Name:Shejal Nilesh Tiwari Div/Roll No: D15C/57

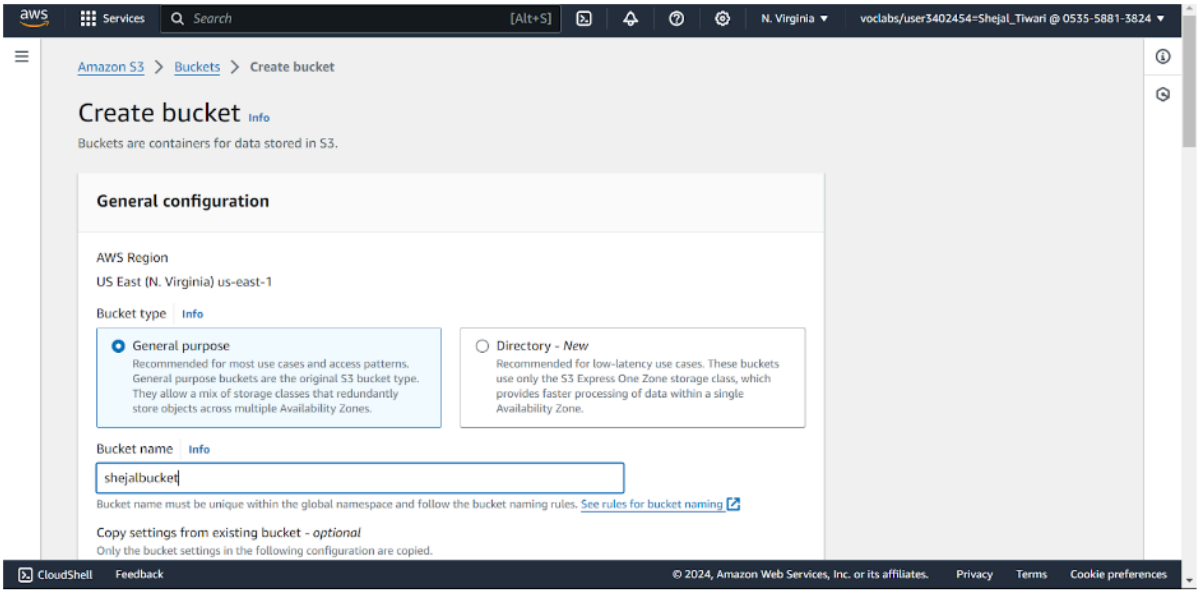
EXPERIMENT 1A

Step 1: Log in to AWS Management Console
Go to AWS Management Console and log in with your AWS account credentials.

Step 2: Navigate to S3 Service
In the AWS Management Console, search for "S3" in the search bar at the top, and click on "S3" under Services.

Step 3: Create a New Bucket
Click on the "Create bucket" button.
Fill in the details:

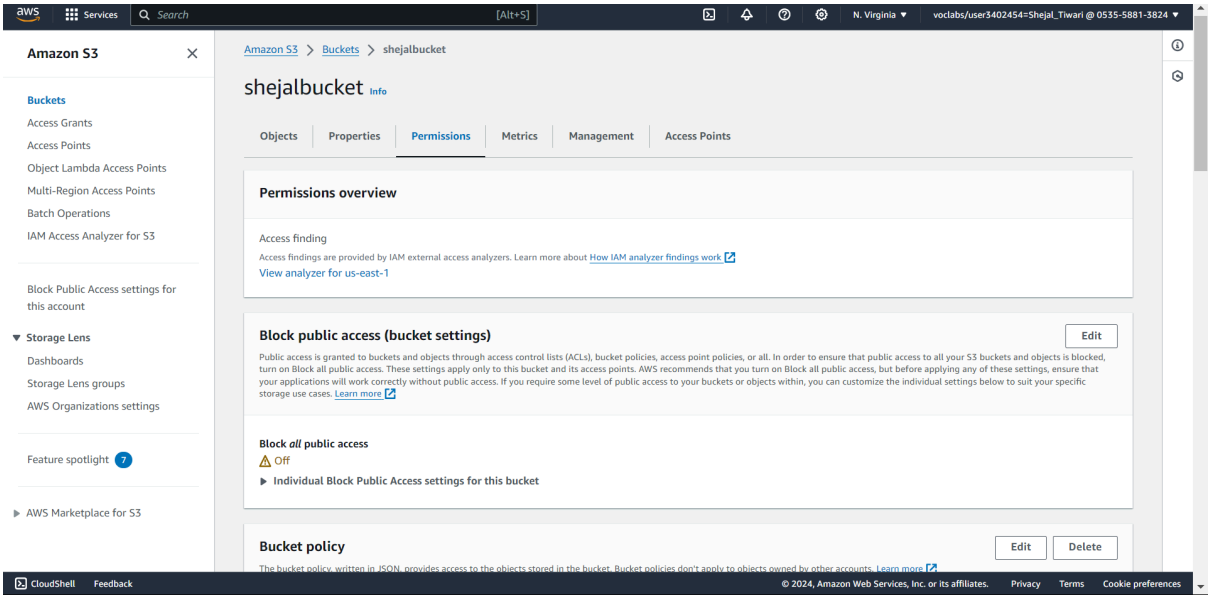
- Bucket Name: Enter a globally unique name for your bucket. The name must be DNS-compliant.
- AWS Region: Select the region where you want to create the bucket. Choose a region closest to you or your users for better performance.



Step 4: Configure Bucket Options

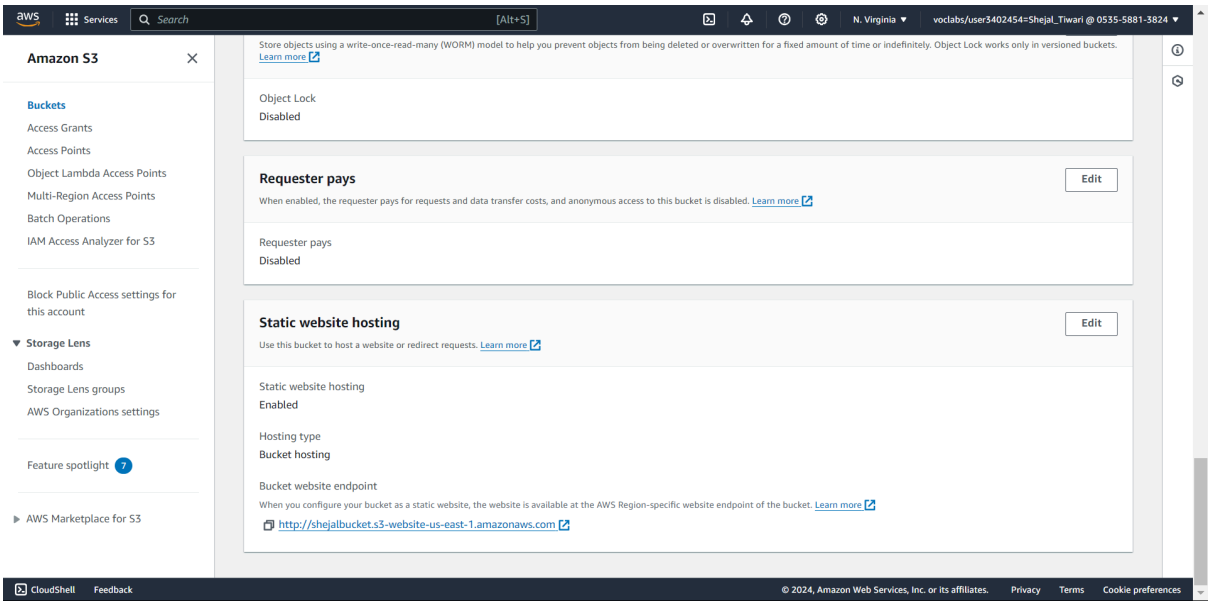
Object Ownership: Decide if you want to disable or enable object ownership. By default, it's set to "Bucket owner preferred."

Block Public Access Settings: You can choose to block all public access to your bucket or configure specific rules. AWS recommends blocking public access unless you specifically need it to be public.



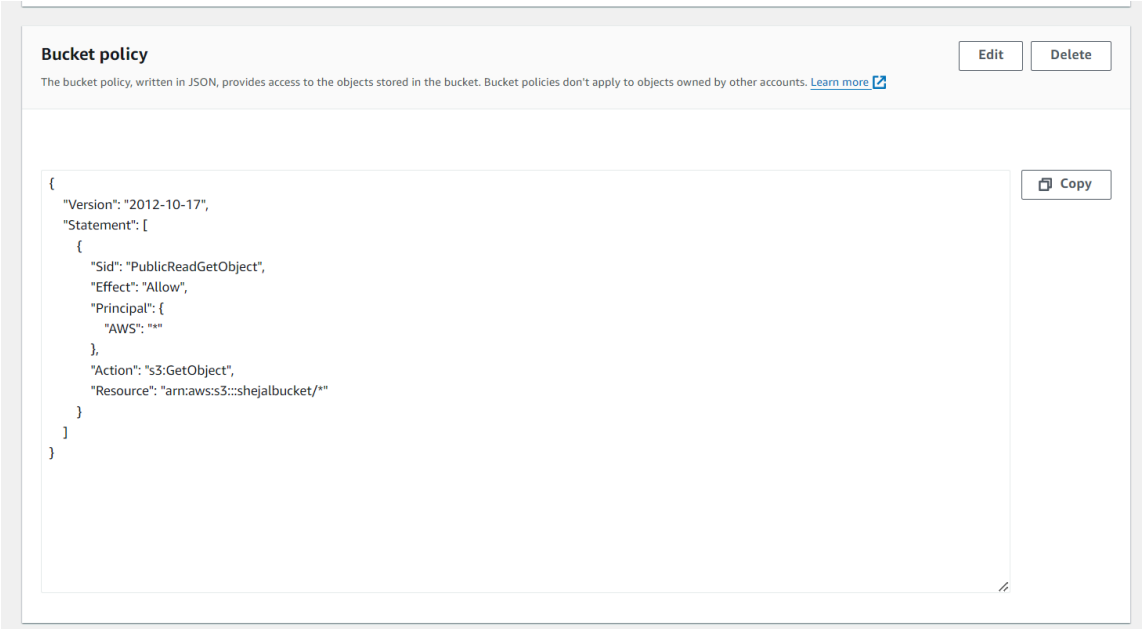
Bucket Versioning: Enable versioning if you want to keep multiple versions of objects in your bucket. This is useful for backup and recovery.

Tags: You can add tags to your bucket to organize and track its costs.



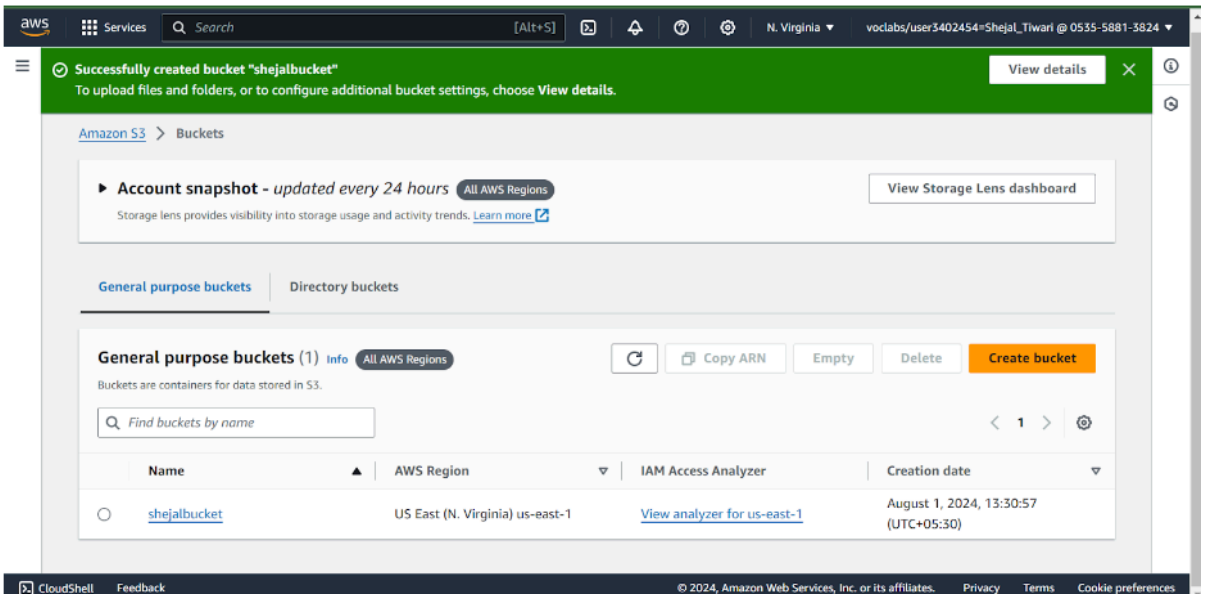
Step 5: Set Policy and Permissions

1. Set permissions based on your requirements. You can keep the default settings for private access or modify them for public or specific user access.



Step 6: Review and Create

Review all the settings you've configured.
If everything looks good, click the "Create bucket" button.



Step 1: Install XAMPP

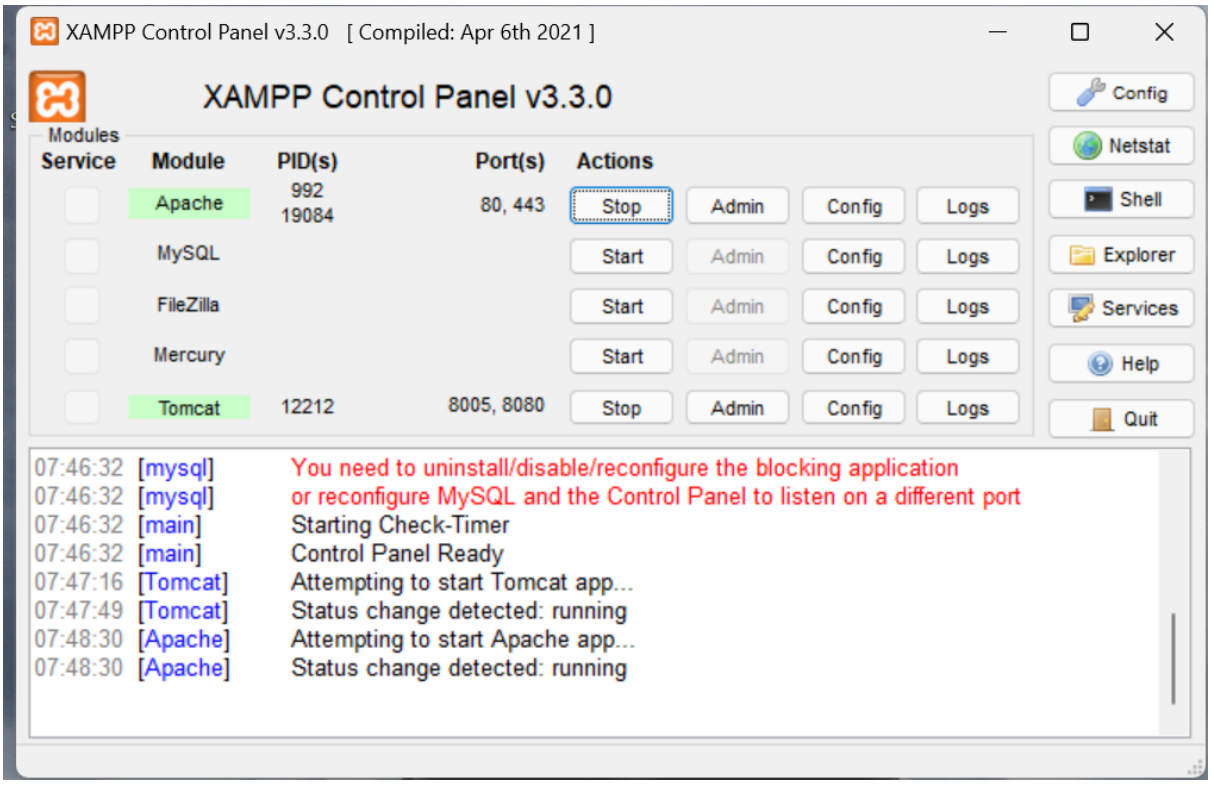
1. **Download XAMPP**: If you haven't already installed XAMPP, you can download it from the official website.
2. **Install XAMPP**: Run the installer and follow the instructions to install XAMPP on your machine.

To host a PHP file on the XAMPP server, follow these steps:

Step 1: Install XAMPP

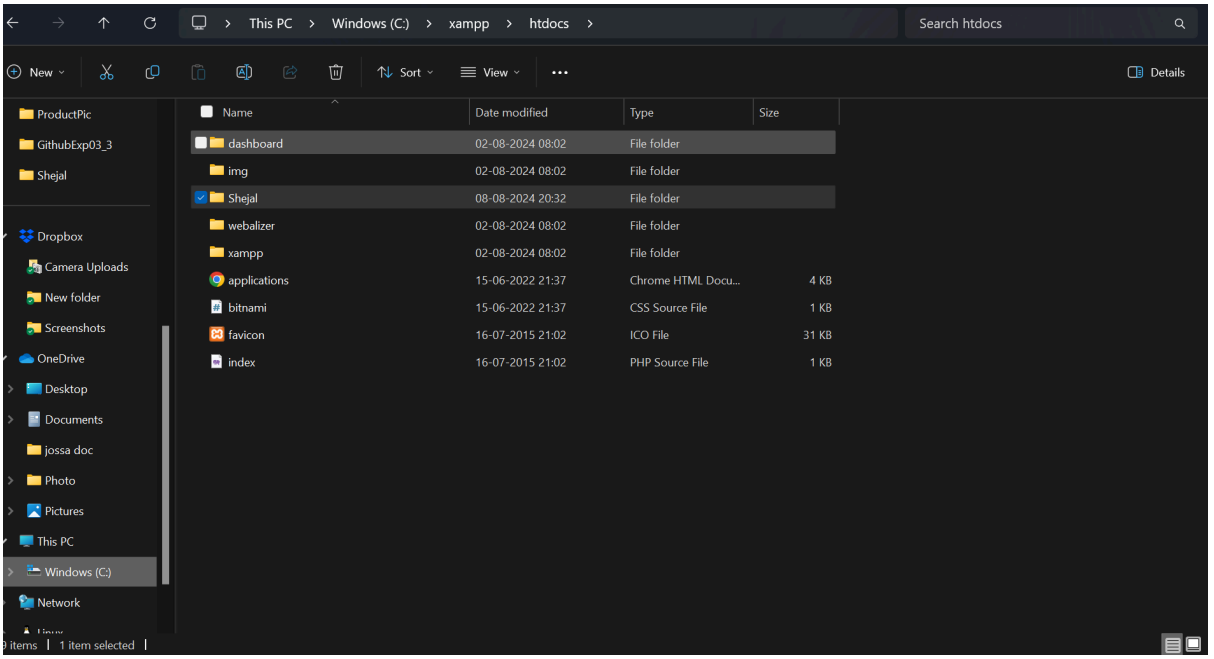
1. **Download XAMPP**: If you haven't already installed XAMPP, you can download it from the official website.
2. **Install XAMPP**: Run the installer and follow the instructions to install XAMPP on your machine.

Step 2: Start Apache Server

1. **Open XAMPP Control Panel**: After installation, open the XAMPP Control Panel.
2. **Start Apache**: Click the "Start" button next to Apache. The status should turn green, indicating that the Apache server is running.
3. 

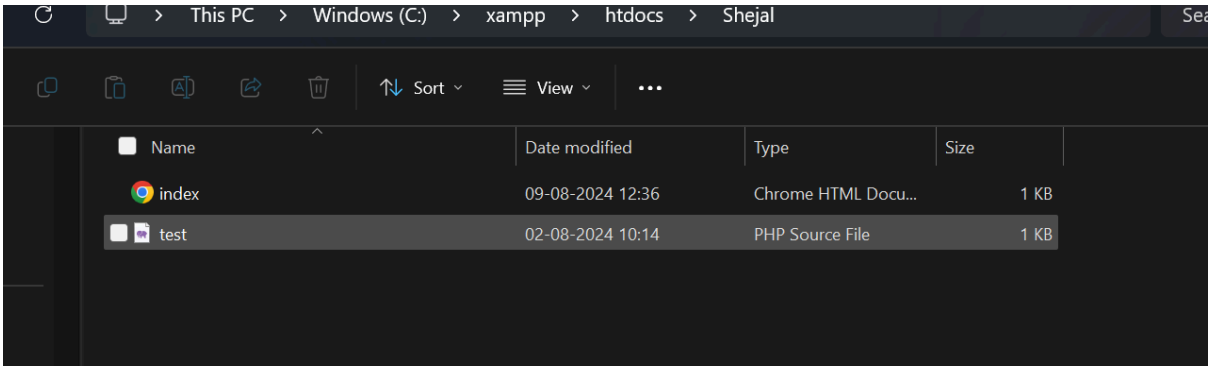
Step 3: Place PHP Files in the htdocs Folder

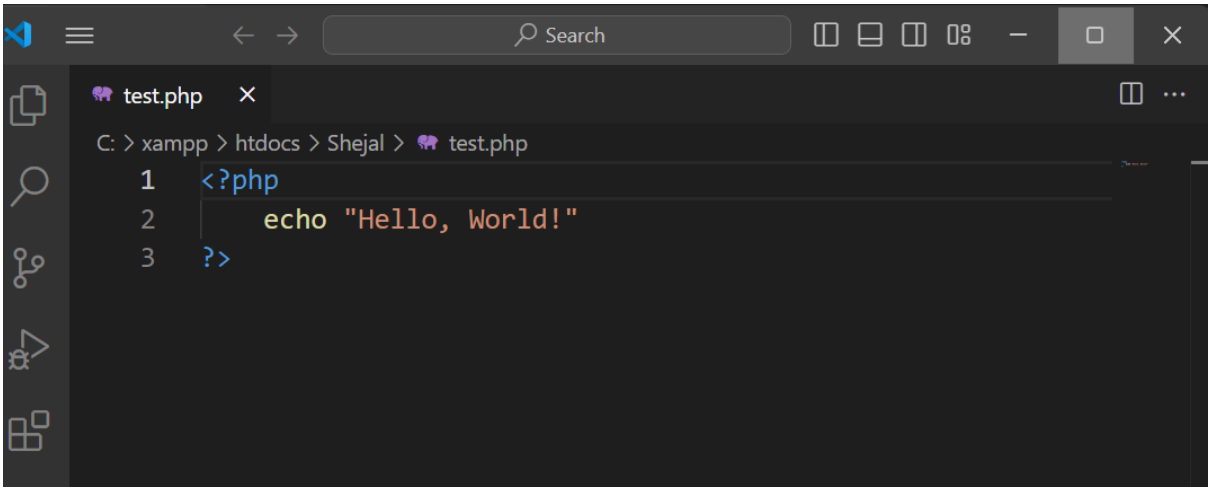
1. **Navigate to XAMPP Directory**: Go to the directory where you installed XAMPP (e.g., C:\xampp on Windows).
2. **Open htdocs Folder**: Inside the XAMPP folder, you'll find a folder named htdocs. This is the root directory where you should place your PHP files.
3. **Create a New Folder (Optional)**: You can create a new folder inside htdocs to organize your project (e.g., C:\xampp\htdocs\myproject).
4. **Place PHP File**: Copy your PHP file (e.g., index.php) into the htdocs directory or your project folder.



Step 4: Access the PHP File via Browser

1. **Open Web Browser**: Open any web browser.
2. **Access the File**: In the address bar, type the following URL:
    - If you placed the file directly in htdocs, use: http://localhost/filename.php
    - If you placed the file inside a folder, use: http://localhost/foldername/filename.php
    - For example, if your file is index.php inside myproject folder, the URL would be: http://localhost/myproject/index.php





Step 5: View the Output

- The PHP file will be executed on the server, and the output will be displayed in your browser.

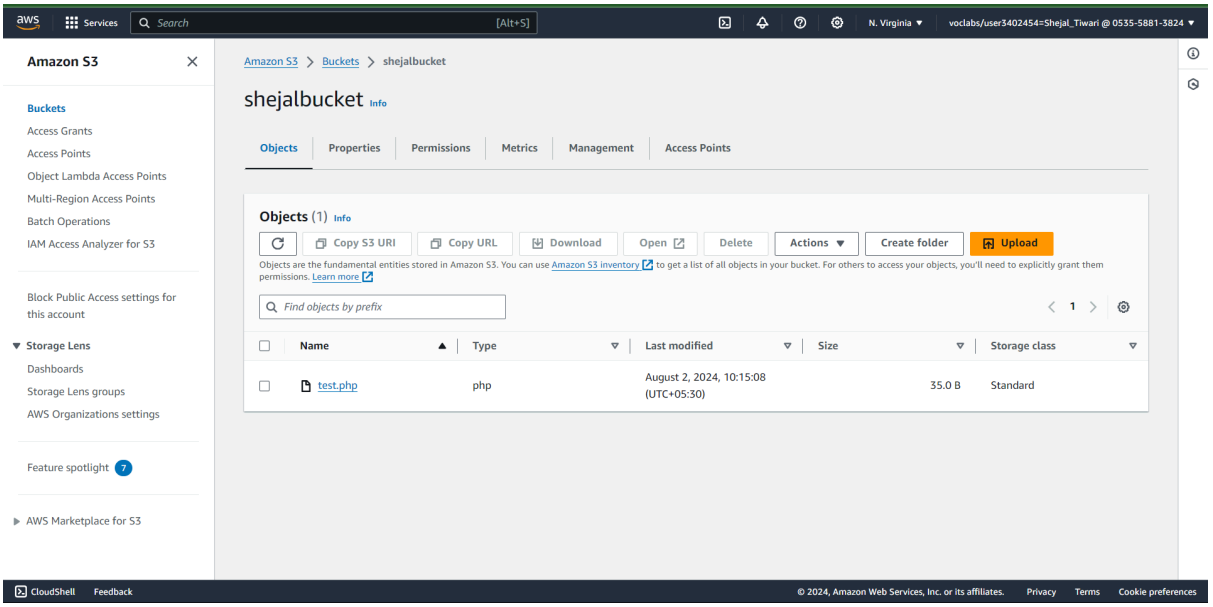
Additional Tips:

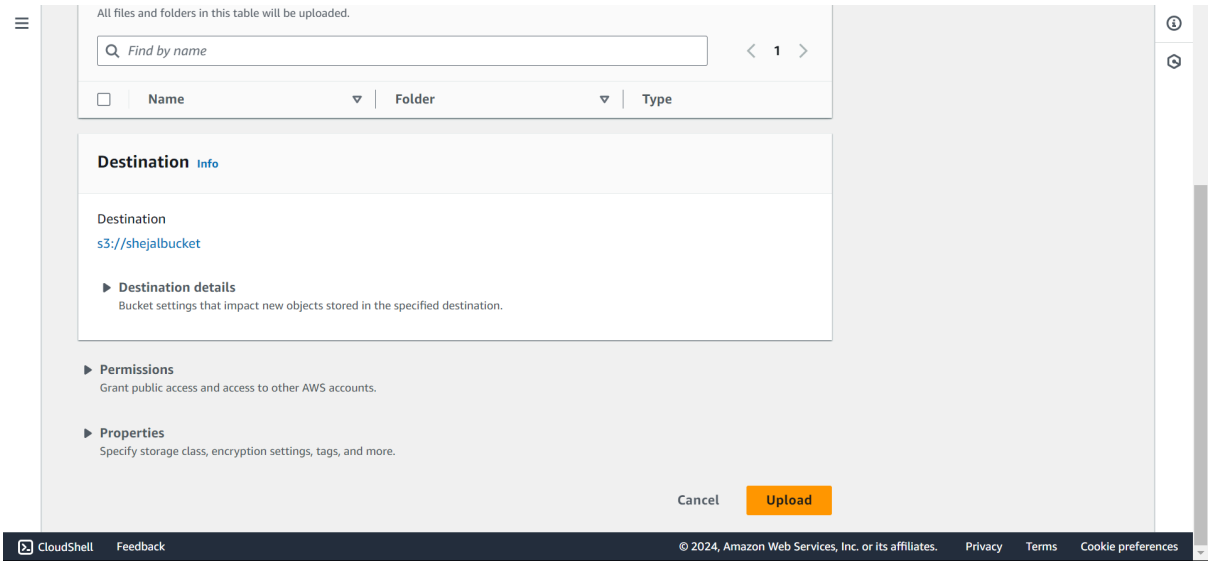
- **PHP Errors**: If there are errors in your PHP code, they will be displayed in the browser. You can also check the error logs in the XAMPP Control Panel under Apache > Logs > PHP Error Log.

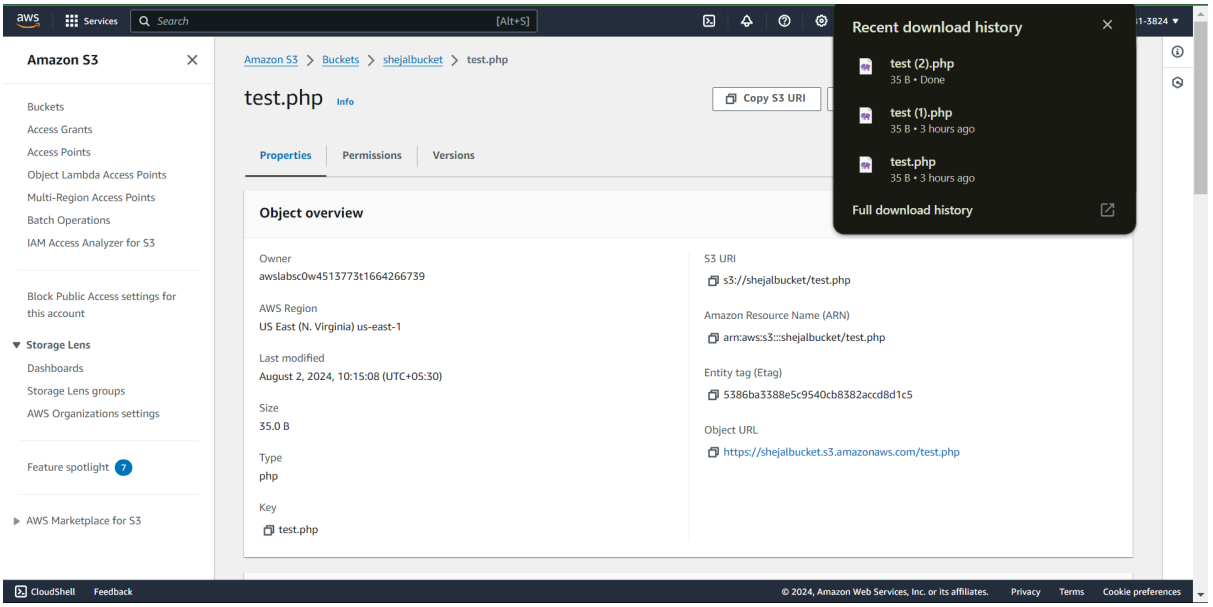
Your PHP file is now hosted on your local XAMPP server!

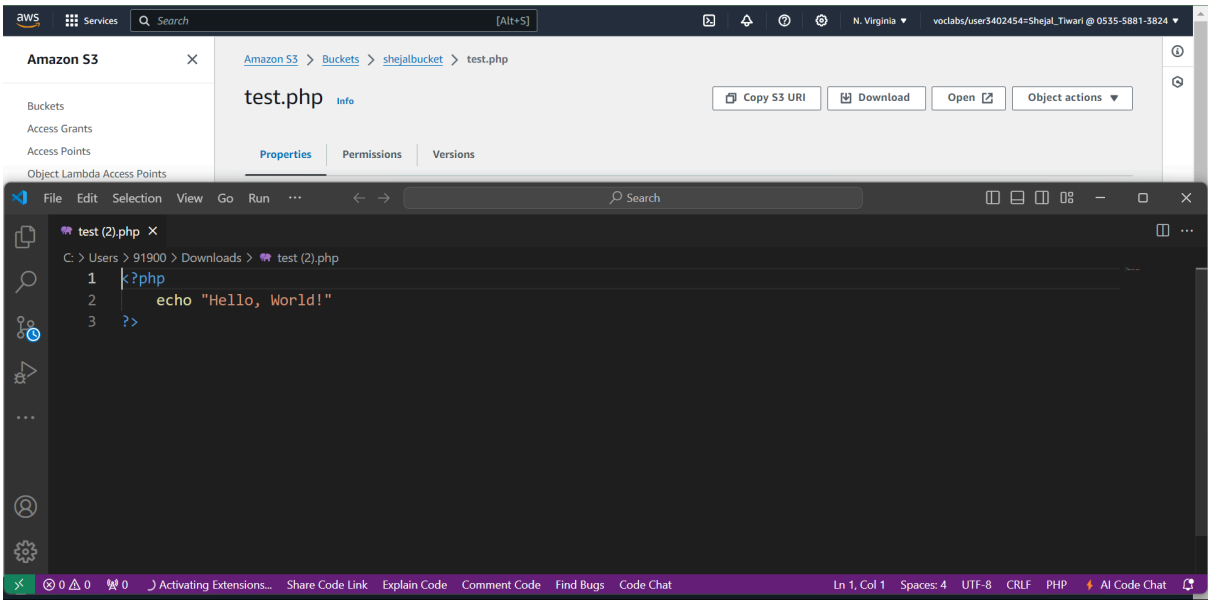
Step 7: Upload Objects (Optional)

1. Once the bucket is created, you can start uploading objects to it by clicking on the bucket name and using the "Upload" button.

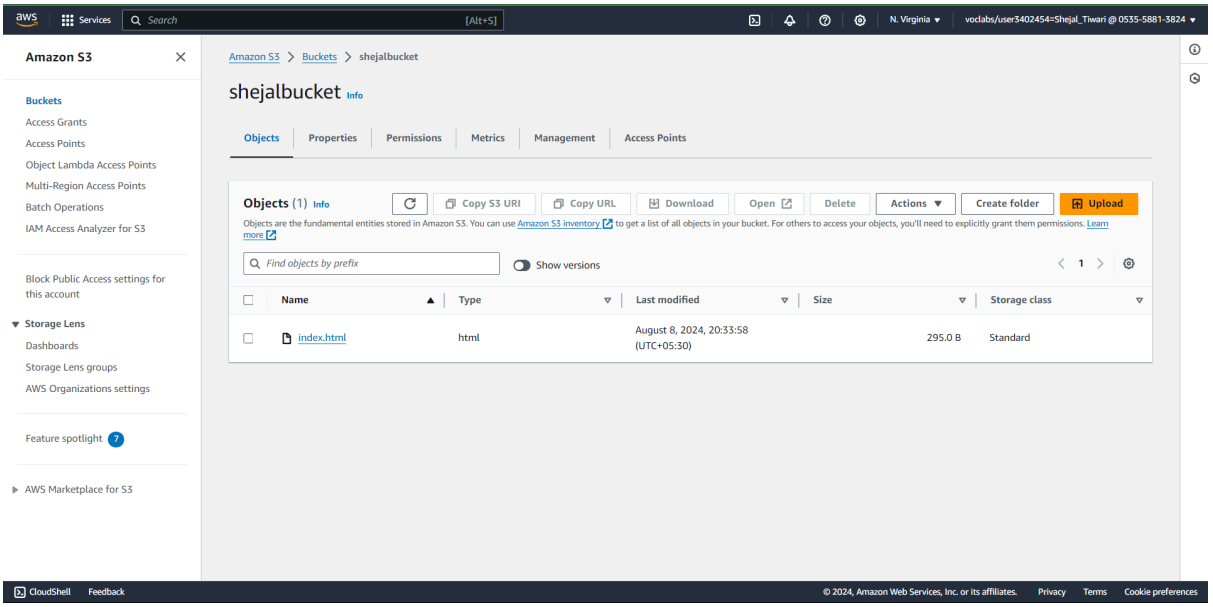




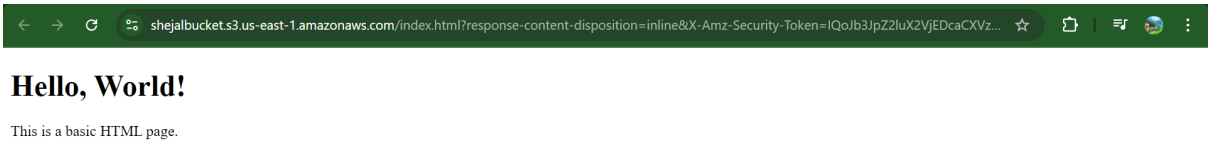

Recent download history
test (2).php
35 B • Done
test (1).php
35 B • 3 hours ago
test.php
35 B • 3 hours ago
Full download history
Amazon S3
Buckets
Access Grants
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
IAM Access Analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
Storage Lens groups
AWS Organizations settings
Feature spotlight
AWS Marketplace for S3
Amazon S3 > Buckets > shejalbucket > test.php
test.php Info
Copy S3 URI
Properties
Permissions
Versions
Object overview
Owner
awslabsc0w4513773t1664266739
AWS Region
US East (N. Virginia) us-east-1
Last modified
August 2, 2024, 10:15:08 (UTC+05:30)
Size
35.0 B
Type
php
Key
test.php
S3 URI
s3://shejalbucket/test.php
Amazon Resource Name (ARN)
arn:aws:s3:::shejalbucket/test.php
Entity tag (Etag)
5386ba3388e5c9540cb8382accd8d1c5
Object URL
https://shejalbucket.s3.amazonaws.com/test.php
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Amazon S3
Buckets
Access Grants
Access Points
Object Lambda Access Points
Amazon S3 > Buckets > shejalbucket > test.php
test.php Info
Copy S3 URI
Download
Open
Object actions
Properties
Permissions
Versions
File Edit Selection View Go Run
test (2).php
C: > Users > 91900 > Downloads > test (2).php
1 <?php
2 echo "Hello, World!"
3 ?>
Activating Extensions...
Share Code Link
Explain Code
Comment Code
Find Bugs
Code Chat
Ln 1, Col 1
Spaces: 4
UTF-8
CRLF
PHP
AI Code Chat

In similar way it is done for index.html file



Click on the URL and your website is hosted on cloud .



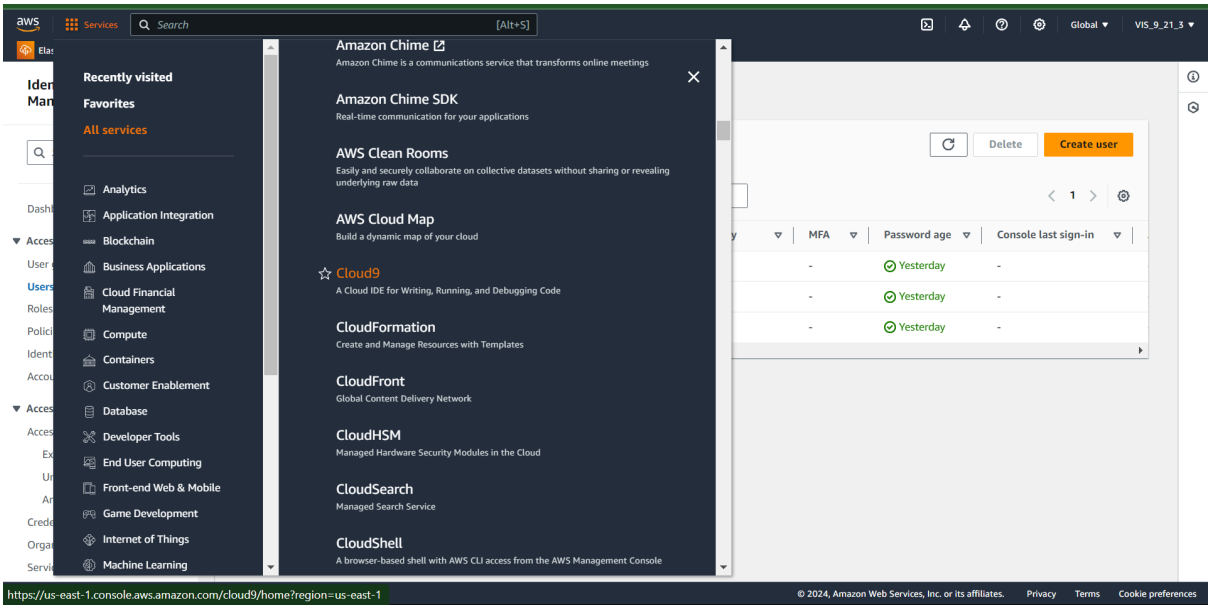
EXPERIMENT 1B

Step 1: Sign in to AWS Management Console

1. Go to the AWS Management Console.
2. Sign in with your AWS credentials.

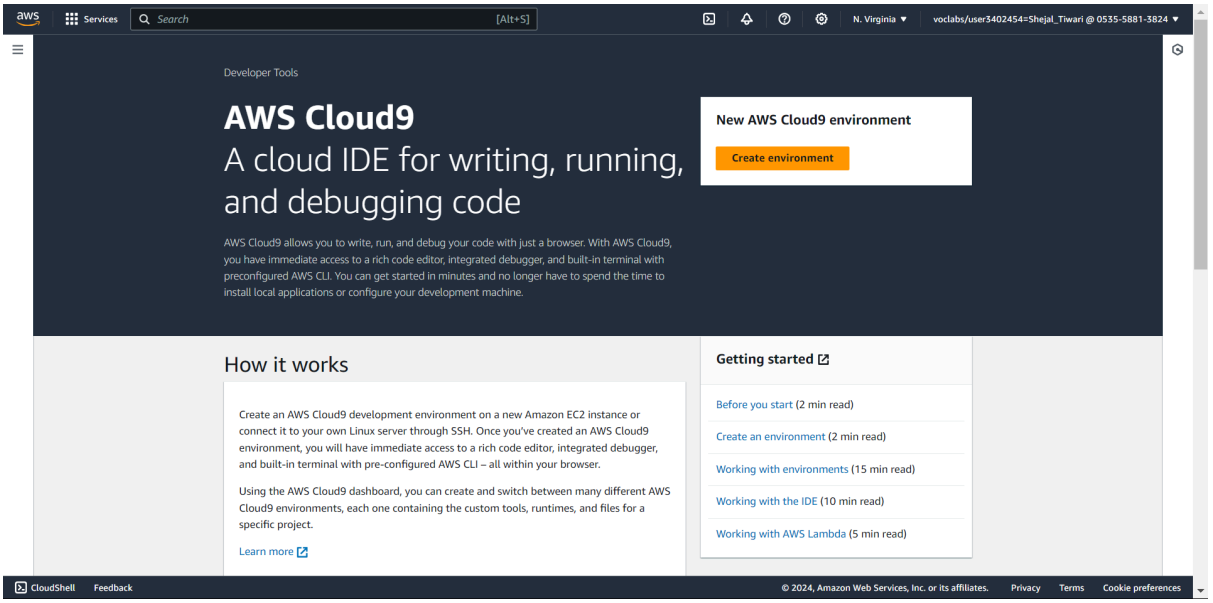
Step 2: Navigate to Cloud9

1. In the AWS Management Console, search for "Cloud9" in the search bar at the top.
2. Click on "Cloud9" under Services.



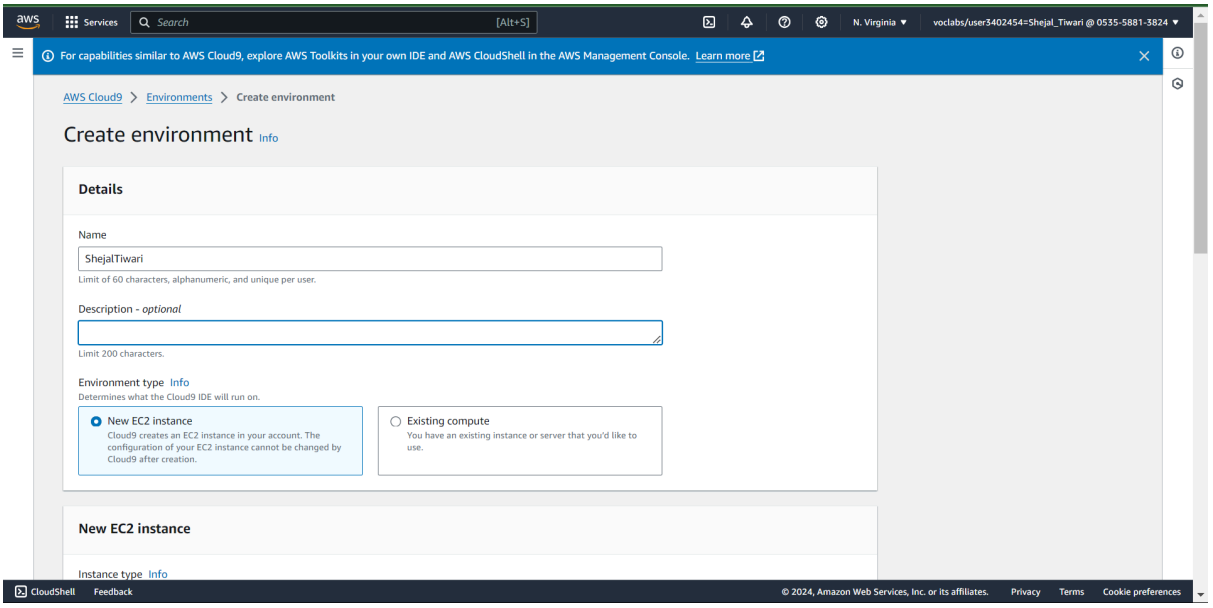
Step 3: Create a New Environment

1. On the Cloud9 dashboard, click on the "Create environment" button.

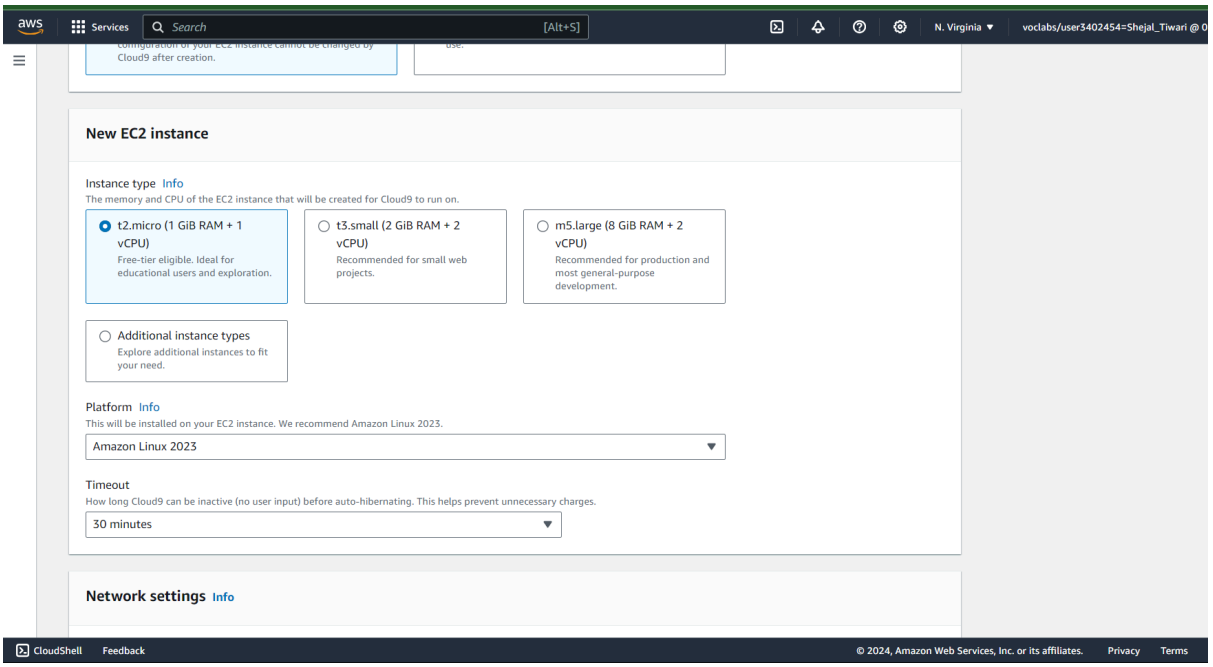


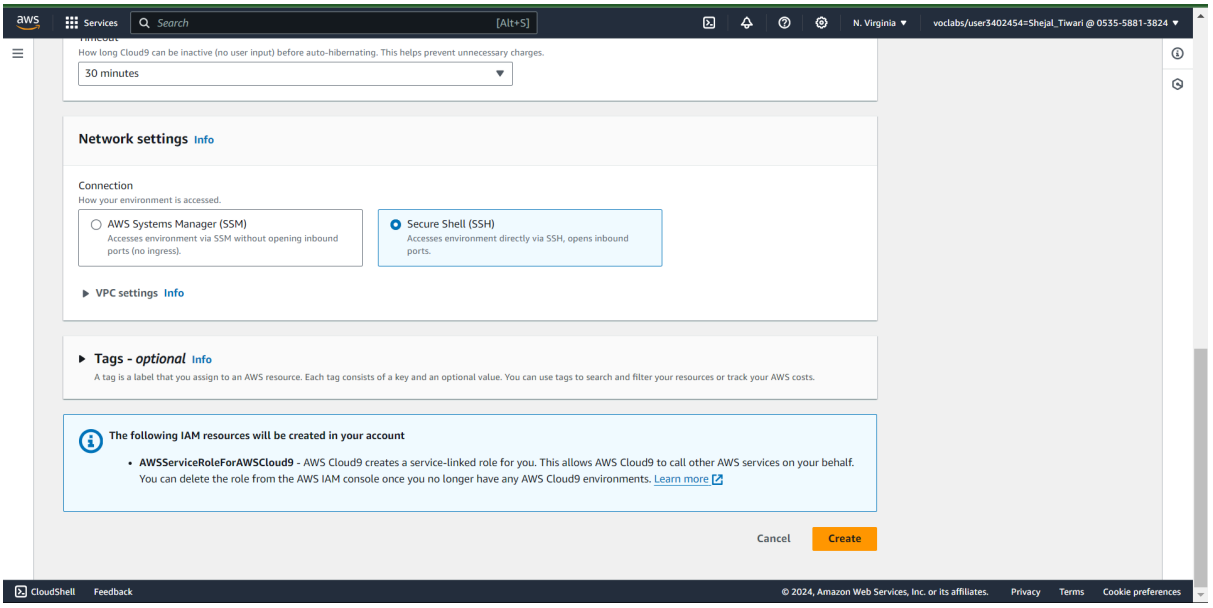
Step 4: Configure Environment Settings

1. **Name and Description**:
    - **Name**: Enter a name for your environment.
    - **Description**: Optionally, provide a description of the environment.
2. **Environment Settings**:
    - **Environment type**: Choose one of the following:
        - **Create a new EC2 instance for environment**: AWS will automatically create an EC2 instance for you. Choose this if you want to run the environment on a new EC2 instance.
        - **Connect to an existing EC2 instance**: If you have an existing EC2 instance, you can select this option to connect Cloud9 to it.
        - **Connect to your own server via SSH**: Use this if you want to connect Cloud9 to an on-premises server or another cloud provider's server.
    - **Instance type**: Select the instance type (e.g., t2.micro for free tier).
    - **Platform**: Choose the platform that suits your development needs (e.g., Amazon Linux, Ubuntu, etc.).
    - **Cost-saving setting**: Set an automatic timeout to stop the EC2 instance when it's not in use, saving costs.



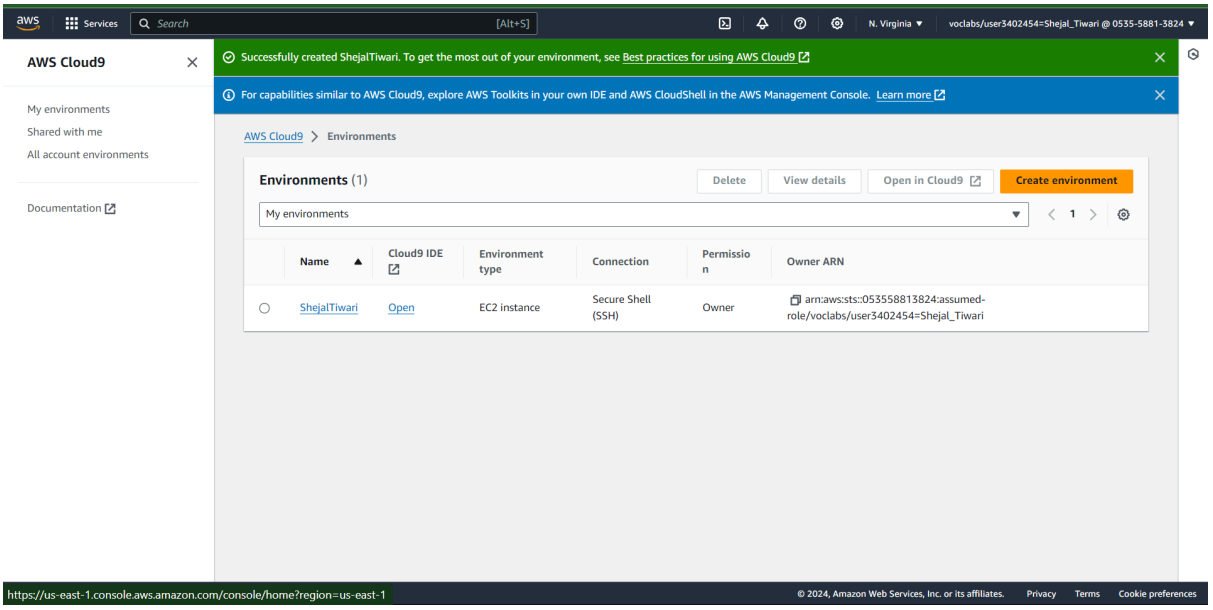
3. **Network Settings**:
    - If you selected "Create a new EC2 instance for environment," you'll need to choose a VPC (Virtual Private Cloud) and a subnet. Usually, the default options will work, but you can customize them if needed.





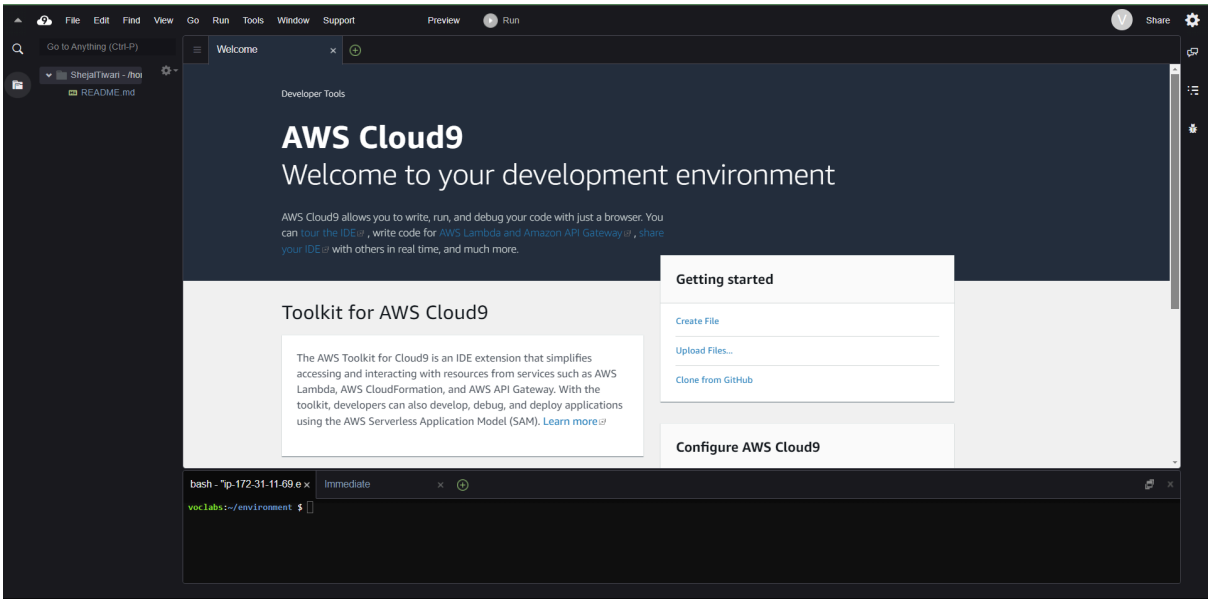
Step 5: Review and Create

1. Review all the settings you configured.
2. If everything looks good, click on the "Create environment" button.



Step 6: Access Your Cloud9 Environment

1. After a few moments, your Cloud9 environment will be ready, and you'll be automatically redirected to the Cloud9 IDE.
2. You can start coding immediately, using the terminal to install packages, run scripts, or manage files as you would on a local machine.



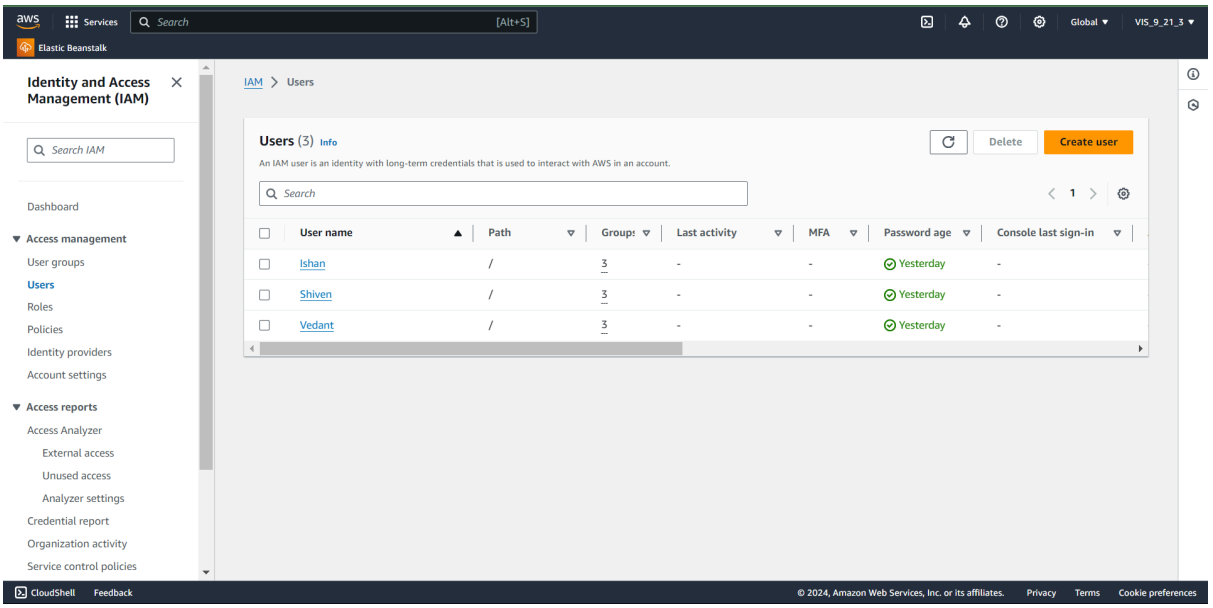
Additional Configuration (Optional)

- **Install additional tools**: You can install additional tools and software using the terminal within Cloud9, just as you would on a regular Linux server.
- **Clone repositories**: Use Git within Cloud9 to clone repositories, manage version control, and collaborate on code.

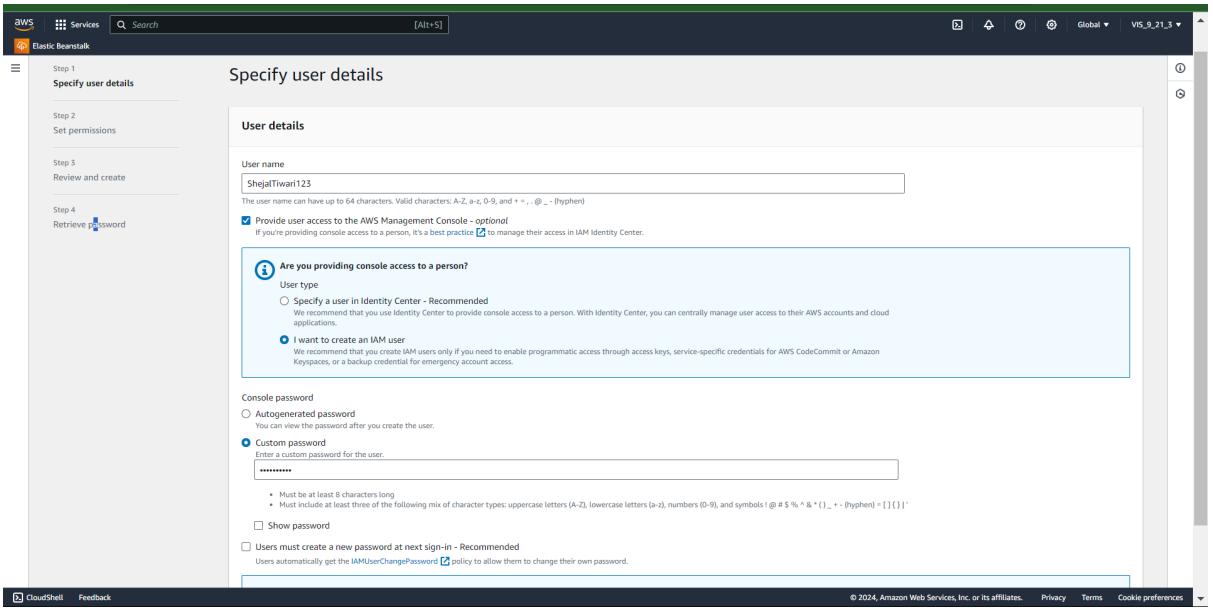
2. Creating an IAM User in AWS

To create a new IAM user in AWS:

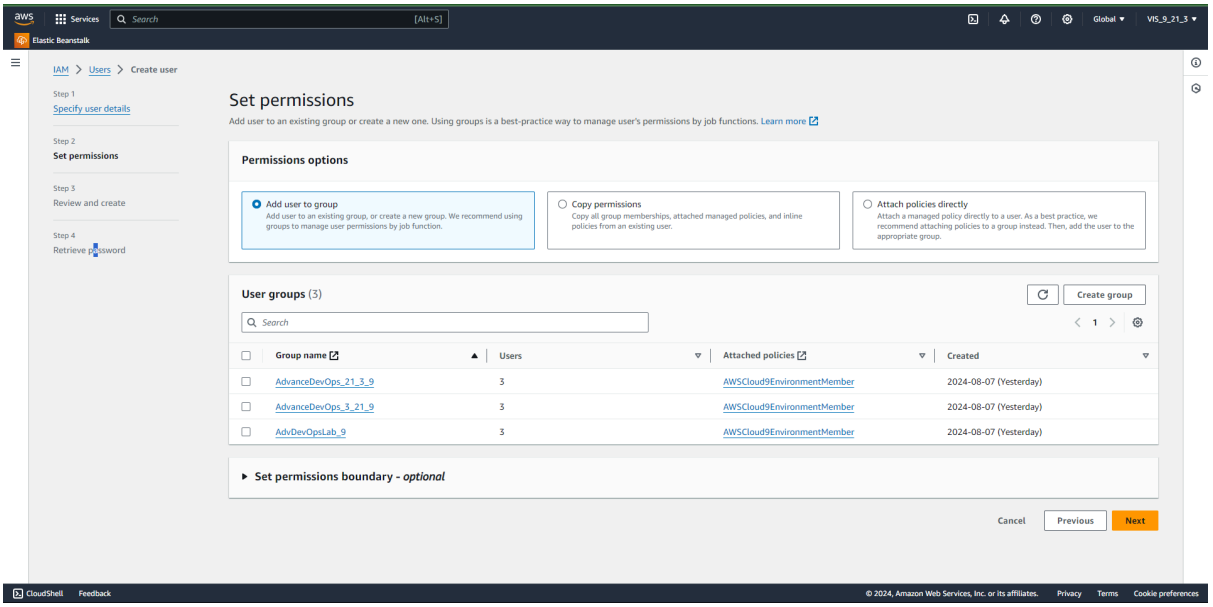
1. **Sign in to AWS Management Console**:
    - Go to the AWS Management Console.
2. **Navigate to IAM**:
    - Search for "IAM" in the services menu and select it.
3. **Create a New User**:
    - Click on "Users" in the left-hand menu.
    - Click on "Add user."



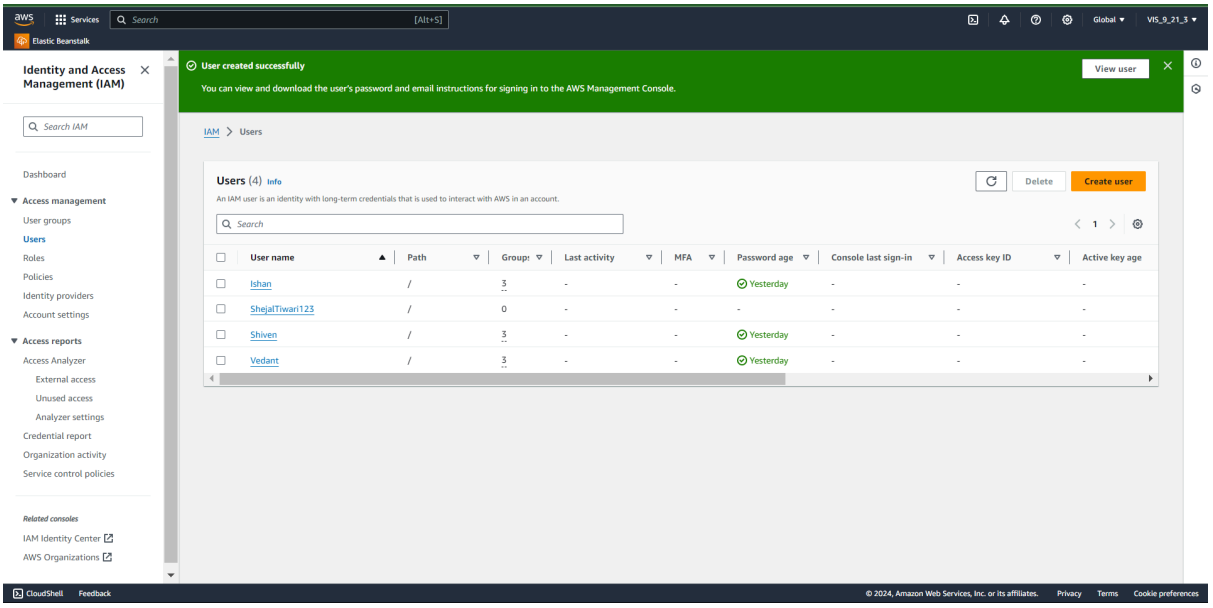
4. **Set User Details**:
    - **User Name**: Enter a name for the new user.
    - **Access Type**:
        - **Programmatic access**: If the user needs access to AWS CLI, SDK, or API.
        - **AWS Management Console access**: If the user needs access to the AWS Management Console.



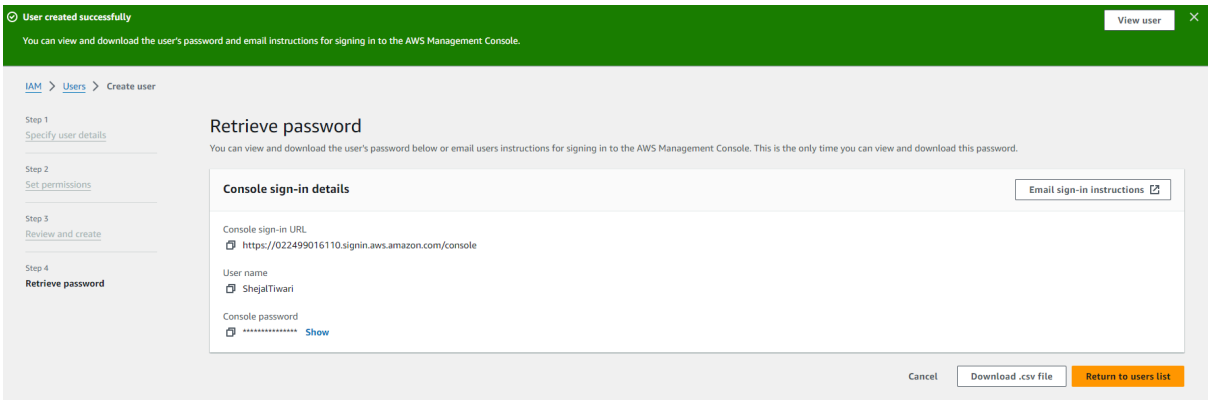
5. **Set Permissions**:
    - You can attach existing policies directly, add the user to a group with predefined permissions, or copy permissions from another user.



6. **Tags (Optional)**:
    - Add tags to the user account for easy identification or organization.
7. **Review and Create**:
    - Review the settings and click "Create user."



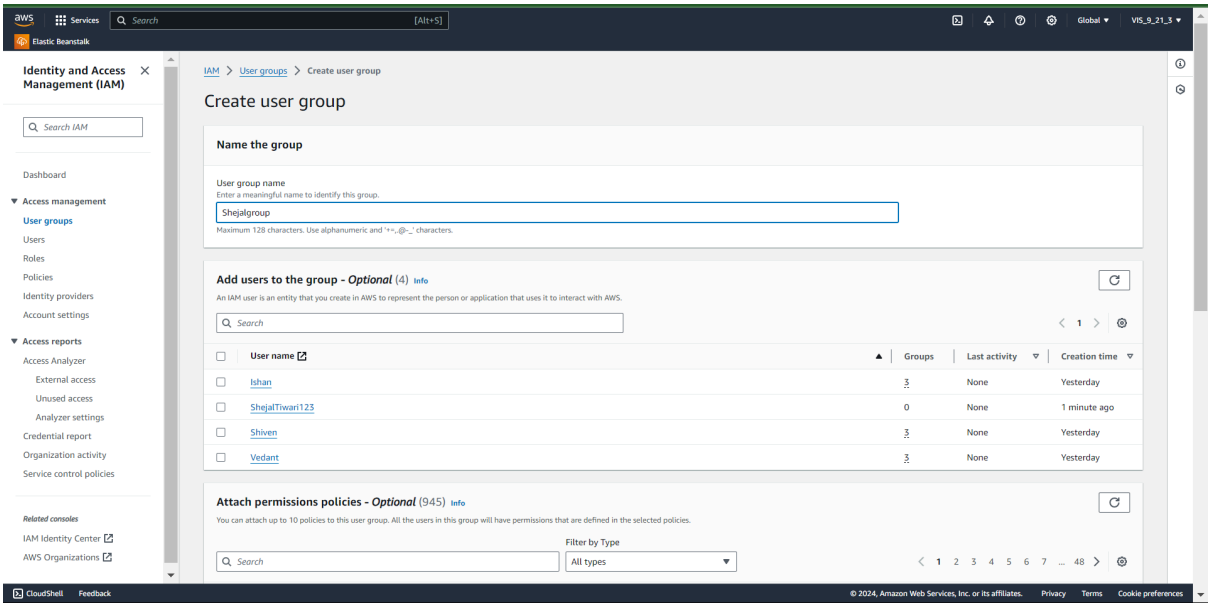
8. **Download Credentials**:
    - On the confirmation page, download the access key and secret key if you've enabled programmatic access. Keep them secure.



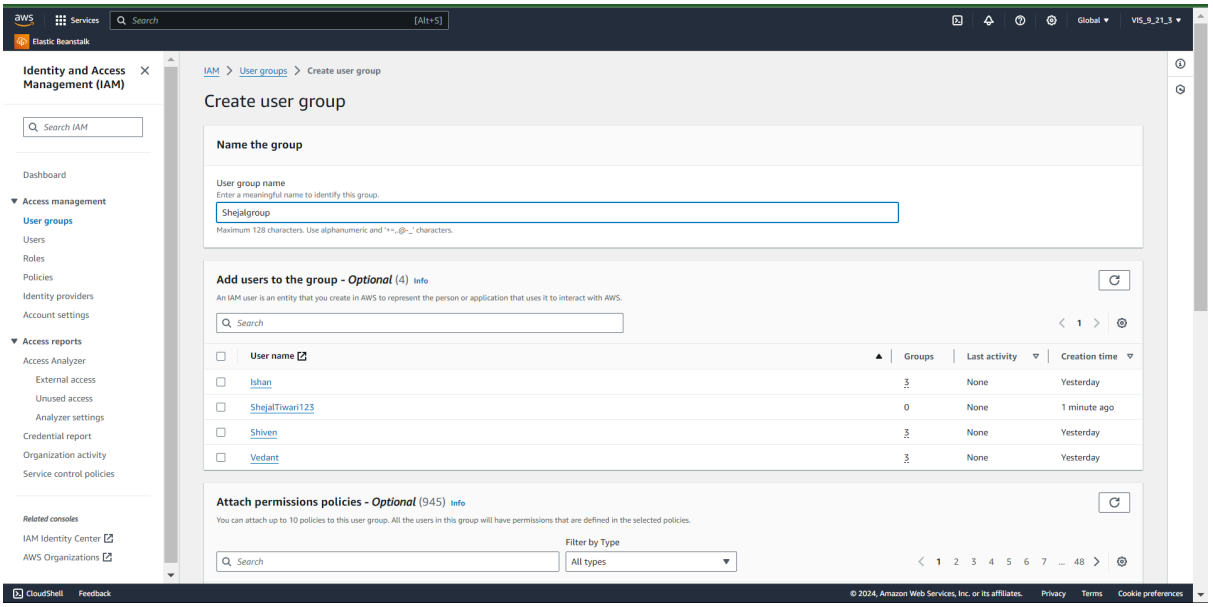
Creating a User Group in AWS IAM

To create a new user group in AWS IAM:

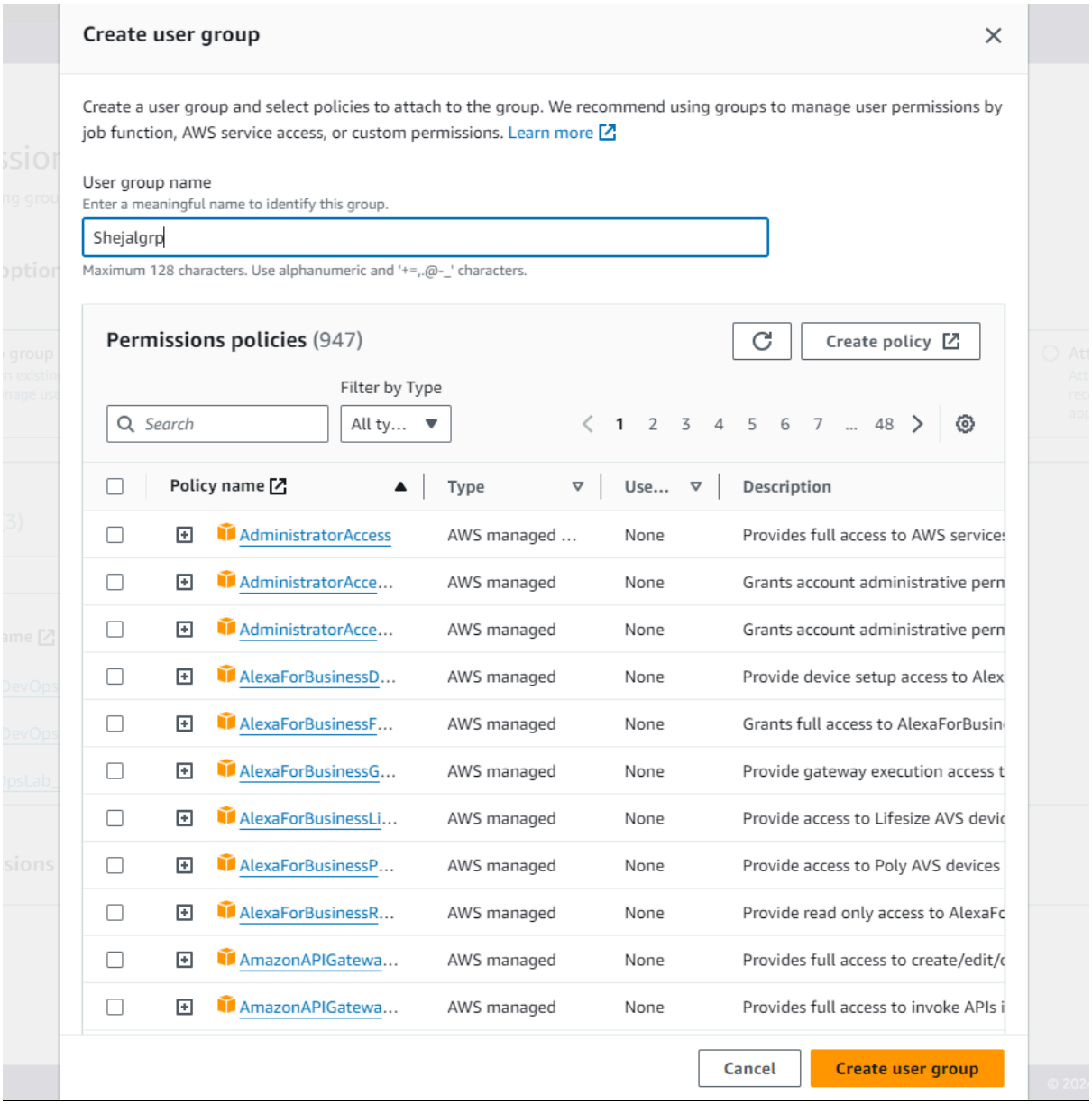
1. **Sign in to AWS Management Console**:
    - Go to the AWS Management Console.
2. **Navigate to IAM**:
    - Search for "IAM" in the services menu and select it.
3. **Create a New Group**:
    - Click on "User groups" in the left-hand menu.
    - Click on "Create group."

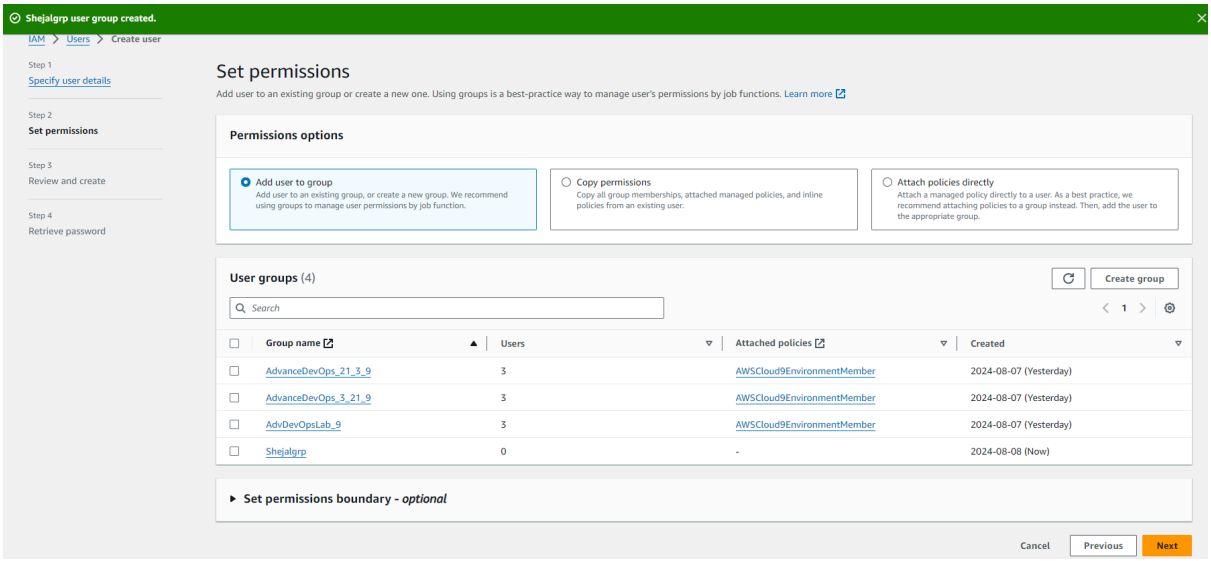


4. **Set Group Name**:
    - Enter a name for your group in the "Group name" field.

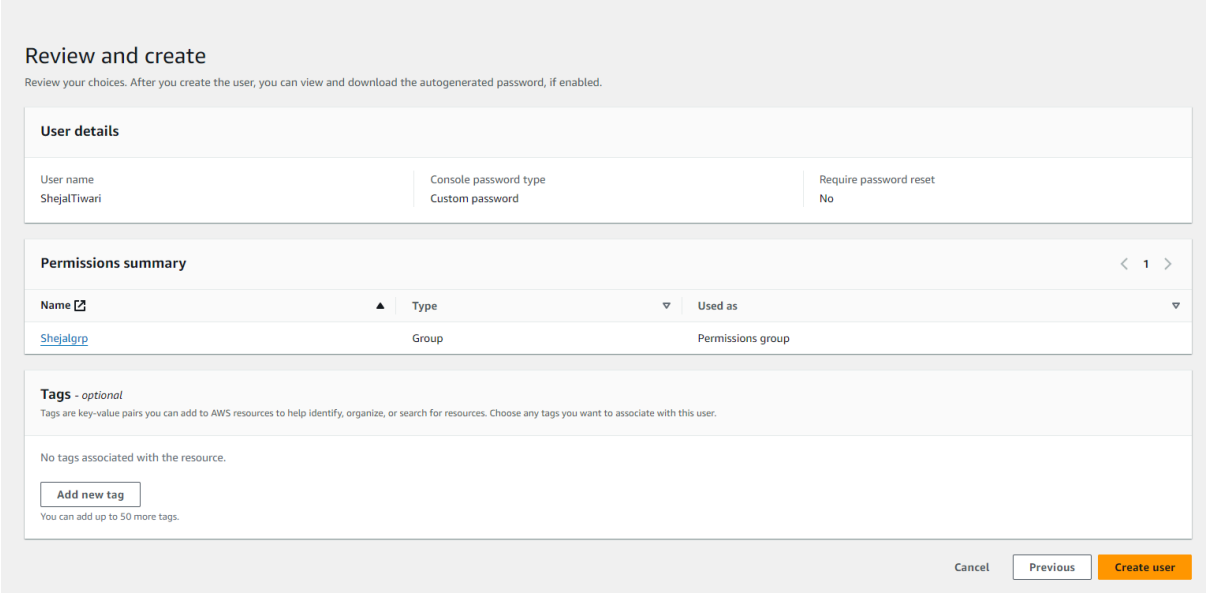


5. **Attach Permissions Policies**:
    - Select the policies you want to attach to the group. These policies define what actions the users in this group can perform.
    - You can attach existing AWS managed policies, or create and attach a custom policy.

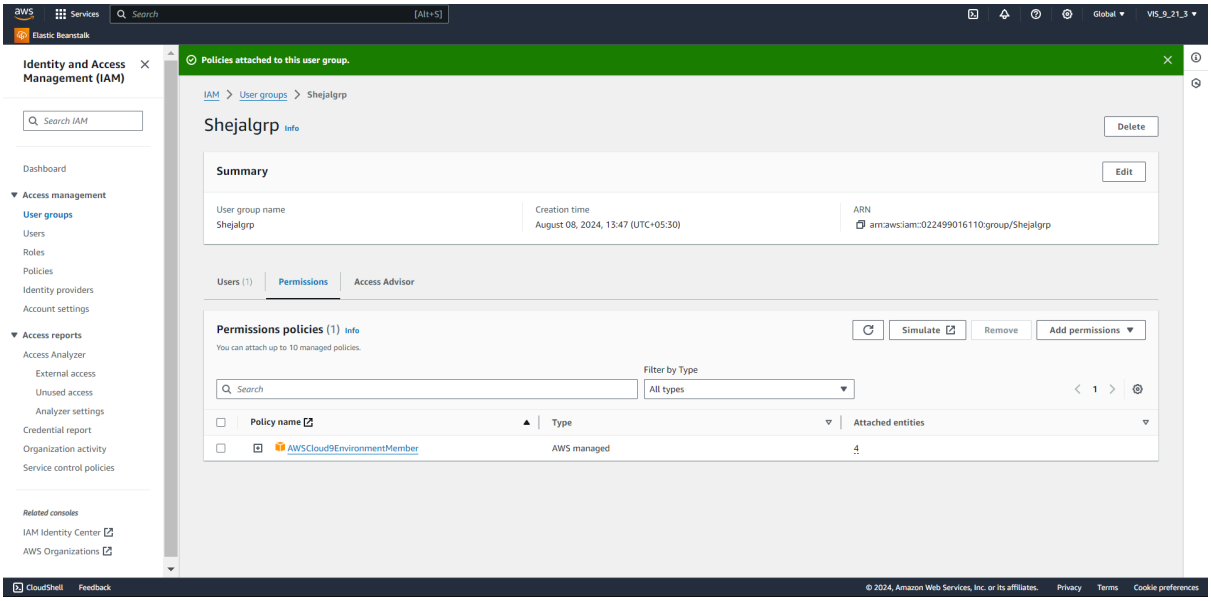

Create user group
Create a user group and select policies to attach to the group. We recommend using groups to manage user permissions by job function, AWS service access, or custom permissions. Learn more
User group name
Enter a meaningful name to identify this group.
Shejalgrp
Maximum 128 characters. Use alphanumeric and '+=,.@-_' characters.
Permissions policies (947)
Create policy
Filter by Type
Search
All ty...
1 2 3 4 5 6 7 ... 48
Policy name
Type
Use...
Description
AdministratorAccess AWS managed ... None Provides full access to AWS service
AdministratorAcce... AWS managed None Grants account administrative perm
AdministratorAcce... AWS managed None Grants account administrative perm
AlexaForBusinessD... AWS managed None Provide device setup access to Alex
AlexaForBusinessF... AWS managed None Grants full access to AlexaForBusin
AlexaForBusinessG... AWS managed None Provide gateway execution access t
AlexaForBusinessLi... AWS managed None Provide access to Lifesize AVS devic
AlexaForBusinessP... AWS managed None Provide access to Poly AVS devices
AlexaForBusinessR... AWS managed None Provide read only access to AlexaFc
AmazonAPIGatewa... AWS managed None Provides full access to create/edit/
AmazonAPIGatewa... AWS managed None Provides full access to invoke APIs i
Cancel
Create user group


Shejalgrp user group created.
IAM > Users > Create user
Step 1
Specify user details
Step 2
Set permissions
Step 3
Review and create
Step 4
Retrieve password
Set permissions
Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. Learn more
Permissions options
Add user to group
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.
Copy permissions
Copy all group memberships, attached managed policies, and inline policies from an existing user.
Attach policies directly
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.
User groups (4)
Create group
Search
Group name
Users
Attached policies
Created
AdvanceDevOps_21_3_9 3 AWSCloud9EnvironmentMember 2024-08-07 (Yesterday)
AdvanceDevOps_3_21_9 3 AWSCloud9EnvironmentMember 2024-08-07 (Yesterday)
AdvDevOpsLab_9 3 AWSCloud9EnvironmentMember 2024-08-07 (Yesterday)
Shejalgrp 0 - 2024-08-08 (Now)
Set permissions boundary - optional
Cancel
Previous
Next

6. **Review and Create**:
    - Review your settings and click on "Create group."



7. **Add Users to the Group** (Optional):
    - After creating the group, you can add users to it by selecting the group, then going to the "Users" tab and clicking on "Add users to group."

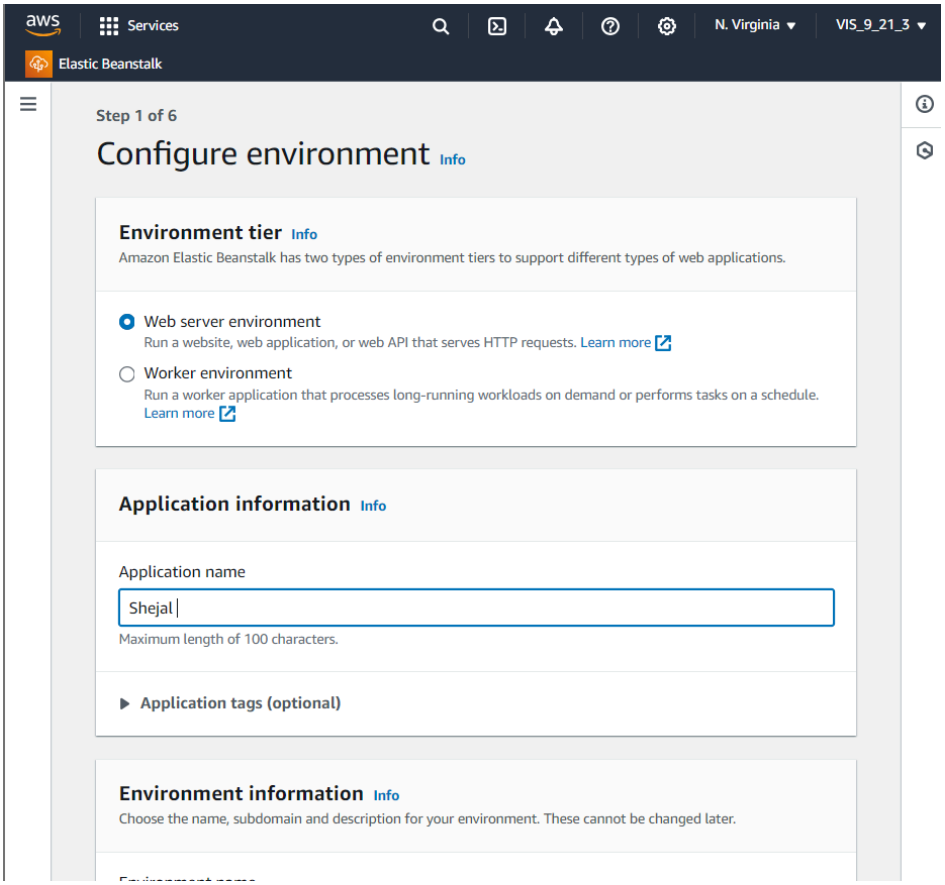


Name:Shejal Nilesh Tiwari Div/Roll No: D15C/57

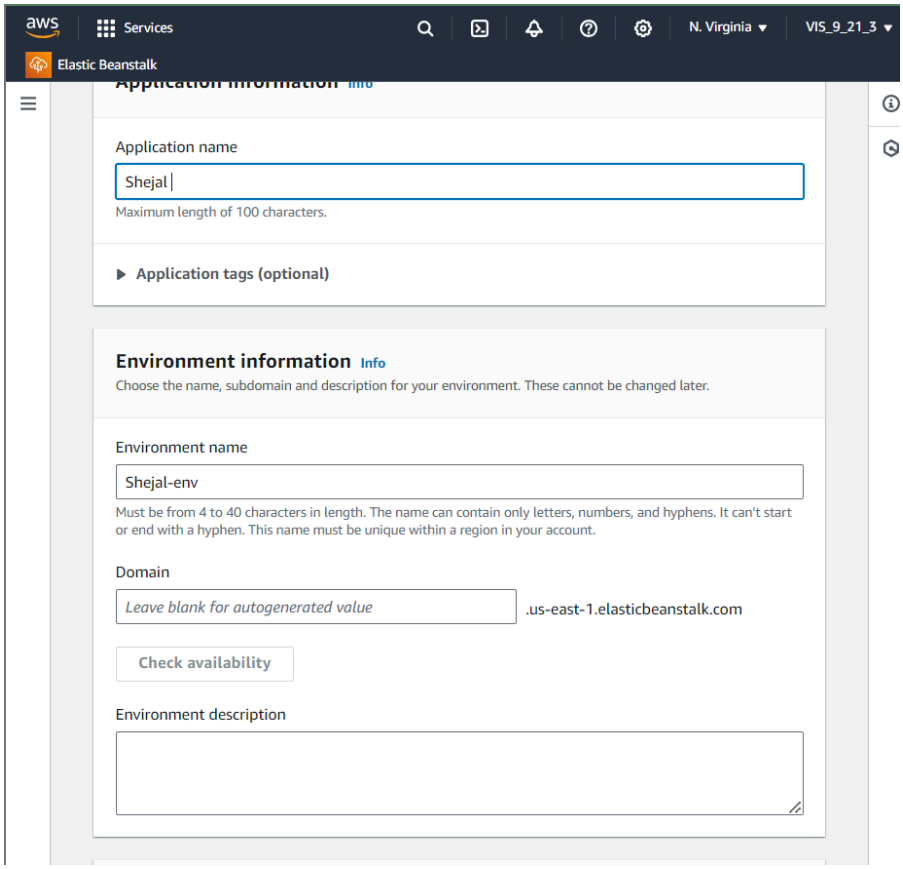
EXPERIMENT NO: 2

Aim: To Build Your Application using AWS CodeBuild and Deploy on S3 / SEBS using AWS CodePipeline, deploy Sample Application on EC2 instance using AWS CodeDeploy. Your continuous deployment pipeline will need a target environment containing virtual servers, or Amazon EC2 instances, where it will deploy sample code. You will prepare this environment before creating the pipeline.

1.Go to the AWS Management Console and navigate to Elastic Beanstalk.
Click on **Create Application**.
Add name to your application

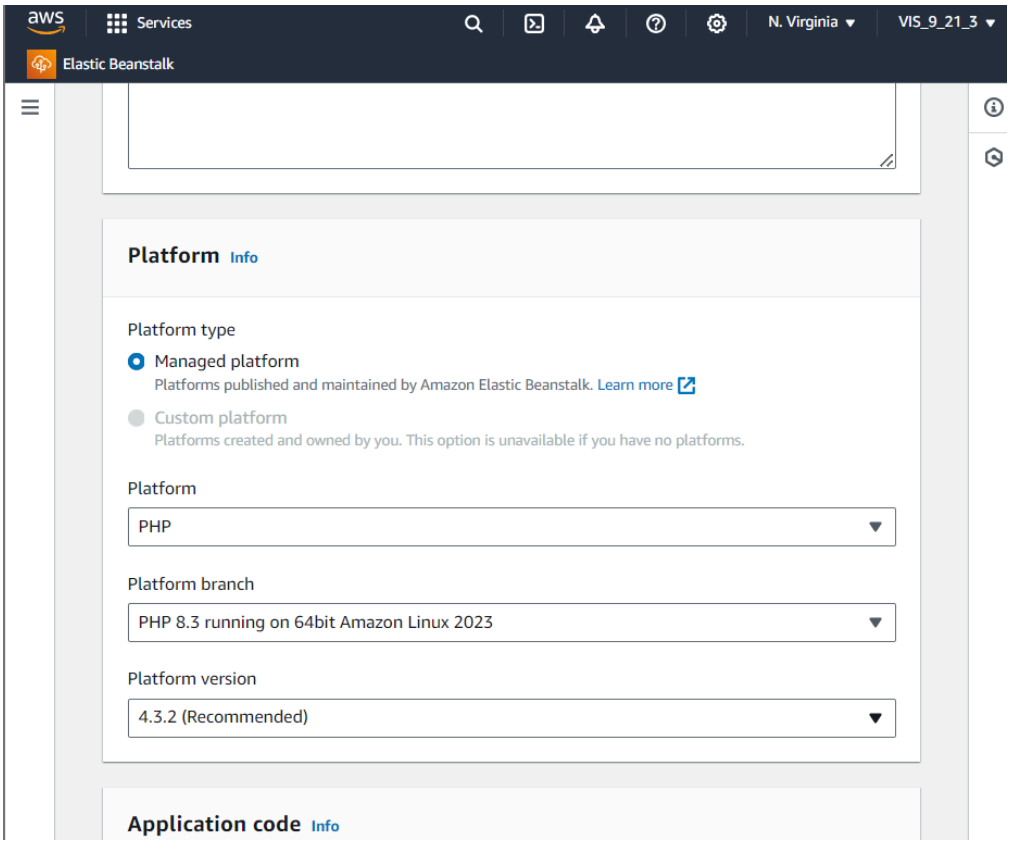


Name:Shejal Nilesh Tiwari Div/Roll No: D15C/57

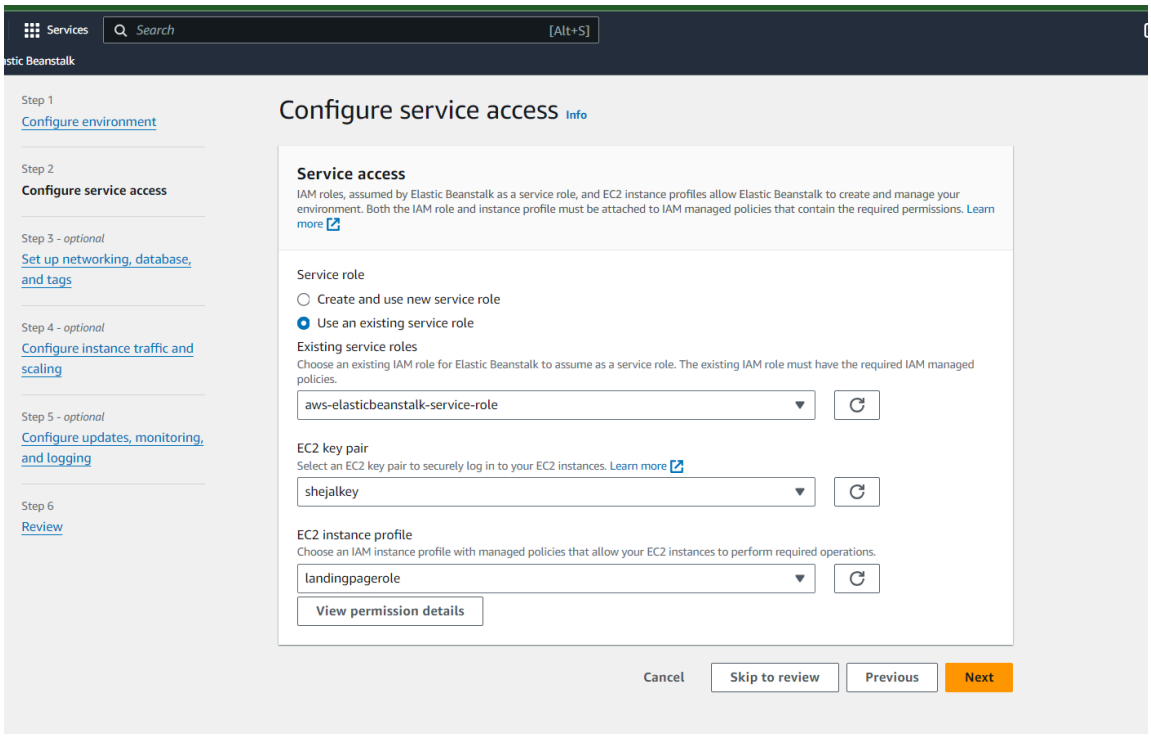


2. Configure PHP Environment:

- Choose PHP as the platform for your Elastic Beanstalk environment, ensuring compatibility with your PHP application.

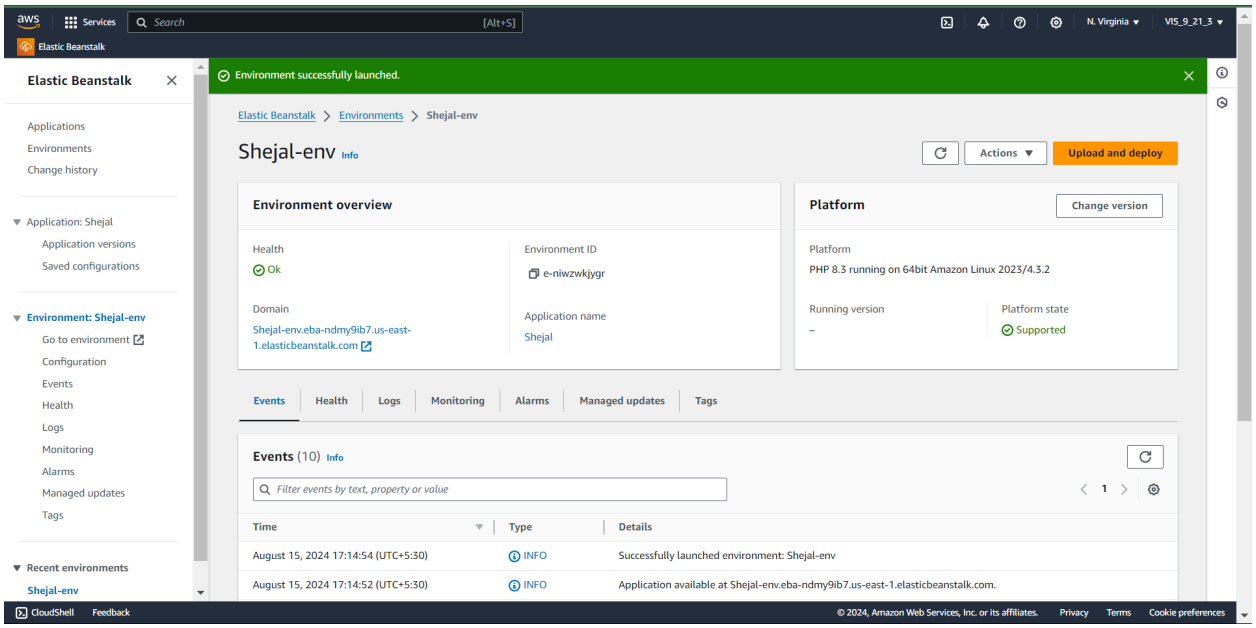


3.In configure service access select “use an existing service role”
Also you need to create key and instance profile first before adding it in the below dropdown.

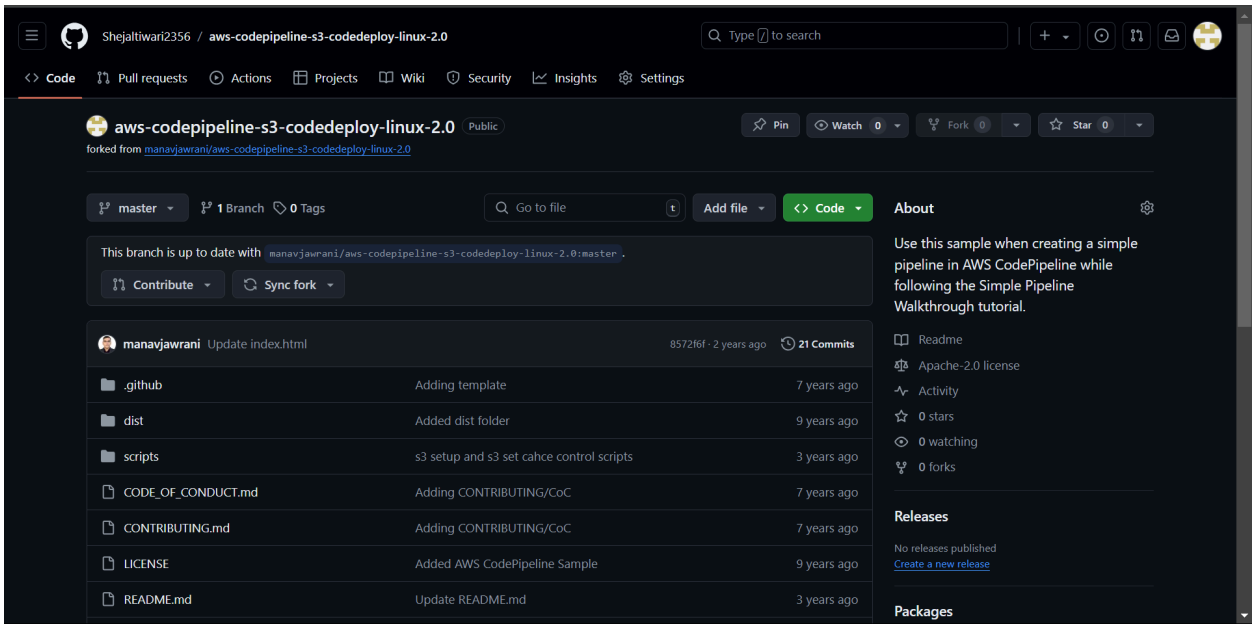


Name:Shejal Nilesh Tiwari Div/Roll No: D15C/57

4.Directly select next to all the optional settings.
Your environment will be created successfully .



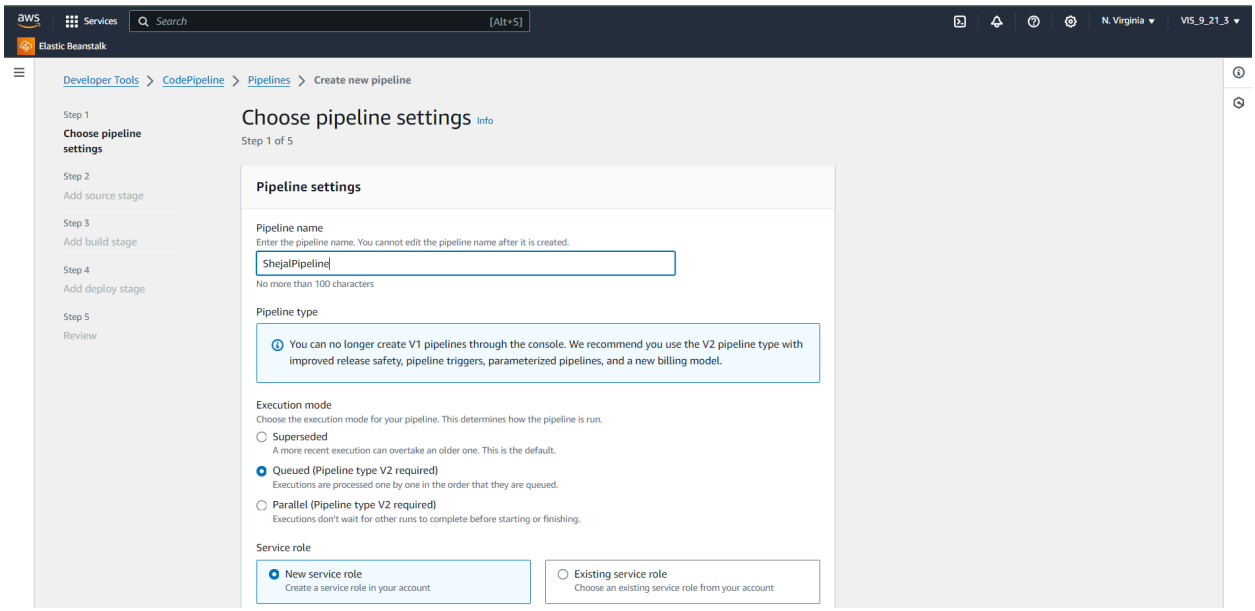
5.You need to fork this repository.



6.Open CodePipeline:

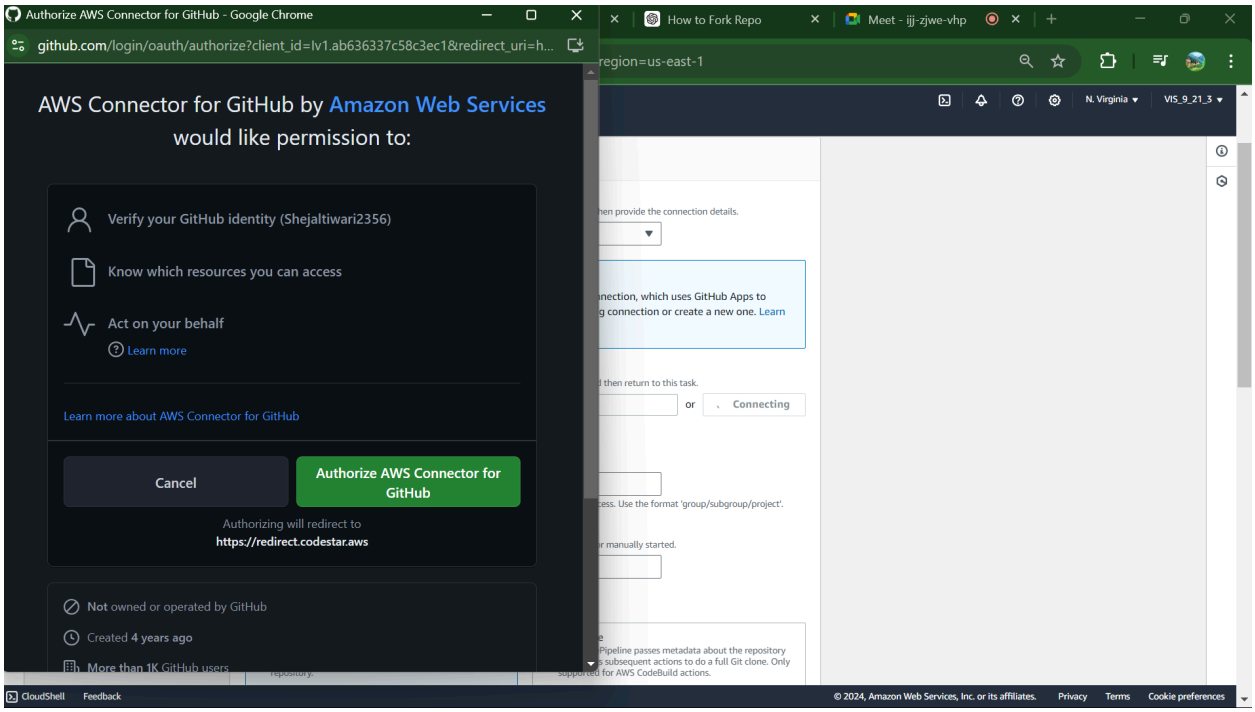
- Navigate to the AWS Management Console and open CodePipeline.
- Click "Create pipeline."

Create a codepipeline add name for your pipeline.



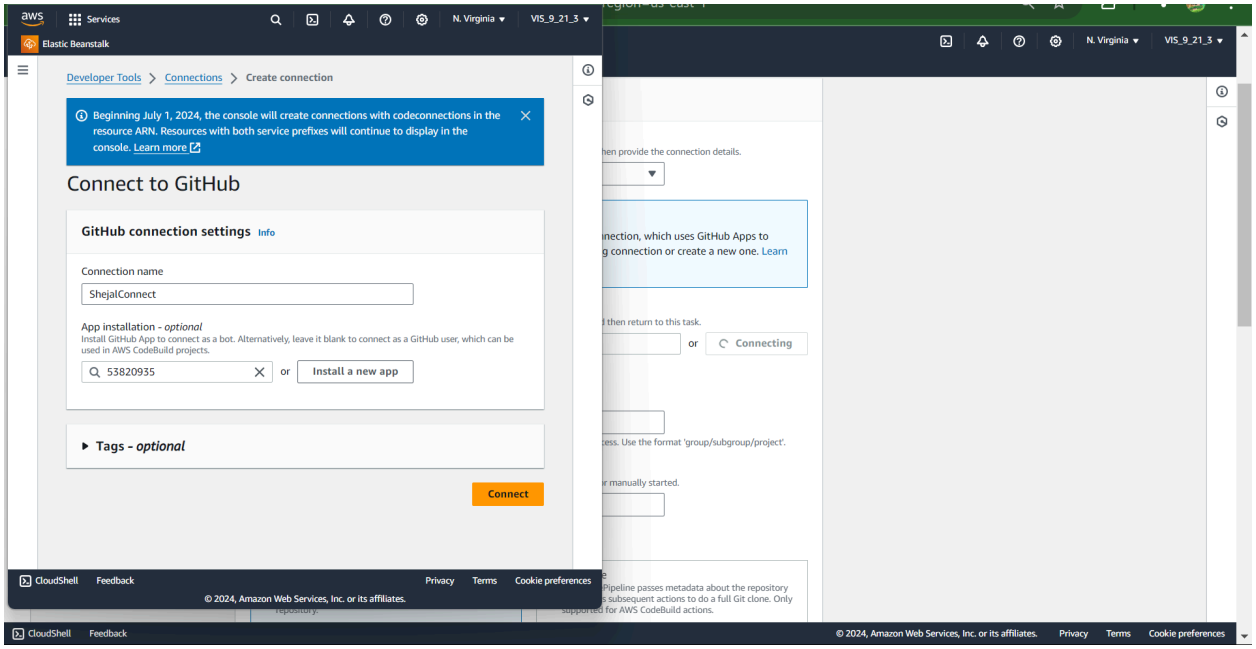
7.After selecting the github version

We need to connect it to the repository for that we will add git connect

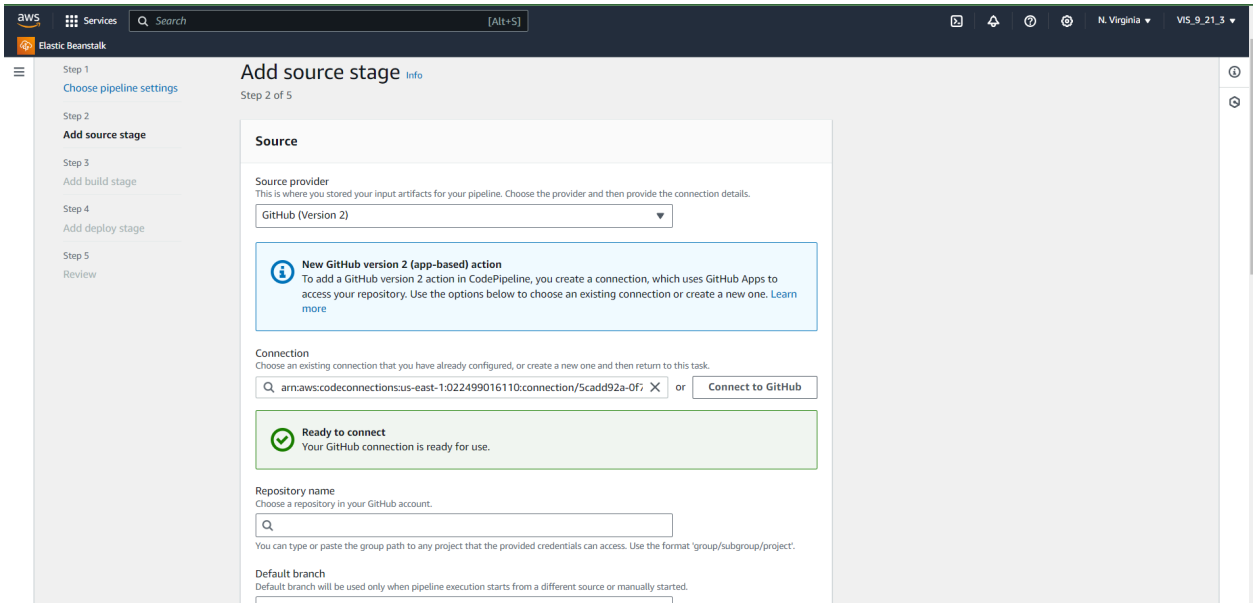


Name:Shejal Nilesh Tiwari Div/Roll No: D15C/57

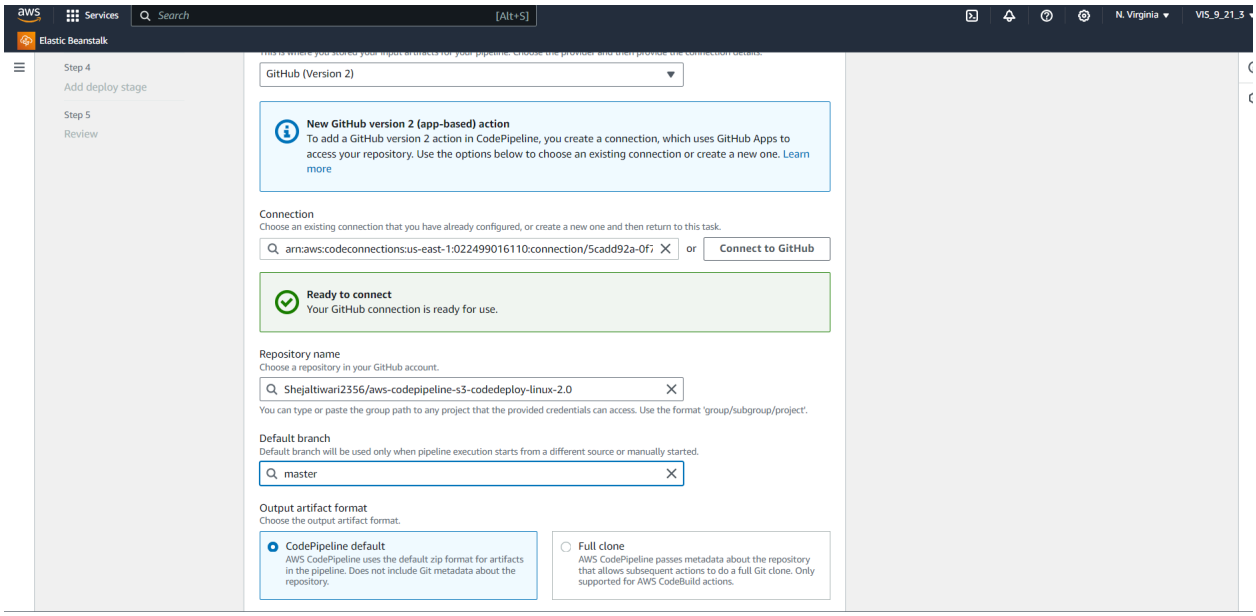
9.Create a connection in your github give connection name.



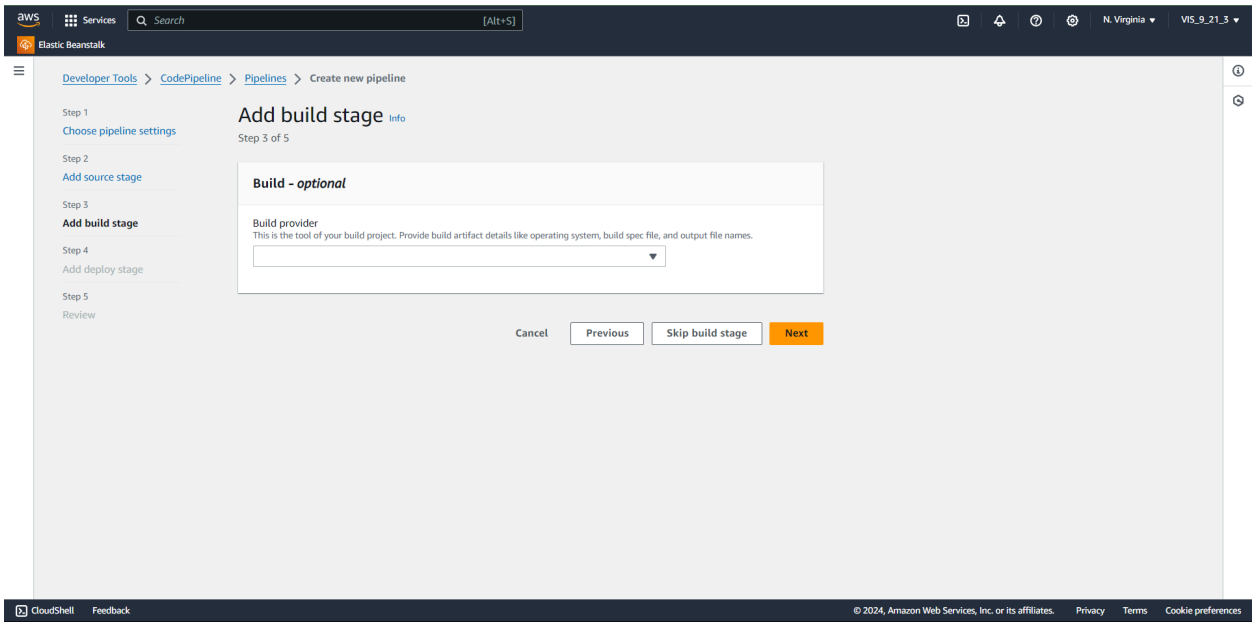
10.select version 2 for your githhub.



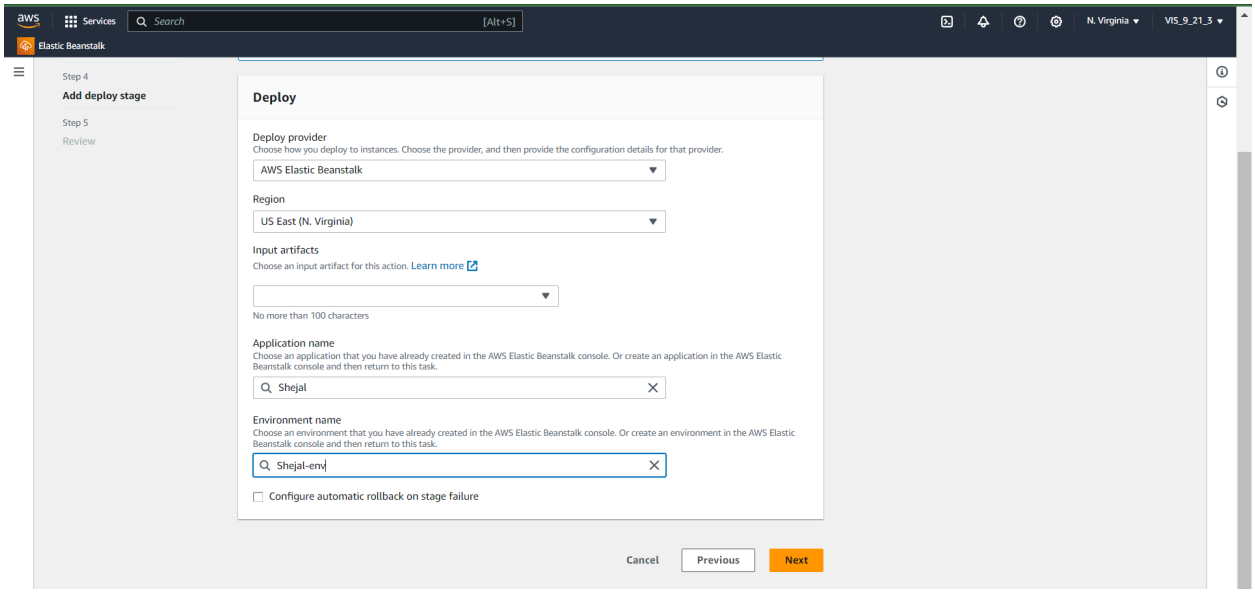
11. Select branch for the repository here we have selected master branch.



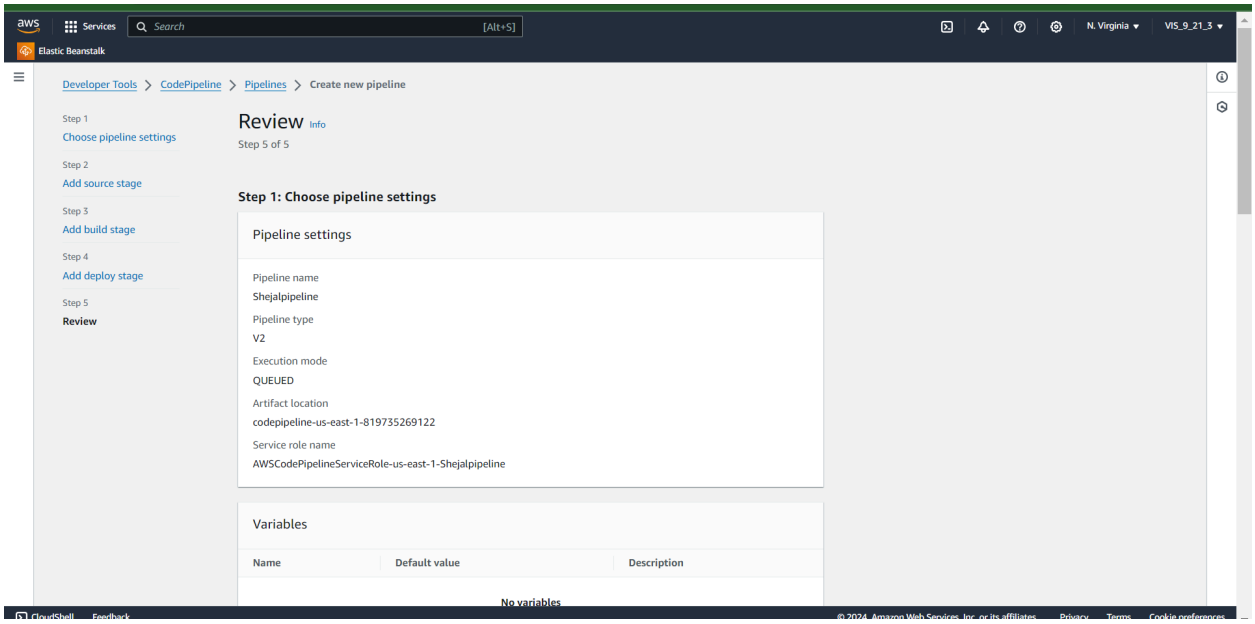
12. You can skip build stage.



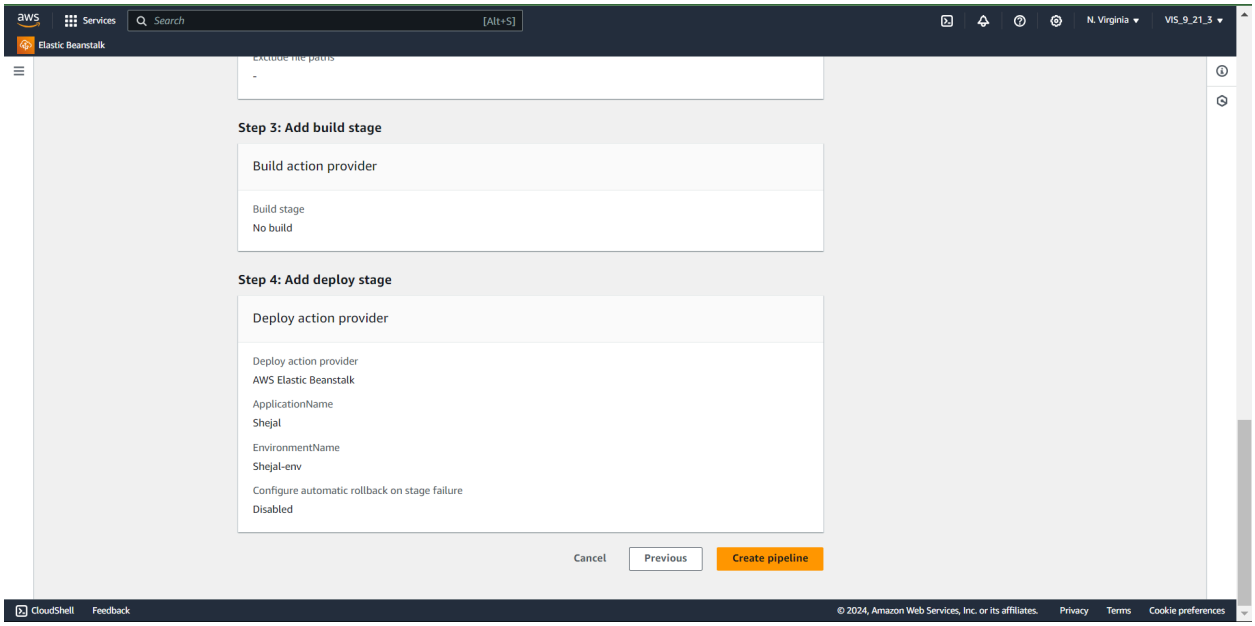
13. To Deploy select on next.



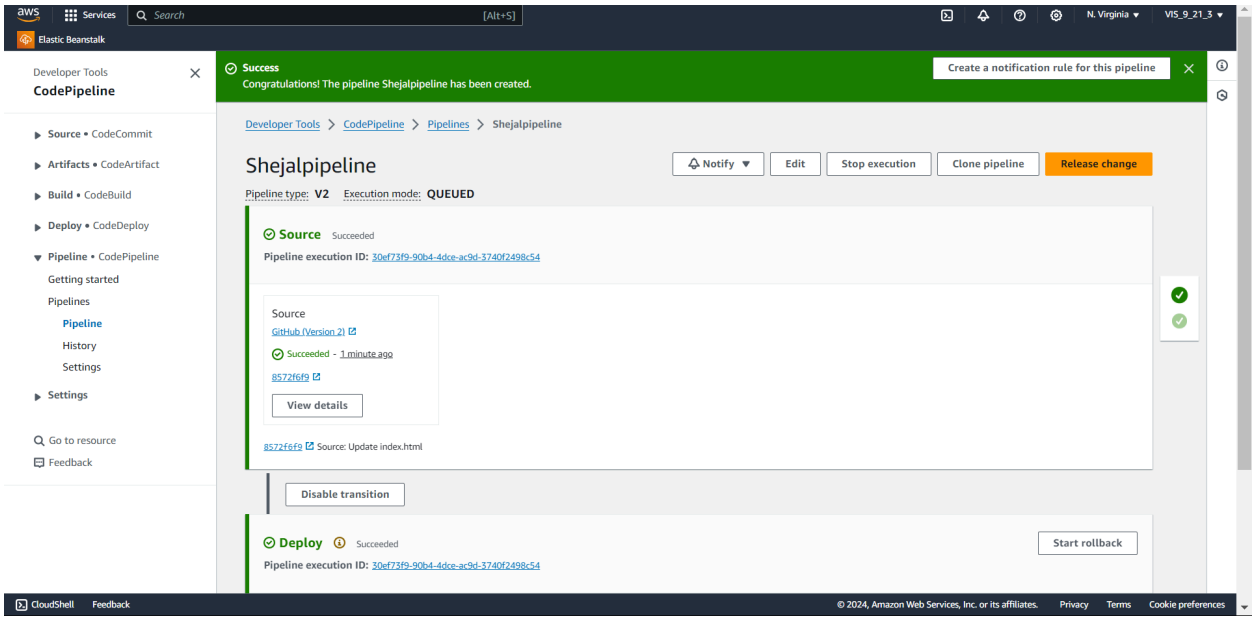
14. Review all the settings created.



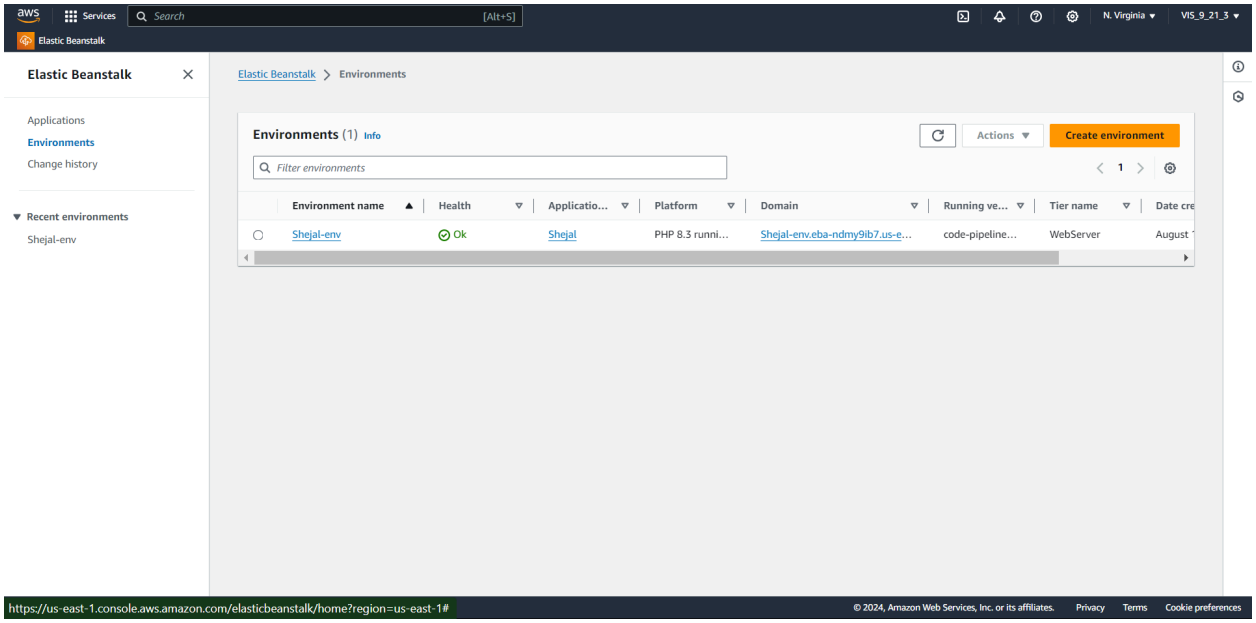
Name:Shejal Nilesh Tiwari Div/Roll No: D15C/57



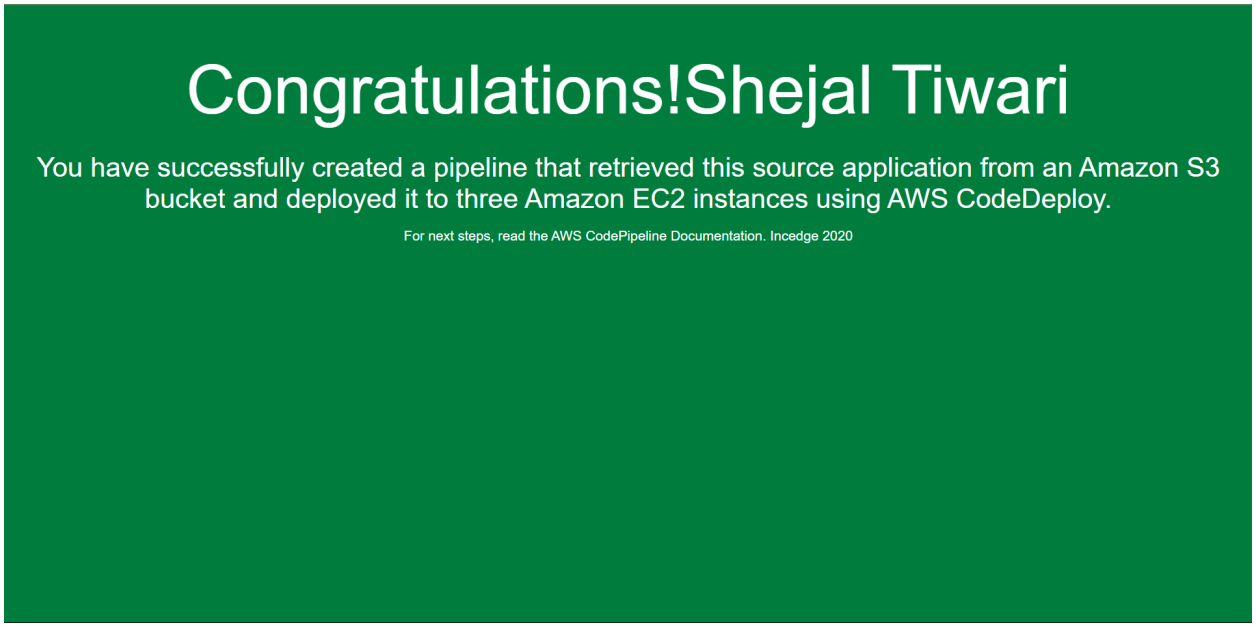
15.Wait till the pipeline gets successfully deployed.



16.Go back to your environment and click on the domain to see the deployed github repository



17. To see the proper working of pipeline. Go to github make the changes in the code commit it. Here I have added the my name in the index.html file. The changes can be seen instantly.



Name: Shejal Nilesh Tiwari Div/Roll no: D15C/57

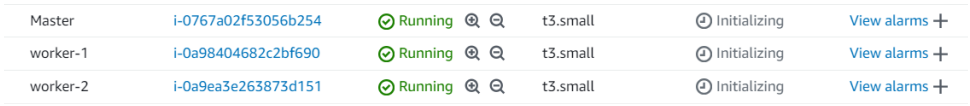
EXPERIMENT NO: 3

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

STEPS :

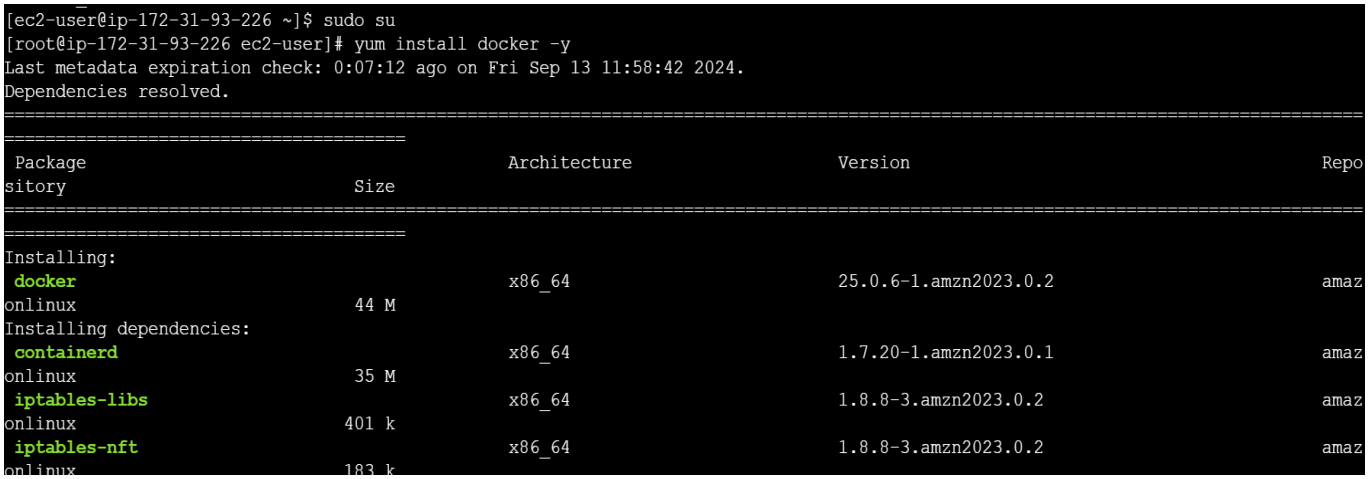
1.**Create Instances:**

We initiated the creation of three virtual machines or instances, naming them Master, Worker1, and Worker2. These instances will act as the nodes in our Kubernetes cluster.



Master	i-0767a02f53056b254	Running	t3.small	Initializing	View alarms +
worker-1	i-0a98404682c2bf690	Running	t3.small	Initializing	View alarms +
worker-2	i-0a9ea3e263873d151	Running	t3.small	Initializing	View alarms +

2.**Install Docker**:
On each instance, execute the following commands to gain superuser privileges and install Docker:
sudo su
yum install docker -y
This step ensures that Docker, a container runtime, is installed on Master, Worker1, and Worker2. Docker is essential for running the containerized applications that Kubernetes will orchestrate.



```
[ec2-user@ip-172-31-93-226 ~]$ sudo su
[root@ip-172-31-93-226 ec2-user]# yum install docker -y
Last metadata expiration check: 0:07:12 ago on Fri Sep 13 11:58:42 2024.
Dependencies resolved.
================================================================================
 Package                 Architecture            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
```

3.Start Docker:

Start the Docker service on each instance to ensure it is running:

```
systemctl start docker
```

- This command activates the Docker service on Master, Worker1, and Worker2, enabling them to run and manage containers.

```
[root@ip-172-31-93-226 ec2-user]# systemctl start docker
[root@ip-172-31-93-226 ec2-user]#
```

4.**Install kubeadm:**
Prepare the system for Kubernetes installation by configuring SELinux and adding the Kubernetes repository:

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

Install Kubernetes components:

```
sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
sudo systemctl enable --now kubelet
```

These steps configure SELinux to run in permissive mode, add the Kubernetes repository, and install Kubernetes tools (kubelet, kubeadm, and kubectl) on Master, Worker1, and Worker2. The kubelet service is also enabled and started to ensure it runs on boot.

```
Installed:
  conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64              cri-tools-1.31.1-150500.1.1.x86_64                      kubeadm-1.31.1-15
0500.1.1.x86_64
  kubectl-1.31.1-150500.1.1.x86_64                         kubelet-1.31.1-150500.1.1.x86_64                        kubernetes-cni-1.
5.1-150500.1.1.x86_64
  libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64       libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64     libnetfilter_queu
e-1.0.5-2.amzn2023.0.2.x86_64

Complete!
[root@ip-172-31-84-46 ec2-user]# sudo systemctl enable --now kubelet
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service → /usr/lib/systemd/system/kubelet.service.
```

5.Confirm Repository:

Verify that the Kubernetes repository is correctly configured by listing all enabled repositories:
bash
Copy code
yum repo list
Execute this command on Master, Worker1, and Worker2 to confirm that the Kubernetes repository has been added and is available for package installations.

```
[root@ip-172-31-84-46 ec2-user]# yum repolist
repo id                                                    repo name
amazonlinux                                                Amazon Linux 2023 repository
kernel-livepatch                                           Amazon Linux 2023 Kernel Livepatch repository
kubernetes                                                 Kubernetes
[root@ip-172-31-84-46 ec2-user]#
```

6.Initialize kubeadm on Master Node:

On the Master node, initialize the Kubernetes cluster using kubeadm. This process sets up the Kubernetes control plane and generates commands for joining worker nodes:
Copy the commands displayed in the output of the initialization process to configure permissions and obtain the join token. This includes a join command link needed for worker nodes to connect to the master.

```
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 172.31.84.46:6443 --token jl06ac.t7cdzxf0x5eddmsl \
        --discovery-token-ca-cert-hash sha256:4a152b913ee4b60dc2126d55f631b86d0dafb7d58132416c4f32f0668ac553be
[root@ip-172-31-84-46 ec2-user]#
```

In the screenshot you can see the commands written in the 3rd 4th and 5th line
Copy that command , this command is used to add right permission to the user
Also copy the 7th line , here it is the credential for the user .
Also copy the last 2 lines it is a link used to join the nodes

7.Join Worker Nodes:

On Worker1 and Worker2, use the join command provided by the Master node to connect them to the Kubernetes cluster:

kubeadm join 172.31.84.46:6443 --token jl06ac.t7cdzxf0x5eddmsl \
    --discovery-token-ca-cert-hash
sha256:4a152b913ee4b60dc2126d55f631b86d0dafb7d58132416c4f32f0668ac553be
This command allows Worker1 and Worker2 to register themselves with the Master node and become part of the Kubernetes cluster.
Verify Nodes:

Name: Shejal Nilesh Tiwari Div/Roll no: D15C/57

On the Master node, check the status of all nodes in the cluster to ensure that Worker1 and Worker2 have successfully joined:

kubectl get nodes

This command lists all nodes and their status within the Kubernetes cluster, helping to confirm successful node registration and connectivity.

```
[root@ip-172-31-84-46 ec2-user]# kubectl get node
NAME                           STATUS     ROLES           AGE   VERSION
ip-172-31-84-46.ec2.internal   NotReady   control-plane   56s   v1.31.1
[root@ip-172-31-84-46 ec2-user]# kubectl get node
The connection to the server 172.31.84.46:6443 was refused - did you specify the right host or port?
[root@ip-172-31-84-46 ec2-user]# kubectl get node
NAME                           STATUS     ROLES           AGE    VERSION
ip-172-31-84-46.ec2.internal   NotReady   control-plane   5m1s   v1.31.1
[root@ip-172-31-84-46 ec2-user]#
```

And this was the output when i tried connecting it with worker 1 and worker 2

```
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Fri Sep 13 12:04:33 2024 from 18.206.107.28
[ec2-user@ip-172-31-87-149 ~]$ sudo su
[root@ip-172-31-87-149 ec2-user]# kubeadm join 172.31.84.46:6443 --token qq448c.8c0zquwywz3eqt0b        --discovery-token-ca-cert-has
h sha256:49cc770e646aab704883a3d9fb30e6146fdd83e782ebd5ec3194dacee59f2257
[preflight] Running pre-flight checks
        [WARNING FileExisting-socat]: socat not found in system path
        [WARNING FileExisting-tc]: tc not found in system path
```

So there was a failure in connecting the worker 1 and 2 with the master even if i provided the joining link.
The Cloud shell didn't proceeds further.

CONCLUSION :
We created and configured three instances (Master, Worker1, and Worker2), installed Docker and Kubernetes on all nodes, and initialized the Kubernetes control plane on the Master node. Despite providing the join command to Worker1 and Worker2, they failed to connect to the Master node.

Connection Refused Errors: The Kubernetes components are failing to connect to the API server at `https://172.31.84.46:6443`, resulting in "connection refused" errors.
No Running Containers: The `docker ps -a` command shows no containers, which means the Kubernetes components are not running in Docker.

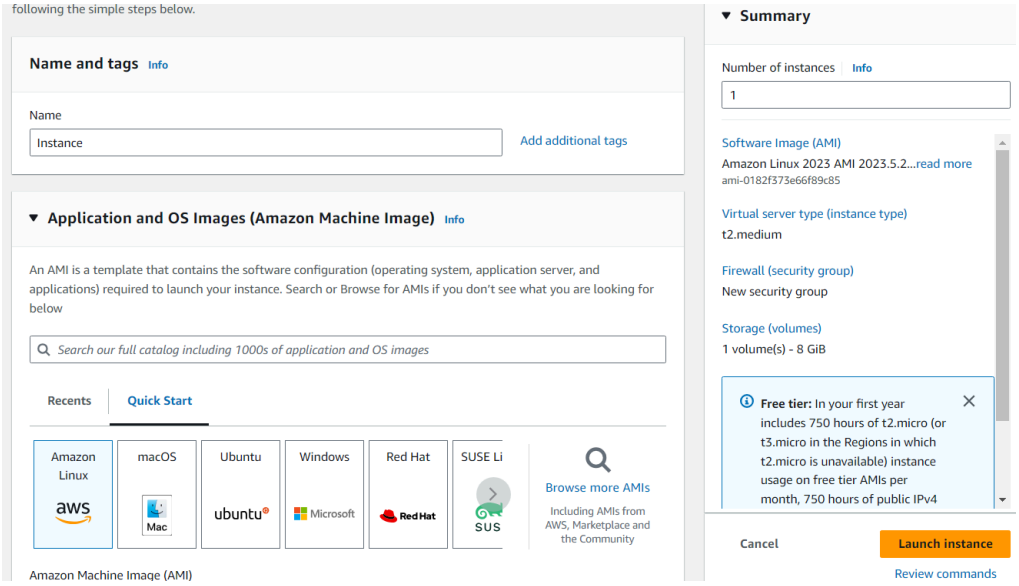
CrashLoopBackOff: There are errors indicating that the containers for Kubernetes components are restarting repeatedly but failing to start properly.
Network Plugin Not Ready: The error message "Network plugin returns error: cni plugin not initialized" suggests issues with the network plugin required for Kubernetes.

Name : Shejal Tiwari Div/Roll No: D15C/ 57

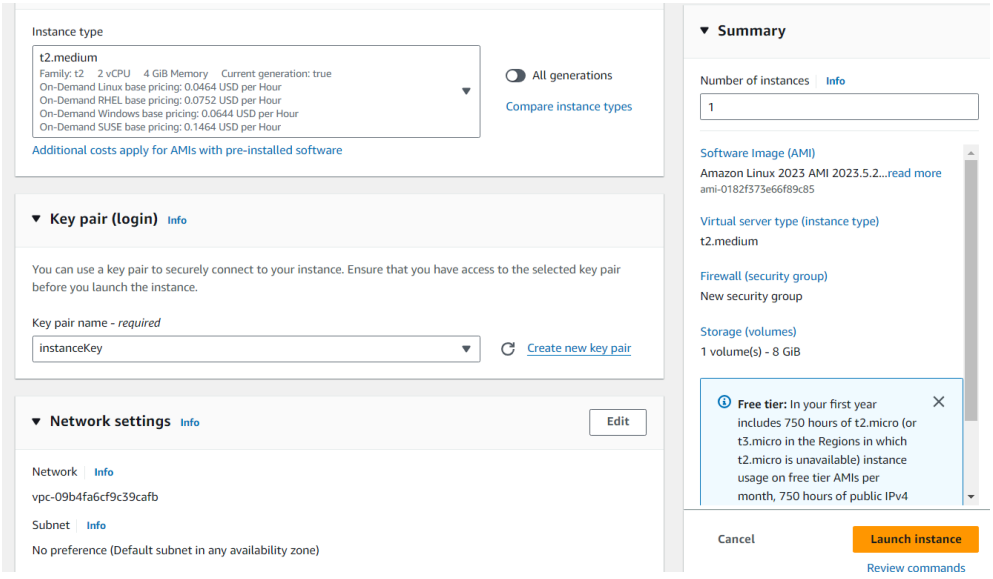
Experiment 4

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

Step 1: Set Up EC2 Instances.Select Amazon Linux with t2.medium and right ssh network configurations in the inbound rules.



2) Select a key pair, to create one. Click on create Also check if the pem file got downloaded in your pc.

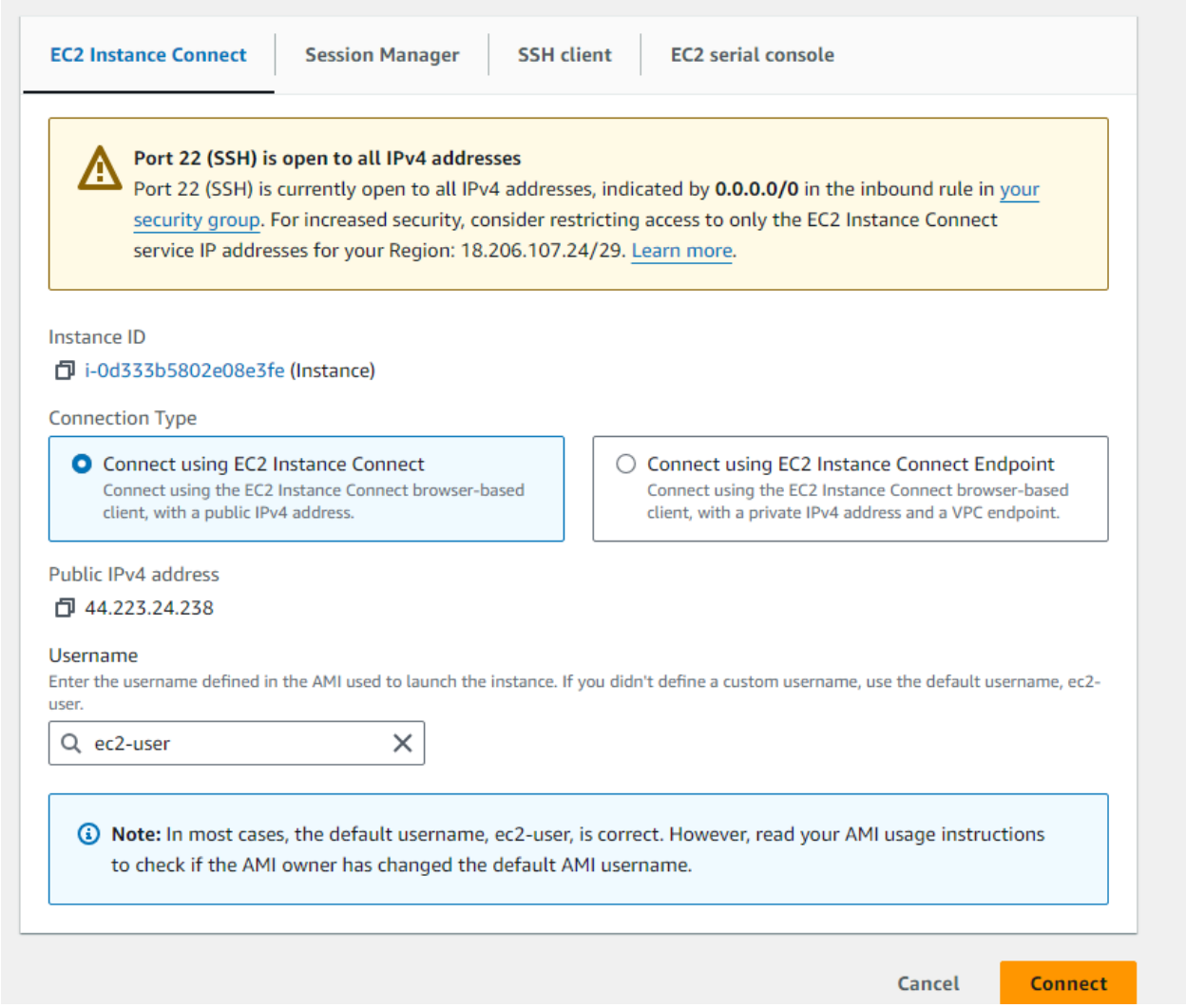


When you are going to perform the command in your cmd/git bash

Name : Shejal Tiwari Div/Roll No: D15C/ 57

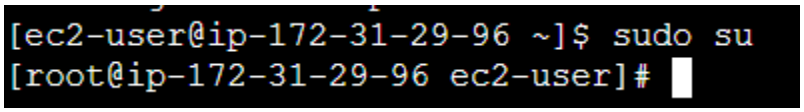
You need to write this command.
ssh -i <keyname> ec2-user@<public id address of your ec2 instances> after launching the instance.

3) Once created, go back to the instances page. Click on the instance id. Then, click on connect.



Step 2: Installation of Docker

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



Name : Shejal Tiwari Div/Roll No: D15C/ 57

2) We can install docker using yum. Use the command ‘yum install docker-y’

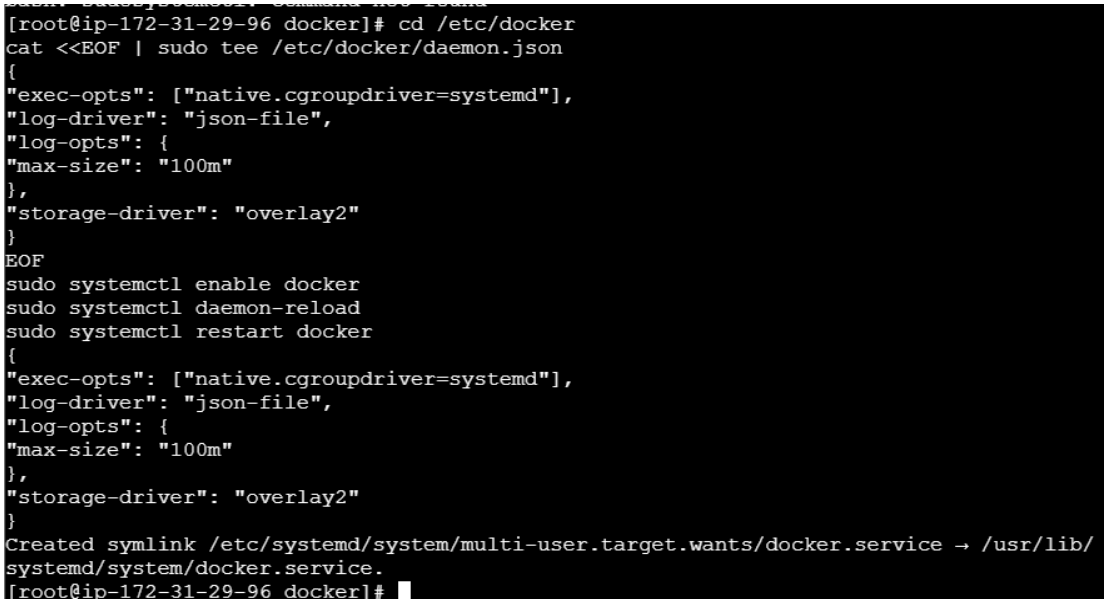
```
[root@ip-172-31-29-96 ec2-user]# yum install docker -y
Last metadata expiration check: 0:04:49 ago on Sun Sep 15 10:36:23 2024.
Dependencies resolved.
================================================================================
 Package                 Architecture            Version
================================================================================
Installing:
 docker                  x86_64                  25.0.6-1.amzn2023.0.2
Installing dependencies:
 containerd              x86_64                  1.7.20-1.amzn2023.0.1
 iptables-libs           x86_64                  1.8.8-3.amzn2023.0.2
 iptables-nft            x86_64                  1.8.8-3.amzn2023.0.2
 libcgroup               x86_64                  3.0-1.amzn2023.0.1
 libnetfilter_conntrack  x86_64                  1.0.8-2.amzn2023.0.2
 libnfnetlink            x86_64                  1.0.1-19.amzn2023.0.2
 libnftnl                x86_64                  1.2.2-2.amzn2023.0.2
 pigz                    x86_64                  2.5-1.amzn2023.0.3
 runc                    x86_64                  1.1.13-1.amzn2023.0.1

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

Total download size: 84 M
Installed size: 317 M
Downloading Packages:
                                                [===          (1/10): containerd
```

```
===  Installing       : docker-25.0.6-1.amzn2023.0.2.x86_64 [=====================================  Installin
                                                                          10/10
  Running scriptlet: docker-25.0.6-1.amzn2023.0.2.x86_64
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
  libcgroup-3.0-1.amzn2023.0.1.x86_64       libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64  libnfnetlink-1.
  pigz-2.5-1.amzn2023.0.3.x86_64            runc-1.1.13-1.amzn2023.0.1.x86_64

Complete!
[root@ip-172-31-29-96 ec2-user]#
```

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

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

```
}, "
storage-driver": "overlay2"
}
EOF
sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker
```

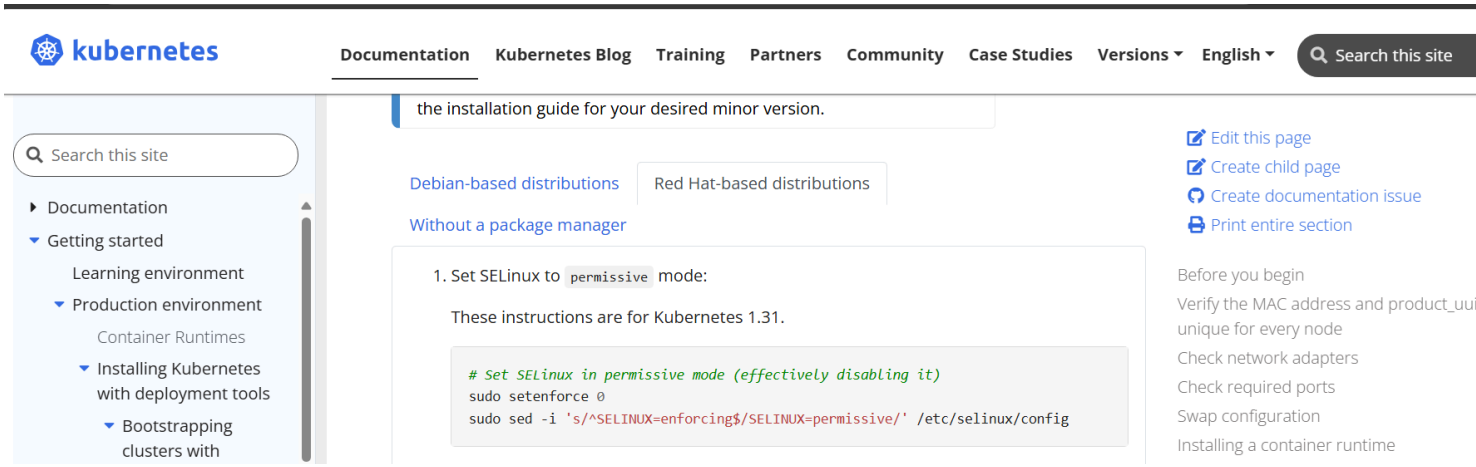


Step 3: Installing Kubernetes

1) For installing kubernetes, we will be using kubeadm, a framework used for creating kubernetes clusters using command line.

https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/

Name : Shejal Tiwari Div/Roll No: D15C/ 57



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

sudo setenforce 0
→sets SELinux to permissive mode

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

Run the following commands:

- sudo setenforce 0
- sudo sed-i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config

```
[root@ip-172-31-29-96 docker]# sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
```

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

Name : Shejal Tiwari Div/Roll No: D15C/ 57

```
[root@ip-172-31-29-96 docker]# cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
[root@ip-172-31-29-96 docker]#
```

- yum repolist

This command shows the repositories created on the machine.

```
[root@ip-172-31-29-96 docker]# yum repolist
repo id                                                                repo name
amazonlinux                                                            Amazon Linux 2023 repository
kernel-livepatch                                                       Amazon Linux 2023 Kernel Livepatch repository
kubernetes                                                             Kubernetes
[root@ip-172-31-29-96 docker]#
```

Next step is to install kubelet, kubeadm, kubectl

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

```
================================  Installing       : kubectl-1.31.1-150500.1.1.x86_64
                                9/9
  Running scriptlet: kubectl-1.31.1-150500.1.1.x86_64
  Verifying        : conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64
  Verifying        : libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64
  Verifying        : libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64
  Verifying        : libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64
  Verifying        : cri-tools-1.31.1-150500.1.1.x86_64
  Verifying        : kubeadm-1.31.1-150500.1.1.x86_64
  Verifying        : kubectl-1.31.1-150500.1.1.x86_64
  Verifying        : kubelet-1.31.1-150500.1.1.x86_64
  Verifying        : kubernetes-cni-1.5.1-150500.1.1.x86_64

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

Complete!
[root@ip-172-31-29-96 docker]#
```

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

- sudo systemctl enable --now kubelet

```
[root@ip-172-31-29-96 docker]# sudo systemctl enable --now kubelet
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service → /usr/lib/systemd/syste
m/kubelet.service.
[root@ip-172-31-29-96 docker]#
```

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

```
[root@ip-172-31-29-96 docker]# sudo swapoff -a
echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf
sudo sysctl -p
net.bridge.bridge-nf-call-iptables=1
net.bridge.bridge-nf-call-iptables = 1
[root@ip-172-31-29-96 docker]#
```

3) Firstly, we need toinitialize kubernetes.For This,run the command:

- sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errors=NumCPU,Mem

```
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 172.31.29.96:6443 --token 0b6cct.1cm4p25mefy05fhl \
        --discovery-token-ca-cert-hash sha256:ae83caa940837900b62231f4f381a06d69b4d25b0207ce5fff9a943e6757b6a8
[root@ip-172-31-29-96 docker]#
```

4) From The Output,we receive the following commands:

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

Run These Commands

```
[root@ip-172-31-29-96 docker]# mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config
[root@ip-172-31-29-96 docker]#
```

5) Add a common networking plugin flannel usingthiscommand

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

Name : Shejal Tiwari Div/Roll No: D15C/ 57

```
[root@ip-172-31-29-96 docker]# kubectl apply -f https://raw.githubusercontent.com/coreos/flan
nel/master/Documentation/kube-fkubectl apply -f https://raw.githubusercontent.com/coreos/flan
nel/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
[root@ip-172-31-29-96 docker]#
```

Step3:Deploy nginx server

1) Now that the cluster is set,applythedeployment fileofnginxusingthiscommand

- kubectl apply-fhttps://k8s.io/examples/pods/simple-pod.yaml

```
[root@ip-172-31-29-96 docker]# kubectl apply -f https://k8s.io/examples/pods/simple-pod.yaml
pod/nginx created
[root@ip-172-31-29-96 docker]#
```

2) Use The Command

- kubectl get pods

To Get the list of pods in cluster.

```
[root@ip-172-31-29-96 docker]# kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
nginx   0/1     Pending   0          44s
[root@ip-172-31-29-96 docker]#
```

This output shows that thepodisina‘PENDING’state, change it to RUNNING state,runtuh following commands.

- kubectl describe pod nginx:Provides Details About Your Pod

This command is used to get details about the pod and potential issues with the pod

```
[root@ip-172-31-29-96 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>
```

```
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason            Age    From               Message
  ----     ------            ----   ----               -------
  Warning  FailedScheduling  2m43s  default-scheduler  0/1 nodes are available: 1 node(s) had
 untolerated taint {node-role.kubernetes.io/control-plane: }. preemption: 0/1 nodes are avail
able: 1 Preemption is not helpful for scheduling.
  Warning  FailedScheduling  1s     default-scheduler  0/1 nodes are available: 1 node(s) had
 untolerated taint {node-role.kubernetes.io/control-plane: }. preemption: 0/1 nodes are avail
able: 1 Preemption is not helpful for scheduling.
[root@ip-172-31-29-96 docker]#
```

3) From this output, we get to know that the node has some untolerated taint. To remove this, use

- kubectl taintnodes --allnode-role.kubernetes.io/control-plane:NoSchedule-

```
[root@ip-172-31-29-96 docker]# kubectl taint nodes --all node-role.kubernetes.io/control-plane:NoSche
dule-                        kubectl taint nodes --all node-role.kubernetes.io/control-plane:NoSche
dule-
node/ip-172-31-29-96.ec2.internal untainted
```

4) Now, we check the status of the pod by running “kubectl get pods” again

```
[root@ip-172-31-29-96 docker]# kubectl get pods
NAME    READY   STATUS    RESTARTS      AGE
nginx   1/1     Running   2 (68s ago)   9m54s
[root@ip-172-31-29-96 docker]#
```

5) Now, change the port to which you want to host your server on using command

- kubectl port-forward nginx <port number you want to host on>:80

```
[root@ip-172-31-29-96 docker]# kubectl port-forward pod/nginx 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
```

6) To check whether the deployment was successful, run the command

- curl--head http://127.0.0.1:<port number given by you>

If the terminal returns a status code of 200, it means that the deployment is successful.

Conclusion:

Few Difficulties faced during the practical:

1. EC2 Instance Connection Issue: After connecting the instances it shows connection failure so with the notification in red at the top for bash so in that case try another instances if still it now works go to your git bash and connection for ssh connection with keyname and public ip address of the instance. And then proceed with the further installation of docker and kubernetes.

2. Kubernetes Installation Issue: Some time the commands for setting up your kubernetes repository doesnt work and goes in failure in that case try reconnecting yours instance and start again with all the commands.

3. Nginx Deployment Issues: The Nginx server might not start because of network problems in Kubernetes or restrictions on the control plane that stop the pod from running therefore in this case you need to re-run the commands again and again if it shows that “have you connected to the right host ?”

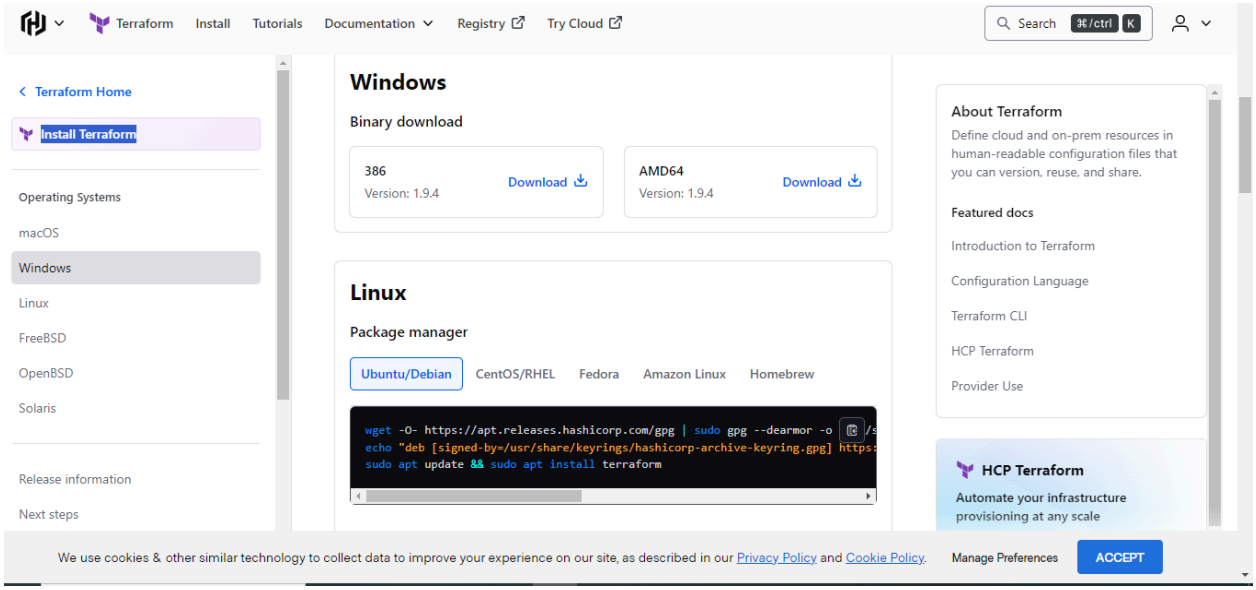
NAME:SHEJAL TIWARI DIV/ROLL NO: D15C/57

EXPERIMENT NO. 05

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

1.Download and Install Terraform:

- If you haven't already, download Terraform from the official website.
- Install Terraform by extracting the downloaded zip file to a directory of your choice. Note the path where Terraform executable (`terraform.exe` on Windows) is located.



2.Find the Terraform Executable Path:

- After installation, locate the directory where Terraform executable (`terraform.exe`) is present. For example, it could be in `C:\terraform` or any other path you chose during installation.

3.Set Environment Variables:

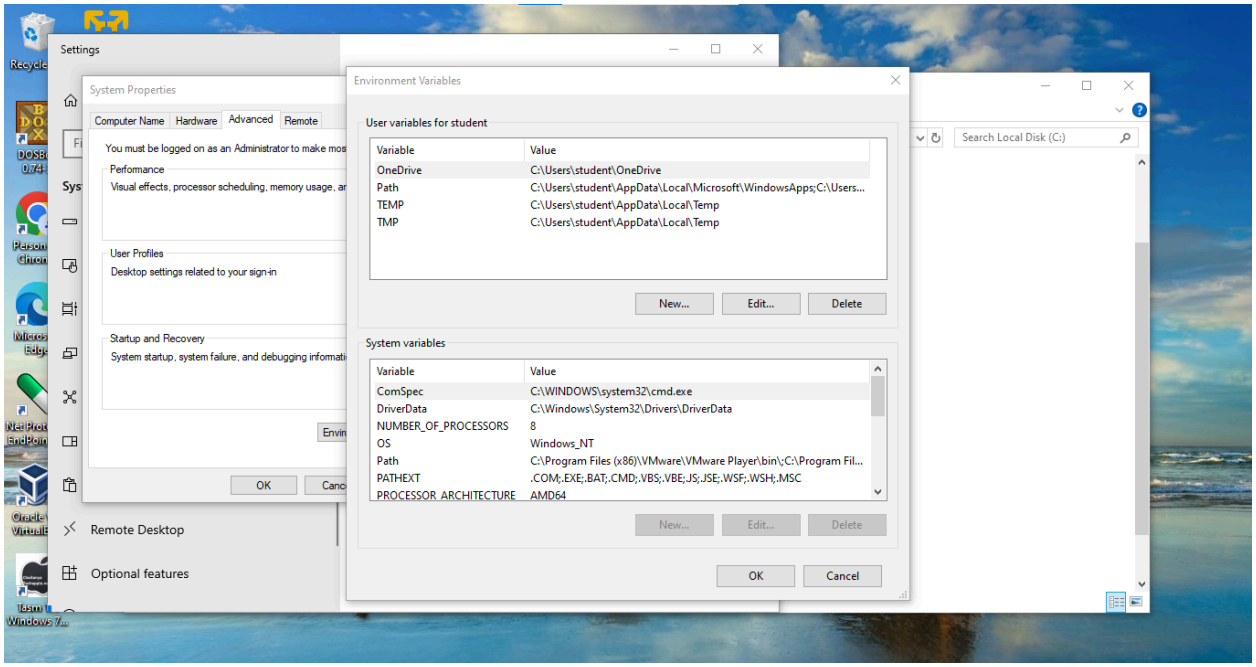
- Right-click on **This PC** or **Computer** on your desktop or in File Explorer, then click **Properties**.
- In the System window, click on **Advanced system settings** on the left-hand side.
- In the System Properties window, click on the **Environment Variables** button.

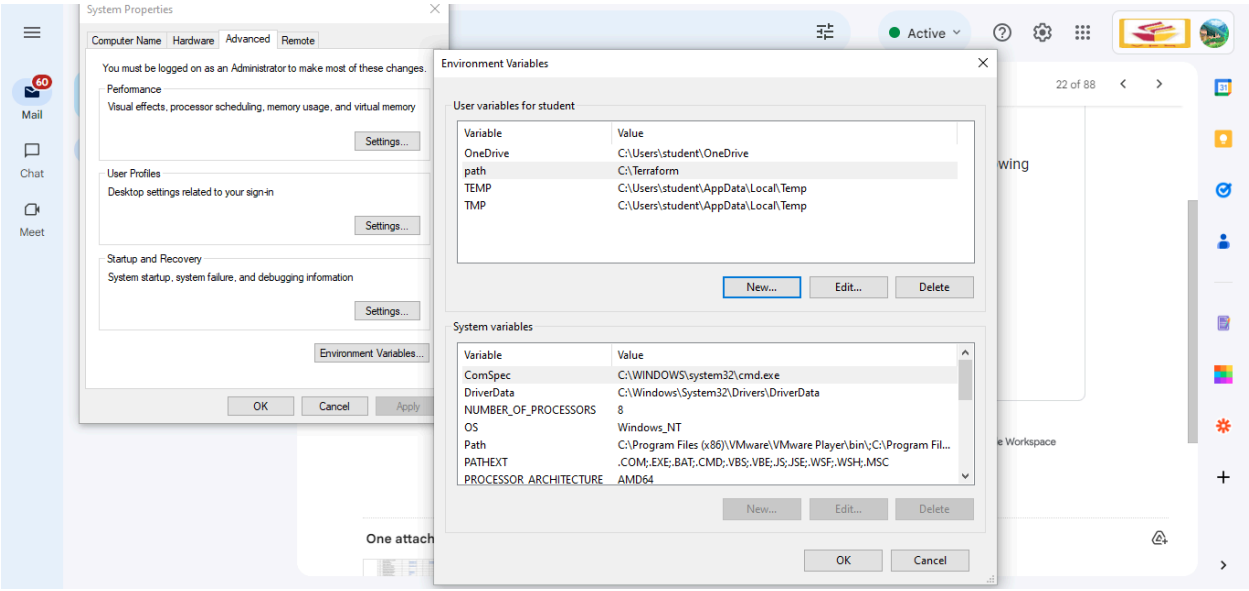
4.Edit System Environment Variables:

- In the Environment Variables window (System Properties), under the section **System Variables**, find and select the variable named **Path**, then click **Edit**.

5.Add Terraform to Path:

- In the Edit Environment Variable window, click **New** to add a new entry.
- Enter the path to the directory where Terraform executable (`terraform.exe`) is located. For example, if Terraform is installed in `C:\terraform`, add `C:\terraform` to the list.





6.Verify Installation:

- Open a new command prompt (to ensure it picks up the updated environment variables).
- Type `terraform --version` and press **Enter**. You should see the Terraform version information printed on the screen if the path configuration was successful.

If PowerShell is unable to recognize the `terraform` command, even though you've installed Terraform. This issue typically occurs when the directory containing the Terraform executable is not included in the `PATH` environment variable that PowerShell is using.

To resolve this issue, you need to add the directory containing the Terraform executable to your PowerShell session's `PATH` variable.

Set the PATH Variable Temporarily for the Current Session:

- In PowerShell, you can set the `PATH` variable for the current session using the following command:
  powershell
  Copy code
  ```
  $env:PATH += ";C:\Terraform"
  ```

NAME:SHEJAL TIWARI DIV/ROLL NO: D15C/57

```
PS C:\Users\student> $env:PATH += ";C:\Terraform"
>>
PS C:\Users\student> terraform --version
Terraform v1.9.4
on windows_amd64
PS C:\Users\student>
```

Name: Shejal Tiwari Div/Roll no: D15C/57

AdvanceDevops Experiment: 6

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

To Create docker image using terraform

We need Download and Install Docker Desktop from https://www.docker.com/

Step 1: Check the docker functionality

```
PS C:\Users\INFT505-16> docker

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

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

```
PS C:\Users\INFT505-16> docker --version
Docker version 24.0.6, build ed223bc
PS C:\Users\INFT505-16>
```

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. Copy the Script into it. **Script:**

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

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

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

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

Step 3: Execute Terraform Init command to initialize the resources

```
C:\Users\INFT505-16>cd desktop\TerraformScripts\Docker

C:\Users\INFT505-16\Desktop\TerraformScripts\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 apply to apply the configuration, which will automatically create and run the Ubuntu Linux container based on our configuration. Using command : “terraform apply”

```
C:\Users\INFT505-16\Desktop\TerraformScripts\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          = [
          + "sleep",
          + "3600",
        ]
      + 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)
```

```
      + output      = (known after apply)
      + repo_digest = (known after apply)
    }

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

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

  Enter a value: yes

docker_image.ubuntu: Creating...
docker_image.ubuntu: Still creating... [10s elapsed]
docker_image.ubuntu: Still creating... [20s elapsed]
docker_image.ubuntu: Still creating... [30s elapsed]
docker_image.ubuntu: Still creating... [40s elapsed]
docker_image.ubuntu: Still creating... [50s elapsed]
docker_image.ubuntu: Creation complete after 53s [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Creating...
docker_container.foo: Creation complete after 2s [id=353bd0cae537e335931797ed86d2b603c682520b71c2af7d5b72f3c09eed2b11]
```

Docker images, After Executing Apply step:

```
C:\Users\INFT505-16\Desktop\TerraformScripts\Docker>docker images
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
ubuntu       latest    edbfe74c41f8   3 weeks ago      78.1MB
sonarqube    latest    3183d6818c6e   10 months ago    716MB
```

Step 5: Execute Terraform destroy to delete the configuration, which will automatically delete the Ubuntu Container.

```
C:\Users\INFT505-16\Desktop\TerraformScripts\Docker>terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Refreshing state... [id=353bd0cae537e335931797ed86d2b603c682520b71c2af7d5b72f3c09eed2b11]

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
```

```
  # 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=353bd0cae537e335931797ed86d2b603c682520b71c2af7d5b72f3c09eed2b11]
docker_container.foo: Destruction complete after 0s
docker_image.ubuntu: Destroying... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_image.ubuntu: Destruction complete after 1s

Destroy complete! Resources: 2 destroyed.
```

Step 6: This command outputs the state or plan in a human-readable format, helping you review the details of your configuration..

```
C:\Users\INFT505-16\Desktop\TerraformScripts\Docker>terraform show
# docker_container.foo:
resource "docker_container" "foo" {
    attach            = false
    bridge            = null
    command           = [
        "sleep",
        "3600",
    ]
    cpu_set           = null
    cpu_shares        = 0
    domainname        = null
    entrypoint        = []
    env               = []
    gateway           = "172.17.0.1"
    hostname          = "353bd0cae537"
    id                = "353bd0cae537e335931797ed86d2b603c682520b71c2af7d5b72f3c09eed2b11"
    image             = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a"
    init              = false
    ip_address        = "172.17.0.2"
    ip_prefix_length  = 16
```

Step 7: This command generates a visual graph of your Terraform resources, which can help you understand the dependencies and relationships between them.

```
C:\Users\INFT505-16\Desktop\TerraformScripts\Docker>terraform graph
digraph G {
  rankdir = "RL";
  node [shape = rect, fontname = "sans-serif"];
  "docker_container.foo" [label="docker_container.foo"];
  "docker_image.ubuntu" [label="docker_image.ubuntu"];
  "docker_container.foo" -> "docker_image.ubuntu";
}
```

Step 8: This command lists all the resources tracked by the Terraform state, allowing you to see which resources have been created and are being managed by Terraform..

```
C:\Users\INFT505-16\Desktop\TerraformScripts\Docker>terraform state list
docker_container.foo
docker_image.ubuntu

C:\Users\INFT505-16\Desktop\TerraformScripts\Docker>
```

Name: Shejal Tiwari Div/Roll No:D15C/57

AdvanceDevops Experiment 7

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

Integrating Jenkins with SonarQube:

Windows installation
Step 1 Install JDK 1.8
Step 2 download and install jenkins
https://www.blazemeter.com/blog/how-to-install-jenkins-on-windows

Ubuntu installation

https://www.digitalocean.com/community/tutorials/how-to-install-java-with-apt on-ubuntu-20-04#installing-the-default-jre-jdk

Step 1 Install JDK 1.8
sudo apt-get install openjdk-8-jre

sudo apt install default-jre

https://www.digitalocean.com/community/tutorials/how-to-install-jenkins-on-ubuntu 20-04

Open SSH

Prerequisites:

- Jenkins installed
- Docker Installed (for SonarQube)

(sudo apt-get install docker-ce=5:20.10.15~3-0~ubuntu-jammy
docker-ce-cli=5:20.10.15~3-0~ubuntu-jammy containerd.io docker-compose-plugin)

- SonarQube Docker Image

Steps to integrate Jenkins with SonarQube

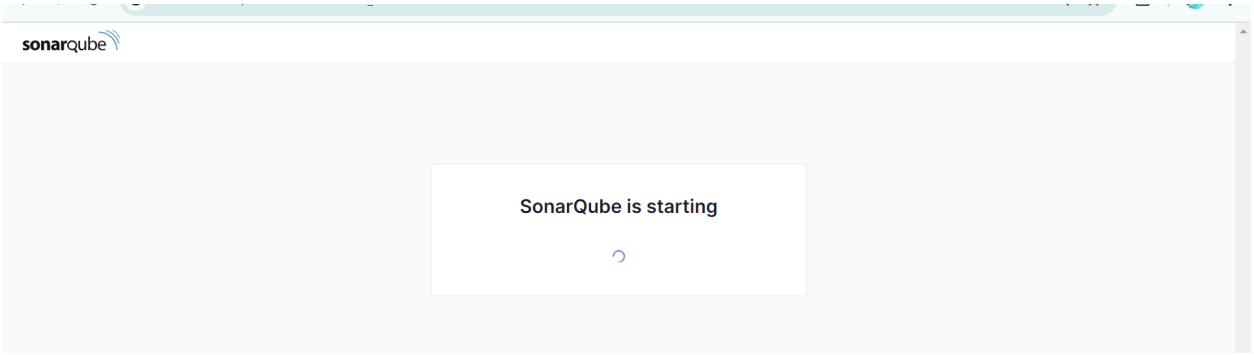
1. Open up Jenkins Dashboard on localhost, port 8080 or whichever port it is at for you.

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

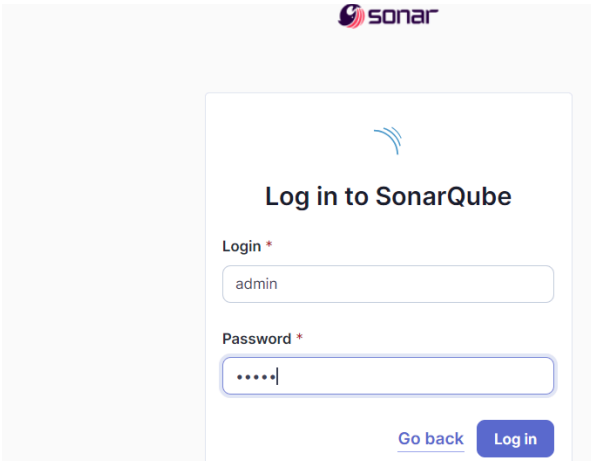
```
PS C:\Users\91900> docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:lates
t
8b2a833004ac39a9b009118bacac47e5808c9ec8df3f59f8657bd23fa23f48f2
```

Warning: run below command only once
docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest

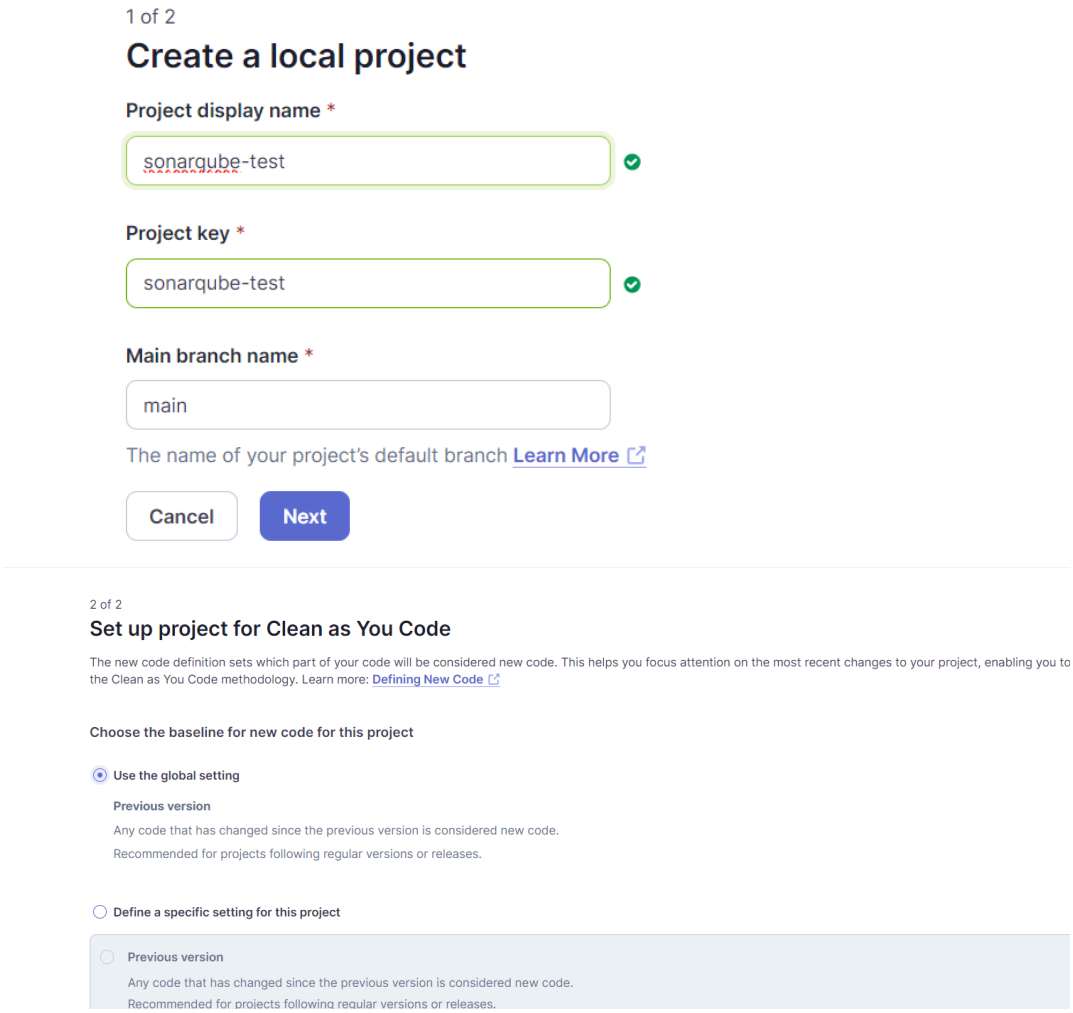
3. Once the container is up and running, you can check the status of SonarQube at localhost port 9000.



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



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

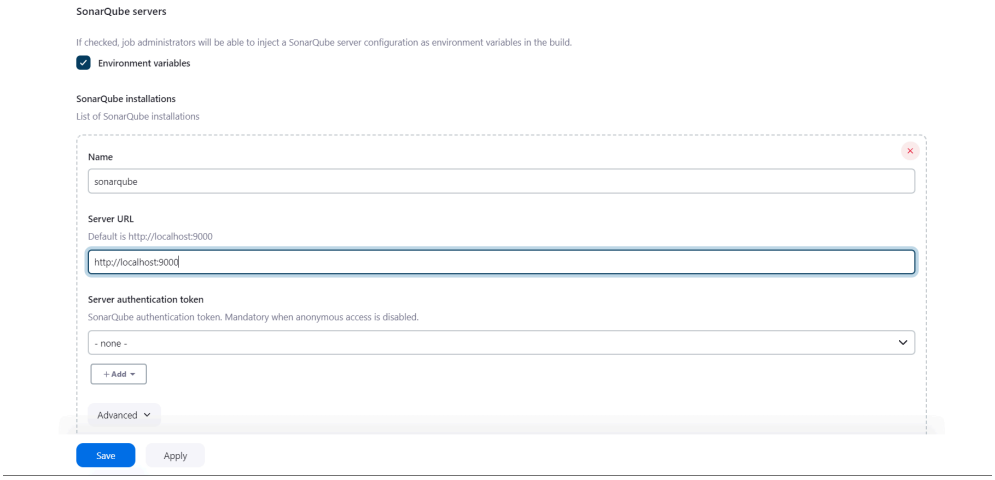


Setup the project and come back to Jenkins Dashboard.

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

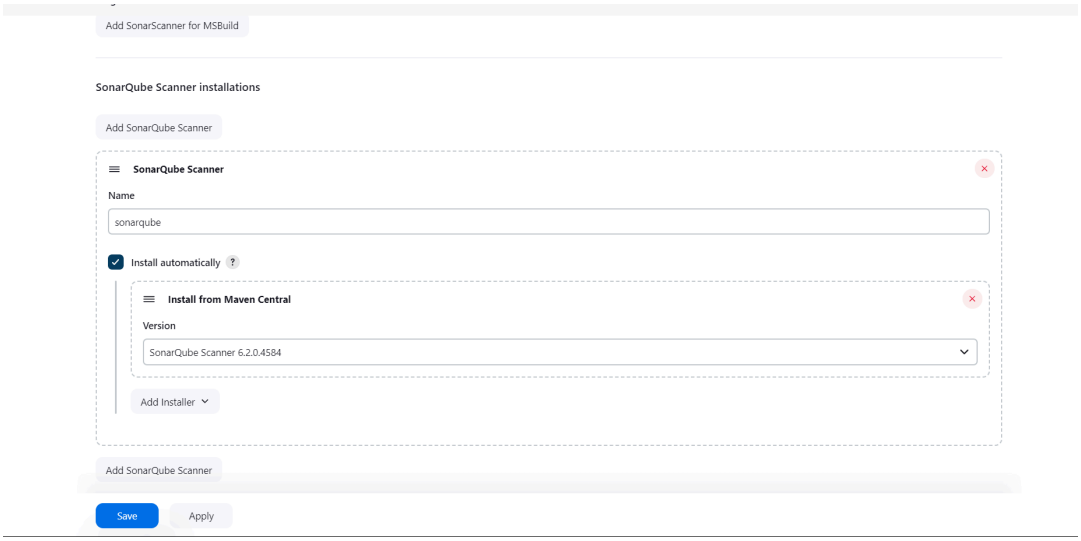
6. Under Jenkins 'Configure System', look for SonarQube Servers and enter the details.

Enter the Server Authentication token if needed.

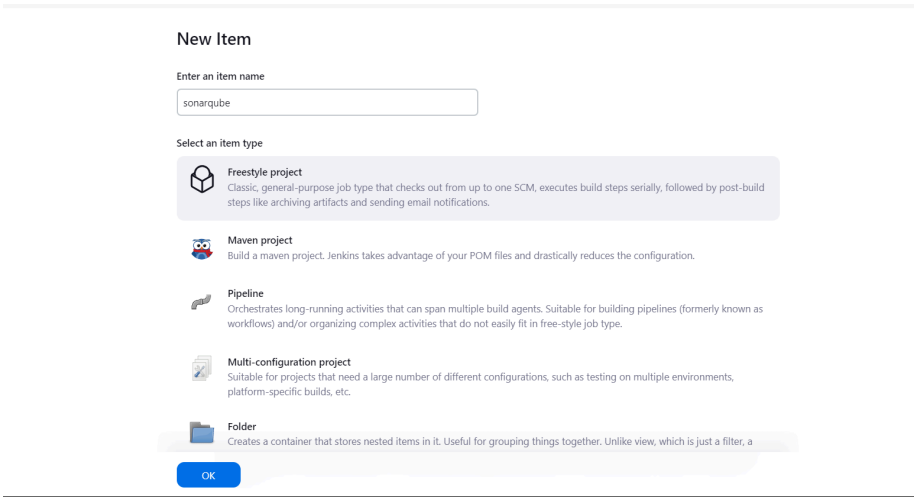


7. Search for SonarQube Scanner under Global Tool Configuration. Choose the latest configuration and choose Install automatically

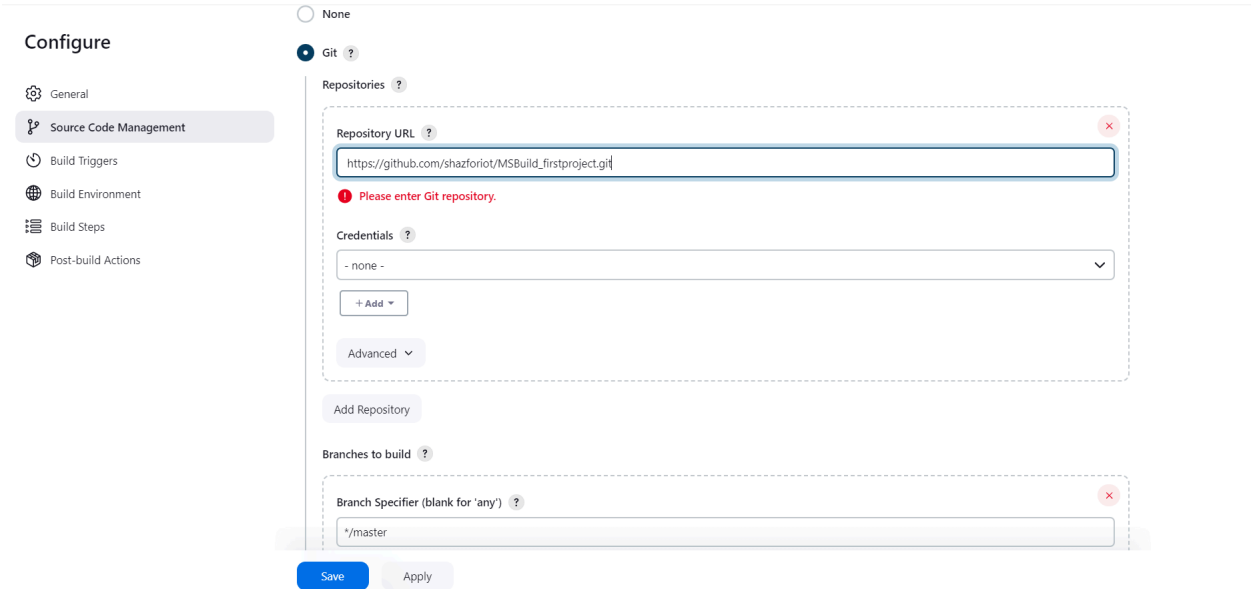
.



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

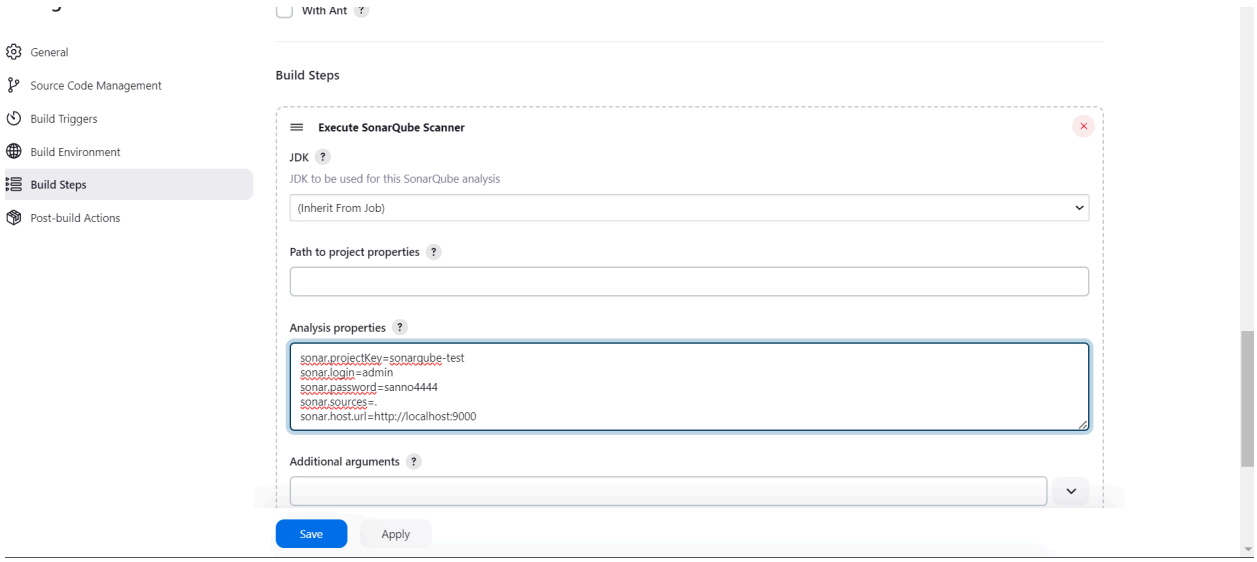


9. Choose this GitHub repository in Source Code Management.
   https://github.com/shazforiot/MSBuild_firstproject.git
   It is a sample hello-world project with no vulnerabilities and issues, just to test

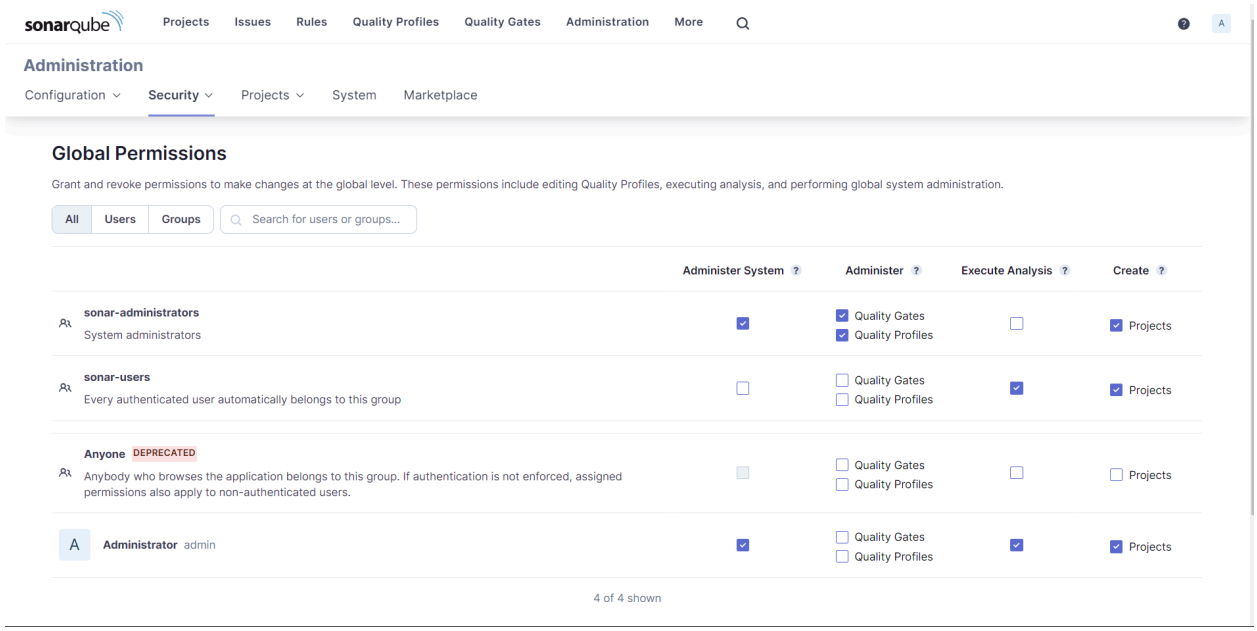


   the integration.

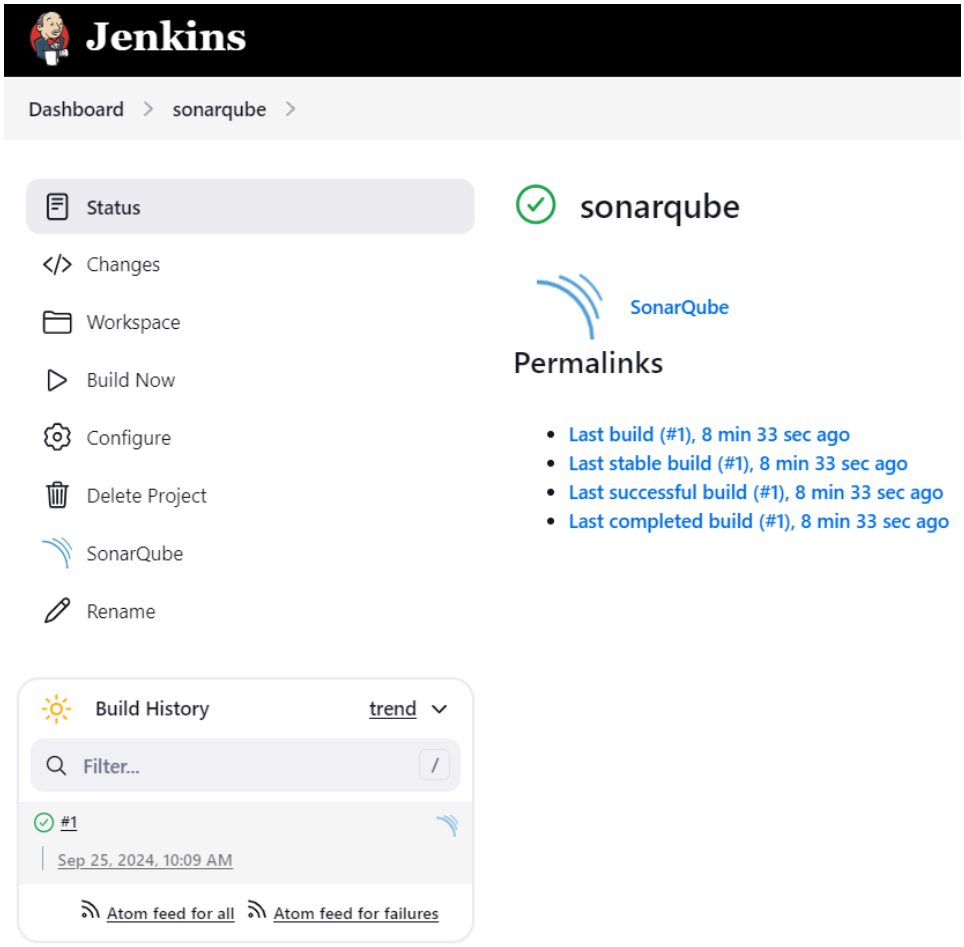
10. Under Build-> Execute SonarQube Scanner, enter these Analysis properties. Mention the SonarQube Project Key, Login, Password, Source path and Host URL.



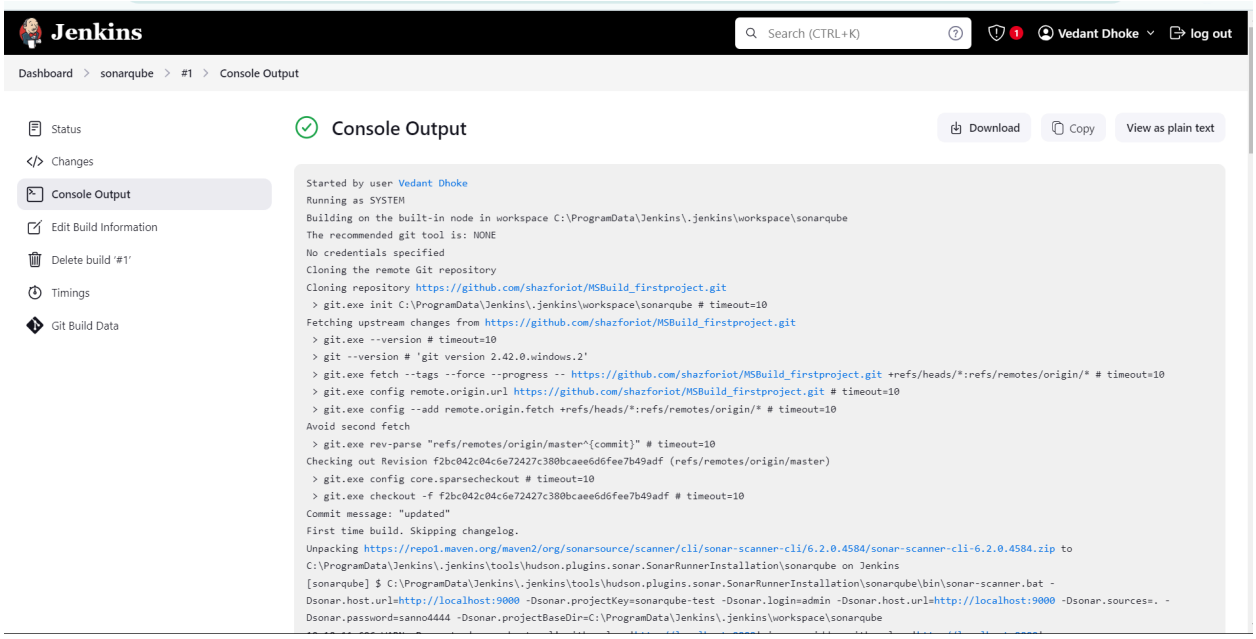
11. Go to http://localhost:9000/<user_name>/permissions and allow Execute Permissions to the Admin user.



12. Run The Build.

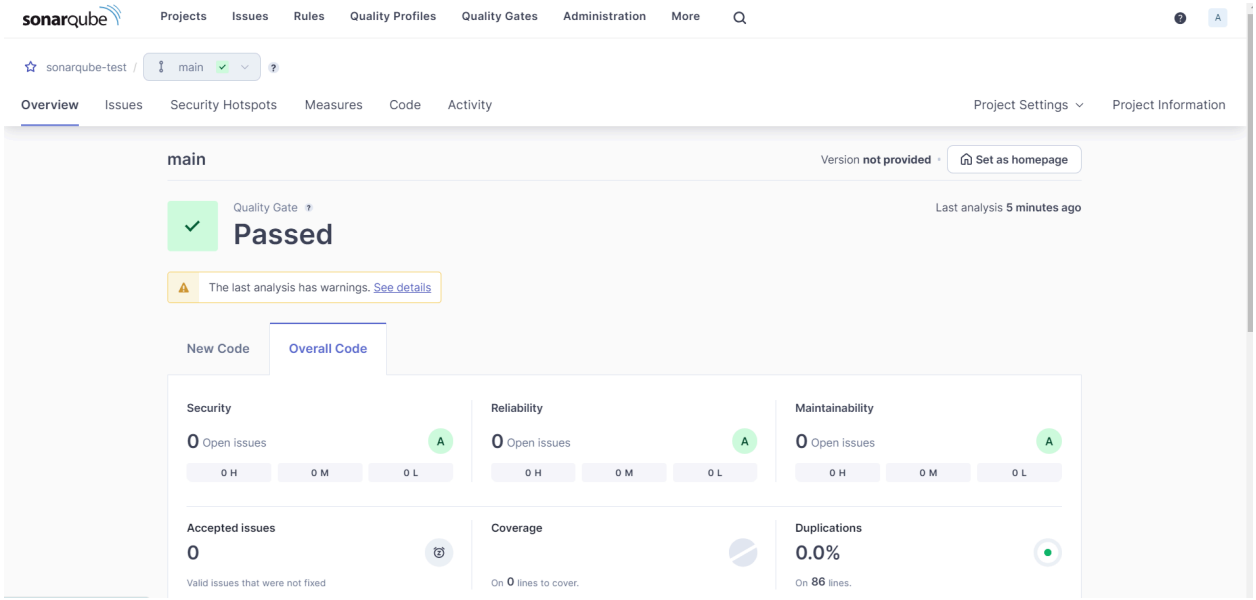


Check the console output.



```
SonarScanner for .NET 5.x or higher, see https://redirect.sonarsource.com/doc/install-configure-scanner-msbuild.html
10:10:57.397 INFO  Sensor C# [csharp] (done) | time=2ms
10:10:57.397 INFO  Sensor Analysis Warnings import [csharp]
10:10:57.399 INFO  Sensor Analysis Warnings import [csharp] (done) | time=4ms
10:10:57.401 INFO  Sensor C# File Caching Sensor [csharp]
10:10:57.405 WARN  Incremental PR analysis: Could not determine common base path, cache will not be computed. Consider setting 'sonar.projectBaseDir'
property.
10:10:57.405 INFO  Sensor C# File Caching Sensor [csharp] (done) | time=5ms
10:10:57.405 INFO  Sensor Zero Coverage Sensor
10:10:57.424 INFO  Sensor Zero Coverage Sensor (done) | time=19ms
10:10:57.428 INFO  SCM Publisher SCM provider for this project is: git
10:10:57.430 INFO  SCM Publisher 4 source files to be analyzed
10:10:58.315 INFO  SCM Publisher 4/4 source files have been analyzed (done) | time=883ms
10:10:58.324 INFO  CPD Executor Calculating CPD for 0 files
10:10:58.363 INFO  CPD Executor CPD calculation finished (done) | time=0ms
10:10:58.372 INFO  SCM revision ID 'f2bc042c04c6e72427c380bcaee6d6fee7b49adf'
10:10:58.843 INFO  Analysis report generated in 226ms, dir size=201.0 kB
10:10:58.903 INFO  Analysis report compressed in 45ms, zip size=22.2 kB
10:10:59.397 INFO  Analysis report uploaded in 491ms
10:10:59.401 INFO  ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=sonarqube-test
10:10:59.402 INFO  Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
10:10:59.403 INFO  More about the report processing at http://localhost:9000/api/ce/task?id=2620d8fe-827f-4a3c-999f-1ed8a7b15249
10:10:59.429 INFO  Analysis total time: 30.223 s
10:10:59.431 INFO  SonarScanner Engine completed successfully
10:10:59.519 INFO  EXECUTION SUCCESS
10:10:59.521 INFO  Total time: 47.815s
Finished: SUCCESS
```

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



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

Conclusion

1. Docker Container Issues: The SonarQube container might not start because your system doesn't have enough memory or processing power. SonarQube needs around 2GB of RAM to work properly, so if your system is low on resources, the container won't run.

2. Login Problems in SonarQube: You might have trouble logging in with the default username (admin) and password (admin). This could happen if there was a configuration issue with SonarQube or if the default password was changed during previous setups.

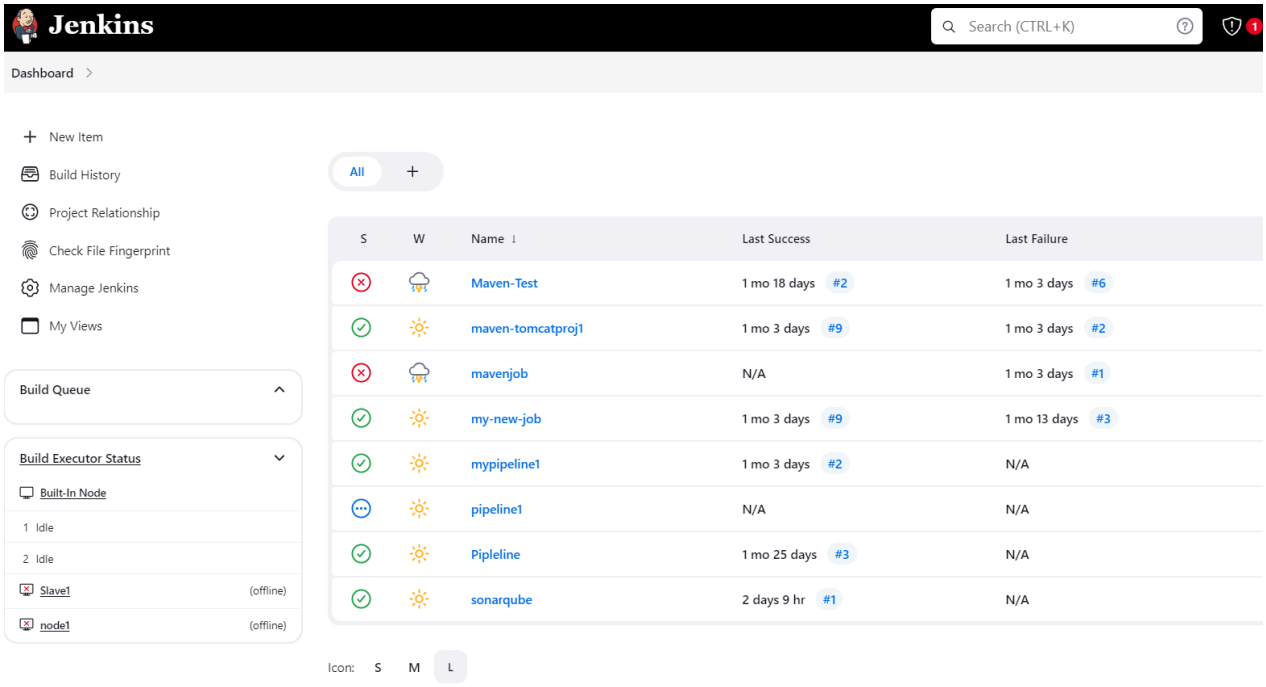
3. Jenkins Plugin Installation Errors: While installing the SonarQube Scanner plugin in Jenkins, you might encounter failures due to network issues or proxy settings, preventing the plugin from downloading correctly.

4. Incorrect SonarQube Configuration in Jenkins: While configuring SonarQube in Jenkins, entering the wrong project key, username, or password can cause the scan to fail. Ensuring accurate information is critical for a successful scan.

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. dive deep into this segment, let's first understand what is meant by the term 'pipeline'?

1. Open Jenkin dashboard.

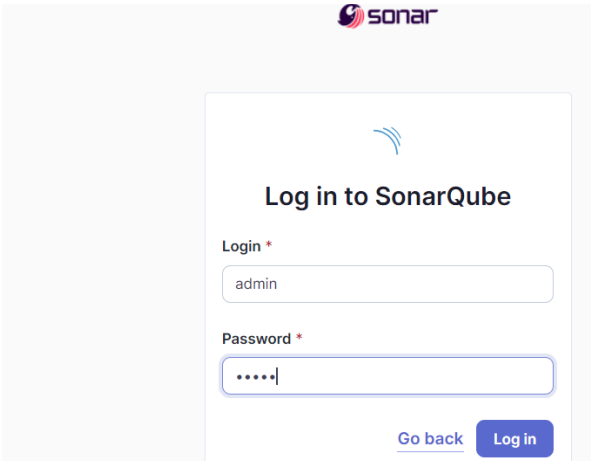


2. Run SonarQube in a Docker container using this command -

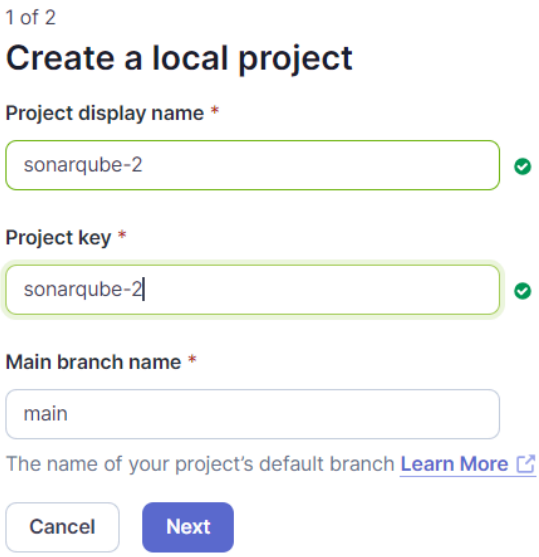
```
C:\Users\91900>docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
d23acccacd96c274f5f87912674ecf2d9adffff185a940c24740f44b29534485
```

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*.

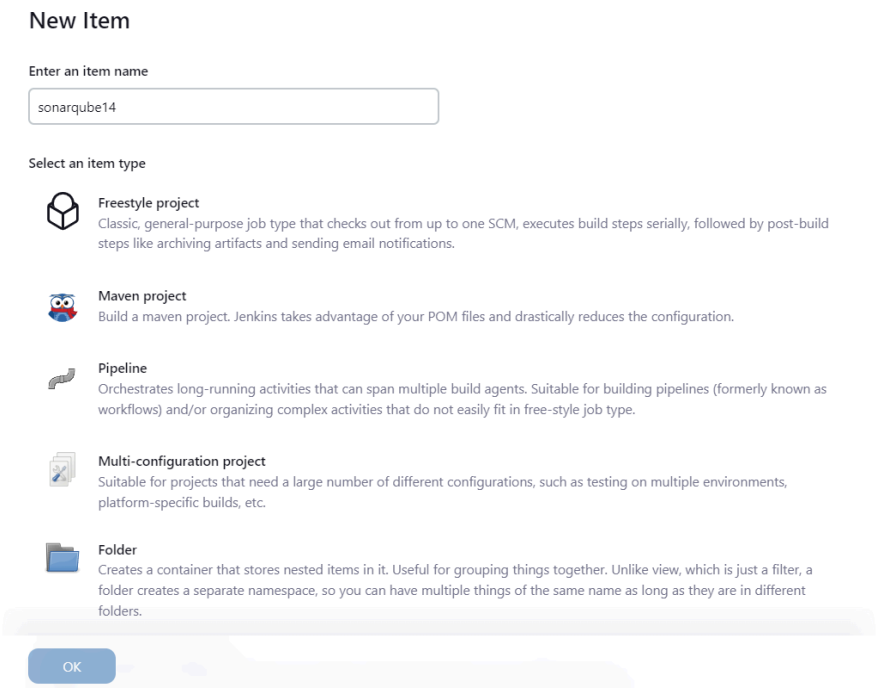


5. Create a manual project in SonarQube with the name **sonarqube-test**



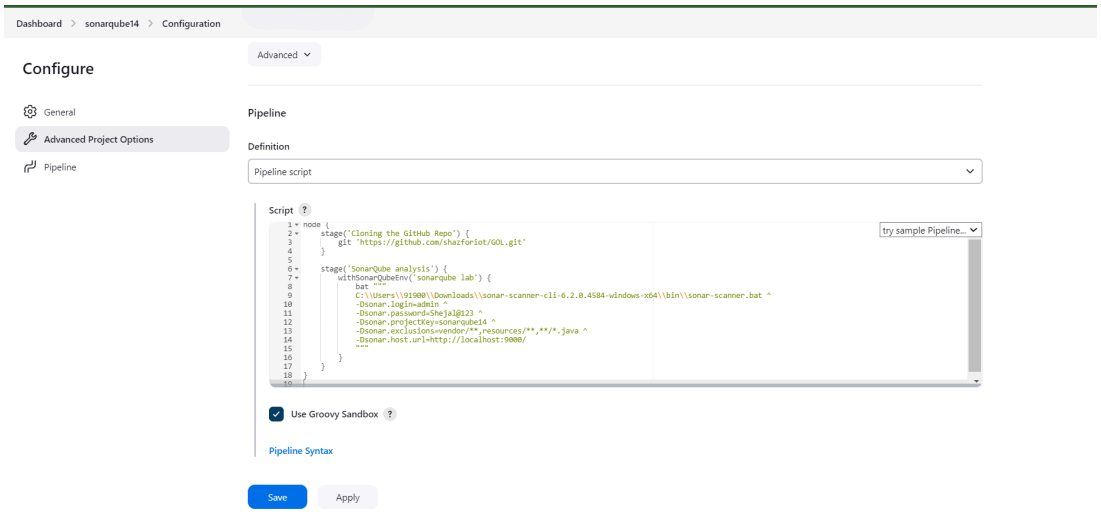
Setup the project and come back to Jenkins Dashboard.

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



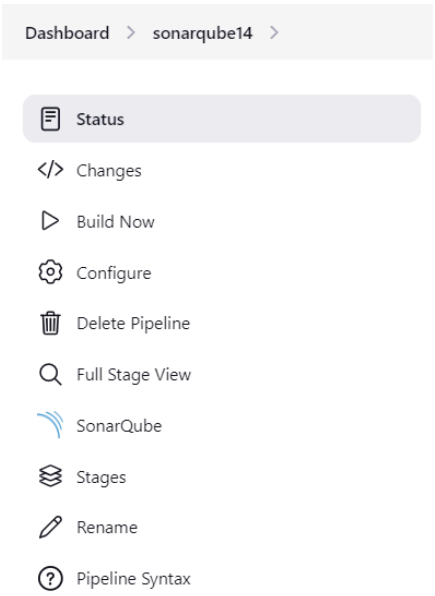
7. Under Pipeline Script, enter the following -

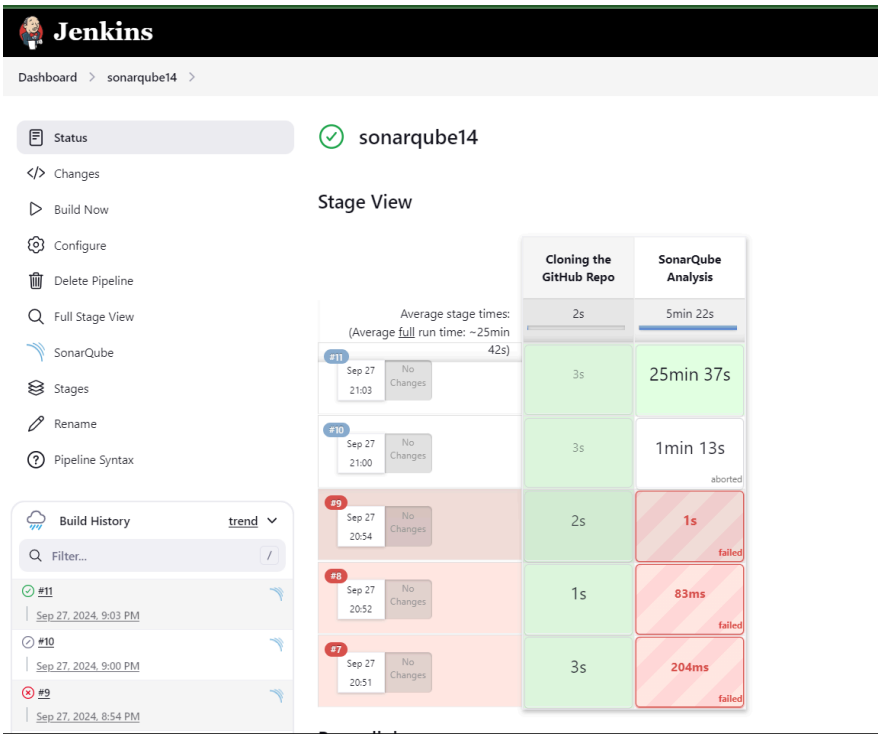
```
node {
   stage('Cloning the GitHub Repo') {
      git 'https://github.com/shazforiot/GOL.git'
   }
   stage('SonarQube analysis') {
      withSonarQubeEnv('sonarqube') {
         sh "<PATH_TO_SONARQUBE_FOLDER>//bin//sonar-scanner \
         -D sonar.login=<SonarQube_USERNAME> \
         -D sonar.password=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



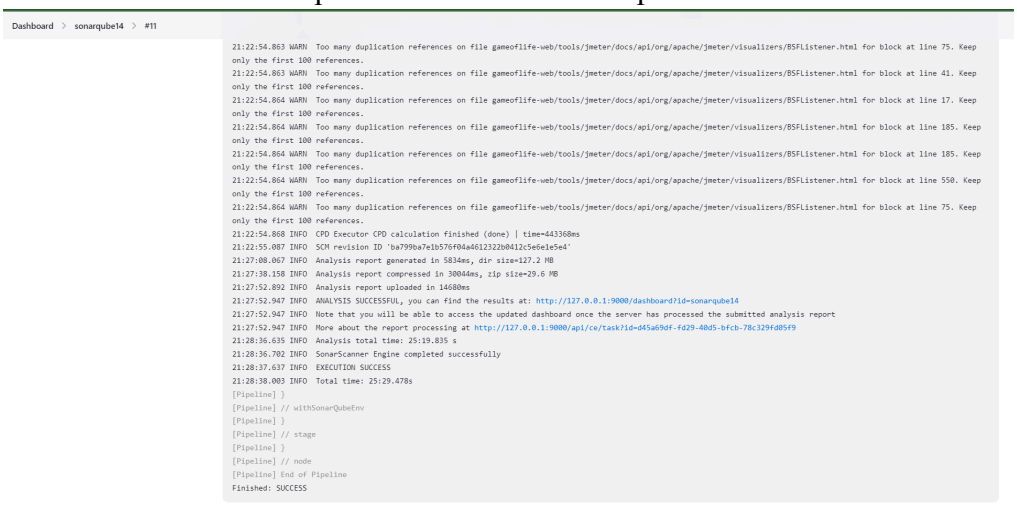
It is a java sample project which has a lot of repetitions and issues that will be detected by SonarQube.

8. Run The Build.

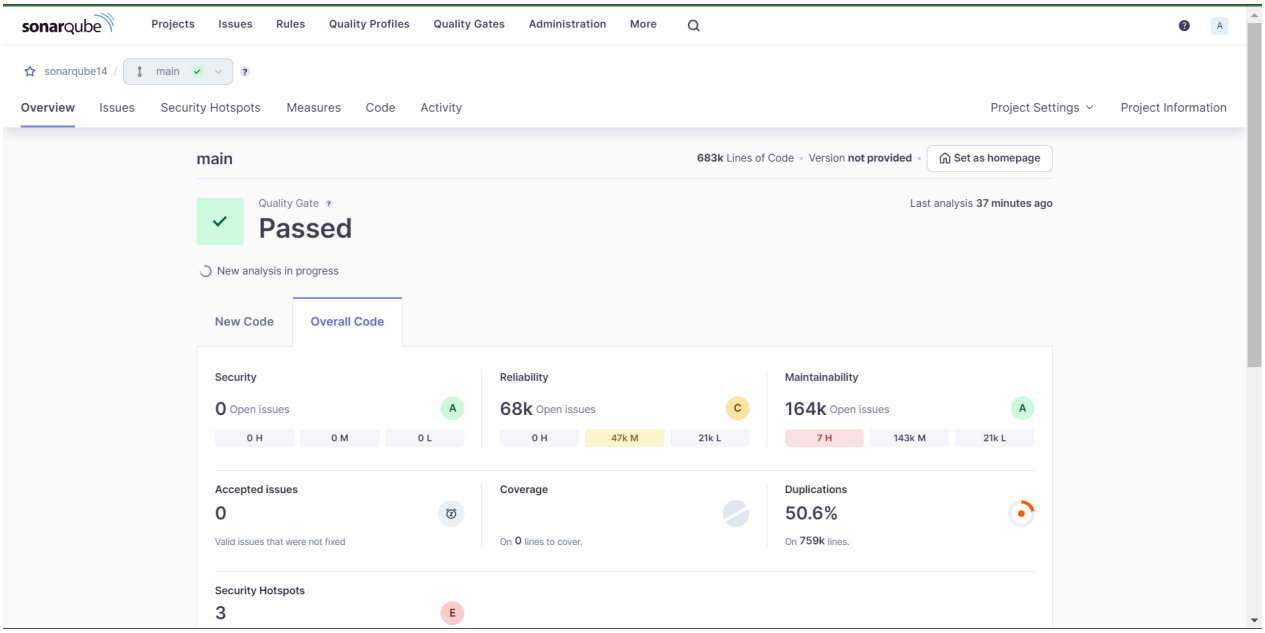




9. Check the console output once the build is complete.



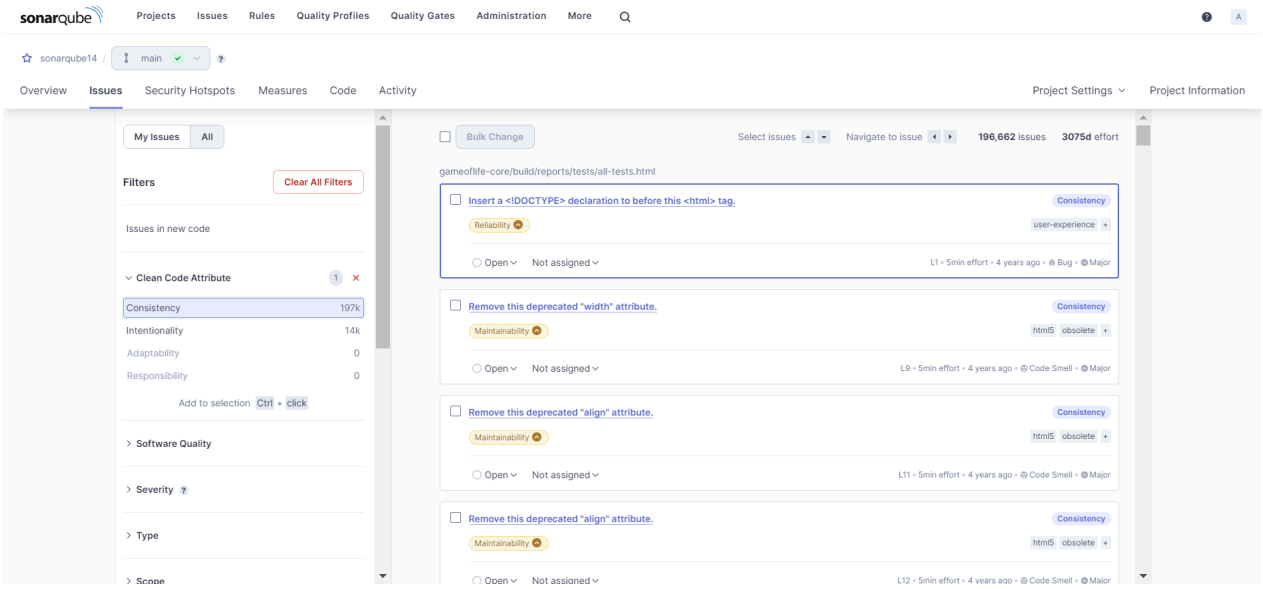
10. After that, check the project in SonarQube.



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

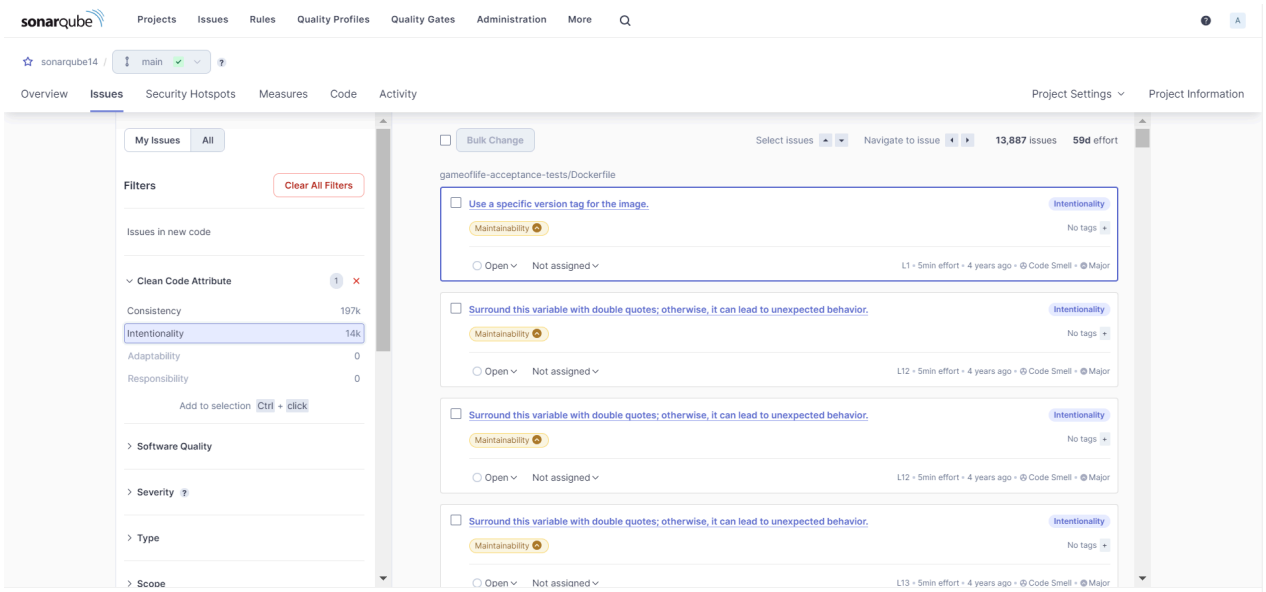
11. **Code Problems -**

Consistency

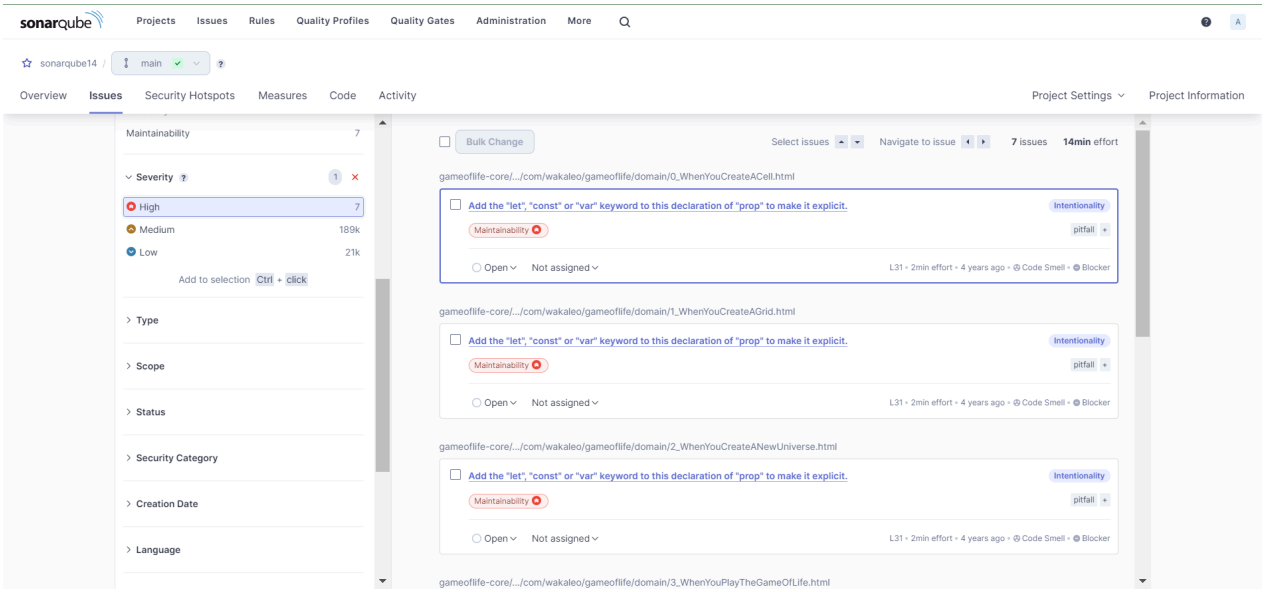


Name:Shejal Tiwari Div/Roll no :D15C/57

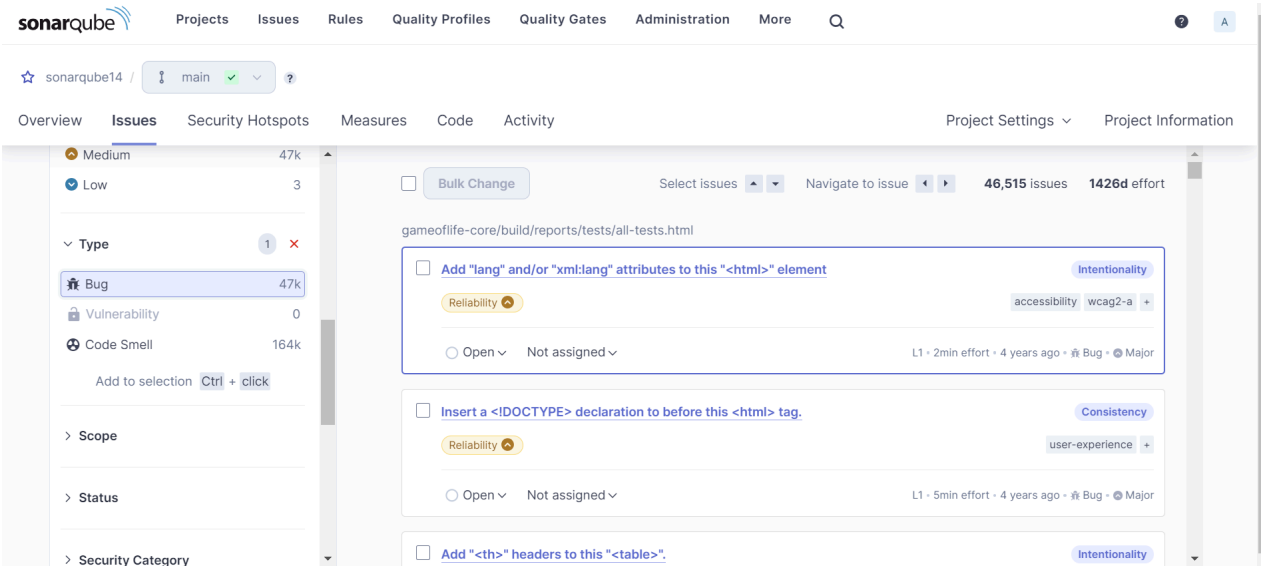
Intentionality

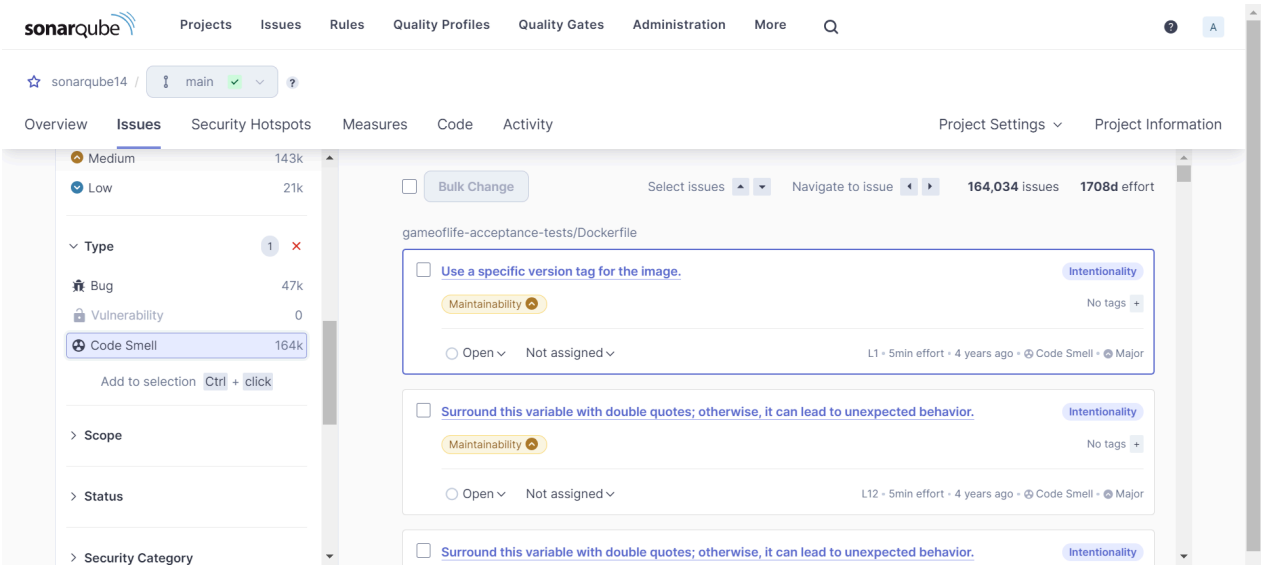


Severity

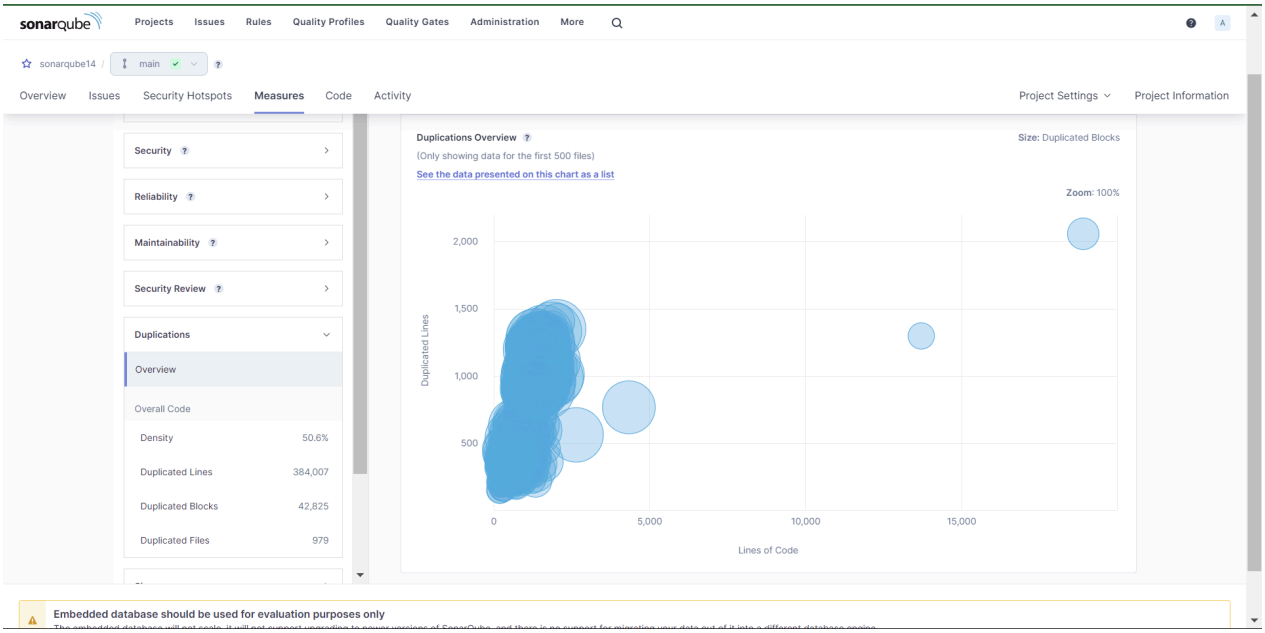


Bugs and Code Smells

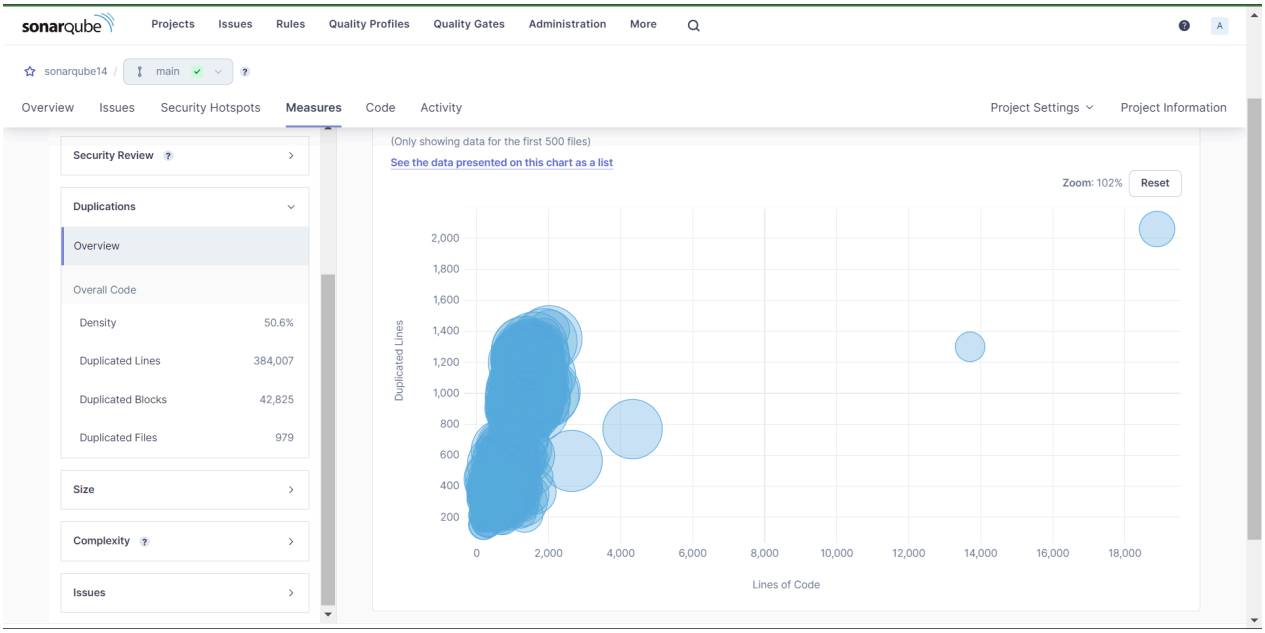




Duplicates



Cyclomatic Complexities



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

Conclusion: In this experiment, we successfully cloned a GitHub repository and integrated it with SonarQube for code analysis. During the analysis, SonarQube highlighted various types of program issues, including:

- **Consistency**: Code standards and formatting issues.
- **Intentionality**: Potential logical or structural errors in code.
- **Severity**: Classification of errors by criticality.
- **Duplicates**: Identification of duplicate code sections.
- **Cyclomatic Complexity**: Measurement of code complexity based on control flow.

These insights provided a comprehensive understanding of the code quality and highlighted areas that needed improvement.

Issues Faced:

1. **SonarQube Scanner Path Error**: The Jenkins pipeline script was initially unable to run correctly due to Jenkins not detecting the correct path for the SonarQube Scanner. This required manual intervention to modify the script and set the proper path for the scanner bash file, allowing the pipeline to function as expected.

Name: Shejal Tiwari Class/ Roll No: D15C/ 57

Experiment No: 9

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

Theory:

What is Nagios?

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

Why We Need Nagios tool?

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

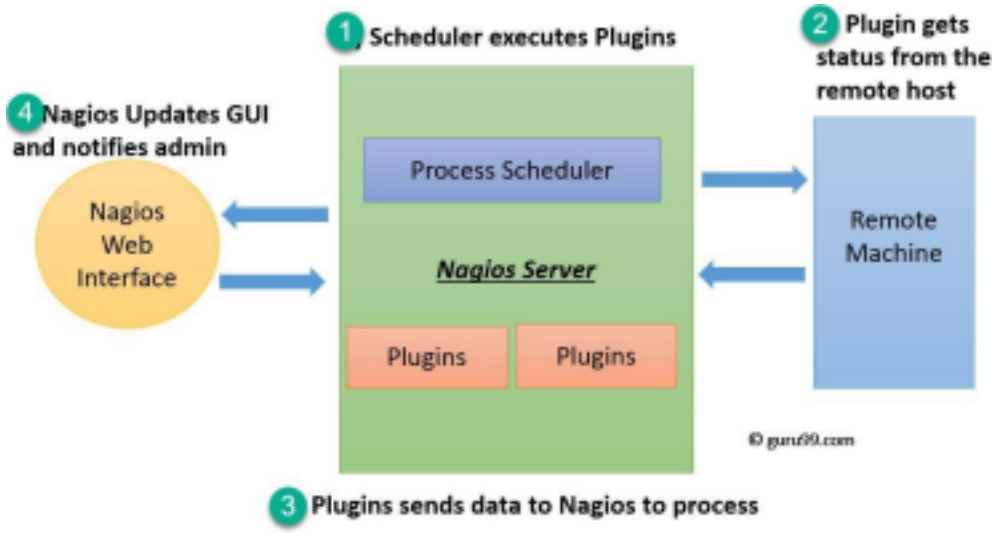
Features of Nagios
Following are the important features of Nagios monitoring tool:
● Relatively scalable, Manageable, and Secure
● Good log and database system
● Informative and attractive web interfaces
● Automatically send alerts if condition changes
● If the services are running fine, then there is no need to do check that host is an alive
● Helps you to detect network errors or server crashes
● You can troubleshoot the performance issues of the server.
● The issues, if any, can be fixed automatically as they are identified during the monitoring process
● You can monitor the entire business process and IT infrastructure with a single pass
● The product’s architecture is easy to write new plugins in the language of your choice

Name: Shejal Tiwari Class/ Roll No: D15C/ 57

● Nagios allows you to read its configuration from an entire directory which helps you to decide how to define individual files
● Utilizes topology to determine dependencies
● Monitor network services like HTTP, SMTP, HTTP, SNMP, FTP, SSH, POP, etc.
● Helps you to define network host hierarchy using parent hosts
● Ability to define event handlers that runs during service or host events for proactive problem resolution
● Support for implementing redundant monitoring hosts

Nagios Architecture

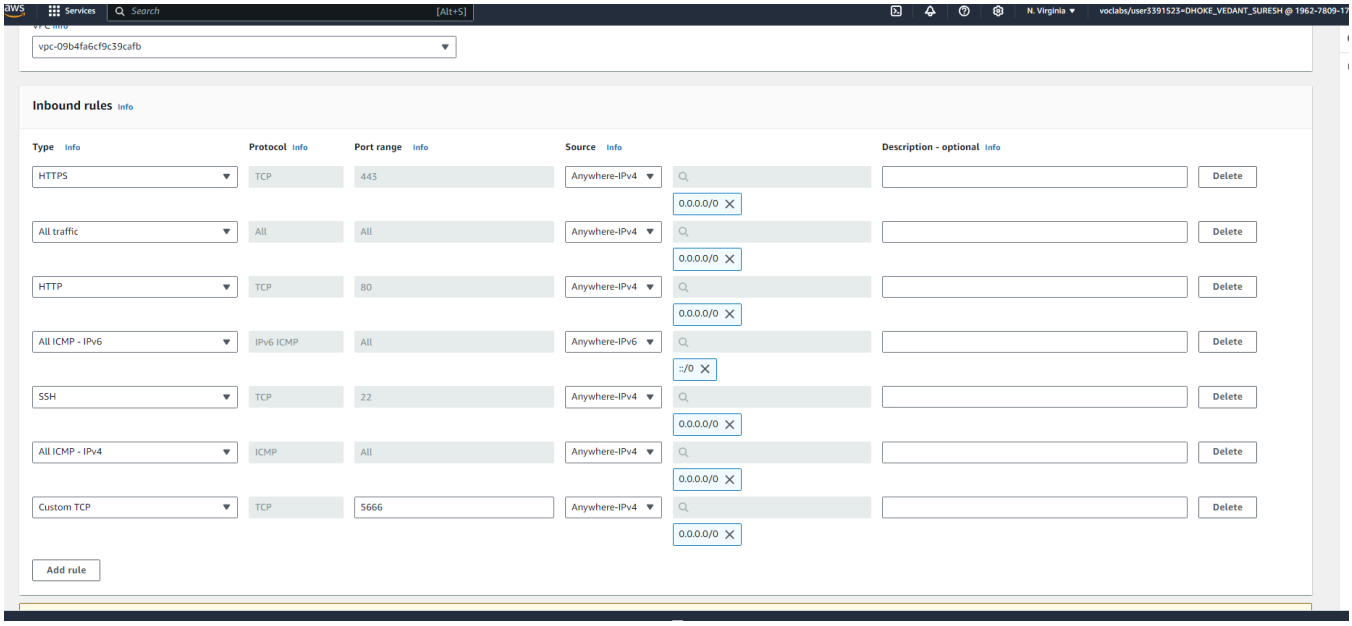
Nagios is a client-server architecture. Usually, on a network, a Nagios server is running on a host, and plugins are running on all the remote hosts which should be monitored.



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

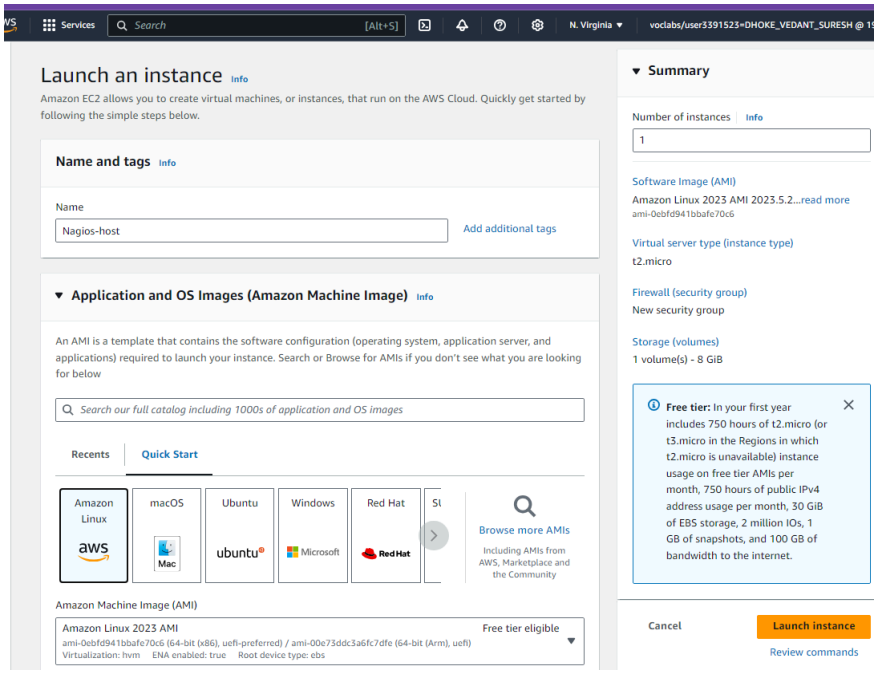
Name: Shejal Tiwari Class/ Roll No: D15C/ 57

Step 1: Login to your AWS account Personal / Academy. Click on EC2 instance then click on Create Security Group. Give the name as Nagios and any description and add the following inbounds rules.



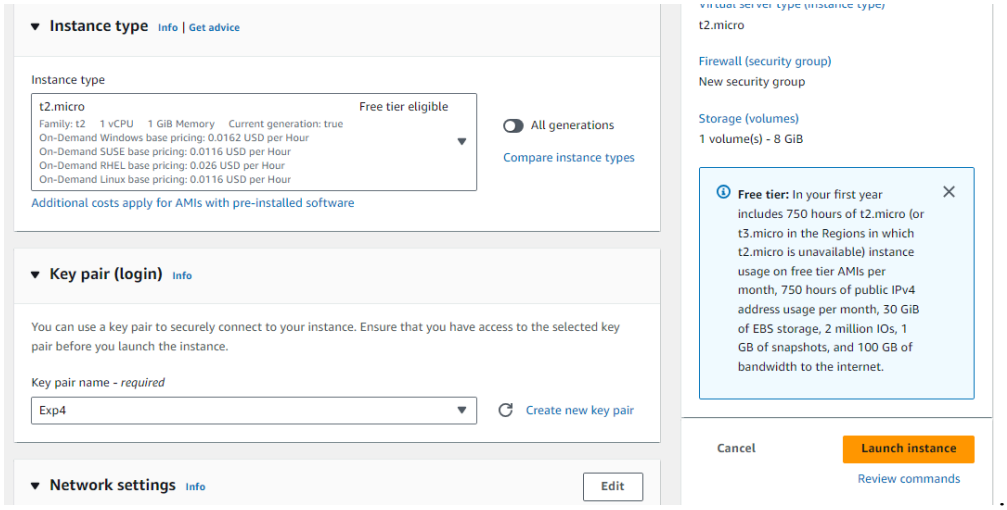
Step 2: Now Create a new EC2 instance. Name: Nagios-host ,AMI: Amazon Linux, Instance Type: t2.micro.

Name: Shejal Tiwari Class/ Roll No: D15C/ 57

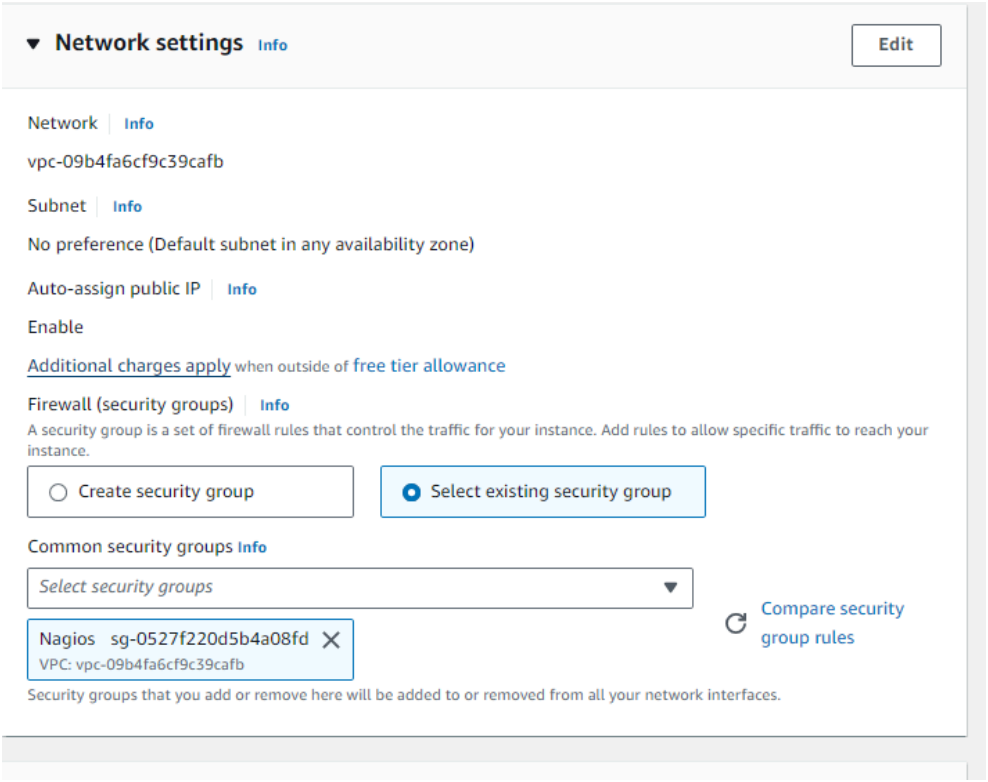


For Key pair : Click on create key and make key of type RSA with extension .pem . Key will be downloaded to your local machine.

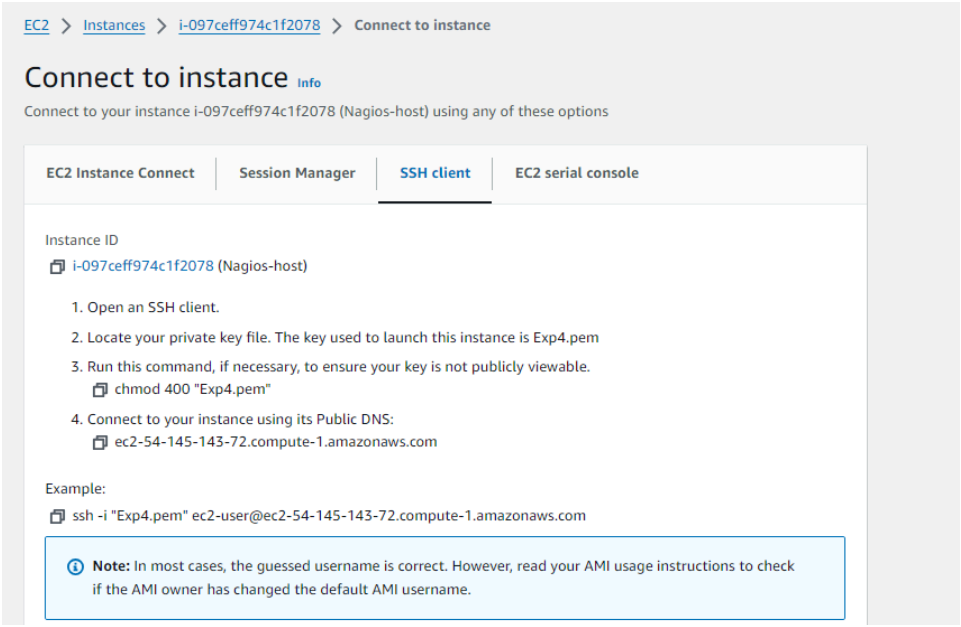
Now select that key in key pair if you already have key with type RSA and extension .pem no need to create new key but you must have that key downloaded.



Select the Existing Security Group and select the Security Group we have created in Step 1



Step 3: Now After creating the EC2 Instance click on connect and then copy the command which is given as example in the SSH Client section .



Now open the terminal in the folder where your key(RSA key with .pem) is located.and paste that copied command.

Successfully connected to the instance.

```
C:\Users\LENOVO>ssh -i "Exp4.pem" ec2-user@ec2-54-210-4-52.compute-1.amazonaws.com
The authenticity of host 'ec2-54-210-4-52.compute-1.amazonaws.com (54.210.4.52)' can't be establ
ished.
ED25519 key fingerprint is SHA256:YqoGBku3mptyrGnwBwnKC060wMjlNnInQm5MPBeB1RM.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-54-210-4-52.compute-1.amazonaws.com' (ED25519) to the list of kn
own hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-31-38-62 ~]$
```

Step 4: Now Run the following command to make a new user.

sudo adduser -m nagios

sudo passwd nagios

```
[ec2-user@ip-172-31-38-62 ~]$ sudo adduser -m nagios
[ec2-user@ip-172-31-38-62 ~]$ sudo passwd nagios
Changing password for user nagios.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[ec2-user@ip-172-31-38-62 ~]$
```

Step 5: Now Run the following command to make a new user group.

sudo groupadd nagcmd

sudo usermod -a -G nagcmd nagios

sudo usermod -a -G nagcmd apache

```
[ec2-user@ip-172-31-38-62 ~]$ sudo groupadd nagcmd
[ec2-user@ip-172-31-38-62 ~]$ sudo usermod -a -G nagcmd nagios
sudo usermod -a -G nagcmd apache
[ec2-user@ip-172-31-38-62 ~]$
```

Step 6: Now make a new directory and go to that directory.

mkdir ~/downloads

cd ~/downloads

```
[ec2-user@ip-172-31-38-62 ~]$ mkdir ~/downloads
cd ~/downloads
```

Step 7: Now to download the Nagios 4.5.5 and Nagios-plugins 2.4.11 run the following commands respectively.

wget https://go.nagios.org/l/975333/2024-09-17/6kqcx

```
[ec2-user@ip-172-31-38-62 downloads]$ wget https://go.nagios.org/l/975333/2024-09-17/6kqcx
--2024-10-02 07:05:23--  https://go.nagios.org/l/975333/2024-09-17/6kqcx
Resolving go.nagios.org (go.nagios.org)... 34.237.219.119, 3.92.120.28, 18.208.125.13, ...
Connecting to go.nagios.org (go.nagios.org)|34.237.219.119|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: http://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.5.tar.gz?utm_source=Nagios.org&utm_content=Download+Form&utm_camp
aign=Core+4.5.5+Download+&pi_content=1e9662c93afb2ed6bd2e3f3cc38771a7f01125e969f2a75b0e2254439d4a81d8 [following]
--2024-10-02 07:05:23--  http://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.5.tar.gz?utm_source=Nagios.org&utm_content=Downloa
d+Form&utm_campaign=Core+4.5.5+Download+&pi_content=1e9662c93afb2ed6bd2e3f3cc38771a7f01125e969f2a75b0e2254439d4a81d8
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|:80... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.5.tar.gz?utm_source=Nagios.org&utm_content=Download+Form&utm_cam
paign=Core+4.5.5+Download+&pi_content=1e9662c93afb2ed6bd2e3f3cc38771a7f01125e969f2a75b0e2254439d4a81d8 [following]
--2024-10-02 07:05:23--  https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.5.tar.gz?utm_source=Nagios.org&utm_content=Downlo
ad+Form&utm_campaign=Core+4.5.5+Download+&pi_content=1e9662c93afb2ed6bd2e3f3cc38771a7f01125e969f2a75b0e2254439d4a81d8
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: '6kqcx'

6kqcx                  100%[==============================>]   1.97M  6.65MB/s    in 0.3s

2024-10-02 07:05:24 (6.65 MB/s) - '6kqcx' saved [2065473/2065473]

[ec2-user@ip-172-31-38-62 downloads]$
```

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

```
[ec2-user@ip-172-31-38-62 downloads]$ wget https://nagios-plugins.org/download/nagios-plugins-2.
4.11.tar.gz
--2024-10-02 07:05:58--  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.t 100%[==================================>]   2.62M  7.32MB/s    in 0.4s

2024-10-02 07:05:59 (7.32 MB/s) - 'nagios-plugins-2.4.11.tar.gz' saved [2753049/2753049]

[ec2-user@ip-172-31-38-62 downloads]$
```

Step 8: Now to extract the files from the downloaded Nagios 4.5.5 run the following command.

tar zxvf 6kqcx

```
[ec2-user@ip-172-31-38-62 downloads]$ tar zxvf 6kqcx
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
```

Step 9: Now change the directory to nagios-4.5.5 (Or which version you have downloaded)

```
[ec2-user@ip-172-31-38-62 downloads]$ cd nagios-4.5.5
```

Step 10: Now run the following command to configure.

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

```
[ec2-user@ip-172-31-38-62 nagios-4.5.5]$ ./configure --with-command-group=nagcmd
checking for a BSD-compatible install... /usr/bin/install -c
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether the compiler supports GNU C... yes
checking whether gcc accepts -g... yes
checking for gcc option to enable C11 features... none needed
checking whether make sets $(MAKE)... yes
checking whether ln -s works... yes
checking for strip... /usr/bin/strip
checking for sys/wait.h that is POSIX.1 compatible... yes
checking for stdio.h... yes
checking for stdlib.h... yes
checking for string.h... yes
checking for inttypes.h... yes
```

At the end we have found the error of cannot find ssl header .

```
checking for type of socket size... size_t
checking for Kerberos include files... configure: WARNING: could not find include files
checking for pkg-config... pkg-config
checking for SSL headers... configure: error: Cannot find ssl headers
[ec2-user@ip-172-31-38-62 nagios-4.5.5]$
```

So run following command to install ssl.

sudo yum install openssl-devel

```
[ec2-user@ip-172-31-38-62 nagios-4.5.5]$ sudo yum install openssl-devel
Last metadata expiration check: 0:10:57 ago on Wed Oct  2 06:57:40 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                  21 MB/s | 3.0 MB     00:00
--------------------------------------------------------------------------------
Total                                                           16 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!
[ec2-user@ip-172-31-38-62 nagios-4.5.5]$
```

Now rerun the command **./configure --with-command-group=nagcmd**

```
[ec2-user@ip-172-31-38-62 nagios-4.5.5]$ ./configure --with-command-group=nagcmd
checking for a BSD-compatible install... /usr/bin/install -c
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether the compiler supports GNU C... yes
checking whether gcc accepts -g... yes
checking for gcc option to enable C11 features... none needed
checking whether make sets $(MAKE)... yes
checking whether ln -s works... yes
checking for strip... /usr/bin/strip
checking for sys/wait.h that is POSIX.1 compatible... yes
checking for stdio.h... yes
```

```
       web interface

  make install-classicui
     - This installs the classic theme for the Nagios
       web interface


*** 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.

[ec2-user@ip-172-31-38-62 nagios-4.5.5]$
```

Step 11: Now run the following commands to setup the Nagios.
sudo make install

```
[ec2-user@ip-172-31-38-62 nagios-4.5.5]$ sudo make install
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
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/html'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/media
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/stylesheets
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/contexthelp
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/docs
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/docs/images
/usr/bin/install -c -m 775 -o nagios -g nagcmd -d /usr/local/nagios/var/spool/checkres
chmod g+s /usr/local/nagios/var/spool/checkresults

*** Main program, CGIs and HTML files installed ***

You can continue with installing Nagios as follows (type 'make'
without any arguments for a list of all possible options):

  make install-init
     - This installs the init script in /lib/systemd/system

  make install-commandmode
     - This installs and configures permissions on the
       directory for holding the external command file

  make install-config
     - This installs sample config files in /usr/local/nagios/etc

make[1]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5'
[ec2-user@ip-172-31-38-62 nagios-4.5.5]$
```

sudo make install-init

```
[ec2-user@ip-172-31-38-62 nagios-4.5.5]$ sudo make install-init
/usr/bin/install -c -m 755 -d -o root -g root /lib/systemd/system
/usr/bin/install -c -m 755 -o root -g root startup/default-service /lib/systemd/system/nagios.service
[ec2-user@ip-172-31-38-62 nagios-4.5.5]$ sudo make install-config
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc/objects
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/nagios.cfg /usr/local/nagios/etc/nagios.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/cgi.cfg /usr/local/nagios/etc/cgi.cfg
/usr/bin/install -c -b -m 660 -o nagios -g nagios sample-config/resource.cfg /usr/local/nagios/etc/resource.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/templates.cfg /usr/local/nagios/etc/objects/templates.
cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/commands.cfg /usr/local/nagios/etc/objects/commands.cf
g
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/contacts.cfg /usr/local/nagios/etc/objects/contacts.cf
g
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/timeperiods.cfg /usr/local/nagios/etc/objects/timeperi
ods.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/localhost.cfg /usr/local/nagios/etc/objects/localhost.
cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/windows.cfg /usr/local/nagios/etc/objects/windows.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/printer.cfg /usr/local/nagios/etc/objects/printer.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/switch.cfg /usr/local/nagios/etc/objects/switch.cfg

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

sudo make install-webconf

```
[ec2-user@ip-172-31-38-62 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 ***

[ec2-user@ip-172-31-38-62 nagios-4.5.5]$
```

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

```
[ec2-user@ip-172-31-38-62 nagios-4.5.5]$ sudo htpasswd -c /usr/local/nagios/etc/htpasswd.use
rs nagiosadmin
New password:
Re-type new password:
Adding password for user nagiosadmin
[ec2-user@ip-172-31-38-62 nagios-4.5.5]$
```

Now to restart the httpd service run the following command.

sudo service httpd restart

```
[ec2-user@ip-172-31-38-62 nagios-4.5.5]$ sudo service httpd restart
Redirecting to /bin/systemctl restart httpd.service
```

Step 12: Now to extract the files from the downloaded Nagios plugin 2.4.11 run the following command first change the directory.

cd ~/downloads

tar zxvf nagios-plugins-2.4.11.tar.gz

```
[ec2-user@ip-172-31-38-62 nagios-4.5.5]$ cd ~/downloads
[ec2-user@ip-172-31-38-62 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/
```

Step 13: Now change the directory to nagios-plugins-2.4.11 and run the config command to configure.
cd nagios-plugins-2.4.11

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

```
[ec2-user@ip-172-31-38-62 nagios-plugins-2.4.11]$ ./configure --with-nagios-user=nagios --with-nagios-group=nagios
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /usr/bin/mkdir -p
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking whether make supports nested variables... yes
checking whether to enable maintainer-specific portions of Makefiles... yes
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
```

Step 14: Run the following commands to check nagios and start it.
sudo chkconfig --add nagios

```
[ec2-user@ip-172-31-38-62 nagios-plugins-2.4.11]$ sudo chkconfig --add nagios
error reading information on service nagios: No such file or directory
[ec2-user@ip-172-31-38-62 nagios-plugins-2.4.11]$
```

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

```
[ec2-user@ip-172-31-38-62 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-172-31-38-62 nagios-plugins-2.4.11]$
```

cd

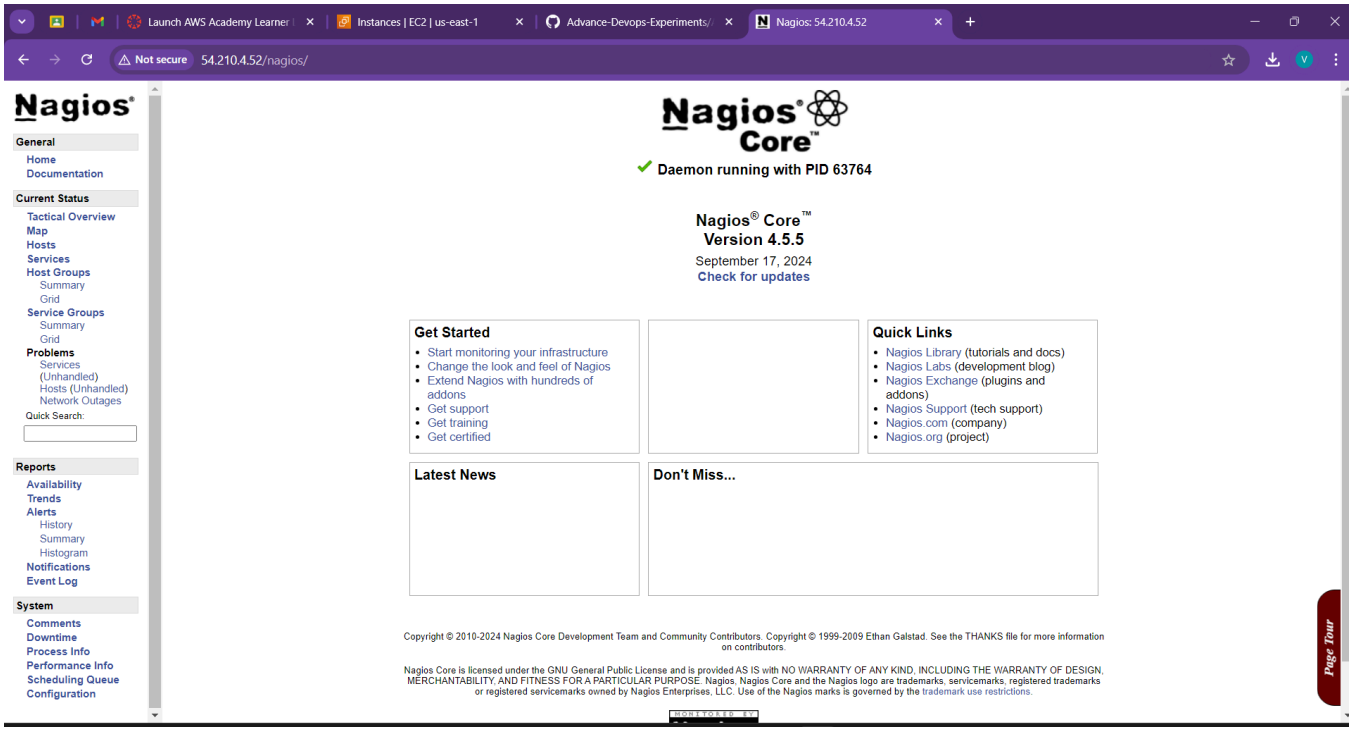
sudo service nagios start

```
[ec2-user@ip-172-31-38-62 nagios-plugins-2.4.11]$ cd
[ec2-user@ip-172-31-38-62 ~]$ sudo service nagios start
Redirecting to /bin/systemctl start nagios.service
[ec2-user@ip-172-31-38-62 ~]$
```

sudo systemctl status nagios

```
[ec2-user@ip-172-31-38-62 ~]$ sudo systemctl status nagios
● nagios.service - Nagios Core 4.5.5
     Loaded: loaded (/usr/lib/systemd/system/nagios.service; disabled; preset: disabled)
     Active: active (running) since Wed 2024-10-02 07:41:50 UTC; 5s ago
       Docs: https://www.nagios.org/documentation
    Process: 63762 ExecStartPre=/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg (c>
    Process: 63763 ExecStart=/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg (code>
   Main PID: 63764 (nagios)
      Tasks: 6 (limit: 1112)
     Memory: 5.4M
        CPU: 74ms
     CGroup: /system.slice/nagios.service
             ├─63764 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─63765 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─63766 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─63767 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─63768 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             └─63769 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg

Oct 02 07:41:50 ip-172-31-38-62.ec2.internal nagios[63764]: qh: core query handler registered
Oct 02 07:41:50 ip-172-31-38-62.ec2.internal nagios[63764]: qh: echo service query handler register>
Oct 02 07:41:50 ip-172-31-38-62.ec2.internal nagios[63764]: qh: help for the query handler register>
Oct 02 07:41:50 ip-172-31-38-62.ec2.internal nagios[63764]: wproc: Successfully registered manager >
Oct 02 07:41:50 ip-172-31-38-62.ec2.internal nagios[63764]: wproc: Registry request: name=Core Work>
Oct 02 07:41:50 ip-172-31-38-62.ec2.internal nagios[63764]: wproc: Registry request: name=Core Work>
Oct 02 07:41:50 ip-172-31-38-62.ec2.internal nagios[63764]: wproc: Registry request: name=Core Work>
Oct 02 07:41:50 ip-172-31-38-62.ec2.internal nagios[63764]: wproc: Registry request: name=Core Work>
Oct 02 07:41:51 ip-172-31-38-62.ec2.internal nagios[63764]: Successfully launched command file work>
Oct 02 07:41:51 ip-172-31-38-62.ec2.internal nagios[63764]: HOST ALERT: localhost;DOWN;SOFT;1;(No o>
lines 1-28/28 (END)
```

Step 15: We can see we have successfully launched the Nagios now . Open **http://<instance public ip >/nagios/** here it is **http://54.210.4.52/nagios** we can see the running web page of nagios.



Conclusion:

In this experiment, we successfully installed and configured Nagios Core, Nagios Plugins, and NRPE on a Linux machine within an AWS EC2 instance. The aim of continuously monitoring a remote system

was achieved by integrating Nagios with the EC2 environment and allowing web access via the Nagios dashboard. We faced several challenges that required troubleshooting:

- **Security Group Configuration:** Setting up the correct inbound rules in the AWS security group was essential but prone to mistakes. Incorrectly configured ports could block HTTP and NRPE communication, preventing access to the Nagios dashboard or monitoring checks.

- **User and Group Permissions:** There were some issues when configuring user and group permissions, especially while adding users to the nagcmd group. If the commands weren't run correctly, Nagios failed to run properly due to incorrect access rights.

- **Dependencies and Package Installation:** While installing Nagios and its plugins, we encountered dependency issues, particularly with OpenSSL. Missing packages or libraries often halted the configuration process. Resolving these involved installing required dependencies and restarting the configuration steps.

Name:Shejal Tiwari Class/Roll No: D15C/ 57

Experiment No: 10

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

Theory:

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

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

Prerequisites:
AWS Academy or Personal account.
Nagios Server running on Amazon Linux Machine. (Refer Experiment No 9)

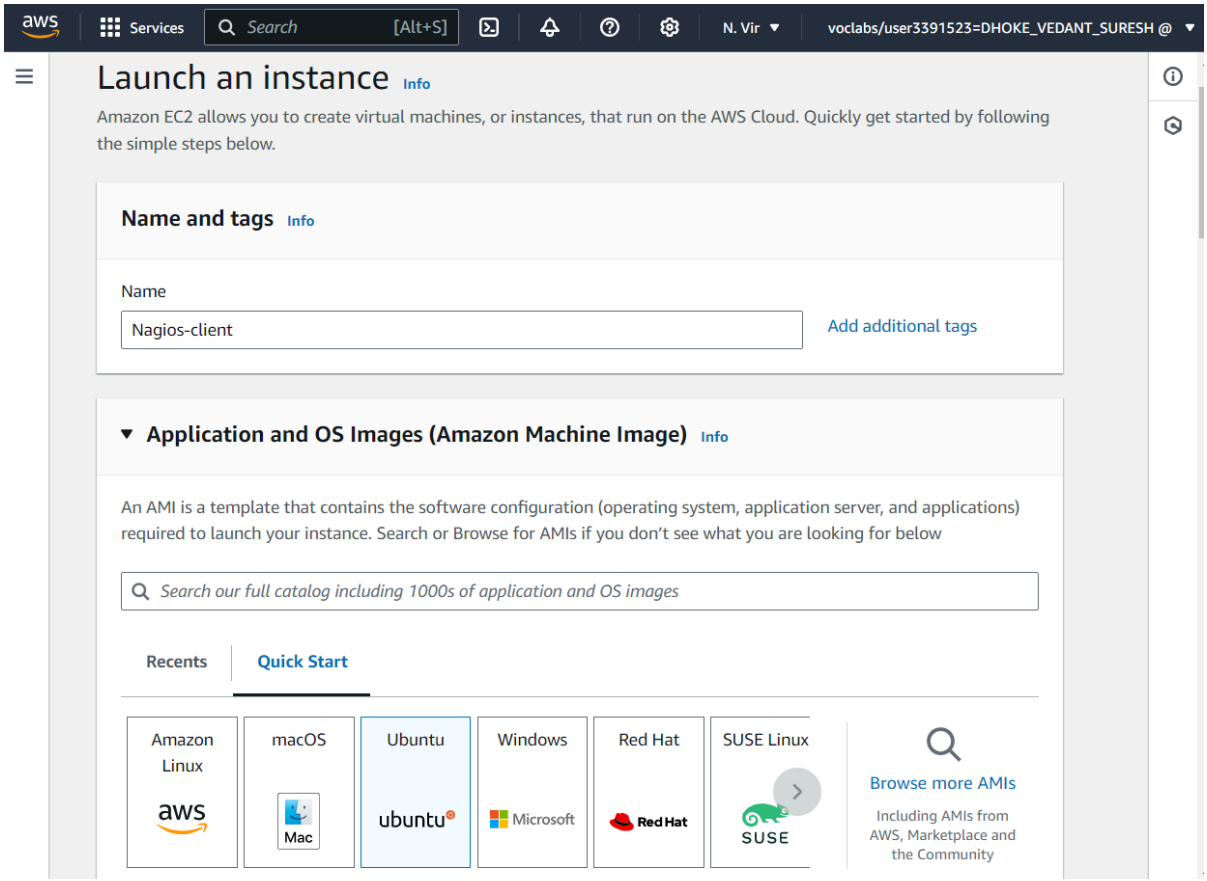
Monitoring Using Nagios:

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

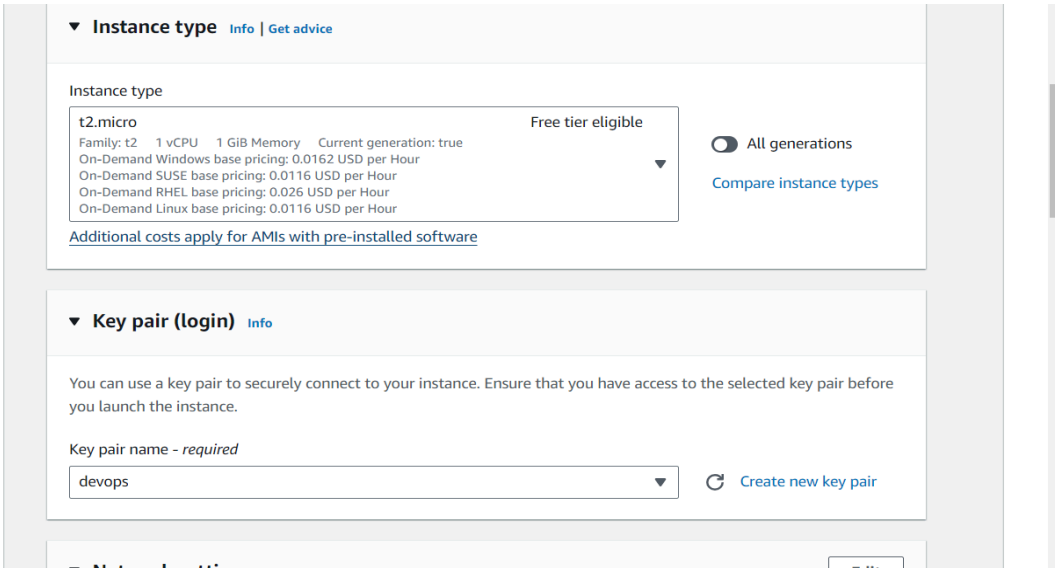
```
ec2-user@ip-172-31-46-196:~
● nagios.service - Nagios Core 4.5.5
     Loaded: loaded (/usr/lib/systemd/system/nagios.service; disabled; preset: disabled)
     Active: active (running) since Sun 2024-10-06 10:58:43 UTC; 4s ago
       Docs: https://www.nagios.org/documentation
    Process: 62217 ExecStartPre=/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
    Process: 62218 ExecStart=/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
   Main PID: 62219 (nagios)
      Tasks: 6 (limit: 1112)
     Memory: 5.4M
        CPU: 74ms
     CGroup: /system.slice/nagios.service
             ├─62219 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─62220 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─62221 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─62222 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─62223 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             └─62224 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg

Oct 06 10:58:43 ip-172-31-46-196.ec2.internal nagios[62219]: qh: core query handler registered
Oct 06 10:58:43 ip-172-31-46-196.ec2.internal nagios[62219]: qh: echo service query handler registered
Oct 06 10:58:43 ip-172-31-46-196.ec2.internal nagios[62219]: qh: help for the query handler registered
Oct 06 10:58:43 ip-172-31-46-196.ec2.internal nagios[62219]: wproc: Successfully registered manager as @wproc with query handler
Oct 06 10:58:43 ip-172-31-46-196.ec2.internal nagios[62219]: wproc: Registry request: name=Core Worker 62223;pid=62223
Oct 06 10:58:43 ip-172-31-46-196.ec2.internal nagios[62219]: wproc: Registry request: name=Core Worker 62221;pid=62221
Oct 06 10:58:43 ip-172-31-46-196.ec2.internal nagios[62219]: wproc: Registry request: name=Core Worker 62222;pid=62222
Oct 06 10:58:43 ip-172-31-46-196.ec2.internal nagios[62219]: wproc: Registry request: name=Core Worker 62220;pid=62220
Oct 06 10:58:43 ip-172-31-46-196.ec2.internal nagios[62219]: Successfully launched command file worker with pid 62224
Oct 06 10:58:43 ip-172-31-46-196.ec2.internal nagios[62219]: HOST ALERT: localhost;DOWN;SOFT;1;(No output on stdout) stderr: execvp(/usr/local/nagios/libex
~
```

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

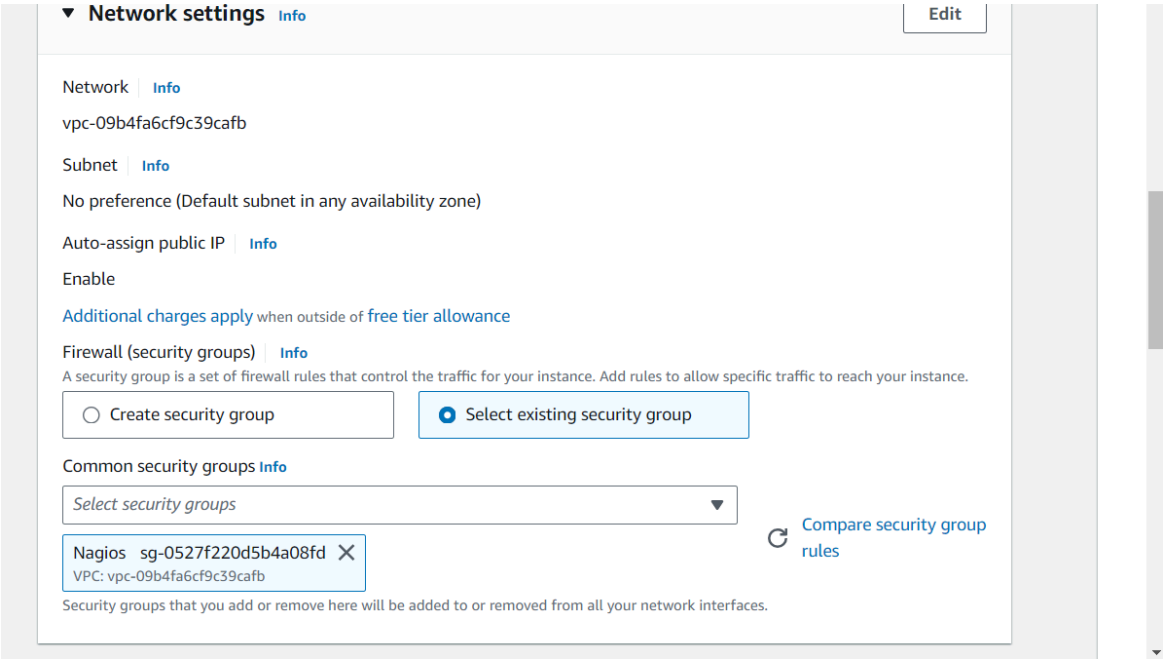
Step 2: Now Create a new EC2 instance. Name: Nagios-client, AMI: Ubuntu Instance Type: t2.micro.



For Key pair : Click on create key and make key of type RSA with extension .pem . Key will be downloaded to your local machine.



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



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

```
PS C:\Users\Vedant> ssh -i "devops.pem" ubuntu@ec2-3-81-218-241.compute-1.amazonaws.com
The authenticity of host 'ec2-3-81-218-241.compute-1.amazonaws.com (3.81.218.241)' can't be establish
ed.
ED25519 key fingerprint is SHA256:7YtdUbwcFY6vK575h5DIfKqnlOf22OVC34blKsmOQcw.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-81-218-241.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 Fri Sep 27 08:38:26 UTC 2024

  System load:    1.36      Processes:             26
  Usage of /home: unknown   Users logged in:       0
  Memory usage:   4%        IPv4 address for eth0: 10.10.10.2
  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.
```

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

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

ps -ef | grep nagios

```
[ec2-user@ip-172-31-46-196 ~]$ ps -ef | grep nagios
nagios      2428       1  0 11:05 ?        00:00:00 /usr/local/nagios/bin/nagios -d /usr/local/nagio
/etc/nagios.cfg
nagios      2430    2428  0 11:05 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local
nagios/var/rw/nagios.qh
nagios      2431    2428  0 11:05 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local
nagios/var/rw/nagios.qh
nagios      2432    2428  0 11:05 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local
nagios/var/rw/nagios.qh
nagios      2433    2428  0 11:05 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local
nagios/var/rw/nagios.qh
nagios      2437    2428  0 11:05 ?        00:00:00 /usr/local/nagios/bin/nagios -d /usr/local/nagio
/etc/nagios.cfg
ec2-user    3367    3273  0 11:16 pts/0    00:00:00 grep --color=auto nagios
[ec2-user@ip-172-31-46-196 ~]$
```

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

sudo su

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

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

```
[ec2-user@ip-172-31-46-196 ~]$ sudo su
[root@ip-172-31-46-196 ec2-user]# mkdir /usr/local/nagios/etc/objects/monitorhosts
[root@ip-172-31-46-196 ec2-user]# mkdir /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
[root@ip-172-31-46-196 ec2-user]#
```

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

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

```
[root@ip-172-31-46-196 ec2-user]# cp /usr/local/nagios/etc/objects/localhost.cfg /usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg
[root@ip-172-31-46-196 ec2-user]#
```

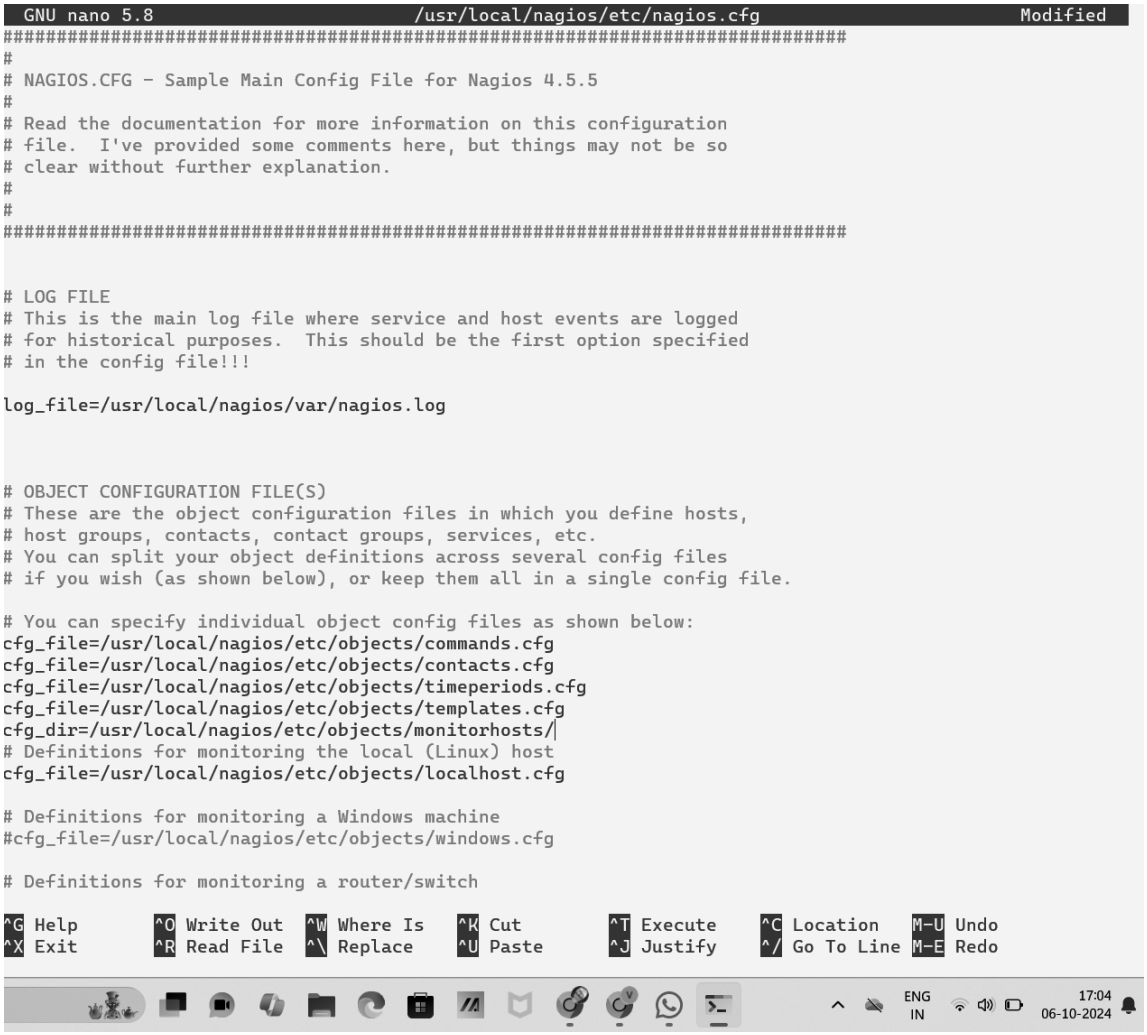
Step 7: Open linuxserver.cfg using nano and make the following changes in all positions?everywhere in file.

Change hostname to **linuxserver**.
Change address to the public IP of your Linux client.
Set hostgroup_name to **linux-servers1**.

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

Step 8: Now update the Nagios config file .Add the following line in the file.

Line to add : cfg_dir=/usr/local/nagios/etc/objects/monitorhosts/
Run the command : **nano /usr/local/nagios/etc/nagios.cfg**



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

```
[root@ip-172-31-46-196 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...
Warning: Duplicate definition found for service 'HTTP' on host 'localhost' (config file '/usr/local/nagios
fg', starting on line 152)
Warning: Duplicate definition found for service 'SSH' on host 'localhost' (config file '/usr/local/nagios/
g', starting on line 138)
Warning: Duplicate definition found for service 'Swap Usage' on host 'localhost' (config file '/usr/local/
rver.cfg', starting on line 125)
Warning: Duplicate definition found for service 'Current Load' on host 'localhost' (config file '/usr/loca
server.cfg', starting on line 112)
Warning: Duplicate definition found for service 'Total Processes' on host 'localhost' (config file '/usr/l
nuxserver.cfg', starting on line 100)
Warning: Duplicate definition found for service 'Current Users' on host 'localhost' (config file '/usr/loc
xserver.cfg', starting on line 86)
Warning: Duplicate definition found for service 'Root Partition' on host 'localhost' (config file '/usr/lo
uxserver.cfg', starting on line 72)
Warning: Duplicate definition found for service 'PING' on host 'localhost' (config file '/usr/local/nagios
fg', starting on line 58)
   Read object config files okay...

Running pre-flight check on configuration data...

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

Total Warnings: 0
Total Errors:   0

[root@ip-172-31-46-196 ec2-user]#
```

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

service nagios restart

```
[root@ip-172-31-46-196 ec2-user]# service nagios restart
Redirecting to /bin/systemctl restart nagios.servicetart
[root@ip-172-31-46-196 ec2-user]#
```

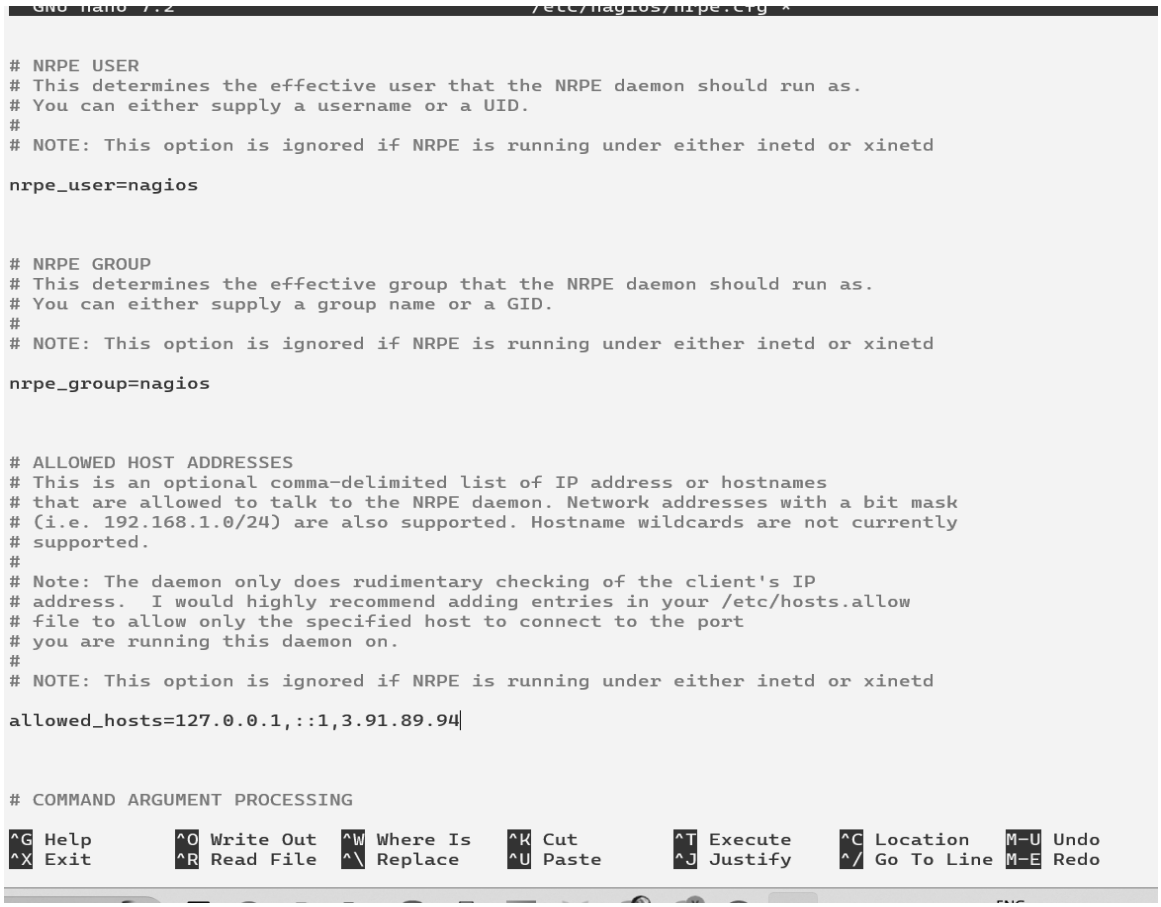
Step 11: Now Go to the Nagios-client ssh terminal and update and install the packages by running the following command.

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

```
ubuntu@ip-172-31-37-150:~$ sudo apt update -y
sudo apt install gcc -y
sudo apt install -y nagios-nrpe-server nagios-plugins
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:6 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [382 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:8 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [83.9 kB]
Get:9 http://security.ubuntu.com/ubuntu noble-security/main amd64 c-n-f Metadata [4704 B]
Get:10 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [277 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:12 http://security.ubuntu.com/ubuntu noble-security/universe Translation-en [117 kB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
Get:16 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [8632 B]
Get:17 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Components [35.0 kB]
Get:18 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 c-n-f Metadata [8328 B]
Get:19 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [537 kB]
Get:20 http://security.ubuntu.com/ubuntu noble-security/universe amd64 c-n-f Metadata [10.4 kB]
Get:21 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Packages [10.9 kB]
```

```
Setting up python3-ldb (2:2.8.0+samba4.19.5+dfsg-4ubuntu9) ...
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-172-31-37-150:~$
```

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



Step 13: Now restart the NRPE server by this command.
sudo systemctl restart nagios-nrpe-server

```
ubuntu@ip-172-31-37-150:~$ sudo systemctl restart nagios-nrpe-server
ubuntu@ip-172-31-37-150:~$
```

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

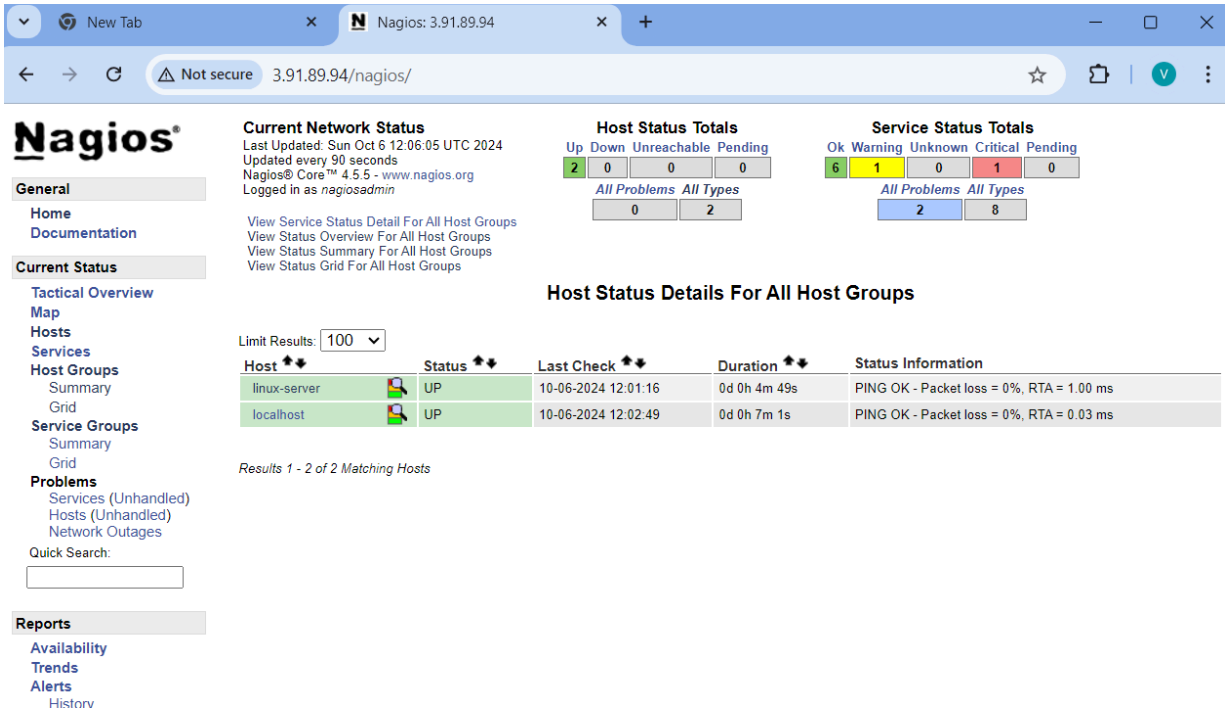
```
[root@ip-172-31-46-196 ec2-user]# sudo systemctl status nagios
● nagios.service - Nagios Core 4.5.5
     Loaded: loaded (/usr/lib/systemd/system/nagios.service; disabled; preset: disabled)
     Active: active (running) since Sun 2024-10-06 11:37:16 UTC; 9min ago
       Docs: https://www.nagios.org/documentation
    Process: 4481 ExecStartPre=/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg (code=exited, st>
    Process: 4482 ExecStart=/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg (code=exited, statu>
   Main PID: 4488 (nagios)
      Tasks: 6 (limit: 1112)
     Memory: 4.1M
        CPU: 108ms
     CGroup: /system.slice/nagios.service
             ├─4488 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─4489 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─4490 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─4491 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─4492 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             └─4497 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg

Oct 06 11:37:16 ip-172-31-46-196.ec2.internal nagios[4488]: HOST ALERT: linux-server;DOWN;SOFT;1;(No output on s>
Oct 06 11:38:16 ip-172-31-46-196.ec2.internal nagios[4488]: HOST ALERT: linux-server;DOWN;SOFT;2;(No output on s>
Oct 06 11:39:16 ip-172-31-46-196.ec2.internal nagios[4488]: HOST ALERT: linux-server;DOWN;SOFT;3;(No output on s>
Oct 06 11:40:16 ip-172-31-46-196.ec2.internal nagios[4488]: HOST ALERT: linux-server;DOWN;SOFT;4;(No output on s>
Oct 06 11:41:16 ip-172-31-46-196.ec2.internal nagios[4488]: HOST ALERT: linux-server;DOWN;SOFT;5;(No output on s>
Oct 06 11:42:16 ip-172-31-46-196.ec2.internal nagios[4488]: HOST ALERT: linux-server;DOWN;SOFT;6;(No output on s>
Oct 06 11:43:16 ip-172-31-46-196.ec2.internal nagios[4488]: HOST ALERT: linux-server;DOWN;SOFT;7;(No output on s>
Oct 06 11:44:16 ip-172-31-46-196.ec2.internal nagios[4488]: HOST ALERT: linux-server;DOWN;SOFT;8;(No output on s>
Oct 06 11:45:16 ip-172-31-46-196.ec2.internal nagios[4488]: HOST ALERT: linux-server;DOWN;SOFT;9;(No output on s>
Oct 06 11:46:16 ip-172-31-46-196.ec2.internal nagios[4488]: HOST ALERT: linux-server;DOWN;HARD;10;(No output on >
lines 1-28/28 (END)
```

sudo systemctl status httpd
sudo systemctl start httpd
sudo systemctl enable httpd

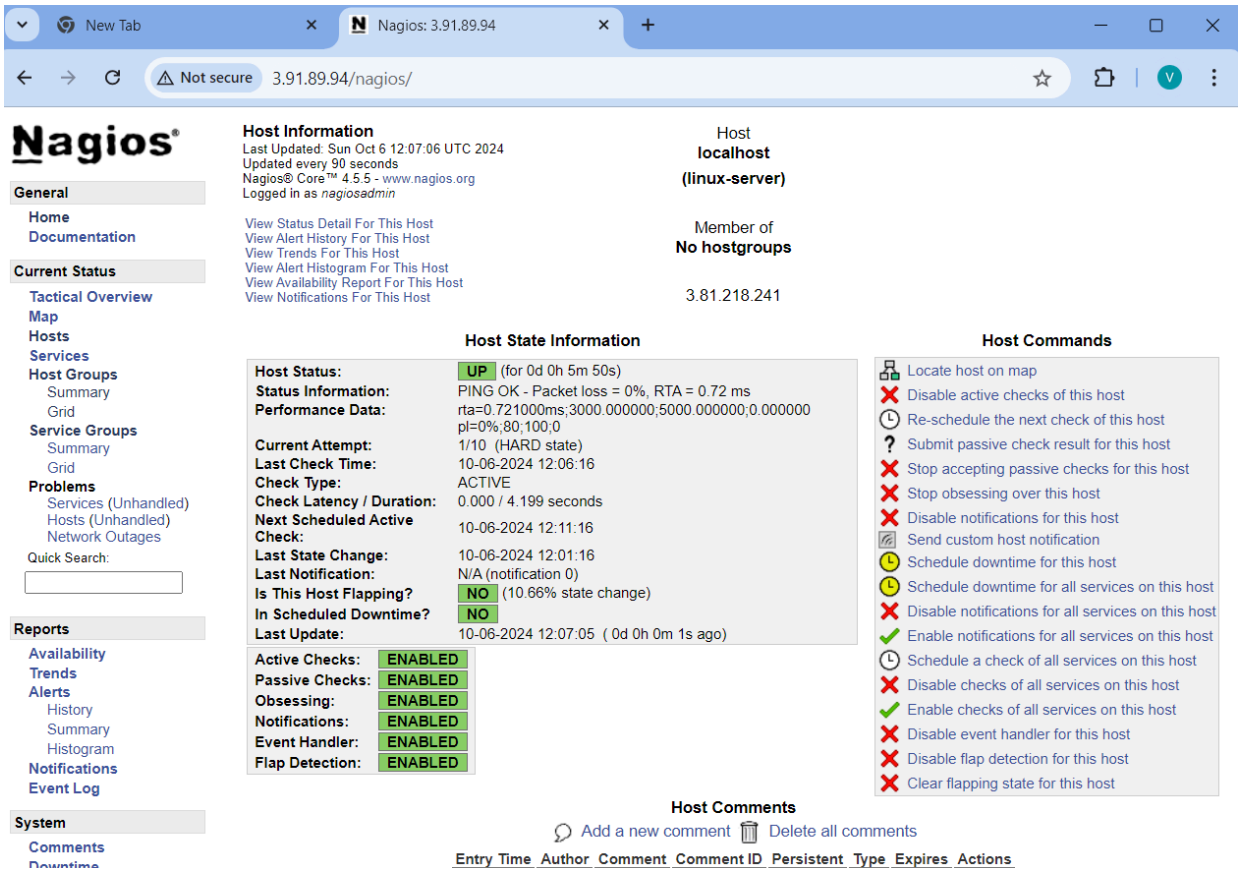
```
[root@ip-172-31-46-196 ec2-user]# sudo systemctl status httpd
● httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset: disabled)
    Drop-In: /usr/lib/systemd/system/httpd.service.d
             └─php-fpm.conf
     Active: active (running) since Sun 2024-10-06 11:08:08 UTC; 42min ago
       Docs: man:httpd.service(8)
   Main PID: 2546 (httpd)
     Status: "Total requests: 48; Idle/Busy workers 100/0;Requests/sec: 0.0188; Bytes served/sec: 121 B/sec"
      Tasks: 230 (limit: 1112)
     Memory: 25.1M
        CPU: 1.834s
     CGroup: /system.slice/httpd.service
             ├─2546 /usr/sbin/httpd -DFOREGROUND
             ├─2548 /usr/sbin/httpd -DFOREGROUND
             ├─2554 /usr/sbin/httpd -DFOREGROUND
             ├─2555 /usr/sbin/httpd -DFOREGROUND
             ├─2556 /usr/sbin/httpd -DFOREGROUND
             └─2889 /usr/sbin/httpd -DFOREGROUND

Oct 06 11:08:07 ip-172-31-46-196.ec2.internal systemd[1]: Starting httpd.service - The Apache HTTP Server...
Oct 06 11:08:08 ip-172-31-46-196.ec2.internal systemd[1]: Started httpd.service - The Apache HTTP Server.
Oct 06 11:08:08 ip-172-31-46-196.ec2.internal httpd[2546]: Server configured, listening on: port 80
[root@ip-172-31-46-196 ec2-user]# sudo systemctl start httpd
[root@ip-172-31-46-196 ec2-user]# sudo systemctl enable httpd
[root@ip-172-31-46-196 ec2-user]#
```

Step 15: Now to check Nagios dashboard go to http://<Nagios-host ip>/nagios .
Now Click on Hosts from left side panel

Name:Shejal Tiwari Class/Roll No: D15C/ 57

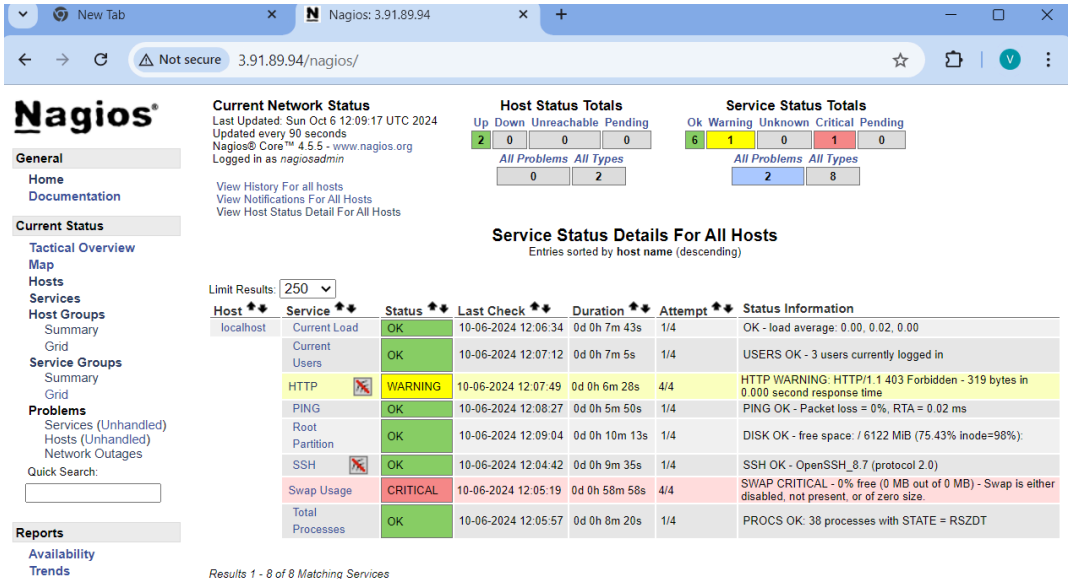


We can see our linuxserver now click on it we can see the host information.



Name:Shejal Tiwari Class/Roll No: D15C/ 57

Current Network Status



Conclusion:

In this experiment, we successfully implemented port, service, and Windows/Linux server monitoring using Nagios, but encountered a few challenges.

- **Configuration Issues:** Setting up monitoring hosts and editing files like linuxserver.cfg led to some errors in file paths and syntax, which required careful review.
- **NRPE Setup:** Configuring NRPE for remote monitoring was tricky due to firewall and permission issues, often causing connectivity problems between the Nagios host and clients.
- **Service Restarts:** Restarting Nagios and NRPE to apply changes didn't always work smoothly, with misconfigurations requiring troubleshooting.
- **Dashboard Access:** Accessing the Nagios dashboard was hindered by incorrect AWS security group rules, needing adjustments to allow proper HTTP and TCP traffic.

Experiment No: 11

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

Theory:

AWS Lambda

A fully managed, serverless computing service where you run code without provisioning or managing servers. Lambda automatically scales your application based on the number of incoming requests or events, ensuring efficient resource utilization. You are only charged for the time your code is running, with no upfront cost, making it cost-effective for on-demand workloads.

Lambda Workflow

- **Create a Function**: Write the function code and define its handler (entry point). You can use the AWS Console, CLI, or upload a deployment package.
- **Set Event Sources**: Define how the function is triggered (e.g., when an object is uploaded to S3 or a DynamoDB table is updated).
- **Execution**: When triggered, Lambda runs your function, executes the logic, and automatically scales to handle the incoming event volume.
- **Scaling and Concurrency**: Lambda scales automatically by launching more instances of the function to handle simultaneous invocations. There are also options for configuring **reserved concurrency** to manage traffic.
- **Monitoring and Logging**: Lambda integrates with Amazon CloudWatch for logging and monitoring. Logs for each invocation are sent to CloudWatch, allowing you to track performance and troubleshoot errors.

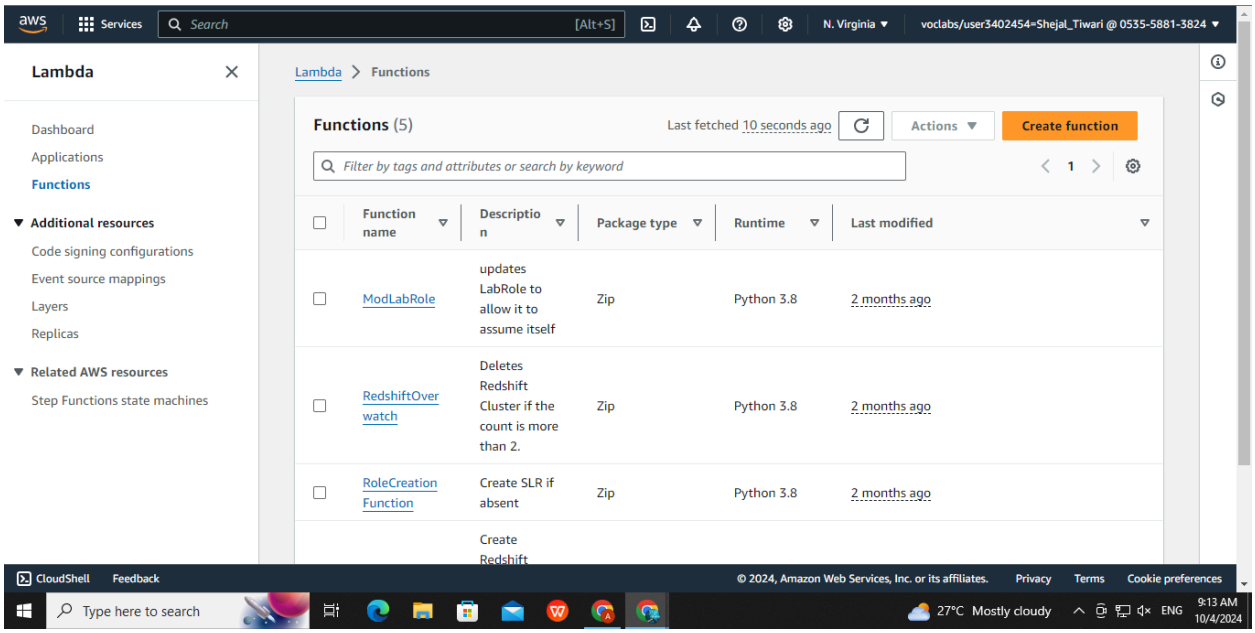
AWS Lambda Functions

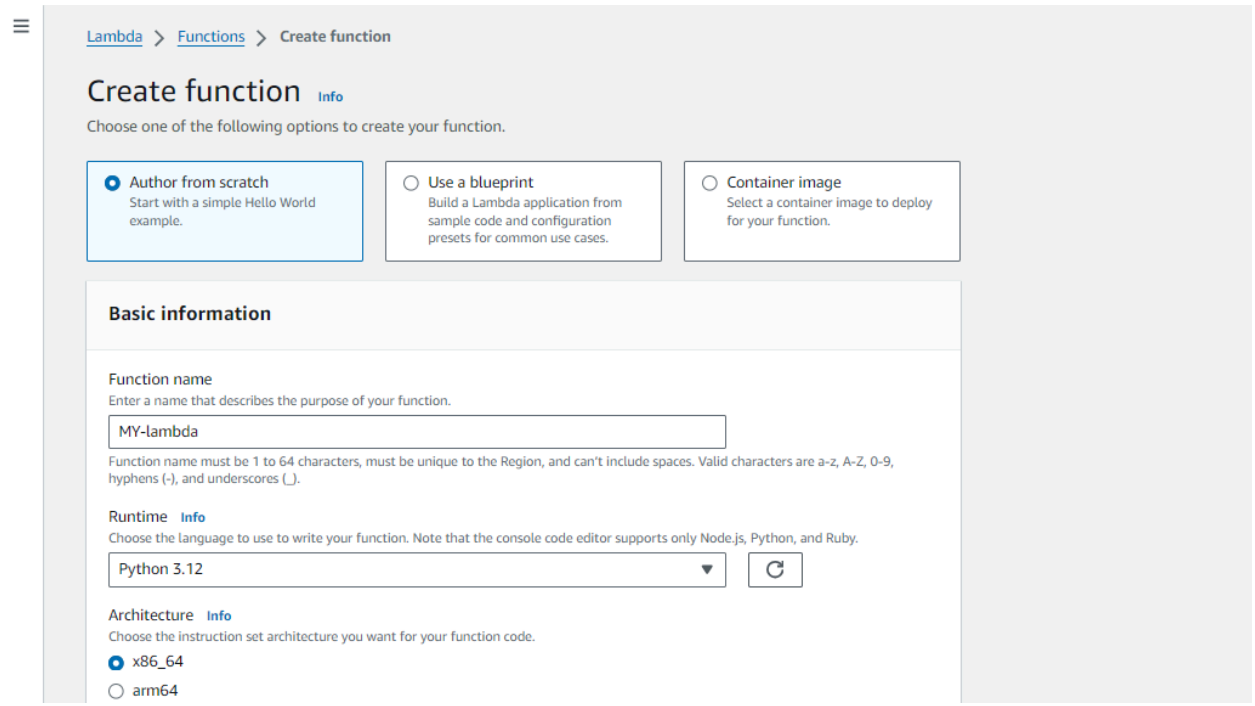
- **Python**: Great for quick development with its rich standard library and support for lightweight tasks.
- **Java**: Typically used for more complex, compute-intensive tasks. While it's robust, cold start times can be higher.
- **Node.js**: Excellent for I/O-bound tasks like handling APIs or streaming data, with fast startup times and efficient memory usage.

Steps To create the lambda function:

Step 1: Login to your AWS Personal/Academy Accout.Open lambda and click on create function button.

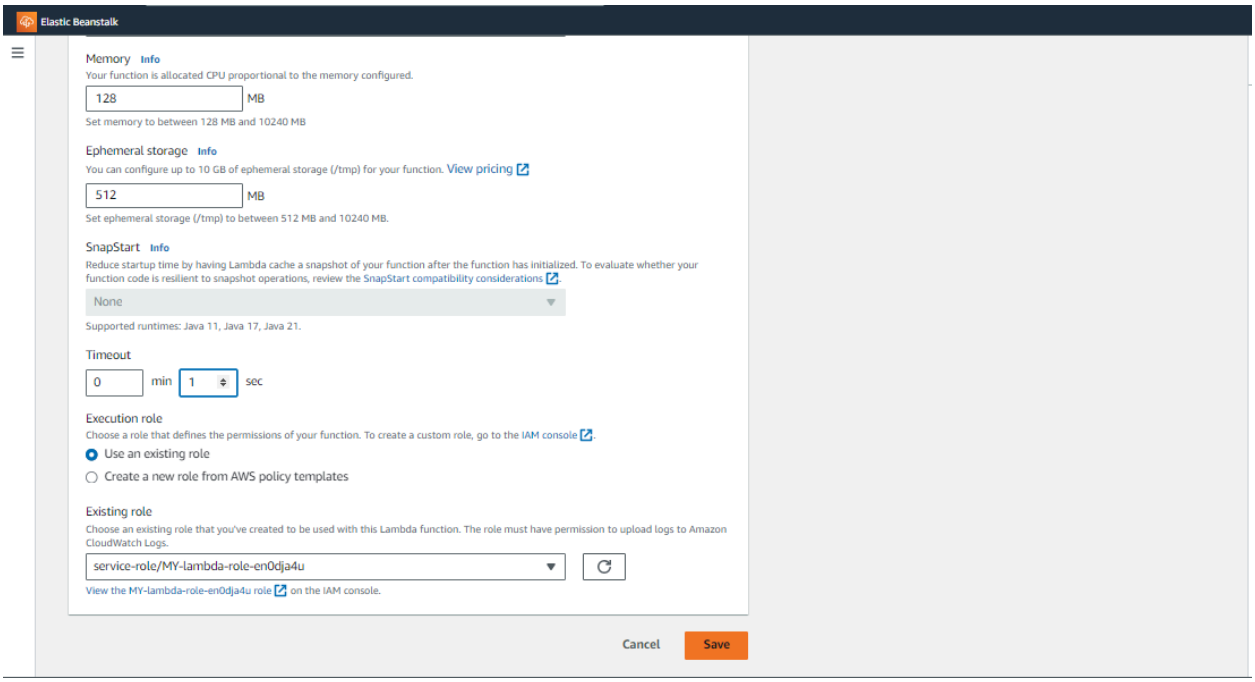
Step 2: Now Give a name to your Lambda function, Select the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby. So will select Python 3.12 , Architecture as x86, and Execution role to Create a new role with basic Lambda permissions.





So See or Edit the basic settings go to configuration then click on edit general setting.

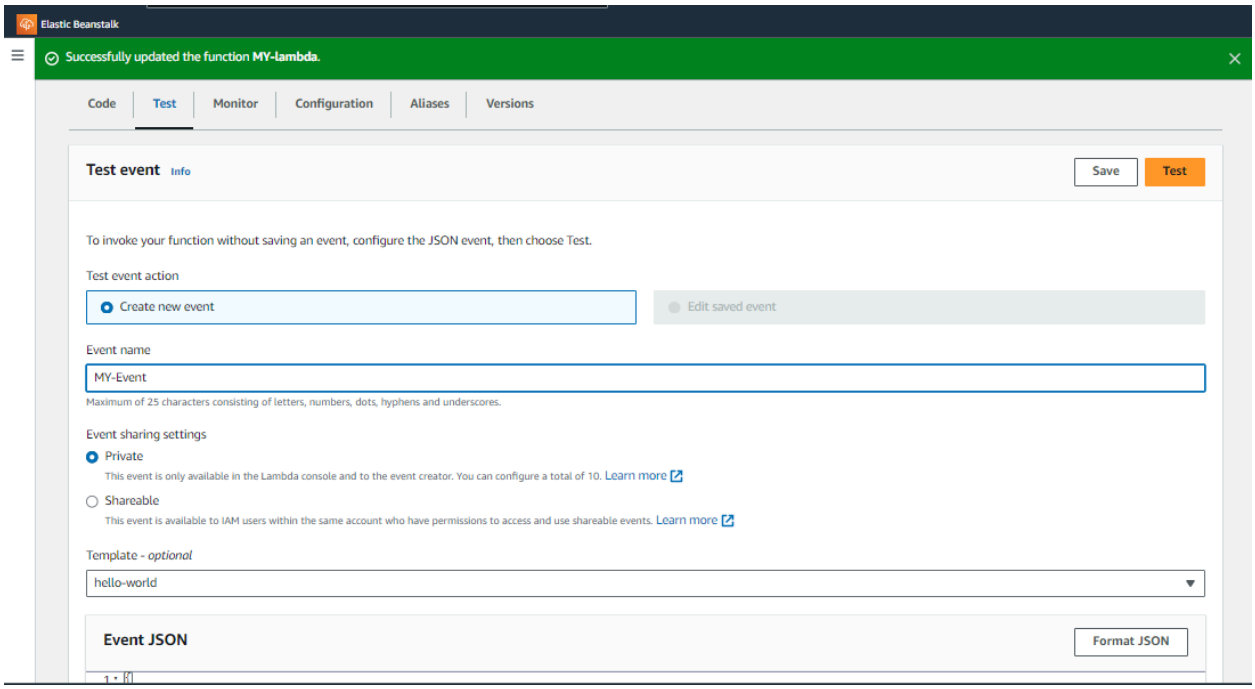
Here, you can enter a description and change Memory and Timeout. I've changed the Timeout period to 1 sec since that is sufficient for now.

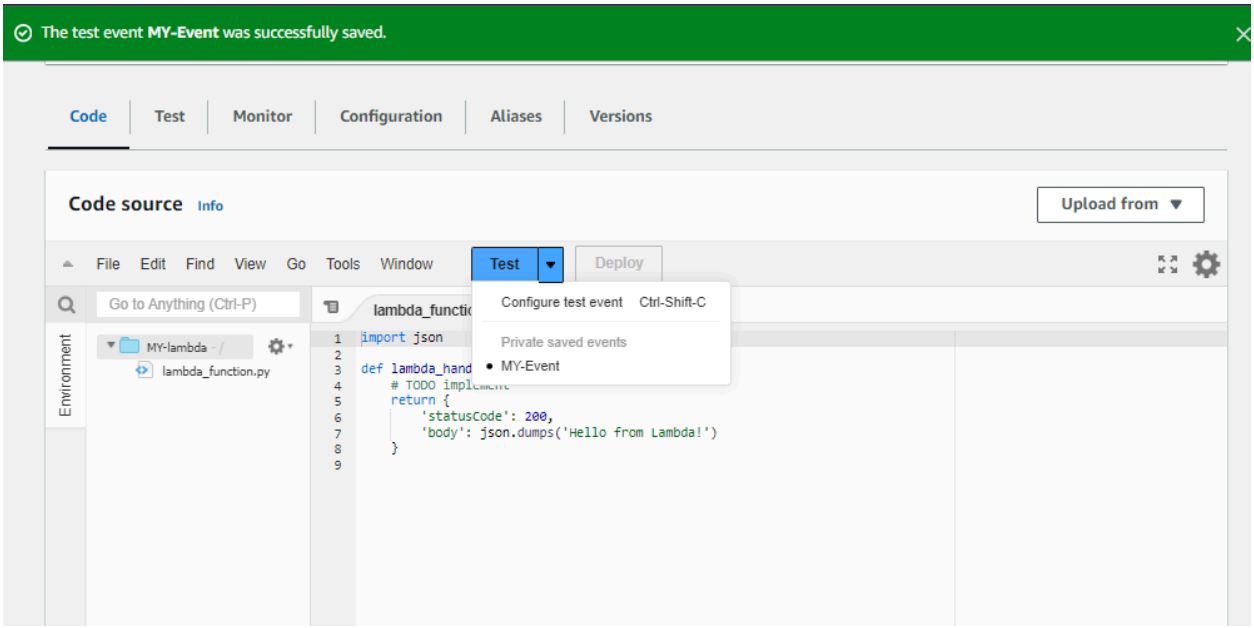


Step 3: Now Click on the Test tab then select Create a new event, give a name to the event and select Event Sharing to private, and select hello-world template.

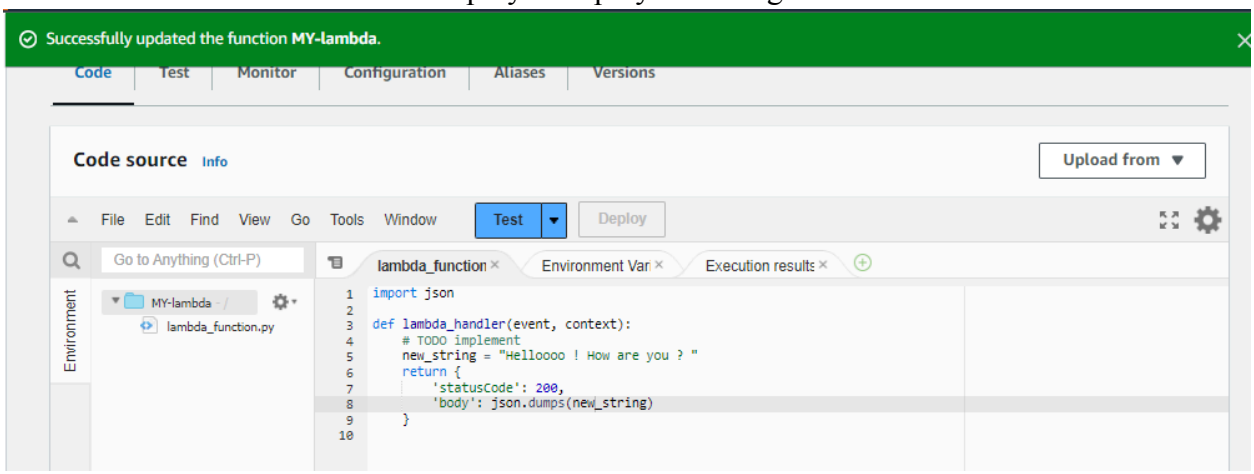
Step 4: Now In Code section select the created event from the dropdown of test then click on test . You will see the below output.

Name: Shejal Tiwari Div/Roll No: D15C/57

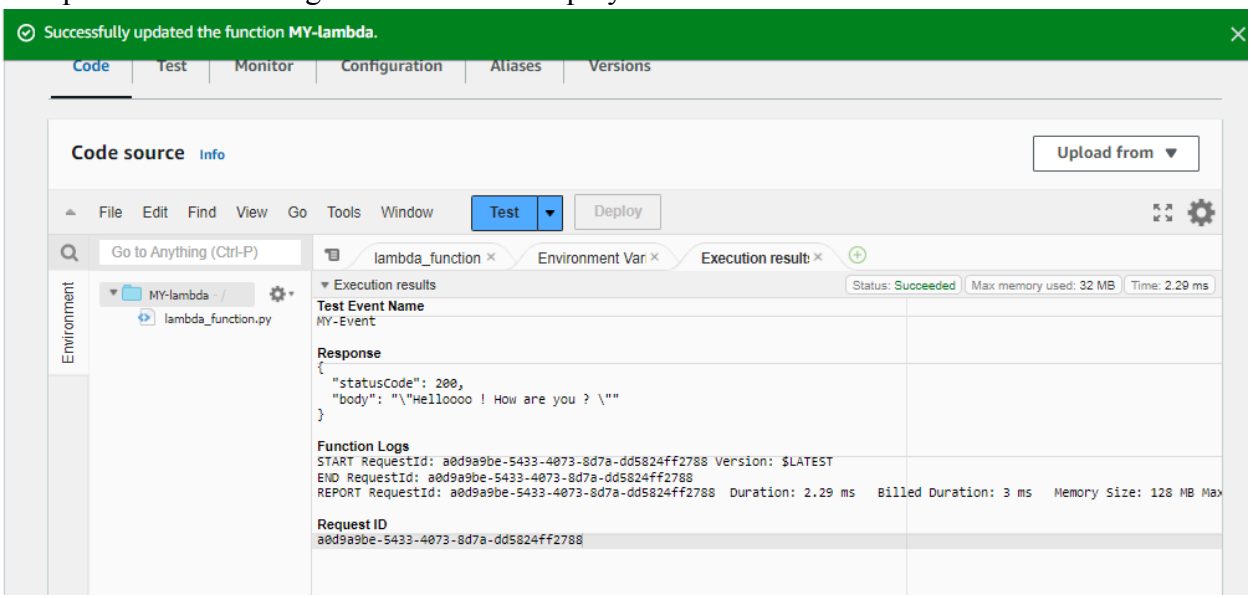




Step 5: You can edit your lambda function code. I have changed the code to display the new String. Now ctrl+s to save and click on deploy to deploy the changes.



Step 6: Now click on the test and observe the output. We can see the status code 200 and your string output and function logs. On successful deployment.



Conclusion:

In this experiment, we successfully created and deployed our first AWS Lambda function using Python, gaining an understanding of its workflow and capabilities. The function was executed and tested, allowing us to observe the output and logs. While the overall process was smooth, there were several challenges that we encountered:

- **Role and Permissions Configuration:** Setting up the correct execution role with the necessary

permissions was critical, as misconfigurations could prevent the function from running or accessing other AWS services. Debugging permission issues was time-consuming, especially when using new roles.

- **Timeout Issues:** Initially, the default timeout setting was insufficient for some operations. Adjusting the timeout to a value that suits the function's workload was necessary, especially for more complex operations beyond basic tasks.

- **Event Testing:** Configuring and testing events within Lambda required careful attention. Choosing the wrong template or incorrect event parameters often led to failures during the test phase, requiring adjustments to the event settings.

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

Theory:

AWS Lambda and S3 Integration:

AWS Lambda allows you to execute code in response to various events, including those triggered by Amazon S3. When an object is added to an S3 bucket, it can trigger a Lambda function to execute, allowing for event-driven processing without managing servers.

Workflow:

1. Create an S3 Bucket:

○ First, create an S3 bucket that will store the objects. This bucket will act as the trigger source for the Lambda function.

2. Create the Lambda Function:

○ Set up a new Lambda function using AWS Lambda’s console. You can choose a runtime environment like Python, Node.js, or Java.

○ Write code that logs a message like “An Image has been added” when triggered.

3. Set Up Permissions:

○ Ensure that the Lambda function has the necessary permissions to access S3. You can do this by attaching an IAM role with policies that allow reading from the bucket and writing logs to CloudWatch.

4. Configure S3 Trigger:

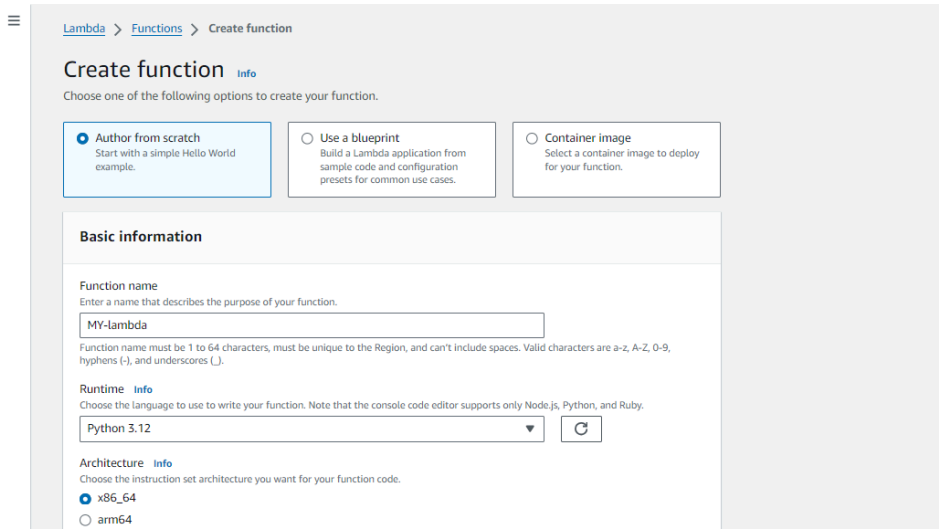
○ Link the S3 bucket to the Lambda function by setting up a trigger. Specify that the function should be triggered when an object is created in the bucket (e.g., when an image is uploaded).

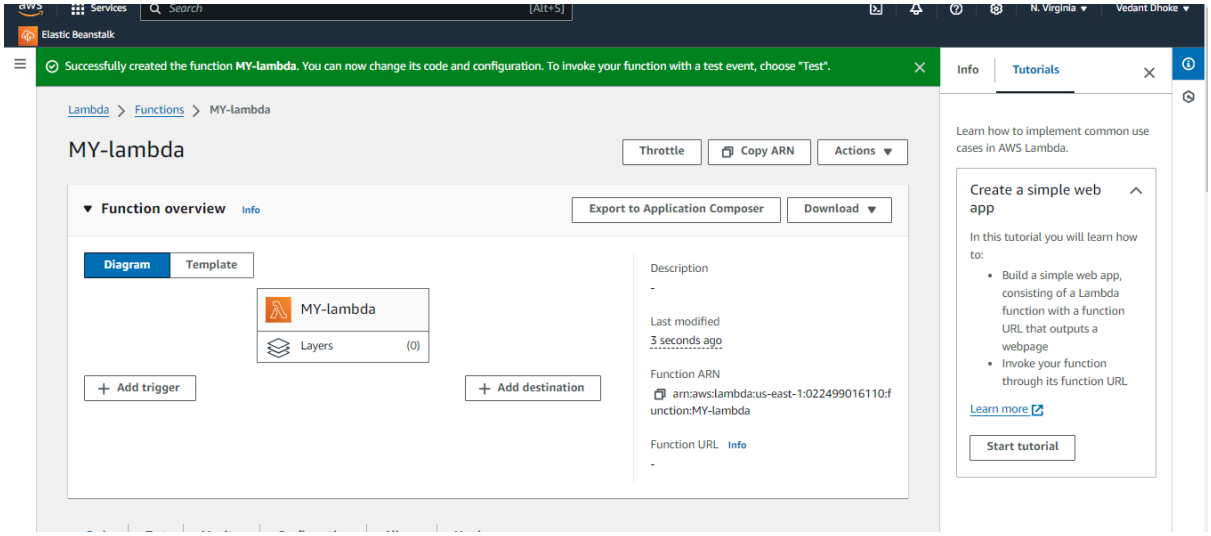
5. Test the Setup:

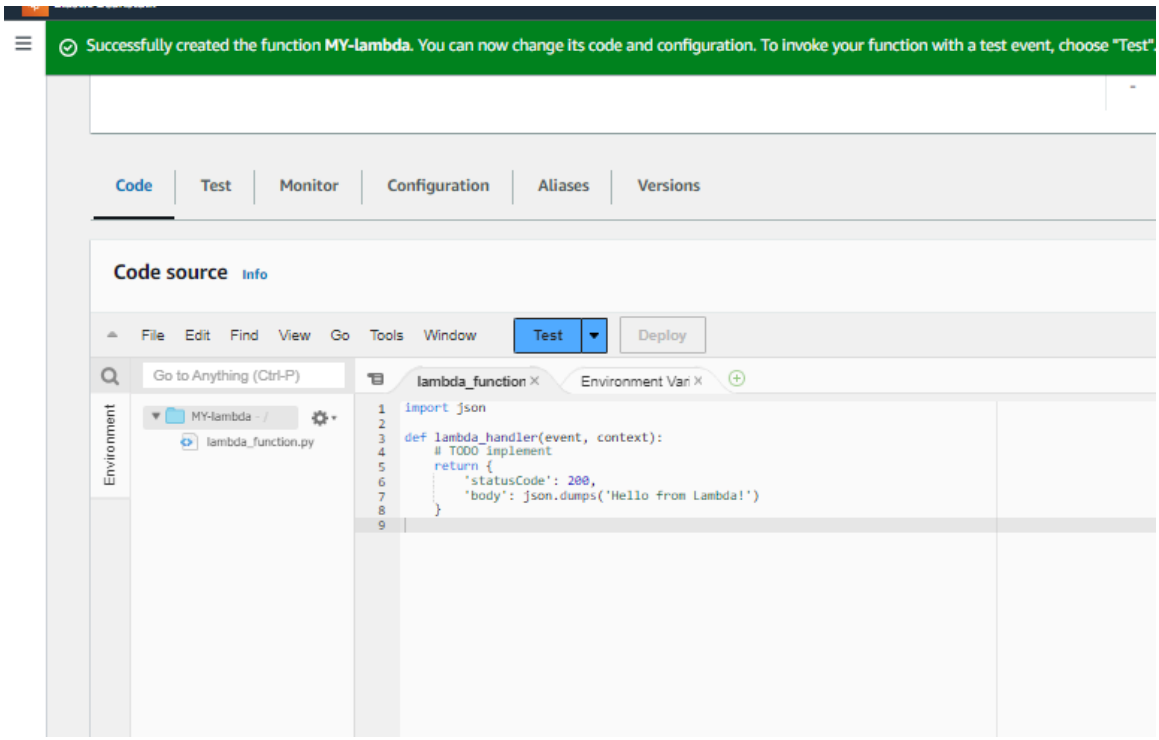
○ Upload an object (e.g., an image) to the S3 bucket to test the trigger. The Lambda function should execute and log the message “An Image has been added” in AWS CloudWatch Logs.

Steps To create the lambda function:

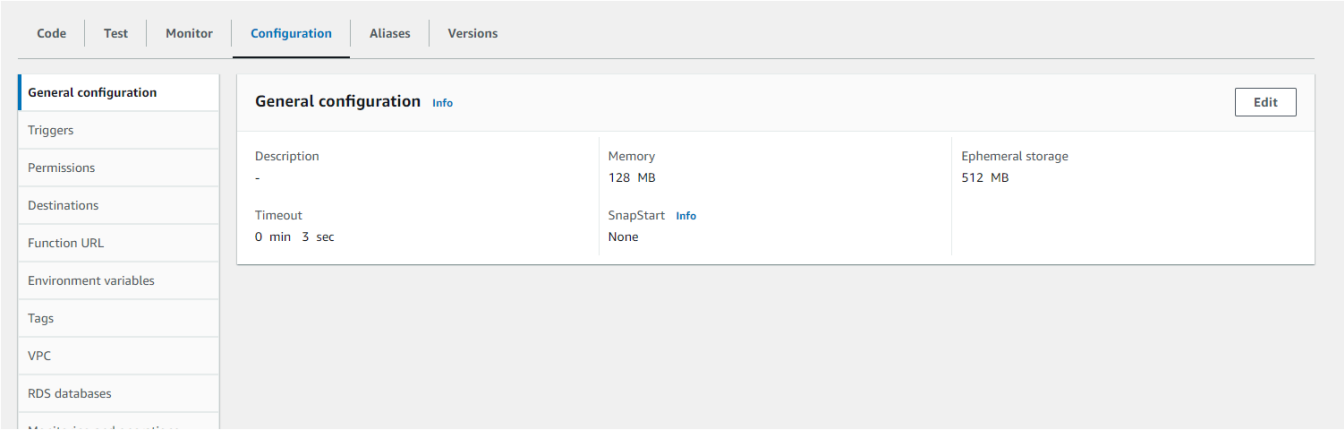
Step 1: Login to your AWS Personal account. Now open S3 from services and click on create S3 bucket.
Step 2: I have used already created bucket v2bucket if you dont have then you can create a basic bucket for this experiment .
Step 3: Open lambda console and click on create function button.
Step 4: Now Give a name to your Lambda function, Select the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby. So will select Python 3.12 , Architecture as x86, and Exceution role to Create a new role with basic Lambda permissions.



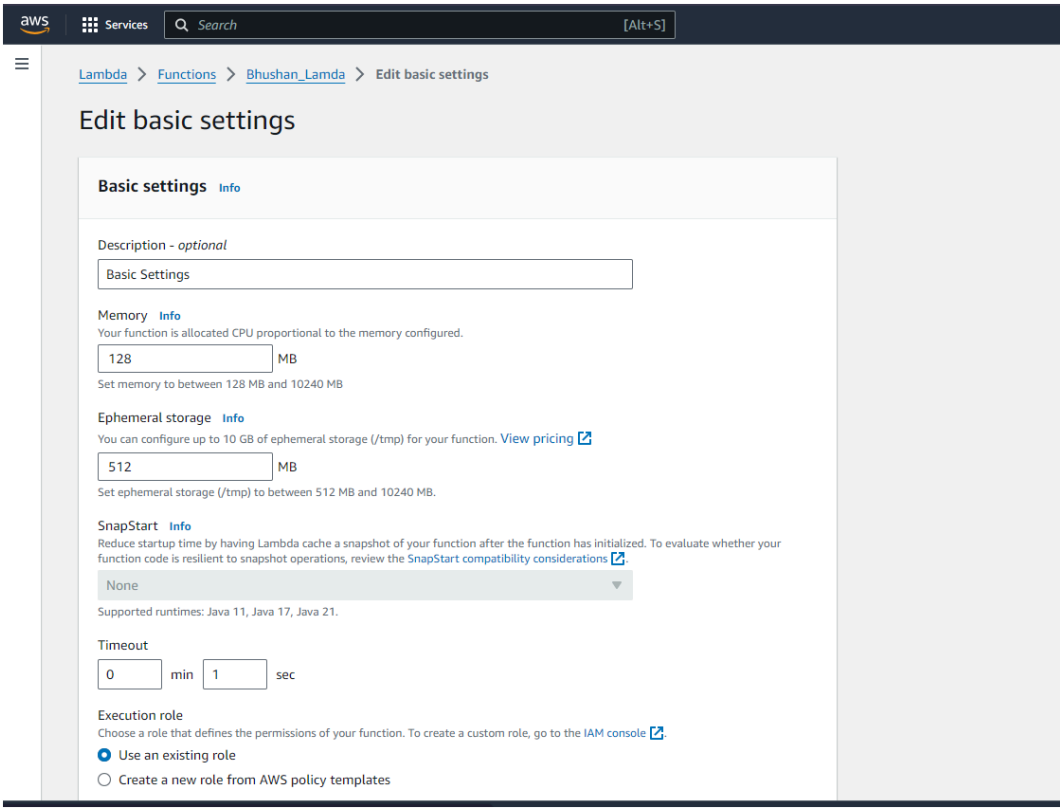




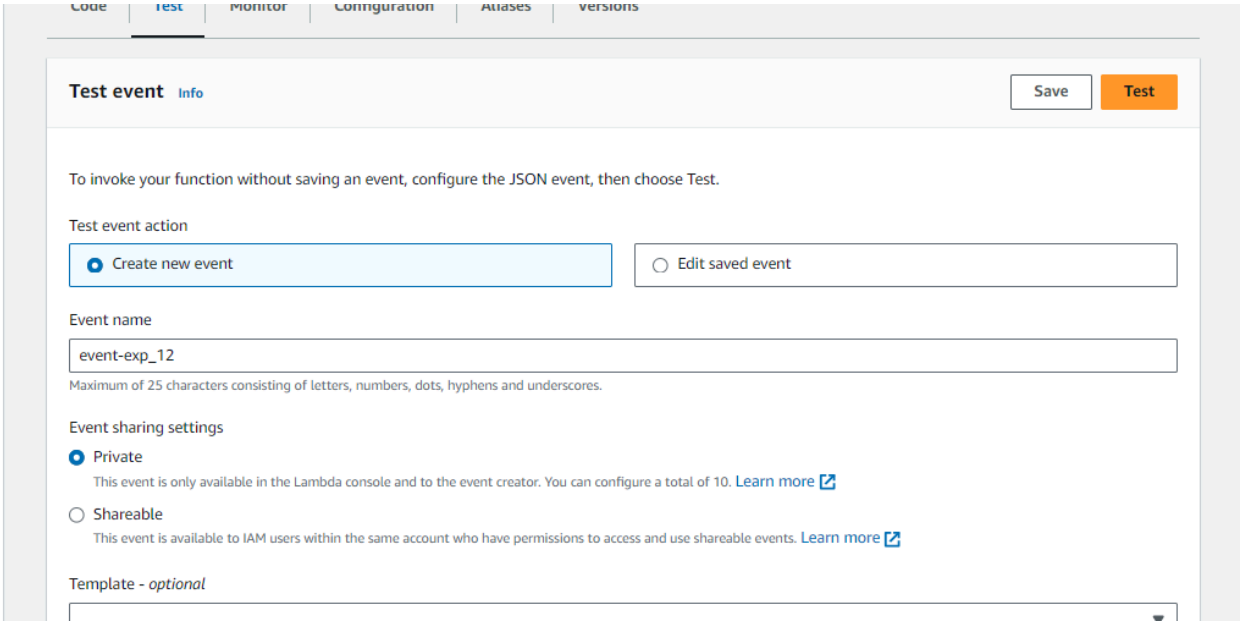
So See or Edit the basic settings go to configuration then click on edit general setting.

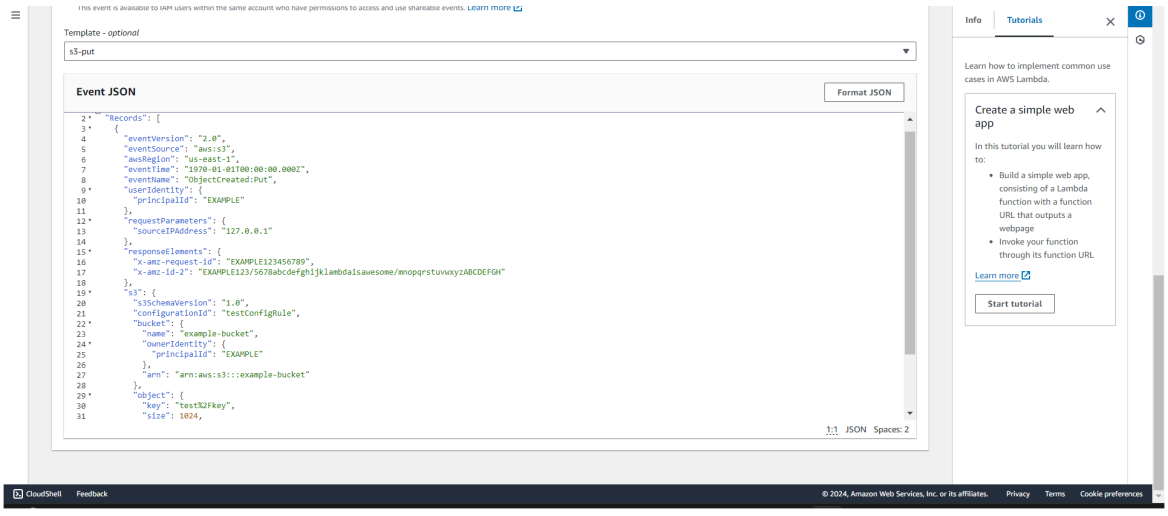


Here, you can enter a description and change Memory and Timeout. I've changed the Timeout period to 1 sec since that is sufficient for now.

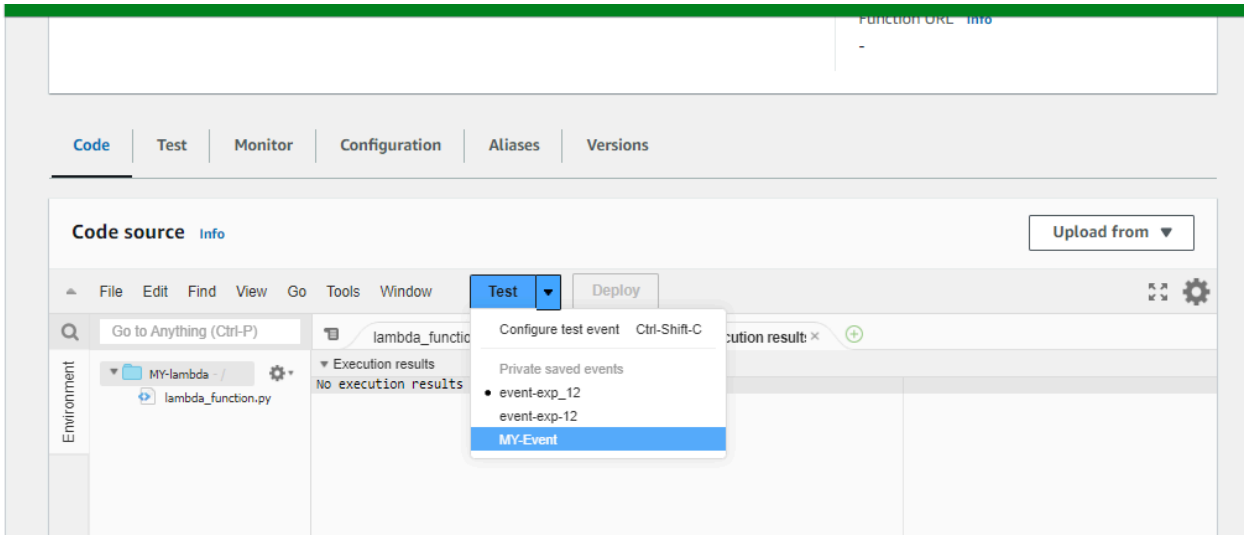


Step 5: Now Click on the Test tab then select Create a new event, give a name to the event and select Event Sharing to private, and select s3 put template.

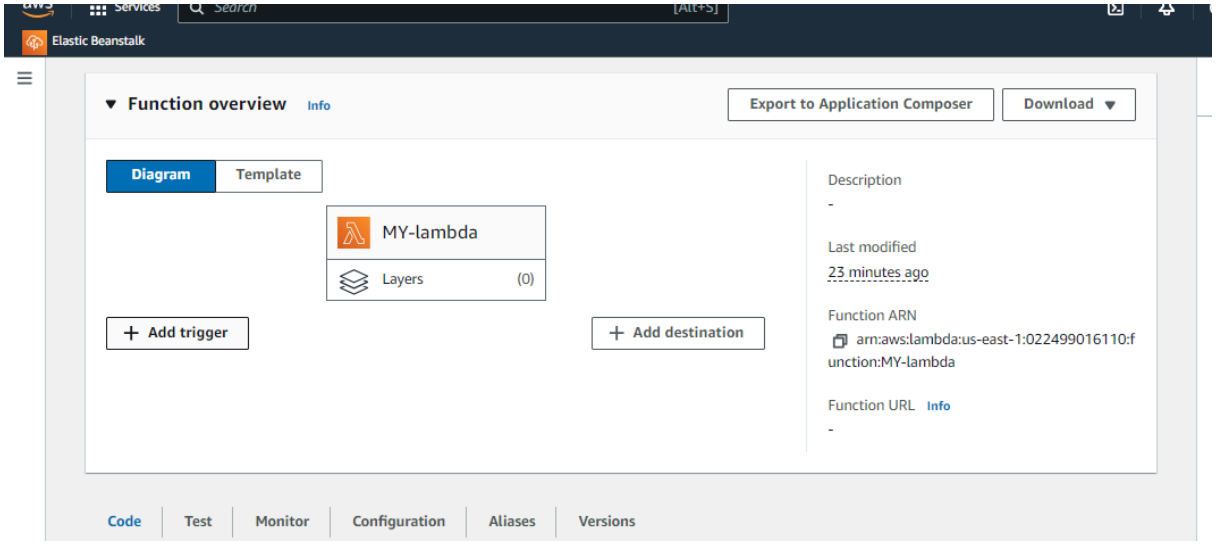


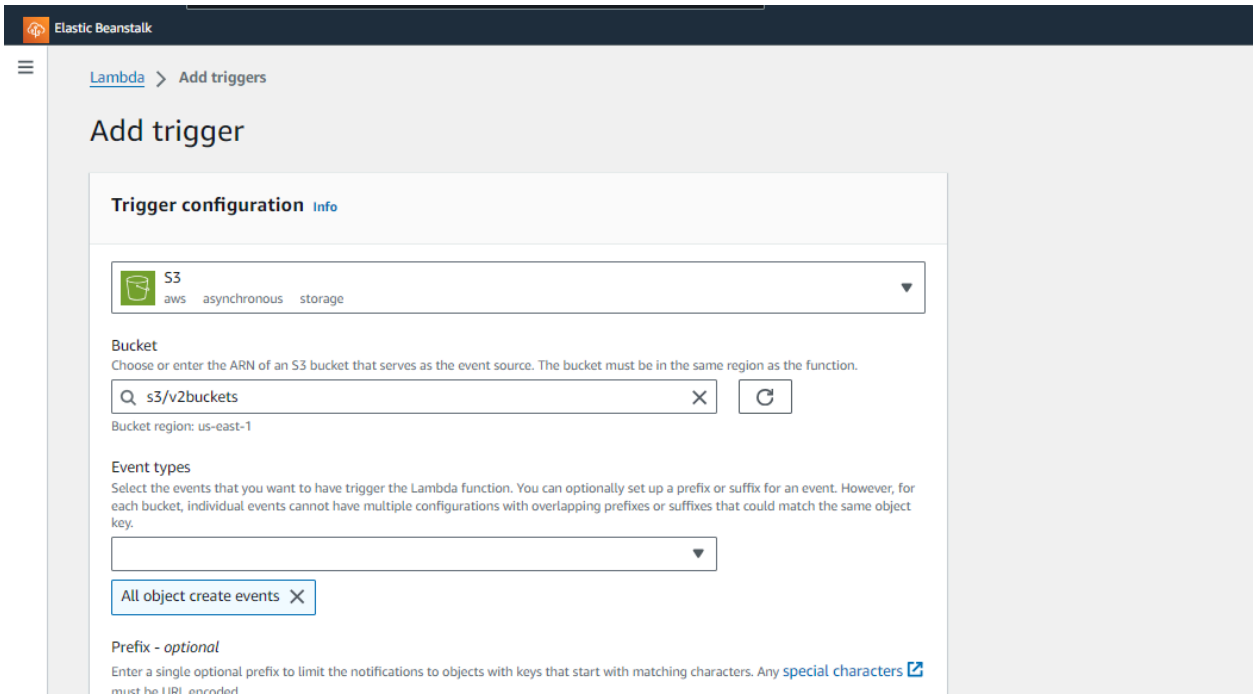


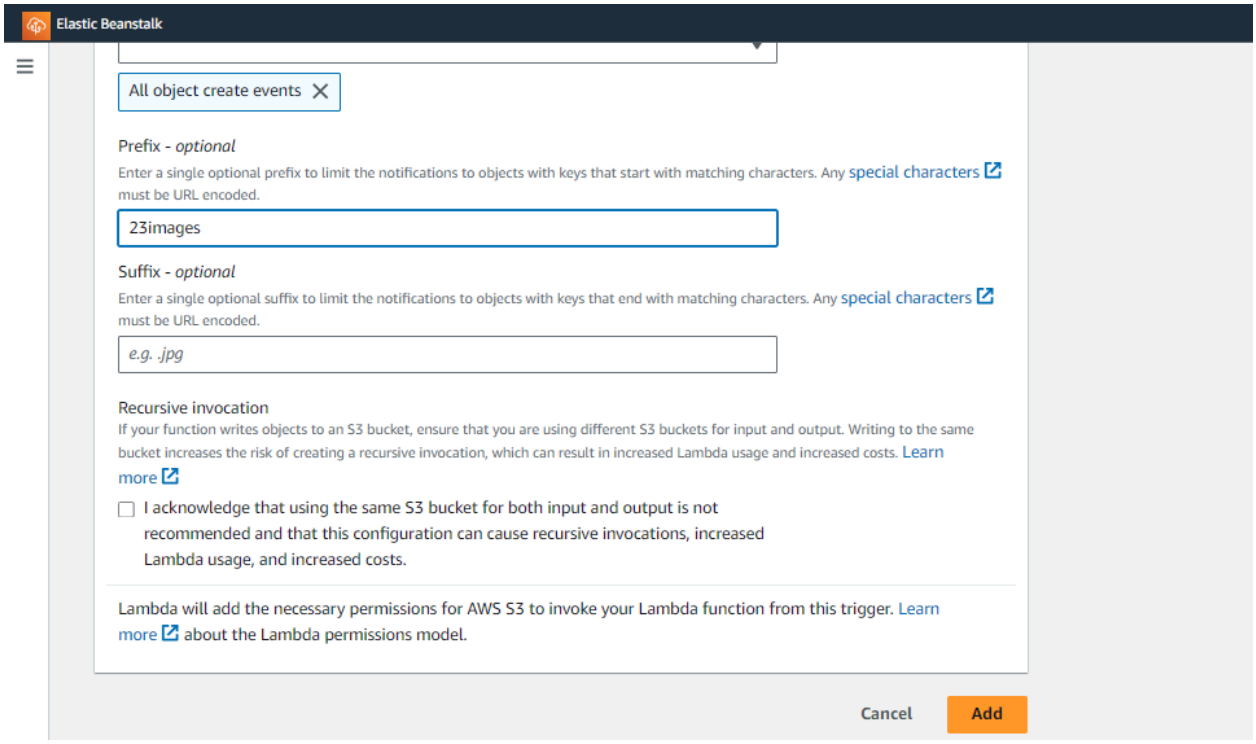
Step 6: Now In Code section select the created event from the dropdown .



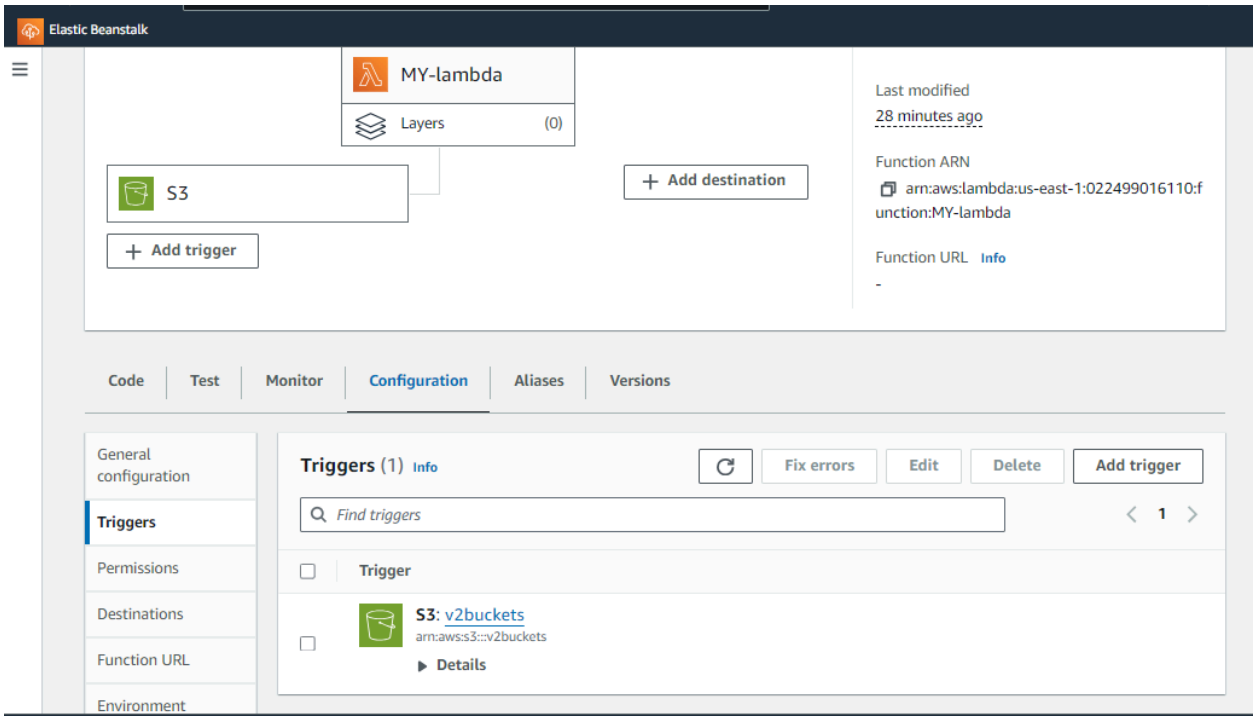
Step 7: Now In the Lambda function click on add tigger.
Now select the source as S3 then select the bucket name from the dropdown, keep other things to default and also you can add prefix to



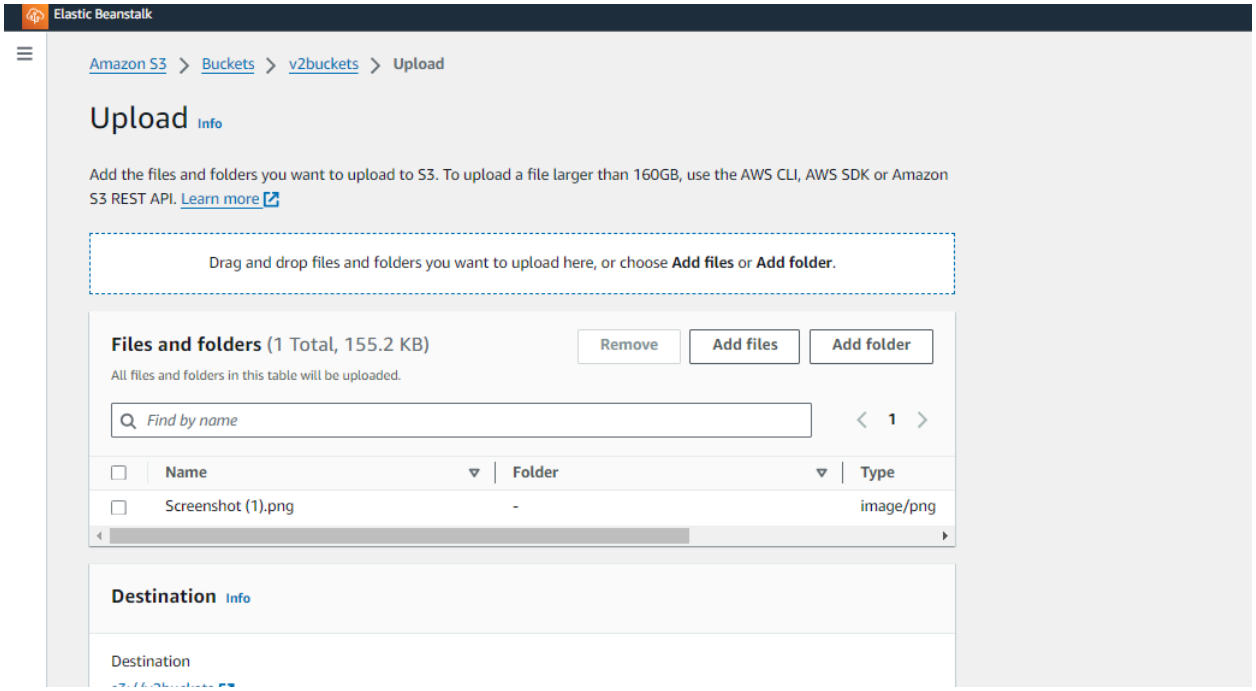




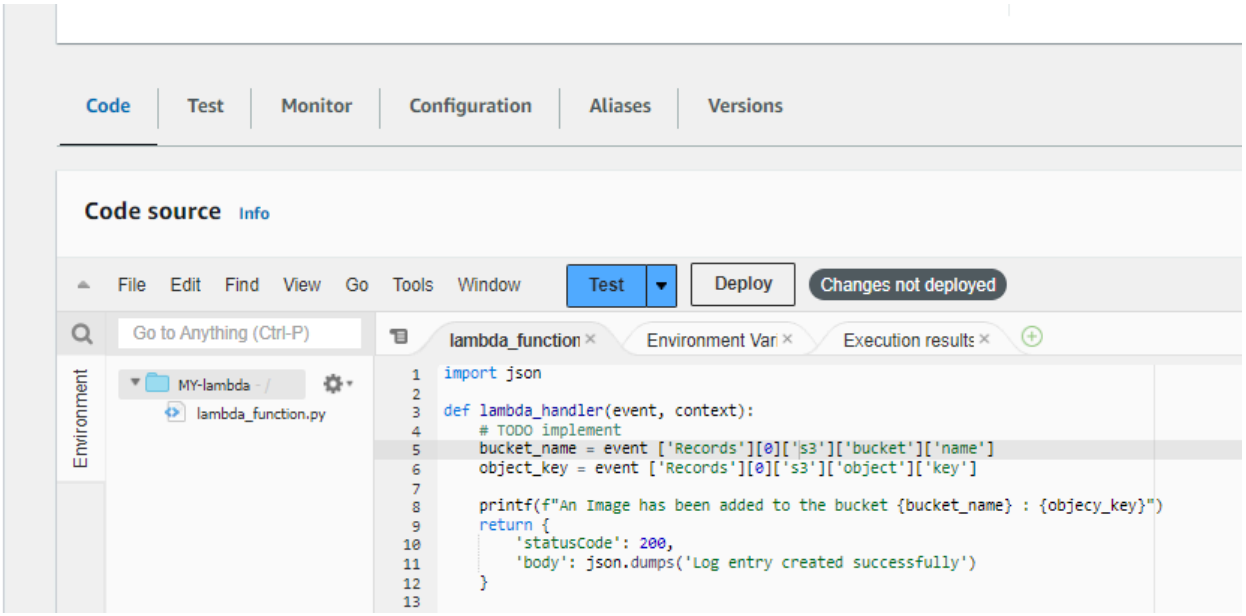
Step 8: Now Write code that logs a message like "An Image has been added" when triggered. Save the file and click on deploy.

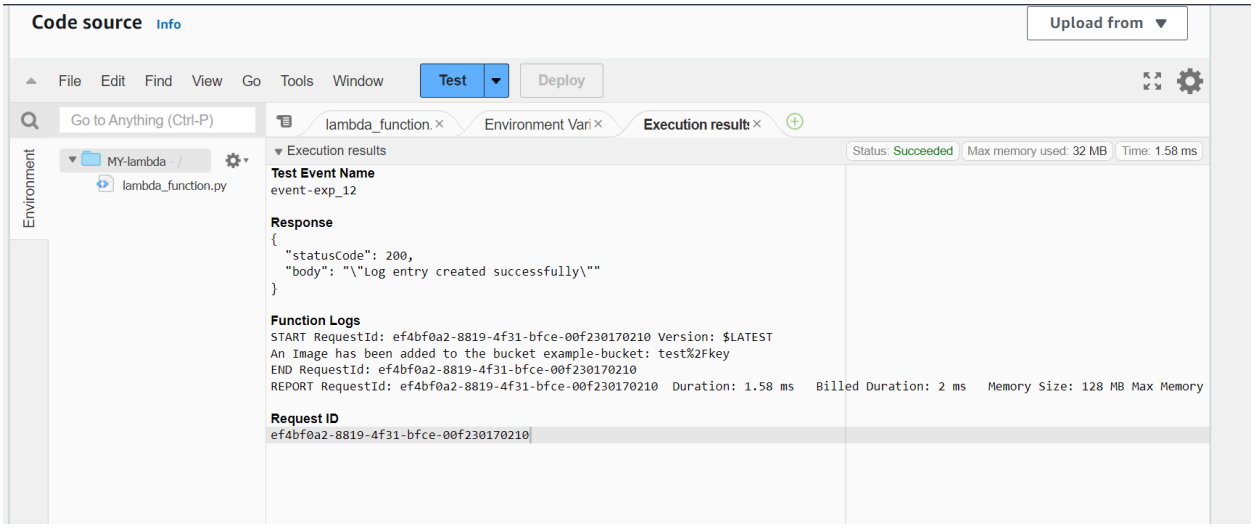


Step 9: Now upload any image to the bucket.

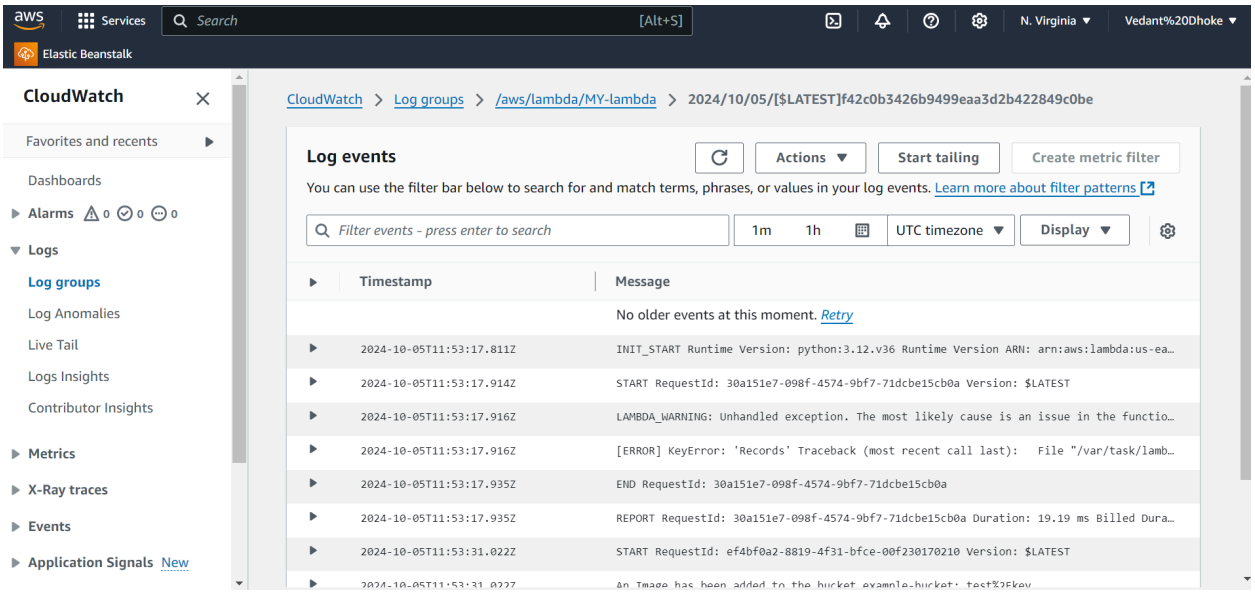


Step 10: Now to click on test in lambda to check whether it is giving log when image is added to S3.





Step 11: Now Lets see the log on Cloud watch.To see it go to monitor section and then click on view cloudwatch logs.



Conclusion: In this experiment, we successfully created an AWS Lambda function that logs a message when an image is uploaded to an S3 bucket.The function was successfully triggered by S3 object uploads, validating the functionality of Lambda's event-driven architecture. This experiment demonstrated how Lambda can efficiently respond to S3 events and how to troubleshoot common issues with event structure.

Errors faced during the experiment :

It is important to configure S3 triggers properly, after changing the code its important to save the code then deploy it and then click on the test to see successful output otherwise we might see an error after clicking on the test.