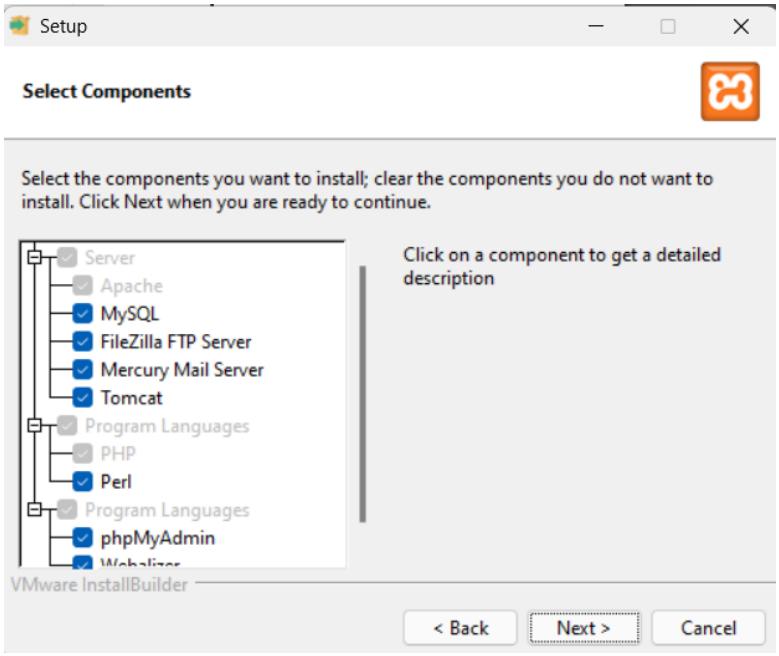
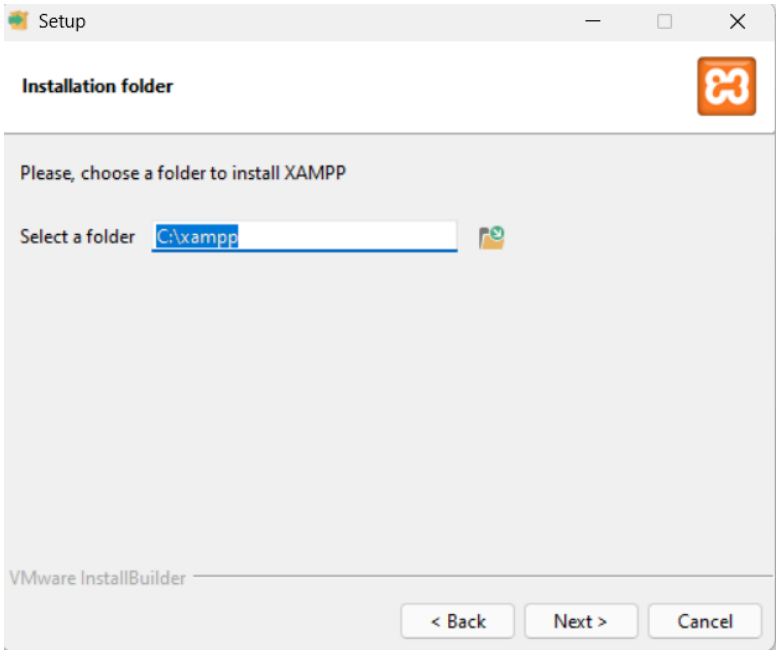
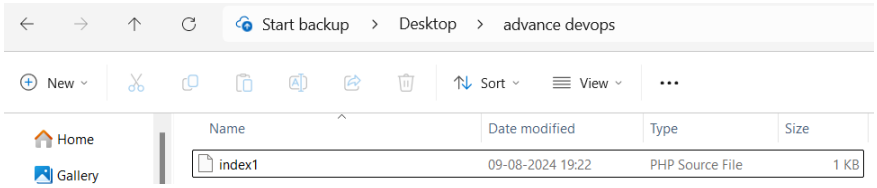
Static Hosting on Local Server (Xampp)

Install Xampp from https://www.apachefriends.org/ and follow the default steps for installing xampp

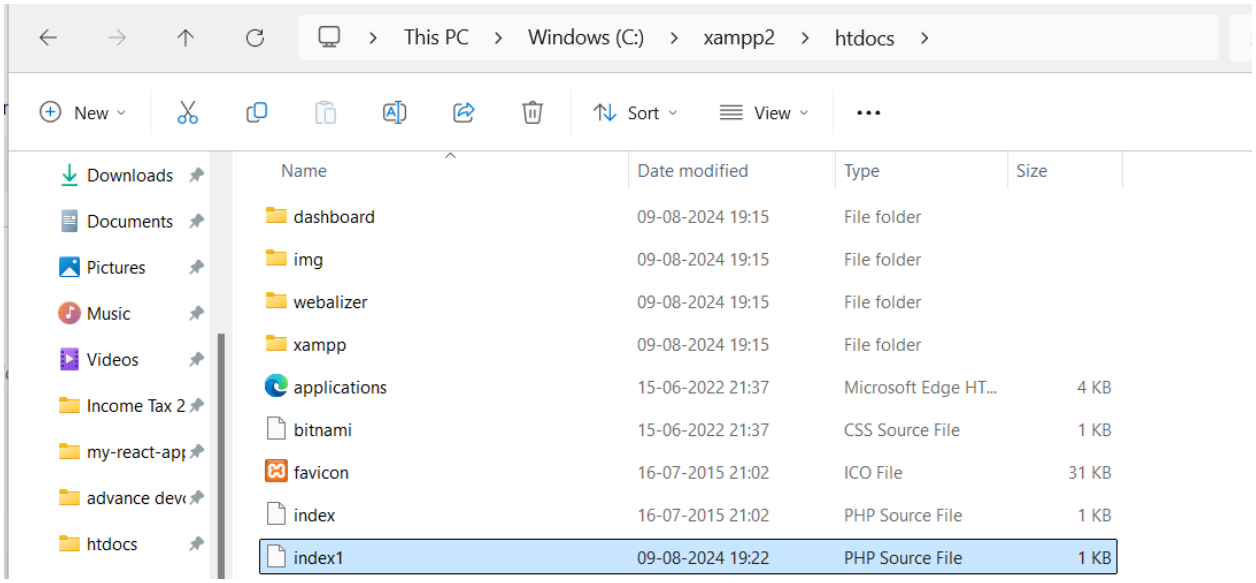




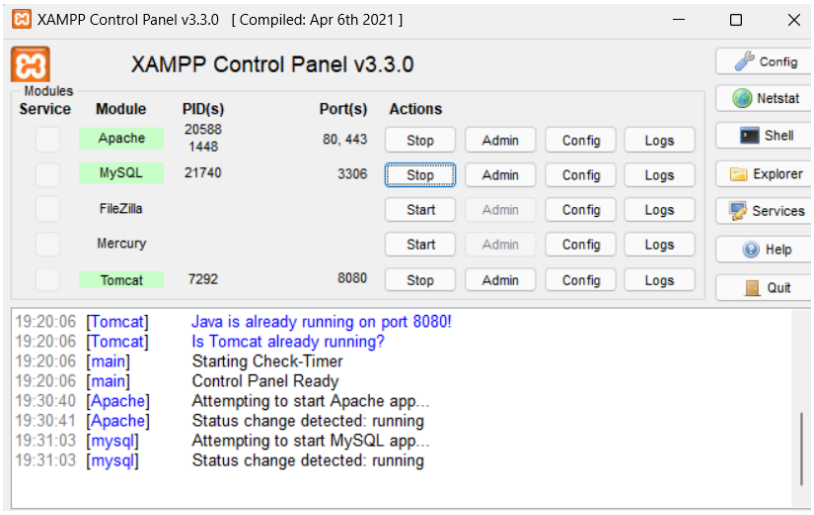
Setup a file that is to be hosted on the server. Make sure the file has extension .php



Go to the directory where XAMPP was installed. Go to htdocs folder. Place your folder in this directory.



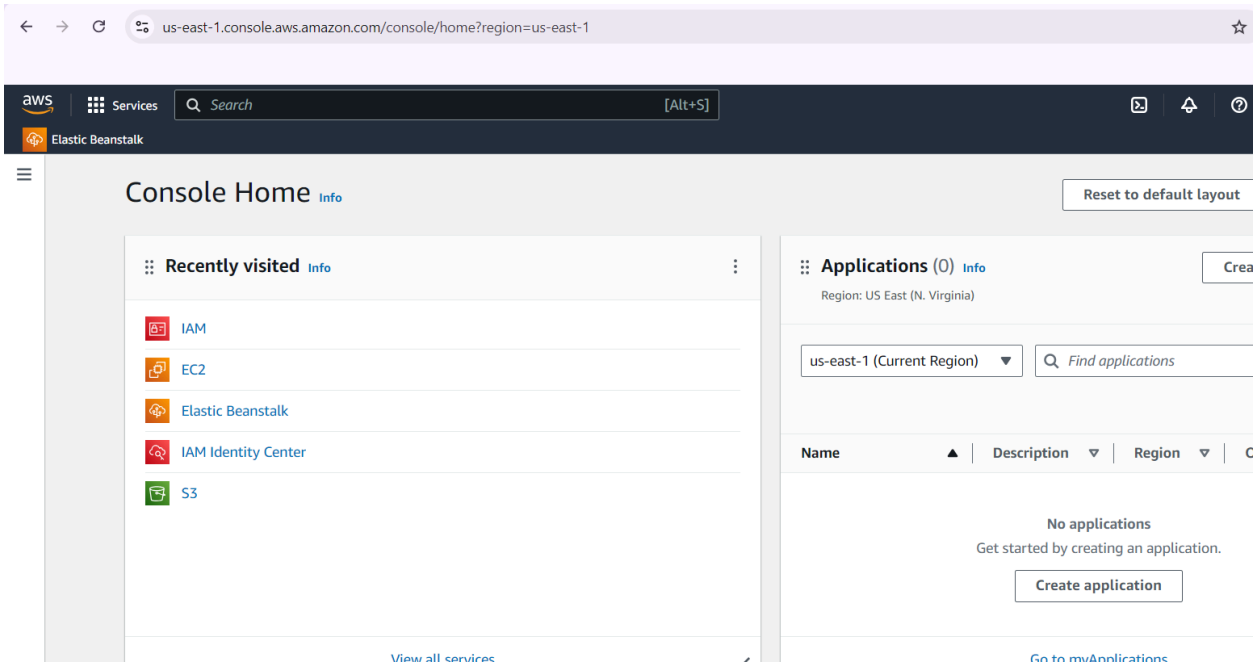
Open XAMPP Control Panel, start the Apache service (Required) and mySQL service (if needed)



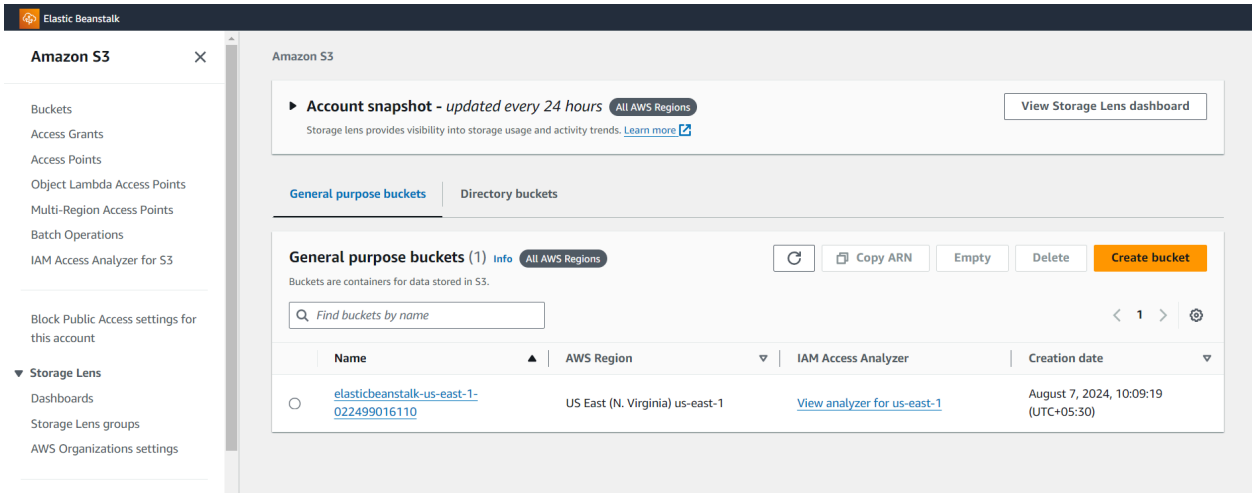
Open your web browser. Type localhost/YOUR_FILENAME.php. This will open your website on your browser.

Hosting a static website on amazon s3

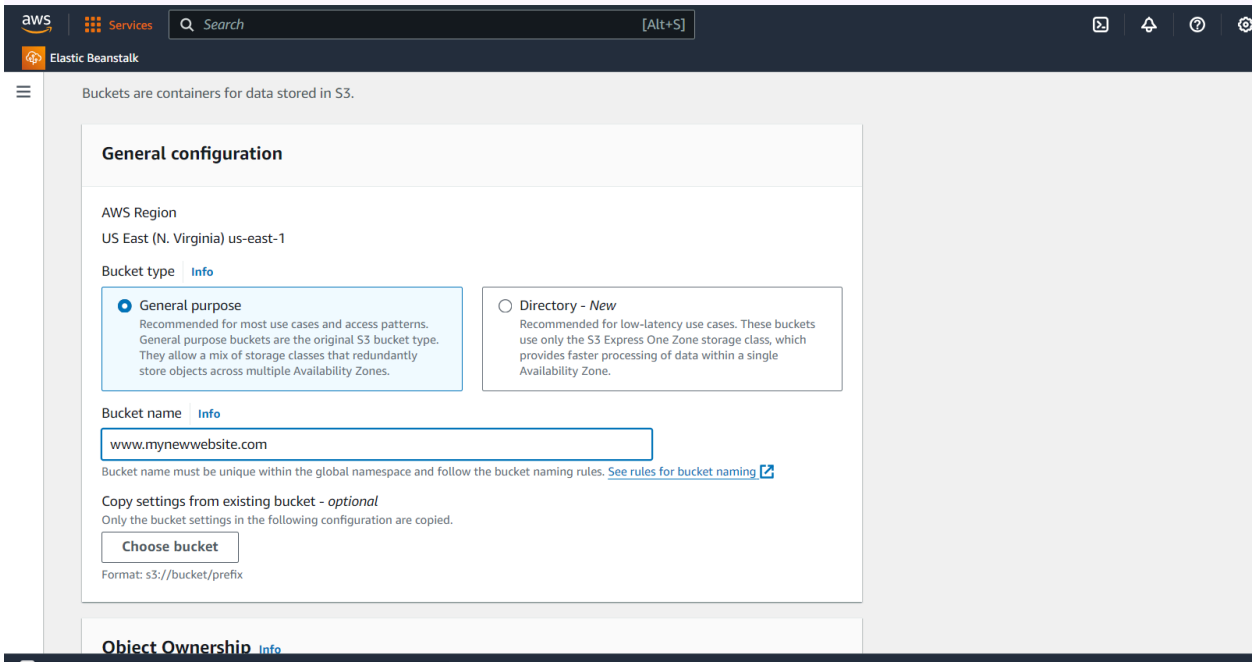
Go to console.aws.amazon.com and click on S3



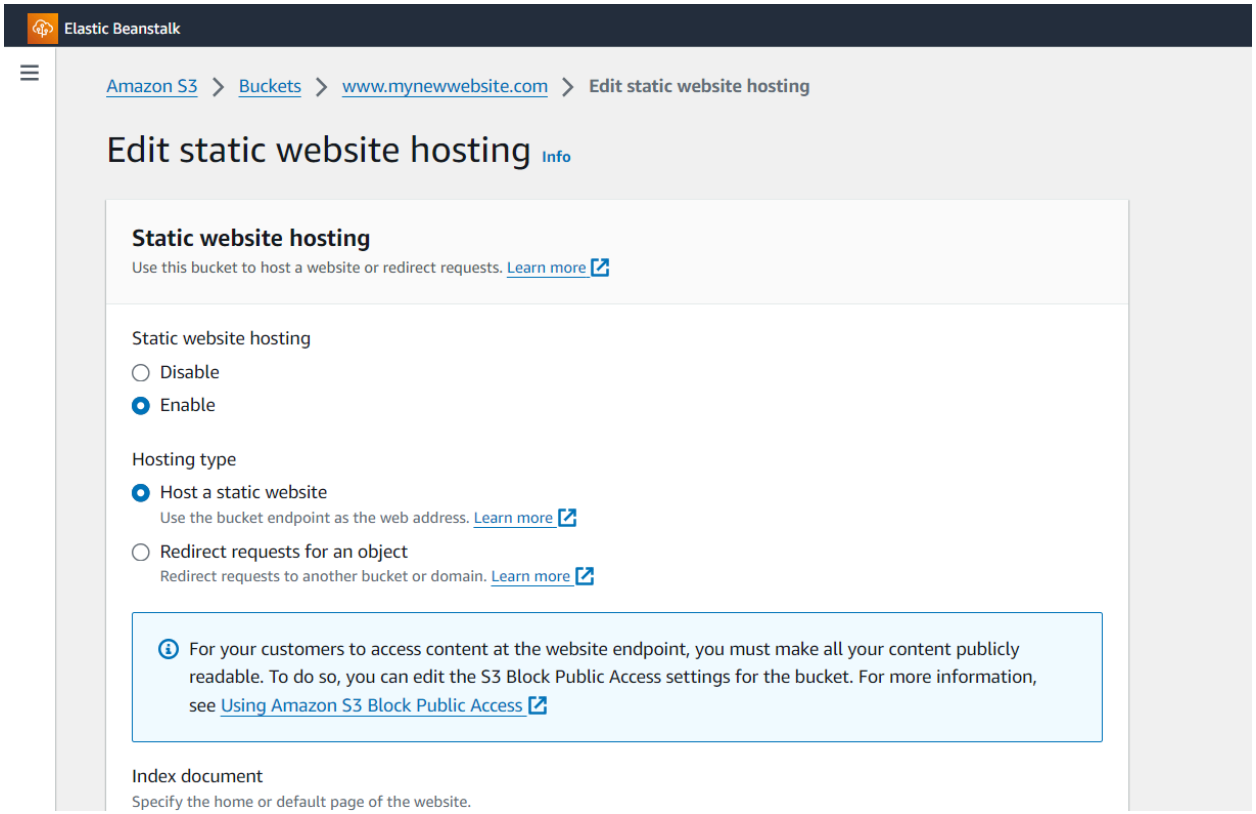
Click on Create Bucket

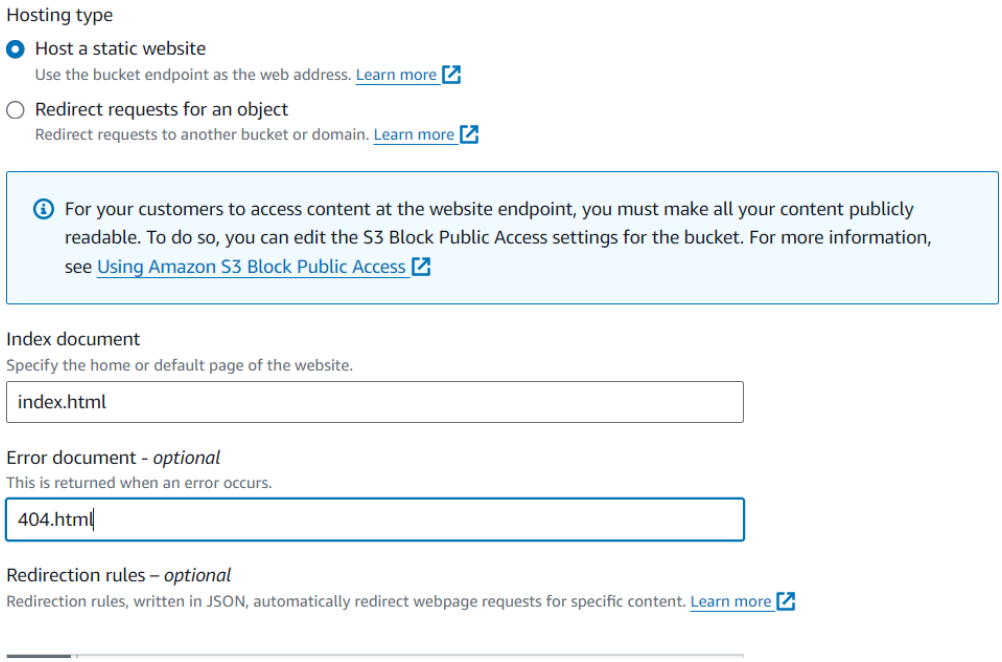


Write the bucket name for your website and click create bucket

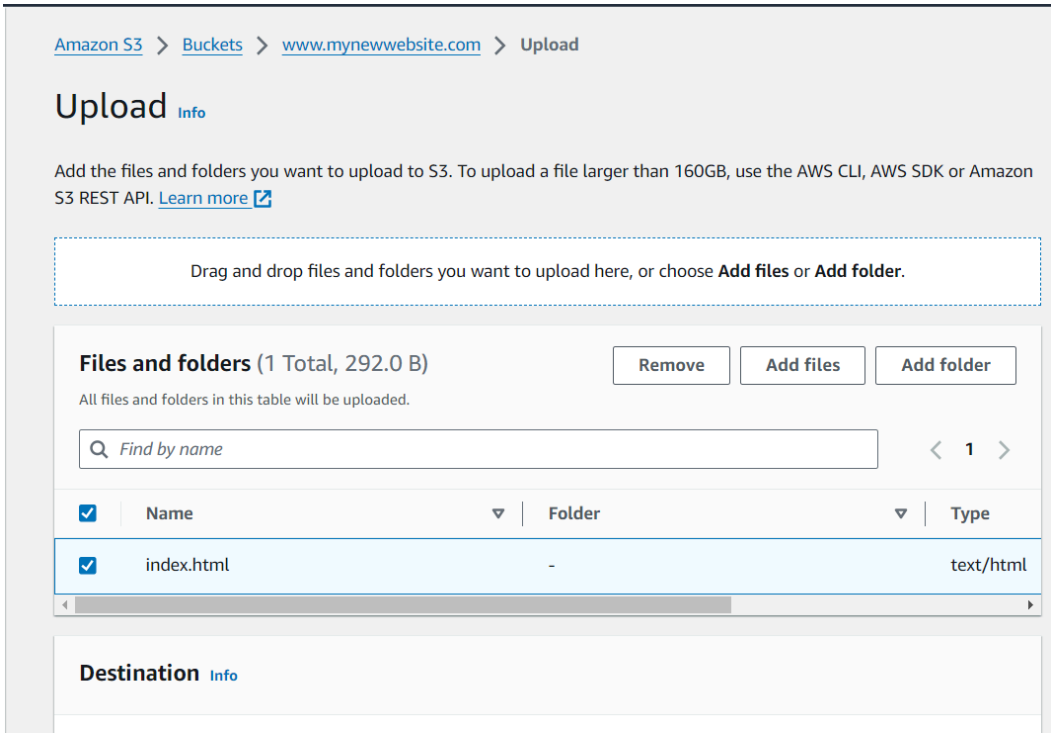


Enable static website hosting after creating the bucket and fill in the details as follows. Then click on Save changes





Now you can access your website from the endpoint http://www.mynewwebsite.com.s3-website-us-east-1.amazonaws.com . Now go to the bucket and click on upload files to upload your files in your website.



By clicking on the url the website gives 403 forbidden error because contents of the bucket are not available for the public users.

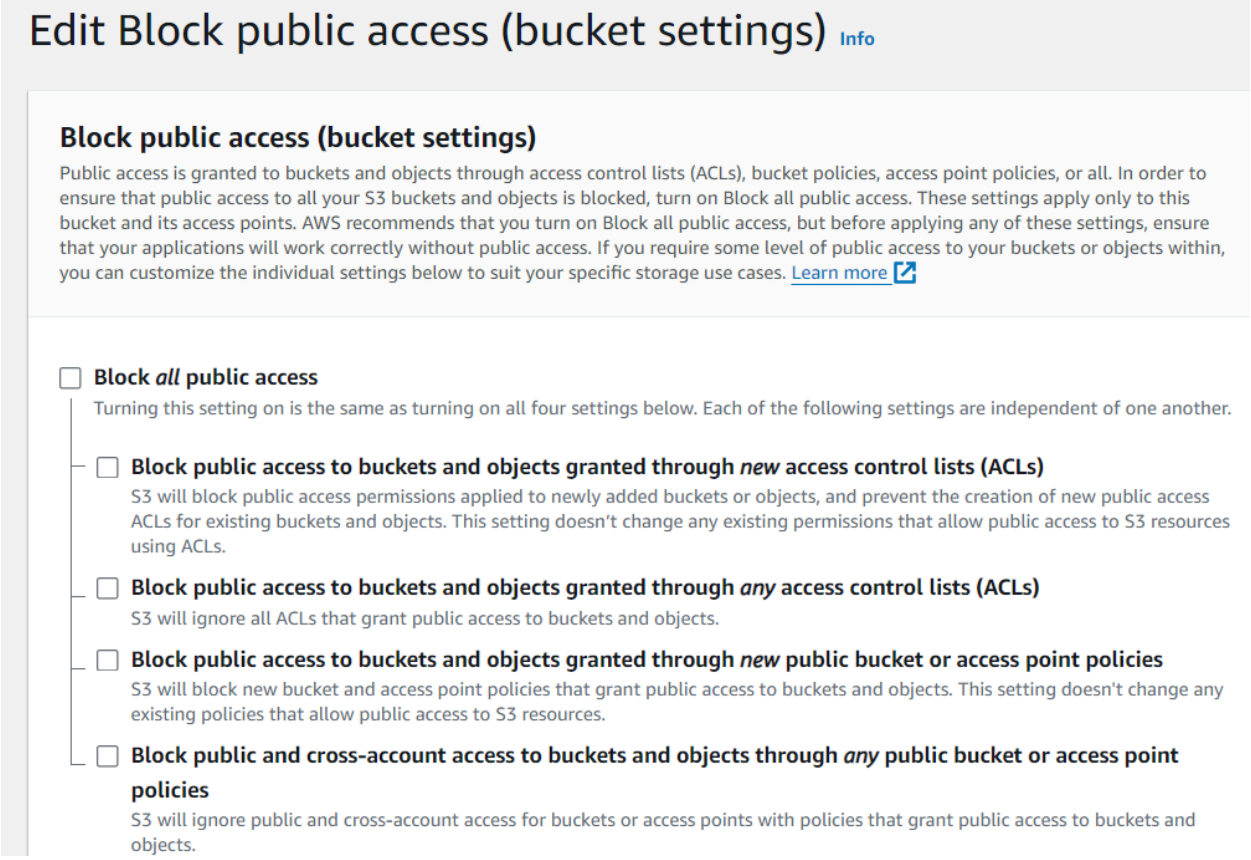
403 Forbidden

- Code: AccessDenied
- Message: Access Denied
- RequestId: Q3K0QMPPMMBQ8TCR
- HostId: 2GN7HpN25IH4RV1BD7b1+l8CCLIxbi7fU046fJj/GwR/dbv1BPlHEeAftyqGhVGVLGH6GrIxanU=

An Error Occurred While Attempting to Retrieve a Custom Error Document

- Code: AccessDenied
- Message: Access Denied

To change this, go to Permissions tab, go to Block public access and click on edit . Uncheck the block all public access checkbox and click save.



Update your bucket policy as follows and click save changes

Bucket policy

The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by oth

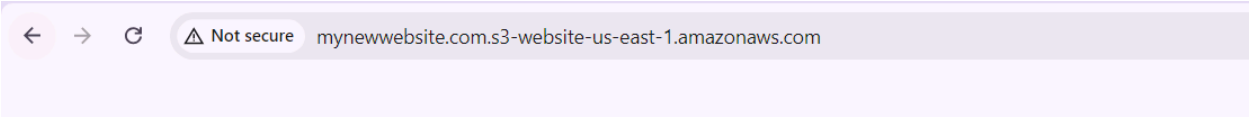
Bucket ARN

arn:aws:s3:::www.mynewwebsite.com

Policy

```
{
"Version": "2012-10-17",
"Statement": [
{
"Sid": "PublicReadGetObject",
"Effect": "Allow",
"Principal": {
"AWS": "*"
},
"Action": "s3:GetObject",
"Resource": "arn:aws:s3:::www.mynewwebsite.com/*"
}
]
}
```

Now after reloading your website, you will be able to see it.

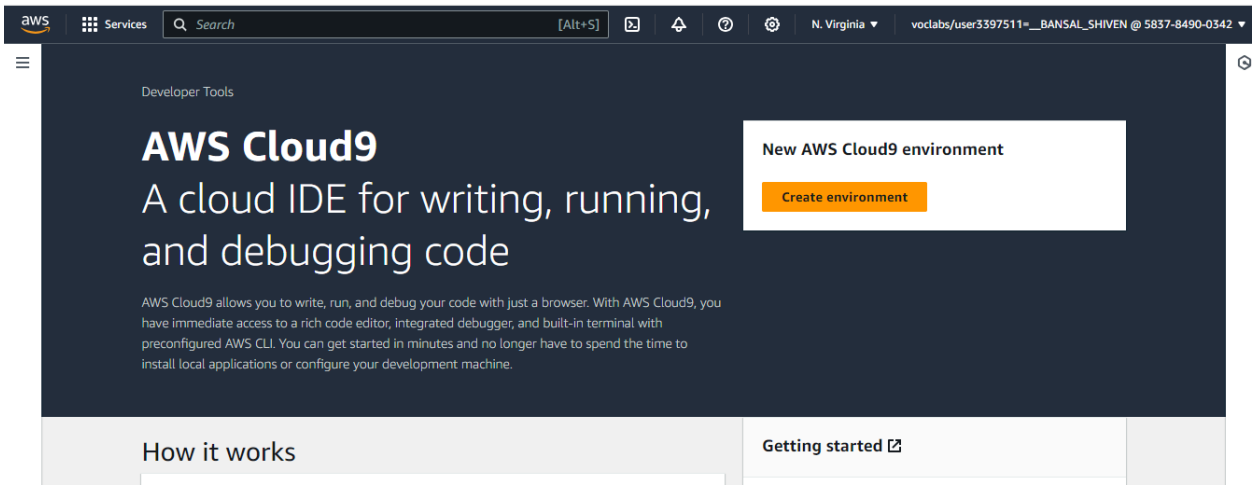


Hello World!

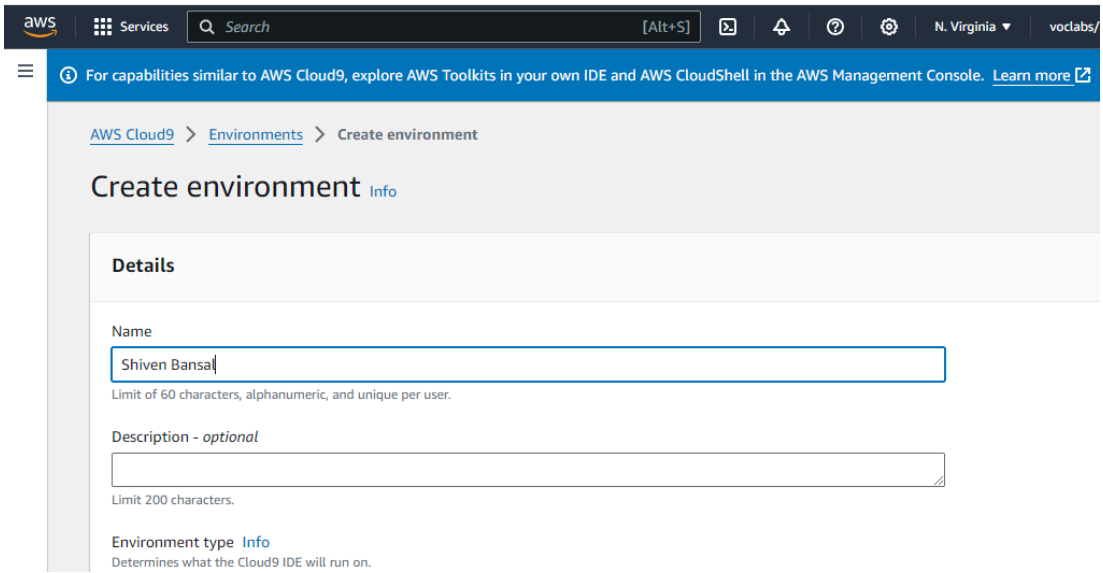
This website was created by Shiven Bansal

Experiment - 1B

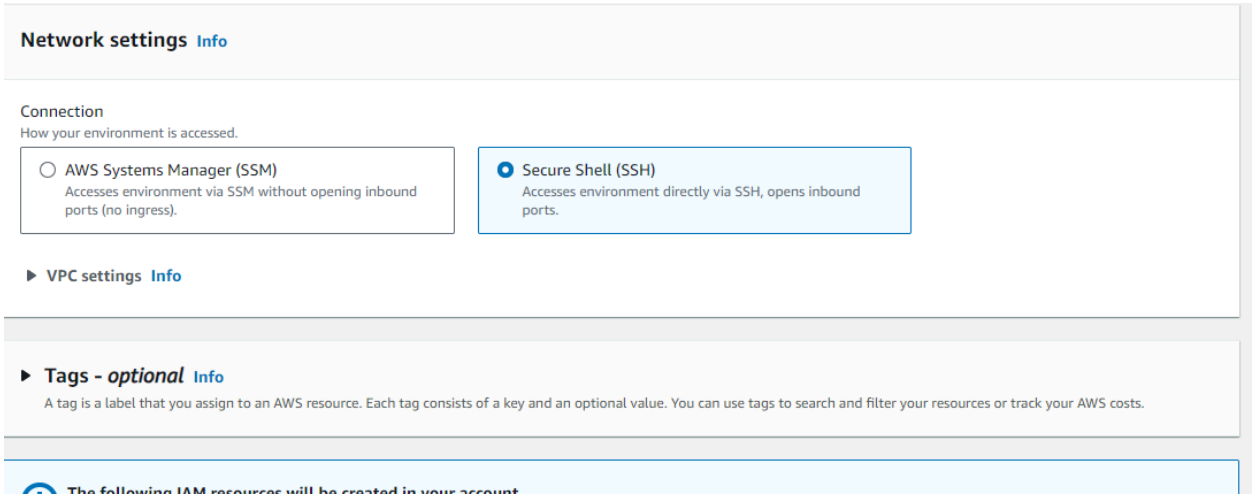
Open the AWS account and search for Cloud9. Click on create environment.



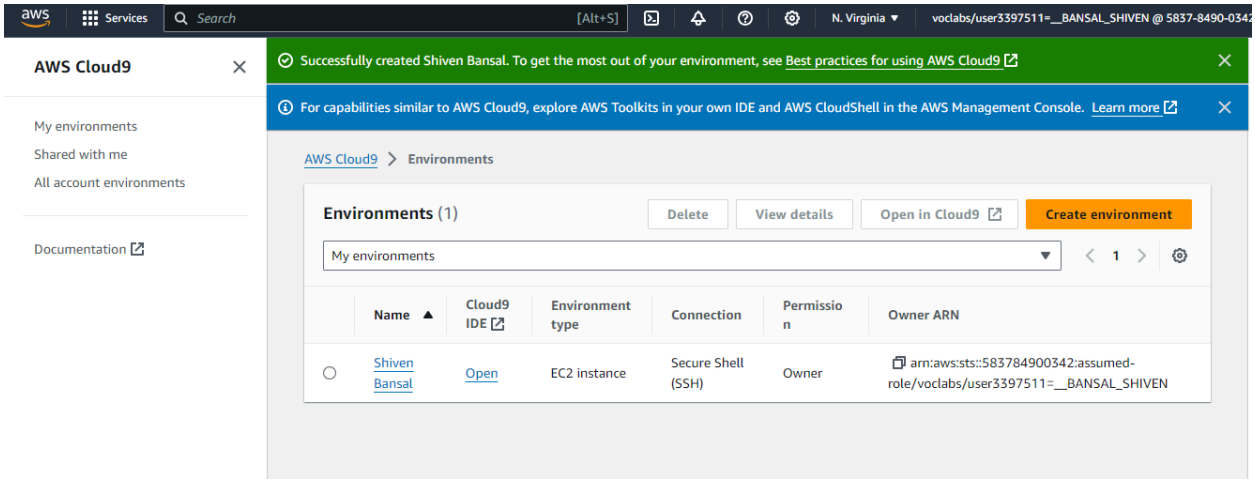
Enter the name and other required configuration for creating an environment. In network settings, using the AWS system manager gives an error while creating the environment. It states there was an error creating IAM resources needed for SSM.



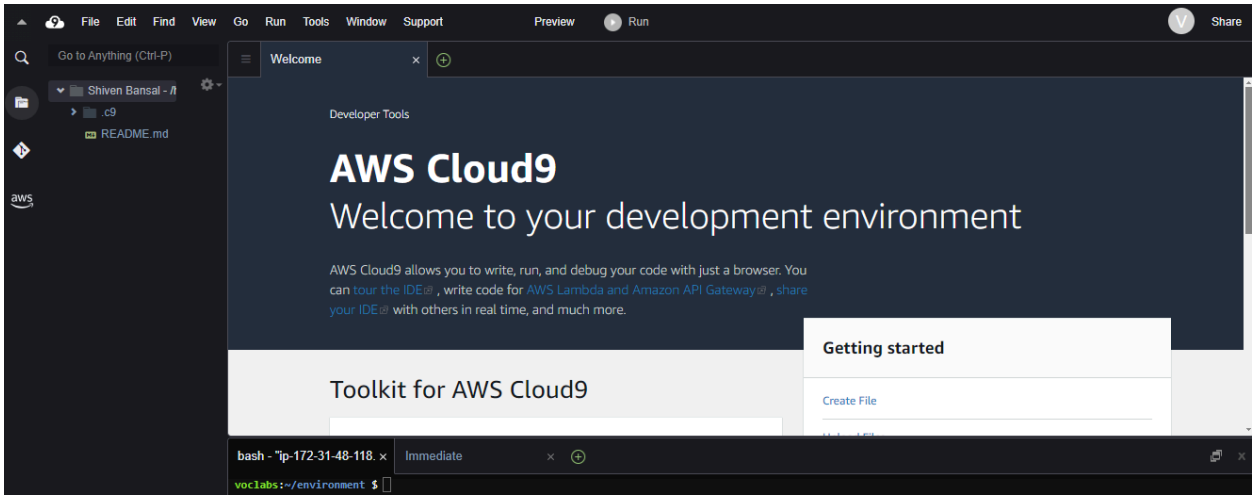
Use the Secure Shell option in Network settings



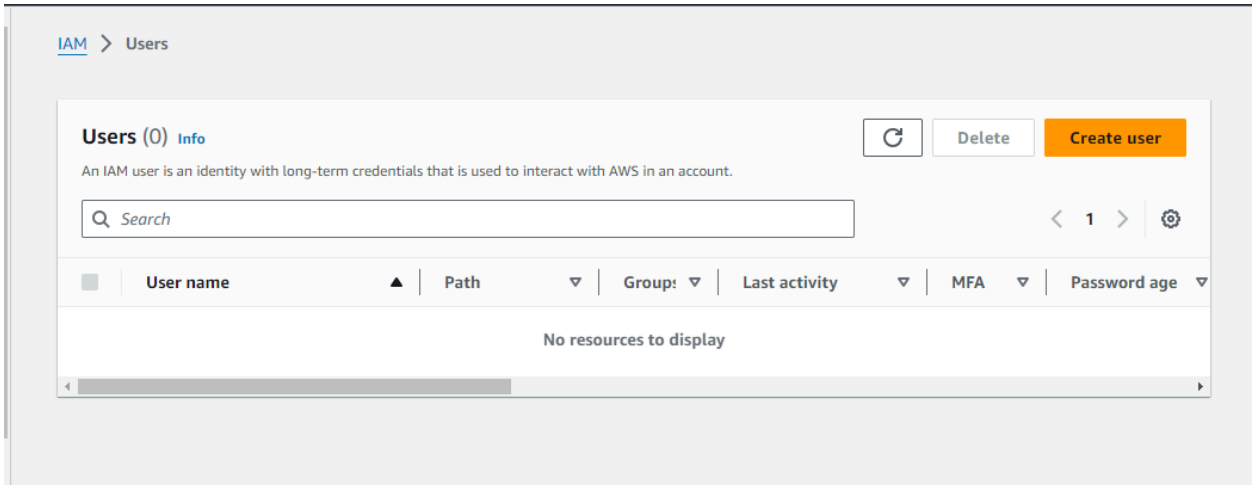
Once the configuration is complete, click on create environment to create a Cloud9 environment.



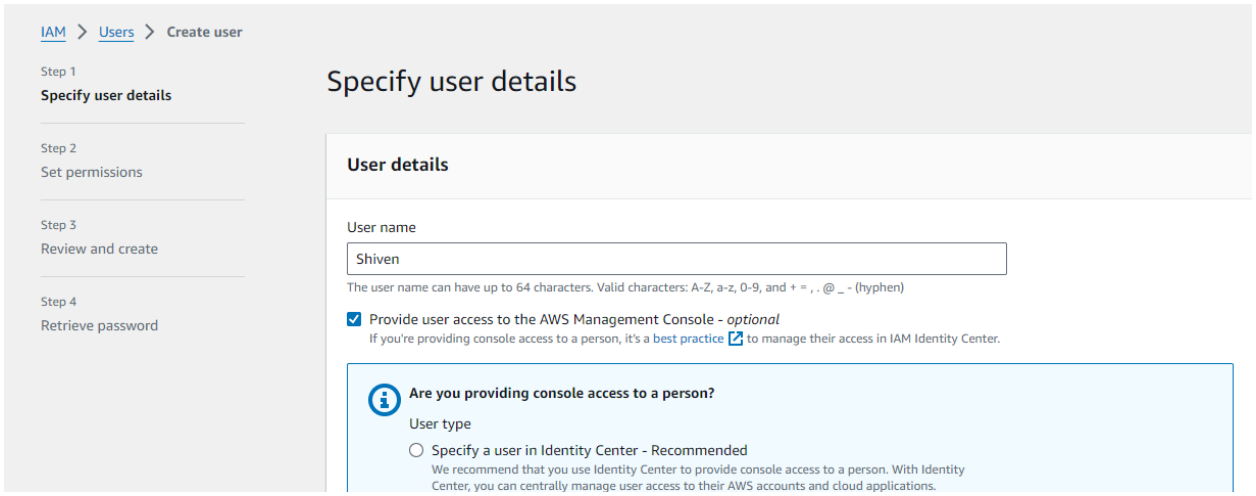
Click on the environment name to open the created Cloud9 Environment.

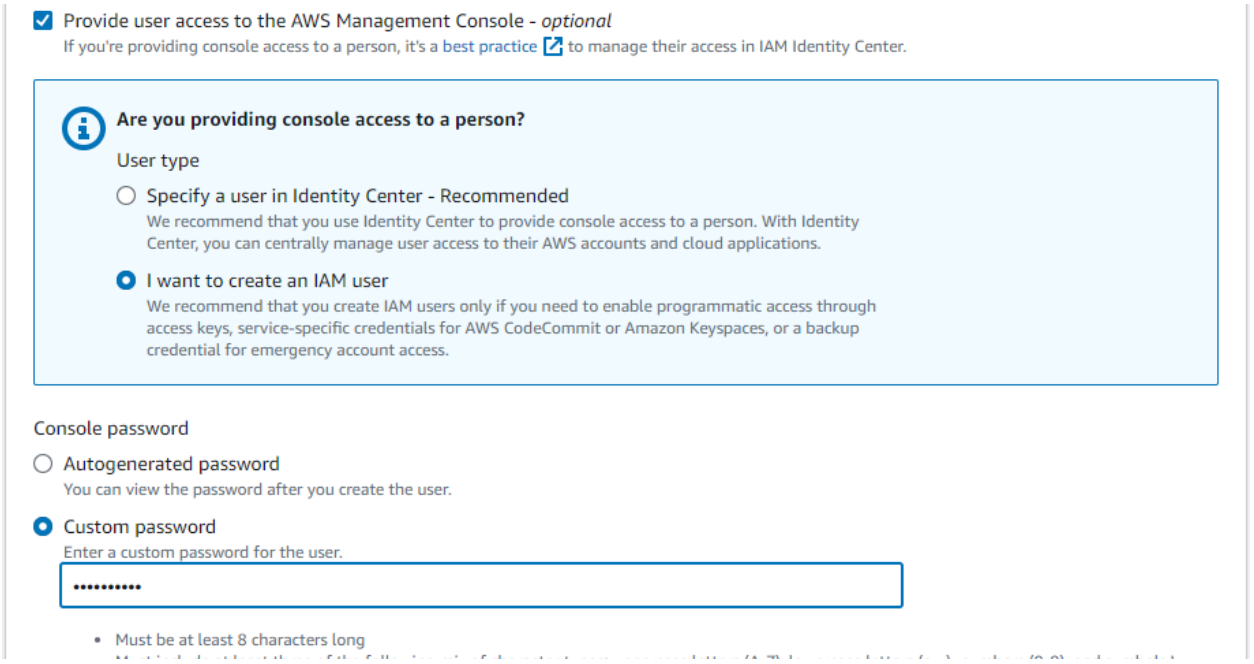


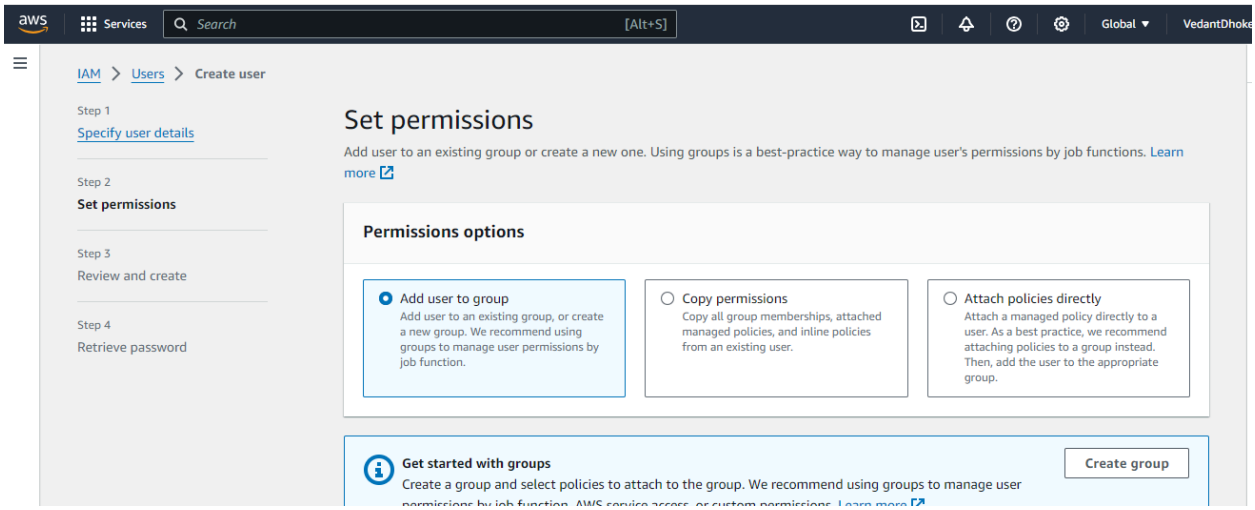
Open the aws account and search for IAM service.Then go to users tab and click on create user to create a new user.



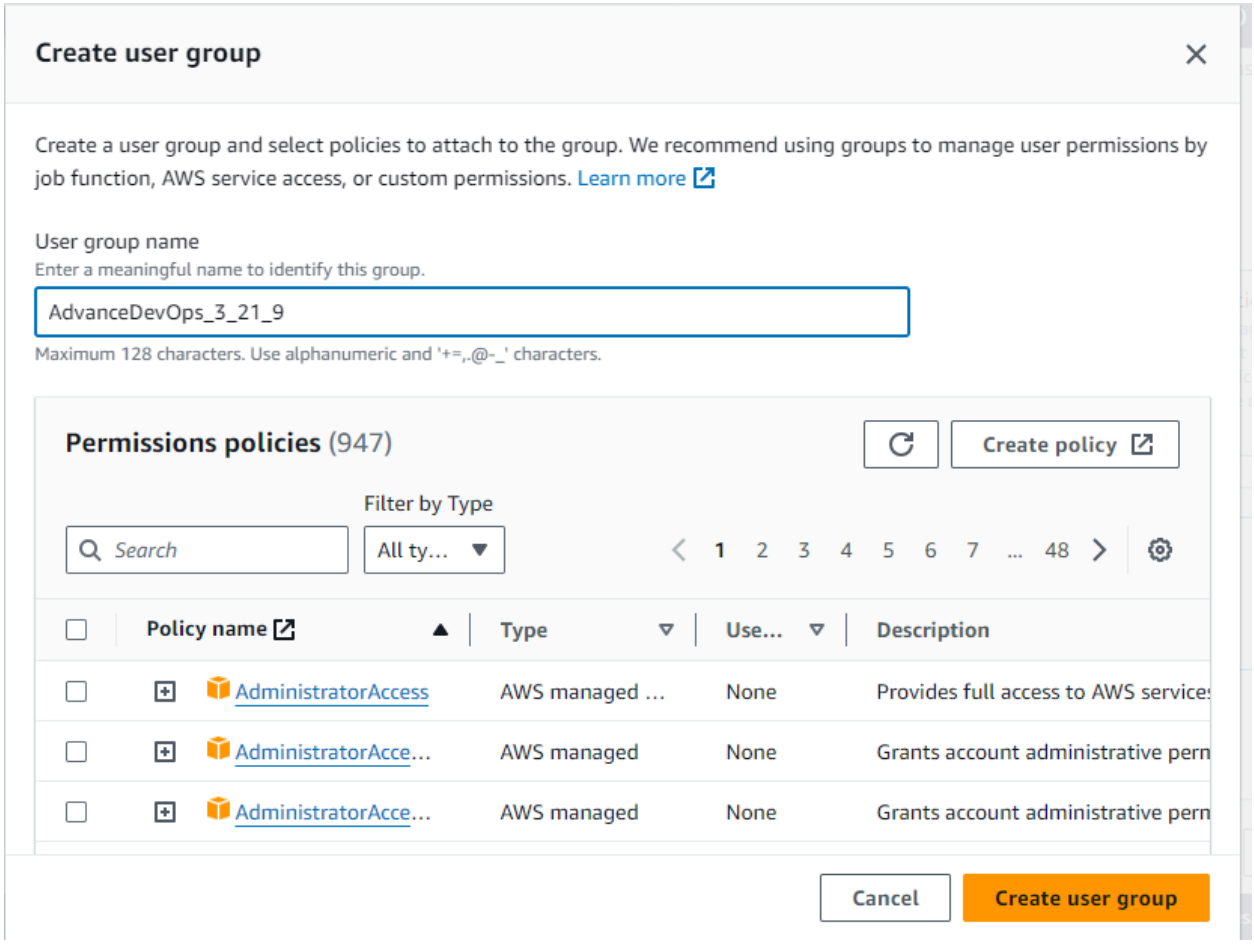
Give the user name, select the “provide user access” checkbox. Also select the option “I want to create an IAM user”. Otherwise we will have to enable the Identity center and specify a user there.


IAM > Users > Create user
Step 1
Specify user details
Step 2
Set permissions
Step 3
Review and create
Step 4
Retrieve password
Specify user details
User details
User name
Shiven
The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , . @ _ - (hyphen)
Provide user access to the AWS Management Console - optional
If you're providing console access to a person, it's a best practice to manage their access in IAM Identity Center.
Are you providing console access to a person?
User type
Specify a user in Identity Center - Recommended
We recommend that you use Identity Center to provide console access to a person. With Identity Center, you can centrally manage user access to their AWS accounts and cloud applications.

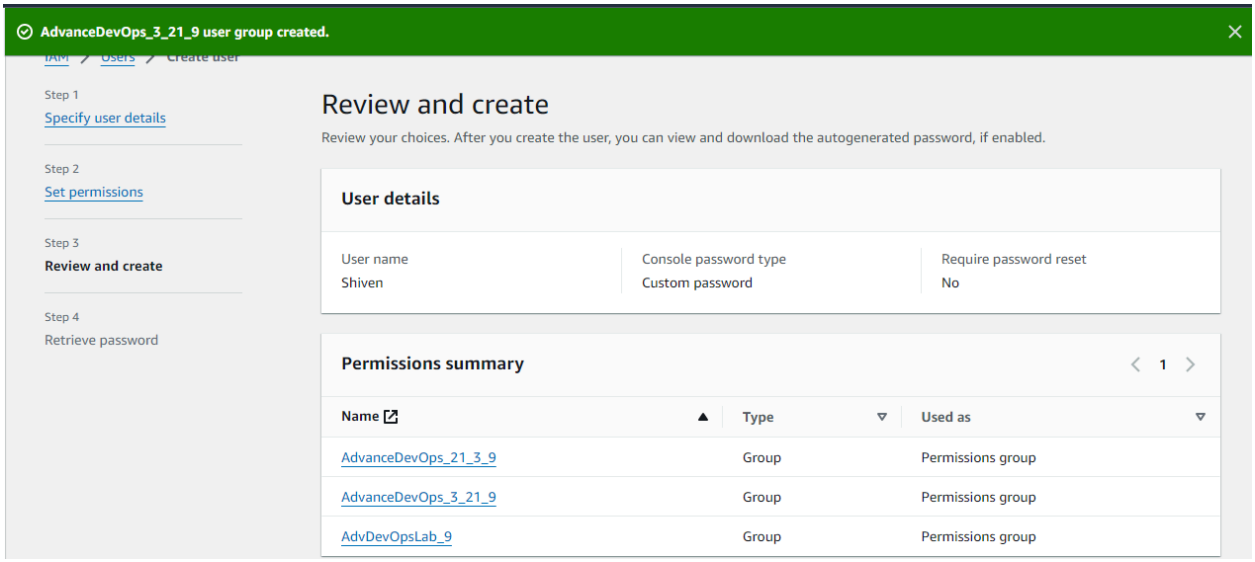

Provide user access to the AWS Management Console - optional
If you're providing console access to a person, it's a best practice to manage their access in IAM Identity Center.
Are you providing console access to a person?
User type
Specify a user in Identity Center - Recommended
We recommend that you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



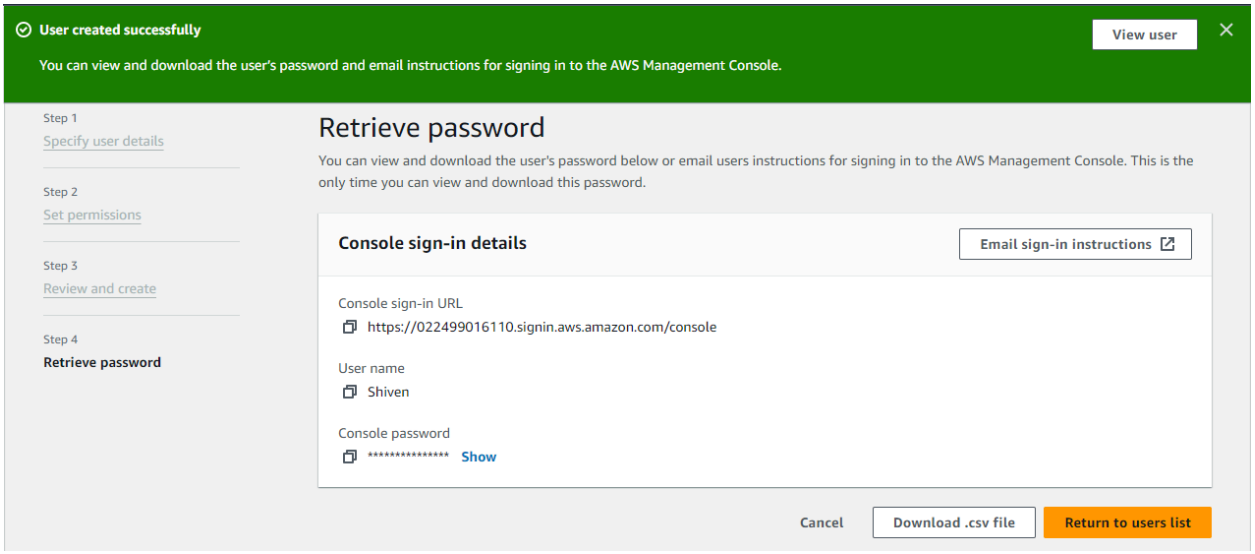
Next click on add user to group. If you do not have a existing group, select create group. Then Give the group name and policies if required, and create a group.



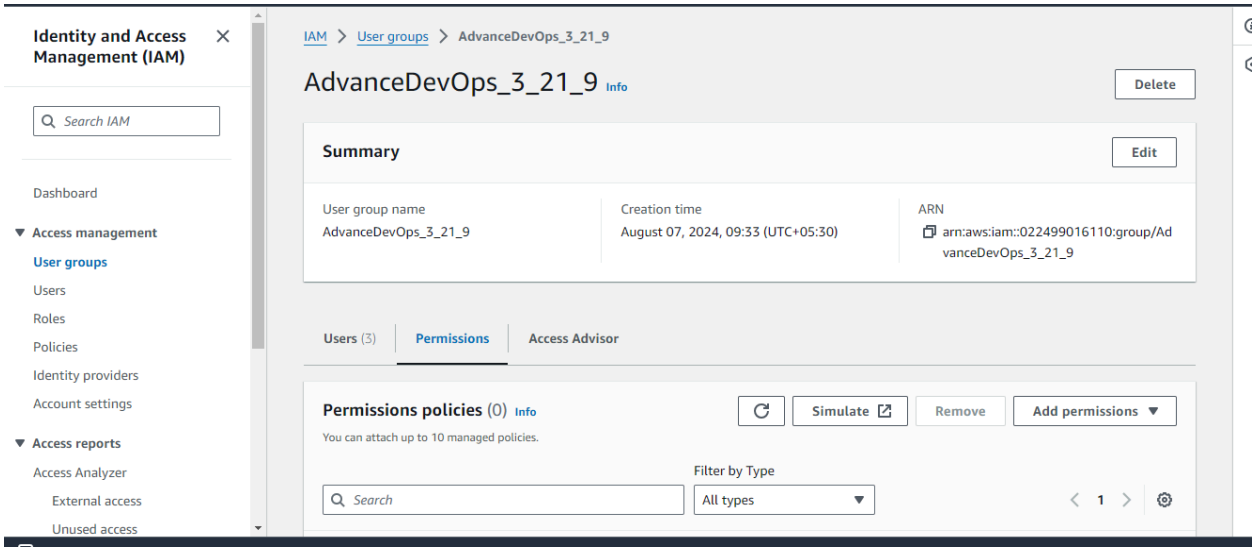
Once the group is created, select the group in which the user should be added.



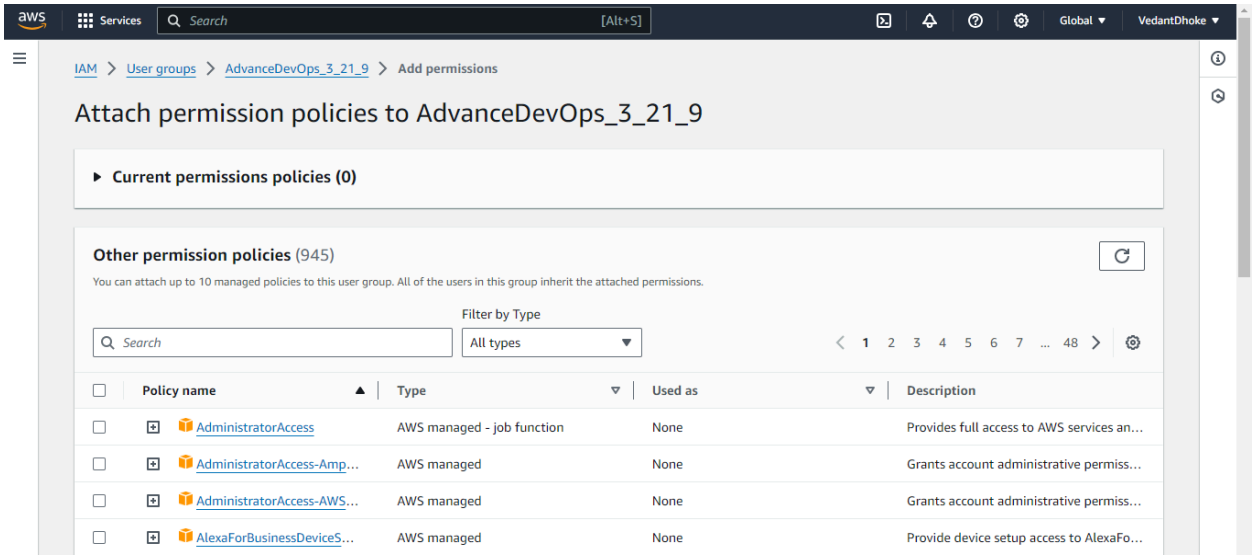
Recheck all the configuration and details of the user and click on create user.Then you will this page.

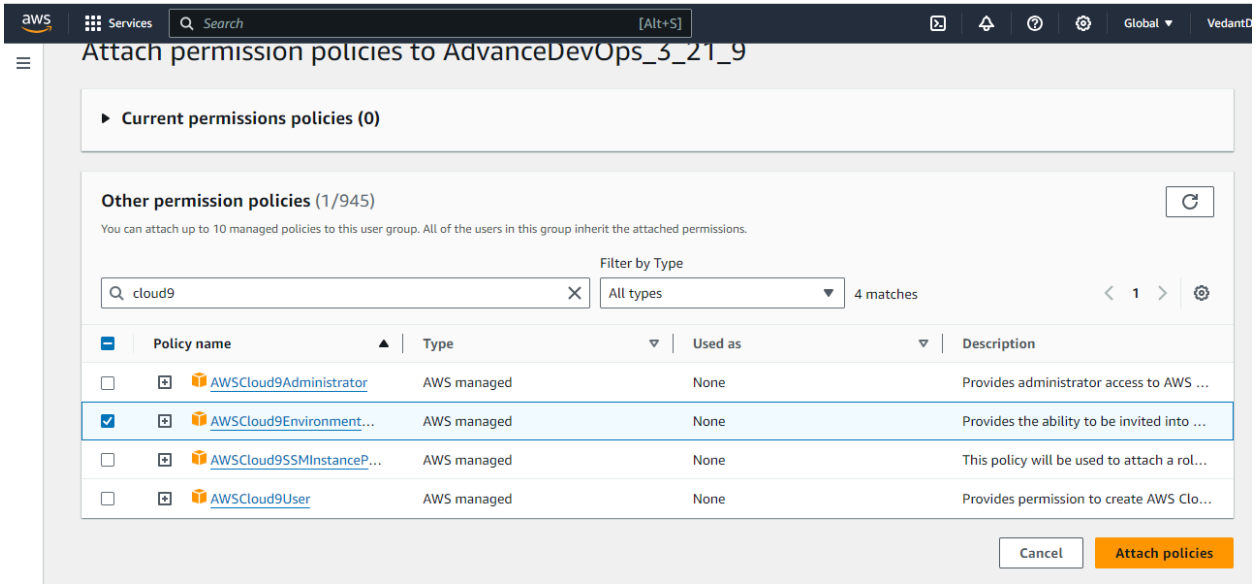


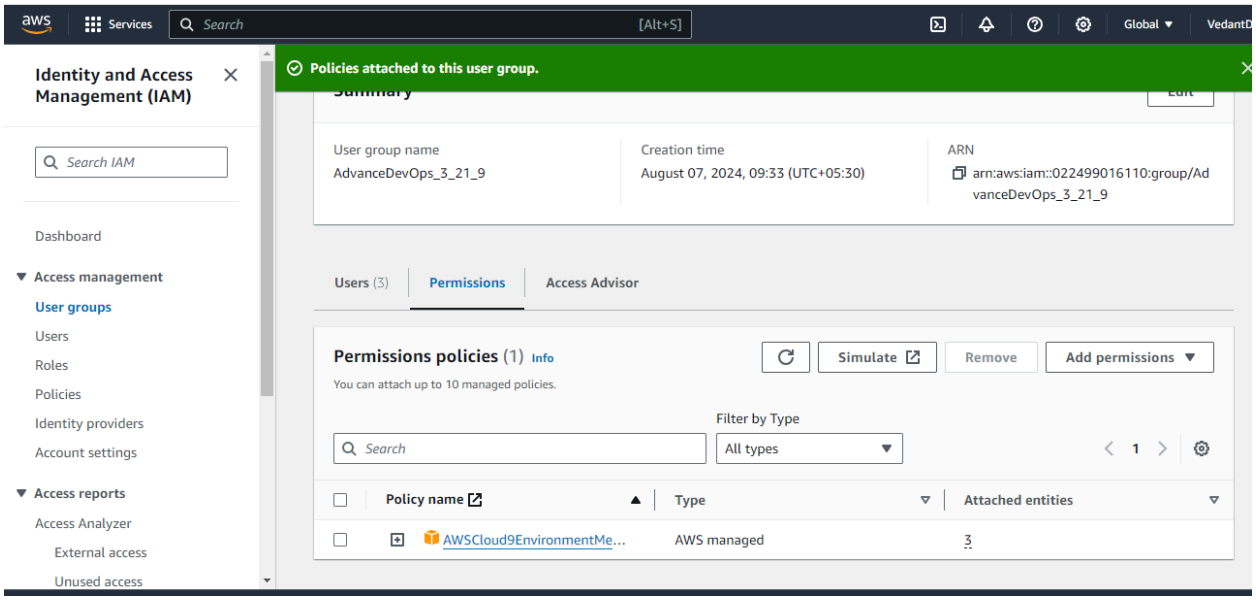
After creation of the user, go to the user groups tab and select the group in which the user has been added. Navigate to the permissions tab. Click on add permissions and attach policy.



Search for the “AWSCloud9EnvironmentMember” policy and attach it.



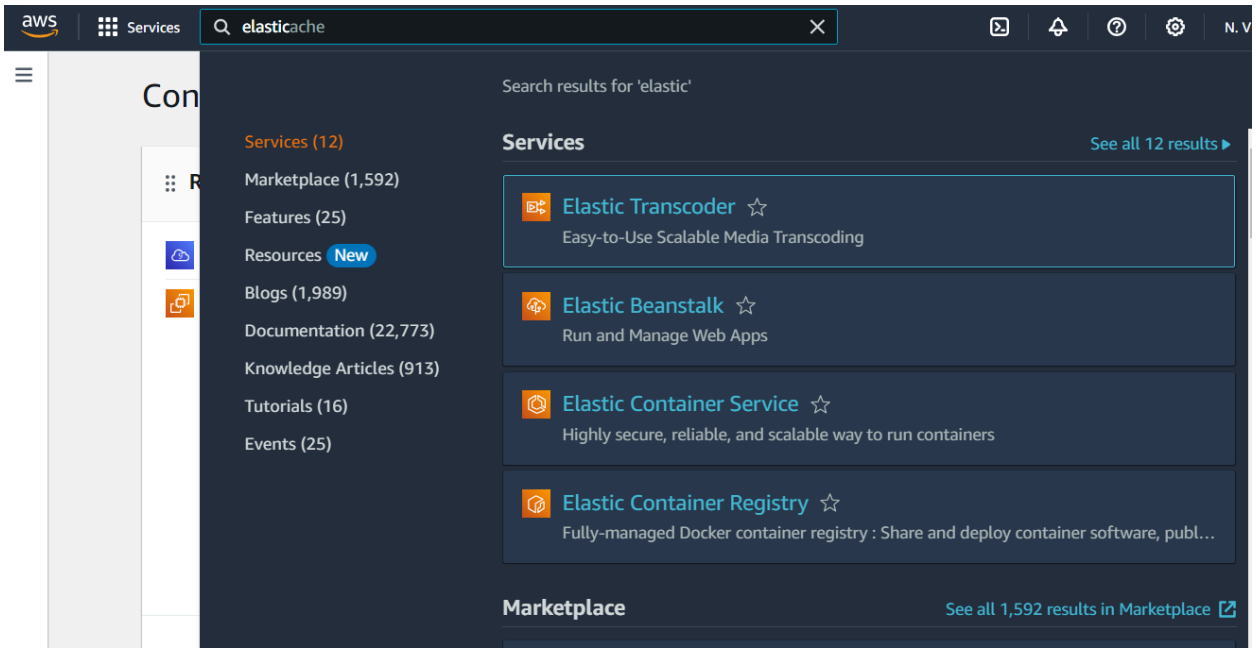

Services
Search
[Alt+S]
Global
Attach permission policies to AdvanceDevOps_3_21_9
Current permissions policies (0)
Other permission policies (1/945)
You can attach up to 10 managed policies to this user group. All of the users in this group inherit the attached permissions.
Filter by Type
cloud9
All types
4 matches
Policy name
Type
Used as
Description
AWSCloud9Administrator
AWS managed
None
Provides administrator access to AWS ...
AWSCloud9Environment...
AWS managed
None
Provides the ability to be invited into ...
AWSCloud9SSMInstanceP...
AWS managed
None
This policy will be used to attach a rol...
AWSCloud9User
AWS managed
None
Provides permission to create AWS Clo...
Cancel
Attach policies


Services
Search
[Alt+S]
Global
Identity and Access Management (IAM)
Search IAM
Dashboard
Access management
User groups
Users
Roles
Policies
Identity providers
Account settings
Access reports
Access Analyzer
External access
Unused access
Policies attached to this user group.
User group name
AdvanceDevOps_3_21_9
Creation time
August 07, 2024, 09:33 (UTC+05:30)
ARN
arn:aws:iam::022499016110:group/AdvanceDevOps_3_21_9
Users (3)
Permissions
Access Advisor
Permissions policies (1) Info
You can attach up to 10 managed policies.
Simulate
Remove
Add permissions
Filter by Type
Search
All types
Policy name
Type
Attached entities
AWSCloud9EnvironmentMe...
AWS managed
3

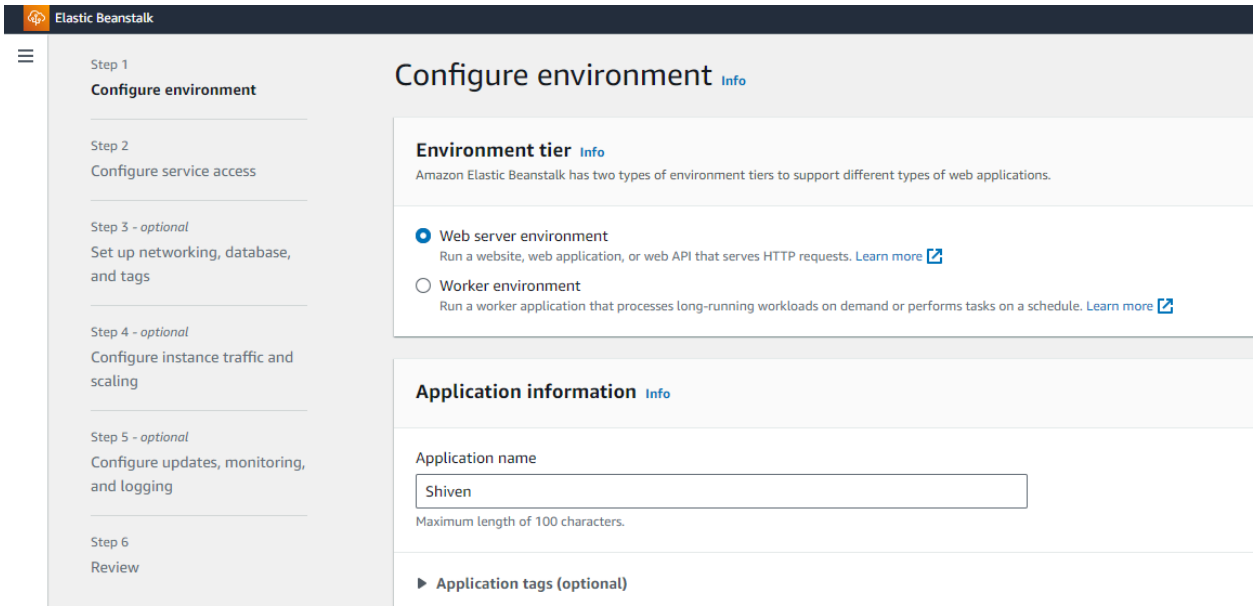
Name: Shiven Bansal **Roll no : 03** **Class: D15C**

Experiment-2

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

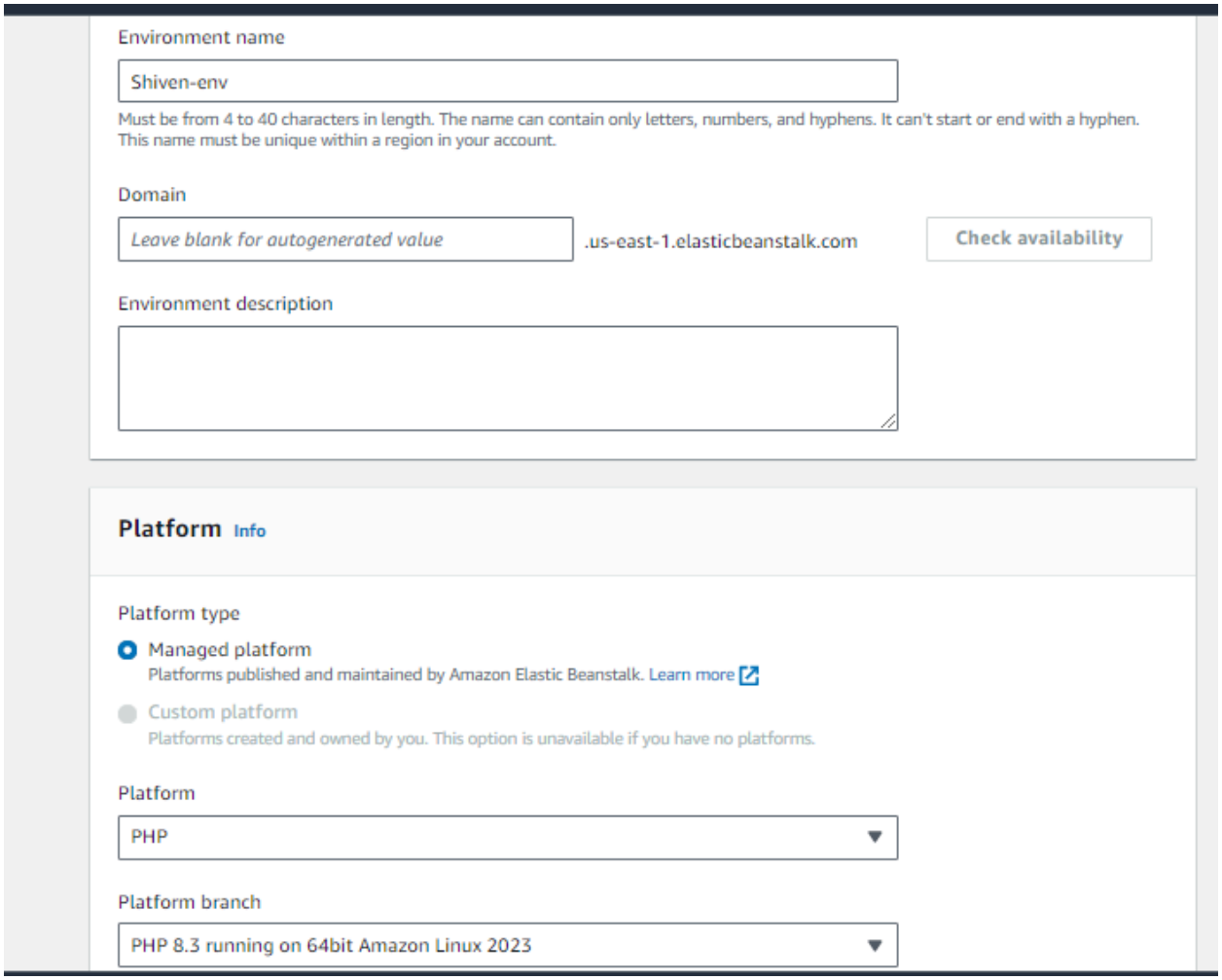


Step 2: Go to Elastic Beanstalk and click on Create Application



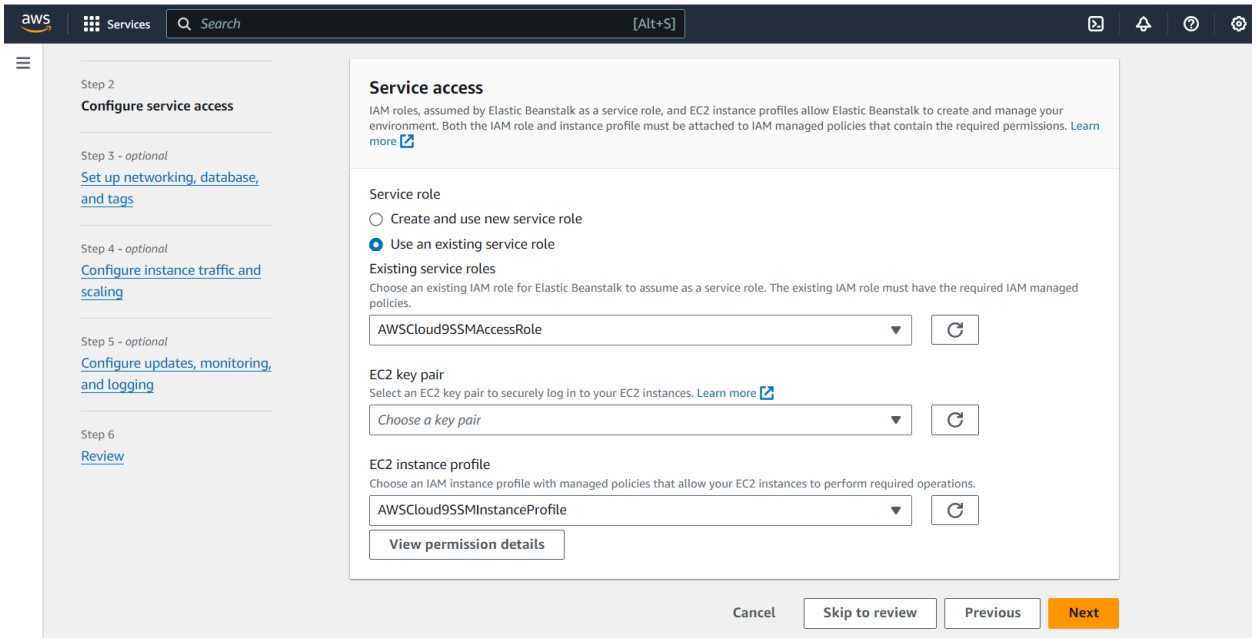
Name: Shiven Bansal **Roll no : 03** **Class: D15C**

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.

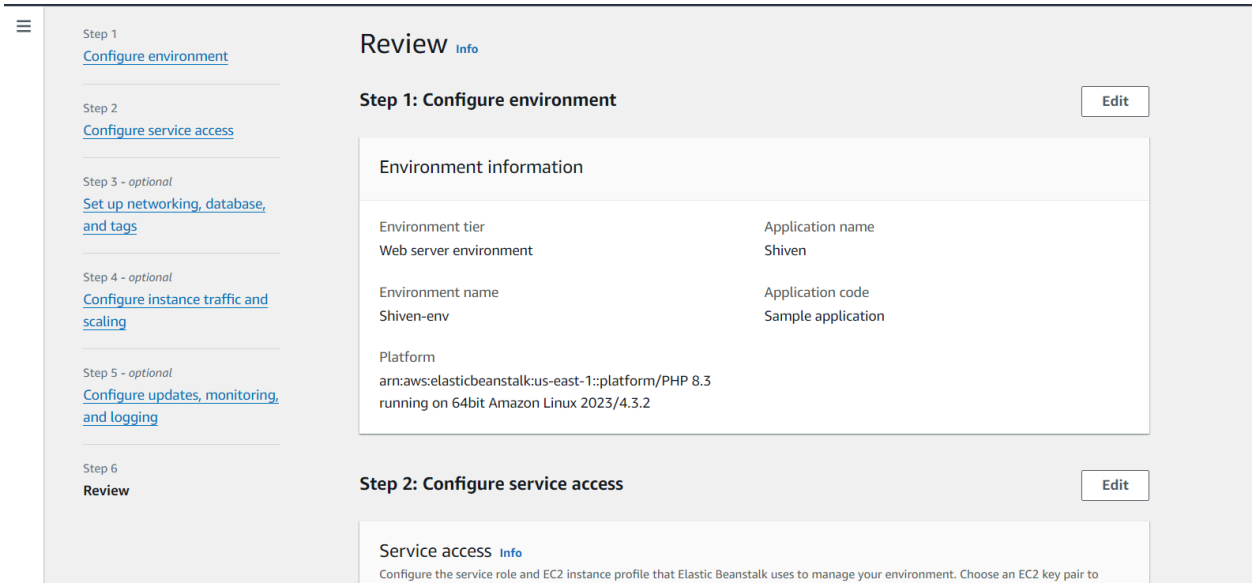


Name: Shiven Bansal **Roll no : 03** **Class: D15C**

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

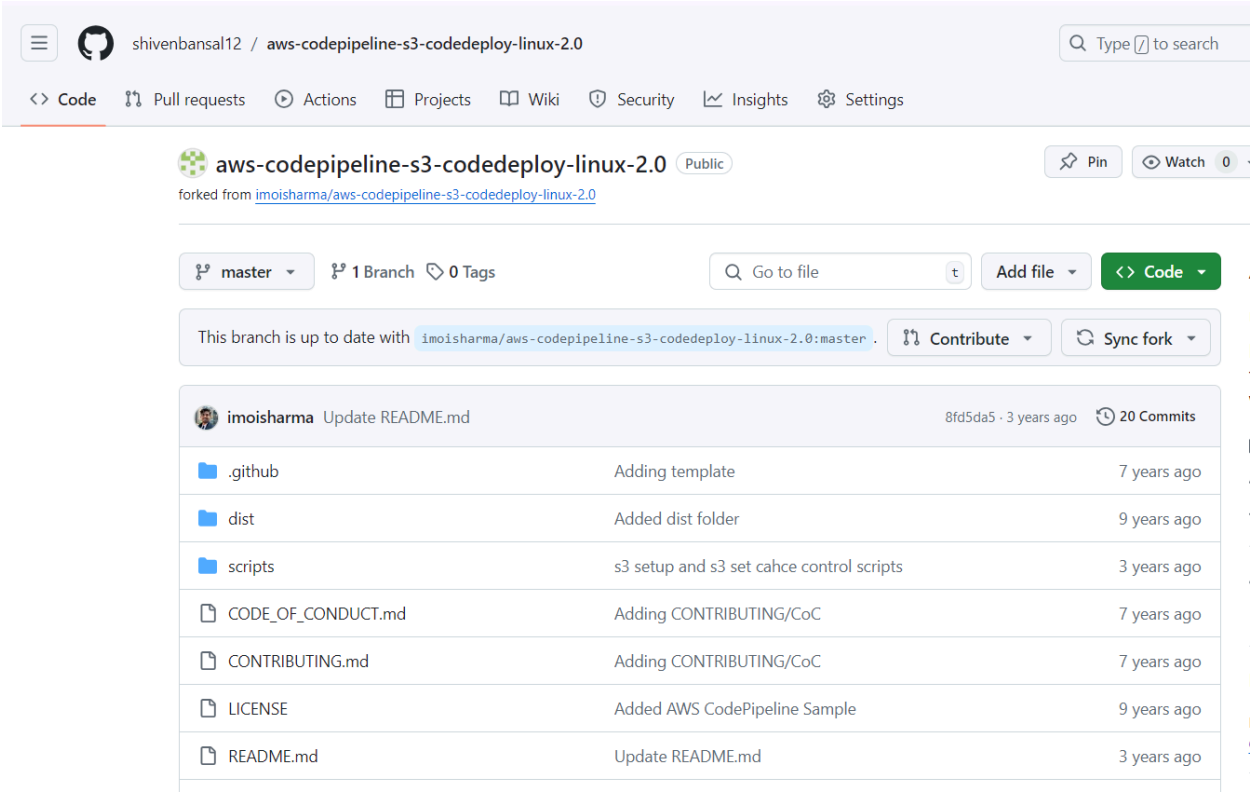


Step 5: Review the settings that you have set up for your application and submit your application.



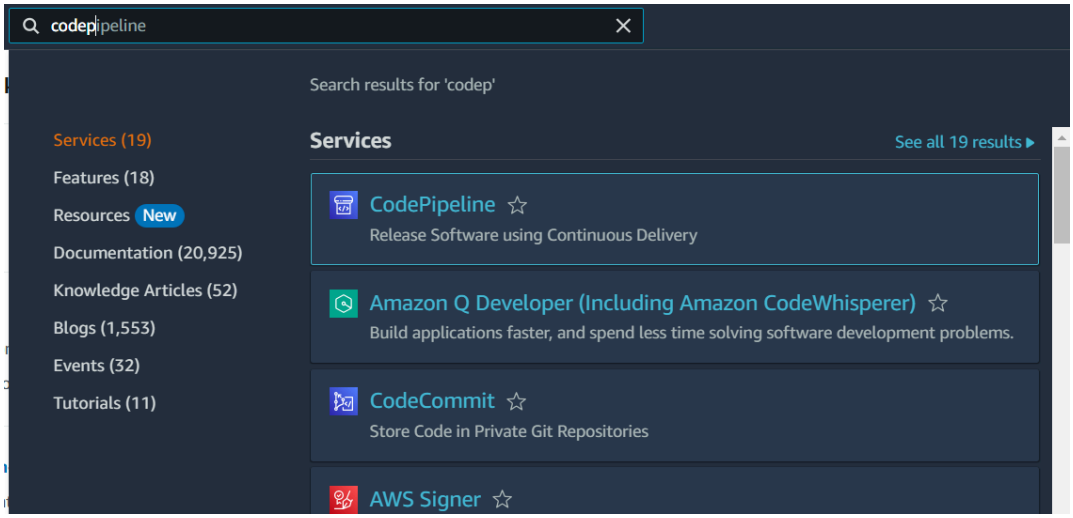
Name: Shiven Bansal **Roll no : 03** **Class: D15C**

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

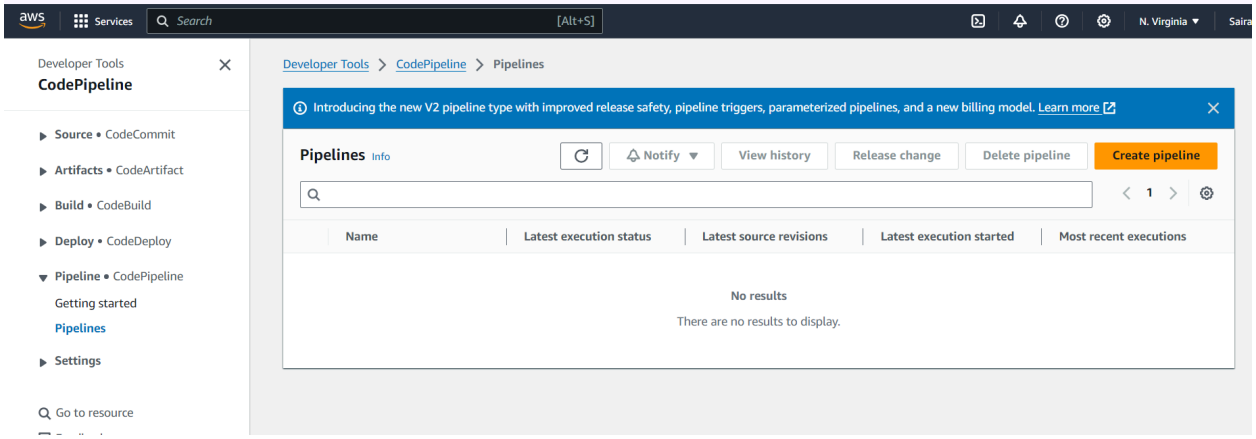


Step 7: Search CodePipeline in the services tab and click on it.

Name: Shiven Bansal **Roll no : 03** **Class: D15C**

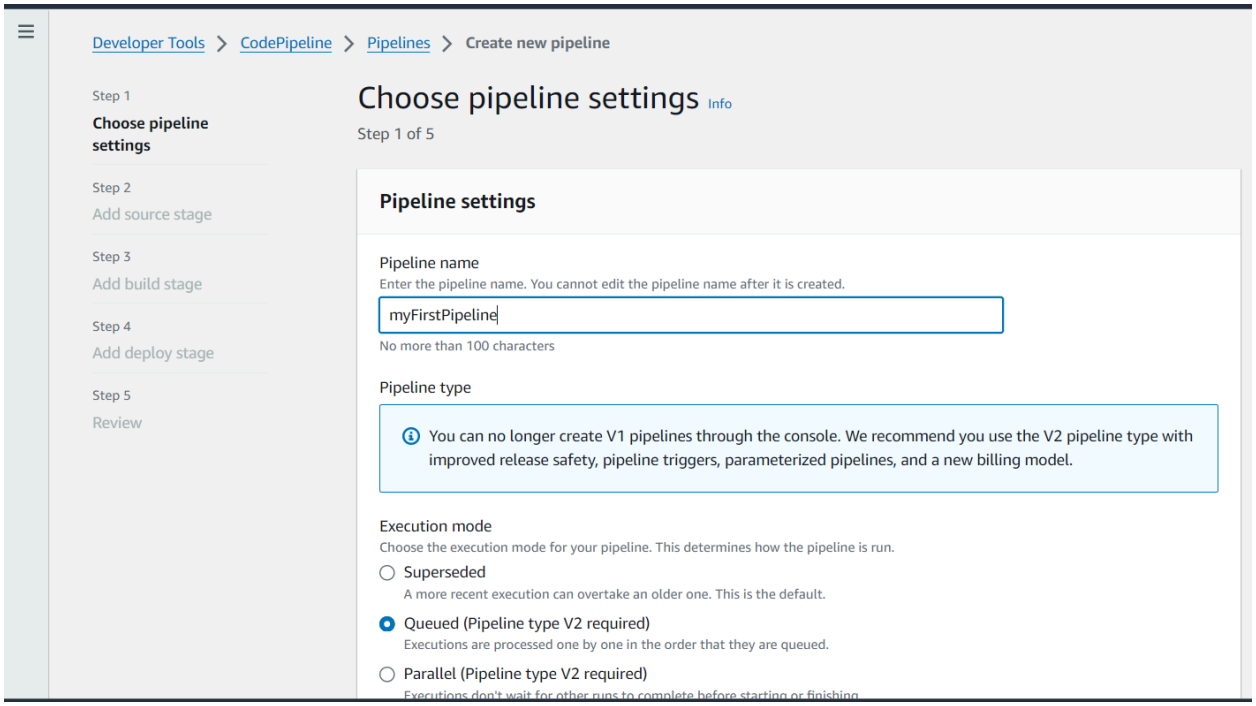


Step 8: Click on Create Pipeline.

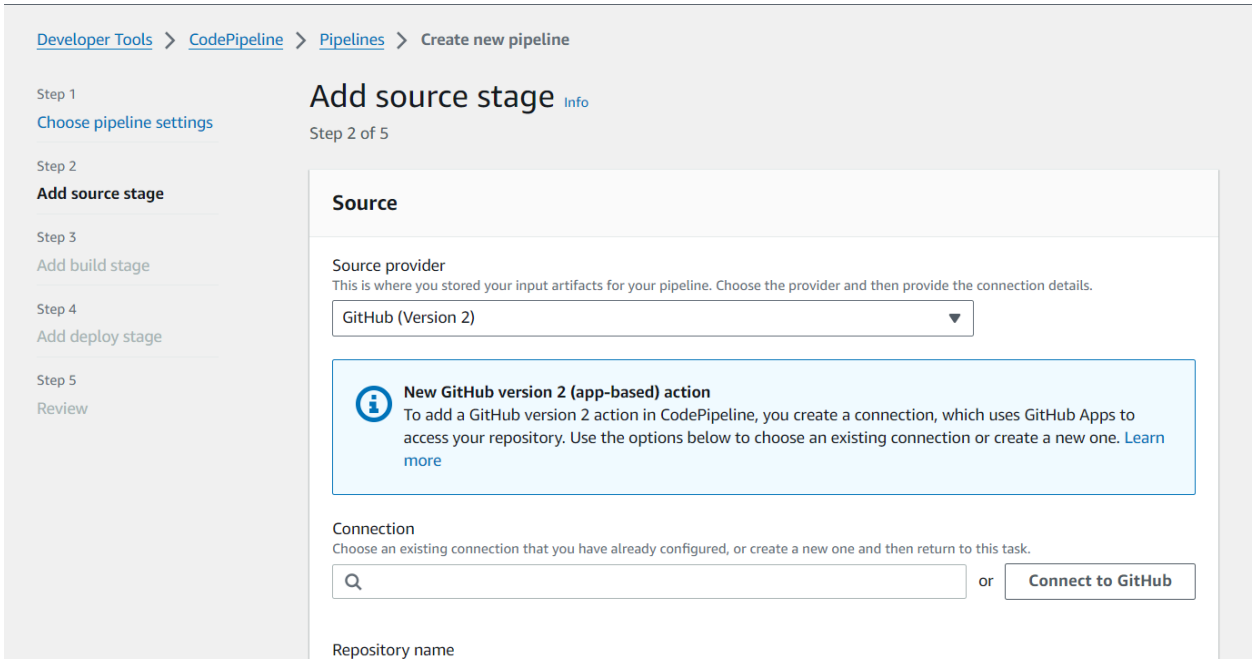


Step 9: Give a name to your Pipeline. A new service role would be created with the name of the Pipeline

Name: Shiven Bansal **Roll no : 03** **Class: D15C**

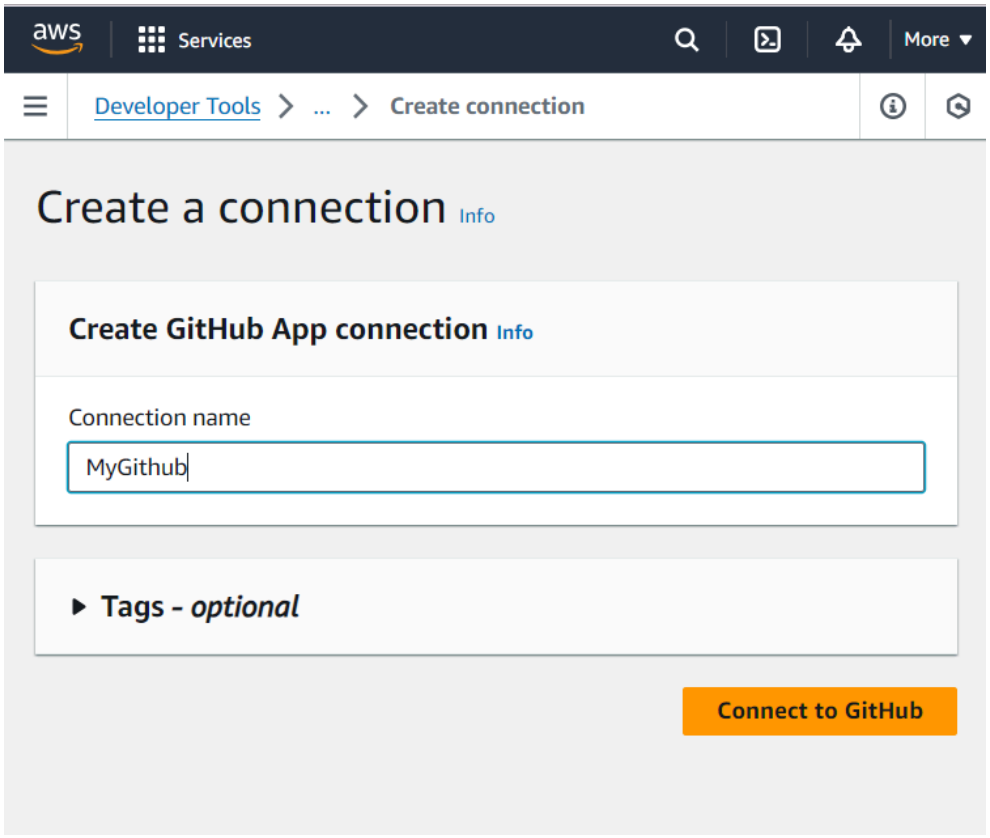


Step 10: Select a source provider (as Github (Version 2)). Click on Connect to Github to connect your github.

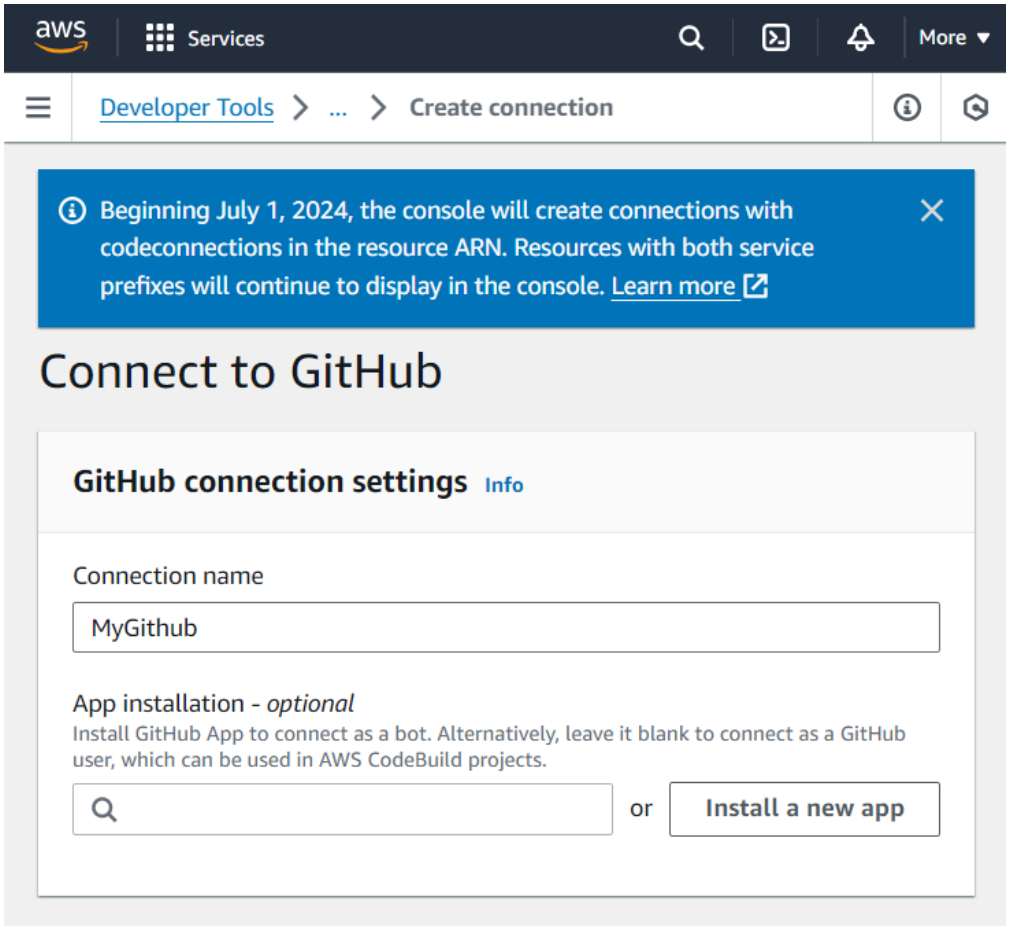


Name: Shiven Bansal **Roll no : 03** **Class: D15C**

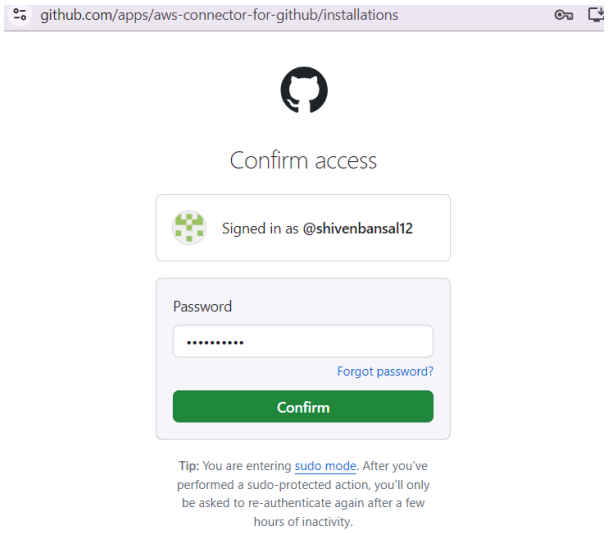
Step 11: Give a name to your GitHub app Connection and click on Connect. This will give you a prompt to either to select a GitHub app or install a new app. If this is your first time, click on Install a new app.



Name: Shiven Bansal **Roll no : 03** **Class: D15C**

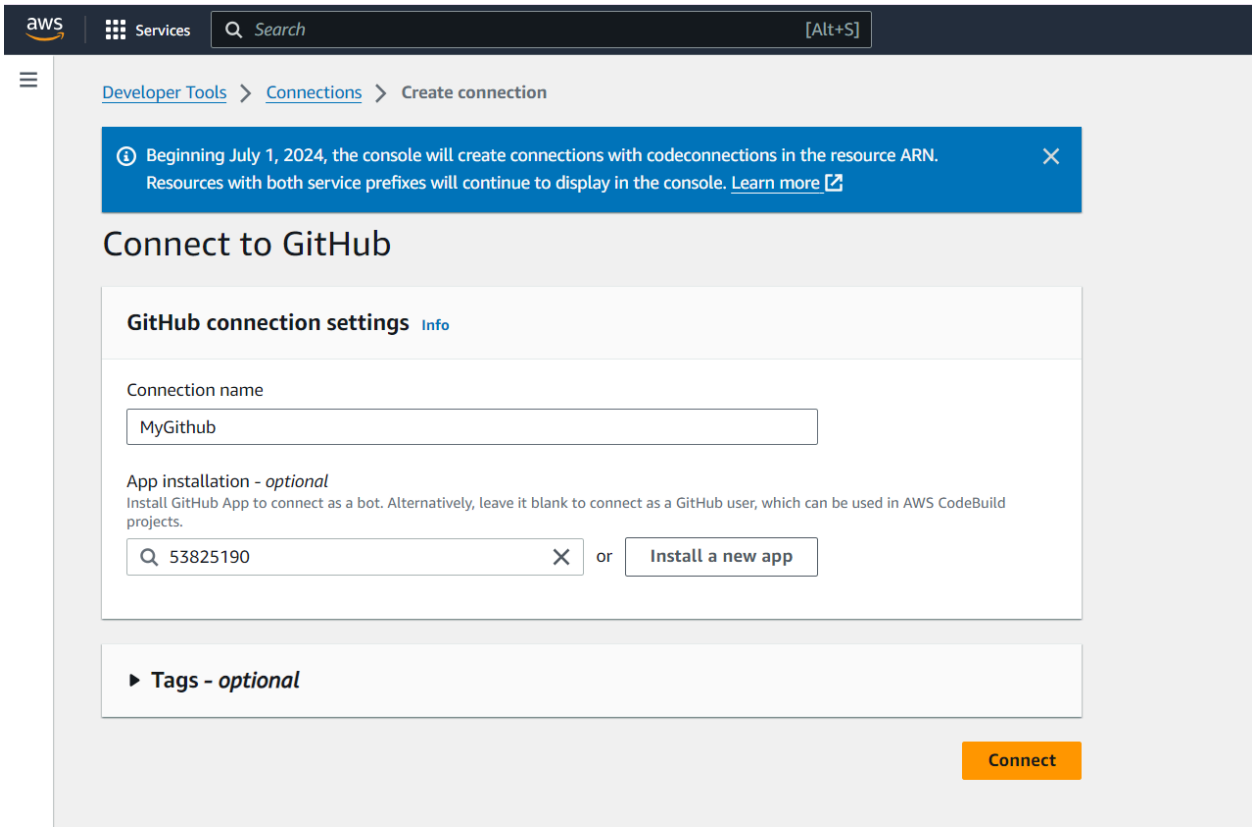


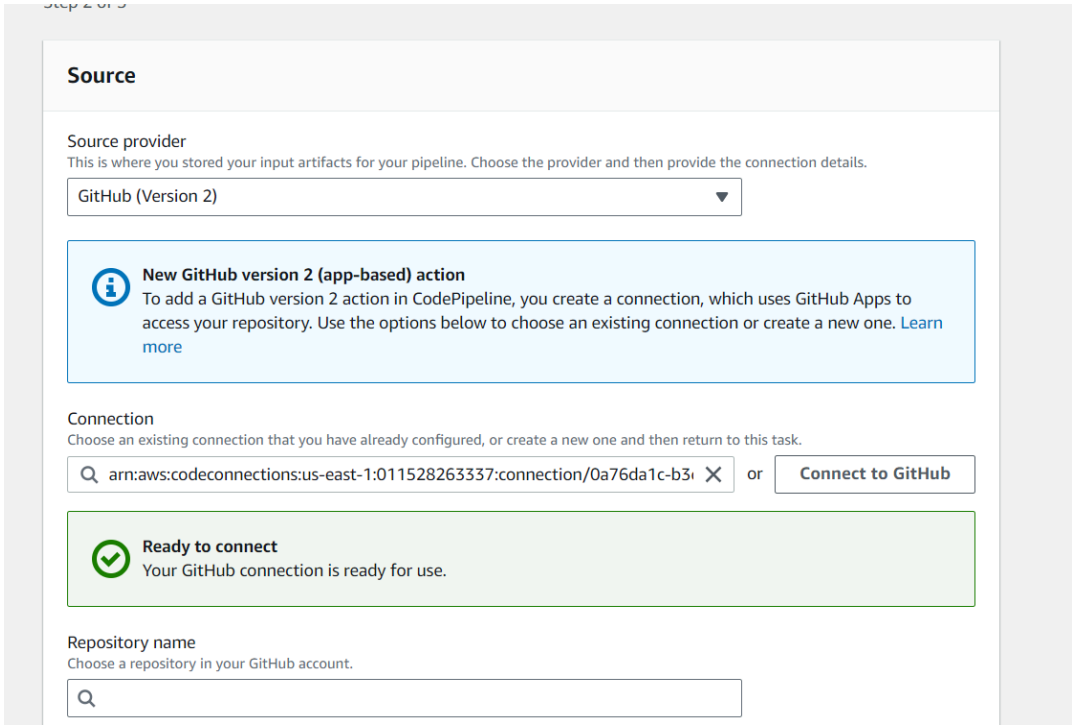
Step 12: This will direct you to install AWS Connector on your GitHub. Install it to your account and give it its permissions



Name: Shiven Bansal **Roll no : 03** **Class: D15C**

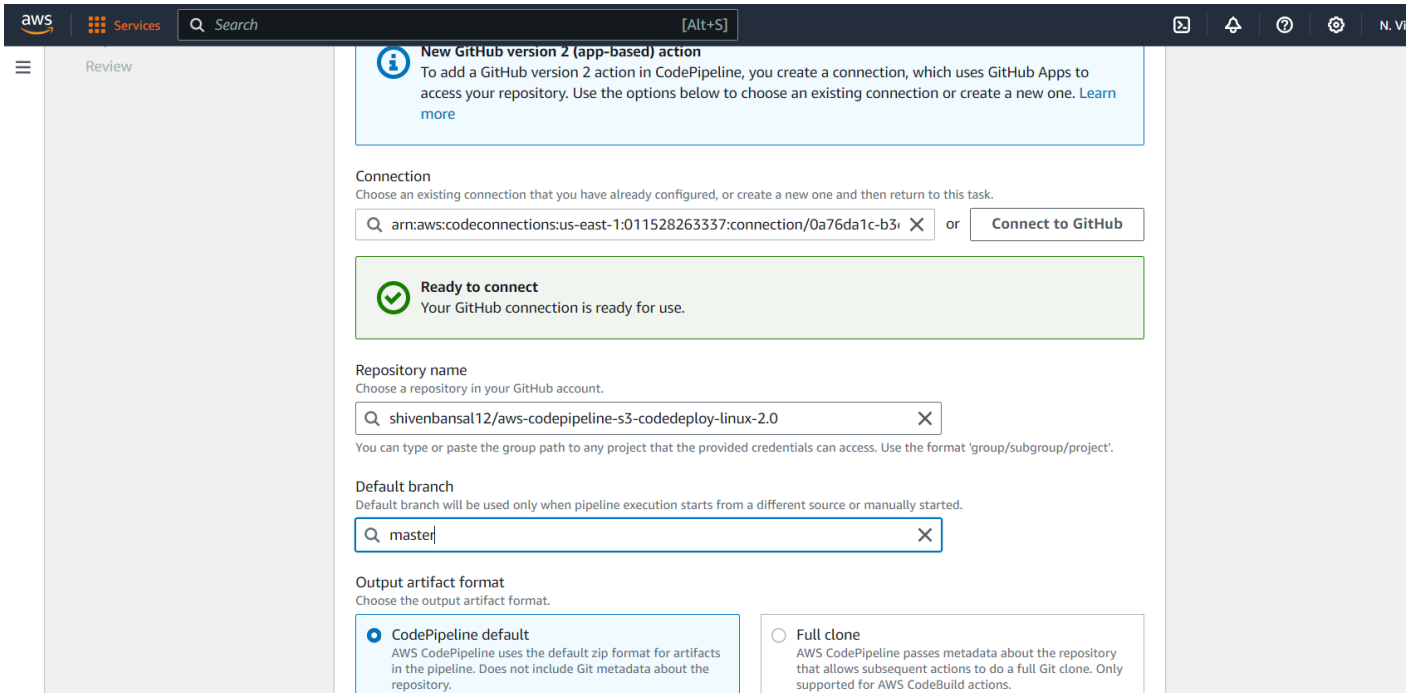
Step 13: After the app is set up, it gives the number in the text field. Click on Connect. After clicking on connect, the link is shown in the connection field and AWS shows that GitHub connection is ready to use.



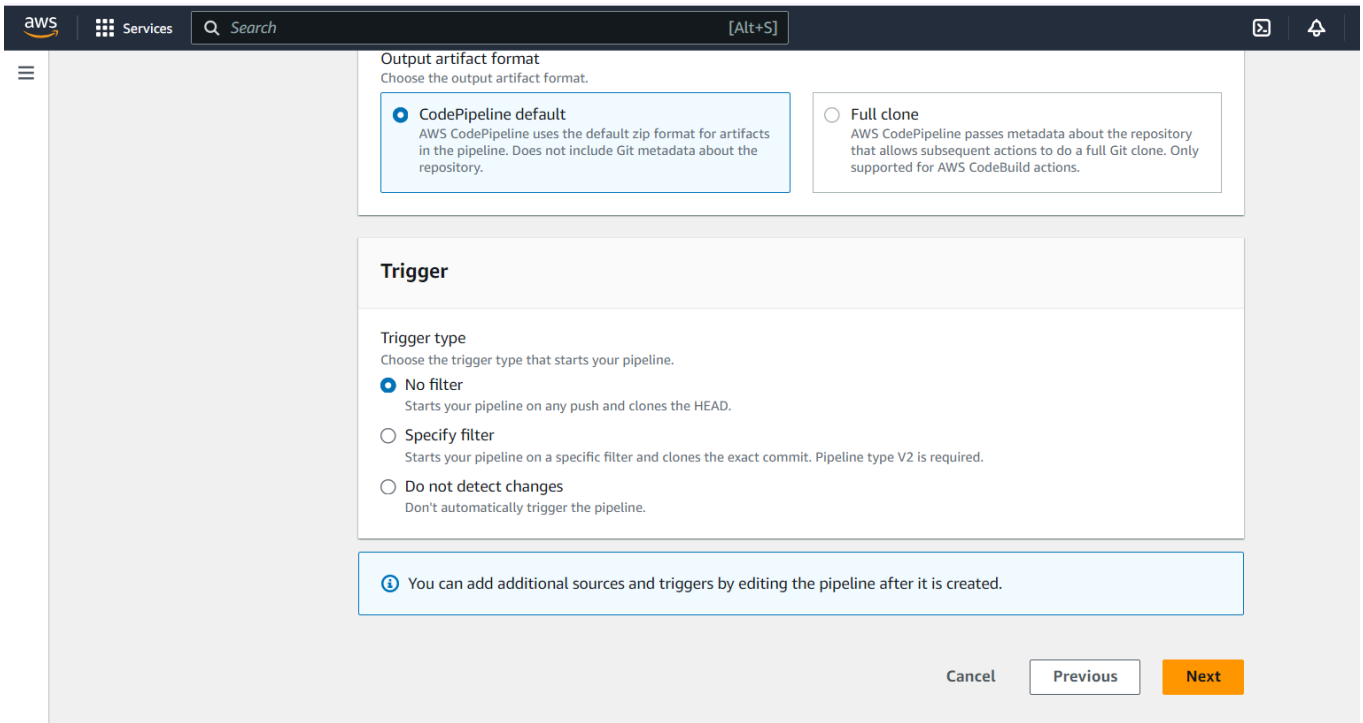


Name: Shiven Bansal **Roll no : 03** **Class: D15C**

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).

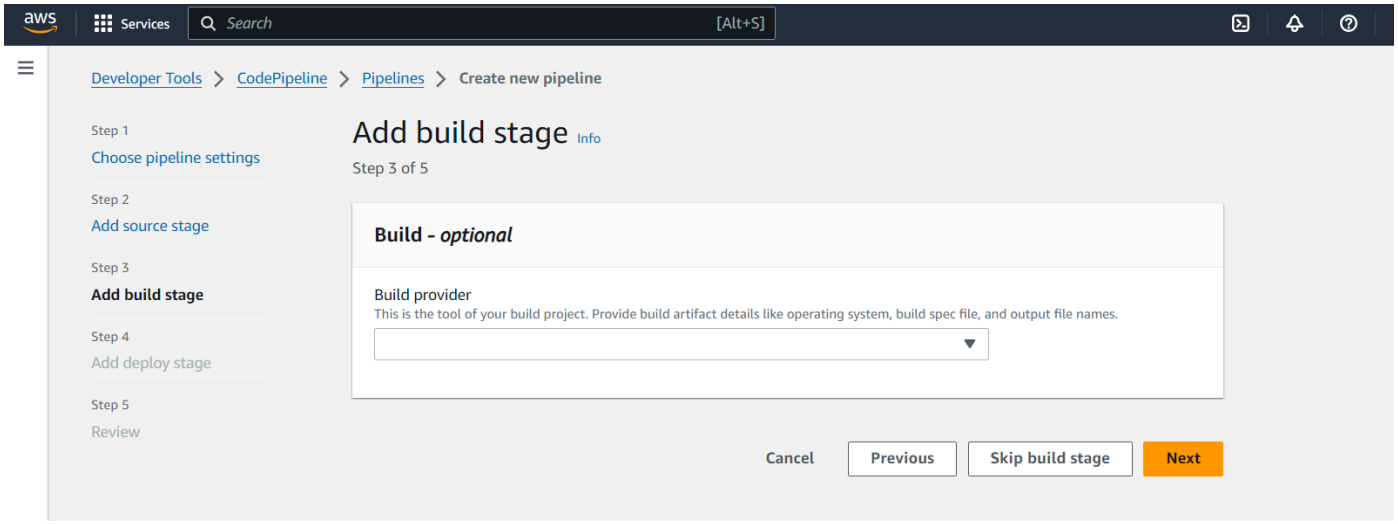


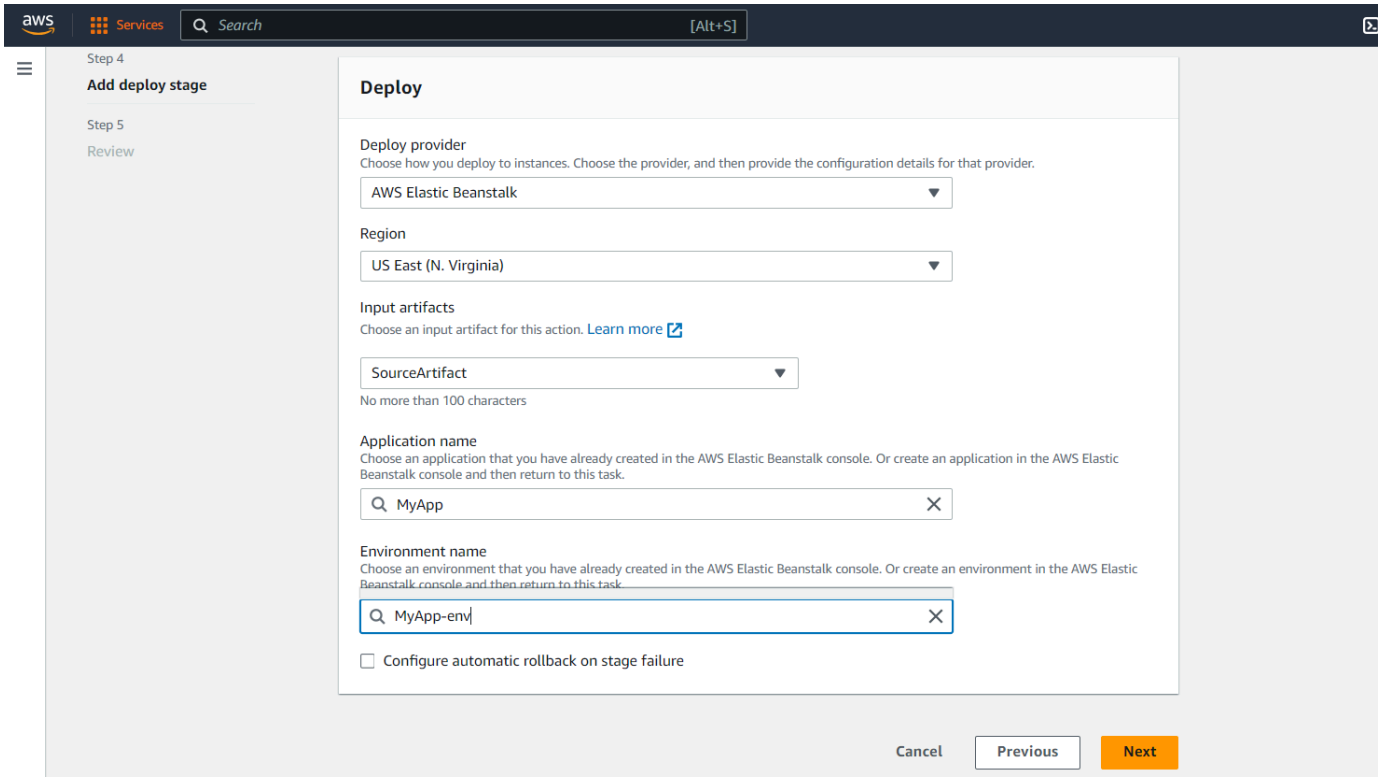
Step 15: Set the Trigger type as no filter. This would allow it to the website to update as soon as some change is made in the github.



Name: Shiven Bansal **Roll no : 03** **Class: D15C**

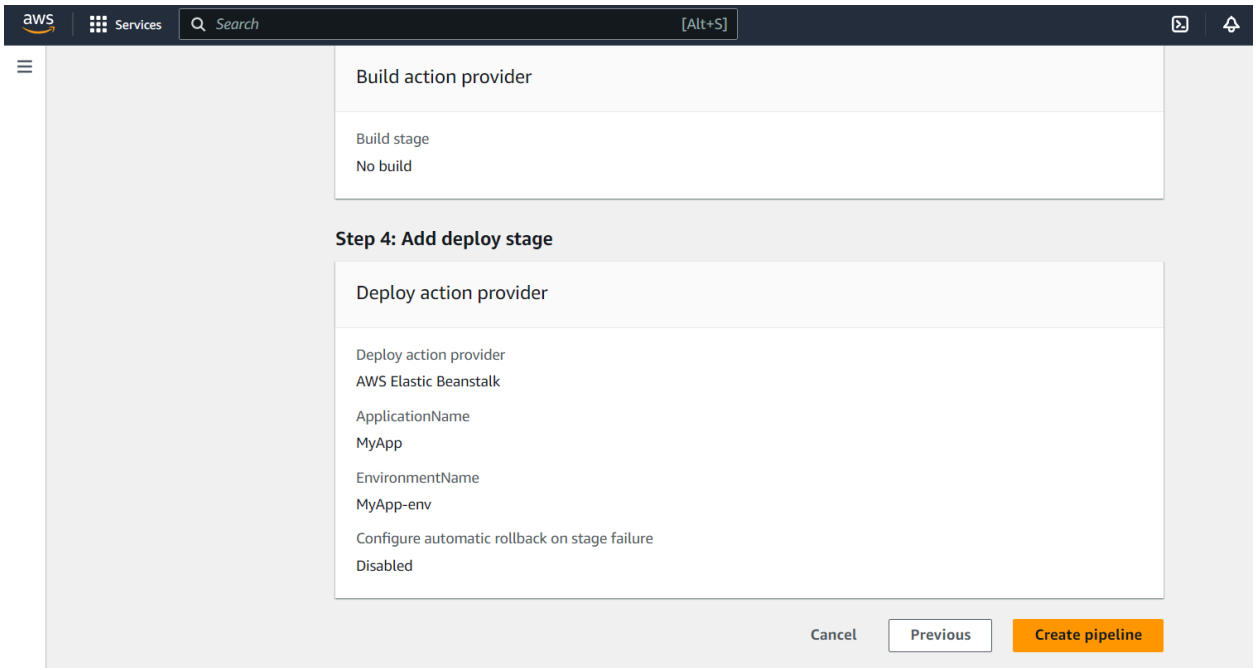
Step 16: Skip the build stage and go to Deploy. Select the deploy provider as AWS Elastic Beanstalk and Input Artifact as SourceArtifact. The application name would be the name of your Elastic Beanstalk. Then click on next.



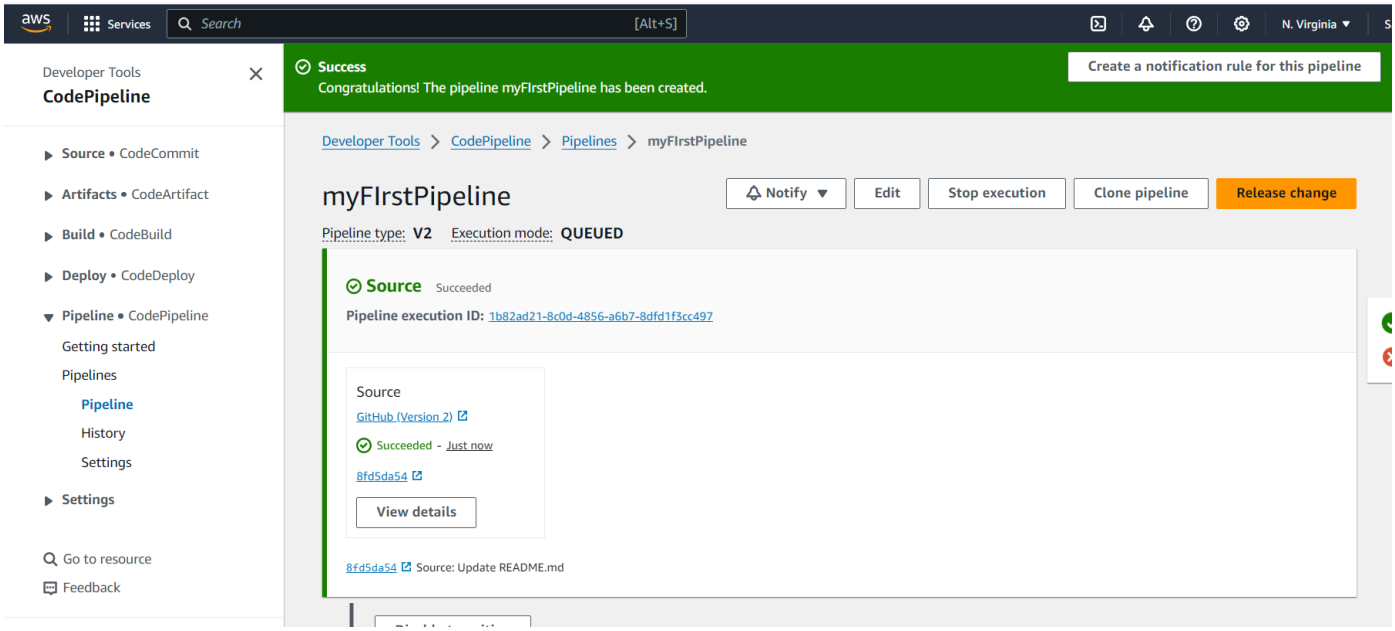


Name: Shiven Bansal **Roll no : 03** **Class: D15C**

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

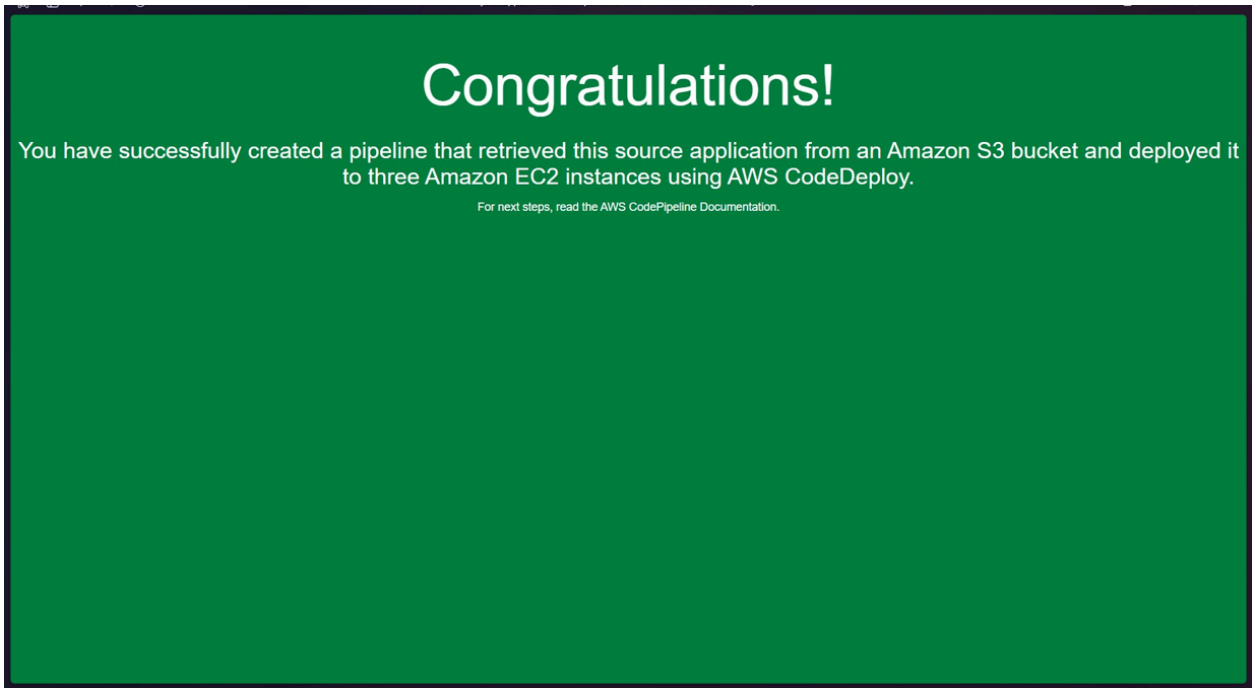


Step 18: If the pipeline is successfully deployed, this screen comes up where the source is set up and then it is transitioned to deploy. Once the deployment is complete, click on the AWS Elastic Beanstalk under Deploy.



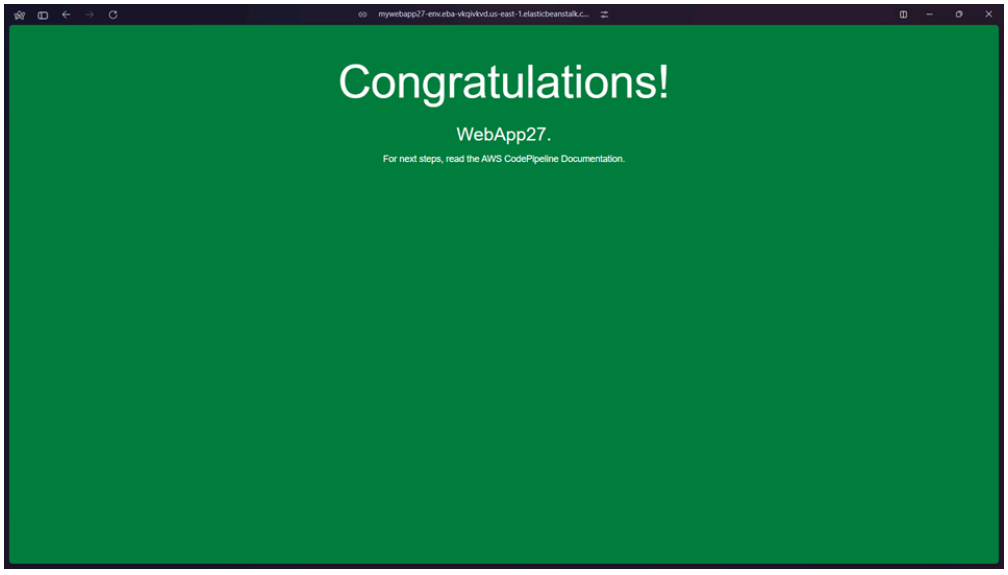
Name: Shiven Bansal **Roll no : 03** **Class: D15C**

Step 19: This will redirect you to the application screen of Elastic Beanstalk. Click on the link shown under Domain.And then this will be shown



Step 20: Now, we make some changes to the index.html file in the github. For eg: If you make some changes to the <h2> tag.

Once the changes are committed, when the website is refreshed, the changes can be seen.

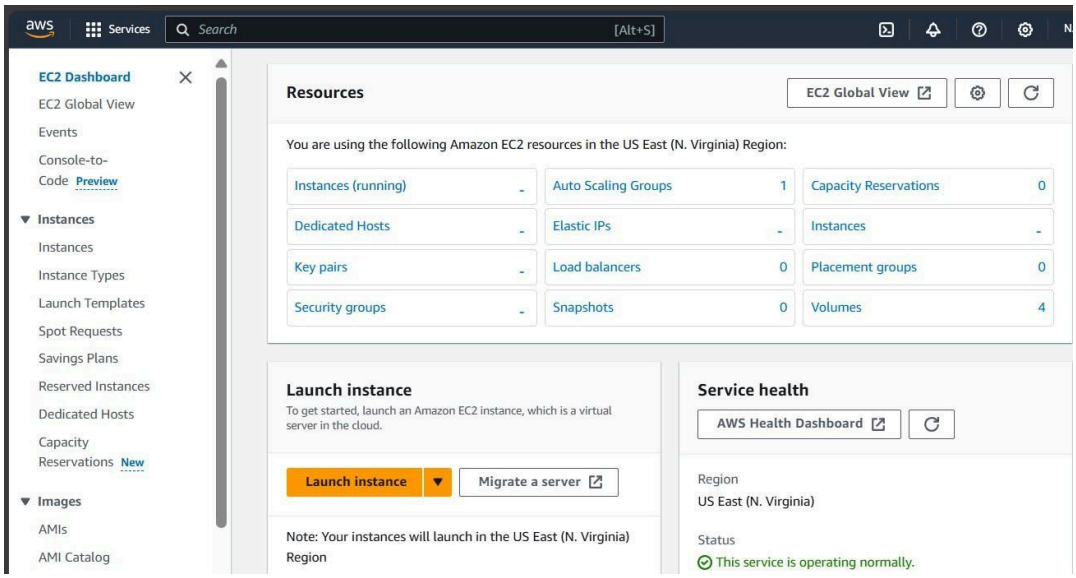


EXPERIMENT NO. 3

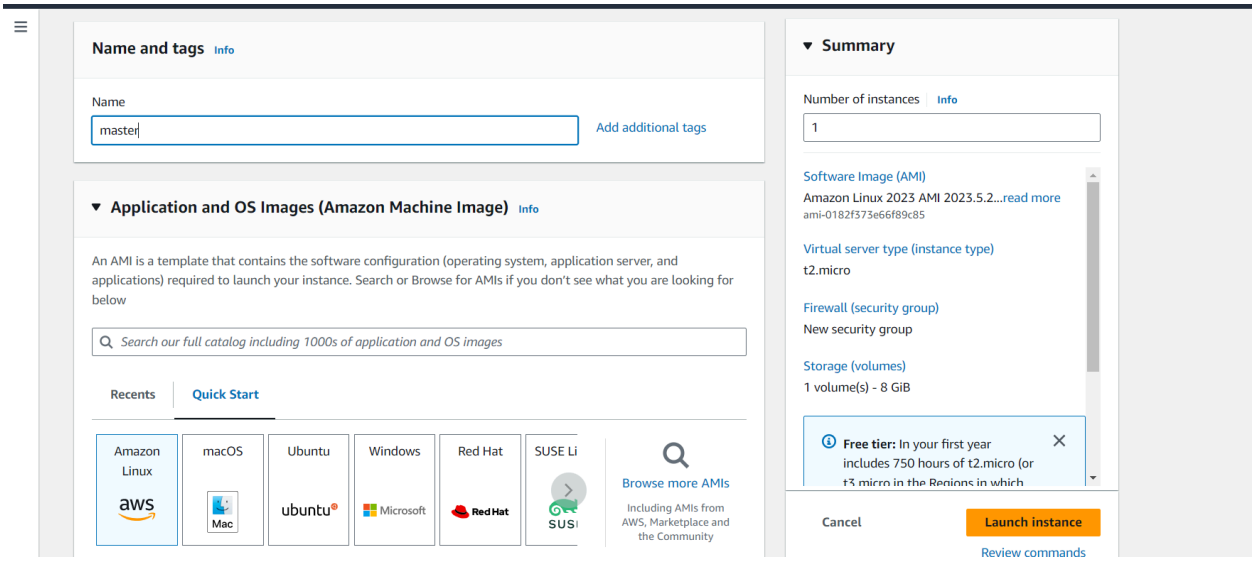
Aim: To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud.

Procedure:

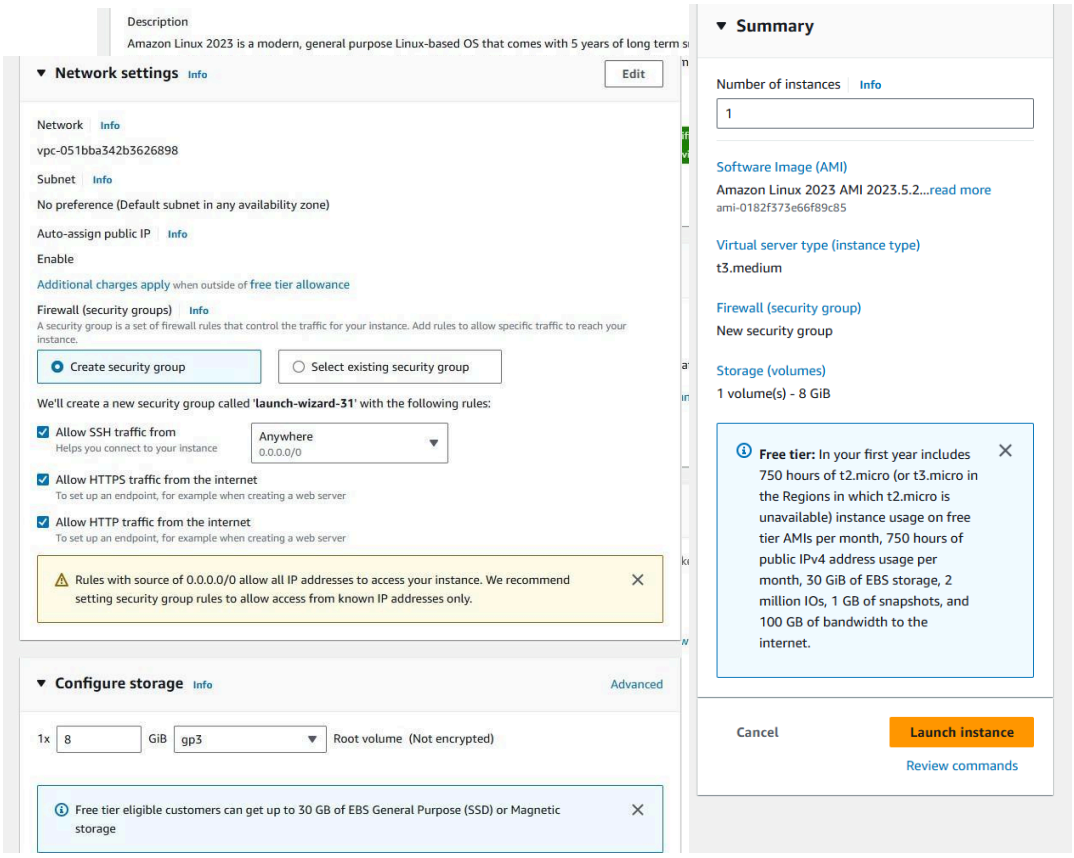
1. **Creation Of Instance**

- 

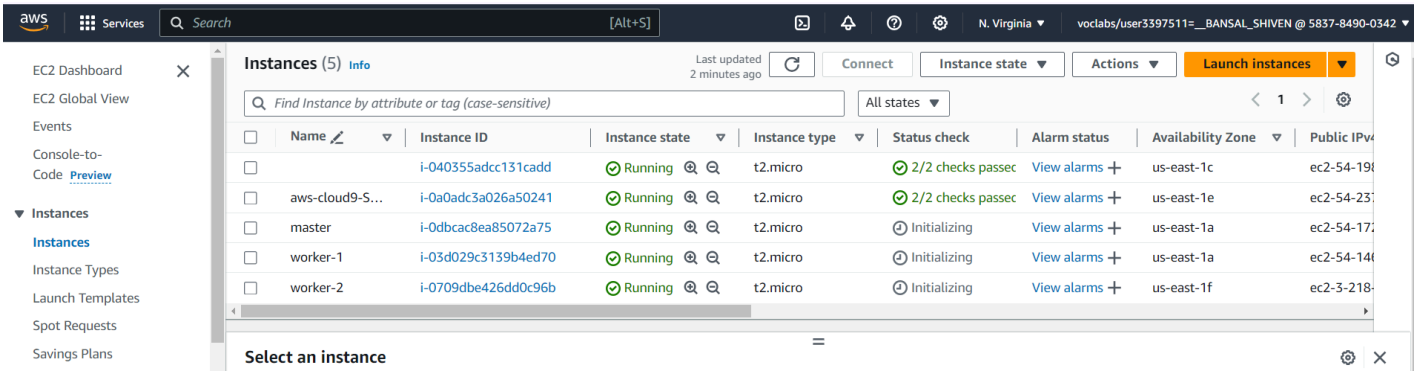
Search EC-2 instance. Then create three EC-2 instances and choose Amazon Linux as OS and also allow ssh traffic from anywhere.



NAME: SHIVEN BANSAL **ROLL NO : 03** **CLASS: D15C**



To efficiently run kubernetes cluster select instance type of at least t3.medium as kubernetes recommends at least 2 vCPU to run smoothly on it.



- Then for making connection through SSH into all 3 machines each in separate terminal Use this following command:
  ssh -i <keyname>.pem ubuntu@<public_ip_address> where keyname is name of the key you created here i created key server.pem and use public IP address.(I have entered this command on git bash where i entered in downloads where server.pem is stored then as the key is not accessible hence we need to change its mode using chmod 400 “key name.pem”. Then use the given command for making connections).

```
ADMIN@DESKTOP-LPV2RP5 MINGW64 ~
$ cd Downloads/

ADMIN@DESKTOP-LPV2RP5 MINGW64 ~/Downloads
$ chmod 400 "server.pem"

ADMIN@DESKTOP-LPV2RP5 MINGW64 ~/Downloads
$ ssh -i "server.pem" ec2-user@ec2-44-192-73-91.compute-1.amazonaws.com
The authenticity of host 'ec2-44-192-73-91.compute-1.amazonaws.com (44.192.73.91
)' can't be established.
ED25519 key fingerprint is SHA256:xIIIlRCIBX1LUOJsm9WHO/YIrDSioPOJfOPEoRKtYJk.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-44-192-73-91.compute-1.amazonaws.com' (ED25519)
to the list of known hosts.
       #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-31-78-148 ~]$
```

2. Installation Of Docker on three machines

- For installation of Docker into all three machines run the following command:
  sudo yum install docker -y

```
[ec2-user@ip-172-31-81-216 ~]$ sudo yum install docker -y
Last metadata expiration check: 0:20:45 ago on Sat Sep 14 06:53:06 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

Total download size: 84 M
Installed size: 317 M
Downloading Packages:
(1/10): iptables-libs-1.8.8-3.amzn2023.0.2.x86_ 5.7 MB/s | 401 kB     00:00
(2/10): iptables-nft-1.8.8-3.amzn2023.0.2.x86_6 6.2 MB/s | 183 kB     00:00
(3/10): libcgroup-3.0-1.amzn2023.0.1.x86_64.rpm 2.2 MB/s |  75 kB     00:00
(4/10): libnetfilter_conntrack-1.0.8-2.amzn2023 1.9 MB/s |  58 kB     00:00
(5/10): libnfnetlink-1.0.1-19.amzn2023.0.2.x86_ 1.2 MB/s |  30 kB     00:00
(6/10): libnftnl-1.2.2-2.amzn2023.0.2.x86_64.rp 2.2 MB/s |  84 kB     00:00
(7/10): pigz-2.5-1.amzn2023.0.3.x86_64.rpm      1.7 MB/s |  83 kB     00:00
(8/10): runc-1.1.13-1.amzn2023.0.1.x86_64.rpm    22 MB/s | 3.2 MB     00:00
(9/10): docker-25.0.6-1.amzn2023.0.2.x86_64.rpm  32 MB/s |  44 MB     00:01
(10/10): containerd-1.7.20-1.amzn2023.0.1.x86_6  23 MB/s |  35 MB     00:01
--------------------------------------------------------------------------------
Total                                            53 MB/s |  84 MB     00:01
Running transaction check
Transaction check succeeded.
  Verifying        : runc-1.1.13-1.amzn2023.0.1.x86_64                 10/10

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!
[ec2-user@ip-172-31-81-216 ~]$ client_loop: send disconnect: Connection reset by
 peer
```

- Then, configure cgroup in a daemon.json file by using following commands cd /etc/docker
  cat <<EOF | sudo tee /etc/docker/daemon.json
  {
  "exec-opts":
  ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
  "max-size": "100m"

},
"storage-driver": "overlay2"
}
EOF

```
[ec2-user@ip-172-31-81-216 ~]$ cd /etc/docker
[ec2-user@ip-172-31-81-216 docker]$ cat <<EOF | sudo tee /etc/docker/daemon.json

{
"exec-opts": ["native.cgroupdriver=systemd"],
"log-driver": "json-file",
"log-opts": {
"max-size": "100m"
},
"storage-driver": "overlay2"
}
EOF
{
"exec-opts": ["native.cgroupdriver=systemd"],
"log-driver": "json-file",
"log-opts": {
"max-size": "100m"
},
"storage-driver": "overlay2"
}
[ec2-user@ip-172-31-81-216 docker]$
```

- Then after this run the following command to enable and start docker and also to load the daemon.json file.
  sudo systemctl enable docker
  sudo systemctl daemon-reload
  sudo systemctl restart docker

```
[ec2-user@ip-172-31-81-216 docker]$ sudo systemctl enable docker
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /us
r/lib/systemd/system/docker.service.
[ec2-user@ip-172-31-81-216 docker]$ sudo systemctl daemon-reload
[ec2-user@ip-172-31-81-216 docker]$ sudo systemctl restart docker
```

- Then check the version of docker installed. docker -v

```
[ec2-user@ip-172-31-81-216 docker]$ docker -v
Docker version 25.0.5, build 5dc9bcc
```

3. Installation Of Kubernetes on three machines

- SELinux needs to be disable before configuring kubelet thus run the following command
  sudo setenforce 0
  sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config

```
[ec2-user@ip-172-31-81-216 ~]$ sudo setenforce 0
[ec2-user@ip-172-31-81-216 ~]$ sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permis
sive/' /etc/selinux/config
```

- Here We are adding kubernetes using the repository whose command is given below. cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
  [kubernetes]
  name=Kubernetes
  baseurl=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/
  enabled=1
  gpgcheck=1
  gpgkey=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/repodata/repomd.xml.key
  exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
  EOF

```
[ec2-user@ip-172-31-81-216 ~]$ cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.
repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
[ec2-user@ip-172-31-81-216 ~]$
```

- After that Run following command to make the updation and also to install kubelet ,kubeadm, kubectl: sudo yum update

```
[ec2-user@ip-172-31-81-216 ~]$ sudo yum update
Kubernetes                                      114 kB/s |  17 kB     00:00
Dependencies resolved.
Nothing to do.
Complete!
```

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

```
[ec2-user@ip-172-31-81-216 ~]$ sudo yum install -y kubelet kubeadm kubectl --dis
ableexcludes=kubernetes
Last metadata expiration check: 0:00:35 ago on Sat Sep 14 07:38:02 2024.
Dependencies resolved.
================================================================================
 Package                 Arch     Version                 Repository     Size
================================================================================
Installing:
 kubeadm                 x86_64   1.30.5-150500.1.1       kubernetes     10 M
 kubectl                 x86_64   1.30.5-150500.1.1       kubernetes     10 M
 kubelet                 x86_64   1.30.5-150500.1.1       kubernetes     17 M
Installing dependencies:
 conntrack-tools         x86_64   1.4.6-2.amzn2023.0.2    amazonlinux   208 k
 cri-tools               x86_64   1.30.1-150500.1.1       kubernetes    8.6 M
 kubernetes-cni          x86_64   1.4.0-150500.1.1        kubernetes    6.7 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
================================================================================
Install  9 Packages

Total download size: 53 M
Installed size: 292 M
Downloading Packages:
(1/9): libnetfilter_cthelper-1.0.0-21.amzn2023. 436 kB/s |  24 kB     00:00
(2/9): libnetfilter_queue-1.0.5-2.amzn2023.0.2. 1.5 MB/s |  30 kB     00:00
(3/9): libnetfilter_cttimeout-1.0.0-19.amzn2023 309 kB/s |  24 kB     00:00
(4/9): conntrack-tools-1.4.6-2.amzn2023.0.2.x86 2.3 MB/s | 208 kB     00:00
(5/9): cri-tools-1.30.1-150500.1.1.x86_64.rpm    34 MB/s | 8.6 MB     00:00
(6/9): kubectl-1.30.5-150500.1.1.x86_64.rpm      21 MB/s |  10 MB     00:00
(7/9): kubeadm-1.30.5-150500.1.1.x86_64.rpm      17 MB/s |  10 MB     00:00
(8/9): kubelet-1.30.5-150500.1.1.x86_64.rpm      32 MB/s |  17 MB     00:00
(9/9): kubernetes-cni-1.4.0-150500.1.1.x86_64.r  21 MB/s | 6.7 MB     00:00
--------------------------------------------------------------------------------
  Verifying        : kubectl-1.30.5-150500.1.1.x86_64                      7/9
  Verifying        : kubelet-1.30.5-150500.1.1.x86_64                      8/9
  Verifying        : kubernetes-cni-1.4.0-150500.1.1.x86_64                9/9

Installed:
  conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64
  cri-tools-1.30.1-150500.1.1.x86_64
  kubeadm-1.30.5-150500.1.1.x86_64
  kubectl-1.30.5-150500.1.1.x86_64
  kubelet-1.30.5-150500.1.1.x86_64
  kubernetes-cni-1.4.0-150500.1.1.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!
[ec2-user@ip-172-31-81-216 ~]$
```

- After installing Kubernetes, we need to configure internet options to allow bridging.
    1. sudo swapoff -a
    2. echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf
    3. sudo sysctl -p

```
[ec2-user@ip-172-31-81-216 ~]$ sudo swapoff -a
[ec2-user@ip-172-31-81-216 ~]$ echo "net.bridge.bridge-nf-call-iptables=1" | sud
o tee -a /etc/sysctl.conf
net.bridge.bridge-nf-call-iptables=1
[ec2-user@ip-172-31-81-216 ~]$ sudo sysctl -p
net.bridge.bridge-nf-call-iptables = 1
```

4. Perform this ONLY on the Master machine

- Initialize kubernetes by typing below command

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

```
[ec2-user@ip-172-31-81-216 ~]$ sudo kubeadm init --pod-network-cidr=10.244.0.0/1
6 --ignore-preflight-errors=all
I0914 07:46:39.398298   28873 version.go:256] remote version is much newer: v1.3
1.0; falling back to: stable-1.30
[init] Using Kubernetes version: v1.30.4
[preflight] Running pre-flight checks
        [WARNING NumCPU]: the number of available CPUs 1 is less than the requir
ed 2

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 172.31.81.216:6443 --token f35qak.rrjyqp89bqsomqrn \
        --discovery-token-ca-cert-hash sha256:593693973f6b40e7bc61dec2c73c617ba2
26caafee64bc2776ee360c42b6f29c
```

- So after initialization you will get token at the end for joining master and worker. Like here I got this :(save this token as it is required later.Then you can join any number of worker nodes by running the following on each as root.)

kubeadm join 172.31.81.216:6443 --token f35qak.rrjyqp89bqsomqrn \
    --discovery-token-ca-cert-hash
sha256:593693973f6b40e7bc61dec2c73c617ba226caafee64bc2776ee360c42b6f29c

- Also,Copy the mkdir and chown commands from the top and execute them mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

```
[ec2-user@ip-172-31-81-216 ~]$ mkdir -p $HOME/.kube
[ec2-user@ip-172-31-81-216 ~]$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube
/config
[ec2-user@ip-172-31-81-216 ~]$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

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

```
[ec2-user@ip-172-31-81-216 ~]$ kubectl apply -f https://raw.githubusercontent.co
m/coreos/flannel/master/Documentation/kube-flannel.yml
error: error validating "https://raw.githubusercontent.com/coreos/flannel/master
/Documentation/kube-flannel.yml": error validating data: failed to download open
api: Get "https://172.31.81.216:6443/openapi/v2?timeout=32s": dial tcp 172.31.81
.216:6443: connect: connection refused; if you choose to ignore these errors, tu
rn validation off with --validate=false
```

This step gives an error

Conclusion:

In this experiment, we successfully set up a Kubernetes cluster across three Amazon Linux EC2 instances, each equipped with Kubernetes components. The master node was initialized using kubeadm, and pod networking was configured with the Flannel network plugin. Worker nodes were integrated into the cluster through the join command generated during the master node's initialization. The process provided a comprehensive understanding of Kubernetes cluster setup on EC2. However, there was a noticeable delay in the worker nodes connecting to the master node, which may have been caused by network connectivity issues or configuration discrepancies.

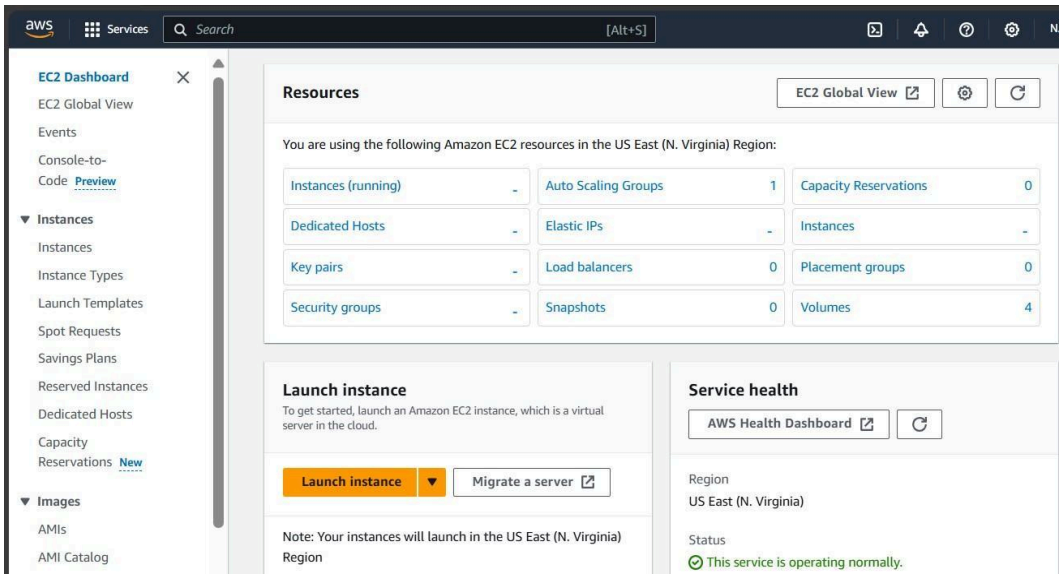
Experiment 4

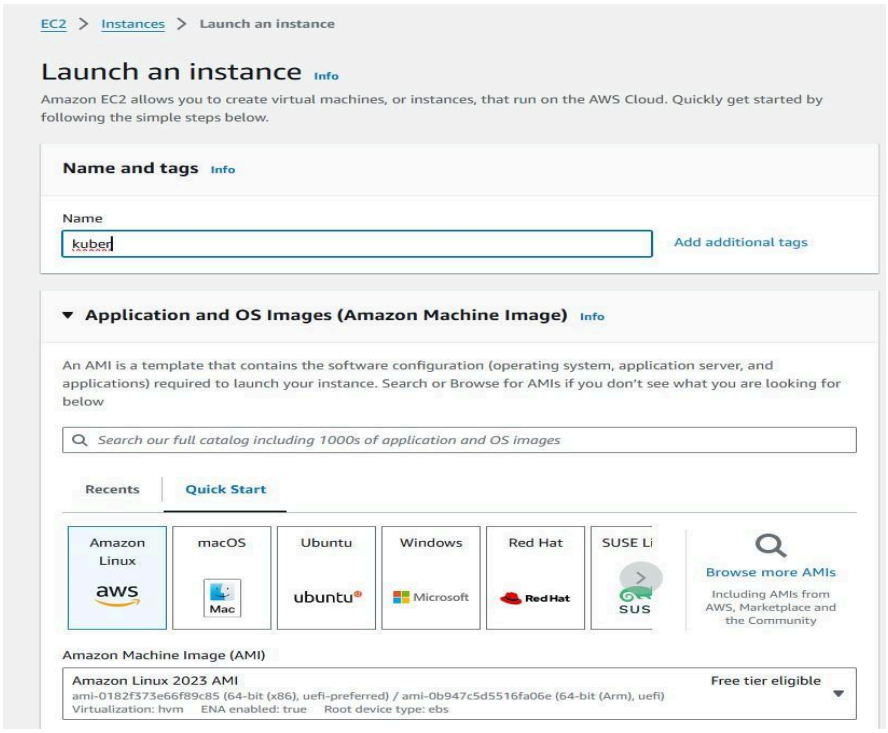
Aim: To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deploy Your First Kubernetes Application.

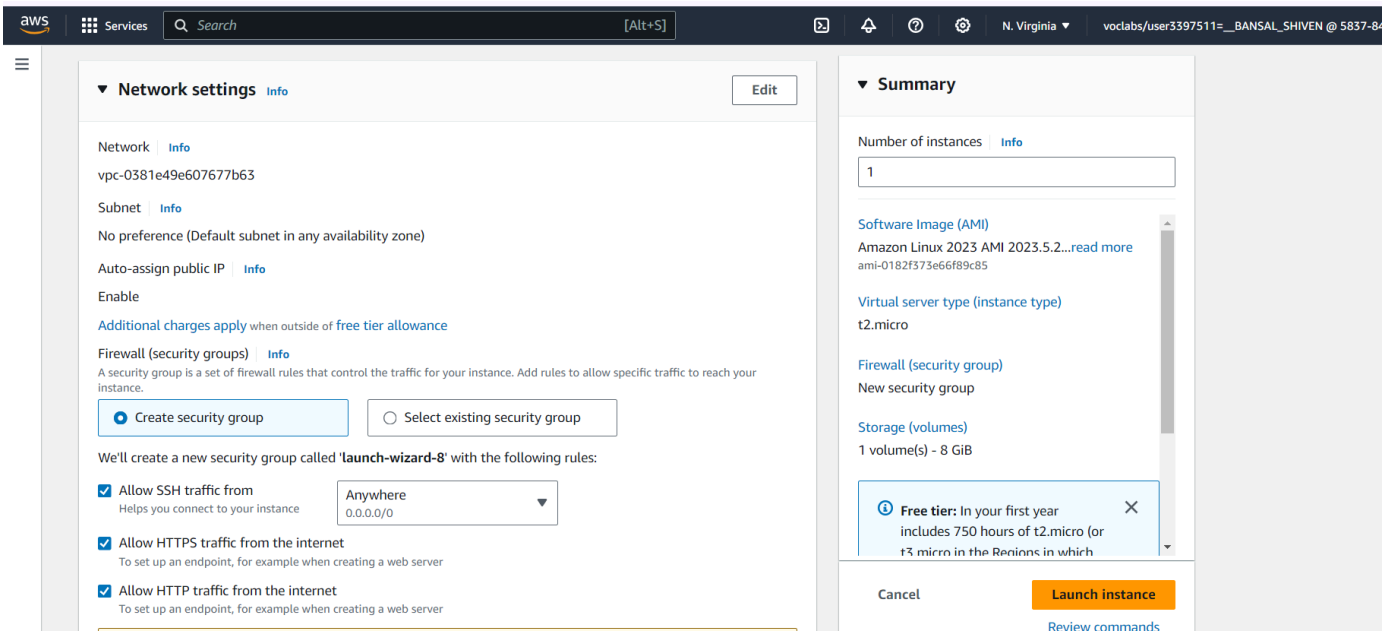
Procedure:

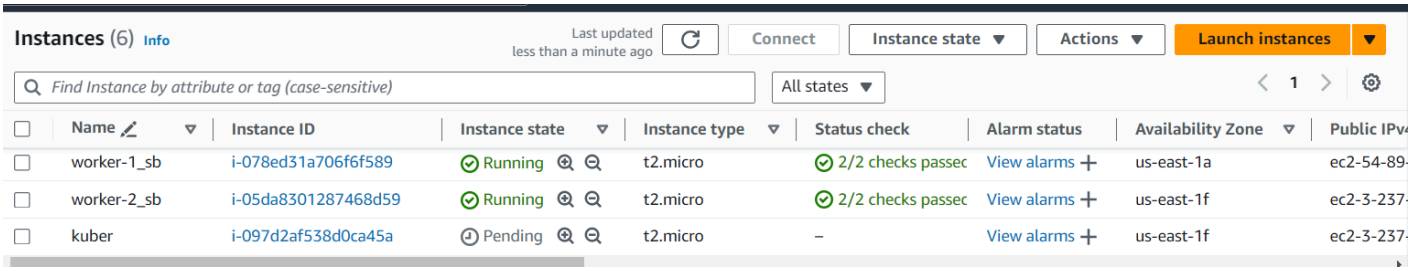
1. **Creation Of EC-2 instance**

- Create an EC2 AWS Linux instance on AWS .also edit the Security Group Inbound Rules to allow SSH. then select the t2.micro instance type

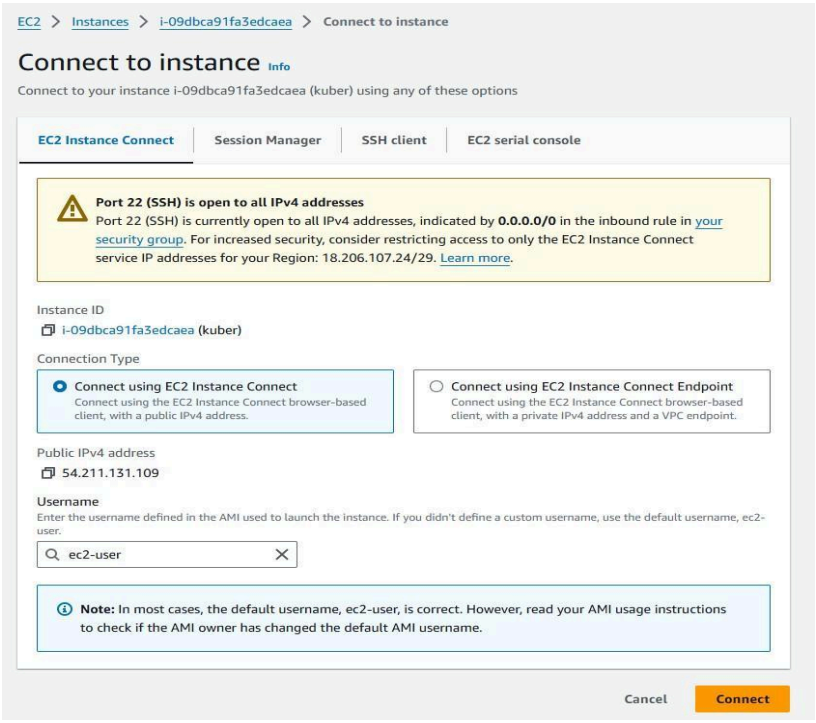








- Thus Kuber named -instance gets created.Then click on Id of that instance then click on connect button you will see this:



- Then go into SSH client where you will get this command
  Chmod 400 “keyname.pem”
  ssh -i <keyname>.pem ubuntu@<public_ip_address> copy it and then connect it and run the following command for establishing connection.(I have entered this command on git bash where i entered in downloads where server.pem is stored then as the key is not accessible hence we need to change its mode using chmod 400 “key name.pem”. Then use the given command for making connections).

```
ADMIN@DESKTOP-LPV2RP5 MINGW64 ~
$ cd Downloads/

ADMIN@DESKTOP-LPV2RP5 MINGW64 ~/Downloads
$ chmod 400 "server.pem"

ADMIN@DESKTOP-LPV2RP5 MINGW64 ~/Downloads
$ ssh -i "server.pem" ec2-user@ec2-3-237-37-35.compute-1.amazonaws.com
The authenticity of host 'ec2-3-237-37-35.compute-1.amazonaws.com (3.237.37.35)'
 can't be established.
ED25519 key fingerprint is SHA256:uhsLQC4WN+6i3np6iMFwW+vGF8aGkTrm89ryXQ4wT1g.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-237-37-35.compute-1.amazonaws.com' (ED25519) t
o the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Sat Sep 14 11:47:26 2024 from 18.206.107.28
[ec2-user@ip-172-31-65-181 ~]$
```

2. Installation of Docker

1. . For installation of Docker into the machines run the following command: sudo yum install docker -y

```
[ec2-user@ip-172-31-65-181 ~]$ sudo yum install docker -y
Last metadata expiration check: 0:08:49 ago on Sat Sep 14 11:42:25 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

Total download size: 84 M
Installed size: 317 M
Downloading Packages:
(1/10): iptables-libs-1.8.8-3.amzn2023.0.2.x86_ 4.5 MB/s | 401 kB     00:00
(2/10): iptables-nft-1.8.8-3.amzn2023.0.2.x86_6 4.5 MB/s | 183 kB     00:00
(3/10): libcgroup-3.0-1.amzn2023.0.1.x86_64.rpm 3.9 MB/s |  75 kB     00:00
(4/10): libnetfilter_conntrack-1.0.8-2.amzn2023 2.6 MB/s |  58 kB     00:00
(5/10): libnfnetlink-1.0.1-19.amzn2023.0.2.x86_ 1.2 MB/s |  30 kB     00:00
(6/10): libnftnl-1.2.2-2.amzn2023.0.2.x86_64.rp 2.2 MB/s |  84 kB     00:00
(7/10): pigz-2.5-1.amzn2023.0.3.x86_64.rpm      2.8 MB/s |  83 kB     00:00
(8/10): runc-1.1.13-1.amzn2023.0.1.x86_64.rpm    30 MB/s | 3.2 MB     00:00
(9/10): containerd-1.7.20-1.amzn2023.0.1.x86_64  38 MB/s |  35 MB     00:00
(10/10): docker-25.0.6-1.amzn2023.0.2.x86_64.rp  34 MB/s |  44 MB     00:01
--------------------------------------------------------------------------------
Total                                            62 MB/s |  84 MB     00:01
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /usr/lib/systemd/system/docker.socket.

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

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
  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
  pigz-2.5-1.amzn2023.0.3.x86_64                  runc-1.1.13-1.amzn2023.0.1.x86_64

Complete!
```

- Then, configure cgroup in a daemon.json file by using following commands cd /etc/docker
  cat <<EOF | sudo tee /etc/docker/daemon.json
  {
  "exec-opts":
  ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
  "max-size": "100m"
  },
  "storage-driver": "overlay2"

}
EOF

```
[ec2-user@ip-172-31-65-181 ~]$ cd /etc/docker
[ec2-user@ip-172-31-65-181 docker]$ cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts":
["native.cgroupdriver=systemd"],
"log-driver": "json-file",
"log-opts": {
"max-size": "100m"
},
"storage-driver": "overlay2"
}
EOF
{
"exec-opts":
["native.cgroupdriver=systemd"],
"log-driver": "json-file",
"log-opts": {
"max-size": "100m"
},
"storage-driver": "overlay2"
}
```

- Then after this run the following command to enable and start docker and also to load the daemon.json file.
  sudo systemctl enable docker
  sudo systemctl daemon-reload
  sudo systemctl restart docker

```
[ec2-user@ip-172-31-81-216 docker]$ sudo systemctl enable docker
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /us
r/lib/systemd/system/docker.service.
[ec2-user@ip-172-31-81-216 docker]$ sudo systemctl daemon-reload
[ec2-user@ip-172-31-81-216 docker]$ sudo systemctl restart docker
```

- docker -v

```
[ec2-user@ip-172-31-65-181 docker]$ docker -v
Docker version 25.0.5, build 5dc9bcc
```

3. Then Install Kubernetes with the following command.

- SELinux needs to be disable before configuring kubelet thus run the following command
  sudo setenforce 0
  sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config

```
[ec2-user@ip-172-31-65-181 docker]$ sudo setenforce 0
[ec2-user@ip-172-31-65-181 docker]$ sudo sed -i 's/^SELINUX=enforcing$/SELINUX=p
ermissive/' /etc/selinux/config
```

- Here We are adding kubernetes using the repository whose command is given below. cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
  [kubernetes]

name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF

```
[ec2-user@ip-172-31-65-181 docker]$ sudo setenforce 0
[ec2-user@ip-172-31-65-181 docker]$ sudo sed -i 's/^SELINUX=enforcing$/SELINUX=p
ermissive/' /etc/selinux/config
[ec2-user@ip-172-31-65-181 docker]$  cat <<EOF | sudo tee /etc/yum.repos.d/kuber
netes.repo
[kubernetes]
name=Kubernetes baseurl=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/ enabled=1
gpgcheck=1 gpgkey=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/repodata/repomd.xm
l.key exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
[kubernetes]
name=Kubernetes baseurl=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/ enabled=1
gpgcheck=1 gpgkey=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/repodata/repomd.xm
l.key exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
[ec2-user@ip-172-31-65-181 docker]$
```

- After that Run following command to make the updation and also to install kubelet ,kubeadm, kubectl: sudo yum update

```
[ec2-user@ip-172-31-65-181 docker]$ sudo yum update
Invalid configuration value: gpgcheck=1 gpgkey=https://pkgs.k8s.io/core:/stable:
/v1.30/rpm/repodata/repomd.xml.key exclude=kubelet kubeadm kubectl cri-tools kub
ernetes-cni in /etc/yum.repos.d/kubernetes.repo; invalid boolean value '1 gpgkey
=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/repodata/repomd.xml.key exclude=kub
elet kubeadm kubectl cri-tools kubernetes-cni'
Error: Cannot find a valid baseurl for repo: kubernetes
Ignoring repositories: kubernetes
Last metadata expiration check: 0:19:14 ago on Sat Sep 14 11:42:25 2024.
Dependencies resolved.
Nothing to do.
Complete!
```

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

```
[ec2-user@ip-172-31-81-216 ~]$ sudo yum install -y kubelet kubeadm kubectl --dis
ableexcludes=kubernetes
Last metadata expiration check: 0:00:35 ago on Sat Sep 14 07:38:02 2024.
Dependencies resolved.
================================================================================
 Package                  Arch     Version                  Repository     Size
================================================================================
Installing:
 kubeadm                  x86_64   1.30.5-150500.1.1        kubernetes     10 M
 kubectl                  x86_64   1.30.5-150500.1.1        kubernetes     10 M
 kubelet                  x86_64   1.30.5-150500.1.1        kubernetes     17 M
Installing dependencies:
 conntrack-tools          x86_64   1.4.6-2.amzn2023.0.2     amazonlinux   208 k
 cri-tools                x86_64   1.30.1-150500.1.1        kubernetes    8.6 M
 kubernetes-cni           x86_64   1.4.0-150500.1.1         kubernetes    6.7 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
================================================================================
Install  9 Packages

Total download size: 53 M
Installed size: 292 M
Downloading Packages:
(1/9): libnetfilter_cthelper-1.0.0-21.amzn2023. 436 kB/s |  24 kB     00:00
(2/9): libnetfilter_queue-1.0.5-2.amzn2023.0.2. 1.5 MB/s |  30 kB     00:00
(3/9): libnetfilter_cttimeout-1.0.0-19.amzn2023 309 kB/s |  24 kB     00:00
(4/9): conntrack-tools-1.4.6-2.amzn2023.0.2.x86 2.3 MB/s | 208 kB     00:00
(5/9): cri-tools-1.30.1-150500.1.1.x86_64.rpm    34 MB/s | 8.6 MB     00:00
(6/9): kubectl-1.30.5-150500.1.1.x86_64.rpm      21 MB/s |  10 MB     00:00
(7/9): kubeadm-1.30.5-150500.1.1.x86_64.rpm      17 MB/s |  10 MB     00:00
(8/9): kubelet-1.30.5-150500.1.1.x86_64.rpm      32 MB/s |  17 MB     00:00
(9/9): kubernetes-cni-1.4.0-150500.1.1.x86_64.r  21 MB/s | 6.7 MB     00:00
--------------------------------------------------------------------------------
Verifying        : kubectl-1.30.5-150500.1.1.x86_64                      7/9
Verifying        : kubelet-1.30.5-150500.1.1.x86_64                      8/9
Verifying        : kubernetes-cni-1.4.0-150500.1.1.x86_64                9/9

Installed:
  conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64
  cri-tools-1.30.1-150500.1.1.x86_64
  kubeadm-1.30.5-150500.1.1.x86_64
  kubectl-1.30.5-150500.1.1.x86_64
  kubelet-1.30.5-150500.1.1.x86_64
  kubernetes-cni-1.4.0-150500.1.1.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!
[ec2-user@ip-172-31-81-216 ~]$
```

- After installing Kubernetes, we need to configure internet options to allow bridging.
    1. sudo swapoff -a
    2. echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf
    3. sudo sysctl -p

```
[ec2-user@ip-172-31-81-216 ~]$ sudo swapoff -a
[ec2-user@ip-172-31-81-216 ~]$ echo "net.bridge.bridge-nf-call-iptables=1" | sud
o tee -a /etc/sysctl.conf
net.bridge.bridge-nf-call-iptables=1
[ec2-user@ip-172-31-81-216 ~]$ sudo sysctl -p
net.bridge.bridge-nf-call-iptables = 1
```

4. Initialize the Kubecluster

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

```
[ec2-user@ip-172-31-80-126 docker]$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
I0913 10:32:44.629146   26680 version.go:256] remote version is much newer: v1.31.0; falling back to: stable-1.30
[init] Using Kubernetes version: v1.30.4
[preflight] Running pre-flight checks
```

```
Your Kubernetes control-plane has initialized successfully!

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

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

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

  export KUBECONFIG=/etc/kubernetes/admin.conf

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

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

kubeadm join 172.31.26.174:6443 --token pv0yyi.xhllqhclfjr50pt8 \
        --discovery-token-ca-cert-hash sha256:8293b2f6d29de466bd859007f5adbcdb3a
ecb0c446ba09033d32a5846b3d434f
```

- copy the token and save for future use .
  kubeadm join 172.31.26.174:6443 --token pv0yyi.xhllqhclfjr50pt8 \--discovery-token-ca-cert-hash sha256:8293b2f6d29de466bd859007f5adbcdb3aecb0c446ba09033d32a5846b3d434f
- Copy the mkdir and chown commands from the top and execute them mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

```
[ec2-user@ip-172-31-80-126 docker]$ ^C
[ec2-user@ip-172-31-80-126 docker]$  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

- Then, add a common networking plugin called flannel as mentioned in the code.

```
kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/D
o cumentation/kube-flannel.yml
```

```
[ec2-user@ip-172-31-26-174 docker]$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
namespace/kube-flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
```

5. Now that the cluster is up and running,we can deploy our nginx server on this cluster. Apply deployment using this following command:

```
kubectl apply -f

https://k8s.io/examples/pods/simple-pod.yaml
```

```
[ec2-user@ip-172-31-26-174 docker]$ kubectl apply -f https://k8s.io/examples/pods/s
imple-pod.yaml
pod/nginx created
```

Then use kubectl get nodes to check whether the pod gets created or not.

```
[ec2-user@ip-172-31-26-174 docker]$ kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
nginx   0/1     Pending   0          12s
```

To convert state from pending to running use following command:

kubectl describe pod nginx This command will help to describe the pods it gives reason for failure as it shows the untolerated taints which need to be untainted.

```
[ec2-user@ip-172-31-26-174 docker]$ kubectl describe pod nginx
Name:             nginx
Namespace:        default
Priority:         0
Service Account:  default
Node:             <none>
Labels:           <none>
Annotations:      <none>
Status:           Pending
IP:
IPs:              <none>
Containers:
  nginx:
    Image:        nginx:1.14.2
    Port:         80/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-k4lj6 (ro)
Conditions:
  Type           Status
  PodScheduled   False
Volumes:
  kube-api-access-k4lj6:
    Type:                     Projected (a volume that contains injected data from m
ultiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 3
00s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for
 300s
Events:
  Type     Reason            Age   From               Message
  ----     ------            ----  ----               -------
  Warning  FailedScheduling  7s    default-scheduler  0/1 nodes are available: 1 no
de(s) had untolerated taint {node-role.kubernetes.io/control-plane: }. preemption:
0/1 nodes are available: 1 Preemption is not helpful for scheduling.
```

```
[ec2-user@ip-172-31-26-174 ~]$ kubectl taint nodes --all node-role.kubernetes.io
/control-plane-
node/ip-172-31-26-174.ec2.internal untainted
```

6. Now check pod status is is running

```
[ec2-user@ip-172-31-26-174 ~]$ kubectl get pods
NAME    READY   STATUS    RESTARTS     AGE
nginx   1/1     Running   1 (6s ago)   90s
```

7. Lastly, mention the port you want to host. Here i have used localhost 8081 then check it.

```
kubectl port-forward nginx 8081:80
```

```
[ec2-user@ip-172-31-26-174 ~]$ kubectl port-forward nginx 8081:80
Forwarding from 127.0.0.1:8081 -> 80
Forwarding from [::1]:8081 -> 80
```

8. Verify your deployment

Open up a new terminal and ssh to your EC2 instance.

Then, use this curl command to check if the Nginx server is running.

```
curl --head http://127.0.0.1:8080
```

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

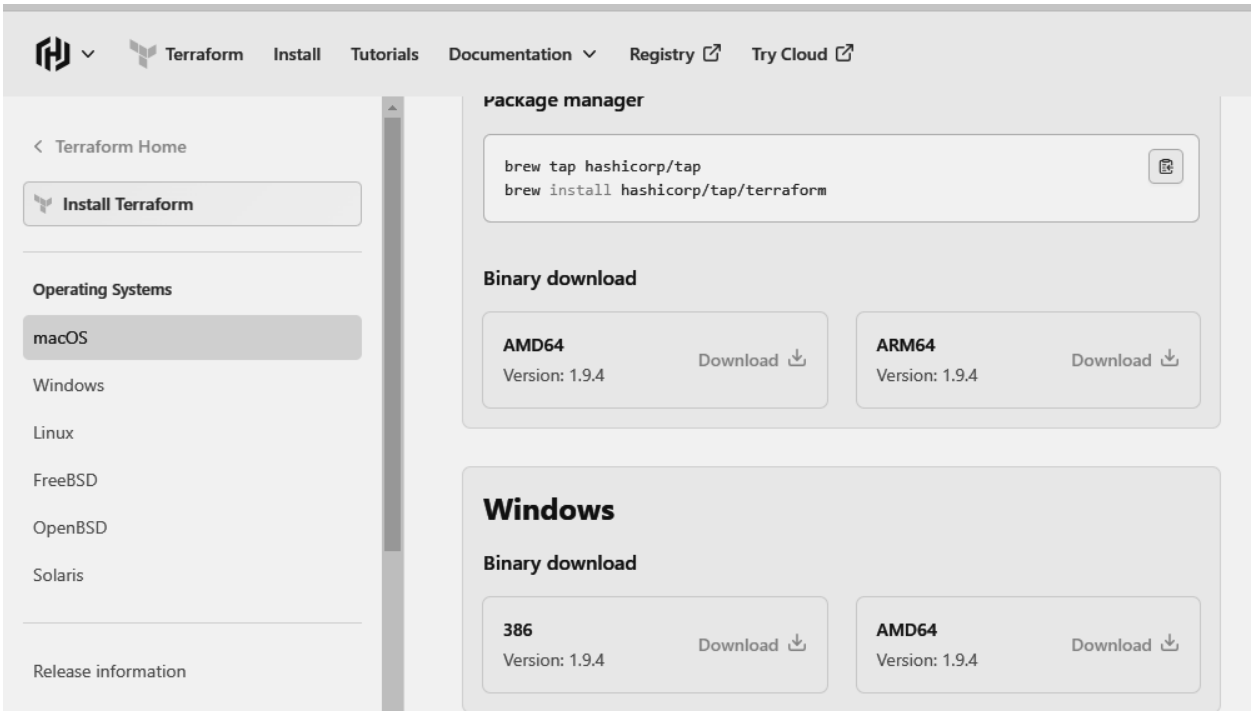
Conclusion:

First, I successfully launched an AWS EC2 instance running Amazon Linux. After that, I installed Docker and Kubernetes on the instance. Following the installation, I initialized the Kubernetes cluster, which provided me with a token, along with chown and mkdir commands. I then executed both the mkdir and chown commands successfully. Next, I installed the Flannel networking plugin without any issues. Initially, there was an error while deploying Nginx, but after correcting it, I successfully deployed Nginx using a simple-pod.yml file. I confirmed its deployment with the get pods command and hosted it locally on http://localhost:8081, which worked as expected.

Name: Shiven Bansal Roll no: 03 Class: D15C

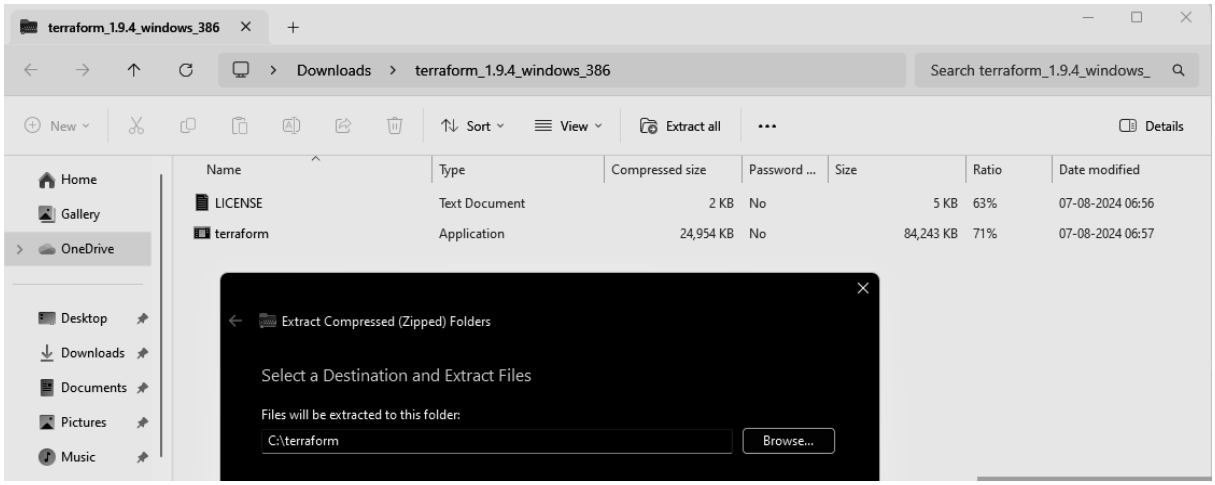
Experiment-5

Step 1: Download terraform
To install Terraform, First Download the Terraform Cli Utility for windows from terraforms official website
https://www.terraform.io/downloads.html
Select the Operating System Windows followed by either 32bit or 64 bit based on your OS type.

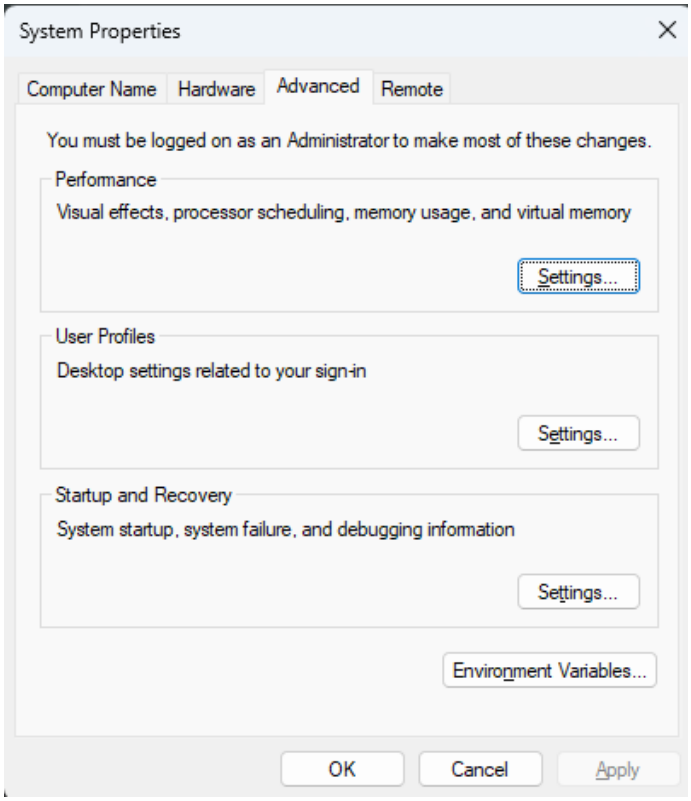


Step 2: Extract the downloaded setup file Terraform.exe in C:\Terraform directory

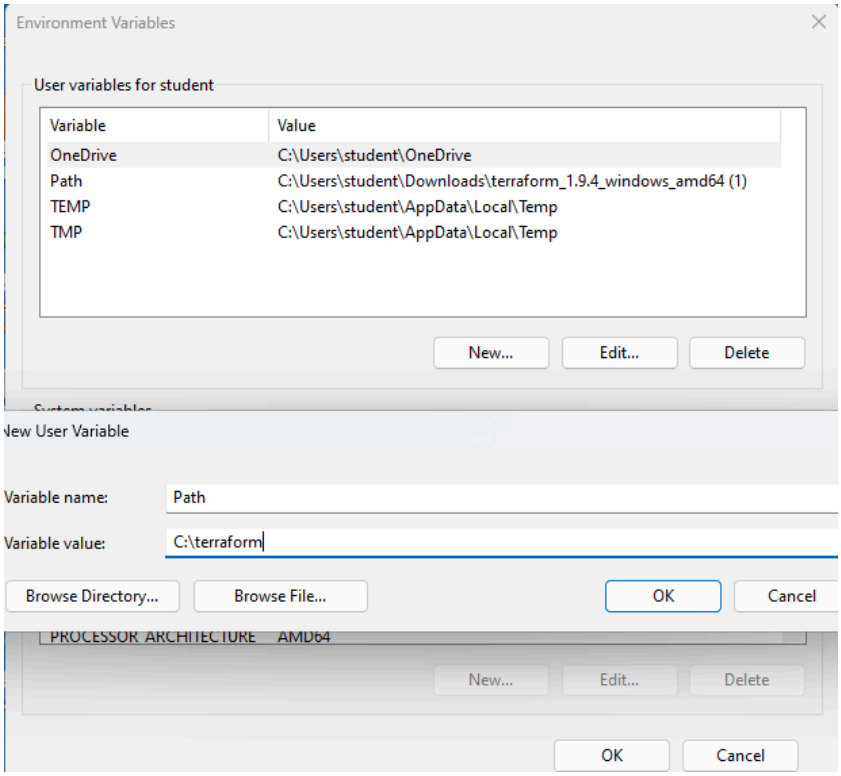
Name: Shiven Bansal Roll no: 03 Class: D15C



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



Name: Shiven Bansal Roll no: 03 Class: D15C



Step 4: Open PowerShell with Admin Access

Step 5 : Open Terraform in PowerShell and check its functionality

Name: Shiven Bansal Roll no: 03 Class: D15C

```
PS C:\Users\student> 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
```

Name : Shiven Bansal **Roll no : 03** **Class : D15C**

Experiment No : 6

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

Implementation:

A. Creating docker image using terraform

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

Step 1: Check the docker functionality

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

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Options:
      --config string      Location of client config files (default
                           "C:\\Users\\student\\.docker")
  -c, --context string     Name of the context to use to connect to the
                           daemon (overrides DOCKER_HOST env var and
                           default context set with "docker context use")
  -D, --debug              Enable debug mode
  -H, --host list          Daemon socket(s) to connect to
  -l, --log-level string   Set the logging level
                           ("debug"|"info"|"warn"|"error"|"fatal")
                           (default "info")
      --tls                Use TLS; implied by --tlsverify
      --tlscacert string   Trust certs signed only by this CA (default
                           "C:\\Users\\student\\.docker\\ca.pem")
      --tlscert string     Path to TLS certificate file (default
                           "C:\\Users\\student\\.docker\\cert.pem")
      --tlskey string      Path to TLS key file (default
                           "C:\\Users\\student\\.docker\\key.pem")
      --tlsverify          Use TLS and verify the remote
  -v, --version            Print version information and quit

Management Commands:
  builder     Manage builds
  buildx*     Docker Buildx (Docker Inc., v0.9.1)
```

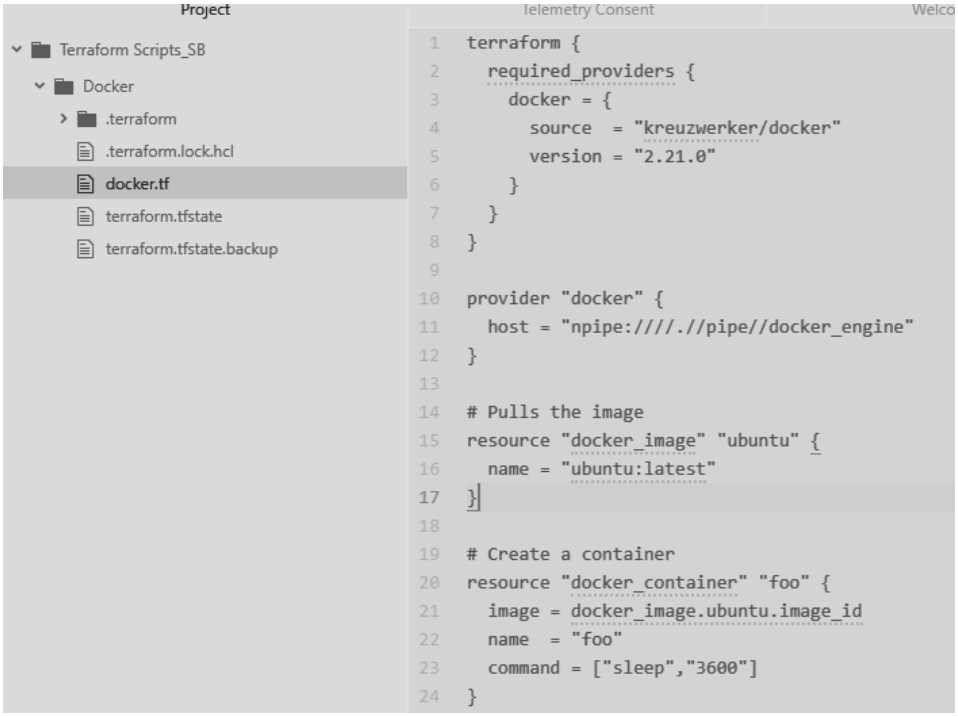
```
PS C:\Users\student> docker --version
Docker version 20.10.17, build 100c701
PS C:\Users\student>
```

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

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

Script:

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

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

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

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



Step 3: Execute Terraform Init command to initialize the resources

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

Terraform has been successfully initialized!

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

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

Step 4: Execute Terraform plan to see the available resources

```
PS C:\Terraform Scripts_SB\Docker> terraform plan

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

Terraform will perform the following actions:

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

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

      + healthcheck (known after apply)

      + labels (known after apply)
    }

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

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

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

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

```
PS C:\Terraform Scripts_SB\Docker> terraform apply

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

Terraform will perform the following actions:

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

      + healthcheck (known after apply)

      + labels (known after apply)
    }

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

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

  Enter a value: yes

docker_container.foo: Creating...
docker_container.foo: Creation complete after 0s [id=ed65bf8e57faf1420fd6a6071b9e3bbaad2de46dedbfd353a66e954bbd7d0881]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Terraform Scripts_SB\Docker>
```

Docker images, Before Executing Apply step:

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

Docker images, After Executing Apply step:

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

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

```
PS C:\Terraform Scripts_SB\Docker> terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Refreshing state... [id=ed65bf8e57faf1420fd6a6071b9e3bbaad2de46dedbfd353a66e954bbd7d0881]

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

Terraform will perform the following actions:

  # docker_container.foo will be destroyed
  - resource "docker_container" "foo" {
      - attach           = false -> null
      - command          = [
          - "sleep",
          - "3600",
        ] -> null
      - cpu_shares       = 0 -> null
      - dns              = [] -> null
      - dns_opts         = [] -> null
      - dns_search       = [] -> null
      - entrypoint       = [] -> null
      - env              = [] -> null
      - gateway          = "172.17.0.1" -> null
      - group_add        = [] -> null
      - hostname         = "ed65bf8e57fa" -> null
      - id               = "ed65bf8e57faf1420fd6a6071b9e3bbaad2de46dedbfd353a66e954bbd7d0881" -> 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
```

```
      - log_opts           = {} -> null
      - logs               = false -> null
      - max_retry_count    = 0 -> null
      - memory             = 0 -> null
      - memory_swap        = 0 -> null
      - must_run           = true -> null
      - name               = "foo" -> null
      - network_data       = [
          - {
              - gateway                   = "172.17.0.1"
              - global_ipv6_prefix_length = 0
              - ip_address                = "172.17.0.2"
              - ip_prefix_length          = 16
              - network_name              = "bridge"
              # (2 unchanged attributes hidden)
            },
        ] -> null
      - network_mode       = "default" -> null
      - privileged         = false -> null
      - publish_all_ports  = false -> null
      - read_only          = false -> null
      - remove_volumes     = true -> null
      - restart            = "no" -> null
      - rm                 = false -> null
      - runtime            = "runc" -> null
      - security_opts      = [] -> null
      - shm_size           = 64 -> null
      - start              = true -> null
      - stdin_open         = false -> null
      - stop_timeout       = 0 -> null
      - storage_opts       = {} -> null
      - sysctls            = {} -> null
      - tmpfs              = {} -> null
      - tty                = false -> null
      # (8 unchanged attributes hidden)
```

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

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

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

  Enter a value: yes

docker_container.foo: Destroying... [id=ed65bf8e57faf1420fd6a6071b9e3bbaad2de46dedbfd353a66e954bbd7d0881]
docker_container.foo: Destruction complete after 1s
docker_image.ubuntu: Destroying... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_image.ubuntu: Destruction complete after 0s

Destroy complete! Resources: 2 destroyed.
PS C:\Terraform Scripts_SB\Docker>
```

Docker images After Executing Destroy step

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

Experiment 7

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

Theory:

What is SAST?

Static application security testing (SAST), or static analysis, is a testing methodology that analyzes source code to find security vulnerabilities that make your organization's applications susceptible to attack. SAST scans an application before the code is compiled. It's also known as white box testing.

What problems does SAST solve?

SAST takes place very early in the software development life cycle (SDLC) as it does not require a working application and can take place without code being executed. It helps developers identify vulnerabilities in the initial stages of development and quickly resolve issues without breaking builds or passing on vulnerabilities to the final release of the application.

SAST tools give developers real-time feedback as they code, helping them fix issues before they pass the code to the next phase of the SDLC. This prevents security-related issues from being considered an afterthought. SAST tools also provide graphical representations of the issues found, from source to sink. These help you navigate the code easier. Some tools point out the exact location of vulnerabilities and highlight the risky code. Tools can also provide in-depth guidance on how to fix issues and the best place in the code to fix them, without requiring deep security domain expertise.

It's important to note that SAST tools must be run on the application on a regular basis, such as during daily/monthly builds, every time code is checked in, or during a code release.

Why is SAST important?

Developers dramatically outnumber security staff. It can be challenging for an organization to find the resources to perform code reviews on even a fraction of its applications. A key strength of SAST tools is the ability to analyze 100% of the codebase. Additionally, they are much faster than manual secure code reviews performed by humans. These tools can scan millions of lines of code in a matter of minutes. SAST tools automatically identify critical vulnerabilities—such as buffer overflows, SQL injection, cross-site scripting, and others—with high confidence. Thus, integrating static analysis into the SDLC can yield dramatic results in the overall quality of the code developed.

What are the key steps to run SAST effectively?

There are six simple steps needed to perform SAST efficiently in organizations that have a very large number of applications built with different languages, frameworks, and platforms.

1. **Finalize the tool.** Select a static analysis tool that can perform code reviews of applications written in the programming languages you use. The tool should also be able to comprehend the underlying framework used by your software.
2. **Create the scanning infrastructure, and deploy the tool.** This step involves handling the licensing requirements, setting up access control and authorization, and procuring the resources required (e.g., servers and databases) to deploy the tool.
3. **Customize the tool.** Fine-tune the tool to suit the needs of the organization. For example, you might configure it to reduce false positives or find additional security vulnerabilities by writing new rules or updating existing ones. Integrate the tool into the build environment, create dashboards for tracking scan results, and build custom reports.
4. **Prioritize and onboard applications.** Once the tool is ready, onboard your applications. If you have a large number of applications, prioritize the high-risk applications to scan first. Eventually, all your applications should be onboarded and scanned regularly, with application scans synced with release cycles, daily or monthly builds, or code check-ins.
5. **Analyze scan results.** This step involves triaging the results of the scan to remove false positives. Once the set of issues is finalized, they should be tracked and provided to the deployment teams for proper and timely remediation.
6. **Provide governance and training.** Proper governance ensures that your development teams are employing the scanning tools properly. The software security touchpoints should be present within the SDLC. SAST should be incorporated as part of your application development and deployment process.

Integrating Jenkins with SonarQube:

Prerequisites:

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

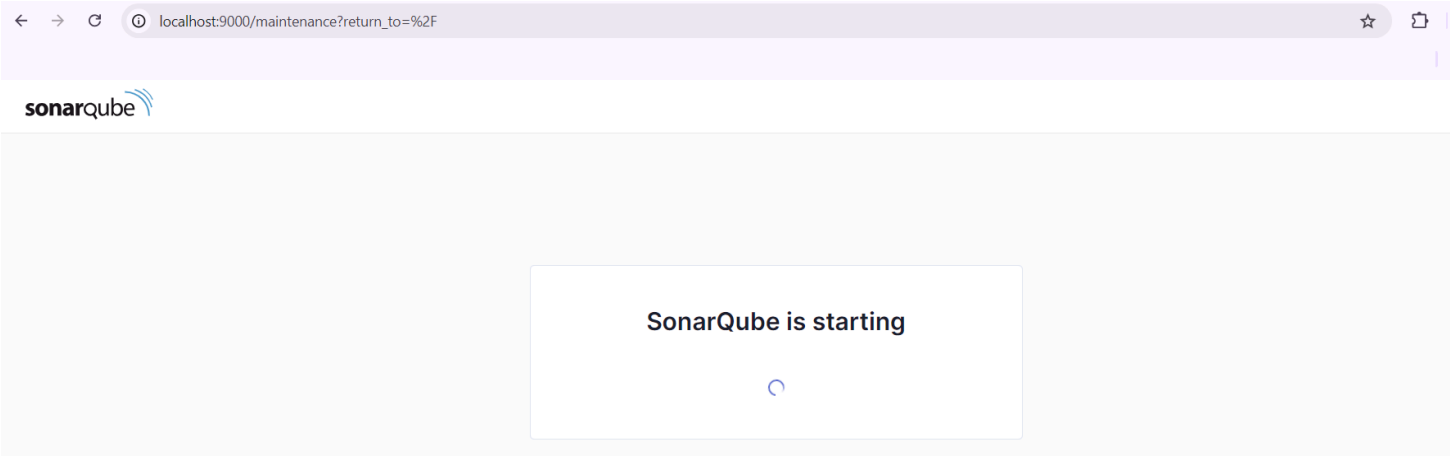
Steps to integrate Jenkins with SonarQube

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

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

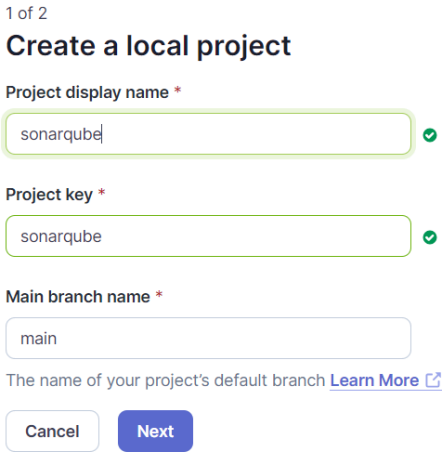
```
C:\Users\ADMIN>docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
Unable to find image 'sonarqube:latest' locally
latest: Pulling from library/sonarqube
7478e0ac0f23: Pull complete
90a925ab929a: Pull complete
7d9a34308537: Pull complete
80338217a4ab: Pull complete
1a5fd5c7e184: Pull complete
7b87d6fa783d: Pull complete
bd819c9b5ead: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:72e9feec71242af83faf65f95a40d5e3bb2822a6c3b2cda8568790f3d31aecde
Status: Downloaded newer image for sonarqube:latest
de76efbeef2054aeb442b86ba54c2916039b8757b388482d9780ffc69f5d8bbe
```

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



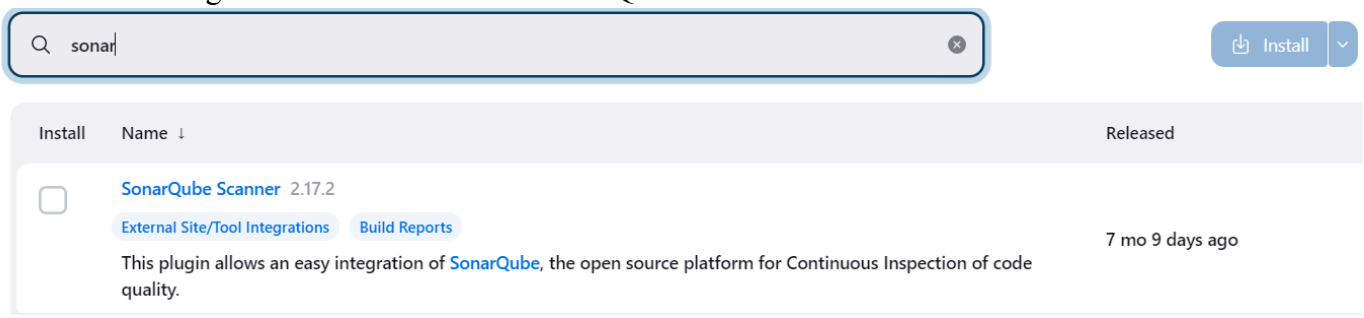
4. Login to SonarQube using username *admin* and password *admin*.

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



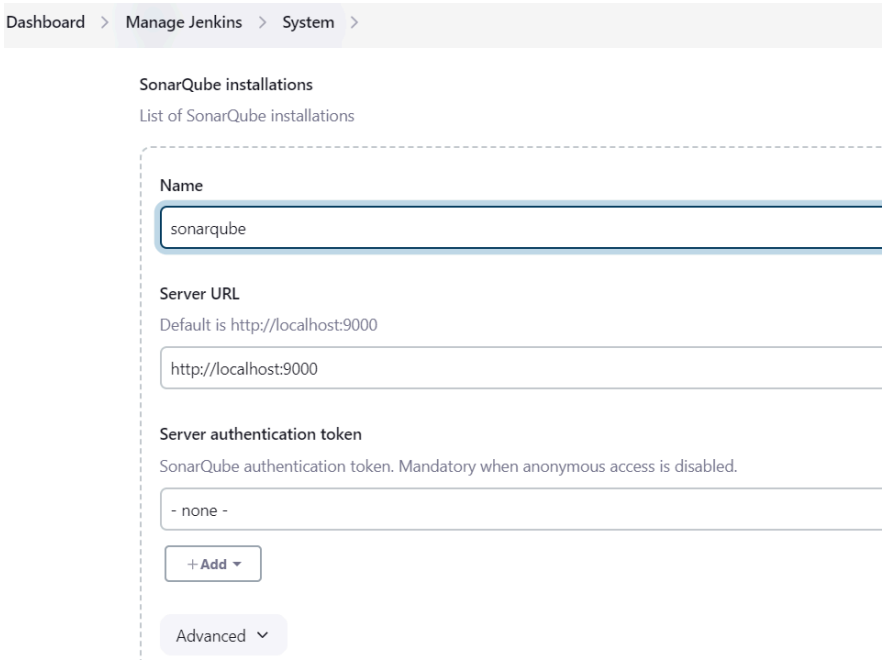
Setup the project and come back to Jenkins Dashboard.

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



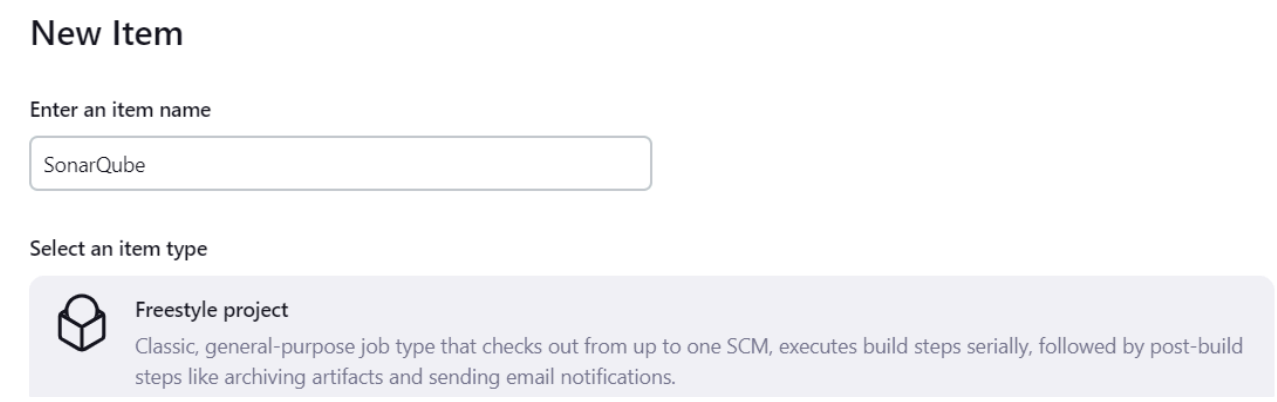
7. Under Jenkins ‘Configure System’, look for SonarQube Servers and enter the details.

   Enter the Server Authentication token if needed.

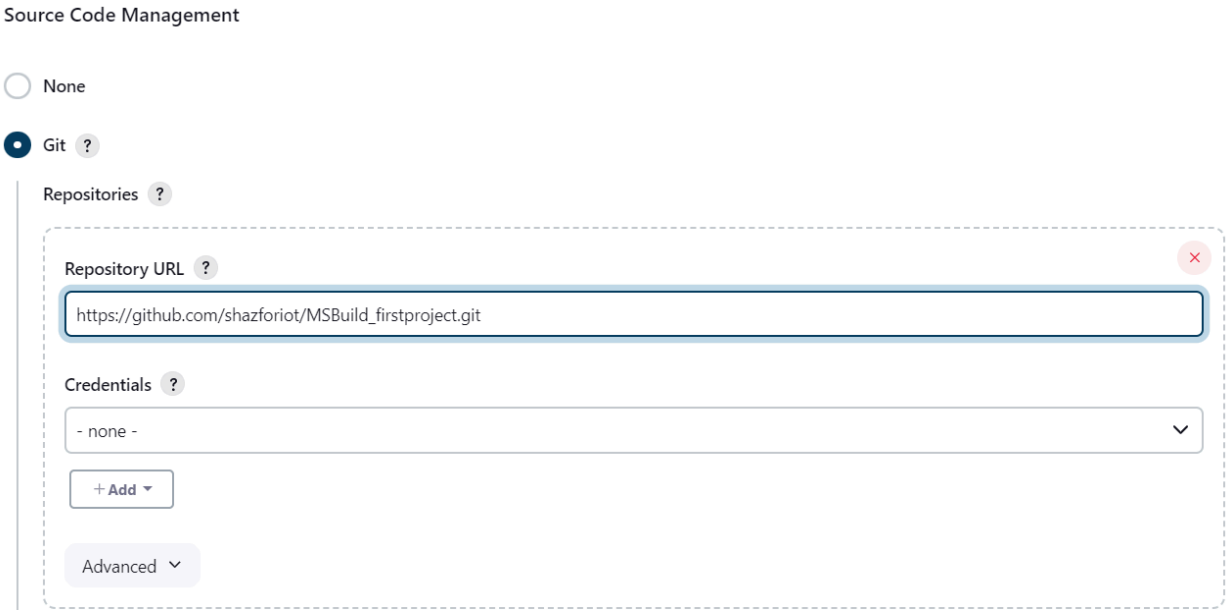


8. Search for SonarQube Scanner under Global Tool Configuration. Choose the latest configuration and choose Install automatically.

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

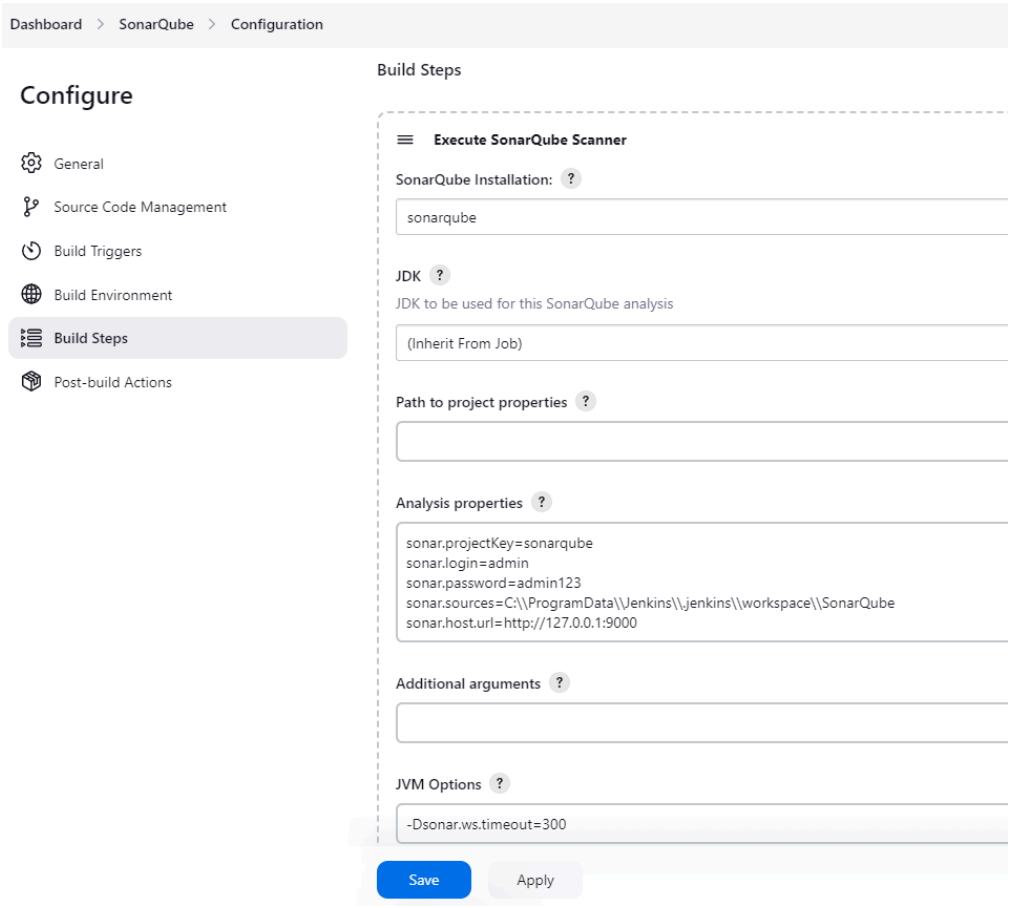


10. Choose this GitHub repository in Source Code Management.
https://github.com/shazforiot/MSBuild_firstproject.git

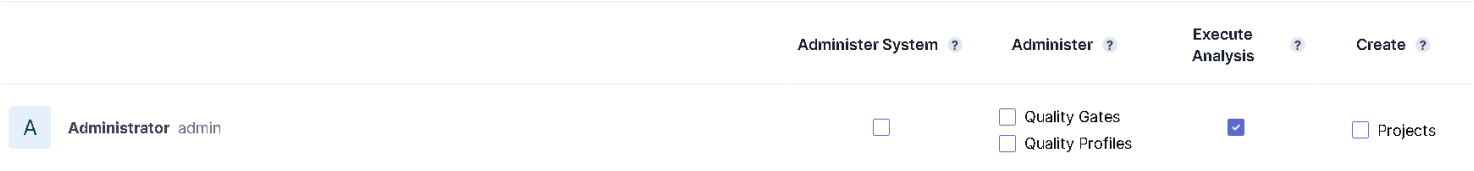


It is a sample hello-world project with no vulnerabilities and issues, just to test the integration.

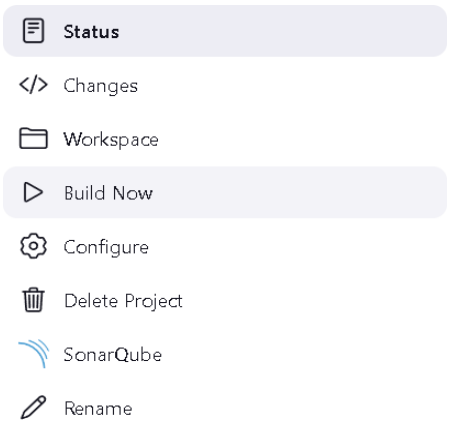
11. Under Build-> Execute SonarQube Scanner, enter these Analysis properties. Mention the SonarQube Project Key, Login, Password, Source path and Host URL.



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



13. Run The Build.



Check the console output.

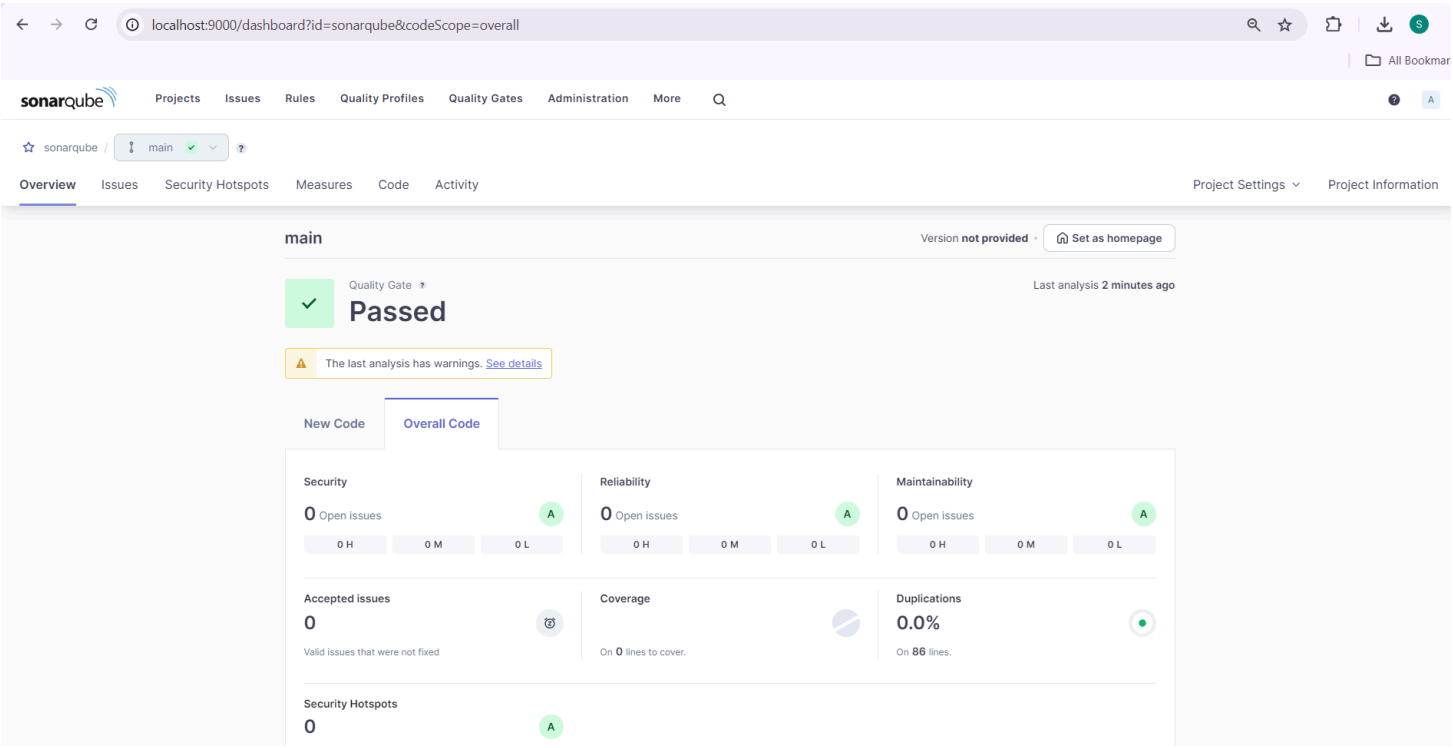
Console Output

Download Copy View as plain tex

```
Started by user Shiven Bansal
Running as SYSTEM
Building on the built-in node in workspace C:\ProgramData\Jenkins\.jenkins\workspace\SonarQube
The recommended git tool is: NONE
No credentials specified
 > git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\SonarQube\.git # timeout=10
Fetching changes from the remote Git repository
 > git.exe config remote.origin.url https://github.com/shazforiot/MSBuild_firstproject.git # timeout=10
Fetching upstream changes from https://github.com/shazforiot/MSBuild_firstproject.git
 > 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.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision f2bc042c04c6e72427c380bcaee6d6fee7b49adf (refs/remotes/origin/master)
 > git.exe config core.sparsecheckout # timeout=10
 > git.exe checkout -f f2bc042c04c6e72427c380bcaee6d6fee7b49adf # timeout=10
Commit message: "updated"
 > git.exe rev-list --no-walk f2bc042c04c6e72427c380bcaee6d6fee7b49adf # timeout=10
[SonarQube] $ C:\ProgramData\Jenkins\.jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\sonarqube\bin\sonar-scanner.bat -Dsonar.host.url=http://localhost:9000 -Dsonar.projectKey=sonarqube -
Dsonar.login=admin -Dsonar.host.url=http://127.0.0.1:9000 -Dsonar.sources=C:\ProgramData\Jenkins\.jenkins\workspace\SonarQube -Dsonar.password=admin123 -
Dsonar.projectBaseDir=C:\ProgramData\Jenkins\.jenkins\workspace\SonarQube
16:16:39.198 WARN  Property 'sonar.host.url' with value 'http://localhost:9000' is overridden with value 'http://127.0.0.1:9000'
16:16:39.206 INFO  Scanner configuration file: C:\ProgramData\Jenkins\.jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\sonarqube\bin\..\conf\sonar-scanner.properties
16:16:39.206 INFO  Project root configuration file: NONE
16:16:39.230 INFO  SonarScanner CLI 6.1.0.4477
16:16:39.230 INFO  Java 21.0.4 Eclipse Adoptium (64-bit)
16:16:39.230 INFO  Windows 11 10.0 amd64
16:16:39.230 INFO  SONAR_SCANNER_OPTS=-Dsonar.ws.timeout=300
16:16:39.254 INFO  User cache: C:\Windows\system32\config\systemprofile\.sonar\cache
```

```
16:16:58.734 INFO  Using git CLI to retrieve untracked files
16:16:58.791 INFO  Analyzing language associated files and files included via "sonar.text.inclusions" that are tracked by git
16:16:58.856 INFO  14 source files to be analyzed
16:16:59.154 INFO  14/14 source files have been analyzed
16:16:59.154 INFO  Sensor TextAndSecretsSensor [text] (done) | time=1306ms
16:16:59.163 INFO  ------------- Run sensors on project
16:16:59.373 INFO  Sensor C# [csharp]
16:16:59.373 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
16:16:59.373 INFO  Sensor C# [csharp] (done) | time=0ms
16:16:59.373 INFO  Sensor Analysis Warnings import [csharp]
16:16:59.379 INFO  Sensor Analysis Warnings import [csharp] (done) | time=0ms
16:16:59.379 INFO  Sensor C# File Caching Sensor [csharp]
16:16:59.379 WARN  Incremental PR analysis: Could not determine common base path, cache will not be computed. Consider setting 'sonar.projectBaseDir' property.
16:16:59.379 INFO  Sensor C# File Caching Sensor [csharp] (done) | time=6ms
16:16:59.379 INFO  Sensor Zero Coverage Sensor
16:16:59.389 INFO  Sensor Zero Coverage Sensor (done) | time=10ms
16:16:59.389 INFO  SCM Publisher SCM provider for this project is: git
16:16:59.389 INFO  SCM Publisher 4 source files to be analyzed
16:16:59.838 INFO  SCM Publisher 4/4 source files have been analyzed (done) | time=449ms
16:16:59.846 INFO  CPD Executor Calculating CPD for 0 files
16:16:59.846 INFO  CPD Executor CPD calculation finished (done) | time=0ms
16:16:59.854 INFO  SCM revision ID 'f2bc042c04c6e72427c380bcaee6d6fee7b49adf'
16:17:00.121 INFO  Analysis report generated in 120ms, dir size=201.1 kB
16:17:00.195 INFO  Analysis report compressed in 57ms, zip size=22.4 kB
16:17:00.393 INFO  Analysis report uploaded in 195ms
16:17:00.394 INFO  ANALYSIS SUCCESSFUL, you can find the results at: http://127.0.0.1:9000/dashboard?id=sonarqube
16:17:00.395 INFO  Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
16:17:00.395 INFO  More about the report processing at http://127.0.0.1:9000/api/ce/task?id=acd819f5-9e70-42ab-bff7-3cc893e2cae4
16:17:00.405 INFO  Analysis total time: 18.743 s
16:17:00.408 INFO  SonarScanner Engine completed successfully
16:17:00.494 INFO  EXECUTION SUCCESS
16:17:00.494 INFO  Total time: 21.288s
Finished: SUCCESS
```

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



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

Conclusion:

In this experiment, I learned how to integrate Jenkins with SonarQube for performing Static Application Security Testing (SAST). I set up SonarQube in a Docker container and configured Jenkins to use the SonarQube scanner. By creating a manual project in SonarQube and configuring the necessary authentication and tools in Jenkins, I established a seamless connection between Jenkins and SonarQube for static code analysis.

I tested the integration with a sample GitHub repository, successfully running a build and analyzing the project's code quality through SonarQube. This hands-on experience enhanced my understanding of the SAST process, Jenkins automation, and SonarQube's capabilities for identifying potential code vulnerabilities.

Name : SHIVEN BANSAL **Roll No : 03** **Class : D15C**

Experiment - 8

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

Theory:

What is SAST?

Static application security testing (SAST), or static analysis, is a testing methodology that analyzes source code to find security vulnerabilities that make your organization's applications susceptible to attack. SAST scans an application before the code is compiled. It's also known as white box testing.

What problems does SAST solve?

SAST takes place very early in the software development life cycle (SDLC) as it does not require a working application and can take place without code being executed. It helps developers identify vulnerabilities in the initial stages of development and quickly resolve issues without breaking builds or passing on vulnerabilities to the final release of the application.

SAST tools give developers real-time feedback as they code, helping them fix issues before they pass the code to the next phase of the SDLC. This prevents security-related issues from being considered an afterthought. SAST tools also provide graphical representations of the issues found, from source to sink. These help you navigate the code easier. Some tools point out the exact location of vulnerabilities and highlight the risky code. Tools can also provide in-depth guidance on how to fix issues and the best place in the code to fix them, without requiring deep security domain expertise.

It's important to note that SAST tools must be run on the application on a regular basis, such as during daily/monthly builds, every time code is checked in, or during a code release.

Why is SAST important?

Developers dramatically outnumber security staff. It can be challenging for an organization to find the resources to perform code reviews on even a fraction of its applications. A key strength of SAST tools is the ability to analyze 100% of the codebase. Additionally, they are much faster than manual secure code reviews performed by humans. These tools can scan millions of lines of code in a matter of minutes. SAST tools automatically identify critical vulnerabilities—such as buffer overflows, SQL injection, cross-site scripting, and others—with high confidence.

What is a CI/CD Pipeline?

CI/CD pipeline refers to the Continuous Integration/Continuous Delivery pipeline. Before we dive deep into this segment, let's first understand what is meant by the term ‘pipeline’?

A pipeline is a concept that introduces a series of events or tasks that are connected in a sequence to make quick software releases. For example, there is a task, that task has got five different stages, and each stage has got some steps. All the steps in phase one have to be completed, to mark the latter stage to be complete.

Now, consider the CI/CD pipeline as the backbone of the DevOps approach. This Pipeline is responsible for building codes, running tests, and deploying new software versions. The Pipeline executes the job in a defined manner by first coding it and then structuring it inside several blocks that may include several steps or tasks.

What is SonarQube?

SonarQube is an open-source platform developed by SonarSource for continuous inspection of code quality. Sonar does static code analysis, which provides a detailed report of bugs, code smells, vulnerabilities, code duplications.
It supports 25+ major programming languages through built-in rulesets and can also be extended with various plugins.

Benefits of SonarQube

- **Sustainability -** Reduces complexity, possible vulnerabilities, and code duplications, optimizing the life of applications.
- **Increase productivity -** Reduces the scale, cost of maintenance, and risk of the application; as such, it removes the need to spend more time changing the code
- **Quality code -** Code quality control is an inseparable part of the process of software development.
- **Detect Errors -** Detects errors in the code and alerts developers to fix them automatically before submitting them for output.
- **Increase consistency -** Determines where the code criteria are breached and enhances the quality
- **Business scaling -** No restriction on the number of projects to be evaluated
- **Enhance developer skills -** Regular feedback on quality problems helps developers to improve their coding skills

Integrating Jenkins with SonarQube:

Prerequisites:

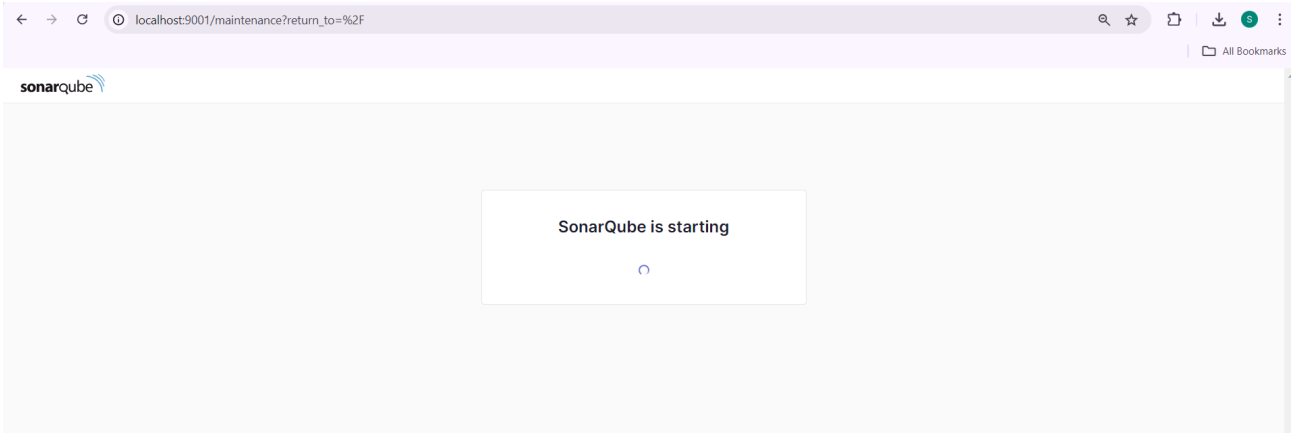
- Jenkins installed
- Docker Installed (for SonarQube)
- SonarQube Docker Image

Steps to create a Jenkins CI/CD Pipeline and use SonarQube to perform SAST

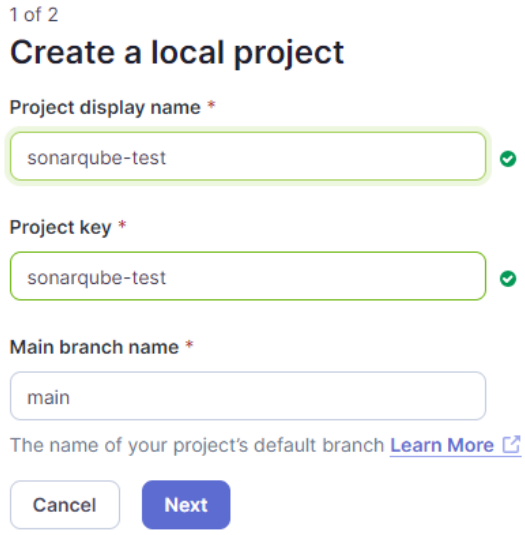
1. Open up Jenkins Dashboard on localhost, port 8080 or whichever port it is at for you.
2. Run SonarQube in a Docker container using this command -

```
C:\Users\ADMIN>docker run -d --name sonarqube2 -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9001:9000 sonarqube:latest
fda86b00e3989f3eb5aca8396b29b2a0adc95bcfe0fc5d85cf1237491e7678b9
```

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

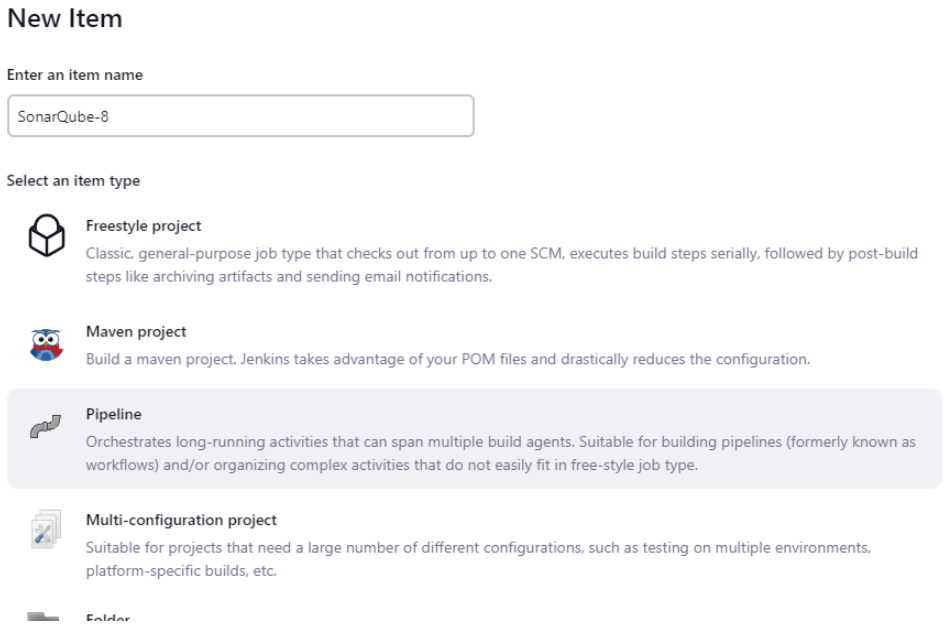


4. Login to SonarQube using username *admin* and password *admin*.
5. Create a manual project in SonarQube with the name **sonarqube-test**



Setup the project and come back to Jenkins Dashboard.

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



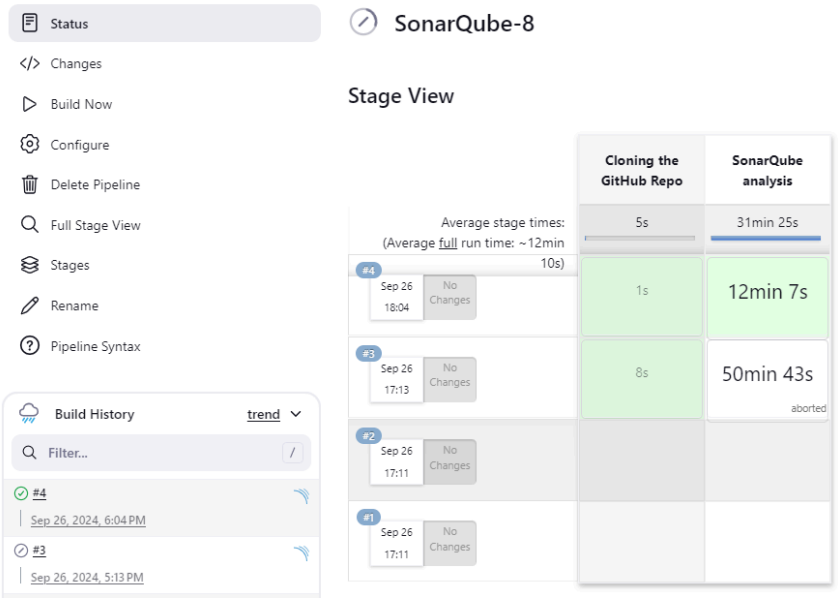
7. Under Pipeline Script, enter the following -

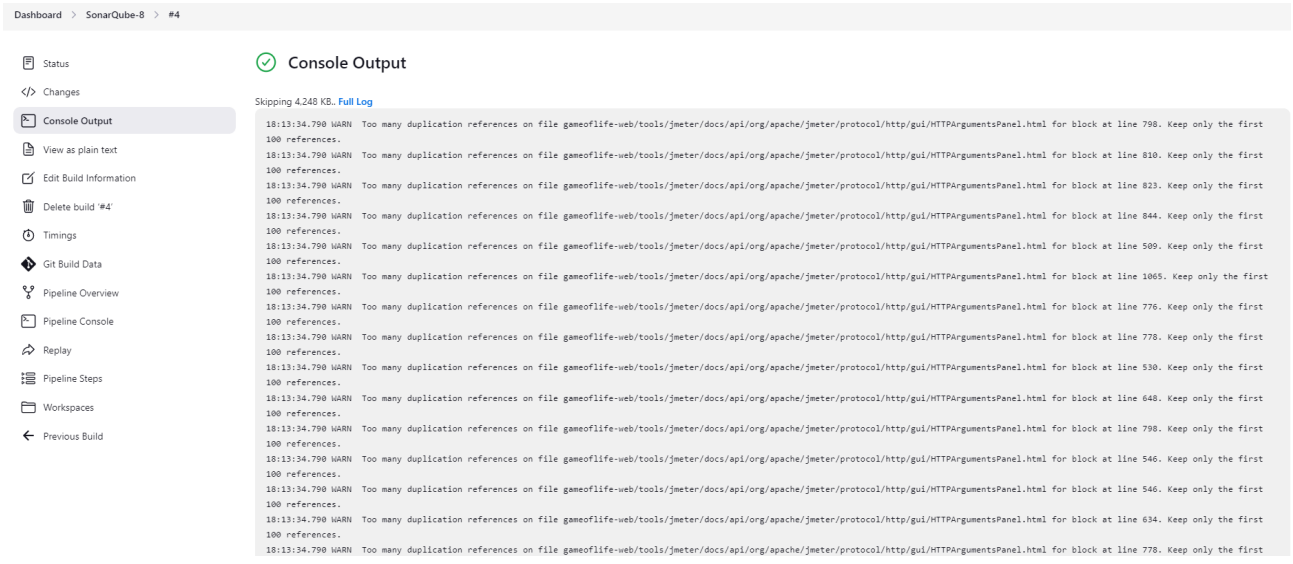
```
node {
  stage('Cloning the GitHub Repo') {
    git 'https://github.com/shazforiot/GOL.git'
  }
  stage('SonarQube analysis') {
    withSonarQubeEnv('sonarqube') {
      sh "<PATH_TO_SONARQUBE_FOLDER>//bin//sonar-scanner \
      -D sonar.login=<SonarQube_USERNAME> \
      -D sonar.password=<SonarQube_PASSWORD> \
      -D sonar.projectKey=<Project_KEY> \
-D sonar.exclusions=vendor/**,resources/**,**/*.java \
      -D sonar.host.url=http://127.0.0.1:9000/"
    }
  }
}
```

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

8. Run The Build.

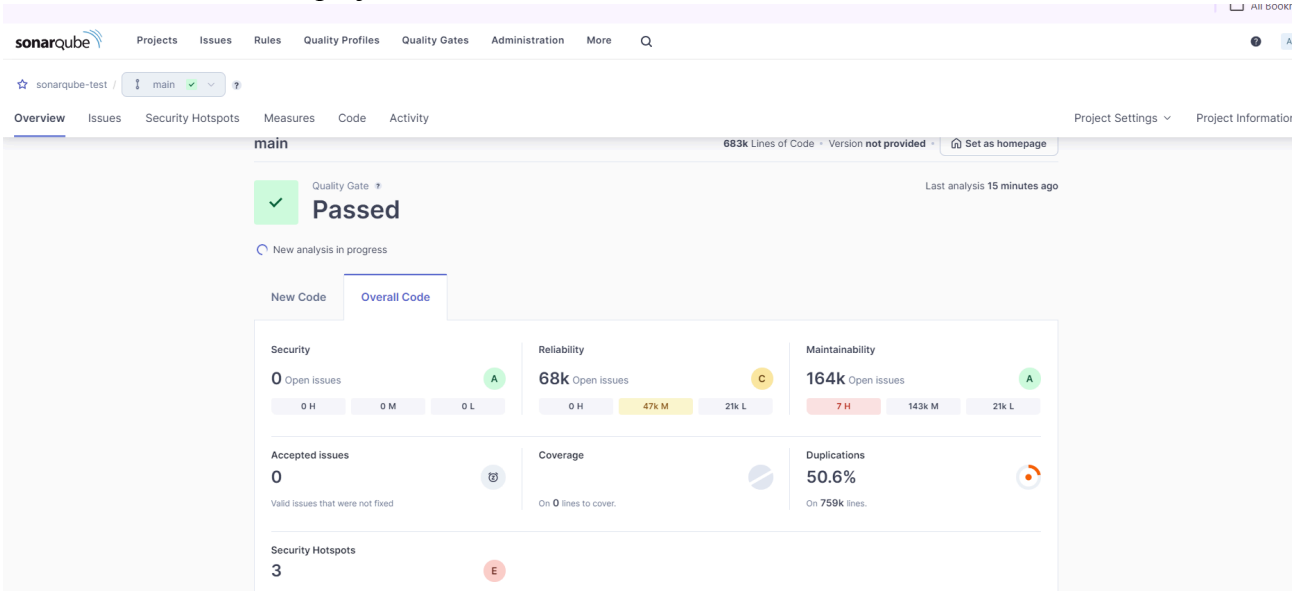
9. Check the console output once the build is complete.





```
references.
18:13:39.657 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/gui/util/TextAreaCellRenderer.html for block at line 75. Keep only the first 100
references.
18:13:39.657 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/gui/util/TextAreaCellRenderer.html for block at line 41. Keep only the first 100
references.
18:13:39.657 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/gui/util/TextAreaCellRenderer.html for block at line 17. Keep only the first 100
references.
18:13:39.657 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/gui/util/TextAreaCellRenderer.html for block at line 296. Keep only the first 100
references.
18:13:39.657 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/gui/util/TextAreaCellRenderer.html for block at line 75. Keep only the first 100
references.
18:13:39.657 INFO  CPD Executor CPD calculation finished (done) | time=158971ms
18:13:39.674 INFO  SCM revision ID 'ba799ba7e1b576f04a4612322b0412c5e6e1e5e4'
18:15:49.696 INFO  Analysis report generated in 5022ms, dir size=127.2 MB
18:16:08.759 INFO  Analysis report compressed in 19048ms, zip size=29.6 MB
18:16:09.884 INFO  Analysis report uploaded in 1125ms
18:16:09.887 INFO  ANALYSIS SUCCESSFUL, you can find the results at: http://127.0.0.1:9001/dashboard?id=sonarqube-test
18:16:09.887 INFO  Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
18:16:09.887 INFO  More about the report processing at http://127.0.0.1:9001/api/ce/task?id=6f22c333-3777-4a21-b058-0ab4c049625c
18:16:22.970 INFO  Analysis total time: 12:02.242 s
18:16:22.975 INFO  SonarScanner Engine completed successfully
18:16:23.699 INFO  EXECUTION SUCCESS
18:16:23.706 INFO  Total time: 12:05.758s
[Pipeline] }
[Pipeline] // withSonarQubeEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

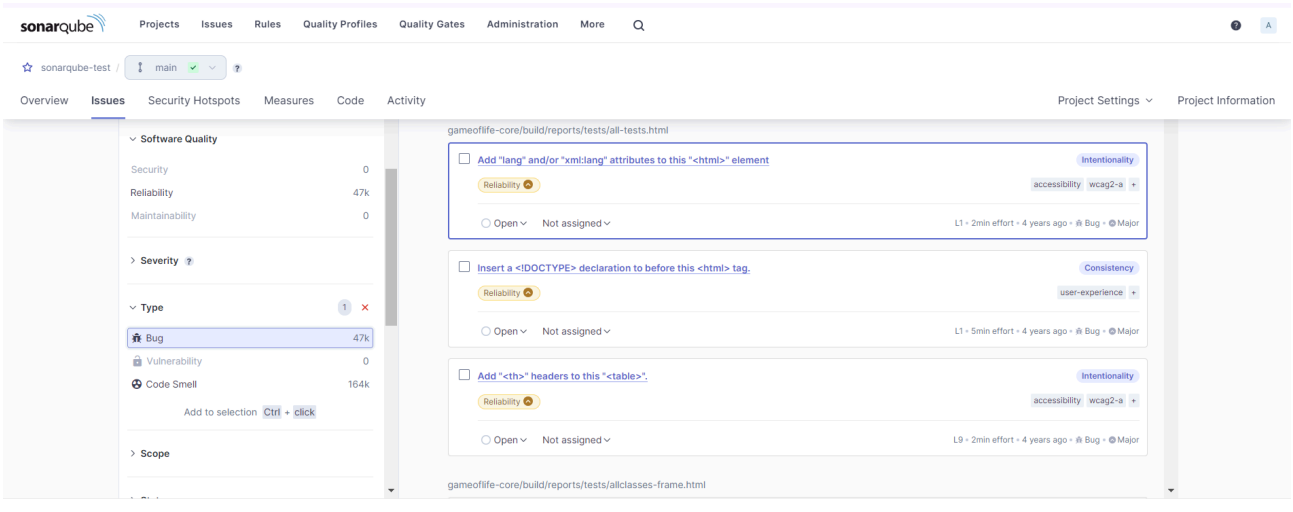
10. After that, check the project in SonarQube.



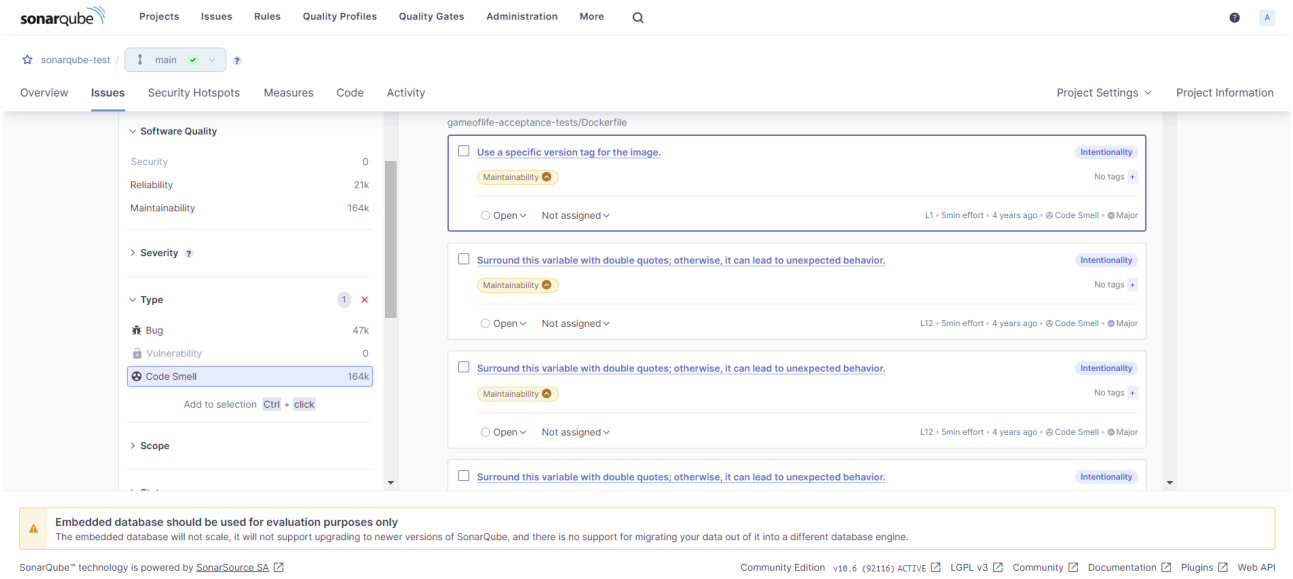
Under different tabs, check all different issues with the code.

11. Code Problems -

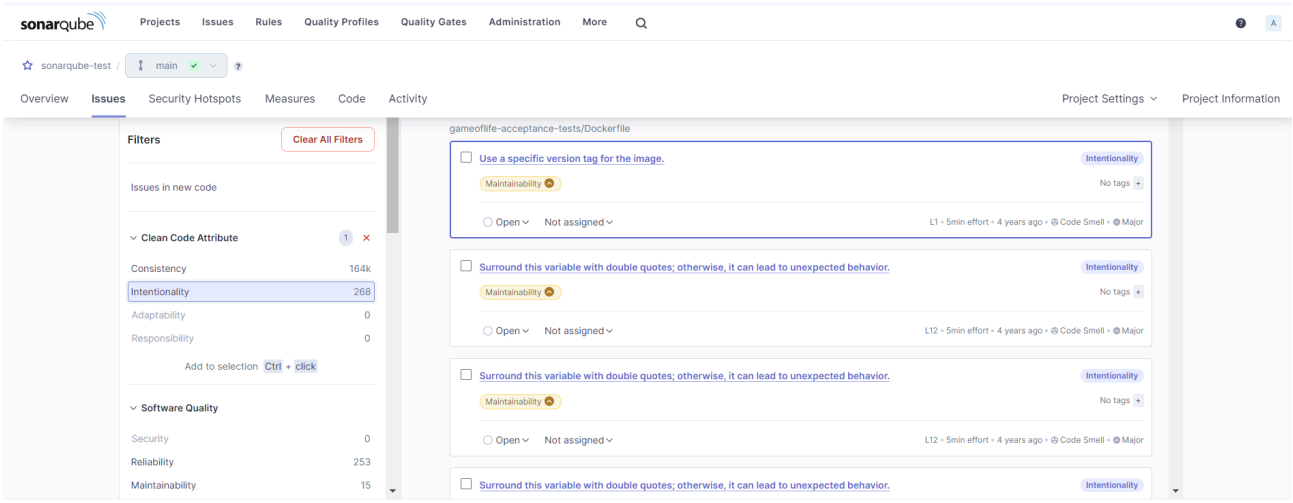
Bugs



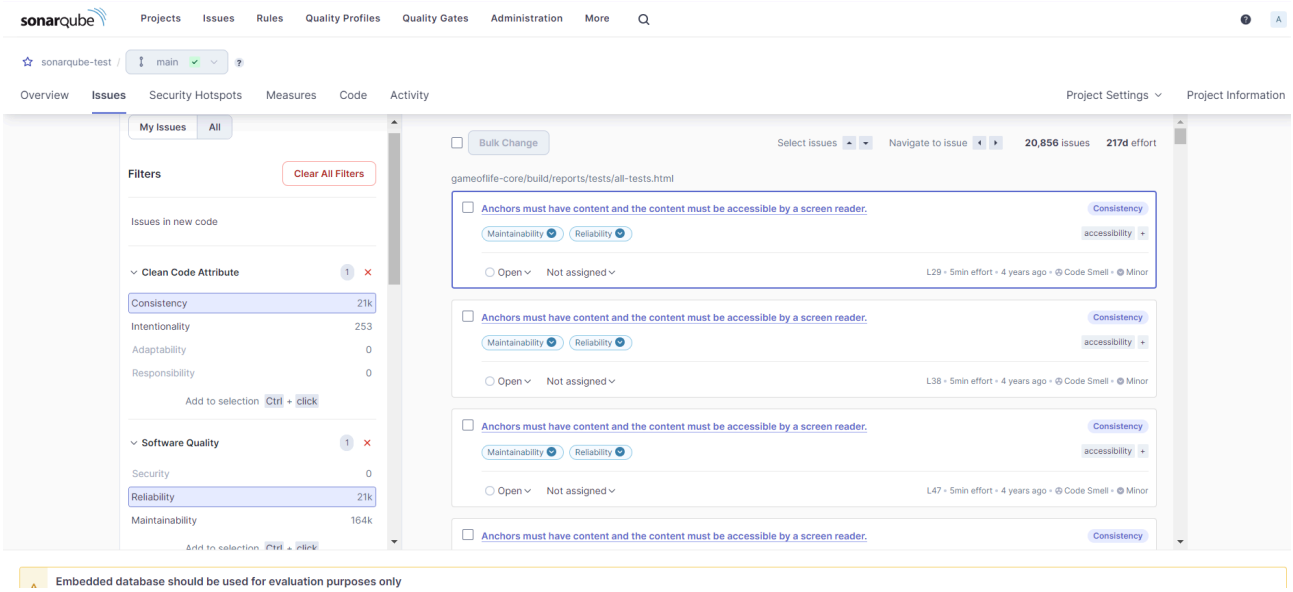
Code Smells



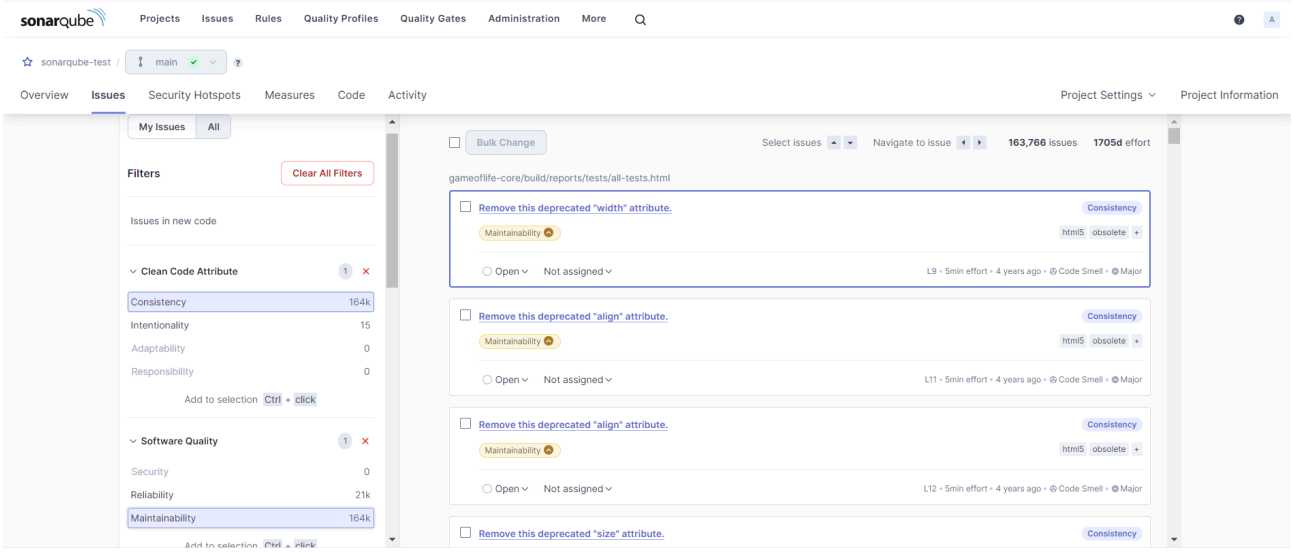
Intentional Issues



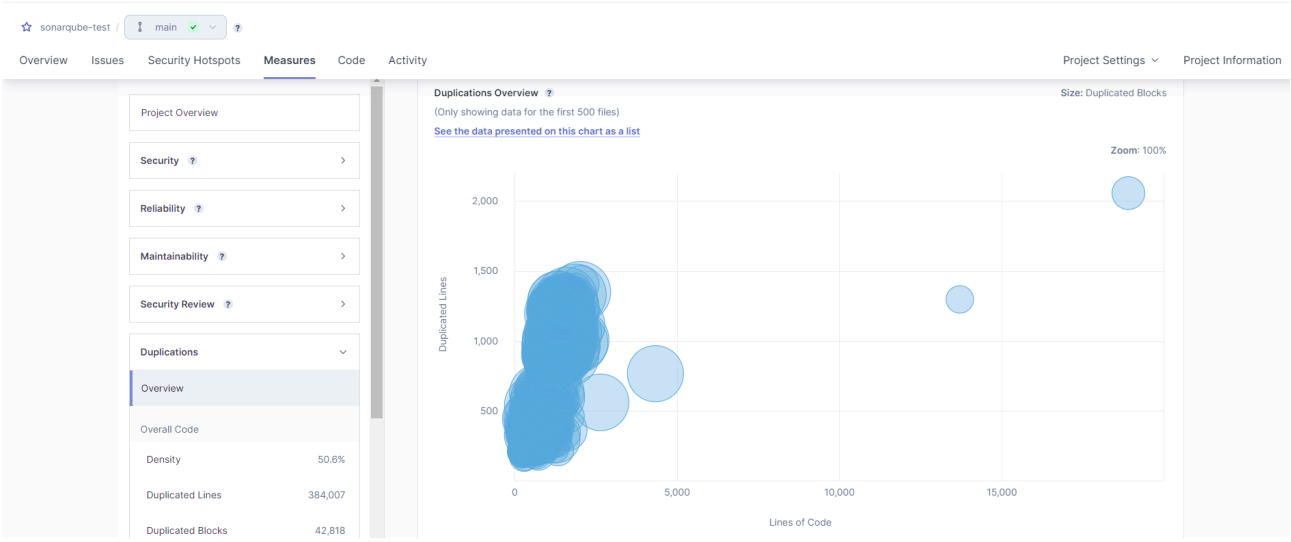
Reliability Issue



Maintainability Issue



Duplicates



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

Conclusion:

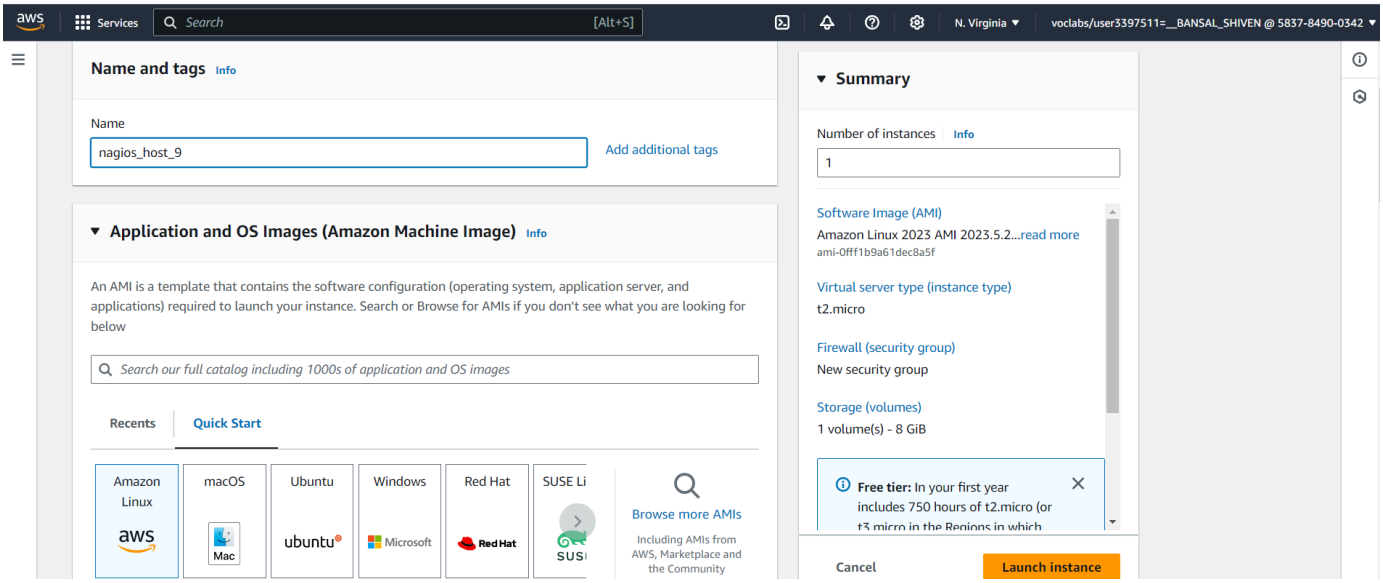
In this experiment, I successfully created a CI/CD pipeline using Jenkins integrated with SonarQube for static code analysis on a sample Java application. I set up SonarQube in a Docker container and configured Jenkins to clone the GitHub repository and perform the analysis. The pipeline detected various issues, including bugs, code smells, and security vulnerabilities, which I reviewed in SonarQube. This experience enhanced my skills in configuring CI/CD tools and highlighted the importance of maintaining code quality through automation. Overall, I gained valuable insights into integrating tools for effective software development practices.

Name : SHIVEN BANSAL **Roll no : 03** **Class : D15C**

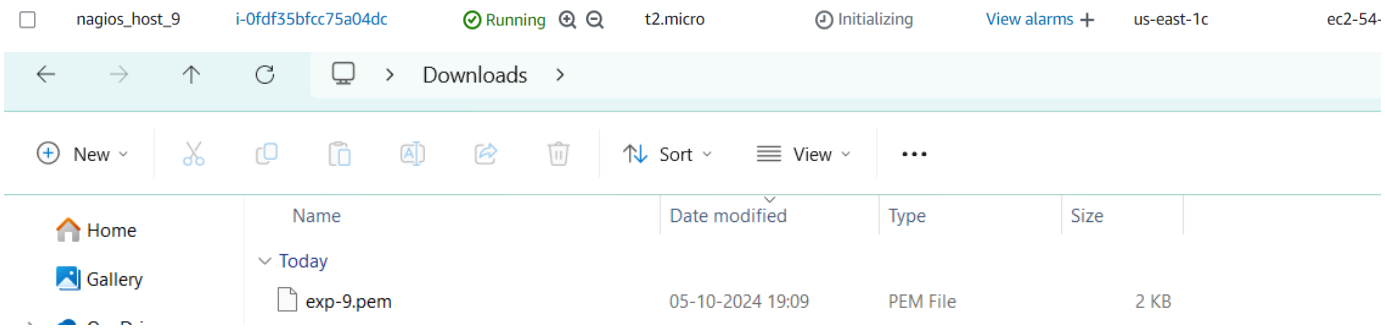
Experiment-9

Steps to perform the experiment:

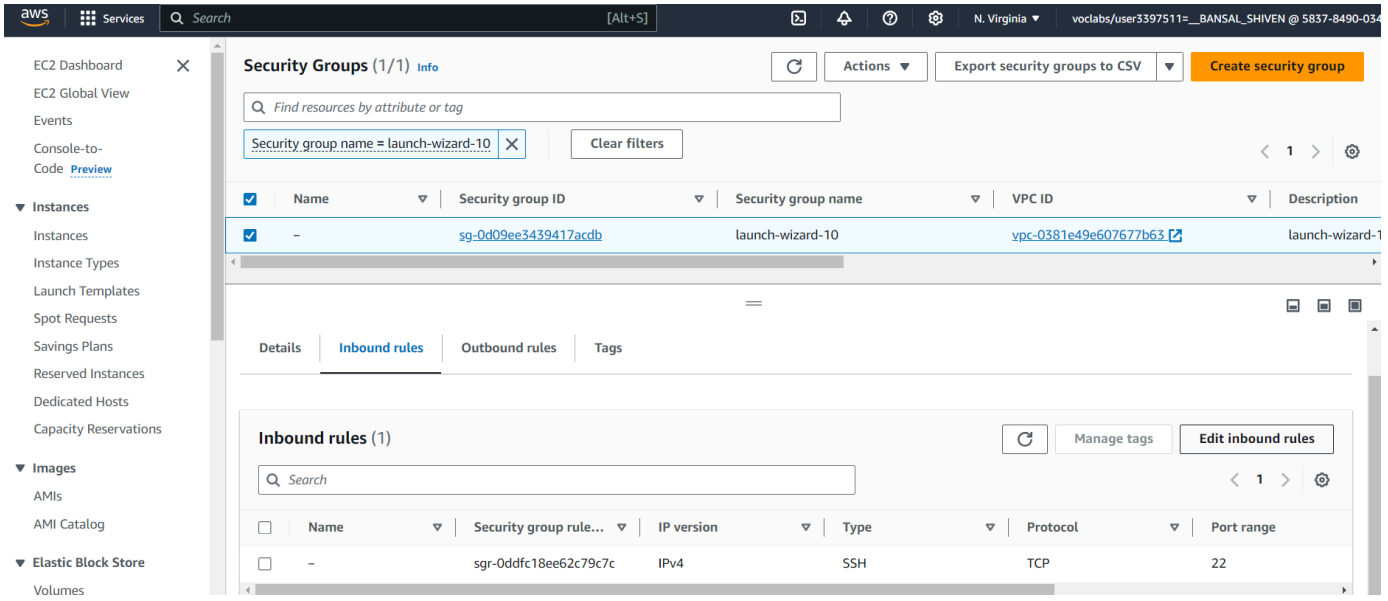
Step1) Create an EC2 instance.keep the settings as default.

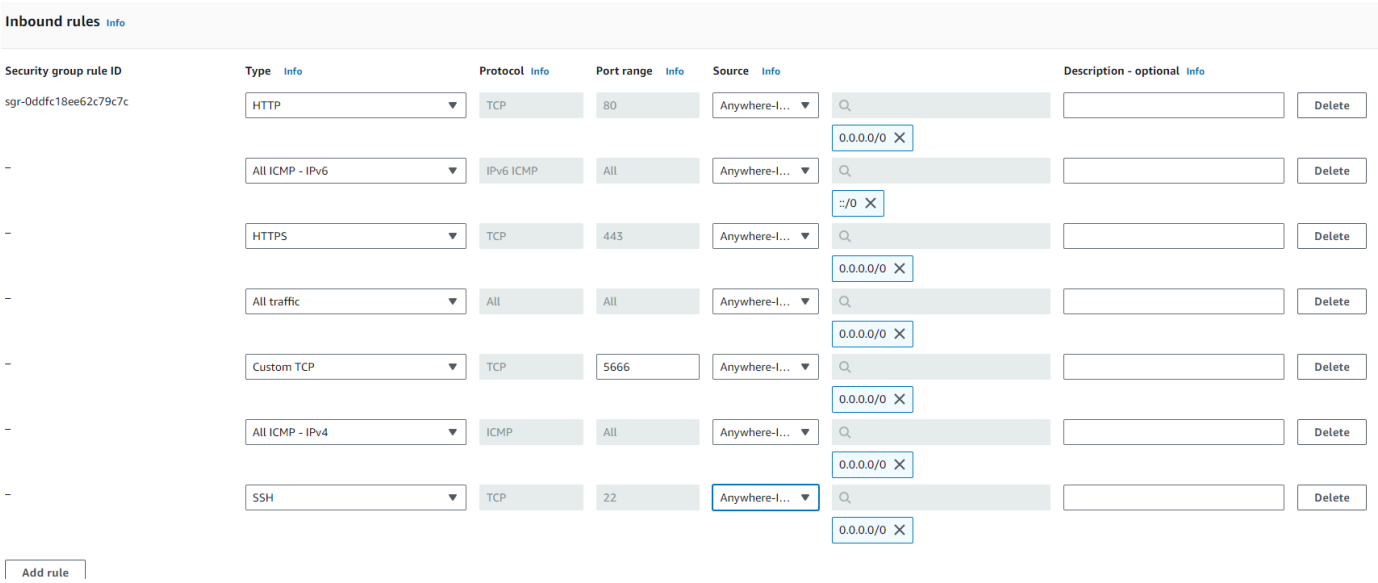


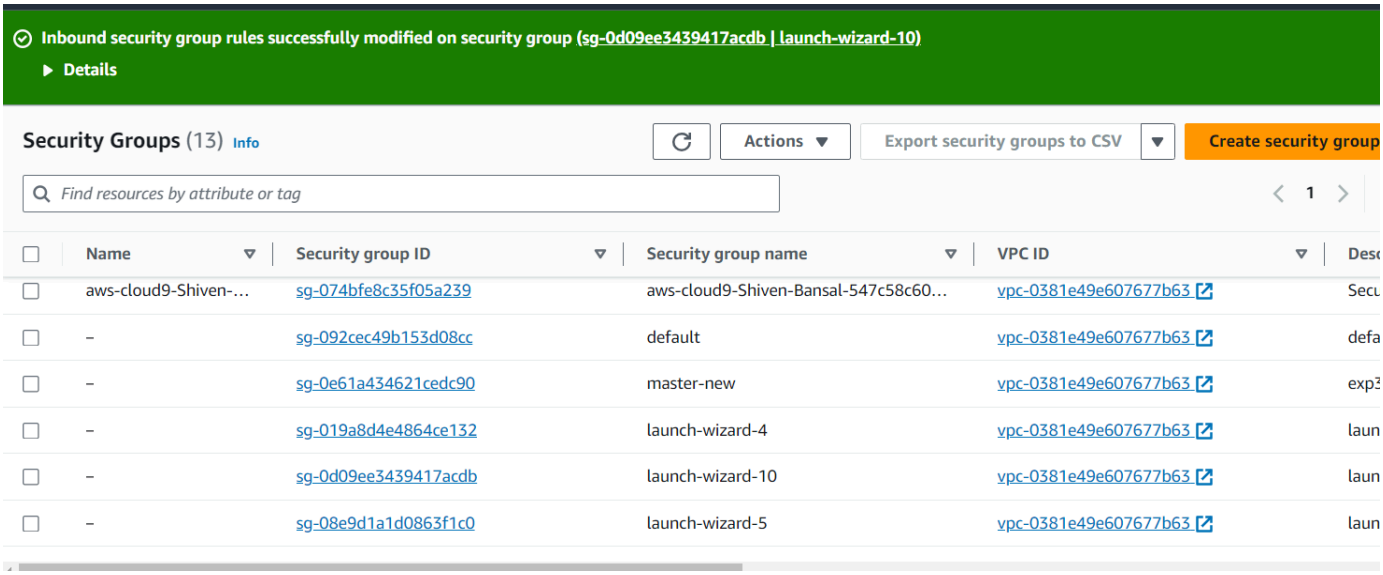
Create a new key pair login and save the downloaded file in a folder of your local desktop.
Also create a new security group. In my case its name will '**launch-wizard-10**'.
Later we will edit rules of this security group.



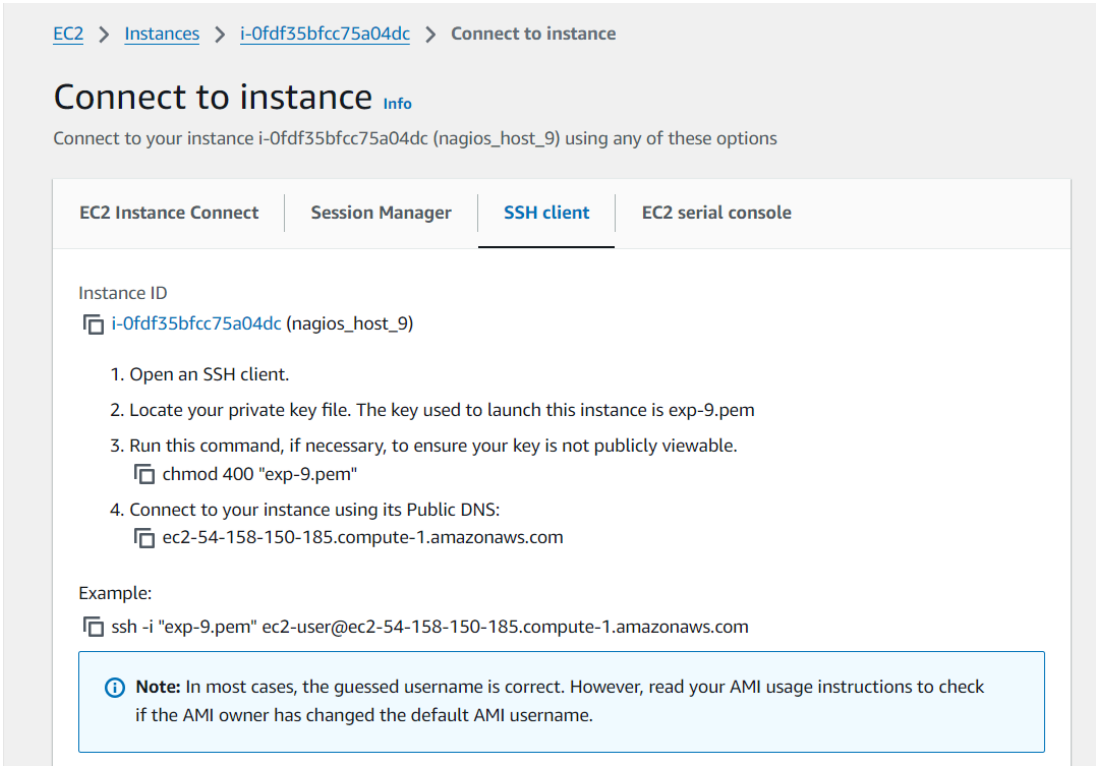
Now to edit security groups,select your security group and click on edit inbound rules.Add these security rules.







now navigate to instances, click on the instance which was created earlier and click on connect. now copy the ssh command and just replace the .pem file with its actual location in your computer.



ssh -i "exp-9.pem" ec2-user@ec2-54-158-150-185.compute-1.amazonaws.com …paste this command in terminal..just replace your.pem file path.

```
C:\Users\ADMIN\Downloads>ssh -i "exp-9.pem" ec2-user@ec2-54-158-150-185.compute-1.amazonaws.com
The authenticity of host 'ec2-54-158-150-185.compute-1.amazonaws.com (54.158.150.185)' can't be established.
ED25519 key fingerprint is SHA256:MLHx7pxctPRettGDRiKHoksaoKOtmmQSzs1BuQKnQeo.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-54-158-150-185.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-31-47-54 ~]$
```

sudo yum update

```
[ec2-user@ip-172-31-47-54 ~]$ sudo yum update
Last metadata expiration check: 0:11:09 ago on Sat Oct  5 13:40:09 2024.
Dependencies resolved.
Nothing to do.
Complete!
```

sudo yum install httpd php

Select y when asked i prompt.

```
[ec2-user@ip-172-31-47-54 ~]$ sudo yum install httpd php
Last metadata expiration check: 0:11:34 ago on Sat Oct  5 13:40:09 2024.
Dependencies resolved.
================================================================================
 Package                 Architecture  Version                    Repository  Size
================================================================================
Installing:
 httpd                   x86_64        2.4.62-1.amzn2023          amazonlinux  48 k
 php8.3                  x86_64        8.3.10-1.amzn2023.0.1      amazonlinux  10 k
Installing dependencies:
 apr                     x86_64        1.7.2-2.amzn2023.0.2       amazonlinux 129 k
 apr-util                x86_64        1.6.3-1.amzn2023.0.1       amazonlinux  98 k
 generic-logos-httpd     noarch        18.0.0-12.amzn2023.0.3     amazonlinux  19 k
 httpd-core              x86_64        2.4.62-1.amzn2023          amazonlinux 1.4 M
 httpd-filesystem        noarch        2.4.62-1.amzn2023          amazonlinux  14 k
 httpd-tools             x86_64        2.4.62-1.amzn2023          amazonlinux  81 k
 libbrotli               x86_64        1.0.9-4.amzn2023.0.2       amazonlinux 315 k
 libsodium               x86_64        1.0.19-4.amzn2023          amazonlinux 176 k
 libxslt                 x86_64        1.1.34-5.amzn2023.0.2      amazonlinux 241 k
 mailcap                 noarch        2.1.49-3.amzn2023.0.3      amazonlinux  33 k
 nginx-filesystem        noarch        1:1.24.0-1.amzn2023.0.4    amazonlinux 9.8 k
 php8.3-cli              x86_64        8.3.10-1.amzn2023.0.1      amazonlinux 3.7 M
 php8.3-common           x86_64        8.3.10-1.amzn2023.0.1      amazonlinux 737 k
 php8.3-process          x86_64        8.3.10-1.amzn2023.0.1      amazonlinux  45 k
 php8.3-xml              x86_64        8.3.10-1.amzn2023.0.1      amazonlinux 154 k
Installing weak dependencies:
 apr-util-openssl        x86_64        1.6.3-1.amzn2023.0.1       amazonlinux  17 k
 mod_http2               x86_64        2.0.27-1.amzn2023.0.3      amazonlinux 166 k
 mod_lua                 x86_64        2.4.62-1.amzn2023          amazonlinux  61 k
 php8.3-fpm              x86_64        8.3.10-1.amzn2023.0.1      amazonlinux 1.9 M
 php8.3-mbstring         x86_64        8.3.10-1.amzn2023.0.1      amazonlinux 528 k
 php8.3-opcache          x86_64        8.3.10-1.amzn2023.0.1      amazonlinux 379 k
 php8.3-pdo              x86_64        8.3.10-1.amzn2023.0.1      amazonlinux  89 k
 php8.3-sodium           x86_64        8.3.10-1.amzn2023.0.1      amazonlinux  41 k

Transaction Summary
================================================================================
Install  25 Packages
```

sudo yum install gcc glibc glibc-common

```
[ec2-user@ip-172-31-47-54 ~]$ sudo yum install gcc glibc glibc-common
Last metadata expiration check: 0:15:43 ago on Sat Oct  5 13:40:09 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
================================================================================
Installing:
 gcc                     x86_64              11.4.1-2.amzn2023.0.2   amazonlinux
Installing dependencies:
 annobin-docs            noarch              10.93-1.amzn2023.0.1    amazonlinux
 annobin-plugin-gcc      x86_64              10.93-1.amzn2023.0.1    amazonlinux
 cpp                     x86_64              11.4.1-2.amzn2023.0.2   amazonlinux
 gc                      x86_64              8.0.4-5.amzn2023.0.2    amazonlinux
 glibc-devel             x86_64              2.34-52.amzn2023.0.11   amazonlinux
 glibc-headers-x86       noarch              2.34-52.amzn2023.0.11   amazonlinux
 guile22                 x86_64              2.2.7-2.amzn2023.0.3    amazonlinux
 kernel-headers          x86_64              6.1.109-118.189.amzn2023 amazonlinux
 libmpc                  x86_64              1.2.1-2.amzn2023.0.2    amazonlinux
 libtool-ltdl            x86_64              2.4.7-1.amzn2023.0.3    amazonlinux
 libxcrypt-devel         x86_64              4.4.33-7.amzn2023       amazonlinux
 make                    x86_64              1:4.3-5.amzn2023.0.2    amazonlinux

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

sudo yum install gd gd-devel

```
[ec2-user@ip-172-31-47-54 ~]$ sudo yum install gd gd-devel
Last metadata expiration check: 0:16:31 ago on Sat Oct  5 13:40:09 2024.
Dependencies resolved.
================================================================================
 Package                 Architecture        Version                 Repository                 S
================================================================================
Installing:
 gd                      x86_64              2.3.3-5.amzn2023.0.3    amazonlinux                13
 gd-devel                x86_64              2.3.3-5.amzn2023.0.3    amazonlinux                 3
Installing dependencies:
 brotli                  x86_64              1.0.9-4.amzn2023.0.2    amazonlinux                31
 brotli-devel            x86_64              1.0.9-4.amzn2023.0.2    amazonlinux                 3
 bzip2-devel             x86_64              1.0.8-6.amzn2023.0.2    amazonlinux                21
 cairo                   x86_64              1.17.6-2.amzn2023.0.1   amazonlinux                68
 cmake-filesystem        x86_64              3.22.2-1.amzn2023.0.4   amazonlinux                 1
 fontconfig              x86_64              2.13.94-2.amzn2023.0.2  amazonlinux                27
 fontconfig-devel        x86_64              2.13.94-2.amzn2023.0.2  amazonlinux                12
 fonts-filesystem        noarch              1:2.0.5-12.amzn2023.0.2 amazonlinux                9.
 freetype                x86_64              2.13.2-5.amzn2023.0.1   amazonlinux                42
 freetype-devel          x86_64              2.13.2-5.amzn2023.0.1   amazonlinux                91
 glib2-devel             x86_64              2.74.7-689.amzn2023.0.2 amazonlinux                48
 google-noto-fonts-common noarch             20201206-2.amzn2023.0.2 amazonlinux                 1
```

sudo adduser -m
nagios sudo passwd

```
[ec2-user@ip-172-31-47-54 ~]$ sudo adduser -m nagios
[ec2-user@ip-172-31-47-54 ~]$ sudo passwd nagios
Changing password for user nagios.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
Sorry, passwords do not match.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: all authentication tokens updated successfully.
[ec2-user@ip-172-31-47-54 ~]$
```

nagios

sudo groupadd nagcmd

```
[ec2-user@ip-172-31-47-54 ~]$ sudo groupadd nagcmd
```

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

```
[ec2-user@ip-172-31-47-54 ~]$ sudo usermod -a -G nagcmd nagios
[ec2-user@ip-172-31-47-54 ~]$ sudo usermod -a -G nagcmd apache
[ec2-user@ip-172-31-47-54 ~]$
```

mkdir ~/downloads
cd ~/downloads

```
[ec2-user@ip-172-31-47-54 ~]$ mkdir ~/downloads
[ec2-user@ip-172-31-47-54 ~]$ cd ~/downloads
[ec2-user@ip-172-31-47-54 downloads]$
```

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

```
[ec2-user@ip-172-31-47-54 downloads]$ wget https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.5.tar.gz
--2024-10-05 14:08:52--  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  5.22MB/s    in 0.4s

2024-10-05 14:08:53 (5.22 MB/s) - 'nagios-4.5.5.tar.gz' saved [2065473/2065473]

[ec2-user@ip-172-31-47-54 downloads]$
```

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

```
[ec2-user@ip-172-31-47-54 downloads]$ wget https://nagios-plugins.org/download/nagios-plugins-2.4.11.tar.gz
--2024-10-05 14:09:32--  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  7.35MB/s    in 0.4s

2024-10-05 14:09:33 (7.35 MB/s) - 'nagios-plugins-2.4.11.tar.gz' saved [2753049/2753049]

[ec2-user@ip-172-31-47-54 downloads]$
```

tar zxvf nagios-4.5.5.tar.gz

```
[ec2-user@ip-172-31-47-54 downloads]$ tar zxvf nagios-4.5.5.tar.gz
nagios-4.5.5/
nagios-4.5.5/.github/
nagios-4.5.5/.github/workflows/
nagios-4.5.5/.github/workflows/test.yml
nagios-4.5.5/.gitignore
nagios-4.5.5/CONTRIBUTING.md
nagios-4.5.5/Changelog
nagios-4.5.5/INSTALLING
nagios-4.5.5/LEGAL
nagios-4.5.5/LICENSE
nagios-4.5.5/Makefile.in
nagios-4.5.5/README.md
nagios-4.5.5/THANKS
nagios-4.5.5/UPGRADING
nagios-4.5.5/aclocal.m4
nagios-4.5.5/autoconf-macros/
nagios-4.5.5/autoconf-macros/.gitignore
nagios-4.5.5/autoconf-macros/CHANGELOG.md
nagios-4.5.5/autoconf-macros/LICENSE
nagios-4.5.5/autoconf-macros/LICENSE.md
nagios-4.5.5/autoconf-macros/README.md
nagios-4.5.5/autoconf-macros/add_group_user
```

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

- **commands to enter:**

  ls (verify whether nagios-4.5.5 exists).Then go inside nagios 4.5.5 using cd.

```
[ec2-user@ip-172-31-47-54 downloads]$ ls
nagios-4.5.5  nagios-4.5.5.tar.gz  nagios-plugins-2.4.11.tar.gz
[ec2-user@ip-172-31-47-54 downloads]$ cd nagios-4.5.5/
[ec2-user@ip-172-31-47-54 nagios-4.5.5]$
```

we now have to install openssl dev library

The OpenSSL development library, or openssl-devel contains include files that help develop applications that use cryptographic algorithms and protocols

commands to enter:

sudo yum install openssl-devel

```
[ec2-user@ip-172-31-47-54 nagios-4.5.5]$ sudo yum install openssl-devel
Last metadata expiration check: 0:31:58 ago on Sat Oct  5 13:40:09 2024.
Dependencies resolved.
================================================================================
 Package              Architecture          Version                    Repository                Size
================================================================================
Installing:
 openssl-devel        x86_64                1:3.0.8-1.amzn2023.0.14    amazonlinux               3.0 M

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

Total download size: 3.0 M
Installed size: 4.7 M
Is this ok [y/N]: y
Downloading Packages:
openssl-devel-3.0.8-1.amzn2023.0.14.x86_64.rpm                 31 MB/s | 3.0 MB     00:00
--------------------------------------------------------------------------------
Total                                                          21 MB/s | 3.0 MB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                            1/1
  Installing       : openssl-devel-1:3.0.8-1.amzn2023.0.14.x86_64               1/1
  Running scriptlet: openssl-devel-1:3.0.8-1.amzn2023.0.14.x86_64               1/1
  Verifying        : openssl-devel-1:3.0.8-1.amzn2023.0.14.x86_64               1/1

Installed:
  openssl-devel-1:3.0.8-1.amzn2023.0.14.x86_64

Complete!
```

Then finally we can run the commands like usual.

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

```
[ec2-user@ip-172-31-47-54 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
```

make all

```
[ec2-user@ip-172-31-47-54 nagios-4.5.5]$ make all
cd ./base && make
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/base'
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o nagios.o ./nagios.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o broker.o broker.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o nebmods.o nebmods.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o ../common/shared.o ../common/shared.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o query-handler.o query-handler.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o workers.o workers.c
In function ‘get_wproc_list’,
    inlined from ‘get_worker’ at workers.c:277:12:
workers.c:253:17: warning: ‘%s’ directive argument is null [-Wformat-overflow=]
  253 |                 log_debug_info(DEBUGL_CHECKS, 1, "Found specialized worker(s) for '%s'", (slash && *slash != '/') ? slash : cmd_name);
      |                 ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o checks.o checks.c
```

```
*** Support Notes *******************************************

If you have questions about configuring or running Nagios,
please make sure that you:

     - Look at the sample config files
     - Read the documentation on the Nagios Library at:
           https://library.nagios.com

before you post a question to one of the mailing lists.
Also make sure to include pertinent information that could
help others help you.  This might include:

     - What version of Nagios you are using
     - What version of the plugins you are using
     - Relevant snippets from your config files
     - Relevant error messages from the Nagios log file

For more information on obtaining support for Nagios, visit:

       https://support.nagios.com

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

Enjoy.
```

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

```
[ec2-user@ip-172-31-47-54 nagios-4.5.5]$ sudo make install
sudo make install-init
sudo make install-config
sudo make install-commandmode
cd ./base && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/base'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagios /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagiostats /usr/local/nagios/bin
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5/base'
cd ./cgi && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
make install-basic
make[2]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/sbin
for file in *.cgi; do \
        /usr/bin/install -c -s -m 775 -o nagios -g nagios $file /usr/local/nagios/sbin; \
done
make[2]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
cd ./html && make install
```

Now the next command will take us to nano editor:

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

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



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

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

define contact {

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



##############################################################################
                                [ Read 51 lines ]
^G Help          ^O Write Out     ^W Where Is      ^K Cut           ^T Execute       ^C Location      M-U Undo         M-A Set Mark     M-] To Bracket
^X Exit          ^R Read File     ^\ Replace       ^U Paste         ^J Justify       ^/ Go To Line    M-E Redo         M-6 Copy         ^Q Where Was
```

Change your email

```
define contact {

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

Press crtl + O and
enter Press crtl + X

```
[ec2-user@ip-172-31-47-54 nagios-4.5.5]$ sudo nano /usr/local/nagios/etc/objects/contacts.cfg
[ec2-user@ip-172-31-47-54 nagios-4.5.5]$
```

sudo make install-webconf

```
[ec2-user@ip-172-31-47-54 nagios-4.5.5]$ sudo make install-webconf
/usr/bin/install -c -m 644 sample-config/httpd.conf /etc/httpd/conf.d/nagios.conf
if [ 0 -eq 1 ]; then \
        ln -s /etc/httpd/conf.d/nagios.conf /etc/apache2/sites-enabled/nagios.conf;
fi

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

Adding password for nagios admin

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

```
[ec2-user@ip-172-31-47-54 nagios-4.5.5]$ sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
New password:
Re-type new password:
Adding password for user nagiosadmin
[ec2-user@ip-172-31-47-54 nagios-4.5.5]$
```

sudo service httpd restart

```
[ec2-user@ip-172-31-47-54 nagios-4.5.5]$ sudo service httpd restart
Redirecting to /bin/systemctl restart httpd.service
[ec2-user@ip-172-31-47-54 nagios-4.5.5]$
```

cd ~/downloads

tar zxvf nagios-plugins-2.4.11.tar.gz

```
[ec2-user@ip-172-31-47-54 nagios-4.5.5]$ cd ~/downloads
tar zxvf nagios-plugins-2.4.11.tar.gz
nagios-plugins-2.4.11/
nagios-plugins-2.4.11/build-aux/
nagios-plugins-2.4.11/build-aux/compile
nagios-plugins-2.4.11/build-aux/config.guess
nagios-plugins-2.4.11/build-aux/config.rpath
nagios-plugins-2.4.11/build-aux/config.sub
nagios-plugins-2.4.11/build-aux/install-sh
nagios-plugins-2.4.11/build-aux/ltmain.sh
nagios-plugins-2.4.11/build-aux/missing
nagios-plugins-2.4.11/build-aux/mkinstalldirs
nagios-plugins-2.4.11/build-aux/depcomp
nagios-plugins-2.4.11/build-aux/snippet/
nagios-plugins-2.4.11/build-aux/snippet/_Noreturn.h
```

cd nagios-plugins-2.4.11

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

```
[ec2-user@ip-172-31-47-54 downloads]$ cd nagios-plugins-2.4.11
./configure --with-nagios-user=nagios --with-nagios-group=nagios
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /usr/bin/mkdir -p
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking whether make supports nested variables... yes
checking whether to enable maintainer-specific portions of Makefiles... yes
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
```

make

sudo make install

```
[ec2-user@ip-172-31-47-54 nagios-plugins-2.4.11]$ make
sudo make install
make  all-recursive
make[1]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11'
Making all in gl
make[2]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11/gl'
rm -f alloca.h-t alloca.h && \
{ echo '/* DO NOT EDIT! GENERATED AUTOMATICALLY! */'; \
  cat ./alloca.in.h; \
} > alloca.h-t && \
mv -f alloca.h-t alloca.h
rm -f c++defs.h-t c++defs.h && \
sed -n -e '/_GL_CXXDEFS/,$p' \
  < ../build-aux/snippet/c++defs.h \
  > c++defs.h-t && \
```

sudo chkconfig --add nagios

sudo chkconfig nagios on

```
[ec2-user@ip-172-31-47-54 nagios-plugins-2.4.11]$ sudo chkconfig --add nagios
sudo chkconfig nagios on
error reading information on service nagios: No such file or directory
Note: Forwarding request to 'systemctl enable nagios.service'.
Created symlink /etc/systemd/system/multi-user.target.wants/nagios.service → /usr/lib/systemd/system/nagios.service.
[ec2-user@ip-172-31-47-54 nagios-plugins-2.4.11]$
```

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

```
[ec2-user@ip-172-31-47-54 nagios-plugins-2.4.11]$ sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

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

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

Running pre-flight check on configuration data...

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

Total Warnings: 0
Total Errors:   0

Things look okay - No serious problems were detected during the pre-flight check
[ec2-user@ip-172-31-47-54 nagios-plugins-2.4.11]$
```

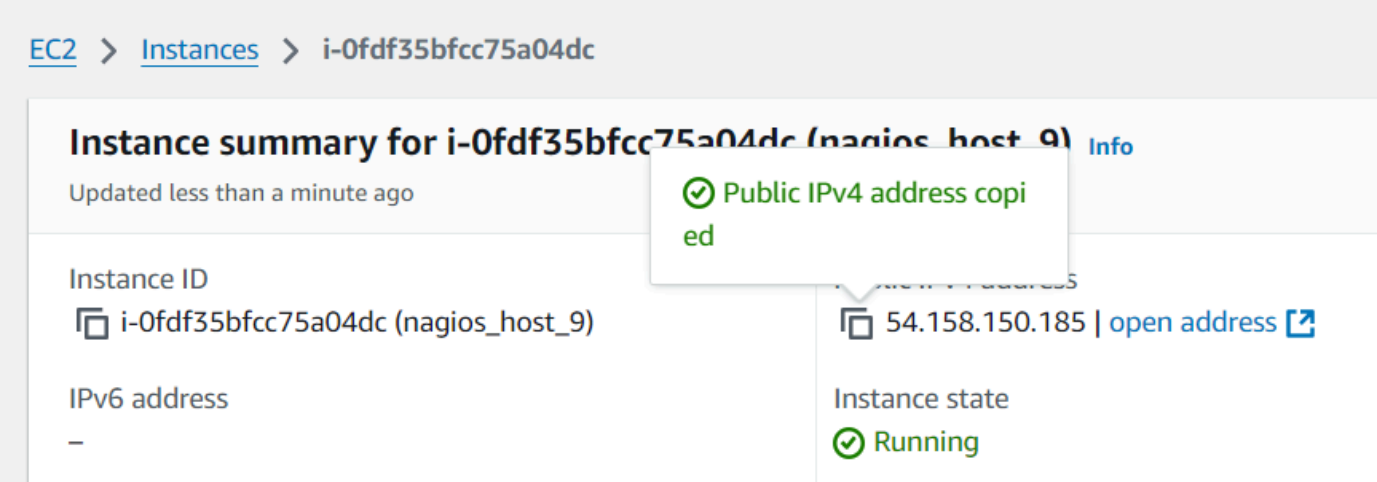
sudo service nagios start

```
[ec2-user@ip-172-31-47-54 nagios-plugins-2.4.11]$ sudo service nagios start
Redirecting to /bin/systemctl start nagios.service
[ec2-user@ip-172-31-47-54 nagios-plugins-2.4.11]$
```

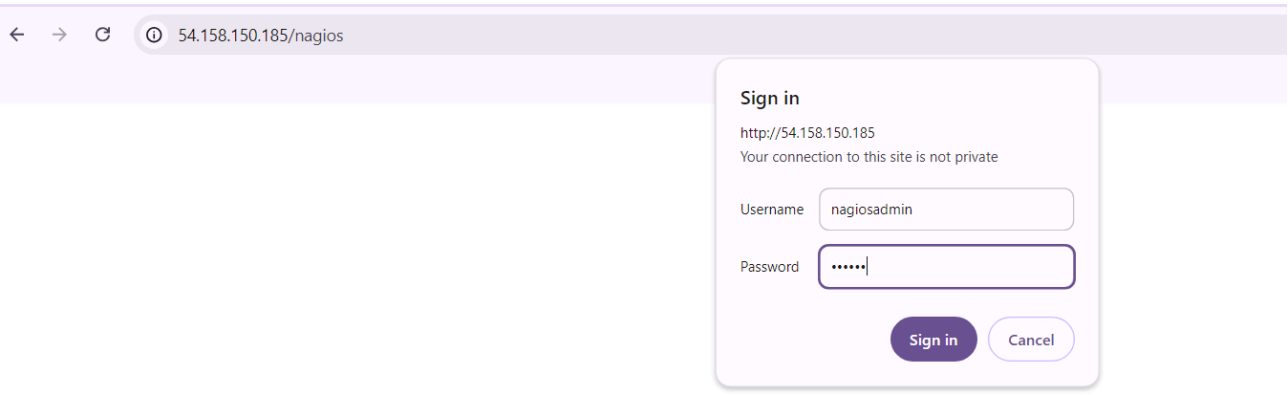
sudo systemctl status nagios

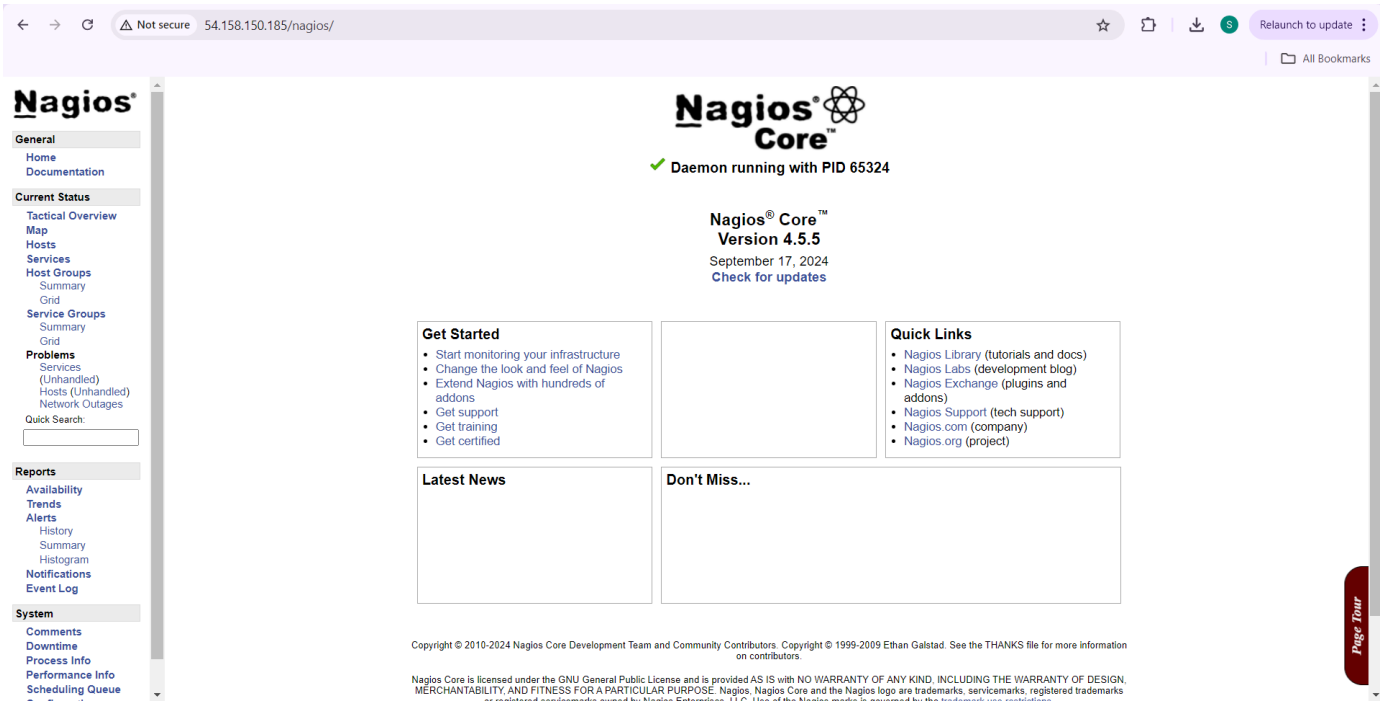
```
[ec2-user@ip-172-31-47-54 nagios-plugins-2.4.11]$ sudo systemctl status nagios
● nagios.service - Nagios Core 4.5.5
     Loaded: loaded (/usr/lib/systemd/system/nagios.service; enabled; preset: disabled)
     Active: active (running) since Sat 2024-10-05 14:36:11 UTC; 22s ago
       Docs: https://www.nagios.org/documentation
    Process: 65322 ExecStartPre=/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
    Process: 65323 ExecStart=/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
   Main PID: 65324 (nagios)
      Tasks: 6 (limit: 1112)
     Memory: 5.7M
        CPU: 80ms
     CGroup: /system.slice/nagios.service
             ├─65324 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─65325 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─65326 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─65327 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─65328 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             └─65329 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
```

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.



Lastly, go to your web browser and type “http:///nagios” Replace public-IPv4-address with the public ip address of your instance which you copied. You will get a prompt to enter the username and password that have been set for nagios admin





Conclusion:

In this experiment, we installed and configured Nagios Core, Nagios Plugins, and NRPE on a Linux system to enable continuous monitoring. Nagios is a vital tool in the DevOps ecosystem, providing real-time detection of network and server issues, which helps ensure the health of the infrastructure. Its scalability, robust security, and automated alerting system enhance the effectiveness of monitoring. By integrating NRPE, we extended Nagios' capabilities to monitor remote hosts, enabling proactive issue resolution. With its highly customizable architecture and extensive plugin support, Nagios proves to be indispensable for maintaining service uptime and operational stability.

Name: Shiven Bansal Roll No: 03 Class: D15C

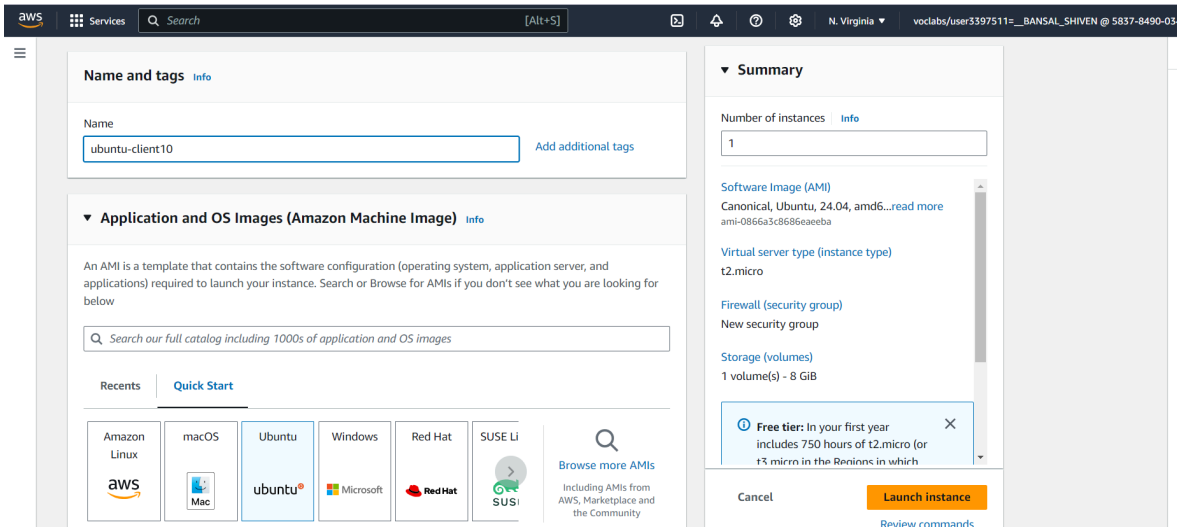
Experiment 10

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

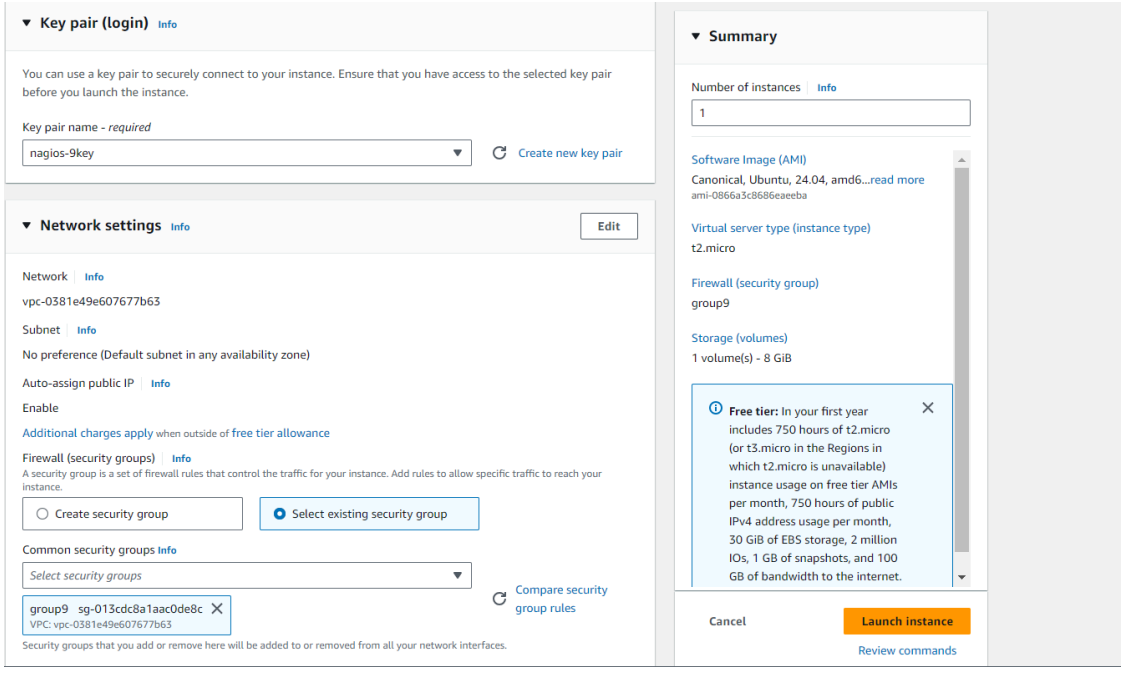
Prerequisites: An Amazon Linux instance with nagios (nagios-server) is already set up.

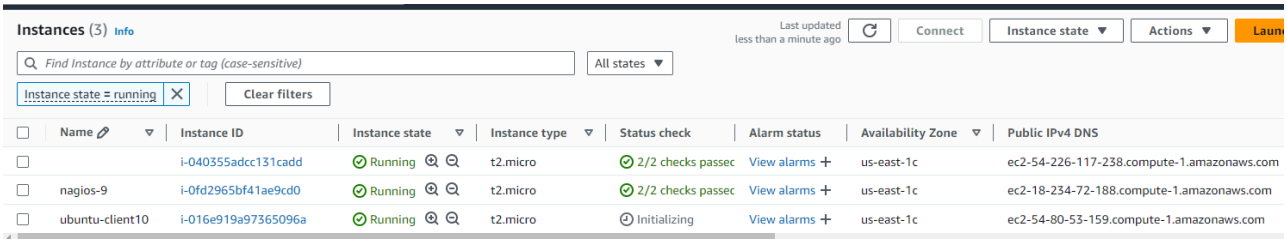
Steps:

Step 1: Navigate to EC2 on the AWS console using the ‘Services’ section and click on ‘Create instance’. Give your instance a name and choose ‘Ubuntu’ as the instance type.



Ensure that you choose the same key pair and security group for the Ubuntu client instance as you did for the Nagios host instance. Then, click on ‘Create instance’.





Your Ubuntu client instance gets created along with the Nagios host instance.

Step 2: Click on the instance ID of your nagios-server instance and click on ‘Connect’. Then, click on ‘SSH client’ and copy the command under ‘Example’. Then, open the terminal in the folder where the .pem file for your instance’s key pair is located and paste the SSH command that you just copied. This connects your instance to your local terminal using SSH.

Step 3: ps -ef | grep nagios
Run the above command on the nagios-host instance. This verifies whether the nagios service is running or not.

```
[ec2-user@ip-172-31-35-113 ~]$ ps -ef | grep nagios
nagios     64399       1  0 13:48 ?        00:00:00 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
nagios     64401   64399  0 13:48 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios     64402   64399  0 13:48 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios     64403   64399  0 13:48 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios     64404   64399  0 13:48 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios     64447   64399  0 13:48 ?        00:00:00 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
ec2-user   65271   65245  0 14:01 pts/0    00:00:00 grep --color=auto nagio
[ec2-user@ip-172-31-35-113 ~]$
```

Step 4: sudo su
mkdir -p /usr/local/nagios/etc/objects/monitorhosts
mkdir -p /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
This makes you the root user and creates two folders with the above paths.

```
[ec2-user@ip-172-31-35-113 ~]$ sudo su
mkdir -p /usr/local/nagios/etc/objects/monitorhosts
mkdir -p /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
[root@ip-172-31-35-113 ec2-user]#
```

Step 5: 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

```
[root@ip-172-31-88-33 ec2-user]# cp /usr/local/nagios/etc/objects/localhost.cfg /usr/local/nagios/etc/objects/monitorhos
ts/linuxhosts/linuxserver.cfg
```

Step 6: 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

1. Change hostname and alias from ‘hostname’ to ‘linuxserver’.
2. Change address to the public ip address of the ubuntu-client instance.

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



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

# Define a host for the local machine

define host {

    use                     linux-server            ; Name of host template to use
                                                    ; This host definition will inherit all variables that are defined
                                                    ; in (or inherited by) the linux-server host template definition.
    host_name               linuxserver
    alias                   linuxserver
    address                 54.80.53.159
}
```

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                 linuxserver             ; Comma separated list of hosts that belong to this group
}
```

Change all the subsequent occurrences of hostname in the file from ‘localhost’ to ‘linuxserver’.

Step 7: Open the Nagios config file using the following command:
nano /usr/local/nagios/etc/nagios.cfg
Then, add the following line to the config file:
cfg_dir=/usr/local/nagios/etc/objects/monitorhosts/

```
  GNU nano 5.8                                  /usr/local/nagios/etc/nagios.cfg

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

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

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

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


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

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


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

Step 8: Now we verify the configuration files and check that they contain no errors using the following command:
/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

```
[root@ip-172-31-35-113 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...
```

```
Checking objects...
        Checked 8 services.
        Checked 2 hosts.
        Checked 2 host groups.
        Checked 0 service groups.
        Checked 1 contacts.
        Checked 1 contact groups.
        Checked 24 commands.
        Checked 5 time periods.
        Checked 0 host escalations.
        Checked 0 service escalations.
Checking for circular paths...
        Checked 2 hosts
        Checked 0 service dependencies
        Checked 0 host dependencies
        Checked 5 timeperiods
Checking global event handlers...
Checking obsessive compulsive processor commands...
Checking misc settings...

Total Warnings: 0
Total Errors:   0

Things look okay - No serious problems were detected during the pre-flight check
```

Step 9: Once the files are verified and it is confirmed that there are no errors, we must restart the server.
service nagios restart

```
[root@ip-172-31-88-33 ec2-user]# service nagios restart
Redirecting to /bin/systemctl restart nagios.service
```

Step 10: systemctl status nagios

Using the above command, we check the status of the nagios server and ensure that it is active (running).

```
[root@ip-172-31-88-33 ec2-user]# systemctl status nagios
● nagios.service - Nagios Core 4.5.5
     Loaded: loaded (/usr/lib/systemd/system/nagios.service; enabled; preset: disabled)
     Active: active (running) since Sun 2024-09-29 12:11:40 UTC; 1min 12s ago
       Docs: https://www.nagios.org/documentation
    Process: 70244 ExecStartPre=/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg (code=exited, status=0>
    Process: 70245 ExecStart=/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SU>
   Main PID: 70246 (nagios)
      Tasks: 6 (limit: 1112)
     Memory: 4.0M
        CPU: 38ms
     CGroup: /system.slice/nagios.service
             ├─70246 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─70247 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─70248 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─70249 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─70250 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             └─70251 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg

Sep 29 12:11:40 ip-172-31-88-33.ec2.internal nagios[70246]: qh: Socket '/usr/local/nagios/var/rw/nagios.qh' successfull>
Sep 29 12:11:40 ip-172-31-88-33.ec2.internal nagios[70246]: qh: core query handler registered
Sep 29 12:11:40 ip-172-31-88-33.ec2.internal nagios[70246]: qh: echo service query handler registered
Sep 29 12:11:40 ip-172-31-88-33.ec2.internal nagios[70246]: qh: help for the query handler registered
Sep 29 12:11:40 ip-172-31-88-33.ec2.internal nagios[70246]: wproc: Successfully registered manager as @wproc with query>
Sep 29 12:11:40 ip-172-31-88-33.ec2.internal nagios[70246]: wproc: Registry request: name=Core Worker 70250;pid=70250
Sep 29 12:11:40 ip-172-31-88-33.ec2.internal nagios[70246]: wproc: Registry request: name=Core Worker 70249;pid=70249
Sep 29 12:11:40 ip-172-31-88-33.ec2.internal nagios[70246]: wproc: Registry request: name=Core Worker 70248;pid=70248
Sep 29 12:11:40 ip-172-31-88-33.ec2.internal nagios[70246]: wproc: Registry request: name=Core Worker 70247;pid=70247
Sep 29 12:11:40 ip-172-31-88-33.ec2.internal nagios[70246]: Successfully launched command file worker with pid 70251
```

Step 11: Connect your ubuntu-client instance to your local terminal using SSH in the same way as you connected the nagios-host instance to your local terminal using SSH (follow Step 2)

```
PS C:\Users\ADMIN> cd .\Downloads\
PS C:\Users\ADMIN\Downloads> ssh -i "nagios-9key.pem" ubuntu@ec2-54-80-53-159.compute-1.amazonaws.com
The authenticity of host 'ec2-54-80-53-159.compute-1.amazonaws.com (54.80.53.159)' can't be established.
ED25519 key fingerprint is SHA256:ictbLzP1p2YFXqDXkAH4CzoAWhCOQfhqnoXYFJGZ5q8.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-54-80-53-159.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-1016-aws x86_64)

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

 System information as of Mon Oct  7 14:15:43 UTC 2024

  System load:  0.0               Processes:             104
  Usage of /:   22.8% of 6.71GB   Users logged in:       0
  Memory usage: 20%               IPv4 address for enX0: 172.31.38.70
  Swap usage:   0%
```

```
Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

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


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


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

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

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

ubuntu@ip-172-31-38-70:~$
```

Step 12: On your ubuntu-client instance, run the following commands:-
sudo apt update -y
sudo apt install gcc -y
sudo apt install -y nagios-nrpe-server nagios-plugins
The above commands check for any new updates and then install gcc, Nagios NRPE server and Nagios plugins.

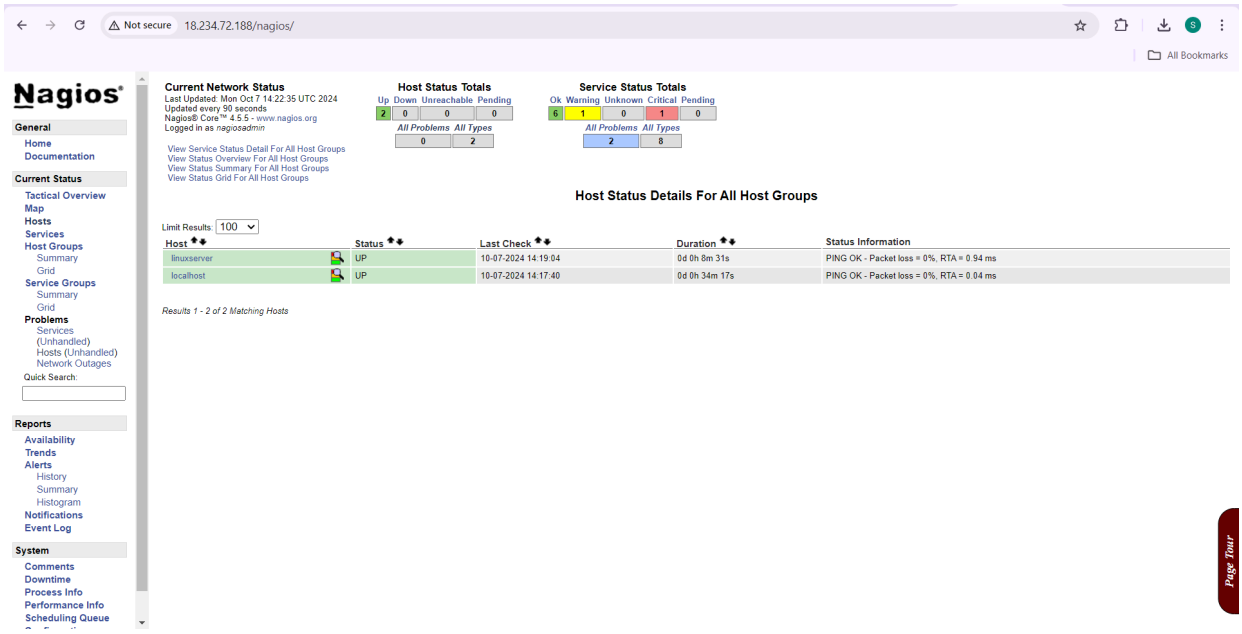
```
ubuntu@ip-172-31-38-70:~$ sudo apt update -y
sudo apt install gcc -y
sudo apt install -y nagios-nrpe-server nagios-plugins
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:7 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [382 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Components [35.0 kB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 c-n-f Metadata [8328 B]
```

```
Setting up samba-dsdb-modules:amd64 (2:4.19.5+dfsg-4ubuntu9) ...
Setting up libsmbclient0:amd64 (2:4.19.5+dfsg-4ubuntu9) ...
Setting up libcups2t64:amd64 (2.4.7-1.2ubuntu7.3) ...
Setting up python3-samba (2:4.19.5+dfsg-4ubuntu9) ...
Setting up smbclient (2:4.19.5+dfsg-4ubuntu9) ...
Setting up samba-common-bin (2:4.19.5+dfsg-4ubuntu9) ...
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for libc-bin (2.39-0ubuntu8.3) ...
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-172-31-38-70:~$
```

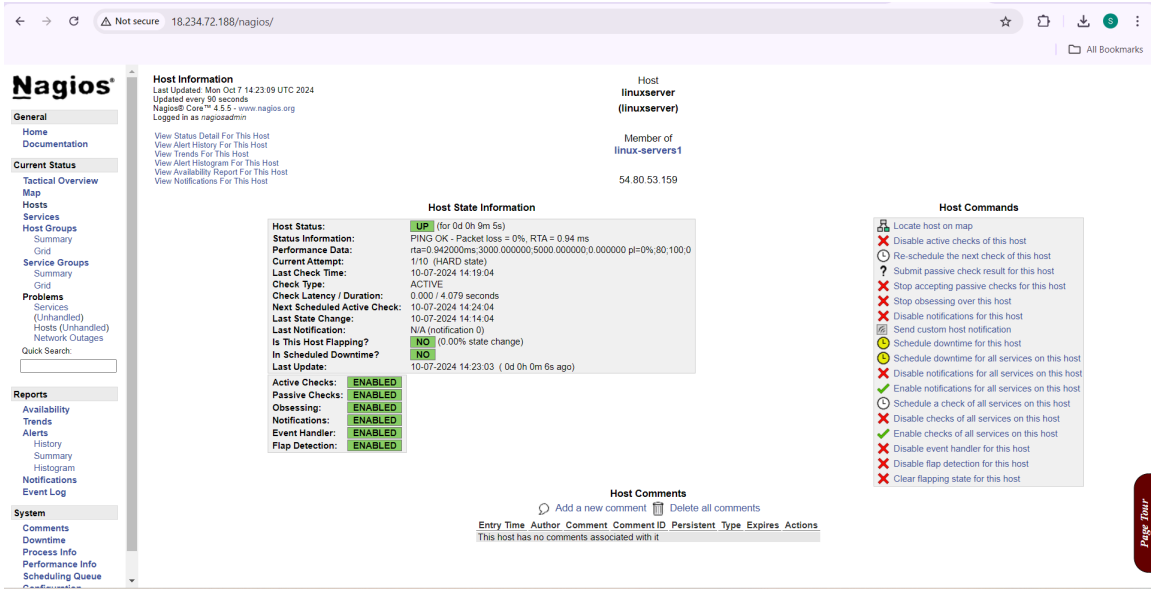
Step 13: Run the following command:
sudo nano /etc/nagios/nrpe.cfg
The above command opens the NRPE config file. Here, we need to add the public IP address of our host nagios-host instance to the NRPE configuration file.
Under allowed_hosts, add the nagios-host public IPv4 address.

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

nrpe_group=nagios



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

allowed_hosts=127.0.0.1,18.234.72.188



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

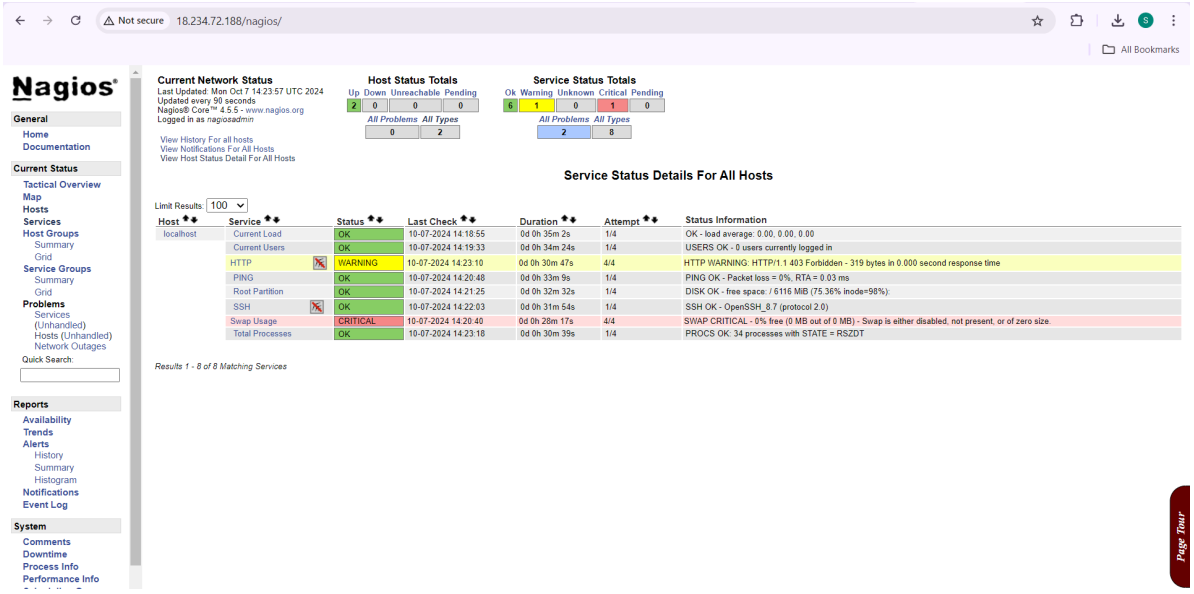
Step 14: Navigate to the Nagios dashboard. Click on ‘hosts’. We see that linuxserver has been added as a host.



Click on ‘linuxserver’. Here, we can access all information about the ‘linuxserver’ host.



Click on ‘Services’. Here, we can see all the services that are being monitored by ‘linuxserver’.



Conclusion : In this experiment, we explored how to monitor ports, services, and both Windows and Linux servers using Nagios. To achieve this, we launched a Nagios-hosted EC2 Linux instance, which served as the platform for running the Nagios server and dashboard. Additionally, we deployed an Ubuntu client instance that connected to the Nagios host.

We configured the necessary settings on the Linux instance, including adding the Ubuntu client’s public IP address. Similarly, we made configuration changes on the Ubuntu client, where we added the IP address of the Nagios-hosted Linux instance. We also ensured that the Linux server instance was permitted as an authorized host on the Ubuntu client. After restarting the NRPE service, we verified that the ‘linuxserver’ host was successfully added for monitoring.

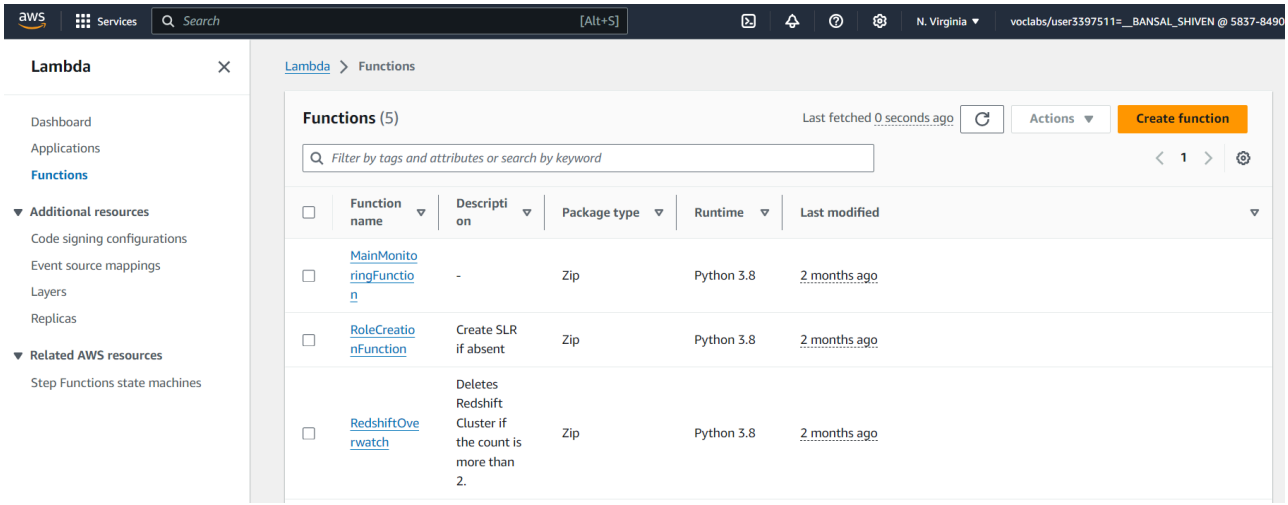
Name: Shiven Bansal Roll No.: 03 Class: D15C

Experiment 11

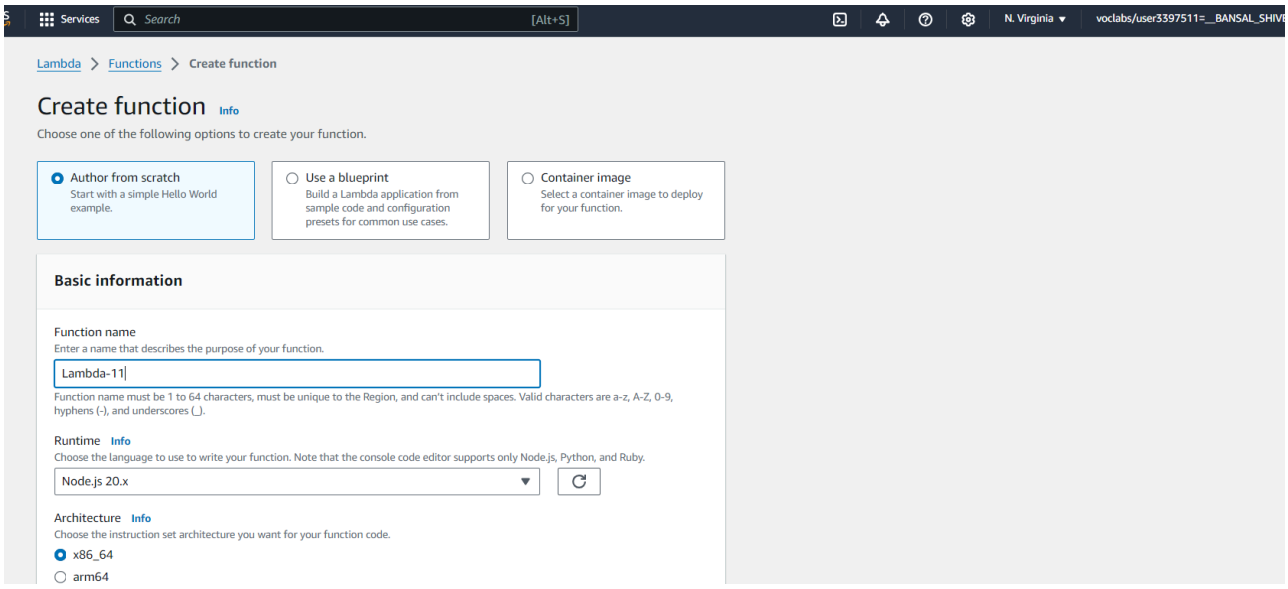
Aim: To understand AWS Lambda, its workflow, various functions and create your first Lambda functions using Python / Java / Nodejs.

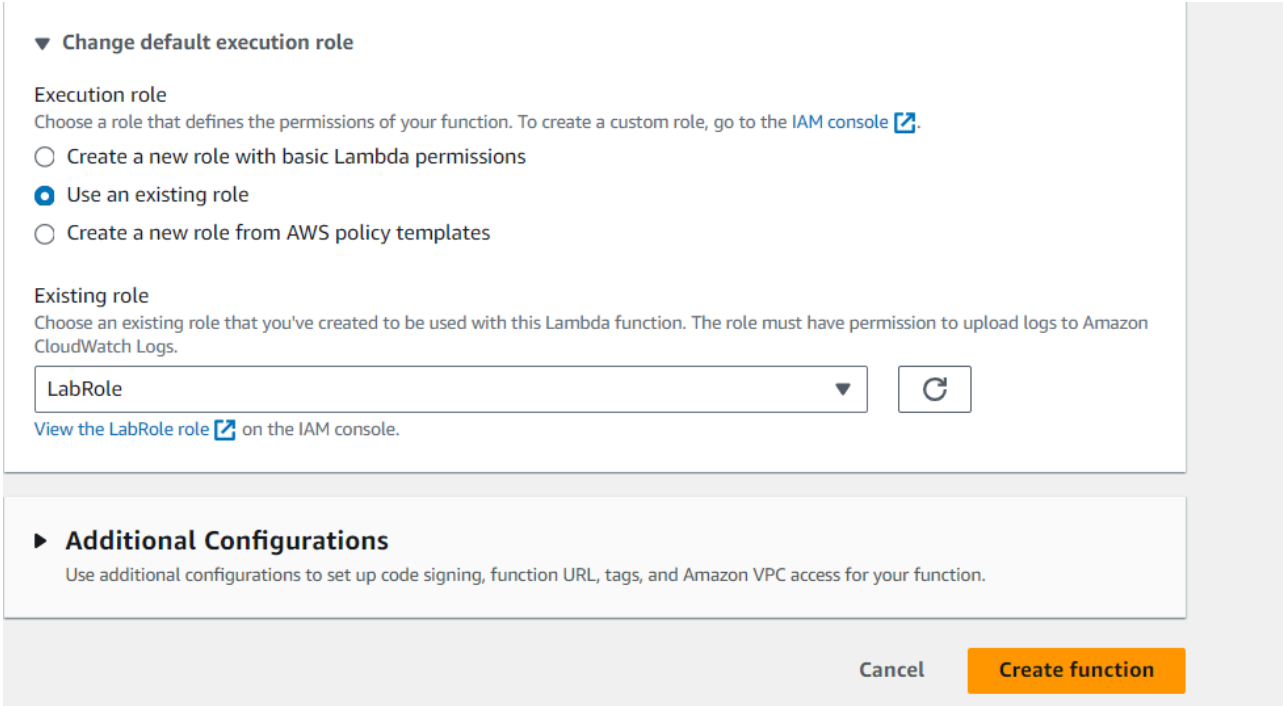
Steps:

Step 1: On your AWS console, click on ‘Lambda’ in the services section and click on ‘Create function’.



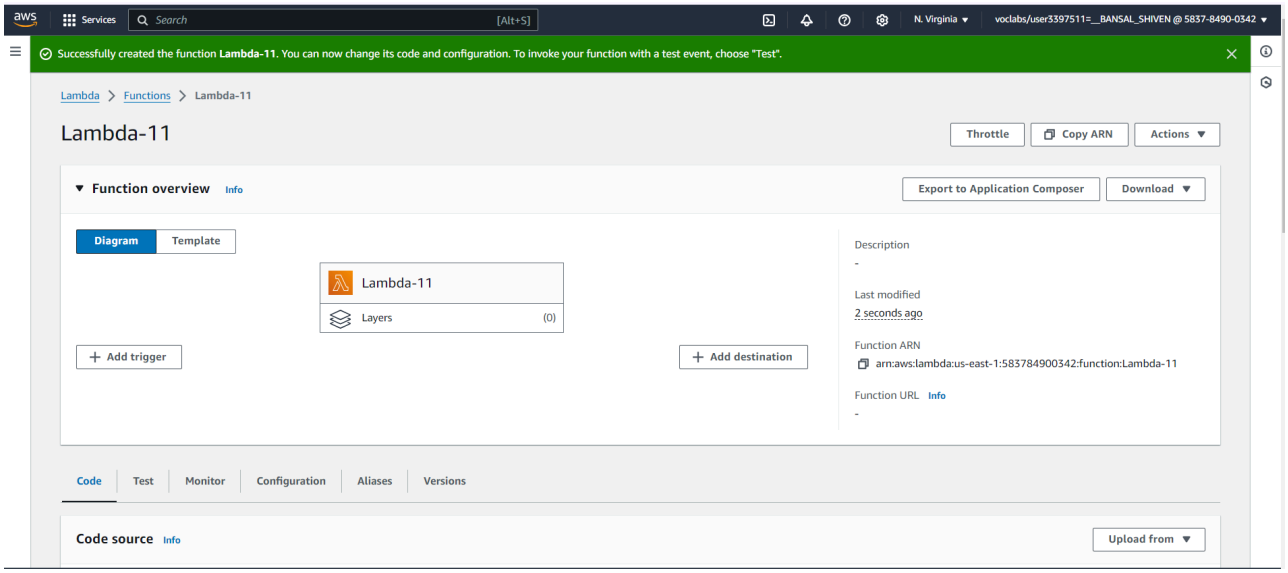
Step 2: Give your Lambda function a name. Select the language to use to write your function (Node.js is the default and what we will use in this experiment). Keep other options as default.

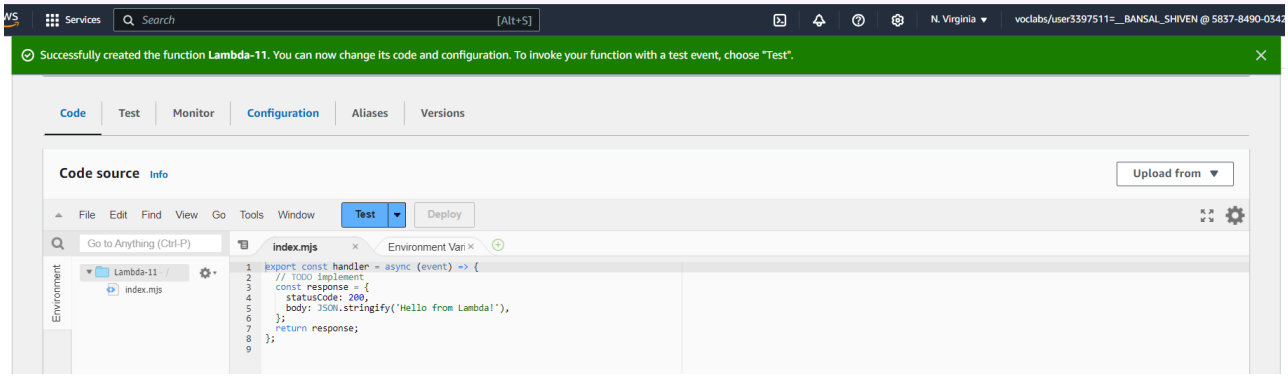




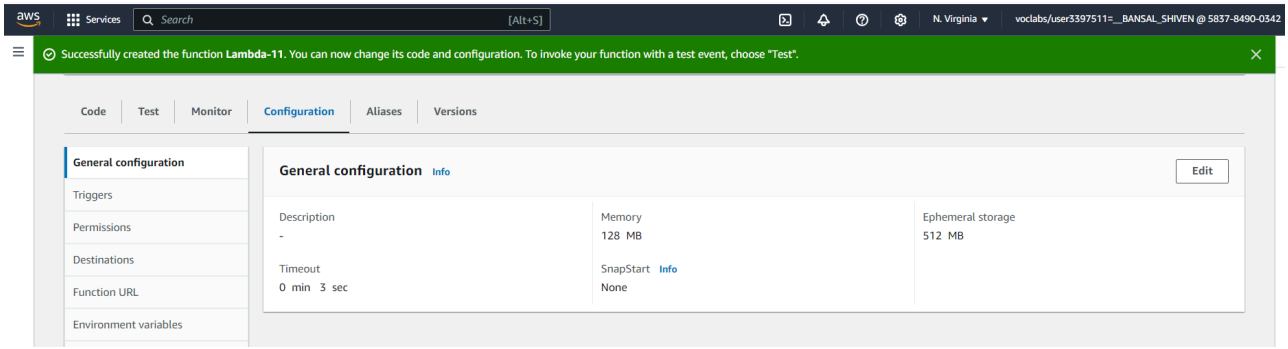
Under ‘Execution role’, choose ‘Use an existing role’ and then choose LabRole. Then, click on ‘Create function’.

Your Lambda function gets created.

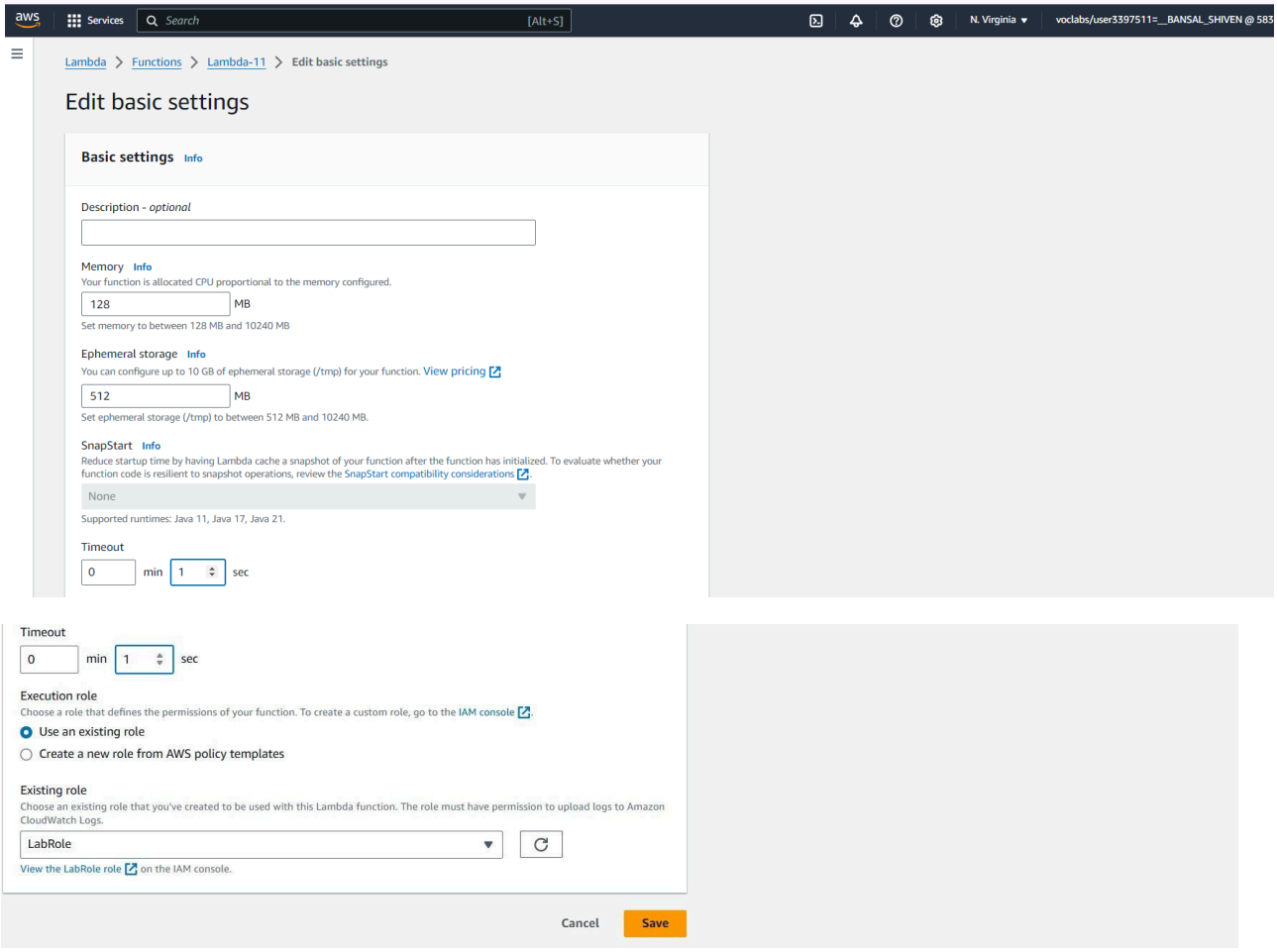




Step 3: The general configuration of the function is visible in the ‘Configuration’ tab. To change the configuration, click on ‘Edit’.

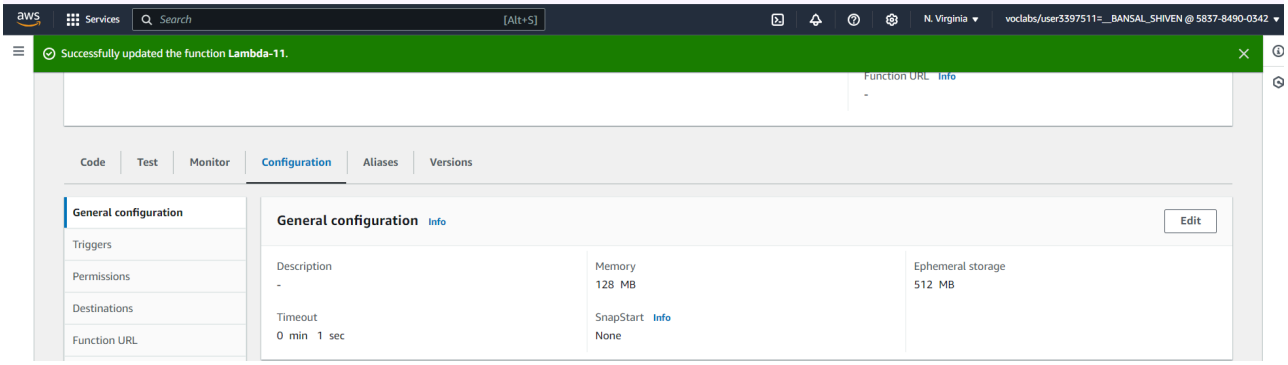


You can change the various parameters of the configuration as per your needs. Here, we can change the ‘Timeout’ period to 1 second as it’s sufficient for our function for now. ‘Timeout’ is the time for which a function can be running before it gets forcibly terminated.

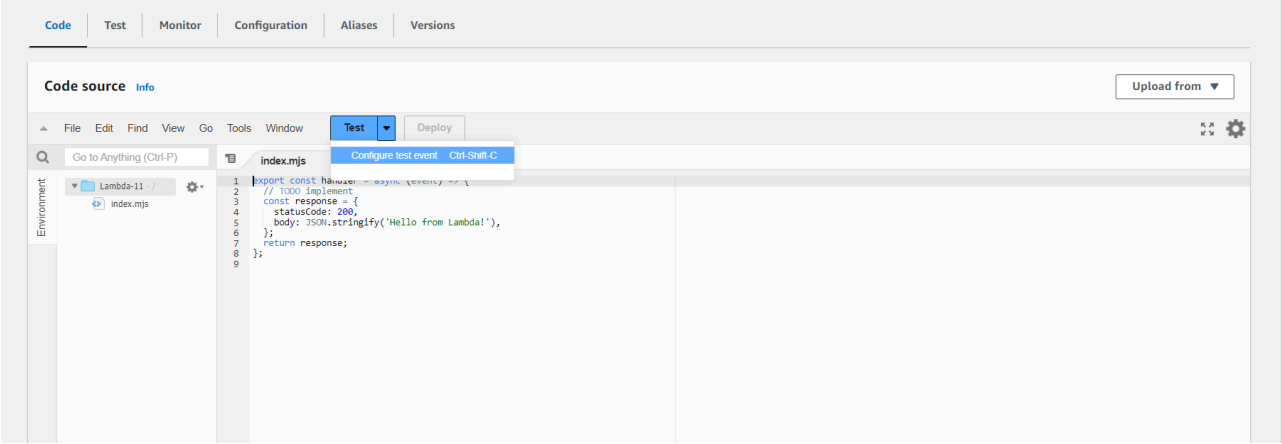


After making the required changes, click on ‘Save’.

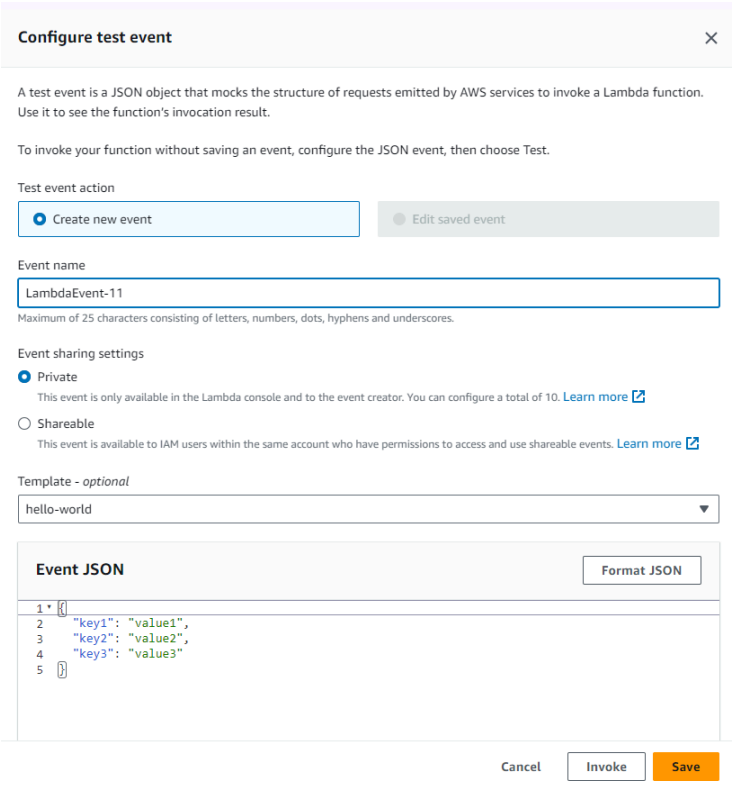
The changes in the general configuration are visible in the function.



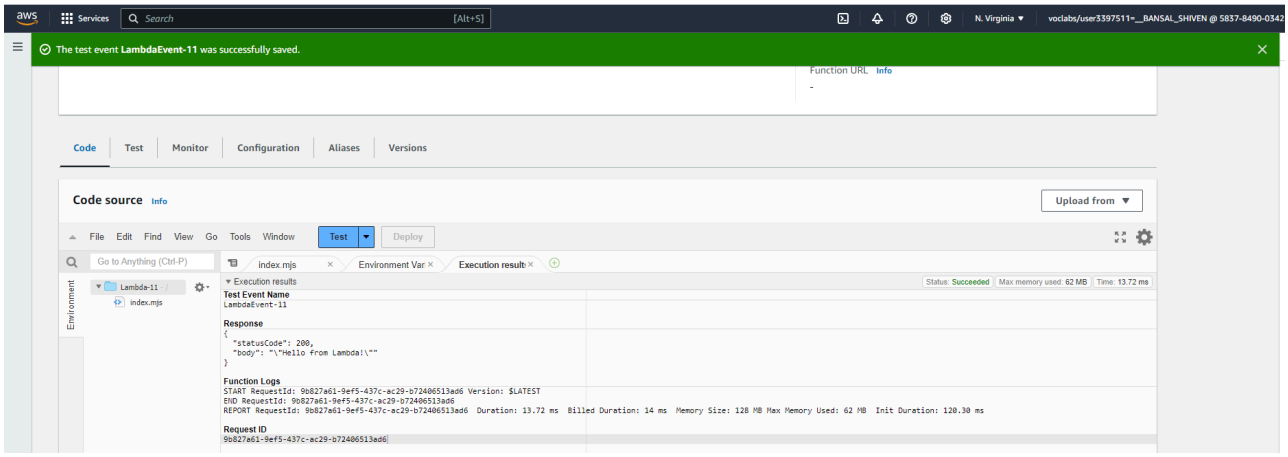
Step 4: In the ‘Code source’ section, click on the arrow next to the ‘Test’ button and click on ‘Configure test event’.



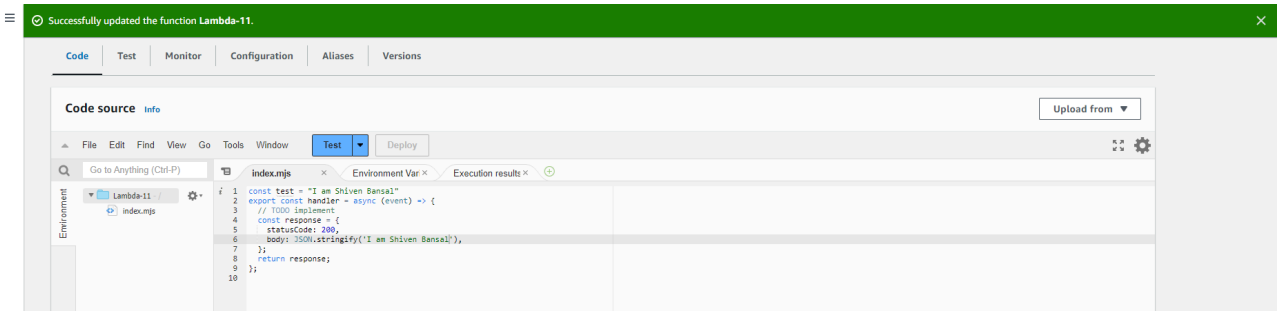
Step 5: Give your test event a name, keep all other options as default and click on ‘Save’.



Step 6: Click on the ‘Test’ button. The following output appears.

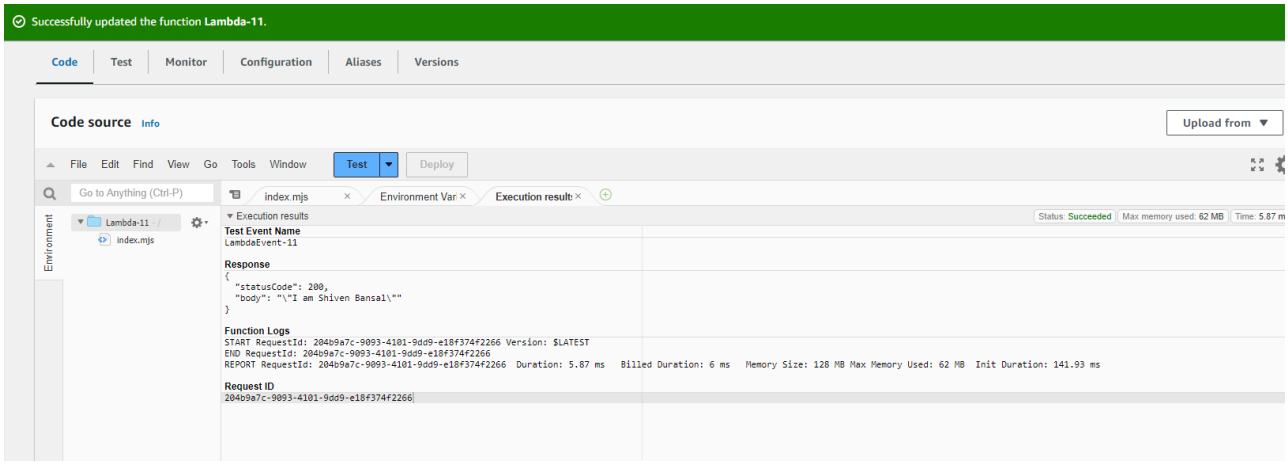


Step 7: You can make changes in the code to observe the difference in the output. Here, we change the code to display a different string as such:-



Once the changes are made, click on ‘Deploy’.

Step 8: Click on ‘Test’ and observe how the output after the changes differs from the output before the changes.



Conclusion:

In this experiment, I explored AWS Lambda and created my first Lambda function using Node.js. I learned how to configure a Lambda function, including selecting execution roles and adjusting parameters like the timeout period. I also tested the function by creating and executing a test event. After making changes to the code, I deployed the new version and observed how the output changed. This experiment gave me hands-on experience with Lambda's workflow, enhancing my understanding of serverless computing and how to deploy and test functions on AWS.

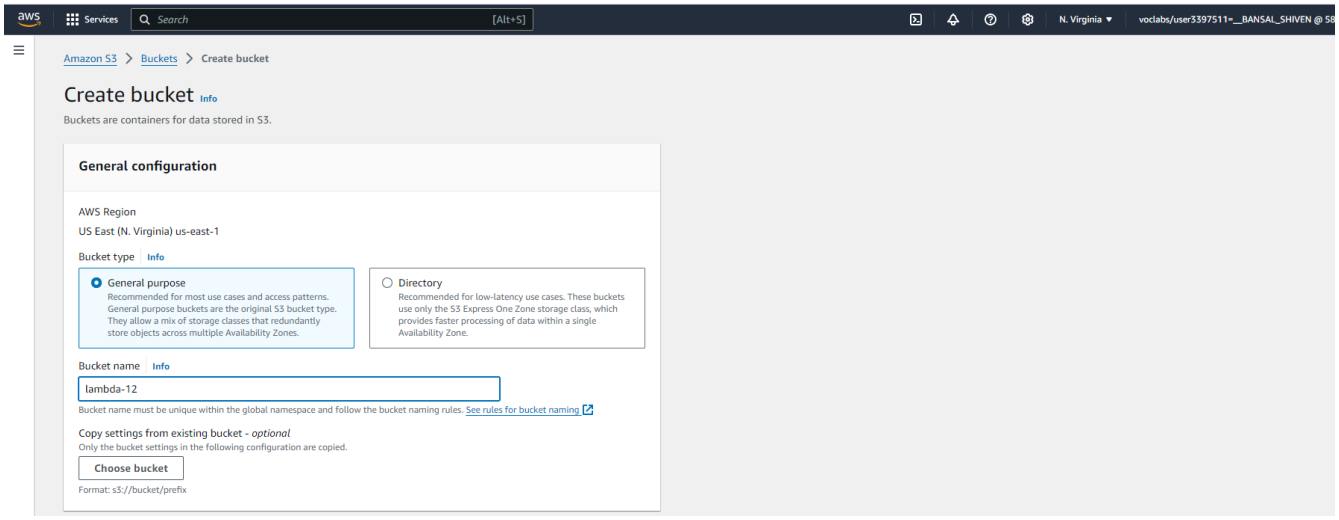
Name: Shiven Bansal Roll No: 03 Class: D15C

Experiment 12

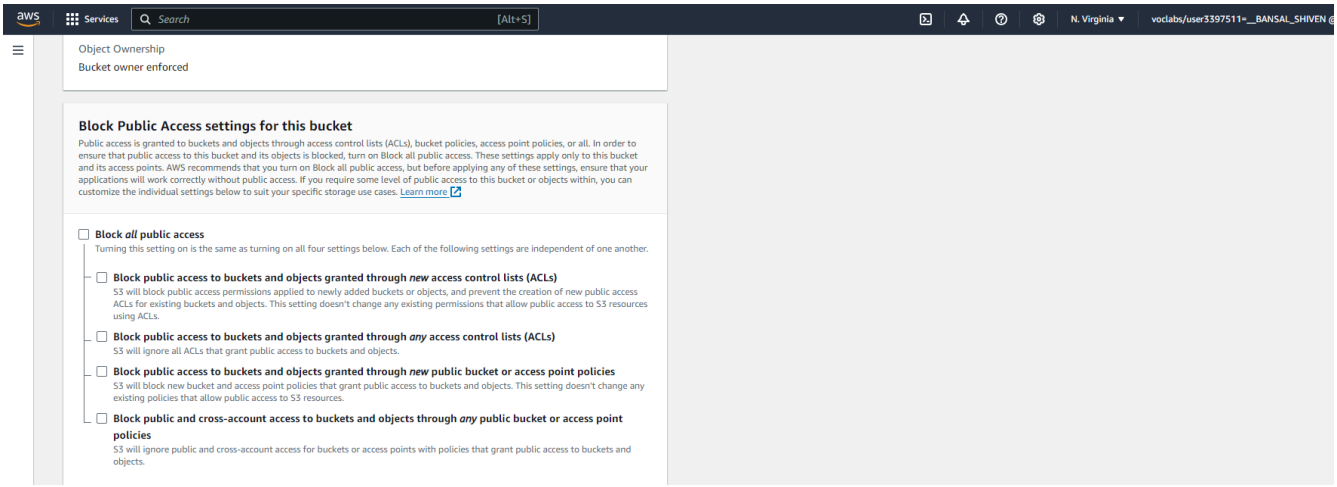
Aim: To create a Lambda function which will log “An Image has been added” once you add an object to a specific bucket in S3.

Steps:

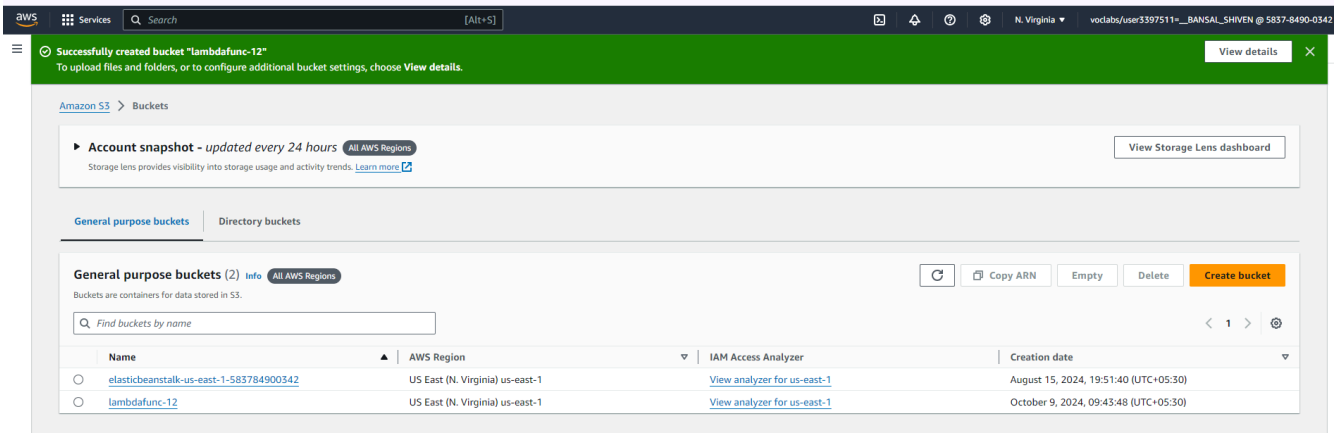
Step 1: On your AWS console, click on ‘S3’ in the services section and click on ‘Create bucket’. Give your bucket a name.



Uncheck the ‘Block all public access’ box.

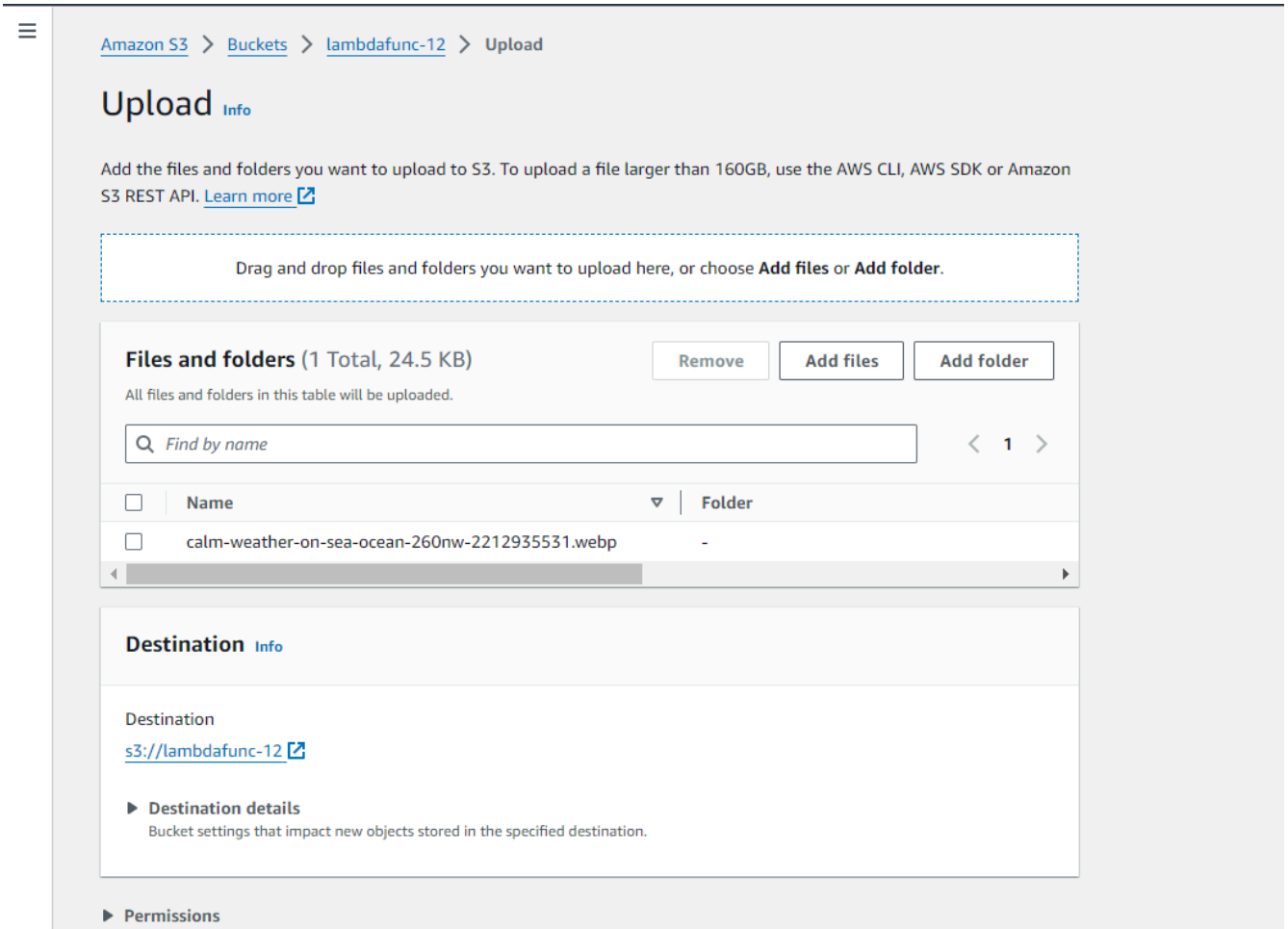


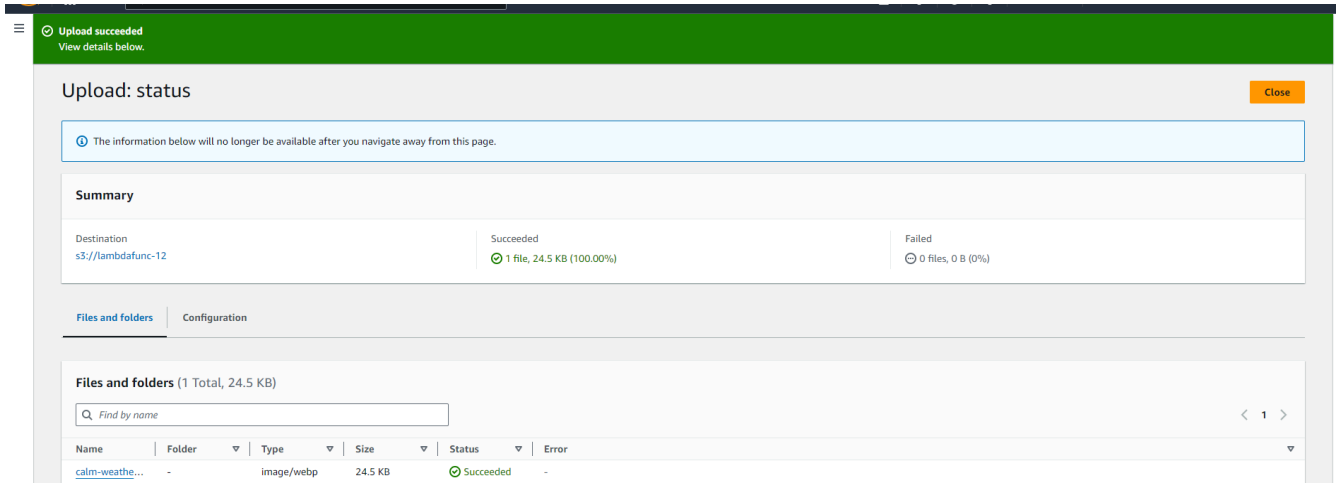
Keep all other options as default and click on ‘Create bucket’.



Your bucket is created.

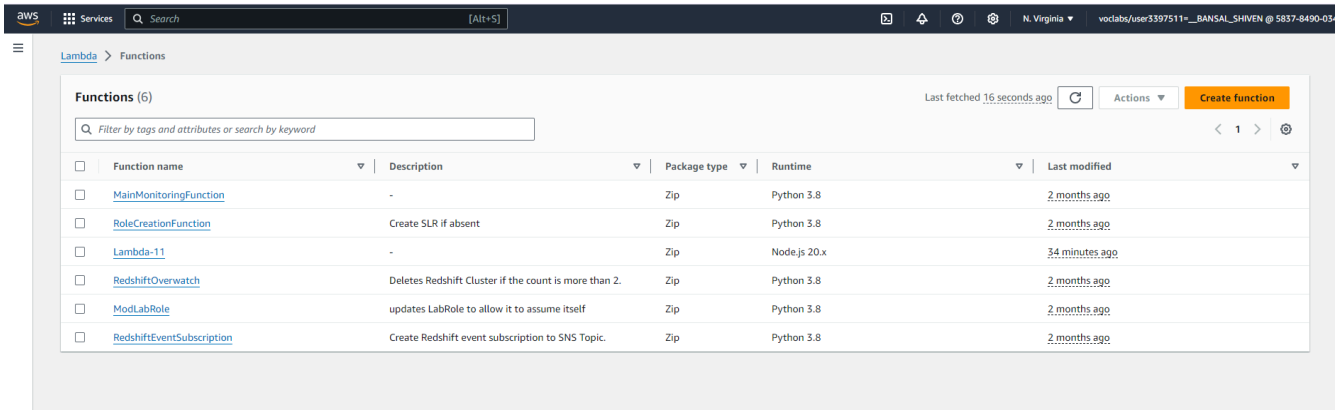
Step 2: Upload an image onto your S3 bucket by clicking on your S3 bucket, clicking on ‘Upload’, clicking on ‘Add files’, navigating to your image and selecting it.



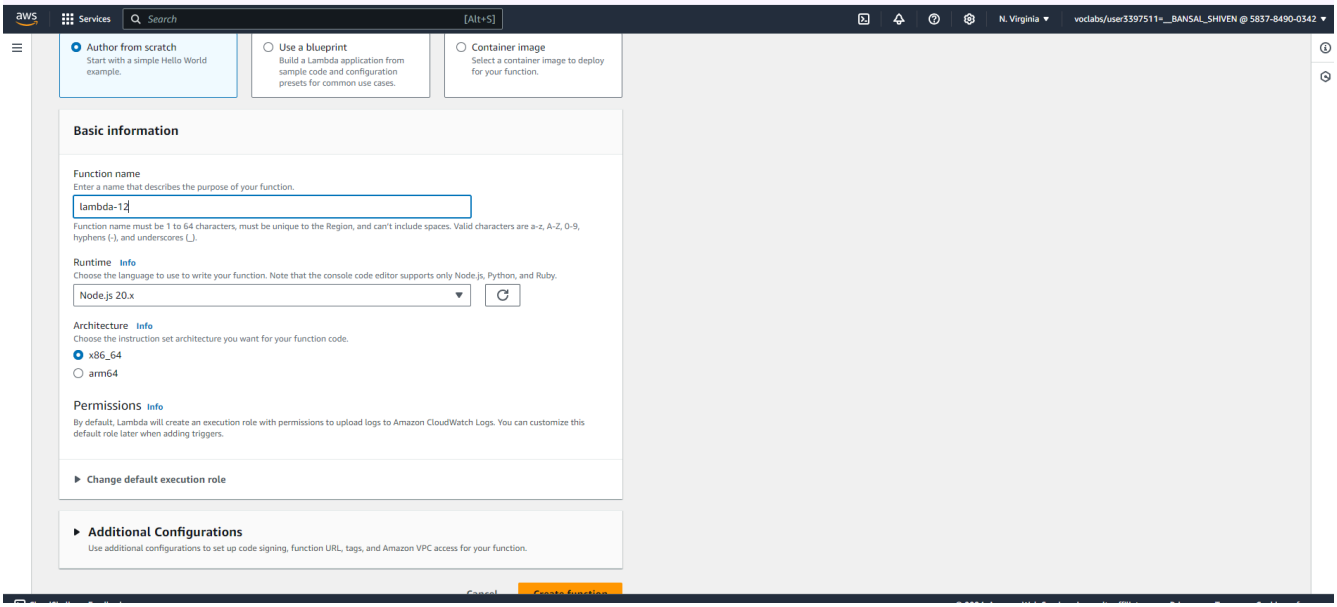


Your image gets uploaded onto the S3 bucket.

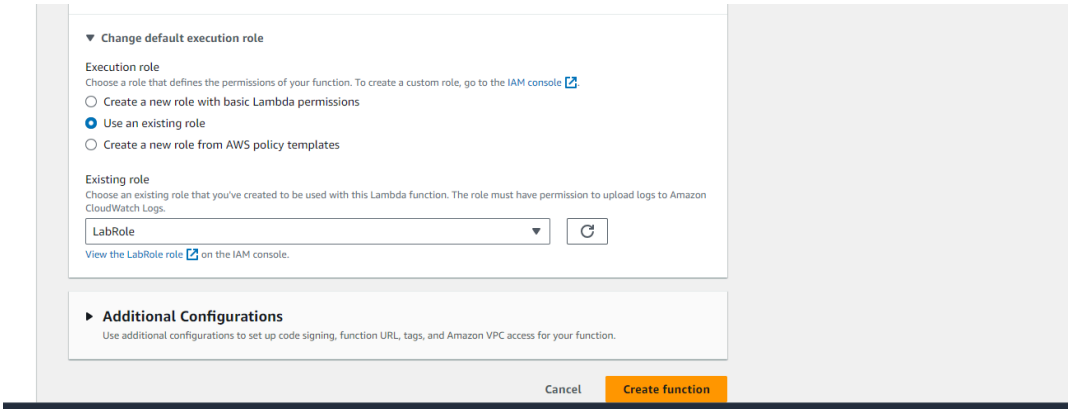
Step 3: Navigate to the AWS Lambda console using the ‘Services’ section. Click on ‘Create function’.



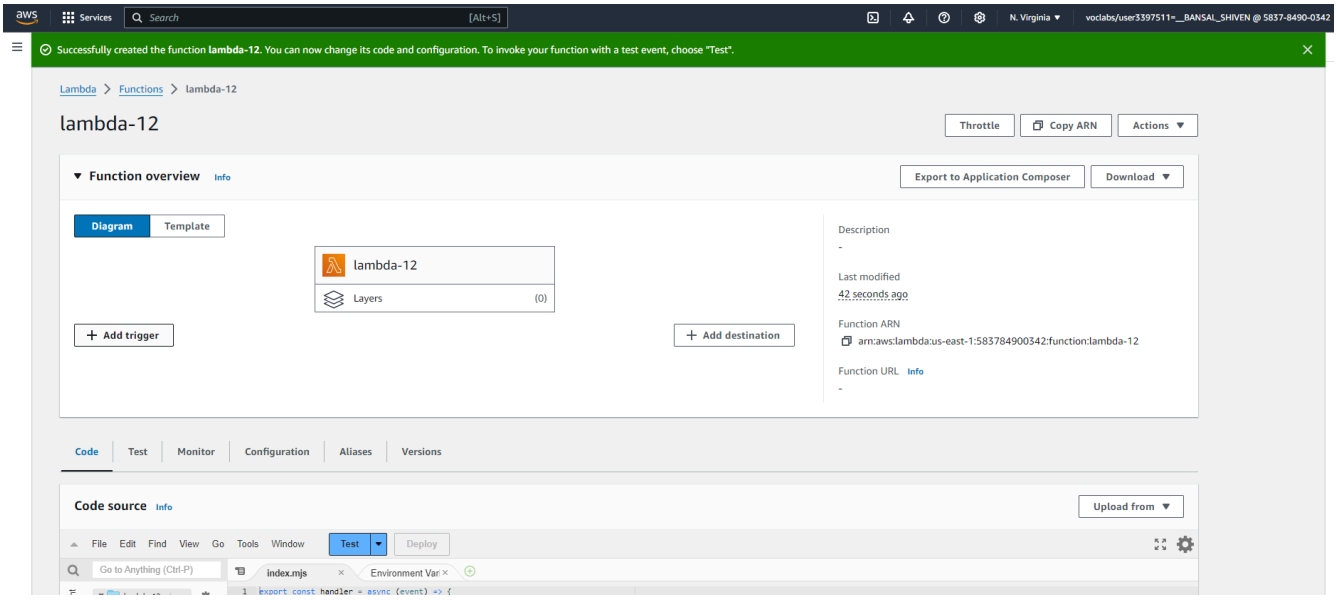
Step 4: Give your function a name and keep other settings as default.



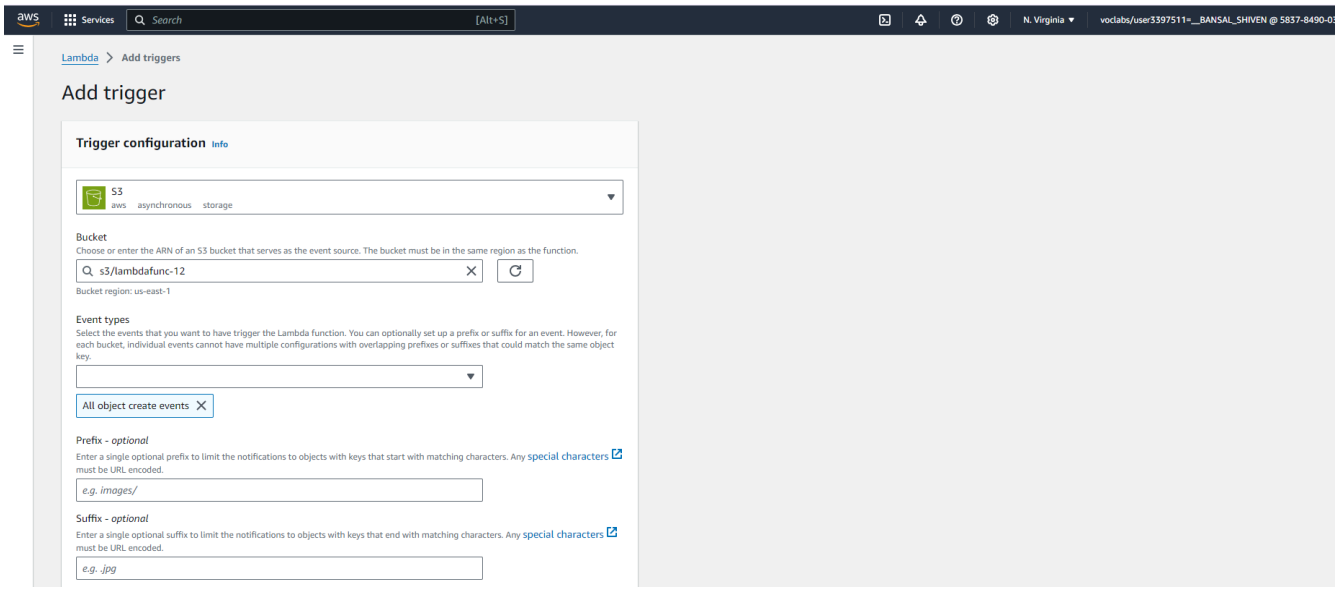
Under ‘Execution role’, choose ‘Use an existing role’ and in the dropdown box below, choose ‘LabRole’. Then, click on ‘Create function’. Your function gets created.

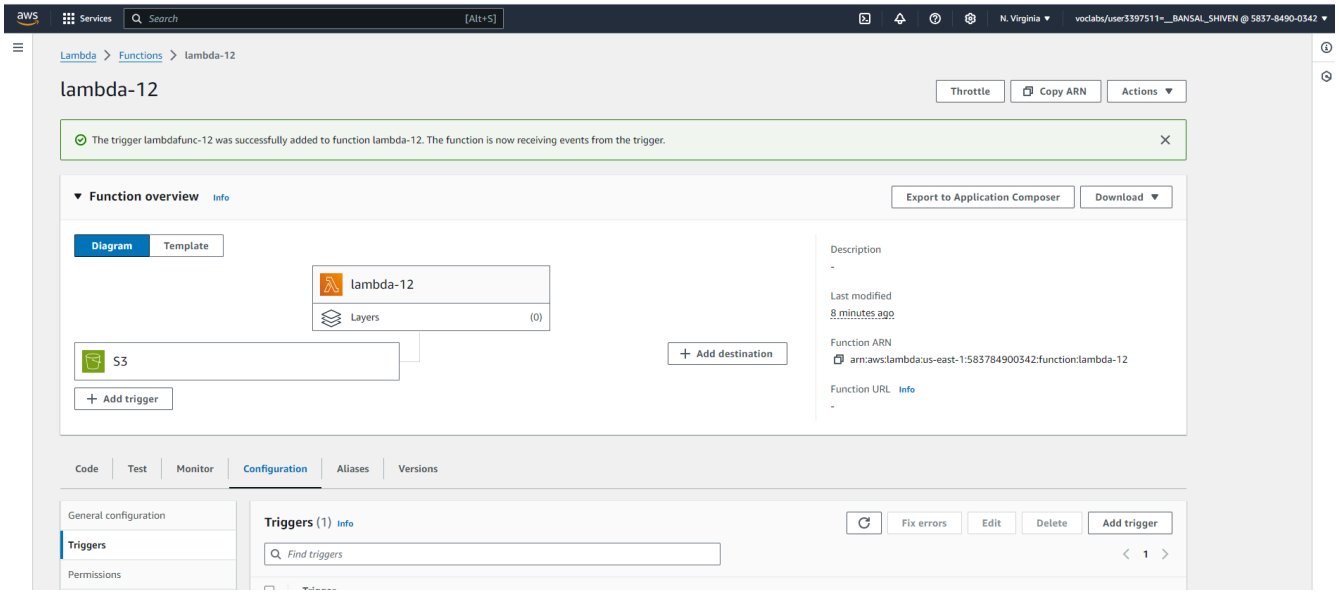


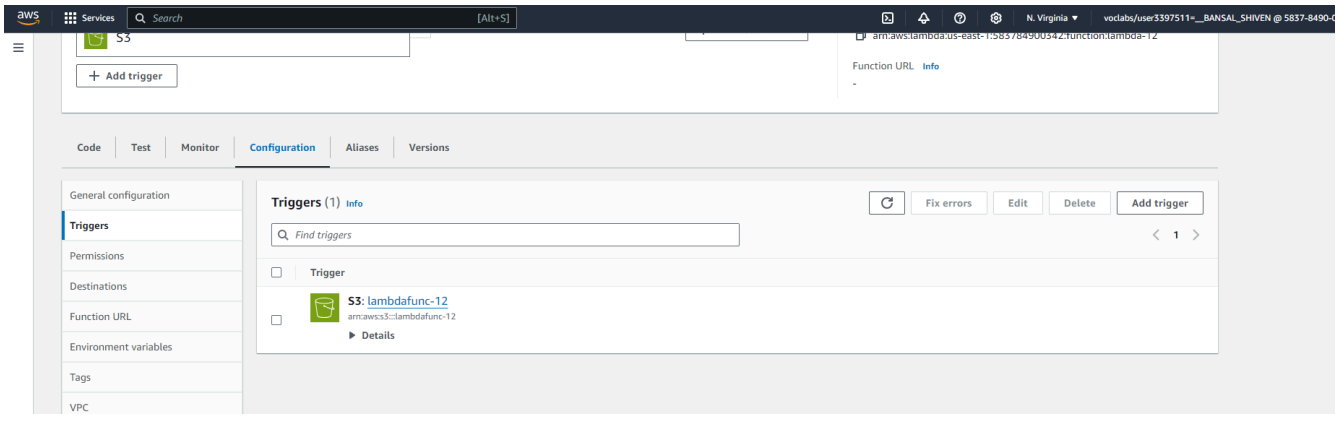
Step 5: On the page of the function you created, click on ‘Add trigger’.



Step 6: Choose ‘Trigger configuration’ as S3 and select the name of your bucket in the dropdown box below it. Keep other options as default and click on ‘Add’.



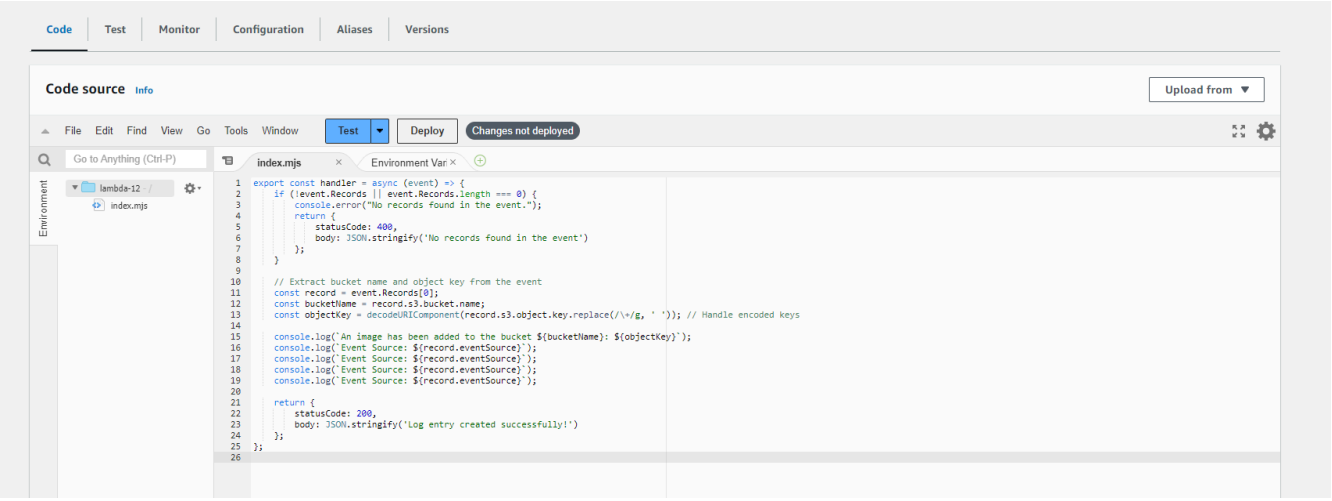




The trigger gets successfully added to your function.

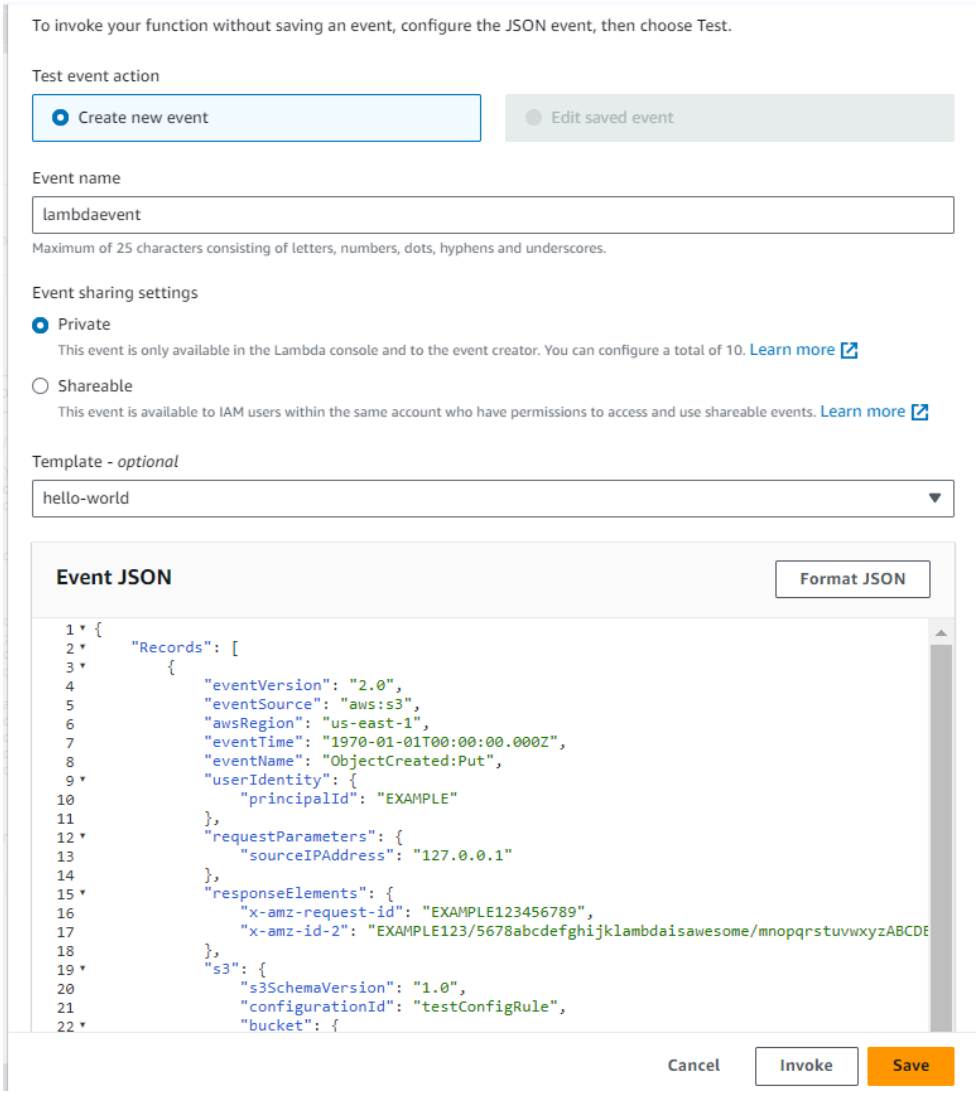
Step 7: In the ‘Code source’ section of your function, paste the following javascript code instead of the existing code:-

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

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

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

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



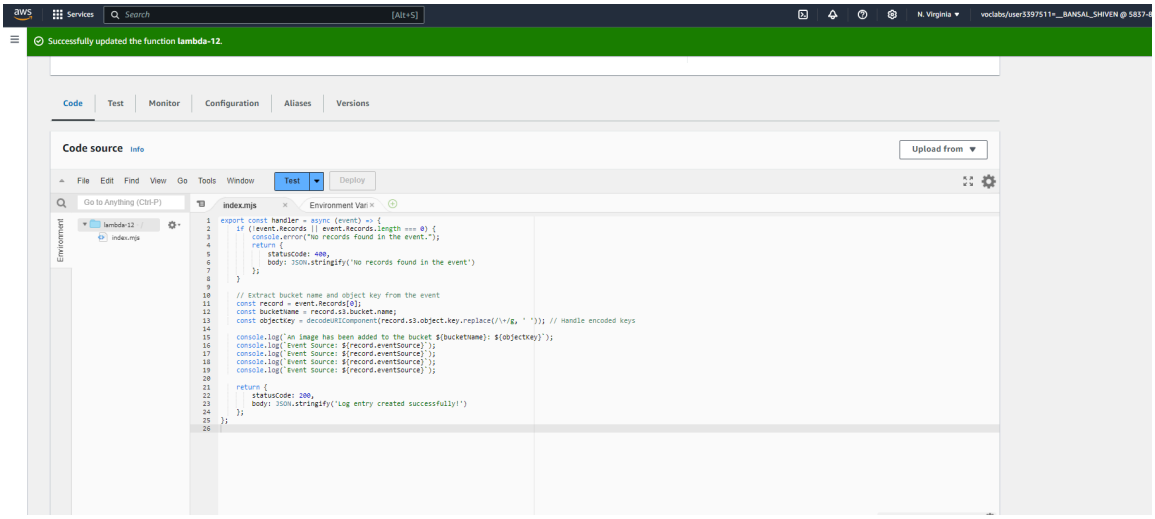
Step 8: Click on the arrow next to the ‘Test’ button and click on ‘Configure test event’. In the popup box that appears, if you have an existing event, enter the name of your event or create a new event and in the ‘Event JSON’ section, paste the following code:-

```
{
  "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"
        }
      }
    }
  ]
}
```

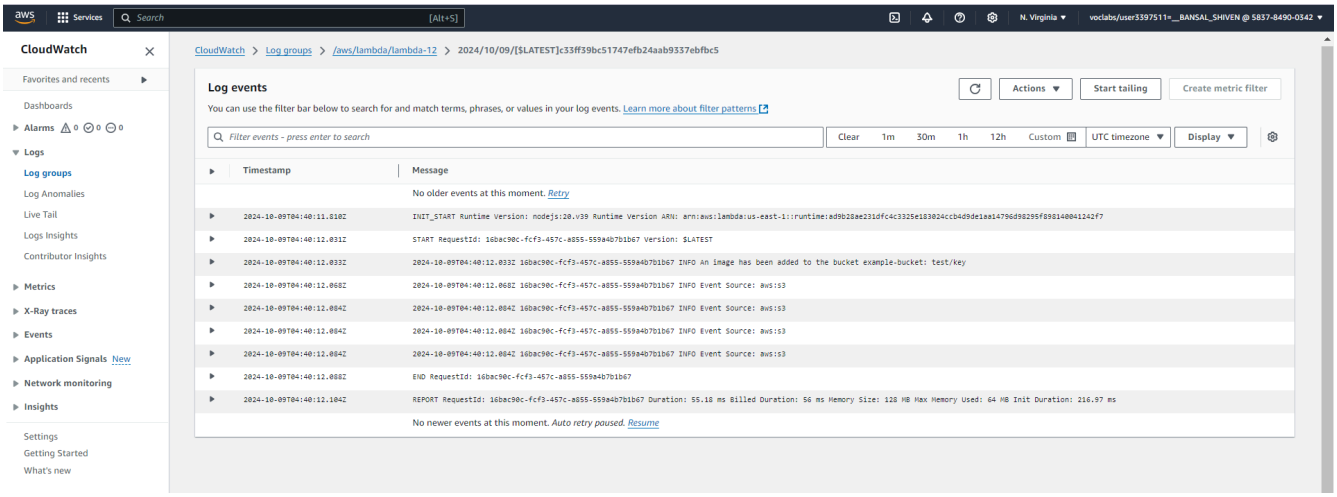


Then, click on ‘Save’. Your function gets successfully updated.

Step 9: Click on ‘Deploy’ and after deployment is successful, click on ‘Test’.



Running the test gives the above output which displays that ‘An Image has been added to the bucket’ and that the log entry was successfully created.



Conclusion:

In this experiment, I successfully created an AWS Lambda function that logs "An Image has been added" when an object is uploaded to a specific S3 bucket. I learned how to set up an S3 bucket, configure a Lambda function, and trigger it with S3 events. The function was tested with a simulated event, and it generated the expected log entry, confirming that the function worked as intended. This experiment helped me understand the integration between AWS Lambda and S3 and how to handle real-time event-based processing in AWS.