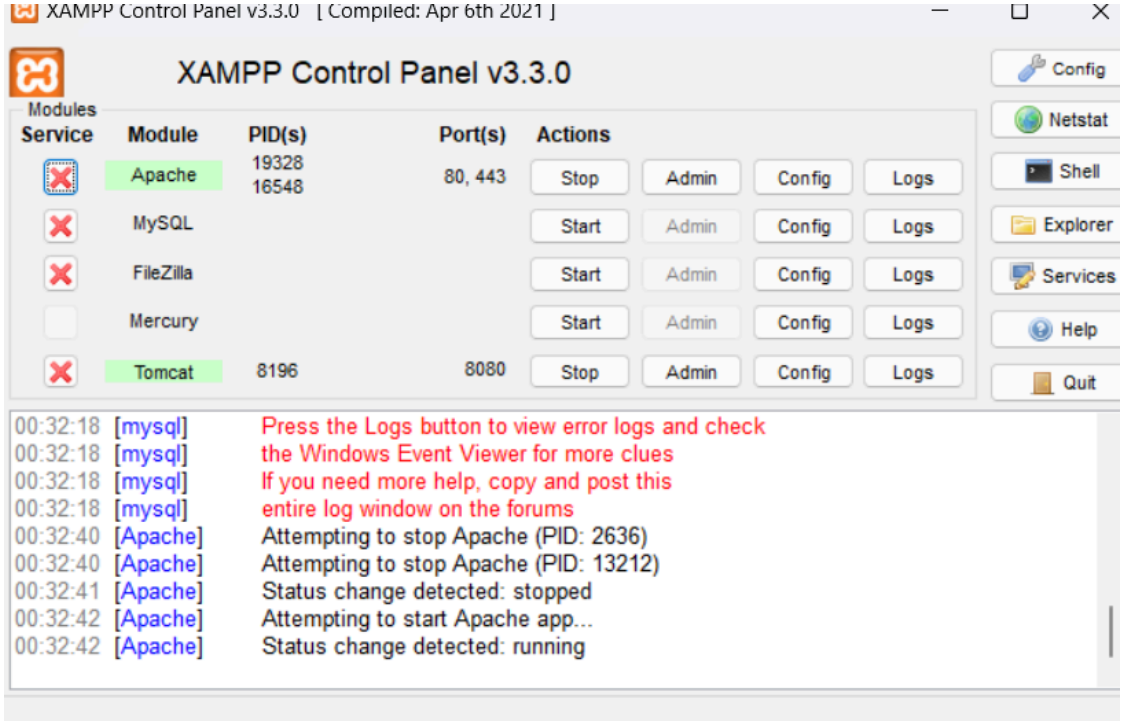
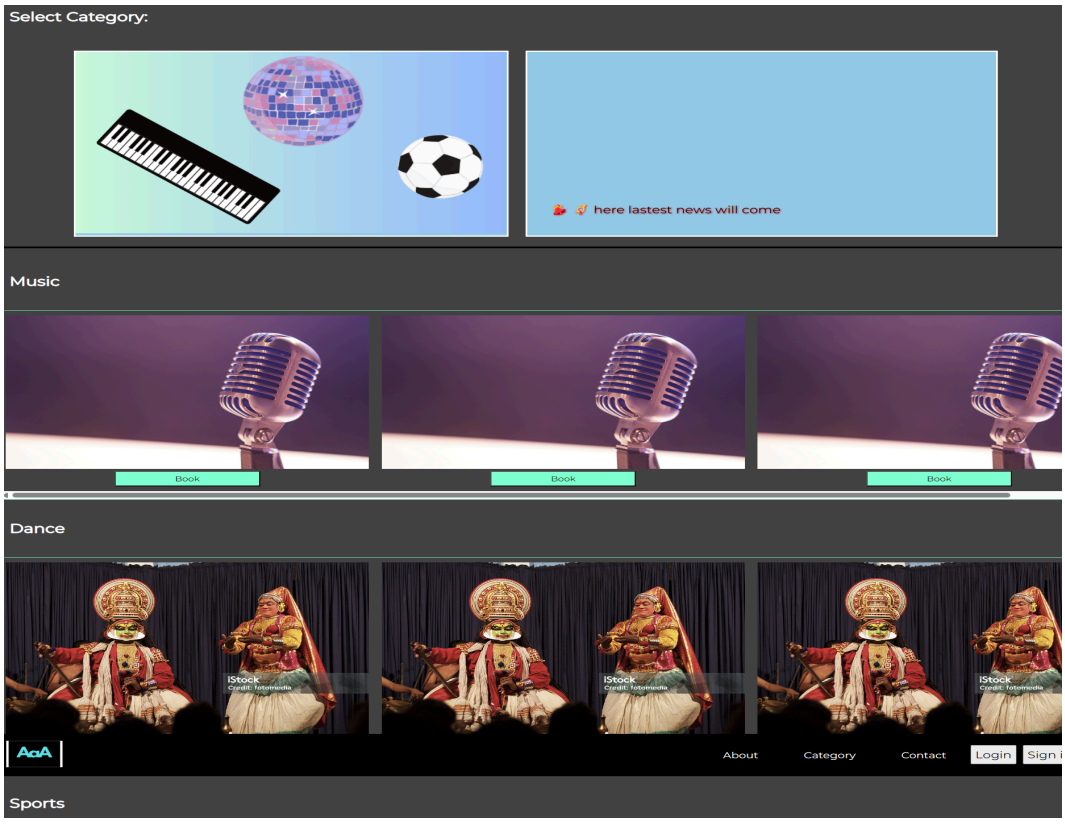
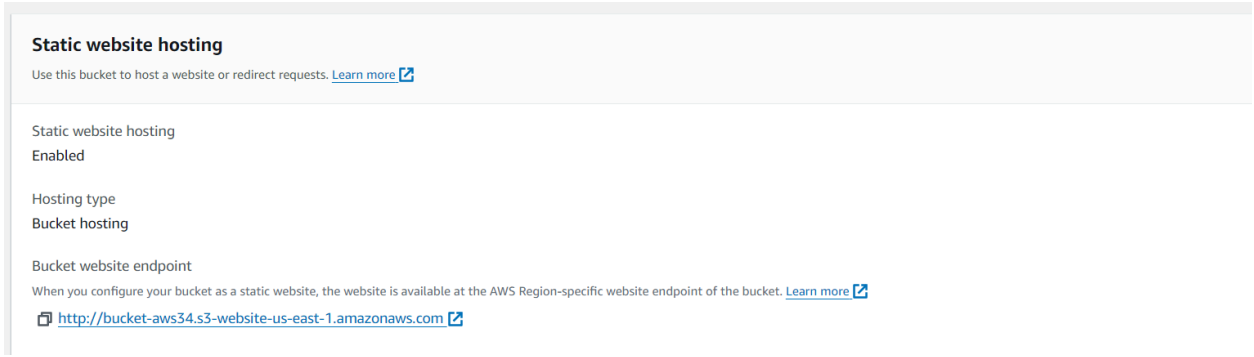
Experiment 1a

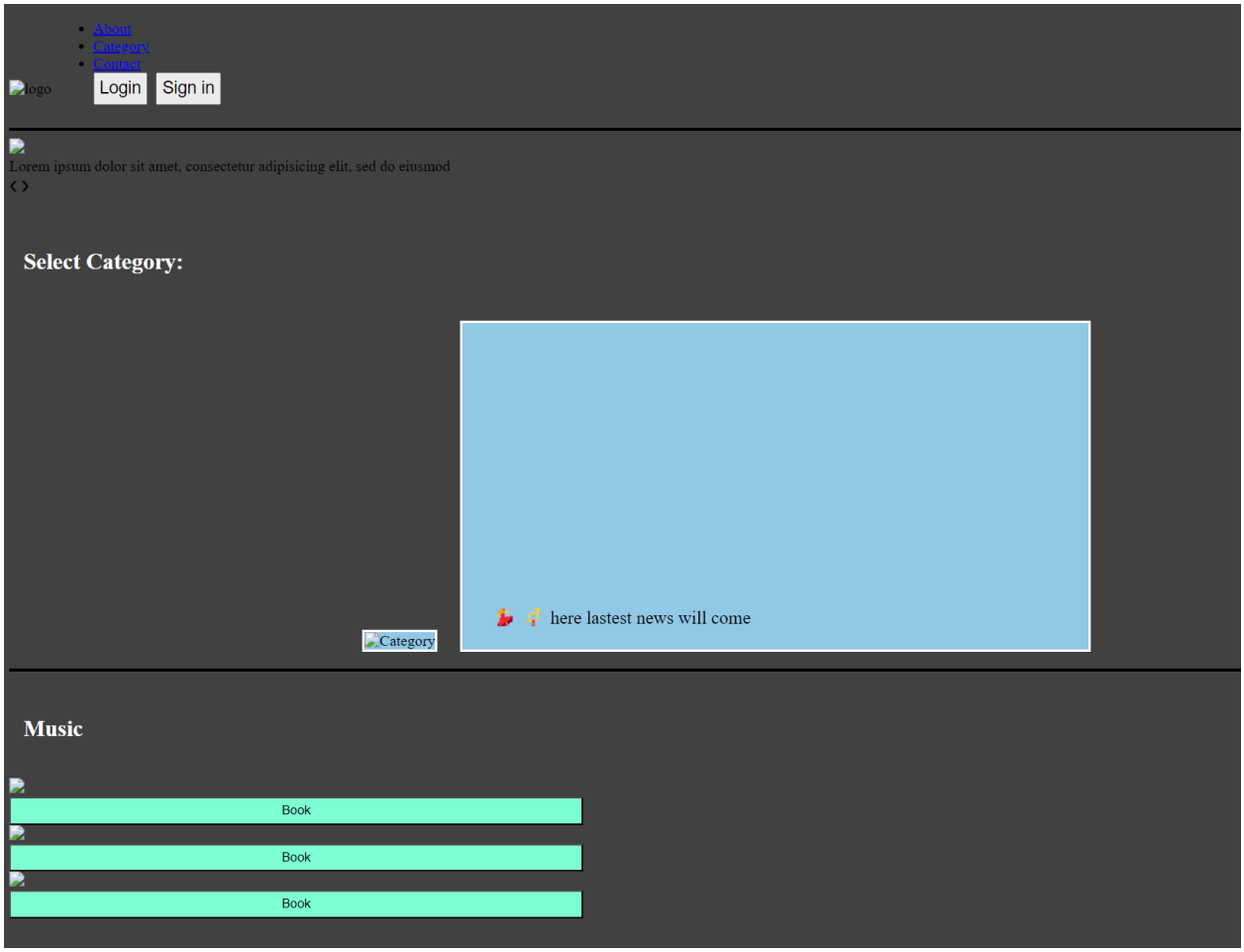
1)Static Hosting using XAMPP



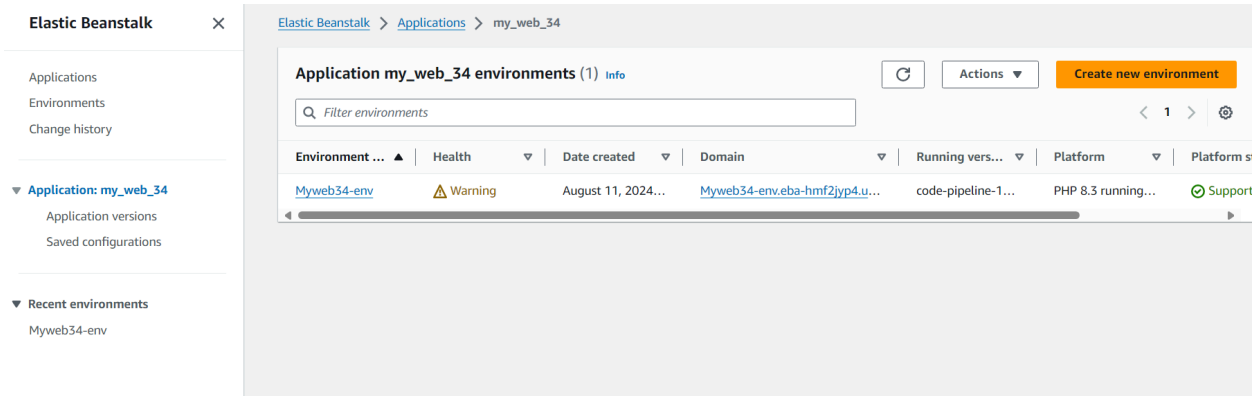


2)Static Hosting using Amazon S3

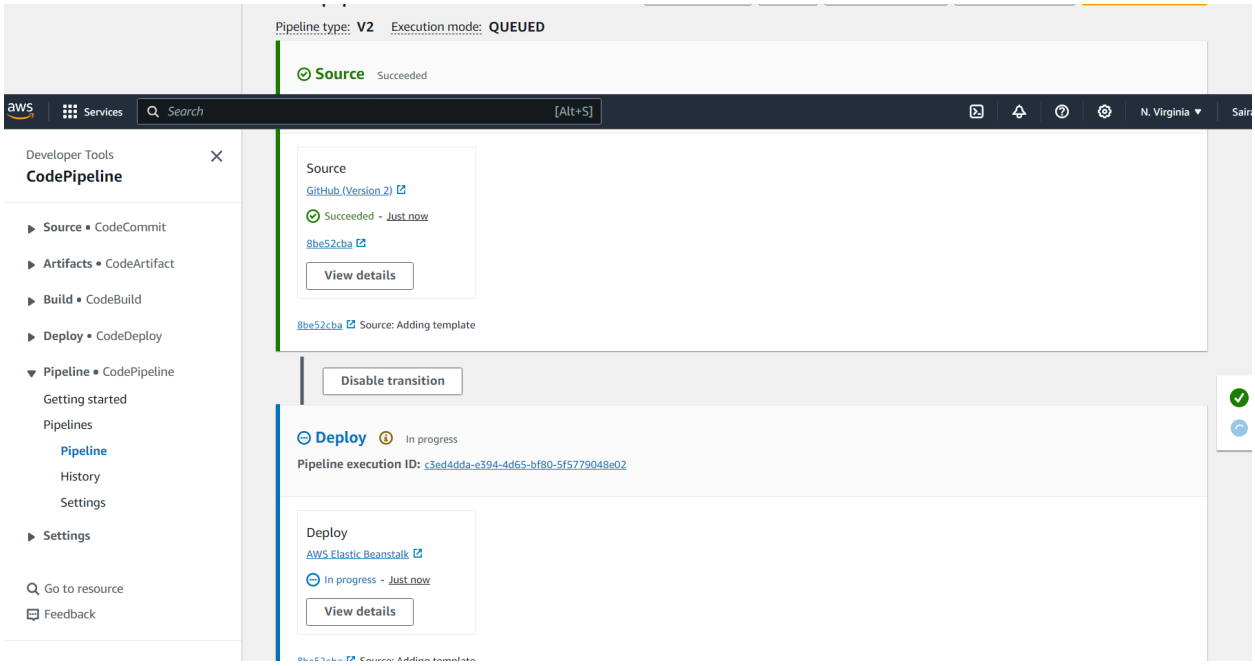




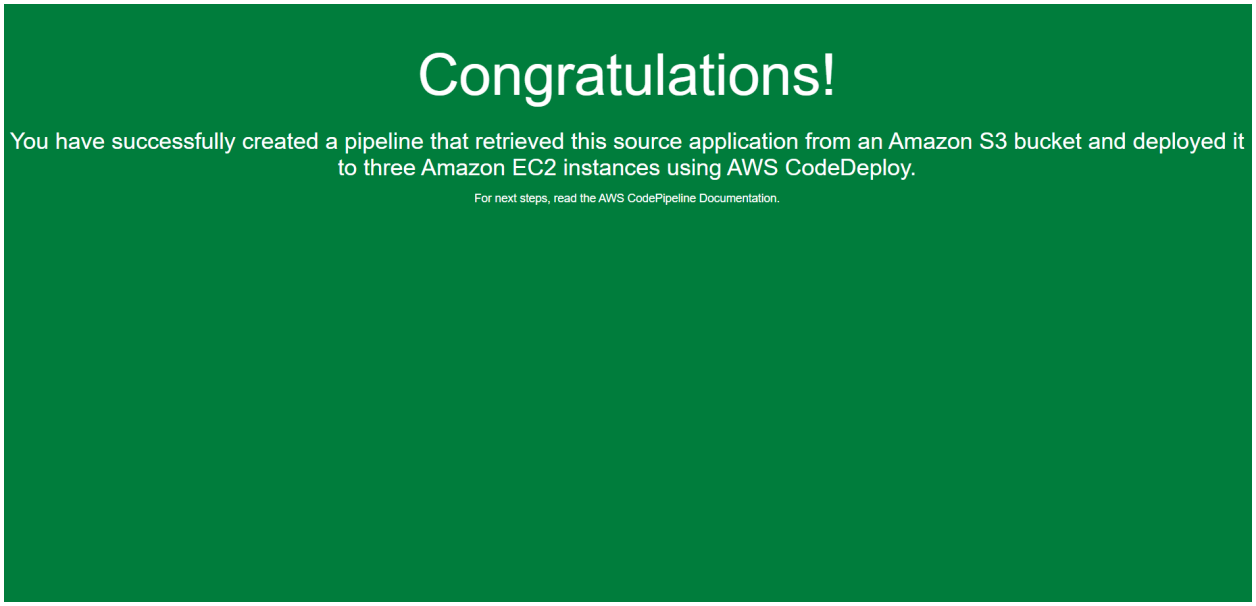
1)Elastic Beanstalk Environment



2)CodePipeline

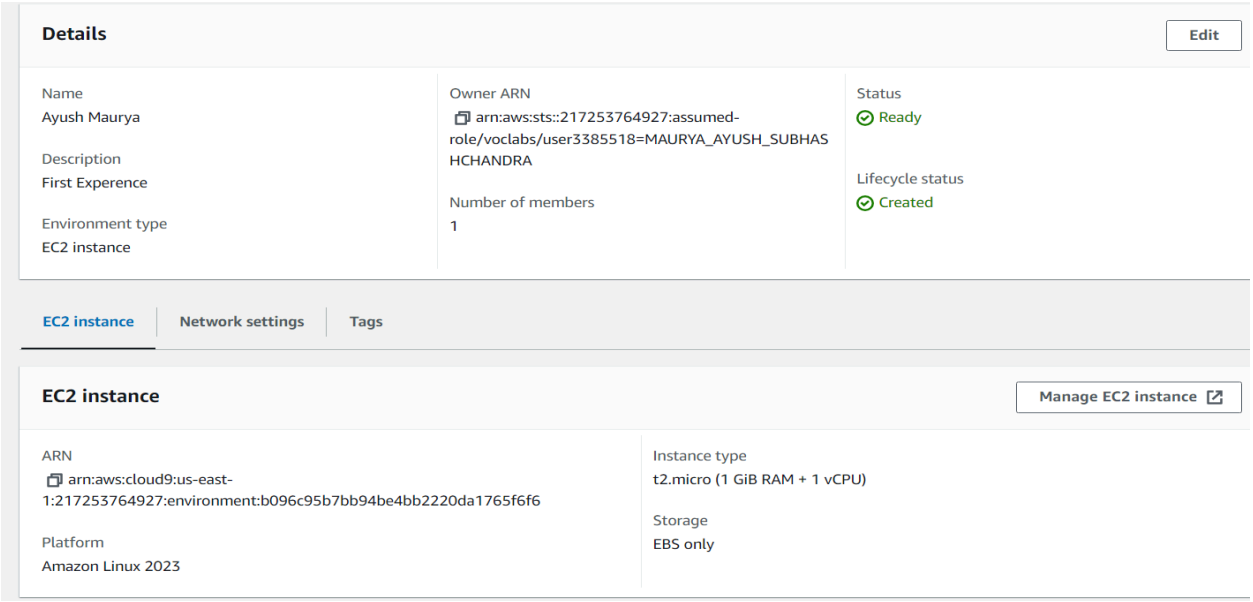


Website Before Change

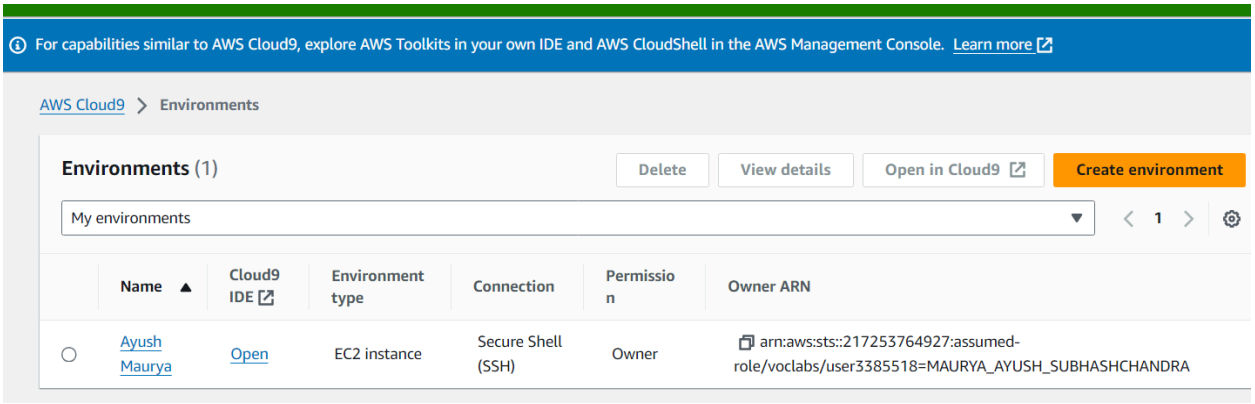


Website After Change

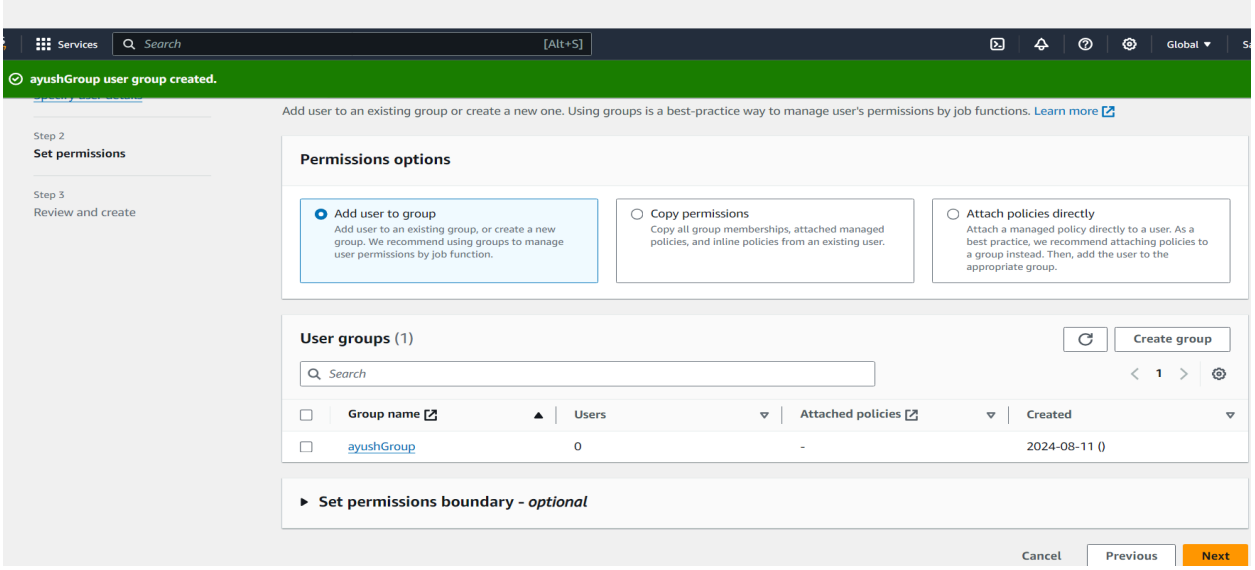
1)EC2 Instance in AWS



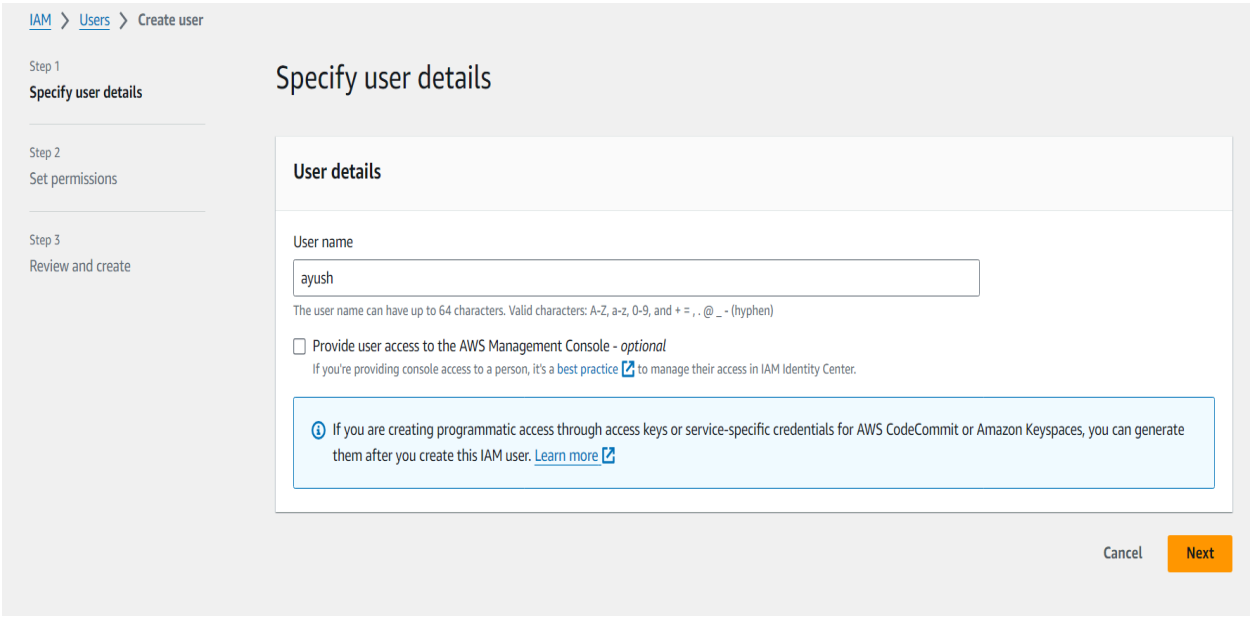
2)IAM Services



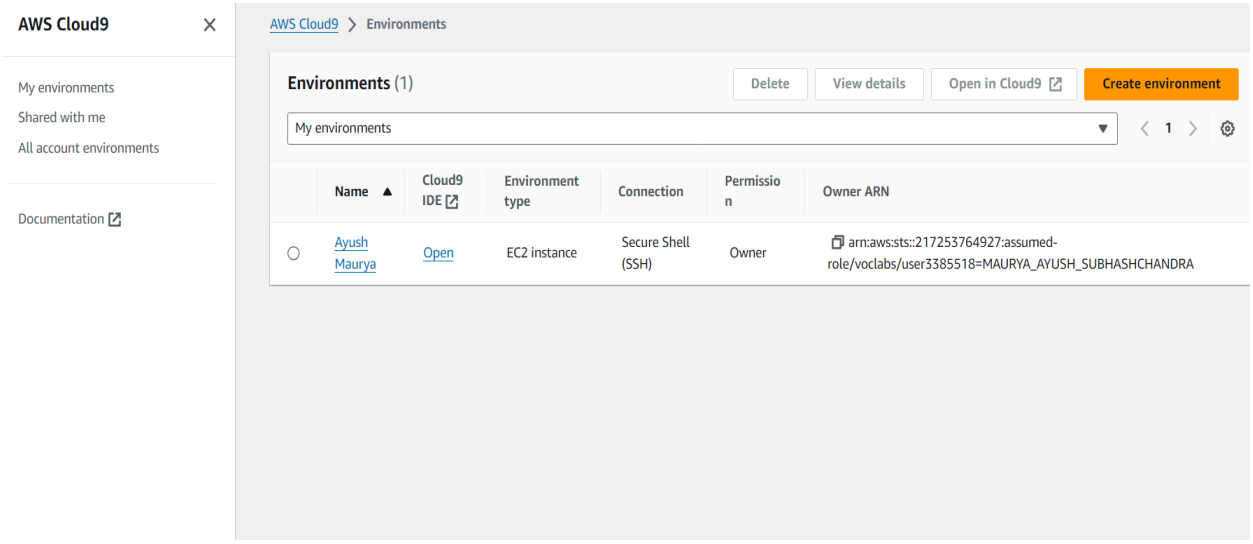
User Group

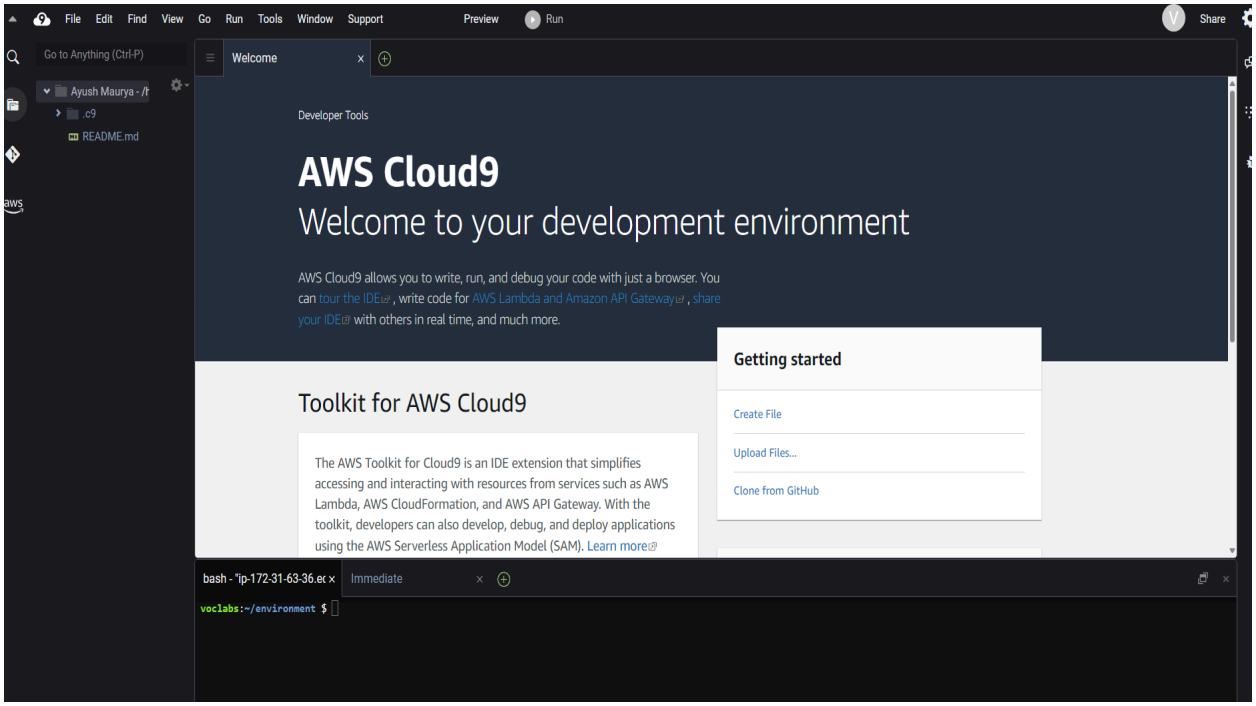


User



3)Cloud 9 and Cloud 9 IDE

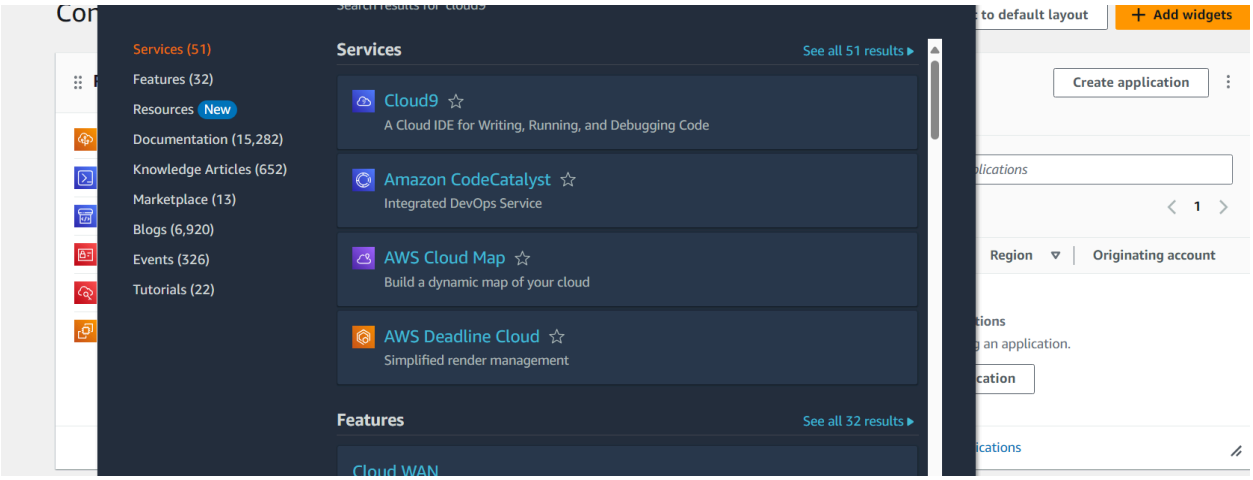



File Edit Find View Go Run Tools Window Support
Preview
Run
Share
Go to Anything (Ctrl-P)
Ayush Maurya - /t
.c9
README.md
Welcome
Developer Tools
AWS Cloud9
Welcome to your development environment
AWS Cloud9 allows you to write, run, and debug your code with just a browser. You can tour the IDE, write code for AWS Lambda and Amazon API Gateway, share your IDE with others in real time, and much more.
Getting started
Create File
Upload Files...
Clone from GitHub
Toolkit for AWS Cloud9
The AWS Toolkit for Cloud9 is an IDE extension that simplifies accessing and interacting with resources from services such as AWS Lambda, AWS CloudFormation, and AWS API Gateway. With the toolkit, developers can also develop, debug, and deploy applications using the AWS Serverless Application Model (SAM). Learn more
bash - "ip-172-31-63-36.ec
Immediate
voclabs:~/environment $

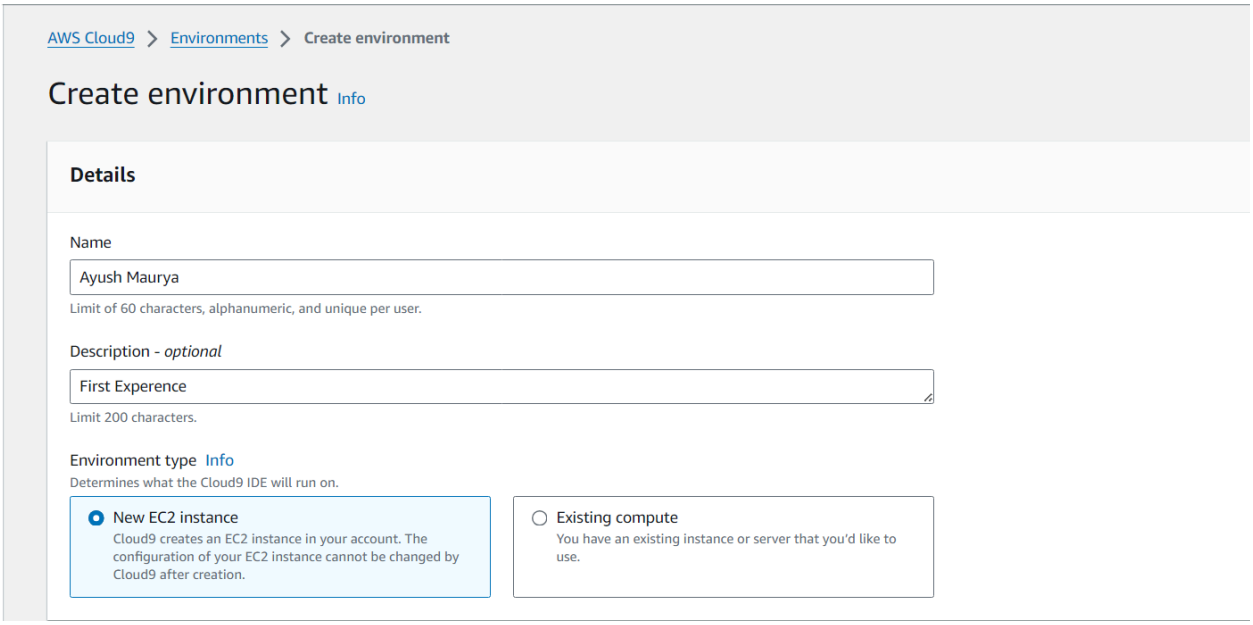
Experiment 1b

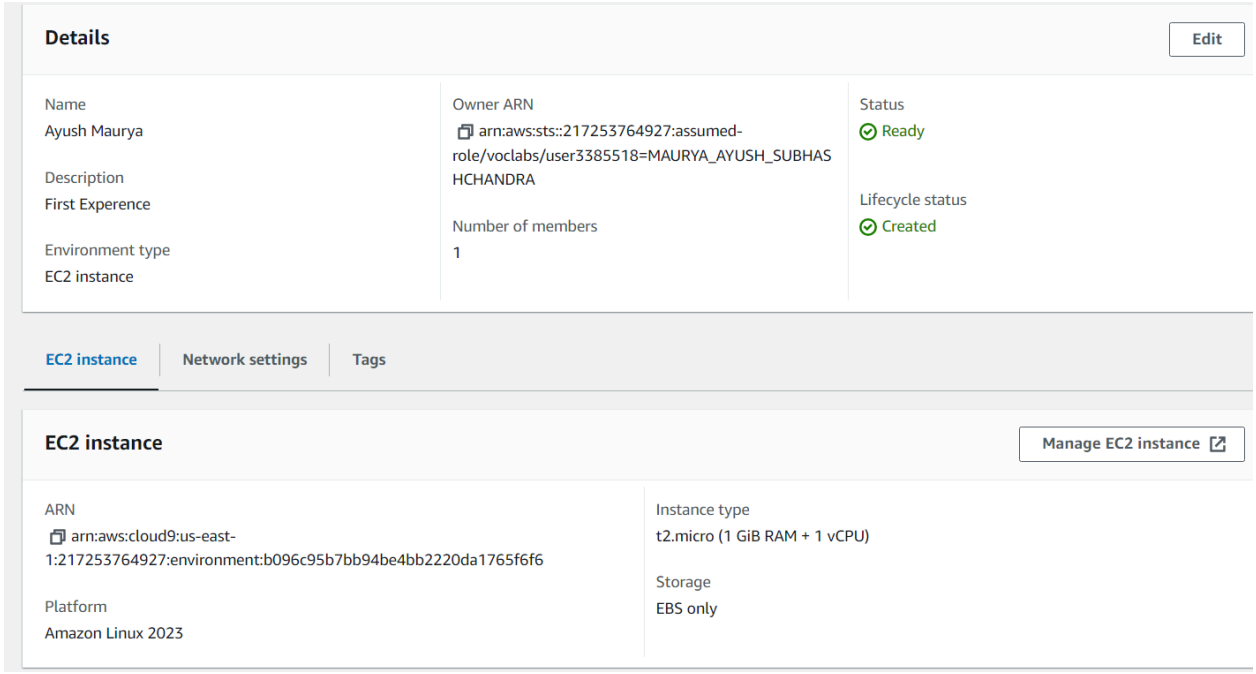
Step 1: Set up Cloud9 environment.

1) Search Cloud9 in the services tab and open it

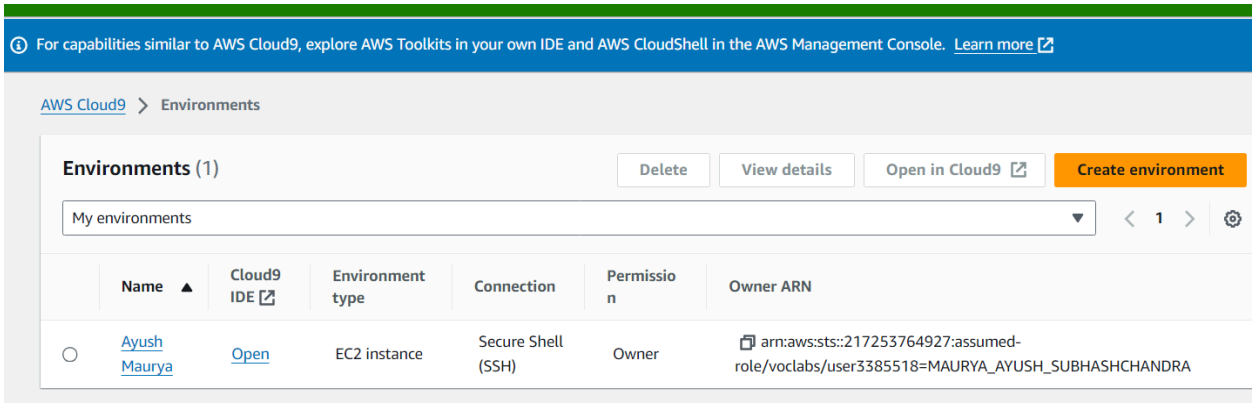


2) Click on Create Environment.
→Give a name to your Cloud9 Environment. You can add a description if needed.
→Select the option new EC2 instance if you do not have one ready for the environment. Give the specifications of that EC2 instance ahead
→On the AWS Academy account, if we select AWS System Manager(SSM) in Network settings, it gives and error as the account does not have permissions to use the setting. So we select Secure Shell (SSH). After that click on Create.



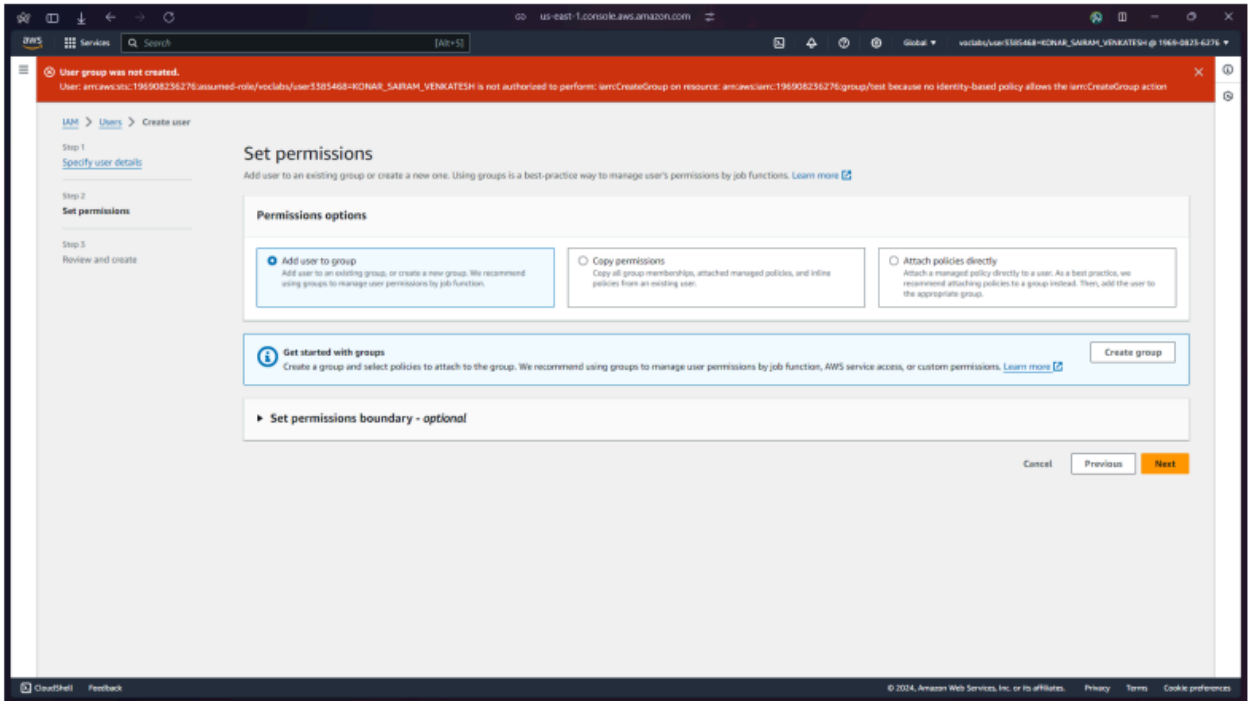


6) The environment is being created.

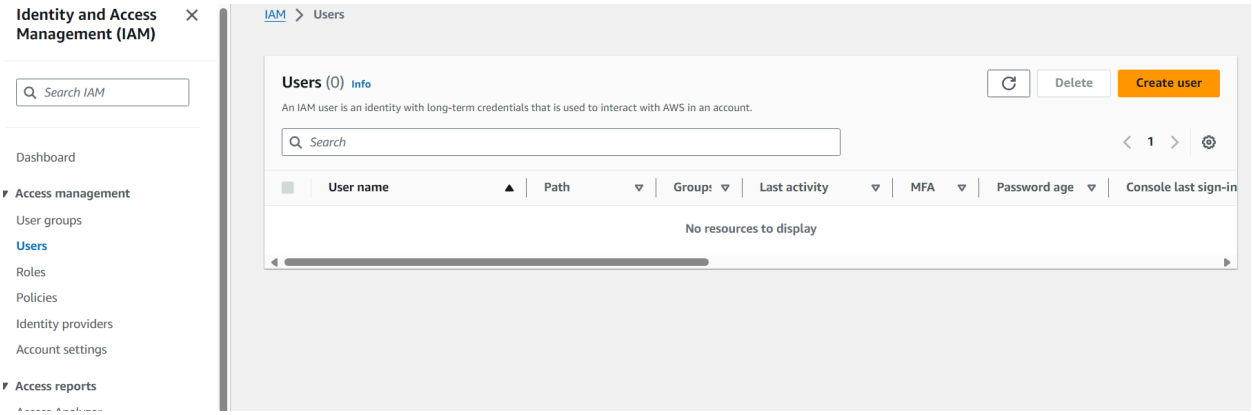


Step 2: Creating IAM user.

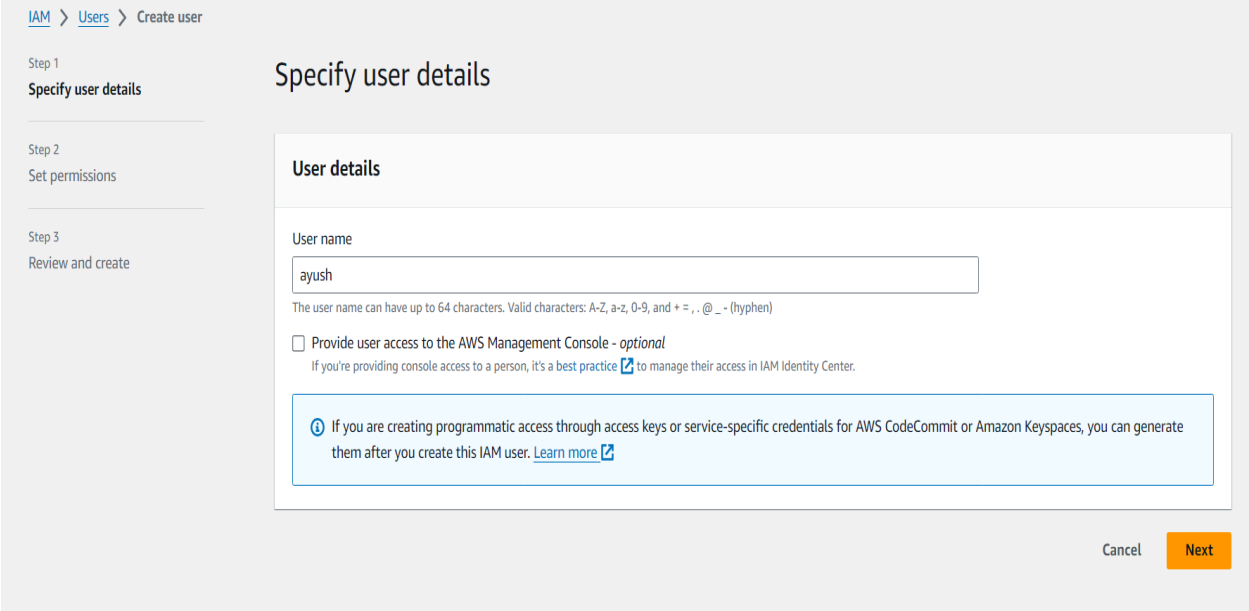
When we go to add user to a group, the AWS Academy account throws an error as we do not have the permissions to create a group. So we have to use our personal AWS account for this part. ‘



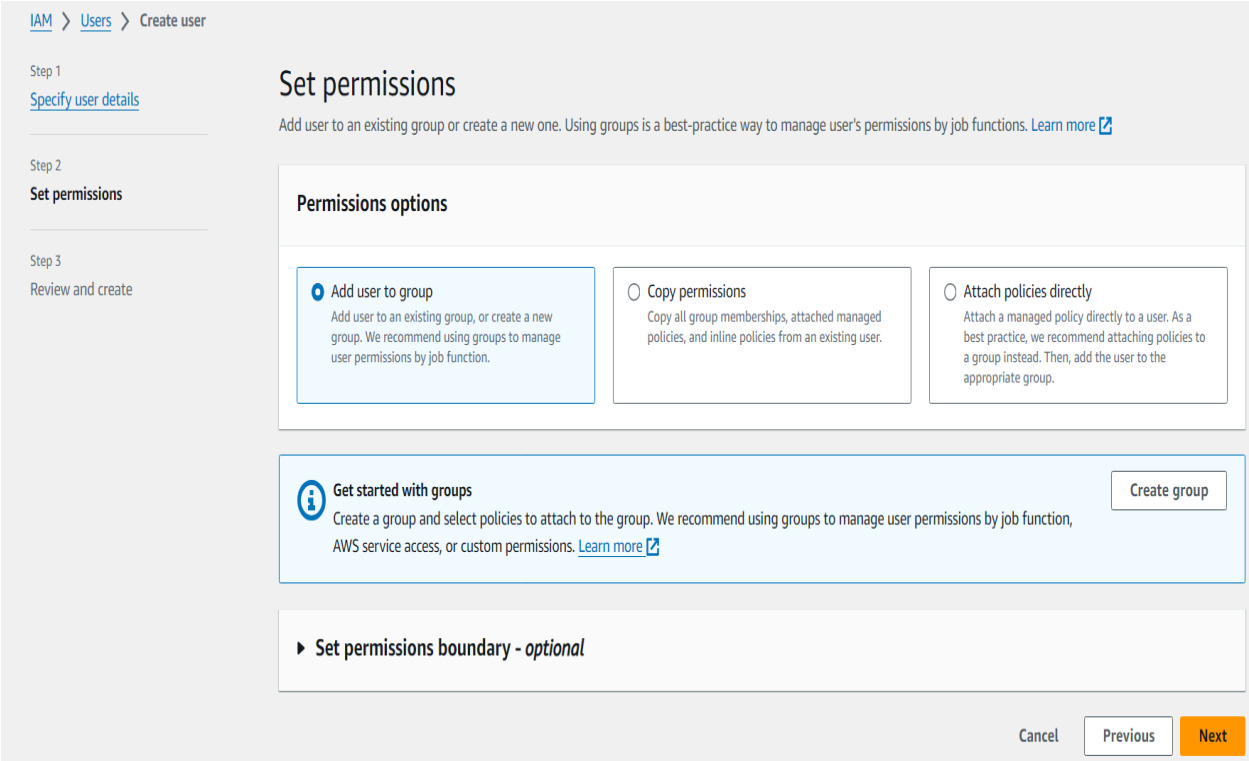
1) Search IAM on the services search bar and open it. Click on Create User.



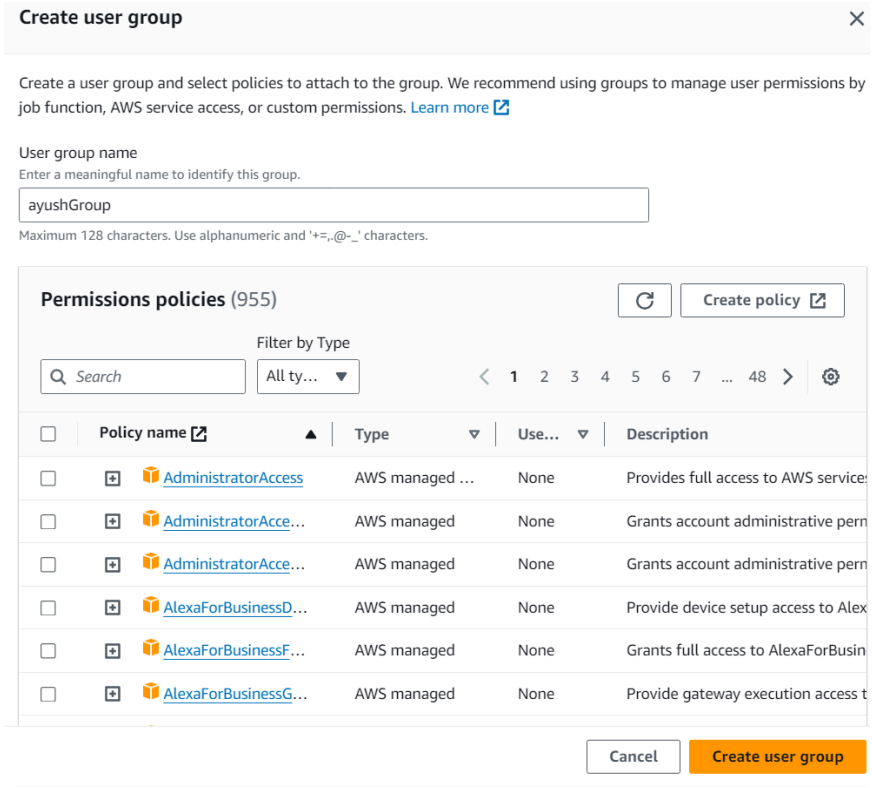
2) Give a username to your user and click Next.



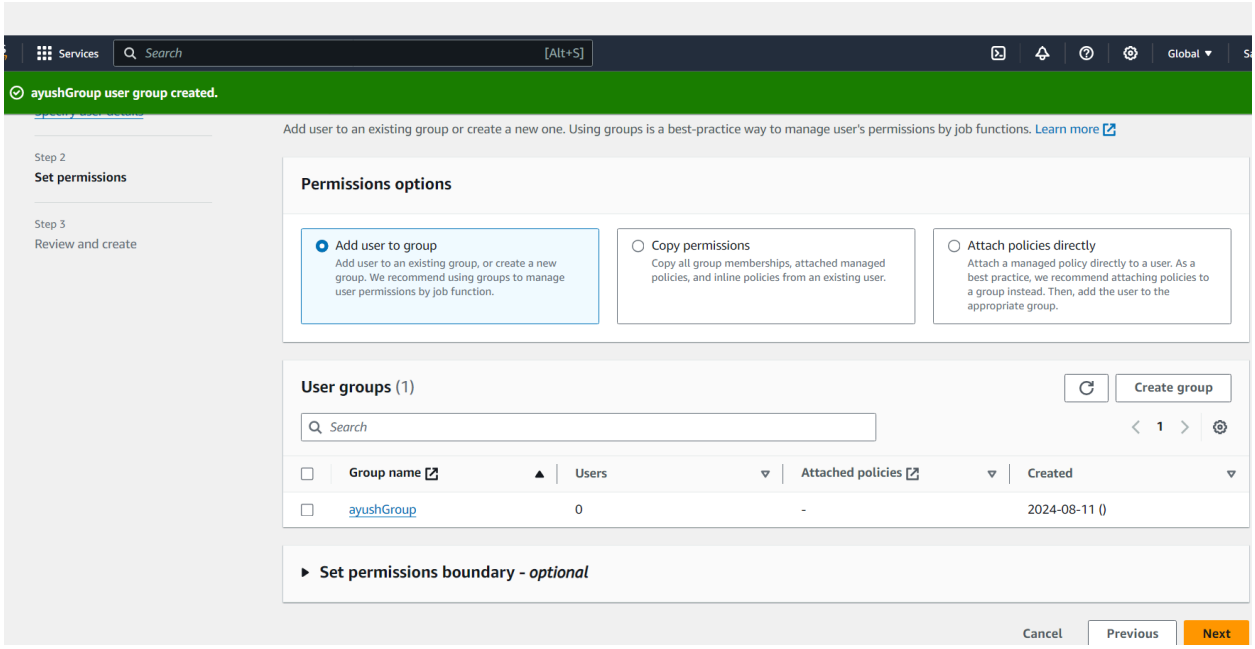
3) Select add User to Group. If there are no user groups on your accounts, you will have to create one. Click on Create Group.



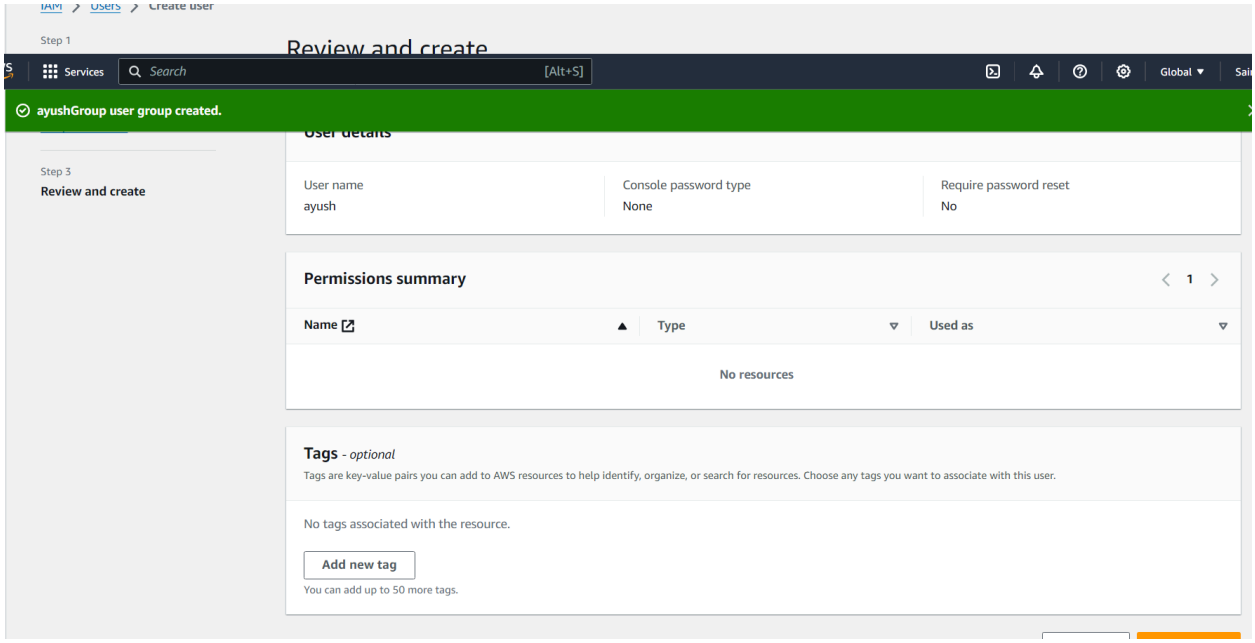
4) Give a name to your user group. Then click on Create User Group.



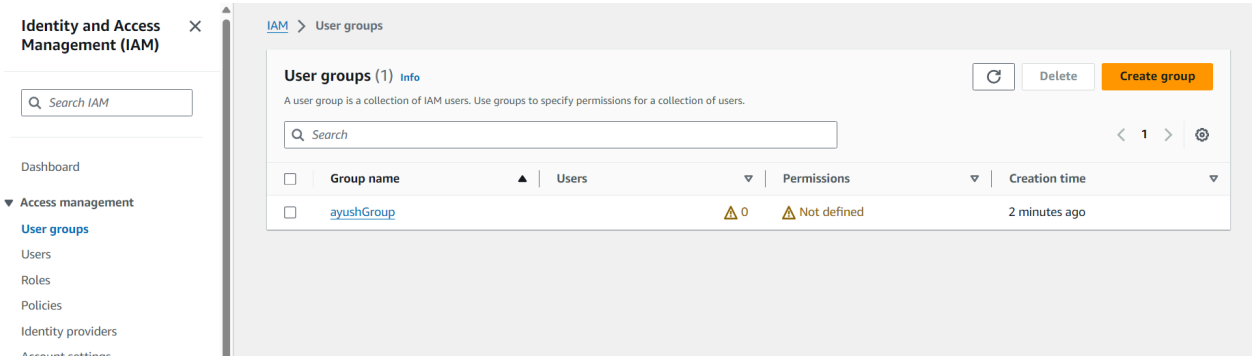
5) The group is created and shown under the groups area, select the group by clicking on the checkbox. Then click Next.



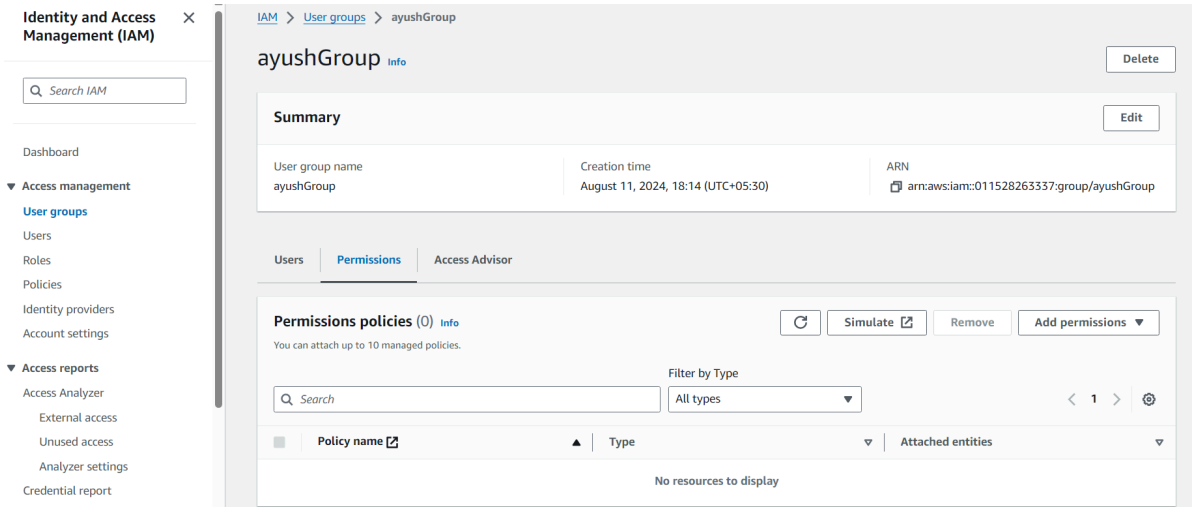
6) Review all the Information, then click on Create user.



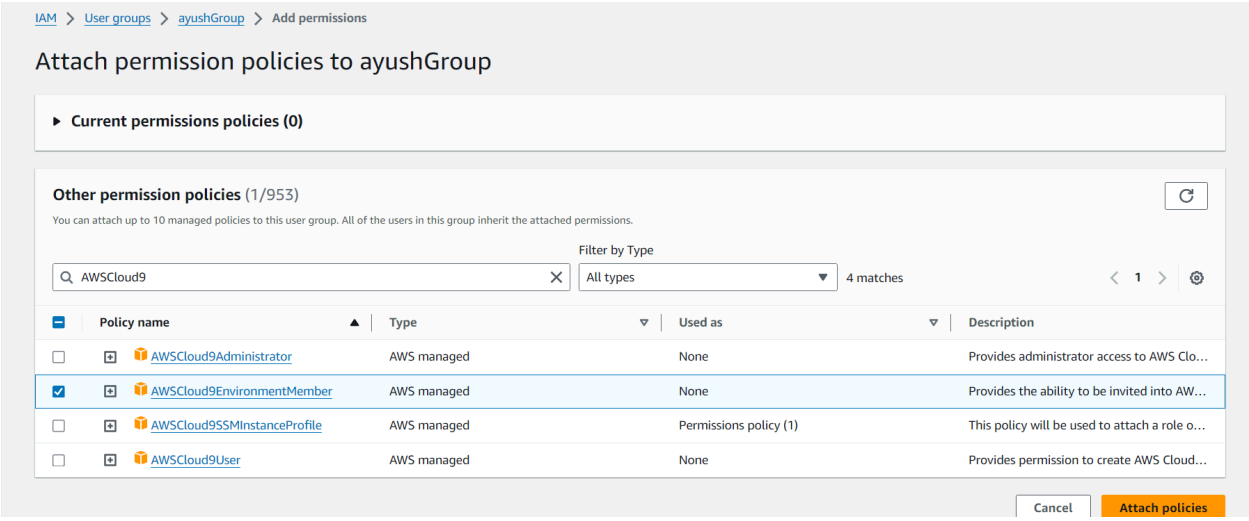
7) Open User Groups tab from the left side option. Click on the name of your group.



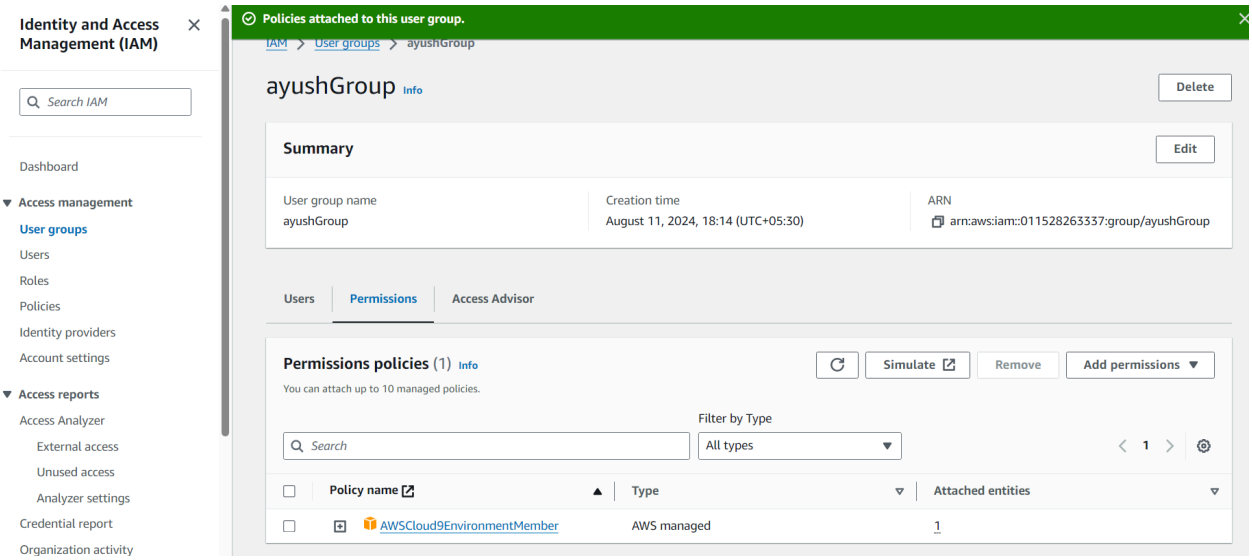
8) Go to permissions and click on Add permissions. Click on Attach Policies.



9) Search for AWSCloud9EnvironmentMember, select it and click on Attach policies

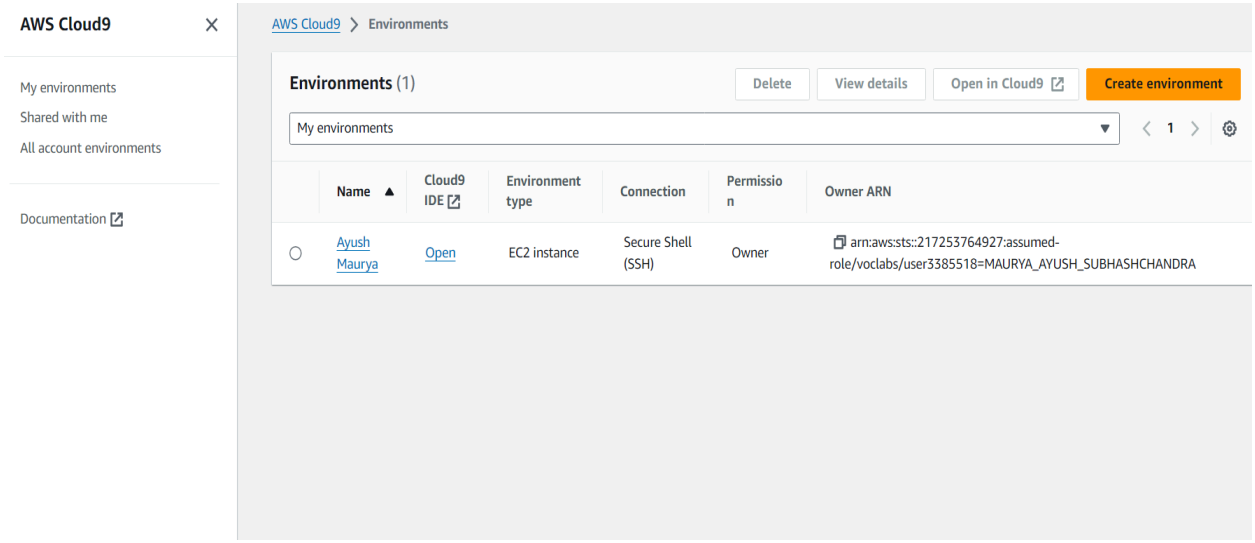


10) The policies have been attached

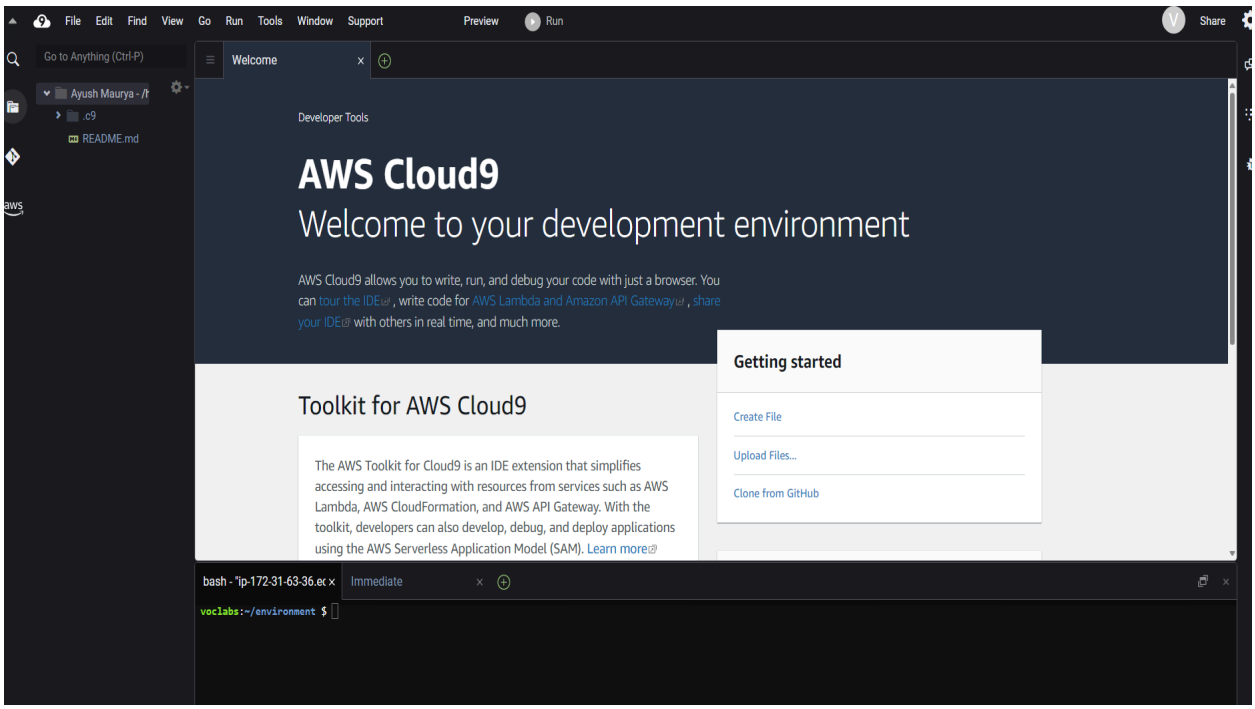


Step 3: Working on Cloud9 IDE

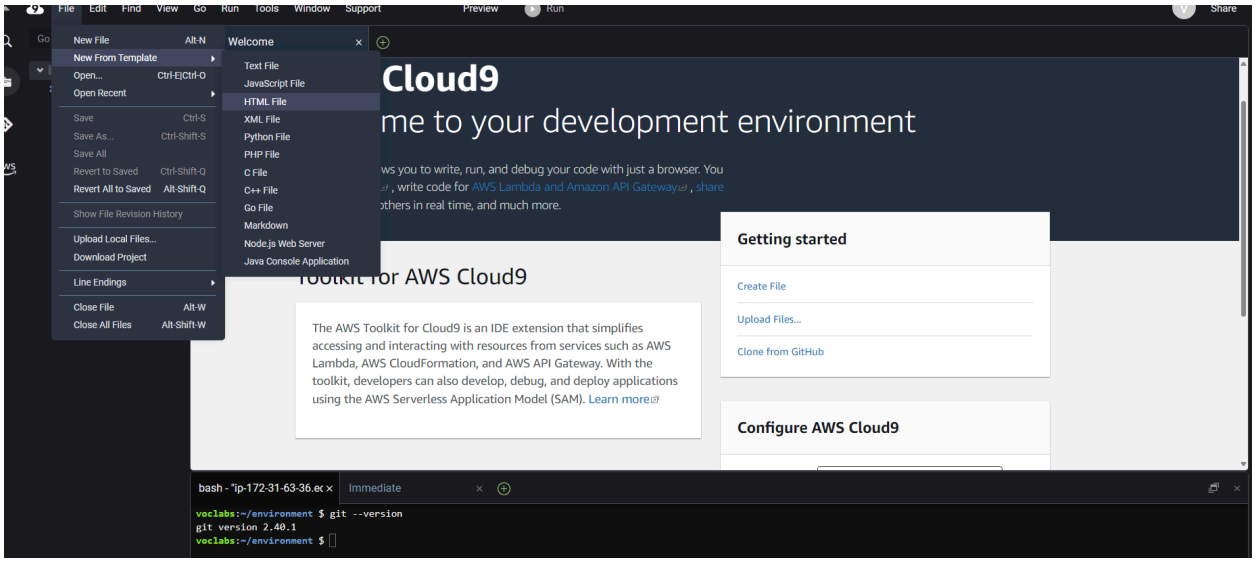
1) Go to Cloud9 services. Click on Open under Cloud9 IDE.



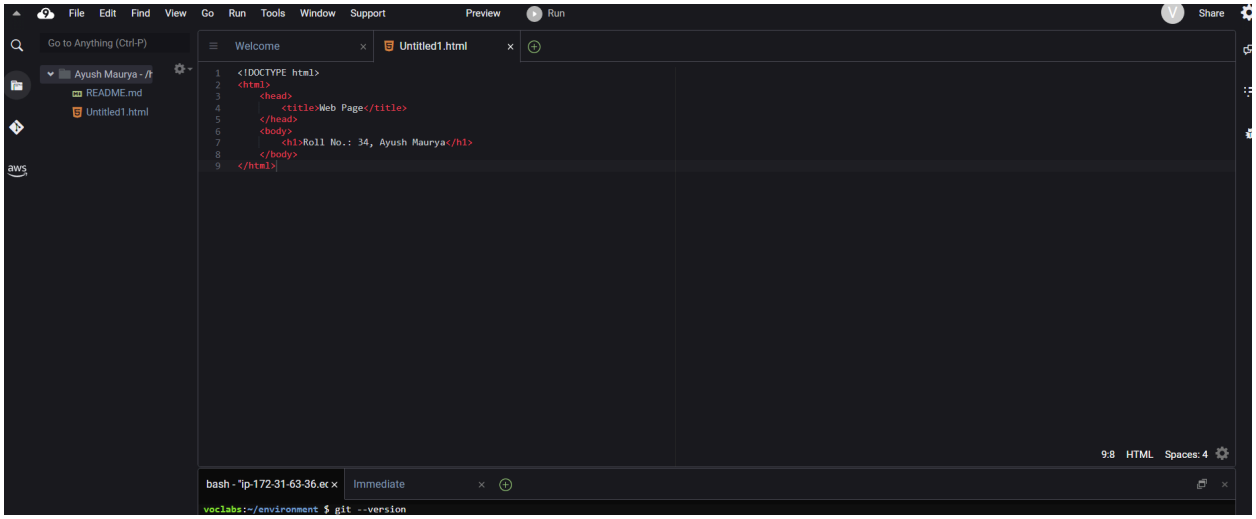
2) This is the Cloud9 IDE interface. The major part of the screen is the coding IDE. There is a command console just below it. For example, the command git --version is run.



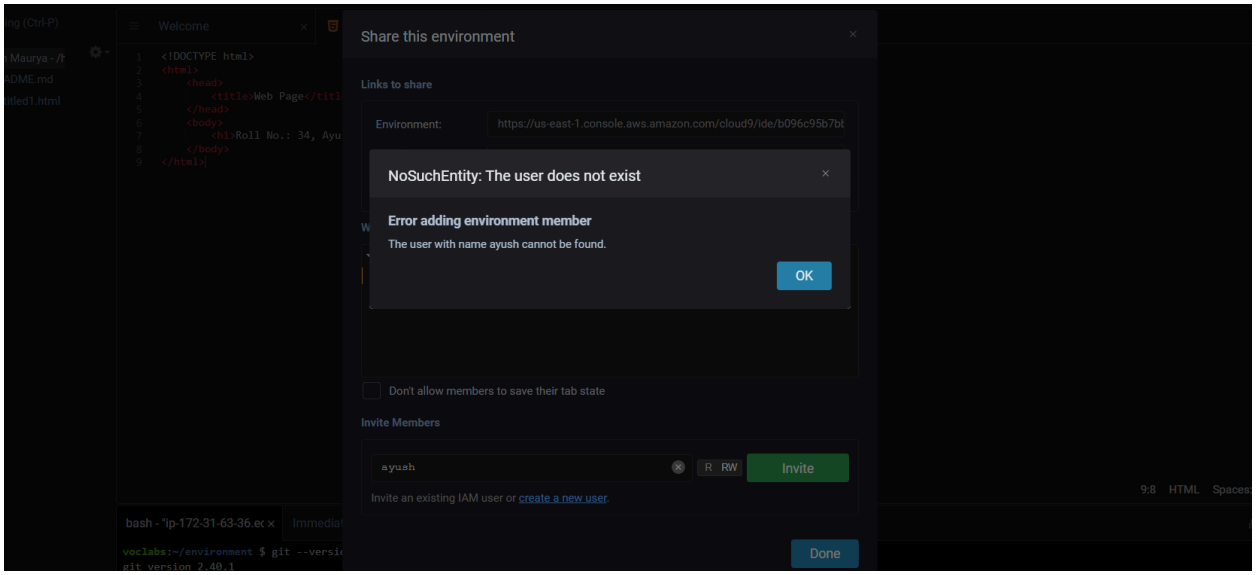
3) To add a file, click on file. For this experiment, we are to add an HTML file. So go to File → New From Template → HTML file. This gives a basic HTML template on the coding IDE.



4) Make a basic website on the HTML template and save it.



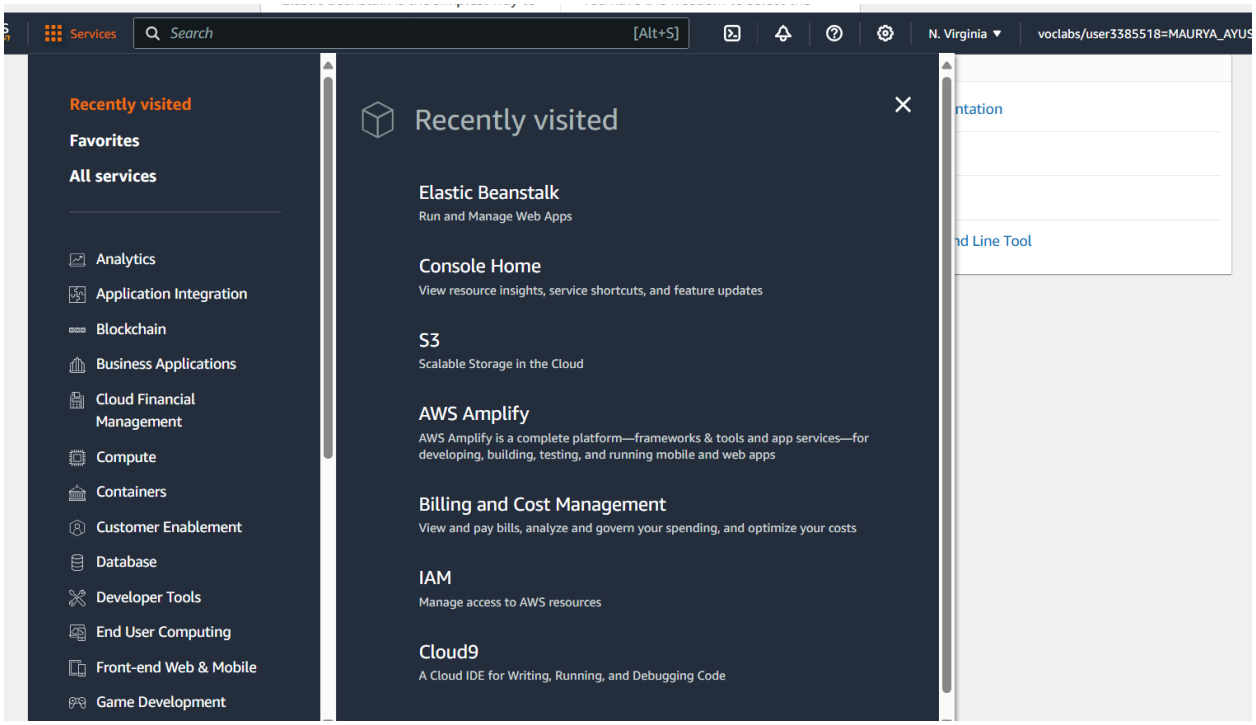
After saving, on the toolbar towards the far right, click on Share. Then put the username that you had put during creating IAM user.



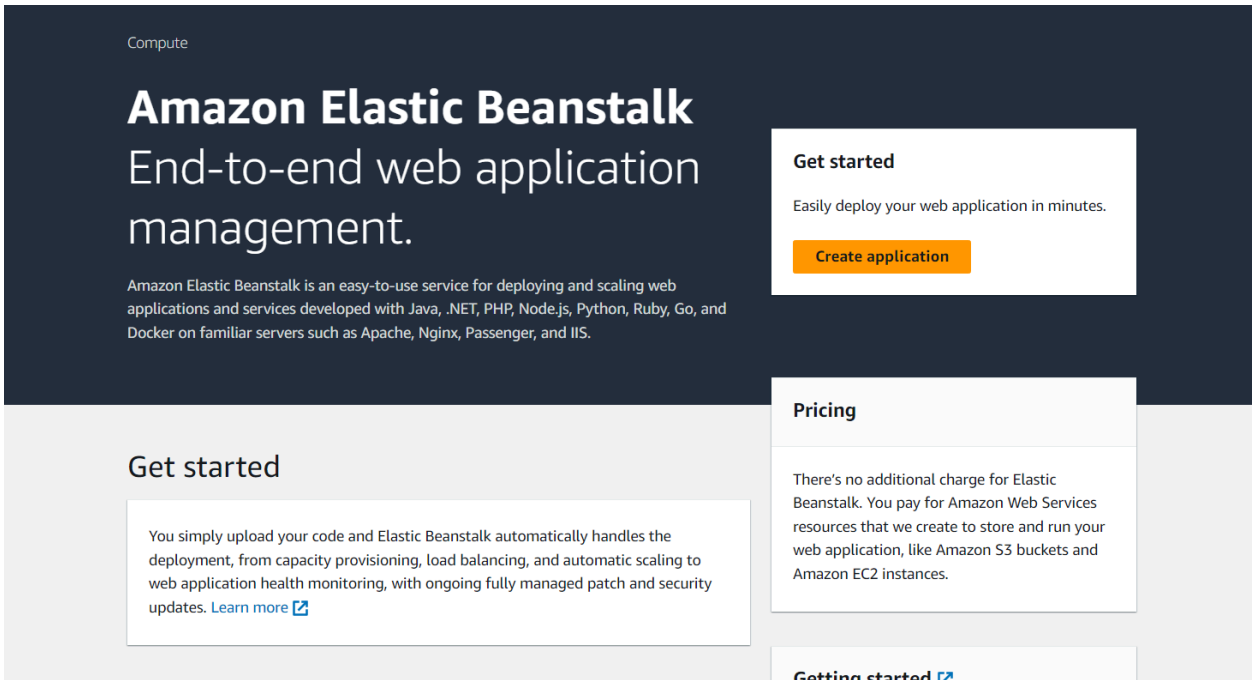
Here, it gives an error as Cloud9 was created on the academy account where creating an IAM group is not available, meanwhile on the personal account, the services of Cloud9 have been deprecated. So currently, it is not possible to integrate the cloud9 and IAM parts of the experiment.

Experiment 2

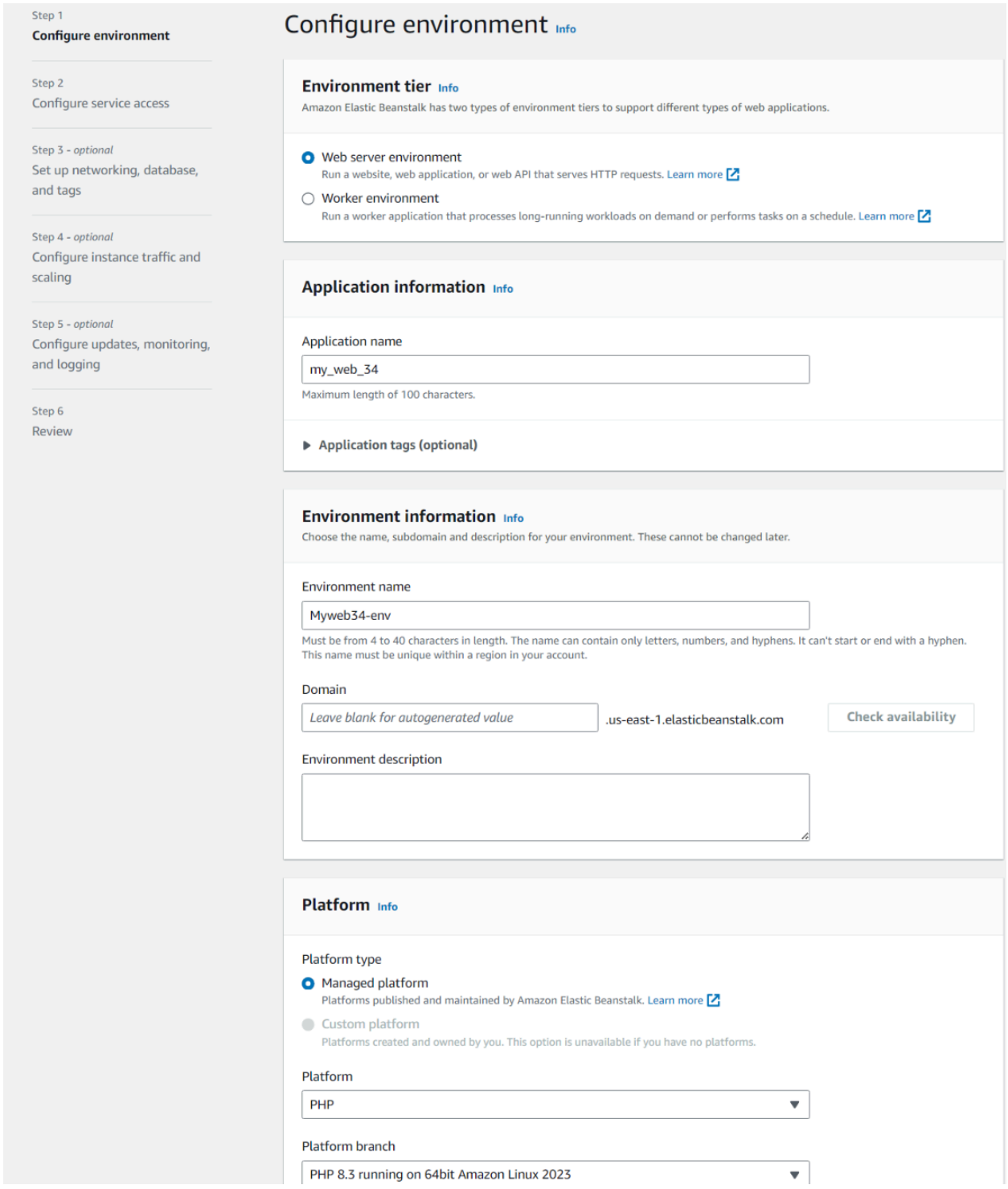
Step 1: Login to your AWS console. Search for Elastic Beanstalk in the searchbar near services.



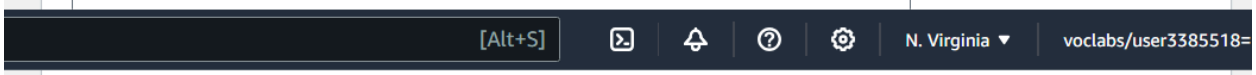
Step 2: Go to Elastic Beanstalk and click on Create Application



Step 3: Enter the name of your application. Scroll down and in the platform, select platform as PHP. Keep the application code as Sample Application. Set the instance to single instance. Click on NEXT



PHP 8.3 running on 64bit Amazon Linux 2023



4.3.1 (Recommended)

Application code Info

- (•) Sample application
- ( ) Existing version
  Application versions that you have uploaded.
- ( ) Upload your code
  Upload a source bundle from your computer or copy one from Amazon S3.

Presets Info

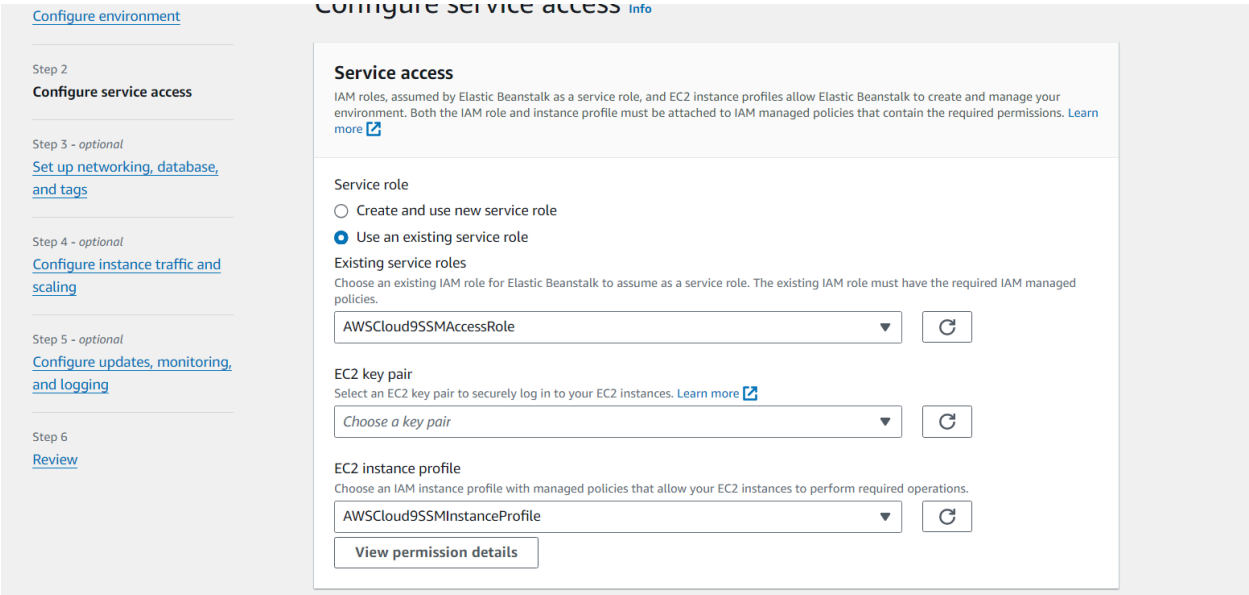
Start from a preset that matches your use case or choose custom configuration to unset recommended values and use the service's default values.

Configuration presets

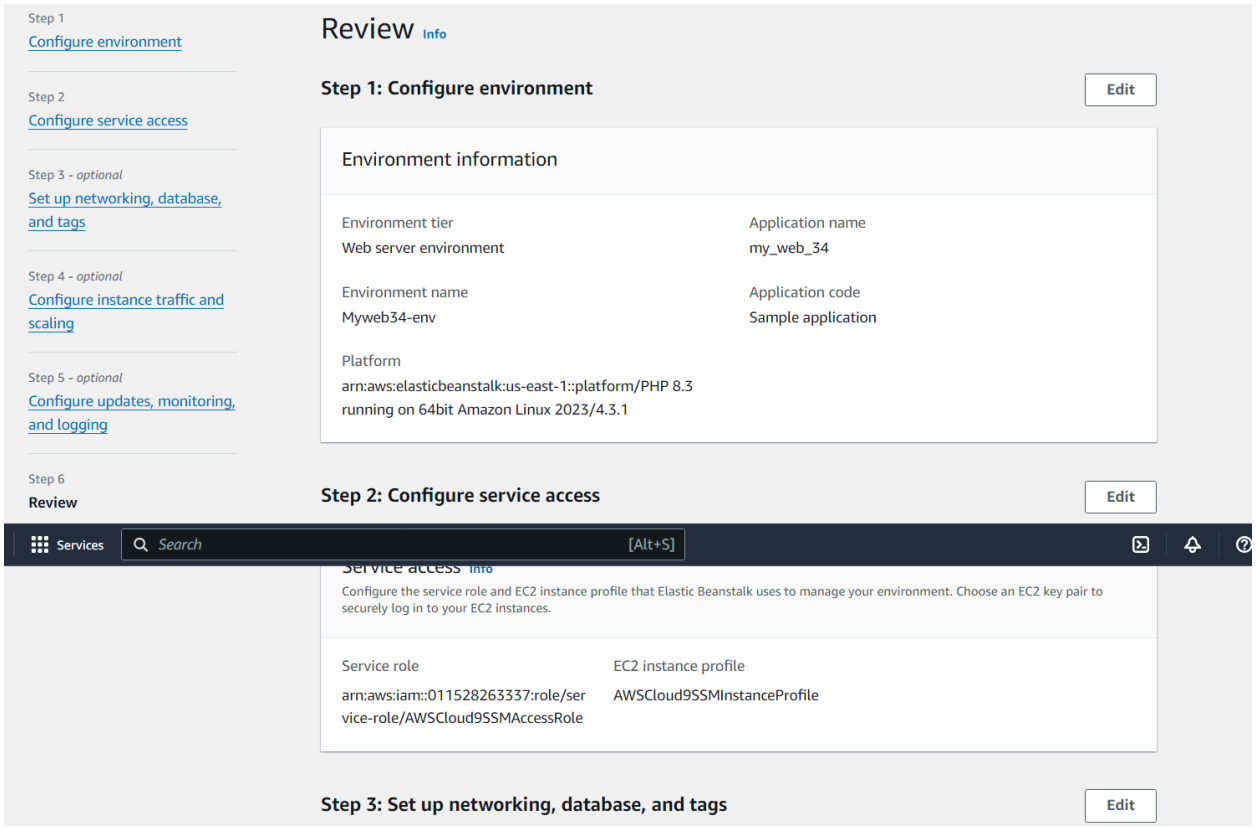
- (•) Single instance (free tier eligible)
- ( ) Single instance (using spot instance)
- ( ) High availability
- ( ) High availability (using spot and on-demand instances)
- ( ) Custom configuration

Cancel Next

Step 4 : Use an existing service role and choose whatever service role is available on your account.



Step 5 : Review the settings that you have set up for your application and submit your application.


Step 1
Configure environment
Step 2
Configure service access
Step 3 - optional
Set up networking, database, and tags
Step 4 - optional
Configure instance traffic and scaling
Step 5 - optional
Configure updates, monitoring, and logging
Step 6
Review
Review Info
Step 1: Configure environment
Edit
Environment information
Environment tier
Web server environment
Application name
my_web_34
Environment name
Myweb34-env
Application code
Sample application
Platform
arn:aws:elasticbeanstalk:us-east-1::platform/PHP 8.3 running on 64bit Amazon Linux 2023/4.3.1
Step 2: Configure service access
Edit
Services
Search
[Alt+S]
Service access Info
Configure the service role and EC2 instance profile that Elastic Beanstalk uses to manage your environment. Choose an EC2 key pair to securely log in to your EC2 instances.
Service role
arn:aws:iam::011528263337:role/service-role/AWSCloud9SSMAccessRole
EC2 instance profile
AWSCloud9SSMInstanceProfile
Step 3: Set up networking, database, and tags
Edit

Define when and how Elastic Beanstalk deploys changes to your environment. Manage your application's monitoring and logging settings, instances, and other environment resources.

Monitoring

System enhanced	Cloudwatch custom metrics - instance —	Cloudwatch custom metrics - environment —
Log streaming Deactivated	Retention 7	Lifecycle false

Updates

Managed updates Activated	Deployment batch size 100	Deployment batch size type Percentage
Command timeout 600	Deployment policy AllAtOnce	Health threshold Ok
Ignore health check	Instance replacement	

[Alt+S]

Platform software

Lifecycle false	Log streaming Deactivated	Allow URL fopen On
Display errors Off	Document root –	Max execution time 60
Memory limit 256M	Zlib output compression Off	Proxy server nginx
Logs retention 7	Rotate logs Deactivated	Update level minor
X-Ray enabled Deactivated		

Environment properties

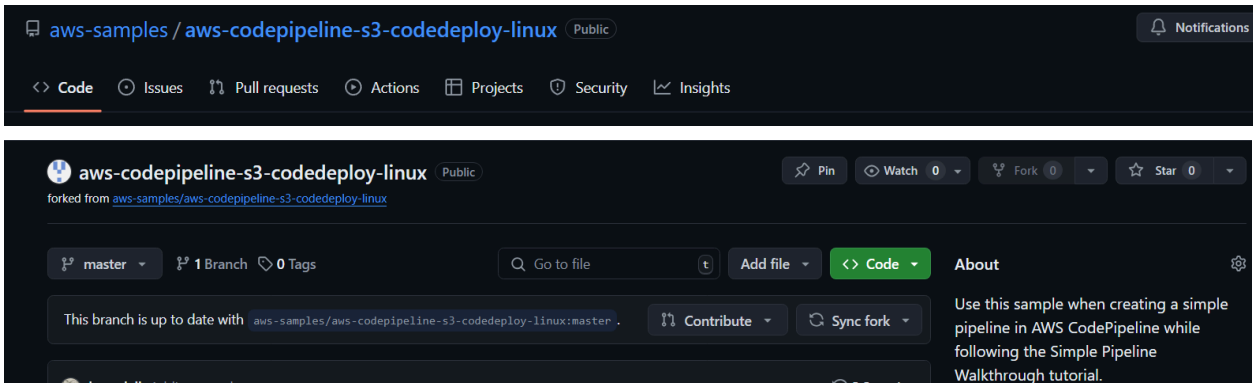
Key	Value

No environment properties

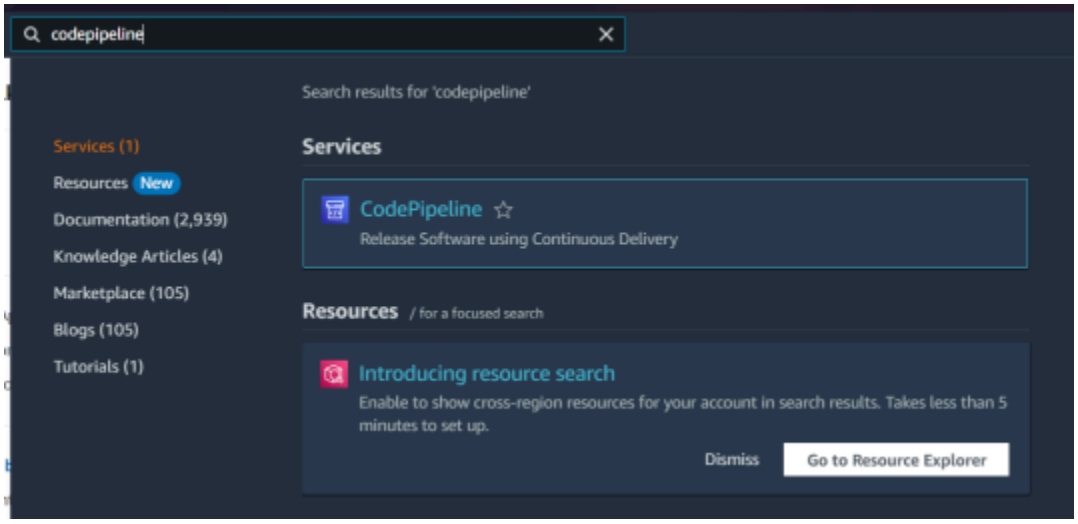
There are no environment properties defined

Cancel | Previous | Submit

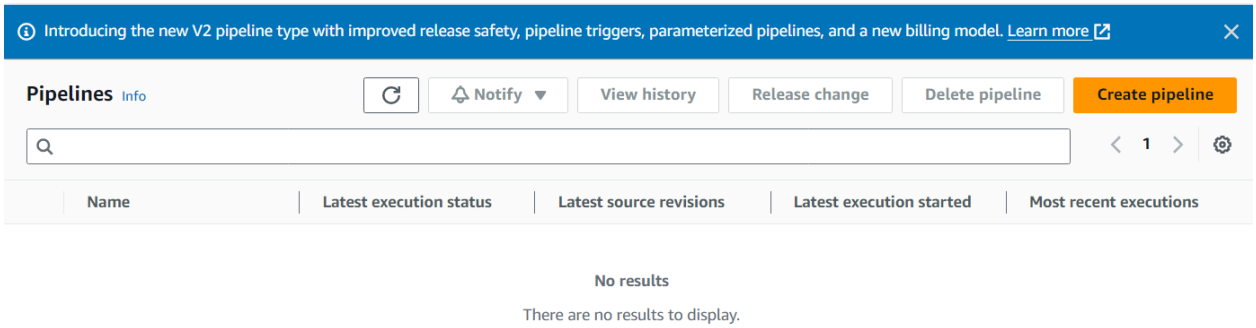
Step 6: Go to the github link below. This is a github with a sample code for deploying a file on AWS CodePipeline. Fork this repository into your personal github.
https://github.com/aws-samples/aws-codepipeline-s3-codedeploy-linux



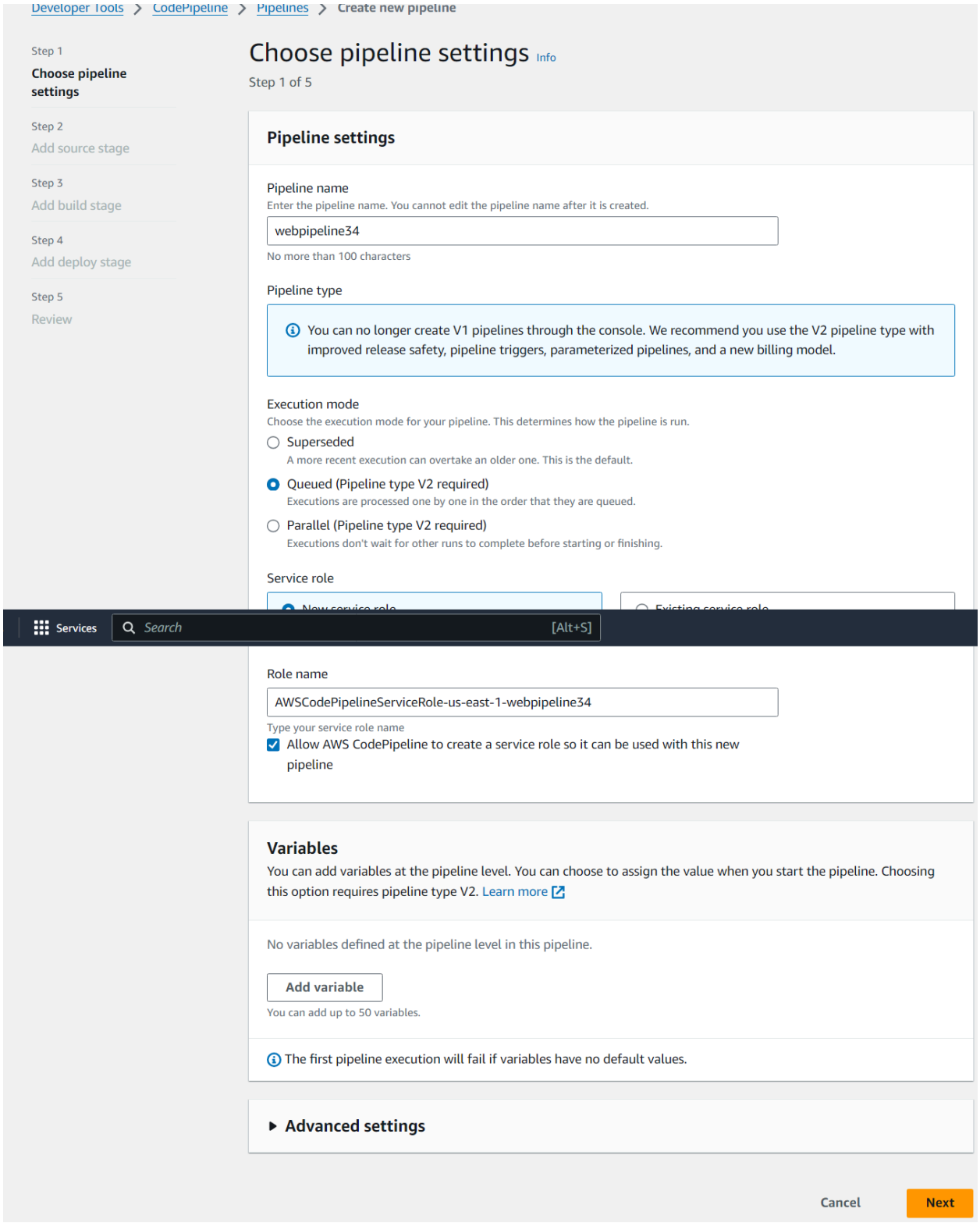
Step 7: Search CodePipeline in the services tab and click on it.



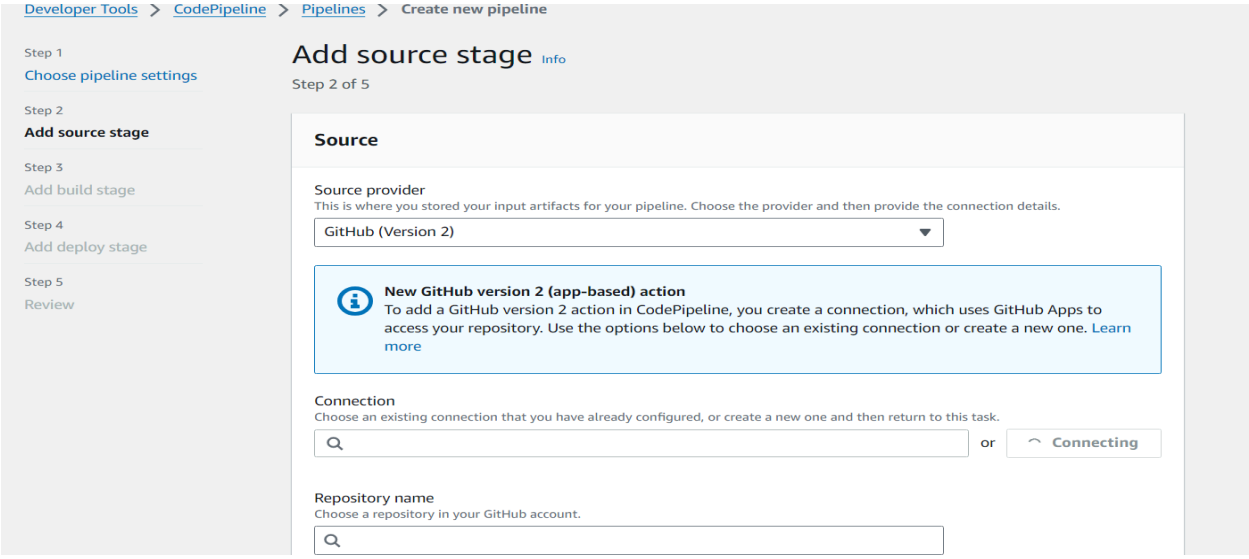
Step 8: Click on Create Pipeline.



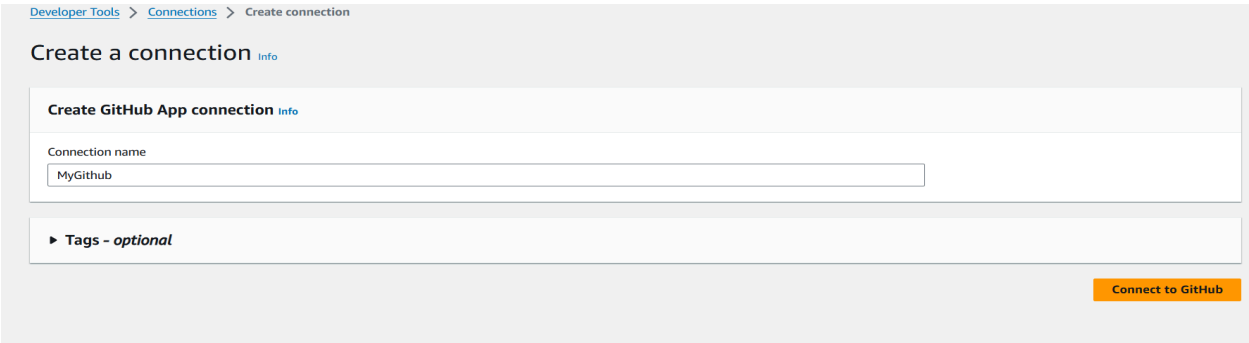
Step 9: Give a name to your Pipeline. A new service role would be created with the name of the pipeline

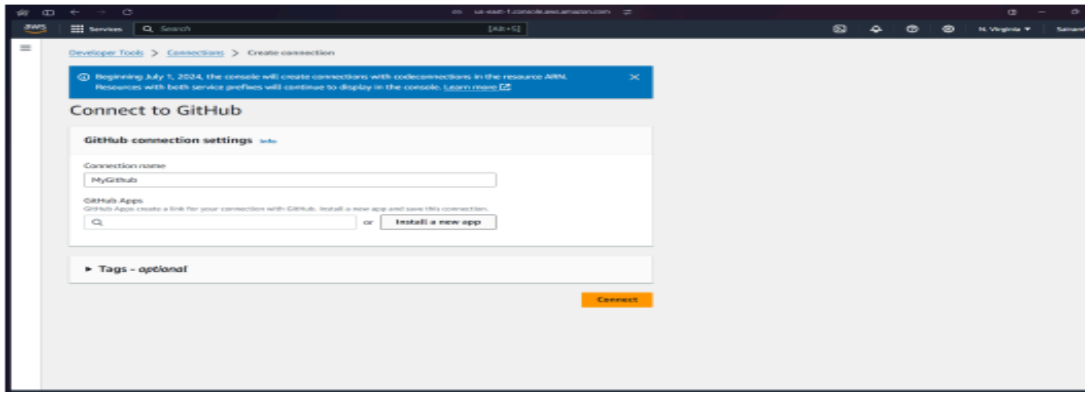


Step 10 : Select a source provider (as Github (Version 2)). Click on Connect to Github to connect your github.

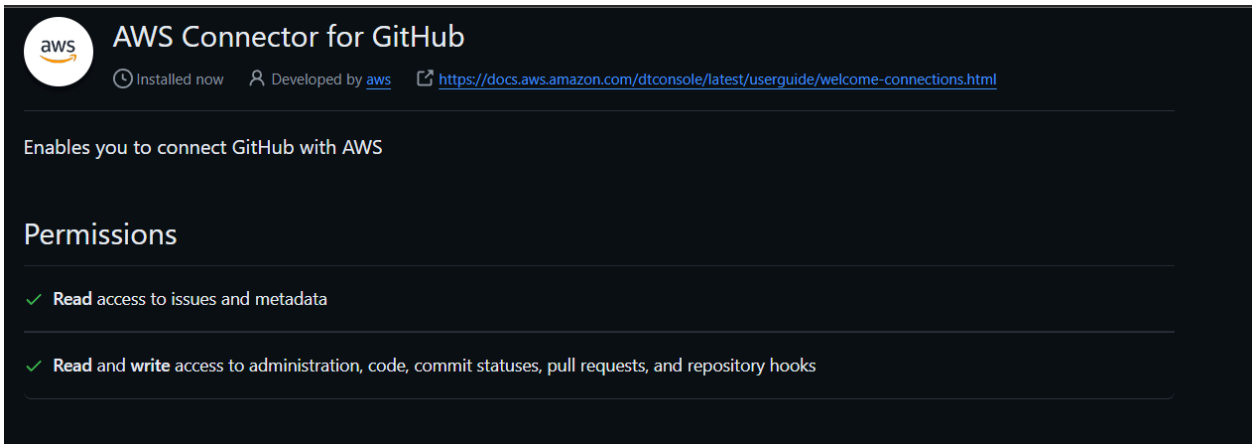


Step 11: Give a name to your GitHub app Connection and click on Connect. This will give you a prompt to either to select a GitHub app or to install a new app. If it is your first time, click on Install a new app

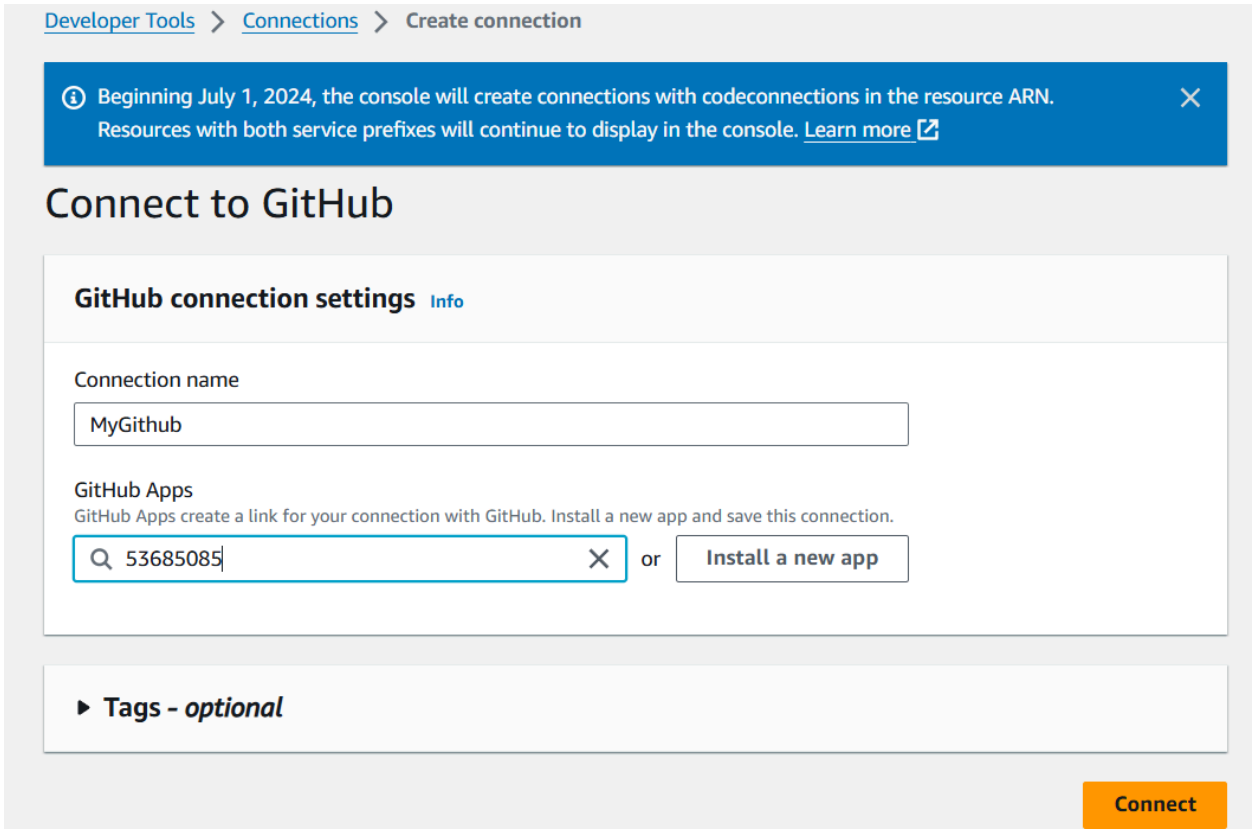


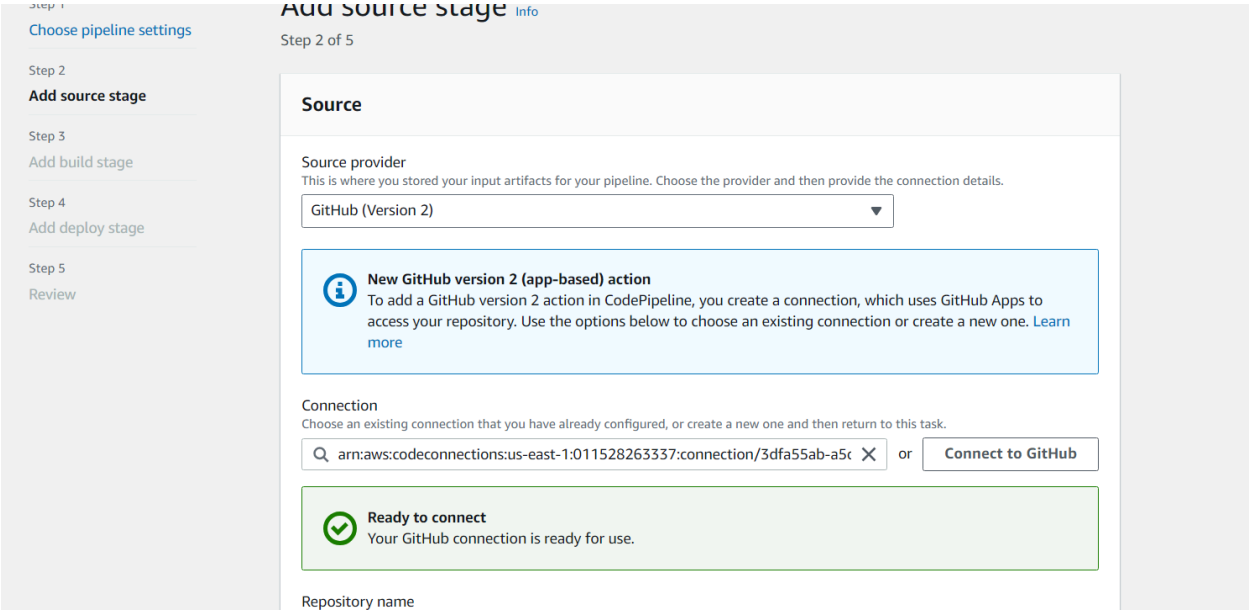


Step 12 : This will direct you to install AWS Connector On Your GitHub. Install it to your account and give it its permissions.

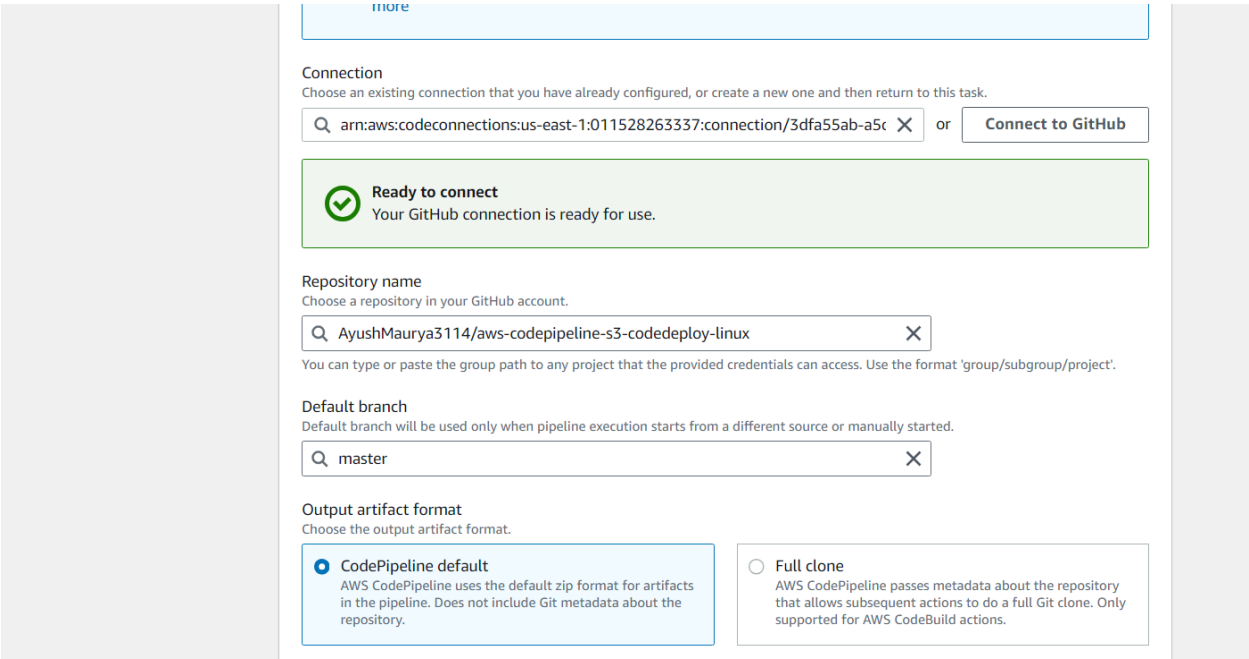


Step 13: After the app is set up, it gives the number in the text field. Click on Connect. After clicking on connect, the link is shown in the connection field and AWS shows that GitHub connection is ready to use

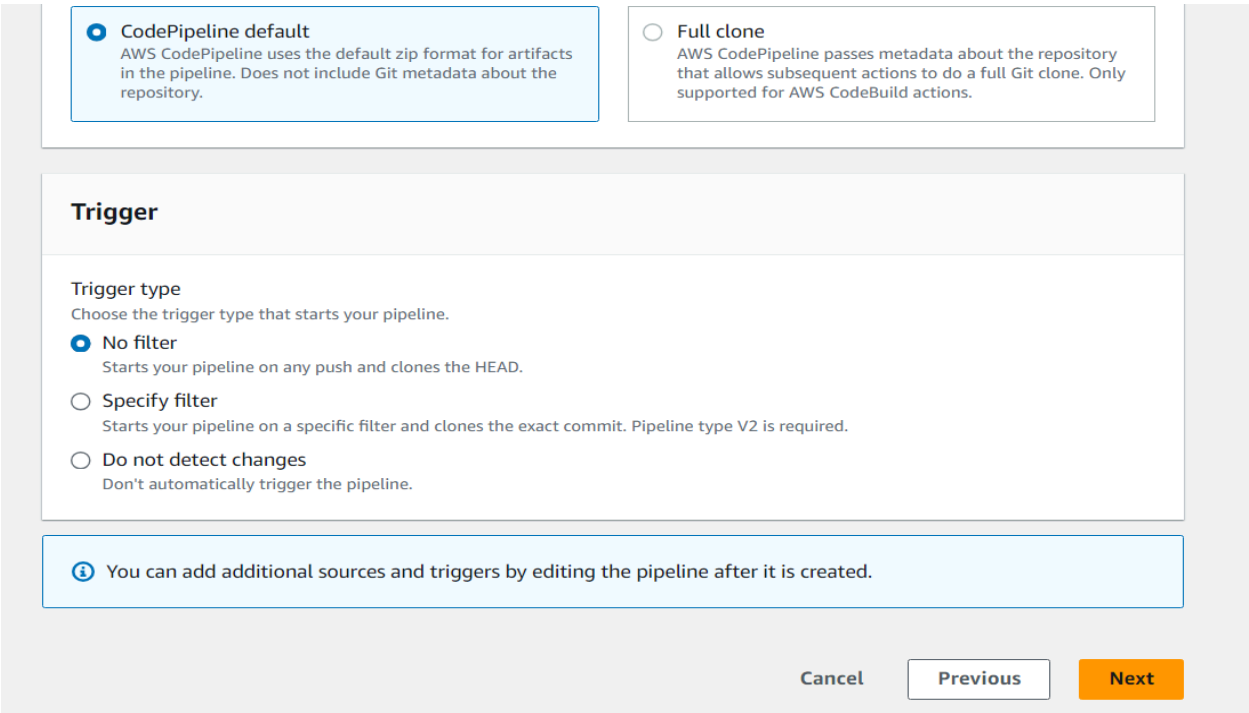




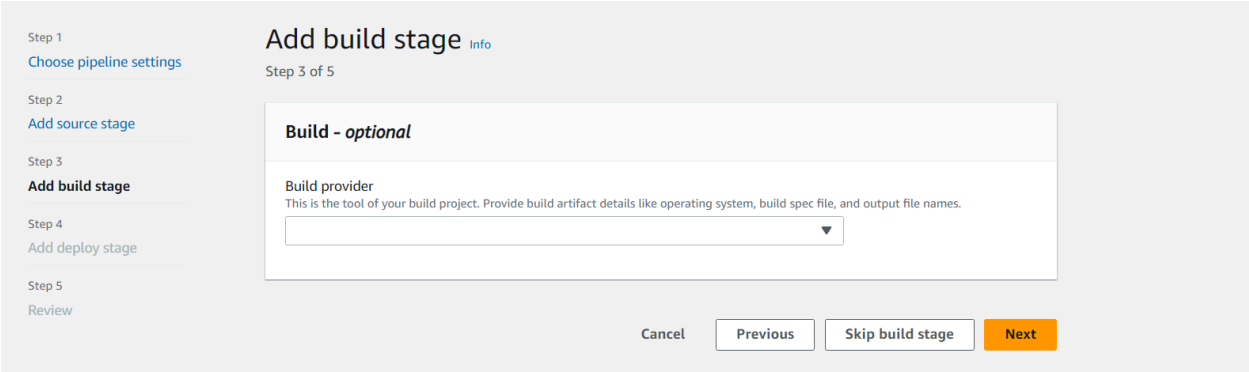
Step 14: Select the repository that you had forked to your GitHub. After that select the branch on which the files are present (default is Master).

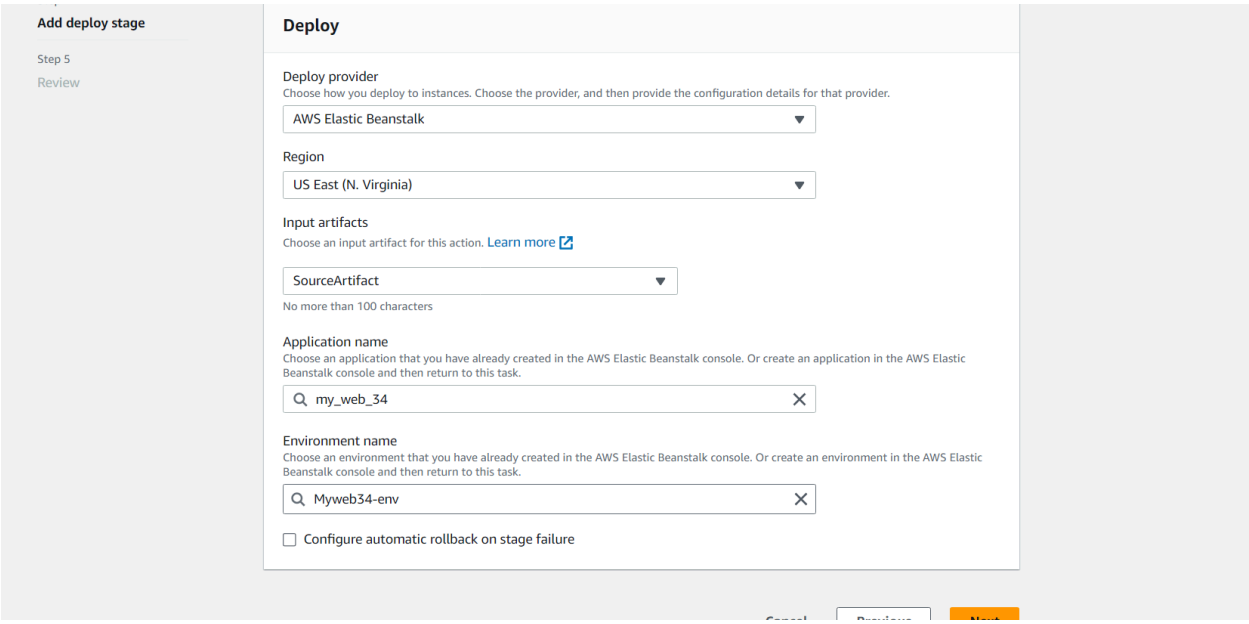


Step 15: Set the Trigger type as no filter. This would allow it to the website to update as soon as some change is made in the github.

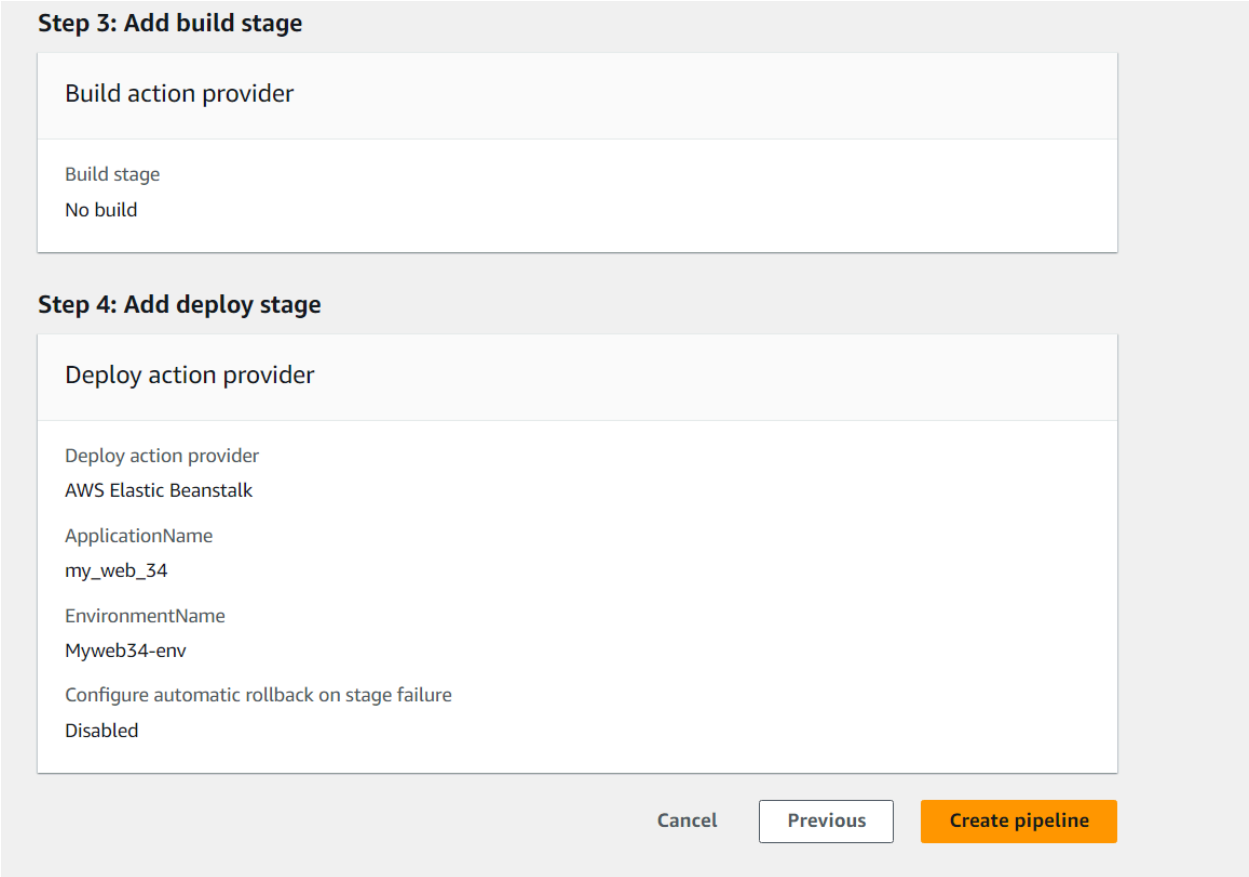


Step 16: Skip the build stage and go to Deploy. Select the deploy provider as AWS Elastic Beanstalk and Input Artifact as SourceArtifact. The application name would be the name of your Elastic Beanstalk. Then click on next.

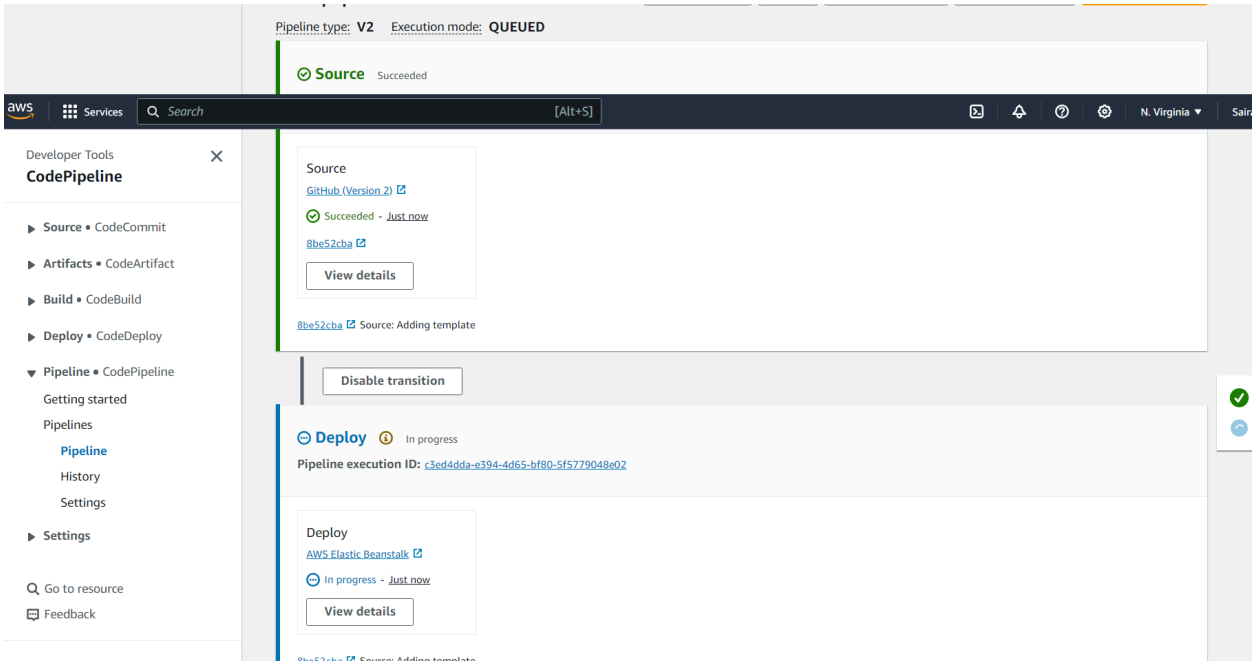




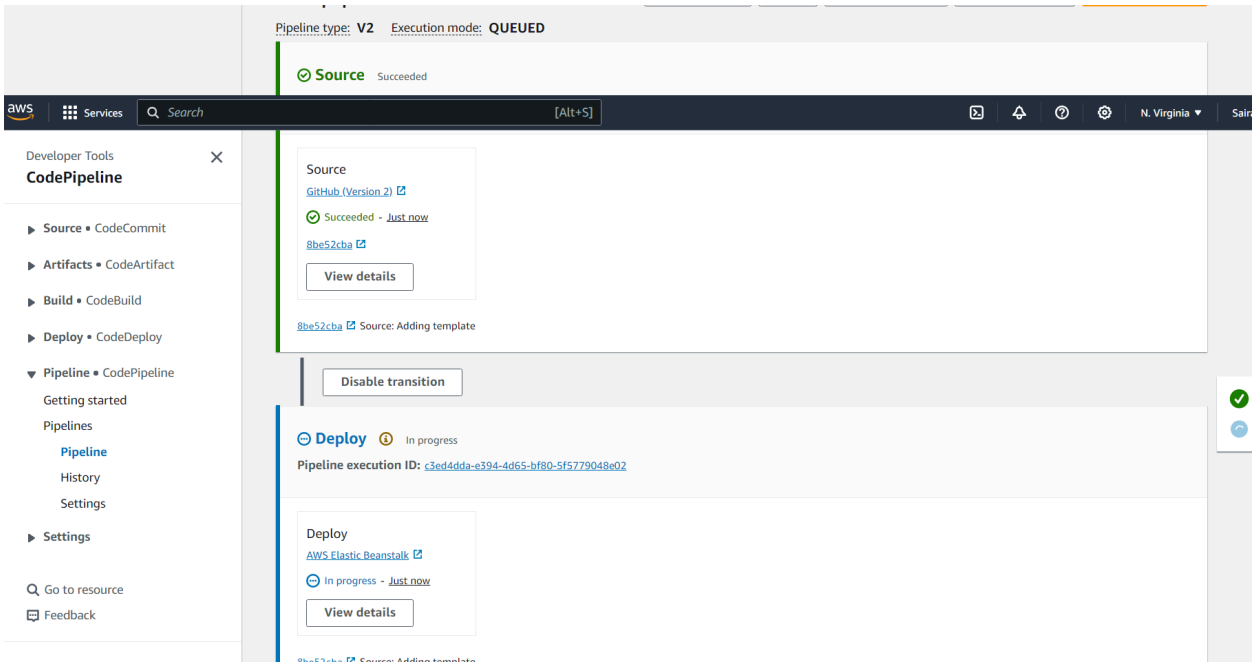
Step 17: Check all the information and click on create Pipeline.



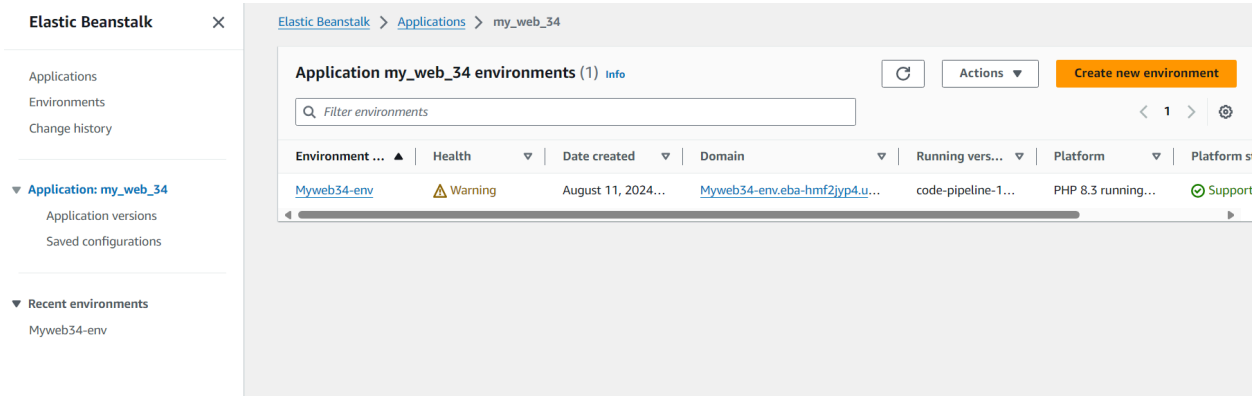
Step 18: If the pipeline is successfully deployed, this screen comes up where the source is set up and then it is transitioned to deploy.



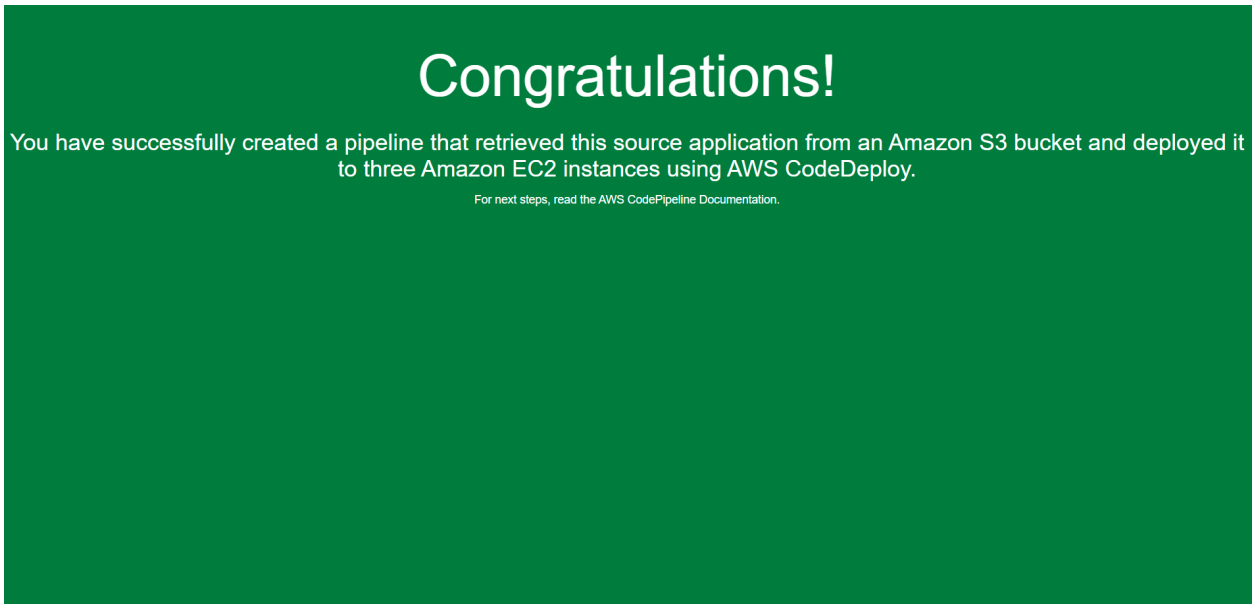
Step 19: Once the deployment is complete, click on the AWS Elastic Beanstalk under Deploy.



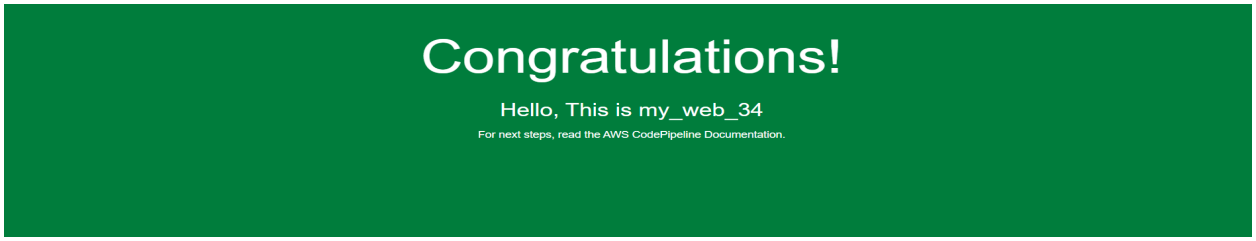
Step 20: This will redirect you to the application screen of Elastic Beanstalk. Click on the link shown under Domain.



Step 21: This will successfully show the sample website hosted.



Step 22: Now, we make some changes to the index.html file in the github.
For eg: If you make some changes to the <h2>tag.
Once the changes are committed, when the website is refreshed, the changes can be seen.



Experiment 3

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

Theory:
Container-based microservices architectures have revolutionized how development and operations teams test and deploy modern software. Containers allow companies to scale and deploy applications more efficiently, but they also introduce new challenges, adding complexity by creating a whole new infrastructure ecosystem.
Today, both large and small software companies are deploying thousands of container instances daily. Managing this level of complexity at scale requires advanced tools. Enter Kubernetes.

Originally developed by Google, Kubernetes is an open-source container orchestration platform designed to automate the deployment, scaling, and management of containerized applications. Kubernetes has quickly become the de facto standard for container orchestration and is the flagship project of the Cloud Native Computing Foundation (CNCF), supported by major players like Google, AWS, Microsoft, IBM, Intel, Cisco, and Red Hat. Kubernetes simplifies the deployment and operation of applications in a microservice architecture by providing an abstraction layer over a group of hosts. This allows development teams to deploy their applications while Kubernetes takes care of key tasks, including:

- Managing resource consumption by applications or teams
- Distributing application load evenly across the infrastructure
- Automatically load balancing requests across multiple instances of an application
- Monitoring resource usage to prevent applications from exceeding resource limits and automatically restarting them if needed
- Moving application instances between hosts when resources are low or if a host fails
- Automatically utilizing additional resources when new hosts are added to the cluster
- Facilitating canary deployments and rollbacks with ease

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

Sign in to AWS Management Console:

Go to AWS Management Console.
Log in with your account credentials.

Navigate to EC2 Service:

In the AWS Console, search for EC2 and select it to open the EC2 dashboard.

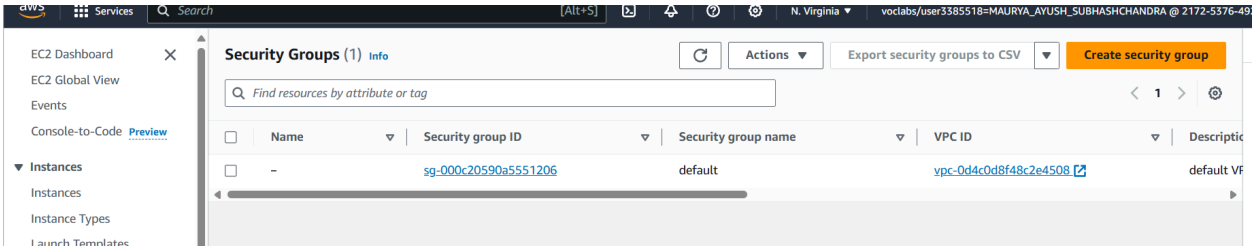
Go to "Security Groups":

On the left-hand navigation pane, under Network & Security, click on Security Groups.

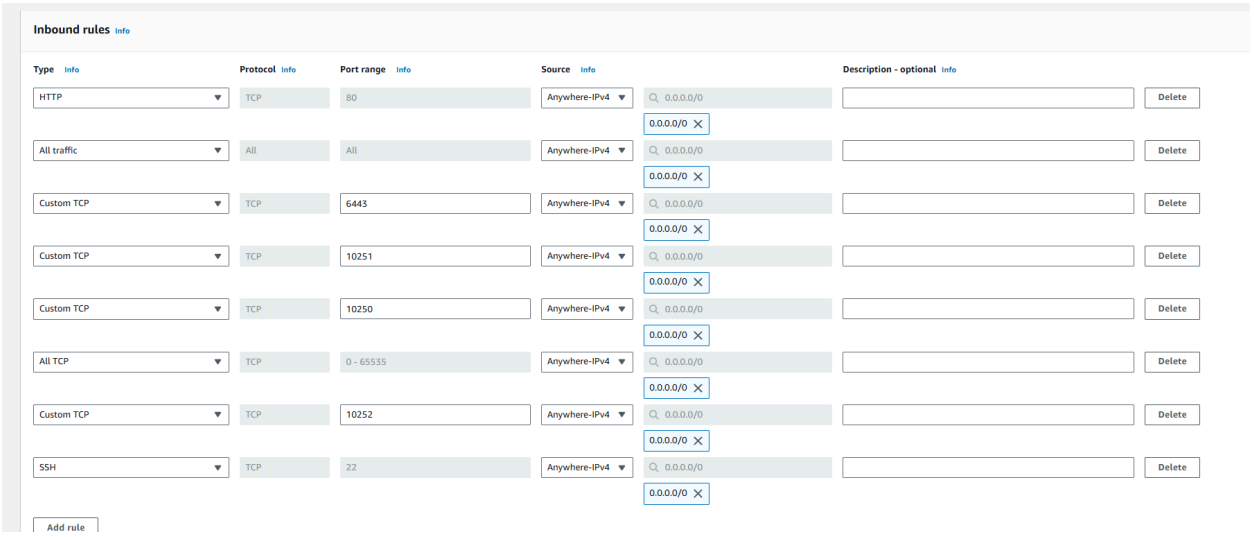
Create Security Group:

Click the Create Security Group button at the top.

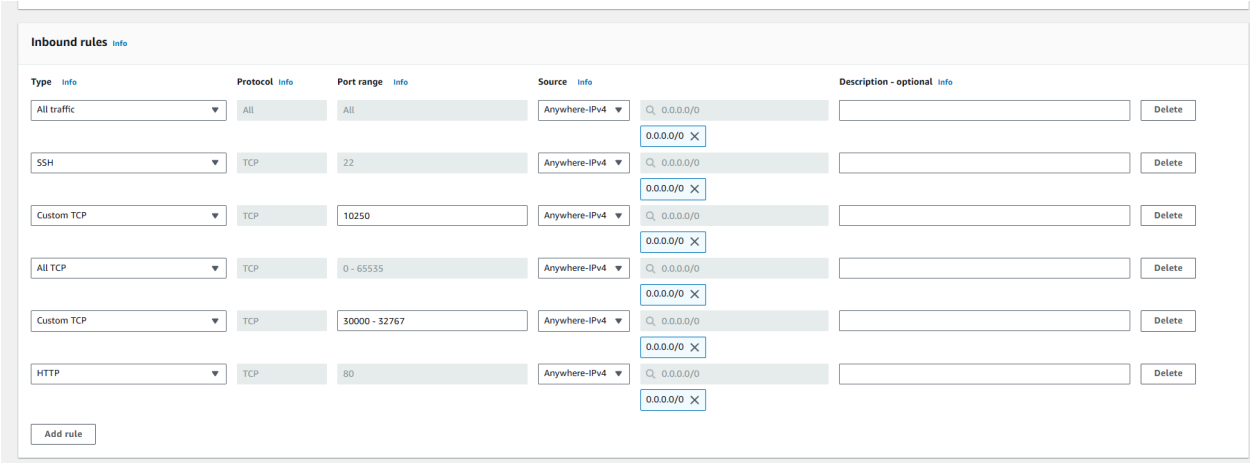
Create 2 Security Groups for Master and Nodes and add the following rules inbound rules in those Groups.



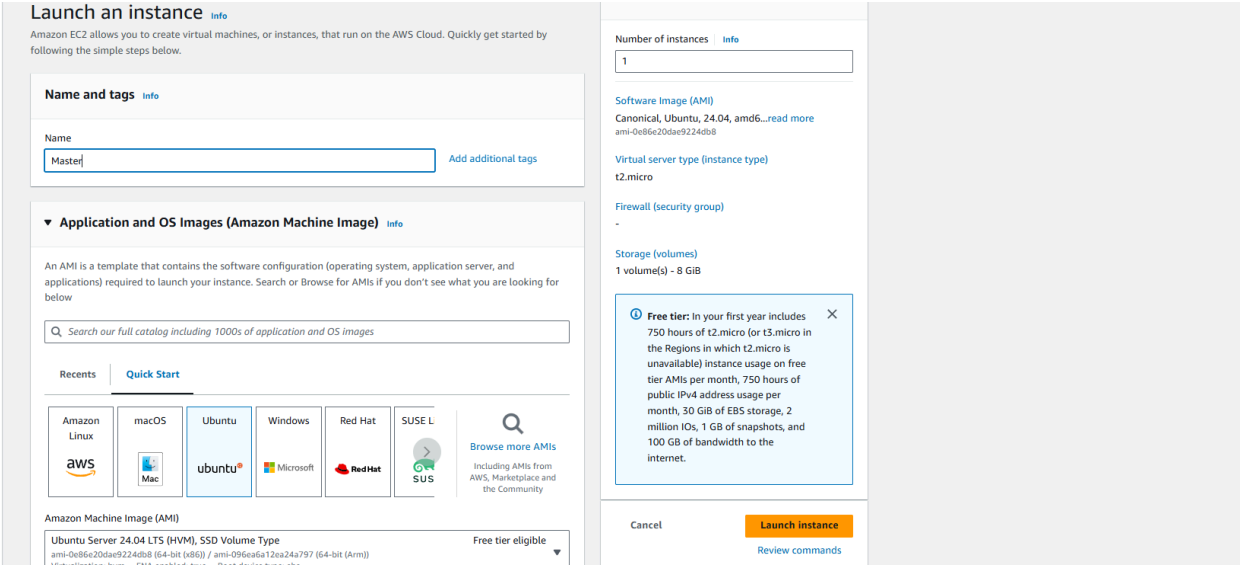
MASTER:

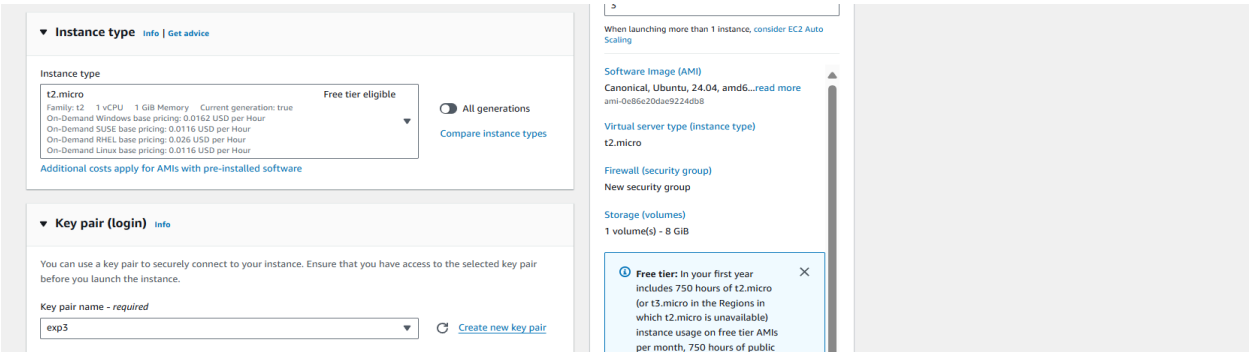


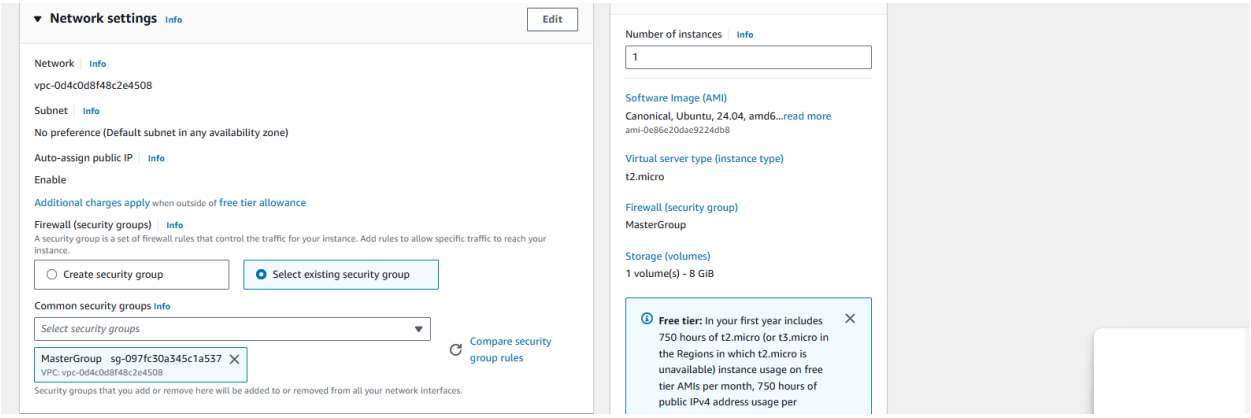
NODE:



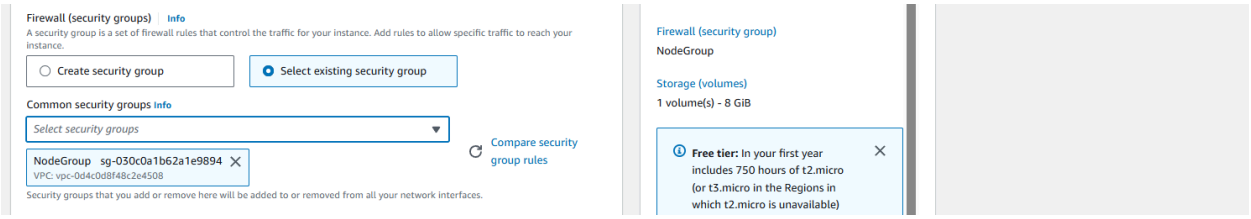
Step 1: Log in to your AWS Academy/personal account and launch 3 new Ec2 Instances. Select Ubuntu as AMI and t2.micro (because in academic account only t2.micro is present) as Instance Type and create a key of type RSA with .pem extension and move the downloaded key to the new folder.We can use 3 Different keys or 1 common key also.



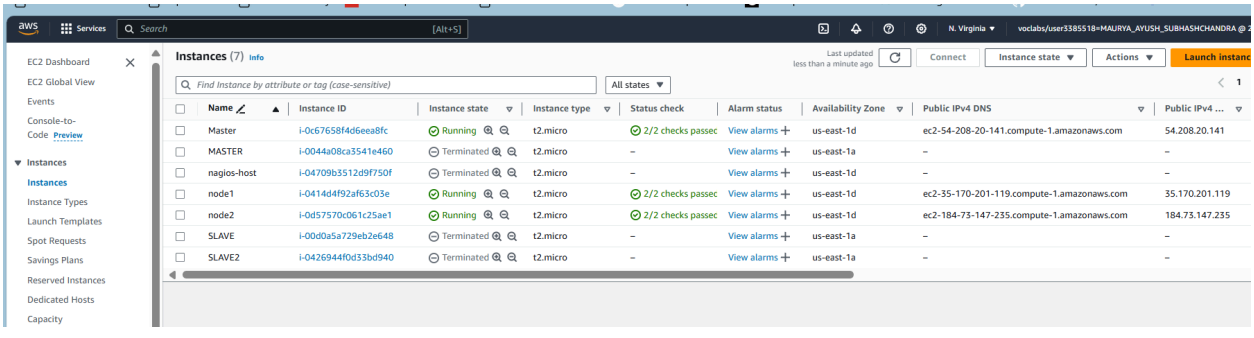




Do Same for 2 Nodes and use security groups of Node for that.

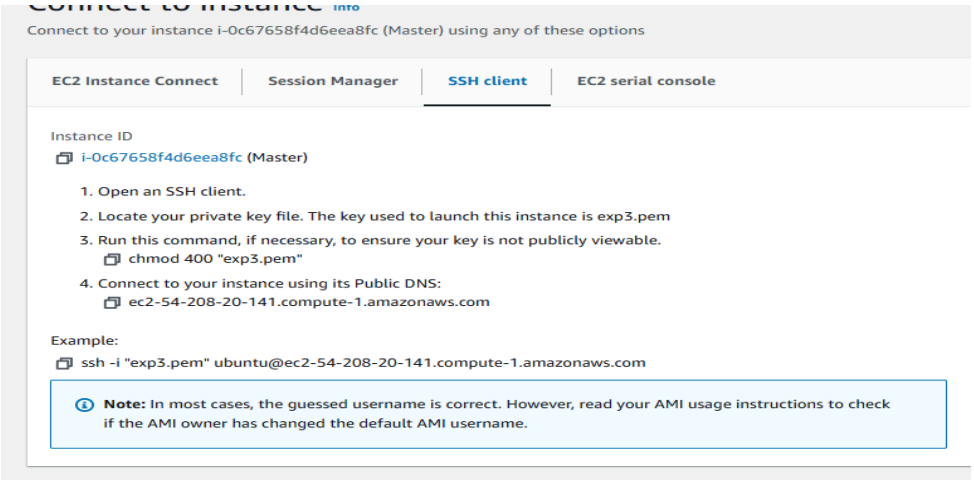


Step 2: After creating the instances click on Connect & connect all 3 instances and navigate to SSH Client.

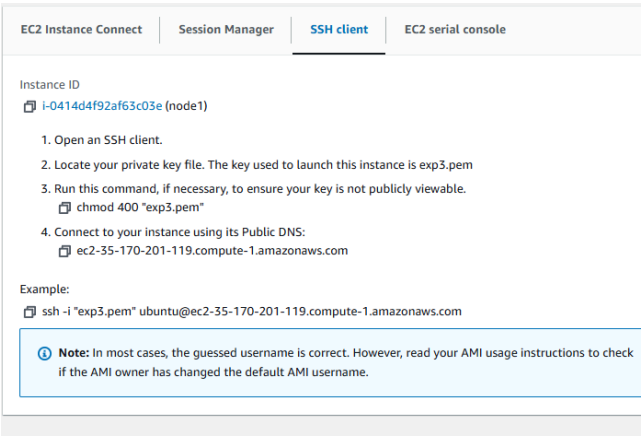


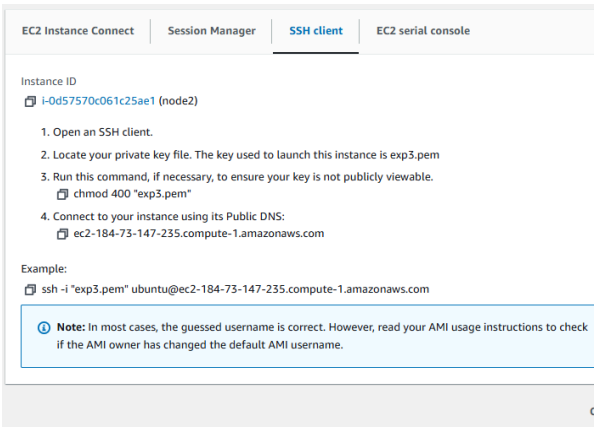
Step 3: Now open the folder in the terminal 3 times for Master, Node1& Node 2 where our .pem key is stored and paste the Example command (starting with ssh -i .....) in the terminal.
ssh -i "<PATH TO FILE>exp3.pem" ubuntu@ec2-54-208-20-141.compute-1.amazonaws.com

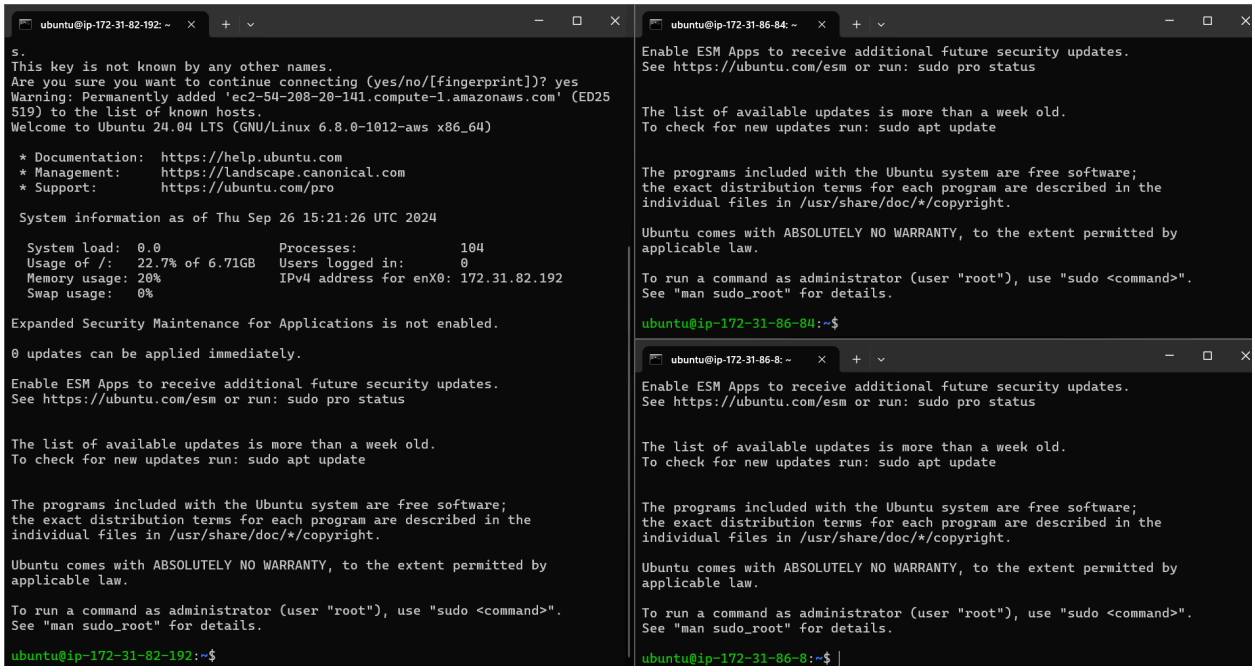
MASTER:



NODE 1&2:







Step 4: Run on Master,Node 1,and Node 2 the below commands to install and setup Docker in Master, Node1, and Node2.

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/trusted.gpg.d/docker.gpg > /dev/null

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

```
ubuntu@ip-172-31-82-192:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
```

```
ubuntu@ip-172-31-82-192:~$ curl -fsSL https://download.docker.com/linux/ubun
tu/gpg | sudo tee /etc/apt/trusted.gpg.d/docker.gpg > /dev/null
ubuntu@ip-172-31-82-192:~$ sudo add-apt-repository "deb [arch=amd64] https:/
/download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
Repository: 'deb [arch=amd64] https://download.docker.com/linux/ubuntu noble
 stable'
Description:
Archive for codename: noble components: stable
More info: https://download.docker.com/linux/ubuntu
Adding repository.
Press [ENTER] to continue or Ctrl-c to cancel.
Adding deb entry to /etc/apt/sources.list.d/archive_uri-https_download_docke
r_com_linux_ubuntu-noble.list
Adding disabled deb-src entry to /etc/apt/sources.list.d/archive_uri-https_d
ownload_docker_com_linux_ubuntu-noble.list
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
 [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelea
se [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 https://download.docker.com/linux/ubuntu noble InRelease [48.8 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Pa
```

```
Get:50 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiv
erse amd64 Components [212 B]
Get:51 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiv
erse amd64 c-n-f Metadata [116 B]
Get:52 https://download.docker.com/linux/ubuntu noble/stable amd64 Packages
[15.3 kB]
Fetched 29.1 MB in 6s (4873 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 h
as an unsupported filetype.
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is st
ored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATI
ON section in apt-key(8) for details.
ubuntu@ip-172-31-82-192:~$
```

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

```
ubuntu@ip-172-31-82-192:~$ 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 InRelea
se
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
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 h
as an unsupported filetype.
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is st
ored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATI
ON section 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-plugin libltdl7 libslirp0 pigz
  slirp4netns
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite
The following NEW packages will be installed:
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli
```

```
r/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.
ubuntu@ip-172-31-82-192:~$
```

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

```
No VM guests are running outdated hypervisor (qemu) binaries on this ho
ubuntu@ip-172-31-82-192:~$ sudo mkdir -p /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
ubuntu@ip-172-31-82-192:~$
```

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

```
ubuntu@ip-172-31-82-192:~$ 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
ubuntu@ip-172-31-82-192:~$
```

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-172-31-82-192:~$ # Add the Kubernetes GPG key and save it to the K
eyring
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gp
g --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

# Add the Kubernetes repository to your APT sources list
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://p
kgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/ku
bernetes.list

# Update package lists
sudo apt update
File '/etc/apt/keyrings/kubernetes-apt-keyring.gpg' exists. Overwrite? (y/N)
 y
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8
s.io/core:/stable:/v1.31/deb/ /
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 InRelea
se
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Hit:6 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/s
table:/v1.31/deb  InRelease [1186 B]
Get:7 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/s
table:/v1.31/deb  Packages [4865 B]
Fetched 6051 B in 1s (6276 B/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
142 packages can be upgraded. Run 'apt list --upgradable' to see them.
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 h
as an unsupported filetype.
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is st
ored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATI
ON section in apt-key(8) for details.
ubuntu@ip-172-31-82-192:~$
```

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

```
ubuntu@ip-172-31-82-192:~$ 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 InRelea
se
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Hit:6 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/s
table:/v1.31/deb  InRelease
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 h
as an unsupported filetype.
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is st
ored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATI
ON section 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 142 not upgraded.
Need to get 87.4 MB of archives.
After this operation, 314 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 conntr
ack amd64 1:1.4.8-1ubuntu1 [37.9 kB]
Get:2 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/s
table:/v1.31/deb  cri-tools 1.31.1-1.1 [15.7 MB]
Get:3 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/s
table:/v1.31/deb  kubeadm 1.31.1-1.1 [11.4 MB]
Get:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/s
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.
ubuntu@ip-172-31-82-192:~$
```

sudo systemctl enable --now kubelet
sudo apt-get install -y containerd

```
ubuntu@ip-172-31-82-192:~$ sudo systemctl enable --now kubelet
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 requir
ed:
  docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras
  docker-compose-plugin libltdl7 libslirp0 pigz slirp4netns
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  runc
The following packages will be REMOVED:
  containerd.io docker-ce
The following NEW packages will be installed:
  containerd runc
0 upgraded, 2 newly installed, 2 to remove and 142 not upgraded.
Need to get 47.2 MB of archives.
After this operation, 53.1 MB disk space will be freed.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd6
```

```
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.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
```

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

```
ubuntu@ip-172-31-82-192:~$ 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 = ""
```

```
[proxy_plugins]

[stream_processors]

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

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

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

[ttrpc]
  address = ""
  gid = 0
  uid = 0
ubuntu@ip-172-31-82-192:~$ |
```

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

```
ubuntu@ip-172-31-82-192:~$ 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; p>
     Active: active (running) since Thu 2024-09-26 16:19:13 UTC; 499ms ago
       Docs: https://containerd.io
   Main PID: 5387 (containerd)
      Tasks: 6
     Memory: 15.0M (peak: 15.4M)
        CPU: 78ms
     CGroup: /system.slice/containerd.service
             └─5387 /usr/bin/containerd

Sep 26 16:19:13 ip-172-31-82-192 containerd[5387]: time="2024-09-26T16:19:1>
Sep 26 16:19:13 ip-172-31-82-192 containerd[5387]: time="2024-09-26T16:19:1>
Sep 26 16:19:13 ip-172-31-82-192 containerd[5387]: time="2024-09-26T16:19:1>
Sep 26 16:19:13 ip-172-31-82-192 containerd[5387]: time="2024-09-26T16:19:1>
Sep 26 16:19:13 ip-172-31-82-192 containerd[5387]: time="2024-09-26T16:19:1>
Sep 26 16:19:13 ip-172-31-82-192 containerd[5387]: time="2024-09-26T16:19:1>
Sep 26 16:19:13 ip-172-31-82-192 containerd[5387]: time="2024-09-26T16:19:1>
Sep 26 16:19:13 ip-172-31-82-192 containerd[5387]: time="2024-09-26T16:19:1>
Sep 26 16:19:13 ip-172-31-82-192 systemd[1]: Started containerd.service - c>
Sep 26 16:19:13 ip-172-31-82-192 containerd[5387]: time="2024-09-26T16:19:1>
lines 1-21/21 (END)
```

sudo apt-get install -y socat

```
ubuntu@ip-172-31-82-192:~$ 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 requir
ed:
  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 NEW packages will be installed:
  socat
0 upgraded, 1 newly installed, 0 to remove and 142 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 1.8.0.0-4build3 [374 kB]
Fetched 374 kB in 0s (15.7 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.
ubuntu@ip-172-31-82-192:~$
```

Step 6: Initialize the Kubercluster. Now perform this on Master Instance.
sudo kubeadm init --pod-network-cidr=10.244.0.0/16

```
ubuntu@ip-172-31-82-192:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errors=NumCPU,Mem
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
        [WARNING NumCPU]: the number of available CPUs 1 is less than the required 2
        [WARNING Mem]: the system RAM (957 MB) is less than the minimum 1700 MB
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action beforehand using 'kubeadm config images pull'
W0926 16:55:13.630110    7239 checks.go:846] detected that the sandbox image "registry.k8s.io/pause:3.8" of the container runtime is inconsistent with that used by kubeadm.It is recommended to use "registry.k8s.io/pause:3.10" as the CRI sandbox image.
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [ip-172-31-82-192 kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 172.31.82.192]
[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-172-31-82-192 localhost] and IPs [172.31.82.192 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [ip-172-31-82-192 localhost] and IPs [172.31.82.192 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "super-admin.conf" kubeconfig file
[kubeconfig] Writing "kubelet.conf" kubeconfig file
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
[kubeconfig] Writing "scheduler.conf" kubeconfig file
[etcd] Creating static Pod manifest for local etcd in "/etc/kubernetes/manifests"
[control-plane] Using manifest folder "/etc/kubernetes/manifests"
[control-plane] Creating static Pod manifest for "kube-apiserver"
[control-plane] Creating static Pod manifest for "kube-controller-manager"
[control-plane] Creating static Pod manifest for "kube-scheduler"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Starting the kubelet
[wait-control-plane] Waiting for the kubelet to boot up the control plane as static Pods from directory "/etc/kubernetes/manifests"
[kubelet-check] Waiting for a healthy kubelet at http://127.0.0.1:10248/healthz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 1.533243039s
[api-check] Waiting for a healthy API server. This can take up to 4m0s
[api-check] The API server is healthy after 11.505850128s
[upload-config] Storing the configuration used in ConfigMap "kubeadm-config" in the "kube-system" Namespace
[kubelet] Creating a ConfigMap "kubelet-config" in namespace kube-system with the configuration for the kubelets in the cluster
[upload-certs] Skipping phase. Please see --upload-certs
[mark-control-plane] Marking the node ip-172-31-82-192 as control-plane by adding the labels: [node-role.kubernetes.io/control-plane node.kubernetes.io/exclude-from-external-load-balancers]
[mark-control-plane] Marking the node ip-172-31-82-192 as control-plane by adding the taints [node-role.kubernetes.io/control-plane:NoSchedule]
[bootstrap-token] Using token: jrhztc.eyi07duk03zq4eqr
[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 172.31.82.192:6443 --token jrhztc.eyi07duk03zq4eqr \
        --discovery-token-ca-cert-hash sha256:a6037b3c6608d5fdadd8dfd100793d17c759dbeb046a5ccd908a90f2e52d2055
```

From this command we get token and ca-
Run this command on master and

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-172-31-82-192:~$ mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
ubuntu@ip-172-31-82-192:~$
```

Step 7: Now Run the command **kubectl get nodes** to see the nodes before executing Join command on nodes.

```
ubuntu@ip-172-31-82-192:~$ kubectl get nodes
NAME              STATUS     ROLES           AGE     VERSION
ip-172-31-82-192   NotReady   control-plane   7m21s   v1.31.1
ubuntu@ip-172-31-82-192:~$
```

Step 8: Now Run the following command on Node 1 and Node 2 to Join to master.
sudo kubeadm join <your-master-node-ip>:6443 --token <your-token>
--discovery-token-ca-cert-hash sha256:<your-ca-cert-hash>

*kubeadm join 172.31.82.192:6443 --token jrhztc.eyi07duk03zq4eqr *
--discovery-token-ca-cert-hash
sha256:a6037b3c6608d5fdadd8dfd100793d17c759dbeb046a5ccd908a90f2e52d2055
(SLASH '\' MIGHT GIVE ERROR IF IT GIVES ERROR THEN TRY WITHOUT ERROR)

NODE1:

```
ubuntu@ip-172-31-86-84:~$ sudo kubeadm join 172.31.82.192:6443 --token jrhztc.eyi07duk03zq4eqr --discovery-token-ca-cert-hash sha256:a6037b3c6608d5fdadd8dfd
100793d17c759dbeb046a5ccd908a90f2e52d2055
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-check] Waiting for a healthy kubelet at http://127.0.0.1:10248/healthz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 1.004592587s
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap

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

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

ubuntu@ip-172-31-86-84:~$
```

NODE2:

```
ubuntu@ip-172-31-86-8:~$ sudo kubeadm join 172.31.82.192:6443 --token jrhztc.eyi07duk03zq4eqr --discovery-token-ca-cert-hash sha256:a6037b3c6608d5fdadd8dfd1
00793d17c759dbeb046a5ccd908a90f2e52d2055
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-check] Waiting for a healthy kubelet at http://127.0.0.1:10248/healthz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 1.003588565s
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap

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

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

Step 9: Now Run the command on Master **kubectl get nodes** to see the nodes after executing Join command on nodes.

```
ubuntu@ip-172-31-82-192:~$ kubectl get nodes
NAME               STATUS     ROLES           AGE     VERSION
ip-172-31-82-192   NotReady   control-plane   55m     v1.31.1
ip-172-31-86-8     NotReady   <none>          7m44s   v1.31.1
ip-172-31-86-84    NotReady   <none>          7m5s    v1.31.1
ubuntu@ip-172-31-82-192:~$
```

Step 10: Since Status is NotReady we have to add a network plugin. And also we have to give the name to the nodes.

kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml

```
ubuntu@ip-172-31-82-192:~$ kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml
poddisruptionbudget.policy/calico-kube-controllers created
serviceaccount/calico-kube-controllers created
serviceaccount/calico-node created
configmap/calico-config created
customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgppeers.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/blockaffinities.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/caliconodestatuses.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/clusterinformations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/felixconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworksets.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/hostendpoints.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamblocks.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamconfigs.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamhandles.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ippools.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipreservations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/kubecontrollersconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networksets.crd.projectcalico.org created
clusterrole.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrole.rbac.authorization.k8s.io/calico-node created
clusterrolebinding.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrolebinding.rbac.authorization.k8s.io/calico-node created
daemonset.apps/calico-node created
deployment.apps/calico-kube-controllers created
ubuntu@ip-172-31-82-192:~$
```

sudo systemctl status kubelet

```
ubuntu@ip-172-31-82-192:~$ sudo systemctl status kubelet
● kubelet.service - kubelet: The Kubernetes Node Agent
     Loaded: loaded (/usr/lib/systemd/system/kubelet.service; enabled; preset: enabled)
    Drop-In: /usr/lib/systemd/system/kubelet.service.d
             └─10-kubeadm.conf
     Active: active (running) since Thu 2024-09-26 16:55:46 UTC; 57min ago
       Docs: https://kubernetes.io/docs/
   Main PID: 7822 (kubelet)
      Tasks: 10 (limit: 1130)
     Memory: 49.2M (peak: 72.1M)
        CPU: 44.924s
     CGroup: /system.slice/kubelet.service
             └─7822 /usr/bin/kubelet --bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kube>

Sep 26 17:52:55 ip-172-31-82-192 kubelet[7822]: E0926 17:52:55.304654    7822 kuberuntime_>
Sep 26 17:52:55 ip-172-31-82-192 kubelet[7822]:         rpc error: code = Unknown desc = f>
Sep 26 17:52:55 ip-172-31-82-192 kubelet[7822]:         : unknown
Sep 26 17:52:55 ip-172-31-82-192 kubelet[7822]:  > pod="kube-system/kube-scheduler-ip-172->
Sep 26 17:52:55 ip-172-31-82-192 kubelet[7822]: E0926 17:52:55.393676    7822 log.go:32] "R>
Sep 26 17:52:55 ip-172-31-82-192 kubelet[7822]:         rpc error: code = Unknown desc = f>
Sep 26 17:52:55 ip-172-31-82-192 kubelet[7822]:         : unknown
Sep 26 17:52:55 ip-172-31-82-192 kubelet[7822]:  > podSandboxID="af90c21fb06b79773120a5ffc>
Sep 26 17:52:55 ip-172-31-82-192 kubelet[7822]: E0926 17:52:55.393747    7822 kuberuntime_>
Sep 26 17:52:55 ip-172-31-82-192 kubelet[7822]: E0926 17:52:55.393869    7822 kubelet.go:1>
lines 1-23/23 (END)
```

Now Run command **kubectl get nodes -o wide** we can see Status is ready.

```
ubuntu@ip-172-31-82-192:~$ kubectl get nodes -o wide
NAME              STATUS   ROLES           AGE     VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION   CONTAINER-RUNTIME
ip-172-31-82-192  Ready    control-plane   58m     v1.31.1   172.31.82.192   <none>        Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
ip-172-31-86-8    Ready    <none>          10m     v1.31.1   172.31.86.8     <none>        Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
ip-172-31-86-84   Ready    <none>          9m42s   v1.31.1   172.31.86.84    <none>        Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
ubuntu@ip-172-31-82-192:~$
```

Now to Rename run this command

kubectl label node <node-ip> kubernetes.io/role=worker

Rename to Node 1:**kubectl label node ip-172-31-86-8 kubernetes.io/role=Node1**

Rename to Node 2:**kubectl label node ip-172-31-86-84 kubernetes.io/role=Node2**

```
ubuntu@ip-172-31-82-192:~$ kubectl label node ip-172-31-86-8 kubernetes.io/role=Node1
^[[A^[[Anode/ip-172-31-86-8 labeled
ubuntu@ip-172-31-82-192:~$ kubectl label node ip-172-31-86-84 kubernetes.io/role=Node2
node/ip-172-31-86-84 labeled
ubuntu@ip-172-31-82-192:~$
```

Step 11: Run command **kubectl get nodes -o wide** . And Hence we can see we have Successfully connected Node 1 and Node 2 to the Master.

```
ubuntu@ip-172-31-82-192:~$ kubectl get nodes -o wide
NAME              STATUS   ROLES           AGE   VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION   CONTAINER-RUNTIME
ip-172-31-82-192  Ready    control-plane   83m   v1.31.1   172.31.82.192   <none>        Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
ip-172-31-86-8    Ready    Node1           35m   v1.31.1   172.31.86.8     <none>        Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
ip-172-31-86-84   Ready    Node2           34m   v1.31.1   172.31.86.84    <none>        Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
^[[Aubuntu@ip-172-31-82-192:~$ kubectl get nodes -o wide
```

Or run **kubectl get nodes**

```
^[[Aubuntu@ip-172-31-82-192kubectl get nodes
NAME              STATUS   ROLES           AGE   VERSION
ip-172-31-82-192  Ready    control-plane   85m   v1.31.1
ip-172-31-86-8    Ready    Node1           37m   v1.31.1
ip-172-31-86-84   Ready    Node2           36m   v1.31.1
ubuntu@ip-172-31-82-192:~$
```

Conclusion:

In this Advanced DevOps Lab experiment, we began by setting up three EC2 Ubuntu instances on AWS, designating one as the Master node and the others as Worker nodes. We then installed Docker and Kubernetes on all instances, ensuring Docker was properly configured. The Kubernetes cluster was initialized on the Master node, and the Flannel networking plugin was applied to facilitate communication between nodes. Finally, we joined the Worker nodes to the cluster using the provided token and hash, resulting in a fully operational Kubernetes cluster ready for managing and scaling containerized applications.

Experiment 4

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

Theory:
Container-based microservices architectures have revolutionized how development and operations teams test and deploy modern software. Containers allow companies to scale and deploy applications more efficiently, but they also introduce new challenges, adding complexity by creating a whole new infrastructure ecosystem.
Today, both large and small software companies are deploying thousands of container instances daily. Managing this level of complexity at scale requires advanced tools. Enter Kubernetes.

Originally developed by Google, Kubernetes is an open-source container orchestration platform designed to automate the deployment, scaling, and management of containerized applications. Kubernetes has quickly become the de facto standard for container orchestration and is the flagship project of the Cloud Native Computing Foundation (CNCF), supported by major players like Google, AWS, Microsoft, IBM, Intel, Cisco, and Red Hat. Kubernetes simplifies the deployment and operation of applications in a microservice architecture by providing an abstraction layer over a group of hosts. This allows development teams to deploy their applications while Kubernetes takes care of key tasks, including:

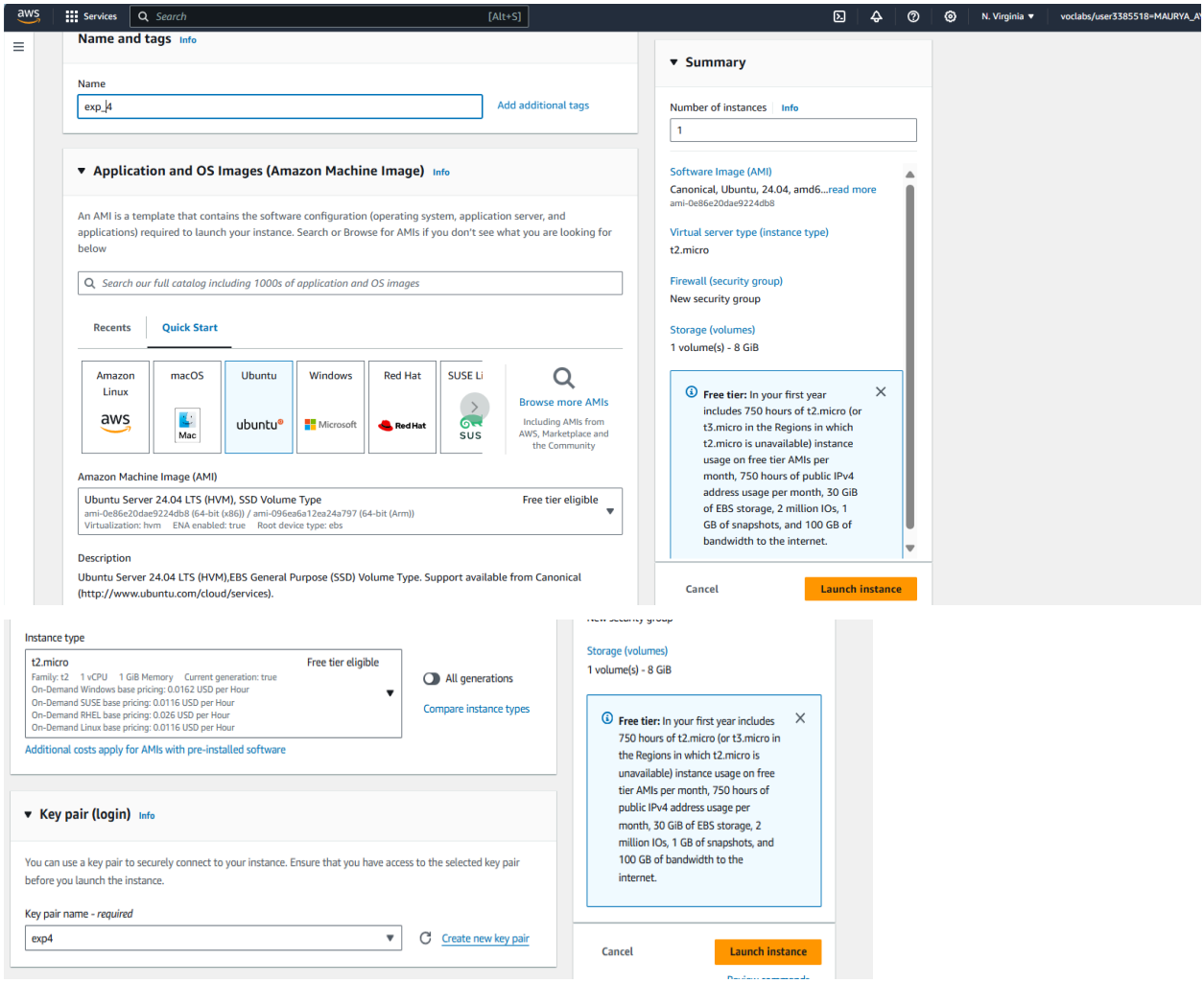
- Managing resource consumption by applications or teams
- Distributing application load evenly across the infrastructure
- Automatically load balancing requests across multiple instances of an application
- Monitoring resource usage to prevent applications from exceeding resource limits and automatically restarting them if needed
- Moving application instances between hosts when resources are low or if a host fails
- Automatically utilizing additional resources when new hosts are added to the cluster
- Facilitating canary deployments and rollbacks with ease

Necessary Requirements:
- EC2 Instance: The experiment required launching a t2.medium EC2 instance with 2 CPUs, as Kubernetes demands sufficient resources for effective functioning.

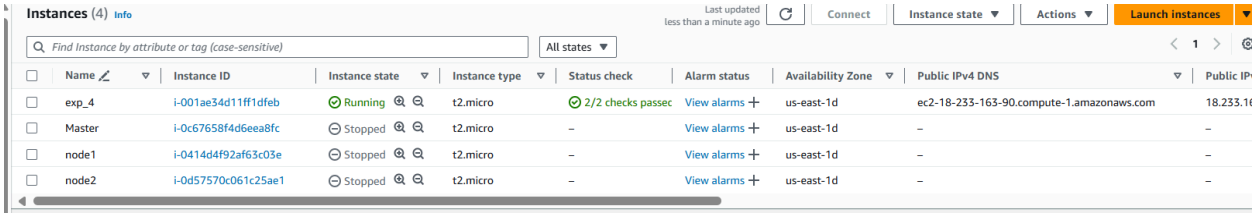
- **Minimum Requirements:**
  - Instance Type: t2.medium
  - CPUs: 2
  - Memory: Adequate for container orchestration.

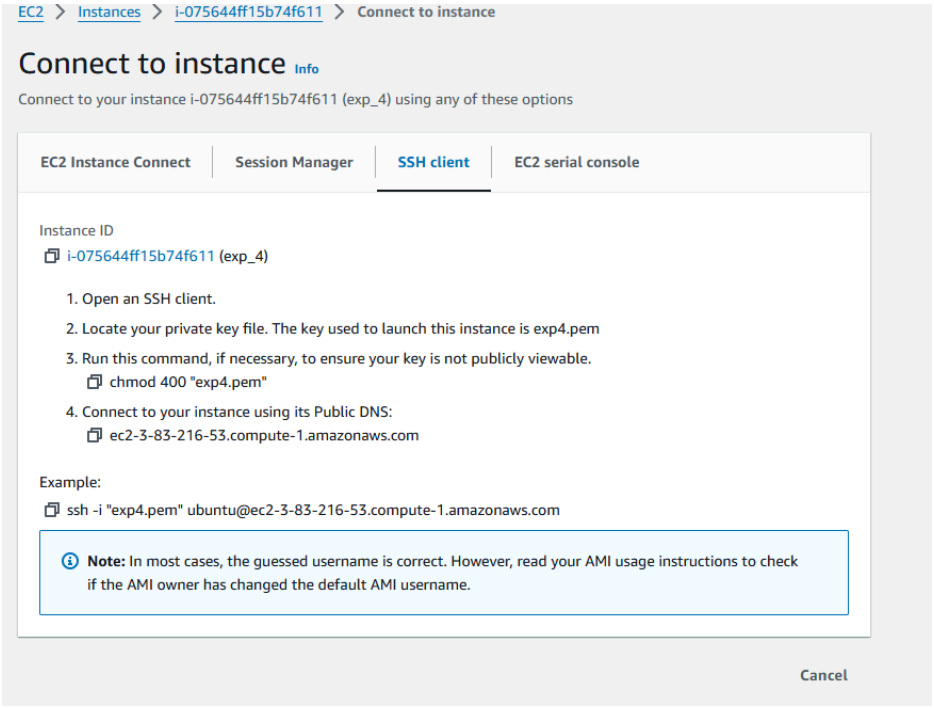
This ensured that the Kubernetes cluster had the necessary resources to function smoothly

Step 1: Log in to your AWS Academy/personal account and launch 3 new Ec2 Instances. Select Ubuntu as AMI and t2.micro (because in academic account only t2.micro is present) as Instance Type and create a key of type RSA with .pem extension and move the downloaded key to the new folder.



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





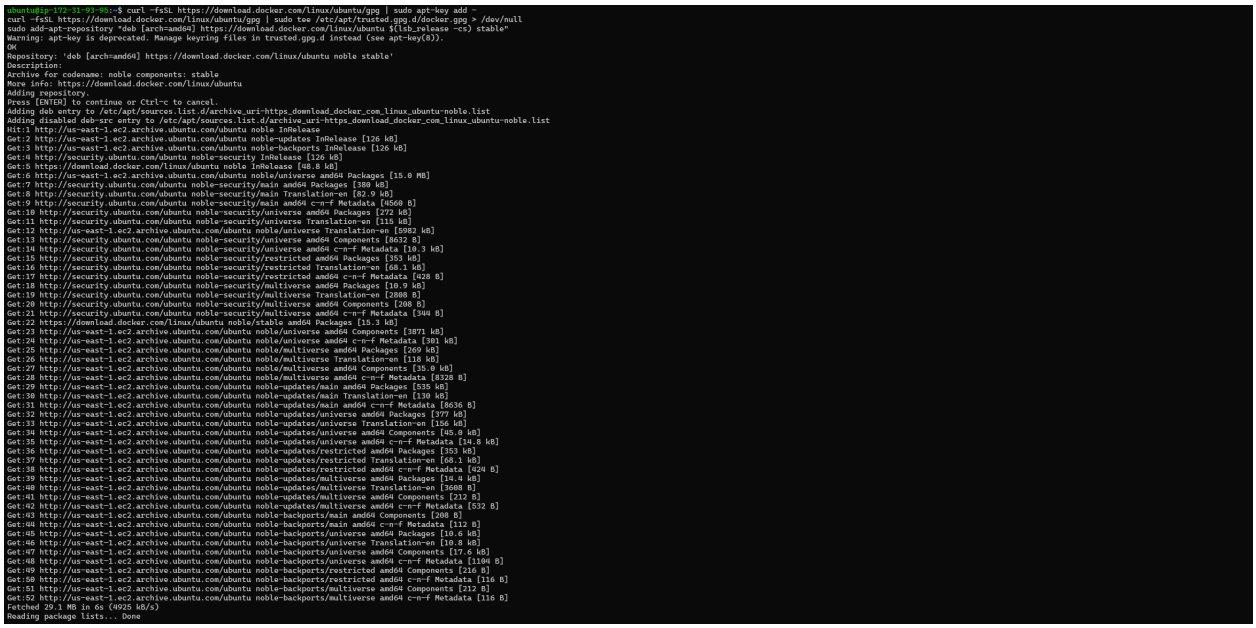
Step 3: Now open the folder in the terminal where our .pem key is stored and paste the Example command (starting with ssh -i .....) in the terminal.
ssh -i "<PATH TO FILE>exp3.pem" ubuntu@ec2-3-83-216-53.compute-1.amazonaws.com

```
C:\Users\Ayush Maurya>ssh -i "Downloads/exp4.pem" ubuntu@ec2-3-83-216-53.compute-1.amazonaws.com
The authenticity of host 'ec2-3-83-216-53.compute-1.amazonaws.com (3.83.216.53)' can't be established.
ED25519 key fingerprint is SHA256:jdAspK3Zoikd8XhOvy+g6Ea5WqRKlgbfnr5S66tYTRg.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-83-216-53.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

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

 System information as of Fri Sep 27 04:30:29 UTC 2024

  System load:  0.03              Processes:             105
  Usage of /:   22.8% of 6.71GB   Users logged in:       0
  Memory usage: 20%               IPv4 address for enX0: 172.31.93.95
```

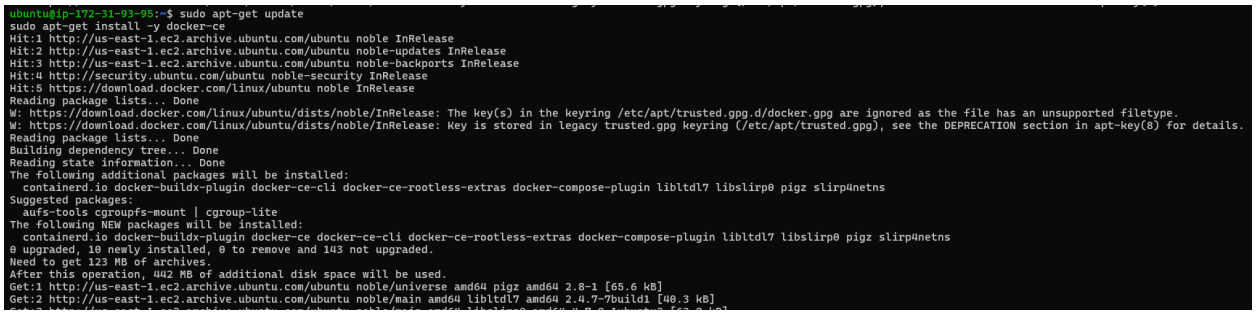
```
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-93-95:~$
```

Step 4: Run below commands to install and setup Docker

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



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



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

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

```
ubuntu@ip-172-31-93-95:~$ sudo mkdir -p /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
ubuntu@ip-172-31-93-95:~$
```

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

```
ubuntu@ip-172-31-93-95:~$ 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
ubuntu@ip-172-31-93-95:~$ A
```

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-172-31-93-95:~$ 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
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
ubuntu@ip-172-31-93-95:~$
```

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

```
ubuntu@ip-172-31-93-95:~$ 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  InRelease [1186 B]
Get:7 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  Packages [4865 B]
Fetched 6051 B in 1s (6278 B/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 an unsupporte
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 section in ap
key(8) for details.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

```
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.
ubuntu@ip-172-31-93-95:~$
```

sudo systemctl enable --now kubelet
sudo apt-get install -y containerd

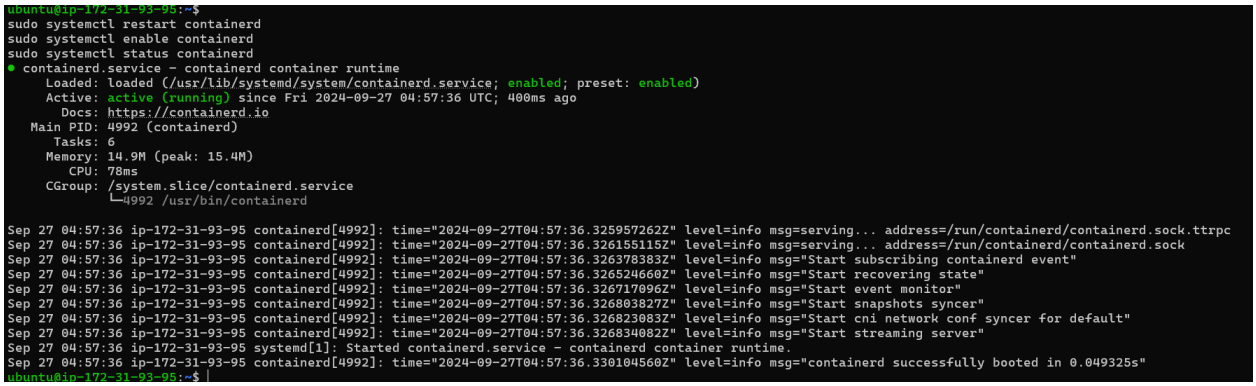
```
ubuntu@ip-172-31-93-95:~$ sudo systemctl enable --now kubelet
sudo apt-get install -y containerd
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  docker-buildx-plugin 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 143 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 (47.0 MB/s)
(Reading database ... 68064 files and directories currently installed.)
Removing docker-ce (5:27.3.1-1~ubuntu.24.04~noble) ...
Removing containerd.io (1.7.22-1) ...
Selecting previously unselected package runc.
(Reading database ... 68044 files and directories currently installed.)
Preparing to unpack .../runc_1.1.12-0ubuntu3.1_amd64.deb ...
Unpacking runc (1.1.12-0ubuntu3.1) ...
```

```
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.
ubuntu@ip-172-31-93-95:~$
```

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

```
ubuntu@ip-172-31-93-95:~$ 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
```

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



```
ubuntu@ip-172-31-93-95:~$
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 Fri 2024-09-27 04:57:36 UTC; 400ms ago
       Docs: https://containerd.io
   Main PID: 4992 (containerd)
      Tasks: 6
     Memory: 14.9M (peak: 15.4M)
        CPU: 78ms
     CGroup: /system.slice/containerd.service
             └─4992 /usr/bin/containerd

Sep 27 04:57:36 ip-172-31-93-95 containerd[4992]: time="2024-09-27T04:57:36.325957262Z" level=info msg=serving... address=/run/containerd/containerd.sock.ttrpc
Sep 27 04:57:36 ip-172-31-93-95 containerd[4992]: time="2024-09-27T04:57:36.326155115Z" level=info msg=serving... address=/run/containerd/containerd.sock
Sep 27 04:57:36 ip-172-31-93-95 containerd[4992]: time="2024-09-27T04:57:36.326378383Z" level=info msg="Start subscribing containerd event"
Sep 27 04:57:36 ip-172-31-93-95 containerd[4992]: time="2024-09-27T04:57:36.326524660Z" level=info msg="Start recovering state"
Sep 27 04:57:36 ip-172-31-93-95 containerd[4992]: time="2024-09-27T04:57:36.326717096Z" level=info msg="Start event monitor"
Sep 27 04:57:36 ip-172-31-93-95 containerd[4992]: time="2024-09-27T04:57:36.326803827Z" level=info msg="Start snapshots syncer"
Sep 27 04:57:36 ip-172-31-93-95 containerd[4992]: time="2024-09-27T04:57:36.326823083Z" level=info msg="Start cni network conf syncer for default"
Sep 27 04:57:36 ip-172-31-93-95 containerd[4992]: time="2024-09-27T04:57:36.326834082Z" level=info msg="Start streaming server"
Sep 27 04:57:36 ip-172-31-93-95 systemd[1]: Started containerd.service - containerd container runtime.
Sep 27 04:57:36 ip-172-31-93-95 containerd[4992]: time="2024-09-27T04:57:36.330104560Z" level=info msg="containerd successfully booted in 0.049325s"
ubuntu@ip-172-31-93-95:~$
```

sudo apt-get install -y socat

```
ubuntu@ip-172-31-93-95:~$ 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 libltdl7 libslirp0 pigz slirp4netns
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  socat
0 upgraded, 1 newly installed, 0 to remove and 143 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 1.8.0.0-4build3 [374 kB]
Fetched 374 kB in 0s (11.9 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.
ubuntu@ip-172-31-93-95:~$
```

Step 6: Initialize the Kubercluster.

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

```
ubuntu@ip-172-31-93-95:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errors=NumCPU,Mem
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
        [WARNING NumCPU]: the number of available CPUs 1 is less than the required 2
        [WARNING Mem]: the system RAM (957 MB) is less than the minimum 1700 MB
[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'
W0927 05:00:10.999529    5254 checks.go:846] detected that the sandbox image "registry.k8s.io/pause:3.8" of the container runtime is inconsistent with that used by kubead
m.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-172-31-93-95 kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local] and IPs
 [10.96.0.1 172.31.93.95]
[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-172-31-93-95 localhost] and IPs [172.31.93.95 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
```

```
  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 172.31.93.95:6443 --token wuhiw8.tqn7cnmejhk5kqey \
        --discovery-token-ca-cert-hash sha256:f9a9d75f6d99fdd71aaf1b049f75e5ece76e902877e420624f4a305cf4125eb7
ubuntu@ip-172-31-93-95:~$
```

TOKEN

kubeadm join 172.31.93.95:6443 --token wuhiw8.tqn7cnmejhk5kqey \
--discovery-token-ca-cert-hash sha256:f9a9d75f6d99fdd71aaf1b049f75e5ece76e902877e420624f4a305cf4125eb7

From this command we get token and ca-

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-172-31-93-95:~$ mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
ubuntu@ip-172-31-93-95:~$
```

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-172-31-93-95:~$ 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-172-31-93-95:~$
```

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-172-31-93-95:~$ kubectl apply -f https://k8s.io/examples/application/deployment.yaml
deployment.apps/nginx-deployment created
ubuntu@ip-172-31-93-95:~$
```

kubectl get pods

```
ubuntu@ip-172-31-93-95:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-wpb7n    0/1     Pending   0          30s
nginx-deployment-d556bf558-wvkzl    0/1     Pending   0          30s
ubuntu@ip-172-31-93-95:~$
```

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

```
ubuntu@ip-172-31-93-95:~$ POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
kubectl port-forward $POD_NAME 8080:80
error: unable to forward port because pod is not running. Current status=Pending
ubuntu@ip-172-31-93-95:~$ POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
ubuntu@ip-172-31-93-95:~$ kubectl port-forward $POD_NAME 8080:80
error: unable to forward port because pod is not running. Current status=Pending
ubuntu@ip-172-31-93-95:~$
```

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-172-31-93-95:~$ kubectl taint nodes --all node-role.kubernetes.io/control-plane-node/ip-172-31-20-171 untainted
kubectl get nodes
error: at least one taint update is required
NAME              STATUS   ROLES           AGE   VERSION
ip-172-31-93-95   Ready    control-plane   20m   v1.31.1
ubuntu@ip-172-31-93-95:~$
```

kubectl get pods

```
ubuntu@ip-172-31-93-95:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-wpb7n    0/1     Pending   0          19m
nginx-deployment-d556bf558-wvkzl    0/1     Pending   0          19m
ubuntu@ip-172-31-93-95:~$
```

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

```
ubuntu@ip-172-31-93-95:~$ 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
```

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

```
ubuntu@ip-172-31-93-95:~$ curl --head http://127.0.0.1:8080
HTTP/1.1 200 OK
Server: nginx/1.14.2
Date: Fri, 27 Sep 2024 06:06:41 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

ubuntu@ip-172-31-93-95:~$
```

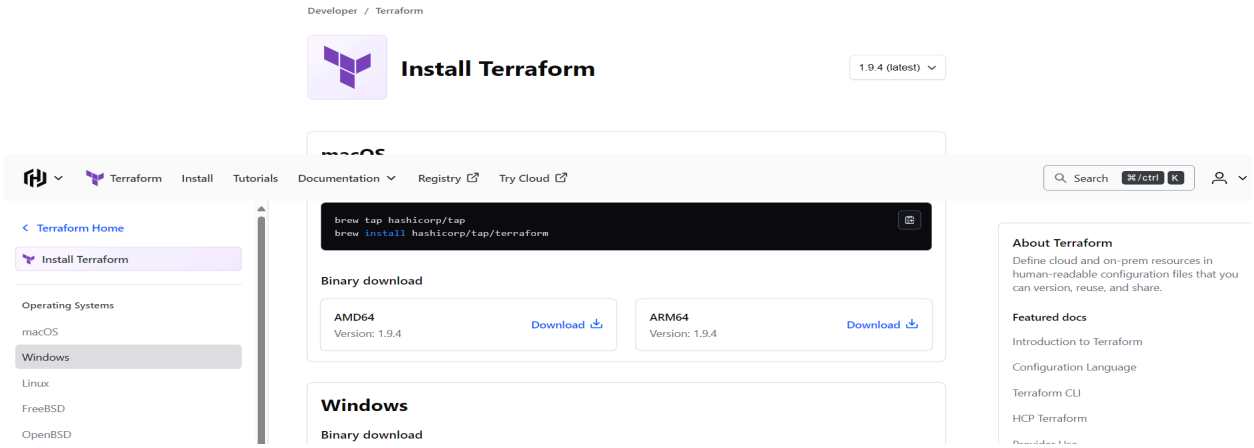
If the response is 200 OK and you can see the Nginx server name, your deployment was successful.
We have successfully deployed our Nginx server on our EC2 instance.

Conclusion: We successfully set up a Kubernetes cluster on AWS EC2, addressing issues related to component setup and residual configurations. We ensured proper cleanup of previous Kubernetes files and mounts, verified the kubelet service, and applied Flannel for networking. Finally, we resolved connectivity issues, and after a thorough review of logs and configuration, we deployed and exposed an NGINX server using Kubernetes services, preparing the cluster for efficient traffic management and scaling.

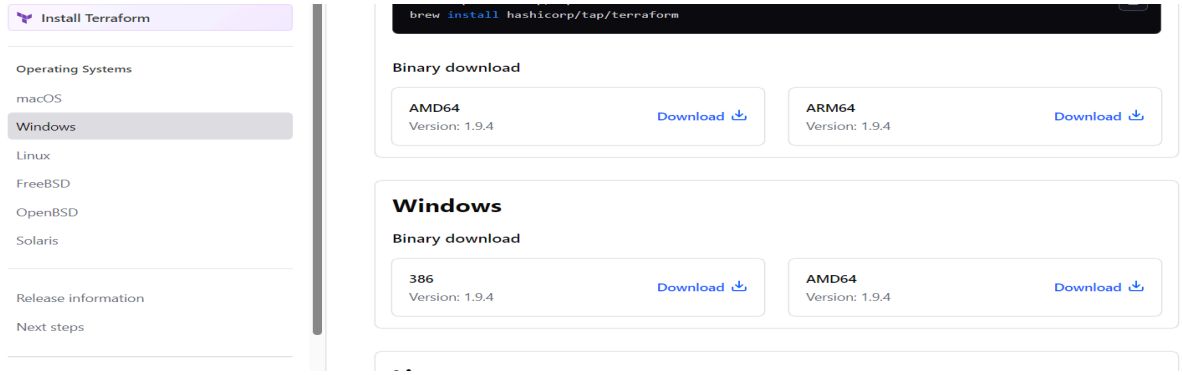
Experiment 5

Step 1: Download terraform

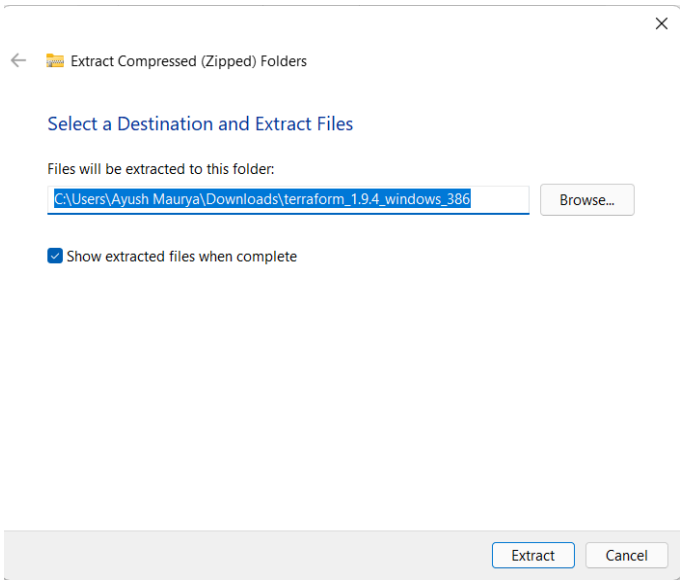
website:https://www.terraform.io/downloads.html



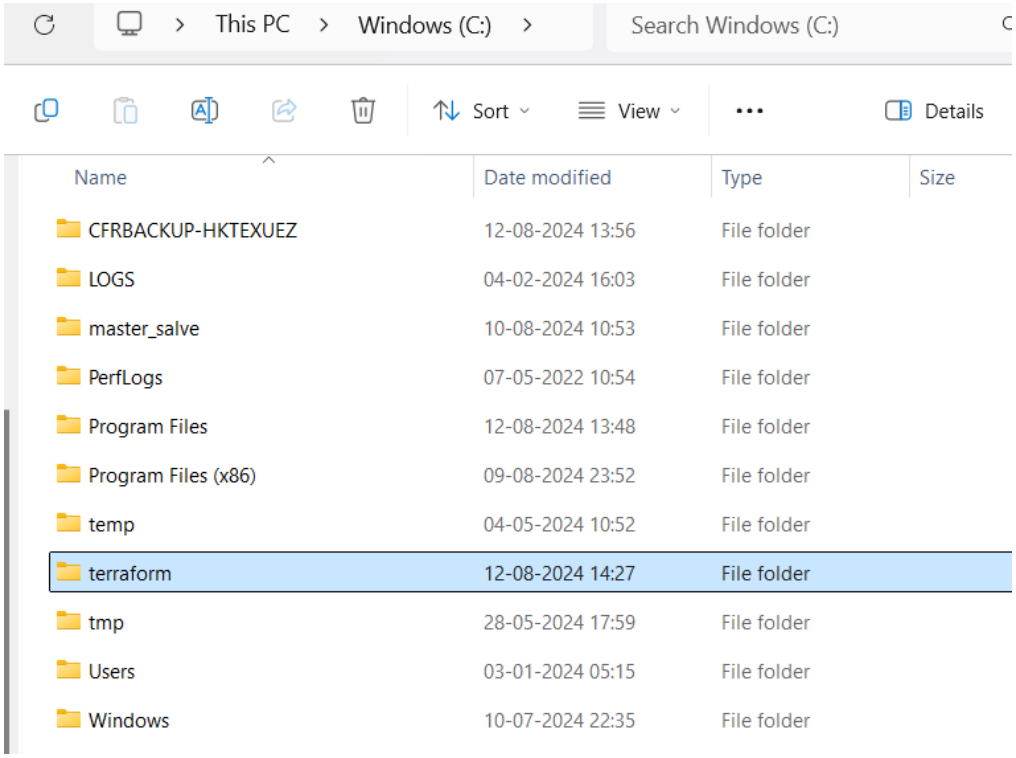
Step 2: Downlaod Windows Binary download 386



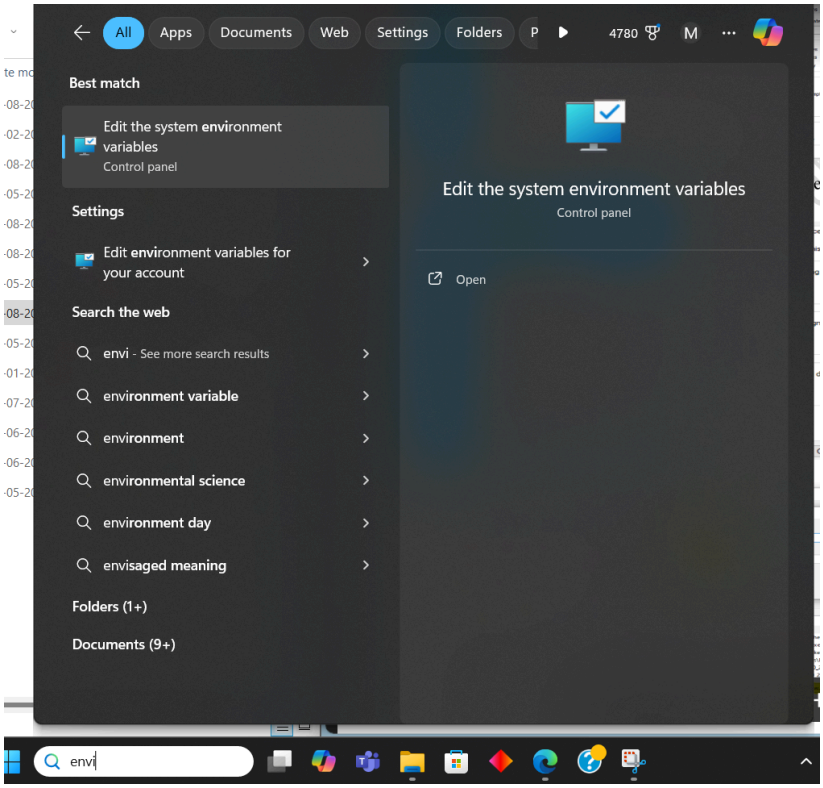
Step 3: Extract the downloaded setup file Terraform.exe in C:\Terraform directory



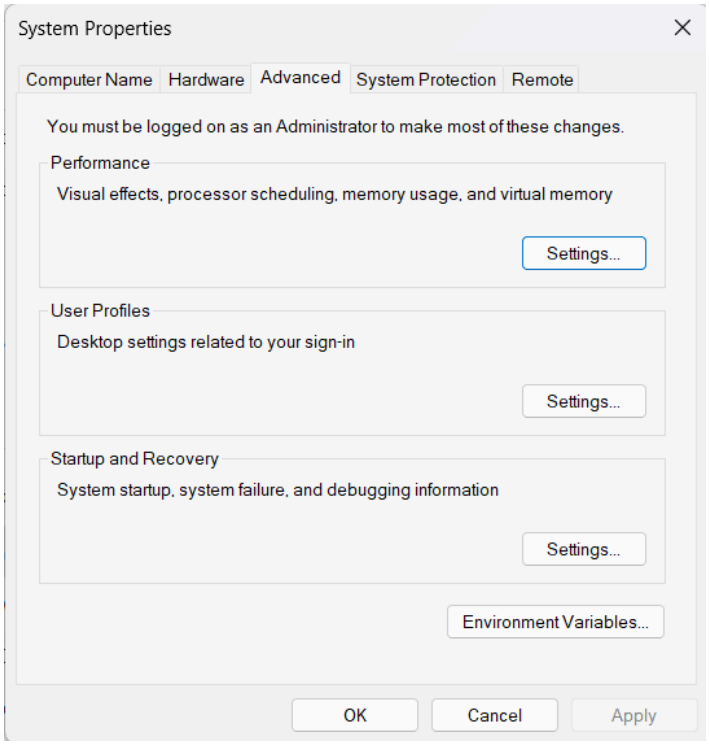
Step 4: Created folder in C: drive for terraform,



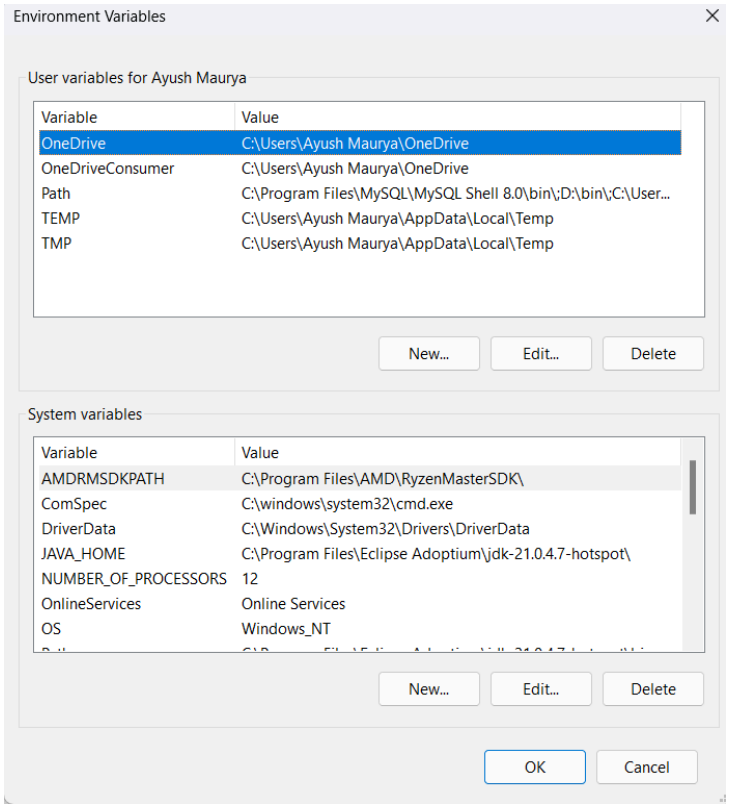
Step 5: Set the System path for Terraform in Environment Variable. Open Environment Variable.



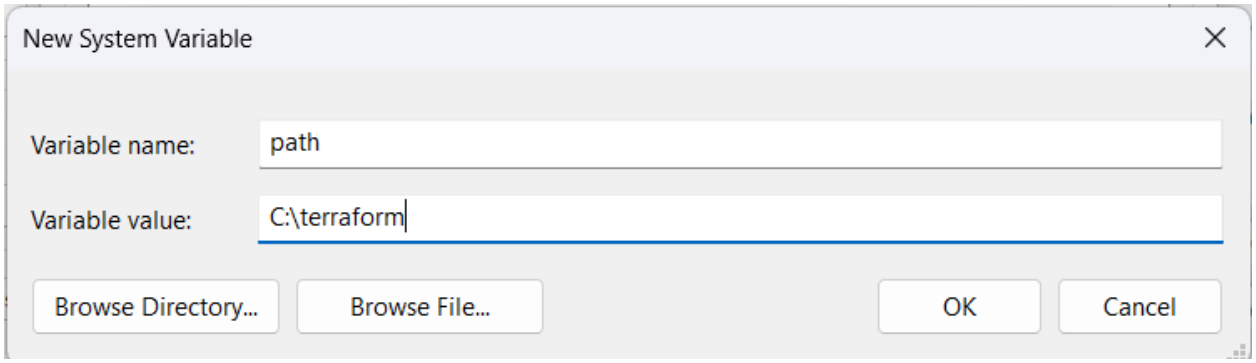
Step 6: System properties will open. Now click on Environment Variables On bottom.



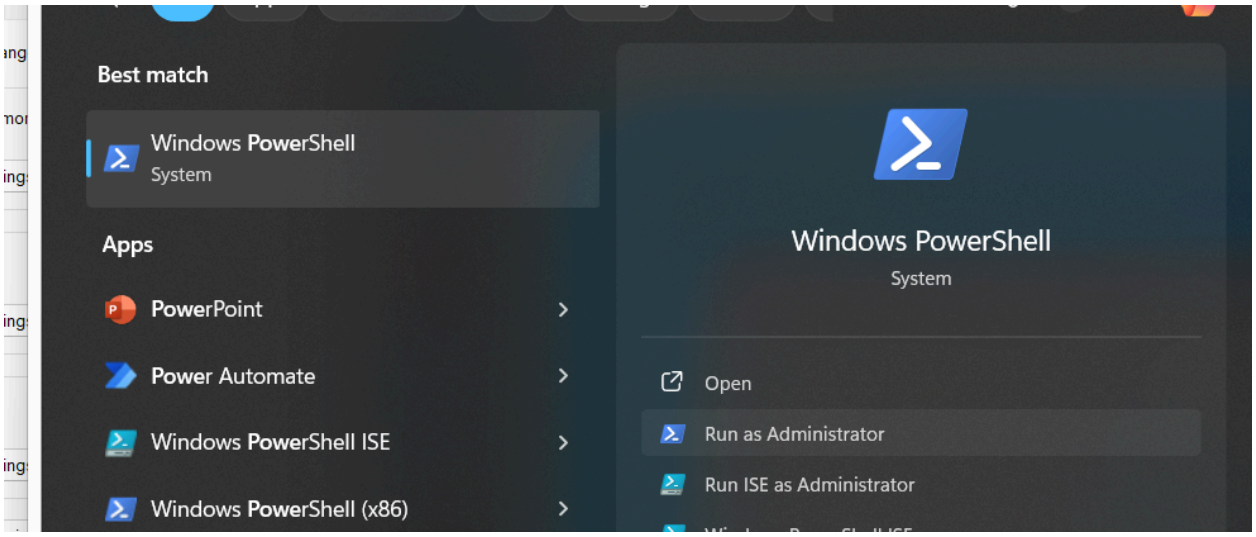
Step 7: Click on ‘New’ to create new system variables.



Step 8: Give variable name as ‘path’ or any and variable value as path of folder in C: drive where terraform is extracted.



Step 9: Open PowerShell with Admin Access



Step 10: Open Terraform in PowerShell and check its functionality

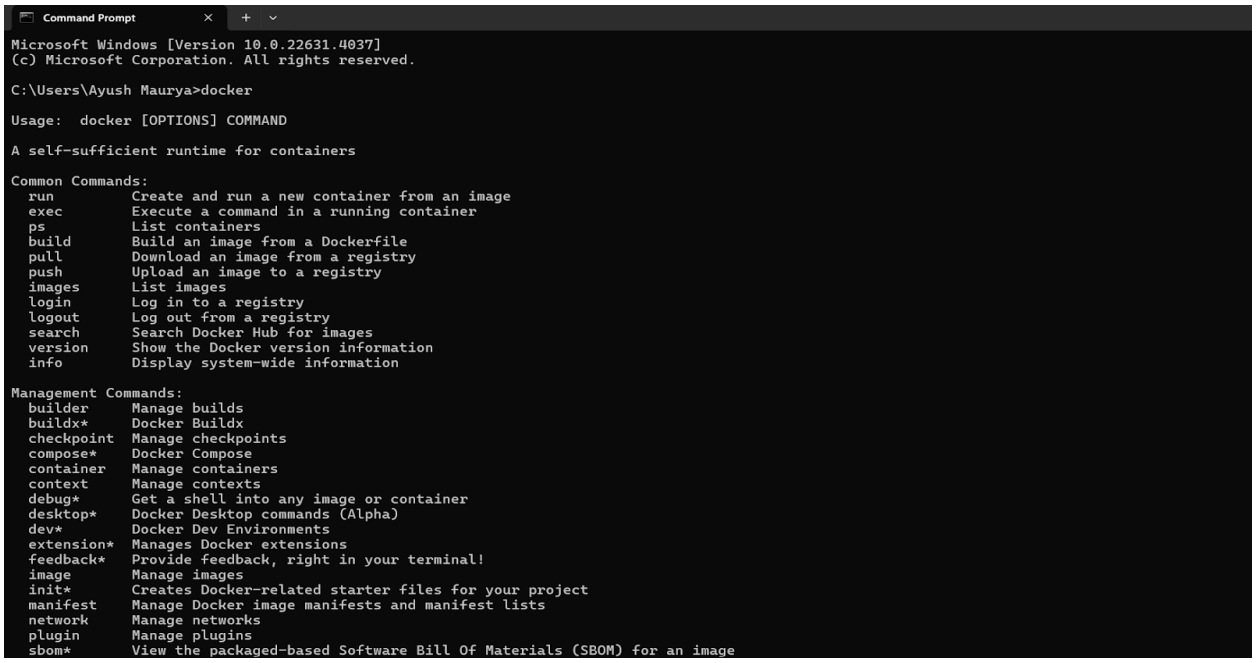
```
PS C:\windows\system32> terraform
Usage: terraform [global options] <subcommand> [args]

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

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

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

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

Experiment 6

A. Creating docker image using terraform
Prerequisite:

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

Step 1: Check the docker functionality



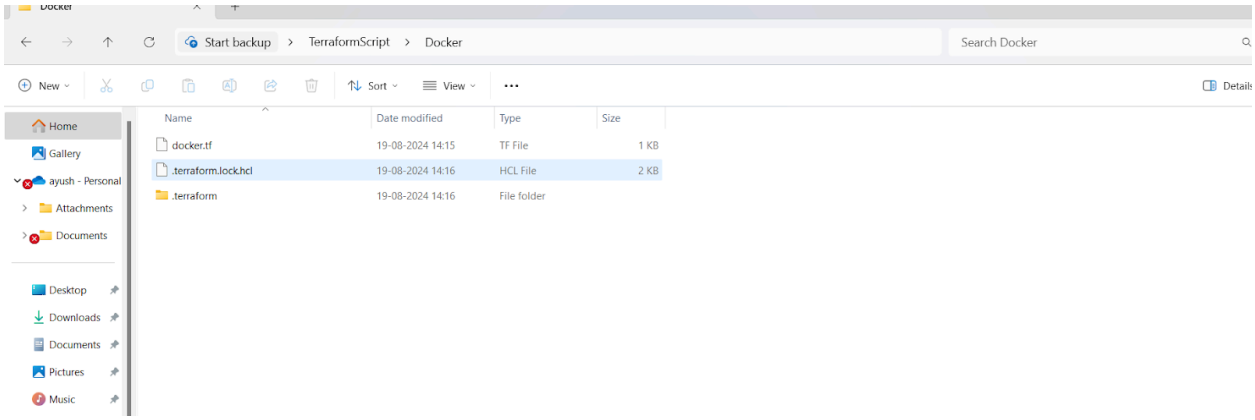
Now, create a folder named ‘Terraform Scripts’ in which we save our different types of scripts which will be further used in this experiment.

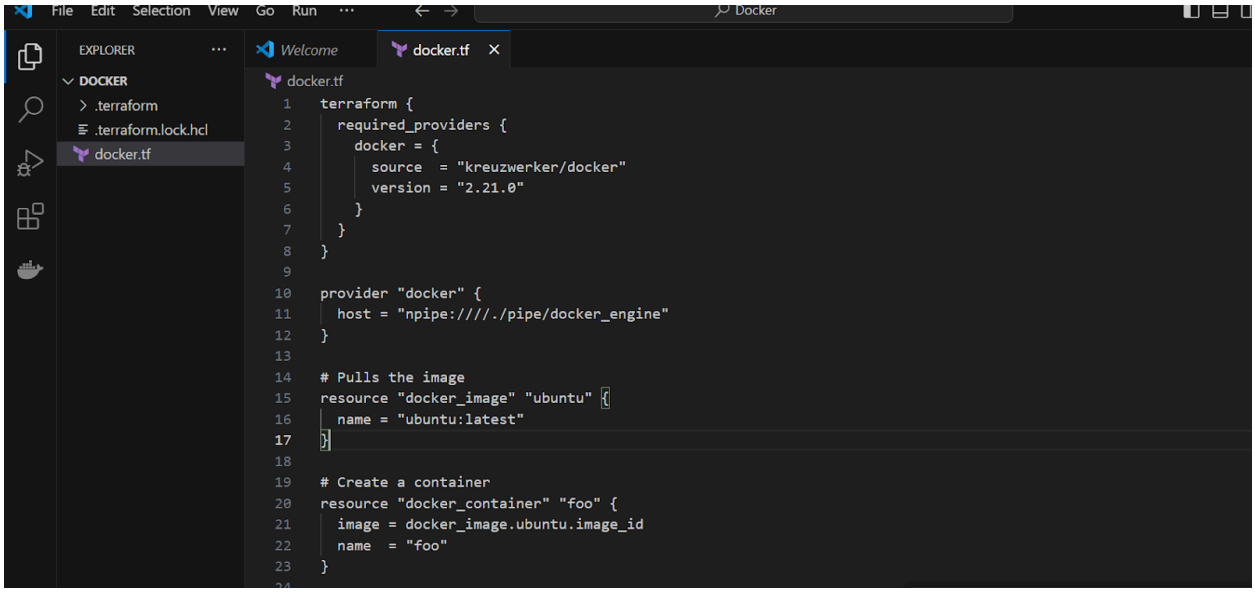
Step 2: Firstly create a new folder named ‘Docker’ in the ‘TerraformScripts’ folder. Then create a new docker.tf file using Atom editor and write the followingcontents into it to create a Ubuntu Linux container.

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

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

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

Terraform has been successfully initialized!

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

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

Step 4: Execute Terraform plan to see the available resources

```
PS C:\Users\Ayush Maurya\Desktop\TerraformScript\Docker> terraform plan

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

Terraform will perform the following actions:

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

```
      + start           = true
      + stdin_open      = false
      + stop_signal     = (known after apply)
      + stop_timeout    = (known after apply)
      + tty             = false

      + healthcheck (known after apply)

      + labels (known after apply)
    }

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

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

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

```
PS C:\Users\Ayush Maurya\Desktop\TerraformScript\Docker> terraform apply

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

Terraform will perform the following actions:

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

```
      + stdin_open       = false
      + stop_signal      = (known after apply)
      + stop_timeout     = (known after apply)
      + tty              = false

      + healthcheck (known after apply)

      + labels (known after apply)
    }

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

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

  Enter a value: yes

docker_container.foo: Creating...
docker_container.foo: Creation complete after 1s [id=25e618f8f29715a205c36e0d31bfdd4a326dc83f513f2c7779aebd492ce9f602]

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

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:

```
Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
PS C:\Users\Ayush Maurya\Desktop\TerraformScript\Docker> docker images
REPOSITORY    TAG       IMAGE ID       CREATED       SIZE
react-img     latest    d8b8903ee063   8 days ago    320MB
ubuntu        latest    edbfe74c41f8   2 weeks ago   78.1MB
```

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

```
PS C:\Users\Ayush Maurya\Desktop\TerraformScript\Docker> terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Refreshing state... [id=601782417e17994f75628bb648b3e4cbbb989037e3a3d302303c4a675a39599b]

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

Terraform will perform the following actions:

  # docker_container.foo will be destroyed
  - resource "docker_container" "foo" {
      - attach            = false -> null
      - command           = [
          - "sh",
          - "-c",
          - "while true; do sleep 1; done",
        ] -> null
      - cpu_shares        = 0 -> null
      - dns               = [] -> null
      - dns_opts          = [] -> null
      - dns_search        = [] -> null
      - entrypoint        = [] -> null
      - env               = [] -> null
      - gateway           = "172.17.0.1" -> null
      - group_add         = [] -> null
      - hostname          = "601782417e17" -> null
      - id                = "601782417e17994f75628bb648b3e4cbbb989037e3a3d302303c4a675a39599b" -> null
      - image             = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a" -> null
      - init              = false -> null
      - ip_address        = "172.17.0.2" -> null
      - ip_prefix_length  = 16 -> null
      - ipc_mode          = "private" -> null
      - links             = [] -> null
      - log_driver        = "json-file" -> null
      - log_opts          = {} -> null
      - logs              = false -> null
      - max_retry_count   = 0 -> null
      - memory            = 0 -> null
      - memory_swap       = 0 -> null
      - must_run          = true -> null
```

```
      - rm               = false -> null
      - runtime          = "runc" -> null
      - security_opts    = [] -> null
      - shm_size         = 64 -> null
      - start            = true -> null
      - stdin_open       = false -> null
      - stop_timeout     = 0 -> null
      - storage_opts     = {} -> null
      - sysctls          = {} -> null
      - tmpfs            = {} -> null
      - tty              = false -> null
        # (8 unchanged attributes hidden)
    }

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

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

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

  Enter a value: yes

docker_container.foo: Destroying... [id=601782417e17994f75628bb648b3e4cbbb989037e3a3d302303c4a675a39599b]
docker_container.foo: Destruction complete after 0s
docker_image.ubuntu: Destroying... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_image.ubuntu: Destruction complete after 1s

Destroy complete! Resources: 2 destroyed.
PS C:\Users\Ayush Maurya\Desktop\TerraformScript\Docker>
```

Docker images After Executing Destroy step

```
Destroy complete! Resources: 2 destroyed.
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>
```

Experiment 7

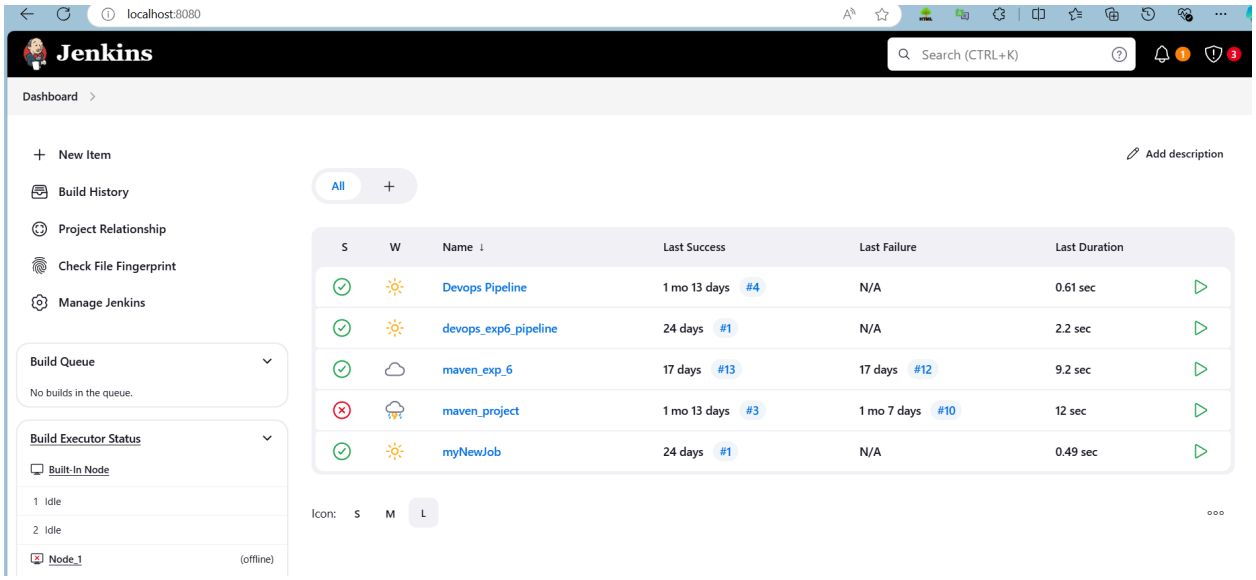
Aim: To understand Static Analysis SAST process and learn to integrate Jenkins SAST to SonarQube/GitLab.

Integrating Jenkins with SonarQube:

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

Steps to integrate Jenkins with SonarQube

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

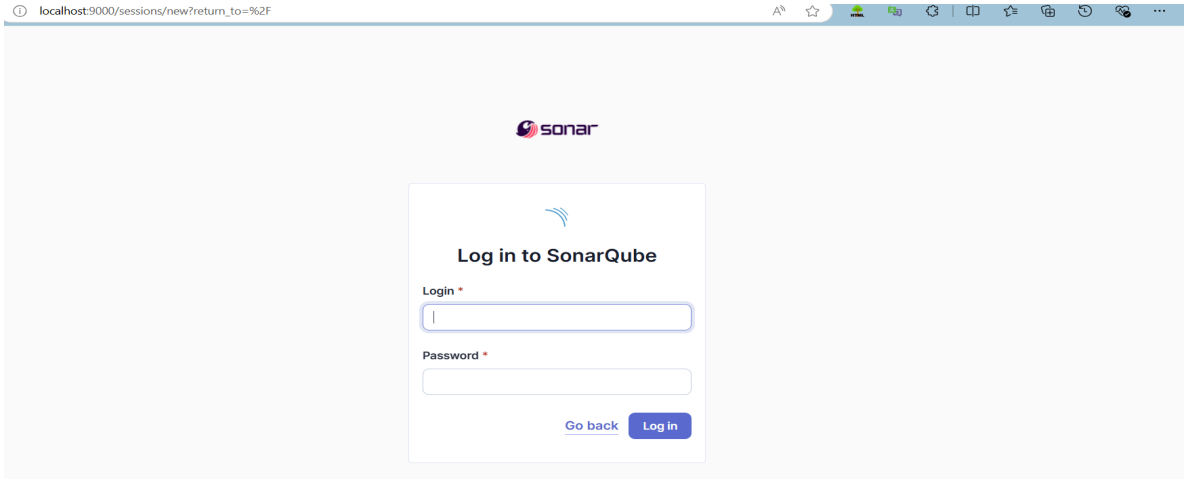


2. Run SonarQube in a Docker container using this command -
docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest

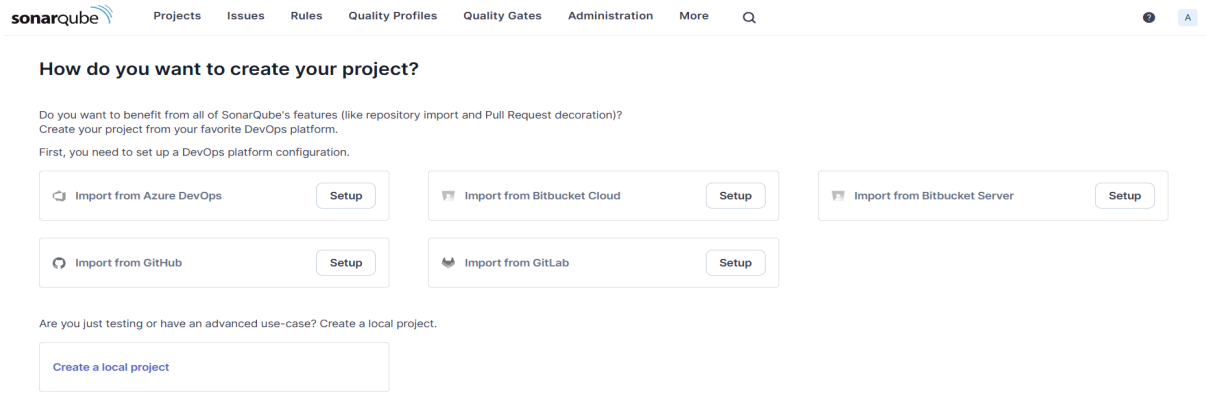
----------Warning: run below command only once

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

PS C:\Users\Ayush Maurya> docker run -d --name sonarqube -e SINAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
Unable to find image 'sonarqube:latest' locally
latest: Pulling from library/sonarqube
762bedf4b1b7: Pull complete
95f9bd9906fa: Pull complete
a32d681e6b99: Pull complete
aabdd0a18314: Pull complete
5161e45ecd8d: Pull complete
aeb0020dfa06: Pull complete
01548d361aea: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:bb444c58c1e04d8a147a3bb12af941c57e0100a5b21d10e599384d59bed36c86
Status: Downloaded newer image for sonarqube:latest
4af48468290f95b22362652ee37b96c935b0bed754945c62cf3b0d5d51a2ac0c
PS C:\Users\Ayush Maurya>
```

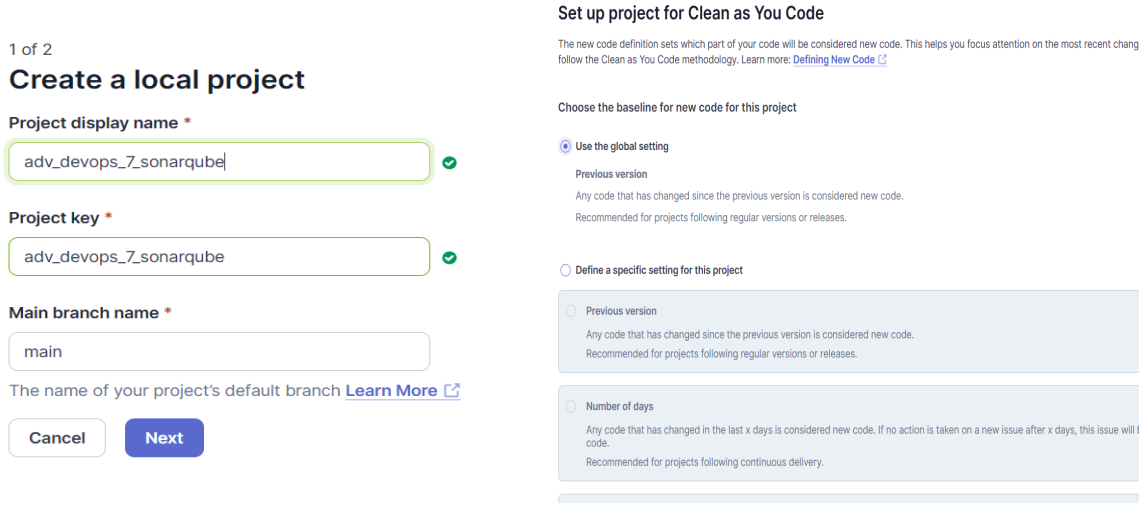
3. Once the container is up and running, you can check the status of SonarQube at localhost port 9000.



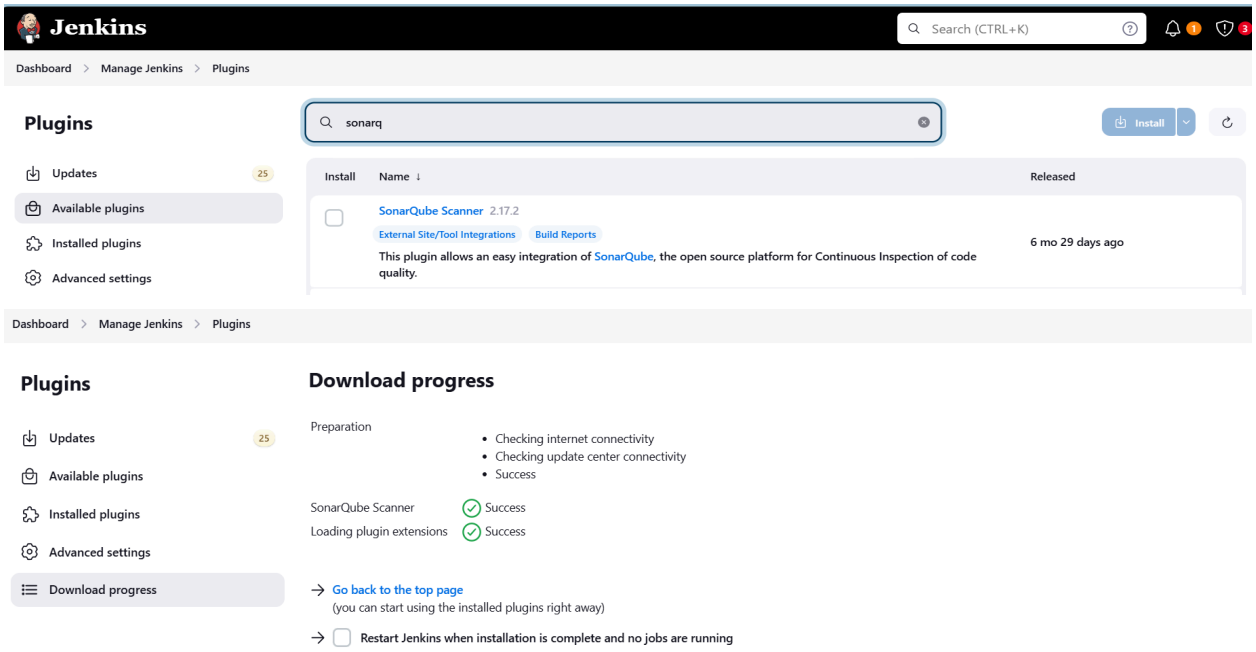
4. Login to SonarQube using username admin and password admin.



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

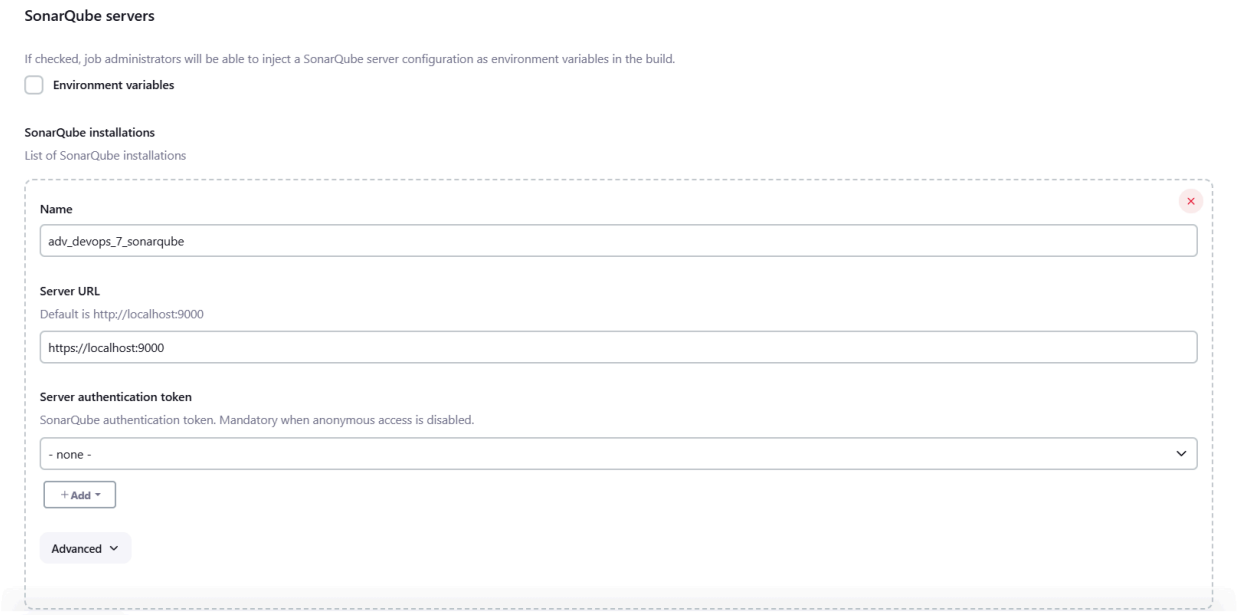


Setup the project and come back to Jenkins Dashboard.
Go to Manage Jenkins and search for SonarQube Scanner for Jenkins and install it.



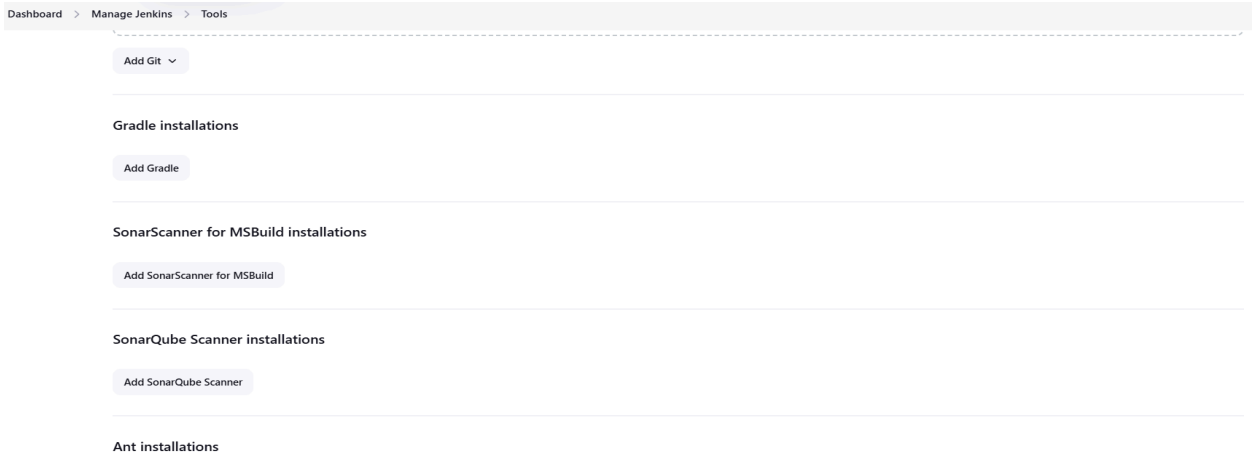
6. Under Jenkins ‘Manage Jenkins’ then go to ‘system’, scroll and look for **SonarQube Servers** and enter the details.
Enter the Server Authentication token if needed.

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

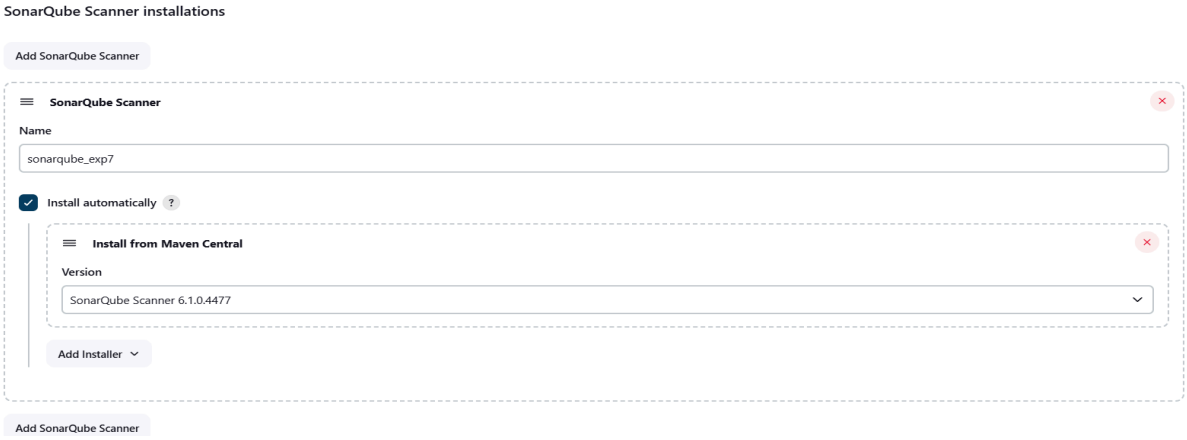


7. Search for SonarQube Scanner under Global Tool Configuration. Choose the latest configuration and choose Install automatically.

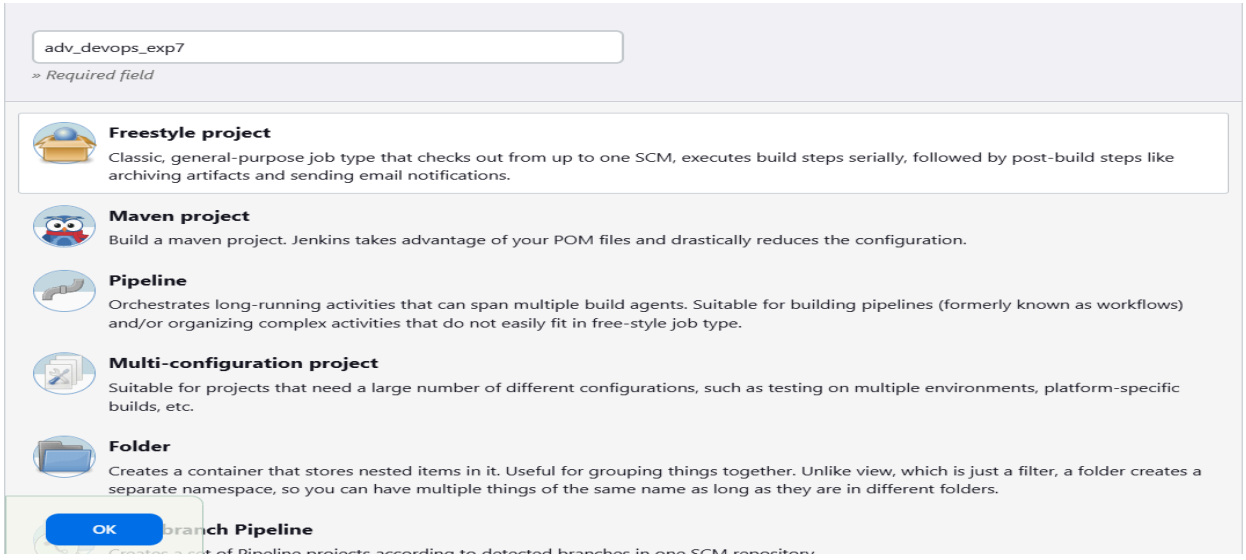
Dashboard > Manage Jenkins > Tools



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

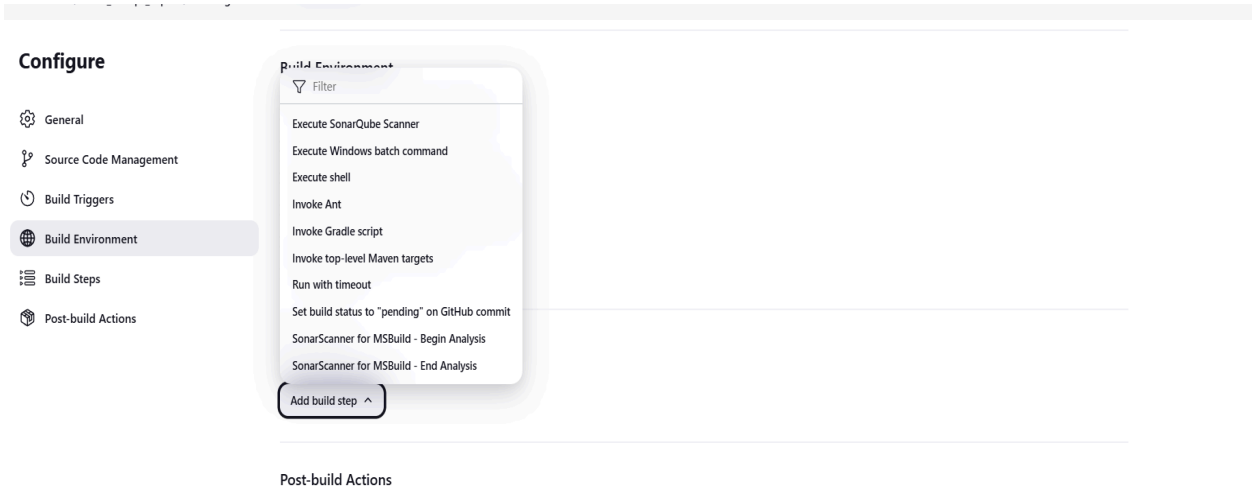


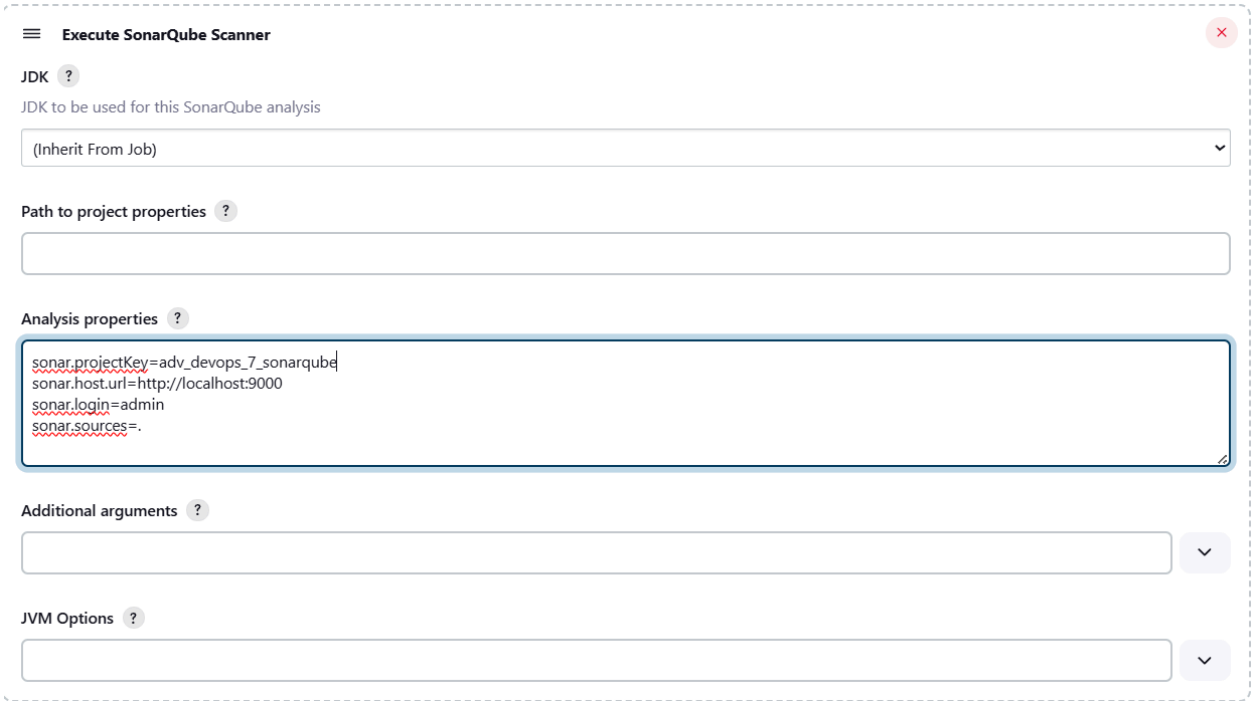
8. After the configuration, create a New Item in Jenkins, choose a freestyle project.



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

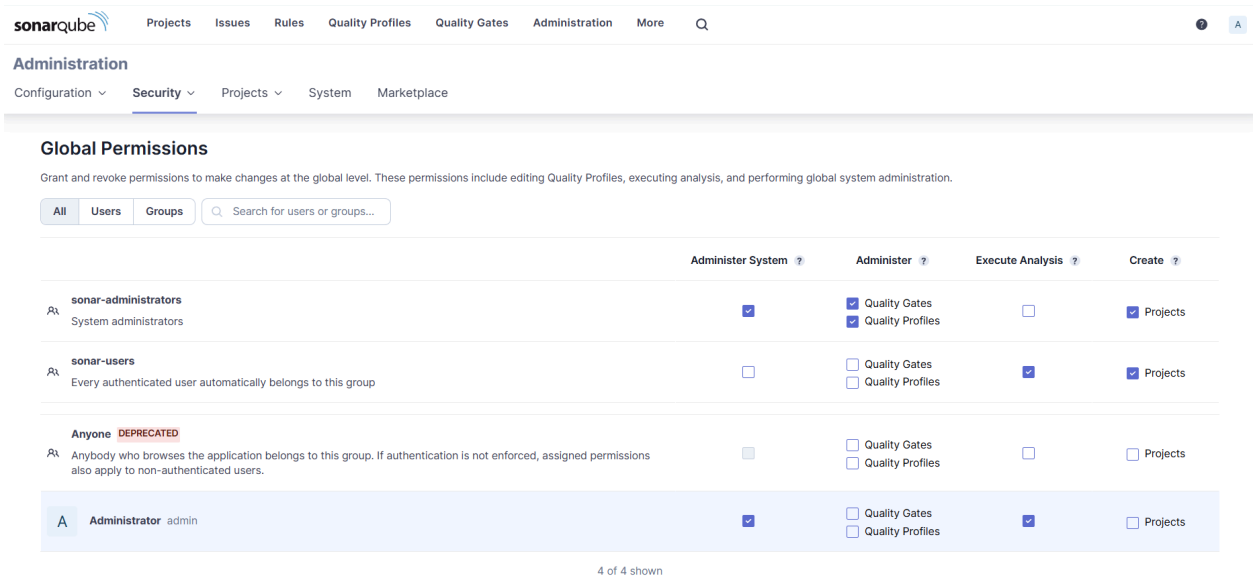
10. Under **Select project → Configuration → Build steps → Execute SonarQube Scanner**, enter these Analysis properties. Mention the SonarQube Project Key, Login, Password, Source path and Host URL.





Then save

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



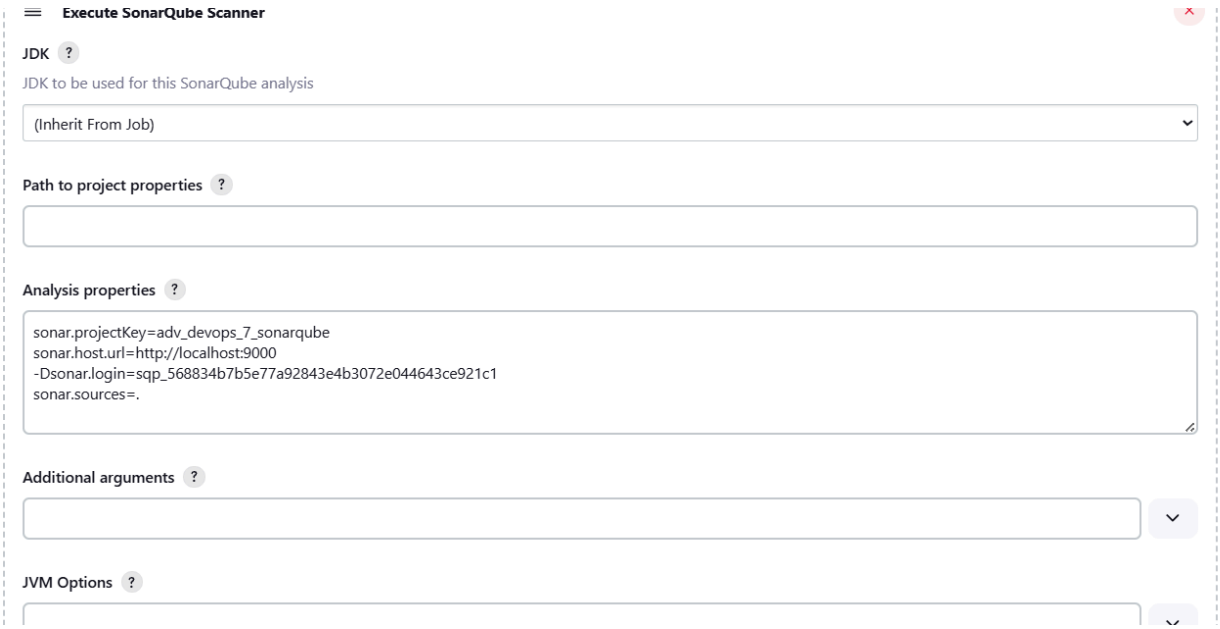
IF CONSOLE OUTPUT FAILED:

Step 1: Generate a New Authentication Token in SonarQube

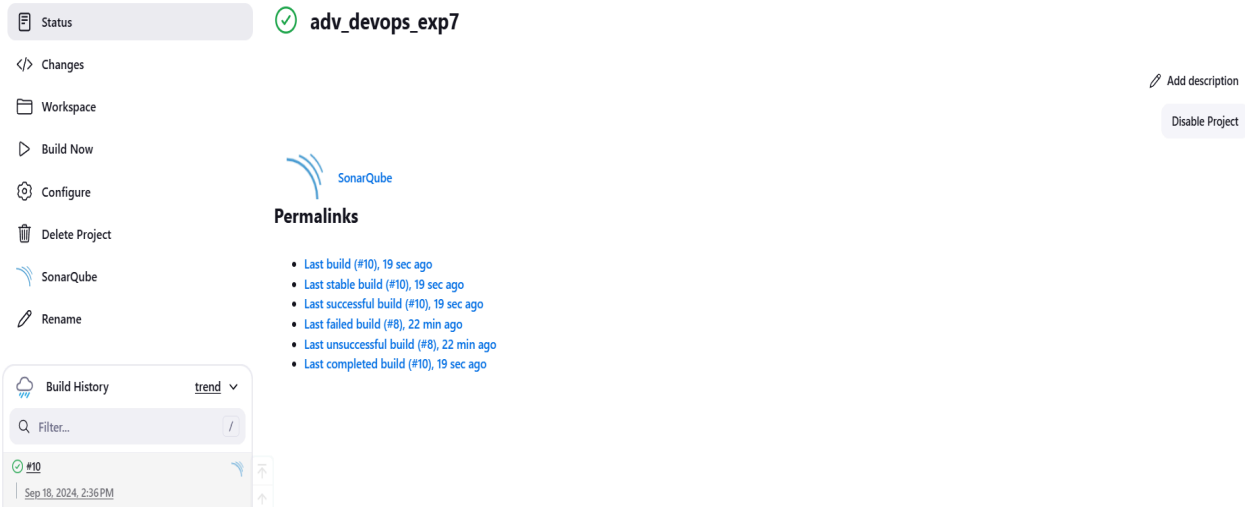
1. **Login to SonarQube:**
    - Open your browser and go to `http://localhost:9000`.
    - Log in with your admin credentials (default username is `admin`, and the password is either `admin` or your custom password if it was changed).
2. **Generate a New Token:**
    - Click on your **username** in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

Step 2: Update the Token in Jenkins

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

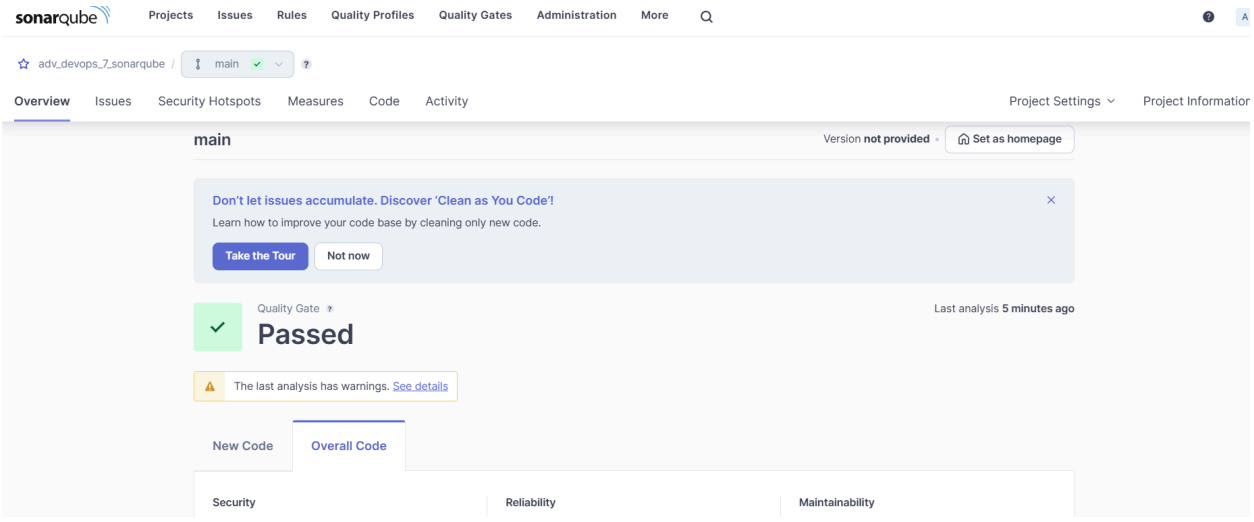


12. Run the Jenkins build.



Check the console Output

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



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

Conclusion:

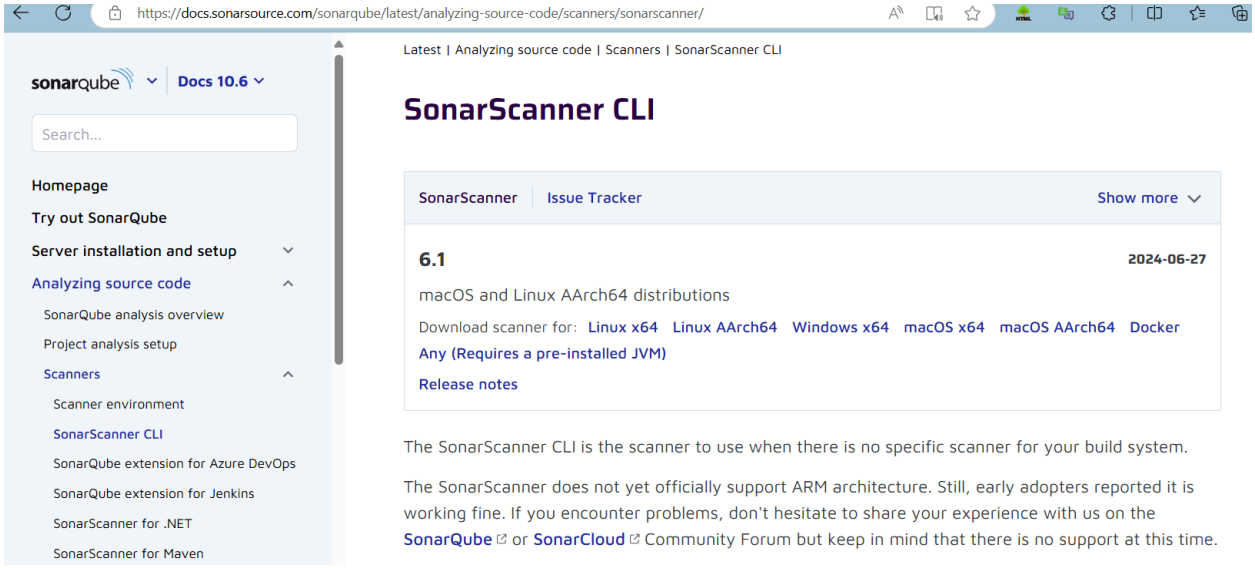
This integration has successfully built a powerful, automated system for enhancing code security and quality. By continuously scanning for vulnerabilities, code smells, and other potential issues, it ensures proactive maintenance of code standards. The seamless connection to GitHub facilitates easy tracking of changes and instant feedback. Automated reports provide valuable insights, allowing developers to address problems early in the development cycle. This streamlined process enhances both efficiency and security. As a result, the workflow becomes more reliable, with improved overall code integrity. Continuous improvement is ensured through consistent monitoring and analysis.

Experiment 8

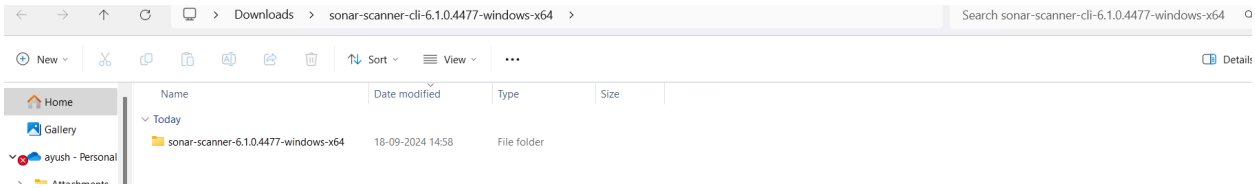
Aim: Create a Jenkins CICD Pipeline with SonarQube / GitLab Integration to perform a static analysis of the code to detect bugs, code smells, and security vulnerabilities on a sample Web / Java / Python application.

Step 1: Download sonar scanner

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



Extract the downloaded zip file in a folder.



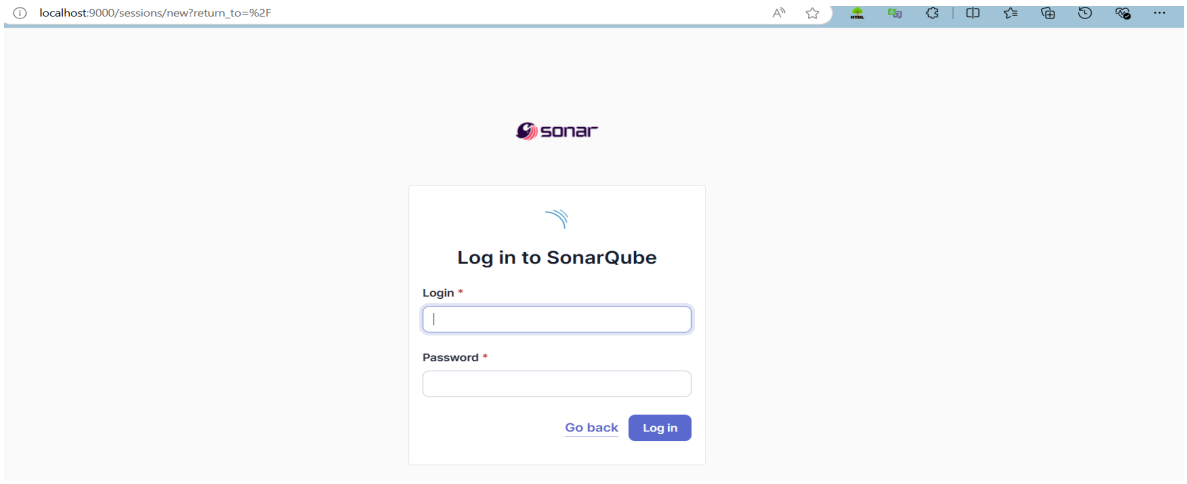
1. Install sonarqube image

Command: **docker pull sonarqube**

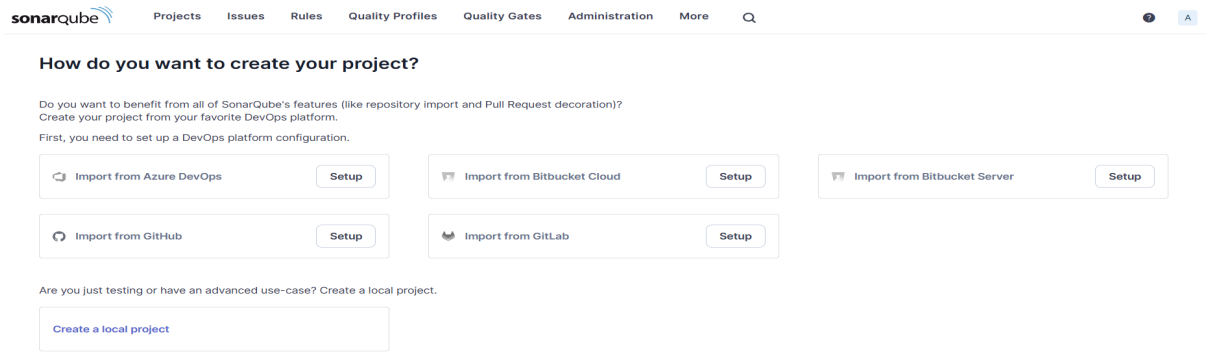
```
C:\Users\Ayush Maurya>docker pull sonarqube
Using default tag: latest
latest: Pulling from library/sonarqube
7478e0ac0f23: Pull complete
90a925ab929a: Pull complete
7d9a34308537: Pull complete
80338217a4ab: Pull complete
1a5fd5c7e184: Pull complete
7b87d6fa783d: Pull complete
bd819c9b5ead: Pull complete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

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

C:\Users\Ayush Maurya>docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
dfe47cea99897c7099833dc0d7fa99279ef09d4f3038622910a74df2630afed5
```

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

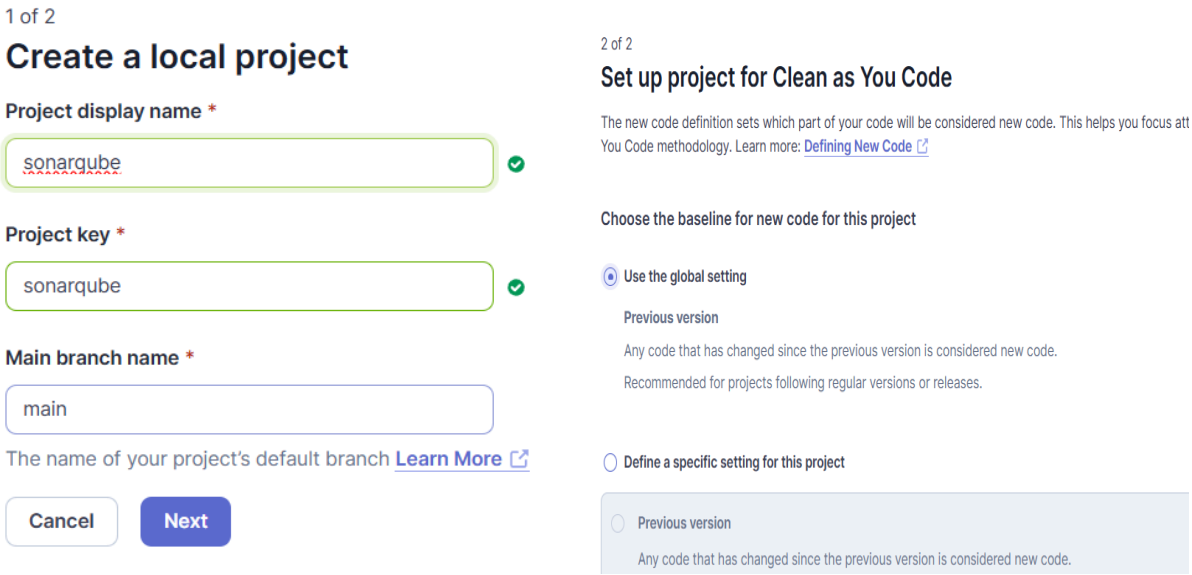
localhost port 9000.



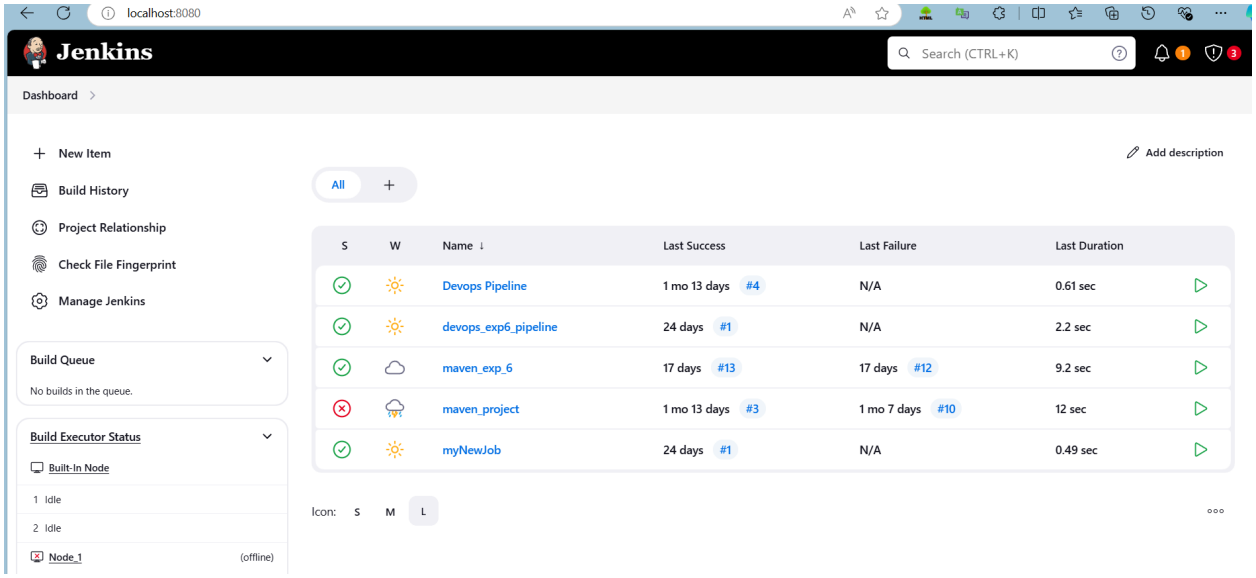
3. Login to SonarQube using username admin and password admin.



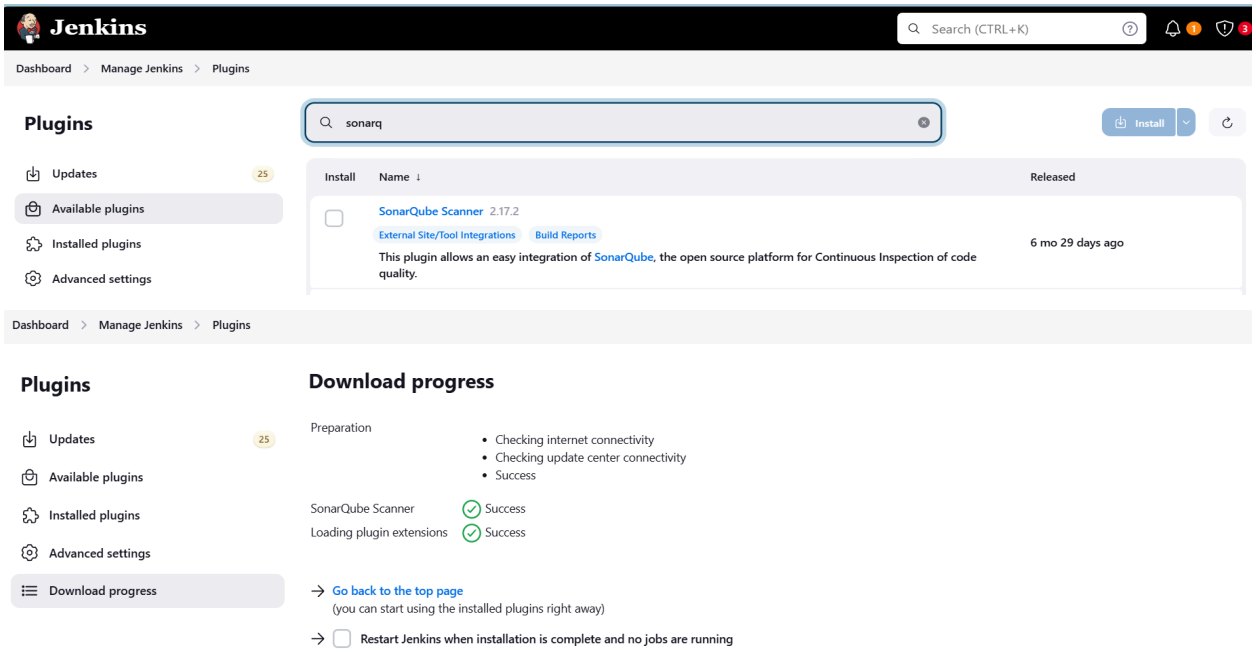
4. Create a manual project in SonarQube with the name sonarqube



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



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

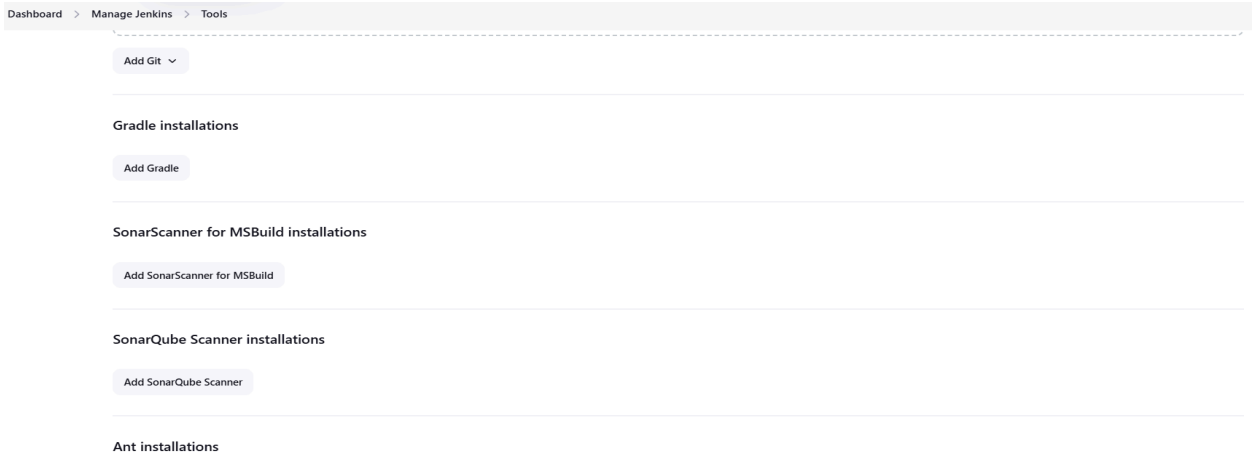


7. Under Jenkins ‘Manage Jenkins’ then go to ‘system’, scroll and look for **SonarQube Servers** and enter the details.
Enter the Server Authentication token if needed.

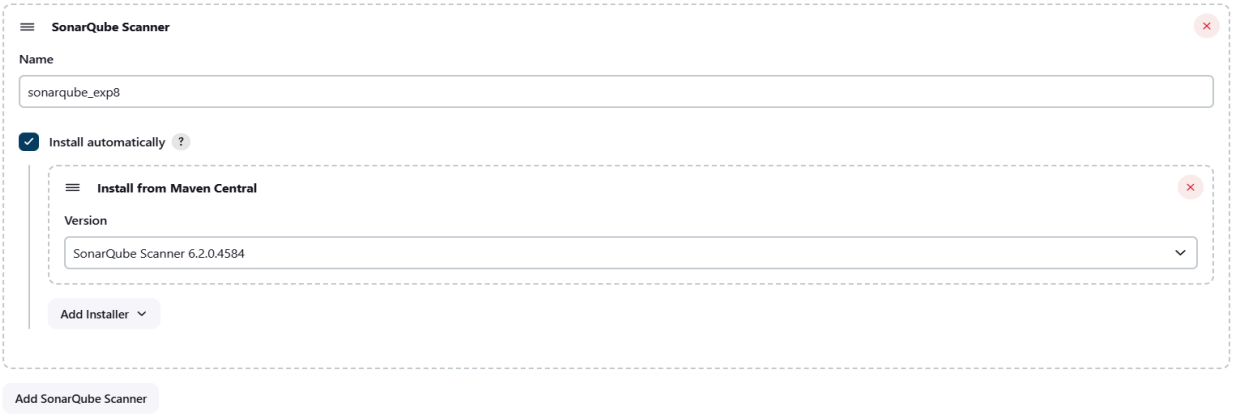
In SonarQube installations: Under **Name** add <project name of sonarqube> for me **adv_devops_7_sonarqube**
In **Server URL** Default is **http://localhost:9000**

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

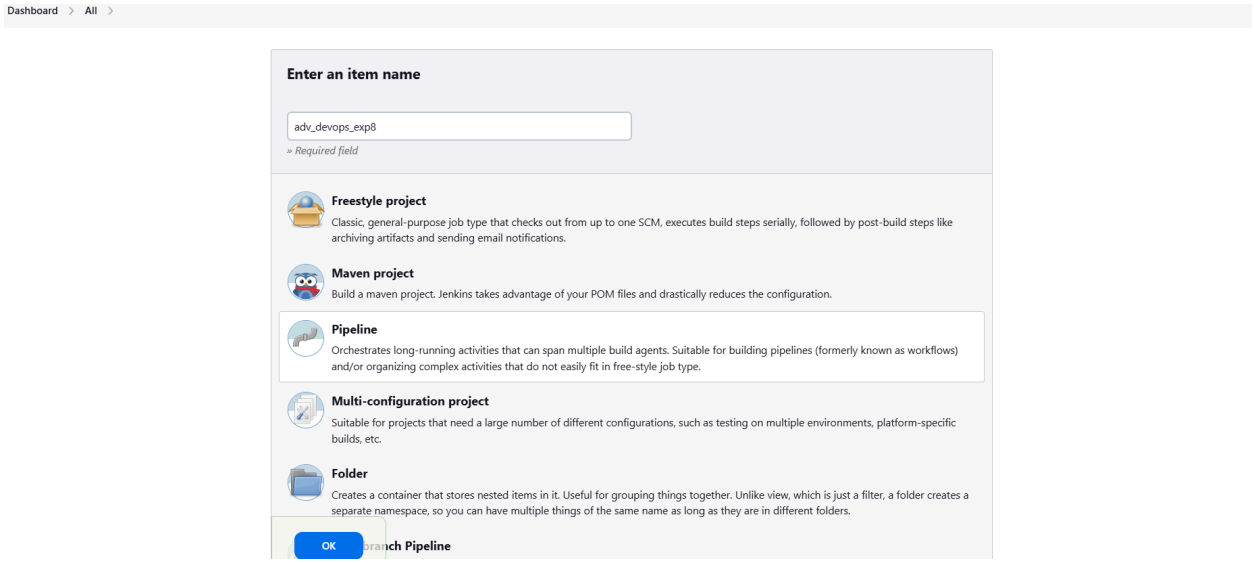
Dashboard > Manage Jenkins > Tools



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



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



10. Under Pipeline script, enter the following:

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

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

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

Pipeline

Definition

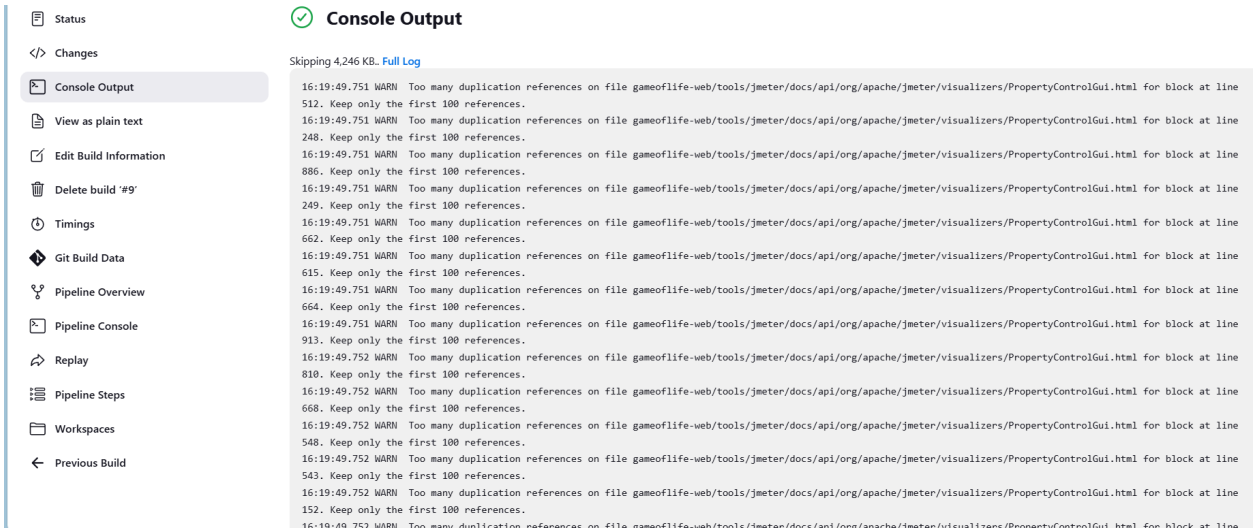
Pipeline script

Script

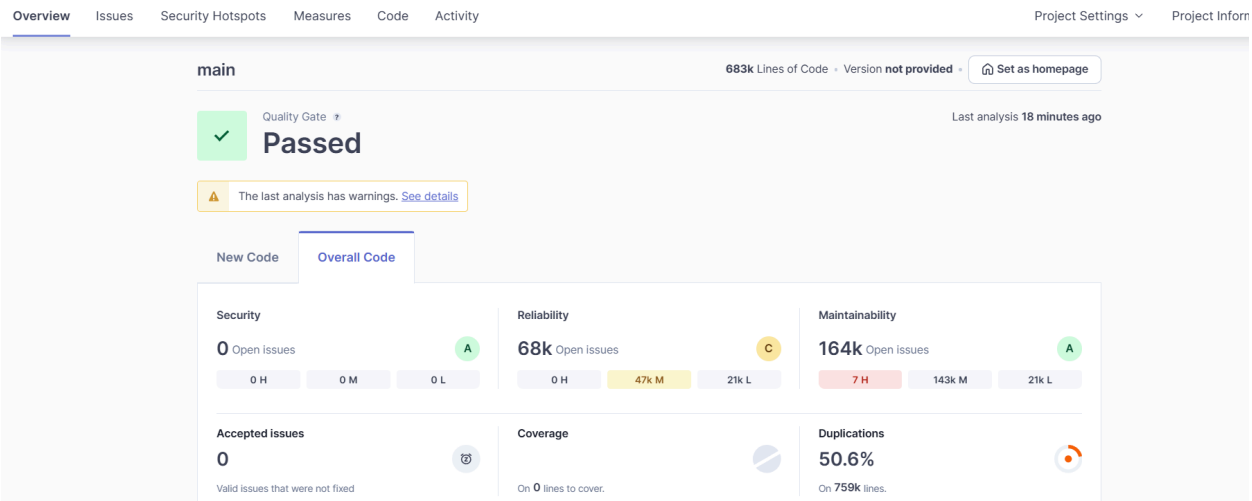
```
node {
    stage('Cloning the GitHub Repo') {
        git 'https://github.com/shazforiot/GOL.git'
    }

    stage('SonarQube analysis') {
        withSonarQubeEnv('sonarqube') {  // Ensure this matches the SonarQube environment name in Jenkins
            bat """
                "C:\\Users\\Ayush Maurya\\Downloads\\sonar-scanner-cli-6.1.0.4477-windows-x64\\sonar-scanner-6.1.0.4477-windows-x64\\bin\\sonar-scanner.
                -D sonar.login=admin ^
                -D sonar.password=Ayush3114 ^
                -D sonar.projectKey=sonarqube ^
                -D sonar.exclusions=vendor/**,resources/**,**/*.java ^
                -D sonar.host.url=http://localhost:9000/
            """
        }
    }
}
```

11. Build project

12. Check console

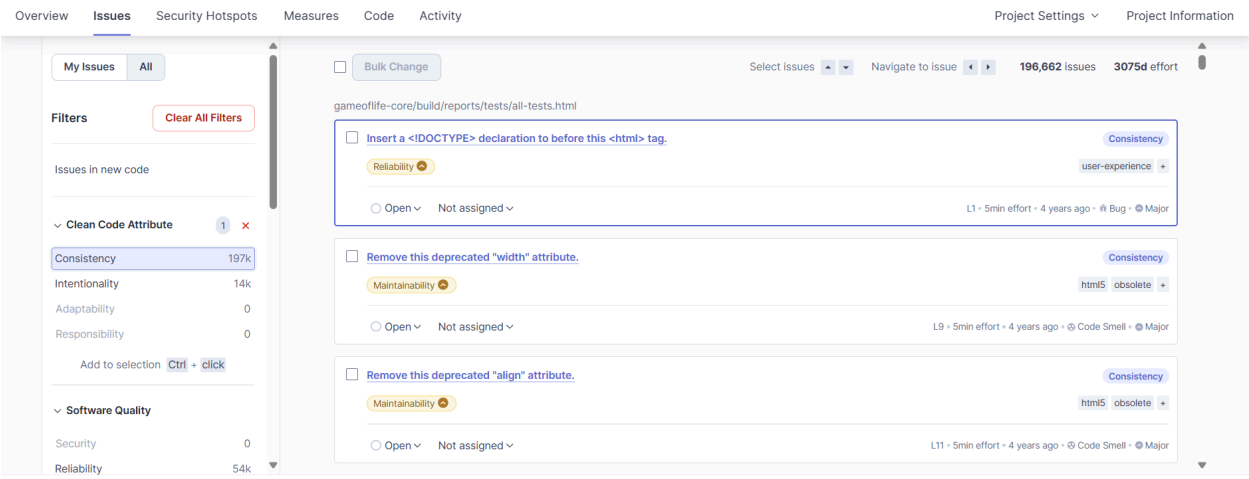


13. Now, check the project in SonarQube

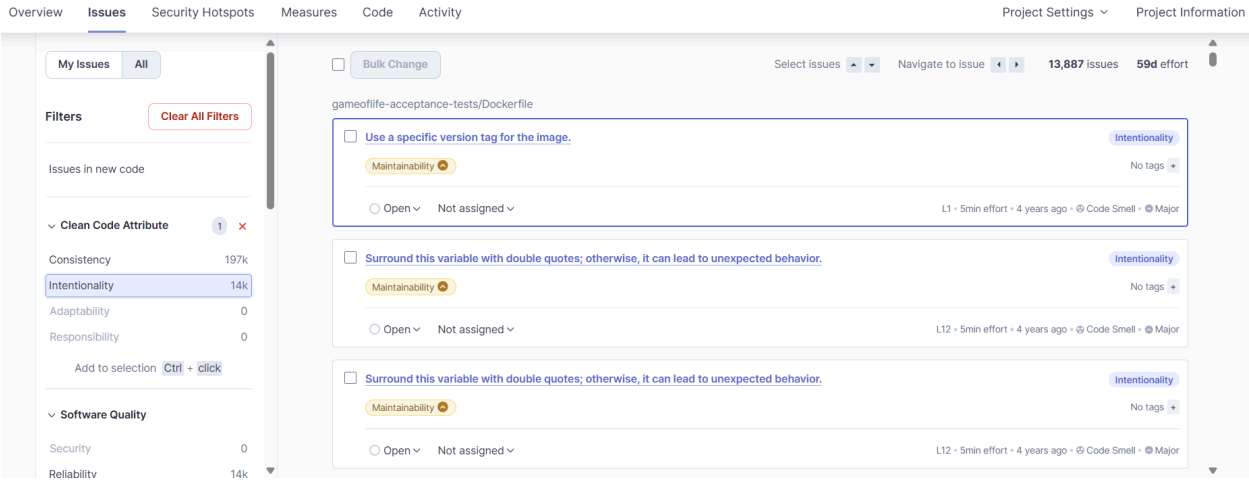


14. Code Problems

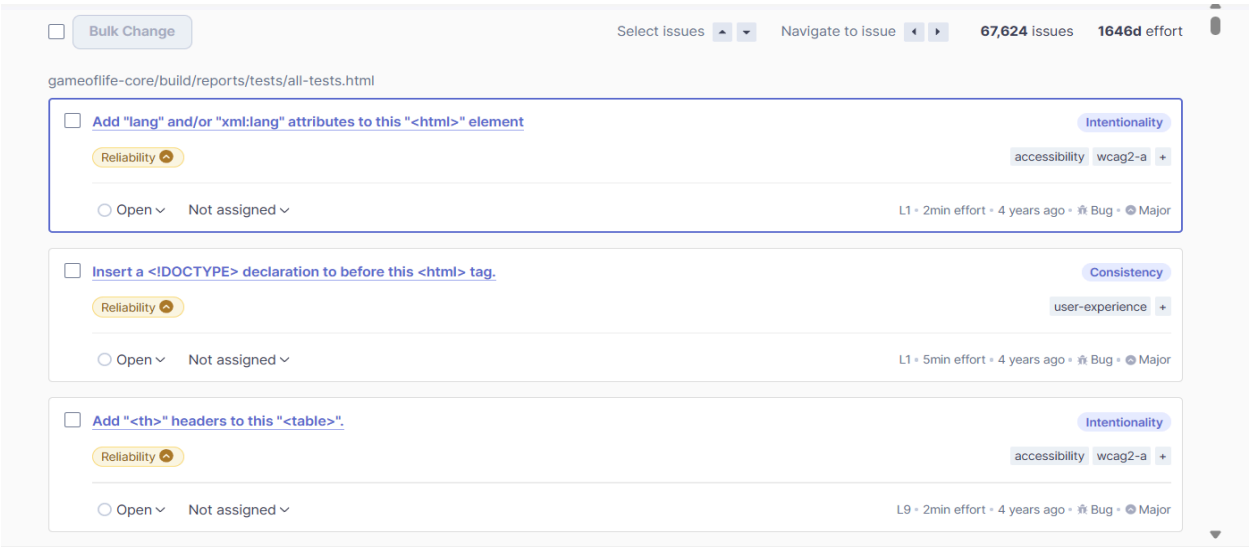
- Consistency



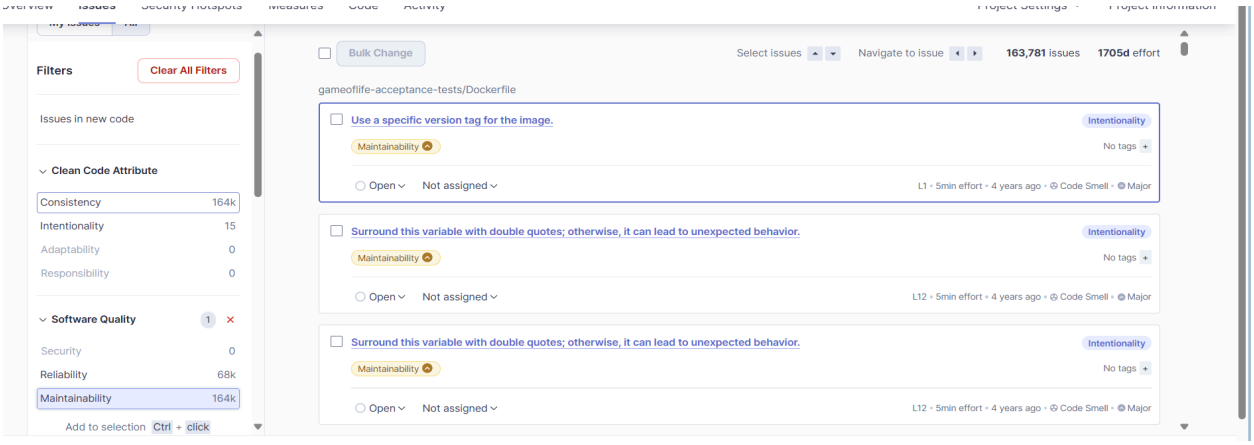
- Intentionality



- Bugs

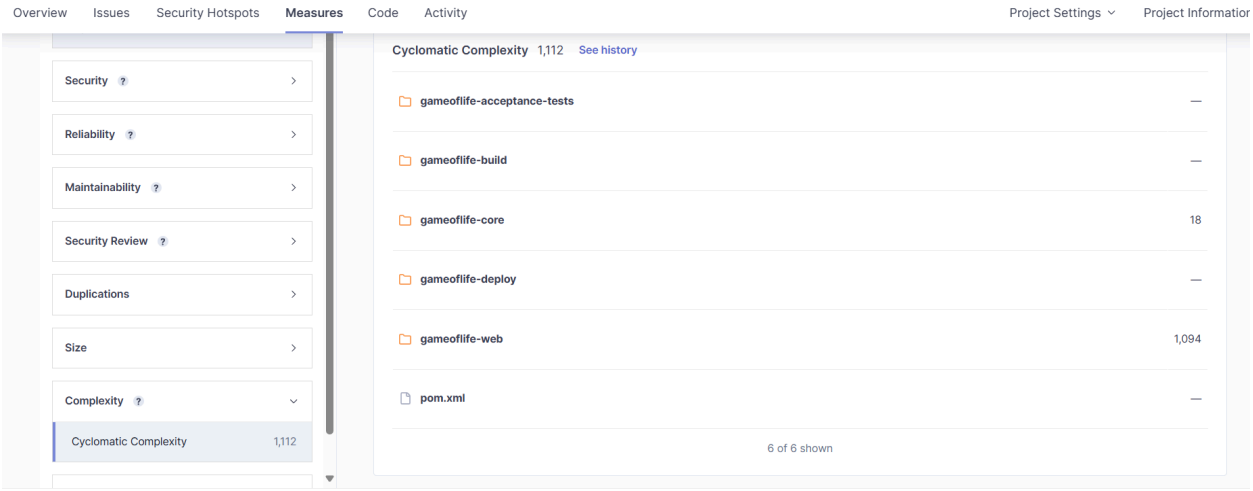


- Code Smells



- Duplications

- Cyclomatic Complexities



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

Conclusion:

This experiment established a seamless integration of Jenkins with SonarQube to automate code quality assessments within the CI/CD pipeline. SonarQube was deployed using Docker, and after setting up a project, it was configured to analyze the codebase for potential quality issues. Jenkins was configured with the necessary SonarQube plugins, allowing automated code checks through a pipeline that cloned a GitHub repository and performed a SonarQube scan. This integration ensures continuous monitoring throughout the development cycle, effectively identifying and addressing bugs, code smells, and security vulnerabilities, thereby enhancing code quality and security.

Experiment 9

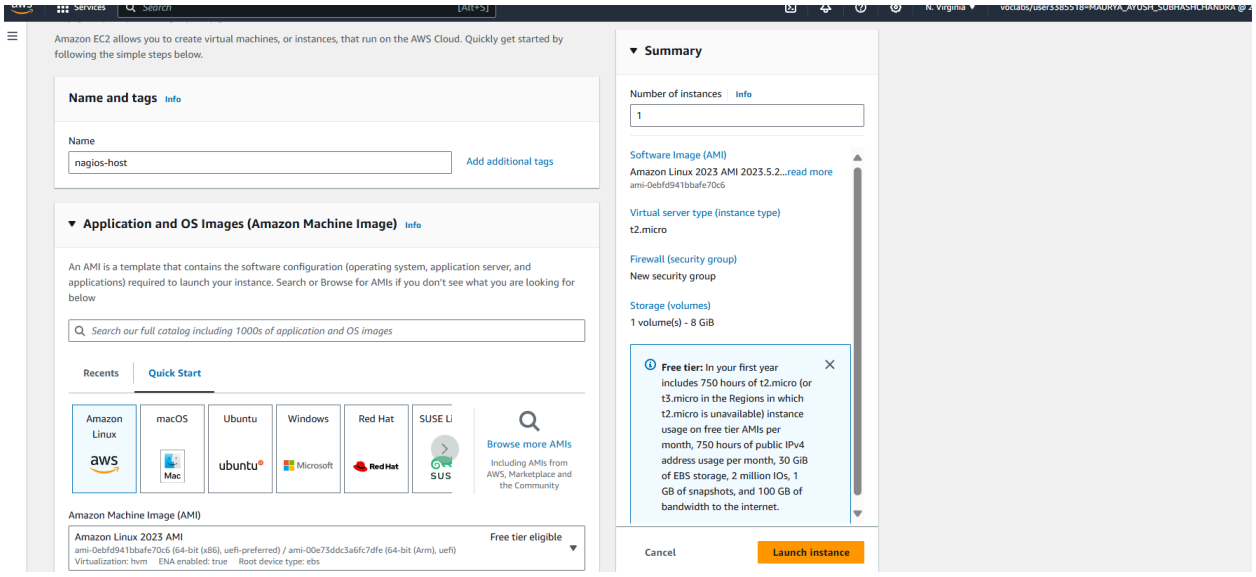
Aim: To Understand Continuous monitoring and Installation and configuration of Nagios Core, Nagios Plugins and NRPE (Nagios Remote Plugin Executor) on Linux Machine.

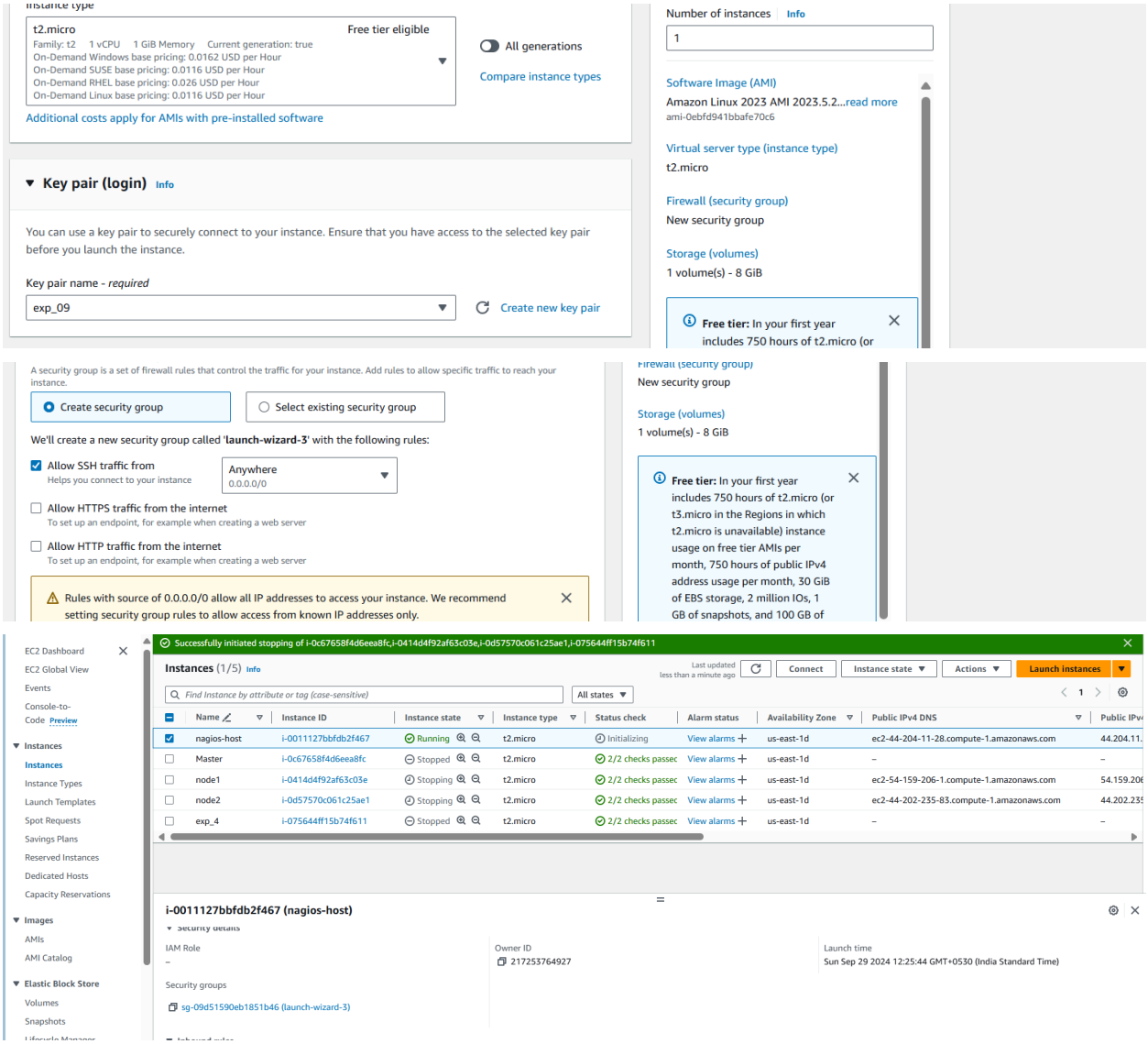
Theory:

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
.
Installation of Nagios

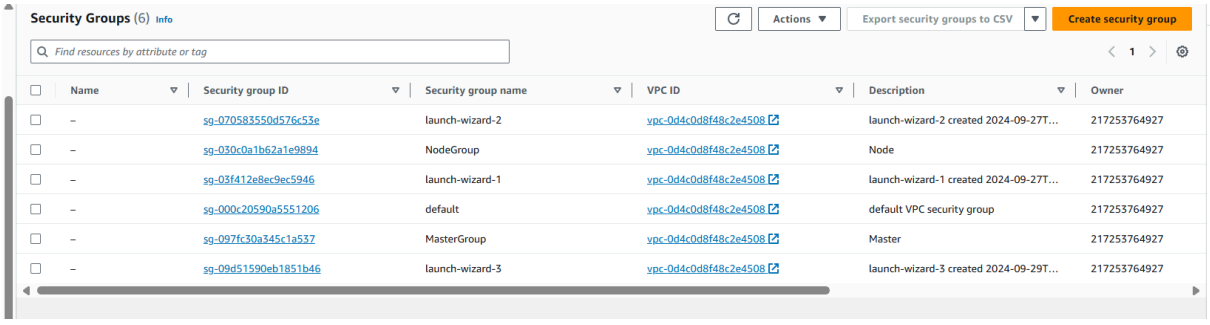
Prerequisites: AWS Free Tier

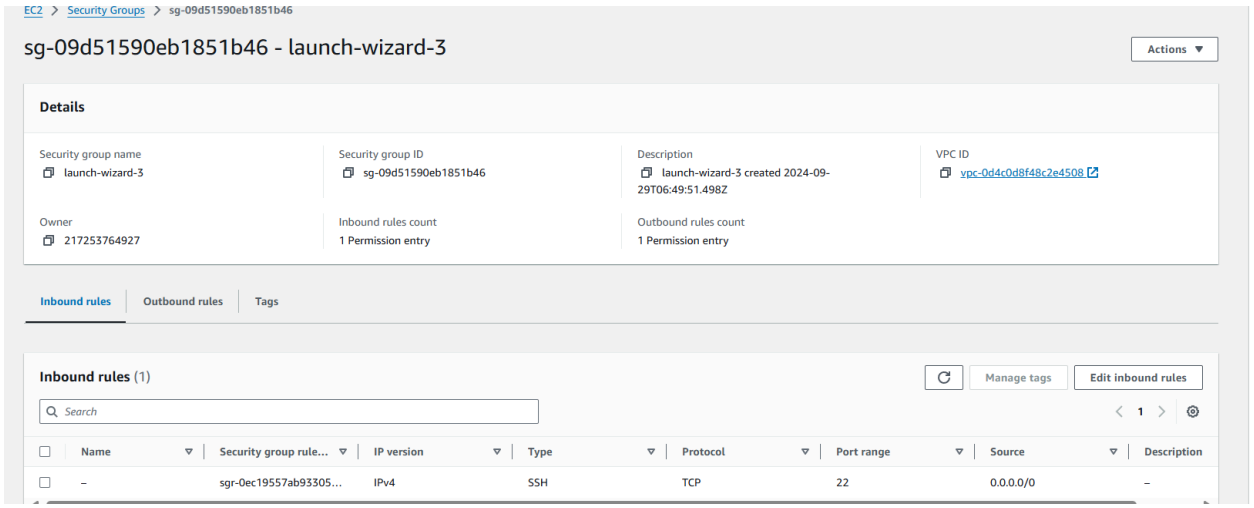
Steps:
1. Create an Amazon Linux EC2 Instance in AWS and name it - nagios-host

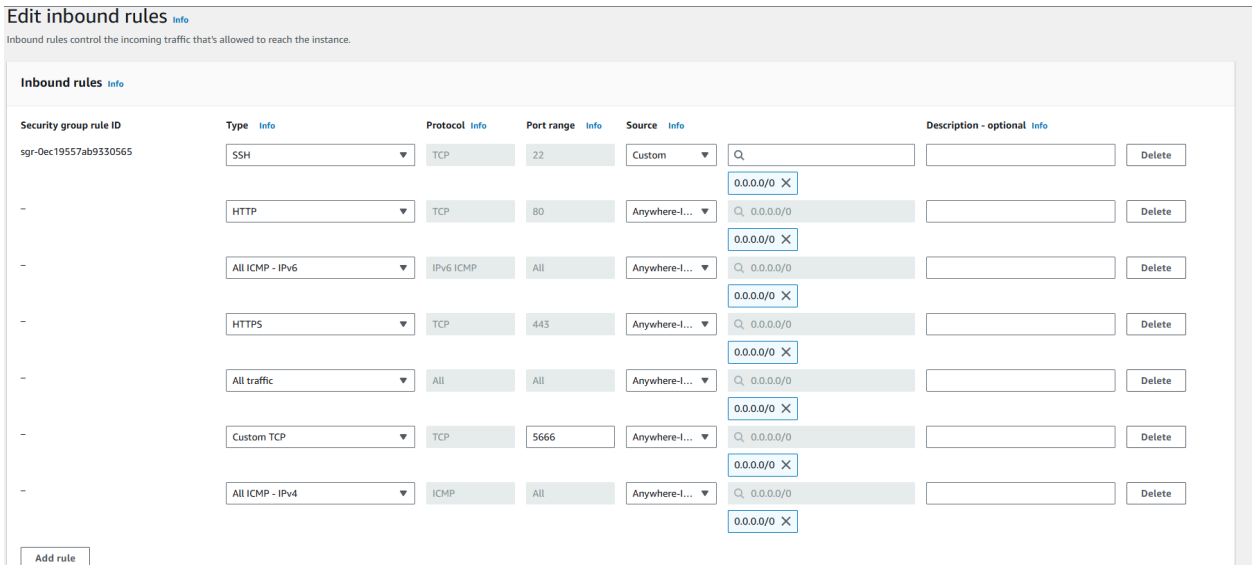


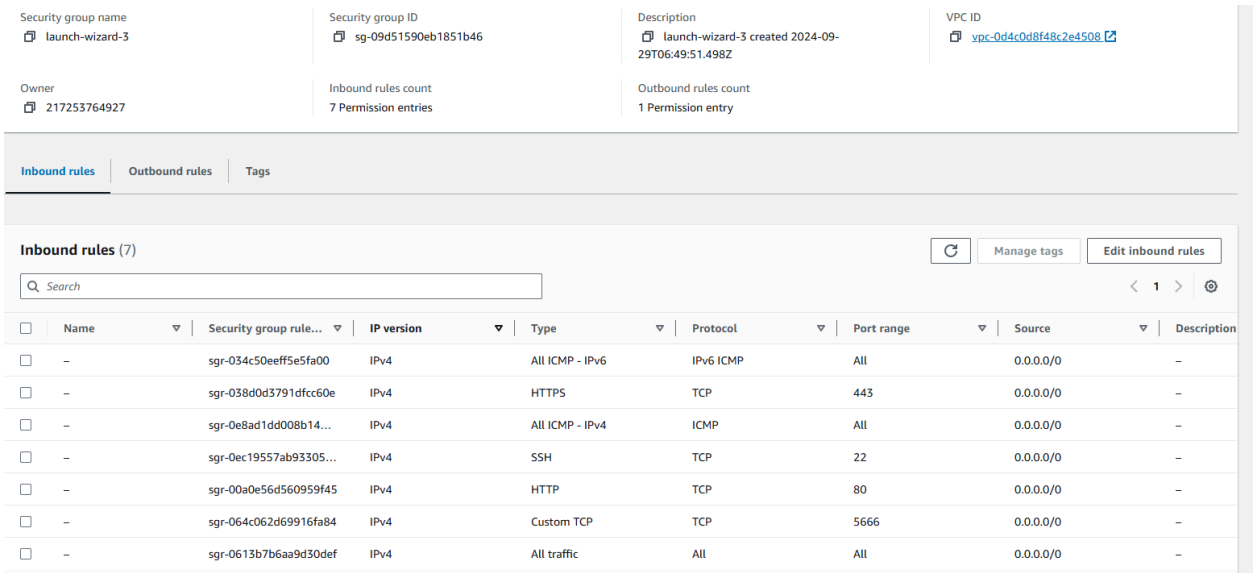


2. Under Security Group, make sure HTTP, HTTPS, SSH, ICMP are open from everywhere.



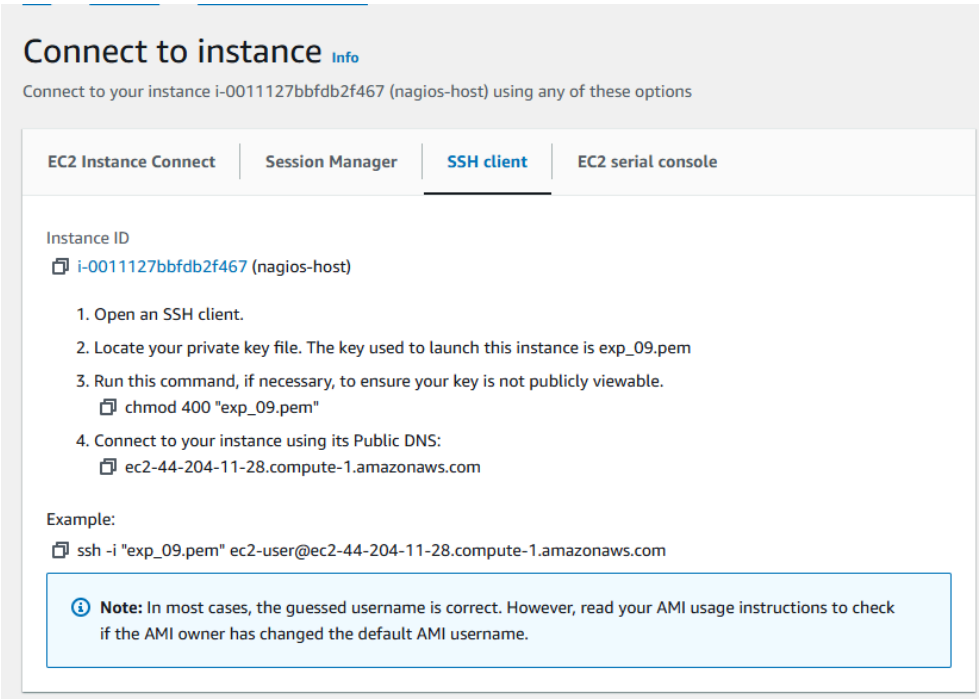






You have to edit the inbound rules of the specified Security Group for this.

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



Or open command prompt and paste ssh command.

```
Microsoft Windows [Version 10.0.22631.4169]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Ayush Maurya>ssh -i "Downloads/exp_09.pem" ec2-user@ec2-44-204-11-28.compute-1.amazonaws.com
The authenticity of host 'ec2-44-204-11-28.compute-1.amazonaws.com (44.204.11.28)' can't be established.
ED25519 key fingerprint is SHA256:v2OKH/ezl9iu7/RT6m8LWkgWzEJnnQIqrG9gKWzwC14.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-44-204-11-28.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
       #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Sun Sep 29 07:11:40 2024 from 18.206.107.27
[ec2-user@ip-172-31-91-91 ~]$
```

sudo yum update

```
[ec2-user@ip-172-31-91-91 ~]$
sudo yum update
Last metadata expiration check: 0:19:03 ago on Sun Sep 29 06:56:15 2024.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-91-91 ~]$
```

sudo yum install httpd php

```
[ec2-user@ip-172-31-91-91 ~]$ sudo yum install httpd php
Last metadata expiration check: 0:19:29 ago on Sun Sep 29 06:56:15 2024.
Dependencies resolved.
================================================================================
 Package                 Architecture  Version                    Repository  Size
================================================================================
Installing:
 httpd                   x86_64        2.4.62-1.amzn2023          amazonlinux  48 k
 php8.3                  x86_64        8.3.10-1.amzn2023.0.1      amazonlinux  10 k
Installing dependencies:
 apr                     x86_64        1.7.2-2.amzn2023.0.2       amazonlinux 129 k
 apr-util                x86_64        1.6.3-1.amzn2023.0.1       amazonlinux  98 k
 generic-logos-httpd     noarch        18.0.0-12.amzn2023.0.3     amazonlinux  19 k
 httpd-core              x86_64        2.4.62-1.amzn2023          amazonlinux 1.4 M
 httpd-filesystem        noarch        2.4.62-1.amzn2023          amazonlinux  14 k
 httpd-tools             x86_64        2.4.62-1.amzn2023          amazonlinux  81 k
 libbrotli               x86_64        1.0.9-4.amzn2023.0.2       amazonlinux 315 k
 libsodium               x86_64        1.0.19-4.amzn2023          amazonlinux 176 k
 libxslt                 x86_64        1.1.34-5.amzn2023.0.2      amazonlinux 241 k
 mailcap                 noarch        2.1.49-3.amzn2023.0.3      amazonlinux  33 k
 nginx-filesystem        noarch        1:1.24.0-1.amzn2023.0.4    amazonlinux 9.8 k
 php8.3-cli              x86_64        8.3.10-1.amzn2023.0.1      amazonlinux 3.7 M
 php8.3-common           x86_64        8.3.10-1.amzn2023.0.1      amazonlinux 737 k
 php8.3-process          x86_64        8.3.10-1.amzn2023.0.1      amazonlinux  45 k
 php8.3-xml              x86_64        8.3.10-1.amzn2023.0.1      amazonlinux 154 k
Installing weak dependencies:
 apr-util-openssl        x86_64        1.6.3-1.amzn2023.0.1       amazonlinux  17 k
 mod_http2               x86_64        2.0.27-1.amzn2023.0.3      amazonlinux 166 k
 mod_lua                 x86_64        2.4.62-1.amzn2023          amazonlinux  61 k
 php8.3-fpm              x86_64        8.3.10-1.amzn2023.0.1      amazonlinux 1.9 M
 php8.3-mbstring         x86_64        8.3.10-1.amzn2023.0.1      amazonlinux 528 k
 php8.3-opcache          x86_64        8.3.10-1.amzn2023.0.1      amazonlinux 379 k
 php8.3-pdo              x86_64        8.3.10-1.amzn2023.0.1      amazonlinux  89 k
 php8.3-sodium           x86_64        8.3.10-1.amzn2023.0.1      amazonlinux  41 k

Transaction Summary
```

```
Total                                                          22 MB/s |  10 MB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                              1/1
  Installing       : php8.3-common-8.3.10-1.amzn2023.0.1.x86_64                   1/25
  Installing       : apr-1.7.2-2.amzn2023.0.2.x86_64                              2/25
  Installing       : apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64                 3/25
  Installing       : apr-util-1.6.3-1.amzn2023.0.1.x86_64                         4/25
  Installing       : mailcap-2.1.49-3.amzn2023.0.3.noarch                         5/25
  Running scriptlet: httpd-filesystem-2.4.62-1.amzn2023.noarch                    6/25
```

sudo yum install gcc glibc glibc-common

```
[ec2-user@ip-172-31-91-91 ~]$ sudo yum install gcc glibc glibc-common
Last metadata expiration check: 0:20:41 ago on Sun Sep 29 06:56:15 2024.
Package glibc-2.34-52.amzn2023.0.11.x86_64 is already installed.
Package glibc-common-2.34-52.amzn2023.0.11.x86_64 is already installed.
Dependencies resolved.
================================================================================
 Package                 Architecture  Version                    Repository  Size
================================================================================
Installing:
 gcc                     x86_64        11.4.1-2.amzn2023.0.2      amazonlinux  32 M
Installing dependencies:
 annobin-docs            noarch        10.93-1.amzn2023.0.1       amazonlinux  92 k
 annobin-plugin-gcc      x86_64        10.93-1.amzn2023.0.1       amazonlinux 887 k
 cpp                     x86_64        11.4.1-2.amzn2023.0.2      amazonlinux  10 M
 gc                      x86_64        8.0.4-5.amzn2023.0.2       amazonlinux 105 k
 glibc-devel             x86_64        2.34-52.amzn2023.0.11      amazonlinux  27 k
 glibc-headers-x86       noarch        2.34-52.amzn2023.0.11      amazonlinux 427 k
 guile22                 x86_64        2.2.7-2.amzn2023.0.3       amazonlinux 6.4 M
 kernel-headers          x86_64        6.1.109-118.189.amzn2023   amazonlinux 1.4 M
 libmpc                  x86_64        1.2.1-2.amzn2023.0.2       amazonlinux  62 k
 libtool-ltdl            x86_64        2.4.7-1.amzn2023.0.3       amazonlinux  38 k
 libxcrypt-devel         x86_64        4.4.33-7.amzn2023          amazonlinux  32 k
 make                    x86_64        1:4.3-5.amzn2023.0.2       amazonlinux 534 k

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

Total download size: 52 M
```

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

Complete!
```

sudo yum install gd gd-devel

```
[ec2-user@ip-172-31-91-91 ~]$ sudo yum install gd gd-devel
Last metadata expiration check: 0:21:30 ago on Sun Sep 29 06:56:15 2024.
Dependencies resolved.
================================================================================
 Package                 Architecture  Version                    Repository  Size
================================================================================
Installing:
 gd                      x86_64        2.3.3-5.amzn2023.0.3       amazonlinux 139 k
 gd-devel                x86_64        2.3.3-5.amzn2023.0.3       amazonlinux  38 k
Installing dependencies:
 brotli                  x86_64        1.0.9-4.amzn2023.0.2       amazonlinux 314 k
 brotli-devel            x86_64        1.0.9-4.amzn2023.0.2       amazonlinux  31 k
 bzip2-devel             x86_64        1.0.8-6.amzn2023.0.2       amazonlinux 214 k
 cairo                   x86_64        1.17.6-2.amzn2023.0.1      amazonlinux 684 k
 cmake-filesystem        x86_64        3.22.2-1.amzn2023.0.4      amazonlinux  16 k
 fontconfig              x86_64        2.13.94-2.amzn2023.0.2     amazonlinux 273 k
```

```
Installed:
  brotli-1.0.9-4.amzn2023.0.2.x86_64                  brotli-devel-1.0.9-4.amzn2023.0.2.x86_64                  bzip2-devel-1.0.8-6.amzn2023.0.2.x86_64
  cairo-1.17.6-2.amzn2023.0.1.x86_64                  cmake-filesystem-3.22.2-1.amzn2023.0.4.x86_64             fontconfig-2.13.94-2.amzn2023.0.2.x86_64
  fontconfig-devel-2.13.94-2.amzn2023.0.2.x86_64      fonts-filesystem-1:2.0.5-12.amzn2023.0.2.noarch           freetype-2.13.2-5.amzn2023.0.1.x86_64
  freetype-devel-2.13.2-5.amzn2023.0.1.x86_64         gd-2.3.3-5.amzn2023.0.3.x86_64                            gd-devel-2.3.3-5.amzn2023.0.3.x86_64
  glib2-devel-2.74.7-689.amzn2023.0.2.x86_64          google-noto-fonts-common-20201206-2.amzn2023.0.2.noarch   google-noto-sans-vf-fonts-20201206-2.amzn2023.0.2.noarch
  graphite2-1.3.14-7.amzn2023.0.2.x86_64              graphite2-devel-1.3.14-7.amzn2023.0.2.x86_64              harfbuzz-7.0.0-2.amzn2023.0.1.x86_64
  harfbuzz-devel-7.0.0-2.amzn2023.0.1.x86_64          harfbuzz-icu-7.0.0-2.amzn2023.0.1.x86_64                  jbigkit-libs-2.1-21.amzn2023.0.2.x86_64
  langpacks-core-font-en-3.0-21.amzn2023.0.4.noarch   libICE-1.0.10-6.amzn2023.0.2.x86_64                       libSM-1.2.3-8.amzn2023.0.2.x86_64
  libX11-1.7.2-3.amzn2023.0.4.x86_64                  libX11-common-1.7.2-3.amzn2023.0.4.noarch                 libX11-devel-1.7.2-3.amzn2023.0.4.x86_64
  libX11-xcb-1.7.2-3.amzn2023.0.4.x86_64              libXau-1.0.9-6.amzn2023.0.2.x86_64                        libXau-devel-1.0.9-6.amzn2023.0.2.x86_64
  libXext-1.3.4-6.amzn2023.0.2.x86_64                 libXpm-3.5.15-2.amzn2023.0.3.x86_64                       libXpm-devel-3.5.15-2.amzn2023.0.3.x86_64
  libXrender-0.9.10-14.amzn2023.0.2.x86_64            libXt-1.2.0-4.amzn2023.0.2.x86_64                         libblkid-devel-2.37.4-1.amzn2023.0.4.x86_64
  libffi-devel-3.4.4-1.amzn2023.0.1.x86_64            libicu-67.1-7.amzn2023.0.3.x86_64                         libicu-devel-67.1-7.amzn2023.0.3.x86_64
  libjpeg-turbo-2.1.4-2.amzn2023.0.5.x86_64           libjpeg-turbo-devel-2.1.4-2.amzn2023.0.5.x86_64           libmount-devel-2.37.4-1.amzn2023.0.4.x86_64
  libpng-2:1.6.37-10.amzn2023.0.6.x86_64              libpng-devel-2:1.6.37-10.amzn2023.0.6.x86_64              libselinux-devel-3.4-5.amzn2023.0.2.x86_64
  libsepol-devel-3.4-3.amzn2023.0.3.x86_64            libtiff-4.4.0-4.amzn2023.0.18.x86_64                      libtiff-devel-4.4.0-4.amzn2023.0.18.x86_64
  libwebp-1.2.4-1.amzn2023.0.6.x86_64                 libwebp-devel-1.2.4-1.amzn2023.0.6.x86_64                 libxcb-1.13.1-7.amzn2023.0.2.x86_64
  libxcb-devel-1.13.1-7.amzn2023.0.2.x86_64           libxml2-devel-2.10.4-1.amzn2023.0.6.x86_64                pcre2-devel-10.40-1.amzn2023.0.3.x86_64
  pcre2-utf16-10.40-1.amzn2023.0.3.x86_64             pcre2-utf32-10.40-1.amzn2023.0.3.x86_64                   pixman-0.40.0-3.amzn2023.0.3.x86_64
  sysprof-capture-devel-3.40.1-2.amzn2023.0.2.x86_64  xml-common-0.6.3-56.amzn2023.0.2.noarch                   xorg-x11-proto-devel-2021.4-1.amzn2023.0.2.noarch
  xz-devel-5.2.5-9.amzn2023.0.2.x86_64                zlib-devel-1.2.11-33.amzn2023.0.5.x86_64

Complete!
```

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

(password : *ayushmau)*

```
Complete!
[ec2-user@ip-172-31-91-91 ~]$ sudo adduser -m nagios
sudo passwd nagios
Changing password for user nagios.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
Sorry, passwords do not match.
New password:
BAD PASSWORD: The password contains the user name in some form
Retype new password:
Sorry, passwords do not match.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[ec2-user@ip-172-31-91-91 ~]$
```

6. Create a new user group

sudo groupadd nagcmd

```
[ec2-user@ip-172-31-91-91 ~]$ sudo groupadd nagcmd
[ec2-user@ip-172-31-91-91 ~]$
```

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

```
[ec2-user@ip-172-31-91-91 ~]$ sudo usermod -a -G nagcmd nagios
sudo usermod -a -G nagcmd apache
[ec2-user@ip-172-31-91-91 ~]$
```

8. Create a new directory for Nagios downloads

mkdir ~/downloads

cd ~/downloads

```
[ec2-user@ip-172-31-91-91 ~]$ mkdir ~/downloads
cd ~/downloads
```

9. Use wget to download the source zip files.

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

```
[ec2-user@ip-172-31-91-91 downloads]$ cd ..
[ec2-user@ip-172-31-91-91 ~]$ cd ~/downloads
[ec2-user@ip-172-31-91-91 downloads]$ wget https://assets.nagios.com/downloa
ds/nagioscore/releases/nagios-4.5.5.tar.gz
--2024-09-29 09:11:59--  https://assets.nagios.com/downloads/nagioscore/rele
ases/nagios-4.5.5.tar.gz
Resolving assets.nagios.com (assets.nagios.com)... 45.79.49.120, 2600:3c00::
f03c:92ff:fef7:45ce
Connecting to assets.nagios.com (assets.nagios.com)|45.79.49.120|:443... con
nected.
HTTP request sent, awaiting response... 200 OK
Length: 2065473 (2.0M) [application/x-gzip]
Saving to: ‘nagios-4.5.5.tar.gz’

nagios-4.5.5.tar.g 100%[=================>]   1.97M  5.07MB/s    in 0.4s

2024-09-29 09:11:59 (5.07 MB/s) - ‘nagios-4.5.5.tar.gz’ saved [2065473/20654
73]

[ec2-user@ip-172-31-91-91 downloads]$
```

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

```
[ec2-user@ip-172-31-91-91 nagios-4.5.5]$ cd ..
[ec2-user@ip-172-31-91-91 downloads]$ wget https://nagios-plugins.org/downlo
ad/nagios-plugins-2.4.11.tar.gz
--2024-09-29 09:14:28--  https://nagios-plugins.org/download/nagios-plugins-
2.4.11.tar.gz
Resolving nagios-plugins.org (nagios-plugins.org)... 45.56.123.251
Connecting to nagios-plugins.org (nagios-plugins.org)|45.56.123.251|:443...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 2753049 (2.6M) [application/x-gzip]
Saving to: ‘nagios-plugins-2.4.11.tar.gz’

nagios-plugins-2.4 100%[=================>]   2.62M  6.92MB/s    in 0.4s
```

10. Use tar to unzip and change to that directory.
tar zxvf nagios-4.5.5.tar.gz

```
[ec2-user@ip-172-31-91-91 downloads]$ tar zxvf nagios-4.0.8.tar.gz
nagios-4.0.8/
nagios-4.0.8/.gitignore
nagios-4.0.8/Changelog
nagios-4.0.8/INSTALLING
nagios-4.0.8/LEGAL
nagios-4.0.8/LICENSE
nagios-4.0.8/Makefile.in
nagios-4.0.8/README
nagios-4.0.8/README.asciidoc
nagios-4.0.8/THANKS
nagios-4.0.8/UPGRADING
nagios-4.0.8/base/
nagios-4.0.8/base/.gitignore
```

11. Run the configuration script with the same group name you previously created.
./configure --with-command-group=nagcmd

Here we go an error

```
[ec2-user@ip-172-31-91-91 downloads]$ ./configure --with-command-group=nagcmd
-bash: ./configure: No such file or directory
[ec2-user@ip-172-31-91-91 downloads]$
```

Solution

Navigate to nagios folder in downloads

```
[ec2-user@ip-172-31-91-91 downloads]$ ls
nagios-4.0.8  nagios-4.0.8.tar.gz  nagios-plugins-2.0.3.tar.gz
[ec2-user@ip-172-31-91-91 downloads]$ cd nagios-4.0.8
[ec2-user@ip-172-31-91-91 nagios-4.0.8]$
```

Error 2: Cannot find SSL headers.
Solution: Install openssl dev library

Steps:
sudo yum install openssl-devel

```
[ec2-user@ip-172-31-91-91 nagios-4.5.5]$ sudo yum install openssl-devel
Last metadata expiration check: 2:24:05 ago on Sun Sep 29 06:56:15 2024.
Dependencies resolved.
================================================================================
 Package          Arch      Version                     Repository      Size
================================================================================
Installing:
 openssl-devel    x86_64    1:3.0.8-1.amzn2023.0.14     amazonlinux    3.0 M

Transaction Summary
================================================================================
Install  1 Package

Total download size: 3.0 M
Installed size: 4.7 M
Is this ok [y/N]: y
Downloading Packages:
```

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

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

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


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

[ec2-user@ip-172-31-91-91 nagios-4.5.5]$
```

12. Compile the source code.
make all

```
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/base'
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_
CONFIG_H -DNSCORE -c -o nagios.o ./nagios.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_
CONFIG_H -DNSCORE -c -o broker.o broker.c
```

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-172-31-91-91 nagios-4.5.5]$ make all

sudo make install
sudo make install-init
sudo make install-config
sudo make install-commandmode
cd ./base && make
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/base'
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_
CONFIG_H -DNSCORE -c -o nagios.o ./nagios.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_
CONFIG_H -DNSCORE -c -o broker.o broker.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_
CONFIG_H -DNSCORE -c -o nebmods.o nebmods.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_
CONFIG_H -DNSCORE -c -o ../common/shared.o ../common/shared.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_
CONFIG_H -DNSCORE -c -o query-handler.o query-handler.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_
CONFIG_H -DNSCORE -c -o workers.o workers.c
In function ‘get_wproc_list’,
    inlined from ‘get_worker’ at workers.c:277:12:
workers.c:253:17: warning: ‘%s’ directive argument is null [-Wformat-overflo
w=]
  253 |                 log_debug_info(DEBUGL_CHECKS, 1, "Found specialized
worker(s) for '%s'", (slash && *slash != '/') ? slash : cmd_name);
      |                 ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_
CONFIG_H -DNSCORE -c -o checks.o checks.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_
CONFIG_H -DNSCORE -c -o config.o config.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE
```

14. Edit the config file and change the email address.
sudo nano /usr/local/nagios/etc/objects/contacts.cfg

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

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

define contact {

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



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

# We only have one contact in this simple configuration file, so there is
# no need to create more than one contact group.

define contactgroup {

    contactgroup_name       admins
    alias                   Nagios Administrators
    members                 nagiosadmin
}

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

And change email with your email

15. Configure the web interface.
sudo make install-webconf

```
[ec2-user@ip-172-31-91-91 nagios-4.5.5]$ sudo make install-webconf
/usr/bin/install -c -m 644 sample-config/httpd.conf /etc/httpd/conf.d/nagios
.conf
if [ 0 -eq 1 ]; then \
        ln -s /etc/httpd/conf.d/nagios.conf /etc/apache2/sites-enabled/nagi
s.conf; \
fi

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

[ec2-user@ip-172-31-91-91 nagios-4.5.5]$
```

16. 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-172-31-91-91 nagios-4.5.5]$ sudo htpasswd -c /usr/local/nagios/
etc/htpasswd.users nagiosadmin
New password:
Re-type new password:
Adding password for user nagiosadmin
[ec2-user@ip-172-31-91-91 nagios-4.5.5]$
```

Password: Ayushmau

17. Restart Apache
sudo service httpd restart

```
Adding password for user nagiosadmin
[ec2-user@ip-172-31-91-91 nagios-4.5.5]$ sudo service httpd restart
Redirecting to /bin/systemctl restart httpd.service
[ec2-user@ip-172-31-91-91 nagios-4.5.5]$
```

18. 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-172-31-91-91 downloads]$ cd ~/downloads
[ec2-user@ip-172-31-91-91 downloads]$ tar zxvf nagios-plugins-2.4.11.tar.gz
nagios-plugins-2.4.11/
nagios-plugins-2.4.11/build-aux/
nagios-plugins-2.4.11/build-aux/compile
nagios-plugins-2.4.11/build-aux/config.guess
nagios-plugins-2.4.11/build-aux/config.rpath
nagios-plugins-2.4.11/build-aux/config.sub
nagios-plugins-2.4.11/build-aux/install-sh
nagios-plugins-2.4.11/build-aux/ltmain.sh
nagios-plugins-2.4.11/build-aux/missing
nagios-plugins-2.4.11/build-aux/mkinstalldirs
nagios-plugins-2.4.11/build-aux/depcomp
nagios-plugins-2.4.11/build-aux/snippet/
nagios-plugins-2.4.11/build-aux/snippet/_Noreturn.h
nagios-plugins-2.4.11/build-aux/snippet/arg-nonnull.h
nagios-plugins-2.4.11/build-aux/snippet/c++defs.h
nagios-plugins-2.4.11/build-aux/snippet/warn-on-use.h
nagios-plugins-2.4.11/build-aux/test-driver
nagios-plugins-2.4.11/config_test/
```

19. Compile and install plugins

cd nagios-plugins-2.4.11

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

```
[ec2-user@ip-172-31-91-91 downloads]$ cd nagios-plugins-2.4.11
./configure --with-nagios-user=nagios --with-nagios-group=nagios
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /usr/bin/mkdir -p
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking whether make supports nested variables... yes
checking whether to enable maintainer-specific portions of Makefiles... yes
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking whether gcc understands -c and -o together... yes
checking whether make supports the include directive... yes (GNU style)
checking dependency style of gcc... gcc3
checking how to run the C preprocessor... gcc -E
checking for grep that handles long lines and -e... /usr/bin/grep
checking for egrep... /usr/bin/grep -E
checking for Minix Amsterdam compiler... no
checking for ar... ar
checking for ranlib... ranlib
```

make
sudo make install

```
[ec2-user@ip-172-31-91-91 nagios-plugins-2.4.11]$ make
sudo make install
make  all-recursive
make[1]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11'
Making all in gl
make[2]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11/
gl'
rm -f alloca.h-t alloca.h && \
{ echo '/* DO NOT EDIT! GENERATED AUTOMATICALLY! */'; \
  cat ./alloca.in.h; \
} > alloca.h-t && \
mv -f alloca.h-t alloca.h
rm -f c++defs.h-t c++defs.h && \
sed -n -e '/_GL_CXXDEFS/,$p' \
  < ../build-aux/snippet/c++defs.h \
  > c++defs.h-t && \
mv c++defs.h-t c++defs.h
rm -f warn-on-use.h-t warn-on-use.h && \
sed -n -e '/^.ifndef/,$p' \
  < ../build-aux/snippet/warn-on-use.h \
  > warn-on-use.h-t && \
mv warn-on-use.h-t warn-on-use.h
rm -f arg-nonnull.h-t arg-nonnull.h && \
sed -n -e '/GL_ARG_NONNULL/,$p' \
  < ../build-aux/snippet/arg-nonnull.h \
  > arg-nonnull.h-t && \
mv arg-nonnull.h-t arg-nonnull.h
/usr/bin/mkdir -p arpa
```

```
make[1]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11'
make[2]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11'
make[2]: Nothing to be done for 'install-exec-am'.
make[2]: Nothing to be done for 'install-data-am'.
make[2]: Leaving directory '/home/ec2-user/downloads/nagios-plugins-2.4.11'
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-plugins-2.4.11'
[ec2-user@ip-172-31-91-91 nagios-plugins-2.4.11]$
```

20. Start Nagios
Add Nagios to the list of system services
sudo chkconfig --add nagios
sudo chkconfig nagios on

```
[ec2-user@ip-172-31-91-91 nagios-plugins-2.4.11]$ sudo chkconfig --add nagio
s
sudo chkconfig nagios on
Note: Forwarding request to 'systemctl enable nagios.service'.
Synchronizing state of nagios.service with SysV service script with /usr/lib
/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable nagios
Created symlink /etc/systemd/system/multi-user.target.wants/nagios.service →
 /usr/lib/systemd/system/nagios.service.
[ec2-user@ip-172-31-91-91 nagios-plugins-2.4.11]$
```

Verify the sample configuration files
sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

Error

```
[ec2-user@ip-172-31-91-91 nagios-plugins-2.0.3]$ sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

Nagios Core 4.0.8
Copyright (c) 2009-present Nagios Core Development Team and Community Contributors
Copyright (c) 1999-2009 Ethan Galstad
Last Modified: 08-12-2014
License: GPL

Website: http://www.nagios.org
Reading configuration data...
Error in configuration file '/usr/local/nagios/etc/nagios.cfg' - Line 452 (Check result path '/usr/local/nagios/var/spool/checkres
lts' is not a valid directory)
   Error processing main config file!
```

Solution:

 # Create the missing directory: If the directory is missing, create it with the necessary permissions:

sudo mkdir -p /usr/local/nagios/var/spool/checkresults

sudo chown nagios:nagios /usr/local/nagios/var/spool/checkresults

sudo chmod 775 /usr/local/nagios/var/spool/checkresults

```
[ec2-user@ip-172-31-91-91 nagios-plugins-2.0.3]$ sudo mkdir -p /usr/local/nagios/var/spool/checkresults
sudo chown nagios:nagios /usr/local/nagios/var/spool/checkresults
sudo chmod 775 /usr/local/nagios/var/spool/checkresults
```

Now run again

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

```
[ec2-user@ip-172-31-91-91 nagios-plugins-2.4.11]$ sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

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

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

Running pre-flight check on configuration data...

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

Total Warnings: 0
Total Errors:   0
```

sudo service nagios start

```
[ec2-user@ip-172-31-91-91 nagios-plugins-2.4.11]$ sudo service nagios start
Starting nagios (via systemctl):                           [  OK  ]
```

21. Check the status of Nagios

sudo systemctl status nagios

```
[ec2-user@ip-172-31-91-91 nagios-plugins-2.0.3]$ sudo systemctl status nagios
● nagios.service - LSB: Starts and stops the Nagios monitoring server
     Loaded: loaded (/etc/rc.d/init.d/nagios; generated)
     Active: active (running) since Sun 2024-09-29 08:04:30 UTC; 37s ago
       Docs: man:systemd-sysv-generator(8)
    Process: 68037 ExecStart=/etc/rc.d/init.d/nagios start (code=exited, status=0/SUCCESS)
      Tasks: 6 (limit: 1112)
     Memory: 2.0M
        CPU: 47ms
     CGroup: /system.slice/nagios.service
             ├─68059 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─68061 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─68062 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─68063 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─68064 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             └─68065 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg

Sep 29 08:04:30 ip-172-31-91-91.ec2.internal nagios[68059]: wproc: Registry request: name=Core Worker 68063;pid=68063
Sep 29 08:04:30 ip-172-31-91-91.ec2.internal nagios[68059]: wproc: Registry request: name=Core Worker 68062;pid=68062
Sep 29 08:04:30 ip-172-31-91-91.ec2.internal nagios[68059]: wproc: Registry request: name=Core Worker 68064;pid=68064
Sep 29 08:04:30 ip-172-31-91-91.ec2.internal nagios[68059]: wproc: Registry request: name=Core Worker 68061;pid=68061
Sep 29 08:04:30 ip-172-31-91-91.ec2.internal nagios[68059]: Warning: Could not open object cache file '/usr/local/nagios/var/objec>
Sep 29 08:04:30 ip-172-31-91-91.ec2.internal nagios[68059]: Error: Unable to create temp file '/usr/local/nagios/var/nagios.tmpx2N>
Sep 29 08:04:30 ip-172-31-91-91.ec2.internal nagios[68059]: Successfully launched command file worker with pid 68065
Sep 29 08:04:39 ip-172-31-91-91.ec2.internal nagios[68059]: Error: Unable to create temp file '/usr/local/nagios/var/nagios.tmpTmg>
Sep 29 08:04:49 ip-172-31-91-91.ec2.internal nagios[68059]: Error: Unable to create temp file '/usr/local/nagios/var/nagios.tmpAfy>
Sep 29 08:04:59 ip-172-31-91-91.ec2.internal nagios[68059]: Error: Unable to create temp file '/usr/local/nagios/var/nagios.tmpCtQ>
lines 1-26/26 (END)
```

Error:

The log messages suggest that Nagios is unable to create temporary files, particularly in the directory `/usr/local/nagios/var/`. This is typically caused by permission issues, or the directory might not exist.

Solution:

Firstly check whether `/usr/local/nagios/var/` is there or not. If yes.....

`ls -ld /usr/local/nagios/var/`

Change ownership: Set the correct ownership for the Nagios user and group:

sudo chown -R nagios:nagcmd /usr/local/nagios/var

Set permissions: Ensure the directory has the right permissions:

sudo chmod -R 775 /usr/local/nagios/var

Restart Nagios: After adjusting the ownership and permissions, restart the Nagios service:

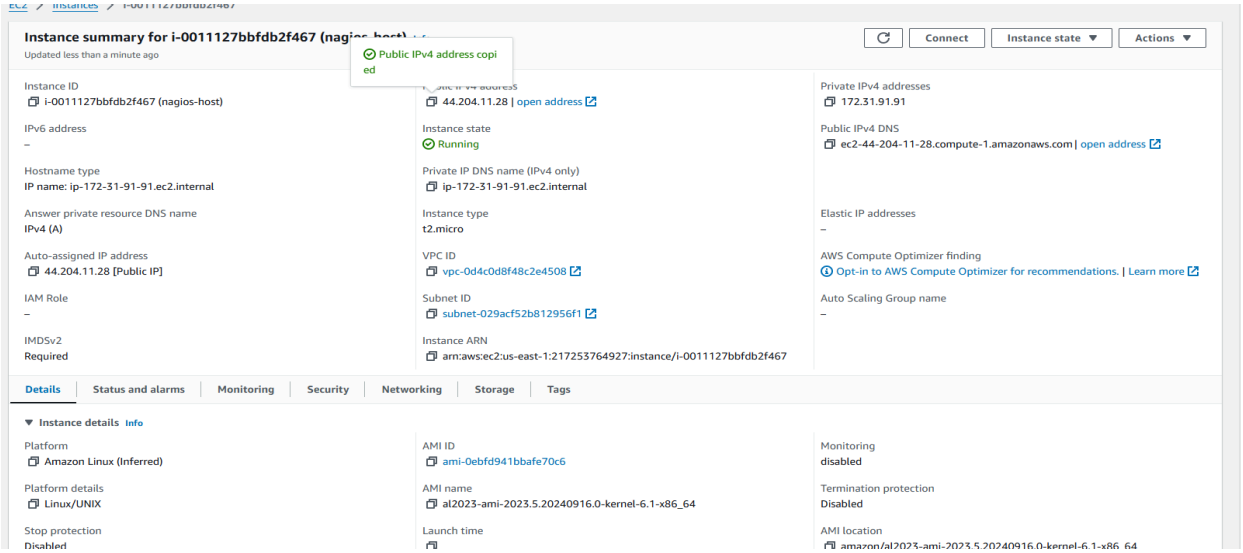
sudo systemctl restart nagios

```
drwxr-xr-x. 4 root root 112 Sep 29 08:04 /usr/local/nagios/var/
[ec2-user@ip-172-31-91-91 nagios-plugins-2.0.3]$ sudo chown -R nagios:nagcmd /usr/local/nagios/var
[ec2-user@ip-172-31-91-91 nagios-plugins-2.0.3]$ sudo chmod -R 775 /usr/local/nagios/var
[ec2-user@ip-172-31-91-91 nagios-plugins-2.0.3]$ sudo systemctl restart nagios
[ec2-user@ip-172-31-91-91 nagios-plugins-2.0.3]$
```

Now run again

```
[ec2-user@ip-172-31-91-91 nagios-plugins-2.4.11]$ sudo systemctl status nagios
● nagios.service - Nagios Core 4.5.5
     Loaded: loaded (/usr/lib/systemd/system/nagios.service; enabled; preset: disabled)
     Active: active (running) since Sun 2024-09-29 08:51:47 UTC; 42min ago
       Docs: https://www.nagios.org/documentation
      Tasks: 6 (limit: 1112)
     Memory: 2.9M
        CPU: 562ms
     CGroup: /system.slice/nagios.service
             ├─71188 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─71190 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─71191 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─71192 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─71193 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             └─71194 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg

Sep 29 08:51:47 ip-172-31-91-91.ec2.internal nagios[71188]: wproc: Registry request: name=Core Worker 71191;pid=71191
Sep 29 08:51:47 ip-172-31-91-91.ec2.internal nagios[71188]: wproc: Registry request: name=Core Worker 71190;pid=71190
Sep 29 08:51:47 ip-172-31-91-91.ec2.internal nagios[71188]: Successfully launched command file worker with pid 71194
Sep 29 08:59:22 ip-172-31-91-91.ec2.internal nagios[71188]: SERVICE ALERT: localhost;HTTP;WARNING;HARD;4;HTTP WARNING: HTTP/1.1 403 Forbidden - 319 bytes i>
Sep 29 09:11:52 ip-172-31-91-91.ec2.internal nagios[71188]: SERVICE NOTIFICATION: nagiosadmin;localhost;Swap Usage;CRITICAL;notify-service-by-email;SWAP CR>
Sep 29 09:11:52 ip-172-31-91-91.ec2.internal nagios[71188]: wproc: NOTIFY job 10 from worker Core Worker 71192 is a non-check helper but exited with return>
Sep 29 09:11:52 ip-172-31-91-91.ec2.internal nagios[71188]: wproc:   host=localhost; service=Swap Usage; contact=nagiosadmin
Sep 29 09:11:52 ip-172-31-91-91.ec2.internal nagios[71188]: wproc:   early_timeout=0; exited_ok=1; wait_status=32512; error_code=0;
Sep 29 09:11:52 ip-172-31-91-91.ec2.internal nagios[71188]: wproc:   stderr line 01: /bin/sh: line 1: /bin/mail: No such file or directory
Sep 29 09:11:52 ip-172-31-91-91.ec2.internal nagios[71188]: wproc:   stderr line 02: /usr/bin/printf: write error: Broken pipe
lines 1-25/25 (END)
```

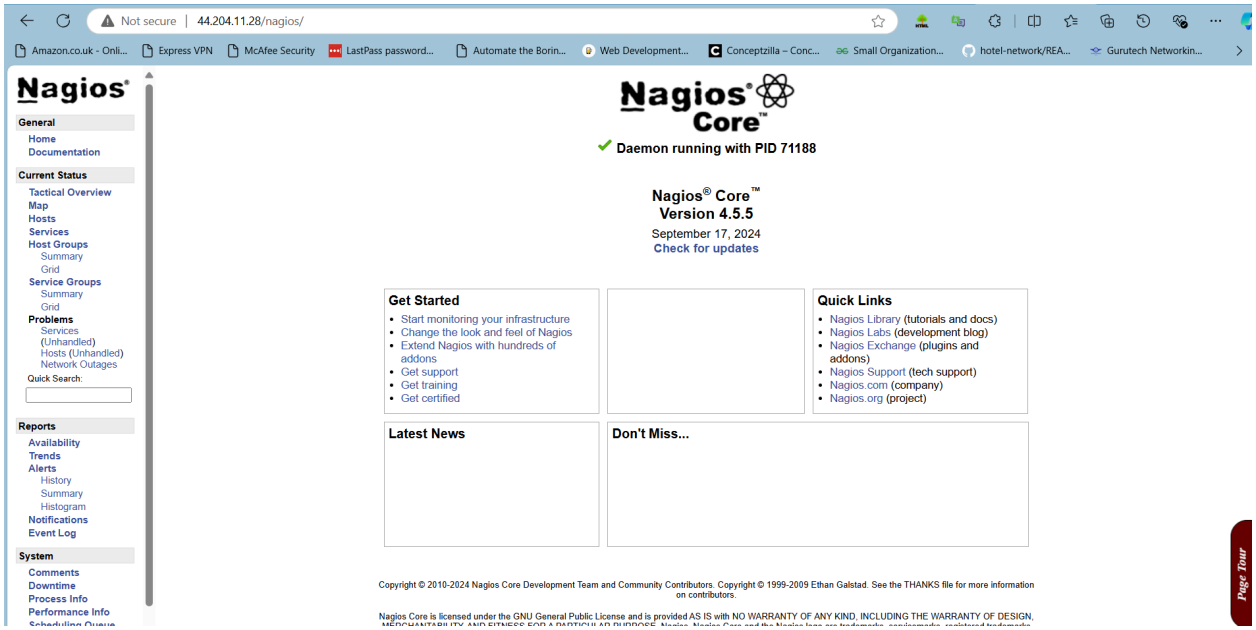
22. Go back to EC2 Console and copy the Public IP address of this instance



23. 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.

24. 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:

In this practical, we successfully installed and configured Nagios Core along with Nagios plugins and NRPE on an Amazon EC2 instance. We created a Nagios user, set up necessary permissions, and resolved common installation errors. Finally, we verified the setup by accessing the Nagios web interface, confirming that our monitoring system was fully operational.

Experiment 10

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

Theory:

Port and Service Monitoring

Port and service monitoring in Nagios involves checking the availability and responsiveness of network services running on specific ports. This ensures that critical services (like HTTP, FTP, or SSH) are operational. Nagios uses plugins to ping the ports and verify whether services are up and responding as expected, allowing administrators to be alerted in case of outages.

Windows/Linux Server Monitoring

Windows/Linux server monitoring with Nagios entails tracking the performance and health of servers running these operating systems. It includes monitoring metrics such as CPU usage, memory consumption, disk space, and system logs. Nagios employs various plugins to gather data, enabling administrators to ensure optimal performance, identify potential issues, and maintain uptime across their server infrastructure.

Prerequisites:

AWS Academy or Personal account.

Nagios Server running on Amazon Linux Machine. (Refer Experiment No 9)

Monitoring Using Nagios:

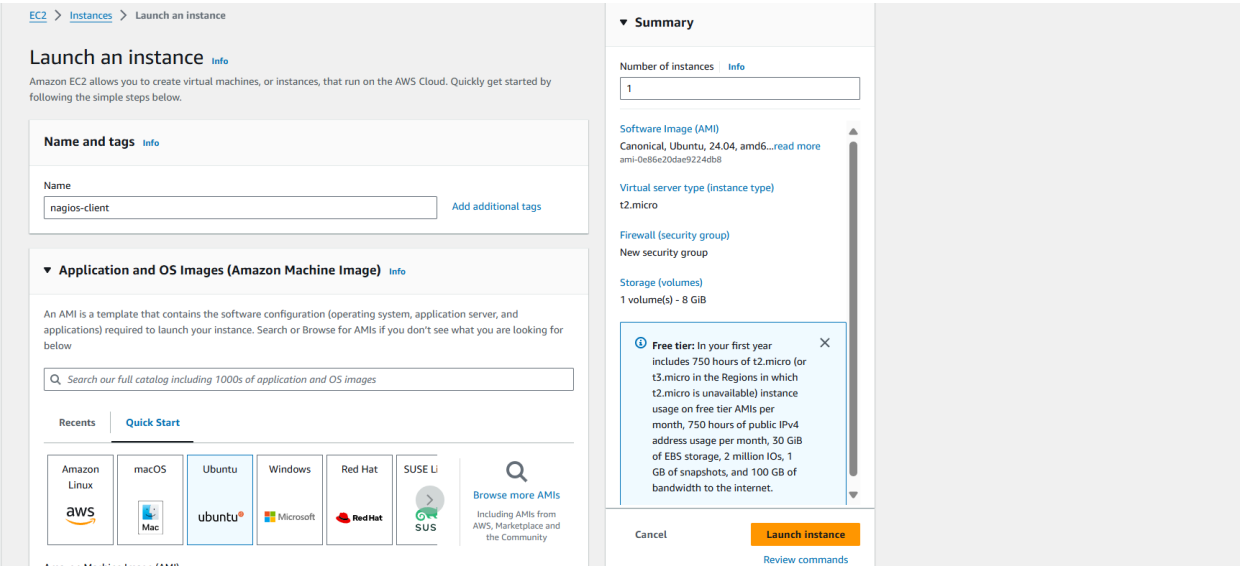
Step 1: To Confirm Nagios is running on the server side Perform the following command on your Amazon Linux Machine (Nagios-host).

sudo systemctl status nagios

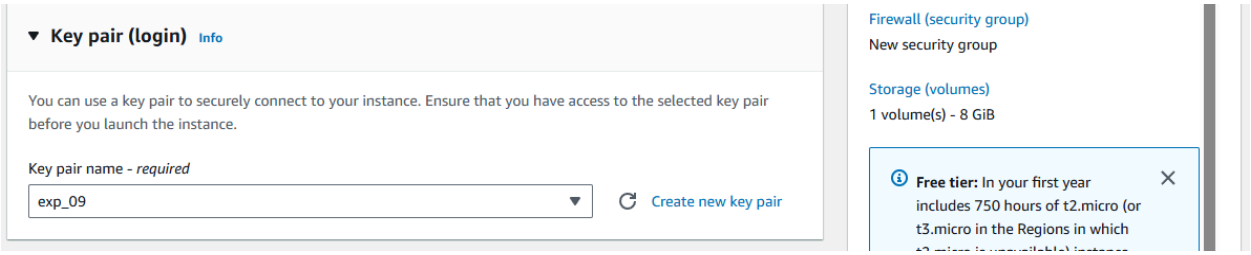
```
[ec2-user@ip-172-31-91-91 ~]$ sudo systemctl status nagios
● nagios.service - Nagios Core 4.5.5
     Loaded: loaded (/usr/lib/systemd/system/nagios.service; enabled; preset: disabled)
     Active: active (running) since Sun 2024-09-29 16:18:08 UTC; 21min ago
       Docs: https://www.nagios.org/documentation
    Process: 1942 ExecStartPre=/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
    Process: 1944 ExecStart=/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
   Main PID: 1946 (nagios)
      Tasks: 8 (limit: 1112)
     Memory: 7.7M
        CPU: 387ms
     CGroup: /system.slice/nagios.service
             ├─1946 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─1947 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─1948 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─1949 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─1950 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─1956 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─3088 /usr/local/nagios/libexec/check_ping -H 127.0.0.1 -w 3000.0,80% -c 5000.0,100% -p 5
             └─3089 /usr/bin/ping -n -U -w 30 -c 5 127.0.0.1

Sep 29 16:18:08 ip-172-31-91-91.ec2.internal systemd[1]: Starting nagios.service - Nagios Core 4.5.5...
Sep 29 16:18:08 ip-172-31-91-91.ec2.internal systemd[1]: Started nagios.service - Nagios Core 4.5.5.
Sep 29 16:20:00 ip-172-31-91-91.ec2.internal nagios[1946]: SERVICE FLAPPING ALERT: localhost;HTTP;STARTED; Service appears to have started flapping (20.0% >
Sep 29 16:20:00 ip-172-31-91-91.ec2.internal nagios[1946]: SERVICE ALERT: localhost;HTTP;CRITICAL;HARD;4;connect to address 127.0.0.1 and port 80: Connecti>
Sep 29 16:22:30 ip-172-31-91-91.ec2.internal nagios[1946]: SERVICE NOTIFICATION: nagiosadmin;localhost;Swap Usage;CRITICAL;notify-service-by-email;SWAP CRI>
Sep 29 16:22:30 ip-172-31-91-91.ec2.internal nagios[1946]: wproc: NOTIFY job 2 from worker Core Worker 1948 is a non-check helper but exited with return co>
Sep 29 16:22:30 ip-172-31-91-91.ec2.internal nagios[1946]: wproc:   host=localhost; service=Swap Usage; contact=nagiosadmin
Sep 29 16:22:30 ip-172-31-91-91.ec2.internal nagios[1946]: wproc:   early_timeout=0; exited_ok=1; wait_status=32512; error_code=0;
Sep 29 16:22:30 ip-172-31-91-91.ec2.internal nagios[1946]: wproc:   stderr line 01: /bin/sh: line 1: /bin/mail: No such file or directory
Sep 29 16:22:30 ip-172-31-91-91.ec2.internal nagios[1946]: wproc:   stderr line 02: /usr/bin/printf: write error: Broken pipe
lines 1-30/30 (END)
```

You can now proceed if you get the above message/output.

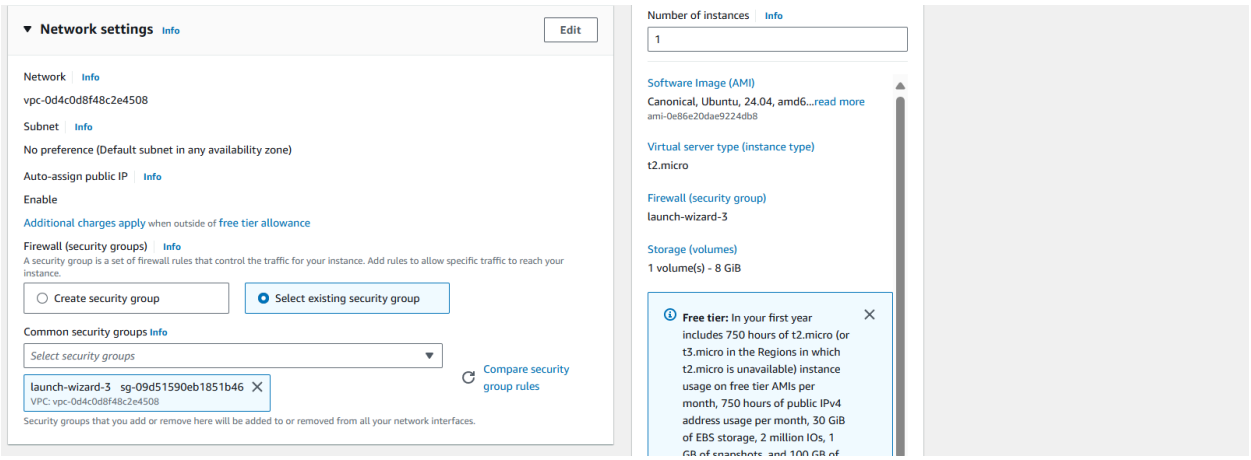
Step 2: Now Create a new EC2 instance. Name: Nagios-client, AMI: Ubuntu Instance Type: t2.micro.



For Key pair : Click on create key and make key of type RSA with extension .pem . Key will be downloaded to your local machine.
Now select that key in key pair if you already have key with type RSA and extension .pem no need to create new key but you must have that key downloaded.



Select the Existing Security Group and select the Security Group that we have created in Experiment no 9 or the same one you have used for the Nagios server (Nagios-host).



Step 3: Now After creating the EC2 Instance click on connect and then copy the command which is given as example in the SSH Client section .
Now open the terminal in the folder where your key(RSA key with .pem) is located. and paste that copied command.

```
PS C:\Users\Ayush Maurya> ssh -i "Downloads/exp_09.pem" ubuntu@ec2-44-206-245-149.compute-1.amazonaws.com
The authenticity of host 'ec2-44-206-245-149.compute-1.amazonaws.com (44.206.245.149)' can't be established.
ED25519 key fingerprint is SHA256:DT+AA+mkcydh3kOJ2vEpm4ZsA6FL+LM4m1QSImddAHg.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-44-206-245-149.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

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

```
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-92-146:~$
```

Now perform all the commands on the Nagios-host till step 10

Step 4: Now on the server Nagios-host run the following command.

ps -ef | grep nagios

```
[ec2-user@ip-172-31-91-91 ~]$ ps -ef | grep nagios
nagios      1946       1  0 16:18 ?        00:00:00 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
nagios      1947    1946  0 16:18 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios      1948    1946  0 16:18 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios      1949    1946  0 16:18 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios      1950    1946  0 16:18 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios      1956    1946  0 16:18 ?        00:00:00 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
root        3090    3055  0 16:40 pts/0    00:00:00 sudo systemctl status nagios
root        3092    3090  0 16:40 pts/1    00:00:00 sudo systemctl status nagios
root        3093    3092  0 16:40 pts/1    00:00:00 systemctl status nagios
ec2-user    3914    3890  0 16:59 pts/2    00:00:00 grep --color=auto nagios
[ec2-user@ip-172-31-91-91 ~]$
```

Step 5: Now Become root user and create root directories.

sudo su

mkdir /usr/local/nagios/etc/objects/monitorhosts

mkdir /usr/local/nagios/etc/objects/monitorhosts/linuxhosts

```
[ec2-user@ip-172-31-91-91 ~]$ sudo su
[root@ip-172-31-91-91 ec2-user]# mkdir /usr/local/nagios/etc/objects/monitorhosts mkdir /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
[root@ip-172-31-91-91 ec2-user]#
```

Step 6: Copy the sample localhost.cfg to linuxhost.cfg by running the following command.(Below command should come in one line see screenshot below)

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

```
[root@ip-172-31-91-91 ec2-user]# cp /usr/local/nagios/etc/objects/localhost.cfg /usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg
[root@ip-172-31-91-91 ec2-user]#
```

Step 7: Open linuxserver.cfg using nano and make the following changes in all positions?everywhere in file.

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

Change hostname to **linuxserver.**

Change address to the **public IP of your Linux client**.

Set hostgroup_name to **linux-servers1.**

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

# Define a host for the local machine

define host {

    use                     linux-server            ; Name of host template to use
                                                    ; This host definition will inherit all variables that are defined
                                                    ; in (or inherited by) the linux-server host template definition.
    host_name               linuxserver
    alias                   localhost
    address                 172.31.92.146
}



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

# Define an optional hostgroup for Linux machines

define hostgroup {

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

Step 8: Now update the Nagios config file .Add the following line in the file. Line to add :
> nano /usr/local/nagios/etc/nagios.cfg

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

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


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

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



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

# You can specify individual object config files as shown below:
cfg_file=/usr/local/nagios/etc/objects/commands.cfg
cfg_file=/usr/local/nagios/etc/objects/contacts.cfg
cfg_file=/usr/local/nagios/etc/objects/timeperiods.cfg
cfg_file=/usr/local/nagios/etc/objects/templates.cfg
cfg_dir=/usr/local/nagios/etc/objects/monitorhosts/
# Definitions for monitoring the local (Linux) host
cfg_file=/usr/local/nagios/etc/objects/localhost.cfg
```

Step 9: Now Verify the configuration files by running the following commands.
/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

```
Things look okay - No serious problems were detected during the pre-flight check
[root@ip-172-31-91-91 ec2-user]# /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

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

```
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
```

Step 10: Now restart the services of nagios by running the following command.
service nagios restart

```
Things look okay - No serious problems were detected during the pre-flight check
[root@ip-172-31-91-91 ec2-user]# service nagios restart
Restarting nagios (via systemctl):                         [  OK  ]
[root@ip-172-31-91-91 ec2-user]#
```

Step 11: Now Go to the Nagios-client ssh terminal and update and install the packages by running the following command.
sudo apt update -y
sudo apt install gcc -y
sudo apt install -y nagios-nrpe-server nagios-plugins

```
ubuntu@ip-172-31-92-146:~$ 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://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:7 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [380 kB]
Get:8 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [82.9 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:10 http://security.ubuntu.com/ubuntu noble-security/main amd64 c-n-f Metadata [4560 B]
Get:11 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [272 kB]
Get:12 http://security.ubuntu.com/ubuntu noble-security/universe Translation-en [115 kB]
Get:13 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [8632 B]
Get:14 http://security.ubuntu.com/ubuntu noble-security/universe amd64 c-n-f Metadata [10.3 kB]
Get:15 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Packages [353 kB]
Get:16 http://security.ubuntu.com/ubuntu noble-security/restricted Translation-en [68.1 kB]
Get:17 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 c-n-f Metadata [428 B]
Get:18 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Packages [10.9 kB]
Get:19 http://security.ubuntu.com/ubuntu noble-security/multiverse Translation-en [2808 B]
Get:20 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [208 B]
Get:21 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 c-n-f Metadata [344 B]
Get:22 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
Get:23 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
Get:24 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
```

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

No containers need to be restarted.

User sessions running outdated binaries:
 ubuntu @ session #2: sshd[992,1102]
 ubuntu @ session #7: sshd[1190,1248]
ubuntu@ip-172-31-92-146:~$
```

Step 12: Open nrpe.cfg file to make changes.Under allowed_hosts, add your nagios host IP address.

sudo nano /etc/nagios/nrpe.cfg

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

nrpe_group=nagios



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

allowed_hosts=127.0.0.1,::1,34.207.68.187



# COMMAND ARGUMENT PROCESSING
# This option determines whether or not the NRPE daemon will allow clients
# to specify arguments to commands that are executed.  This option only works
# if the daemon was configured with the --enable-command-args configure script
# option.
#
# *** ENABLING THIS OPTION IS A SECURITY RISK! ***
```

Step 13: Now restart the NRPE server by this command.

sudo systemctl restart nagios-nrpe-server

```
0 upgraded, 0 newly installed, 0 to remove and 139 not upgraded.
ubuntu@ip-172-31-92-146:~$ sudo nano /etc/nagios/nrpe.cfg
ubuntu@ip-172-31-92-146:~$ sudo systemctl restart nagios-nrpe-server
ubuntu@ip-172-31-92-146:~$
```

Step 14: Now again check the status of Nagios by running this command on Nagios-host and also check httpd is active and run the command to active it.

sudo systemctl status nagios

```
[ec2-user@ip-172-31-91-91 ~]$ sudo systemctl status nagios
● nagios.service - Nagios Core 4.5.5
     Loaded: loaded (/usr/lib/systemd/system/nagios.service; enabled; preset: disabled)
     Active: active (running) since Sun 2024-09-29 17:20:07 UTC; 12min ago
       Docs: https://www.nagios.org/documentation
    Process: 4761 ExecStartPre=/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
    Process: 4762 ExecStart=/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
   Main PID: 4763 (nagios)
      Tasks: 6 (limit: 1112)
     Memory: 4.1M
        CPU: 234ms
     CGroup: /system.slice/nagios.service
             ├─4763 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─4764 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─4765 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─4766 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─4767 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             └─4768 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg

Sep 29 17:20:07 ip-172-31-91-91.ec2.internal nagios[4763]: Warning: Duplicate definition found for service 'Current Users' on host 'localhost' (config file
Sep 29 17:20:07 ip-172-31-91-91.ec2.internal nagios[4763]: Warning: Duplicate definition found for service 'Root Partition' on host 'localhost' (config fil
Sep 29 17:20:07 ip-172-31-91-91.ec2.internal nagios[4763]: Warning: Duplicate definition found for service 'PING' on host 'localhost' (config file '/usr/lo
Sep 29 17:20:07 ip-172-31-91-91.ec2.internal nagios[4763]: Successfully launched command file worker with pid 4768
Sep 29 17:22:30 ip-172-31-91-91.ec2.internal nagios[4763]: SERVICE NOTIFICATION: nagiosadmin;localhost;Swap Usage;CRITICAL;notify-service-by-email;SWAP CRI
Sep 29 17:22:30 ip-172-31-91-91.ec2.internal nagios[4763]: wproc: NOTIFY job 1 from worker Core Worker 4766 is a non-check helper but exited with return co
Sep 29 17:22:30 ip-172-31-91-91.ec2.internal nagios[4763]: wproc:   host=localhost; service=Swap Usage; contact=nagiosadmin
Sep 29 17:22:30 ip-172-31-91-91.ec2.internal nagios[4763]: wproc:   early_timeout=0; exited_ok=1; wait_status=32512; error_code=0;
Sep 29 17:22:30 ip-172-31-91-91.ec2.internal nagios[4763]: wproc:   stderr line 01: /bin/sh: line 1: /bin/mail: No such file or directory
Sep 29 17:22:30 ip-172-31-91-91.ec2.internal nagios[4763]: wproc:   stderr line 02: /usr/bin/printf: write error: Broken pipe
lines 1-28/28 (END)
```

sudo systemctl status httpd

```
[ec2-user@ip-172-31-91-91 ~]$ sudo systemctl status httpd
○ httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset: disabled)
    Drop-In: /usr/lib/systemd/system/httpd.service.d
             └─php-fpm.conf
     Active: inactive (dead)
       Docs: man:httpd.service(8)
[ec2-user@ip-172-31-91-91 ~]$
```

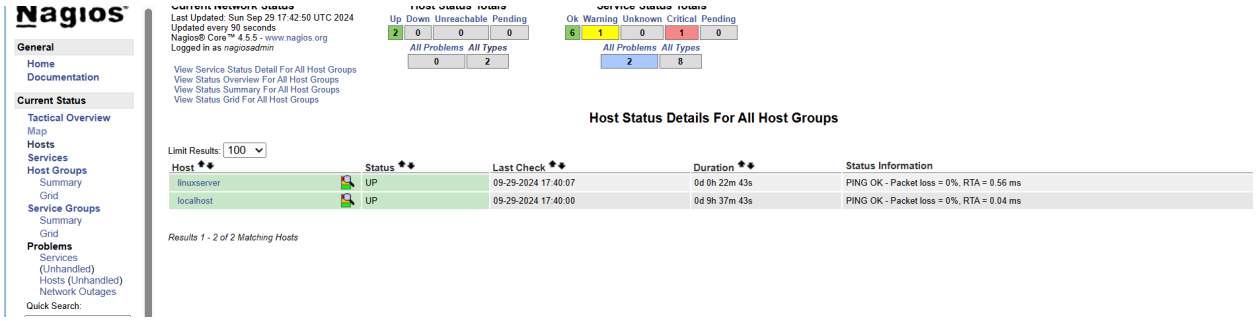
sudo systemctl start httpd
sudo systemctl enable httpd

```
[ec2-user@ip-172-31-91-91 ~]$ sudo systemctl start httpd
[ec2-user@ip-172-31-91-91 ~]$ sudo systemctl enable httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/lib/systemd/system/httpd.service.
[ec2-user@ip-172-31-91-91 ~]$
```

Step 15: Now to check Nagios dashboard go to http://<nagios host ip>/nagios
Eg. http://34.207.68.187/nagios

Enter username as nagiosadmin and password which you set in Exp 9.

Now Click on Hosts from left side panel



Conclusion:

In this practical, we set up a Nagios host and client to monitor services and server performance on both Linux and Windows servers. We configured Nagios on an Amazon Linux machine to monitor critical services like HTTP, SSH, and system resources, ensuring their availability and health. By creating and configuring a new EC2 instance as the Nagios client, we enabled seamless communication between the client and host for efficient service monitoring. This setup helps ensure uptime and quick detection of issues across the infrastructure.

Experiment 11

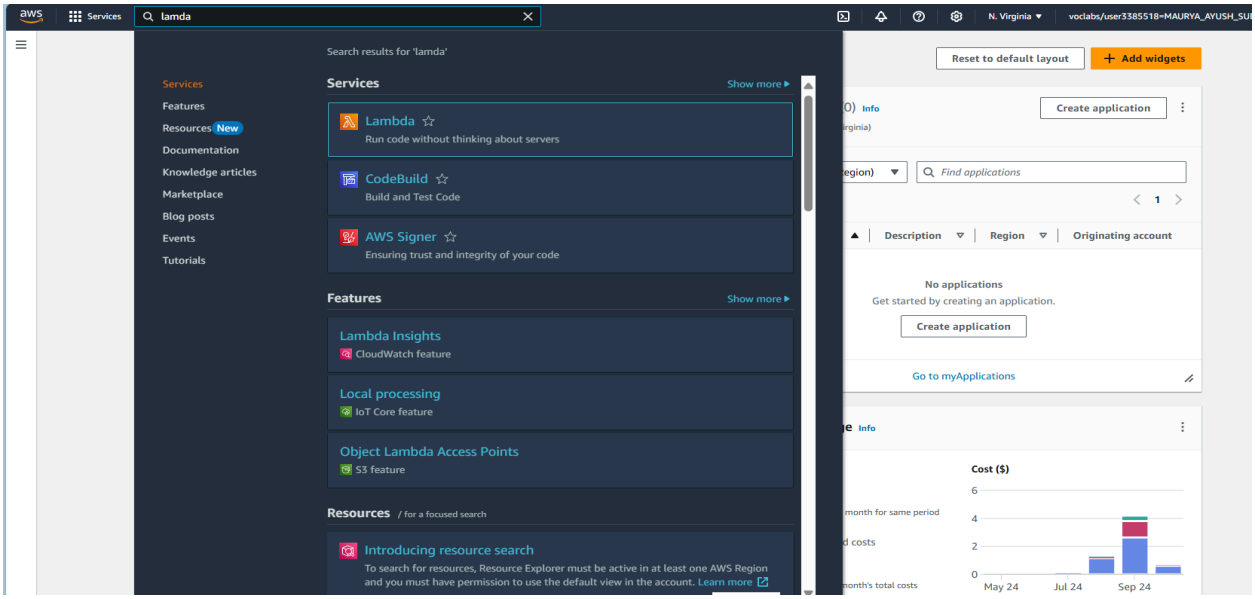
Aim: To understand AWS Lambda, its workflow, various functions and create your first Lambda functions using Python / Java / Nodejs.

Prerequisites:

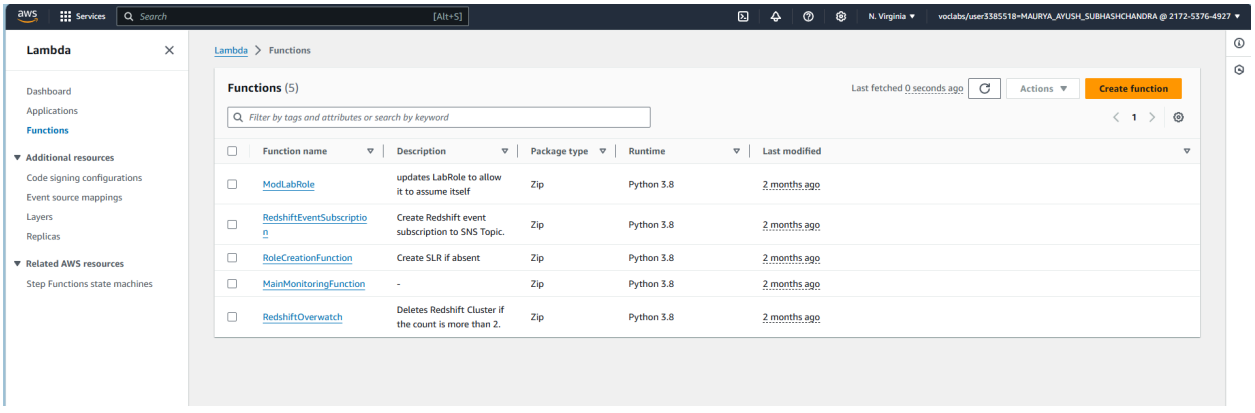
1) AWS account (academy recommended)

Step 1: Set up AWS Lambda Function

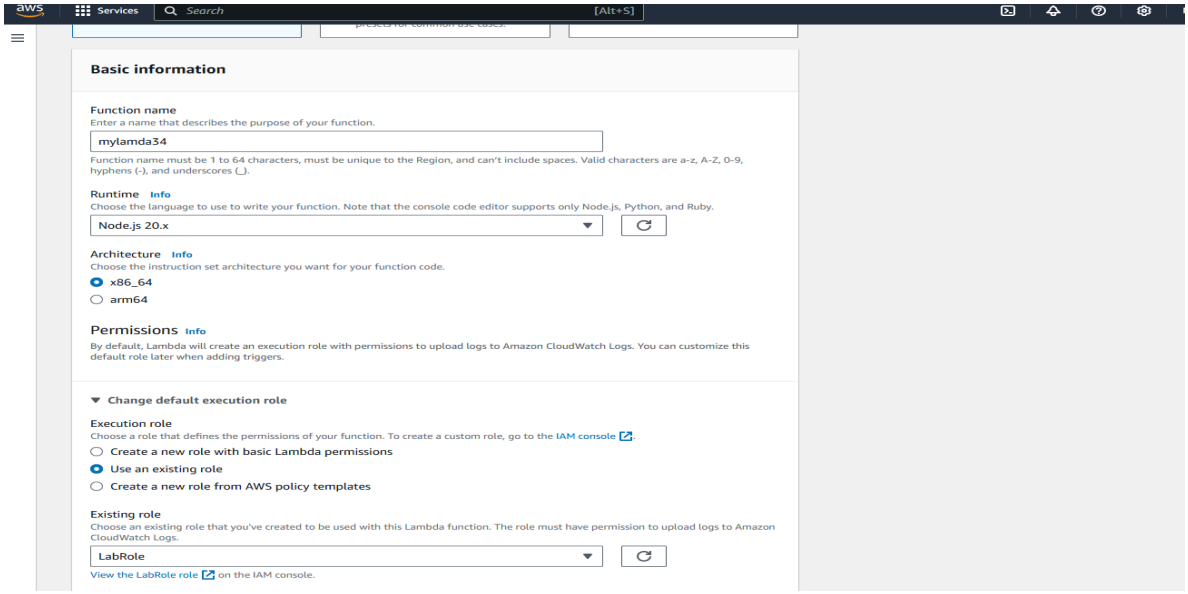
1) Search for Lambda in the services tab. Click on it once found.



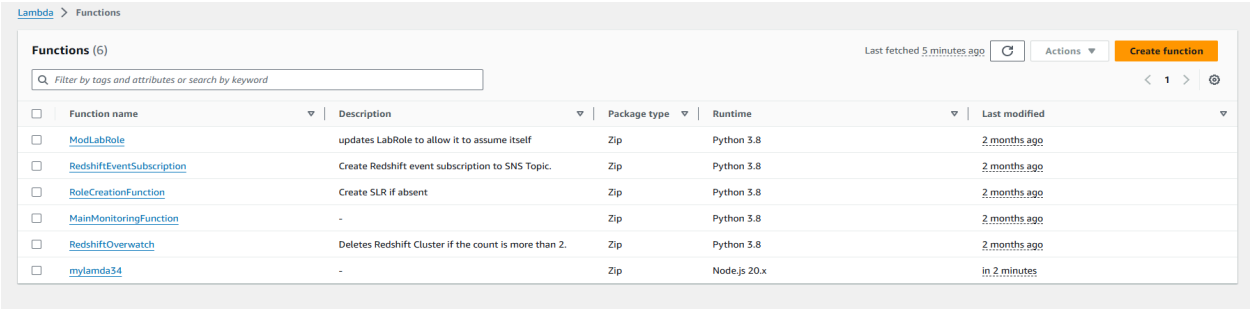
2) Click on create functions.



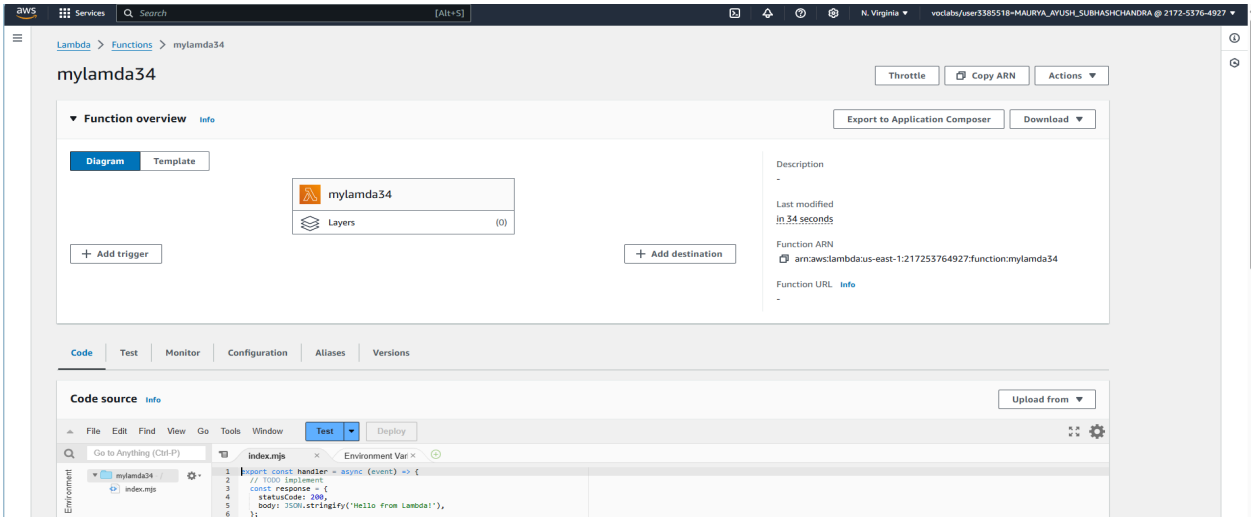
3) Give a name to your Lambda function. Select the runtime as Node.js 20.x (You can also use python). Select the architecture as x86_64. Set the default execution role as LabRole if you are doing this on your academy account. (Use an existing role → LabRole)



4) Once the function is created, click on the name of the function.



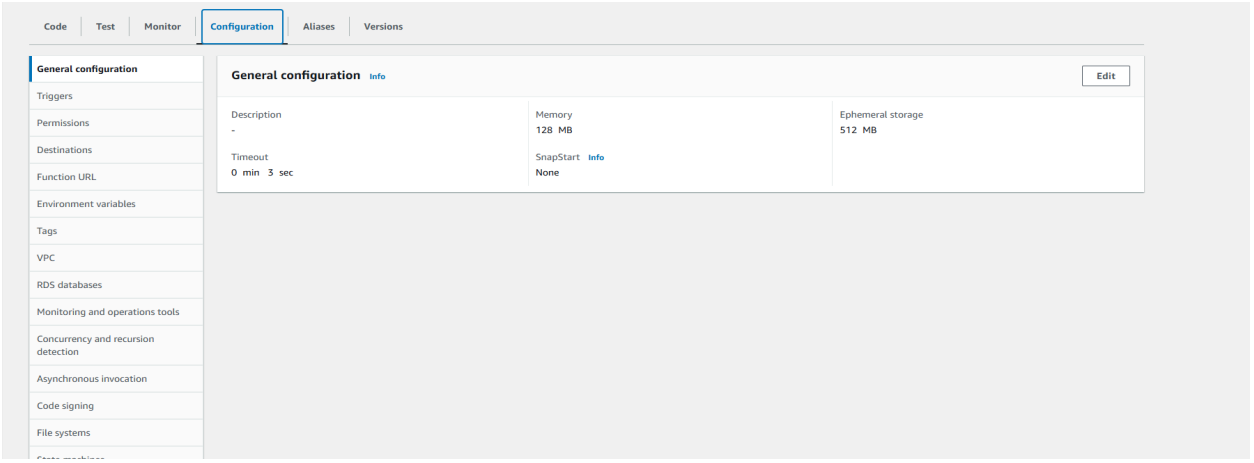
5) This is the dashboard of our lambda function.



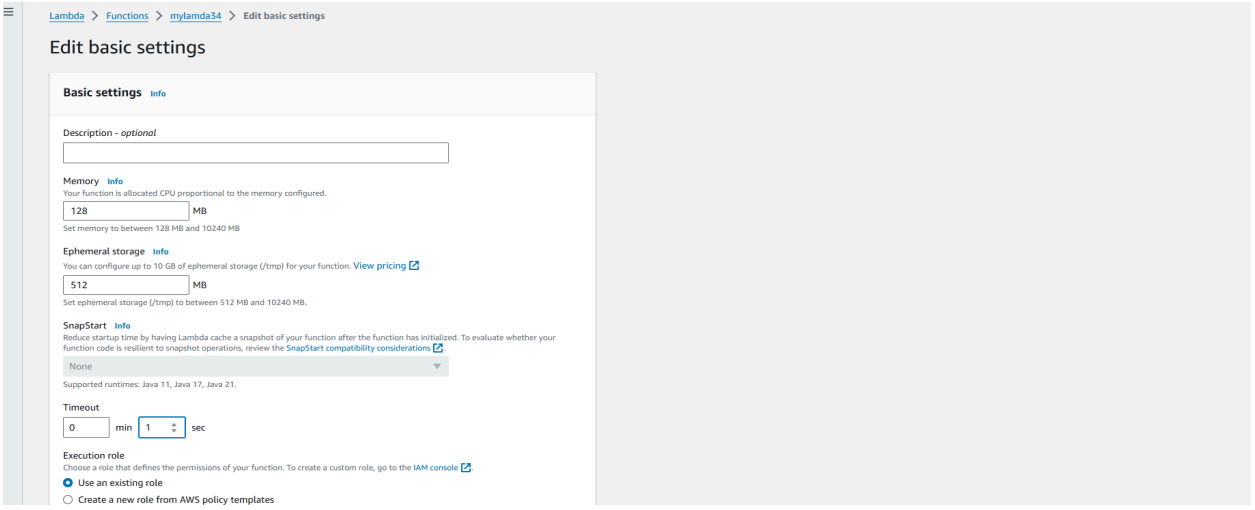
6) This function has the following default code, which is used to print “Hello form Lambda!”.

Step 2: Set up configurations and test events

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

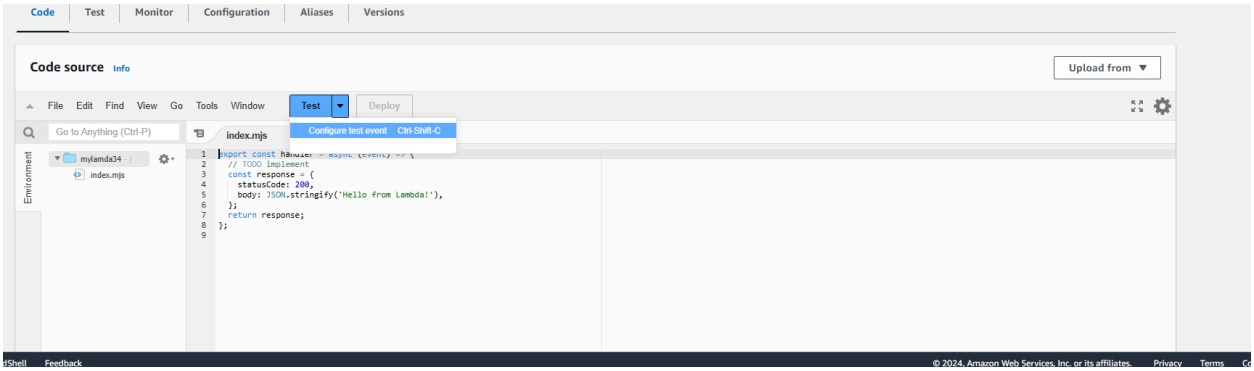


2) Here, change the Timeout to 1 sec. This is the time for which the function can be running before it is forcibly terminated.

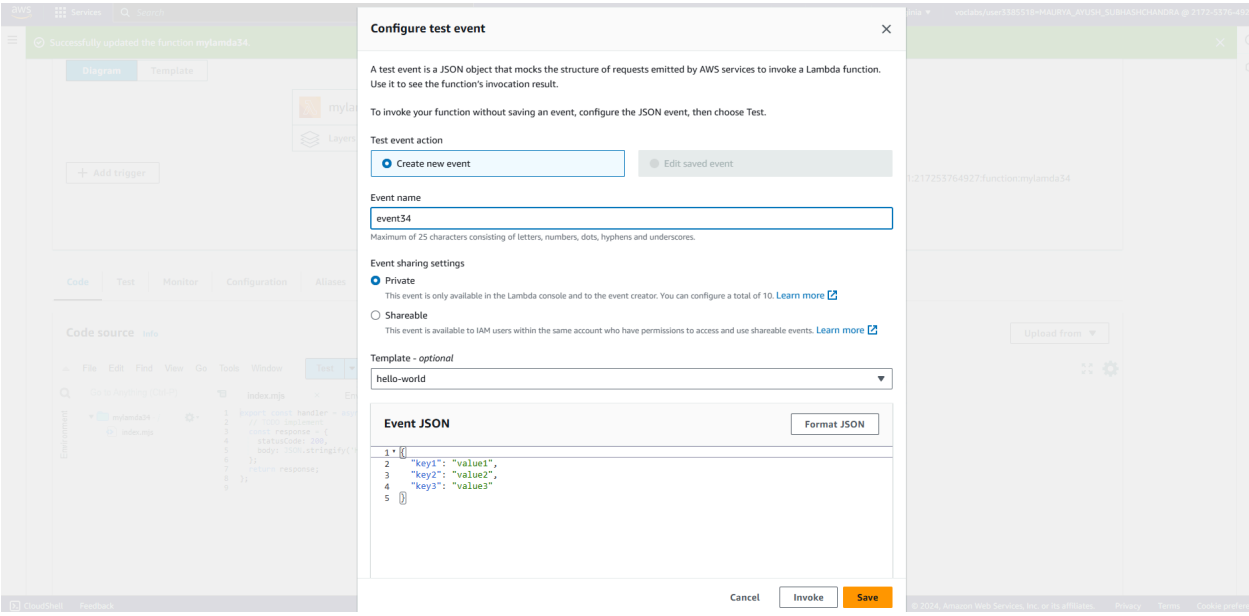


3) We can see the executed changes.

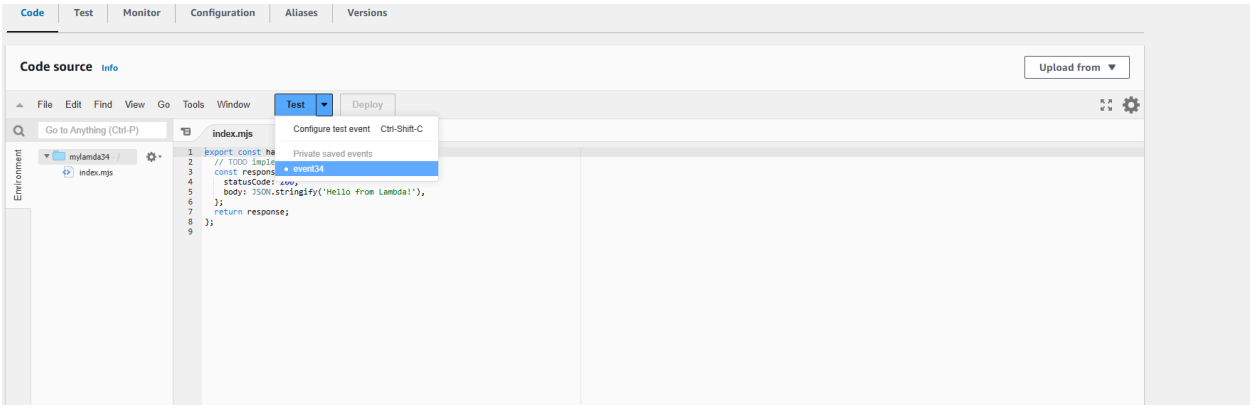
4) Switch back to the code tab. Click on the dropdown arrow near test. Then select configure test event.



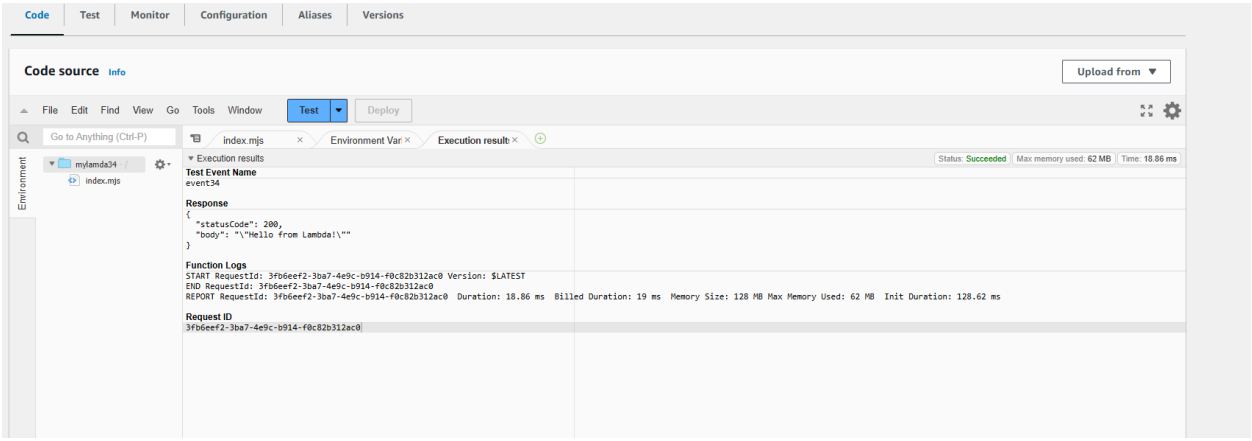
5) Here, create a new event, keep the other options default and save the event.



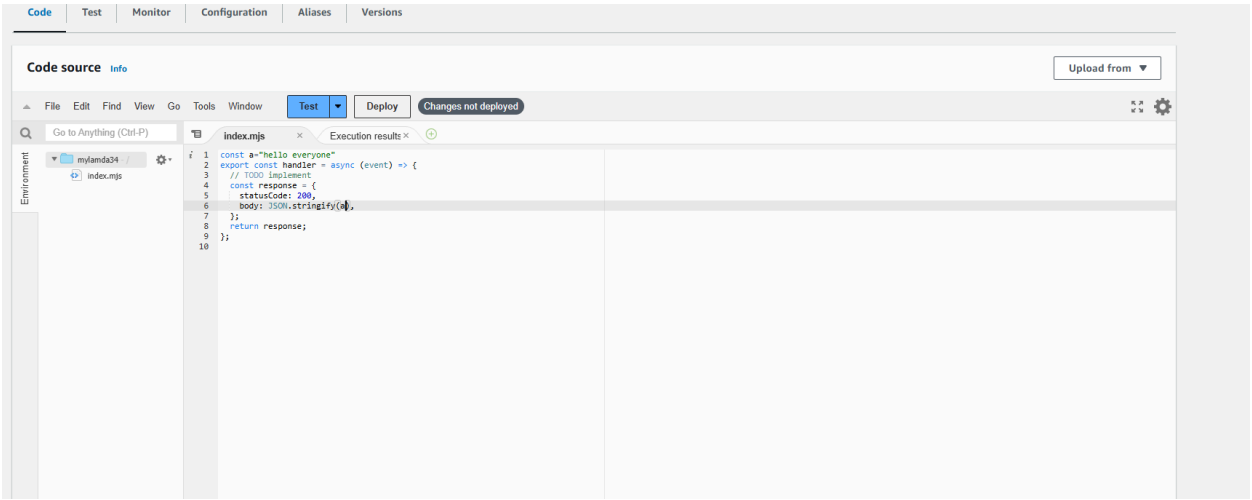
6) Now, again click on the dropdown. This time, select the event you have created. Then, click on TEST.



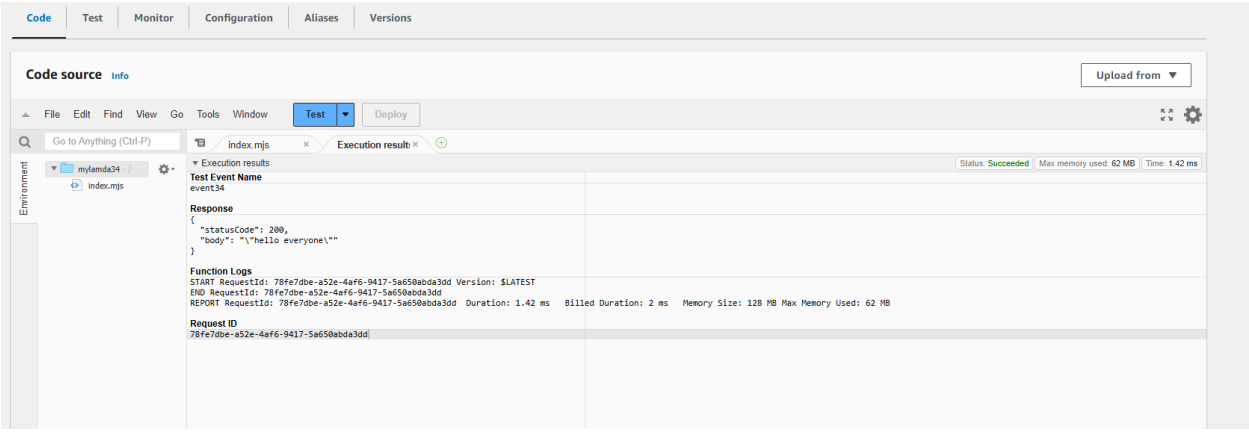
7) We can see the expected output for the sample code.



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



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



Conclusion:

In this experiment, we explored AWS Lambda by creating and configuring Lambda functions using Node.js. We learned to set up a function, adjust configurations like timeout settings, and test it with custom events. This hands-on experience provided us with foundational skills in serverless computing, enabling us to develop scalable applications efficiently. Moving forward, we can investigate integrating AWS Lambda with other services to enhance our serverless applications.

Experiment 12

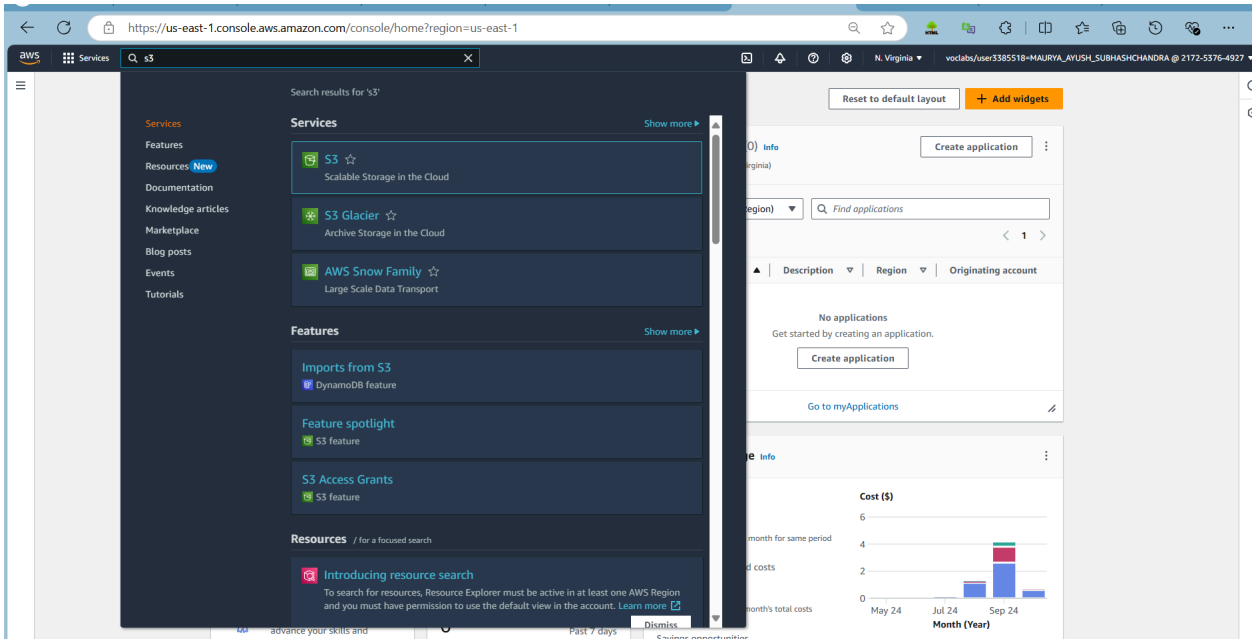
Aim: To create a Lambda function which will log “An Image has been added” once you add an object to a specific bucket in S3

Prerequisites:

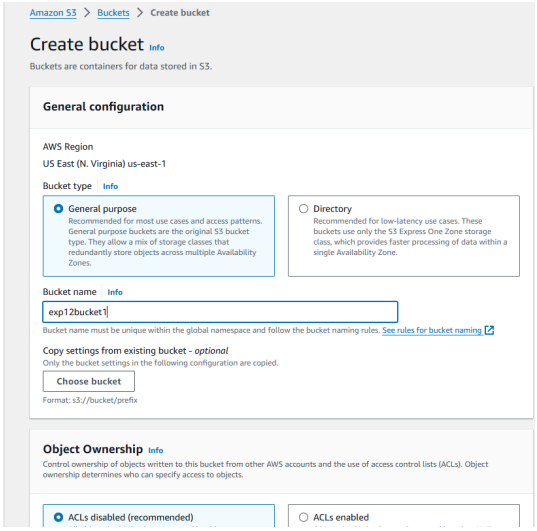
1) AWS account (academy preferable)
2) Lambda function (created in the previous experiment).

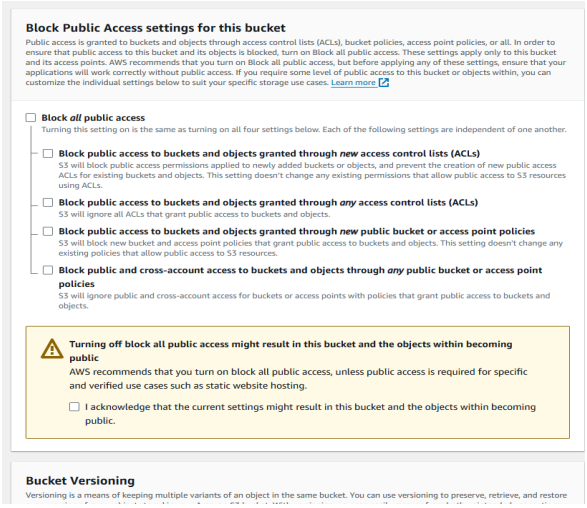
Step 1: Create a s3 bucket.

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

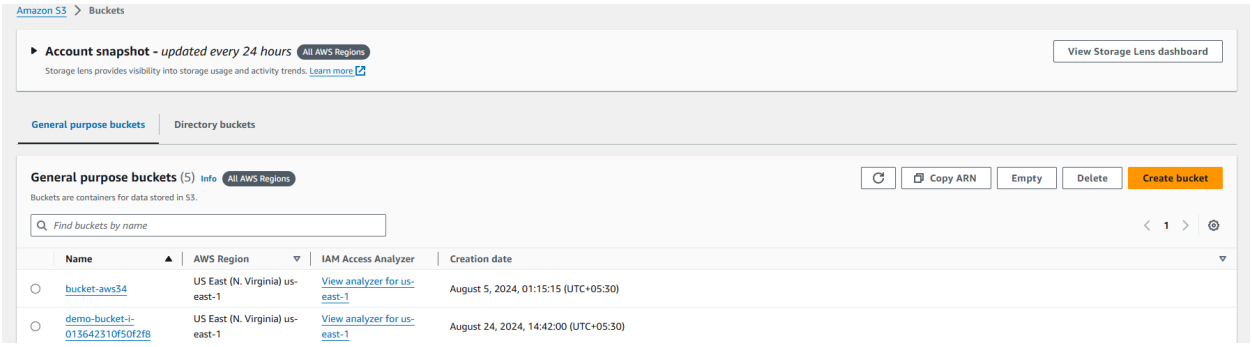


2) Keep the bucket as a general purpose bucket. Give a name to your bucket. 3) Uncheck block all public access.

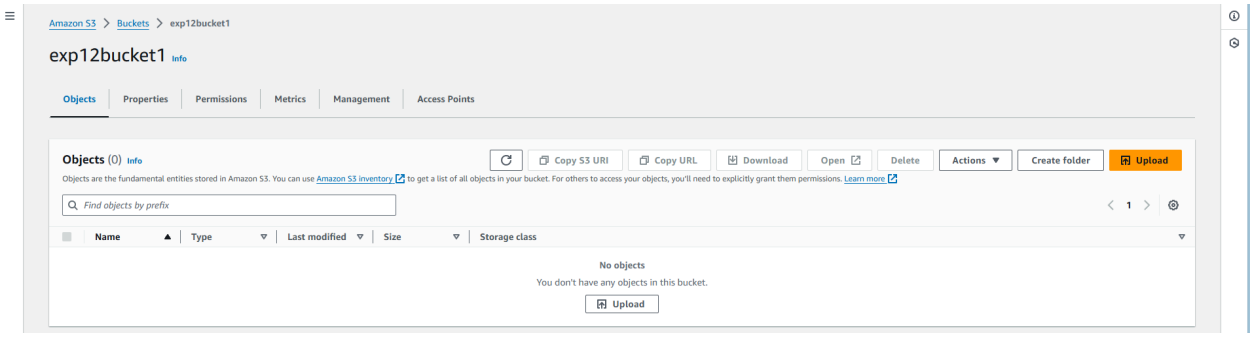


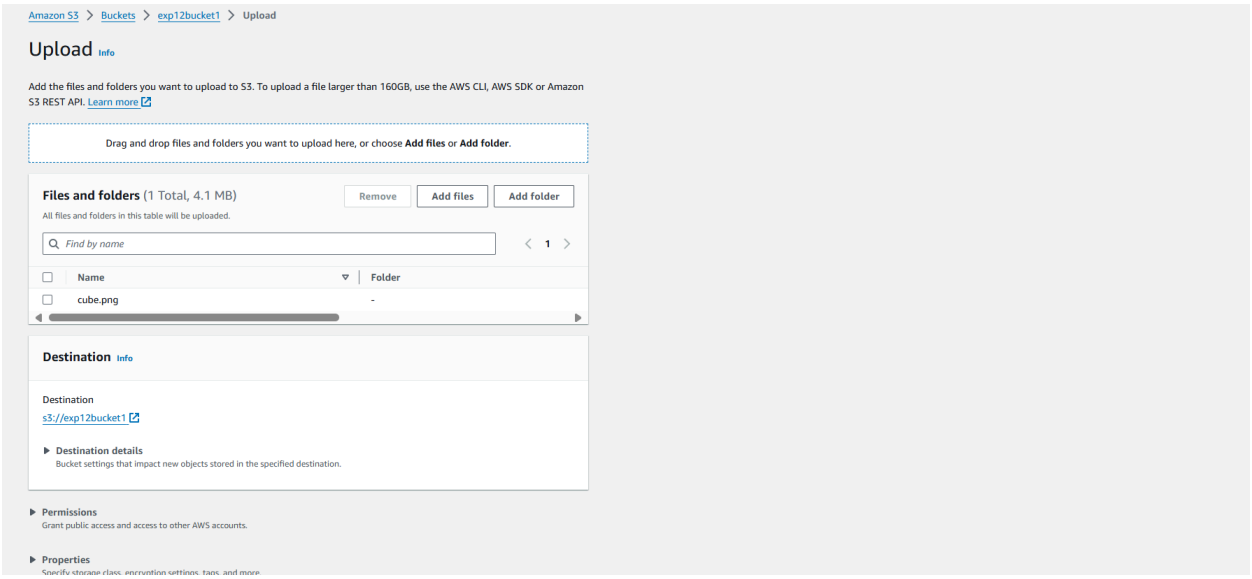


3) Keeping all other options the same, click on create. This would create your bucket. Now click on the name of the bucket.

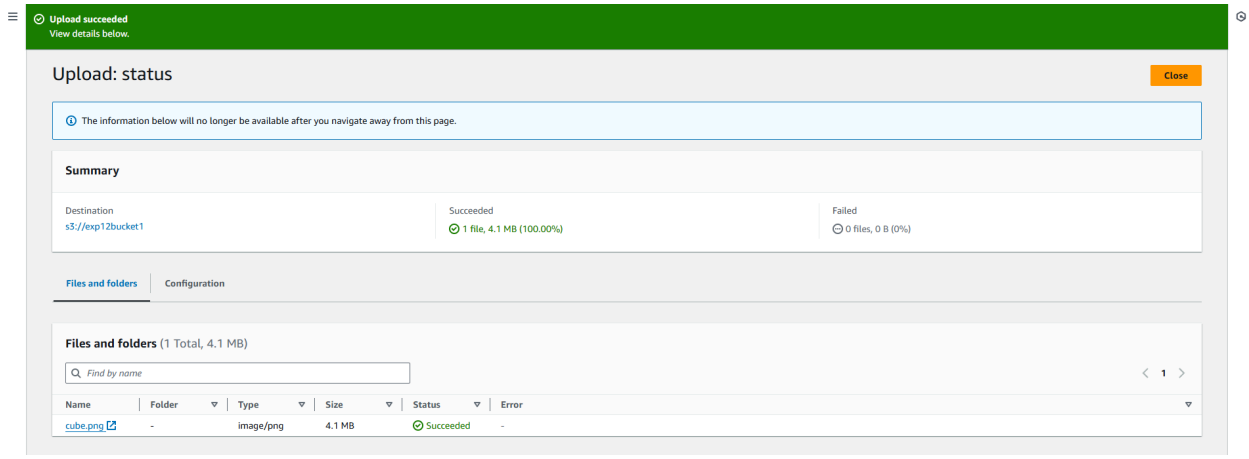


4) Here, click on upload, then add files. Select any image that you want to upload in the bucket and click on upload.



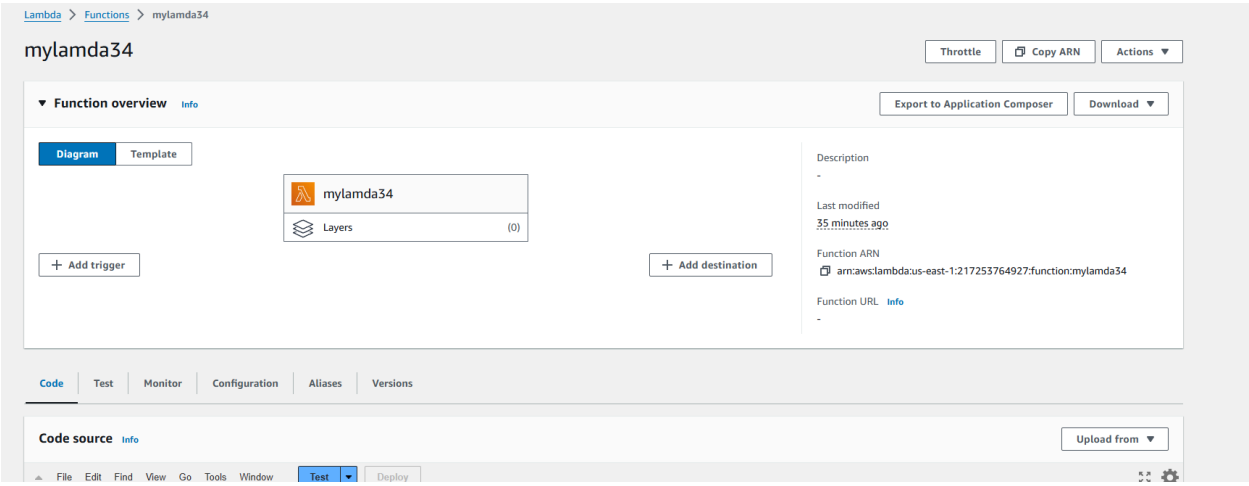


5) The image has been uploaded to the bucket.

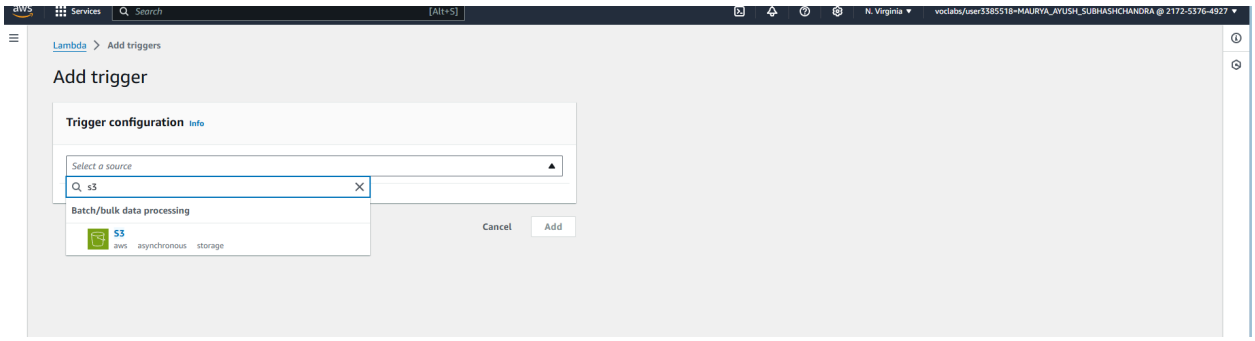


Step 2: Configure Lambda function

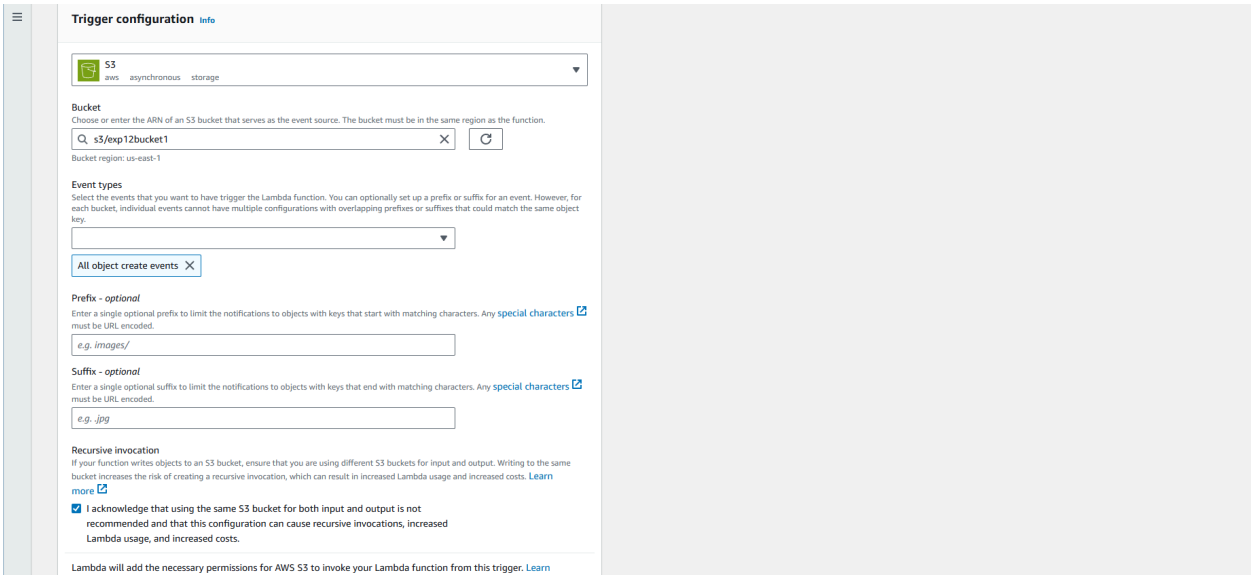
1) Go to the lambda function you had created before. (Services → Lambda → Click on name of function). Here, click on add trigger.

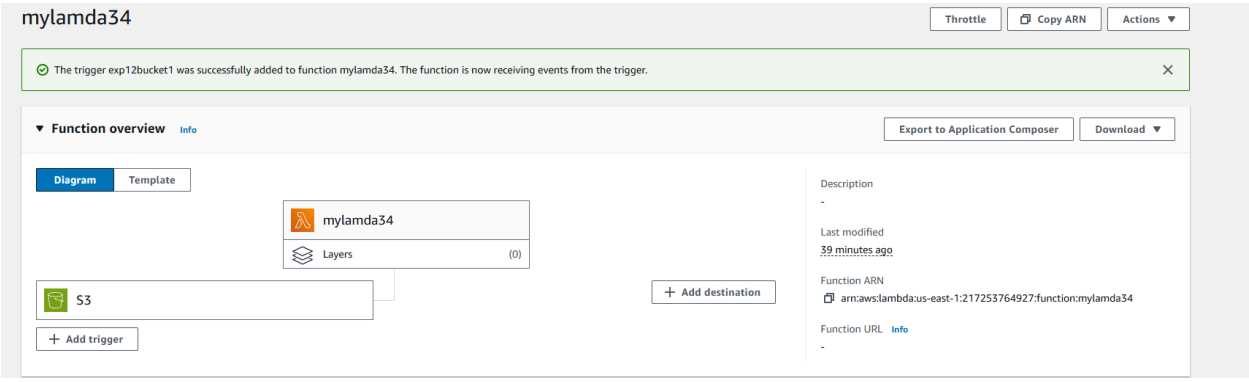


2) Under trigger configuration, search for S3 and select it.



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



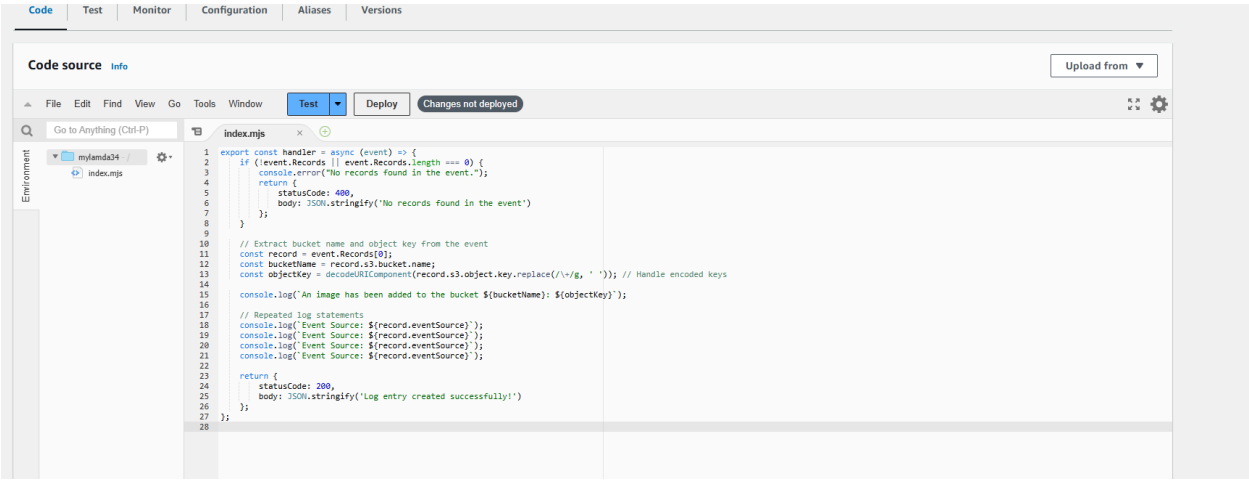


4) Scroll down to the code section of the function. Add the following javascript code to the code area by replacing the existing code.

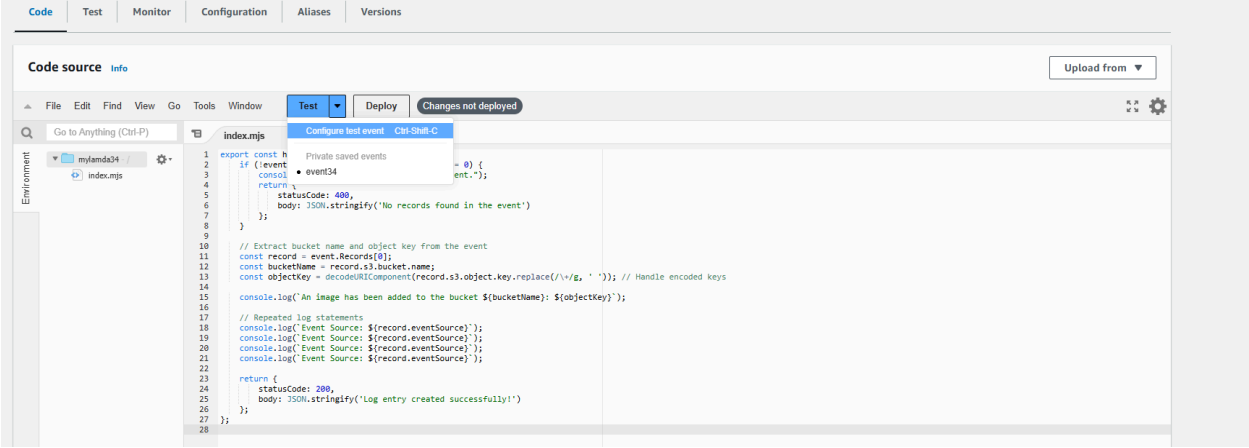
```
export const handler = async (event) => {
  if (!event.Records || event.Records.length === 0) {
    console.error("No records found in the event.");
    return {
      statusCode: 400,
      body: JSON.stringify('No records found in the event')
    };
  }

  // Extract bucket name and object key from the event
  const record = event.Records[0];
```

```
    const bucketName = record.s3.bucket.name;
    const objectKey = decodeURIComponent(record.s3.object.key.replace(/\+/g, ' ')); //
Handle encoded keys

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

    // Repeated log statements
    console.log(`Event Source: ${record.eventSource}`);
    console.log(`Event Source: ${record.eventSource}`);
    console.log(`Event Source: ${record.eventSource}`);
    console.log(`Event Source: ${record.eventSource}`);

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

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



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

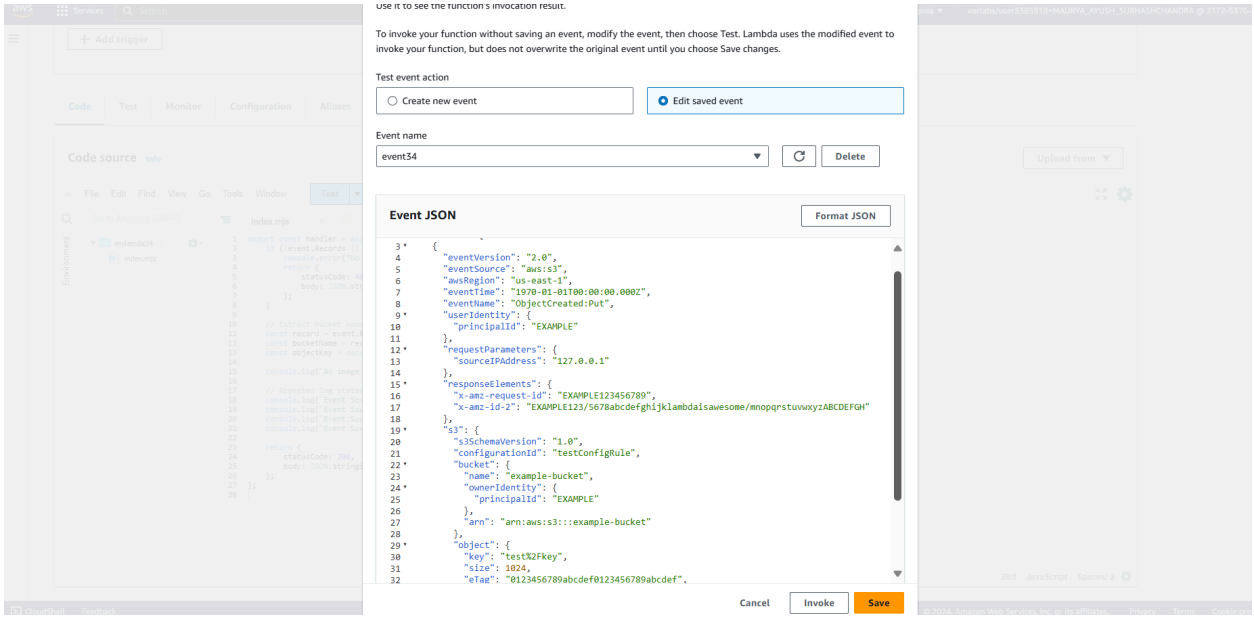


6) Here, select edit saved event. Select the event that you had created before. Under Event JSON, paste the following code.

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

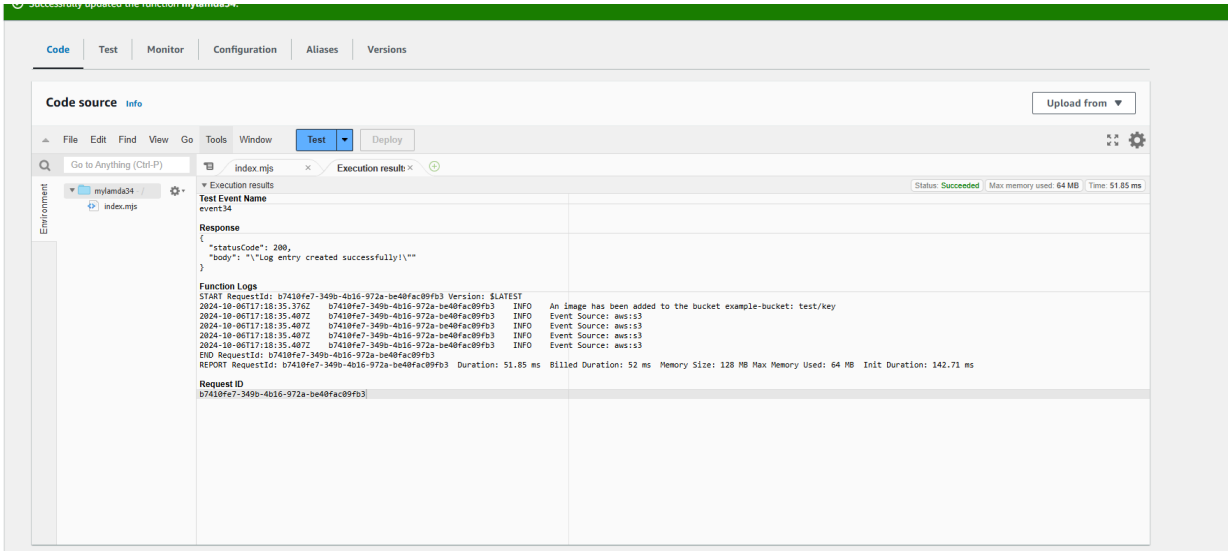
```
        },
        "arn": "arn:aws:s3:::example-bucket"
      },
      "object": {
        "key": "test%2Fkey",
        "size": 1024,
        "eTag": "0123456789abcdef0123456789abcdef",
        "sequencer": "0A1B2C3D4E5F678901"
      }
    }
  }
 ]
}
```

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



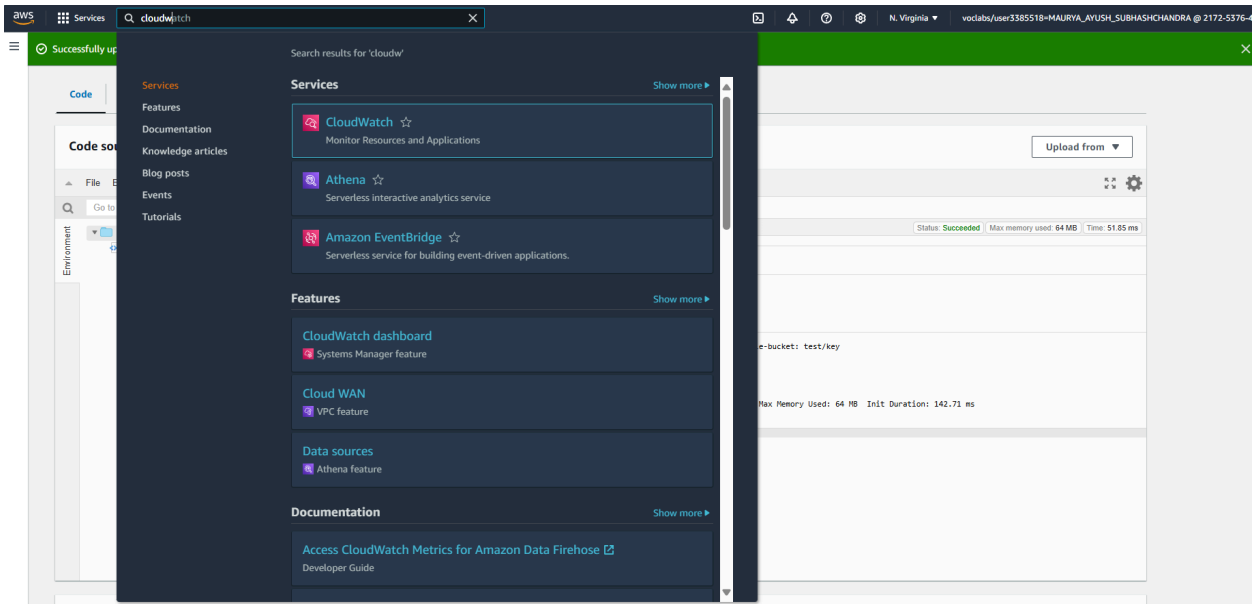
Save the changes. Then deploy the code changes by clicking on deploy

7) After deploying, click on Test. The console output shows that ‘an image has been added to the bucket’
The JSON response shows that the log entry was created successfully.

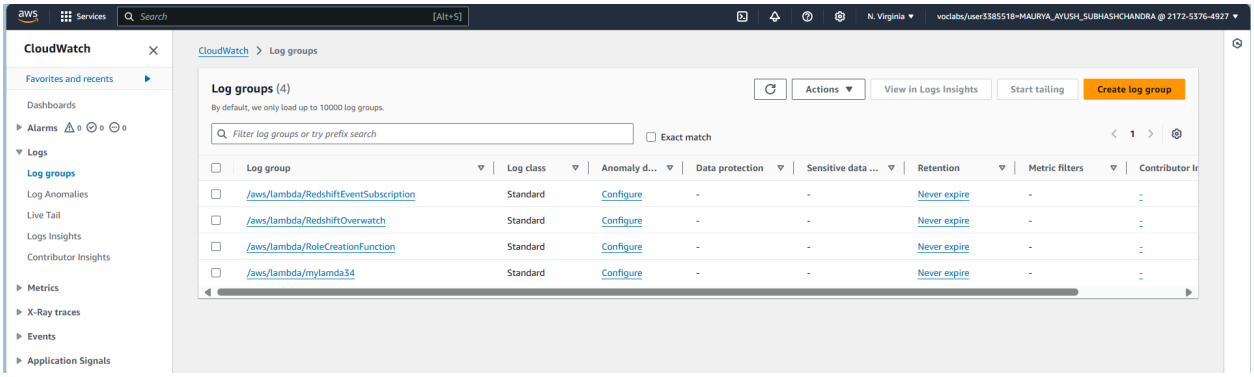


Step 3: Check the logs

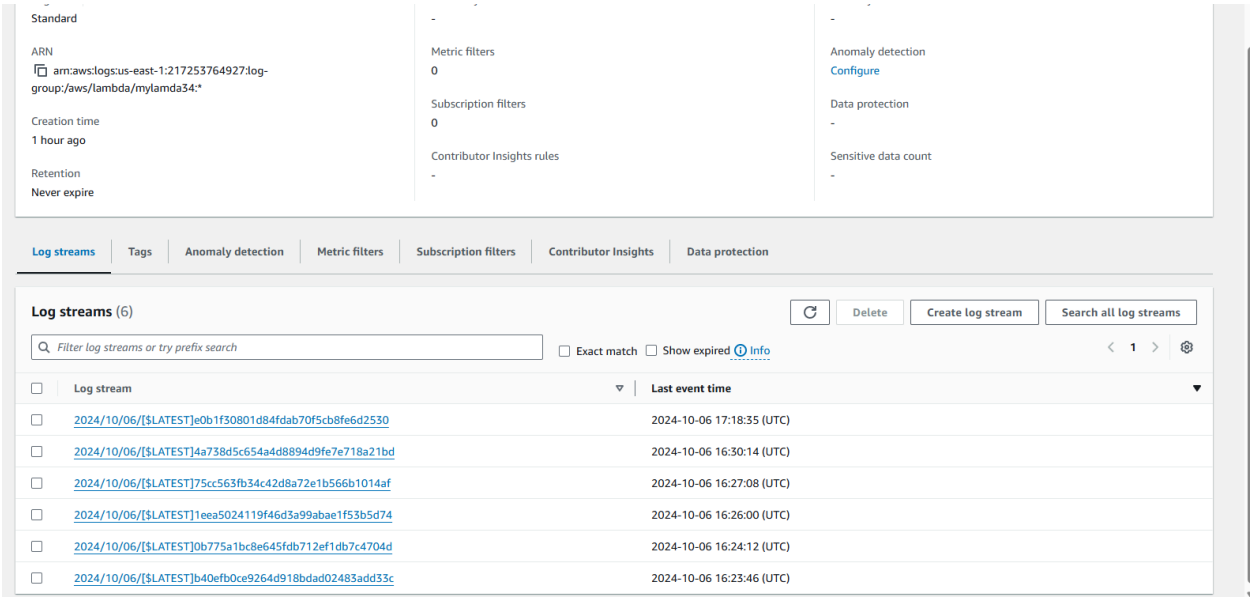
1) To check the logs explicitly, search for CloudWatch on services and open it in a new tab



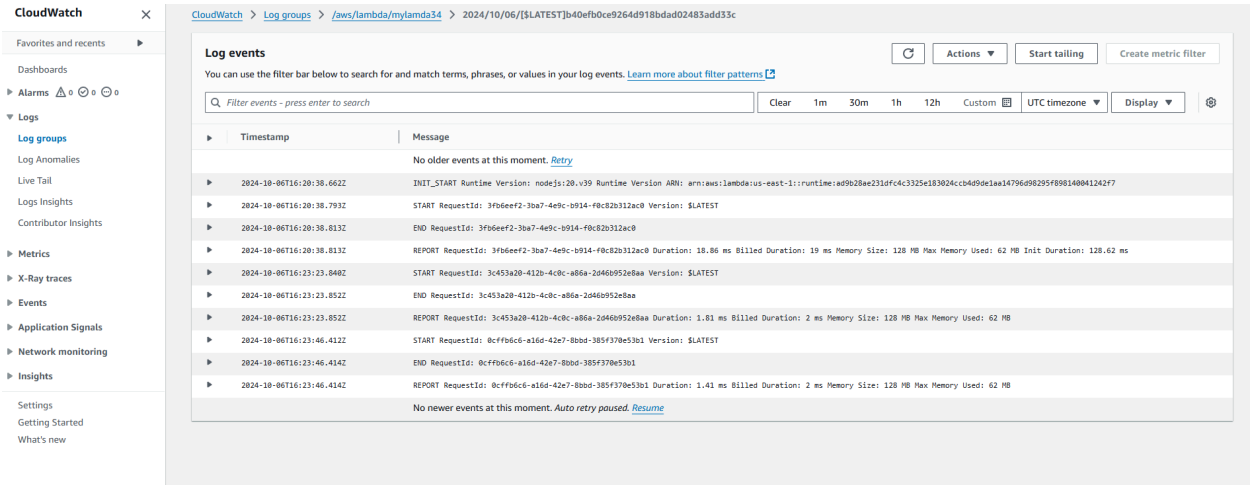
2) Here, Click on Logs → Log Groups. Select the log that has the lambda function name you just ran.



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 and configured an AWS Lambda function to log when an image is added to a specific S3 bucket. We explored the integration between AWS Lambda and S3, demonstrating how event-driven processes can automate tasks. By setting up an S3 bucket trigger, we enabled the Lambda function to detect object uploads and log essential details like the bucket name and object key. After deploying and testing the function, we verified the logs in CloudWatch, confirming that the function worked as expected, accurately detecting and logging the addition of images to the bucket. This experiment showcases how AWS services can seamlessly collaborate for automation.