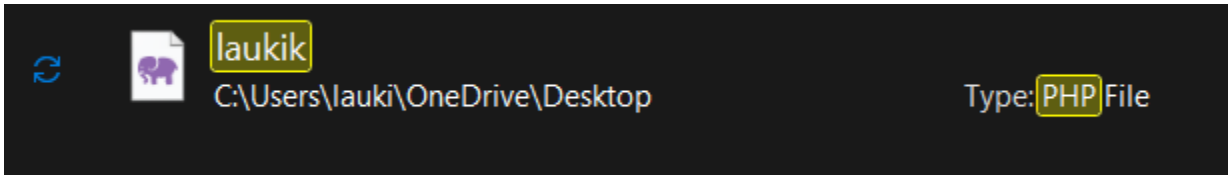
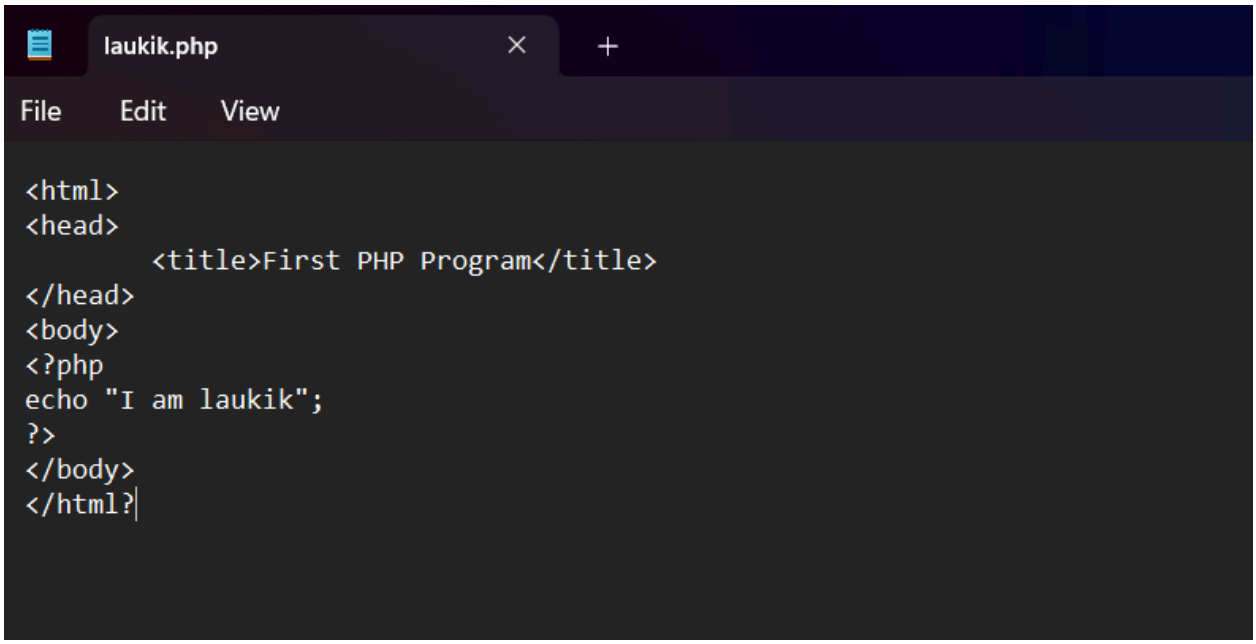
Exp 1a : Static Hosting

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

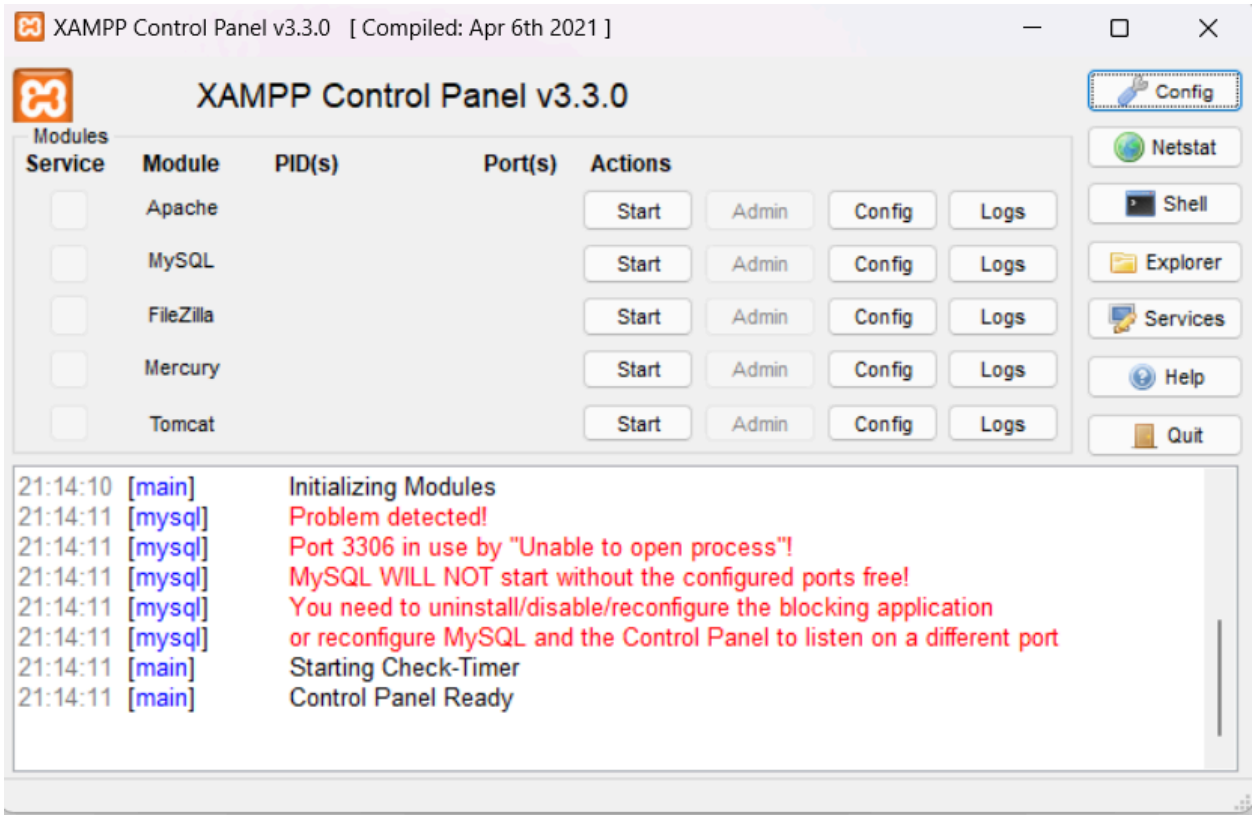
1. Create a .php file in some local repository



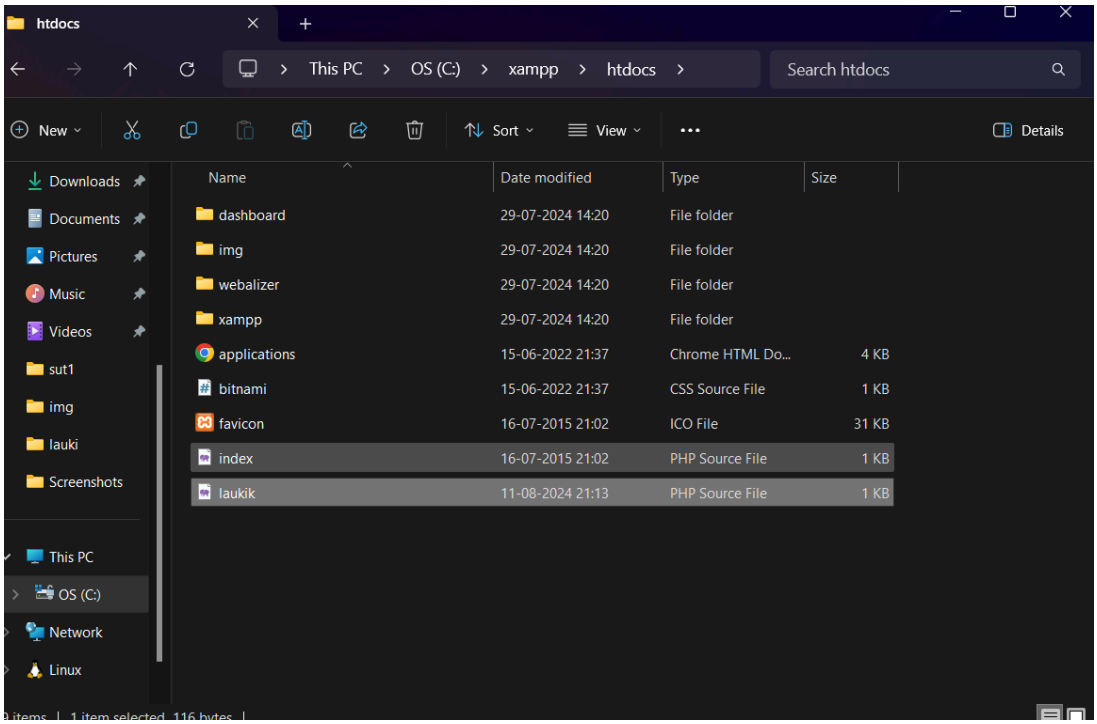
2. Make changes in that .php file as per your desire and save it



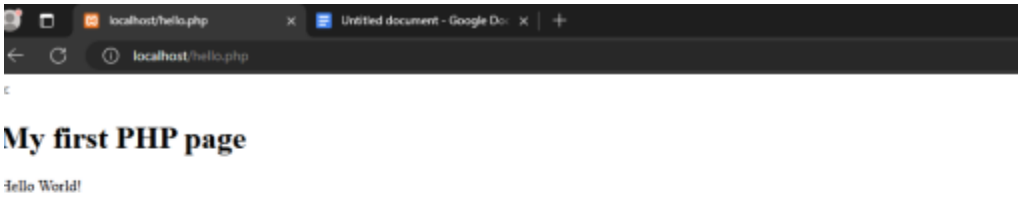
3. Make sure you have installed Xampp on your local machine. After the installation, start Xampp control panel and start modules named Apache and MySql



4. Ensure that you relocate your .php file in the htdocs folder inside the Xampp folder

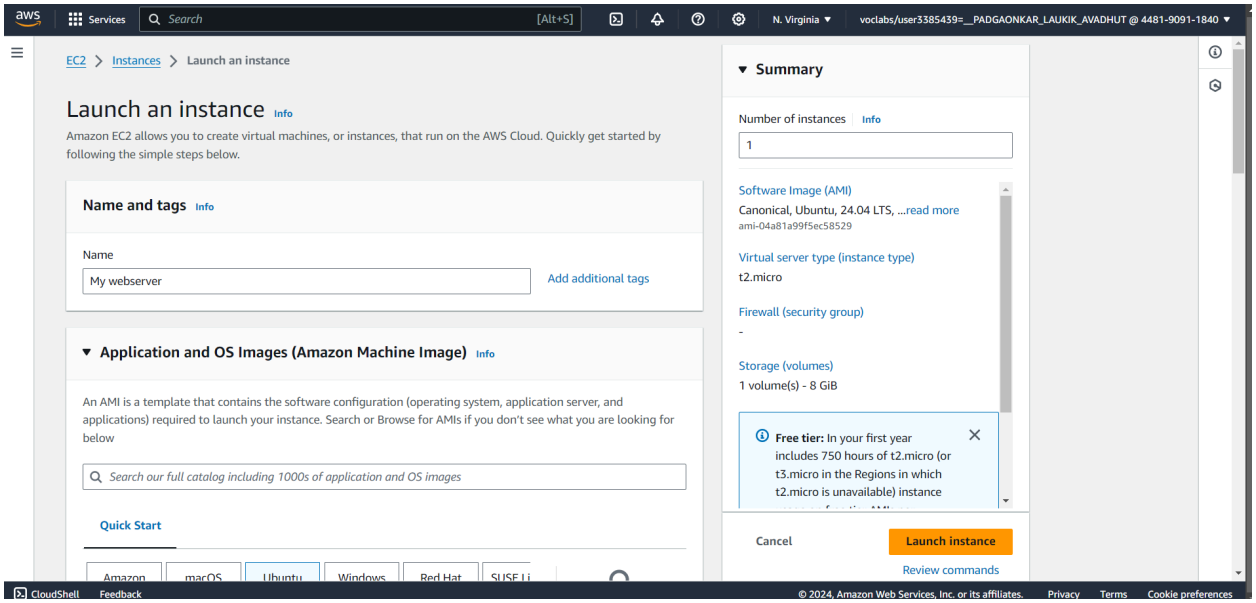


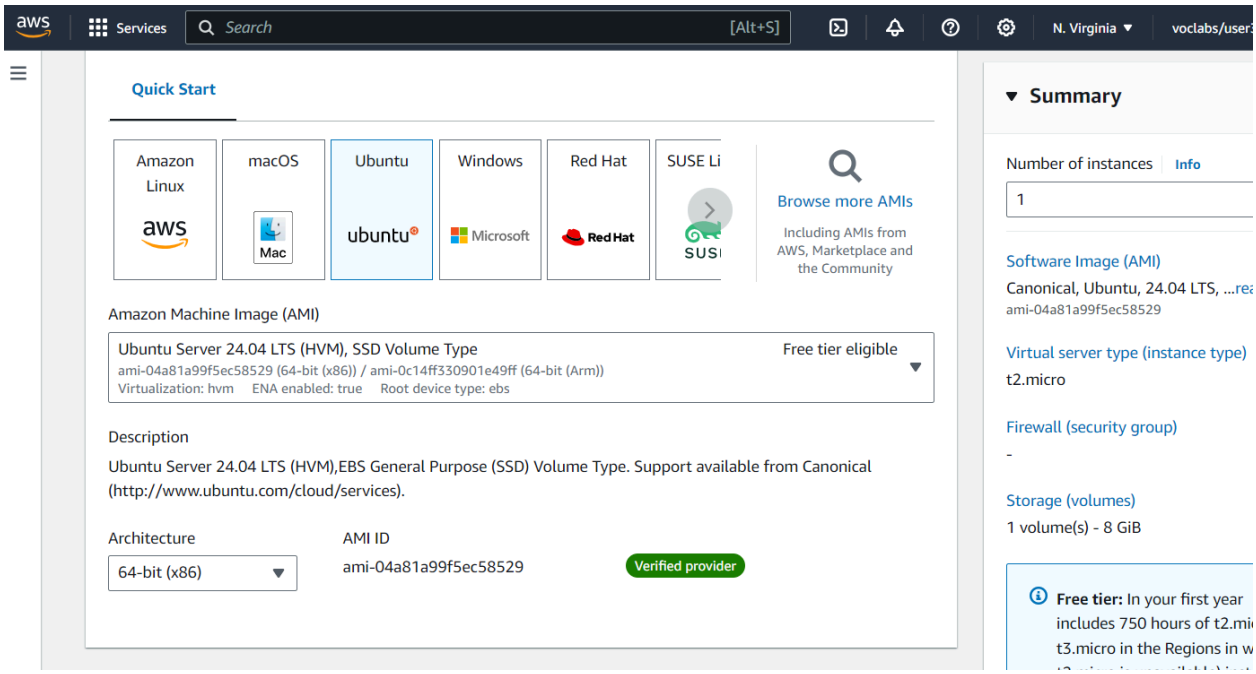
5. Access the contents of the php file by typing localhost/your_file.php on your browser. We have successfully hosted our php file on our local machine using Xampp.

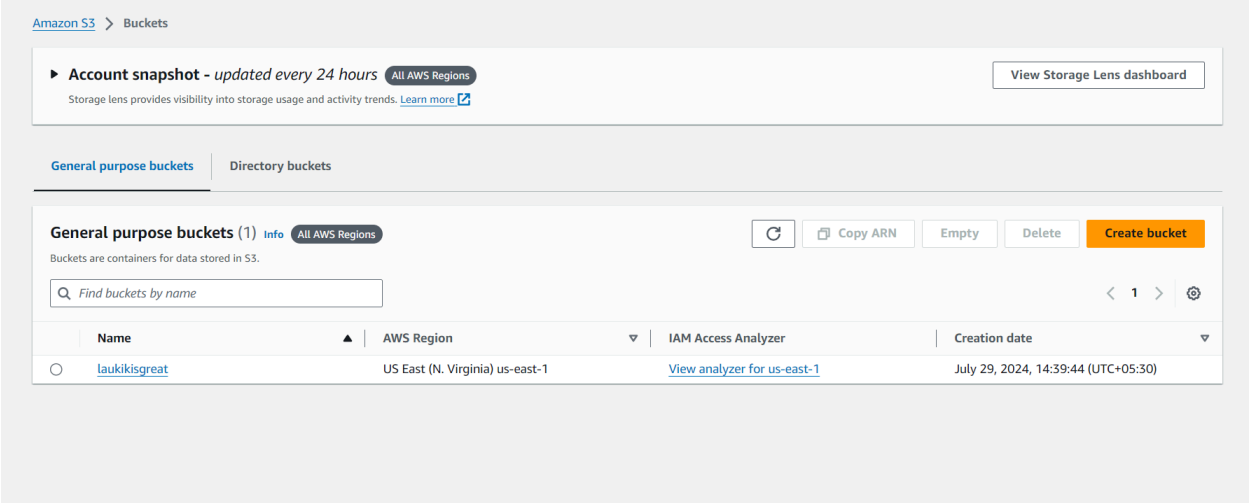


b) Static hosting using AWS S3 bucket

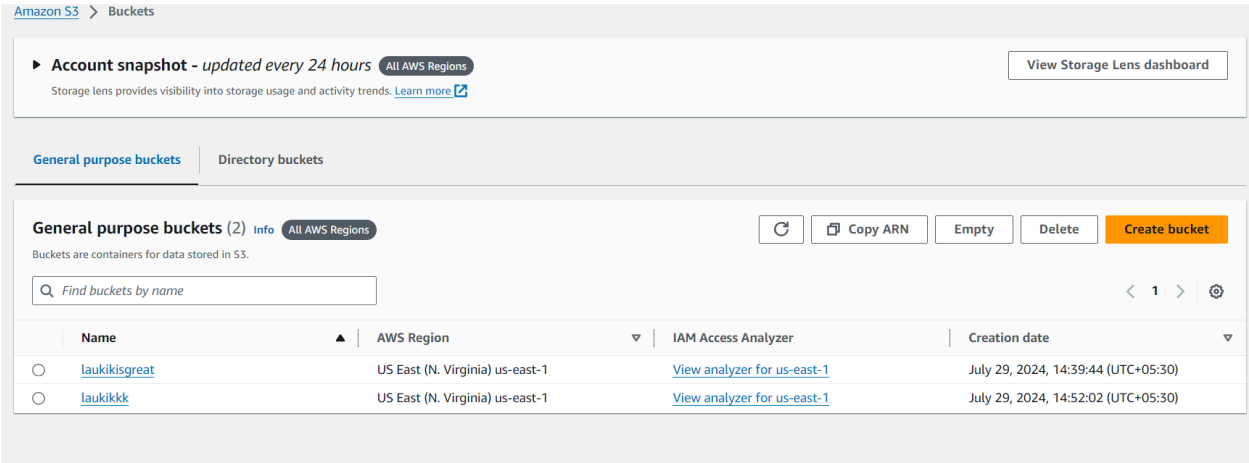
1. Navigate to S3 inside services and create a bucket inside S3. Select the additional settings related to the bucket.



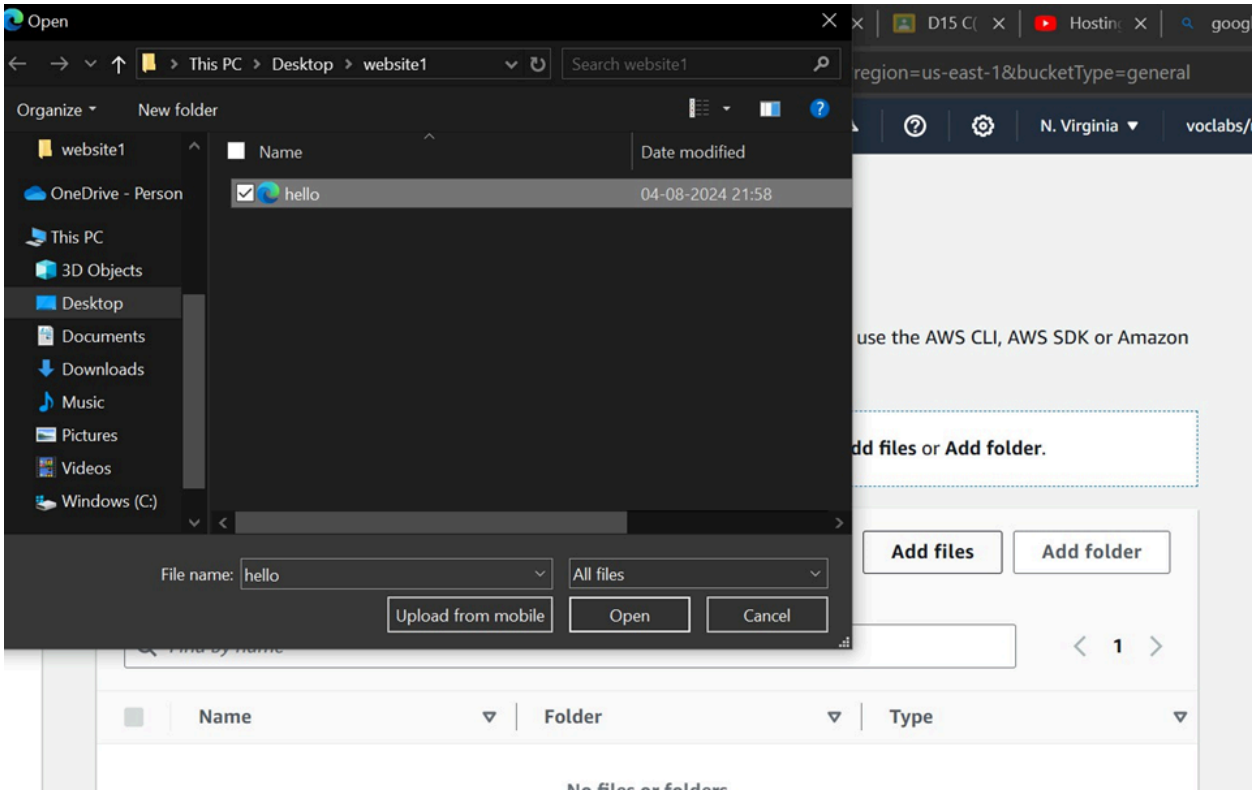




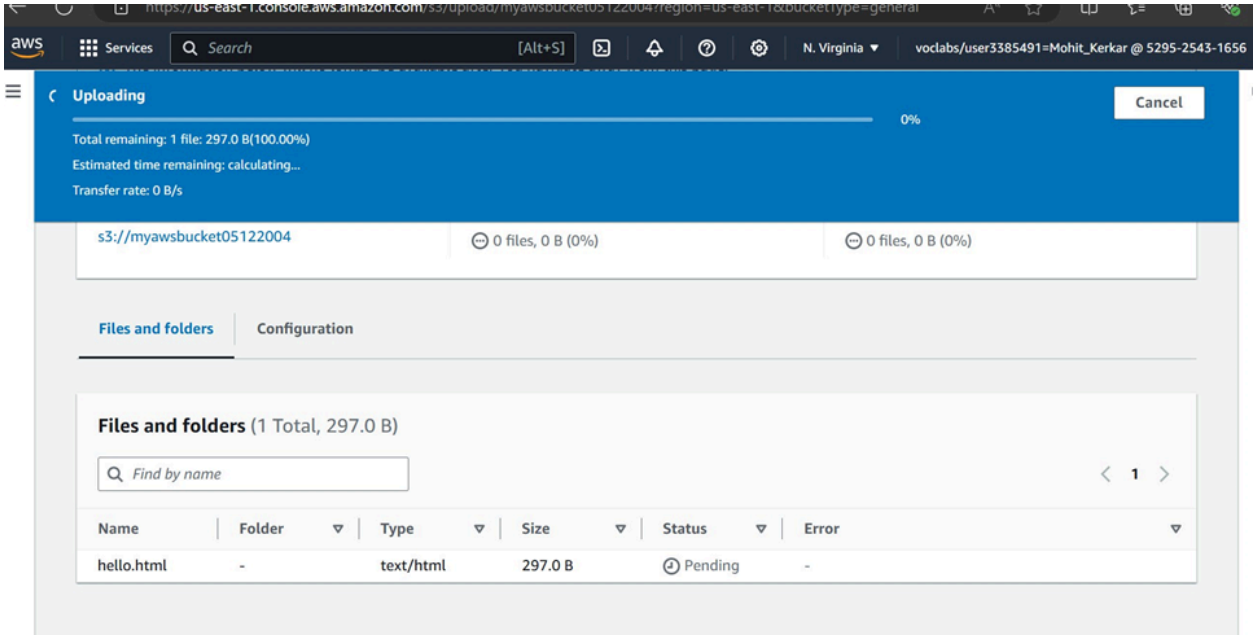
2. Our bucket was successfully created. Now, we would want to add/upload our local files onto our bucket

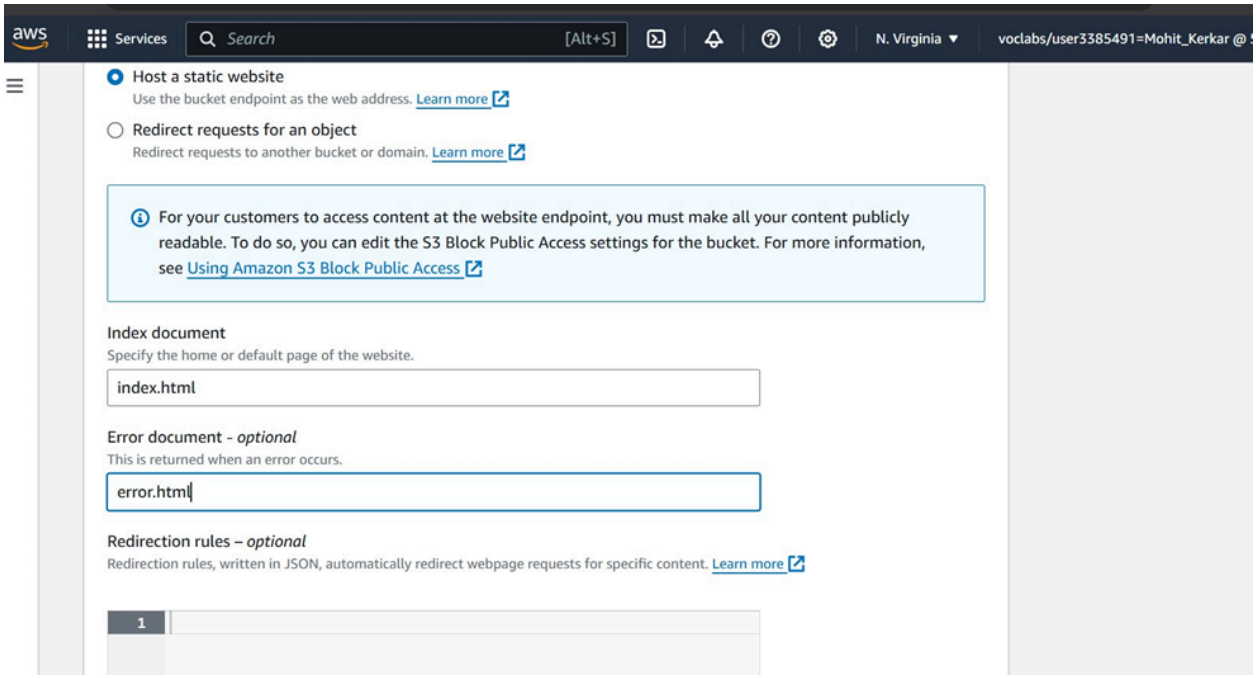


3. Create a new html file, modify it and save it with a desired file name. Now, add/upload this file onto your S3 bucket.

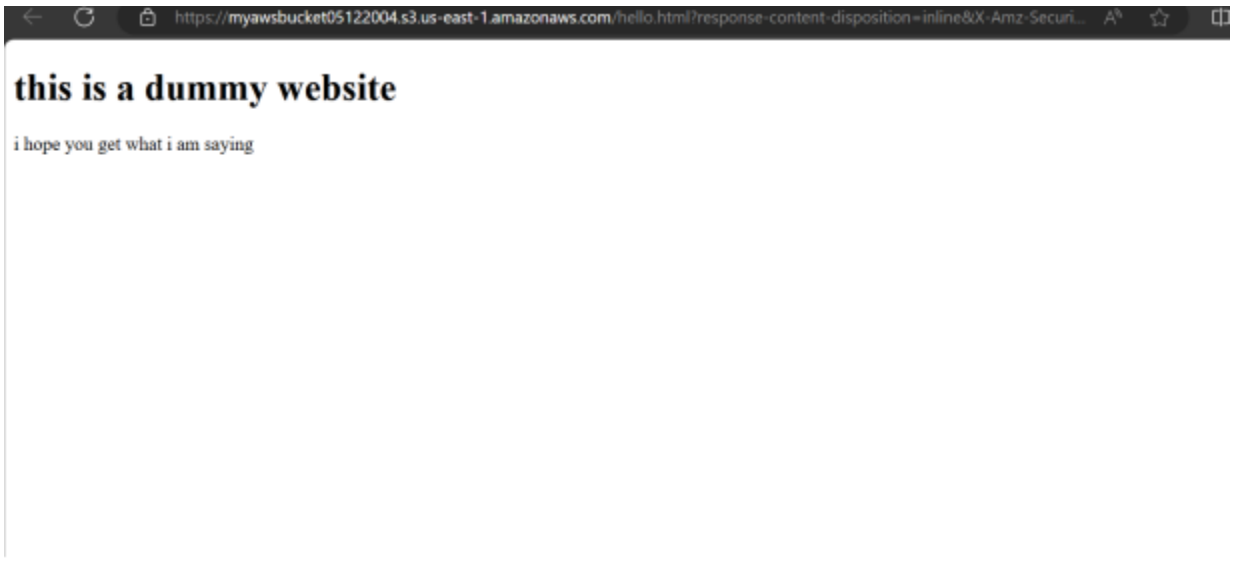


4. For enabling static website hosting, we must enable this Static website hosting option





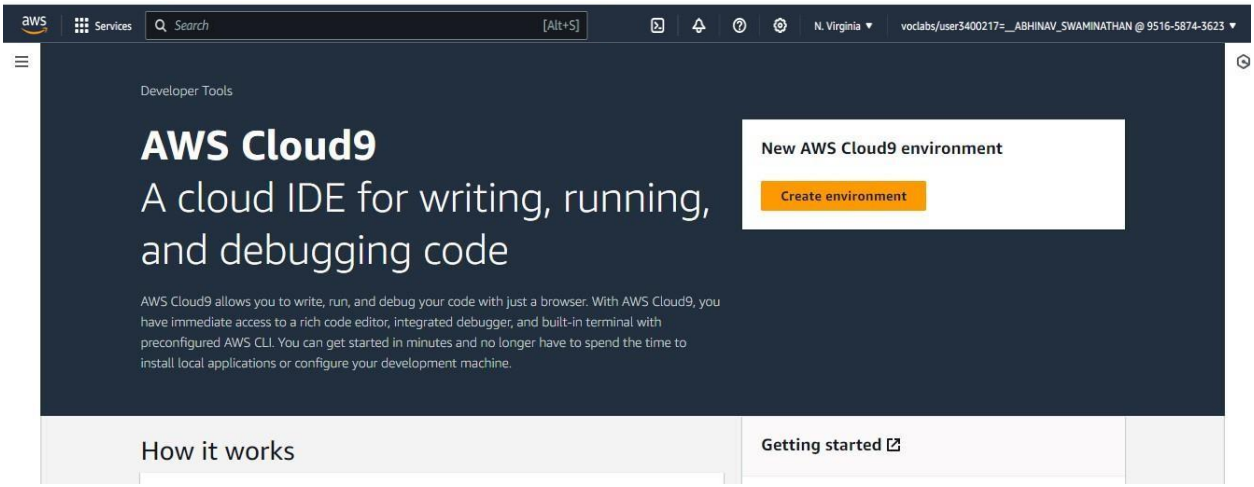
5. Open the website using the S3 bucket. You'd be able to see the contents of you html file. Thus, we have successfully and



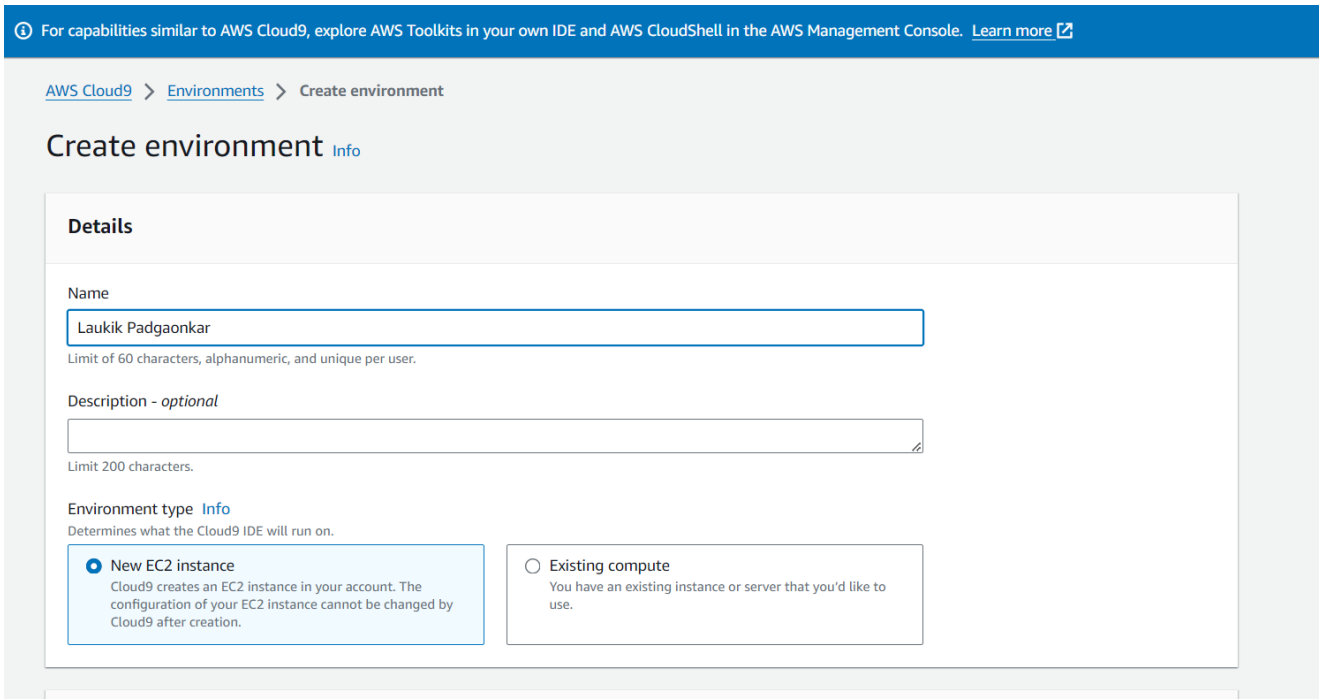
Name: Laukik Padgaonkar D15C Roll No:37

Experiment No. 1B

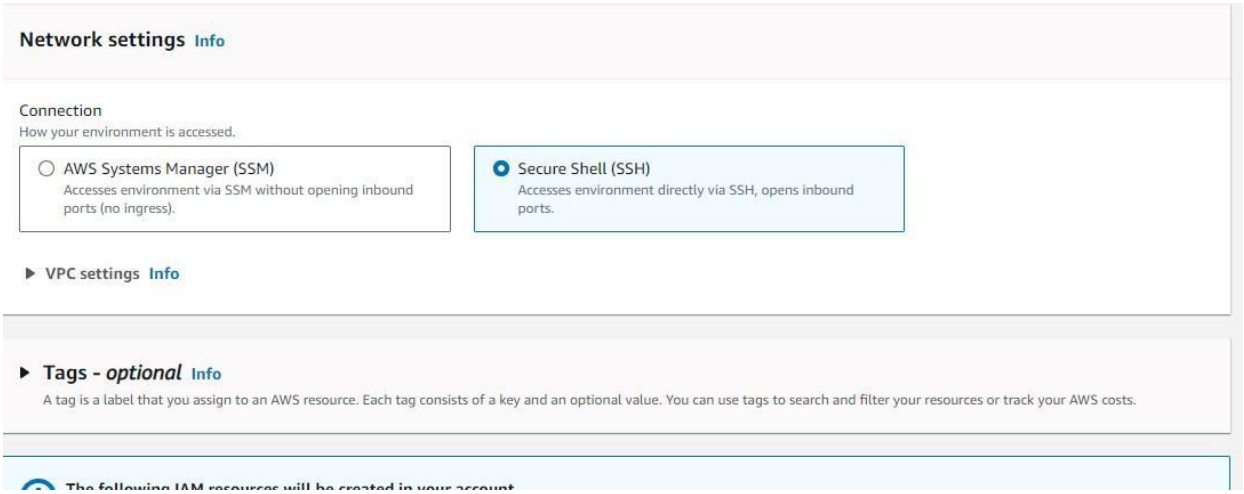
Open the AWS account and search for Cloud9. Click on create environment.



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

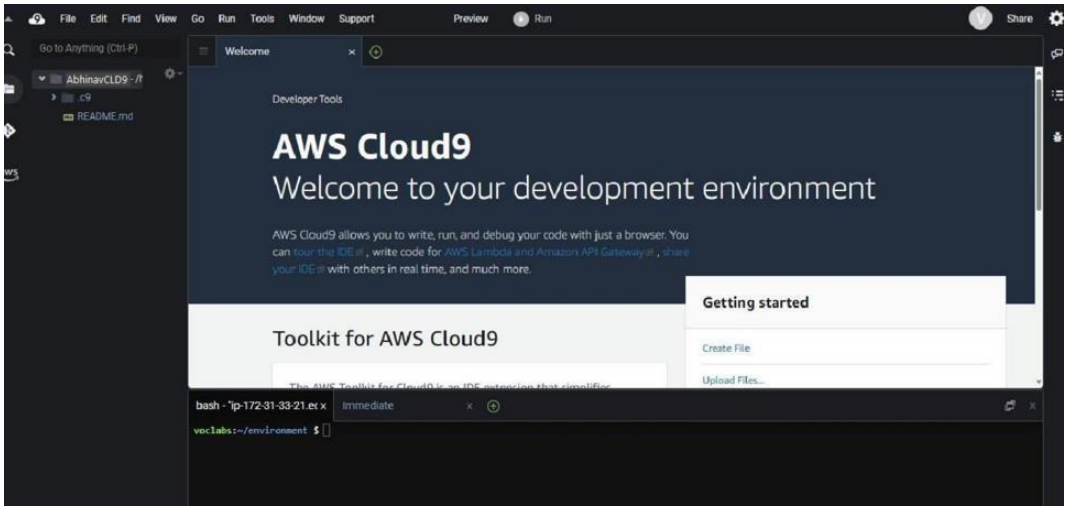


Use the Secure Shell option in Network settings

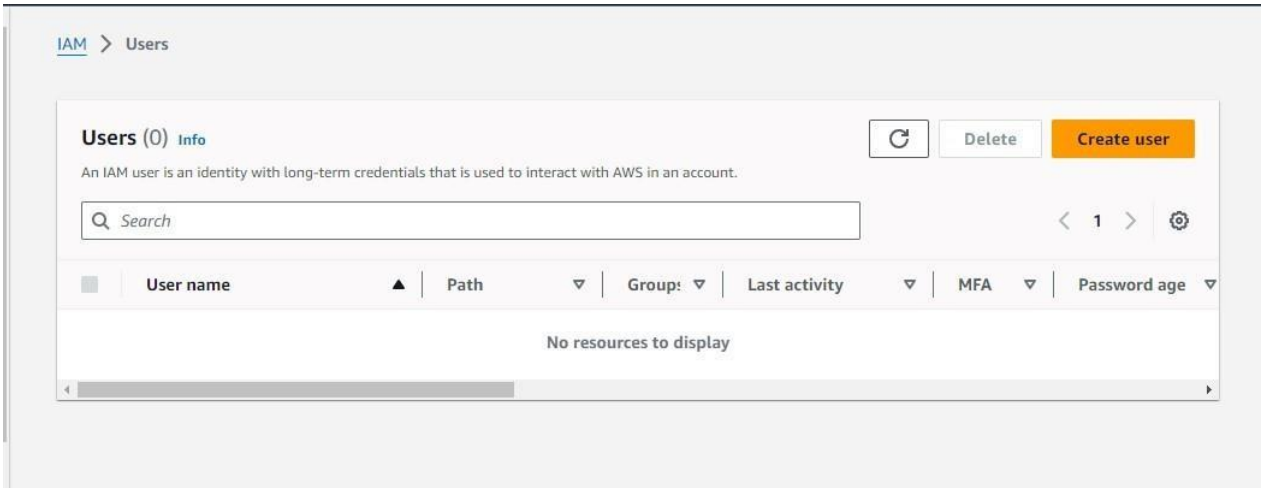


Once the configuration is complete, click on create environment to create a Cloud9 environment.

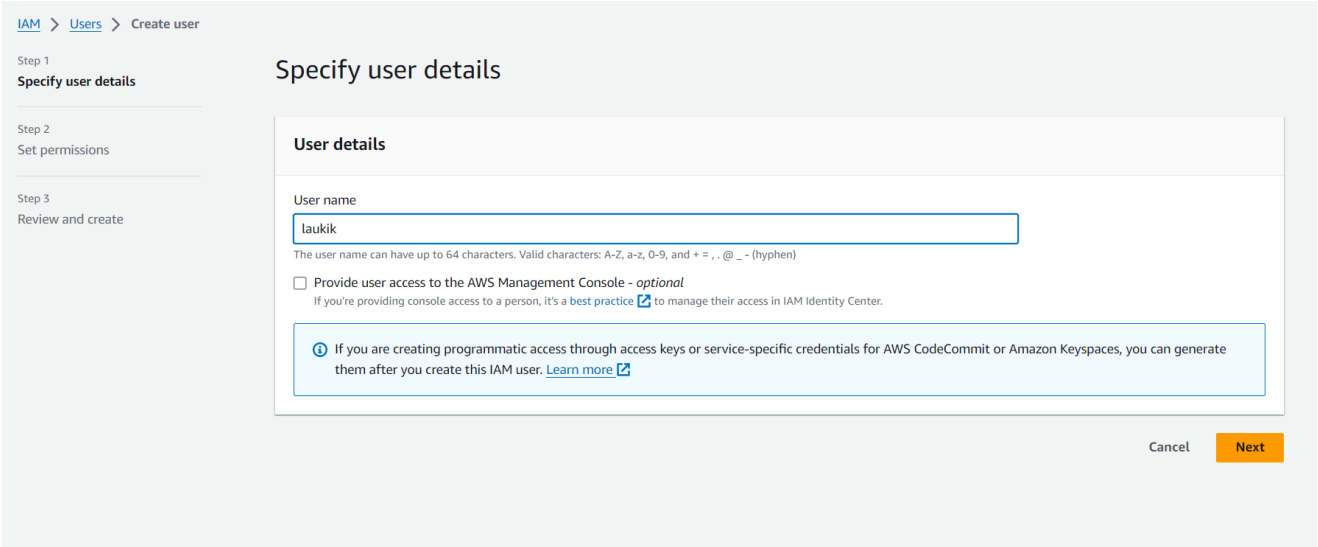
Click on the environment name to open the created Cloud9 Environment.

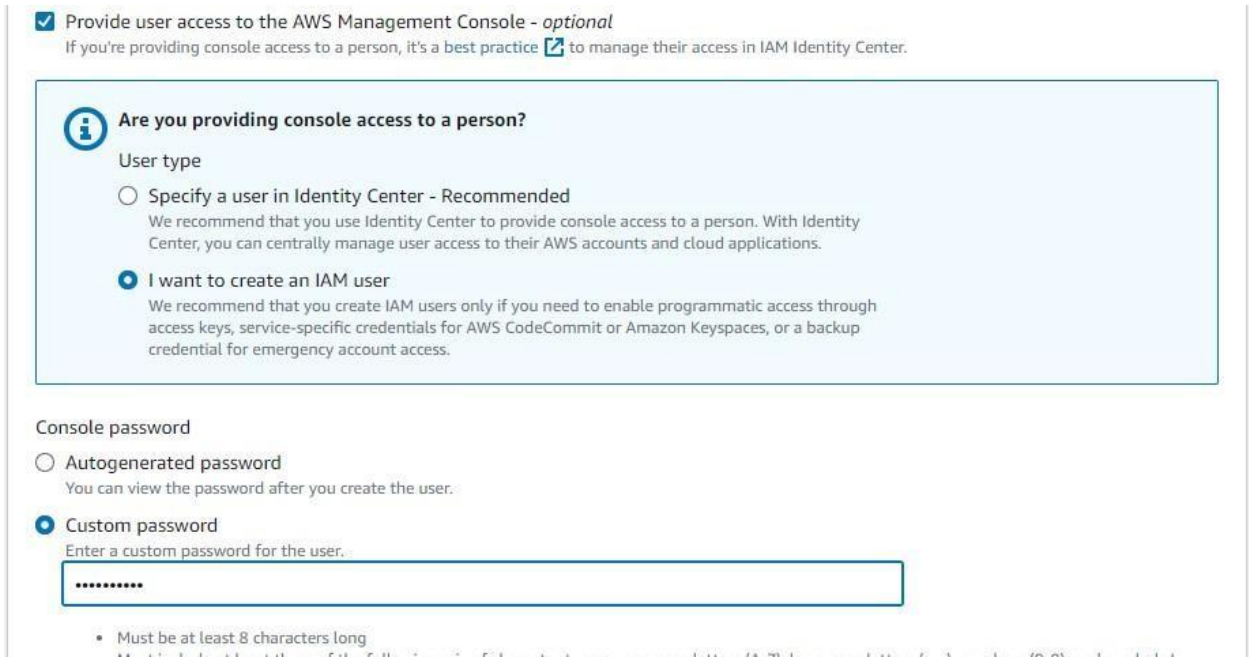


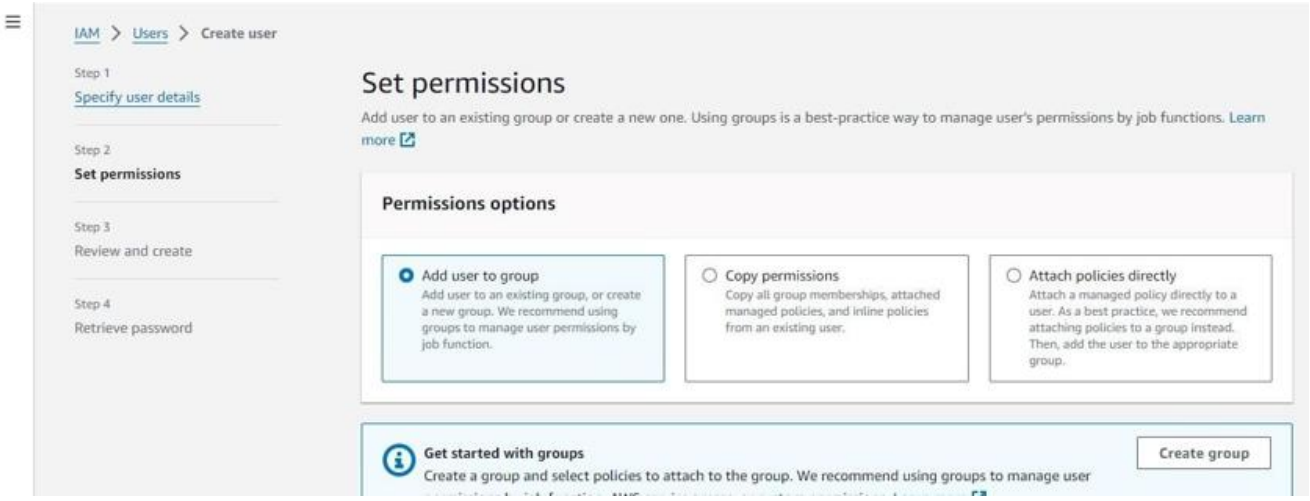
Open the aws account and search for IAM service. Then go to users tab and click on create user to create a new user.



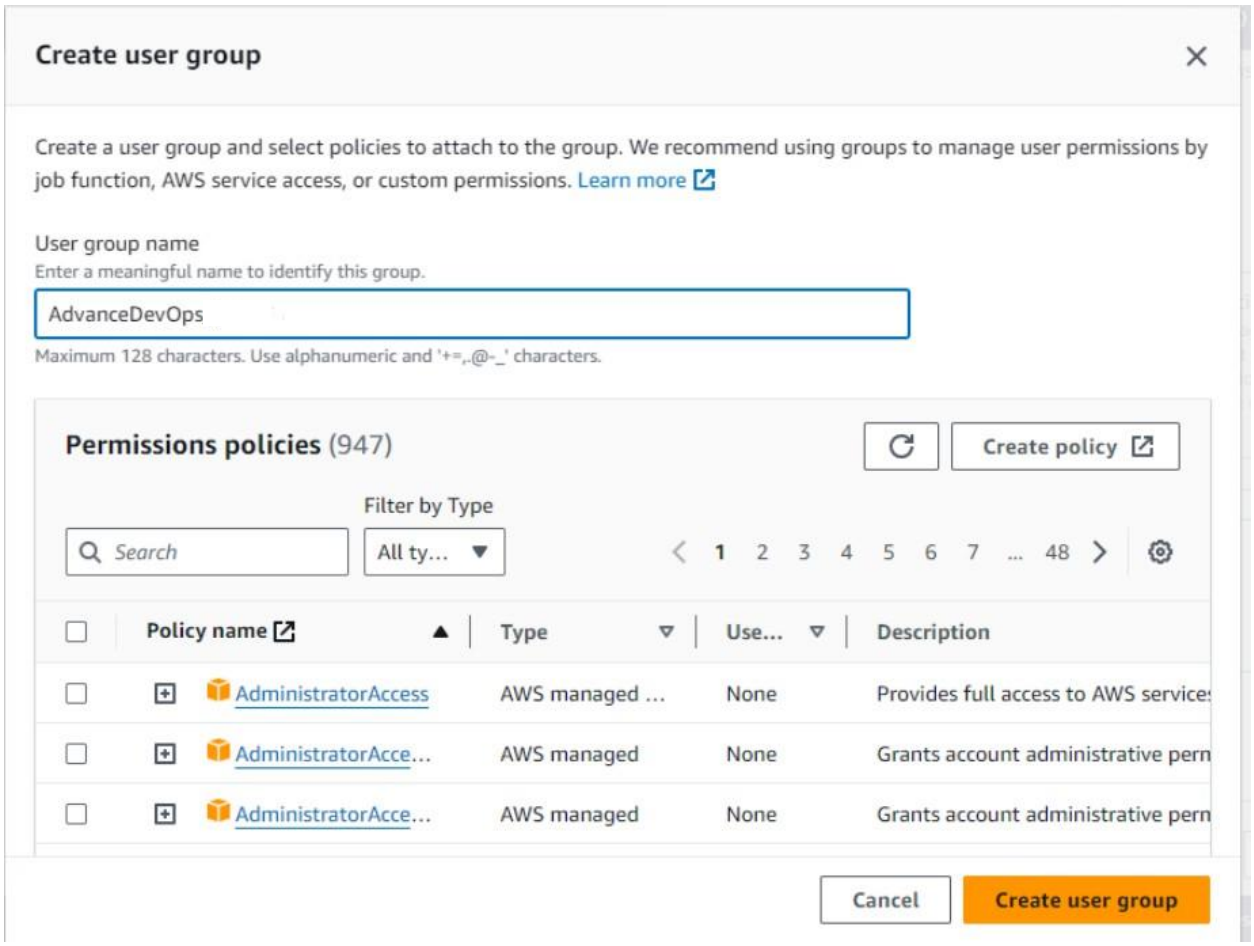
Give the user name, select the “provide user access” checkbox. Also select the option “I want to create an IAM user”. Otherwise we will have to enable the Identity center and specify a user there.



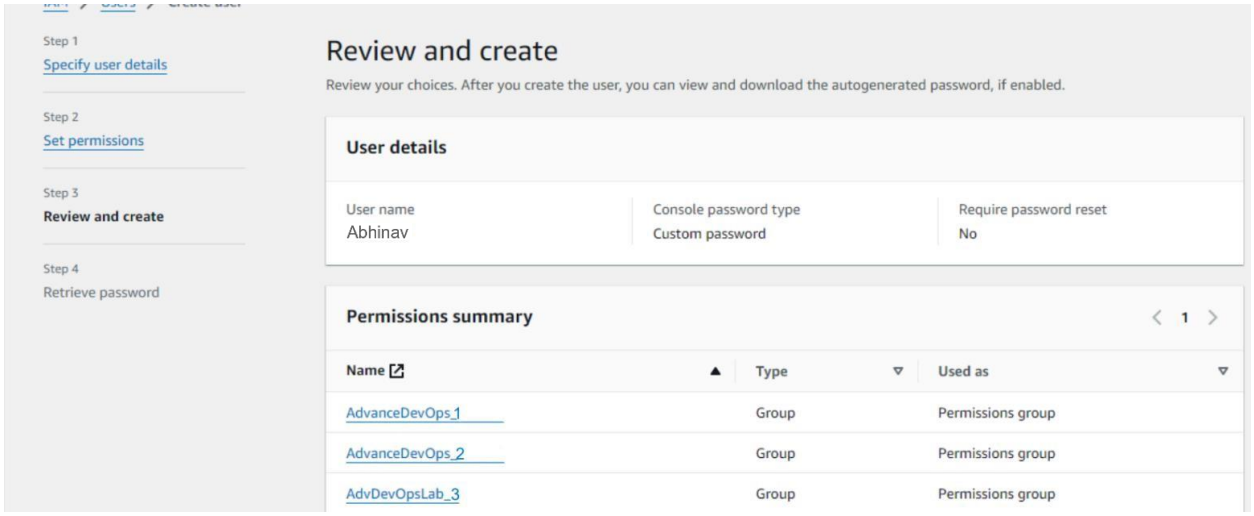

Provide user access to the AWS Management Console - optional
If you're providing console access to a person, it's a best practice to manage their access in IAM Identity Center.
Are you providing console access to a person?
User type
Specify a user in Identity Center - Recommended
We recommend that you use Identity Center to provide console access to a person. With Identity Center, you can centrally manage user access to their AWS accounts and cloud applications.
I want to create an IAM user
We recommend that you create IAM users only if you need to enable programmatic access through access keys, service-specific credentials for AWS CodeCommit or Amazon Keyspaces, or a backup credential for emergency account access.
Console password
Autogenerated password
You can view the password after you create the user.
Custom password
Enter a custom password for the user.
••••••••••
Must be at least 8 characters long


IAM > Users > Create user
Step 1
Specify user details
Step 2
Set permissions
Step 3
Review and create
Step 4
Retrieve password
Set permissions
Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. Learn more
Permissions options
Add user to group
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.
Copy permissions
Copy all group memberships, attached managed policies, and inline policies from an existing user.
Attach policies directly
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.
Get started with groups
Create a group and select policies to attach to the group. We recommend using groups to manage user
Create group

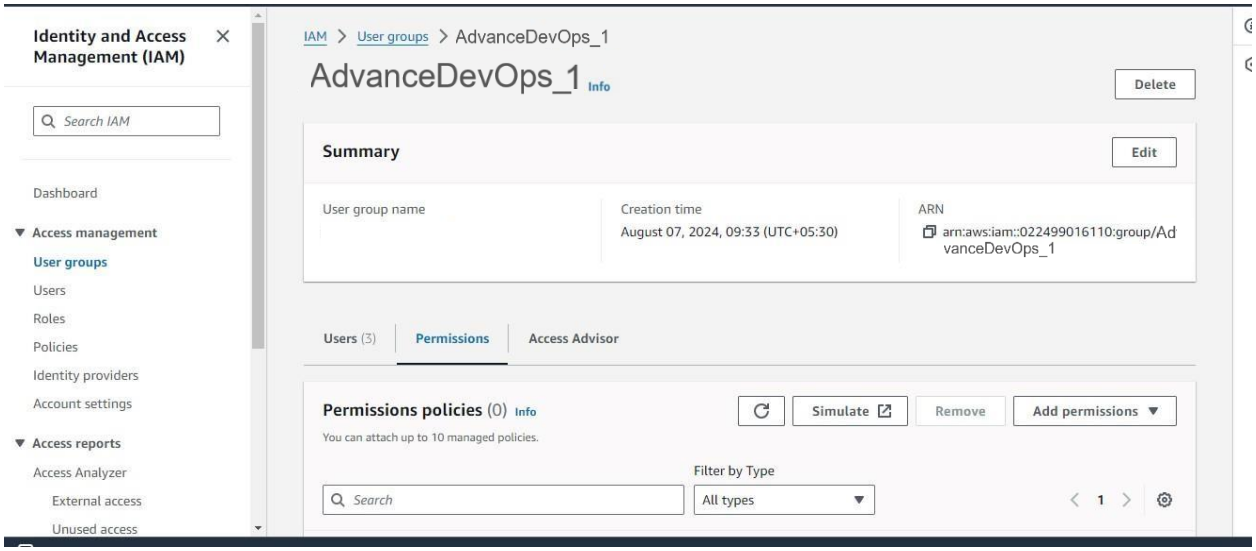
Next click on add user to group. If you do not have an existing group, select create group. Then Give the group name and policies if required and create a group.



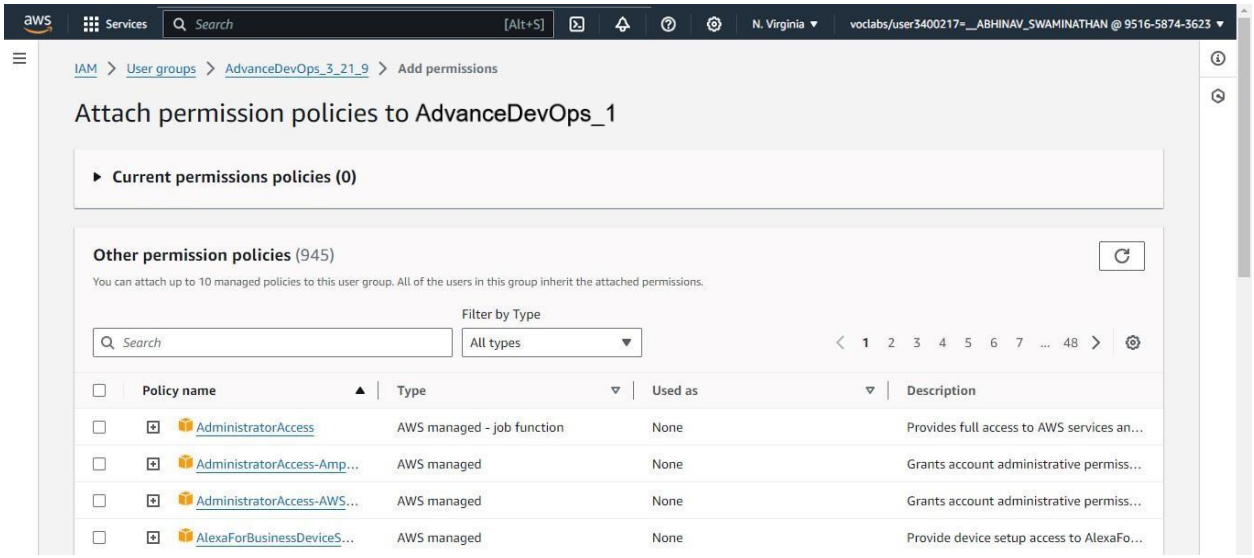
Once the group is created, select the group in which the user should be added.

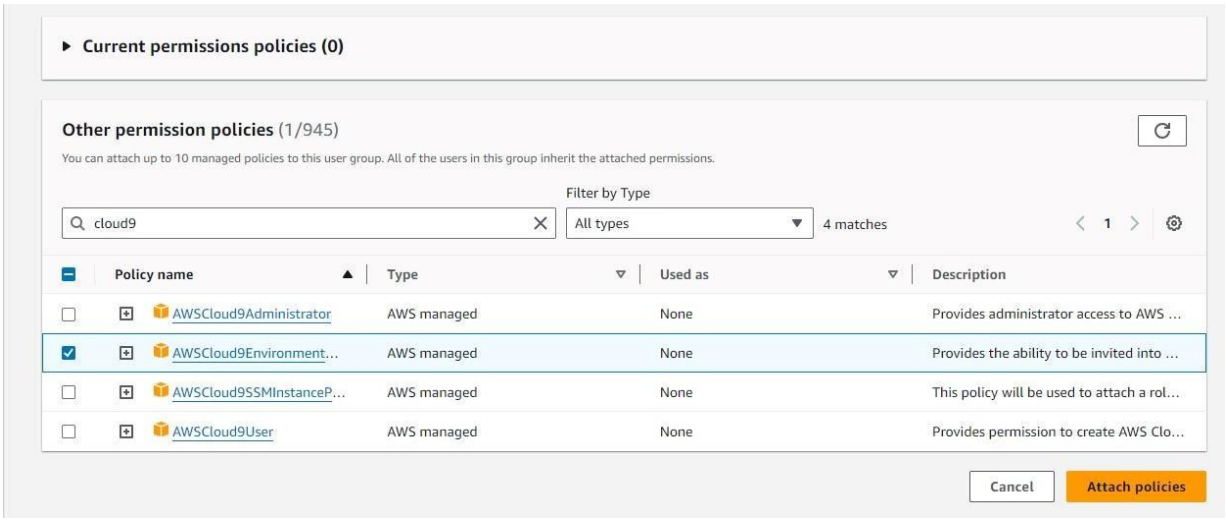


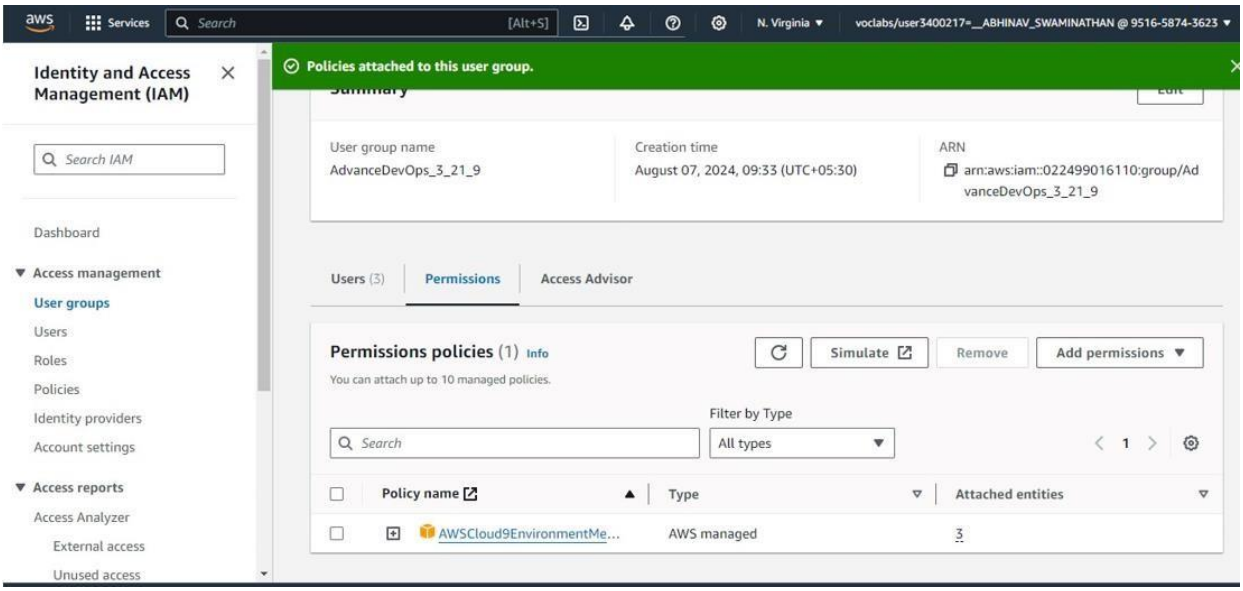
After creation of the user, go to the user groups tab and select the group in which the user has been added. Navigate to the permissions tab. Click on add permissions and attach policy.



Search for the “AWSCloud9EnvironmentMember” policy and attach it.



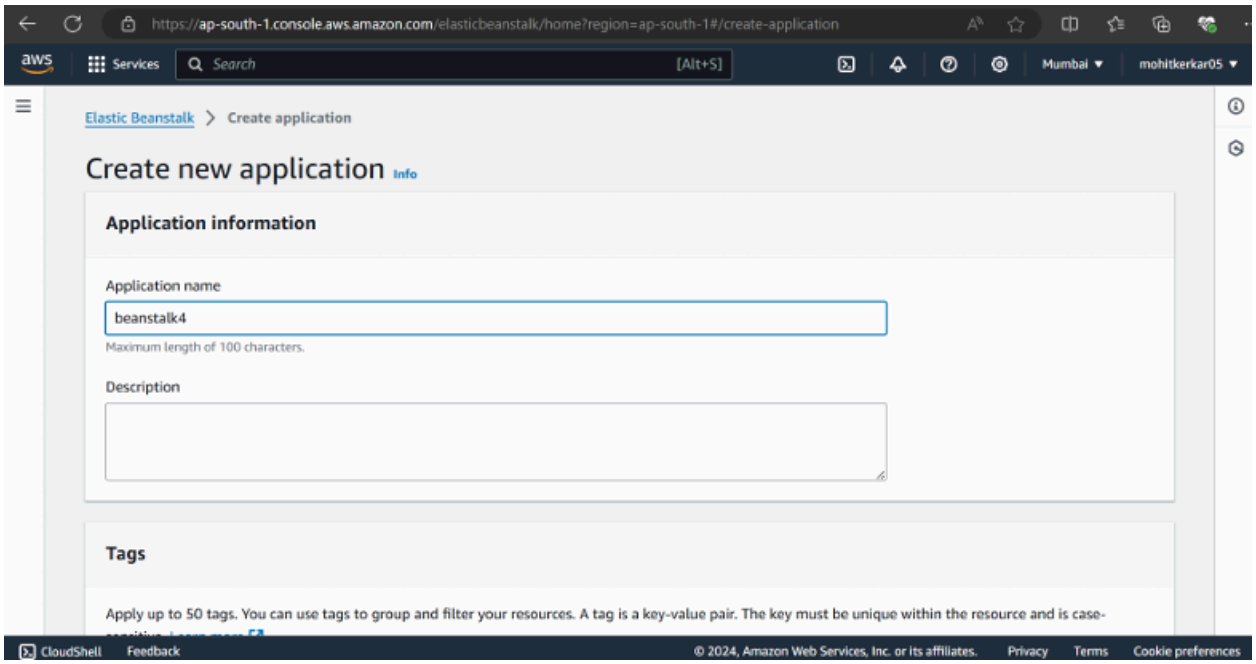


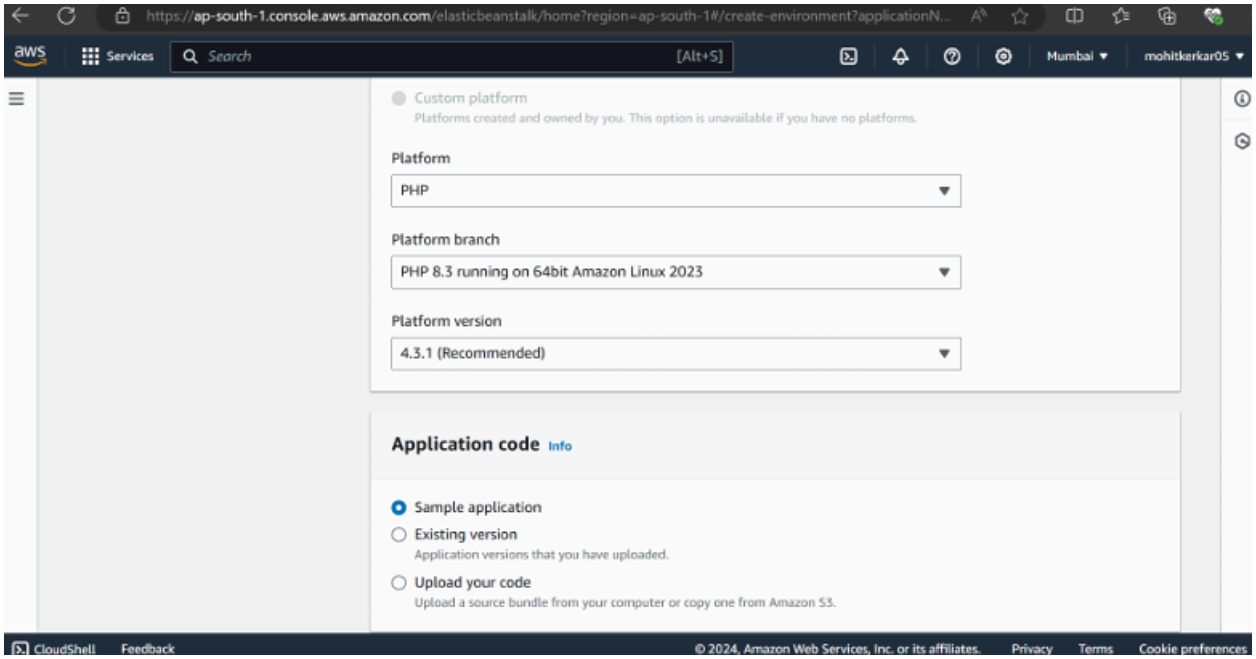


Exp 02:To Build Your Application using AWS CodeBuild and Deploy on S3 / SEBS using AWSCodePipeline, deploy Sample Application on EC2 instance using AWS CodeDeploy.

Step 1: Create our ElasticBeanstalk Environment

Login into your AWS account and navigate to services. Search for Elastic Beanstalk service and click on create application. Give your application a suitable name. For the platform, select PHP. Rest of the configuration settings are to be kept as default.

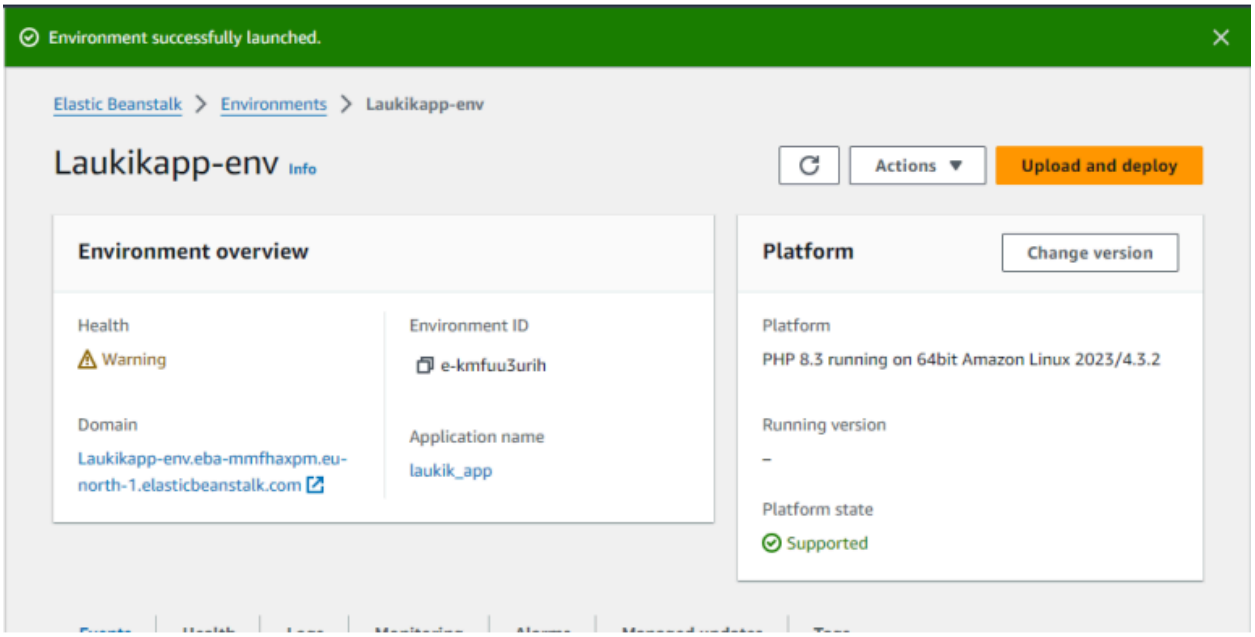


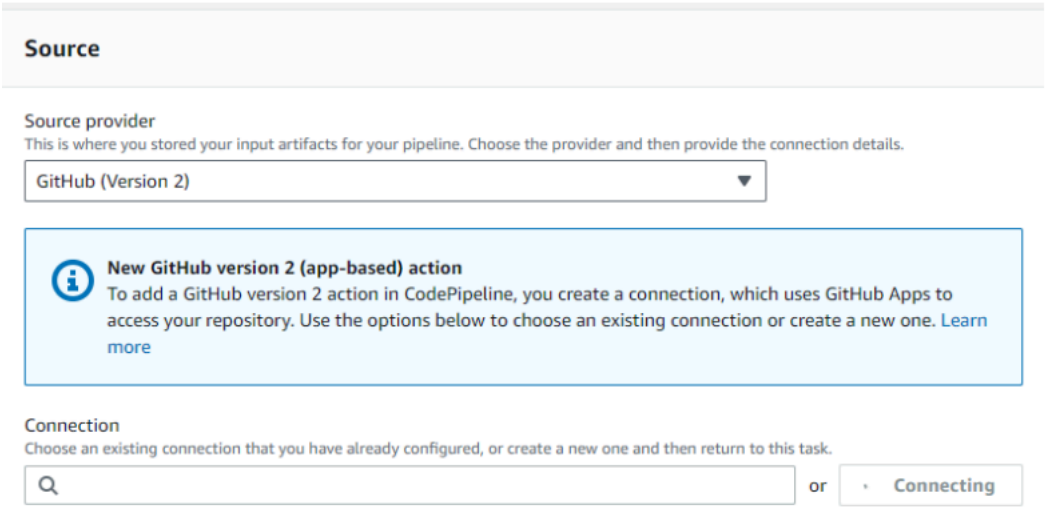


Now, while creating the environment, we are asked to provide an IAM role with the necessary EC2 permissions. We are supposed to make sure that we have made an existing IAM role with the following set of permissions:

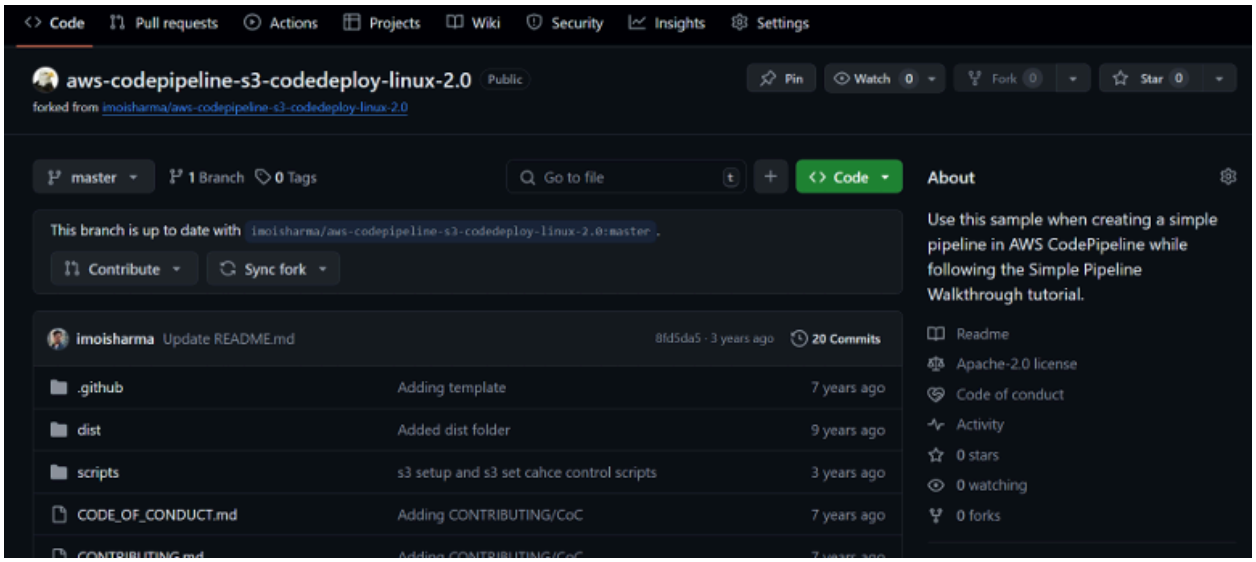
1. AWSElasticBeanStalkWebTier
2. AWSElasticBeanStalkWorkerTier
3. AWSElasticBeanStalkMulticontainerDocker

Wecan skip the steps to follow after the initial few steps mentioned above and move straight to review the settings of our environment. After reviewing everything properly, our environment can successfully be created.

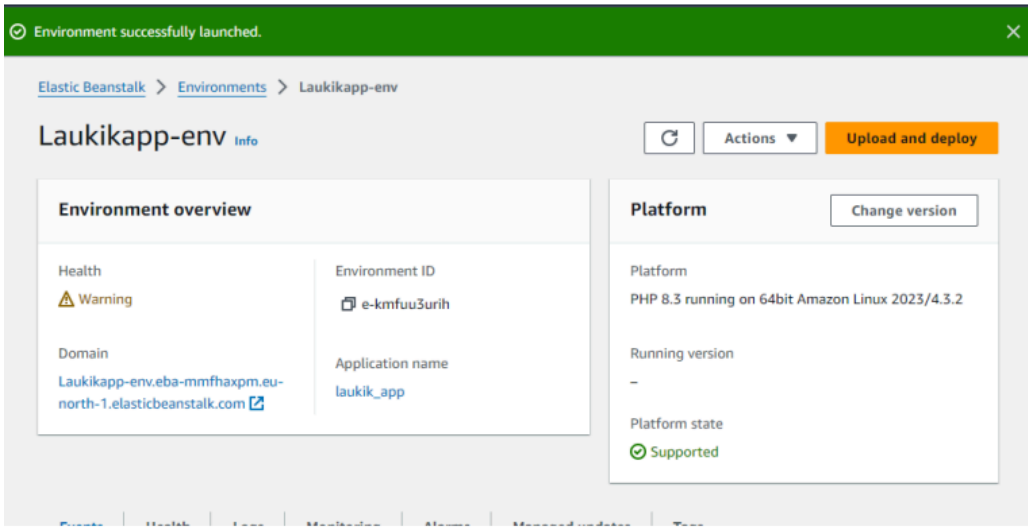




Step 2: Fork the required repository onto our github account
The repository to be forked is- imoisharma/aws-codepipeline-s3-codedeploy-linux-2.0
This step is necessary for the execution of the steps to follow. It will be helpful in the creation of a pipeline.



Step 3: Creation of the Pipeline
Navigate to Codepipeline inside Developer Tools. Give a suitable name to the pipeline you want to create.
And click on next …

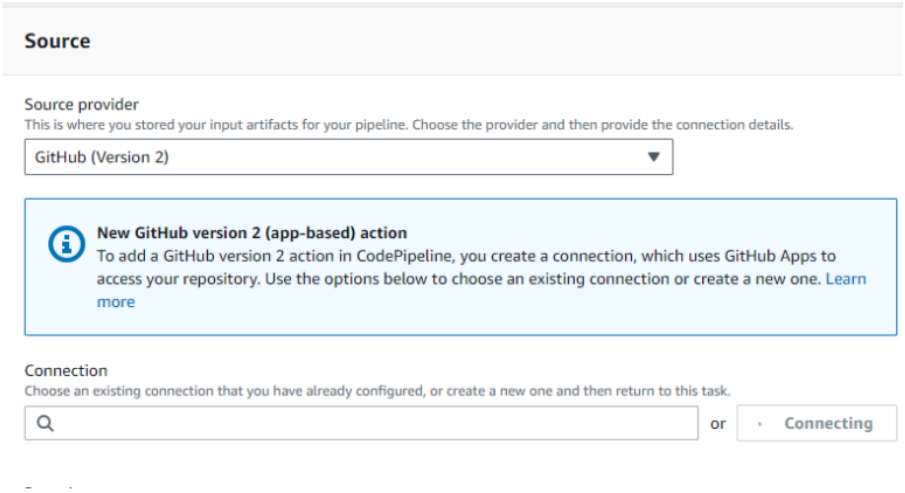


Step 4: Github connection

In this step, we are supposed to create a github connection and add our existing repository over

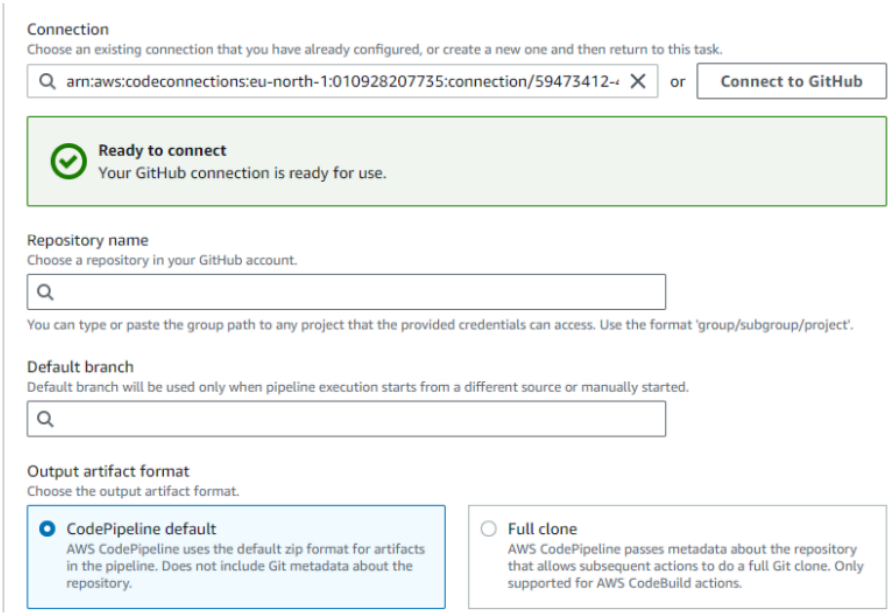
here i.e the one we forked earlier

Weare supposed to enter our github username so as to proceed towards making the connection

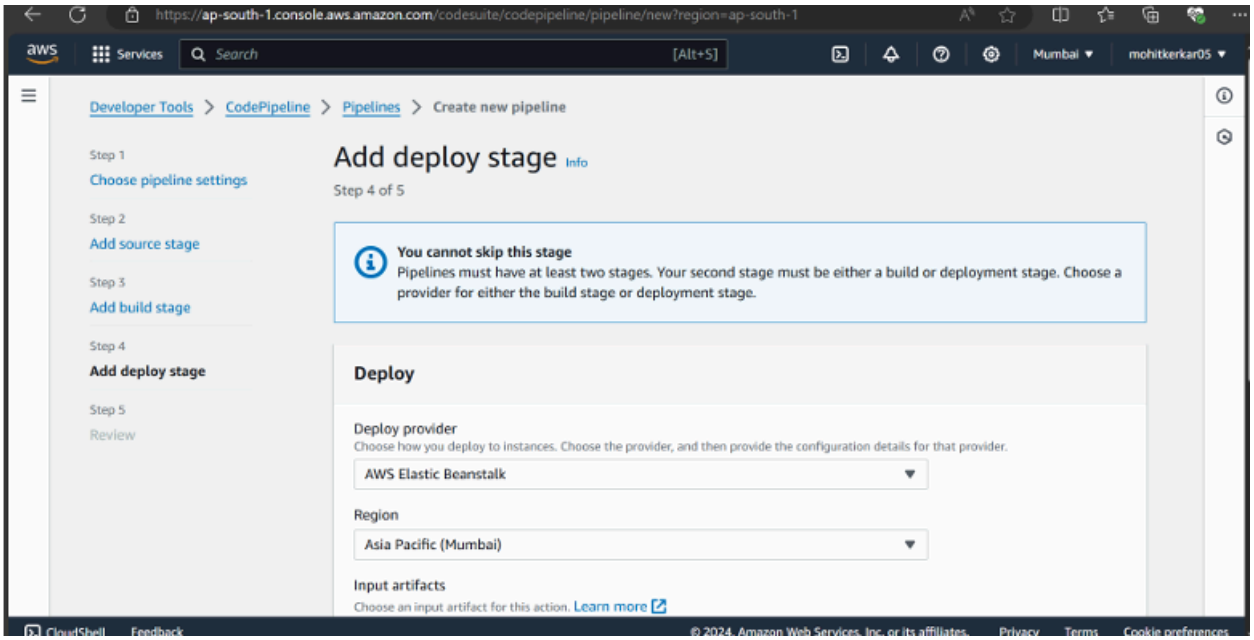


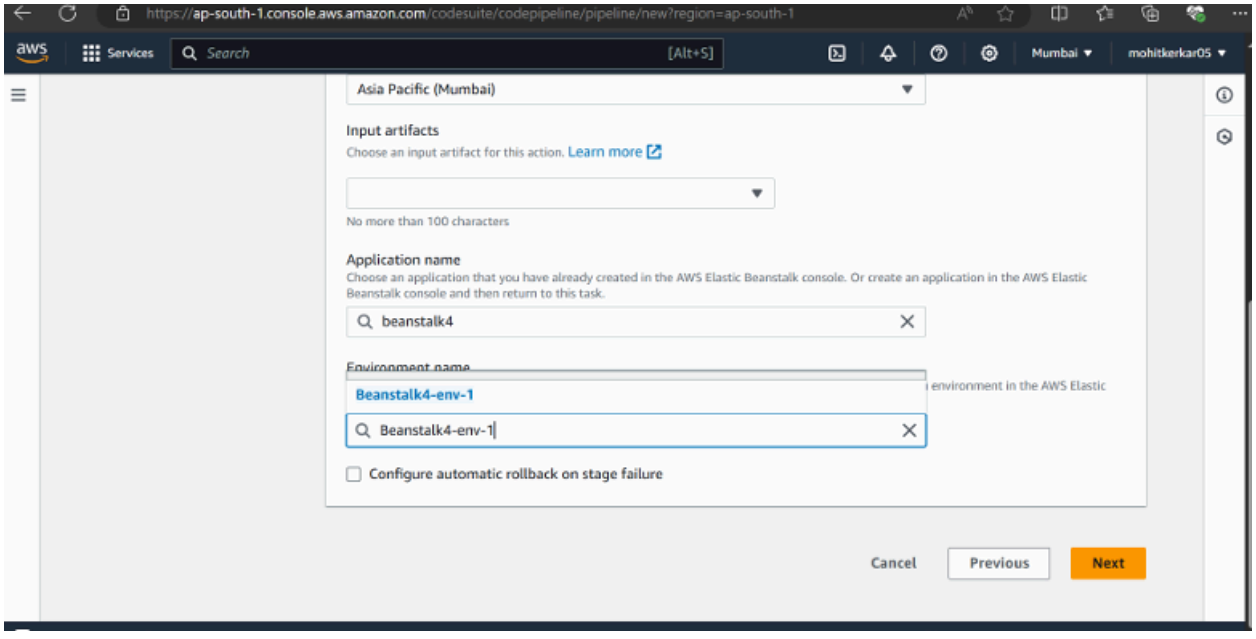
Now to finalize our connection, we are to install an application which connects AWS to our github account and repository.

Post the establishment of the connection, this is the message that is displayed. We can further select the branch of our repository that we want to connect.

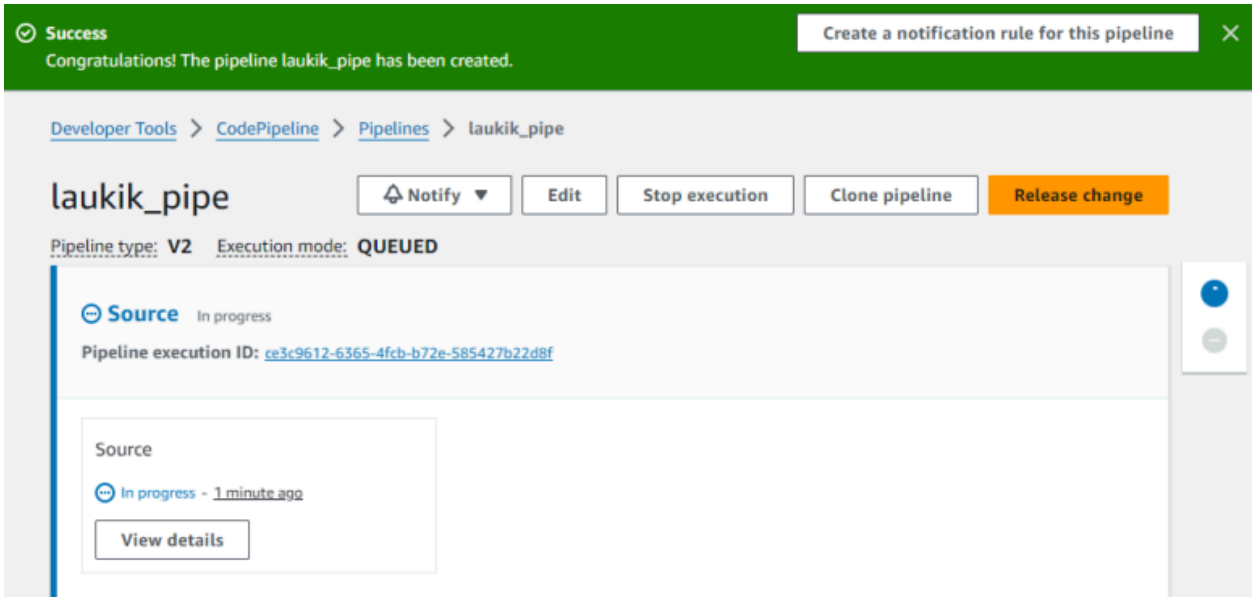


Step 5: Deployment stage:
We are expected to skip the build stage and move towards the deployment step. In the deployment step we are supposed to choose the Elastic Beanstalk application and the environment that we created earlier and proceed with our pipeline creation



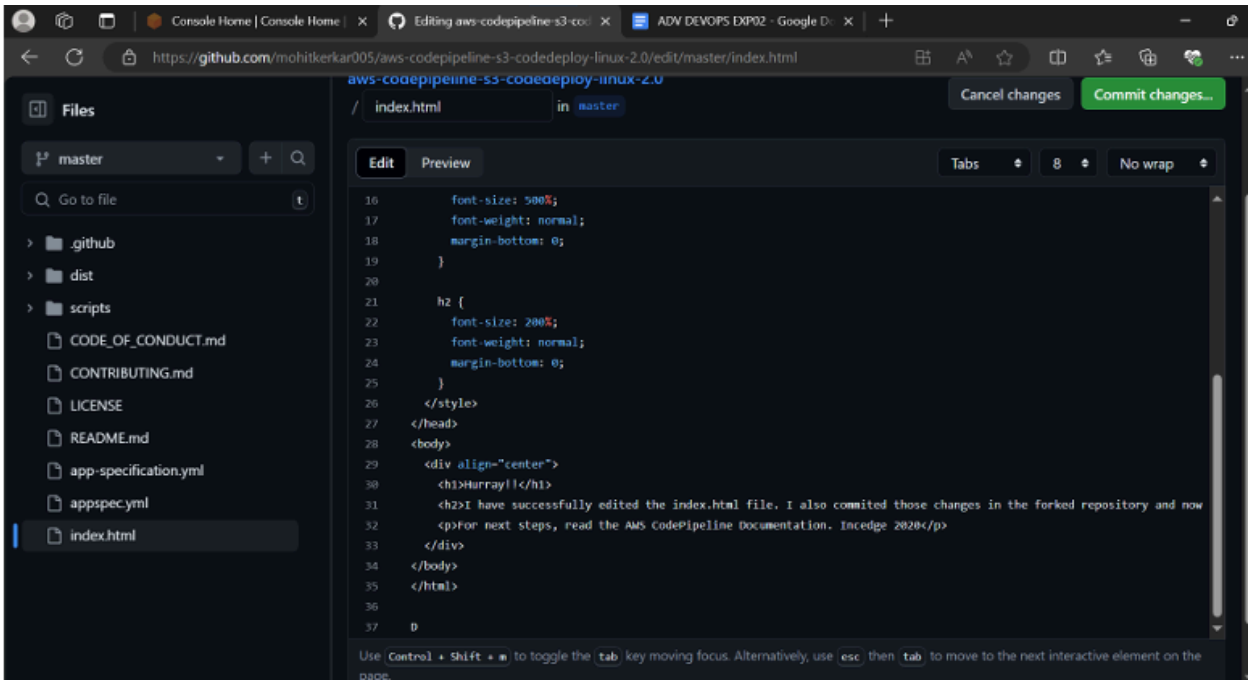


Step 6: Post deployment stage: When all the stages run successfully, this is what is displayed onto the screen. It shows us that our application and our environment have successfully been deployed using a dedicated pipeline created



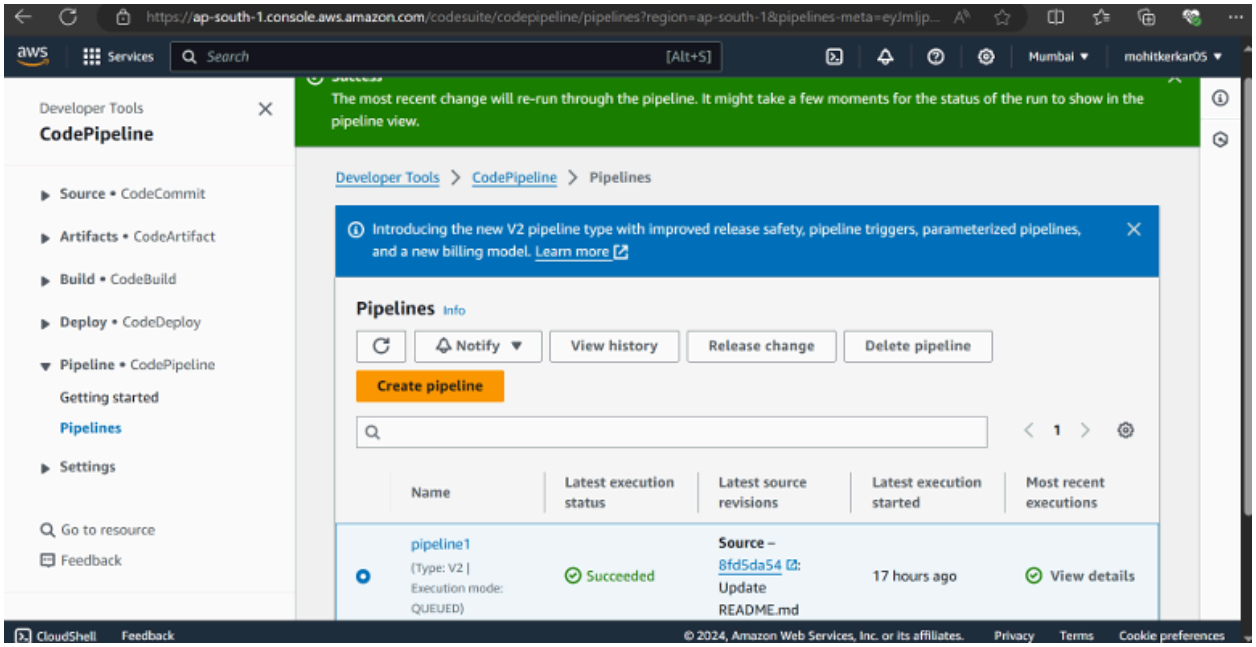
Step 7: Committing changes to your github code

Now, we will go to our forked repository and make some changes to the index.html file. On making the desired changes, we are supposed to commit those changes on our forked repository. Write a good commit message so as to recognize it when it appears on the pipeline.

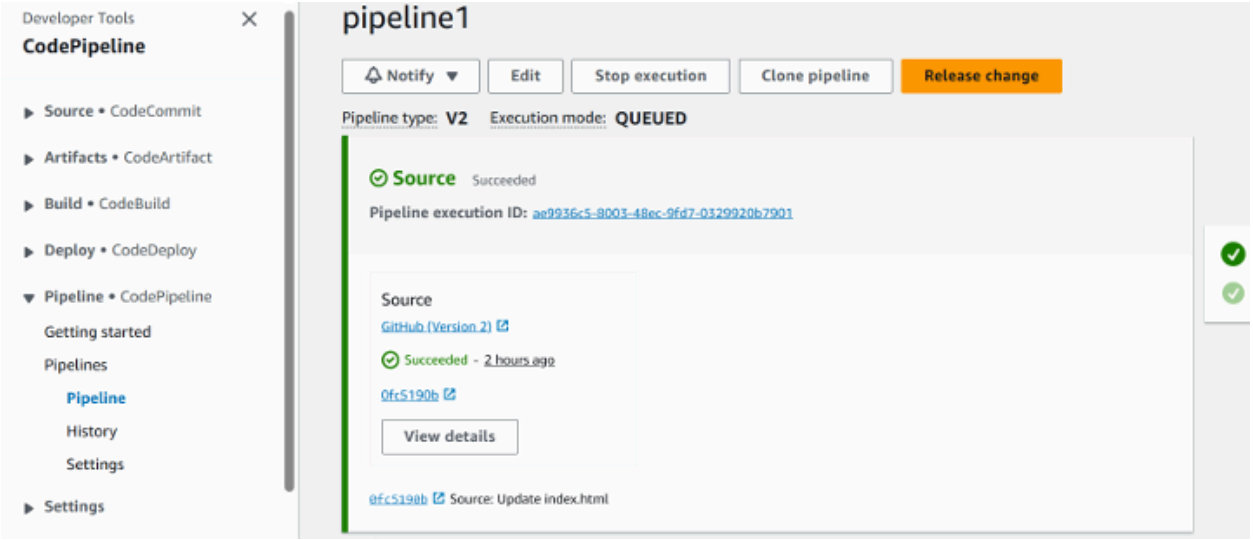


Step 8: Apply the newly made changes in index.html onto our pipeline

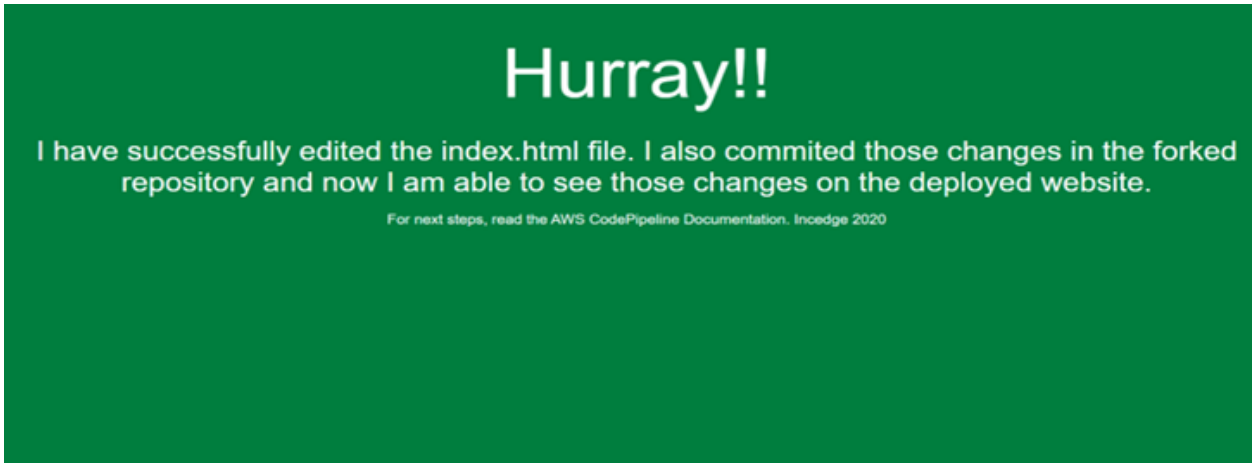
Come back to the Codepipeline section and select the pipeline through which we successfully created and deployed our application. Click on the release change option to apply the latest changes/commits from our github repository to our pipeline



Once the changes have been applied, we see the commit message that we wrote for the latest commit on our repository being reflected on our pipeline. Over here, it would be seen somewhere near the bottom of the image that is attached. “Update index.html” was the latest commit message in the github repository



Step 9: Open the Domain of our Elastic Beanstalk environment
Now, we navigate back to our Elastic Beanstalk environment and open the environment domain of our deployed application
The text in this image is clearly distinguishable from the earlier website’s text meaning that the changes that we made to our code in index.html has successfully been applied to the website that we deployed



Advanced DevOps Lab
Experiment:3

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

Reference: https://www.youtube.com/watch?v=Cz7hSJNq2GU

Theory:

Container-based microservices architectures have profoundly changed the way development and operations teams test and deploy modern software. Containers help companies modernize by making it easier to scale and deploy applications, but containers have also introduced new challenges and more complexity by creating an entirely new infrastructure ecosystem.

Large and small software companies alike are now deploying thousands of container instances daily, and that's a complexity of scale they have to manage. So how do they do it?

Enter the age of Kubernetes.

Originally developed by Google, Kubernetes is an open-source container orchestration platform designed to automate the deployment, scaling, and management of containerized applications. In fact, Kubernetes has established itself as the defacto standard for container orchestration and is the flagship project of the Cloud Native Computing Foundation (CNCF), backed by key players like Google, AWS, Microsoft, IBM, Intel, Cisco, and Red Hat.

Kubernetes makes it easy to deploy and operate applications in a microservice architecture. It does so by creating an abstraction layer on top of a group of hosts so that development teams can deploy their applications and let Kubernetes manage the following activities:

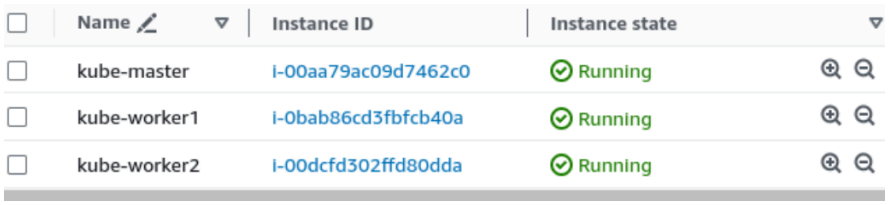
- Controlling resource consumption by application or team
- Evenly spreading application load across a hosting infrastructure
- Automatically load balancing requests across the different instances of an application ● Monitoring resource consumption and resource limits to automatically stop applications from consuming too many resources and restarting the applications again
- Moving an application instance from one host to another if there is a shortage of resources in a host, or if the host dies
- Automatically leveraging additional resources made available when a new host is added to the cluster

- Easily performing canary deployments and rollbacks

Steps:

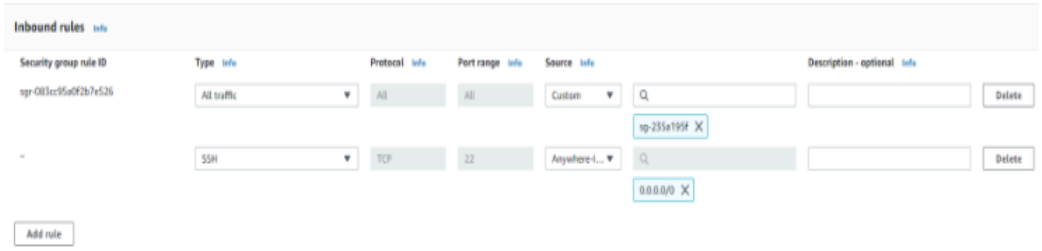
1. Create 3 EC2 Ubuntu Instances on AWS.

   (Name 1 as Master, the other 2 as worker-1 and worker-2)



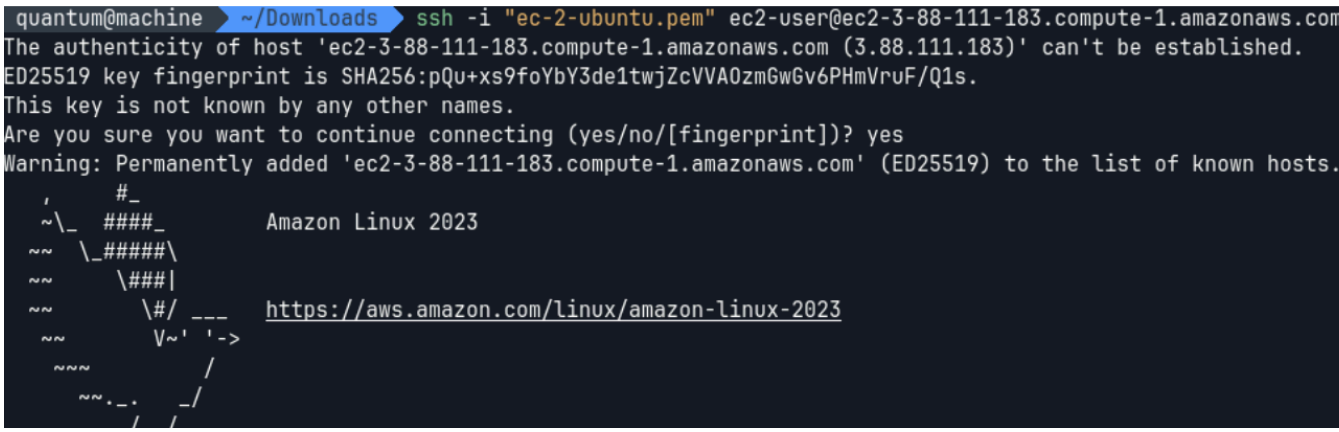
Name	Instance ID	Instance state
kube-master	i-00aa79ac09d7462c0	Running
kube-worker1	i-0bab86cd3fbfcb40a	Running
kube-worker2	i-00dcfd302ffd80dda	Running

2. Edit the Security Group Inbound Rules to allow SSH



3. SSH into all 3 machines

   ssh -i <keyname>.pem ubuntu@<public_ip_address>



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

   Install Docker

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

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

```
ec2-user@ip-172-31-92-18 ~]$ sudo yum install docker -y
ast metadata expiration check: 0:09:56 ago on Wed Sep 11 15:19:39 2024.
ependencies resolved.
```

Then, configure cgroup in a daemon.json file.

```
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
```

Install Kubernetes on all 3 machines

```
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg |
sudo apt-key add -
cat << EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb https://apt.kubernetes.io/ kubernetes-xenial main EOF
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
```

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

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

5. Perform this **ONLY on the Master machine**

Initialize the Kubecluster

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

--ignore-preflight-errors=all

```
[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'
W0914 09:57:27.006694    9935 checks.go:846] detected that the sandbox image "registry.k8s.io/pause:3.8" of the container runtime is inconsistent wit
y 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] Using existing ca certificate authority
[certs] Using existing apiserver certificate and key on disk
[certs] Using existing apiserver-kubelet-client certificate and key on disk
[certs] Using existing front-proxy-ca certificate authority
[certs] Using existing front-proxy-client certificate and key on disk
[certs] Using existing etcd/ca certificate authority
[certs] Using existing etcd/server certificate and key on disk
[certs] Using existing etcd/peer certificate and key on disk
[certs] Using existing etcd/healthcheck-client certificate and key on disk
[certs] Using existing apiserver-etcd-client certificate and key on disk
[certs] Using the existing "sa" key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
```

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

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

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

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

```
[ec2-user@ip-172-31-81-63 docker]$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel
namespace/kube-flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
```

Check the created pod using this command

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

```
watch kubectl get nodes
```

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

```
sudo kubeadm join <ip> --token <token> \
        --discovery-token-ca-cert-hash <hash>
```

Now, notice the changes on the master terminal

```
[root@ip-172-31-85-89 ec2-user]# kubectl get nodes
NAME                          STATUS     ROLES           AGE    VERSION
ip-172-31-85-89.ec2.internal  NotReady   control-plane   119s   v1.26.0
ip-172-31-89-46.ec2.internal  NotReady   <none>          19s    v1.26.0
ip-172-31-94-70.ec2.internal  NotReady   <none>          12s    v1.26.0
[root@ip-172-31-85-89 ec2-user]#
```

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

Conclusion: In this experiment, the goal was to set up and configure a Kubernetes cluster using kubeadm on a Linux environment with yum as the package manager.

Name: Laukik Padgaonkar **Academic Year:2024-2025**

Division: D15C **Roll No: 37**

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

Theory:

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

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

Necessary Requirements:

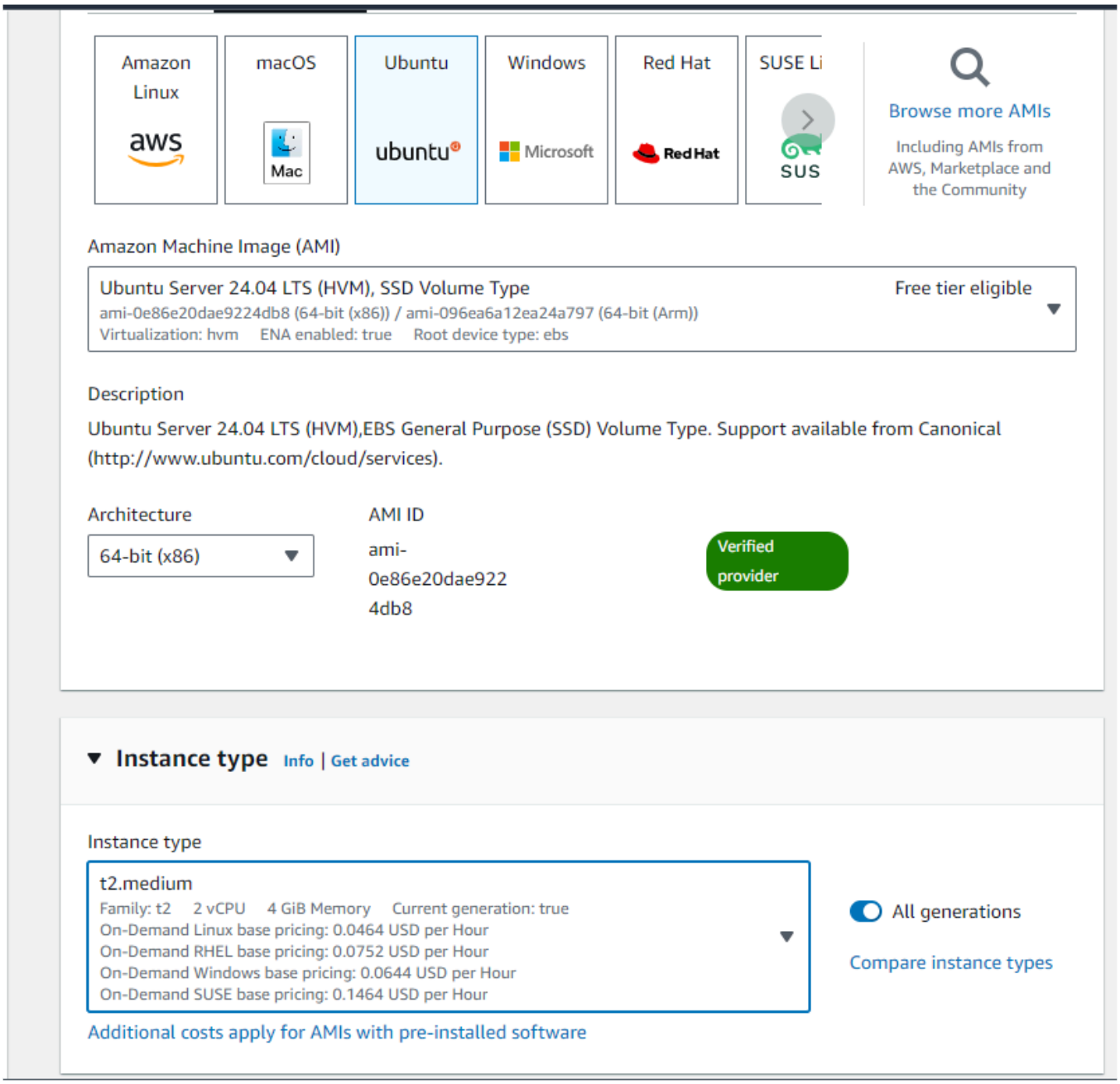
- **EC2 Instance:** The experiment required launching a t2.medium EC2 instance with 2 CPUs, as Kubernetes demands sufficient resources for effective functioning.
- **Minimum Requirements:**
    - **Instance Type:** t2.medium
    - **CPUs:** 2
    - **Memory:** Adequate for container orchestration.

This ensured that the Kubernetes cluster had the necessary resources to function smoothly.

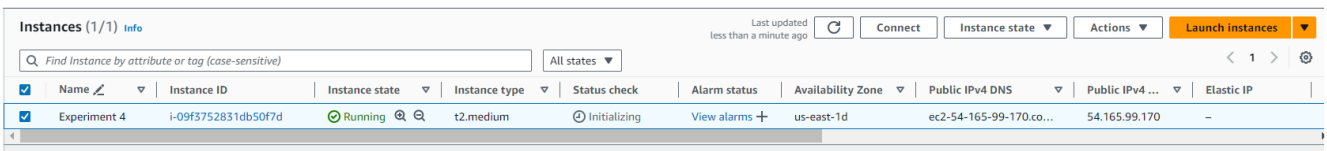
Name: Laukik Padgaonkar **Academic Year:2024-2025**

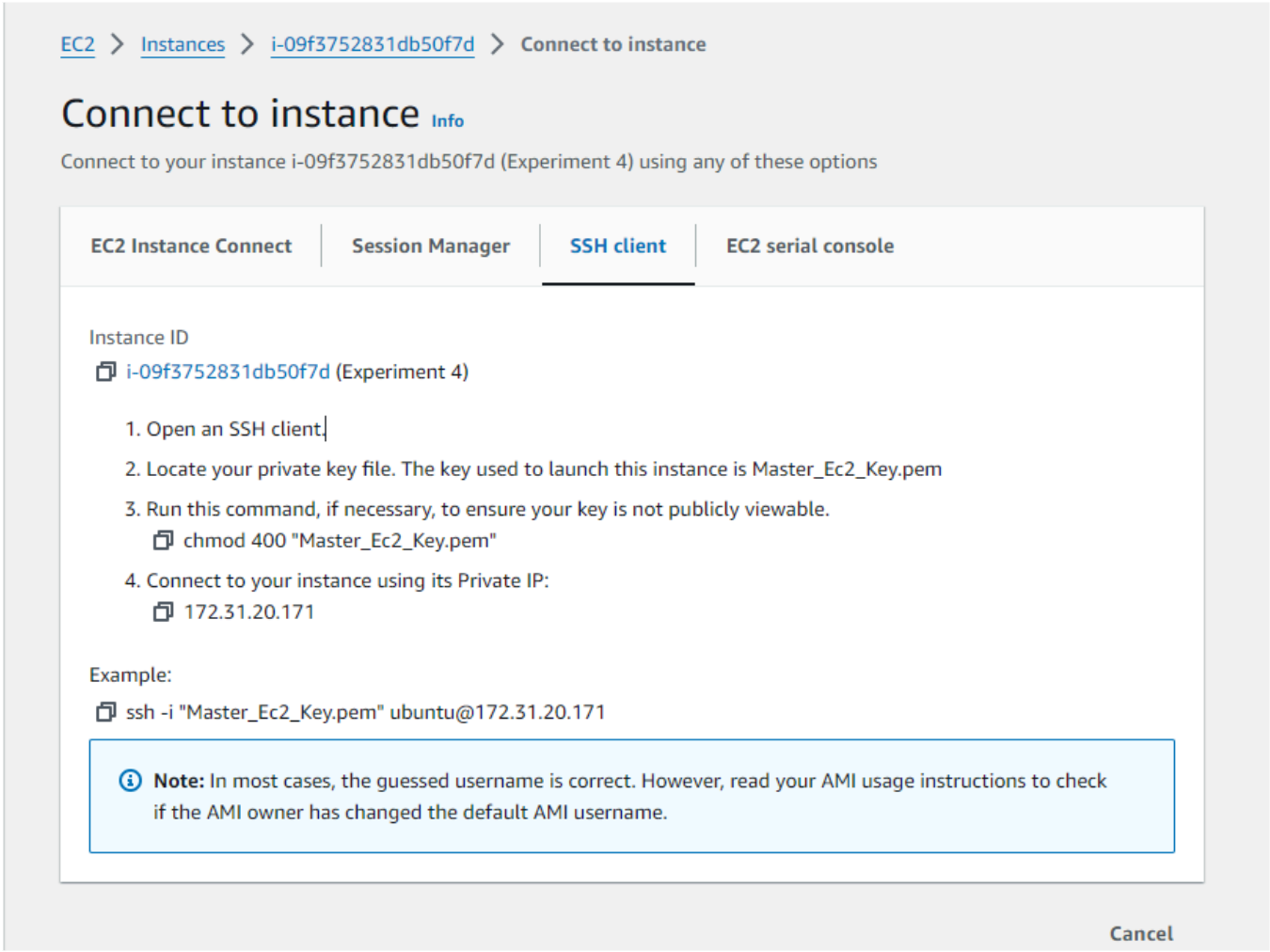
Division: D15C **Roll No: 37**

Step 1: Log in to your AWS personal account and launch a new Ec2 Instance.
Select Ubuntu as AMI and t2.medium as Instance Type and create a key of type RSA with .pem extension and move the downloaded key to the new folder.
Note: A minimum of 2 CPUs are required so Please select t2.medium and do not forget to stop the instance after the experiment because it is not available in the free tier.



Step 2: After creating the instance click on Connect the instance and navigate to SSH Client.





Step 3: Now open the folder in the terminal where our .pem key is stored and paste the Example command (starting with ssh -i …..) in the terminal.(ssh -i "Master_Ec2_Key.pem" ubuntu@ec2-54-196-129-215.compute-1.amazonaws.com)

Step 4: Run the below commands to install and setup Docker.

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-172-31-20-171:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/trusted.gpg.d/docker.gpg > /dev/null
ubuntu@ip-172-31-20-171:~$ 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.
Found existing deb entry in /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-noble.list
Adding deb entry to /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-noble.list
Found existing deb-src entry in /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
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
Get:4 https://download.docker.com/linux/ubuntu noble InRelease [48.8 kB]
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:6 https://download.docker.com/linux/ubuntu noble/stable amd64 Packages [13.8 kB]
Fetched 62.6 kB in 0s (128 kB/s)
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: The key(s) in the keyring /etc/apt/trusted.gpg.d/docker.gpg are ignored as the file has a
n unsupported filetype.
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATION s
ection in apt-key(8) for details.
```

sudo apt-get update

sudo apt-get install -y docker-ce

```
ubuntu@ip-172-31-17-25:~$ sudo apt-get update
sudo apt-get install -y docker-ce
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 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:5 https://download.docker.com/linux/ubuntu noble InRelease
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy tru
ection in apt-key(8) for details.
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-pl
  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-
  libslirp0 pigz slirp4netns
0 upgraded, 10 newly installed, 0 to remove and 133 not upgraded.
Need to get 122 MB of archives.
After this operation, 440 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
```

Name: Laukik Padgaonkar Academic Year:2024-2025

Division: D15C Roll No: 37

```
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 slirp4netns amd64 1.2.1-1build2 [34.9 kB]
Get:6 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-buildx-plugin amd64 0.16.2-1~ubuntu.24.04~noble [29.9 MB]
Get:7 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-ce-cli amd64 5:27.2.1-1~ubuntu.24.04~noble [15.0 MB]
Get:8 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-ce amd64 5:27.2.1-1~ubuntu.24.04~noble [25.6 MB]
Get:9 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-ce-rootless-extras amd64 5:27.2.1-1~ubuntu.24.04~noble [9571 kB]
Get:10 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-compose-plugin amd64 2.29.2-1~ubuntu.24.04~noble [12.5 MB]
Fetched 122 MB in 2s (71.3 MB/s)
Selecting previously unselected package pigz.
(Reading database ... 67741 files and directories currently installed.)
Preparing to unpack .../0-pigz_2.8-1_amd64.deb ...
Unpacking pigz (2.8-1) ...
Selecting previously unselected package containerd.io.
Preparing to unpack .../1-containerd.io_1.7.22-1_amd64.deb ...
Unpacking containerd.io (1.7.22-1) ...
Selecting previously unselected package docker-buildx-plugin.
Preparing to unpack .../2-docker-buildx-plugin_0.16.2-1~ubuntu.24.04~noble_amd64.deb ...
Unpacking docker-buildx-plugin (0.16.2-1~ubuntu.24.04~noble) ...
Selecting previously unselected package docker-ce-cli.
Preparing to unpack .../3-docker-ce-cli_5%3a27.2.1-1~ubuntu.24.04~noble_amd64.deb ...
Unpacking docker-ce-cli (5:27.2.1-1~ubuntu.24.04~noble) ...
Selecting previously unselected package docker-ce.
Preparing to unpack .../4-docker-ce_5%3a27.2.1-1~ubuntu.24.04~noble_amd64.deb ...
Unpacking docker-ce (5:27.2.1-1~ubuntu.24.04~noble) ...
Selecting previously unselected package docker-ce-rootless-extras.
Preparing to unpack .../5-docker-ce-rootless-extras_5%3a27.2.1-1~ubuntu.24.04~noble_amd64.deb ...
Unpacking docker-ce-rootless-extras (5:27.2.1-1~ubuntu.24.04~noble) ...
Selecting previously unselected package docker-compose-plugin.
Preparing to unpack .../6-docker-compose-plugin_2.29.2-1~ubuntu.24.04~noble_amd64.deb ...
Unpacking docker-compose-plugin (2.29.2-1~ubuntu.24.04~noble) ...
Selecting previously unselected package libltdl7:amd64.
Preparing to unpack .../7-libltdl7_2.4.7-7build1_amd64.deb ...
Unpacking libltdl7:amd64 (2.4.7-7build1) ...
Selecting previously unselected package libslirp0:amd64.
Preparing to unpack .../8-libslirp0_4.7.0-1ubuntu3_amd64.deb ...
Unpacking libslirp0:amd64 (4.7.0-1ubuntu3) ...
Selecting previously unselected package slirp4netns.
Preparing to unpack .../9-slirp4netns_1.2.1-1build2_amd64.deb ...
Unpacking slirp4netns (1.2.1-1build2) ...
Setting up docker-buildx-plugin (0.16.2-1~ubuntu.24.04~noble) ...
Setting up containerd.io (1.7.22-1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /usr/lib/systemd/system/containerd.service.
```

```
Unpacking slirp4netns (1.2.1-1build2) ...
Setting up docker-buildx-plugin (0.16.2-1~ubuntu.24.04~noble) ...
Setting up containerd.io (1.7.22-1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /usr/lib/systemd/system/containerd.service.
Setting up docker-compose-plugin (2.29.2-1~ubuntu.24.04~noble) ...
Setting up libltdl7:amd64 (2.4.7-7build1) ...
Setting up docker-ce-cli (5:27.2.1-1~ubuntu.24.04~noble) ...
Setting up libslirp0:amd64 (4.7.0-1ubuntu3) ...
Setting up pigz (2.8-1) ...
Setting up docker-ce-rootless-extras (5:27.2.1-1~ubuntu.24.04~noble) ...
Setting up slirp4netns (1.2.1-1build2) ...
Setting up docker-ce (5:27.2.1-1~ubuntu.24.04~noble) ...
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /usr/lib/systemd/system/docker.socket.
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for libc-bin (2.39-0ubuntu8.2) ...
Scanning processes...
Scanning linux images...

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.
```

sudo mkdir -p /etc/docker

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

```
ubuntu@ip-172-31-20-171:~$ sudo mkdir -p /etc/docker
ubuntu@ip-172-31-20-171:~$ cat <<EOF | sudo tee /etc/docker/daemon.json
{
    "exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
{
    "exec-opts": ["native.cgroupdriver=systemd"]
}
```

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

```
ubuntu@ip-172-31-17-25:~$ sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart 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
```

Step 5: Run the below command to install Kubernets.
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-172-31-20-171:~$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
ubuntu@ip-172-31-20-171:~$ 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
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl

```
ubuntu@ip-172-31-17-25:~$ sudo apt-get update-25:~$ sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
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 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:5 https://download.docker.com/linux/ubuntu noble InRelease
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  I
Get:7 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  P
Fetched 6051 B in 1s (10.4 kB/s)
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trus
ection in apt-key(8) for details.
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 133 not upgraded.
Need to get 87.4 MB of archives.
After this operation, 314 MB of additional disk space will be used.
```

```
Unpacking cri-tools (1.31.1-1.1) ...
Selecting previously unselected package kubeadm.
Preparing to unpack .../2-kubeadm_1.31.1-1.1_amd64.deb ...
Unpacking kubeadm (1.31.1-1.1) ...
Selecting previously unselected package kubectl.
Preparing to unpack .../3-kubectl_1.31.1-1.1_amd64.deb ...
Unpacking kubectl (1.31.1-1.1) ...
Selecting previously unselected package kubernetes-cni.
Preparing to unpack .../4-kubernetes-cni_1.5.1-1.1_amd64.deb ...
Unpacking kubernetes-cni (1.5.1-1.1) ...
Selecting previously unselected package kubelet.
Preparing to unpack .../5-kubelet_1.31.1-1.1_amd64.deb ...
Unpacking kubelet (1.31.1-1.1) ...
Setting up conntrack (1:1.4.8-1ubuntu1) ...
Setting up kubectl (1.31.1-1.1) ...
Setting up cri-tools (1.31.1-1.1) ...
Setting up kubernetes-cni (1.5.1-1.1) ...
Setting up kubeadm (1.31.1-1.1) ...
Setting up kubelet (1.31.1-1.1) ...
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.

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

```
ubuntu@ip-172-31-17-25:~$ sudo systemctl restart containerd
sudo systemctl enable containerd
sudo systemctl status containerd
● containerd.service - containerd container runtime
     Loaded: loaded (/usr/lib/systemd/system/containerd.service; enabled; preset: enabled)
     Active: active (running) since Sun 2024-09-15 17:38:22 UTC; 272ms ago
       Docs: https://containerd.io
   Main PID: 9670 (containerd)
      Tasks: 7
     Memory: 13.4M (peak: 13.7M)
        CPU: 66ms
     CGroup: /system.slice/containerd.service
             └─9670 /usr/bin/containerd

Sep 15 17:38:22 ip-172-31-17-25 containerd[9670]: time="2024-09-15T17:38:22.267775291Z" level=info msg=serving... address=/run/containerd/containerd.sock.ttrpc
Sep 15 17:38:22 ip-172-31-17-25 containerd[9670]: time="2024-09-15T17:38:22.268641365Z" level=info msg=serving... address=/run/containerd/containerd.sock
Sep 15 17:38:22 ip-172-31-17-25 containerd[9670]: time="2024-09-15T17:38:22.268711553Z" level=info msg="Start subscribing containerd event"
Sep 15 17:38:22 ip-172-31-17-25 containerd[9670]: time="2024-09-15T17:38:22.268738456Z" level=info msg="Start recovering state"
Sep 15 17:38:22 ip-172-31-17-25 containerd[9670]: time="2024-09-15T17:38:22.268779583Z" level=info msg="Start event monitor"
Sep 15 17:38:22 ip-172-31-17-25 containerd[9670]: time="2024-09-15T17:38:22.268794095Z" level=info msg="Start snapshots syncer"
Sep 15 17:38:22 ip-172-31-17-25 containerd[9670]: time="2024-09-15T17:38:22.268801586Z" level=info msg="Start cni network conf syncer for default"
Sep 15 17:38:22 ip-172-31-17-25 containerd[9670]: time="2024-09-15T17:38:22.268807177Z" level=info msg="Start streaming server"
Sep 15 17:38:22 ip-172-31-17-25 containerd[9670]: time="2024-09-15T17:38:22.268848440Z" level=info msg="containerd successfully booted in 0.027
```

Name: Laukik Padgaonkar Academic Year:2024-2025

Division: D15C Roll No: 37

Now We have got an error.
So we have to perform some additional commands as follow.

sudo apt-get install -y containerd

```
To see the stack trace of this error execute with --v=5 or higher   ubuntu@ip-172-31-20-171:~$ 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 130 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 (74.5 MB/s)
(Reading database ... 68068 files and directories currently installed.)
Removing docker-ce (5:27.2.1-1~ubuntu.24.04~noble) ...
Removing containerd.io (1.7.22-1) ...
Selecting previously unselected package runc.
(Reading database ... 68048 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.

No containers need to be restarted.
```

```
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.
```

Name: Laukik Padgaonkar Academic Year:2024-2025

Division: D15C Roll No: 37

sudo mkdir -p /etc/containerd

```
ubuntu@ip-172-31-20-171:~$ 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 = ""
  tcp_tls_key = ""
  uid = 0

[metrics]
  address = ""
  grpc_histogram = false

[plugins]

  [plugins."io.containerd.gc.v1.scheduler"]
    deletion_threshold = 0
```

...

```
[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
```

Name: Laukik Padgaonkar **Academic Year:2024-2025**

Division: D15C **Roll No: 37**

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

```
ubuntu@ip-172-31-20-171:~$ sudo systemctl restart containerd
sudo systemctl enable containerd
ubuntu@ip-172-31-20-171:~$ sudo systemctl status containerd
● containerd.service - containerd container runtime
     Loaded: loaded (/usr/lib/systemd/system/containerd.service; en>[redacted]
     Active: active (running) since Sun 2024-09-15 07:49:23 UTC; 5s>[redacted]
       Docs: https://containerd.io
   Main PID: 8398 (containerd)
      Tasks: 7
     Memory: 13.5M (peak: 14.0M)
        CPU: 70ms
     CGroup: /system.slice/containerd.service
             └─8398 /usr/bin/containerd

Sep 15 07:49:23 ip-172-31-20-171 containerd[8398]: time="2024-09-15>[redacted]
Sep 15 07:49:23 ip-172-31-20-171 containerd[8398]: time="2024-09-15>[redacted]
Sep 15 07:49:23 ip-172-31-20-171 containerd[8398]: time="2024-09-15>[redacted]
Sep 15 07:49:23 ip-172-31-20-171 containerd[8398]: time="2024-09-15>[redacted]
Sep 15 07:49:23 ip-172-31-20-171 containerd[8398]: time="2024-09-15>[redacted]
Sep 15 07:49:23 ip-172-31-20-171 containerd[8398]: time="2024-09-15>[redacted]
Sep 15 07:49:23 ip-172-31-20-171 containerd[8398]: time="2024-09-15>[redacted]
Sep 15 07:49:23 ip-172-31-20-171 containerd[8398]: time="2024-09-15>[redacted]
Sep 15 07:49:23 ip-172-31-20-171 systemd[1]: Started containerd.ser>[redacted]
Sep 15 07:49:23 ip-172-31-20-171 containerd[8398]: time="2024-09-15>[redacted]
```

sudo apt-get install -y socat

```
ubuntu@ip-172-31-17-25:~$ sudo apt-get install -y socat
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 libl
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  socat
0 upgraded, 1 newly installed, 0 to remove and 133 not upgraded.
Need to get 374 kB of archives.
After this operation, 1649 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 socat amd64 1.8.0.(
Fetched 374 kB in 0s (16.1 MB/s)
Selecting previously unselected package socat.
(Reading database ... 68108 files and directories currently installed.)
Preparing to unpack .../socat_1.8.0.0-4build3_amd64.deb ...
Unpacking socat (1.8.0.0-4build3) ...
Setting up socat (1.8.0.0-4build3) ...
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.

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.
```

Name: Laukik Padgaonkar **Academic Year:2024-2025**

Division: D15C **Roll No: 37**

Step 6: Initialize the Kubecluster

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

```
ubuntu@ip-172-31-17-25:~$ 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'
W0915 17:42:02.713394    9994 checks.go:846] detected that the sandbox image "registry.k8s.io/pause:3.8" of the container runtime is in
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-172-31-17-25 kubernetes kubernetes.default kubernetes.default.svc kubernetes
local] and IPs [10.96.0.1 172.31.17.25]
[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-172-31-17-25 localhost] and IPs [172.31.17.25 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [ip-172-31-17-25 localhost] and IPs [172.31.17.25 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
```

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

mkdir -p $HOME/.kube

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

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

```
ubuntu@ip-172-31-20-171:~$  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

```
[mark-control-plane] Marking the node ip-172-31-20-171 as control-plane by adding the taints [node-role.kubernetes.io/control-plane:NoSchedule]
[bootstrap-token] Using token: 7acddu.inheshzwxti0372v
[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC Roles
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to get nodes
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get long term certificate credentials
[bootstrap-token] Configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Bootstrap Token
[bootstrap-token] Configured RBAC rules to allow certificate rotation for all node client certificates in the cluster
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotatable kubelet client certificate and key
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!

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

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

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

  export KUBECONFIG=/etc/kubernetes/admin.conf

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

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

kubeadm join 172.31.20.171:6443 --token 7acddu.inheshzwxti0372v \
        --discovery-token-ca-cert-hash sha256:aed5faf97bac361d1bb7f33a89fb05d2bb28c7fc065024eac2302a734c330a36
```

Name: Laukik Padgaonkar Academic Year:2024-2025

Division: D15C Roll No: 37

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

```
ubuntu@ip-172-31-17-25:~$ 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
ubuntu@ip-172-31-17-25:~$
```

Step 7: Now that the cluster is up and running, we can deploy our nginx server on this cluster.Apply this deployment file using this command to create a deployment

kubectl apply -f https://k8s.io/examples/application/deployment.yaml

```
ubuntu@ip-172-31-17-25:~$ kubectl apply -f https://k8s.io/examples/application/deployment.yaml
deployment.apps/nginx-deployment created
```

kubectl get pods

```
ubuntu@ip-172-31-17-25:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-fbphm    0/1     Pending   0          37s
nginx-deployment-d556bf558-ljnqb    0/1     Pending   0          37s
```

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

```
ubuntu@ip-172-31-20-171:~$ POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
ubuntu@ip-172-31-20-171:~$ kubectl port-forward $POD_NAME 8080:80
error: unable to forward port because pod is not running. Current status=Pending
```

Note : We have faced an error as pod status is pending so make it running run below commands then again run above 2 commands.

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

kubectl get nodes

```
ubuntu@ip-172-31-20-171:~$ kubectl taint nodes --all node-role.kubernetes.io/control-plane-
node/ip-172-31-20-171 untainted
ubuntu@ip-172-31-20-171:~$ kubectl get nodes
NAME               STATUS   ROLES           AGE     VERSION
ip-172-31-20-171   Ready    control-plane   5m23s   v1.31.1
```

Name: Laukik Padgaonkar Academic Year:2024-2025

Division: D15C Roll No: 37

kubectl get pods

```
ubuntu@ip-172-31-17-25:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-fbphm   1/1     Running   0          14m
nginx-deployment-d556bf558-ljnqb   1/1     Running   0          14m
ubuntu@ip-172-31-17-25:~$
```

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

```
ubuntu@ip-172-31-20-171:~$ POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
kubectl port-forward $POD_NAME 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
Handling connection for 8080
```

Step 8: Verify your deployment
Open up a new terminal and ssh to your EC2 instance.
Then, use this curl command to check if the Nginx server is running.

curl --head http://127.0.0.1:8080

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

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

PS C:\Users\bhush\one drive 2\OneDrive\Desktop\New folder (4)> ssh -i "Master_Ec2_Key.pem" ubuntu@ec2-54-196-129-215.compute-1.amazonaws.com
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 Sun Sep 15 07:58:53 UTC 2024

  System load:  0.15              Processes:             152
  Usage of /:   55.3% of 6.71GB   Users logged in:       1
  Memory usage: 20%               IPv4 address for enX0: 172.31.20.171
  Swap usage:   0%

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

   https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

132 updates can be applied immediately.
38 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

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


Last login: Sun Sep 15 07:27:23 2024 from 152.58.2.47
```

Name: Laukik Padgaonkar
Academic Year:2024-2025
Division: D15C
Roll No: 37

```
ubuntu@ip-172-31-17-25:~$ curl --head http://127.0.0.1:8080
HTTP/1.1 200 OK
Server: nginx/1.14.2
Date: Sun, 15 Sep 2024 18:03:53 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
```

If the response is 200 OK and you can see the Nginx server name, your deployment was successful.
We have successfully deployed our Nginx server on our EC2 instance.

Conclusion:
In this experiment, we successfully installed Kubernetes on an EC2 instance and deployed an Nginx server using Kubectl commands. During the process, we encountered two main errors: the Kubernetes pod was initially in a pending state, which was resolved by removing the control-plane taint using kubectl taint nodes --all, and we also faced an issue with the missing containerd runtime, which was fixed by installing and starting containerd. We used a **t2.medium EC2 instance with 2 CPUs** to meet the necessary resource requirements for the Kubernetes setup and deployment.

Practical Manual on Terraform

Terraform is an infrastructure as code (IaC) tool that allows you to build, change, and version infrastructure safely and efficiently. This includes low-level components such as compute instances, storage, and networking, as well as high level components such as DNS entries, SaaS features, etc.

Terraform can manage infrastructure on multiple cloud platforms. Terraform's state allows you to track resource changes throughout your deployments. You can commit your configurations to version control to safely collaborate on infrastructure. Terraform plugins called providers let Terraform interact with cloud platform and other services via their application programming interfaces (APIs).

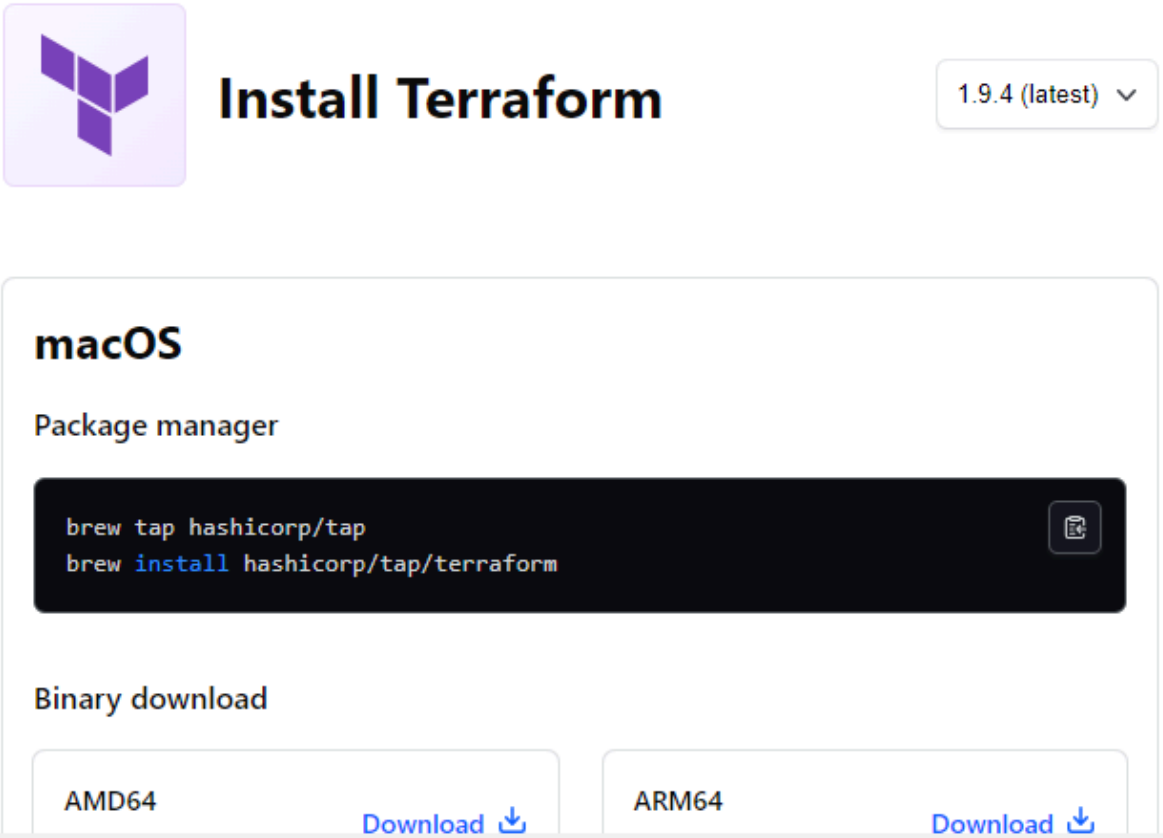
A) Installation and Configuration of Terraform in Windows

Step 1: Download terraform

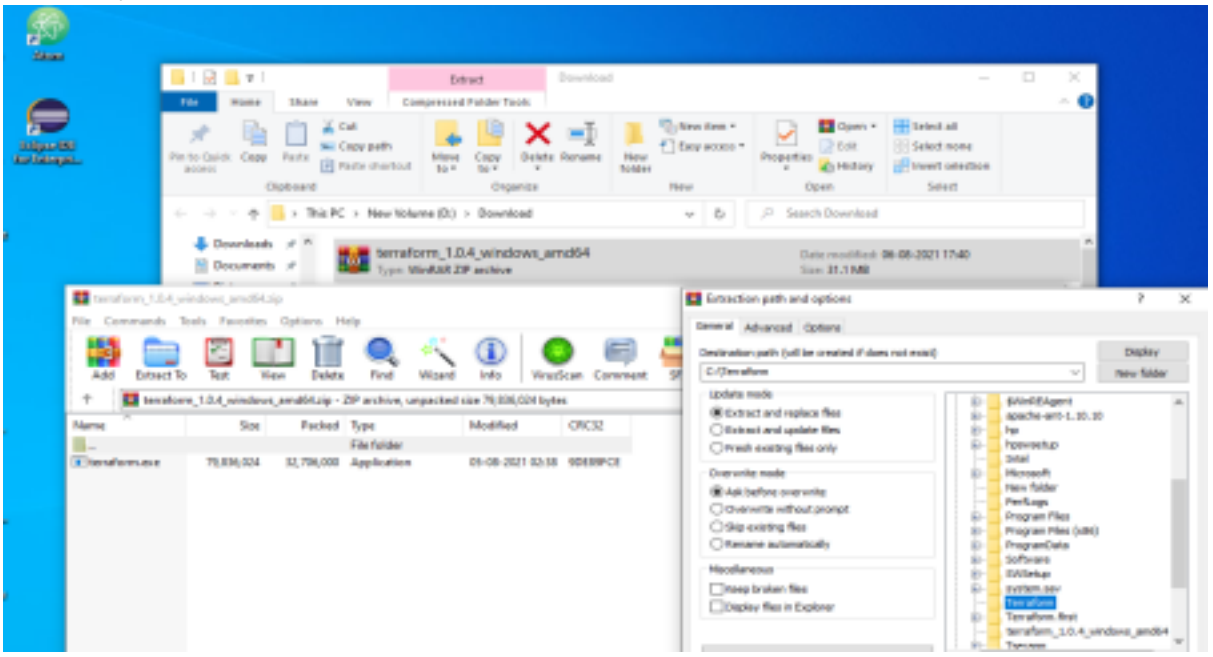
To install Terraform, First Download the Terraform Cli Utility for windows from terraforms official website

website:https://www.terraform.io/downloads.html

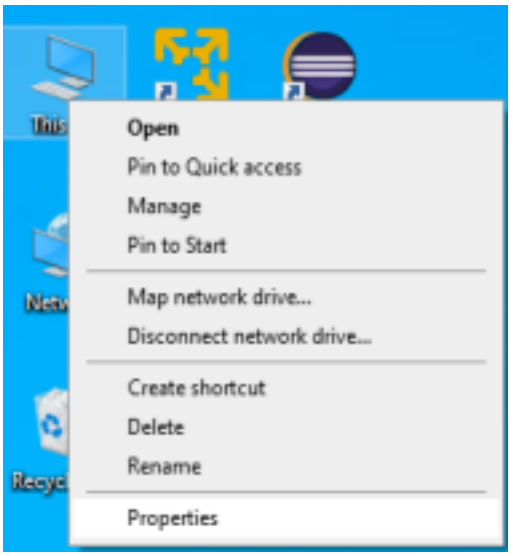
Select the Operating System Windows followed by either 32bit or 64 bit based on your OS type.

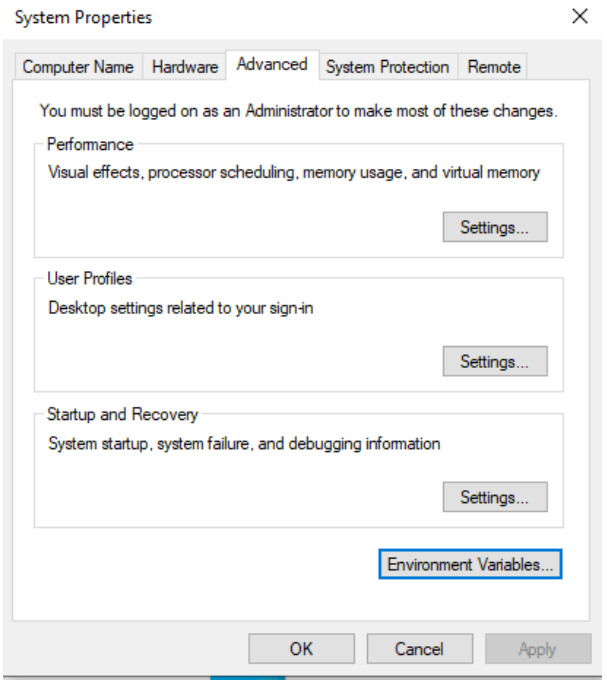


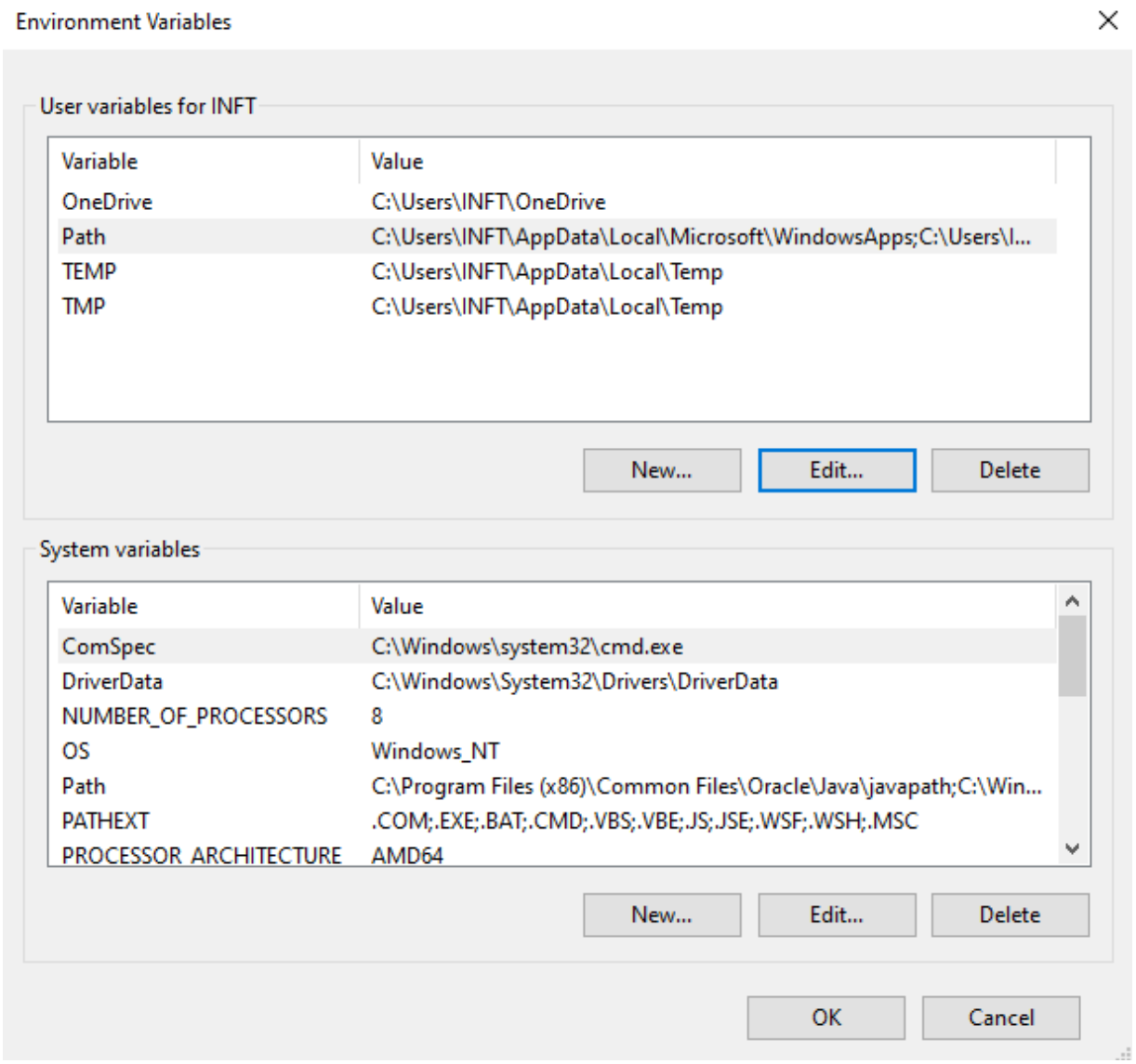
Step 2: Extract the downloaded setup file Terraform.exe in C:\Terraform directory

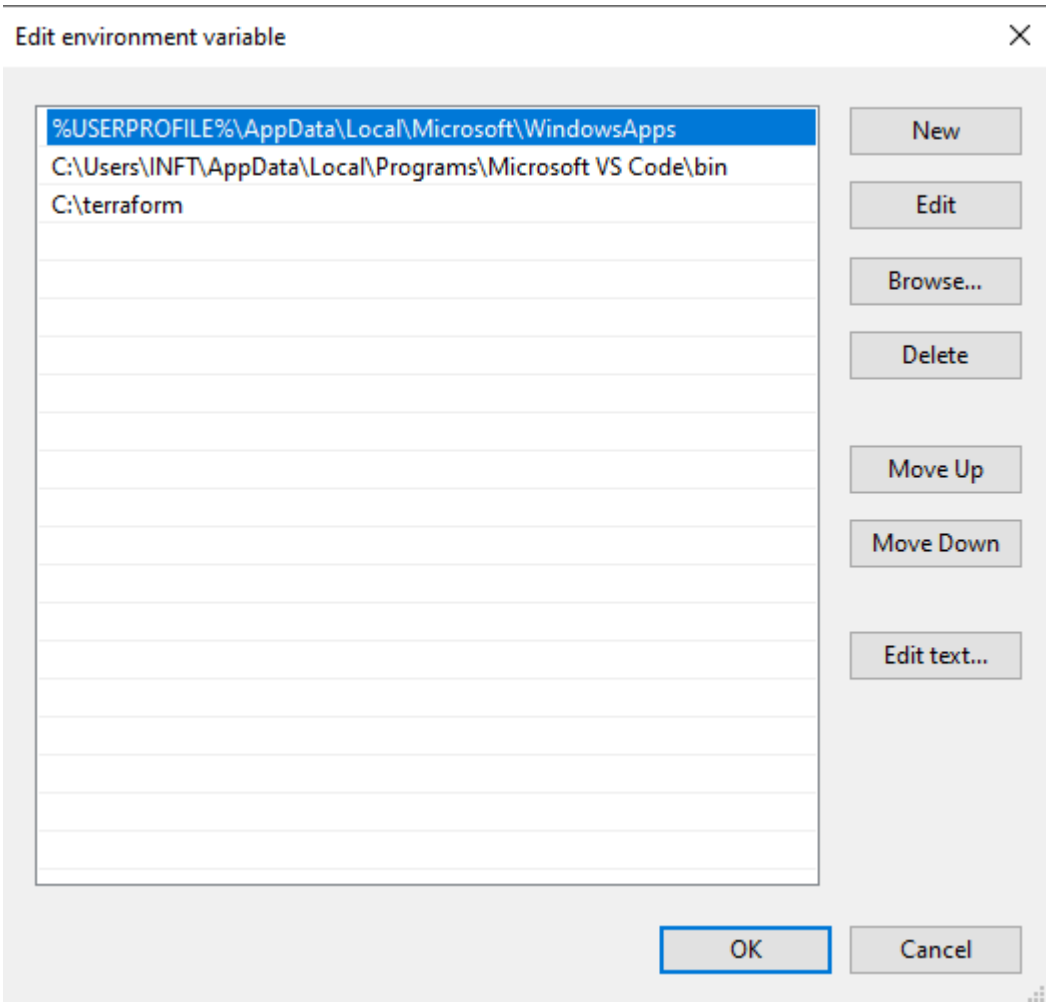


Step 3: Set the System path for Terraform in Environment Variables

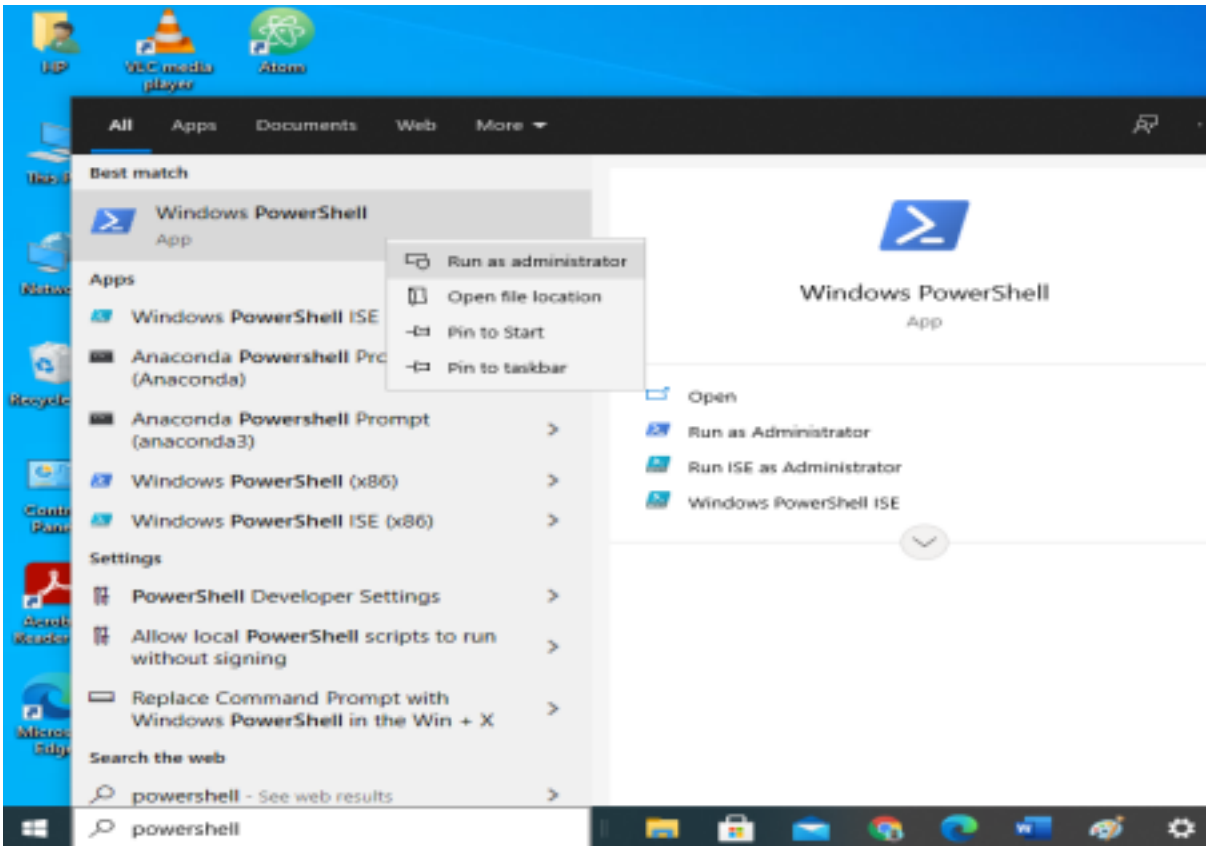




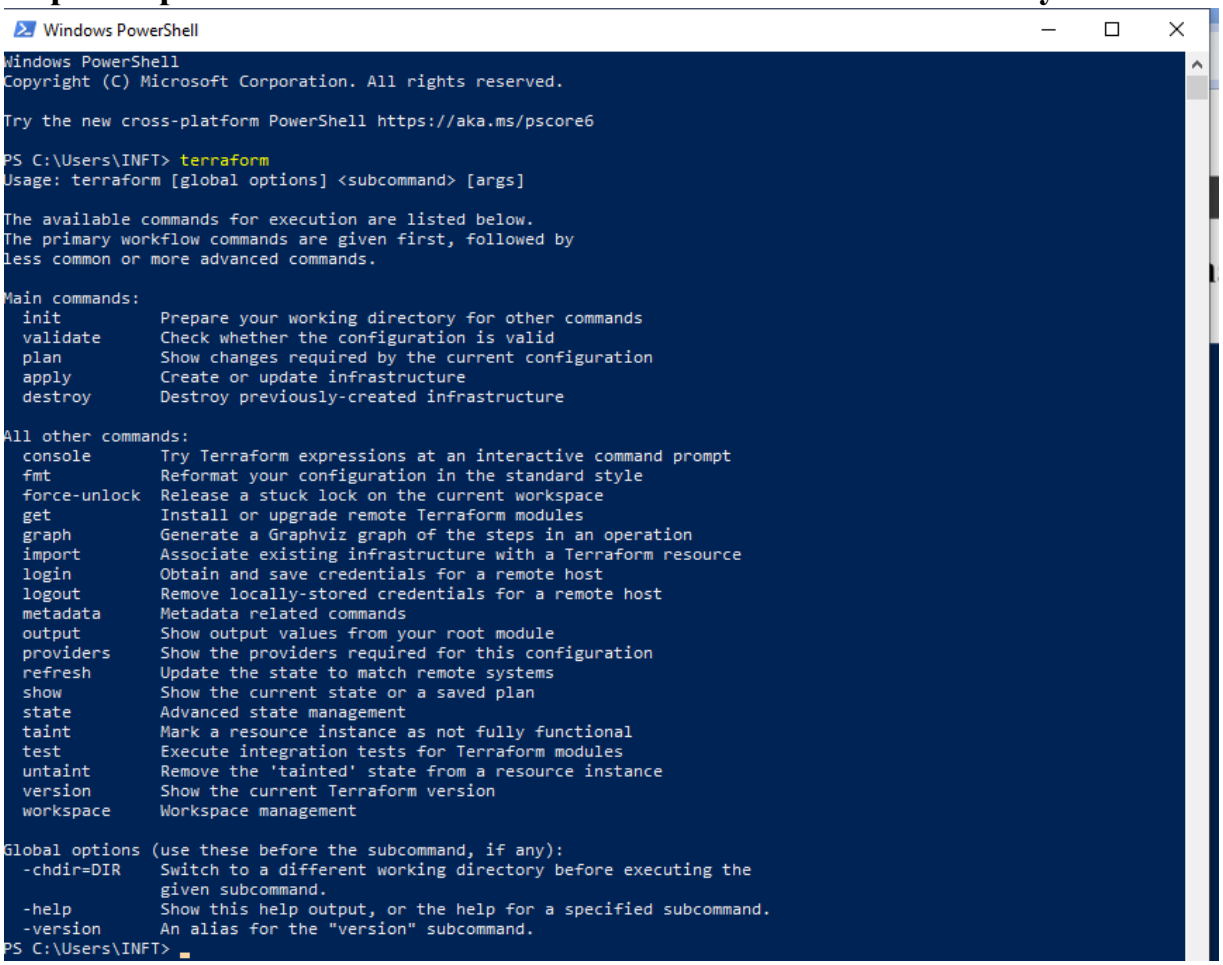

Environment Variables
User variables for INFT
Variable
Value
OneDrive
C:\Users\INFT\OneDrive
Path
C:\Users\INFT\AppData\Local\Microsoft\WindowsApps;C:\Users\I...
TEMP
C:\Users\INFT\AppData\Local\Temp
TMP
C:\Users\INFT\AppData\Local\Temp
New...
Edit...
Delete
System variables
Variable
Value
ComSpec
C:\Windows\system32\cmd.exe
DriverData
C:\Windows\System32\Drivers\DriverData
NUMBER_OF_PROCESSORS
8
OS
Windows_NT
Path
C:\Program Files (x86)\Common Files\Oracle\Java\javapath;C:\Win...
PATHEXT
.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE
AMD64
New...
Edit...
Delete
OK
Cancel



Step 4: Open PowerShell with Admin Access



Step 5 : Open Terraform in PowerShell and check its functionality



Note: If any error comes, then please recheck or set the path of Terraform in Environment variable again.

Experiment No.: 6

Implementation:

A. Creating docker image using terraform

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

Step 1: Check the docker functionality

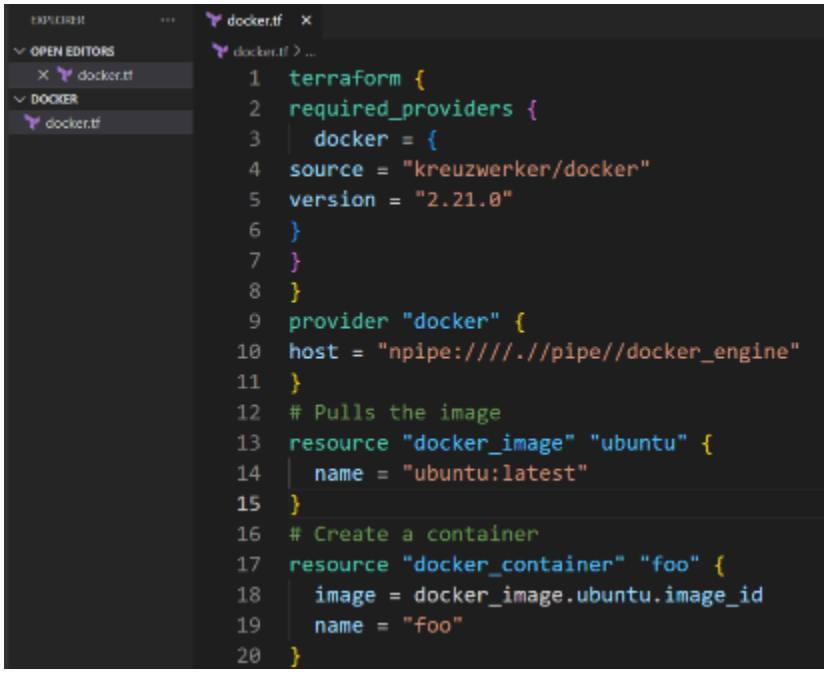
```
Microsoft Windows [Version 10.0.22621.4037]
(c) Microsoft Corporation. All rights reserved.

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

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

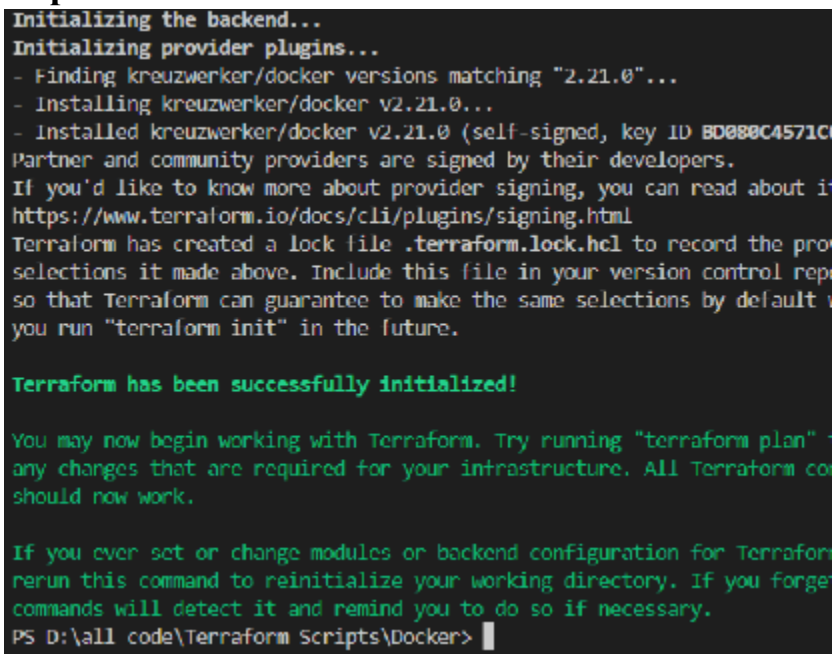
```
terraform
  { required_providers
  {docker = {
     source = "kreuzwerker/docker"
     version = "2.21.0"
    }
  }
}

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

# Pulls the image
resource "docker_image" "ubuntu"
```

{name = "ubuntu:latest"
}

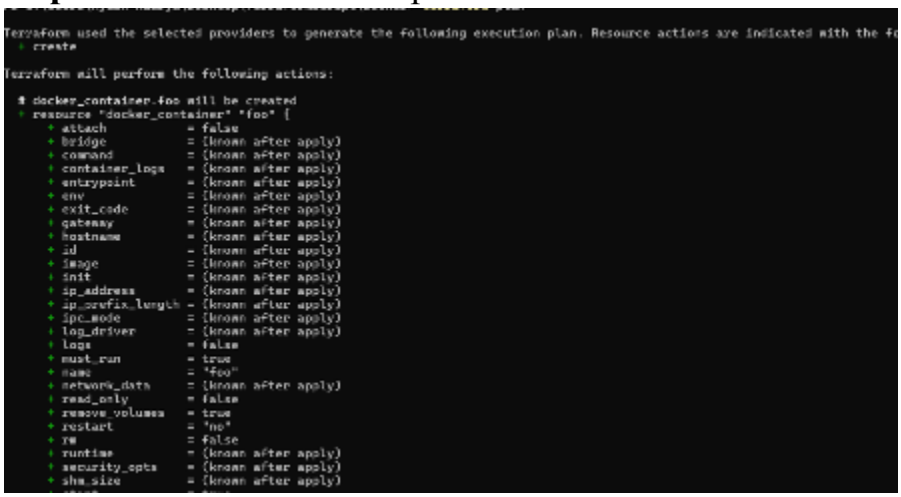
Create a container
resource "docker_container" "foo"
{ image =
docker_image.ubuntu.image_idname =
"foo"
}



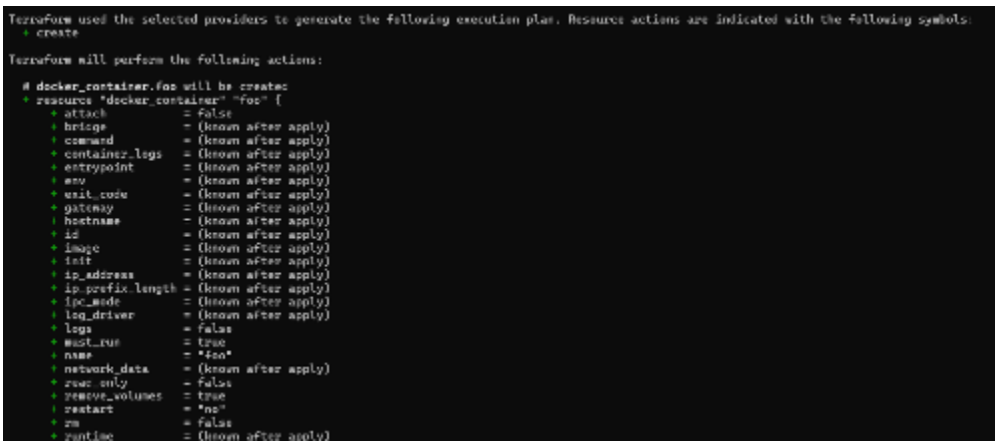
Step 3: Execute Terraform Init command to initialize the resources



Step 4: Execute Terraform plan to see the available resources



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



Docker images,

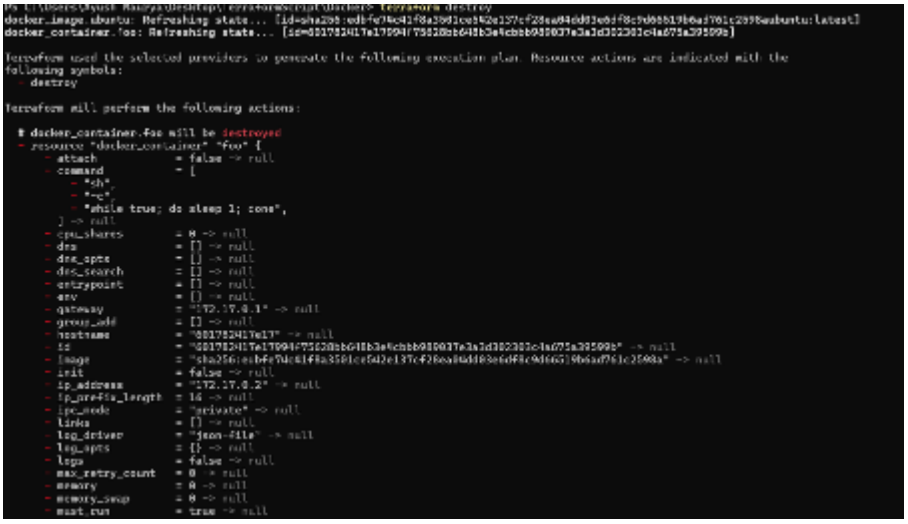
Before Executing Apply step:

```
C:\Users\Ayush Maurya>docker images
REPOSITORY   TAG       IMAGE ID       CREATED      SIZE
react-img    latest    d8b8903ee063   8 days ago   320MB
```

Docker images, After Executing Apply step:

```
Windows PowerShell
PS E:\Terraform Script\Docker> docker images
REPOSITORY                      TAG       IMAGE ID       CREATED          SIZE
docker101tutorial               latest    e5bc7d28c9a0   30 minutes ago   28.9MB
manavjawrani/docker101tutorial  latest    e5bc7d28c9a0   30 minutes ago   28.9MB
ubuntu                          latest    2dc39ba059dc   2 weeks ago      77.8MB
alpine/git                      latest    692618a0d74d   2 weeks ago      43.4MB
PS E:\Terraform Script\Docker>
```

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



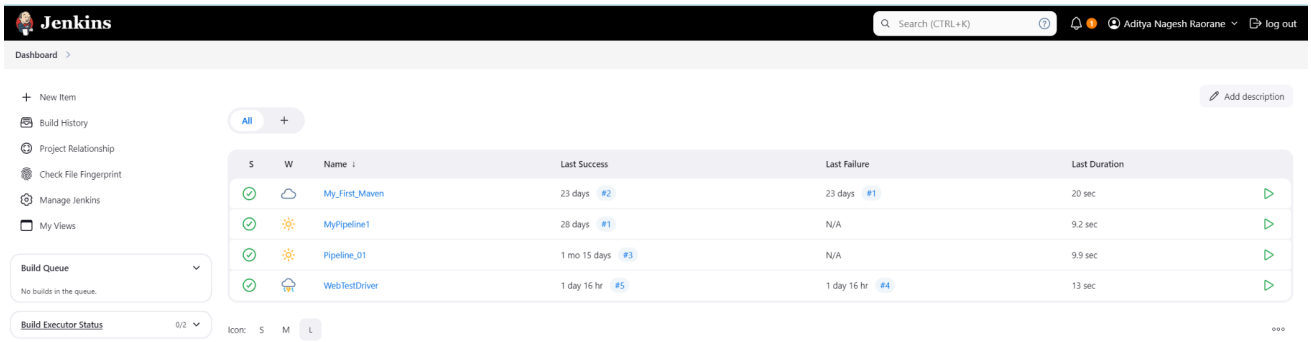
Docker images After Executing Destroy step

```
PS C:\Users\Ayush Maurya\Desktop\TerraformScript\Docker> docker images
REPOSITORY   TAG       IMAGE ID       CREATED      SIZE
react-img    latest    d8b8903ee063   8 days ago   320MB
PS C:\Users\Ayush Maurya\Desktop\TerraformScript\Docker>
```

Name: Laukik Padgaonkar Class: D15C/ Batch B Roll No: 37

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

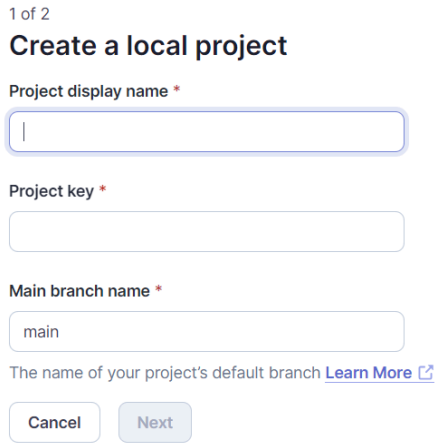
1. Open up Jenkins Dashboard on localhost, port 8080 or whichever port it is at for you.



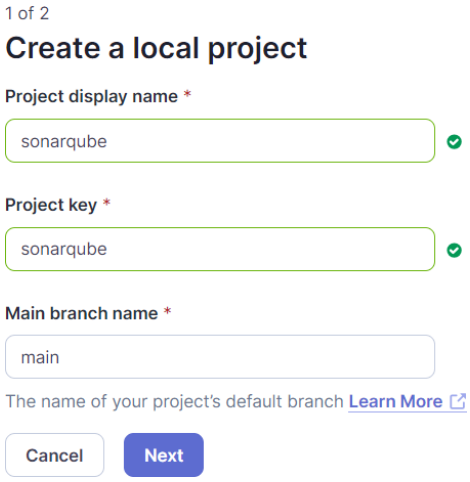
2. Run SonarQube in a Docker container using this command :- a] docker -v
b] docker pull sonarqube
c] docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest

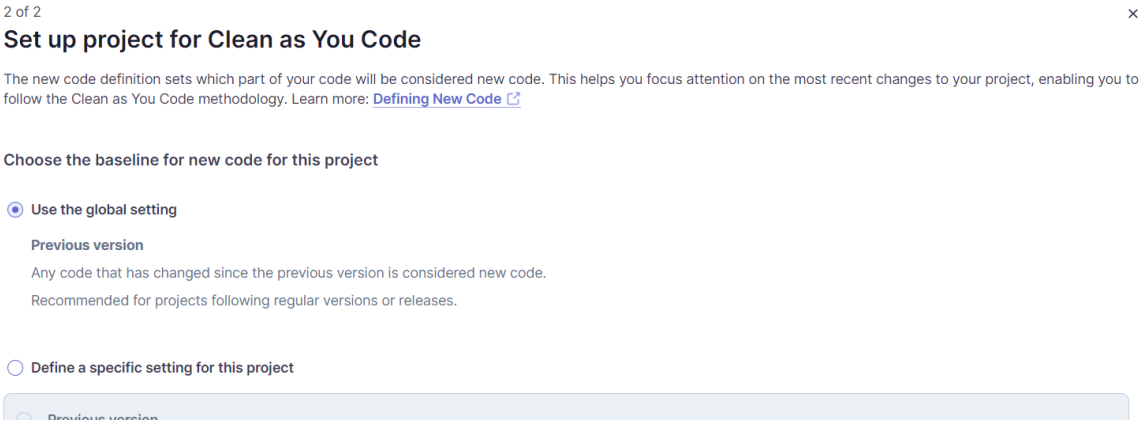
```
PS C:\Users\lauki> docker pull sonarqube
Using default tag: latest
latest: Pulling from library/sonarqube
bd819c9b5ead: Download complete
80338217a4ab: Download complete
1a5fd5c7e184: Download complete
4f4fb700ef54: Download complete
7b87d6fa783d: Downloading [==========>
7478e0ac0f23: Download complete
90a925ab929a: Download complete
7d9a34308537: Download complete
```

3. Once the container is up and running, you can check the status of SonarQube at **localhost port 9000**. The login id is “**admin**” and the password is “**aditya**”.

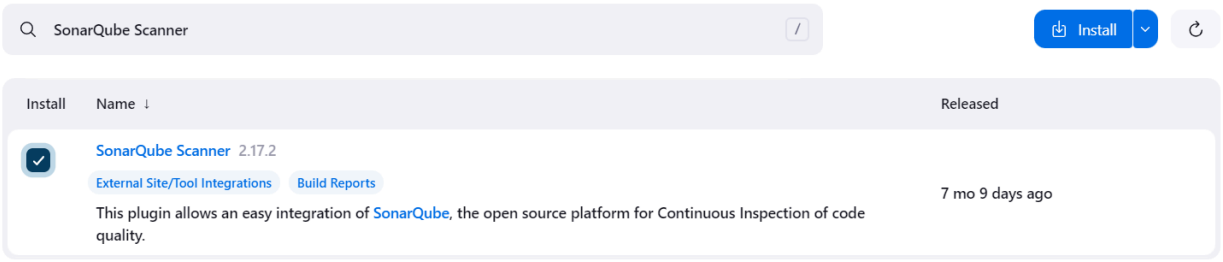


4. Create a local project in SonarQube with the name **sonarqube**





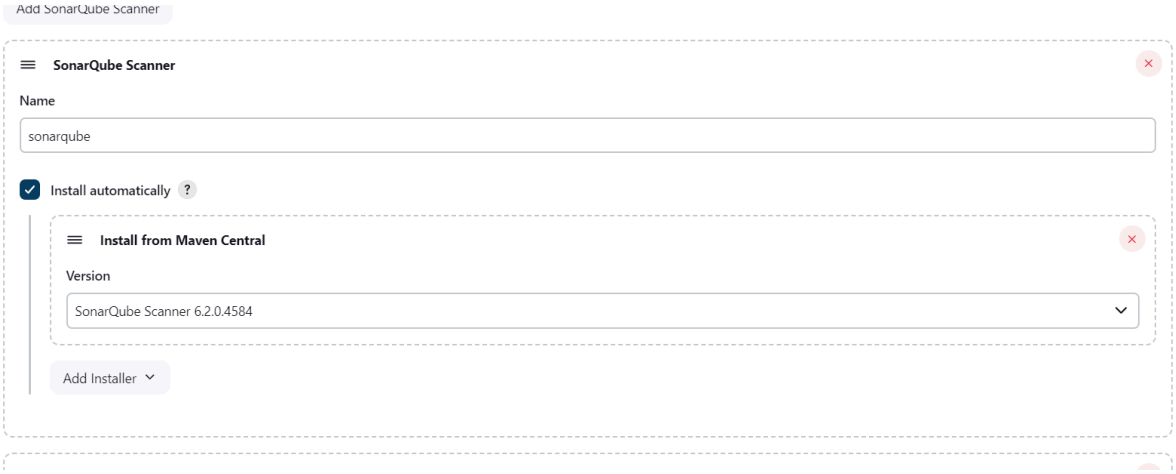
5. Setup the project and come back to Jenkins Dashboard. Go to **Manage Jenkins** → **Plugins** and search for **SonarQube Scanner** in **Available Plugins** and install it.



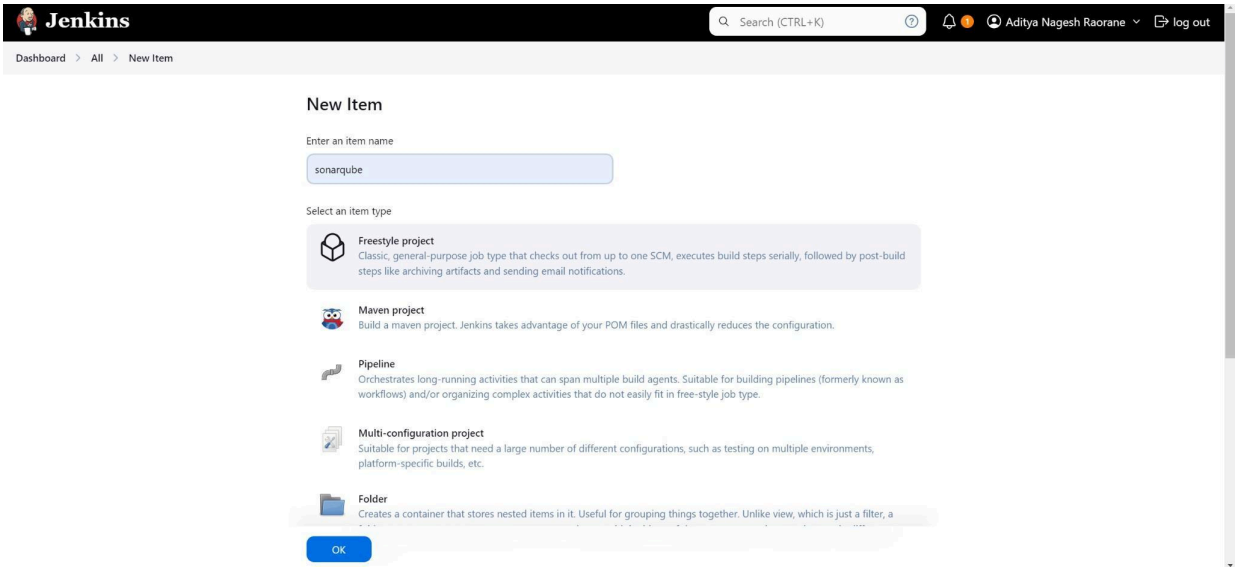
6. Under '**Manage Jenkins** → **System**', look for **SonarQube Servers** and enter these details.
Name : sonarqube
Server URL : http://localhost:9000

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

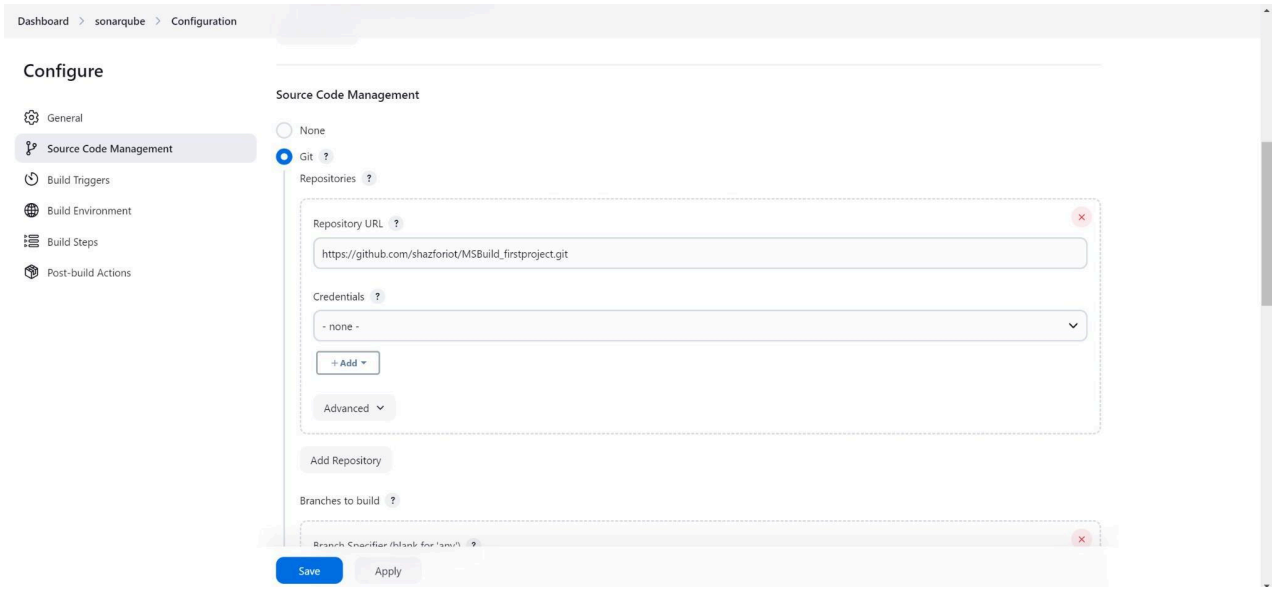
Manage Jeknins → Tools → SonarQube Scanner Installation



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



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

sonar.projectKey=sonarqube

sonar.login=admin

sonar.password=aditya

sonar.sources=.

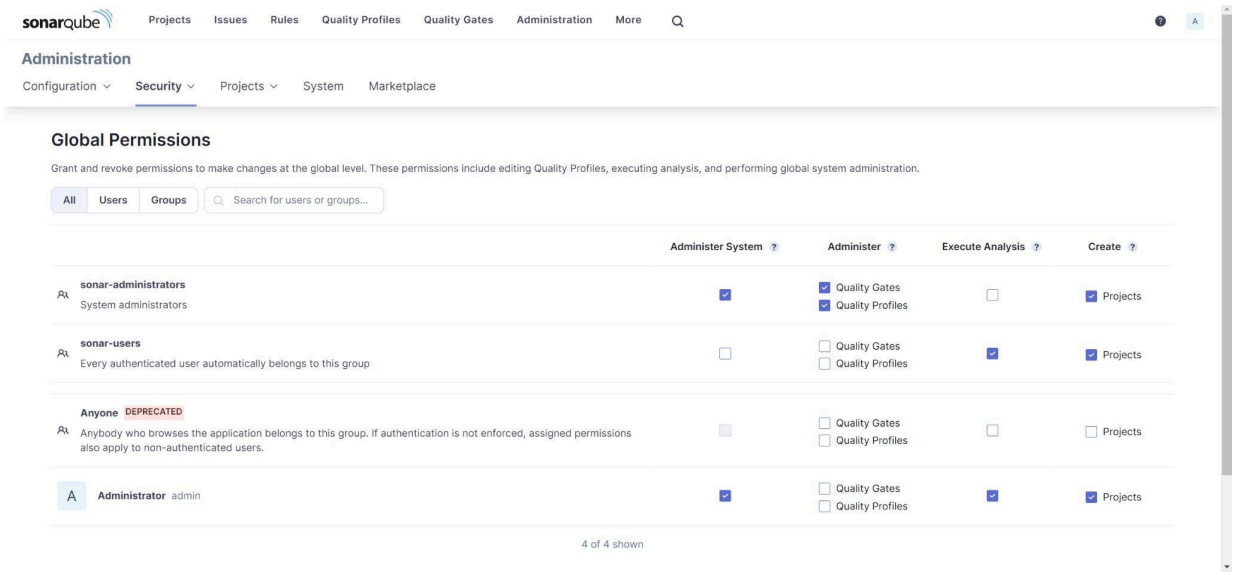
sonar.host.url=http://localhost:9000

Analysis properties

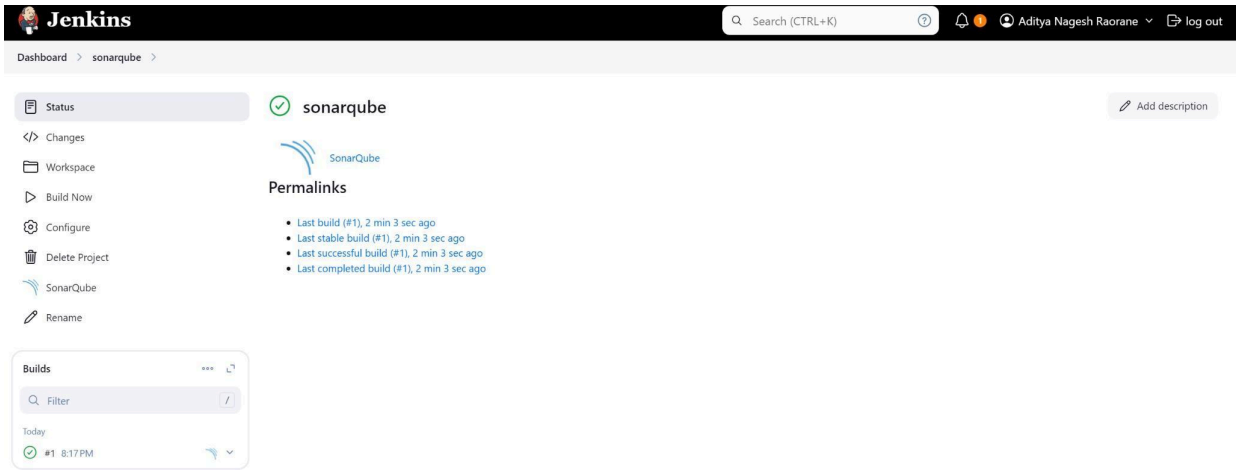
```
sonar.projectKey=sonarqube
sonar.login=admin
sonar.password=laukik
sonar.sources=.
sonar.host.url=http://localhost:9000
```

Additional arguments

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



12. Run The **Build** and check the **console output**.

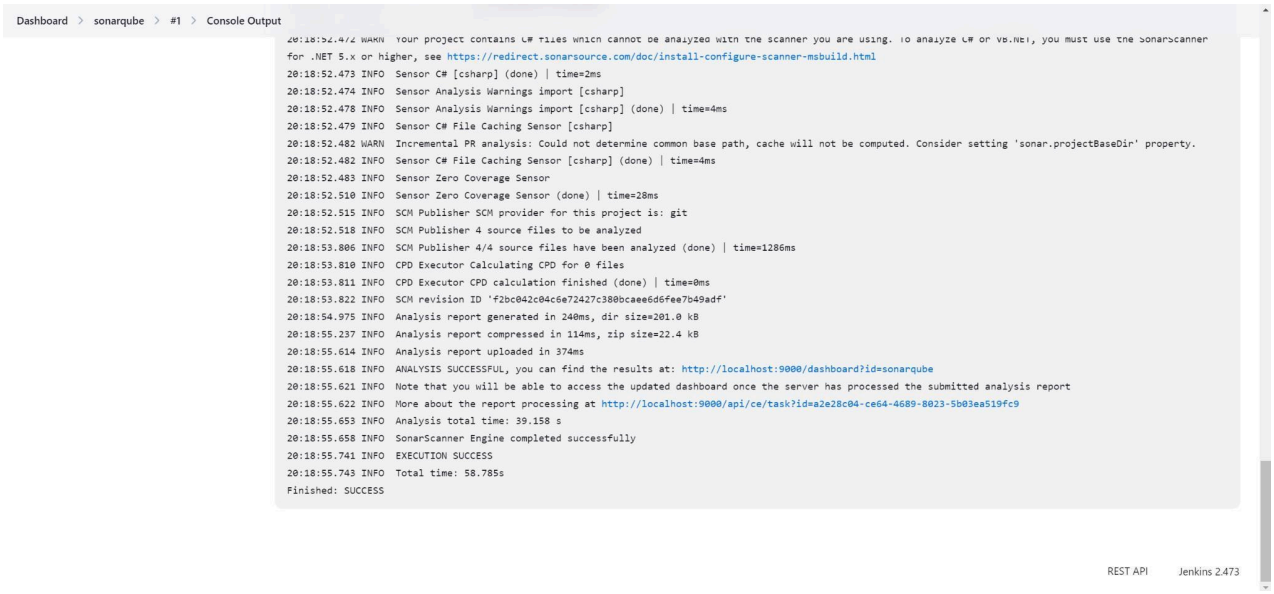


Console Output

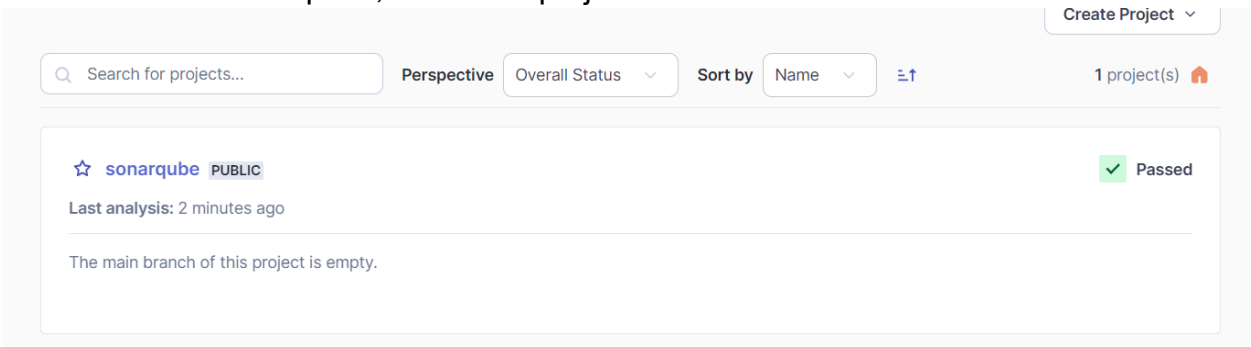
Download Copy View as plain text

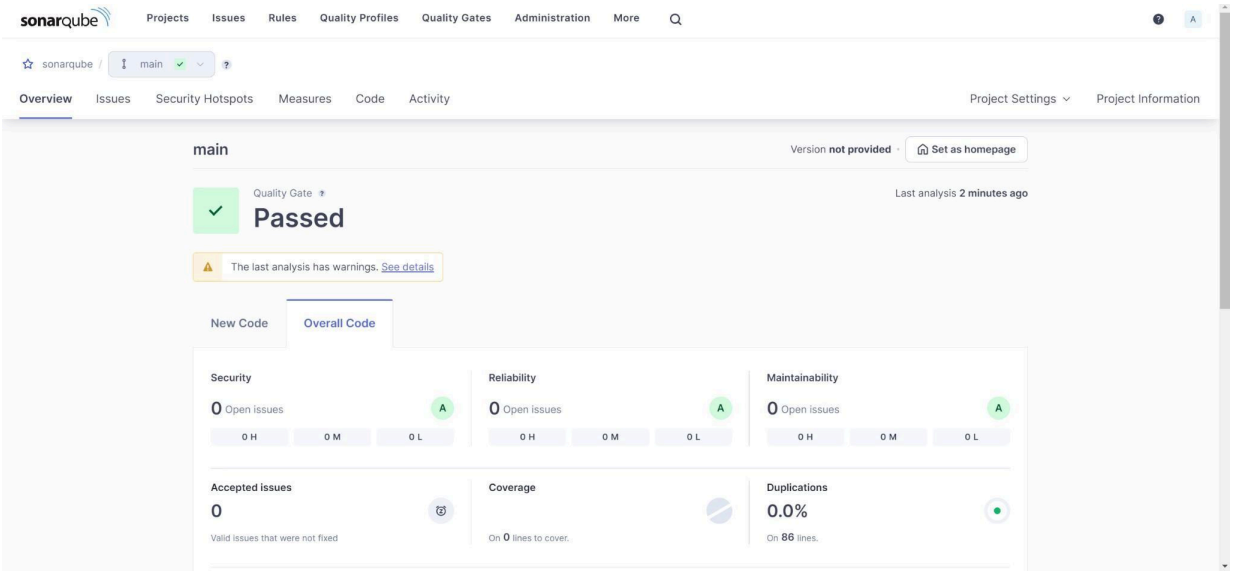
```
Started by user laukik padgaonkar
Running as SYSTEM
Building on the built-in node in workspace C:\ProgramData\Jenkins\.jenkins\workspace\sonarqube
The recommended git tool is: NONE
No credentials specified
 > git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\sonarqube\.git # timeout=10
Fetching changes from the remote Git repository
 > git.exe config remote.origin.url https://github.com/shazforiot/MSBuild_firstproject # timeout=10
Fetching upstream changes from https://github.com/shazforiot/MSBuild_firstproject
 > git.exe --version # timeout=10
 > git --version # 'git version 2.43.0.windows.1'
 > git.exe fetch --tags --force --progress -- https://github.com/shazforiot/MSBuild_firstproject +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision f2bc042c04c6e72427c380bcaee6d6fee7b49adf (refs/remotes/origin/master)
 > git.exe config core.sparsecheckout # timeout=10
 > git.exe checkout -f f2bc042c04c6e72427c380bcaee6d6fee7b49adf # timeout=10
Commit message: "updated"
 > git.exe rev-list --no-walk f2bc042c04c6e72427c380bcaee6d6fee7b49adf # timeout=10
[sonarqube] $ C:\ProgramData\Jenkins\.jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\sonarqube\bin\sonar-scanner.bat -
Dsonar.host.url=http://localhost:9000 -Dsonar.projectKey=sonarqube -Dsonar.login=admin -Dsonar.host.url=http://localhost:9000 -Dsonar.sources=. -
Dsonar.password=laukik -Dsonar.projectBaseDir=C:\ProgramData\Jenkins\.jenkins\workspace\sonarqube
18:54:47.035 WARN  Property 'sonar.host.url' with value 'http://localhost:9000' is overridden with value 'http://localhost:9000'
18:54:47.044 INFO  Scanner configuration file:
```

Name: Laukik Padgaonkar Class: D15C/ Batch B Roll No: 37



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





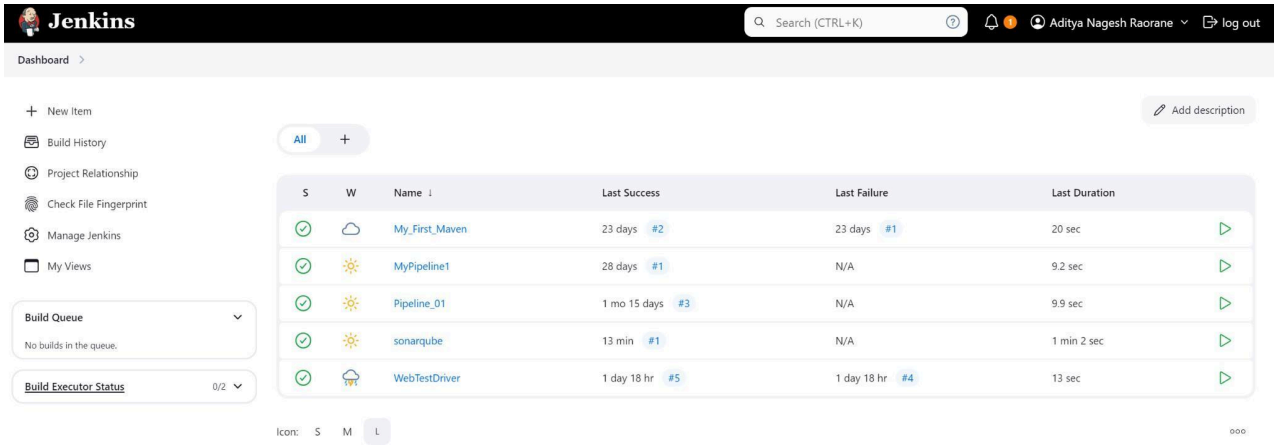
In this way, we have integrated Jenkins with SonarQube for SAST.

Conclusion:

In this experiment, we have understood the importance of SAST and have successfully integrated Jenkins with SonarQube for Static Analysis and Code Testing.

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.

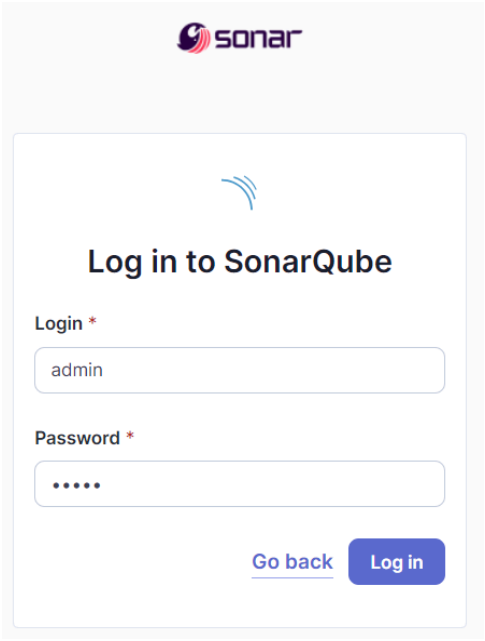
1. Open up Jenkins Dashboard on localhost:8080.



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

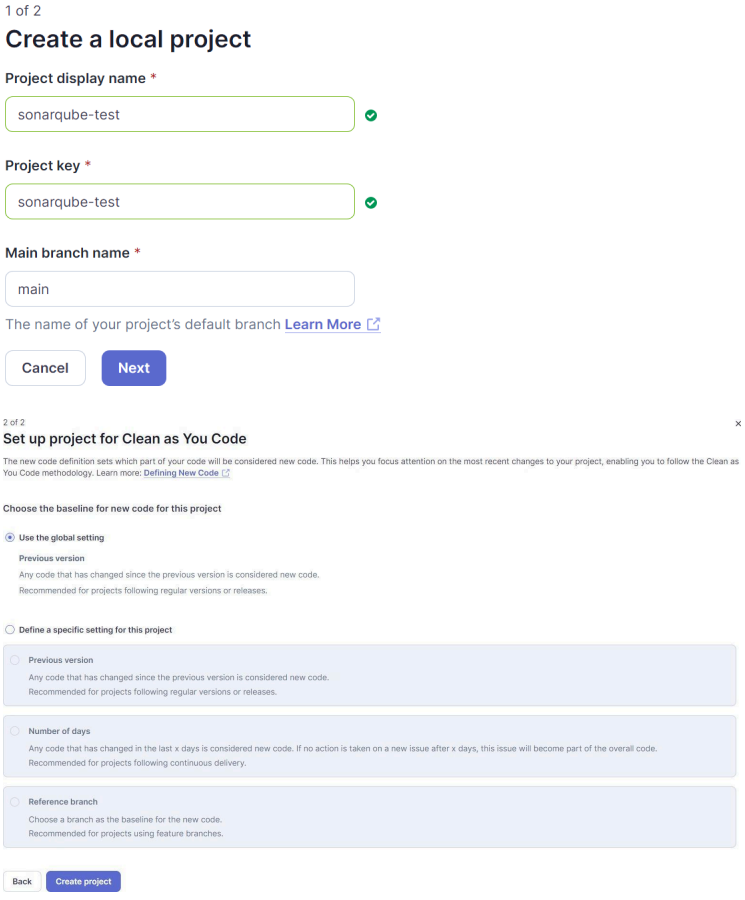
```
C:\Users\adity>docker -v
Docker version 27.0.3, build 7d4bcd8

C:\Users\adity>docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
Unable to find image 'sonarqube:latest' locally
latest: Pulling from library/sonarqube
7478e0ac0f23: Pull complete
90a925ab929a: Pull complete
7d9a34308537: Pull complete
80338217a4ab: Pull complete
1a5fd5c7e184: Pull complete
7b87d6fa783d: Pull complete
bd819c9b5ead: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:72e9feec71242af83faf65f95a40d5e3bb2822a6c3b2cda8568790f3d31aecde
Status: Downloaded newer image for sonarqube:latest
4a6e73f4472de892b1ddead1abe77372a85a7b09408cce3a0abd37c5ab6b49a4
```

3. Once the container is up and running, you can check the status of SonarQube at **localhost port 9000**. The login id is “**admin**” and the password is



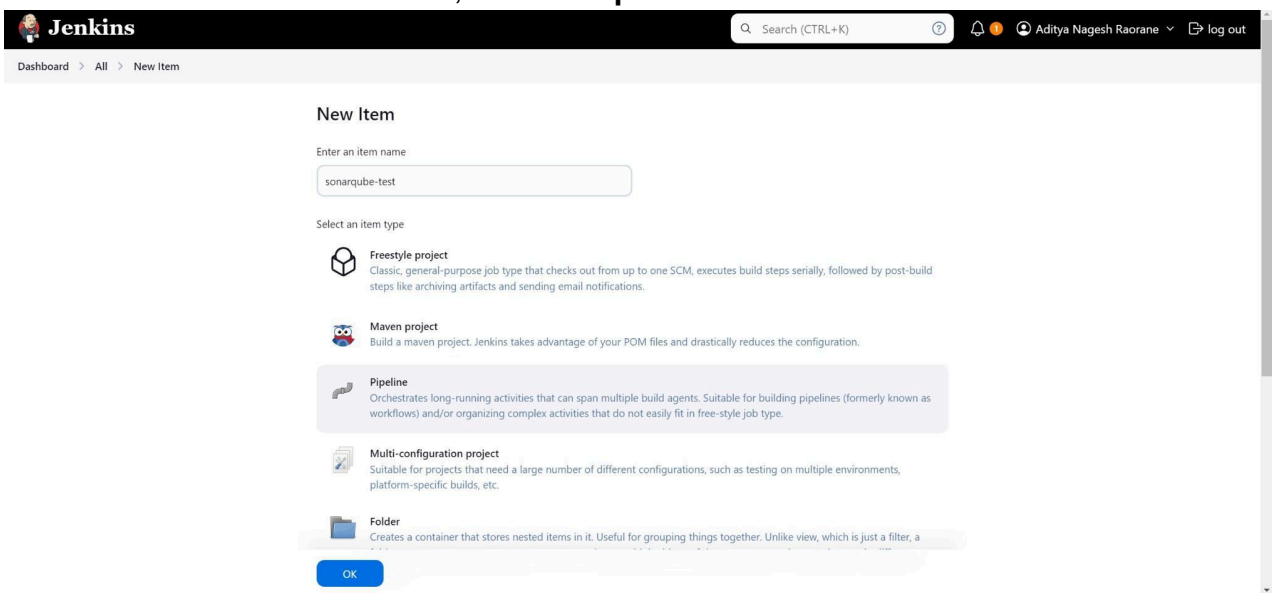
“**aditya**”.

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



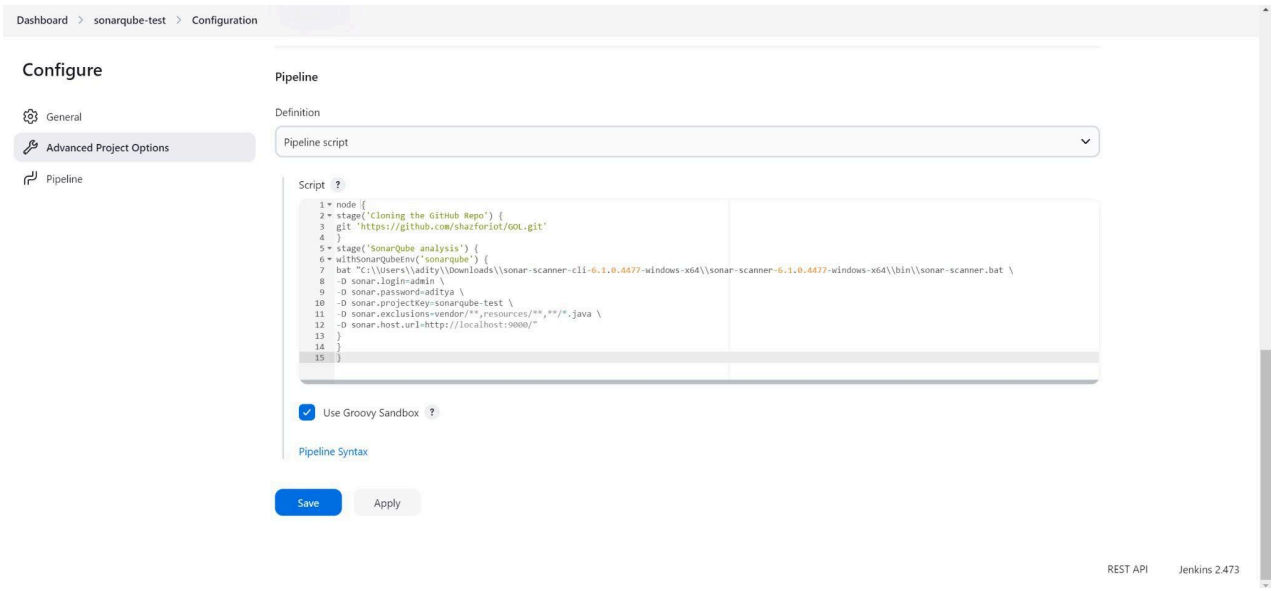
Setup the project and come back to Jenkins Dashboard.

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



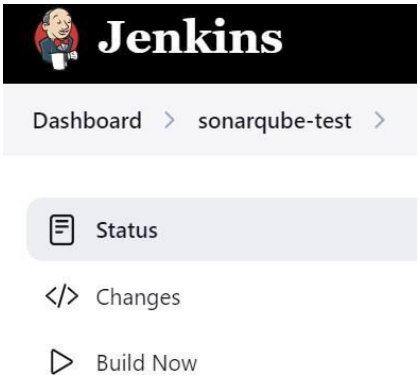
7. Under **Pipeline Script**, enter the following -

```
node {
        stage('Cloning the GitHub Repo')
        {
                git 'https://github.com/shazforiot/GOL.git'
        }
stage('SonarQube analysis') {
        withSonarQubeEnv('sonarqube') {
                bat
        "C:\\Users\\adity\\Downloads\\sonar-scanner-cli-6.1.0.4477-windows-x64\\sonar-
        s canner-6.1.0.4477-windows-x64\\bin\\sonar-scanner.bat \
                -D sonar.login=<YOUR ID> \
                -D sonar.password=<YOUR PASSWORD> \
                -D sonar.projectKey=<YOUR PROJECT KEY> \
                -D sonar.exclusions=vendor/**,resources/**,**/*.java \
                -D sonar.host.url=http://localhost:9000/"
                }
        }
}
```

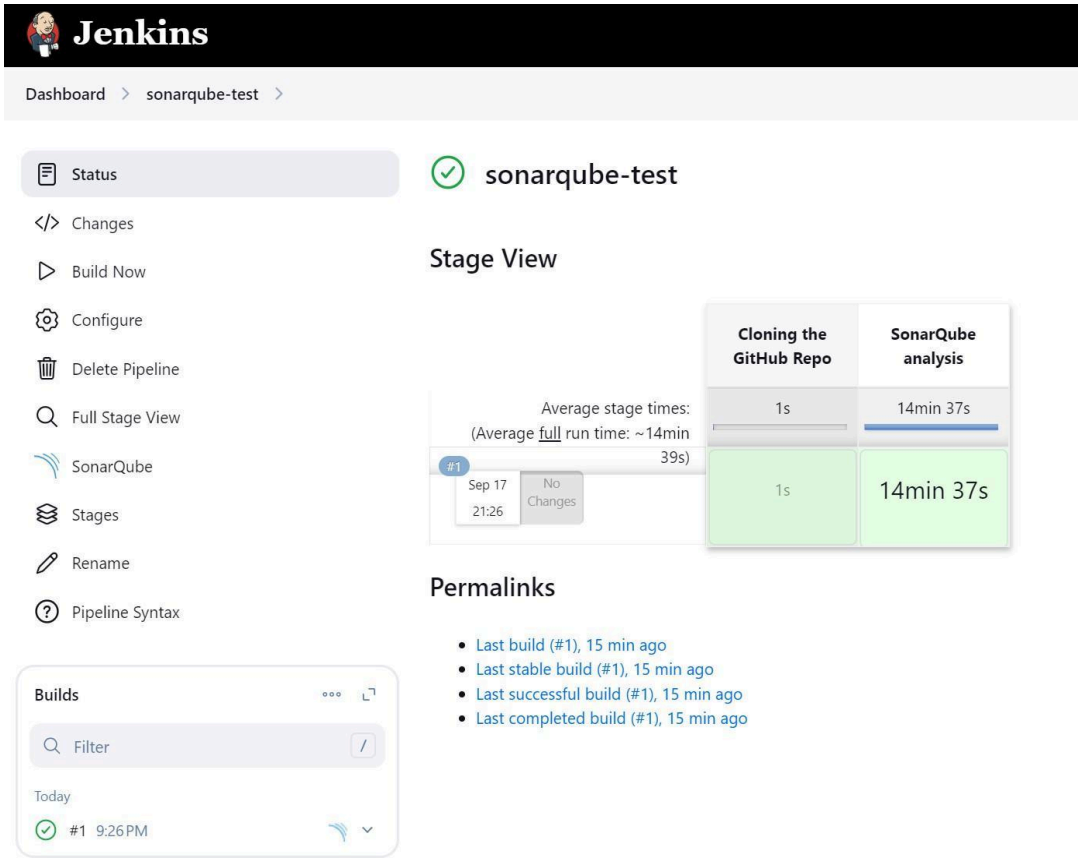


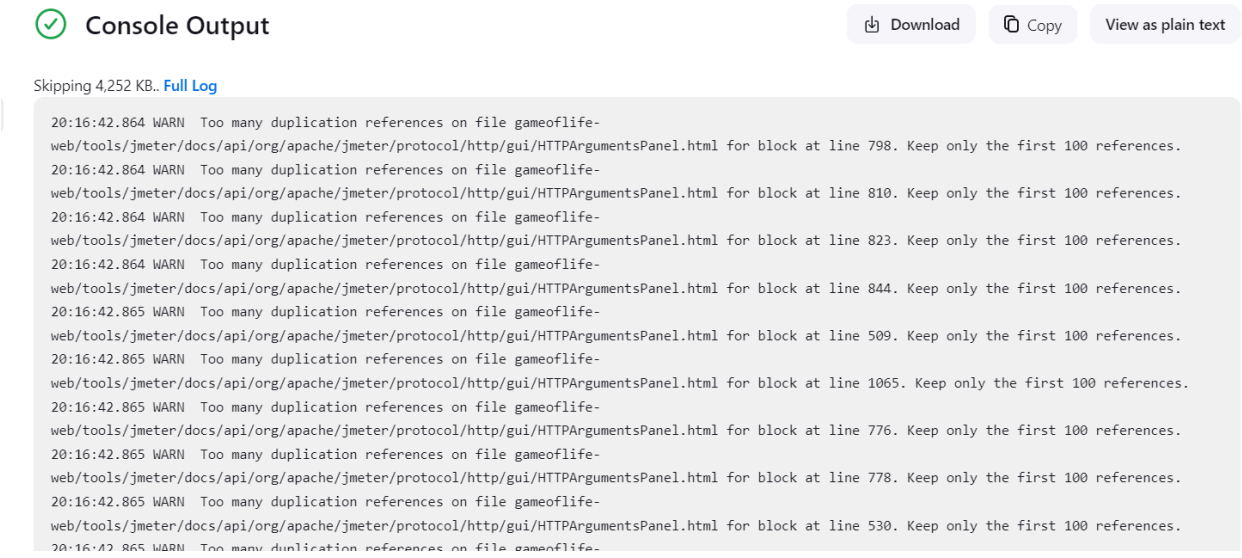
It is a java sample project which has a lot of repetitions and issues that will be detected by SonarQube.

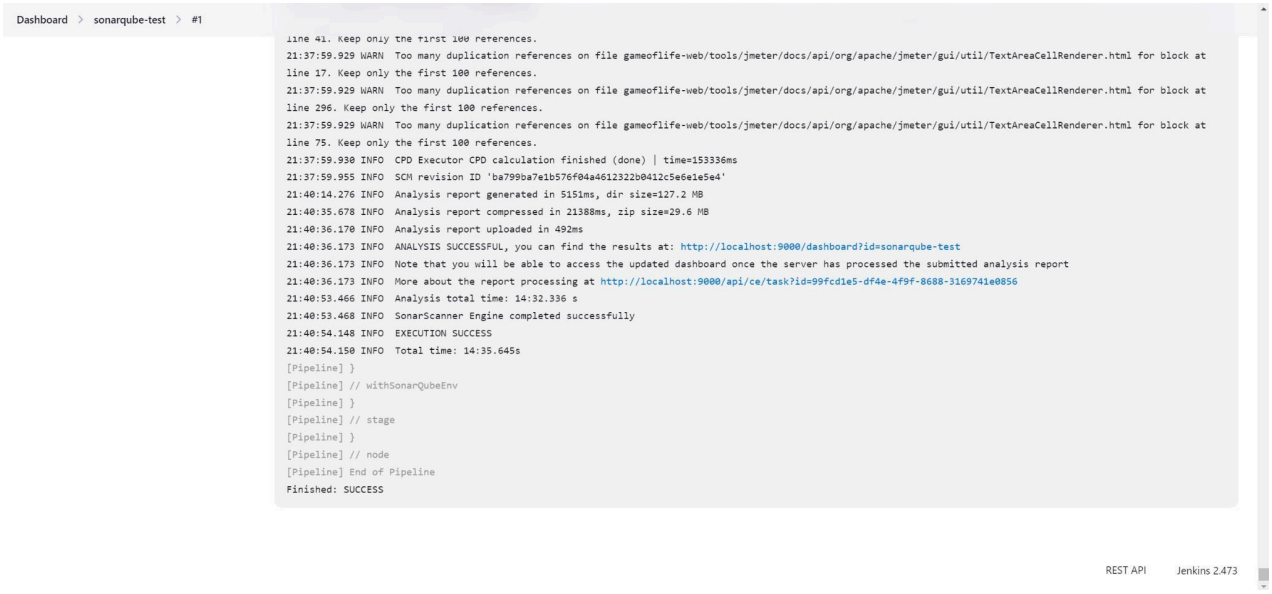
8. Run The Build.



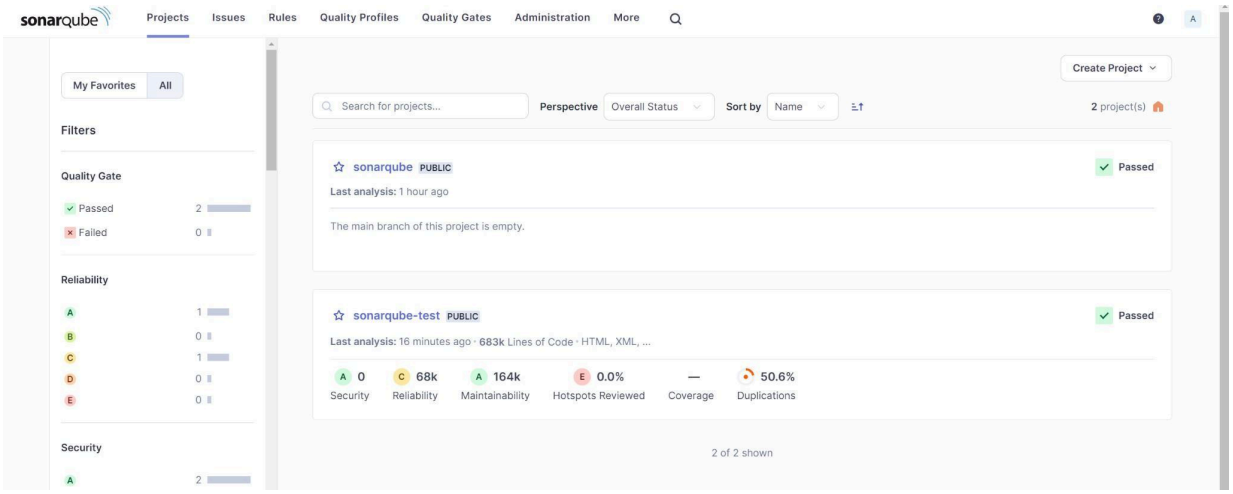
9. Check the console output once the build is complete.



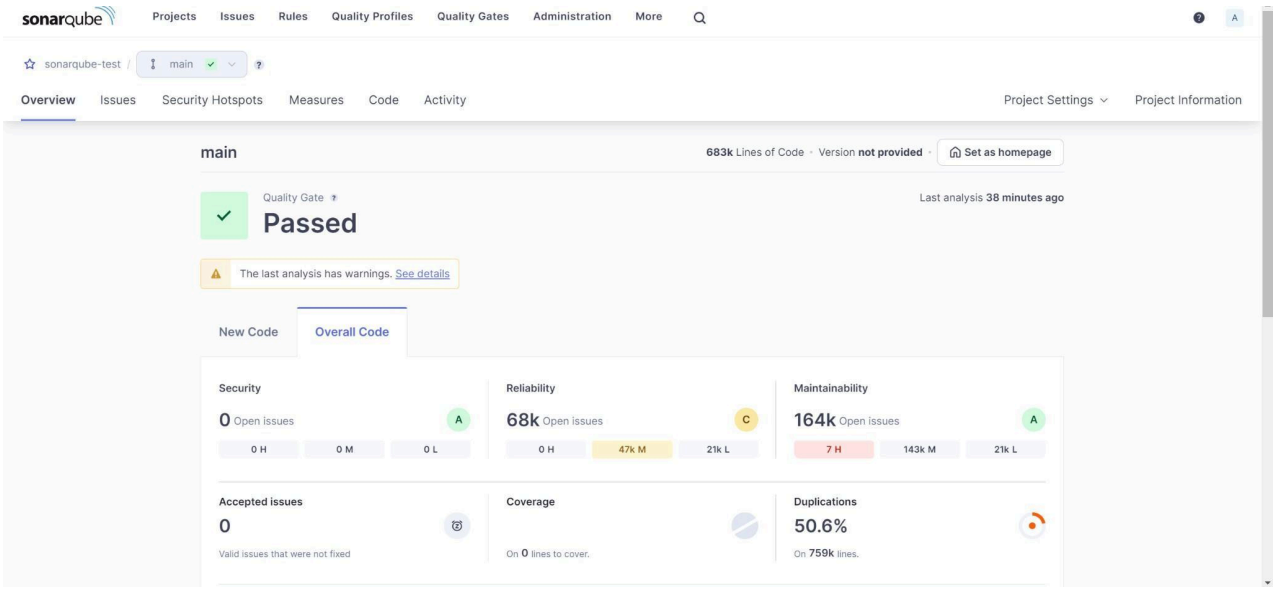




10. After that, check the project in SonarQube.



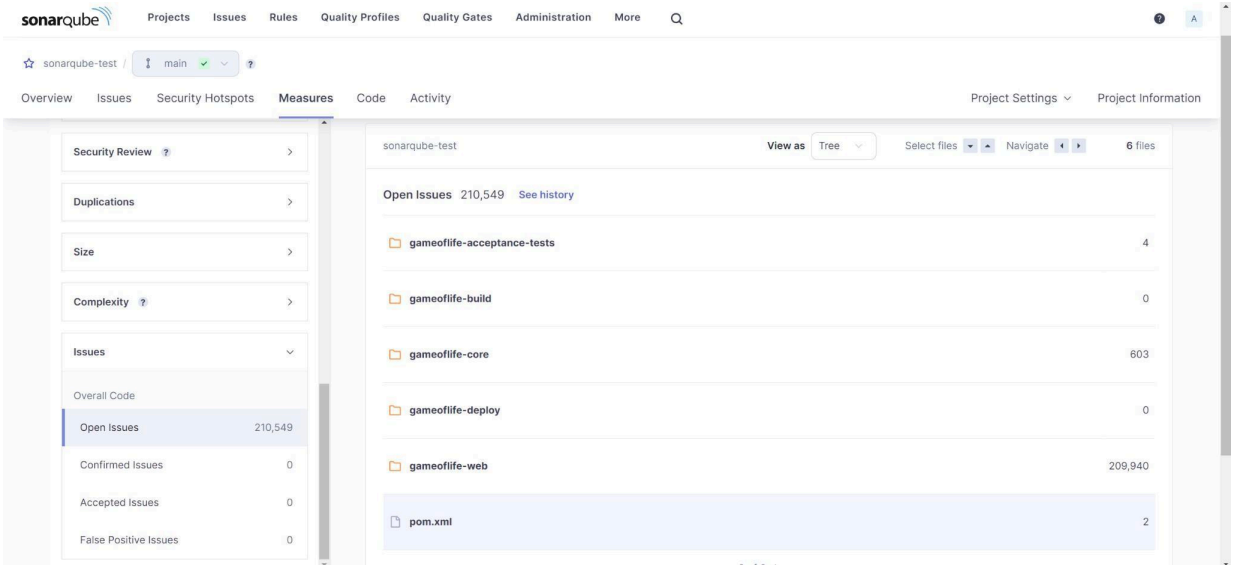
Name: Laukik Padgaonkar Class: D15C/ Batch B Roll No: 37



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

11. **Code**

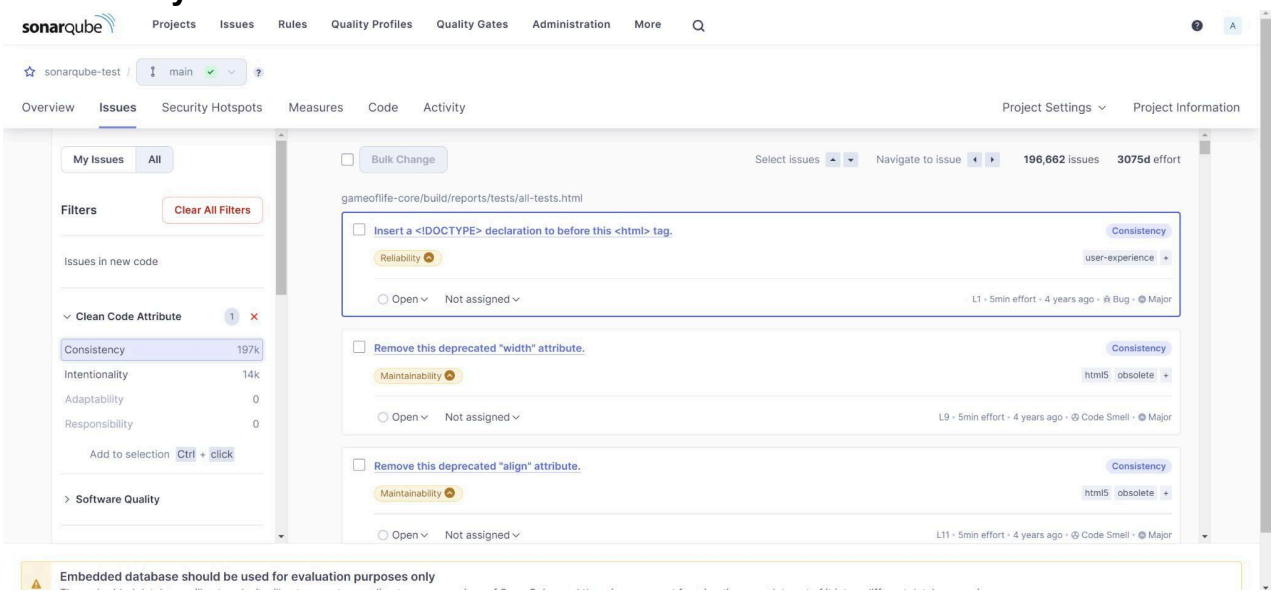
Problems - Open



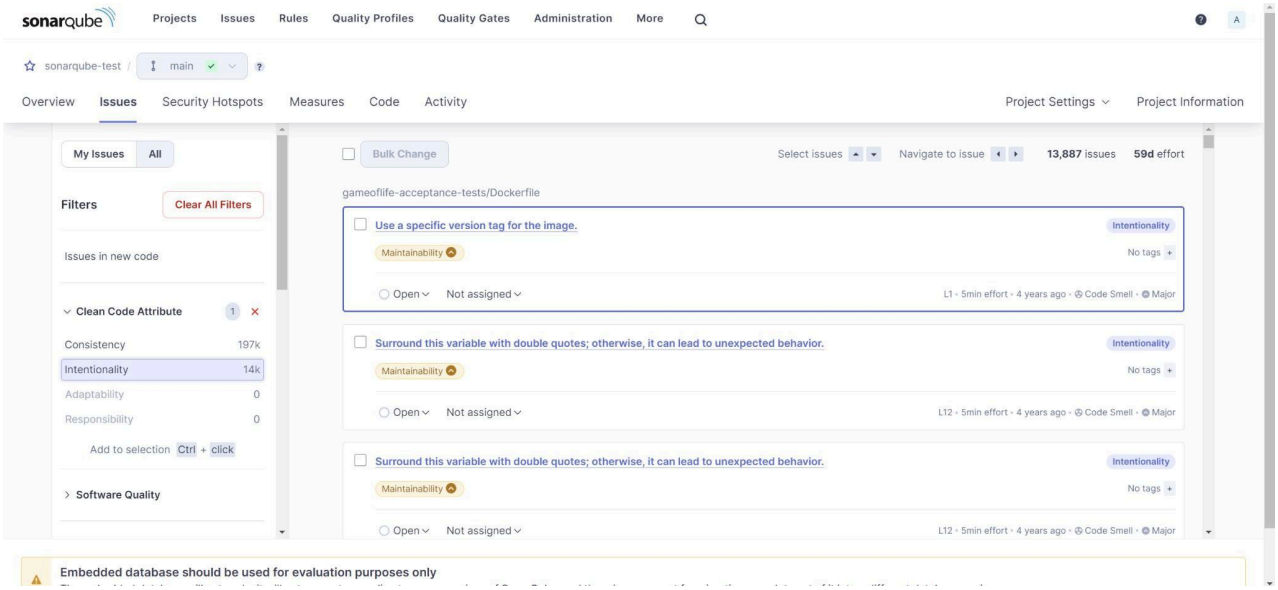
Issues

Name: Laukik Padgaonkar Class: D15C/ Batch B Roll No: 37

Consistency

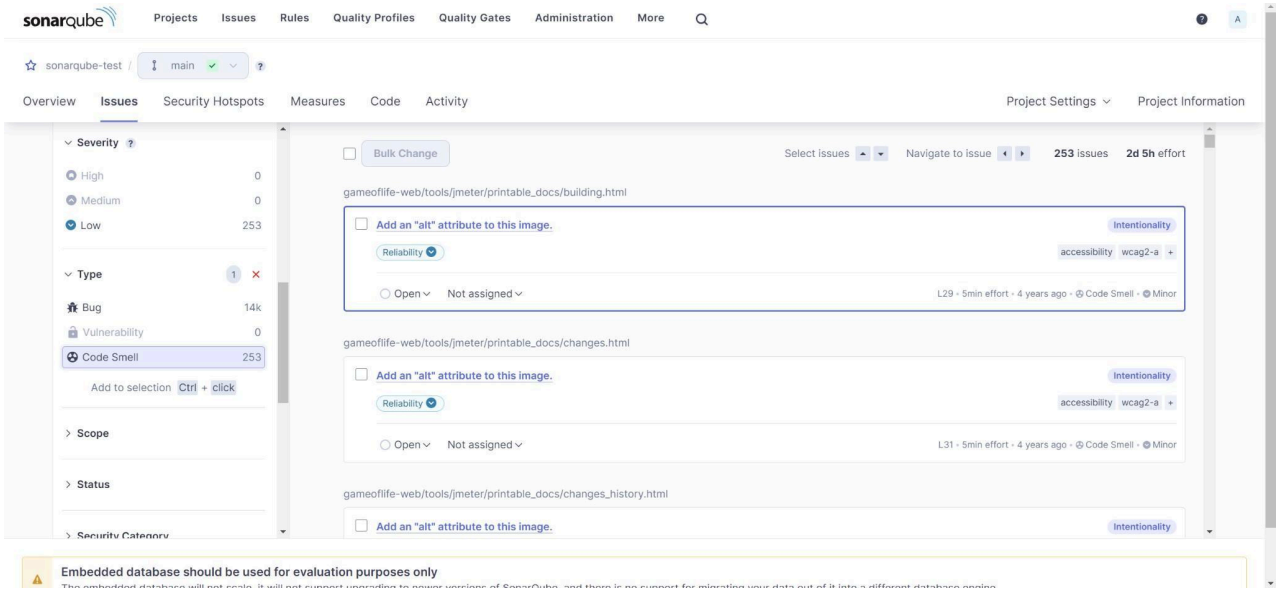


Intentionality

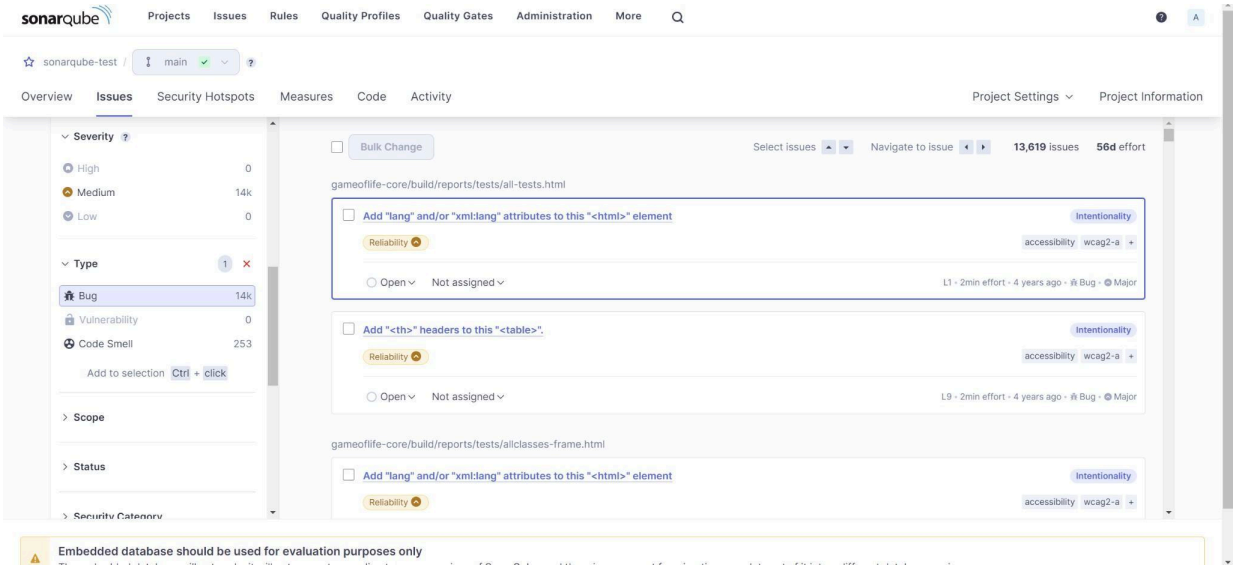


Code Smells

Name: Laukik Padgaonkar Class: D15C/ Batch B Roll No: 37

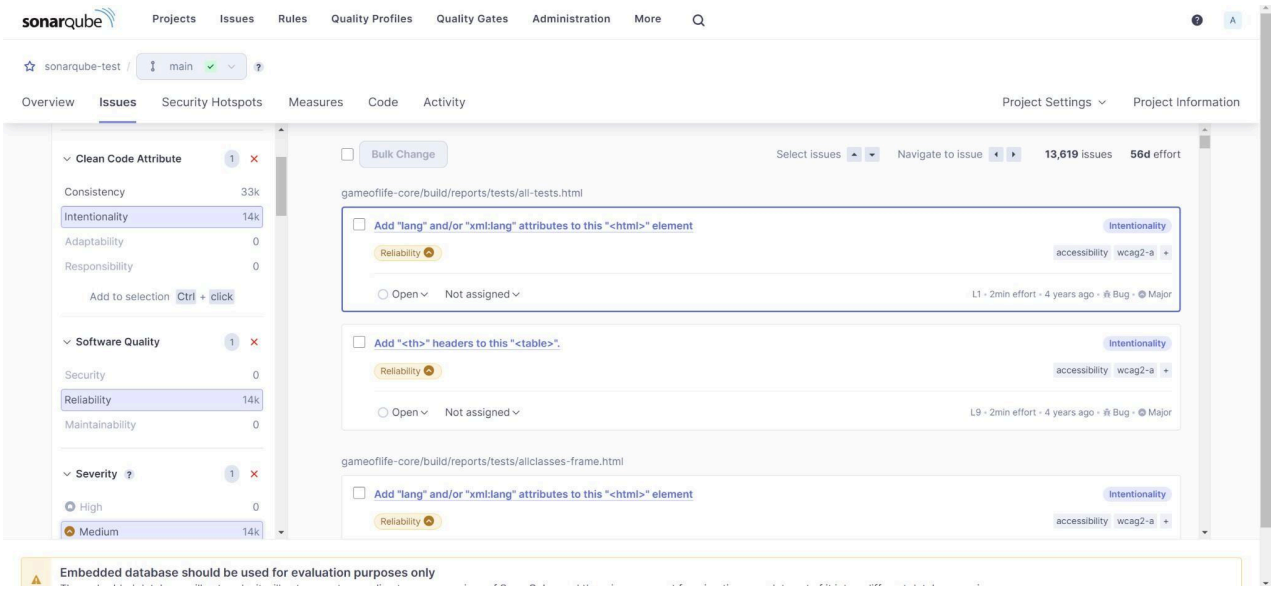


Bugs

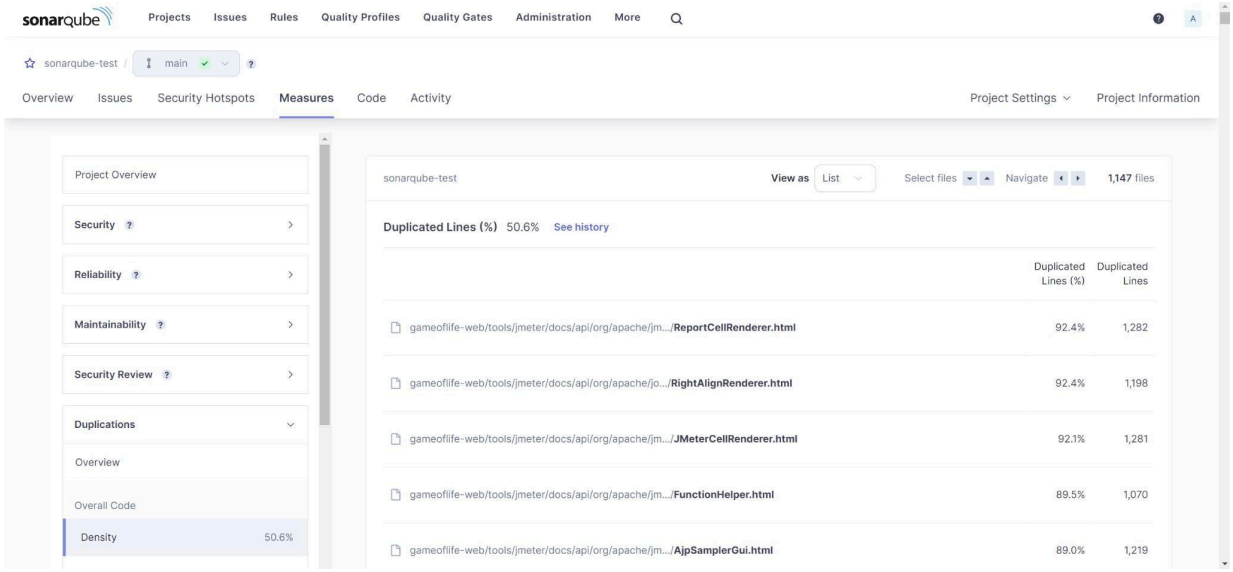


Name: Laukik Padgaonkar Class: D15C/ Batch B Roll No: 37

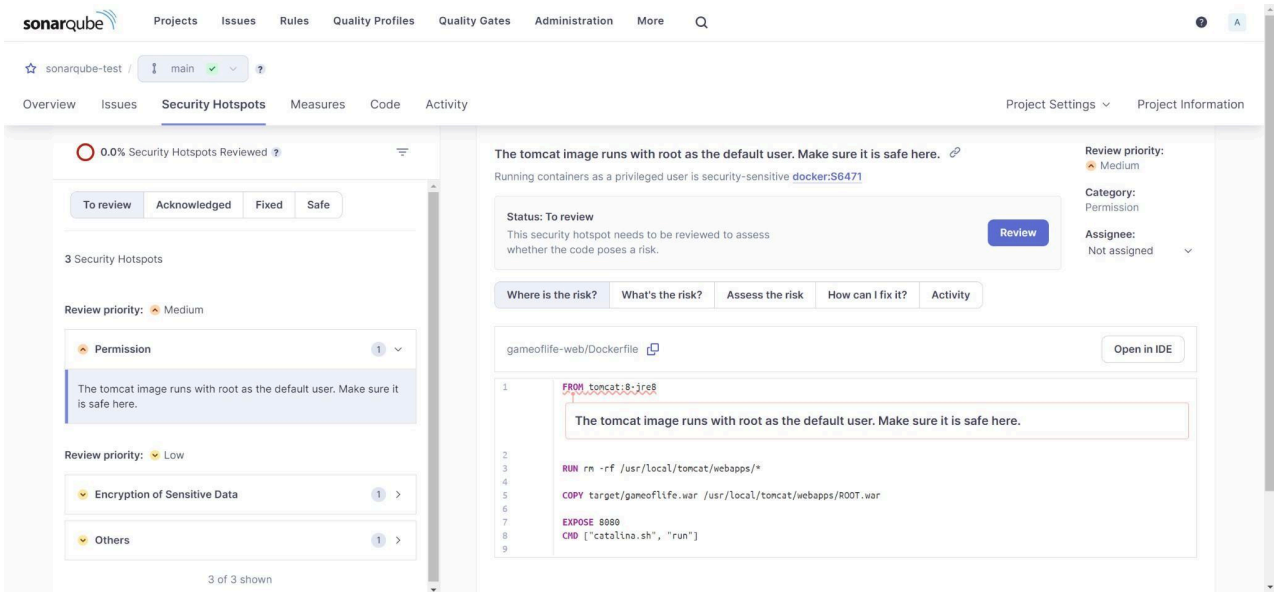
Reliability



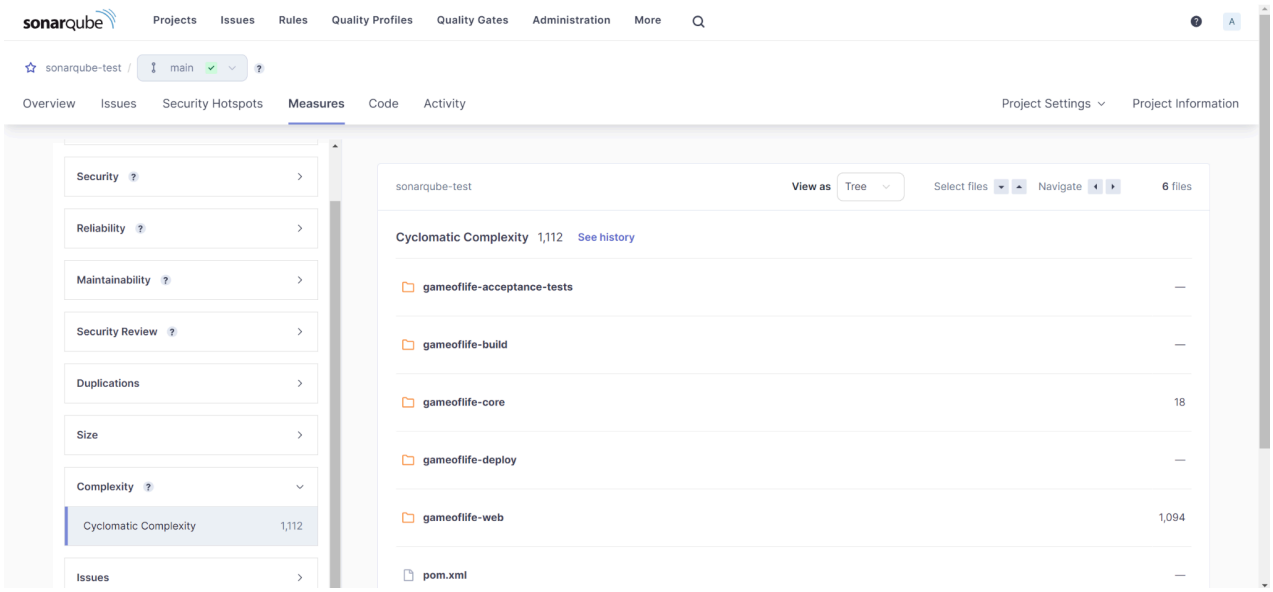
Duplicates



Security Hotspot



Cyclomatic Complexity



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

Conclusion:

In this experiment, we performed a static analysis of the code to detect bugs, code smells, and security vulnerabilities on our sample Java application.

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

Theory:

What is Nagios?

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

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

Why We Need Nagios tool?

Here are the important reasons to use Nagios monitoring tool:

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

Features of Nagios

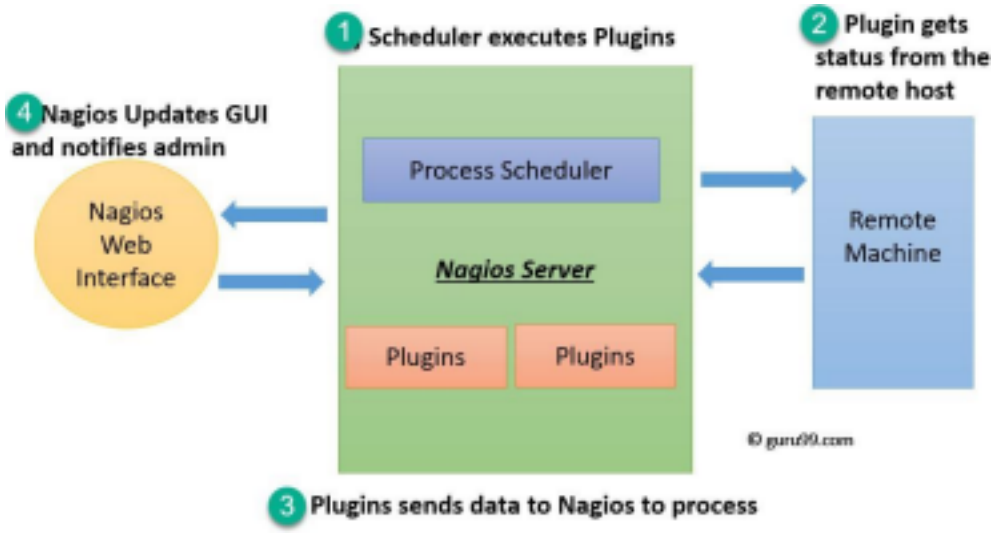
Following are the important features of Nagios monitoring tool:

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

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

Nagios Architecture

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



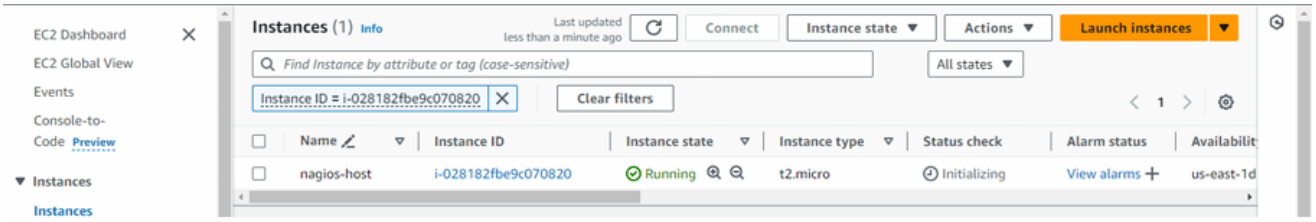
1. The scheduler is a component of the server part of Nagios. It sends a signal to execute the plugins at the remote host.
2. The plugin gets the status from the remote host
3. The plugin sends the data to the process scheduler
4. The process scheduler updates the GUI and notifications are sent to admins.

Installation of Nagios

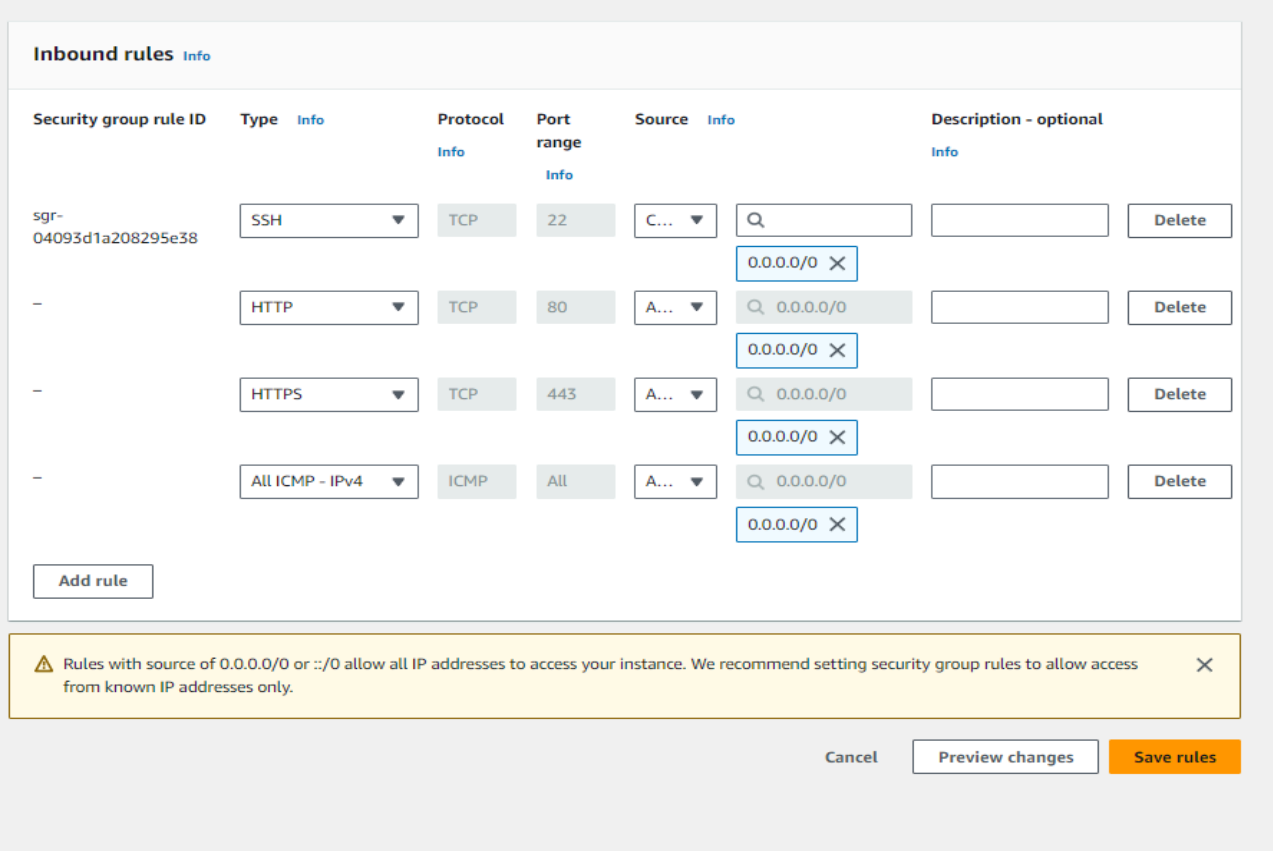
Prerequisites: AWS Free Tier

Steps:

1. Create an Amazon Linux EC2 Instance in AWS and name it - nagios-host



2. Under Security Group, make sure HTTP, HTTPS, SSH, ICMP are open from everywhere. You have to edit the inbound rules of the specified Security Group for this.



3. SSH into Your EC2 instance or simply use EC2 Instance Connect from the browser.

4. Update the package indices and install the following packages using yum

sudo yum update

```
sudo yum install httpd php
sudo yum install gcc glibc glibc-common
sudo yum install gd gd-devel
```

```
yum install: error: unrecognized arguments: -Oy
[ec2-user@ip-172-31-38-150 ~]$ sudo yum install gd gd-devel -y
Last metadata expiration check: 0:05:12 ago on Sun Oct  6 11:15:04 2024.
Dependencies resolved.
```

Package	Architecture	Version	Repository	Si
Installing:				
gd	x86_64	2.3.3-5.amzn2023.0.3	amazonlinux	139
gd-devel	x86_64	2.3.3-5.amzn2023.0.3	amazonlinux	38
Installing dependencies:				
brotli	x86_64	1.0.9-4.amzn2023.0.2	amazonlinux	314
brotli-devel	x86_64	1.0.9-4.amzn2023.0.2	amazonlinux	31
bzip2-devel	x86_64	1.0.8-6.amzn2023.0.2	amazonlinux	214
cairo	x86_64	1.17.6-2.amzn2023.0.1	amazonlinux	684
cmake-filesystem	x86_64	3.22.2-1.amzn2023.0.4	amazonlinux	16
fontconfig	x86_64	2.13.94-2.amzn2023.0.2	amazonlinux	273
fontconfig-devel	x86_64	2.13.94-2.amzn2023.0.2	amazonlinux	128
fonts-filesystem	noarch	1:2.0.5-12.amzn2023.0.2	amazonlinux	9.5
freetype	x86_64	2.13.2-5.amzn2023.0.1	amazonlinux	423
freetype-devel	x86_64	2.13.2-5.amzn2023.0.1	amazonlinux	912
glib2-devel	x86_64	2.74.7-689.amzn2023.0.2	amazonlinux	486
google-noto-fonts-common	noarch	20201206-2.amzn2023.0.2	amazonlinux	15
google-noto-sans-vf-fonts	noarch	20201206-2.amzn2023.0.2	amazonlinux	492
graphite2	x86_64	1.3.14-7.amzn2023.0.2	amazonlinux	97
graphite2-devel	x86_64	1.3.14-7.amzn2023.0.2	amazonlinux	21
harfbuzz	x86_64	7.0.0-2.amzn2023.0.1	amazonlinux	868
harfbuzz-devel	x86_64	7.0.0-2.amzn2023.0.1	amazonlinux	404
harfbuzz-icu	x86_64	7.0.0-2.amzn2023.0.1	amazonlinux	18
jbigkit-libs	x86_64	2.1-21.amzn2023.0.2	amazonlinux	54
langpacks-core-font-en	noarch	3.0-21.amzn2023.0.4	amazonlinux	10
libICE	x86_64	1.0.10-6.amzn2023.0.2	amazonlinux	71
libSM	x86_64	1.2.3-8.amzn2023.0.2	amazonlinux	42

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

```
sudo adduser -m nagios
sudo passwd nagios
```

```
Complete!
[ec2-user@ip-172-31-38-150 ~]$ sudo adduser -m nagios
[ec2-user@ip-172-31-38-150 ~]$ sudo passwd nagios
Changing password for user nagios.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[ec2-user@ip-172-31-38-150 ~]$
```

6. Create a new user group

```
sudo groupadd nagcmd
```

7. Use these commands so that you don't have to use sudo for Apache and Nagios

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

8. Create a new directory for Nagios downloads

```
mkdir ~/downloads
cd ~/downloads
```

```
passwd: all authentication tokens updated successfully.
[ec2-user@ip-172-31-38-150 ~]$ sudo groupadd nagcmd
[ec2-user@ip-172-31-38-150 ~]$ sudo usermod -a -G nagcmd nagios
[ec2-user@ip-172-31-38-150 ~]$ sudo usermod -a -G nagcmd apache
[ec2-user@ip-172-31-38-150 ~]$ mkdir ~/downloads
[ec2-user@ip-172-31-38-150 ~]$ cd ~/downloads
[ec2-user@ip-172-31-38-150 downloads]$ wget https://go.nagios.org/l
24-09-17/6kqcx
```

9. Use wget to download the source zip files.

```
wget https://go.nagios.org/l/975333/2024-09-17/6kqcx
wget https://nagios-plugins.org/download/nagios-plugins-2.4.11.tar.gz
```

```
[ec2-user@ip-172-31-38-150 downloads]$ wget https:
24-09-17/6kqcx
--2024-10-06 11:23:50--  https://go.nagios.org/l/9
Resolving go.nagios.org (go.nagios.org)... 3.92.12
72.219, ...
Connecting to go.nagios.org (go.nagios.org)|3.92.1
HTTP request sent, awaiting response... 302 Found
Location: http://assets.nagios.com/downloads/nagio
.tar.gz?utm_source=Nagios.org&utm_content=Download
.5+Download+&pi_content=1e9662c93afb2ed6bd2e3f3cc3
```

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

```
tar zxvf nagios-4.5.5.tar.gz
cd nagios-4.5
```

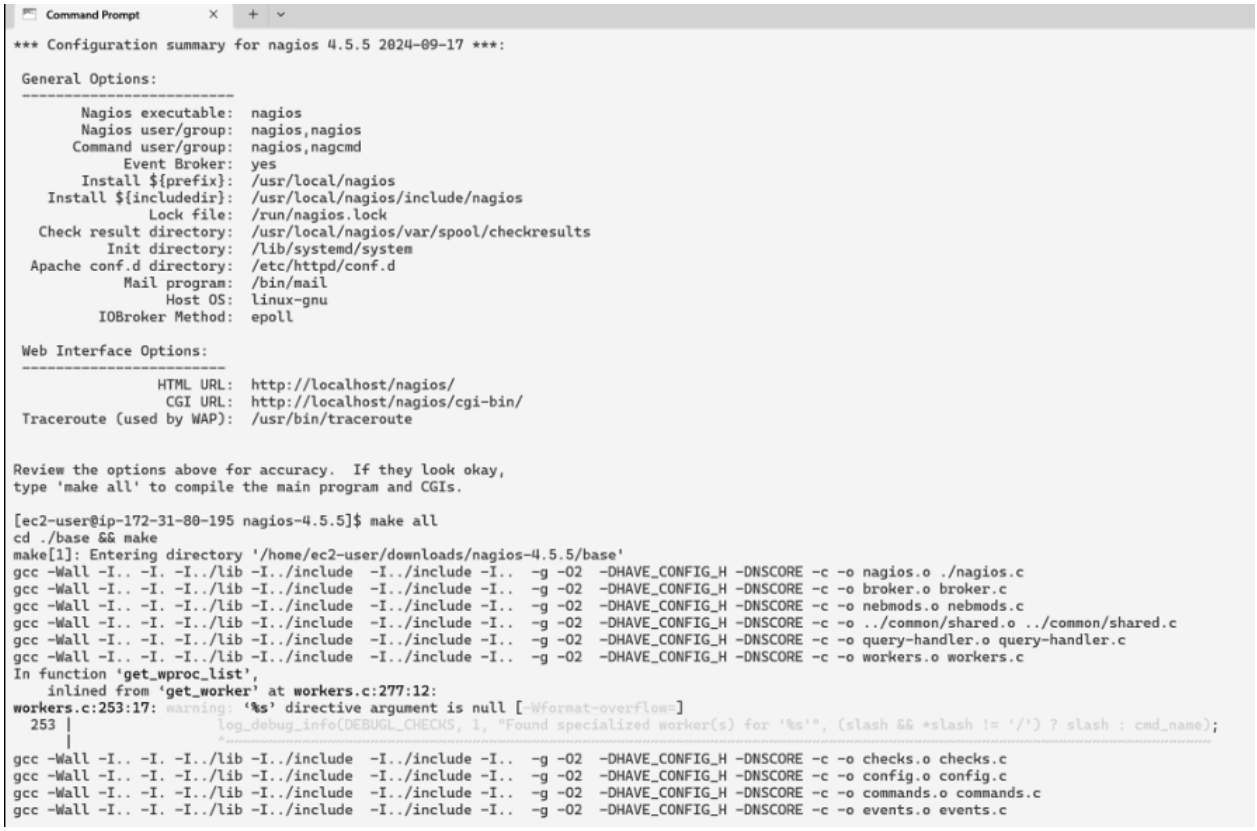
```
24-09-17/6kqcx
--2024-10-06 11:23:50--  https://go.nagios.org/l/975333/2024-09-17/6kqcx
Resolving go.nagios.org (go.nagios.org)... 3.92.120.28, 52.54.96.194, 3.215
72.219, ...
Connecting to go.nagios.org (go.nagios.org)|3.92.120.28|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: http://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5
.tar.gz?utm_source=Nagios.org&utm_content=Download+Form&utm_campaign=Core+4
.5+Download+&pi_content=1e9662c93afb2ed6bd2e3f3cc38771a7f01125e969f2a75b0e2
4439d4a81d8 [following]
--2024-10-06 11:23:50--  http://assets.nagios.com/downloads/nagioscore/rele
es/nagios-4.5.5.tar.gz?utm_source=Nagios.org&utm_content=Download+Form&utm_
mpaign=Core+4.5.5+Download+&pi_content=1e9662c93afb2ed6bd2e3f3cc38771a7f011
e969f2a75b0e2254439d4a81d8
Resolving assets.nagios.com (assets.nagios.com)... 45.79.49.120, 2600:3c00:
03c:92ff:fef7:45ce
Connecting to assets.nagios.com (assets.nagios.com)|45.79.49.120|:80... con
```

11. Run the configuration script with the same group name you previously created.

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

12. Compile the source code.

```
make all
```



13. Install binaries, init script and sample config files. Lastly, set permissions on the external command directory.

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

```
[ec2-user@ip-172-31-38-150 nagios-4.4.6]$ sudo make install-webconf
/usr/bin/install -c -m 644 sample-config/httpd.conf /etc/httpd/conf.d/nagio
s.conf
if [ 0 -eq 1 ]; then \
        ln -s /etc/httpd/conf.d/nagios.conf /etc/apache2/sites-enabled/nagi
os.conf; \
fi

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

[ec2-user@ip-172-31-38-150 nagios-4.4.6]$
```

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

```
[ec2-user@ip-172-31-38-150 nagios-plugins-2.4.11]$ sudo yum install openssl
evel
Last metadata expiration check: 0:13:31 ago on Sun Oct  6 11:15:04 2024.
Dependencies resolved.
================================================================================
================================================
 Package                    Architecture         Version
           Repository                  Size
================================================================================
================================================
Installing:
 openssl-devel              x86_64               1:3.0.8-1.amzn2023.0.14
           amazonlinux                3.0 M

Transaction Summary
```

14. Configure the web interface.

```
sudo make install-webconf
```

```
[ec2-user@ip-172-31-38-150 nagios-4.4.6]$ sudo make install-webconf
/usr/bin/install -c -m 644 sample-config/httpd.conf /etc/httpd/conf.d/nagio
s.conf
if [ 0 -eq 1 ]; then \
        ln -s /etc/httpd/conf.d/nagios.conf /etc/apache2/sites-enabled/nagi
os.conf; \
fi

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

[ec2-user@ip-172-31-38-150 nagios-4.4.6]$
```

15. Create a nagiosadmin account for nagios login along with password. You'll have to specify the password twice.

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

```
[ec2-user@ip-172-31-38-150 nagios-4.4.6]$ sudo htpasswd -c /usr/local/nagic
s/etc/htpasswd.users nagiosadmin          sudo htpasswd -c /usr/local/nagic
s/etc/htpasswd.users nagiosadmin
New password:
Re-type new password:
Adding password for user nagiosadmin
[ec2-user@ip-172-31-38-150 nagios-4.4.6]$
```

16. Restart Apache

```
sudo service httpd restart
```

```
[ec2-user@ip-172-31-38-150 nagios-4.4.6]$ sudo service httpd restart
Redirecting to /bin/systemctl restart httpd.service
[ec2-user@ip-172-31-38-150 nagios-4.4.6]$
```

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

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

```
[ec2-user@ip-172-31-38-150 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
```

18. Compile and install plugins

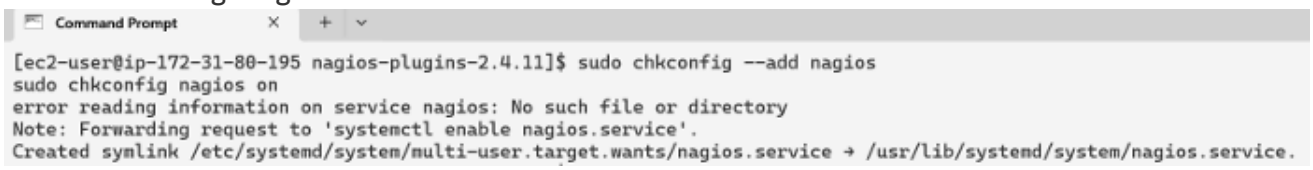
```
cd nagios-plugins-2.4.11
./configure --with-nagios-user=nagios --with-nagios-group=nagios
make
sudo make install
```

```
[ec2-user@ip-172-31-38-150 downloads]$ cd nagios-plugins-2.4.11
[ec2-user@ip-172-31-38-150 nagios-plugins-2.4.11]$ ./configure --with-nagios-user=nagios
 --with-nagios-group=nagios                        /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
```

19. Start Nagios

Add Nagios to the list of system services

```
sudo chkconfig --add nagios
sudo chkconfig nagios on
```



Verify the sample configuration files

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

```
[ec2-user@ip-172-31-38-150 nagios-plugins-2.4.11]$ sudo /usr/local/nagios/bin/nagios -v
/usr/local/nagios/etc/nagios.cfg                    sudo /usr/local/nagios/bin/nagios -v
/usr/local/nagios/etc/nagios.cfg

Nagios Core 4.4.6
Copyright (c) 2009-present Nagios Core Development Team and Community Contributors
Copyright (c) 1999-2009 Ethan Galstad
Last Modified: 2020-04-28
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.
```

If there are no errors, you can go ahead and start Nagios.

```
sudo service nagios start
```

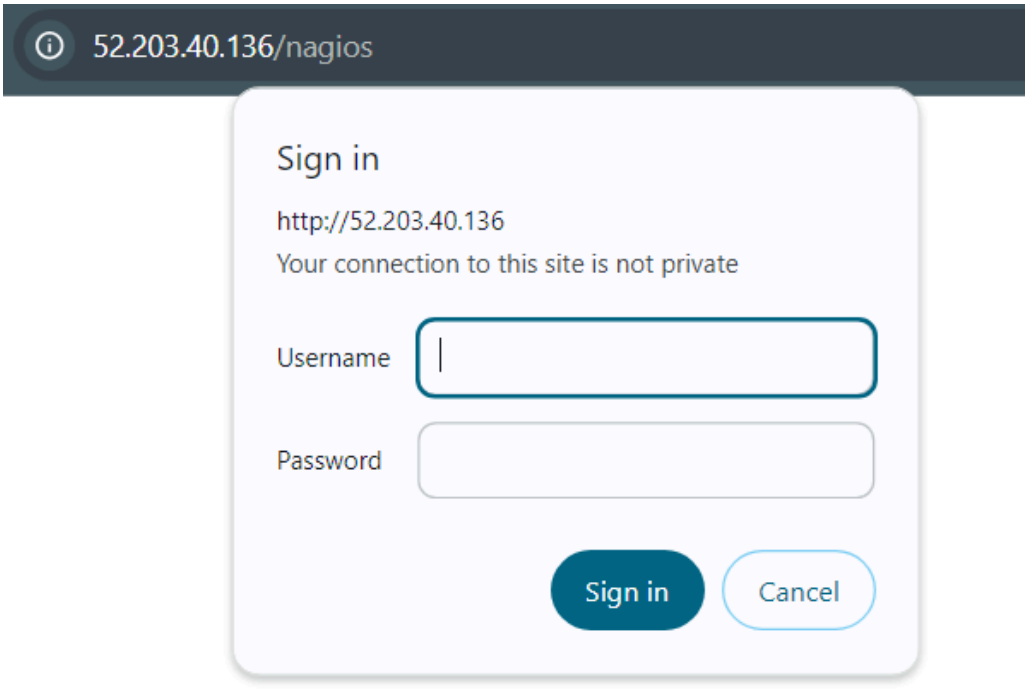
```
Things look okay - No serious problems were detected during the pre-flight check
[ec2-user@ip-172-31-38-150 nagios-plugins-2.4.11]$ sudo service nagios start
Redirecting to /bin/systemctl start nagios.service
[ec2-user@ip-172-31-38-150 nagios-plugins-2.4.11]$
```

20. Check the status of Nagios

```
sudo systemctl status nagios
Redirecting to /bin/systemctl start nagios.service
[ec2-user@ip-172-31-38-150 nagios-plugins-2.4.11]$ sudo systemctl status nagios
● nagios.service - Nagios Core 4.4.6
     Loaded: loaded (/usr/lib/systemd/system/nagios.service; enabled; preset: disabled
     Active: active (running) since Sun 2024-10-06 11:51:46 UTC; 1min 33s ago
       Docs: https://www.nagios.org/documentation
   Main PID: 89956 (nagios)
      Tasks: 6 (limit: 1112)
     Memory: 2.4M
        CPU: 36ms
     CGroup: /system.slice/nagios.service
             ├─89956 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─89957 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/na
os.qh
             ├─89958 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/na
os.qh
             ├─89959 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/na
os.qh
             ├─89960 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/na
os.qh
             └─89961 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg

Oct 06 11:51:46 ip-172-31-38-150.ec2.internal nagios[89956]: qh: Socket '/usr/local/na
os/var/rw/nagios.qh' successfull
```

21. Go back to EC2 Console and copy the Public IP address of this instance

22. Open up your browser and look for **http://<your_public_ip_address>/nagios**



Enter username as nagiosadmin and password which you set in Step 16.

23. After entering the correct credentials, you will see this page.

This means that Nagios was correctly installed and configured with its plugins so far.

Conclusion: We have successfully installed and configured Nagios Core, Nagios Plugins, and NRPE on a Linux machine. This enables us to effectively manage system performance and proactively address potential issues.

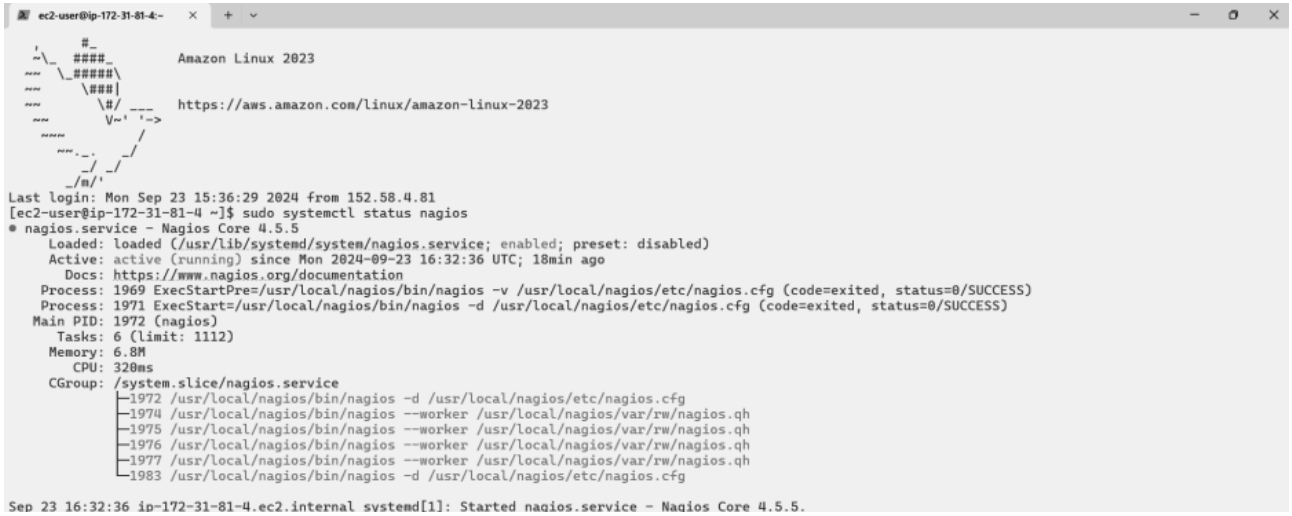
Advanced DevOps
Lab Experiment 10

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

Nagios. **Steps:**

Prerequisites: AWS Free Tier, Nagios Server running on Amazon Linux Machine.

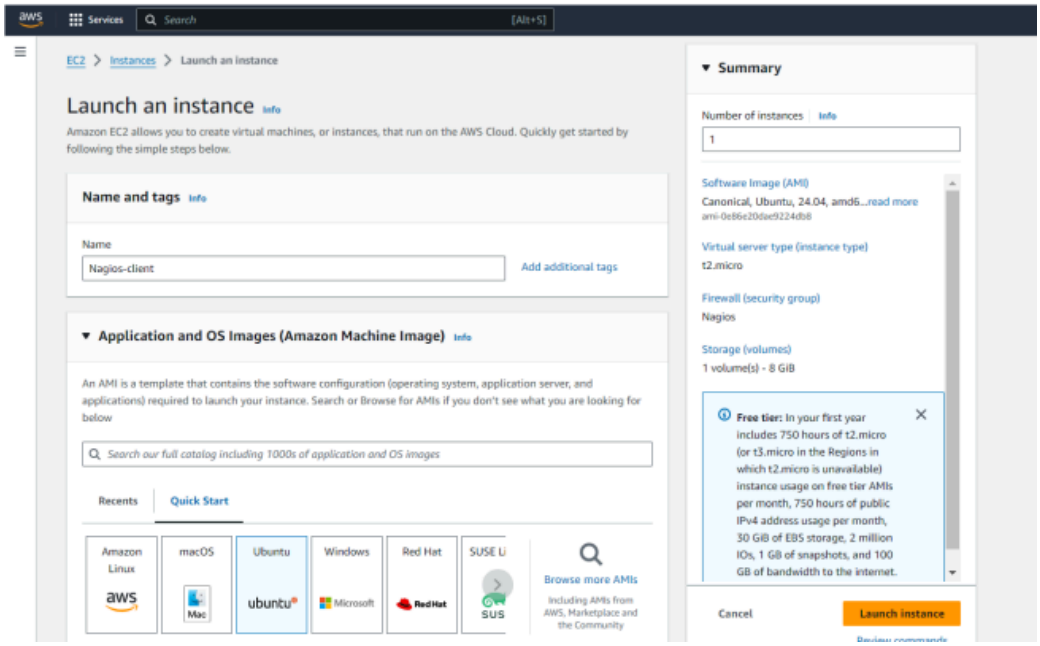
1. To Confirm that Nagios is running **on the server side**, run this
   sudo systemctl status nagios on the "NAGIOS HOST".



   You can proceed if you get this message.

2. Before we begin,
   To monitor a Linux machine, create an Ubuntu 20.04 server EC2 Instance in AWS.

   Provide it with the same security group as the Nagios Host and name it
   'linux-client' alongside the host.



For now, leave this machine as is, and go back to your nagios HOST machine.

3. On the server, run this command

```
ps -ef | grep nagios
```

```
Last login: Sat Oct  5 16:58:17 2024 from 42.111.112.18
[ec2-user@ip-172-31-43-65 ~]$ ps -ef | grep nagios
nagios     97412       1  0 17:34 ?        00:00:00 /usr/local/nagios/bin/nagios -d /usr/lo
nagios     97413   97412  0 17:34 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /
s.qh
nagios     97414   97412  0 17:34 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /
s.qh
nagios     97415   97412  0 17:34 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /
s.qh
nagios     97416   97412  0 17:34 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /
s.qh
nagios     97417   97412  0 17:34 ?        00:00:00 /usr/local/nagios/bin/nagios -d /usr/lo
ec2-user   98423   98399  0 17:51 pts/2    00:00:00 grep --color=auto nagios
```

4. Become a root user and create 2 folders

```
sudo su
mkdir /usr/local/nagios/etc/objects/monitorhosts
mkdir /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
```

```
[ec2-user@ip-172-31-43-65 ~]$ sudo su
mkdir /usr/local/nagios/etc/objects/monitorhosts
mkdir /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
[root@ip-172-31-43-65 ec2-user]#
```

5. Copy the sample localhost.cfg file to linuxhost folder

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

```
[root@ip-172-31-81-4 ec2-user]# cp /usr/local/nagios/etc/objects/localhost.cfg /usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg
[root@ip-172-31-81-4 ec2-user]#
```

6. Open linuxserver.cfg using nano and make the following changes

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

Change the hostname to linuxserver (EVERYWHERE ON THE FILE) Change address to the public IP address of your **LINUX CLIENT**.

```
  GNU nano 5.8                          /usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg
##############################################################################
# LOCALHOST.CFG - SAMPLE OBJECT CONFIG FILE FOR MONITORING THIS MACHINE
#
#
# NOTE: This config file is intended to serve as an *extremely* simple
#       example of how you can create configuration entries to monitor
#       the local (Linux) machine.
#
##############################################################################



##############################################################################
#
# HOST DEFINITION
#
##############################################################################

# Define a host for the local machine

define host {

    use                     linux-server            ; Name of host template to use
                                                    ; This host definition will inherit all variables that are defined
                                                    ; in (or inherited by) the linux-server host template definition.
    host_name               localhost
    alias                   localhost
    address                 127.0.0.1
}



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

Change hostgroup_name under hostgroup to linux-servers1

Everywhere else on the file, change the hostname to linuxserver instead of localhost.

7. Open the Nagios Config file and add the following line

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

```
##Add this line
cfg_dir=/usr/local/nagios/etc/objects/monitorhosts/
```

8. Verify the configuration files

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

```
Running pre-flight check on configuration data...

Checking objects...
        Checked 8 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-172-31-43-65 ec2-user]#
```

You are good to go if there are no errors.

9. Restart the nagios service

```
service nagios restart
```

```
[root@ip-172-31-81-4 ec2-user]# service nagios restart
Redirecting to /bin/systemctl restart nagios.service
[root@ip-172-31-81-4 ec2-user]# sudo systemctl status nagi
```

Now it is time to switch to the client machine.

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

11. Make a package index update and install gcc, nagios-nrpe-server and the plugins.

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

```
ubuntu@ip-172-31-33-76:~$ 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://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:6 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [382 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Components [35.0 kB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 c-n-f Metadata [8328 B]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [537 kB]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [132 kB]
Get:16 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 c-n-f Metadata [8860 B]
Get:17 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [384 kB]
Get:18 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [159 kB]
Get:19 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [45.0 kB]
Get:20 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 c-n-f Metadata [14.9 kB]
Get:21 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Packages [14.4 kB]
Get:22 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse Translation-en [3608 B]
Get:23 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [212 B]
```

12. Open nrpe.cfg file to make changes.

```
sudo nano /etc/nagios/nrpe.cfg
```

Under allowed_hosts, add your nagios host IP address like so

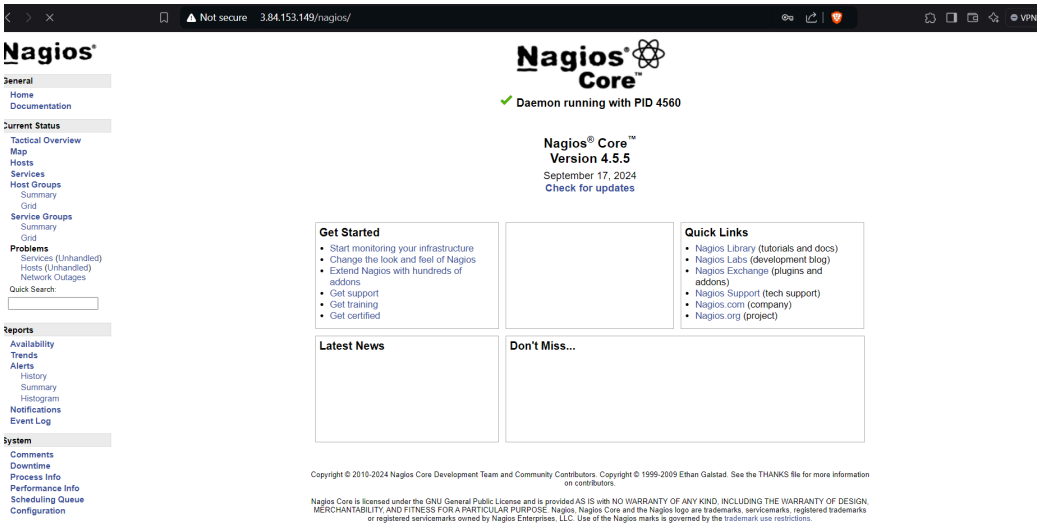
13. Restart the NRPE server

```
sudo systemctl restart nagios-nrpe-server
```

```
ubuntu@ip-172-31-83-152:~$ sudo nano /etc/nagios/nrpe.cfg

ubuntu@ip-172-31-83-152:~$ sudo systemctl restart nagios-nrpe-server
ubuntu@ip-172-31-83-152:~$
```

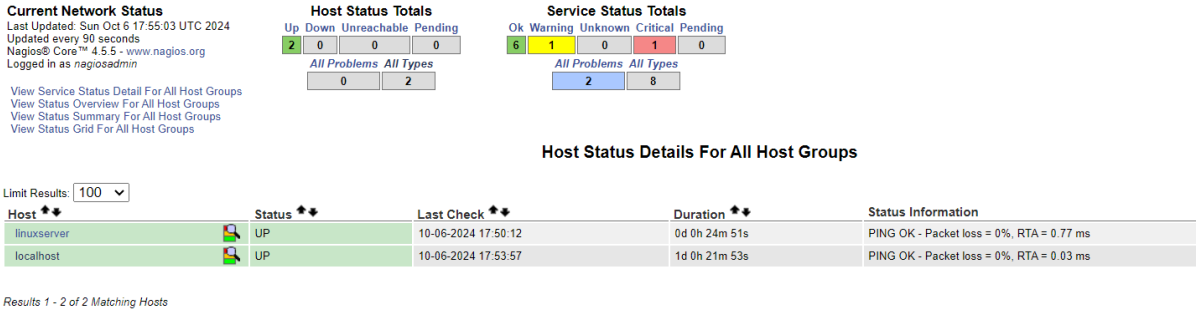
14. Now, check your nagios dashboard and you'll see a new host being added.

    Click on Hosts.



NAME: LAUKIK PADGAONKAR DIV: D15C ROLL NO: 37

Click on linuxserver to see the host details



Current Network Status
Last Updated: Sun Oct 6 17:55:03 UTC 2024
Updated every 90 seconds
Nagios® Core™ 4.5.5 - www.nagios.org
Logged in as *nagiosadmin*

View Service Status Detail For All Host Groups
View Status Overview For All Host Groups
View Status Summary For All Host Groups
View Status Grid For All Host Groups

Host Status Totals

Up	Down	Unreachable	Pending
2	0	0	0

All Problems	All Types
0	2

Service Status Totals

Ok	Warning	Unknown	Critical	Pending
6	1	0	1	0

All Problems	All Types
2	8

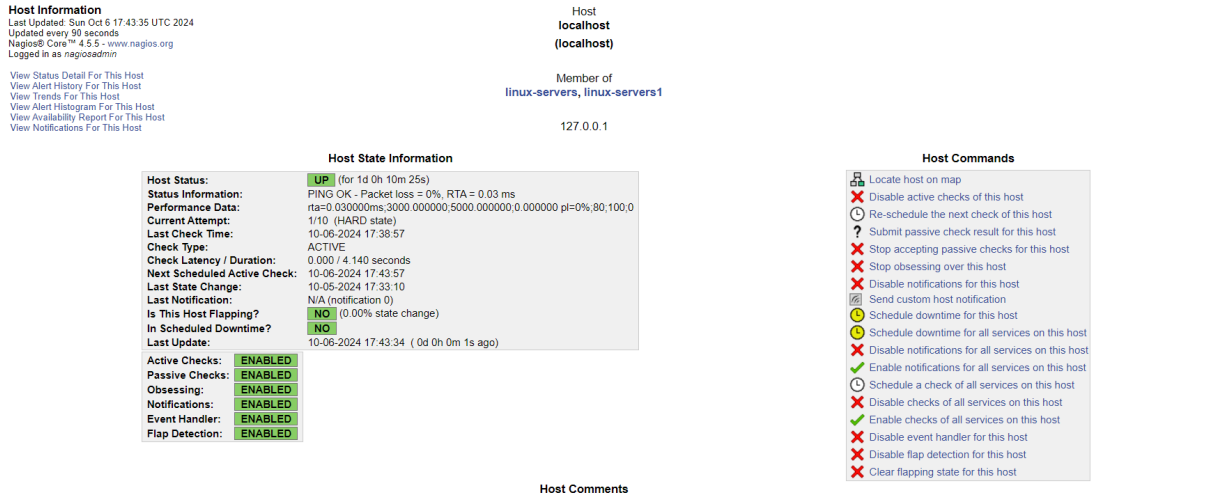
Host Status Details For All Host Groups

Limit Results: 100

Host	Status	Last Check	Duration	Status Information
linuxserver	UP	10-06-2024 17:50:12	0d 0h 24m 51s	PING OK - Packet loss = 0%, RTA = 0.77 ms
localhost	UP	10-06-2024 17:53:57	1d 0h 21m 53s	PING OK - Packet loss = 0%, RTA = 0.03 ms

Results 1 - 2 of 2 Matching Hosts

You can click Services to see all services and ports being monitored.



Host Information
Last Updated: Sun Oct 6 17:43:35 UTC 2024
Updated every 90 seconds
Nagios® Core™ 4.5.5 - www.nagios.org
Logged in as *nagiosadmin*

View Status Detail For This Host
View Alert History For This Host
View Trends For This Host
View Alert Histogram For This Host
View Availability Report For This Host
View Notifications For This Host

Host
localhost
(localhost)

Member of
linux-servers, linux-servers1

127.0.0.1

Host State Information

Host Status:	UP (for 1d 0h 10m 25s)
Status Information:	PING OK - Packet loss = 0%, RTA = 0.03 ms
Performance Data:	rta=0.030000ms;3000.000000;5000.000000;0.000000 pl=0%;80;100;0
Current Attempt:	1/10 (HARD state)
Last Check Time:	10-06-2024 17:38:57
Check Type:	ACTIVE
Check Latency / Duration:	0.000 / 4.140 seconds
Next Scheduled Active Check:	10-06-2024 17:43:57
Last State Change:	10-05-2024 17:33:10
Last Notification:	N/A (notification 0)
Is This Host Flapping?	NO (0.00% state change)
In Scheduled Downtime?	NO
Last Update:	10-06-2024 17:43:34 (0d 0h 0m 1s ago)

Active Checks:	ENABLED
Passive Checks:	ENABLED
Obsessing:	ENABLED
Notifications:	ENABLED
Event Handler:	ENABLED
Flap Detection:	ENABLED

Host Commands

Locate host on map
Disable active checks of this host
Re-schedule the next check of this host
Submit passive check result for this host
Stop accepting passive checks for this host
Stop obsessing over this host
Disable notifications for this host
Send custom host notification
Schedule downtime for this host
Schedule downtime for all services on this host
Disable notifications for all services on this host
Enable notifications for all services on this host
Schedule a check of all services on this host
Disable checks of all services on this host
Enable checks of all services on this host
Disable event handler for this host
Disable flap detection for this host
Clear flapping state for this host

Host Comments

Add a new comment Delete all comments

Entry Time	Author	Comment	Comment ID	Persistent	Type	Expires	Actions
This host has no comments associated with it							

Current Network Status
Last Updated: Sun Oct 6 17:58:02 UTC 2024
Updated every 90 seconds
Nagios® Core™ 4.5.5 - www.nagios.org
Logged in as *nagiosadmin*

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

Host Status Totals

Up	Down	Unreachable	Pending
2	0	0	0

All Problems	All Types
0	2

Service Status Totals

Ok	Warning	Unknown	Critical	Pending
6	1	0	1	0

All Problems	All Types
2	8

Service Status Details For All Hosts

Limit Results: 100

Host	Service	Status	Last Check	Duration	Attempt	Status Information
localhost	Current Load	OK	10-06-2024 17:56:27	1d 0h 24m 52s	1/4	OK - load average: 0.00, 0.00, 0.00
	Current Users	OK	10-06-2024 17:57:42	1d 0h 24m 14s	1/4	USERS OK - 6 users currently logged in
	HTTP	WARNING	10-06-2024 17:53:57	0d 0h 19m 5s	4/4	HTTP WARNING: HTTP/1.1 403 Forbidden - 319 bytes in 0.001 second response time
	PING	OK	10-06-2024 17:55:12	1d 0h 22m 59s	1/4	PING OK - Packet loss = 0%, RTA = 0.03 ms
	Root Partition	OK	10-06-2024 17:57:04	1d 0h 22m 22s	1/4	DISK OK - free space: / 5567 MiB (68.59% inode=98%):
	SSH	OK	10-06-2024 17:53:19	1d 0h 21m 44s	1/4	SSH OK - OpenSSH_8.7 (protocol 2.0)
	Swap Usage	CRITICAL	10-06-2024 17:54:34	1d 0h 31m 7s	4/4	SWAP CRITICAL - 0% free (0 MB out of 0 MB) - Swap is either disabled, not present, or of zero size.
	Total Processes	OK	10-06-2024 17:56:22	1d 0h 20m 29s	1/4	PROCS OK: 39 processes with STATE = RSZDT

Results 1 - 8 of 8 Matching Services

As you can see, we have our linuxserver up and running. It is showing critical status on HTTP due to permission errors and swap because there is no partition created.

In this case, we have monitored -

Servers: 1 linux server
Services: swap
Ports: 22, 80 (ssh, http)
Processes: User status, Current load, total processes, root partition, etc.

Recommended Cleanup

- Terminate both of your EC-2 instances to avoid charges.
- Delete the security group if you created a new one (it won't affect your bill, you may avoid it)

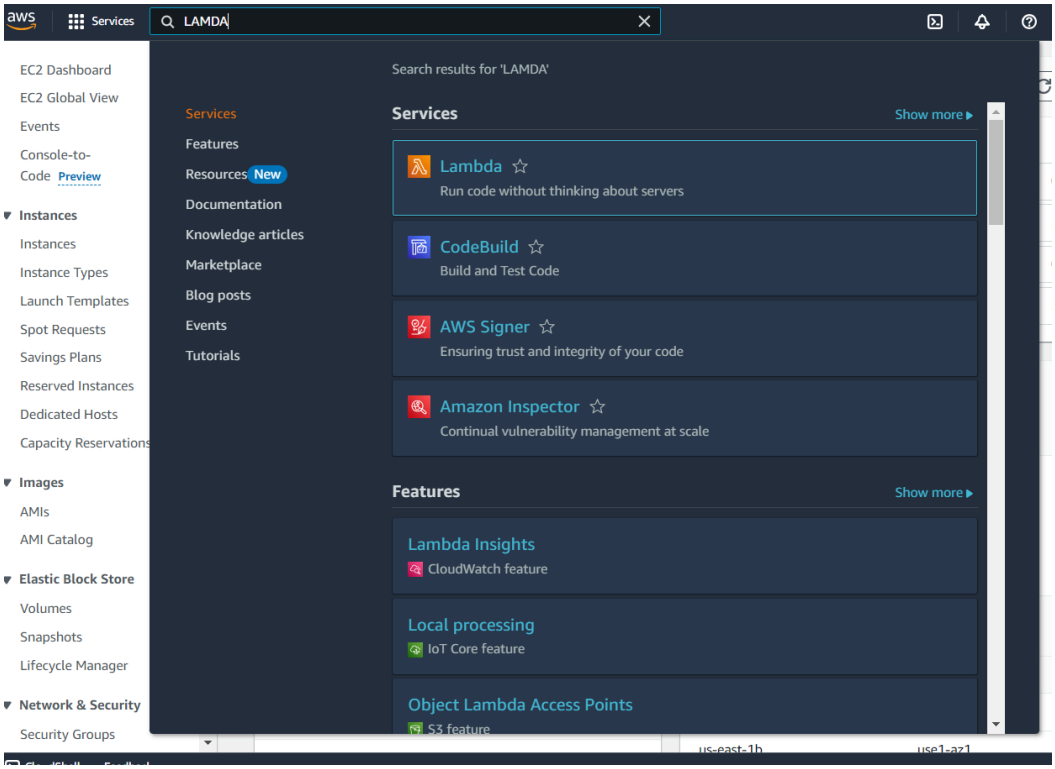
Conclusion: In conclusion, the experiment focused on monitoring ports, services, and a Linux server using Nagios. Through the step-by-step process, we successfully configured Nagios to monitor essential network services on the Linux server. By setting up both the Nagios host and client, we were able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

Name:Laukik Padgaonkar Std:D15C ROLL:37

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

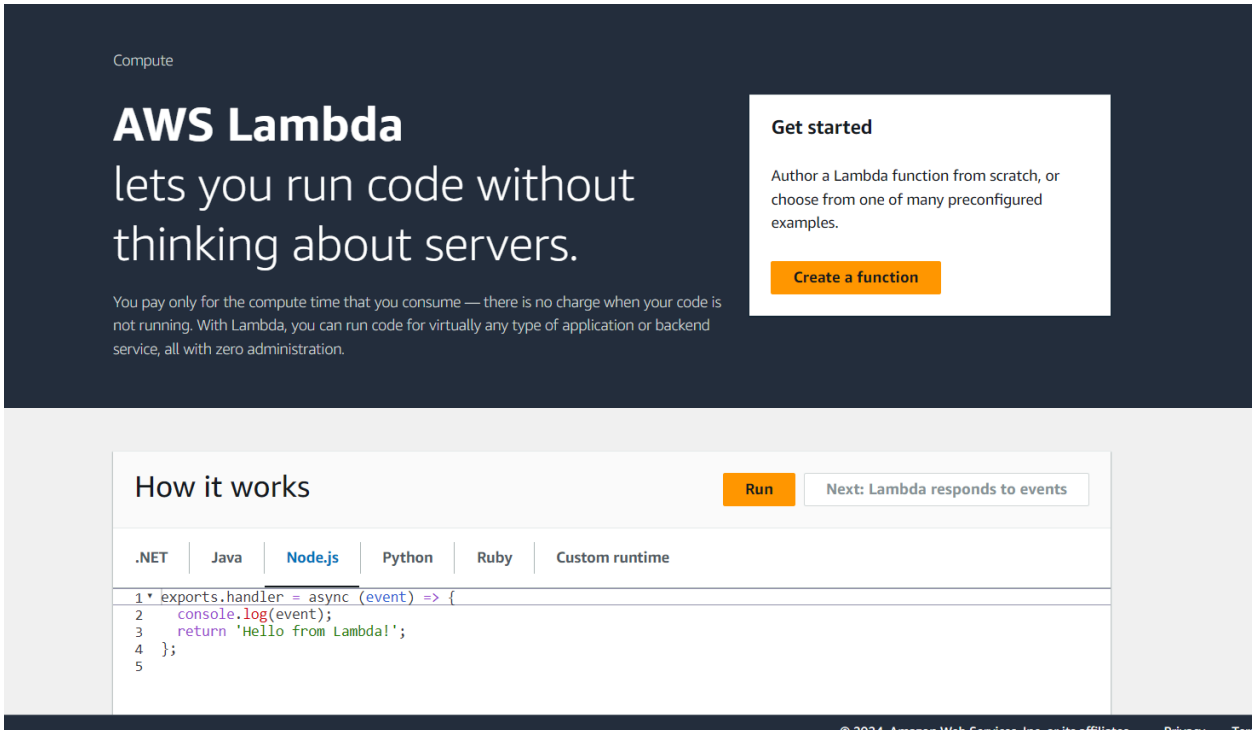
Step 1: Accessing AWS

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



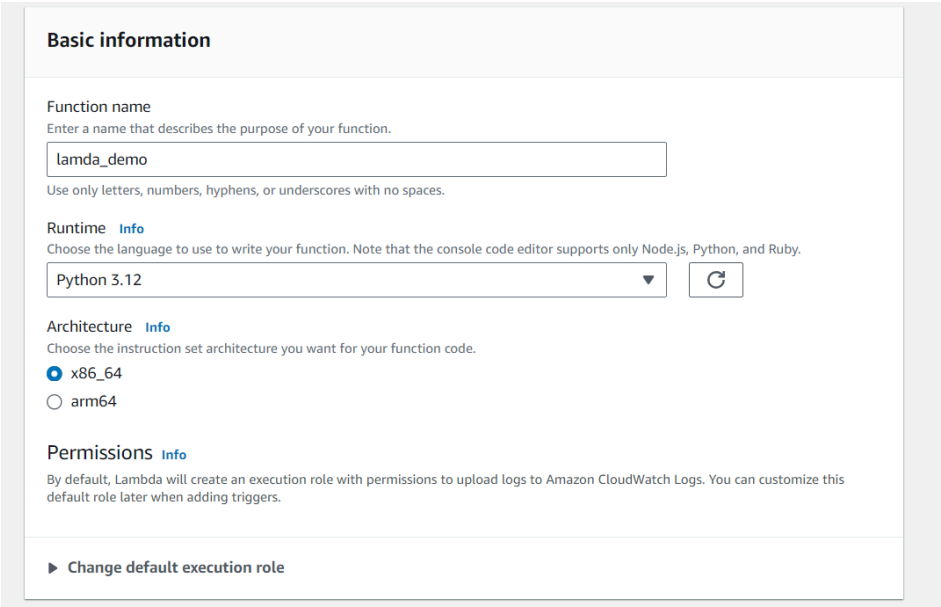
Step 2: Creating a New Lambda Function

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



Step 3: Configuring Basic Settings

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



Permissions Info

By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▼ **Change default execution role**

Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.

- (●) Create a new role with basic Lambda permissions
- ( ) Use an existing role
- ( ) Create a new role from AWS policy templates

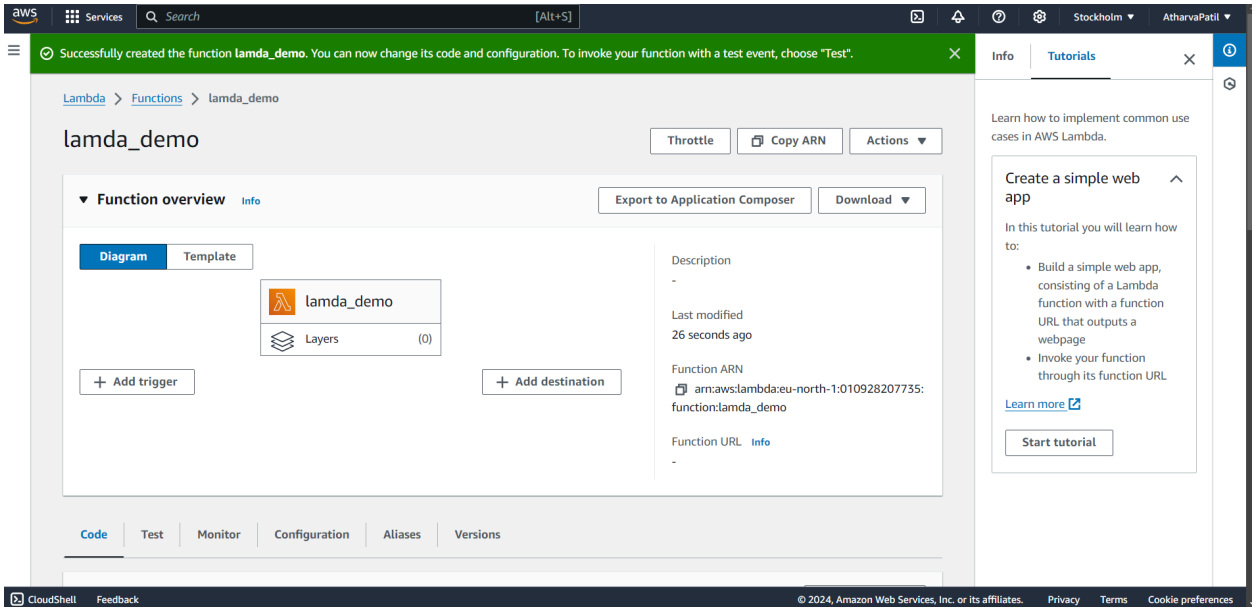
ⓘ Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.

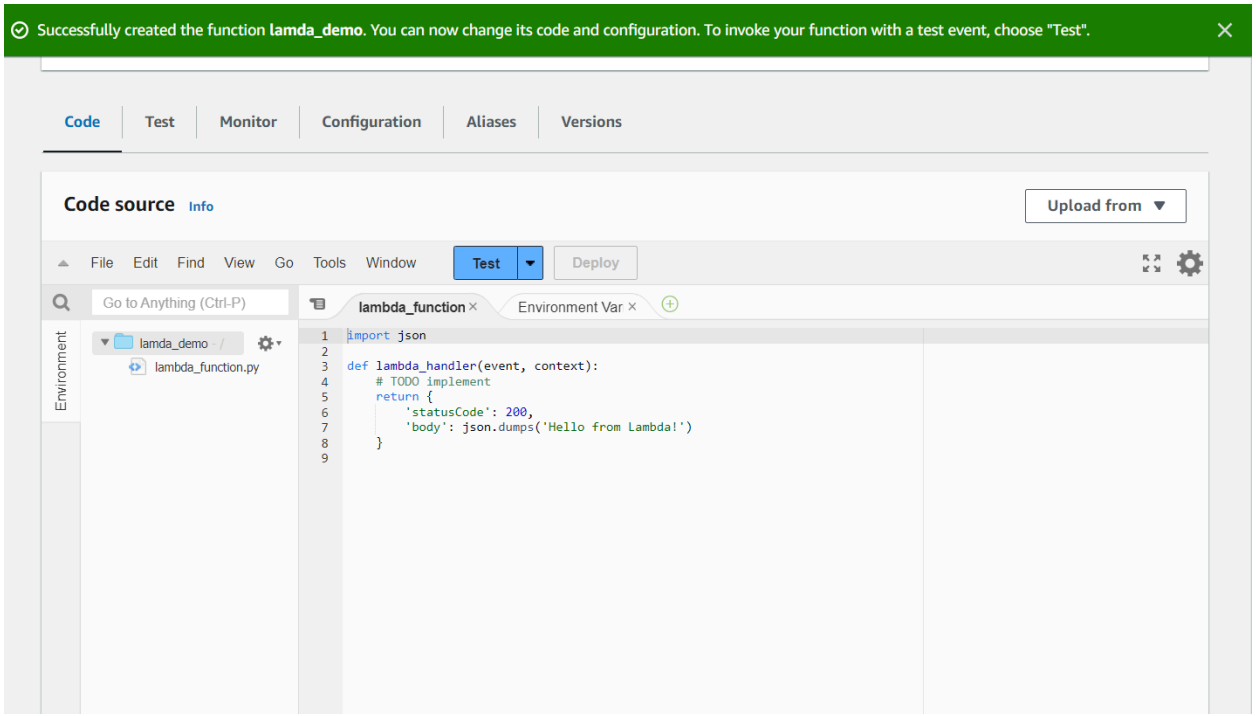
Lambda will create an execution role named ATHARV_LAMDA-role-0u7c9ooi, with permission to upload logs to Amazon CloudWatch Logs.

▶ **Additional Configurations**

Use additional configurations to set up code signing, function URL, tags, and Amazon VPC access for your function.

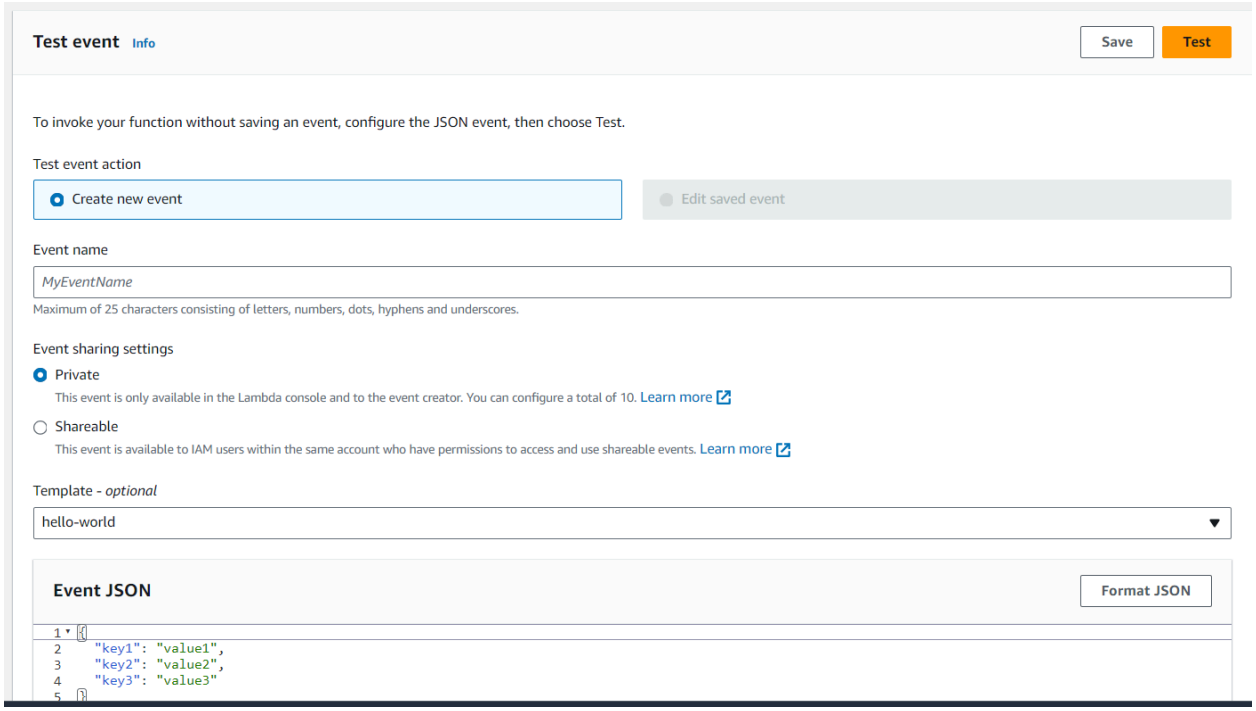
Cancel | **Create function**

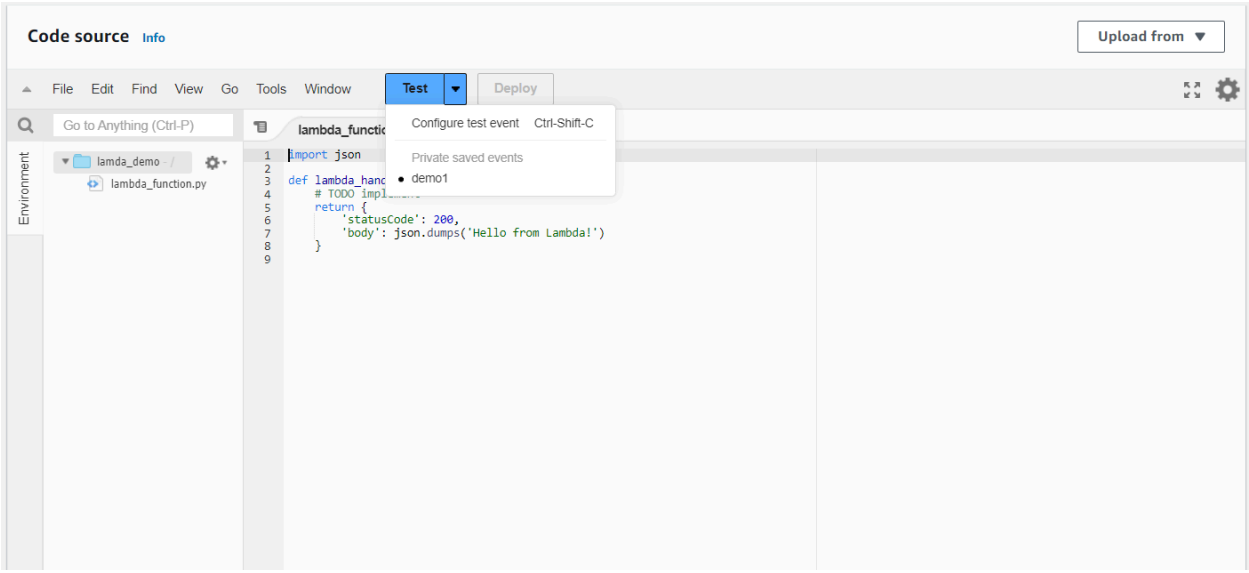


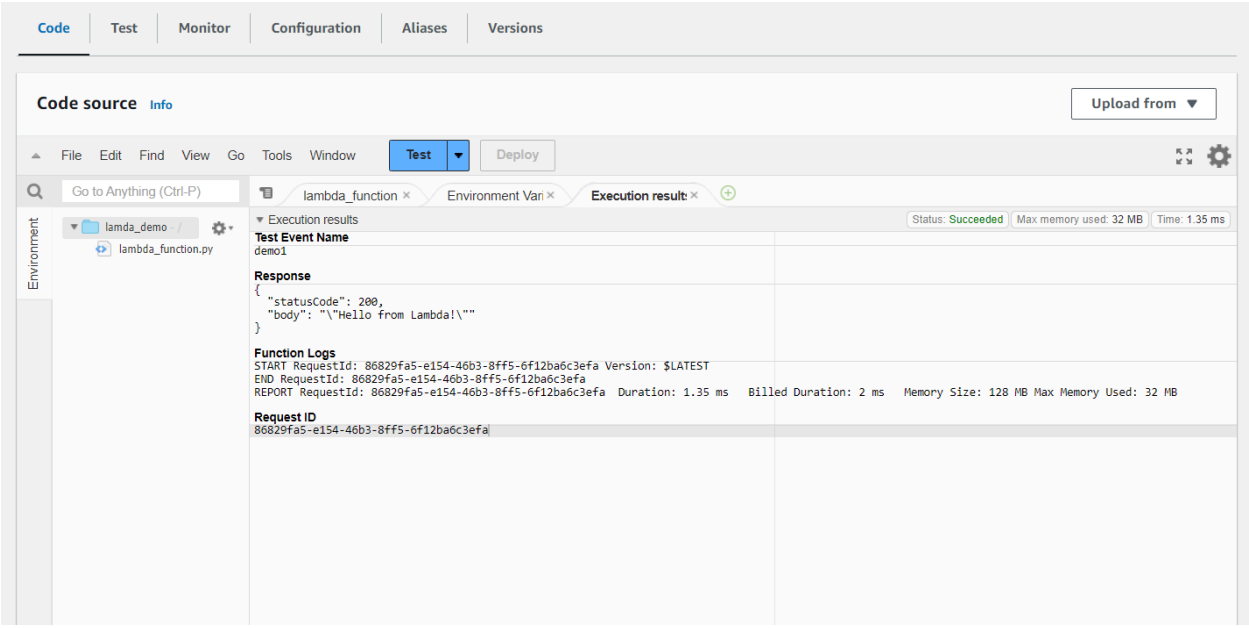


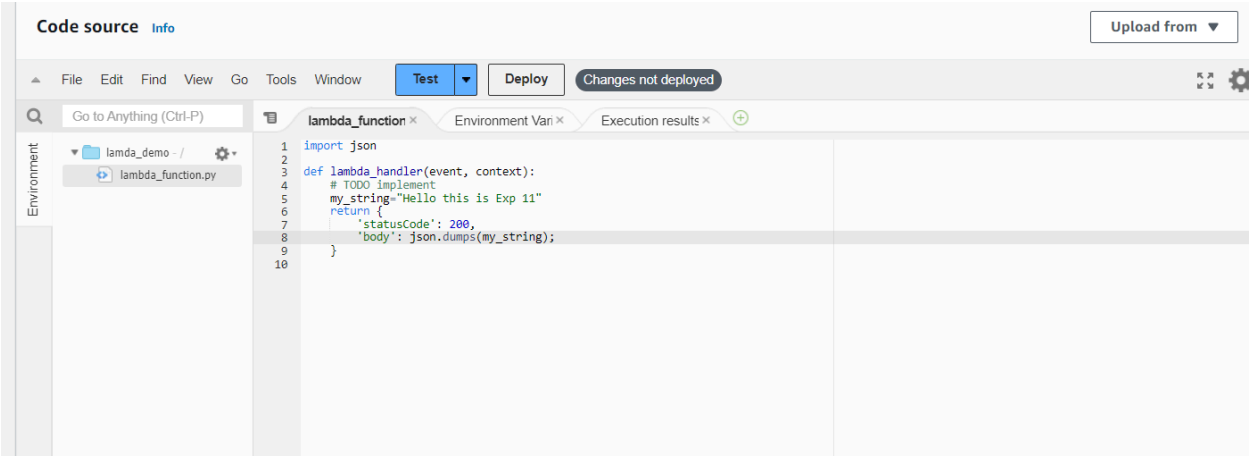
Step 4: Testing the Function

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



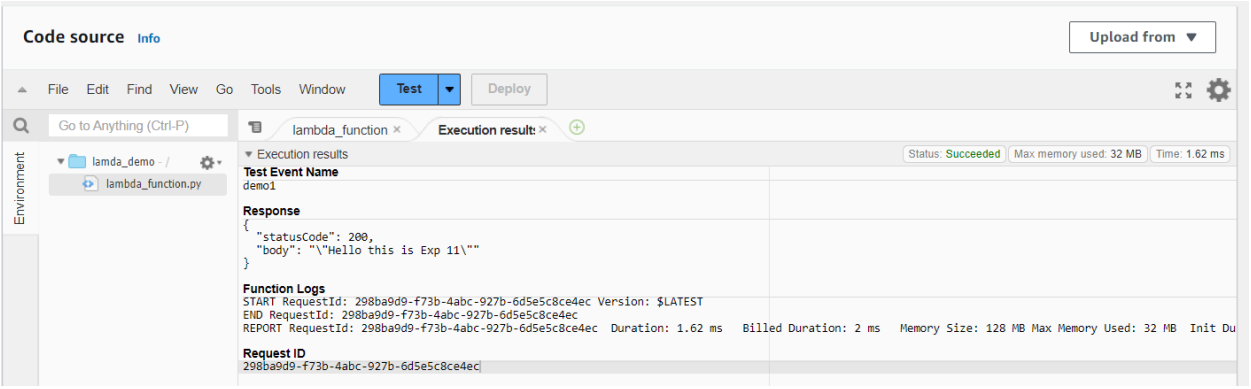

Code source Info
Upload from
File Edit Find View Go Tools Window
Test
Deploy
Configure test event Ctrl-Shift-C
Private saved events
demo1
Go to Anything (Ctrl-P)
lamda_demo
lambda_function.py
Environment
import json
def lambda_hand
# TODO impl
return {
'statusCode': 200,
'body': json.dumps('Hello from Lambda!')
}


Code
Test
Monitor
Configuration
Aliases
Versions
Code source Info
Upload from
File Edit Find View Go Tools Window
Test
Deploy
Go to Anything (Ctrl-P)
lamda_demo
lambda_function.py
Environment
lambda_function
Environment Vari
Execution result:
Execution results
Status: Succeeded
Max memory used: 32 MB
Time: 1.35 ms
Test Event Name
demo1
Response
{
"statusCode": 200,
"body": "\"Hello from Lambda!\""
}
Function Logs
START RequestId: 86829fa5-e154-46b3-8ff5-6f12ba6c3efa Version: $LATEST
END RequestId: 86829fa5-e154-46b3-8ff5-6f12ba6c3efa
REPORT RequestId: 86829fa5-e154-46b3-8ff5-6f12ba6c3efa Duration: 1.35 ms Billed Duration: 2 ms Memory Size: 128 MB Max Memory Used: 32 MB
Request ID
86829fa5-e154-46b3-8ff5-6f12ba6c3efa



Step 5: Running the Test

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

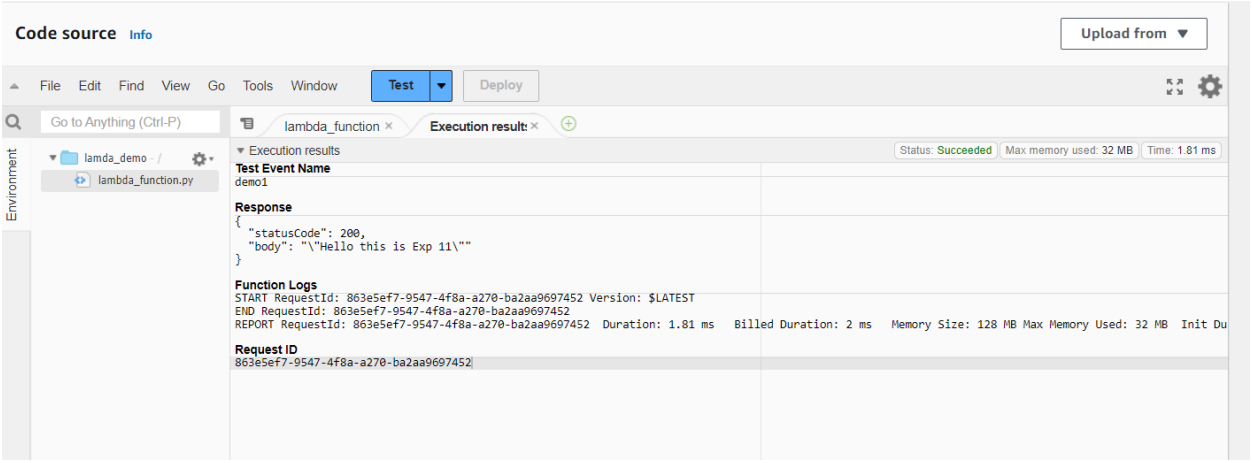


Step 6: Editing and Deploying the Code

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

Step 7: Final Testing

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

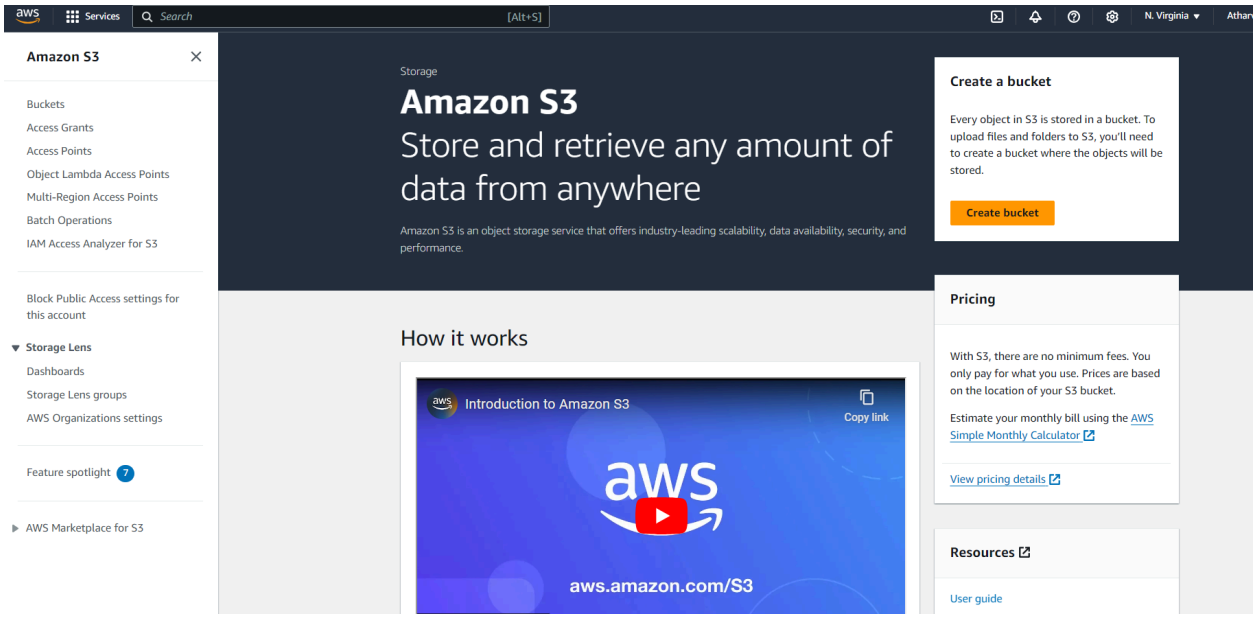


Conclusion:

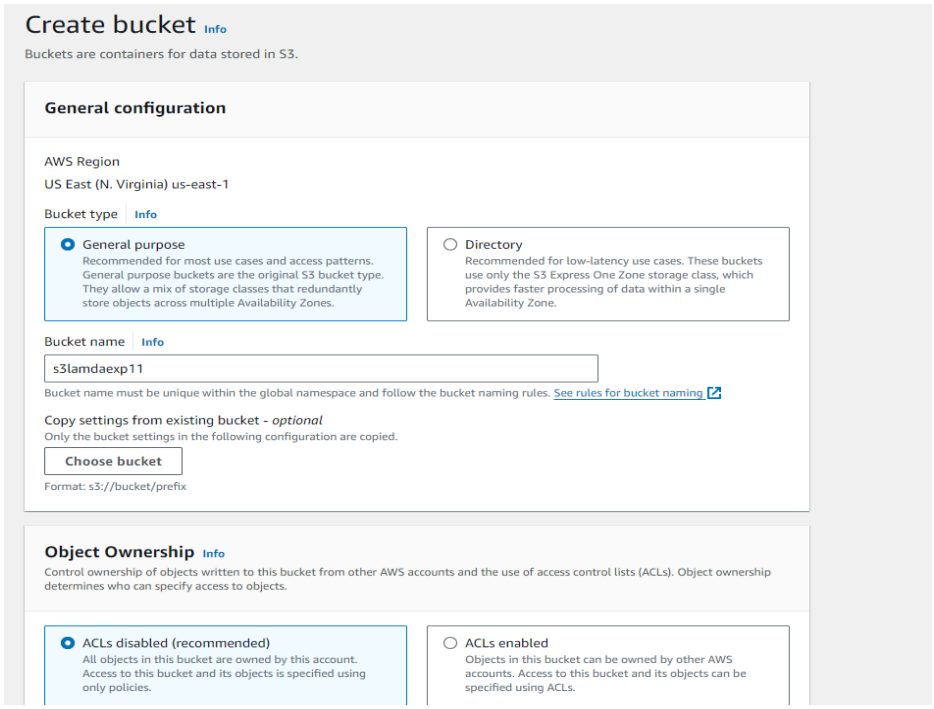
AWS Lambda simplifies the deployment of serverless applications by eliminating the need for server management, automatic scaling, and pay-as-you-go pricing. Creating Lambda functions with Python, Node.js, or Java is straightforward, making it accessible for developers to build efficient, scalable, and cost-effective solutions. Its ability to trigger functions based on events from various AWS services makes Lambda a powerful tool for automating cloud workflows.

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

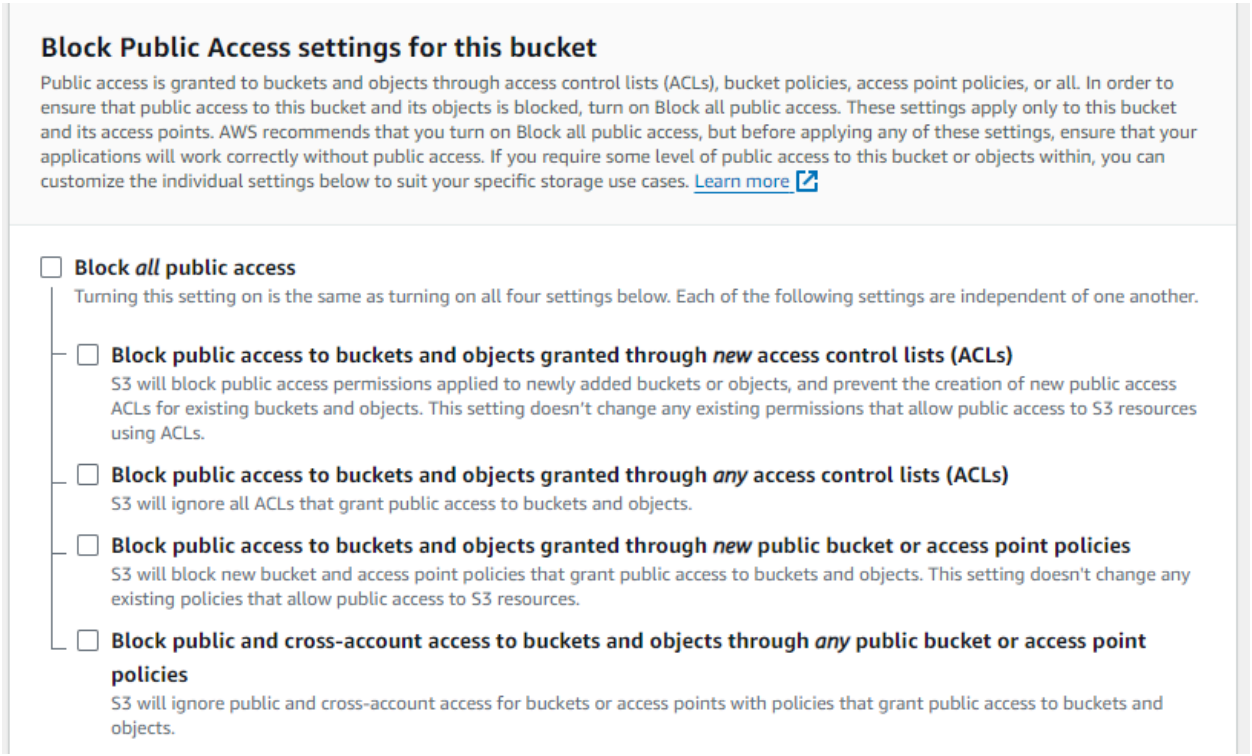
Step 1: Create a s3 bucket. 1) Search for S3 bucket in the services search. Then click on create bucket.



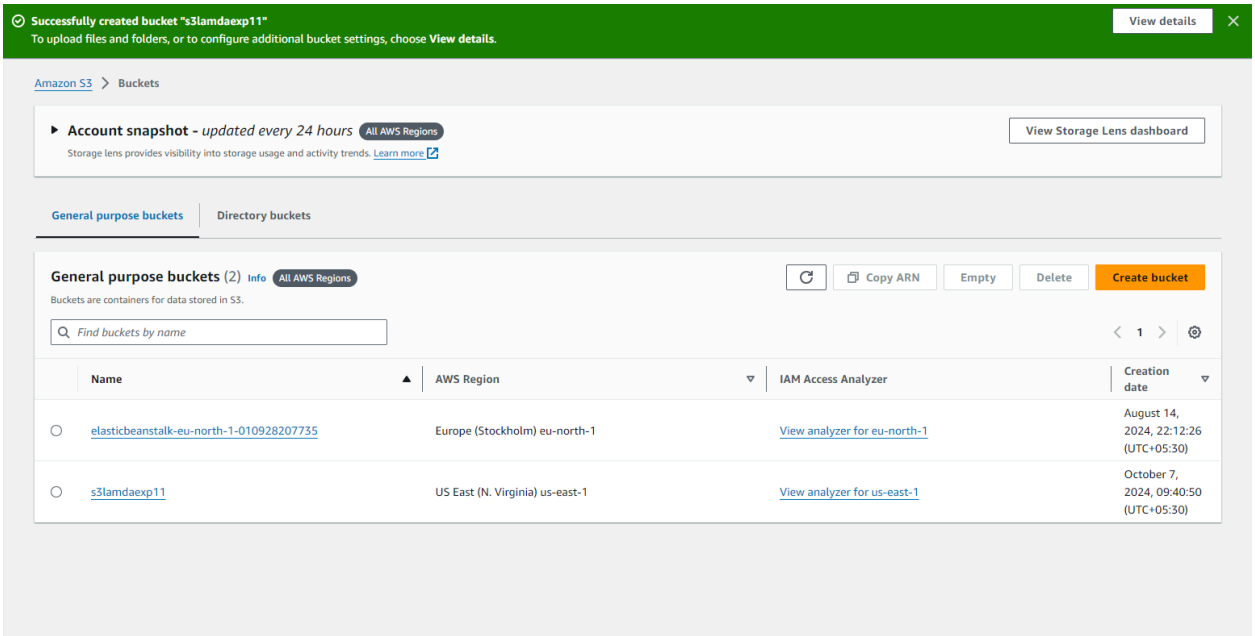
2) Keep the bucket as a general purpose bucket. Give a name to your bucket.



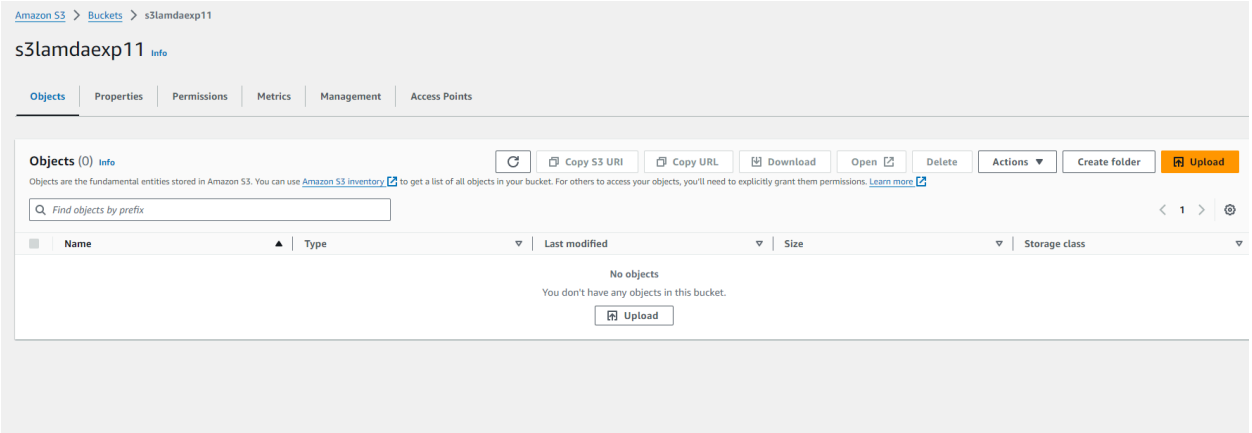
3) Uncheck block all public access

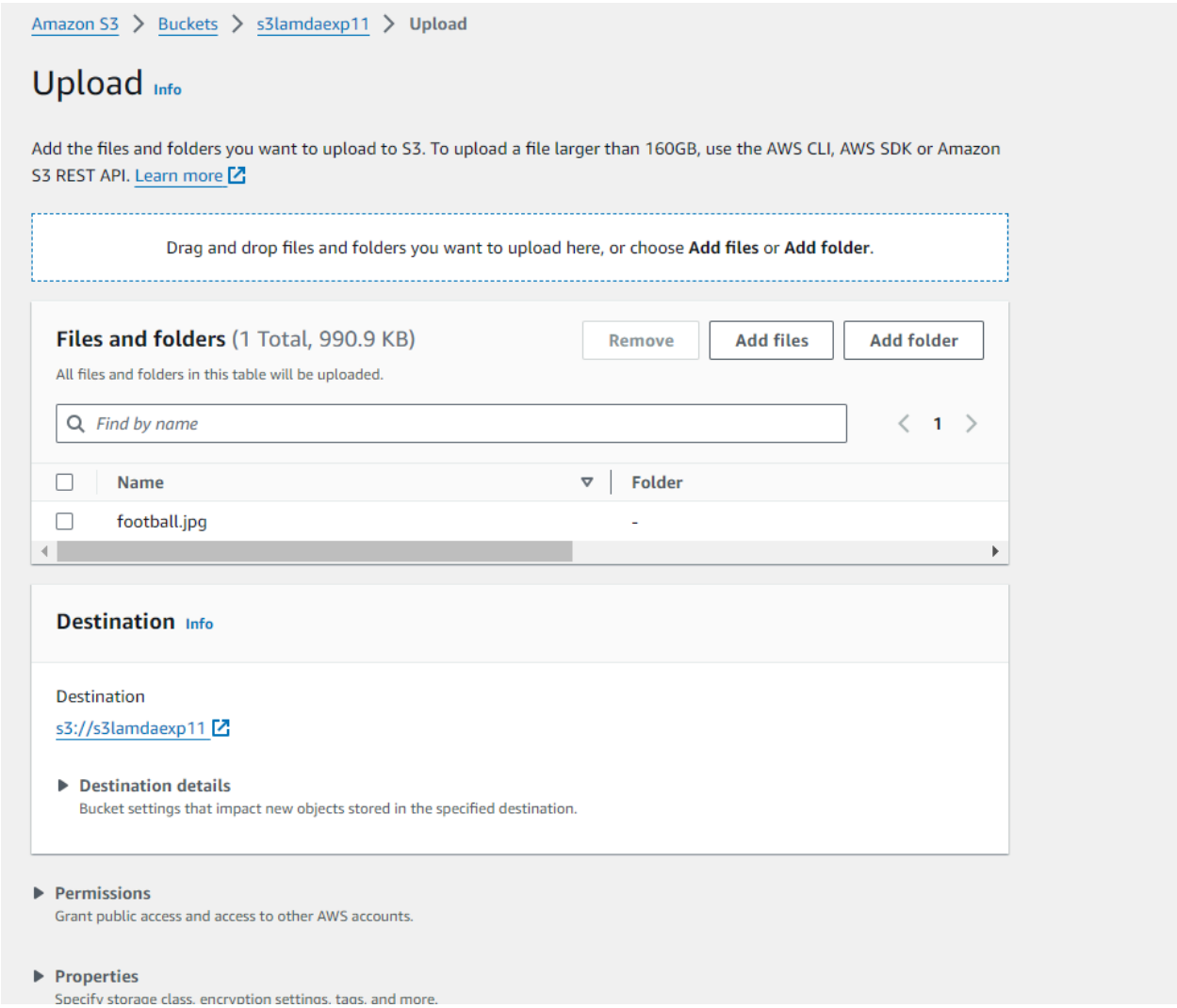


4) Keeping all other options same, click on create. This would create your bucket. Now click on the name of the bucket

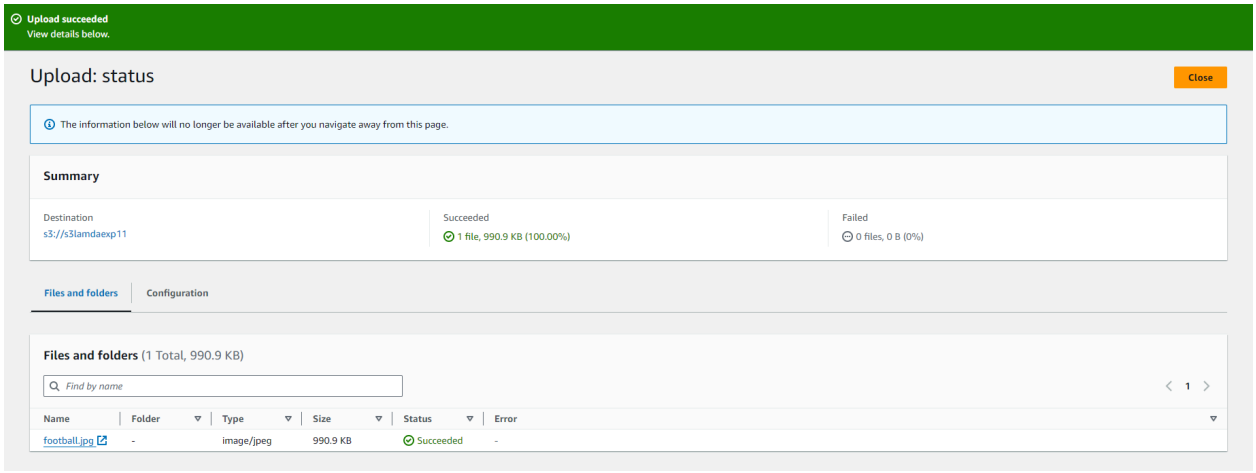


5) Here, click on upload, then add files. Select any image that you want to upload in the bucket and click on upload



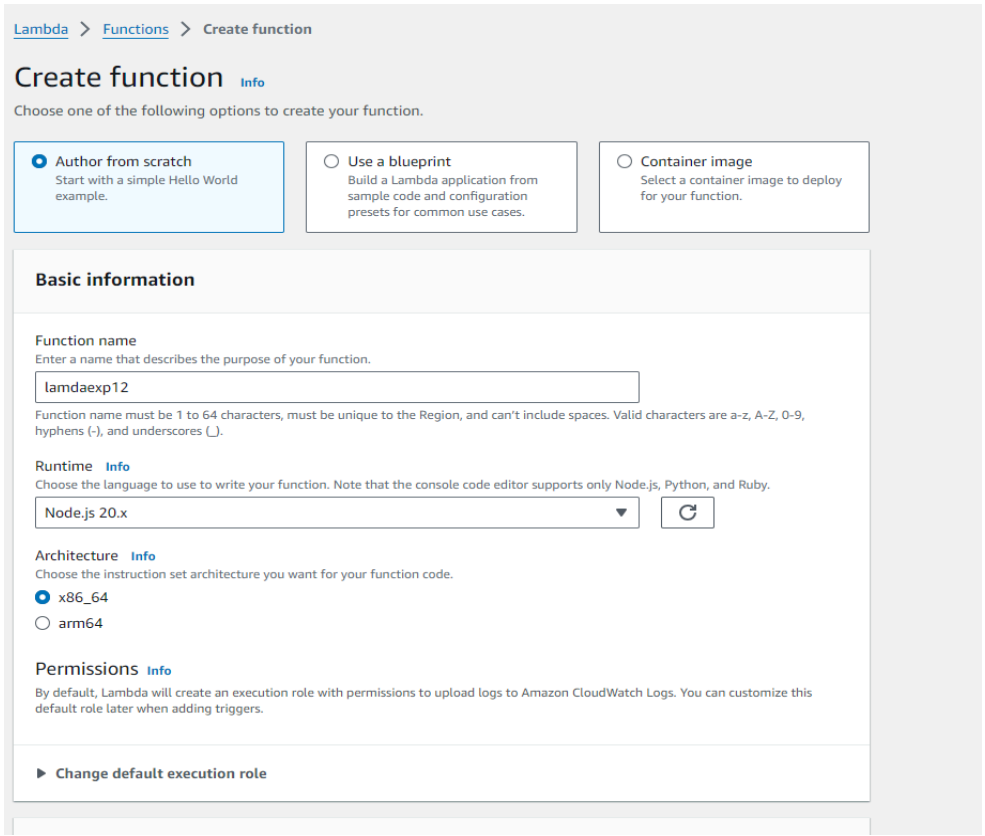


6) The image has been uploaded to the bucket

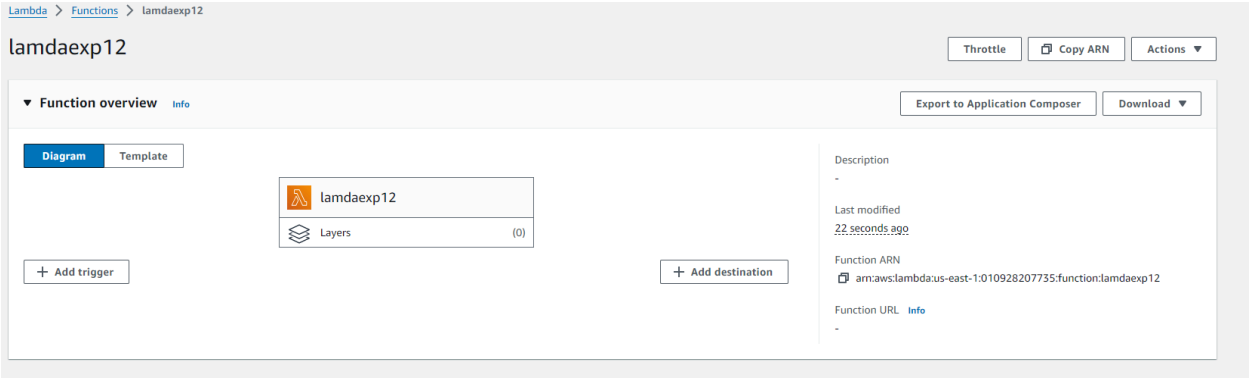


Step 2: Configure Lambda function

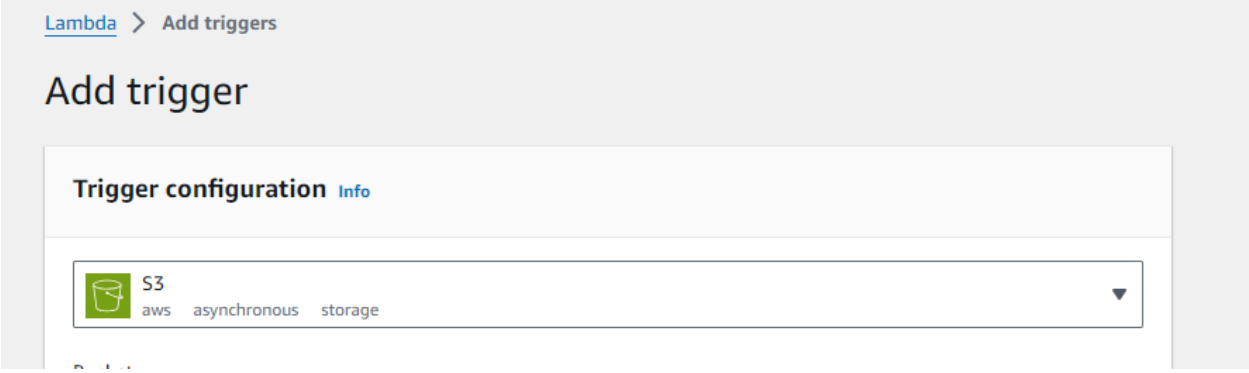
1) Go to the lambda function you had created berfor. (Services → Lambda → Click on name of function). Here, click on add trigger

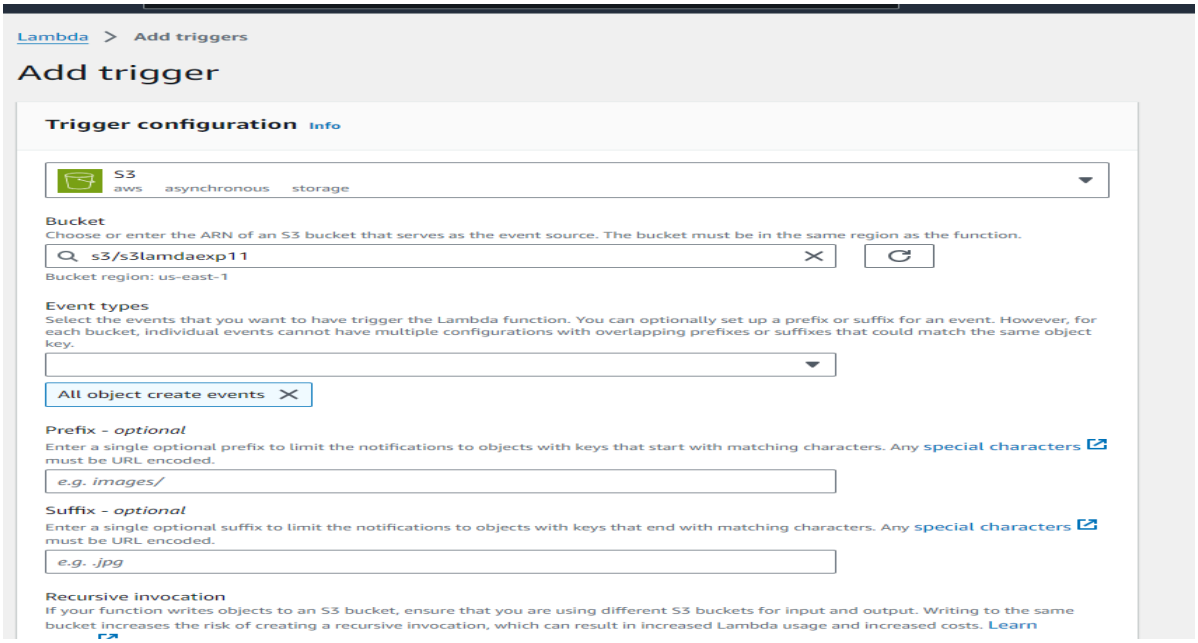


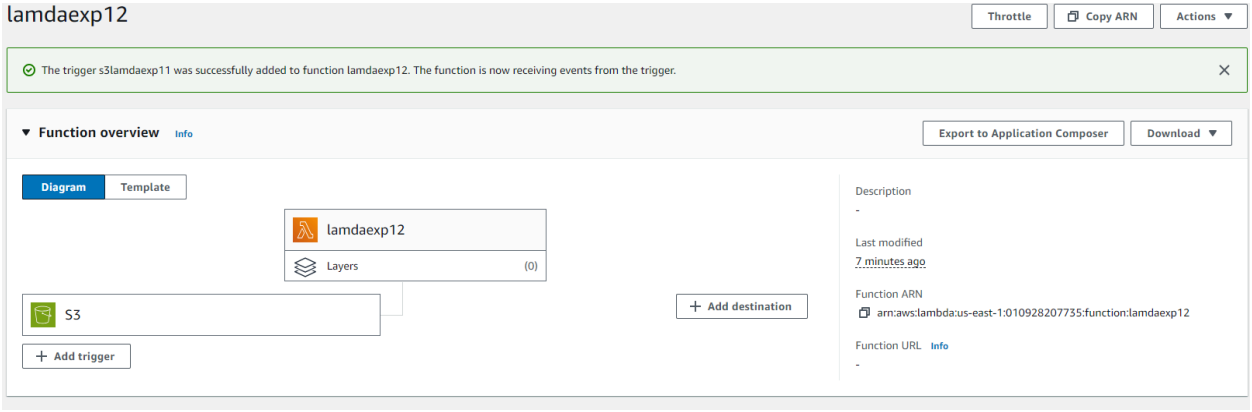
2) Under trigger configuration, search for S3 and select it.



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



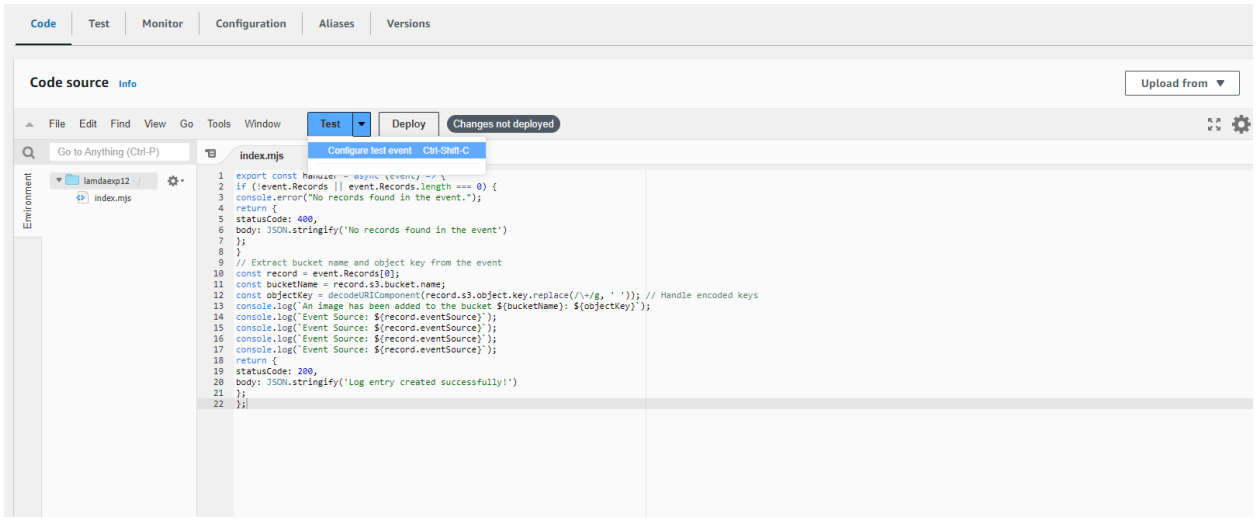




4) Scroll down to the code section of the function. Add the following javascript code to the code area by replacing the existing code

```
export const handler = async (event) => {
if (!event.Records || event.Records.length === 0) {
console.error("No records found in the event.");
return {
statusCode: 400,
body: JSON.stringify('No records found in the event')
};
}
// Extract bucket name and object key from the event
const record = event.Records[0];
const bucketName = record.s3.bucket.name;
const objectKey = decodeURIComponent(record.s3.object.key.replace(/\+/g, ' ')); // Handle encoded keys
console.log(`An image has been added to the bucket ${bucketName}: ${objectKey}`);
console.log(`Event Source: ${record.eventSource}`);
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 JSON structure represents an S3 event notification triggered when an object is uploaded to an S3 bucket. It contains details about the event, including the bucket name (example-bucket), the object key (test/key), and metadata like the object's size, the event source (aws:s3), and the event time.



Private

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

Shareable

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

Template - *optional*

hello-world

Event JSON Format JSON

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

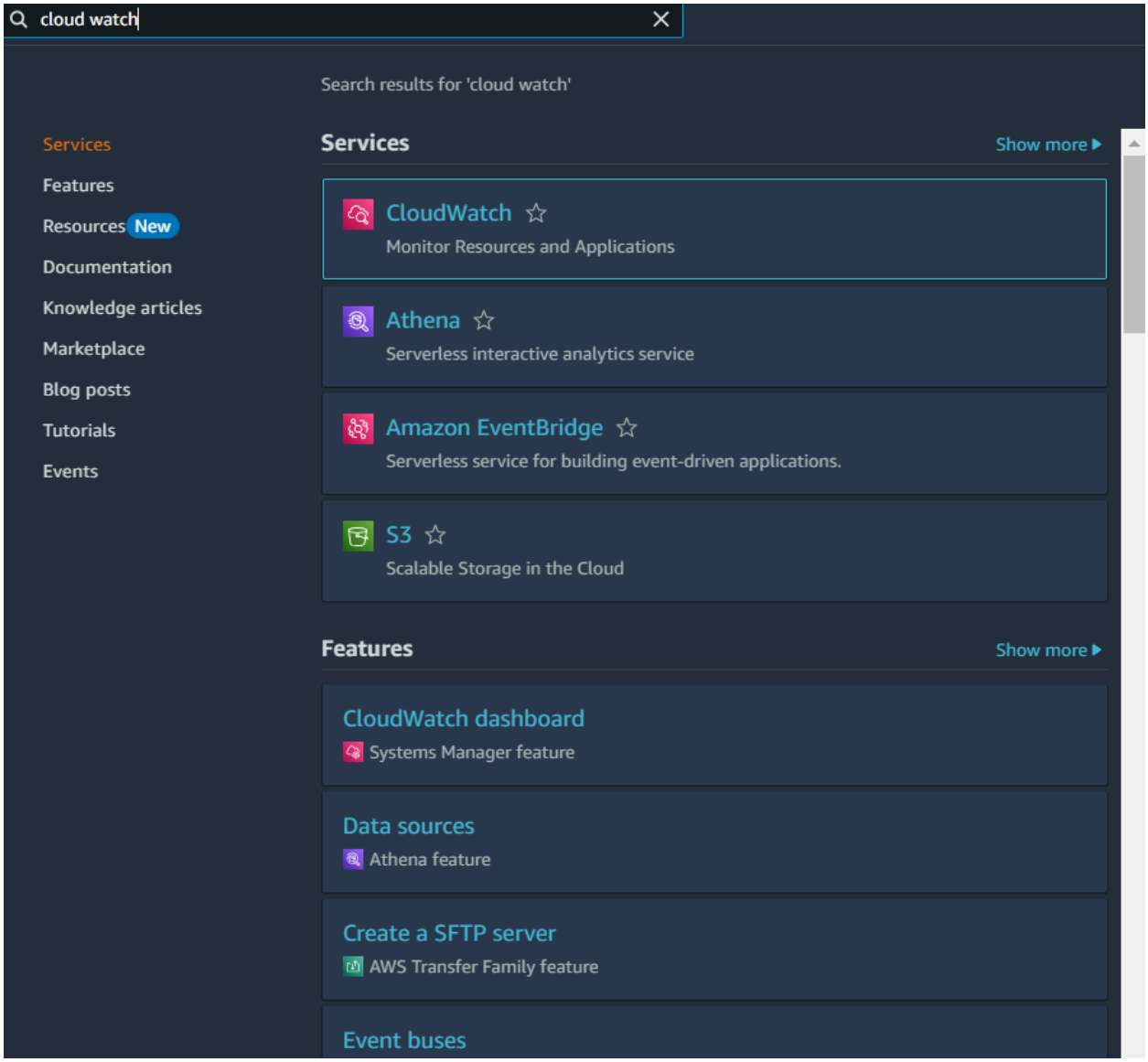
1:1 JSON Spaces: 2

Cancel Invoke Save

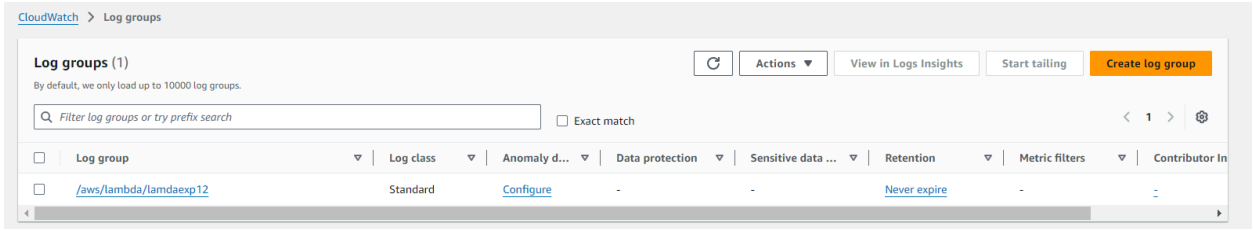
The test event **myevent1** was successfully saved.

Step 3: Check the logs

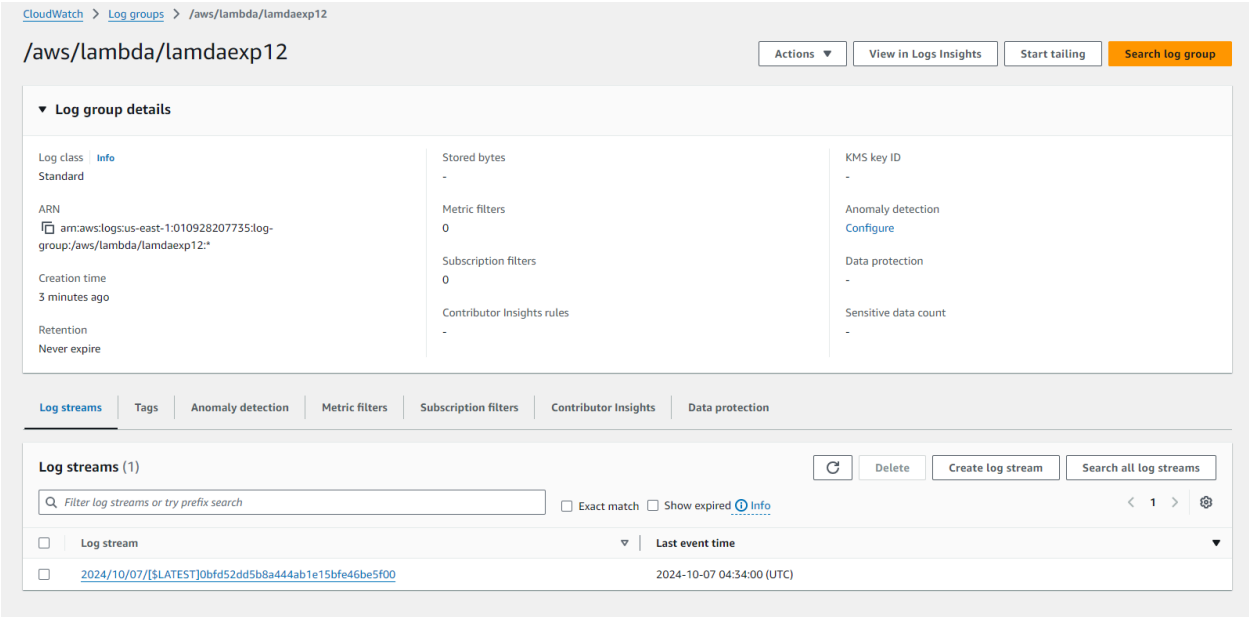
1) To check the logs explicitly, search foe CloudWatch on services and open it in a new tab



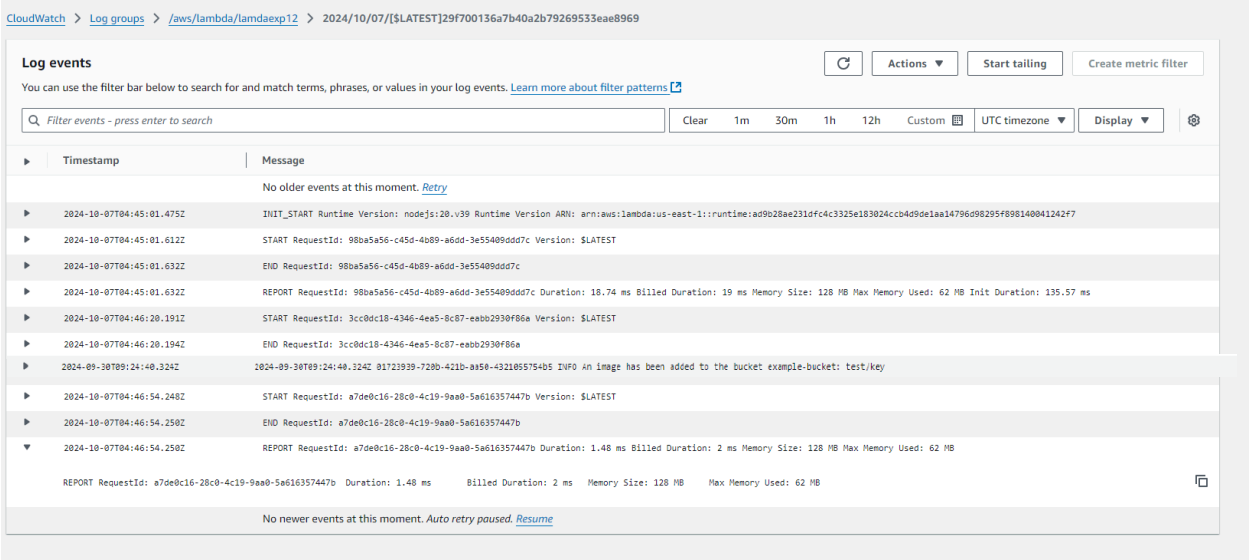
2) Here, Click on Logs → Log Groups. Select the log that has the lambda function name you just ran.



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



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



Conclusion: In this experiment, we successfully created a Lambda function that logs a message when an image is added to a specific S3 bucket. By configuring an S3 bucket trigger for the Lambda function, we demonstrated how AWS services can work together to automate tasks. The function logged important details about the event, including the bucket name and object key. After deploying the function and testing with a sample event, we verified the logs in CloudWatch, confirming that the Lambda function correctly detected and logged the addition of an image to the bucket. This experiment highlighted the seamless integration between AWS Lambda and S3 for event-driven processes.