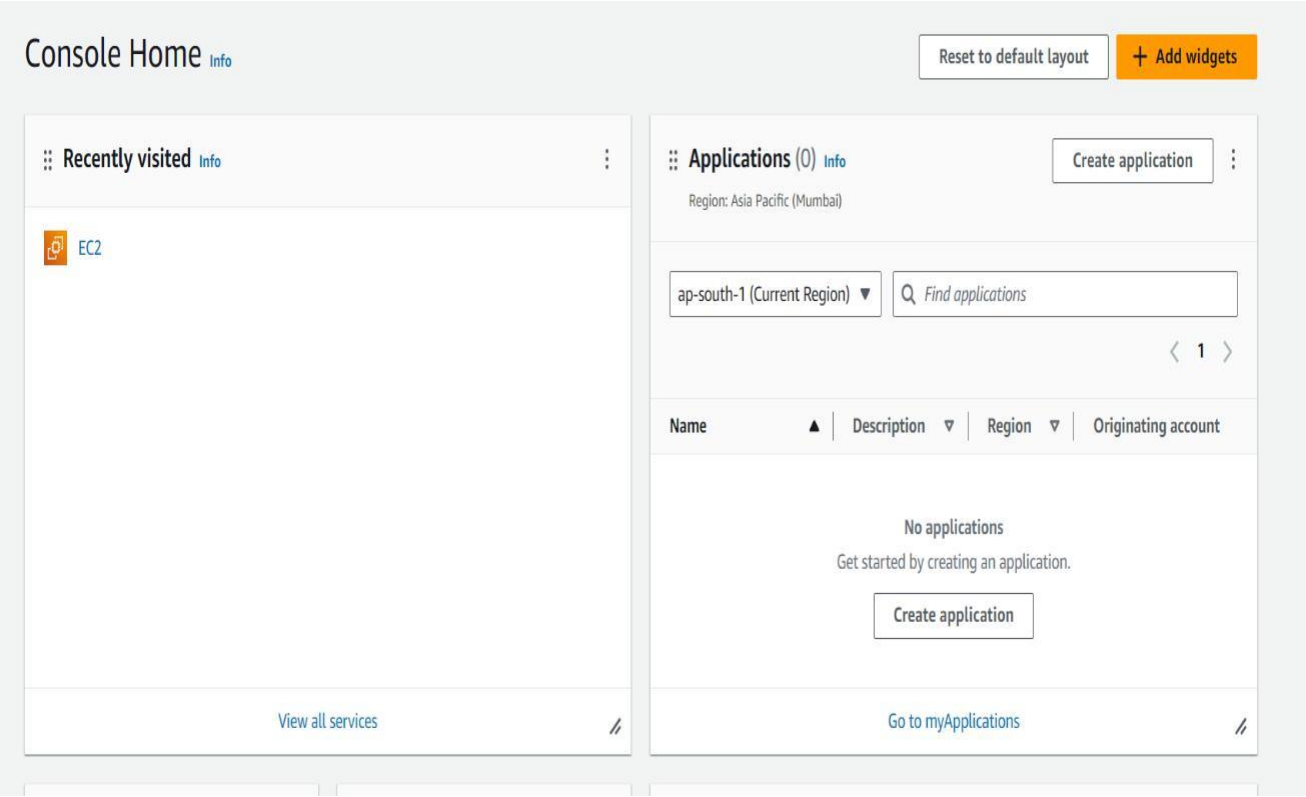
Name : Himesh Pathai
Div : D10A
Roll No. : 35

EXPERIMENT NO. 1

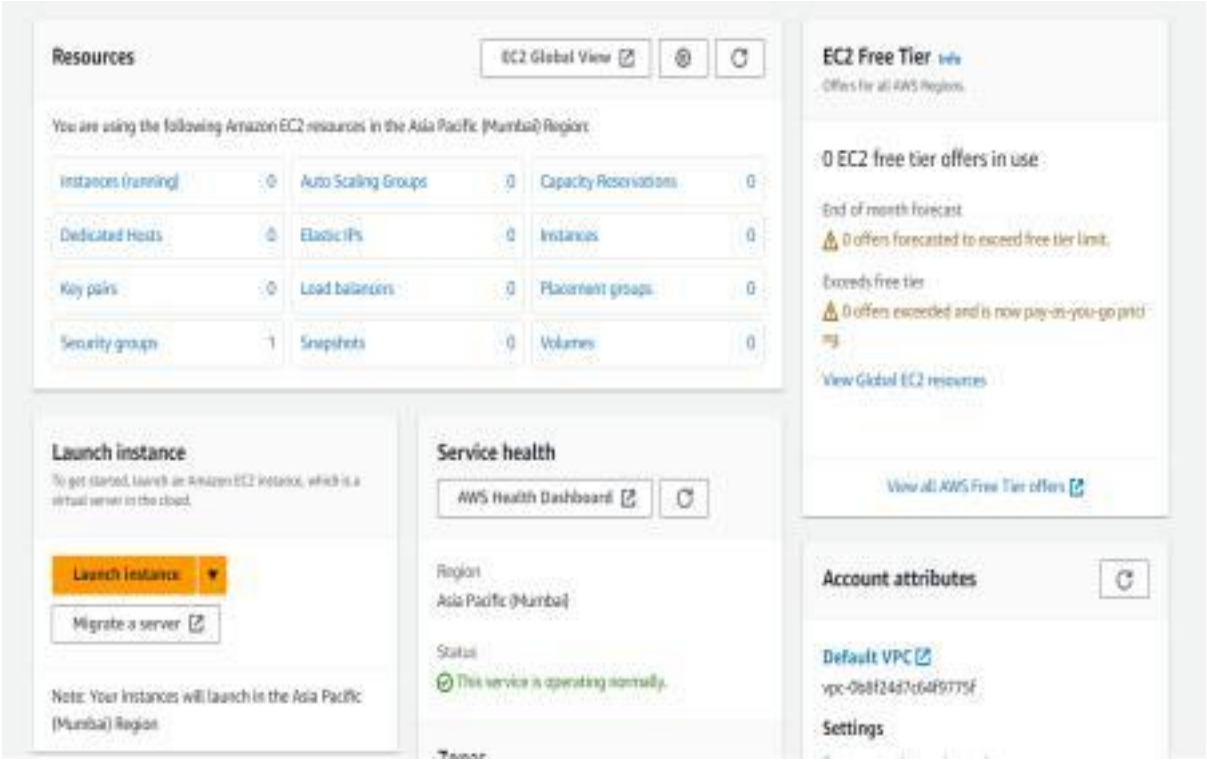
Aim : To understand the benefits of Cloud Infrastructure and Setup AWS Cloud9 IDE, Launch AWS Cloud9 IDE and Perform Collaboration Demonstration.

EC2 Instance Creation and static site hosting

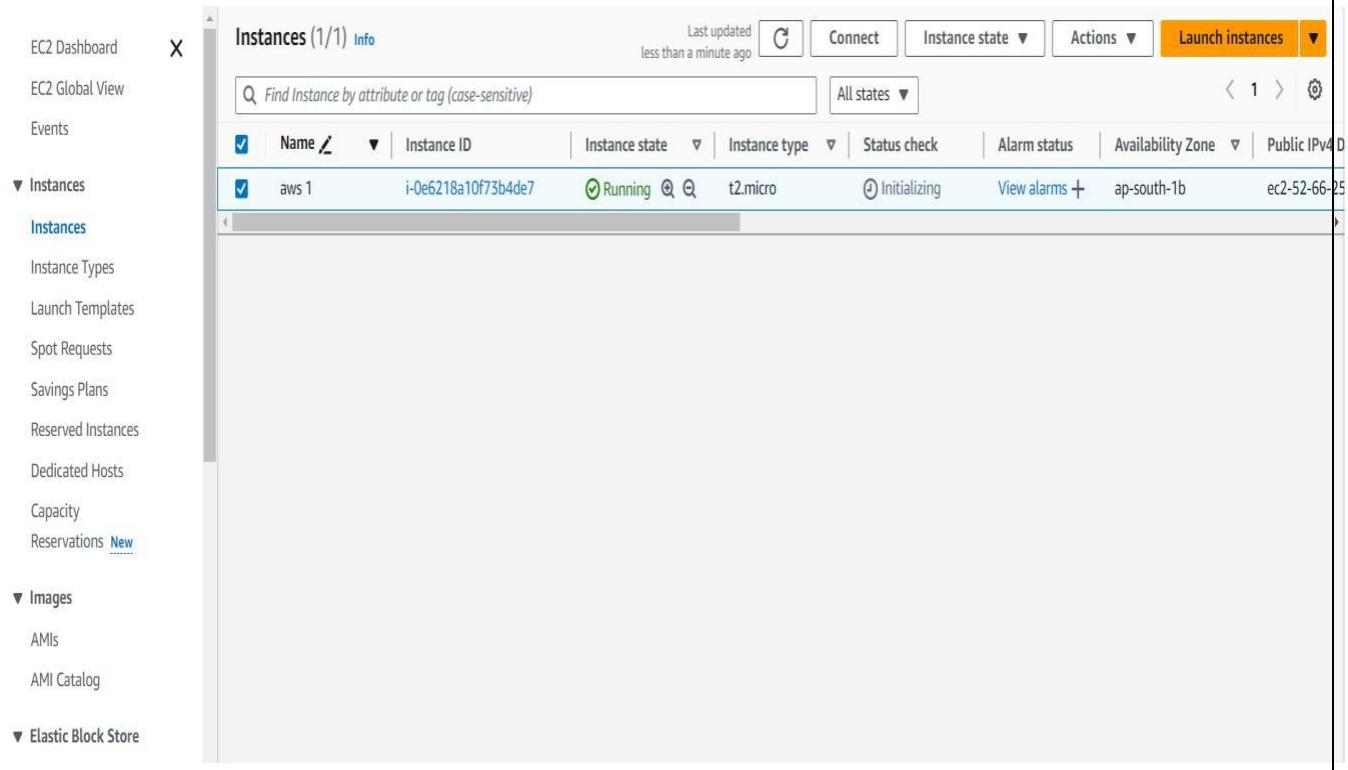
1. Login to your AWS account



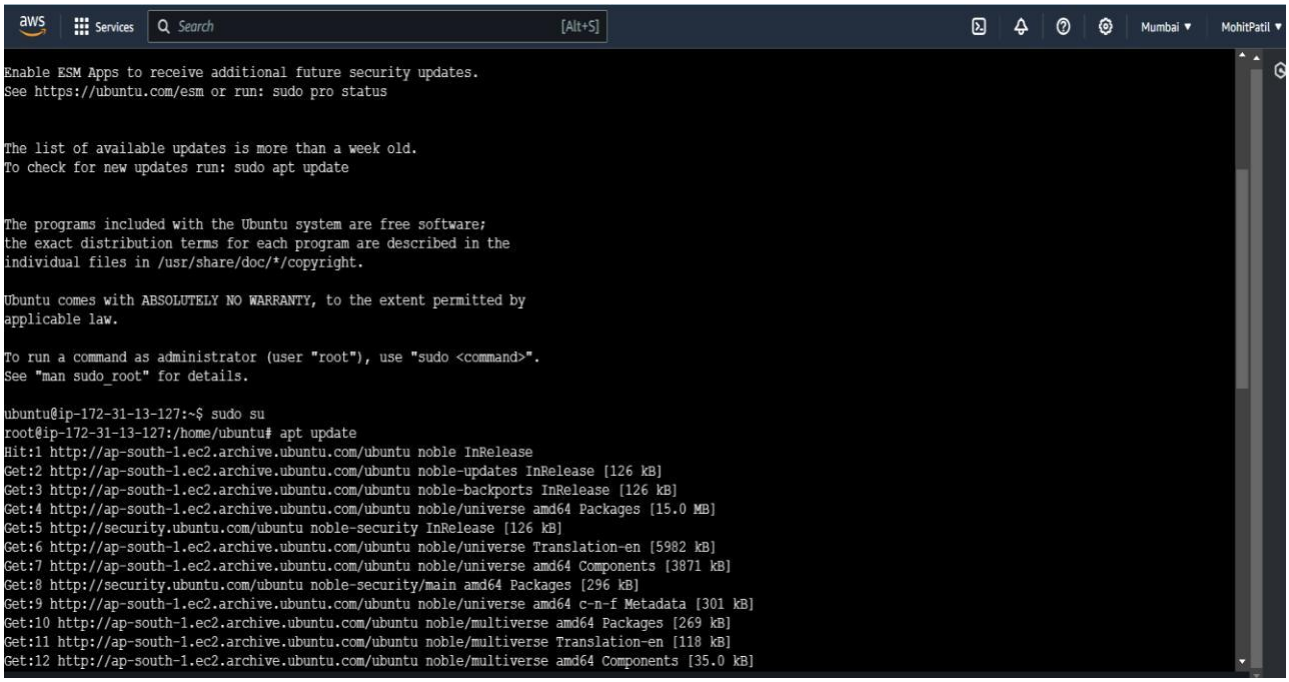
2. Click on EC2 and then create an instance by clicking on instances



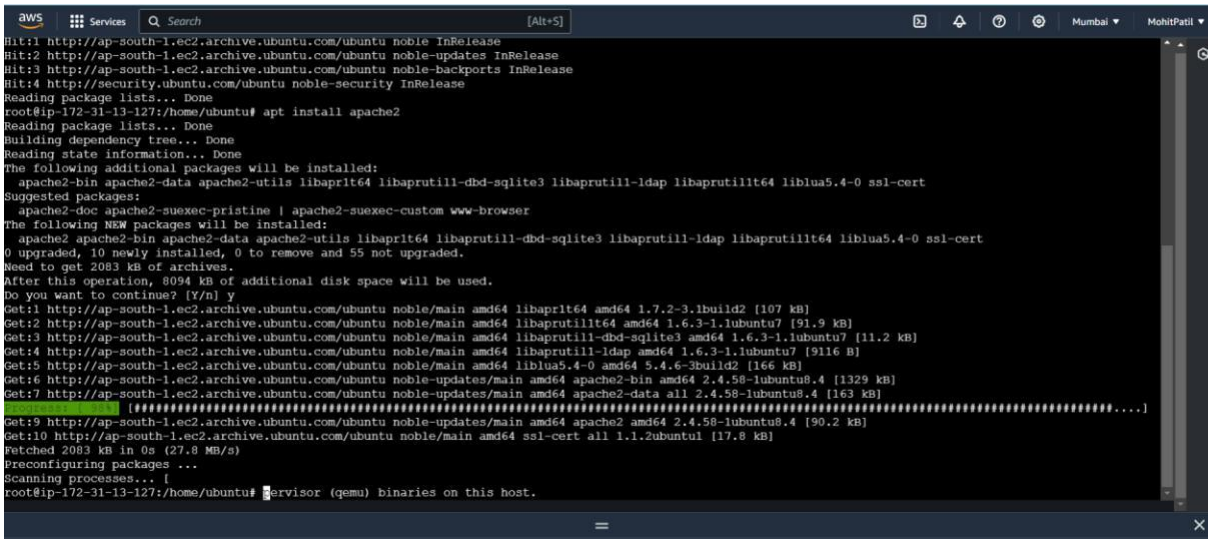
3. After an instance is created wait for it to come to Running state



4. After doing that you will see this UI

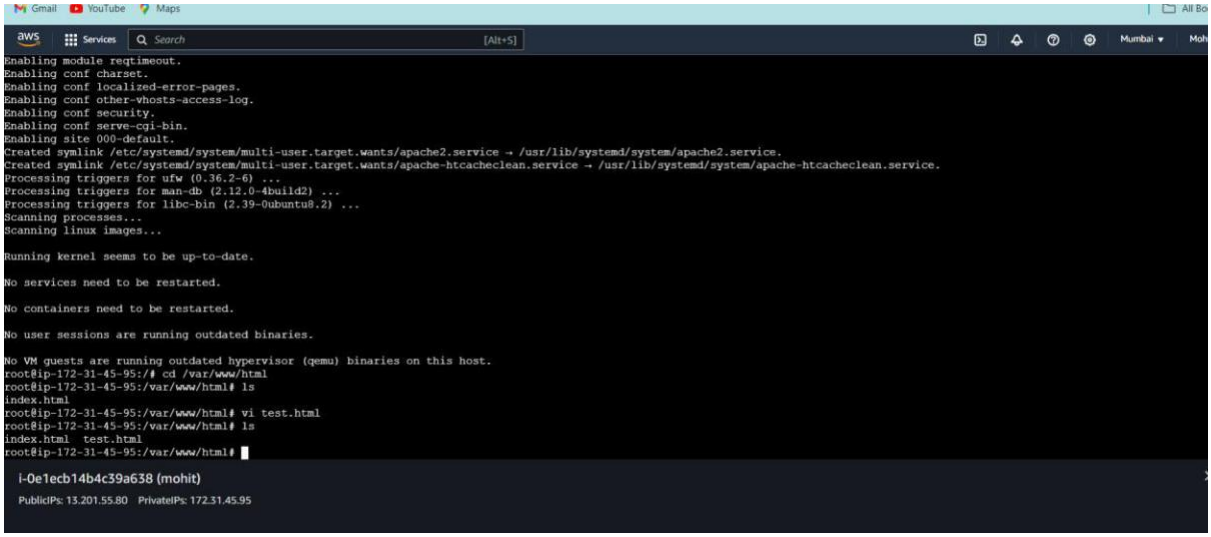


5. Follow these steps and then run these commands

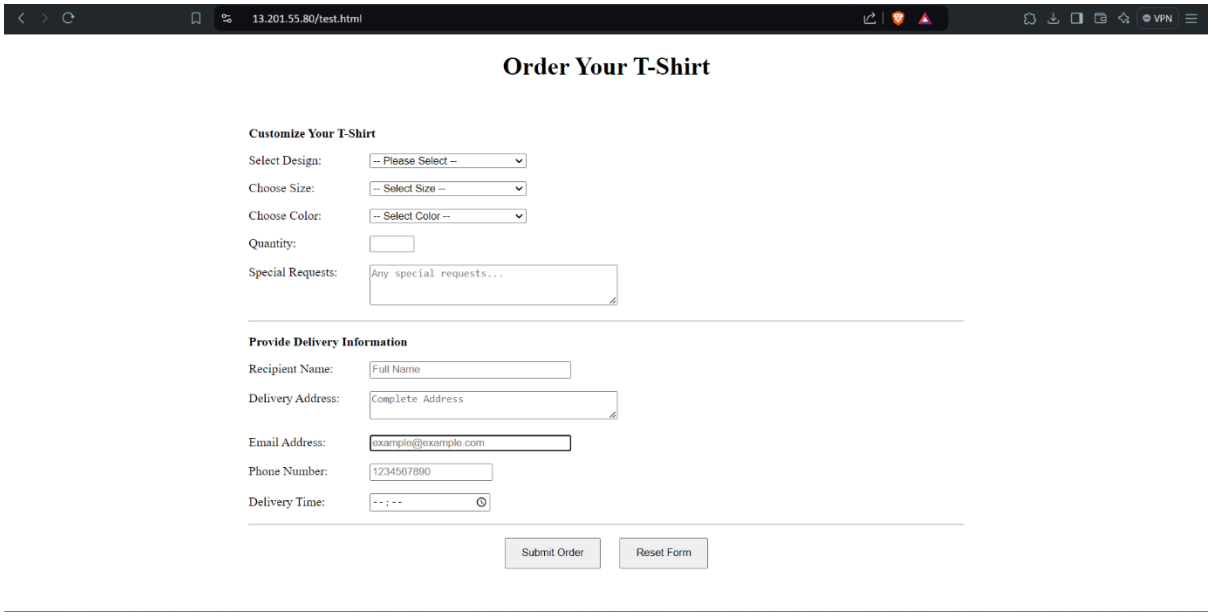


6. After that the ip-address which was given while running the instance, copy that and paste that on chrome, make sure that it is http and not https

7. Create a file using vi command and save it using : wq

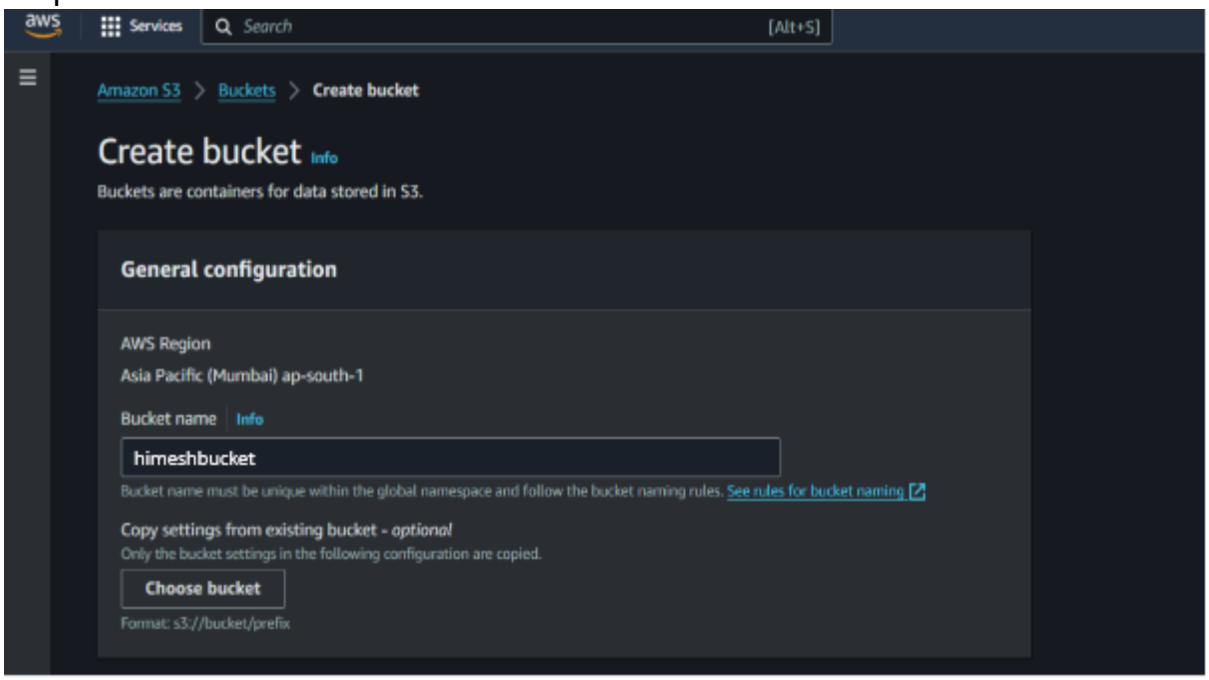


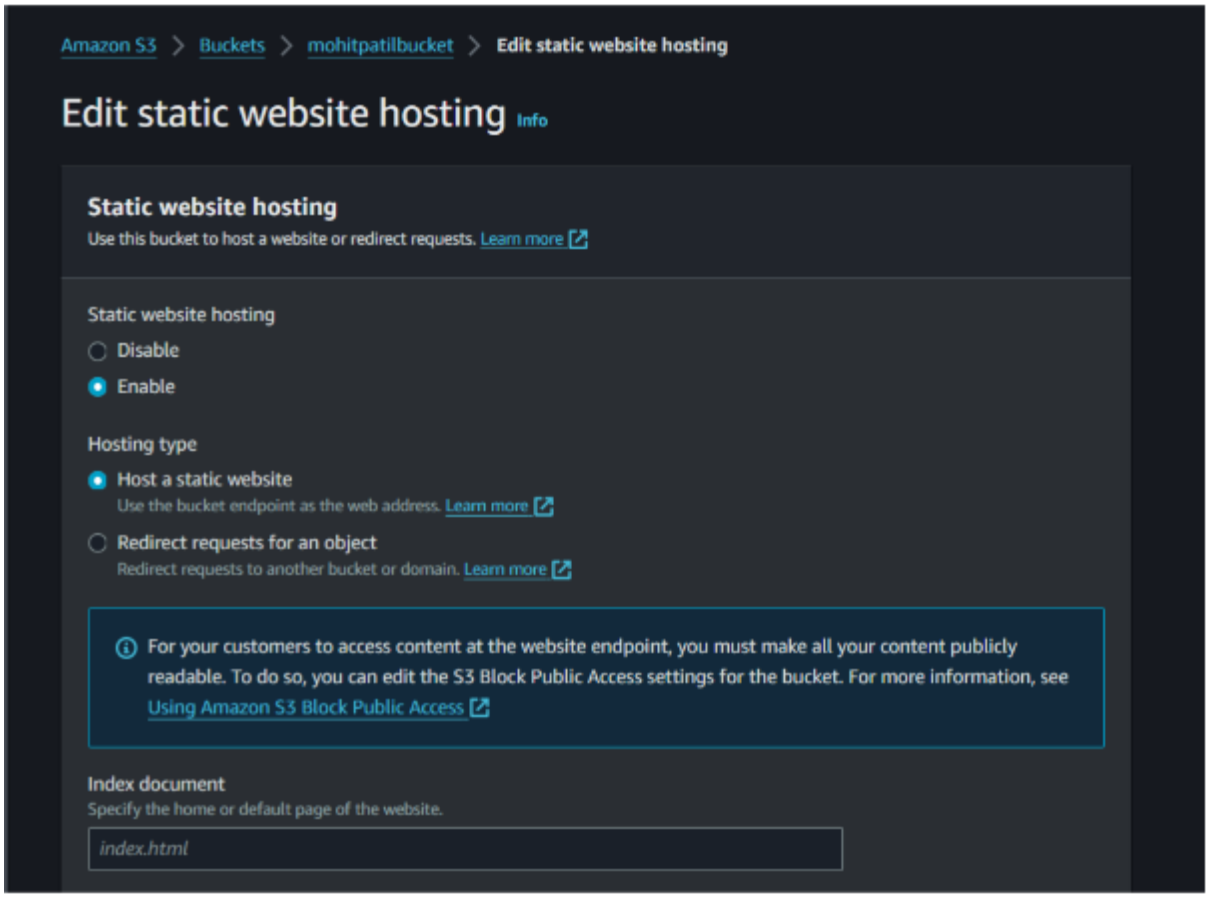
8. After saving that file go that page where ubuntu is listed and then on the link add “/your_file_name.html” and then whatever you saved on that file will be displayed



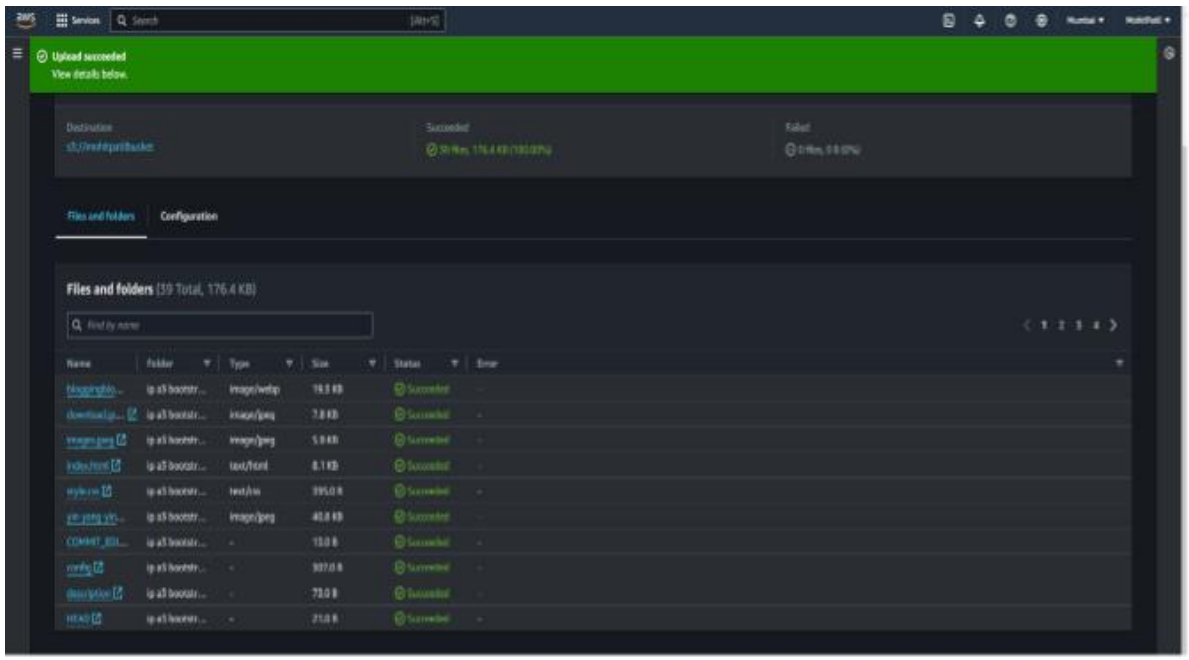
Static Hosting using S3 bucket

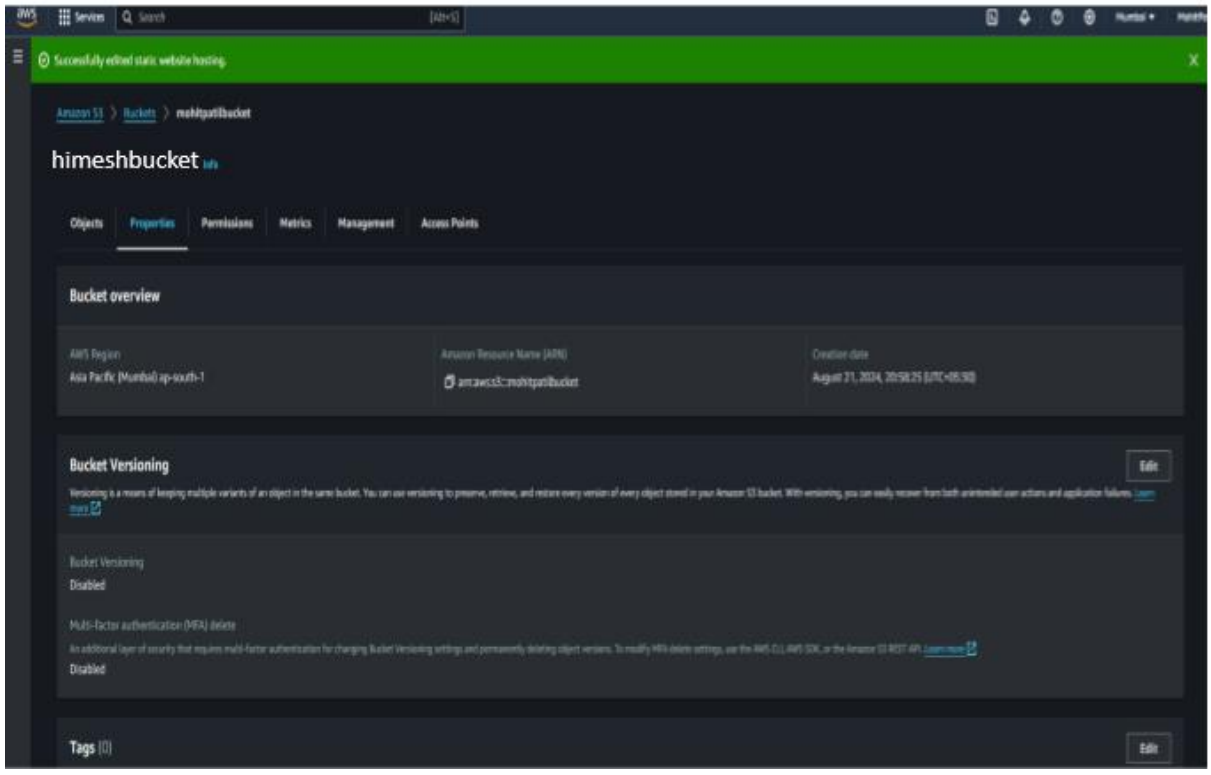
Step1: Create bucket

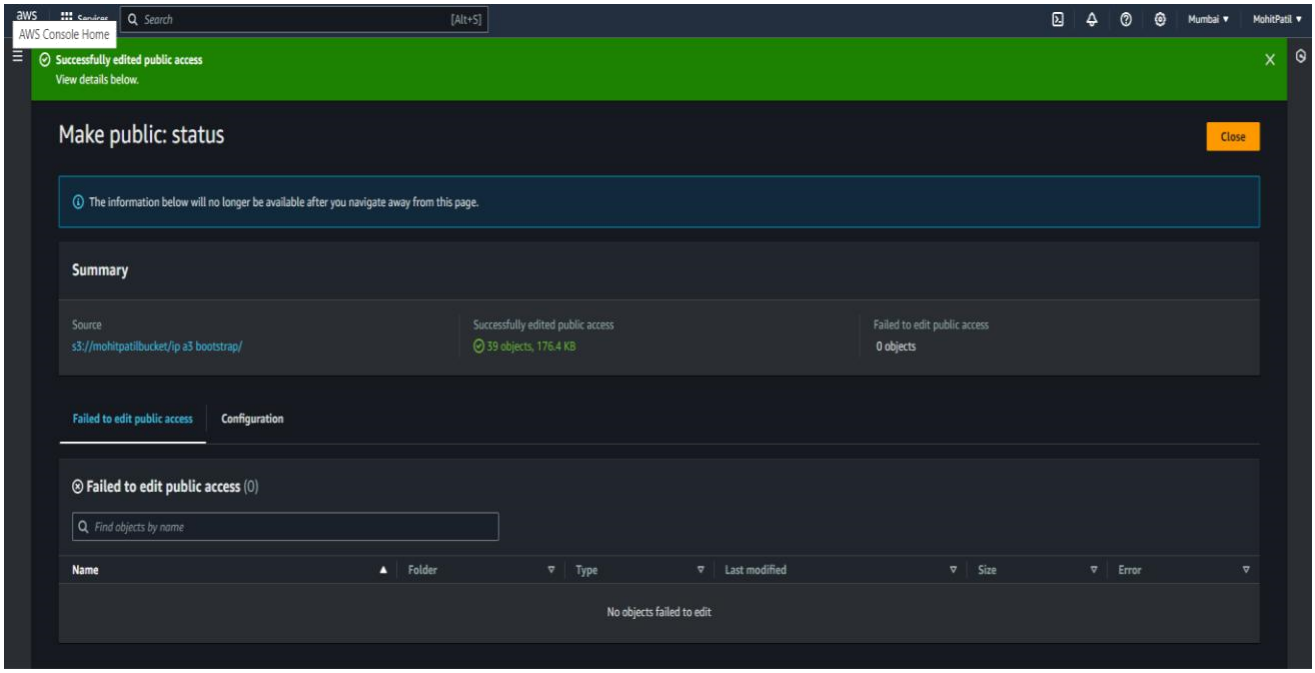




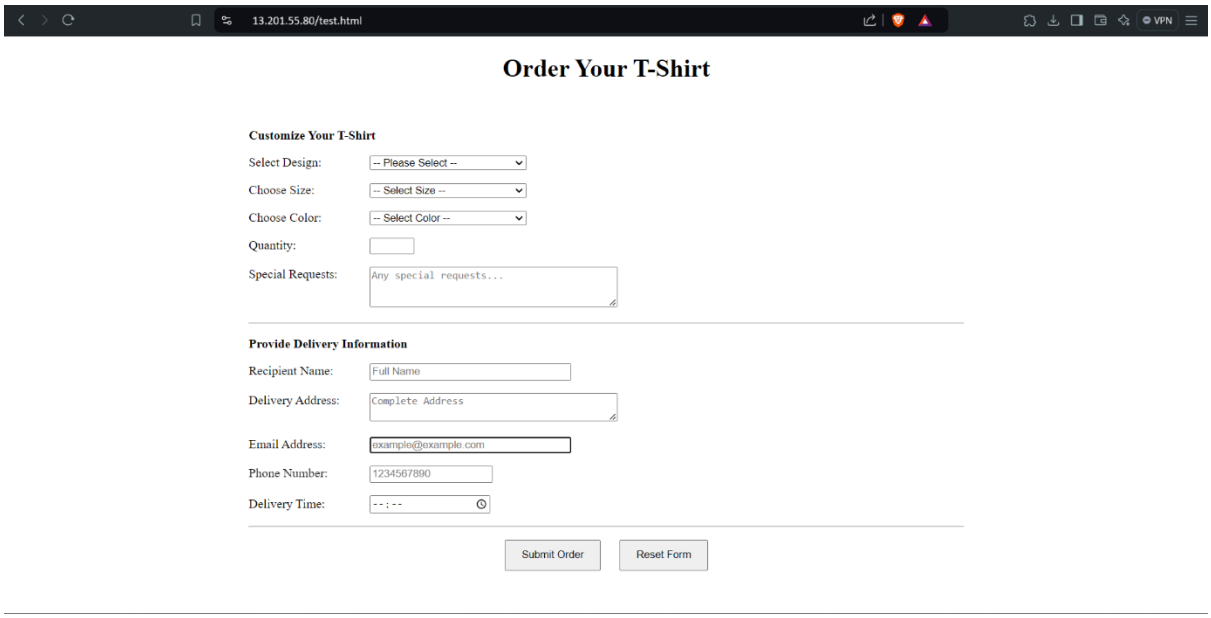
Step 2: Add resources





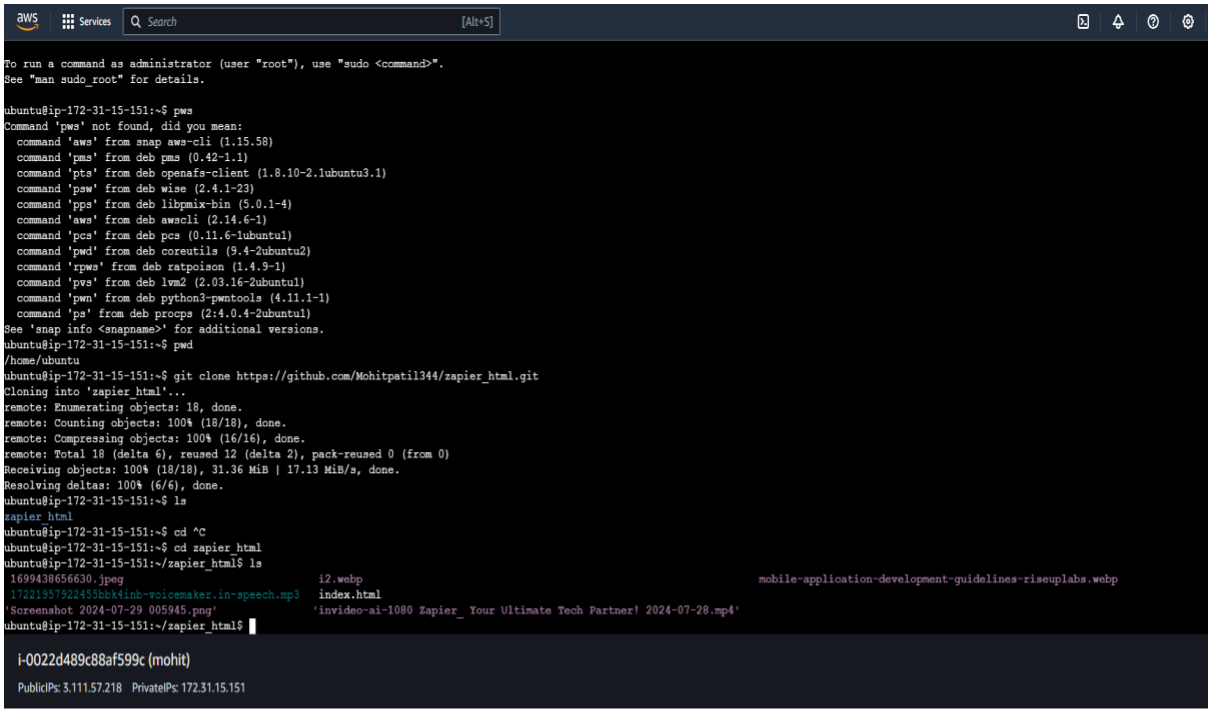


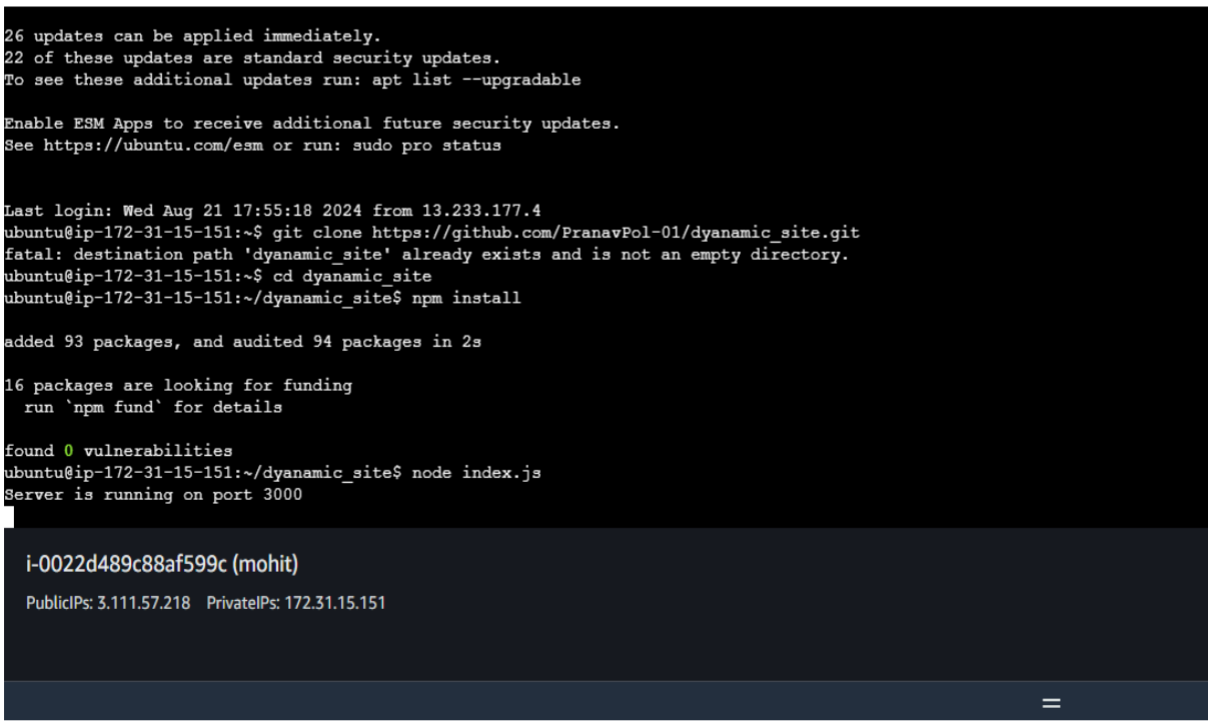
Step 3 : visit hosted website



EC2 Dynamic Site Hosting

Step 1 : Open Console and clone the github repository



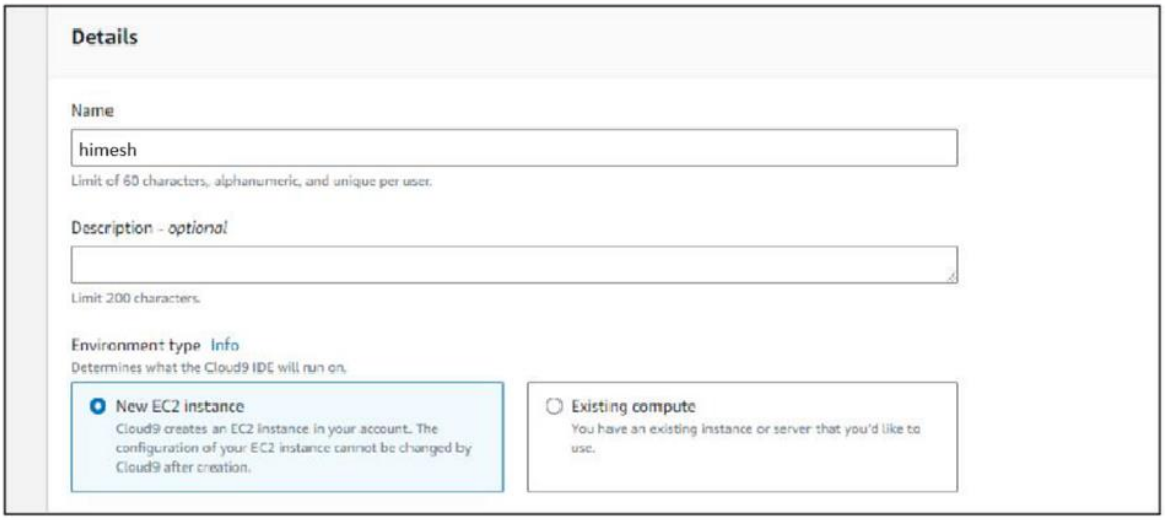


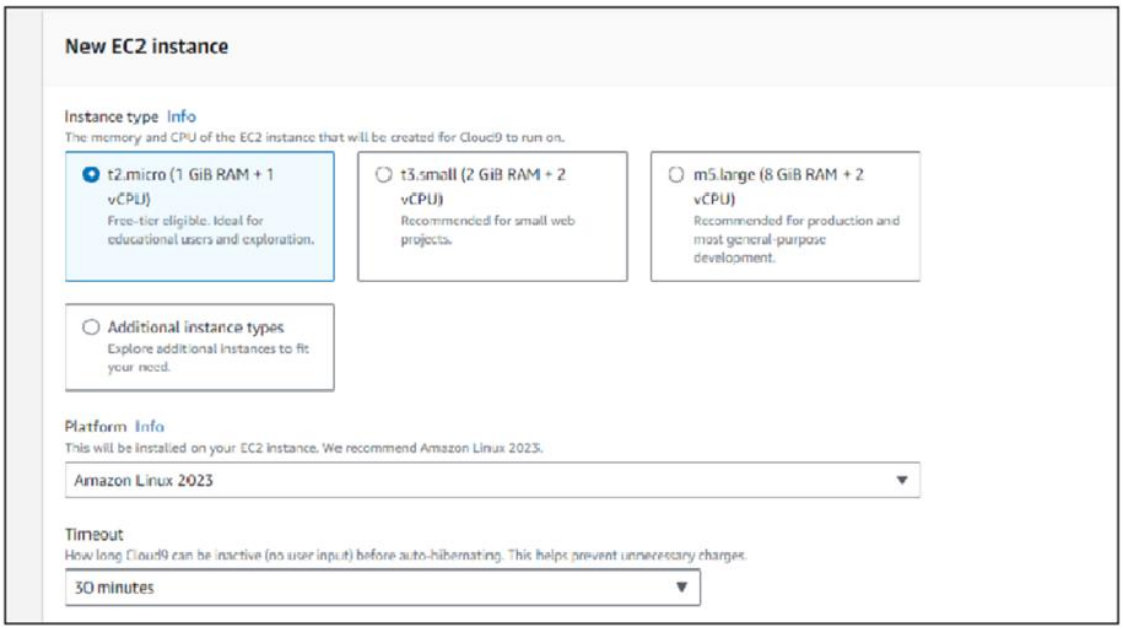
Step 2 : Install necessary Packages and run website on port 3000

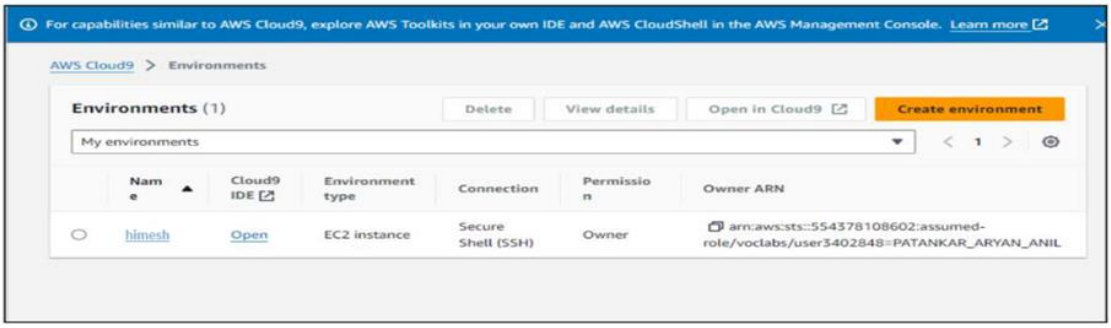
Hey this is Dynamic Website.

Hey this is about page.

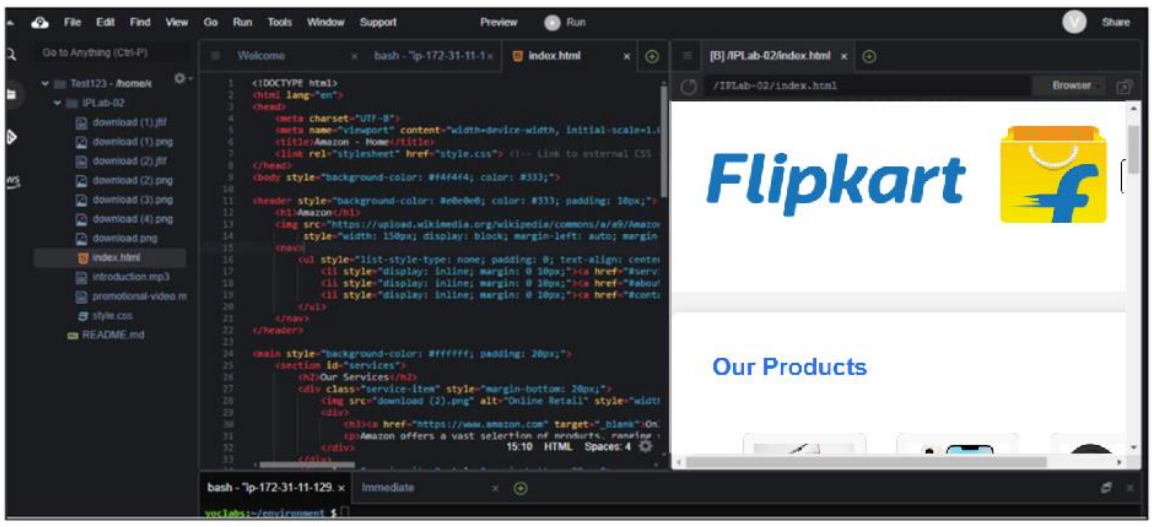
Cloud 9 IDE Site Hosting






For capabilities similar to AWS Cloud9, explore AWS Toolkits in your own IDE and AWS CloudShell in the AWS Management Console. Learn more
AWS Cloud9 > Environments
Environments (1)
Delete
View details
Open in Cloud9
Create environment
My environments
Name
Cloud9 IDE
Environment type
Connection
Permission
Owner ARN
himesh
Open
EC2 instance
Secure Shell (SSH)
Owner
arn:aws:sts::554378108602:assumed-role/voclabs/user3402848=PATANKAR_ARYAN_ANIL


Welcome
Developer Tools
AWS Cloud9
Welcome to your development environment
AWS Cloud9 allows you to write, run, and debug your code with just a browser. You can , write code for , with others in real time, and much more.
Getting started
Toolkit for AWS Cloud9
The AWS Toolkit for Cloud9 is an IDE extension that simplifies accessing and interacting with resources from services such as AWS Lambda, AWS CloudFormation, and AWS API Gateway. With the toolkit, developers can also develop, debug, and deploy applications


Flipkart
Our Products

Name : Himesh Pathai

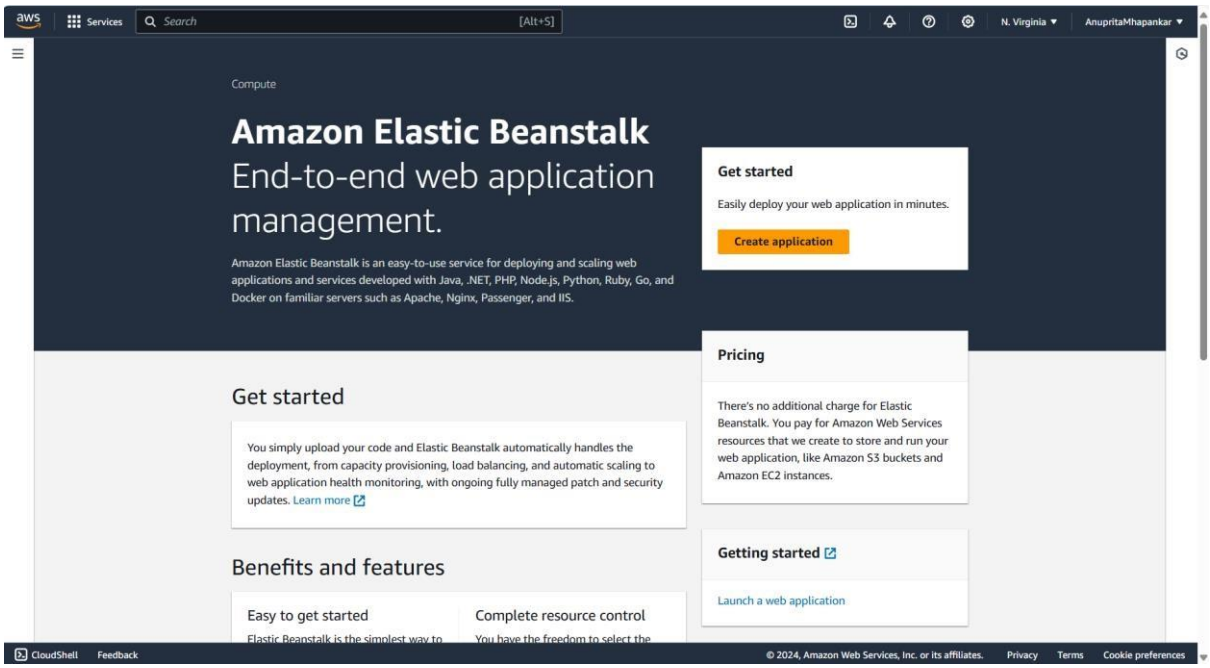
Div : D10A

Roll No. : 35

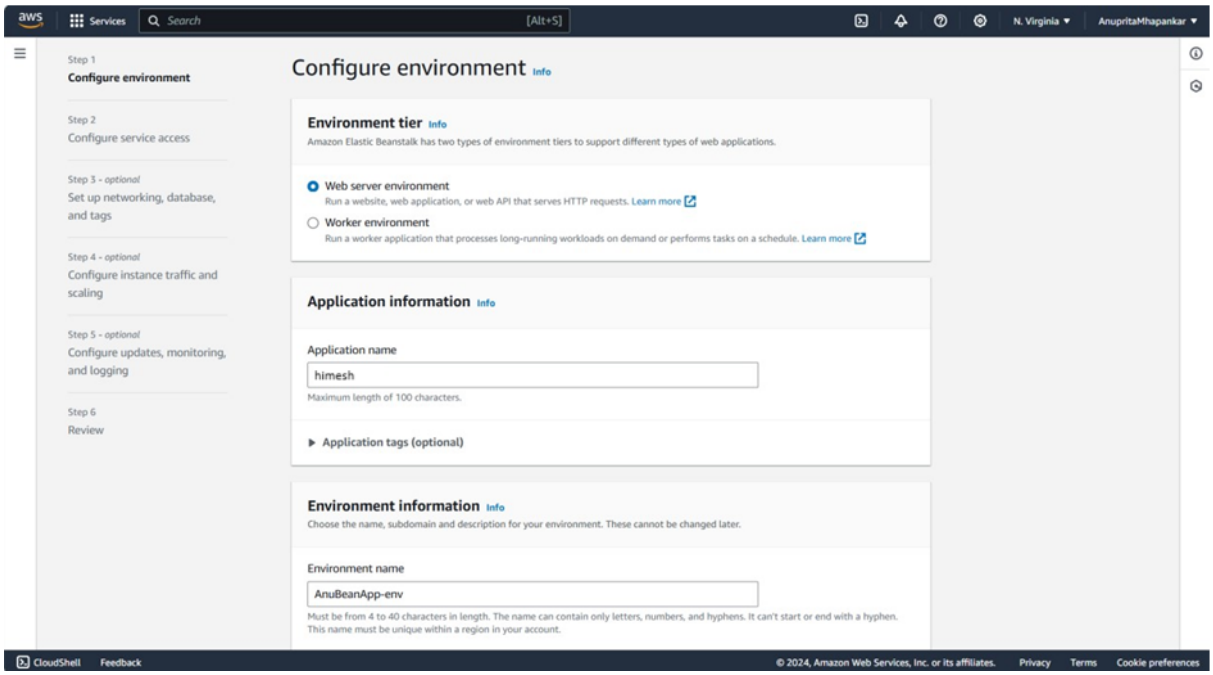
Experiment 2

Using Beanstalk

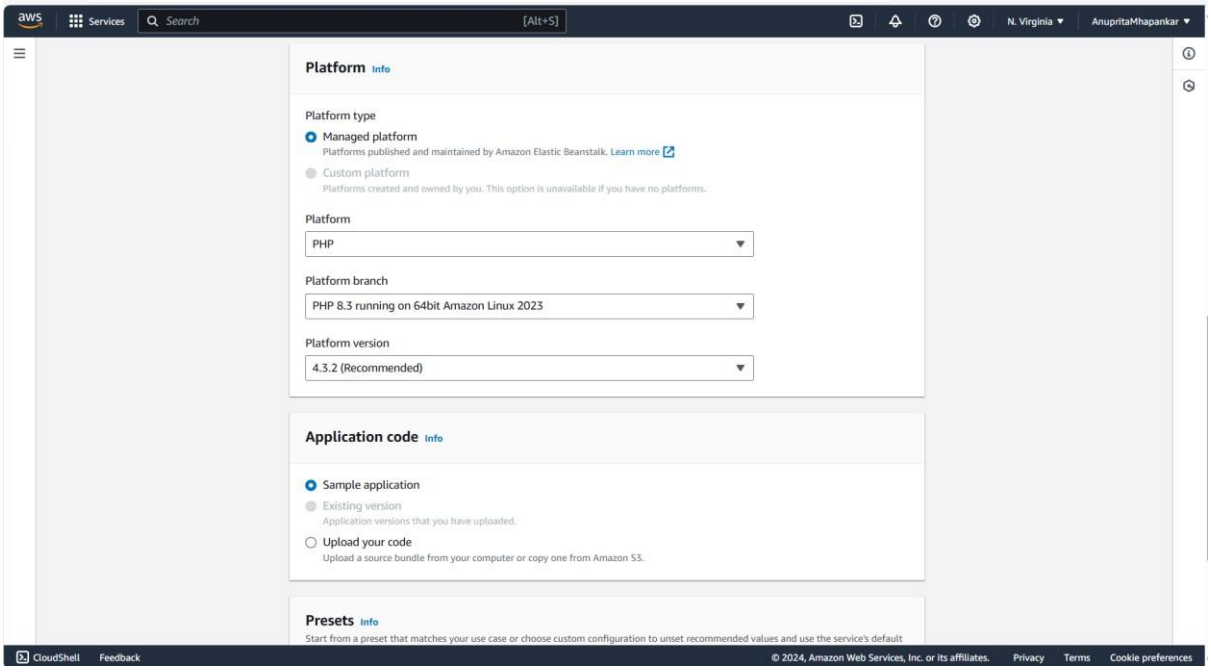
1. Search Elastic Beanstalk from Developer Tools



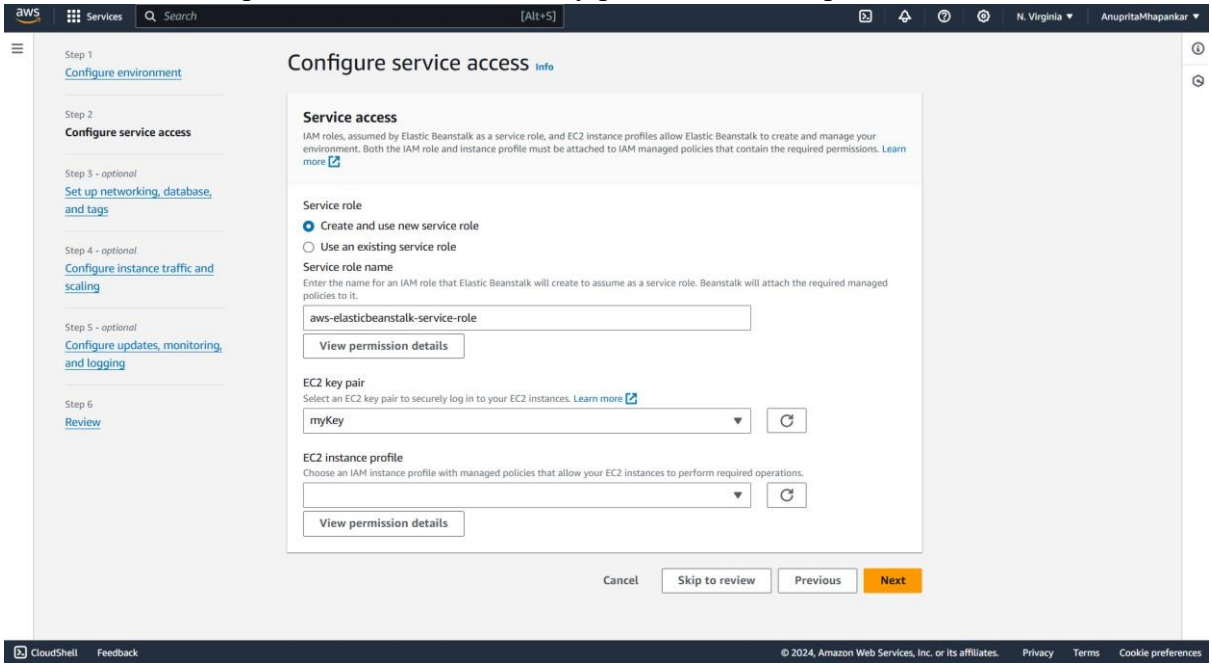
2. Click on create application and configure the environment



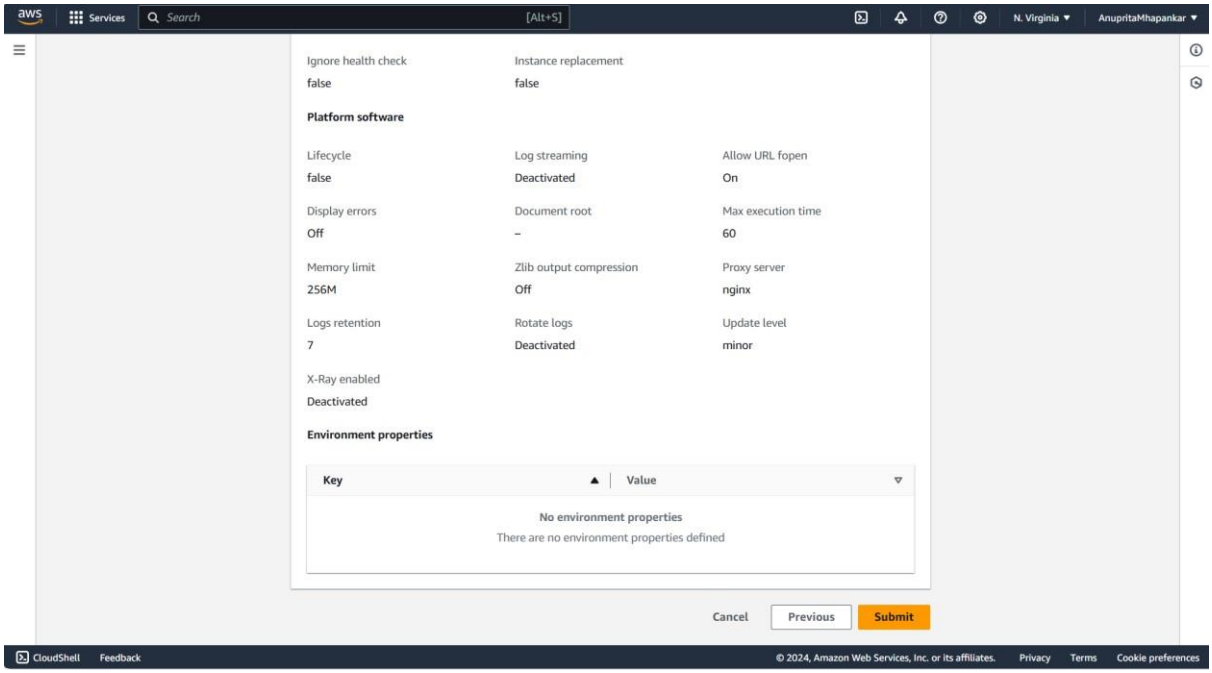
3. Choose PHP from the dropdown menu and click next



4. From the dropdown menu select the key pair and instance profile

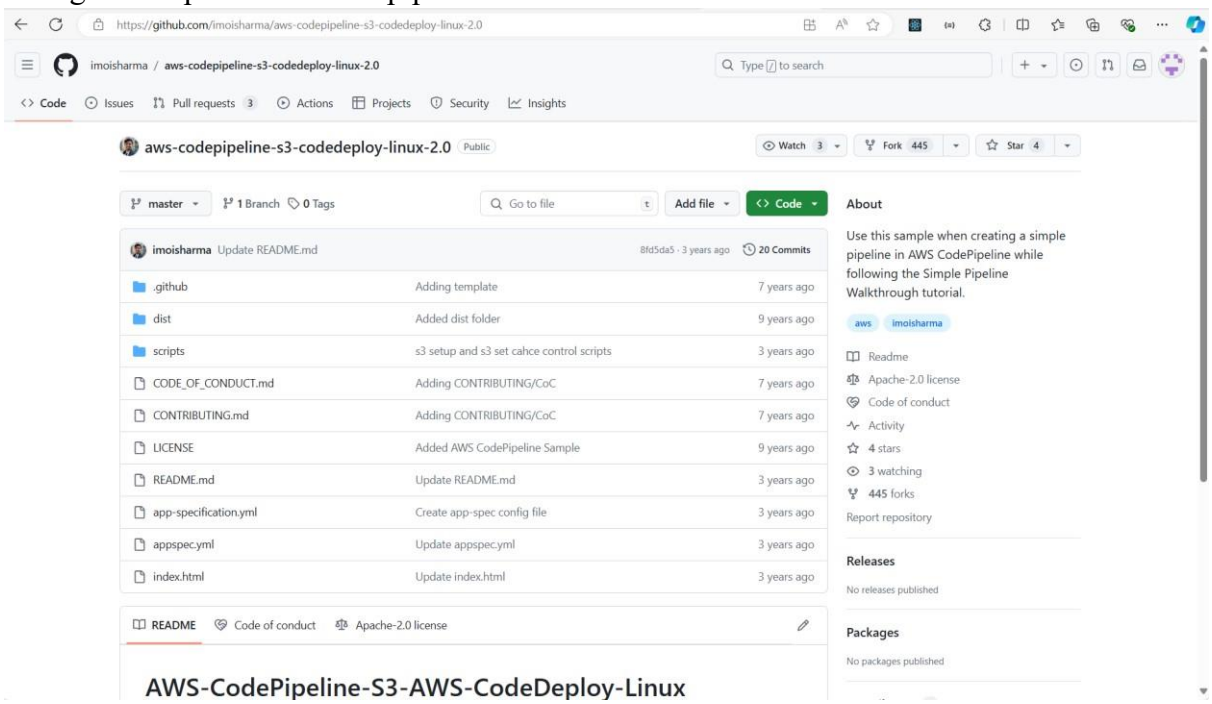


5. Review the changes made and click on Submit

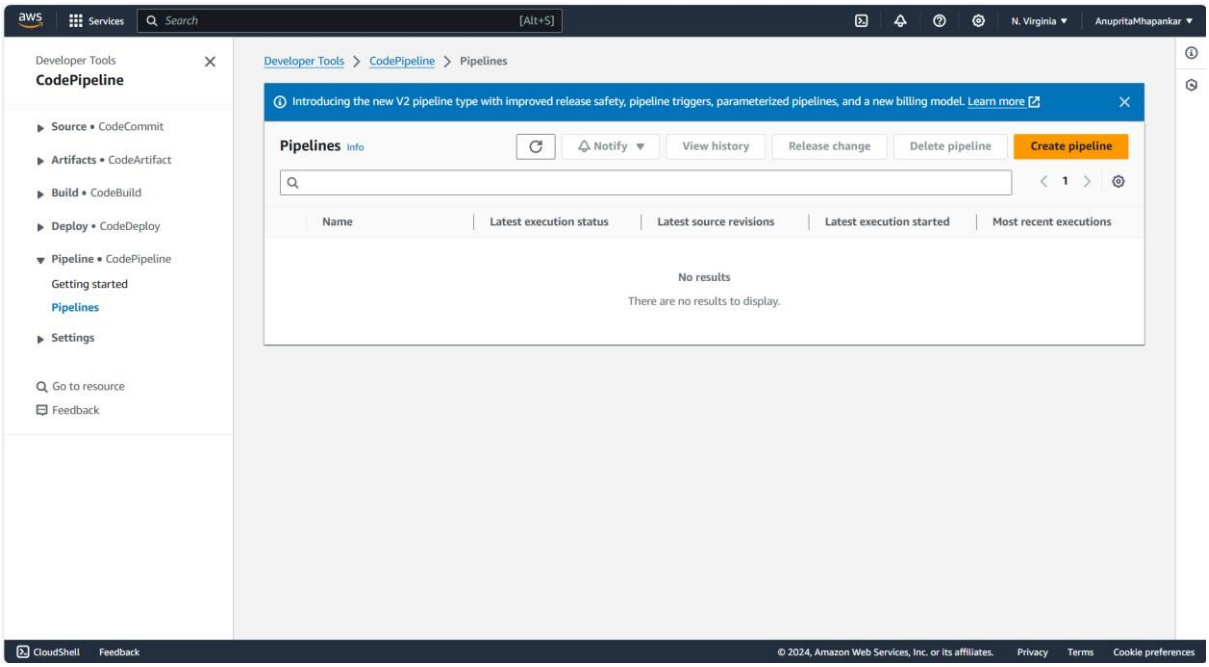


Pipeline Creation :

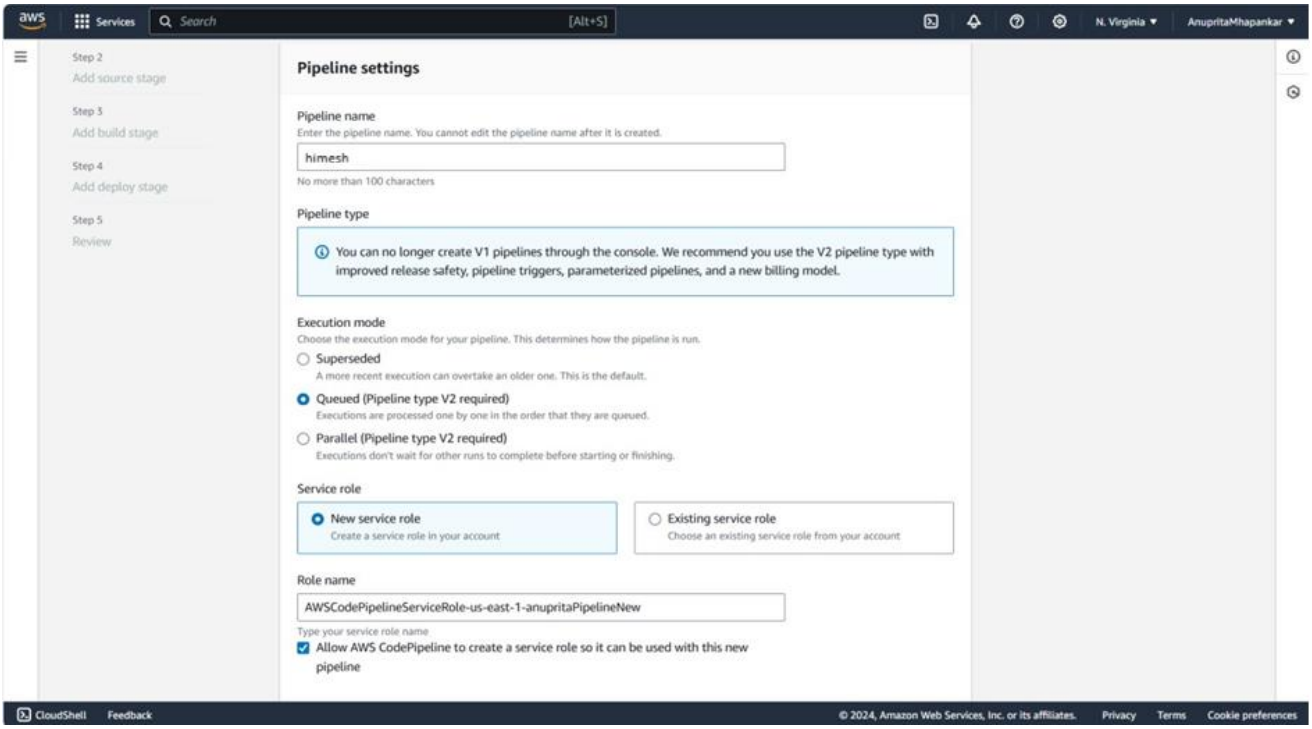
1. Fork a github repo for aws codepipeline.

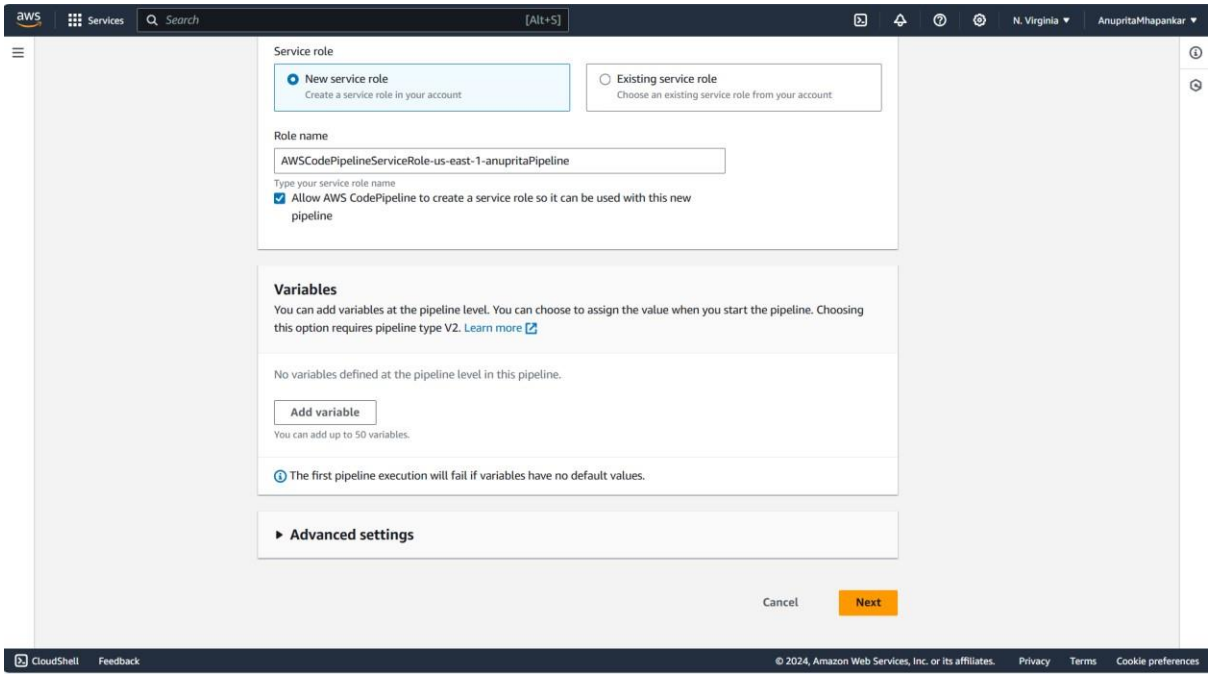


2. Go to developer tools and select CodePipeline and create a new pipeline

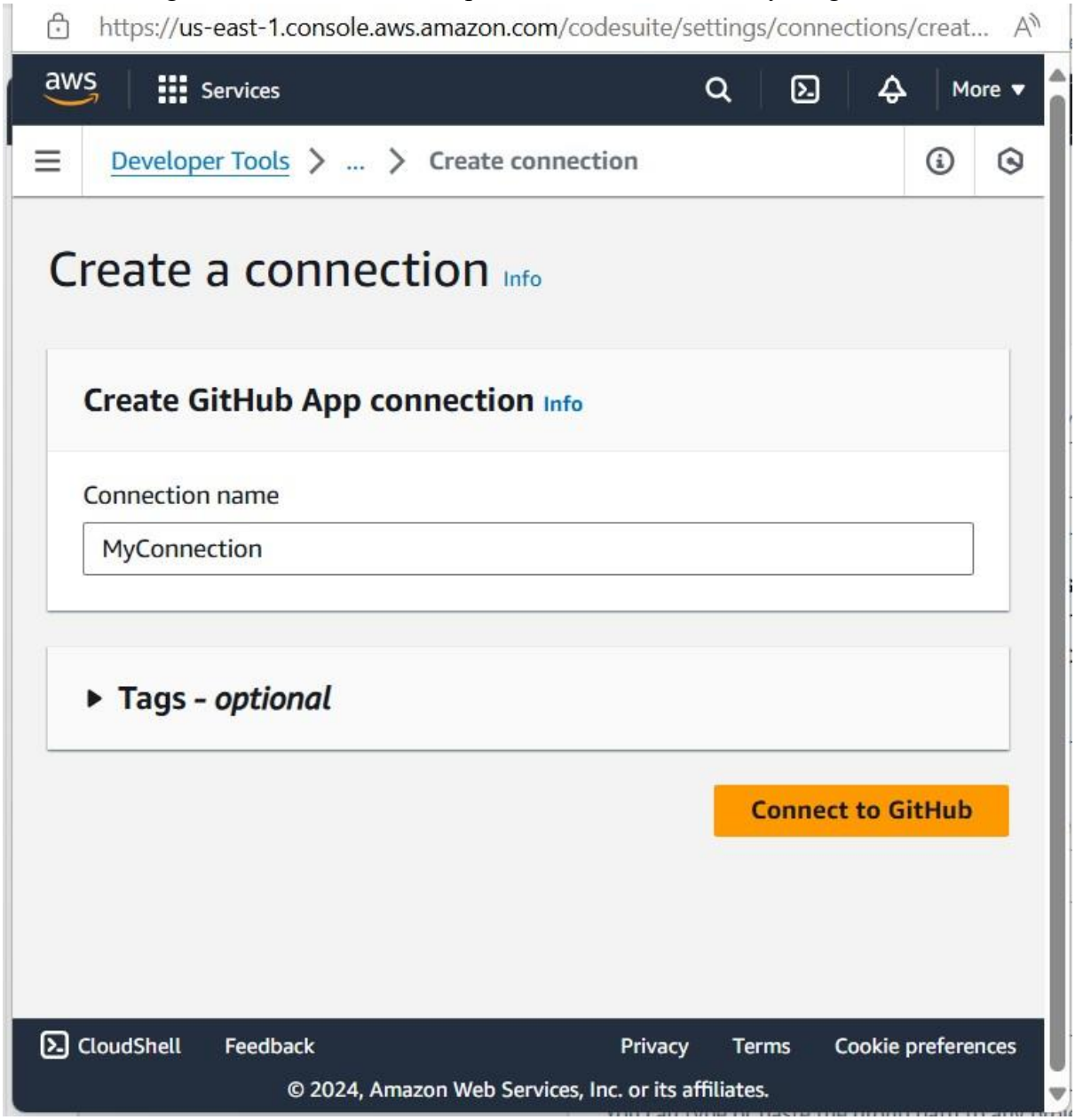


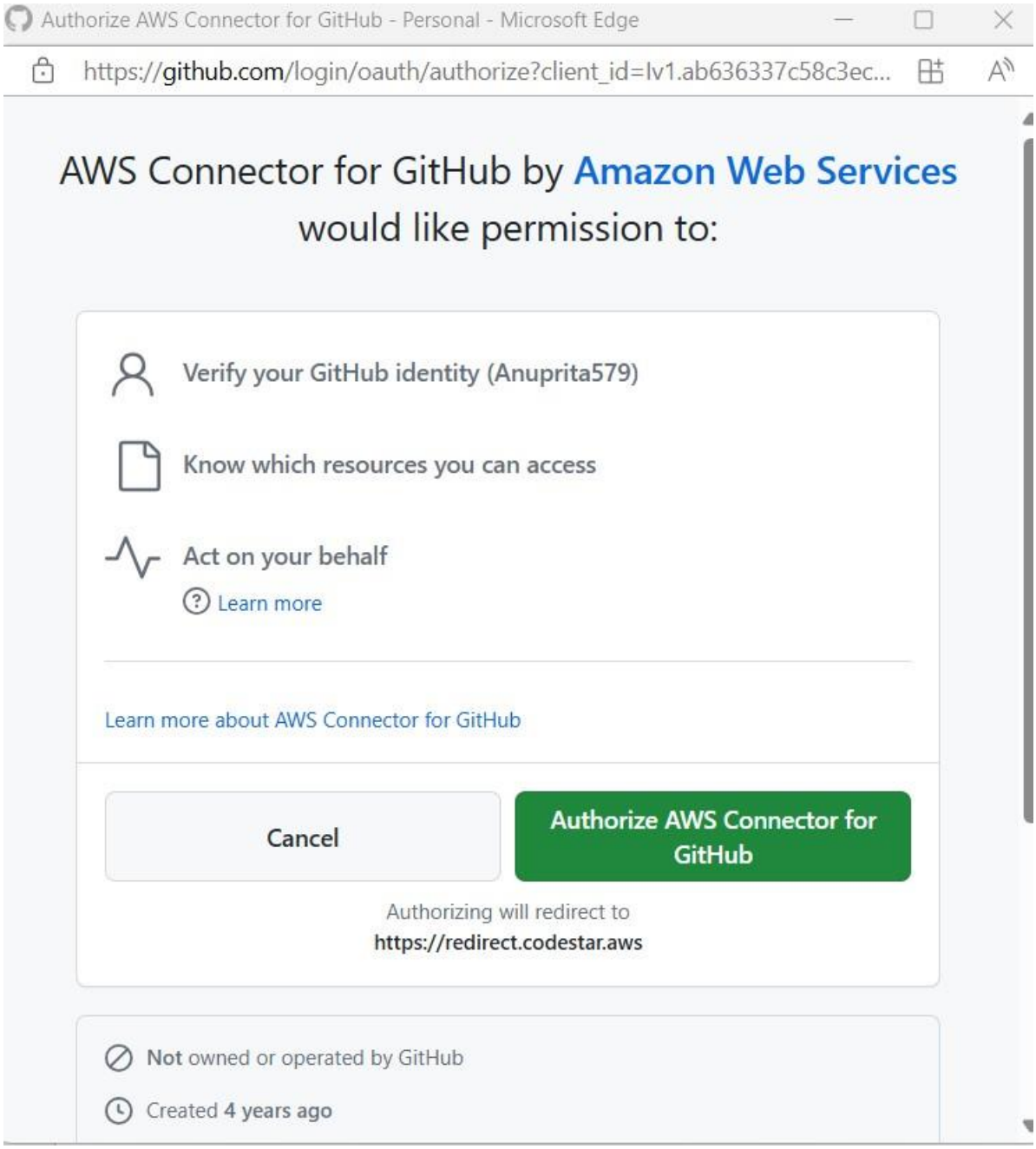
3. Name your pipeline and select the desired service role

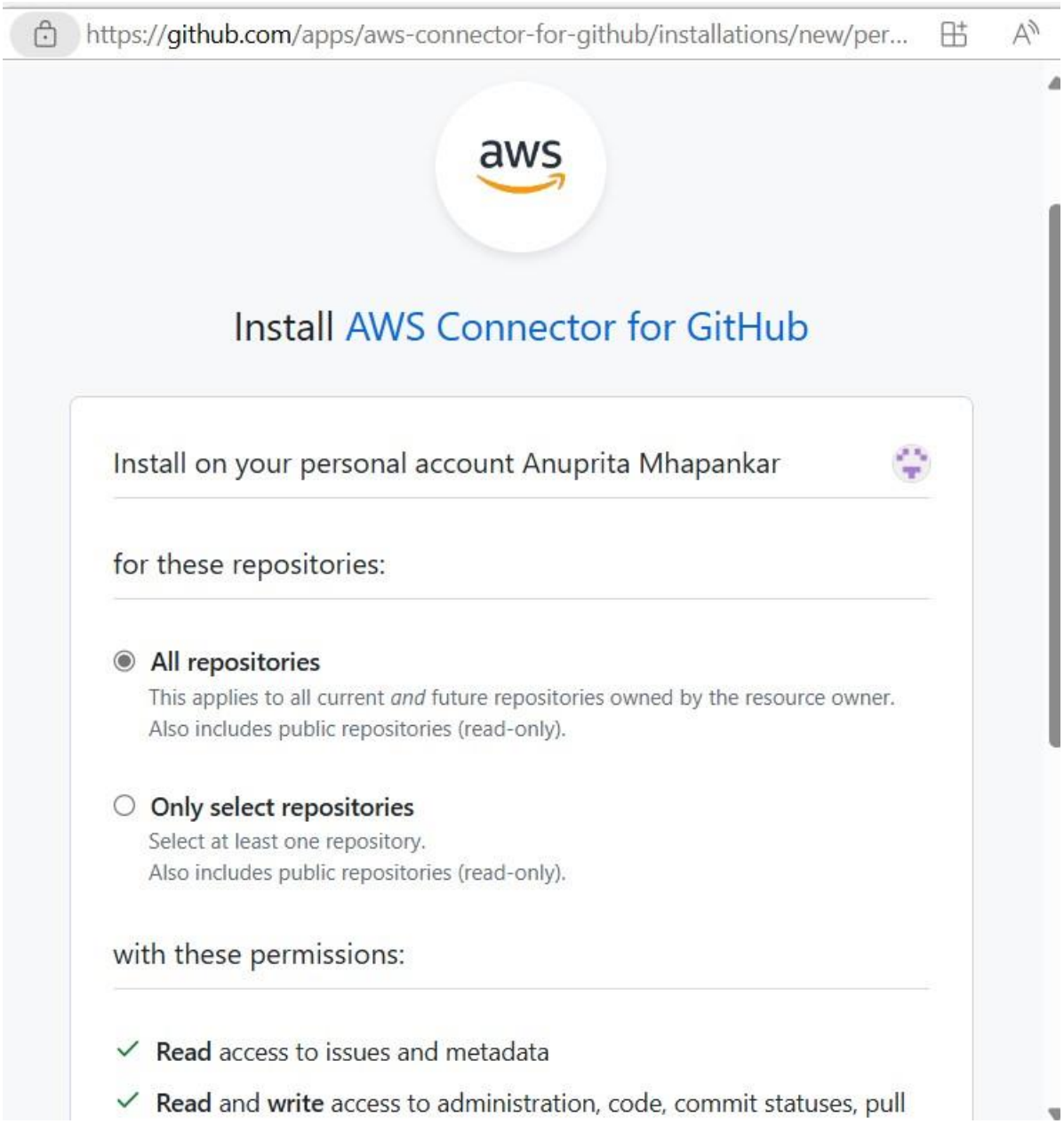



aws
Services
Search
[Alt+S]
N. Virginia
AnupritaMhapankar
Service role
New service role
Create a service role in your account
Existing service role
Choose an existing service role from your account
Role name
AWSCodePipelineServiceRole-us-east-1-anupritaPipeline
Type your service role name
Allow AWS CodePipeline to create a service role so it can be used with this new pipeline
Variables
You can add variables at the pipeline level. You can choose to assign the value when you start the pipeline. Choosing this option requires pipeline type V2. Learn more
No variables defined at the pipeline level in this pipeline.
Add variable
You can add up to 50 variables.
The first pipeline execution will fail if variables have no default values.
Advanced settings
Cancel
Next
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

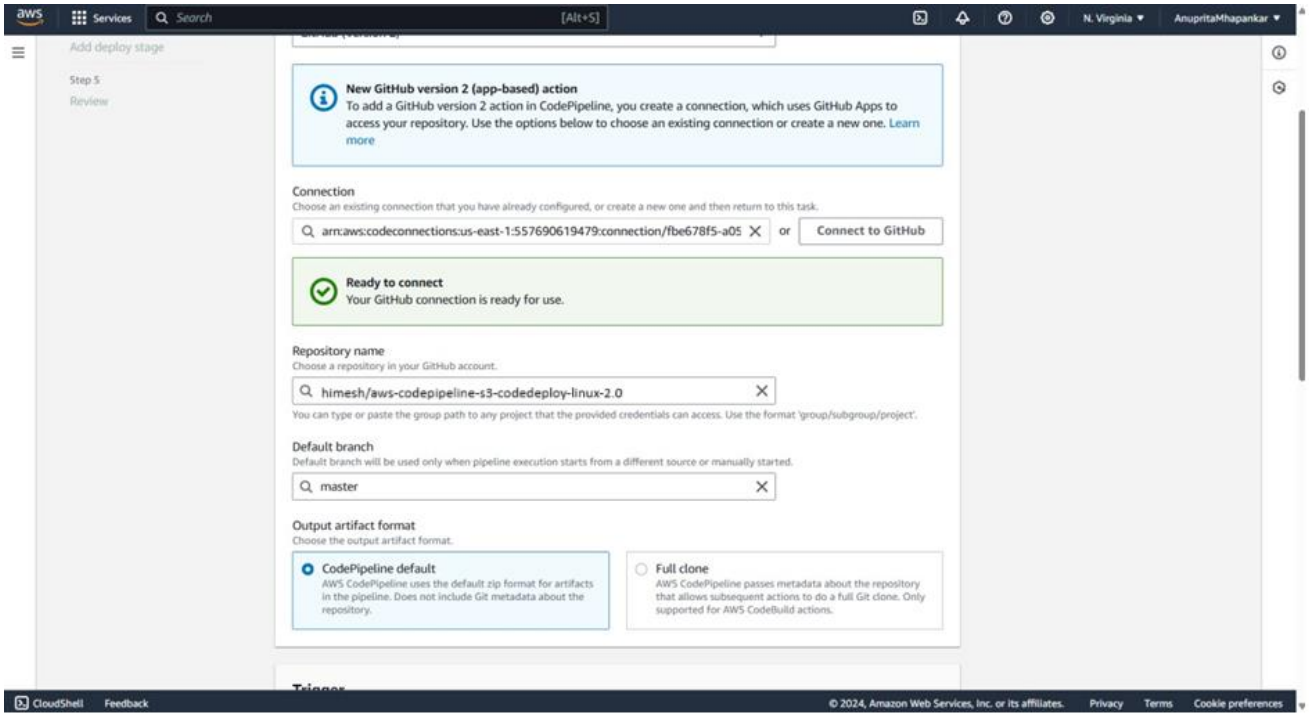
4. In the source stage select Github v2 as the provider and then connect your github



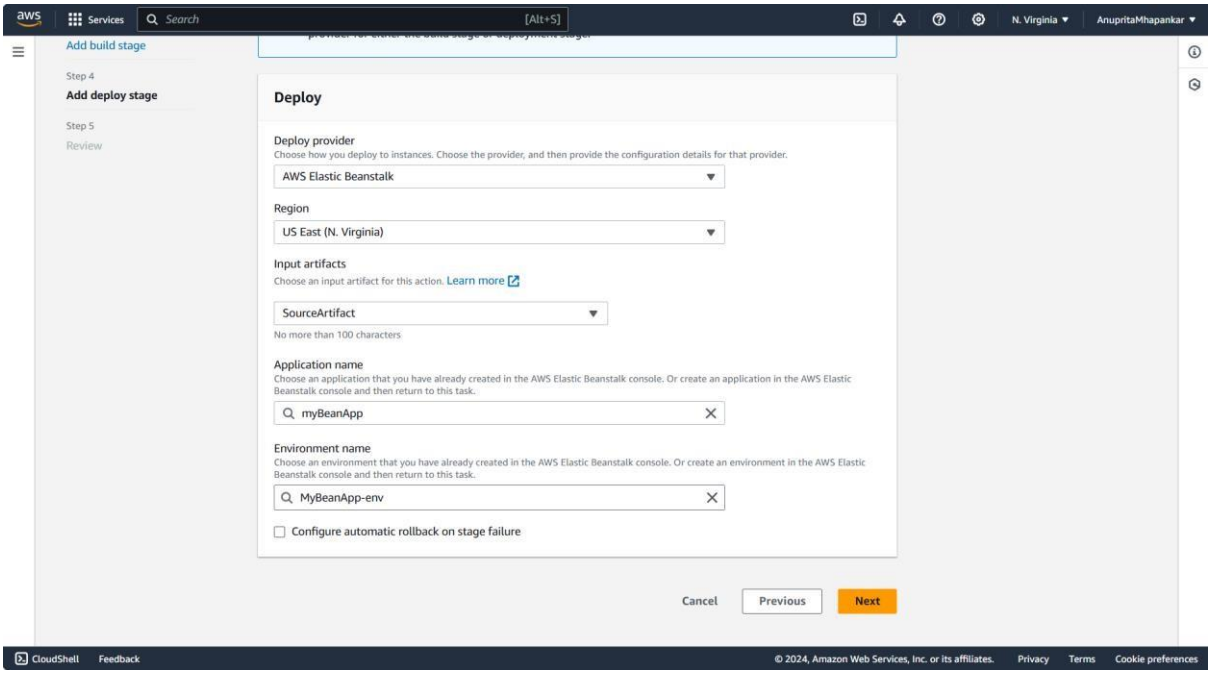

Authorize AWS Connector for GitHub - Personal - Microsoft Edge
https://github.com/login/oauth/authorize?client_id=Iv1.ab636337c58c3ec...
AWS Connector for GitHub by Amazon Web Services
would like permission to:
Verify your GitHub identity (Anuprita579)
Know which resources you can access
Act on your behalf
Learn more
Learn more about AWS Connector for GitHub
Cancel
Authorize AWS Connector for GitHub
Authorizing will redirect to
https://redirect.codestar.aws
Not owned or operated by GitHub
Created 4 years ago



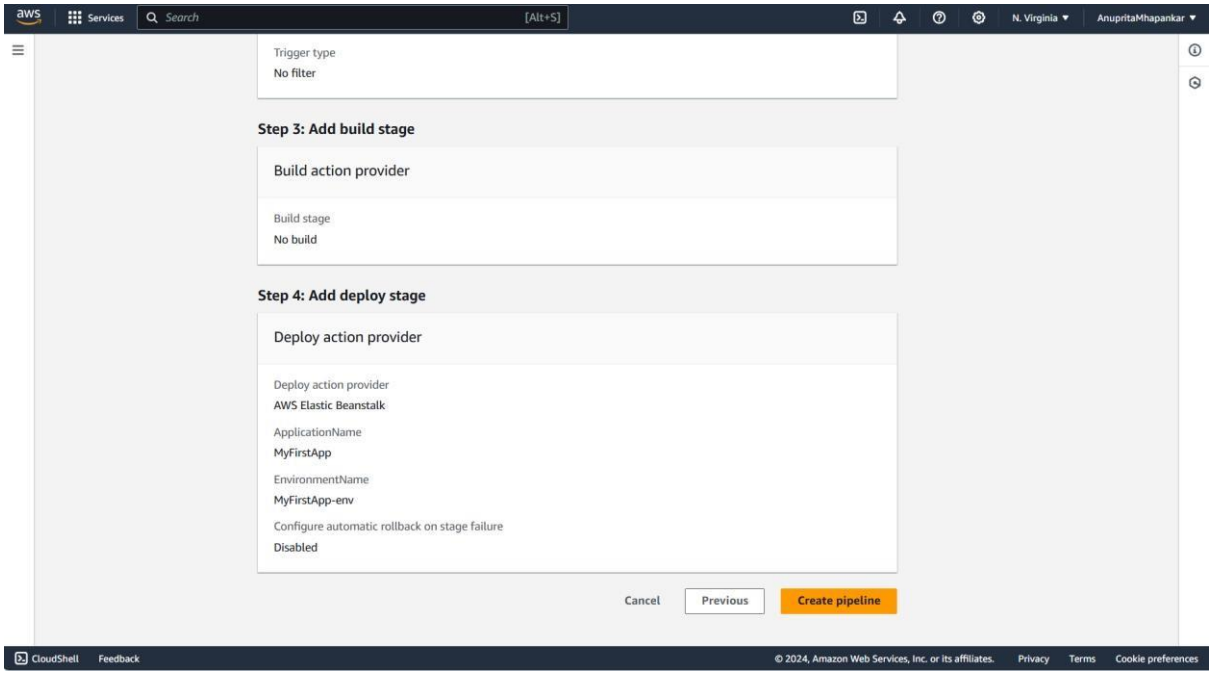
5. Once the connection is established from the drop down menu select the repository and the branch



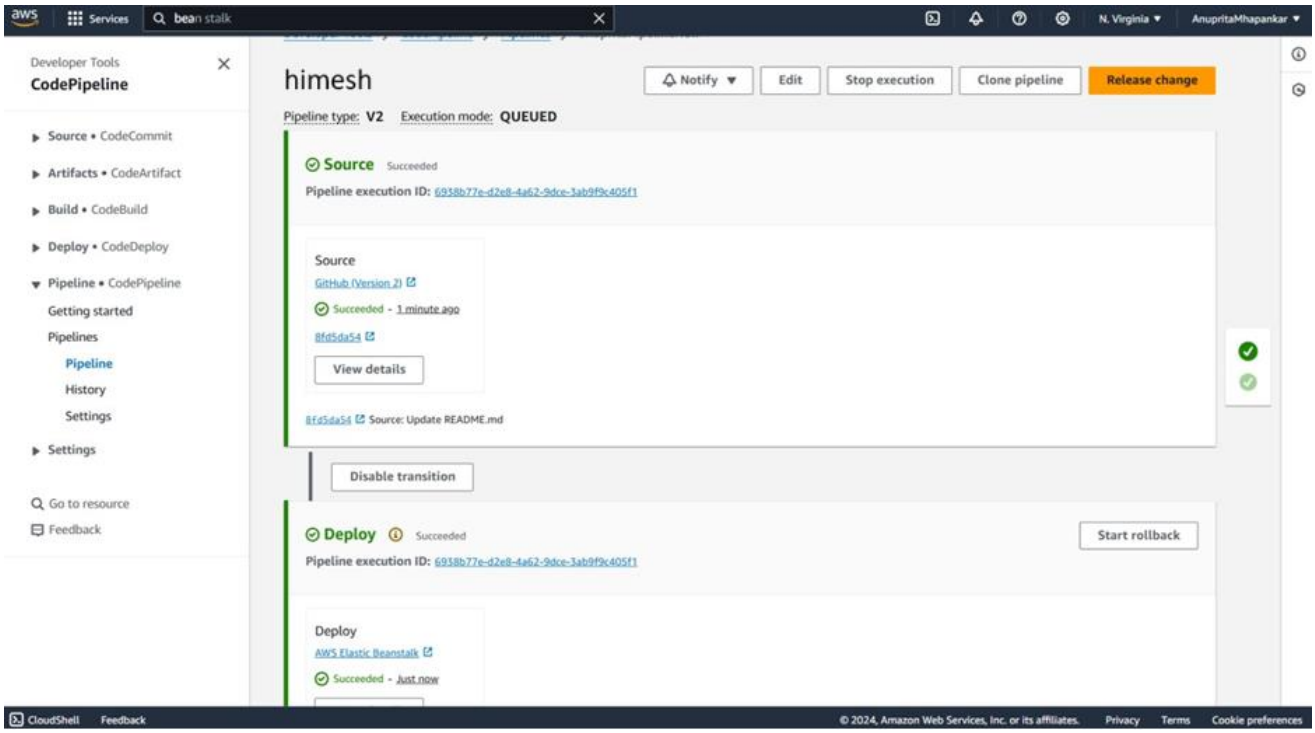
6. Skip the build stage



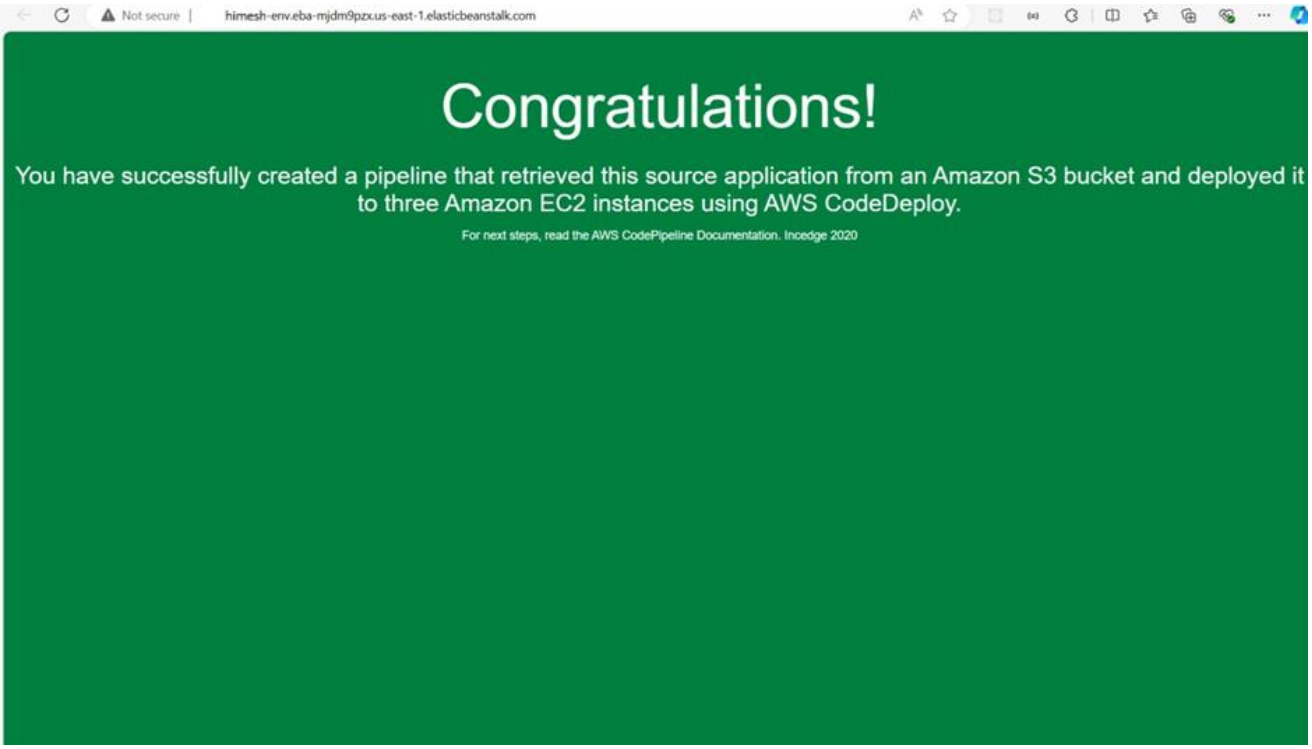
7. Review the settings and click on create pipeline



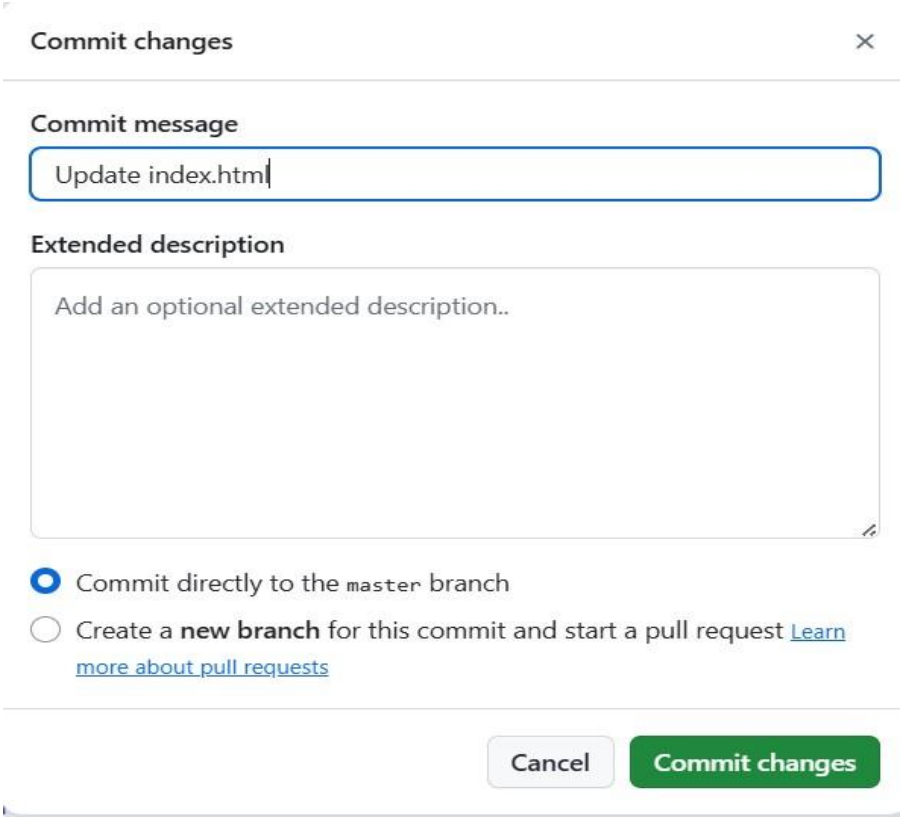
8. Check the URL provided in the EBS environment.
9.



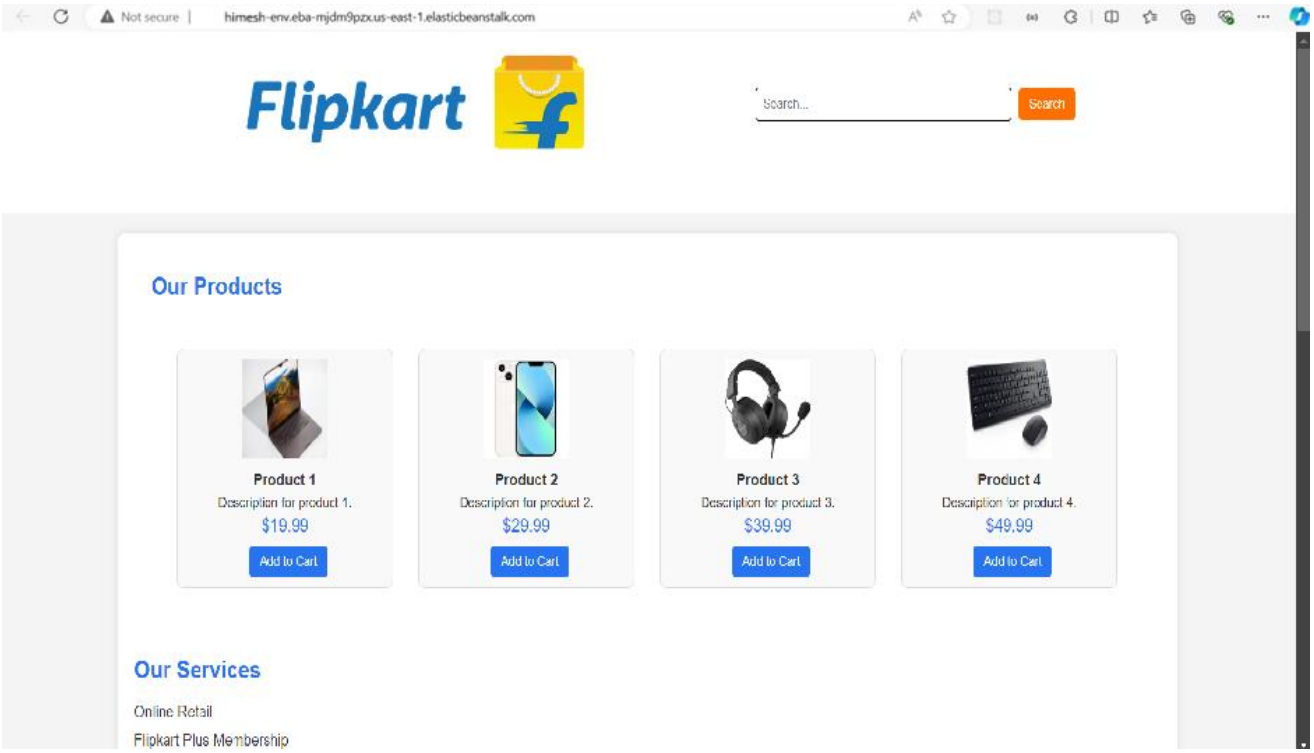
9. The website is hosted from the forked repo in our beanstalk environment



10. Now, Edit index.html file and then commit the changes



11. Visit the deployed link again, the changes will be reflected in the website.



Name : Himesh Pathai
Div : D10A
Roll No. : 35

Advanced DevOps Lab
Experiment:3

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

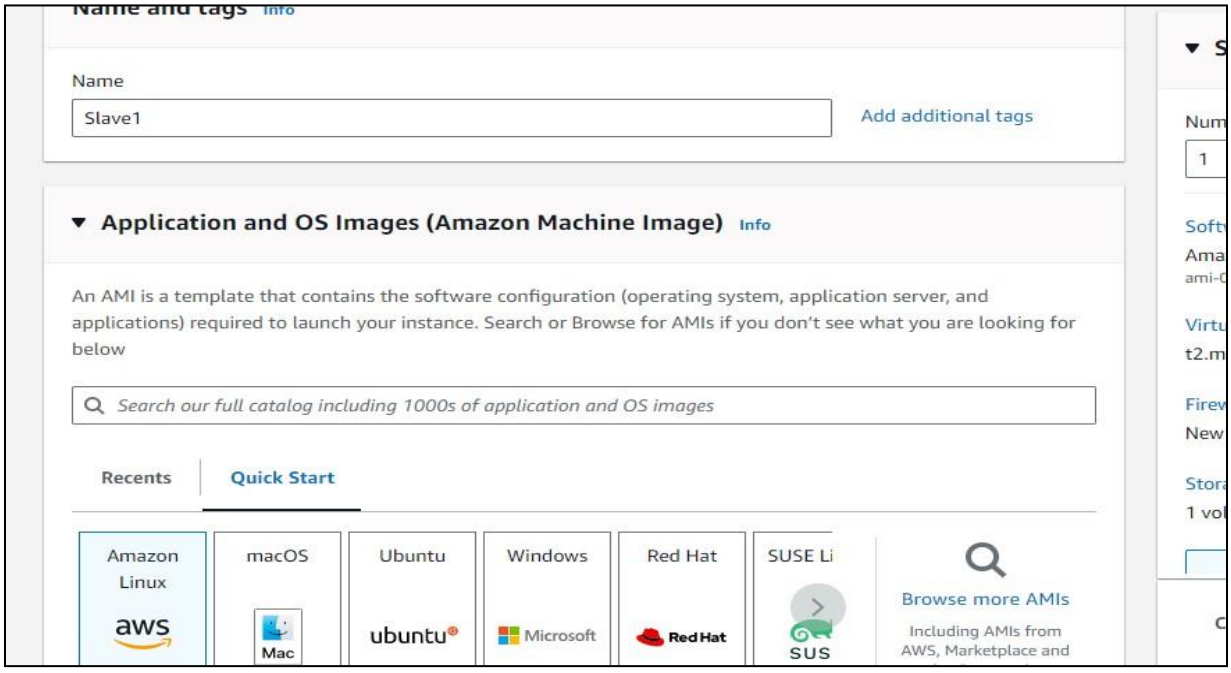
Theory:

To understand Kubernetes Cluster Architecture and how to install and spin up a Kubernetes cluster on Linux machines or cloud platforms, it's essential to grasp the fundamental components and design principles of Kubernetes.

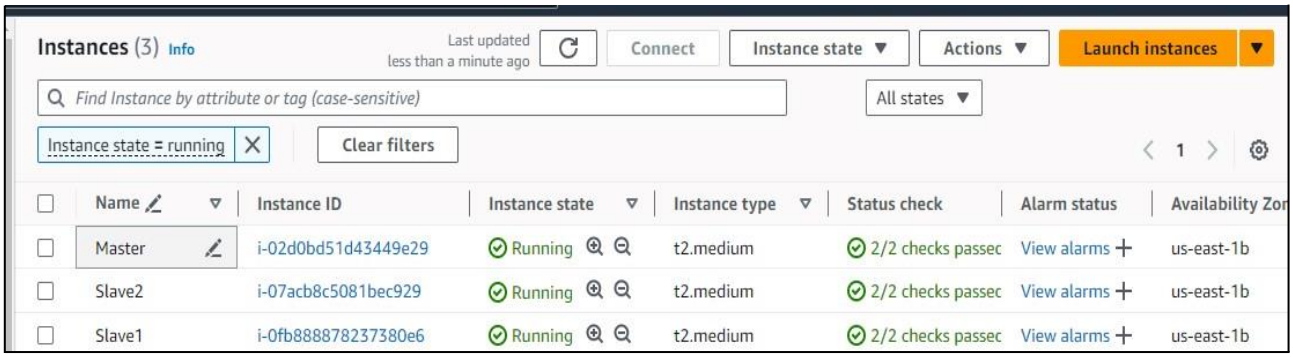
Steps:

1. Create 3 EC2 Ubuntu Instances on AWS.

(Name 1 as Master, the other 2 as Slave1 and Slave2)

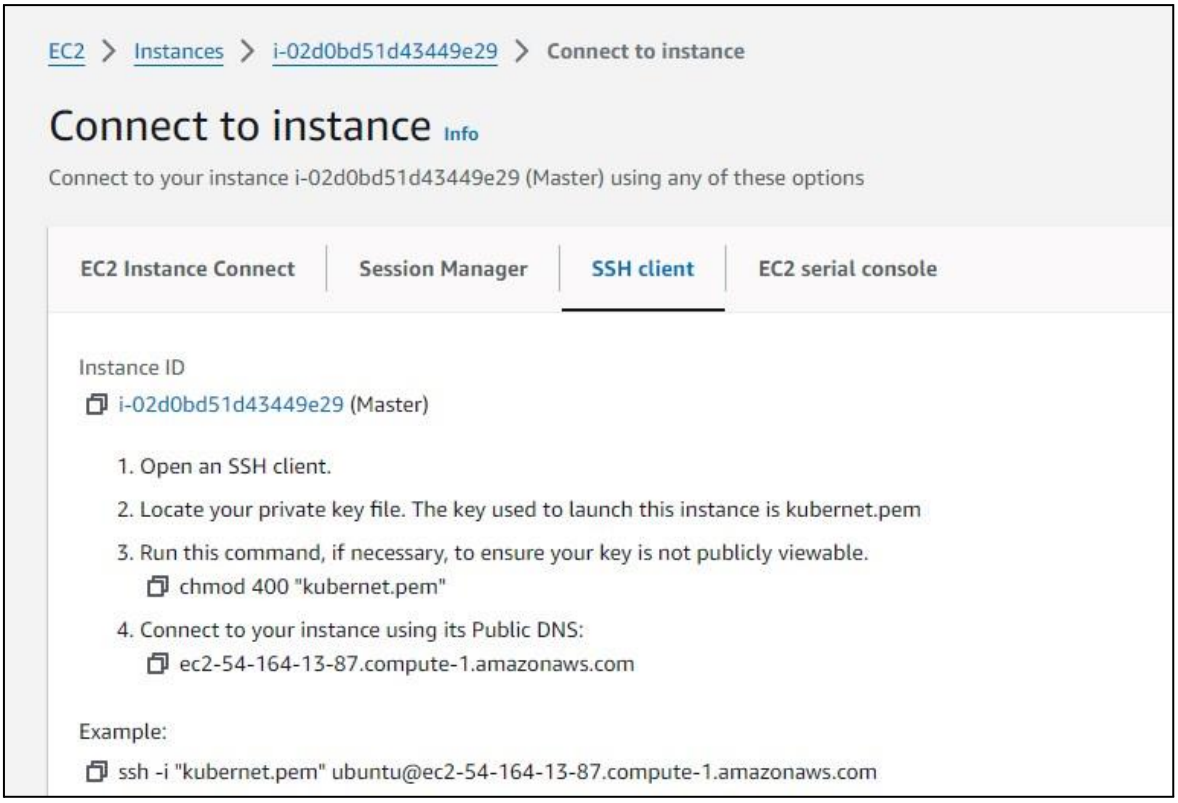


1. Created a master and 2 slaves:



2. Now click on connect to instance, then click on SSH client.

3. Now copy the ssh from the example and paste it on command prompt.(I used gitbash)



Commands:

4. Now since you are on GitBash, first type sudo su to perform the command as a root user.

5. After this type on all 3 machines
Yum install docker -y

```
[ec2-user@ip-172-31-84-37 ~]$ sudo su
[root@ip-172-31-84-37 ec2-user]# yum install docker -y
Last metadata expiration check: 0:18:22 ago on Thu Aug 29 08:52:52 2024.
Dependencies resolved.
================================================================================
 Package                 Architecture    Version                 Repository    Size
================================================================================
Installing:
 docker                  x86_64          25.0.6-1.amzn2023.0.1   amazonlinux   44 M
Installing dependencies:
 containerd              x86_64          1.7.20-1.amzn2023.0.1   amazonlinux   35 M
 iptables-libs           x86_64          1.8.8-3.amzn2023.0.2    amazonlinux  401 k
 iptables-nft            x86_64          1.8.8-3.amzn2023.0.2    amazonlinux  183 k
 libcgroup               x86_64          3.0-1.amzn2023.0.1      amazonlinux   75 k
 libnetfilter_conntrack  x86_64          1.0.8-2.amzn2023.0.2    amazonlinux   58 k
 libnfnetlink            x86_64          1.0.1-19.amzn2023.0.2   amazonlinux   30 k
 libnftnl                x86_64          1.2.2-2.amzn2023.0.2    amazonlinux   84 k
 pigz                    x86_64          2.5-1.amzn2023.0.3      amazonlinux   83 k
 runc                    x86_64          1.1.11-1.amzn2023.0.1   amazonlinux  3.0 M
```

```
  Running scriptlet: docker-25.0.6-1.amzn2023.0.1.x86_64
  Installing       : docker-25.0.6-1.amzn2023.0.1.x86_64
  Running scriptlet: docker-25.0.6-1.amzn2023.0.1.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.amzn2023.0.1.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-1.1.11-1.amzn2023.0.1.x86_64

Installed:
  containerd-1.7.20-1.amzn2023.0.1.x86_64       docker-25.0.6-1.amzn2023.0.1.x86_64      iptables-libs-1.8.8-3.amzn2023.0.2.x86_64
  iptables-nft-1.8.8-3.amzn2023.0.2.x86_64      libcgroup-3.0-1.amzn2023.0.1.x86_64      libnetfilter_conntrack-1.0.8-2.amzn2023.0
  libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64     libnftnl-1.2.2-2.amzn2023.0.2.x86_64     pigz-2.5-1.amzn2023.0.3.x86_64
  runc-1.1.11-1.amzn2023.0.1.x86_64

Complete!
```

6. To start the docker on master and slave perform this command: Systemctl start docker

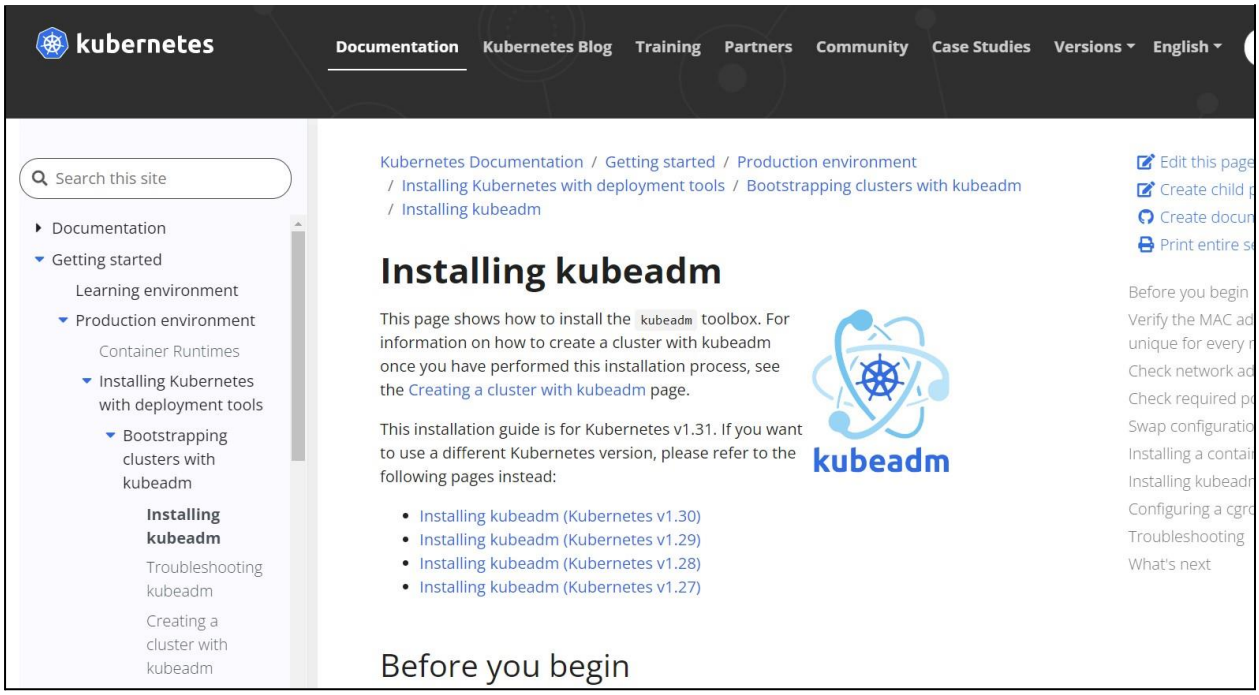
Extra

7. To check if docker is Installed successfully:
Docker -v or Docker –version

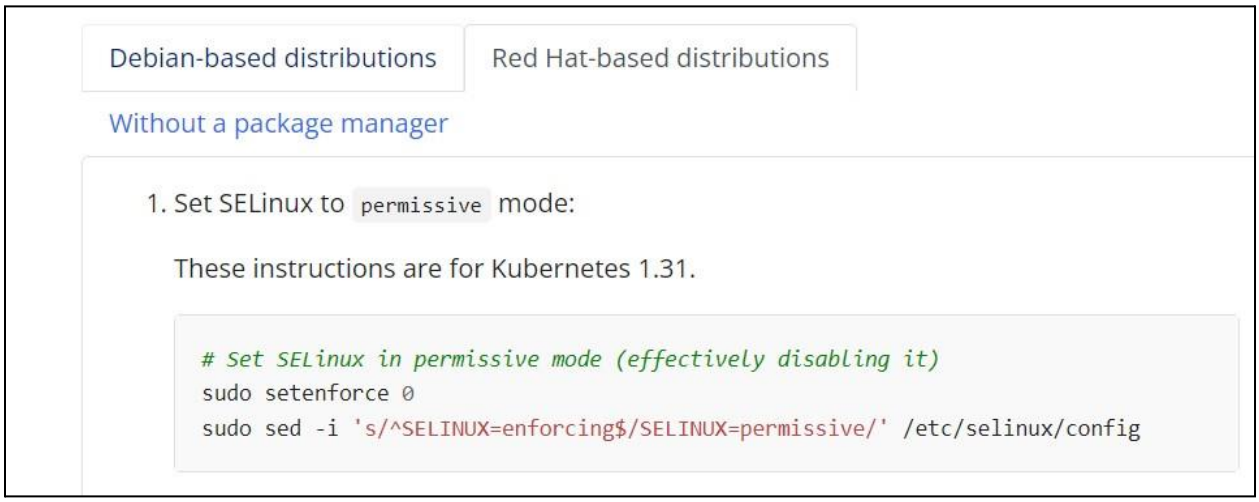
```
[root@ip-172-31-84-37 ec2-user]# systemctl start docker
[root@ip-172-31-84-37 ec2-user]# sudo su
[root@ip-172-31-84-37 ec2-user]# yum repolist
repo id                                                 repo name
amazonlinux                                             Amazon Linux 2023 repository
kernel-livepatch                                        Amazon Linux 2023 Kernel Livepatch repository
[root@ip-172-31-84-37 ec2-user]# docker --version
Docker version 25.0.5, build 5dc9bcc
```

8. Now to install kubeadm on master and slaves :

Installing kubeadm:
Go the official documentation off kubeadm.



9. Scroll down and select Red Hat based distributions:



10. Now copy the command on all 3 machines:

    Set SELinux to permissive mode:

    These instructions are for Kubernetes 1.31.

    # Set SELinux in permissive mode (effectively disabling it) sudo setenforce 0
    sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/'
    /etc/selinux/config

11. Now copy all the commands on the GitBash on all the 3 machines:

    # This overwrites any existing configuration in /etc/yum.repos.d/kubernetes.repo 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 kubelet, kubeadm and kubectl: sudo yum install -y kubelet kubeadm

    kubectl --disableexcludes=kubernetes #(Optional) Enable the kubelet service

    before running kubeadm:

    sudo systemctl enable --now kubelet

```
  Installing       : kubectl-1.31.0-150500.1.1.x86_64
  Running scriptlet: kubectl-1.31.0-150500.1.1.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        : socat-1.7.4.2-1.amzn2023.0.2.x86_64
  Verifying        : cri-tools-1.31.1-150500.1.1.x86_64
  Verifying        : kubeadm-1.31.0-150500.1.1.x86_64
  Verifying        : kubectl-1.31.0-150500.1.1.x86_64
  Verifying        : kubelet-1.31.0-150500.1.1.x86_64
  Verifying        : kubernetes-cni-1.5.0-150500.2.1.x86_64

Installed:
  conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64                              cr
  kubeadm-1.31.0-150500.1.1.x86_64                                         ku
  kubelet-1.31.0-150500.1.1.x86_64                                         ku
  libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64                       li
  libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64                           so

Complete!
[root@ip-172-31-84-37 ec2-user]# sudo systemctl enable --now kubelet
```

12. Type yum repolist to check the repository of kubernetes

```
[root@ip-172-31-84-143 ec2-user]# yum repolist
repo id                                                    repo name
amazonlinux                                                Amazon Linux 2023 repository
kernel-livepatch                                           Amazon Linux 2023 Kernel Livepatch repository
kubernetes                                                 Kubernetes
```

EXTRA

Got an error in initialization kubeadm

```
[root@ip-172-31-31-240 ec2-user]# kubeadm init
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
W0908 11:25:45.820964    2320 checks.go:1080] [preflight] WARNING: Couldn't create the interface used for talking t
RI runtime service: validate service connection: validate CRI v1 runtime API for endpoint "unix:///var/run/containe
ble desc = connection error: desc = "transport: Error while dialing: dial unix /var/run/containerd/containerd.sock:
        [WARNING FileExisting-tc]: tc not found in system path
error execution phase preflight: [preflight] Some fatal errors occurred:
        [ERROR FileContent--proc-sys-net-ipv4-ip_forward]: /proc/sys/net/ipv4/ip_forward contents are not set to 1
[preflight] If you know what you are doing, you can make a check non-fatal with `--ignore-preflight-errors=...`
To see the stack trace of this error execute with --v=5 or higher
```

Error was resolved:
(after again starting from scratch)

13. Initialize the kubeadm by the command kubeadm init only on master:

Kubeadm initialized successfully:

```
[root@ip-172-31-26-66 ec2-user]# kubeadm init
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
        [WARNING FileExisting-socat]: socat not found in system path
        [WARNING FileExisting-tc]: tc not found in system path
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your intern
[preflight] You can also perform this action beforehand using 'kubeadm config imag
W0912 06:07:49.475553   28037 checks.go:846] detected that the sandbox image "regi
 that used by kubeadm.It is recommended to use "registry.k8s.io/pause:3.10" as the
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [ip-172-31-26-66.ec2.intern
efault.svc.cluster.local] and IPs [10.96.0.1 172.31.26.66]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
```

14. After this we will get 3 things:

- The directory
- Some export Statement
- The most important thing - the token to connect the slaves with the master.

15. Copy them one by one and paste it on the slaves:

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

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

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

  export KUBECONFIG=/etc/kubernetes/admin.conf

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

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

kubeadm join 172.31.26.66:6443 --token grw4r4.gb3kkhb7392dnvjp \
        --discovery-token-ca-cert-hash sha256:b61f1de7eedb2c0dc0cc237d4629e9631920b63dd6634c3e22e76aaa36d01920
```

16. After pasting type kubectl get nodes:

The nodes are connected successfully:

```
ubuntu@ip-172-31-17-23:~$ kubectl get nodes
NAME                STATUS   ROLES           AGE     VERSION
ip-172-31-17-23     Ready    control-plane   3m56s   v1.29.0
ip-172-31-18-12     Ready    <none>          37s     v1.29.0
ip-172-31-26-153    Ready    <none>          24s     v1.29.0
ubuntu@ip-172-31-17-23:~$ kubectl get nodes
NAME                STATUS   ROLES           AGE     VERSION
ip-172-31-17-23     Ready    control-plane   9m34s   v1.29.0
ip-172-31-18-12     Ready    <none>          6m15s   v1.29.0
ip-172-31-26-153    Ready    <none>          6m2s    v1.29.0
ubuntu@ip-172-31-17-23:~$
```

Name : Himesh Pathai
Div : D10A
Roll No. : 35

Advanced DevOps Lab
Experiment 4

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

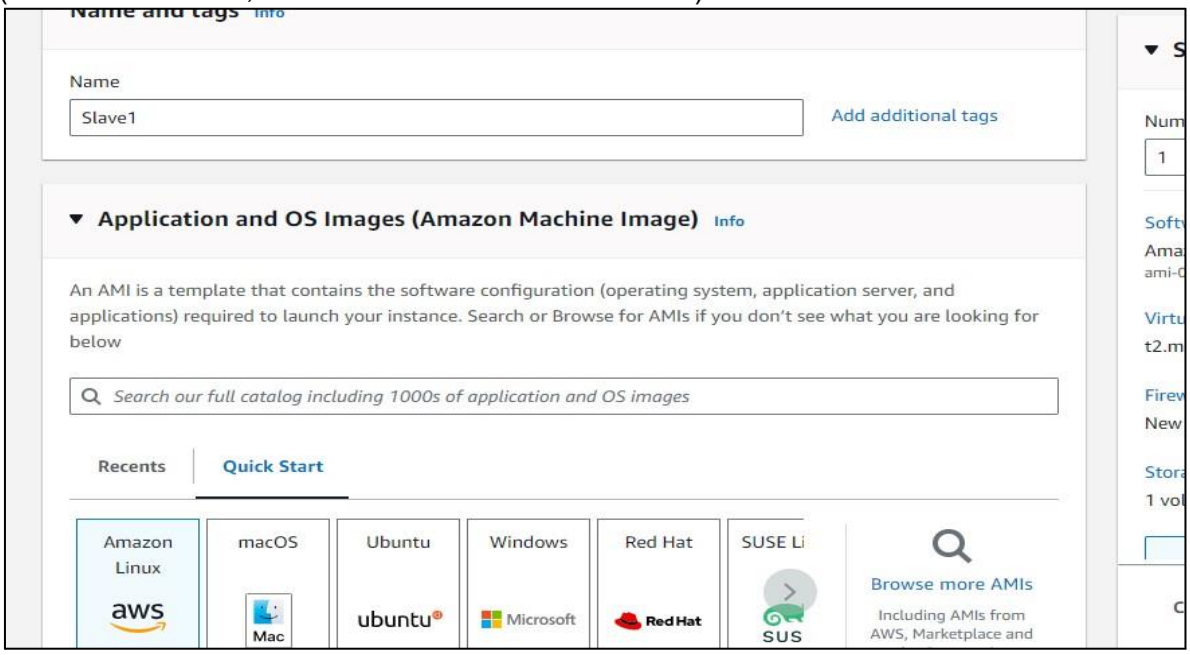
Theory:

Kubernetes, often referred to as K8s, is an open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications. Originally developed by Google, it has become the industry standard for managing container workloads due to its flexibility and robust features.

Steps:

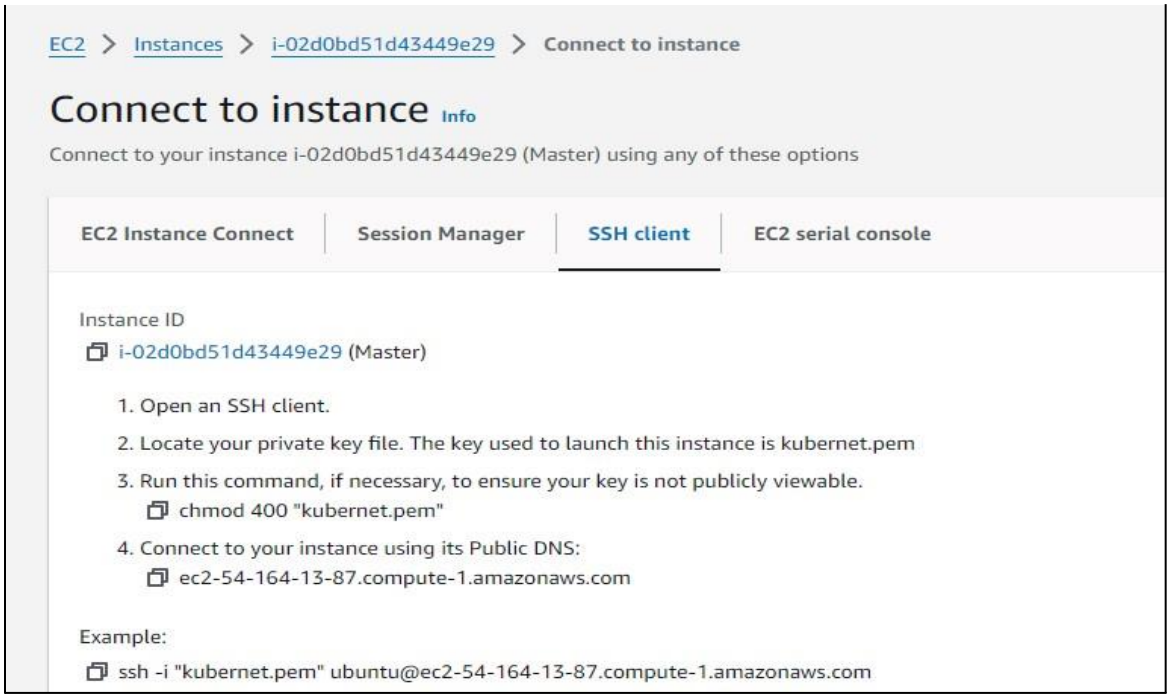
1. Create 3 EC2 Ubuntu Instances on AWS.

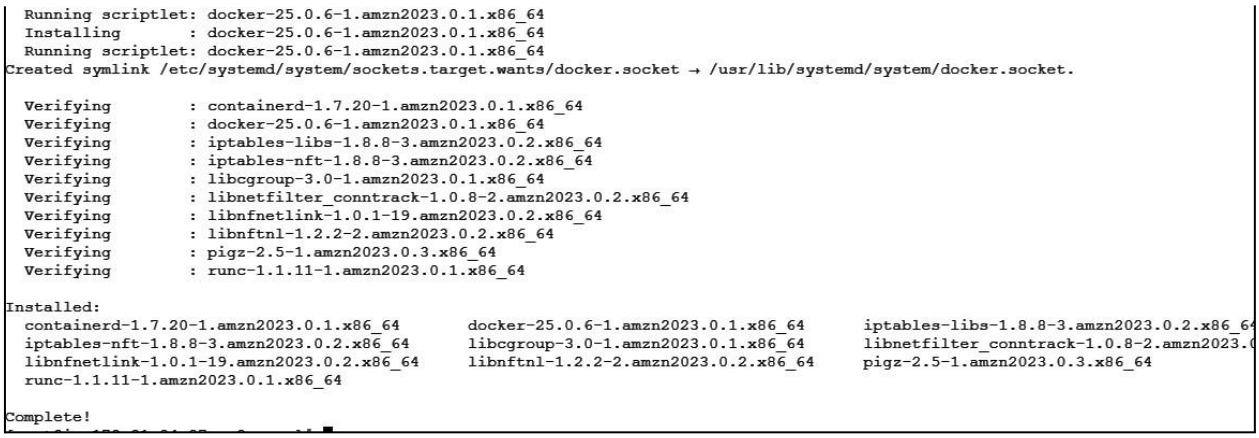
(Name 1 as Master, the other 2 as Slave1 and Slave2)



2. Now click on connect to instance, then click on SSH client.

3. Now copy the ssh from the example and paste it on command prompt.(I used gitbash)





Commands:

4. Now since you are on GitBash, first type sudo su to perform the command as a root user.

5. After this type on GitBash
Yum install docker -y

```
[ec2-user@ip-172-31-84-37 ~]$ sudo su
[root@ip-172-31-84-37 ec2-user]# yum install docker -y
Last metadata expiration check: 0:18:22 ago on Thu Aug 29 08:52:52 2024.
Dependencies resolved.
================================================================================
 Package                 Architecture        Version                 Repository      Size
================================================================================
Installing:
 docker                  x86_64              25.0.6-1.amzn2023.0.1   amazonlinux     44 M
Installing dependencies:
 containerd              x86_64              1.7.20-1.amzn2023.0.1   amazonlinux     35 M
 iptables-libs           x86_64              1.8.8-3.amzn2023.0.2    amazonlinux    401 k
 iptables-nft            x86_64              1.8.8-3.amzn2023.0.2    amazonlinux    183 k
 libcgroup               x86_64              3.0-1.amzn2023.0.1      amazonlinux     75 k
 libnetfilter_conntrack  x86_64              1.0.8-2.amzn2023.0.2    amazonlinux     58 k
 libnfnetlink            x86_64              1.0.1-19.amzn2023.0.2   amazonlinux     30 k
 libnftnl                x86_64              1.2.2-2.amzn2023.0.2    amazonlinux     84 k
 pigz                    x86_64              2.5-1.amzn2023.0.3      amazonlinux     83 k
 runc                    x86_64              1.1.11-1.amzn2023.0.1   amazonlinux    3.0 M
```

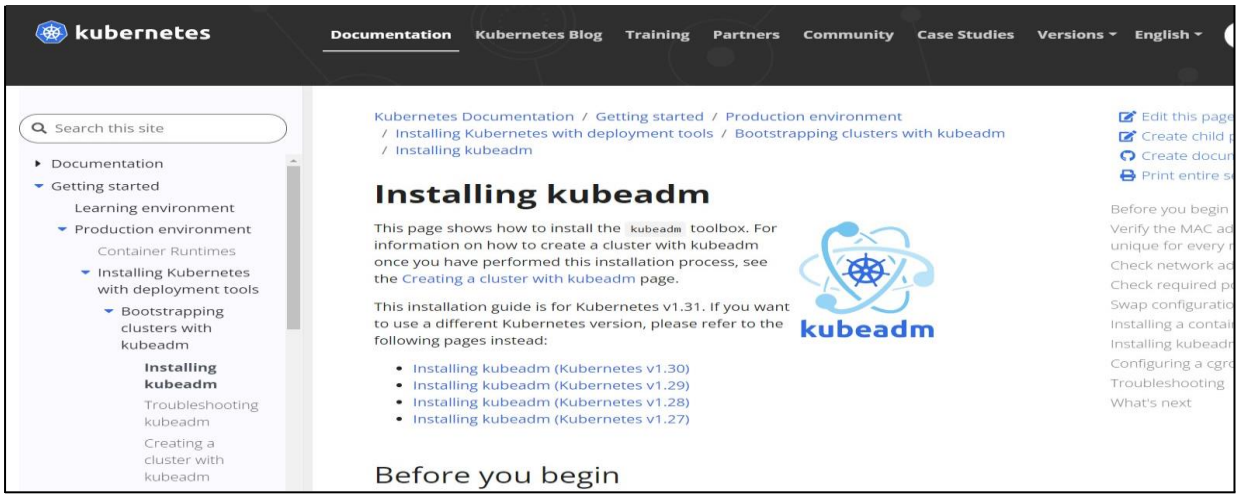
6. To start the docker perform this command: Systemctl start docker

7. To check if docker is Installed successfully:
Docker -v or Docker –version

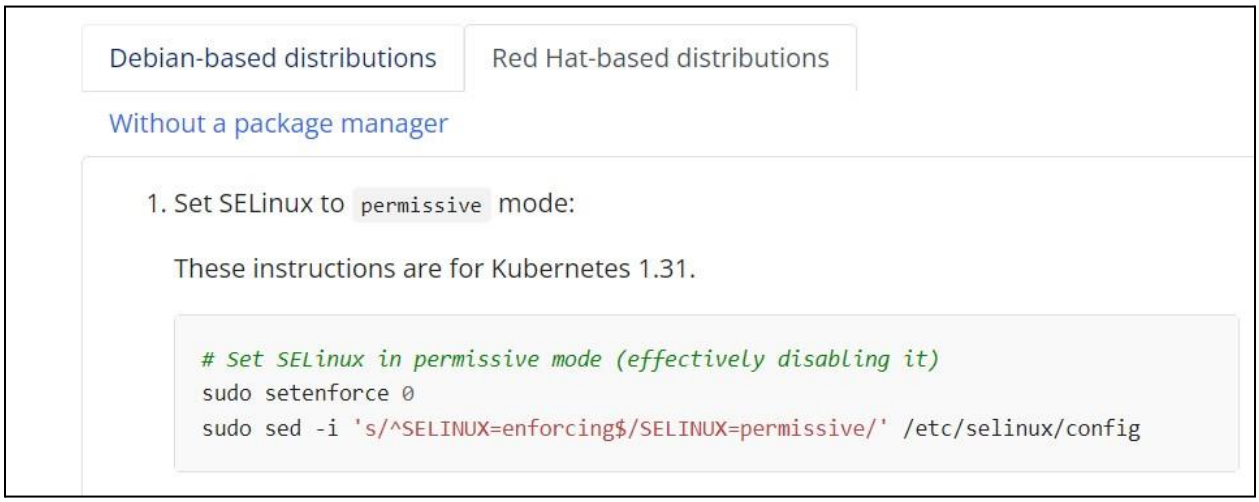
```
[root@ip-172-31-84-37 ec2-user]# systemctl start docker
[root@ip-172-31-84-37 ec2-user]# sudo su
[root@ip-172-31-84-37 ec2-user]# yum repolist
repo id                                                repo name
amazonlinux                                            Amazon Linux 2023 repository
kernel-livepatch                                       Amazon Linux 2023 Kernel Livepatch repository
[root@ip-172-31-84-37 ec2-user]# docker --version
Docker version 25.0.5, build 5dc9bcc
```

8. Now to install kubeadm :

Installing kubeadm:
Go the official documentation off kubeadm.



9. Scroll down and select Red Hat based distributions:



10. Now copy the command:

    Set SELinux to permissive mode:

    These instructions are for Kubernetes 1.31.

    # Set SELinux in permissive mode (effectively disabling it) sudo setenforce 0
    sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/'
    /etc/selinux/config

11. Now copy all the commands on the GitBash:

```
# This overwrites any existing configuration in /etc/yum.repos.d/kubernetes.repo
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.ke
y exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF

#Install kubelet, kubeadm and kubectl: sudo yum install -y kubelet

kubeadm kubectl --disableexcludes=kubernetes #(Optional) Enable the

kubelet service before running kubeadm:

sudo systemctl enable --now kubelet
```

```
  Installing       : kubeadm-1.31.0-150500.1.1.x86_64
  Installing       : kubectl-1.31.0-150500.1.1.x86_64
  Running scriptlet: kubectl-1.31.0-150500.1.1.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        : socat-1.7.4.2-1.amzn2023.0.2.x86_64
  Verifying        : cri-tools-1.31.1-150500.1.1.x86_64
  Verifying        : kubeadm-1.31.0-150500.1.1.x86_64
  Verifying        : kubectl-1.31.0-150500.1.1.x86_64
  Verifying        : kubelet-1.31.0-150500.1.1.x86_64
  Verifying        : kubernetes-cni-1.5.0-150500.2.1.x86_64

Installed:
  conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64                              cr
  kubeadm-1.31.0-150500.1.1.x86_64                                         ku
  kubelet-1.31.0-150500.1.1.x86_64                                         ku
  libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64                       li
  libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64                           so

Complete!
[root@ip-172-31-84-37 ec2-user]# sudo systemctl enable --now kubelet
```

12. Type yum repolist to check the repository of kubernetes

```
[root@ip-172-31-84-143 ec2-user]# yum repolist
repo id                                                    repo name
amazonlinux                                                Amazon Linux 2023 repository
kernel-livepatch                                           Amazon Linux 2023 Kernel Livepatch repository
kubernetes                                                 Kubernetes
```

EXTRA

Got an error in initialization kubeadm

```
[root@ip-172-31-31-240 ec2-user]# kubeadm init
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
W0908 11:25:45.820964    2320 checks.go:1080] [preflight] WARNING: Couldn't create the interface used for talking t
RI runtime service: validate service connection: validate CRI v1 runtime API for endpoint "unix:///var/run/containe
ble desc = connection error: desc = "transport: Error while dialing: dial unix /var/run/containerd/containerd.sock:
        [WARNING FileExisting-tc]: tc not found in system path
error execution phase preflight: [preflight] Some fatal errors occurred:
        [ERROR FileContent--proc-sys-net-ipv4-ip_forward]: /proc/sys/net/ipv4/ip_forward contents are not set to 1
[preflight] If you know what you are doing, you can make a check non-fatal with `--ignore-preflight-errors=...`
To see the stack trace of this error execute with --v=5 or higher
```

Error was resolved:
(after again starting from scratch)

13. Initialize the kubeadm by the command kubeadm init :

Kubeadm initialized successfully:

```
[root@ip-172-31-26-66 ec2-user]# kubeadm init
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
        [WARNING FileExisting-socat]: socat not found in system path
        [WARNING FileExisting-tc]: tc not found in system path
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your intern
[preflight] You can also perform this action beforehand using 'kubeadm config imag
W0912 06:07:49.475553   28037 checks.go:846] detected that the sandbox image "regi
 that used by kubeadm.It is recommended to use "registry.k8s.io/pause:3.10" as the
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [ip-172-31-26-66.ec2.intern
efault.svc.cluster.local] and IPs [10.96.0.1 172.31.26.66]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
```

14. After this we will get 3 things:

- The directory
- Some export Statement
- The most important thing - the token to connect the slaves with the master.

15. Copy them

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

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

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

  export KUBECONFIG=/etc/kubernetes/admin.conf

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

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

kubeadm join 172.31.26.66:6443 --token grw4r4.gb3kkhb7392dnvjp \
        --discovery-token-ca-cert-hash sha256:b61f1de7eedb2c0dc0cc237d4629e9631920b63dd6634c3e22e76aaa36d01920
```

16. After pasting type kubectl get nodes:

The nodes are connected successfully:

```
ubuntu@ip-172-31-17-23:~$ kubectl get nodes
NAME               STATUS   ROLES           AGE     VERSION
ip-172-31-17-23    Ready    control-plane   3m56s   v1.29.0
ip-172-31-18-12    Ready    <none>          37s     v1.29.0
ip-172-31-26-153   Ready    <none>          24s     v1.29.0
ubuntu@ip-172-31-17-23:~$ kubectl get nodes
NAME               STATUS   ROLES           AGE     VERSION
ip-172-31-17-23    Ready    control-plane   9m34s   v1.29.0
ip-172-31-18-12    Ready    <none>          6m15s   v1.29.0
ip-172-31-26-153   Ready    <none>          6m2s    v1.29.0
ubuntu@ip-172-31-17-23:~$
```

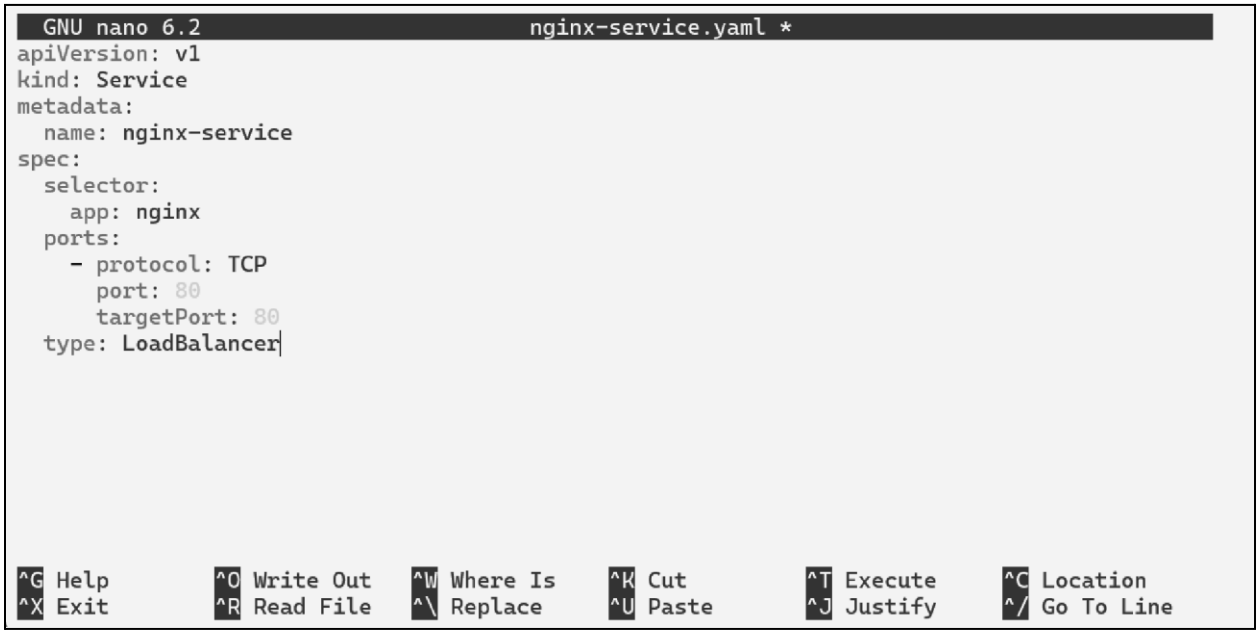
17. Create two YAML files named nginx-deployment.yaml and nginx-service.yaml
(I used nano editor for the same)

```
ubuntu@ip-172-31-17-23:~$ nano nginx-deployment.yaml
ubuntu@ip-172-31-17-23:~$ nano nginx-service.yaml
```

18. Then add the deployment and service configuration in it, respectively:

Deployment:

Service:



19. Now since we have configured our files we would now proceed for applying both the deployment and the service files.

Deployment :

```
ubuntu@ip-172-31-17-23:~$ kubectl apply -f nginx-deployment.yaml
deployment.apps/nginx-deployment created
```

Service:

```
ubuntu@ip-172-31-17-23:~$ kubectl apply -f nginx-service.yaml
service/nginx-service created
```

20. After deployment its time for verifying the same:

For deployment:

```
ubuntu@ip-172-31-17-23:~$ kubectl get deployments
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx              1/1     1            1           14m
nginx-deployment   2/2     2            2           39s
```

For services:

```
ubuntu@ip-172-31-17-23:~$ kubectl get services
NAME            TYPE           CLUSTER-IP       EXTERNAL-IP   PORT
(S)        AGE
kubernetes      ClusterIP      10.96.0.1        <none>        443/
TCP        70m
nginx           NodePort       10.109.245.143   <none>        80:3
0306/TCP   37m
nginx-service   LoadBalancer   10.99.247.105    <pending>     80:3
1130/TCP   36s
```

For pods:

```
ubuntu@ip-172-31-17-23:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AG
E
nginx-7854ff8877-mxrqg              1/1     Running   0          15
m
nginx-deployment-6b4d6fdbf-5rb6h    1/1     Running   0          65
s
nginx-deployment-6b4d6fdbf-6q2jj    1/1     Running   0          65
s
```

Extra:

```
ubuntu@ip-172-31-17-23:~$ kubectl get namespaces
NAME              STATUS   AGE
default           Active   55m
kube-node-lease   Active   55m
kube-public       Active   55m
kube-system       Active   55m
```

21. Now Lastly, port forward the deployment to your localhost so that you can view it.

```
ubuntu@ip-172-31-17-23:~$ kubectl port-forward service/nginx 8080:
80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
```

22. You can open the browser and check on

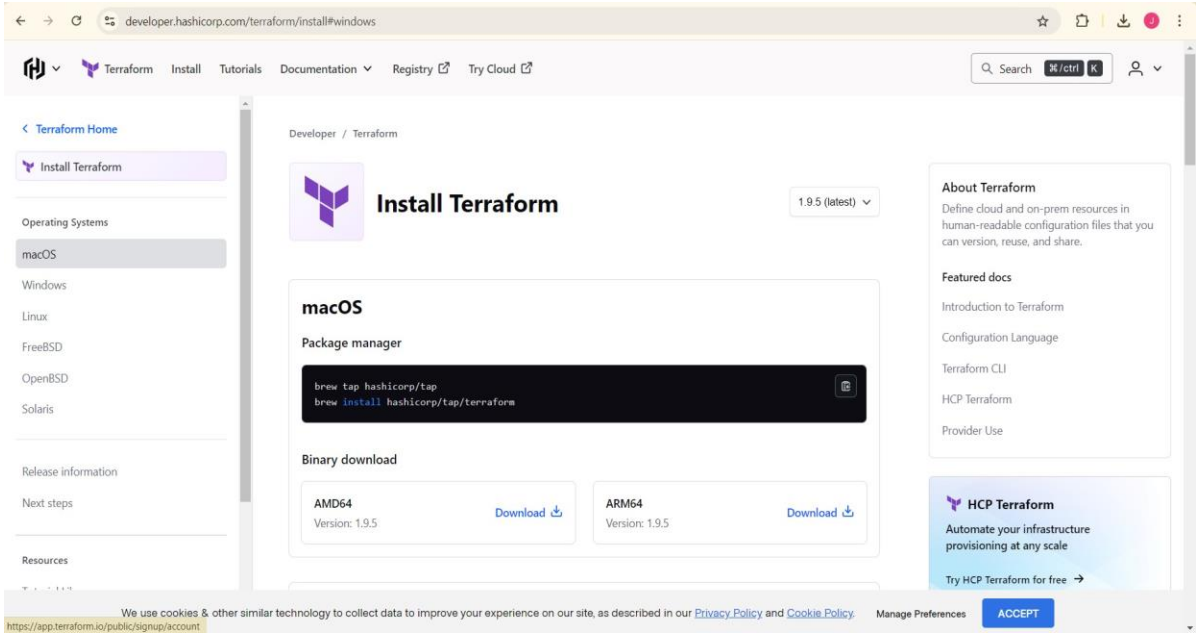
    http://localhost:8080

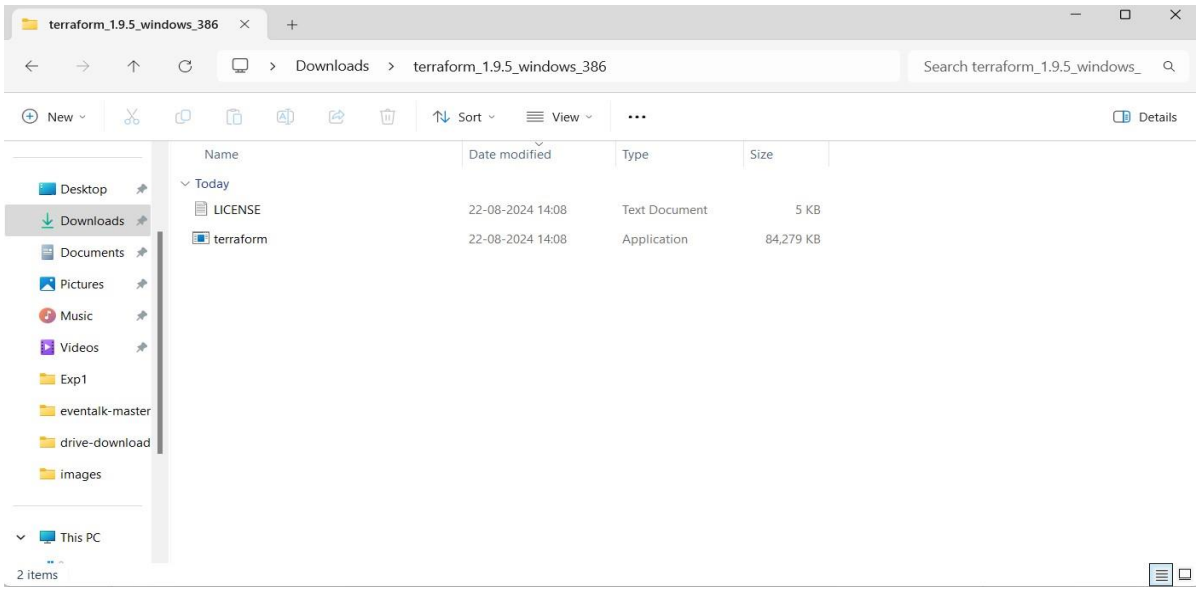
Name : Himesh Pathai
Roll No. : 35
Div. : D15A

Advance DevOps Exp : 5

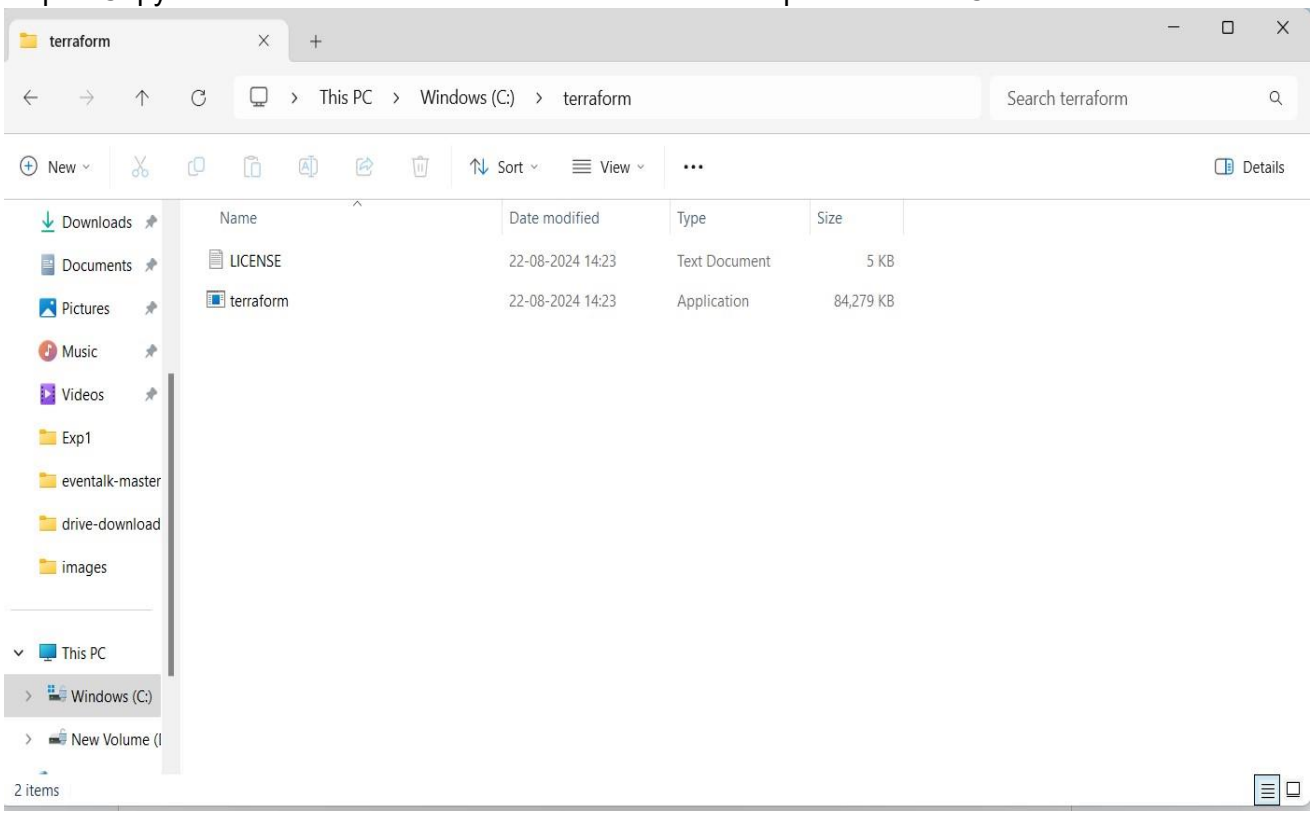
Aim: To understand terraform lifecycle, core concepts/terminologies and install it on a windows

Step1: Download Terraform from the official website

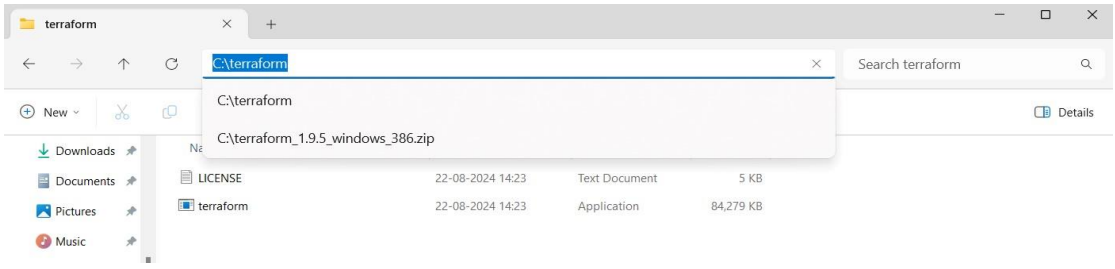




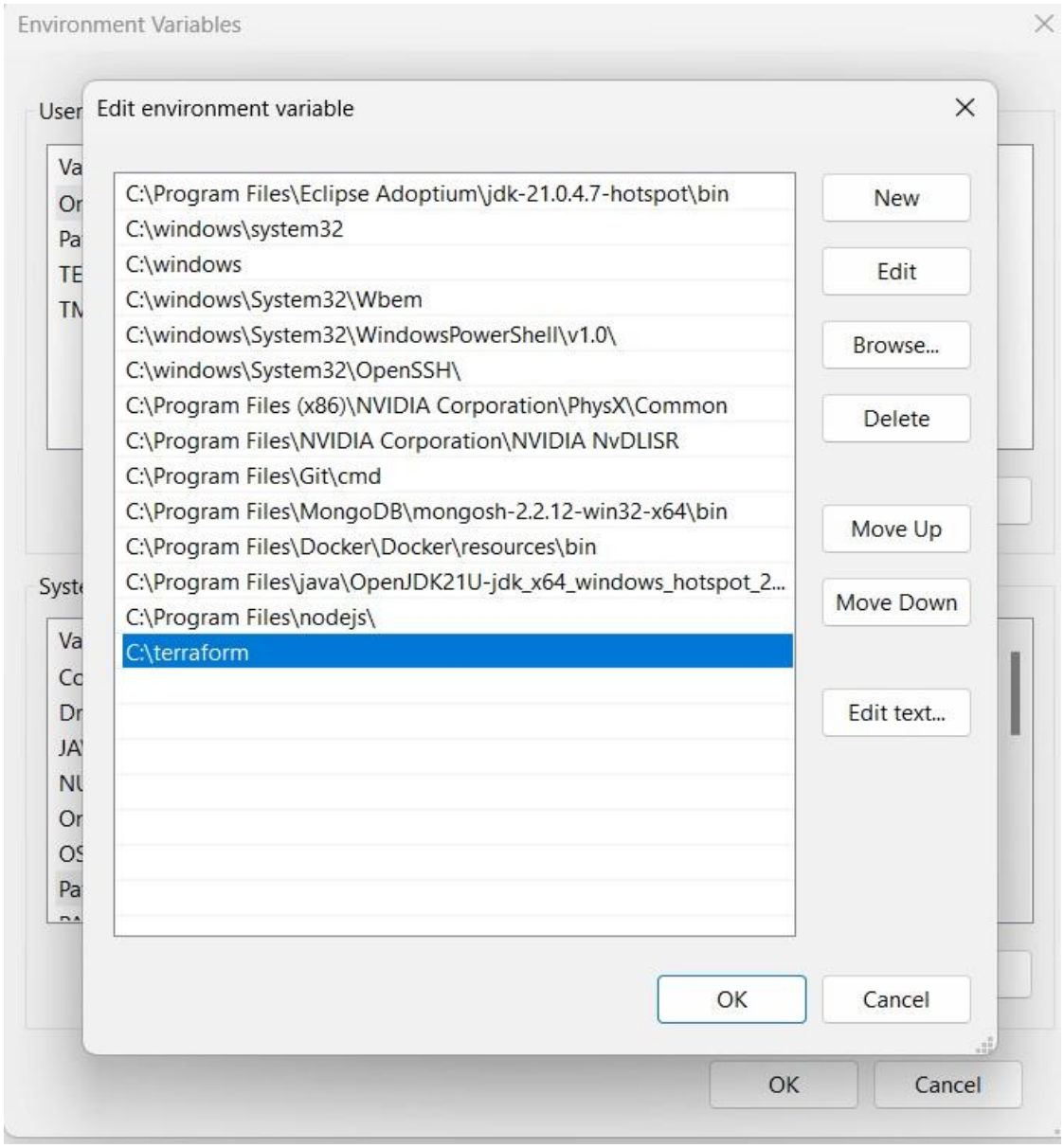
Step 2: Copy and extract Terraform from the downloads and paste it in the C drive



Step 3: Copy the file path to paste in the environment variables



Step 4: Set the environment variables for terraform



Step 5: Check weather the terraform is installed

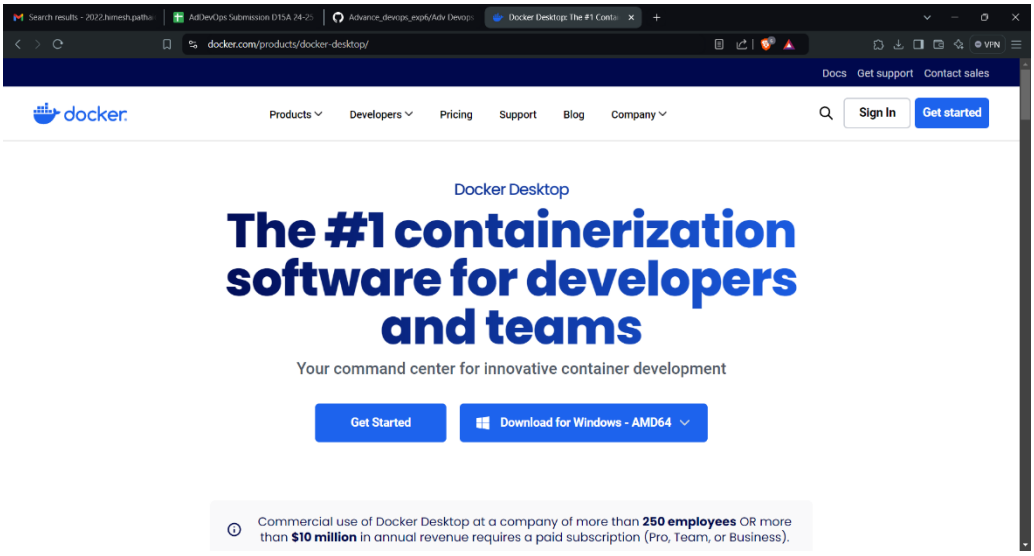
```
Microsoft Windows [Version 10.0.22631.4037]
(c) Microsoft Corporation. All rights reserved.

C:\Users\himes>terraform --version
Terraform v1.9.5
on windows_386

C:\Users\himes>
```

Name : Himesh Pathai
Roll No. : 35
Div. : D15A

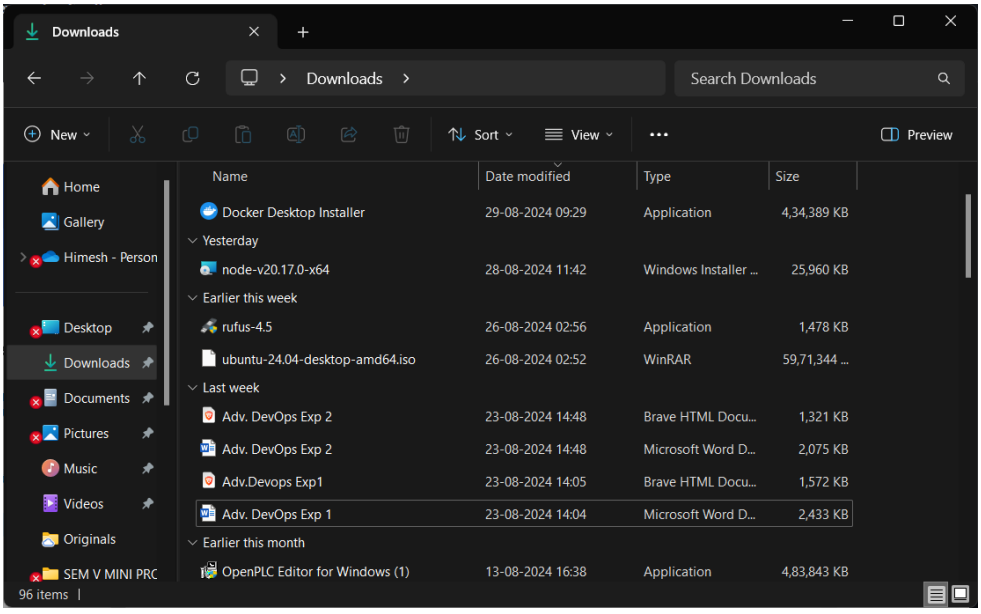
Advance DevOps Exp : 6

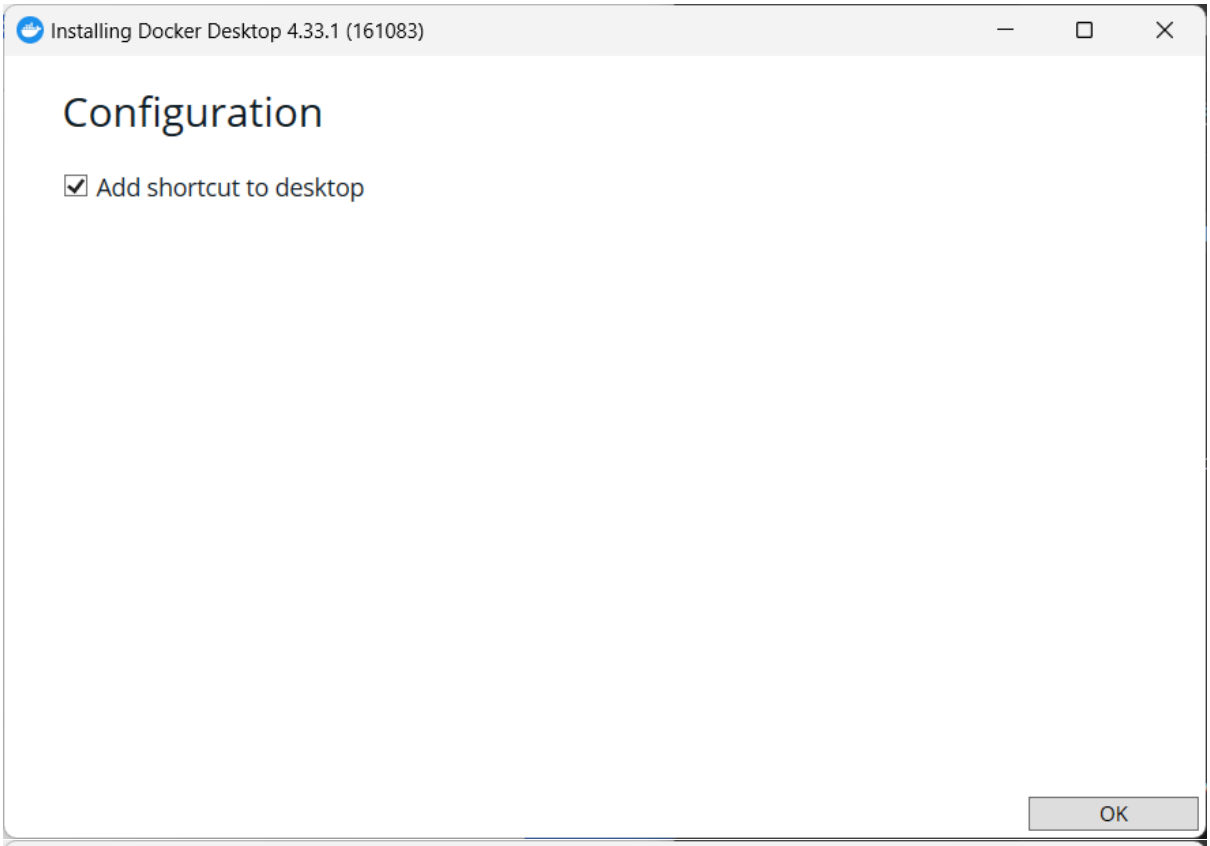
Aim: To Build, change, and destroy AWS / GCP /Microsoft Azure/ DigitalOcean infrastructure Using Terraform. (S3 bucket or Docker)

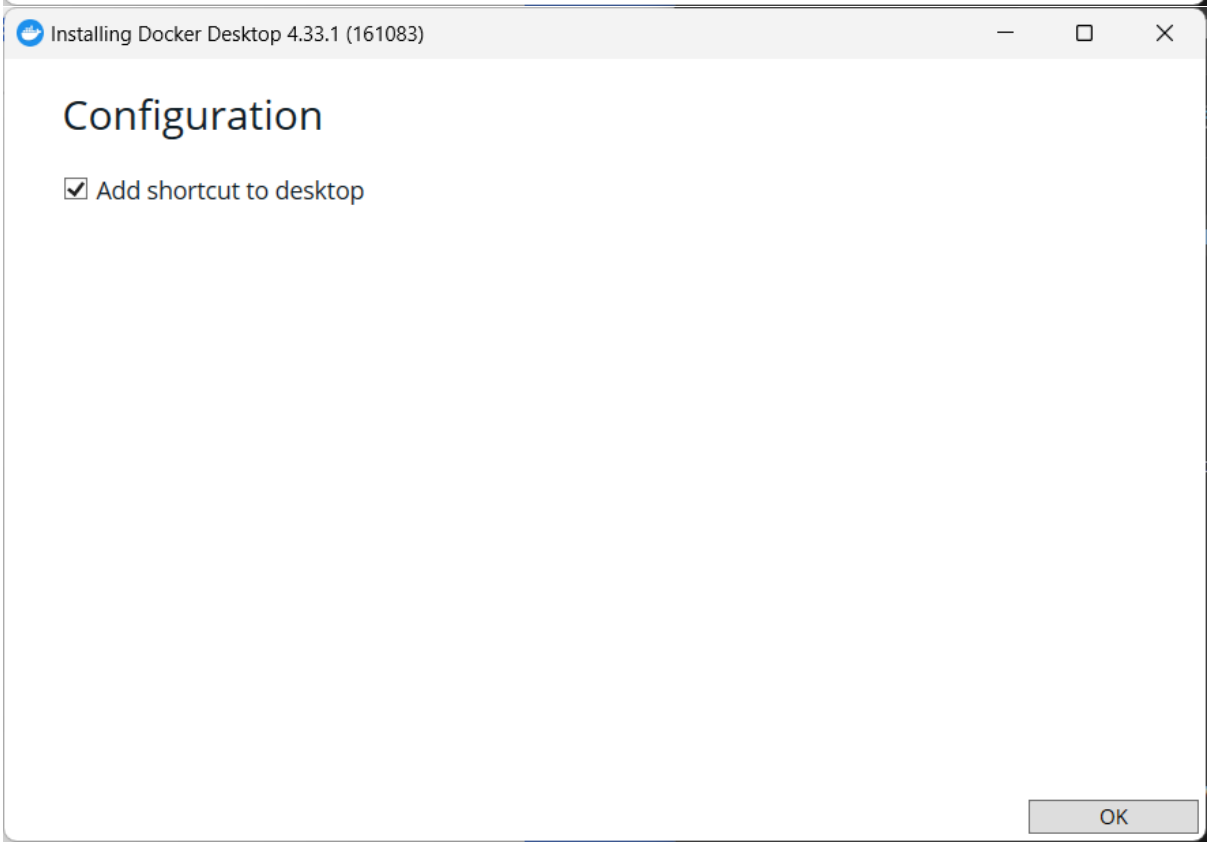
Step 1: Download Docker form www.docker.com



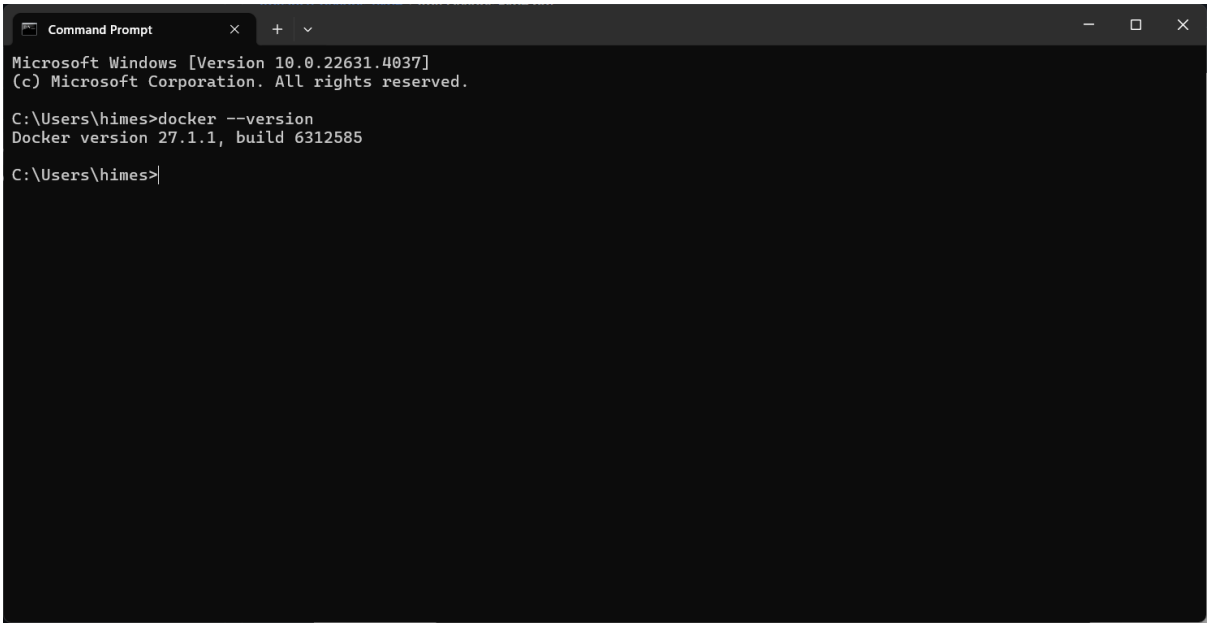
Step 2: Now, Docker is successfully downloaded.



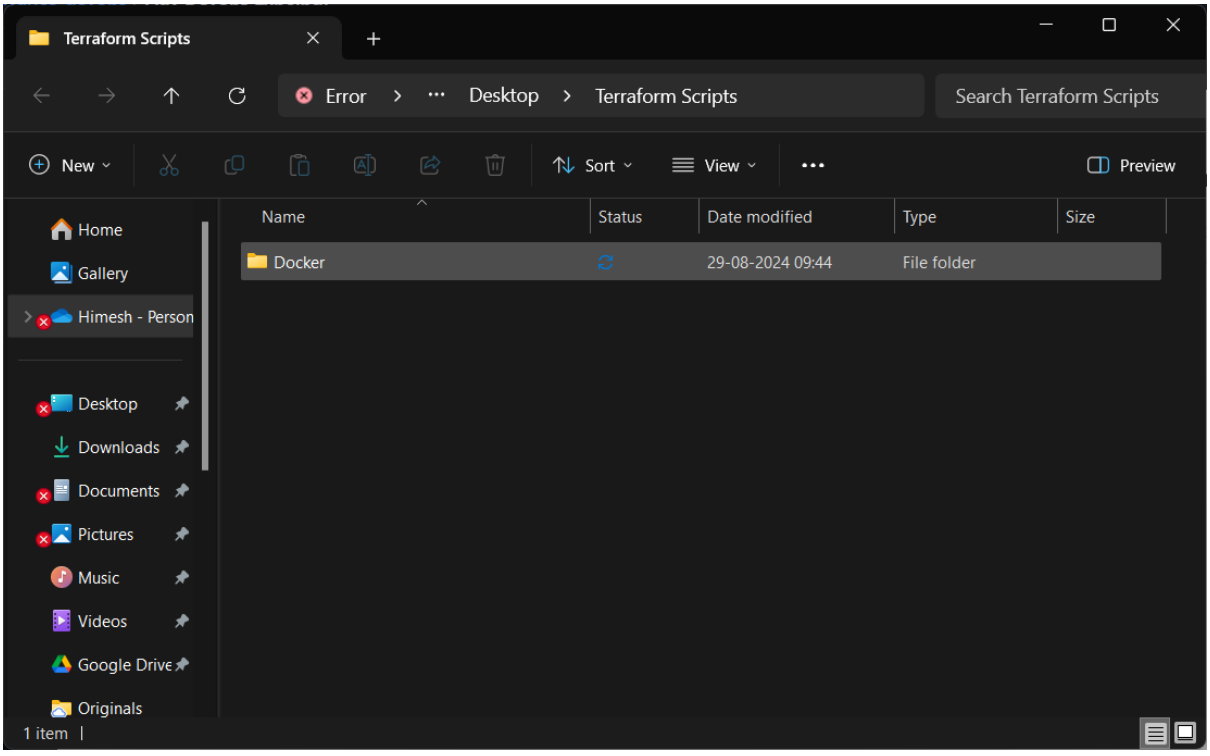

Installing Docker Desktop 4.33.1 (161083)
Configuration
Add shortcut to desktop
OK


Installing Docker Desktop 4.33.1 (161083)
Configuration
Add shortcut to desktop
OK

Step 3: Open Command Prompt and enter the command docker –version, to check whether the docker is successfully installed.



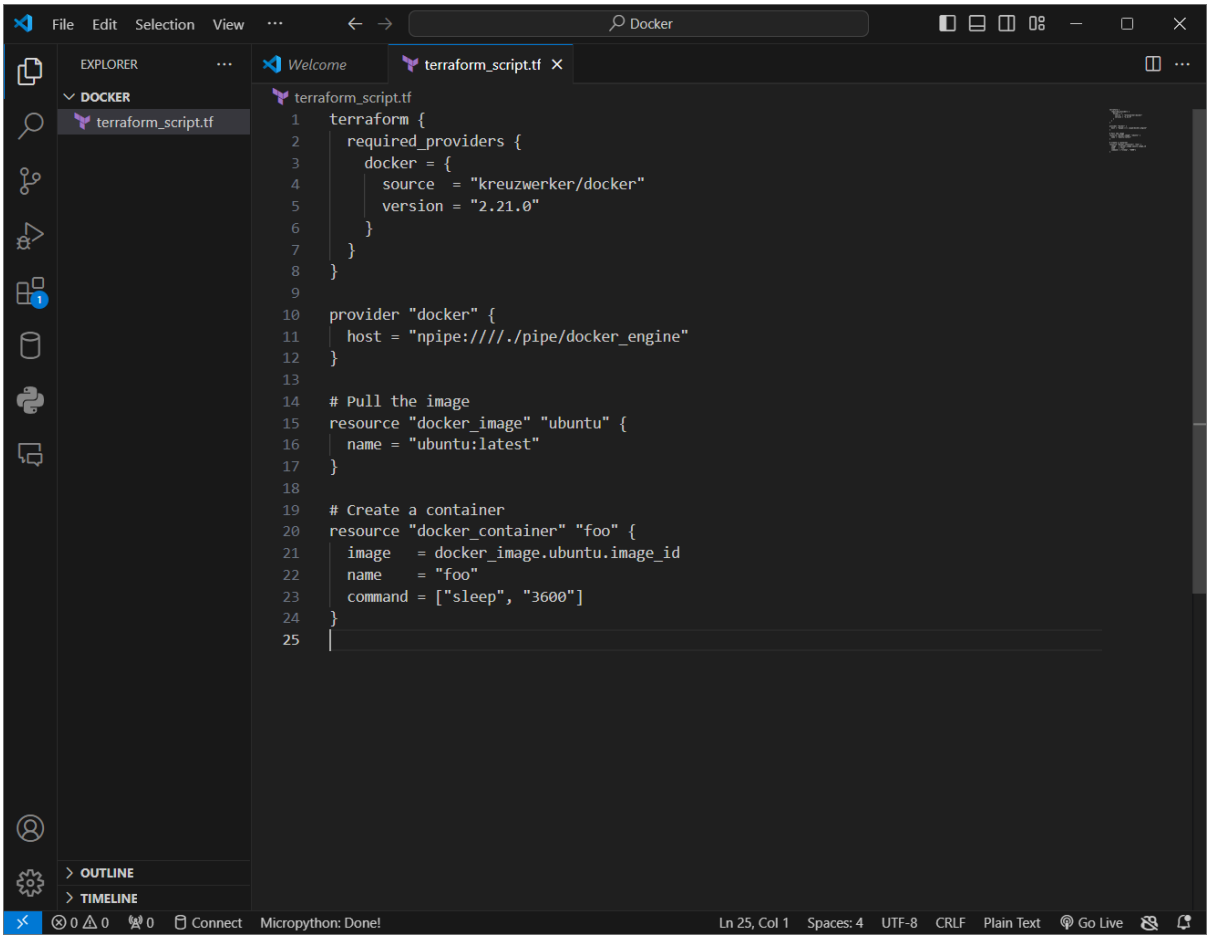
Step 4: Create a folder Terraform_scripts and inside it create a folder named Docker.



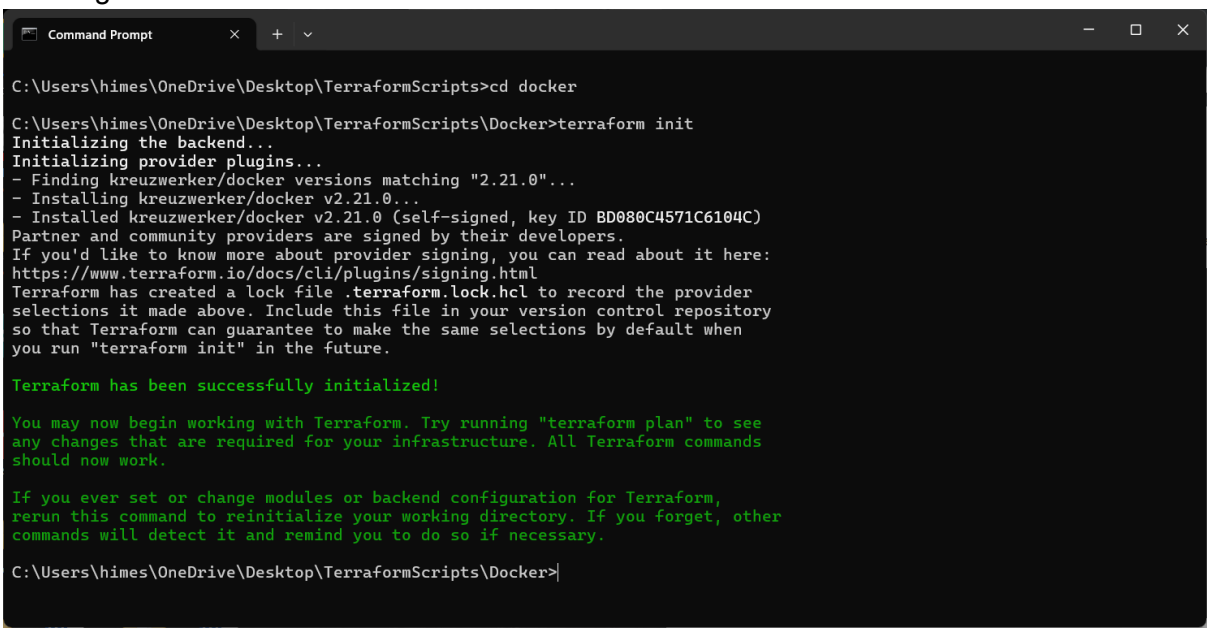
Step 5: create a new folder named ‘Terraform’ in the ‘TerraformScripts’ folder. Then create a new terraform_script.tf file using vs code.

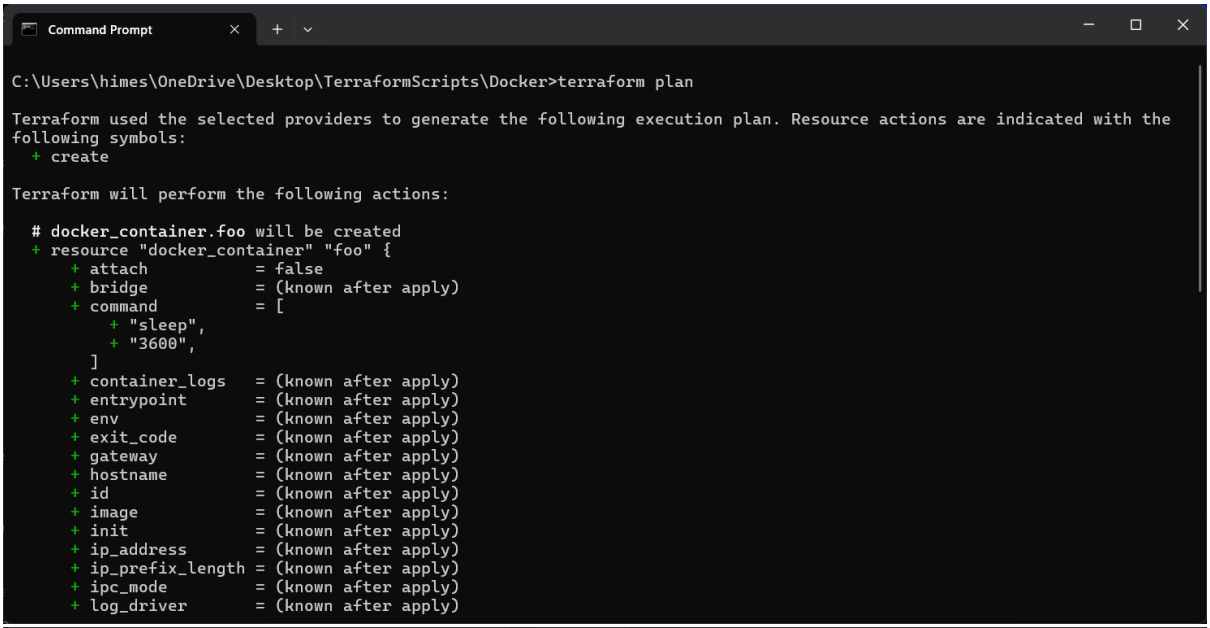
Run the following script in the VS Code.

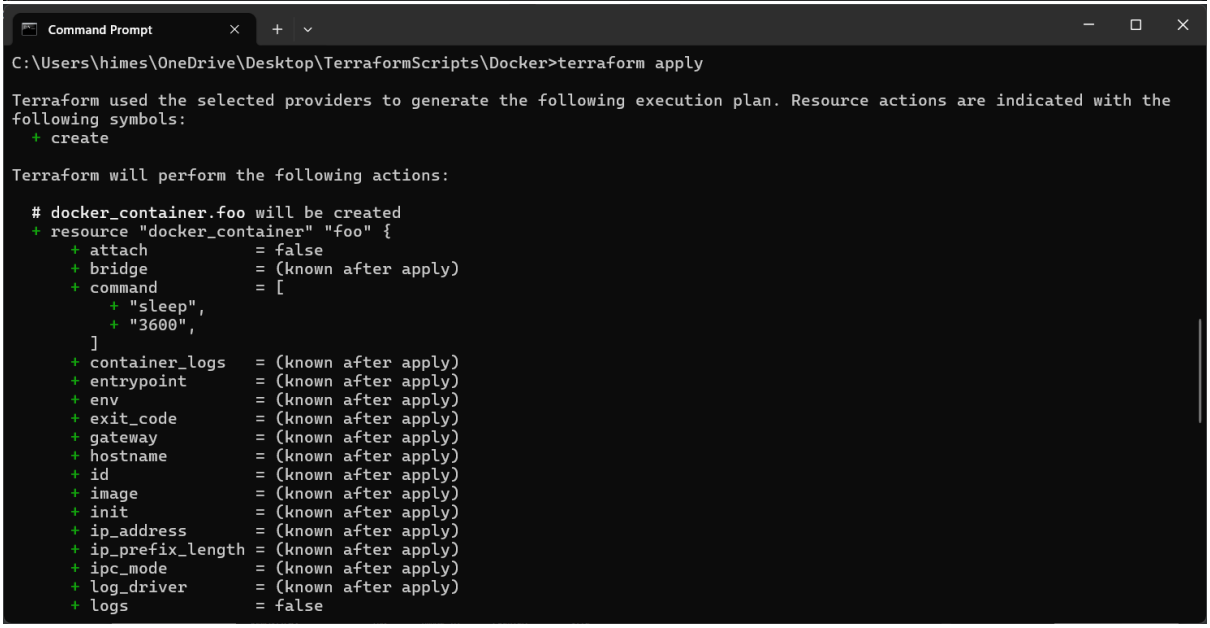
```
terraform {
required_providers {
    docker = {
      source  = "kreuzwerker/docker"
      version = "2.21.0"
    }
  }
}

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

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

# Create a container resource
"docker_container" "foo" {   image =
docker_image.ubuntu.image_id
  name = "foo"
  command = ["sleep", "3600"]
}
```

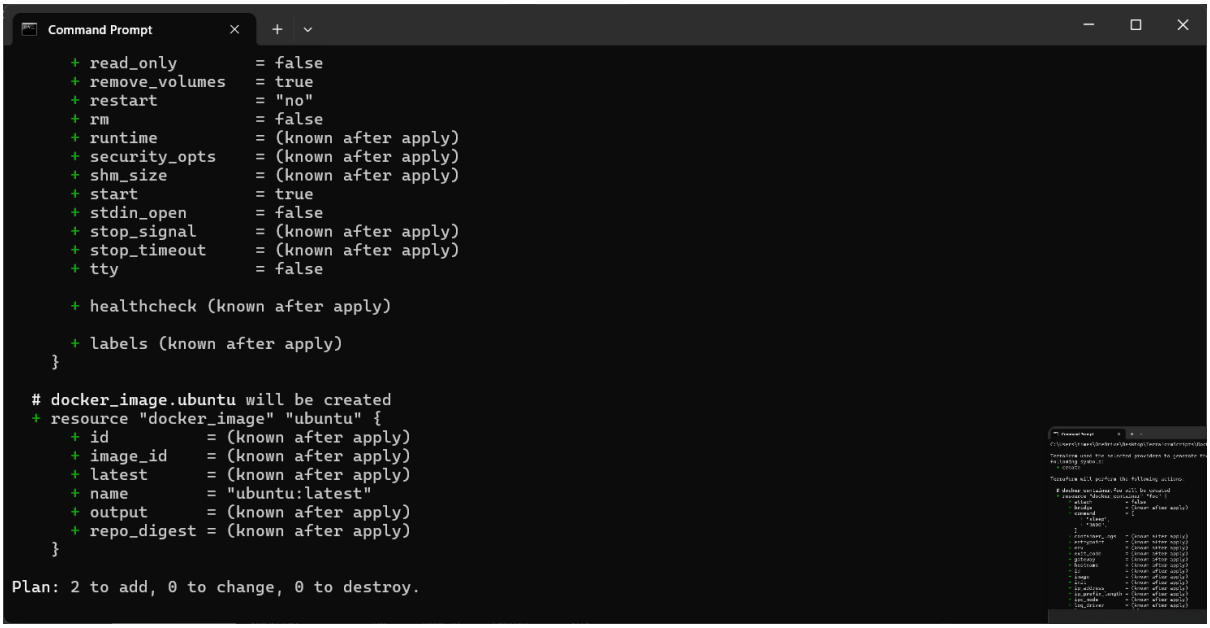


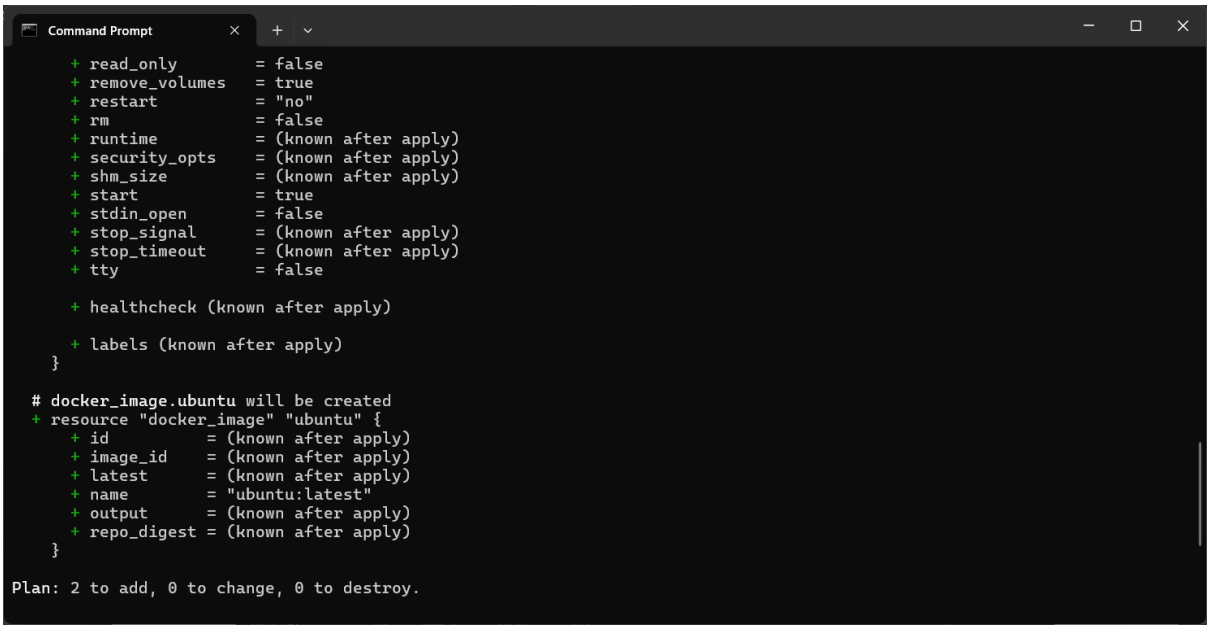
Step 6: Open Windows Explorer and run the following command terraform init, terraform plan, terraform apply, terraform destroy, terraform provider, terraform validate, terraform state list and docker images.

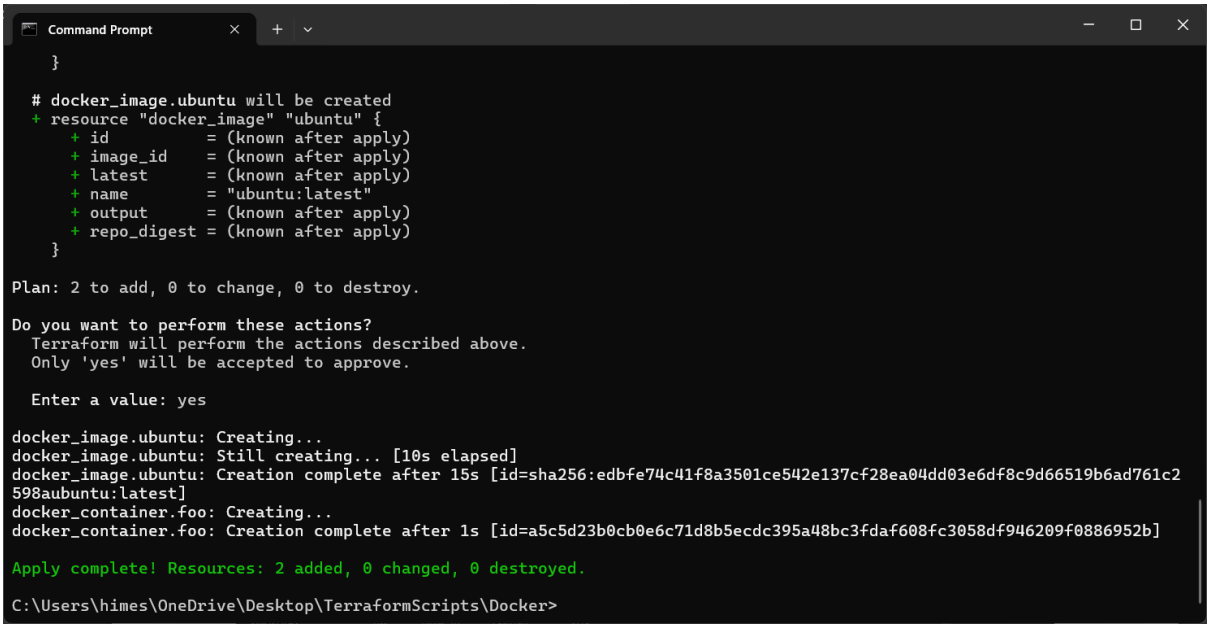


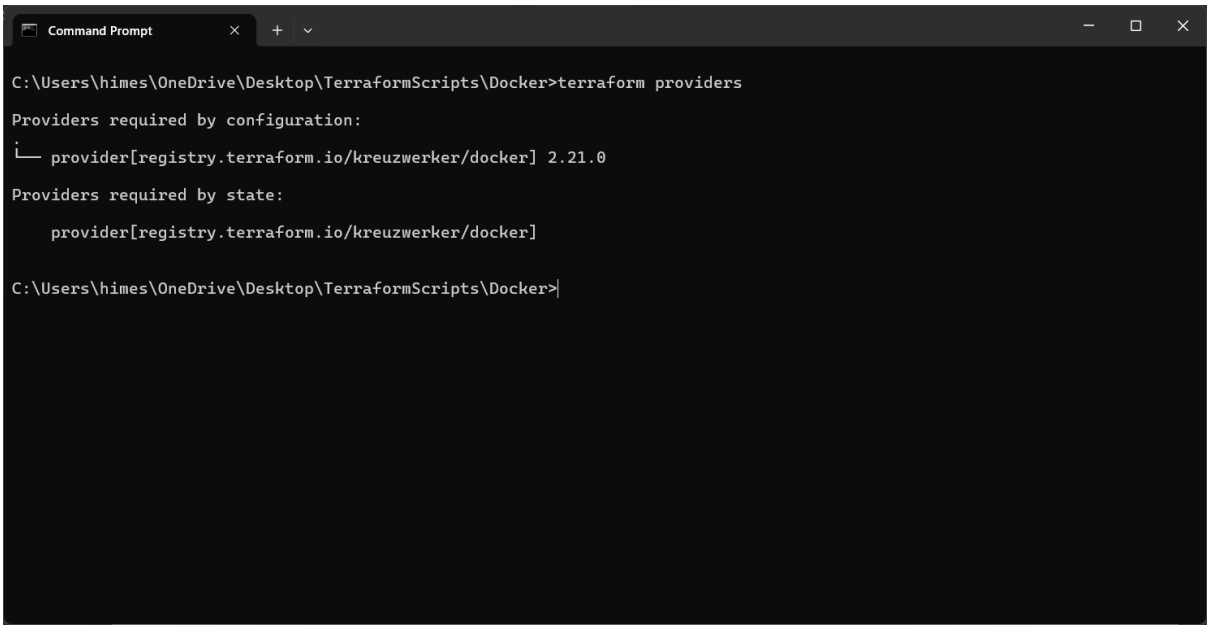

Command Prompt
C:\Users\himes\OneDrive\Desktop\TerraformScripts\Docker>terraform plan
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create
Terraform will perform the following actions:
# docker_container.foo will be created
+ resource "docker_container" "foo" {
+ attach = false
+ bridge = (known after apply)
+ command = [
+ "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)
+ ipc_mode = (known after apply)
+ log_driver = (known after apply)


Command Prompt
C:\Users\himes\OneDrive\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)
+ ipc_mode = (known after apply)
+ log_driver = (known after apply)
+ logs = false


Command Prompt
+ read_only = false
+ remove_volumes = true
+ restart = "no"
+ rm = false
+ runtime = (known after apply)
+ security_opts = (known after apply)
+ shm_size = (known after apply)
+ start = true
+ stdin_open = false
+ stop_signal = (known after apply)
+ stop_timeout = (known after apply)
+ tty = false
+ healthcheck (known after apply)
+ labels (known after apply)
}
# docker_image.ubuntu will be created
+ resource "docker_image" "ubuntu" {
+ id = (known after apply)
+ image_id = (known after apply)
+ latest = (known after apply)
+ name = "ubuntu:latest"
+ output = (known after apply)
+ repo_digest = (known after apply)
}
Plan: 2 to add, 0 to change, 0 to destroy.


Command Prompt
+ read_only = false
+ remove_volumes = true
+ restart = "no"
+ rm = false
+ runtime = (known after apply)
+ security_opts = (known after apply)
+ shm_size = (known after apply)
+ start = true
+ stdin_open = false
+ stop_signal = (known after apply)
+ stop_timeout = (known after apply)
+ tty = false
+ healthcheck (known after apply)
+ labels (known after apply)
}
# docker_image.ubuntu will be created
+ resource "docker_image" "ubuntu" {
+ id = (known after apply)
+ image_id = (known after apply)
+ latest = (known after apply)
+ name = "ubuntu:latest"
+ output = (known after apply)
+ repo_digest = (known after apply)
}
Plan: 2 to add, 0 to change, 0 to destroy.


Command Prompt
}
# docker_image.ubuntu will be created
+ resource "docker_image" "ubuntu" {
+ id = (known after apply)
+ image_id = (known after apply)
+ latest = (known after apply)
+ name = "ubuntu:latest"
+ output = (known after apply)
+ repo_digest = (known after apply)
}
Plan: 2 to add, 0 to change, 0 to destroy.
Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.
Enter a value: yes
docker_image.ubuntu: Creating...
docker_image.ubuntu: Still creating... [10s elapsed]
docker_image.ubuntu: Creation complete after 15s [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Creating...
docker_container.foo: Creation complete after 1s [id=a5c5d23b0cb0e6c71d8b5ecdc395a48bc3fdaf608fc3058df946209f0886952b]
Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
C:\Users\himes\OneDrive\Desktop\TerraformScripts\Docker>


Command Prompt
C:\Users\himes\OneDrive\Desktop\TerraformScripts\Docker>terraform providers
Providers required by configuration:
.
└── provider[registry.terraform.io/kreuzwerker/docker] 2.21.0
Providers required by state:
provider[registry.terraform.io/kreuzwerker/docker]
C:\Users\himes\OneDrive\Desktop\TerraformScripts\Docker>

```
C:\Users\himes\OneDrive\Desktop\TerraformScripts\Docker>terraform validate
Success! The configuration is valid.


C:\Users\himes\OneDrive\Desktop\TerraformScripts\Docker>
```

```
C:\Users\himes\OneDrive\Desktop\TerraformScripts\Docker>terraform state list
docker_container.foo
docker_image.ubuntu

C:\Users\himes\OneDrive\Desktop\TerraformScripts\Docker>SS
```

```
C:\Users\himes\OneDrive\Desktop\TerraformScripts>docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
ubuntu       latest    edbfe74c41f8   3 weeks ago   78.1MB

C:\Users\himes\OneDrive\Desktop\TerraformScripts>S
```

ADVANCE DEVOPS EXP 7

Name : **Himesh Pathai**
Class : **D15A**
Roll No. : **35**

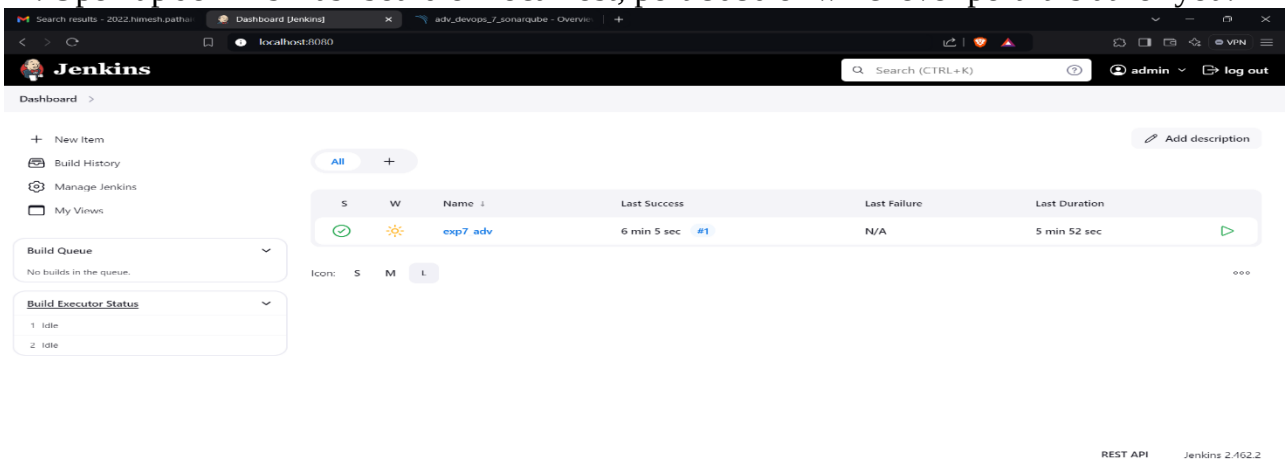
Aim: To understand Static Analysis SAST process and learn to integrate Jenkins SAST to SonarQube/GitLab.

Integrating Jenkins with SonarQube:

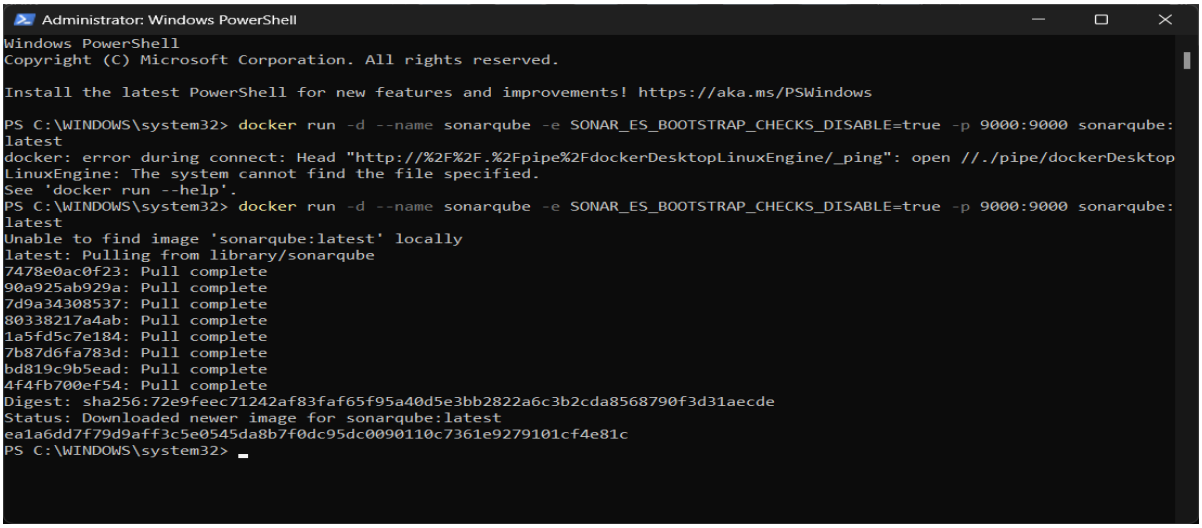
- Jenkins installed
- Docker Installed (for SonarQube)
- SonarQube Docker Image

Steps to integrate Jenkins with SonarQube

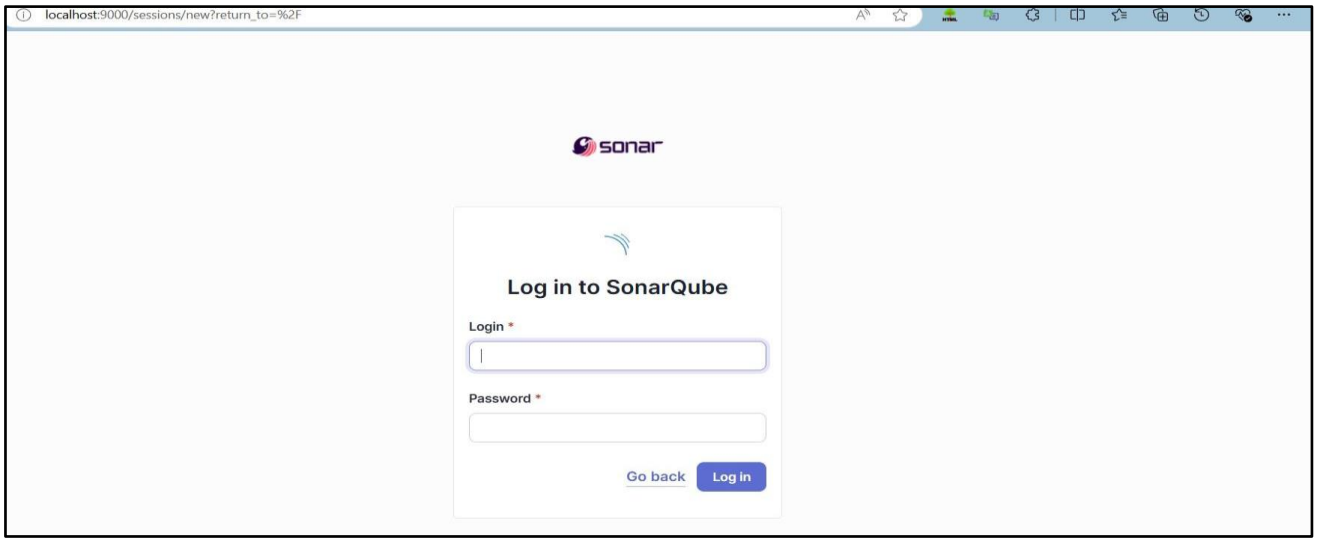
1. Open up Jenkins Dashboard on localhost, port 8080 or whichever port it is at for you.



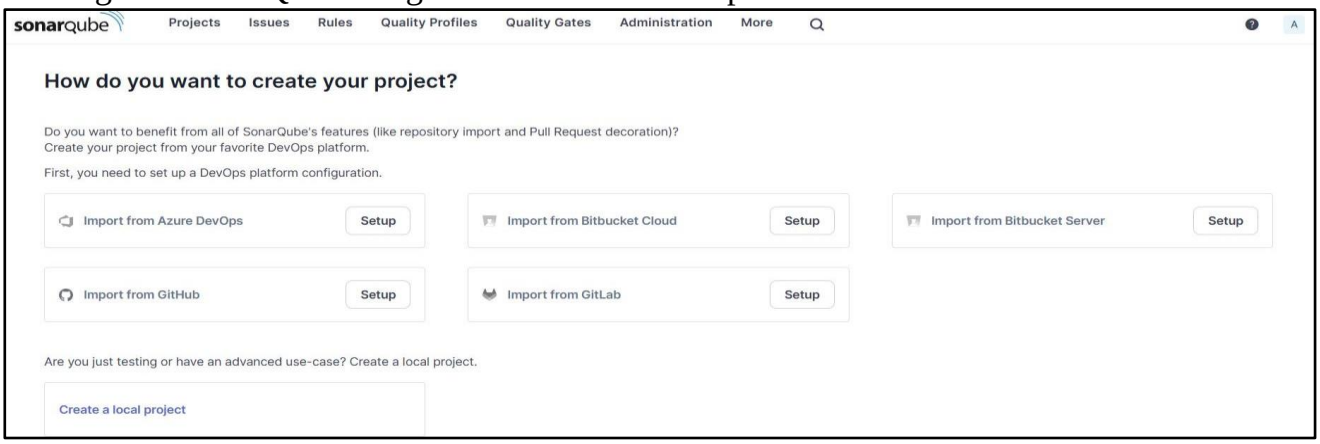
2. Run SonarQube in a Docker container using this command -
docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest



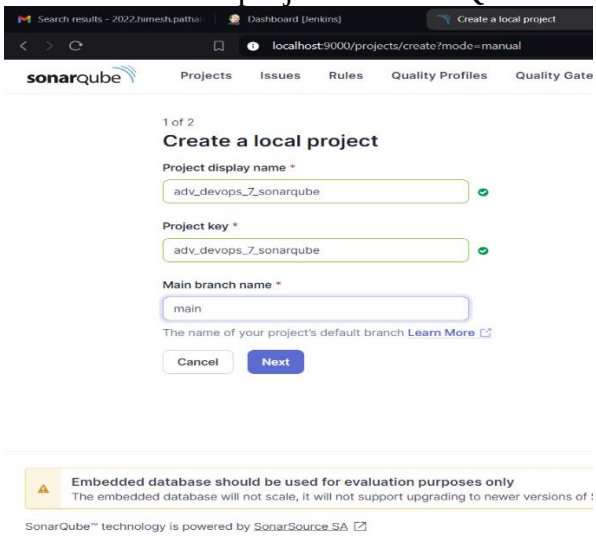
3. Once the container is up and running, you can check the status of SonarQube at localhost port 9000.



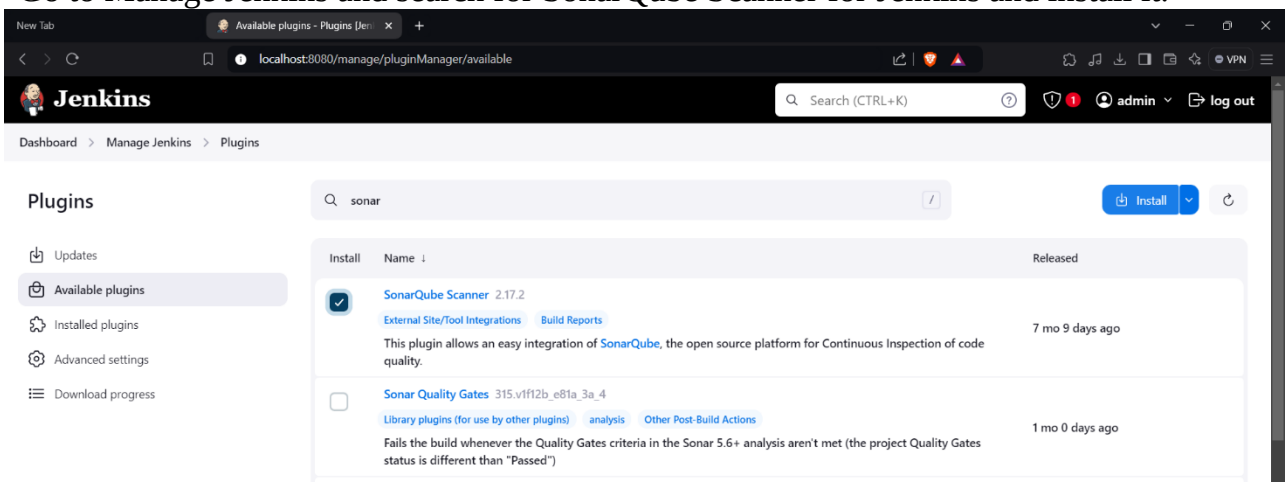
4. Login to SonarQube using username admin and password admin.

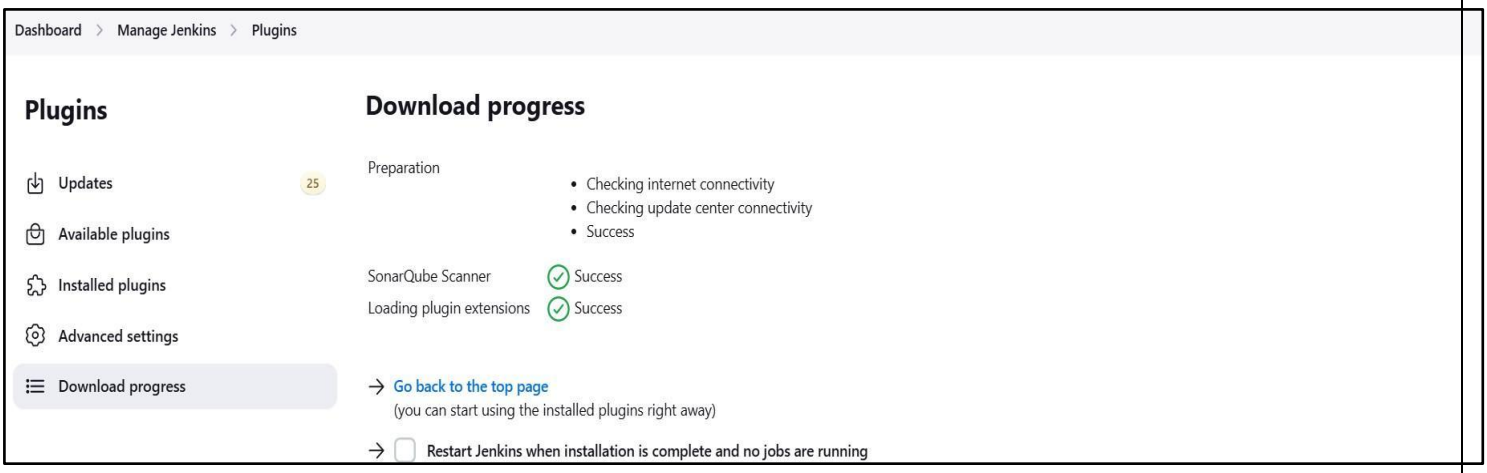


5. Create a manual project in SonarQube with the name sonarqube



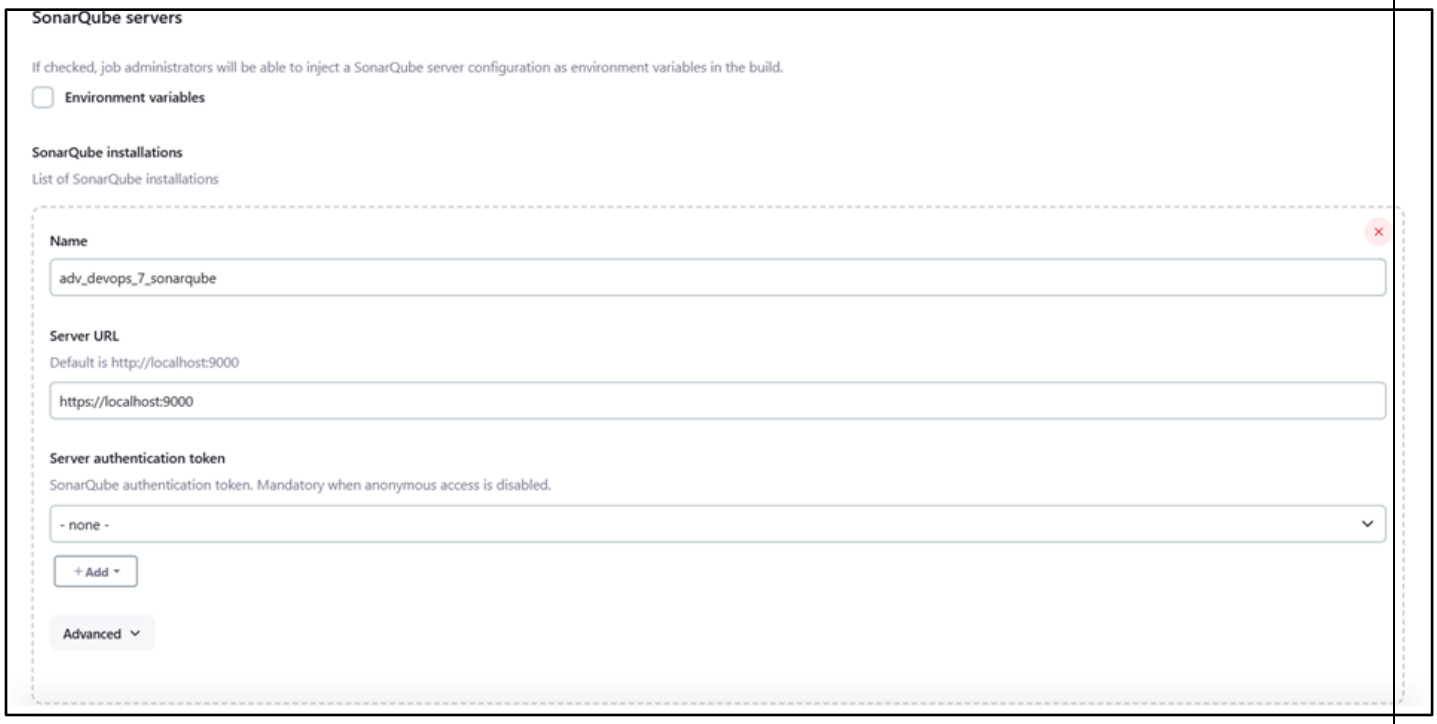
Setup the project and come back to Jenkins Dashboard.
Go to Manage Jenkins and search for SonarQube Scanner for Jenkins and install it.





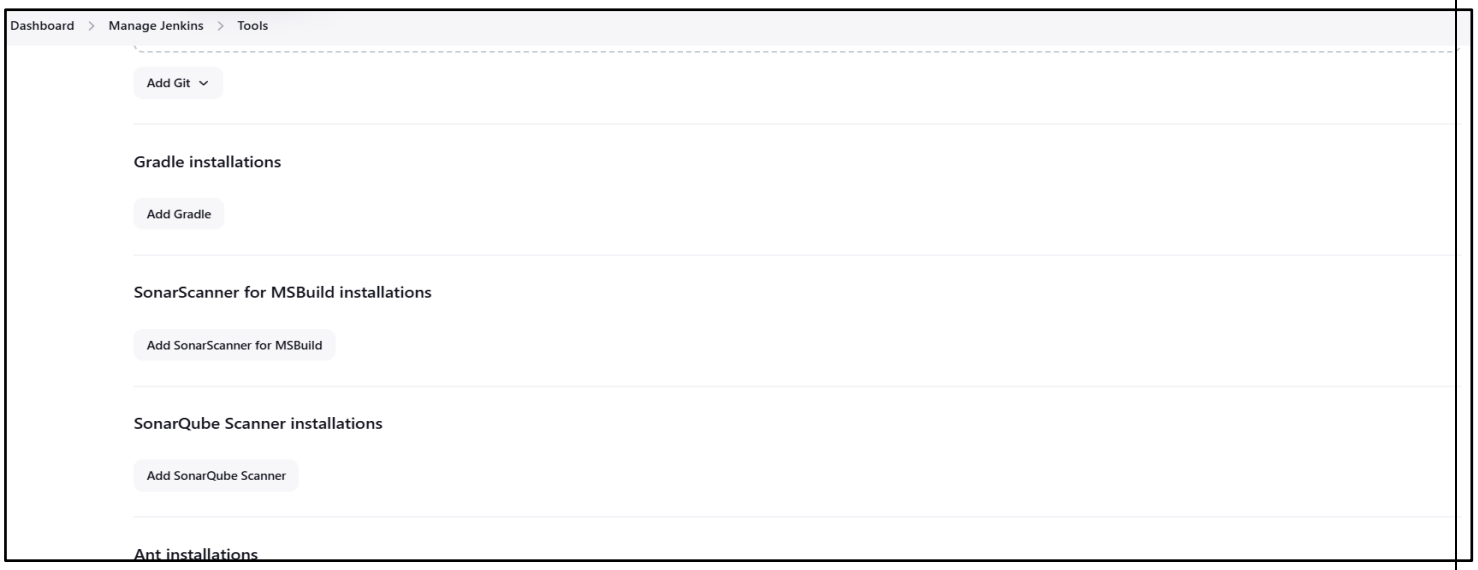
6. Under Jenkins ‘Manage Jenkins’ then go to ‘system’, scroll and look for **SonarQube Servers**
and enter the details.
Enter the Server Authentication token if needed.

In SonarQube installations: Under **Name** add <project name of sonarqube>,here we have named it as **adv_devops_7_sonarqube**
In **Server URL** Default is **http://localhost:9000**


SonarQube servers
If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.
Environment variables
SonarQube installations
List of SonarQube installations
Name
adv_devops_7_sonarqube
Server URL
Default is http://localhost:9000
https://localhost:9000
Server authentication token
SonarQube authentication token. Mandatory when anonymous access is disabled.
- none -
+ Add
Advanced

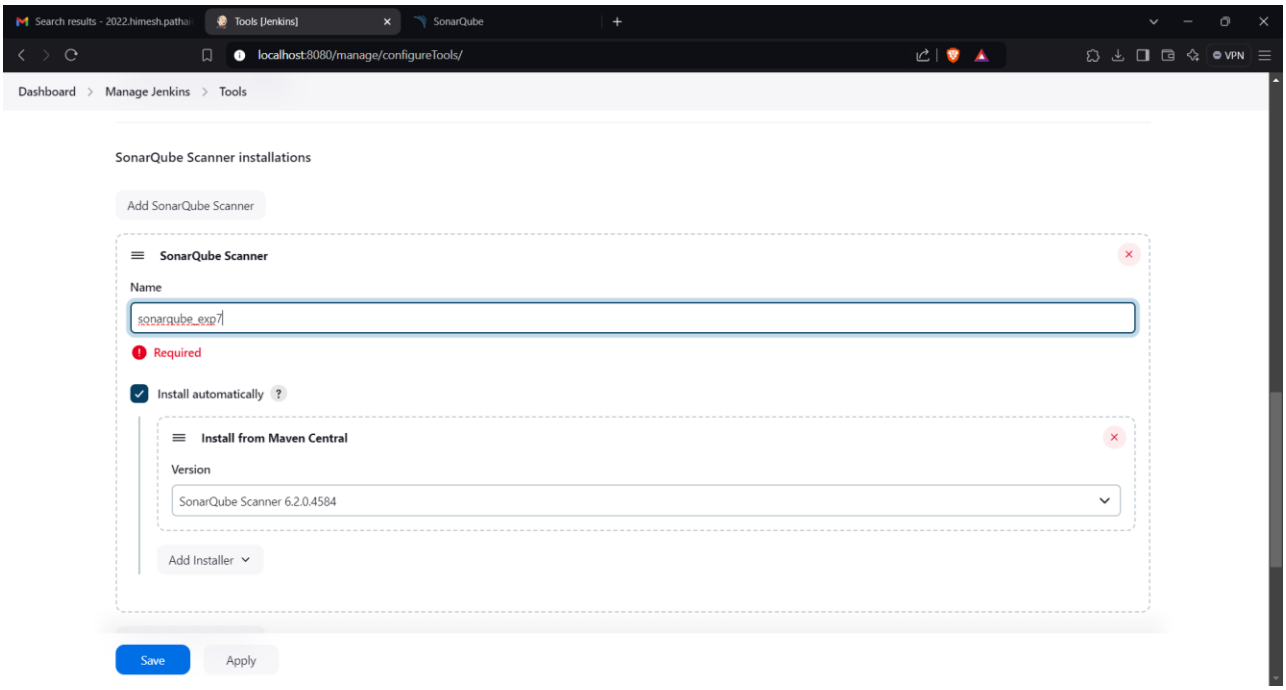
7. Search for SonarQube Scanner under Global Tool Configuration.

Choose the latest configuration and choose Install automatically.

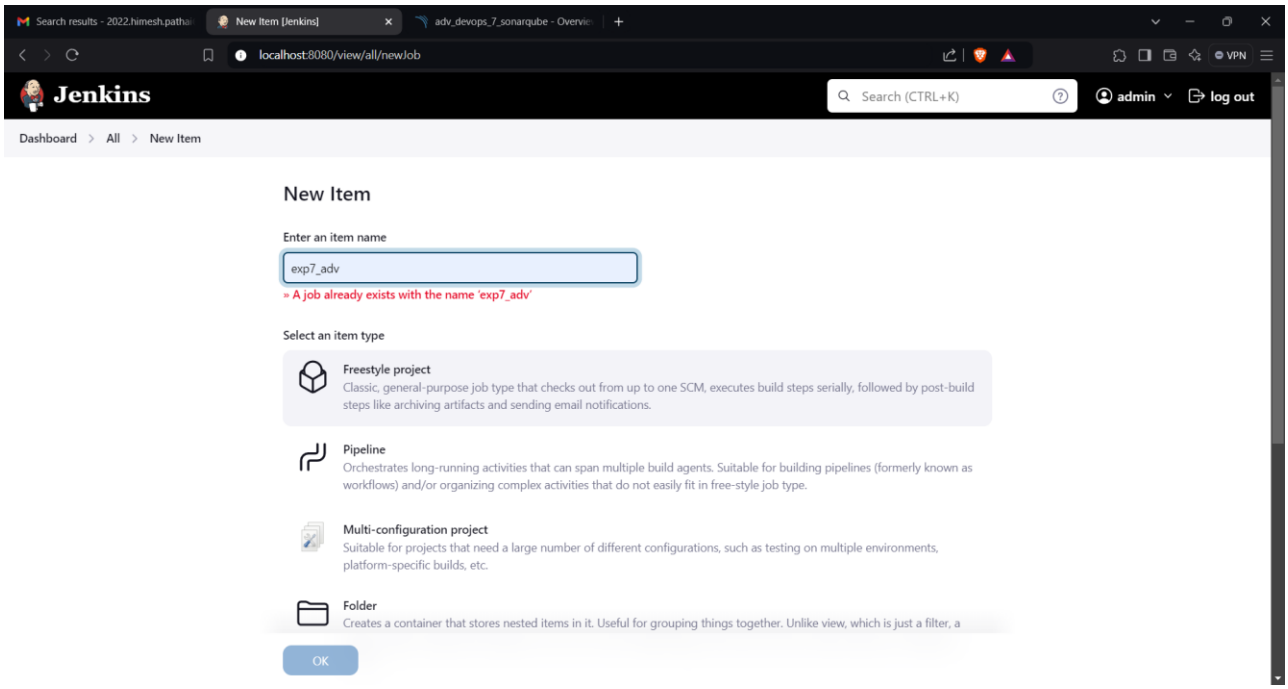


Dashboard > Manage Jenkins > Tools

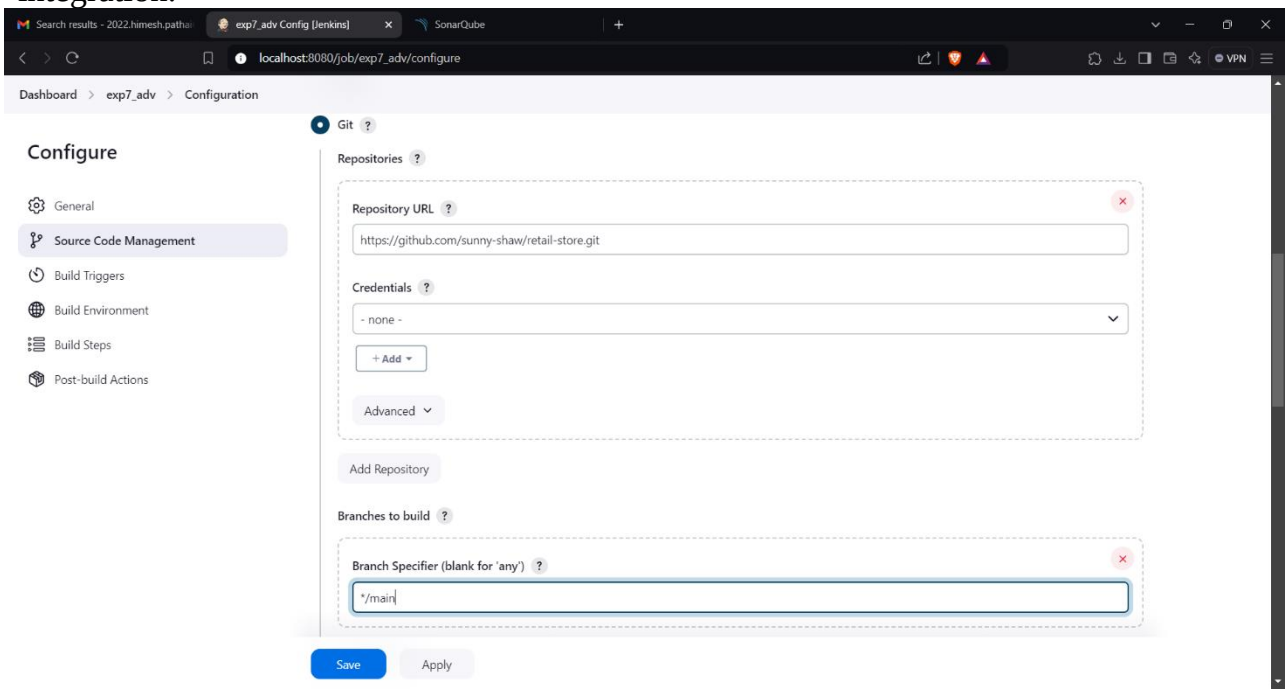
Check the “Install automatically” option. → Under name any name as identifier → Check the “Install automatically” option.



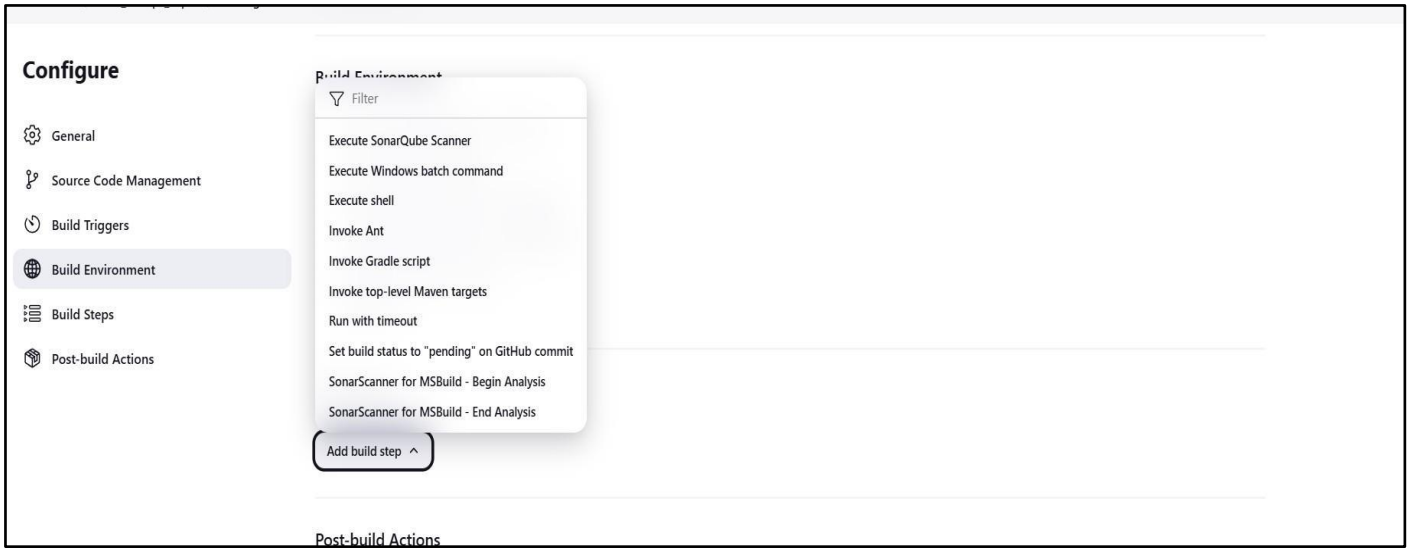
8. After the configuration, create a New Item in Jenkins, choose a freestyle project.



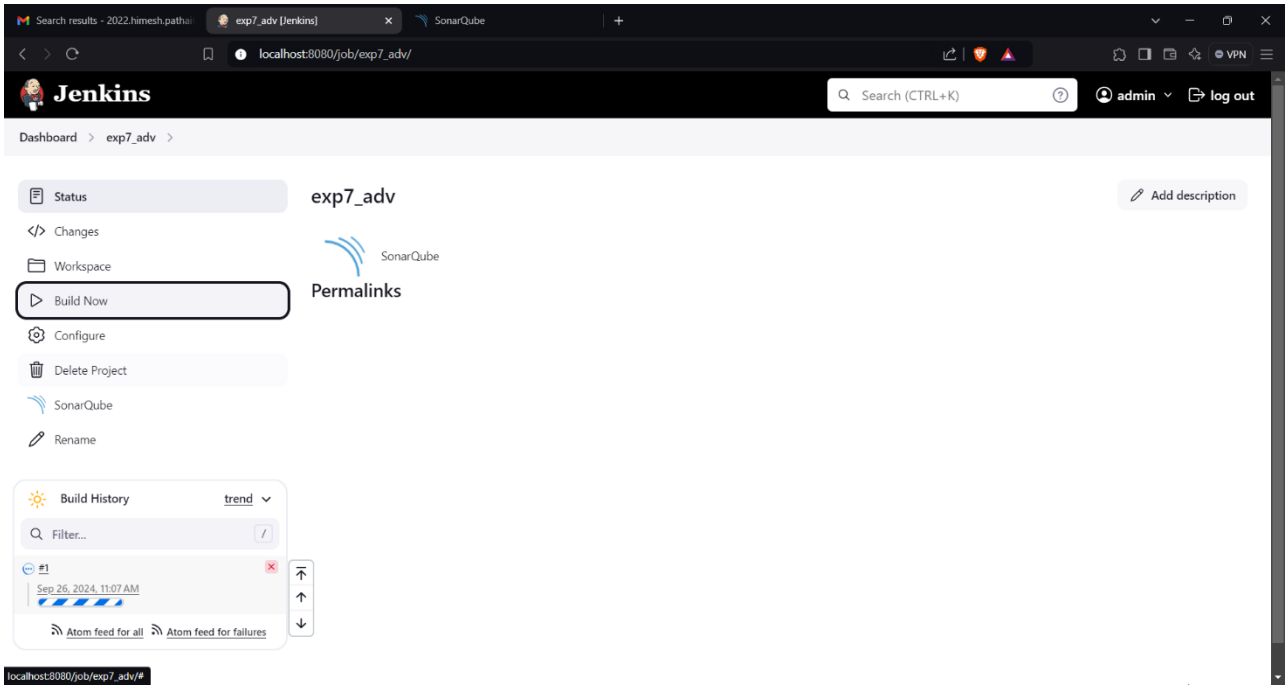
9. Choose this GitHub repository in Source Code Management.
https://github.com/shazforiot/MSBuild_firstproject.git
It is a sample hello-world project with no vulnerabilities and issues, just to test the integration.



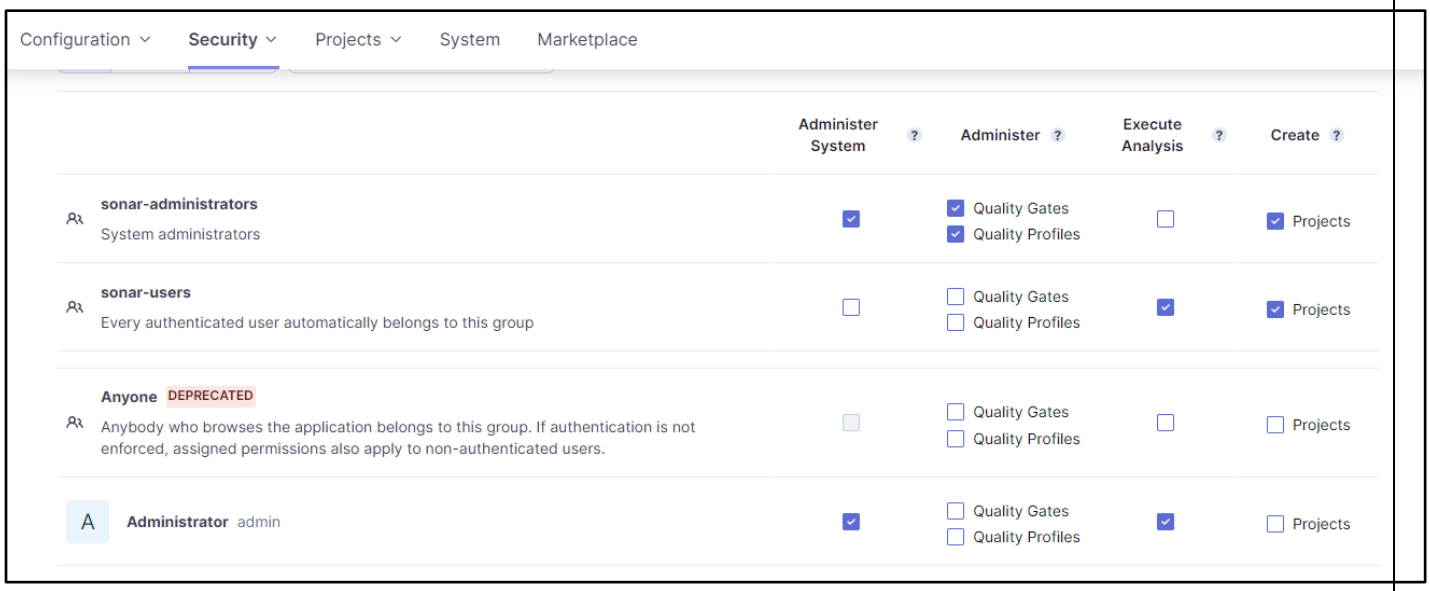
10. Under **Select project → Configuration → Build steps → Execute SonarQube Scanner**, enter these Analysis properties. Mention the SonarQube Project Key, Login, Password, Source path and Host URL.



Then save



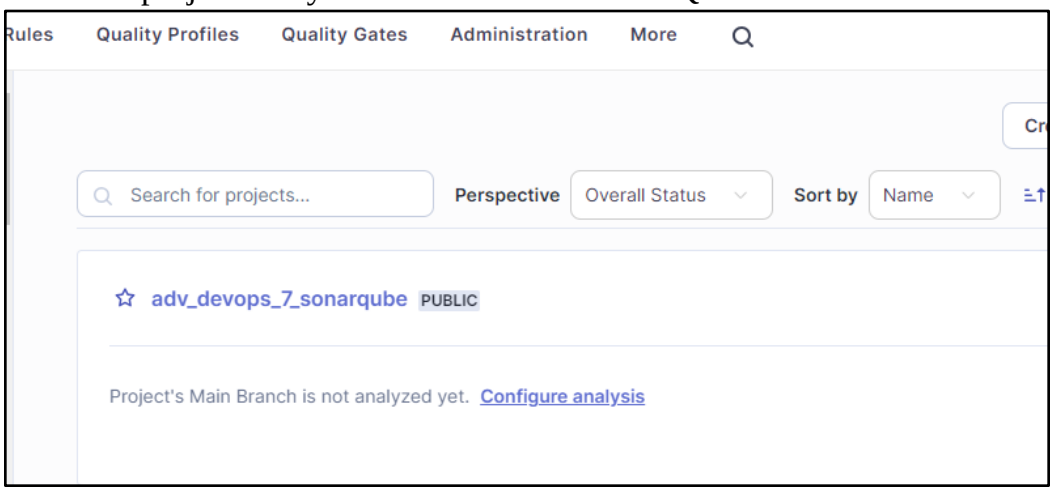
11. Go to http://localhost:9000/<user_name>/permissions and allow Execute Permissions to the Admin user


Configuration
Security
Projects
System
Marketplace
Administer System
Administer
Execute Analysis
Create
sonar-administrators
System administrators
Quality Gates
Quality Profiles
Projects
sonar-users
Every authenticated user automatically belongs to this group
Quality Gates
Quality Profiles
Projects
Anyone
DEPRECATED
Anybody who browses the application belongs to this group. If authentication is not enforced, assigned permissions also apply to non-authenticated users.
Quality Gates
Quality Profiles
Projects
A
Administrator
admin
Quality Gates
Quality Profiles
Projects

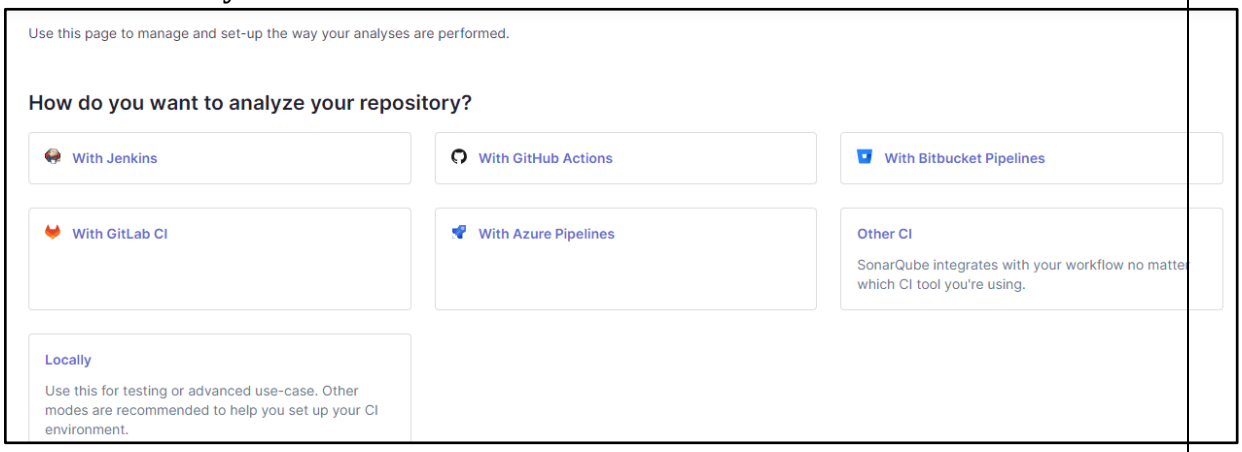
IF CONSOLE OUTPUT FAILED:

Step 1: Generate a New Authentication Token in SonarQube

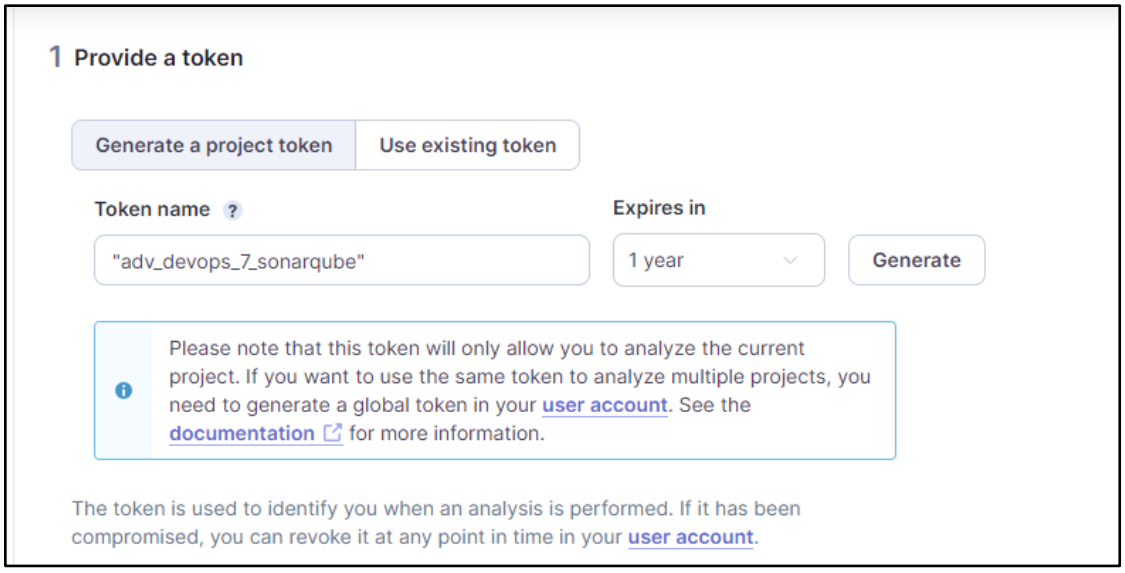
1. **Login to SonarQube:**
   - Open your browser and go to **http://localhost:9000**.
   - Log in with your admin credentials (default username is **admin**, and the password is either **admin** or your custom password if it was changed).
2. **Generate a New Token:**
   - Go to the project that you have created on SonarQube.



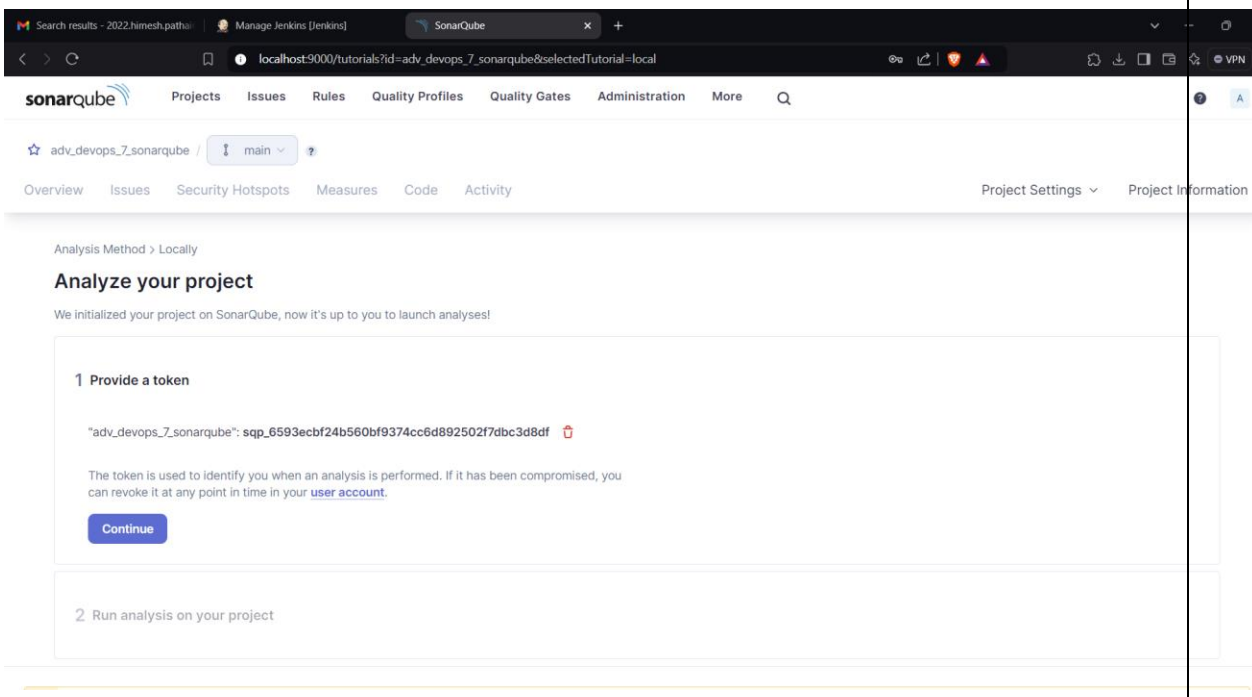
   - Click on **Locally**



   - Further,Generate a Project token with the following details and click on generate.



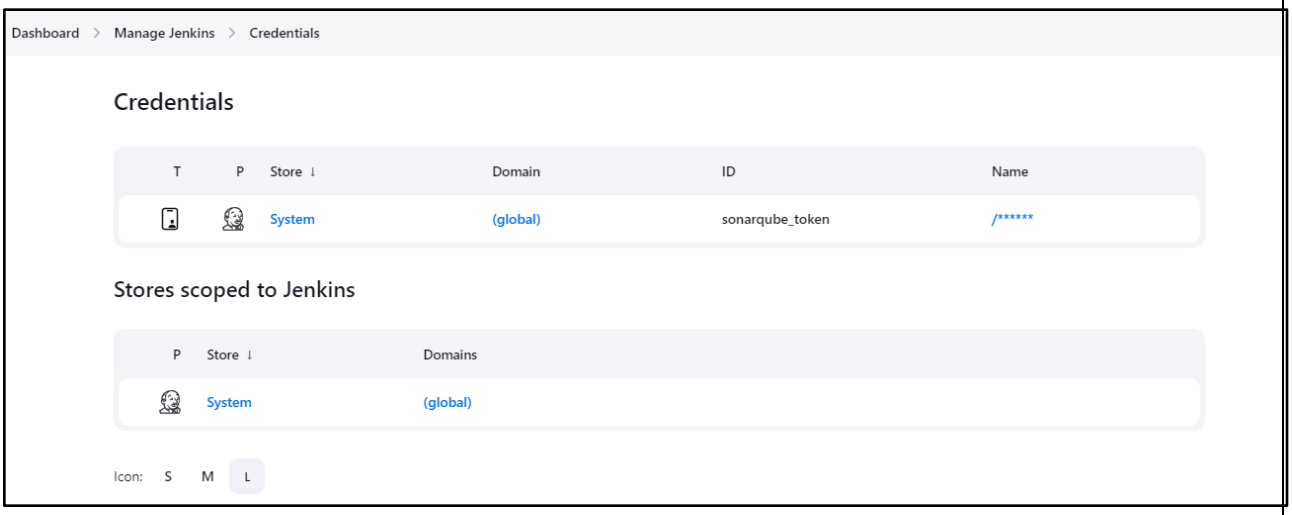
- Copy the token you get here and save it securely as we would need it in Jenkins.



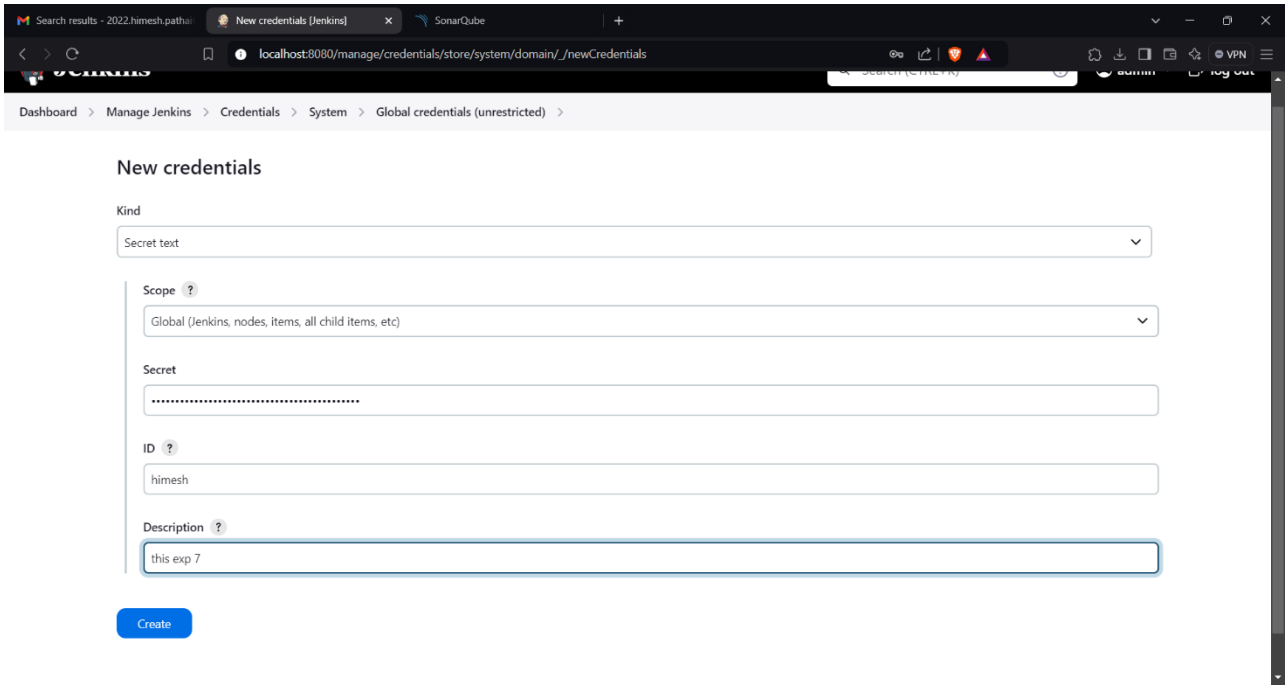
Step 2: Update the Token in Jenkins

1. **Go to Jenkins Dashboard:**
   - Open Jenkins and log in with your credentials.

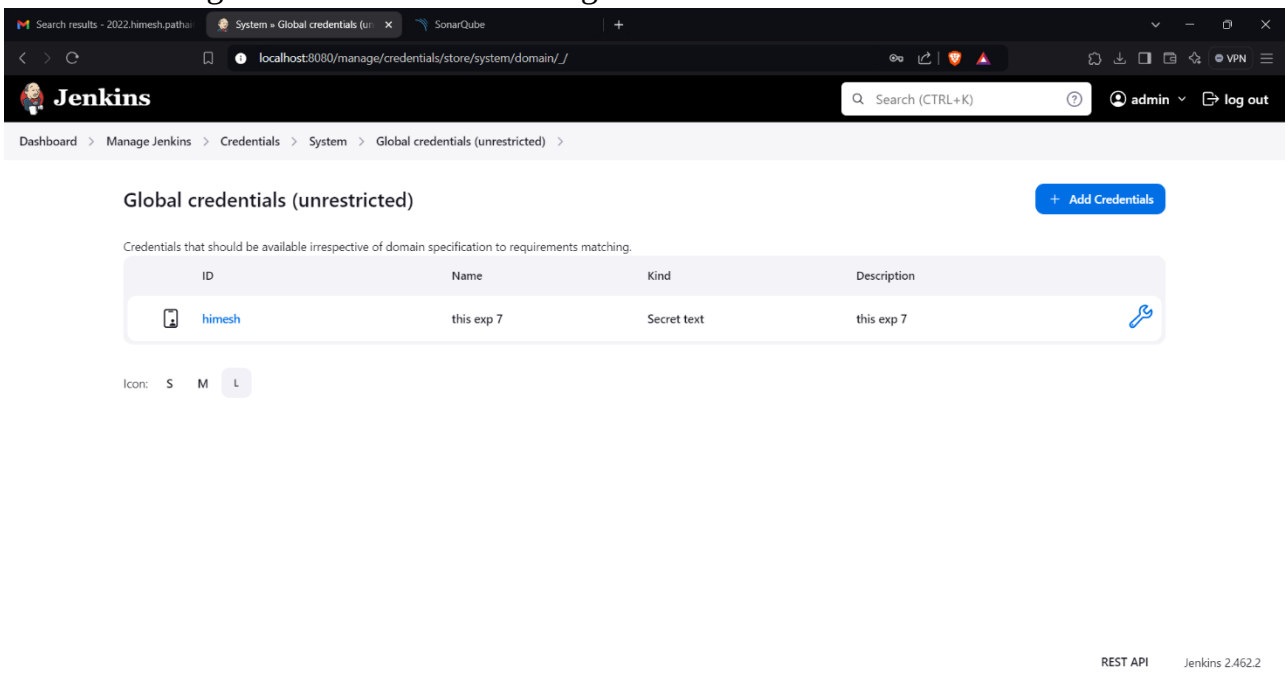
2. Go to Dashboard—>Manage Jenkins—>Credentials



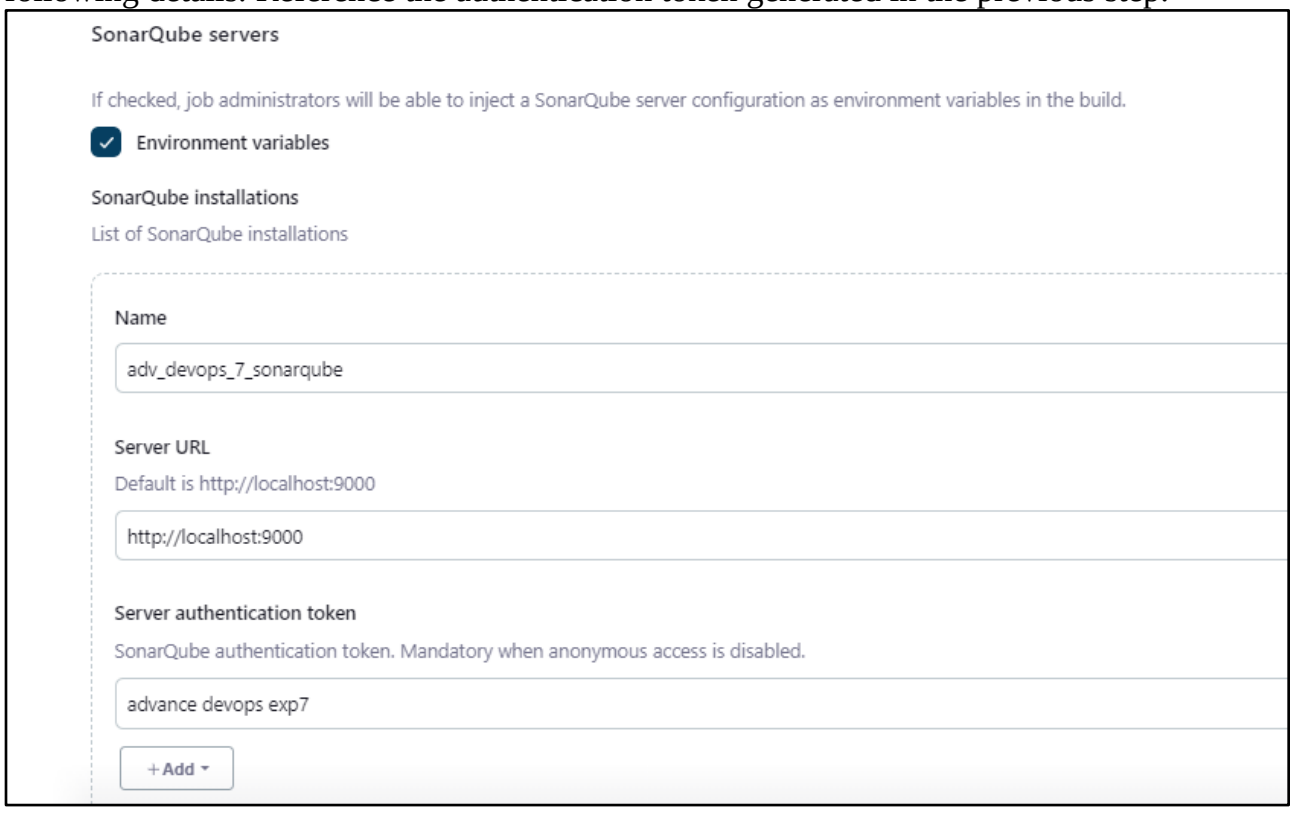
3. Click on **global** under the domains part of Stores scoped to Jenkins section.Further click on add credentials.Proceed with the following details.Make sure to copy the token generated earlier in sonarqube and give any suitable name as the ID.



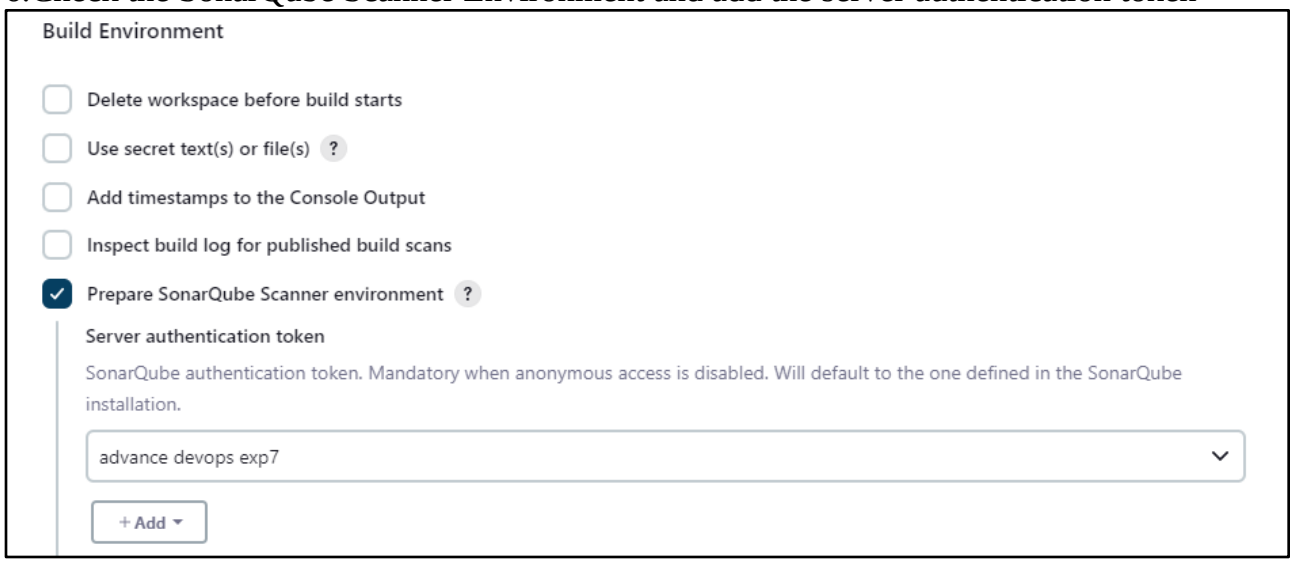
4.After clicking on create we see that the given token has been added in Jenkins credentials.



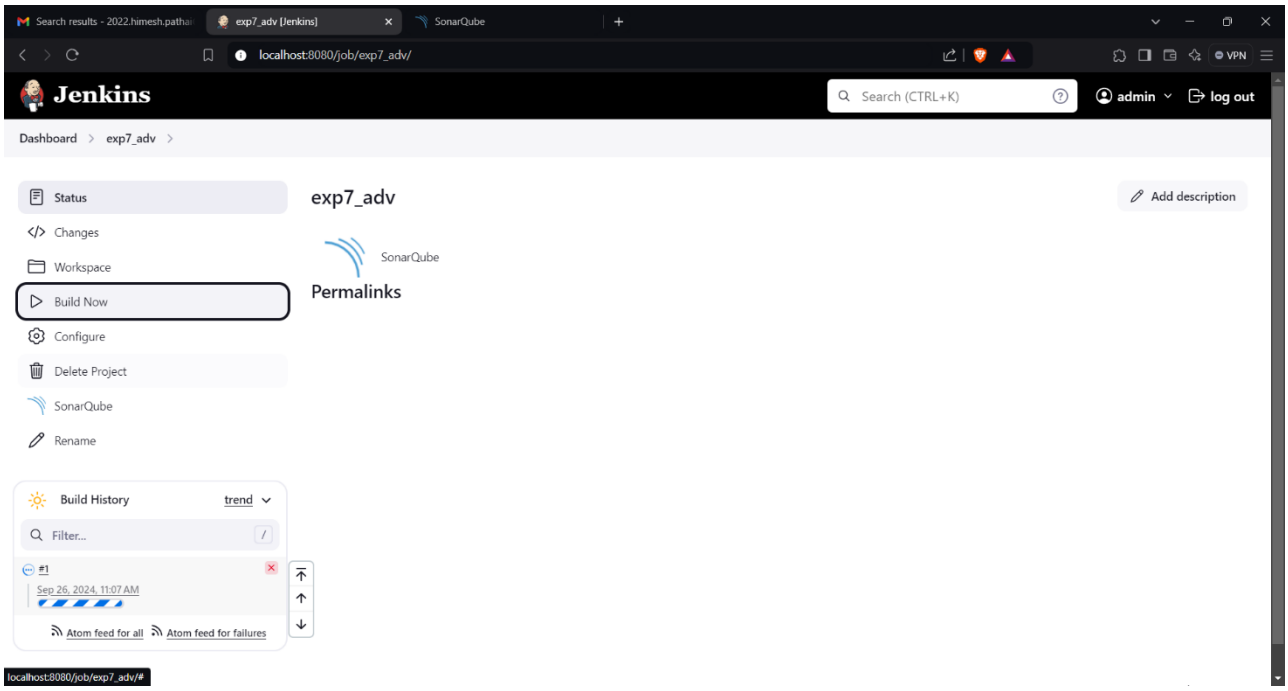
5.Now go to **Manage Jenkins—>System—>SonarQube servers** and proceed with the following details. Reference the authentication token generated in the previous step.



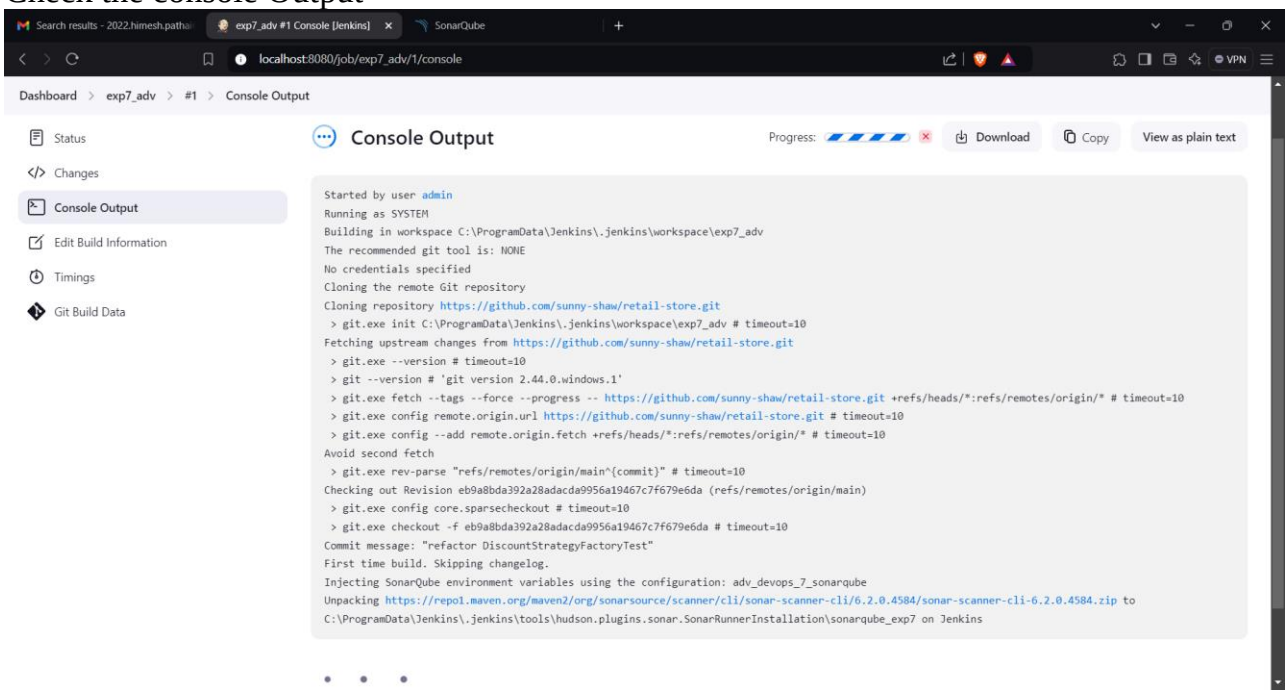
6.Check the SonarQube Scanner Environment and add the server authentication token



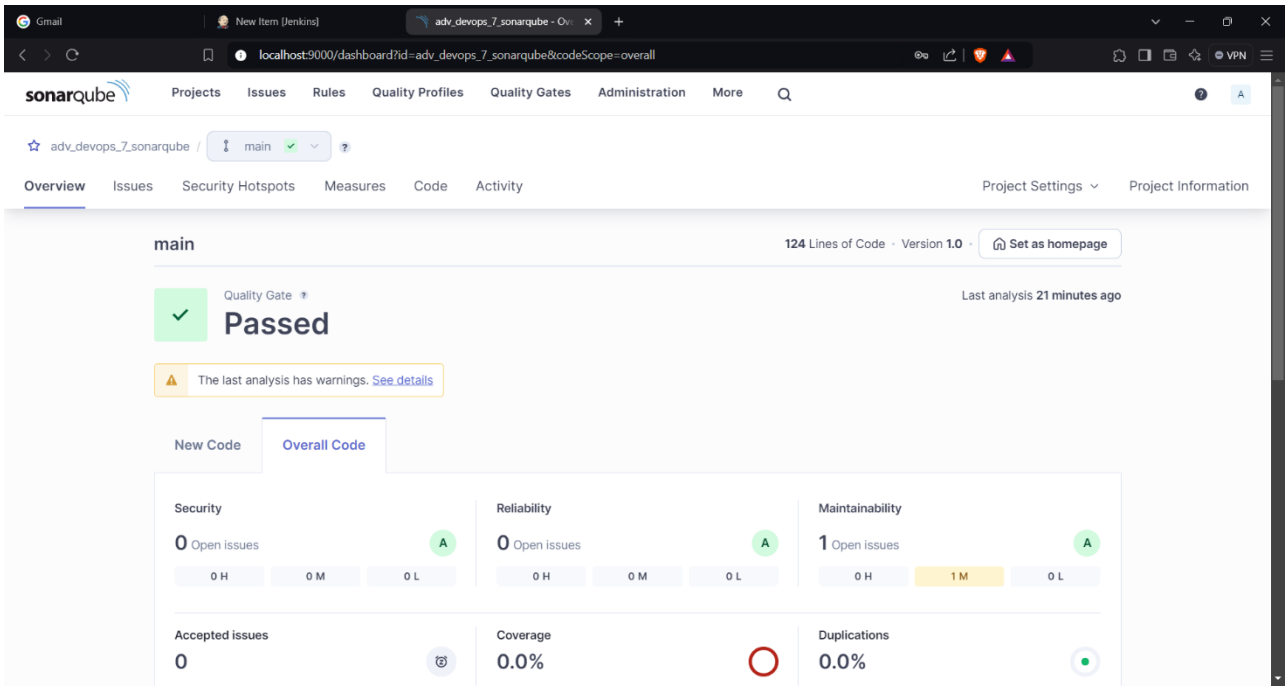
12.Run the Jenkins build.



Check the console Output



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



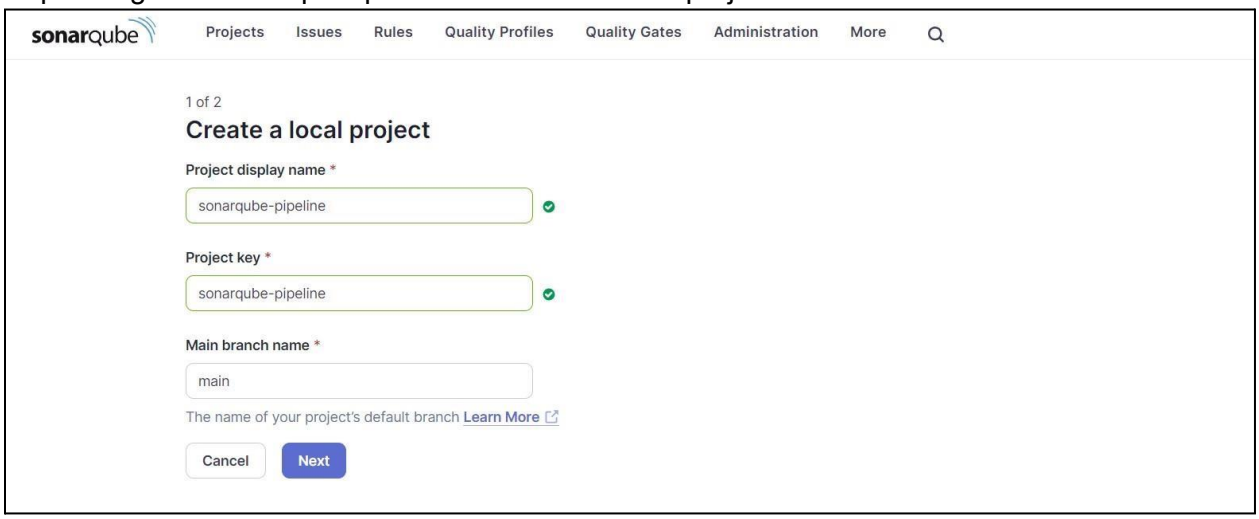
In this way, we have integrated Jenkins with SonarQube for SAST.

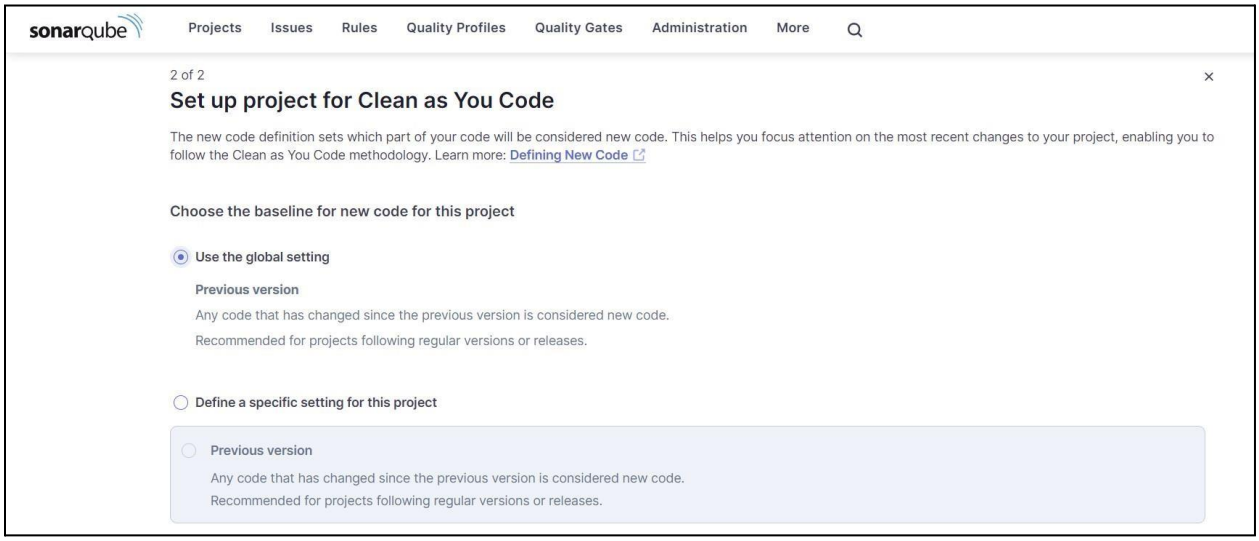
Name : **Himesh Pathai**
Class : **D15A**
Roll No. : **35**

Experiment 8

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

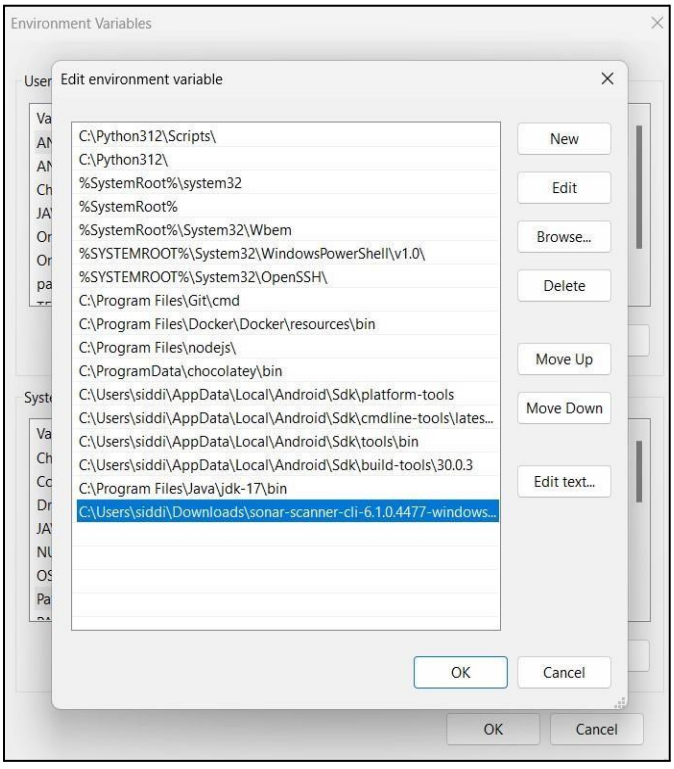
Step 1: Log in to sonarqube portal and create a local project.



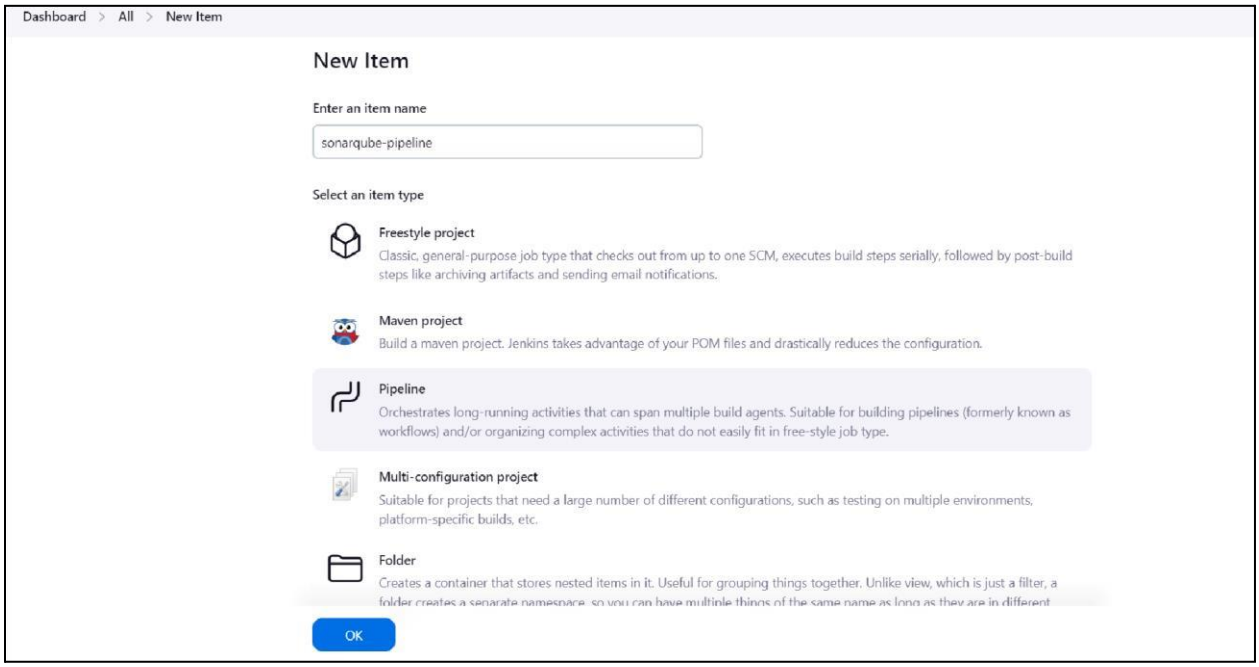


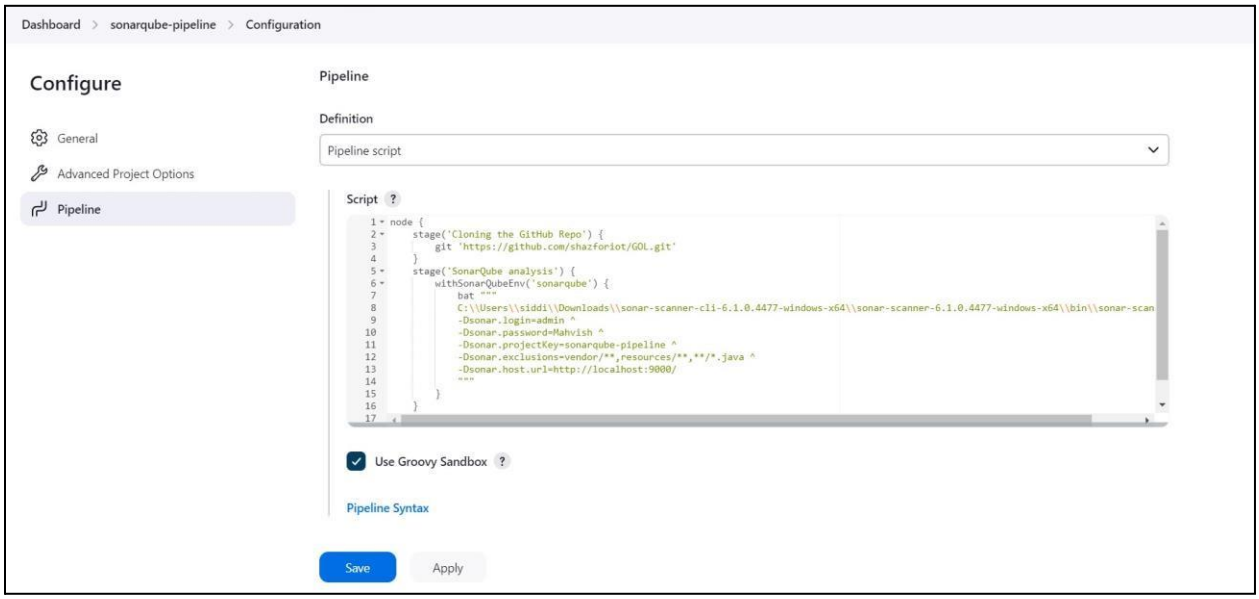
Step 2: Go to [download sonarscanner](#) to download sonar scanner

After the download is complete, extract the file and copy the path to bin folder Go to environment variables, system variables and click on path Add a new path, paste the path copied earlier.

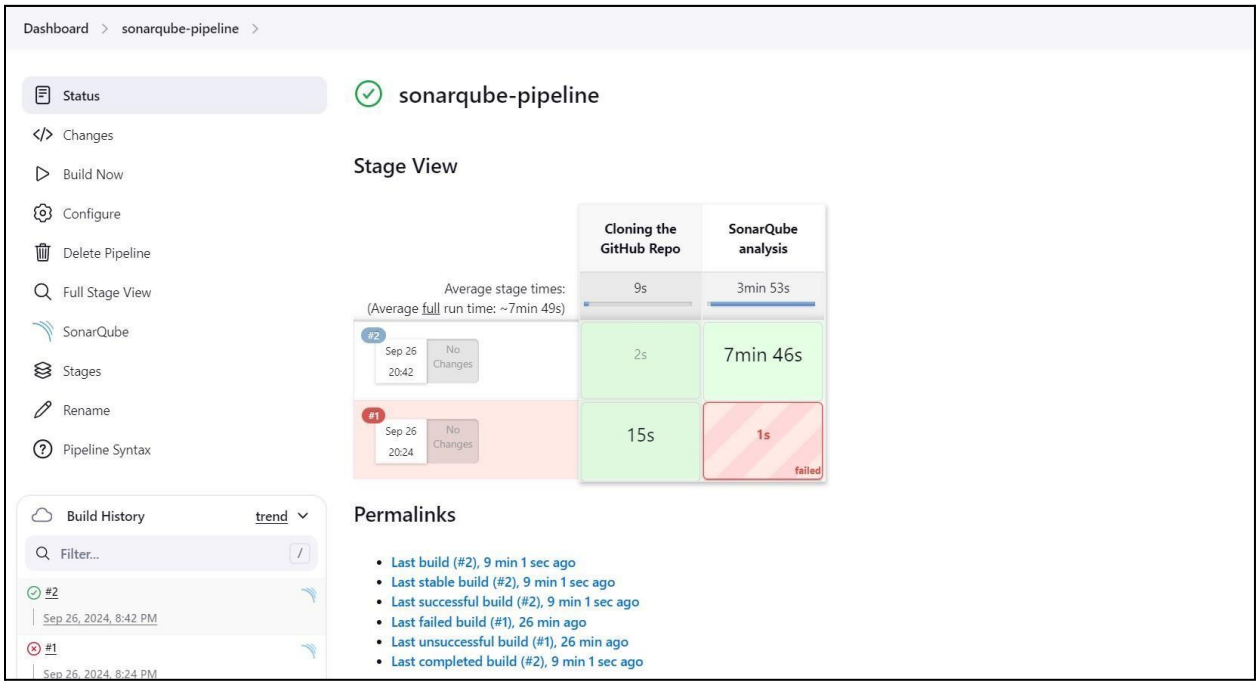


Step 3: Create a New Item in Jenkins, choose Pipeline.

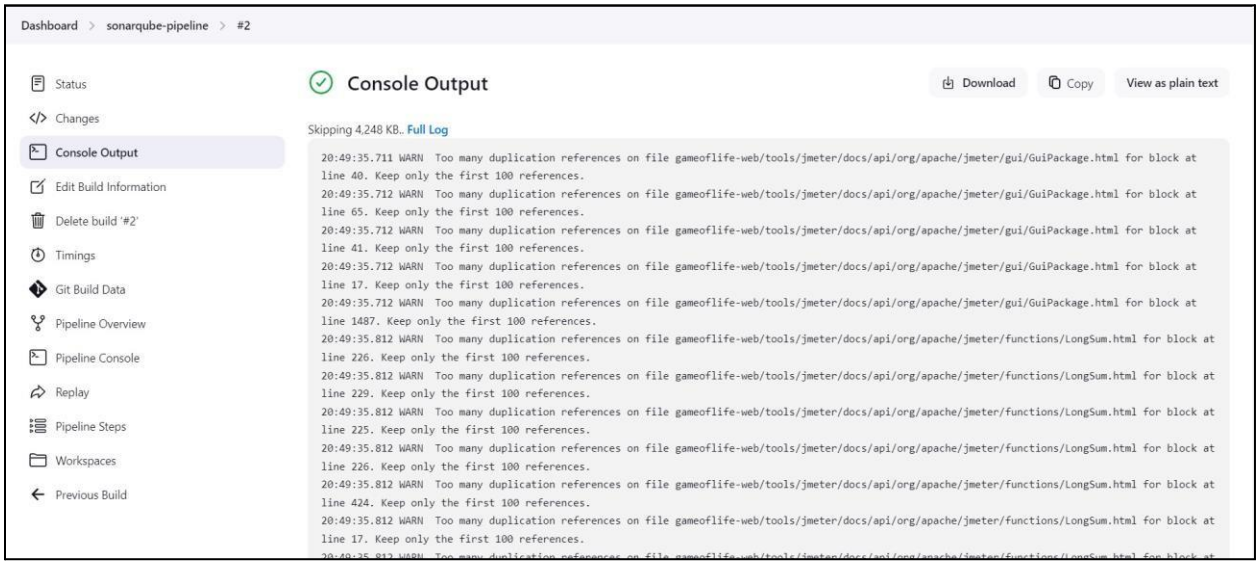




Step 4: Save the pipeline and build it.

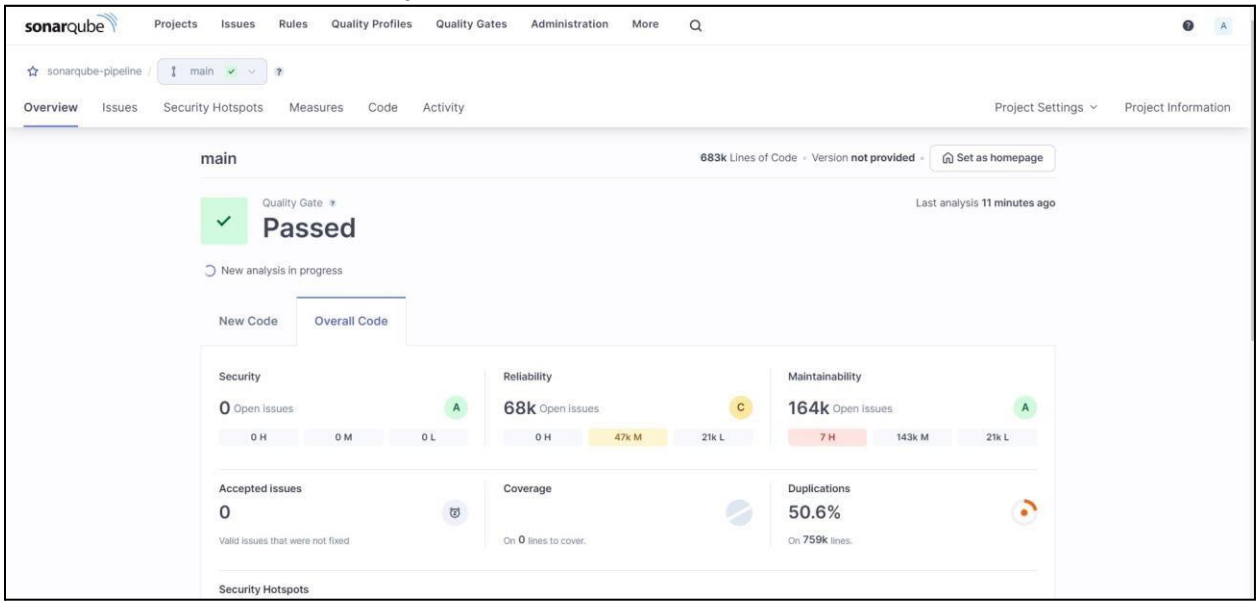


Console output:

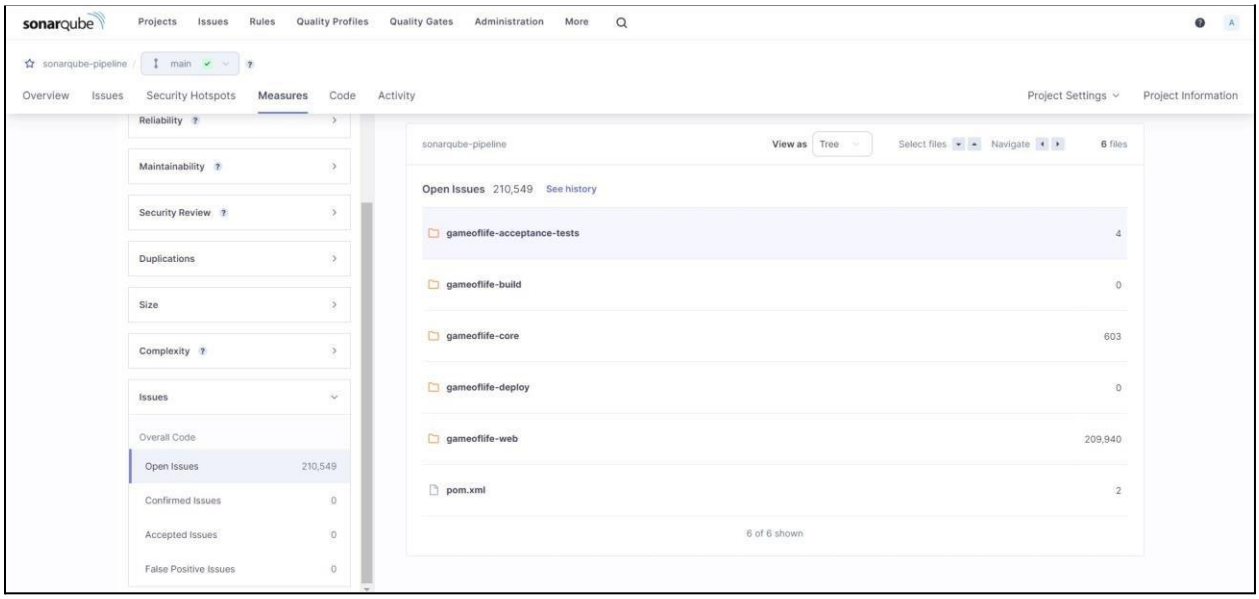


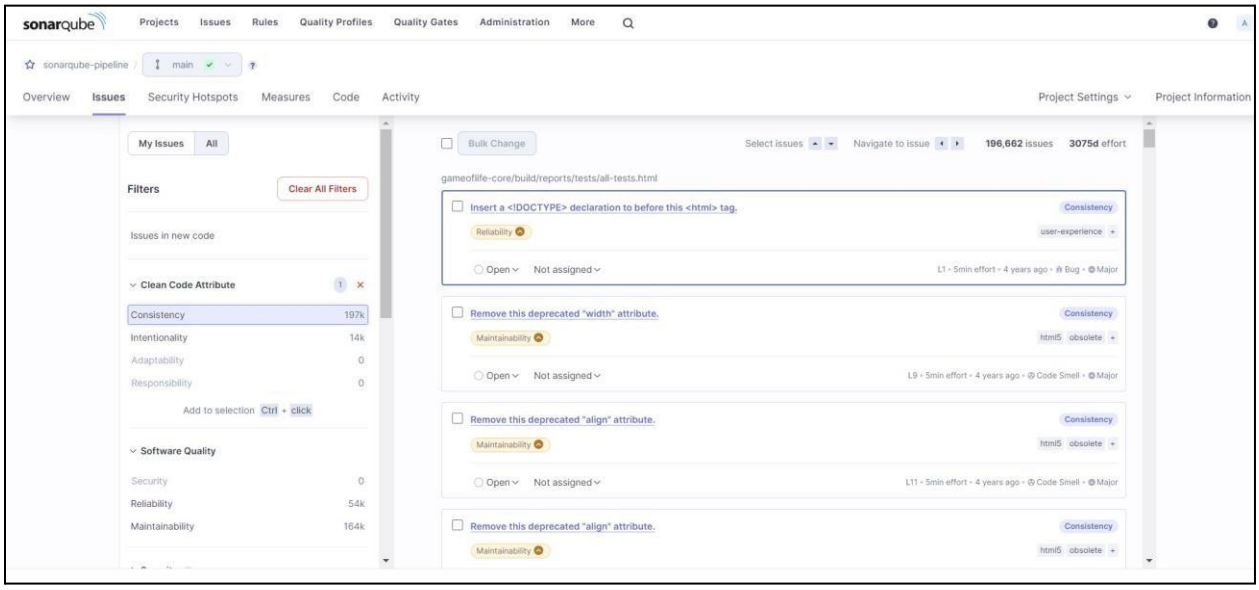
```
20:50:01.832 INFO  ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=sonarqube-pipeline
20:50:01.832 INFO  Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
20:50:01.832 INFO  More about the report processing at http://localhost:9000/api/ce/task?id=159a9d05-1f5f-4e17-bd27-3643a32a836a
20:50:12.108 INFO  Analysis total time: 7:37.235 s
20:50:12.110 INFO  SonarScanner Engine completed successfully
20:50:12.849 INFO  EXECUTION SUCCESS
20:50:12.851 INFO  Total time: 7:44.878s
[Pipeline] }
[Pipeline] // withSonarQubeEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

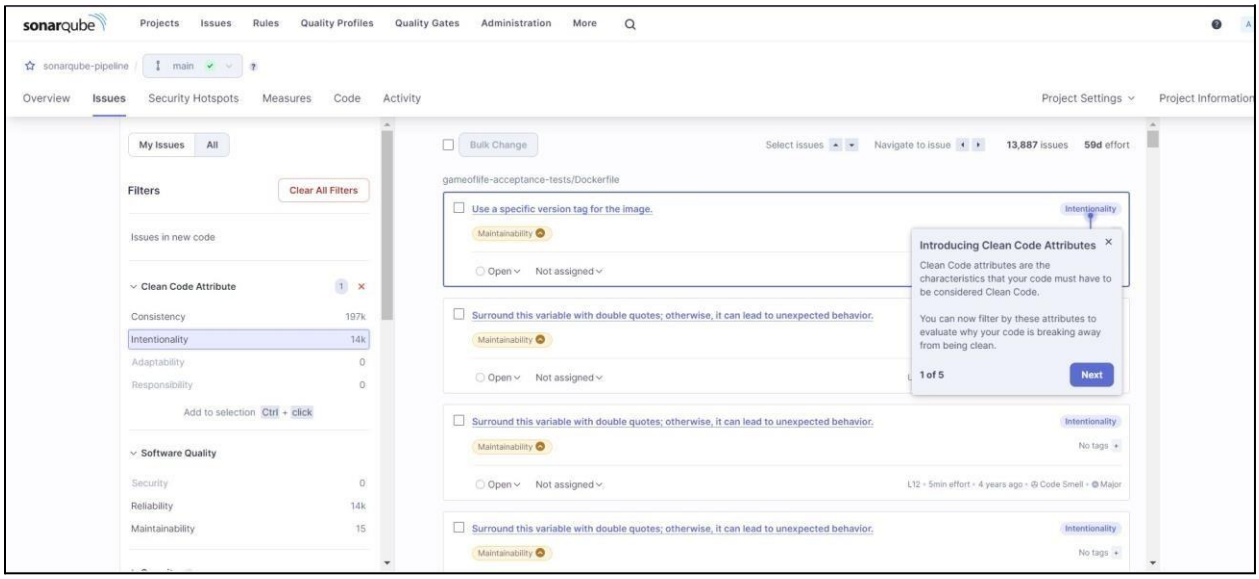
Step 5: After that, check the project in SonarQube

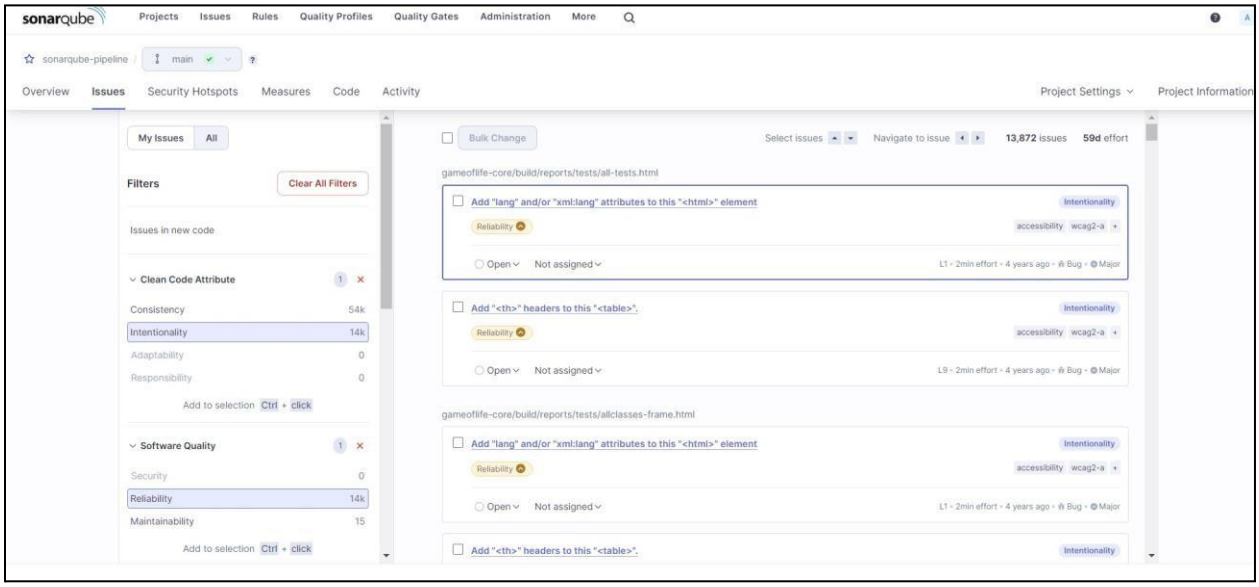


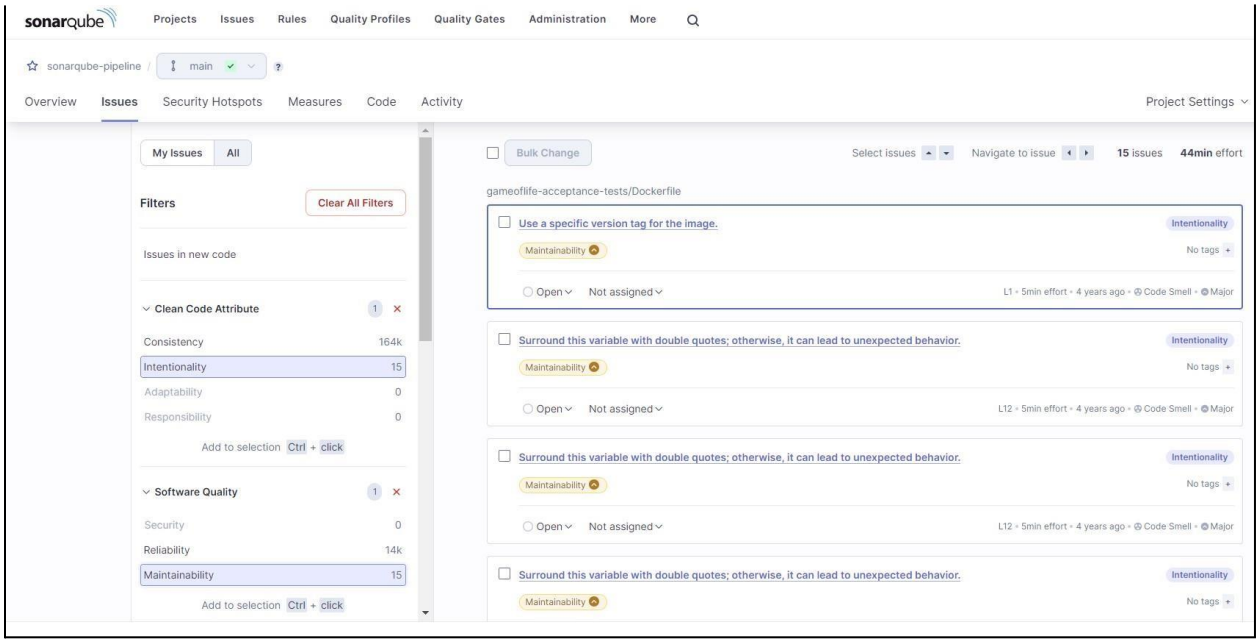
Under different tabs, check all different issues with the code.

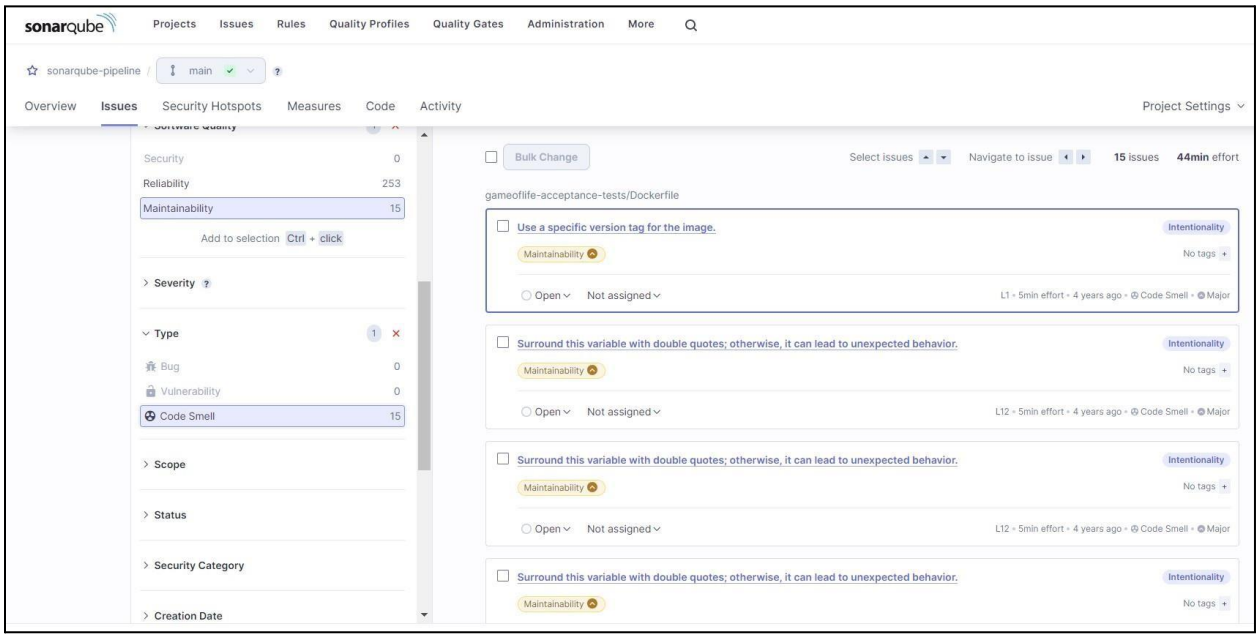

sonarqube
Projects
Issues
Rules
Quality Profiles
Quality Gates
Administration
More
sonarqube-pipeline
main
Overview
Issues
Security Hotspots
Measures
Code
Activity
Project Settings
Project Information
Reliability
Maintainability
Security Review
Duplications
Size
Complexity
Issues
Overall Code
Open Issues 210,549
Confirmed Issues 0
Accepted Issues 0
False Positive Issues 0
sonarqube-pipeline
View as Tree
Select files
Navigate
6 files
Open Issues 210,549 See history
gameoflife-acceptance-tests 4
gameoflife-build 0
gameoflife-core 603
gameoflife-deploy 0
gameoflife-web 209,940
pom.xml 2
6 of 6 shown

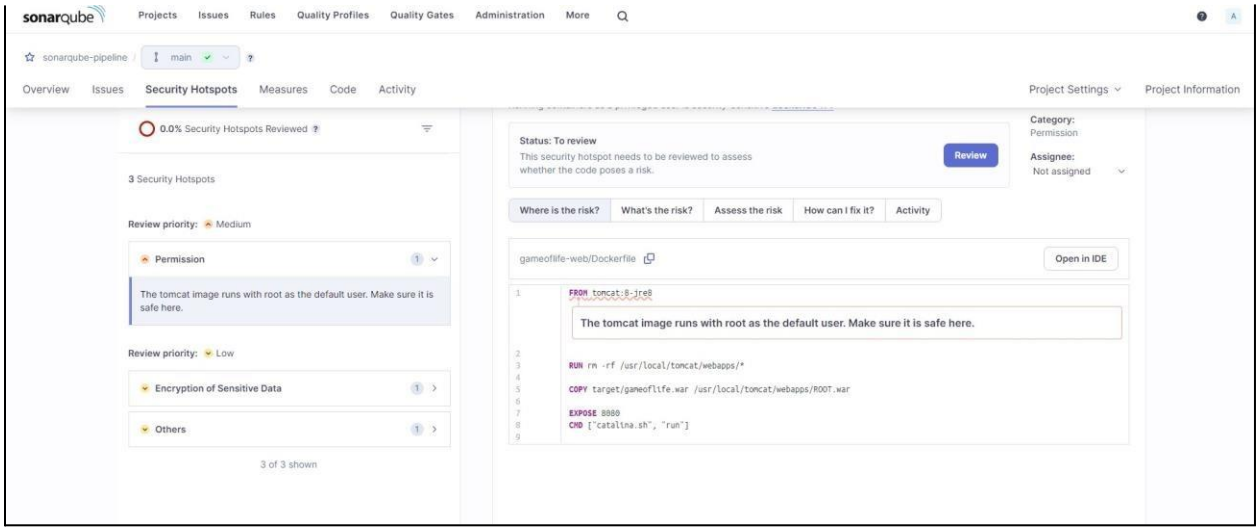

sonarqube
Projects
Issues
Rules
Quality Profiles
Quality Gates
Administration
More
sonarqube-pipeline
main
Overview
Issues
Security Hotspots
Measures
Code
Activity
Project Settings
Project Information
My Issues
All
Filters
Clear All Filters
Issues in new code
Clean Code Attribute
Consistency 197k
Intentionality 14k
Adaptability 0
Responsibility 0
Add to selection Ctrl + click
Software Quality
Security 0
Reliability 54k
Maintainability 164k
Bulk Change
Select issues
Navigate to issue
196,662 issues
3075d effort
gameoflife-core/build/reports/tests/all-tests.html
Insert a <!DOCTYPE> declaration to before this <html> tag.
Consistency
Reliability
user-experience
Open
Not assigned
L1 · 5min effort · 4 years ago · Bug · Major
Remove this deprecated "width" attribute.
Consistency
Maintainability
html5 obsolete
Open
Not assigned
L9 · 5min effort · 4 years ago · Code Smell · Major
Remove this deprecated "align" attribute.
Consistency
Maintainability
html5 obsolete
Open
Not assigned
L11 · 5min effort · 4 years ago · Code Smell · Major
Remove this deprecated "align" attribute.
Consistency
Maintainability
html5 obsolete

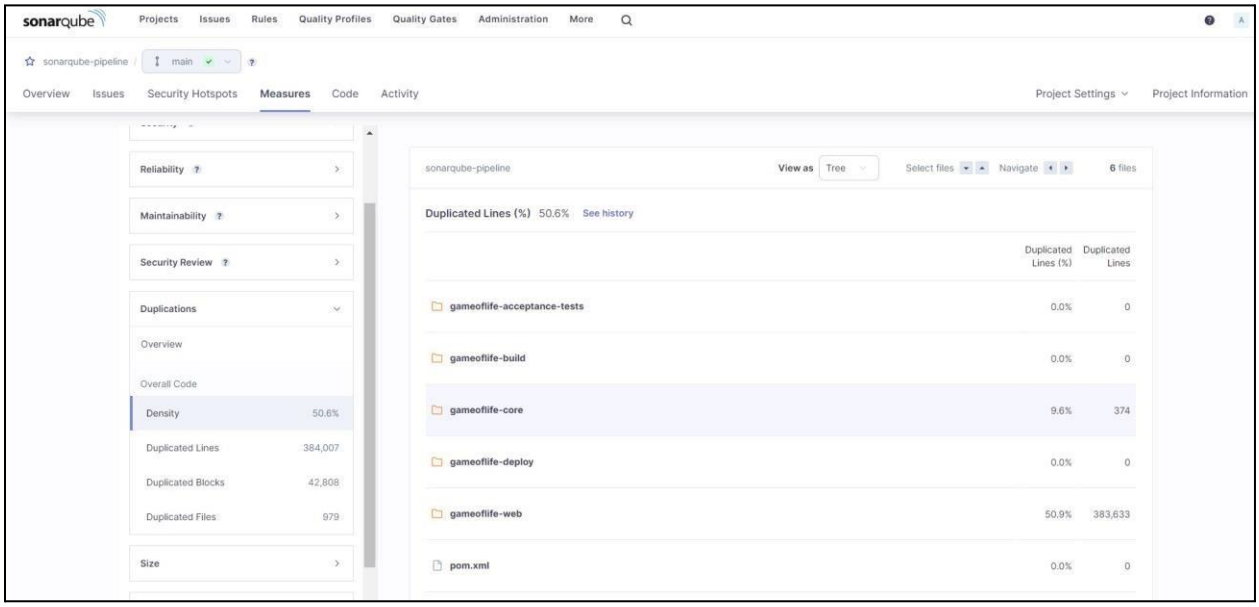

sonarqube
Projects
Issues
Rules
Quality Profiles
Quality Gates
Administration
More
sonarqube-pipeline
main
Overview
Issues
Security Hotspots
Measures
Code
Activity
Project Settings
Project Information
My Issues
All
Filters
Clear All Filters
Issues in new code
Clean Code Attribute
Consistency 197k
Intentionality 14k
Adaptability 0
Responsibility 0
Add to selection Ctrl + click
Software Quality
Security 0
Reliability 14k
Maintainability 15
Bulk Change
Select issues
Navigate to issue
13,887 issues
59d effort
gameoflife-acceptance-tests/Dockerfile
Use a specific version tag for the image.
Intentionality
Maintainability
Open
Not assigned
Introducing Clean Code Attributes
Clean Code attributes are the characteristics that your code must have to be considered Clean Code.
You can now filter by these attributes to evaluate why your code is breaking away from being clean.
1 of 5
Next
Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.
Maintainability
Open
Not assigned
Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.
Intentionality
Maintainability
No tags
Open
Not assigned
L12 • 5min effort • 4 years ago • Code Smell • Major
Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.
Intentionality
Maintainability
No tags

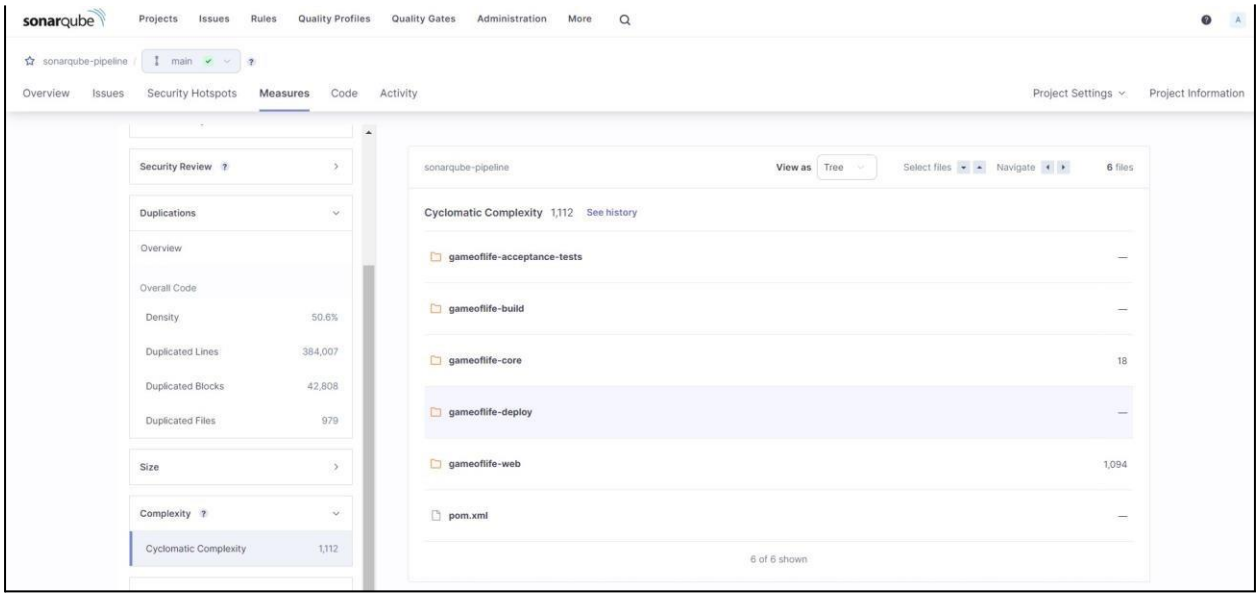

sonarqube
Projects
Issues
Rules
Quality Profiles
Quality Gates
Administration
More
sonarqube-pipeline
main
Overview
Issues
Security Hotspots
Measures
Code
Activity
Project Settings
Project Information
My Issues
All
Filters
Clear All Filters
Issues in new code
Clean Code Attribute
Consistency 54k
Intentionality 14k
Adaptability 0
Responsibility 0
Add to selection Ctrl + click
Software Quality
Security 0
Reliability 14k
Maintainability 15
Bulk Change
Select issues
Navigate to issue
13,872 issues
59d effort
gameoflife-core/build/reports/tests/all-tests.html
Add "lang" and/or "xml:lang" attributes to this "<html>" element
Intentionality
Reliability
accessibility wcag2-a
Open
Not assigned
L1 • 2min effort • 4 years ago • Bug • Major
Add "<th>" headers to this "<table>".
Intentionality
Reliability
accessibility wcag2-a
Open
Not assigned
L9 • 2min effort • 4 years ago • Bug • Major
gameoflife-core/build/reports/tests/allclasses-frame.html
Add "lang" and/or "xml:lang" attributes to this "<html>" element
Intentionality
Reliability
accessibility wcag2-a
Open
Not assigned
L1 • 2min effort • 4 years ago • Bug • Major
Add "<th>" headers to this "<table>".
Intentionality


sonarqube
Projects
Issues
Rules
Quality Profiles
Quality Gates
Administration
More
sonarqube-pipeline
main
Overview
Issues
Security Hotspots
Measures
Code
Activity
Project Settings
My Issues
All
Filters
Clear All Filters
Issues in new code
Clean Code Attribute
Consistency 164k
Intentionality 15
Adaptability 0
Responsibility 0
Add to selection Ctrl + click
Software Quality
Security 0
Reliability 14k
Maintainability 15
Add to selection Ctrl + click
Bulk Change
Select issues
Navigate to issue
15 issues
44min effort
gameoflife-acceptance-tests/Dockerfile
Use a specific version tag for the image.
Intentionality
Maintainability
No tags
Open
Not assigned
L1 • 5min effort • 4 years ago • Code Smell • Major
Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.
Intentionality
Maintainability
No tags
Open
Not assigned
L12 • 5min effort • 4 years ago • Code Smell • Major
Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.
Intentionality
Maintainability
No tags
Open
Not assigned
L12 • 5min effort • 4 years ago • Code Smell • Major
Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.
Intentionality
Maintainability
No tags


sonarqube
Projects
Issues
Rules
Quality Profiles
Quality Gates
Administration
More
sonarqube-pipeline
main
Overview
Issues
Security Hotspots
Measures
Code
Activity
Project Settings
Security 0
Reliability 253
Maintainability 15
Add to selection Ctrl + click
Severity
Type
Bug 0
Vulnerability 0
Code Smell 15
Scope
Status
Security Category
Creation Date
Bulk Change
Select issues
Navigate to issue
15 issues
44min effort
gameoflife-acceptance-tests/Dockerfile
Use a specific version tag for the image.
Intentionality
Maintainability
No tags
Open
Not assigned
L1 • 5min effort • 4 years ago • Code Smell • Major
Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.
Intentionality
Maintainability
No tags
Open
Not assigned
L12 • 5min effort • 4 years ago • Code Smell • Major
Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.
Intentionality
Maintainability
No tags
Open
Not assigned
L12 • 5min effort • 4 years ago • Code Smell • Major
Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.
Intentionality
Maintainability
No tags


sonarqube
Projects
Issues
Rules
Quality Profiles
Quality Gates
Administration
More
sonarqube-pipeline
main
Overview
Issues
Security Hotspots
Measures
Code
Activity
Project Settings
Project Information
0.0% Security Hotspots Reviewed
3 Security Hotspots
Review priority: Medium
Permission
The tomcat image runs with root as the default user. Make sure it is safe here.
Review priority: Low
Encryption of Sensitive Data
Others
3 of 3 shown
Status: To review
This security hotspot needs to be reviewed to assess whether the code poses a risk.
Review
Category:
Permission
Assignee:
Not assigned
Where is the risk?
What's the risk?
Assess the risk
How can I fix it?
Activity
gameoflife-web/Dockerfile
Open in IDE
FROM tomcat:8-jre8
The tomcat image runs with root as the default user. Make sure it is safe here.
RUN rm -rf /usr/local/tomcat/webapps/*
COPY target/gameoflife.war /usr/local/tomcat/webapps/ROOT.war
EXPOSE 8080
CMD ["catalina.sh", "run"]


sonarqube
Projects
Issues
Rules
Quality Profiles
Quality Gates
Administration
More
sonarqube-pipeline
main
Overview
Issues
Security Hotspots
Measures
Code
Activity
Project Settings
Project Information
Reliability
Maintainability
Security Review
Duplications
Overview
Overall Code
Density 50.6%
Duplicated Lines 384,007
Duplicated Blocks 42,808
Duplicated Files 979
Size
sonarqube-pipeline
View as Tree
Select files
Navigate
6 files
Duplicated Lines (%) 50.6%
See history
Duplicated Lines (%)
Duplicated Lines
gameoflife-acceptance-tests 0.0% 0
gameoflife-build 0.0% 0
gameoflife-core 9.6% 374
gameoflife-deploy 0.0% 0
gameoflife-web 50.9% 383,633
pom.xml 0.0% 0


sonarqube
Projects
Issues
Rules
Quality Profiles
Quality Gates
Administration
More
sonarqube-pipeline
main
Overview
Issues
Security Hotspots
Measures
Code
Activity
Project Settings
Project Information
Security Review
Duplications
Overview
Overall Code
Density 50.6%
Duplicated Lines 384,007
Duplicated Blocks 42,808
Duplicated Files 979
Size
Complexity
Cyclomatic Complexity 1,112
sonarqube-pipeline
View as Tree
Select files
Navigate
6 files
Cyclomatic Complexity 1,112 See history
gameoflife-acceptance-tests —
gameoflife-build —
gameoflife-core 18
gameoflife-deploy —
gameoflife-web 1,094
pom.xml —
6 of 6 shown

Name : **Himesh Pathai**

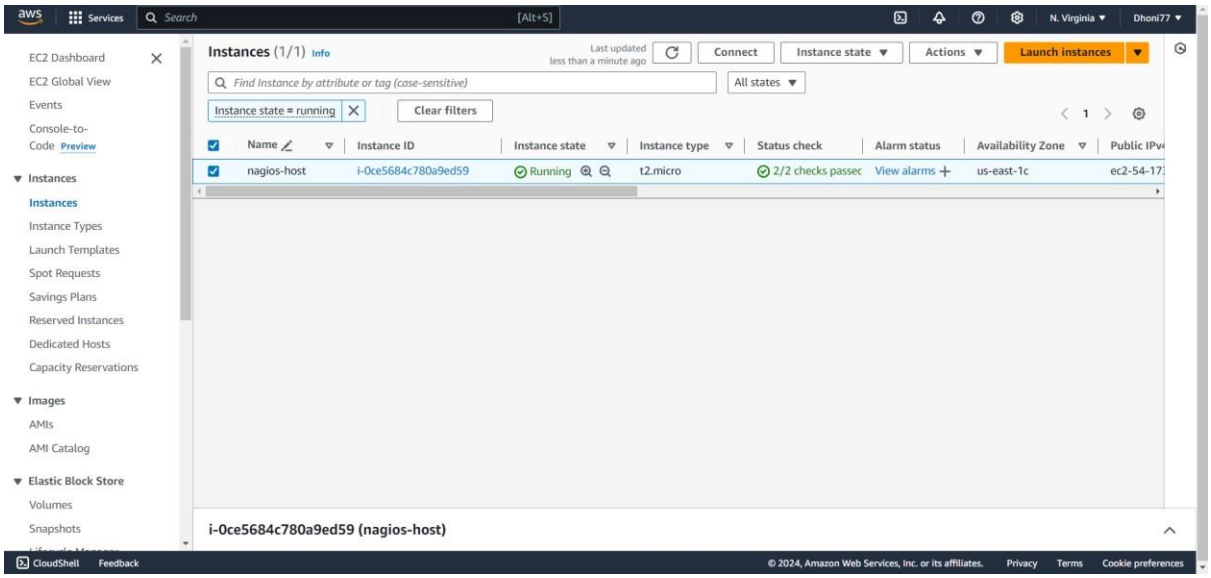
Class : **D15A**

Roll No. : **35**

EXPERIMENT 9

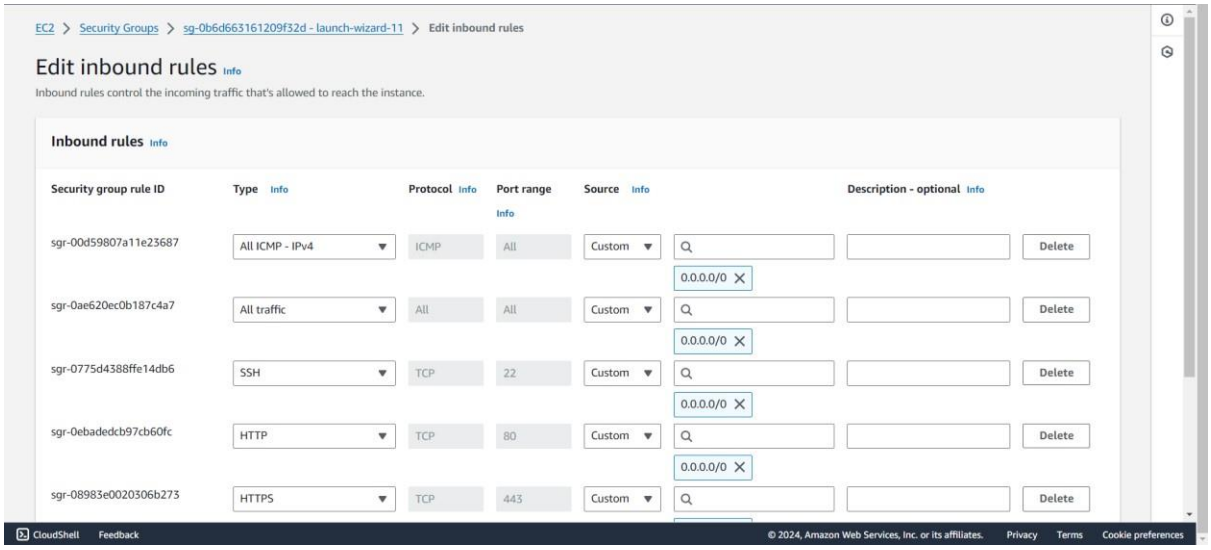
Aim :- To Understand Continuous monitoring and Installation and configuration of Nagios Core, Nagios Plugins and NRPE (Nagios Remote Plugin Executor) on Linux Machine.

1. Create an Amazon Linux EC2 Instance



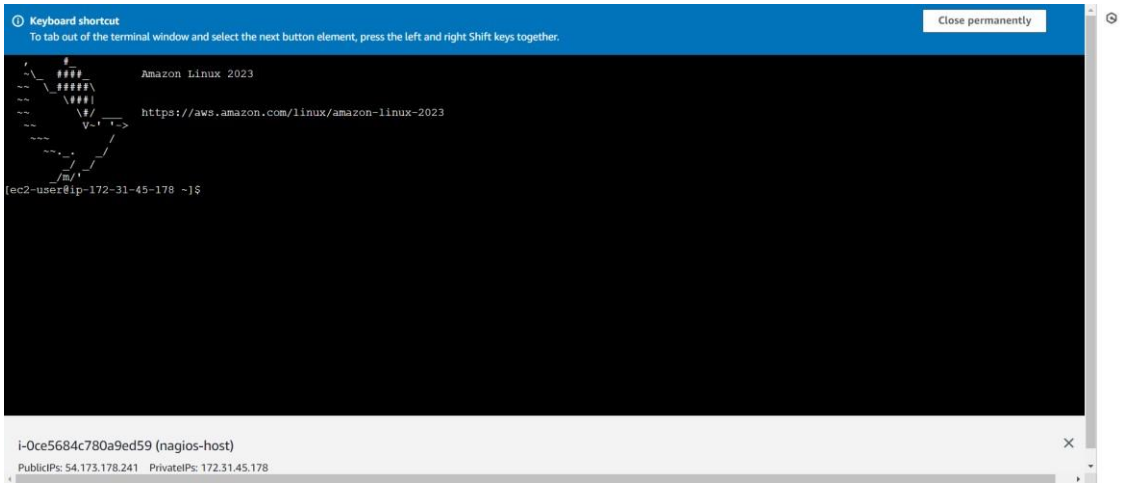
2. Configure Security Group

- Ensure HTTP, HTTPS, SSH, and ICMP are open from everywhere.
- Edit the inbound rules of the specified Security Group

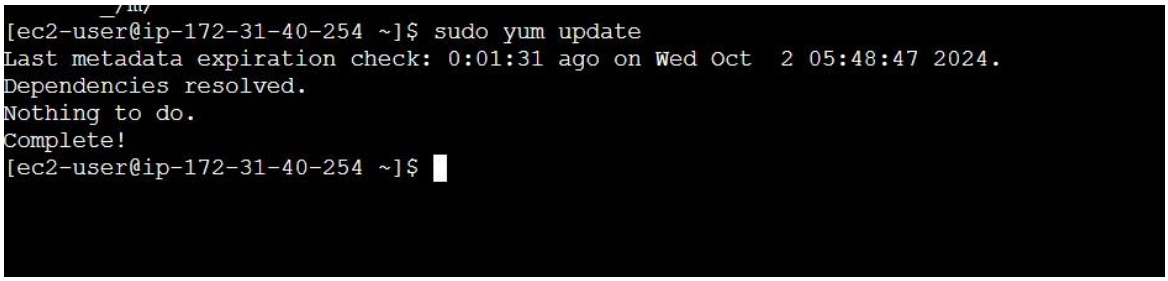


You have to edit the inbound rules of the specified Security Group for this.

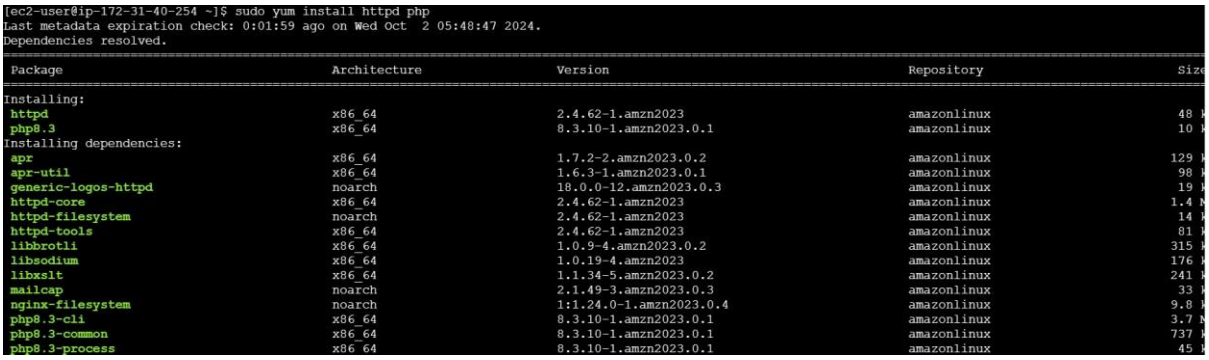
3. **SSH into Your EC2 instance or simply use EC2 Instance Connect from the browser.**



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



sudo yum install httpd php



sudo yum install gcc glibc glibc-common

```
[ec2-user@ip-172-31-40-254 ~]$ sudo yum install gcc glibc glibc-common
Last metadata expiration check: 0:02:41 ago on Wed Oct  2 05:48:47 2024.
Package glibc-2.34-52.amzn2023.0.11.x86_64 is already installed.
Package glibc-common-2.34-52.amzn2023.0.11.x86_64 is already installed.
Dependencies resolved.
```

Package	Architecture	Version	Repository	Size
Installing:				
gcc	x86_64	11.4.1-2.amzn2023.0.2	amazonlinux	32 M
Installing dependencies:				
annobin-docs	noarch	10.93-1.amzn2023.0.1	amazonlinux	92 k
annobin-plugin-gcc	x86_64	10.93-1.amzn2023.0.1	amazonlinux	887 k
cpp	x86_64	11.4.1-2.amzn2023.0.2	amazonlinux	10 M
gc	x86_64	8.0.4-5.amzn2023.0.2	amazonlinux	105 k
glibc-devel	x86_64	2.34-52.amzn2023.0.11	amazonlinux	27 k
glibc-headers-x86	noarch	2.34-52.amzn2023.0.11	amazonlinux	427 k
guile22	x86_64	2.2.7-2.amzn2023.0.3	amazonlinux	6.4 M
kernel-headers	x86_64	6.1.109-118.189.amzn2023	amazonlinux	1.4 M
libmpc	x86_64	1.2.1-2.amzn2023.0.2	amazonlinux	62 k
libtool-ltdl	x86_64	2.4.7-1.amzn2023.0.3	amazonlinux	38 k
libxcrypt-devel	x86_64	4.4.33-7.amzn2023	amazonlinux	32 k
make	x86_64	1:4.3-5.amzn2023.0.2	amazonlinux	534 k
Transaction Summary				

sudo yum install gd gd-devel

```
[ec2-user@ip-172-31-40-254 ~]$ sudo yum install gd gd-devel
Last metadata expiration check: 0:03:46 ago on Wed Oct  2 05:48:47 2024.
Dependencies resolved.
```

Package	Architecture	Version	Repository	Size
Installing:				
gd	x86_64	2.3.3-5.amzn2023.0.3	amazonlinux	139 k
gd-devel	x86_64	2.3.3-5.amzn2023.0.3	amazonlinux	38 k
Installing dependencies:				
brotli	x86_64	1.0.9-4.amzn2023.0.2	amazonlinux	314 k
brotli-devel	x86_64	1.0.9-4.amzn2023.0.2	amazonlinux	31 k
bzip2-devel	x86_64	1.0.8-6.amzn2023.0.2	amazonlinux	214 k
cairo	x86_64	1.17.6-2.amzn2023.0.1	amazonlinux	684 k
cmake-filesystem	x86_64	3.22.2-1.amzn2023.0.4	amazonlinux	16 k
fontconfig	x86_64	2.13.94-2.amzn2023.0.2	amazonlinux	273 k
fontconfig-devel	x86_64	2.13.94-2.amzn2023.0.2	amazonlinux	128 k
fonts-filesystem	noarch	1:2.0.5-12.amzn2023.0.2	amazonlinux	9.5 k
freetype	x86_64	2.13.2-5.amzn2023.0.1	amazonlinux	423 k
freetype-devel	x86_64	2.13.2-5.amzn2023.0.1	amazonlinux	912 k
glib2-devel	x86_64	2.74.7-689.amzn2023.0.2	amazonlinux	486 k
google-noto-fonts-common	noarch	20201206-2.amzn2023.0.2	amazonlinux	15 k
google-noto-sans-vf-fonts	noarch	20201206-2.amzn2023.0.2	amazonlinux	492 k
graphite2	x86_64	1.3.14-7.amzn2023.0.2	amazonlinux	97 k
graphite2-devel	x86_64	1.3.14-7.amzn2023.0.2	amazonlinux	21 k
harfbuzz	x86_64	7.0.0-2.amzn2023.0.1	amazonlinux	868 k
harfbuzz-devel	x86_64	7.0.0-2.amzn2023.0.1	amazonlinux	404 k
harfbuzz-icu	x86_64	7.0.0-2.amzn2023.0.1	amazonlinux	18 k

5. **Create a new Nagios User with its password. You'll have to enter the password twice for confirmation.** sudo adduser -m nagios sudo passwd nagios

```
[ec2-user@ip-172-31-40-254 ~]$ sudo adduser -m nagios
[ec2-user@ip-172-31-40-254 ~]$ sudo passwd nagios
Changing password for user nagios.
New password:
BAD PASSWORD: The password is a palindrome
Retype new password:
Sorry, passwords do not match.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[ec2-user@ip-172-31-40-254 ~]$
```

6. **Create a new user group** sudo groupadd nagcmd

```
[ec2-user@ip-172-31-40-254 ~]$ sudo groupadd nagcmd
[ec2-user@ip-172-31-40-254 ~]$
```

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

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

```
[ec2-user@ip-172-31-40-254 ~]$ sudo groupadd nagcmd
[ec2-user@ip-172-31-40-254 ~]$ sudo usermod -a -G nagcmd nagios
sudo usermod -a -G nagcmd apache
[ec2-user@ip-172-31-40-254 ~]$
```

8. **Create a new directory for Nagios downloads** mkdir ~/downloads cd ~/downloads

```
[ec2-user@ip-172-31-40-254 ~]$ mkdir ~/downloads
cd ~/downloads
[ec2-user@ip-172-31-40-254 downloads]$
```

9. **Use wget to download the source zip files.**

Wget https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.4.6.tar.gz wget

```
[ec2-user@ip-172-31-40-254 downloads]$ Wget https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.4.6.tar.gz
wget https://nagios-plugins.org/download/nagios-plugins-2.3.3.tar.gz
-bash: Wget: command not found
--2024-10-02 06:15:45--  https://nagios-plugins.org/download/nagios-plugins-2.3.3.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: 2782610 (2.7M) [application/x-gzip]
Saving to: 'nagios-plugins-2.3.3.tar.gz'

nagios-plugins-2.3.3.tar.gz               0%[                                                                                 ]       0  --.-KB/s
nagios-plugins-2.3.3.tar.gz              23%[=================>                                                               ] 632.00K  3.02MB/s
nagios-plugins-2.3.3.tar.gz             100%[=============================================================================>]   2.65M  8.10MB/s    in 0.3s

2024-10-02 06:15:46 (8.10 MB/s) - 'nagios-plugins-2.3.3.tar.gz' saved [2782610/2782610]

[ec2-user@ip-172-31-40-254 downloads]$
```

https://nagios-plugins.org/download/nagios-plugins-2.3.3.tar.gz

```
[ec2-user@ip-172-31-40-254 downloads]$ wget https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.4.6.tar.gz
--2024-10-02 06:17:24--  https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.4.6.tar.gz
Resolving assets.nagios.com (assets.nagios.com)... 45.79.49.120, 2600:3c00::f03c:92ff:fef7:45ce
Connecting to assets.nagios.com (assets.nagios.com)|45.79.49.120|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 11333414 (11M) [application/x-gzip]
Saving to: 'nagios-4.4.6.tar.gz'

nagios-4.4.6.tar.gz                       0%[                                                                                 ]       0  --.-KB/s
nagios-4.4.6.tar.gz                       4%[==>                                                                              ] 495.62K  2.40MB/s
nagios-4.4.6.tar.gz                      30%[=========>                                                                       ]   3.26M  7.99MB/s
nagios-4.4.6.tar.gz                      63%[====================>                                                                ]   6.91M  11.0MB/s
nagios-4.4.6.tar.gz                      96%[==============================>                                                      ]  10.46M  12.6MB/s
nagios-4.4.6.tar.gz                     100%[=============================================================================>]  10.81M  12.9MB/s    in 0.8s

2024-10-02 06:17:25 (12.9 MB/s) - 'nagios-4.4.6.tar.gz' saved [11333414/11333414]

[ec2-user@ip-172-31-40-254 downloads]$
```

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

tar zxvf nagios-4.4.6.tar.gz cd
nagios-4.4.6

```
[ec2-user@ip-172-31-40-254 downloads]$ tar zxvf nagios-4.4.6.tar.gz
nagios-4.4.6/
nagios-4.4.6/.gitignore
nagios-4.4.6/.travis.yml
nagios-4.4.6/CONTRIBUTING.md
nagios-4.4.6/Changelog
nagios-4.4.6/INSTALLING
nagios-4.4.6/LEGAL
nagios-4.4.6/LICENSE
nagios-4.4.6/Makefile.in
nagios-4.4.6/README.md
nagios-4.4.6/THANKS
nagios-4.4.6/UPGRADING
nagios-4.4.6/aclocal.m4
nagios-4.4.6/autoconf-macros/
nagios-4.4.6/autoconf-macros/.gitignore
nagios-4.4.6/autoconf-macros/CHANGELOG.md
nagios-4.4.6/autoconf-macros/LICENSE
nagios-4.4.6/autoconf-macros/LICENSE.md
nagios-4.4.6/autoconf-macros/README.md
nagios-4.4.6/autoconf-macros/add_group_user
nagios-4.4.6/autoconf-macros/ax_nagios_get_distrib
nagios-4.4.6/autoconf-macros/ax_nagios_get_files
nagios-4.4.6/autoconf-macros/ax_nagios_get_inetd
nagios-4.4.6/autoconf-macros/ax_nagios_get_init
nagios-4.4.6/autoconf-macros/ax_nagios_get_os
nagios-4.4.6/autoconf-macros/ax_nagios_get_paths
nagios-4.4.6/autoconf-macros/ax_nagios_get_ssl
nagios-4.4.6/base/
nagios-4.4.6/base/.gitignore
nagios-4.4.6/base/Makefile.in
nagios-4.4.6/base/broker.c
```

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

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

```
[ec2-user@ip-172-31-40-254 nagios-4.4.6]$ ./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 we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking whether make sets $(MAKE)... yes
checking whether ln -s works... yes
checking for strip... /usr/bin/strip
checking how to run the C preprocessor... gcc -E
checking for grep that handles long lines and -e... /usr/bin/grep
checking for egrep... /usr/bin/grep -E
checking for ANSI C header files... yes
checking whether time.h and sys/time.h may both be included... yes
checking for sys/wait.h that is POSIX.1 compatible... yes
checking for sys/types.h... yes
checking for sys/stat.h... yes
checking for stdlib.h... yes
checking for string.h... yes
checking for memory.h... yes
checking for strings.h... yes
checking for inttypes.h... yes
checking for stdint.h... yes
checking for unistd.h... yes
checking arpa/inet.h usability... yes
checking arpa/inet.h presence... yes
checking for arpa/inet.h... yes
```

12. Compile the source code.

make all

```
[ec2-user@ip-172-31-40-254 nagios-4.4.6]$ make all
cd ./base && make
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.4.6/base'
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o nagios.o nagios.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o broker.o broker.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o nebmods.o nebmods.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o ../common/shared.o ../common/shared.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o query-handler.o query-handler.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o workers.o workers.c
In function 'get_wproc_list',
    inlined from 'get_worker' at workers.c:277:12:
workers.c:253:17: warning: '%s' directive argument is null [-Wformat-overflow=]
  253 |                 log_debug_info(DEBUGL_CHECKS, 1, "Found specialized worker(s) for '%s'", (slash && *slash != '/') ? slash : cmd_name);
      |                 ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o checks.o checks.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o config.o config.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o commands.o commands.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o events.o events.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o flapping.o flapping.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o logging.o logging.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o macros-base.o ../common/macros.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o netutils.o netutils.c
netutils.c: In function 'my_tcp_connect':
netutils.c:50:47: warning: '%d' directive output may be truncated writing between 1 and 11 bytes into a region of size 6 [-Wformat-truncation=]
   50 |         snprintf(port_str, sizeof(port_str), "%d", port);
      |                                               ^~
netutils.c:50:46: note: directive argument in the range [-2147483648, 65535]
   50 |         snprintf(port_str, sizeof(port_str), "%d", port);
      |                                              ^~~~
netutils.c:50:9: note: 'snprintf' output between 2 and 12 bytes into a destination of size 6
   50 |         snprintf(port_str, sizeof(port_str), "%d", port);
      |         ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o notifications.o notifications.c
```

```
  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.
```

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

./sudo make install sudo make

install-init sudo make install-

config sudo make install-

commandmode

```
[ec2-user@ip-172-31-40-254 nagios-4.4.6]$ ./sudo make install
sudo make install-init
sudo make install-config
sudo make install-commandmode
-bash: ./sudo: No such file or directory
/usr/bin/install -c -m 755 -d -o root -g root /lib/systemd/system
/usr/bin/install -c -m 755 -o root -g root startup/default-service /lib/systemd/system/nagios.service
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc/objects
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/nagios.cfg /usr/local/nagios/etc/nagios.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/cgi.cfg /usr/local/nagios/etc/cgi.cfg
/usr/bin/install -c -b -m 660 -o nagios -g nagios sample-config/resource.cfg /usr/local/nagios/etc/resource.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/templates.cfg /usr/local/nagios/etc/objects/templates.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/commands.cfg /usr/local/nagios/etc/objects/commands.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/contacts.cfg /usr/local/nagios/etc/objects/contacts.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/timeperiods.cfg /usr/local/nagios/etc/objects/timeperiods.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/localhost.cfg /usr/local/nagios/etc/objects/localhost.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/windows.cfg /usr/local/nagios/etc/objects/windows.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/printer.cfg /usr/local/nagios/etc/objects/printer.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/switch.cfg /usr/local/nagios/etc/objects/switch.cfg

*** Config files installed ***

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

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

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

14. Edit the config file and change the email address.

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

```
  GNU nano 5.8                                  /usr/local/nagios/etc/objects/contacts.cfg                                  Modified
# Just one contact defined by default - the Nagios admin (that's you)
# This contact definition inherits a lot of default values from the
# 'generic-contact' template which is defined elsewhere.

define contact {

    contact_name            nagiosadmin             ; Short name of user
    use                     generic-contact         ; Inherit default values from generic-contact template (defined above)
    alias                   Nagios Admin            ; Full name of user
    email                   bhagyeshpatil0702@gmail.com ; <<***** CHANGE THIS TO YOUR EMAIL ADDRESS ******
}



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

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

define contactgroup {

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

15. Configure the web interface.

sudo make install-webconf

```
[ec2-user@ip-172-31-40-254 nagios-4.4.6]$ 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-172-31-40-254 nagios-4.4.6]$
```

16. Create a nagiosadmin account for nagios login along with password. You'll have to specify the password twice. sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin

```
[ec2-user@ip-172-31-40-254 nagios-4.4.6]$ sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
New password:
Re-type new password:
Adding password for user nagiosadmin
[ec2-user@ip-172-31-40-254 nagios-4.4.6]$
```

17. Restart Apache sudo systemctl restart httpd

```
Adding password for user nagiosadmin
[ec2-user@ip-172-31-40-254 nagios-4.4.6]$ sudo systemctl restart httpd
[ec2-user@ip-172-31-40-254 nagios-4.4.6]$
```

cd ~/downloads tar zxvf nagios-

plugins-2.3.3.tar.gz cd nagios-

plugins-2.3.3

```
[ec2-user@ip-172-31-40-254 nagios-4.4.6]$ sudo systemctl restart httpd
[ec2-user@ip-172-31-40-254 nagios-4.4.6]$ cd ~/downloads
tar zxvf nagios-plugins-2.3.3.tar.gz
 cd nagios-plugins-2.3.3
nagios-plugins-2.3.3/
nagios-plugins-2.3.3/perlmods/
nagios-plugins-2.3.3/perlmods/Config-Tiny-2.14.tar.gz
nagios-plugins-2.3.3/perlmods/parent-0.226.tar.gz
nagios-plugins-2.3.3/perlmods/Test-Simple-0.98.tar.gz
nagios-plugins-2.3.3/perlmods/Makefile.in
nagios-plugins-2.3.3/perlmods/version-0.9903.tar.gz
nagios-plugins-2.3.3/perlmods/Makefile.am
nagios-plugins-2.3.3/perlmods/Module-Runtime-0.013.tar.gz
nagios-plugins-2.3.3/perlmods/Module-Metadata-1.000014.tar.gz
nagios-plugins-2.3.3/perlmods/Params-Validate-1.08.tar.gz
nagios-plugins-2.3.3/perlmods/Class-Accessor-0.34.tar.gz
nagios-plugins-2.3.3/perlmods/Try-Tiny-0.18.tar.gz
nagios-plugins-2.3.3/perlmods/Module-Implementation-0.07.tar.gz
nagios-plugins-2.3.3/perlmods/Makefile
nagios-plugins-2.3.3/perlmods/Perl-OSType-1.003.tar.gz
nagios-plugins-2.3.3/perlmods/install_order
nagios-plugins-2.3.3/perlmods/Nagios-Plugin-0.36.tar.gz
nagios-plugins-2.3.3/perlmods/Math-Calc-Units-1.07.tar.gz
nagios-plugins-2.3.3/perlmods/Module-Build-0.4007.tar.gz
nagios-plugins-2.3.3/ABOUT-NLS
nagios-plugins-2.3.3/configure.ac
nagios-plugins-2.3.3/Makefile.in
nagios-plugins-2.3.3/config.h.in
nagios-plugins-2.3.3/ChangeLog
nagios-plugins-2.3.3/AUTHORS
nagios-plugins-2.3.3/lib/
nagios-plugins-2.3.3/lib/parse_ini.h
nagios-plugins-2.3.3/lib/extra_opts.c
nagios-plugins-2.3.3/lib/Makefile.in
```

18. Go back to the downloads folder and unzip the plugins zip file. ./configure --with-nagios-user=nagios --with-nagios-

group=nagios make sudo make install

```
[ec2-user@ip-172-31-80-22 nagios-plugins-2.3.3]$ ./configure --with-nagios-user=nagios --with-nagios-group=nagios
make
sudo make install
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 to disable maintainer-specific portions of Makefiles... yes
checking build system type... x86_64-unknown-linux-gnu
checking host system type... x86_64-unknown-linux-gnu
checking for gcc... gcc
checking for C compiler default output file name... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking for style of include used by make... GNU
checking dependency style of gcc... gcc3
checking how to run the C preprocessor... gcc -E
checking for grep that handles long lines and -e... /usr/bin/grep
```

19. Compile and install plugins sudo chkconfig --add nagios sudo chkconfig nagios on

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

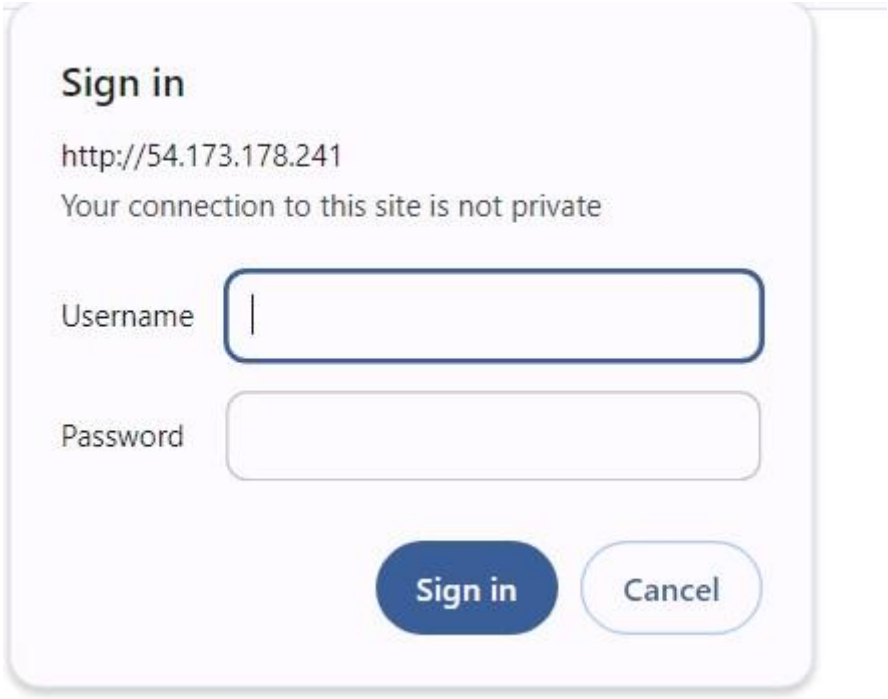
systemctl start nagios

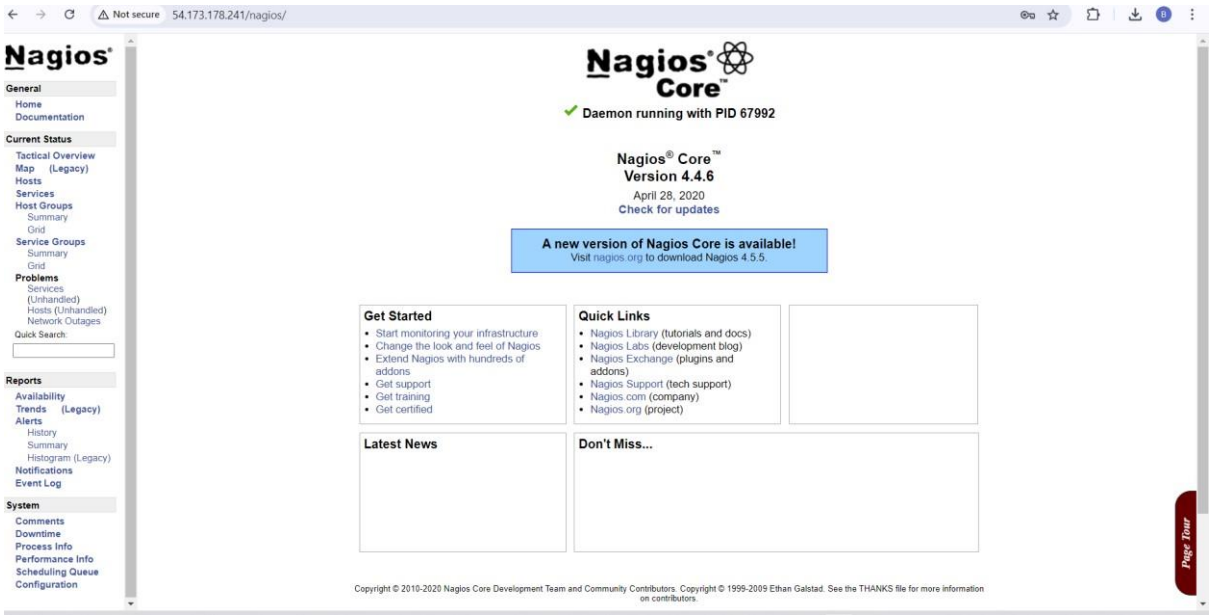
```
[ec2-user@ip-172-31-80-22 nagios-plugins-2.3.3]$ sudo chkconfig --add nagios
sudo chkconfig nagios on
sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg
sudo systemctl start nagios
Error reading information on service nagios: No such file or directory
Note: Forwarding request to 'systemctl enable nagios.service'.
Created symlink /etc/systemd/system/multi-user.target.wants/nagios.service → /usr/lib/systemd/system/nagios.service.

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

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

Running pre-flight check on configuration data...

Checking objects...
       Checked 8 services.
       Checked 1 hosts.
       Checked 1 host groups.
```

20. Check the status of Nagios

```
Things look okay - No serious problems were detected during the pre-flight check
[ec2-user@ip-172-31-45-178 nagios-plugins-2.3.3]$ sudo systemctl status nagios
● nagios.service - Nagios Core 4.4.6
     Loaded: loaded (/usr/lib/systemd/system/nagios.service; enabled; preset: disabled)
     Active: active (running) since Wed 2024-10-02 05:37:36 UTC; 14s ago
       Docs: https://www.nagios.org/documentation
    Process: 67990 ExecStartPre=/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
    Process: 67991 ExecStart=/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
   Main PID: 67992 (nagios)
      Tasks: 6 (limit: 1112)
     Memory: 2.0M
        CPU: 16ms
     CGroup: /system.slice/nagios.service
             ├─67992 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─67993 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─67994 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─67995 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─67996 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             └─67997 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg

Oct 02 05:37:36 ip-172-31-45-178.ec2.internal nagios[67992]: qh: Socket '/usr/local/nagios/var/rw/nagios.qh' successfully initialized
Oct 02 05:37:36 ip-172-31-45-178.ec2.internal nagios[67992]: qh: core query handler registered
```

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


Sign in
http://54.173.178.241
Your connection to this site is not private
Username
Password
Sign in
Cancel


Not secure 54.173.178.241/nagios/
Nagios
General
Home
Documentation
Current Status
Tactical Overview
Map (Legacy)
Hosts
Services
Host Groups
Summary
Grid
Service Groups
Summary
Grid
Problems
Services (Unhandled)
Hosts (Unhandled)
Network Outages
Quick Search:
Reports
Availability
Trends (Legacy)
Alerts
History
Summary
Histogram (Legacy)
Notifications
Event Log
System
Comments
Downtime
Process Info
Performance Info
Scheduling Queue
Configuration
Nagios Core
Daemon running with PID 67992
Nagios® Core™
Version 4.4.6
April 28, 2020
Check for updates
A new version of Nagios Core is available!
Visit nagios.org to download Nagios 4.5.5.
Get Started
Start monitoring your infrastructure
Change the look and feel of Nagios
Extend Nagios with hundreds of addons
Get support
Get training
Get certified
Quick Links
Nagios Library (tutorials and docs)
Nagios Labs (development blog)
Nagios Exchange (plugins and addons)
Nagios Support (tech support)
Nagios.com (company)
Nagios.org (project)
Latest News
Don't Miss...
Page Tour
Copyright © 2010-2020 Nagios Core Development Team and Community Contributors. Copyright © 1999-2009 Ethan Galstad. See the THANKS file for more information on contributors.

Name : Himesh Pathai
Class : D15A
Roll No. : 35

EXPERIMENT 10

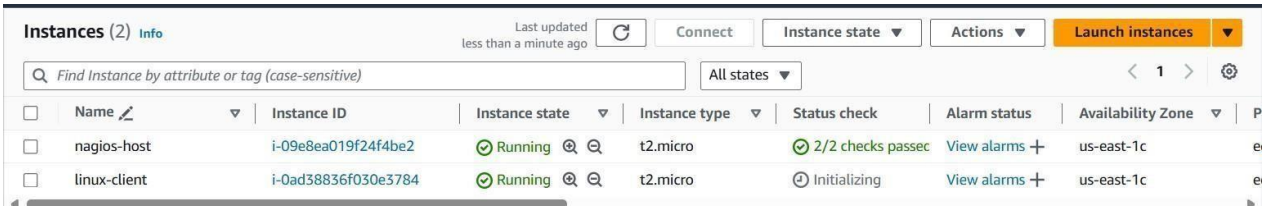
Aim: To perform Port, Service monitoring, Windows/Linux server monitoring using Nagios.

Procedure:-

Check if the nagios service is running by executing following command
sudo systemctl status nagios

```
ubuntu@ip-172-31-89-161:~$ sudo systemctl status nagios
● nagios.service - Nagios Core 4.4.6
     Loaded: loaded (/usr/lib/systemd/system/nagios.service; enabled; preset: enabled)
     Active: active (running) since Sat 2024-09-28 16:08:58 UTC; 1min 2s ago
       Docs: https://www.nagios.org/documentation
    Process: 15743 ExecStartPre=/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
    Process: 15753 ExecStart=/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
   Main PID: 15764 (nagios)
      Tasks: 6 (limit: 1130)
     Memory: 2.4M (peak: 3.2M)
        CPU: 29ms
     CGroup: /system.slice/nagios.service
             ├─15764 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─15765 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─15766 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─15767 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─15768 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             └─15769 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg

Sep 28 16:08:58 ip-172-31-89-161 nagios[15764]: qh: Socket '/usr/local/nagios/var/rw/nagios.qh' successfully initialized
Sep 28 16:08:58 ip-172-31-89-161 nagios[15764]: qh: core query handler registered
Sep 28 16:08:58 ip-172-31-89-161 nagios[15764]: qh: echo service query handler registered
Sep 28 16:08:58 ip-172-31-89-161 nagios[15764]: qh: help for the query handler registered
Sep 28 16:08:58 ip-172-31-89-161 nagios[15764]: wproc: Successfully registered manager as @wproc with query handler
Sep 28 16:08:58 ip-172-31-89-161 nagios[15764]: wproc: Registry request: name=Core Worker 15765;pid=15765
Sep 28 16:08:58 ip-172-31-89-161 nagios[15764]: wproc: Registry request: name=Core Worker 15766;pid=15766
Sep 28 16:08:58 ip-172-31-89-161 nagios[15764]: wproc: Registry request: name=Core Worker 15767;pid=15767
```

Now, create a new EC2 instance on AWS



Now perform the following commands on nagios-host EC2 instance.On the server, run this command

ps -ef | grep nagios

```
ubuntu@ip-172-31-89-161:~$ ps -ef | grep nagios
nagios     15764       1  0 16:08 ?        00:00:00 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
nagios     15765   15764  0 16:08 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios     15766   15764  0 16:08 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios     15767   15764  0 16:08 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios     15768   15764  0 16:08 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios     15769   15764  0 16:08 ?        00:00:00 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
ubuntu     15957    1342  0 16:13 pts/0    00:00:00 grep --color=auto nagios
ubuntu@ip-172-31-89-161:~$
```

Sudo su

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

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

```
ubuntu@ip-172-31-89-161:~$ sudo su
mkdir /usr/local/nagios/etc/objects/monitorhosts
mkdir /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
root@ip-172-31-89-161:/home/ubuntu#
```

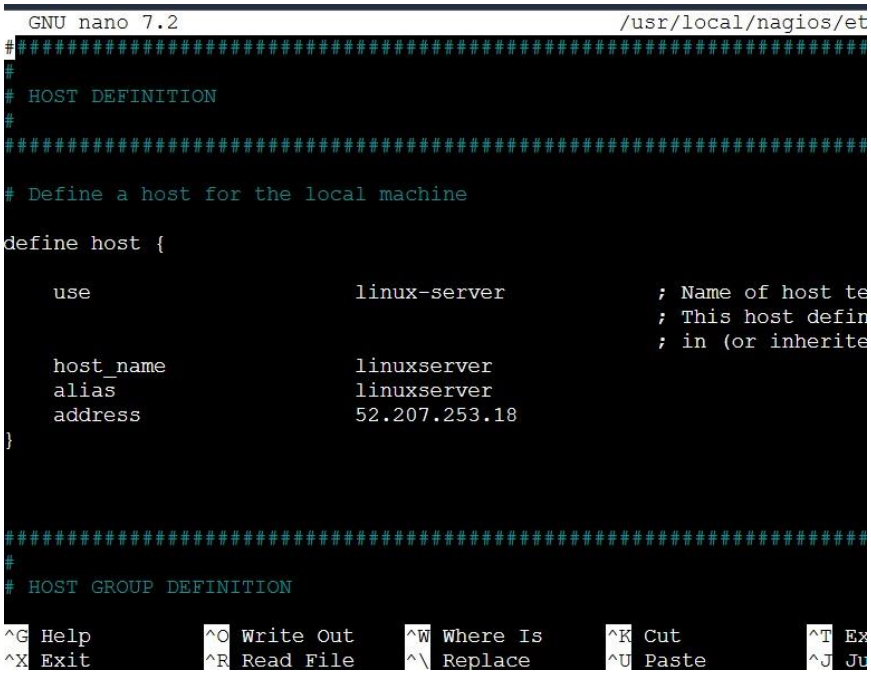
Copy localhost.cfg file to the mentioned location

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

```
root@ip-172-31-89-161:/usr/local/nagios/etc/objects# cp /usr/local/nagios/etc/objects/localhost.cfg /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
cp: cannot create regular file '/usr/local/nagios/etc/objects/monitorhosts/linuxhosts': No such file or directory
root@ip-172-31-89-161:/usr/local/nagios/etc/objects# sudo mkdir -p /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
root@ip-172-31-89-161:/usr/local/nagios/etc/objects# cp /usr/local/nagios/etc/objects/localhost.cfg /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
root@ip-172-31-89-161:/usr/local/nagios/etc/objects#
```

Open the nano editor for localhost.cfg file and make these changes. Add the Ip address of the linux-client for the address field.

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



Note - Here replace hostname with linuxserver

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

Add the following line to the nagios.cfg file

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

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

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


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

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

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

After making the changes in nagios.cfg file now check validate the file by typing the following command in the terminal.

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

```
License: GPL

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

Running pre-flight check on configuration data...

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

Total Warnings: 0
Total Errors:   0

Things look okay - No serious problems were detected during the pre-flight check
root@ip-172-31-89-161:/usr/local/nagios/etc/objects/monitorhosts/linuxhosts#
```

Now restart the service by using this command

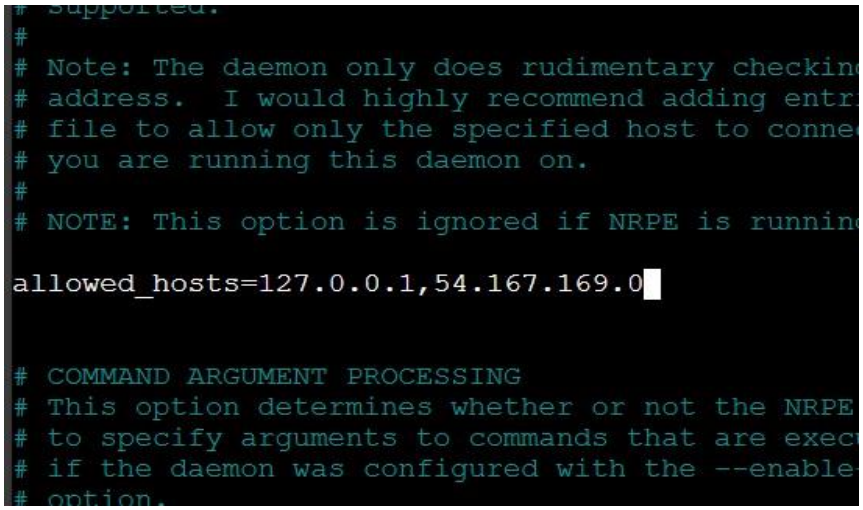
service nagios restart

```
root@ip-172-31-89-161:/usr/local/nagios/etc/objects/monitorhosts/linuxhosts# service nagios restart
root@ip-172-31-89-161:/usr/local/nagios/etc/objects/monitorhosts/linuxhosts# systemctl status nagios
● nagios.service - Nagios Core 4.4.6
     Loaded: loaded (/usr/lib/systemd/system/nagios.service; enabled; preset: enabled)
     Active: active (running) since Sat 2024-09-28 17:36:35 UTC; 19s ago
       Docs: https://www.nagios.org/documentation
    Process: 1870 ExecStartPre=/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
    Process: 1872 ExecStart=/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
   Main PID: 1874 (nagios)
      Tasks: 8 (limit: 1130)
     Memory: 3.0M (peak: 3.2M)
        CPU: 24ms
     CGroup: /system.slice/nagios.service
             ├─1874 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─1875 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─1876 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─1877 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─1878 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─1879 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─1880 /usr/local/nagios/libexec/check_ping -H 52.207.253.18 -w 3000.0,80% -c 5000.0,100% -p 5
             └─1881 /usr/bin/ping -n -U -w 30 -c 5 52.207.253.18

Sep 28 17:36:35 ip-172-31-89-161 nagios[1874]: qh: Socket '/usr/local/nagios/var/rw/nagios.qh' successfully initialized
Sep 28 17:36:35 ip-172-31-89-161 nagios[1874]: qh: core query handler registered
Sep 28 17:36:35 ip-172-31-89-161 nagios[1874]: qh: echo service query handler registered
Sep 28 17:36:35 ip-172-31-89-161 nagios[1874]: qh: help for the query handler registered
Sep 28 17:36:35 ip-172-31-89-161 nagios[1874]: wproc: Successfully registered manager as @wproc with query handler
Sep 28 17:36:35 ip-172-31-89-161 nagios[1874]: wproc: Registry request: name=Core Worker 1875;pid=1875
lines 1-26
```

Now using this command update the apt repository of ubuntu (linux-client), install gcc, nagios-nrpe-server and nagios-plugin sudo apt update -y sudo apt install gcc -y

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

Now open nrpe.cfg file and add the ip address of the nagios host as shown. To open the nrpe.cfg file copy this command.



sudo nano /etc/nagios/nrpe.cfg

Now restart nrpe server by using this command

sudo systemctl restart nagios-nrpe-server

Now, check nagios dashboard, you should see linuxserver up and running, if not

Name : **Himesh Pathai**

Class : **D15A**

Roll No. : **35**

EXPERIMENT - 11

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

Theory:

AWS Lambda

AWS Lambda is a serverless computing service provided by Amazon Web Services (AWS). Users of AWS Lambda create functions, self-contained applications written in one of the supported languages and runtimes, and upload them to AWS Lambda, which executes those functions in an efficient and flexible manner.The Lambda functions can perform any kind of computing task, from serving web pages and processing streams of data to calling APIs and integrating with other AWS services.

The concept of "serverless" computing refers to not needing to maintain your own servers to run these functions. AWS Lambda is a fully managed service that takes care of all the infrastructure for you. And so "serverless" doesn't mean that there are no servers involved: it just means that the servers, the operating systems, the network layer and the rest of the infrastructure have already been taken care of so that you can focus on writing application code.

Features of AWS Lambda

- AWS Lambda easily scales the infrastructure without any additional configuration. It reduces the operational work involved.
- It offers multiple options like AWS S3, CloudWatch, DynamoDB, API Gateway, Kinesis, CodeCommit, and many more to trigger an event.
- You don't need to invest upfront. You pay only for the memory used by the lambda function and minimal cost on the number of requests hence cost-efficient.
- AWS Lambda is secure. It uses AWS IAM to define all the roles and security policies.

Packaging Functions

Lambda functions need to be packaged and sent to AWS. This is usually a process of compressing the function and all its dependencies and uploading it to an S3 bucket. And letting AWS know that you want to use this package when a specific event takes place. To help us with this process we use the Serverless Stack Framework (SST). We'll go over this in detail later on in this guide.

Execution Model

The container (and the resources used by it) that runs our function is managed completely by AWS. It is brought up when an event takes place and is turned off if it is not being used. If additional requests are made while the original event is being served, a new container is brought up to serve a request. This means that if we are undergoing a usage spike, the cloud provider simply creates multiple instances of the container with our function to serve those requests.

This has some interesting implications. Firstly, our functions are effectively stateless. Secondly, each request (or event) is served by a single instance of a Lambda function. This means that you are not going to be handling concurrent requests in your code. AWS brings up a container whenever there is a new request. It does make some optimizations here. It will hang on to the container for a few minutes (5 - 15mins depending on the load) so it can respond to subsequent requests without a cold start.

Stateless Functions

The above execution model makes Lambda functions effectively stateless. This means that every time your Lambda function is triggered by an event it is invoked in a completely new environment. You don't have access to the execution context of the previous event.

However, due to the optimization noted above, the actual Lambda function is invoked only once per container instantiation. Recall that our functions are run inside containers. So when a function is first invoked, all the code in our handler function gets executed and the handler function gets invoked. If the container is still available for subsequent requests, your function will get invoked and not the code around it.

For example, the `createNewDbConnection` method below is called once per container instantiation and not every time the Lambda function is invoked. The `myHandler` function on the other hand is called on every invocation.

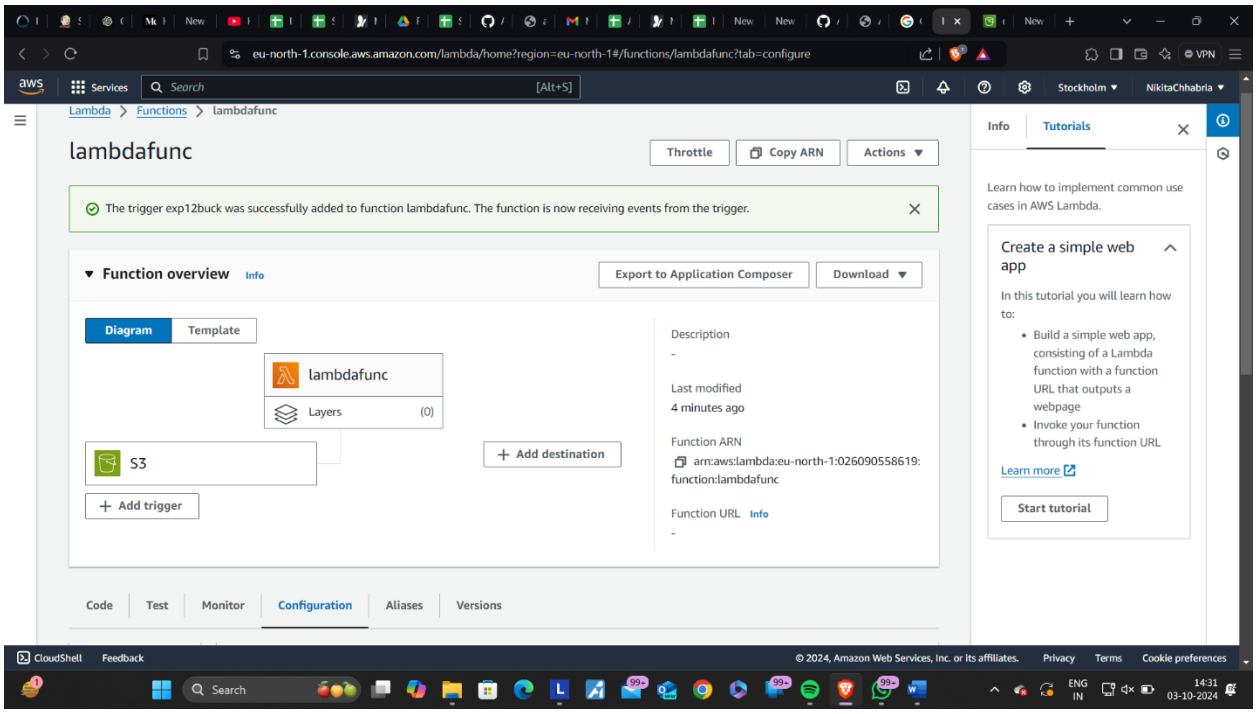
Common Use Cases for Lambda

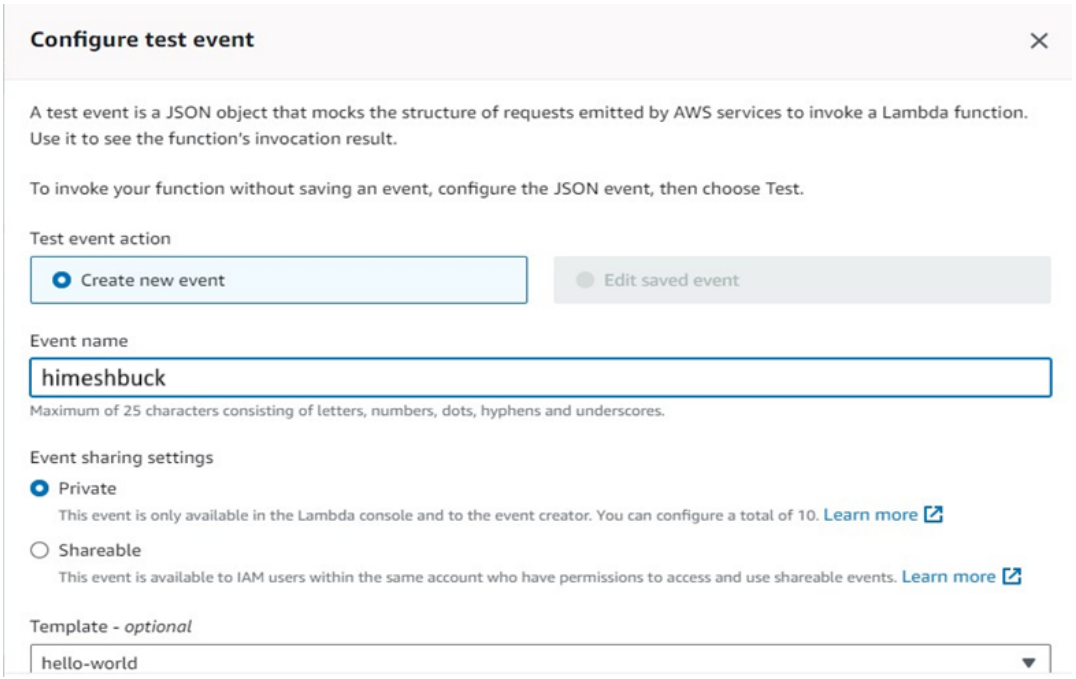
Due to Lambda's architecture, it can deliver great benefits over traditional cloud computing setups for applications where:

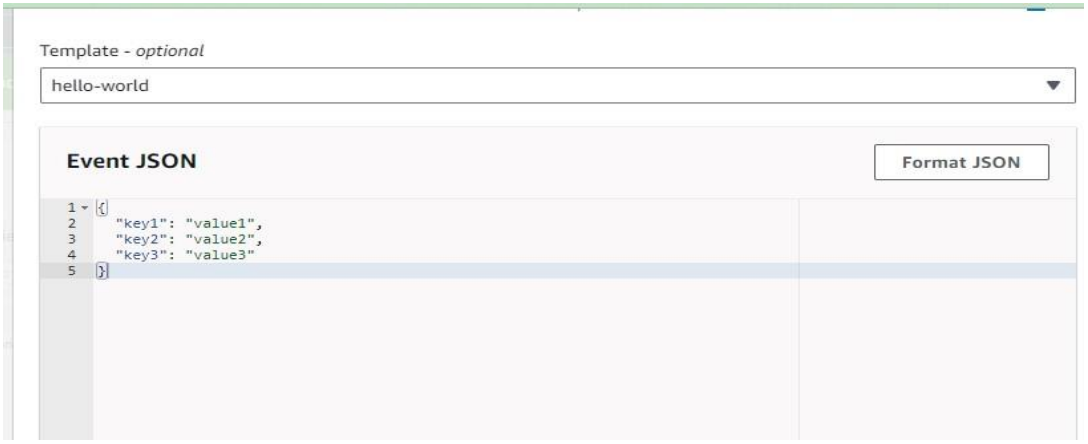
1. Individual tasks run for a short time;
2. Each task is generally self-contained;
3. There is a large difference between the lowest and highest levels in the workload of the application.

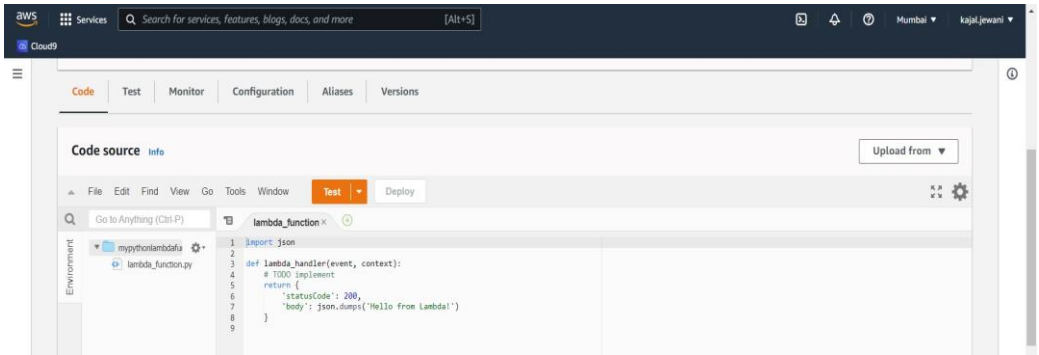
Some of the most common use cases for AWS Lambda that fit these criteria are: Scalable APIs. When building APIs using AWS Lambda, one execution of a Lambda function can serve a single HTTP request. Different parts of the API can be routed to different Lambda functions via Amazon API Gateway. AWS Lambda automatically scales individual functions according to the demand for them, so different parts of your API can scale differently according to current usage levels. This allows for cost-effective and flexible API setups.

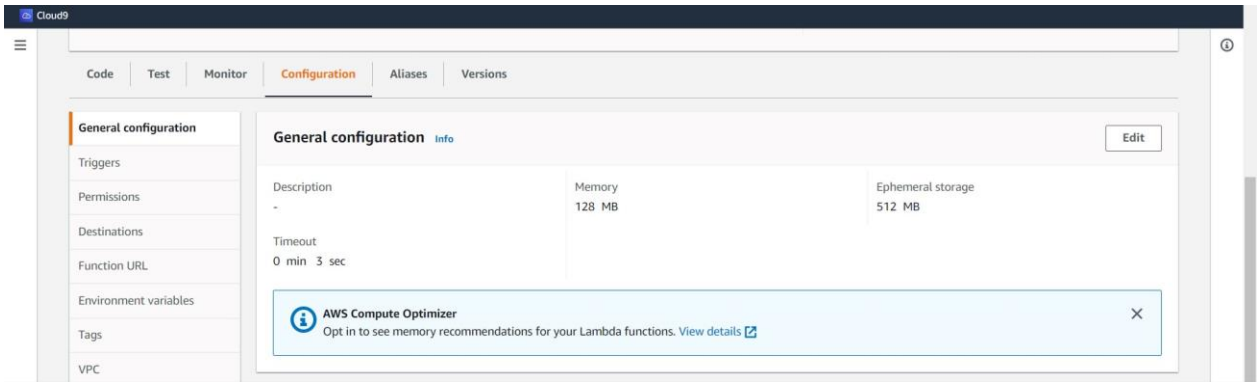
Data processing. Lambda functions are optimized for event-based data processing. It is easy to integrate AWS Lambda with data sources like Amazon DynamoDB and trigger a Lambda function for specific kinds of data events. For example, you could employ Lambda to do some work every time an item in DynamoDB is created or updated, thus making it a good fit for things like notifications, counters and analytics.

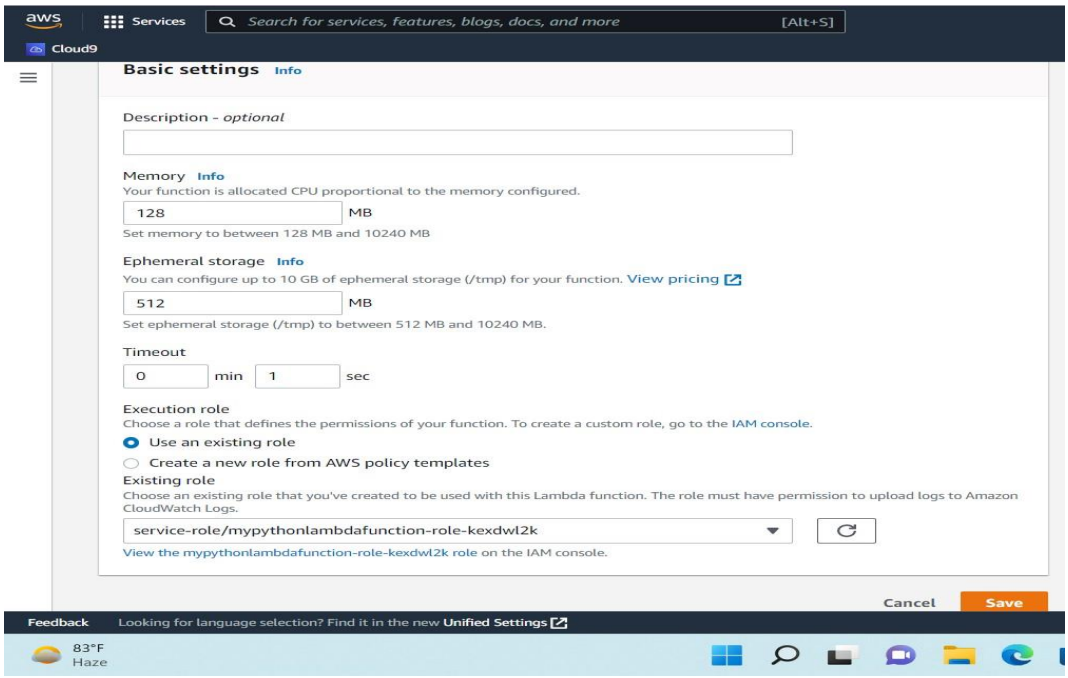


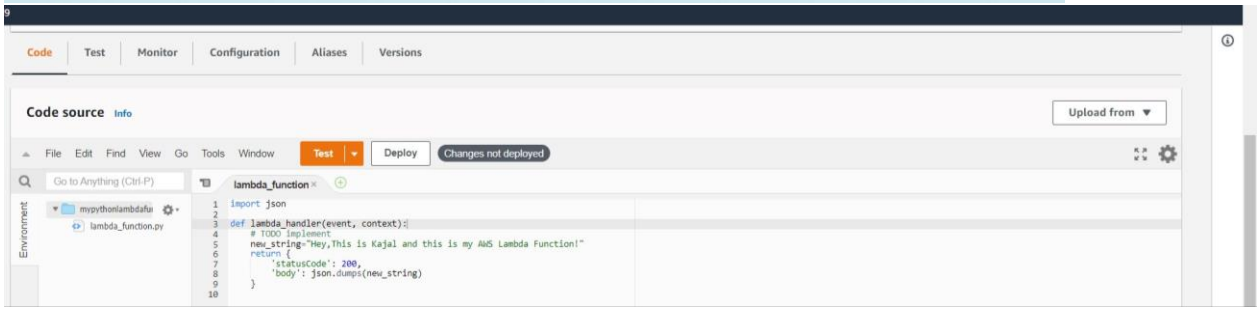

Configure test event
A test event is a JSON object that mocks the structure of requests emitted by AWS services to invoke a Lambda function. Use it to see the function's invocation result.
To invoke your function without saving an event, configure the JSON event, then choose Test.
Test event action
Create new event
Edit saved event
Event name
himeshbuck
Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.
Event sharing settings
Private
This event is only available in the Lambda console and to the event creator. You can configure a total of 10. Learn more
Shareable
This event is available to IAM users within the same account who have permissions to access and use shareable events. Learn more
Template - optional
hello-world

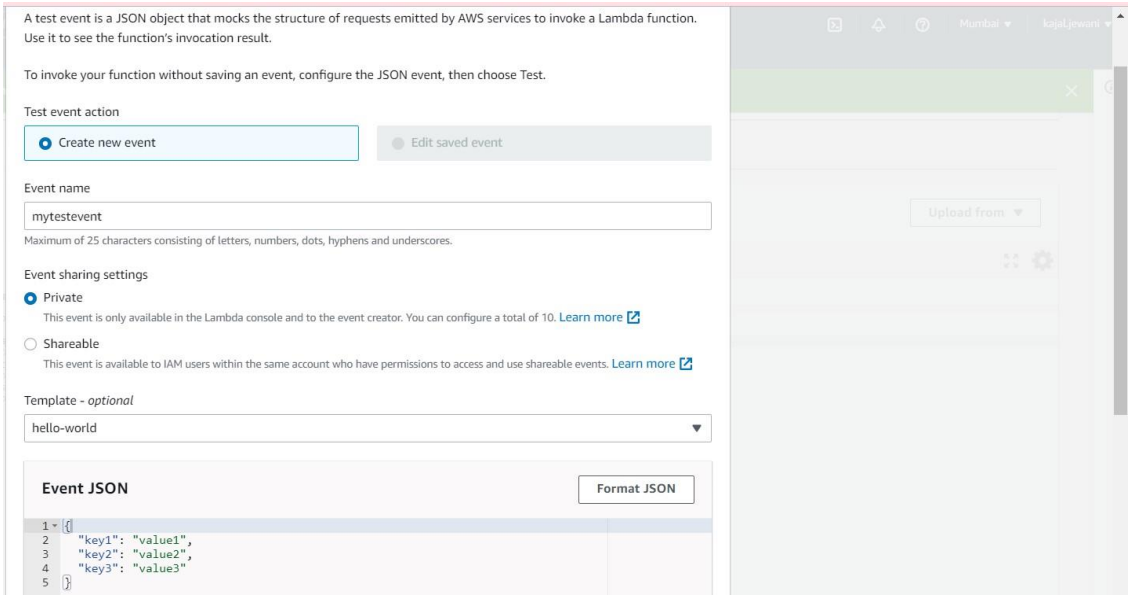

Template - optional
hello-world
Event JSON
Format JSON
{
"key1": "value1",
"key2": "value2",
"key3": "value3"
}

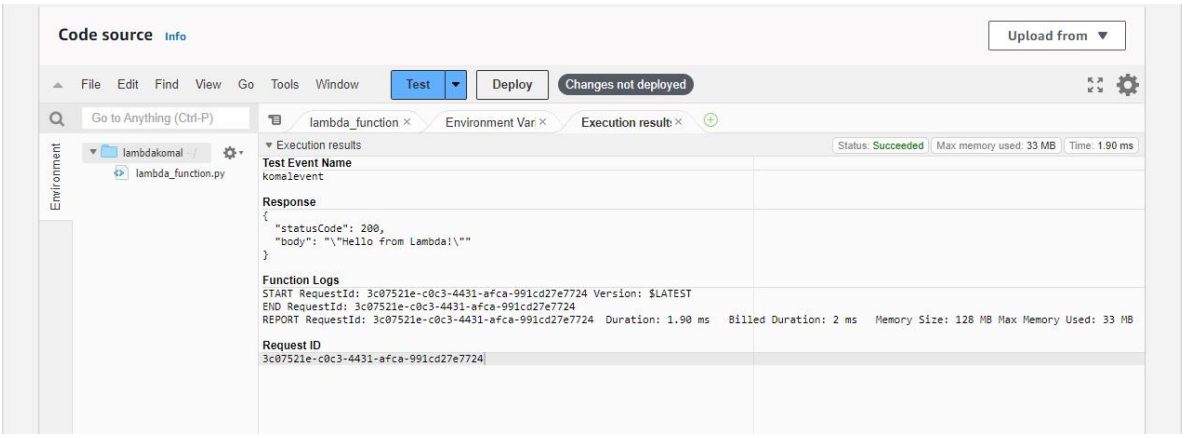

Services
Search for services, features, blogs, docs, and more
[Alt+S]
Mumbai
kajal.jewani
Cloud9
Code
Test
Monitor
Configuration
Aliases
Versions
Code source Info
Upload from
File Edit Find View Go Tools Window
Test
Deploy
Go to Anything (Ctrl-P)
lambda_function
Environment
mypythonlambdafa
lambda_function.py
import json
def lambda_handler(event, context):
# TODO implement
return {
'statusCode': 200,
'body': json.dumps('Hello from Lambda!')
}


Cloud9
Code
Test
Monitor
Configuration
Aliases
Versions
General configuration
Triggers
Permissions
Destinations
Function URL
Environment variables
Tags
VPC
General configuration Info
Edit
Description
-
Memory
128 MB
Ephemeral storage
512 MB
Timeout
0 min 3 sec
AWS Compute Optimizer
Opt in to see memory recommendations for your Lambda functions. View details


aws
Services
Search for services, features, blogs, docs, and more
[Alt+S]
Cloud9
Basic settings Info
Description - optional
Memory Info
Your function is allocated CPU proportional to the memory configured.
128
MB
Set memory to between 128 MB and 10240 MB
Ephemeral storage Info
You can configure up to 10 GB of ephemeral storage (/tmp) for your function. View pricing
512
MB
Set ephemeral storage (/tmp) to between 512 MB and 10240 MB.
Timeout
0
min
1
sec
Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.
Use an existing role
Create a new role from AWS policy templates
Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.
service-role/mypythonlambdafunction-role-kexdwl2k
View the mypythonlambdafunction-role-kexdwl2k role on the IAM console.
Cancel
Save
Feedback
Looking for language selection? Find it in the new Unified Settings
83°F
Haze


Code
Test
Monitor
Configuration
Aliases
Versions
Code source Info
Upload from
File Edit Find View Go Tools Window
Test
Deploy
Changes not deployed
Go to Anything (Ctrl-P)
lambda_function
Environment
lambda_function.py
import json
def lambda_handler(event, context):
# TODO implement
new_string="Hey,This is Kajal and this is my AWS Lambda Function!"
return {
'statusCode': 200,
'body': json.dumps(new_string)
}





Conclusion:

AWS Lambda is a serverless computing service that allows you to run code without managing servers, making it highly scalable, cost-effective, and easy to use. It automatically manages the compute resources, executes your code in response to specific events such as API calls, file uploads, or database updates, and scales based on the demand.

Name : **Himesh Pathai**
Class : **D15A**
Roll No. : **35**

EXPERIMENT NO - 12

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

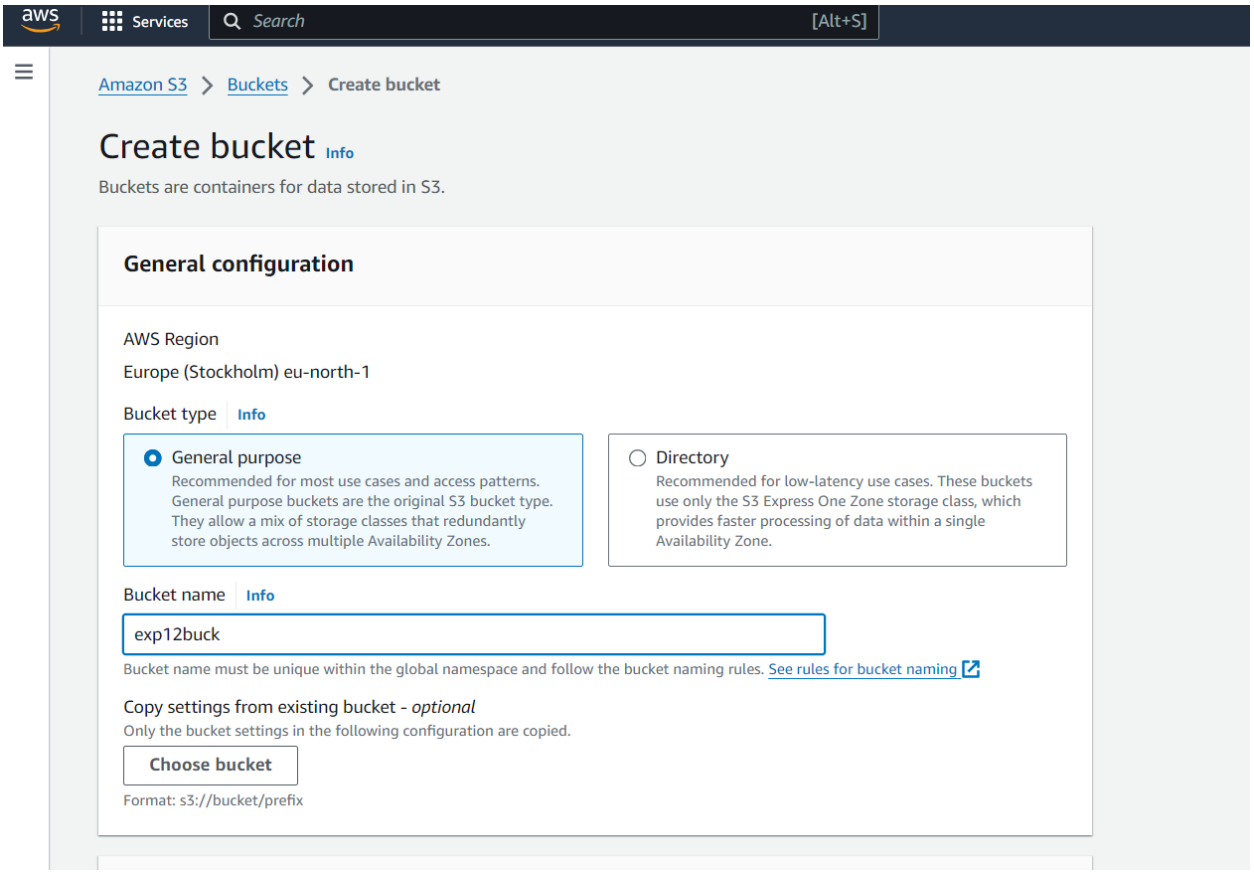
Theory:

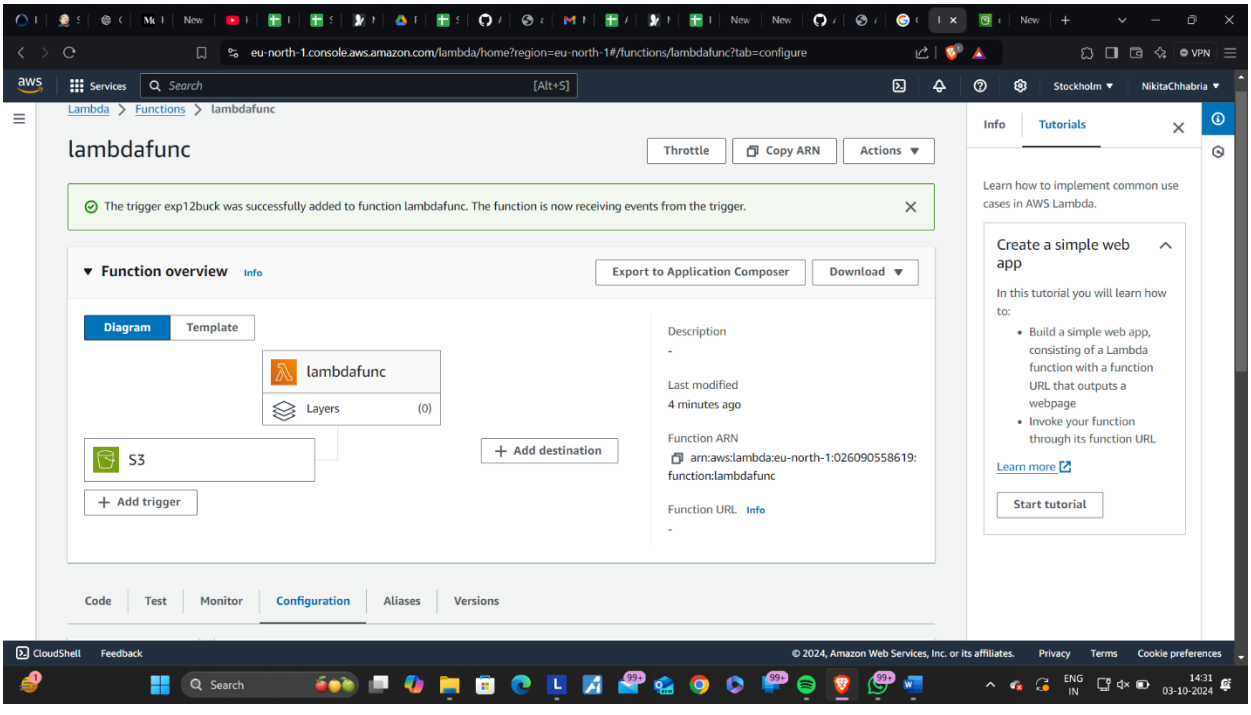
AWS Lambda and S3 Integration: AWS Lambda allows you to execute code in response to various events, including those triggered by Amazon S3. When an object is added to an S3 bucket, it can trigger a Lambda function to execute, allowing for event-driven processing without managing servers.

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.

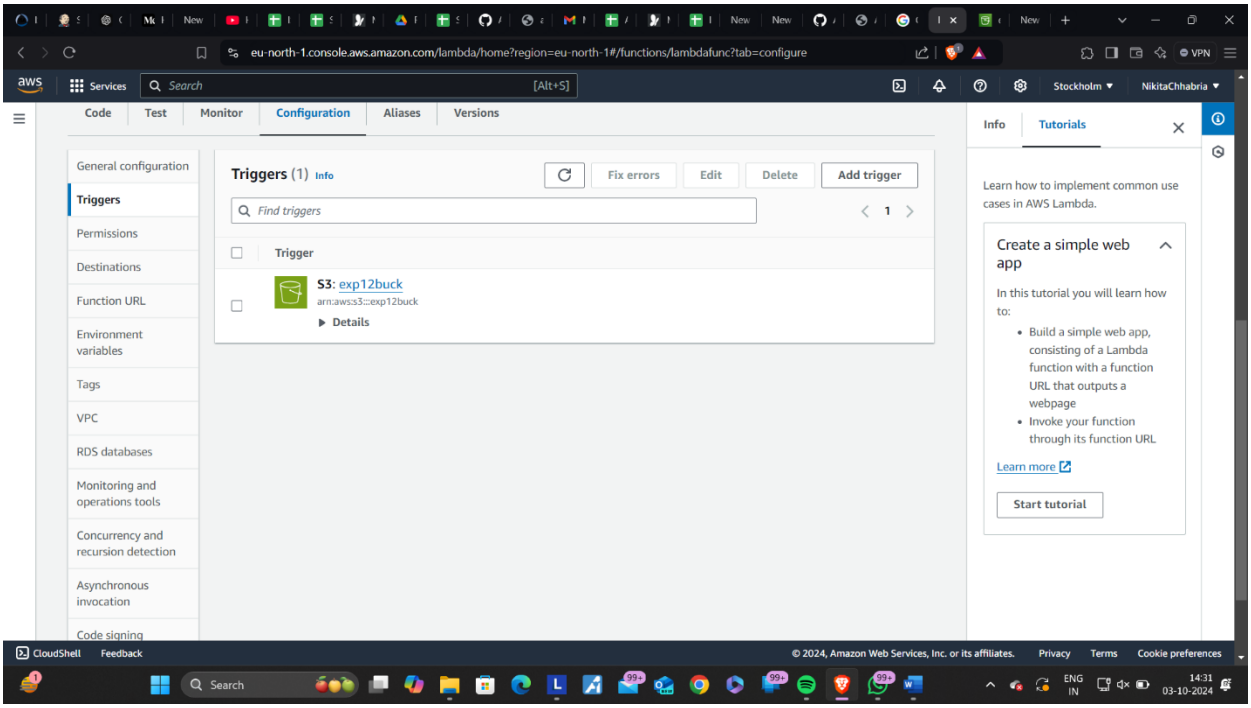
Outcomes:

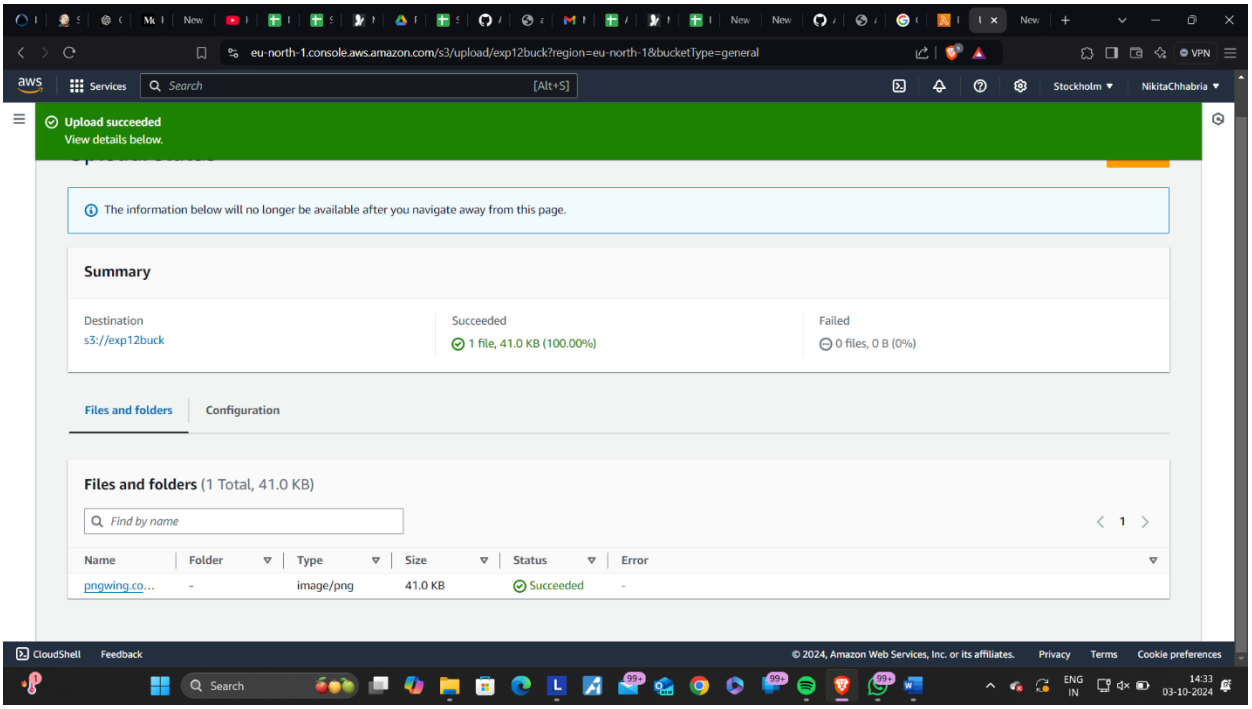

aws
Services
Search
[Alt+S]
Amazon S3 > Buckets > Create bucket
Create bucket Info
Buckets are containers for data stored in S3.
General configuration
AWS Region
Europe (Stockholm) eu-north-1
Bucket type Info
General purpose
Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.
Directory
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.
Bucket name Info
exp12buck
Bucket name must be unique within the global namespace and follow the bucket naming rules. See rules for bucket naming
Copy settings from existing bucket - optional
Only the bucket settings in the following configuration are copied.
Choose bucket
Format: s3://bucket/prefix

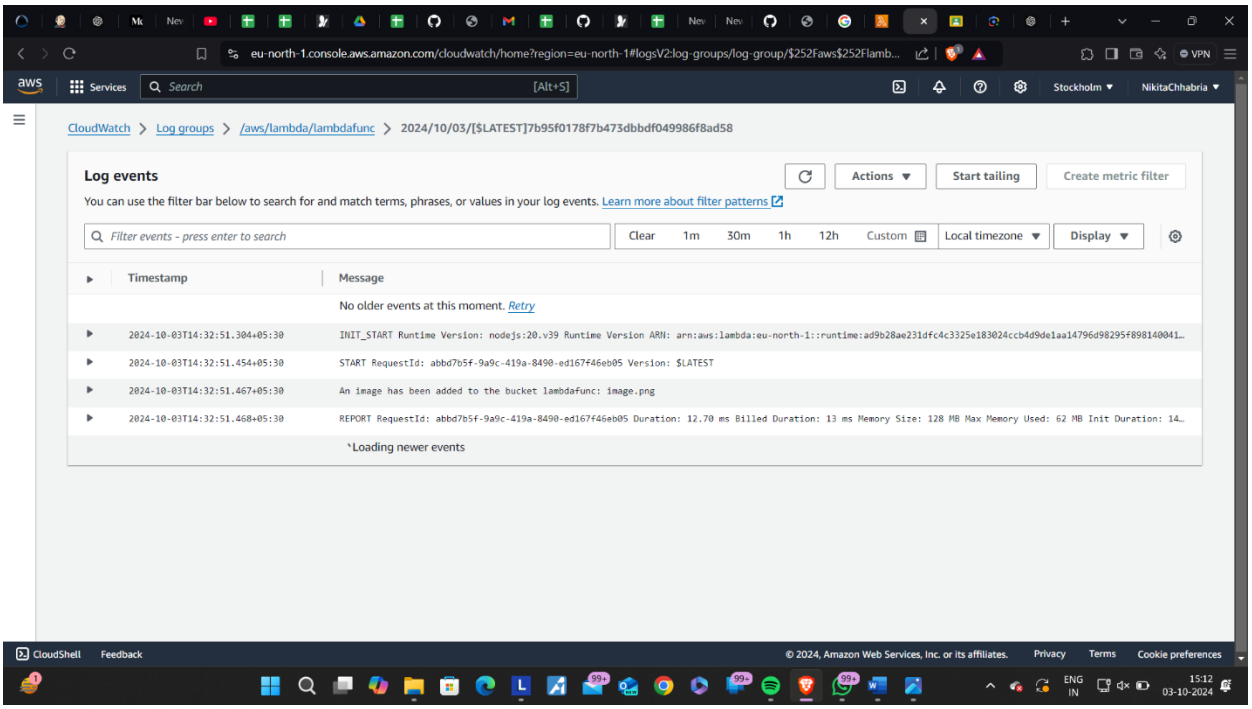

eu-north-1.console.aws.amazon.com/lambda/home?region=eu-north-1#/functions/lambdafunc?tab=configure
Lambda > Functions > lambdafunc
lambdafunc
Throttle
Copy ARN
Actions
The trigger exp12buck was successfully added to function lambdafunc. The function is now receiving events from the trigger.
Function overview Info
Export to Application Composer
Download
Diagram
Template
lambdafunc
Layers (0)
S3
Add trigger
Add destination
Description
-
Last modified
4 minutes ago
Function ARN
arn:aws:lambda:eu-north-1:026090558619:function:lambdafunc
Function URL Info
-
Code
Test
Monitor
Configuration
Aliases
Versions
Info
Tutorials
Learn how to implement common use cases in AWS Lambda.
Create a simple web app
In this tutorial you will learn how to:
Build a simple web app, consisting of a Lambda function with a function URL that outputs a webpage
Invoke your function through its function URL
Learn more
Start tutorial
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

lambda_function × Environment Var ×

```python
import json

def lambda_handler(event, context):
    # Extract bucket name and object key from the event
    bucket_name = event['Records'][0]['s3']['bucket']['name']
    object_key = event['Records'][0]['s3']['object']['key']

    # Log a message
    print(f"An Image has been added to the bucket {bucket_name}: {object_key}")

    return {
        'statusCode': 200,
        'body': json.dumps('Log entry created successfully')
    }
```







Conclusion:

Integrating AWS Lambda with S3 allows for real-time, automated processing of events such as file uploads. In this example, a Lambda function is configured to log a message whenever an image is added to a specific S3 bucket.

Mahesh Pathai
D15A/35

Ad. DevOps Assignment -1

04/03

Q1 Use S3 bucket and host video streaming.

→ To use amazon S3 bucket for video streaming we need to use S3 buckets as a container and cloudfront as a Content Delivery network (CDN).

S1:- Set up Amazon S3 bucket.

1) search for S3 on the services section. Click on it, then click on it. then click on create bucket. This will direct you to the bucket creation page. Here, give a name to you bucket. It is better to block all public access so that unauthorized people do not use the video.

Maintaining the other options as default, click on create bucket.

2) The bucket has been created. Now we need to add our video in the bucket. For that, click on the name of the bucket, this will redirect you to the objects screen which shows the objects of your bucket. Click on Upload.

Then select add files (you can also drag and drop your file). An mp4 extension file is needed as we need to host a video. And the Upload

Step2: Set up Cloudfront

1) As the video is being uploaded. Search for cloudfront on the services tab & open it in a new tab.

2) On the left pane, under security you will find origin access. Click on it, then click on Identities (legacy). Click on Creat origin access identity.

Now go back to Distribution on the left pane & Click on create.

FOR EDUCATIONAL USE

a cloudfront distribution.

3) Here, in the origin field, select the S3 bucket where video is uploaded. Under Origin access, select access identities. Here, Under Origin access identities the identity that you have created. Under bucket Select Yes, Update bucket policy.

Step 3: Accessing the hosted video

1) Once the distribution is deployed, copy the domain of your distribution.
2) Now, go to the S3 bucket and click on its name. Click on the name of the video you have uploaded you will find a key, copy that.
3) Combine the domain name of the distribution and the key of the video to make your final link of the video that is streamed.
<domain name of distribution>/<key of video>

Q2] Discuss BMW and Hot star case studies using AWS.

→ BMW: Overview: BMW uses Amazon Web Services (AWS) to build and scale its connected car platform, providing services like real-time traffic updates, remote vehicle diagnostics. AWS enables BMW to handle petabytes of data requests daily from over 20 million connected vehicles, leveraging tools like Amazon Elastic Kubernetes Service (EKS) for scalability and Amazon S3 for real-time data analytics.

Sundaram
FOR EDUCATIONAL USE

BMW case study using AWS:

Overview:

BMW utilizes Amazon Web Services (AWS) to power its connected car platform, delivering a suite of digital vehicle services such as real-time traffic updates, remote control, & diagnostic capabilities.

Key points:

1 Scalability:

- BMW leverages AWS's data services, such as
- BMW ~~uses~~ Amazon Elastic Kubernetes Service (EKS) to manage and scale its microservices-based architecture which powers the connected car platform.

2 Data Analytics:

- BMW leverages AWS's data services such as Amazon S3 for storage, Amazon EMR for big data processing, and Amazon Athena for querying, to analyze enormous amounts of vehicle & user data.

3 Security:

- Security is paramount in BMW's architecture, especially when dealing with sensitive vehicle and user data. AWS Identity & Access Management (IAM) & AWS Key Management Service (KMS) are employed to securely manage access controls & encryption.

4 Supply chain Transparency (Past Chain Platform):

- To improve traceability & transparency in its supply chain, BMW has built the past chain platform on Amazon EKS & blockchain technology.

FOR EDUCATIONAL USE

5. Cost Optimization:

BMW uses Amazon EC2 Auto scaling to adjust compute capacity based on real-time demand, preventing over provisioning & reducing operational costs.

6. Employee Upskilling & Innovation:

AWS plays a pivotal role in training BMW's workforce. BMW aims to upskill 5,000 engineers & certify 2,000 employees in cloud service like machine learning & data analytics.

~~Thus~~

Through AWS, BMW accelerates the development of smart and autonomous vehicles & enhance digital customer services, aligning with its long-term strategy of connected, efficient, & sustainable automotive solutions.

Hotstar:

Overview: Disney+ Hotstar is a popular Indian subscription streaming service owned by Star India, a subsidiary of The Walt Disney Company India. It offers two plans paid plans: VIP, focusing on domestic content & sports like IPL, & Premium, featuring international content from platforms like HBO & showtime.

Services for Hotstar AWS used are explained:

1. Amazon Route 53

The name itself suggests that at port no. 53, the AWS provides the DNS services to its application. It easily and effectively connects the EC2 instances or Amazon S3 buckets, & it also provides the routing information to the outer side of the infrastructure. This makes AWS more user friendly.

Sundaram
FOR EDUCATIONAL USE

2. Amazon EC2

Provide Scalability & reliability to computing capacity in the AWS cloud which makes less to hardware & more to developing & deploying applications on the cloud.

3. Amazon CloudFront

Low latency & high transfer speed of 5700 Gbps for both can is somehow possible through cloudfront as it provides content Delivery Network Services (CDN).

4. Amazon S3

Storing the data & fetching them as per need is the advantage of the AWS services.

Q3) What is Kubernetes & advantages & disadvantages of Kubernates. Explain How adidas uses Kubernetes.

→ Kubernetes is an open-source container orchestration platform that automate the deployment, scaling, & managment of containerized applications. It is designed to manage large clusters of containers efficiently.

Key features:

1. Automation: Automates deployment, scaling and operations of application containers across clusters of hosts.
2. Portability: Works across on-premise, hybrid or cloud environments, ensuring flexibility in infrastructure choices.
3. Self-Healing: Detects & replaces failed containers, reschedules them automatically.
4. Service Discovery: Facilitates easy discovery of services within a cluster, without hard-coding network settings.

Sundaram
FOR EDUCATIONAL USE

5 Load Balancing: Automatically distributes network traffic across the different containers.

Advantages:

1. Scalability: Easily scales applications horizontally based on real-time demand.
2. Portability: Allows applications to be moved seamlessly between environments (e.g., from testing to production).
3. Resilience: Offers self-healing capabilities, replacing failed containers without downtime.

Disadvantage of Kubernetes:

1. Complexity: Kubernetes has a steep learning curve and can be difficult to set up and manage, especially for beginners.
2. Overhead: Can introduce overhead in terms of computing and operational costs, requiring more resources for orchestration.
3. Security Management: While powerful, Kubernetes requires strong expertise to ensure proper security configurations.

Adidas

In recent years, the adidas team was happy with its software choices from a technology perspective - but accessing all the tools was a problem.

Challenge: In recent years, the adidas team was happy with its software choices from a technology perspective - but accessing all of the tools was a problem.

Solution: To improve the process, "we started from the developer point of view," & looked for ways to shorten the time it took to get a project up

Sundaram
FOR EDUCATIONAL USE

running and into the adidas infrastructure, Sanjay Senivasan Director of platform Engineering Fernando Cornago. They found the soln with containerization, agile develop, continuous delivery, & a cloud native platform that includes Kubernetes & Prometheus.

- Kubernetes, an open-source container orchestration platform, has provided several benefits to Adidas.

→ Scalability & flexibility: Kubernetes enables Adidas to scale its application & service dynamically to meet fluctuating demand. During high-traffic events like product releases or major sports events, the platform automatically allocates additional resources to ensure smooth operations.

→ Faster Development & Deployment: Kubernetes simplifies the deployment & management of containerized application. This allows Adidas to develop & release new features & updates more rapidly.

→ High Availability & Reliability: Kubernetes ensures high availability by automatically distributing workloads across multiple containers & nodes. This means Adidas can provide uninterrupted online shopping experiences to customers worldwide.

→ Resource Efficiency: With Kubernetes, Adidas can optimize resource utilization. Containers share the underlying infrastructure efficiently, reducing waste & resource

Sundaram
FOR EDUCATIONAL USE

→ CI CD : Kubernetes integrates seamlessly with CI/CD
Adidas can automate the testing & deployment of co
ensuring that new features & update are delivered to
quickly & reliably.

Impact : just six months after the project began
of the adidas e-commerce site was running on Kub
load time for the e-commerce site was reduced
Releases went from every 4-6 weeks to 3-4 time

Conclusion : Kubernetes is a powerful & flexible platf
for container orchestration that can simplify the d
, managment, & scaling of your applications while
enhancing their reliability & resource efficiency.
It's extensive ecosystem & active community make it a
compelling choices for modern application deployme
managment.

Q 4) what are Nagios & explain how Nagios a
in IT-services?

→ Nagios is an open-source monitoring system that en
organizations to monitor their IT infrastructure, includ
servers, network devices, & applications. It helps ensu
systems are running smoothly by providing real-tim
monitoring, alerting & reporting capabilities.

* How Nagios is used in IT-Services

1. Infrastructure Monitoring: Nagios continuously mon
servers, databases & network devices in an en
enviroment to ensure they are operational. This inc
checking the availability of web servers, database

Sundaram
FOR EDUCATIONAL USE

and other critical components.

2. Performance Monitoring: It tracks various performance metrics such as CPU load, memory usage, disk space, & network traffic.

3. Alerting & Notifications: Nagios can send alerts via email, SMS, or other communication methods when it detects issues.

4. Service Monitoring: It monitors specific e-services such as web applications, APIs, and email servers to ensure they are functioning correctly.

5. Log Monitoring - Nagios can analyze log files for error messages or anomalies, providing insights into potential issues within the e-services.

6. Integration with Other tools: Nagios can be integrated with other monitoring and managment tools, allowing for comprehensive oversight of the IT environment and enabling more sophisticated alerting & reporting mechanisms.

7. Customizable Dashboards: It provides dashboards that give a visual overview of the monitored infrastructure, helping teams to quickly assess the health of their e-services.

→ By using Nagios, e-service provider can ensure high availability & performance of their applications, enhancing user satisfaction & trust in their services.

Sundaram
FOR EDUCATIONAL USE

Himesh Pathak
D15A/35

05/03

Advance DevOps Assignment 2 DATE:

Create a REST API with serverless framework.
Creating REST API with serverless framework is an efficient way to deploy serverless applications that can scale automatically without managing server i) Serverless framework: A powerful tool that deployment of services and serverless applications across various cloud periods such as AWS, AZWR and Google cloud.
ii) Serverless architecture: This design model allows developers to build applications without worrying about underlying infrastructure, enabling focus on code & business logic.
iii) REST API: Representational State Transfer is architecture style for designing network applications.

Steps for creating RESTAPI for serverless framework
1) Install serverless framework:
You start by installing serverless framework CLI Globally using node package manager (npm) This allow you to manage serverless applications directly from your terminal.
2) Creating a Node js serverless project:
A directory is created for your project, where you will initialize a serverless service (project). This service will house all your lamba functions configurations and cloud resources. Using the command serverless create you set up a template for AWS Node js microservices that will eventually deploy to AWS lambda.

FOR EDUCATIONAL USE

DATE:

3) Project Structure:-
The project scaffold creates essential files like handler.js (which contains code for lambda functions and serverless.yml.

4) Create a REST API Resource:
In the serverless.yml file you define function that handles post requests of HTTP.

5) Deploy the service:
with the 'sls deploy' command serverless framework packages your applications, uploads necessary resources to AWS and set up the infrastructure.

6) Testing the API: Once deployed you can test REST API using tools like curl or Postman by making post request to generated API.

7) Storing data in Dynamodb: To store submitted candidate data you integrate AWS DynamoDB as a database.

8) Adding more functionalities: Adding functionalities like 'list all candidates, get candidates by ID'.

9) AWS IAM Permissions:
you need to ensure that serverless framework is given right permissions to interact with AWS resources like DynamoDB.

10) Monitoring and Maintenance
After deployment serverless framework provides service information like deployed endpoints, API key, log streams.

Q.2 Case study for SonarQube
Creating your own profile in Sonarqube for testing

FOR EDUCATIONAL USE

DATE:

Project quality. Use Sonarqube to analyze your github code. Install sonarlint in your Sonarqube to analyse Java in Eclipse ide and analyse java code. Analyze Python project with sonarqube.

→ Sonarqube is an open source platform used for continuous inspection of quality. It detects bugs, code smells and security valenerabilities in project across various programming languages.

1) Profile Creation in SonarQube:

Quality profiles in sonarqube are essential configurations that define rules applied during code analysis. Each project has a quality profile for every supported language with default being 'Sonar way' profiles comes built in for all languages. Custom profiles can be created by copying or extending existing ones. Copying creates an independent profile, while extending inherit rules from parent profile & reflects future changes automatically. You can activate or deactivate rules, prioritize certain rules and configure parameter to tailor profile to specific projects. Permissions to manage quality profile are restricted to users with administrative privileges. Sonarqube allows for the comparion of two profiles to check for differences in activated rules and users can track changes via event log. Quality profiles to specific projects. Permissions to can also be imported from other instances via backup and restore.

FOR EDUCATIONAL USE

DATE:

To ensure profiles include new rules its important to check against updated built in profiles or use sonarqube rules page.

2) Using Sonarcloud to analyze GitHub Code:
Sonarcloud is cloud-based counterpart of Sonar Qube that integrates directly with GitHub, BitBucket, Azure and Github repositions. To get started with sonarcloud via Github Signup via Sonarcloud product page and connect your Github organizations or personal account. Once connect, sonarcloud mirrors your github setup with each project corresponding to Github repository. After setting up the organization choose subscription plan (free for public repo). Next import repositories into your sonarcloud organization where each Github repo becomes a Sonarcloud project. Define 'new code' to focus on recent changes and choose between automatic analysis or CI-based analysis. Automatic analysis happens directly in Sonarcloud, while CI based analysis integrates with your build process once the analysis is complete results can be viewed in both Sonarcloud and Github including security import issue.

3) Sonarlint in Java IDE:
Sonarlint is an IDE that performs on-the-fly code analysis as you write code. It helps developers in the development environment

FOR EDUCATIONAL USE

DATE:

such as Intellic idea or Eclipse. To set it up install the sonarlint plugin, configure the connection with sonarQube or sonarcloud and select the project profile to analyse Java code. This approach ensures immediate feedback on code quality, promoting clean & maintainable code from begining.

4) Analyzing Python Projects with SonarQube:
SonarQube supports Python test coverage reporting but it requires third party tool like coverage. Py to generate the coverage port To enable coverage adjust your build process so that coverage tools runs before sonar scanner and ensures report file is saved in diff. path.
For set up, you can use Tox, PyTest, and coverage. Py to configure and run test. In your tox.ini include configurations for Py test and coverage to generate coverage report in XML format. The build process can also be automated using Github Actions, which install dependancies runs tests and invokes SonarQube scan. Ensure report in cobertura XML format and place where scanner can access it

5) Analyzing Node.js Projects with SonarQube
For Node.js Project SonarQube can analyse Java script and Typescript code. Similar to the Python setup, you can configure SonarQube to analyse node.js projects by installing the appropriate plugin and using Sonar scanner to

FOR EDUCATIONAL USE

DATE:

scan the Projects. SonarQube will check the code against Industry standard rules and best Practices, flagging issues related to security vulnerabilities bugs and performances optimization.

3. At a large organization, your centralized operation team may get many repetitive infrastructure requests you can use Terra-Form to build a self service infrastructure modes that lets product team manage their own infrastructure independently. You can creati and use Terraform modules that codify the standards for deploying & managing services in your organizations allowing teams to efficiently deploy services in compliance with your organizations practices. Terraform cloud can also integrate with ticketing system like service now to automatically generate new infrastructure requests.

ins Implementing a 'self-serve' infrastructure model using Terraform can transform how large organization manage their infrastructure independtly. Organization can enhance efficiency. Reduce bottlenecks and ensure compliance with established needs.

- The need for self-service infrastructure:
  In large organization, centralized operations teams often face an overwhelming number of

FOR EDUCATIONAL USE

DATE:

Repetitive requests. This can lead to delay in service delivery and frustration among product teams who need to move quickly. A self-service model allows teams to provision and manage their infrastructure without relying on the operations team for every request.

- Benefits of using Terraform.

1. Modularity & Reusability:
- Terraform modules encapsulate standard configurations for various infrastructure components (eg networks, databases, compute resources).
- Teams can reuse these modules across different projects, reducing redundancy and minimizing the risk of errors.

2. Standardizations
- By defining best practices within modules, organizations can ensure that all deployments comply with internal policies and standards.
- This consistency helps maintain security and operational integrity across the organization.

3. Increased Efficiency:
- Product teams can deploy services quickly by using pre-defined modules, significantly reducing the time spent on infrastructure setup.
- This allows team to focus on developing features rather than managing infrastructure.

4. Integration with ticketing systems
- Terraform cloud can integrate with ticketing systems like service now.

FOR EDUCATIONAL USE

DATE:

to automate the generation of infrastructure requests.
- This integration streamlines workflows by allowing teams to initiate requests directly from their ticketing platform, reducing manual intervention.

⇒ Implementation steps

1. Identify Infrastructure components
- Begin by identifying which components of your infrastructure can be modularized (eg VPCs, security groups, load balancers)

2. Develop Terraform modules.
- Create reusable modules that define the desired configuration and resources.
- Ensure each module includes input variables for customizations and outputs for integration with other modules.

3. Establish Governance and Best Practices:
- Define guidelines for module usage, versioning and documentation to ensure clarity and maintainability.
- Encourage teams to contribute to module development and share improvements.

4. Testing and Validation.
- Implement a testing framework to validate module functionality before development.

- Best practices for module management.
- utilize the terraform registry.
- leverage existing community modules from the

FOR EDUCATIONAL USE

DATE:

Terraform Registry to avoid reinventing solutions and ensure adherence to best practices.

- Version control: Implement versioning for your modules to track changes over time. This helps manage dependencies effectively and minimize disruptions during updates.
- Documentation: Maintain comprehensive documentation for each module including usage examples, input/output descriptions and any dependency
- Encourage collaboration: Foster a culture of collaboration by sharing modules across teams. This promotes consistency in deployments and facilitates knowledge within the organization

By adopting a self-service infrastructure model with Terraform organizations can empower product teams to efficiently manages their own infrastructure while ensuring compliance with established, standard. This approach not only streamlines processes but also enhances agility in responding to changing business needs. Ultimately, it leads to a more responsive IT environment that supports innovation and growth within the organization.

FOR EDUCATIONAL USE