-4.5.5/INSTALLING
nagios-4.5.5/LEGAL
nagios-4.5.5/LICENSE
nagios-4.5.5/Makefile.in
nagios-4.5.5/README.md
nagios-4.5.5/THANKS
nagios-4.5.5/UPGRADING
nagios-4.5.5/aclocal.m4
nagios-4.5.5/autoconf-macros/
nagios-4.5.5/autoconf-macros/.gitignore
nagios-4.5.5/autoconf-macros/CHANGELOG.md
nagios-4.5.5/autoconf-macros/LICENSE
nagios-4.5.5/autoconf-macros/LICENSE.md
nagios-4.5.5/autoconf-macros/README.md
nagios-4.5.5/autoconf-macros/add_group_user
nagios-4.5.5/autoconf-macros/ax_nagios_get_distrib
nagios-4.5.5/autoconf-macros/ax_nagios_get_files
nagios-4.5.5/autoconf-macros/ax_nagios_get_inetd
nagios-4.5.5/autoconf-macros/ax_nagios_get_init
nagios-4.5.5/autoconf-macros/ax_nagios_get_os
nagios-4.5.5/autoconf-macros/ax_nagios_get_paths
nagios-4.5.5/autoconf-macros/ax_nagios_get_ssl
```

```
nagios-4.5.5/xdata/.gitignore
nagios-4.5.5/xdata/Makefile.in
nagios-4.5.5/xdata/xcddefault.c
nagios-4.5.5/xdata/xcddefault.h
nagios-4.5.5/xdata/xodtemplate.c
nagios-4.5.5/xdata/xodtemplate.h
nagios-4.5.5/xdata/xpddefault.c
nagios-4.5.5/xdata/xpddefault.h
nagios-4.5.5/xdata/xrddefault.c
nagios-4.5.5/xdata/xrddefault.h
nagios-4.5.5/xdata/xsddefault.c
nagios-4.5.5/xdata/xsddefault.h
[ec2-user@ip-172-31-39-94 downloads]$
```

Step 16: We need to ensure that Nagios uses a specific group (in this case, `nagcmd`) for executing external commands.
./configure --with-command-group=nagcmd

An error could be encountered here: ./configure: no such path or directory

Solution: Navigate to the nagios-4.5.5 folder in downloads. (version could vary)

Steps: ls

```
[ec2-user@ip-172-31-39-94 downloads]$ ls
nagios-4.5.5  nagios-4.5.5.tar.gz  nagios-plugins-2.4.11.tar.gz
```

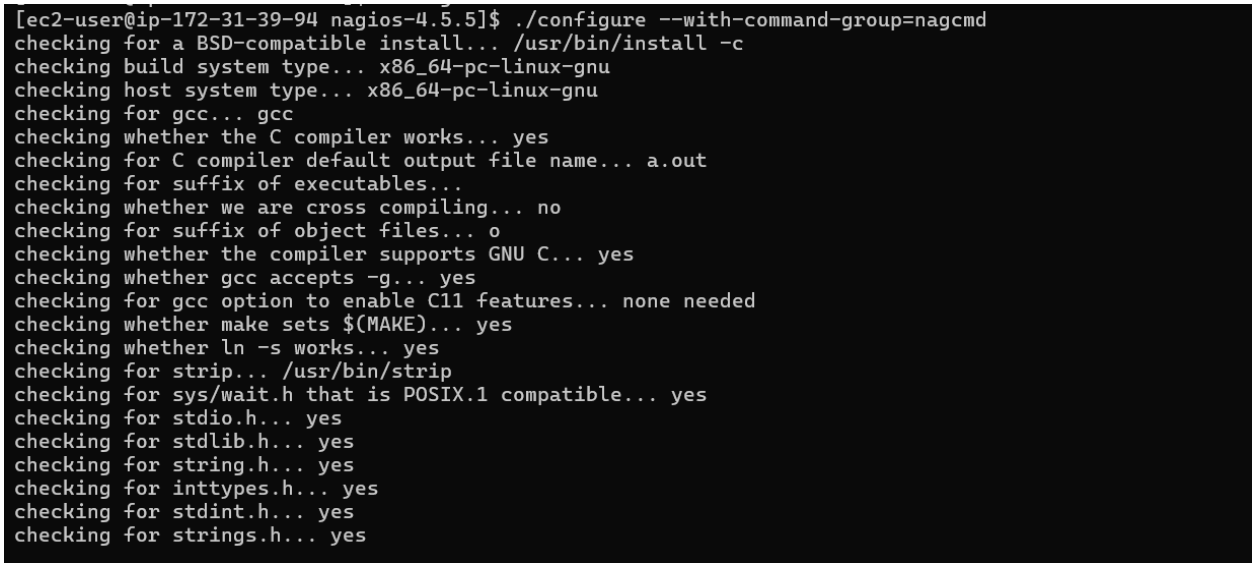
- cd nagios-4.5.5 (use the version shown by your ls command)
- ./configure --with-command-group=nagcmd

Another error could be Cannot find SSL headers.

To solve this, we need to install OpenSSL Dev Library

Steps:
sudo yum install openssl-devel

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



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

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

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


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

[ec2-user@ip-172-31-39-94 nagios-4.5.5]$
```

Step 17: We need to compile all components of this software according to the instruction in the Makefile. For that, use this command:
make all

Then,
sudo make install
sudo make install-init
sudo make install-config
sudo make install-commandmode

```
[ec2-user@ip-172-31-39-94 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/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: cannot stat 'nagios': No such file or directory
make[1]: *** [Makefile:188: install] Error 1
make[1]: Leaving directory '/home/ec2-user/nagios-4.5.5/base'
make: *** [Makefile:287: install] Error 2
/usr/bin/install -c -m 755 -d -o root -g root /lib/systemd/system
/usr/bin/install -c -m 755 -o root -g root startup/default-service /lib/systemd/system/nagios.service
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc/objects
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/nagios.cfg /usr/local/nagios/etc/nagios.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/cgi.cfg /usr/local/nagios/etc/cgi.cfg
/usr/bin/install -c -b -m 660 -o nagios -g nagios sample-config/resource.cfg /usr/local/nagios/etc/resource.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/templates.cfg /usr/local/nagios/etc/objects/templates.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/commands.cfg /usr/local/nagios/etc/objects/commands.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/contacts.cfg /usr/local/nagios/etc/objects/contacts.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/timeperiods.cfg /usr/local/nagios/etc/objects/timeperiods.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/localhost.cfg /usr/local/nagios/etc/objects/localhost.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/windows.cfg /usr/local/nagios/etc/objects/windows.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/printer.cfg /usr/local/nagios/etc/objects/printer.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/switch.cfg /usr/local/nagios/etc/objects/switch.cfg
```

```
*** Config files installed ***

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

/usr/bin/install -c -m 775 -o nagios -g nagcmd -d /usr/local/nagios/var/rw
chmod g+s /usr/local/nagios/var/rw

*** External command directory configured ***
```

Step 18: We need to update the email linked with this server to our email for it to send notifications (if any needed).
sudo nano /usr/local/nagios/etc/objects/contacts.cfg

```
ec2-user@ip-172-31-39-94:~/
  GNU nano 5.8                                  /usr/local/nagios/etc/objects/contacts.cfg
s##############################################################################
# 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)
# 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 above)
    alias                   Nagios Admin            ; Full name of user
    email                   2022.bhushan.malpani@ves.ac.in ; <****** CHANGE THIS TO YOUR EMAIL ADDRESS ******
}
```

Here, change the email under ‘define contact{}’ to your email address.
To save this use the following shortcut sequence CTRL+O→Enter→CTRL+X.
CTRL+O: Overwrite the existing file with edited file
CTRL+X: Exit nano editor.

Step 19: We need to install the necessary configuration files for the Nagios web interface. Run the following command.
sudo make install-webconf

```
[ec2-user@ip-172-31-39-94 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 ***
```

Step 20: Now we need to setup a user to access this nagios web interface. So we run this command to create a user called ‘nagiosadmin’.

Keep this username and password saved as it is needed to login to the web interface.
sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin

```
[ec2-user@ip-172-31-39-94 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-172-31-39-94 nagios-4.5.5]$
```

Step 21: Restart the apache server to apply all the recent configurations.
sudo service httpd restart

```
[ec2-user@ip-172-31-39-94 nagios-4.5.5]$ sudo service httpd restart
Redirecting to /bin/systemctl restart httpd.service
[ec2-user@ip-172-31-39-94 nagios-4.5.5]$
```

Step 22: Now we go back to the downloads folder and extract the files of nagios plugin.
cd ~/downloads
tar zxvf nagios-plugins-2.4.11.tar.gz (Version may vary)

```
[ec2-user@ip-172-31-39-94 downloads]$ tar zxvf nagios-plugins-2.4.11.tar.gz
nagios-plugins-2.4.11/
nagios-plugins-2.4.11/build-aux/
nagios-plugins-2.4.11/build-aux/compile
nagios-plugins-2.4.11/build-aux/config.guess
nagios-plugins-2.4.11/build-aux/config.rpath
nagios-plugins-2.4.11/build-aux/config.sub
nagios-plugins-2.4.11/build-aux/install-sh
nagios-plugins-2.4.11/build-aux/ltmain.sh
nagios-plugins-2.4.11/build-aux/missing
nagios-plugins-2.4.11/build-aux/mkinstalldirs
nagios-plugins-2.4.11/build-aux/depcomp
nagios-plugins-2.4.11/build-aux/snippet/
nagios-plugins-2.4.11/build-aux/snippet/_Noreturn.h
nagios-plugins-2.4.11/build-aux/snippet/arg-nonnull.h
nagios-plugins-2.4.11/build-aux/snippet/c++defs.h
nagios-plugins-2.4.11/build-aux/snippet/warn-on-use.h
nagios-plugins-2.4.11/build-aux/test-driver
nagios-plugins-2.4.11/config_test/
nagios-plugins-2.4.11/config_test/Makefile
nagios-plugins-2.4.11/config_test/run_tests
```

```
nagios-plugins-2.4.11/plugins-root/t/check_dhcp.t
nagios-plugins-2.4.11/plugins-root/t/check_icmp.t
nagios-plugins-2.4.11/po/
nagios-plugins-2.4.11/po/Makefile.in.in
nagios-plugins-2.4.11/po/remove-potcdate.sin
nagios-plugins-2.4.11/po/Makevars
nagios-plugins-2.4.11/po/POTFILES.in
nagios-plugins-2.4.11/po/fr.po
nagios-plugins-2.4.11/po/de.po
nagios-plugins-2.4.11/po/fr.gmo
nagios-plugins-2.4.11/po/de.gmo
nagios-plugins-2.4.11/po/nagios-plugins.pot
nagios-plugins-2.4.11/po/stamp-po
nagios-plugins-2.4.11/po/ChangeLog
nagios-plugins-2.4.11/po/LINGUAS
nagios-plugins-2.4.11/release
[ec2-user@ip-172-31-39-94 downloads]$
```

Step 23: Again, we need to install the configurations for these files.
cd nagios-plugins-2.4.11 (version may vary)
./configure --with-nagios-user=nagios --with-nagios-group=nagios

```
[ec2-user@ip-172-31-39-94 ~]$ cd nagios-plugins-2.4.11
[ec2-user@ip-172-31-39-94 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 that generated files are newer than configure... done
configure: creating ./config.status
config.status: creating gl/Makefile
config.status: creating nagios-plugins.spec
config.status: creating tools/build_perl_modules
config.status: creating Makefile
config.status: creating tap/Makefile
config.status: creating lib/Makefile
config.status: creating plugins/Makefile
config.status: creating lib/tests/Makefile
config.status: creating plugins-root/Makefile
config.status: creating plugins-scripts/Makefile
config.status: creating plugins-scripts/utils.pm
config.status: creating plugins-scripts/utils.sh
config.status: creating perlmods/Makefile
config.status: creating test.pl
config.status: creating pkg/solaris/pkginfo
config.status: creating po/Makefile.in
config.status: creating config.h
config.status: config.h is unchanged
config.status: executing depfiles commands
config.status: executing libtool commands
config.status: executing po-directories commands
config.status: creating po/POTFILES
config.status: creating po/Makefile
[ec2-user@ip-172-31-39-94 nagios-plugins-2.4.11]$
```

Step 24: We need to compile all components of this software according to the instruction in the Makefile. For that, use the commands:
make
sudo make install

```
fi
make[1]: Leaving directory '/home/ec2-user/nagios-plugins-2.4.11/po'
make[1]: Entering directory '/home/ec2-user/nagios-plugins-2.4.11'
make[2]: Entering directory '/home/ec2-user/nagios-plugins-2.4.11'
make[2]: Nothing to be done for 'install-exec-am'.
make[2]: Nothing to be done for 'install-data-am'.
make[2]: Leaving directory '/home/ec2-user/nagios-plugins-2.4.11'
make[1]: Leaving directory '/home/ec2-user/nagios-plugins-2.4.11'
[ec2-user@ip-172-31-39-94 nagios-plugins-2.4.11]$
```

Step 25: We need to register the Nagios service with the system, which would make it able to manage the server status. So run the following commands
sudo chkconfig --add nagios
sudo chkconfig nagios on

Step 26: We need to verify the Nagios configuration for any syntax errors or issues before starting or restarting the Nagios service.
sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

```
[ec2-user@ip-172-31-39-94 nagios-4.5.5]$ sudo /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...
Error: Unexpected token or statement in file '/usr/local/nagios/etc/objects/contacts.cfg' on line 1.
   Error processing object config files!


***> One or more problems was encountered while processing the config files...

     Check your configuration file(s) to ensure that they contain valid
     directives and data definitions.  If you are upgrading from a previous
     version of Nagios, you should be aware that some variables/definitions
     may have been removed or modified in this version.  Make sure to read
     the HTML documentation regarding the config files, as well as the
     'Whats New' section to find out what has changed.
```

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)
It is an error in processing main config file!

Solution: Create the missing directory, set the permissions, verify it.

- sudo mkdir -p /usr/local/nagios/var/spool/checkresults (Create)
- sudo chown nagios:nagios /usr/local/nagios/var/spool/checkresults
- sudo chmod 775 /usr/local/nagios/var/spool/checkresults (permissions)

Now rerun the commmad
sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

```
[ec2-user@ip-172-31-39-94 nagios-4.5.5]$ sudo /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 8 services.
        Checked 1 hosts.
        Checked 1 host groups.
        Checked 0 service groups.
        Checked 1 contacts.
        Checked 1 contact groups.
        Checked 24 commands.
        Checked 5 time periods.
        Checked 0 host escalations.
        Checked 0 service escalations.
Checking for circular paths...
        Checked 1 hosts
        Checked 0 service dependencies
        Checked 0 host dependencies
        Checked 5 timeperiods
Checking global event handlers...
Checking obsessive compulsive processor commands...
Checking misc settings...

Total Warnings: 0
Total Errors:   0

Things look okay - No serious problems were detected during the pre-flight check
[ec2-user@ip-172-31-39-94 nagios-4.5.5]$
```

Step 27:

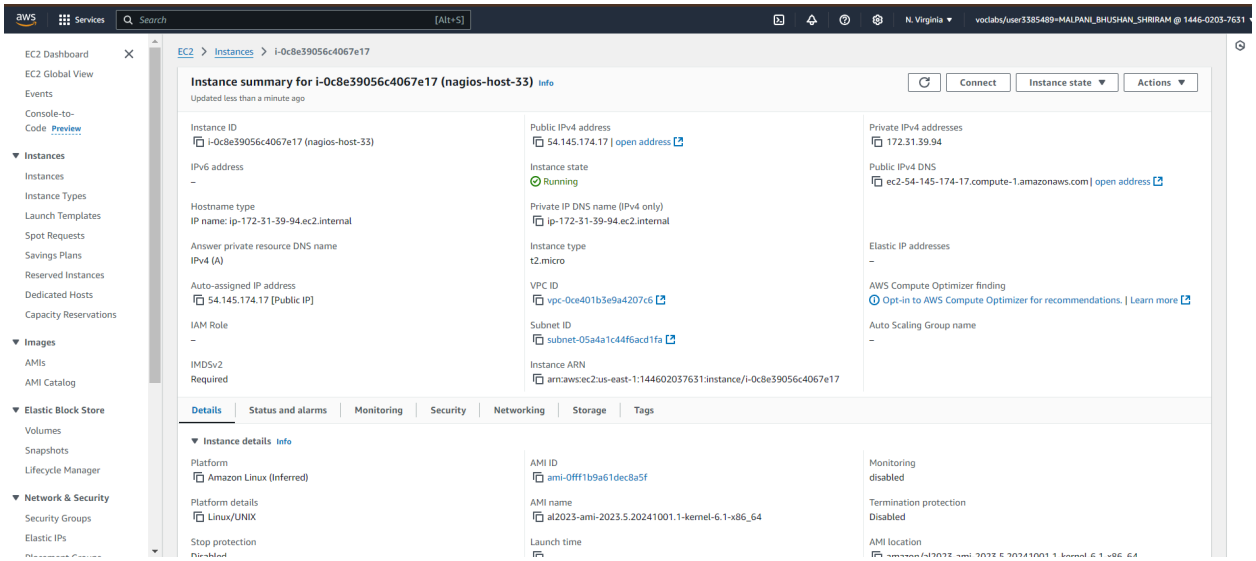
sudo service nagios start

```
[ec2-user@ip-172-31-39-94 nagios-4.5.5]$ sudo service nagios start
Redirecting to /bin/systemctl start nagios.service
```

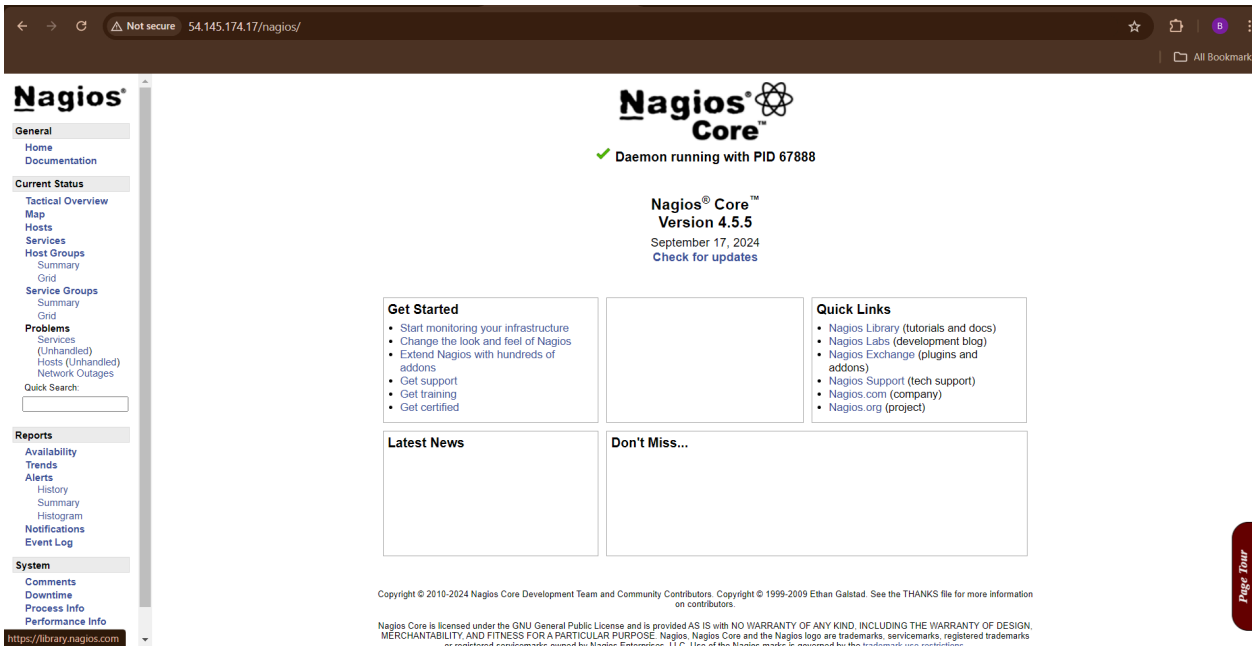
sudo systemctl status nagios

```
● nagios.service - Nagios Core 4.5.5
     Loaded: loaded (/usr/lib/systemd/system/nagios.service; enabled; preset: disabled)
     Active: active (running) since Sun 2024-10-06 10:41:35 UTC; 16s ago
       Docs: https://www.nagios.org/documentation
    Process: 67265 ExecStartPre=/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
    Process: 67272 ExecStart=/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
   Main PID: 67273 (nagios)
      Tasks: 6 (limit: 1112)
     Memory: 5.7M
        CPU: 65ms
     CGroup: /system.slice/nagios.service
             ├─67273 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─67275 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─67276 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─67277 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─67278 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             └─67318 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg

Oct 06 10:41:35 ip-172-31-39-94.ec2.internal nagios[67273]: qh: Socket '/usr/local/nagios/var/rw/nagios.qh' successfully initialized
Oct 06 10:41:35 ip-172-31-39-94.ec2.internal nagios[67273]: qh: core query handler registered
Oct 06 10:41:35 ip-172-31-39-94.ec2.internal nagios[67273]: qh: echo service query handler registered
Oct 06 10:41:35 ip-172-31-39-94.ec2.internal nagios[67273]: qh: help for the query handler registered
Oct 06 10:41:35 ip-172-31-39-94.ec2.internal nagios[67273]: wproc: Successfully registered manager as @wproc with query handler
Oct 06 10:41:35 ip-172-31-39-94.ec2.internal nagios[67273]: wproc: Registry request: name=Core Worker 67277;pid=67277
Oct 06 10:41:35 ip-172-31-39-94.ec2.internal nagios[67273]: wproc: Registry request: name=Core Worker 67278;pid=67278
Oct 06 10:41:35 ip-172-31-39-94.ec2.internal nagios[67273]: wproc: Registry request: name=Core Worker 67275;pid=67275
Oct 06 10:41:35 ip-172-31-39-94.ec2.internal nagios[67273]: wproc: Registry request: name=Core Worker 67276;pid=67276
Oct 06 10:41:35 ip-172-31-39-94.ec2.internal nagios[67273]: Successfully launched command file worker with pid 67318
```

Step 28: 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.



Step 29: Open a new tab. In the address bar type http://<publicipaddress>/nagios.
This would be in the output



Conclusion:

In this experiment, we have learned how to install and configure Nagios Core, Nagios Plugins and NRPE on a Linux machine. We are using an Amazon Linux OS instance configured with the need security rules. We need to make sure that the Nagios-core and Nagios-plugins links that are used are the ones which are up-to dat (wget commands).It is needed to extract and configure these files so that no issues are detected while starting the server. Once all the setup

is complete, we can start the nagios server. Using the public IP address of the EC2 instance, we can access the nagios dashboard by navigating that IP to nagios.

NAME: Bhushan Malpani DIV: D15C ROLL NO: 33

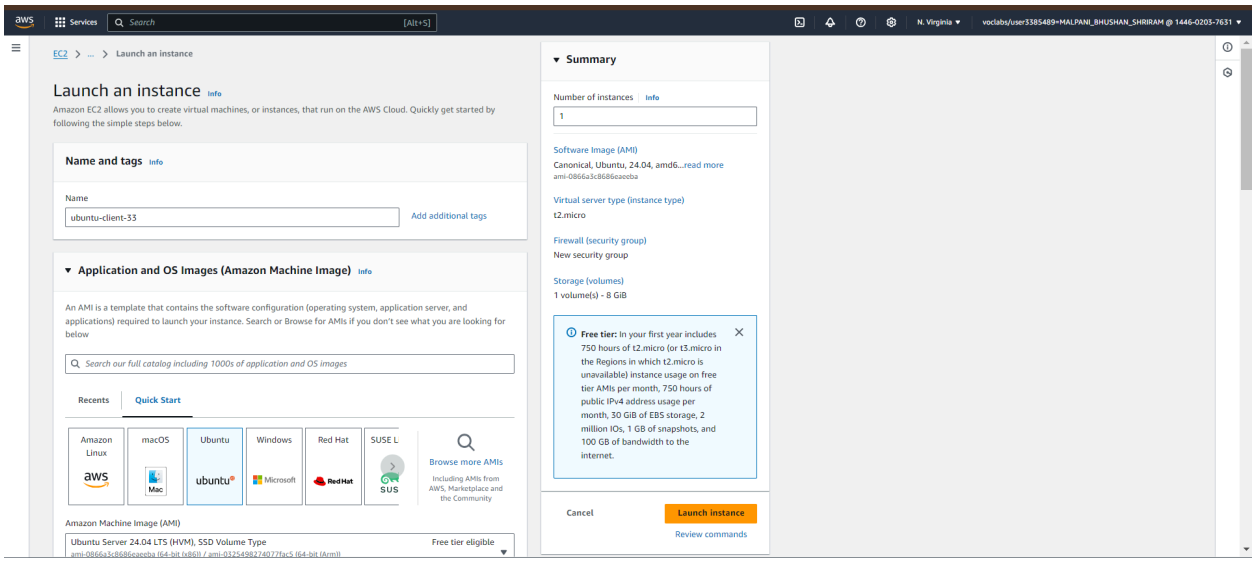
Aim: To perform Port, Service monitoring, Windows/Linux server monitoring using Nagios.

Prerequisites:

1) An Amazon Linux instance with nagios already set up.

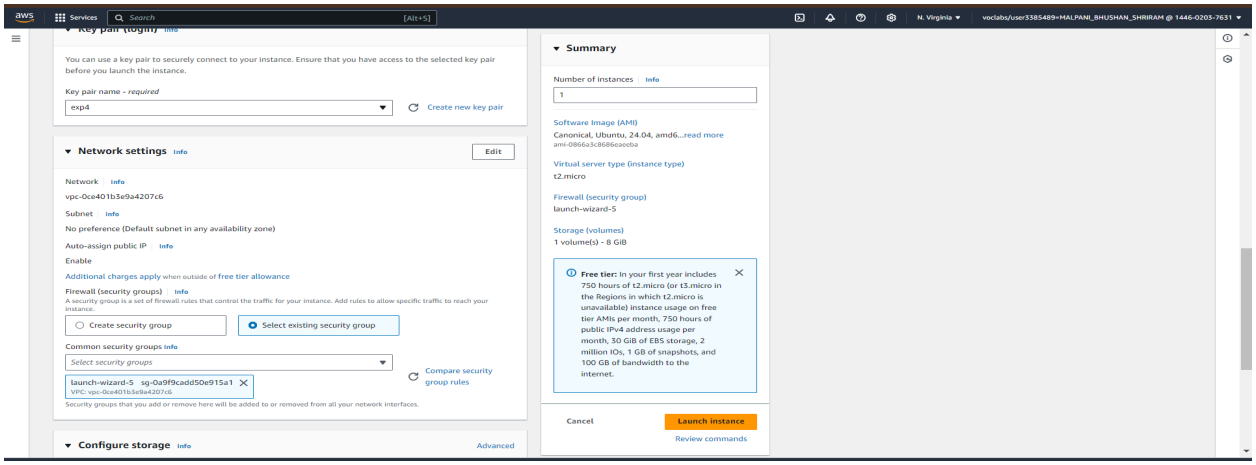
Step 1: Set up ubuntu instance

1) Login to your AWS account. Search for EC2 on services. Open the interface and click on Create Instance.



Select The OS Image as Ubuntu.

2) Make sure to select the same private key that you created for the Amazon Linux instance. Also select the same security group as you created for the Linux instance.



NAME: Bhushan Malpani DIV: D15C ROLL NO: 33

3) Now come back to the instances screen. Click on the instance ID of your instance. Then click on Connect. Click on SSH client. Copy the example command. Now, we have to connect our local OS terminal to the instance using SSH. For this, open terminal wher the private key file is located (.pem). Paste the copied SSH command and run it.

Step 2: Execute the following on Nagios Host machine (Linux)

1) We need to verify whether the nagios service is running or not. Fo that, run this command.
   ps -ef | grep nagios

```
ubuntu@ip-172-31-36-126:~$ ps -ef | grep nagios
ubuntu      1151    1139  0 11:12 pts/0    00:00:00 grep --color=auto nagios
```

2) Now, make yourself as the root user, and create a folder with the path '/usr/local/nagios/etc/objects/monitorhosts/linuxhosts'
   sudo su
   mkdir -p /usr/local/nagios/etc/objects/monitorhosts/linuxhosts

```
ubuntu@ip-172-31-36-126:~$ sudo su
mkdir -p /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
root@ip-172-31-36-126:/home/ubuntu#
```

3) We need to create a config file in this folder. So, copy the contents of the existing localhost config to the new file 'linuxserver.cfg'.
   cp /usr/local/nagios/etc/objects/localhost.cfg
   /usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg

4) We need to make some changes in this config file. Open it using nano editor.
   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)

```
root@ip-172-31-36-126: /hom
  GNU nano 7.2              /usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg *
define host {
    use                     linux-server      ; Name of host template
    host_name               linuxserver       ; The name of the host
    alias                   linuxserver       ; A short name for the host
    address                 54.166.101.67     ; Public IP address of your Linux server
}
```

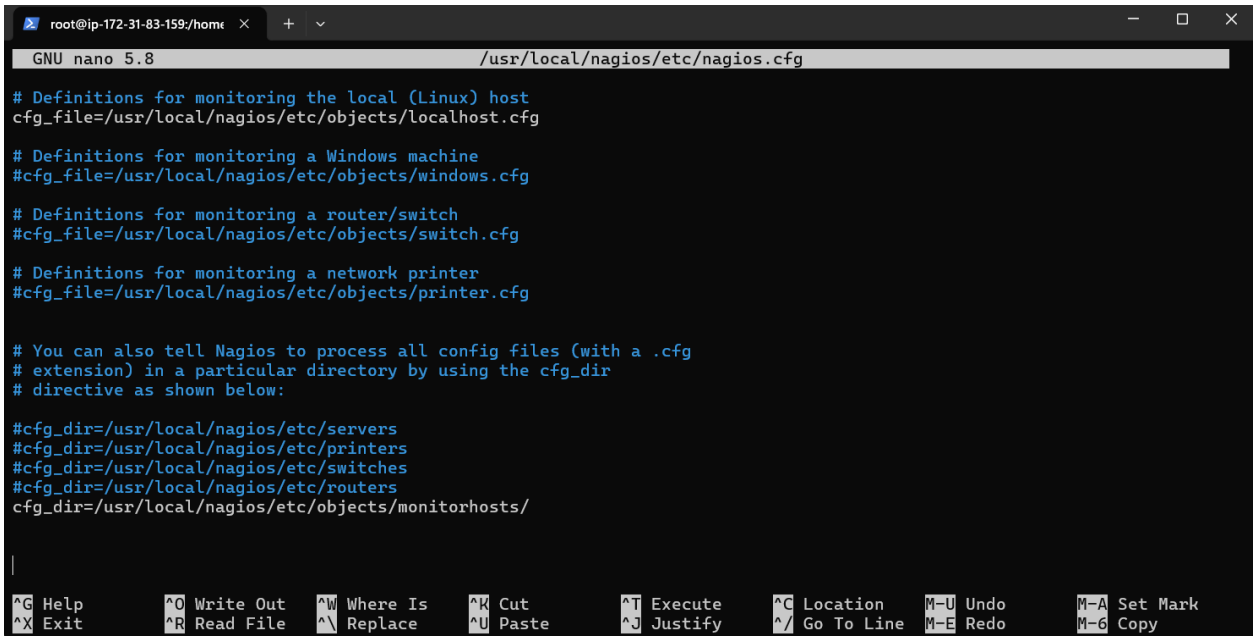
Change hostgroup_name to **linux-servers1**

NAME: Bhushan Malpani DIV: D15C ROLL NO: 33

```
define hostgroup{
        hostgroup_name      linux-server1 ; The name of the hostgroup
        alias               linux Servers ; Long name of the group
        members             localhost     ; Comma separated list of hosts t>
}
```

Change the **occurrences of hostname** further in the document from **localhost** to **linuxserver**

5) Now, we need to edit the nagios configuration file to add this directory.
   nano /usr/local/nagios/etc/nagios.cfg
   Run this command and add the following line
   cfg_dir=/usr/local/nagios/etc/objects/monitorhosts/



6) Now we verify the configuration files.
   /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

NAME: Bhushan Malpani DIV: D15C ROLL NO: 33

```
[root@ip-172-31-83-157 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 16 services.
        Checked 2 hosts.
        Checked 2 host groups.
        Checked 0 service groups.
        Checked 1 contacts.
        Checked 1 contact groups.
        Checked 24 commands.
        Checked 5 time periods.
        Checked 0 host escalations.
        Checked 0 service escalations.
Checking for circular paths...
        Checked 2 hosts
        Checked 0 service dependencies
        Checked 0 host dependencies
        Checked 5 timeperiods
Checking global event handlers...
Checking obsessive compulsive processor commands...
Checking misc settings...

Total Warnings: 0
Total Errors:   0

Things look okay - No serious problems were detected during the pre-flight check
[root@ip-172-31-83-157 ec2-user]#
```

7) Once the files are verified, we need to restart the server.
   service nagios restart

```
[root@ip-172-31-83-159 nagios-plugins-2.0.3]# service nagios restart
Restarting nagios (via systemctl):                         [  OK  ]
[root@ip-172-31-83-159 nagios-plugins-2.0.3]#
```

Step 3: Execute the following on Nagios Client machine (Ubuntu)

NAME: Bhushan Malpani DIV: D15C ROLL NO: 33

1) First, we check for any new updates, then we install gcc, nagios nrpe server and nagios plugins.
   sudo apt update -y
   sudo apt install gcc -y
   sudo apt install -y nagios-nrpe-server nagios-plugins

```
ubuntu@ip-172-31-81-89:~$ sudo apt update -y
sudo apt install gcc -y
sudo apt install -y nagios-nrpe-server nagios-plugins
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:5 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:7 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [380 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
```

```
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 #4: sshd[1495,1569]
 ubuntu @ user manager service: systemd[1500]

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-81-89:~$
```

2) We need to add the public IP address of our host Nagios machine (Linux) to the nrpe configuration file.

NAME: Bhushan Malpani DIV: D15C ROLL NO: 33

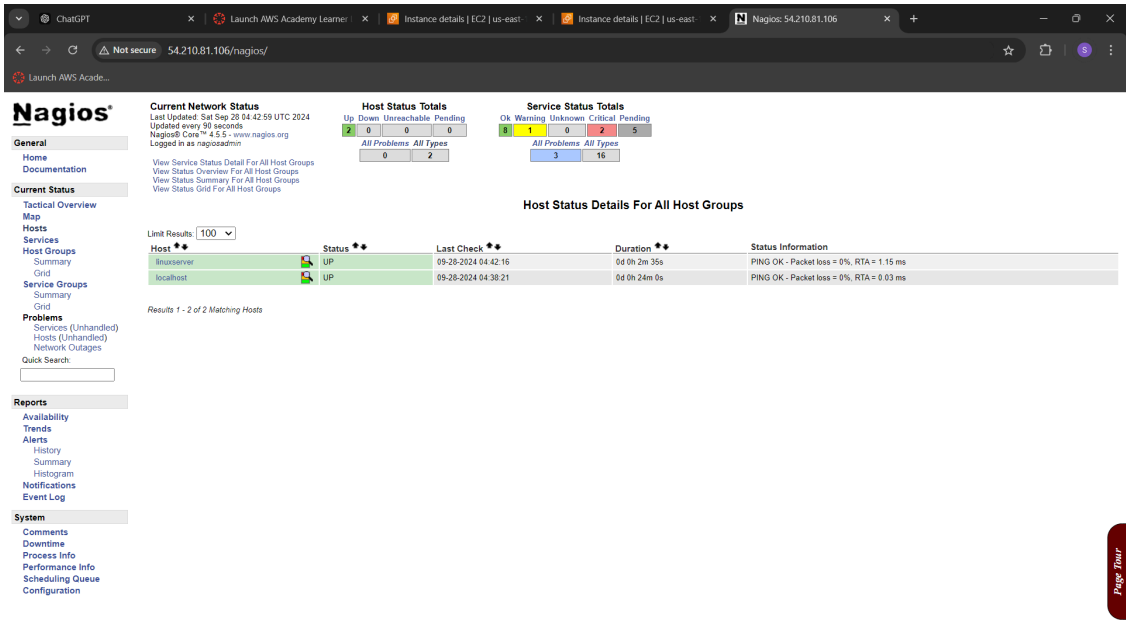
sudo nano /etc/nagios/nrpe.cfg

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

```
  GNU nano 7.2                                    /etc/nagios/nrpe.cfg *
# You can either supply a username or a UID.
#
# NOTE: This option is ignored if NRPE is running under either inetd or xinetd

nrpe_user=nagios



# NRPE GROUP
# This determines the effective group that the NRPE daemon should run as.
# You can either supply a group name or a GID.
#
# NOTE: This option is ignored if NRPE is running under either inetd or xinetd

nrpe_group=nagios



# ALLOWED HOST ADDRESSES
# This is an optional comma-delimited list of IP address or hostnames
# that are allowed to talk to the NRPE daemon. Network addresses with a bit mask
# (i.e. 192.168.1.0/24) are also supported. Hostname wildcards are not currently
# supported.
#
# Note: The daemon only does rudimentary checking of the client's IP
# address.  I would highly recommend adding entries in your /etc/hosts.allow
# file to allow only the specified host to connect to the port
# you are running this daemon on.
#
# NOTE: This option is ignored if NRPE is running under either inetd or xinetd

allowed_hosts=127.0.0.1,54.210.81.106



# COMMAND ARGUMENT PROCESSING
# This option determines whether or not the NRPE daemon will allow clients
# to specify arguments to commands that are executed.  This option only works
# if the daemon was configured with the --enable-command-args configure script
# option.

^G Help          ^O Write Out     ^W Where Is      ^K Cut           ^T Execute       ^C Location      M-U Undo         M-A Set Mark     M-] To Bracket
^X Exit          ^R Read File     ^\ Replace       ^U Paste         ^J Justify       ^/ Go To Line    M-E Redo         M-6 Copy         ^Q Where Was
```

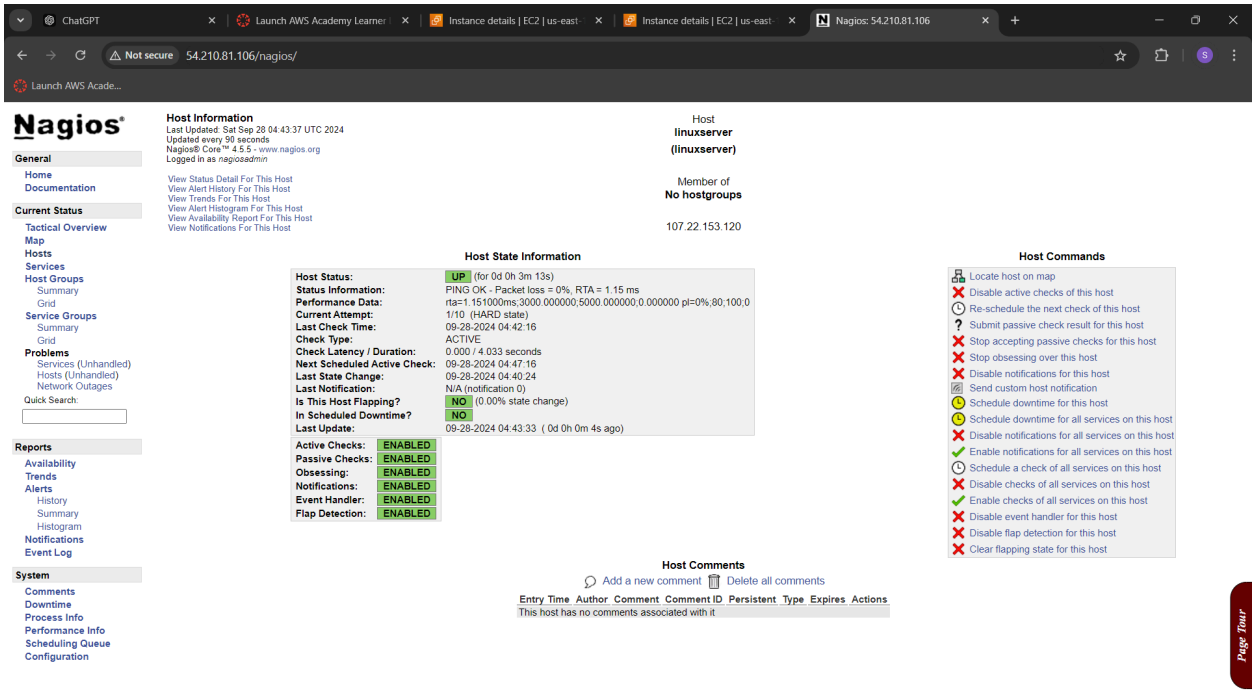
Step 4: Check the Nagios Dashboard

1) Go to Nagios dashboard, click on hosts.
   Here, we can see that the linuxserver is also added as a host.

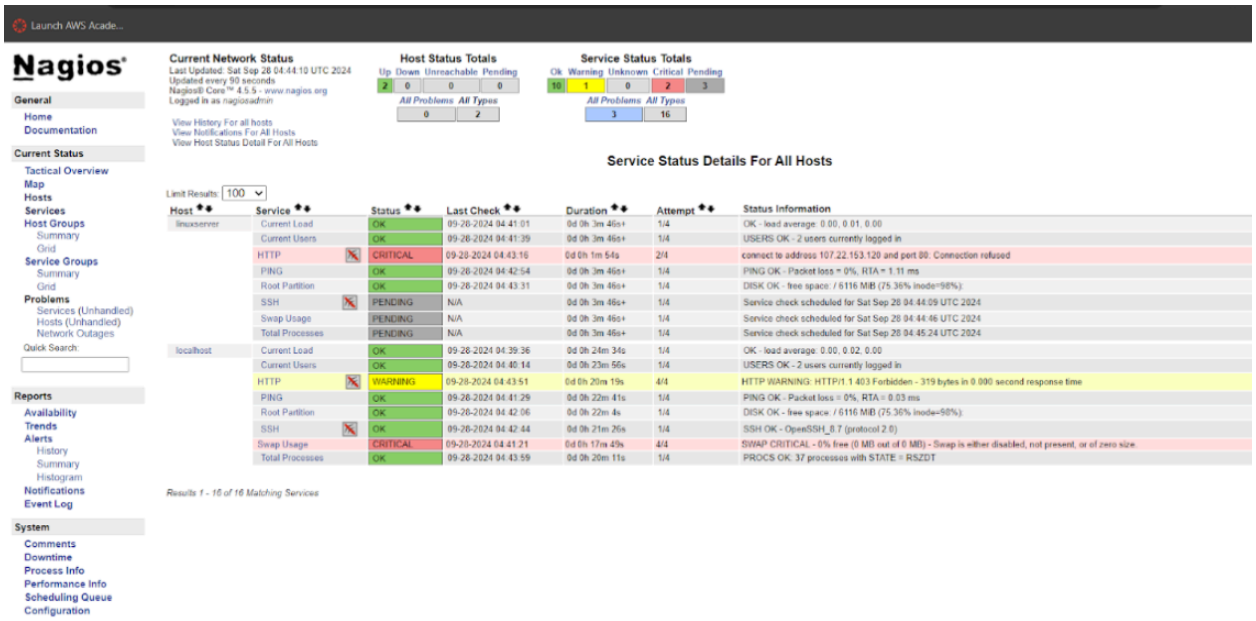


2) Click on linuxserver. Here, we can check all the information about linuxserver host.

NAME: Bhushan Malpani DIV: D15C ROLL NO: 33



3) Click on services. Here we an see all the services that are being monitored by linuxserver.



In this case, we have monitored -
Servers: 1 linux server
Services: swap

NAME: Bhushan Malpani DIV: D15C ROLL NO: 33

Ports: 22, 80 (ssh, http)
Processes: User status, Current load, total processes, root partition, etc.

Conclusion:

In this experiment, we learned to perform port service monitoring and server monitoring using Nagios. For this, we need the Linux instance used to host the Nagios dashboard and server. Also, we would need an Ubuntu instance which would be linked to a second host. We need to set up some configurations on the Linux instance and add the IP address of the Ubuntu instance. After that, we need to make the same initial setup on the ubuntu instance as the linu instance. Add the Ip address of linux instance in allowed hosts. After restarting the NRPE server, we can see the ‘linuxserver’ host added.

NAME: Bhushan Malpani **DIV:D15C** **ROLL NO:33**

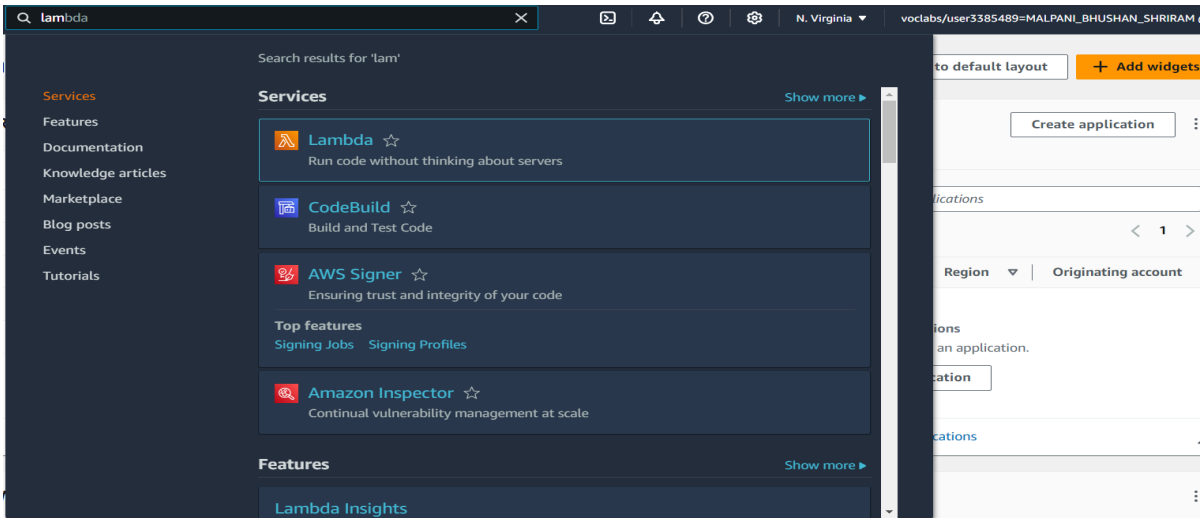
Aim: To understand AWS Lambda, its workflow, various functions and create your first Lambda functions using Python / Java / Nodejs.

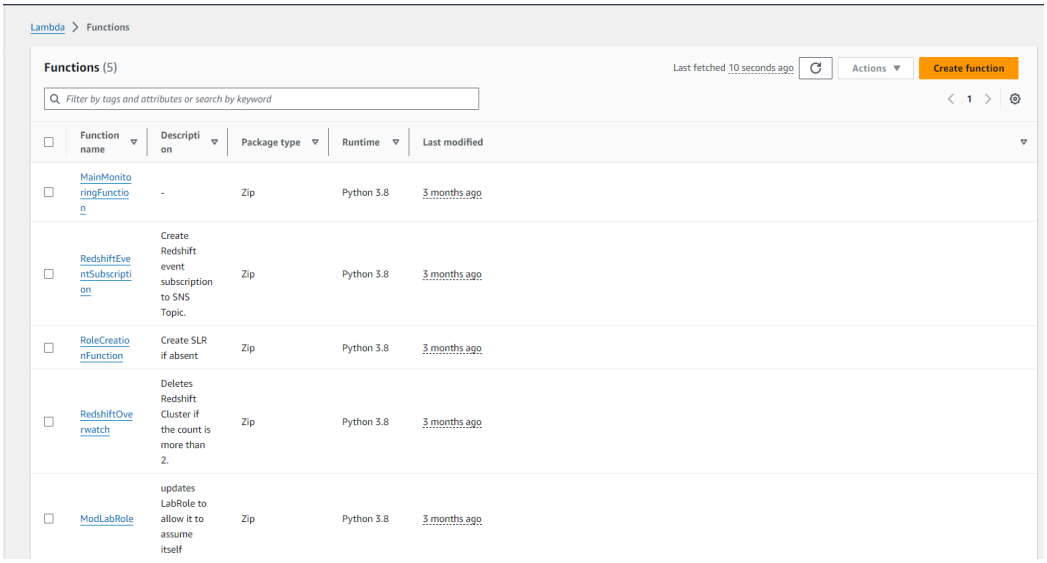
Prerequisites:

1) AWS account (academy recommended)

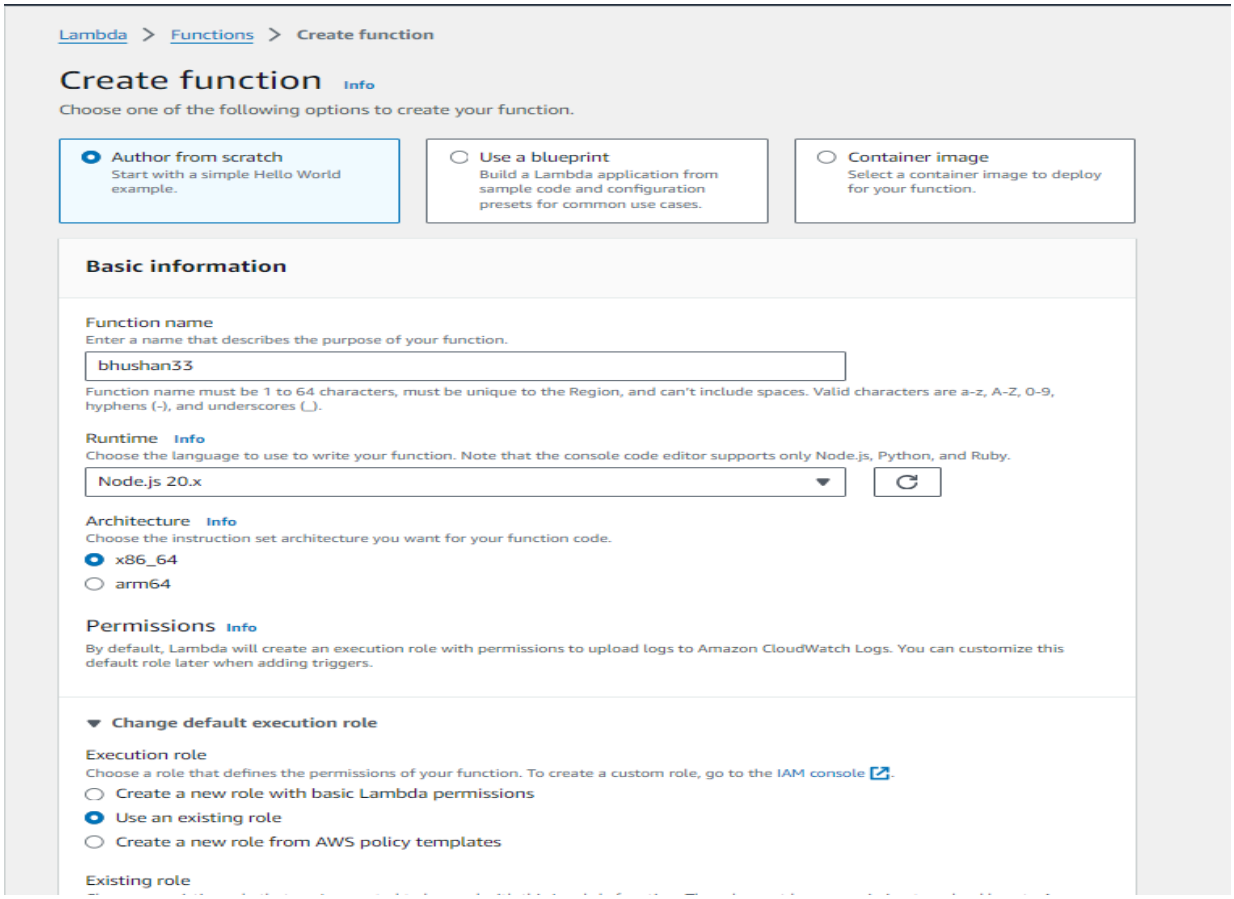
Step 1: Set up AWS Lambda Function

1) Search for Lambda in the services tab. Click on it once found.

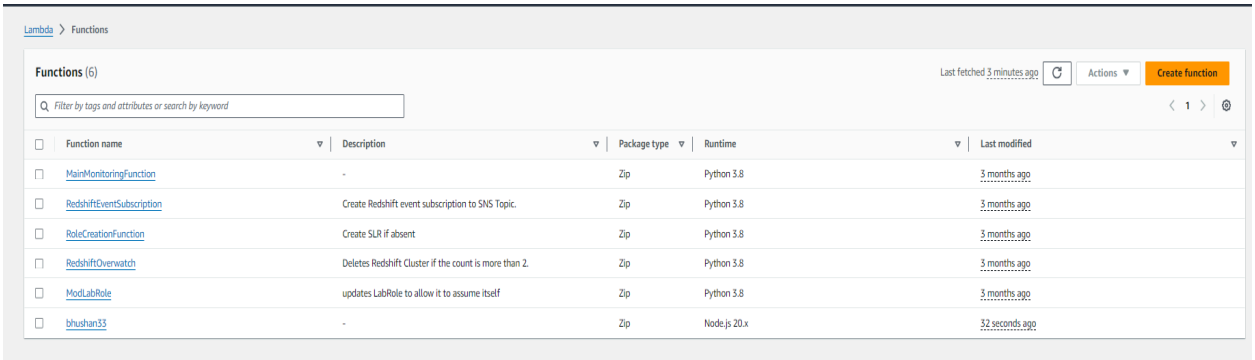




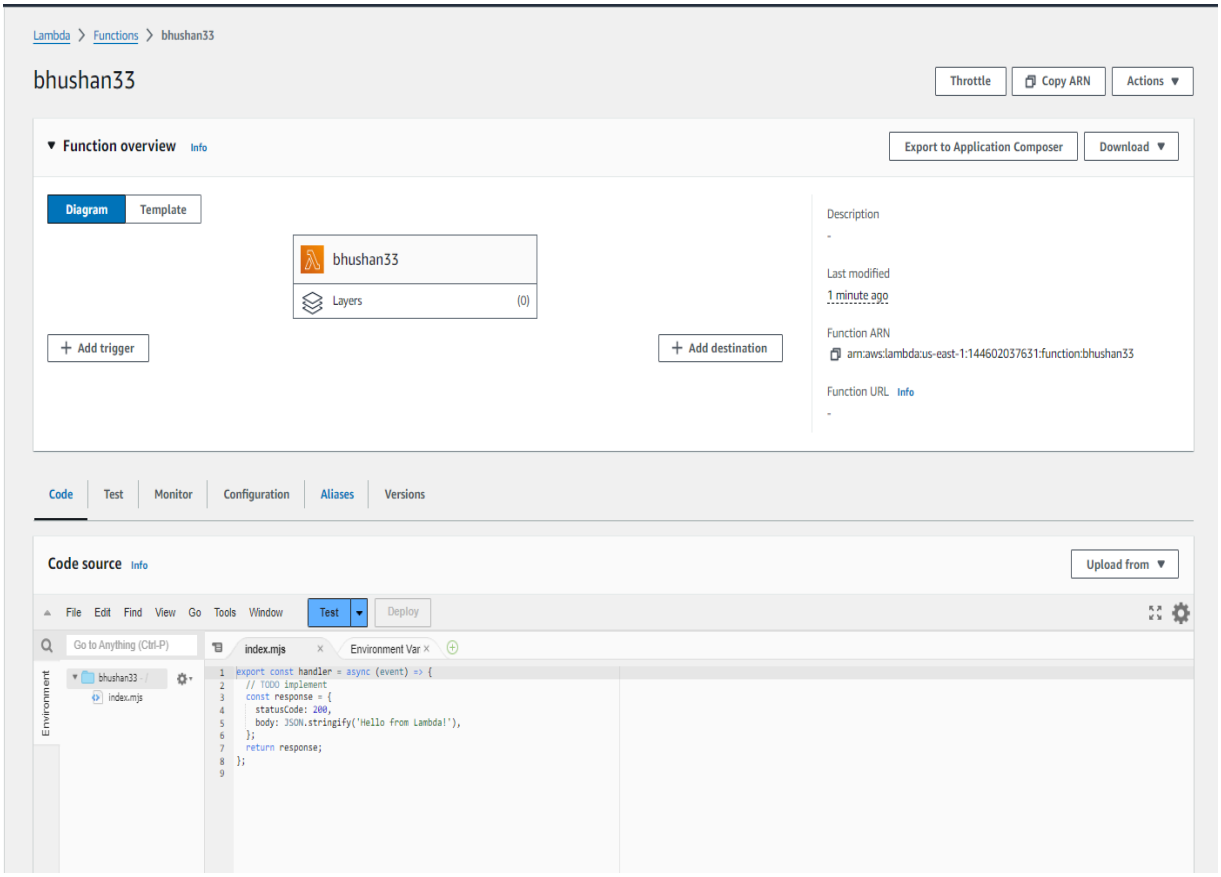
2) Click on create functions.
3) Give a name to your Lambda function. Select the runtime as Node.js 20.x (You can also use python). Select the architecture as x86_64. Set the default execution role as LabRole if you are doing this on your academy account. (Use an existing role → LabRole)



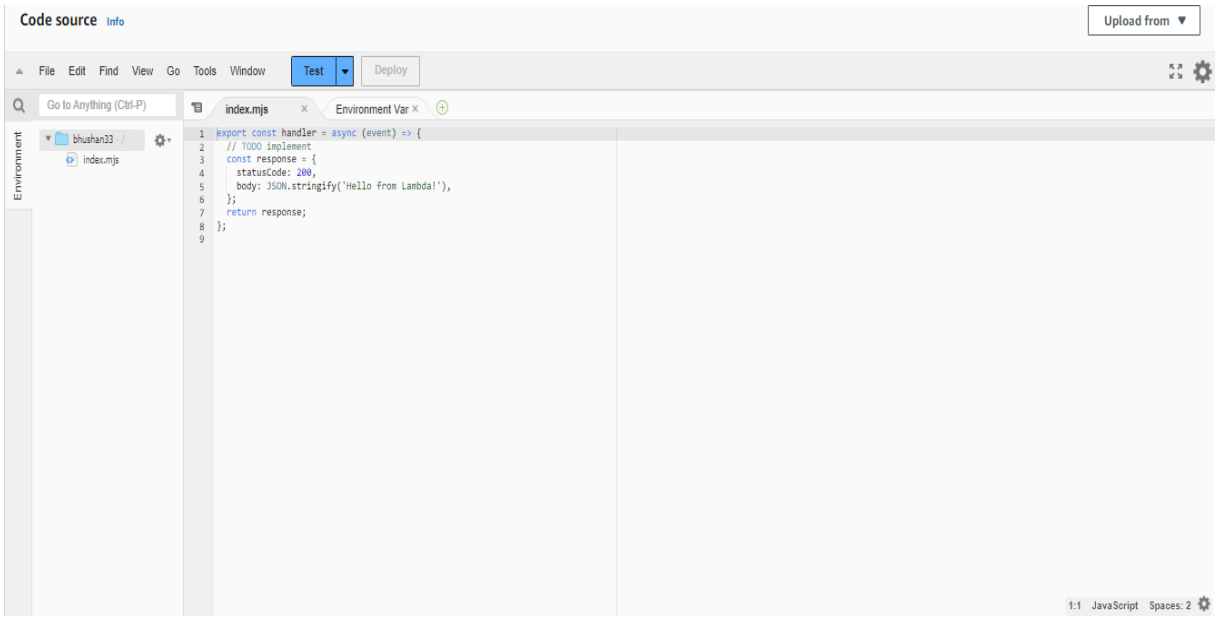
4) Once the function is created, click on the name of the function (myLambda27 in my case).



5) This is the dashboard of our lambda function.

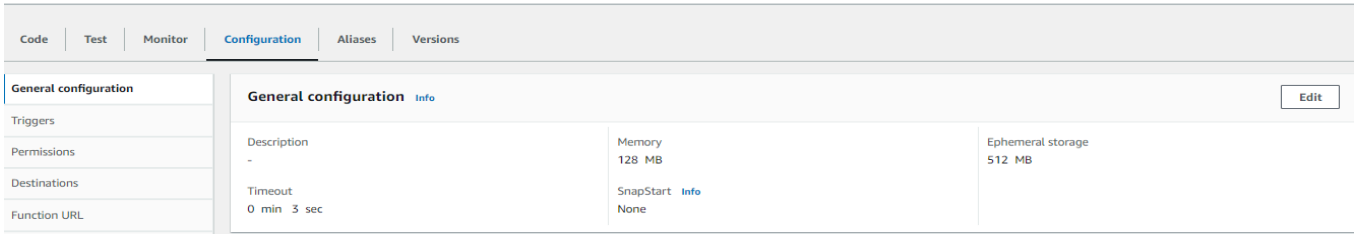


6) This function has the following default code, which is used to print “Hello form Lambda!”.

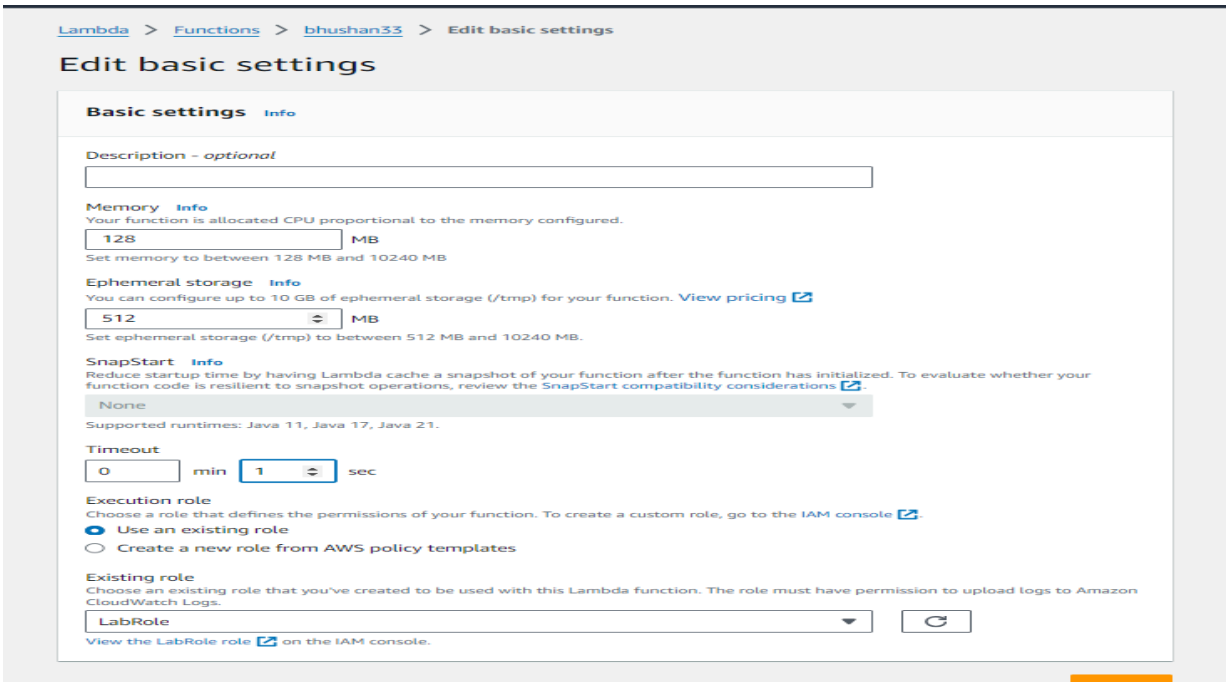


Step 3: Set up configurations and test events

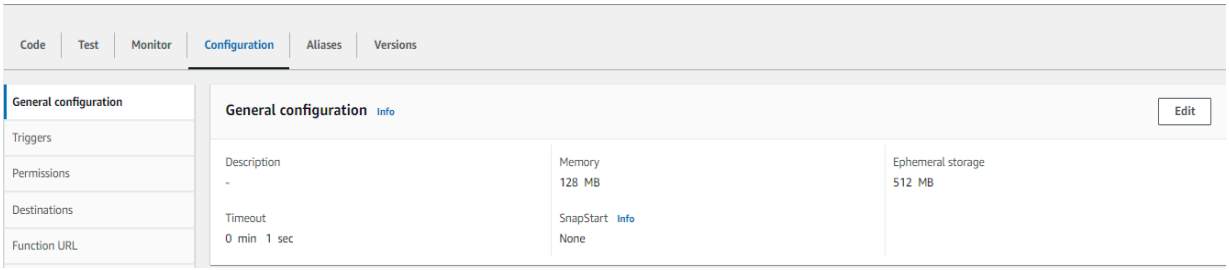
1) Just above the test code, you would find Configuration, click on it. Then click on Edit.



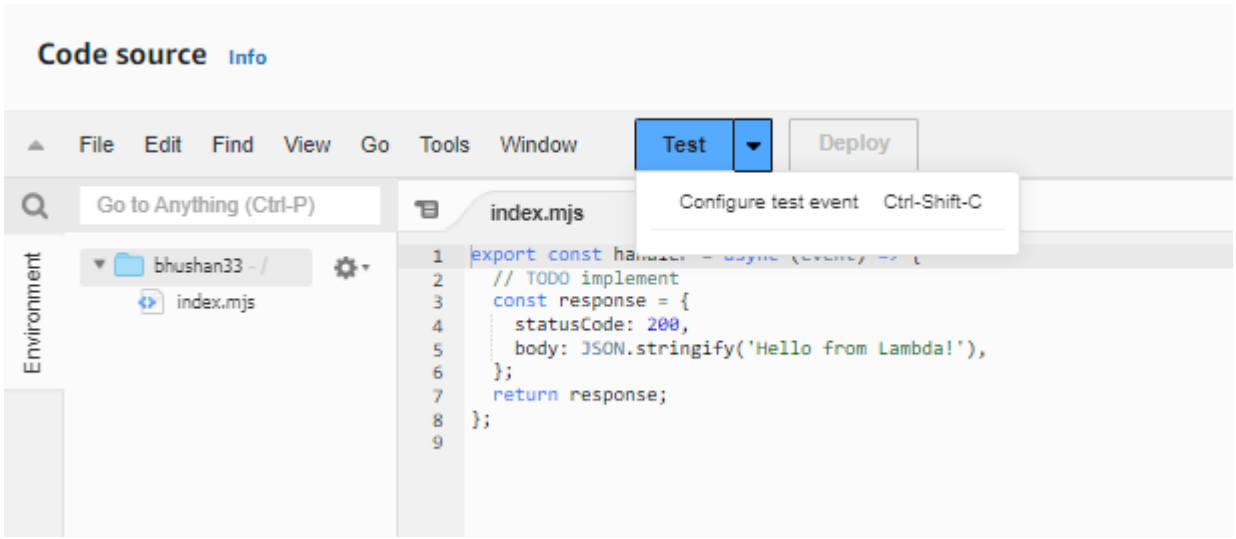
2) Here, change the Timeout to 1 sec. This is the time for which the function can be running before it is forcibly terminated.



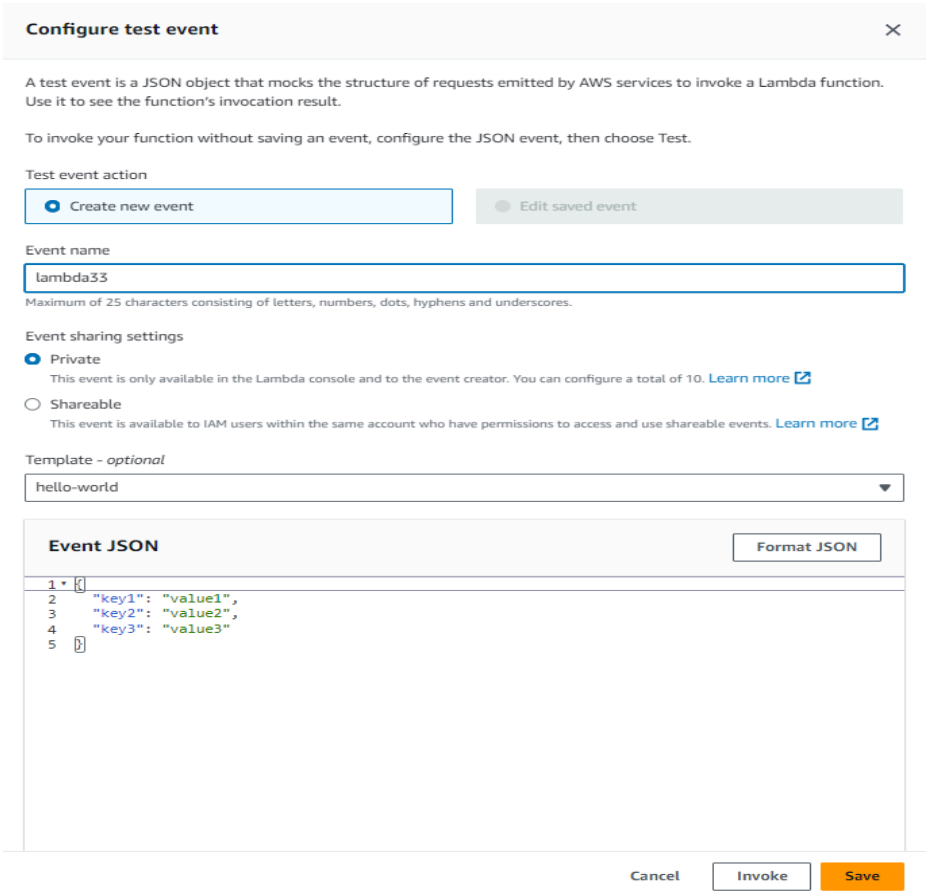
3) We can see the executed changes.



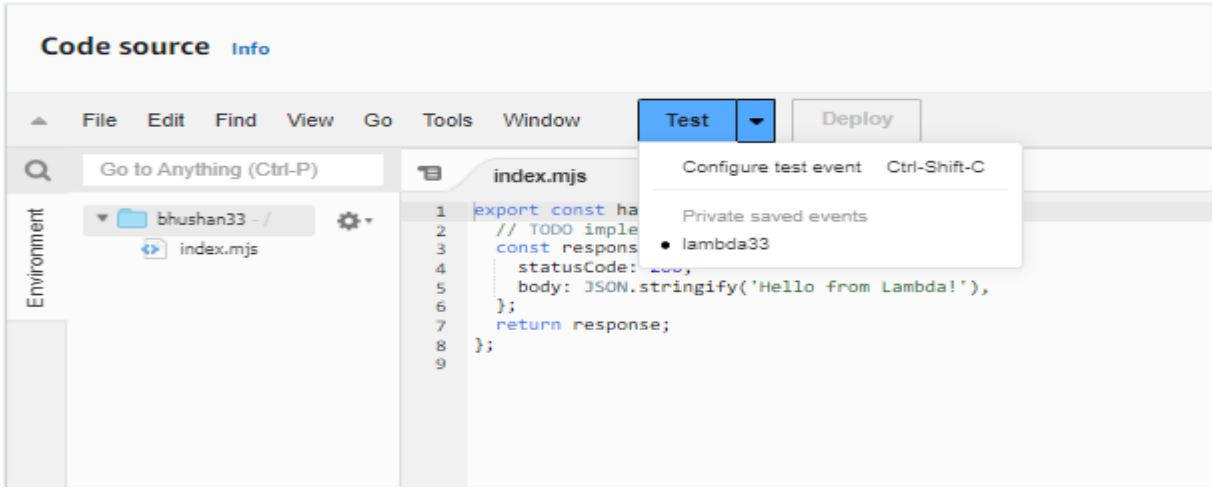
4) Switch back to the code tab. Click on the dropdown arrow near test. Then select configure test event.



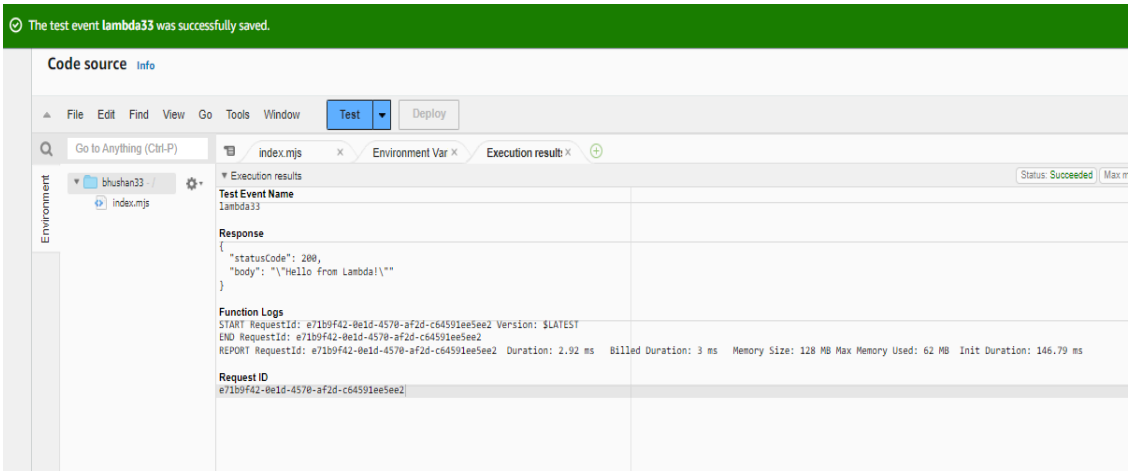
5) Here, create a new event, keep the other options default and save the event.



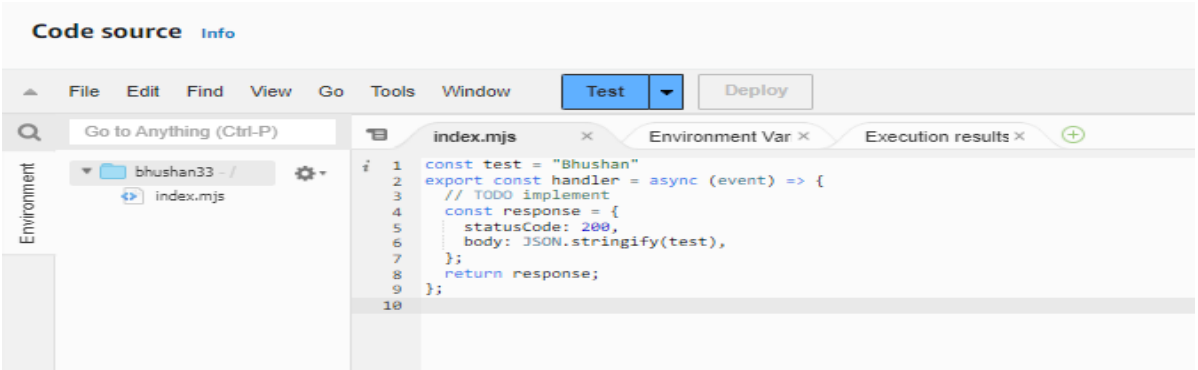
6) Now, again click on the dropdown. This time, select the event you have created. Then, click on TEST.



7) We can see the expected output for the sample code.



8) For a test, declare a string and call it in line 6. After making the changes click on deplpoy.



9) Run the test. We can see that the string we declared has come in the output.

Conclusion:

In this experiment, we effectively investigated the AWS Lambda service by developing and configuring Lambda functions using Python, Java, or Node.js. We discovered how to establish a Lambda function, tweak its settings (like modifying the timeout duration), and evaluate the function with personalized events. Throughout this experience, we noted how Lambda manages executions, including timeout handling and producing expected results according to modifications in the code.

NAME: Bhushan Malpani **DIV: D15C** **ROLL NO:33**

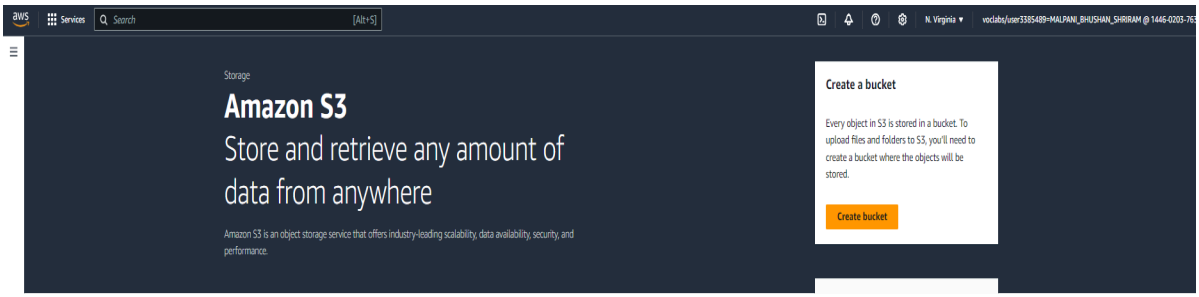
Aim: To create a Lambda function which will log “An Image has been added” once you add an object to a specific bucket in S3.

Prerequisites:

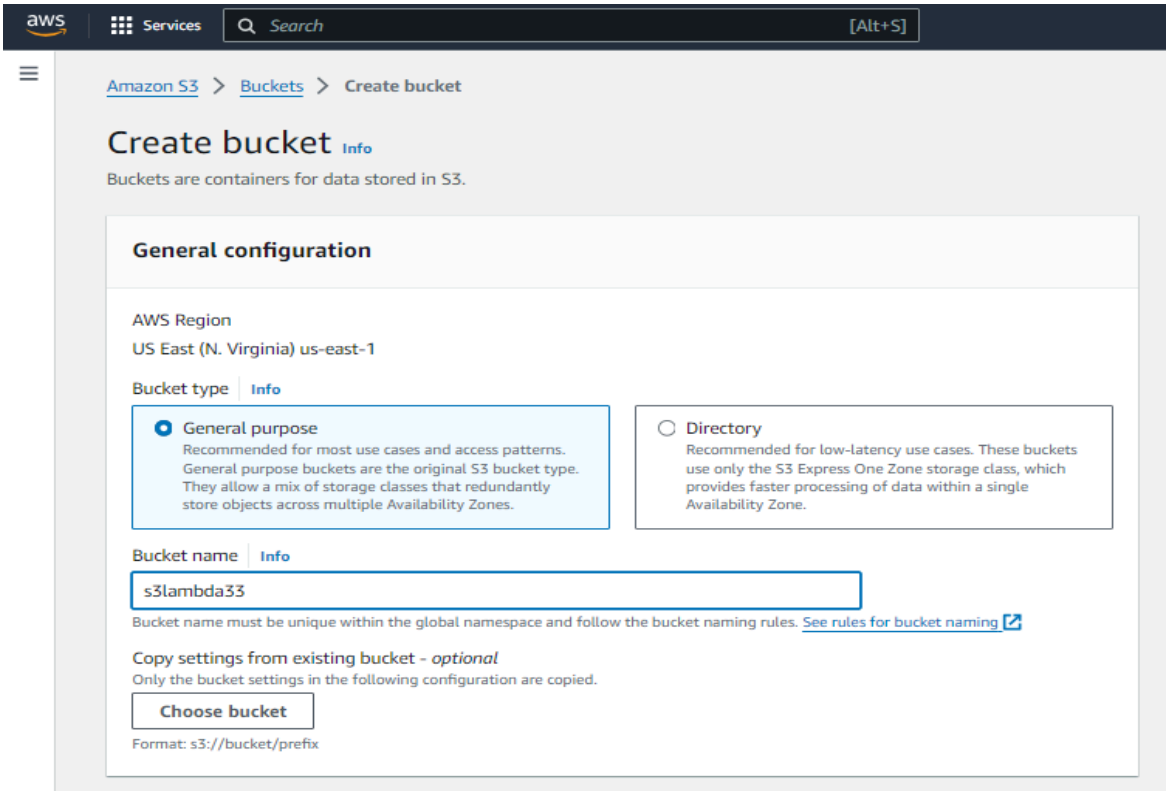
1) AWS account (acacdemy preferable)
2) Lambda function (created in the previous experiment).

Step 1: Create a s3 bucket.

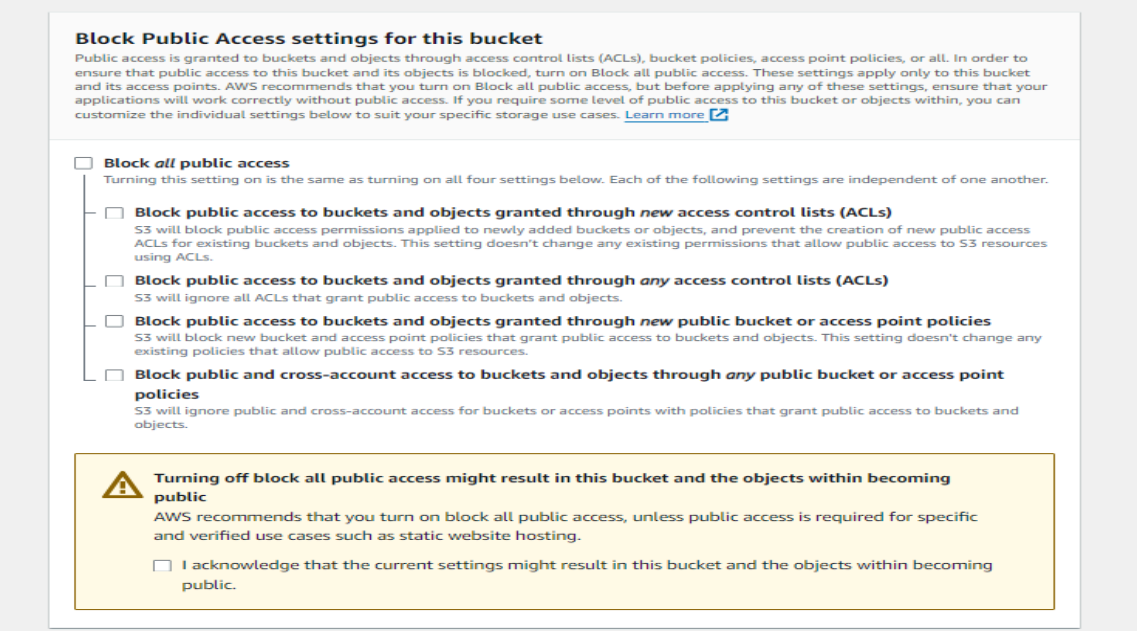
1) Search for S3 bucket in the services search. Then click on create bucket.



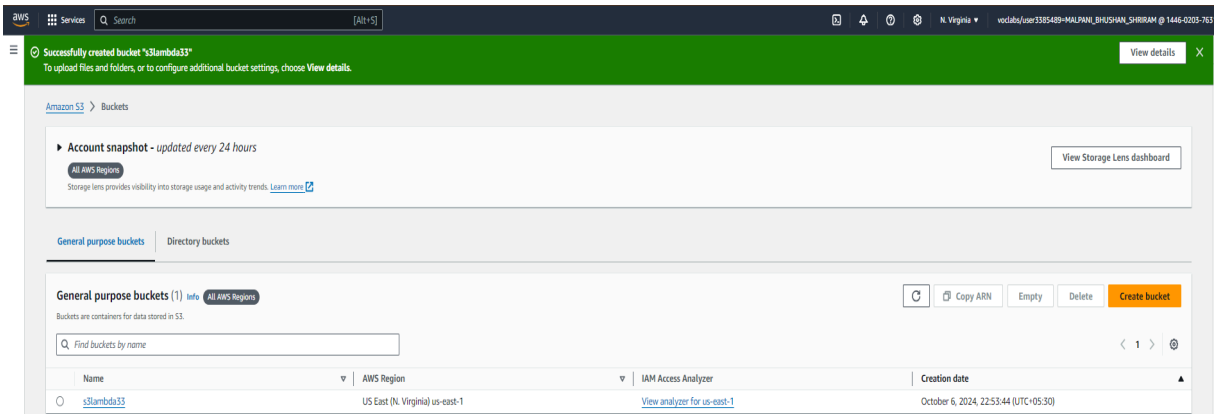
2) Keep the bucket as a general purpose bucket. Give a name to your bucket.



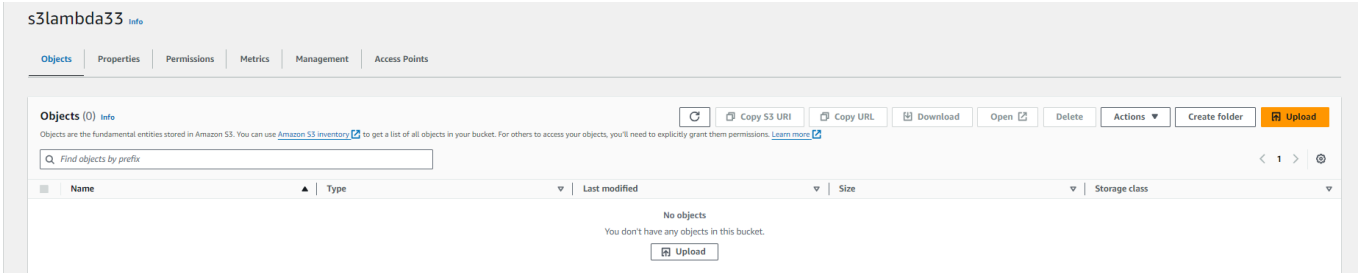
3) Uncheck block all public access.



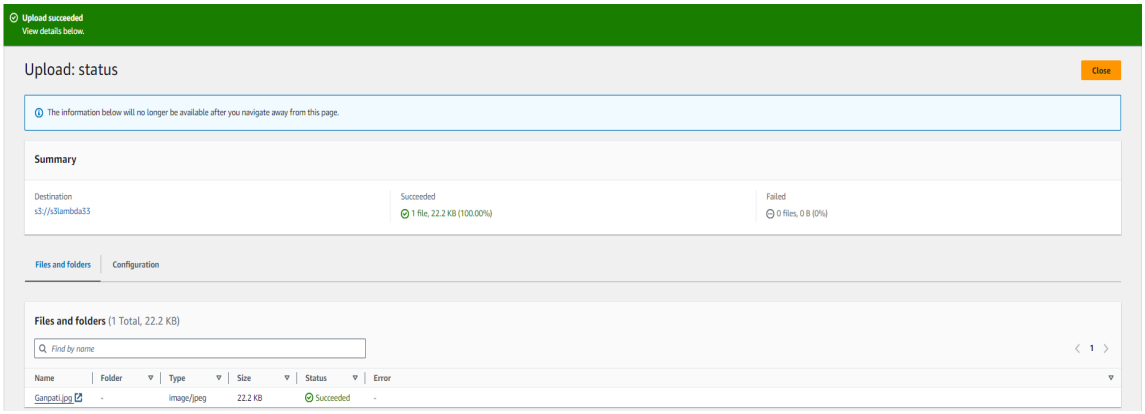
4) Keeping all other options same, click on create. This would create your bucket. Now click on the name of the bucket.



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

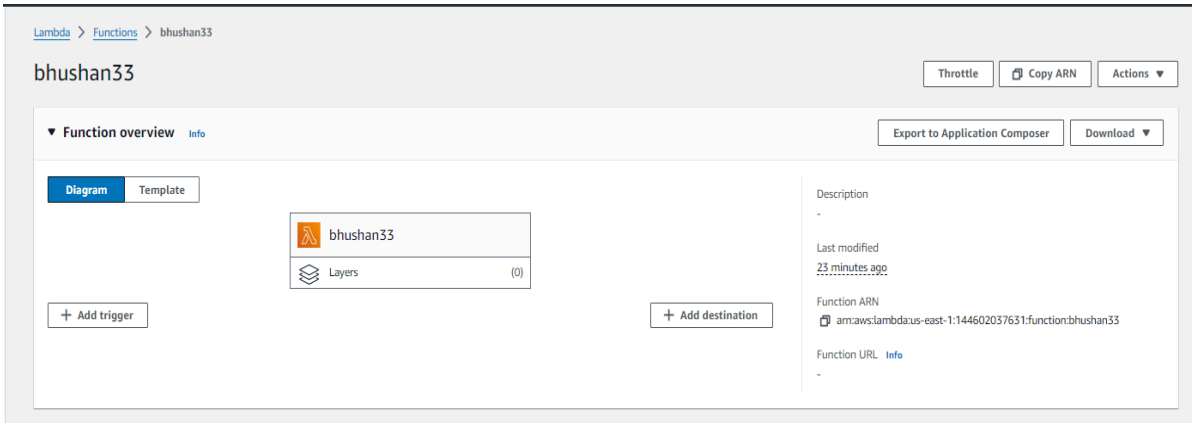


6) The image has been uploaded to the bucket.

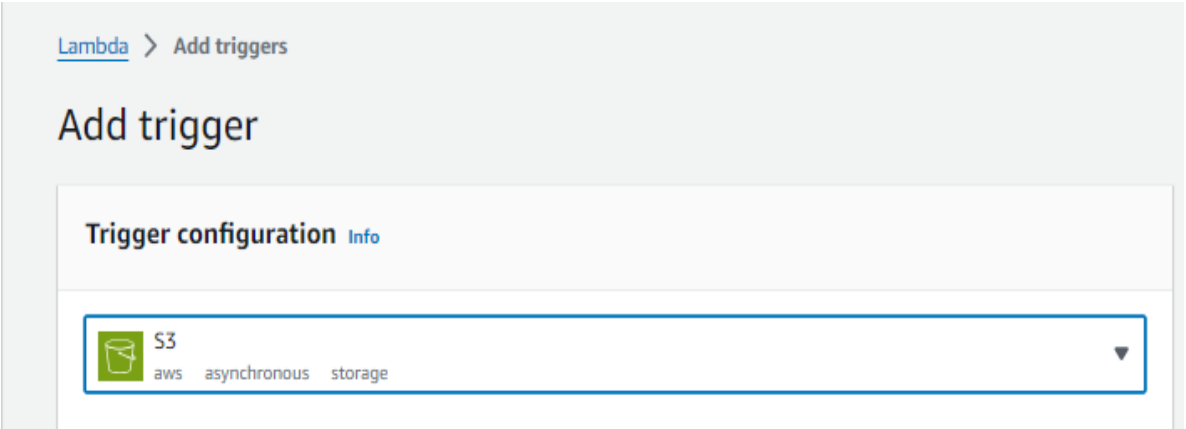


Step 2: Configure Lambda function

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

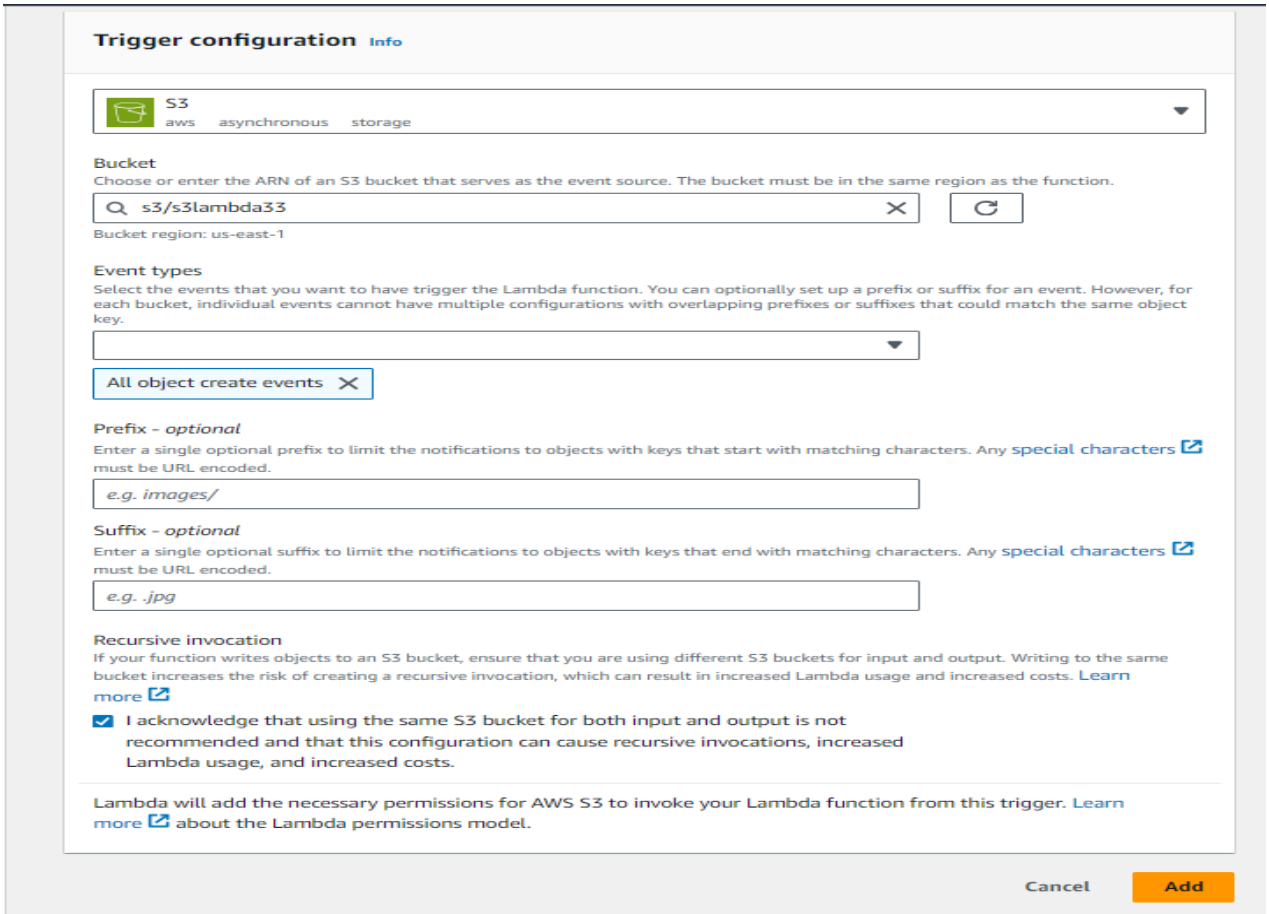


2) Under trigger configuration, search for S3 and select it.



Name:Sahil Motiramani Div :D15C Roll no:35

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



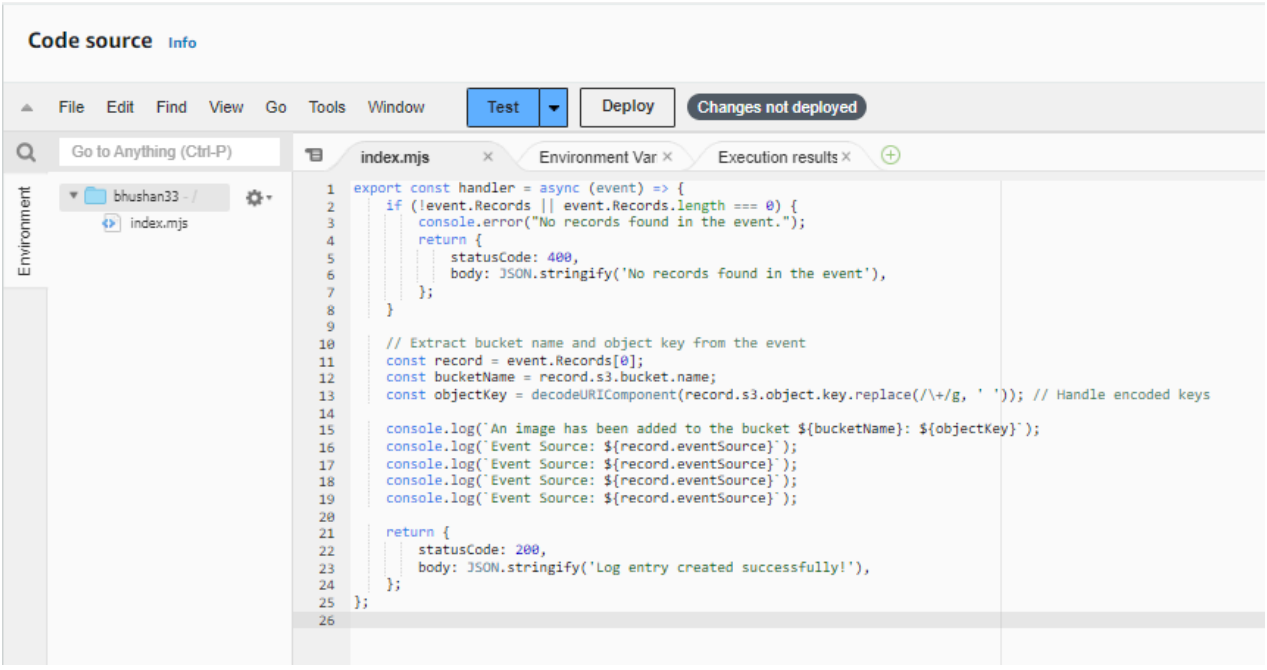
Scroll down to the code section of the function. Add the following javascript code to the code area by replacing the existing code.

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

Name:Sahil Motiramani Div :D15C Roll no:35

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

console.log(`An image has been added to the bucket ${bucketName}:
${objectKey}`); console.log(`Event Source: ${record.eventSource}`);
console.log(`Event Source: ${record.eventSource}`); console.log(`Event Source:
${record.eventSource}`); console.log(`Event Source: ${record.eventSource}`); return
{
    statusCode: 200, body: JSON.stringify('Log entry
    created successfully!')
  };
};
```

This code checks for records in the event, extracts the bucket name and object key, logs the details, and returns a success message if an image is added to the bucket.



Now, click on the dropdown near test, then click on configure test event.

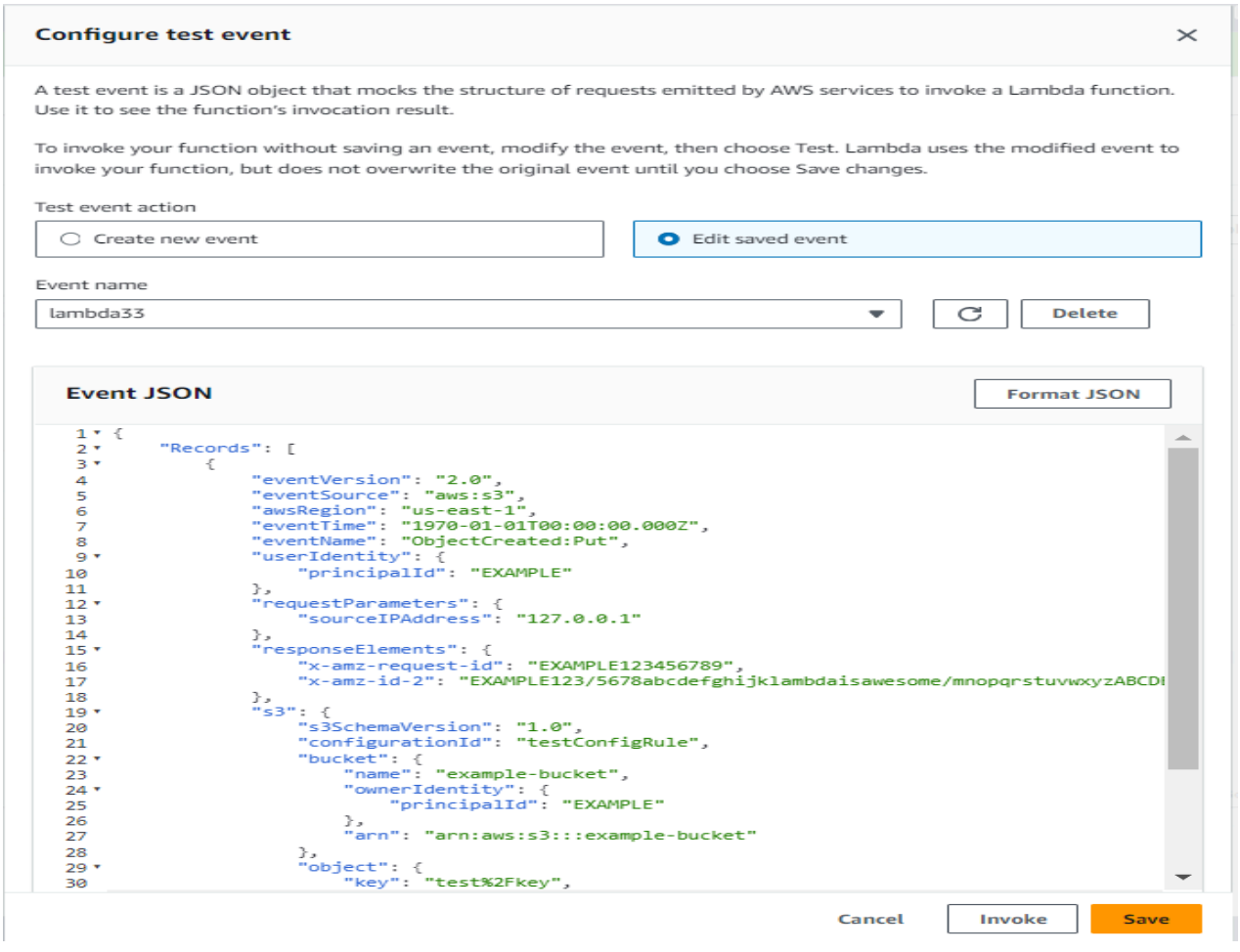
6) Here, select edit saved event. Select the event that you had created before. Under Event JSON, paste the following code.

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

"bucket": {
          "name": "example-bucket",
    "ownerIdentity": {
            "principalId": "EXAMPLE"
          },
          "arn": "arn:aws:s3:::example-bucket"
        },
        "object": {
```

Name:Sahil Motiramani Div :D15C Roll no:35


```
            "key": "test%2Fkey",
            "size": 1024,
            "eTag": "0123456789abcdef0123456789abcdef",
            "sequencer": "0A1B2C3D4E5F678901"
          }
        }
      }
    ]
  }
```


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

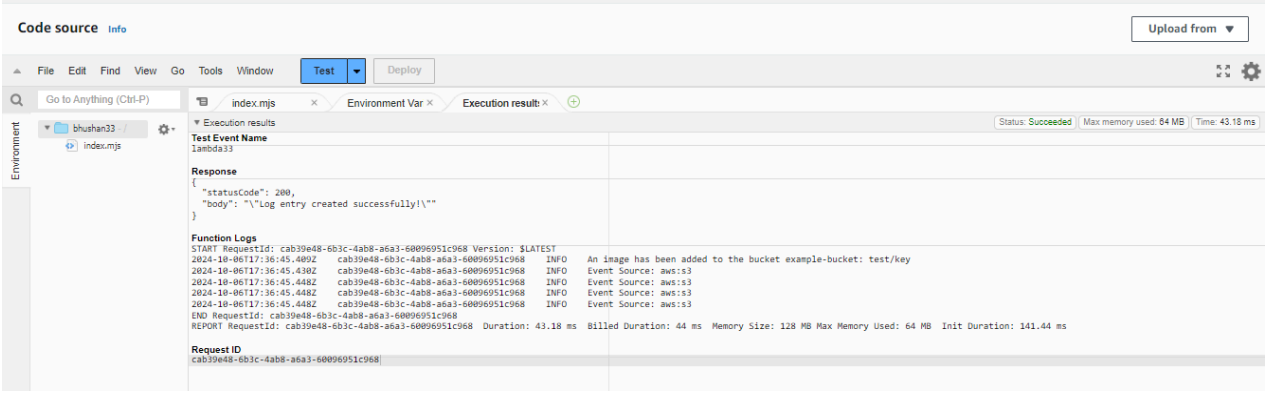


Name:Sahil Motiramani Div :D15C Roll no:35

Save the changes. Then deploy the code changes by clicking on deploy.

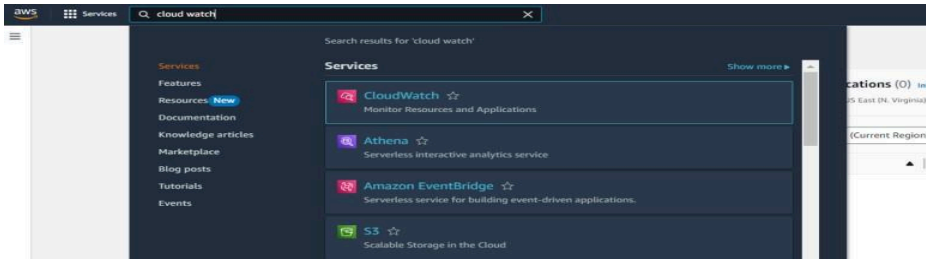
7) After deploying, click on Test. The console output shows that ‘an image has been added to the bucket’

The JSON response shows that the log entry was created successfully.

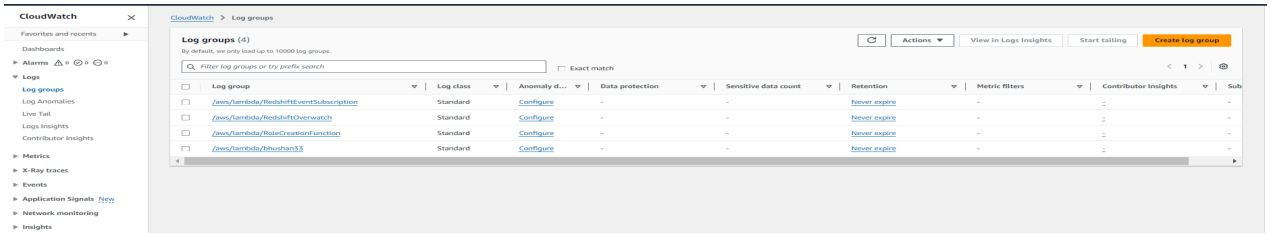


Step 3: Check the logs

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

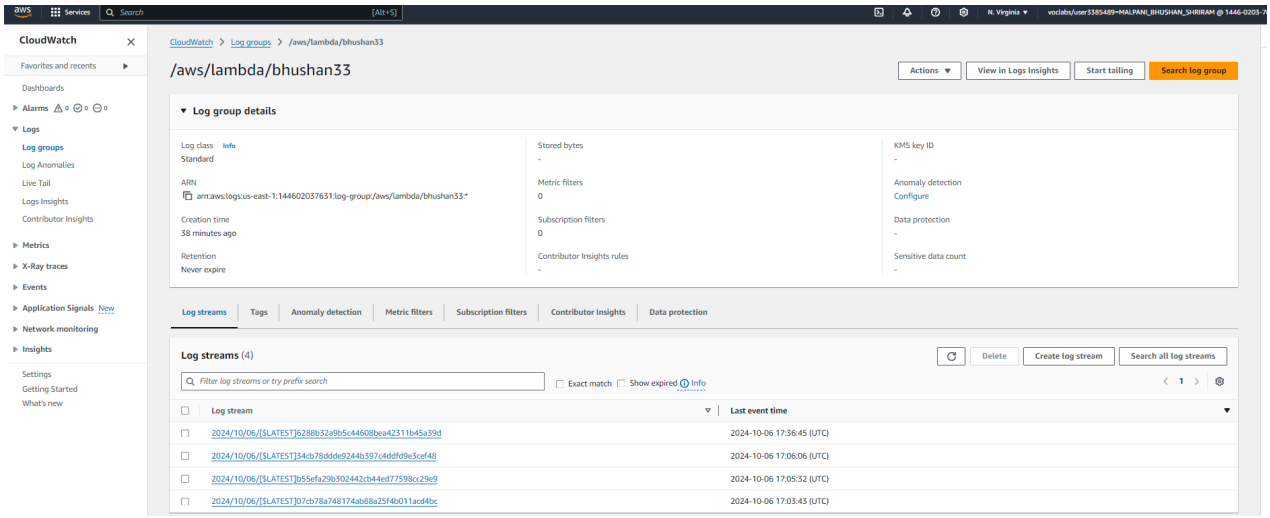


2) Here, Click on Logs → Log Groups. Select the log that has the lambda function name you just ran.

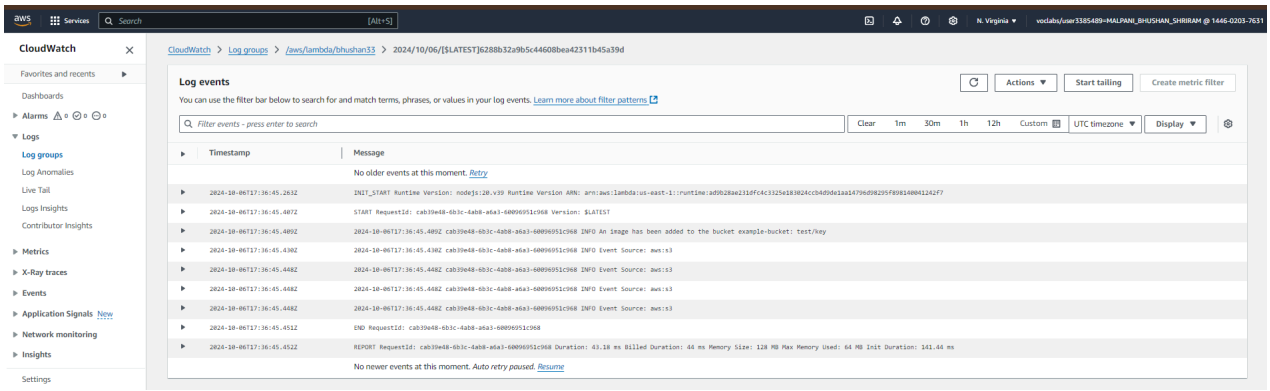


Name:Sahil Motiramani Div :D15C Roll no:35

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



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



Conclusion:

In this experiment, we developed and deployed a Lambda function designed to respond to file uploads in an S3 bucket. The function was triggered automatically whenever a new object was added to the bucket, illustrating how AWS services can efficiently automate workflows. The Lambda function extracted and logged key details from the event, such as the bucket's name and the object's key. We tested this by uploading a sample file, and upon reviewing the logs in CloudWatch, we confirmed that the function executed successfully, capturing the upload event. This experiment demonstrated the powerful synergy between AWS Lambda and S3, enabling seamless, event-driven automation.