Name: Nishant Khetal **Div: D15C** **Roll No: 24**

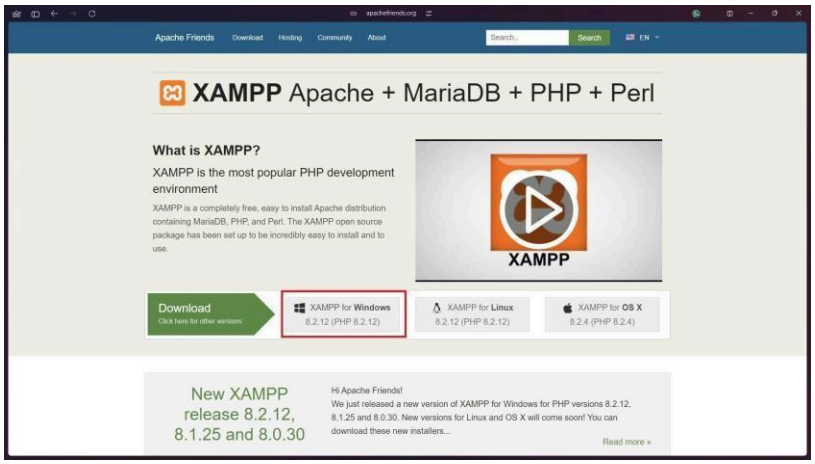
Experiment No. 1A

Static Hosting

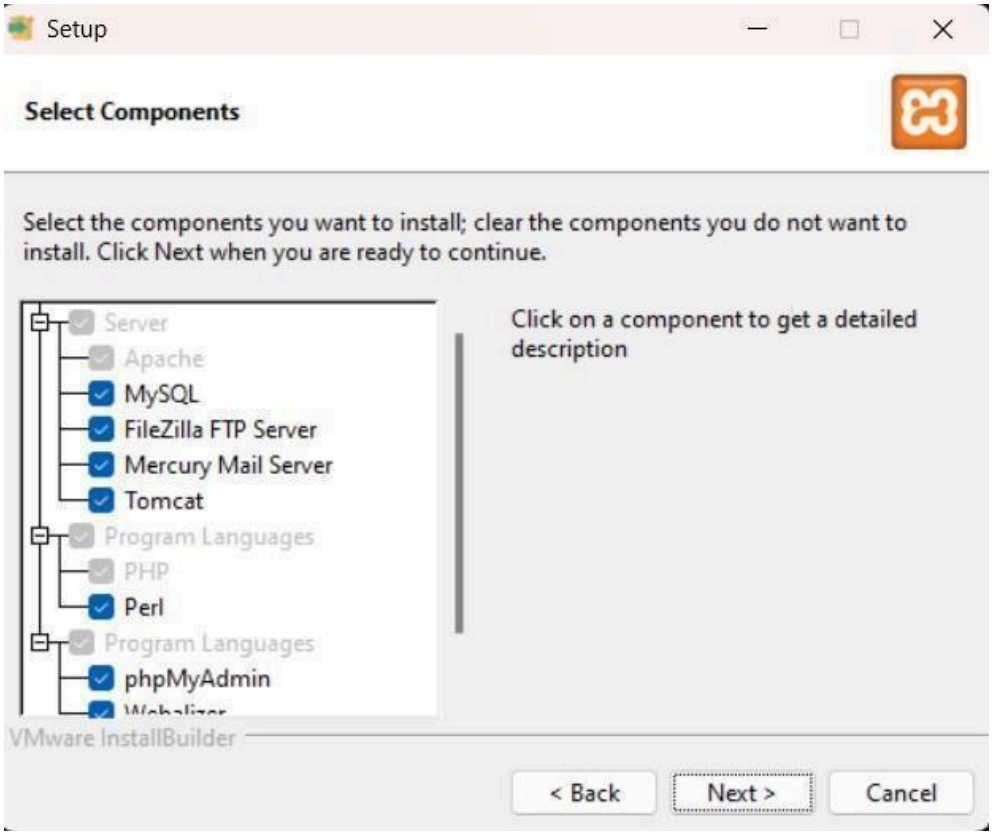
On local server (XAMPP)

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

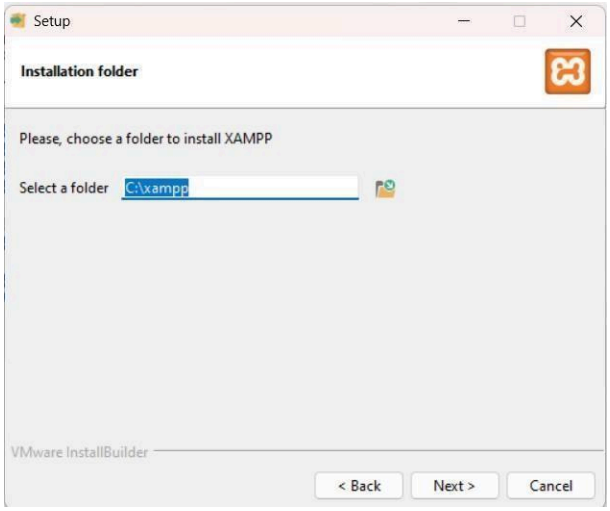
1) Pick your OS, and the download will start automatically.



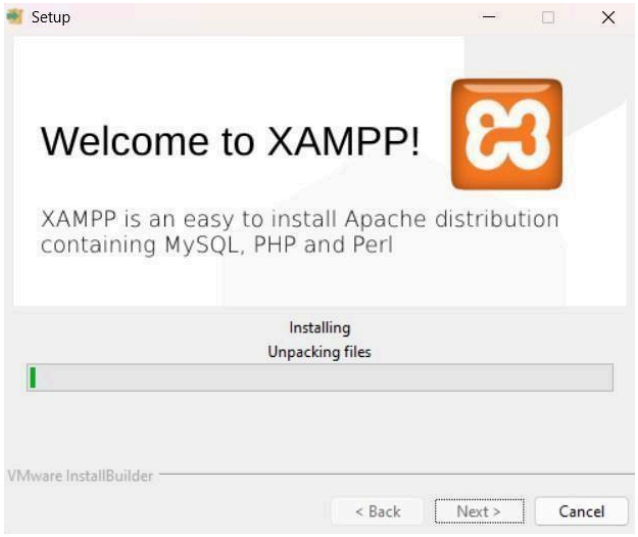
2) Open the setup file, select the required components, and click 'Next'.



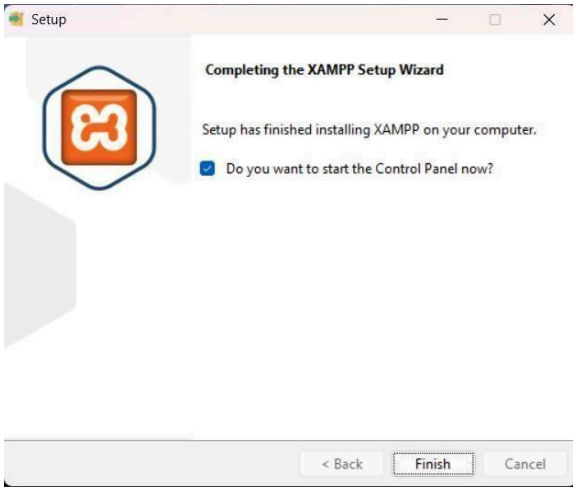
3) Choose an empty folder for XAMPP installation and click 'Next'.



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



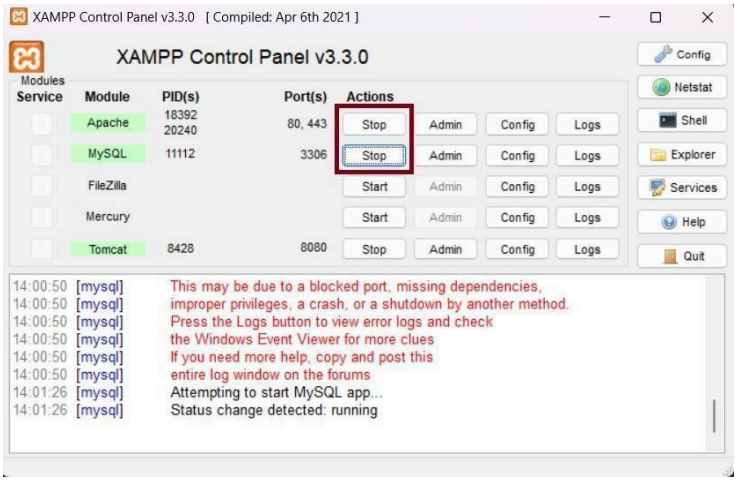
5) The installation is complete; click 'Finish'.



Step 2: Create a file to be hosted on the server, ensuring it has a `.php` extension.

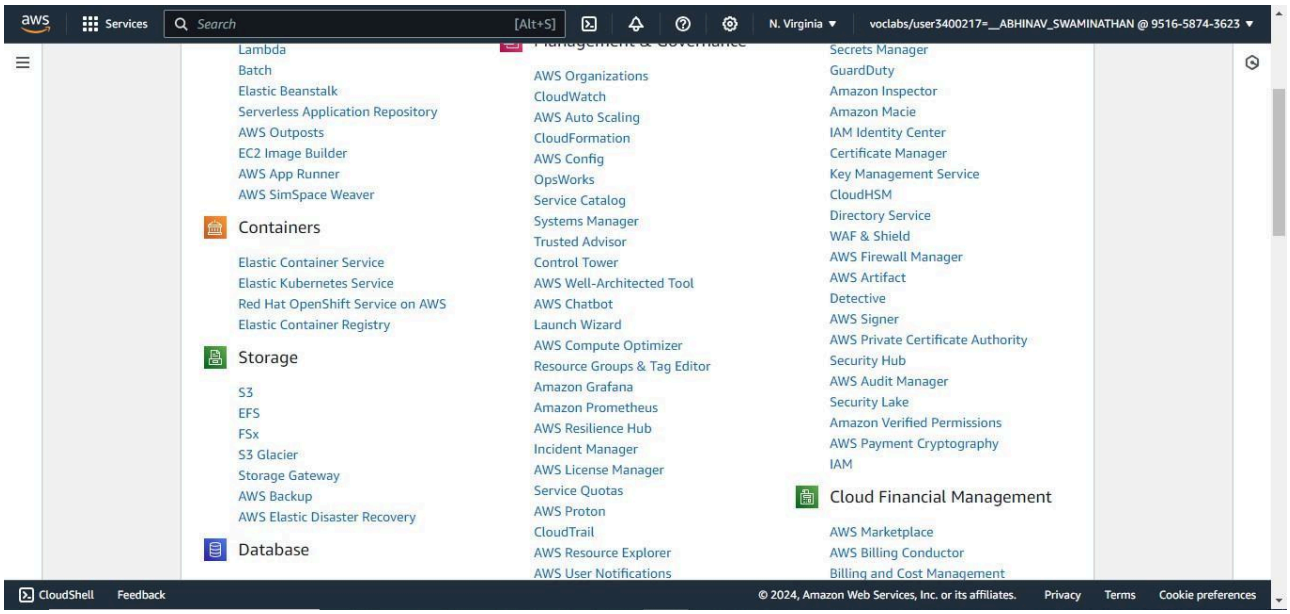
Step 3: Navigate to the directory where XAMPP is installed and open the `htdocs` folder. Place your project folder within this directory.

Step 4: Open the XAMPP Control Panel and start the Apache service (required) and the MySQL service (if needed)



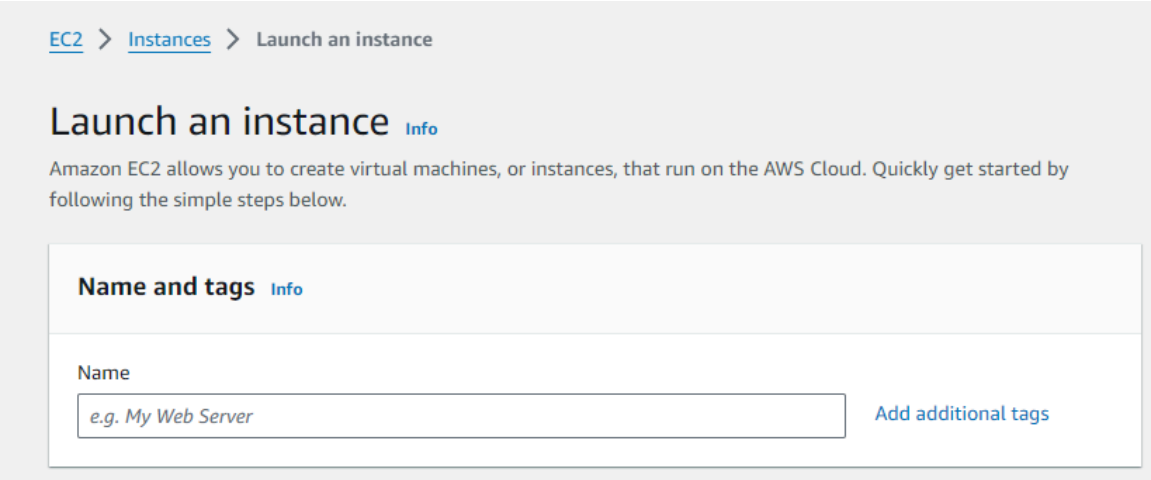
Step 5: Open your web browser. Type localhost/YOUR_FILENAME.php. This will open your website on your browser.

AWS EC2 Instances

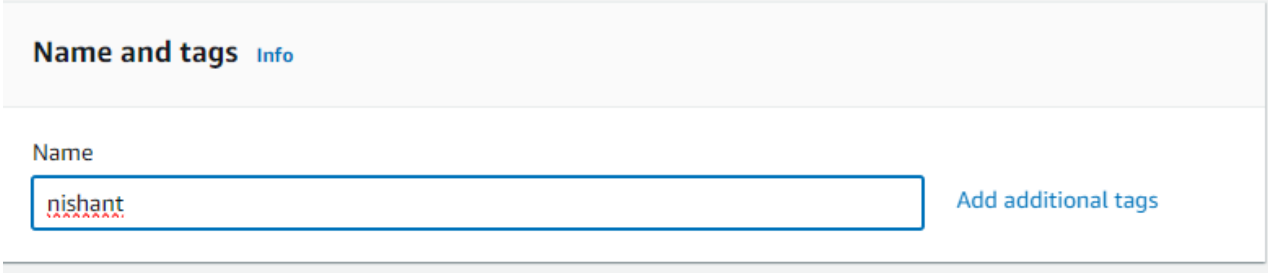


Step 1: Log in to your AWS account. Go to services and open EC2.

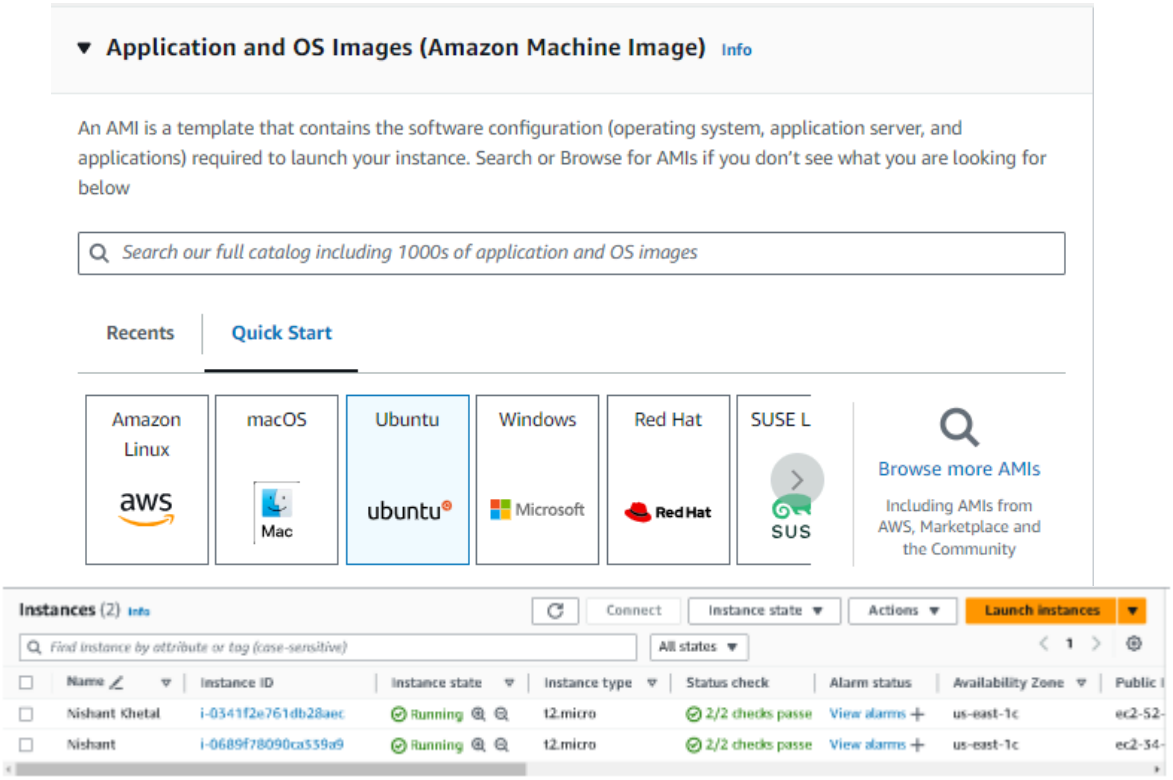
Step 2: Click on Launch an Instance:



Step 3: Assign a name to your EC2 instance, keeping other options default. Scroll down and click on 'Launch Instance'.



Step 4: Click on the name of your EC2 instance and go to the 'Details' section



Step 5: Create a Connection.

Establishing Connection ...

```
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1009-aws x86_64)

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

 System information as of Sun Aug  4 05:22:27 UTC 2024

  System load:  0.08              Processes:             103
  Usage of /:   22.9% of 6.71GB   Users logged in:       0
  Memory usage: 19%               IPv4 address for enX0: 172.31.32.106
  Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

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


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


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

Step 6: Run the commands:

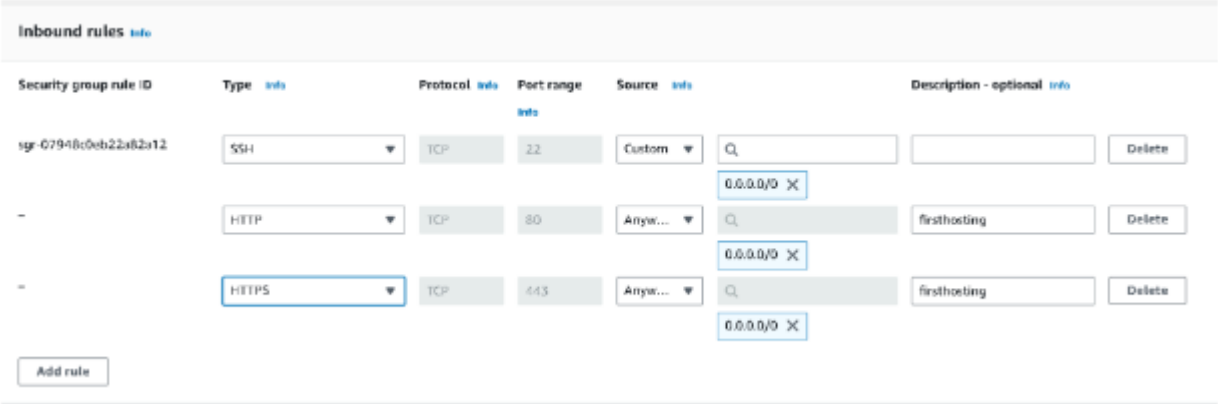
```
root@ip-172-31-32-106:~/temp# wget https://www.free-css.com/assets/files/free-css-templates/download/page296/neogym.zip
--2024-08-04 05:39:23--  https://www.free-css.com/assets/files/free-css-templates/download/page296/neogym.zip
Resolving www.free-css.com (www.free-css.com)... 217.160.0.242, 2001:8d8:100f:f000::28f
Connecting to www.free-css.com (www.free-css.com)|217.160.0.242|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 724954 (708K) [application/zip]
Saving to: 'neogym.zip'

neogym.zip                    0%[                                                                    ]       0  --.-KB/s          ne
ogym.zip                      9%[======>                                                             ]  63.73K   294KB/s          neog
ym.zip                       33%[==========================>                                         ] 239.73K   563KB/s          neogym
.zip                        100%[===================================================================>] 707.96K  1.17MB/s    in 0.6s

2024-08-04 05:39:24 (1.17 MB/s) - 'neogym.zip' saved [724954/724954]

root@ip-172-31-32-106:~/temp# █
```

Step 7: Upload the zip file of website:


```
root@ip-172-31-32-106:~/temp# wget https://www.free-css.com/assets/files/free-css-templates/download/page296/neogym.zip
--2024-08-04 05:39:23--  https://www.free-css.com/assets/files/free-css-templates/download/page296/neogym.zip
Resolving www.free-css.com (www.free-css.com)... 217.160.0.242, 2001:8d8:100f:f000::28f
Connecting to www.free-css.com (www.free-css.com)|217.160.0.242|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 724954 (708K) [application/zip]
Saving to: 'neogym.zip'

neogym.zip                    0%[                                                                    ]       0  --.-KB/s          ne
ogym.zip                      9%[======>                                                             ]  63.73K   294KB/s          neog
ym.zip                       33%[==========================>                                         ] 239.73K   563KB/s          neogym
.zip                        100%[===================================================================>] 707.96K  1.17MB/s    in 0.6s

2024-08-04 05:39:24 (1.17 MB/s) - 'neogym.zip' saved [724954/724954]

root@ip-172-31-32-106:~/temp# █
```


```
root@ip-172-31-32-106:~/temp# cd neogym-html;
root@ip-172-31-32-106:~/temp/neogym-html# ls -lrt
total 56
-rw-r--r-- 1 root root  6738 Sep 15  2020 why.html
-rw-r--r-- 1 root root  6842 Sep 15  2020 trainer.html
-rw-r--r-- 1 root root 17875 Sep 15  2020 index.html
-rw-r--r-- 1 root root  5649 Sep 15  2020 contact.html
drwxr-xr-x 2 root root  4096 Sep 15  2020 js
drwxr-xr-x 2 root root  4096 Sep 15  2020 images
drwxr-xr-x 2 root root  4096 Sep 15  2020 css
root@ip-172-31-32-106:~/temp/neogym-html# mv * /var/www/html/
root@ip-172-31-32-106:~/temp/neogym-html# cd /var/www/html/
root@ip-172-31-32-106:/var/www/html# ls -lrt
total 56
-rw-r--r-- 1 root root  6738 Sep 15  2020 why.html
-rw-r--r-- 1 root root  6842 Sep 15  2020 trainer.html
-rw-r--r-- 1 root root 17875 Sep 15  2020 index.html
-rw-r--r-- 1 root root  5649 Sep 15  2020 contact.html
drwxr-xr-x 2 root root  4096 Sep 15  2020 js
drwxr-xr-x 2 root root  4096 Sep 15  2020 images
drwxr-xr-x 2 root root  4096 Sep 15  2020 css
root@ip-172-31-32-106:/var/www/html# █
```

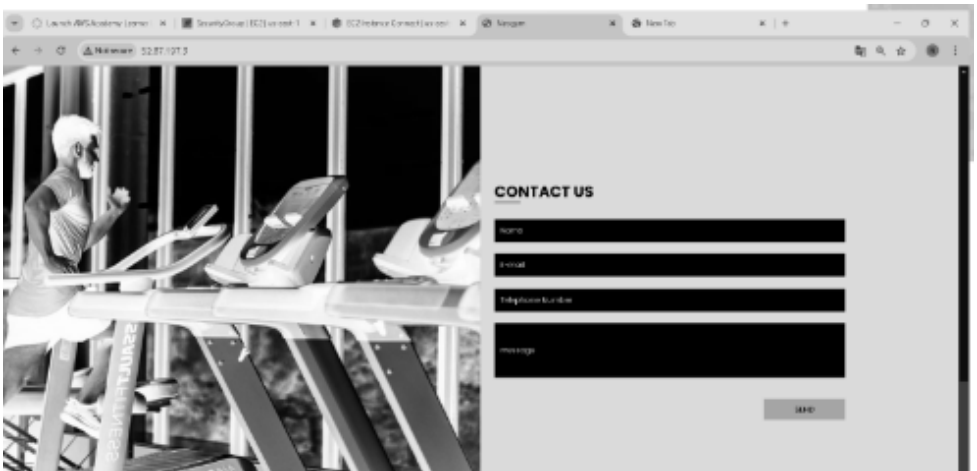
Step 8 : Provide all the required Inbound rules:



Step 9: Start the Apache Server:

```
root@ip-172-31-32-106:/var/www/html# sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/apache2.service; enabled; preset: enabled)
     Active: active (running) since Sun 2024-08-04 05:30:27 UTC; 23min ago
       Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 14033 (apache2)
      Tasks: 55 (limit: 1130)
     Memory: 7.2M (peak: 7.4M)
        CPU: 137ms
     CGroup: /system.slice/apache2.service
             ├─14033 /usr/sbin/apache2 -k start
             ├─14036 /usr/sbin/apache2 -k start
             └─14037 /usr/sbin/apache2 -k start

Aug 04 05:30:27 ip-172-31-32-106 systemd[1]: Starting apache2.service - The Apache HTTP Server...
Aug 04 05:30:27 ip-172-31-32-106 systemd[1]: Started apache2.service - The Apache HTTP Server.
```

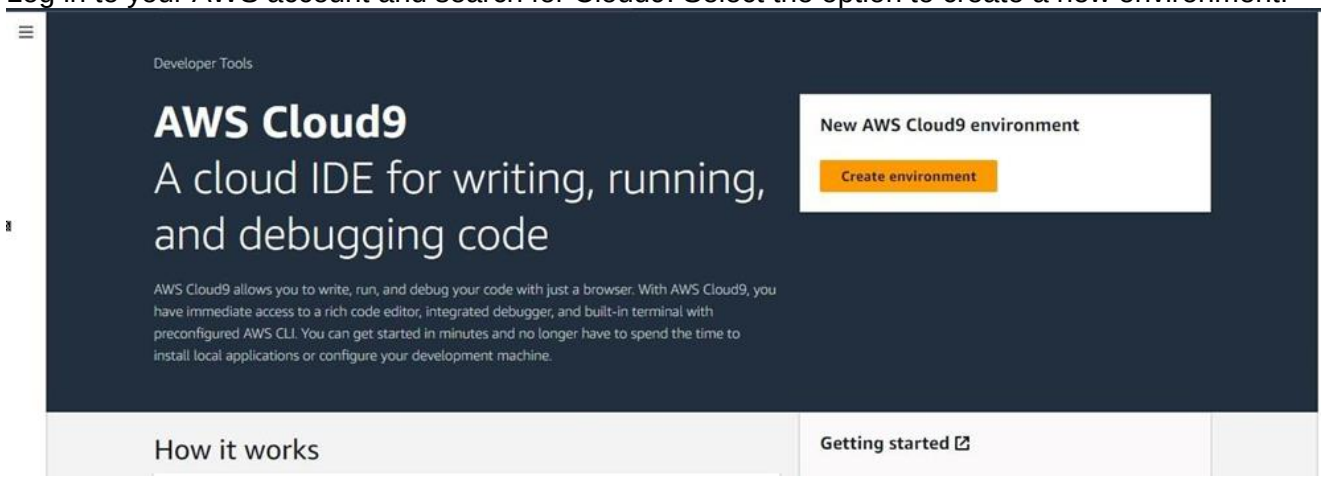
Step 10 : Now reload the website. You can see your website with public IP:



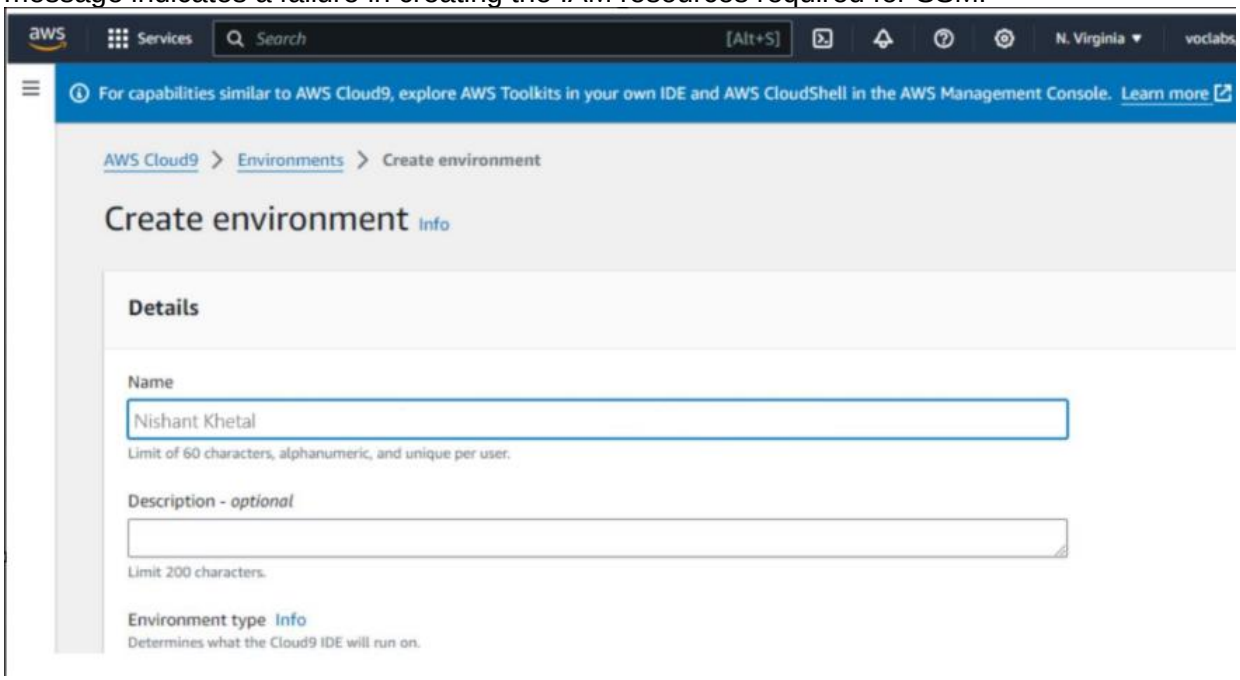
Name: Nishant S Khetal D15C Roll No:24

Experiment No. 1B

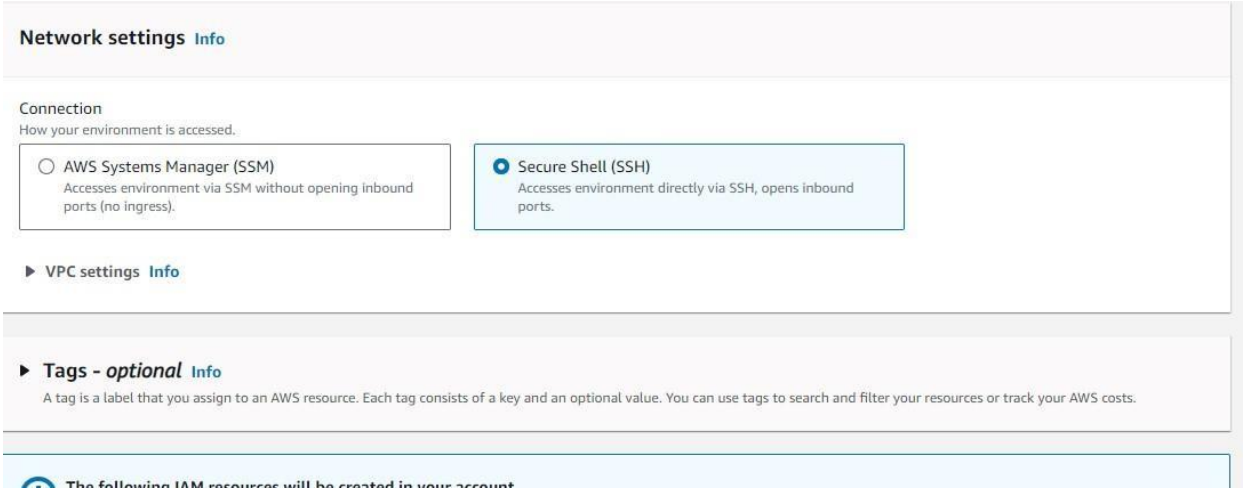
Log in to your AWS account and search for Cloud9. Select the option to create a new environment.



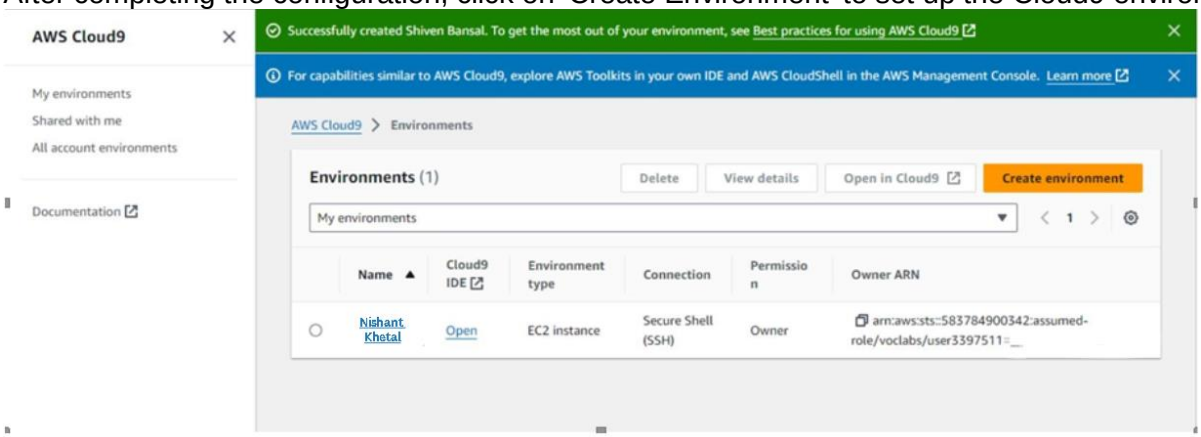
Provide the name and other necessary configurations to create the environment. When configuring network settings, attempting to use the AWS Systems Manager may result in an error. The error message indicates a failure in creating the IAM resources required for SSM.



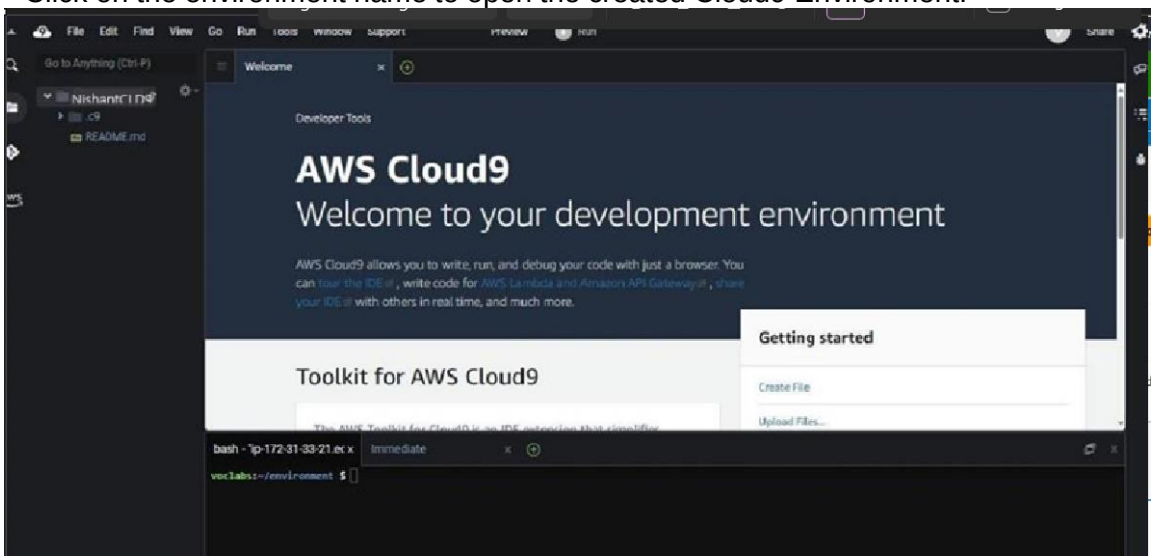
Use the Secure Shell option in Network settings



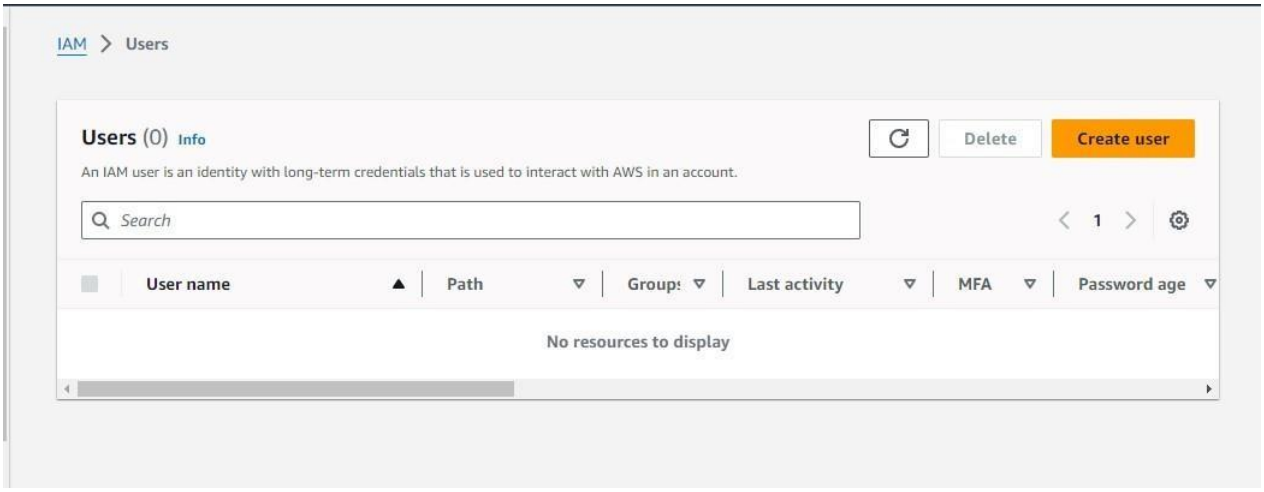
After completing the configuration, click on 'Create Environment' to set up the Cloud9 environment.



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

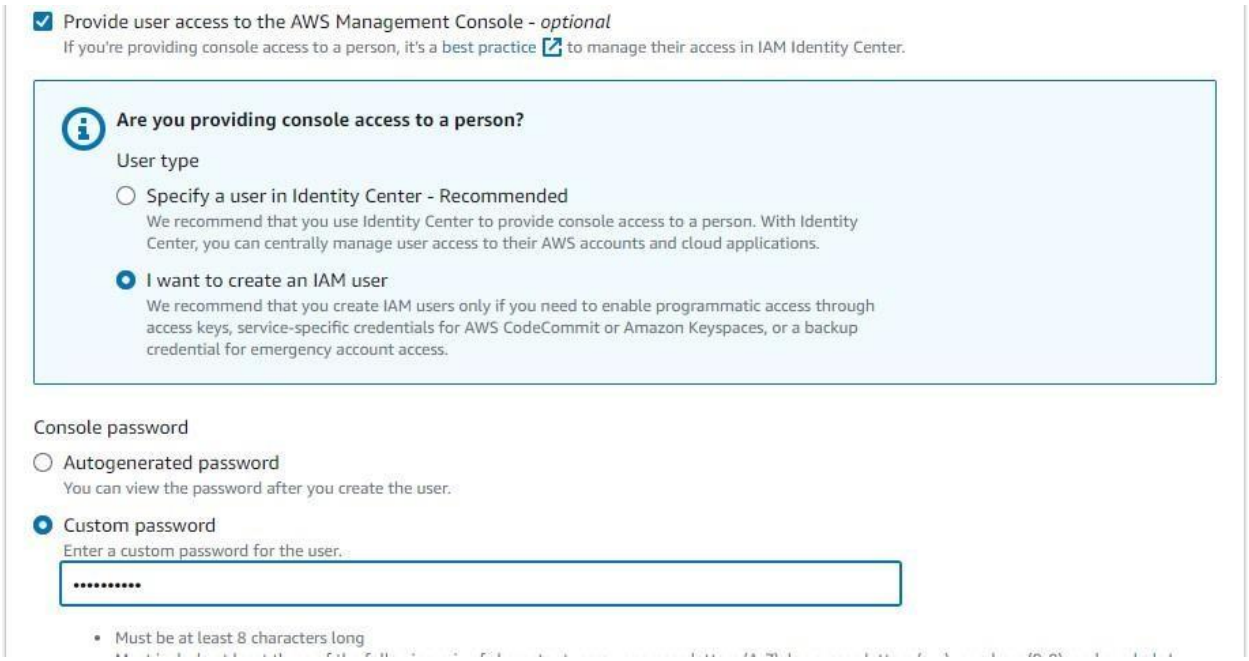


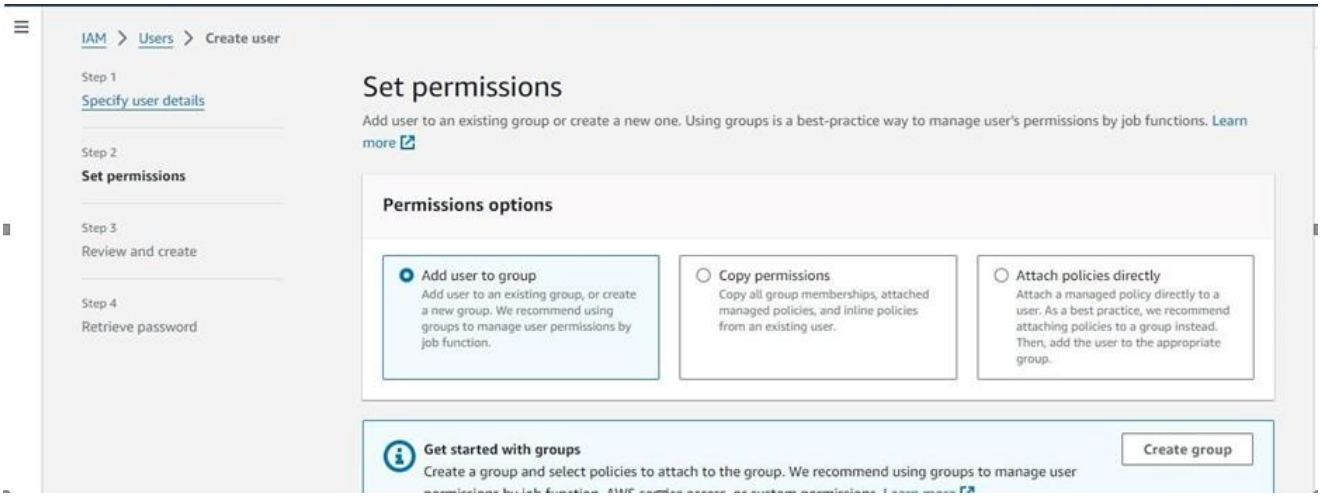
Log in to your AWS account and search for the IAM service. Navigate to the 'Users' tab and click on



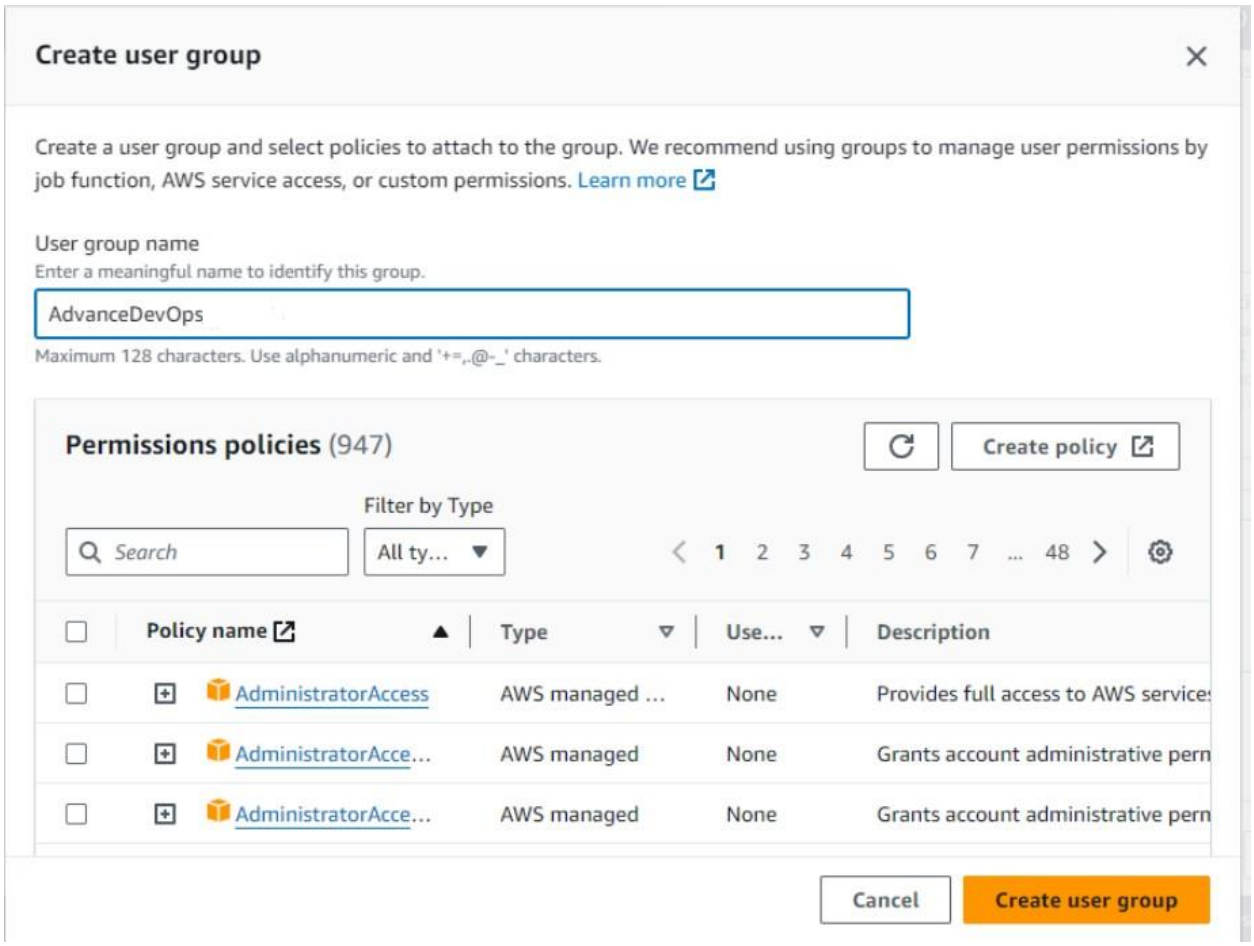
'Create User' to add a new user.

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

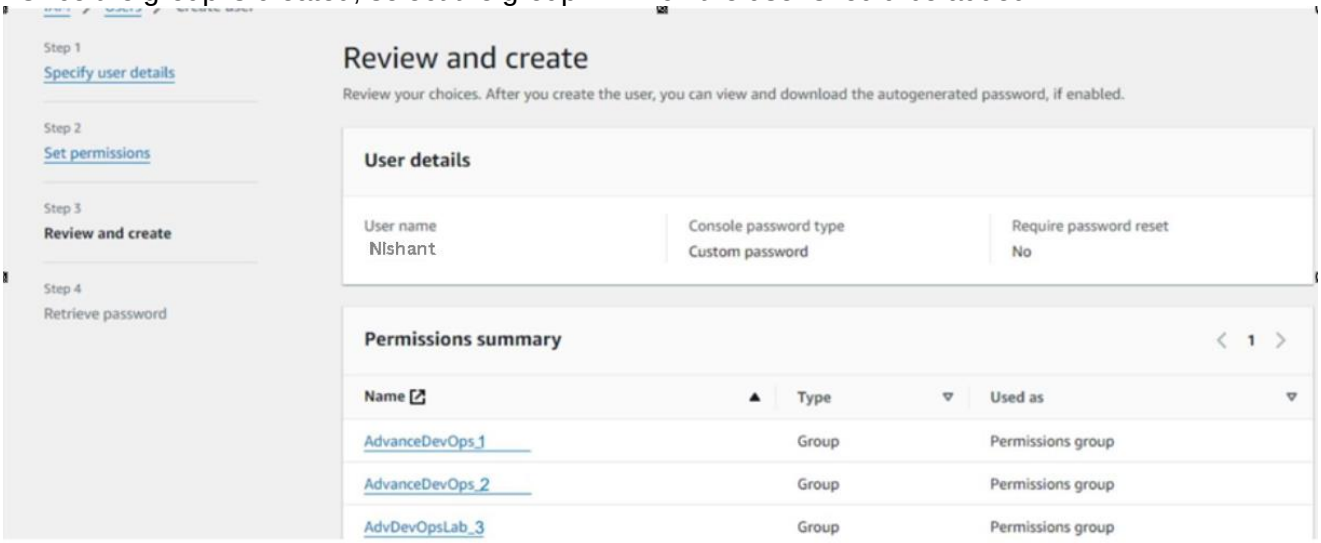

Provide user access to the AWS Management Console - optional
If you're providing console access to a person, it's a best practice to manage their access in IAM Identity Center.
Are you providing console access to a person?
User type
Specify a user in Identity Center - Recommended
We recommend that you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


IAM > Users > Create user
Step 1
Specify user details
Step 2
Set permissions
Step 3
Review and create
Step 4
Retrieve password
Set permissions
Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. Learn more
Permissions options
Add user to group
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.
Copy permissions
Copy all group memberships, attached managed policies, and inline policies from an existing user.
Attach policies directly
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.
Get started with groups
Create a group and select policies to attach to the group. We recommend using groups to manage user
Create group

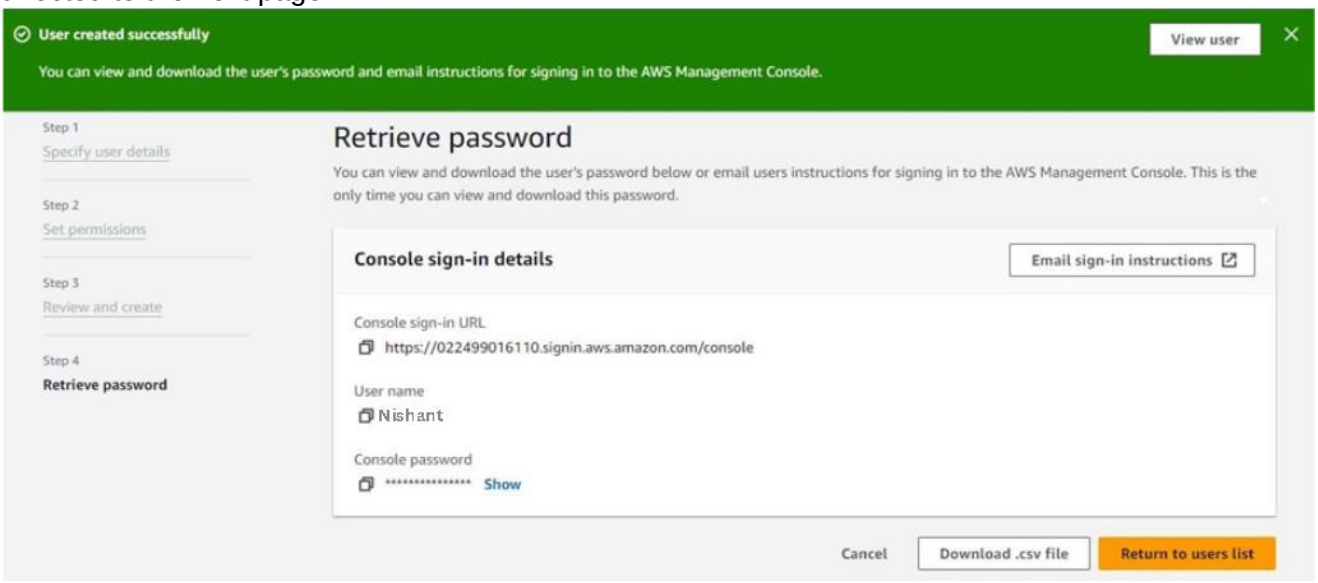
Next, click on 'Add User to Group.' If no existing group is available, choose 'Create Group.' Assign a name to the group, select any necessary policies, and then create the group.



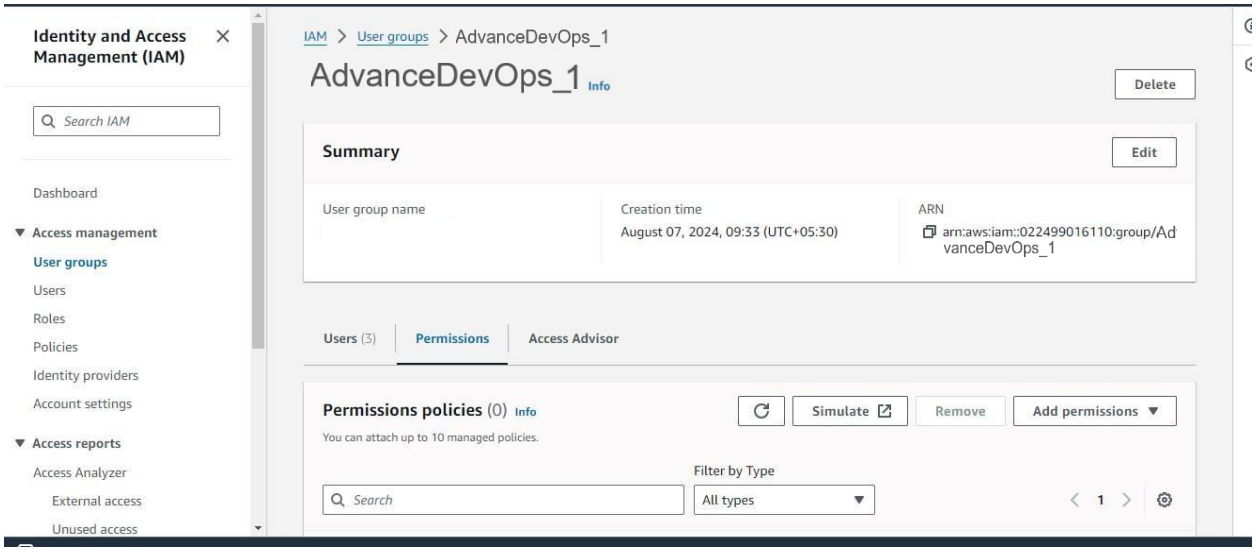
Once the group is created, select the group in which the user should be added.



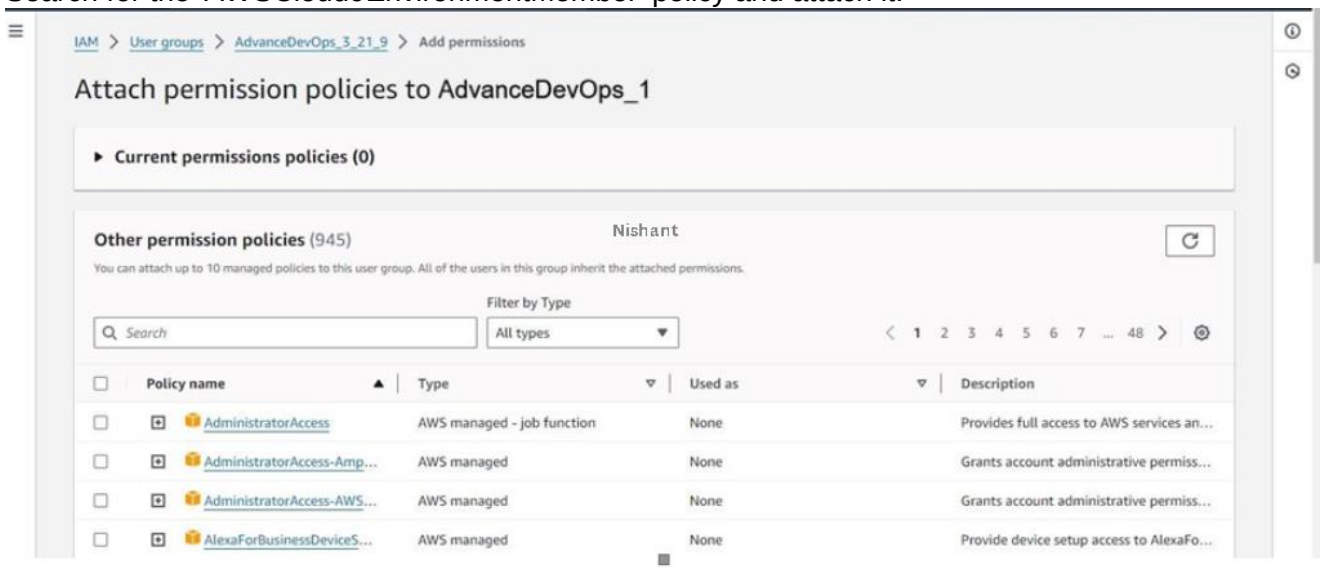
Review all the configurations and user details, then click on 'Create User.' Afterward, you will be directed to the next page.

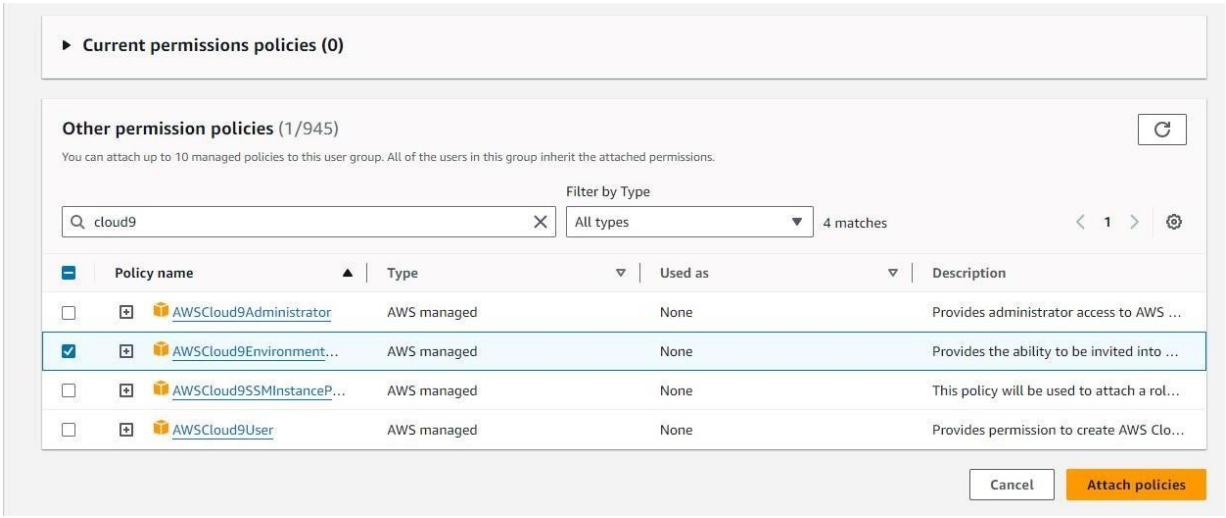


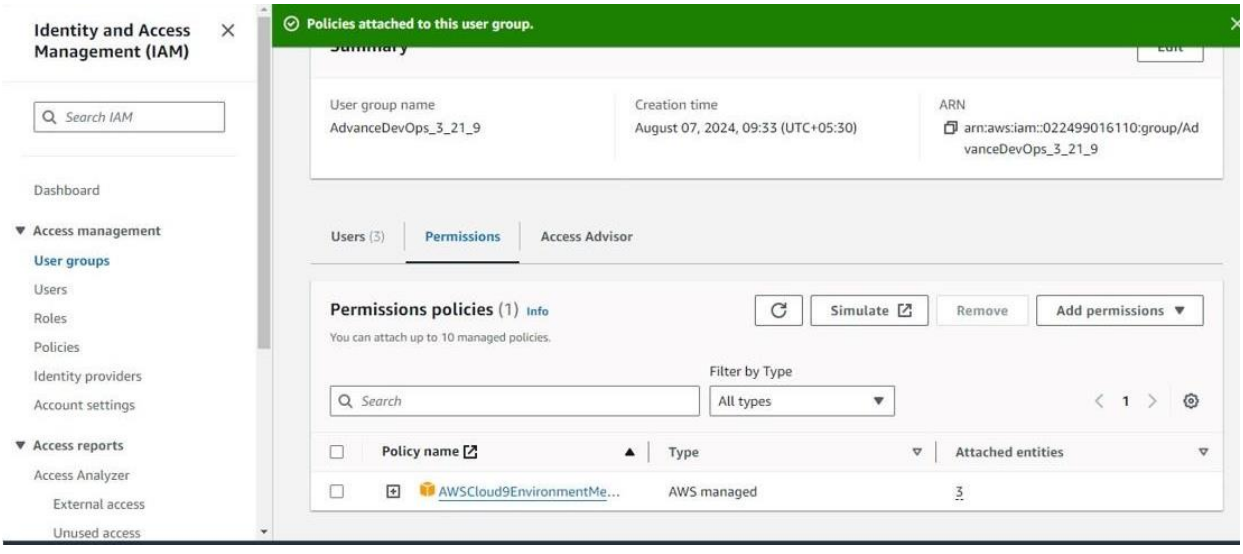
After creation of the user, go to the user groups tab and select the group in which the user has been added. Navigate to the permissions tab. Click on add permissions and attach policy.



Search for the 'AWSCloud9EnvironmentMember' policy and attach it.





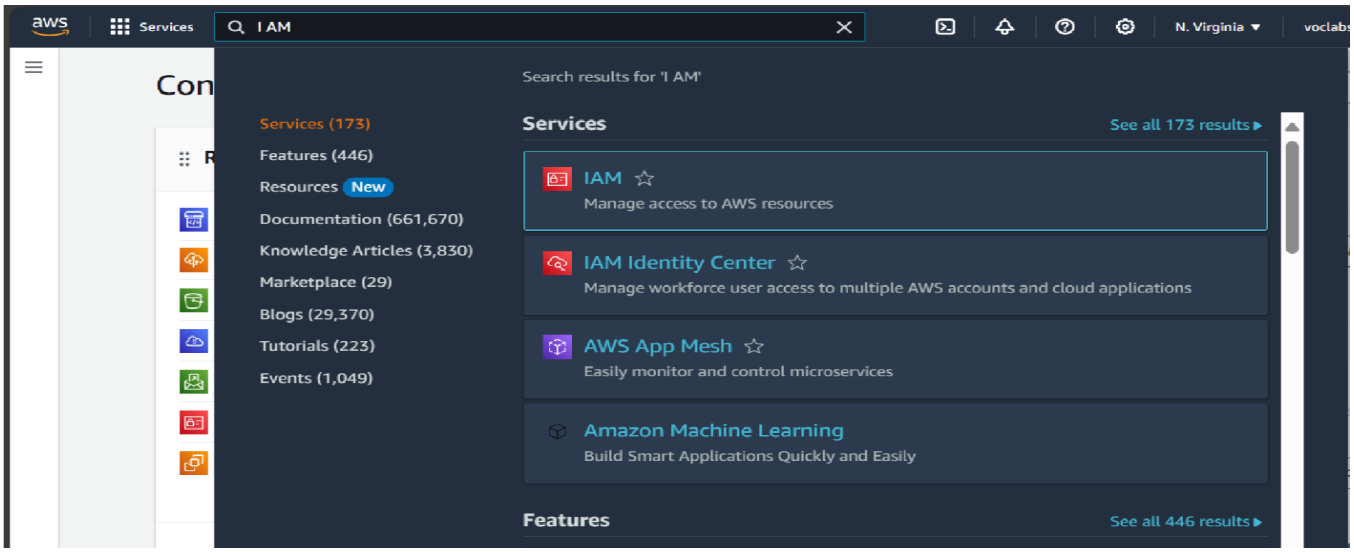


Name: Nishant S Khetal **D15C** **Roll No: 24**

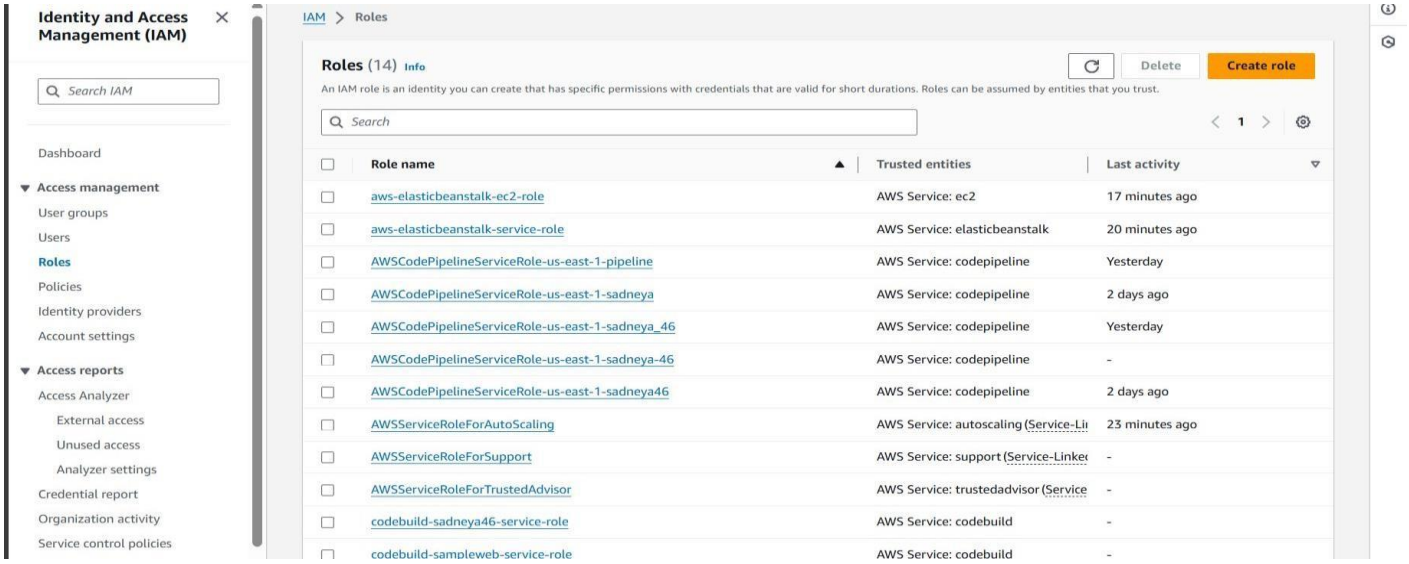
Experiment No: 2

Step1 :- Creation of role:

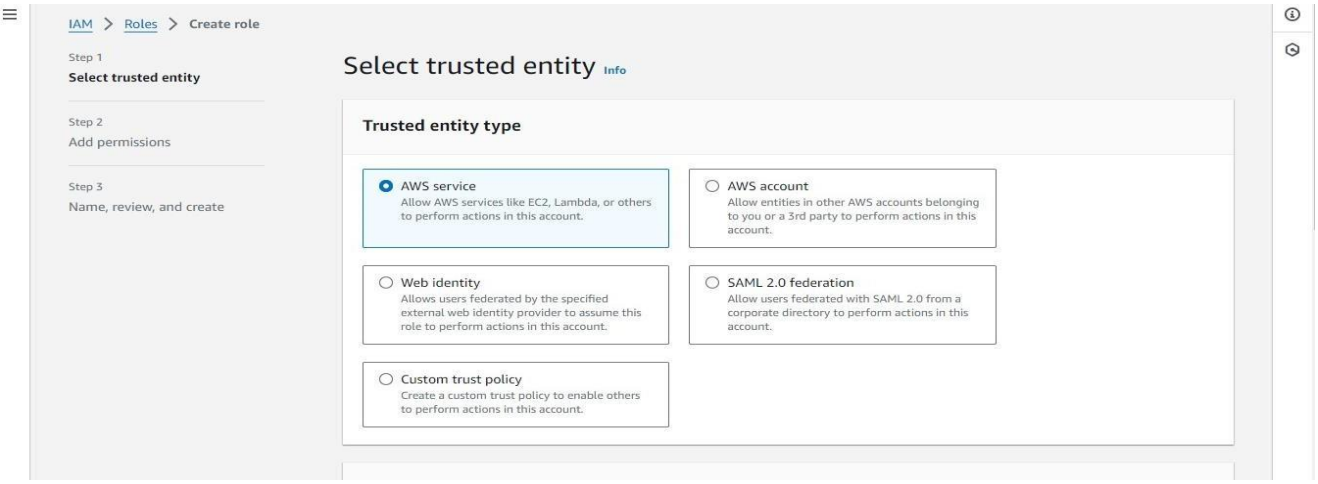
1. Login to your AWS account and search for IAM



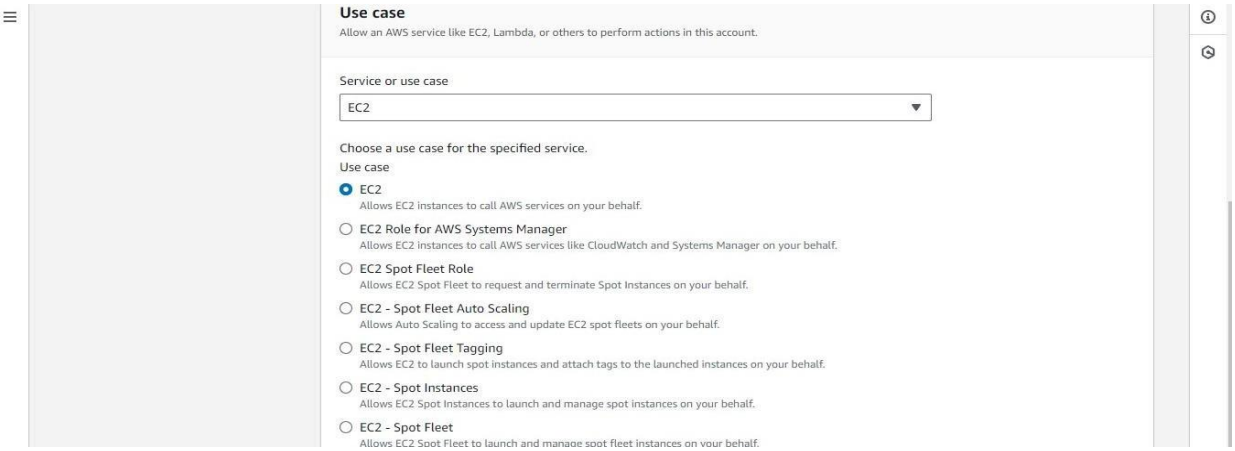
2. Then go into the role section and click on create role.



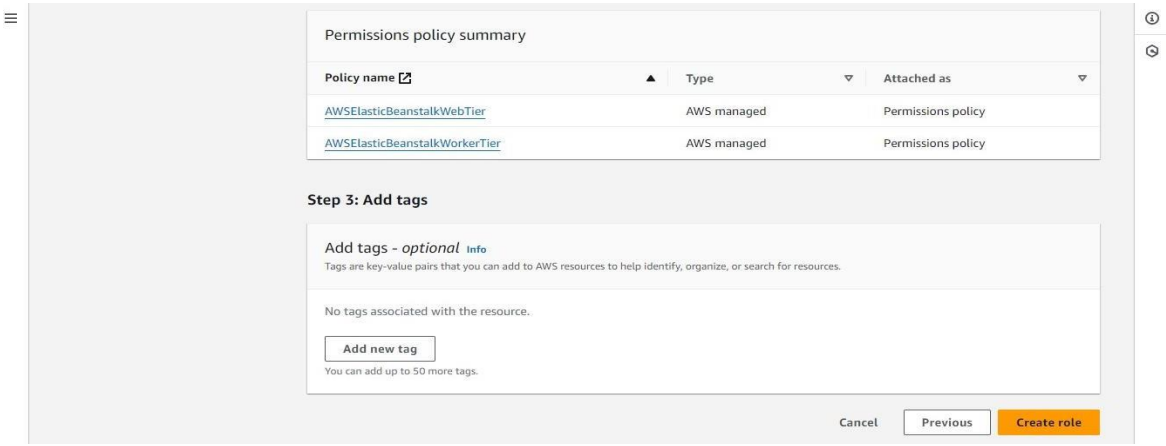
3. Then select a trusted entity as AWS service.



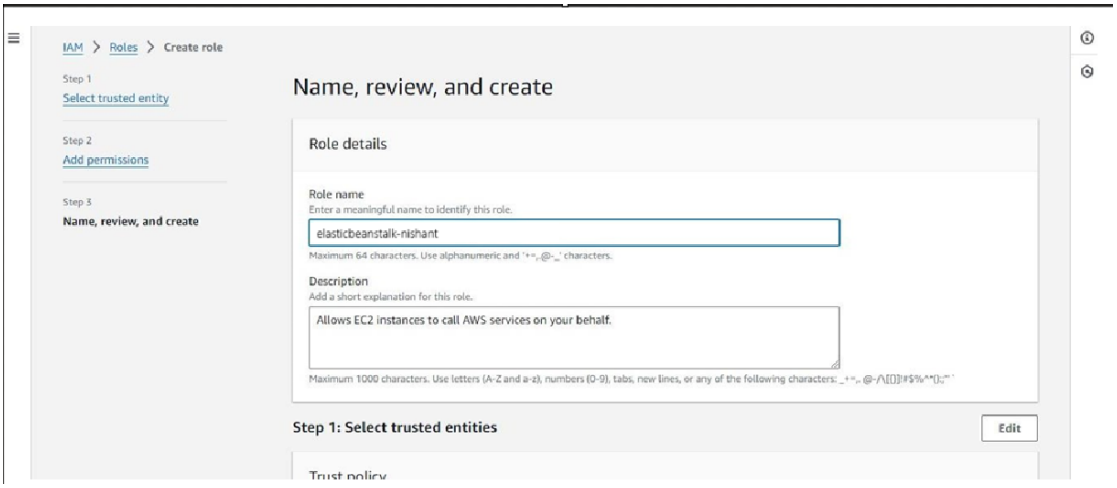
4. Select use case as EC2.



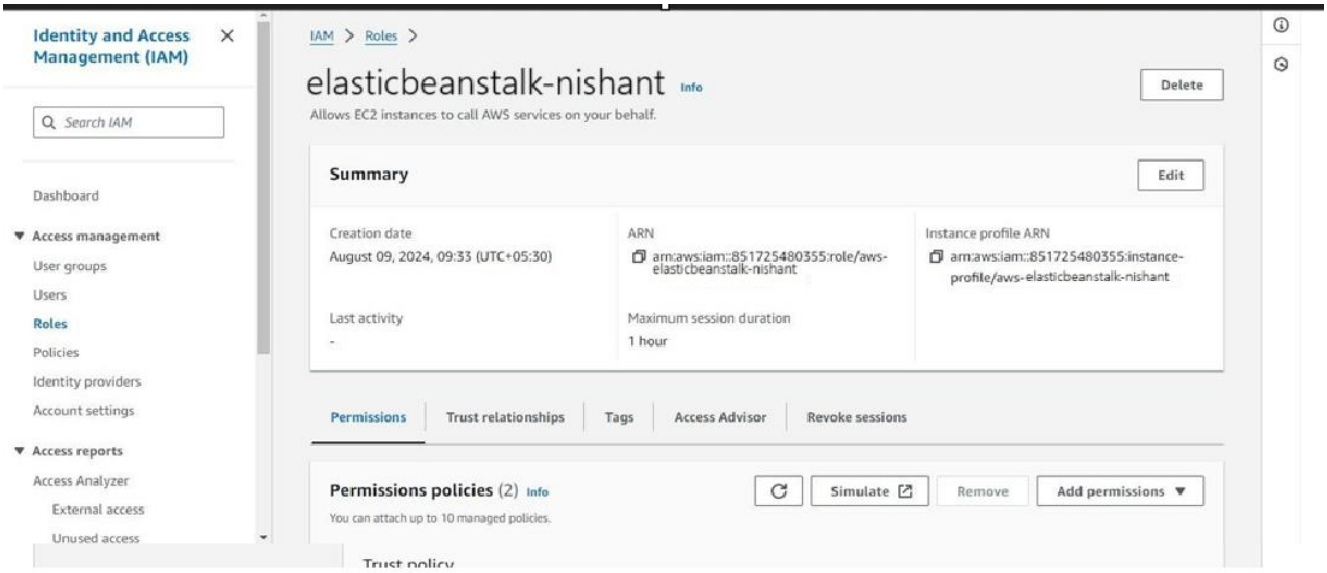
5. Select permissions as AWS Elastic Beanstalk Web Tier and AWS elastic Beanstalk worker tier.



6. Give a name to Role.

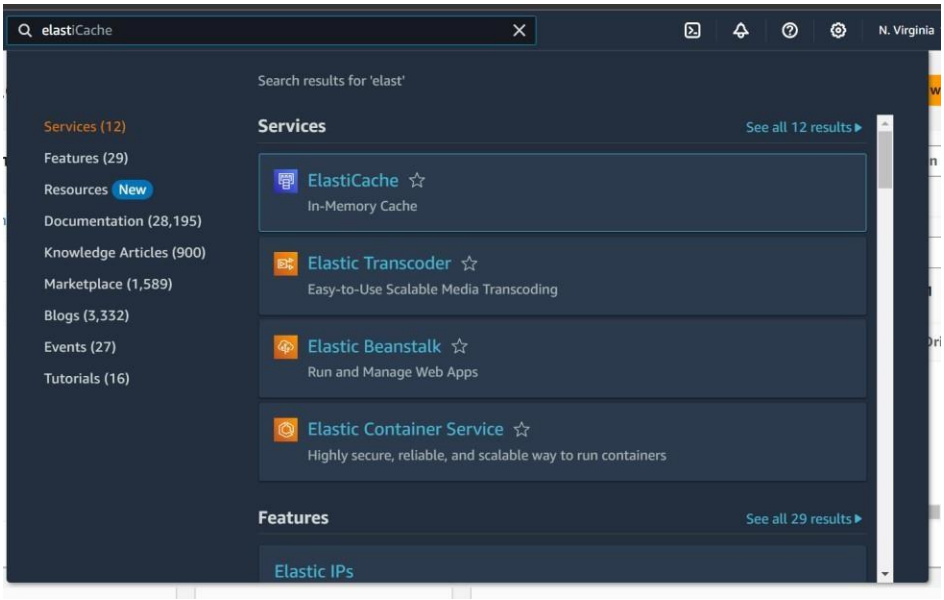


7. Then the role gets created

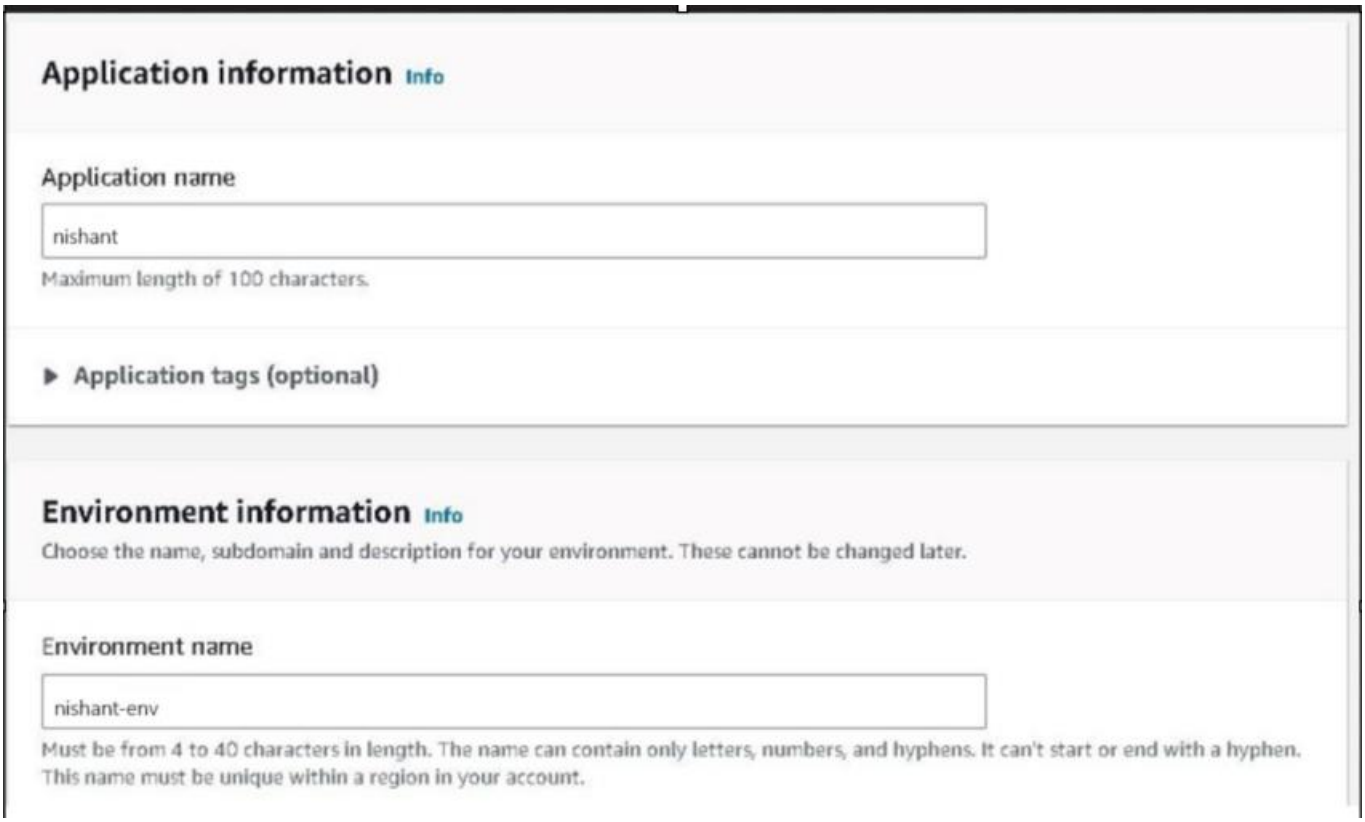


Step 2 :- Creation Elastic Beanstalk Environment

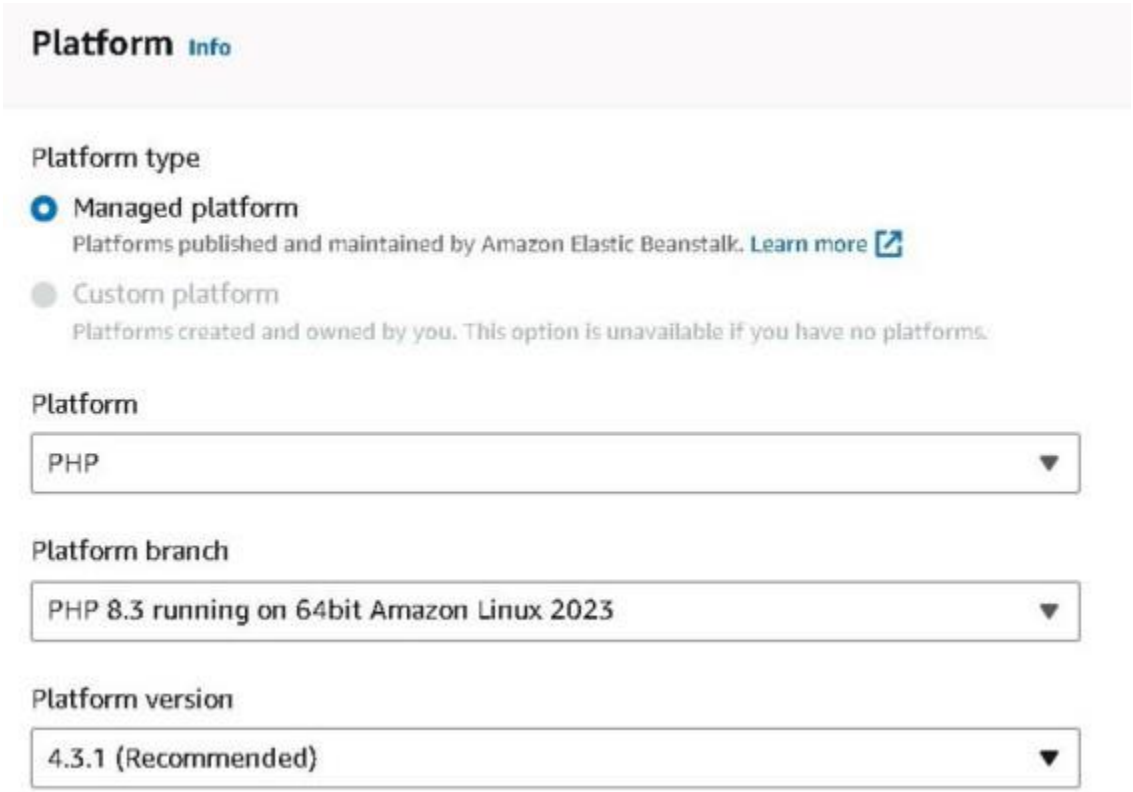
1. search for Elastic Beanstalk in the search box.



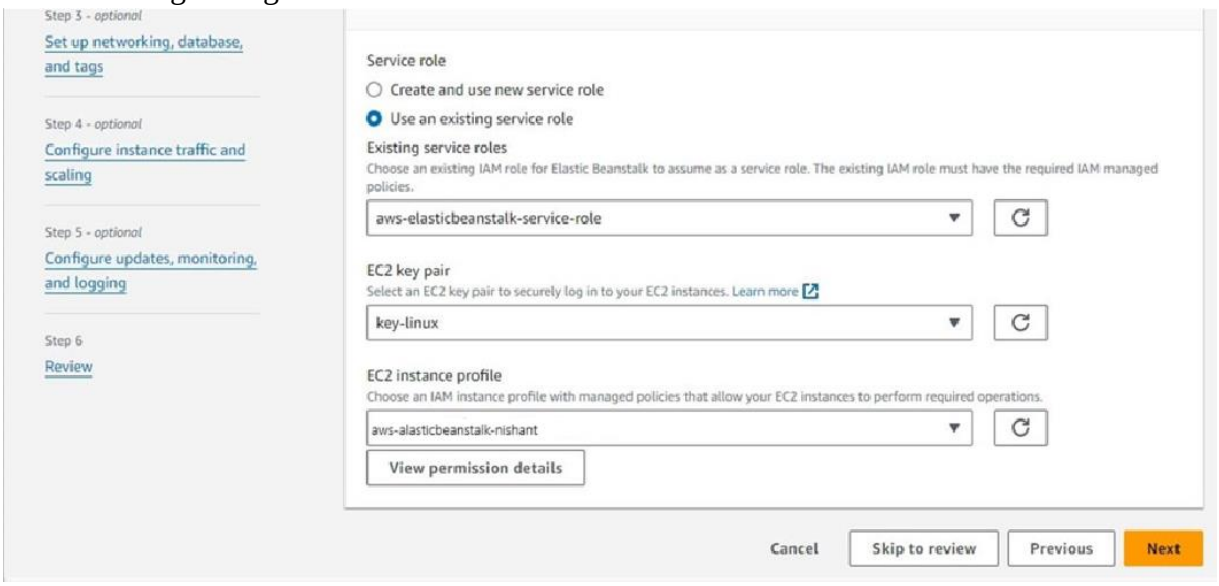
2. Open up Elastic Beanstalk and name your web app.



3. Select platform as PHP.

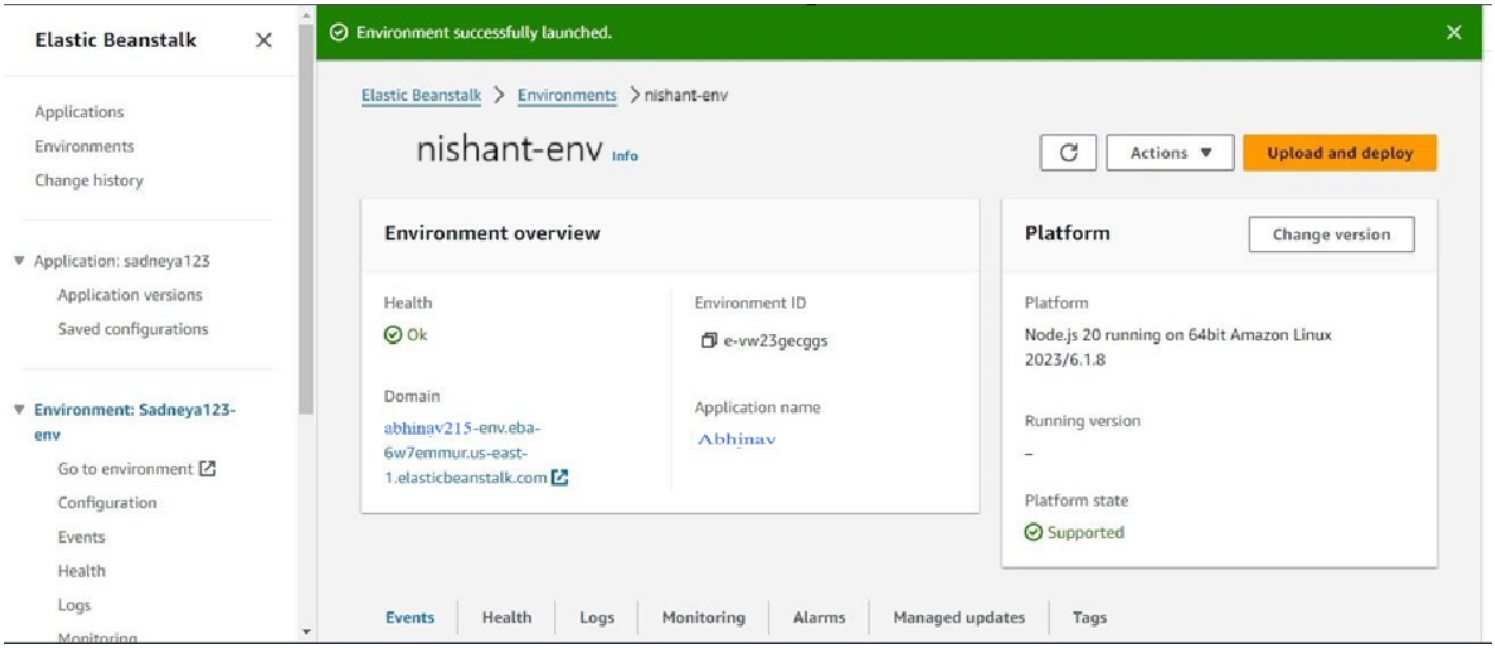


4. After clicking on next you need to select the use existing role. Then you will see the existing role select it like here it is aws-elasticbeanstalk-service-role. Which we created in 1st part. Select role, then select key you have created then profile will be automatically selected according to role. then click on create application by keeping all the remaining settings as it is.



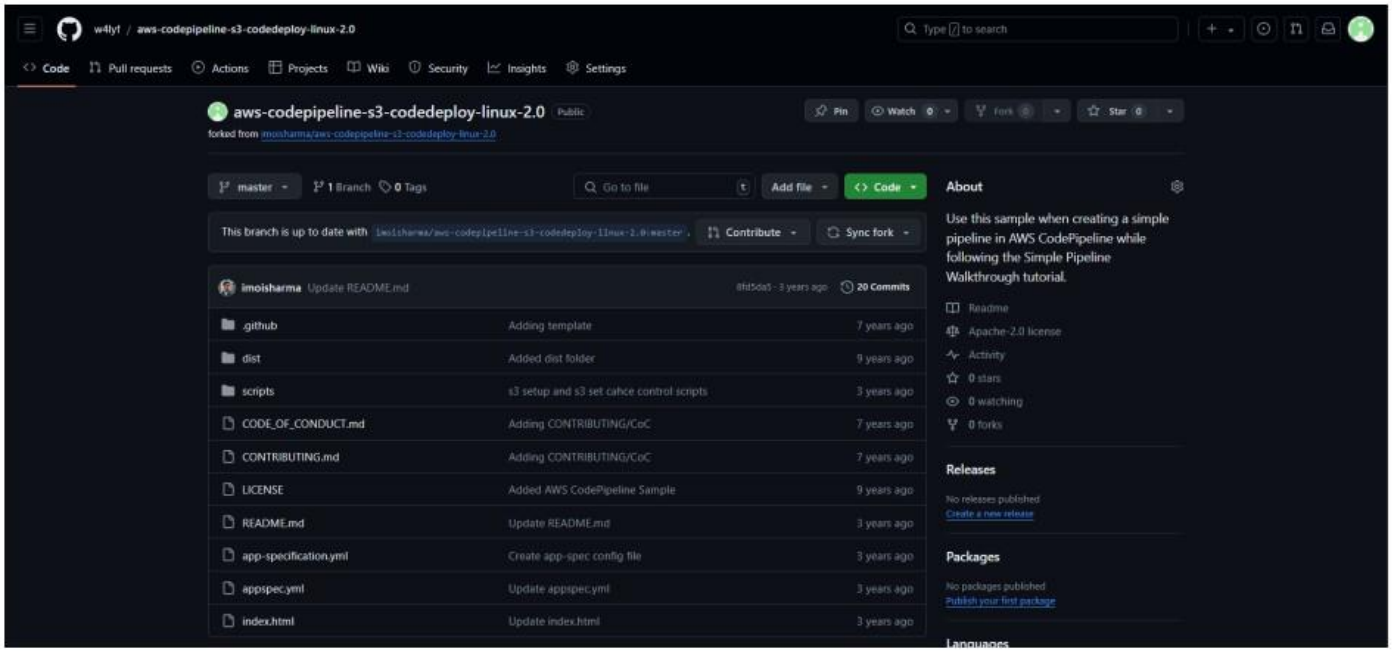
Keep Set up networking, database and tags, Configure instance traffic and scaling, Configure updates, monitoring and logging all these default.

5. Beanstalk creates a sample environment for you to deploy your application. By default, it creates an EC2 instance, a security group, an Auto Scaling group, an Amazon S3 Bucket, Amazon CloudWatch alarms and a domain name for your Application.



Step 3: Get a copy of your sample code

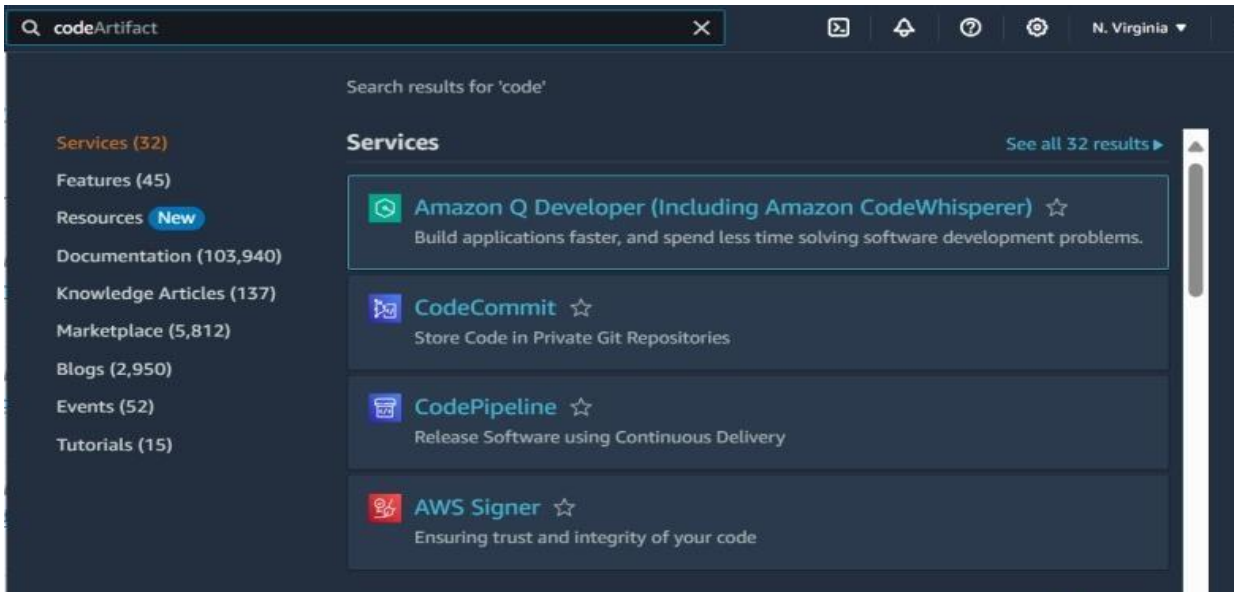
In this step, we will get the sample code from this GitHub Repository to later host it. The pipeline takes code from the source and then performs actions on it.
For this experiment, as a source, we will use this forked GitHub repository. We can alternatively also use Amazon S3 and AWS CodeCommit.
Go to the repository shared above and simply fork it.



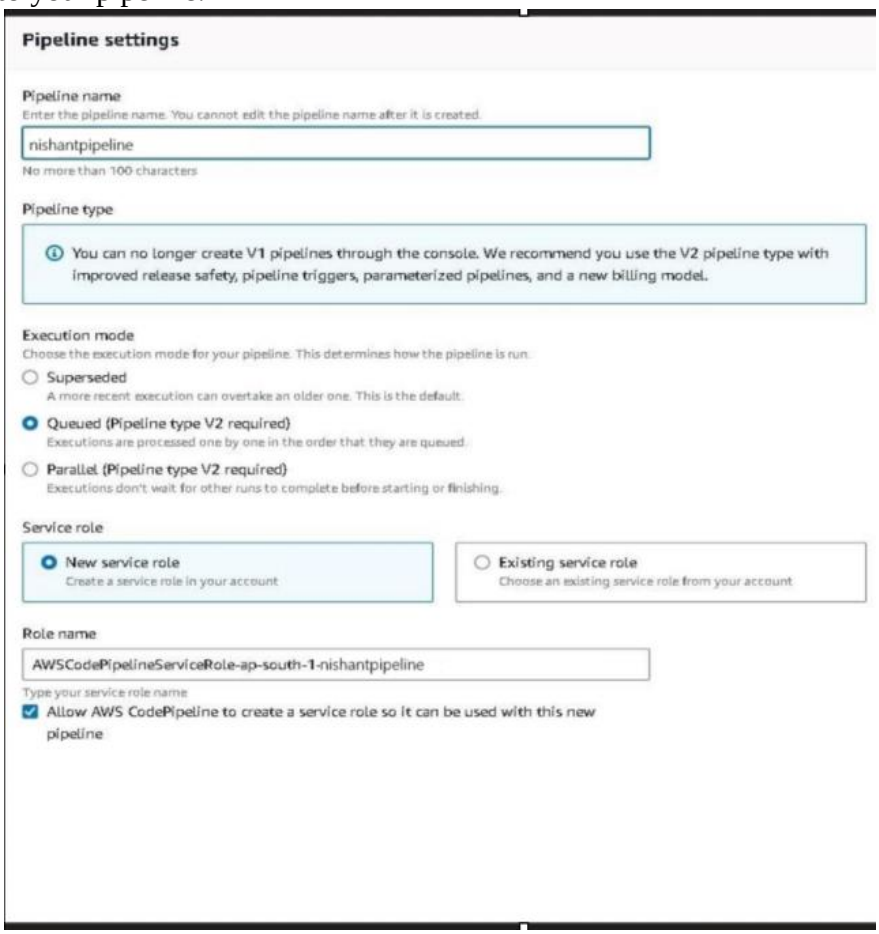
Step 4: Creating a CodePipeline

In this step, we'll create a simple pipeline that has its source and deployment information. In this case, however, we will skip the build stage where you get to plug in our preferred build provider.

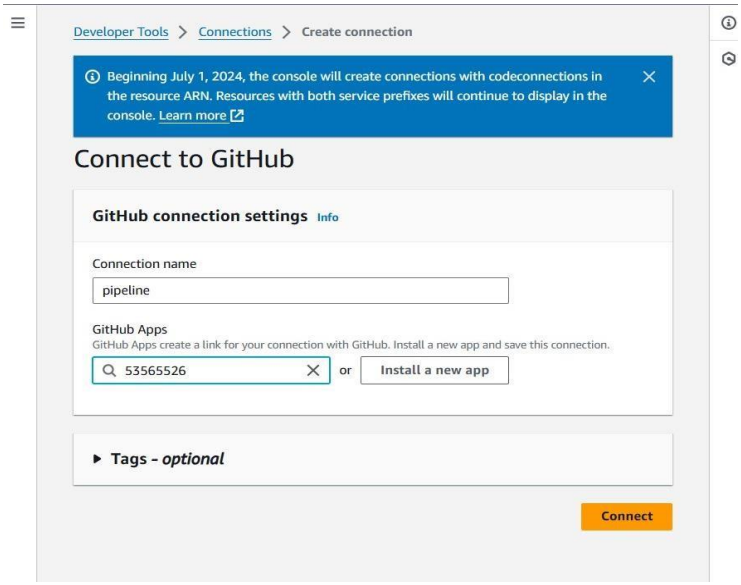
1. Search CodePipeline in the search bar and click on create a new Pipeline.

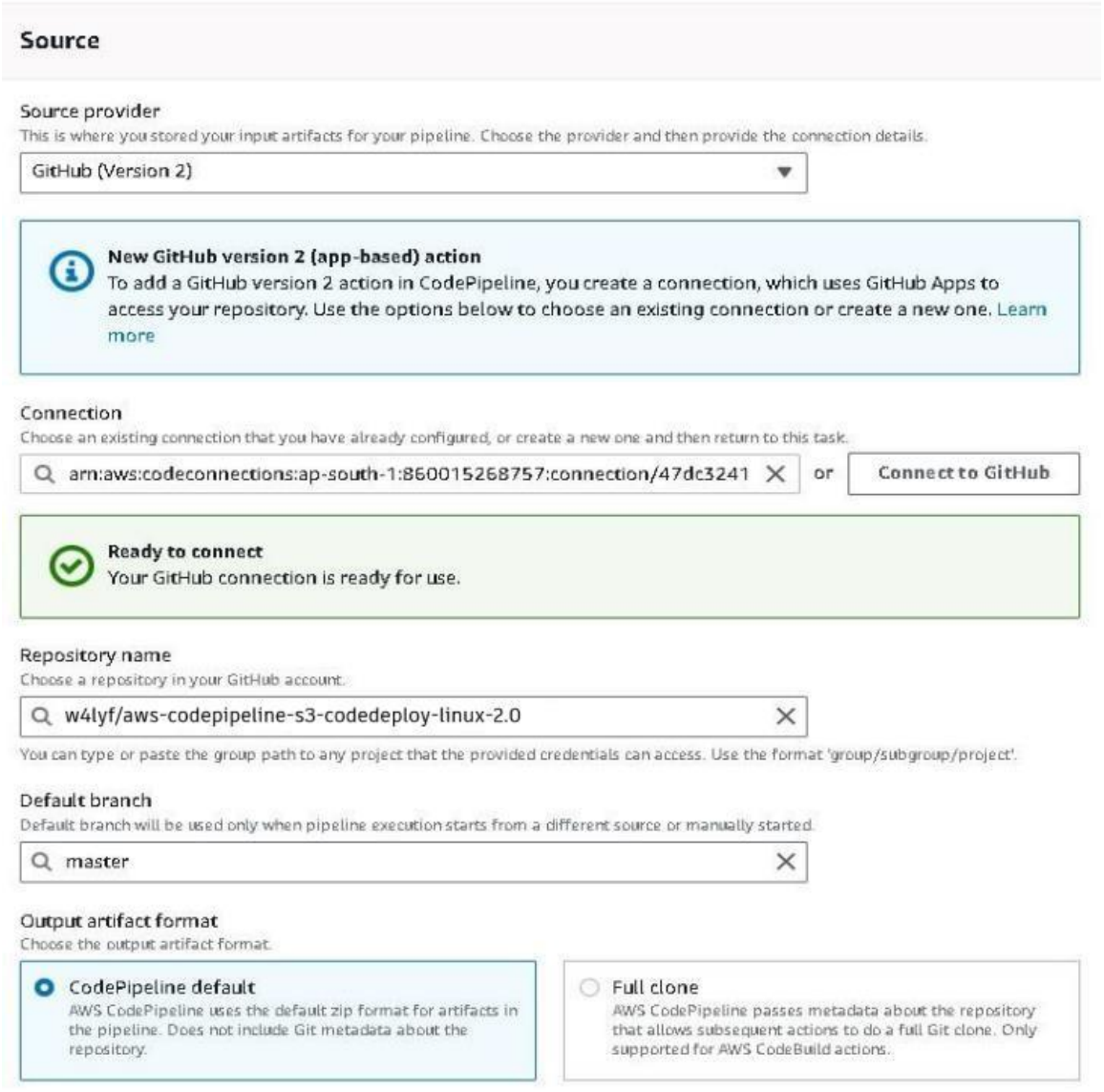


2. Give a name to your pipeline.



3. In the source stage, choose GitHub v2 as the provider, then connect your GitHub account to AWS by creating a connection. You'd need your GitHub credentials and then you'd need to authorize and install AWS on the forked GitHub Repository.



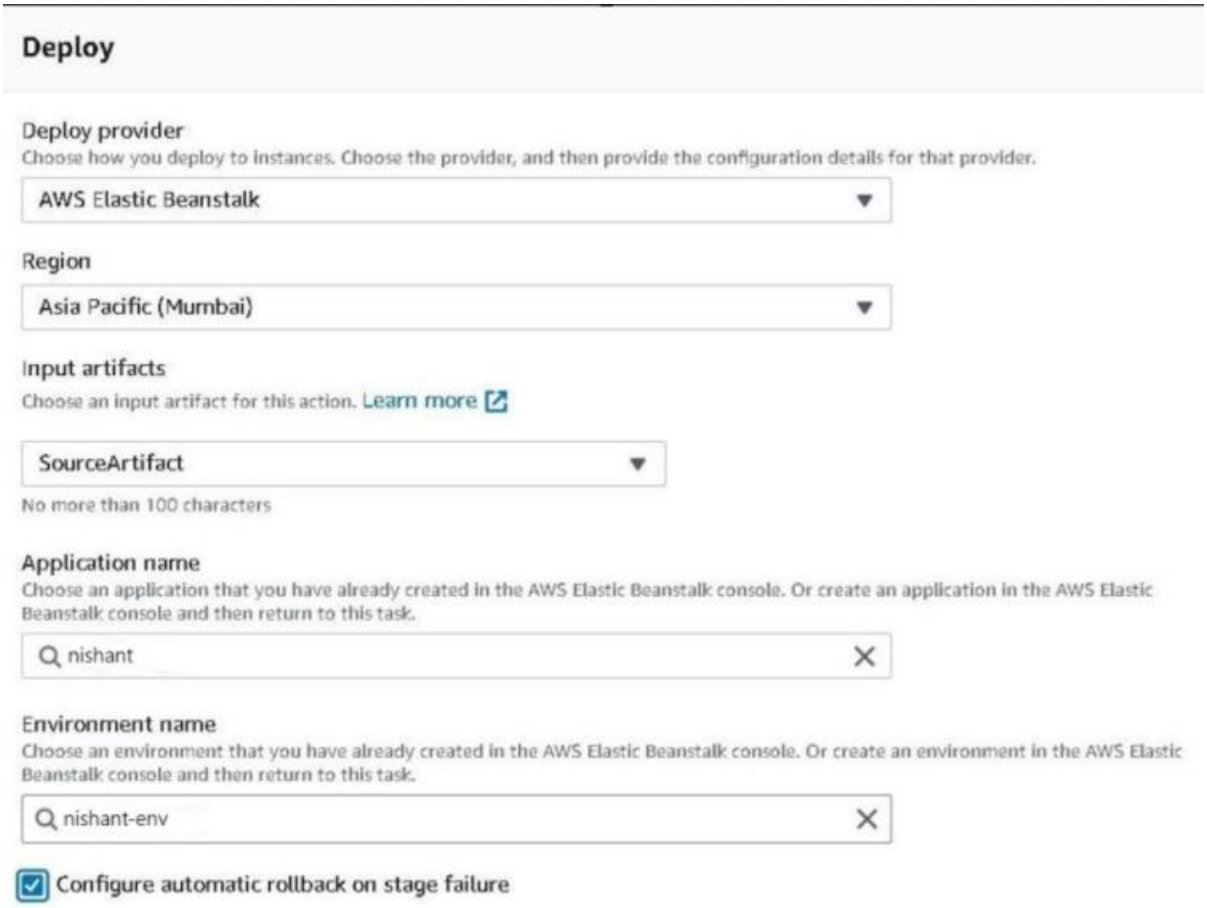


4. Then select trigger type none.

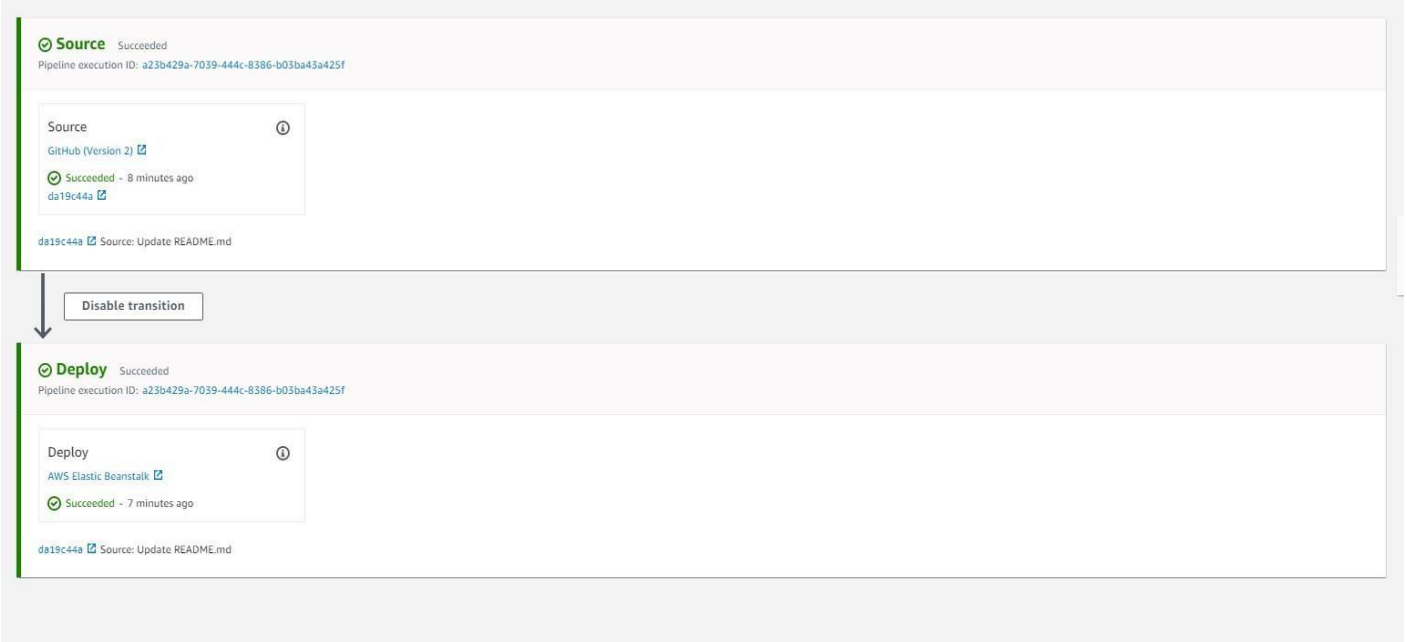
After that, click Continue and skip the build stage. Proceed to the Deployment stage.

Step 5: Deployment

1. Choose Beanstalk as the Deploy Provider, same region as the Bucket and Beanstalk, name and environment name.



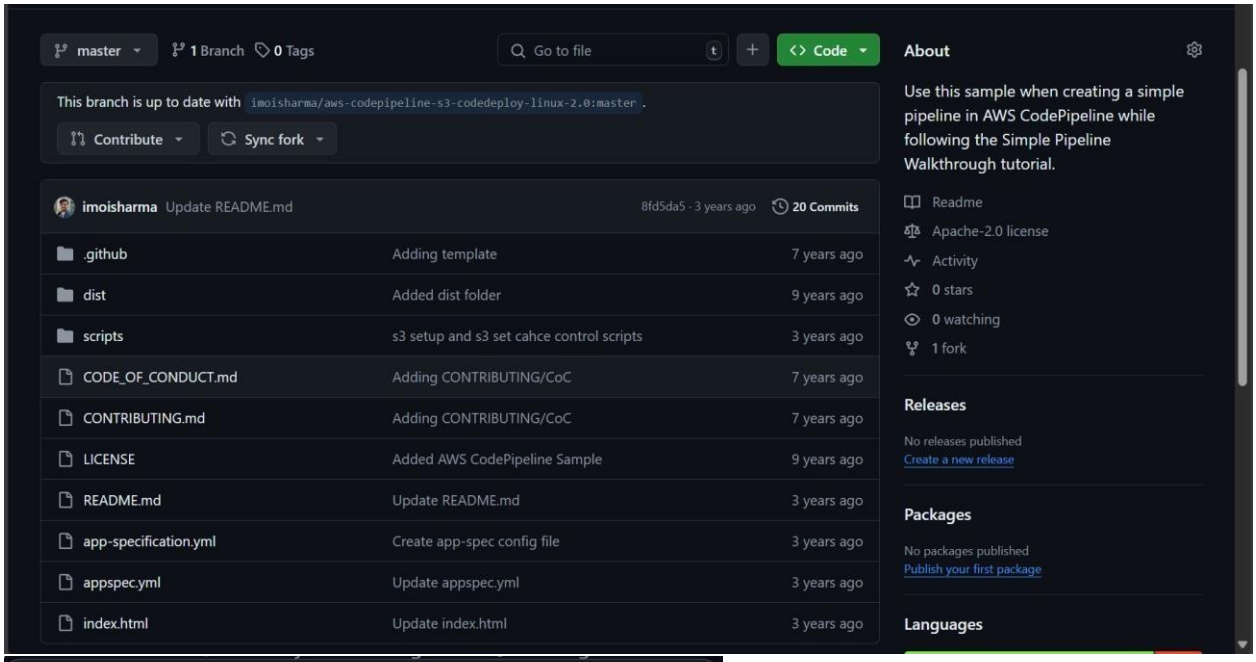
2. Then it will give you this result on screen. i.e. deployed successfully.

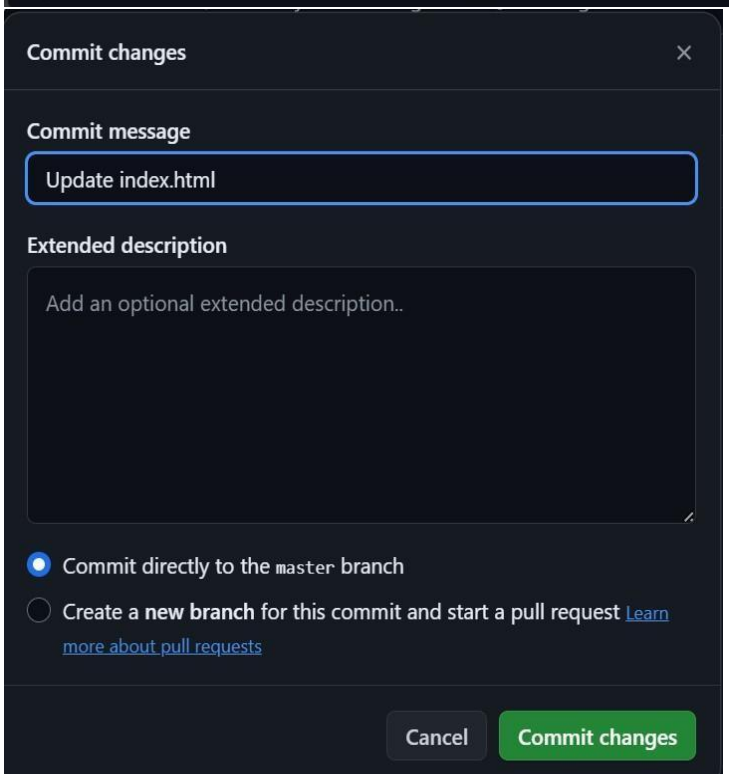


If you can see this, that means that you successfully created an automated software using CodePipeline.

Step 6: Committing changes to update app

1. In this we make some changes in the file. Open github.com then open the forked repository. Then update the changes in the index.html file and finally commit those changes.





2. Then again start the deployment of the pipeline. And check the changes in the website

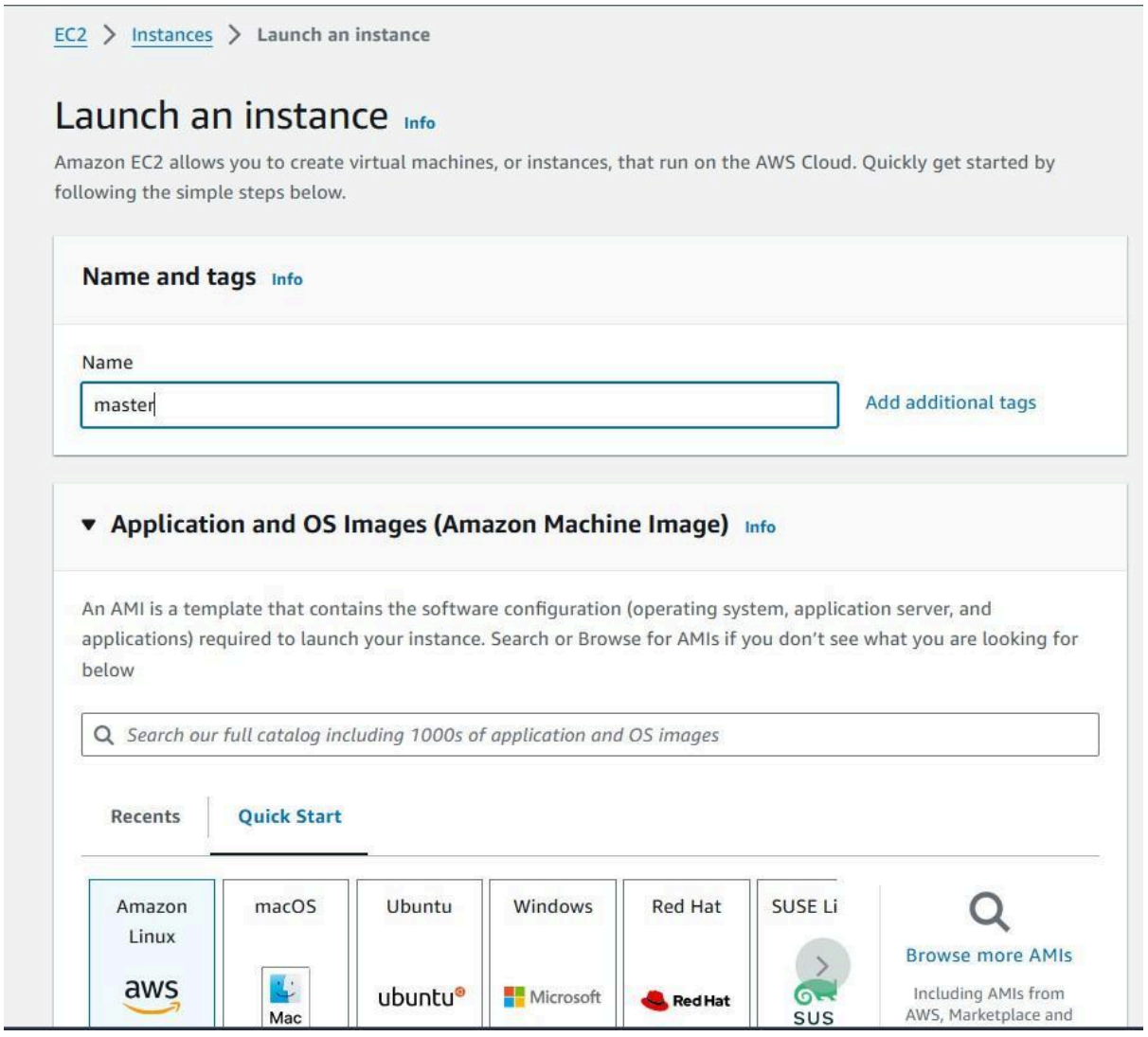
Name: Nishant S Khetal **D15C** **Roll No: 24**

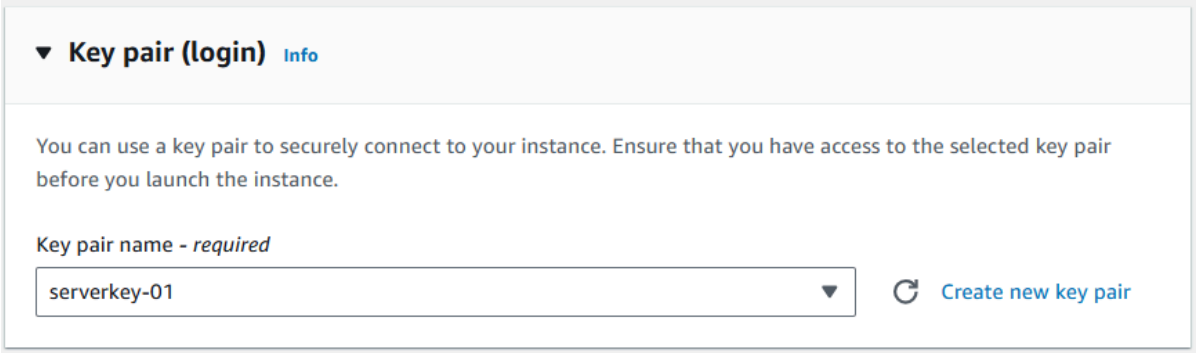
Experiment No. 3

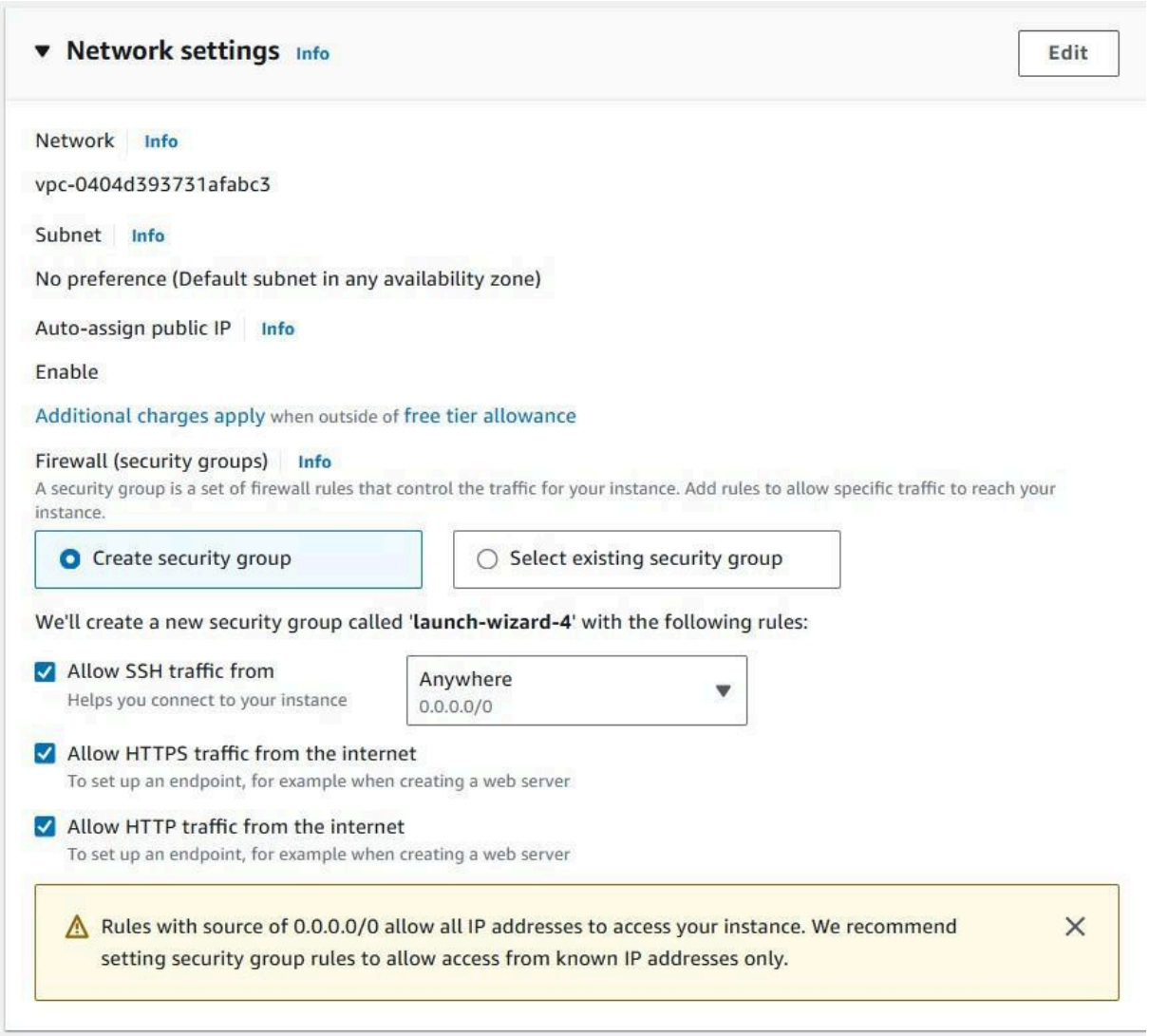
Aim: To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud.

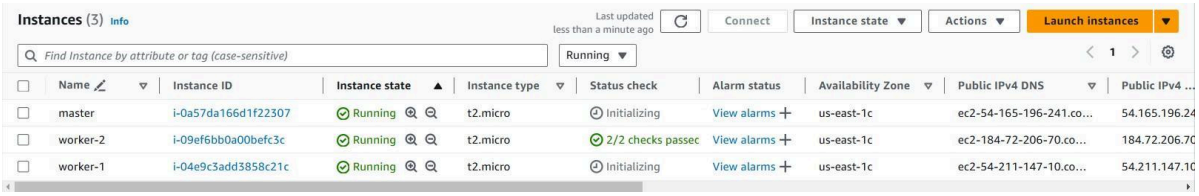
Steps:

1. **Create 3 EC2 Ubuntu Instances on AWS. (Name 1 as Master, the other 2 as worker-1 and worker-2)**



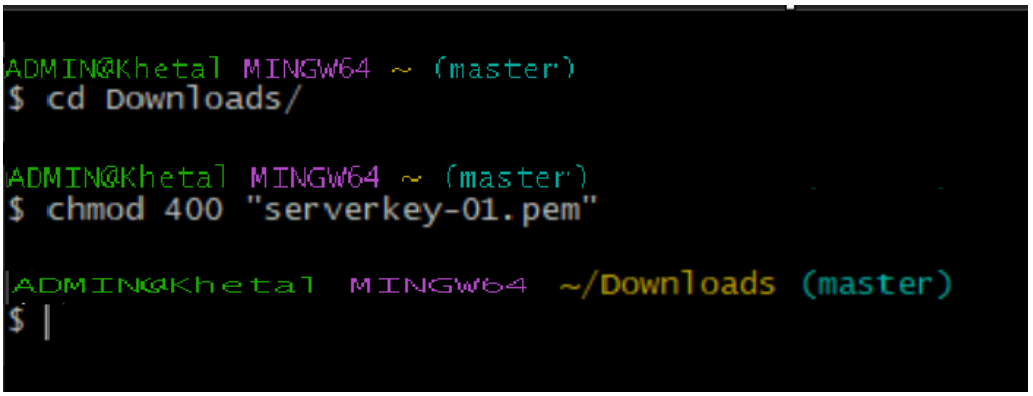






2. SSH into all 3 machines

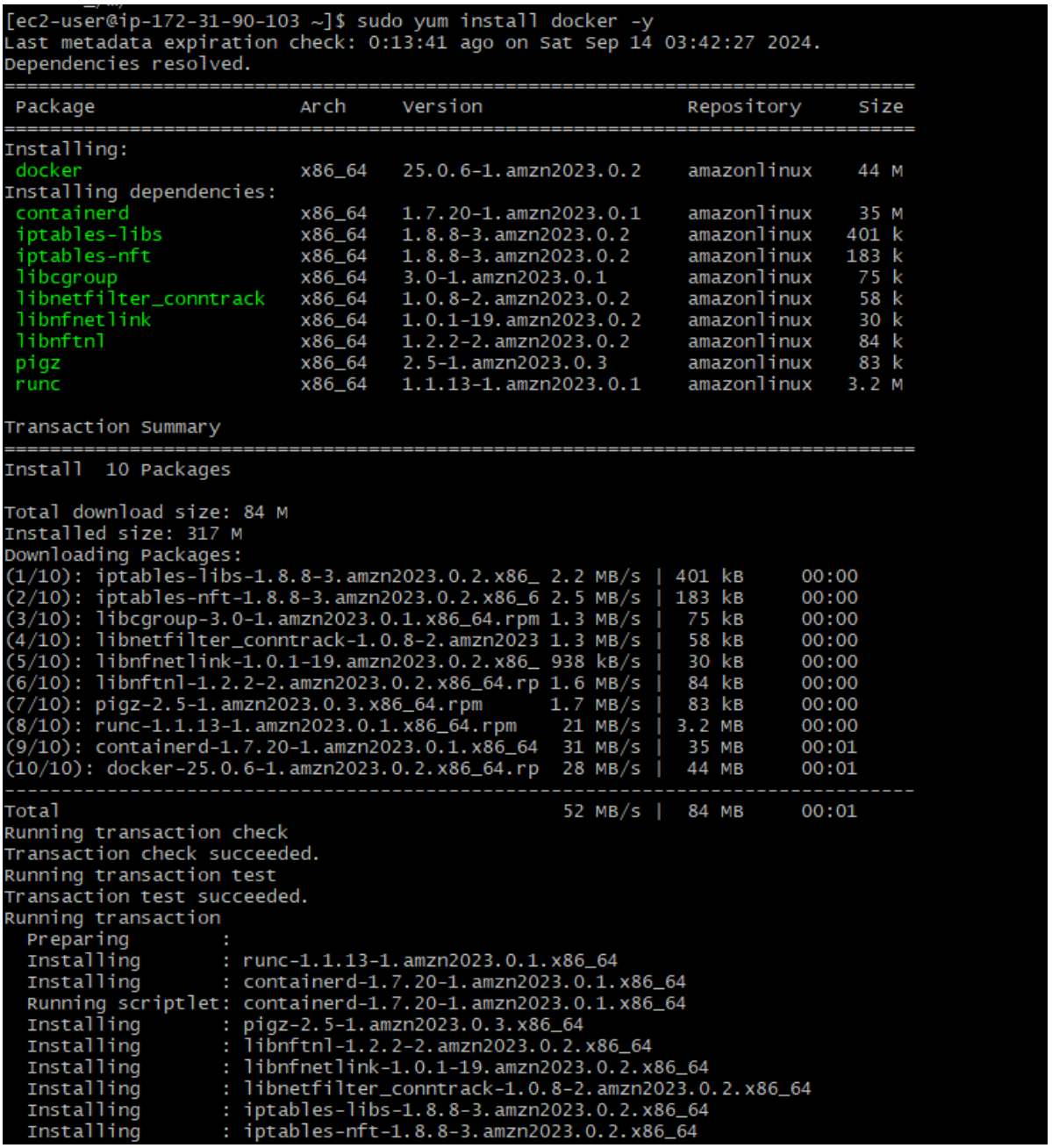
- Give permissions to the current user to the downloaded pem file using -
  chmod 400 <security_filename.pem>



- ssh into all the three machines using –

  ssh -i (keyname).pem (username)@(public ipv4 dns address)
  where keyname is name of the key you created. (server-01.pem). Other details can be found on the Instance dashboard.

3. Installation Of Docker on three machines



- **sudo yum install docker -y**

```
Installed:
  containerd-1.7.20-1.amzn2023.0.1.x86_64              docker-25.0.6-1.amzn2023.0.2.x86_64
  libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64   libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64

Complete!
[ec2-user@ip-172-31-90-103 ~]$
```

- Configure cgroup in a daemon.json

  (this can be done by creating the file and using **nano** text editor)

  {

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

  "log-driver": "json-file",

  "log-opts": {

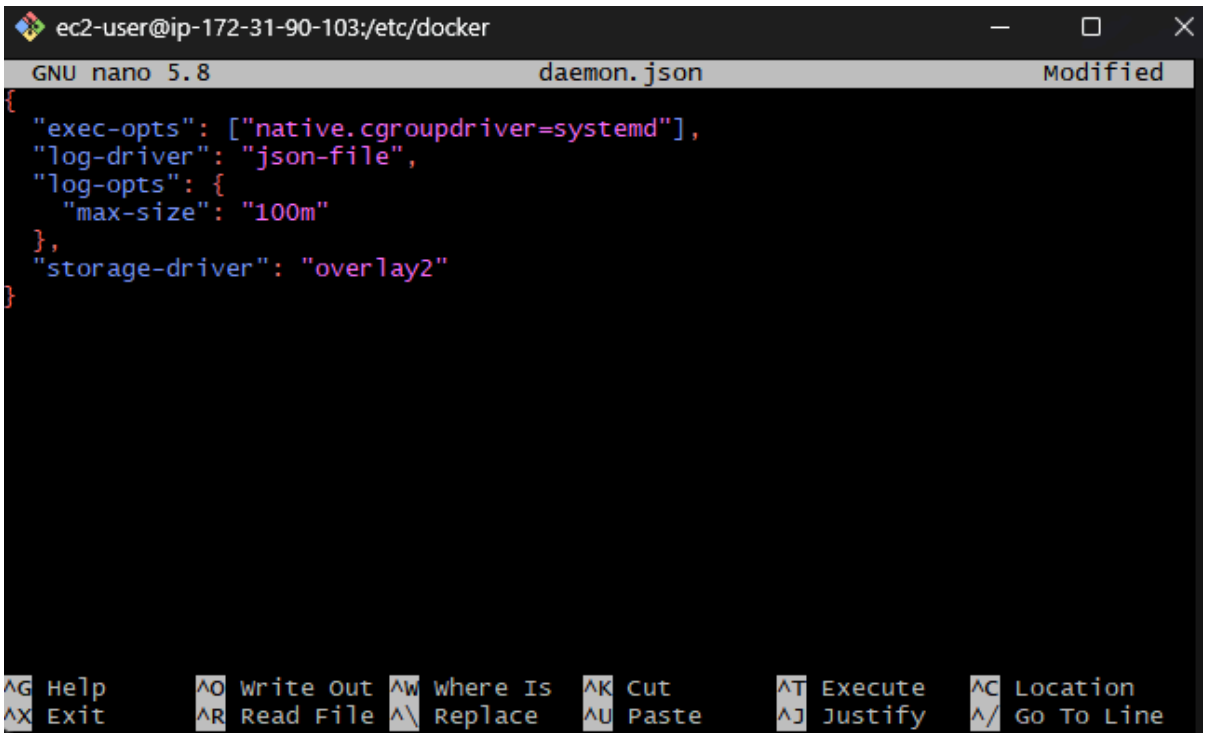
  "max-size": "100m"

  },

  "storage-driver": "overlay2"

  }

```
[ec2-user@ip-172-31-90-103 docker]$ sudo nano daemon.json
[ec2-user@ip-172-31-90-103 docker]$
```



- Enable and start docker and also load the daemon.json

  sudo systemctl enable docker

  sudo systemctl daemon-reload

  sudo systemctl restart docker

```
[ec2-user@ip-172-31-90-103 docker]$ sudo systemctl enable docker
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /us
r/lib/systemd/system/docker.service.
[ec2-user@ip-172-31-90-103 docker]$ sudo systemctl daemon-reload
[ec2-user@ip-172-31-90-103 docker]$
sudo systemctl restart docker
[ec2-user@ip-172-31-90-103 docker]$
```

- Check if docker is installed

```
[ec2-user@ip-172-31-90-103 docker]$ docker --version
Docker version 25.0.5, build 5dc9bcc
[ec2-user@ip-172-31-90-103 docker]$
```

4. Install Kubernetes on all 3 machines

- SELinux needs to be disabled before configuring kubelet

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

```
[ec2-user@ip-172-31-90-103 docker]$ sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
[ec2-user@ip-172-31-90-103 docker]$
```

- Add Kubernetes using the repo

  (this is done by creating **kubernetes.repo** file in **/etc/yum.repos.d** and configuring it using **nano** editor)
  [kubernetes]
  name=Kubernetes
  baseurl=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/
  enabled=1
  gpgcheck=1
  gpgkey=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/repodata/repomd.xml.key
  exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni

```
[ec2-user@ip-172-31-90-103 docker]$ cd /etc/yum.repos.d/
[ec2-user@ip-172-31-90-103 yum.repos.d]$ ls
amazonlinux.repo  kernel-livepatch.repo
[ec2-user@ip-172-31-90-103 yum.repos.d]$ sudo nano kubernetes.repo
[ec2-user@ip-172-31-90-103 yum.repos.d]$ ls
amazonlinux.repo  kernel-livepatch.repo  kubernetes.repo
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

```
ec2-user@ip-172-31-90-103:/etc/yum.repos.d
 GNU nano 5.8                    kubernetes.repo                    Modified
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line
```

- Update packages list using **sudo yum update**

```
[ec2-user@ip-172-31-90-103 yum.repos.d]$ sudo yum update
Kubernetes                                      125 kB/s |  17 kB     00:00
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

- Install kubelet kubeadm kubectl

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

```
[ec2-user@ip-172-31-90-103 yum.repos.d]$ sudo yum install -y kubelet kubeadm kub
ectl --disableexcludes=kubernetes
Last metadata expiration check: 0:00:42 ago on Sat Sep 14 04:08:20 2024.
Dependencies resolved.
================================================================================
 Package                 Arch     Version                  Repository     Size
================================================================================
Installing:
 kubeadm                 x86_64   1.30.5-150500.1.1        kubernetes     10 M
 kubectl                 x86_64   1.30.5-150500.1.1        kubernetes     10 M
 kubelet                 x86_64   1.30.5-150500.1.1        kubernetes     17 M
Installing dependencies:
 conntrack-tools         x86_64   1.4.6-2.amzn2023.0.2     amazonlinux   208 k
 cri-tools               x86_64   1.30.1-150500.1.1        kubernetes    8.6 M
 kubernetes-cni          x86_64   1.4.0-150500.1.1         kubernetes    6.7 M
 libnetfilter_cthelper   x86_64   1.0.0-21.amzn2023.0.2    amazonlinux    24 k
 libnetfilter_cttimeout  x86_64   1.0.0-19.amzn2023.0.2    amazonlinux    24 k
 libnetfilter_queue      x86_64   1.0.5-2.amzn2023.0.2     amazonlinux    30 k

Transaction Summary
================================================================================
Install  9 Packages

Total download size: 53 M
Installed size: 292 M
Downloading Packages:
(1/9): libnetfilter_cttimeout-1.0.0-19.amzn2023 448 kB/s |  24 kB     00:00
(2/9): libnetfilter_cthelper-1.0.0-21.amzn2023. 409 kB/s |  24 kB     00:00
(3/9): libnetfilter_queue-1.0.5-2.amzn2023.0.2. 1.5 MB/s |  30 kB     00:00
(4/9): conntrack-tools-1.4.6-2.amzn2023.0.2.x86 1.8 MB/s | 208 kB     00:00
(5/9): cri-tools-1.30.1-150500.1.1.x86_64.rpm    28 MB/s | 8.6 MB     00:00
(6/9): kubectl-1.30.5-150500.1.1.x86_64.rpm      23 MB/s |  10 MB     00:00
(7/9): kubeadm-1.30.5-150500.1.1.x86_64.rpm      18 MB/s |  10 MB     00:00
(8/9): kubelet-1.30.5-150500.1.1.x86_64.rpm      37 MB/s |  17 MB     00:00
(9/9): kubernetes-cni-1.4.0-150500.1.1.x86_64.r  20 MB/s | 6.7 MB     00:00
--------------------------------------------------------------------------------
Total                                            56 MB/s |  53 MB     00:00
```

```
Installed:
  conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64
  cri-tools-1.30.1-150500.1.1.x86_64
  kubeadm-1.30.5-150500.1.1.x86_64
  kubectl-1.30.5-150500.1.1.x86_64
  kubelet-1.30.5-150500.1.1.x86_64
  kubernetes-cni-1.4.0-150500.1.1.x86_64
  libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64
  libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64
  libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64

Complete!
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

- After installing Kubernetes, we need to configure internet options to allow bridging.

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

```
[ec2-user@ip-172-31-90-103 yum.repos.d]$ . sudo swapoff -a
-bash: sudo: No such file or directory
[ec2-user@ip-172-31-90-103 yum.repos.d]$ sudo swapoff -a
[ec2-user@ip-172-31-90-103 yum.repos.d]$ echo "net.bridge.bridge-nf-call-iptable
s=1" | sudo tee -a /etc/sysctl.conf
net.bridge.bridge-nf-call-iptables=1
[ec2-user@ip-172-31-90-103 yum.repos.d]$ sudo sysctl -p
net.bridge.bridge-nf-call-iptables = 1
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

5. **Perform this ONLY on the Master machine Initialize the Kubecluster**
   sudo kubeadm init --podnetwork-cidr=10.244.0.0/16

```
[ec2-user@ip-172-31-90-103 yum.repos.d]$ sudo kubeadm init --pod-network-cidr=10
.244.0.0/16 --ignore-preflight-errors=all
I0914 04:12:17.448521   27990 version.go:256] remote version is much newer: v1.3
1.0; falling back to: stable-1.30
[init] Using Kubernetes version: v1.30.4
[preflight] Running pre-flight checks
        [WARNING NumCPU]: the number of available CPUs 1 is less than the requir
ed 2
        [WARNING Mem]: the system RAM (949 MB) is less than the minimum 1700 MB
        [WARNING FileExisting-socat]: socat not found in system path
        [WARNING FileExisting-tc]: tc not found in system path
        [WARNING Service-Kubelet]: kubelet service is not enabled, please run 's
ystemctl enable kubelet.service'
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your inte
rnet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config
 images pull'
W0914 04:12:17.711154   27990 checks.go:844] detected that the sandbox image "re
gistry.k8s.io/pause:3.8" of the container runtime is inconsistent with that used
 by kubeadm.It is recommended to use "registry.k8s.io/pause:3.9" as the CRI sand
box image.
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
```

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

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

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

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

  export KUBECONFIG=/etc/kubernetes/admin.conf

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

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

kubeadm join 172.31.90.103:6443 --token 0zk8w3.xyegkydsy42vfscm \
        --discovery-token-ca-cert-hash sha256:31c672892b19dcb869fc46362d189234128f5bfc302bd41ae8c6078c56173f00
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

- Save the token

```
kubeadm join 172.31.90.103:6443 --token 0zk8w3.xyegkydsy42vfscm \
        --discovery-token-ca-cert-hash
sha256:31c672892b19dcb869fc46362d189234128f5bfc302bd41ae8c6078c56173f00
```

- Copy the mkdir and chown commands from the top and execute them

mkdir -p $HOME/.kube

```
[ec2-user@ip-172-31-90-103 yum.repos.d]$ mkdir -p $HOME/.kube
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

```
[ec2-user@ip-172-31-90-103 yum.repos.d]$ sudo cp -i /etc/kubernetes/admin.conf $
HOME/.kube/config
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

sudo chown $(id -u):$(id -g) $HOME/.kube/config

```
[ec2-user@ip-172-31-90-103 yum.repos.d]$
sudo chown $(id -u):$(id -g) $HOME/.kube/config
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

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

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

```
[ec2-user@ip-172-31-90-103 yum.repos.d]$ kubectl apply -f https://raw.githubuser
content.com/coreos/flannel/master/Documentation/kube-flannel.yml
namespace/kube-flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

6. **Perform this ONLY on the worker machines**

- Paste the below command on all 2 worker machines

  sudo yum install iproute-tc -y
  sudo systemctl enable kubelet
  sudo systemctl restart kubelet

- Now use the token from earlier to join into worker instances

```
kubeadm join 172.31.90.103:6443 --token 0zk8w3.xyegkydsy42vfscm \
        --discovery-token-ca-cert-hash
sha256:31c672892b19dcb869fc46362d189234128f5bfc302bd41ae8c6078c56173f00
```

- **kubectl get nodes** to check whether master and worker nodes are connected successfully

```
[ec2-user@ip-54-211-147-10 docker]$ kubectl get nodes
NAME                            STATUS   ROLES           AGE     VERSION
ip-172-31-90-103.ec2.internal   Ready    control-plane   3m21s   v1.30.5
```

Conclusion:

An EC2 instance was created on AWS Linux, and Docker, Kubernetes, Kubelet, Kubeadm, and Kubectl were installed. Kubernetes was initialized on the master node, which provided a token for connecting the master and worker nodes. On the slave node, `iproute` was installed, and Kubelet was enabled and restarted. However, there was an issue with joining the slave node to the cluster, resulting in only the master node being listed when running `kubectl get nodes`

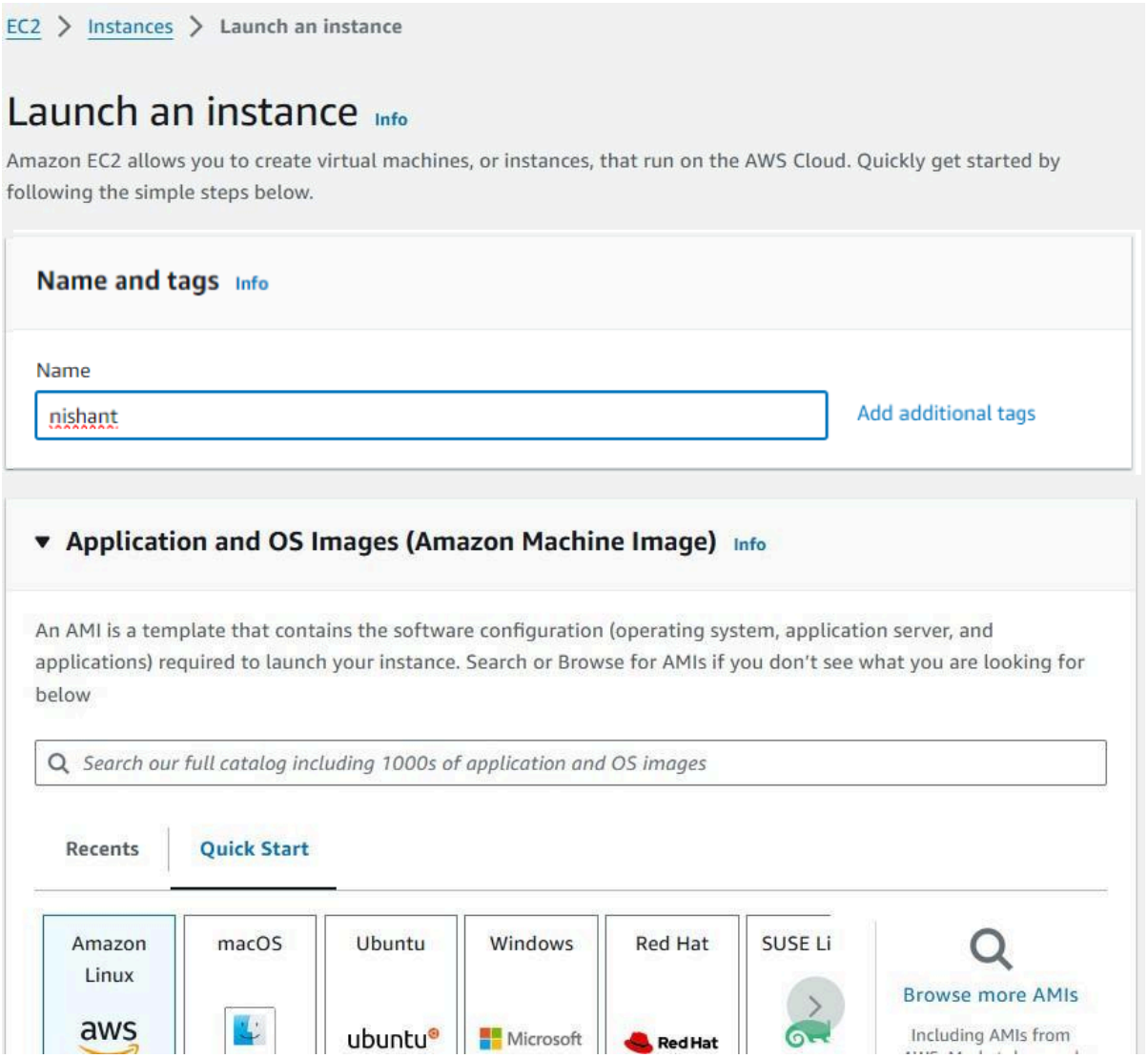
Name: Nishant S Khetal **D15C** **Roll No: 24**

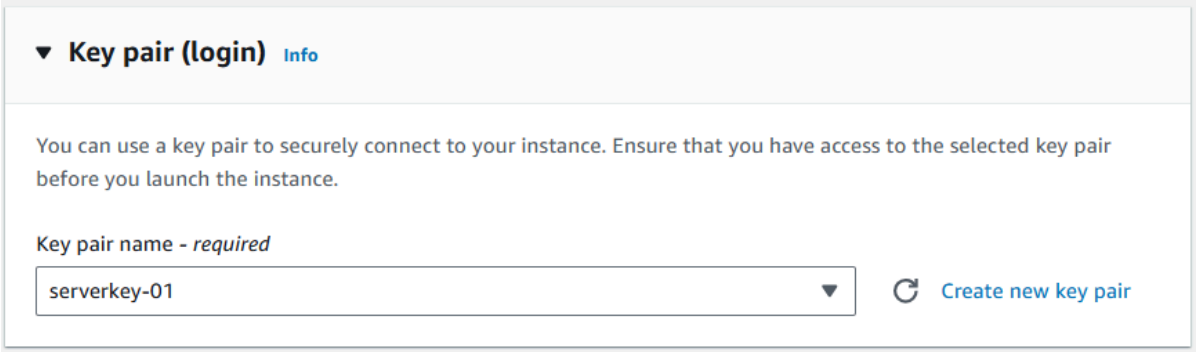
Experiment No. 4

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

Steps:

1. **Create an EC2 Linux Instances on AWS.**





Network settings Info

Edit

Network Info

vpc-0404d393731afabc3

Subnet Info

No preference (Default subnet in any availability zone)

Auto-assign public IP Info

Enable

Additional charges apply when outside of free tier allowance

Firewall (security groups) Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

- (•) Create security group
- ( ) Select existing security group

We'll create a new security group called '**launch-wizard-4**' with the following rules:

- [x] Allow SSH traffic from
  Helps you connect to your instance
  Anywhere 0.0.0.0/0
- [x] Allow HTTPS traffic from the internet
  To set up an endpoint, for example when creating a web server
- [x] Allow HTTP traffic from the internet
  To set up an endpoint, for example when creating a web server

⚠ Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only. ×

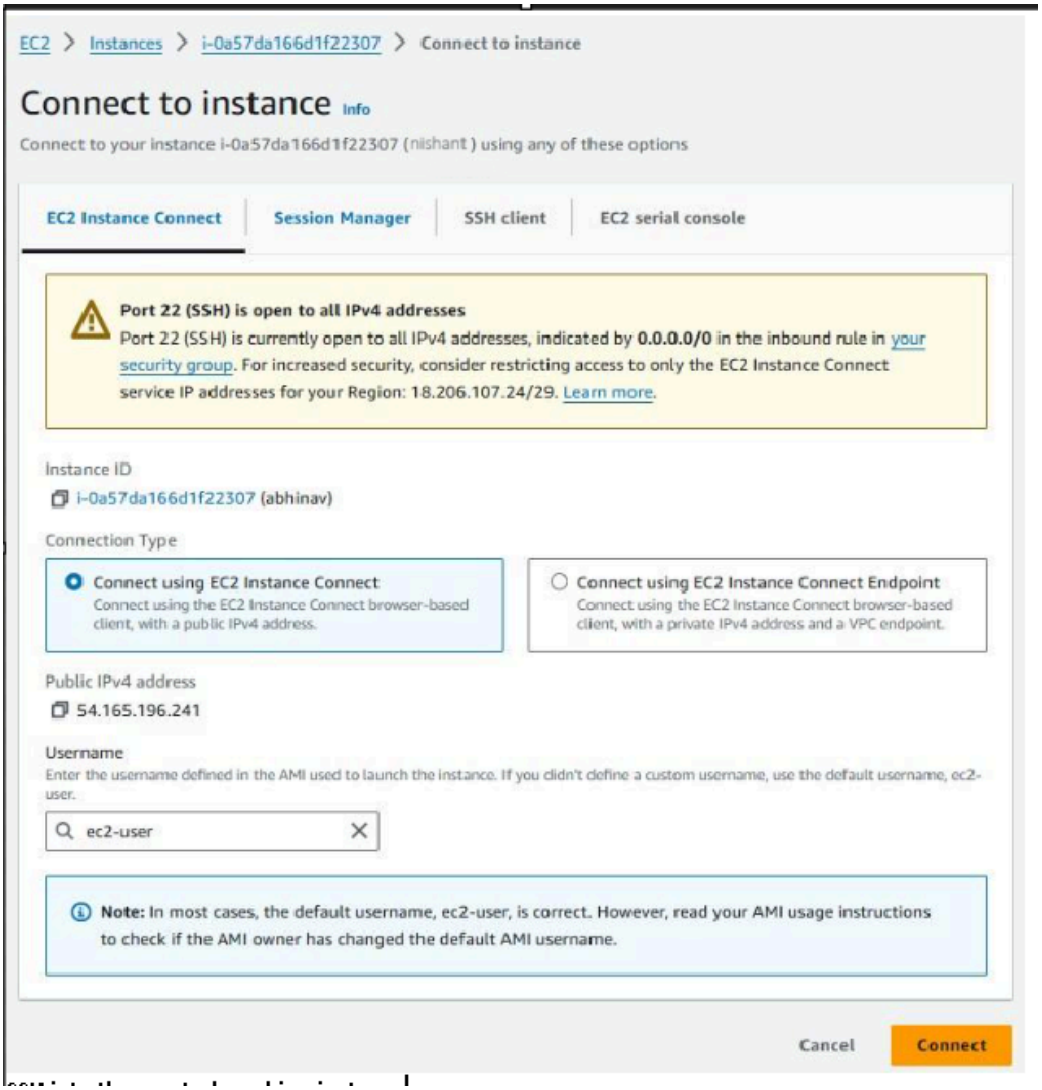
Instances (2) Info

Connect | Instance state | Actions | Launch instances

Find instance by attribute or tag (case-sensitive) | All states | 1

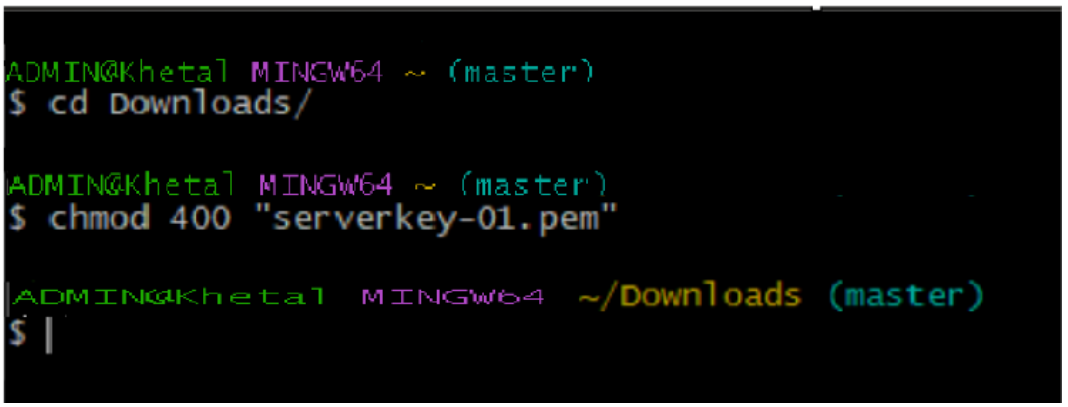
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public
Nishant Khetal	i-0341f2e761db28aec	Running	t2.micro	2/2 checks passe	View alarms +	us-east-1c	ec2-52-
Nishant	i-0689f78090ca539a9	Running	t2.micro	2/2 checks passe	View alarms +	us-east-1c	ec2-34-

2. **Then click on Id of that instance then click on connect**



3. **SSH into the created machine instance**

   - Give permissions to the current user to the downloaded pem file using -
     chmod 400 <security_filename.pem>

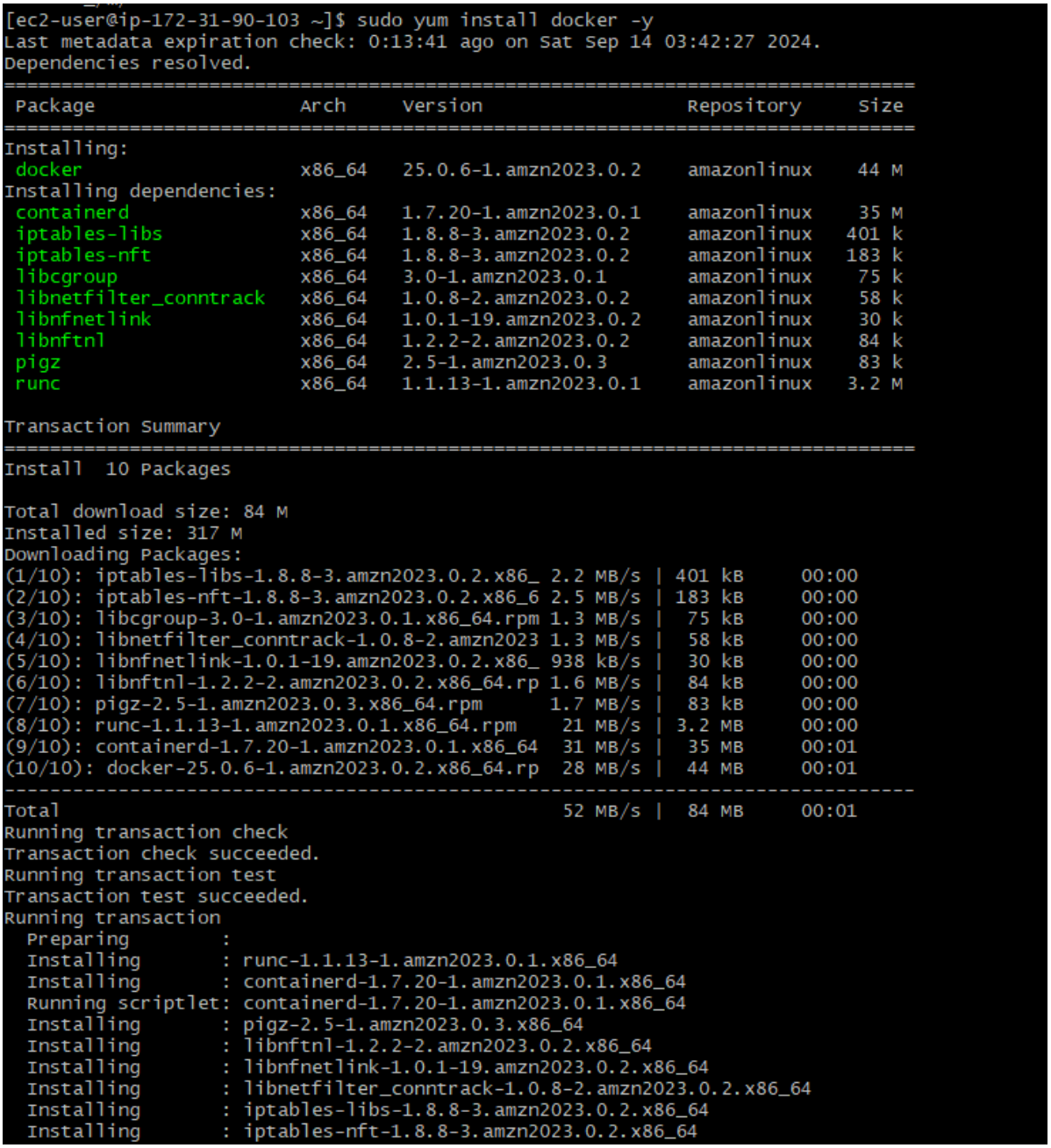


- Ssh using –

  ssh -i (keyname).pem (username)@(public ipv4 dns address)
  where keyname is name of the key you created. (server-01.pem). Other details can be found on the Instance dashboard.

4. Installation Of Docke007

- **sudo yum install docker -y**



```
Installed:
  containerd-1.7.20-1.amzn2023.0.1.x86_64              docker-25.0.6-1.amzn2023.0.2.x86_64
  libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64   libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64

Complete!
[ec2-user@ip-172-31-90-103 ~]$
```

- Configure cgroup in a daemon.json

  (this can be done by creating the file and using **nano** text editor)

  {

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

  "log-driver": "json-file",

  "log-opts": {

  "max-size": "100m"

  },

  "storage-driver": "overlay2"

  }

```
[ec2-user@ip-172-31-90-103 docker]$ sudo nano daemon.json
[ec2-user@ip-172-31-90-103 docker]$
```

```
ec2-user@ip-172-31-90-103:/etc/docker
  GNU nano 5.8                        daemon.json                        Modified
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line
```

- Enable and start docker and also load the daemon.json

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

```
[ec2-user@ip-172-31-90-103 docker]$ sudo systemctl enable docker
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /us
r/lib/systemd/system/docker.service.
[ec2-user@ip-172-31-90-103 docker]$ sudo systemctl daemon-reload
[ec2-user@ip-172-31-90-103 docker]$
sudo systemctl restart docker
[ec2-user@ip-172-31-90-103 docker]$
```

- Check if docker is installed

```
[ec2-user@ip-172-31-90-103 docker]$ docker --version
Docker version 25.0.5, build 5dc9bcc
[ec2-user@ip-172-31-90-103 docker]$
```

5. Install Kubernetes

- SELinux needs to be disabled before configuring kubelet

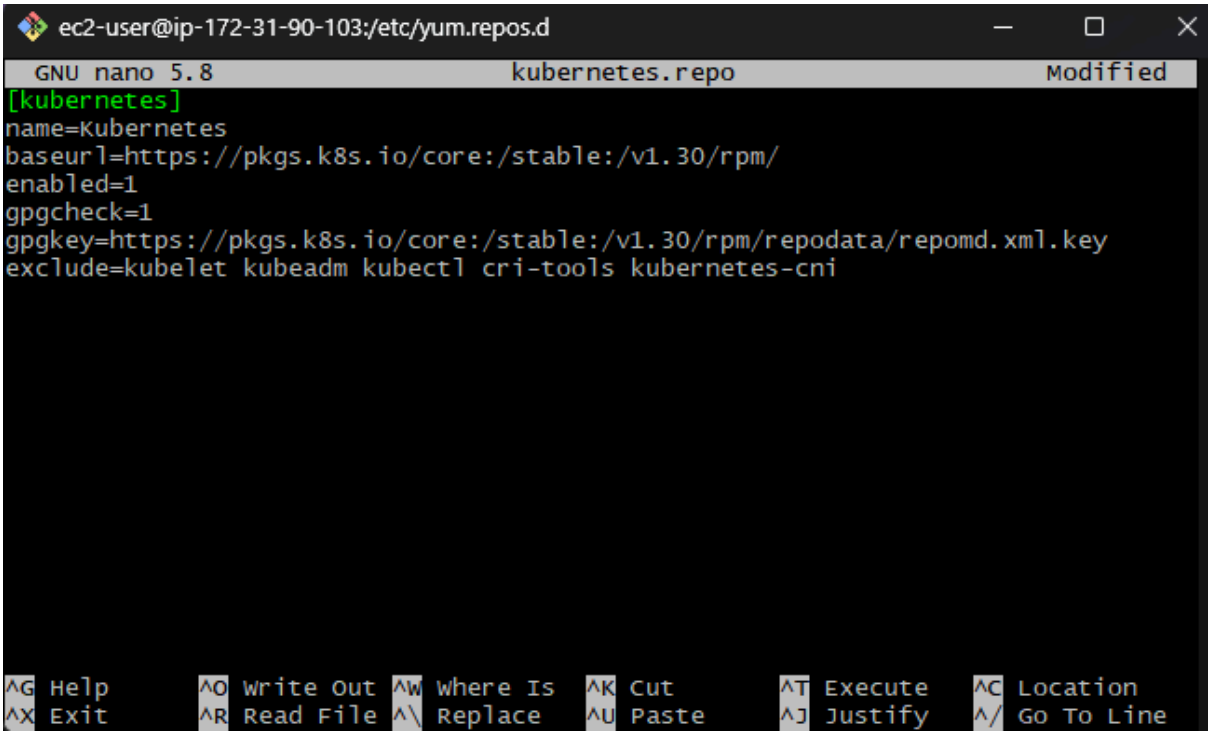
  sudo setenforce 0
  sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config

```
[ec2-user@ip-172-31-90-103 docker]$ sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
[ec2-user@ip-172-31-90-103 docker]$
```

- Add Kubernetes using the repo

  (this is done by creating **kubernetes.repo** file in **/etc/yum.repos.d** and configuring it using **nano** editor)
  [kubernetes]
  name=Kubernetes
  baseurl=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/
  enabled=1
  gpgcheck=1
  gpgkey=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/repodata/repomd.xml.key
  exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni

```
[ec2-user@ip-172-31-90-103 docker]$ cd /etc/yum.repos.d/
[ec2-user@ip-172-31-90-103 yum.repos.d]$ ls
amazonlinux.repo  kernel-livepatch.repo
[ec2-user@ip-172-31-90-103 yum.repos.d]$ sudo nano kubernetes.repo
[ec2-user@ip-172-31-90-103 yum.repos.d]$ ls
amazonlinux.repo  kernel-livepatch.repo  kubernetes.repo
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```



- Update packages list using **sudo yum update**

```
[ec2-user@ip-172-31-90-103 yum.repos.d]$ sudo yum update
Kubernetes                                      125 kB/s |  17 kB     00:00
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

- Install kubelet kubeadm kubectl

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

```
[ec2-user@ip-172-31-90-103 yum.repos.d]$ sudo yum install -y kubelet kubeadm kub
ectl --disableexcludes=kubernetes
Last metadata expiration check: 0:00:42 ago on Sat Sep 14 04:08:20 2024.
Dependencies resolved.
================================================================================
 Package                 Arch     Version                  Repository     Size
================================================================================
Installing:
 kubeadm                 x86_64   1.30.5-150500.1.1        kubernetes     10 M
 kubectl                 x86_64   1.30.5-150500.1.1        kubernetes     10 M
 kubelet                 x86_64   1.30.5-150500.1.1        kubernetes     17 M
Installing dependencies:
 conntrack-tools         x86_64   1.4.6-2.amzn2023.0.2     amazonlinux   208 k
 cri-tools               x86_64   1.30.1-150500.1.1        kubernetes    8.6 M
 kubernetes-cni          x86_64   1.4.0-150500.1.1         kubernetes    6.7 M
 libnetfilter_cthelper   x86_64   1.0.0-21.amzn2023.0.2    amazonlinux    24 k
 libnetfilter_cttimeout  x86_64   1.0.0-19.amzn2023.0.2    amazonlinux    24 k
 libnetfilter_queue      x86_64   1.0.5-2.amzn2023.0.2     amazonlinux    30 k

Transaction Summary
================================================================================
Install  9 Packages

Total download size: 53 M
Installed size: 292 M
Downloading Packages:
(1/9): libnetfilter_cttimeout-1.0.0-19.amzn2023 448 kB/s |  24 kB     00:00
(2/9): libnetfilter_cthelper-1.0.0-21.amzn2023. 409 kB/s |  24 kB     00:00
(3/9): libnetfilter_queue-1.0.5-2.amzn2023.0.2. 1.5 MB/s |  30 kB     00:00
(4/9): conntrack-tools-1.4.6-2.amzn2023.0.2.x86 1.8 MB/s | 208 kB     00:00
(5/9): cri-tools-1.30.1-150500.1.1.x86_64.rpm    28 MB/s | 8.6 MB     00:00
(6/9): kubectl-1.30.5-150500.1.1.x86_64.rpm      23 MB/s |  10 MB     00:00
(7/9): kubeadm-1.30.5-150500.1.1.x86_64.rpm      18 MB/s |  10 MB     00:00
(8/9): kubelet-1.30.5-150500.1.1.x86_64.rpm      37 MB/s |  17 MB     00:00
(9/9): kubernetes-cni-1.4.0-150500.1.1.x86_64.r  20 MB/s | 6.7 MB     00:00
--------------------------------------------------------------------------------
Total                                            56 MB/s |  53 MB     00:00
```

```
Installed:
  conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64
  cri-tools-1.30.1-150500.1.1.x86_64
  kubeadm-1.30.5-150500.1.1.x86_64
  kubectl-1.30.5-150500.1.1.x86_64
  kubelet-1.30.5-150500.1.1.x86_64
  kubernetes-cni-1.4.0-150500.1.1.x86_64
  libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64
  libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64
  libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64

Complete!
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

- After installing Kubernetes, we need to configure internet options to allow bridging.

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

```
[ec2-user@ip-172-31-90-103 yum.repos.d]$ . sudo swapoff -a
-bash: sudo: No such file or directory
[ec2-user@ip-172-31-90-103 yum.repos.d]$ sudo swapoff -a
[ec2-user@ip-172-31-90-103 yum.repos.d]$ echo "net.bridge.bridge-nf-call-iptable
s=1" | sudo tee -a /etc/sysctl.conf
net.bridge.bridge-nf-call-iptables=1
[ec2-user@ip-172-31-90-103 yum.repos.d]$ sudo sysctl -p
net.bridge.bridge-nf-call-iptables = 1
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

6. **Initialize the Kubecluster**

sudo kubeadm init --podnetwork-cidr=10.244.0.0/16

```
[ec2-user@ip-172-31-90-103 yum.repos.d]$ sudo kubeadm init --pod-network-cidr=10
.244.0.0/16 --ignore-preflight-errors=all
I0914 04:12:17.448521   27990 version.go:256] remote version is much newer: v1.3
1.0; falling back to: stable-1.30
[init] Using Kubernetes version: v1.30.4
[preflight] Running pre-flight checks
        [WARNING NumCPU]: the number of available CPUs 1 is less than the requir
ed 2
        [WARNING Mem]: the system RAM (949 MB) is less than the minimum 1700 MB
        [WARNING FileExisting-socat]: socat not found in system path
        [WARNING FileExisting-tc]: tc not found in system path
        [WARNING Service-Kubelet]: kubelet service is not enabled, please run 's
ystemctl enable kubelet.service'
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your inte
rnet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config
 images pull'
W0914 04:12:17.711154   27990 checks.go:844] detected that the sandbox image "re
gistry.k8s.io/pause:3.8" of the container runtime is inconsistent with that used
 by kubeadm.It is recommended to use "registry.k8s.io/pause:3.9" as the CRI sand
box image.
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
```

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

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

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

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

  export KUBECONFIG=/etc/kubernetes/admin.conf

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

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

kubeadm join 172.31.90.103:6443 --token 0zk8w3.xyegkydsy42vfscm \
        --discovery-token-ca-cert-hash sha256:31c672892b19dcb869fc46362d189234128f5bfc302bd41ae8c6078c56173f00
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

- Save the token

```
kubeadm join 172.31.90.103:6443 --token 0zk8w3.xyegkydsy42vfscm \
        --discovery-token-ca-cert-hash
sha256:31c672892b19dcb869fc46362d189234128f5bfc302bd41ae8c6078c56173f00
```

- Copy the mkdir and chown commands from the top and execute them

mkdir -p $HOME/.kube

```
[ec2-user@ip-172-31-90-103 yum.repos.d]$ mkdir -p $HOME/.kube
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

```
[ec2-user@ip-172-31-90-103 yum.repos.d]$ sudo cp -i /etc/kubernetes/admin.conf $
HOME/.kube/config
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

sudo chown $(id -u):$(id -g) $HOME/.kube/config

```
[ec2-user@ip-172-31-90-103 yum.repos.d]$
sudo chown $(id -u):$(id -g) $HOME/.kube/config
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

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

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

```
[ec2-user@ip-172-31-90-103 yum.repos.d]$ kubectl apply -f https://raw.githubuser
content.com/coreos/flannel/master/Documentation/kube-flannel.yml
namespace/kube-flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

7. Deploy nginx server

- Apply deployment using this following command:

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

```
[ec2-user@ip-172-31-90-103 docker]$ kubectl apply -f https://k8s.io/examples/pods/s
imple-pod.yaml
pod/nginx created
```

- use **kubectl get nodes** to check whether the pod gets created or not

```
[ec2-user@ip-172-31-90-103 docker]$ kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
nginx   0/1     Pending   0          12s
```

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

```
[ec2-user@ip-172-31-90-103 docker]$ kubectl describe pod nginx
Name:             nginx
Namespace:        default
Priority:         0
Service Account:  default
Node:             <none>
Labels:           <none>
Annotations:      <none>
Status:           Pending
IP:
IPs:              <none>
Containers:
  nginx:
    Image:        nginx:1.14.2
    Port:         80/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-k4lj6 (ro)
    Type:                     Projected (a volume that contains injected data from m
ultiple sources)
    TokenExpirationSeconds:   3607
    ConfigMapName:            kube-root-ca.crt
    ConfigMapOptional:        <nil>
    DownwardAPI:              true
QoS Class:                    BestEffort
Node-Selectors:               <none>
Tolerations:                  node.kubernetes.io/not-ready:NoExecute op=Exists for 3
00s
                              node.kubernetes.io/unreachable:NoExecute op=Exists for
 300s
Events:
  Type     Reason            Age   From               Message
  ----     ------            ----  ----               -------
  Warning  FailedScheduling  7s    default-scheduler  0/1 nodes are available: 1 no
de(s) had untolerated taint {node-role.kubernetes.io/control-plane: }. preemption:
0/1 nodes are available: 1 Preemption is not helpful for scheduling.
```

- check pod status

```
[ec2-user@ip-172-31-90-103 ~]$ kubectl get pods
NAME    READY   STATUS    RESTARTS      AGE
nginx   1/1     Running   1 (6s ago)    90s
[ec2-user@ip-172-31-90-103 ~]$ kubectl port-forward nginx 8081:80
Forwarding from 127.0.0.1:8081 -> 80
Forwarding from [::1]:8081 -> 80
```

8. **Verify your deployment**

- Open up a new terminal and ssh to your EC2 instance. Then, use this curl command to check if the Nginx server is running. **curl --head http://127.0.0.1:8080** If the response is 200 OK and you can see the Nginx server name, your deployment was successful. We have successfully deployed our Nginx server on our EC2 instance.

```
[ec2-user@172-31-90-103 ~]$ curl --head http://127.0.0.1:8080
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Linux)
Date: Sat, 14 Sep 2024 12:31:53 GMT
Content-Type: text/html
Content-Length: 612
Connection: keep-alive
```

Conclusion:

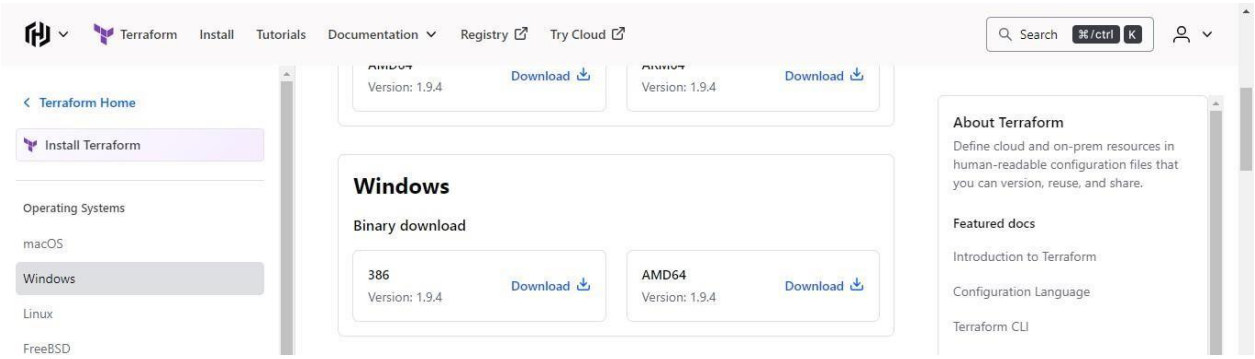
An AWS EC2 Linux instance was set up, and Docker and Kubernetes were installed. Kubernetes was initialized successfully, and the required commands were executed. Flannel was installed as a networking plugin. Although there was an initial error with the Nginx deployment, it was eventually deployed successfully using the `simple-pod.yml` file and accessed via localhost on port 8080.

Name: Nishant S Khetal **D15C** **Roll No: 24**

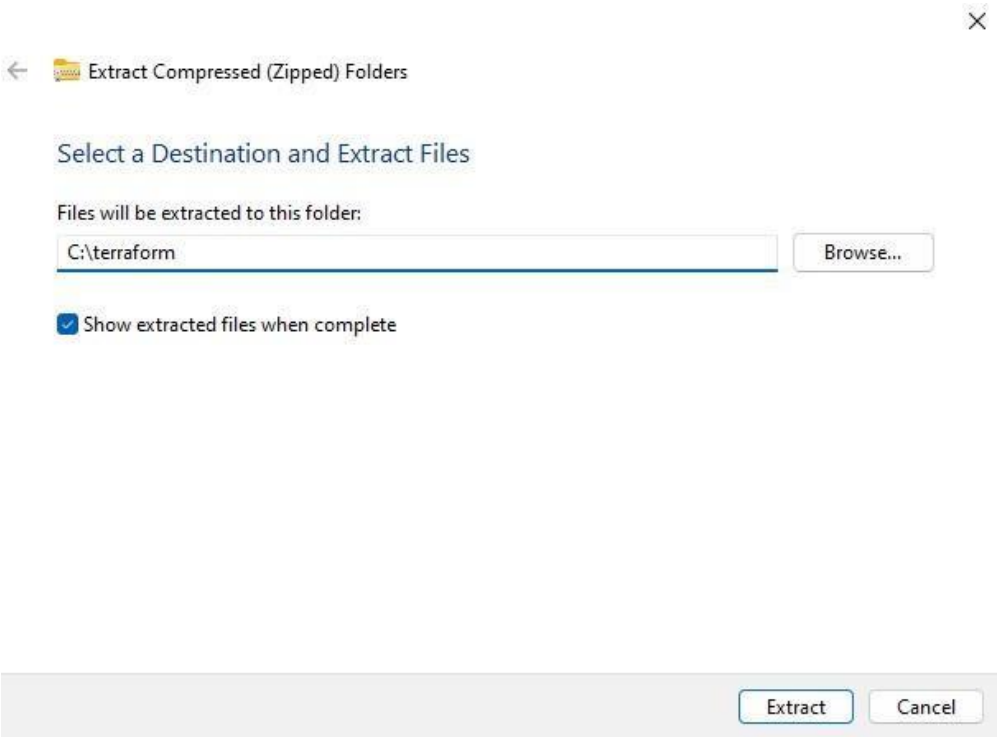
Experiment No. 5

Installation of Terraform

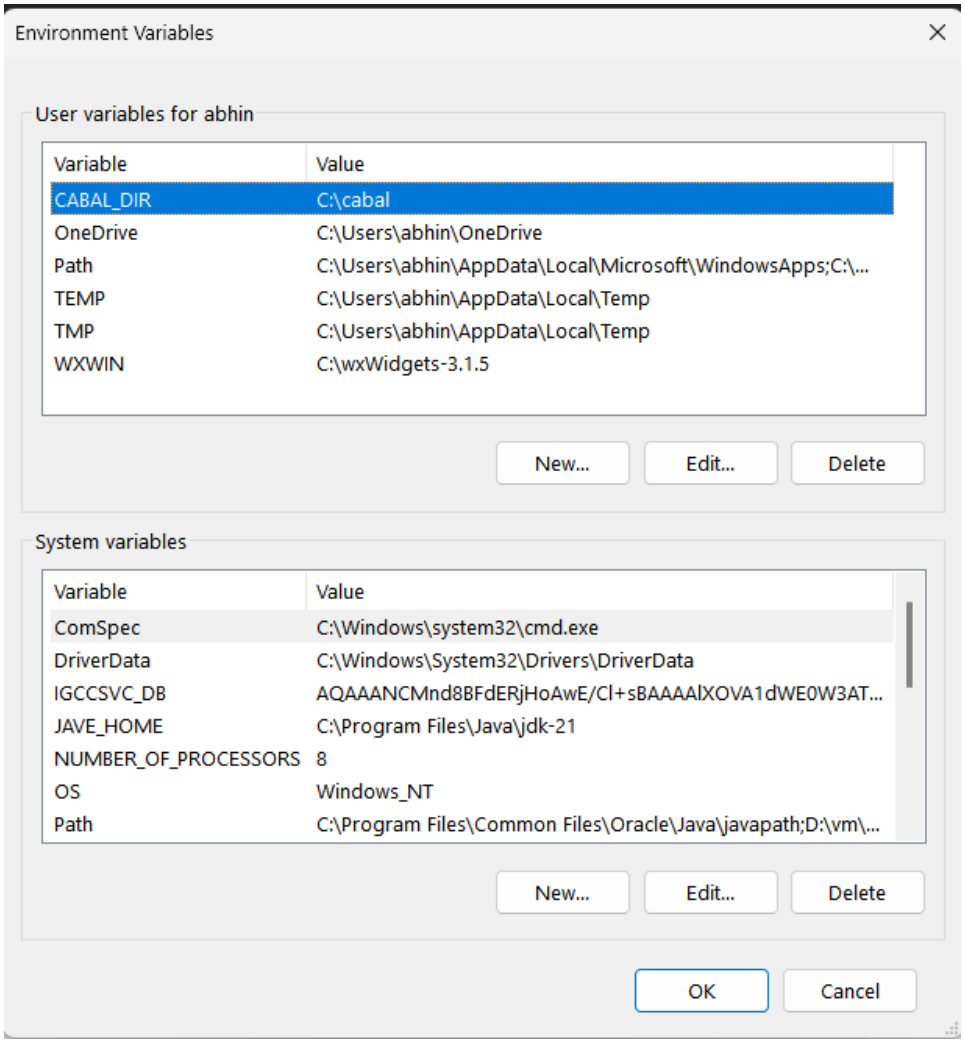
Step 1: To install Terraform, go to the official Terraform website linked below. Navigate to the Downloads section, choose Windows, and download the 64-bit version for your system.
Website: https://www.terraform.io/downloads.html



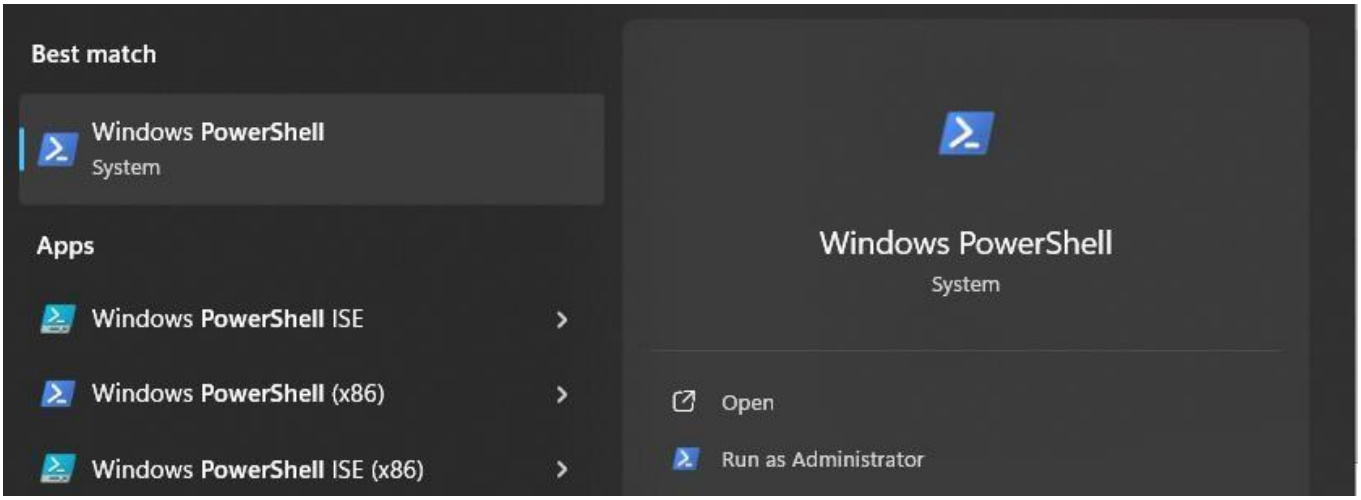
Step 2: Extract the downloaded `Terraform.exe` file to the `C:\Terraform` directory on your computer.



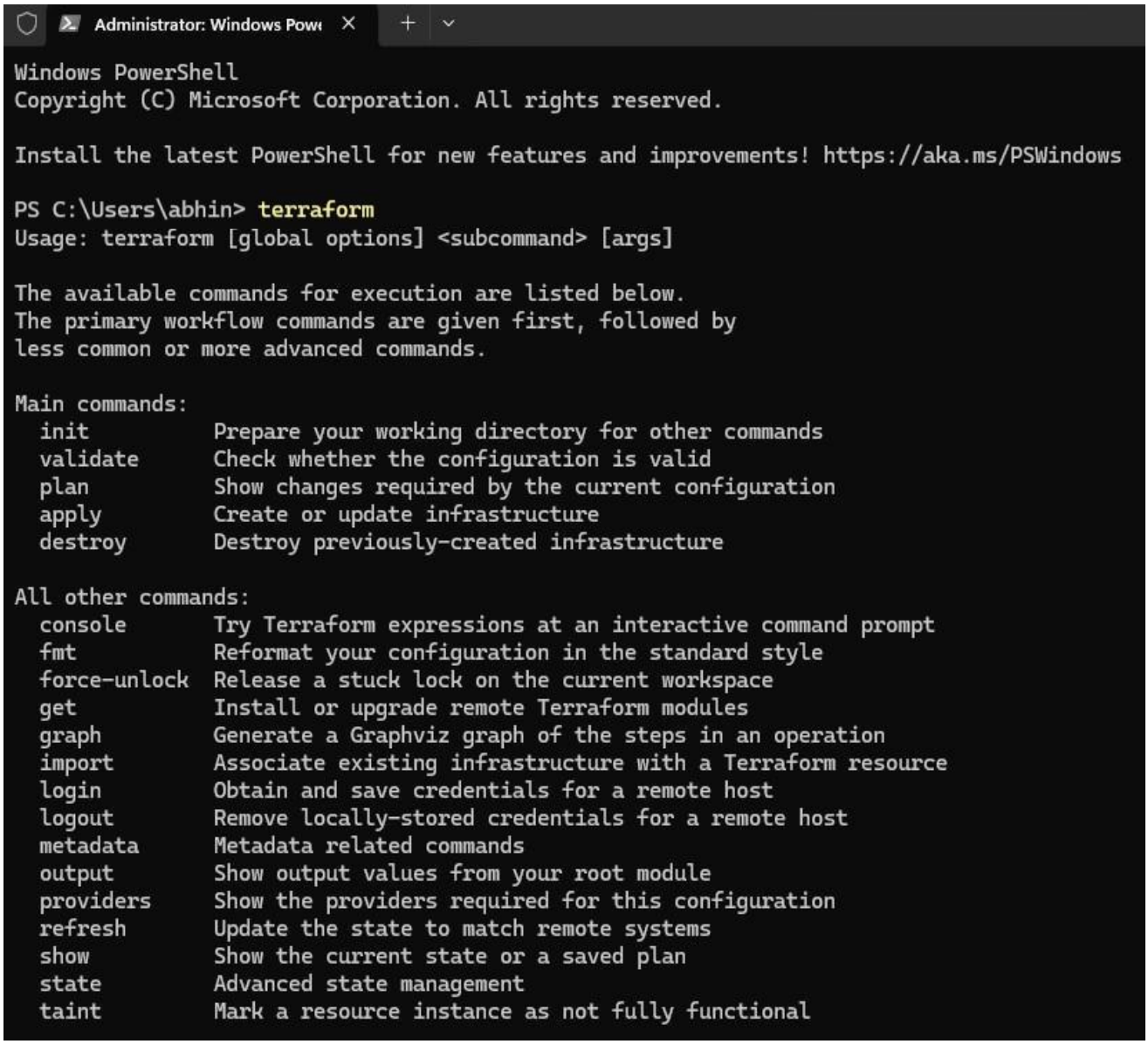
Step 3: Configure the system path for Terraform in the Environment Variables.



Step 4: Launch Windows PowerShell with administrator privileges.



Step 5: Run the command `terraform` to confirm its functionality. If any errors occur, review and update the Terraform path in your environment variables as needed.



Name: Nishant S Khetal **D15C** **Roll No: 24**

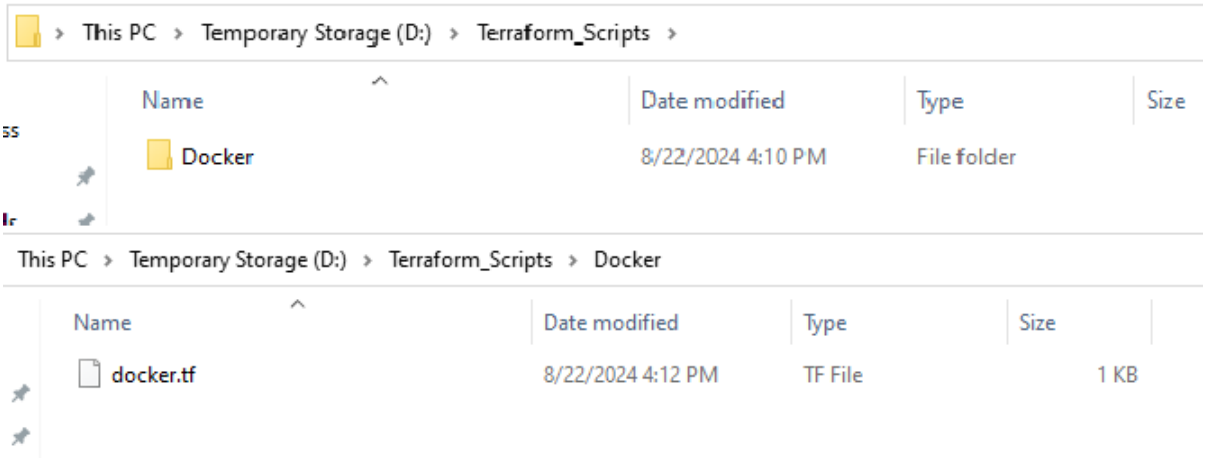
Experiment No.6

1. Creating a docker image using terraform

```
C:\Users\nishant>docker --version
Docker version 26.1.3, build b72abbb

C:\Users\nishant>docker

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

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

Management Commands:
  builder     Manage builds
  buildx*     Docker Buildx
  compose*    Docker Compose
  container   Manage containers
  context     Manage contexts
```

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

3. Create a new folder named ‘Docker’ in the ‘TerraformScripts’ folder. Then create a new docker.tf file using a text editor and write the following contents into it to create a Ubuntu Linux container.



docker.tf - Notepad

File Edit Format View Help

```
terraform {
        required_providers {
                docker = {
                source = "kreuzwerker/docker"
                version = "2.21.0"
                }
        }
}
provider "docker" {
host = "npipe:////.//pipe//docker_engine"
}

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

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

4. Execute Terraform Init command to initialize the resources

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

Terraform has been successfully initialized!

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

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

D:\Terraform_Scripts\Docker>
```

5. Execute Terraform plan to see the available resources

```
D:\Terraform_Scripts\Docker>terraform plan

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

Terraform will perform the following actions:

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

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

```
D:\Terraform_Scripts\Docker>terraform apply

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

Terraform will perform the following actions:

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

      + healthcheck (known after apply)

      + labels (known after apply)
    }

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

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

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

  Enter a value: yes

docker_image.ubuntu: Creating...
docker_image.ubuntu: Still creating... [19s elapsed] docker_image.ubuntu: Still creating... (20s elapsed) docker_image.
ubuntu: Still creating... [30s elapsed]
docker_image.ubuntu: Creation complete after 30s [id=sha256:263966596d42ad38ae9914716692777ba9ff8779a62ad93a74fe82e3e1f
ubuntu:latest] docker_container.foo: Creating...
```

7. Check Docker images, Before and After Executing Apply step

```
D:\Terraform_Scripts\Docker>docker images
REPOSITORY                                  TAG                                 IMAGE ID       CREATED       SIZE
mcr.microsoft.com/dotnet/framework/aspnet   4.8-windowsservercore-ltsc2022      0b1ef1176a57   6 weeks ago   5.43GB
mcr.microsoft.com/dotnet/framework/sdk      4.8-windowsservercore-ltsc2022      c3f8c2735565   6 weeks ago   9.04GB
mcr.microsoft.com/dotnet/framework/runtime  4.8-windowsservercore-ltsc2022      e69ea8a5ec1b   6 weeks ago   5.1GB
mcr.microsoft.com/windows/servercore        ltsc2022                            e60f47e635b7   7 weeks ago   4.84GB
mcr.microsoft.com/windows/nanoserver        ltsc2022                            f0ca29645006   7 weeks ago   292MB

D:\Terraform_Scripts\Docker>
```

```
D:\Terraform_Scripts\Docker>docker images
REPOSITORY                                  TAG                                 IMAGE ID       CREATED          SIZE
mcr.microsoft.com/dotnet/framework/aspnet   4.8-windowsservercore-ltsc2022      0b1ef1176a57   6 weeks ago      5.43GB
mcr.microsoft.com/dotnet/framework/sdk      4.8-windowsservercore-ltsc2022      c3f8c2735565   6 weeks ago      9.04GB
mcr.microsoft.com/dotnet/framework/runtime  4.8-windowsservercore-ltsc2022      e69ea8a5ec1b   6 weeks ago      5.1GB
mcr.microsoft.com/windows/servercore        ltsc2022                            e60f47e635b7   7 weeks ago      4.84GB
mcr.microsoft.com/windows/nanoserver        ltsc2022                            f0ca29645006   7 weeks ago      292MB
ubuntu                                      Latest                              2dc39ba859dc   2 minutes ago    77.8MB
```

8. Execute Terraform destroy to delete the configuration, which will automatically delete the Ubuntu Container.

```
D:\Terraform_Scripts\Docker>terraform destroy
docker_image.ubuntu: Refreshing state... [idrsha256:2dc29b50dcd2d30101475692777ba087762d92de0221fubuntu:latest)
Terrafore used the selected providers to generate the following execution plan. Resource actions are indicated with the following sy
mbols:
    destroy
Terraform will perform the following actions:

  #  docker_image.ubuntu will be destroyed
       resource "docker_image" "ubuntu" {
           - id           = "sha256:2dc39b859dc2ade8ae30147b5692777ba9ff9779a62ad93a78de82e3elfubuntu:latest" -> null
           - image_id     = "sha256:2dc39b859dc2ade8ae30147b5692777ba9ff9779a62ad93a78de82e3elf" -> null
           - Latest       = "sha256:2dc39b859dc2ade8ae30147b5692777ba9ff9779a62ad93a78de82e3elf" -> null
           - name         = "ubuntu:latest" -> null
           - repo digest  = "ubuntu@sha256:204a3d7bb4d7723452be3923b06cd7043704030041c83c#7856c1" -> null
       }

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

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is so unde. Only 'yes' will be accepted to confirm.

  Enter a value: yes

docker image.ubuntu: Destroying... [id-sha256:2de99b59cd42ade83814765692777ba5ff8779a62ad93ad62e3e1fubuntu:latest]
docker image.ubuntu: Destruction complete after is

Destroy complete! Resources: 1 destroyed.
```

9. Check Docker images, After Executing Destroy step

```
D:\Terraform_Scripts\Docker>docker images
REPOSITORY                                  TAG                             IMAGE ID       CREATED       SIZE
mcr.microsoft.com/dotnet/framework/aspnet   4.8-windowsservercore-ltsc2022  0b1ef1176a57   6 weeks ago   5.43GB
mcr.microsoft.com/dotnet/framework/sdk      4.8-windowsservercore-ltsc2022  c3f8c2735565   6 weeks ago   9.04GB
mcr.microsoft.com/dotnet/framework/runtime  4.8-windowsservercore-ltsc2022  e69ea8a5ec1b   6 weeks ago   5.1GB
mcr.microsoft.com/windows/servercore        ltsc2022                        e60f47e635b7   7 weeks ago   4.84GB
mcr.microsoft.com/windows/nanoserver        ltsc2022                        f0ca29645006   7 weeks ago   292MB

D:\Terraform_Scripts\Docker>
```

Name : Nishant S Khetal **Class: D15C** **Roll No : 24**

Experiment No: 07

Aim: To investigate Static Analysis (SAST) techniques and demonstrate the integration of Jenkins with SonarQube for assessing code quality..

Theory:

Overview of SAST:

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

Challenges Addressed by SAST:

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

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

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

Importance of SAST:

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

Key Steps for Effective SAST:

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

1. **Finalize the tool.** Select a static analysis tool that can perform code reviews of applications written in the programming languages you use. The tool should also be able to comprehend the underlying framework used by your software.

2. **Setup Infrastructure:** This step involves handling the licensing requirements, setting up access control and authorization, and procuring the resources required (e.g., servers and databases) to deploy the tool.

3. **Customize the tool.** Fine-tune the tool to suit the needs of the organization. For example, you might configure it to reduce false positives or find additional security vulnerabilities by writing new rules or updating existing ones. Integrate the tool into the build environment, create dashboards for tracking scan results, and build custom reports.

4. **Prioritize and onboard applications.** Once the tool is ready, onboard your applications. If you have a large number of applications, prioritize the high-risk applications to scan first. Eventually, all your applications should be onboarded and scanned regularly, with application scans synced with release cycles, daily or monthly builds, or code check-ins.

5. **Analyze scan results.** This step involves triaging the results of the scan to remove false positives. Once the set of issues is finalized, they should be tracked and provided to the deployment teams for proper and timely remediation.

6. **Educate developers.** Proper governance ensures that your development teams are employing the scanning tools properly. The software security touchpoints should be present within the SDLC. SAST should be incorporated as part of your application development and deployment process.

Integrating Jenkins with SonarQube:

Prerequisites:

- Installed Jenkins
- Docker setup for SonarQube
- Access to the SonarQube Docker image

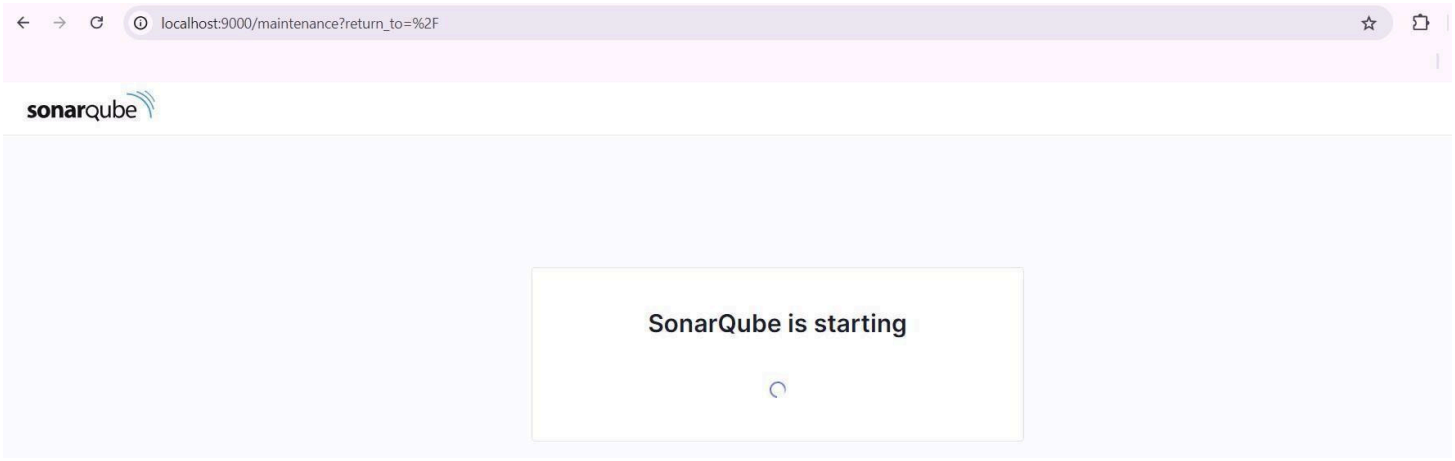
Steps to integrate Jenkins with SonarQube

1. Navigate to Jenkins at localhost:8000.

2. Launch SonarQube using Docker and confirm accessibility at port 9000.

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

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



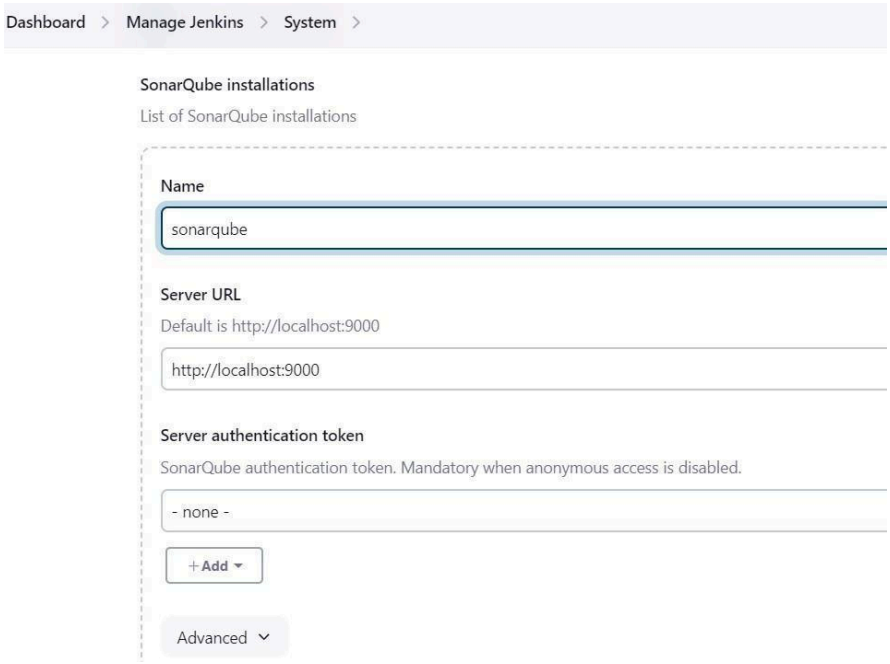
4. Login to SonarQube using username ***admin*** and password ***admin***.

5. In Jenkins, install the SonarQube Scanner plugin to facilitate the analysis.

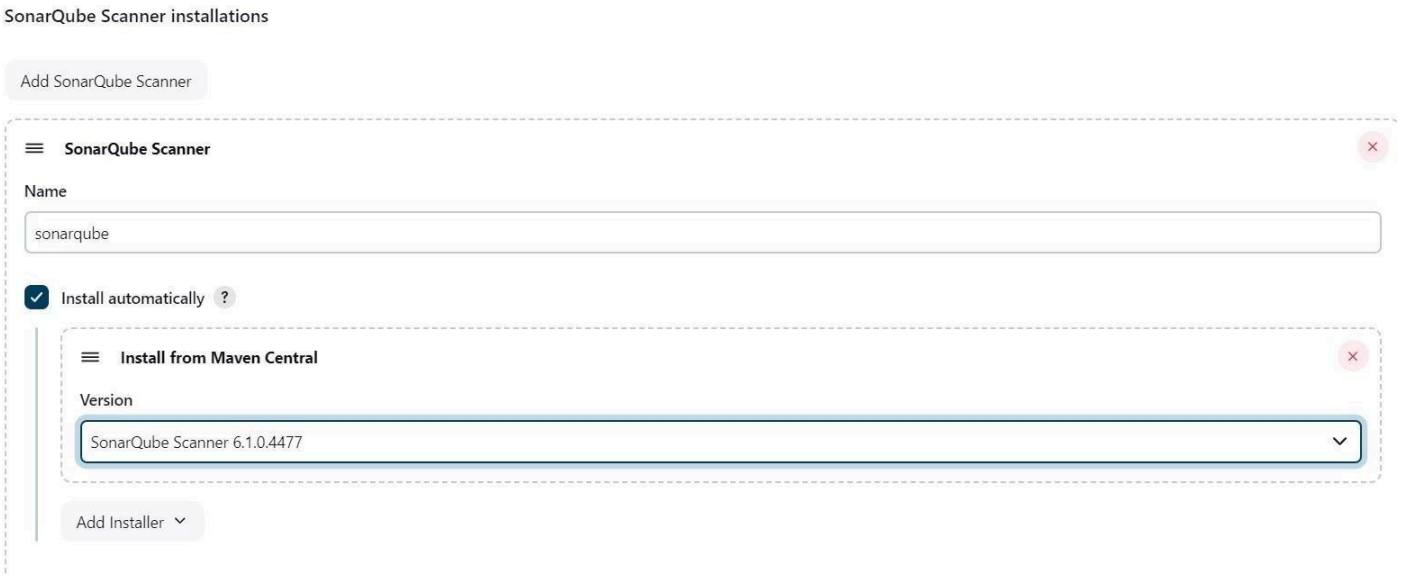
   Setup the project and come back to Jenkins Dashboard.

6. Configure the connection settings for SonarQube within Jenkins, entering credentials

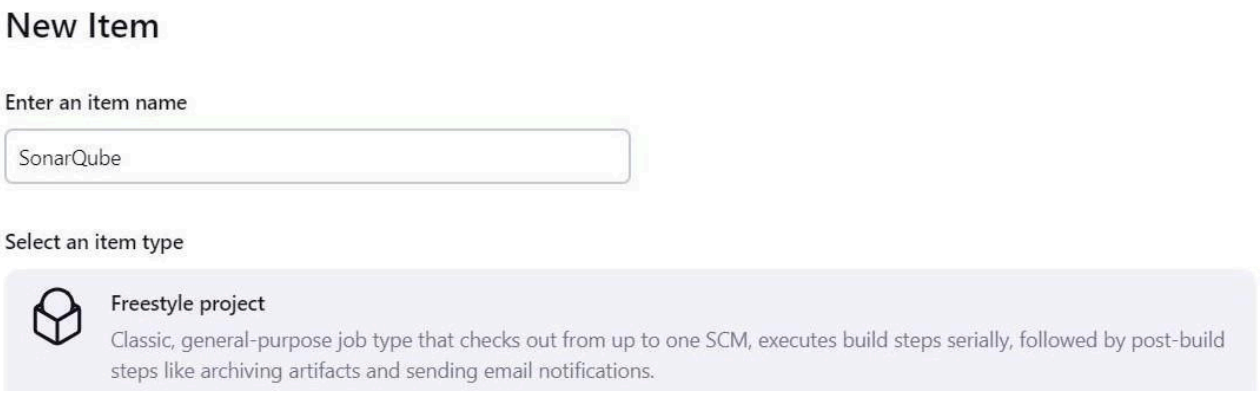
7. Create a new freestyle project in Jenkins and link it to a test repository hosted on GitHub.



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



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

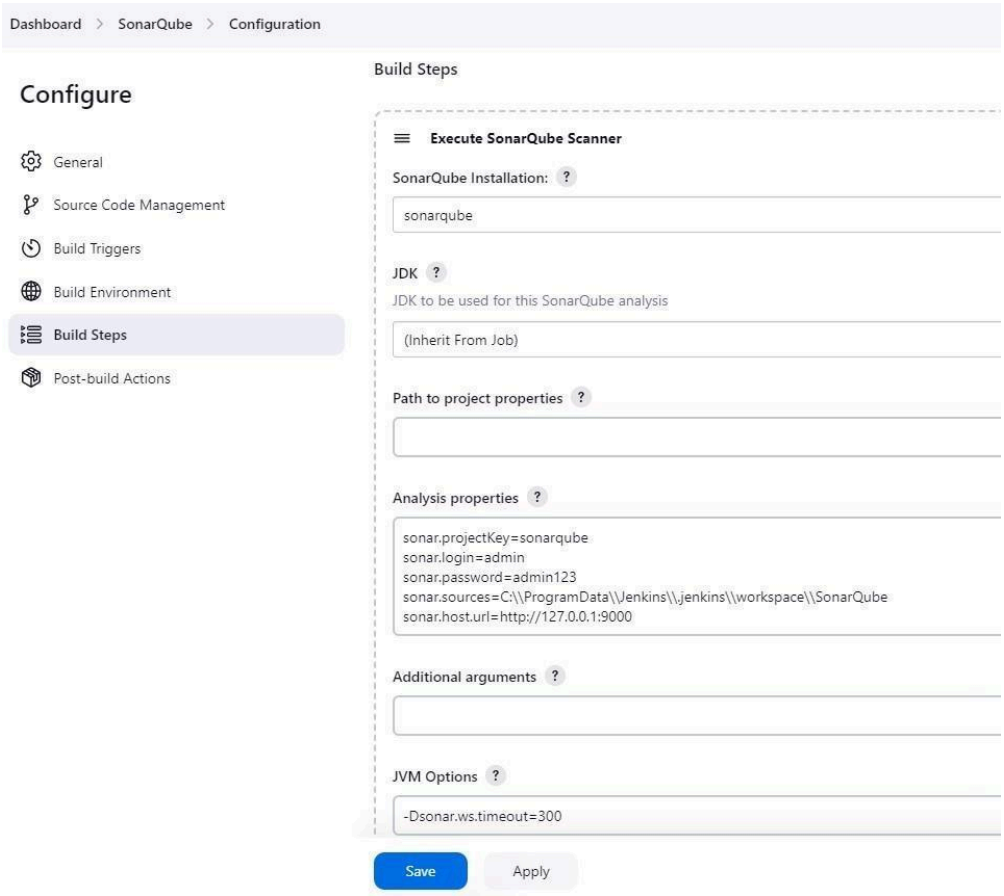


10. Choose this GitHub repository in Source Code Management.

https://github.com/shazforiot/MSBuild_firstproject.git

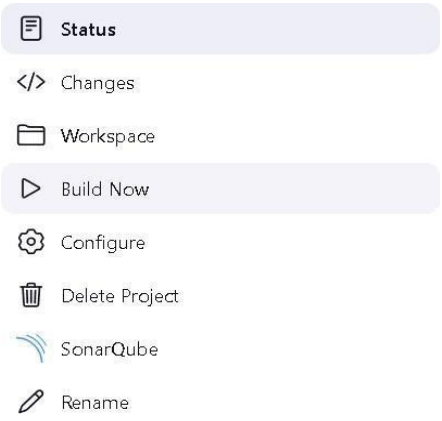
It is a sample hello-world project with no vulnerabilities and issues, just to test the integration.

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

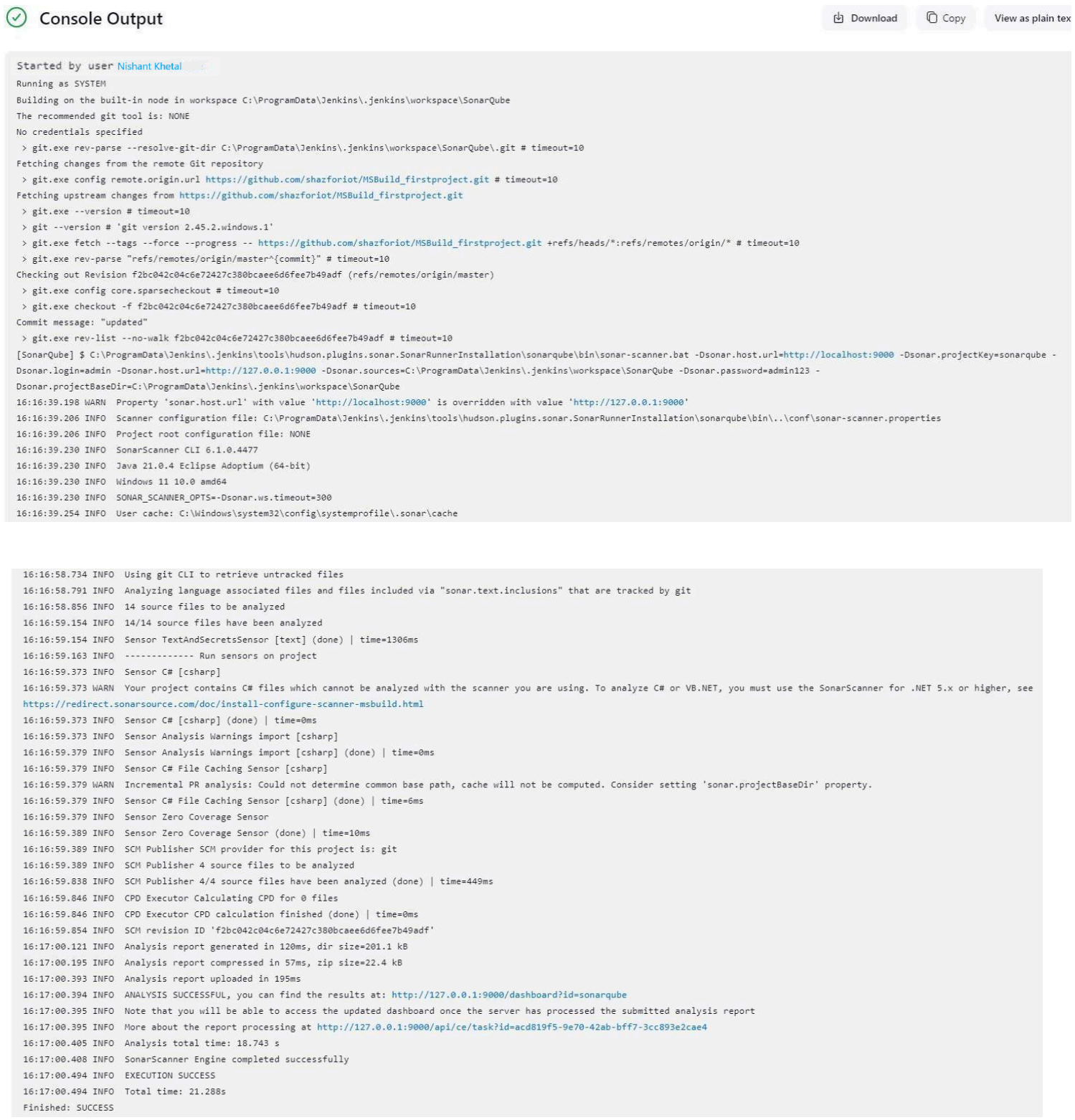


12. Go to http://localhost:9000/<user_name>/permissions and allow Execute Permissions to the Admin user.

13. Run The Build.



Check the console output.

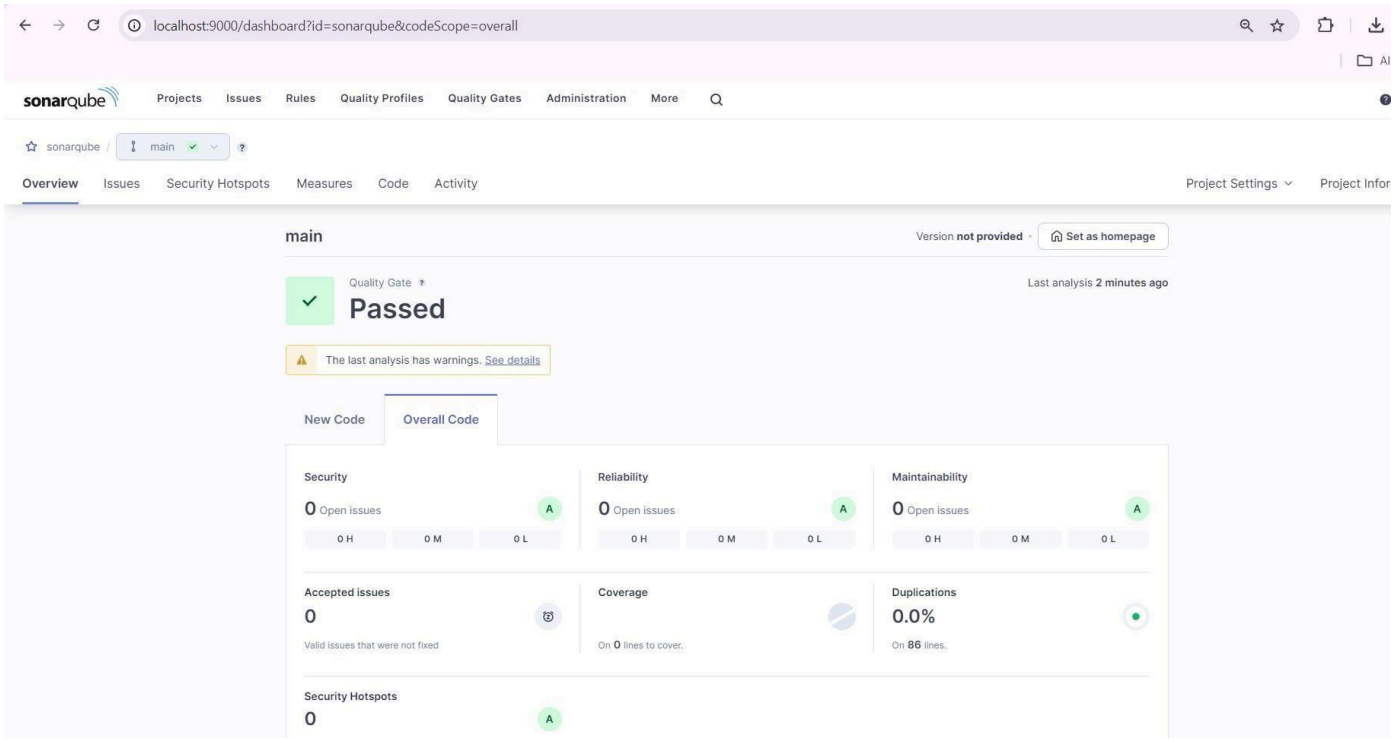


Console Output

Download Copy View as plain tex

```
Started by user Nishant Khetal
Running as SYSTEM
Building on the built-in node in workspace C:\ProgramData\Jenkins\.jenkins\workspace\SonarQube
The recommended git tool is: NONE
No credentials specified
 > git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\SonarQube\.git # timeout=10
Fetching changes from the remote Git repository
 > git.exe config remote.origin.url https://github.com/shazforiot/MSBuild_firstproject.git # timeout=10
Fetching upstream changes from https://github.com/shazforiot/MSBuild_firstproject.git
 > git.exe --version # timeout=10
 > git --version # 'git version 2.45.2.windows.1'
 > git.exe fetch --tags --force --progress -- https://github.com/shazforiot/MSBuild_firstproject.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision f2bc042c04c6e72427c380bcaee6d6fee7b49adf (refs/remotes/origin/master)
 > git.exe config core.sparsecheckout # timeout=10
 > git.exe checkout -f f2bc042c04c6e72427c380bcaee6d6fee7b49adf # timeout=10
Commit message: "updated"
 > git.exe rev-list --no-walk f2bc042c04c6e72427c380bcaee6d6fee7b49adf # timeout=10
[SonarQube] $ C:\ProgramData\Jenkins\.jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\sonarqube\bin\sonar-scanner.bat -Dsonar.host.url=http://localhost:9000 -Dsonar.projectKey=sonarqube -
Dsonar.login=admin -Dsonar.host.url=http://127.0.0.1:9000 -Dsonar.sources=C:\ProgramData\Jenkins\.jenkins\workspace\SonarQube -Dsonar.password=admin123 -
Dsonar.projectBaseDir=C:\ProgramData\Jenkins\.jenkins\workspace\SonarQube
16:16:39.198 WARN  Property 'sonar.host.url' with value 'http://localhost:9000' is overridden with value 'http://127.0.0.1:9000'
16:16:39.206 INFO  Scanner configuration file: C:\ProgramData\Jenkins\.jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\sonarqube\bin\..\conf\sonar-scanner.properties
16:16:39.206 INFO  Project root configuration file: NONE
16:16:39.230 INFO  SonarScanner CLI 6.1.0.4477
16:16:39.230 INFO  Java 21.0.4 Eclipse Adoptium (64-bit)
16:16:39.230 INFO  Windows 11 10.0 amd64
16:16:39.230 INFO  SONAR_SCANNER_OPTS=-Dsonar.ws.timeout=300
16:16:39.254 INFO  User cache: C:\Windows\system32\config\systemprofile\.sonar\cache
16:16:58.734 INFO  Using git CLI to retrieve untracked files
16:16:58.791 INFO  Analyzing language associated files and files included via "sonar.text.inclusions" that are tracked by git
16:16:58.856 INFO  14 source files to be analyzed
16:16:59.154 INFO  14/14 source files have been analyzed
16:16:59.154 INFO  Sensor TextAndSecretsSensor [text] (done) | time=1306ms
16:16:59.163 INFO  ------------- Run sensors on project
16:16:59.373 INFO  Sensor C# [csharp]
16:16:59.373 WARN  Your project contains C# files which cannot be analyzed with the scanner you are using. To analyze C# or VB.NET, you must use the SonarScanner for .NET 5.x or higher, see
https://redirect.sonarsource.com/doc/install-configure-scanner-msbuild.html
16:16:59.373 INFO  Sensor C# [csharp] (done) | time=0ms
16:16:59.373 INFO  Sensor Analysis Warnings import [csharp]
16:16:59.379 INFO  Sensor Analysis Warnings import [csharp] (done) | time=0ms
16:16:59.379 INFO  Sensor C# File Caching Sensor [csharp]
16:16:59.379 WARN  Incremental PR analysis: Could not determine common base path, cache will not be computed. Consider setting 'sonar.projectBaseDir' property.
16:16:59.379 INFO  Sensor C# File Caching Sensor [csharp] (done) | time=6ms
16:16:59.379 INFO  Sensor Zero Coverage Sensor
16:16:59.389 INFO  Sensor Zero Coverage Sensor (done) | time=10ms
16:16:59.389 INFO  SCM Publisher SCM provider for this project is: git
16:16:59.389 INFO  SCM Publisher 4 source files to be analyzed
16:16:59.838 INFO  SCM Publisher 4/4 source files have been analyzed (done) | time=449ms
16:16:59.846 INFO  CPD Executor Calculating CPD for 0 files
16:16:59.846 INFO  CPD Executor CPD calculation finished (done) | time=0ms
16:16:59.854 INFO  SCM revision ID 'f2bc042c04c6e72427c380bcaee6d6fee7b49adf'
16:17:00.121 INFO  Analysis report generated in 120ms, dir size=201.1 kB
16:17:00.195 INFO  Analysis report compressed in 57ms, zip size=22.4 kB
16:17:00.393 INFO  Analysis report uploaded in 195ms
16:17:00.394 INFO  ANALYSIS SUCCESSFUL, you can find the results at: http://127.0.0.1:9000/dashboard?id=sonarqube
16:17:00.395 INFO  Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
16:17:00.395 INFO  More about the report processing at http://127.0.0.1:9000/api/ce/task?id=acd819f5-9e70-42ab-bff7-3cc893e2cae4
16:17:00.405 INFO  Analysis total time: 18.743 s
16:17:00.408 INFO  SonarScanner Engine completed successfully
16:17:00.494 INFO  EXECUTION SUCCESS
16:17:00.494 INFO  Total time: 21.288s
Finished: SUCCESS
```

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



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

Conclusion:

This experiment effectively showcased the integration of Jenkins with SonarQube for conducting static code analysis. Through the configuration of both tools, we were able to assess a sample project, gaining valuable insights into the SAST process and the role of Jenkins in automating the identification of code vulnerabilities.

Name: Nishant S Khetal **Class: D15C** **Roll No: 24**

Experiment No. 08

Aim: To establish a Jenkins CI/CD Pipeline integrated with SonarQube/GitLab for conducting static analysis on a sample web application (Java/Python) to identify bugs, code smells, and security vulnerabilities.

Theory:

Understanding SAST:

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

Challenges Addressed by SAST:

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

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

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

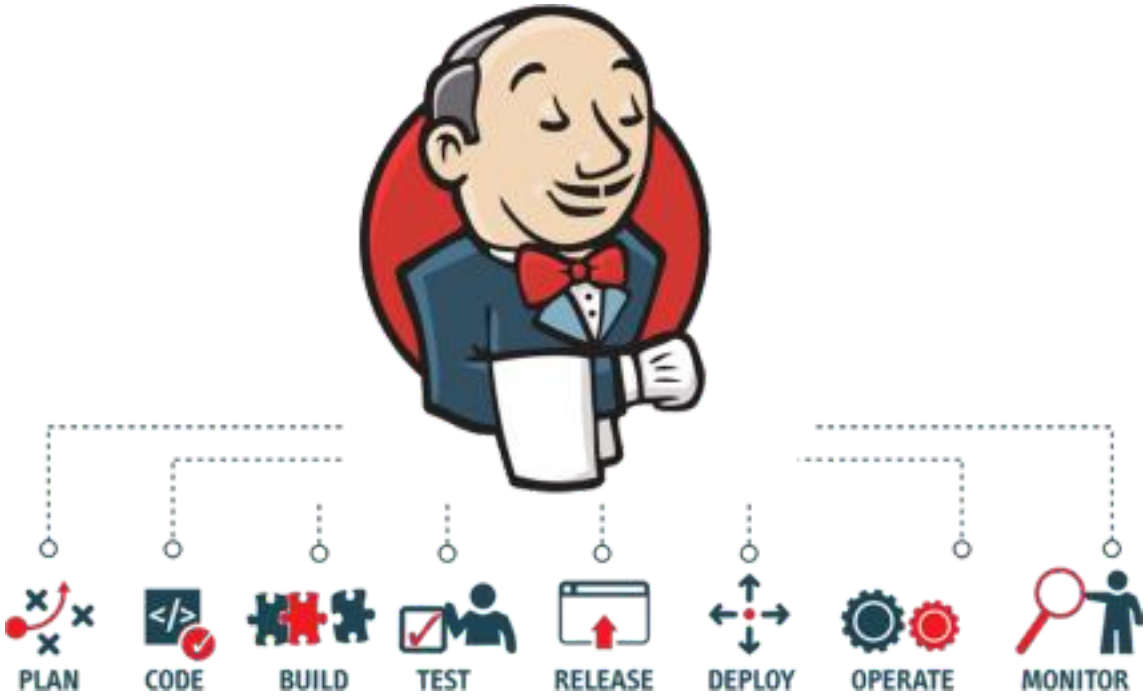
Importance of SAST:

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

What is a CI/CD Pipeline?

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

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



Now, consider the CI/CD pipeline as the backbone of the DevOps approach. This Pipeline is responsible for building codes, running tests, and deploying new software versions. The CI/CD pipeline serves as the foundation of the DevOps methodology, managing code builds, tests, and deployment of new software versions in a structured manner.

Introduction to SonarQube:

SonarQube is an open-source platform developed by SonarSource for continuous inspection of code quality. Sonar does static code analysis, which provides a detailed report of bugs, code smells, vulnerabilities, code duplications.

It supports 25+ major programming languages through built-in rulesets and can also be extended with various plugins.

Advantages of SonarQube

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

Integrating Jenkins with SonarQube:

Prerequisites:

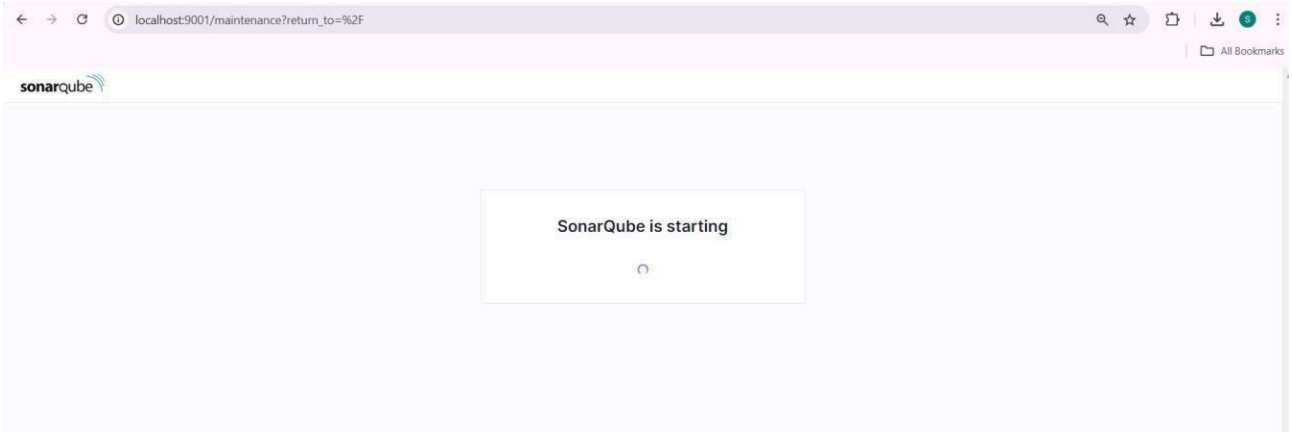
- Jenkins installed
- Docker Installed (for SonarQube)
- SonarQube Docker Image

Steps to Create a Jenkins CI/CD Pipeline and Utilize SonarQube for SAST:

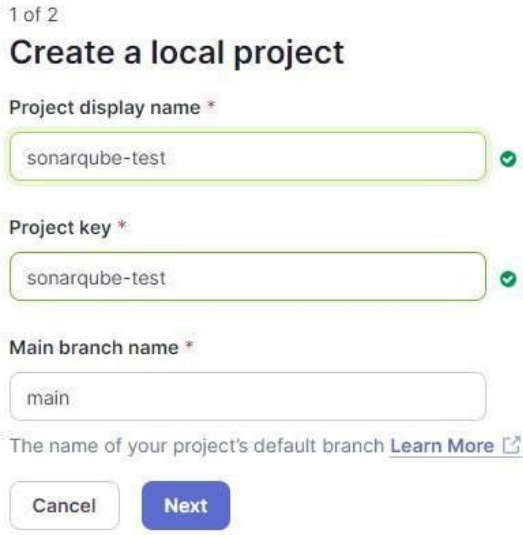
1. Open up Jenkins Dashboard on localhost, port 8080 or whichever port it is at for you.
2. Run SonarQube in a Docker container using this command –

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

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

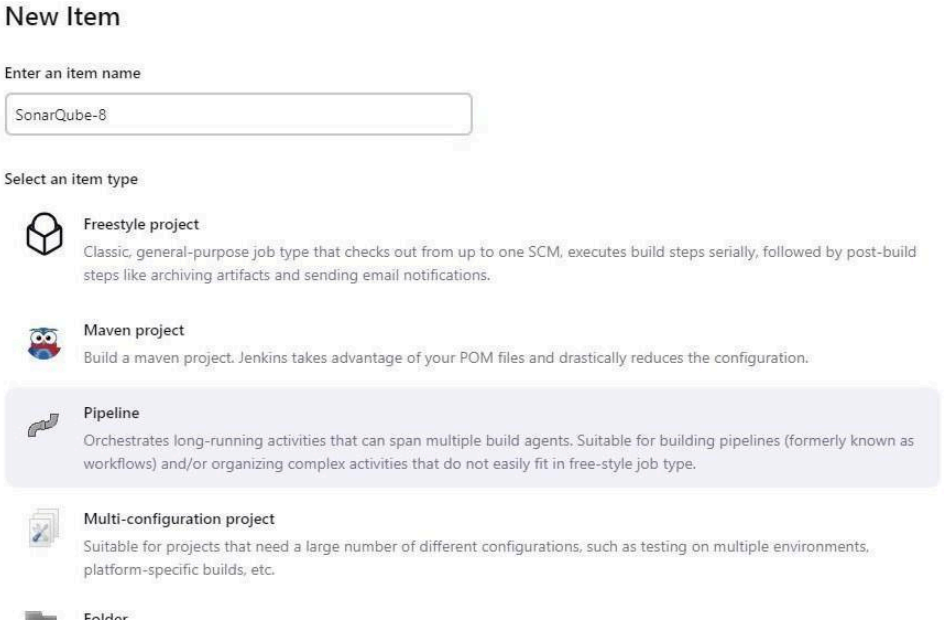


4. Login to SonarQube using username *admin* and password *admin*.
5. Create a manual project in SonarQube with the name **sonarqube-test**



Setup the project and come back to Jenkins Dashboard.

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



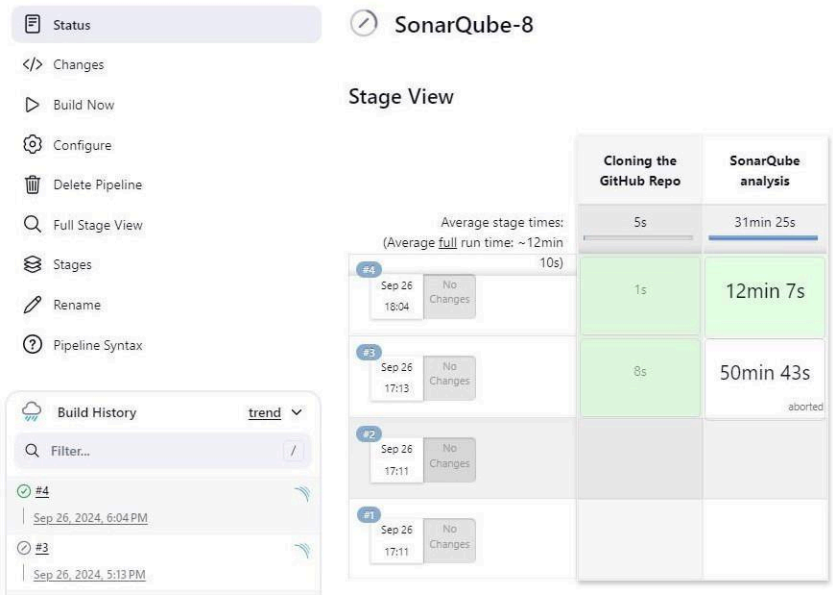
7. Input the following script under Pipeline Script:

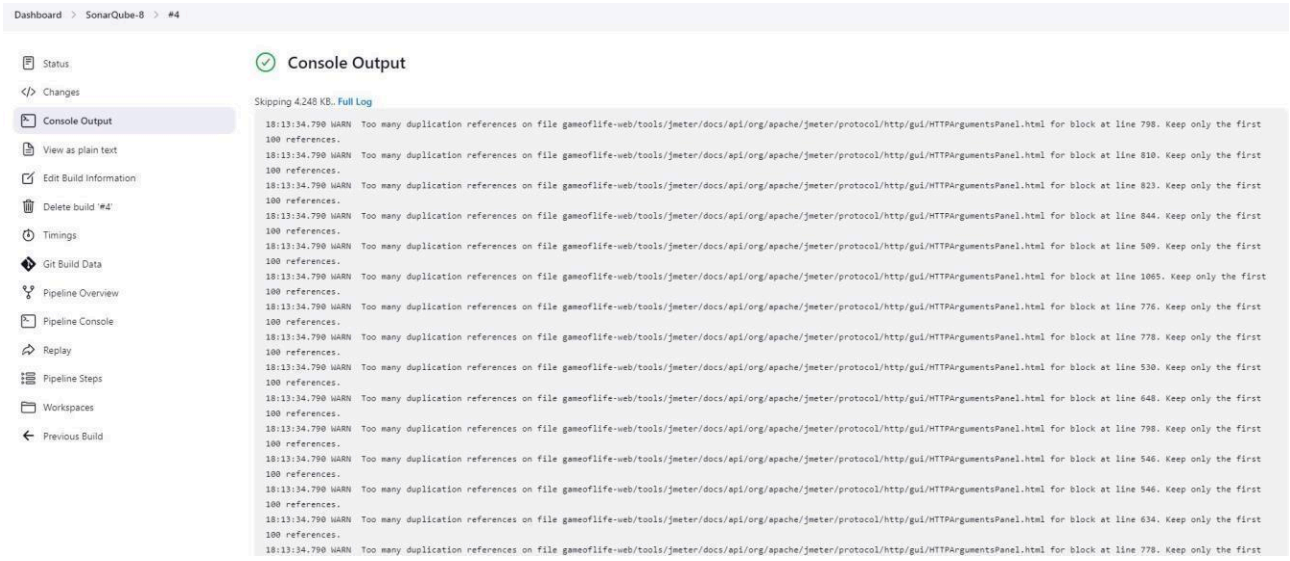
```
node {
  stage('Cloning the GitHub Repo') {
    git 'https://github.com/shazforiot/GOL.git'
  } stage('SonarQube
  analysis') {
    withSonarQubeEnv('sonarqube') {
      sh "<PATH_TO_SONARQUBE_FOLDER>//bin//sonar-scanner \
      -D sonar.login=<SonarQube_USERNAME> \
      -D sonar.password=<SonarQube_PASSWORD>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

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

8. Execute The Build.

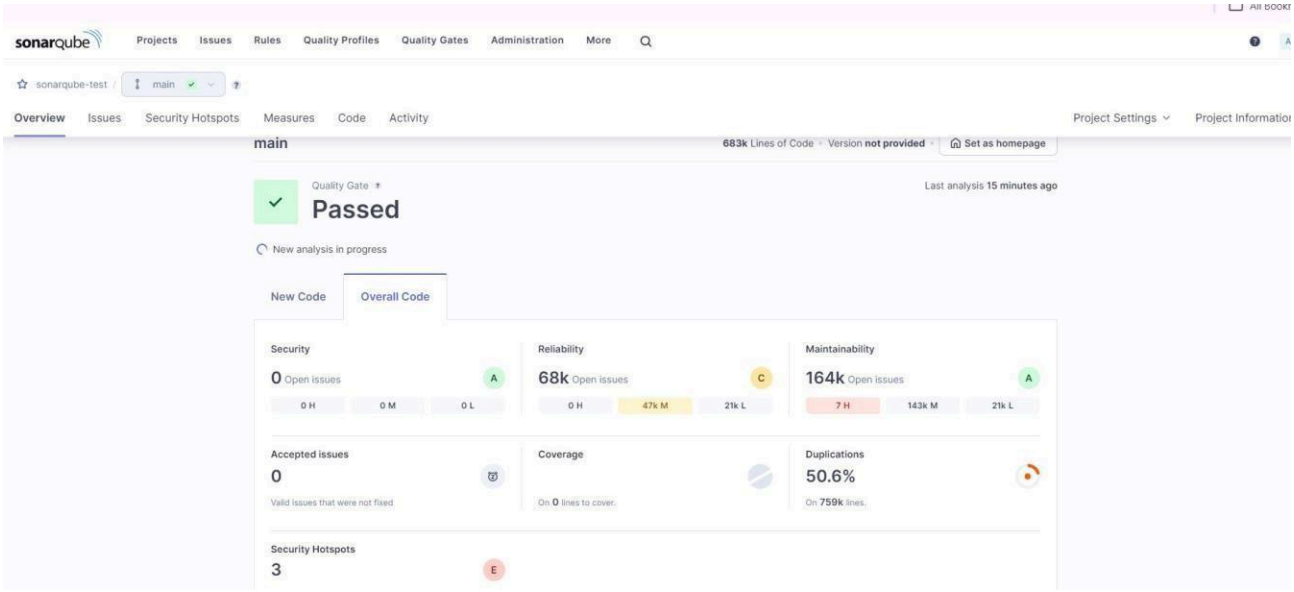
9. Review the console output upon build completion.





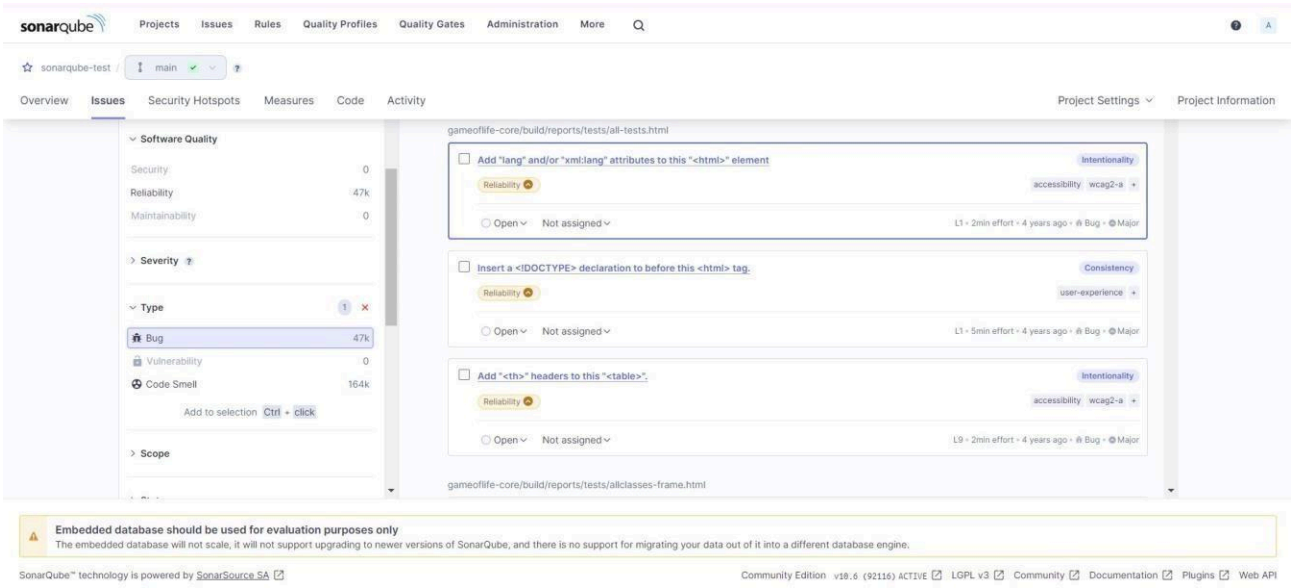

```
references.
18:13:39.657 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/gui/util/TextAreaCellRenderer.html for block at line 75. Keep only the first 100
references.
18:13:39.657 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/gui/util/TextAreaCellRender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

10. After that, check the project in SonarQube.

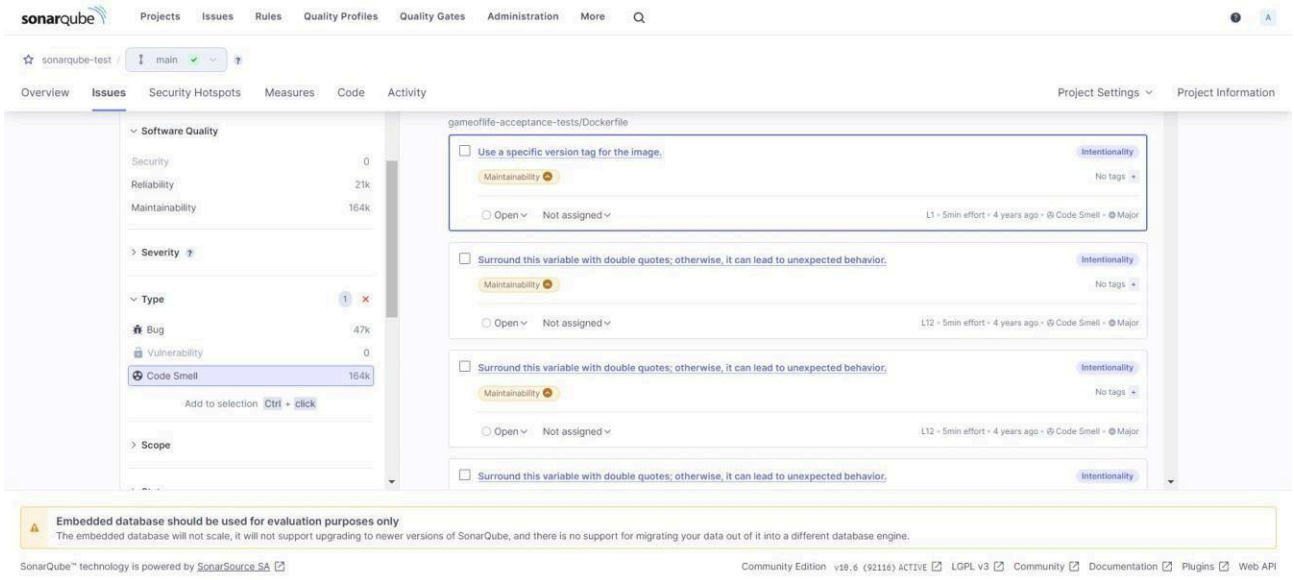


Under different tabs, check all different issues with the code.

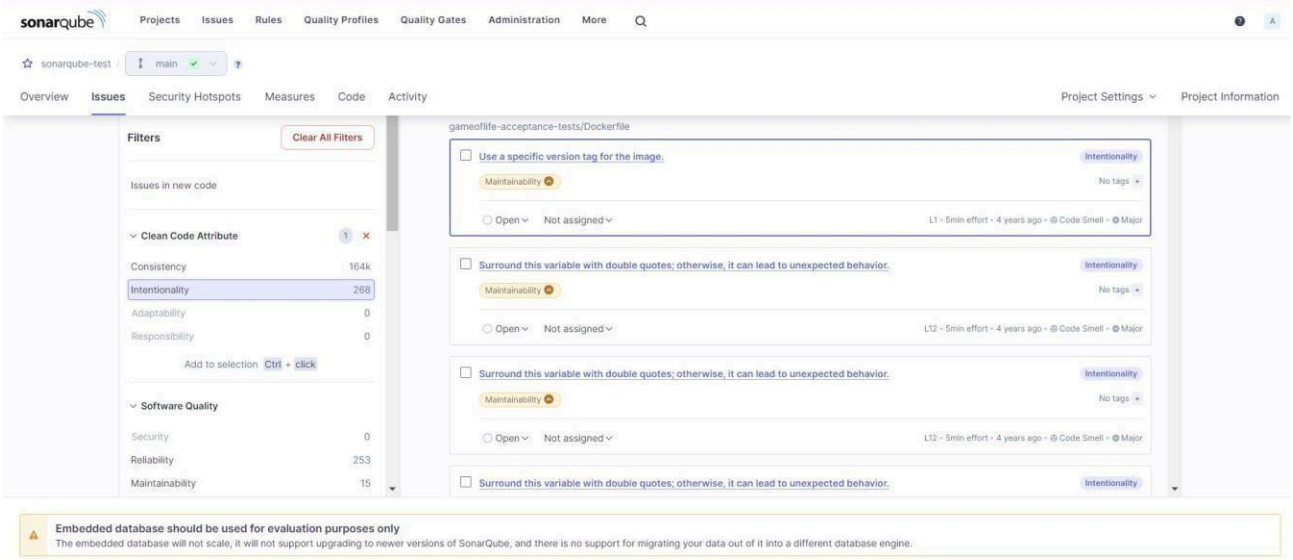
11. **Categories of Code Problems:**



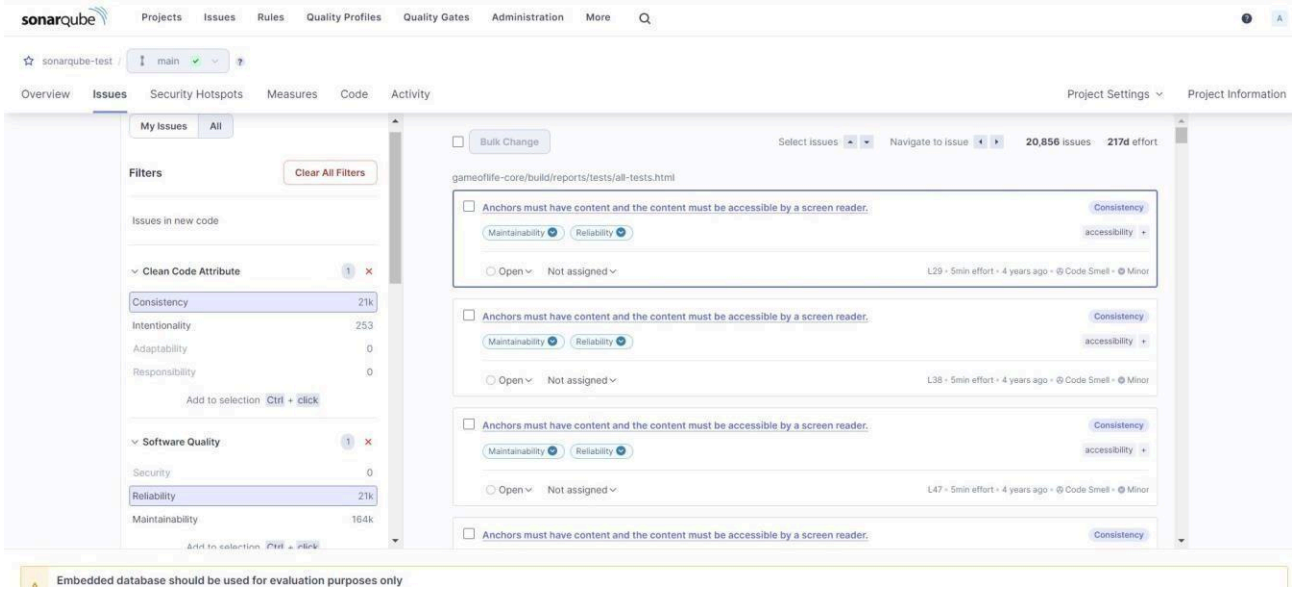
Code Smells



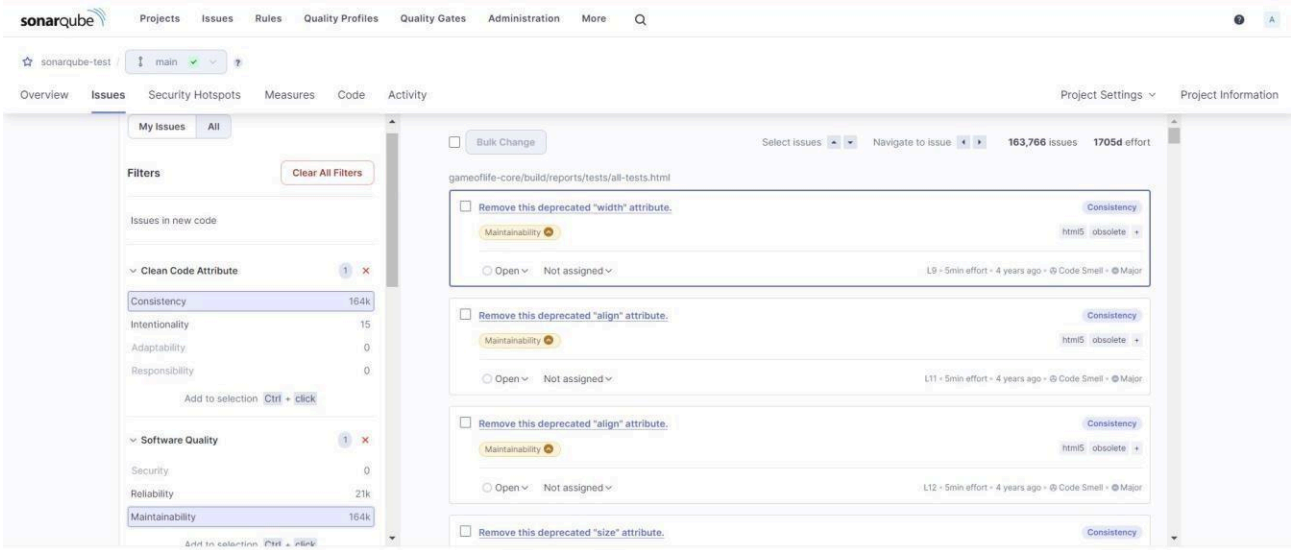
Intentional Issues



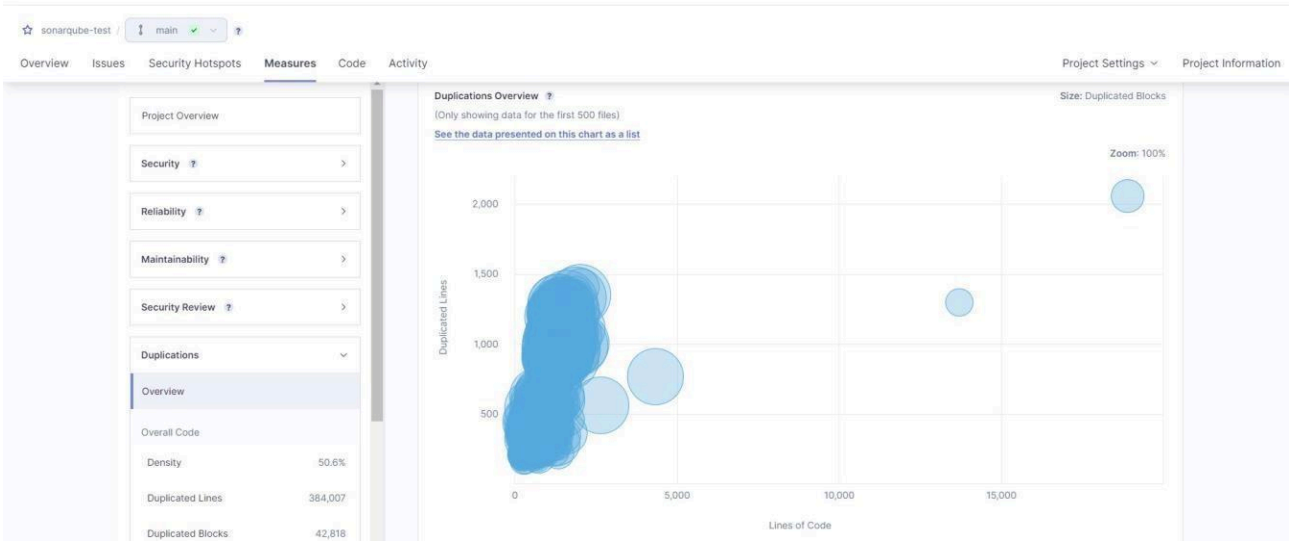
Reliability Issue



Maintainability Issue



Duplicates



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

Conclusion:

I successfully established a CI/CD pipeline using Jenkins integrated with SonarQube for static code analysis of a sample Java application. I configured SonarQube within a Docker container and set up Jenkins to clone the GitHub repository for analysis. The pipeline effectively identified various issues, including bugs and security vulnerabilities, which I reviewed in SonarQube. This experience improved my skills in CI/CD tool configuration and underscored the significance of automated code quality maintenance. Overall, I gained valuable insights into the integration of tools for effective software development practices.

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

Theory:

What is Nagios?

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

Nagios is used for continuous monitoring of systems, applications, service and business processes in a DevOps culture.

Why We Need Nagios tool?

Here are the important reasons to use Nagios monitoring tool:

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

Features of Nagios

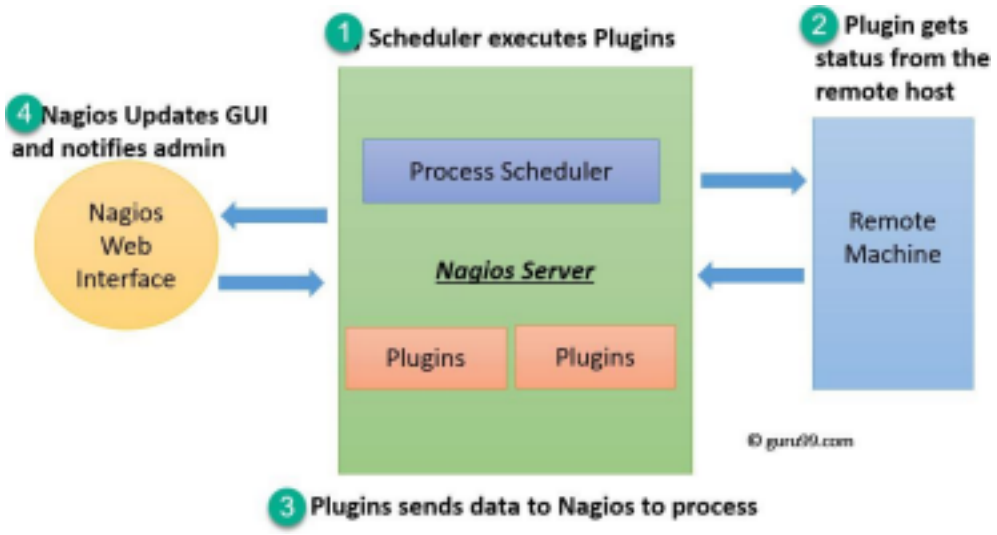
Following are the important features of Nagios monitoring tool:

- Relatively scalable, Manageable, and Secure
- Good log and database system
- Informative and attractive web interfaces
- Automatically send alerts if condition changes
- If the services are running fine, then there is no need to do check that host is an alive
- Helps you to detect network errors or server crashes
- You can troubleshoot the performance issues of the server.
- The issues, if any, can be fixed automatically as they are identified during the monitoring process
- You can monitor the entire business process and IT infrastructure with a single pass ● The product's architecture is easy to write new plugins in the language of your choice ● Nagios allows you to read its configuration from an entire directory which helps you to decide how to define individual files

- Utilizes topology to determine dependencies
- Monitor network services like HTTP, SMTP, HTTP, SNMP, FTP, SSH, POP, etc.
- Helps you to define network host hierarchy using parent hosts
- Ability to define event handlers that runs during service or host events for proactive problem resolution
- Support for implementing redundant monitoring hosts

Nagios Architecture

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



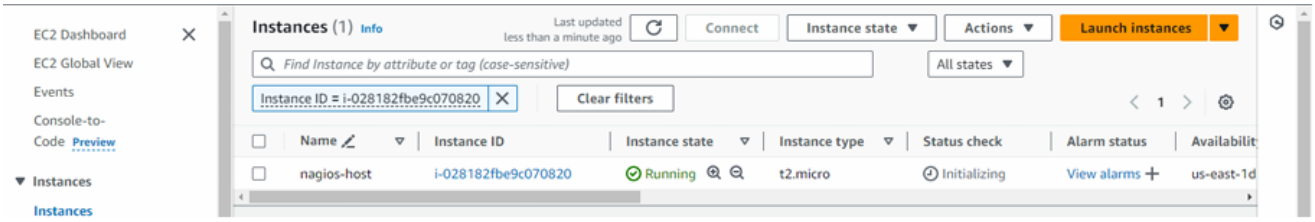
1. The scheduler is a component of the server part of Nagios. It sends a signal to execute the plugins at the remote host.
2. The plugin gets the status from the remote host
3. The plugin sends the data to the process scheduler
4. The process scheduler updates the GUI and notifications are sent to admins.

Installation of Nagios

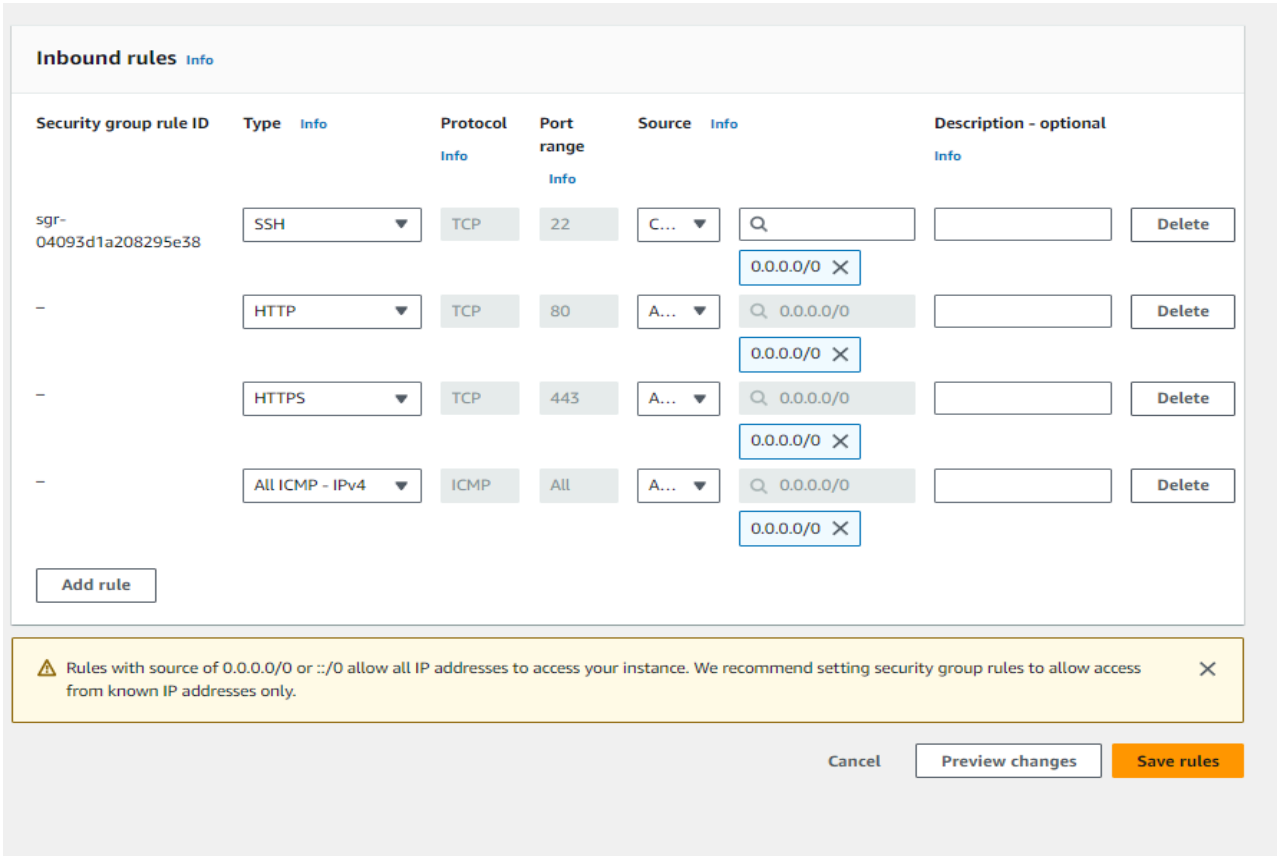
Prerequisites: AWS Free Tier

Steps:

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



2. Under Security Group, make sure HTTP, HTTPS, SSH, ICMP are open from everywhere. You have to edit the inbound rules of the specified Security Group for this.



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

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
sudo yum install gd gd-devel
```

```
yum install: error: unrecognized arguments: -Oy
[ec2-user@ip-172-31-38-150 ~]$ sudo yum install gd gd-devel -y
Last metadata expiration check: 0:05:12 ago on Sun Oct  6 11:15:04 2024.
Dependencies resolved.
```

Package	Architecture	Version	Repository	Si
Installing:				
gd	x86_64	2.3.3-5.amzn2023.0.3	amazonlinux	139
gd-devel	x86_64	2.3.3-5.amzn2023.0.3	amazonlinux	38
Installing dependencies:				
brotli	x86_64	1.0.9-4.amzn2023.0.2	amazonlinux	314
brotli-devel	x86_64	1.0.9-4.amzn2023.0.2	amazonlinux	31
bzip2-devel	x86_64	1.0.8-6.amzn2023.0.2	amazonlinux	214
cairo	x86_64	1.17.6-2.amzn2023.0.1	amazonlinux	684
cmake-filesystem	x86_64	3.22.2-1.amzn2023.0.4	amazonlinux	16
fontconfig	x86_64	2.13.94-2.amzn2023.0.2	amazonlinux	273
fontconfig-devel	x86_64	2.13.94-2.amzn2023.0.2	amazonlinux	128
fonts-filesystem	noarch	1:2.0.5-12.amzn2023.0.2	amazonlinux	9.5
freetype	x86_64	2.13.2-5.amzn2023.0.1	amazonlinux	423
freetype-devel	x86_64	2.13.2-5.amzn2023.0.1	amazonlinux	912
glib2-devel	x86_64	2.74.7-689.amzn2023.0.2	amazonlinux	486
google-noto-fonts-common	noarch	20201206-2.amzn2023.0.2	amazonlinux	15
google-noto-sans-vf-fonts	noarch	20201206-2.amzn2023.0.2	amazonlinux	492
graphite2	x86_64	1.3.14-7.amzn2023.0.2	amazonlinux	97
graphite2-devel	x86_64	1.3.14-7.amzn2023.0.2	amazonlinux	21
harfbuzz	x86_64	7.0.0-2.amzn2023.0.1	amazonlinux	868
harfbuzz-devel	x86_64	7.0.0-2.amzn2023.0.1	amazonlinux	404
harfbuzz-icu	x86_64	7.0.0-2.amzn2023.0.1	amazonlinux	18
jbigkit-libs	x86_64	2.1-21.amzn2023.0.2	amazonlinux	54
langpacks-core-font-en	noarch	3.0-21.amzn2023.0.4	amazonlinux	10
libICE	x86_64	1.0.10-6.amzn2023.0.2	amazonlinux	71
libSM	x86_64	1.2.3-8.amzn2023.0.2	amazonlinux	42

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
```

```
Complete!
[ec2-user@ip-172-31-38-150 ~]$ sudo adduser -m nagios
[ec2-user@ip-172-31-38-150 ~]$ sudo passwd nagios
Changing password for user nagios.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[ec2-user@ip-172-31-38-150 ~]$
```

6. Create a new user group

```
sudo groupadd nagcmd
```

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
```

8. Create a new directory for Nagios downloads

```
mkdir ~/downloads
cd ~/downloads
```

```
passwd: all authentication tokens updated successfully.
[ec2-user@ip-172-31-38-150 ~]$ sudo groupadd nagcmd
[ec2-user@ip-172-31-38-150 ~]$ sudo usermod -a -G nagcmd nagios
[ec2-user@ip-172-31-38-150 ~]$ sudo usermod -a -G nagcmd apache
[ec2-user@ip-172-31-38-150 ~]$ mkdir ~/downloads
[ec2-user@ip-172-31-38-150 ~]$ cd ~/downloads
[ec2-user@ip-172-31-38-150 downloads]$ wget https://go.nagios.org/l
24-09-17/6kqcx
```

9. Use wget to download the source zip files.

```
wget https://go.nagios.org/l/975333/2024-09-17/6kqcx
wget https://nagios-plugins.org/download/nagios-plugins-2.4.11.tar.gz
```

```
[ec2-user@ip-172-31-38-150 downloads]$ wget https:
24-09-17/6kqcx
--2024-10-06 11:23:50--  https://go.nagios.org/l/9
Resolving go.nagios.org (go.nagios.org)... 3.92.12
72.219, ...
Connecting to go.nagios.org (go.nagios.org)|3.92.1
HTTP request sent, awaiting response... 302 Found
Location: http://assets.nagios.com/downloads/nagio
.tar.gz?utm_source=Nagios.org&utm_content=Download
.5+Download+&pi_content=1e9662c93afb2ed6bd2e3f3cc3
```

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

```
tar zxvf nagios-4.5.5.tar.gz
cd nagios-4.5.5
```

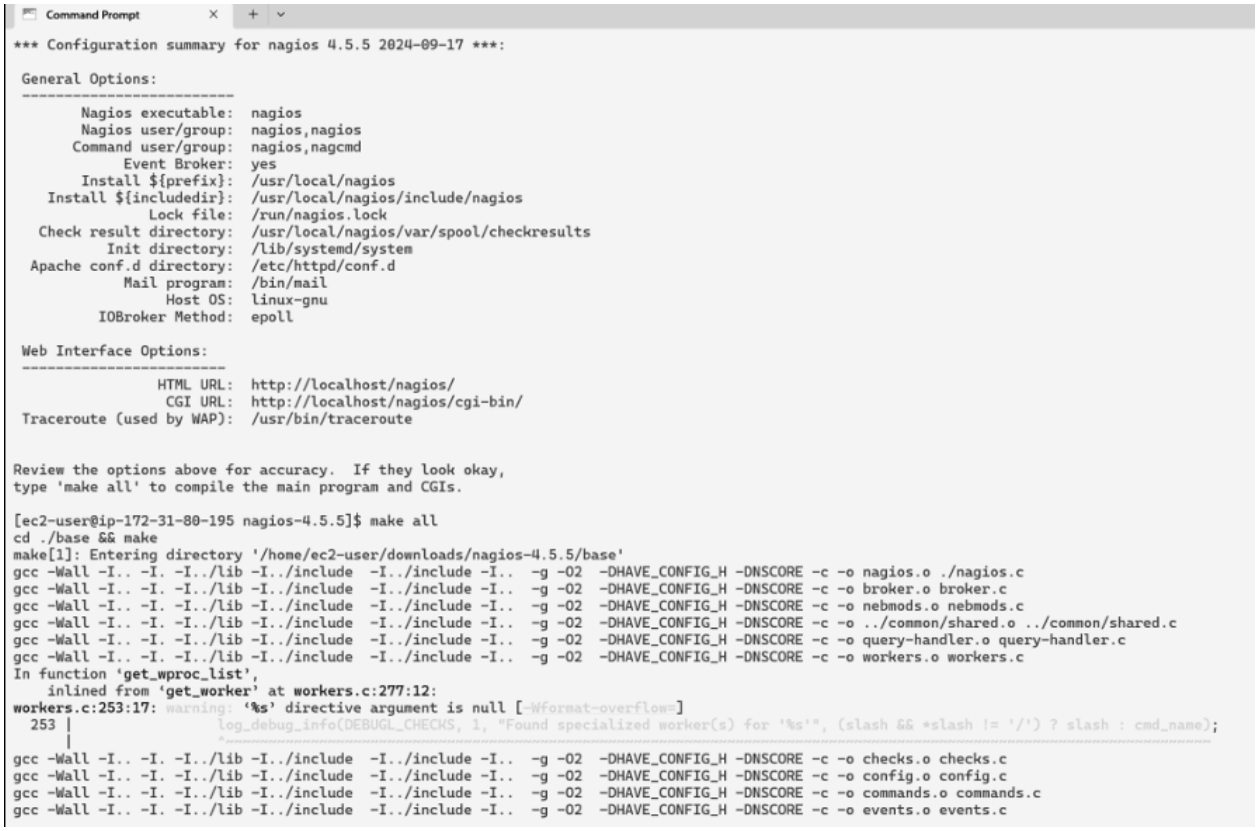
```
24-09-17/6kqcx
--2024-10-06 11:23:50--  https://go.nagios.org/l/975333/2024-09-17/6kqcx
Resolving go.nagios.org (go.nagios.org)... 3.92.120.28, 52.54.96.194, 3.215
72.219, ...
Connecting to go.nagios.org (go.nagios.org)|3.92.120.28|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: http://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5
.tar.gz?utm_source=Nagios.org&utm_content=Download+Form&utm_campaign=Core+4
.5+Download+&pi_content=1e9662c93afb2ed6bd2e3f3cc38771a7f01125e969f2a75b0e2
4439d4a81d8 [following]
--2024-10-06 11:23:50--  http://assets.nagios.com/downloads/nagioscore/rele
es/nagios-4.5.5.tar.gz?utm_source=Nagios.org&utm_content=Download+Form&utm_
mpaign=Core+4.5.5+Download+&pi_content=1e9662c93afb2ed6bd2e3f3cc38771a7f011
e969f2a75b0e2254439d4a81d8
Resolving assets.nagios.com (assets.nagios.com)... 45.79.49.120, 2600:3c00:
03c:92ff:fef7:45ce
Connecting to assets.nagios.com (assets.nagios.com)|45.79.49.120|:80... con
```

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

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

12. Compile the source code.

```
make all
```



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-172-31-38-150 nagios-4.4.6]$ sudo make install-webconf
/usr/bin/install -c -m 644 sample-config/httpd.conf /etc/httpd/conf.d/nagio
s.conf
if [ 0 -eq 1 ]; then \
        ln -s /etc/httpd/conf.d/nagios.conf /etc/apache2/sites-enabled/nagi
os.conf; \
fi

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

[ec2-user@ip-172-31-38-150 nagios-4.4.6]$
```

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

```
[ec2-user@ip-172-31-38-150 nagios-plugins-2.4.11]$ sudo yum install openssl
evel
Last metadata expiration check: 0:13:31 ago on Sun Oct  6 11:15:04 2024.
Dependencies resolved.
================================================================================
 Package                 Architecture        Version
           Repository                 Size
================================================================================
Installing:
 openssl-devel           x86_64              1:3.0.8-1.amzn2023.0.14
           amazonlinux               3.0 M

Transaction Summary
```

14. Configure the web interface.

```
sudo make install-webconf
```

```
[ec2-user@ip-172-31-38-150 nagios-4.4.6]$ sudo make install-webconf
/usr/bin/install -c -m 644 sample-config/httpd.conf /etc/httpd/conf.d/nagio
s.conf
if [ 0 -eq 1 ]; then \
        ln -s /etc/httpd/conf.d/nagios.conf /etc/apache2/sites-enabled/nagi
os.conf; \
fi

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

[ec2-user@ip-172-31-38-150 nagios-4.4.6]$
```

15. 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-172-31-38-150 nagios-4.4.6]$ sudo htpasswd -c /usr/local/nagio
s/etc/htpasswd.users nagiosadmin          sudo htpasswd -c /usr/local/nagio
s/etc/htpasswd.users nagiosadmin
New password:
Re-type new password:
Adding password for user nagiosadmin
[ec2-user@ip-172-31-38-150 nagios-4.4.6]$
```

16. Restart Apache

```
sudo service httpd restart
```

```
[ec2-user@ip-172-31-38-150 nagios-4.4.6]$ sudo service httpd restart
Redirecting to /bin/systemctl restart httpd.service
[ec2-user@ip-172-31-38-150 nagios-4.4.6]$
```

17. Go back to the downloads folder and unzip the plugins zip file.

```
cd ~/downloads
tar zxvf nagios-plugins-2.4.11.tar.gz
```

```
[ec2-user@ip-172-31-38-150 downloads]$ tar zxvf nagios-plugins-2.4.11.tar.gz
nagios-plugins-2.4.11/
nagios-plugins-2.4.11/build-aux/
nagios-plugins-2.4.11/build-aux/compile
nagios-plugins-2.4.11/build-aux/config.guess
nagios-plugins-2.4.11/build-aux/config.rpath
nagios-plugins-2.4.11/build-aux/config.sub
nagios-plugins-2.4.11/build-aux/install-sh
nagios-plugins-2.4.11/build-aux/ltmain.sh
nagios-plugins-2.4.11/build-aux/missing
nagios-plugins-2.4.11/build-aux/mkinstalldirs
```

18. Compile and install plugins

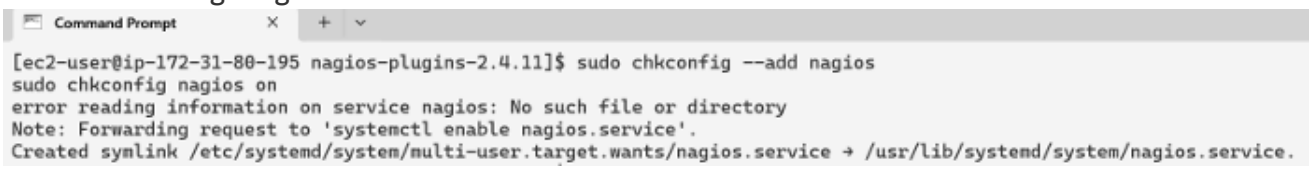
```
cd nagios-plugins-2.4.11
./configure --with-nagios-user=nagios --with-nagios-group=nagios
make
sudo make install
```

```
[ec2-user@ip-172-31-38-150 downloads]$ cd nagios-plugins-2.4.11
[ec2-user@ip-172-31-38-150 nagios-plugins-2.4.11]$ ./configure --with-nagios-user=nagios
 --with-nagios-group=nagios                        /configure --with-nagios-user=nagios
 --with-nagios-group=nagios
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /usr/bin/mkdir -p
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking whether make supports nested variables... yes
checking whether to enable maintainer-specific portions of Makefiles... yes
```

19. Start Nagios

Add Nagios to the list of system services

```
sudo chkconfig --add nagios
sudo chkconfig nagios on
```



Verify the sample configuration files

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

```
[ec2-user@ip-172-31-38-150 nagios-plugins-2.4.11]$ sudo /usr/local/nagios/bin/nagios -v
/usr/local/nagios/etc/nagios.cfg                    sudo /usr/local/nagios/bin/nagios -v
/usr/local/nagios/etc/nagios.cfg

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.
        Checked 0 service groups.
        Checked 1 contacts.
        Checked 1 contact groups.
        Checked 24 commands.
        Checked 5 time periods.
```

If there are no errors, you can go ahead and start Nagios.

```
sudo service nagios start
```

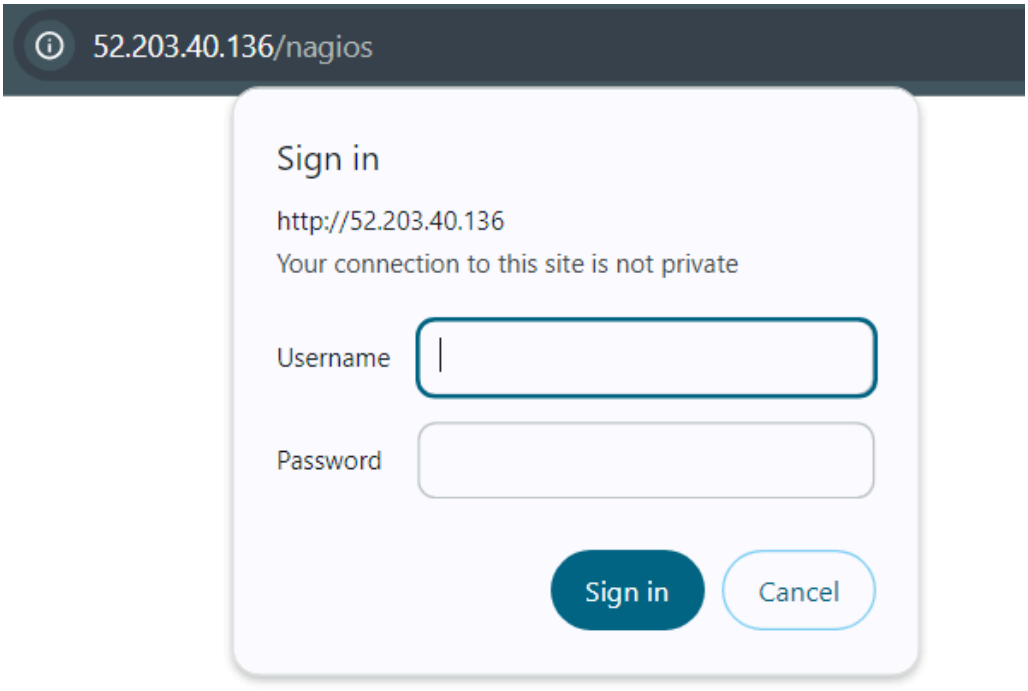
```
Things look okay - No serious problems were detected during the pre-flight check
[ec2-user@ip-172-31-38-150 nagios-plugins-2.4.11]$ sudo service nagios start
Redirecting to /bin/systemctl start nagios.service
[ec2-user@ip-172-31-38-150 nagios-plugins-2.4.11]$
```

20. Check the status of Nagios

```
sudo systemctl status nagios
Redirecting to /bin/systemctl start nagios.service
[ec2-user@ip-172-31-38-150 nagios-plugins-2.4.11]$ 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 Sun 2024-10-06 11:51:46 UTC; 1min 33s ago
       Docs: https://www.nagios.org/documentation
   Main PID: 89956 (nagios)
      Tasks: 6 (limit: 1112)
     Memory: 2.4M
        CPU: 36ms
     CGroup: /system.slice/nagios.service
             ├─89956 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─89957 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/na
os.qh
             ├─89958 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/na
os.qh
             ├─89959 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/na
os.qh
             ├─89960 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/na
os.qh
             └─89961 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg

Oct 06 11:51:46 ip-172-31-38-150.ec2.internal nagios[89956]: qh: Socket '/usr/local/na
os/var/rw/nagios.qh' successfully
Oct 06 11:51:46 ip-172-31-38-150.ec2.internal nagios[89956]: qh: core query handler re
```

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

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



Enter username as nagiosadmin and password which you set in Step 16.

23. After entering the correct credentials, you will see this page.

This means that Nagios was correctly installed and configured with its plugins so far.

Conclusion: We have successfully installed and configured Nagios Core, Nagios Plugins, and NRPE on a Linux machine. This enables us to effectively manage system performance and proactively address potential issues.

NAME: NISHANT S KHETAL **DIV: D15C** **ROLL NO: 24**

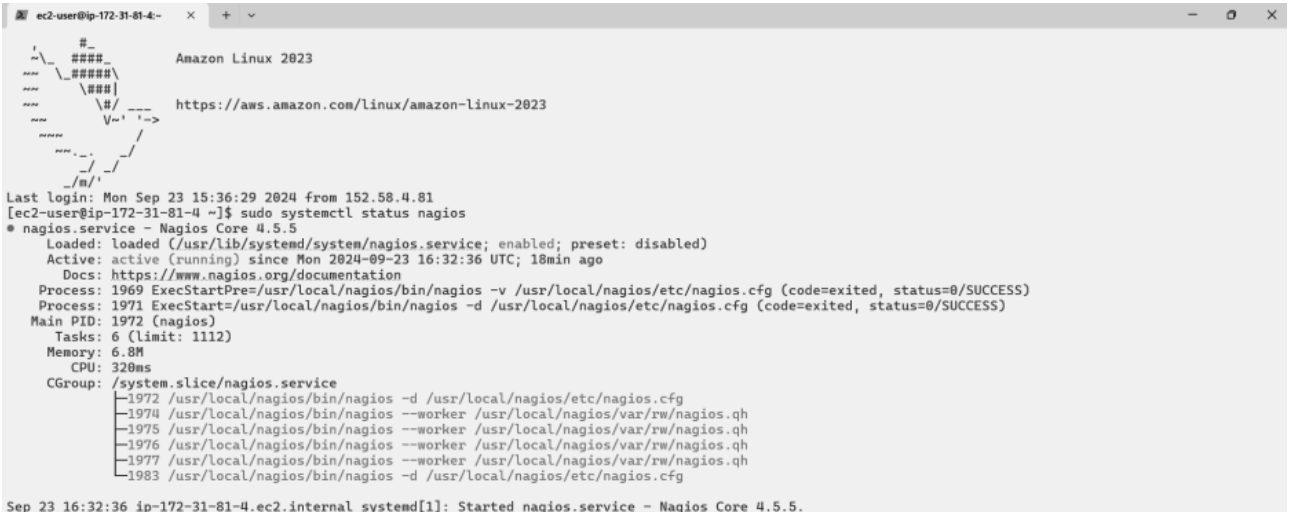
Advanced DevOps
Lab Experiment 10

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

Nagios. **Steps:**

Prerequisites: AWS Free Tier, Nagios Server running on Amazon Linux Machine.

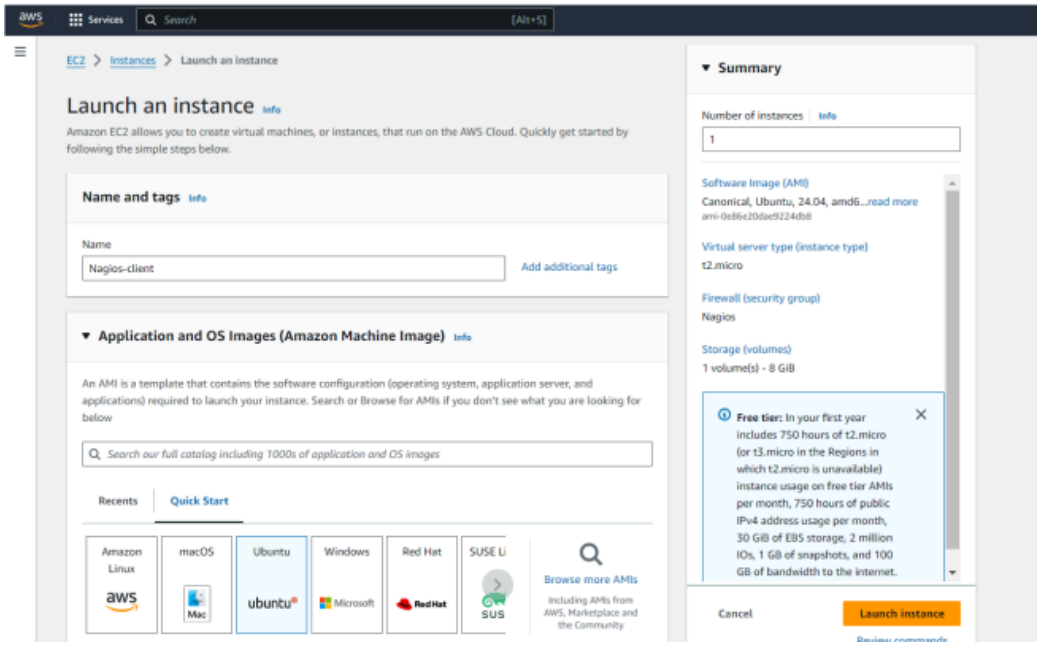
1. To Confirm that Nagios is running **on the server side**, run this
   sudo systemctl status nagios on the "NAGIOS HOST".



   You can proceed if you get this message.

2. Before we begin,
   To monitor a Linux machine, create an Ubuntu 20.04 server EC2 Instance in AWS.

   Provide it with the same security group as the Nagios Host and name it
   'linux-client' alongside the host.



For now, leave this machine as is, and go back to your nagios HOST machine.

3. On the server, run this command

```
ps -ef | grep nagios
```

```
Last login: Sat Oct  5 16:58:17 2024 from 42.111.112.18
[ec2-user@ip-172-31-43-65 ~]$ ps -ef | grep nagios
nagios     97412       1  0 17:34 ?        00:00:00 /usr/local/nagios/bin/nagios -d /usr/lo
nagios     97413   97412  0 17:34 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /
s.qh
nagios     97414   97412  0 17:34 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /
s.qh
nagios     97415   97412  0 17:34 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /
s.qh
nagios     97416   97412  0 17:34 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /
s.qh
nagios     97417   97412  0 17:34 ?        00:00:00 /usr/local/nagios/bin/nagios -d /usr/lo
ec2-user   98423   98399  0 17:51 pts/2    00:00:00 grep --color=auto nagios
```

4. Become a root user and create 2 folders

```
sudo su
mkdir /usr/local/nagios/etc/objects/monitorhosts
mkdir /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
```

```
[ec2-user@ip-172-31-43-65 ~]$ sudo su
mkdir /usr/local/nagios/etc/objects/monitorhosts
mkdir /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
[root@ip-172-31-43-65 ec2-user]#
```

5. Copy the sample localhost.cfg file to linuxhost folder

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

```
[root@ip-172-31-81-4 ec2-user]# cp /usr/local/nagios/etc/objects/localhost.cfg /usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg
[root@ip-172-31-81-4 ec2-user]#
```

6. Open linuxserver.cfg using nano and make the following changes

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

Change the hostname to linuxserver (EVERYWHERE ON THE FILE) Change address to the public IP address of your **LINUX CLIENT**.

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



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

# Define a host for the local machine

define host {

    use                     linux-server            ; Name of host template to use
                                                    ; This host definition will inherit all variables that are defined
                                                    ; in (or inherited by) the linux-server host template definition.
    host_name               localhost
    alias                   localhost
    address                 127.0.0.1
}



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

Change hostgroup_name under hostgroup to linux-servers1

Everywhere else on the file, change the hostname to linuxserver instead of localhost.

7. Open the Nagios Config file and add the following line

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

```
##Add this line
cfg_dir=/usr/local/nagios/etc/objects/monitorhosts/
```

8. Verify the configuration files

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

```
Running pre-flight check on configuration data...

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

Total Warnings: 0
Total Errors:   0

Things look okay - No serious problems were detected during the pre-flight check
[root@ip-172-31-43-65 ec2-user]#
```

You are good to go if there are no errors.

9. Restart the nagios service

```
service nagios restart
```

```
[root@ip-172-31-81-4 ec2-user]# service nagios restart
Redirecting to /bin/systemctl restart nagios.service
[root@ip-172-31-81-4 ec2-user]# sudo systemctl status nagi
```

Now it is time to switch to the client machine.

10. SSH into the machine or simply use the EC2 Instance Connect feature.

11. Make a package index update and install gcc, nagios-nrpe-server and the plugins.

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

```
ubuntu@ip-172-31-33-76:~$ sudo apt update -y
sudo apt install gcc -y
sudo apt install -y nagios-nrpe-server nagios-plugins
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:6 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [382 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Components [35.0 kB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 c-n-f Metadata [8328 B]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [537 kB]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [132 kB]
Get:16 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 c-n-f Metadata [8860 B]
Get:17 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [384 kB]
Get:18 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [159 kB]
Get:19 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [45.0 kB]
Get:20 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 c-n-f Metadata [14.9 kB]
Get:21 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Packages [14.4 kB]
Get:22 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse Translation-en [3608 B]
Get:23 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [212 B]
```

12. Open nrpe.cfg file to make changes.

```
sudo nano /etc/nagios/nrpe.cfg
```

Under allowed_hosts, add your nagios host IP address like so

NAME: NISHANT S KHETAL **DIV: D15C** **ROLL NO: 24**

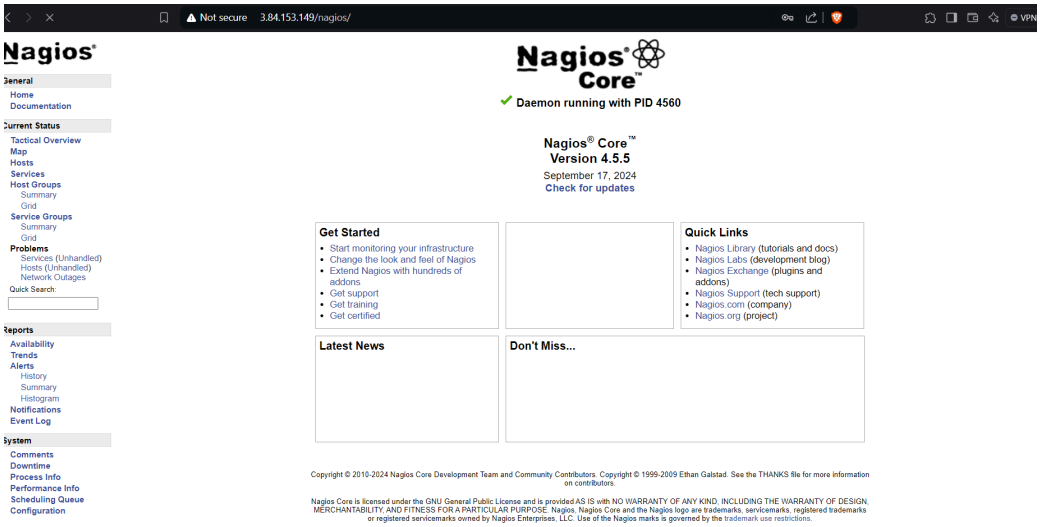
13. Restart the NRPE server

```
sudo systemctl restart nagios-nrpe-server
```

```
ubuntu@ip-172-31-83-152:~$ sudo nano /etc/nagios/nrpe.cfg

ubuntu@ip-172-31-83-152:~$ sudo systemctl restart nagios-nrpe-server
ubuntu@ip-172-31-83-152:~$
```

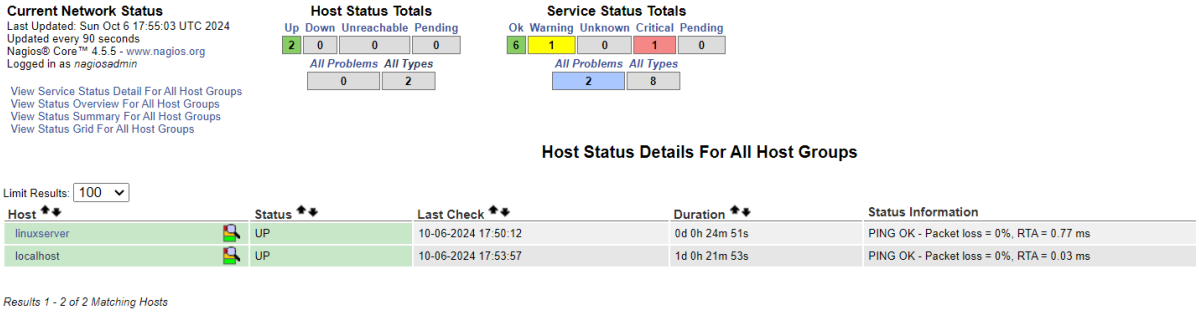
14. Now, check your nagios dashboard and you'll see a new host being added.

    Click on Hosts.



NAME: NISHANT S KHETAL **DIV: D15C** **ROLL NO: 24**

Click on linuxserver to see the host details



Current Network Status
Last Updated: Sun Oct 6 17:55:03 UTC 2024
Updated every 90 seconds
Nagios® Core™ 4.5.5 - www.nagios.org
Logged in as *nagiosadmin*

View Service Status Detail For All Host Groups
View Status Overview For All Host Groups
View Status Summary For All Host Groups
View Status Grid For All Host Groups

Host Status Totals

Up	Down	Unreachable	Pending
2	0	0	0

All Problems *All Types*
0 2

Service Status Totals

Ok	Warning	Unknown	Critical	Pending
6	1	0	1	0

All Problems *All Types*
2 8

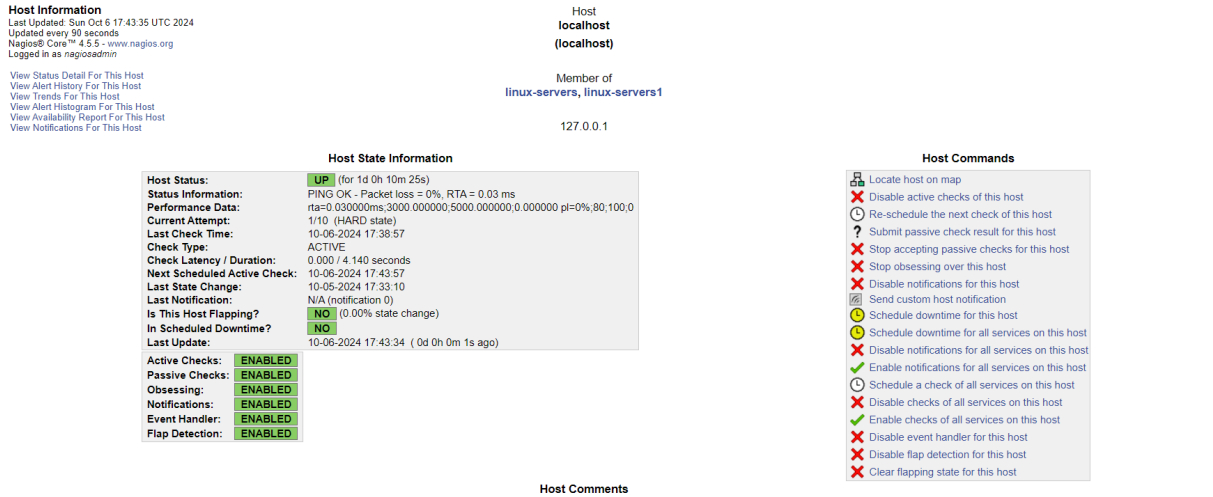
Host Status Details For All Host Groups

Limit Results: 100

Host	Status	Last Check	Duration	Status Information
linuxserver	UP	10-06-2024 17:50:12	0d 0h 24m 51s	PING OK - Packet loss = 0%, RTA = 0.77 ms
localhost	UP	10-06-2024 17:53:57	1d 0h 21m 53s	PING OK - Packet loss = 0%, RTA = 0.03 ms

Results 1 - 2 of 2 Matching Hosts

You can click Services to see all services and ports being monitored.



Host Information
Last Updated: Sun Oct 6 17:43:35 UTC 2024
Updated every 90 seconds
Nagios® Core™ 4.5.5 - www.nagios.org
Logged in as *nagiosadmin*

View Status Detail For This Host
View Alert History For This Host
View Trends For This Host
View Alert Histogram For This Host
View Availability Report For This Host
View Notifications For This Host

Host
localhost
(localhost)

Member of
linux-servers, linux-servers1

127.0.0.1

Host State Information

Host Status:	UP (for 1d 0h 10m 25s)
Status Information:	PING OK - Packet loss = 0%, RTA = 0.03 ms
Performance Data:	rta=0.030000ms;3000.000000;5000.000000;0.000000 pl=0%;80;100;0
Current Attempt:	1/10 (HARD state)
Last Check Time:	10-06-2024 17:38:57
Check Type:	ACTIVE
Check Latency / Duration:	0.000 / 4.140 seconds
Next Scheduled Active Check:	10-06-2024 17:43:57
Last State Change:	10-05-2024 17:33:10
Last Notification:	N/A (notification 0)
Is This Host Flapping?	NO (0.00% state change)
In Scheduled Downtime?	NO
Last Update:	10-06-2024 17:43:34 (0d 0h 0m 1s ago)

Active Checks:	ENABLED
Passive Checks:	ENABLED
Obsessing:	ENABLED
Notifications:	ENABLED
Event Handler:	ENABLED
Flap Detection:	ENABLED

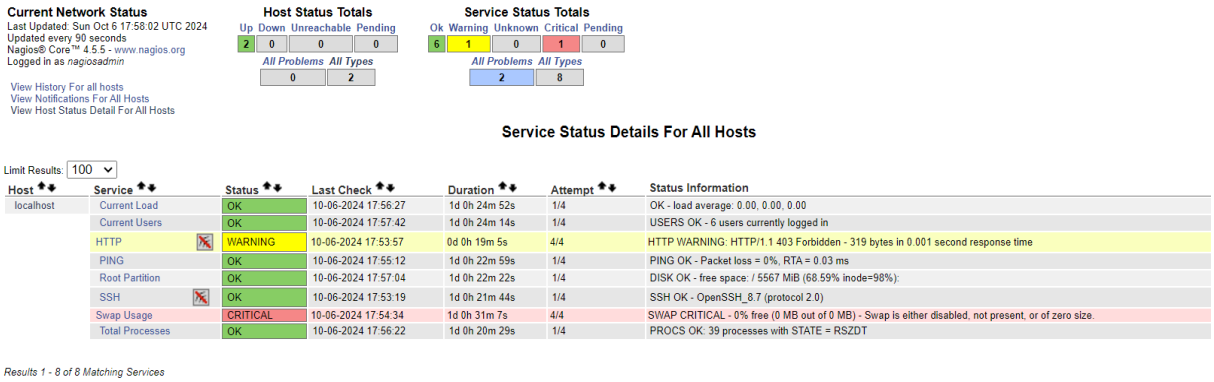
Host Commands

Locate host on map
Disable active checks of this host
Re-schedule the next check of this host
Submit passive check result for this host
Stop accepting passive checks for this host
Stop obsessing over this host
Disable notifications for this host
Send custom host notification
Schedule downtime for this host
Schedule downtime for all services on this host
Disable notifications for all services on this host
Enable notifications for all services on this host
Schedule a check of all services on this host
Disable checks of all services on this host
Enable checks of all services on this host
Disable event handler for this host
Disable flap detection for this host
Clear flapping state for this host

Host Comments

Add a new comment Delete all comments

Entry Time	Author	Comment	Comment ID	Persistent	Type	Expires	Actions
This host has no comments associated with it							



Current Network Status
Last Updated: Sun Oct 6 17:58:02 UTC 2024
Updated every 90 seconds
Nagios® Core™ 4.5.5 - www.nagios.org
Logged in as *nagiosadmin*

View History For all hosts
View Notifications For All Hosts
View Host Status Detail For All Hosts

Host Status Totals

Up	Down	Unreachable	Pending
2	0	0	0

All Problems *All Types*
0 2

Service Status Totals

Ok	Warning	Unknown	Critical	Pending
6	1	0	1	0

All Problems *All Types*
2 8

Service Status Details For All Hosts

Limit Results: 100

Host	Service	Status	Last Check	Duration	Attempt	Status Information
localhost	Current Load	OK	10-06-2024 17:56:27	1d 0h 24m 52s	1/4	OK - load average: 0.00, 0.00, 0.00
	Current Users	OK	10-06-2024 17:57:42	1d 0h 24m 14s	1/4	USERS OK - 6 users currently logged in
	HTTP	WARNING	10-06-2024 17:53:57	0d 0h 19m 5s	4/4	HTTP WARNING: HTTP/1.1 403 Forbidden - 319 bytes in 0.001 second response time
	PING	OK	10-06-2024 17:55:12	1d 0h 22m 59s	1/4	PING OK - Packet loss = 0%, RTA = 0.03 ms
	Root Partition	OK	10-06-2024 17:57:04	1d 0h 22m 22s	1/4	DISK OK - free space: / 5567 MiB (68.59% inode=98%):
	SSH	OK	10-06-2024 17:53:19	1d 0h 21m 44s	1/4	SSH OK - OpenSSH_8.7 (protocol 2.0)
	Swap Usage	CRITICAL	10-06-2024 17:54:34	1d 0h 31m 7s	4/4	SWAP CRITICAL - 0% free (0 MB out of 0 MB) - Swap is either disabled, not present, or of zero size.
	Total Processes	OK	10-06-2024 17:56:22	1d 0h 20m 29s	1/4	PROCS OK: 39 processes with STATE = RSZDT

Results 1 - 8 of 8 Matching Services

As you can see, we have our linuxserver up and running. It is showing critical status on HTTP due to permission errors and swap because there is no partition created.

In this case, we have monitored -

Servers: 1 linux server
Services: swap
Ports: 22, 80 (ssh, http)
Processes: User status, Current load, total processes, root partition, etc.

Recommended Cleanup

- Terminate both of your EC-2 instances to avoid charges.
- Delete the security group if you created a new one (it won't affect your bill, you may avoid it)

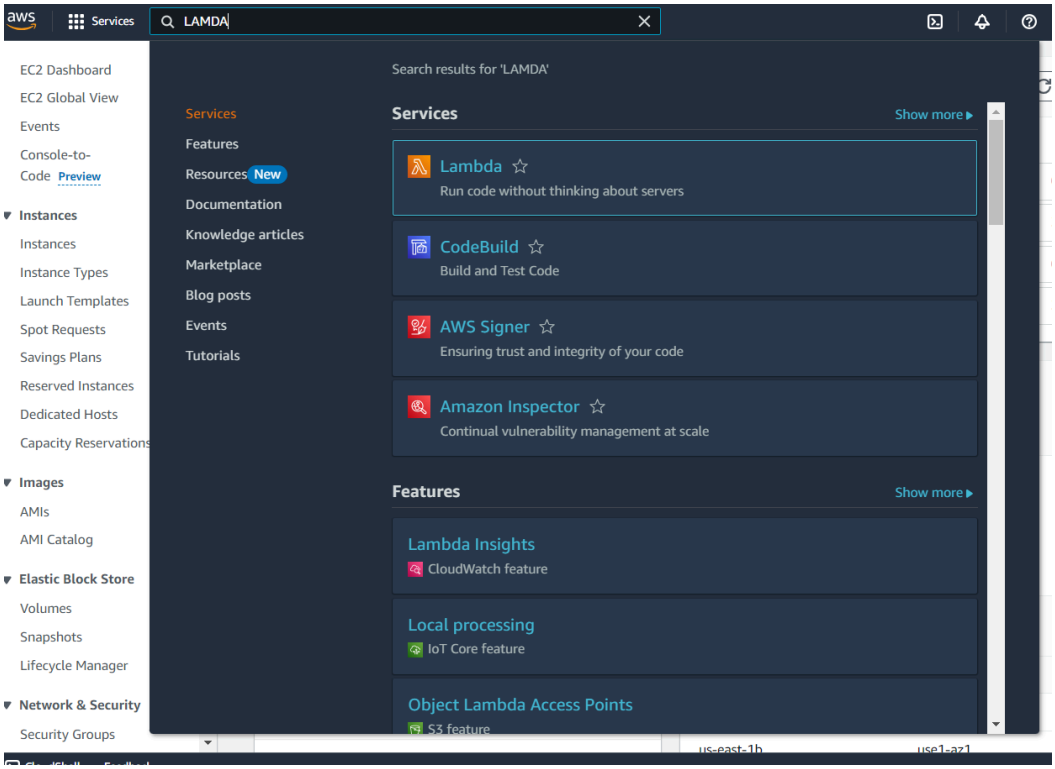
Conclusion: In conclusion, the experiment focused on monitoring ports, services, and a Linux server using Nagios. Through the step-by-step process, we successfully configured Nagios to monitor essential network services on the Linux server. By setting up both the Nagios host and client, we were able to track system performance, ensure service availability, and monitor key metrics like CPU and memory usage.

Name : Nishant S Khetal **Div : D15C** **Roll No : 24**

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

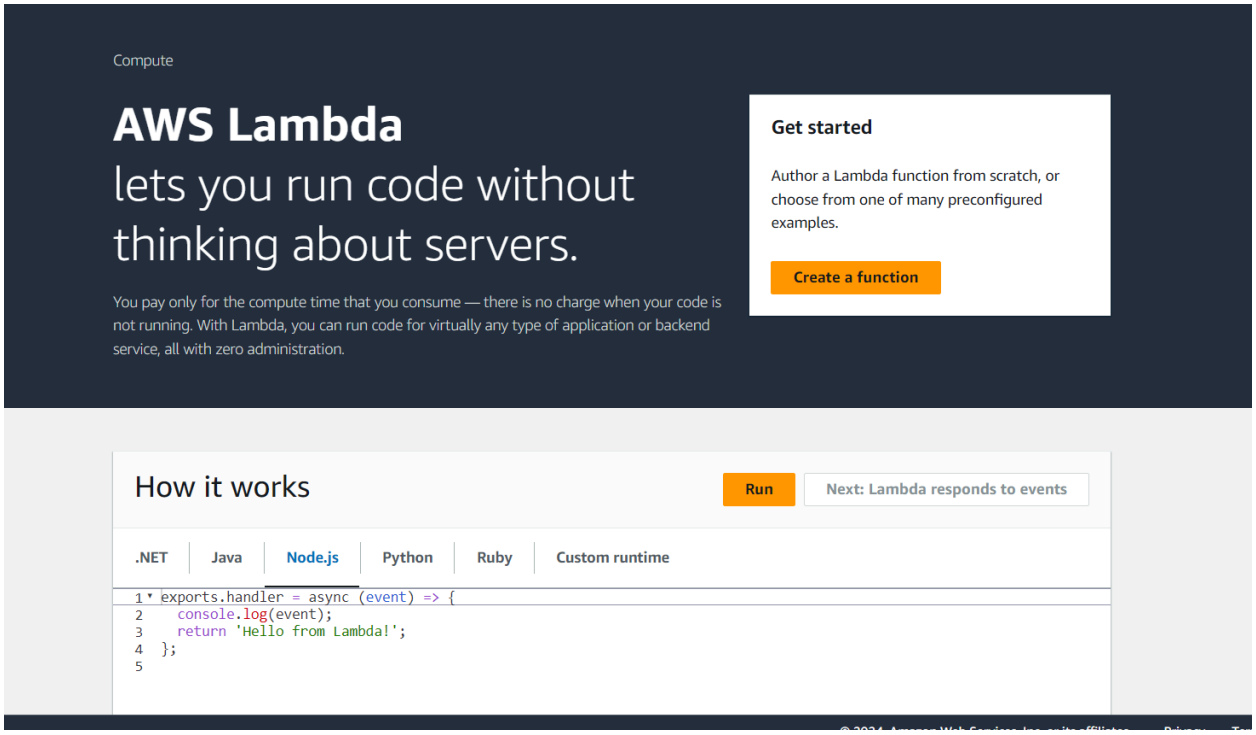
Step 1: Accessing AWS

Log in to your AWS Personal/Academy account. Navigate to the Lambda service by searching for "Lambda" in the AWS Management Console.



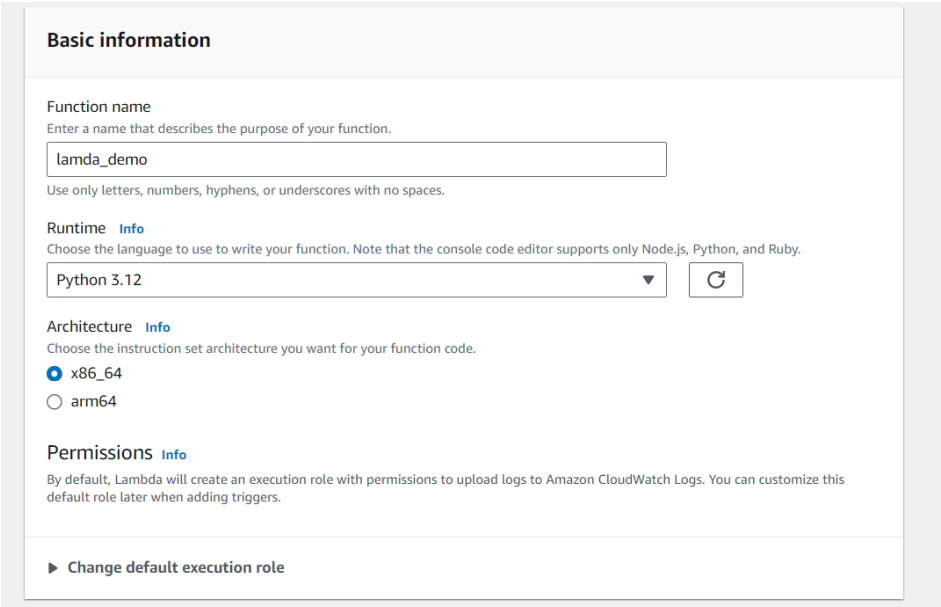
Step 2: Creating a New Lambda Function

Click on the "Create function" button. Provide a name for your Lambda function and select the language you wish to use, such as Python 3.12. For architecture, choose x86, and for execution role, opt to create a new role with basic Lambda g permissions.



Step 3: Configuring Basic Settings

To modify the basic settings, navigate to the "Configuration" tab and click on "Edit" under General Settings. Here, you can add a description and adjust the memory and timeout settings. For this experiment, I set the timeout to 1 second, which is sufficient for testing.



Permissions Info

By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▼ **Change default execution role**

Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.

- (●) Create a new role with basic Lambda permissions
- ( ) Use an existing role
- ( ) Create a new role from AWS policy templates

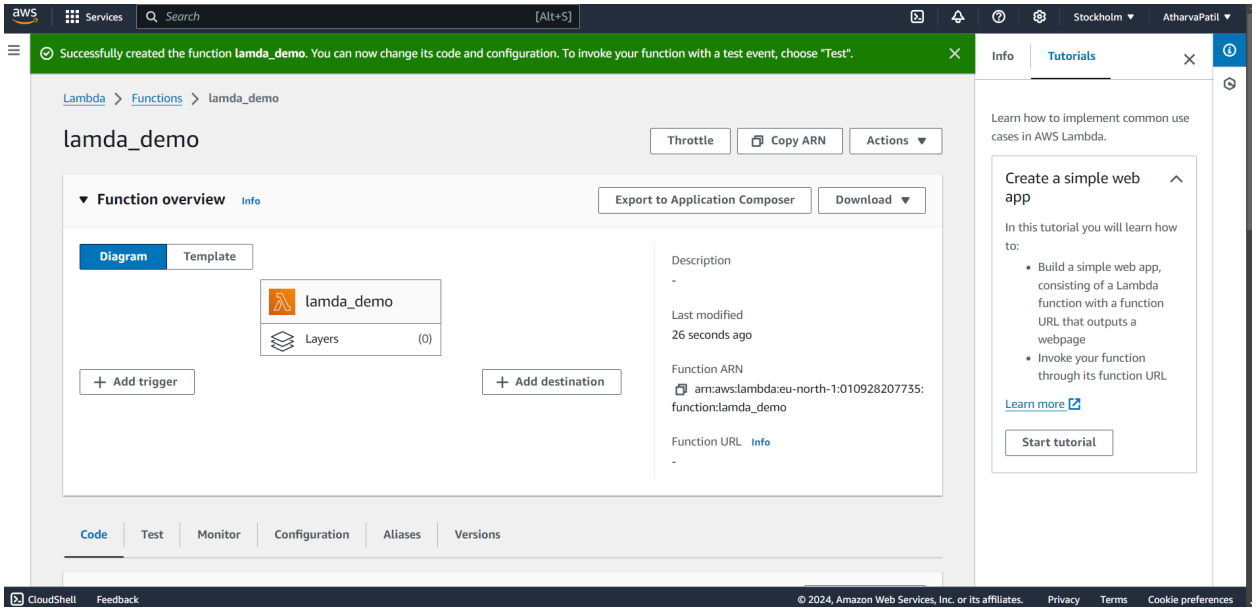
ⓘ Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.

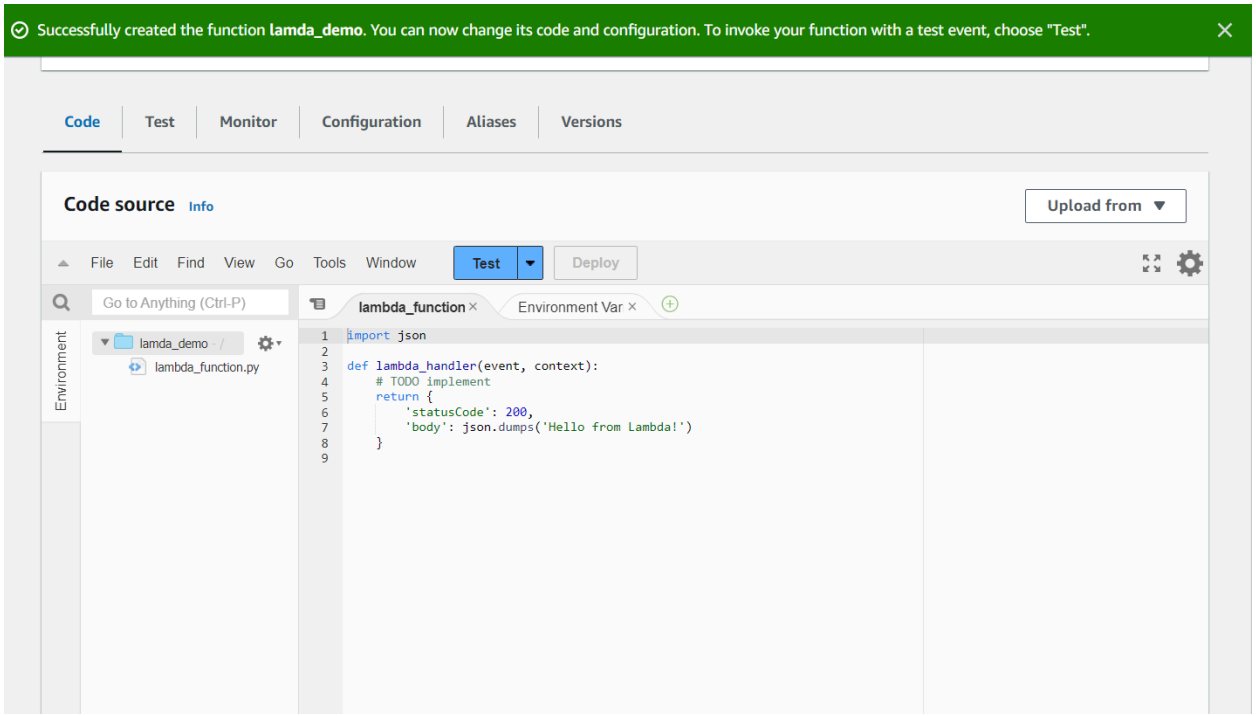
Lambda will create an execution role named ATHARV_LAMDA-role-0u7c9ooi, with permission to upload logs to Amazon CloudWatch Logs.

▶ **Additional Configurations**

Use additional configurations to set up code signing, function URL, tags, and Amazon VPC access for your function.

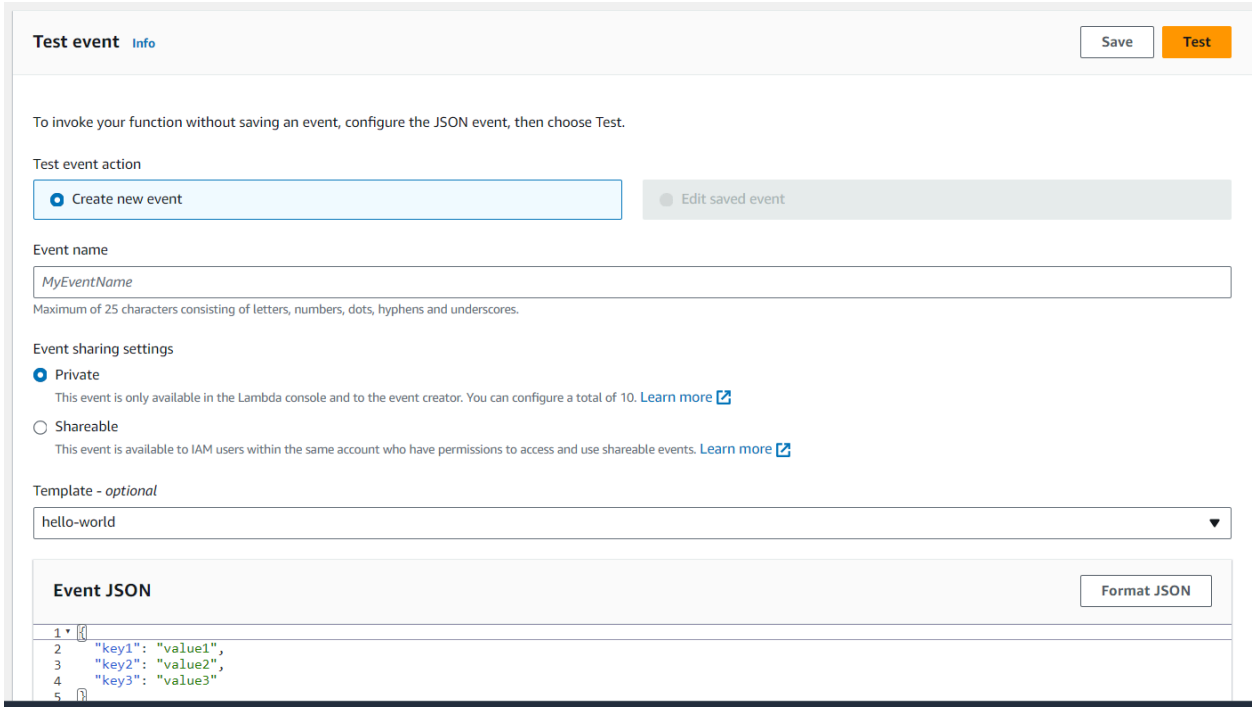
Cancel | **Create function**

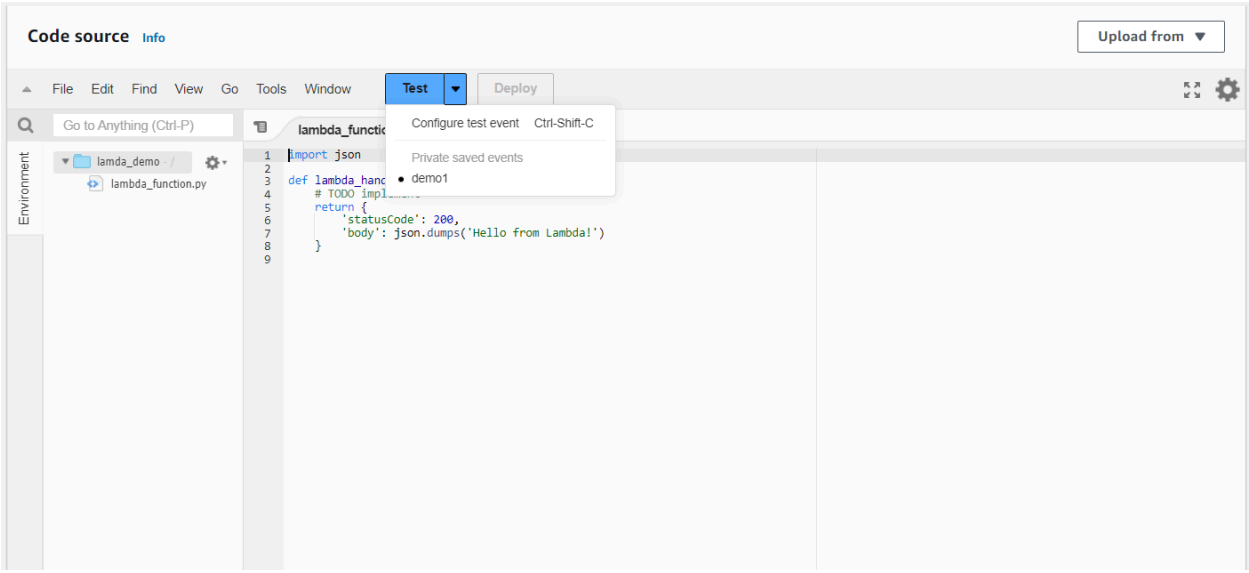


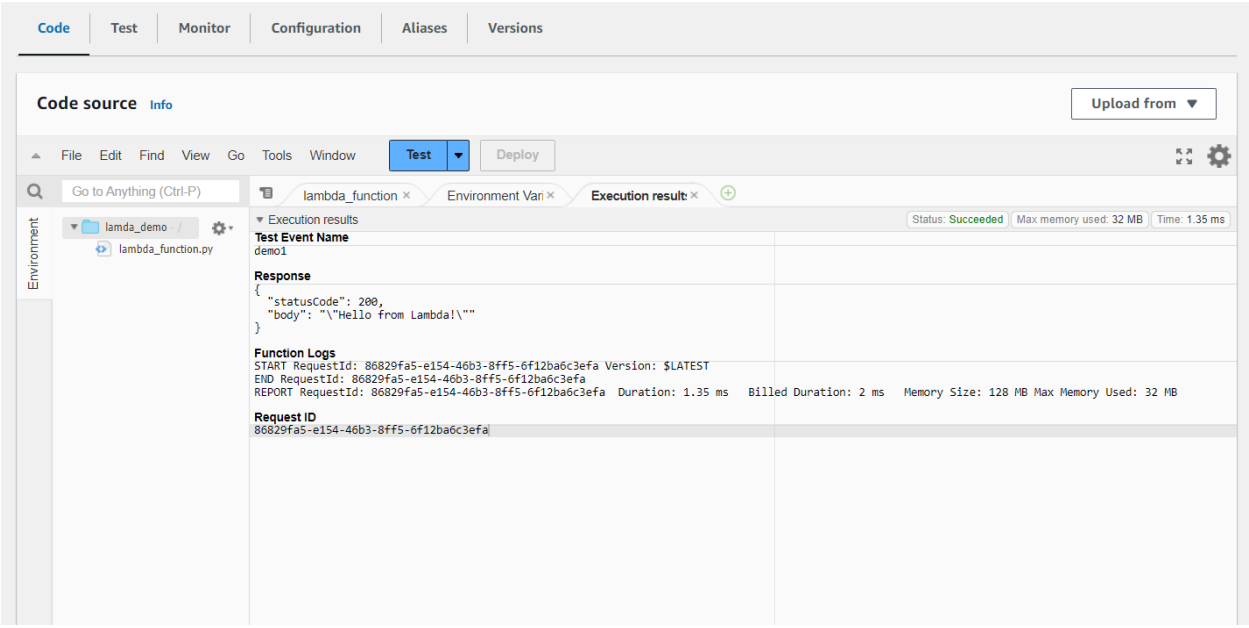


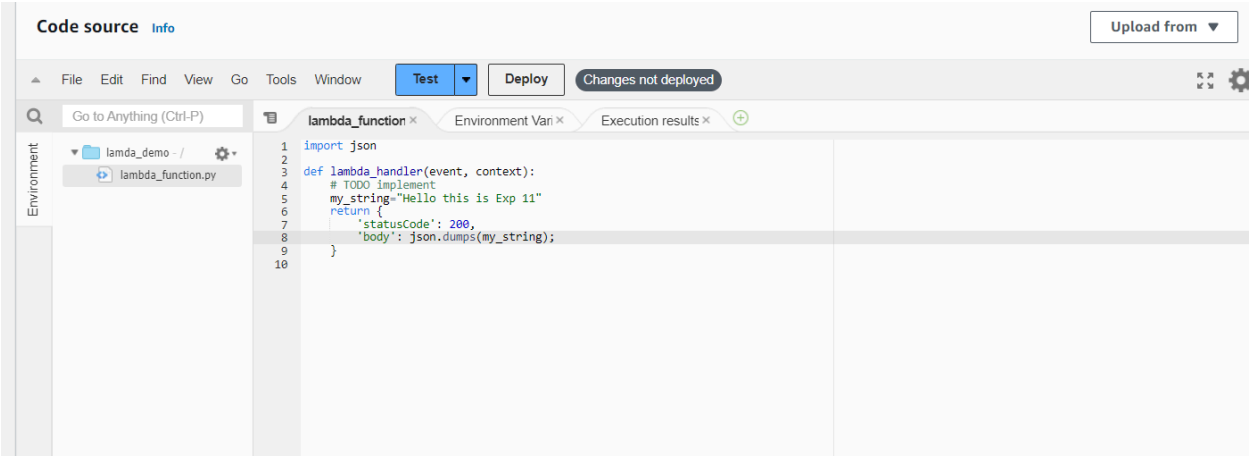
Step 4: Testing the Function

Click on the "Test" tab and select "Create a new event." Name your event, set the event sharing to private, and choose the "hello-world" template.



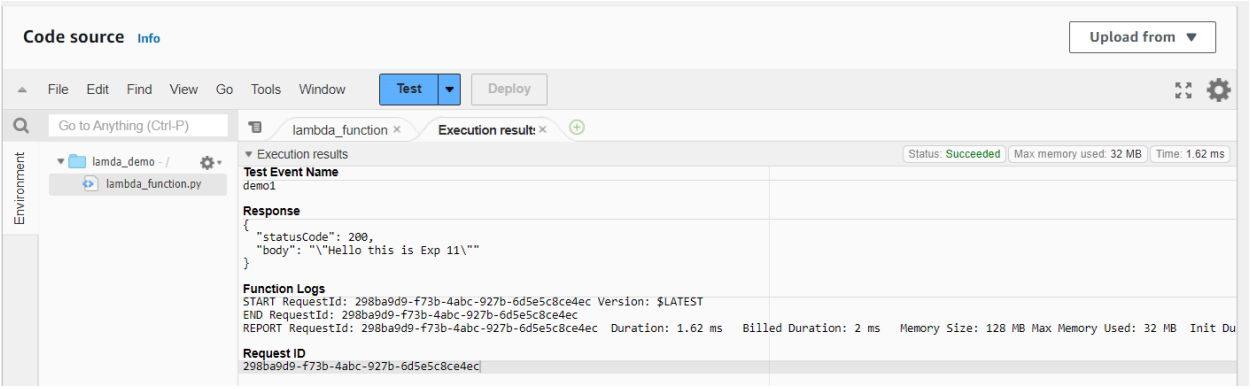

Code source Info
Upload from
File Edit Find View Go Tools Window
Test
Deploy
Configure test event Ctrl-Shift-C
Private saved events
demo1
Go to Anything (Ctrl-P)
lamda_demo
lambda_function.py
Environment
import json
def lambda_hand
# TODO impl
return {
'statusCode': 200,
'body': json.dumps('Hello from Lambda!')
}


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)
lamda_demo
lambda_function.py
Environment
lambda_function
Environment Vari
Execution result:
Execution results
Status: Succeeded
Max memory used: 32 MB
Time: 1.35 ms
Test Event Name
demo1
Response
{
"statusCode": 200,
"body": "\"Hello from Lambda!\""
}
Function Logs
START RequestId: 86829fa5-e154-46b3-8ff5-6f12ba6c3efa Version: $LATEST
END RequestId: 86829fa5-e154-46b3-8ff5-6f12ba6c3efa
REPORT RequestId: 86829fa5-e154-46b3-8ff5-6f12ba6c3efa Duration: 1.35 ms Billed Duration: 2 ms Memory Size: 128 MB Max Memory Used: 32 MB
Request ID
86829fa5-e154-46b3-8ff5-6f12ba6c3efa



Step 5: Running the Test

In the Code section, select the newly created event from the dropdown menu and click on "Test." You should see the output displayed below.

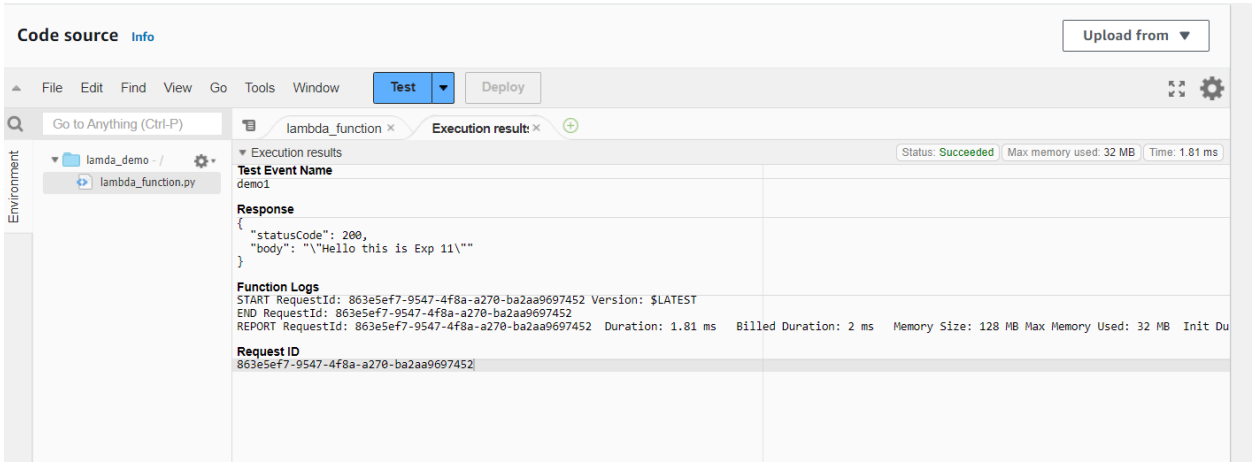


Step 6: Editing and Deploying the Code

You can modify your Lambda function's code as needed. I updated the code to display a new string. After making changes, press `Ctrl + S` to save and then click on "Deploy" to apply the updates.

Step 7: Final Testing

Return to the "Test" tab and execute the test again to observe the output. You should see a status code of 200 along with your string output and function logs confirming a successful deployment.



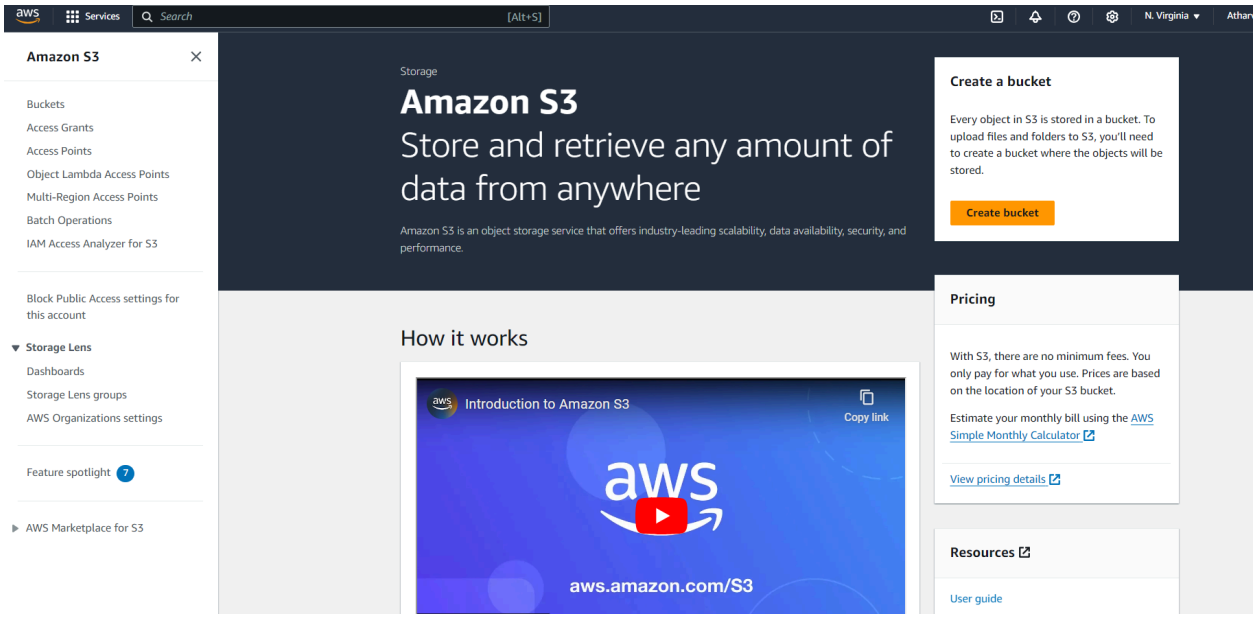
Conclusion:

In this experiment, you created and tested your first AWS Lambda function using Python. You learned to navigate the AWS Management Console, configure basic settings, and modify your function's code. This experience highlights the ease of deploying serverless applications with Lambda, allowing you to focus on coding rather than infrastructure management. You now have a foundational understanding to explore more complex serverless solutions and integrate AWS services for greater functionality.

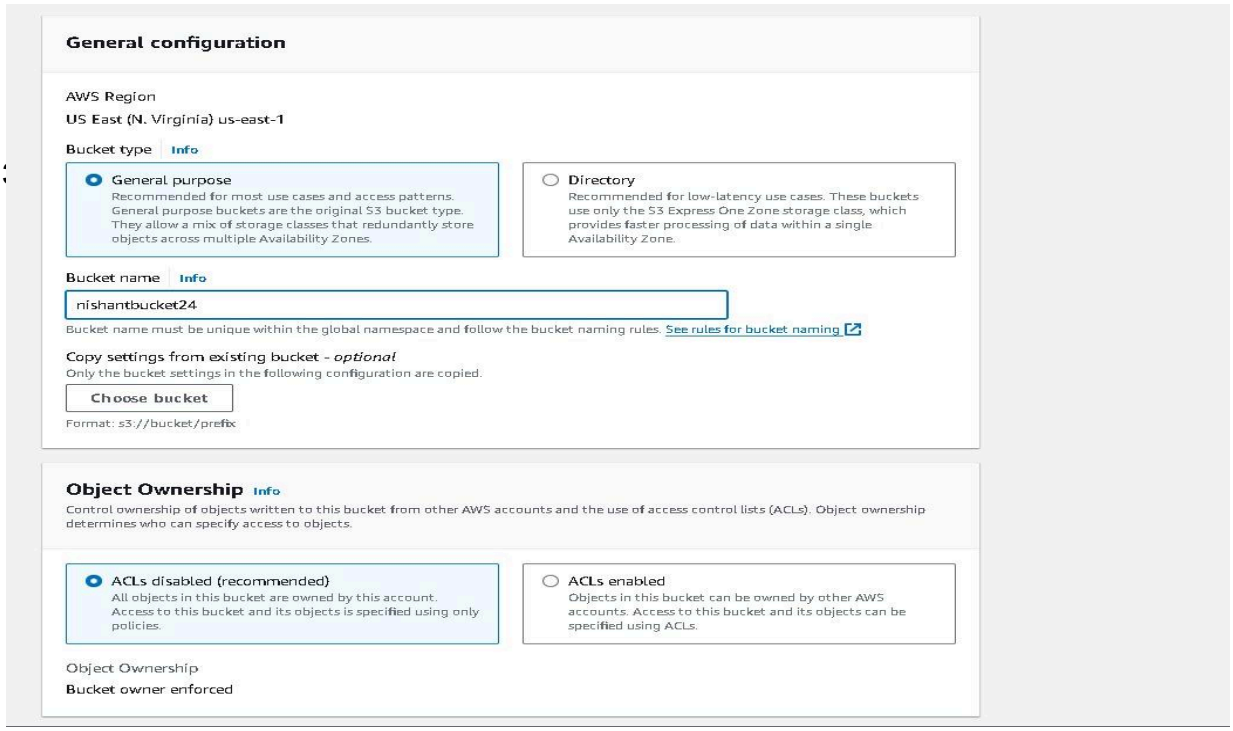
Name : Nishant S Khetal **Class : D15C** **Roll No : 24**

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.

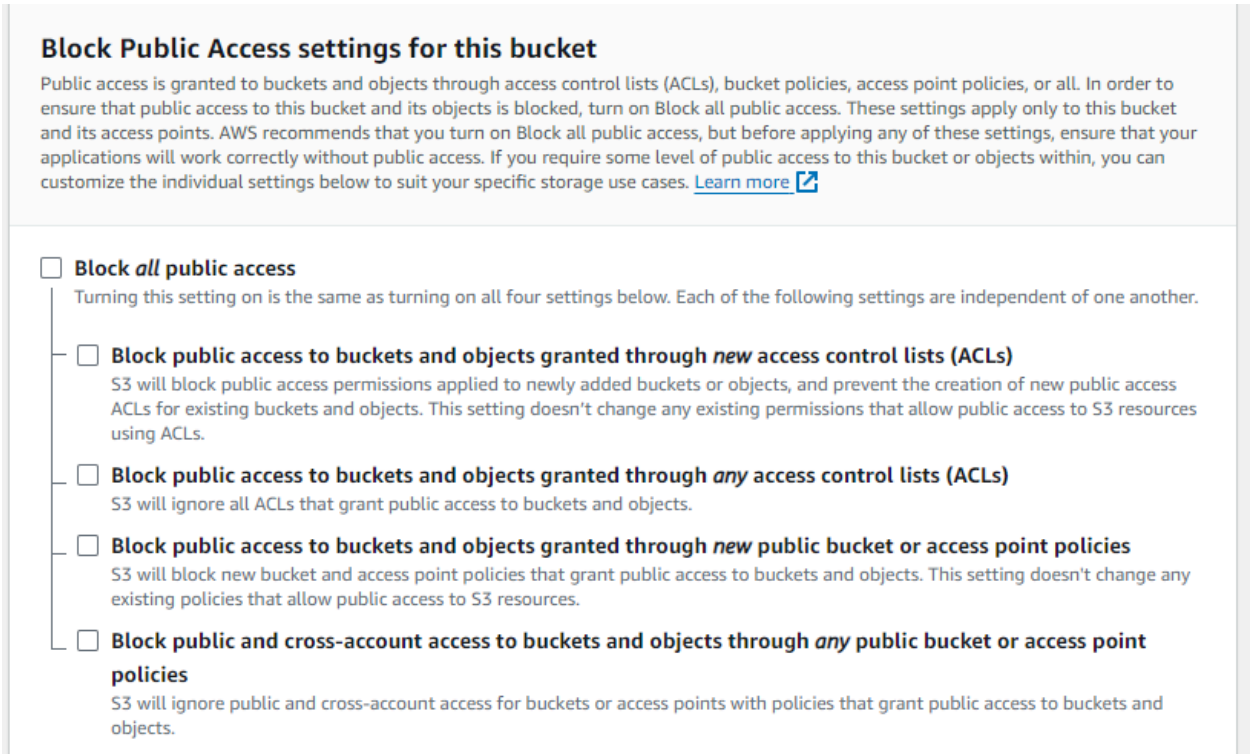
Step 1: Create a s3 bucket. 1) Search for S3 bucket in the services search. Then click on create bucket.



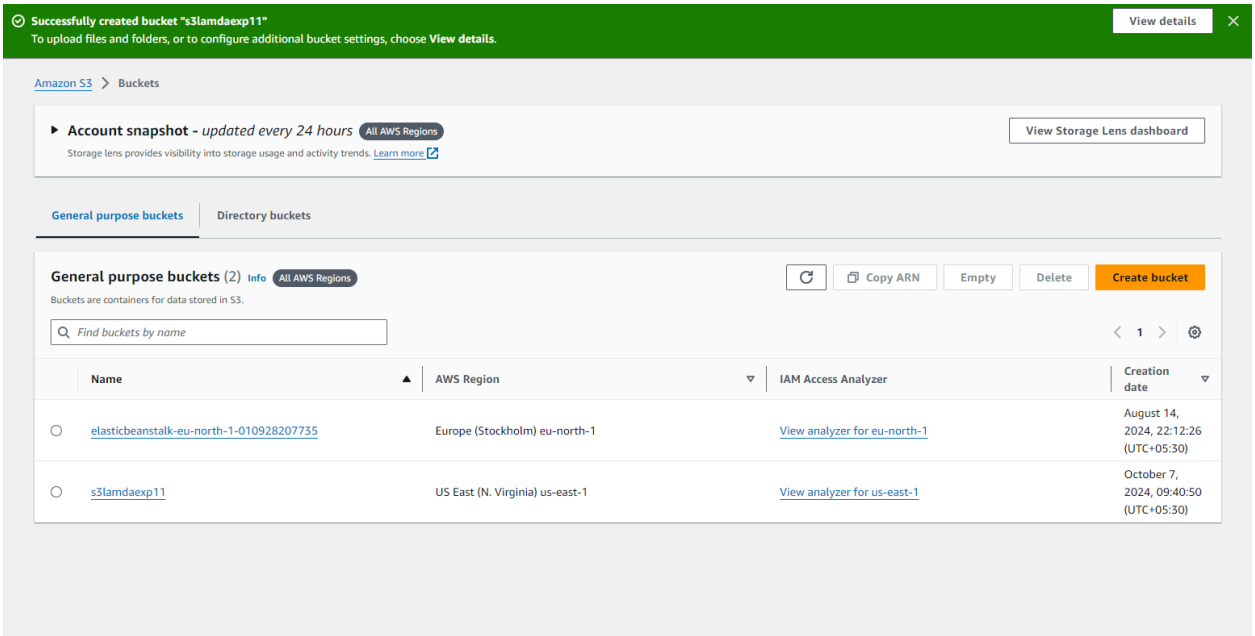
2) Keep the bucket as a general purpose bucket. Give a name to your bucket.



Name : Nishant S Khetal **Class : D15C** **Roll No : 24**

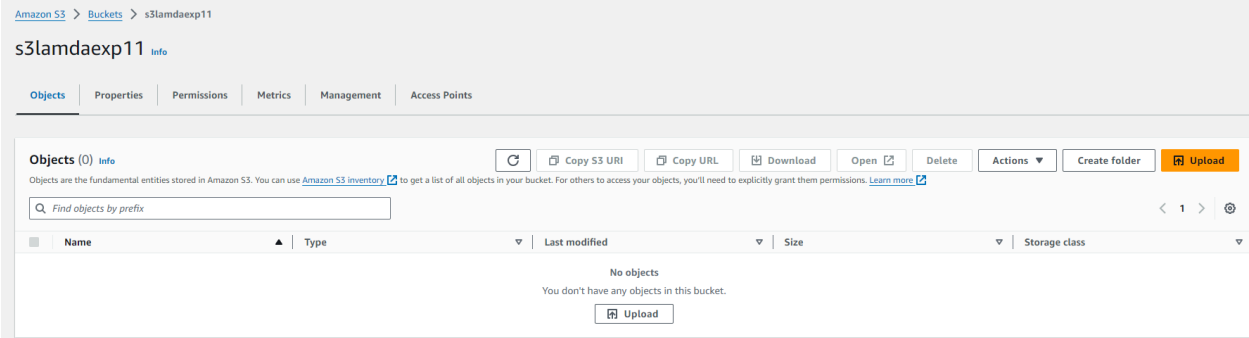


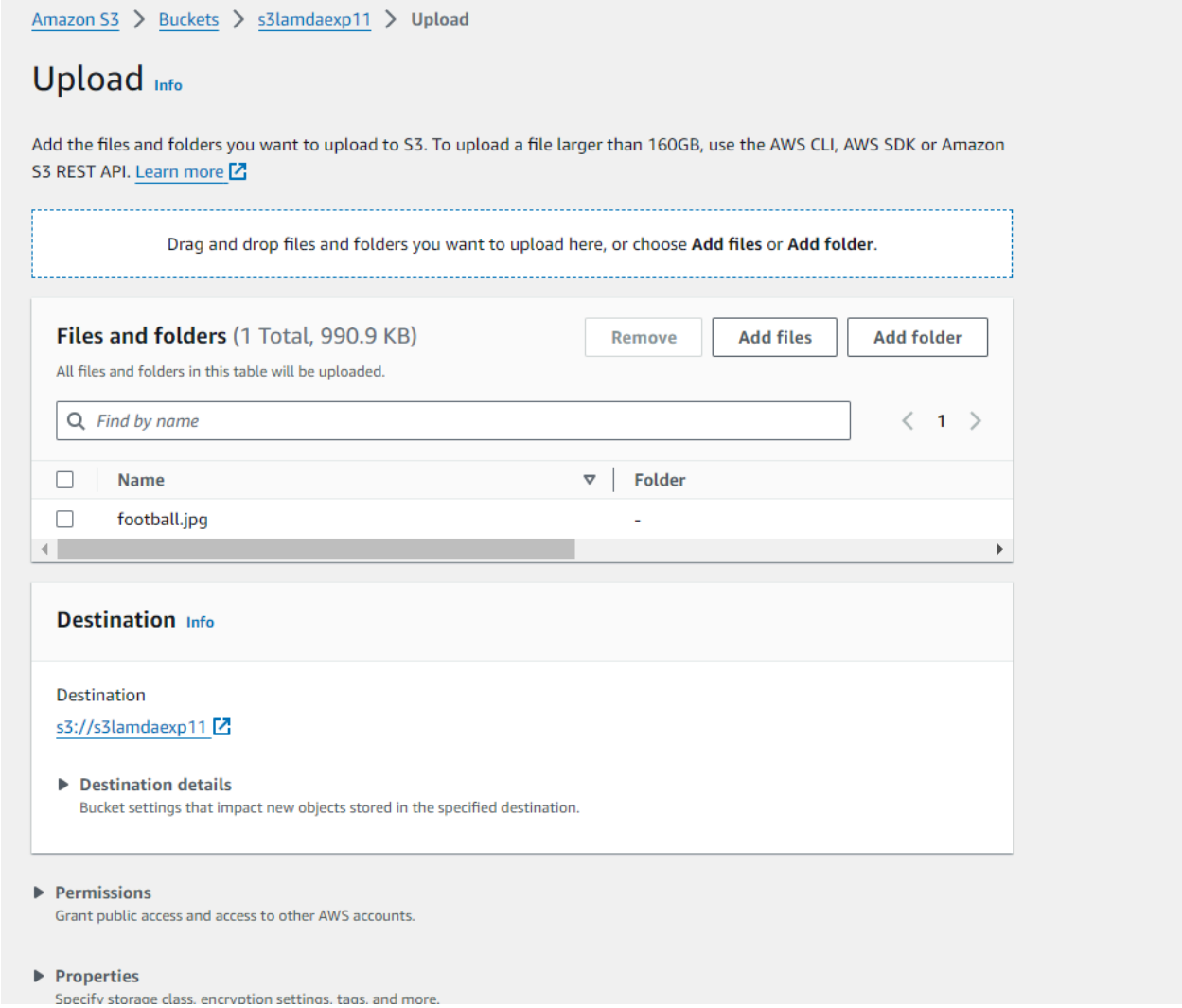
4) Keeping all other options the same, click on create. This would create your bucket. Now click on the name of the bucket.



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

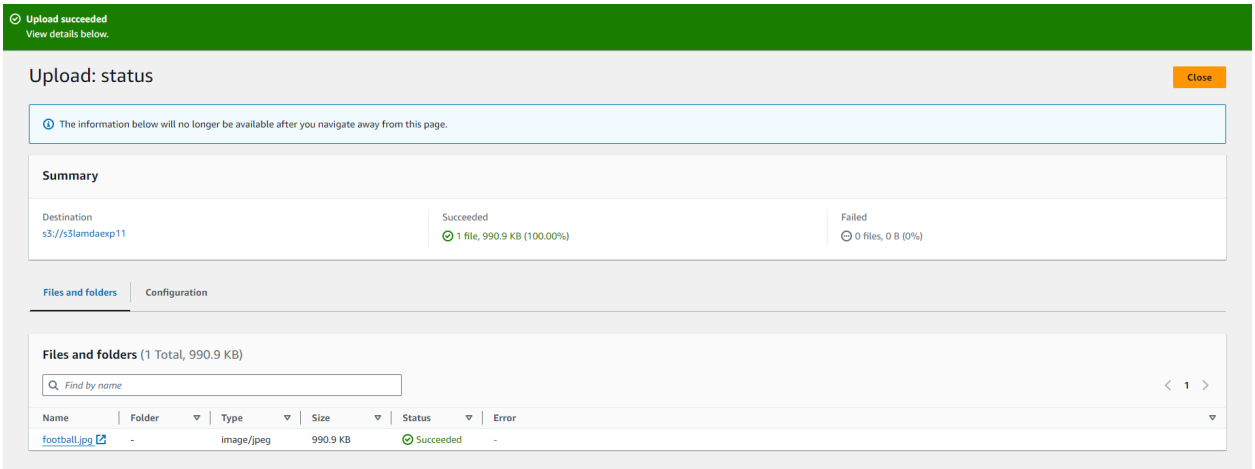
Name : Nishant S Khetal **Class : D15C** **Roll No : 24**





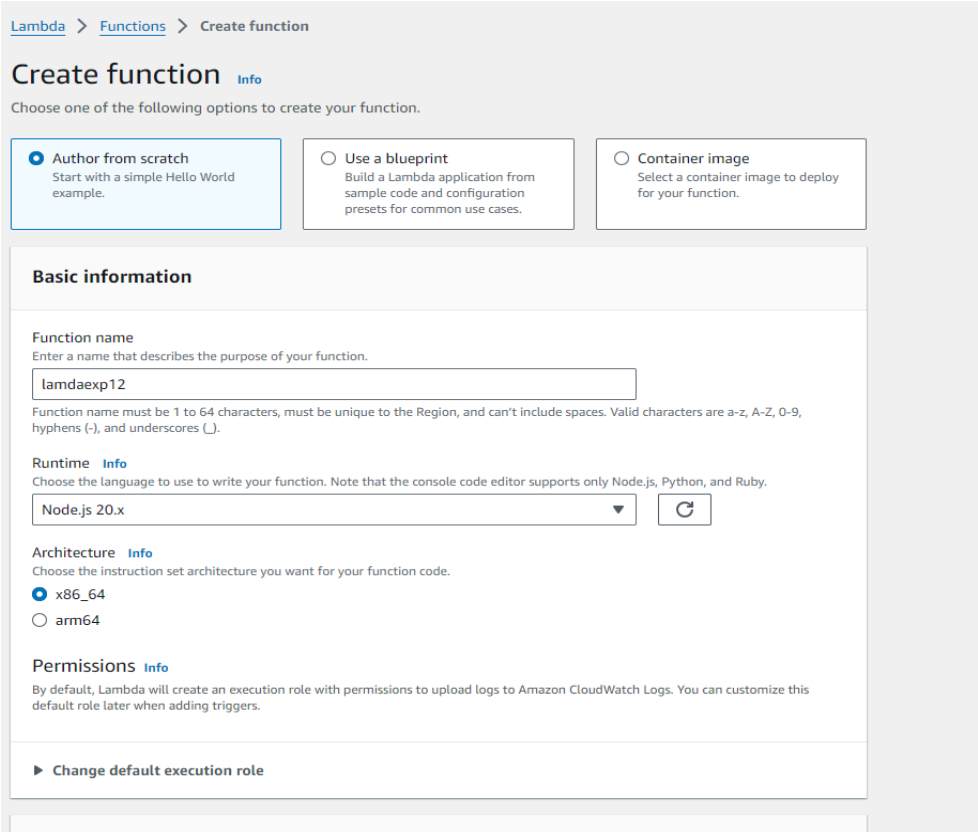
6) The image has been uploaded to the bucket.

Name : Nishant S Khetal **Class : D15C** **Roll No : 24**



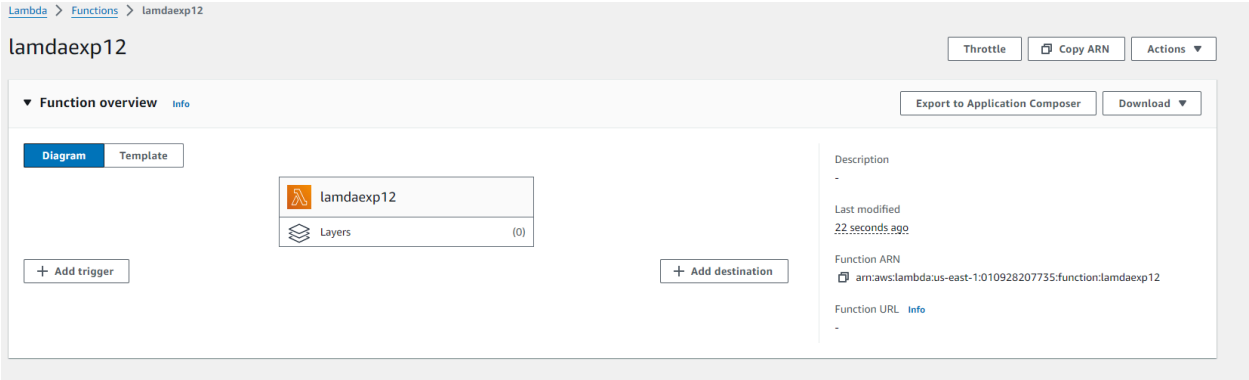
Step 2: Configure Lambda function

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

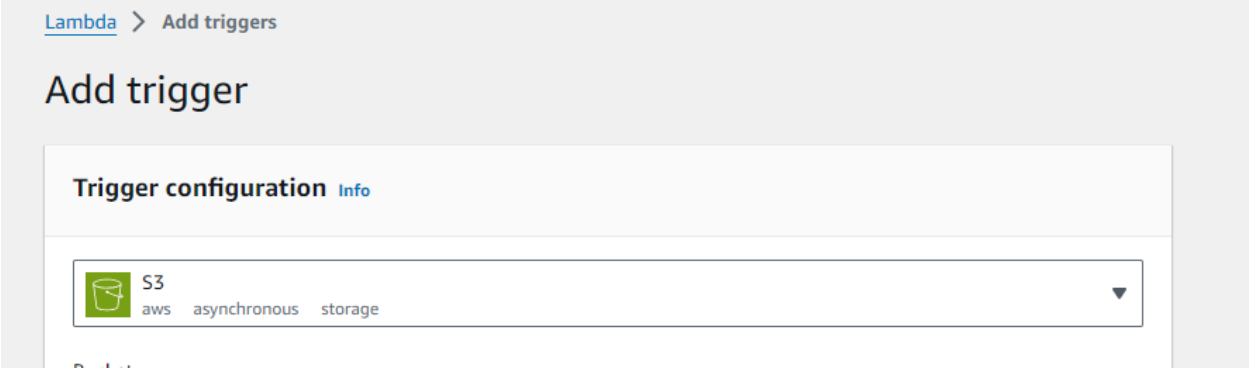


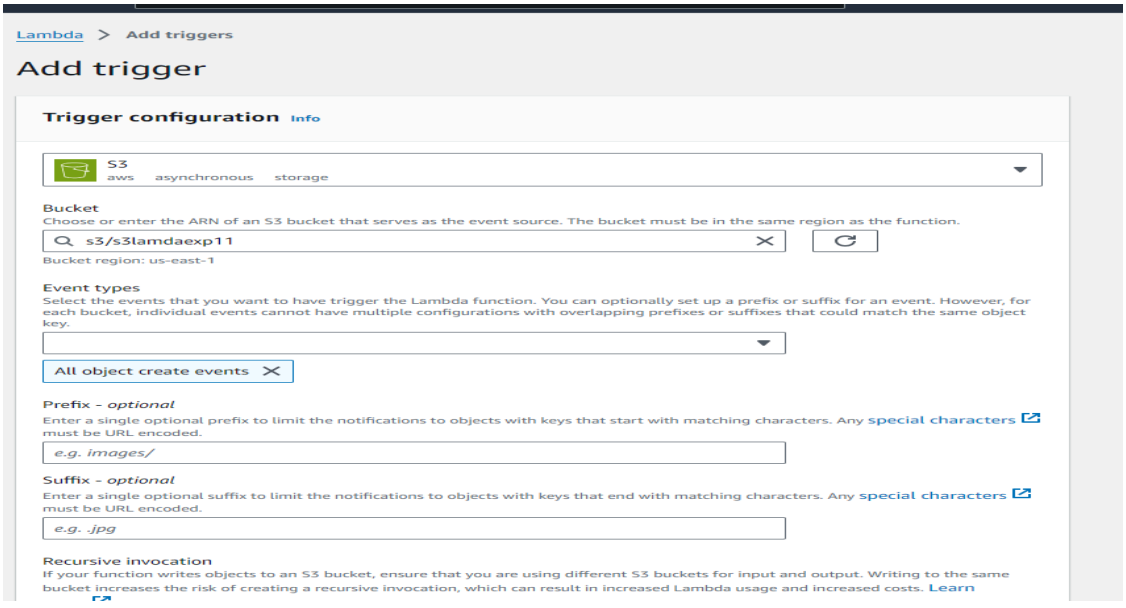
2) Under trigger configuration, search for S3 and select it.

Name : Nishant S Khetal **Class : D15C** **Roll No : 24**

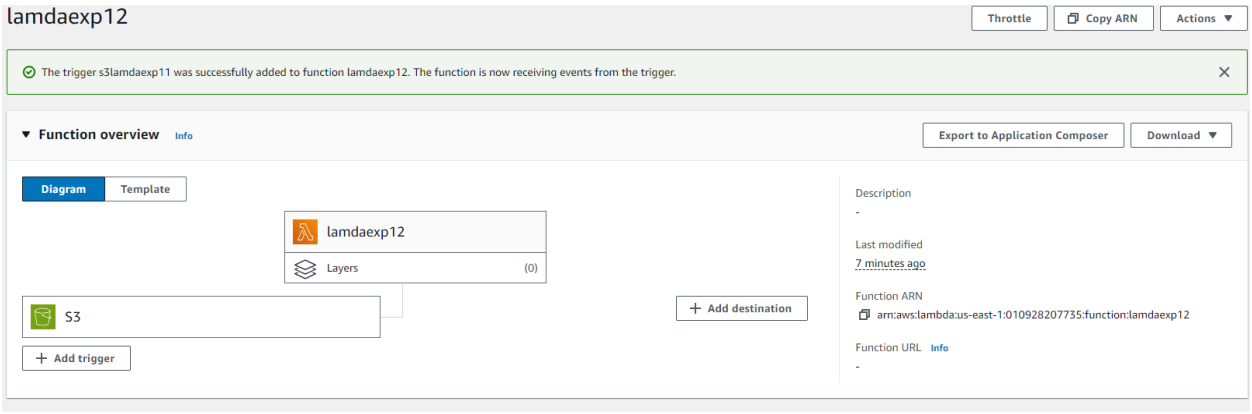


3) Here, select the S3 bucket you created for this experiment. Acknowledge the condition given by AWS. then click on Add. This will add the S3 bucket trigger to your function





Name : Nishant S Khetal **Class : D15C** **Roll No : 24**

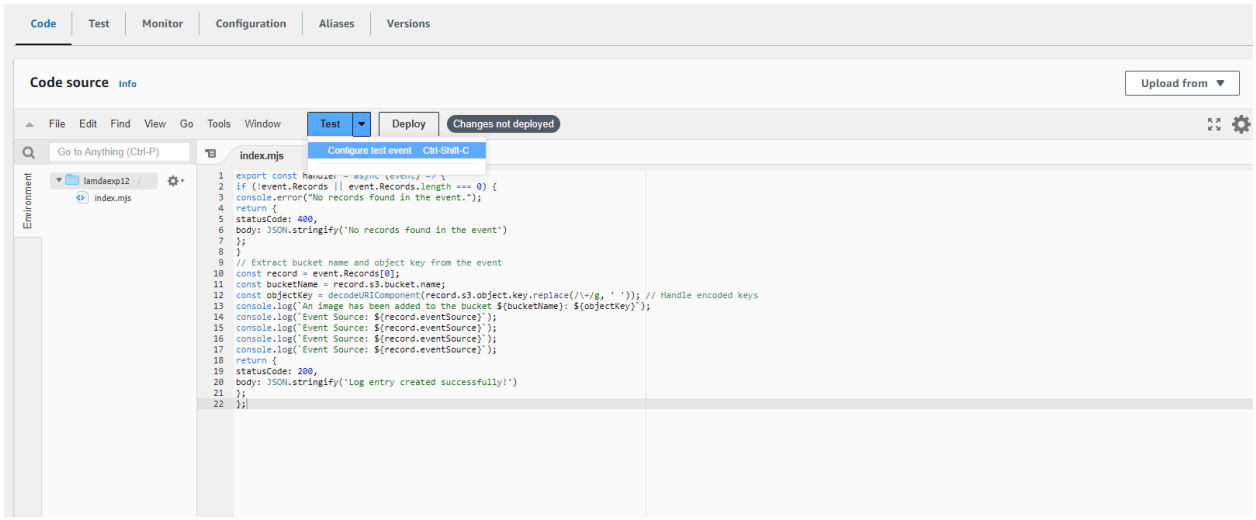


4) Scroll down to the code section of the function. Add the following javascript code to the code area by replacing the existing code

```
export const handler = async (event) => {
if (!event.Records || event.Records.length === 0) {
console.error("No records found in the event.");
return {
statusCode: 400,
body: JSON.stringify('No records found in the event')
};
}
// Extract bucket name and object key from the event
const record = event.Records[0];
const bucketName = record.s3.bucket.name;
const objectKey = decodeURIComponent(record.s3.object.key.replace(/\+/g, ' ')); // Handle encoded keys
console.log(`An image has been added to the bucket ${bucketName}: ${objectKey}`);
console.log(`Event Source: ${record.eventSource}`);
console.log(`Event Source: ${record.eventSource}`);
console.log(`Event Source: ${record.eventSource}`);
console.log(`Event Source: ${record.eventSource}`);
return {
statusCode: 200,
body: JSON.stringify('Log entry created successfully!')
};
};
```

Name : Nishant S Khetal **Class : D15C** **Roll No : 24**

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



Name : Nishant S Khetal **Class : D15C** **Roll No : 24**

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

Event JSON Format JSON

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

1:1 JSON Spaces: 2

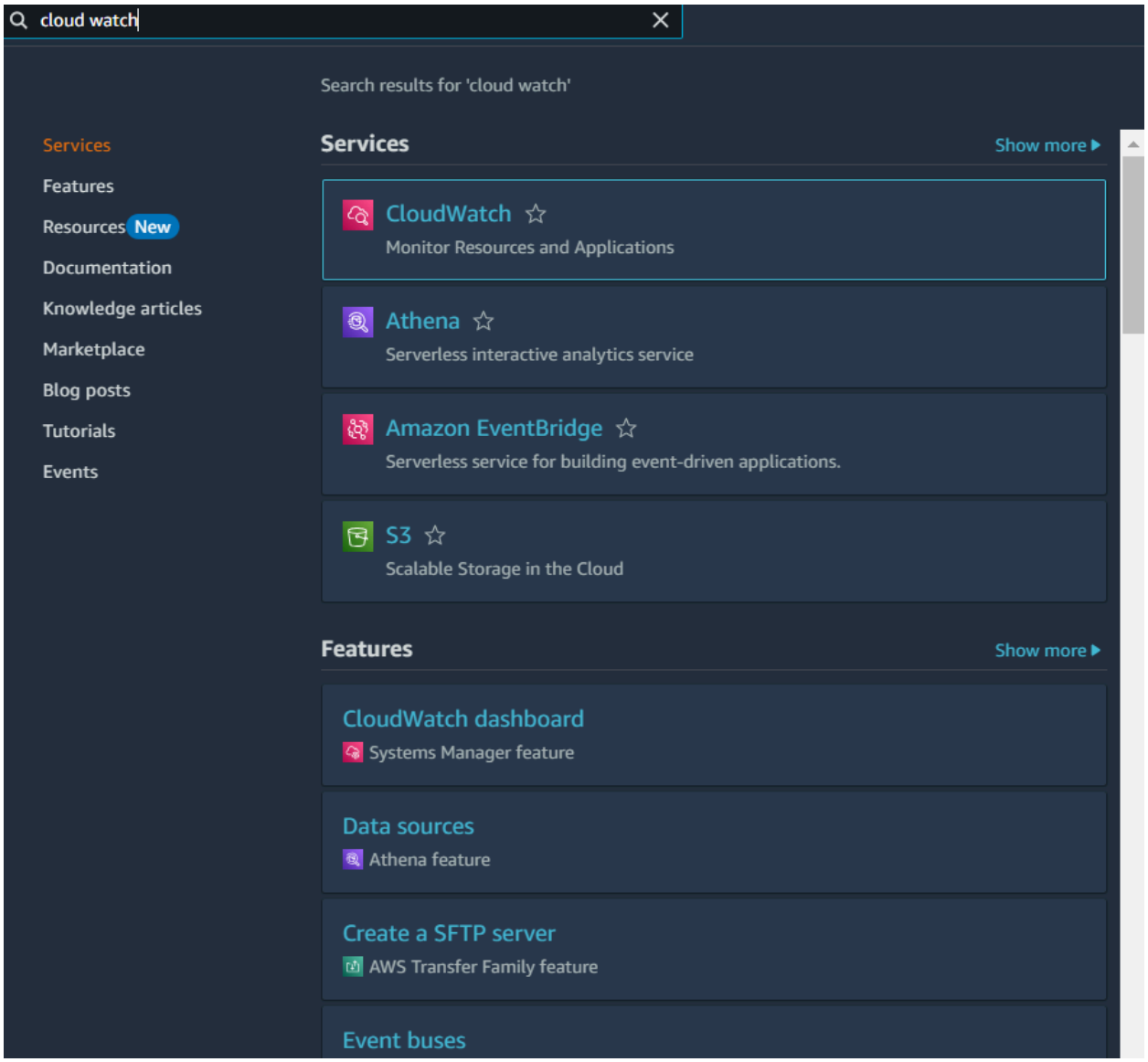
Cancel Invoke Save

The test event **myevent1** was successfully saved.

Name : Nishant S Khetal **Class : D15C** **Roll No : 24**

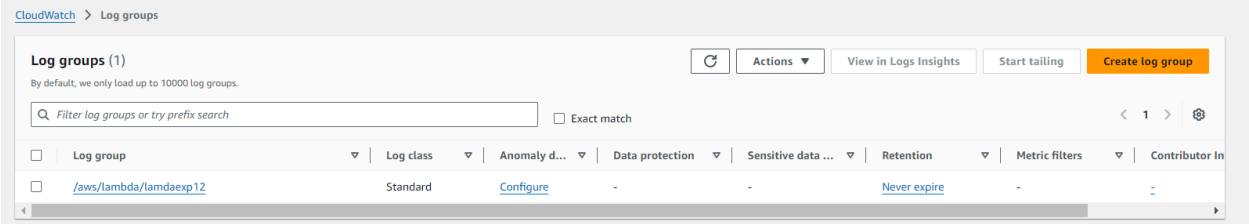
Step 3: Check the logs

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

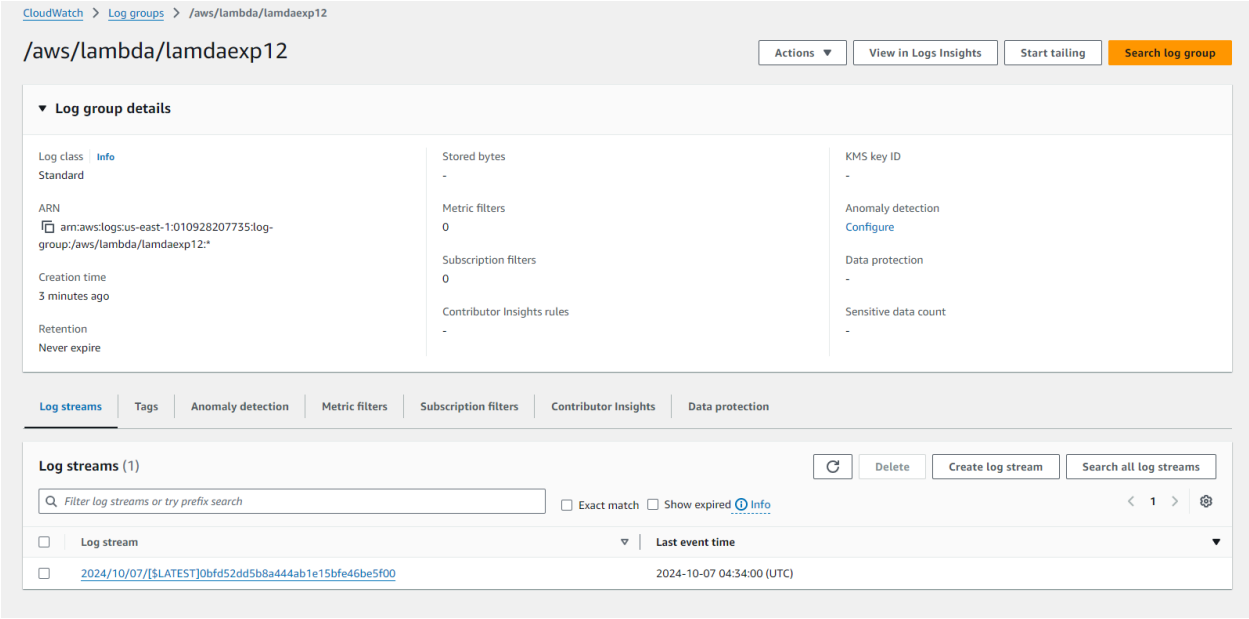


2) Here, Click on Logs → Log Groups. Select the log that has the lambda function name you just ran.

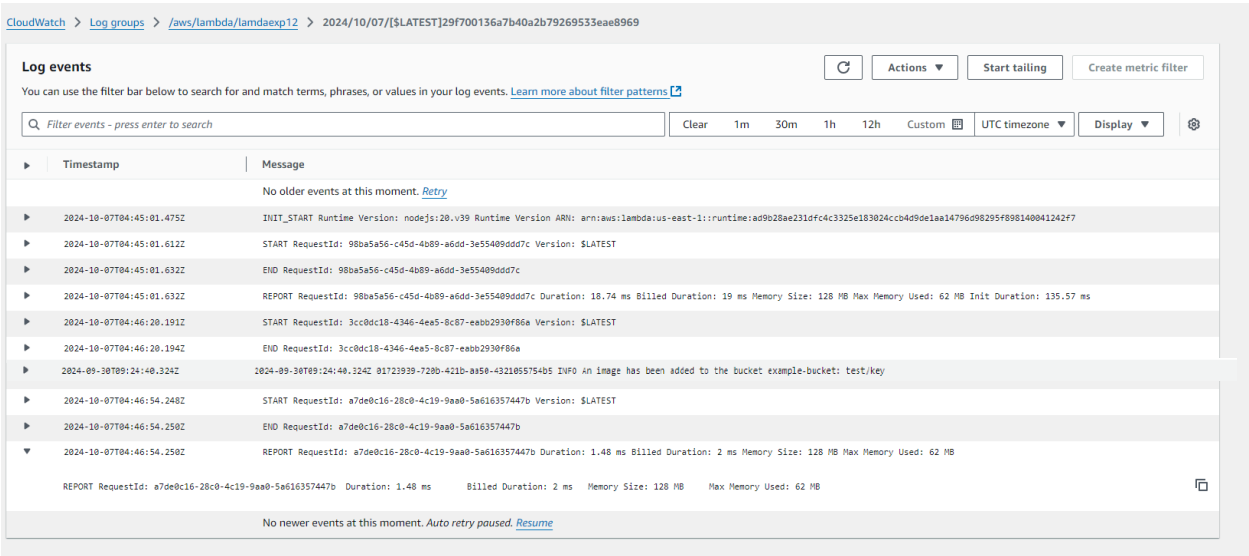
Name : Nishant S Khetal **Class : D15C** **Roll No : 24**



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



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



Name : Nishant S Khetal **Class : D15C** **Roll No : 24**

Conclusion:

In this experiment, In addition to demonstrating the integration of AWS Lambda with S3, this experiment showcases the scalability and flexibility of serverless architectures. By leveraging these services, we can build applications that respond in real-time to changes in data, such as the addition of files to S3 buckets, without the need for managing underlying server infrastructure. This not only enhances efficiency but also reduces operational costs, allowing developers to focus on building features rather than maintaining systems. Furthermore, the ability to log and monitor events through CloudWatch opens opportunities for further automation and analytics, paving the way for more complex workflows and data processing solutions.