Sairam Konar D15C 27 2024-25

Experiment 1(a)

Aim: To develop a website and host it on
i) local machine or virtual machine
ii) Amazon S3 Bucket

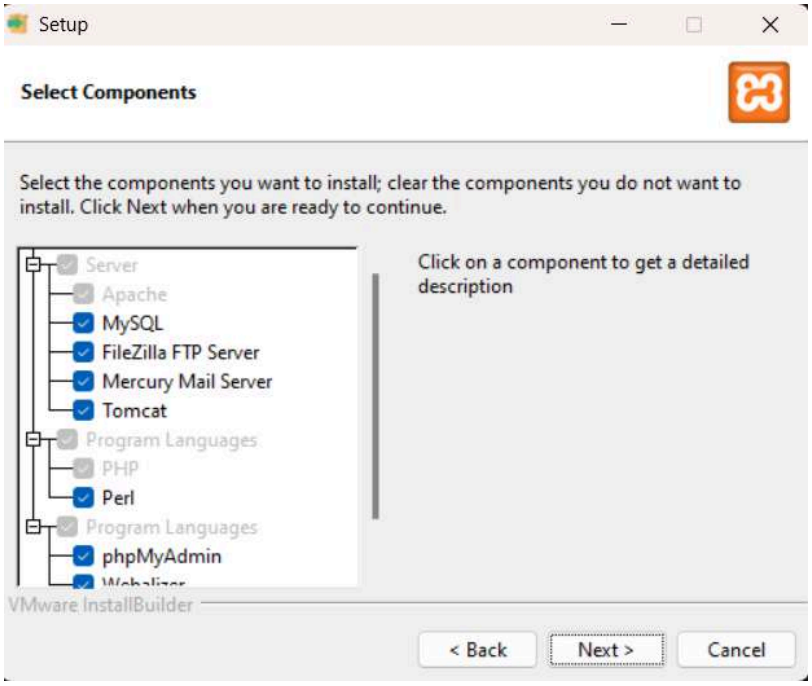
Static Hosting:

1) On local server (XAMPP)

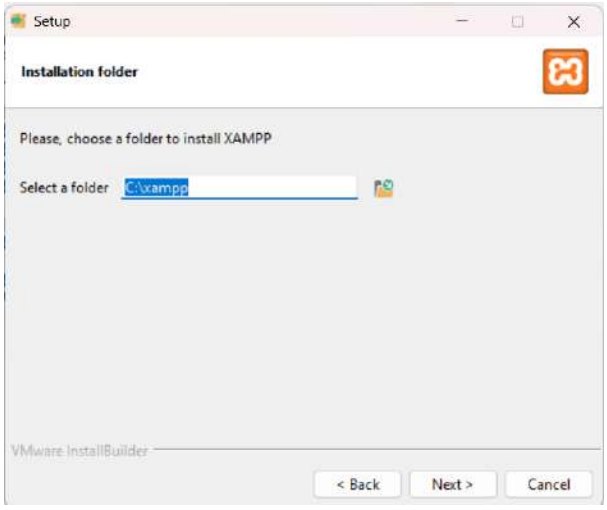
Step 1: Install XAMPP from https://www.apachefriends.org/ .

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

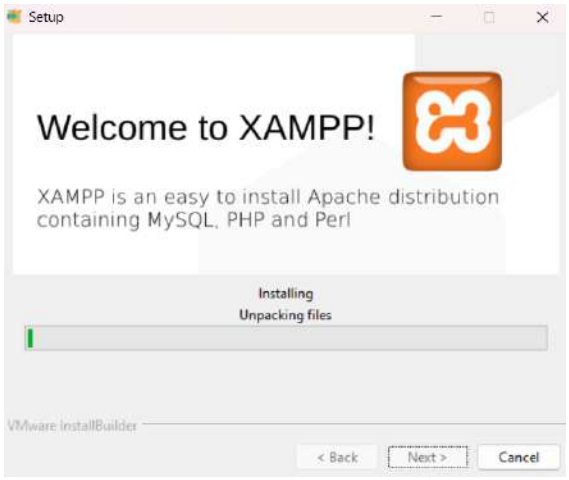
2) Open the setup file. Select all the required components and click next.



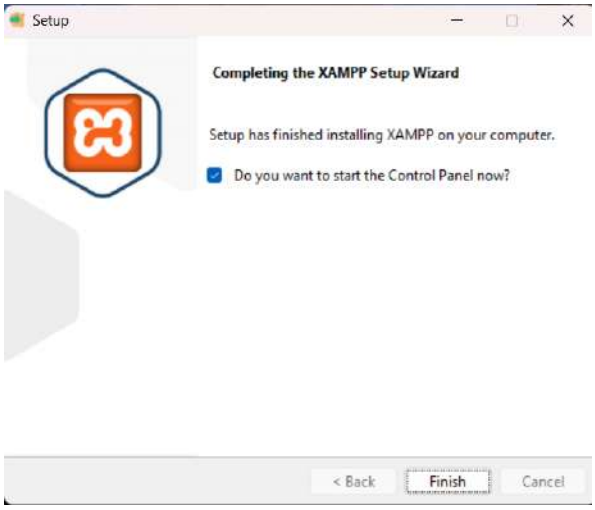
3) Choose the folder to install XAMPP in. Make sure the folder is empty. Click next



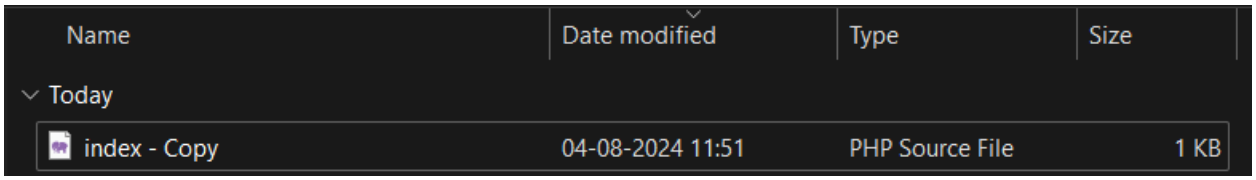
4) Select the language, click next. XAMPP starts to install



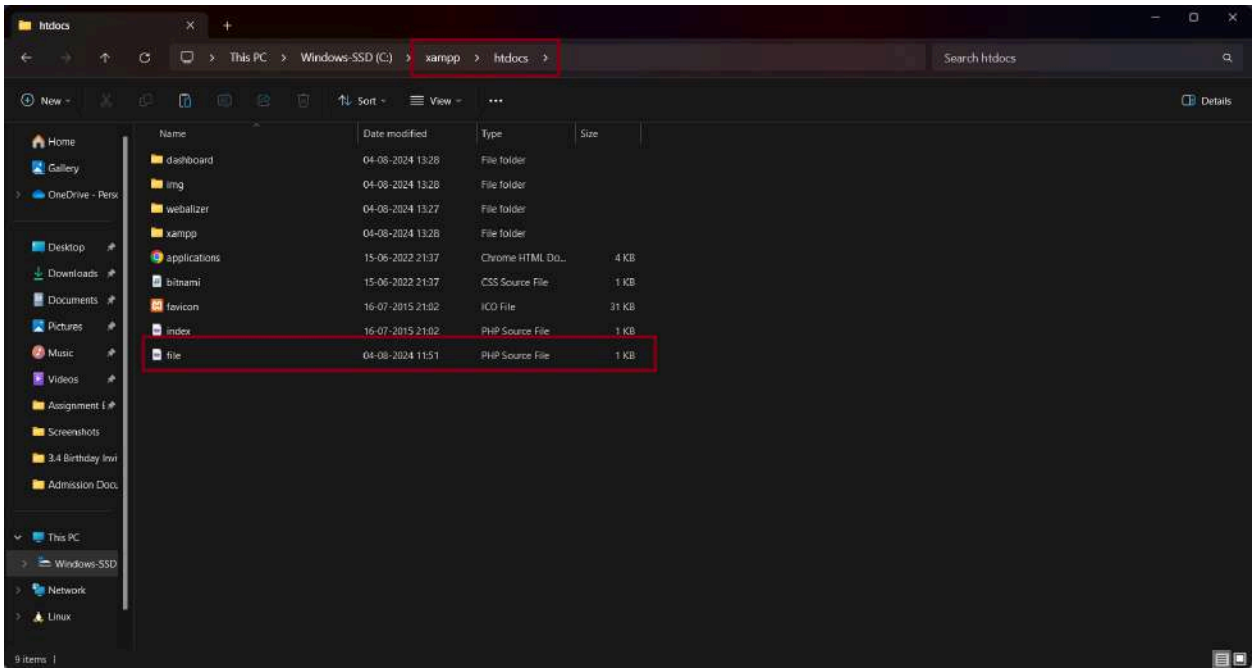
5) The installation is complete. Click Finish



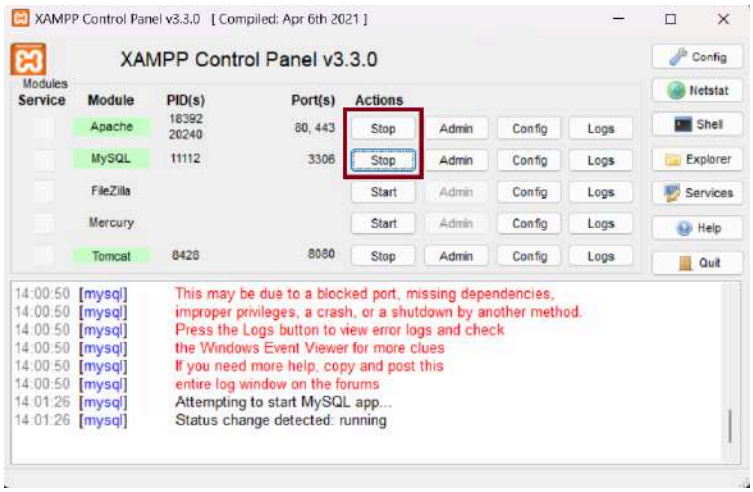
Step 2: Setup a file that is to be hosted on the server. Make sure the file has extension **.php**



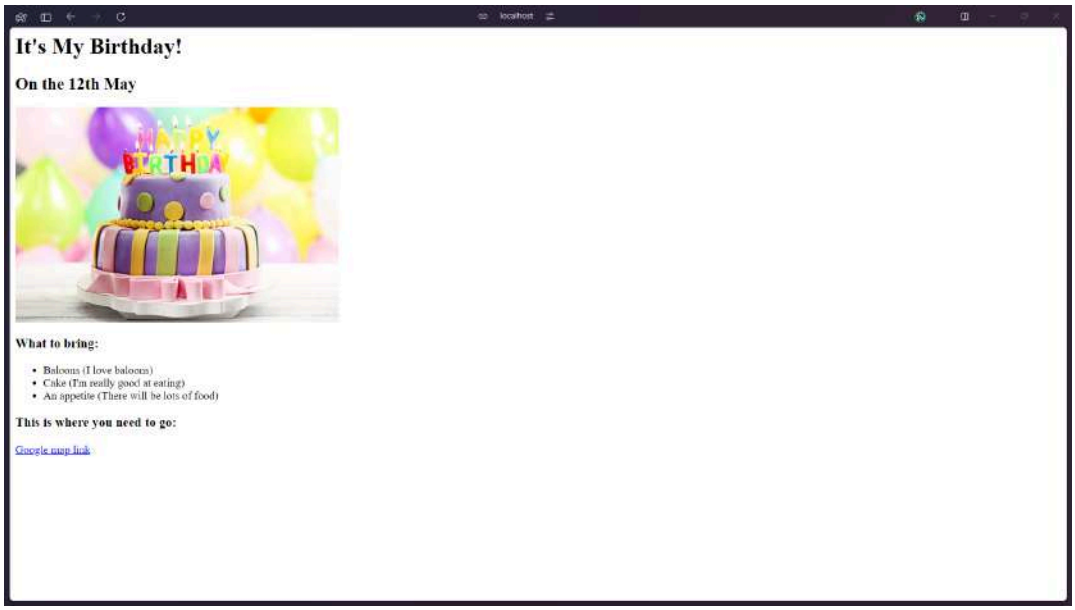
Step 3: Go to the directory where XAMPP was installed. Go to **htdocs** folder. Place your folder in this directory.



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

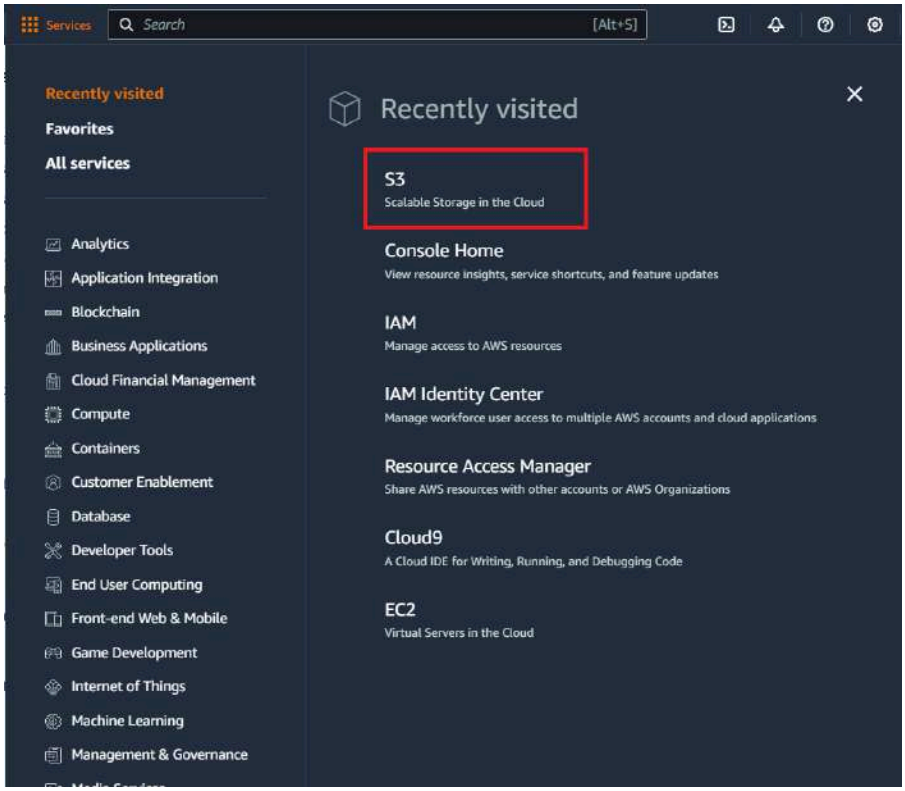


Step 5: Open your web browser. Type localhost/YOUR_FILENAME.php. This will open your website on your browser.

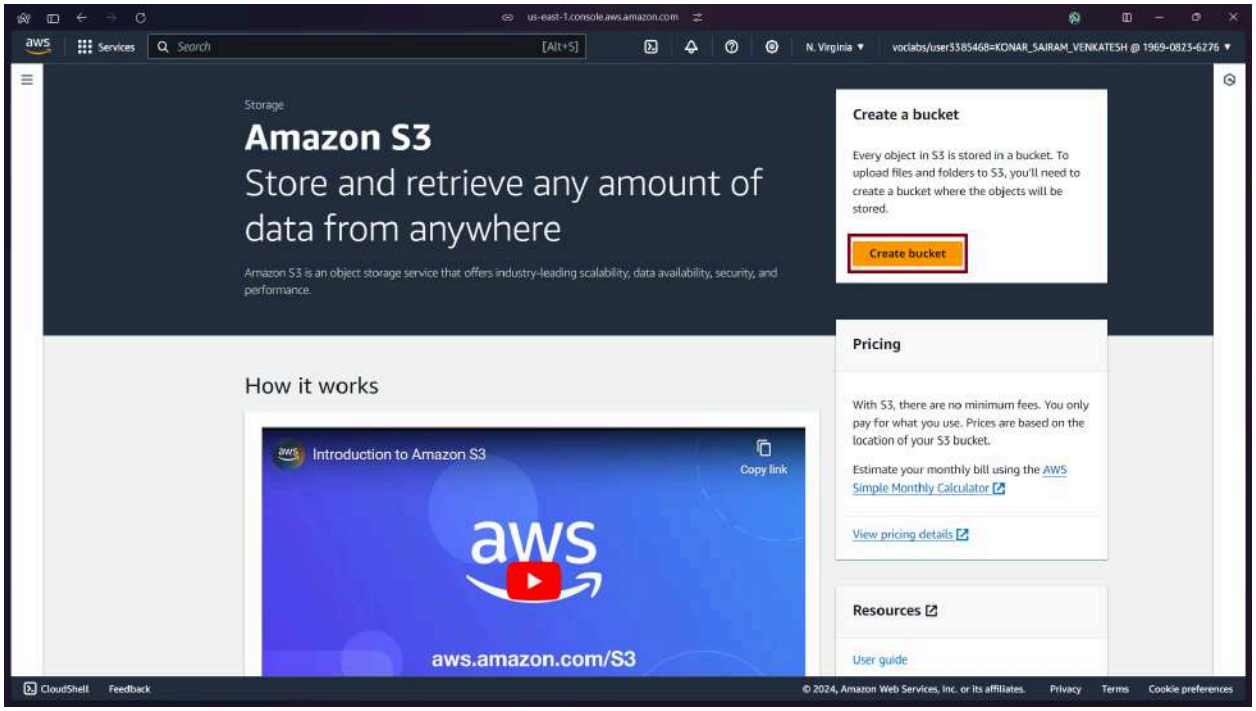


2) AWS S3

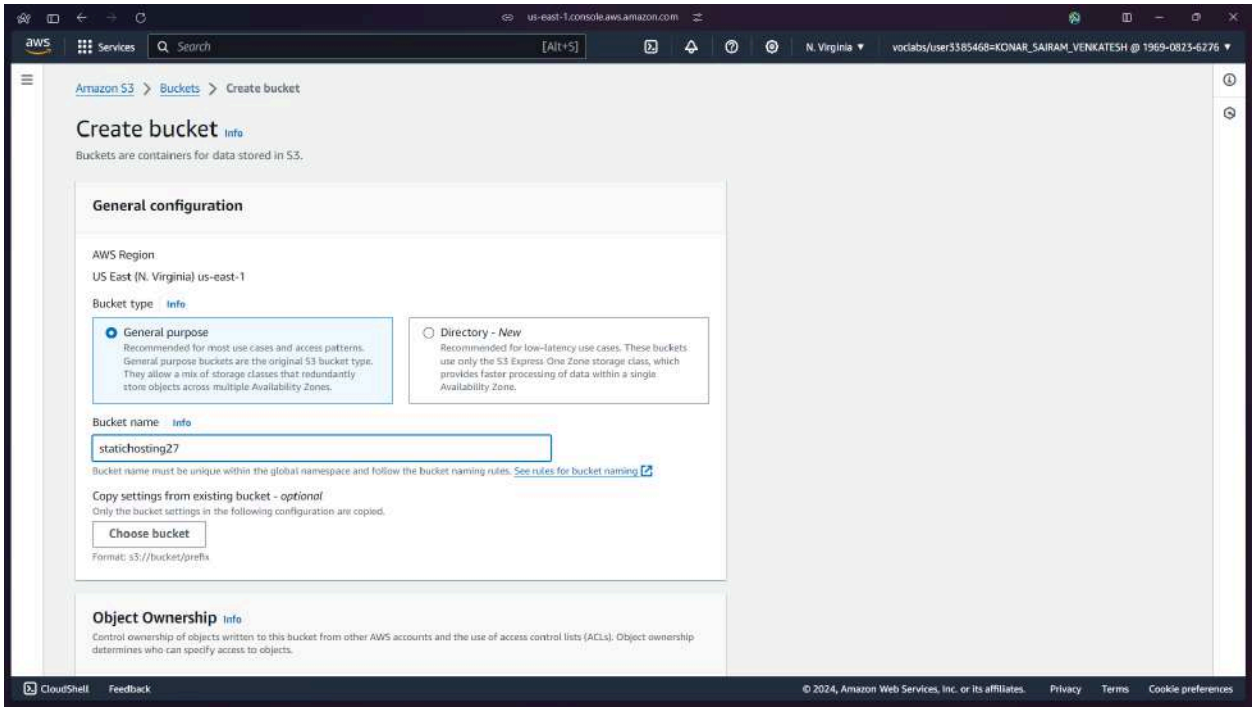
Step 1: Login to your AWS account. Go to services and open **S3.**



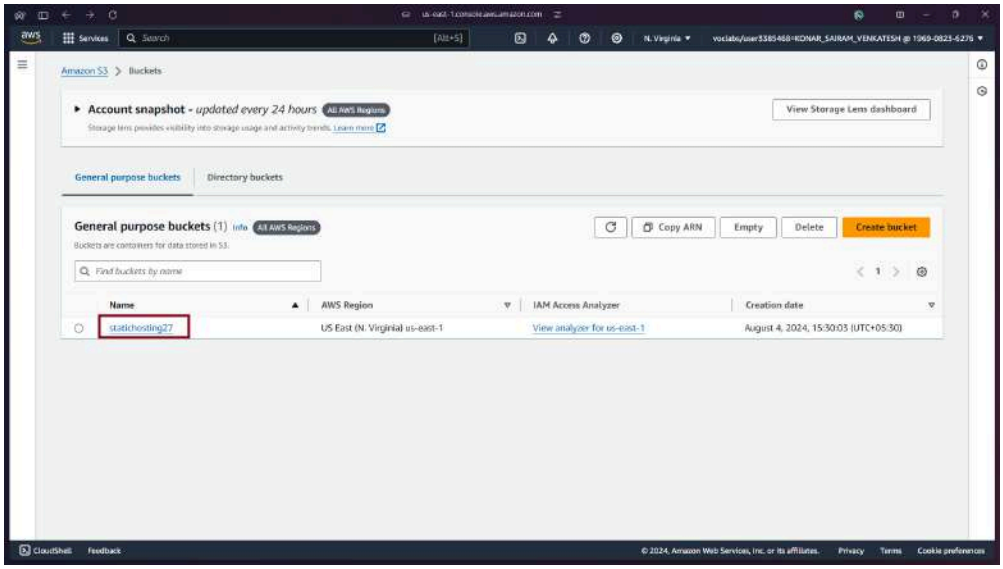
Step 2: Click on Create Bucket.

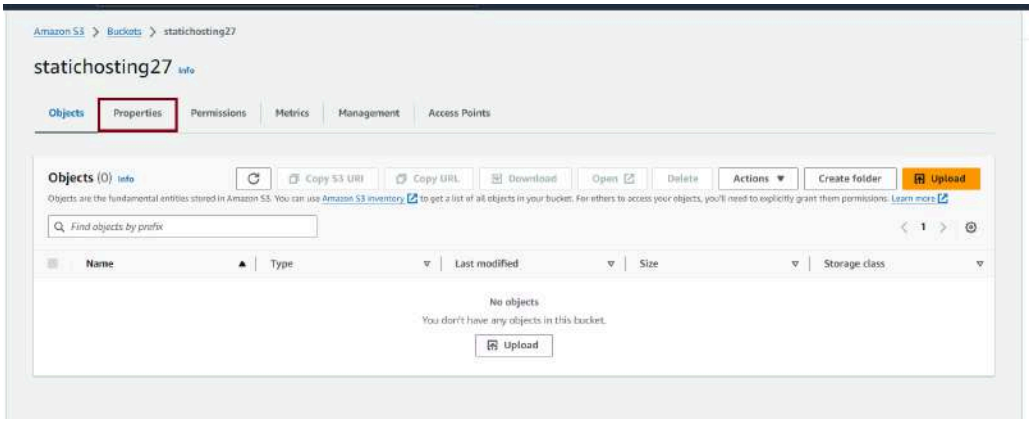


Step 3: Give a name to your bucket, keeping other options default, scroll down and click on Create Bucket.

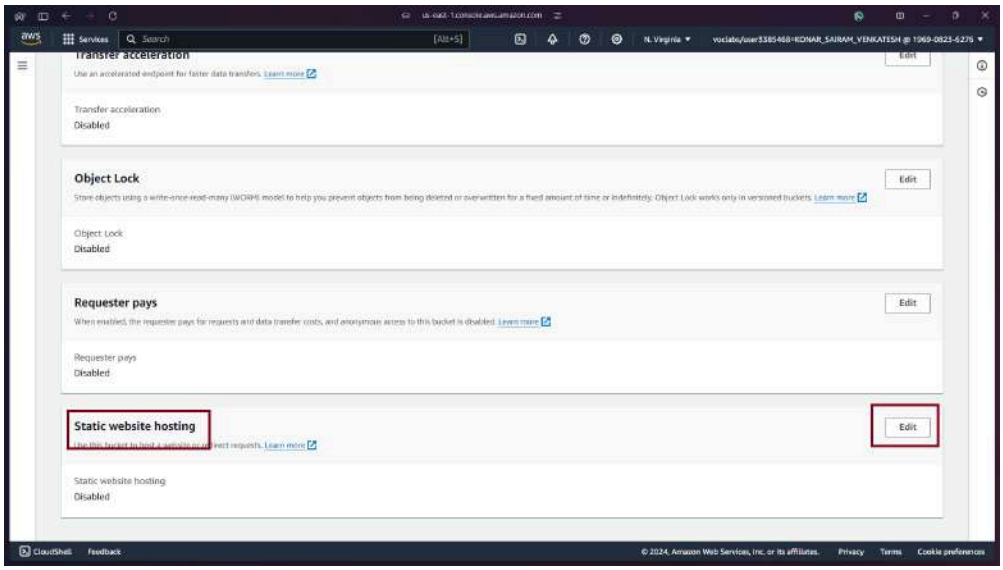


Step 4: Click on the name of your bucket and goto Properties

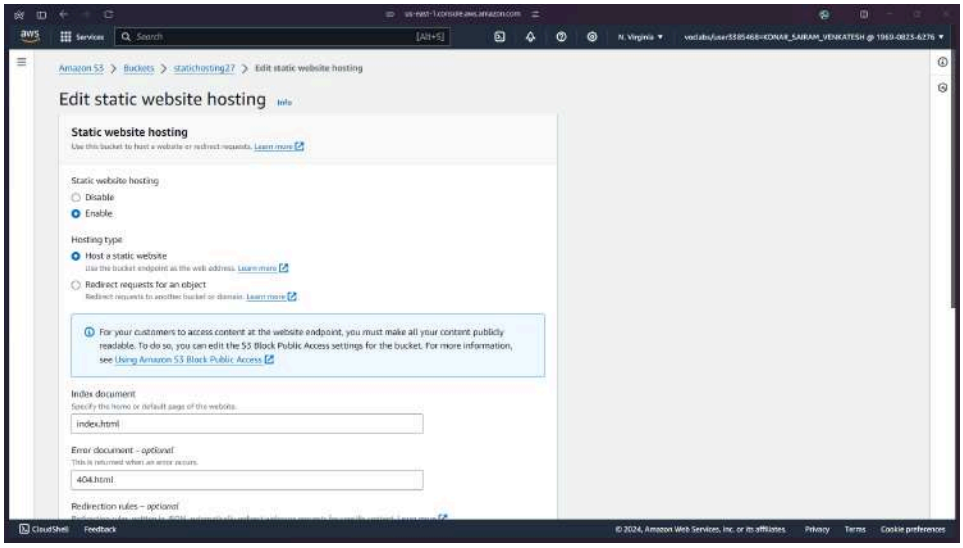




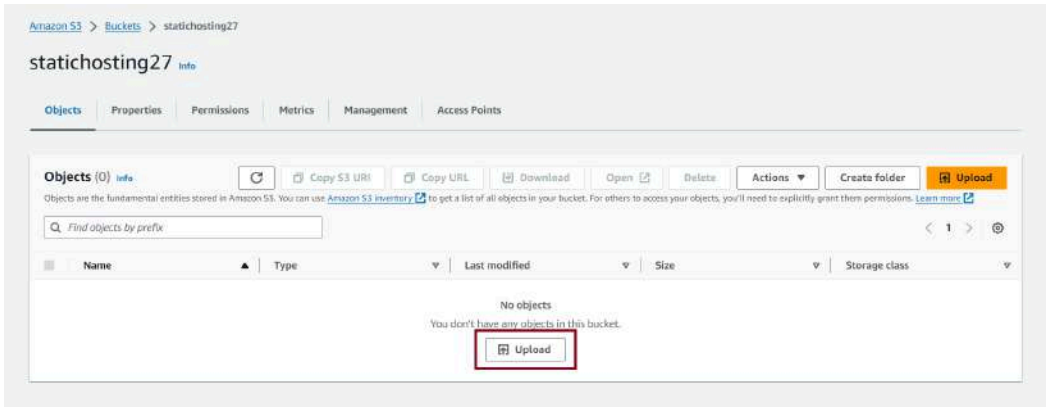
Step 5: Scroll down till you find Static website hosting, click on edit



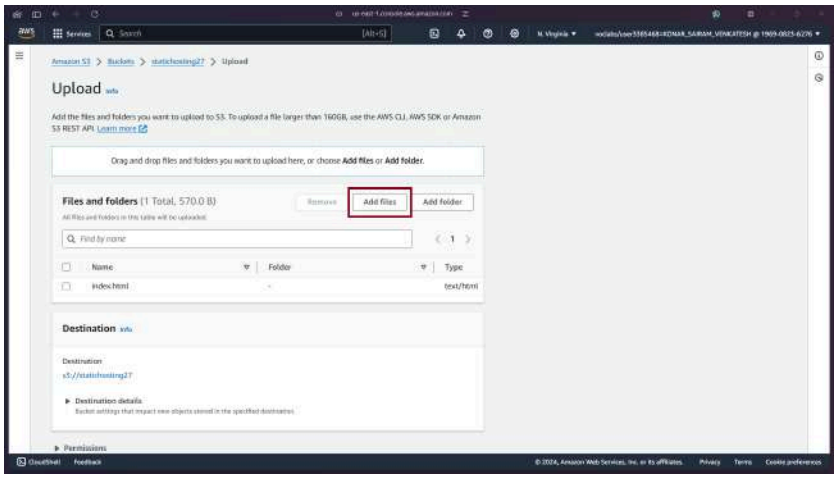
Step 6: Enable static website hosting, in Index document, write the name of your document and in error document, give name as 404.html. Save your changes.



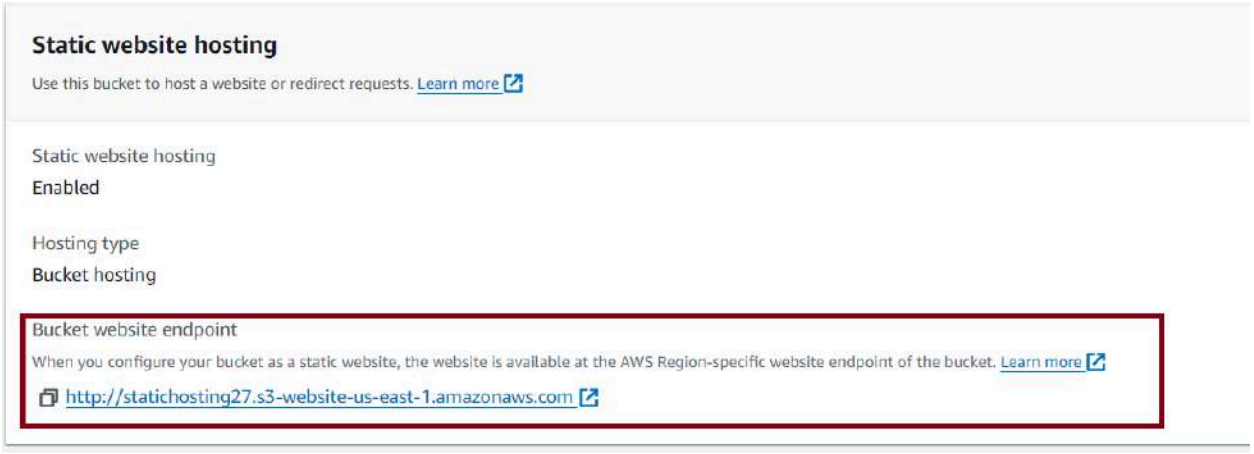
Step 7: Go to Objects tab and click on upload file.



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



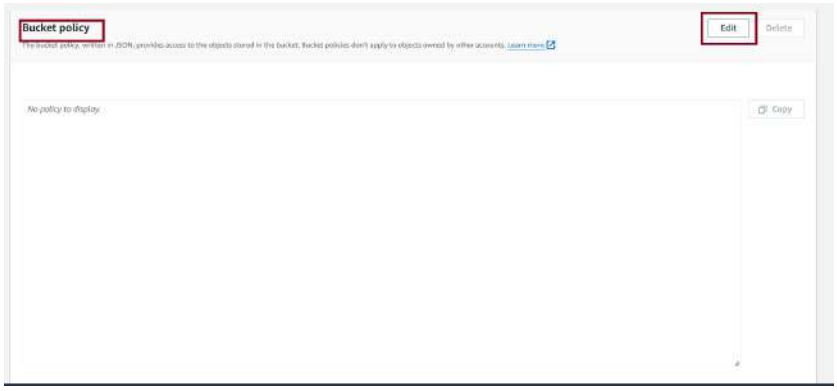
Step 9: This will take you to the Objects screen. Switch to Properties and scroll down to Static Website Hosting. There you would find the link (Bucket website endpoint) to your website.



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

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

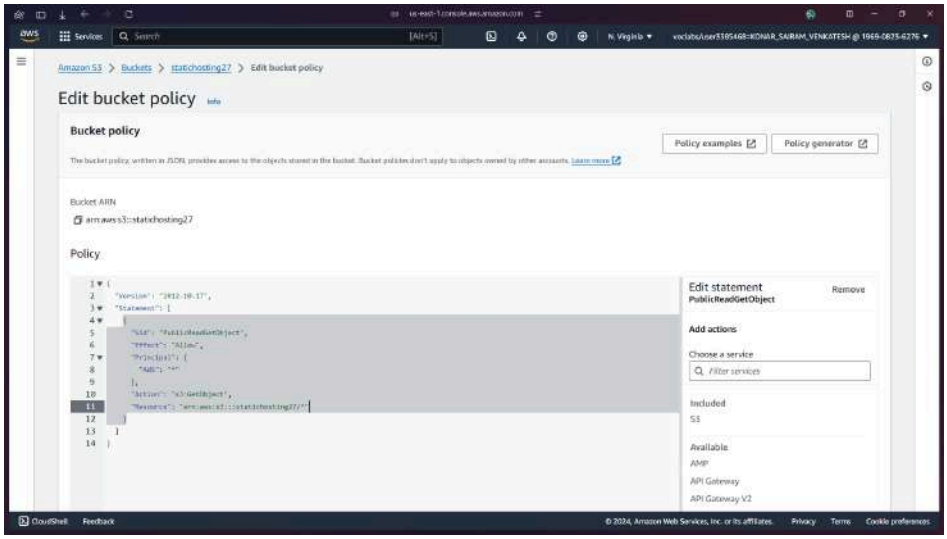
Step 12: Scroll down to bucket policy and click edit



Step 13:

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

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



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

Conclusion:

In this experiment, we learned how to host a static website both on a local server using XAMPP and on AWS S3. By setting up XAMPP, we were able to deploy a local web server and access the hosted site through the browser using localhost. Additionally, we configured an S3 bucket for static website hosting on AWS, where we uploaded files, enabled public access, and applied the necessary bucket policies to make the website accessible. This hands-on process illustrated the differences between local and cloud hosting, emphasizing the ease of deployment and configuration flexibility that each method offers.

Sairam Konar D15C 27 2024-25

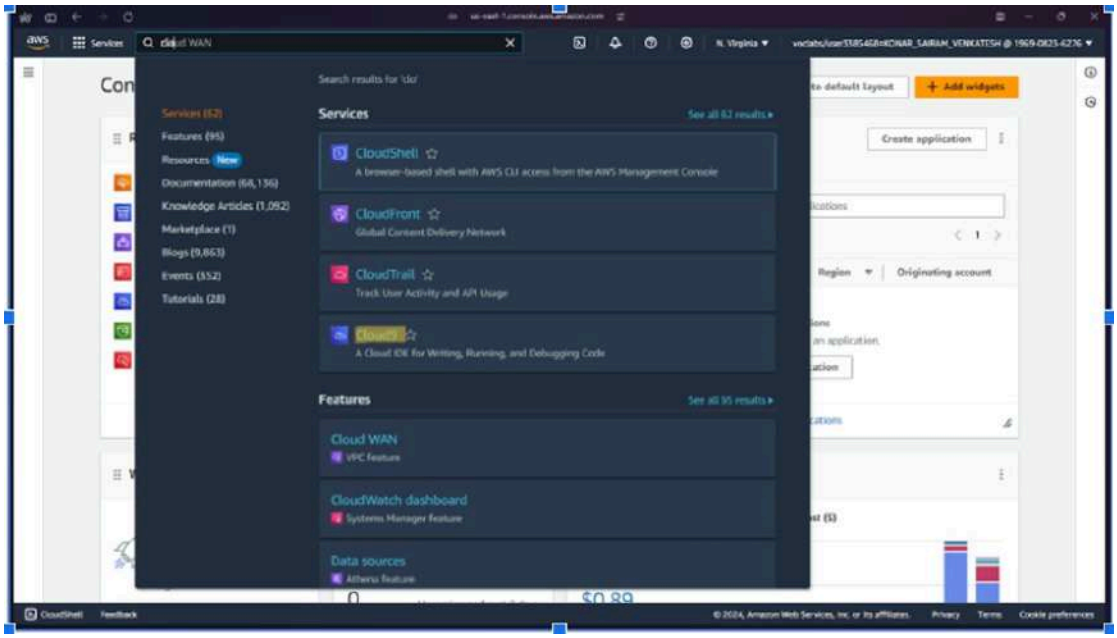
Experiment 1(b)

Aim: To understand the benefits of Cloud Infrastructure and Setup AWS Cloud9 IDE, Launch AWS Cloud9 IDE and Perform Collaboration Demonstration.

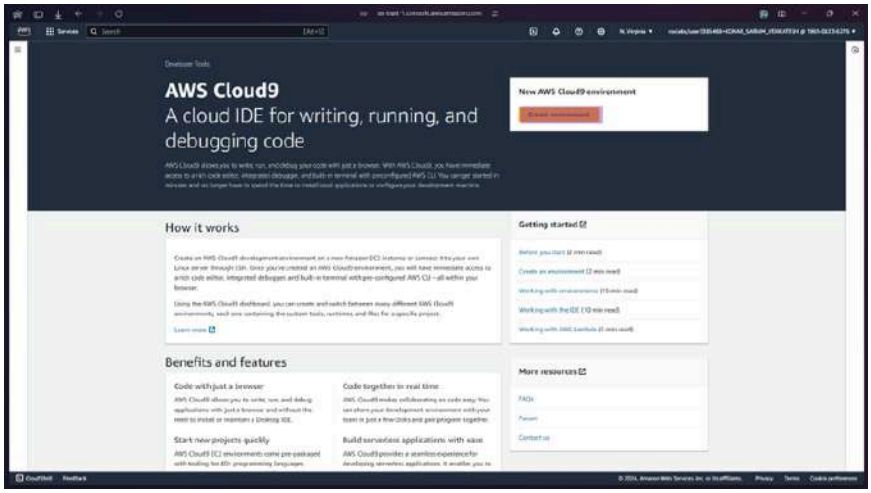
Steps:

Step 1: Set up Cloud9 environment.

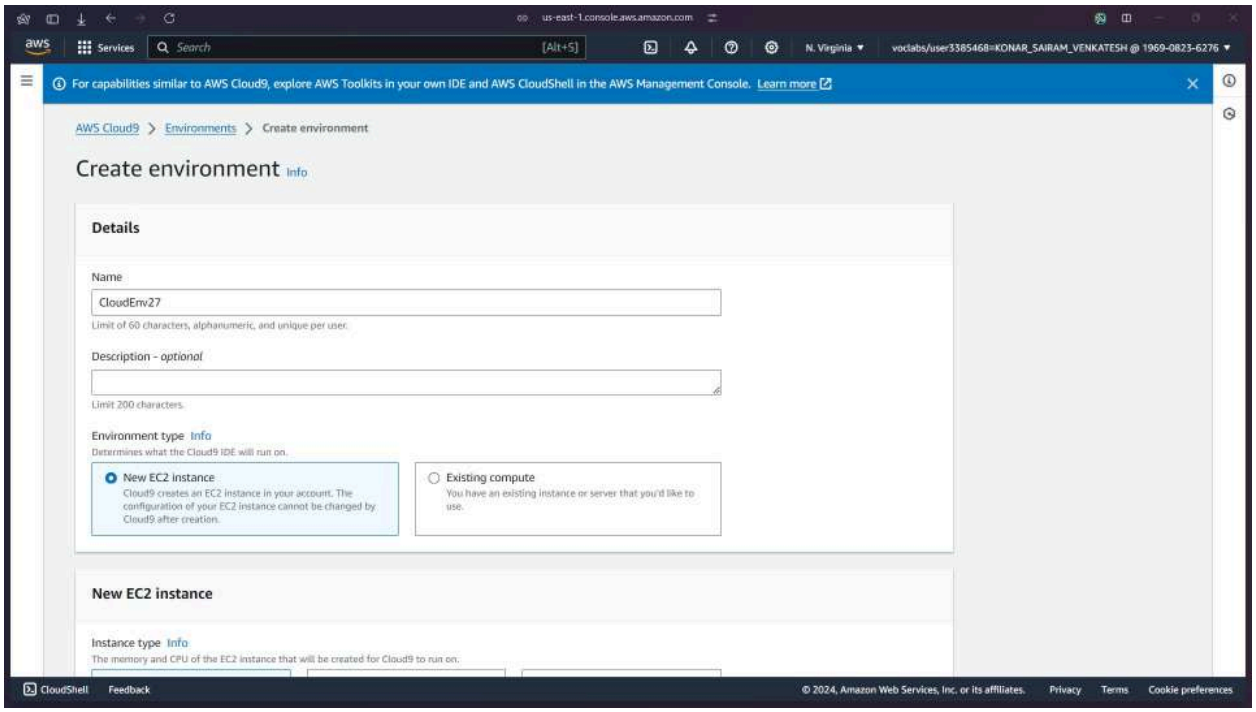
1) Search Cloud9 in the services tab and open it



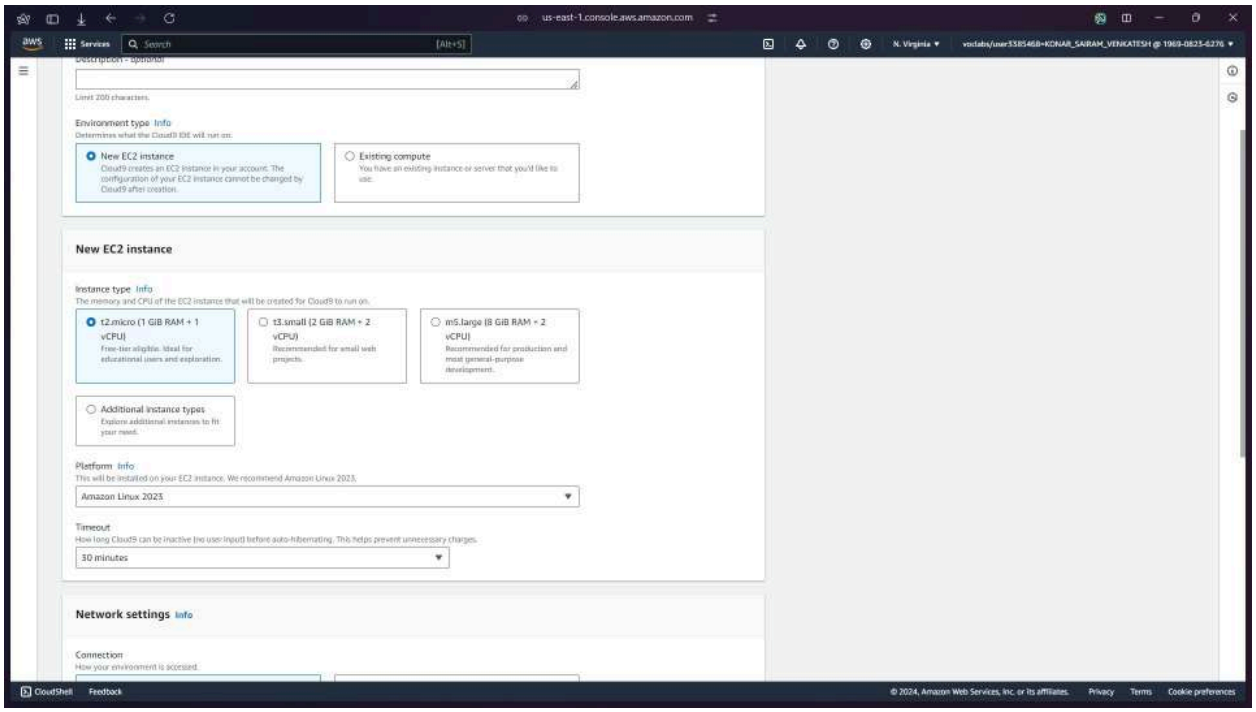
2) Click on Create Environment



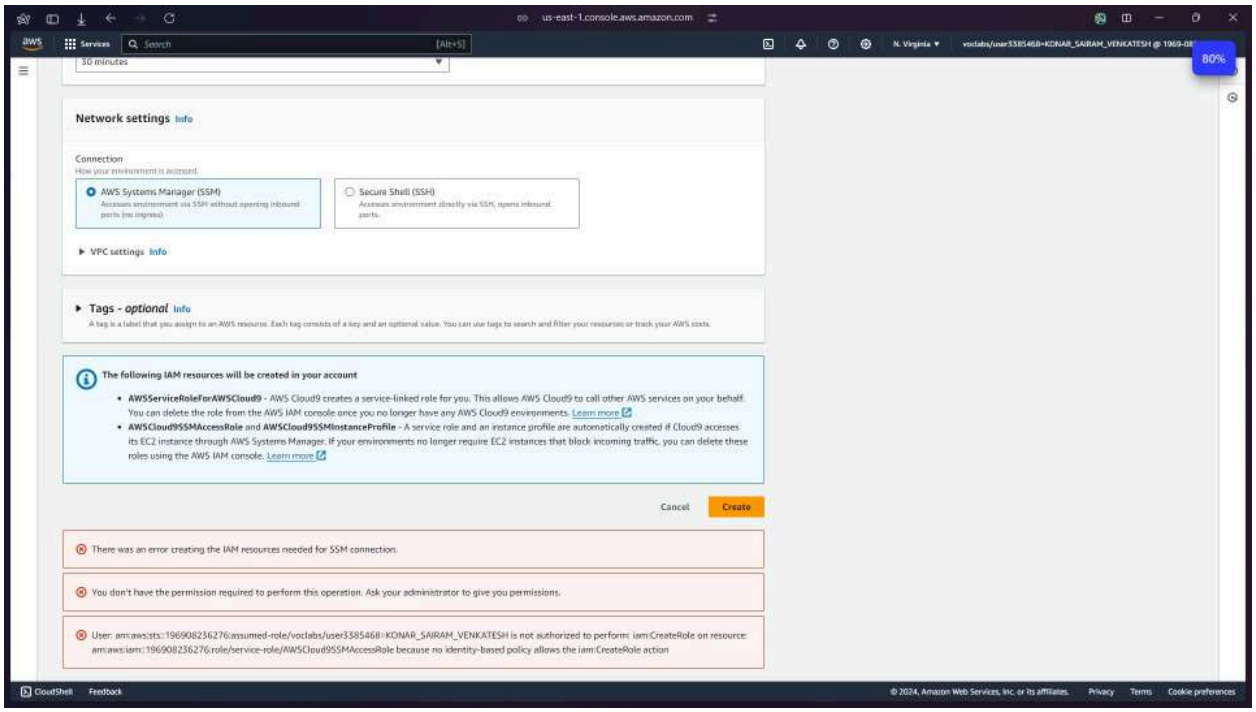
3) Give a name to your Cloud9 Environment. You can add a description if needed.

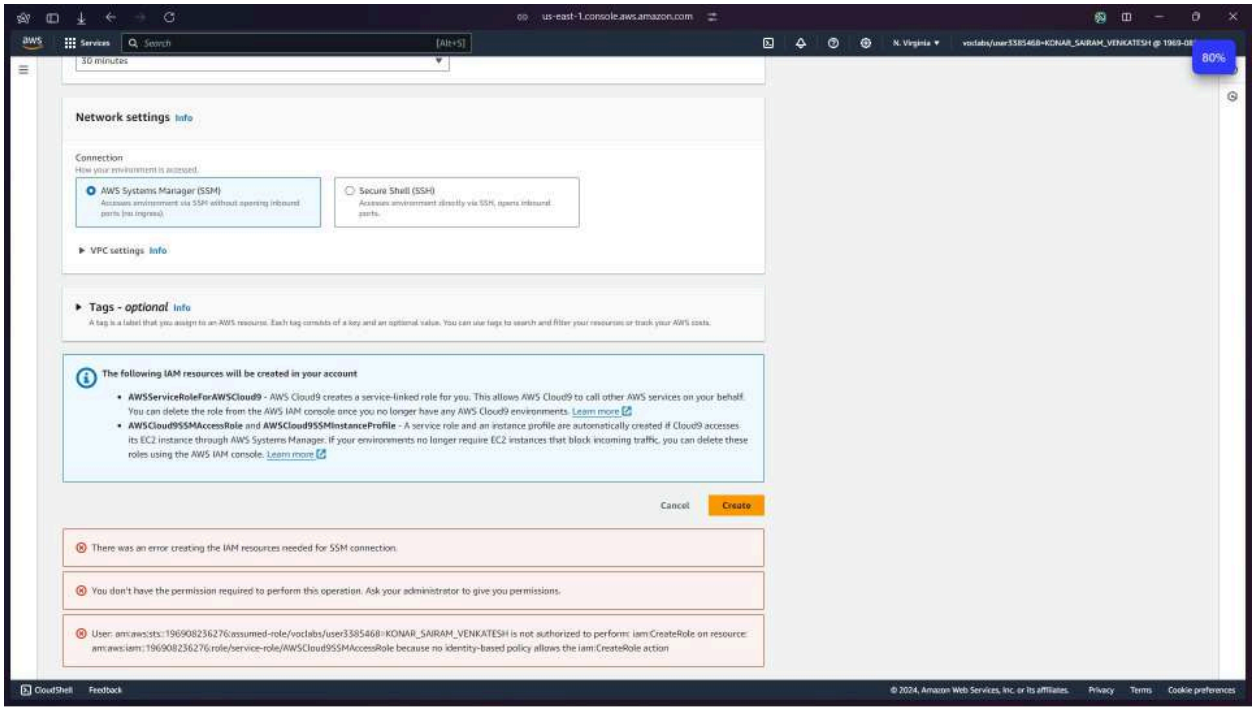


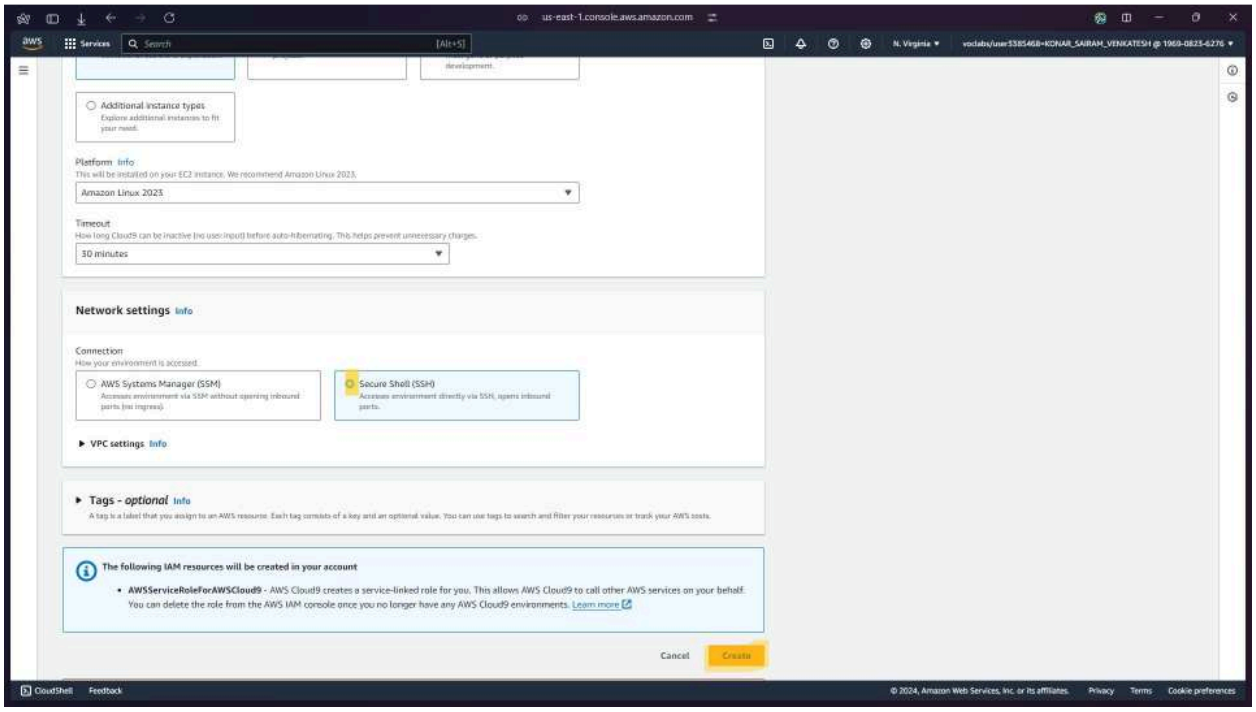
4) Select the option new EC2 instance if you do not have one ready for the environment. Give the specifications of that EC2 instance ahead.



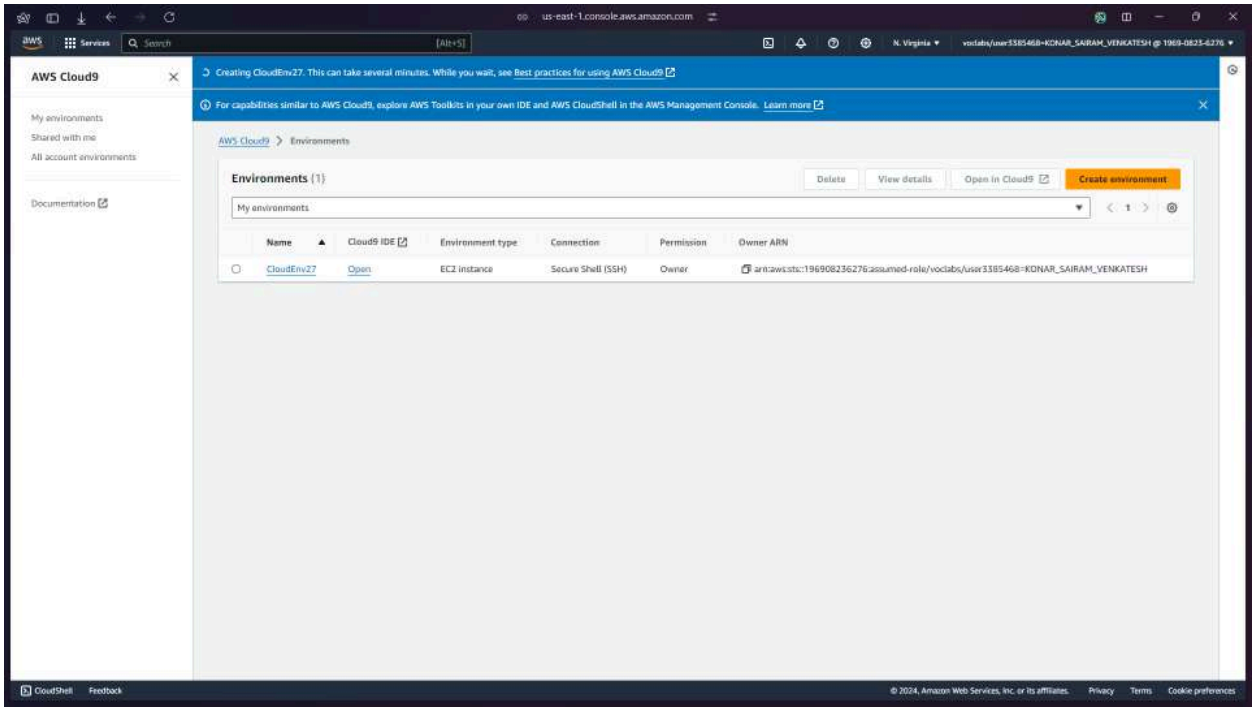
5) On the AWS Academy account, if we select AWS System Manager (SSM) in Network settings, it gives an error as the account does not have permissions to use the setting. So we select Secure Shell (SSH). After that click on Create.




us-east-1.console.aws.amazon.com
Services
Search
[Alt+S]
N. Virginia
30 minutes
Network settings Info
Connection
How your environment is accessed.
AWS Systems Manager (SSM)
Accesses environment via SSM without opening inbound ports (no ingress).
Secure Shell (SSH)
Accesses environment directly via SSH, opens inbound ports.
VPC settings Info
Tags - optional Info
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.
The following IAM resources will be created in your account
AWSServiceRoleForAWSCloud9 - AWS Cloud9 creates a service-linked role for you. This allows AWS Cloud9 to call other AWS services on your behalf. You can delete the role from the AWS IAM console once you no longer have any AWS Cloud9 environments. Learn more
AWSCloud9SSMAccessRole and AWSCloud9SSMInstanceProfile - A service role and an instance profile are automatically created if Cloud9 accesses its EC2 instance through AWS Systems Manager. If your environments no longer require EC2 instances that block incoming traffic, you can delete these roles using the AWS IAM console. Learn more
Cancel
Create
There was an error creating the IAM resources needed for SSM connection.
You don't have the permission required to perform this operation. Ask your administrator to give you permissions.
User: arn:aws:sts::196908236276:assumed-role/voclabs/user3385468=KONAR_SAIRAM_VENKATESH is not authorized to perform: iam:CreateRole on resource: arn:aws:iam::196908236276:role/service-role/AWSCloud9SSMAccessRole because no identity-based policy allows the iam:CreateRole action
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

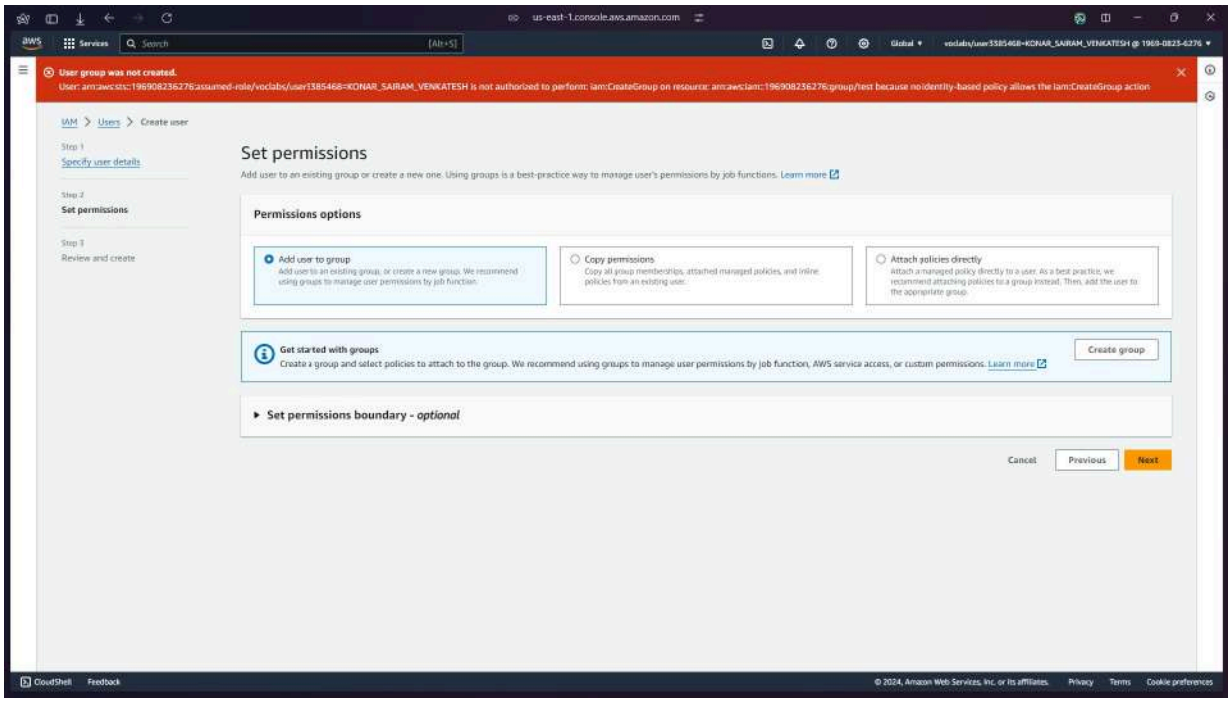

us-east-1.console.aws.amazon.com
Services
Search
[Alt+S]
N. Virginia
Additional instance types
Explore additional instances to fit your need.
Platform Info
This will be installed on your EC2 instance. We recommend Amazon Linux 2023.
Amazon Linux 2023
Timeout
How long Cloud9 can be inactive (no user input) before auto-hibernating. This helps prevent unnecessary charges.
30 minutes
Network settings Info
Connection
How your environment is accessed.
AWS Systems Manager (SSM)
Accesses environment via SSM without opening inbound ports (no ingress).
Secure Shell (SSH)
Accesses environment directly via SSH, opens inbound ports.
VPC settings Info
Tags - optional Info
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.
The following IAM resources will be created in your account
AWSServiceRoleForAWSCloud9 - AWS Cloud9 creates a service-linked role for you. This allows AWS Cloud9 to call other AWS services on your behalf. You can delete the role from the AWS IAM console once you no longer have any AWS Cloud9 environments. Learn more
Cancel
Create
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

6) The environment is being created.



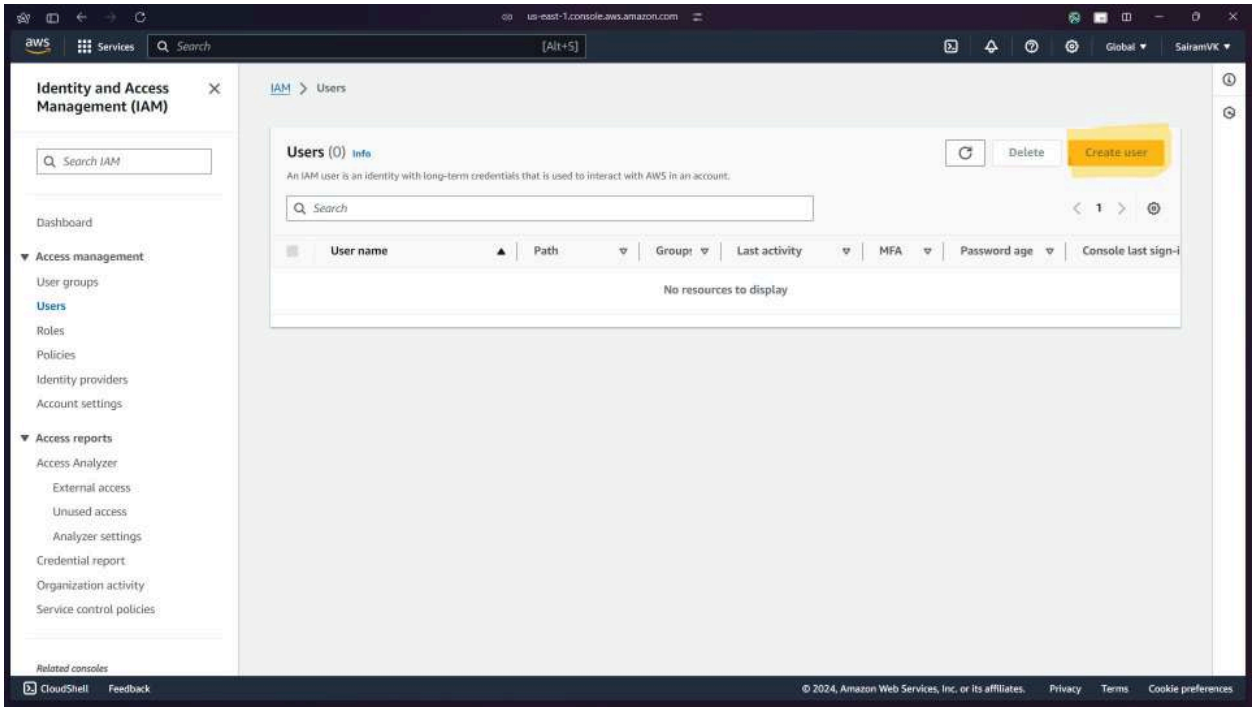
Step 2: Creating IAM user.

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.

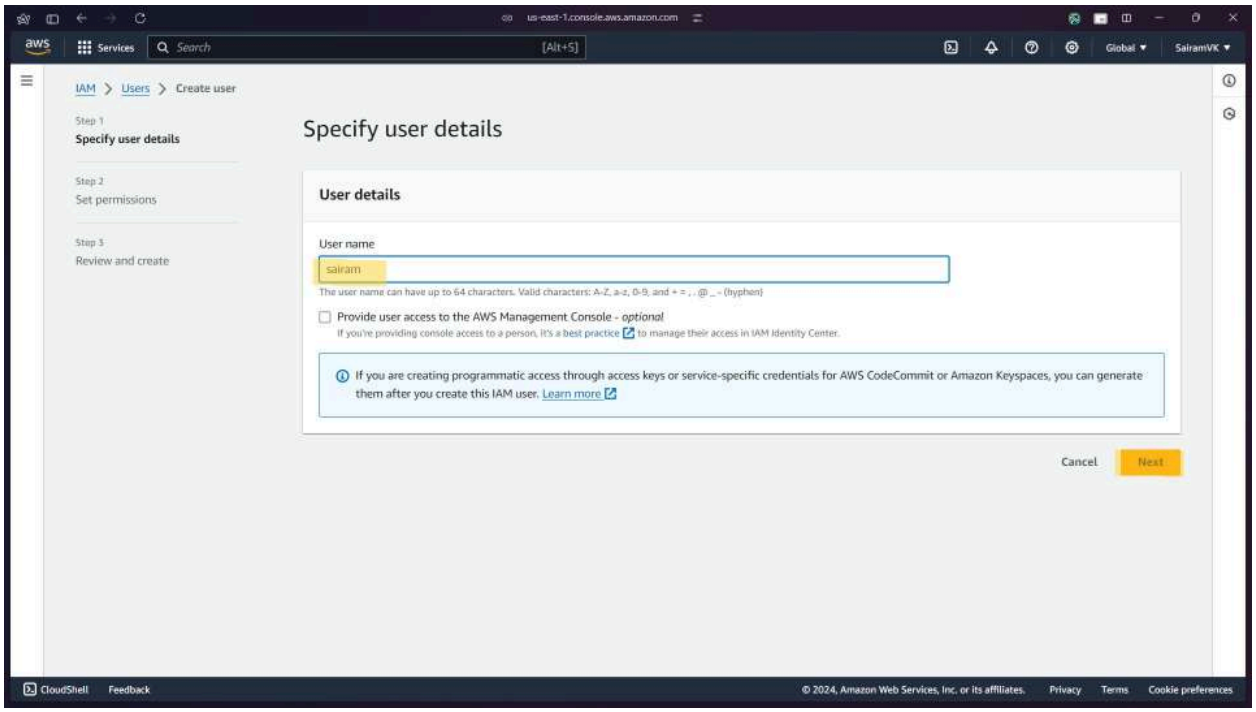


Sairam Konar D15C 27 2024-25

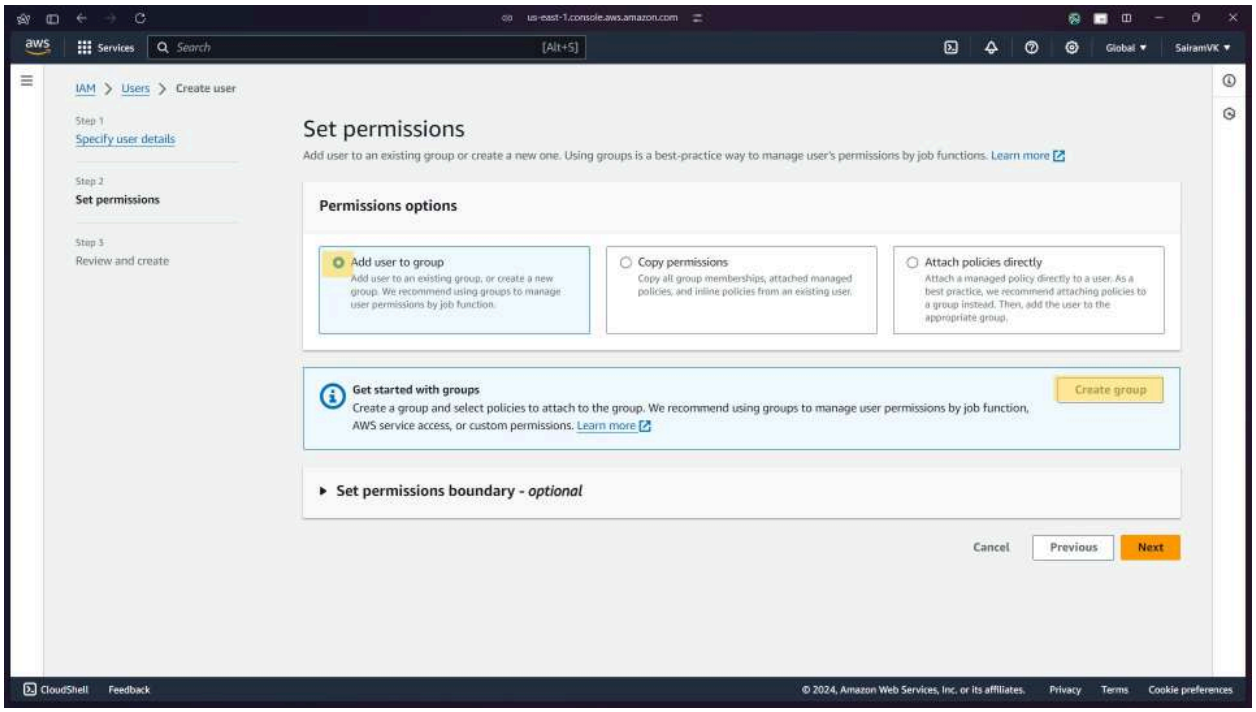
1) Search IAM on the services search bar and open it. Click on Create User.



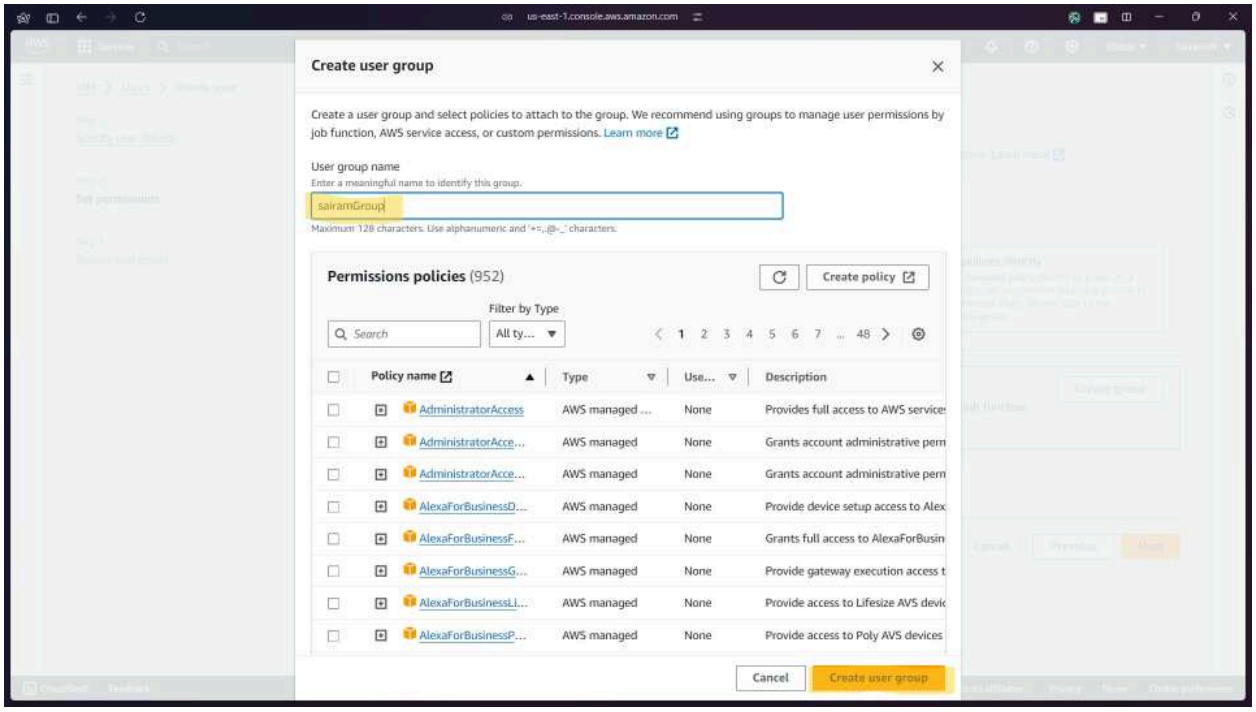
2) Give a username to your user and click Next.



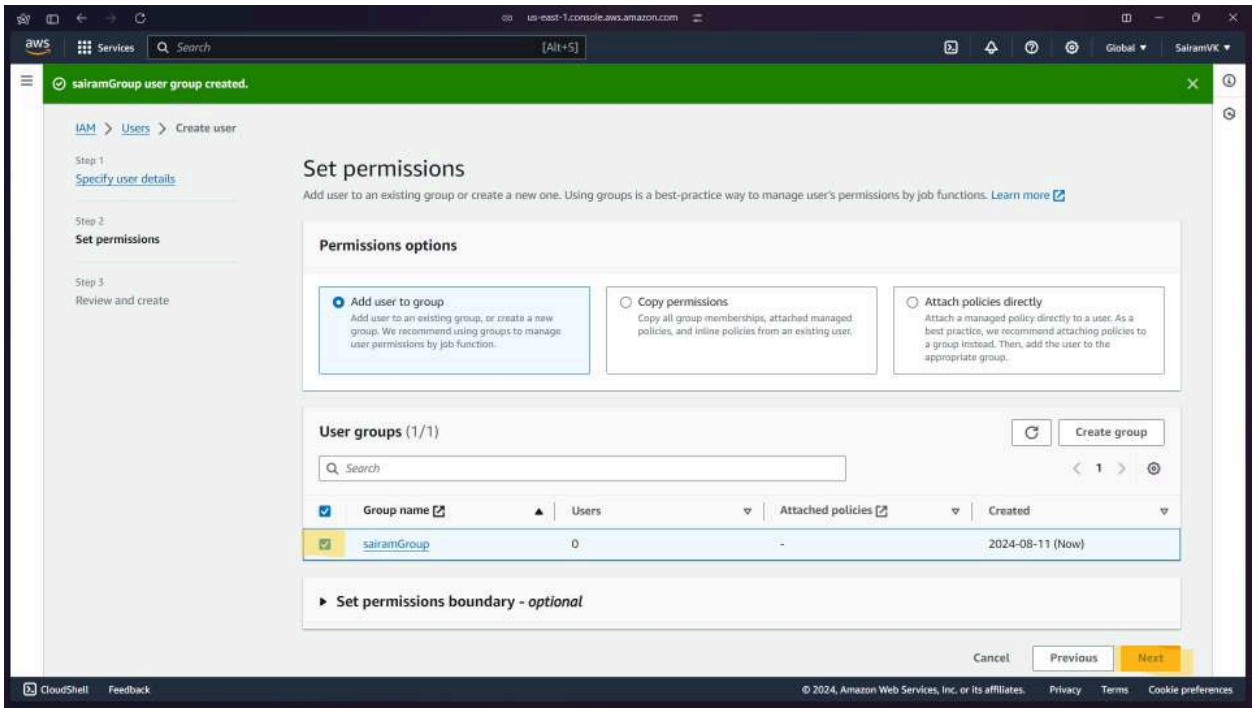
3) Select add User to Group. If there are no user groups on your accounts, you will have to create one. Click on Create Group.



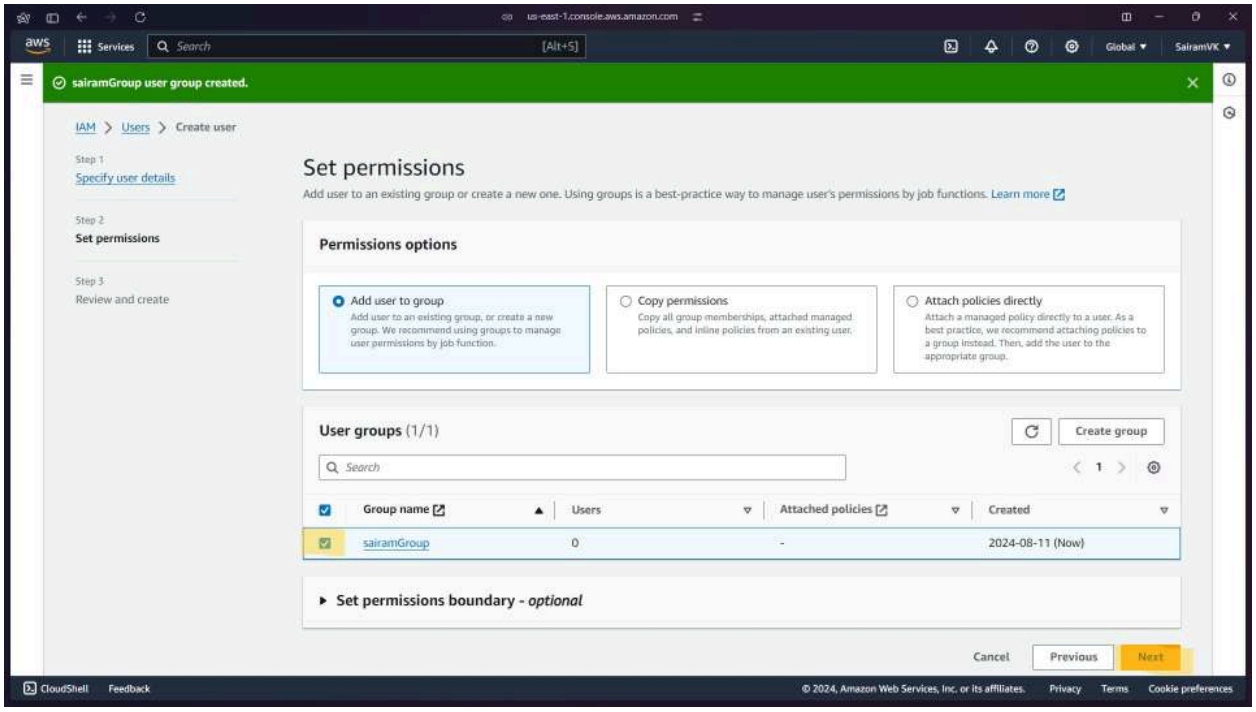
4) Give a name to your user group. Then click on Create User Group.



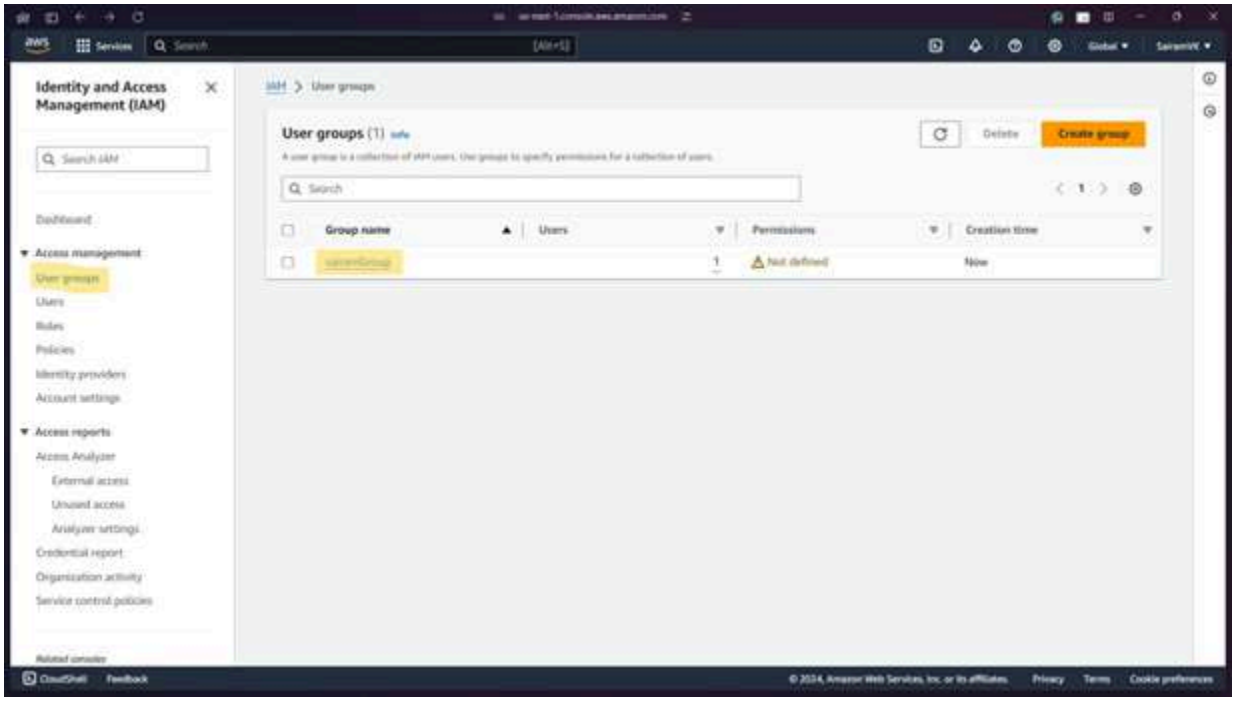
5) The group is created and shown under the groups area, select the group by clicking on the checkbox. Then click Next.



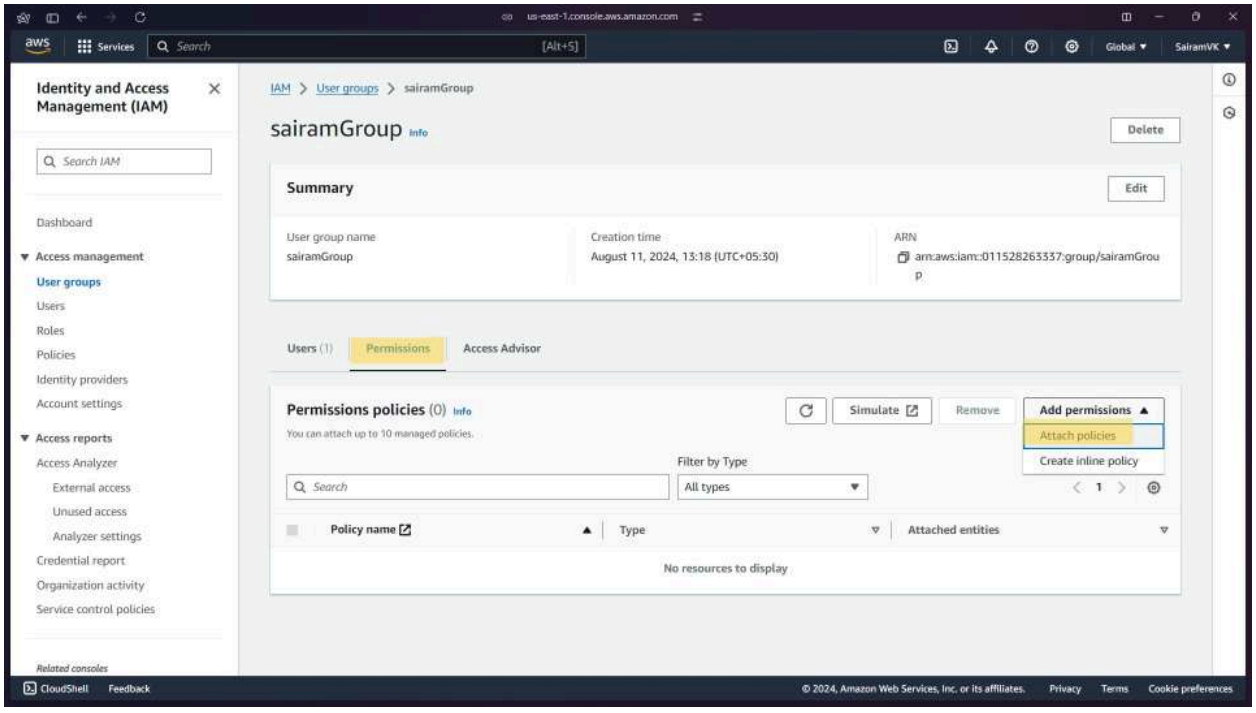
6) Review all the Information, then click on Create user.



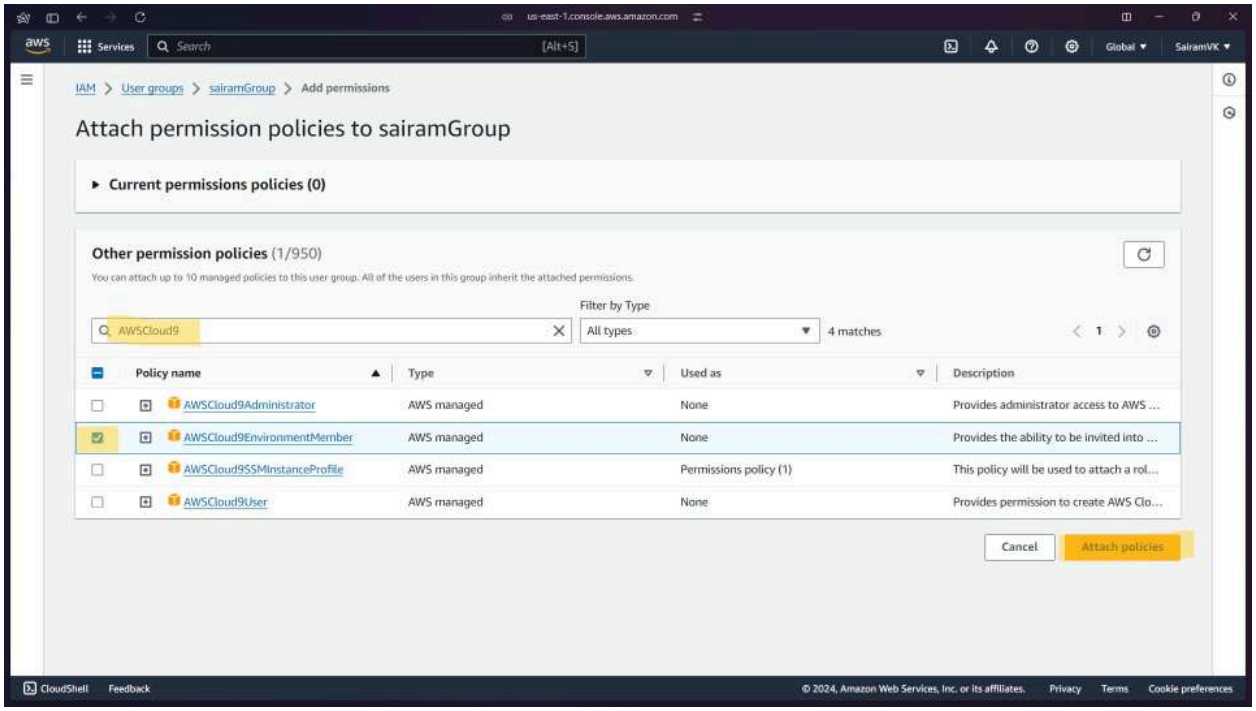
7) Open User Groups tab from the left side option. Click on the name of your group.



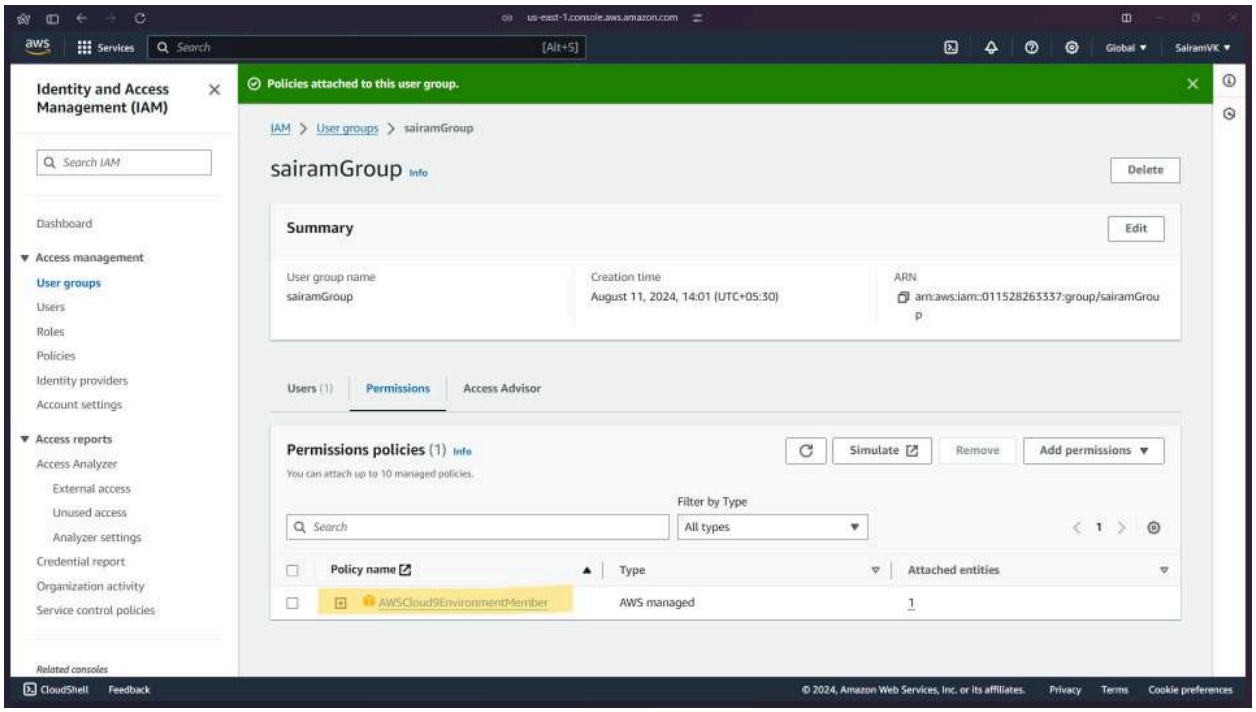
8) Go to permissions and click on Add permissions. Click on Attach Policies.



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

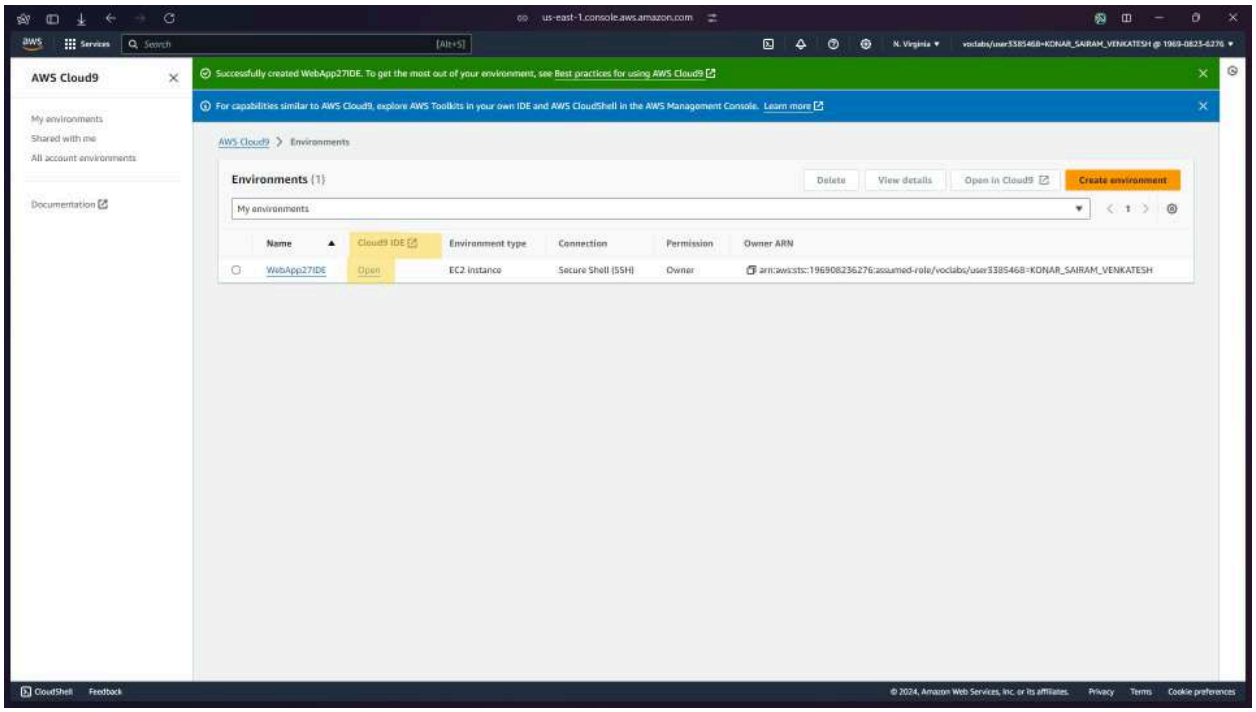


10) The policies have been attached.

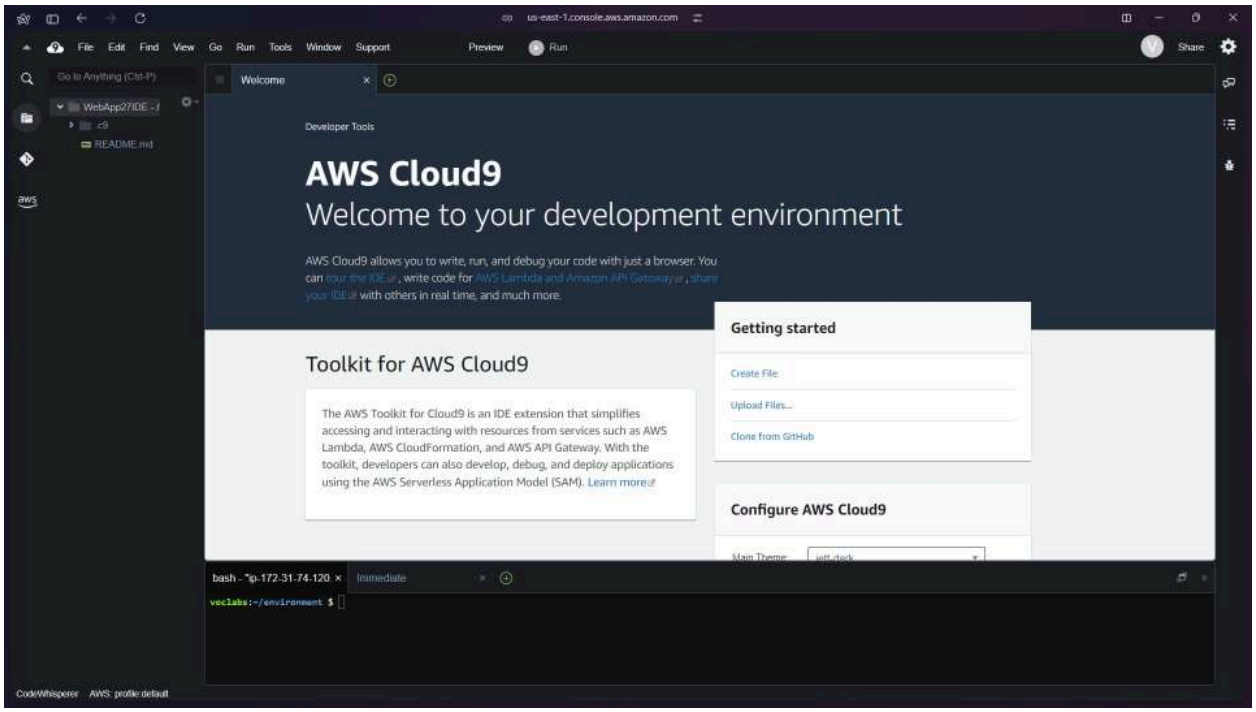


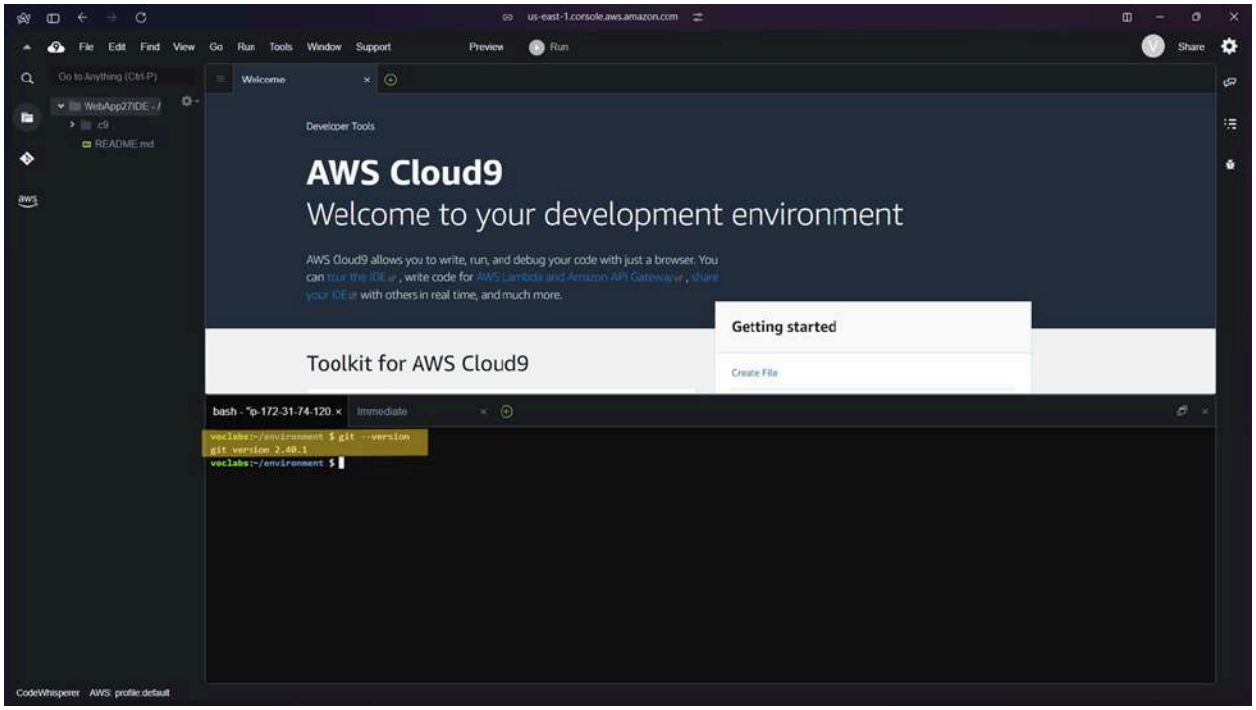
Step 3: Working on Cloud9 IDE

1) Go to Cloud9 services. Click on Open under Clou9 IDE.

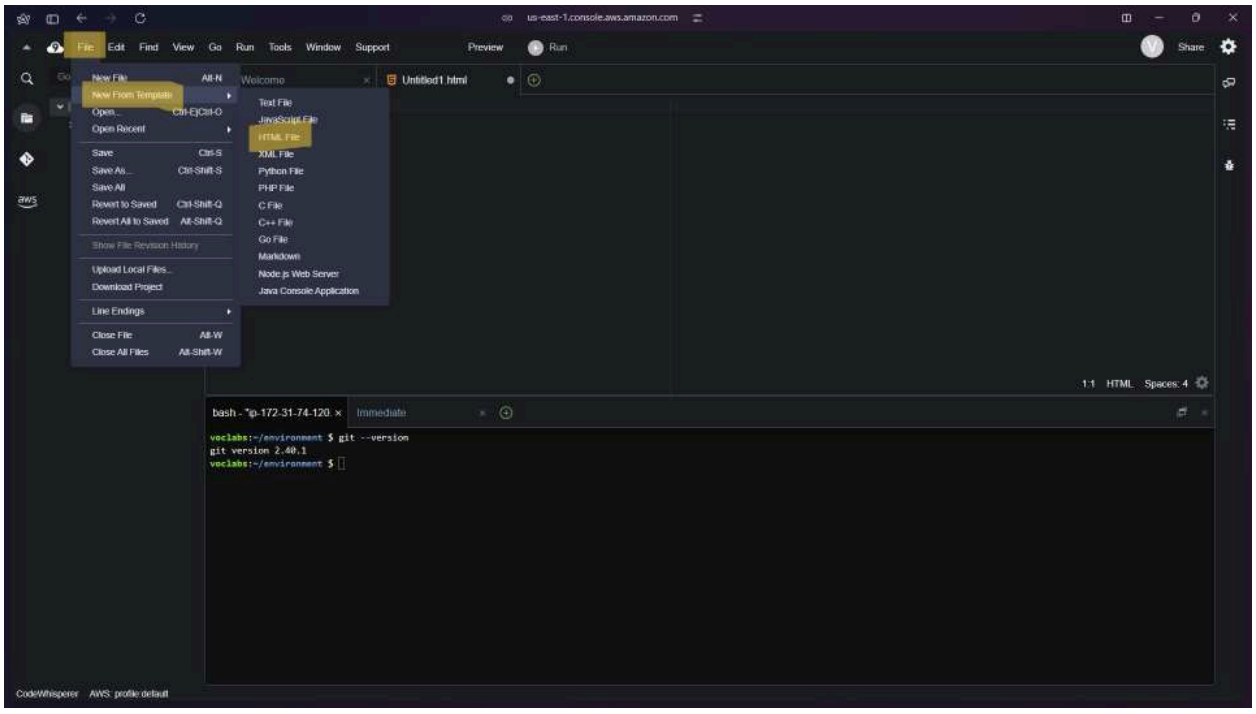


2) This is the Cloud9 IDE interface. The major part of the screen is the coding IDE. There is a command console just below it. For example, the command git --version is run.

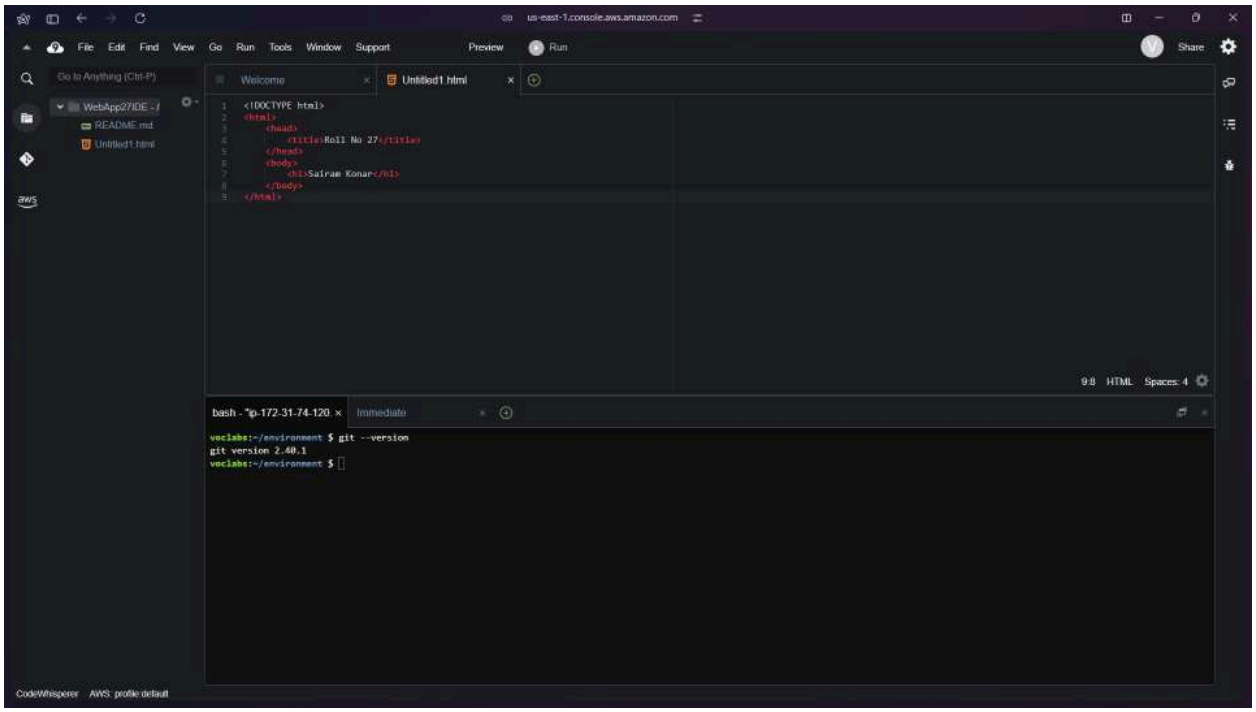




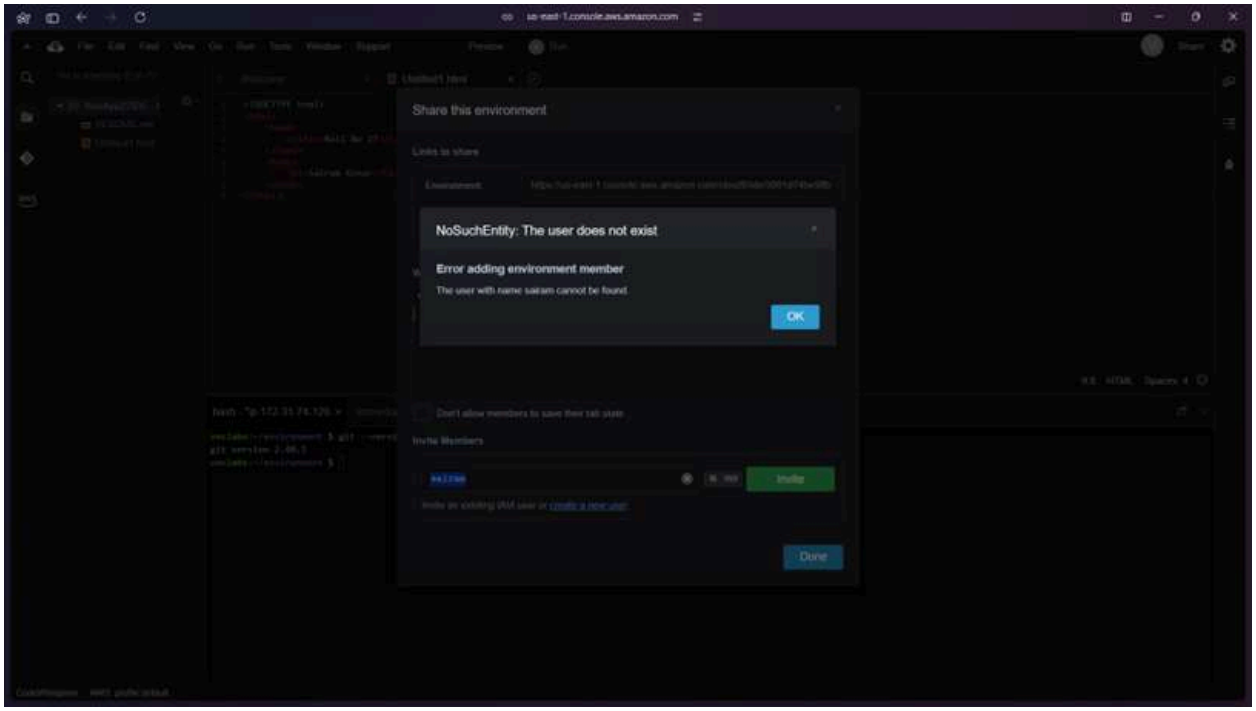
3) To add a file, click on file. For this experiment, we are to add an HTML file.
So go to File → New From Template → HTML file. This gives a basic HTML template on the coding IDE.



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 entered 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, while 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.

Conclusion:

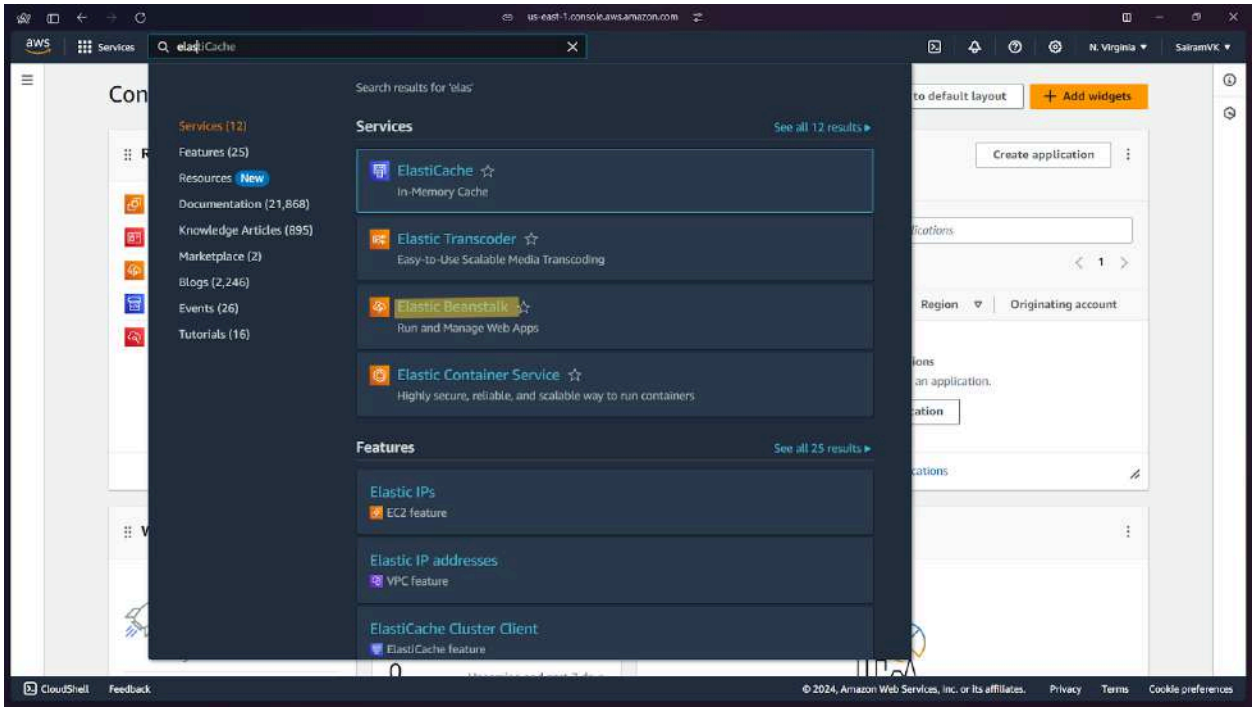
This experiment highlights how to set up and work with AWS Cloud9, although account limitations on AWS Academy prevent full collaboration demonstration. By understanding the steps and limitations, users can better leverage cloud infrastructure for development and teamwork when fully functional accounts are available.

Sairam Konar D15C 27 2024-25

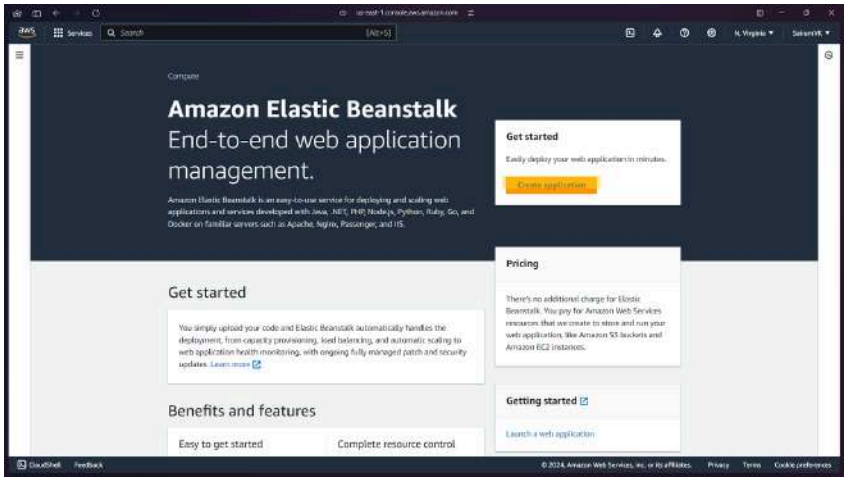
Experiment 2

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

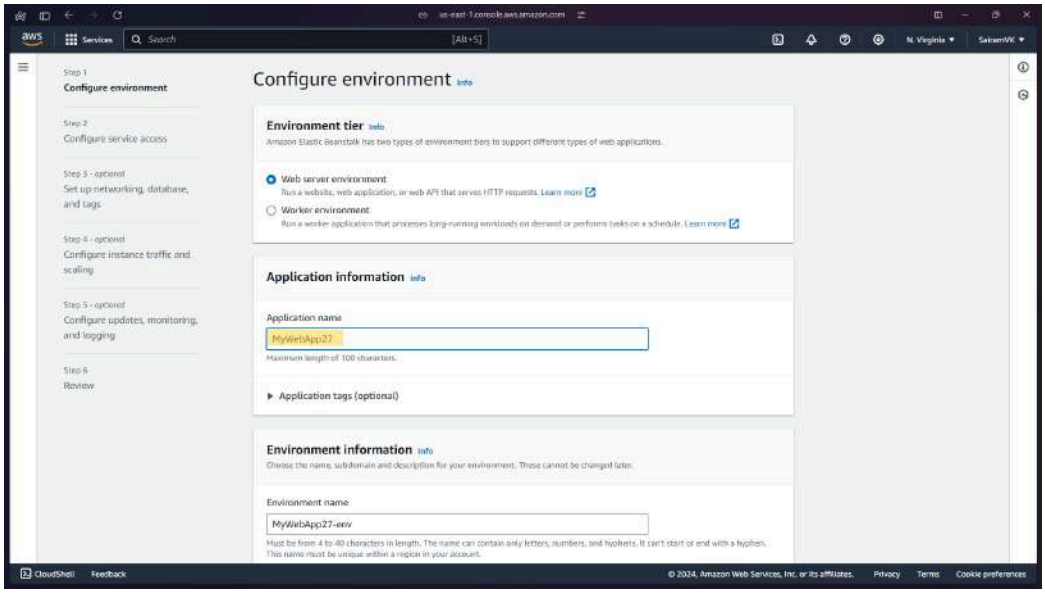
Step 1: Login to your AWS console. Search for Elastic Beanstalk in the searchbar near services.

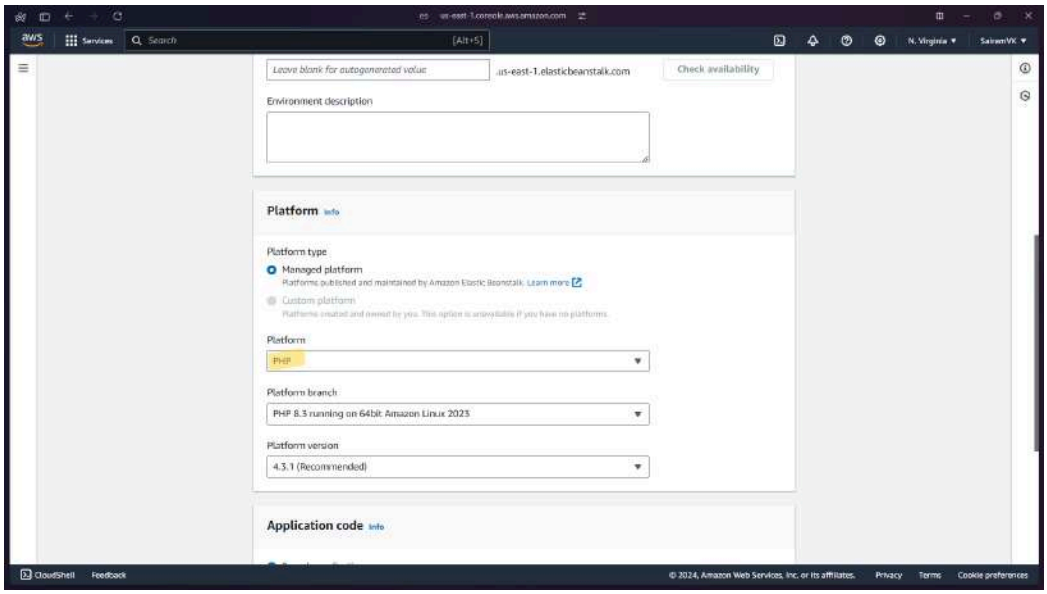


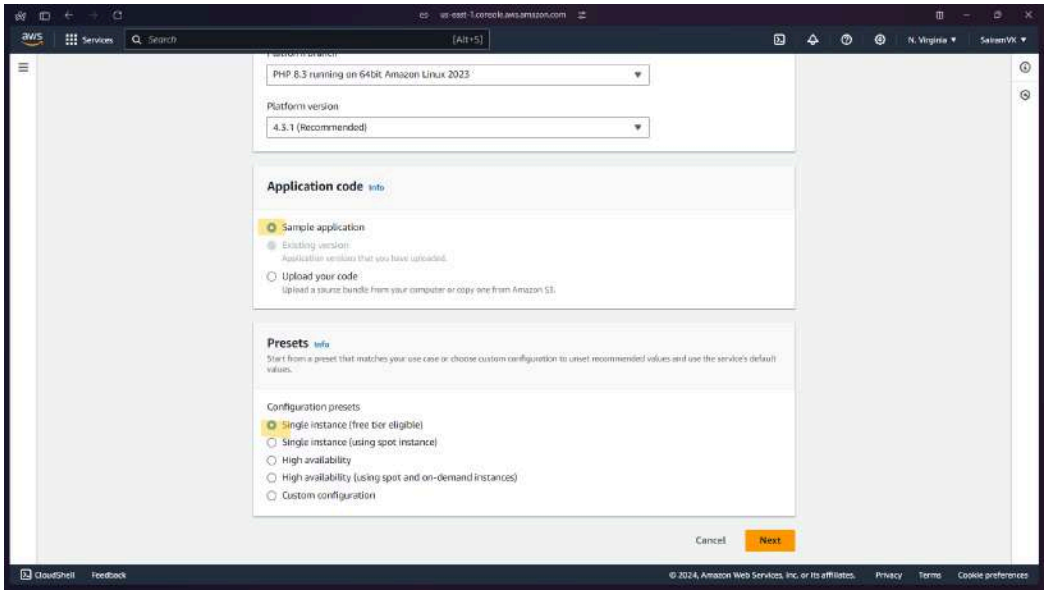
Step 2: Go to Elastic Beanstalk and click on Create Application



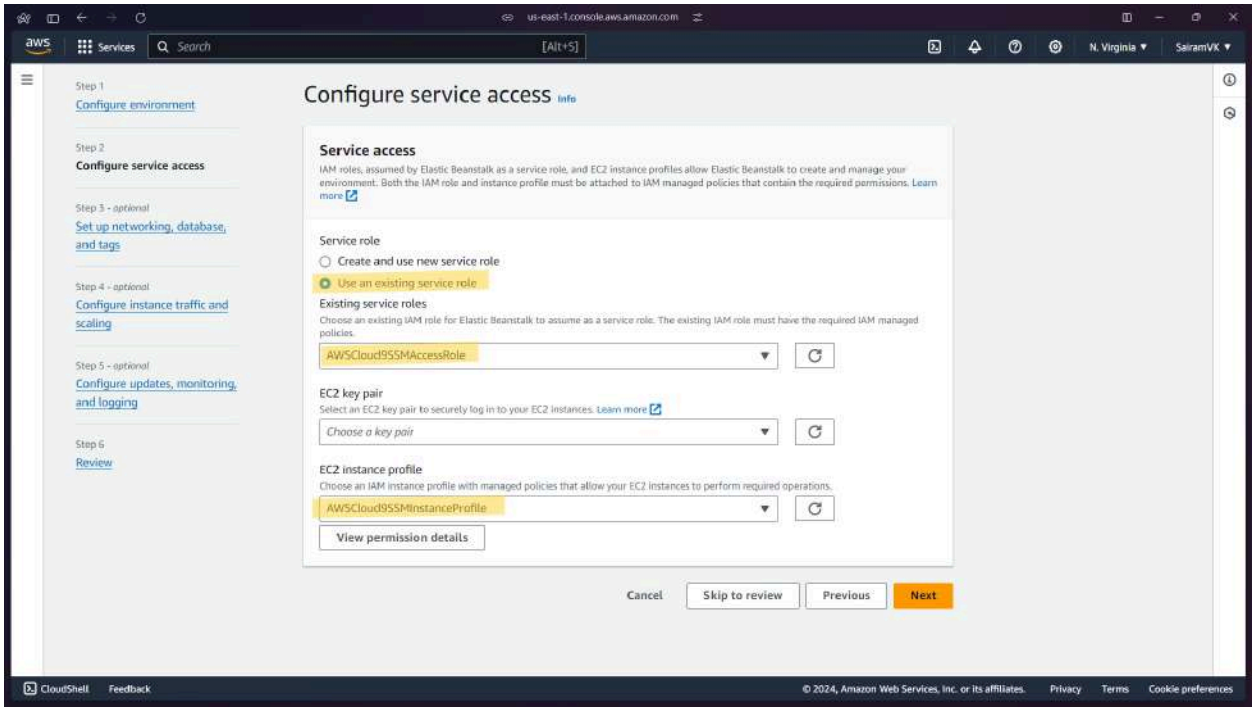
Step 3: Enter the name of your application. Scroll down and in the platform, select platform as PHP. Keep the application code as Sample Application. Set the instance to single instance. Click on NEXT.



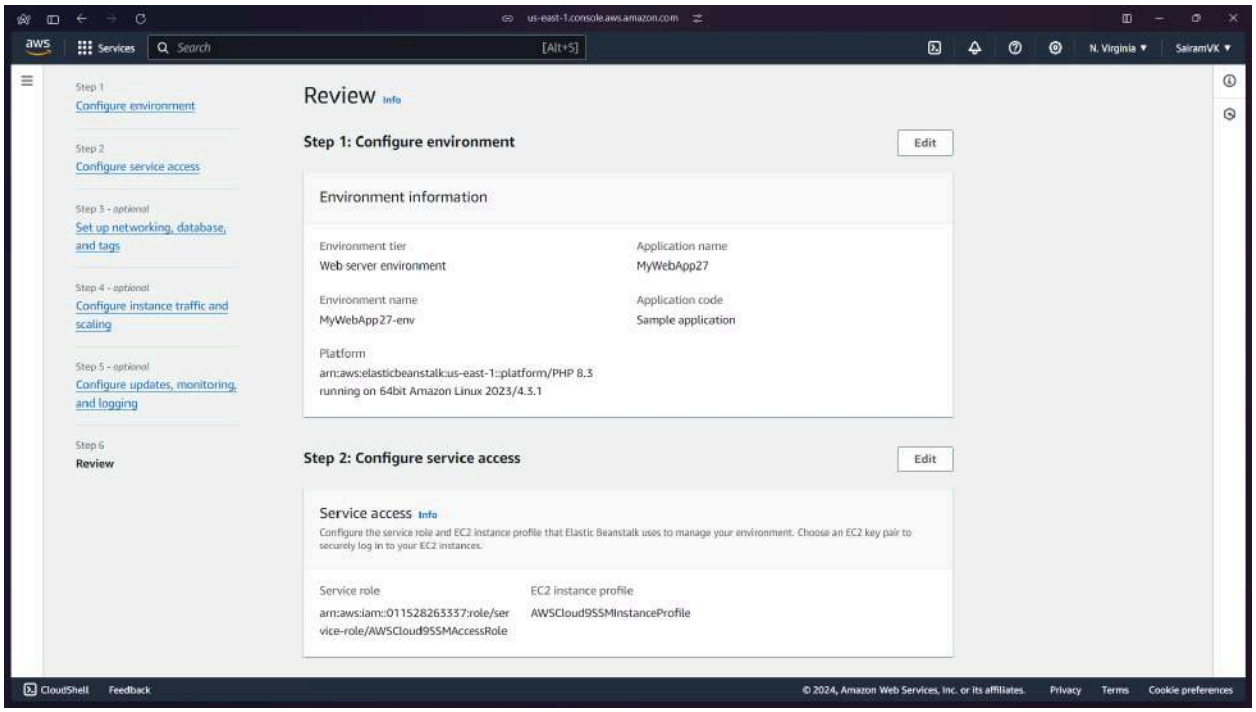


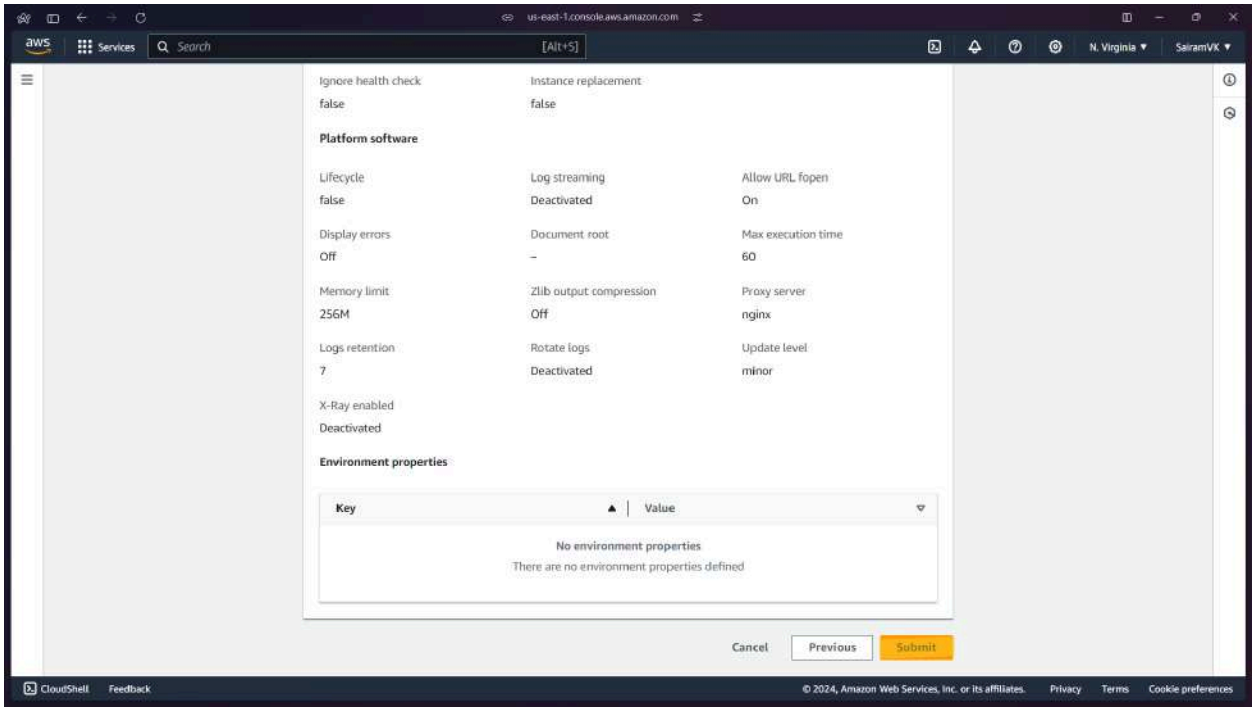


Step 4: Use an existing service role and choose whatever service role is available on your account.

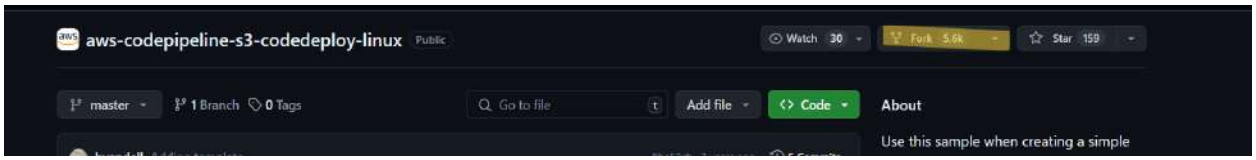


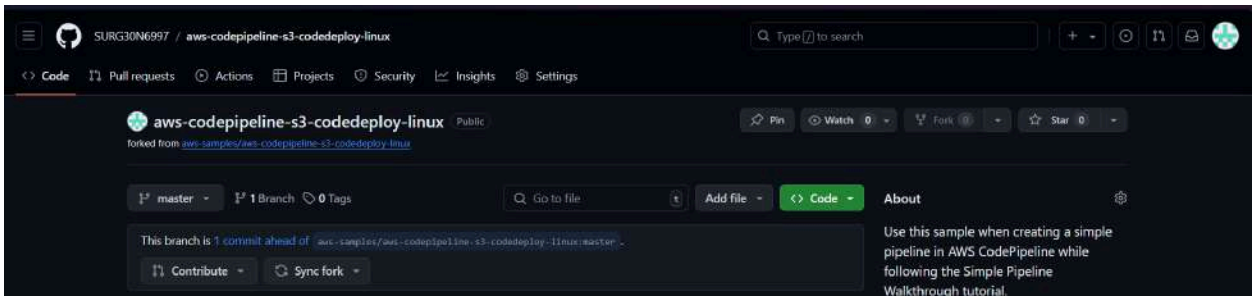
Step 5: Review the settings that you have set up for your application and submit your application.



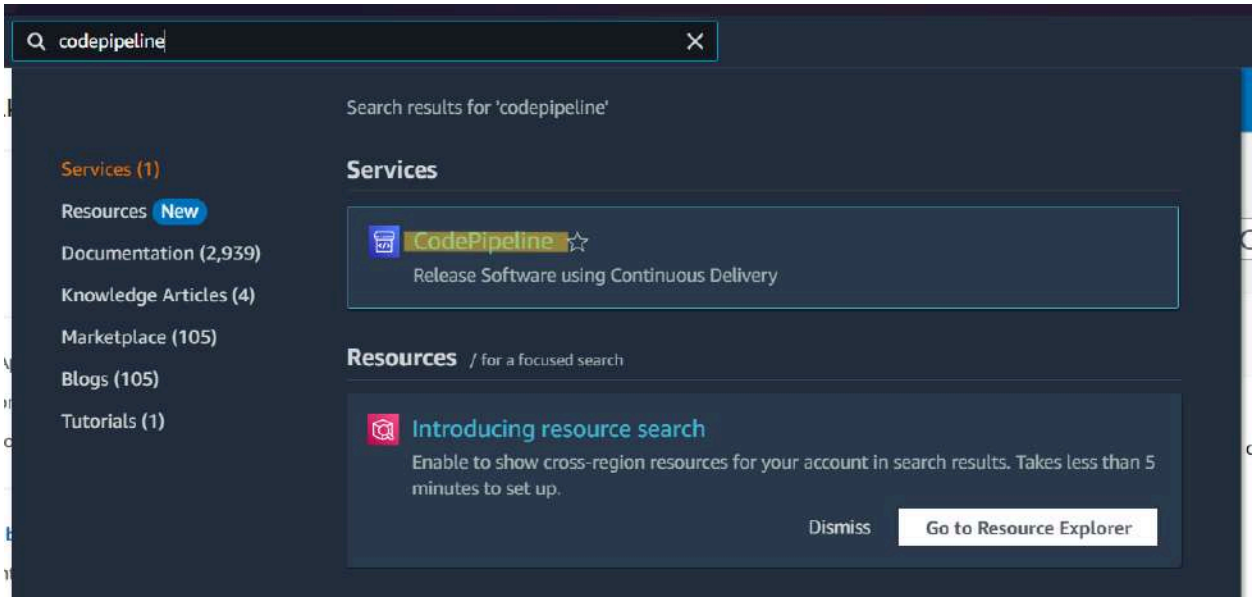


Step 6: Go to the github link below. This is a github with a sample code for deploying a file on AWS CodePipeline. Fork this repository into your personal github.
https://github.com/aws-samples/aws-codepipeline-s3-codedeploy-linux

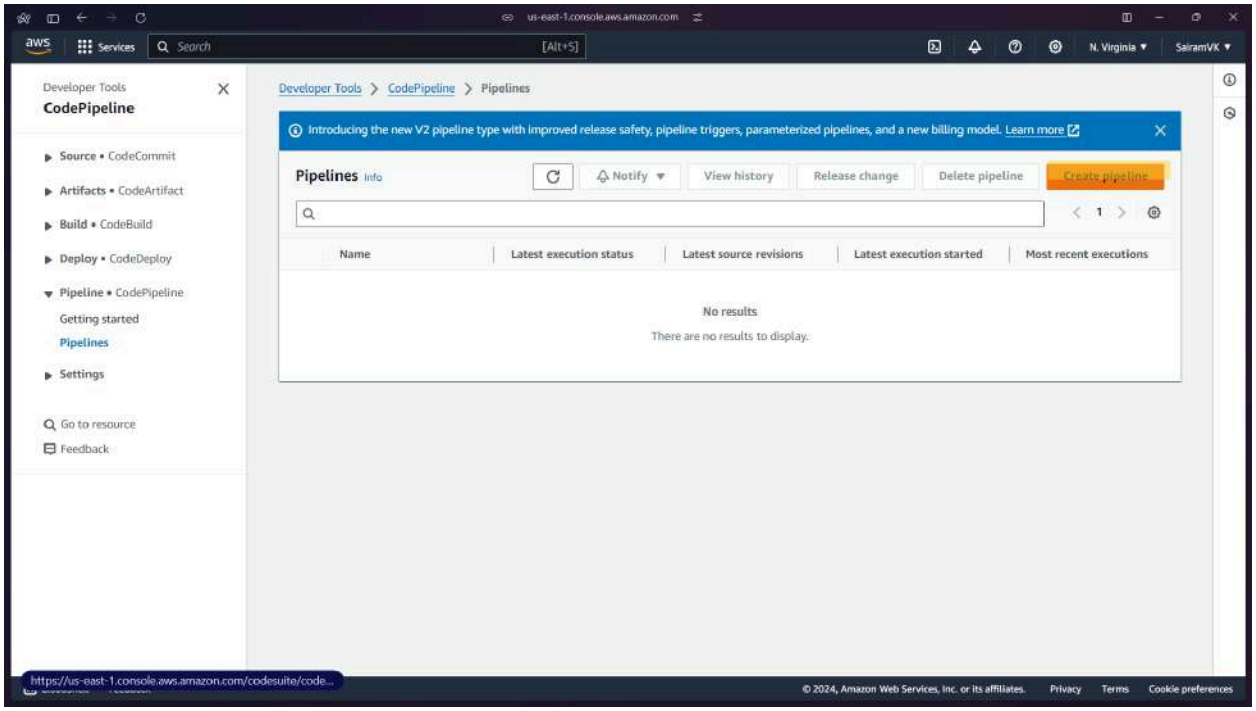




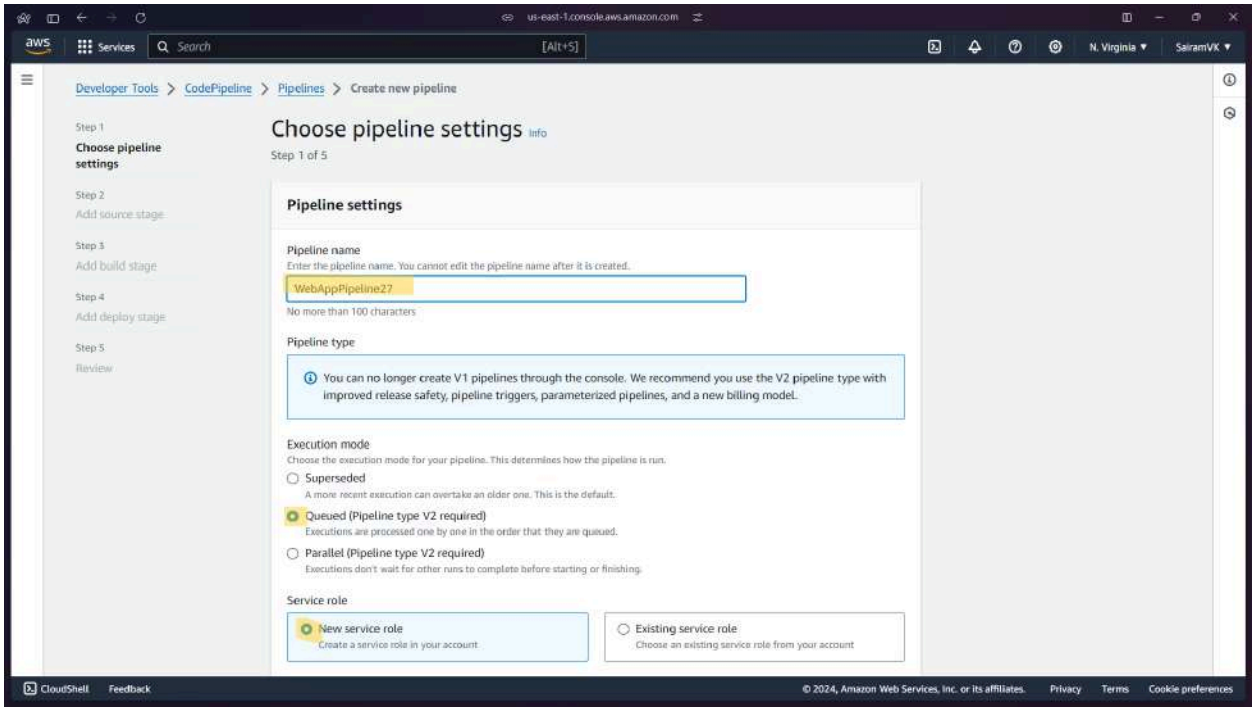
Step 7: Search CodePipeline in the services tab and click on it.

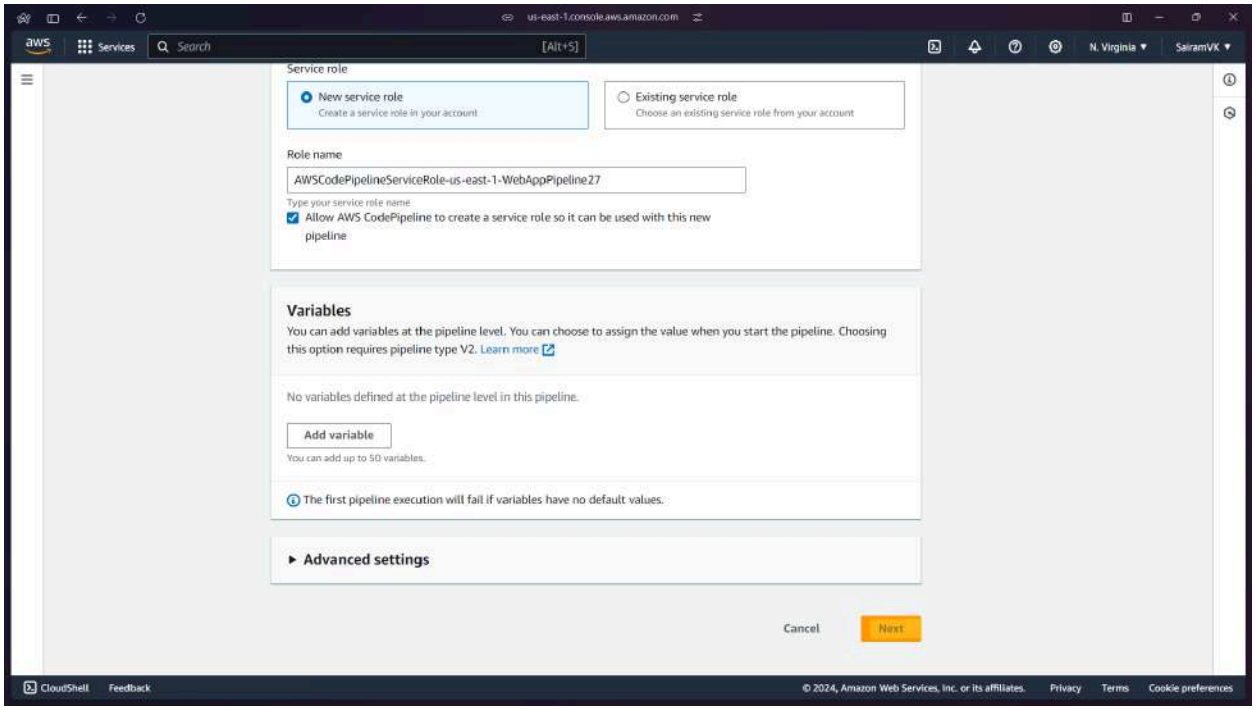


Step 8: Click on Create Pipeline.

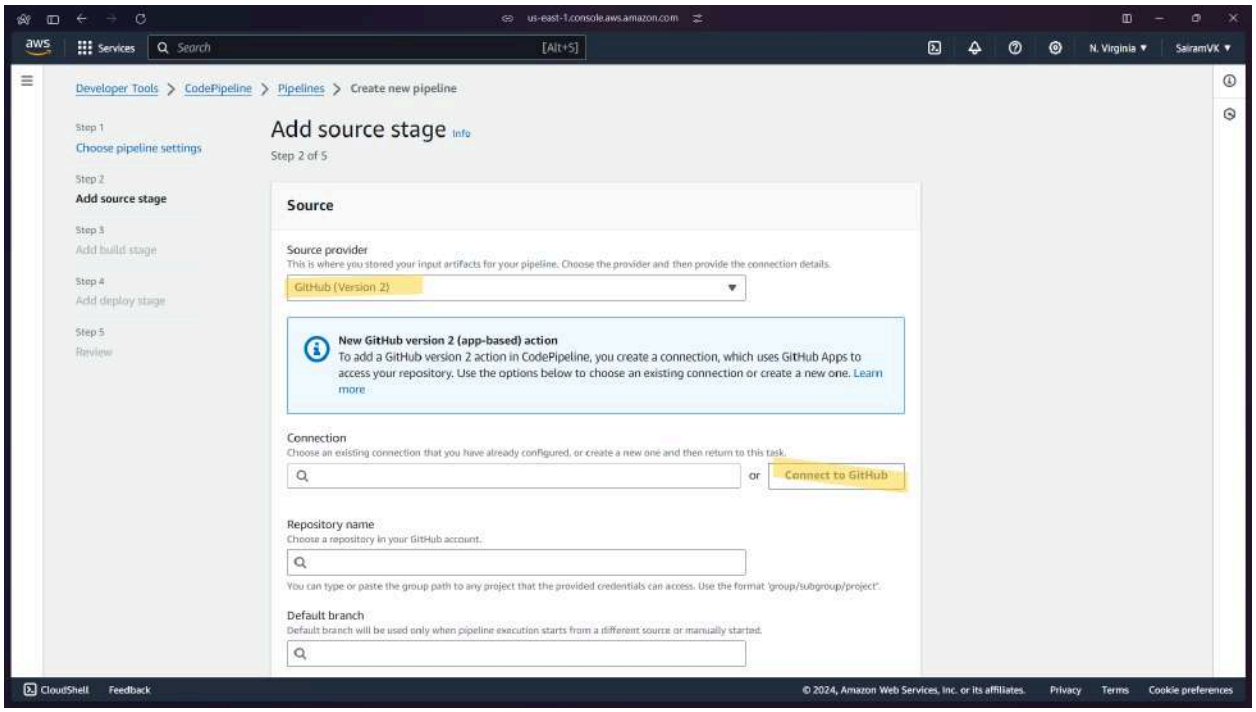


Step 9: Give a name to your Pipeline. A new service role would be created with the name of the Pipeline.

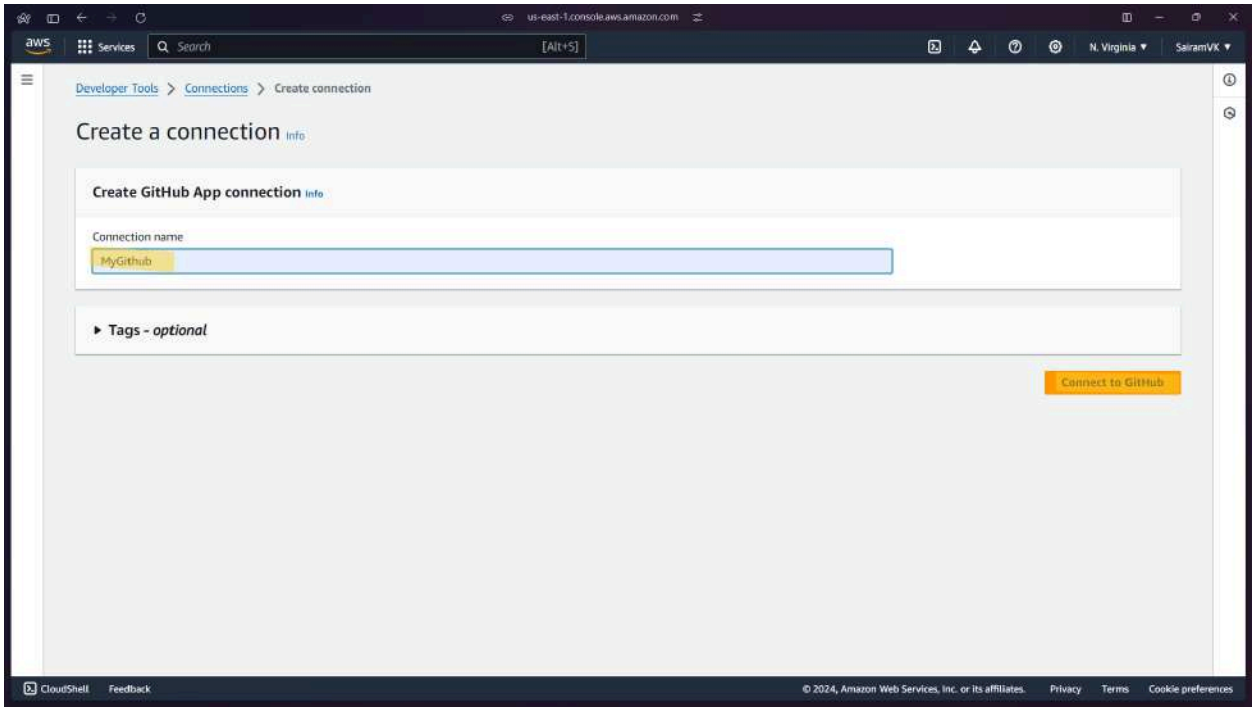


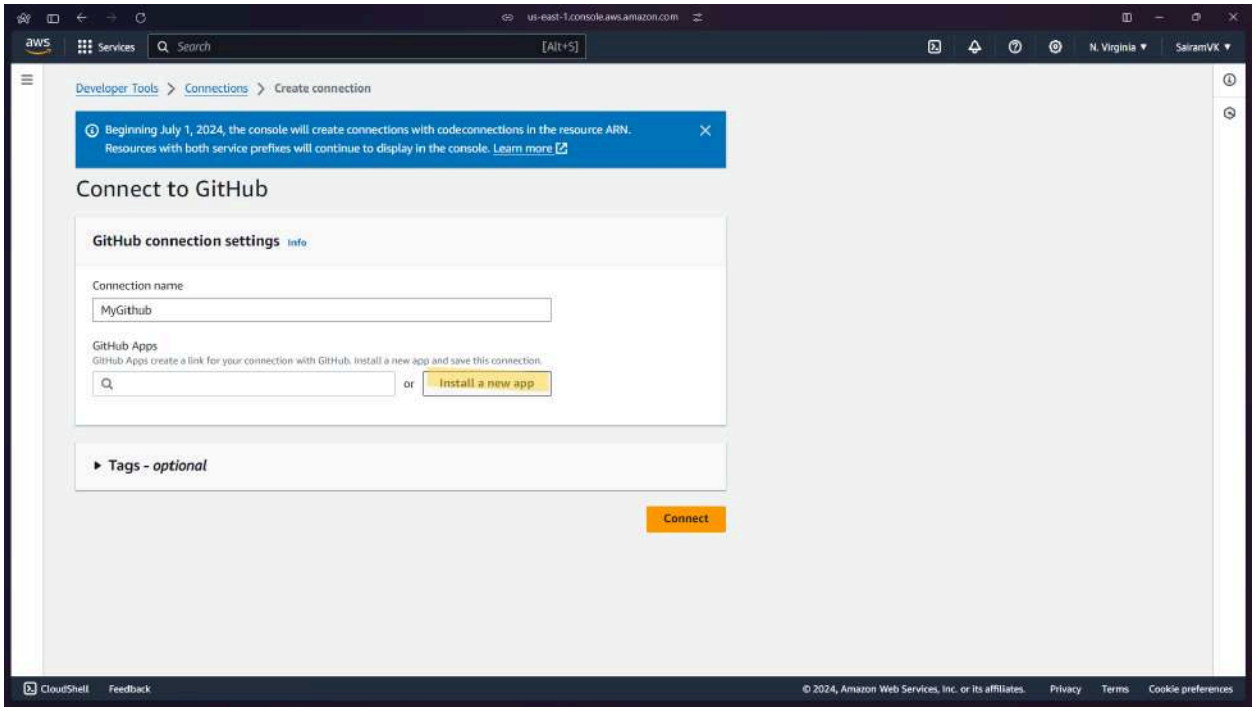


Step 10: Select a source provider (as Github (Version 2)). Click on Connect to Github to connect your github.

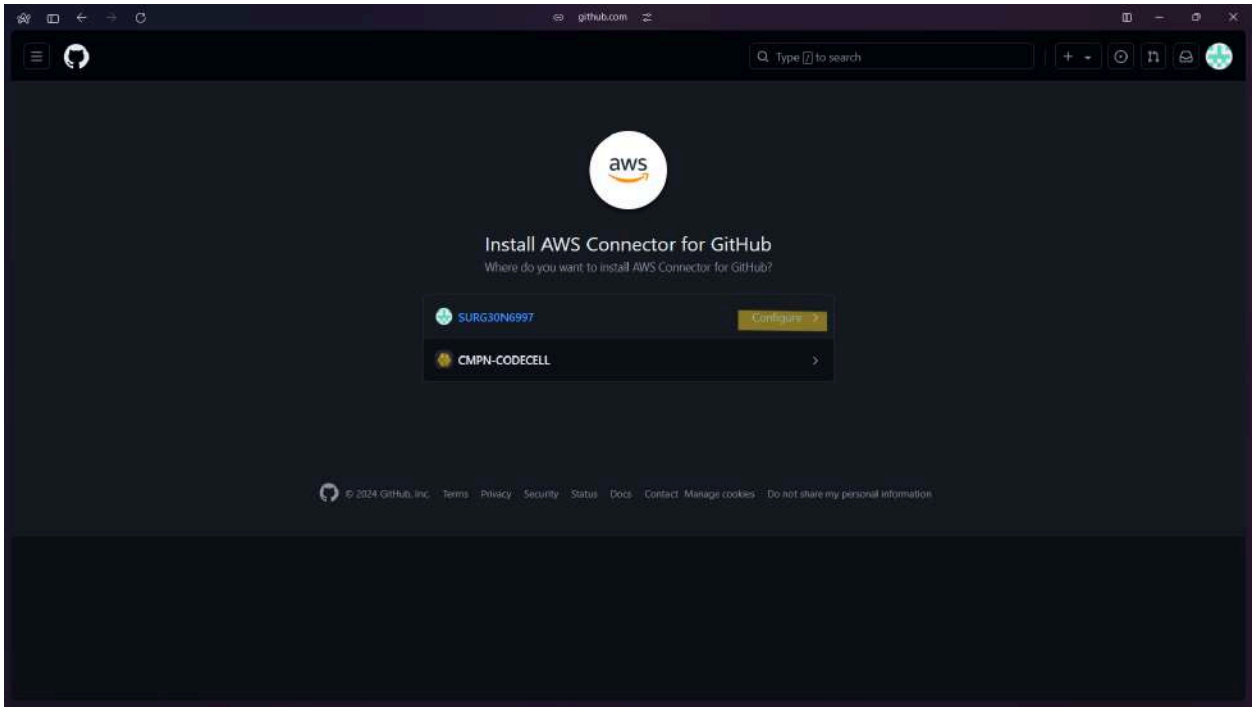


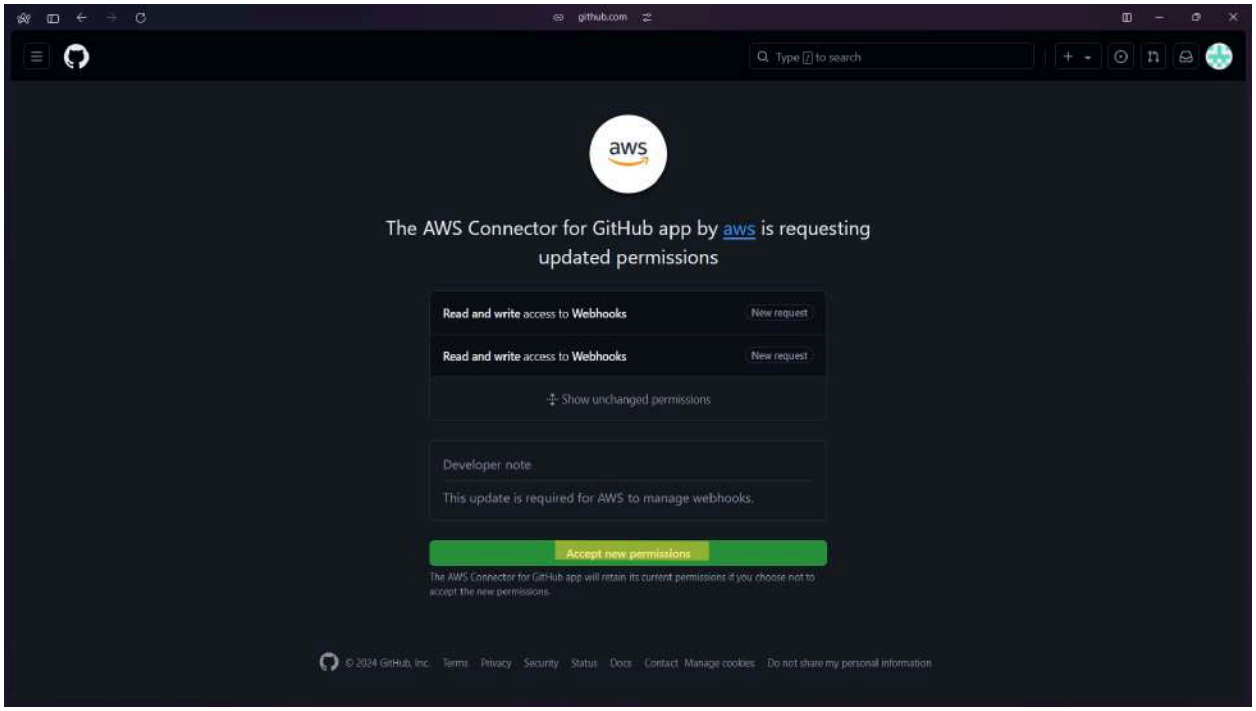
Step 11: Give a name to your GitHub app Connection and click on Connect. This will give you a prompt to either to select a GitHub app or install a new app. If this is your first time, click on Install a new app.



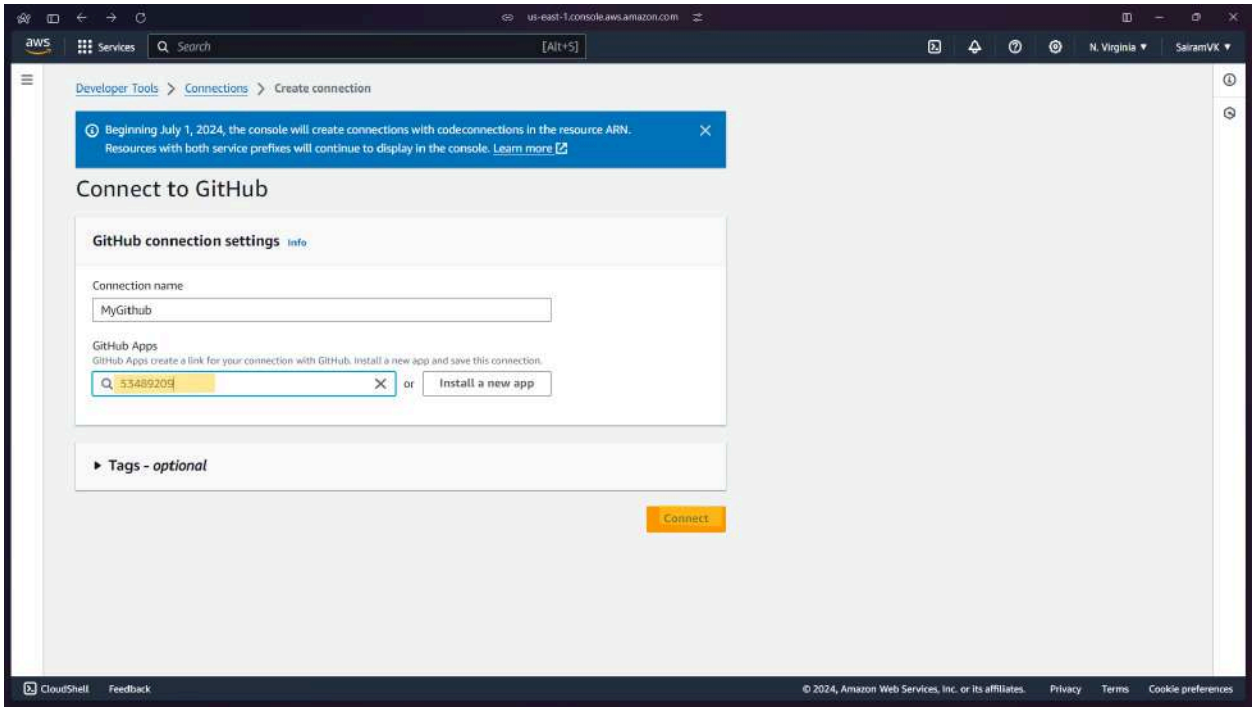


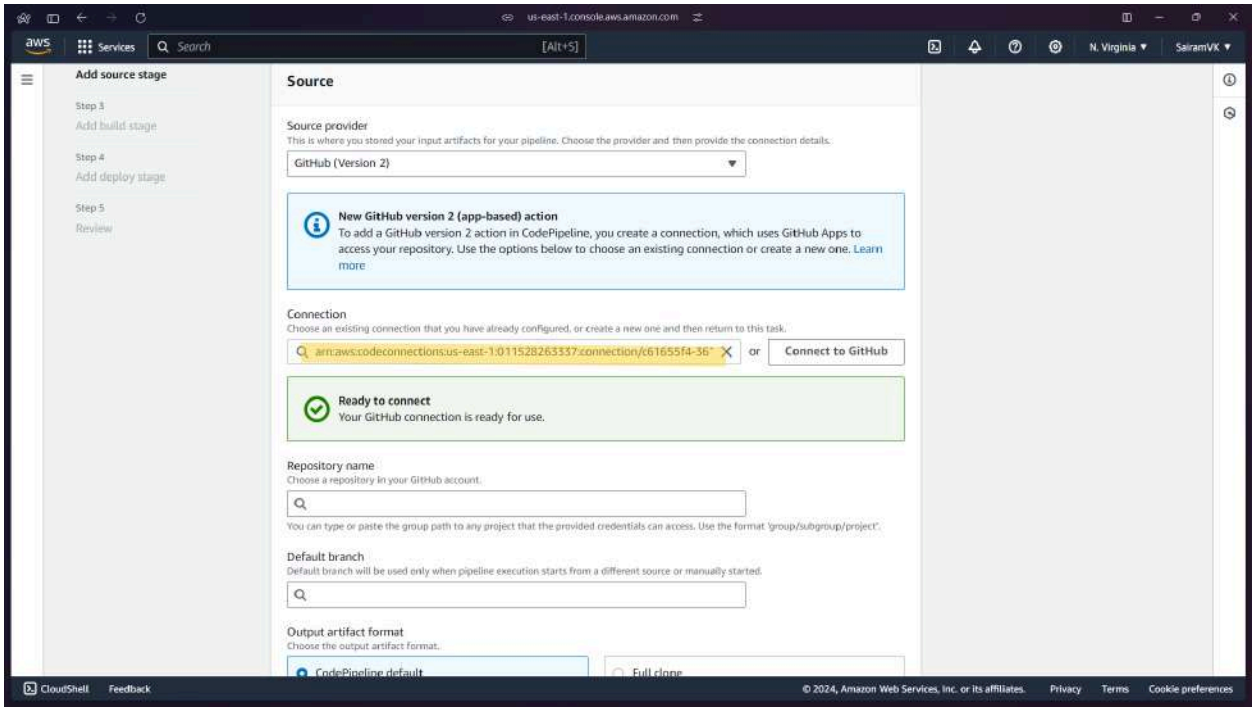
Step 12: This will direct you to install AWS Connector on your GitHub. Install it to your account and give it its permissions.



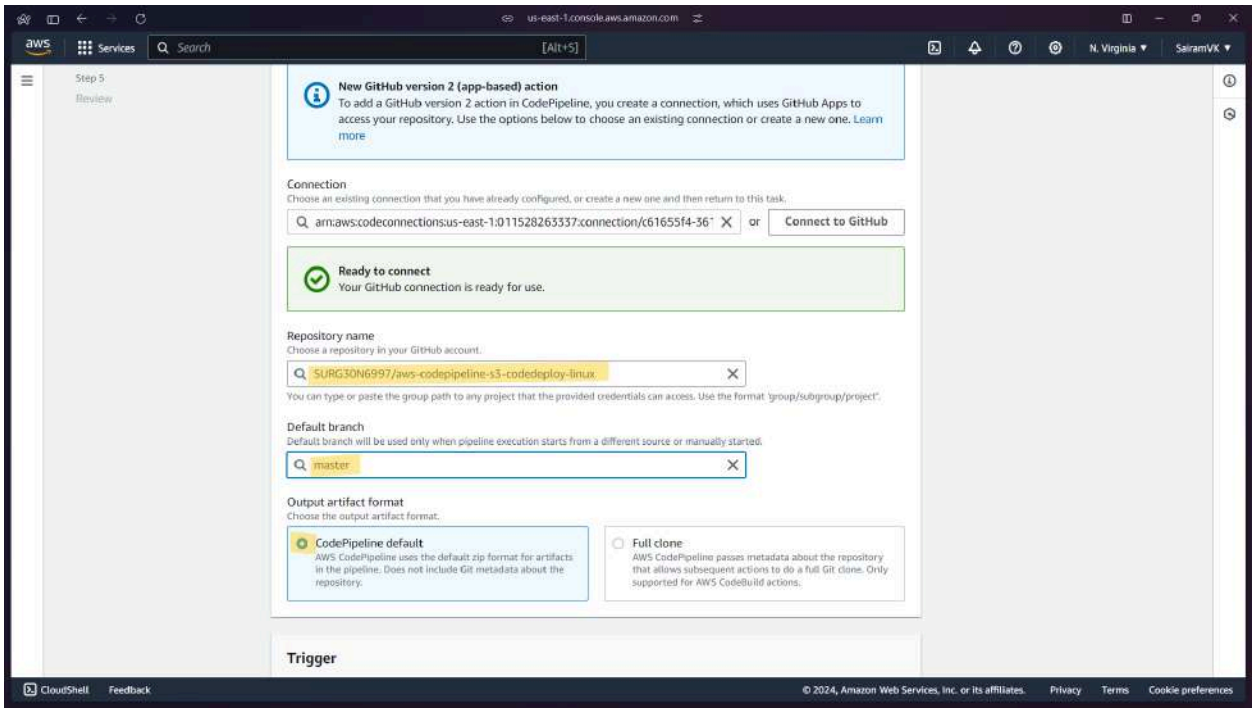


Step 13: After the app is set up, it gives the number in the text field. Click on Connect.
After clicking on connect, the link is shown in the connection field and AWS shows that GitHub connection is ready to use.

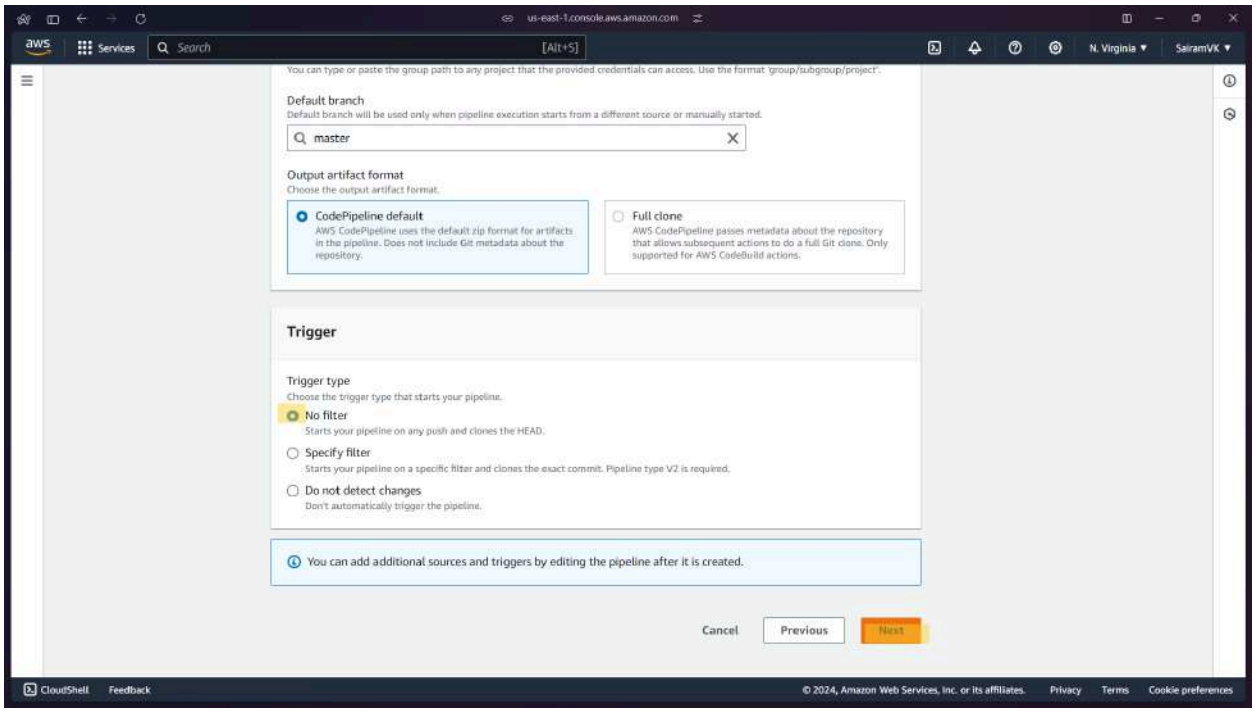




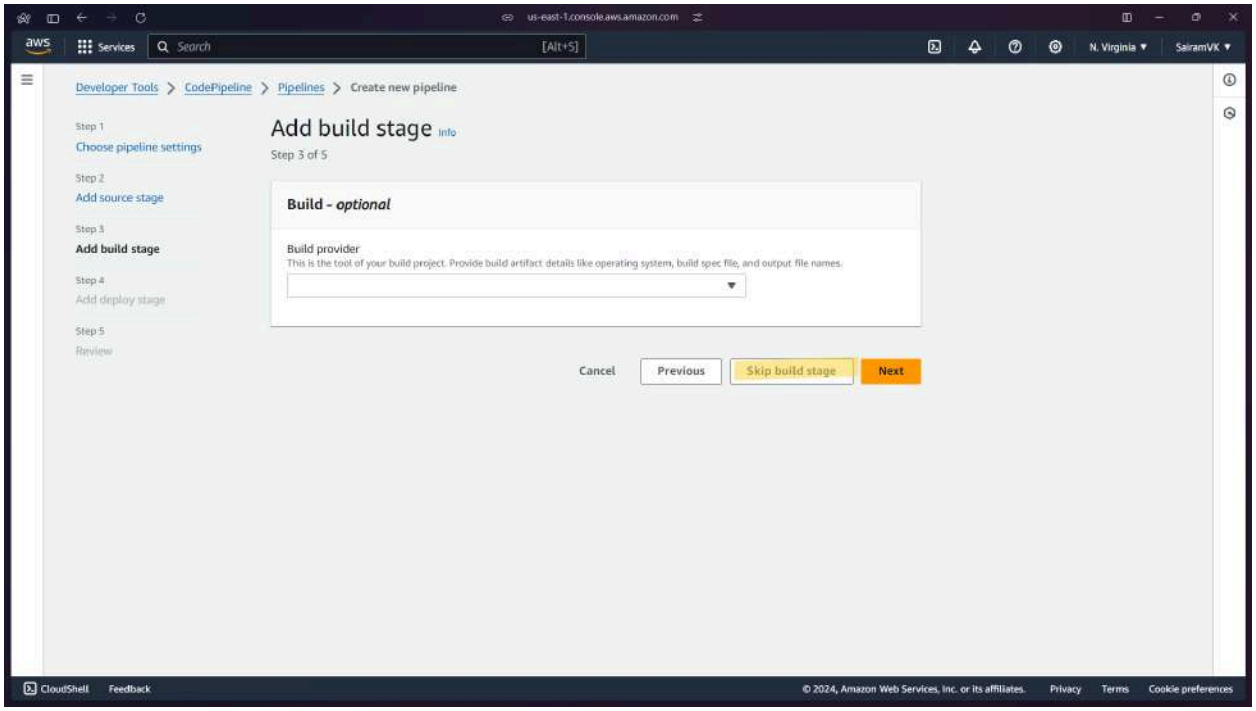
Step 14: Select the repository that you had forked to your GitHub. After that select the branch on which the files are present (default is Master).

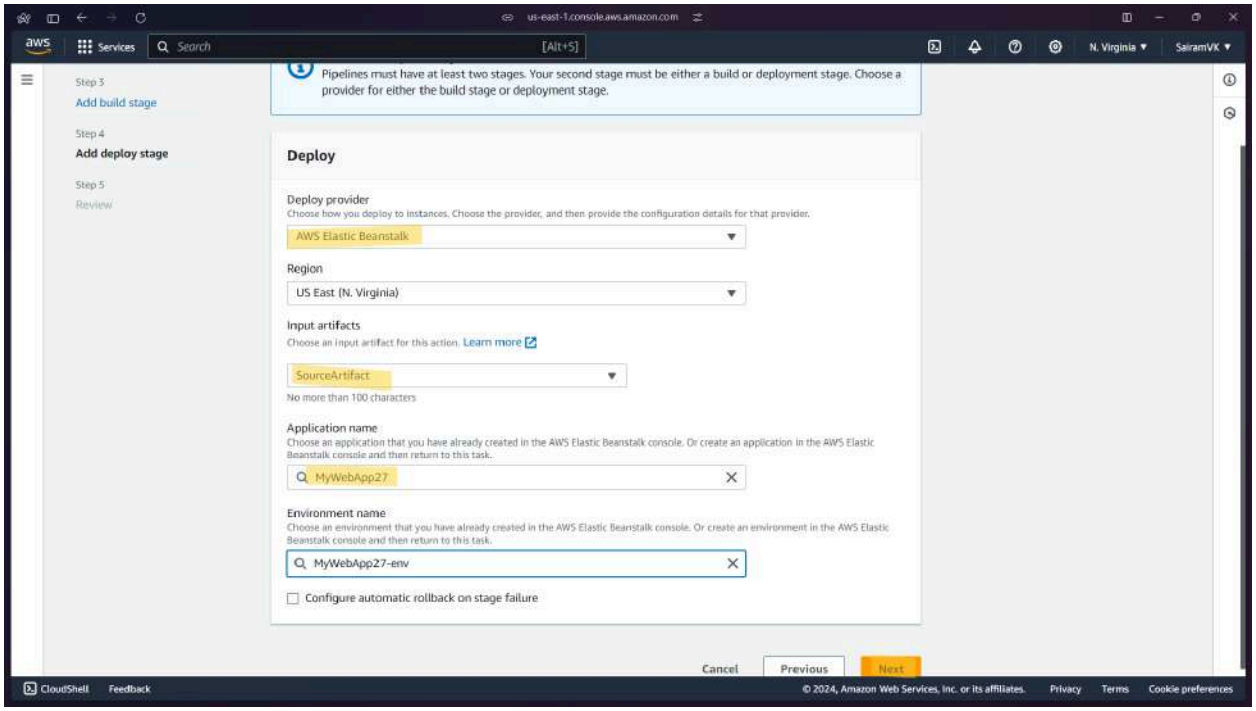


Step 15: Set the Trigger type as no filter. This would allow it to the website to update as soon as some change is made in the github.

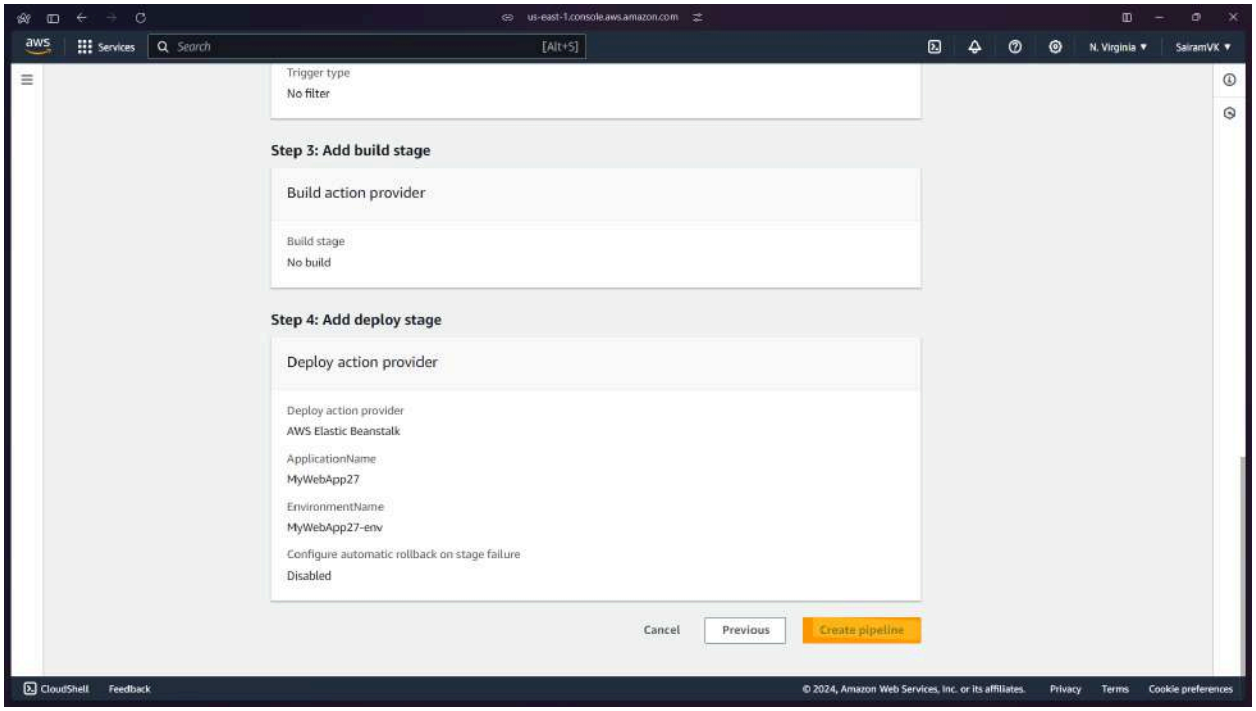


Step 16: Skip the build stage and go to Deploy. Select the deploy provider as AWS Elastic Beanstalk and Input Artifact as SourceArtifact. The application name would be the name of your Elastic Beanstalk. Then click on next.

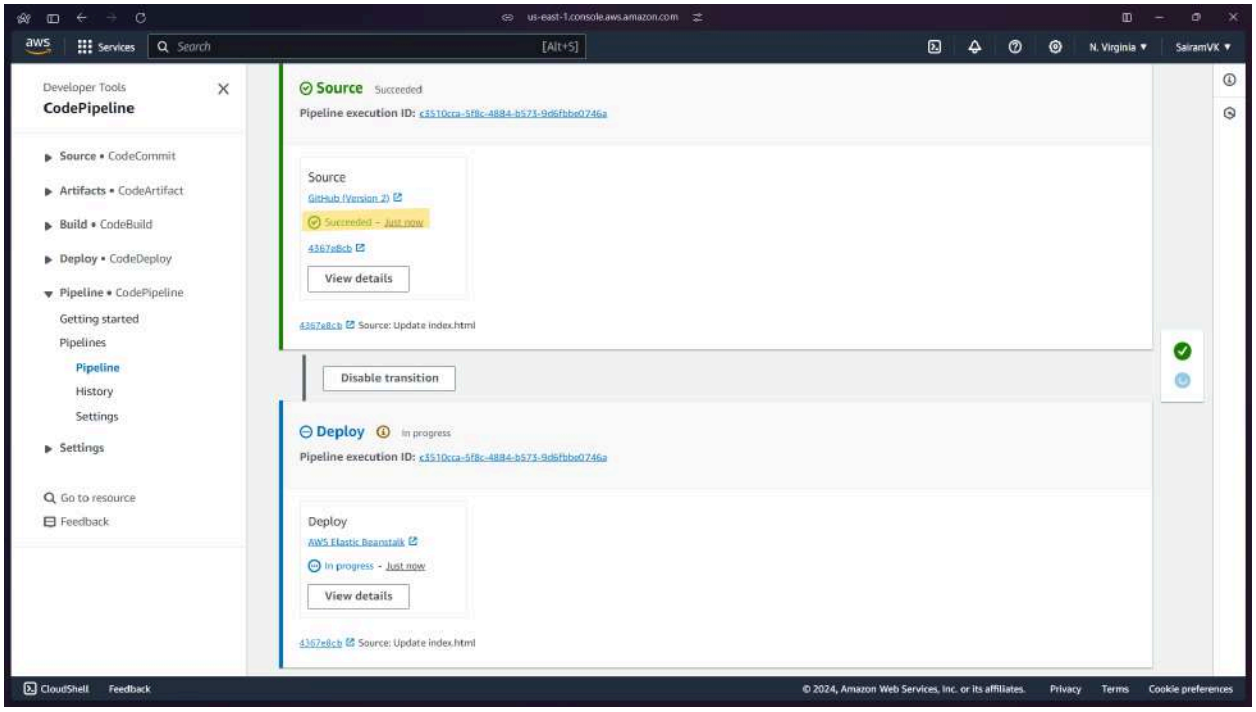




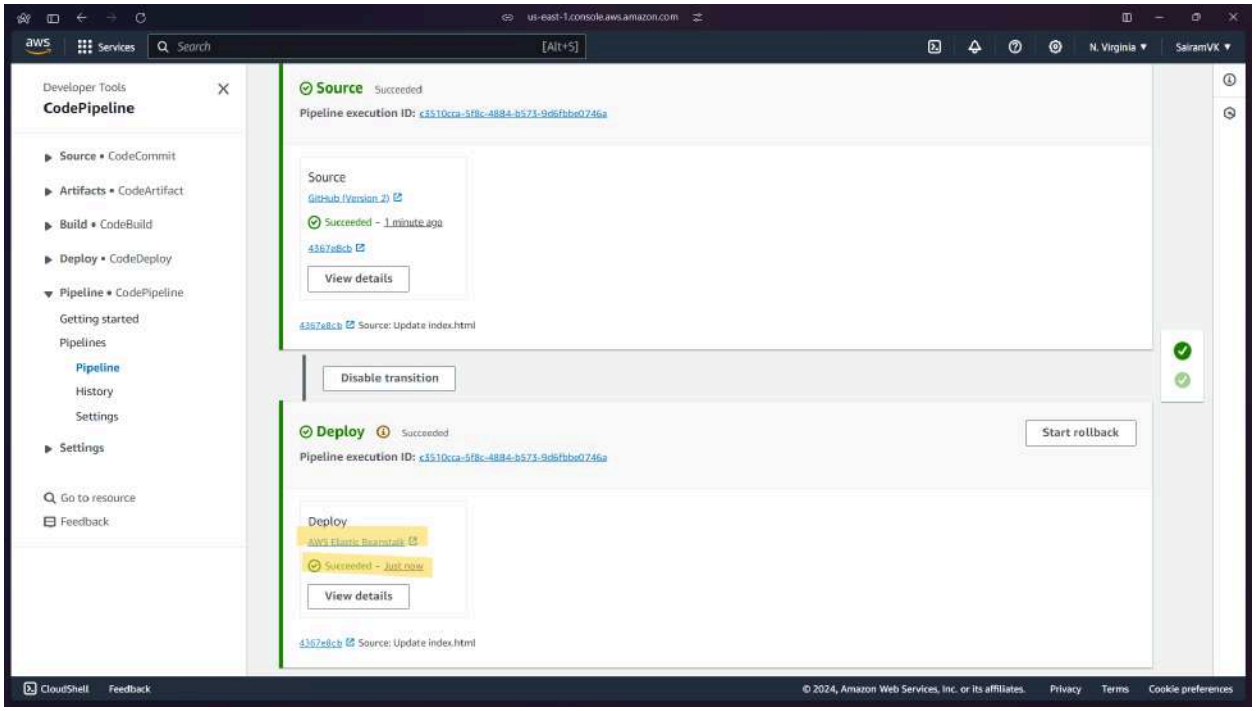
Step 17: Check all the information and click on create Pipeline.



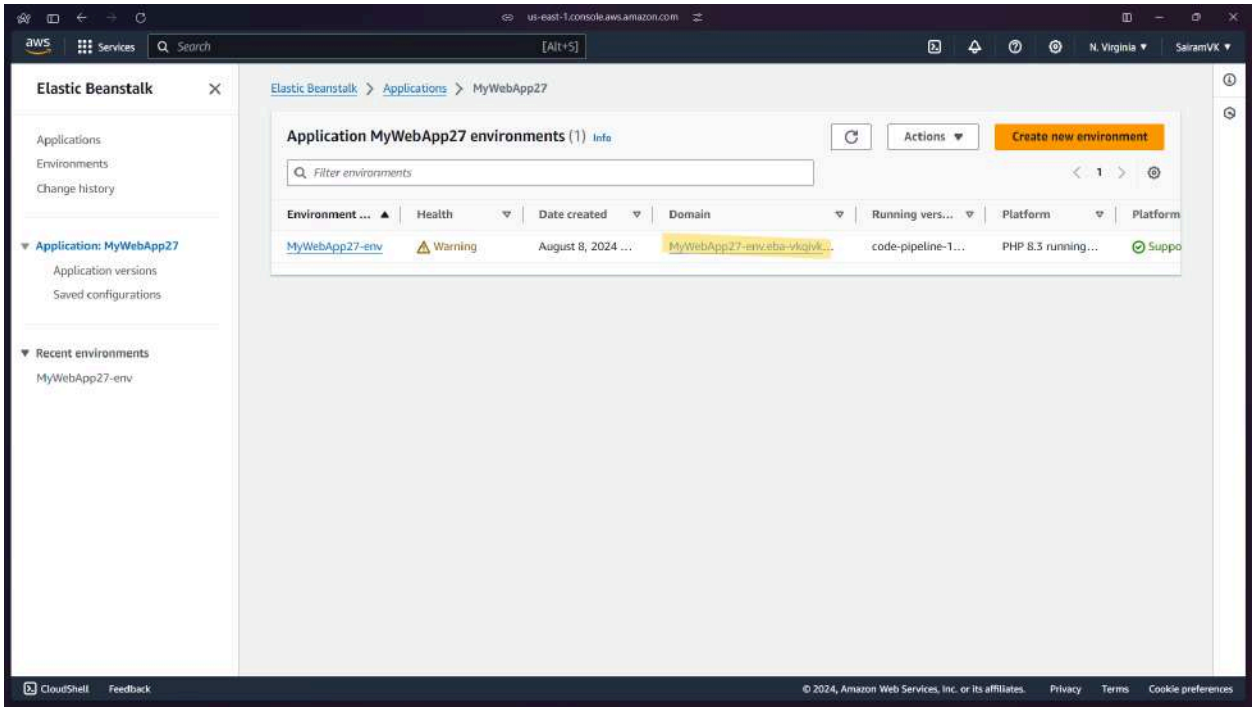
Step 18: If the pipeline is successfully deployed, this screen comes up where the source is set up and then it is transitioned to deploy.



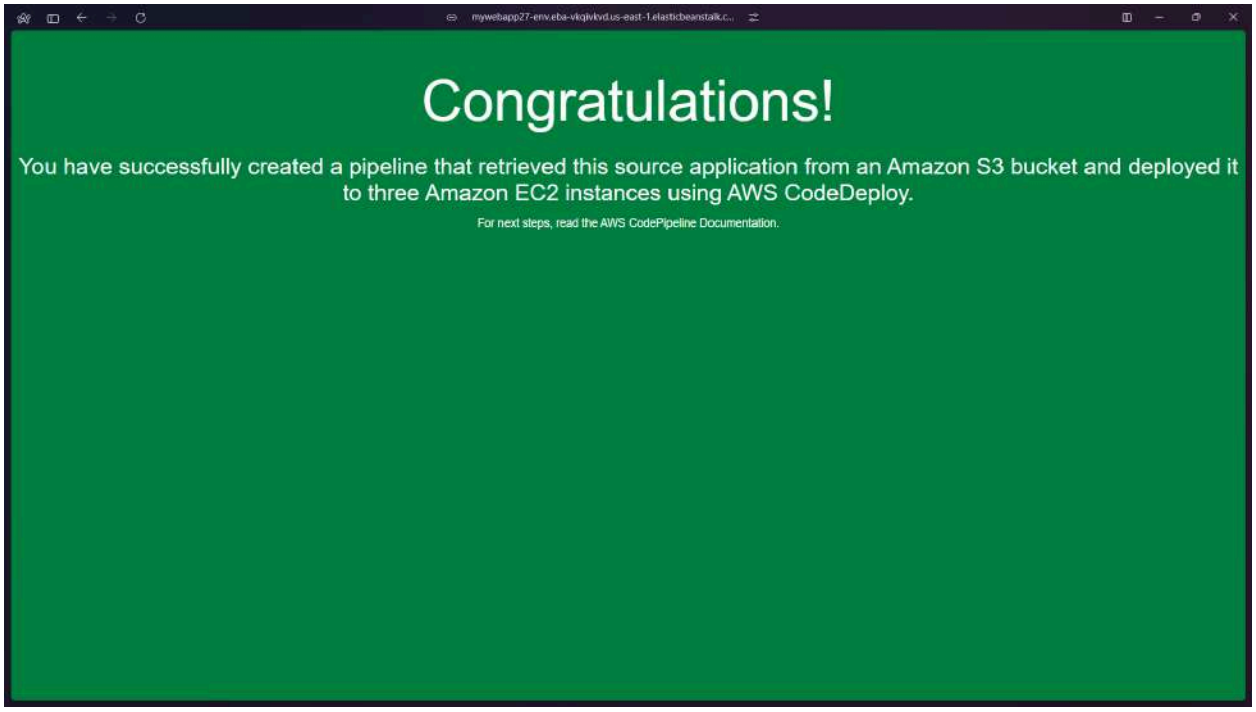
Step 19: Once the deployment is complete, click on the AWS Elastic Beanstalk under Deploy.



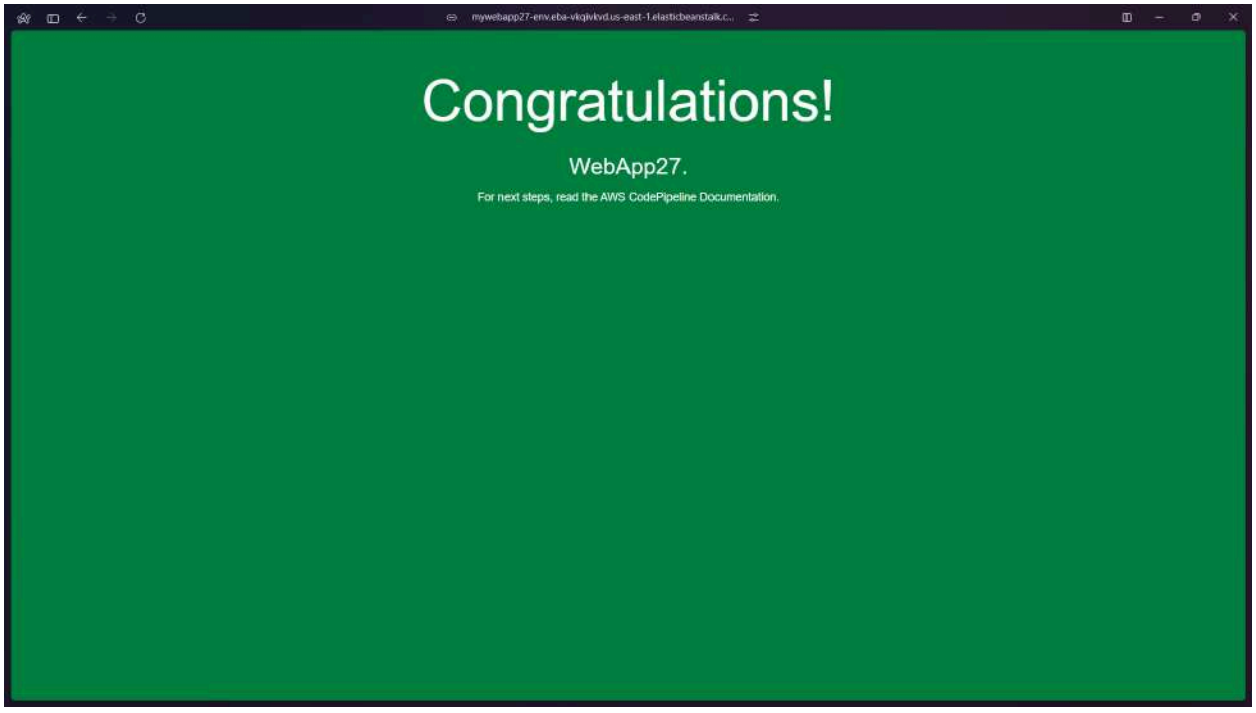
Step 20: This will redirect you to the application screen of Elastic Beanstalk. Click on the link shown under Domain.



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.



Conclusion: This process showcases the ability to automate the entire lifecycle of building, deploying, and managing updates for an application using AWS services. By utilizing AWS CodeBuild, AWS CodePipeline, and Elastic Beanstalk, you streamline the deployment of a sample application and ensure continuous delivery. Additionally, by integrating GitHub, any changes to the source code are automatically deployed, making the process efficient and scalable. This approach simplifies application management, reduces manual intervention, and enhances collaboration, making it an ideal choice for modern cloud-based software development.

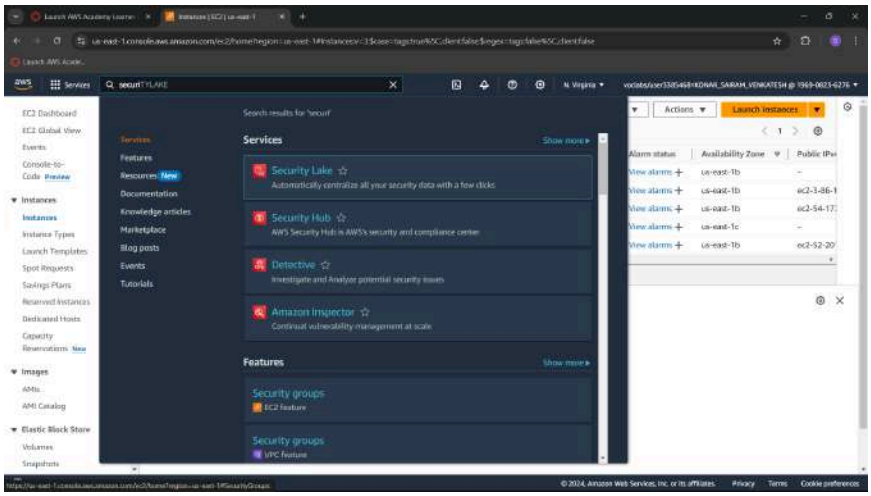
Sairam Konar D15C 27 2024-25

Experiment 3

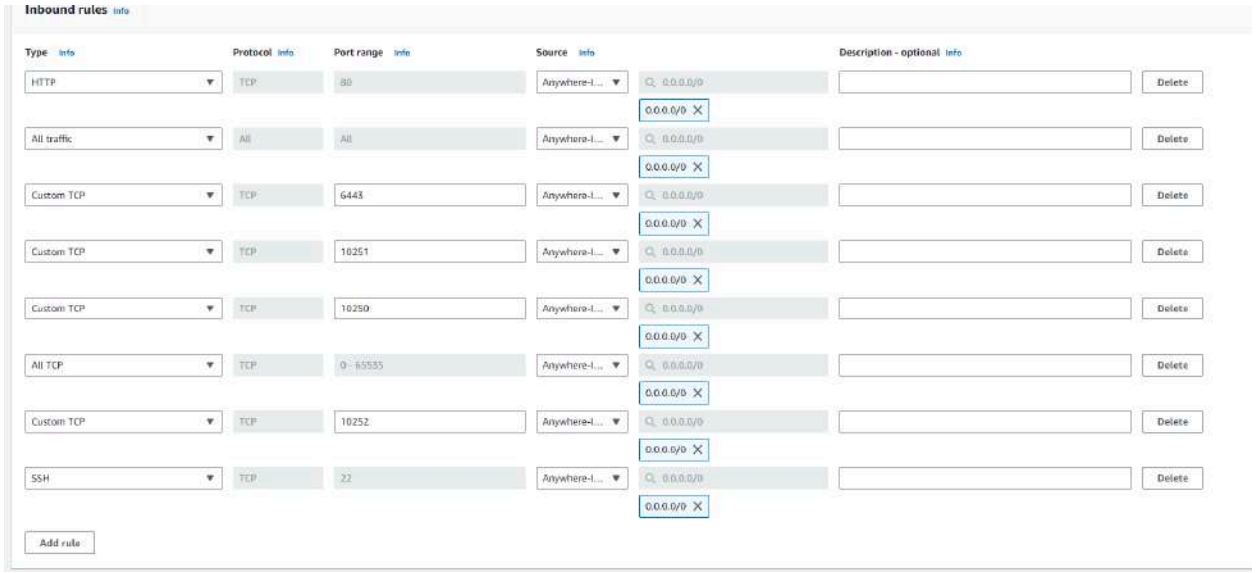
Aim: To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud Platforms.

Prerequisites:

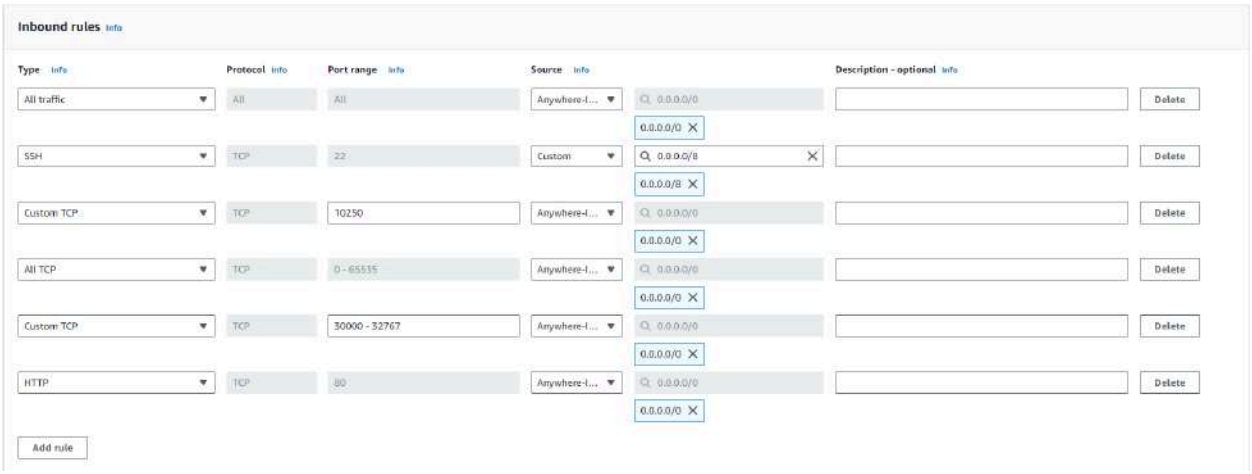
Create 2 security groups by navigating to Search → Security Groups.



Group 1: Master

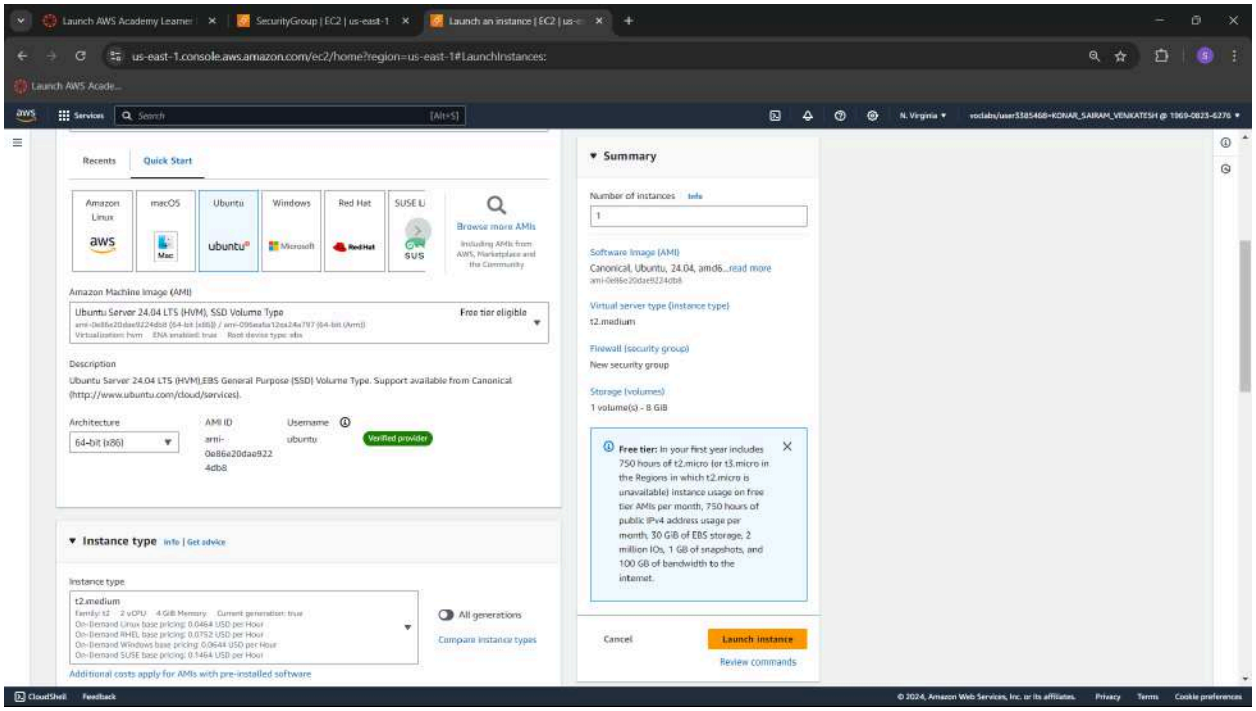


Group 2: Nodes

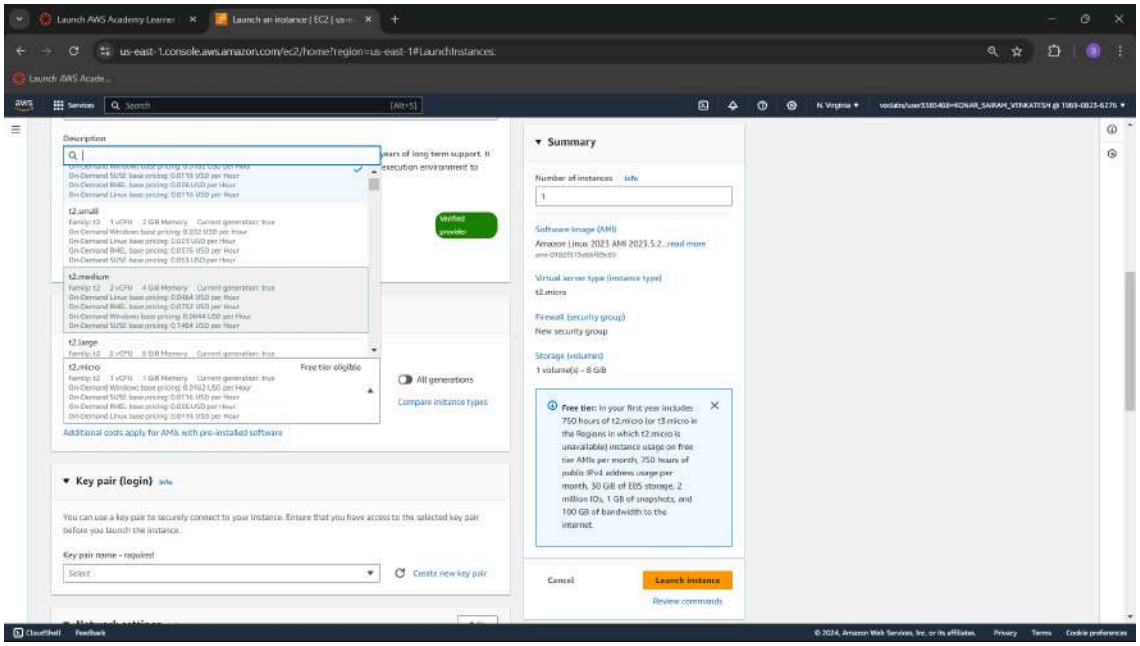


Step 1: Set Up EC2 Instances.

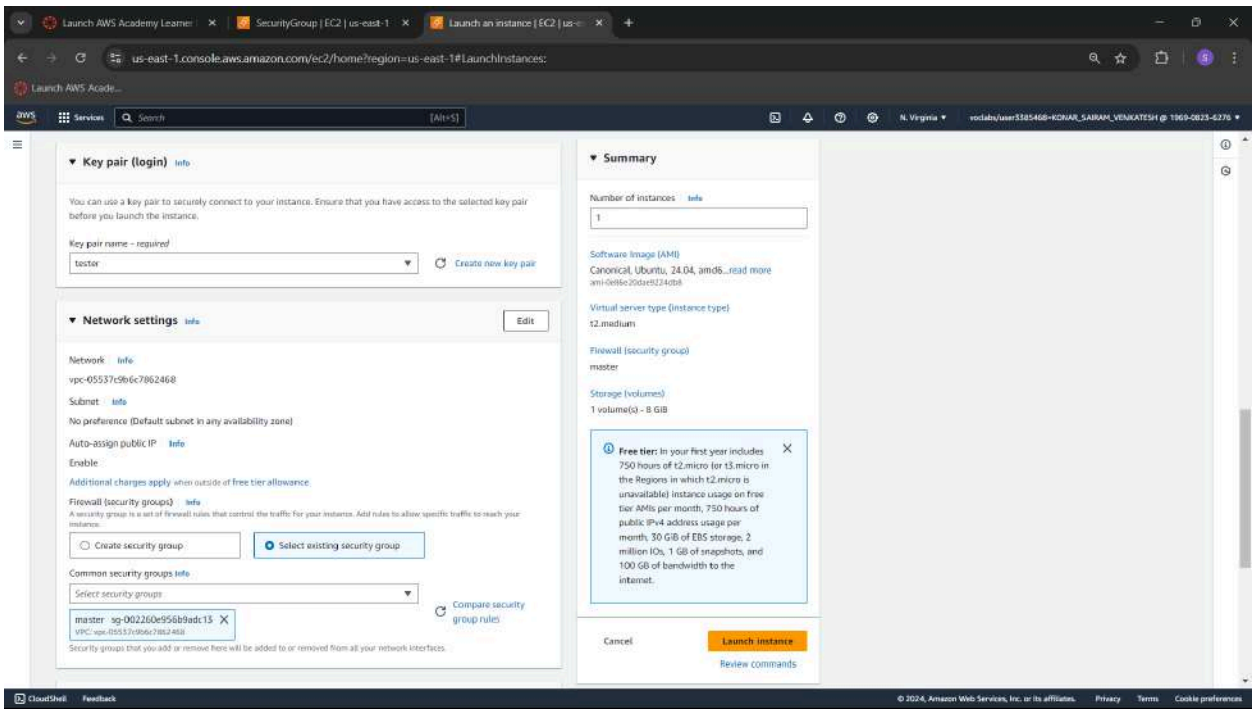
1) Set up 3 EC2 instances called master, node1, node2
   Select Amazon Linux as the OS image.



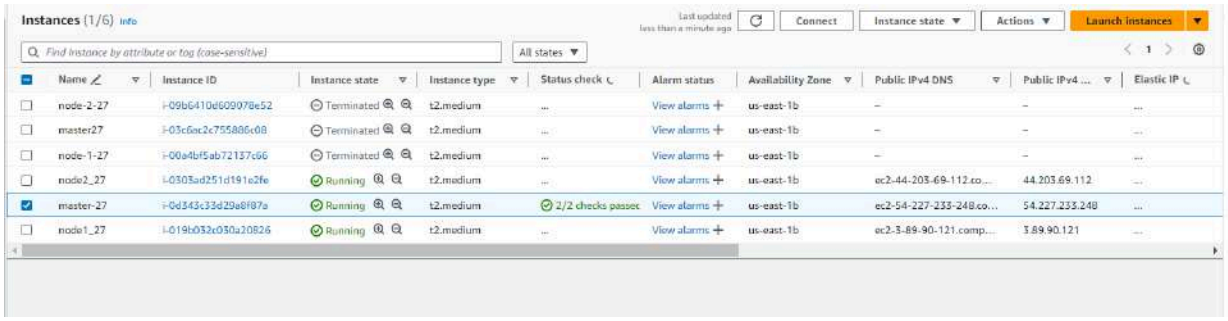
IMPORTANT: The default instance type and free one provided by AWS is t2.micro, which provides only 1CPU and 1 GiB of memory. For running Kubernetes, a minimum of 2 CPUs and 2GiB of RAM is required, hence change **t2.micro** to **t2.medium.**



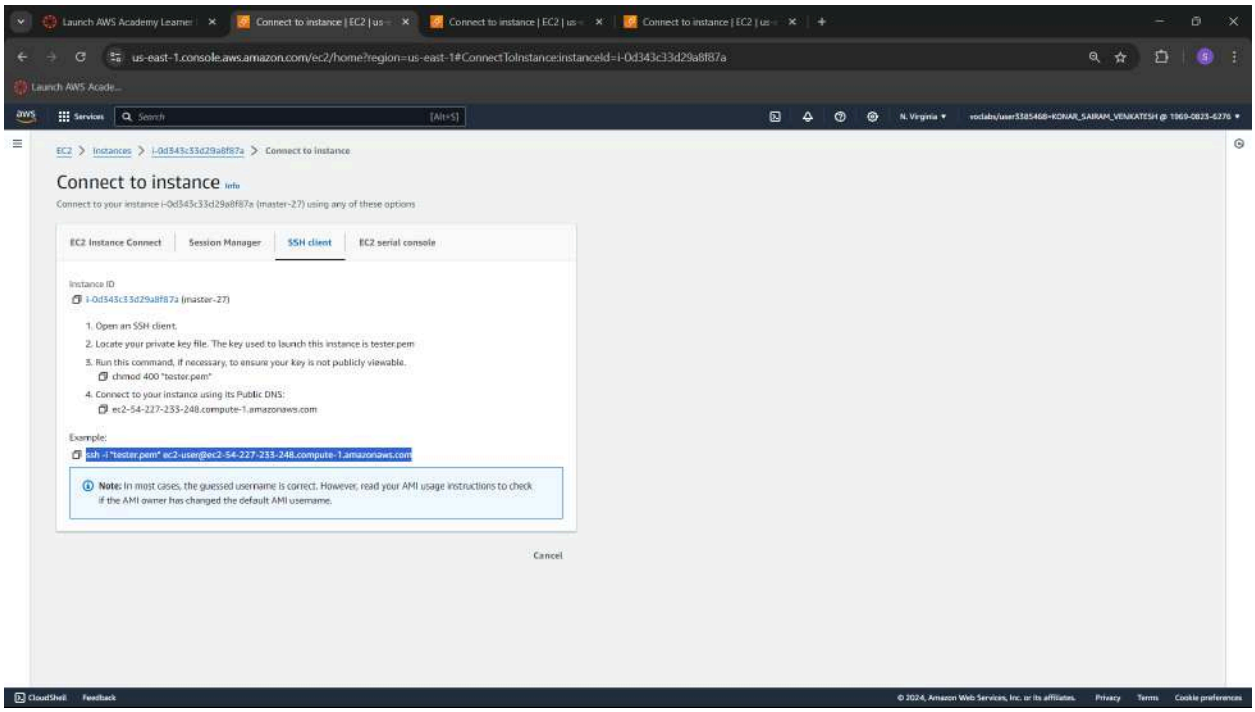
2) Create a key pair as you need the .pem file (private key file) on your system.
3) Click on ‘Select existing security group’ and select the master group for master instance, node group for both node instances.



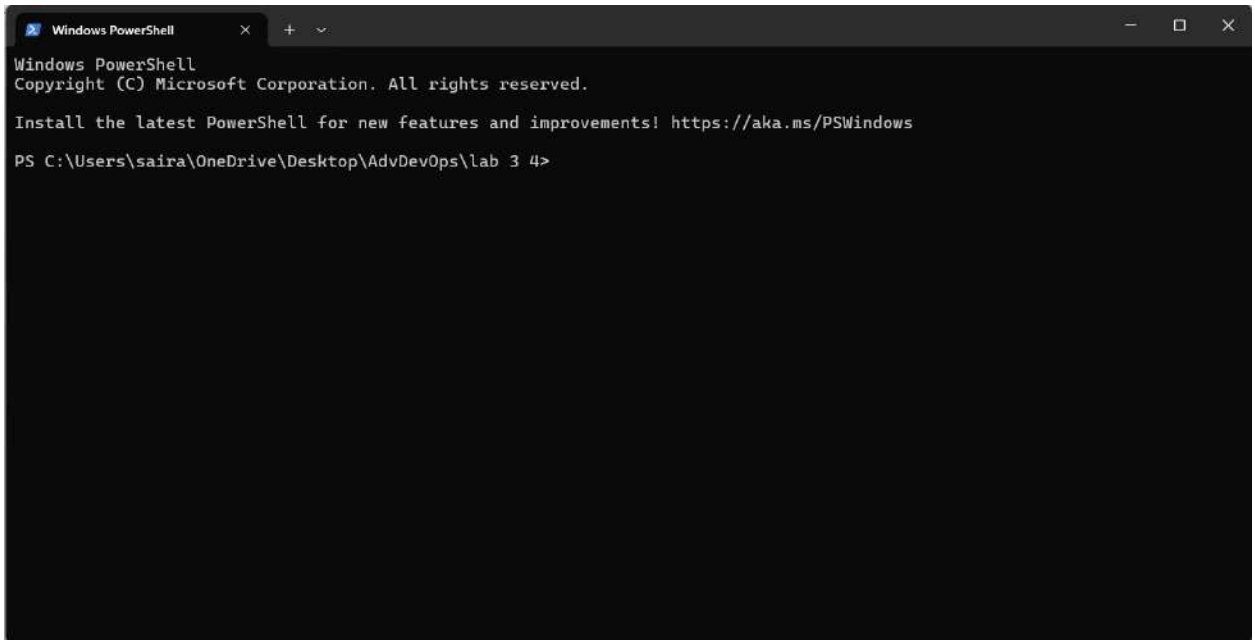
4) These are the instances that are created. Click on the Instance ID of master.



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



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

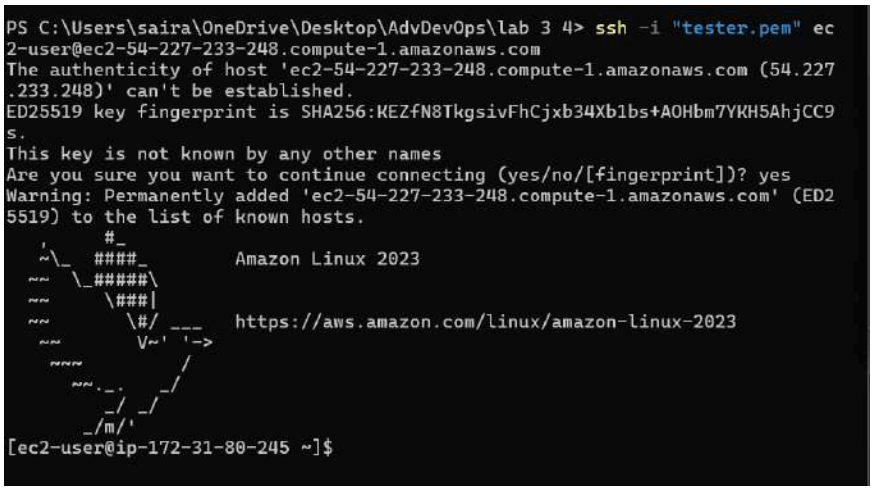


Paste the SSH command here and run it.

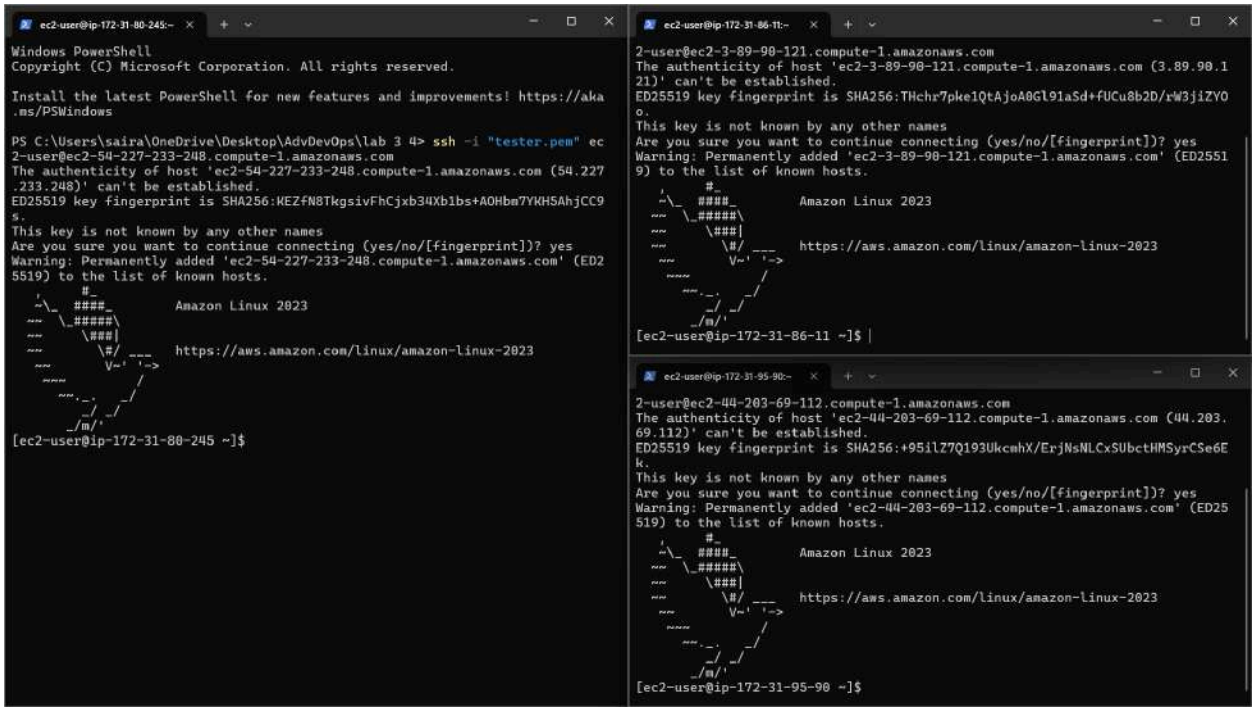
You might get the error of UNPROTECTED KEY FILE. This is because the .pem access Is with all users of teh system. Run the following commands to change the access to only the current user.

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

Now rerun the SSH command.



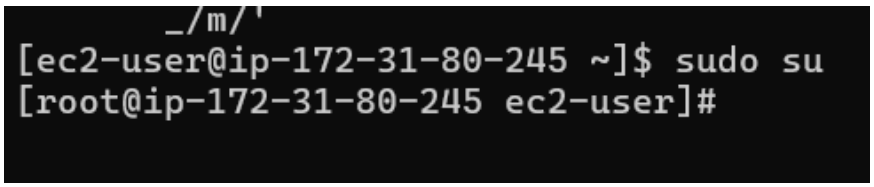
7) After doing the above steps, the terminal for all 3 nodes can be seen.



PERFORM THE FOLLOWING STEPS ON ALL 3 MACHINES

Step 2: Installation of Docker

1) Use command
   'sudo su'
   This allows you to act as the root user of the terminal



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

```
[root@ip-172-31-80-245 ec2-user]# yum install docker -y
Last metadata expiration check: 0:05:22 ago on Thu Sep 26 03:53:56 2024.
Dependencies resolved.
================================================================================
 Package                 Arch    Version                 Repository    Size
================================================================================
Installing:
 docker                  x86_64  25.0.6-1.amzn2023.0.2   amazonlinux   44 M
Installing dependencies:
 containerd              x86_64  1.7.20-1.amzn2023.0.1   amazonlinux   35 M
 iptables-libs           x86_64  1.8.8-3.amzn2023.0.2    amazonlinux  401 k
 iptables-nft            x86_64  1.8.8-3.amzn2023.0.2    amazonlinux  183 k
 libcgroup               x86_64  3.0-1.amzn2023.0.1      amazonlinux   75 k
 libnetfilter_conntrack  x86_64  1.0.8-2.amzn2023.0.2    amazonlinux   58 k
 libnfnetlink            x86_64  1.0.1-19.amzn2023.0.2   amazonlinux   30 k
 libnftnl                x86_64  1.2.2-2.amzn2023.0.2    amazonlinux   84 k
 pigz                    x86_64  2.5-1.amzn2023.0.3      amazonlinux   83 k
 runc                    x86_64  1.1.13-1.amzn2023.0.1   amazonlinux  3.2 M

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

```
Installed:
  containerd-1.7.20-1.amzn2023.0.1.x86_64
  docker-25.0.6-1.amzn2023.0.2.x86_64
  iptables-libs-1.8.8-3.amzn2023.0.2.x86_64
  iptables-nft-1.8.8-3.amzn2023.0.2.x86_64
  libcgroup-3.0-1.amzn2023.0.1.x86_64
  libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64
  libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64
  libnftnl-1.2.2-2.amzn2023.0.2.x86_64
  pigz-2.5-1.amzn2023.0.3.x86_64
  runc-1.1.13-1.amzn2023.0.1.x86_64

Complete!
[root@ip-172-31-80-245 ec2-user]#
```

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

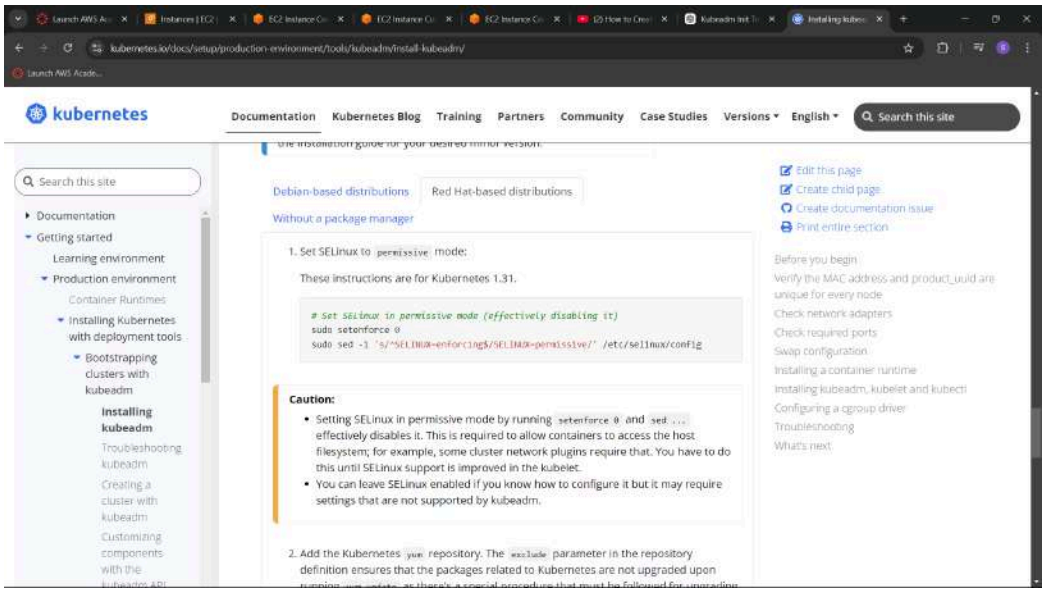
- cd /etc/docker
- cat <<EOF | sudo tee /etc/docker/daemon.json
  {
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
  "max-size": "100m"
  },
  "storage-driver": "overlay2"
  }
  EOF
- sudo systemctl enable docker

- sudo systemctl daemon-reload
- sudo systemctl restart docker

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

Step 3: Installing Kubernetes

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



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

sudo setenforce 0
→ sets SELinux to permissive mode

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

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

Run the following commands:

- sudo setenforce 0
- sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
- cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
  [kubernetes]
  name=Kubernetes
  baseurl=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/
  enabled=1
  gpgcheck=1
  gpgkey=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/repodata/repomd.xml.key
  exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
  EOF

This comamnd is a repository script to create a kubernetes repository

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

- yum repolist
  This command shows the repositories created on the machine.

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

Next step is to install kubelet, kubeadm, kubectl

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

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

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

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

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

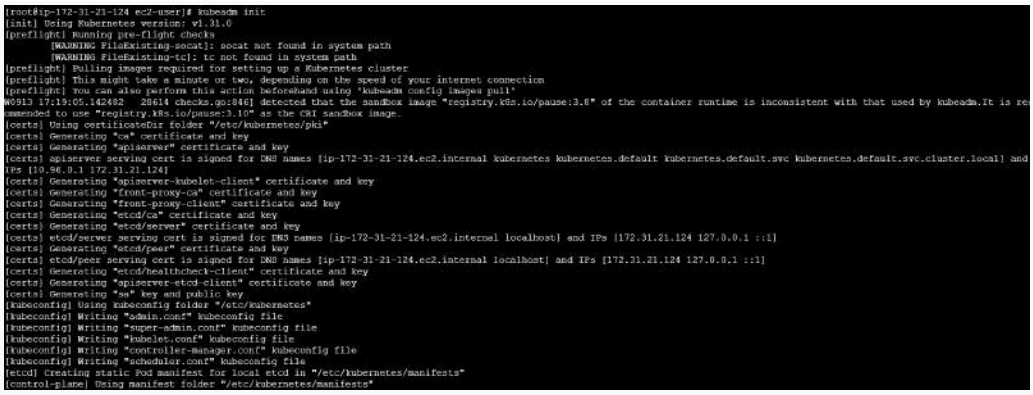
Now, we need to enable the kubelet service. Run the command

- sudo systemctl enable --now kubelet

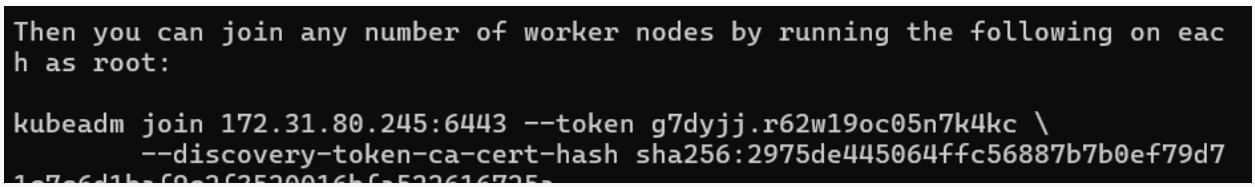
```
[root@ip-172-31-80-245 docker]# sudo systemctl enable --now kubelet
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service
→ /usr/lib/systemd/system/kubelet.service.
[root@ip-172-31-80-245 docker]#
```

PERFORM THE FOLLOWING ON ONLY THE MASTER MACHINE

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



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



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

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

   Run these commands.

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

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

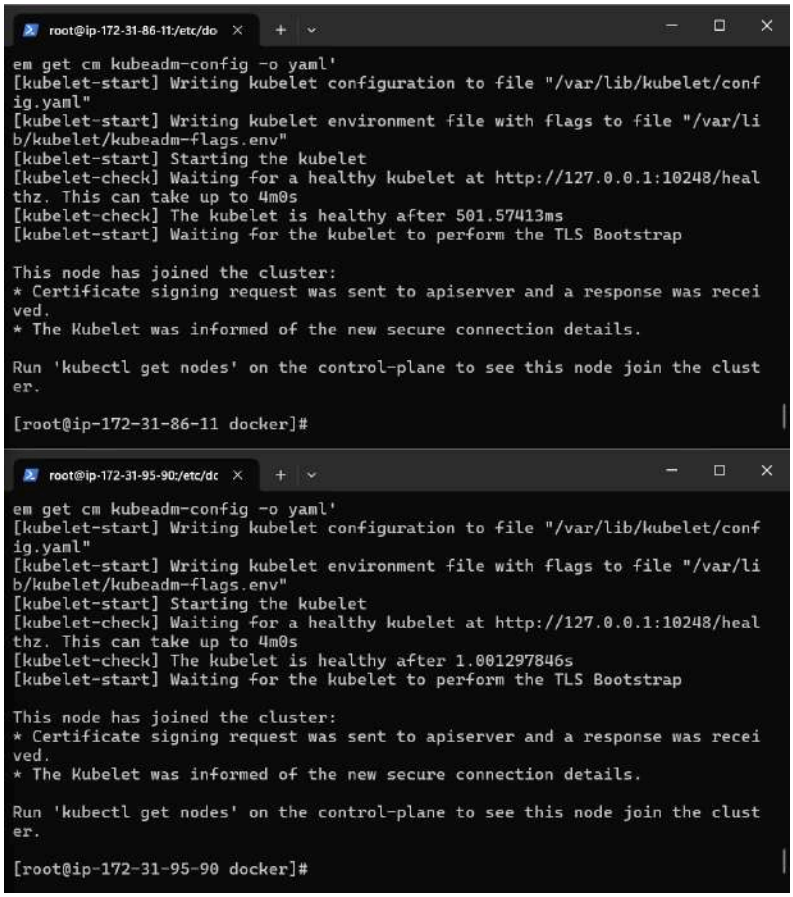
3) To check whether nodes are connected, run the command
   - kubectl get nodes

This output shows only master is connected right now.

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

PERFORM THE FOLLOWING ONLY ON THE NODE MACHINES

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



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

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

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

Use the following command only on the master machine::

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

Now run ‘kubectl get nodes’ again.

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

Conclusion:

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

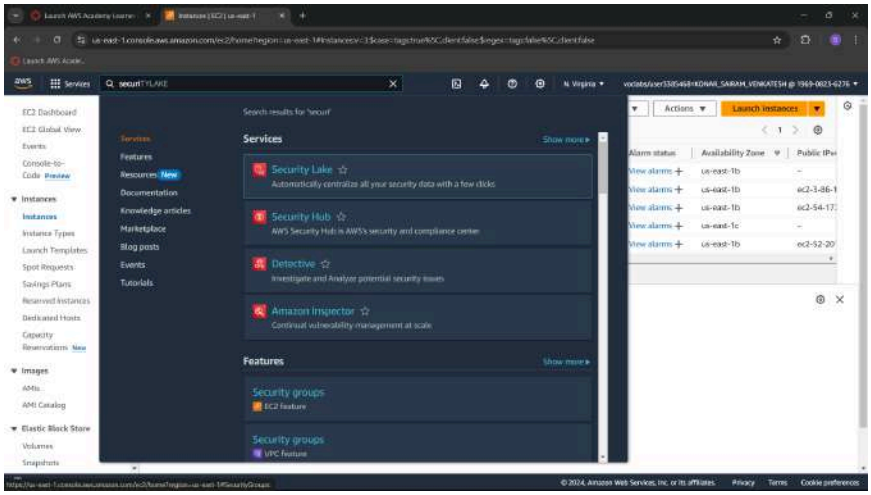
Sairam Konar D15C 27 2024-25

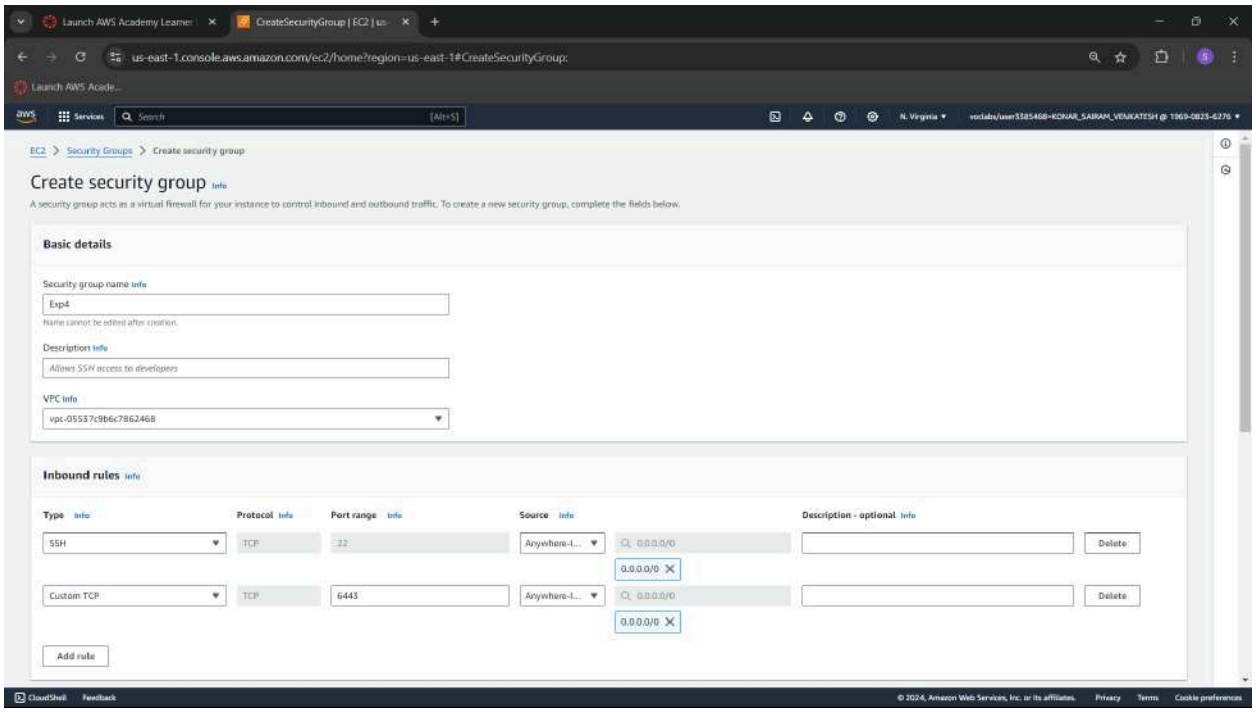
Experiment 4

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

Prerequisites:

Create a security group by navigating to Search → Security Groups.

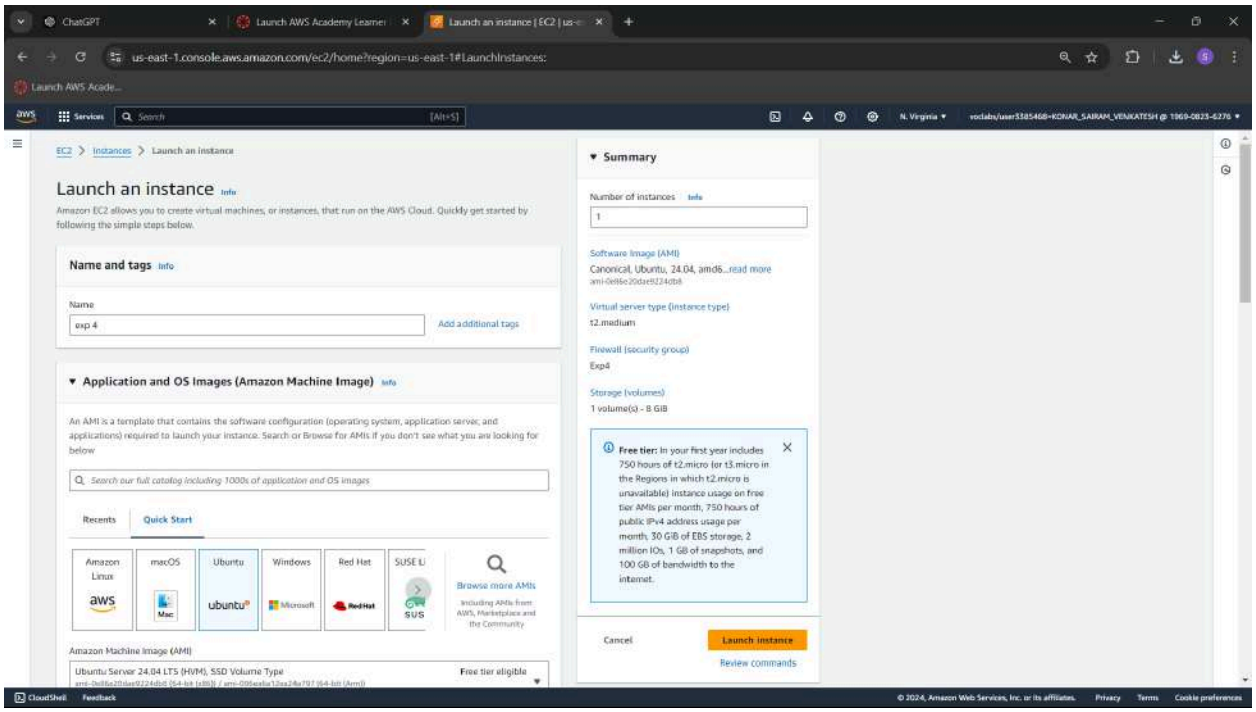




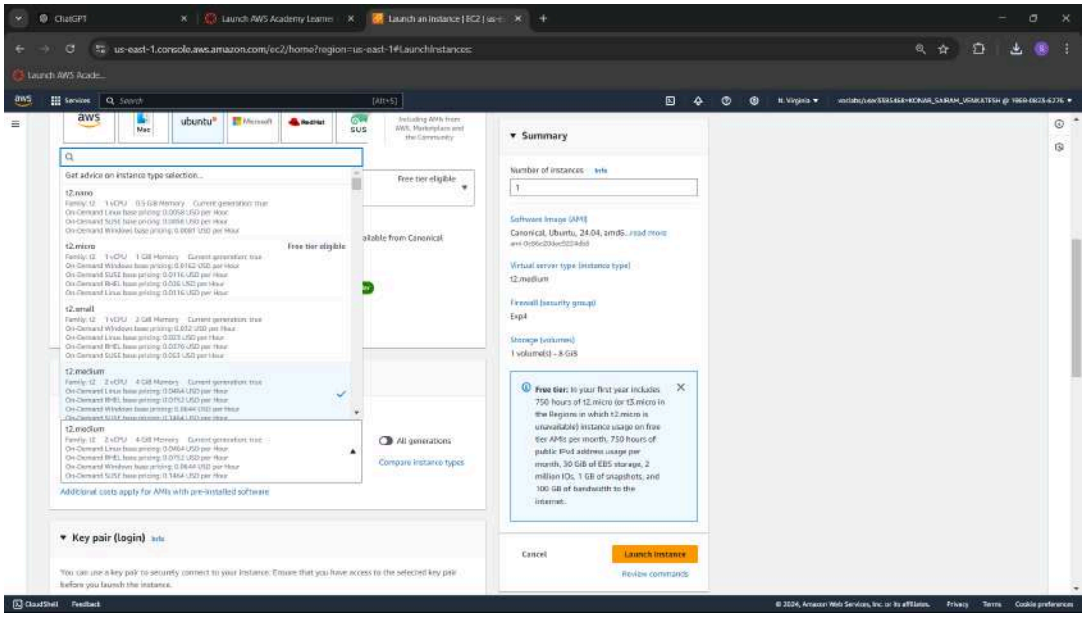
Sairam Konar D15C 27 2024-25

Step 1: Creation of instances.

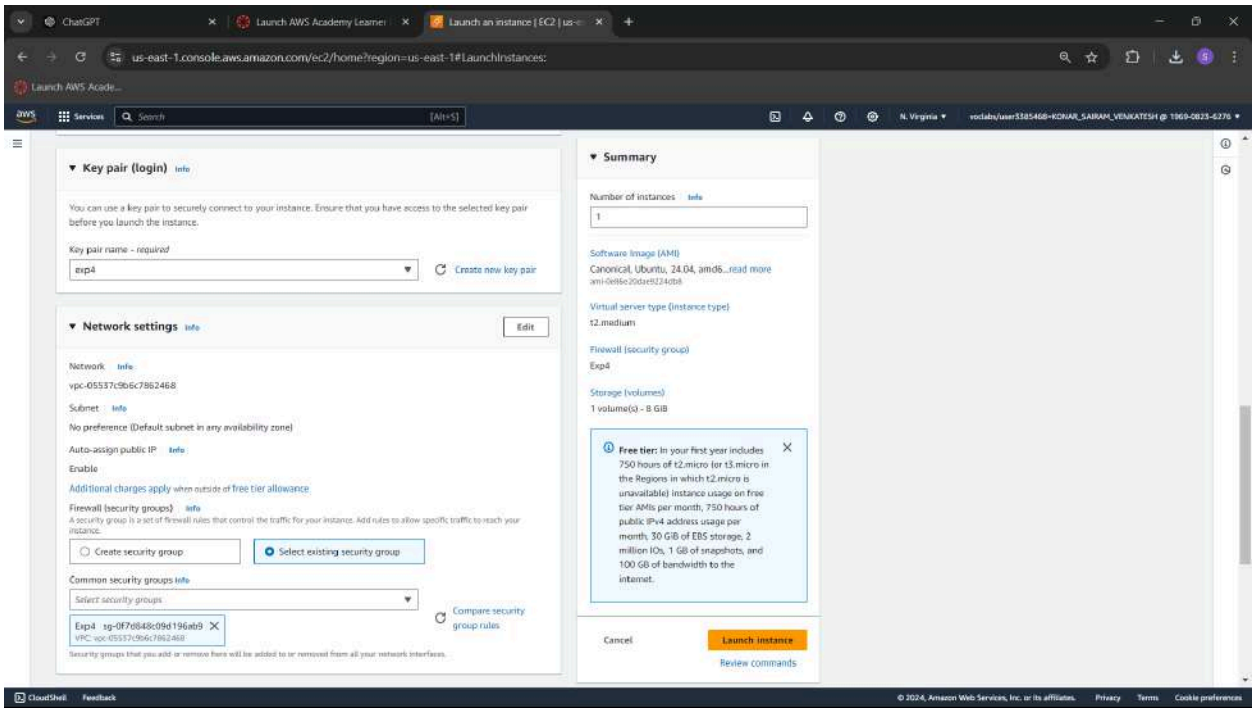
1) Go to the AWS Dashboard. In Services, search for EC2. Open it. Click on Launch instance,



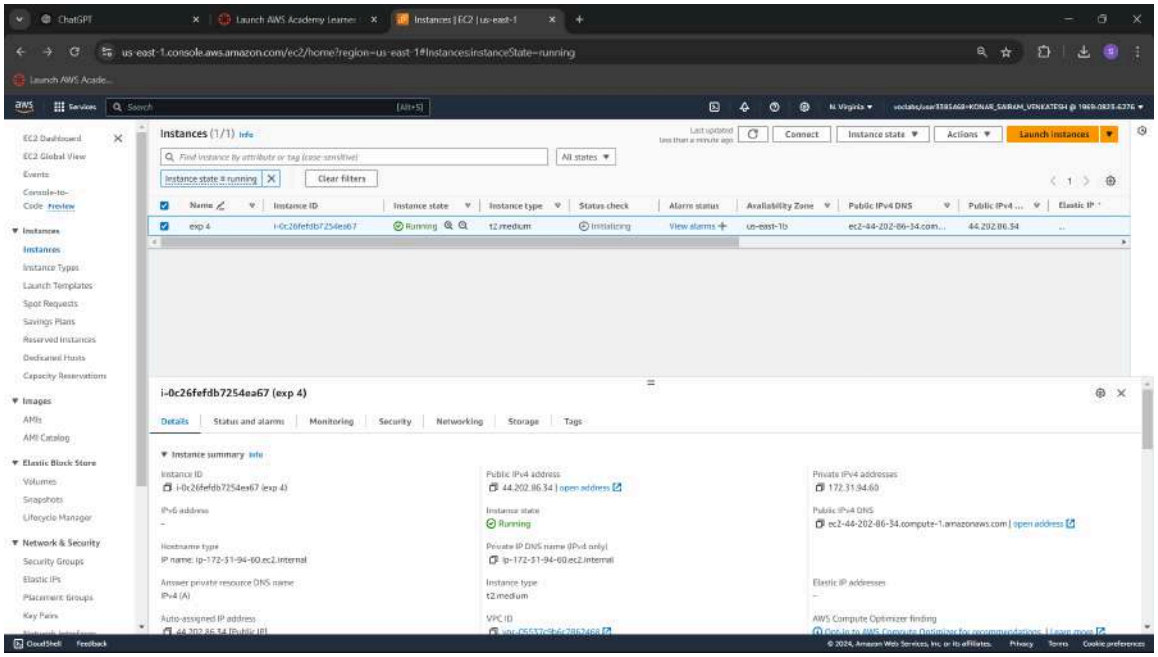
2) Select ubuntu 22.04 as you OS image.
IMPORTANT: The default instance type and free one provided by AWS is t2.micro, which provides only 1CPU and 1 GiB of memory. For running Kubernetes, a minimum of 2 CPUs and 2GiB of RAM is required, hence change **t2.micro** to **t2.medium.**



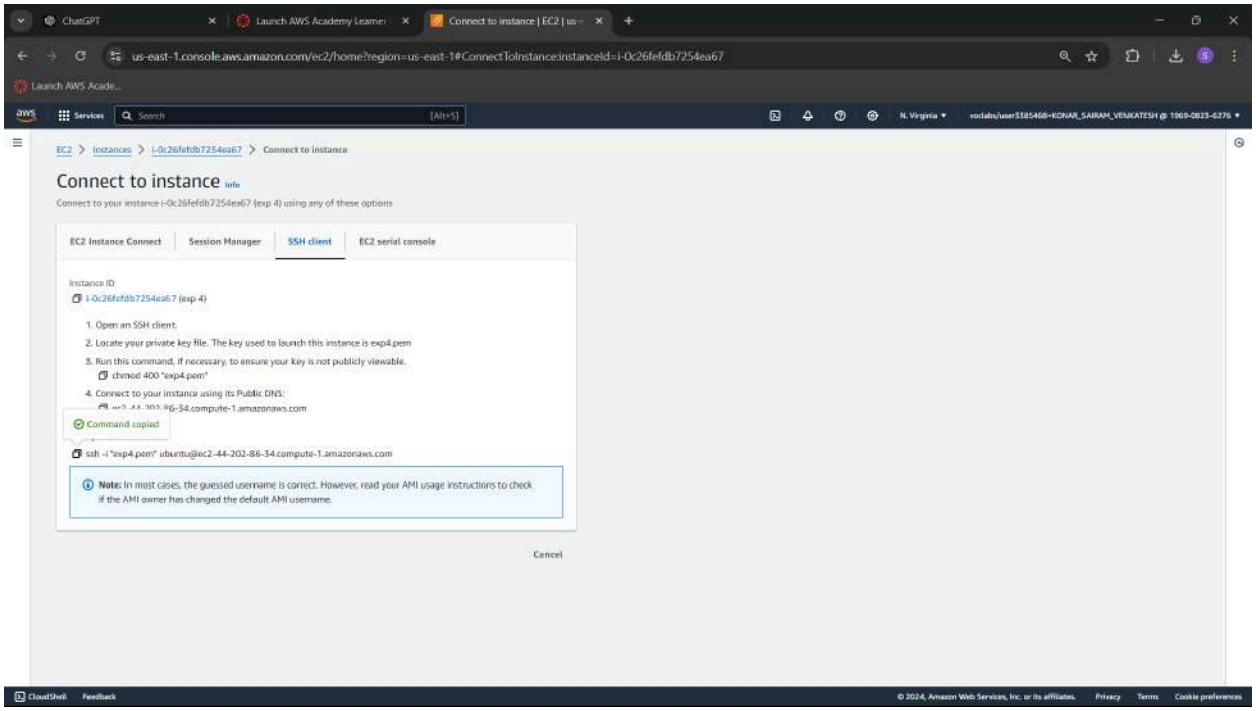
3) Create a key pair as you need the .pem file (private key file) on your system.
4) Click on ‘Select existing security group’ and select the exp4 group for the instance.



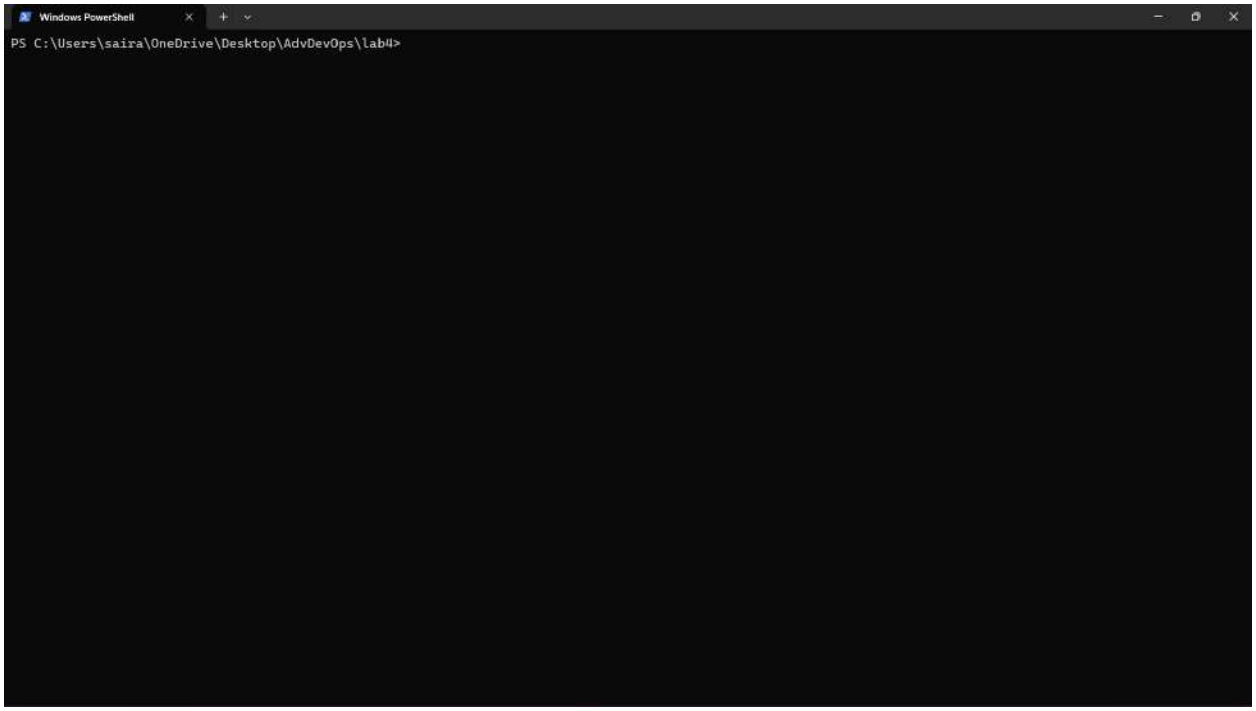
5) The instance is created. Click on the instance id, then click on connect.



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



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

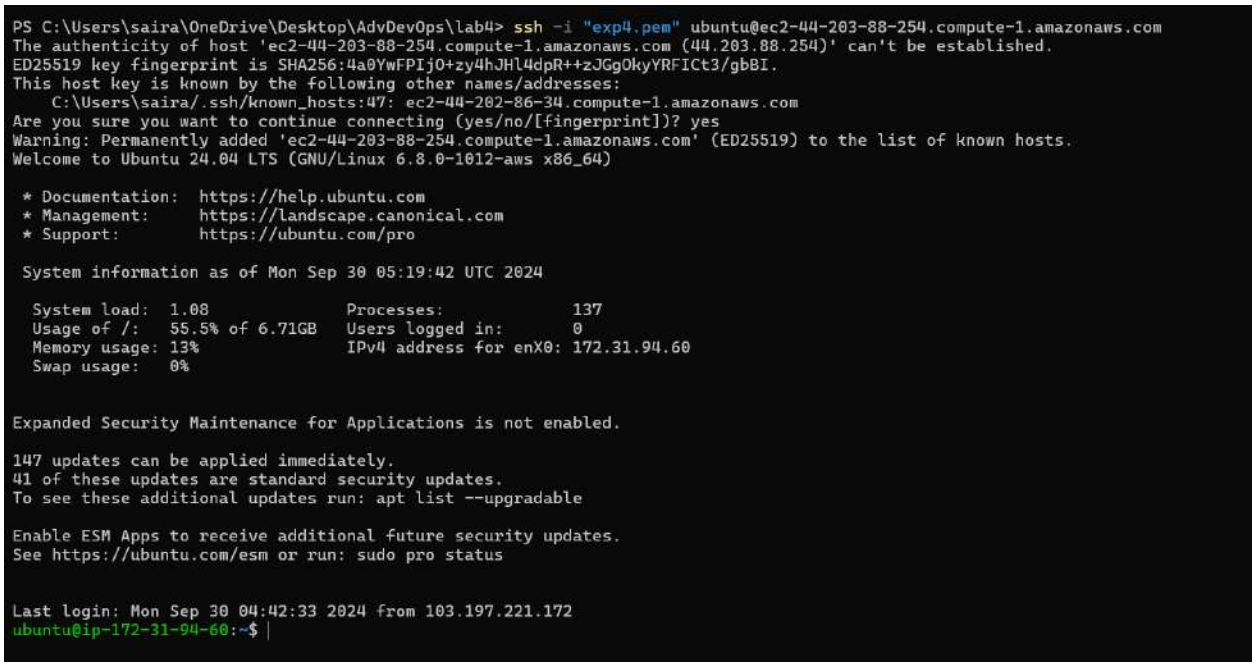


Paste the SSH command here and run it.

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

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

Now rerun the SSH command.



Step 2: Setup Docker

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

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

```
ubuntu@ip-172-31-94-60:~$ 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
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
ubuntu@ip-172-31-94-60:~$ 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
W: The key(s) in the keyring /etc/apt/trusted.gpg.d/docker.gpg are ignored as the file has an unsupported filetype.
ubuntu@ip-172-31-94-60:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/trusted.gpg.d/dock
er.gpg > /dev/null
ubuntu@ip-172-31-94-60:~$
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
Repository: 'deb [arch=amd64] https://download.docker.com/linux/ubuntu noble stable'
Description:
Archive for codename: noble components: stable
More info: https://download.docker.com/linux/ubuntu
Adding repository.
Press [ENTER] to continue or Ctrl-c to cancel.
Adding deb entry to /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-noble.list
Adding disabled deb-src entry to /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-noble.list
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
```

```
Get:45 http://security.ubuntu.com/ubuntu noble-security/restricted Translation-en [68.1 kB]
Get:46 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 Components [212 B]
Get:47 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 c-n-f Metadata [428 B]
Get:48 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Packages [10.9 kB]
Get:49 http://security.ubuntu.com/ubuntu noble-security/multiverse Translation-en [2808 B]
Get:50 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 c-n-f Metadata [116 B]
Get:51 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [208 B]
Get:52 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 c-n-f Metadata [344 B]
Fetched 29.1 MB in 4s (7836 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/dock
er.gpg are ignored as the file has an unsupported filetype.
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt
/trusted.gpg), see the DEPRECATION section in apt-key(8) for details.
ubuntu@ip-172-31-94-60:~$
```

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

```
ubuntu@ip-172-31-94-60:~$ 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 https://download.docker.com/linux/ubuntu noble InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports 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/dock
er.gpg are ignored as the file has an unsupported filetype.
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt
```

```
Setting up slirp4netns (1.2.1-1build2) ...
Setting up docker-ce (5:27.3.1-1~ubuntu.24.04~noble) ...
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /usr/lib/systemd/system/docker.socket.
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for libc-bin (2.39-0ubuntu8.2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-94-60:~$
```

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

```
ubuntu@ip-172-31-94-60:~$ 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-172-31-94-60:~$
```

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

```
ubuntu@ip-172-31-94-60:~$ sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
```

Step 3: Set up Kubernetes

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

```
ubuntu@ip-172-31-94-60:~$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /et
c/apt/keyrings/kubernetes-apt-keyring.gpg
ubuntu@ip-172-31-94-60:~$ echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/s
table:/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-172-31-94-60:~$
```

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

```
ubuntu@ip-172-31-94-60:~$ 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
```

```
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-172-31-94-60:~$
```

Step 4: Initialise the kubecluster

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

```
ubuntu@ip-172-31-94-60:~$ sudo systemctl enable --now kubelet
sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
W0930 04:29:33.758696    4337 checks.go:1080] [preflight] WARNING: Couldn't create the interface used for talking to the
 container runtime: failed to create new CRI runtime service: validate service connection: validate CRI v1 runtime API f
or endpoint "unix:///var/run/containerd/containerd.sock": rpc error: code = Unimplemented desc = unknown service runtime
.v1.RuntimeService
        [WARNING FileExisting-socat]: socat not found in system path
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action beforehand using 'kubeadm config images pull'
error execution phase preflight: [preflight] Some fatal errors occurred:
failed to create new CRI runtime service: validate service connection: validate CRI v1 runtime API for endpoint "unix://
/var/run/containerd/containerd.sock": rpc error: code = Unimplemented desc = unknown service runtime.v1.RuntimeService[p
reflight] If you know what you are doing, you can make a check non-fatal with `--ignore-preflight-errors=...`
To see the stack trace of this error execute with --v=5 or higher
```

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

- sudo apt-get install -y containerd

```
ubuntu@ip-172-31-94-60:~$ 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.
```

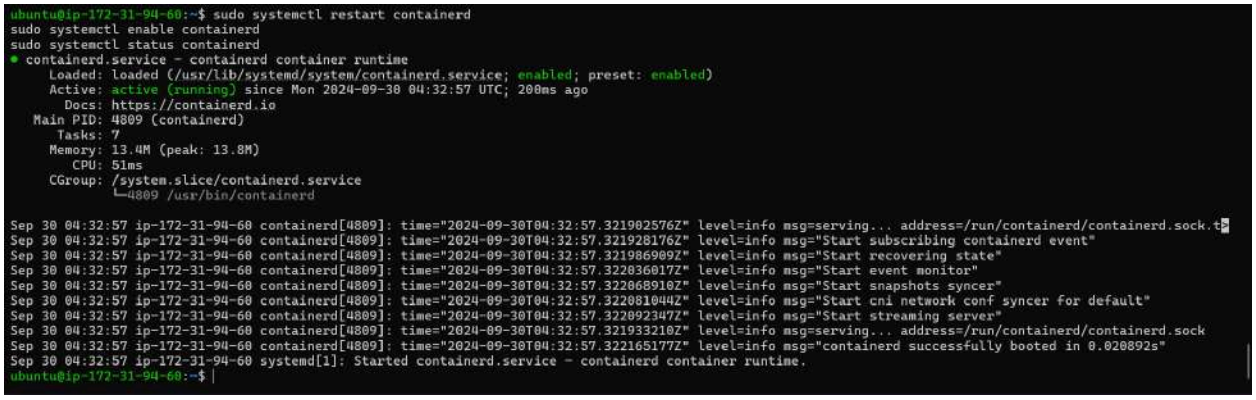
```
Running kernel seems to be up-to-date.

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-172-31-94-60:~$
```

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

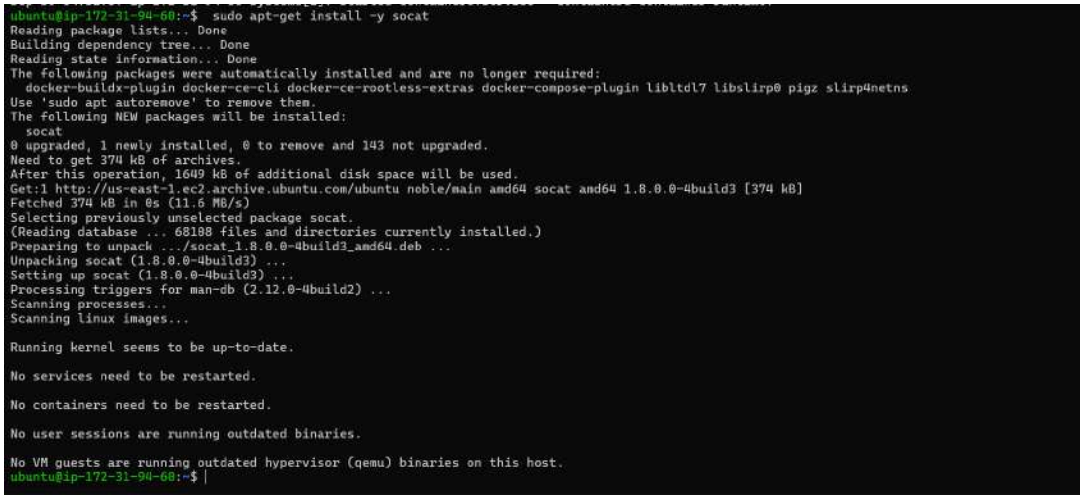
```
ubuntu@ip-172-31-94-60:~$ sudo mkdir -p /etc/containerd
sudo containerd config default | sudo tee /etc/containerd/config.toml
disabled_plugins = []
imports = []
oom_score = 0
plugin_dir = ""
required_plugins = []
root = "/var/lib/containerd"
state = "/run/containerd"
temp = ""
version = 2

[cgroup]
  path = ""

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

```
[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-172-31-94-60:~$
```

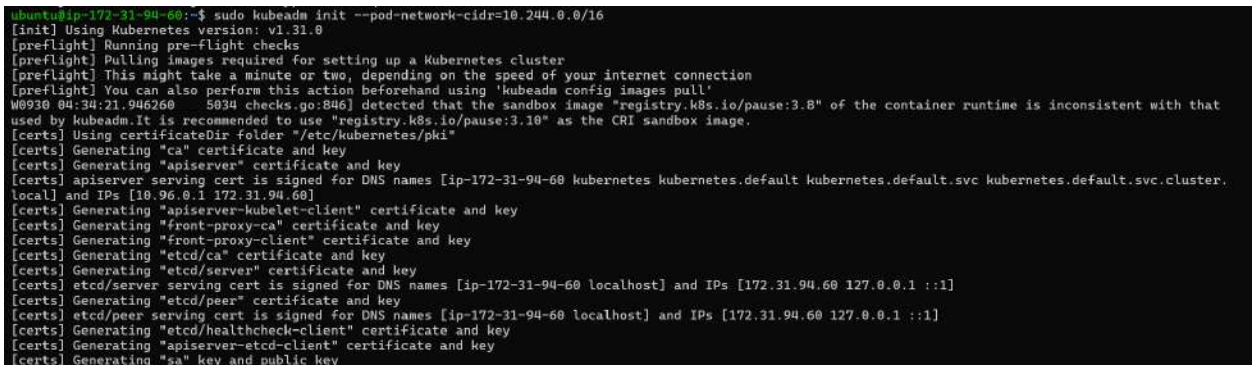
- sudo systemctl restart containerd
- sudo systemctl enable containerd
- sudo systemctl status containerd



- sudo apt-get install -y socat



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



```
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 172.31.94.60:6443 --token bhzgoy.c4u84a2hc4bf7u19 \
        --discovery-token-ca-cert-hash sha256:9d14667afdeed67b83ac64ae48deb6b315e62070260dd78c7d18b1db4c94b195
ubuntu@ip-172-31-94-60:~$
```

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

```
ubuntu@ip-172-31-94-60:~$ 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-172-31-94-60:~$
```

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

```
ubuntu@ip-172-31-94-60:~$ kubectl apply -f https://k8s.io/examples/application/deployment.yaml
deployment.apps/nginx-deployment created
```

Step 5: Connect Ngix server to pod.

- kubectl get pods

```
ubuntu@ip-172-31-94-60:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-4cbnz    0/1     Pending   0          28s
nginx-deployment-d556bf558-zk989    0/1     Pending   0          28s
ubuntu@ip-172-31-94-60:~$
```

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

- kubectl port-forward $POD_NAME 8080:80

```
ubuntu@ip-172-31-94-60:~$ POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
ubuntu@ip-172-31-94-60:~$ kubectl port-forward $POD_NAME 8080:80
error: unable to forward port because pod is not running. Current status=Pending
ubuntu@ip-172-31-94-60:~$
```

Here, another error is generated as the pods are not ready. To make them ready, we need to untaint them. Run the following commands.

- kubectl taint nodes --all node-role.kubernetes.io/control-plane:NoSchedule-

- kubectl get nodes

```
ubuntu@ip-172-31-94-60:~$ kubectl taint nodes --all node-role.kubernetes.io/control-plane:NoSchedule-
node/ip-172-31-94-60 untainted
ubuntu@ip-172-31-94-60:~$ kubectl get nodes
NAME              STATUS   ROLES           AGE     VERSION
ip-172-31-94-60   Ready    control-plane   5m36s   v1.31.1
ubuntu@ip-172-31-94-60:~$
```

- kubectl get pods

```
ubuntu@ip-172-31-94-60:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-4cbnz    1/1     Running   0          4m36s
nginx-deployment-d556bf558-zk989    1/1     Running   0          4m36s
ubuntu@ip-172-31-94-60:~$
```

Now rerun the following command.

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

```
ubuntu@ip-172-31-94-60:~$ 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
```

Open another terminal and connect ssh to that terminal as well.

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

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

PS C:\Users\saira> cd '.\OneDrive\Desktop\AdvDevOps\lab4'
PS C:\Users\saira\OneDrive\Desktop\AdvDevOps\lab4> ssh -i "exp4.pem" ubuntu@ec2-44-202-86-34.compute-1.amazonaws.com
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

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

 System information as of Mon Sep 30 04:42:33 UTC 2024

  System load:  0.12              Processes:             153
  Usage of /:   55.3% of 6.71GB   Users logged in:       1
  Memory usage: 19%               IPv4 address for enX0: 172.31.94.60
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

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

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


Last login: Mon Sep 30 04:22:55 2024 from 103.197.221.172
ubuntu@ip-172-31-94-60:~$
```

Run the command curl --head http://127.0.0.1:8080

```
ubuntu@ip-172-31-94-60:~$  curl --head http://127.0.0.1:8080
HTTP/1.1 200 OK
Server: nginx/1.14.2
Date: Mon, 30 Sep 2024 04:43:48 GMT
Content-Type: text/html
Content-Length: 612
Last-Modified: Tue, 04 Dec 2018 14:44:49 GMT
Connection: keep-alive
ETag: "5c0692e1-264"
Accept-Ranges: bytes
```

If it shows status code of 200 OK, it means that the deployment of nginx server was successful.

Go back to the previous terminal. After running curl, the output for port forward looks like this

```
ubuntu@ip-172-31-94-60:~$ POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")

kubectl port-forward $POD_NAME 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
Handling connection for 8080
```

Conclusion:

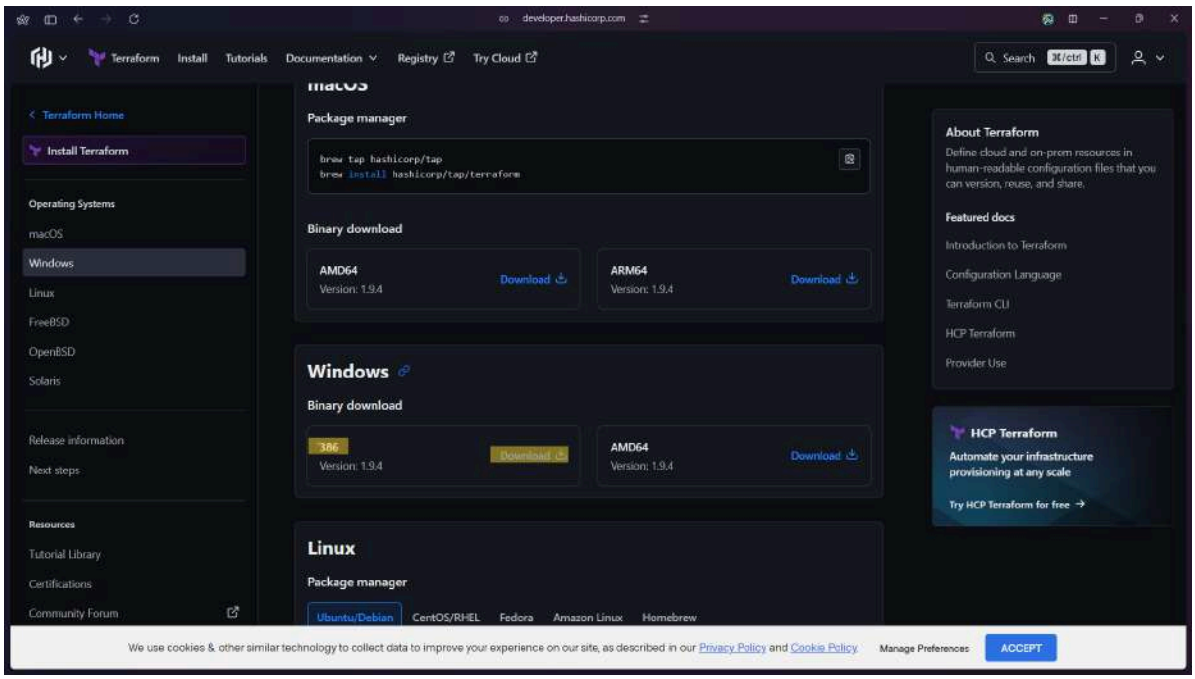
In this experiment, we have successfully installed kubectl and deployed our first Kubernetes Application. After setting up the instance, we connect it to our local terminal using SSH. The application we deployed was an nginx server. To deploy this, we had to install flannel, a common networking plugin. There were a few errors. One being while initializing the cluster, which was solved by installing the missing dependencies. The second one was when we checked the pods that were created. These pods were in the not ready state as they had some issues. To solve this, we had to untaint them. Once all the errors were resolved, we run the port command to initiate the connection. When we run the curl command, we get a OK status. Hence the deployment was complete.

Sairam Konar D15C 27 2024-25

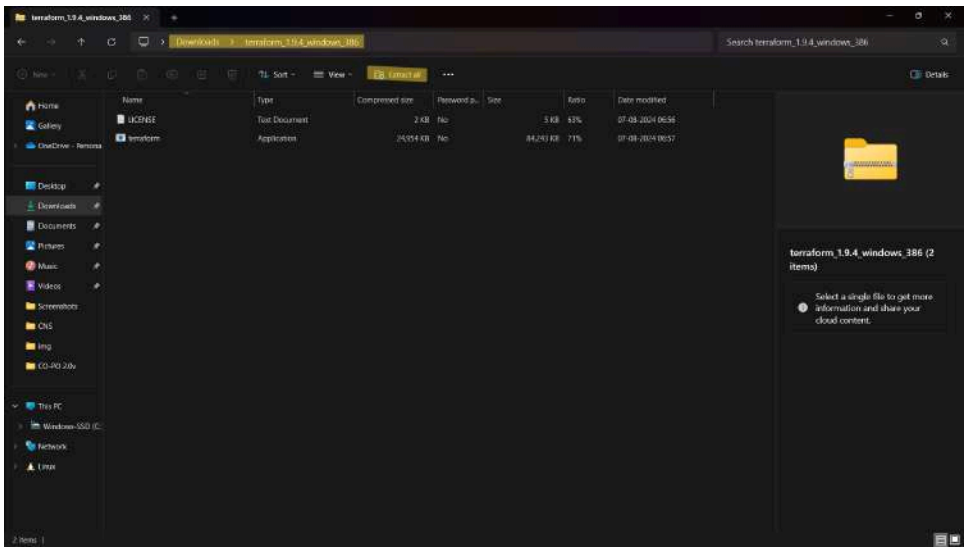
Experiment 5

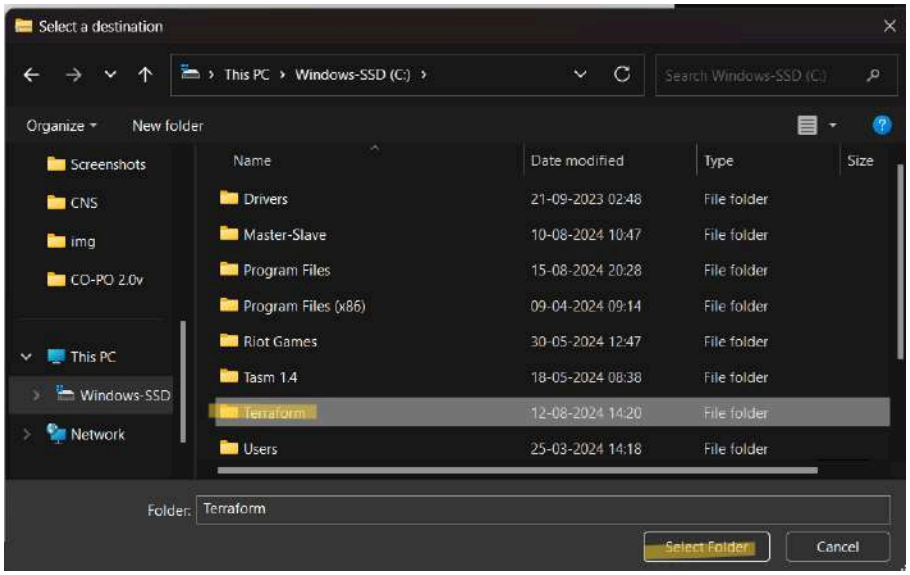
Aim: To understand terraform lifecycle, core concepts/terminologies and install it on a Linux Machine and Windows.

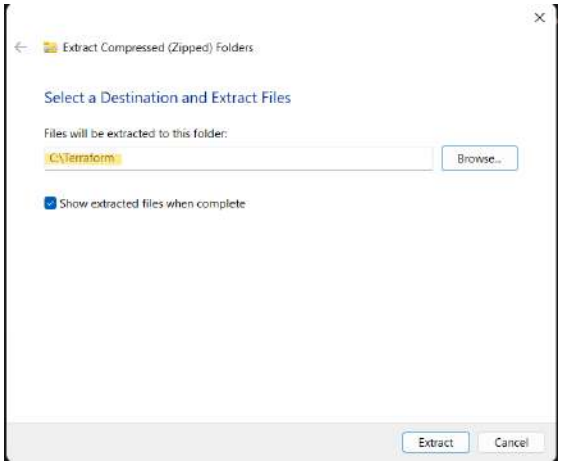
Step 1: Visit https://developer.hashicorp.com/terraform/install. Scroll down to the operating system on which you want to install Terraform. For Windows, select 386 option.

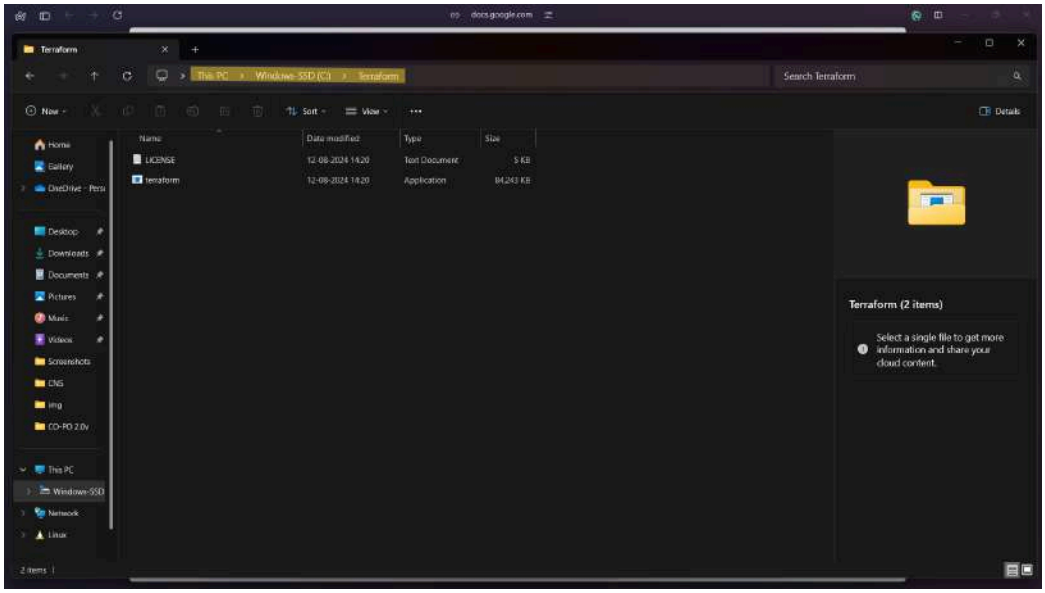


Step 2: A zip file gets downloaded to your system. Create a folder on your C-drive named as Terraform. Go to the zip you downloaded, click on extract all, and select the C-drive file we just created. This fill extract the Folder in C:\Terraform.

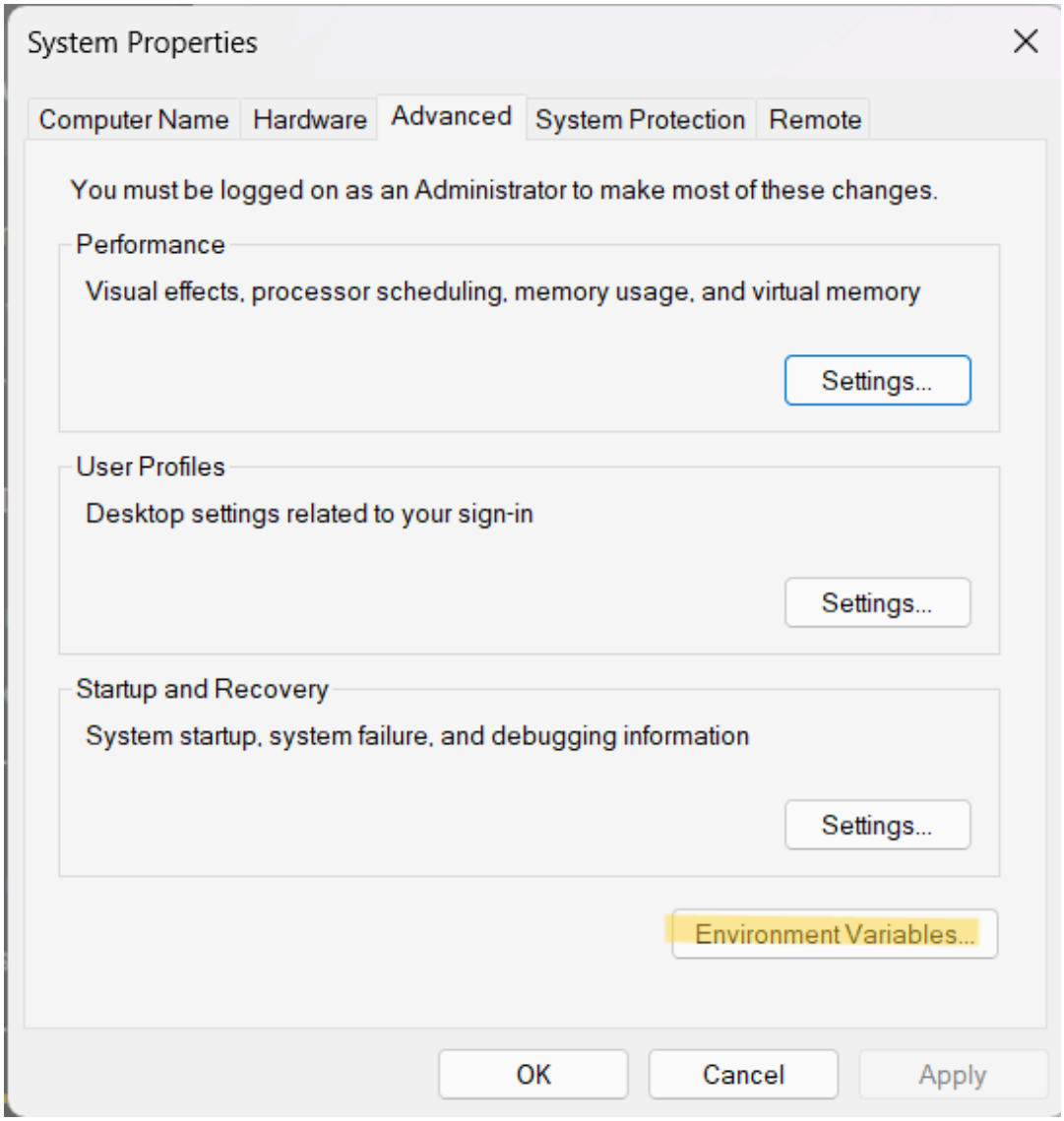



Select a destination
This PC › Windows-SSD (C:) ›
Search Windows-SSD (C:)
Organize
New folder
Screenshots
CNS
img
CO-PO 2.0v
This PC
Windows-SSD
Network
Name
Date modified
Type
Size
Drivers 21-09-2023 02:48 File folder
Master-Slave 10-08-2024 10:47 File folder
Program Files 15-08-2024 20:28 File folder
Program Files (x86) 09-04-2024 09:14 File folder
Riot Games 30-05-2024 12:47 File folder
Tasm 1.4 18-05-2024 08:38 File folder
Terraform 12-08-2024 14:20 File folder
Users 25-03-2024 14:18 File folder
Folder: Terraform
Select Folder
Cancel

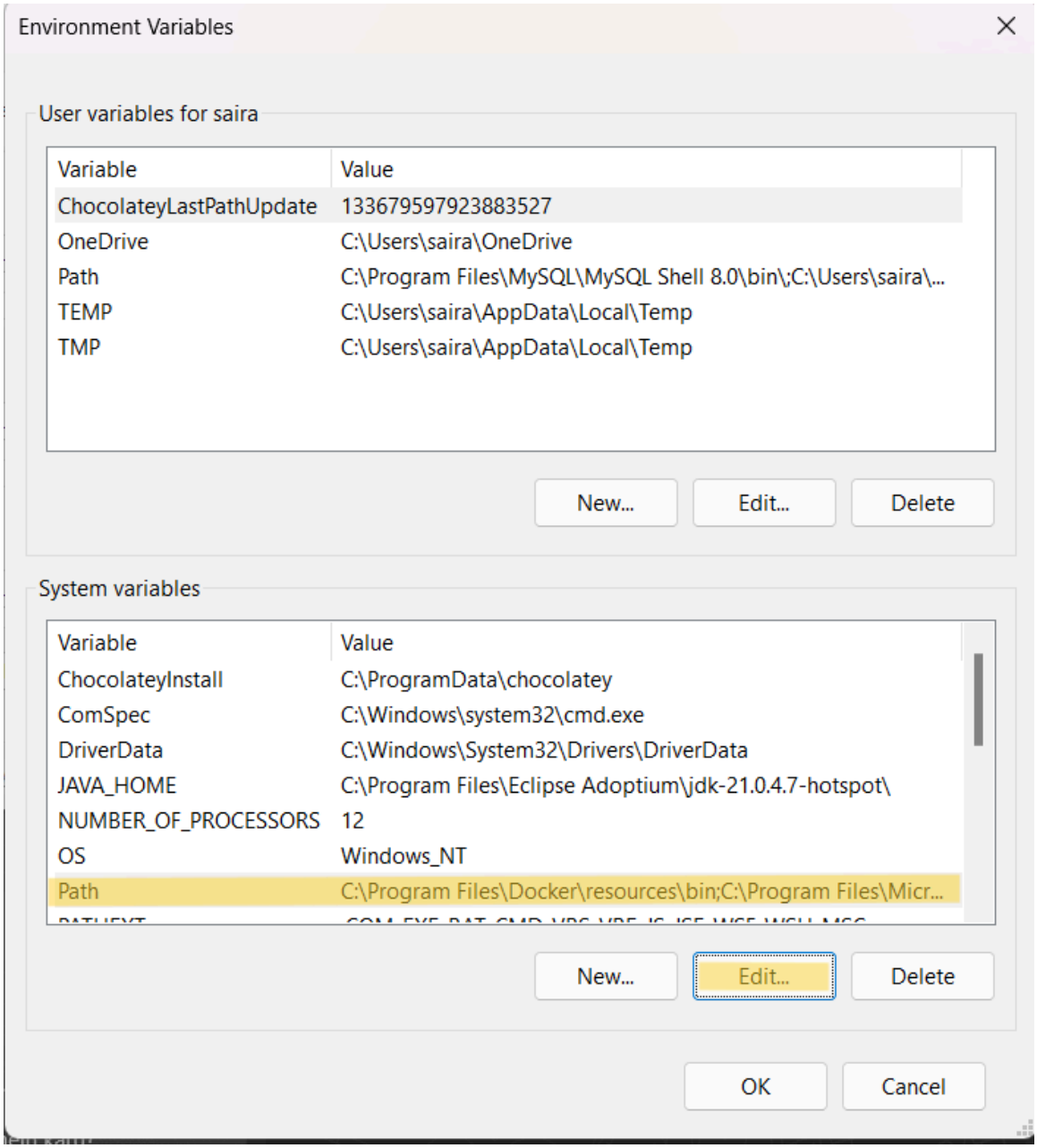

Extract Compressed (Zipped) Folders
Select a Destination and Extract Files
Files will be extracted to this folder:
C:\Terraform
Browse...
Show extracted files when complete
Extract
Cancel


Terraform
Search Terraform
Name
Date modified
Type
Size
LICENSE
terraform
Terraform (2 items)
Select a single file to get more information and share your cloud content.

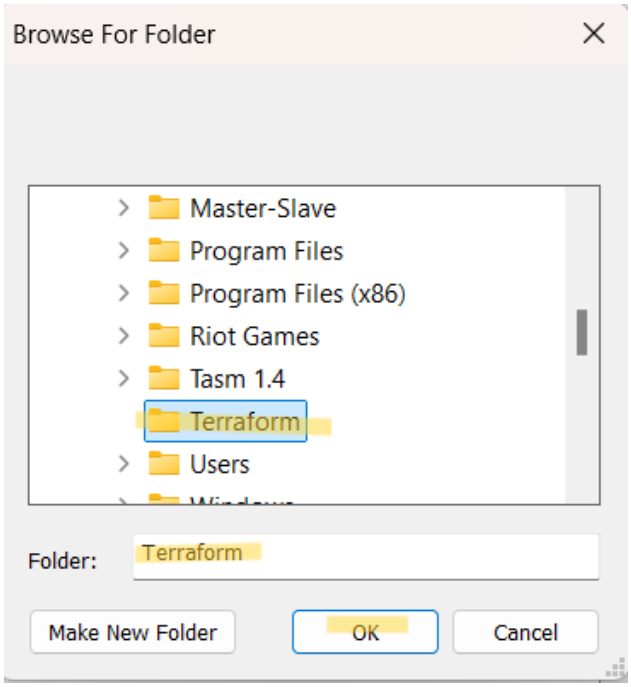
Step 3: After extracting the files, we need to add the folder path as a User System variable so that the commands could be used in the command prompt or powershell. For this, search Edit the system environment variables on you system. Open it, click on Environment Variables.

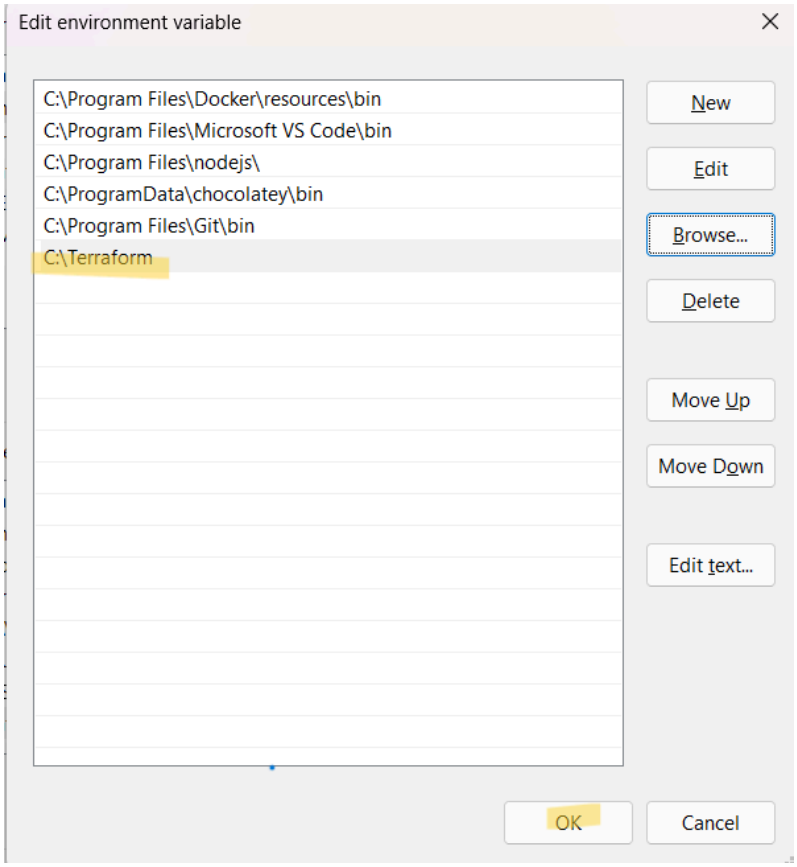


Step 4: This opens up an Environment Variables window. Here, under System Variables:
1) If a variable called Path does not exist, create the variable.
2) If Path variable exists, click on it; this will give you an option to edit the variable. Click on Edit.

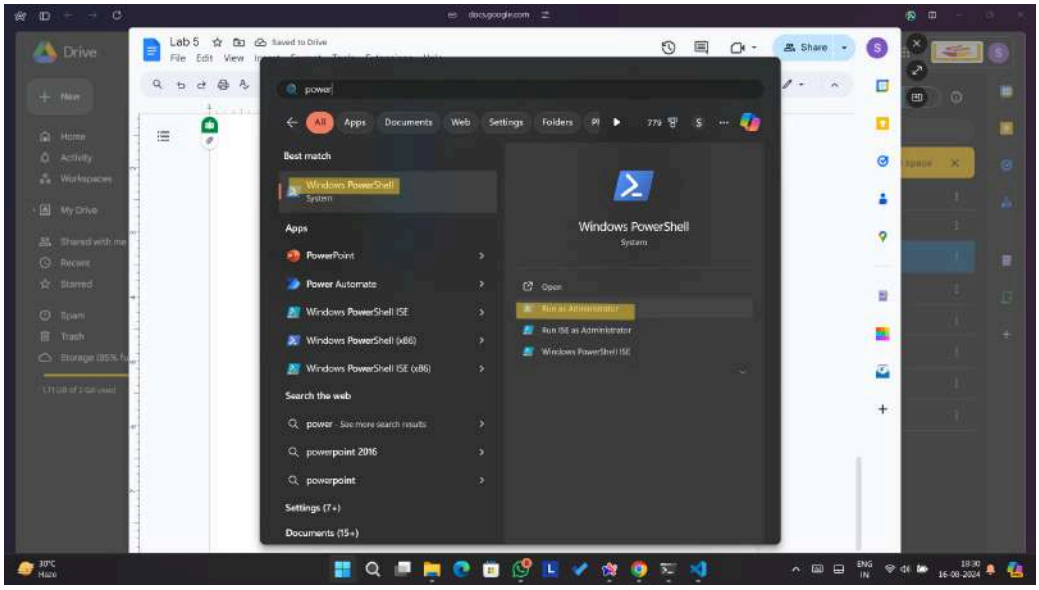


Step 5: This gives you a screen to add a path to the existing paths. Click on Browse, search for the Terraform folder (C:\Terraform) and click on Ok. This adds the link and it can be seen in the list of paths. Click on Ok.



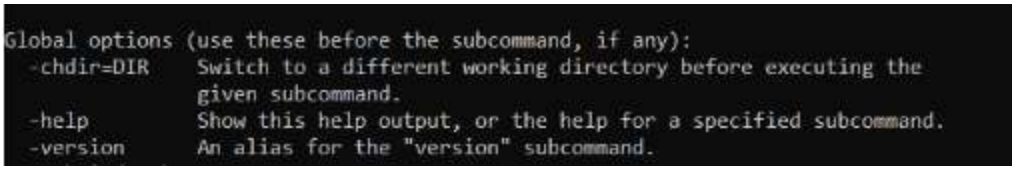


Step 6: Click on OK until you exit from all the Environment Variable screens. Now go to your search and search for Windows Powershell. Run this as an administrator.



Step 7: As the powershell opens, run the command terraform. This would give you all the main commands, other commands, and global options. If this is your output, then you have successfully installed Terraform on your system.

```
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
```



Conclusion:

The installation of Terraform on a Windows machine involves downloading the appropriate version, setting up the Terraform folder on the system, and configuring the system's environment variables to recognize Terraform commands. By adding the folder to the system path and verifying the installation via PowerShell, users can effectively utilize Terraform for infrastructure automation. This setup is essential for integrating Terraform into the system's command line, enabling smooth execution of Terraform commands for managing cloud infrastructure.

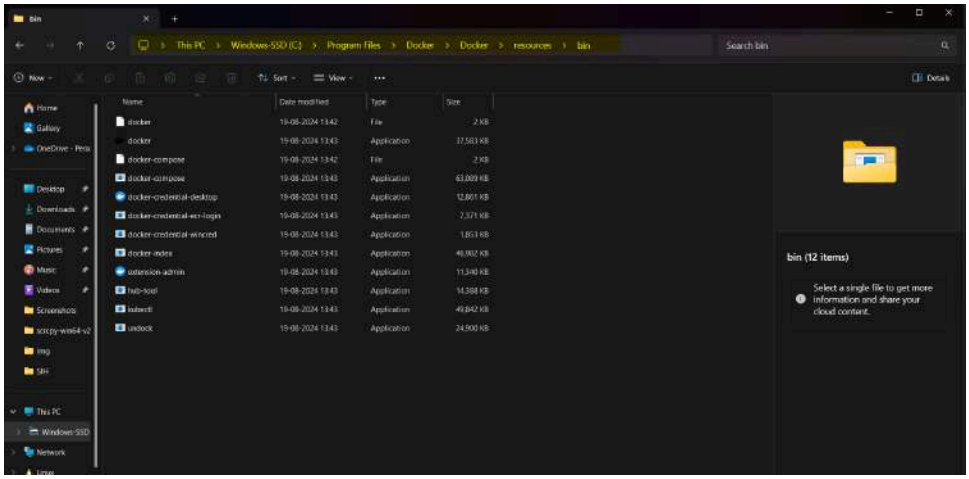
Experiment 6

Aim: To Build, change, and destroy AWS / GCP /Microsoft Azure/ DigitalOcean infrastructure Using Terraform.
(S3 bucket or Docker) fdp.

Step 1: For this experiment, you need to install docker on your computer. Go to https://www.docker.com/ and download the file according to the OS you have.
Open the file and start the installation.
Once installed, open your terminal and run ‘docker’ command.
If this is your output, then docker is installed successfully.

```
Microsoft Windows [Version 10.0.22631.3880]
(c) Microsoft Corporation. All rights reserved.

C:\Users\saira>docker

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

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

Management Commands:
  builder     Manage builds
  buildx*     Docker Buildx
  compose*    Docker Compose
  container   Manage containers
  context     Manage contexts
  debug*      Get a shell into any image or container
  desktop*    Docker Desktop commands (Alpha)
  dev*        Docker Dev Environments
  extension*  Manages Docker extensions
  feedback*   Provide feedback, right in your terminal!
  image       Manage images
  init*       Creates Docker-related starter files for your project
  manifest    Manage Docker image manifests and manifest lists
  network     Manage networks
  plugin      Manage plugins
  sbom*       View the packaged-based Software Bill Of Materials (SBOM) for an image
  scout*      Docker Scout
  system      Manage Docker
  trust       Manage trust on Docker images
  volume      Manage volumes

Swarm Commands:
  swarm       Manage Swarm

Commands:
  attach      Attach local standard input, output, and error streams to a running container
  commit      Create a new image from a container's changes
  cp          Copy files/folders between a container and the local filesystem
  create      Create a new container
  diff        Inspect changes to files or directories on a container's filesystem
  events      Get real time events from the server
  export      Export a container's filesystem as a tar archive
  history     Show the history of an image
  import      Import the contents from a tarball to create a filesystem image
  inspect     Return low-level information on Docker objects
  kill        Kill one or more running containers
  load        Load an image from a tar archive or STDIN
  logs        Fetch the logs of a container
  pause       Pause all processes within one or more containers
  port        List port mappings or a specific mapping for the container
  rename      Rename a container
  restart     Restart one or more containers
  rm          Remove one or more containers
```

If you det an error like ‘docker is not an internal or external command’, you need to add the bin path of docker to your environment variables.
Go to File Explorer, and follow this path: C drive → Program Files → Docker → Docker → Resources → bin. Copy this path by clicking on the bar having the path and using shortcut CTRL + C.



Open ‘Edit the System Environment Variables’ on your system. Click on Environment Variables. Now, check for a ‘Path’ variable under System variables, if it exists, click on it, then click on edit. Else, click on New and add the variable ‘Path’.
If the variable existed, click on Edit, then on New. This will give you a text box. Paste the path you copied here and click on ok until you close all the tabs.

Now run the docker command again and the output would appear.

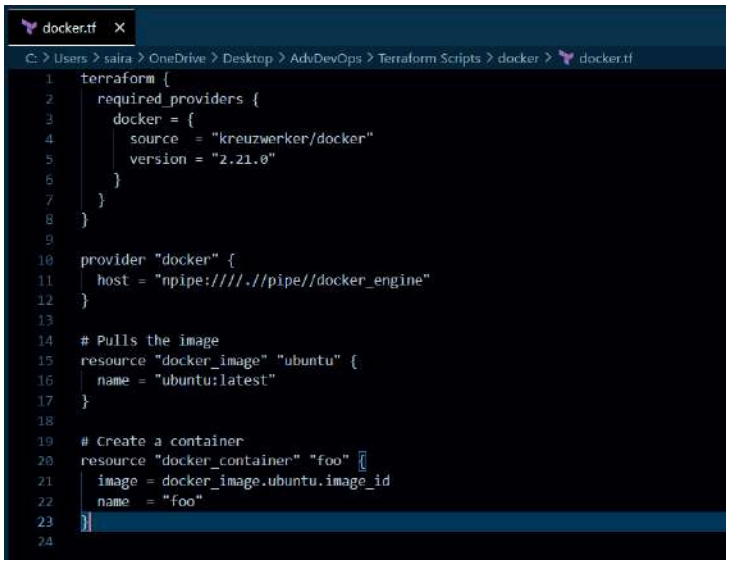
Alternatively, you could also run ‘docker –version’ to check whether docker is started on terminal.

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

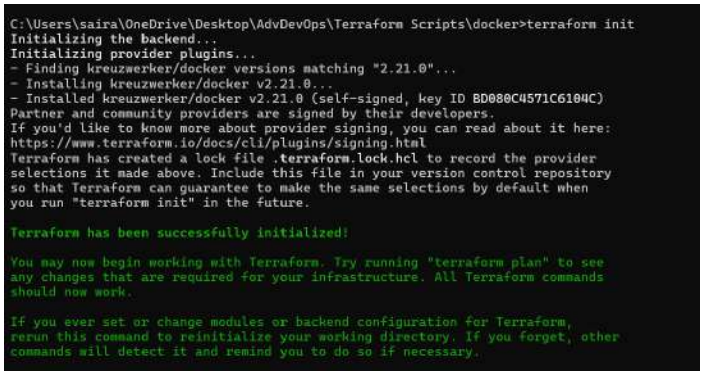
Step 2: Create a file called as ‘docker.tf’. Open the file and put the following code.

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

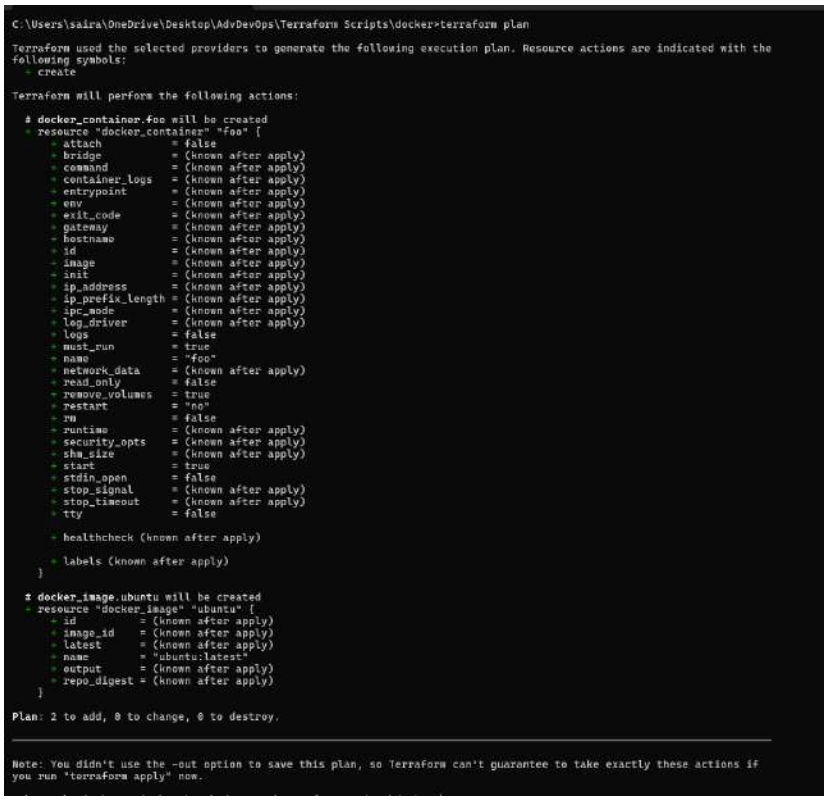
```
provider "docker" {
  host = "npipe:////.//pipe//docker_engine"
}

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

# Create a container
resource "docker_container" "foo" {
  image = docker_image.ubuntu.image_id
  name  = "foo"
}
```



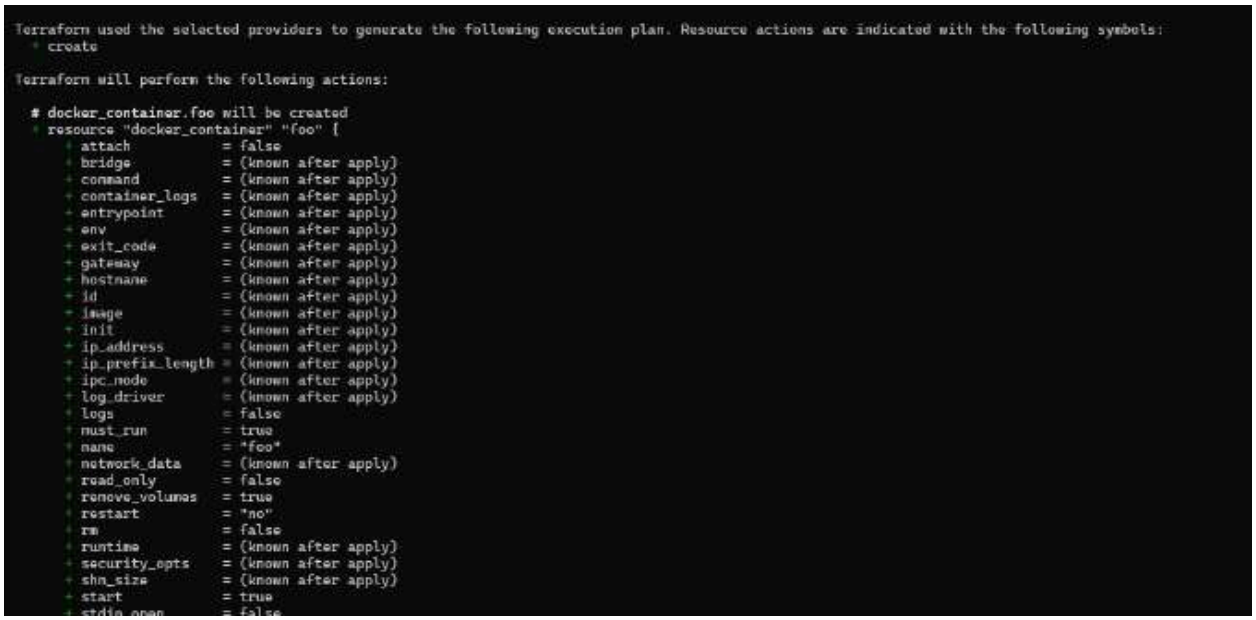
Step 3: Open the folder where the docker.tf is present on your terminal. Execute the command ‘terrafom init’. This will initialize terraform in the directory.



Step 4: Run the command ‘terraform plan’. This creates an execution plan and lets you overview changes that are going to happen in your infrastructure.



Step 5: Next, run command ‘terraform apply’. This command will carry out the changes that were to be made when ‘terrafrom plan’ command was executed.



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

      + healthcheck (known after apply)

      + labels (known after apply)
    }

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

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

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

  Enter a value: yes

docker_image.ubuntu: Creating...
docker_image.ubuntu: Creation complete after 8s [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Creating...

Error: container exited immediately

  with docker_container.foo,
  on docker.tf line 20, in resource "docker_container" "foo":
```

The script that we are using is going to throw an error.
Error: container exited immediately
This is because the script used is way too small or took a lot less time to execute. To fix this, we add a line to the code. ‘Command = [“sleep”, “infinity”]’.

This line of code lets docker know to keep the program in sleep mode for an infinite amount of time so that the output can be observed rather than stopping after running immediately.

Now rerun the ‘terraform apply’ code. It will ask you to enter yes to execute it. Type yes. The code gets executed and the image is formed.

```
C:\Users\saira\OneDrive\Desktop\AdvDevOps\Terraform Scripts\docker>terraform apply
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]

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

Terraform will perform the following actions:

  # docker_container.foo will be created
  + resource "docker_container" "foo" {
      + attach           = false
      + bridge           = (known after apply)
      + command          = [
          + "sleep",
          + "infinity",
        ]
      + 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            = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a"
      + init             = (known after apply)
      + ip_address       = (known after apply)
      + ip_prefix_length = (known after apply)
      + ipc_mode         = (known after apply)
      + log_driver       = (known after apply)
      + logs             = false
      + must_run         = true
      + name             = "foo"
      + network_data     = (known after apply)
      + read_only        = false
```

```
      + remove_volumes   = true
      + restart          = "no"
      + rm               = false
      + runtime          = (known after apply)
      + security_opts    = (known after apply)
      + shm_size         = (known after apply)
      + start            = true
      + stdin_open       = false
      + stop_signal      = (known after apply)
      + stop_timeout     = (known after apply)
      + tty              = false

      + healthcheck (known after apply)

      + labels (known after apply)
    }

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

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

  Enter a value: yes

docker_container.foo: Creating...
docker_container.foo: Creation complete after 1s [id=4baf3d214e59ae23d73cdef4193903a60d94f4d4b0f2ff8c2131891b7a652fca]

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

Using ‘docker images’ command, you can check the images that are present in docker.

‘docker images’ before ‘terraform apply’ is executed

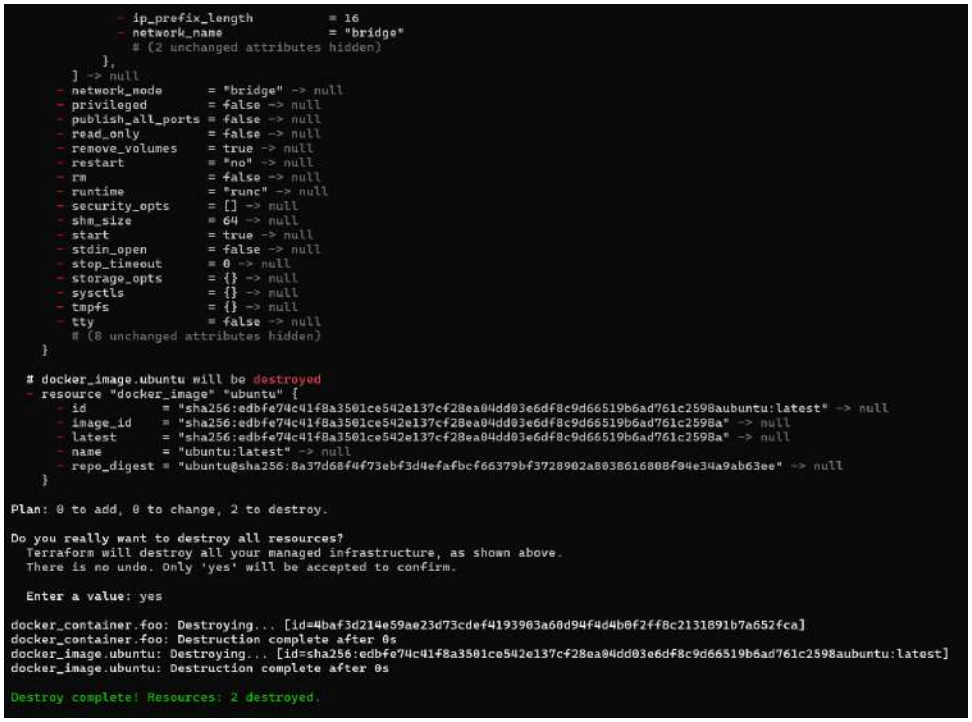
```
C:\Users\saira\OneDrive\Desktop\AdvDevOps\Terraform Scripts\docker>docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
```

‘docker images’ command after ‘terraform apply’ is executed

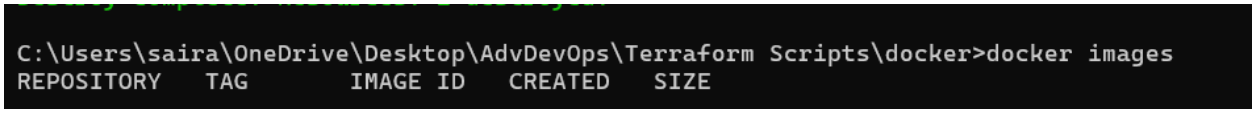
```
C:\Users\saira\OneDrive\Desktop\AdvDevOps\Terraform Scripts\docker>docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
ubuntu       latest    edbfe74c41f8   2 weeks ago   78.1MB
```

Step 6: Now that the image is created, we have to destroy it. For this, we use the ‘terraform destroy’ command. AGain, this command will ask for a promt to enter yes, as a confirmation to destroy the image we created. Type Yes.

```
C:\Users\saira\OneDrive\Desktop\AdvDevOps\Terraform Scripts\docker>terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Refreshing state... [id=4baf3d214e59ae23d73cdef4193903a60d94f4d4b0f2ff8c2131891b7a652fca]

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

Terraform will perform the following actions:

  # docker_container.foo will be destroyed
  - resource "docker_container" "foo" {
      - attach           = false -> null
      - command          = [
          - "sleep",
          - "infinity",
        ] -> null
      - cpu_shares       = 0 -> null
      - dns              = [] -> null
      - dns_opts         = [] -> null
      - dns_search       = [] -> null
      - entrypoint       = [] -> null
      - env              = [] -> null
      - gateway          = "172.17.0.1" -> null
      - group_add        = [] -> null
      - hostname         = "4baf3d214e59" -> null
      - id               = "4baf3d214e59ae23d73cdef4193903a60d94f4d4b0f2ff8c2131891b7a652fca" -> 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
```



Run the ‘docker images’ command again to check whether the image is destroyed or not.



Thus, we have created an image on docker using terraform and destroyed it.

Conclusion:

This experiment demonstrates how to build, modify, and destroy infrastructure using Terraform by interacting with Docker containers. By creating a Terraform configuration file (docker.tf), initializing Terraform in the directory, and applying changes, users can pull Docker images and manage containers efficiently. The key commands—terraform init, terraform plan, terraform apply, and terraform destroy—enable infrastructure automation. This setup showcases Terraform's capability to manage Docker containers, making it a versatile tool for multi-cloud and containerized infrastructure management.

Experiment 7

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

Prerequisites:

1) Docker
   Run docker -v command.g
   Use this command to check if docker is installed and running on your system.

```
PS C:\Users\saira\OneDrive\Desktop\AdvDevOps\lab7> docker -v
Docker version 27.0.3, build 7d4bcd8
```

2) Install SonarQube image
   Command: docker pull sonarqube
   This command helps you to install an image of SonarQube that can be used on the local system without actually installing the SonarQube installer.

```
PS C:\Users\saira\OneDrive\Desktop\AdvDevOps\lab7> docker pull sonarqube
Using default tag: latest
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
docker.io/library/sonarqube:latest

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

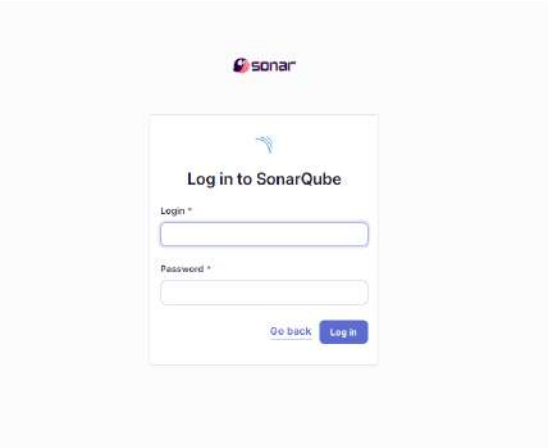
3) Keep jenkins installed on your system.

Experiment Steps:

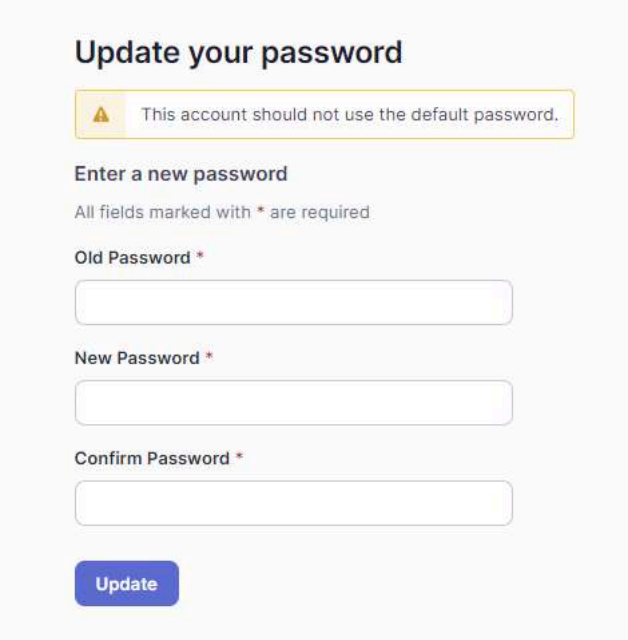
Step 1: Run SonarQube image
docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
This command will run the SonarQube image that was just installed using docker.

```
PS C:\Users\saira\OneDrive\Desktop\AdvDevOps\lab7> docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=t
rue -p 9000:9000 sonarqube:latest
36ff8a656bd28857ba9a28bf2bb0174099ae3232a9fc9ba2766d46f0c14d08a6
```

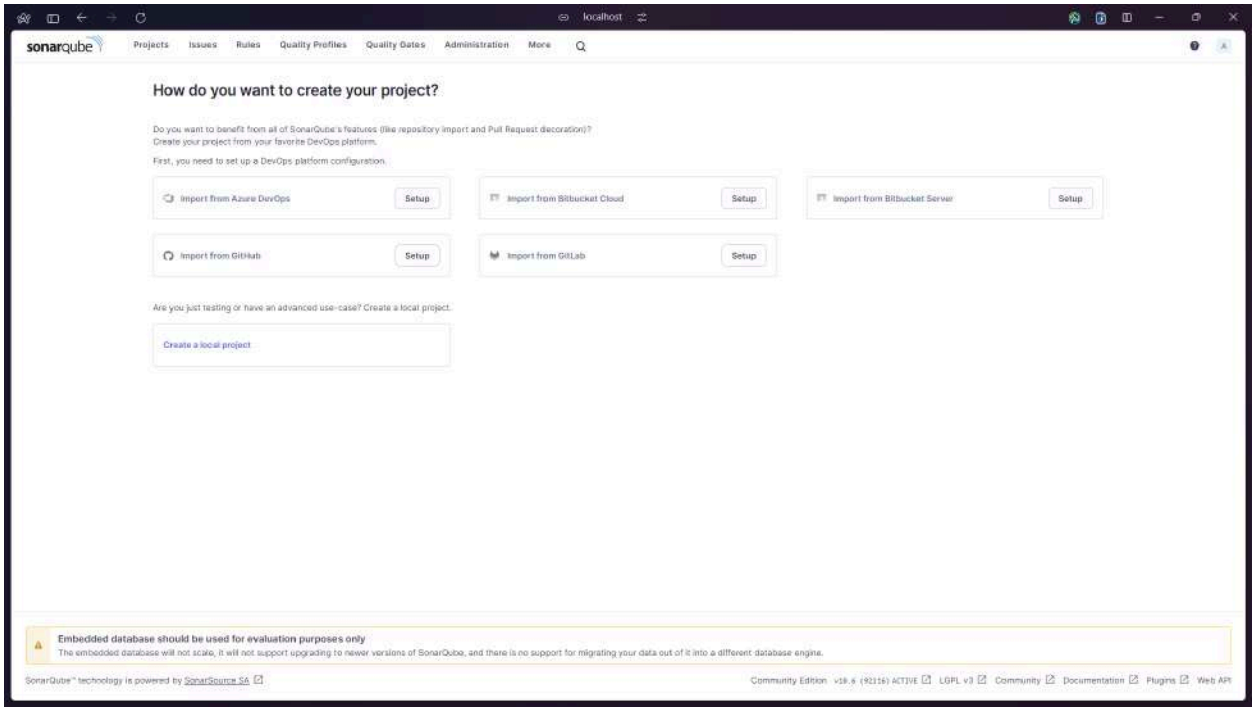
Step 2: Once the SonarQube image is started, you can go to http://localhost:9000 to find the SonarQube that has started



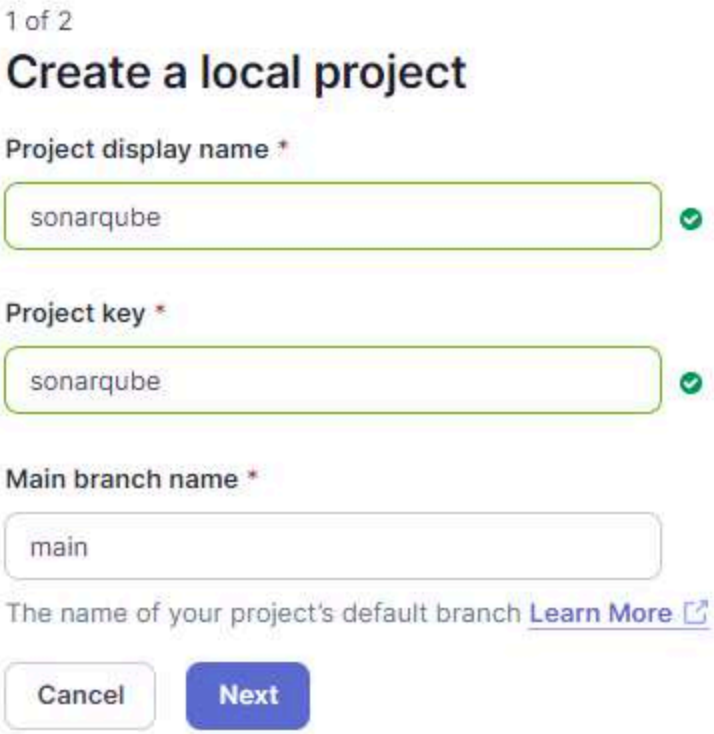
Step 3: On this interface, login with username = ‘admin’ and password = ‘admin’. Once logged in successfully, SonarQube will ask you to reset this password. Reset it and remember this password.



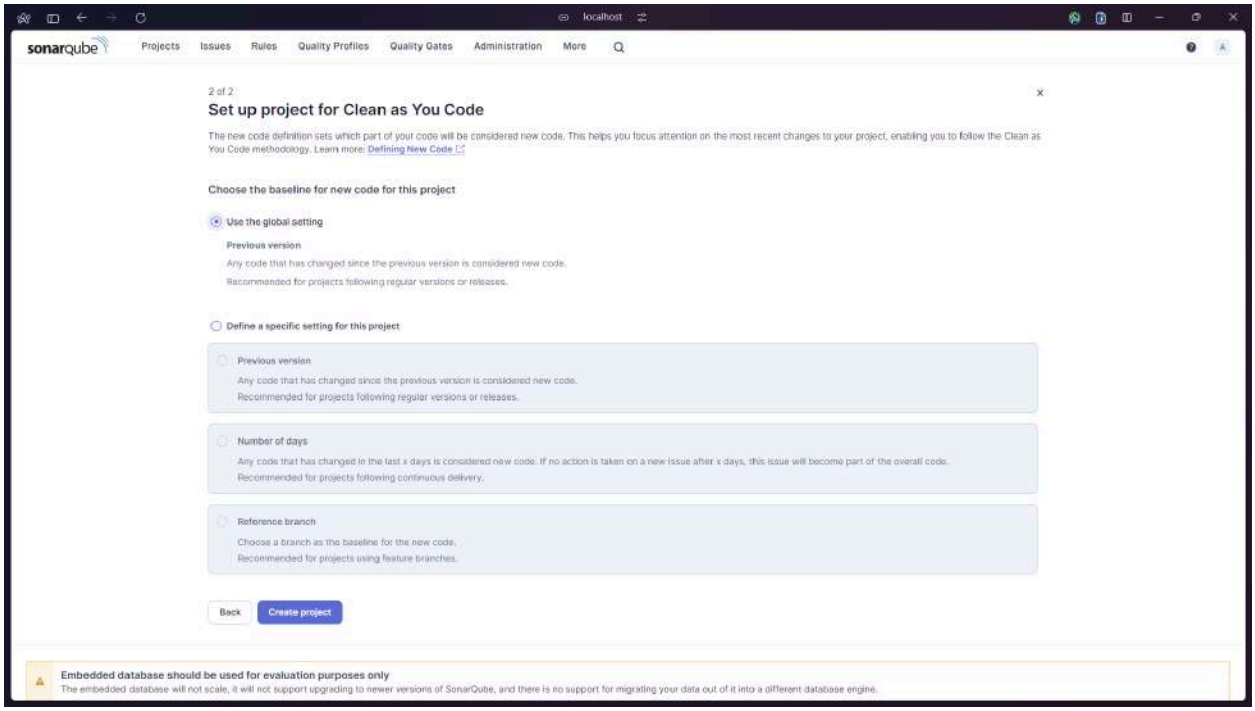
Step 4: After changing the password, you will be directed to this screen. Click on Create a Loal Project.



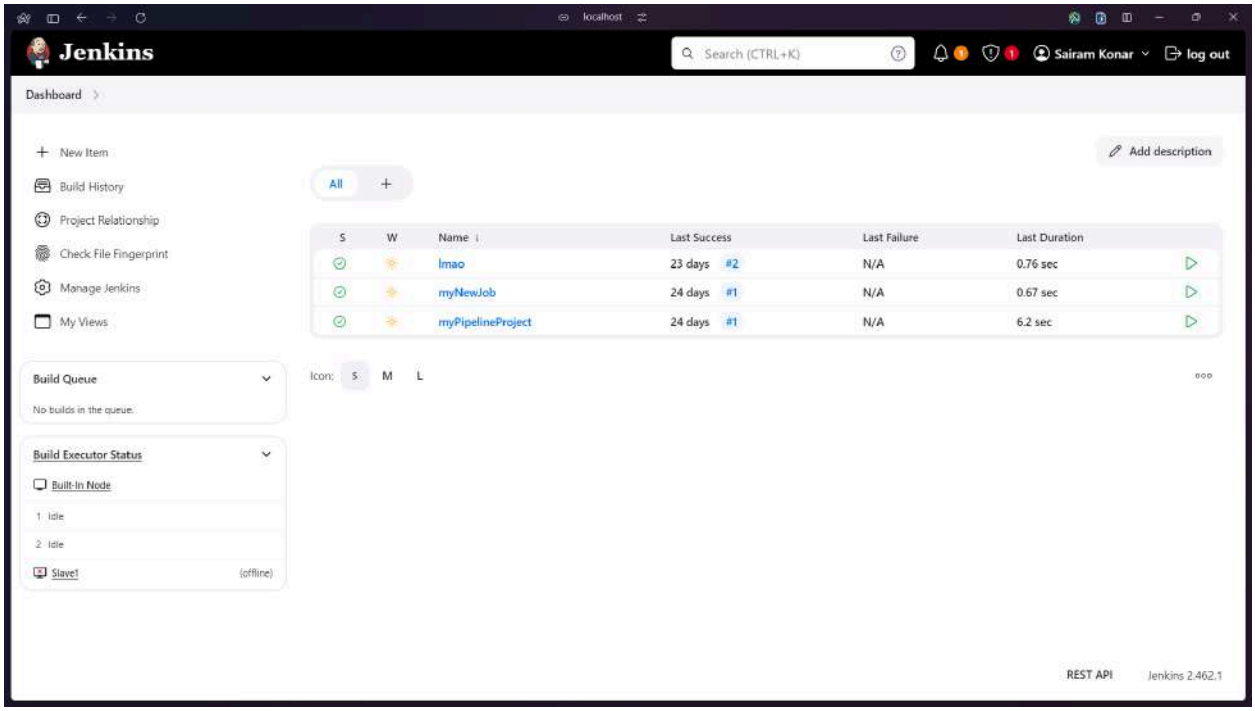
Give the project a display name and project key



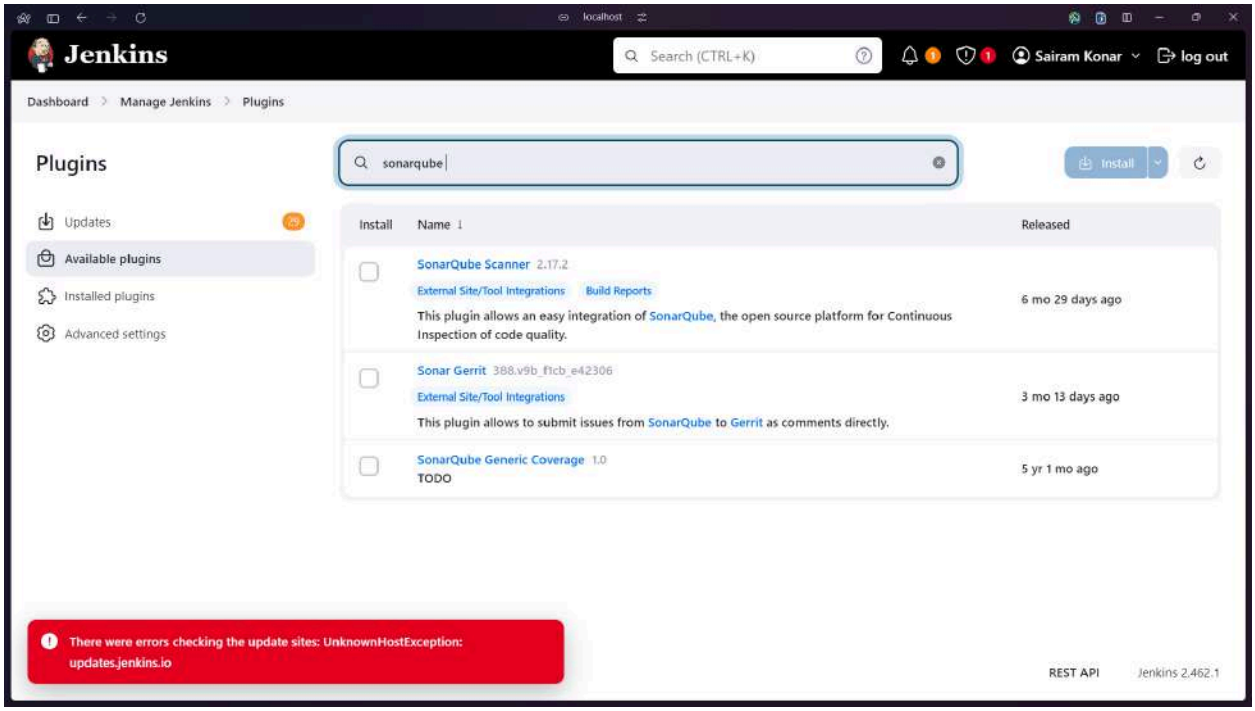
Set up the project as required and click on create.



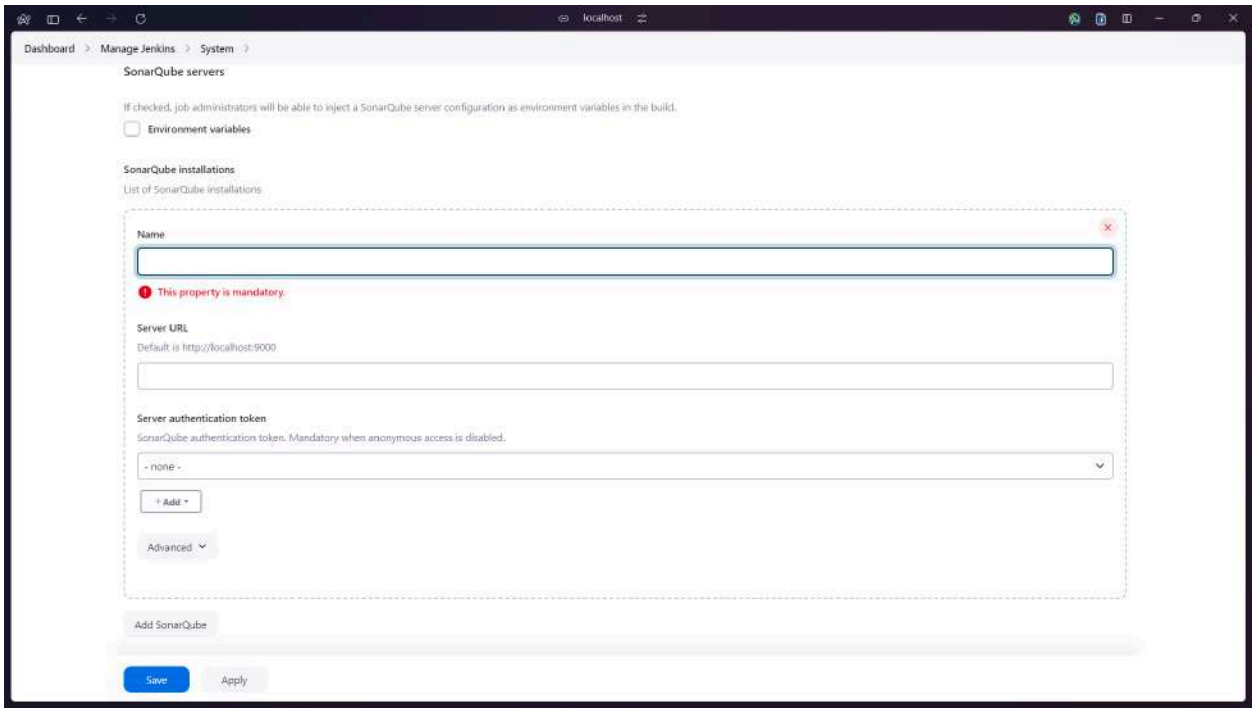
Step 5: Open Jenkins on whichever port it is installed. (http://loaclhost:<port_number>).



Step 6: Go to manage jenkins → Search for Sonarqube Scanner for Jenkins and install it.

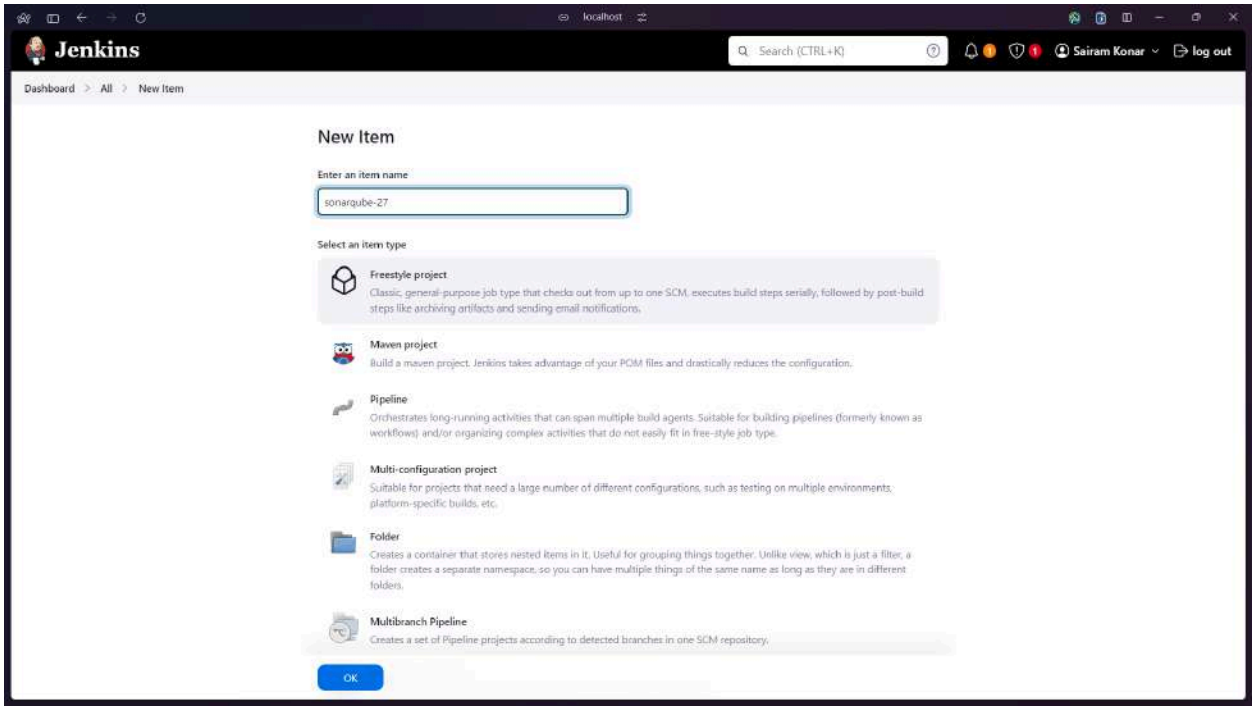


Step 7: Now, go to Manage Jenkins → System. Under Sonarqube servers, add a server. Add server authentication token if needed.

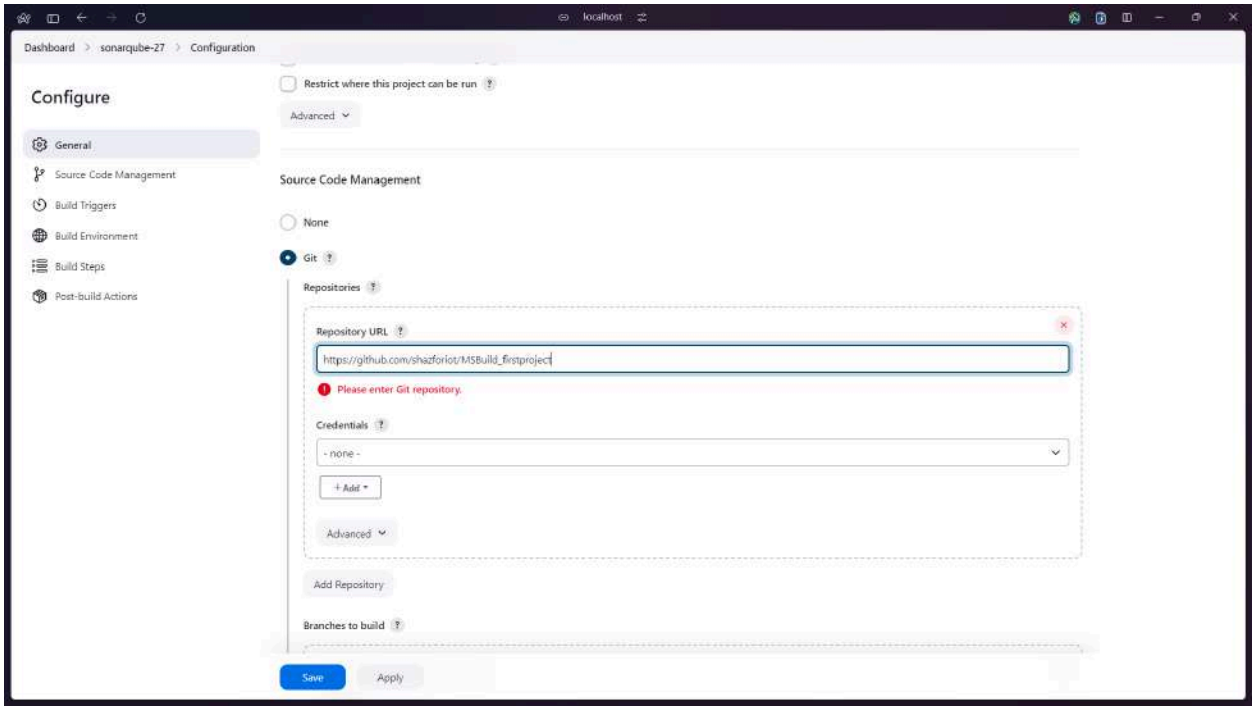


Step 8: Go to Manage Jenkins → Tools. Go to SonarQube scanner, choose the latest configuration and choose install automatically.

Step 9: After configuration, create a New Item → choose a freestyle project.



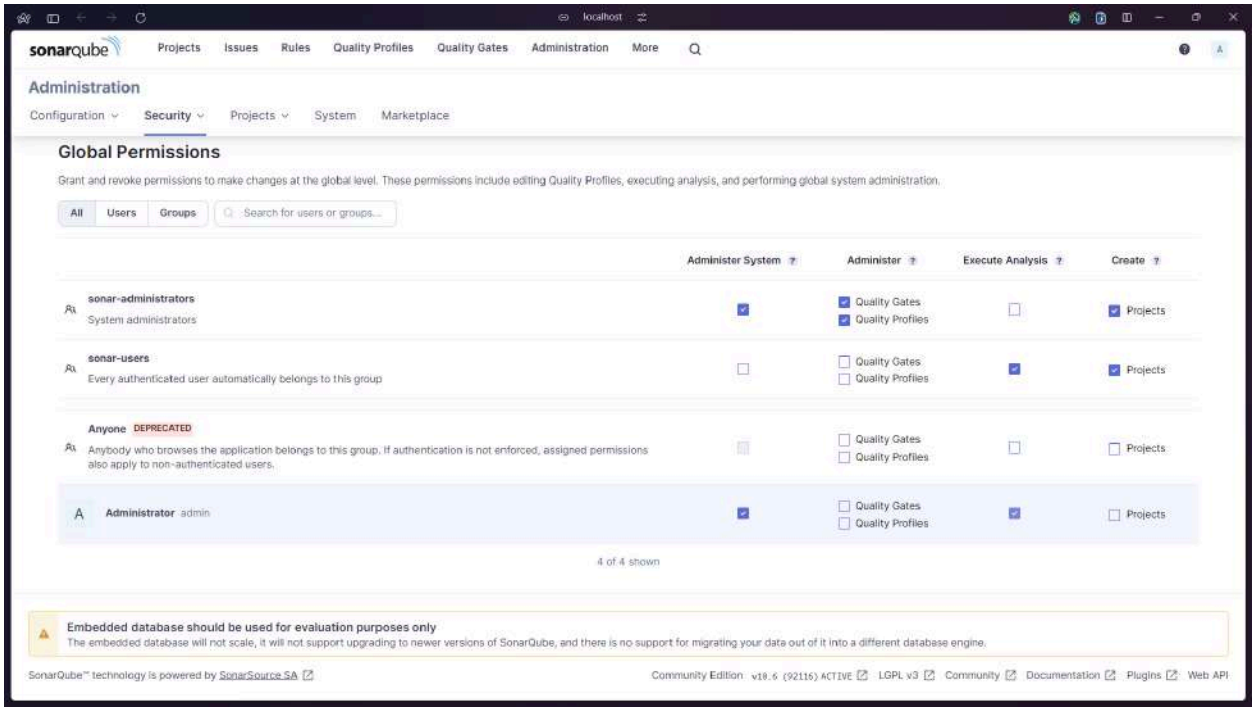
Step 10: Use this github repository in Source Code Management.
https://github.com/shazforiot/MSBuild_firstproject
It is a sample hello-world project with no vulnerabilities.



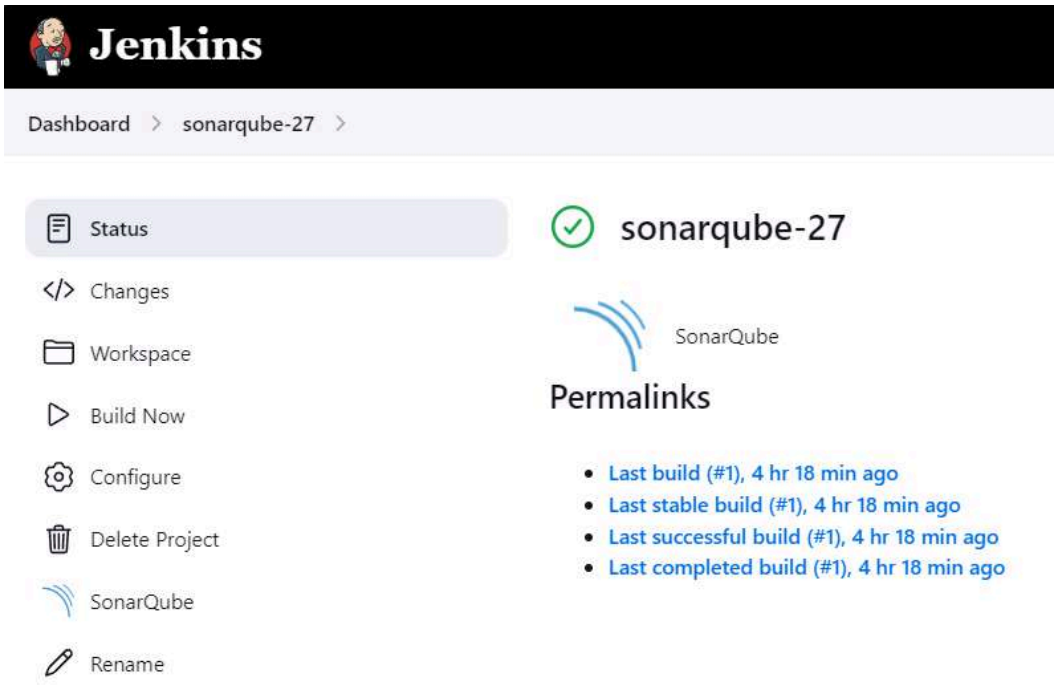
Step 11: Under Build Steps, enter Sonarqube Scanner, enter these Analysis properties. Mention the SonarQube Project Key, Login, Password, Source path and Host URL.

Step 12: Now, you need to grant the local user (here admin user) permissions to Execute the Analysis stage on SoanrQube. For this, go to http://loaclhost:<port_number>/admin/permissions and check the ‘Execute Analysis’

checkbox under Administrator.



Step 13: Go back to jenkins. Go to the job you had just built and click on Build Now.

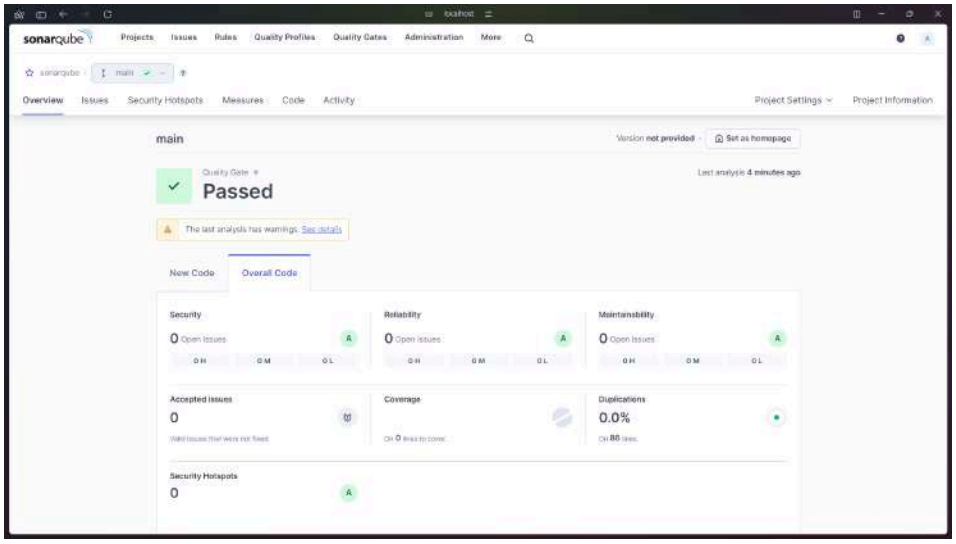


Check the console Output

```
Started by user Sairam Konar
Running as SYSTEM
Building on the built-in node in workspace C:\ProgramData\Jenkins\.jenkins\workspace\sonarqube-27
The recommended git tool is: NONE
No credentials specified
 > git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\sonarqube-27\.git # timeout=10
Fetching changes from the remote Git repository
 > git.exe config remote.origin.url https://github.com/shazforiot/MSBuild_firstproject # timeout=10
Fetching upstream changes from https://github.com/shazforiot/MSBuild_firstproject
 > git.exe --version # timeout=10
 > git --version # 'git version 2.45.2.windows.1'
 > git.exe fetch --tags --force --progress -- https://github.com/shazforiot/MSBuild_firstproject +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision f2bc042c04c6e72427c380bcaee6d6fee7b49adf (refs/remotes/origin/master)
 > git.exe config core.sparsecheckout # timeout=10
 > git.exe checkout -f f2bc042c04c6e72427c380bcaee6d6fee7b49adf # timeout=10
Commit message: "updated"
First time build. Skipping changelog.
[sonarqube-27] $ C:\ProgramData\Jenkins\.jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\sonarqube_lab\bin\sonar-scanner.bat -Dsonar.projectKey=sonarqube -
Dsonar.login=admin -Dsonar.host.url=http://localhost:9000 -Dsonar.sources=. -Dsonar.password=123456 -
Dsonar.projectBaseDir=C:\ProgramData\Jenkins\.jenkins\workspace\sonarqube-27
09:24:56.227 INFO  Scanner configuration file: C:\ProgramData\Jenkins\.jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\sonarqube_lab\bin\..\conf\sonar-
scanner.properties
09:24:56.231 INFO  Project root configuration file: NONE
```

```
09:25:27.830 WARN  Your project contains C# files which cannot be analyzed with the scanner you are using. To analyze C# or VB.NET, you must use the SonarScanner for .NET
5.x or higher, see https://redirect.sonarsource.com/doc/install-configure-scanner-msbuild.html
09:25:27.831 INFO  Sensor C# [csharp] (done) | time=1ms
09:25:27.831 INFO  Sensor Analysis Warnings import [csharp]
09:25:27.833 INFO  Sensor Analysis Warnings import [csharp] (done) | time=2ms
09:25:27.834 INFO  Sensor C# File Caching Sensor [csharp]
09:25:27.835 WARN  Incremental PR analysis: Could not determine common base path, cache will not be computed. Consider setting 'sonar.projectBaseDir' property.
09:25:27.835 INFO  Sensor C# File Caching Sensor [csharp] (done) | time=1ms
09:25:27.835 INFO  Sensor Zero Coverage Sensor
09:25:27.852 INFO  Sensor Zero Coverage Sensor (done) | time=18ms
09:25:27.856 INFO  SCM Publisher SCM provider for this project is: git
09:25:27.858 INFO  SCM Publisher 4 source files to be analyzed
09:25:28.360 INFO  SCM Publisher 4/4 source files have been analyzed (done) | time=501ms
09:25:28.363 INFO  CPD Executor Calculating CPD for 0 files
09:25:28.363 INFO  CPD Executor CPD calculation finished (done) | time=0ms
09:25:28.369 INFO  SCM revision ID 'f2bc042c04c6e72427c380bcaee6d6fee7b49adf'
09:25:28.697 INFO  Analysis report generated in 119ms, dir size=201.0 kB
09:25:28.754 INFO  Analysis report compressed in 43ms, zip size=22.4 kB
09:25:28.977 INFO  Analysis report uploaded in 221ms
09:25:28.981 INFO  ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=sonarqube
09:25:28.981 INFO  Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
09:25:28.981 INFO  More about the report processing at http://localhost:9000/api/ce/task?id=288a1357-213b-46ce-aeab-eb463fb79292
09:25:28.996 INFO  Analysis total time: 22.822 s
09:25:28.997 INFO  SonarScanner Engine completed successfully
09:25:29.032 INFO  EXECUTION SUCCESS
09:25:29.034 INFO  Total time: 32.809s
Finished: SUCCESS
```

Step 14: Once the build is complete, go back to SonarQube and check the project linked.



Conclusion:

In this experiment, we have learned how to perform Jenkins SAST using SonarQube. For this, we used a docker image of SonarQube so as to not install it locally on our system. After installing the required configurations on Jenkins, using a coe from a gihub repository, we analyze its code using SonarQube. Once we build the project, we can see that SonarQube project displays that the code has no errors.

Experiment 8

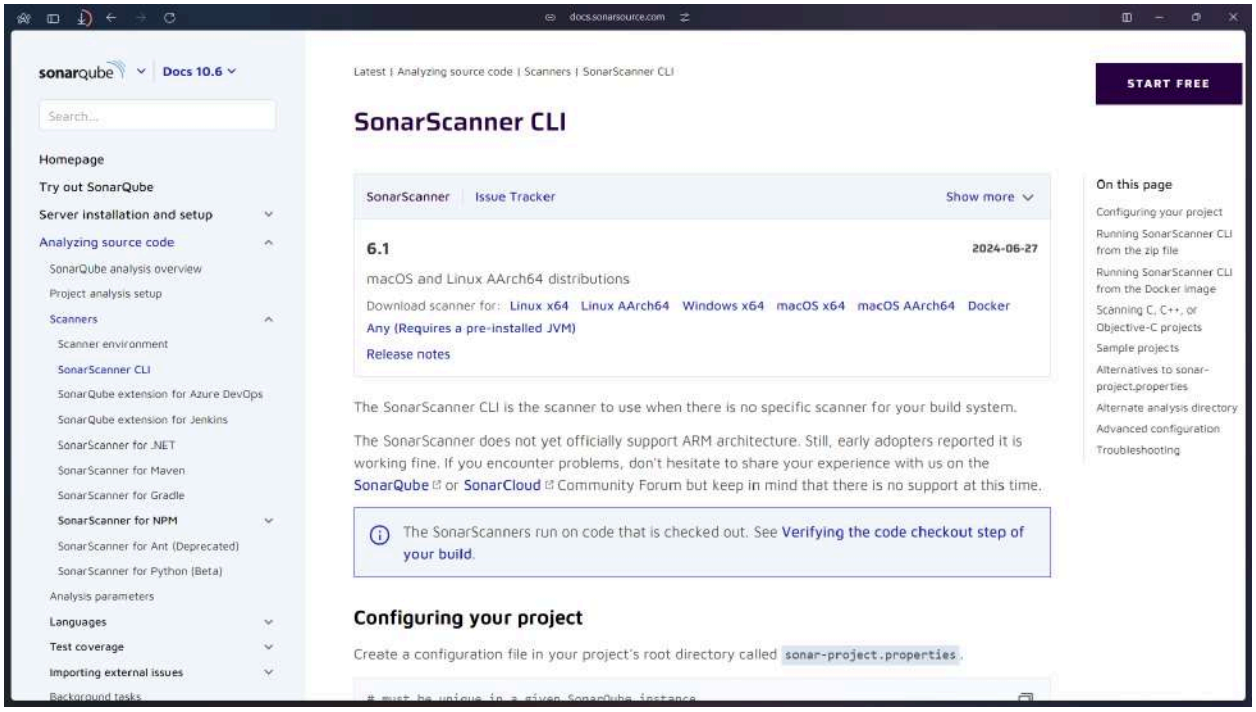
Sairam Konar D15C 27 2024-25

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.

Prerequisites:

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) Docker
   Run docker -v command.

```
PS C:\Users\saira\OneDrive\Desktop\AdvDevOps\lab7> docker -v
Docker version 27.0.3, build 7d4bcd8
```

2) Install sonarqube image

Command: docker pull sonarqube

```
PS C:\Users\saira\OneDrive\Desktop\AdvDevOps\lab7> docker pull sonarqube
Using default tag: latest
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
docker.io/library/sonarqube:latest

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

3) Keep Jenkins installed on your system.

Experiment Steps:

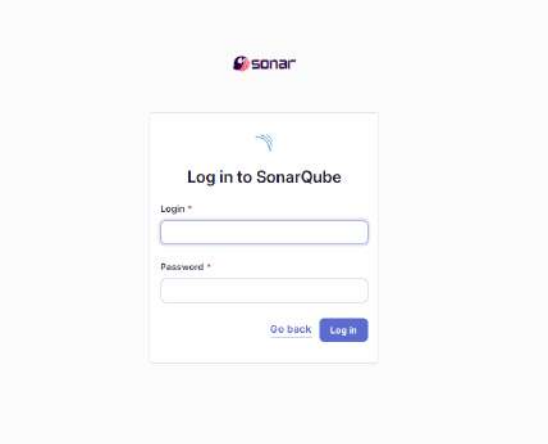
Step 1)Run SonarQube image

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

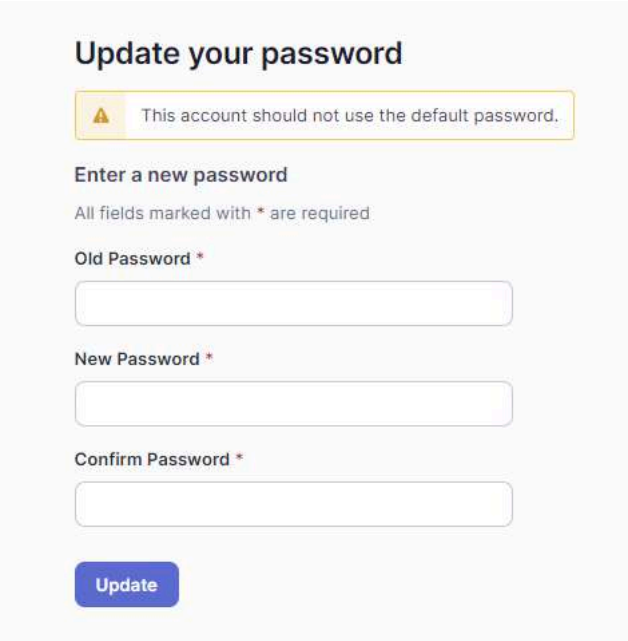
This command will run the SonarQube image that was just installed using docker.

```
PS C:\Users\saira\OneDrive\Desktop\AdvDevOps\lab7> docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=t
rue -p 9000:9000 sonarqube:latest
36ff8a656bd28857ba9a28bf2bb0174099ae3232a9fc9ba2766d46f0c14d08a6
```

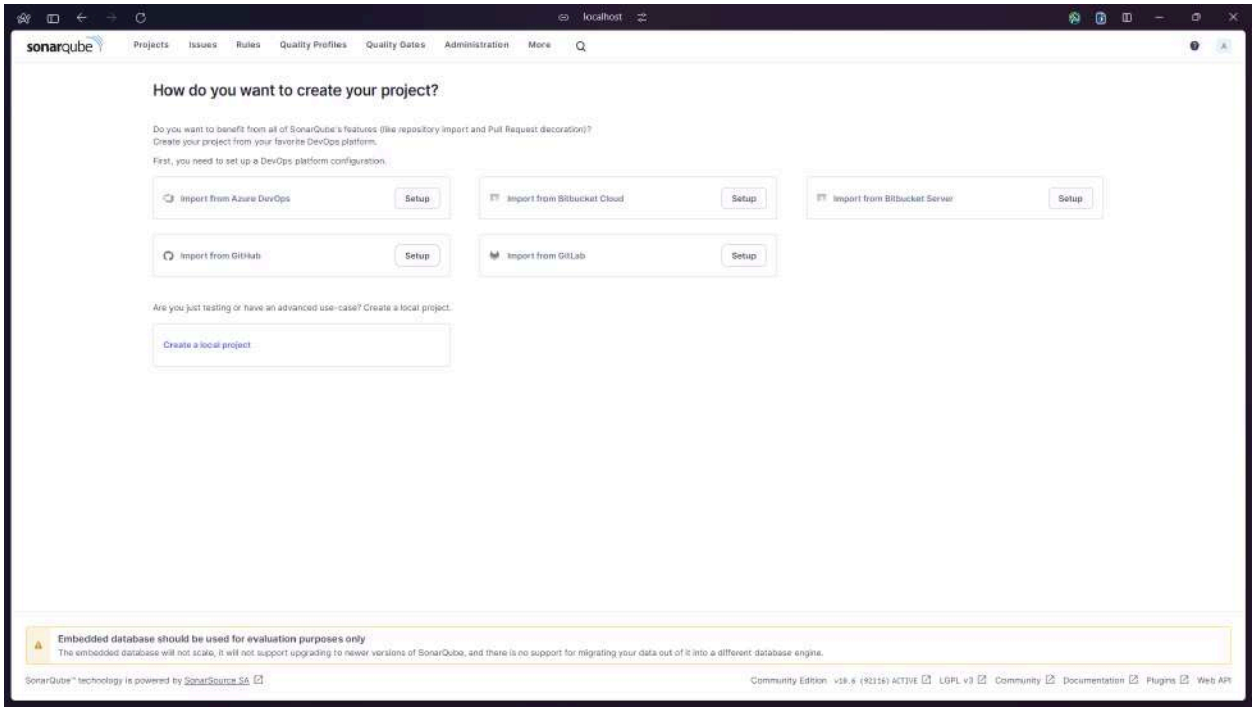
Step 2) Once the SonarQube image is started, you can go to http://localhost:9000 to find the SonarQube that has started



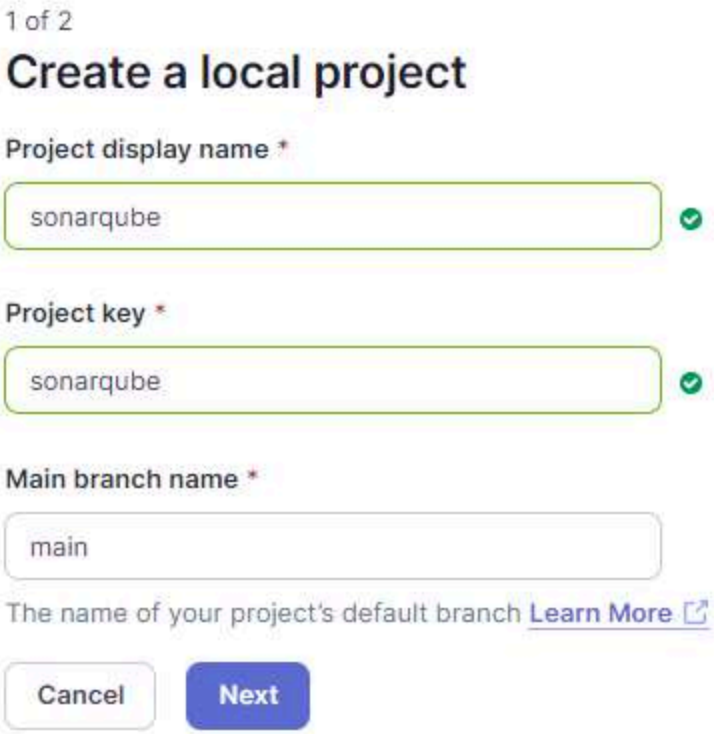
Step 3) On this interface, login with username = ‘admin’ and password = ‘admin’. Once logged in successfully, SonarQube will ask you to reset this password. Reset it and remember this password.



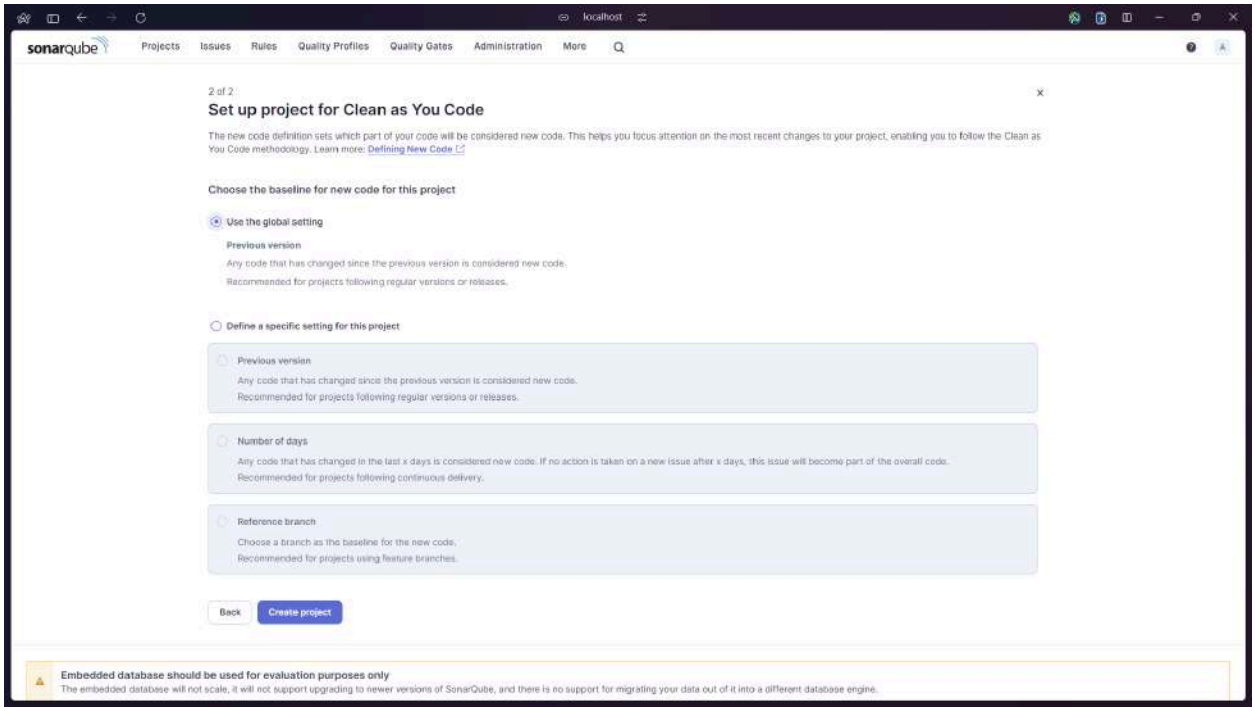
Step 4) After changing the password, you will be directed to this screen. Click on Create a Loal Project.



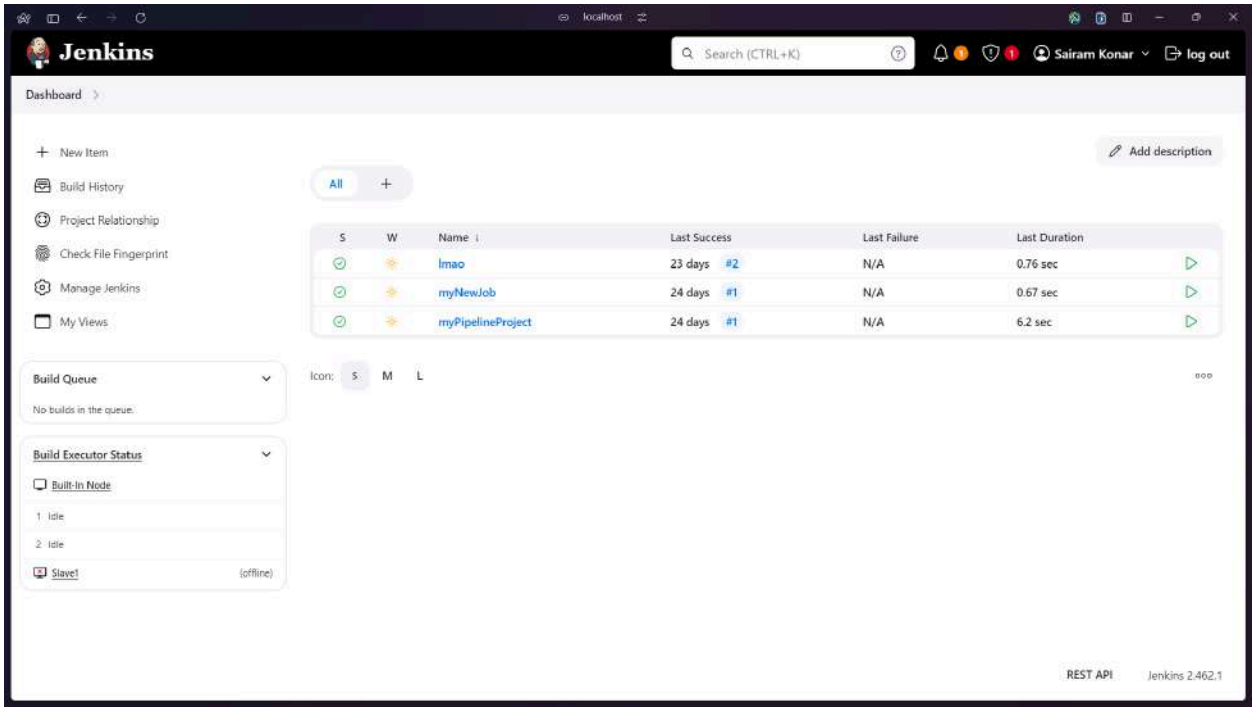
Give the project a display name and project key



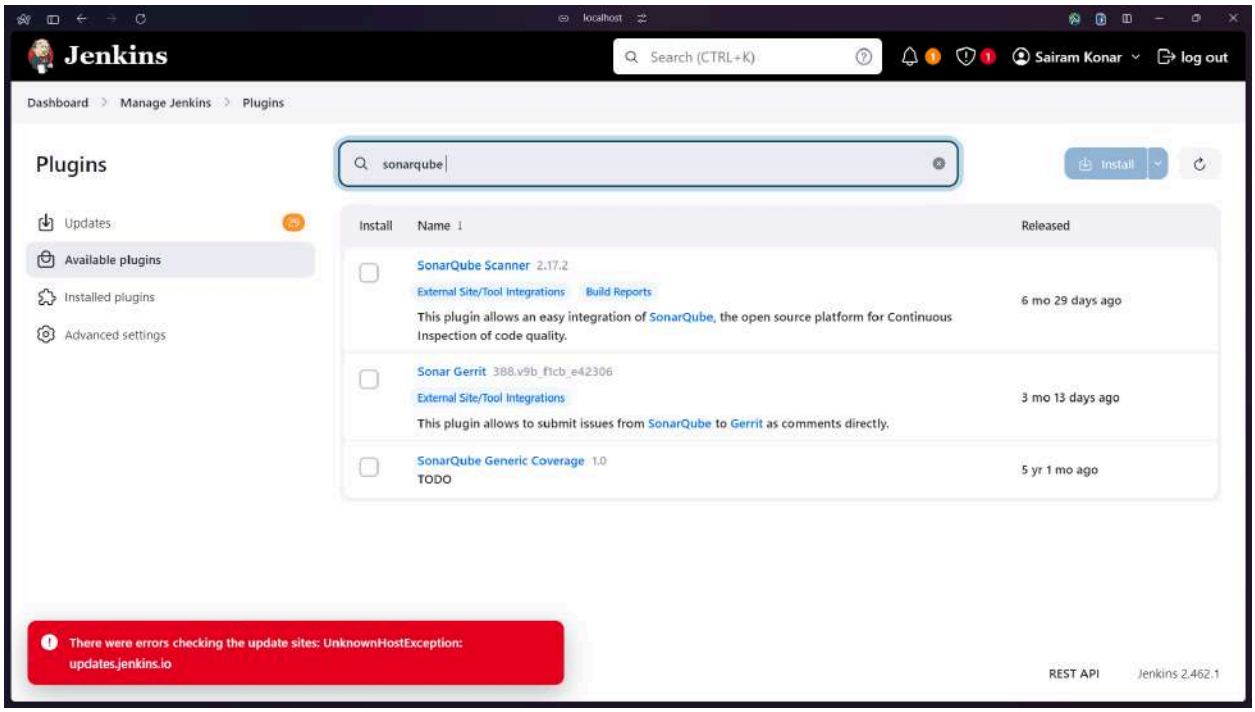
Set up the project as required and click on create.



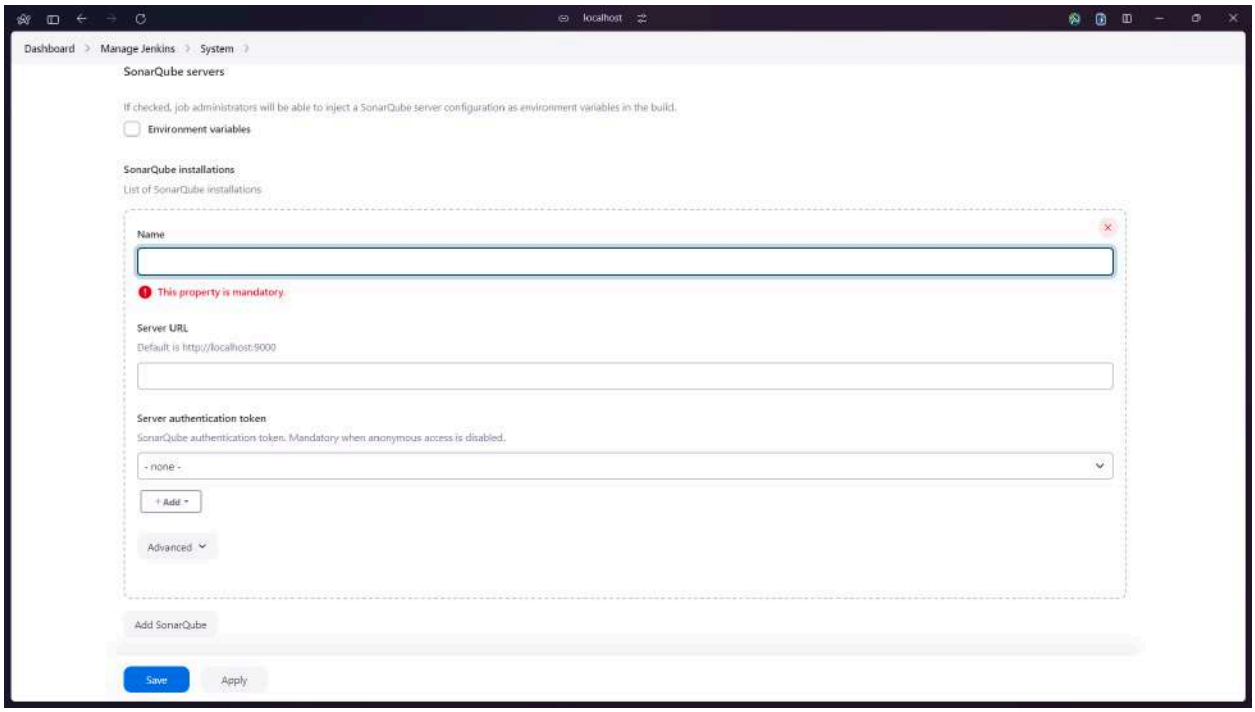
Step 5) Open Jenkins on whichever port it is installed. (http://loaclhost:<port_number>).



Step 6) Go to manage jenkins → Search for Sonarqube Scanner for Jenkins and install it.

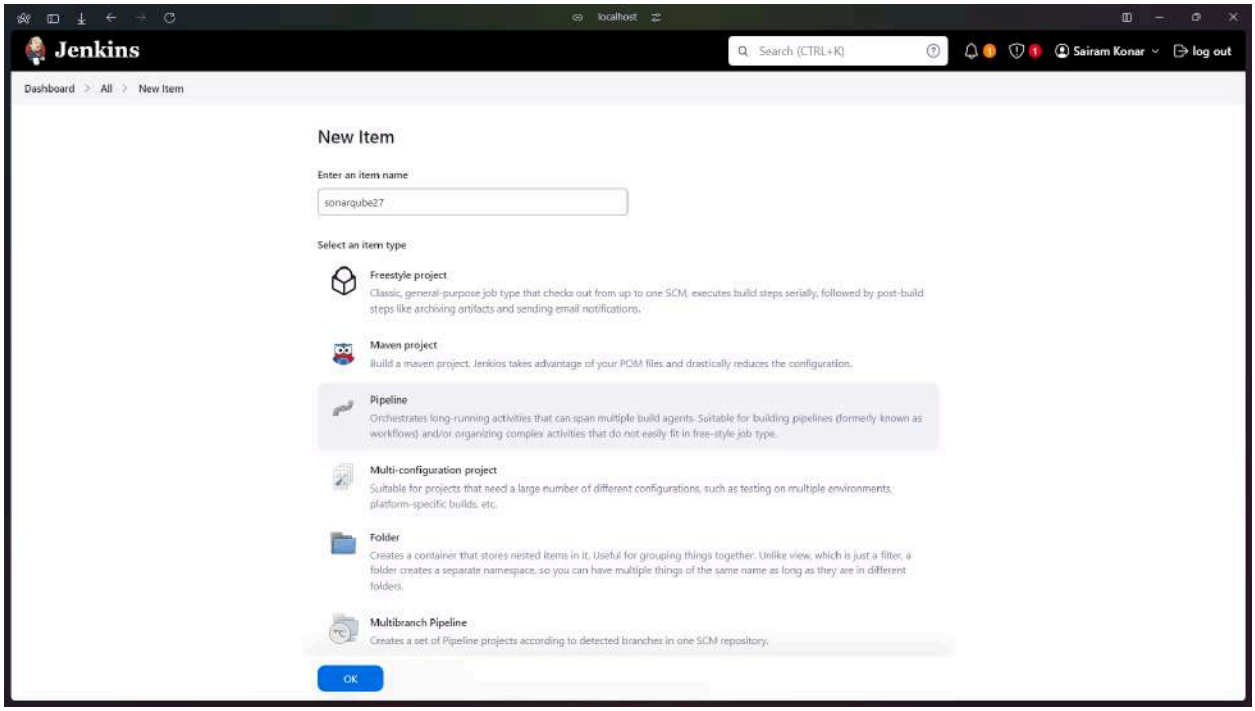


Step 7) Now, go to Manage Jenkins → System. Under Sonarqube servers, add a server. Add server authentication token if needed.



Step 8) Go to Manage Jenkins → Tools. Go to SonarQube scanner, choose the latest configuration and choose install automatically.

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

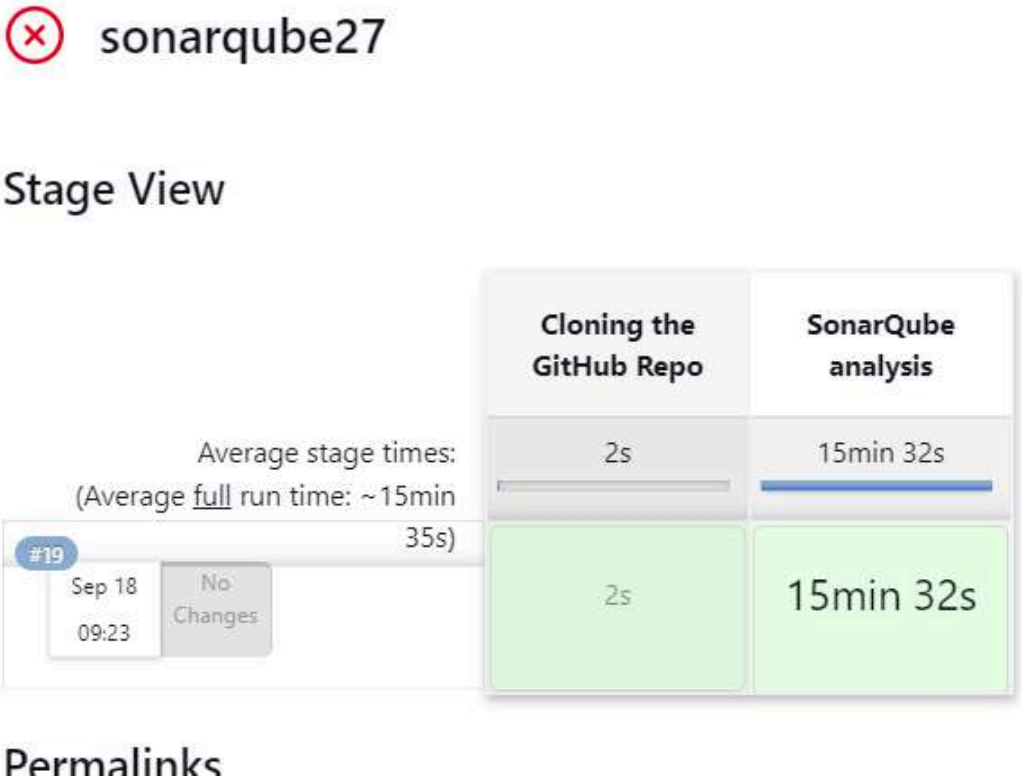


Step 10) Under Pipeline script, enter the following:

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

         stage('SonarQube analysis') {
                withSonarQubeEnv('sonarqube lab') {
                bat """
                        <PATH_TO_SONARSCANNER_FOLDER>\\bin\\sonar-scanner.bat ^
        -D sonar.login=<SONARQUBE_LOGIN> ^
        -D sonar.password=<SONARQUBE_PASSWORD> ^
        -D sonar.projectKey=<PROJECT_KEY> ^
        -D sonar.exclusions=vendor/**,resources/**,**/*.java ^
        -D sonar.host.url=http://localhost:9000/
         """
        }
    }
}
```

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

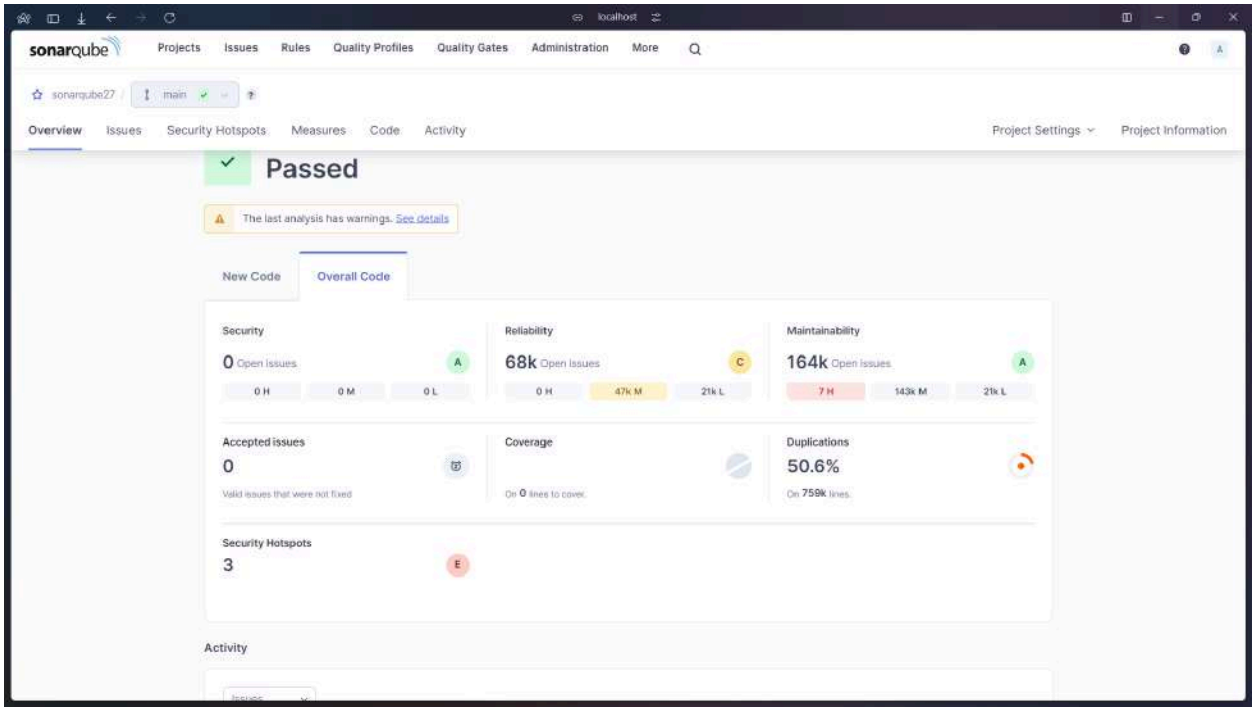
Step 11) Go back to jenkins. Go to the job you had just built and click on Build Now.



Check the console output once

```
first 100 references.
09:36:08.013 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/gui/action/Cut.html for block at line 75. Keep only the
first 100 references.
09:36:08.029 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/protocol/jdbc/config/package-summary.html for block at
line 39. Keep only the first 100 references.
09:36:08.029 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/protocol/jdbc/config/package-summary.html for block at
line 40. Keep only the first 100 references.
09:36:08.054 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/protocol/http/util/accesslog/Generator.html for block at
line 40. Keep only the first 100 references.
09:36:08.054 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/protocol/http/util/accesslog/Generator.html for block at
line 41. Keep only the first 100 references.
09:36:08.055 INFO  CPD Executor CPD calculation finished (done) | time=163405ms
09:36:08.069 INFO  SCM revision ID 'ba799ba7e1b576f04a4612322b0412c5e6e1e5e4'
09:38:20.750 INFO  Analysis report generated in 4390ms, dir size=127.2 MB
09:38:45.857 INFO  Analysis report compressed in 25089ms, zip size=29.6 MB
09:38:46.532 INFO  Analysis report uploaded in 675ms
09:38:46.533 INFO  ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=sonarqube27
09:38:46.533 INFO  Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
09:38:46.533 INFO  More about the report processing at http://localhost:9000/api/ce/task?id=46576333-cbde-4277-89d7-471ee554de32
09:39:00.038 INFO  Analysis total time: 15:24.256 s
09:39:00.041 INFO  SonarScanner Engine completed successfully
09:39:00.810 INFO  EXECUTION SUCCESS
09:39:00.811 INFO  Total time: 15:29.301s
[Pipeline] }
[Pipeline] // withSonarQubeEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

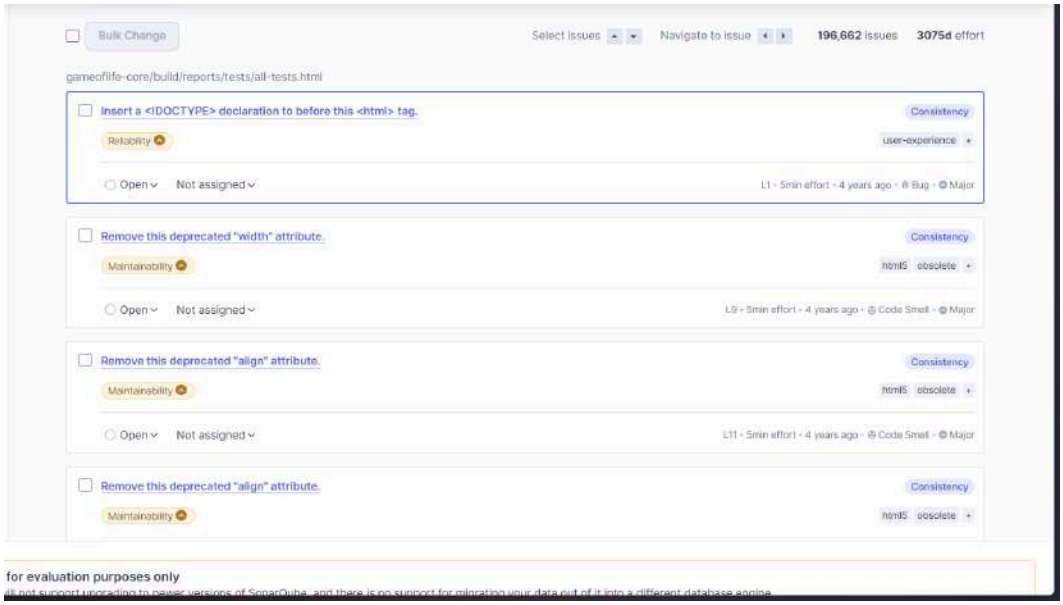
Step 12) Once the build is complete, go back to SonarQube and check the project linked.



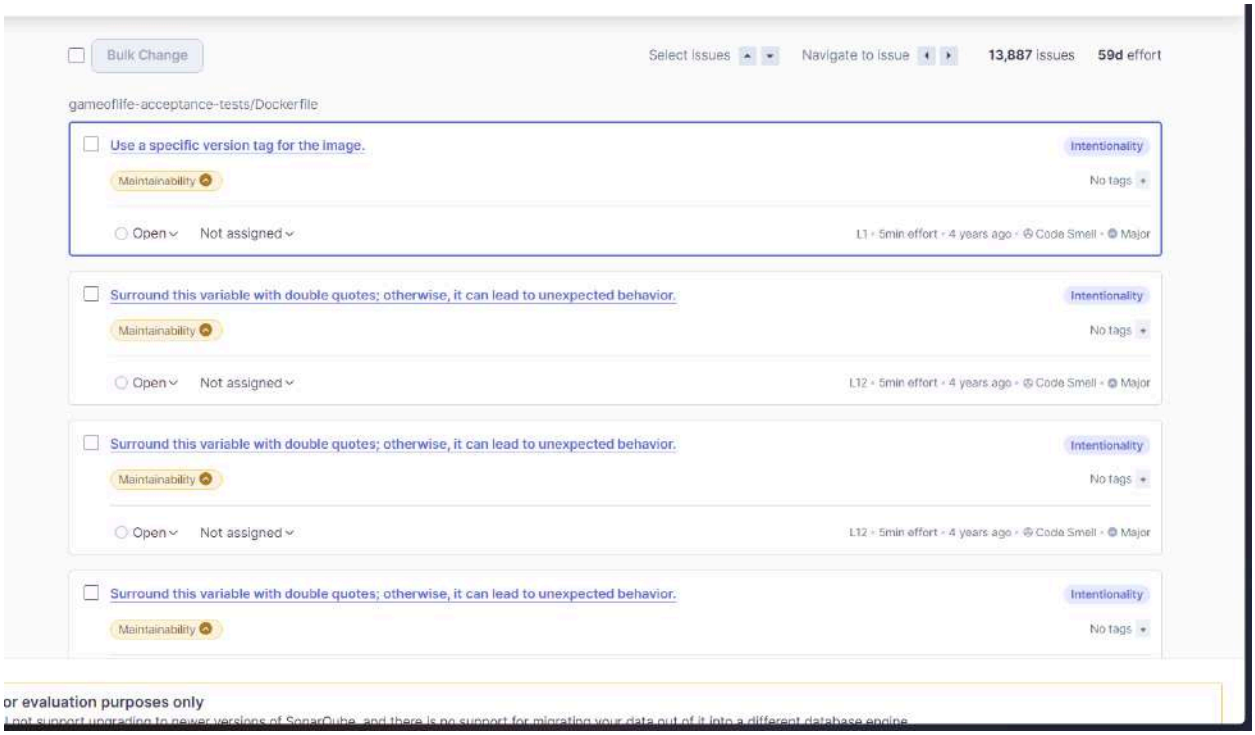
Under different tabs, check all the issues with the code.

- Code Problems
- Consistency

Sairam Konar D15C 27 2024-25

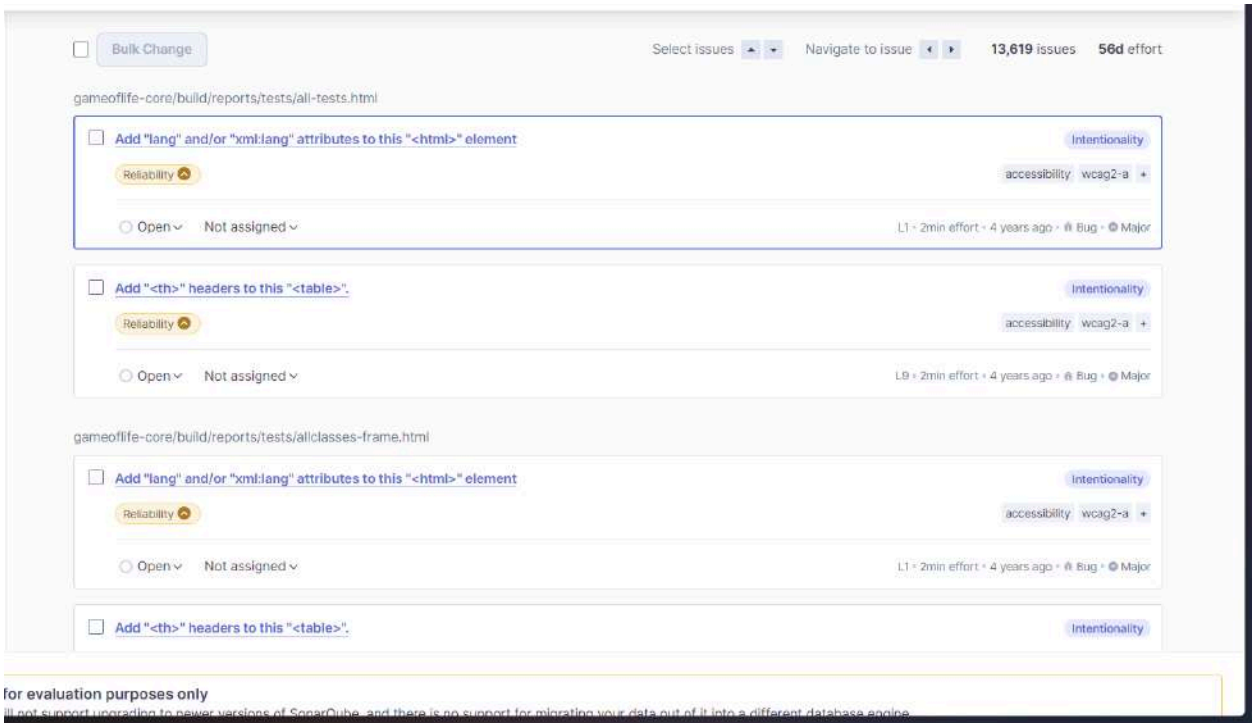


- Intentionality

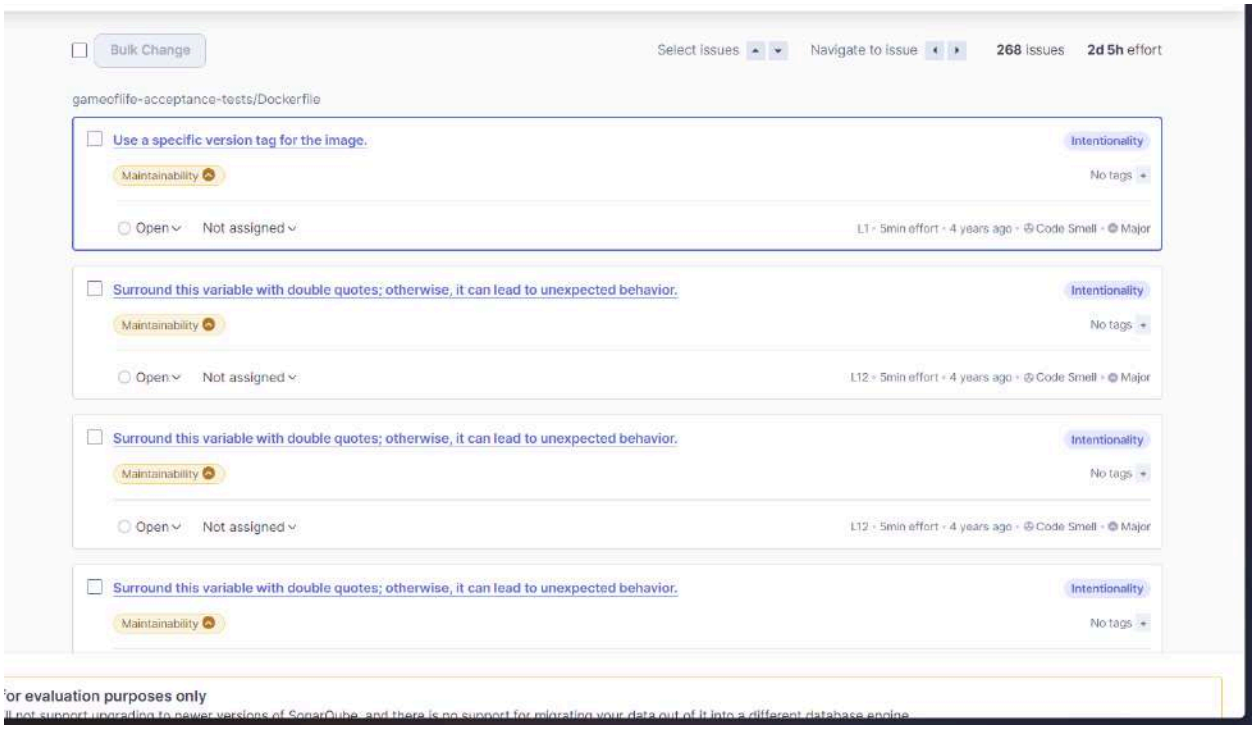


Sairam Konar D15C 27 2024-25

- Bugs

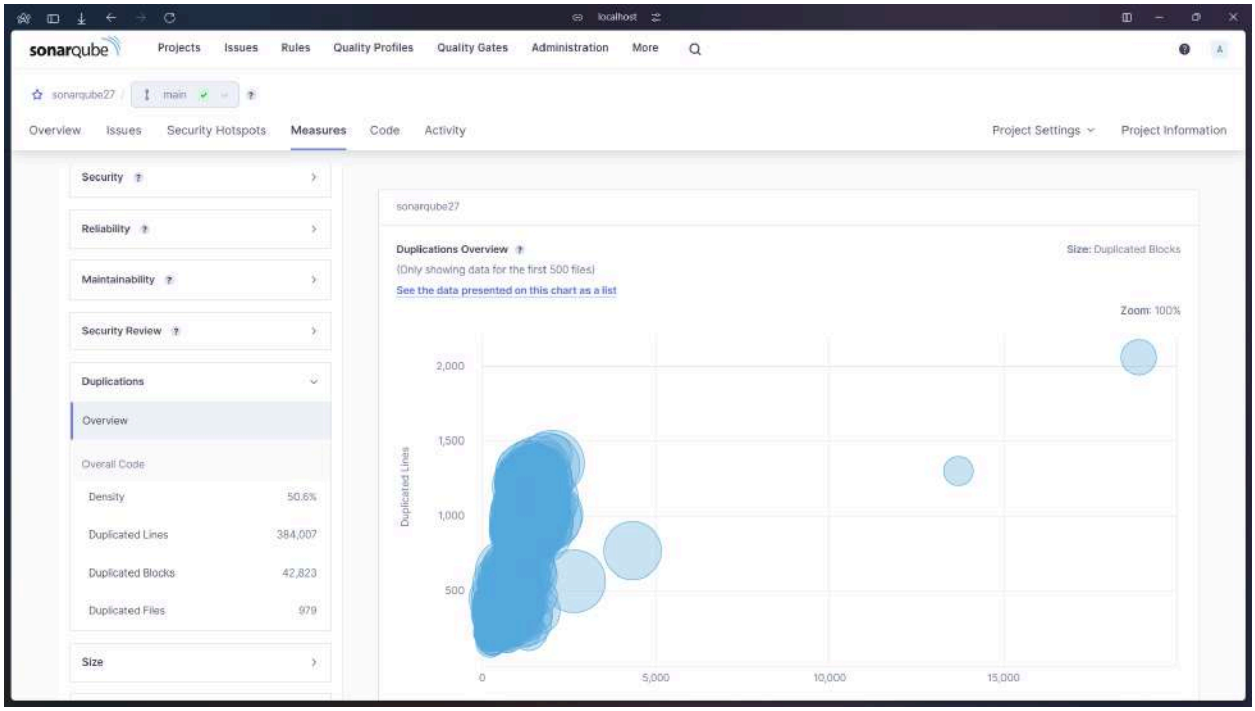


- Code Smells

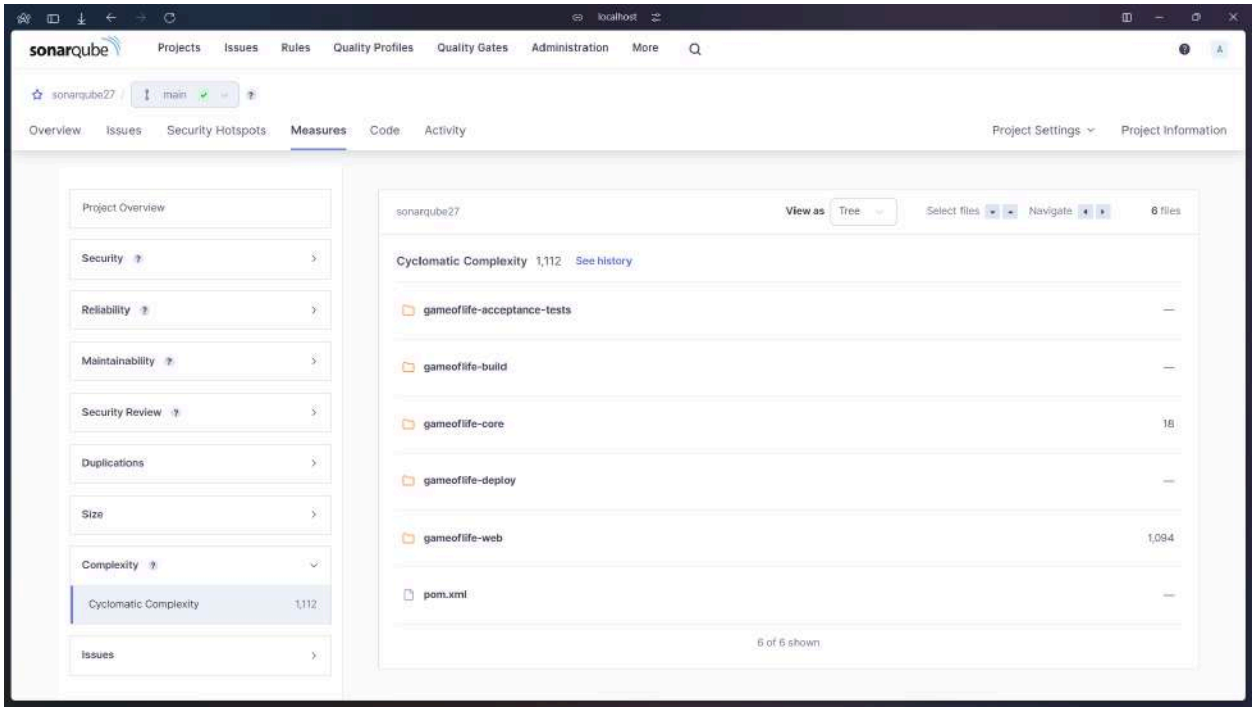


Sairam Konar D15C 27 2024-25

- Duplications



- Cyclomatic Complexities

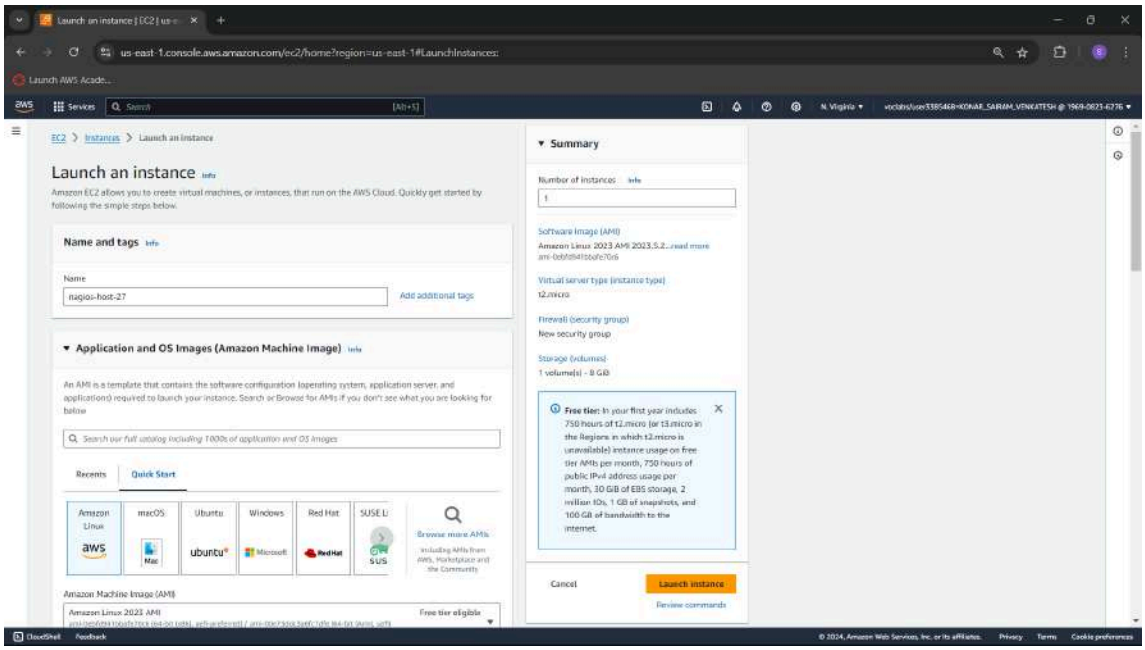


Conclusion: In this experiment, we have learned how to perform static analysis of a code using Jenkins CI/CD Pipeline with SonarQune analysis. A pipeline project is to be created which is given a pipeline script. This script contains all the information needed for the project to run the SonarQube analysis. After the necessary configurations are made on jenkins, the Jenkins project is built. The code provided in this experiment contains lots of error, bugs, duplications which can be checked on the SonarQube project linked with this build.

Experiment 9

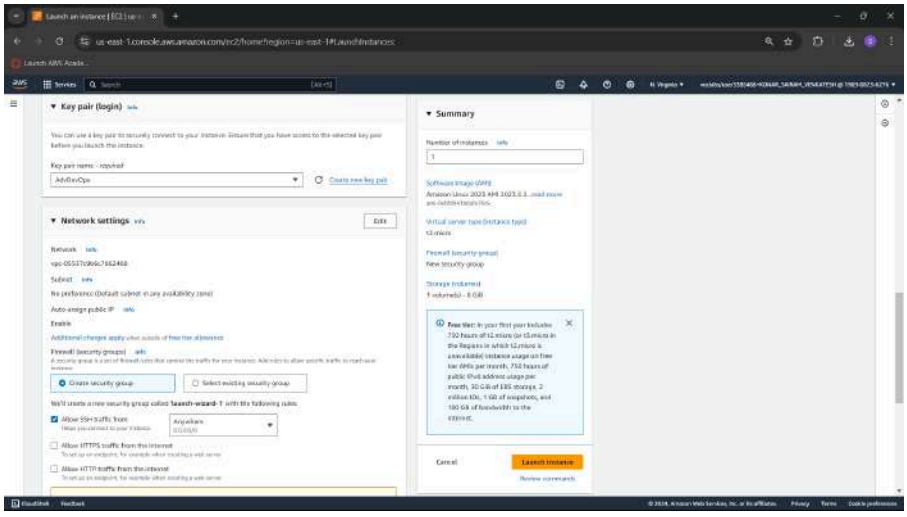
Aim: To Understand Continuous monitoring and Installation and configuration of Nagios Core, Nagios Plugins and NRPE (Nagios Remote Plugin Executor) on Linux Machine.

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

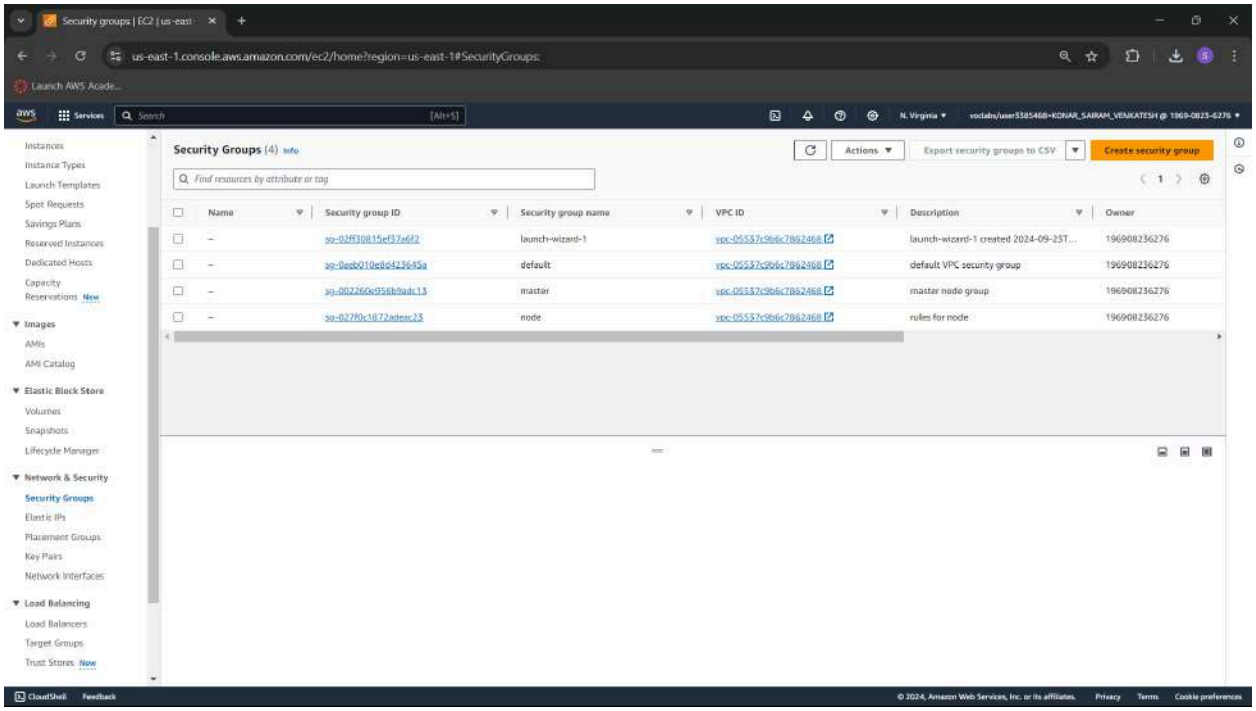


Select The OS Image as Amazon Linux.

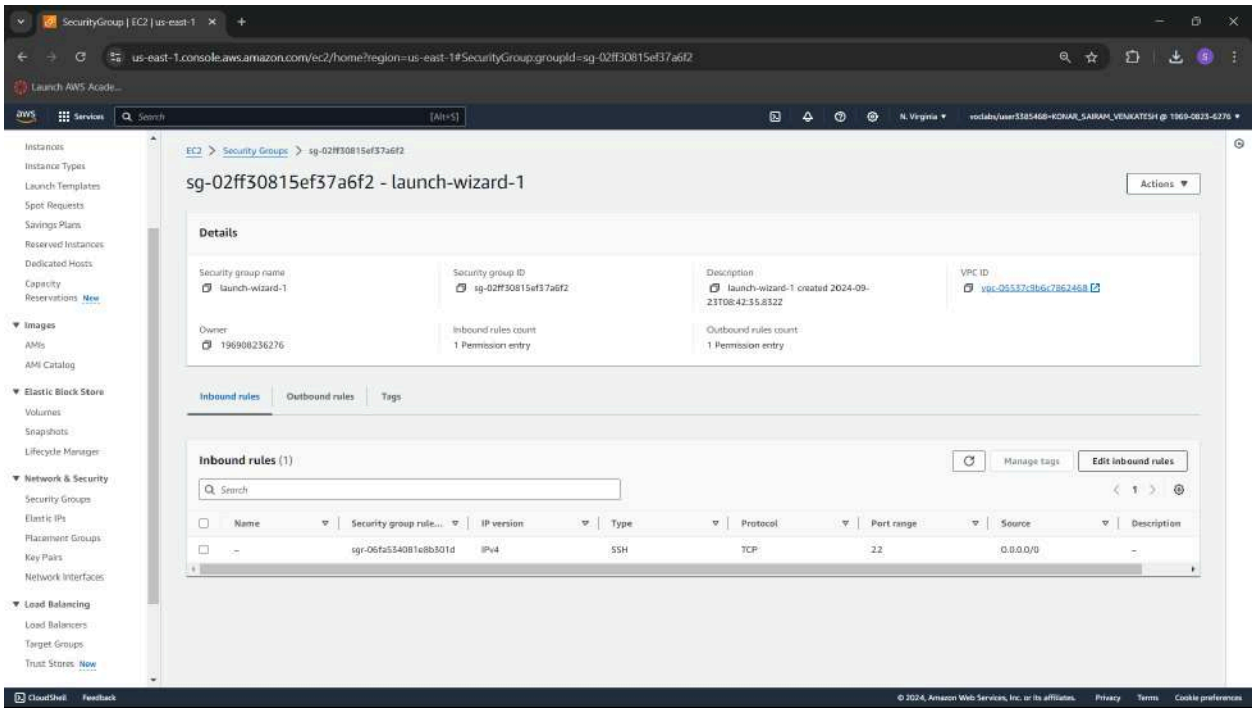
Step 2: If you do not have a private key created or a .pem file created, click on create a key pair. Else select the key pair that you had created before. (Make sure you know where the .pem file for that key is present on your system)



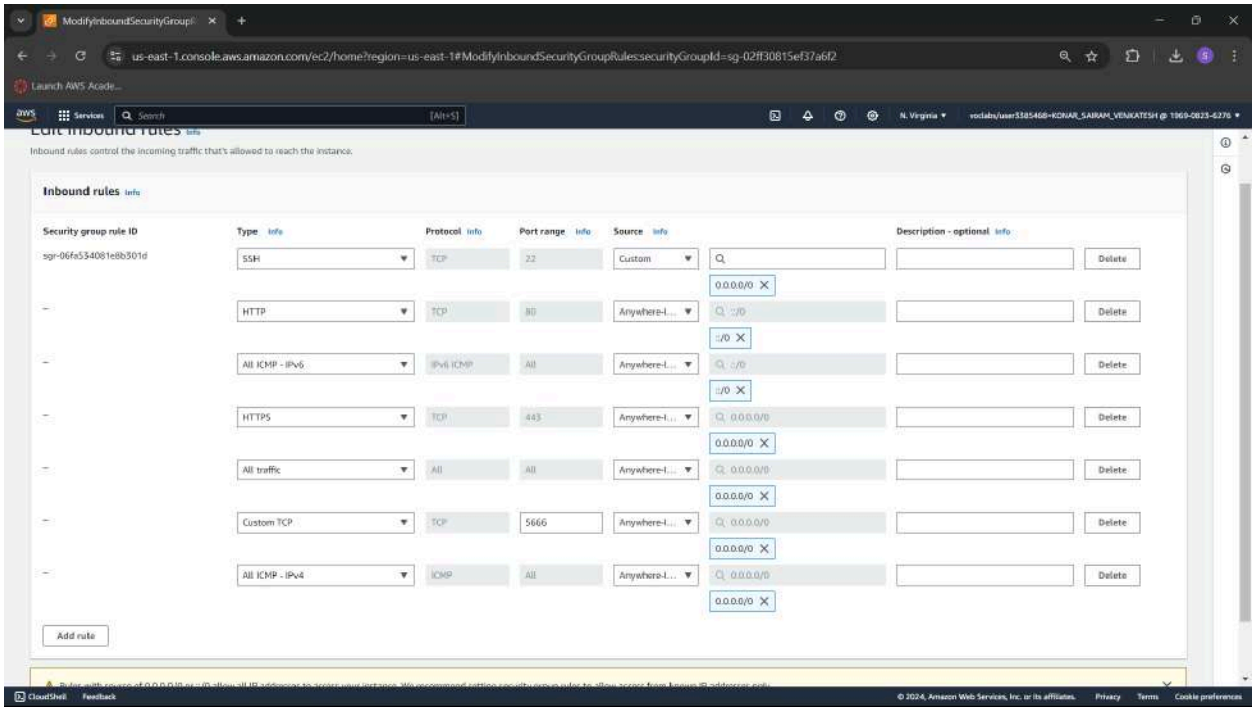
AWS will create a security group for this instance. Keep the name of that instance saved.
Step 3: After creating the instance, click on Security Groups from the left side pane. Find the security group that was created for your instance. Click on the instance ID for that group.



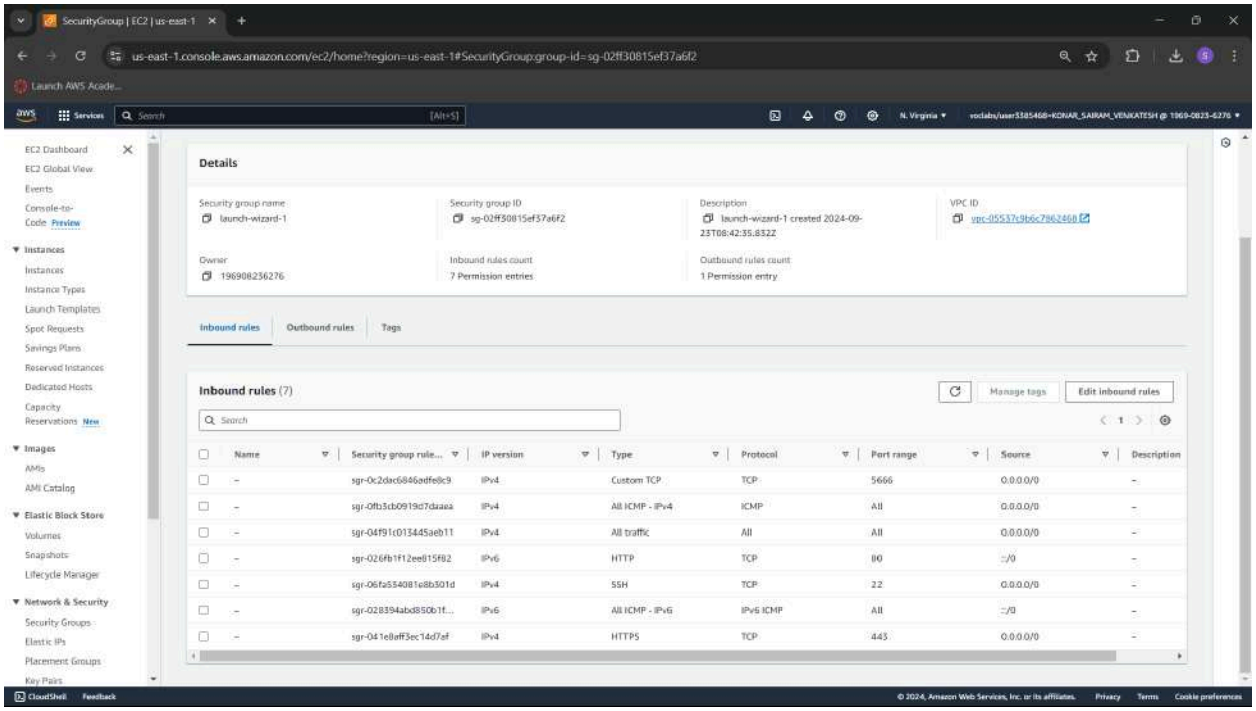
Here, click on Edit Inbound Rules.



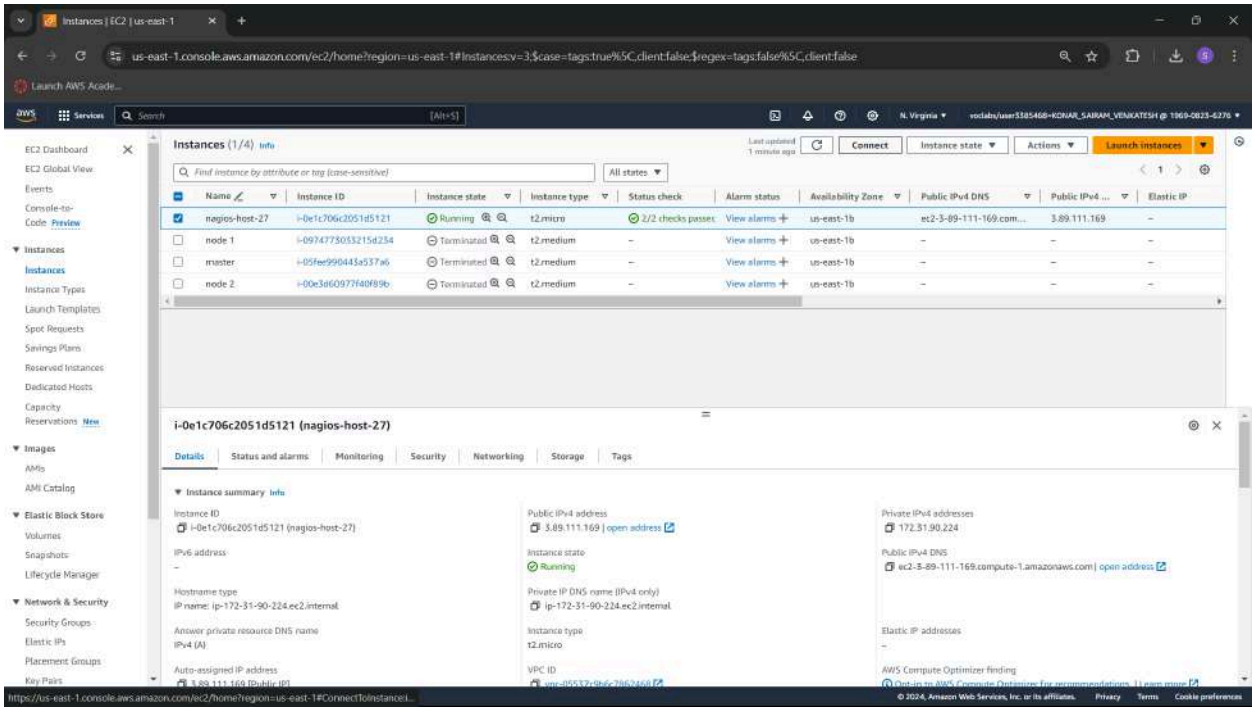
Now, click on add rules, and add teh rules for the following protocols:
HTTP, All ICMP - IPv6, HTTPS, All traffic, Custom TCP (Port 5666), All ICMP - IPv4



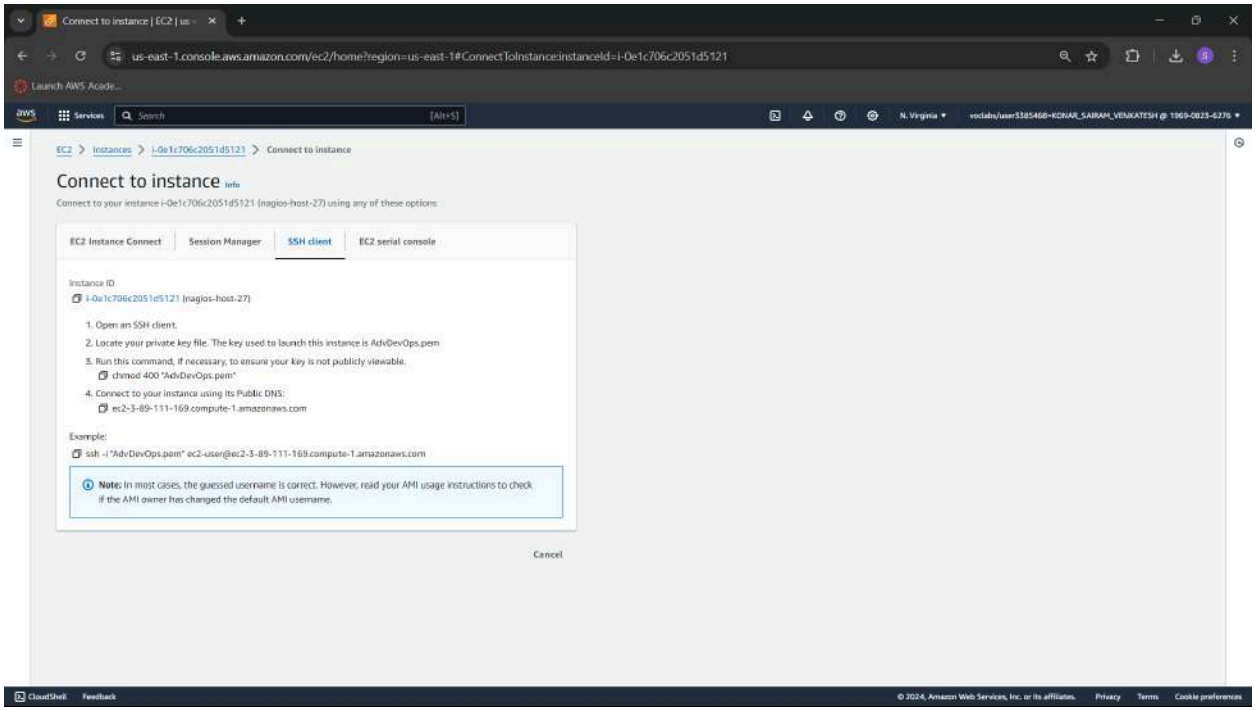
Click on save. This will add all the inbound rules to the security group.



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

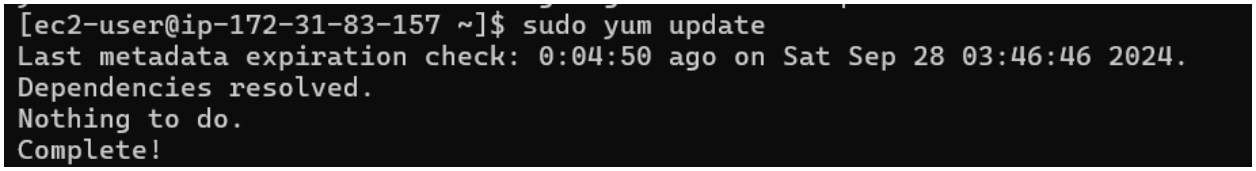


Click on SSH client. Copy the example command.

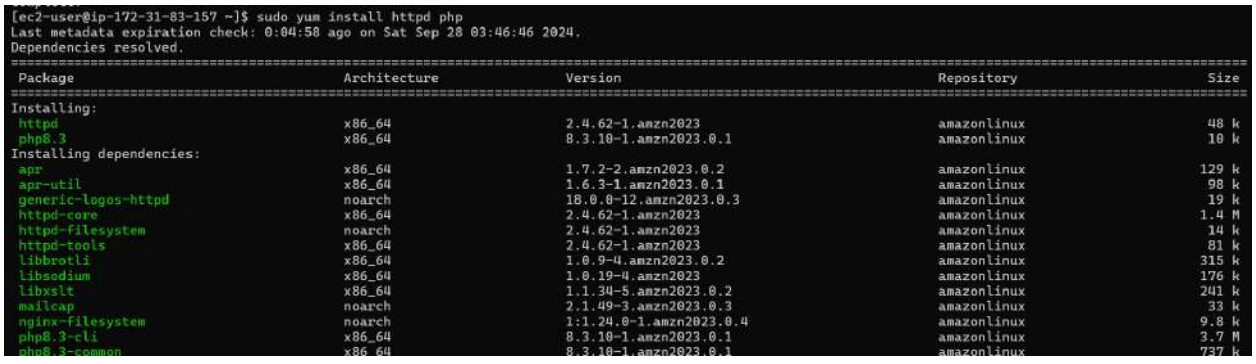


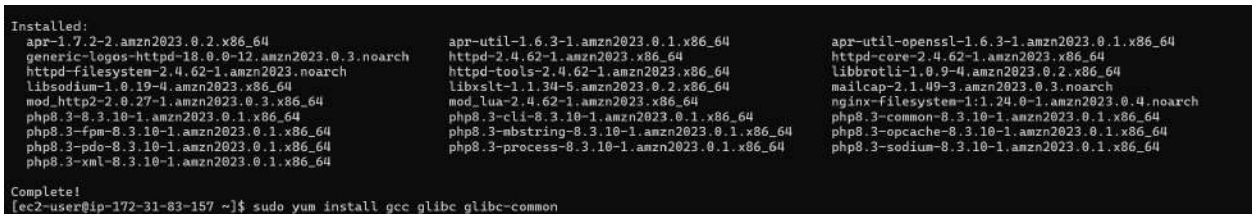
Step 5: Now, we have to connect our local OS terminal to the instance using SSH. For this,
Open terminal wher the private key file is located (.pem)
Paste the copied SSH command and run it.

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

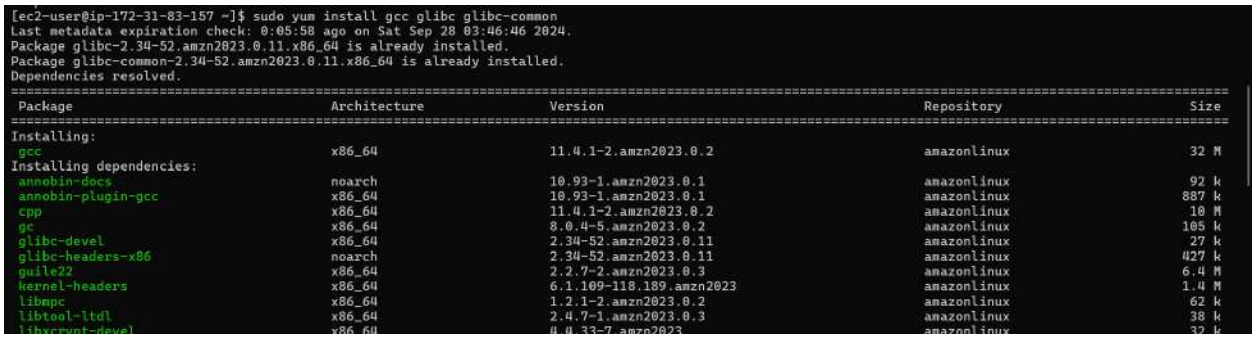


Step 7: We are going to install an Apache server and a PHP on this instance. For that, run this command.
sudo yum install httpd php





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

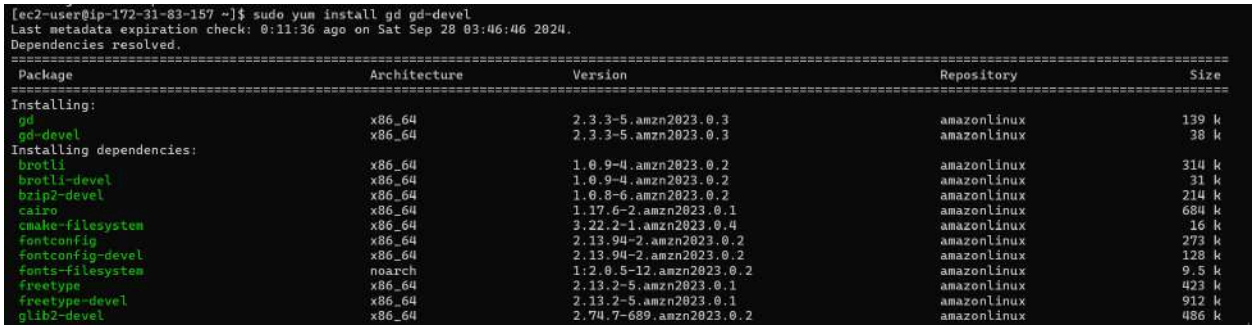


```
[ec2-user@ip-172-31-83-157 ~]$ sudo yum install gcc glibc glibc-common
Last metadata expiration check: 0:05:58 ago on Sat Sep 28 03:46:46 2024.
Package glibc-2.34-52.amzn2023.0.11.x86_64 is already installed.
Package glibc-common-2.34-52.amzn2023.0.11.x86_64 is already installed.
Dependencies resolved.
```

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

```
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!
[ec2-user@ip-172-31-83-157 ~]$
```

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



```
[ec2-user@ip-172-31-83-157 ~]$ sudo yum install gd gd-devel
Last metadata expiration check: 0:11:36 ago on Sat Sep 28 03:46:46 2024.
Dependencies resolved.
```

Package	Architecture	Version	Repository	Size
Installing:				
gd	x86_64	2.3.3-5.amzn2023.0.3	amazonlinux	139 k
gd-devel	x86_64	2.3.3-5.amzn2023.0.3	amazonlinux	38 k
Installing dependencies:				
brotli	x86_64	1.0.9-4.amzn2023.0.2	amazonlinux	314 k
brotli-devel	x86_64	1.0.9-4.amzn2023.0.2	amazonlinux	31 k
bzip2-devel	x86_64	1.0.8-6.amzn2023.0.2	amazonlinux	214 k
cairo	x86_64	1.17.6-2.amzn2023.0.1	amazonlinux	684 k
cmake-filesystem	x86_64	3.22.2-1.amzn2023.0.4	amazonlinux	16 k
fontconfig	x86_64	2.13.94-2.amzn2023.0.2	amazonlinux	273 k
fontconfig-devel	x86_64	2.13.94-2.amzn2023.0.2	amazonlinux	128 k
fonts-filesystem	noarch	1:2.0.5-12.amzn2023.0.2	amazonlinux	9.5 k
freetype	x86_64	2.13.2-5.amzn2023.0.1	amazonlinux	423 k
freetype-devel	x86_64	2.13.2-5.amzn2023.0.1	amazonlinux	912 k
glib2-devel	x86_64	2.74.7-689.amzn2023.0.2	amazonlinux	486 k

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

Complete!
[ec2-user@ip-172-31-83-157 ~]$
```

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

```
[ec2-user@ip-172-31-83-157 ~]$ sudo adduser -m nagios
sudo passwd nagios
Changing password for user nagios.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: all authentication tokens updated successfully.
```

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

```
[ec2-user@ip-172-31-83-157 ~]$ sudo groupadd nagcmd
[ec2-user@ip-172-31-83-157 ~]$
```

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

```
[ec2-user@ip-172-31-83-157 ~]$ sudo usermod -a -G nagcmd nagios
sudo usermod -a -G nagcmd apache
[ec2-user@ip-172-31-83-157 ~]$
```

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

```
[ec2-user@ip-172-31-83-157 ~]$ mkdir ~/downloads
cd ~/downloads
[ec2-user@ip-172-31-83-157 downloads]$
```

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

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

```
[ec2-user@ip-172-31-83-157 downloads]$ wget https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.5.tar.gz
--2024-09-28 04:04:23--  https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.5.tar.gz
Resolving assets.nagios.com (assets.nagios.com)... 45.79.49.120, 2600:3c00::f03c:92ff:fef7:45ce
Connecting to assets.nagios.com (assets.nagios.com)|45.79.49.120|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2065473 (2.0M) [application/x-gzip]
Saving to: 'nagios-4.5.5.tar.gz'

nagios-4.5.5.tar.gz                 100%[=============================================================================>]   1.97M  5.36MB/s    in 0.4s

2024-09-28 04:04:24 (5.36 MB/s) - 'nagios-4.5.5.tar.gz' saved [2065473/2065473]

[ec2-user@ip-172-31-83-157 downloads]$
```

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

```
[ec2-user@ip-172-31-83-157 downloads]$ wget https://nagios-plugins.org/download/nagios-plugins-2.4.11.tar.gz
--2024-09-28 04:06:15--  https://nagios-plugins.org/download/nagios-plugins-2.4.11.tar.gz
Resolving nagios-plugins.org (nagios-plugins.org)... 45.56.123.251
Connecting to nagios-plugins.org (nagios-plugins.org)|45.56.123.251|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2753049 (2.6M) [application/x-gzip]
Saving to: 'nagios-plugins-2.4.11.tar.gz'

nagios-plugins-2.4.11.tar.gz          100%[=============================================================================>]   2.62M  5.22MB/s    in 0.5s

2024-09-28 04:06:16 (5.22 MB/s) - 'nagios-plugins-2.4.11.tar.gz' saved [2753049/2753049]

[ec2-user@ip-172-31-83-157 downloads]$
```

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

```
[ec2-user@ip-172-31-83-157 downloads]$ tar zxvf nagios-4.5.5.tar.gz
nagios-4.5.5/
nagios-4.5.5/.github/
nagios-4.5.5/.github/workflows/
nagios-4.5.5/.github/workflows/test.yml
nagios-4.5.5/.gitignore
nagios-4.5.5/CONTRIBUTING.md
nagios-4.5.5/Changelog
nagios-4.5.5/INSTALLING
nagios-4.5.5/LEGAL
nagios-4.5.5/LICENSE
nagios-4.5.5/Makefile.in
nagios-4.5.5/README.md
nagios-4.5.5/THANKS
nagios-4.5.5/UPGRADING
nagios-4.5.5/aclocal.m4
nagios-4.5.5/autoconf-macros/
nagios-4.5.5/autoconf-macros/.gitignore
nagios-4.5.5/autoconf-macros/CHANGELOG.md
nagios-4.5.5/autoconf-macros/LICENSE
nagios-4.5.5/autoconf-macros/LICENSE.md
```

```
nagios-4.5.5/xdata/.gitignore
nagios-4.5.5/xdata/Makefile.in
nagios-4.5.5/xdata/xcddefault.c
nagios-4.5.5/xdata/xcddefault.h
nagios-4.5.5/xdata/xodtemplate.c
nagios-4.5.5/xdata/xodtemplate.h
nagios-4.5.5/xdata/xpddefault.c
nagios-4.5.5/xdata/xpddefault.h
nagios-4.5.5/xdata/xrddefault.c
nagios-4.5.5/xdata/xrddefault.h
nagios-4.5.5/xdata/xsddefault.c
nagios-4.5.5/xdata/xsddefault.h
[ec2-user@ip-172-31-83-157 downloads]$
```

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

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

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

Steps: ls

```
[ec2-user@ip-172-31-83-157 downloads]$ ls
nagios-4.5.5  nagios-4.5.5.tar.gz  nagios-plugins-2.4.11.tar.gz
```

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

Another error could be Cannot find SSL headers.

To solve this, we need to install OpenSSL Dev Library

Steps:
sudo yum install openssl-devel

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

```
[ec2-user@ip-172-31-83-157 nagios-4.5.5]$ ./configure --with-command-group=nagcmd
checking for a BSD-compatible install... /usr/bin/install -c
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether the compiler supports GNU C... yes
checking whether gcc accepts -g... yes
checking for gcc option to enable C11 features... none needed
checking whether make sets $(MAKE)... yes
checking whether ln -s works... yes
checking for strip... /usr/bin/strip
checking for sys/wait.h that is POSIX.1 compatible... yes
checking for stdio.h... yes
checking for stdlib.h... yes
checking for string.h... yes
checking for inttypes.h... yes
checking for stdint.h... yes
checking for strings.h... yes
checking for sys/stat.h... yes
```

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

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

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


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

[ec2-user@ip-172-31-83-157 nagios-4.5.5]$
```

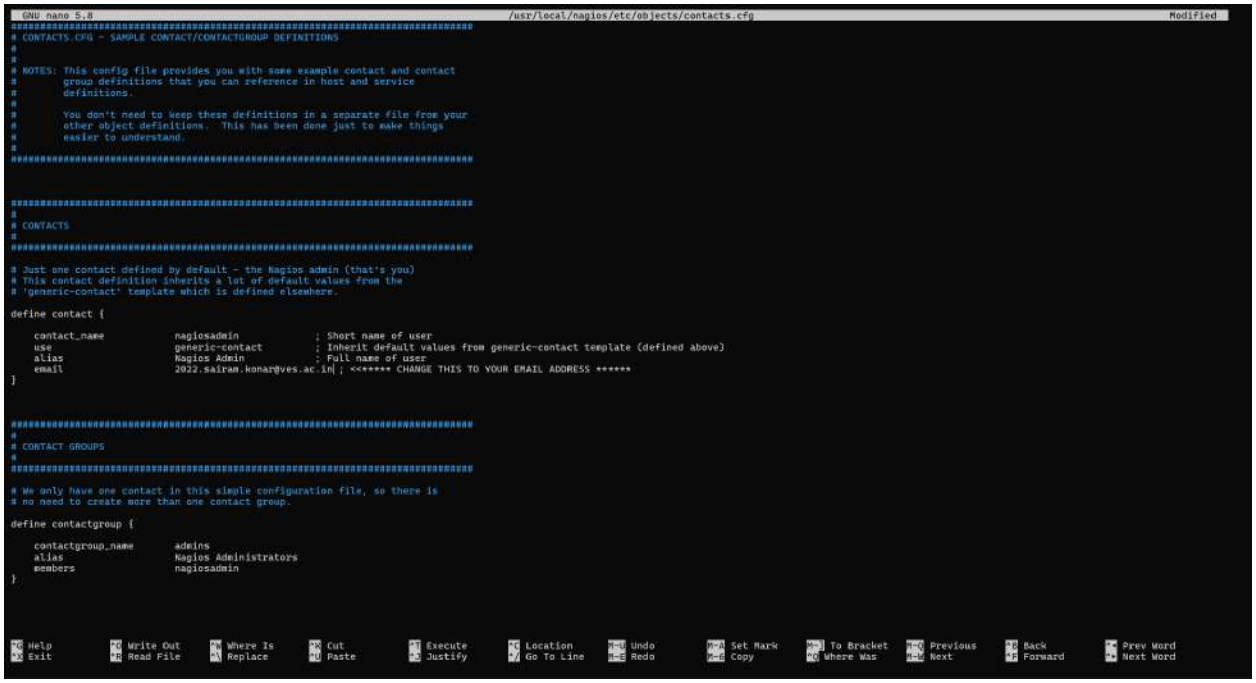
Step 17: We need to compile all components of this software according to the instruction in the Makefile. For that, use this command:
make all

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

```
[ec2-user@ip-172-31-83-157 nagios-4.5.5]$ sudo make install
sudo make install-init
sudo make install-config
sudo make install-commandmode
cd ./base && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/base'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagios /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagiostats /usr/local/nagios/bin
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5/base'
cd ./cgi && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
make install-basic
make[2]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/sbin
```

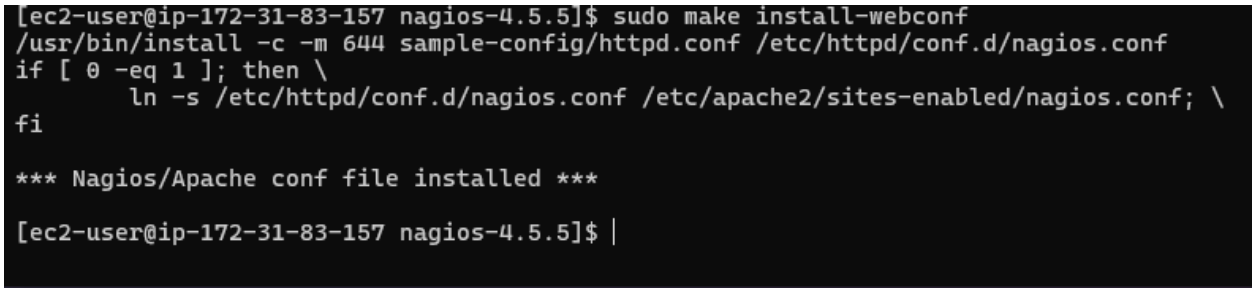
```
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5'
/usr/bin/install -c -m 755 -d -o root -g root /lib/systemd/system
/usr/bin/install -c -m 755 -o root -g root startup/default-service /lib/systemd/system/nagios.service
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc/objects
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/nagios.cfg /usr/local/nagios/etc/nagios.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/cgi.cfg /usr/local/nagios/etc/cgi.cfg
/usr/bin/install -c -b -m 660 -o nagios -g nagios sample-config/resource.cfg /usr/local/nagios/etc/resource.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/templates.cfg /usr/local/nagios/etc/objects/templates.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/commands.cfg /usr/local/nagios/etc/objects/commands.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/contacts.cfg /usr/local/nagios/etc/objects/contacts.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/timeperiods.cfg /usr/local/nagios/etc/objects/timeperiods.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/localhost.cfg /usr/local/nagios/etc/objects/localhost.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/windows.cfg /usr/local/nagios/etc/objects/windows.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/printer.cfg /usr/local/nagios/etc/objects/printer.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/switch.cfg /usr/local/nagios/etc/objects/switch.cfg

*** Config files installed ***

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

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

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

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



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

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



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

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

```
[ec2-user@ip-172-31-83-157 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
```

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

```
[ec2-user@ip-172-31-83-157 nagios-4.5.5]$ sudo service httpd restart
Redirecting to /bin/systemctl restart httpd.service
```

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

```
[ec2-user@ip-172-31-83-157 downloads]$ tar zxvf nagios-plugins-2.4.11.tar.gz
nagios-plugins-2.4.11/
nagios-plugins-2.4.11/build-aux/
nagios-plugins-2.4.11/build-aux/compile
nagios-plugins-2.4.11/build-aux/config.guess
nagios-plugins-2.4.11/build-aux/config.rpath
nagios-plugins-2.4.11/build-aux/config.sub
nagios-plugins-2.4.11/build-aux/install-sh
nagios-plugins-2.4.11/build-aux/ltmain.sh
nagios-plugins-2.4.11/build-aux/missing
nagios-plugins-2.4.11/build-aux/mkinstalldirs
nagios-plugins-2.4.11/build-aux/depcomp
nagios-plugins-2.4.11/build-aux/snippet/
nagios-plugins-2.4.11/build-aux/snippet/_Noreturn.h
nagios-plugins-2.4.11/build-aux/snippet/arg-nonnull.h
nagios-plugins-2.4.11/build-aux/snippet/c++defs.h
nagios-plugins-2.4.11/build-aux/snippet/warn-on-use.h
nagios-plugins-2.4.11/build-aux/test-driver
nagios-plugins-2.4.11/config_test/
nagios-plugins-2.4.11/config_test/Makefile
nagios-plugins-2.4.11/config_test/run_tests
nagios-plugins-2.4.11/config_test/child_test.c
nagios-plugins-2.4.11/gl/
nagios-plugins-2.4.11/gl/m4/
```

```
nagios-plugins-2.4.11/po/fr.gmo
nagios-plugins-2.4.11/po/de.gmo
nagios-plugins-2.4.11/po/nagios-plugins.pot
nagios-plugins-2.4.11/po/stamp-po
nagios-plugins-2.4.11/po/ChangeLog
nagios-plugins-2.4.11/po/LINGUAS
nagios-plugins-2.4.11/release
[ec2-user@ip-172-31-83-157 downloads]$
```

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

```
[ec2-user@ip-172-31-83-157 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
```

```
config.status: creating plugins-scripts/utils.sh
config.status: creating perlmods/Makefile
config.status: creating test.pl
config.status: creating pkg/solaris/pkginfo
config.status: creating po/Makefile.in
config.status: creating config.h
config.status: config.h is unchanged
config.status: executing depfiles commands
config.status: executing libtool commands
config.status: executing po-directories commands
config.status: creating po/POTFILES
config.status: creating po/Makefile
[ec2-user@ip-172-31-83-157 nagios-plugins-2.4.11]$
```

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

```
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-plugins-2.4.11/po'
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-172-31-83-157 nagios-plugins-2.4.11]$
```

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

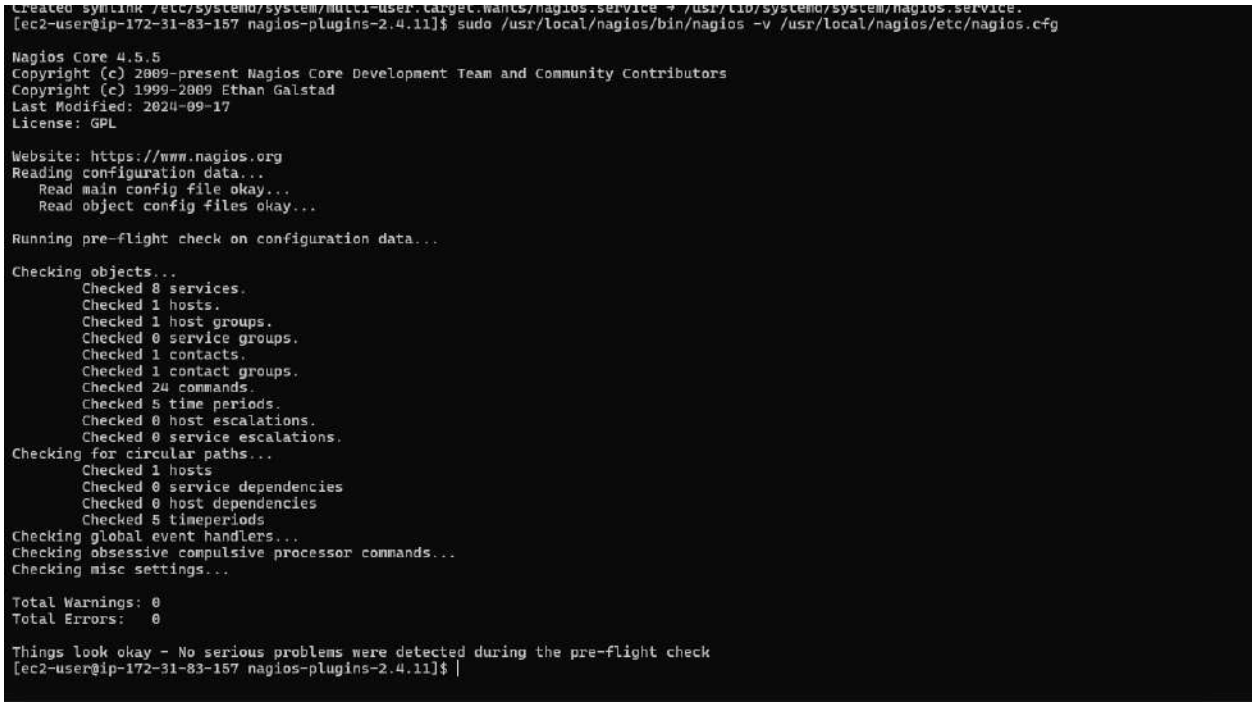
Step 26: We need to verify the Nagios configuration for any syntax errors or issues before starting or restarting the Nagios service.
sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

Error: Error in configuration file '/usr/local/nagios/etc/nagios.cfg' - Line 452 (Check result path '/usr/local/nagios/var/spool/checkresults' is not a valid directory)
It is an error in processing main config file!

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

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

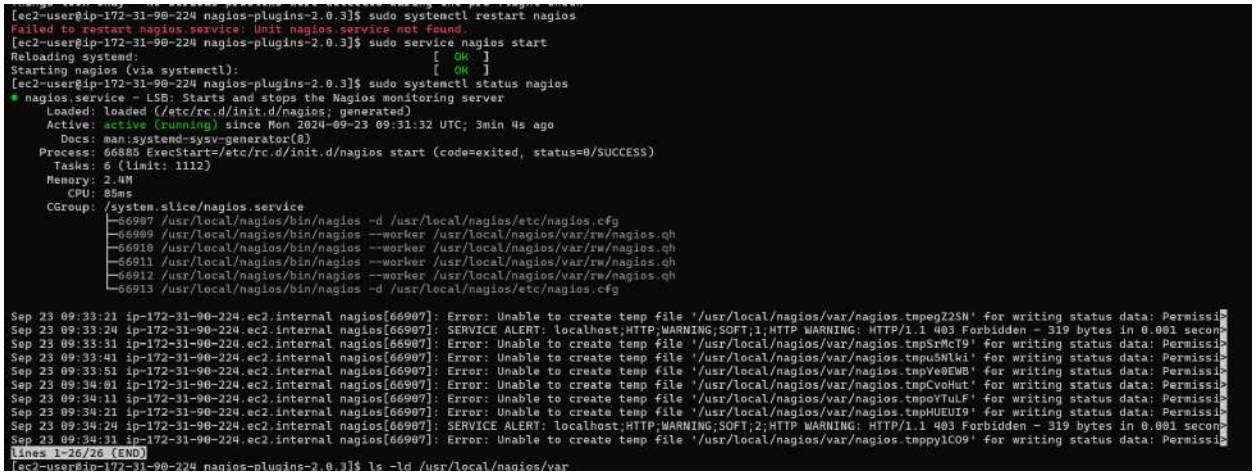
Now rerun the commmad
sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg



Step 27:
sudo service nagios start

```
[ec2-user@ip-172-31-83-157 nagios-plugins-2.4.11]$ sudo service nagios start
Redirecting to /bin/systemctl start nagios.service
[ec2-user@ip-172-31-83-157 nagios-plugins-2.4.11]$
```

sudo systemctl status nagios



Error:Sep 23 09:34:31 ip-172-31-90-224.ec2.internal nagios[66907]: Error: Unable to create temp file '/usr/local/nagios/var/nagios.tmppy1CO9' for writing status data: Permissi>

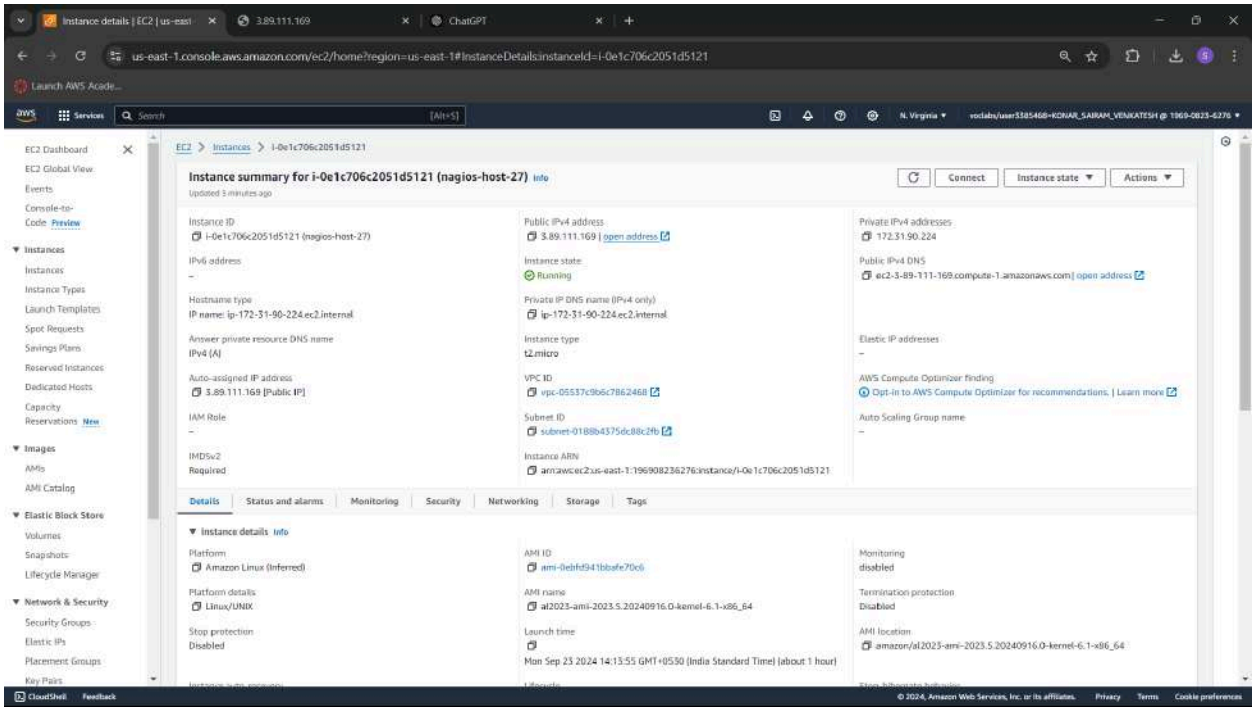
Solution:

- ls -ld /usr/local/nagios/var
- sudo chown -R nagios:nagios /usr/local/nagios/var
- sudo chmod -R 755 /usr/local/nagios/var
- sudo systemctl restart nagios

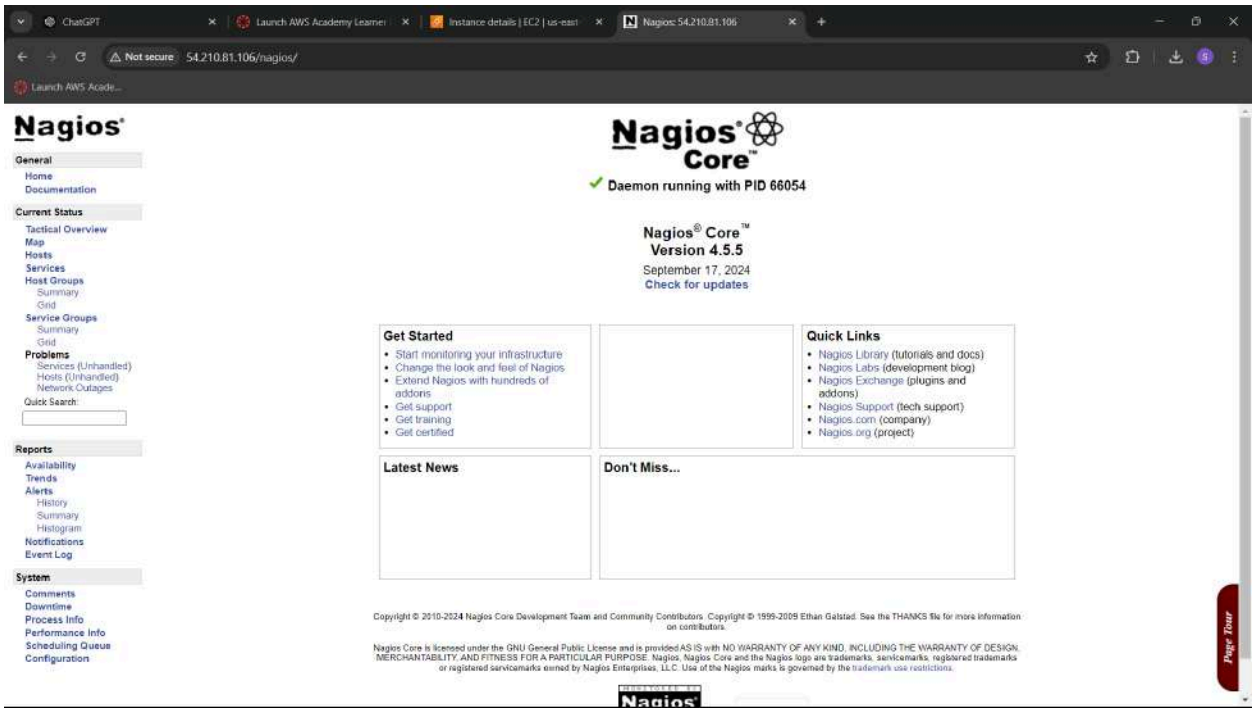
Rerun the command
sudo systemctl status nagios

```
[ec2-user@ip-172-31-83-157 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 Sat 2024-09-28 04:18:59 UTC; 12s ago
       Docs: https://www.nagios.org/documentation
    Process: 66052 ExecStartPre=/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
    Process: 66053 ExecStart=/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
   Main PID: 66054 (nagios)
      Tasks: 6 (limit: 1112)
     Memory: 5.6M
        CPU: 77ms
     CGroup: /system.slice/nagios.service
             ├─66054 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─66055 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─66056 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─66057 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─66058 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             └─66059 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg

Sep 28 04:18:59 ip-172-31-83-157.ec2.internal nagios[66054]: qh: Socket '/usr/local/nagios/var/rw/nagios.qh' successfully initialized
Sep 28 04:18:59 ip-172-31-83-157.ec2.internal nagios[66054]: qh: core query handler registered
Sep 28 04:18:59 ip-172-31-83-157.ec2.internal nagios[66054]: qh: echo service query handler registered
Sep 28 04:18:59 ip-172-31-83-157.ec2.internal nagios[66054]: qh: help for the query handler registered
Sep 28 04:18:59 ip-172-31-83-157.ec2.internal nagios[66054]: wproc: Successfully registered manager as @wproc with query handler
Sep 28 04:18:59 ip-172-31-83-157.ec2.internal nagios[66054]: wproc: Registry request: name=Core Worker 66057;pid=66057
Sep 28 04:18:59 ip-172-31-83-157.ec2.internal nagios[66054]: wproc: Registry request: name=Core Worker 66058;pid=66058
Sep 28 04:18:59 ip-172-31-83-157.ec2.internal nagios[66054]: wproc: Registry request: name=Core Worker 66056;pid=66056
Sep 28 04:18:59 ip-172-31-83-157.ec2.internal nagios[66054]: wproc: Registry request: name=Core Worker 66055;pid=66055
Sep 28 04:18:59 ip-172-31-83-157.ec2.internal nagios[66054]: Successfully launched command file worker with pid 66059
[ec2-user@ip-172-31-83-157 nagios-plugins-2.4.11]$
```

Step 28: Now, go to EC2 instance and click on instance id. Then, click on the copy icon just before the public ip address on public IP.



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



Conclusion:

In this experiment, we have learned how to install and configure Nagios Core, Nagios Plugins and NRPE on a Linux machine. We are using an Amazon Linux OS instance configured with the need security rules. We need to make sure that the Nagios-core and Nagios-plugins links that are used are the ones which are up-to dat (wget commands).It is needed to extract and configure these files so that no issues are detected while starting the server. Once all the setup is complete, we can start the nagios server. Using the public IP address of the EC2 instance, we can access the nagios dashboard by navigating that IP to nagios.

Sairam Konar D15C 27 2024-25

Experiment 10

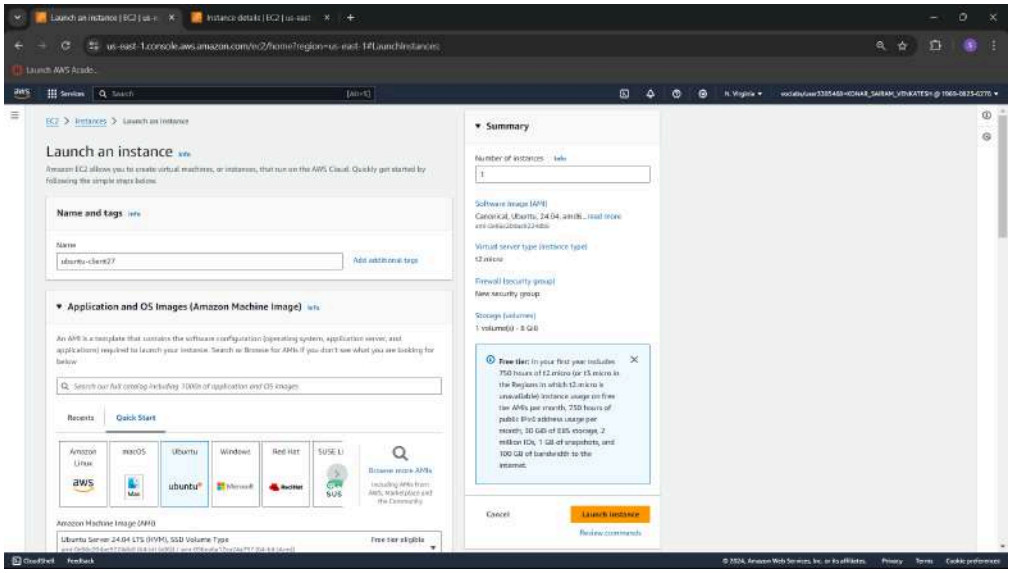
Aim: To perform Port, Service monitoring, Windows/Linux server monitoring using Nagios.

Prerequisites:

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

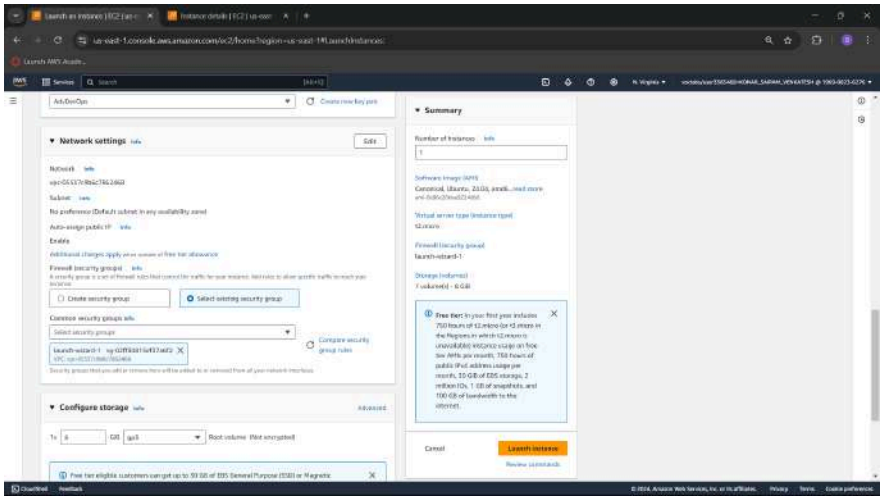
Step 1: Set up ubuntu instance

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



Select The OS Image as Ubuntu.

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



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

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

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

```
[ec2-user@ip-172-31-83-157 ~]$ ps -ef | grep nagios
nagios     66054       1  0 04:18 ?        00:00:00 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
nagios     66055   66054  0 04:18 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios     66056   66054  0 04:18 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios     66057   66054  0 04:18 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios     66058   66054  0 04:18 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios     66059   66054  0 04:18 ?        00:00:00 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
ec2-user   66758   66657  0 04:29 pts/0    00:00:00 grep --color=auto nagios
[ec2-user@ip-172-31-83-157 ~]$
```

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

```
[ec2-user@ip-172-31-83-157 ~]$ sudo su
mkdir /usr/local/nagios/etc/objects/monitorhosts
mkdir /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
[root@ip-172-31-83-157 ec2-user]#
```

3) We need to create a config file in this folder. So, copy the contents of the existing localhost config to the new file 'linuxserver.cfg'.

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

4) We need to make some changes in this config file. Open it using nano editor.
nano /usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg

Change **hostname** and **alias** to **linuxserver**
Change address to **public ip address of client instance** (Ubuntu instance)

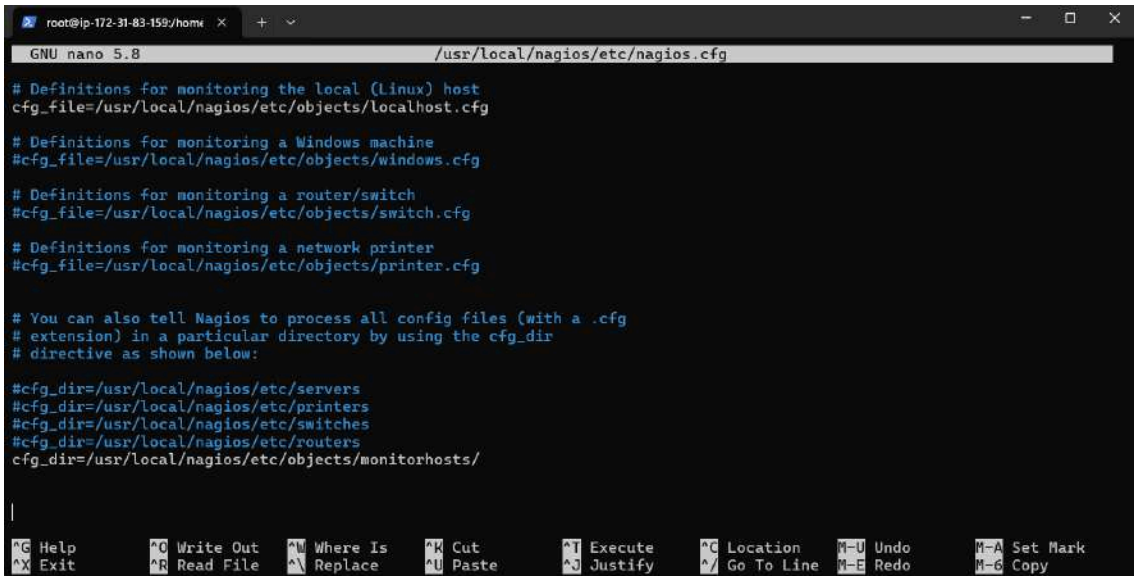
```
# Define a host for the local machine

define host{
        use                     linux-server            ; Name of host temp>
                                                        ; This host definit>
                                                        ; in (or inherited >
        host_name               linuxserver
        alias                   linuxserver
        address                 32.226.136.73
        }
```

Change hostgroup_name to **linux-servers1**

```
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 >
        }
```

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

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



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

```
[root@ip-172-31-83-157 ec2-user]# /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

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

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

Running pre-flight check on configuration data...

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

Total Warnings: 0
Total Errors:   0

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

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

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

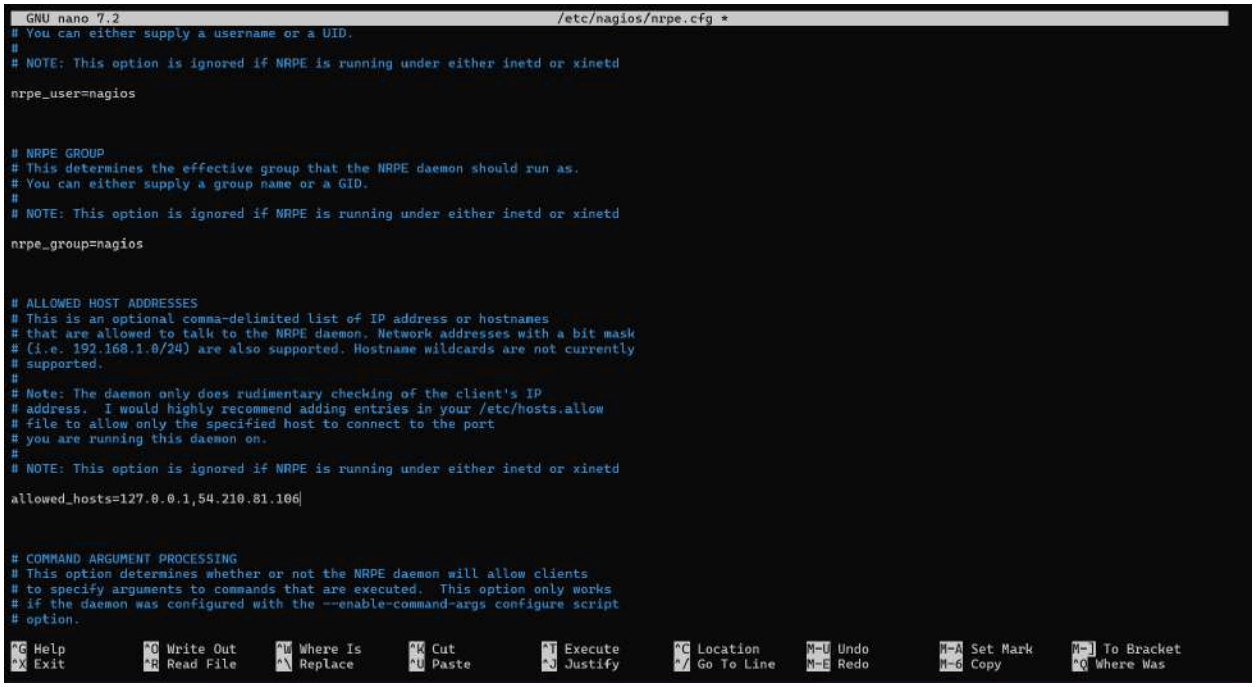
Step 3: Execute the following on Nagios Client machine (Ubuntu)

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

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

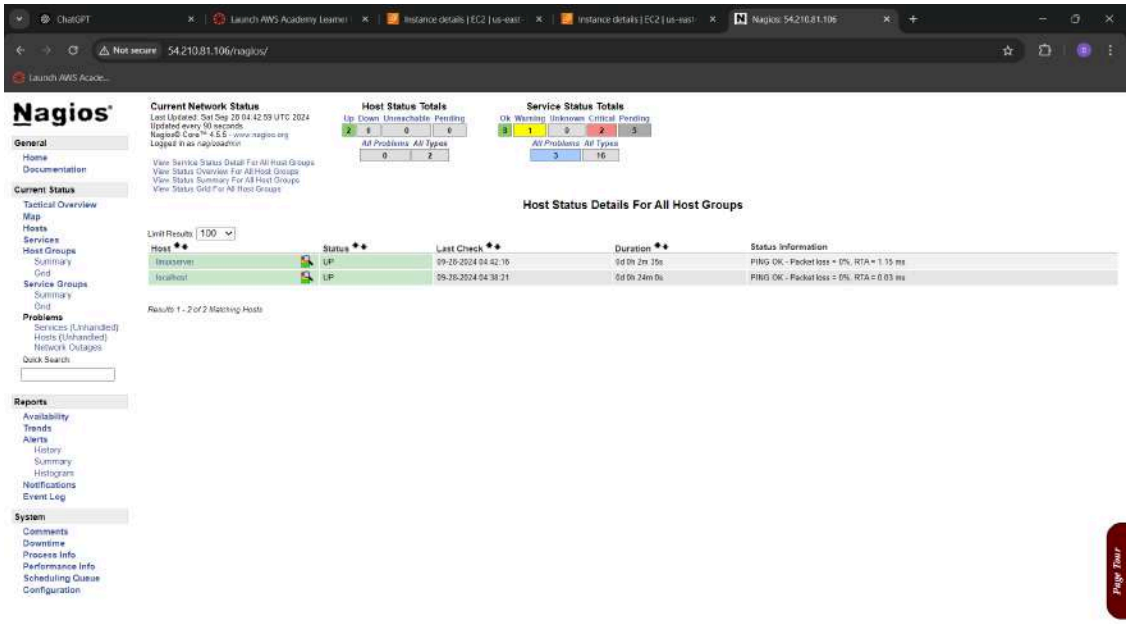
```
Running kernel seems to be up-to-date.

Restarting services...

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

No containers need to be restarted.

User sessions running outdated binaries:
 ubuntu @ session #4: sshd[1495,1569]
 ubuntu @ user manager service: systemd[1500]

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

2) We need to add the public IP address of our host Nagios machine (Linux) to the nrpe configuration file.
   sudo nano /etc/nagios/nrpe.cfg
   Under allowed_hosts, add the nagios host ip address (public)

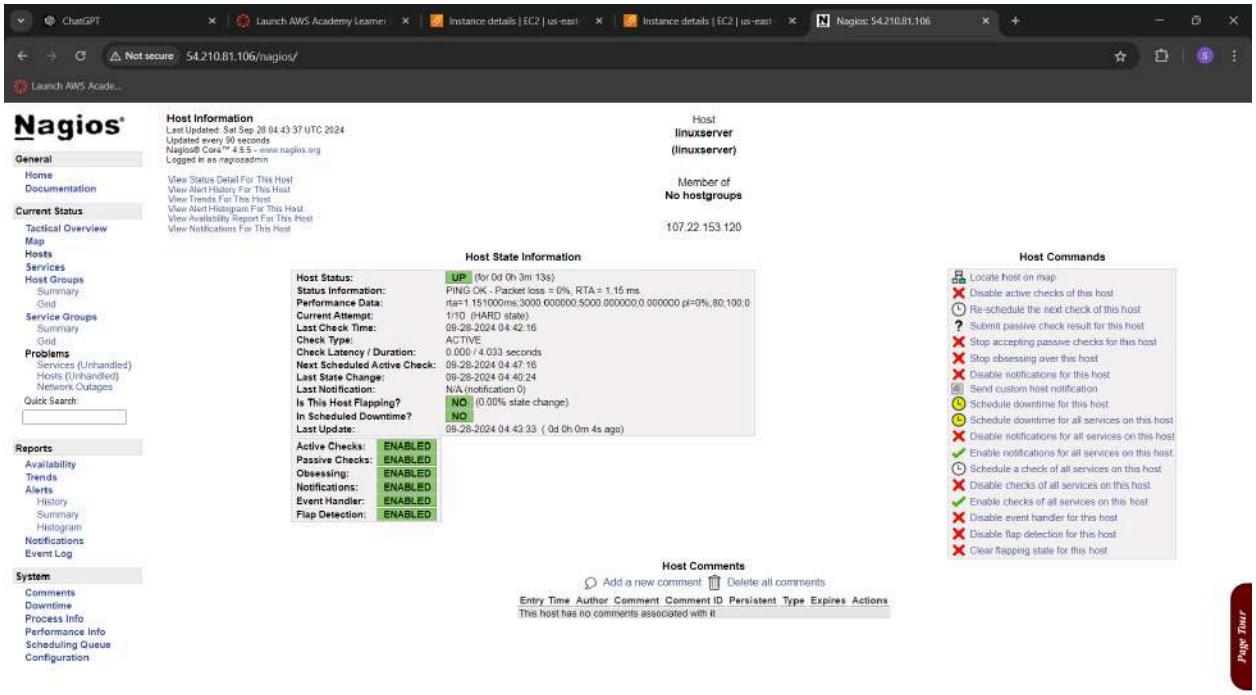


Step 4: Check the Nagios Dashboard

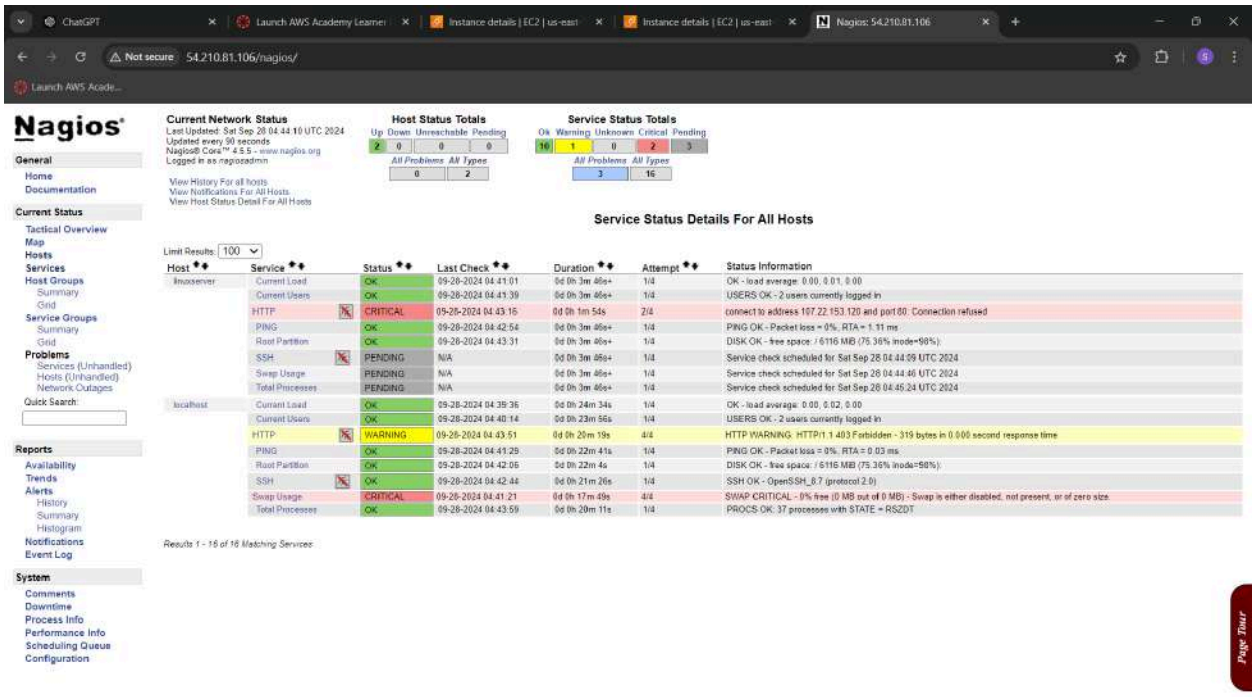
1) Go to Nagios dashboard, click on hosts.
   Here, we can see that the linuxserver is also added as a host.



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



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



In this case, we have monitored -
Servers: 1 linux server
Services: swap
Ports: 22, 80 (ssh, http)
Processes: User status, Current load, total processes, root partition, etc.

Conclusion:

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

Sairam Konar D15C 27 2024-25

Experiment 11

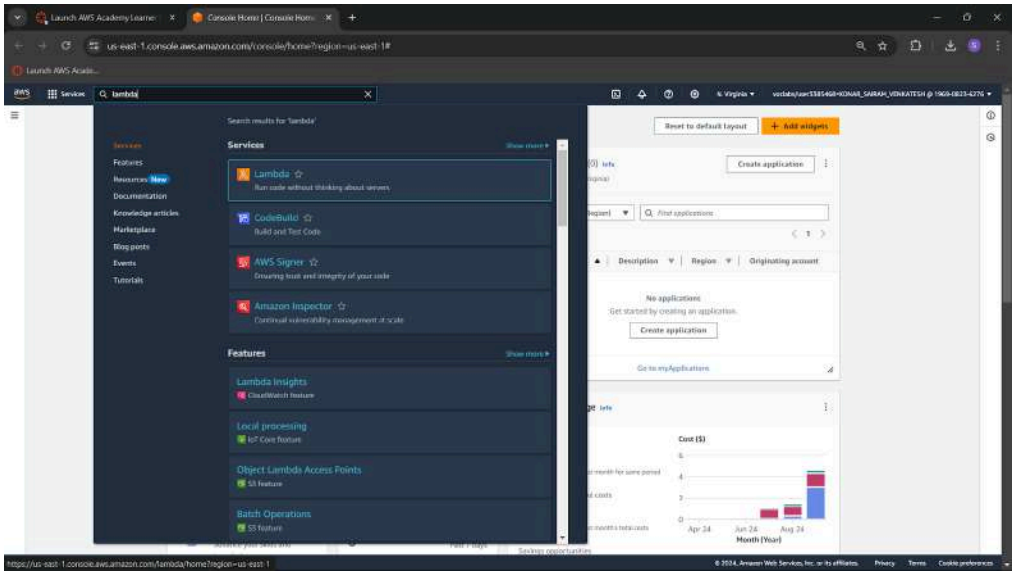
Aim: To understand AWS Lambda, its workflow, various functions and create your first Lambda functions using Python / Java / Nodejs.

Prerequisites:

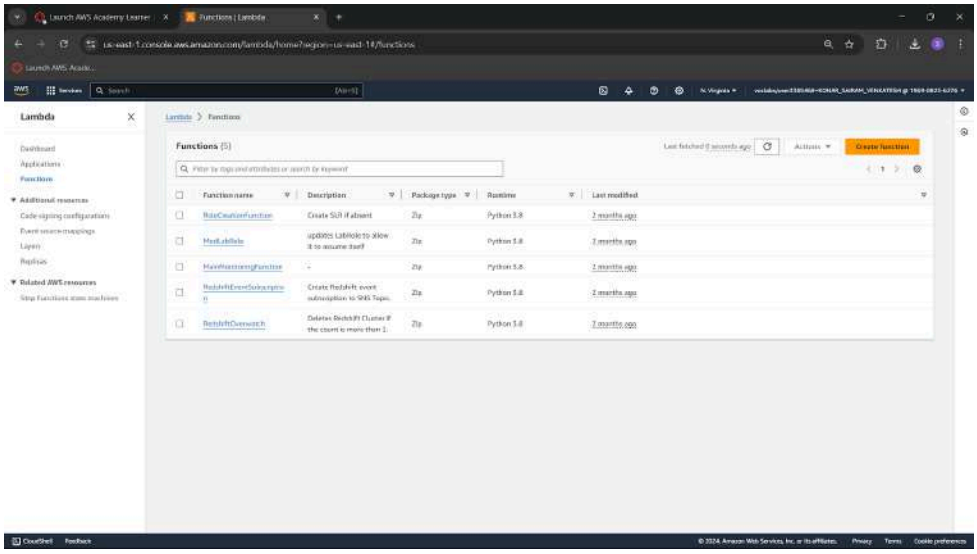
1) AWS account (academy recommended)

Step 1: Set up AWS Lambda Function

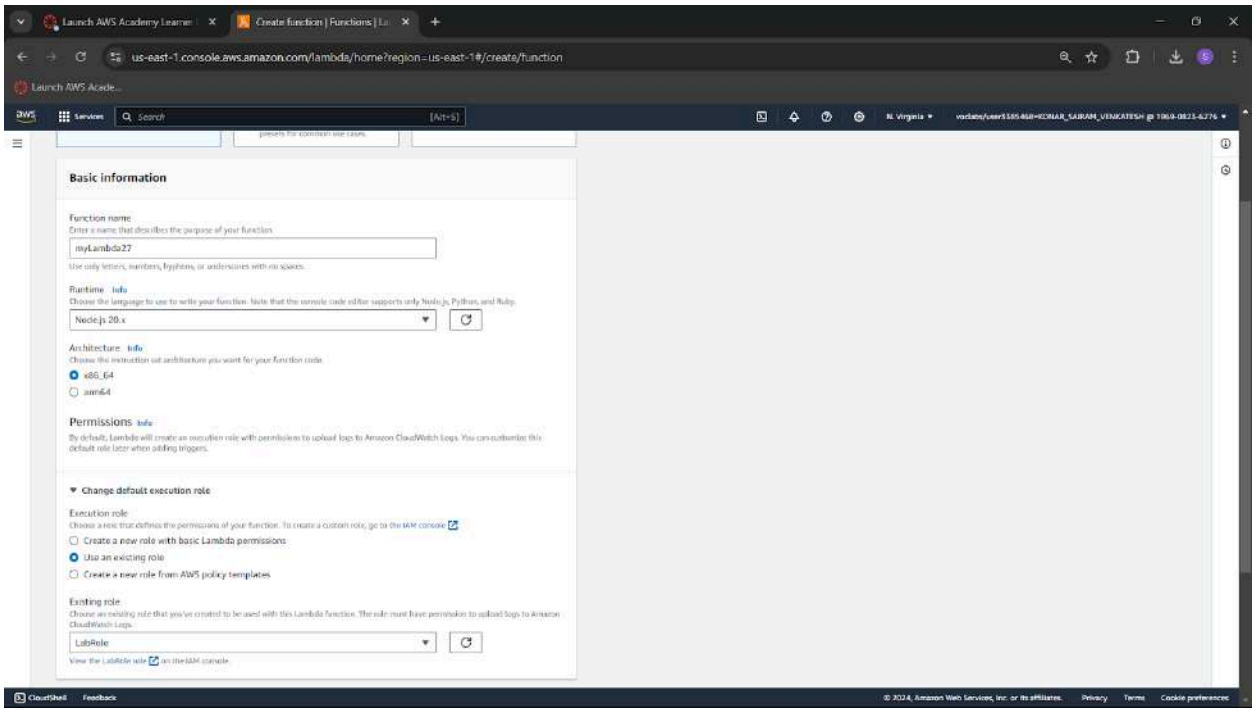
1) Search for Lambda in the services tab. Click on it once found.



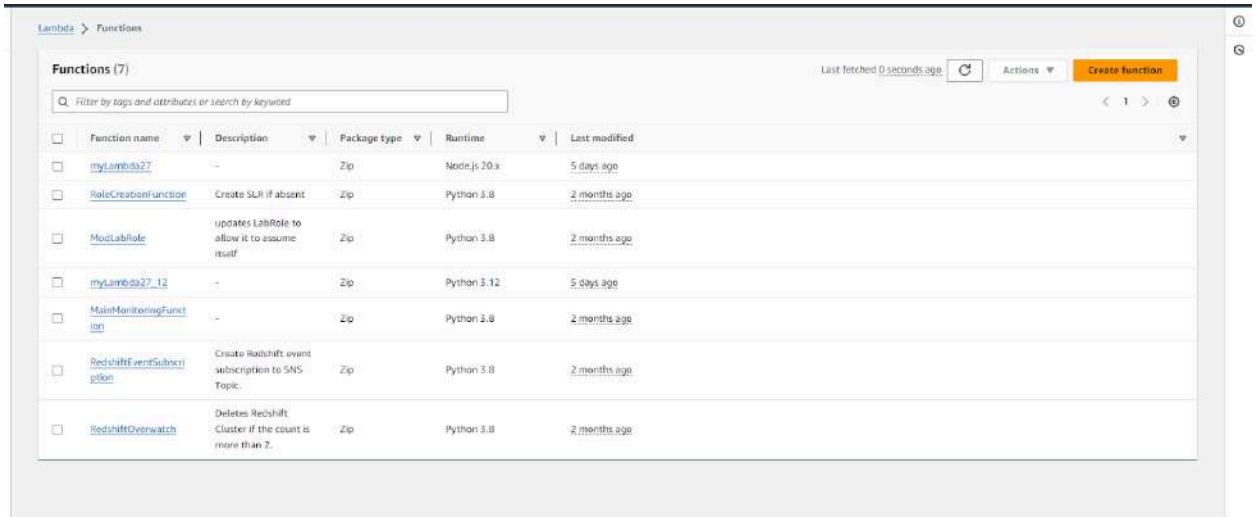
2) Click on create functions.



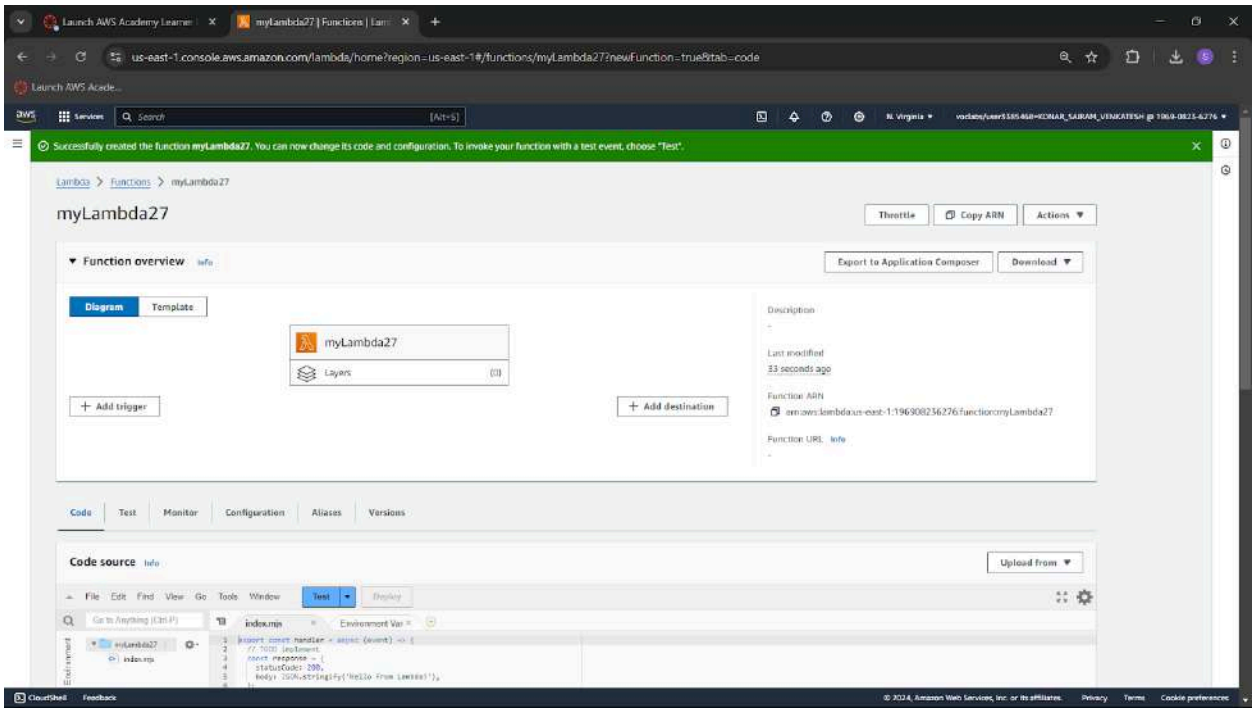
3) Give a name to your Lambda function. Select the runtime as Node.js 20.x (You can also use python). Select the architecture as x86_64. Set the default execution role as LabRole if you are doing this on your academy account. (Use an existing role → LabRole)



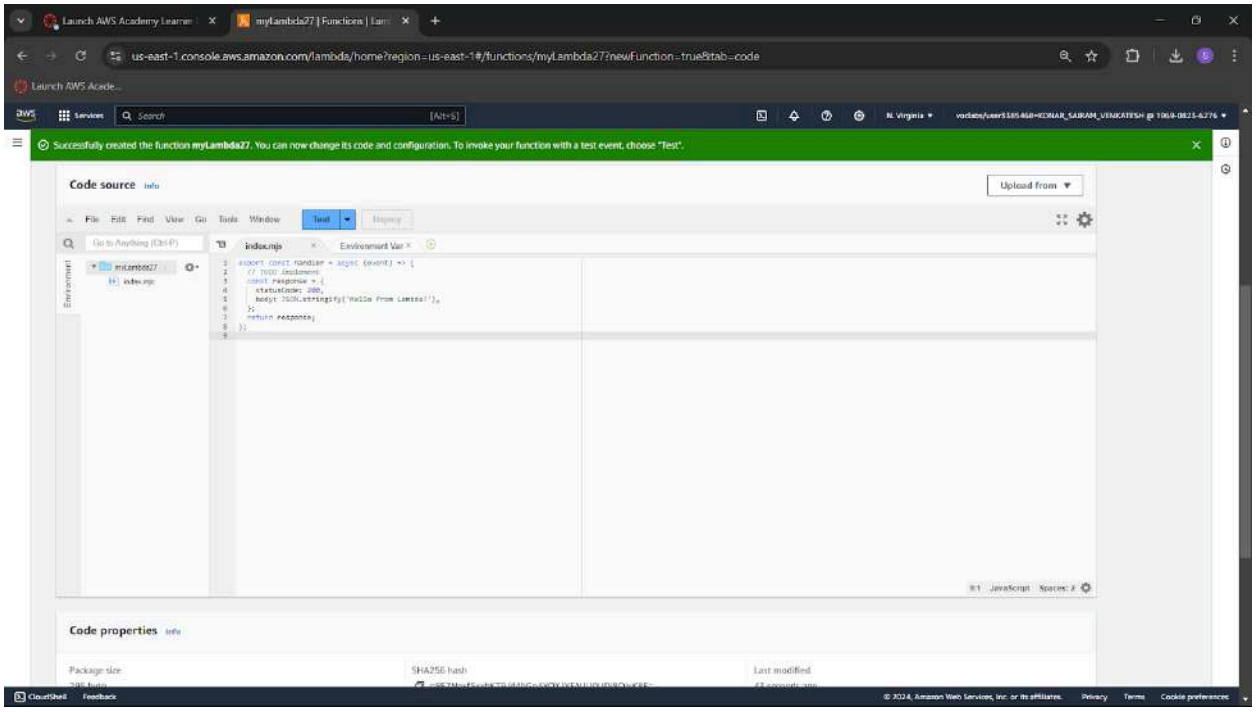
4) Once the function is created, click on the name of the function (myLambda27 in my case).



5) This is the dashboard of our lambda function.

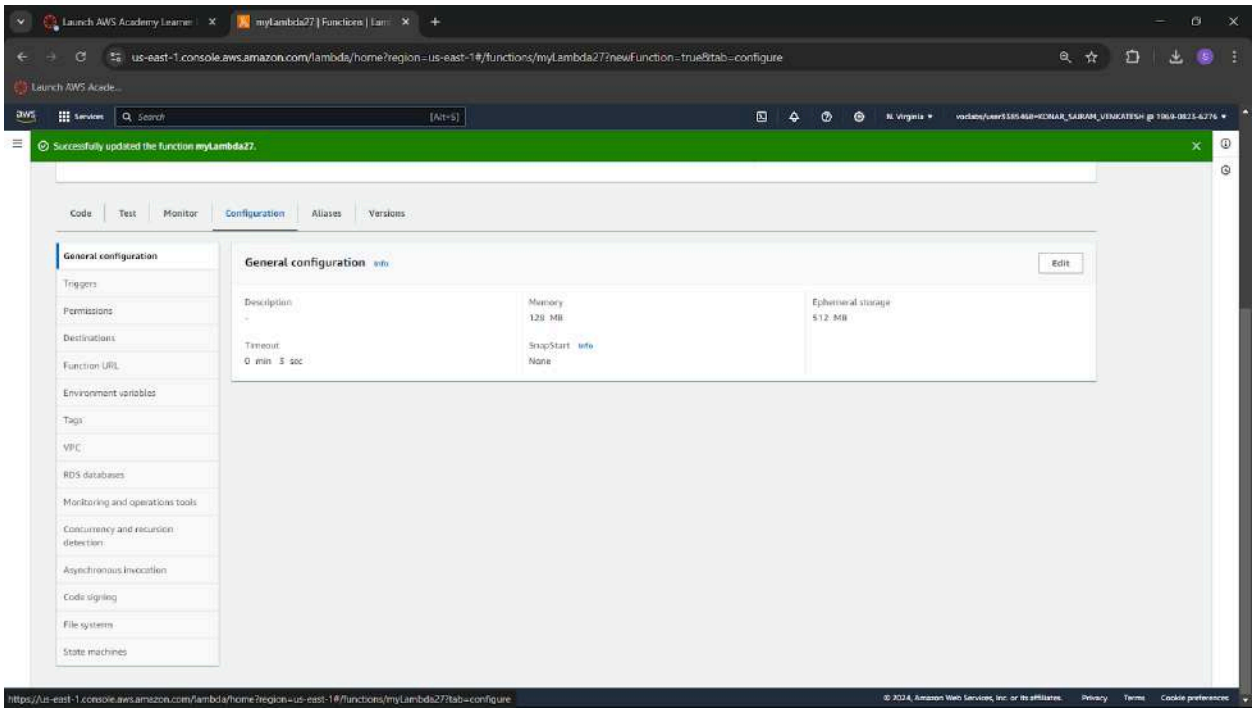


6) This function has the following default code, which is used to print “Hello form Lambda!”.

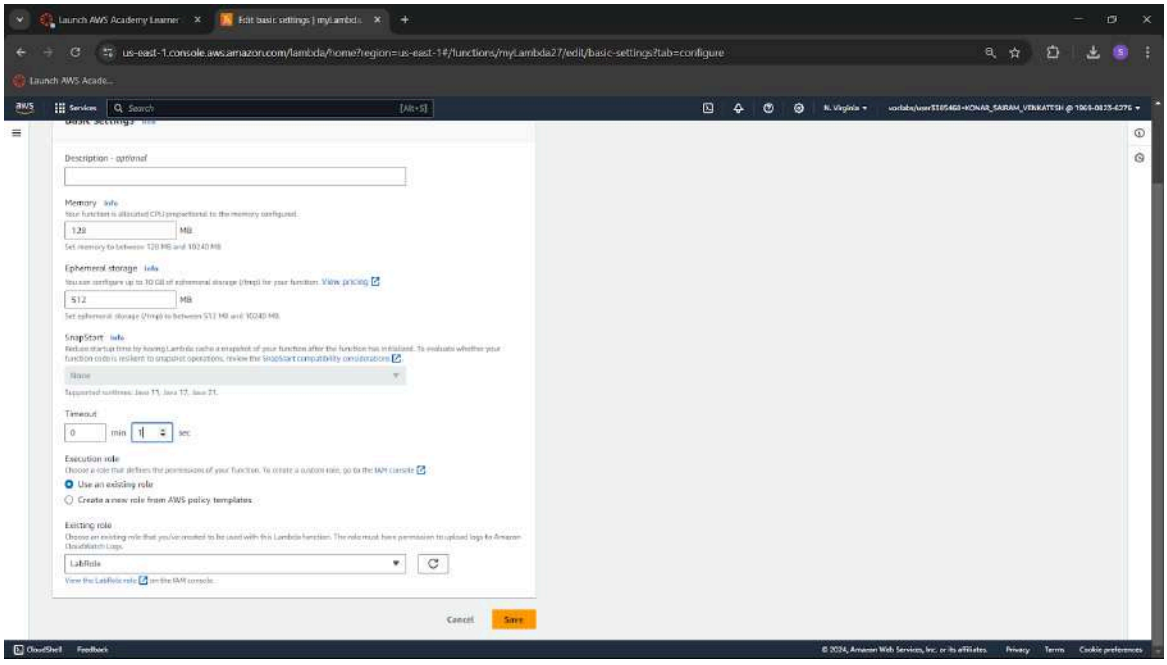


Step 3: Set up configurations and test events

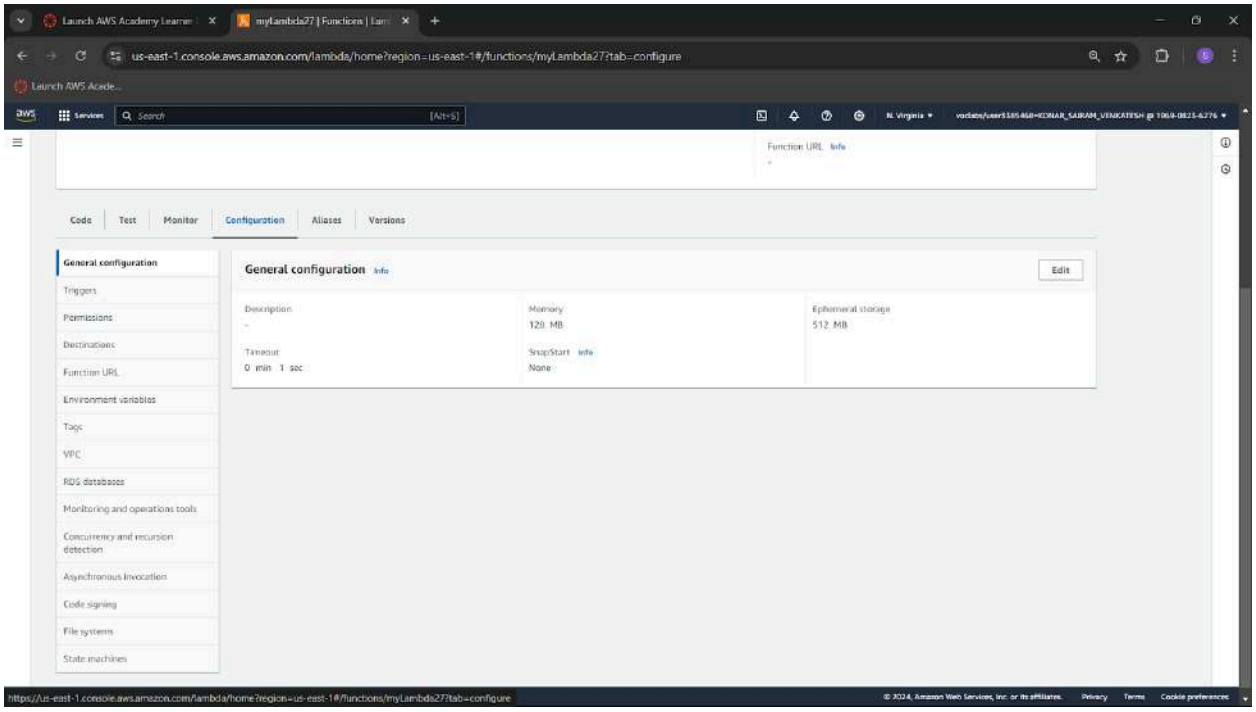
1) Just above the test code, you would find Configuration, click on it. Then click on Edit.



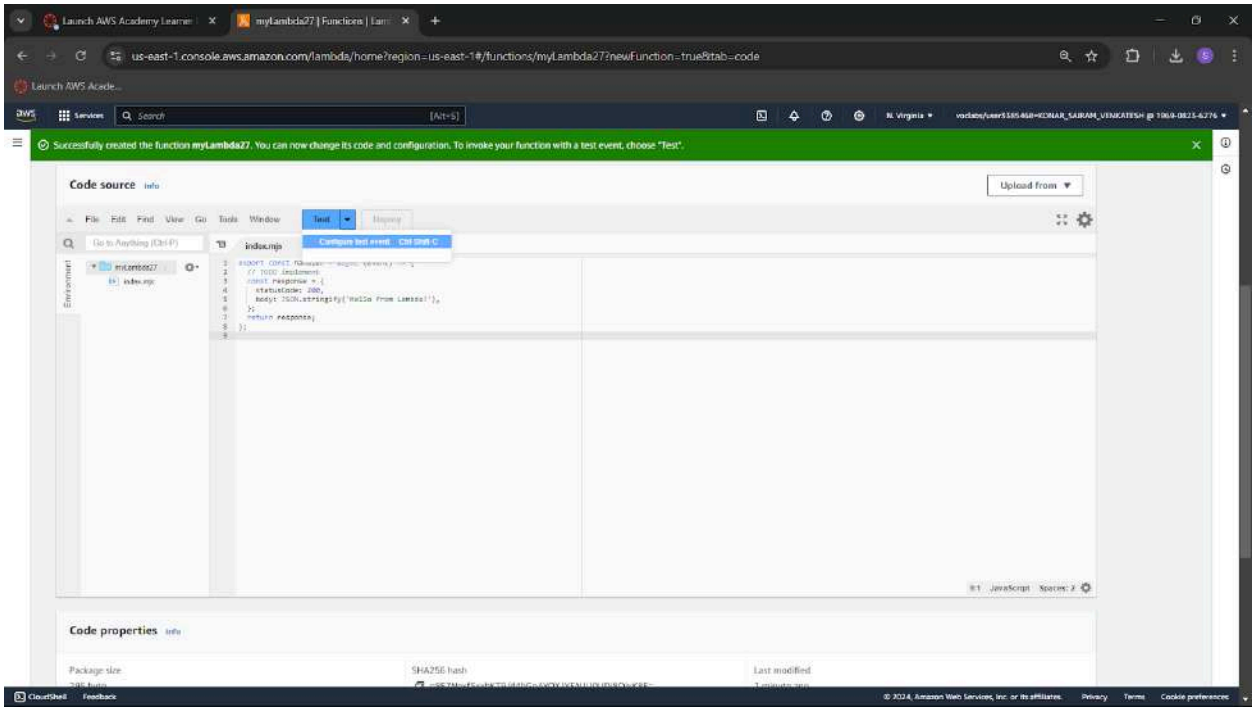
2) Here, change the Timeout to 1 sec. This is the time for which the function can be running before it is forcibly terminated.



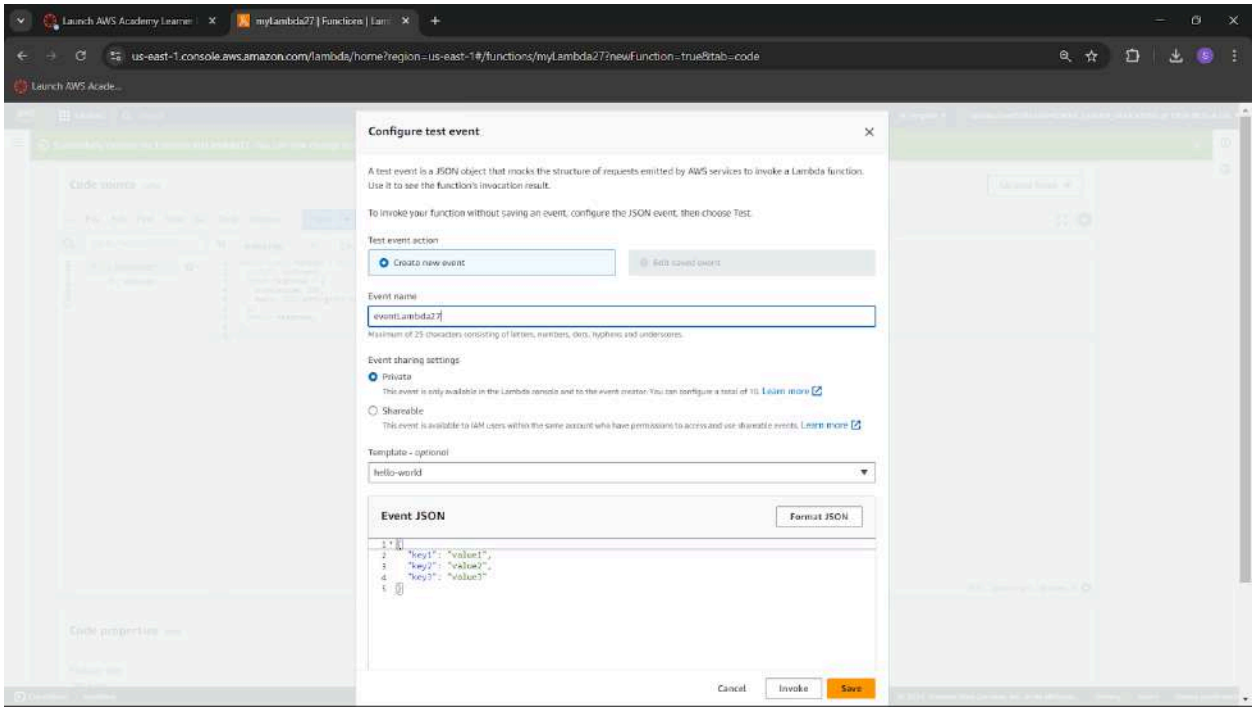
3) We can see the executed changes.



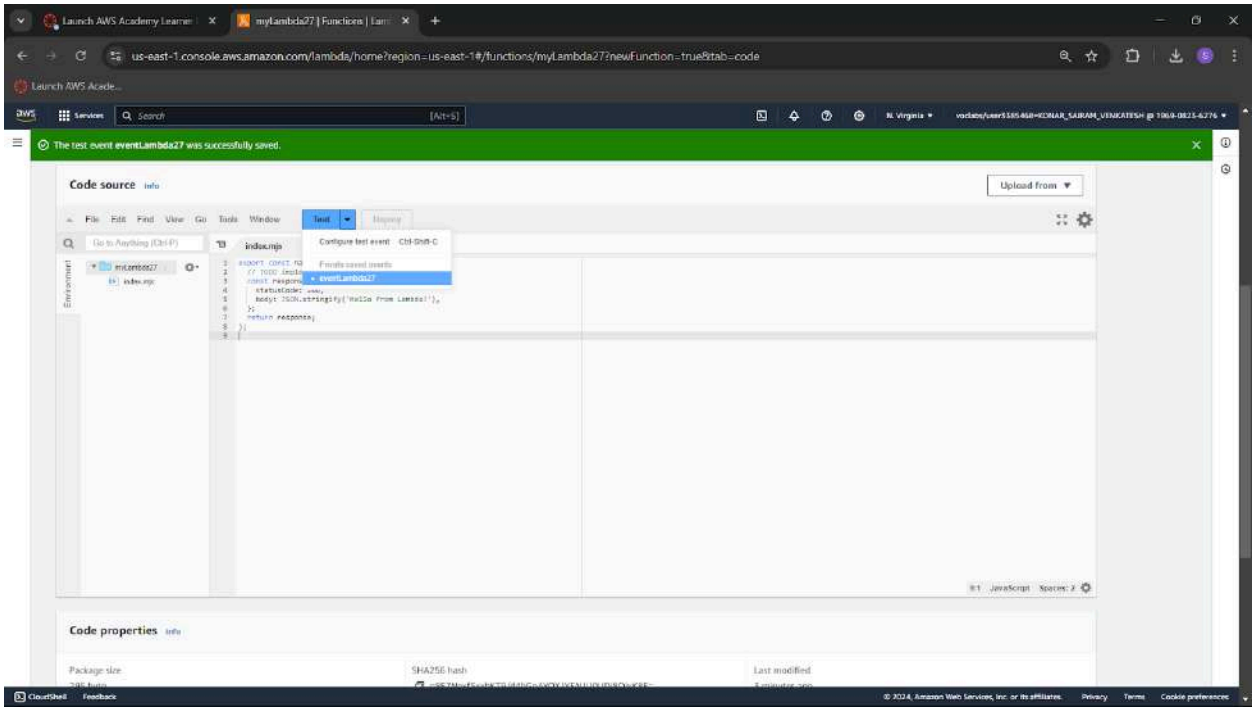
4) Switch back to the code tab. Click on the dropdown arrow near test. Then select configure test event.



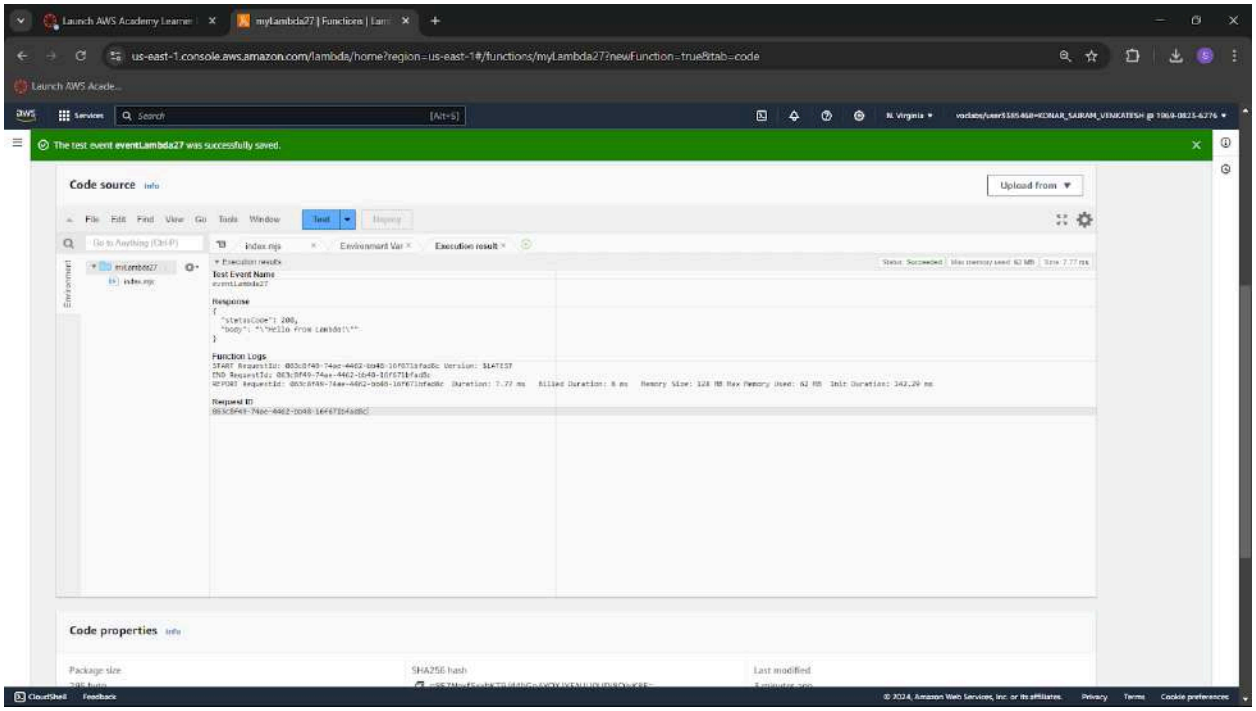
5) Here, create a new event, keep the other options default and save the event.



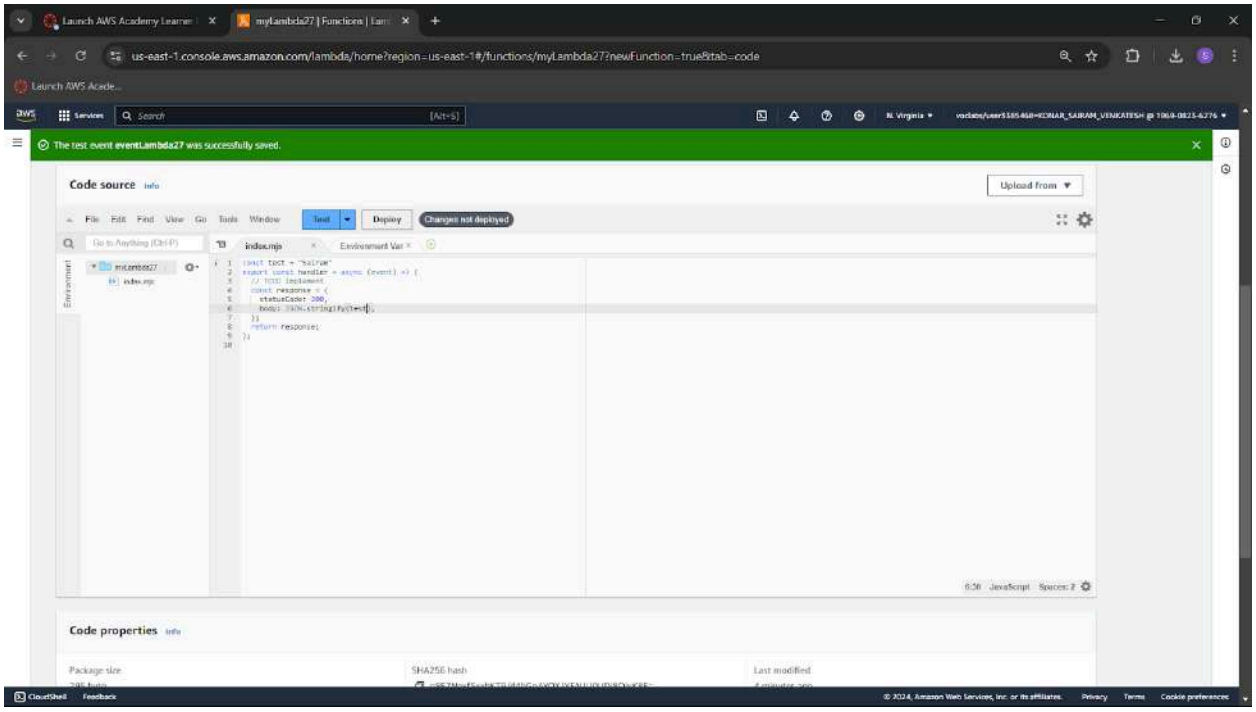
6) Now, again click on the dropdown. This time, select the event you have created. Then, click on TEST.



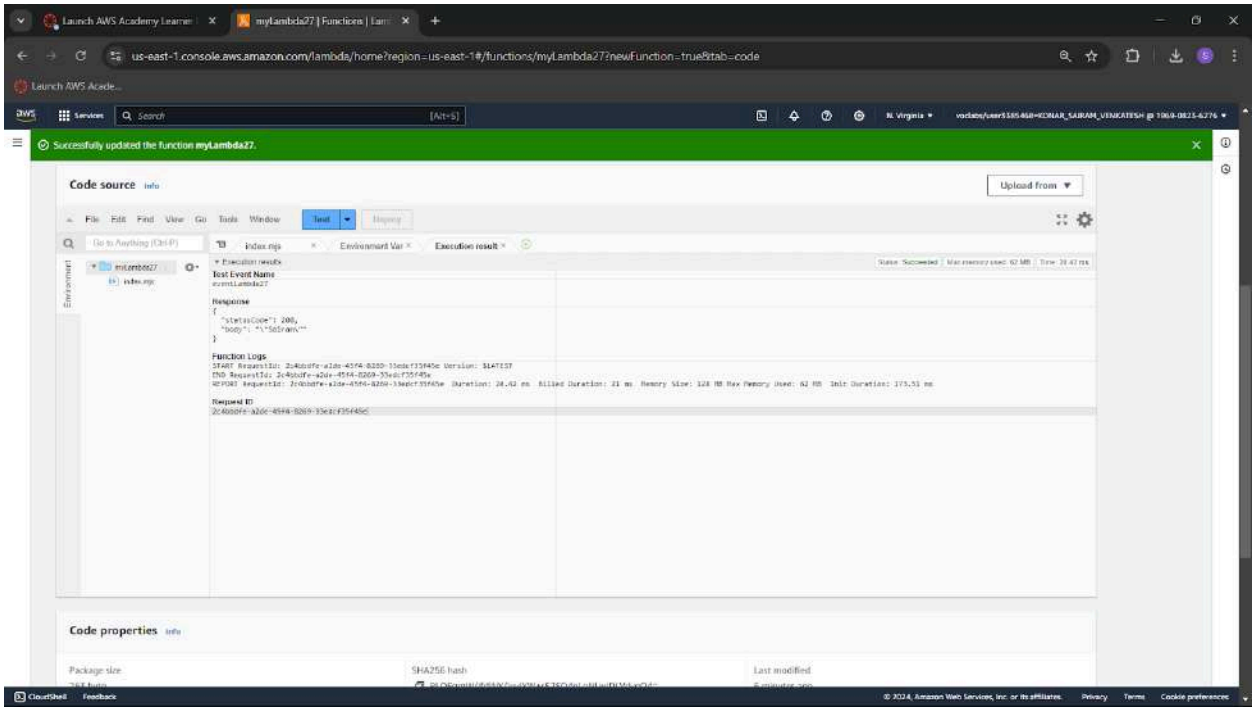
7) We can see the expected output for the sample code.



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



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



Conclusion:

In this experiment, we successfully explored the AWS Lambda service by creating and configuring Lambda functions using Python, Java, or Node.js. We learned how to set up a Lambda function, modify its configuration (such as adjusting the timeout), and test the function with custom events. Through this process, we observed how Lambda handles executions, including managing timeouts and returning expected outputs based on the code changes.

Sairam Konar D15C 27 2024-25

Experiment 12

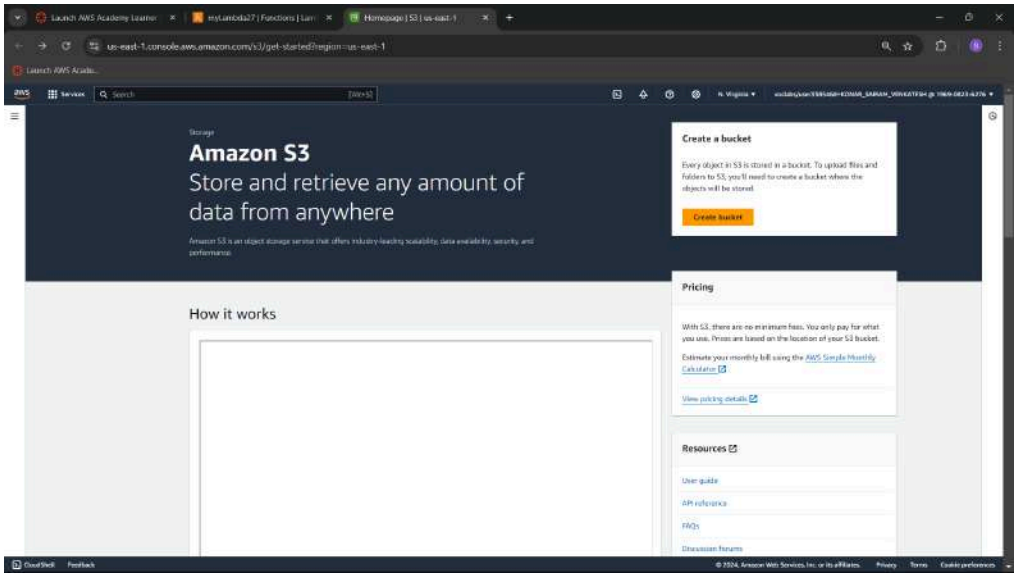
Aim: To create a Lambda function which will log “An Image has been added” once you add an object to a specific bucket in S3.

Prerequisites:

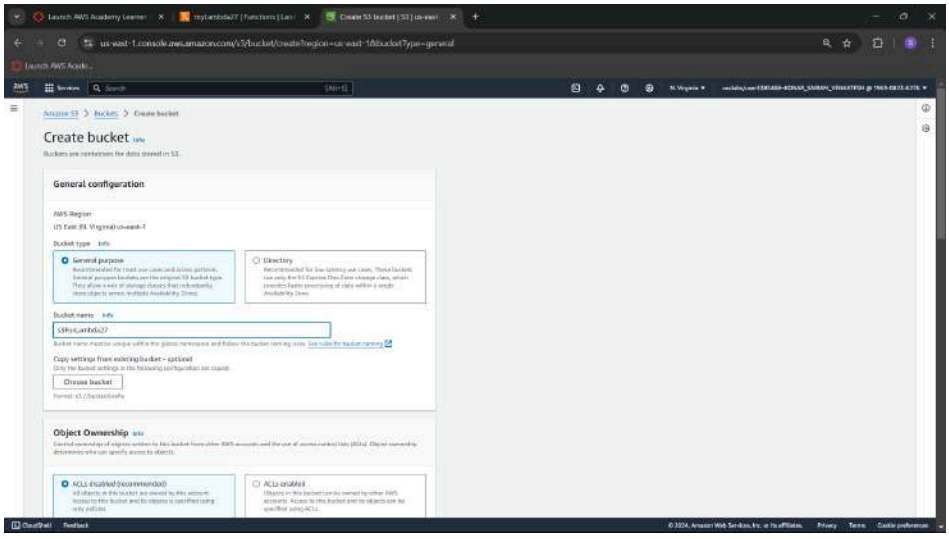
1) AWS account (acacdemy preferable)
2) Lambda function (created in the previous experiment).

Step 1: Create a s3 bucket.

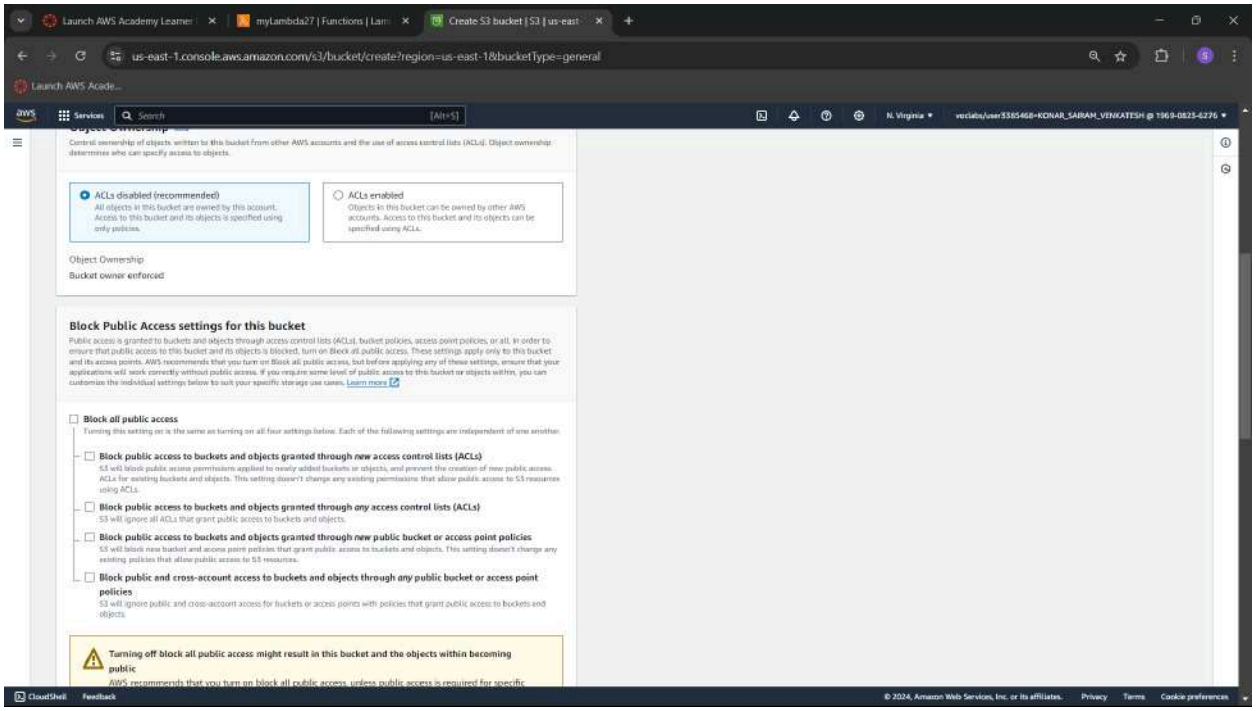
1) Search for S3 bucket in the services search. Then click on create bucket.



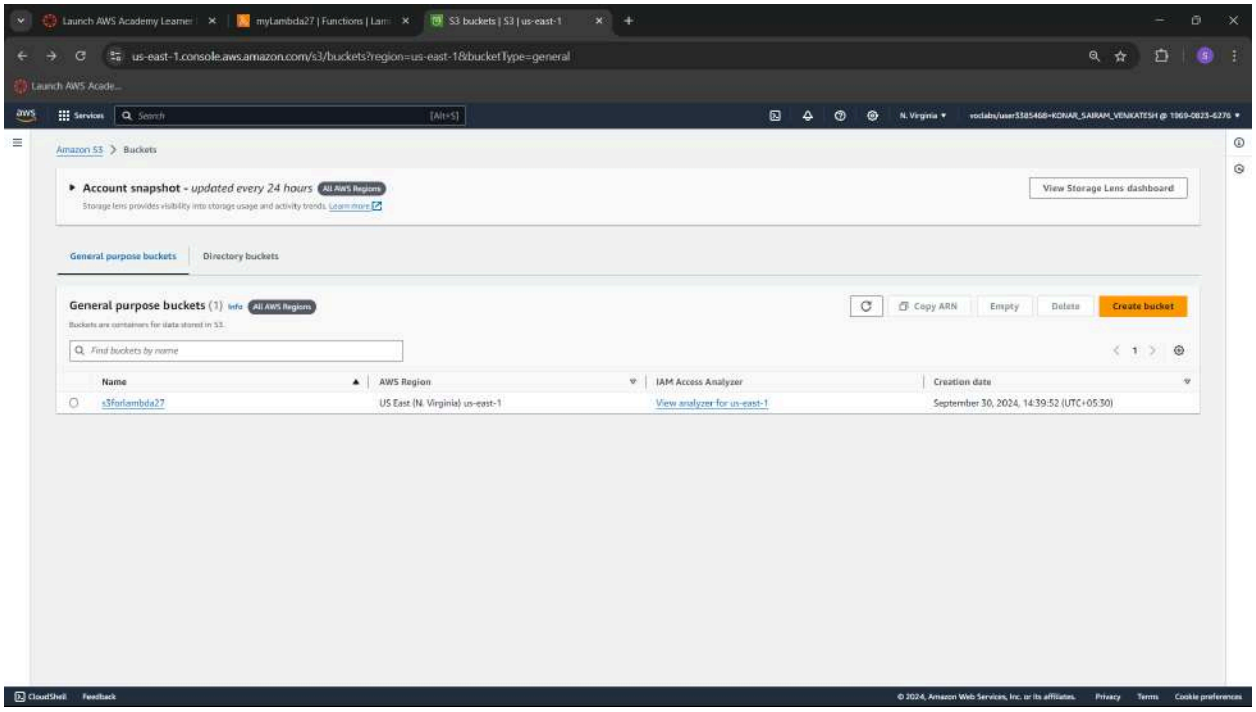
2) Keep the bucket as a general purpose bucket. Give a name to your bucket.



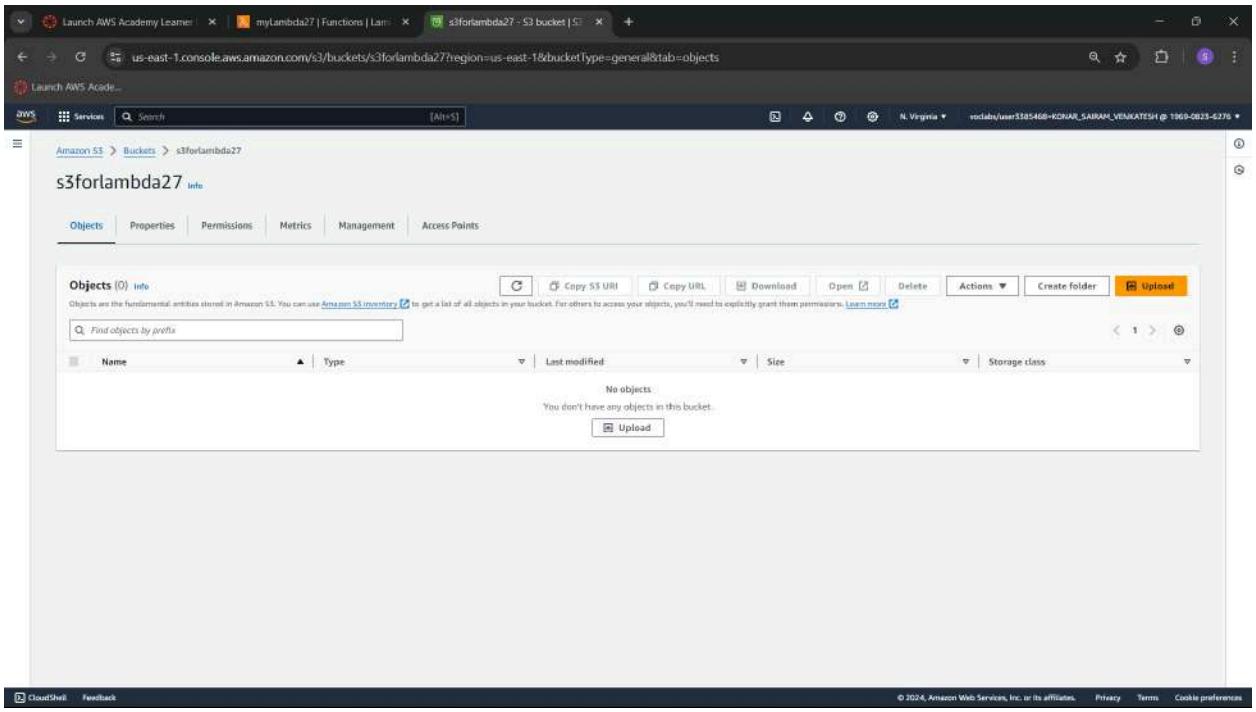
3) Uncheck block all public access.

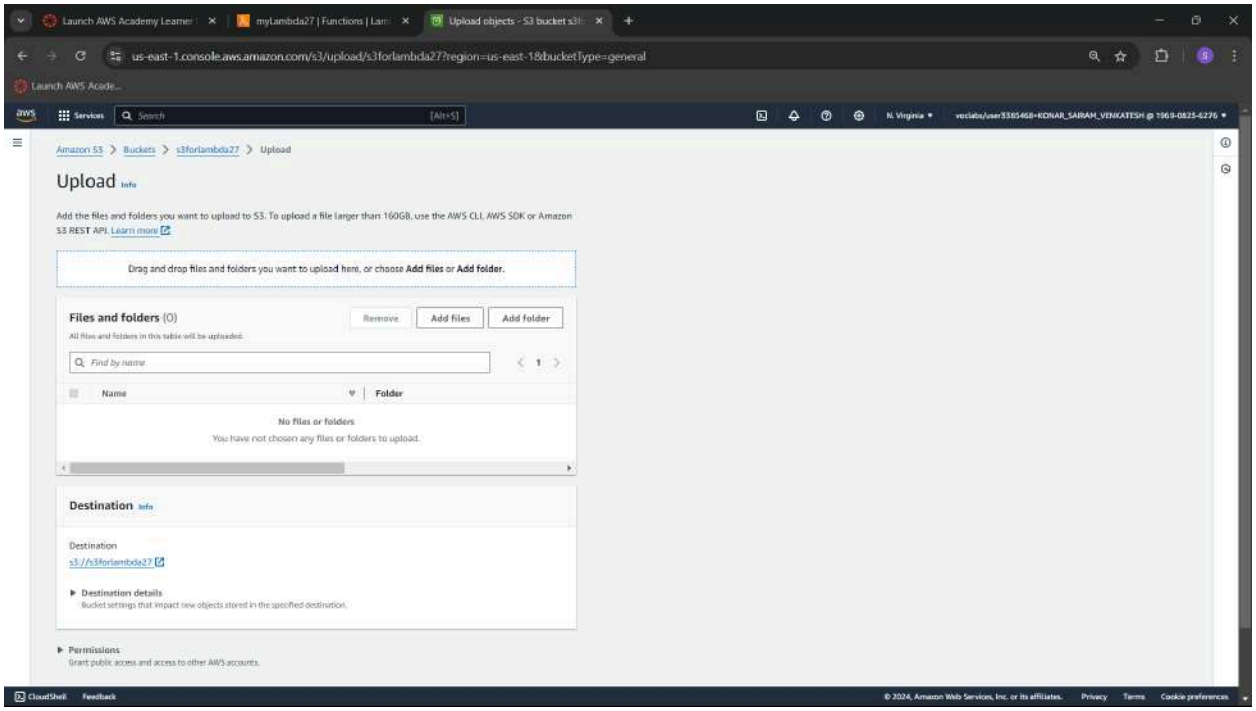


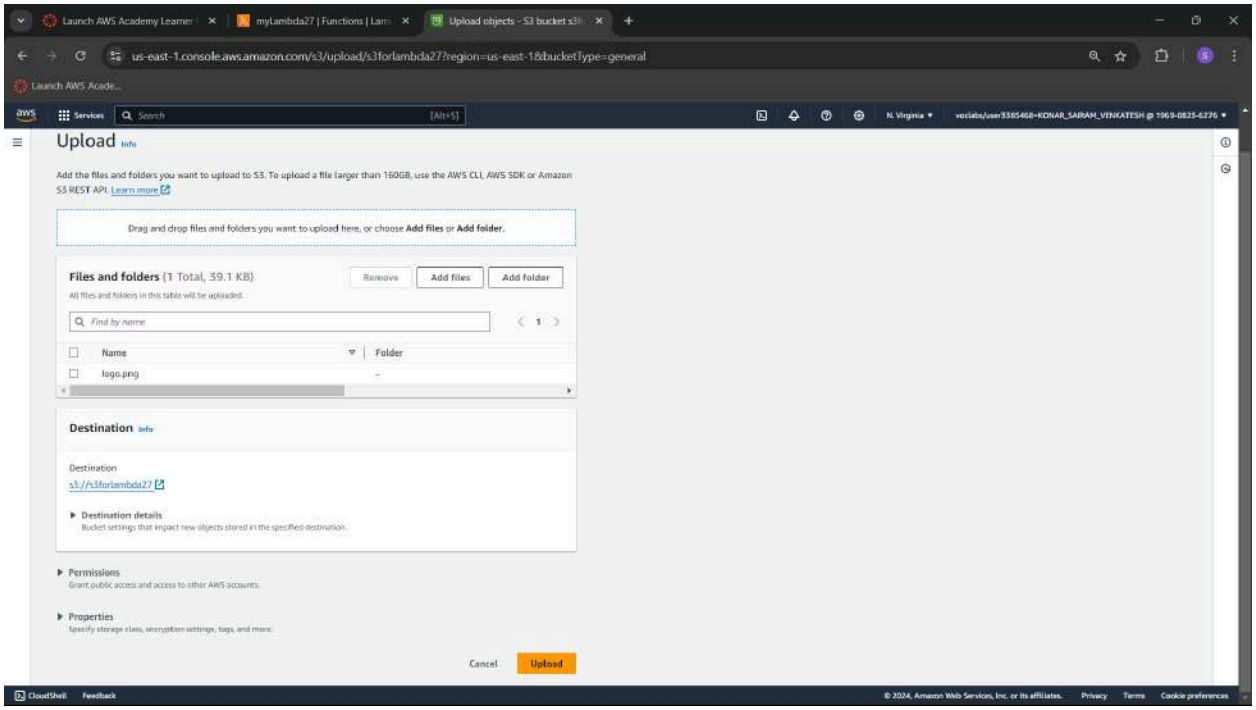
4) Keeping all other options same, click on create. This would create your bucket. Now click on the name of the bucket.



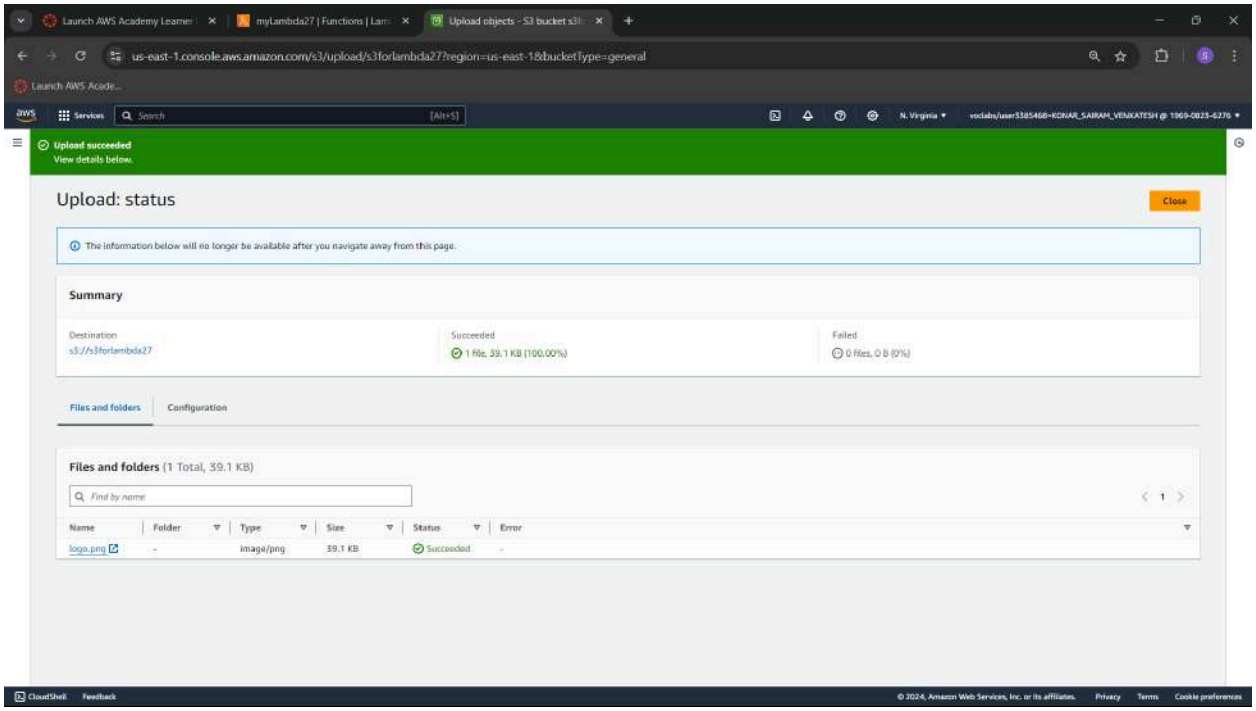
5) Here, click on upload, then add files. Select any image that you want to upload in the bucket and click on upload.





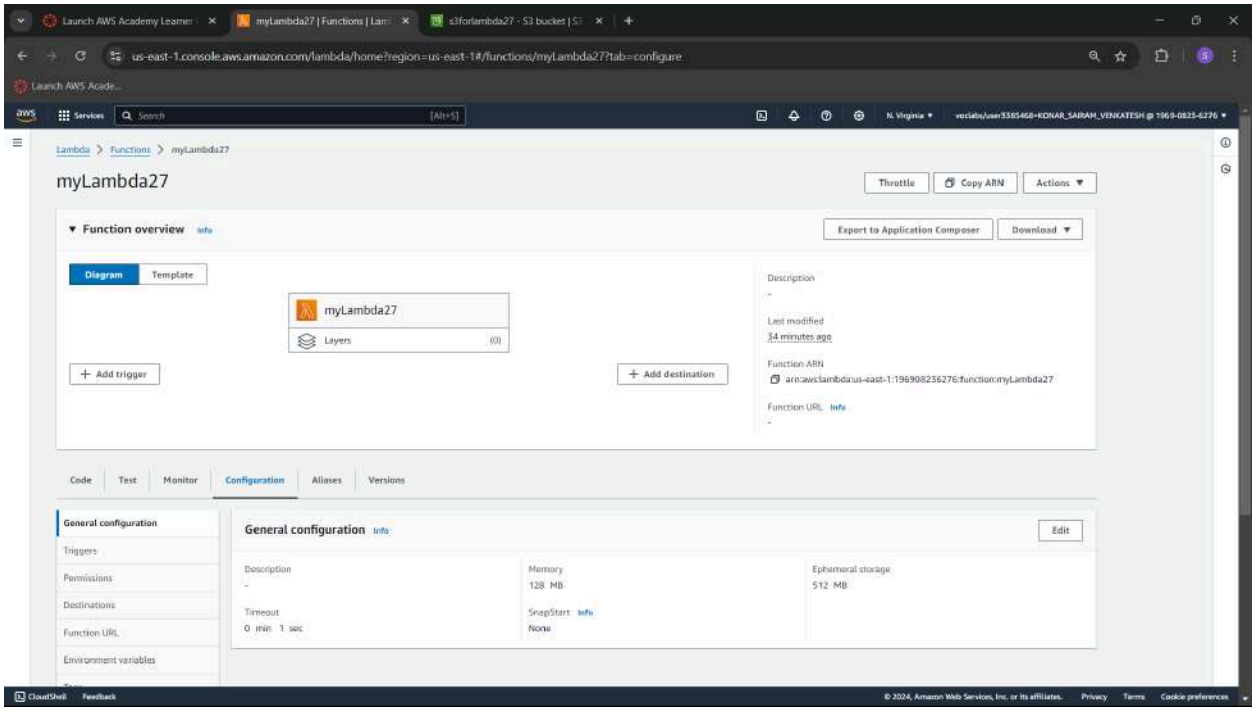


6) The image has been uploaded to the bucket.

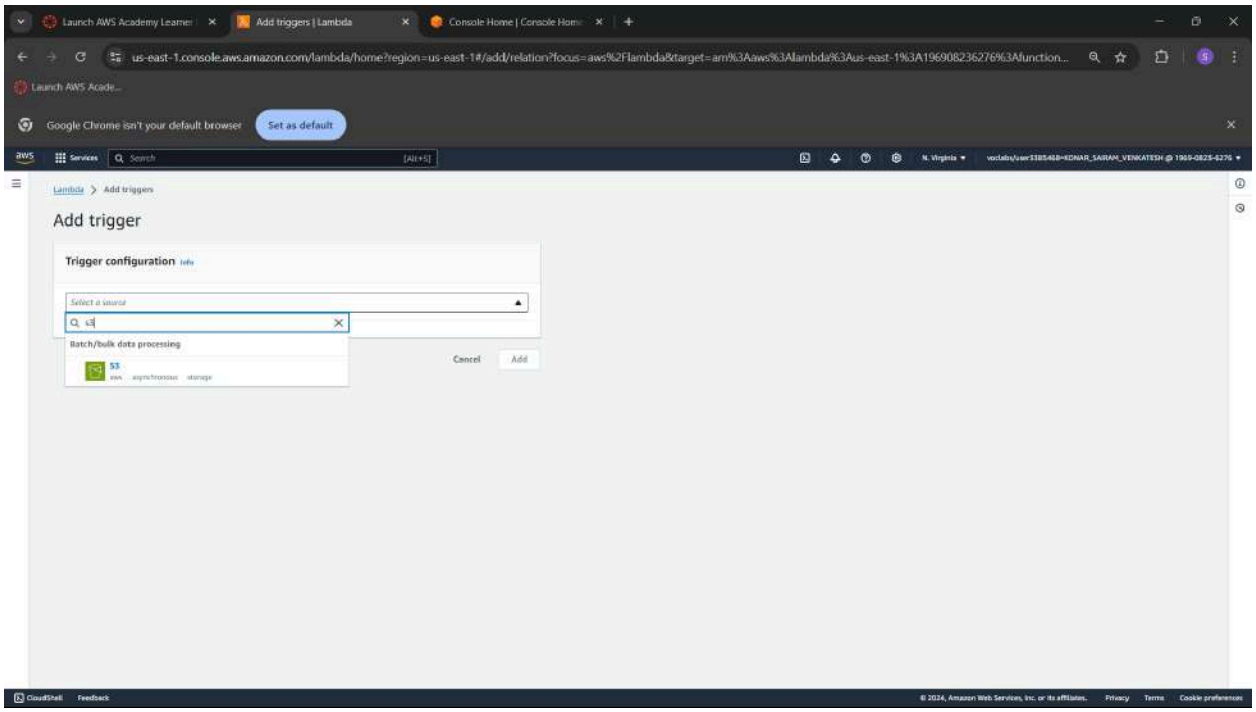


Step 2: Configure Lambda function

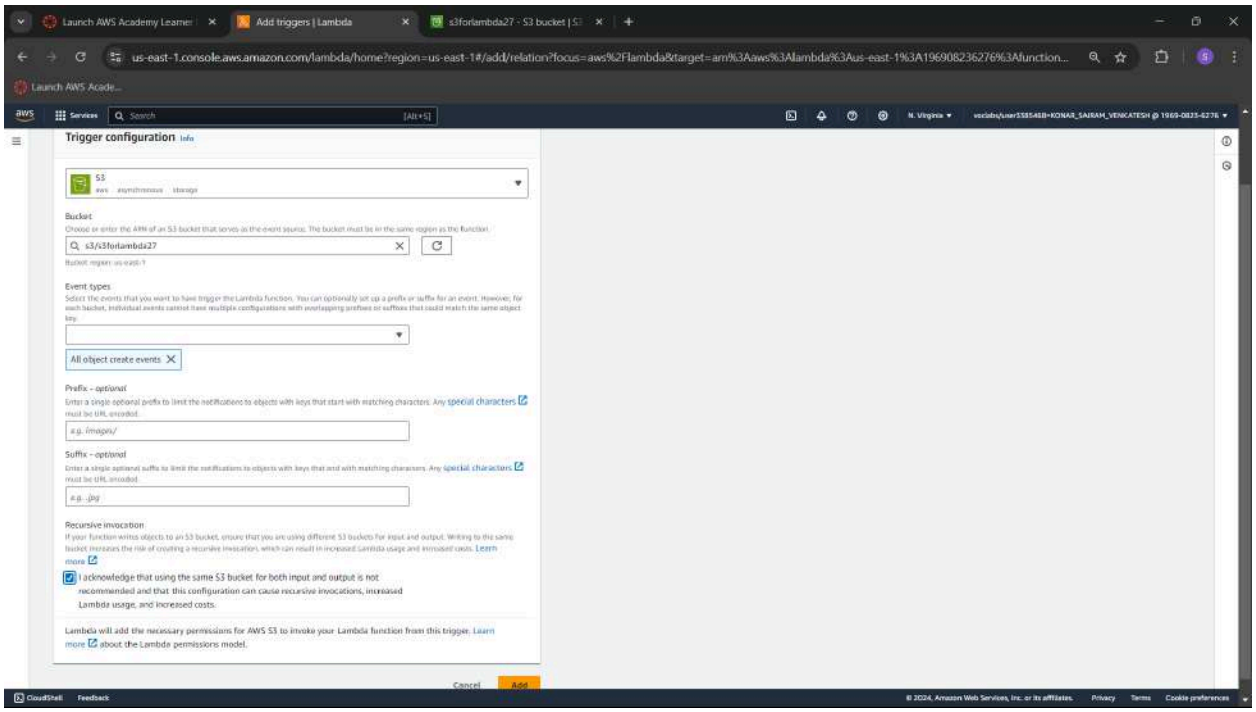
1) Go to the lambda function you had created berfor. (Services → Lambda → Click on name of function). Here, click on add trigger.

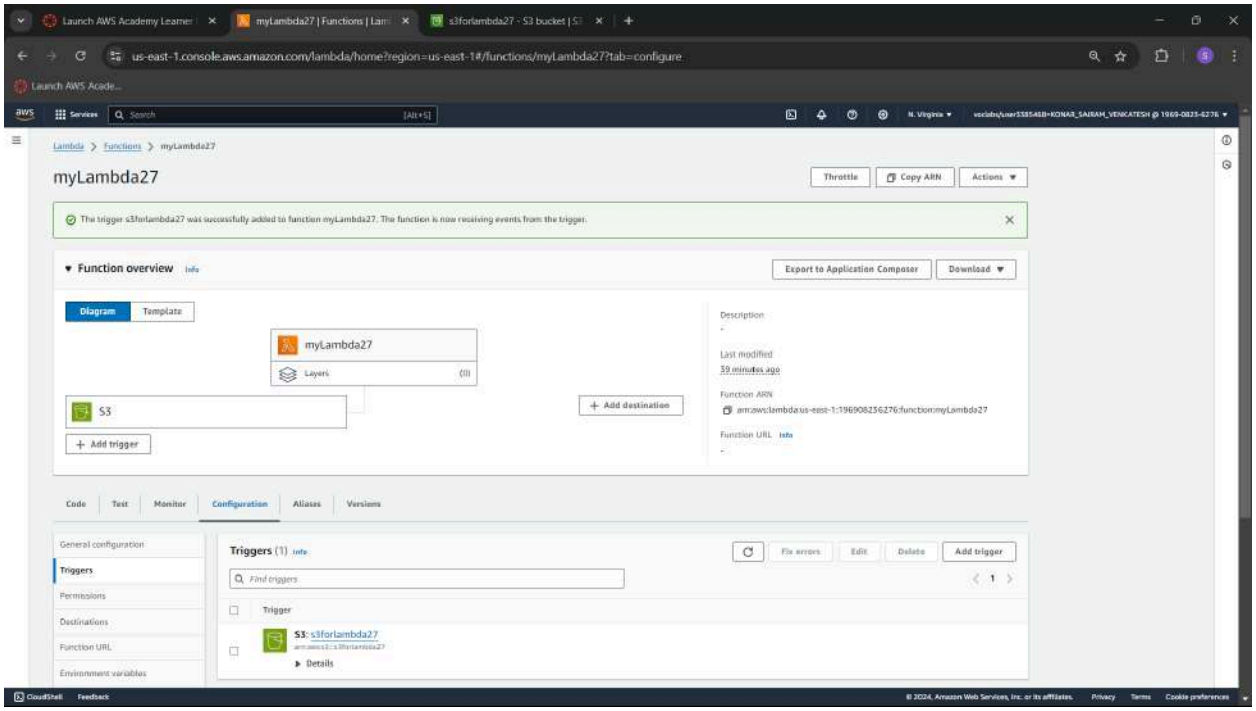


2) Under trigger configuration, search for S3 and select it.



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



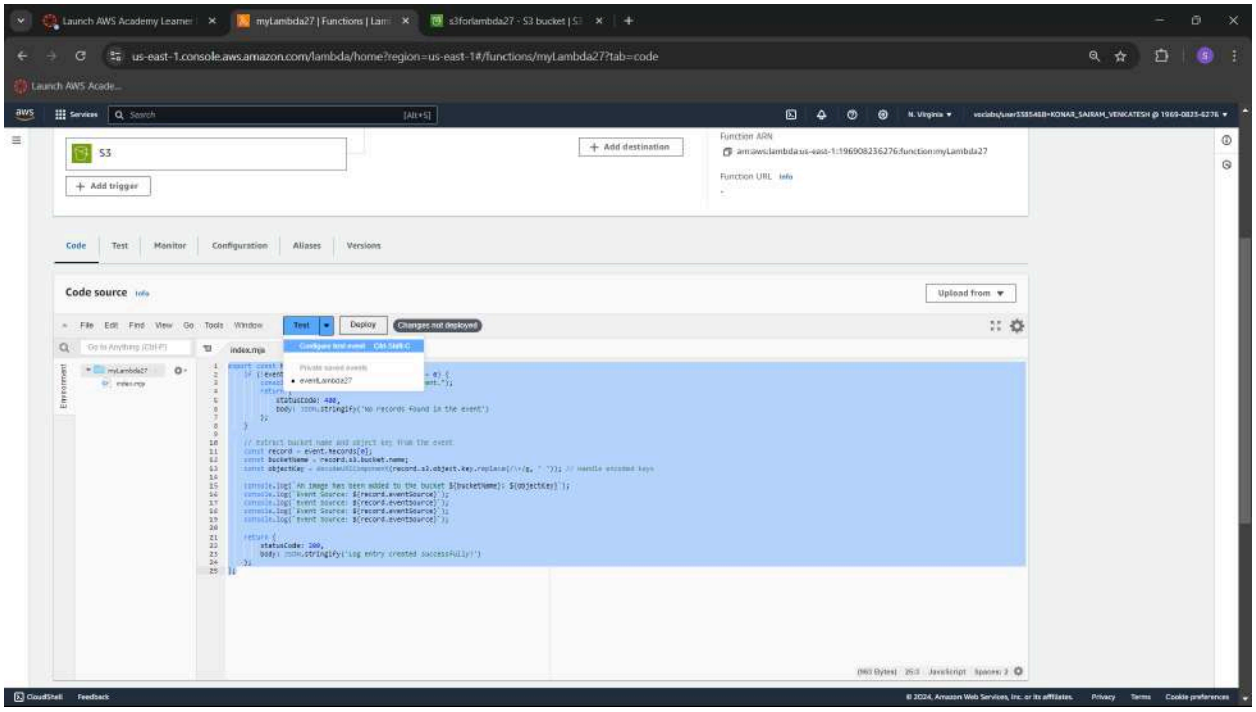


4) Scroll down to the code section of the function. Add the following javascript code to the code area by replacingthe existing code.

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

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

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

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

This 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 test, then click on configure test event.

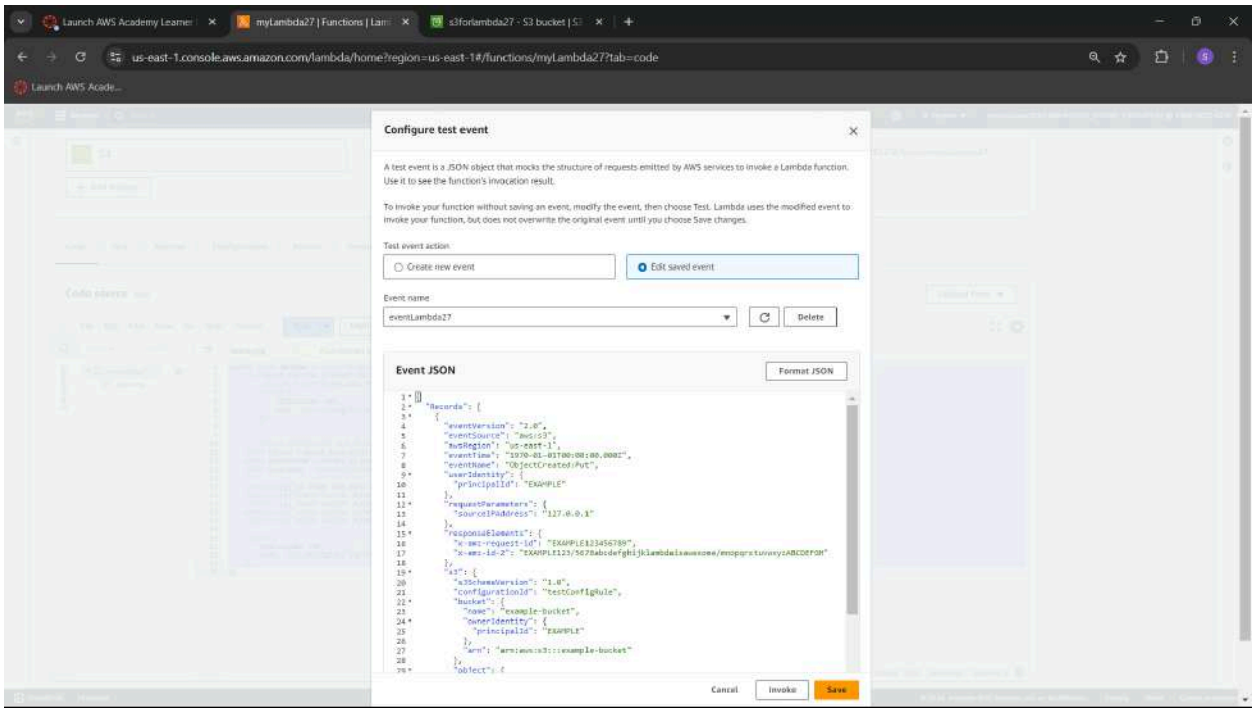


6) Here, select edit saved event. Select the event taht you had created before. Under Event JSON, paste the following code.

```
{
  "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",
```

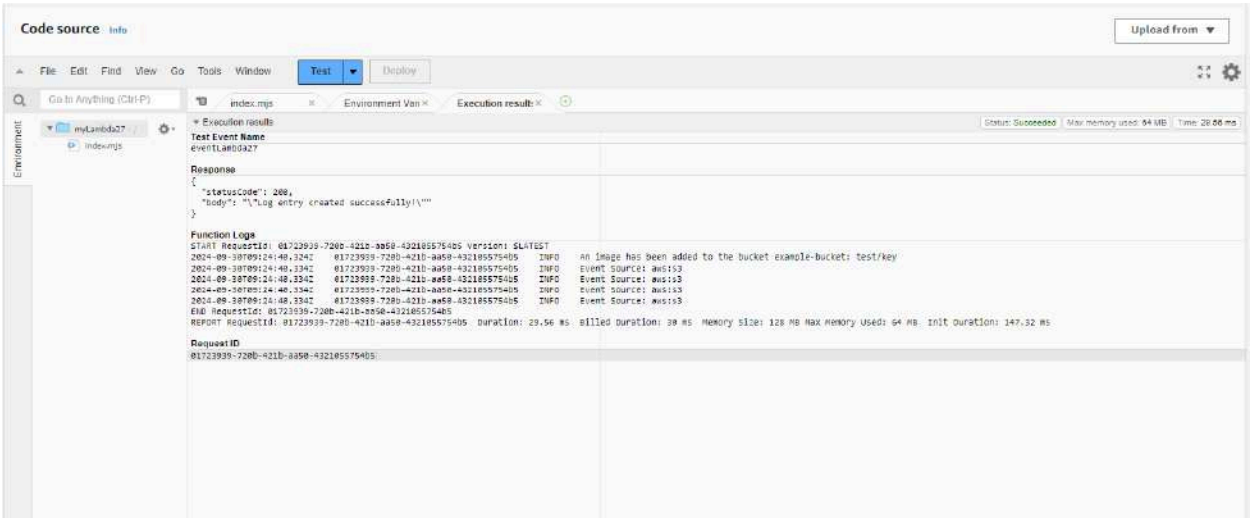
```
      "bucket": {
        "name": "example-bucket",
        "ownerIdentity": {
          "principalId": "EXAMPLE"
        },
        "arn": "arn:aws:s3:::example-bucket"
      },
      "object": {
        "key": "test%2Fkey",
        "size": 1024,
        "eTag": "0123456789abcdef0123456789abcdef",
        "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.



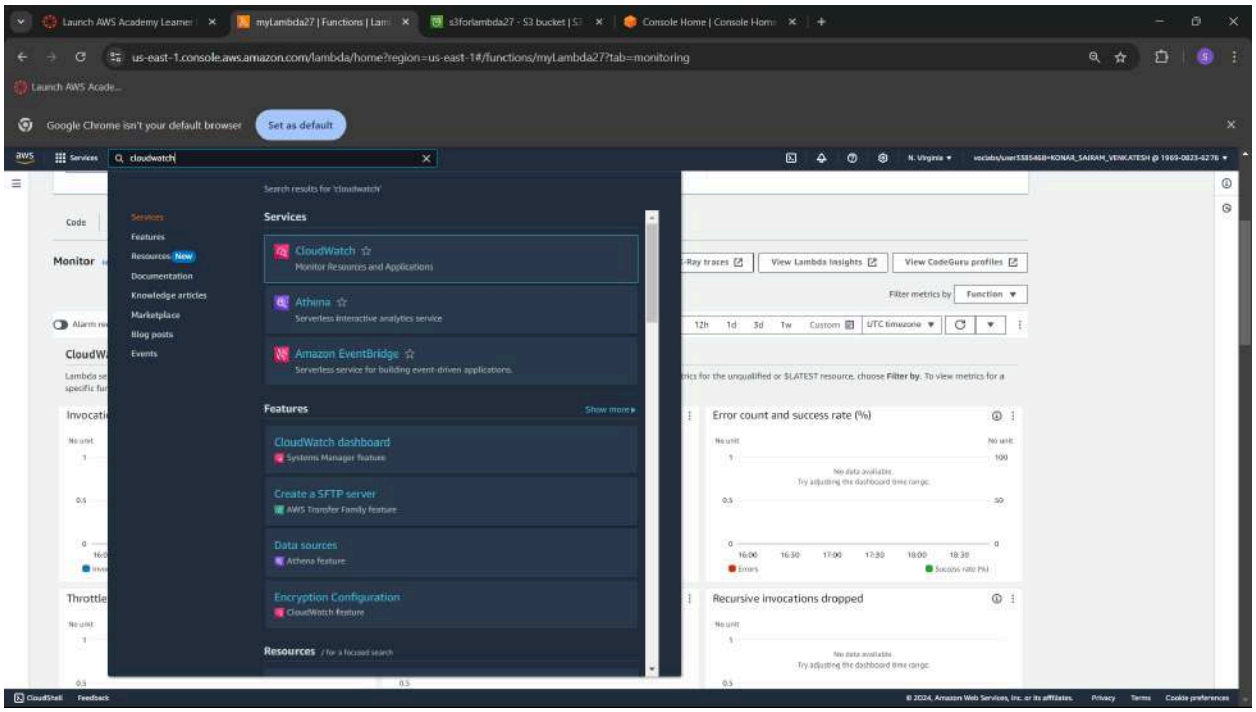
Save the changes. Then deploy the code changes by clicking on deploy.

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.

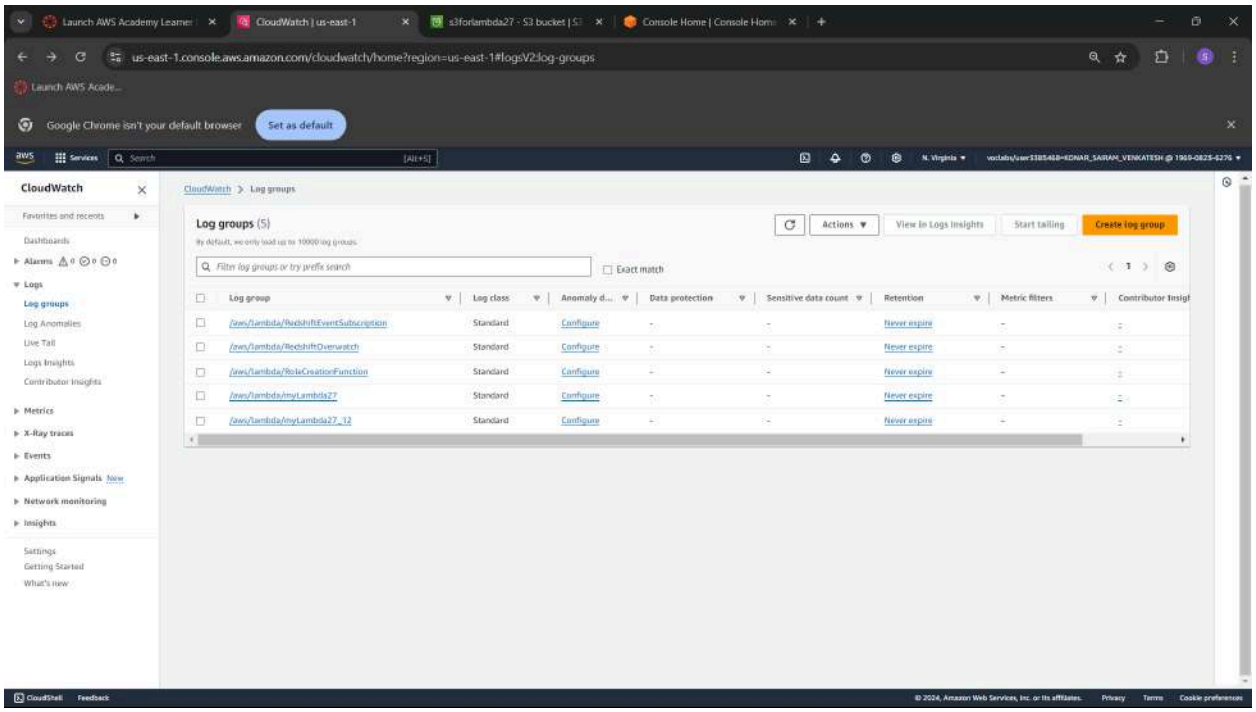


Step 3: Check the logs

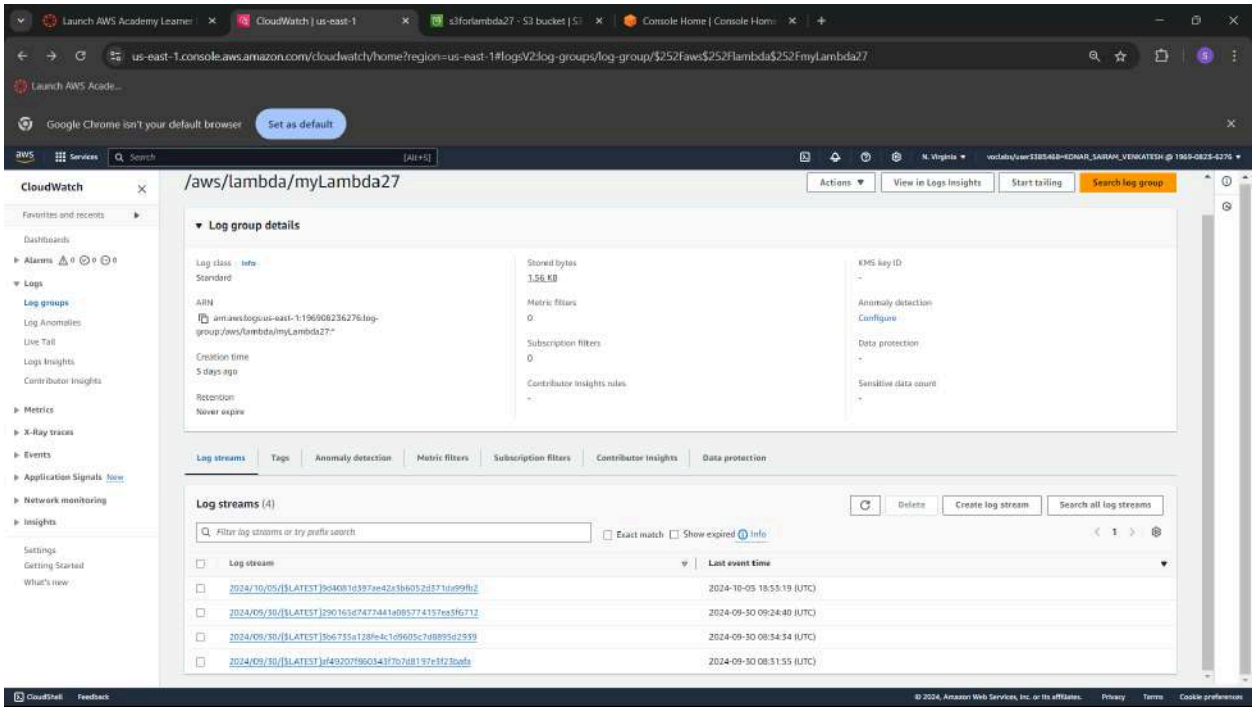
1) To check the logs explicitly, search foe CloudWatch on services and open it in a new tab.



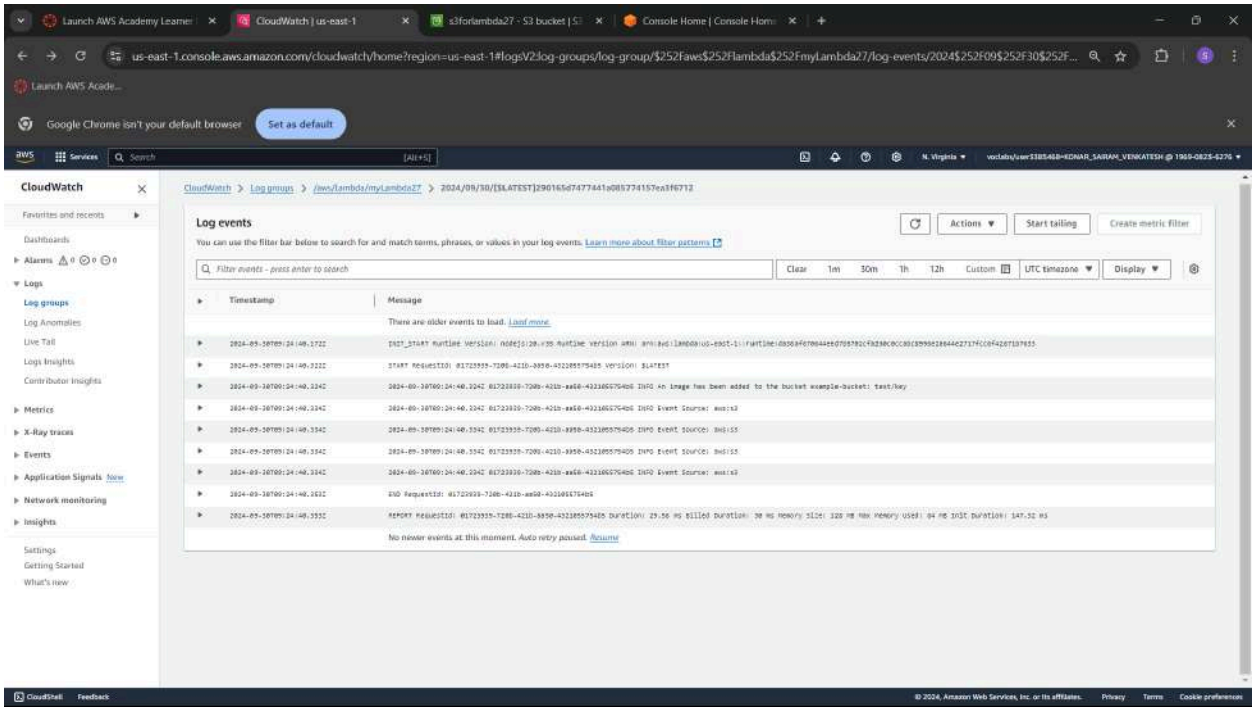
2) Here, Click on Logs → Log Groups. Select the log that has the lambda function name you just ran.



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



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



Conclusion:

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